An introduction and application of optical character recognition

James Forrest Bankston
Graduate Computer Science Student
Troy University
Troy, AL
jbankston@troy.edu

***Abstract*— Optical Character Recognition (OCR)** is an amazing concept and research topic within Computer Vision. There are several ways of defining OCR but the one that makes the most sense to me is the electronic/mechanical conversion of images into machine-encoded text which can then be outputted to the user in some readable format. The concept behind this is that many people take images or scan important documents as a means of digitizing them. However, the biggest problem with this is that any text from the image is not directly digitized as well without some kind of extra software. There are many reasons why one would want for the text to be digitized that will be discussed. This paper will introduce a method to solving this problem, OCR. After OCR has been introduced, this paper will showcase my own personal implementation in python. I have made some modifications in pre-processing and accessibility to create my own customized OCR application.

Keywords—Optical Character Recognition, Computer Vision, Text Extraction, Image Pre-processing

I. Introduction

The core problem with images of text is that the text is not “usable.” This means that the text cannot be manipulated by the machine. There are many ways a machine can manipulate text such as popular user functions like copy and pasting or searching the text for specific values. Humans are very capable of recognizing and identifying text. However, humans cannot manipulate such text without the help of a machine. Consider a case in which a textbook has been scanned and digitized to be available on a computer. This is a very powerful tool but is not using a computer to its full potential. A scanned image alone only has the power of human vision without the manipulation of a computer. Under normal circumstances, a human would have to directly type out all the text from the textbook to have it in a usable format which is very time-consuming. With the powerful advances in technology we have seen over the recent years, tools such as OCR have emerged that provide a solution to this problem. OCR allows the manipulation of text through images by extracting the text from the image in some format for the user. This is a fantastic tool overall but runs into a few problems in practice.

The visual system of humans is very powerful. However, computers only have a visual system that we as humans can program them to have. Giving a computer a visual system, and more specifically OCR is an overall complex problem for many reasons. For example, there are many different fonts and styles that text can be written in. It would be quite impossible to directly code in every single font and style into a computer for recognition. Even then, handwritten characters can always be written differently, and new fonts can be added. Therefore, a computer must be programmed to recognize different fonts which is much harder for a machine than a human. Oftentimes additional grammar/spell-checking algorithms are applied as well but run into difficulties when facing images with text of varying languages. Overall, OCR is a solution to a problem that brings with it many more problems and solutions. The following sections will introduce OCR more in-depth along with the solutions to these problems and my own personal implementation and solutions.

II. The Problem

The problem of character recognition is by no means new and actually predates that of computers **[1].** Very early OCR systems were basically mechanical character readers that performed at very slow speeds combined with poor accuracy as well. One of the absolute earliest works of OCR is introduced in **[2]** describing a reading robot named GISMO from 1951. This device was capable of reading some musical notes as well as 23 characters of the alphabet. Similar devices were introduced in the 1950’s but were only capable of reaching speeds of up to roughly one character a minute while still suffering from poor accuracy. Therefore, not much attention was paid to these devices as they were infinitely better in theory than in practice. **Figure 1** below shows a model from the original patent for GISMO: According to the original patent for this device, it was intended for “Accordingly, an object of my invention is to provide

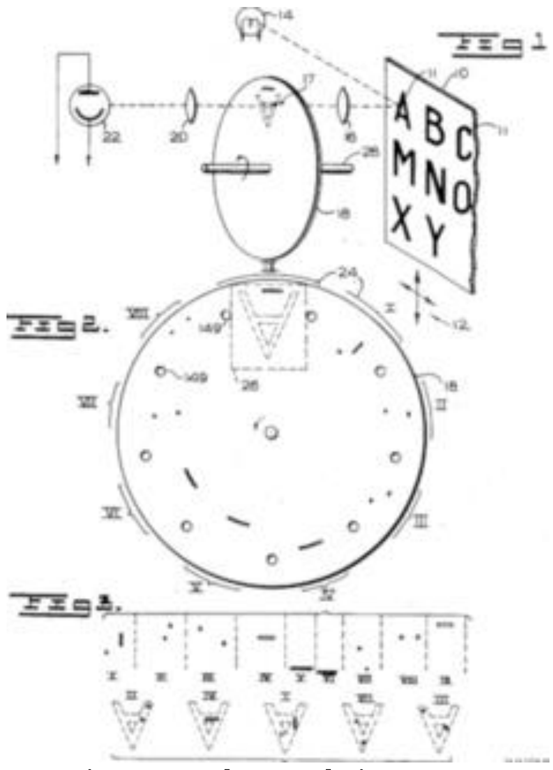


Figure 1: Early OCR device GISMO

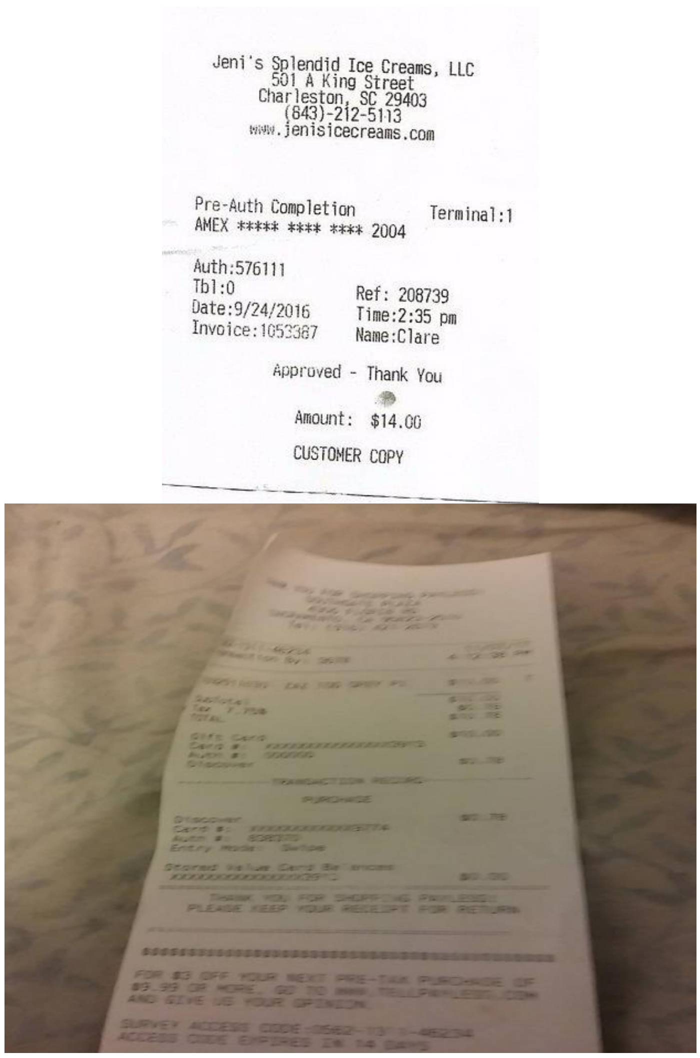
improved methods and apparatus for reading and interpreting printed or other information into various media."

These early OCR systems are very notable, but technology has come a long way since. More modern OCR research and development is generally split between two categories based on the actual input: handwriting recognition and printed character recognition. To put it simply, both of these categories are extremely useful and also have many varying applications in many different fields. However, overall, handwriting recognition is more challenging and computationally taxing. The reasoning is that a machine will probably have an easier time recognizing and classifying characters written by a machine rather than by a human. Handwritten characters have much more variance and are generally not written in a set font. With unending amounts of variance in human writing, it becomes increasingly challenging for a computer to classify. Handwriting character recognition is definitely possible but very challenging. However, this paper will focus on OCR methods for machine-printed text. OCR has overall contributed to a significant improvement of process in many different real-world applications. A good example of this would be the healthcare system in the United States, or theoretically any country. There is always a great amount of patient forms in these industries and they are oftentimes scanned into a computer for later data entry. Employing OCR in this case can reduce the process of data entry to almost nothing compared to without such a tool. OCR simplifies the whole process by turning the text from patient forms into easily editable and searchable data. The healthcare system is only one of many possible examples, such as a banking system, a billing agency, and many more. When using software in this way, we can call it software as a service **[3].** The different phases of OCR will now be given:

1. **Image Acquisition:** An image must first be acquired from some external source such as a camera or photo/document scanner
2. **Pre-processing:** Image pre-processing is easily one of the most important parts of OCR and consists of several steps in an attempt to improve the quality of the image for the purpose of accuracy. Pre-processing will be discussed and described more in-depth
3. **Character segmentation:** After pre-processing, characters are separated so that they can be individually processed
4. **Feature extraction:** Once the characters have been segmented, their features are extracted and analyzed for the next step
5. **Character classification:** After all the characters features have been individually extracted and analyzed, they can be classified based on their features
6. **Post-processing:** Some kind of natural language processing technique such as grammar/spell checkers can be applied after classification to improve accuracy.

III. Popular OCR Pre-processing Methods

As stated, OCR on machine written text is generally going to be easier, meaning that it will be more accurate and require less image pre-processing. Additionally, there are two different types of text in images that will be discussed: Structured text and Unstructured text. Structured text is generally in the form of a scanned document and consists of a good quality image without the presence of too much noise. Unstructured text is typically in the form of a photograph and consists of text on an image with a low or high presence of noise. In the context of OCR, noise can be many different things. For example, unaligned text, text that is too small, shadows on the text, etc. Unsurprisingly, structured text without much noise will generally be easier for OCR and require less pre-processing. An example of an image with structured text as well as an image with unstructured text is given below in **Figure 2:**


Jeni's Splendid Ice Creams, LLC
501 A King Street
Charleston, SC 29403
(843)-212-5113
www.jenisicecreams.com

Pre-Auth Completion Terminal:1
AMEX ***** **** **** 2004

Auth:576111
Tbl:0 Ref: 208739
Date:9/24/2016 Time:2:35 pm
Invoice:1053387 Name:Clare

Approved - Thank You

Amount: $14.00

CUSTOMER COPY

Figure 2:
Top: Structured Text (Scanned receipt)
Bottom: Unstructured Text (Photo of a receipt)

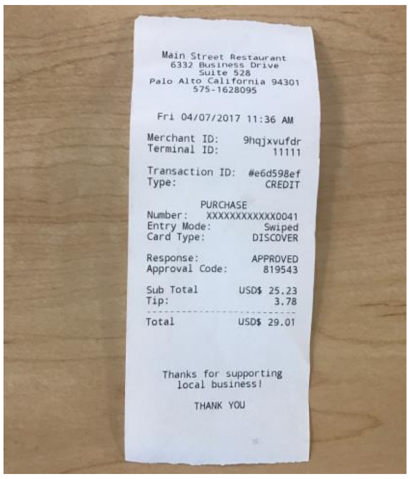
The top image is very suitable for OCR and would receive good accuracy with some slight skew correction. The bottom image, however, is quite poor. This image would need extensive pre-processing to extract the text and even then, would probably struggle with accuracy. Nevertheless, even if the system fails to fully recognize every phrase in the receipt, it is likely to provide some useful text that can be used in manipulation such as the total cost, the date of the transaction, or even the payment method used in the transaction. There are many countless real-life applications of OCR, however, before jumping into accurate recognition for these documents, it is important to understand some of the image pre-processing techniques to make these documents readable by a computer. This paper will now introduce some of the most popular methods used for pre-processing in OCR to obtain better quality images and therefore better accuracy.

A. *Resizing*

Oftentimes scanned images or regular images will be too small to get an accurate result from an OCR system. Zooming of the image on the important part of the text will help us get a more accurate result. However, increasing an image's size and enhancing it will be more computationally taxing than making it smaller.

B. *Grayscale*

Converting the image to grayscale is usually one of the first parts of pre-processing. Color is not a useful feature for extracting text with OCR and should therefore be discarded. Having a Grayscale image also makes another important pre-processing technique applied in the last few stages easier. An example image is shown in **Figure 3** to demonstrate some of the different phases of pre-processing. The image on top is the original image and will be edited through the section to show the effects of pre-processing.



Main Street Restaurant
6332 Business Drive
Suite 528
Palo Alto California 94301
575-1628095

Fri 04/07/2017 11:36 AM

Merchant ID: 9hqjxvufdr
Terminal ID: 11111

Transaction ID: #e6d598ef
Type: CREDIT

PURCHASE
Number: XXXXXXXXXXXX0041
Entry Mode: Swiped
Card Type: DISCOVER

Response: APPROVED
Approval Code: 819543

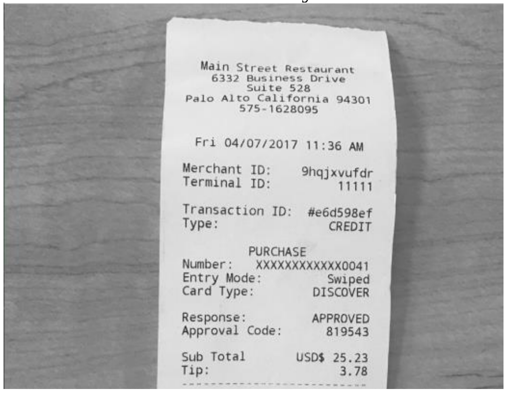
Sub Total USD$ 25.23
Tip: 3.78

Total USD$ 29.01

Thanks for supporting
local business!

THANK YOU

Resize and Grayscale



Main Street Restaurant
6332 Business Drive
Suite 528
Palo Alto California 94301
575-1628095

Fri 04/07/2017 11:36 AM

Merchant ID: 9hqjxvufdr
Terminal ID: 11111

Transaction ID: #e6d598ef
Type: CREDIT

PURCHASE
Number: XXXXXXXXXXXX0041
Entry Mode: Swiped
Card Type: DISCOVER

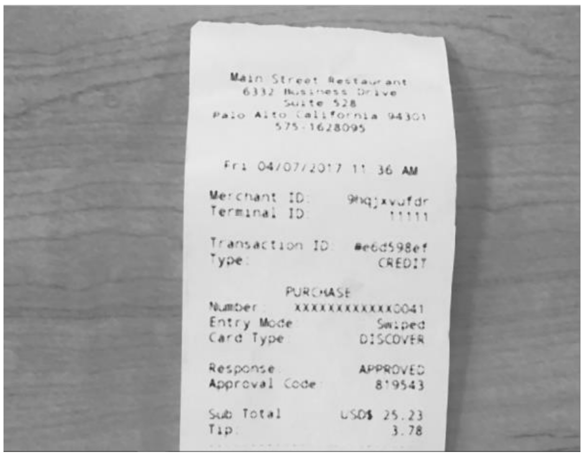
Response: APPROVED
Approval Code: 819543

Sub Total USD$ 25.23
Tip: 3.78

Figure 3: Original image vs. image after resizing and Grayscale

C. *Blurring*

Although it might seem odd, blurring an image can actually improve accuracy of character recognition. The blurred image might actually look more complicated to a human; however, we are attempting to make the image readable to a machine. There are several different types of popular blurring methods available such as average blurring, median blurring, and even gaussian blurring. Overall, blurring is generally a technique to remove noise that might be present in the image. The implementation shown below in **Figure 4** uses median blurring:



Main Street Restaurant
6332 Business Drive
Suite 528
Palo Alto California 94301
575-1628095

Fri 04/07/2017 11 36 AM

Merchant ID: 9hqjxvufdr
Terminal ID: 11111

Transaction ID: #e6d598ef
Type: CREDIT

PURCHASE
Number: XXXXXXXXXXXX0041
Entry Mode: Swiped
Card Type: DISCOVER

Response: APPROVED
Approval Code: 819543

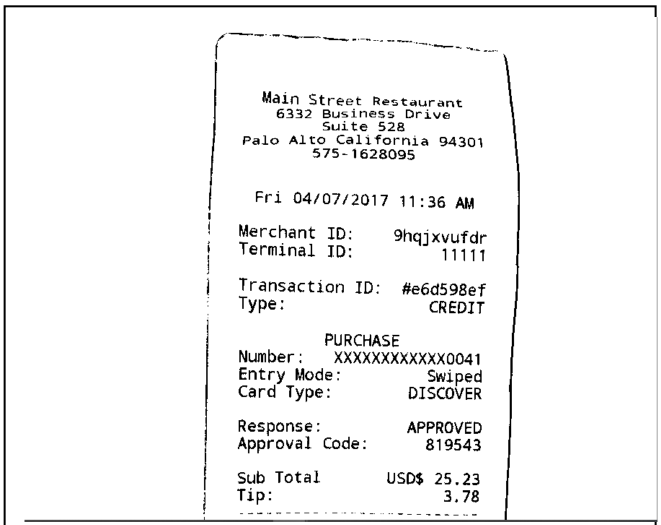
Sub Total USD$ 25.23
Tip: 3.78

Figure 4: Image after median blurring

D. *Binarization*

Image binarization is one of the absolute most important pre-processing techniques for OCR. Binarization creates a black and white version of an image in which the black pixels are considered to be the foreground, and white pixels are the background of the document. This turns the text on the image into something considerably easier to recognize for a computer. Global thresholding is a very popular version of binarization in which a grayscale intensity threshold is picked and then each pixel will be set to black or white depending on if it is lighter or darker than some set threshold **[4].**

Another extremely popular binarization method would be adaptive thresholding. This technique will try to compensate for an inconsistent contrast and brightness in images by selecting some threshold for every pixel that is based upon properties of a small part of the image that surrounds this pixel instead of the whole image. The final binarization method is one that I favor a lot and have chosen to use in my implementation of OCR: Otsu method. This is a very common of binarization that assumes two different classes of pixels being the foreground the background. A histogram of grayscale values from the image is used to choose the unique threshold that will maximize accuracy **[5].** This threshold is not guaranteed to be the best threshold but typically works well on clean documents or documents that have been scanned well with little to no noise. **Figure 5** below shows an example of Otsu thresholding. As you can see, the background of the image has been effectively removed and the text is now much more readable. This image could definitely be pre-processed slightly more but is much suitable for OCR now.

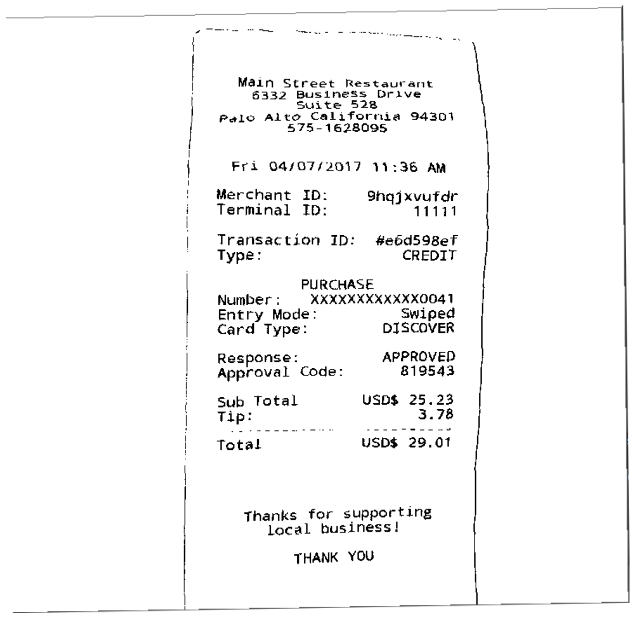


```
Main Street Restaurant
6332 Business Drive
Suite 528
Palo Alto California 94301
575-1628095

Fri 04/07/2017 11:36 AM

Merchant ID:        9hqjxvufdr
Terminal ID:             11111

Transaction ID:     #e6d598ef
Type:                   CREDIT

          PURCHASE
Number:     XXXXXXXXXXXX0041
Entry Mode:             Swiped
Card Type:            DISCOVER

Response:             APPROVED
Approval Code:          819543

Sub Total           USD$ 25.23
Tip:                      3.78
```

Figure 5: Image after Otsu thresholding and almost ready for character recognition

E. *Skew Correction (Deskewing)*

Skew correction is a pre-processing method that can either be applied towards the end or beginning of the process. Before character recognition, it is very important to ensure that the image is not incorrectly aligned and that the text is lined up evenly at a 180-degree angle. The process of skew correction typically involves rotating the image until the text matches up to this angle. Images should not need very much skew correction as long as they are scanned or photographed well. This image after binary thresholding would likely receive fairly good accuracy without the skew correction. However, as you can see the top of the receipt especially could use some skew correction. Many other images could possibly need much more skew correction than this, and therefore it should almost always be a part of an OCR pre-processing list **[1]. Figure 6** shows the final image ready for OCR after skew correction:



```
Main Street Restaurant
6332 Business Drive
Suite 528
Palo Alto California 94301
575-1628095

Fri 04/07/2017 11:36 AM

Merchant ID:        9hqjxvufdr
Terminal ID:             11111

Transaction ID:     #e6d598ef
Type:                   CREDIT

          PURCHASE
Number:     XXXXXXXXXXXX0041
Entry Mode:             Swiped
Card Type:            DISCOVER

Response:             APPROVED
Approval Code:          819543

Sub Total           USD$ 25.23
Tip:                      3.78

Total               USD$ 29.01


   Thanks for supporting
      local business!

        THANK YOU
```

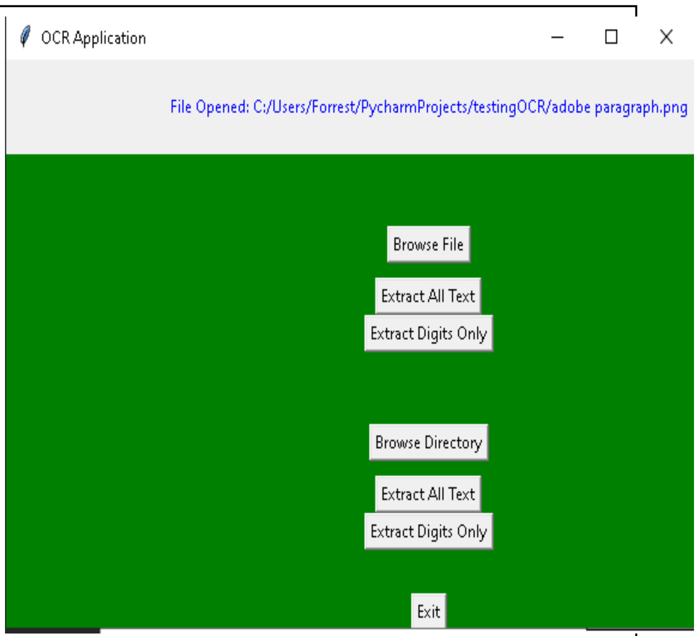
Figure 6: De-skewed image ready for OCR

IV. My OCR Implemenation

My goals for this project were as follows:

- Use what I have learned from this course and my own research to create an OCR application in python without using too many libraries
- Implement my own code in order to make my own blend of pre-processing and customizable options for the user
- The application should be extremely accurate on scanned images and moderately accurate on images from a camera after pre-processing methods

I used pytesseract, the python wrapper for tesseract which is an optical character recognition system. I used a combination of OpenCV and my own python logic for pre-processing. After pre-processing and text extraction, the contents of the extraction will then be written to a text file which can be manipulated. The user also has the option of extracting only digits from the text, and even extracting text from a whole folder of images and appending everything to the same file. This is done cleanly with a Tkinter GUI. An overview of my OCR application is given below in **Figure 7:**



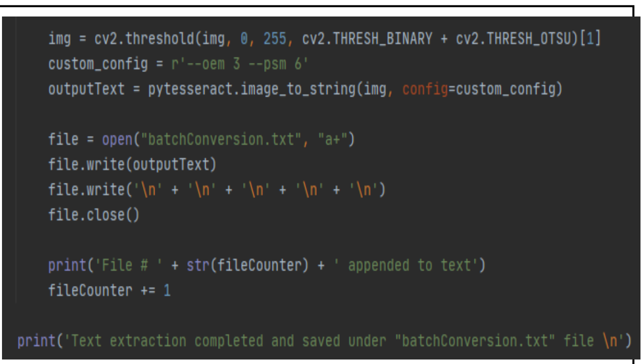
File Opened: C:/Users/Forrest/PycharmProjects/testingOCR/adobe paragraph.png

Figure 7: GUI of my OCR Application

As you can see, there are initially several options available to the user. There are two main buttons that can be selected based on the user's plans of operation. If the user would like to only perform OCR on a single image, then they can simply click on "Browse File." If they would like to perform OCR on a whole folder of images at once, the user can click on "Browse Directory." After the user has picked either a file or a folder for conversion, the file path to the object will be displayed at the top of the application. To begin OCR and extract all text from an image or folder of images, the user can then click "Extract All Text." If the user decides to only perform OCR on a single image, then the text content from performing OCR will be written to a file with the same name as the image in the same folder. If the user decides to perform OCR on a whole folder of images at once, then a text file will be created in this same folder with the name “batchConversion.txt.” This text file will have all the contents of OCR from the folder conversion appended to the same text file and can be used with a variety of purposes. There is an additional option available to the user under the “Extract Digits Only” button. This button will of course perform the same OCR functionality but only extract numerical digits from the text instead. There are many cases in which only digits would want to be extracted and not all text such as when an ID number is being searched for.

A. *Example of my Code*

Note that the following example does not include comments in order to save space, but that the actual code in my submission includes many comments. The following is an example of my code showing part of the process of the “Extract All Text” button under the “Browse Directory” button. This code shows the ending process of pre-processing using Otsu thresholding, followed by the text extraction with pytesseract. Following this is a file being created/opened with the name “batchConversion.txt” and appending the extracted text to this file. There is some formatting to separate text from different images, and then the file is closed. The system then prints the file counter variable which is incremented by one for each file appended to the text file. After the process is finished and the for loop is escaped, the system prints that the process is complete and that the file is printed. **Figure 8** contains this example of



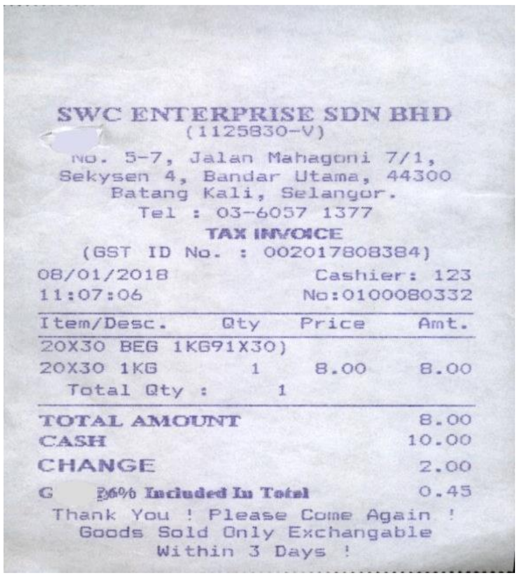
```
    img = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]
    custom_config = r'--oem 3 --psm 6'
    outputText = pytesseract.image_to_string(img, config=custom_config)

    file = open("batchConversion.txt", "a+")
    file.write(outputText)
    file.write('\n' + '\n' + '\n' + '\n' + '\n')
    file.close()

    print('File # ' + str(fileCounter) + ' appended to text')
    fileCounter += 1

print('Text extraction completed and saved under "batchConversion.txt" file \n')
```

Figure 8: Example from my code (no comments)

OCR on a whole folder:

B. *Real-Life Example with results*

To do a real-life test case of my OCR application, I used a subsection of the dataset from the IDCAR 2019 Robust Reading Challenge on Scanned Receipts. The subsection of the dataset that I used along with my results are included with my project submission. A link to the competition page will be included in my project submission as well. The receipts in this dataset are of varying qualities. A large percentage of them are scanned receipts of a good quality with little to no noise. However, a fair percentage of the receipts either have several sources of noise or are not scanned well. The purpose of testing the application on a dataset like this is because it is a very real application. Consider a family that has collected all of their various receipts over the course of two years. They have done a good job of digitizing these receipts and saving them in their computer, however, around 25% are either not scanned well or have some other sources of noise. In theory, my application should receive very good results on the images of good quality while maintaining solid accuracy on the worse images by capturing the most important characteristics. Let us show an example to display the power of my OCR application on a dataset. Consider the pre-processed receipt given on the left in **Figure 9.**



```
        SWC ENTERPRISE SDN BHD
             (1125830-V)
     No. 5-7, Jalan Mahagoni 7/1,
   Sekysen 4, Bandar Utama, 44300
       Batang Kali, Selangor.
         Tel : 03-6057 1377
            TAX INVOICE
   (GST ID No. : 002017808384)
08/01/2018                Cashier: 123
11:07:06               No:0100080332
Item/Desc.      Qty     Price     Amt.
20X30 BEG 1KG91X30)
20X30 1KG         1      8.00     8.00
   Total Qty :      1
TOTAL AMOUNT                      8.00
CASH                             10.00
CHANGE                            2.00
G      6% Included In Total       0.45
 Thank You ! Please Come Again !
   Goods Sold Only Exchangable
         Within 3 Days !
```

Figure 9: Pre-processed example

This receipt is quite simple but has plenty of useful data that we can extract from it. Of course, we begin by opening the application and browsing to the file. After performing OCR, the extracted text is in a text file with the same name in the same folder. The text output after performing OCR on this receipt is given below in **Figure 10:**

```
SWC ENTERPRISE SDN BHD
- (1125830-V)

nO. 3-7, Jalan Mahagoni 7/1,
Sekysen 4, Bandar Utama, 44300

Batang Kali, Selangor.
Tel : 03-6037 1377
TAX INVOICE
(GST ID No. : 002017808384)

08/01/2018 Cashier: 123
11:07:06 No #0100080332
Item/Desc. Qty Price Ant.
20X30 BEG 1KG91X30)
20X30 1KG 1 8.09 B.00

Total Qty : i
TOTAL AMOUNT 8.00
CASH 10.00
CHANGE 2.00
G 26% Induded In Tetz] 0.45
Thank You ! Please Come Again !

Goods Sold Only Exchangable

Within 3 Days ! /
```

Figure 10: Output text after my OCR

As you can see, we have successfully pre-processed the image and outputted it into a text format. As you will also notice quite quickly, this text extraction and output is not completely perfect by any means. You will see that my OCR application made a few errors such as for "Total Qty," my system recognized a "1" as an "i." My application also made a mistake when spelling the word "included." However, this part is written in a much different font than the rest of the receipt which makes it somewhat more understandable. Overall, however, this application was extremely successful. My system correctly recognized the location from which the receipt was printed along with other important details such as location, telephone number, and date of receipt. My application was also able to successfully capture the cashier number from the receipt which is a commonly used feature when logging receipts. The application also captured the payment method used from this receipt, being cash. The cash amount that was used and change given was captured as well. All of these features are incredibly important to any business owner or consumer.

Now that we have shown an example from just one receipt in this dataset, let us showcase an example from testing my OCR application throughout this whole dataset. As stated, I used a subsection of the "IDCAR 2019 Robust Reading Challenge on Scanned Receipts OCR and Information Extraction" dataset of 359 receipts for this test. A link to this dataset can be found here: https://rrc.cvc.uab.es/?ch=13&com=downloads

This dataset is amazing for testing purposes because it has many instances, and they are not all of perfect quality. The receipt used above in **Figure 9** would be considered one of the worse images from the dataset so it is safe to say that my OCR application should perform even better on most of the images. The execution time of my application on this dataset was fairly length but is not too crucial for determining the applications overall success as long as it is not horrendous. There are many examples from this dataset, such as **Figure 9** above which slow the execution time due to requiring more pre-processing. Images with more noise may require several iterations of pre-processing before text extraction which can slow the program significantly. Nevertheless, my program finally completed its execution on the dataset and brought great results. The "batchConversion.txt" from this process will be uploaded in the same folder as my project submission for viewing results. Upon scrolling through the extracted text, you will see that it is overall quite accurate to the original images.

Once again, consider this a case in which a small business owner has collected all of these receipts over the course of several months. A small business owner could have a situation in which a customer is attempting to dispute some charge made at the business. For proof, the customer has brought in their receipt from that day with the claimed faulty charges. A business owner would not accept the customer's claim at face value, it would be important to have proof. However, how could this business owner have proof? Even if the business owner has the physical receipts, or copies of them scanned onto the computer, it would be nearly impossible to find the actual receipt from that day by hand. Therefore, the business owner could apply OCR to this folder of images to extract the text. The business owner could then search certain keywords from the receipt such as "ID NO" or "Invoice NO" to locate the receipt. After finding the receipt, the business owner could then have definitive proof on whether or not the customer has a legitimate claim.

C. *Testing my OCR vs. an Online OCR App*

I definitely wanted to test my own application of OCR vs a popular and accessible method out there. The tool I used to for test results against my own method is the OCR tool available at: https://ocr.space/

For testing purposes, I used the same image of a receipt shown in **Figure 9.** I used the default settings, except that I did enable receipt scanning mode for maximum accuracy. The OCR output from this online OCR application is given below in **Figure 11.** As you will see from the figure, the results are overall quite similar. The most interesting part to me is that the online OCR application definitely made mistakes when recognizing this image. The results were similar in quantity to my own OCR application but quite different. Overall, however, this online application also captured the most important parts of the image such as cashier number, cash amount, etc.

```
            SWC ENTERPRISE SDN
            5—7, Mahagcjni 7/1,
                    4, 44300
            Patang Kali, Selangor.
            Tel : 03-6057 1377
                 TAX INVCCE
       (GST ID No. : 002017808384)
       08/01/2018          Cashier: 123
       I tem/Desc          Qty Price
            20130 BEG 1KG91X30)
  20130 IKG          1            8.00       8.00
            Total Qty :           1
            TOTAL AMOUNT
                 CASH    10.00
            CHANGE              2.00
       C raduded In Tofel              0.40
       Thank You           ! Please Again
       Goods Sold Only Exchangable
                 Within 3 Days
```

Figure 11: Text extraction output from online OCR application

D. *Conclusion/Future Work*

As you can see, based off of my results when compared to the online OCR application, I definitely accomplished my first goal of creating an OCR application in python without using too many libraries. My application is capable of taking and processing single images and extracting the text to write to a text file. My application is also capable of performing this same functionality instead on folders of images which opens up endless possibilities for real-life use and application. As for my second goal, I would say that I mostly accomplished this goal as well. The application also has the added customization functionality of being able to extract only digits which creates endless opportunities for many other real-world applications. I also was able to use my own personal blend of pre-processing by combining python logic and popular library functions such as OpenCV. As for my third goal in the project, I would say that I mostly accomplished this goal as well. Throughout all my testing, I have thus far received very solid results on scanned images and have found that my OCR application is extremely useful and has many applications. I have personally wanted to develop some kind of OCR system for a while and was very happy to get to do so with this project. I also plan to continue my work on this OCR application in an attempt to improve it in a number of ways for the future. Some of the major functionalities and improvements that I hope to implement for this project are as follows:

- Implement another row of buttons specifically for handwritten character recognition
- Implement different pre-processing/post-processing techniques for handwritten character recognition
- Improve existing pre-processing accuracy to match that of other popular OCR systems
- Improve existing pre-processing execution time to match that of other popular OCR systems
- Improve the quality of the GUI with better styling

References

[1] NomanIslam,Zeeshan Islam,Nazia, *A Survey on Optical Character Recognition System*, Journal of Information & Communication Technology-JICT Vol. 10 Issue. 2, December 2016

[2] Satti,D.A., 2013,Offline Urdu Nastaliq, *OCR for Printed Text using Analytical Approach* MS thesisreport Quaid-i-Azam University: Islamabad,Pakistan. p. 141.

[3] Mehdi Assefi, *OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract,ABBYY FineReader, and Transym,* University of Georgia 2016

[4] Lund, W.B., Kennard, D.J., & Ringger, E.K. (2013).*Combining Multiple Thresholding Binarization Values to Improve OCR Output* presented in Document Recognition and Retrieval XXConference 2013, California, USA, 2013. USA:SPIE

[5] Narendra Sahu1andManoj Sonkusare, *A study on optical character recognition techniques*, The International Journal of Computational Science, Information Technology and Control Engineering (IJCSITCE) Vol.4, No.1, January 2017