Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Automatic Recognition of graphic patterns

Bunna CHOM

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Bunna Chom

Systems Engineering and Informatics
Informatics

Thesis title

Automatic recognition of graphic patterns

Objectives of thesis

The aim of the work is the practical use of OCR technologies (ideally in C # or Java) for reading characters and graphic templates (logos, icons, signatures)

Methodology

Follow the following methodology:

- perform a research of the current state
- design a suitable application architecture
- program the application in the selected language and using selected technologies
- test the result
- define conclusions

The proposed extent of the thesis

30-40

Keywords

ORC, Shape Recognition, Artificial Intelligence, Java, C#

Recommended information sources

Authors: Horst Bunk, Patrick S P Wang. Handbook of Character Recognition and Document image Analysis ISBN: 981-02-2270-X May 1997 852 Pages

Authors: Sergios Theodoridis, Konstantinos Koutroumbas, Pattern Recognition 4th Edition ISBN: 9781597492720 October 984 Pages

Authors: Shunji Mori, Hirobumi Nishida, Hiromitsu Yamada. Optical Character Recognition ISBN: 978-0-471-30819-5 April 1999 560 Pages

Dougherty, Geoff, Pattern Recognition and Classification: An Introduction ISBN 978-1-4614-5323-9 January 2012 203 Pages

Chaudhuri, A. Mandaviya, K. , Badelia, P. , K Ghosh, S. Optical Character Recognition System for Different Languages with Soft Computing. ISBN 978-3-319-50252-6

Expected date of thesis defence

2021/22 WS – FEM

The Bachelor Thesis Supervisor

Ing. Josef Pavlíček, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 19. 11. 2020

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 19. 11. 2020

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 15. 03. 2022

Declaration

I declare that I have worked on my bachelor thesis titled "Automatic Recognition graphic pattern" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 15/03/2022 _____Bunna CHOM_____

Acknowledgement

I would like to thank Ing. Josef Pavlíček, Ph.D, For his suggestions and encouragement during my efforts on this work.

Automatic Recognition of Graphic Patterns

Abstract

Pattern recognition is the automated detection of shapes or patterns and reliabilities in data. It is the performance of algorisms and techniques such as image analysis, statistical data analysis, Artificial Intelligence (AI) and machine learning analyze documents in the form of image with an insistence on those for graphics recognition and translation. Optical character recognition (OCR) is a branch of research in Pattern recognition. It is the technique which implement a system to determine letter or characters in many languages into human's verbal expression without human involvement. Meanwhile, a number of companies work with multiple kind of documents, some of which are digitized and some of which are undigitized. Such undigitized type document are not editable must be re-typed manually into the system. This bachelor's thesis aims to leverage on OCR technologies to build an application that is capable of reading characters and graphic templates (logos, icons, signatures) and transforming those to digitize document.

Keywords: ORC, Shape Recognition, Pattern Recognition, Artificial Intelligence, Tesseract.js

Automatické rozpoznání grafických vzorů

Abstrakt

Rozpoznávání vzorů je automatická detekce tvarů nebo vzorů a spolehlivosti dat. Jedná se o provádění algoritmů a technik, jako je analýza obrazu, statistická analýza dat, umělá inteligence (AI) a strojové učení analyzující dokumenty ve formě obrazu s důrazem na rozpoznávání a překlad grafiky. Optické rozpoznávání znaků (OCR) je odvětvím výzkumu v rozpoznávání vzorů. Je to technika, která implementuje systém pro určování písmen nebo znaků v mnoha jazycích do lidského verbálního vyjádření bez lidské účasti. Mezitím řada společností pracuje s více druhy dokumentů, z nichž některé jsou digitalizované a některé nedigitalizované. Takto nedigitalizovaný typový dokument, který nelze upravovat, je nutné do systému přepsat ručně. Tato bakalářská práce si klade za cíl využít technologie OCR k vytvoření aplikace, která je schopna číst znaky a grafické šablony (loga, ikony, podpisy) a transformovat je pro digitalizaci dokumentu.

Klíčová slova: ORC, rozpoznávání tvaru, rozpoznávání vzoru, umělá inteligence, Tesseract.js

Table of content

List of pictures

List of tables

Chapter 1: Introduction

ABBYY, one of the world's major providers of commercial optical character recognition (OCR) solutions and services, defines OCR as follows:“ The specific processes that enable people to identify things are still being researched, but scientists are already aware of the three fundamental principles that govern object recognition: integrity, purposefulness, and adaptability (IPA). These ideas form the foundation of ABBYY FineReader OCR, which enables it to duplicate natural or human-like recognition in documents.Examine the way FineReader OCR identifies text in the following example: First, the application examines the document image's overall structure and composition. Text blocks, tables, photos, and other page components are divided into sections by this element. Following that, the lines are broken down into words and finally into characters. As soon as the characters have been identified, the algorithm compares them to a collection of pattern pictures to determine their identity. In order to figure out who this character is, it makes a lot of assumptions. Following these theories, the software examines several variations of line breaking into words and characters, as well as numerous versions of word breaking into characters. After evaluating a large number of such probabilistic possibilities, the software ultimately makes a choice and displays the text that has been identified as the correct one.” [1] According to Adnan Ul-2016 Hasan's PhD thesis,“ the job of printed Optical Character Recognition (OCR) is deemed a "solved" problem by many Pattern Recognition (PR) scholars, who believe that the work of printed Optical Character Recognition (OCR) is a "solved" issue. The performance on many different types of documents seems to be extremely excellent, even when using obsolete techniques and on a variety of different typefaces and writing forms. He goes on to say that, although the issue is often regarded to have addressed, this seems to be true only in the case of text written in the most frequently used languages and scripts, which he believes is a mistake. He primarily focuses on the possibilities of building a highly accurate OCR that may be used for scripts and languages that are not often researched.” [2].“ OCR software makes it possible to edit and search a wide variety of documents, including scanned paper documents, pdf files, and digital camera photos. An OCR system has become one of the most effective patterns recognition and artificial intelligence applications. Despite the availability of numerous commercial OCR systems for a wide range of uses, these

machines cannot yet match the accuracy levels of humans when it comes to document reading. It is a part of the machine-recognition family, which is used to identify objects automatically. It is the process of using a recognition system to identify items, collect data about them, and enter data into computer systems without the need for any human intervention. Images, audio, and movies are analyzed to obtain the external data. A transducer is used to convert the actual image or sound into a digital file in order to record data. Afterwards, the file is saved so that it may be studied later on." [3]

Chapter 2: Objectives and Methodology

2.1 Objectives

The aim of the work is the practical use of Optical Character Recognition technologies using JavaScript (Tesseract JS) for reading characters and graphic template (logos, icon, signatures)

2.2 Methodology

Follow the following methodology:

- Perform research of the current state
- Design a suitable application architecture
- Program the application in the selected language and using selected technologies.
- Test the result
- Define conclusions

Chapter 3: Literature Review

3.1 History of OCR

In the field of pattern recognition, character recognition is a subset. Pattern recognition and image processing are both used to develop several concepts and techniques used in optical character recognition (OCR).[4]

Character recognition, on the other hand, was the impetus that propelled pattern recognition and picture analysis into the realm of scientific and technical maturity.

The act of writing, which has long been the most natural manner of gathering information, storing it and transmitting it, is today used not only for communication between humans, but also for communication between humans and machines. In the field of optical character recognition (OCR), there has been an intense research effort not only because of the difficulty in simulating human reading, but also because it provides efficient applications such as the automatic processing of large amounts of paper, the transfer of data into machines, and the creation of a web interface to paper documents. To mimic human functions with machines and to have machines execute ordinary tasks such as reading is a long-held goal of many people. A retina scanner, which was an image transmission device employing a mosaic of photocells, was developed in 1870 by C.R. Carey of Boston, Massachusetts [5–7]. This was the first time that characters were recognized. In the beginning, each character had to be taught with photos of its counterparts, and each font had to be worked on one by one.

The origins of optical character recognition (OCR) can be traced back to 1900, when the Russian scientist Tyuring attempted to design an aid for the visually impaired [8]. As a result of the development of digital computers[9] the first character recognizers debuted in the middle of the 1940s. Early research into automatic character recognition focused on either machine-printed text or a small set of well-distinguished handwritten text or symbols, with the latter serving as the basis for the former. During this time period, machine-printed OCR systems were typically based on template matching, in which an image is compared to a database of images. To extract feature vectors from a binary image of handwritten text, low-level image processing techniques have been applied to the image to produce the feature vectors. Statistical classifiers are then given the results of the various areas of character recognition 13. Successful but limited algorithms have been implemented mostly for Latin characters

and numerals, with some success in other languages. However, considerable research on Japanese, Chinese, Hebrew, Indian, Cyrillic, Greek, and Arabic characters and numerals, in both machine-printed and handwritten cases, has been undertaken[9] as well as on Cyrillic, Greek, and Arabic numerals.

Nipkow[10] followed this up with the invention of the sequential scanner, which was a significant breakthrough for both modern television and reading machines. Several attempts were made to construct devices to assist the blind through trials with optical character recognition (OCR) during the first few decades of the nineteenth century[7]. The contemporary version of optical character recognition (OCR) did not arrive until the mid-1940s, when the first digital computer was introduced. The inspiration for the creation of optical character recognition systems began shortly after that, when people began to consider possible business and commercial uses. By 1950, the technological revolution[5] was accelerating at a breakneck pace, and electronic data processing was emerging as a promising and essential area of research and development. Commercial character recognizers were initially made accessible in the 1950s, when electronic tablets were used to capture the x–y coordinate data of a pen tip's movement. The researchers were able to work on the online handwriting recognition challenge because of this breakthrough[9]. Punched cards were used to enter the data into the computer. The need for a cost-effective method of dealing with the growing amount of data arose as a result. At the same time, the technology for machine reading was maturing to the point where it could be used in practical applications. By the mid-1950s, commercially available optical character recognition (OCR) machines were available[7]. In 1954, Reader's Digest installed the world's first optical character recognition (OCR) reading machine[9]. A typewriter was used to turn typewritten sales reports into punched cards, which were then fed into a computer using this equipment.

The commercial optical character recognition (OCR) devices that appeared between 1960 and 1965 were commonly referred to as first generation OCR[9]. The letter shapes of the OCR machines of this generation were mostly limited, which was characteristic of this generation. The symbols were created specifically for automatic reading by machines. When multi-font machines first appeared on the scene, they could read up too many different fonts. The pattern recognition algorithm used and template matching, which compares the character picture with a library of prototype images for each character of each font, both served to limit the number of fonts that could be created. The reading machines of the second generation first debuted in the mid-1960s and early 1970s[9]. In addition to being able to

distinguish standard machine-printed characters, these systems also possessed the ability to recognize hand-printed characters. When hand printed characters were taken into consideration, the character set was limited to numerals and a few letters and symbols, with no more options. It was IBM 1287 in 1965 that was the first and most well-known system of this type. During this time, Toshiba built the world's first automatic letter sorting system that could recognize postal code digits automatically. Hitachi also developed the world's first optical character recognition (OCR) machine, which was both high-performing and low-cost. There was a lot of effort done in the domain of standardization during this time. In 1966, a comprehensive examination of OCR requirements was completed, and the OCR–A character set was declared as the American standard OCR character set. This font was heavily styled and meant to be easily recognized by optical recognition software while remaining readable by humans. A European typeface, OCR–B, was also created, which had more natural fonts than the American standard font. Attempts were attempted to combine two fonts into a single standard using computers that were capable of reading both standards.

3.2 Techniques of Optical Character Recognition Systems

The essential notion in automatic pattern recognition is to first teach the machine which classes of patterns are possible and what they look like [9]. The characters that appear in OCR patterns include letters, numerals, and some special symbols such as commas, question marks, and other symbols. In order to teach the computer, it is necessary to show it samples of characters belonging to all distinct classes. A prototype or description of each class of characters is created by the machine using the examples provided. During the recognition process, the unknown characters are compared to previously obtained descriptions and allocated to the class that best matches the descriptions. The character recognition training procedure is completed in advance by the majority of commercial character recognition systems nowadays. When new classes of characters are introduced into a system, some systems incorporate training facilities to prepare players for the change.

An OCR system,[6,9], is composed of various components. With the use of an optical scanner, the initial step is to digitize the analog document. A procedure known as segmentation is used to extract each symbol from a region containing text that has been identified. In order to aid feature extraction, the retrieved symbols are pre-processed in order to remove noise. The identity of each symbol is determined by comparing the extracted attributes to descriptions of symbol classes that were gained during a previous learning stage. When it

comes to reconstructing words and numbers from an original text, contextual information comes in handy.

Optical Scanning

OCR starts with optical scanning. Scanner captures digital image of original document. On-line transcription (OCR) uses optical scanners that transform light intensity into grey levels. Documents are printed in black on white. OCR converts multilevel images to bi-level black and white. Thresholding is used on scanners to save memory and computational resources. In fact, the quality of the bi-level image determines the results of recognition. OCR system components are employed where gray levels below the threshold are black and levels above are white. It is sufficient to pre-set a threshold for documents with strong contrast. However, actual documents have a wide range. In these circumstances, more advanced thresholding procedures are required. The best thresholding methods modify threshold to document attributes like contrast and brightness.

Layered scanning of documents demands additional memory and processing power [5].

Location Segmentation

Location segmentation is the next OCR step. Image segmentation identifies image elements. Il faut distinguish between data-filled areas of the paper and figures and graphics-filled areas. An address must be located and isolated from other prints like stamps and corporate logos before being recognized by an automatic mail sorting system. Text segmentation is character or word separation. Most OCR systems break words down into single characters. This is usually done by segmenting each connected component. If characters contact or are fragmented, this strategy fails. Segmentation issues include: (a) extraction of touched and fractured characters (b) noise separation from text (c) misreading images and geometry as text [5].

Pre-processing

Pre-processing is the third OCR component. The raw data is subjected to a number of preliminary processing procedures, depending on the manner of data gathering, in order to make it usable in the descriptive stages of character analysis. A certain degree of noise may be present in the image created by the scanning process. The characters may be smeared or broken depending on the scanner resolution and intrinsic thresholding. Some of these flaws, which can lead to low identification rates, are addressed by smoothing digitized characters

in the pre-processor. Filling and thinning are both involved in smoothing. Filling fills in minor gaps, gaps, and holes in digital characters, while thinning narrows the line width. The most popular method for smoothing involves moving a window across a binary image of a character and applying certain rules to its contents. Normalization and smoothing are included in pre-processing. Normalization is used to create characters with consistent size, slant, and rotation. The angle determines the correct rotation. Hough transform variations are often employed to identify skew in rotatable pages and lines of text [5].

Segmentation

An image with sufficient shape information, good compression, and minimal noise is obtained after pre-processing. Segmentation follows OCR. The character picture has been subdivided here. A character's recognition rate is directly affected by how well it can be segmented. Internal segmentation isolates lines and curves in the cursive scripts. Cursive character segmentation remains an unsolved topic despite numerous remarkable methodologies and strategies created in the past. (1) Explicit, (2) implicit and (3) hybrid character segmentation techniques [5].

Representation

Representation is the fifth OCR element. Any recognition system relies heavily on picture representation. To begin, a recognizer receives images in gray or binary format. To reduce complexity and improve accuracy of optical character recognition systems, most systems use the following techniques: A more compact and distinctive representation is necessary for the algorithms. At the same time, they are invariant to class-specific differences [5]. (a) global transformation and series expansion (b) statistical representation (c) geometric and topological representation (d) [5]

Feature Extraction

Feature extraction is an OCR component. Feature extraction captures crucial symbol features. Feature extraction is one of the most difficult pattern recognition challenges. The simplest approach to describe character is by raster image. Another method is to remove unnecessary qualities but keep the important ones. Techniques for extracting such properties include: distribution of points, transformations and series expansions, and structural analysis. The characteristics are rated for noise sensitivity, deformation, implementation, and usability. (a) resilience against noise, distortions and style variation; (b) practicality in

terms of recognition speed, implementation complexity, and independence. Techniques for extracting features include template matching and correlation, transformations, point distribution, and structural analysis [5].

Training And Recogniton

Seventh OCR component is training and recognition. OCR systems heavily rely on pattern recognition to categorize unknown samples. These include (a) template matching (b) statistical techniques (c) structural techniques (d) and ANNs (a). These approaches aren't mutually exclusive. Occasionally, an OCR technique in one approach can be deemed a member of another. All of the above OCR techniques use holistic or analytic strategies for training and recognition. The holistic technique eliminates segmentation by perceiving the complete character from above. This computational benefit comes at the cost of limiting OCR's vocabulary. Also, the complexity of a single character or stroke reduces recognition accuracy. Conversely, analytic solutions work from the stroke or letter level up to produce meaningful text. It is necessary to use explicit or implicit segmentation procedures, which not only add complexity but also introduce segmentation mistake. With the use of segmentation, the difficulty is simplified to simple single characters or strokes, which can handle unbounded vocabulary with great recognition rates [5].

Post-Processing

Post-processing is an OCR component. Post-processing tasks include grouping and error identification and correction. Strings are linked to 35 text grouping symbols. Plain text symbol recognition yields a set of individual symbols. However, these symbols frequently lack information. Words and numerals are made up of separate symbols. Symbols are grouped into strings based on their document position. Symbols that are close enough are grouped. For fixed pitch fonts, grouping is simple as each character's position is known. The spacing between typeset characters varies. Words are much closer together than letters, allowing for grouping. The issues arise when handwritten text is slanted. The context in which each character appears has not been explored. In advanced optical text recognition challenges, a single character recognition system is insufficient. Even the greatest recognition systems cannot identify every character [5]. Context can only detect or repair some of these mistakes. There are two methods. The first makes use of character groups occurring together. This is done by employing word syntactic rules. The odds of two or more characters

appearing in sequence can be estimated for various languages and used to detect errors. It is thought that a combination of k after h in a word is a mistake in English. Another way is to use dictionaries to detect and rectify errors. A term with a mistake is looked up in a dictionary. If the word is not in the dictionary, it is repaired by replacing it with the closest synonym. Classification probabilities help identify incorrectly classified characters. This technique does not detect errors that change lawful words. The downside of dictionaries is the time spent searching and comparing.[5]

3.3 Tesseract

Tesseract started as a PhD project at HP Labs in Bristol. Between 1984 and 1994, HP created it and it became popular. Tesseract was provided as open-source software by HP in 2005[11]. Google has been working on it since 2006, Tesseract is also thought to be used in Google Cloud Vision AI[11]. Tesseract is a text recognition (OCR) engine that is open source and licensed under the Apache 2.0 license. It can be used directly or through an API (for programmers) to extract written text from photos. It works with a wide range of languages[12]. The most recent version 4, launched in October 2018, has a new OCR engine that employs an LSTM-based neural network approach, which should greatly improve accuracy. Version 4 comes with 123 languages pre-installed[11].

3.4 Architecture

Tesseract OCR is a sophisticated layer-based engine. As indicated in the block diagram in fig. 1, it works in a step-by-step manner. The adaptive thresholding stage detects the image's colour intensities and converts the image to binary images[13].
The image is then subjected to linked component analysis[11], which extracts character outlines as the second phase. This stage is the most important part of the cycle since it performs OCR on images with white text and black backgrounds[14].

Tesseract was perhaps the first[11] to process the input image using these cycles. Following that, the image's outlines are turned into Blobs (Binary Long Objects). The data is then grouped into 13 lines and areas, with subsequent analysis focused on a specific location[11].

The extracted components are cut into words and separated by spaces after extraction. The text recognition process then begins, which is a two-pass procedure. The initial part, as indicated in image 1, is when each word is attempted to be recognized. Each acceptable word is accepted, and the second pass begins to collect the remaining words. This is where the adaptive classifier comes in. The adaptive classifier will then classify text more precisely. To work correctly, the adaptive classifier must first be trained. When the classifier receives data, it must fix any errors and assign the text to its right location[15].

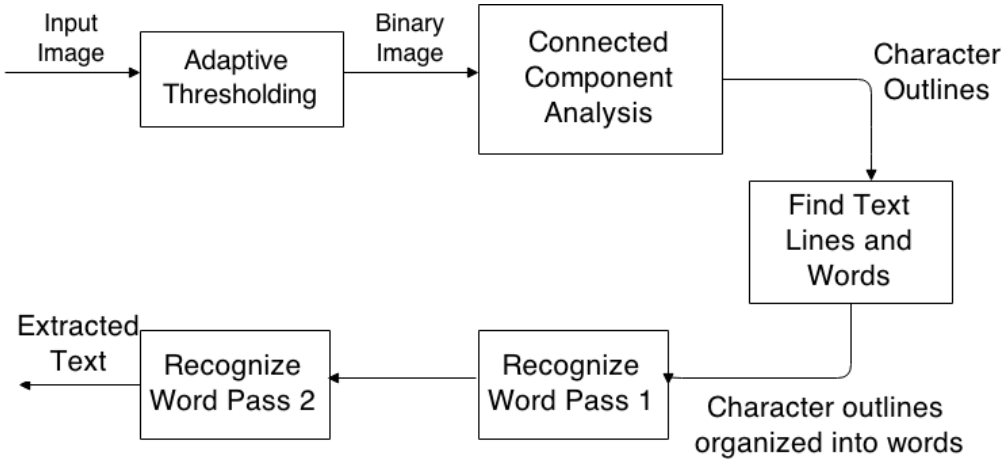


Figure 1 Tesseract architecture

3.5 Tesseract's Operation

Tesseract functions in a similar way as a scanner. Its interface is rather straightforward, as it accepts input from the command line using only the most fundamental instructions. We'll need to upload any image that has text. For example, the illustration in fig.3[16] illustrates this. Tesseract then takes over and processes the data. Figure 3 depicts the command that should be used to do this. The Tesseract command accepts only two arguments in its most

Figure 2 Image contain text

basic form: [16] The first argument is an image that contains text, and the second argument is an output text file, which is often a text file. Tesseract automatically selects the.txt extension for the output file by default. [17] There is no need to specify the output file extension directly because it is automatically detected.

Tesseract is available in a number of different languages. Each language includes a trained language data file that has been customized for that language. The language file must be stored in a location that Tesseract is familiar with. It is recommended that you keep it within the project folder when utilizing it in the project. This folder serves as the Tesseract home folder on your computer. Due to the fact that we are attempting to extract English characters from the photographs in this study, we must maintain English data files. As soon as the processing step is completed, the output file is generated, as illustrated in fig. 4. In straightforward photos, whether in color or black and white (gray scale).

Figure 3 Output of processing done by Tesseract

The Tesseract analytic architecture is created using an iterative pipeline technique that includes revisiting previous steps[11]. The recognition is carried out twice: the first time, a static classifier is employed, and the second time, an adaptive classifier is utilized[11]. Tesseract is designed to recognize text with a tiny skew without needing to deskew the image, even though it is desirable for better recognition to have the text horizontal[11]. The connected component analysis is the initial step in the recognition process[11]. Line detection, baseline fitting, character and word segmentation are all included[11]. The static classifier's output is then input into the adaptive classifier for training[11]. The adaptive classifier is employed during the second recognition, and words that were not previously recognized by the static classifier may now be recognized.

3.5.1 Line Finding

Blob filtering and line creation are essential for determining a baseline. A blob is any content that isn't related to the rest of the image, such as a word or a symbol. The average height of the blobs is estimated, which aids the engine in filtering and removing all little blobs, such as punctuation or noise[11,18].

The blobs will most likely fit together without overlapping and will be parallel to the same slope. The blobs aren't always identified in the correct order. The blobs are given an x-coordinate to keep track of where they should go[11].

When skew is present, this decreases the negative consequences of assigning an inaccurate text. After each blob has been assigned a line, the baselines are approximated using the least median of squares fit. The blobs are then reinserted into their respective lines[11].

3.5.2 Baseline Fitting

Tesseract investigates the lines of blobs a little more closely once they've been discovered. A quadratic spline, or four parallel lines that analyse the blob, is used to set the baselines

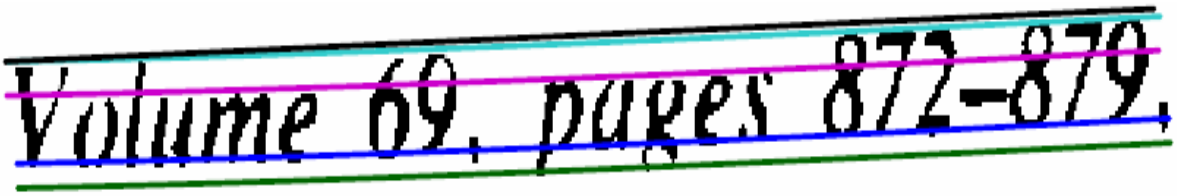


Figure 4 Baseline Fitting

more precisely. This function is extremely helpful in assisting Tesseract in handling curved words, such as those seen in scanned books where the words are frequently curved in the centre near the book bindings[6].

3.5.3 Word Segmentation

Words with identically sized letters (fixed pitch) are treated as a particular situation, with the words cut evenly based on the pitch and marked for identification. Characters in words, on the other hand, frequently have various pitches and must be addressed independently.

Tesseract measures the gaps in a limited vertical range between the baseline and mean line to handle varying pitches[11]. When it comes across spaces that are too close to the threshold, it labels them as fuzzy. Tesseract then defers the judgment until later in the second recognition, when the adaptive classifier may have gathered more helpful information.

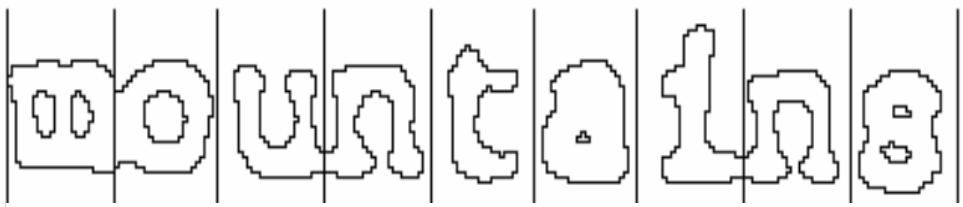


Figure 5 Word segmentation

3.5.4 Character Segmentation

Tesseract seeks to resolve character segmentation by cutting the blob with the lowest confidence provided by the character classifier. The concave vertices of a polygonal approximation of the outlines are used to find potential candidate chop spots[11].

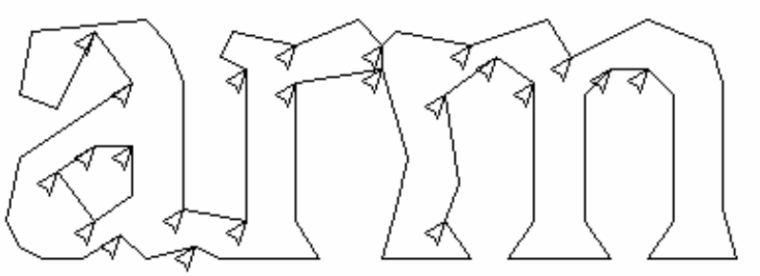


Figure 6 character segmentation

The chops are arranged in a logical order. Any chop that does not boost the result's confidence will be undone, but not completely discarded. The broken character associator will examine it once more[11].

3.5.5 Associating broken characters

If all the possible chops have been exhausted and the term remains unsatisfactory, it is given to the associator. The associator evaluates candidate chops by identifying unclassified

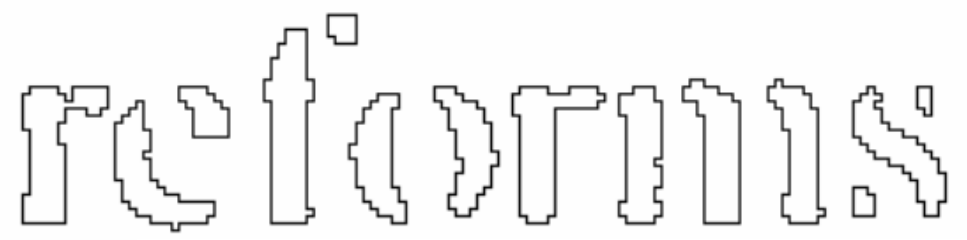


Figure 7 association broken characters

combinations of fragments after trying them out from a prioritized queue. According to R. Smith, this method is inefficient at best and likely to overlook essential chops at worst. The advantage is that words that are lacking crucial portions of fragments can still be identified[11,19].

3.5.6 Character Classification

There are two levels of character classification: pass 1 and pass 2. The static classifier is used initially, followed by the adaptive classifier[19]. The outlines are taken from the character

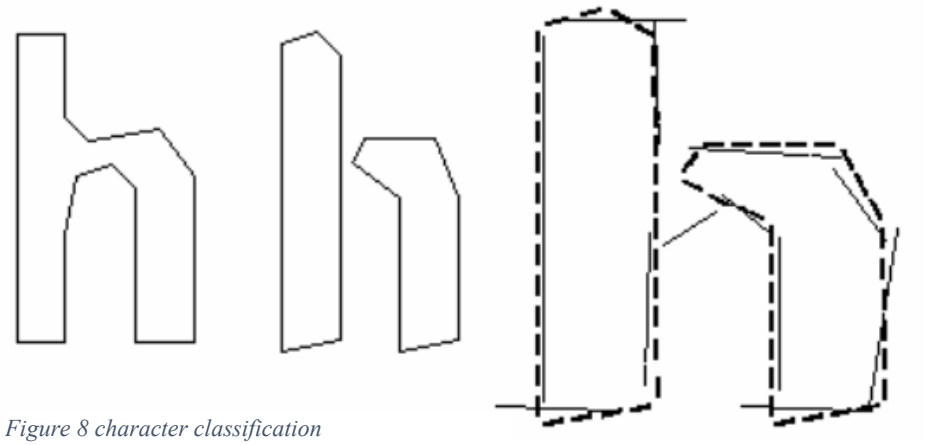


Figure 8 character classification

to be recognized in the first stage, which involves a transit through the static classifier. From the outlines, different sized fixed-length features are retrieved and matched many-to-one to a reference from the training data. Even when character outlines lack characteristics that can be directly checked against a reference, the engine can use polygonal approximation to match linked broken characters to a reference (see Figure 9)[19].

The adaptive classifier is trained on every character that the static classifier correctly matches. This training is required for the second step of the recognition process, in which it will employ the static classifier's general knowledge to help it match previously unsuccessful characters. The adaptive classifier now considers all of the fuzzy spaces as well as the words; once a character passes the adaptive classifier, it is safe to assume it is a match[19].

3.5.7 Word Classification

Tesseract also recognizes words. The characters in a word are separated and then stitched back together one by one. The program constantly compares the current word to a linguistic model. It is undone if the current character does not boost the word's confidence. The highest confidence is considered a complete term[11]. Figure 9shows a DAS 2014 graphic example.

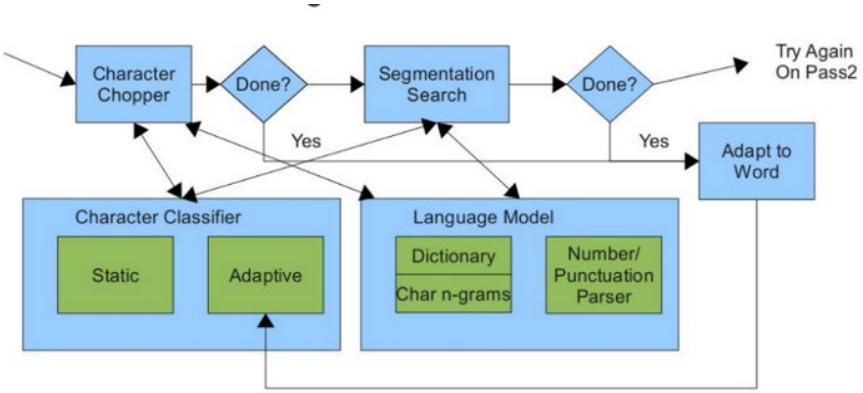


Figure 9 word classicfication

Chapter 4: Current And Future technology

Optical character recognition (OCR) is a well-established technology with a wide range of applications in the corporate world. In essence, it allows users to extract text from a picture for use in a word processor or database. In other words, it transforms a difficult-to-use visual into useful information[20]. So, what are some of the most common OCR framework?

4.1 OCR Library

4.1.1 Google Tesseract OCR

Tesseract was originally created at HP Bristol and HP Greeley in Colorado between 1985 and 1994, with significant adjustments in 1996 for Windows porting and C++ in 1998. HP opened sourced Tesseract in 2005. Google developed it from 2006 to November 2018[21].
It is now one of the most accurate open-source OCR engines available on Linux, Windows, and Mac OS X. The source code is also available for Android and iPhone. It supports 149 languages in various packages. Tesseract is an example-based text detection system, thus we must either obtain the package or train the engine with our own samples.
The advantages of utilizing this engine are that it supports multiple languages and may be compiled to run on multiple systems[22,23].

4.1.2 GORC

GOCR is a free and open-source character recognition software released under the GNU General Public License. GOCR or JOCR – The abbreviation GOCR was used originally. GNU Optical Character Recognition is the name of the software. However, at the time, it had already been taken. After Jörg Schulenburg, the acronym JOCR (Jörg's Optical Character Recognition) was coined (Initial Developer).
GOCR promises to be capable of translating barcodes and covering single-column sans-serif fonts with a height of 20 to 60 pixels. It could also be used for other projects as a command-line interface. It is compatible with the operating systems Linux, Windows, and OS/2[24].

4.1.3 Cuneiform

Cuneiform is a free and open-source solution that is currently known as "Cognitive OpenOCR." There's a database and output built in. It has a total of 23 languages included. It also scans text formats, performs document layout analysis, and performs identification.
OpenOCR is a freeware/BSD application developed by Cognitive Technologies. It is cross-platform; however, it does not provide a graphical user interface for Linux.
Its wrapper library, Puma.NET, makes character recognition work in any.NET Framework 2.0 or above application run more smoothly. To increase recognition quality, it runs a dictionary check during the procedure[24].

4.1.4 Kraken

Kraken was created to address Ocropus's difficulties while not interfering with its other features.
It uses its CLSTM neural network library, which gives it new data experience from past projects. It requires some third-party libraries to run on different platforms.
This data is then used to predict upcoming data validation concerns better correctly. Later, the company's work procedure aids in the development of new models[24].

4.1.5 A9T5

A9T9 is a free and open-source optical character reading and recognition software for Windows developed by Microsoft. It offers an application system for Windows Store that is very simple to use and install.

Other characteristics include zero advertising and a spyware-free operating system. It also features easy-to-customize source codes for improved development and modification possibilities.

OCRopus, Calamari, and Ocrad are some of the other options available[24].

4.2 OCR Applications

The computerized conversion of photos to written text is known as OCR (optical character reader/recognition). There is a lot of OCR software out there that can help you extract text from photographs and turn them into searchable files. These programs take a wide range of image formats and convert them to well-known file formats such as Word, Excel, and Plain Text.

The following is a hand-picked list of OCR Software, along with popular features and links to respective websites. The list includes both free and paid open source and commercial applicationsl[25].

Many different applications have benefited from the use of optical character recognition (OCR). Some of the literatures covering these topics are written in languages other than English, such as Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), Devanagari, Tamil, Chinese, Japanese, and Korean, among others. Latin, Cyrillic, Arabic, Hebrew, Indic the document management method was revolutionized when it was first introduced in the 1950s[26]. It was used across a wide range of sectors and became a highlight of the era. As a result of optical character recognition, scanned documents may now be transformed into more than simply image files, becoming fully searchable documents with text content that is recognized by computers. OCR is becoming increasingly popular. Instead of manually retyping the text, optical character recognition retrieves the important information and automatically enters it into an electronic database, thereby saving time and money. Practical applications, invoice imaging, the legal profession, banking, and healthcare are just a few examples of the many applications available in the field of optical character recognition. OCR is also widely used in a variety of other fields, such as captcha, institutional repositories and digital libraries, Optical Music Recognition (which does not require any human correction or effort), Automatic number plate recognition, Handwritten Recognition, and other industries, amongst others [25].

4.2.1 Nanonents

NanoNets is an artificial intelligence-based OCR program that digitizes data from a variety of corporate documents and photos [2]. By capturing only, the data/information you require, you can automate manual data extraction operations[12]. Automate time-consuming and error-prone manual document processing tasks to boost productivity[12].
Features:

- Data can be extracted from invoices, tax forms, purchase orders, bank statements, insurance forms, medical forms, id cards, and a variety of other sources.
- Only the information you require will be exported to customized Excel, CSV, JSON, XML, or Word files.
- The ability to search PDF
- API response times are lightning fast.
- ERPs, databases, and cloud storage services can all be integrated.
- Compliant with the GDPR
- The software can be installed on your own computer.

4.2.2 Adobe Acrobat

PDF files and photos are converted into searchable and editable documents using Adobe Acrobat, which is an optical character recognition technology. It offers custom fonts that are close in appearance to printouts[27].
Features:

- You can make immediate changes to any printed document.
- It allows you to quickly cut and paste text into other applications without difficulty.
- Exporting the file to Microsoft Office is made possible by Acrobat.
- Using the PDF format, you may easily convert scanned documents and move the data from one area to another.
- This tool assists you in maintaining the appearance and feel of papers that are similar to the original.

4.2.3 ABBYY Cloud Reader

ABBYY Cloud Reader is a software application that recognizes a complete printed or handwritten page in its entirety. It is capable of detecting more than 200 different languages. This program assists you in converting a PDF or image into a searchable MS Word, Excel, PDF, or other format[1].

Features:

- It is compatible with mobile devices as well as desktop computers.
- Receipts and business cards can be recognized by this technology.
- ABBYY Cloud Reader is a RESTful service (Representational State Transfer).
- It transforms recognized data into XML format (Extensible Markup Language).
- This tool includes a library that may be used with Java, .NET, iOS, and Python.

4.2.4 OmniPage Ultimate

OmniPage Ultimate is a piece of software that can convert your document into something that is easy to edit and search for. It can scan files and convert them to any format with little effort[28].

Features:

- Provide document formats that are ready to utilize.
- This program can be used in conjunction with mobile devices and printers.
- There are numerous apps supported, including Microsoft Office and HTML, amongst other things.
- You have the option of opening this program from a network connection.
- This application can recognize more than 120 different languages.

4.2.5 OnlineOCR

Using OnlineOCR, characters and text from PDF documents and photos are recognized. A maximum of 15 photos can be converted into editable text formats every hour with this software[29].

Features:

- It is available in more than 46 languages, including English, Chinese, and French, among others.

- OnlineOCR can handle a variety of file types, including BMP (Bit Map), PNG (Portable Network Graphics), zip files, and more.
- Text can be converted into a variety of formats, including Word, Excel, RTF, and plain text.
- It is possible to incorporate converted files into your website using this service.

4.3 Future Technology

The methods of optical character recognition (OCR) systems have evolved over time from primitive schemes that were suitable only for reading stylized printed numerals to more complex and sophisticated techniques that can recognize a wide range of typeset fonts[30] as well as hand printed characters. Because of the advancement of computer technology and the reduction of computational constraints, new methods for character recognition are constantly being developed[4,9,30]. The greatest promise, however, comes in maximizing the effectiveness of existing methods by hybridizing technologies and making greater use of context. The integration of segmentation and contextual analysis improves the recognition of characters that have been joined and split. Additionally, higher level contextual analysis that examines the semantics of complete phrases is beneficial. In general, there is greater possibility in utilizing context to a greater extent than is currently done. Additional research has shown that a combination of numerous independent feature sets and classifiers, where the weakness of one technique is balanced by the strength of another, can increase recognition of individual characters [5]. Researchers are pushing the boundaries of character recognition research even farther, with the goal of recognizing advanced cursive script, which is handwritten connected or calligraphic characters, soon. In this field, some interesting techniques are being developed that deal with recognition of complete words rather than individual characters.

Chapter 5: Implementation

Tesseract is a C++ engine that runs outside of a browser. As a result, the C++ engine can only be used by sending a picture from a web application to a server, running it through the machine, and returning the text.[11]

However, for a few years, there has been a JavaScript port of the Tesseract C++ engine that works in a browser without any server-side code. Tesseract.js is the name of the library.

I'm using Tesseract.js to create an OCR web application for this work, the OCR frontend is built with HTML, CSS, JavaScript, and CSS library bootstrap

5.1 Setup Languages Selection

Create language selector in html file inside <body> section.

Use <select> element to create drop-down list.

When one language gets selected it will trigger JavaScript at backend.

```
<select id="langsel">
  <option value='eng' selected> English </option>
</select>
```

Figure 11 Language selector setup

```
<body>
    <main>
      <div class="container mt-3">
     <div class="row">
       <div class="col-12 col-md-4 ">
         <select id="langsel">
           <option value='afr'     > Afrikaans            </option>
           <option value='ara'     > Arabic               </option>
           <option value='aze'     > Azerbaijani          </option>
           <option value='bel'     > Belarusian           </option>
           <option value='ben'     > Bengali              </option>
           <option value='bul'     > Bulgarian            </option>
           <option value='cat'     > Catalan              </option>
           <option value='ces'     > Czech                </option>
           <option value='chi_sim' > Chinese              </option>
           <option value='chi_tra' > Traditional Chinese  </option>
           <option value='chr'     > Cherokee             </option>
           <option value='dan'     > Danish               </option>
           <option value='deu'     > German               </option>
           <option value='ell'     > Greek                </option>
           <option value='eng'     selected> English      </option>
```

Figure 10 Language selector

The bridge between frontend and backend is id attribute #langsel.

```
const worker = new Tesseract.TesseractWorker();
worker.recognize(file, $("#langsel").val())
```

Figure 12 TesseractWork language selector

5.2 Setup html element

Image file selector

Using <input> tag which is specifies an input field where the user can upload the image.

Follow the code below:

```
    <div class="col-12 col-md-4 mt-3 mt-md-0">
          <div class="box">
            <input type="file" name="file-1[]" id="file-1" class="inputfile inputfile-1" data-multiple-caption="{count} files selected" multiple />
            <label for="file-1"><svg xmlns="http://www.w3.org/2000/svg" width="20" height="17" viewBox="0 0 20 17"><path d="M10 0l-5.2 4.9h3.3v5.1h3.8v-5.1h3.3l-5.2-4.
3.6l3.4-2.6h-2l-3.2 2.1c-.4.3-.7 1-.6 1.5l.6 3.1c.1.5.7.9 1.2.9h16.3c.6 0 1.1-.4 1.3-.9l.6-3.1c.1-.5-.2-1.2-.7-1.5z"/></svg> <span>Choose a file&hellip;</span></label>
          </div>
        </div>
```

Figure 13 Image input html element

Thumbnail

Using <img> tag which is used to embed an image in an HTML page.

This tag allows us to create thumbnail to preview image selected

Follow code below:

```
<div class="col-12 col-md-5">
        <div class="image-container"><img id="selected-image"  src="images/Funny-Minion-Quotes.jpg" class="col-12 p-0" /></div>
  </div>
```

Figure 14 Image thumbnail

Placeholder of results after processing

The code below creates a box, which is show result after processing.

```
<div class="col-12 col-md-6">
        <div id="log">
            <span id="startPre">
              <a id="startLink" href="#">Results</a>

            </span>
   </div>
```

Figure 15 Placeholder html element

5.3 Tesseract: Setup and Run

In addition, we'll create a TesseractWorker. Then use the recognize feature. This function returns a TesseractJobobject and runs asynchronously.

```
function recognizeFile(file){
  $("#log").empty();
    const corePath = window.navigator.userAgent.indexOf("Edge") > -1
    ? 'js/tesseract-core.asm.js'
    : 'js/tesseract-core.wasm.js';


  const worker = new Tesseract.TesseractWorker({
    corePath,
  });

  worker.recognize(file,
    $("#langsel").val()
  )
    .progress(function(packet){
      console.info(packet)
      progressUpdate(packet)

    })
    .then(function(data){
      console.log(data)
      progressUpdate({ status: 'done', data: data })
    })
}
```

Figure 16JavaScript recognize function

The text result can be obtained by utilizing the then () method inside a callback function. Additionally, use the progress () function to add a callback to track the status and progress of the OCR operation.

5.4 Display Progress and Results

Use the TesseractJob object that is returned to display the results.
It returns a status string and the percentage of work performed during the processing stage.
The returned result includes a confidence level as well as the text extracted from the image.

Figure 18 Result after Image processing

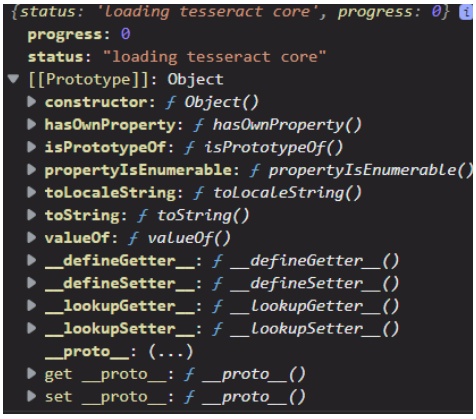


Figure 17 Result after Image processing 2

The location of the word inside the image is also included in the array of words. To show it to the user, we now use the progressUpdate method.

```
function progressUpdate(packet){
  var log = document.getElementById('log');

  if(log.firstChild && log.firstChild.status === packet.status){
    if('progress' in packet){
      var progress = log.firstChild.querySelector('progress')
      progress.value = packet.progress
    }
  }else{
    var line = document.createElement('div');
    line.status = packet.status;
    var status = document.createElement('div')
    status.className = 'status'
    status.appendChild(document.createTextNode(packet.status))
    line.appendChild(status)

    if('progress' in packet){
      var progress = document.createElement('progress')
      progress.value = packet.progress
      progress.max = 1
      line.appendChild(progress)
    }


    if(packet.status == 'done'){
      log.innerHTML = ''
      var pre = document.createElement('pre')
      pre.appendChild(document.createTextNode(packet.data.text.replace(/\n\s*\n/g, '\n')))
      line.innerHTML = ''
      line.appendChild(pre)
      $(".fas").removeClass('fa-spinner fa-spin')
      $(".fas").addClass('fa-check')
    }

    log.insertBefore(line, log.firstChild)
  }
}
```

Figure 19 progressUpadat function

5.5 Architecture

We conducted a thorough search of the existing application in the preceding stages. Preprocessing is essential for higher accuracy in all these applications. In addition, most applications support picture input via still camera capture and browser image mode. Let's begin by sketching out the suggested system's broad architectural layout[31].

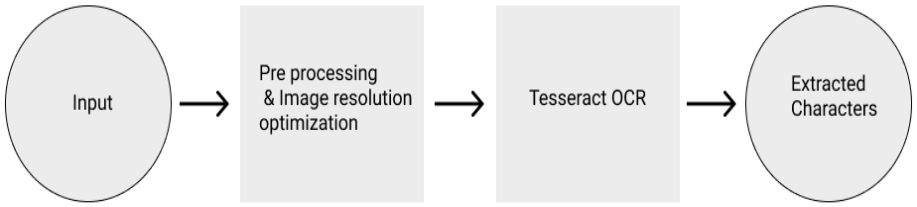


Figure 20 Application process

The system's overall architecture is seen in the graphic to the right. Essentially, it is divided into two major subsystems. The Preprocessing step comes first, followed by the Tesseract API step. The preprocessing stage, which works on the input image to prepare it for use by the Tesseract engine, is the primary emphasis area of this study, and it is described in detail below. In this case, it is important to remember that there is a compromise between processing speed and accuracy. The more time you spend on preprocessing, the greater the accuracy; however, the longer the runtime will be. The relationship is very clearly seen in the diagram below[31].

When it comes to processing and accuracy, the graphic below demonstrates how the distribution/weight for distinct processes in OCR are distributed and weighted. On the processing side, the extraction step consumes most of the available time. It is since the better the characters are extracted, the better the results will be for the recognition stage. Afterwards, recognition and translation are about equal in importance. On the accuracy front, the recognition phase is where most of the error is committed. It is the responsibility of the rule matching subsystem to recognize and match characters in an effective manner[31].

Preprocessing:

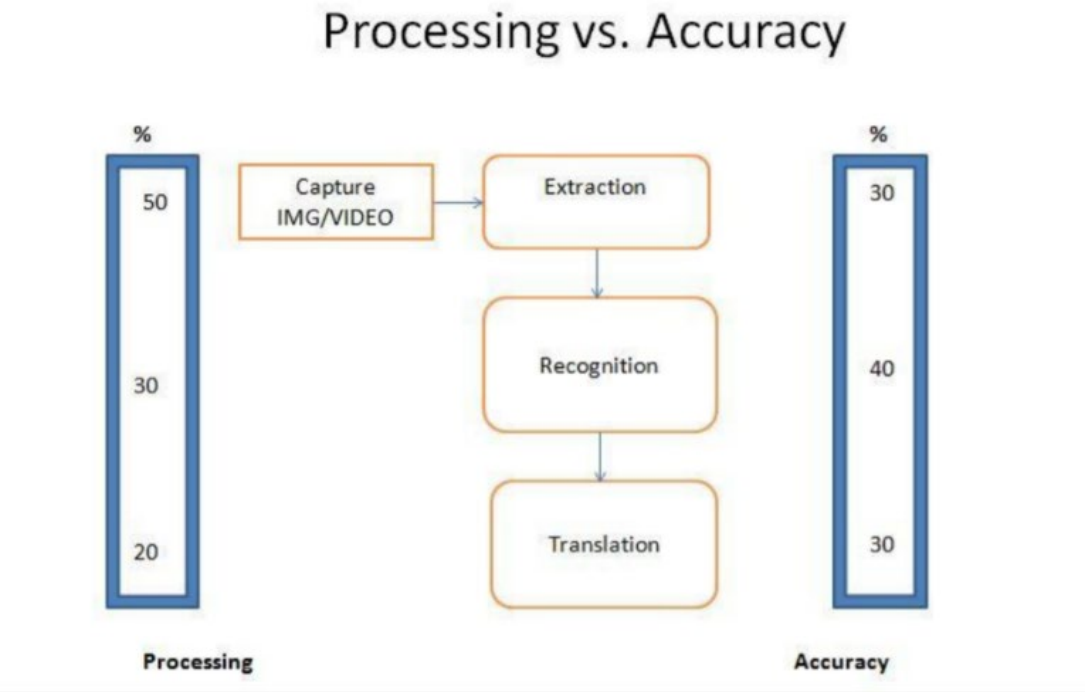


Figure 21 Processing vs Accuracy

The Preprocessing subsystem of the proposed application is depicted in the illustration to the right. After that, the photos are subjected to the Luminosity grayscaling algorithm, which is applied to them after they have been rotated, optimized for resolution, and adjusted for DPI. The processed image is then supplied into Tesseract as an input signal. In case the camera was not pointed at a zero-degree angle when the image was captured, the rotation step rotates the image. Compared to other steps, this is a rather minor change. A smaller step than resolution optimization, compression of larger images to the greatest possible resolution

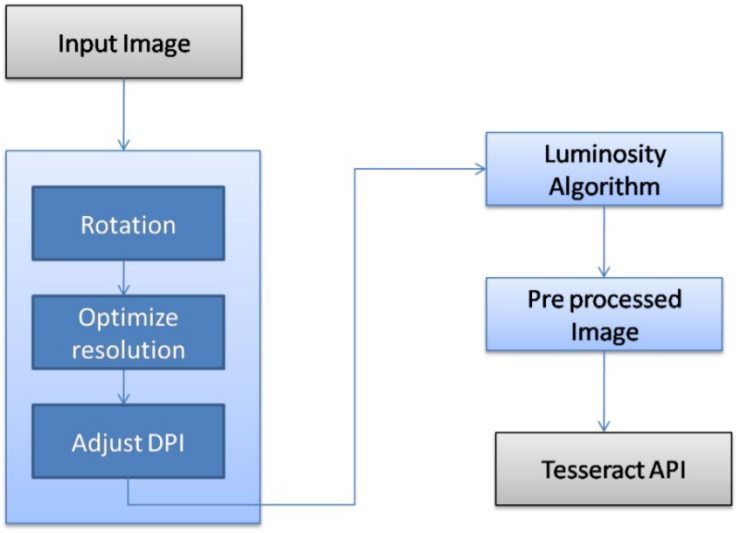


Figure 22 Processing subsystem

for Tesseract is also performed The Dpi adjustment and gray scaling algorithm take up most of the time. They are the primary processes in preparing the image for use with the Tesseract algorithm[31].

Web Application Architecture:

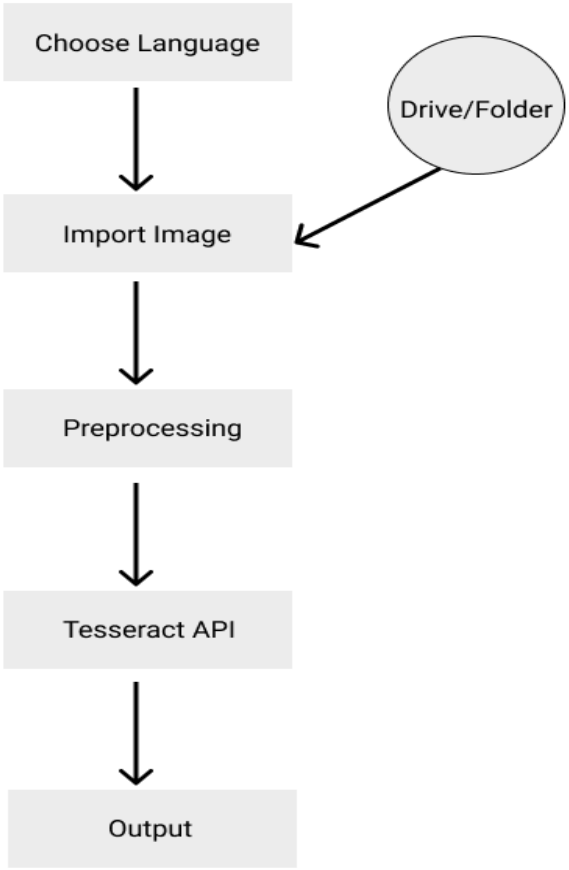


Figure 23 Application Architecture

As seen in the graphic above, the architecture is explained from the perspective of a Web application. The following is the flow of the application: The first step is to select a language, and the second step is to select an input image, which can be selected from a drive or folder where the image is stored. The image is passed to the preprocessing subsystem after it has been selected. Following processing, the image is passed into the Tesseract API, which extracts characters from the image and displays the results on the screen[31].

5.6 Preprocessing step

5.6.1 Luminosity technique

Luminosity threshold grayscale conversion is a technique for turning a picture into grayscale while retaining some of the color intensities in the original image. The Luminosity technique is almost identical to the average color method, but it is more advanced because it considers the human perception of color.

Certain colors are more sensitive to the human eye than others. For example, it is more sensitive to green and less sensitive to blue[31].

$$20 + 70 + 150 over 3$$

Each pixel is converted using the formula provided above. The images below are examples of grayscale images that have been converted using both methods of gray scaling.

Result:

Figure 25 Output of luminosity vs simple gray scale

The following algorithm describes how we process the picture luminosity technique:

In this approach, we explore the image pixel by pixel and then store the results in an array of numbers. Then the luminosity formula, as shown in the following illustration, is used.

```
luminosity =  (0.2126 * red portion + 0.7152 * green portion + 0.0722 * blue potion);
```

This is the primary stage in which we change each pixel to extract the true details from it. Finally, we reassemble all the pixels to reassemble the image once more.

5.6.2 DPI Enhancement

To obtain the best effects out of the photograph, we must also correct the DPI. When there is no distortion, light effects, or other imperfections in the image, gray scaling alone would be effective. Every time we look at something, we won't see something as perfect as this. The following are some of the things we need to do to improve DPI: DPI should be fixed (if needed) Tesseract requires a minimum resolution of 300 DPI. The extraction process is improved because of a wider DPI range[31,32].

```
// Get buffered image from input file;

// iterate all the pixels in the image with width=w and height=h
for int w=0 to w=width
{
        For int h=0 to h=height
        {
                // call BufferedImage.getRGB() saves the color of the pixel

                // call Color(int) to grab the RGB value in pixel

                Color= new color();

                // now use red,green,black components to calc average.

                int luminosity = (int)(0.2126 * red + 0.7152 *green + 0.0722 *blue;

                // now create new values

                Color lum = new ColorLum

                Image.set(lum)

                // set the pixel in the new formed object
        }
}
```

Figure 26algorithm

The rationale for this stage is that we will be using this program on a variety of smart phones, which makes it necessary. There are differences in camera specifications and pixel density between each of them. As a result, it is preferable to normalize the photo before saving it in the gallery to ensure that it is consistent with the Tesseract algorithm[31].

Because the photographs in our case are not completely snapped and ideal, and because we want to keep the procedure small and suited for mobile phones, we'll limit the scope of the work to merely DPI improvement. For our photos, we set the resolution to 300 pixels per inch (as required by Tesseract). If the photographs are larger than this, we reduce the size of the images to a size where we can achieve 300 dots per inch or higher resolution[31].

```
start edge extract (low, high){

            // define edge
               Edge edge;
            //  form image matrix
                Int imgx[3][3]={
               }
               Int imgy[3][3]={
               }
               Img height;
               Img width;

               //Get diff in dpi on X edge

              // get diff in dpi on y edge
               diffx= height* width;
               diffy=r_Height*r_Width;
    img magnitude= sizeof(int)* r_Height*r_Width);
    memset(diffx, 0, sizeof(int)* r_Height*r_Width);
    memset(diffy, 0, sizeof(int)* r_Height*r_Width);
    memset(mag, 0, sizeof(int)* r_Height*r_Width);

             // this computes the angles
             // and magnitude in input img
               For ( int y=0 to y=height)
                   For (int x=0 to x=width)
                         Result_xside +=pixel*x[dy][dx];
                         Result_yside=pixel*y[dy][dx];

              // return recreated image
               result=new Image(edge, r_Height, r_Width)
     return result;

}
```

Figure 27 Algorithm for DPI enhancement

This algorithm's main function is to normalize the pixel density, which can be summarized as follows: Each pixel has a specific size, and the depth of an image is determined by the number of pixels that are packed into it. As previously described, the system may face images with a lower density than the minimum required or a picture that is significantly larger than the minimum required[31].

Chapter 6: Test and Results

6.1 Experiment result of OCR by Tesseract

We have acquired photos of 20 distinct sorts of number plates from various types of automobiles and performed optical character recognition (OCR) on these images to extract the vehicle number. As shown in Table I, Tesseract has 61 percent accuracy with color images and 70 percent accuracy with gray scale photos when dealing with color images. Consequently, when comparing gray scale photographs to color images, it may be concluded that Tesseract delivers greater accuracy[33,34]. On a computer equipped with an Intel Pentium (R) 4 2.4GHz processor and 1 GB of RAM, this experiment was carried out. 5-megapixel camera is used to capture the photos of number plates on the road. If color photos are transformed to gray scale and then provided as input to Tesseract, we can see that the accuracy of text extraction is boosted. When text extraction accuracy is 100 percent or close to 100 percent in color photos, and the image is transformed to gray scale, the text extraction accuracy remains the same or increases somewhat. Tesseract is unable to produce more than 40% accuracy in some color photos; therefore, we have transformed these images into gray scale images using the algorithm outlined in the preceding section and then fed these images into Tesseract as input images for the algorithm. As a result of completing this procedure, the average accuracy with which the characters from the car number plate are extracted improves significantly[35]. It is possible to see in Table I that the accuracy of individual image processing varies from 16 percent to 100 percent, as shown by the image numbers 10 to 18. In addition, it has been noted that the processing time for extracting characters from gray scale photos has been reduced as well. It is cut by 10%, bringing the total down to 50%. As a result, we may conclude that Tesseract processes gray scale number plate images quickly and accurately, and that it gives greater text extraction accuracy than other methods[32,36]. Comparison Study of Tesseract OCR with Transym.

Image No	Image Type	Number of character in image	No of characters extracted	Accuracy of OCR of color images (in Percentages)	Time taken for OCR (in Seconds)	Image Type	No of characters extracted after converting color to gray scale image	Accuracy of OCR of gray scale images (in Percentages)	Time taken for OCR (in Seconds)	Change in Accuracy (In percentages)
1	color	**12**	5	42	0.4	gray scale	5	42	0.397	
2	color	**12**	12	100	0.202	color	12	100	0.202	
3	color	**12**	8	67	0.301	gray scale	8	67	0.601	
4	color	**9**	9	100	0.5	color	9	100	0.5	
5	color	**8**	8	100	0.505	color	8	100	0.505	
6	color	**9**	7	78	0.909	gray scale	7	78	0.909	
7	color	**8**	8	100	0.805	color	8	100	0.805	
8	color	**9**	7	78	1.01	gray scale	7	78	1.01	
9	color	**10**	7	70	0.85	gray scale	7	70	0.798	
10	color	**9**	4	44	0.907	gray scale	5	56	0.402	20
11	color	**10**	1	10	1.007	gray scale	4	40	0.548	75
12	color	**10**	4	40	0.699	gray scale	7	70	0.402	42.86
13	color	**10**	3	30	1.51	gray scale	4	40	0.701	25
14	color	**9**	0	0	1.008	gray scale	4	44	0.705	100
15	color	**9**	0	0	1.815	gray scale	2	22	0.7	100
16	color	**11**	6	55	1.619	gray scale	8	73	1.717	25
17	color	**9**	5	56	0.99	gray scale	6	67	0.806	16.67
18	color	**11**	5	45	0.907	gray scale	6	55	0.596	16.67
19	color	**9**	9	100	3.048	color	9	100	3.048	
20	color	**9**	9	100	1.007	color	9	100	1.007	
			Average Accuracy	**61**			**Average Accuracy**	**70**		

Table 1 Tesseract OCR Result analysis

Transym OCR is one of the proprietary Optical Character Recognition tools, and it also gives a high level of accuracy. As a result, we attempted to run OCR on the same collection of photos that were described in Section 4.4 to see what kind of results Transym produced. Transym converts color photos to grayscale images, after which it performs optical character recognition (OCR) on the images. As a result, when utilizing Transym, there is no need to transform a color image to a grayscale image. Table II presents a summary of the OCR processing results obtained by Transym OCR, which demonstrates that Transym achieves an average accuracy of roughly 47 percent in our set of data[36–38].

A comparison between Tesseract and Transym is carried out to determine which is superior. Many additional optical character recognition (OCR) tools are proprietary and not open source. Several other tools are not capable of providing the same level of accuracy as Tesseract. Because Transym gives a high level of accuracy, we used the trial version of the software to perform OCR on the images given in Table II. Even though it cannot be utilized in the form of a Dynamic Link Library (DLL), Tesseract can be used in another application in the form of a DLL for a variety of purposes since it is not a complete tool but an OCR engine[32,33,35].

When comparing the accuracy of color photos and gray scale images, we can see that Tesseract delivers a greater accuracy of 61 percent for color images and 70 percent for gray

scale images when compared to Transym, which only provides a 47 percent accuracy in our collection of data. In comparison to Transym, Tesseract is significantly faster because it processes a single image in an average of 1 second and 0.82 seconds for color and gray scale images, respectively, whereas Transym processes a single image in an average of 6 seconds and 0.82 seconds. For color and gray scale images, the Standard Deviations of Accuracy offered by Tesseract are 34.21 and 24.64, respectively, which are both fewer than the Standard Deviation of Accuracy provided by Transym, which is 40. As a result, Tesseract is more consistent in terms of giving accuracy for the data set described above. The standard deviations of processing time for color images and grayscale images, respectively, are 0.63 and 0.61, respectively, which are significantly smaller than the standard deviation of processing time for Transym, which is 12 (for color photos). As a result, Tesseract is more consistent in this situation of processing photos in less time[16,39].

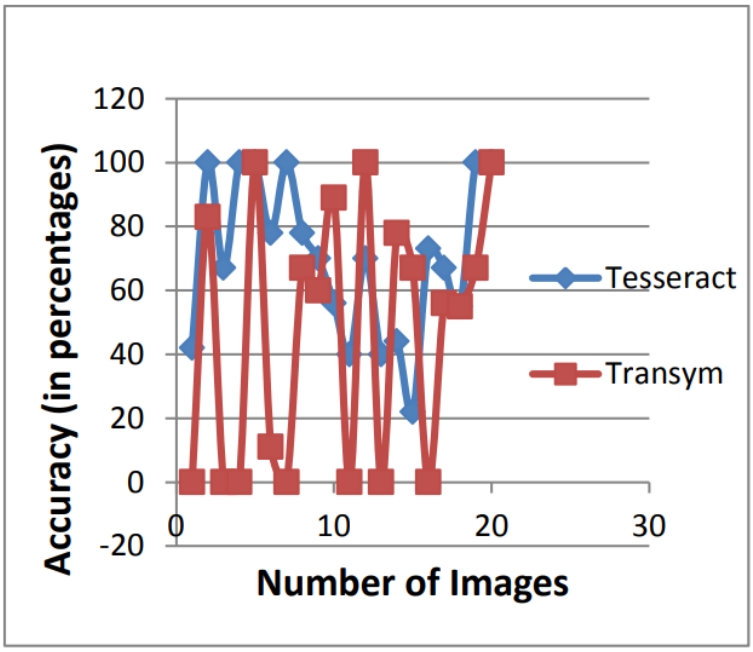


Figure 28 Comparison of accuracy

Table 3 Transym OCR Result

Image No	Image Type	Number of characters in image	No of characters extracted	Accuracy of OCR (in Percentages)	Time taken for OCR (in Seconds)
1	color	12	0	0	54.44
2	color	12	10	83	2.115
3	color	12	0	0	10.58
4	color	9	0	0	2.01
5	color	8	8	100	1.466
6	color	9	1	11	10.725
7	color	8	0	0	7.591
8	color	9	6	67	4.589
9	color	10	6	60	1.816
10	color	9	8	89	2.416
11	color	10	0	0	3.264
12	color	10	10	100	1.104
13	color	10	0	0	1.7
14	color	9	7	78	3.827
15	color	9	6	67	2.904
16	color	11	0	0	2.717
17	color	9	5	56	0.502
18	color	11	6	55	2.504
19	color	9	6	67	17.256
20	color	9	9	100	1.513
			Average Accuracy	47	

Table 2 Tesseract and Transym OCR Comparison

Feature	Tesseract OCR	Transym OCR
Free	Yes	No (Trial Version is available)
Open Source	Yes	No
License	Apache	Proprietary
Online	No	No
Operating System	Window, MaC, Linux	Windows
Latest Stable version	3.01	3
Release Year	2010	2008
DLL Available	Yes	No
Accuracy (For extracting character from vehicle number plate)	61% (color images) 70% (gray scale images)	47%
Average Time	1 second (color images) 0.82 Seconds gray scale images)	6.75 seconds
σ_{AC}	34.21(For color Images) 24.64(For Gray Scale Images)	40
σ_T	0.63 (For color images) 0.61(For gray scale images)	12

σ_{AC} = Standard deviation of Accuracy

σ_T = Standard deviation of Time taken to perform OCR

Figure 29 depicts a comparison graph of the Tesseract and Transym tools, which allows you to evaluate the performance of both tools when the accuracy parameter is taken into consideration. Fig 9 shows that Transym gives 0 percent or near 0 percent accuracy for some photos, indicating that it is incapable of extracting any character from those images. Tesseract, on the other hand, gives 100 percent accuracy for none of the photos, meaning that it is capable of extracting at least some characters from any image. It is necessary to make this comparison once color photos have been converted to gray scale for use with Tesseract[37,38].

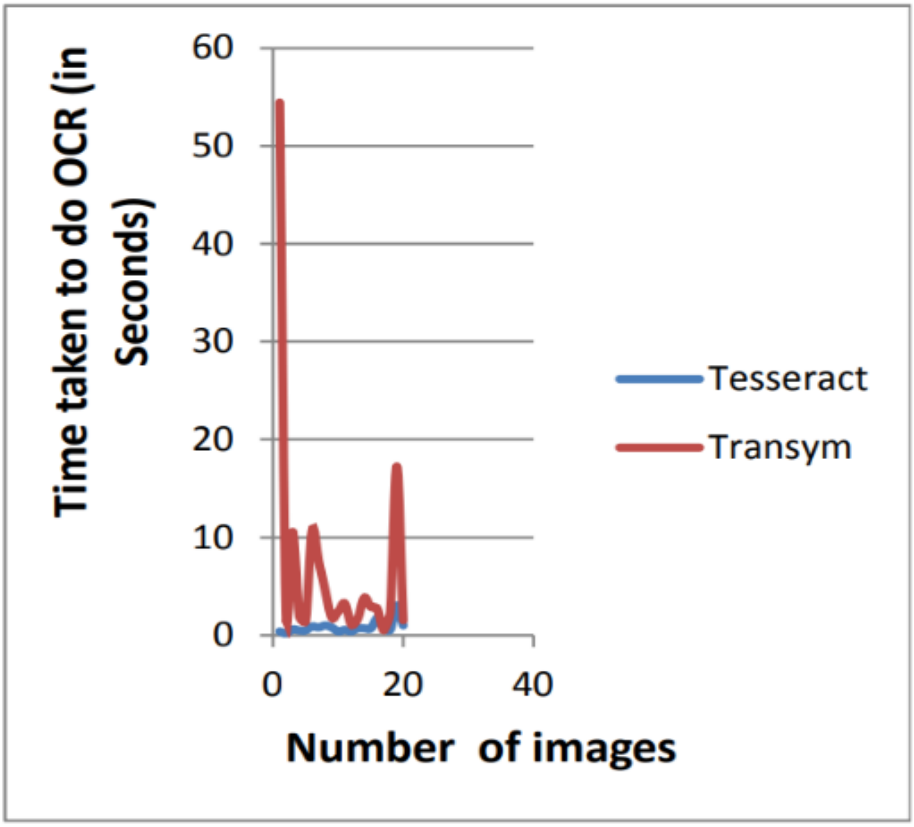


Figure 29 Comparison of processing time of OCR

Another comparison is made between these two tools based on the amount of time it takes to process OCR images. Figure 10 shows that Tesseract takes less than 1 second to OCR several images, whereas Transym takes more than 1 second to OCR any image, as seen in the graph. OCR processing can take anything from one to eighteen seconds for some photos. Figure 10 shows that one of the photographs took more than 50 seconds to complete, as can be seen for one of the images. As a result, we can see that Tesseract is significantly faster than Transym when it comes to OCR processing[34,36–38].

6.2 Test and results on application

6.2.1 Select language

The Tesseract OCR engine is capable of processing data in a variety of languages, for the OCR engine to recognize characters from a specific language, the language must be

Figure 30 Language selection

provided before image have been selected.

While performing OCR, the Language Abbreviation instructs the OCR engine to look for a specific language in the Language Data Path, which should contain the data file for the relevant language if one exists. In this data file, you will find all the information that was utilized to train the OCR engine in the first place.

If you are looking for language abbreviations and data files for Tesseract OCR, you can find them at one of the following links:

- https://github.com/tesseract-ocr/tessdata/
- https://tesseract-ocr.github.io/tessdoc/Data-Files

6.2.2 Input Image

After language has selected, choose image that has some texts in it from folder where its stored. Will be incorporating the Tesseract Engine to turn the image into text in this project. Tesseract scans the image and provides text, which we store and show on our computer.

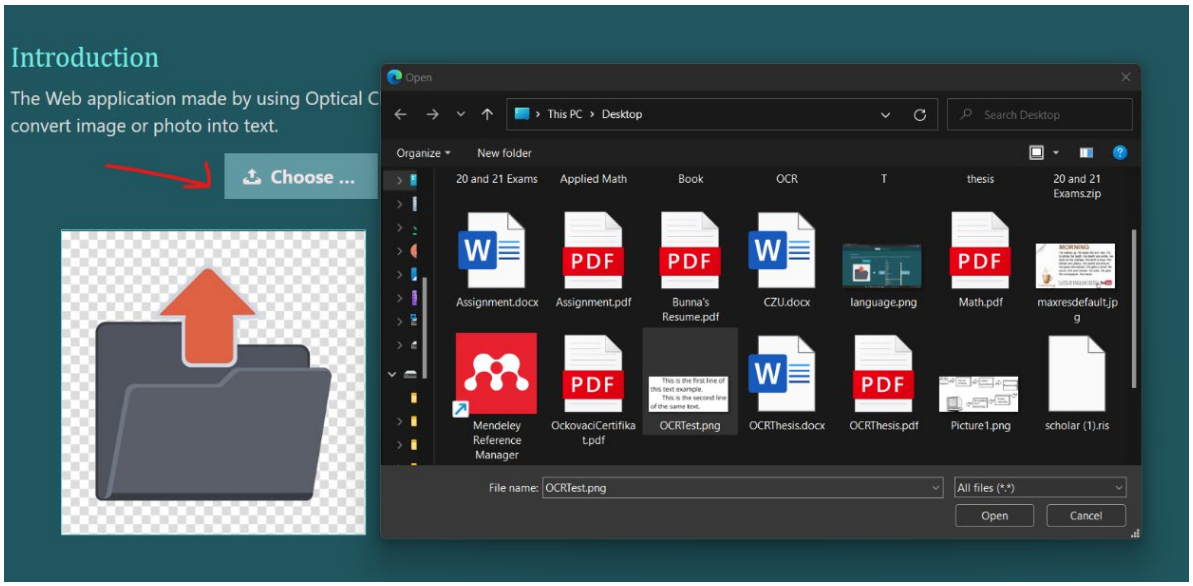


Figure 31 Input Image

6.2.3 Processing

After image have been added and because we're utilizing the capability in ocr/progress, we're seeing the image load immediately.

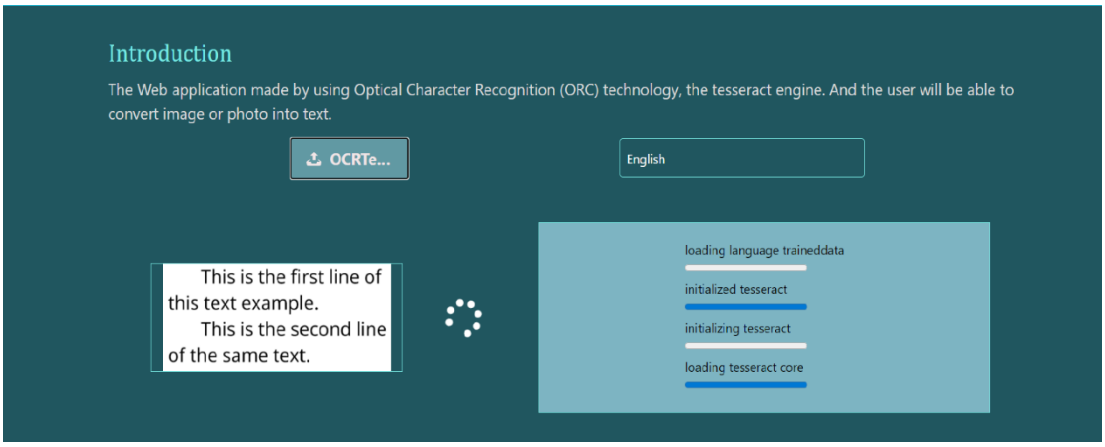
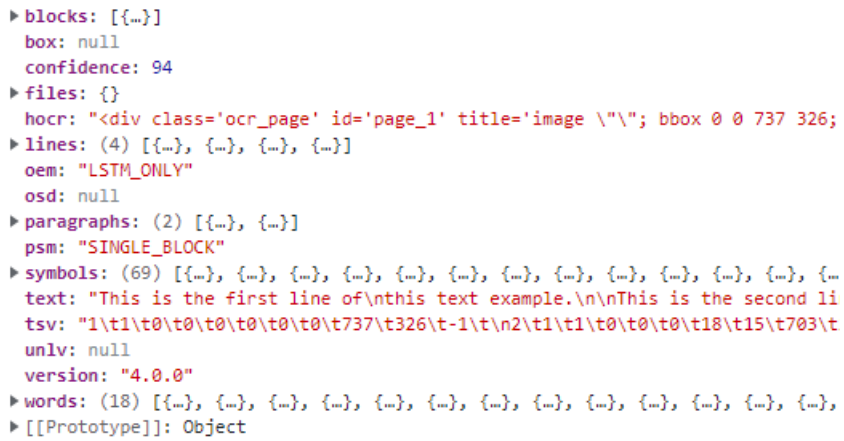


Figure 32 Processing

6.2.4 Results

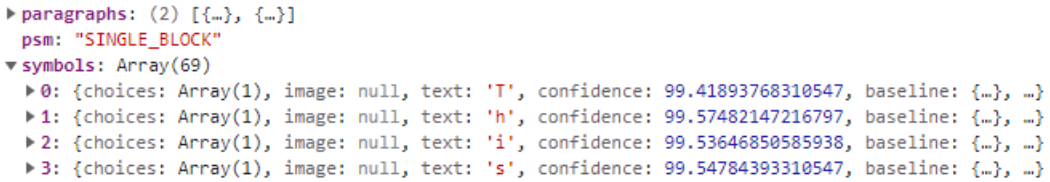
The promise returned by the recognize () method is a Promise. The object you receive because of a successful call comprises the property data, which contains information about the recognized text, as well as other information. You can either access the text through the text property, which contains the recognized text as a single string, or through the lines, paragraphs, words, and symbols properties, which each hold a single string of the recognized text. Each group element carries a confidence score, which indicates how confident the engine is in the group element. Typically, scores range between zero and one hundred; higher values indicate greater confidence.



When you access the text using the properties lines and paragraphs, you will see that the text has been divided into sections by the properties.

You can access each character through the symbols field, which has an array containing every recognized word in the dictionary.

A bounding box is represented by the x and y coordinates of the bounding box, and each element of these properties contains a property called bbox. In the demo application, I use this information to draw a rectangle around the text that has been selected.



As an illustration of an element in the words array, consider the following. The text property holds the word, and the confidence property indicates how confident we should be in the term. line and paragraph are used to refer to the line and paragraph object where this word is found. Each character is represented as a separate element in the array symbols (T, H, I, S).

And, in user interface we will see the result of extracted text in the right box as picture below.

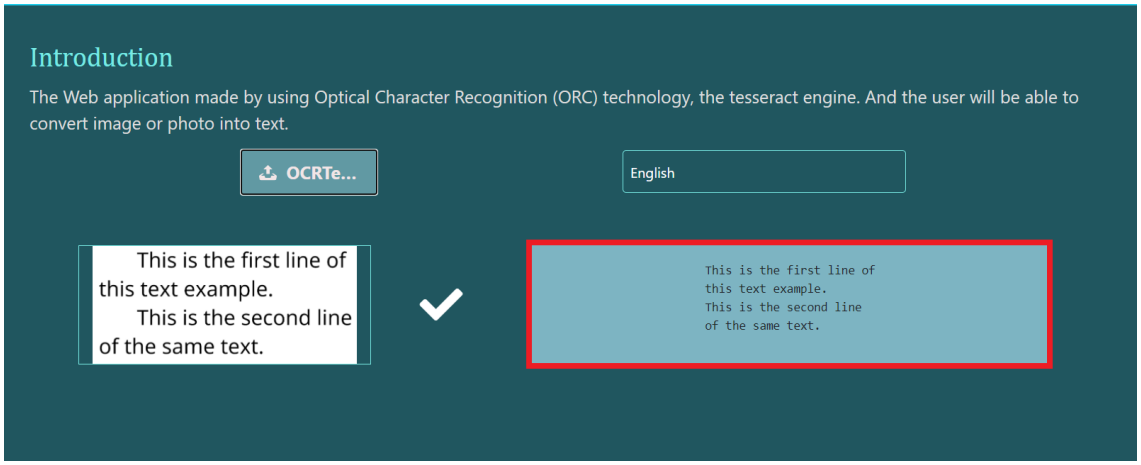


Figure 33 Result box

Chapter 7: Conclusion

As a result of my experiments with various photos, I was able to identify both advantages and disadvantage of Tesseract.js. advantages: Multiple languages are supported with standard fonts and a clear background; the accuracy is quite great.

disadvantage: In noisy environments, it didn't perform as well, several bespoke fonts can cause confusion,

Nonetheless, I believe it to be a fantastic JavaScript library in every respect. When combined with web-based OCR, it provides a wealth of new possibilities for developers and web-based OCR users.

Many papers are still produced on paper today, but automatic data recognition systems are becoming increasingly common.... The document is continually copied and updated during subsequent processing steps, resulting in many distinct copies of the document. In some applications, they can be extremely beneficial to people, but in others, they are completely ineffective. Even though scholars have proposed a variety of complex ideas and strategies to cope with the recognition of unconstrained and connected characters, current optical character recognition (OCR) systems are hampered by a lack of such qualities. It is since the claims made by the researchers have not been adequately substantiated by exposure of the systems to real working environments/conditions that there is a lack of practical feasibility of such advanced techniques with the currently available hardware from an economic standpoint. As a

result of these limits and the lack of shown results, it may be inferred that the capacity of machines to read text with the same fluency as humans is still an unachievable objective, even though a significant amount of effort has previously been invested in the subject. character recognition refers to the recognition of single and unrestrained hand drawn characters, such as numbers, upper-case and lowercase characters of a certain alphabet, that are not part of a larger text. However, the frontiers of character recognition have now expanded to include the recognition of cursive script, which is the recognition of characters that may be joined or written in calligraphy, as well as the recognition of. Characters.

I've learned a lot of things while working on the project, such as using optical character recognition framework (tesseract js) to build very useful application. as we know OCR play very important role in business nowadays. The same time this work gave me chance to improve my programming skills to another level. And I am really to study in this university, especially my department that give me opportunities to working on this great project.

Chapter 8: References

1. ABBYY Cloud OCR SDK - Text recognition via Web API | ABBYY. Accessed March 9, 2022. https://www.abbyy.com/cloud-ocr-sdk/
2. Ul-Hasan A. Generic Text Recognition using Long Short-Term Memory Networks. Published online January 11, 2016. doi:10.13140/RG.2.1.3256.6168
3. OCR Guide - SimpleOCR. Accessed March 10, 2022. https://www.simpleocr.com/ocr-guide/
4. Dholakia K. A survey on handwritten character recognition techniques for various indian languages. *International Journal of Computer Applications*. 2015;115(1).
5. Bunke H, Wang PSP. Handbook of Character Recognition and Document Image Analysis. *Handbook of Character Recognition and Document Image Analysis*. Published online May 1997. doi:10.1142/2757
6. Nagy G, Nartker TA, Rice S v. Optical character recognition: An illustrated guide to the frontier. In: *Document Recognition and Retrieval VII*. Vol 3967. SPIE; 1999:58-69.
7. Schantz HF. The history of OCR, optical character recognition. Published online 1982:114. Accessed March 7, 2022. https://books.google.com/books/about/The_History_of_OCR_Optical_Character_Rec.html?hl=cs&id=VehRAAAAMAAJ
8. Mantas J. An overview of character recognition methodologies. *Pattern recognition*. 1986;19(6):425-430.
9. Chaudhuri A, Mandaviya K, Badelia P, Ghosh SK. Optical character recognition systems. In: *Optical Character Recognition Systems for Different Languages with Soft Computing*. Springer; 2017:9-41.
10. Young TY, Liu PS. VLSI array architecture for pattern analysis and image processing. In: *Handbook of Pattern Recognition and Image Processing*. Academic Press; 1986:471-496.
11. Smith R. An Overview of the Tesseract OCR Engine. Accessed March 9, 2022. http://code.google.com/p/tesseract-ocr.
12. [Tutorial] Tesseract OCR in Python with Pytesseract & OpenCV. Accessed March 9, 2022. https://nanonets.com/blog/ocr-with-tesseract/
13. Shafait F, Keysers D, Breuel TM. Efficient implementation of local adaptive thresholding techniques using integral images. In: *Document Recognition and Retrieval XV*. Vol 6815. International Society for Optics and Photonics; 2008:681510.
14. Patel C, Patel A, Patel D. Optical character recognition by open source OCR tool tesseract: A case study. *International Journal of Computer Applications*. 2012;55(10):50-56.
15. Suen CY. Automatic Recognition of Handwritten Characters. *Fundamentals in Handwriting Recognition*. Published online 1994:70-80. doi:10.1007/978-3-642-78646-4_4
16. text1-how-to-create-typographic-wallpaper.jpg (570×356). Accessed March 9, 2022. https://1stwebdesigner.com/wp-content/uploads/2009/11/typography-tutorial/text1-how-to-create-typographic-wallpaper.jpg
17. Patel Smt Chandaben Mohanbhai CPACMSPSCMD. Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*. 2012;55(10):975-8887.

18. Rousseeuw PJ, Leroy AM. *Robust Regression and Outlier Detection*. John wiley & sons; 2005.
19. Breaking down Tesseract OCR | Machine Learning Medium. Accessed March 9, 2022. https://machinelearningmedium.com/2019/01/15/breaking-down-tesseract-ocr/
20. Where and How is OCR Used | Dynamsoft Document Image Blog. Accessed March 9, 2022. https://www.dynamsoft.com/blog/insights/where-and-how-is-ocr-used/
21. GitHub - tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository). Accessed March 9, 2022. https://github.com/tesseract-ocr/tesseract
22. Text Recognition using Google Tesseract. Accessed March 9, 2022. https://www.folio3.ai/blog/text-recognition-using-google-tesseract/
23. 7 Best OCR libraries as of 2022 - Slant. Accessed March 9, 2022. https://www.slant.co/topics/2579/~best-ocr-libraries#1
24. List of Top 5 Open Source OCR Tools. Accessed March 9, 2022. https://www.hitechnectar.com/blogs/open-source-ocr-tools/
25. 20+ Best Free OCR Software in Feb 2022. Accessed March 9, 2022. https://www.guru99.com/free-ocr-software-tools.html
26. Bacchuwar KS, Singh A, Bansal G, Tiwari S. An Experimental Evaluation of Preprocessing Parameters for GA Based OCR Segmentation. In: *Proceedings of 2010 The 3rd International Conference on Computational Intelligence and Industrial Application (Volume 2)*. ; 2010.
27. How to use OCR software for PDFs in 4 easy steps | Adobe Acrobat DC. Accessed March 9, 2022. https://www.adobe.com/acrobat/how-to/ocr-software-convert-pdf-to-text.html?mv=affiliate&mv2=red
28. Kofax OmniPage Ultimate User's Guide. Published online 2019.
29. Free Online OCR - Image to text and PDF to Doc converter. Accessed March 9, 2022. https://www.onlineocr.net/
30. Cheriet M, Kharma N, Suen C, Liu CL. *Character Recognition Systems: A Guide for Students and Practitioners*. John Wiley & Sons; 2007.
31. Badla S. IMPROVING THE EFFICIENCY OF TESSERACT OCR ENGINE. doi:10.31979/etd.5avd-kf2g
32. Roy A, Ghoshal DP. Number Plate Recognition for use in different countries using an improved segmentation. In: *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*. IEEE; 2011:1-5.
33. Plötz T, Fink GA. Markov models for offline handwriting recognition: a survey. *International Journal on Document Analysis and Recognition (IJDAR)*. 2009;12(4):269-298.
34. Bataineh B, Abdullah SNHS, Omar K. An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows. *Pattern Recognition Letters*. 2011;32(14):1805-1813.
35. Pal U, Roy PP, Tripathy N, Lladós J. Multi-oriented Bangla and Devnagari text recognition. *Pattern Recognition*. 2010;43(12):4124-4136.
36. Desai AA. Gujarati handwritten numeral optical character reorganization through neural network. *Pattern recognition*. 2010;43(7):2582-2589.
37. Jiao J, Ye Q, Huang Q. A configurable method for multi-style license plate recognition. *Pattern Recognition*. 2009;42(3):358-369.
38. Kocer HE, Cevik KK. Artificial neural networks based vehicle license plate recognition. *Procedia Computer Science*. 2011;3:1033-1037.

39. Patel Smt Chandaben Mohanbhai C, Patel A, Chandaben Mohanbhai S, Patel Smt Chandaben Mohanbhai D. Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study. *International Journal of Computer Applications*. 2012;55(10):975-8887.

Chapter 9: Annex

The Web application available here: https://ocr-tesseract-js.web.app/