

REPUBLIC OF TURKEY
FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND
APPLIED SCIENCE



AN ANDROID BASED RECEIPT TRACKER SYSTEM
USING OPTICAL CHARACTER RECOGNITION

KAREZ ABDULWAHHAB HAMAD

Master Thesis
Department: Software Engineering
Supervisor: Asst. Prof. Dr. Mehmet KAYA

JULY – 2017

REPUBLIC OF TURKEY
FIRAT UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

**AN ANDROID BASED RECEIPT TRACKER SYSTEM USING
OPTICAL CHARACTER RECOGNITION**

MASTER THESIS
KAREZ ABDULWAHHAB HAMAD
(Student No: 151137113)

Department of Software Engineering
Submission Date to the Institute: June 7, 2017
Thesis Presentation Date: July 4, 2017

Thesis Supervisor: Asst. Prof. Dr. Mehmet KAYA (ADYU)

Jury member: Prof. Dr. Asaf VAROL (FU)

Jury member: Assoc. Prof. Dr. Cihan VAROL (SHSU, USA)



July-2017

ACKNOWLEDGEMENTS

First, thanks to **ALLAH**, the Almighty, for granting me the will and strength, with which this master thesis was accomplished; it will be the first step to propose much more great scientific researches.

I would like to acknowledge my thankfulness and appreciation to my supervisor Asst. Prof. Dr. Mehmet KAYA for his guidance, assistance encouragement, wisdom suggestions, and valuable advice that made the completion of the present master thesis possible.

Last but not the least; I want to express my special thankfulness to my lovely parents, and special gratitude to all members of my family and friends.

Special thanks to my lovely uncle Assoc. Prof. Dr. Yadgar Rasool, who helped me and encouraged me a lot during my study.

TABLE OF CONTENTS

	<u>Page No</u>
ACKNOWLEDGEMENTS	II
TABLE OF CONTENTS	III
ABSTRACT	VI
ÖZET	VII
LIST OF FIGURES.....	VIII
LIST OF TABLES.....	XI
LIST OF ABBREVIATIONS	XII
1. INTRODUCTION	1
1.1. Background.....	1
1.2. Problems Statement	5
1.3. General Aims and Objectives	5
1.4. Thesis Layout.....	7
2. THEORETICAL TECHNIQUES AND BACKGROUND OF OCR	9
2.1. OCR Challenges	9
2.1.1. Complexity of scene.....	9
2.1.2. Uneven lighting problem.....	10
2.1.3. Skewness problem.....	11
2.1.4. Un-focus and deterioration.....	13
2.1.5. Aspect ratios.....	13
2.1.6. Tilting problem	14
2.1.7. Fonts.....	15
2.1.8. Multilingual environments	15
2.1.9. Warping problem	16
2.2. OCR Applications.....	17
2.2.1. Hand-writing recognition applications	17
2.2.2. Healthcare applications	17
2.2.3. Financial tracking applications.....	17
2.2.4. Legal industry	18
2.2.5. Banking application	18
2.2.6. Captcha breaking application.....	18

2.2.7. Automatic number plate recognition application (ANPR)	19
2.3. OCR Phases	19
2.3.1. Image pre-processing phase	19
2.3.2. Segmentation phase.....	24
2.3.3. Normalization phase	26
2.3.4. Feature extraction phase.....	26
2.3.5. Classification phase.....	27
2.3.6. Post-processing phase	29
2.4. OCR Engines	29
2.4.1. GOCR engine.....	29
2.4.2. Ocrad engine	30
2.4.3. OCRopus.....	30
2.4.4. Tesseract OCR engine.....	31
3. PROPOSED TECHNIQUES	38
3.1. System Overview	38
3.1.1. Receipt region detection.....	40
3.1.2. Receipt image pre-processing phase	43
3.1.3. Recognition phase	51
3.1.4. Regular expression (Regex) phase	60
3.1.5. Database phase.....	62
3.2. Implementation and Practical Work	62
3.3. System Screenshots	68
4. QUERIES AND EXPERIMENTAL RESULTS.....	72
4.1. User Queries	72
4.1.1. Spend analyzer	72
4.1.2. Receipt image discovering	74
4.1.3. Total money expended	75
4.1.4. Total money expended for a particular item.....	77
4.2. Experimental Outcomes.....	78
4.2.1. Capability metrics	79
4.2.2. Examination corpus.....	80
4.2.3. Fake receipt font experimental outcomes.....	81

4.2.4. Merchant copy font experimental outcomes	90
4.2.5. Evaluation of outcomes experienced.....	99
5. CONCLUSION AND FUTURE WORKS.....	102
6. REFERENCES	104
CURRICULUM VITA	111



ABSTRACT

AN ANDROID BASED RECEIPT TRACKER SYSTEM USING OPTICAL CHARACTER RECOGNITION

Since demands for innovating and implementing mobile apps gets deeper, therefore innovations on designing and creating desktop OCR Apps moved and shifted to propose and innovate mobile OCR Apps. Optical Character Recognition (OCR) is the technology that converts the text from handwritten images, text printed images or scanned images to the alterable text for further analysis and process. In this research, we suggested an Android OCR Application for automatically extracting and recognizing text on the receipt images. This research presented the main and powerful techniques proposed for better performing OCR technology on the receipt images acquired through cameras of hand-held devices to obtain and reaching a powerful and efficient system for tracking daily marketing receipts easily. Of course, receipt images have their specifics, therefore OCR applications must be trained for such kind of images else OCR technology cannot perform well-recognition. Unusual text fonts, very small font size, also compressed characters, words and lines on receipt images are the most different characteristics of receipt images from other documents. The main aim or purpose of this research is to find and investigate whether OCR technology is feasible for an Android application to recognize text on receipt images or not. In the recognition stage, for extracting and recognizing text on receipt images, we utilized Tesseract OCR engine which is an open source OCR engine. We proved and showed that instantly submitting receipt images to the Tesseract without applying various techniques suggested in this research will produce useless and bad outcomes which are 58.06% as the percentage of word accuracy and 84.14% as the percentage of character accuracy. But with utilizing all the suggested techniques for two different fonts, the suggested Android application yielded 88.72 % as the percentage of word accuracy, 96.61 % as the percentage of character accuracy and 6.56 sec as the time performance of the suggested Android application.

Keywords: Receipt tracker system, OCR technology, Android Application, Tesseract OCR engine.

ÖZET

OPTİK KARAKTER ALGILAMAYA DAYALI ANDROİD TABANLI FATURA TAKİP SİSTEMİ

Yenilikçi mobil uygulamalara olan ihtiyaç artıkça, Optik Karakter Algılama (OCR) sistemleri de masaüstü ortamlardan mobil platformlara kaymıştır. OCR elyazısı yada herhangi bir metin içeren basılı veya taranmış döküman ve resimlerden metinleri dijital olarak değiştirilebilir ortama çıkarıp daha fazla analiz ve işleme olanak veren bir sistemdir. Bu çalışmada, fatura görüntülerinden metinleri otomatik olarak algılayıp çıkaran bir OCR android uygulaması önerilmiştir. Bu çalışmada Mobil cihazların kamerlarından elde edilen günlük fatura görüntülerinin OCR teknikleri ile etkili bir şekilde işlenmesi için gerekli temel teknikler araştırılmıştır. Fatura görüntülerinin kendine özel bazı karakterisitiklerinden dolayı tatmin edici sonuçlara ulaşmak için OCR sistemleri özel olarak bu görüntüleri işlemek için bir öğrenme sürecine sokulmalıdır. Olağandışı yazı tipleri ve boyutları, kısaltılmış kelime veya cümleler fatura dökümanlarını diğer dökümanlardan ayıran en önemli özelliklerden bazılarıdır. Bu çalışmanın temel amacı OCR teknolojisinin bir android uygulaması içerisinde fatura takibi amacı ile kullanılmasının uygun olup olmadığını araştırmaktır. Algılama safhasında, fatura resimlerindeki metinleri algılayıp çıkarmak için açık-kaynak kodlu Tesseract OCR kütüphanesi kullanılmıştır. Bu çalışma göstermeiştirki fatura görüntülerinin önerdiğimiz teknikleri uygulamadan Tesseract kütüphanesine direk olarak gönderilmesi oldukça düşük doğruluk değerleri ile sonuçlanmaktadır: %58,06 kelime doğruluğu ve %84,14 karakter doğruluğu. Ancak önerilen bütün teknikler uygulandığında örnek olarak kullanılan iki yazı tipi için Android uygulaması %88,72 kelime ve %96,61 karakter doğruluğu vermiştir. Ayrıca Android uygulamasının bir görüntüyü işlem süresi 6,56 saniye olarak hesaplanmıştır.

Anahtar Kelimeler: Fatura takip sistemi, OCR teknolojisi, Android App, Tesseract OCR motoru.

LIST OF FIGURES

	<u>Page No</u>
Figure 1.1 A sample of a receipt image used in this research for testing the suggested Android application.	4
Figure 2.1 An image that has scene complexity and a complicated background.	10
Figure 2.2 Irregular illumination and shadow problem on an image.	11
Figure 2.3 An image that has problem of skewing and its result after applying a de-skewing technique.....	12
Figure 2.4 Two samples of images that have the problem of un-focus and deterioration. ...	13
Figure 2.5 Two samples of images that have various aspect ratios.....	14
Figure 2.6 A sample of an image that has problem of tilting.	15
Figure 2.7 Two samples of text fonts utilized for testing suggested Android application.	15
Figure 2.8 Two samples of images that have the problem of bent text.	16
Figure 2.9 Result of performing canny edge detection algorithm on an image.....	21
Figure 2.10 Results of performing Gaussian filter on an image.....	22
Figure 2.11 Intensity gradient calculation and edge direction finding process on an image.....	23
Figure 2.12 The result of applying non-maximum suppression technique on an image.	23
Figure 2.13 Final results produced by applying Canny edge detection algorithm on an image.....	24
Figure 2.14 Step-by-steps of processes adopted by Tesseract OCR engine.....	34
Figure 2.15 An image contains 7 spots and baselines were identified by the baseline finding technique.	34
Figure 2.16 An example of a word that has different character pitches.	35
Figure 2.17 An example of a word with joined characters.....	36
Figure 2.18 An example of broken characters in a word.....	36
Figure 2.19 Broken and joined characters in a word can be recognized by static classifier algorithm.....	37
Figure 3.1 Step-by-step proposed techniques for the implemented receipt tracker application.....	38
Figure 3.2 The result of performing CED algorithm on a receipt image.....	40
Figure 3.3 receipt region shape detection.....	41

Figure 3.4 The receipt region is accurately identified and detected by using canny edge detection algorithm.	42
Figure 3.5 Image preprocessing algorithms adopted and applied in this research.	43
Figure 3.6 Result of performing contrast method on an image of the receipt.	44
Figure 3.7 The result of performing gray-scale function on a receipt image.	45
Figure 3.8 The result of performing thresholding algorithm over an image of the receipt..	47
Figure 3.9 Results of performing Median filter on an image of the receipt.	48
Figure 3.10 Adding black pixels to the holes of characters by using Erosion algorithm. ...	49
Figure 3.11 Handling skewing problem on an image of receipt by using a de-skewing function.	50
Figure 3.12 An example of training image was used for generating training file by Tesseract.	54
Figure 3.13 Editing a box file generated by Tesseract by using JTessBoxEditor software.	55
Figure 3.14 Tesseract proposed a list of ten-page segmentation methods.....	58
Figure 3.15 Proposed database ER-diagram in this research.	62
Figure 3.16 Structure of the proposed android application in term of client-server architecture.....	63
Figure 3.17 Structure of the client side's step-by-step processes in the suggested Android application.....	65
Figure 3.18 Structure of the server side's step-by-step processes in the suggested Android application.....	66
Figure 3.19 Screenshots of the login and registration page of the suggested Android application.....	68
Figure 3.20 A screenshot of the home page of the application.	69
Figure 3.21 Screenshots of detecting region of receipt by canny edge detection algorithm.....	69
Figure 3.22 Screenshots of submitting receipt image to the server and showing the result.	70
Figure 3.23 A screenshot of showing a message to the users of the suggested Android application.....	71
Figure 4.1 Screenshots of using Expend inspector query.....	73
Figure 4.2 Screenshots of using (Discovering images of receipts) query.	75
Figure 4.3 Screenshots of using total money expended query.	76

Figure 4.4 Screenshots of using (Total money expended for a particular item) query.....	78
Figure 4.5 Word and character rates histogram for fake receipt font in the first case.	83
Figure 4.6 Word and character rates histogram for fake receipt font in the second case. ...	85
Figure 4.7 Word and character rates histogram for fake receipt font in the third case.....	88
Figure 4.8 Word and character rates histogram for fake receipt font in the fourth case.....	90
Figure 4.9 Word and character rates histogram for merchant copy font in the first case. ...	93
Figure 4.10 Word and character rates histogram for merchant copy font in the second case.	95
Figure 4.11 Word and character rates histogram for merchant copy font in the third case.	97
Figure 4.12 Word and character rates histogram for merchant copy font in the fourth case.	99

LIST OF TABLES

	<u>Page No</u>
Table 2.1 Main techniques or algorithms of image pre-processing with a brief discussion.	20
Table 2.2 Various techniques of segmentation process utilized and suggested by researchers with their results.	26
Table 2.3 Various techniques of feature extraction utilized and suggested by researchers with their results.	27
Table 2.4 Neural network based OCR applications with the outcomes achieved.	28
Table 4.1 Outcomes obtained for fake receipt text font in the first examination.....	82
Table 4.2 Outcomes obtained for fake receipt text font in the second examination.	84
Table 4.3 Outcomes obtained for fake receipt text font in the third examination.	86
Table 4.4 Outcomes obtained for fake receipt text font in the fourth examination.	89
Table 4.5 Outcomes obtained for merchant copy text font in the first examination.....	92
Table 4.6 Outcomes obtained for merchant copy text font in the second examination.	94
Table 4.7 Outcomes obtained for merchant copy text font in the third examination.....	96
Table 4.8 Outcomes obtained for merchant copy text font in the fourth examination.	98
Table 4.9 Average percentages of outcomes obtained for two different fonts.	100

LIST OF ABBREVIATIONS

ANN	: Artificial neural network
ANPR	: Automatic number plate recognition
API	: Application programming interface
CED	: Canny edge detection
CPU	: Central processing unit
ERD	: Entity relationship diagram
GPL	: General Public License
GUI	: Graphical user interface
IDE	: Integrated development environment
ISO	: International Standards Organization
JRE	: Java runtime environment
OCR	: Optical Character Recognition
OpenCV	: Open source computer vision
OS	: Operating system
PSM	: Page segmentation method
RAST	: Rapid Annotation using Subsystem Technology
Regex	: Regular expression
SDK	: Software Development Kit
SQL	: Structured Query Language
SVM	: Support vector machine
TIFF	: Tag Image File Format
UTF8	: Universal Transformation Format-8

1. INTRODUCTION

Introduction chapter covers four subsections, in the first section, an overview about optical character recognition technology is discussed and why a study should be done about (an android based receipt tracker system using optical character recognition) is described, in the second section, problem statement of the study is presented, in the third section, the aims of the research are clearly shown and finally thesis layout is categorized.

1.1. Background

Nowadays there are huge demands for designing machines and tools which recognize and identify patterns such as pattern recognition machines often called fingerprint recognition machines, speech recognition machines, optical character recognition machines and many other types of machines. Of course, effective and accurate implementation of machines has only been focused on recently by researchers. Understanding the ways of finding solutions for a particular problem is the natural way for designing and implementing effective and accurate pattern recognition machines [1].

In our day to day life, we often need to reprint texts or modify them in some way. However, in many cases, the printable document of the text does not remain available for editing. For example, if a newspaper article was published 10 years ago, it is quite possible that the text is not available in an editable document such as a word or text file. So, the only choice remaining is to type the entire text which is a very exhaustive process if the text is large. The solution for this problem is optical character recognition [2]. Optical character recognition is a process which takes images as inputs and generates the texts contained in the input. So, a user can take an image of the text that he or she wants to print, feed the image into an optical character recognition (OCR) software and then the software will generate an editable text file for the user which is amendable. This file can be used to print or publish the required text.

Similarly, optical character Recognition (OCR) is the technology that converts the text from handwritten images, text printed images [3] or scanned images to the alterable text for further analysis and process. The capability of machines for automatically recognizing texts on images can only be achieved by OCR technology. For proposing accurate OCR Application, Several challenges should be handled. In some cases, a small visible difference

can be observed between digits and letters. Letter “o” and digit “0” looks like each other. This is an example of a situation that is often difficult for OCR machines to recognize.

In the literature review, researchers performed and applied OCR technology to different fields such as recognizing texts on license plates, recognizing text on images that were taken in a natural view [4], recognizing texts on images obtained through scanners. The quality of the images and text fonts on the documents are some examples of recent challenges related to the implementation of powerful OCR applications. For better applying OCR on different images, several different techniques and image processing algorithms must be investigated and studied during the designing process related to applications for optical character recognition technologies.

Detecting and recognizing texts on different types of documents by the human eye is a real life example for optical character recognition technology [4]. This real life example has similarities with the steps required for implementing a powerful OCR machine. The text of the document can be seen and identified by the human eye, then theoretically the human mind will process the detected text and provide the ability to the human to understand the text. Certainly, Man’s ability is much greater than OCR application’s ability, but understanding the way for solving text recognition problems in the nature way are useful. For example, if the text on the documents is not clear to the human eye, then human’s brain cannot provide well text understanding, of course, if the quality of the images is low, then OCR application cannot perform well-recognition.

In earlier decades, researchers focused on recognizing texts from printed documents with a fixed size and only one font. But recently, researchers focused on recognizing document texts that have several different fonts and sizes [5]. Recently demands converted to recognize text on the images obtained by mobile devices cameras instead of images obtained by the scanner. There are several different challenges related to the images obtained through mobile device cameras that must be handled for suggesting an effective and a powerful OCR application, therefore, research continues in the field of optical character recognition technology.

OCR research studies have a great influence on pattern recognition applications, for example; face recognition, fingerprint recognition and iris recognition. Such applications are used for some security issues such as criminal tracking. Recently, some systems have

integrated OCR with new research topics such as automatic translation and voice commands. These systems will play an important role in developing such new topics [6].

OCR technology has been performed in several different ways such as designing OCR on servers, OCR applications for mobile devices, OCR applications for desktops and so forth. Earlier, OS developers focused on implementing and designing powerful OS for mainframes and desktops, but now are focused on enhancing and maintaining operating systems for mobile devices. These are the current innovations in which developers are mainly focused. Proposing such operating systems as currently available for mobile devices encouraged developers of mobile applications to innovate and implement more types of useful applications for mobile devices. Optical character recognition technology is not far from such recent innovations since demands for innovating mobile apps get deeper, therefore innovations on designing and creating desktop OCR applications moved and shifted to propose and innovate mobile OCR application.

Currently, usage of mobile devices is at an all-time high. For the purpose of exchanging data around the world, desktop applications and internet on desktops proposed several different approaches to connect people around the world. But both desktop applications and internet usage on desktops cannot connect people anytime and anywhere. Such features, connecting people anytime and anywhere, can only be obtained through different mobile device applications and the internet on mobile devices. The current usage of mobile devices by people reached 4.61 billion in 2016 [7]. Recently usage of the internet on mobile devices is greater than the usage of internet on desktops or laptops.

Android operating system is an OS for hand-held devices currently being used by many mobile devices such as Huawei Mobile device, Samsung mobile devices and so on. Android platform is an efficient and powerful operating system platform designed and maintained by Google. Google focuses continuously on improving and maintaining the Android platform.

Since innovations and demands for implementing applications associated with optical character recognition technology has grown significantly, this research suggests the development of an android application for automatically extracting and recognizing text on the receipt of images. This research presents some new techniques for better performing text recognition technology on the receipt of images captured through cameras on hand-held devices to obtain and reach a powerful and efficient system for tracking everyday receipts

on hand-held devices.

Of course, receipt images have their specifics, therefore OCR applications must be trained for such kinds of images or else OCR technology cannot perform well-recognition. Receipt images have some characteristics that are different from other documents which must be considered during implementing OCR application for recognizing text on receipt images. Unusual text fonts, very small font size, also compressed characters, words and lines on receipt images are the most difficult characteristics of receipt images from other documents. Figure 1.1, is a sample of a receipt image which is used in this research for testing the suggested Android application.



Figure 1.1 A sample of a receipt image used in this research for testing the suggested Android application.

1.2. Problems Statement

There are some issues which are the main reasons for consuming time and making our lives more difficult if we want to save marketing receipts manually. Some of these problems are:

- Writing-down text written on the receipt images to a file requires a lot of time if the receipt image contains large-scale text and also some text might be missed during writing.
- Losing receipts is one of the issues if we want to save a number of receipts manually for a long period.
- Storing large-scale marketing receipts is the way for space consuming.
- Awareness for gathering receipts is the way that makes people worrying.
- Storing a number of receipts manually might cause security risks.
- If we have a large-scale number of receipts, then manually calculating total money expended for a period of time is a hard task.
- Also finding information from a large-scale number of receipts cannot be feasible such as searching for purchased items, prices of items purchased, the name of markets and much more.
- Receipt destroying and useless are possible when we want to store receipts for a long time.
- When people want to travel, they must create room in their bags for saving and holding marketing receipts.

1.3. General Aims and Objectives

Assume that you purchased a dozen eggs from a market for making a delicious meal for your brunch and you have a plan to travel somewhere in the afternoon with your close friends. Suddenly you observe yourself becoming sick after eating the eggs at brunch. Then you go to the doctor and tell him/her what happened. If the doctor finds that you are sick because you ate an expired food, then what could be your next action? Of course, you will

write a petition to the market that sold you the expired eggs. However, in order to show that you bought the expired eggs at that market you will need proof. The only way for proving proof is by showing them the receipt which contains all the information related to the market and the purchased eggs.

Certainly, for a period of time, there are some people who hold or save marketing receipts. However, there are probably many people that will not save or hold a receipt for even a second.

On the other hand there are many people who might want to return an item that has recently expired. Think about the time and worrying when we want to manually search for a receipt among a large-scale number of receipts for finding the info related to the market who purchased unwanted or expired items! And think about an application to do all the mentioned task in seconds. In this research, we want to implement and suggest an android application that can easily find any information related to the purchased items.

This research suggests an Android app that combines optical character recognition technology with mobile devices in one OCR system. In this study, several different techniques, algorithms of image processing and OCR techniques and engines are studied and investigated in order to offer a new OCR application for easily tracking daily marketing receipts. By utilizing the suggested Android application, users of the application can easily capture a receipt image or browse to a receipt image from the mobile device's gallery and submit it to the system for identification and recognition. The information that can be identified and recognized by the suggested Android applications would be such things as the name of the market, number of markets, phone numbers of the market, website of the market and addresses of the market, Purchasing time and date, ID number of Receipt and item name, price and ID, and Total money expended, total money paid for tax, total money paid for cash, change due money. Such suggested Android application in this research can easily return the time requires when we want to search for the mentioned information in the big pile of receipts manually.

By utilizing the suggested Android application, users of the application can simply write down "egg" in a search box and get all the receipt images that include searched name based on the date of marketing. Then users of the application can select a receipt from the sorted receipt images and get the evidence for proving the situation to the market.

Or if the users of the application want to return a purchased item to the market due to any problem, they must show or provide the receipt as evidence of purchase. In such a situation, users can write down the information into the designed search box like the name of the market, date of marketing and so on for finding the receipt that the users of the application require as evidence.

Two other beneficial queries are implemented in the suggested Android application. The first one is expending inspector, in a nicely formatted chart, the users of the application can see the expend money history of one year ago for all months separately. The second one is total money expended for a particular item, where the query will find and computes total money expended for let say “coffee” within in two months ago.

The main aims and purposes of this research are to find and investigate whether OCR technology is feasible for an application like this or not. This research investigates and studies several different techniques for suggesting an efficient and accurate OCR application for recognizing text on receipt images. In the market there are some OCR applications available commercially like what proposed in this study, proposing a better OCR application is the main aim of this study by suggesting nicely formatted GUI, much more queries and much more features. Also, academically showing and presenting techniques utilized by commercial app developers for implementing and designing such applications are another aim of this research.

1.4. Thesis Layout

Description of the rest chapters are listed below:

Chapter Two: [THEORETICAL TECHNIQUES AND BACKGROUND OF OCR]

This chapter covers four important subsections. First, the main issues associated with the images through cameras of mobile devices and OCR techniques are described for implementing accurate and efficient OCR applications. Second, several different usages of OCR technology in different fields are discussed. Third, the main pipeline of steps and techniques required for designing OCR applications are categorized and discussed. Finally, recent and powerful proposed OCR engines are listed and discussed.

Chapter Three: [PROPOSED TECHNIQUES]

This chapter covers and discusses the main and most important techniques and algorithms used in this research to overcome the problem of performing optical character recognition technology on receipt images. Then the general system overview is discussed and then the practical side of the system and screenshots of the suggested Android application are showed and presented.

Chapter Four: [QUERIES AND EXPERIMENTAL RESULTS]

Showing or presenting outcomes of created queries and experimental outcomes practiced in this research for the suggested Android application is the main concept presented in this chapter.

Chapter Five: [CONCLUSION AND FUTURE WORK]

This chapter covers a final discussion of the techniques and algorithms utilized in this research for handling the issue of applying OCR technology on receipt images. Also final results obtained in this research are concluded and presented. Finally, possible future works related to this research are showed and presented.

2. THEORETICAL TECHNIQUES AND BACKGROUND OF OCR

This chapter covers four important subsections. First, the main issues associated with the images through cameras of mobile devices and OCR techniques are described for implementing accurate and efficient OCR applications. Second, several different usages of OCR technology in different fields are discussed. Third, the main pipeline of steps and techniques required for designing OCR applications are categorized and discussed. Finally, recent and powerful proposed OCR engines are listed and discussed.

2.1. OCR Challenges

For implementing and innovating powerful and accurate OCR applications, the input to the OCR application which is images contains text should be enhanced and eliminated from any noise that causes OCR producing bad and unpromising outcomes. Preparing and enhancing images before submitting to the next phase in the OCR pipeline is one way of improving applications and obtaining better results. Usually, images acquired through mobile device cameras face much more challenges than images acquired through scanner devices. Therefore optical character recognition machines perform better and provide better outcomes when the input to the OCR engine is an image acquired through scanner devices. Means images acquired through mobile device cameras encounter more challenges and require more techniques to be smoothed and enhanced than images acquired through scanner devices. There are several different challenges facing images obtained from mobile devices that must be considered for improving OCR applications. These challenges are listed and discussed in the following subsections.

2.1.1. Complexity of scene

In the real world, several different objects exist that have similarities with texts such as symbols, buildings and paintings. Sometimes these objects look like characters in text. When we want to capture an image that includes both text and symbols and then if we want to submit these images to the OCR engine without separating symbols from the real text, then OCR cannot perform well-recognition and produces bad outcomes [8]. Again capturing an image that has a complicated background and feeding such an image to the OCR engine produces bad performing of segmentation and feature extraction phase. Hence

bad outcomes will be produced and seen. Figure 2.1 [9] is an example of the kinds of challenges that have complicated backgrounds.

For our case, text on the receipt images are understandable and receipt images do not include any symbols that might have similarities with the characters in texts. Usually background of receipts are clear and only contains the text. The text is the information related to the market and purchased items.



Figure 2.1 An image that has scene complexity and a complicated background.

2.1.2. Uneven lighting problem

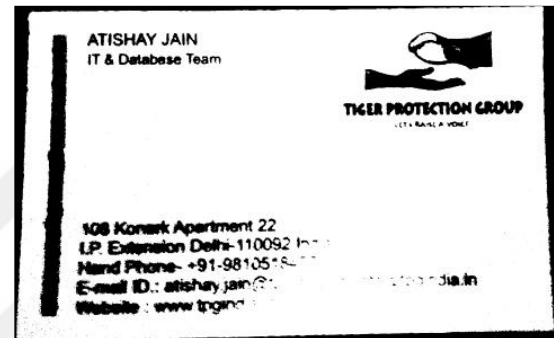
Usually capturing images through mobile device cameras suffer from the problem of shadow and irregular illumination on the images. Such problems introduce a new challenge for OCR that must be handled in order to improve applications [8].

Researchers in the literature suggest various dissimilar techniques for handling the problem of irregular illumination and shadows on the images. The most well-known and most utilized techniques are binarizing images by global and adaptive thresholding. Global thresholding uses one value of thresholding for binarizing all the pixels inside an image, whereas adaptive thresholding uses a single and unique value of the thresholding for binarizing each pixel that exists on an image. For handling the problem of shadows and irregular illumination on images, adaptive thresholding is preferable to utilizing global thresholding [10, 11]. Since adaptive thresholding utilizes a unique value of thresholding for binarizing each pixel on the images, it requires more time and causes the application to

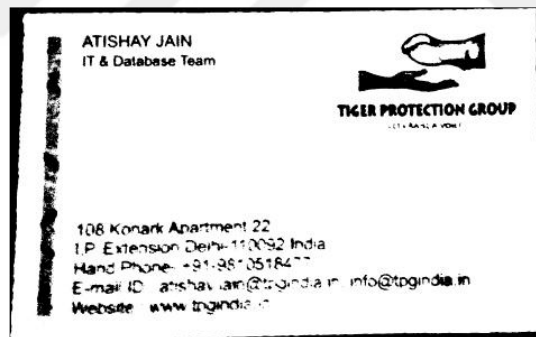
be inefficient. This research utilizes the global thresholding technique for binarizing receipt images and handling the problem of irregular illumination and shadows on the receipt images. Since, the Android application suggested provides an opportunity to the users of the application for re-taking images or re-browsing to an image if the results of the OCR are useless, therefore for this study, problem of irregular illumination and shadows on the receipt images less occur. Figure 2.2 [12], is an example of the images that faced such problems.



a) Irregular illumination and shadow on an image.



b) A global thresholding (Otsu's algorithm) result [13].



c) An adaptive thresholding (Sauvola algorithm) result [11].

Figure 2.2 Irregular illumination and shadow problem on an image.

2.1.3. Skewness problem

Of course taking images through utilizing mobile device cameras is one of the ways that causes the problem of skewing to exist on the images. Images acquired through scanner devices have less skewing problem. The skewing problem causes skew of text lines from

the background on an image. Submitting such images that have skewing problem to the OCR engine causes bad and useless performing during the segmentation phase which is the main and most important phase in the OCR process for promoting the rate of recognition. Researchers in the literature review suggest various dissimilar techniques for handling the skewing problem on images. These techniques are the Fourier transformation algorithm, Hough transform technique, RAST method, techniques of Projection Profile, ImageMagick library command (deskew command) and so on.

In this research for properly handling the skewing problem on receipt images, two techniques are utilized. The first one is utilizing an algorithm of edge identification which is called the canny edge detection algorithm. Behind utilizing canny edge detection algorithm for the purpose of identifying and extracting the region of receipt from background of receipt, also it was used to handle the problem of skewing on receipt images. The second technique is utilizing the de-skew function [14] from a well-known image processing library which is called ImageMagick library. A sample of an image that has the skewing and results of it after performing some deskew techniques is shown in Figure 2.3 [15].

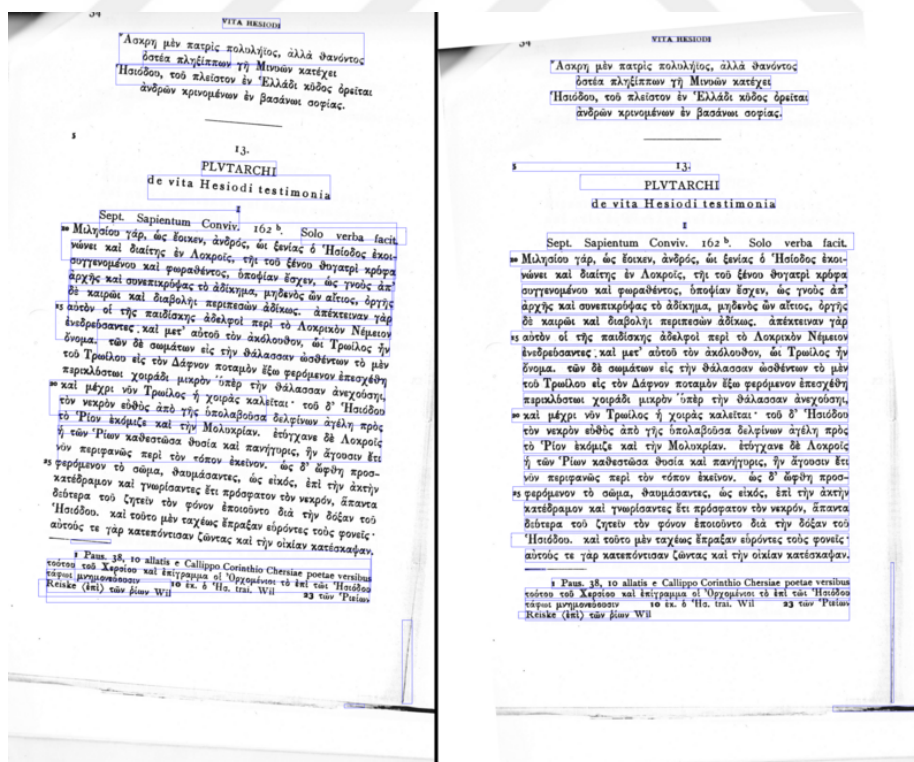
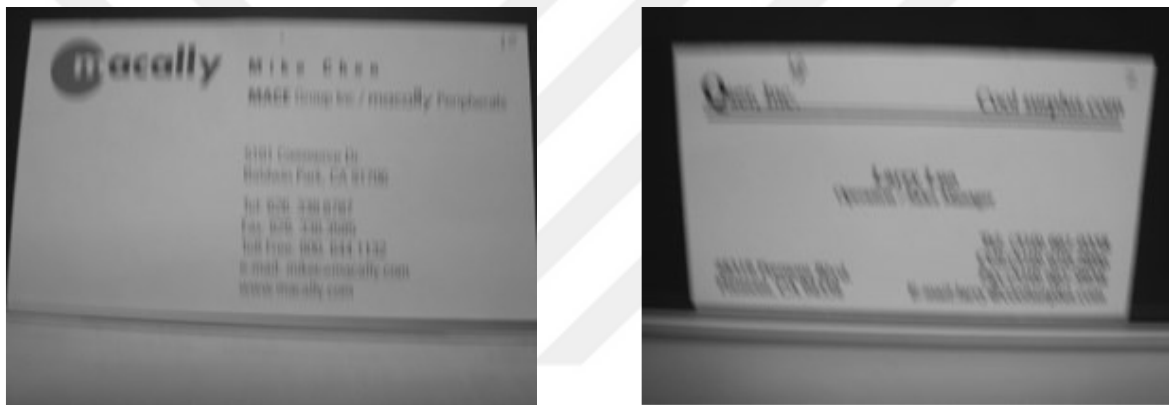


Figure 2.3 An image that has problem of skewing and its result after applying a de-skewing technique.

2.1.4. Un-focus and deterioration

Again, images obtained through scanner devices suffer less problems of un-focusing and deterioration. In contrast, images acquired through mobile device cameras experience more problems. Because images can be taken over a variety of distances by mobile device cameras. This problem can occur in two ways. First when mobile device cameras are out of focus and second when the mobile device cameras moves during capturing images. The new smart mobile devices have a feature called auto-focus, which eliminates mobile device cameras from taking images that have blurring or degradation problems. In Figure 2.4 [16], a sample of an image that has problems of out of focus or deterioration is shown.



a) Out of focus business card image.

b) Deterioration problem on a business card image.

Figure 2.4 Two samples of images that have the problem of un-focus and deterioration.

2.1.5. Aspect ratios

Of course, documents that we need to recognize texts on have different lengths and scales. During the implementation of any application associated with OCR, the length of the text should be considered for better and efficiently apply OCR related techniques. The main objectives for considering such problems are eliminating the computational complexity of the OCR applications. Since texts on the receipt images have different lengths, sometimes they might be long and sometimes they might be short, this research considers this issue by performing heavy techniques and methods on the server side, instead of applying on the client side. In Figure 2.5, we showed two samples of images, in the first sample, the length of the text is long, whereas, in the second sample, the text length is short.

ADVERTISEMENT.

THIS publication of the Works of JOHN KNOX, it is supposed, will extend to Five Volumes. It was thought advisable to commence the series with his History of the Reformation in Scotland, as the work of greatest importance. The next volume will thus contain the Third and Fourth Books, which continue the History to the year 1564; at which period his historical labours may be considered to terminate. But the Fifth Book, forming a sequel to the History, and published under his name in 1644, will also be included. His Letters and Miscellaneous Writings will be arranged in the subsequent volumes, as nearly as possible in chronological order; each portion being introduced by a separate notice, respecting the manuscript or printed copies from which they have been taken.

It may perhaps be expected that a Life of the Author should have been prefixed to this volume. The Life of Knox, by DR. M'CRIE, is however a work so universally known, and of so much historical value, as to supersede any attempt that might be made for a detailed bio-



a) A sample of the image has long text length.

b) A sample of the image has short text length.

Figure 2.5 Two samples of images that have various aspect ratios.

2.1.6. Tilting problem

The Tilting problem is another challenge that should be considered during implementing any OCR application for accurate recognition. When an image has the problem of tilting, text lines which are far from the mobile device cameras seem smaller than the text lines that are close to the mobile device cameras. Tilting never occurs on images acquired by scanner devices, since scanner sensors are properly parallel to the document when they scan an image. This problem can only occur on images that are acquired by mobile device cameras.

The research considered and handled the tilting problem by applying an edge identification algorithm called canny edge detection algorithm. Behind utilizing this algorithm for identifying and extracting the region of receipt from the background of receipts, this method was also utilized for handling the tilting problem on receipt images. Figure 2.6 [12] is a sample of an image that has the problem of tilting.

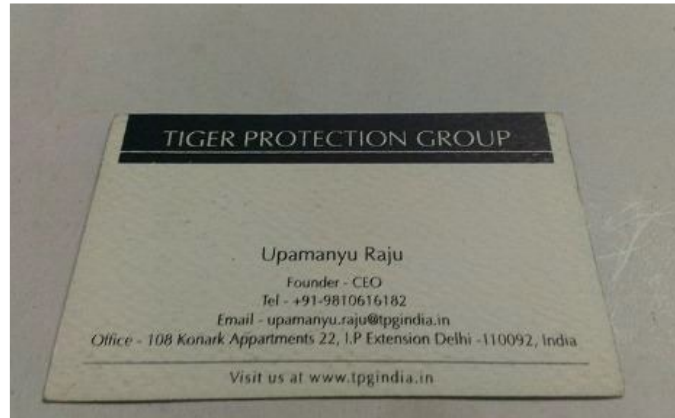


Figure 2.6 A sample of an image that has problem of tilting.

2.1.7. Fonts

The most important issue that should be considered during implementing any application associated with OCR technology is the text font of the images. Various text fonts exist in which different shapes and different characteristics differentiate text fonts from each other. Directly feeding unusual text fonts into the OCR engines without training the OCR engine results in bad preformation of the main OCR stage which is segmentation phase [17] and produces unpromising outcomes. The research suggested an Android application that can recognize two unusual text fonts on receipt images. The text fonts are fake receipt text font [18] and merchant copy text font [19]. A sample of both text fonts utilized in this research is showed in Figure 2.7 [18, 19].

FAKE RECEIPT REGULAR

a) A sample of fake receipt text font.

Merchant Copy Regular

b) A sample of merchant copy text font.

Figure 2.7 Two samples of text fonts utilized for testing suggested Android application.

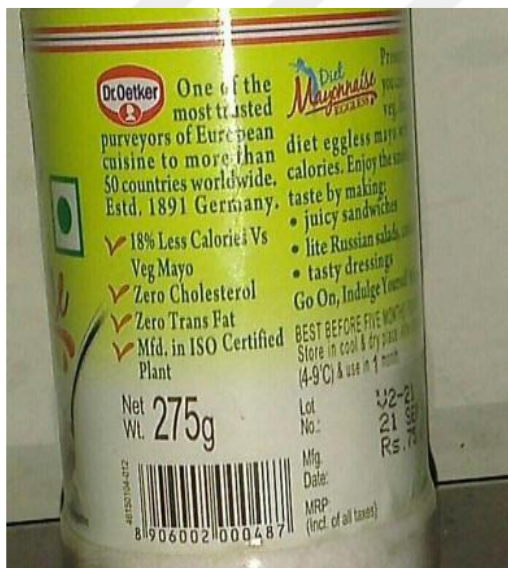
2.1.8. Multilingual environments

Some languages like English, French, and Spanish and so on, have a specific number of character classes. Some other languages like the Korean language, Chinese language, and

Japanese language have a huge number of character classes. Some other languages like the Arabic language have a special characteristic which is changing character shapes according to the context. Such challenges in several different languages still remain and need to be improved. In this research, the suggested Android application can only recognize the English language text on the receipt images for two different and unusual text fonts. One of the future areas would be improving the suggested Android application to recognize more language texts on receipt images like the Arabic language, Turkish language and so on.

2.1.9. Warping problem

Another challenge in the pipeline of problems associated with the OCR technology is the bent text on the objects. This situation never occurs when input images are acquired through the use of scanner sensors. However, it is possible for the images acquired by using mobile device cameras. Such problems can be seen in Figure 2.8 [12]. Where the first image is a code written on a bent delivery holder and the second image contains text on a warped bottle of milk. Since receipt images can be acquired instantly after marketing by mobile device cameras, therefore warping of text on receipt images rarely happens.



a) A bended text on a bottle of milk.



b) A bended text code on a delivery holder.

Figure 2.8 Two samples of images that have the problem of bent text.

2.2. OCR Applications

Optical character recognition technology has been applied and performed on various dissimilar fields for handling problem associated with optical character recognition. This section lists and discusses various fields where optical character recognition technology plays a major role.

2.2.1. Hand-writing recognition applications

One of the usages of optical character recognition technology is to recognize hand-written text on images [20]. Applying OCR to images that have hand-written text poses and introduces new challenges that should be handled for improving the rate of recognition by OCR. One of the challenges is that numerous different shapes and characters exist. In such cases, OCR engines should properly train for different shapes of the same characters. Usage of hand-written recognition technology can be divided into two different application areas. These are offline hand-written recognition applications and online hand-written recognition applications. Offline hand-writing recognition applications can recognize hand-writing texts on the documents. On the other hand, online hand-writing applications can recognize hand-written texts during writing instantly. For example writing on the screen of a device by a pen or human finger.

2.2.2. Healthcare applications

Optical character recognition is a useful technology because it can be utilized in different fields. Another important usage for optical character recognition is to recognize texts on medical forms and other sources of printed papers [21]. Researchers tried to implement and design OCR applications that can easily identify and recognize useful information on the documents related to medical patients. Such applications are helpful for doctors and experts in medicine for extracting patient information on medical papers and saving this information to a database for later use.

2.2.3. Financial tracking applications

Another innovation of optical character recognition technology is implementing OCR

applications to observe and track financial transactions [21]. Most of the well-known organizations and companies use OCR applications among several different applications for simplifying tasks and efficiently managing the organization's tasks. Barcode recognition is one example which utilizes OCR technology for recognizing barcodes on things related to the organization. Barcode recognition technology simplifies tasks and efficiently manages the organization's tasks.

2.2.4. Legal industry

Utilizing optical character recognition technology in legal industries is another usage of OCR [21]. Legal industries use optical character recognition technology for extracting and recognizing text on judicial documents for further processing and analysis. Extracted texts can be saved to the database and later judicial experts can obtain benefits from this information by only writing a word in a search box.

2.2.5. Banking application

Nowadays, utilizing innovations in OCR technology has been expanded to recognize and extract texts on the printed documents from financial institutions and banks [21]. OCR applications related to banks can extract and recognize useful information on check for deeper analysis and processing. A check can be inserted into an OCR machine and then the OCR machine will extract the bank participant's information from the check and compare the information with the database. Finally, the institution can respond to the bank participant's demands. Such innovations can accurately recognize information on the printed checks and hand-written checks.

2.2.6. Captcha breaking application

CAPTCHAs [22] are the security tests that are most often found on websites that need to register for login to the website. Usually, this security test provides a sequence of characters or numbers by an image that users of the website must write down. The characters or numbers are manually entered into the text box in order to login to the website. These security tests are for ensuring that the website is utilized by a human rather than an automatic machine (attackers). This way prevents fake logins to the websites by attackers.

For the purpose of breaking such kinds of security tests, several different ways are proposed in the literature review. The most utilized way is through using optical character recognition technology where OCR applications can recognize and extract texts on some of the security test images. Then this information can be written down to the text box for login to the websites who uses text-based Captcha.

2.2.7. Automatic number plate recognition application (ANPR)

Another important usage of optical character recognition technology is recognizing text on the registration plates of vehicles [23]. Such application is useful for police forces. The application will capture an image of the registration plate of vehicles and then will submit the captured image to the OCR engine for extracting and recognizing text on the submitted image. Finally it will save both captured image and recognized information to the database for later use. Recognition of registration plates of vehicles still remains an issue to be researched further, because registration plates on vehicles vary from country to country.

2.3. OCR Phases

Main stages and methods of optical character recognition technology are listed and discussed in this section. These stages are the preprocessing of image stage, character and word segmentation stage, normalization stage, character feature extraction stage, character classification stage and finally post-processing stage. For promoting the rate of recognition by OCR application, we should understand and follow the main instructions for handling challenges that might occur at each stage. In the literature review, several different techniques and methods have been suggested by researchers to improve OCR applications in different fields. Based on this study and investigations a series of techniques and algorithms have been found that are useful for our case when recognizing texts on receipt images. These OCR stages are listed and discussed in the following subsections with the techniques and methods used in each stage.

2.3.1. Image pre-processing phase

The main purpose of applying image processing algorithms to images before feeding them into the OCR engine is to eliminate noise and enhance the image for achieving better

recognition rates. There are no differences between binarized images, colored images and gray-scaled images. Means applying image pre-processing algorithms on the images are required and important for every type of images. Processing colored images by OCR applications is computationally expensive, therefore the most important pre-processing stage is binarizing images before submitting colored images to the next OCR stage. For better application and performance, other OCR stages especially the segmentation stage highly depends on performing image preprocessing techniques.

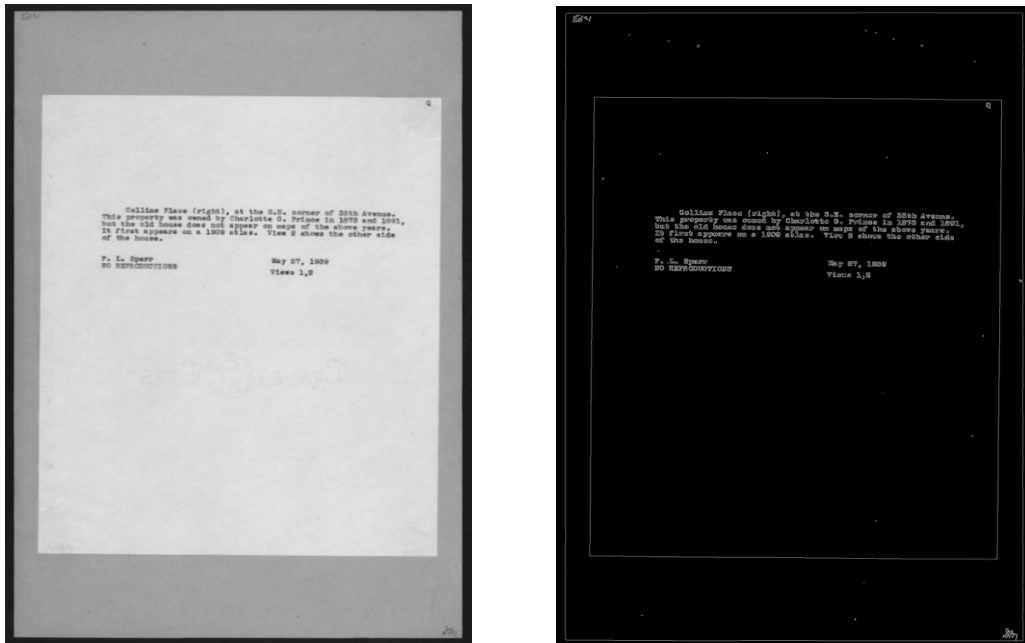
For different fields associated with OCR, challenges related to the image acquired through mobile device cameras must be handled for the successive performing of OCR stages. Image pre-processing techniques can be divided into two techniques. The two techniques are first applying an edge identification algorithm to the images for identifying and extracting the region of text. The second technique is applying some image processing algorithms for smoothing and enhancing text regions for better applying other OCR stages. Both techniques are showed and discussed in detail in this section. Some algorithms or techniques of image processing that should be applied to the images are listed and discussed in Table 2.1.

Table 2.1 Main techniques or algorithms of image pre-processing with a brief discussion.

Technique	Discussion
Skeletonization and thinning	the Skeletonization technique is thinning of text, that adjusts the shape of the text till achieving the width of text with one pixel.
Thresholding technique	It is the process of separating text pixels from the background pixels.
Morphological operations	It is the process of adding black pixels to the holes of characters and adding white pixels to the unwanted black pixels.
De-skewing technique	The skew problem on the images acquired through cameras of mobiles devices can occur, there for the de-skew technique is requires.
Reduction of noise	A technique for Reduction of noises and small speckles should perform on images such as median filter technique.
Binarize process	Binarization is the most important technique of image pre-processing, that converts gray or color image to the black and white image.

In the literature, researchers applied various edge detection algorithms to the different problems related to machine learning. The edge detection algorithms are such as Prewitt's operator [24], the algorithm of canny edge detection [25], Laplacian of Gaussian [26], operator of Robert's cross [27], operator of Sobel [28] and so on. In this research, we used the canny edge detection algorithm for identifying the region of receipts from the background of images.

The most well-known and most utilized edge identification algorithm is the canny edge detection algorithm. John F. Canny was the designer of this edge detection algorithm introduced in 1986. These algorithms use various calculations and algorithms for finding and extracting edges of shapes in an image. A paper [25] proposed by the designer of the canny edge detection algorithm, that illustrates everything related to this algorithm including the algorithm's capabilities, architecture and so on. Among various strong edge identification methods, the Canny edge detection algorithms is the most utilized edge identification algorithm by researchers for handling several different problems associated with machine learning. Accurately and efficiently identifying edges on an image are the main factors that the canny edge detection algorithms addressed. Identifying edges on an image by use of the canny edge detection algorithms are shown in Figure 2.9 [29].



a) An image contains text.

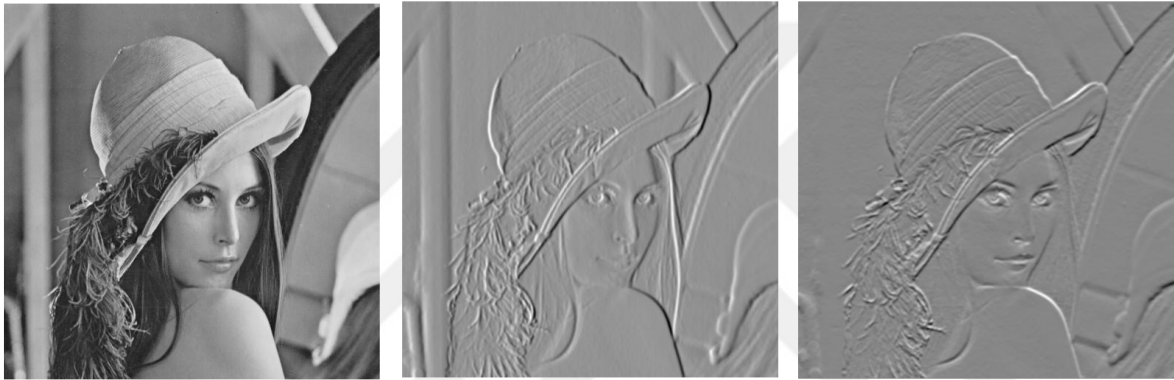
b) Canny edge detection algorithm effects.

Figure 2.9 Result of performing canny edge detection algorithm on an image.

Step-by-step of the structure or architecture of the canny edge detection algorithm is discussed in the following subsections:

A. Gaussian filter method

The major factor that affects every edge detection algorithm is the phenomenon known as noise on the images. Therefore for accurate edge identification noises or small speckles should be eliminated and removed. To achieve this goal, the canny edge detection algorithm uses an algorithm as the first technique for improving accuracy called the Gaussian filter technique. Figure 2.10, shows the effects of the Gaussian filter algorithm on an image.



a) Lina image.

b) Gaussian's X-Derivative result.

c) Gaussian's Y-Derivative result.

Figure 2.10 Results of performing Gaussian filter on an image.

B. Computing intensity gradient and directions of edges

The Canny edge detection algorithm applies four filters for finding the direction of the edges. The directions which are possible for every edge are vertical direction, horizontal direction and diagonal direction.

The process of finding the direction of edges is divided into two phases. At the first phase, an operator of edge identification such as Sobel, Prewitt or Roberts will compute the first derivative for both X-direction and Y-direction. From these directions at the second phase, it will compute directions and gradient intensity of edges in an image. The effects of this process are presented and shown in the Figure 2.11.



Figure 2.11 Intensity gradient calculation and edge direction finding process on an image.

C. Non-maximum suppression process

After finding intensity gradient and directions of edges on the images, then the canny edge detection algorithm will apply a technique called the non-maximum suppression technique for the purpose of thinning edges. This technique will find and establishes strong candidate pixels that are possible for being a part of an edge. This means that every pixel which is not a strong candidate will be removed and eliminated. The effects of this process are presented and shown in the Figure 2.12.



Figure 2.12 The result of applying non-maximum suppression technique on an image.

D. Hysteresis thresholding

The final stage in the processes of canny edge detection algorithm is thresholding images by using two different threshold values. The first threshold value will be a small value and the second threshold value will be a large value. After thresholding images, then the canny edge detection algorithm will apply a technique for making a decision on pixels in the images to be a part of an edge or not. The technique will go through every pixel that exists on an image and will compare the intensity gradient of the pixels with the two selected threshold values (big and small threshold value). If a pixel's intensity gradient value is greater than the selected big threshold value, then the canny edge detection will accept this pixel as an edge. On the other hand, if a pixel's intensity gradient value is smaller than the selected small threshold value, then the canny edge detection will ignore this pixel for being an edge. In a situation where the pixel's intensity gradient value is between both selected threshold values, then the canny edge detection algorithm will accept this pixel only if it is associated to a pixel that was accepted to be an edge. The final result of using the canny edge detection algorithm is shown in Figure 2.13.



Figure 2.13 Final results produced by applying Canny edge detection algorithm on an image.

2.3.2. Segmentation phase

The most important and effective process in the stack of processes of OCR application is the segmentation of text lines, words and characters that accurate performing of

segmentation process have impacts on promoting the recognition rate. Applications related to the OCR requires an image as input to the OCR application and produces a text file as the outcome of the OCR application. The images will first be pre-processed and enhanced by applying a series of image processing algorithms such as gray-mode operation, Binarization, reduction of noise, morphological operations, de-skewing process and so forth. After the image pre-processing stage, then for segmenting text lines, segmenting words and segmenting characters, the image will be submitted to the segmentation stage. The segmentation stage will removes unwanted regions from images such as background of images, following this, the algorithm will find text lines and then applies word and character segmentation. For segmenting documents of text, researchers utilized one of the three techniques of document segmentation [30]. The techniques are:

- Top to down algorithms,
- Bottom to up algorithms,
- Hybrid algorithms.

The first techniques for segmenting text on the images are through utilizing top to down algorithms. The main procedure for this approach is recursively segmenting large districts of text to the smaller district until reaching segmentation of all the characters properly. When all the characters are segmented properly then the procedure will stop. The second technique for segmentation utilizes bottom to up algorithms. The algorithms start by finding pixels that are strong candidates to be a pixel in a character, then the algorithm will merge all the strong candidate pixels to create an image for a character. After finding all the characters, then the algorithm will merges characters for creating words, then from this words the algorithm will create lines of texts and finally produces a block of text from the text lines. When both techniques (top to down algorithms and bottom to up algorithms) are mixed and used during segmentation process in an OCR application, then these techniques called hybrid techniques. In literature review, researchers suggested various techniques and methods for segmentation purpose. Some of the techniques and their results are listed and presented in Table 2.2.

Table 2.2 Various techniques of segmentation process utilized and suggested by researchers with their results.

Researches	Techniques	Result %
[31]	Word's border finding for Urdu language.	96.10 %
[32]	Algorithms of projection profile (vertical and horizontal).	98 %
[33]	Segmenting lines by interline distance and vertical projection. Segmenting words by interword distance and horizontal projection.	87.1%
[34]	Neighborhood Connected Component Analysis technique.	93.35 %
[35]	hypothetical water flows technique.	91.44 % and 90.34 % for Bengali hand-written images and English document images respectively.
[36]	A Hough transform based technique.	85.7 %, 88 % and 94.6 % images of documents, images of surveillance camera and images of business cards respectively.

2.3.3. Normalization phase

When the procedure of character segmentation is finished, then the output will be a set of images of characters, for better-applying techniques of feature extraction, these images of characters should be normalized and minimized to a certain size. This procedure is important because it removes and eliminates unwanted information from the character images without having any influence on the significant information. In this way, the technique will promote and rise accuracy of feature extraction from the images of characters for better performing classification algorithms [37].

2.3.4. Feature extraction phase

Feature extraction phase is another important phase in the stack of processes required for designing and implementing an efficient and accurate OCR application. Feature extraction

is the procedure of obtaining features from each character for building feature vector. Later classification algorithm will utilize these feature vector for classifying between characters and recognizing each character [38]. These feature vector make recognizing dissimilar characters easy by classification algorithm [39].

Structural features and statistical features are the two feature classes suggested and proposed by Suen [40]. The first feature class is structural features. This class of features uses the geometry of the characters. Several different features belong to structural features such as number of character holes, features of concavity and features of convexity related to the characters and so forth. On the other hand, another class of features is the statistical features class. These features use the matrix of characters. Several different features belong to structural features such as projection histograms feature, Fourier transforms feature, crossings feature, moments feature and zoning feature [41]. In the literature review, researchers suggested various methods for the purpose of feature extraction. Some of the methods and their results are listed and presented in Table 2.3 [41].

Table 2.3 Various techniques of feature extraction utilized and suggested by researchers with their results.

Researchers	Techniques	Result %
[42]	Both structural and statistical features are used.	90.18 %
[43]	Fused statistical features.	91.38 %
[44]	linear discriminant analysis classifier.	67.30 %
[45]	Modified direction features.	89.01 %
[46]	Directional features.	Low: 70.22 % high: 84.83 %
[47]	Hybrid feature extraction method.	85.08 %

2.3.5. Classification phase

After obtaining feature vector from the feature extraction phase, then the classification phase will start classifying each character to a predefined class by utilizing feature vector. Usually, classification phase is the final stage in OCR applications for making a decision on the character images to be recognized by the classifier algorithm. For obtaining accurate

and efficient classification of characters in any application associated with OCR, training classifier algorithm for different shapes of character images is the most well-known factor. In the literature, researchers suggested several different classifiers for different fields associated with OCR. Selecting appropriate algorithms for classification purpose for different fields depends on several different factors that must be considered during implementing any application associated with OCR. The factors are available training dataset, classifier's parameters and so on.

There are two basic steps to using the classifier: training and classification. The most important progress in the classification phase that have great impacts on raising recognition rate is training classifier for the unknown classes. Training is the process of taking content that is known to belong to specified classes and creating a classifier on the basis of that known content. Classification is the process of taking a classifier built with such a training content set and running it on unknown content to determine class membership for the unknown content. Training is an iterative process whereby you build the best classifier possible, and classification is a one-time process designed to run on unknown content.

The most utilized classification algorithms in the literature are support vector machine method (SVM), template matching technique, artificial neural network algorithms (ANN), statistical methods and hybrid classification techniques [48]. The most well-known and most utilized classification algorithm in the literature for OCR applications is the neural network algorithms. Table 2.4 shows some research and their outcomes. The neural network algorithm is utilized by these researchers for handling different problems associated with OCR.

Table 2.4 Neural network based OCR applications with the outcomes achieved.

Researches	OCR system	Result %
[49]	OCR application for broken character recognition.	68.3 %
[50]	OCR application for recognizing text on the Urdu documents.	98.30 %
[51]	Automatic Number Plate Recognition.	97.30 %
[52]	Recognition of chassis number.	95.49 %

2.3.6. Post-processing phase

The Post-processing phase is an optional phase in the stack of processes required for designing and implementing OCR applications. This means it is not necessary to use some techniques in the post-processing phase. However, for designing and implementing an efficient and accurate OCR engine, it is important to consider using some techniques in the post processing phase. For raising the rate of accuracy of OCR applications, there are some techniques that should be considered during implementing any application associated with OCR. One of the techniques of post-processing phase is by using a dictionary. When the classifier produced the output, then a technique will compare the words in the recognized text with an English dictionary for correcting wrong character detections.

2.4. OCR Engines

The conventional pipeline of designing and implementing applications associated with OCR are presented and discussed in the previous section. Another approach or technique for implementing applications associated with OCR is through utilizing OCR engines. This research follows utilizing an open source OCR engine which is the Tesseract OCR engine. For handling the problem of performing OCR on the images of different fields, various OCR engines exist in the literature. Some of these are listed and discussed in the following subsections. Based on experiments and investigations through various researches suggested in the literature, finally the research figured out that Tesseract OCR engine fulfills requirements for implementing a powerful and efficient OCR application for handling the problem of applying OCR on the receipt images.

2.4.1. GOCR engine

GOCR is also known as JOCR [53]. It is an engine for optical character recognition technology initially designed and implemented by Joerg Schulenburg. Now it is managed by a team of engineers for the purpose of continuously enhancing the GOCR engine. GOCR takes the images contained in the text and produces a text file containing the editable text. This OCR engine utilizes a feature extraction technique for extracting features from the character images known as the spatial feature extraction technique. GOCR can recognize text on several different image formats. GOCR can be installed on several different

operating systems because it is designed in a way that is compatible with various operating systems. The operating systems that GOCR can be utilized with and support are Mac OS, Linux OS and Windows OS. The GOCR engine is written in a powerful programming language named C programming language. The final version of GOCR which was published in 2013 is the 0.50 GOCR version. The research ignored utilizing GOCR for handling the problem of applying OCR on receipt images, the main reason behind this decision is because GOCR is not a well-documented OCR engine and it is more preferable for commercial OCR applications instead for academic OCR applications.

2.4.2. Ocrad engine

The Ocrad engine [54] is another OCR engine suggested and utilized in the literature. It is published under the GNU GPL license. The OCR engine can efficiently recognize English text on images with a clear background. Ocrad can only be utilized with Linux OS. For the purpose of classification, Ocrad utilizes techniques of feature extraction and techniques of template matching. One of the techniques of image preprocessing is known as document layout analysis. This is a useful technique for identifying and extracting text regions from images containing shapes and tables. These techniques are also adopted in the Ocrad engine.

The Ocrad engine is an open source engine and is written by an efficient programming language named C++ programming language. This engine was designed and implemented in 2003. Ocrad can be used through command prompt by command lines, because Ocrad engine does not have GUI. This study found that the Ocrad engine is not preferable for recognizing text on receipt images because of two main factors. The factors are first, weak documentation of the Ocrad engine and second, the OCR engine did not perform in various areas associated with OCR.

2.4.3. OCRopus

The OCRopus engine [55] is another OCR engine suggested and utilized in the literature. It is published under the Apache License 2.0. A German Artificial Intelligence research center called the Kaiserslautern Center implemented and suggested the OCRopus Engine. Today this system is maintained by Google. OCRopus is written in two different

programming languages which are C++ and Python programming languages. For the purpose of text recognition, two different OCR engine are used by OCRopus. Earlier, before the publication of the 0.4 OCRopus version, the engine used the Tesseract OCR engine during the recognition phase. Later, until this writing, it uses a neural network during the recognition phase. Developers of OCRopus still improve the engine for better recognition of text. It has been acknowledged that the accuracy of OCRopus is lower than the accuracy of the Tesseract OCR engine [56].

For achieving benefits from the usage of the OCRopus engine, there are some specifications of machines recommended in the literature for better applying OCRopus engine on images, the most important specifications are installing OCRopus on the Ubuntu OS, a memory greater than 4 Gbytes and a speedy CPU [57]. Accurate recognition of text cannot be achieved with OCRopus engine on the images contained pictures and multicolumn text lines. The study also rejected utilizing OCRopus for handling the problem of applying OCR on receipt images. The main reason behind this decision is because OCRopus is not a well-documented OCR engine and in the literature it is not much utilized by researchers for handling different problems associated with OCR.

2.4.4. Tesseract OCR engine

The well-known and most utilized OCR engine in the literature is the Tesseract OCR engine [58]. This engine has performed successfully and efficiently in several different areas associated with OCR. The Tesseract OCR engine was designed and implemented by R. Smith when he was a Ph.D. researcher in Bristol in November 1987. Investigations and examinations for improving the accuracy of Tesseract were carried out until 1995. After 1995 the investigations for improving Tesseract were postponed until 2005. Later in 2005, HP published the Tesseract as an open source engine and it was published under the Apache License 2.0. Now, the Google Company supports and maintains the Tesseract OCR engine. In February 2016, Google published a new version of Tesseract named 3.04.01 Tesseract version. The Tesseract OCR engine can only be utilized across command line interface because the Tesseract OCR engine does not have GUI (graphical user interface). For utilizing Tesseract in any application associated with OCR, researchers have to implement a GUI. Tesseract is written in two different powerful programming languages which are C++ and C programming languages. The Tesseract OCR engine has API for both Android and

IOS programming languages.

The Tesseract OCR engine is the most utilized OCR engine among various OCR engines available in the literature. Earlier versions of Tesseract only supported tiff format of images such as version 2.0 and older versions. Also these versions can only recognize text on documents that have one simple column of text. Earlier versions of Tesseract also have less efficient results on the documents contains pictures and shapes, because document layout analysis techniques are not supported by such versions, but recent versions of Tesseract supports document layout analysis technique and contains various page segmentation methods such as 3.0 Tesseract version.

Originally, the Tesseract OCR engine was trained for different shapes of English characters in a comprehensive way. The Tesseract OCR engine was trained for 8 different text fonts with 3 different text styles for each font (italic, bold and normal), and for each of characters from 94 characters, and 20 sample images utilized for training purposes. This means in total the Tesseract is trained by 60160 image samples. Tesseract has been adopted to handle problems in various areas associated with OCR. One of the usages of using Tesseract is recognizing symbols on the documents [59] because Tesseract can be trained for new fonts, new language and new shapes. The main and most important feature of Tesseract is that new languages or new fonts can be recognized by Tesseract after training Tesseract for these new fonts or new languages. Currently, Tesseract can recognize text on the documents in most world languages. It has been designed to recognize more than 100 different languages.

This research used the Tesseract OCR engine for recognizing text on the receipt images. Among several different OCR engines, finally, the research found that Tesseract fulfills the requirements for handling the problem of applying OCR on receipt images efficiently and accurately. There are several different reasons and factors behind this decision, some of these factors are listed below:

- A comprehensive documentation of Tesseract OCR engine is proposed in the official web page of tesseract [60].
- New languages or new fonts can be introduced to the Tesseract, means Tesseract OCR engine is a trainable OCR engine [61].

- Various problems in the literature have been handled efficiently and successfully by applying Tesseract OCR engine [4, 62, 63, 64].
- Tesseract can recognize text on different documents accurately, recently in average 99 % is reported as the rate of recognition by Tesseract on several different images contain text [58].
- Tesseract OCR engine have API for both Android and IOS programming languages.
- Currently, Tesseract can recognize text on the documents in most world languages. More than one hundred different languages are supported by the Tesseract OCR engine.
- The founder of the Tesseract (R.Smith) suggested and proposed a comprehensive paper about Tesseract [65] in 2007, where all the details about Tesseract were shown and presented such as Tesseract's definition, Tesseract's architecture and techniques.

The Tesseract OCR engine is implemented and designed by using several different techniques and algorithms in two phases. These phases are the connected component analysis phase and the recognition phase. In the first phase, Tesseract applies some techniques related to the connected component analysis. These techniques are fixing broken or joined characters, word segmentation, baseline fitting and line finding. In the second phase which is the recognition phase, two different classifiers called respectively a static classifier and an adaptive classifier for rising recognition accuracy are used. In the recognition phase, Tesseract uses two different techniques or algorithms for classifying characters and words on the images. At the first stage, the system uses a static classifier algorithm and in the second stage, it applies an adaptive classifier for better recognition of characters and words on the images that was not recognized by the static classifier. The step-by-step of architecture of Tesseract OCR engine are shown in the Figure 2.14.

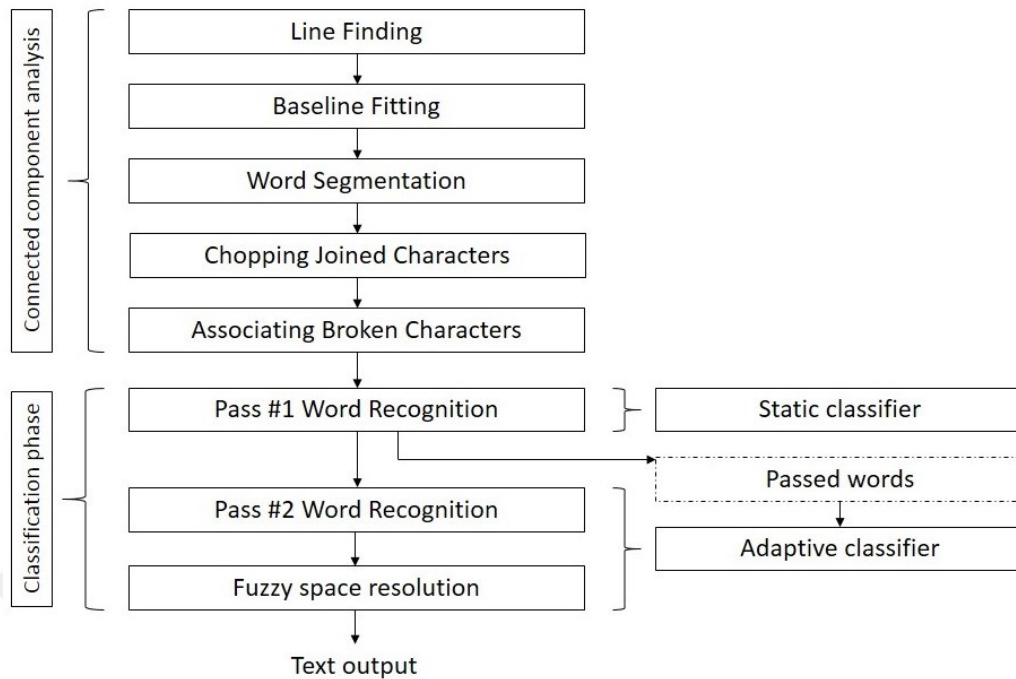


Figure 2.14 Step-by-steps of processes adopted by Tesseract OCR engine.

The architecture or processes adopted by Tesseract for text recognition is discussed and presented by the designer of Tesseract OCR engine in [65]. In the following subsections, all the processes adopted by Tesseract are shown shortly:

A. Baselines finding

The most important act in this process is spot filtering and line building or construction. A spot can be any content in an image like a symbol, character or word. See Figure 2.15 which is an example of baseline finding. This process will calculate average height of the spots and will remove noise and small speckles on the images.



Figure 2.15 An image contains 7 spots and baselines were identified by the baseline finding technique.

This process reduces noise and incorrect assigning baselines in the situation when the skewing problem of text exists. After finding all the spots on an image, then a technique

will be used by Tesseract for finding baselines called least median of squares fit.

B. Fitting of baselines

After identifying and finding all the spots on an image and then finding baselines, then Tesseract will apply another technique for fitting baselines. The technique that will be applied is called quadratic. This technique will fit baselines more accurately.

C. Segmentation of words

This is the process of segmenting words to the characters. Based on the character's pitches, Tesseract will apply a technique for segmenting words to characters. The words that contain characters with the same width (fixed pitch) will be assigned directly to the recognition phase without any problems. But in most cases, characters in words have different widths which pose a challenge to Tesseract for word segmentation. An example of a word that has different pitches is presented in Figure 2.16.

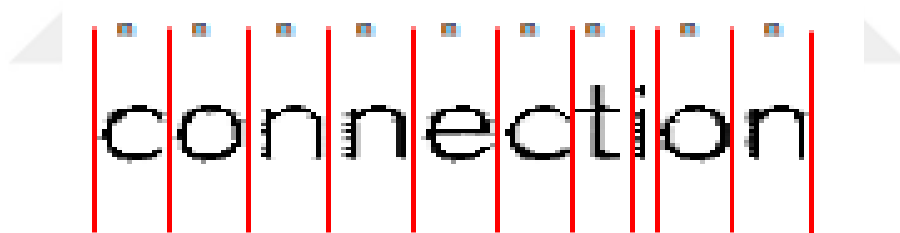


Figure 2.16 An example of a word that has different character pitches.

For handling the problem of different character pitches, Tesseract uses a technique. First, the process will find the unusual pitches in a character by calculating spaces in the limited vertical range between the mean line and baseline. When the unusual spaces have been found, then Tesseract will assign the space as a fuzzy space. The characters that have the problem of fuzzy space will be recognized in the second classification stage.

D. Chopping joined characters

Another important process that will be applied by Tesseract to the spots on an image is chopping joined characters in the words. In this situation, Tesseract will solve and fix

problems encountered during the segmentation of characters. For that purpose Tesseract will apply a technique that first searches for and finds the candidate chop point. See Figure 2.17 [65] which shows an example of joined characters in a word.

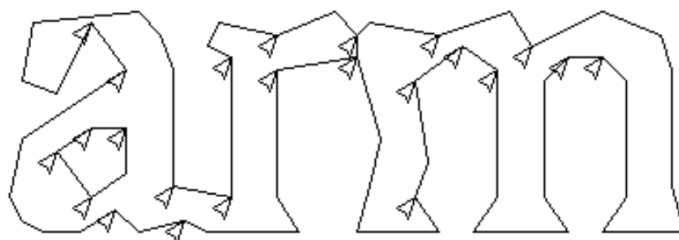


Figure 2.17 An example of a word with joined characters.

If this process cannot improve the accuracy of the output, then Tesseract OCR engine will not discard the word to be recognized by the classifier. After this process, the system will apply another process called merging broken characters to a word.

E. Merging broken characters

In this case, if candidate chops in a spot are tested and the spots are not recognized by the classifier, then Tesseract will feed the spots to a process called merging broken characters technique. These techniques will try various chop candidates from spots and will merge broken characters as much as possible. An example of broken characters in a word can be seen in Figure 2.18 [65].

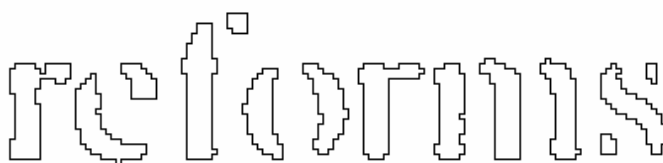


Figure 2.18 An example of broken characters in a word.

F. Character classification

After properly segmenting characters, then Tesseract will feed the character images into the classification phase. In the classification phase, Tesseract will classify and recognize characters by applying two classifier algorithms during two different stages. In the first stage, Tesseract will apply a static classifier algorithm and in the second stage, it will apply

an adaptive classifier algorithm. For the purpose of matching many-to-one to a source from the training information, various size fixed length features will be identified in the first stage. Sometimes characters might have weak information to be extracted and recognized by the classifier. In these cases Tesseract will use polygonal approximation for recognizing characters that have weak information. See Figure 2.19 [60].



Figure 2.19 Broken and joined characters in a word can be recognized by static classifier algorithm.

The adaptive classifier in the second stage will be successfully trained by the characters recognized in the first stage by the static classifier. The information that has been gathered from the static classifier algorithm is essential and important information for the adaptive classifier in the second stage. Also, the characters that have the problem of fuzzy space will be recognized by applying the adaptive classifier algorithm in the second stage.

3. PROPOSED TECHNIQUES

The previous chapter covered a general introduction and background about optical character recognition technology, where the steps and techniques are mentioned that requires for designing successive OCR system. This chapter covers and discusses important techniques and algorithms used in this research to overcome the problem of performing optical character recognition technology on receipt images. Where the general system overview is discussed and then the practical side of the system and screenshots of the suggested Android application are shown and presented.

3.1. System Overview

This section discusses and presents important techniques and algorithms used in this research for overcoming or handling the issue of applying OCR to receipt images. The main steps such as receipt region detection, image enhancement and preprocessing, applying Tesseract OCR engine, regular expression and database storing are reviewed. The following activity diagram which is presented in Figure 3.1 is the illustration and step-by-step process for applying several different methods and techniques of performing optical character recognition technology on receipt images.

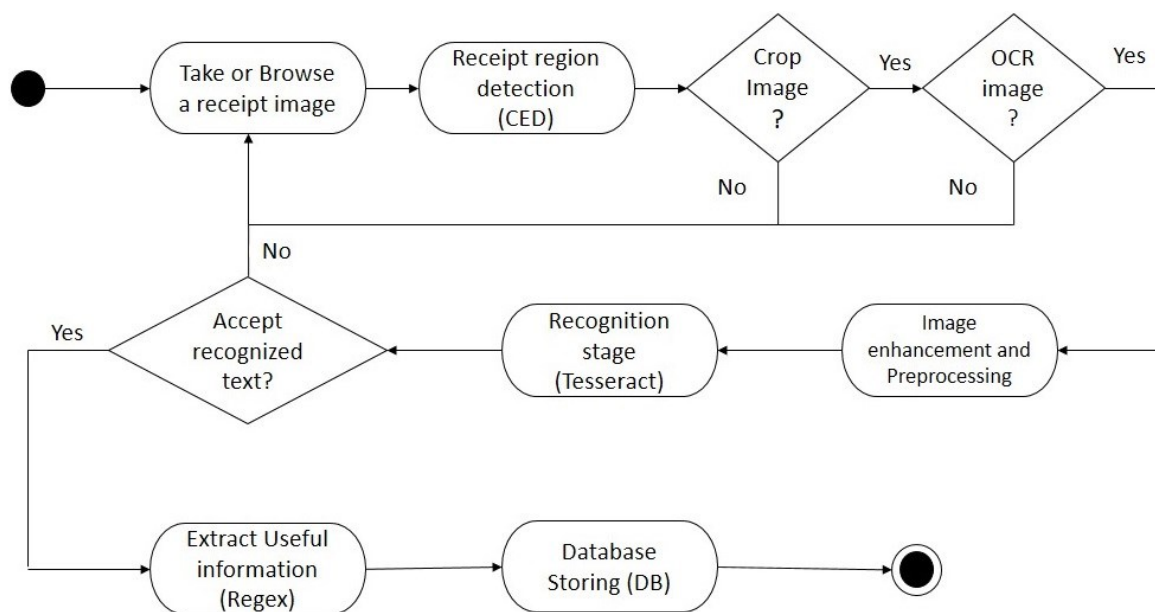


Figure 3.1 Step-by-step proposed techniques for the implemented receipt tracker application.

The prior work in the proposed system is first users have to capture a receipt image or browse to a receipt images from the mobile's gallery. After the users of the system selected an image of receipt then the system will apply canny edge detection (CED) algorithm for the detection and extraction of the region of receipt from the background of images. Then users can accept or reject the detection process by using the canny edge detection algorithm.

Following the act of detecting region of receipt by Canny edge detection algorithm, then the suggested system applies a series of techniques and methods for enhancing and smoothing images before feeding these images into the Tesseract OCR engine for improving recognition rate. The techniques applied in this thesis are contrast process, gray-mode operation, Binarization, reduction of noise, morphological operations and finally de-skewing process.

Now that the receipt image is enhanced and smoothed, it is ready to go through the recognition stage. In the recognition stage, the suggested Android application adopted and applied the Tesseract OCR engine for extracting and recognizing text on the receipt images. The research found several different techniques for improving and raising recognition rate by Tesseract OCR engine. The improvements include such things as training Tesseract OCR engine for the new fonts, improving by regex (regular expression), selecting suitable page segmentation method (PSM) and improving by introducing a new dictionary to the Tesseract OCR engine.

After the process of recognition, then the suggested Android application shows the recognized text to the user to either accept or reject. If the users of the application accept the recognized text, then the system will extract and recognize all relevant information such as the market name, address, and goods purchased. After identifying the useful information, finally the suggested system will save that information to a database for future use.

The Following subsections are the full description and illustration of the main techniques and algorithms adopted and applied in this research. The following subsections illustrate the step-by-step of performing algorithms and methods on the receipt images by presenting the effects of the suggested techniques.

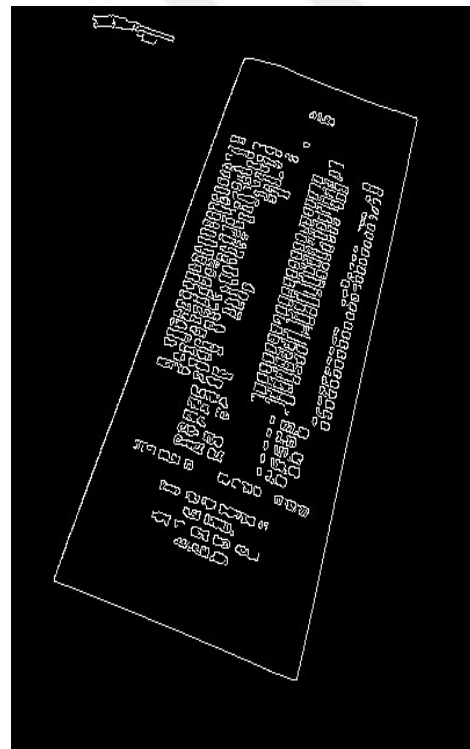
3.1.1. Receipt region detection

The prior act in the suggested Android application is the identification and extraction region of receipt from the background of the image by using the well-known and most utilized edge detection algorithm which is canny edge detection algorithm. In this way, image preprocessing algorithms can enhance the image better than by applying the image preprocessing algorithms to the whole image of the receipt with the background.

In the literature, researchers used and applied various edge detection algorithms for the different problems related to the machine learning. The edge detection algorithms such as Prewitt's operator, algorithm of canny edge detection, Laplacian of Gaussian, operator of Robert's cross, and operator of Sobel were used. In this research, the canny edge detection algorithm for extracting and identifying the region of receipt from background of the images was used. This is because after investigating several different research studies it was concl that the canny edge detection algorithm performs better for this case. The full description of what the canny edge detection algorithm is and how it works is discussed in the previous chapter.



a) An image of receipt



b) Canny edge detection algorithm effects

Figure 3.2 The result of performing CED algorithm on a receipt image.

In the proposed Android application from the OpenCV library, cvCanny function was used for applying and performing canny edge detection algorithm on the images of receipts. The effects of applying Canny edge detection algorithm on the images of receipt are clearly explained and shown in Figure 3.2.

The Canny edge detection algorithm applies a series of methods for identifying edges of an image. As the first process, the algorithm performs a noise reduction algorithm known as the Gaussian algorithm. Following this process, the algorithm calculates gradient intensity of the images then canny edge detection performs another process which is Non-maximum suppression process, by applying this process the algorithm will remove unwanted pixels. Finally, it will apply thresholding techniques to the images by utilizing two different threshold value.

After applying cvCanny function, then another function from OpenCV library was used for detecting different shapes (contours) on the image called cvFindContours function. After finding different shapes on the image, then the technique will choose the biggest shape which represents the receipt region by using another important function from OpenCV library called cvBoundingRect function as presented in Figure 3.3.

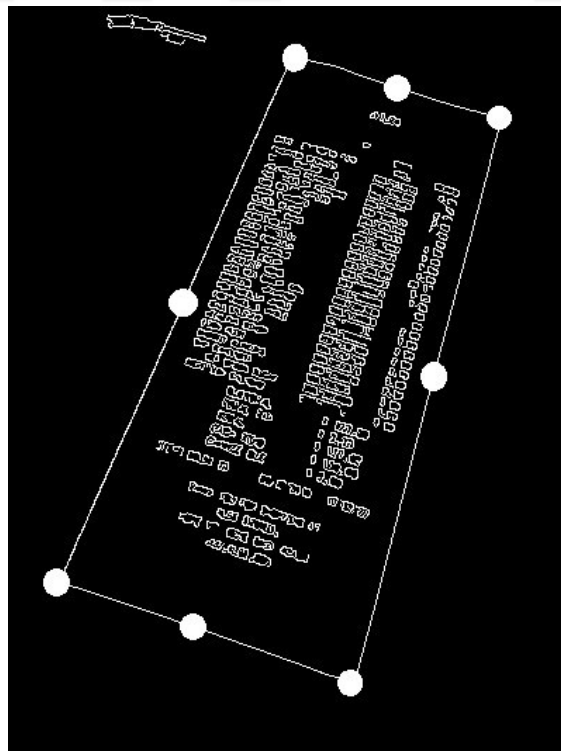


Figure 3.3 receipt region shape detection.

Now all the processes related to the receipt region detection is finished. At this point the proposed Android application gives the authority to the users of the application to do one of the following points:

- In some cases when images of receipts have a lot of noise, edge detection algorithms cannot perform well. If something happens like this, then application users can just reject the process and resubmit another image of the receipt.
- The proposed android application gave the authority to the app users to edit the detection or manually select the region of receipt.
- However, if there is no problem with the detection of receipt region and canny edge detection performed well, then app users can click on the crop button and submit the image.

The result of the all the functions discussed are presented in Figure 3.4. The first image is the original receipt image captured by the users with the detection of receipt region and the second image is the receipt region image which is ready to go through further processing proposed in this research.



a) Original receipt image with the detection performed by CED



b) region of receipt is ready.

Figure 3.4 The receipt region is accurately identified and detected by using canny edge detection algorithm.

The canny edge detection algorithm have more affects such are correcting the skewing problem and correcting tilting problem on the images of receipts.

3.1.2. Receipt image pre-processing phase

Image preprocessing and enhancement is the initial process before feeding images to the OCR engine. This process reduces noise on the receipt images and raises character recognition. After identifying regions on the receipt images by using the Canny edge detection algorithm, the prior step is to applying a series of image preprocessing algorithms.

In this research, several different images preprocessing algorithms are studied in order to raise recognition rate by Tesseract OCR engine. The image preprocessing algorithms applied to the images of receipts are contrast technique, gray-scaling technique means changing images from color mode to the gray mode, binarizing technique by using global thresholding algorithm, noise reduction technique, morphological operations and de-skewing procedure of the text. The architecture of image preprocessing algorithms applied to the images of receipts in the suggested Android application is shown in Figure 3.5.

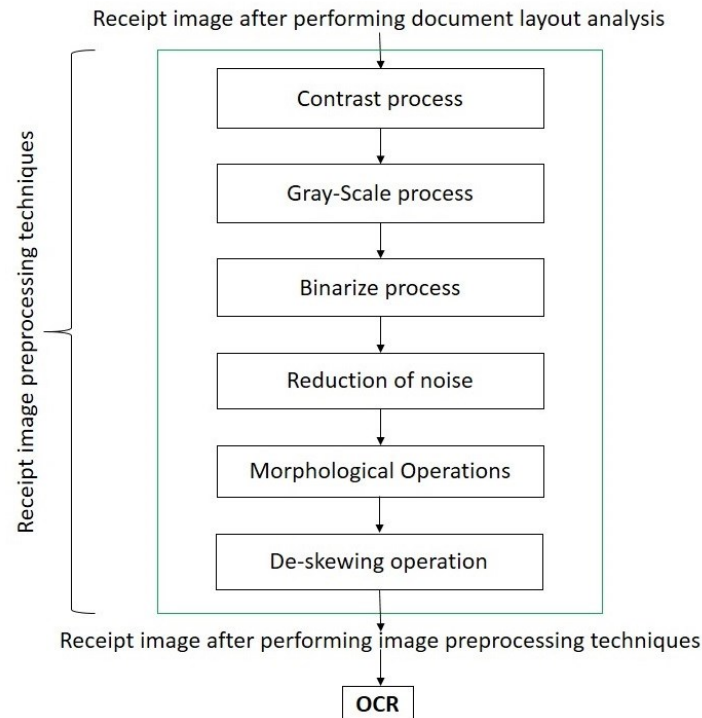
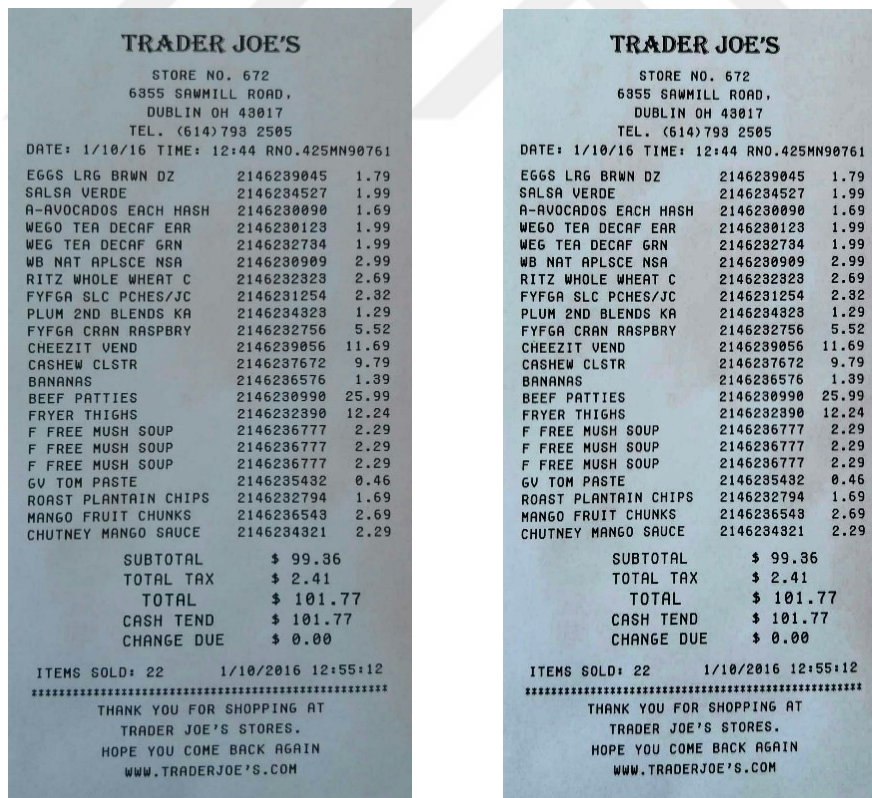


Figure 3.5 Image preprocessing algorithms adopted and applied in this research.

All of the mentioned image preprocessing algorithms applied and performed on the images of receipts by using an open source image processing library freely available called ImageMagick library [66]. ImageMagick library supports over 100 image formats and also it supports videos. Most of the image processing algorithms is supported by the ImageMagick library and includes all of the algorithms used in this research.

A. Contrasting

Contrasting receipt images before feeding receipt images to the other preprocessing algorithms is one of the processes for better applying other image preprocessing algorithms. Since images taken by mobile devices cameras suffer from light variation more than images obtained by the scanner, applying contrast algorithm as the initial process is preferable. For benefiting from the usage of contrast algorithm, a function from ImageMagick library was used called (-contrast). The result of applying contrast method on a receipt image is clearly shown and presented in Figure 3.6.



a) An image of receipt

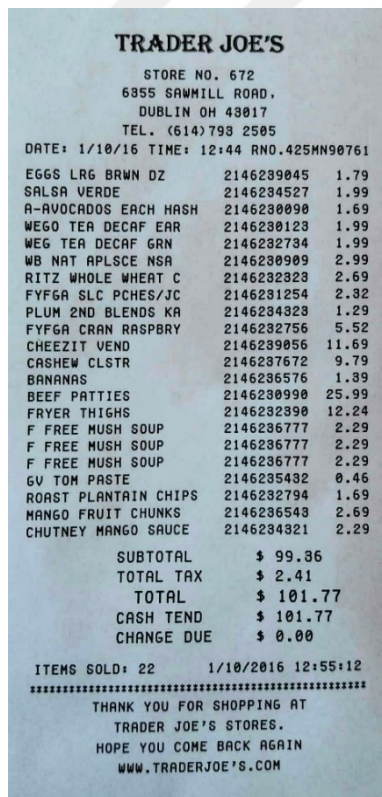
b) Effects of contrast function

Figure 3.6 Result of performing contrast method on an image of the receipt.

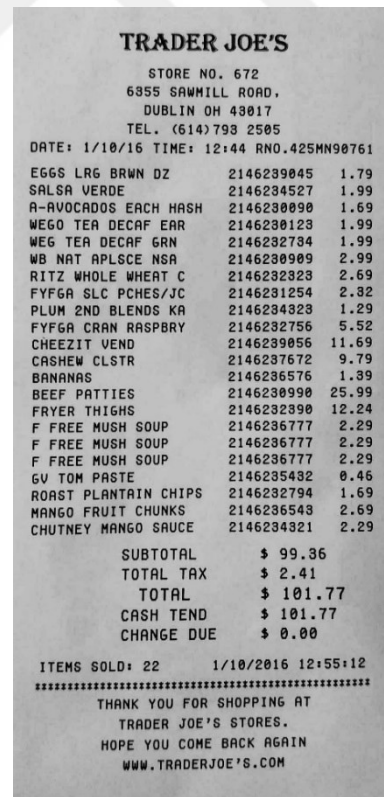
B. Gray-scaling

The gray-scaling process is the process of conversion an image from color mode to the gray mode. After performing contrast function on receipt images, the second process is applying gray-scaling function on the receipt images. Since performing binarization process on the colored images takes more time than on the gray images, therefore the main important point is to apply gray scale method on the images of receipts before applying binarization function.

For benefiting from the usage of gray-scaling function, another function from ImageMagick library was used called (-colorspace gray) function. The results of applying or performing gray-scaling function on an image of receipt are clearly shown and presented in Figure 3.7. It clearly appears that contrasted images have color variations, but after performing the gray-scaling function on the contrasted image, the color variation is eliminated and converted to gray mode. This process helps the proposed system efficiently perform the binarization process.



a) An image of receipt is contrasted



b) An image of receipt is gray-scaled

Figure 3.7 The result of performing gray-scale function on a receipt image.

C. Binarization

The binarizing process is the process of thresholding images before feeding or submitting them into the OCR engine. The Binarization process is an important process that must be performed in any application related to the optical character recognition technology. To learn the OCR engine to only detect and process texts on the images, the binarization process must be applied before feeding images to the OCR engine. The Binarization process simplifies segmentation of characters on the images. The process of binarization separates pixels in an image, and marks the image background with a white pixel and marks the text on the image as a black pixel.

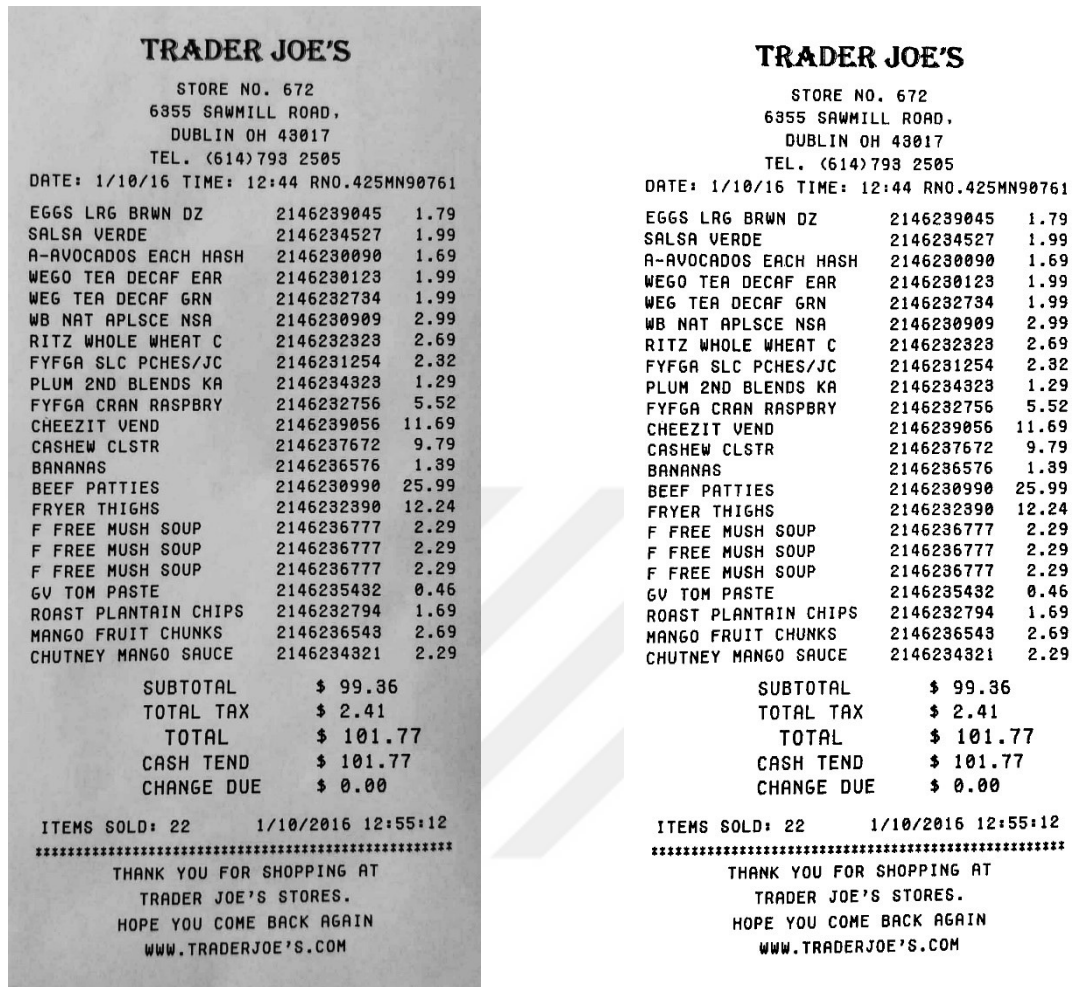
Researchers in general used thresholding algorithms for binarizing images for several different problems related to the optical character recognition technology before submitting images to the OCR engines. There are two well-known and most utilized thresholding algorithms used by researchers. These are local thresholding algorithm and global thresholding algorithm. The Local thresholding utilizes a unique threshold value for each pixel on the images, whereas the global thresholding algorithm utilizes a threshold value for the entire pixels that exist on the images.

Improving or better-applying the segmentation process and recognition process are the factors for performing thresholding algorithm in the suggested Android application. Handling the problem of uneven lighting or light variation on images are another important usage of the thresholding technique. Since local thresholding utilizes a unique threshold value for every pixel on the images which takes considerable time, due to that purpose, global thresholding is selected, based on results and findings, the study notified that suggested Android application can process receipt images more efficiently when global thresholding algorithm used rather than using local thresholding algorithm.

For applying global thresholding algorithm on the receipt images another function from ImageMagick library is used called (-threshold *value* %) function. Several different threshold values are tested, finally based on investigations and tests, the research found that 40% as the threshold value performs better than other values.

The effects of the binarization process by using global thresholding algorithm on a receipt image is clearly shown and presented in Figure 3.8. Contrast and gray-scaling algorithm are the two functions performed on the first image and the binarization process

produced the second image in which the text on the second image clearly appears.



a) A receipt image contrasted and gray-scaled

b) An image of receipt in which binarized

Figure 3.8 The result of performing thresholding algorithm over an image of the receipt.

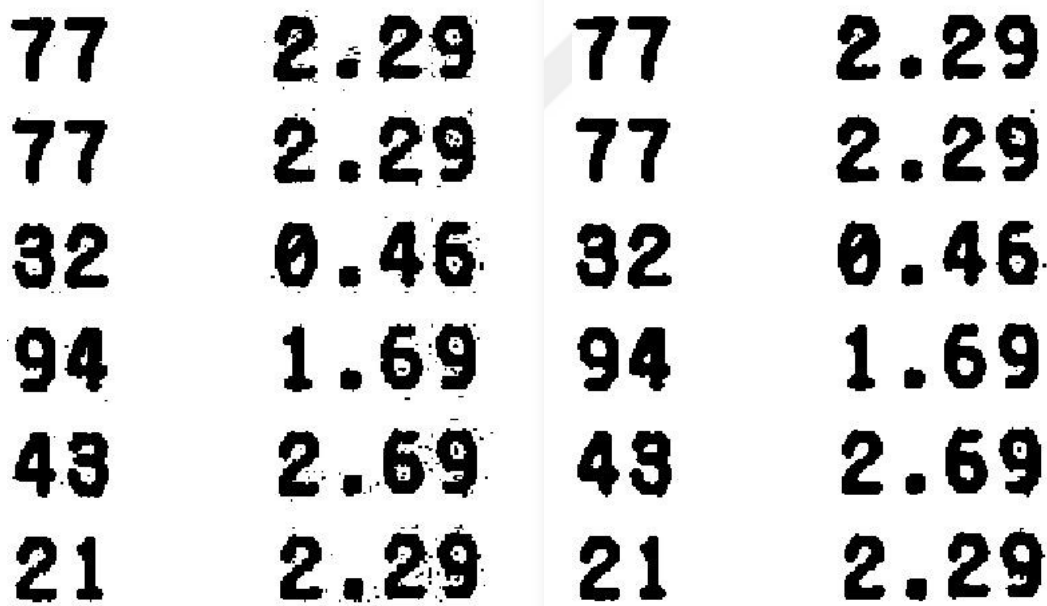
D. Noise reduction

Today's mobile device's cameras are capable of capturing images in high resolution, but there is still some noise and small speckles should be eliminated before feeding images to the feature extraction process. Because directly submitting images to the feature extraction phase without removing or eliminating noises, classification phase will produce unpromising results.

Applying an algorithm for the purpose of eliminating or removing unwanted information or noises on the binarized image is the best way and most comprehensive process for better

applying the feature extraction phase and classification phase in the systems related to the optical character recognition technology. Median filters and Gaussian blur filters are the two well-known and comprehensive algorithms used by researchers to overcome the problem of having noises on the images. For removing noises or unwanted information on the receipt images, we followed utilizing Median filters. In an image, Median filter takes an area (3x3, 5x5...etc.) and produces a list of value from the lowest value to the highest value, then it will select the median value from that list and finally it will replace or change the value of the center pixel that was taken by median filter to the median value.

ImageMagick library has a function named (-median *radius*) function, the study followed applying these function to the receipt images for the purpose of eliminating noises or unwanted information from binarized images. The radius value is (3), because based on the results and experiments obtained, it was found that median filter applies better when the value of radius is (3). The result of applying median filter algorithm on an image of receipt is clearly shown and presented in Figure 3.9.



a) Noise or small speckles around the text.

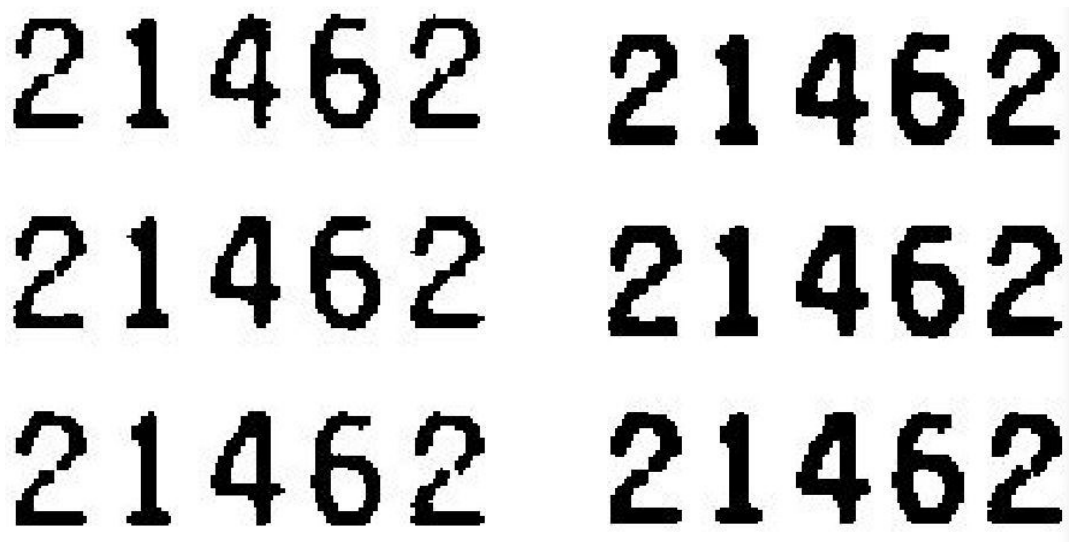
b) Most of the noises are eliminated.

Figure 3.9 Results of performing Median filter on an image of the receipt.

E. Morphological operations

Performing various image preprocessing such as contrast function, gray-scale function and binarization process might produce a hole to some significant part of the images. Having such holes on the characters makes segmenting characters or words hard in the segmentation phase because a hole might separate a character or a word to two characters or two words. For solving such problems on the binarized image, the morphological operation is the best choice.

The two well-known and most utilized algorithms of morphological operations are the Dilation and Erosion algorithm. Erosion algorithm is the process of attaching or adding black pixels to the edges of characters on a binarized image, whereas Dilation is the process of attaching or adding white pixels to the edges of characters on a binarized image. Applying various image processing algorithms on receipt images will add some holes to the characters, therefore for adding black pixels, Erosion algorithm has been applied on the images of receipts in the proposed Android application. For utilizing Erosion algorithm on the receipt images, we utilized a function from ImageMagick library which is (-morphology *Erode Rectangle: 1x1*). The function will attach black pixels to edges of characters in the binarized image by utilizing a rectangle with the radius of 1x1. The result of performing Erosion algorithm on a binarized image is clearly shown and presented in Figure 3.10.



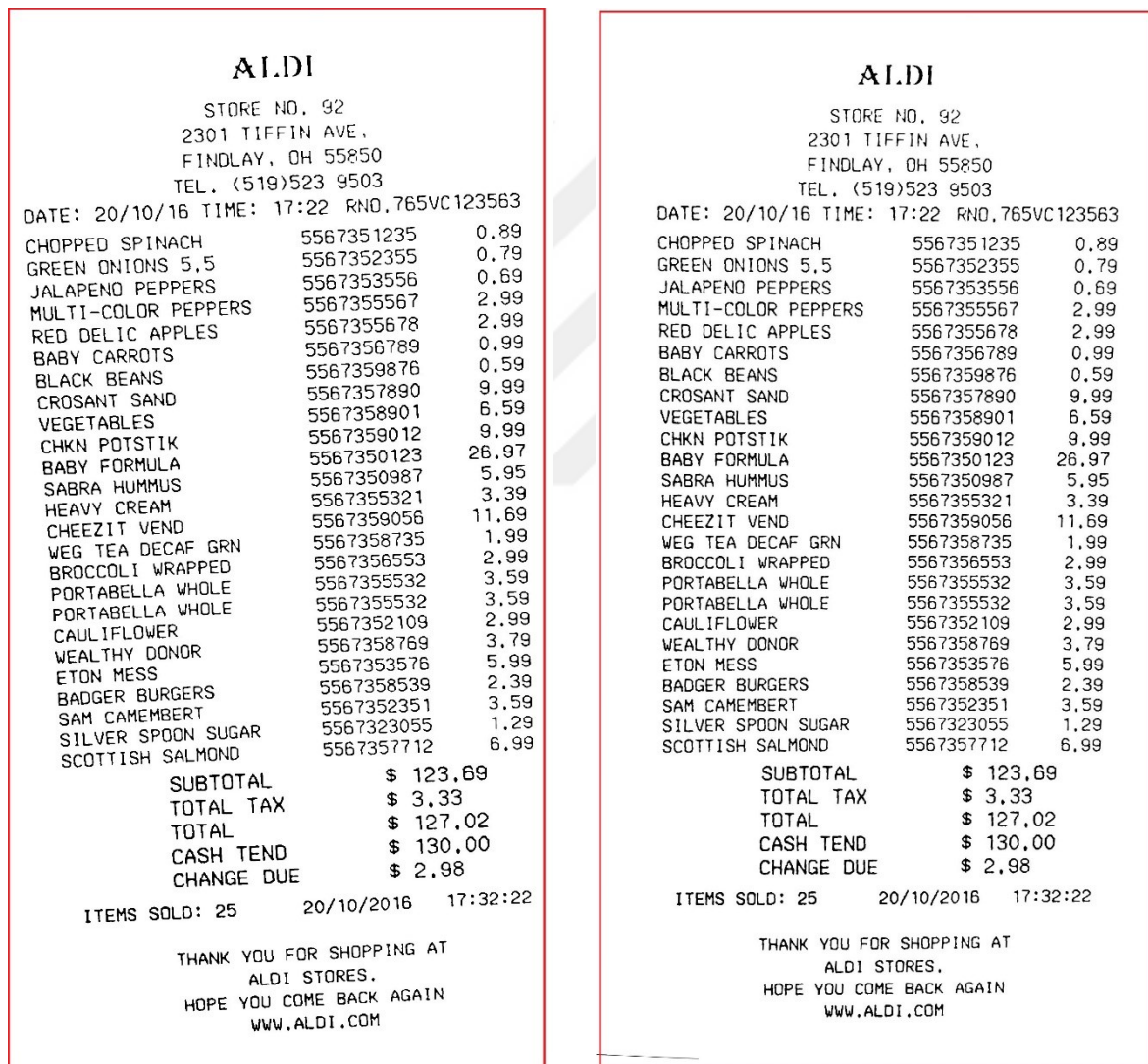
a) Holes are clear on some characters.

b) Applying Erosion algorithm on a binarized image.

Figure 3.10 Adding black pixels to the holes of characters by using Erosion algorithm.

F. De-Skewing process

The De-skewing process is also one of the main processes for raising recognition rates. Directly submitting skewed images into the OCR engines has less than satisfactory results, therefore the problem of skewing text on the images must be handled before feeding into the OCR engines. Since today's innovations are focused more on creating mobile applications, capturing images through mobile device's cameras suffers skewing of text more than images obtained by scanners.



a) An image of receipt in which skewed.

b) An image of receipt in which de-skewed.

Figure 3.11 Handling skewing problem on an image of receipt by using a de-skewing function.

Two different methods of skewing text were applied to receipt images using the Android application. The first method simplifies the skewing problem by using the canny edge detection algorithm. Since the canny edge detection algorithm cannot correct the skewing problem with one hundred percent accuracy, a second option must be used from the ImageMagick library which is the `(-deskew threshold {%f})` function. Based on the recommendations of ImageMagick [67] and results of the application, the study found that (40%) as the threshold value for the deskew function applies very well. The results of performing deskew function from ImageMagick library on an image of receipt that has the problem of skewing is clearly shown and presented in the Figure 3.11.

3.1.3. Recognition phase

The suggested Android application extracts region of receipt from background of receipt image by using canny edge detection algorithm, following, the proposed system will apply a series of image processing algorithms. In the recognition phase, the application produces a text file by using an OCR engine called Tesseract OCR engine.

In the previous chapter, a detail description is showed about two different approaches used by researchers for implementing applications associated with the optical character recognition technology. The two different approaches are conventional step-by-step of recognition techniques (segmentation phase, feature extraction phase and classification phase) and utilizing OCR engines. This research followed utilizing Tesseract OCR engine [58]. Introduction about Tesseract, architecture or the structure of Tesseract and the main important advantages of Tesseract are discussed and presented in the previous chapter.

Tesseract OCR engine is an open source engine used by researchers to overcome or handle various issues associated with optical character recognition technology. It is programmed by utilizing two programming languages which are C and C++ programming languages. For other programming languages like PHP, JAVA...etc. it has a wrapper. The Tesseract OCR engine can only be utilized across command line interface because Tesseract OCR engines do not have GUI (graphical user interface). For utilizing Tesseract in any application associated with OCR, researchers have to implement a GUI.

For the researchers who want to implement applications associated with OCR, Tesseract OCR engine offers various options and parameters to fulfill the requirements associated

with their applications. From the official website of Tesseract OCR engine an article [68] can be obtained that contains the various options and parameters available to researchers. Various options and parameters offered by Tesseract OCR engine were investigated, and finally the research found that managing some of the option and techniques are helpful for improving OCR accuracy on the receipt images. The techniques that improved and raised recognition rate on the receipt images in the suggested Android application is shown and discussed in the following subsections.

In the Windows environment, the Tesseract OCR engine can only be utilized across command line prompt, a simple performing Tesseract OCR engine on a receipt image is presented below:

```
Tesseract receiptImage.jpg outputText
```

Above example is an example of performing Tesseract OCR engine on a receipt image named receiptImage.jpg in Windows environment across command line interface. By default, Tesseract expects English language on the image. But for example, if we want to tell Tesseract that the image contains the text written in Turkish language, then we must provide this information to the Tesseract OCR engine as bellow:

```
Tesseract receiptImage.jpg outputText -l tur
```

Now, Tesseract knows that the receipt image named receiptImage.jpg contains the text and the text is written in the Turkish language. As a result, it compares with Turkish trained data.

Tesseract OCR engine is trained to detect and recognize a number of different fonts. There are some fonts that Tesseract OCR is not able to identify. Moreover, the Tesseract Official web page published a sequence of techniques for raising recognition rate by Tesseract OCR engine. The Study found that some of the techniques are helpful for raising recognition rate by Tesseract on the receipt images.

The essential and important techniques that have been performed on the receipt images for raising recognition rate by Tesseract are first applying Canny edge detection algorithm for identifying and extracting region of receipt from background of receipt images and

second filtering region of receipt by applying a series of image processing algorithms. Behind these two important techniques, there are some techniques related to the Tesseract that can greatly raise recognition rate such as introducing new fonts to the Tesseract OCR engine, introducing new dictionary to the Tesseract OCR engine, selecting a suitable page segmentation method and finally Tesseract accuracy improving by regex functions. Each of these techniques is discussed in detail in the following subsections.

A. Tesseract OCR engine training

Tesseract OCR engine provides the opportunity to introduce Tesseract for new fonts or the font that Tesseract is unfamiliar with. In this way, the researchers who want to implement or design applications associated with the OCR can easily apply Tesseract to the documents with the functionality of training for the new languages or new fonts. Two Unusual text fonts are tested to be recognized by Tesseract, the fonts are fake receipt font [18] and merchant copy font [19]. For the purpose of Tesseract training for the new fonts, a step-by-step article provided by Tesseract official webpage [69] are implemented. The step-by-step guideline provided by Tesseract official web page are discussed in the following subsections.

1. Preparing images for training

For creating training files, Tesseract OCR engine accepts images in tiff format only. The images used for training Tesseract OCR engine must be obtained by one of the following factors:

- The images can be obtained through a mobile device's camera.
- The training images can be obtained by converting a text file to an image electronically.
- The training images can be obtained by a scanner.

Some receipt images are captured by a hand-held device and submitted to the Tesseract OCR engine as the training image files. After capturing the images of receipt by a hand-held device, then manually the images are enhanced and preprocessed to be suitable for training procedure. Then the images are manually converted to the tiff format because

generating training files by Tesseract only accepts images in tiff format. For generating training file, six receipt images are utilized, the text on the three of the images has been written in fake receipt font and others have been written in merchant copy font. The receipt images contain up to 1000 characters and written in the English language. The following image is an example of a receipt image that was used as a training image for generating a training file by Tesseract OCR engine. See Figure 3.12.

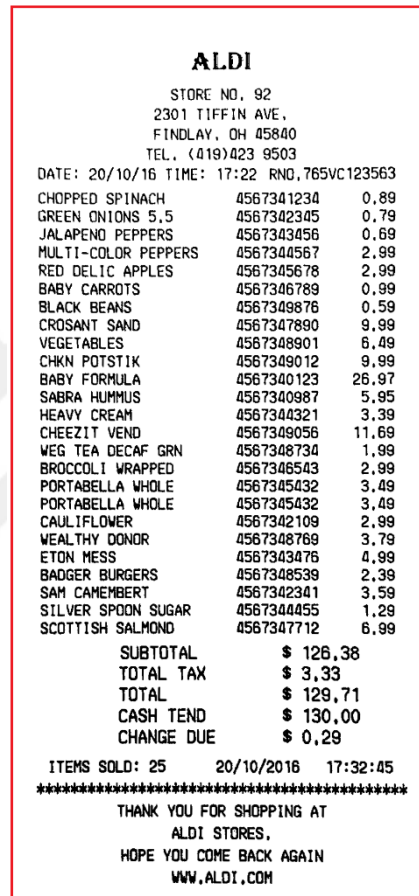


Figure 3.12 An example of training image was used for generating training file by Tesseract.

2. Box file generating and editing

During the process of training or introducing the Tesseract OCR engine for the new fonts, Tesseract cannot treat with the images directly, instead, it treats with the box files to train itself for the new fonts. The box file is a text file that contains the characters exist on the training image and it list every characters one per line with writing every character's

coordinates. Through command prompt in Windows environment, the following command line is utilized to generate box files on the receipt images:

```
>tesseract -l eng -psm 6 lan.fontname.exp0.tif lan.fontname.exp0 batch.nochop makebox
```

Above command line will generate a file called box file with .box extension. But in most cases the Tesseract cannot recognize or detect every character properly. Therefore every wrong recognitions of Tesseract should be corrected. JTessBoxEditor program [70] is utilized to edit box files and correcting wrong recognitions. JTessBoxEditor program is a program for training Tesseract or editing box files generated by Tesseract, it requires Java runtime environment (JRE) because it is programmed by Java programming language.

By using JTessBoxEditor program, every incorrect recognition of Tesseract must be corrected. This section of working is the highest time-consuming part, since for each image it is necessary to go through every character that exist on the images to correct wrong information. The JTessBoxEditor program provided different features such as box deleting, box adding, box merging and box splitting. A screenshot of the working of JTessBoxEditor program is shown in the Figure 3.13.

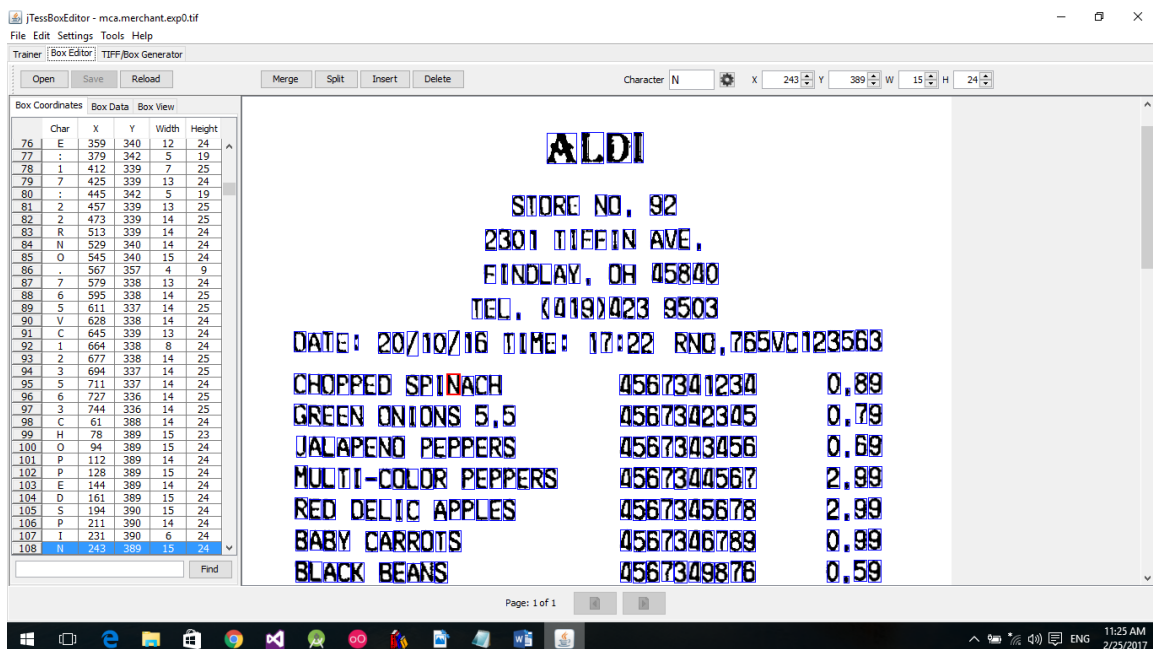


Figure 3.13 Editing a box file generated by Tesseract by using JTessBoxEditor software.

3. Tesseract running for training

Now it is time to run Tesseract for generating training file. Both box file and tiff image must be provided to the Tesseract for training itself for the new fonts. After running Tesseract for this purpose, then the Tesseract will generate a file called a training file. This file contains the information about training. For running Tesseract to generate training files the following command line is utilized in the command prompt in the Windows environment:

```
>tesseract -psm 6 lan.fontname.exp0.tif lan.fontname.exp0.box nobatch box.train
```

4. Preparing a file called Unicharset

For introducing Tesseract the possible characters that Tesseract can create as output, a file called Unicharset must be generated. For generating or creating the Unicharset file the following command line is utilized in the command prompt in the Windows environment:

```
>unicharset_extractor lan.fontname.exp0.box
```

5. Font properties setting

During adding a new font to Tesseract OCR engine, the font properties must be provided to the Tesseract OCR engine. The font properties are such as italic style, bold style and so forth. For that purpose, a text file must be provided to the Tesseract with font properties for each font per line. The line must be as follow:

Fra.fake.exp0.box 0 0 0 0 0. These line fulfils the following requirements:

 <italic> <bold> <fixed> <serif> <fraktur>

Since this study just focused on recognizing standard font without any font styles, therefore all the fields filled with 0.

6. Clustering

For creating prototypes of characters, we need to cluster all the alphabetical features of characters after all the features of characters has been extracted from the training images. For this purpose it is necessary to run two command lines in the command prompt. These prompts generate four files. The four files are pffmtable file, inttemp file, shapetable file and normproto file.

```
>mftraining -F font_properties -U unicharset -O lang.unicharset *.tr  
>cntraining *.tr
```

7. Creating trained data

The final step in the procedure of training Tesseract is to generate a file called traineddata. This file contains all the information Tesseract needs to recognize the new font when it will be tested by uploading new images. This step merges all the files created or generated in the previous steps (normproto, inttemp, unicharset and shapetable). But before merging all these files, the files must be renamed with a lang. prefix (like frb). For that purpose the following command lines were used.

```
>move inttemp lan.inttemp  
>move normproto lan.normproto  
>move pffmtable lan.pffmtable  
>move shapetable lan.shapetable
```

Following the above command line, then a command line must be run for producing a file called traineddata:

```
>combine_tessdata lan.
```

After running the above command line, then Tesseract will produce traineddata file.

This file must be stored in the tessdata directory, finally Tesseract can properly recognize the new fonts.

B. Page segmentation methods (psm)

The process that have a great impact on improving the accuracy of OCR applications is segmentation process since other processes are highly depends on the segmentation process such as feature extraction phase and classification phase. Breaking down images to several images or segmenting lines, words and characters from an image is a process called segmentation procedure. Researchers proposed several different segmentation techniques for segmenting characters, word and lines in various applications associated with OCR. One of the advantages that Tesseract provides to the researchers is proposing several different page segmentation methods. When researchers want to apply Tesseract on the applications associated with OCR, they have the opportunity to choose or select the most helpful segmentation method for their cases. By default, Tesseract expects a page full of text when it tries to recognize text on an image. Based on tests and findings, it has been observed that the sixth-page segmentation method is superior to other page segmentation methods on receipt images. Every page segmentation methods suggested by Tesseract was tested on the receipt images. Expecting a single block of text is performed by Sixth-page segmentation method. The list of ten-page segmentation methods suggested by Tesseract is shown and presented in Figure 3.14.

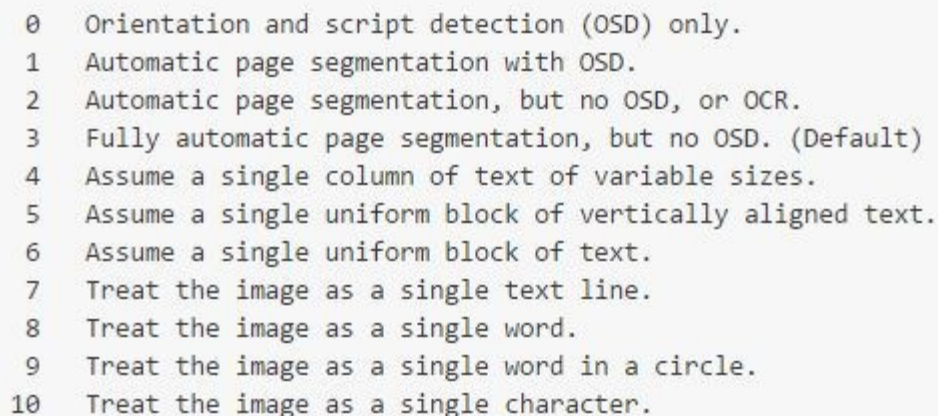
- 
- 0 Orientation and script detection (OSD) only.
 - 1 Automatic page segmentation with OSD.
 - 2 Automatic page segmentation, but no OSD, or OCR.
 - 3 Fully automatic page segmentation, but no OSD. (Default)
 - 4 Assume a single column of text of variable sizes.
 - 5 Assume a single uniform block of vertically aligned text.
 - 6 Assume a single uniform block of text.
 - 7 Treat the image as a single text line.
 - 8 Treat the image as a single word.
 - 9 Treat the image as a single word in a circle.
 - 10 Treat the image as a single character.

Figure 3.14 Tesseract proposed a list of ten-page segmentation methods.

C. Introducing a new dictionary to Tesseract

Providing a dictionary as post-processing mechanism is one of the procedures that improve OCR outputs. Applications associated with OCR will check the produced text by OCR engine with a dictionary to correct the words that were recognized wrongly. Tesseract OCR engine is one of the OCR engines that provided an English dictionary to detect words in the dictionary. But usually, for recognizing text on the receipt images, most of the words on the receipt images are not English words. For such issue, Tesseract provided the ability to the researchers to disable the dictionary provided by Tesseract and introduce a new dictionary to the Tesseract.

The default dictionary provided by Tesseract OCR engine are disabled by setting to false two configuration variables. The two configuration variables are (load_freq_dawg) and (load_system_dawg). Then a new dictionary is introduced and enabled to the Tesseract OCR engine which contained words exist on the receipt images. Based on tests and findings, the great impacts of such procedure are observed for improving recognition accuracy by Tesseract OCR engine.

D. Regular expression's improvements

Regular expression functions are the functions that have many advantages for several different fields. This study used functions of regular expression in two different ways. First, for extracting relevant information in the text produced by OCR engine, second for improving OCR results. We discussed the utilization of such mechanism by the following example:

```
if (preg_match('/([0-9]{1,4}[ ]*(\.,|_|-)[ ]*[0-9]{2}[ ]*)$/', $Lines-of-recognizedText))  
then, preg_replace('/(\.,|_|-)/', '.');
```

Tesseract OCR engine or let say most of the OCR system cannot differentiate some symbols from each other, for example, dot (.) symbol have similarity to (.,|_|-). If such situations happened in our application, the application will detect such wrong recognitions and will amend. in the above example, the application is learned to change the (.,|_|-) with

the dot (.) between numbers, because on the receipt images all the times the numbers such as total money spent have dot between numbers instead some other symbols such as (_|,|-).

3.1.4. Regular expression (Regex) phase

When all the phases related to extracting and recognizing text on the receipt images are finished, then the Tesseract will produce a text file contains recognized text on the receipt images. From the produced text file, the relevant information should be extracted and store to the database for the future uses. The relevant information are:

- Name of the market, the number of market, phone number of market, the website of market and address of market.
- Purchasing time and date.
- ID number of Receipt.
- Item name purchased, price of item purchased and ID number of item purchased.
- Information related to expending money such as subtotal money expended, Total money expended, total money paid for tax, total money paid for cash, change due money.

Regular expression mechanism is the procedure for identifying and getting a part from a text file by using a sequence of characters that each character represents a string to be detected in the text file. In this research, functions of regex were utilized to identify and extract above-mentioned information which are name of market, address and so on. Since regular expression functions provide the facility to parse and discuss the text, therefore it can be defined as a mini programming language. The most important expressions in regex were utilized for getting relevant information are shown and presented in table 3.1.

Table 3.1 The most important regex symbols and tier meaning that utilized in this research.

Symbols	Meaning
[A-Z]	From A (uppercase) to Z (uppercase) will be extracted by this symbol.
[0-9]	From 0 to 9 will be extracted by this symbol.
[a-z]	From a (lowercase) to z (lowercase) will be extracted by this symbol.
m+	Any string will extract that includes one m at least.
m?	any string includes zero m or more will be extracted by this symbol. The same before.
m*	any string includes zero m or more will be extracted by this symbol.
m{N}	Any string will extract that includes a series of N m.
[a-Z]	From a (lowercase) to Z (uppercase) will be extract by this symbol.
m\$	Any string has m at the end will be detect by this symbol.
m{2, }	Any string includes at least two m will be extracted by this symbol.
^m	Any string have m at the start of the string will be extracted by this symbol.
m{2,3}	Any string includes two or three m will be extract by this symbol.

OCR outputs vary from each other. Therefore searching or identifying a string of a text is a challenging task. It is necessary to collect every potential case to detect as much information as possible. One of the examples suggested with the Android application for identifying and finding total money expended for tax from the recognized text file is shown below.

```
preg_match('/(^([A-Za-z0-9]{5} [A-Za-z0-9]{3})[ ]+[$|s|5|[ ]+([0-9]{1,4}[ ]*(\.,|_|-)[ ]*[0-9]{2}[ ]*)$)|(T|TOTAL TAX)|(T0TAL TAX)|(TOTRL TAX)|(T0TRL TAX)|(TOTAL TRX)|(T0TAL TRX)|(TOTRL TRX)|(T0TRL TRX)/',$Recognized-Text)
```

The example in which presented in the above black box is one of the examples utilized in this research for extracting total money expended for tax in the recognized text file. It is clear that all the potential cases are defined for detecting total money expended for tax.

3.1.5. Database phase

Once every process from the stack of suggested techniques are finished, then the obtained significant data must be stored in a database for answering user's demands later by implementing queries. A database was designed for storing all the information extracted from the recognized text file. Three tables concluded the database designed for the suggested Android application. The first table will store data about the users of the application such as ID of user, name of user, email and password of user. The second table will store data related to the market in which users have purchased their goods from there and total money expended and also time and date of marketing. Primary key and foreign key is utilized to handle the relationship between all the three tables. The ER- diagram (entity relationship model) of the proposed database is shown in the Figure 3.15.

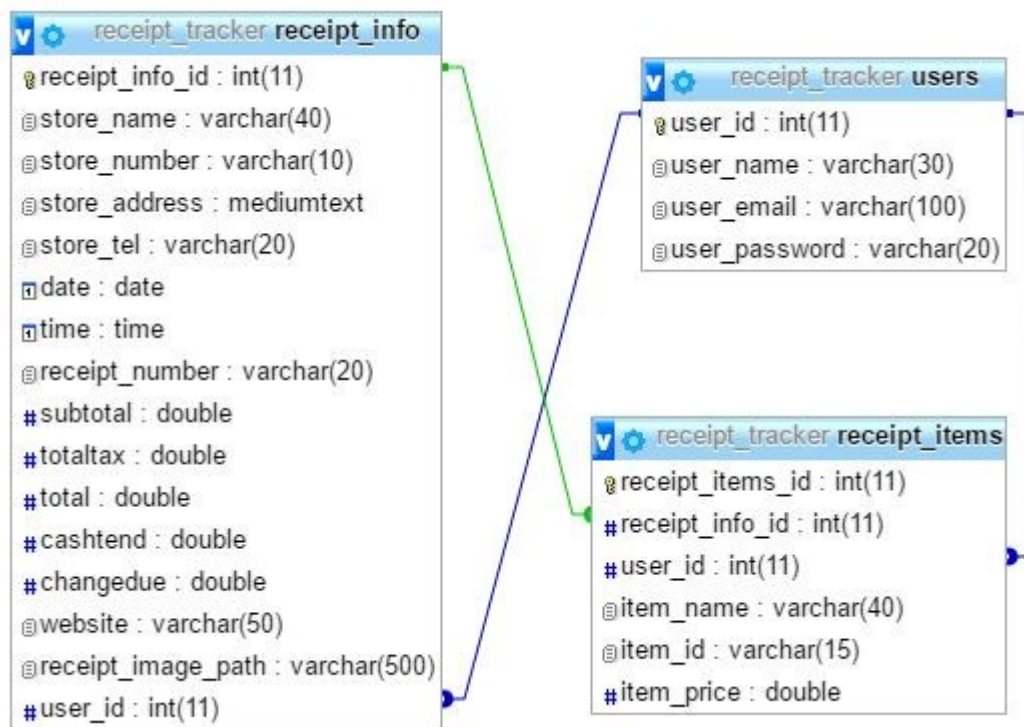


Figure 3.15 Proposed database ER-diagram in this research.

3.2. Implementation and Practical Work

An Android application was implemented that can extract the receipt region from the background of the receipt images and later the application will submit the image to the server side for text recognition. Client-server architecture was selected for the suggested

android application. The client mode will capture a receipt image and send the image to the server side. This means that the application will depend on the client and server architecture instead of performing all the processes in the client mode. After the image is uploaded to the server, then the system will apply all the producers related to recognizing and extracting text on the uploaded image. Architecture of client-server is chosen because of the following important points:

- Performing processes like image processing algorithms and OCR processes on the client mode will effect the performance of the application.
- Saving receipt images to the memory of mobile devices instead of saving to the server will consume a lot of the memory spaces of the mobile devices.

Java as Android programming language and PHP are the two programming languages that were utilized in this research to design an Android application for extracting and recognizing text on the receipt images. Android Studio [71] is utilized as an official and most utilized IDE for Android. For implementing features related to the client mode, Java as Android programming language is used. For implementing features related to the server side, PHP is used which is a server-side scripting language. Client-server architecture of the system is shown in Figure 3.16.

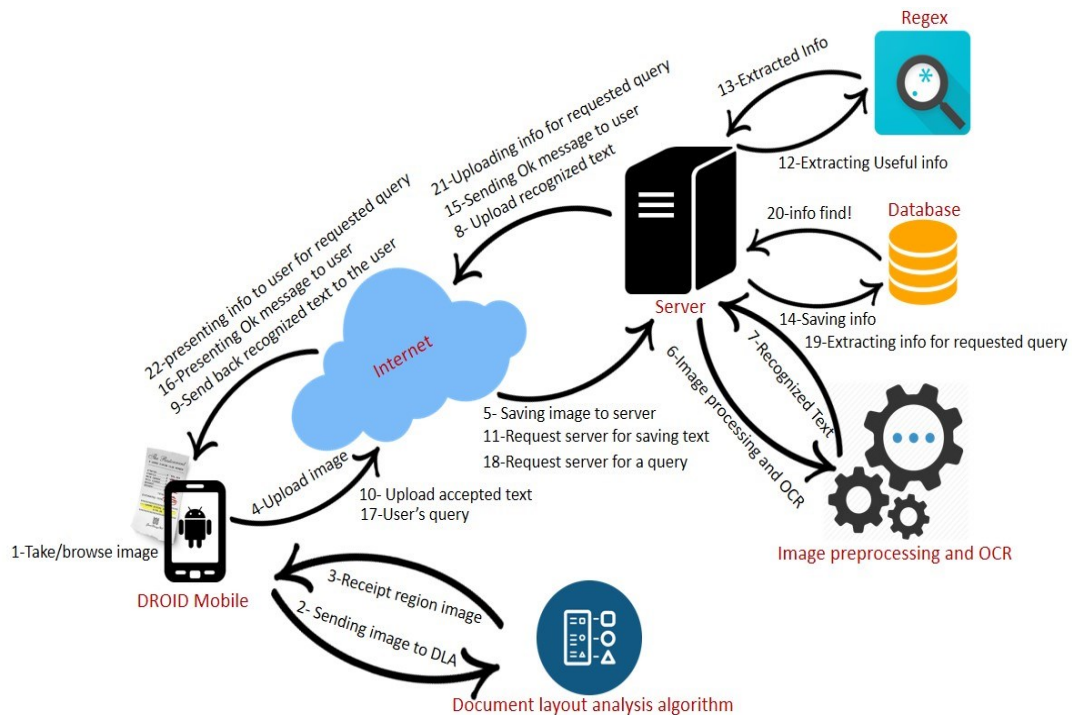


Figure 3.16 Structure of the proposed android application in term of client-server architecture.

The suggested Android application will take some procedures in the client mode. The procedures in client mode are such as capturing receipt images or browsing to a receipt image which exists in the mobile device's gallery, then it will run canny edge detection algorithm for identifying and extracting receipt region from the background of receipt images and finally it will submit the prepared image to the server side. In the other side, the application will run the most important procedures such as image processing algorithms, running Tesseract OCR engine on the receipt image, identifying and extracting beneficial information from the recognized text file and finally it will store the relevant information to the database. In the following subsections, the details for implementing both client and server side of the proposed android application is shown and discussed.

A. Client side

The language of Java as Android programming is utilized In Windows environment by using android studio [71] for designing and implementing the client side of the suggested Android application. Android platform is a comprehensive and effective platform that was utilized by the most mobile devices. For capturing receipt images, the proposed android application requires permission from the device to use the cameras on mobile devices. Such permission to entry the mobile's cameras is supplied by Android SDK (software development kit).

In client side, some important functions were applied in the suggested Android application for extracting and recognizing text on the receipt images. For making and beginning the connection between client and server side, client side plays vital responsibility.

Every user of the proposed Android application must have a distinctive account. This means that users must register or login to the application as the first step. The login's GUI is designed in the client side. This means that when some information is written to the username and password text box in the login page, then the client side will seek the server for checking the correctness of the entered username and password with the usernames registered to the system. Then the server will check the accuracy of the information entered. If the server matches the information in the database, then users are able to login to the application and use the functionality of the suggested Android application. But if there is no

such username information stored in the database, then the server will simply replay a message and tells there is no such username in the system, in such situation, user registration is required. The architecture of the step-by-step processes implemented in the client mode is shown and presented in the Figure 3.17.

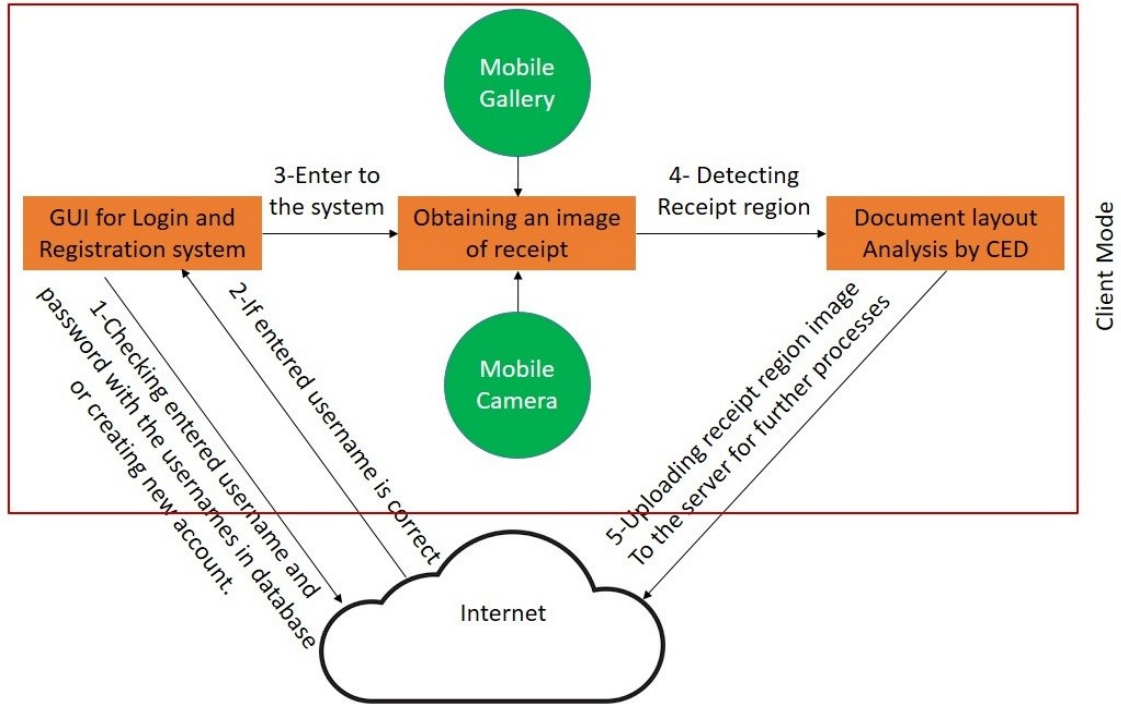


Figure 3.17 Structure of the client side's step-by-step processes in the suggested Android application.

The following step in the client side of the suggested Android application is browsing a receipt image that exists in the mobile device's gallery or capturing a new receipt image by mobile device cameras. The procedure for identifying the region of receipt by utilizing the canny edge detection algorithm will start after acquiring a receipt image. ScanLibrary [72] is a library for Android which is utilized in the suggested Android application for doing the task of identifying and getting receipt region from background of receipt image. The library implemented by using OpenCV functions, which uses canny function from OpenCV for identifying and getting the region of receipt.

The final step in the client side will take place when the region of receipt is ready to submit to the server side for applying additional process in the server side by simply clicking on the button of OCR in the client side. Before saving information to the database, the server-side will post the output of OCR to the client side after applying processes

included in the server side. This procedure is another way for improving OCR accuracy since users can examine and correct the text posted by the server manually before storing information to the database. Users query's GUI pages is also designed in the client side.

B. Server side

When the tasks of client mode have been finished, then the client side will submit the receipt region image to the server. Then the server will start applying other processes on the receipt image for identifying and getting text on the receipt images. The two most important processes in the suggested Android application are image processing algorithms and applying Tesseract on the receipt images. PHP as the server-side scripting language is utilized for implementing all the processes exist on the server side. The architecture or structure of the step-by-step processes implemented on the server side is shown and presented in the Figure 3.18.

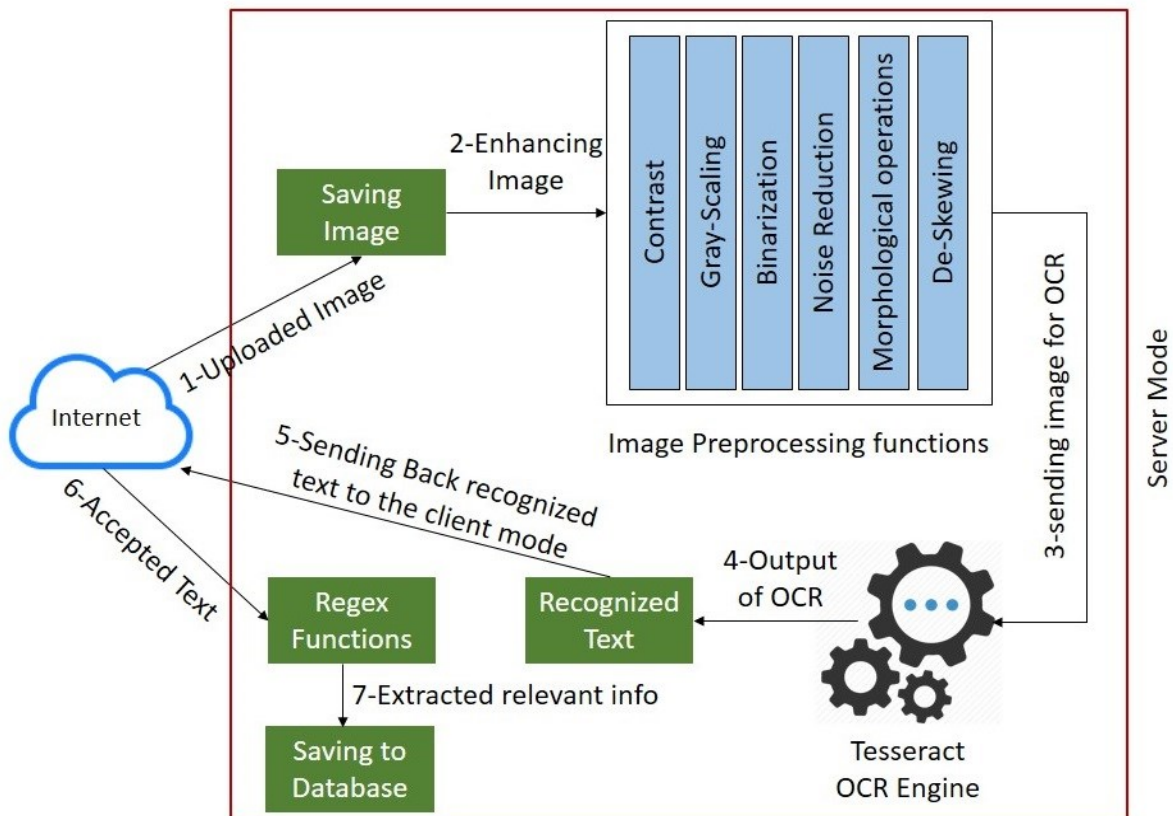


Figure 3.18 Structure of the server side's step-by-step processes in the suggested Android application.

When the receipt region image is submitted to the server side for performing additional processes on the server side, then the server side will start applying important processes exist on the server side.

Saving receipt image to a folder is the first task that will be performed by the server side. Then server side will apply a series of image processing algorithms on the receipt images. The image processing algorithms are the process of contrast, the process of gray-scale, the process of thresholding, reduction of noise, morphological operations and de-skewing operation. For applying all of these image processing algorithms, through PHP a shell script is called. ImageMagick library [66] is utilized as an open source image processing library for performing all the mentioned image processing algorithms on the receipt images in the suggested Android application.

The next step in the server side is performing a process on receipt images which is applying Tesseract OCR engine on the images for identifying and getting text on the receipt images. There are several different ways and techniques exist and suggested by researchers for identifying and getting text on the images, based on tests and finding, performing Tesseract on the receipt images fulfill the requirements for suggesting a strong Android application. The research found that there are some techniques that can greatly raise the rate of recognition by Tesseract such as introducing new fonts to the Tesseract OCR engine, introducing new dictionary to the Tesseract OCR engine, selecting a suitable page segmentation method and finally Tesseract accuracy improving by regex functions. All of these techniques have been performed in the server mode. Composer [73] is a common dependency manager for PHP which we utilized for running and installing Tesseract on the server side.

After all the processes related to the server side are finished, then a text file will be produced. Then the text in the text file will be posted by the server side to the client side. In the client side, a page will show the recognized text to the user of the application. The users can either click on the (ok) button for saving information to the database or click on the (cancel) button if the recognized text is not well-recognized. If the text accepted and submitted to the server, then the server will apply two other important processes. The first process is applying regex function on the text through PHP shell scripts for identifying and getting beneficial data. The next step which is the final process in the stack of the suggested techniques is saving beneficial data to the database for future use.

3.3. System Screenshots

This section presents and shows the screenshots of every page in the suggested Android application. A GUI is implemented for getting benefits from the usage of the techniques were adopted and applied in this research. The system on which the application is running is Samsung Galaxy J5 (2016) and the operating system is Android 6.0 Marshmallow.

Figure 3.19, is the login and registration page of the suggested Android application. This page will appear as the first page when users start using the suggested Android application. If the user is already registered to the system, then he/she can login to the system with his/her username and password, or else he/she must register himself/herself to the system for getting benefits from the usage of the suggested Android application.

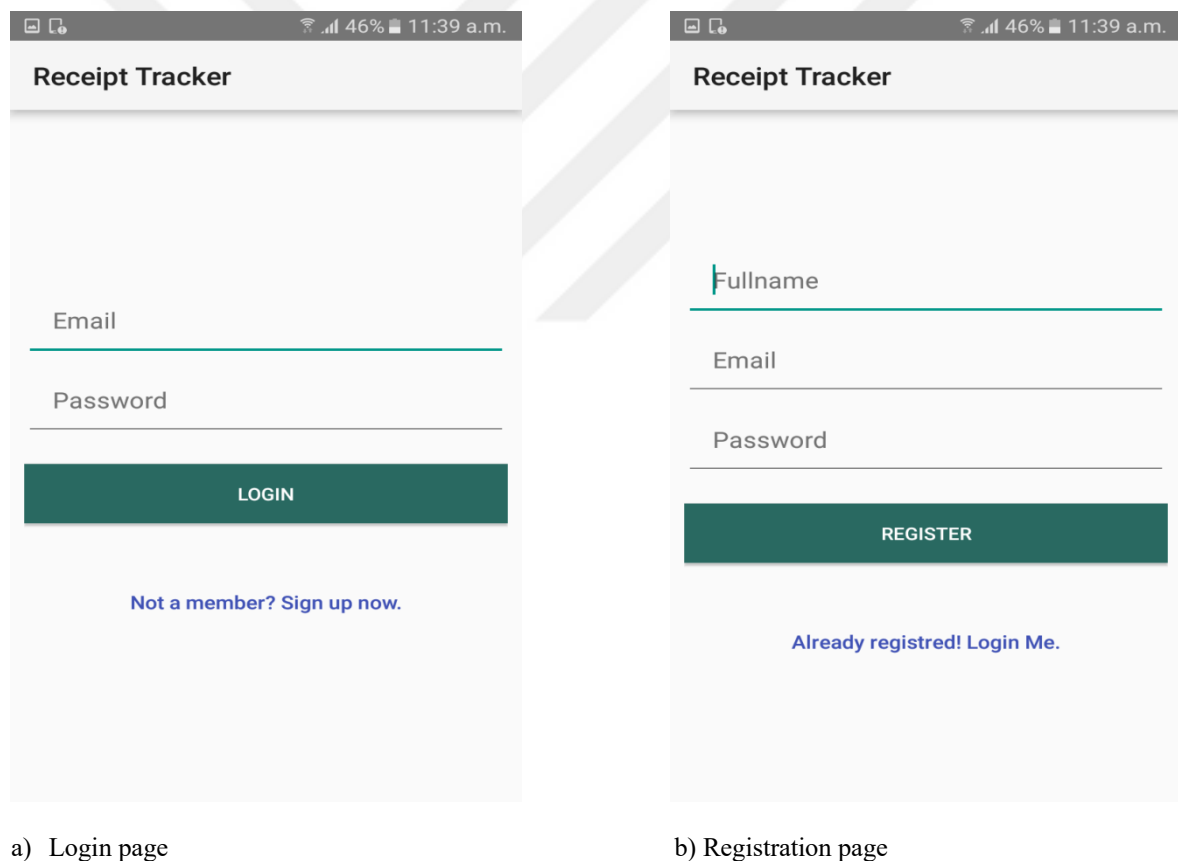


Figure 3.19 Screenshots of the login and registration page of the suggested Android application.

When the users of the application registered or login to the system then the home page will appear. Figure 3.20 is the home page of the application. In the home page, users can either click on the camera button or media button for obtaining an image of the receipt.

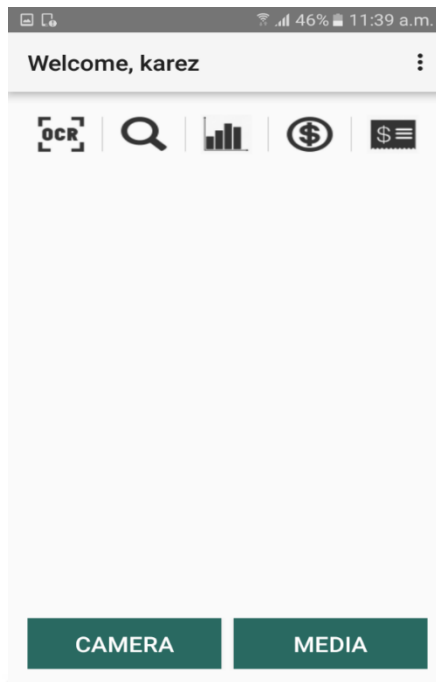


Figure 3.20 A screenshot of the home page of the application.



a) Detecting receipt region and cropping



b) Accepting receipt Image

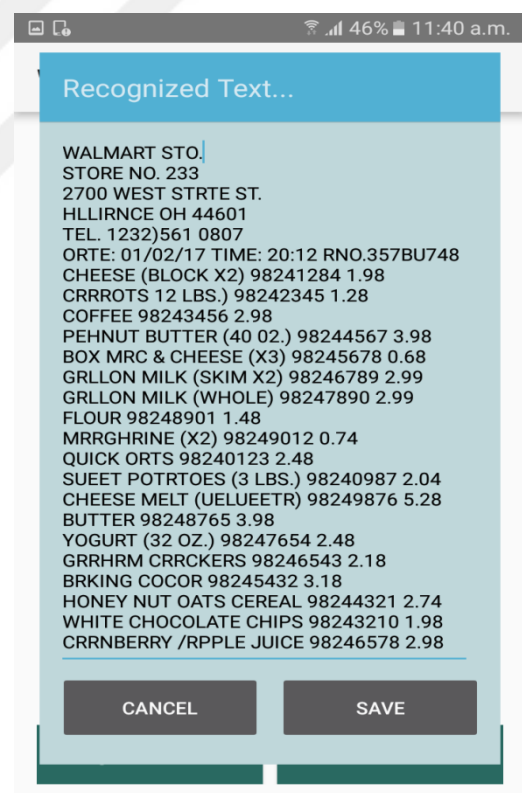
Figure 3.21 Screenshots of detecting region of receipt by canny edge detection algorithm.

After obtaining a receipt image, then the canny edge detection algorithm will run for identifying and separating the region of the receipt from the background of the receipt image. Figure 3.21 shows the screenshots of detecting the region of receipt from the background of a receipt image by utilizing canny edge detection algorithm. If the users of the application cropped the region of receipt properly, then users can click on the (done) button and perform following steps for finalizing the process, else users can retake or re-browse to another image.

After clicking on the (done) button, then the receipt image will appear on the home page and it is ready to submit to the server for recognizing text on it. In the home page, the users of the application can simply submit the image to the server by clicking on the (OCR) button, see Figure 3.22.a. Then the server will post the recognized text to the client side. On the client side, the application will show the recognized text, see Figure 3.22.b.



a) Image is ready for OCR



b) Recognized Text

Figure 3.22 Screenshots of submitting receipt image to the server and showing the result.

As appears in the Figure 3.22.b, the users can either click on the cancel button or save button. If the users of the application click on the cancel button the recognized text will be

ignored and the users of the application can again start the processes of submitting another receipt image. But if the users of the application click on the save button, then the application will submit the text to the server for extracting relevant information and storing the information in the database for answering demands later of the application users.

After performing all the above processes, to ensure the users of the application that all the processes have been performed successfully, the suggested Android application will replay by a message to the users of the application and says (your data successfully saved!). See Figure 3.23. The screenshots of using four users' queries in the suggested Android application is attached to the 4.1 section in chapter four.

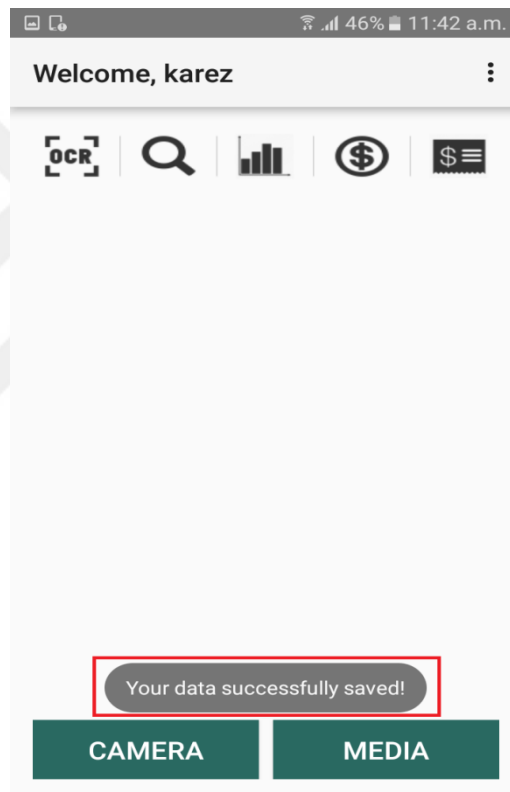


Figure 3.23 A screenshot of showing a message to the users of the suggested Android application.

4. QUERIES AND EXPERIMENTAL RESULTS

Presenting outcomes of created queries and experimental outcomes practiced is the main concept in which presented in this chapter. For showing the consequential aims of the suggested Android application, four queries were implemented for answering demands of users. The first section in this chapter covers the discussion of the implemented queries. The second section in this chapter presents the experimental outcomes practiced in this thesis. For showing the significant of the suggested techniques in this research, the practices are selected to be for different examinations.

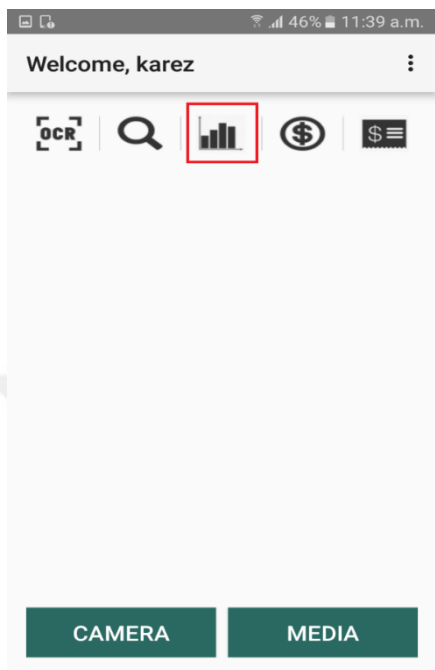
4.1. User Queries

In the suggested Android application, four queries are designed to fulfill the main concept and exceptional aims of the suggested research. The queries used here are the spend analyzer, receipt image discovering, determining total money expended from a date to date, and determining total money expended for a particular item. Such suggested queries have great impact on saving time in the case if the people want to calculate this information manually by searching across multiple receipts. All queries discussions and purposes are shown and presented in the following subsections.

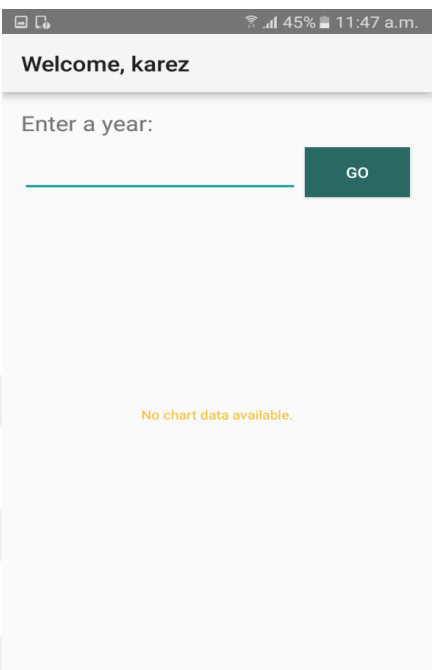
4.1.1. Spend analyzer

One of the queries in which implemented in the suggested Android application is spend analyzer. In a satisfactorily formatted chart, the users of the application can see the expended money history of one year ago for all months separately. The users of the application must write down a particular year to a box of text and then submit the year to the side of the server. When the server received the year, then for discovering the information related to the query, the server will perform an SQL command line for that purpose. The SQL command line will process the information related to the user demands. First, it will search in the database and discover all the receipt images. Then the SQL command line will determine and compute the total amount of money expended for the year entered by the user for all months individually. When the server finishes determining the requested demanded information, then it will post the information to the client mode in a

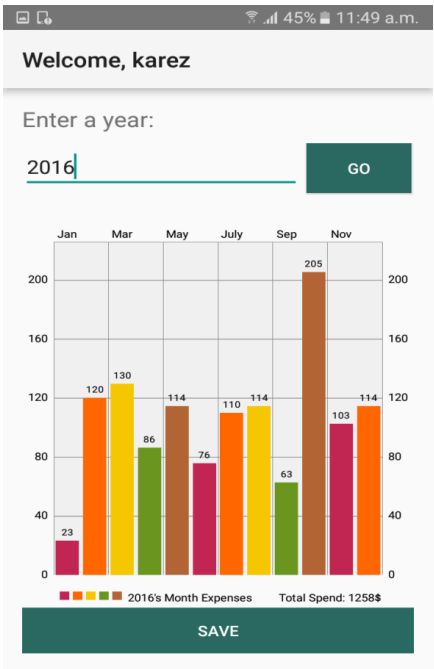
satisfactorily formatted chart. The chart in client mode is designed and implemented by using an Android library called the MPAndroidChart library [74]. Figure 4.1 is the screenshots used in this query.



a) Query's button.



b) Query's home page.



c) (Spend Analyzer) result.

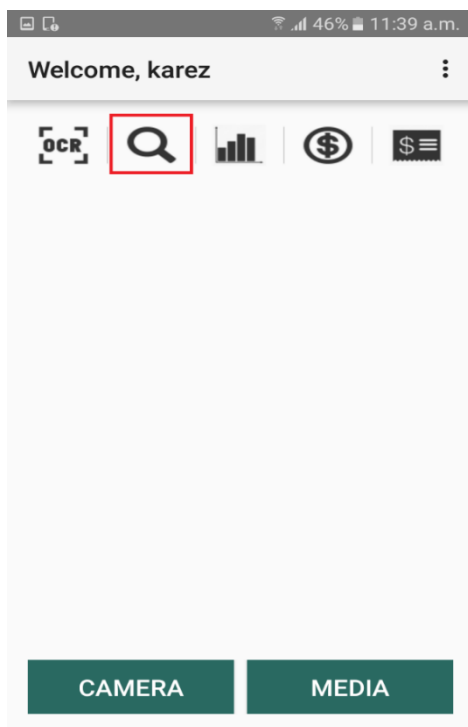
Figure 4.1 Screenshots of using Expend inspector query.

4.1.2. Receipt image discovering

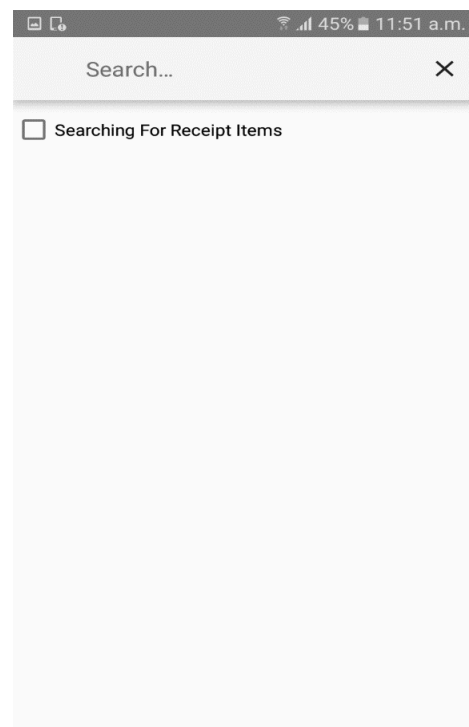
Discovering an image of receipt from the database of the suggested Android application is another query. The users of the application can simply write down one of the following information to the search box:

- Name of the market,
- Date of marketing,
- Or the name of the items which is bought by users.

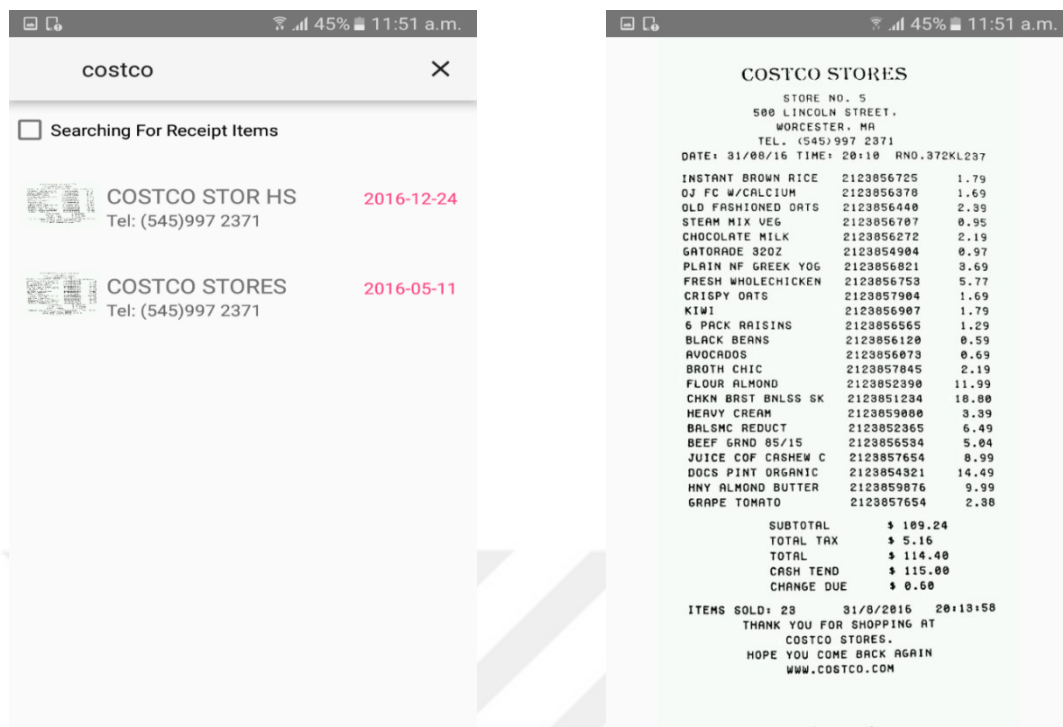
Then the users have to submit the information to the side of the server by clicking on the search button. Then for discovering the receipt images demanded by the user, the server will perform an SQL command line for that purpose. The SQL command line will process the information related to the user who requests the information. The SQL command line will discover the images of receipts that include the information submitted by the users who made the query. Then the server will post all the discovered images to the client side. Finally, the client-side will sort images in a designed GUI and presents the discovered images to the users of the application. Figure 4.2 shows the screenshots used in this query.



a) Query's button.



b) Query's home page.



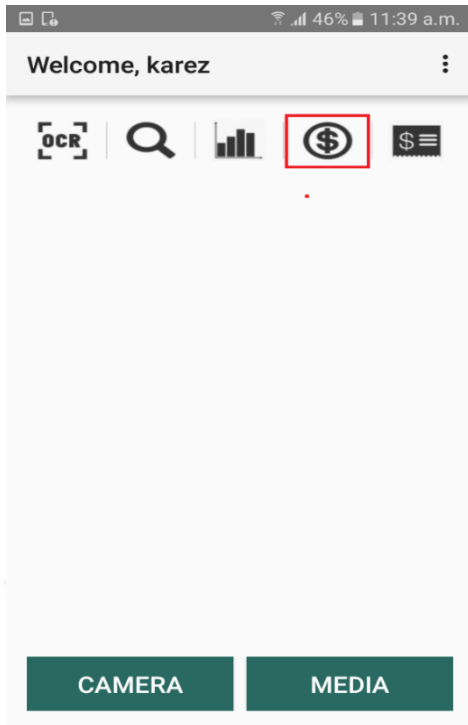
C) Finding receipt images of Costco stores.

d) Clicked Receipt image.

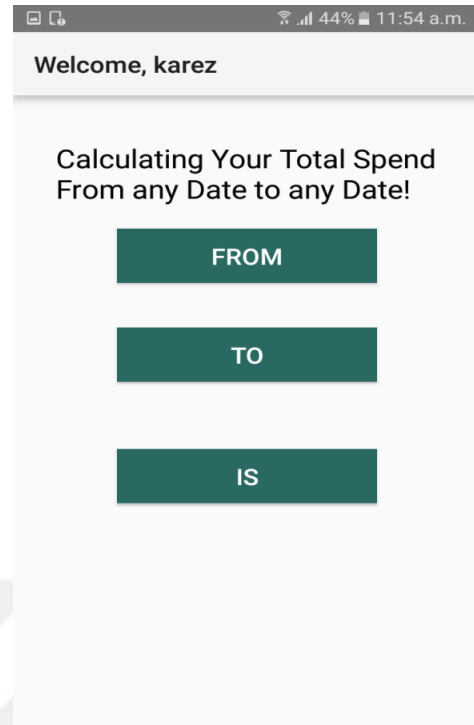
Figure 4.2 Screenshots of using (Discovering images of receipts) query.

4.1.3. Total money expended

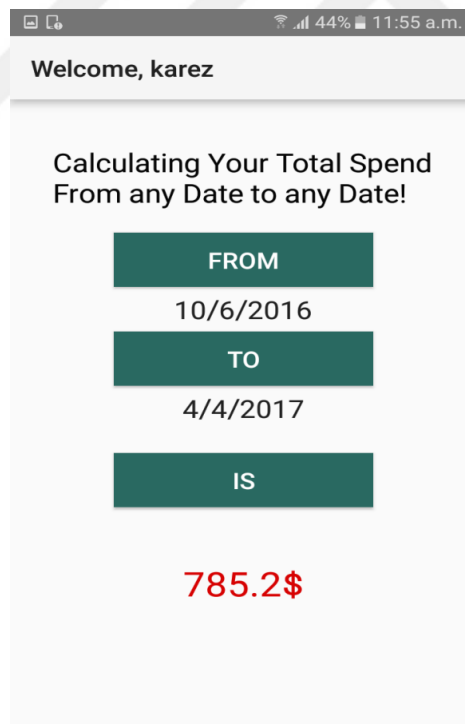
Determining the total amount of money expended from one date to another is another query suggested in this Android application. A GUI was implemented for that purpose, where the users must provide and write down the duration of marketing for computing total money expended during the provided information. When the users wrote down the dates, then the users have to submit the information to the server side by clicking on the (find) button. When the server received the info submitted by the users of the application, then for determining total money expended, the server will perform an SQL command line for that purpose. The SQL command line will process the information related to the user who demands. From provided date to the provided date, the SQL command line will discover all the receipt information associated with the user who demands. Then the SQL command line will compute the total amount of money expended and post the information on the server side. Then a page on the client-side will present and show the computed total money expended to the user of the application. Figure 4.3 is the screenshots of utilizing this query.



a) Query's button.



b) Query's home page.

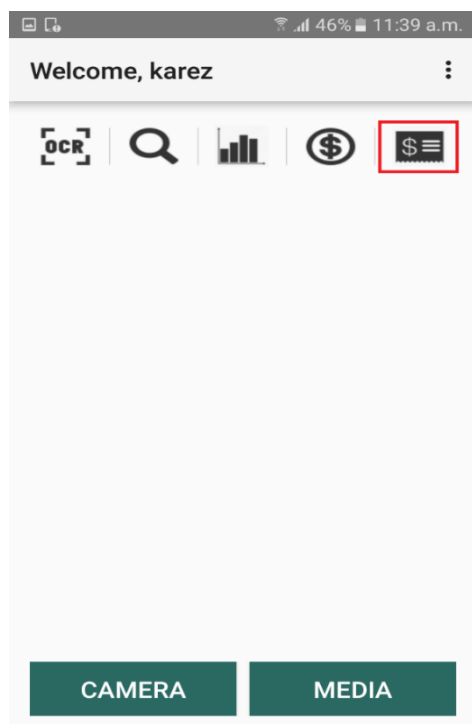


c) (Total money spent) result.

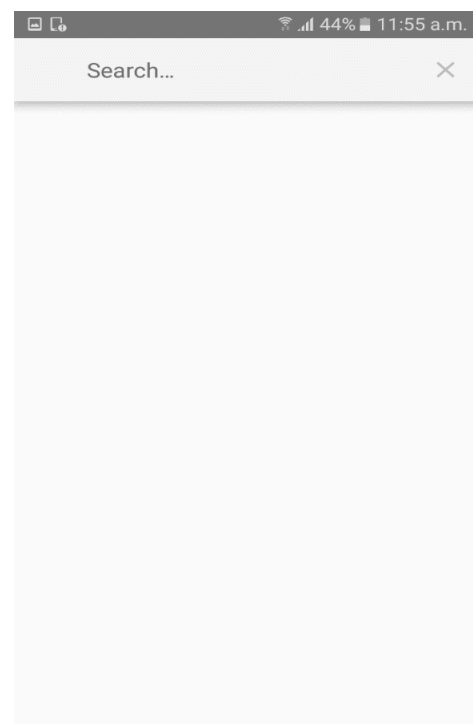
Figure 4.3 Screenshots of using total money expended query.

4.1.4. Total money expended for a particular item

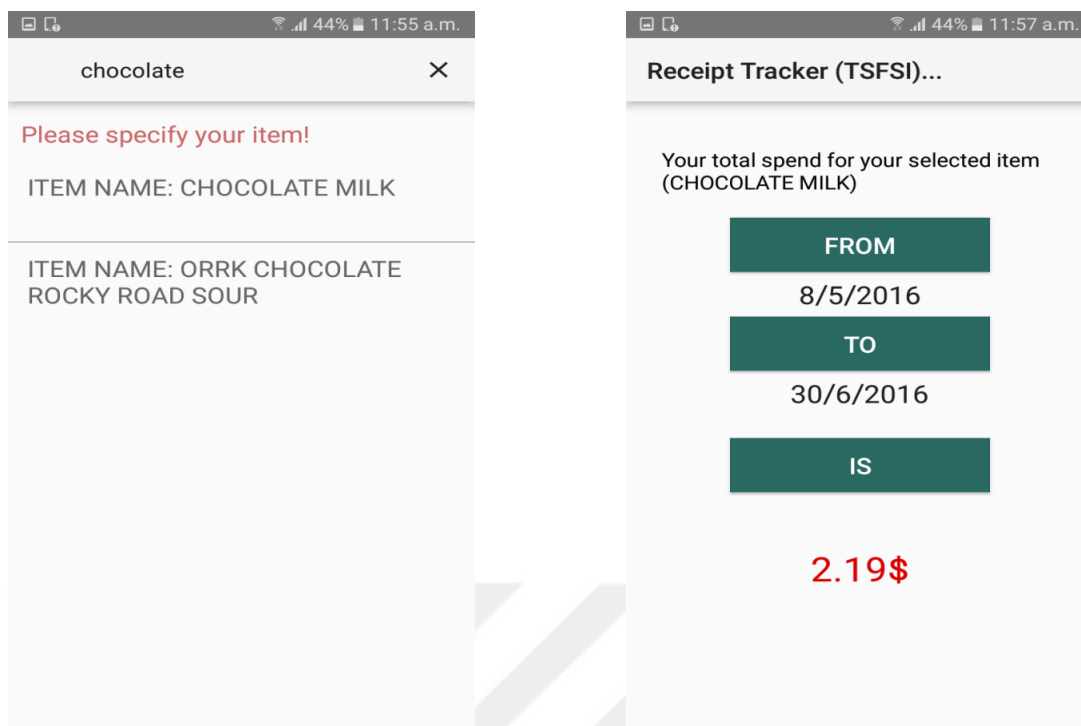
Determining the total money expended for a particular item from one date to another date is another implemented query. As the first process, the name of the item which the user of the application wants to determine total money expended for that particular item must be written down in the search box and submitted to the side of the server. Then the server will run an SQL command line for determining all the items contains provided name. Then the server will post all the determined item names to the client mode. The next step in the suggested query is selecting an item name and re-submitting the item name with the duration (dates) of the calculation. When the server receives the information submitted by the users of the application, then for determining the total amount of money expended for the provided item name, the server will perform an SQL command line for that purpose. The SQL command line will process the information related to the user. The server will run an SQL command line and will compute the total calculation for that particular item from the provided date to the provided date. As the final step, the server will post the computed information to the client side. A page on the client-side will present and show the computed total money expended for the provided item name to the user of the application. Figure 4.4 is the screenshot for this query.



a) Query's button.



b) Query's home page.



c) Finding goods contain the chocolate name.

d) Total money spends for chocolate.

Figure 4.4 Screenshots of using (Total money expended for a particular item) query.

4.2. Experimental Outcomes

Checking accuracy of character and word recognition of the applications and also checking the performance of the applications are the well-known procedures for evaluating applications that OCR plays a major role. Presenting or showing the suggested Android application's efficiency and accuracy percentage is the main concept of this section from this chapter. There are five subsections in this section. The discussion of the key points utilized for checking the capability of the suggested Android application covers the first section. The second section describes the corpus of the examination utilized for testing the suggested Android application. Experimental outcomes are shown in the third and fourth section in the histogram and tabular manner. Two unusual text fonts are tested. This means that two different datasets were utilized to evaluate the performance of the suggested Android application for the two different fonts. The Fake receipt text font on the receipt images established the first dataset. In addition the Merchant copy font on the receipt images established the second dataset. Experimental outcomes for the first dataset are shown in the third section, and experimental outcomes for the second dataset are shown in

the fourth section. The fifth and final section shows the comparisons and evaluations of the experimental outcomes achieved.

4.2.1. Capability metrics

In the Fifth Annual Test accuracy of OCR applications [110], an institute at the Nevada University suggested two important metrics for quantifying the capability of applications where optical character recognition technology plays a crucial role. Character and word accuracy metrics are the two metrics suggested by the mentioned annual and used here for showing capability of the proposed android application. The detail discussion of both metrics was shown in the following subsections.

A) Character accuracy

Checking Character accuracy is one of the metrics suggested for evaluating the capability of OCR applications. The Character accuracy mechanism will determine a percentage that presents the total number of characters successfully recognized. The equation for determining accuracy in terms of character accuracy are shown below:

$$\frac{\text{total number of characters} - \text{error numbers}}{\text{total number of characters}} 100\% \quad (4.1)$$

The equation will subtract the number of characters which were not successfully recognized from the total number of characters and find the number of characters successfully recognized. Then for computing character accuracy, the suggested equation will divide the number of characters that were recognized successfully by the total number of characters. As the final step for computing the percentage, the equation will multiply obtained number by 100%.

B) Word accuracy

Word accuracy checking is another metrics suggested for evaluating the capability of OCR applications. The equation for determining accuracy in terms of word accuracy is shown below:

$$\frac{\text{The Number of accurate words}}{\text{total words}} 100\% \quad (4.2)$$

Since a single mismatched character from a word will contemplate the word as not-recognized, therefore checking Word accuracy is very strict. Also, checking word accuracy is case-insensitive. This means that if a character has the wrong case, then the policy of checking word accuracy does not contemplate the word as the wrong detection.

The efficiency of the suggested Android application in terms of time performance was also calculated. This is another important metrics for showing the capability of the OCR applications. Usually, client-server architecture systems have the problem of latency of the network. Such issue does not contemplated during testing the suggested Android application.

4.2.2. Examination corpus

By following common recommendations, a number of receipts were created for finding the capability of the suggested Android application. This receipts contains common information such as Name of market, number of market, phone number of market, website of market, address of market, purchasing time and date, ID number of Receipt, Item name purchased, price of item purchased. Information related to expending money such as subtotal money expended, Total money expended, total money paid for tax, total money paid for cash, and change due money are also shown. Two unusual text fonts used on the receipts, the text fonts are fake receipt text font and merchant copy text font, the size of text is very small, also text lines on the receipts are near to each other. The research shows that receipts created fulfill the requirements and create appropriate receipts.

Through utilizing a mobile device camera, receipt images are obtained under the various situations. For example, a number of receipt images have the problem of tilting, skewing and un-even lighting, also a number of receipt images were captured in a good situation. Two different datasets of receipt images were created. Each dataset contains 10 receipt images, and the images of both datasets are captured in the same situation. The only difference that differentiates both datasets from each other is the difference of the text fonts. Fake receipt text font is the text font on the receipt images of the first dataset, also the merchant copy text font is the text font on the receipt images of the second dataset. This shows the capability of the suggested Android application for different text fonts. Up to 190 words and up to 975 characters were used in the receipts.

4.2.3. Fake receipt font experimental outcomes

This section presents or shows the capability of the suggested Android application for recognizing fake receipt text font on the receipt images. 10 receipt images concluded the dataset for the fake receipt text font. Under four different examinations, this text font tested by the suggested Android application. The first examines the suggested Android application by utilizing all the algorithms selected for promoting the rate of recognition by Tesseract OCR engine. The second examines the suggested Android application by utilizing all the algorithms except algorithms of image processing (except canny edge detection algorithm and thresholding algorithm). The third examines the suggested Android application by utilizing all the algorithms except techniques selected and performed for raising recognition rate by Tesseract. The Fourth examines the suggested Android application by instantly submitting receipt images to Tesseract without utilizing any algorithms selected for promoting the recognition rate by Tesseract (image processing algorithms and Tesseract improvements). Time performance of the application, word accuracy percentage and character accuracy percentage are all computed for these examinations. The following subsections cover the details and show the outcomes of all the examinations.

A. Examination 1

The capability of the suggested Android application is examined in this examination for detecting fake receipt text font by utilizing all the algorithms selected for promoting the rate of recognition by Tesseract OCR engine. The algorithms or techniques that have great impact on improving the recognition rate are techniques related to the Tesseract for promoting rate of recognition (introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract OCR engine, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions) and algorithms of image processing (process of contrast, process of gray-scale, process of thresholding, reduction of noise, morphological operations and operation of de-skewing). Time performance of the application, character accuracy percentage and word accuracy percentage are computed for all the images in this examination.

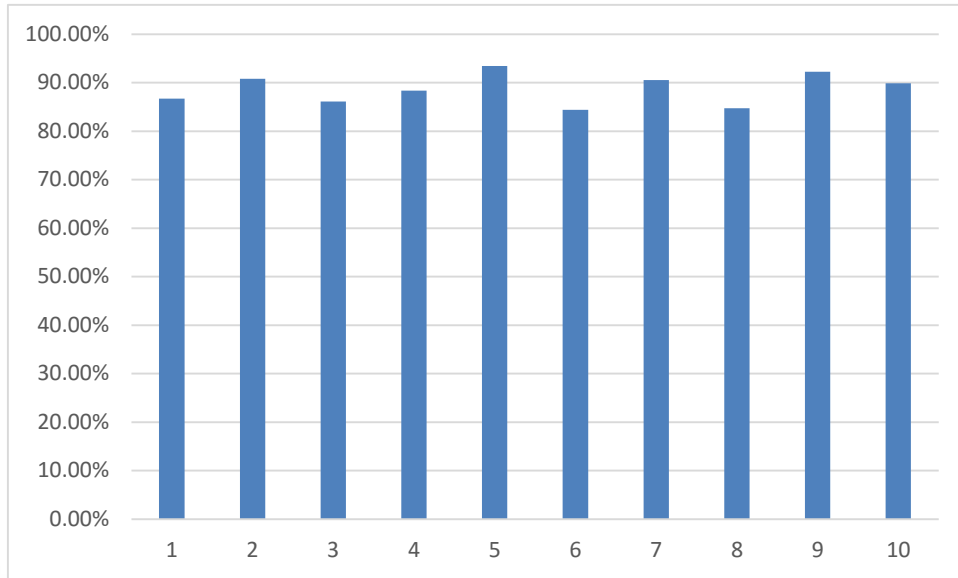
The experimental outcomes are presented in Table 4.1. For each receipt image, the table includes number of words and characters (without spaces), the performance of the

application in term of time, the percentage of word accuracy and percentage of character accuracy. The average number in percentage average is computed for the columns exist in the table.

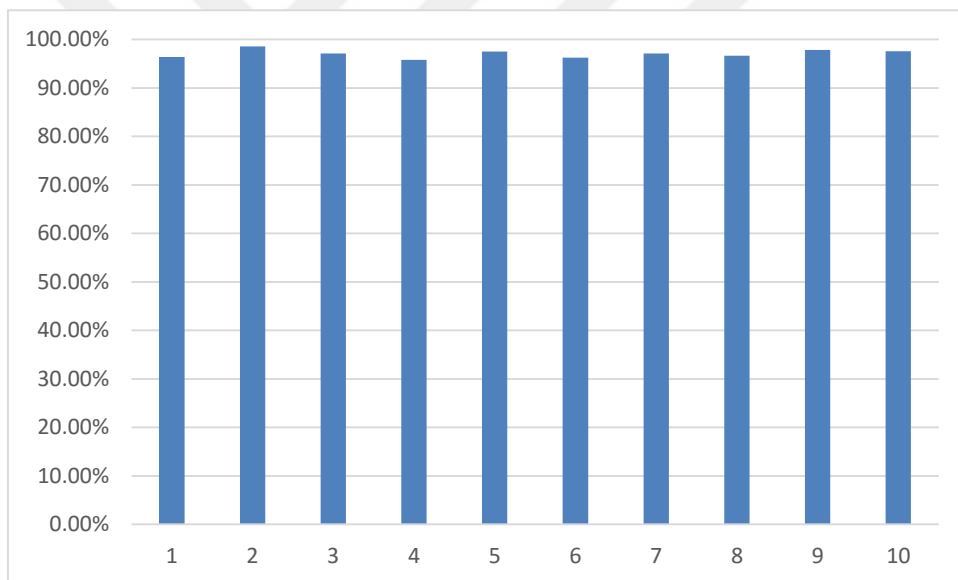
Table 4.1 Outcomes obtained for fake receipt text font in the first examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	86.70 %	96.40 %	6.81s
2	141	725	90.78 %	98.56 %	5.91s
3	151	839	86.09 %	97.13 %	6.86s
4	163	715	88.34 %	95.80 %	5.73s
5	153	722	93.46 %	97.50 %	5.84s
6	173	930	84.39 %	96.23 %	6.77s
7	159	802	90.56 %	97.13 %	6.37s
8	157	929	84.71 %	96.66 %	7.24s
9	168	888	92.26 %	97.86 %	6.83s
10	158	878	89.87 %	97.60 %	6.76s
Avg.	161	840	88.71 %	97.08 %	6.51s

The results for this examination are 88.71 % for word accuracy percentage and 97.08 % for character accuracy percentage. The performance in terms of time of the suggested Android application is 6.51 sec. This examination proves that the suggested techniques have a great impact for promoting the recognition rate. As mentioned, if there is a single mismatch character from a word, the word will be counted as a wrong recognition. Therefore usually word accuracy percentage and character accuracy percentage in the most researches are totally different from each other. This means a big gap will appear between word and character accuracy. The histogram of outcomes experienced during this examination for both word and character accuracy percentages are presented in Figure 4.5.



a) Word accuracy histogram



b) Character accuracy histogram

Figure 4.5 Word and character rates histogram for fake receipt font in the first case.

B. Examination 2

The capability of the suggested Android application is examined for detecting fake receipt text font without utilizing some algorithms of image processing (process of contrast, process of gray-scale, reduction of noise, morphological operations and operation of de-skewing), but with utilizing all the Tesseract improvement techniques (introducing

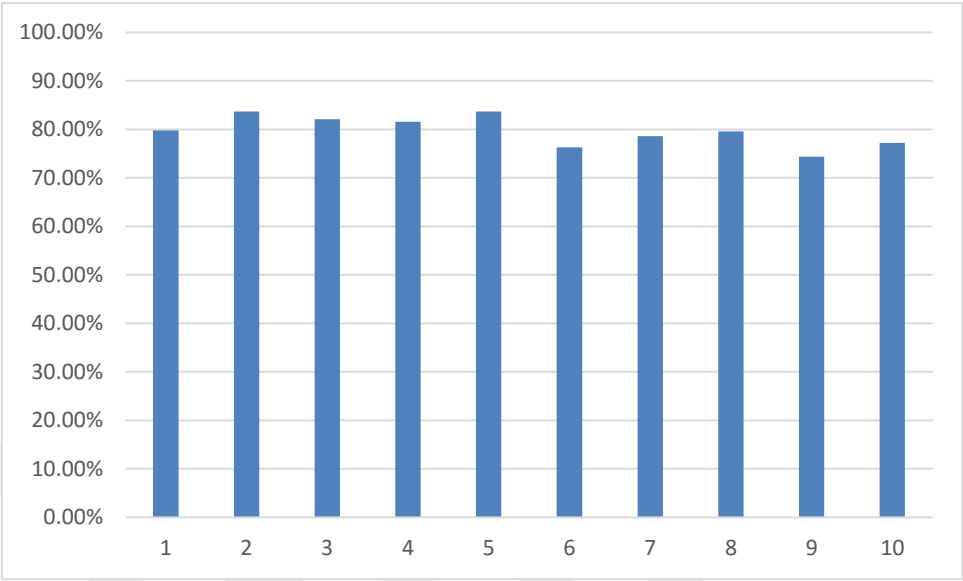
new fonts to the Tesseract, introducing new dictionary to the Tesseract, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions). This research proved the impacts of the algorithms of image processing suggested. Time performance of the application, word accuracy percentage and character accuracy percentage is computed for all the images. The experimental outcomes are presented in Table 4.2.

Table 4.2 Outcomes obtained for fake receipt text font in the second examination.

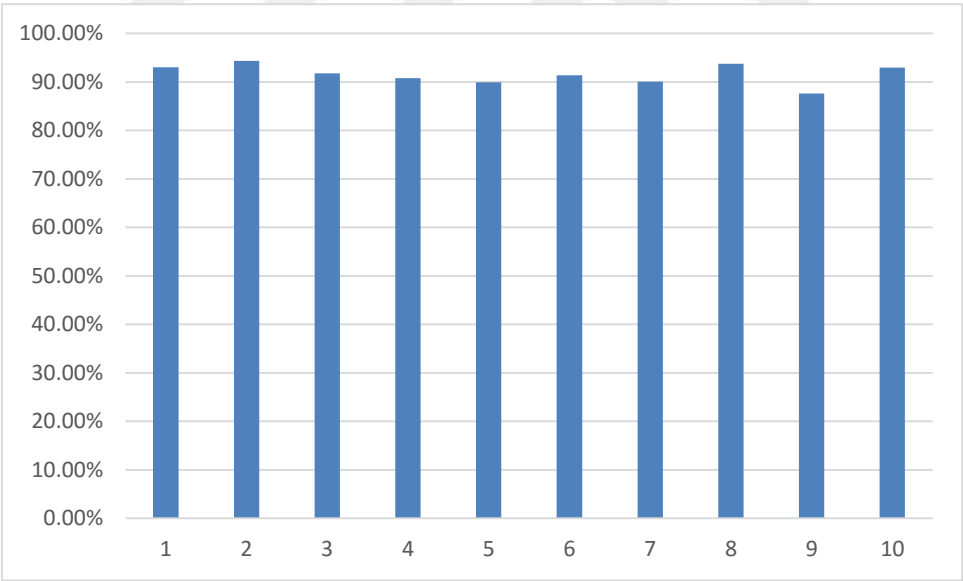
No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	79.78 %	93.01 %	5.97s
2	141	725	83.68 %	94.34 %	4.96s
3	151	839	82.11 %	91.77 %	5.03s
4	163	715	81.59 %	90.76 %	5.10s
5	153	722	83.66 %	89.88 %	5.25s
6	173	930	76.30 %	91.39 %	6.80s
7	159	802	78.61 %	90.02 %	5.18s
8	157	929	79.61 %	93.75 %	5.91s
9	168	888	74.40 %	87.61 %	5.16s
10	158	878	77.21 %	92.93 %	5.76s
Avg.	161	840	79.69 %	91.54 %	5.51s

The results of this examination are 79.69 % for word accuracy percentage and 91.54 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 5.51 sec. This examination proves that the suggested algorithms of image processing have great impacts for promoting the rate of recognition, because the results of this examination decreased by 9.02 % and 5.54 % respectively for word and character accuracy percentages when compared with the first

examination. The histogram of outcomes experienced during this examination for both word accuracy percentages and character accuracy percentages are presented in Figure 4.6.



a) Word accuracy histogram



b) Character accuracy histogram

Figure 4.6 Word and character rates histogram for fake receipt font in the second case.

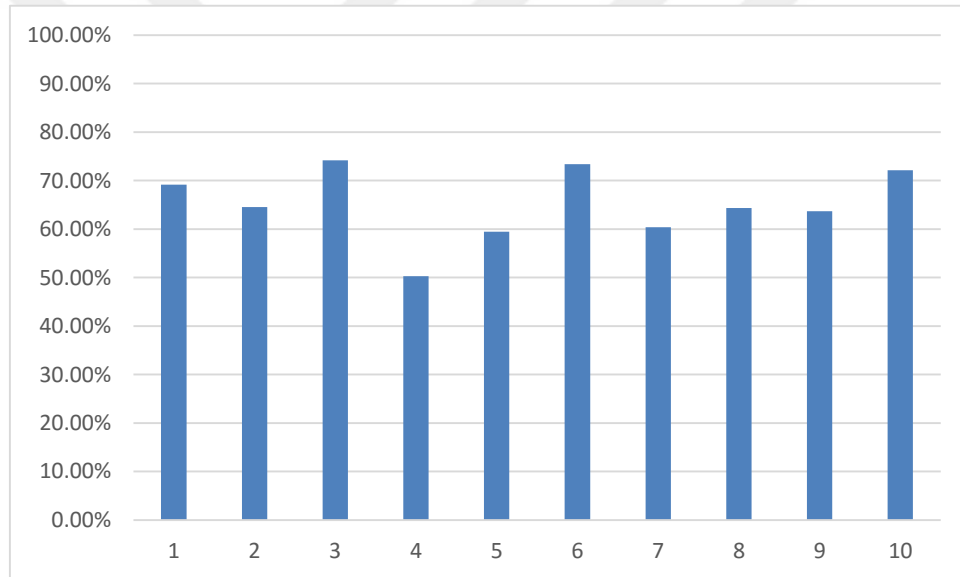
C. Examination 3

The capability of the suggested Android application was examined for detecting fake receipt text font with utilizing all image processing algorithms (process of contrast, process of gray-scale, process of thresholding, reduction of noise, morphological operations and operation of de-skewing), but without utilizing all the Tesseract improvement techniques (introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract OCR engine, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions). The research proved the impacts of the Tesseract improvement techniques suggested. Again, Time performance of the application, word accuracy percentage and character accuracy percentage is computed for all the images in this examination. The experimental outcomes are presented in Table 4.3.

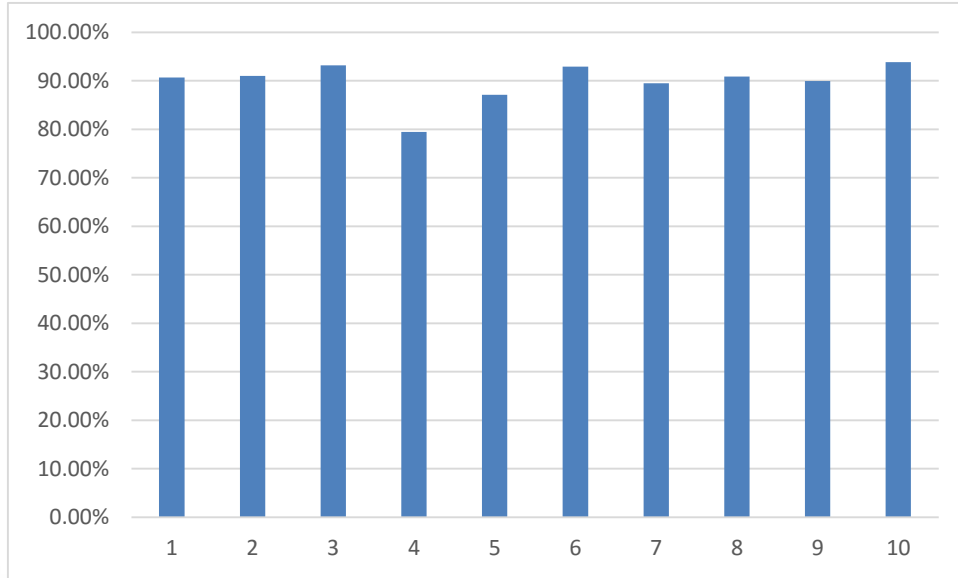
Table 4.3 Outcomes obtained for fake receipt text font in the third examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	69.14 %	90.65 %	6.54s
2	141	725	64.53 %	91.03 %	5.67s
3	151	839	74.17 %	93.20 %	6.57s
4	163	715	50.30 %	79.44 %	5.65s
5	153	722	59.47 %	87.11 %	6.14s
6	173	930	73.41 %	92.90 %	6.54s
7	159	802	60.37 %	89.52 %	5.96s
8	157	929	64.33 %	90.85 %	7.05s
9	168	888	63.69 %	89.97 %	6.27s
10	158	878	72.15 %	93.84 %	6.36s
Avg.	161	840	65.15 %	89.85 %	6.27s

The results of this examination are 65.15 % for word accuracy percentage and 89.85 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 6.27 sec. This examination proves that the suggested techniques related to the Tesseract improvement improve the recognition rates. Also these techniques have more affects than some of the algorithms of image processing, because when some of the algorithms of image processing are ignored, the results decreased by 9.02 % and 5.54 % respectively for word and character accuracy percentages, but ignoring techniques related to the Tesseract improvement decreased the results by 23.56 % and 7.23 % respectively for word and character accuracy percentages. The histogram of outcomes for both word accuracy percentages and character accuracy percentages are presented in Figure 4.7.



a) Word accuracy histogram



b) Character accuracy histogram.

Figure 4.7 Word and character rates histogram for fake receipt font in the third case.

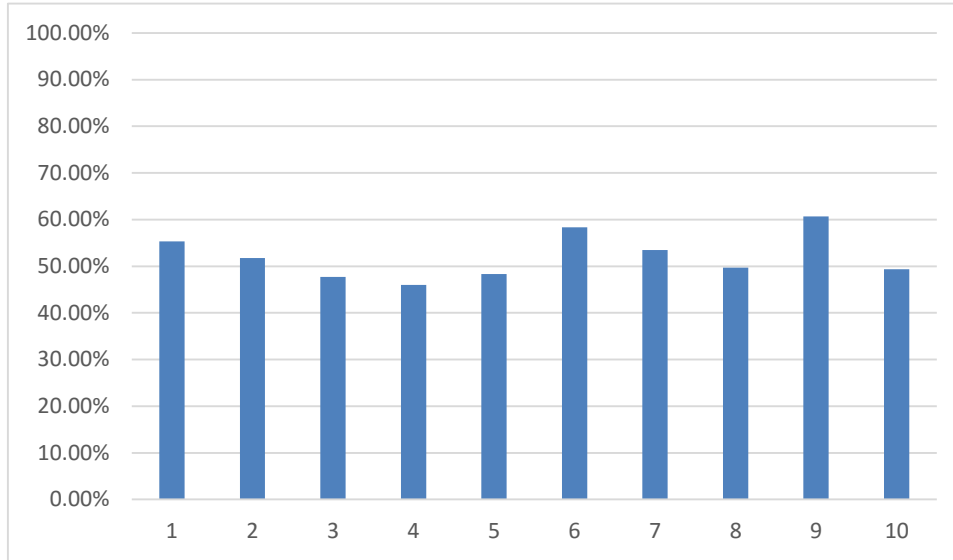
D. Examination 4

The capability of the suggested Android application is examined here for detecting fake receipt text font while ignoring all the techniques suggested for promoting the rate of recognition. Techniques suggested are image processing algorithms (the process of contrast, the process of gray-scale and so forth) and all the techniques related to the Tesseract improvement (introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract OCR engine and so on). The research proved the great impacts of the techniques suggested. Time performance of the application, word accuracy and character accuracy are computed for all the images. The experimental outcomes are presented in Table 4.4.

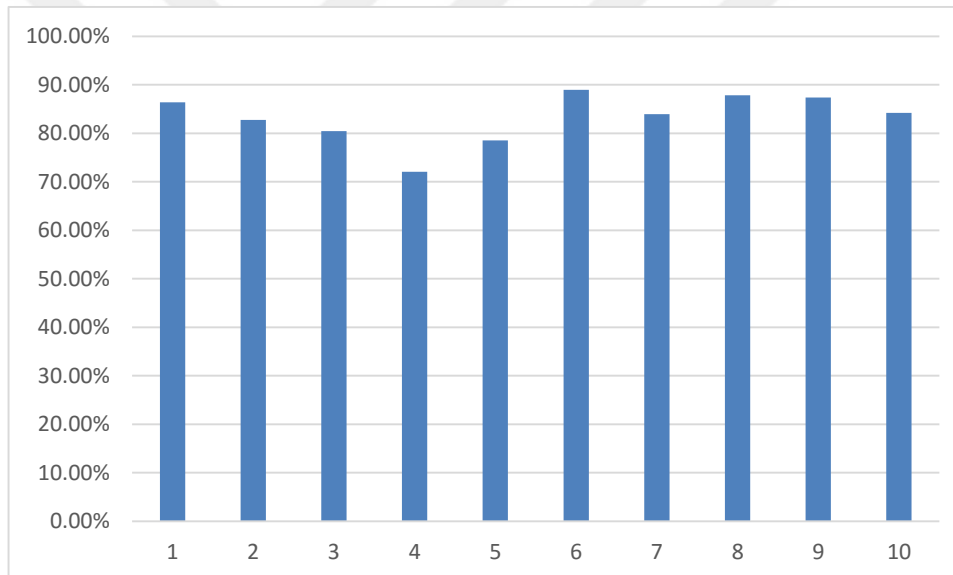
Table 4.4 Outcomes obtained for fake receipt text font in the fourth examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	55.31 %	86.34 %	5.84s
2	141	725	51.77 %	82.75 %	4.97s
3	151	839	47.68 %	80.45 %	5.40s
4	163	715	46.01 %	72.02 %	4.36s
5	153	722	48.36 %	78.53 %	5.33s
6	173	930	58.38 %	88.92 %	5.27s
7	159	802	53.45 %	83.91 %	5.08s
8	157	929	49.68 %	87.83 %	5.88s
9	168	888	60.71 %	87.38 %	5.70s
10	158	878	49.36 %	84.16 %	5.51s
Avg.	161	840	52.07 %	83.22 %	5.33s

The results of this examination are 52.07 % for word accuracy percentage and 83.22 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 5.33 sec. This examination shows that suggested techniques have great impacts for promoting the recognition rate. Because the results of this examination decreased by 36.64 % and 13.86 % respectively for word and character accuracy percentages when compared with the first examination. This shows that instantly submitting receipt images to Tesseract for recognition of text without utilizing techniques discussed will result in worse outcomes. The histogram of outcomes experienced for both word accuracy percentages and character accuracy percentages are presented in Figure 4.8.



a) Word accuracy histogram



b) Character accuracy histogram.

Figure 4.8 Word and character rates histogram for fake receipt font in the fourth case.

4.2.4. Merchant copy font experimental outcomes

This section presents or shows the capability of the suggested Android application for recognizing merchant copy text font on the receipt images. 10 receipt images concluded the dataset for the merchant copy text font. The Merchant copy text font dataset is almost the same as the first dataset. The only difference that differentiates both datasets from each other is the difference from their text fonts. This text font was tested in four different ways

using the suggested Android application. The first examines the suggested Android application by utilizing all the algorithms selected for promoting the rate of recognition by Tesseract OCR engine. The second examines the suggested Android application by utilizing all the algorithms except algorithms of image processing (except canny edge detection algorithm and thresholding algorithm). The third examines the suggested Android application by utilizing all the algorithms except techniques selected and performed for raising the recognition rate by Tesseract. The Fourth examines the suggested Android application by instantly submitting receipt images to Tesseract without utilizing any algorithms selected for promoting the rate of recognition by Tesseract (image processing algorithms and Tesseract improvements). Time performance of the application, word accuracy percentage and character accuracy percentage is computed for all the examinations. The Following subsections cover the outcomes of all the examinations.

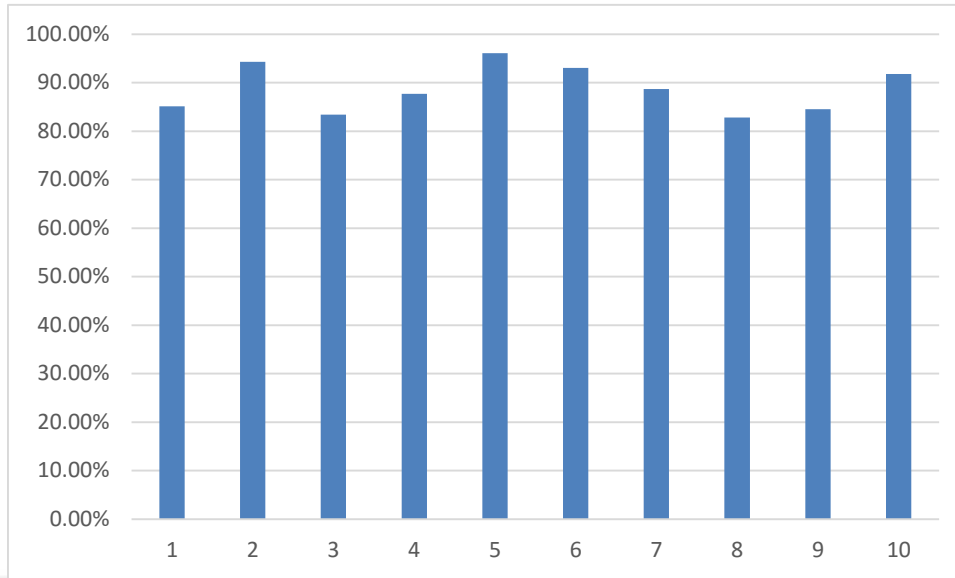
A. Examination 1

The capability of the application was examined for detecting merchant copy text font by utilizing all the algorithms and techniques proposed in this research for improving OCR accuracy. The techniques that have great impact on improving the recognition rates are techniques related to the Tesseract (introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions) and algorithms of image processing (contrast, gray-scale, thresholding, reduction of noise, morphological operations and de-skewing). The experimental outcomes are presented in Table 4.5.

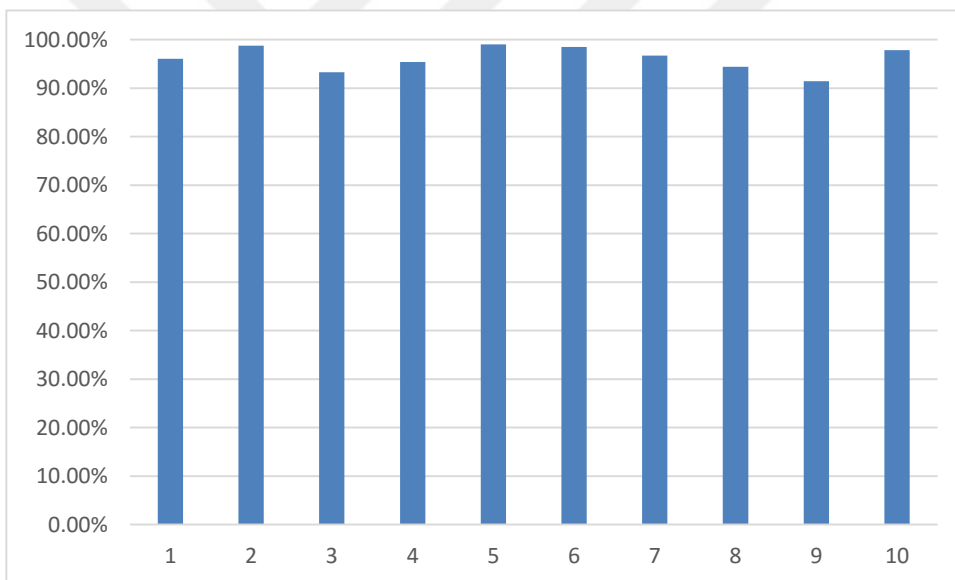
Table 4.5 Outcomes obtained for merchant copy text font in the first examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	85.10 %	96.09 %	7.43s
2	141	725	94.32 %	98.75 %	6.15s
3	151	839	83.44 %	93.32 %	6.41s
4	163	715	87.73 %	95.38 %	6.07s
5	153	722	96.07 %	99.03 %	6.28s
6	173	930	93.06 %	98.49 %	7.44s
7	159	802	88.67 %	96.75 %	6.48s
8	157	929	82.80 %	94.40 %	7.90s
9	168	888	84.52 %	91.44 %	6.13s
10	158	878	91.77 %	97.83 %	5.97s
Avg.	161	840	88.74 %	96.14 %	6.62s

The results of this examination were 88.74 % for word accuracy percentage and 96.14 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 6.62 sec. This examination proved that suggested techniques greatly improve the recognition rates. As mentioned if there is a single mismatch character from a word, the word will be counted as a wrong recognition. Therefore the percentage of accurate words and characters in this study are totally different from each other. The histogram of outcomes experienced for both word and character accuracy percentages are presented in Figure 4.9.



a) Word accuracy histogram.



b) Character accuracy histogram.

Figure 4.9 Word and character rates histogram for merchant copy font in the first case.

B. Examination 2

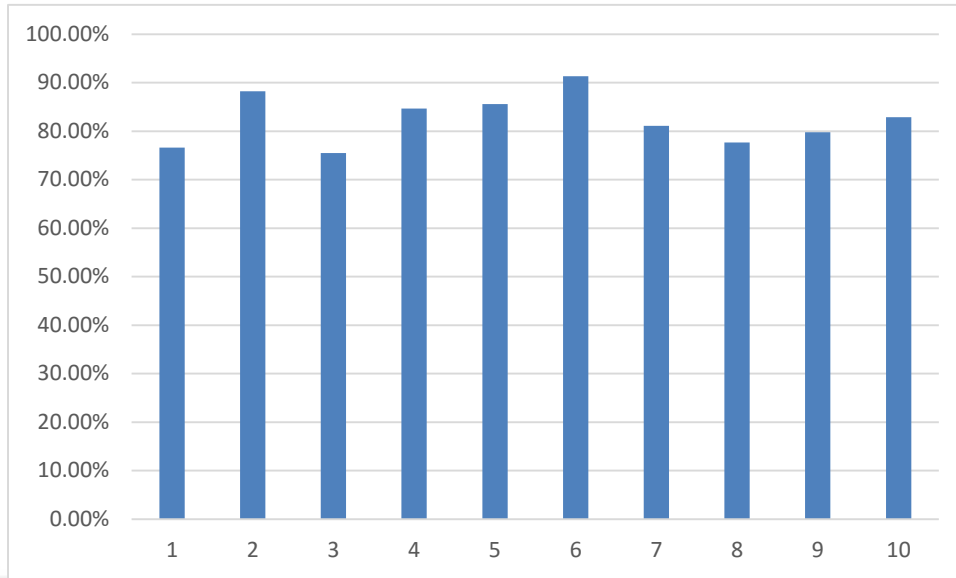
The capability of the application is examined for detecting merchant copy text fonts without utilizing some image processing algorithms (contrast, gray-scale, reduction of noise, morphological operations and de-skewing), but by utilizing all the Tesseract improvement techniques (introducing new fonts to the Tesseract, introducing new

dictionary to the Tesseract, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions). The benefits of using these algorithms for image processing are clearly evident. The experimental outcomes are presented in Table 4.6.

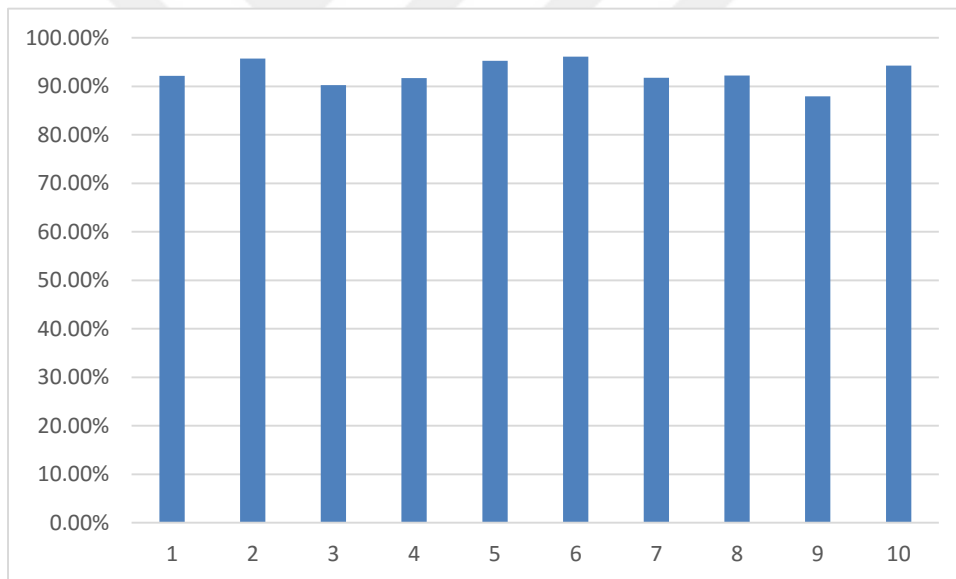
Table 4.6 Outcomes obtained for merchant copy text font in the second examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	76.59 %	92.19 %	6.83s
2	141	725	88.25 %	95.72 %	5.48s
3	151	839	75.49 %	90.22 %	4.61s
4	163	715	84.66 %	91.72 %	5.89s
5	153	722	85.62 %	95.29 %	5.96s
6	173	930	91.32 %	96.12 %	6.01s
7	159	802	81.13 %	91.77 %	6.15s
8	157	929	77.70 %	92.24 %	5.02s
9	168	888	79.76 %	87.95 %	5.32s
10	158	878	82.91 %	94.30 %	5.51s
Avg.	161	840	82.34 %	92.75 %	5.67s

The results of this examination were 82.34 % and 92.75 % respectively for word and character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 5.67 sec. This examination proves that suggested algorithms of image processing greatly improves recognition rates. Because the results of this examination decreased by 6.40 % and 3.39 % respectively for word and character accuracy percentages when compared with the first examination in this experiment. The histogram of outcomes experienced during this examination for both word accuracy percentages and character accuracy percentages are presented in the Figure 4.10.



a) Word accuracy histogram.



b) Character accuracy histogram.

Figure 4.10 Word and character rates histogram for merchant copy font in the second case.

C. Examination 3

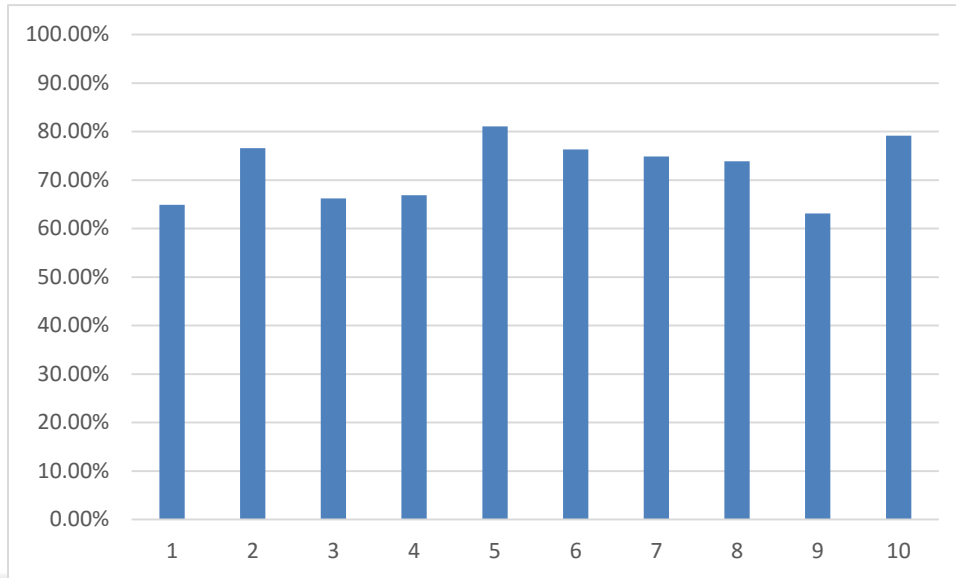
The capability of the application is examined for detecting Merchant copy text font by utilizing all the image processing algorithms (contrast, gray-scale and so on), but without utilizing all the Tesseract improvement techniques (introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract OCR engine and so on). This experiment proves

that there are many benefits techniques related to the Tesseract for improving recognition rates. The experimental outcomes are presented in Table 4.7.

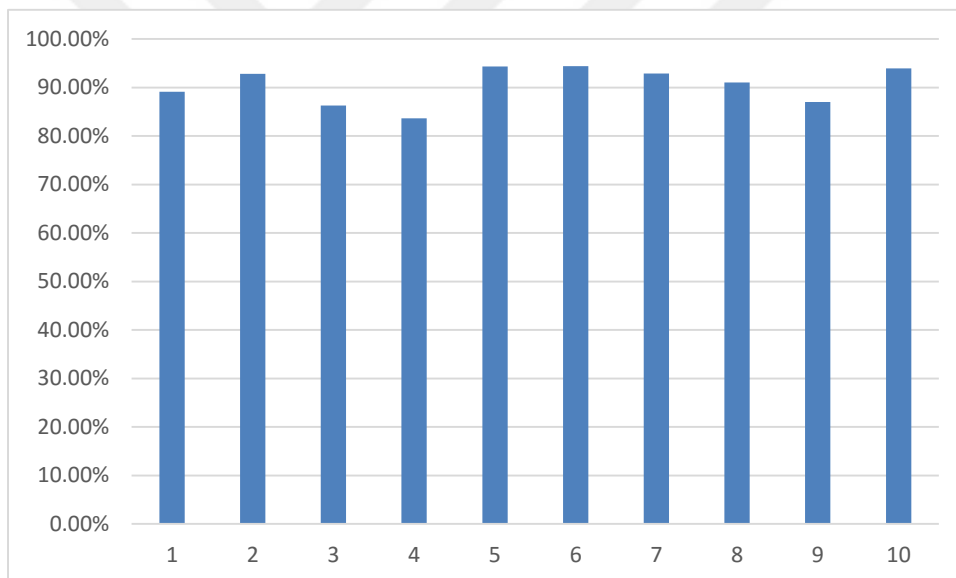
Table 4.7 Outcomes obtained for merchant copy text font in the third examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	64.89 %	89.11 %	6.73s
2	141	725	76.59 %	92.82 %	6.18s
3	151	839	66.22 %	86.29 %	6.37s
4	163	715	66.87 %	83.63 %	5.67s
5	153	722	81.04 %	94.32 %	5.93s
6	173	930	76.30 %	94.40 %	7.36s
7	159	802	74.84 %	92.89 %	6.25s
8	157	929	73.88 %	91.06 %	6.87s
9	168	888	63.09 %	87.04 %	6.32s
10	158	878	79.15 %	93.96 %	5.76s
Avg.	161	840	72.28 %	90.55 %	6.34s

The results of this examination were 72.28 % for word accuracy percentage and 90.55 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 6.34 sec. This examination proved that suggested Tesseract improvement techniques greatly improve recognition rates, and also these techniques are more important than some of the image processing algorithms. Because when some of the image processing algorithms are ignored, the results decreased by 6.40 % and 3.39 % respectively for word and character accuracy percentages, but ignoring Tesseract improvement techniques decreased the results by 16.46 % and 5.59 % respectively for word and character accuracy percentages. The histogram of outcomes for both word and character accuracy percentages are presented in the Figure 4.11.



a) Word accuracy histogram.



b) Character accuracy histogram.

Figure 4.11 Word and character rates histogram for merchant copy font in the third case.

D. Examination 4

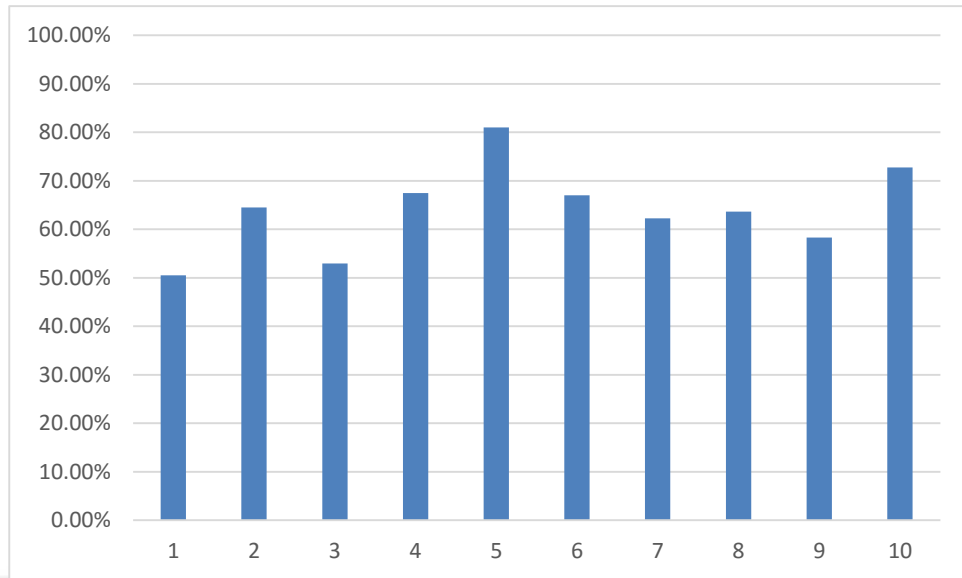
The capability of the application is examined in this process for detecting Merchant copy text font by ignoring all the techniques suggested for improving the recognition rates. Techniques suggested are all the image processing algorithms (the process of contrast, the process of gray-scale and so forth) and all the Tesseract improvement techniques

(introducing new fonts to the Tesseract, introducing new dictionary to the Tesseract OCR engine and so on). The experimental outcomes are presented in Table 4.8.

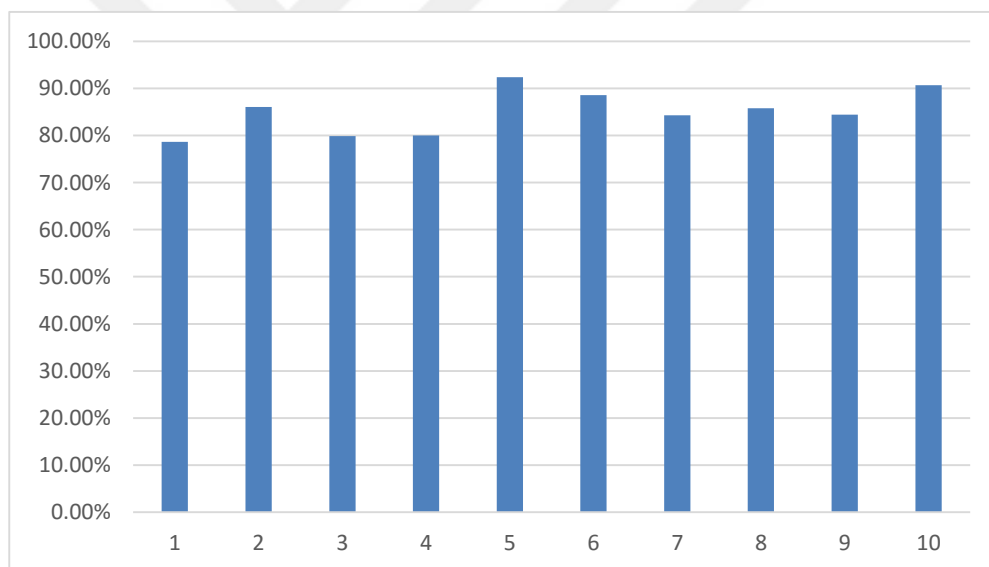
Table 4.8 Outcomes obtained for merchant copy text font in the fourth examination.

No. Of image	Word and character numbers		Percentages of accuracy		Performance in time (sec)
	Word no.	Character no.	Accuracy of words	Accuracy of characters	
1	188	974	50.53 %	78.64 %	5.21s
2	141	725	64.53 %	86.06 %	4.95s
3	151	839	52.98 %	79.85 %	4.53s
4	163	715	67.48 %	80.00 %	5.73s
5	153	722	81.04 %	92.38 %	5.13s
6	173	930	67.05 %	88.60 %	5.07s
7	159	802	62.26 %	84.28 %	4.41s
8	157	929	63.69 %	85.79 %	5.84s
9	168	888	58.33 %	84.45 %	4.61s
10	158	878	72.78 %	90.66 %	4.21s
Avg.	161	840	64.06 %	85.07 %	4.96s

The results of this examination were 64.06 % for word accuracy percentage and 85.07 % for character accuracy percentage. The performance in terms of time of the suggested Android application in this examination is 4.96 sec. This examination proves that suggested techniques have great impacts for promoting the recognition rates. Because the results of this examination decreased by 24.68% and 11.07% respectively for word and character accuracy percentages when compared with the results of the first examination. This shows that instantly submitting receipt images to Tesseract for recognition of text without utilizing techniques discussed will result in less than satisfactory results. The histogram of outcomes for both word and character accuracy percentages are presented in Figure 4.12.



a) Word accuracy histogram.



b) Character accuracy histogram.

Figure 4.12 Word and character rates histogram for merchant copy font in the fourth case.

4.2.5. Evaluation of outcomes experienced

This section shows outcome percentages experienced for two different fonts. All the results obtained are summarized in table 4.9. Where word accuracy percentage is denoted by W%, character accuracy percentage is denoted by C% and performance in term of time is denoted by Ts. Also examination1, examination2, examination3, and examination4 are

denoted by E1, E2, E3 and E4 respectively. The outcome percentages of all four examinations are shown for both fonts, finally average number for both fonts are computed.

Table 4.9 Average percentages of outcomes obtained for two different fonts.

Text Fonts	Capability Metrics	Examinations			
		E1	E2	E3	E4
Results for Fake receipt text font	W%	88.71%	79.69%	65.15%	52.07%
	C%	97.08%	91.54%	89.85%	83.22%
	Ts	6.51	5.51	6.27	5.33
Results for Merchant copy text font	W%	88.74%	82.34%	72.28%	64.06%
	C%	96.14%	92.75%	90.55%	85.07%
	Ts	6.62	5.67	6.34	4.96
Both text fonts average	W%	88.72%	81.01%	68.71%	58.06%
	C%	96.61%	92.14%	90.20%	84.14%
	Ts	6.56	5.59	6.30	5.14

The suggested Android application was tested to detect and recognize two different text fonts on the receipt images, we proved that suggested Android application can recognize both fonts properly. In the first examination in both experiments (fake receipt font experimental outcomes and merchant copy experimental outcomes), the suggested Android application was examined by utilizing all the algorithms and techniques selected for improving the recognition rates by Tesseract. With utilizing all the suggested techniques for both fonts, the suggested Android application produced 88.72 % as the percentage of word accuracy, 96.61 % as the percentage of character accuracy and 6.56 sec as the time performance of the suggested Android application for processing a single receipt image.

The recognition rates were improved by 20.01 % and 6.41 % respectively for word and character accuracy by suggesting and applying the Tesseract improvement techniques on receipt images, the techniques are such as introducing new fonts to the Tesseract,

introducing new dictionary to the Tesseract, selecting a suitable page segmentation method and Tesseract accuracy improving by regex functions. Also, the recognition rates were improved by 7.71 % and 4.47 % respectively for word and character accuracy by performing some of the image processing algorithms on the receipt images, the algorithms of image processing are such as process of contrast, process of gray-scale, process of thresholding, reduction of noise, morphological operations and operation of de-skewing.

Instantly submitting receipt images to the Android application with discarding all the techniques suggested in this research, produces useless outcomes which are 58.06% as the percentage of word accuracy and 84.14% as the percentage of character accuracy. Behind improving Tesseract outcomes on the receipt image by suggesting and applying several different techniques, we also proved and showed that instantly submitting receipt images to the Tesseract for text recognition gave useless and bad outcomes. The proposed techniques improved the recognition rate in this research by 30.66 % and 12.47% respectively for word and character accuracy.

5. CONCLUSION AND FUTURE WORKS

In this research, OCR technology for handling the problem of applying OCR on receipt images was used and studied in detail. Optical character recognition is a process which takes images as inputs and generates the texts contained in the input image. It is the technology that converts the text from handwritten images, text printed images or scanned images to the alterable text for further analysis and process. This research suggested an OCR system that integrated OCR technology with the hand-held devices. Various techniques, algorithms of image processing and OCR techniques and engines are studied in order to suggest and innovate a new OCR application for tracking daily marketing receipts easily. A comprehensive literature review of the OCR field is described such as First, the main issues associated with the images acquired through cameras of mobile device and OCR techniques are described, second, several different usages of OCR technology in different fields are discussed, third, the main pipeline of steps and techniques requires for designing OCR applications are categorized and discussed, finally, recent and powerful proposed OCR engines are listed and discussed.

Five main processes concluded the suggested Android application, processes are receipt region detection, image enhancement and preprocessing, applying Tesseract OCR engine, regular expression and database storing. First, the suggested Android application applies an algorithm for identifying the region of receipt from the background of images called canny edge detection (CED) algorithm. Then the suggested system applies a series of the techniques and methods of image processing for smoothing images such as contrast process, gray-mode operation, Thresholding, noise reduction, morphological operations and finally de-skewing process. In the recognition stage, the proposed Android application will apply Tesseract OCR engine for extracting and recognizing text on the receipt images. The study found that there are various techniques for raising recognition rates by Tesseract. The improvements are such as training Tesseract for the new fonts, improving by regex (regular expression), selecting suitable page segmentation method and finally improving by introducing new dictionary to the Tesseract.

20 receipt images were utilized in this research for testing the capability of the proposed Android application. Up to 188 words and up to 974 characters were included in the receipts. For the purpose of showing the many benefits of using these techniques and

methods applied in this research, the proposed Android application under four various examinations was examined. By utilizing all the proposed techniques for two fonts, the suggested Android application yielded 88.72 % as the percentage of word accuracy, 96.61 % as the percentage of character accuracy and 6.56 sec as the time performance of the suggested Android application. The research shows that using the Tesseract engine will greatly improve the recognition of receipt images. The study shows that instantly submitting receipt images to the Tesseract without applying various techniques proposed in this research will produce less than satisfactory results which are 58.06% as the percentage of word accuracy and 84.14% as the percentage of character accuracy. Finally, four queries are designed to fulfill the main concept and exceptional aims of the research. The queries are expend inspector, receipt image discovering, determining the amount of money expended from one date to another date, determining the amount of money expended for a particular item from one date to another date.

For the future work, proposed Android application can be improved in various ways such as studying and investigating additional methods and techniques for improving the recognition rates on receipt images, proposing an OCR application for recognizing text on receipt images surrounded by any shape or format, recognizing text in other languages on receipt images such as Arabic language or Turkish language.

6. REFERENCES

- [1] **Duda, R.O., Hart, P.E. and Stork, D.G.,** 2012. Pattern classification. John Wiley & Sons.
- [2] **Omee, F.Y., Himel, S.S., Bikas, M. and Naser, A.,** 2012. A complete workflow for development of bangla OCR. arXiv preprint arXiv:1204.1198.
- [3] **Shinde, A.A. and Chougule, D.G.,** 2012. Text pre-processing and text segmentation for OCR. International Journal of Computer Science Engineering and Technology, pp.810-812.
- [4] **Patel, C., Patel, A. and Patel, D.,** 2012. Optical character recognition by open source OCR tool tesseract: A case study. International Journal of Computer Applications, 55(10).
- [5] **Dervisevic, I.,** 2006. Machine Learning Methods for Optical Character Recognition.
- [6] **Chang, Y., Chen, D., Zhang, Y. and Yang, J.,** 2009. An image-based automatic Arabic translation system. Pattern Recognition, 42(9), pp.2127-2134.
- [7] **Number of mobile phone users worldwide from 2013 to 2019 (in billions).** [Online]. Available at: <https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/> (Accessed on 7 Jan. 2017).
- [8] **Ye, Q. and Doermann, D.,** 2015. Text detection and recognition in imagery: A survey. IEEE transactions on pattern analysis and machine intelligence, 37(7), pp.1480-1500.
- [9] **Feild, J. and Learned-Miller, E.G.,** 2012. Scene text recognition with bilateral regression. UMass Amherst Technical Report.
- [10] **Trier, O.D. and Jain, A.K.,** 1995. Goal-directed evaluation of binarization methods. IEEE transactions on Pattern analysis and Machine Intelligence, 17(12), pp.1191-1201.
- [11] **Sauvola, J., Seppanen, T., Haapakoski, S. and Pietikainen, M.,** 1997, August. Adaptive document binarization. In Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on (Vol. 1, pp. 147-152). IEEE.
- [12] **Jain, A., Dubey, A., Gupta, R., Jain, N. and Tripathi, P.,** 2013. Fundamental challenges to mobile based ocr. vol, 2, pp.86-101.

- [13] **Otsu, N.**, 1979. A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), pp.62-66.
- [14] **Deskew command line from Imagemagick library.** [Online]. Available at: <https://www.imagemagick.org/script/command-line-options.php#deskew> (Accessed on 19 Jan. 2017).
- [15] **Rotation/Deskewing for improving quality of images for OCR.** [Online]. Available at: <https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality#rotation--deskewing> (Accessed on 19 Jan. 2017).
- [16] **Qi, X.Y., Zhang, L. and Tan, C.L.**, 2005, August. Motion deblurring for optical character recognition. In Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on (pp. 389-393). IEEE.
- [17] **Liu, J., Li, H., Zhang, S. and Liang, W.**, 2011, September. A novel italic detection and rectification method for Chinese advertising images. In Document Analysis and Recognition (ICDAR), 2011 International Conference on (pp. 698-702). IEEE.
- [18] **Fake Receipt Font.** [Online]. Available at: <http://www.1001fonts.com/fake-receipt-font.html> (Accessed on 20 Jan. 2017).
- [19] **Merchant Copy Font Family.** [Online]. Available at: <http://www.1001fonts.com/merchant-copy-font.html> (Accessed on 20 Jan. 2017).
- [20] **Plamondon, R. and Srihari, S.N.**, 2000. Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on pattern analysis and machine intelligence, 22(1), pp.63-84.
- [21] **Ganis, M.D., Wilson, C.L. and Blue, J.L.**, 1998. Neural network-based systems for handprint OCR applications. IEEE Transactions on Image Processing, 7(8), pp.1097-1112.
- [22] **Gossweiler, R., Kamvar, M. and Baluja, S.**, 2009, April. What's up CAPTCHA?: a CAPTCHA based on image orientation. In Proceedings of the 18th international conference on World wide web (pp. 841-850). ACM.
- [23] **Gao, J., Blasch, E., Pham, K., Chen, G., Shen, D. and Wang, Z.**, 2013, May. Automatic vehicle license plate recognition with color component texture detection and template matching. In SPIE Defense, Security, and Sensing (pp. 87390Z-87390Z). International Society for Optics and Photonics.

- [24] **Dong, W. and Shisheng, Z.**, 2008, December. Color image recognition method based on the Prewitt operator. In Computer Science and Software Engineering, 2008 International Conference on (Vol. 6, pp. 170-173). IEEE.
- [25] **Canny, J.**, 1986. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, (6), pp.679-698.
- [26] **Wang, X.**, 2007. Laplacian operator-based edge detectors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(5), pp.886-890.
- [27] **Senthilkumaran, N. and Rajesh, R.**, 2009. Edge detection techniques for image segmentation—a survey of soft computing approaches. International journal of recent trends in engineering, 1(2), pp.250-254.
- [28] **Doronicheva, A.V., Socolov, A.A. and Savin, S.Z.**, 2014. Using Sobel operator for automatic edge detection in medical images. Journal of Mathematics and System Science, 4(4).
- [29] **Finding blocks of text in an image using Python, OpenCV and numpy.** [Online]. Available at: <http://www.danvk.org/2015/01/07/finding-blocks-of-text-in-an-image-using-python-opencv-and-numpy.html> (Accessed on 24 Jan. 2017).
- [30] **Kaur, S., Mann, P.S. and Khurana, S.**, 2013. Page Segmentation in OCR System-A Review. International Journal of Computer Science and Information Technologies, 4(3), pp.420-2.
- [31] **Akram, M. and Hussain, S.**, 2010, August. Word segmentation for urdu OCR system. In Proceedings of the 8th Workshop on Asian Language Resources, Beijing, China (pp. 88-94).
- [32] **Shinde, A.A. and Chougule, D.G.**, 2012. Text pre-processing and text segmentation for OCR. International Journal of Computer Science Engineering and Technology, pp.810-812.
- [33] **Namboodiri, A.M. and Jain, A.K.**, 2004. Online handwritten script recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(1), pp.124-130.

- [34] **Khandelwal, A., Choudhury, P., Sarkar, R., Basu, S., Nasipuri, M. and Das, N.,** 2009, December. Text line segmentation for unconstrained handwritten document images using neighborhood connected component analysis. In International Conference on Pattern Recognition and Machine Intelligence (pp. 369-374). Springer Berlin Heidelberg.
- [35] **Basu, S., Chaudhuri, C., Kundu, M., Nasipuri, M. and Basu, D.K.,** 2007. Text line extraction from multi-skewed handwritten documents. *Pattern Recognition*, 40(6), pp.1825-1839.
- [36] **Saha, S., Basu, S., Nasipuri, M. and Basu, D.K.,** 2010. A Hough transform based technique for text segmentation. arXiv preprint arXiv:1002.4048.
- [37] **Trier, Ø.D., Jain, A.K. and Taxt, T.,** 1996. Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4), pp.641-662.
- [38] **Sharma, O.P., Ghose, M.K., Shah, K.B. and Thakur, B.K.,** 2013. Recent trends and tools for feature extraction in OCR technology. *International Journal of Soft Computing and Engineering*, 2(6), pp.220-223.
- [39] **Pradeep, J., Srinivasan, E. and Himavathi, S.,** 2011, April. Diagonal based feature extraction for handwritten character recognition system using neural network. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* (Vol. 4, pp. 364-368). IEEE.
- [40] **Suen, C.Y.,** 1986. Character recognition by computer and applications. *Handbook of pattern recognition and image processing*, pp.569-586.
- [41] **Rehman, A. and Saba, T.,** 2014. Neural networks for document image preprocessing: state of the art. *Artificial Intelligence Review*, 42(2), pp.253-273.
- [42] **SABA, T.,** 2012. OFFLINE CURSIVE TOUCHED SCRIPT NON-LINEAR SEGMENTATION (Doctoral dissertation, Universiti Teknologi Malaysia).
- [43] **Rehman** (2010) Offline cursive character recognition based on heuristics techniques. PhD thesis, Universiti Teknologi Malaysia, pp 80–85.
- [44] **Singh, S. and Hewitt, M.,** 2000. Cursive digit and character recognition in CEDAR database. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on* (Vol. 2, pp. 569-572). IEEE.

- [45] **Blumenstein, M., Liu, X.Y. and Verma, B.,** 2007. An investigation of the modified direction feature for cursive character recognition. *Pattern Recognition*, 40(2), pp.376-388.
- [46] **Blumenstein, M., Liu, X.Y. and Verma, B.,** 2004, July. A modified direction feature for cursive character recognition. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on* (Vol. 4, pp. 2983-2987). IEEE.
- [47] **Vamvakas, G., Gatos, B., Pratikakis, I., Stamatopoulos, N., Roniotis, A. and Perantonis, S.J.,** 2007, February. Hybrid off-line OCR for isolated handwritten Greek characters. In *The Fourth IASTED International Conference on Signal Processing, Pattern Recognition, and Applications (SPPRA 2007)* (pp. 197-202).
- [48] **Dongre, V.J. and Mankar, V.H.,** 2011. A review of research on Devnagari character recognition. *arXiv preprint arXiv:1101.2491*.
- [49] **Yetirajam, M., Nayak, M.R. and Chattopadhyay, S.,** 2012. Recognition and classification of broken characters using feed forward neural network to enhance an OCR solution. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* Volume, 1.
- [50] **Shamsher, I., Ahmad, Z., Orakzai, J.K. and Adnan, A.,** 2007, August. OCR for printed urdu script using feed forward neural network. In *Proceedings of World Academy of Science, Engineering and Technology* (Vol. 23, pp. 172-175).
- [51] **Zhai, X., Bensaali, F. and Sotudeh, R.,** 2012, July. OCR-based neural network for ANPR. In *Imaging Systems and Techniques (IST), 2012 IEEE International Conference on* (pp. 393-397). IEEE.
- [52] **Shah, P., Karamchandani, S., Nadkar, T., Gulechha, N., Koli, K. and Lad, K.,** 2009, November. OCR-based chassis-number recognition using artificial neural networks. In *Vehicular Electronics and Safety (ICVES), 2009 IEEE International Conference on* (pp. 31-34). IEEE.
- [53] **Official website of GOCR. [Online].** Available at: <http://jocr.sourceforge.net/> (Accessed on 5 Feb. 2017).
- [54] **Official website of Ocrad. [Online].** Available at: <https://www.gnu.org/software/ocrad/> (Accessed on 5 Feb. 2017).
- [55] **Official website of OCRopus. [Online].** Available at: <https://github.com/tmbdev/ocropy> (Accessed on 5 Feb. 2017).

- [56] **Breuel, T.M.**, 2008, January. The OCRopus open source OCR system. In Electronic Imaging 2008 (pp. 68150F-68150F). International Society for Optics and Photonics.
- [57] **Khan, M.N.H., Siddiqui, F. and Das, A.**, 2014. Pin number detection from mobile phone scratch card using OCR on android platform and build an application for balance recharge (Doctoral dissertation, BRAC University).
- [58] **Official website of Tesseract OCR. [Online].** Available at: <https://github.com/tesseract-ocr> (Accessed on 5 Feb. 2017).
- [59] **Schreiber, M., Poggenhans, F. and Stiller, C.**, 2014, October. Detecting symbols on road surface for mapping and localization using OCR. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on (pp. 597-602). IEEE.
- [60] **Documentation of Tesseract OCR. [Online].** Available at: <https://github.com/tesseract-ocr/tesseract/wiki/Documentation> (Accessed on 6 Feb. 2017).
- [61] **Training Tesseract OCR. [Online].** Available at: <https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract> (Accessed on 6 Feb. 2017).
- [62] **Mishra, N. and Patvardhan, C.**, 2012. ATMA: Android Travel Mate Application. International Journal of Computer Applications, 50(16).
- [63] **Bhaskar, S., Lavassar, N. and Green, S.**, 2007. Implementing Optical Character Recognition on the Android Operating System for Business Cards. Radical Eye Software., Palo Alto, CA.
- [64] **Kastelan, I., Kukolj, S., Pekovic, V., Marinkovic, V. and Marceta, Z.**, 2012, September. Extraction of text on TV screen using optical character recognition. In 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (pp. 153-156). IEEE.
- [65] **Smith, R.**, 2007, September. An overview of the Tesseract OCR engine. In Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (Vol. 2, pp. 629-633). IEEE.
- [66] **Official website of Imagemagick library. [Online].** Available at: <https://www.imagemagick.org/script/index.php> (Accessed on 3 Mar. 2017).

- [67] **Official website of ImageMagick library, command line options. [Online].** Available at: <https://www.imagemagick.org/script/command-line-options.php#deskew> (Accessed on 5 Mar. 2017).
- [68] **Official webpage of Tesseract OCR engine, command line usage. [Online].** Available at: <https://github.com/tesseract-ocr/tesseract/wiki/Command-Line-Usage> (Accessed on 6 Mar. 2017).
- [69] **Official webpage of Tesseract OCR engine, Training Tesseract. [Online].** Available at: <https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract> (Accessed on 10 Mar. 2017).
- [70] **JTessBoxEditor software. [Online].** Available at: <http://vietocr.sourceforge.net/training.html> (Accessed on 15 Feb. 2017).
- [71] **Android Studio, the Official IDE for Android. [Online].** Available at: <https://developer.android.com/studio/index.html> (Accessed on 20 Mar. 2017).
- [72] **ScanLibrary, an android document scanning library. [Online].** Available at: <https://github.com/jhansiredy/AndroidScannerDemo> (Accessed on 21 Mar. 2017).
- [73] **Composer, dependency manager for PHP. [Online].** Available at: <https://getcomposer.org/> (Accessed on 24 Mar. 2017).
- [74] **Official webpage of MPAndroidChart library. [Online].** Available at: <https://github.com/PhilJay/MPAndroidChart> (Accessed on 29 Mar. 2017).

CURRICULUM VITA

Karez Abdulwahhab Hamad

karez.abdulwahab@gmail.com

Nationality: Iraqi
Place of birth: Erbil-Iraq
Date of birth: 20 / 11 / 1991
Marital status: Single

EDUCATION

09/ 2015 – Present MSc (Master of Software Engineering), Firat University, Elazig, Turkey.
09/ 2009 – 07/ 2013 Bachelor degree from Salahaddin University, college of Engineering,
Software Engineering department. Iraq, Erbil.
07/ 2009 Graduated from (Shahid Dr. Abdulrahman) High School for Boys. Iraq,
Erbil, Soran.
07/ 2006 Graduated from (Balakyan) Primary School for Boys and girls. Iraq,
Erbil, Soran.

WORK EXPERIENCES

1. Software Engineer at Soran University. 01/2014-03/2015
2. IT Support at Cihan bank-Soran branch. 03/2015-09/2015
3. IT support and Accountant at Spiba Company for Building equipment. 08/2013-09/2015
4. Software Engineer and technical supporter at Iraqi National ID Card-Erbil. 01/2016-present

PUBLICATIONS

- [1] Hamad, K, Kaya, M. (2016). A Detailed Analysis of Optical Character Recognition Technology. International Journal of Applied Mathematics, Electronics and Computers, 4 (Special Issue-1), 244-249. DOI: 10.18100/ijamec.270374.