Journal of Computer Science

Original Research Paper

An Adaptive Thresholding Algorithm-Based Optical Character Recognition System for Information Extraction in Complex Images

Daniel Akinbade, *Adewale Opeoluwa Ogunde, Mba Obasi Odim and Bosede Oyenike Oguntunde

Department of Computer Science, Redeemer's University, Ede, Nigeria

Article history:
Received: 06-04-20220
Revised: 19-05-2020
Accepted: 12-06-2020

Corresponding Author:
Adewale Opeoluwa Ogunde
Department of Computer Science, Redeemer's University, Ede, Nigeria
Email: ogundea@run.edu.ng

Abstract: Extracting texts from images with complex backgrounds is a major challenge today. Many existing Optical Character Recognition (OCR) systems could not handle this problem. As reported in the literature, some existing methods that can handle the problem still encounter major difficulties with extracting texts from images with sharp varying contours, touching word and skewed words from scanned documents and images with such complex backgrounds. There is, therefore, a need for new methods that could easily and efficiently extract texts from these images with complex backgrounds, which is the primary reason for this work. This study collected image data and investigated the processes involved in image processing and the techniques applied for data segmentation. It employed an adaptive thresholding algorithm to the selected images to properly segment text characters from the image's complex background. It then used Tesseract, a machine learning product, to extract the text from the image file. The images used were coloured images sourced from the internet with different formats like jpg, png, webp and different resolutions. A custom adaptive algorithm was applied to the images to unify their complex backgrounds. This algorithm leveraged on the Gaussian thresholding algorithm. The algorithm differs from the conventional Gaussian algorithm as it dynamically generated the blocksize to apply threshing to the image. This ensured that, unlike conventional image segmentation, images were processed area-wise (in pixels) as specified by the algorithm at each instance. The system was implemented using Python 3.6 programming language. Experimentation involved fifty different images with complex backgrounds. The results showed that the system was able to extract English character-based texts from images with complex backgrounds with 69.7% word-level accuracy and 81.9% character-level accuracy. The proposed method in this study proved to be more efficient as it outperformed the existing methods in terms of the character level percentage accuracy.

Keywords: Adaptive Threshold Algorithm, Complex Backgrounds, Images, Optical Character Recognition, Pattern Recognition

Introduction

The dynamics of today's technological domain, that has seen images play an important role in communication, calls for continuous improvements on the processing of such images, as images do not just convey the structure of places or faces, but now carries meaning but in interpretation and in the fact that more often than not, text is printed on them. According to Ranjan *et al*. (2015), text extraction in this context is a difficult task due to the presence of a complex background that poses challenges such as sharply varying contours and background pixels that have the same intensities as text pixels. The results of some systems recently developed (Rajan and Raj, 2017) showed better precision and recall compared to baseline enhancement algorithms but could not extract text from touching word, scanned documents and images with

Science Publications

© 2020 Daniel Akinbade, Adewale Opeoluwa Ogunde, Mba Obasi Odim and Bosede Oyenike Oguntunde. This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

such complex backgrounds. This study, therefore, moves from the conventional application of OCR to scanned image files or printed digital files like PDFs to the more general and more complex application to conventional photographs and other digital documents with complex backgrounds. This study is important to the fields of image processing, text-to-speech synthesis systems, screen readers, etc., as it will provide the means for such systems to increase accuracy in performance. Optical Character Recognition (OCR) is a piece of software that converts printed text and images into a digitized form such that it can be manipulated by a machine (Islam *et al*., 2016), Unlike the human brain, which has the capability to very easily recognize the text or characters from an image, machines may not have intelligence enough to perceive the information available in an image. A large number of research efforts have been put forward that attempts to transform a document image to format understandable for the machine but many are still challenged. This research is focused on applying such machine learning algorithms to images to be able to extract text from them in a more efficient manner by applying a custom adaptive algorithm to the images to unify their complex backgrounds. The algorithm used leveraged on the Gaussian thresholding algorithm and its different from the conventional Gaussian algorithm as it dynamically generated the blocksize to apply threshing to the image. This ensured that, unlike conventional image segmentation, images were processed area-wise (in pixels) as specified by the algorithm at each instance, which is a major contribution of this work.

The remainder of the paper is organized as follows. Section two reviewed some related works in the literature. Section three described the images and the algorithm used to solve the problem. Section four reports the implementation details and the results obtained from the experiments conducted and finally gave the results from comparison with existing methods. Section five concludes the paper and gave some future directions.

Literature Review

Pattern Recognition Systems

Pattern recognition is the automatic detection of patterns and regularities in data. It is closely related to machine learning and artificial intelligence, applications such as data mining and knowledge discovery in databases. Machine learning is one approach to pattern recognition, while other methods include hand-crafted rules or heuristics; and pattern recognition is one technique to artificial intelligence, while additional methods include symbolic artificial intelligence. “The field of pattern recognition is concerned with the automatic discovery of regularities in data by using computer algorithms and using these regularities to take actions such as classifying data into different categories” (Bishop, 2006). Character Recognition is simply a machine simulation of human reading, also known as optical character recognition (Das *et al*., 2012). The character recognition system is used for recognizing the characters in any document containing handwritten and machine printed text, graphics, videos, etc. and converting them into digitized format in machine-readable or ASCII Codes.

Recognition Engines

Several recognition engines exist to address different needs and proffer solutions in various fields. Some of these engines and application areas they cater for are x-rayed here. Optical Character Recognition (OCR) engines turn machine-printed images into machine-readable characters. These engines play vital roles in screen reading systems, text-to-speech synthesis systems, etc. Intelligent Character Recognition (ICR) reads images of hand-printed characters (not cursive) and changes them into machine-readable characters. Hand-printed character images are taken from a bitmap of the scanned image. ICR recognizes numeric characters more accurately than letter characters. Optical Mark Recognition (OMR) detects the existence of a mark, not its shape. OMR forms usually contain small ovals, referred to as 'bubbles,' or checkboxes that the respondent fills in. OMR cannot recognize alphabetic or numeric characters. It is commonly used in standardized examinations (serving as a marker for test sheets). OMR is the fastest and most accurate technology for data collection, it is also relatively user-friendly. OMR’s accuracy is the result of accurate measurement of a mark’s darkness and sophisticated mark discrimination algorithms to determine whether it is erasure or a mark that is detected. Magnetic Ink Character Recognition (MICR), is a specialized character recognition technology, adopted by the U.S. banking industry to facilitate the processing of cheques. Barcode Recognition is a data representation that can be read by a machine. Barcodes can be read or scanned from an image using the software with optical scanners called barcode readers, it is used in sales systems, authentication systems and card recognition.

Optical Character Recognition Systems (OCR)

The OCR systems can be categorized as handwritten recognition and printed character recognition, based on the type of input. The latter is a relatively more straightforward problem because characters are usually of uniform dimensions and the positions of characters on the page can be predicted (Bhansali and Kumar, 2013). Handwritten character recognition is a very tough job due to the different writing styles of the user as well as various pen movements by the user for the same

character. These systems can be divided into two sub-categories, i.e., on-line and off-line systems. The former is performed in real-time while the users are writing the character. They are less complicated as they can capture the temporal or time-based information, i.e., speed, velocity, number of strokes made, the direction of the writing of strokes, etc. Also, there is no need for thinning techniques as the trace of the pen is a few pixels wide. The offline recognition systems operate on static data, i.e., the input is a bitmap. Hence, it is challenging to perform recognition.

Existing OCR Systems have been used to convert the text in scanned paper documents into ASCII symbols and other encodings. However, current OCR systems do not work well if the text is printed against shaded or hatched backgrounds, as is often found in photographs, maps, monetary documents, engineering drawings and commercial advertisements. Furthermore, these documents are usually scanned in greyscale or color to preserve details of the graphics and pictures which often exist along with the text. For current OCR systems, these scanned images need to be binarized before actual character segmentation and recognition can be done. A typical OCR system does the binarization to separate text from the background by global thresholding. Unfortunately, global thresholding does not perform well on complex images, as noted in the literature (Fletcher and Kasturi, 1988).

Optical Character Recognition is a subset of pattern recognition. OCR borrows various concepts and techniques from pattern recognition and image processing. However, character recognition provided the impetus for making pattern recognition and image analysis as matured fields of science and engineering (Chaudhuri *et al*., 2017). Designing machines that can imitate human attributes has been a significant concern for man. One such imitation of human functions is reading of documents containing different forms of text. Machine reading has grown from a dream to reality over the last few decades, through the development of advanced and effective OCR systems. This technology can convert scanned paper documents, pdf files, or images captured by a digital camera into machine-editable and searchable data. A typical OCR system consists of numerous components, such as input text, optical scanning, location segmentation, pre-processing, segmentation, representation, feature extraction, training and recognition, post-processing and output text.

Thresholding Algorithms

Thresholding deals with converting multilevel images into a bi-level black and white image. This process is essential as the results of recognition are dependent on the quality of the bi-level image. Image thresholding segments a digital image based on a particular characteristic of the pixels (for example, intensity value). The goal is to create a binary representation of the image, classifying each pixel into one of two categories, such as “dark” or “light.” This is a common task in many image processing applications and some computer graphics applications. The two essential categories of thresholding are global and local.

Some of the tools used to achieve the objectives of the work are Tesseract and Pytesseract. Tesseract is an optical character recognition engine for various operating systems (Kay, 2007). It is free software, released under the Apache License, Version 2.0. Python-tesseract (Pytessetact) is an optical character recognition (OCR) tool for python, it can recognize and read–texts embedded in images. It is a wrapper for Google’s Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff and others, whereas tesseract-OCR by default only supports tiff and bmp. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file (Lee, 2018).

Deb *et al*. (2002), suggested a nondominated sorting-based Multiobjective EA (MOEA), called Nondominated Sorting Genetic Algorithm II (NSGA-II), which alleviates most of the difficulties. Specifically, the work presented a fast nondominated sorting approach and a selection operator that creates a mating pool by combining the parent and offspring populations and selecting the best (for fitness and spread) solutions. Nakib *et al*. (2010), reported image thresholding based on Pareto multiobjective optimization which adopted the evolutionary algorithm NSGA-II presented by (Deb *et al*., 2002). This method optimizes several segmentation criteria simultaneously in order to improve the quality of the segmentation. Horng and Jiang (2010), on the other hand, proposed a multilevel image thresholding selection model based on the Firefly Algorithm. Four different methods were implemented and compared to the proposed method: the exhaustive search, the particle swarm optimization, the hybrid cooperative-comprehensive learning-based PSO algorithm and the honey bee mating optimization. The experimental results revealed that the algorithm could search for multiple thresholds, which are very close to the optimal ones examined by the exhaustive search method.

A novel texture-based color image enhancement methodology that focuses on an automatic target image generation is proposed in (Raji *et al*., 2015). The images in the database with highest histogram correlation with input image are identified for extracting different features. Target image is obtained by fusing images selected based on minimum Euclidean distance between extracted features. The proposed method is a simple color image enhancement methodology where the range

(the gamut) of the R, G and B channels is optimally preserved. A new quantitative validation approach is derived to identify visibility loss problem that may occur during enhancement. The maximum possible contrast enhancement is achieved by stretching the intervals of the color levels to the maximum possible extent using a sigmoid function. The proposed method has been proved to be a successful approach to deal with various categories of images.

Rajinikanth and Couceiro (2015), improved on the Firefly algorithm in their work, "RGB Histogram-Based Color Image Segmentation Using Firefly Algorithm". This method considered the RGB histogram of the image for bi-level and multi-level segmentation, optimal thresholds are achieved by maximizing Otsu's between class variance function for each color components. Since the conventional multilevel thresholding approaches exhaustively search the optimal thresholds to optimize objective functions, they are computationally expensive. Liu *et al.* (2015), suggested the Modified Particle Swarm Optimization (MPSO) algorithm to overcome this drawback. The MPSO employs two new strategies to improve the performance of original Particle Swarm Optimization (PSO), which are named Adaptive Inertia (AI) and Adaptive Population (AP), With the help of AI strategy, inertia weight varies with the searching state, which helps MPSO to increase search efficiency and convergence speed.

Moreover, with the help of AP strategy, the population size of MPSO also varies with the searching state, which mainly helps the algorithm to jump out of local optima. More recently, Satapathy *et al.* (2018) proposed an improved bi-level and multi-level threshold procedures based on their histogram using Otsu's between-class variance and a novel Chaotic Bat Algorithm (CBA). Maximization of between-class variance function in Otsu technique is used as the objective function to obtain the optimal thresholds for the considered grayscale images.

This work however, proposes a new adaptive thresholding algorithm that maximizes the blocksize variable of the Gaussian local thresholding algorithm which has been reported as efficient. Unlike the default algorithm where the same blocksize is chosen for the entire local region under consideration, this work uses a blocksize that is relative to the local region and it is selected automatically.

Other Related Works

The subject of text detection and extraction has been growing concerns in the research community. Several research efforts have been made in recent times to improve upon the accuracy and quality of text extraction from images of various forms. A review of some the available and very related research in this area is presented in this section.

A review of detection and extraction of texts from a complex background was carried out in Kavyashere and Rejesh (2018). Several research on the subject were categorised and analysed. They provided a summary work on text detection and extraction; listed the contribution and limitations of various study in this area and made recommendations for future direction. The authors reported that existing research efforts has not been able to address the difficulties inherent in detecting and extracting texts from skewed images. Ding *et al.* (2018), proposed an improved OCR video text recognition technology. The extraction of text in video was done using the edge analysis algorithm and SVM was employed in portioning the pixels in text and non-text pixels. The findings from the results recorded a better recognition accuracy and a higher text location. The recall ratio was 92.8%, with a false alarm of 12.1%. However, there was no report on the word extraction accuracy of the approach and it could not detect vertical text.

Kumuda and Basavaraj (2017), a method for text extraction and analysis from complex image from scene was proposed based on edge segmentation. Discrete Wavelet Transform (DWT) was used to detect edges at the early stage, while the localisation of text region was done based on clustering and AdaBoost classifier. Character extraction was done at the next stage, using morphological operations and heuristic rules. The results of the study on various images from database were impressive with 91% precision and 85% recall rate and 5-6 sec time computation. Nevertheless, the approach could not detect text from structures like window frame. Again, the approach did not report the accuracy of word extraction.

An image processing approach for Segmentation and extraction of text from curved text lines from document images was presented in Shejwal and Bharkad (2017). The curved text segmentation was conducted based on x-line and base line. The proposed technique could detect the words in the document image, which was specified by bounding boxes plotted around the words. Words' segmentation was achieved using properties of connected components. Are used for segmentation of words. This algorithm recorded 77.24% accuracy of proposed characters extraction from curved text lines. The approach was effective in detecting text from handwritten and text like background of same colour.

A hybrid method for extraction of text from natural science image with chaotic background was presented in Satwashil and Pawar (2017). Four stages were involved in the extraction. Character descriptors were used in the first stage to extract superimposed text regions in an image. Character descriptors and SVM classifier were used for text content or non-text content in the second phase. Detection of multiple lines in localized text regions and line segmentation were carried out using

horizontal profiles, in the third step. Finally, vertical profiles of each character were used to extract each character of the segmented line. Images for the test of this study were drawn from ICDAR 2013 and SVT 2010 datasets. The results of the analysis of the classifiers showed 64.40% accuracy of the Ostu, 75.04% of the AdaBoost and 78.80% of the SVM. However, the approach could not detect characters of multilinguistic scripts.

Rajan and Raj (2017) employed fractional Poisson model for mining text character from natural scene images to increases the quality of the images obtained by Laplacian operation. Characters were detected using the Maximally Stable Extremal Region algorithm. The result showed a better precision and recall compared to baseline enhancement algorithms. Regardless, the algorithm could not extract text from touching word and scanned documents. A Gaussian Mixture Model Algorithm and Expectation Maximization Algorithm were employed in Rajesh and Aradhya (2015) for detection of skews signature. The result showed 0.3% average error detection of 300 input samples. However, the study was limited to skew detection without any implementation. In Rajesh and Aradhya (2016), Independent Component Analysis (ICA) and Neural Networks were deployed for identification of Kannada Signature. The dataset was composed of 100 individual Kannada signatures with 50 samples each. The result showed that the larger the percentage of the training set the better the recognition ability.

A deep learning scheme for mining text with a complex background was presented in Nguyen *et al.* (2019) based on the Connectionist Text Proposal Network (CTPN) method. The work was implemented and tested using several books covers of over 6000 images which showed some improvement of feature extraction. However, the scheme needs some improvement in the area of auto-crop image, detecting text line with arbitrary direction, low contrast input image. In addition, there was no record of the word extraction accuracy of the method.

An in-depth presentation of Tesseract OCR engine was carried out in Kaundilya *et al*. (2019). Tesseract was developed by HP Labs and now owned by Google. It was described as the most accurate optical character recognition engine. Texts were extracted from images using text localization, segmentation and binarization. Text extraction was used in creating e-books from scanned books, image searching from a collection of visual data, among others. The result of the analysis showed that Tesseract is efficient OCR system. However, the accuracy of the OCR systems is highly dependable on the quality and nature of the text data. Liu *et al*. (2018) presented a new scheme for detection of the dust image text based on convolutional neural network and Gaussian smoothing. The results obtained revealed that the scheme could be used to detect text regions in dust images with good performance.

Methodology

This section provides a description of the methods employed and model applied to achieve the objectives of this research work.

Data Collection and Description

The data for this work were sourced from web repositories. The web was surfed to gather images with backgrounds that align with the interest of this study, the test experiment was carried out using the collected data and the algorithm was constantly tweaked as adjustments were needed to improve the system. To achieve the aim of the work, the image segmentation method using adaptive thresholding was employed, although this worked remarkably well, yet improvements made to the algorithm by automating the task of selecting block size improved the result. A few simple and basic image pre-processing techniques like sampling, filtering and feature extraction were applied for the sake of easy and smooth running of the experiments.

The Proposed System's Architecture

The new system will provide a means of extracting texts from images with complex (shaded or hatched) background. The method employed an in-built python function to convert the image to greyscale, then implemented a custom adaptive thresholding algorithm, which performed the task of image segmentation using a block size that helped separate essential features of the image based on a minimal number of pixels. Hence, this threshold made the area being segmented to be relative to surrounding (immediate) pixels and not the entire image file. Finally, the threshed image was parsed to tesseract, a commercial OCR tool, to separate the text easily. Tesseract has been proven to be very efficient in extraction of texts (Kaundilya *et al*., 2019). The proposed architecture, shown in Fig. 1, has various components and each of these is defined with a specific purpose and linked to be able to extract texts, i.e., pre-process an image irrespective of the nature of the background and return a text as its output.

The Pre-Process Module

This is where all the processes necessary to adapt images, i.e., segmenting interesting features and preparing the image for text extraction. It comprises three sub-modules to help it perform its task. It houses the adaptive thresholding algorithm shown in Algorithm 1.

The Image to Grey Submodule

This sub-module helps convert the input image to grayscale, implementing python and openCV functions to perform this task.

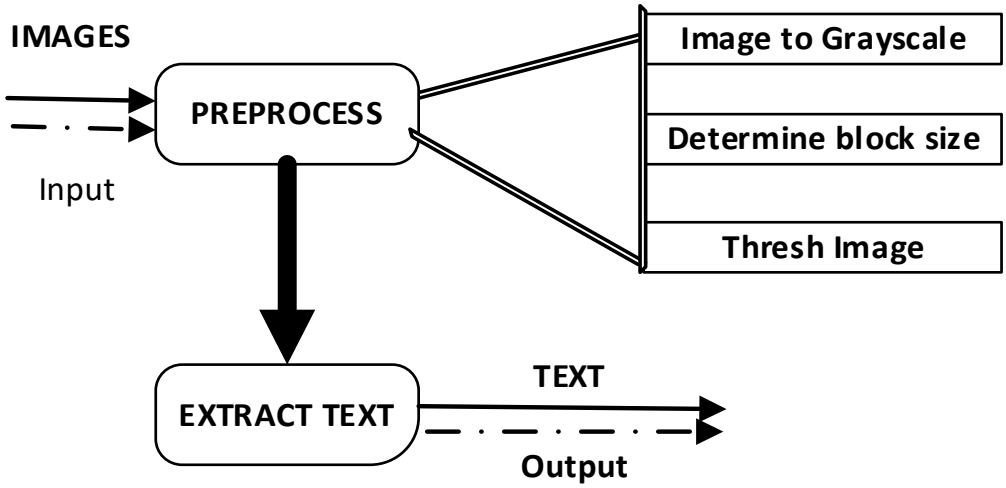

Fig. 1: System's architecture

The Set Block Size Submodule

The set block size sub module helps to determine the pixel to use per area of the image. The adaptive thresholding algorithm was employed to achieve this. It also automatically removes the noises and the artefacts in the image as noise can drastically reduce the overall quality of the OCR process. Noise and artefacts could result from poor quality of original image or poor scanning of image.

The Thresh IMAGE Submodule

This sub-module returns the image that has been segmented based on previous processes. This threshed image is parsed to the Extract Text module for character extraction.

The Extract Text Module

The extract module is where the text on the pre-processed image is extracted and displayed on the screen of the system. The extracted text can also be saved from here.

Custom Adaptive Algorithm

The custom adaptive algorithm used for this work is represented in Algorithm 1. This algorithm leverages on the Gaussian thresholding algorithm, which is callable in python. The algorithm differs from the conventional Gaussian in that it uses a dynamically generated blocksize to apply threshing to the image. The algorithm has a variable (src) that holds an image as an array and another variable (grayImage) that outputs an image as an array. Line 3 shows a procedure/function that converts src to a grayscale image.

Algorithm 1: Adaptive Thresholding

```
1:    InputArray image: src
2:    OutputArray image: grayImage
3:    Procedure ConvertToGrayScale(src)
4:      grayImage = [ ]
5:      row,col,CHANNEL = src.shape
6:        for i in range (row):
7:            for j in range (col):
8:                  a = (src[i,j,0]*0.07 + src[i,j,1]*0.72 +
                    src[i,j,2] *0.21)
9:                  grayImage.append(a)
10:           end for
11:     return grayImage
12:   end procedure
13:     InputArray: Grayimage
14:     OutputArray: Threshedimage
15:     procedure AdaptiveThresholding(grayImage)
16:       src = grayImage
17:       blocksize = 1
18:       constant = 12
19:       maxValue = 255
20:       adaptiveType =
          cv2.ADAPTIVE_THRESH_GAUSSIAN_C
21:       thresholdType = cv2.THRESH_BINARY
22:       threshImage = cv2.adaptiveThreshold(src,
          maxValue, adaptiveType, thresholdType,
          blocksize, constant)
23:       if src(i, j) > (i, j) then
24:         threshedImage (i, j) = maxValue
25:       else
26:           threshedImage (i, j) = 0
27:        blocksize = blocksize + 2 Goto Line 22
28:       end if
29:    return threshedImage
30:   end procedure
```

Line 4 shows the grayImage being initialized as an empty string and in line 5, the input image is given a shape attribute, this enables the image to be properly signified in dimensions. Hence, the image array is formed and can be iterated through. Lines 6 and 7 show how the image is being looped through; this is possible as a result of shaping the image. Line 8 shows a variable that holds the value used for converting the input image (SCR) to grayscale, where, the first value represents Red, the second, Green and the third, Blue. In line 9, the new variable is appended to the empty array grayImage. Hence, grayImage becomes the grayscale version of src and this value is returned and the procedure ends. Line 15 shows another procedure being defined. This is the adaptive thresholding algorithm; it takes as input the output from the previous algorithm (grayImage) and returns threshedImage which will hold a threshed Image. Lines 16 to 19 initializes some variables, src which is set to the input image, blocksize with an initial value of 1, this variable holds odd values otherwise the final system will not execute, constant initialized to 10, this holds the value that specifies the area that should be threshed per iteration and maxValue set to 255 represents the highest pixel value the threshed image can be and 0 representing the minimum pixel value. Lines 20 to 22 implements the Gaussian algorithm. Lines 23 to 28 implements a process to check if the resulting image from thresh is optimal if it is not the block size is increased by 2 and the image is threshed again else the threshedImage is returned as the output of this procedure.

Systems Activity Diagram

The system's activity diagram, represented in Fig. 2, shows the activities embedded in the system; it represents each process and decision making of the system. It shows that the system initializes and expects an image as input, if the image is loaded correctly then the system converts the loaded image to grayscale, else, it terminates the current process, a dynamic thresholding blocksize is then set on the grayscale image, which is then threshed according to the dynamic block size to make character segmentation easier, the text characters from the resulting image are then separated and displayed as output of the entire process.

System Sequence Diagram

The system's sequence diagram, shown in Fig. 3, displays the system operating order, the system's primary function provides a GUI and allows the user to load the image using a file dialog, the input image is parsed to a function which converts it to grayscale image, this is then parsed to the function which dynamically sets the block size, the result of this is parsed to the thresh function which uses the Gaussian threshing to segment the image, then, based on this threshing, the extract text module extracts the interesting characters from the image and returns these texts as response on the GUI.

System Class Diagram

The system was implemented using Object-Oriented Programming. Hence the class diagram comprises 2 classes as shown in Fig. 4. The classes are categorised here.

Ui_MainWindow.py

This class creates the GUI. It inherits the QMainWindow class from PyQT. Gui package. It takes an input image and also displays the output extracted text

Threshh.py

This class receives the input image from the Ui_MainWindow class and performs several operations depending on the button clicked on the GUI. It essentially converts the image to greyscale, applies the adaptive thresholding algorithm and extracts the text by employing tesseract OCR using pytesseract module of python; the extracted text is sent back to the Ui_MainWindow. This class can also display the globally threshed image and the image that has adaptive thresholding applied to it.

The Conceptual Model of the OCR System

The conceptual model of the OCR system, as shown in Fig. 5, shows that the system takes in the image as input, carries out image conversion to grayscale, determine block size to apply, uses the custom thresh to the binarized image, extract text from pre-processed image and finally returns texts as the output of the whole process.

The Essential Processes in the Model Are

Convert Image to Grayscale

This is the process of converting the input image to grayscale. That is all forms of color, apart from white and black in different shades and intensity is removed from the original image. OpenCV has an inbuilt function to help with this.

Determine Blocksize

This entails setting a value or values for different areas of the image. Our adaptive algorithm employs different blocksize to a different part of the image based on the properties of the interested region. This employs Algorithm 1.

Thresh Image, which applies the image threshing algorithms to segment text from background. This involves the custom algorithm together with openCV's mean_c threshing algorithm.

Extract Text, this extracts the text/character part of the result of the Thresh image process and gets the characters ready to be saved in a text file. This process employs the OCR tool.

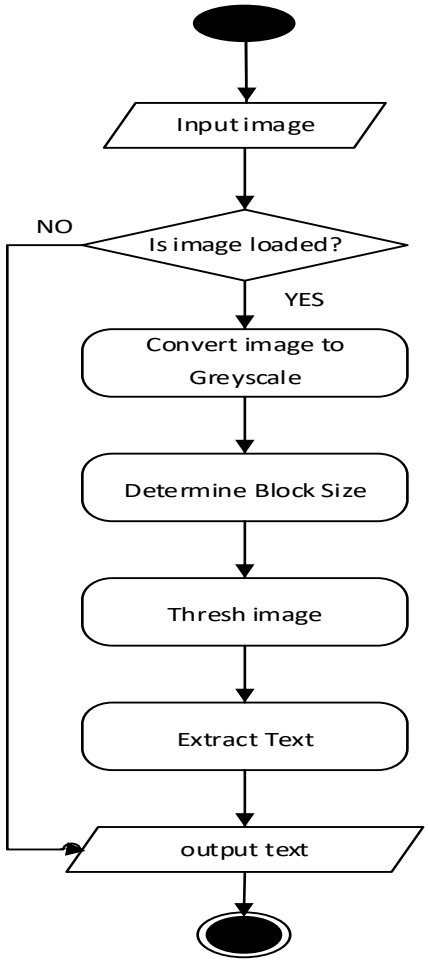

Fig. 2: System's activity diagram

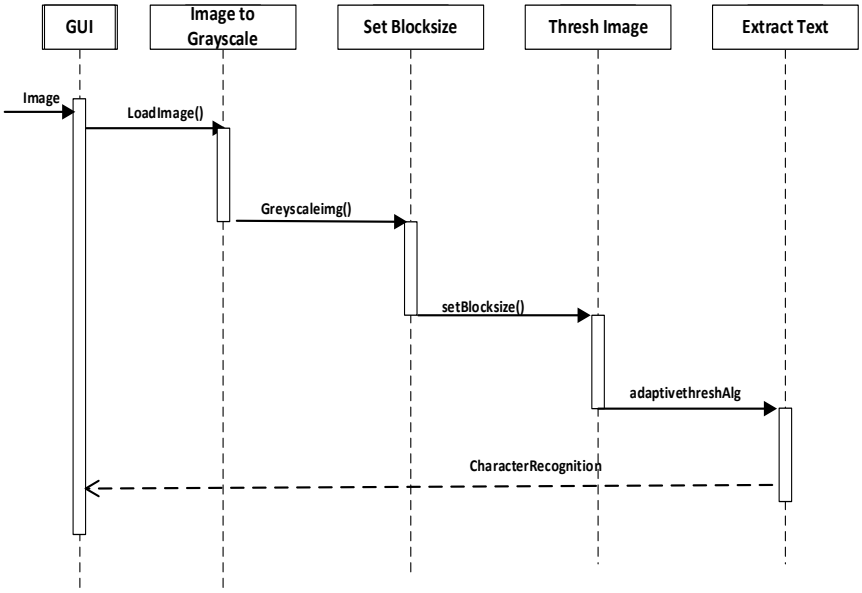

Fig. 3: System's sequence diagram

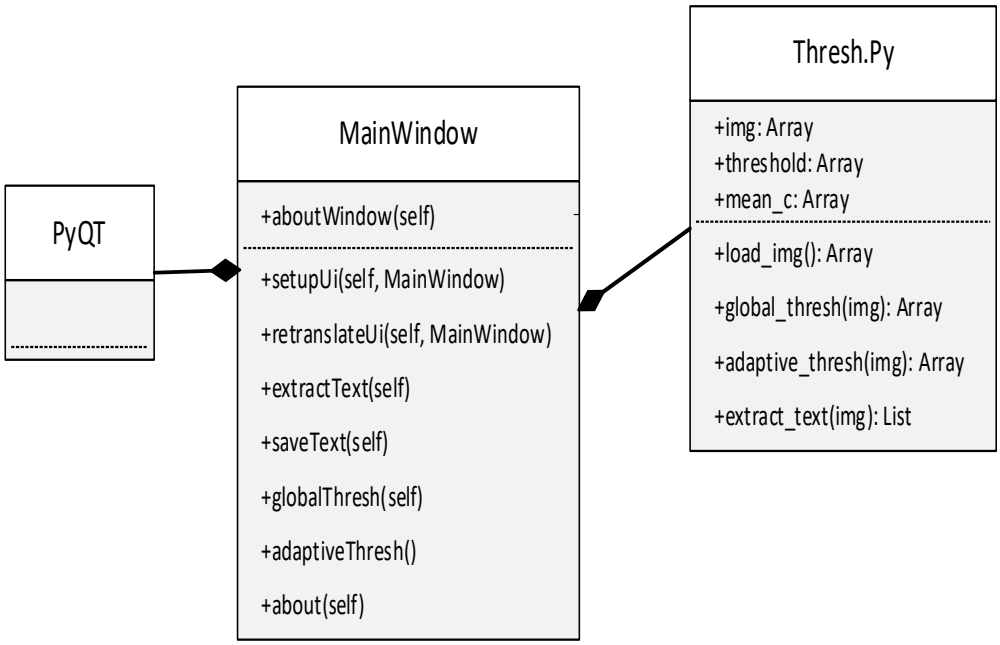

Fig. 4: System's class diagram

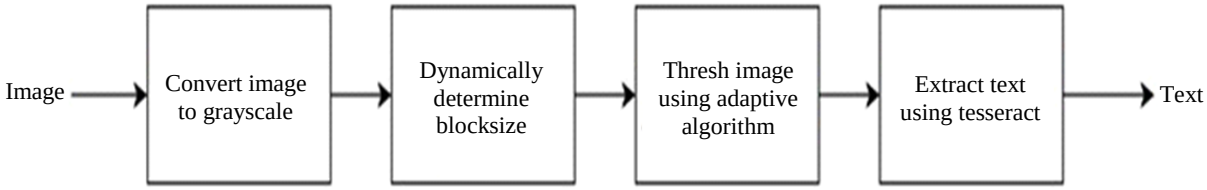

Fig. 5: Conceptual model of the OCR system

Implementation and Results

In this section, a detailed description of the system classes and the tools used in its implementation were outlined.

Software Used for the Implementation

The system's implementation was carried out using the Python programming language (version 3.6.4) with PyQt (version 5). Also, the 'Pillow' library was used for image analysis as well as the Tkinter module for the file dialog. PyQ, which is a third-party package for building Graphic User Interface (GUI), was employed to design the user interface of the system. Python 3.6.4 was used for the software development because of the advantages it has; its text processing capabilities, large number of extensive libraries available, high dynamic data types and the provision of third-party modules (e.g., Qt) for developmental tasks among many other benefits are a reason for using it. PyQt5 is a third-party tool used for GUI design. It is a component of the Digia's Qt cross-platform application development framework. Python IDLE (Integrated Development and Learning Environment) is an integrated development environment for Python, which has been bundled with the default implementation of the language. It was written in Python and Tkinter GUI toolkit. IDLE is a simple IDE, which is cross-platformed and avoids feature clutter. IDLE's main features are Multi-window text editing with syntax highlighting, auto-completion and smart indent, Python shell with syntax highlighting, integrated debugger with stepping, persistent breakpoints and call stack visibility.

Implementing the Thresh Class

This class implements the core processes of the system. It has four functions, as explained below.

Load_img()

This method makes use of the file dialog functionality of the Tkinter module to enable the system to navigate the OS of the host machine and uses the openCV module to select whatever image file that is intended to be analysed (image with text to be extracted).

Global_Thresh(img)

This method makes use of the 'cv2.THRESH_BINARY' openCV default global thresholding algorithm. The values are set at 110 and 255. It returns a threshed image (array).

Adaptive_Thresh(img)

This method makes use of the custom algorithm to select the block size and uses that to implement the Gaussian mean_c openCV thresholding algorithm. It returns a threshed image (array)

Extract_Text(img)

This method takes as input an image retrieved from the adaptive_thresh method and engages the tesseract OCR engine by using pytesseract to separate the textual data from the rest of the image.

Implementing the Ui_MainWindow Class

This class brings about the functionality of widgets from PyQt5, making it possible to create a GUI for the software. It inherits the MainWindow class from the PyQt5 module and it has the following methods.

About (self)

This method sets the text in the text area to the custom text that displays information about the system.

Setup Ui (Self, Main Window)

This method creates and displays the GUI for the software. It invokes the Qt widget creator.

Retranslate Ui (Self, Main Window)

This is a Qt method that translates the GUI, converting all characters to a uniform encoding.

On_PushButton_Clicked (self)

This method calls the load_img() method from the th class of the thresh module. Then, it initiates the call to load an image.

Extract Text(self)

This method responds to the click action to extract text from the threshed image by the push button labeled 'EXTRACT TEXT.' It calls the extract_text() method from the th class of the thresh module.

Save Text(self)

This responds to the click action on the push button labeled 'SAVE TEXT and it calls the save_text() method from the th class of the thresh module.

Global Thresh(self)

This method responds to the call of the push button labeled 'GLOBAL THRESH IMG,' and it calls the global_thresh() method from the th class of the thresh module. It displays a globally threshed image.

Adaptive Thresh(self)

This method responds to the call of the push button labeled 'ADAPTIVE THRESH IMG,' and it calls the adaptive_thresh() method from the th class of the thresh module. This function displays an image that has been processed using custom adaptive thresholding.

System Evaluation

The system's performance was checked at word error level and character error level. The Word Error Rate (WER) is used to compute the error rate at the word level and Character Error Rate (CER) was used at the character level. The formula is shown in equations 1 and 2 respectively:

$$WER = \frac{S + D + 1}{N} * 100 \quad (1)$$

$$CER = \frac{S + D + 1}{N} * 100 \quad (2)$$

where, *S*, *D* and *I* are the number of substitutions, Deletions and Insertions made in the transliterated word and *N* is the total number of words (the same applies to *CER*, but *N* is the number of characters) in the input English word.

Results and Discussion

The system takes in an image (with complex background) as input and returns the text written on the input image as output. As part of the product, a user can view a sample global threshed image and adaptive threshed image of the input image. The user can also choose to save the extracted text; this gets saved with a timestamp to the primary directory of the software. Figure 6 shows the outlook of the system's interface.

Five images (Fig. 7 to 11) were used to test the developed system. Properties of the five of them are summarized in Table 1.

The outputs of the system, when tested on Fig. 7 to 11, are displayed in Fig. 12 to 16.

Tables 2 and 3 show the result of the system's evaluation according to the formulas in Equations 1 and 2. Table 2 shows the performance evaluation of the system on word level for the five (5) test images used. Figures 7 and 8, which are similar, although, with different font sizes are the best performers with 100% accuracy each, whereas, Fig. 10 and 11 are the worst performers at word level of testing with 50.6% and 55.3% accuracy, respectively. The table also shows a Word Error Rate of 30.3% across the board, which is massively influenced by Fig. 10 and 11.

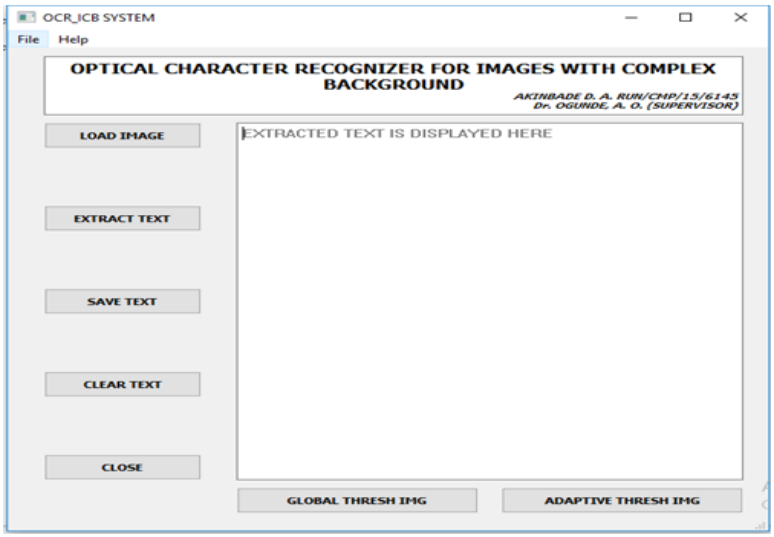

Fig. 6: OCR System for Images with Complex Backgrounds

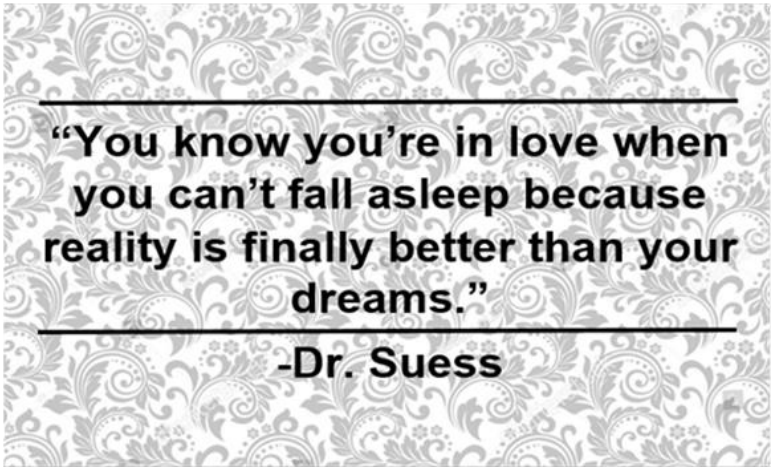

Fig. 7: Image with complex background – 1

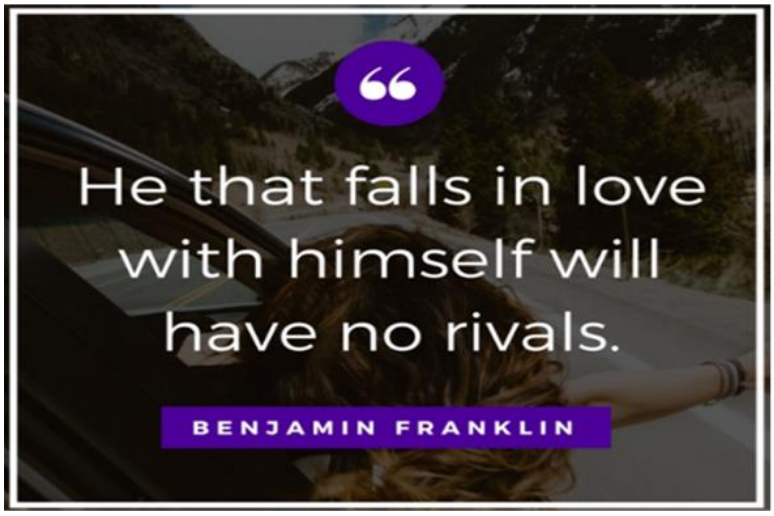

Fig. 8: Image with complex background – 2

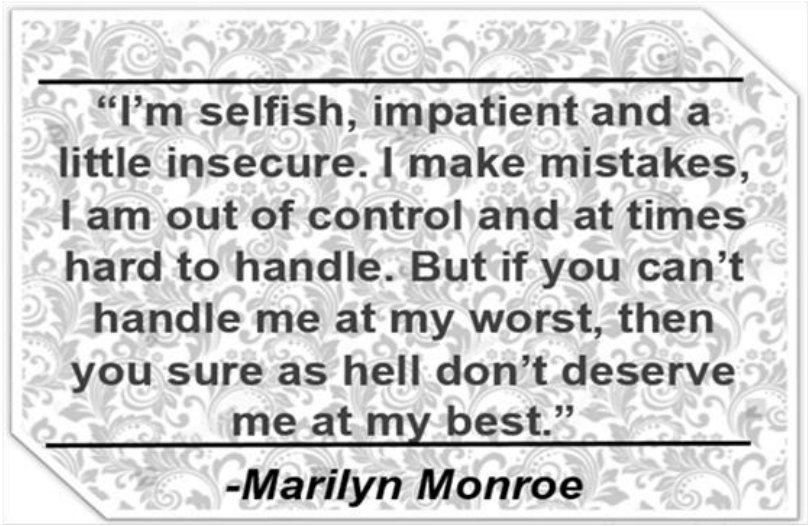

Fig. 9: Image with complex background - 3

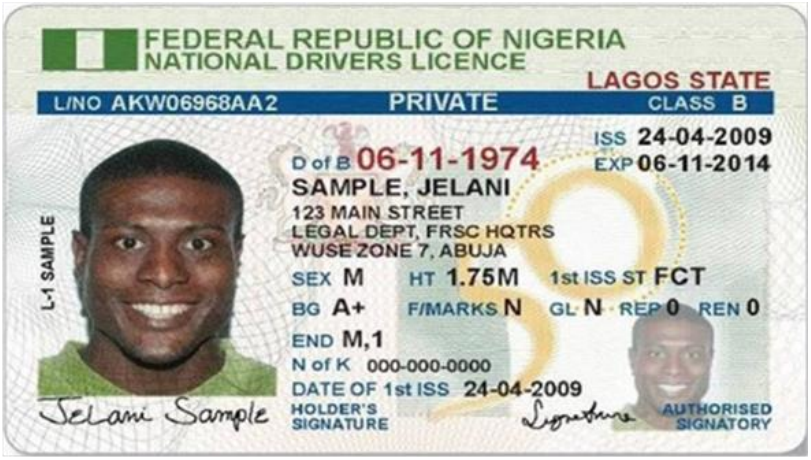

Fig. 10: Image with complex background - 4

Fig. 11: Image with complex background - 5

DOI: 10.3844/jcssp.2020.784.801

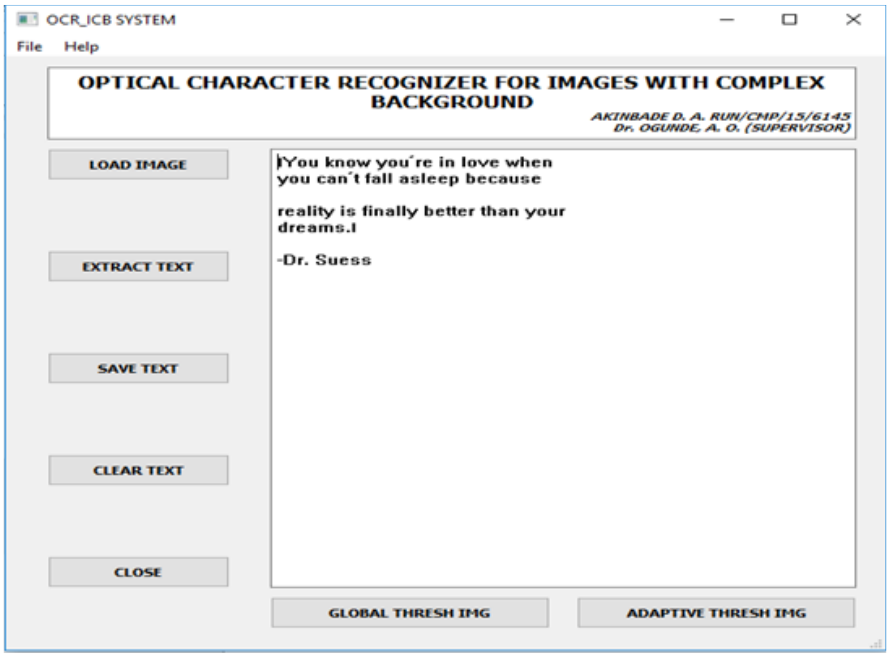

Fig. 12: System output for Fig. 7

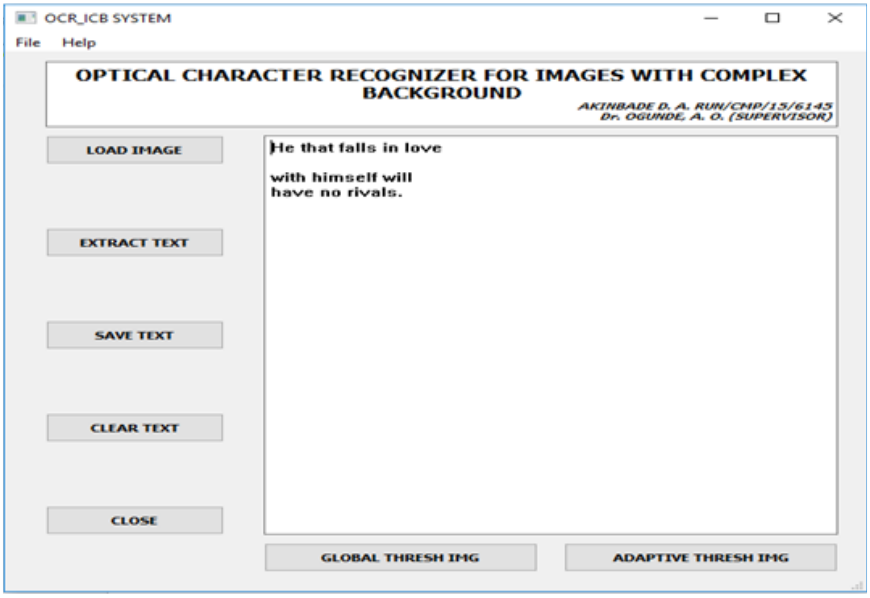

Fig. 13: System output for Fig. 8

Table 3 shows the performance evaluation of the system on a character level for five (5) test images used. Figures 7 and 8 are again the best performers with 100% accuracy each, whereas, Fig. 8 to 11 are the worst performers at character level of testing with 73.8, 75.4% and 71.1% accuracy, respectively. Table 2 also shows a Character Error Rate of 18.1% across the board, which is massively influenced by the accuracy of Fig. 7 and 9.

The result of the performance evaluation of the system showed impressive results. As shown in Table 4, the system was able to give an accuracy of over 81% at character level and about 70% at word level, however, it was observed that the system performs best on Arial font type, larger font and boldly printed texts irrespective of the complexity of the background and also when the color of the text on the image is not too varied, for example, if the system receives an

image that has two contrasting color test, it will select either of both as the region of interest and return only the tests with that color.

Validation of Results

The results obtained in this study was compared with existing works in order to validate the efficacy of the method. The metric used was the character level accuracy of the methods as most of the methods did not report on the word level accuracy their proposed methods. Results from the comparison carried out (Table 5) showed that the proposed method in this study outperformed the existing used for the comparison.

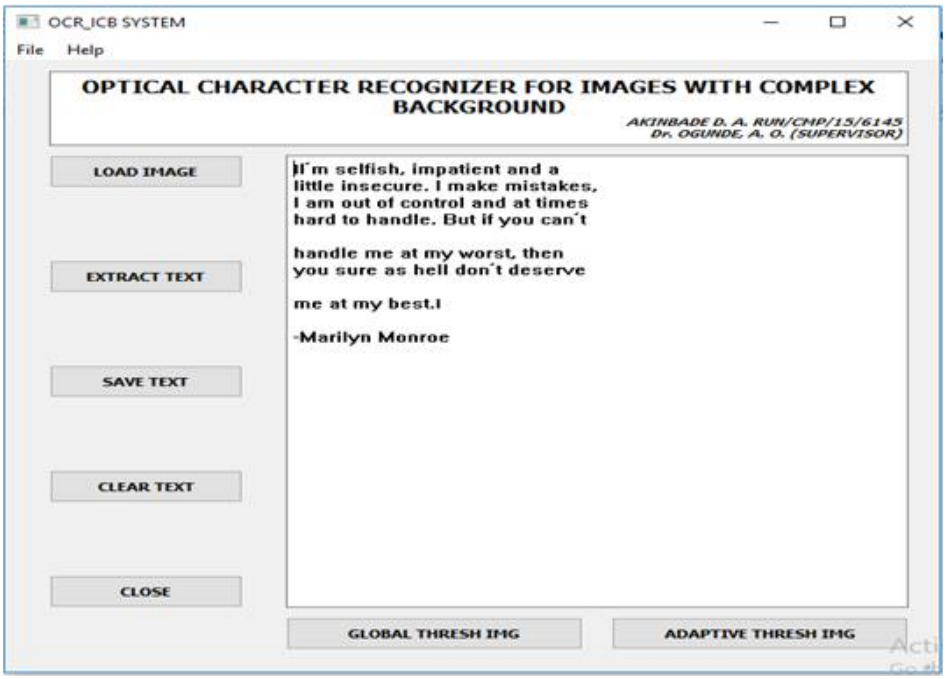

Fig. 14: System output for Fig. 9

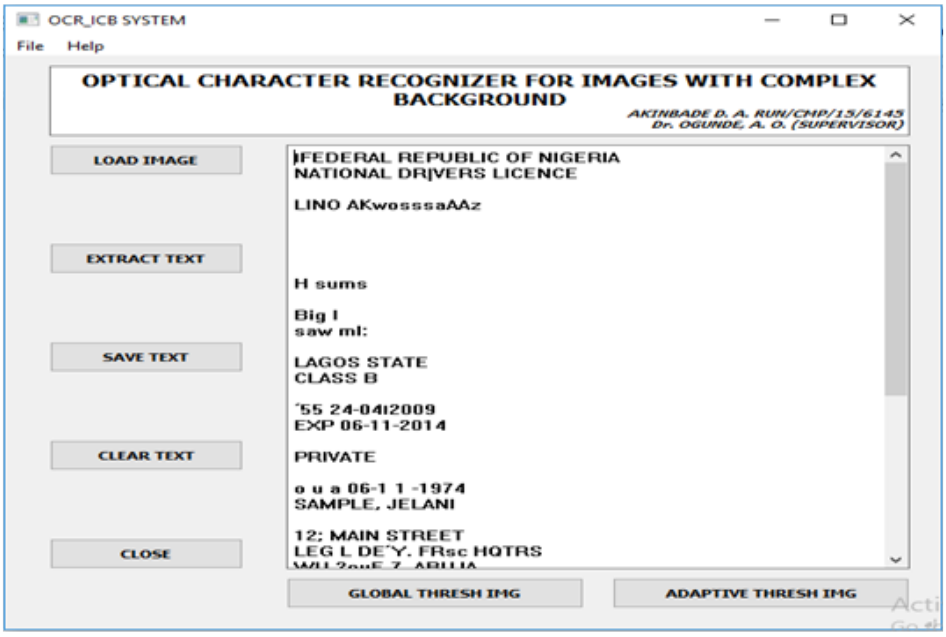

Fig. 15: System output for Fig. 10

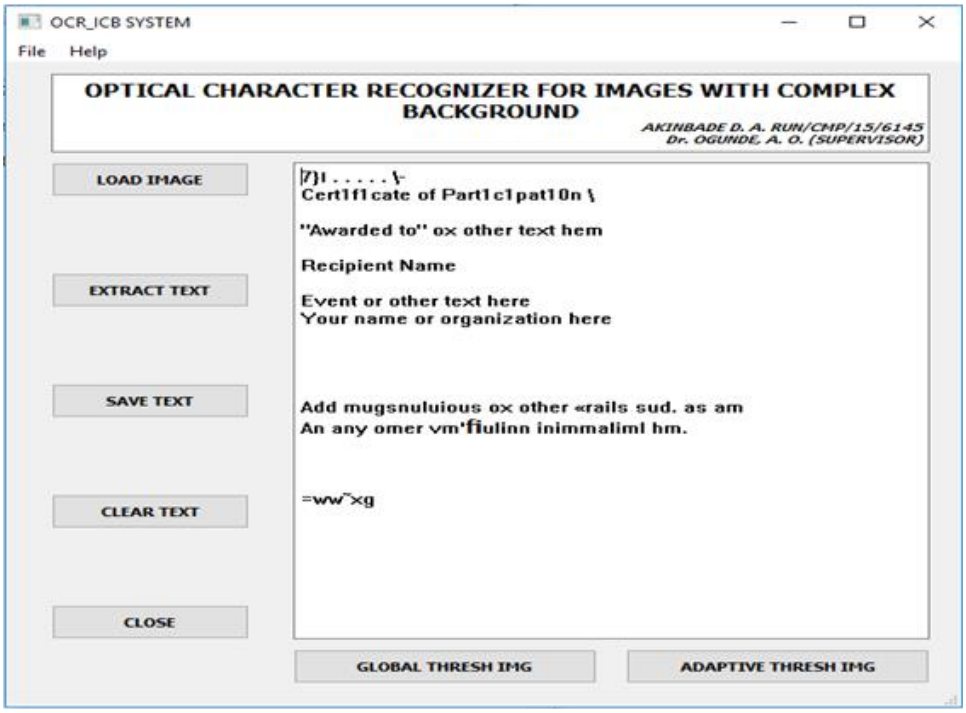

Fig. 16: System output for figure 11

Table 1: Description of experimental test images

Proper-ties	Figure 7	Figure 8	Figure 9	Figure 10	Figure 11
Size in pixels	580 x 350	589 x 388	558 x 366	590 x 333	625x 360
Resolution (ppi)	600 x 600	600 x 600	600 x 600	600 x 600	600x 600
Colour space	RGB	RGB	RGB	RGB	RGB
Precision (gamma integer)	8-bit	8-bit	8-bit	8-bit	8-bit
Size in memory (MB)	2.1	2.4	2.1	2.0	2.3
Number of pixels	203000	228532	204228	196470	225000
Number of layers	1	1	1	1	1

Table 2: System's performance evaluation (Word)

Figure	7	8	9	10	11	Total
No. of words	20	13	43	81	38	195
No. of substitutions	0	0	0	33	14	47
No. of deletions	0	2	0	2	2	6
no. of insertions	0	0	0	5	1	6
Word Error Rate (WER %)	0.0	15.0	0.0	49.4	44.7	30.3%

Table 3: System's performance evaluation (Character)

Figure	7	8	9	10	11/12	Total
No. of Characters	96	61	178	317	223	875
No. of Substitutions	0	0	0	57	36	93
No. of Deletions	0	16	0	5	11	32
No. of Insertions	0	0	0	16	17	33
Character Error Rate (CER %)	0.0	26.2	0.0	24.6	28.9	18.1%

Table 4: System performance summary

	Error rate	Accuracy
WORD	30.3	69.7
CHARACTER	18.1	81.9

Table 5: Comparison with existing methods

Method	Character level accuracy (%)
Segmentation and OCR (Shejwal and Bharkad, 2017)	77.24
Ostu (Satwashil and Pawar, 2017)	64.40
AdaBoost (Satwashil and Pawar, 2017)	75.04
SVM (Satwashil and Pawar, 2017)	78.80
GMM algorithm, (Rajesh and Aradhya, 2015)	70
Rajan and Raj (2017)	Failed to extract text
The proposed method	81.9%

Conclusion and Future Works

With the dynamic nature of today's connected world, information sharing has reached a point where there are understandably no limits to what can be shared, be it on social networks, via emails, blog posts, etc. Pictures are taking center stage in communication; hence, the intended message must be obtained when shared as texts appended to images. However, screen readers and most OCR systems only perform well when such images contain texts printed on plain backgrounds, an OCR system that can perform well on images with texts on complex background becomes a fundamental necessity in addressing this problem. This work has attempted to address the issue by designing an algorithm that leverages on tesseract, an already existing OCR system to improve on its performance on interesting images (images with complex backgrounds). The work built an OCR system using a custom adaptive thresholding algorithm and then bundling the designed algorithm with tesseract OCR using the Python programming language and the pytesseract wrapper to achieve this. Qt GUI designer was used to implement a user-friendly interface for the app and ported into python using the PyQt library. The motivation behind this work was to provide a means to better extract essential pieces of information from images, as a huge percentage of these are lost during communication and also, to offer the opportunity to easily digitize information, i.e., convert the information in image files to ASCII encoding and other machine-readable codes. In this study, an adaptive thresholding algorithm was applied, this leveraged on the positives of the Gaussian mean C algorithm, but the block size variable was dynamically determined and changes according to the image pixels per area of the entire image. The system was tested on five images using Word Error Rate (WER) and Character Error Rate (CER). The WER calculates the number of words that is incorrectly extracted from the given image, i.e., words substituted with other words, words deleted and words wrongly inserted in the output, whereas, the CER calculates the number of characters that are incorrectly extracted from the given image, i.e., characters substituted with other characters, characters deleted and characters wrongly inserted in the output. It was found that the system has WER of 30.3%, i.e., an accuracy of 69.7% and CER of 18.1%, i.e., an accuracy level of 81.9%, which appeared to be more impressive as it was able to recognize some of the test images which has mixed font types, tiny prints, skewed texts etc., which were the major difficulties of most existing methods. Validation and comparison carried out showed that the proposed method in this study outperformed the existing methods in terms of the character level percentage accuracy. Future works will review the adaptive thresholding algorithm to improve on the output of the system seeing that this system still produces up to an 18%-character-level error. Future work will consider the development of a more robust system that can extract texts from very high contrasting backgrounds. Other machine learning products asides tesseract could be bundled into the program to improve results and validate the system.

Acknowledgement

The authors acknowledge the Department of Computer Science, Redeemer's University for providing access to facilities in their software laboratory.

Author's Contributions

Daniel Akinbade: He contributed to all the sections of the paper. He conceived the idea, formulated the design and worked on the implementation. He was involved in data gathering and experimentation.

Adewale Opeoluwa Ogunde: He contributed to all the sections of the paper. He coordinated the data gathering, design and execution of all experiments and organized the paper. He participated in correcting the paper and responding to all reviewers' comments.

Mba Obasi Odim: He contributed to all the sections of the paper. He also contributed in image gathering and processing, writing, editing and formatting of the research paper. He participated in correcting the paper and responding to all reviewers' comments.

Bosede Oyenike Oguntunde: She contributed to all the sections of the paper. She also contributed in the research analysis, writing, editing and formatting the

research paper. He participated in correcting the paper and responding to all reviewers' comments.

Ethics

There are no ethical issues associated with the publication of this work.

References

Bhansali, M. and P. Kumar, 2013. An alternative method for facilitating cheque clearance using smart phones application. Int. J. Applic. Innov. Eng. Manage., 2: 211-217.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. 1st Edn., Springer, New York, ISBN-10: 0387310738.

Chaudhuri, A., K. Mandaviya, P. Badelia and S.K. Ghosh, 2017. Optical Character Recognition Systems for Different Languages with Soft Computing. 1st Edn., Springer International Publishing, ISBN-13: 9783319502519.

Das, R.L., B.K. Prasad and G. Sanyal, 2012. HMM based offline handwritten writer independent English character recognition using global and local feature extraction. Int. J. Comput. Applic., 46: 45-50. DOI: 10.5120/6948-9428

Deb, K., A. Pratap, S. Agarwal and T.A.M.T. Meyarivan, 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolut. Comput., 6: 182-197. DOI: 10.1109/4235.996017

Ding, J., G. Zhao and F. Xu, 2018. Research on video text recognition technology based on OCR. Proceedings of the 10th International Conference on Measuring Technology and Mechatronics Automation, Feb. 10-11, IEEE Xplore Press, Changsha, China, pp: 457-462. DOI: 10.1109/ICMTMA.2018.00117

Fletcher, L.A.R.K., 1988. A robust algorithm for text string separation from mixed text/graphics images. IEEE Trans. Patt. Anal. Mach. Intell., 10: 910-918. DOI: 10.1109/34.9112

Horng, M.H. and T.W. Jiang, 2010. Multilevel image thresholding selection based on the firefly algorithm. Proceedings of the Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing, Oct. 26-29, IEEE Xplore Press, Xian, Shaanxi, China, pp: 58-63. DOI: 10.1109/UIC-ATC.2010.47

Islam, N., Z. Islam and N. Noor, 2016. A survey on optical character recognition system. J. Inform. Commun. Technol., 10: 1-4.

Kaundilya, C., D. Chawla and Y. Chopra, 2019. Automated text extraction from images using OCR system. Proceedings of the 6th International Conference on Computing for Sustainable Global Development, Mar. 13-15, IEEE Xplore Press, New Delhi, India, pp: 145-150.

Kavyashere, D. and T.M. Rejesh, 2018. Analysis of text detection and extraction from complex background images. Imanager's J. Patt. Recog., 5: 37-45. DOI: 10.26634/jpr.5.3.15260

Kay, A., 2007. Tesseract: An open-source optical character recognition engine. Linux J.

Kumuda, T. and L. Basavaraj, 2017. Edge based segmentation approach to extract text from scene images. Proceedings of the 7th International Advance Computing Conference, Jan. 5-7, IEEE Xplore Press, Hyderabad, India, pp: 1-4. DOI: 10.1109/IACC.2017.0147

Lee, M., 2018. Python tesseract.

Liu, H., C. Li, S. Jia and D. Zhang, 2018. Text detection for dust image based on deep learning. Proceedings of the 33rd Youth Academic Annual Conference of Chinese Association of Automation, May 18-20, IEEE Xplore Press, Nanjing, China. DOI: 10.1109/YAC.2018.8406472

Liu, Y., C. Mu, W. Kou and J. Liu, 2015. Modified particle swarm optimization-based multilevel thresholding for image segmentation. Soft Comput., 19: 1311-1327.

Nakib, A., H. Oulhadj and P. Siarr, 2010. Image thresholding based on Pareto multiobjective optimization. Eng. Applic. Artif. Intell., 23: 313-320. DOI: 10.1016/j.engappai.2009.09.002

Nguyen, T.N., C.N.N. Hoang, T.S. Le and T.A. Tran, 2019. A system for text extraction in complex-background document images. Proceedings of the International Conference on Advanced Computing and Applications, Nov. 26-28, IEEE Xplore Press, Nha Trang, Vietnam, pp: 65-69. DOI: 10.1109/ACOMP.2019.00017

Rajan, V. and S. Raj, 2017. Text detection and character extraction in natural scene images using fractional Poisson model. Proceedings of the International Conference on Computing Methodologies and Communication, Jul. 18-19, IEEE Xplore Press, Erode, India. DOI: 10.1109/ICCMC.2017.8282651

Rajesh, T.M. and V.M. Aradhya, 2016. ICA and neural networks for Kannada signature identification. Int. J. Latest Trends Eng. Technol., 7: 271-278. DOI: 10.21172/1.73.\537

Rajesh, T.M. and V.M. Aradhya, 2015. An application of GMM in signature skew detection. I-manager's J. Pattern Recognition, 2: 8-15. DOI: 10.26634/jpr.2.3.3757

Raji, R., D. Mishra and M.S. Nair, 2015. A novel texture based automated histogram specification for color image enhancement using image fusion. Proceedings of the International Conference on Information and Communication Technologies, Dec. 3-5, IEEE Xplore Press, India, pp: 1501-1509. DOI: 10.1016/j.procs.2015.02.070

Rajinikanth, V. and M.S. Couceiro, 2015. RGB histogram based color image segmentation using firefly algorithm. Proceedings of the International Conference on Information and Communication Technologies, Dec. 3-5, IEEE Xplore Press, India, pp: 1449-1457. DOI: 10.1016/j.procs.2015.02.064

Ranjan, R., P. Venugopal, S. Prithvi and N.S. Priyanka, 2015. Text extraction from images with complex background. Int. J. Eng. Res. Technol., 3: 1-5.

Satapathy, S.C., N.S.M. Raja, V. Rajinikanth, A.S. Ashour and N. Dey, 2018. Multi-level image thresholding using Otsu and chaotic bat algorithm. Neural Comput. Applic., 29: 1-23. DOI: 10.1007/s00521-016-2645-5

Satwashil, K.S. and V. Pawar, 2017. Integrated natural scene text localization and recognition. Proceedings of the International conference of Electronics, Communication and Aerospace Technology, Apr. 20-22, IEEE Xplore Press, Coimbatore, India, pp: 371-374. DOI: 10.1109/ICECA.2017.8203708

Shejwal, M.A. and S.D. Bharkad, 2017. Segmentation and extraction of text from curved text lines using image processing approach. Proceeding of the International Conference on Information, Communication, Instrumentation and Control, Aug. 17-19, IEEE Xplore Press, Indore, India, pp: 1-5. DOI: 10.1109/ICOMICON.2017.8279138