NANYANG TECHNOLOGICAL UNIVERSITY

CZ4003 COMPUTER VISION

PROJECT

Text Image Segmentation

An Optimal Optical Character Recognition (OCR)

Sam Jian Shen (U1821296L)

Date of Submission: 27th of November 2020

School of Computer Science and Engineering

Nanyang Technological University

Contents

Objectives

1. **Convert** Text Image Binarization from colour or grayscale image into a **binary image** with multiple foreground region (usually characters)
2. **Connected component labeling** that detects each **binarized character region**
3. **Character recognition** by using a classifier

Scope

- **Implement** the **'Ostu' global thresholding algorithm** for binarizing the sample text images and feed the binarized images to the OCR software to **evaluate the OCR accuracy**. Discuss any **problems** with the **Otsu global thresholding algorithm**.
- Design your **algorithms** to **address the problem of the 'Otsu' global thresholding algorithm**, and evaluate OCR accuracy for the binary images as produced by your algorithms. You may **explore different approaches** such as adaptive thresholding, image enhancement, etc., and the **target** is to achieve the **best OCR accuracy**.
- Discuss **how to improve recognition algorithms** for more robust and accurate character recognition while **document images suffer from different types of image degradation**. This is an open and optional task. There will be bonus points if you have good ideas on it.

Assumption

1. Inputs directory only consist of .png and .jpg extension files
2. OCR pre-trained models' accuracy rate is good enough to achieve 100% text recognition.
3. The accuracy is base on our best defined expected outcome.
4. The actual text is base on my best interpretation from the sample images

Tools Used

S/N	Software Name	Recommended Version	Purpose
1	OpenCV	4.5.0 (64-bit)	It is a library that uses for computer vision processes. It helps to implement image processing methods easily such as conversion to grayscale, thresholding images, etc
2	Tesseract	4.1.1 (64-bit)	It is an open-source test recognition API. It converts the image format into text format. It is mainly used to test OCR accuracy.
3	Python	3.8.5 (64-bit)	It is one of the many programming languages that use heavy back-end development which is suitable for image processing.
4	VSCode	1.5.1 (64-bit)	It is the IDE (Integrated Development Environment) software is a graphical user interface (GUI) platform that helps the developer to efficiently develop applications.

Overview Concepts

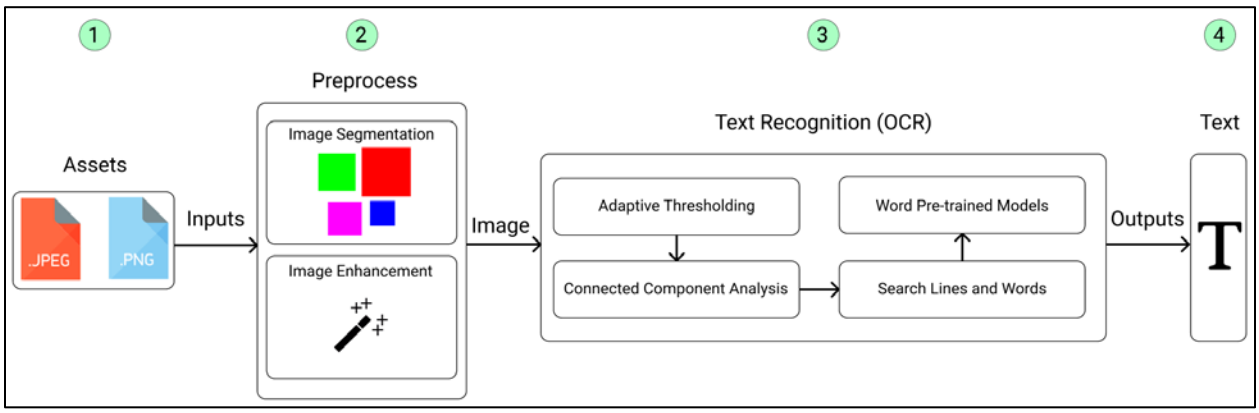


Fig 1. Overview flow of the program

1. The program reads the RAW image from a specific directory that contains all the given files or assets.
2. The image will read and convert into a set of 2D arrays range from 0 to 255 since it is 8-bits per channel. It will be preprocessed such as image partitioning, thresholding, etc
3. The preprocessed image will pass into text recognition, it will formulate the words and line, identify the characters and return a text
4. The text contains a sequential array of letters/words that start from the top left of the letter to the bottom right of the letter in the RAW image. This text may be further postprocessing and be measured for its accuracy.

Analysis

Before we begin the experiment, we first have to understand the underlying problems of the given images such that we can deal with the problem effectively and design the code better. The key features highlighted below may not be implemented depending on the experimentation and outcomes.

Given assets

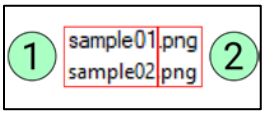


Fig 2. Assets of the project

In this project, we are given 2 images files as shown in the above figure. These files is classified into 2 parts:

1. The filename is the unique identity of the file.
2. The extension of the file indicates a characteristic of the file contents or its intended use. In this case, the PNG (Portable Network Graphics) format is an image file format that supports lossless data compression. It supports a variety type of images such as grayscale, RGB.

Sample 1 Image

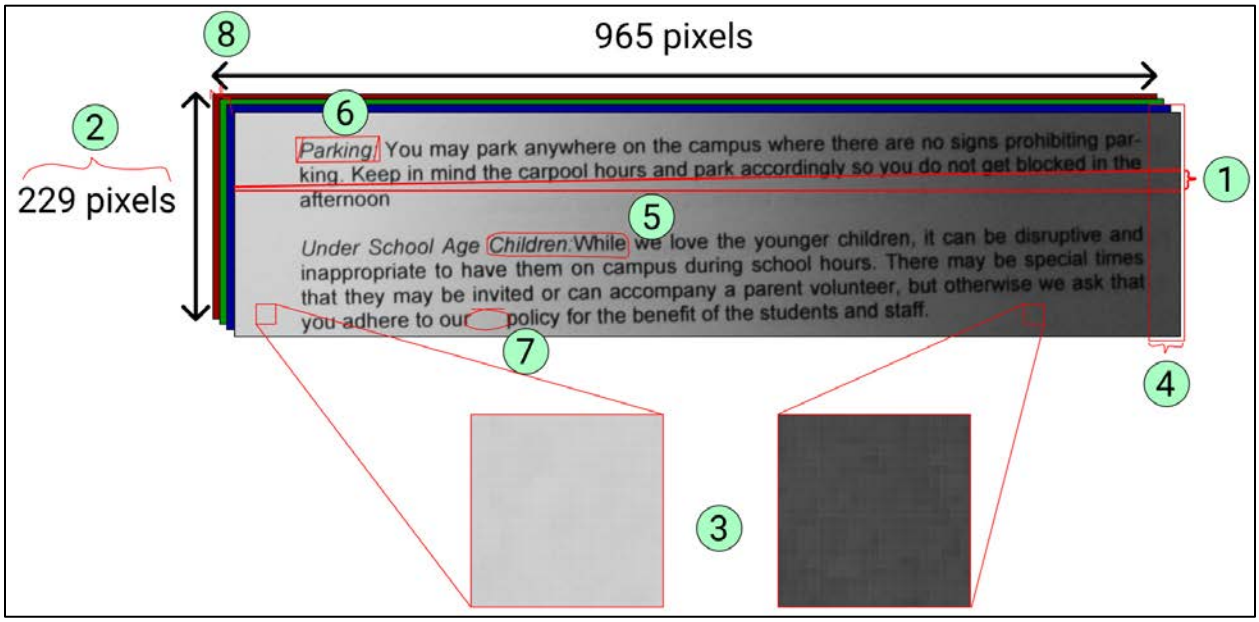


Fig 3. Sample 1 Image Review

1. The image is slightly tilted a few degrees in an anti-clockwise direction. Therefore, image **orientation** needs to be corrected by rotating clockwise direction.
2. The aspect ratio of the image is rather odd. The image might require to **re-scale** proportionally in depend on the letter. This prevents rounding if there is a short column/row of pixels. This may influence the accuracy of the outcome.
3. There is a background gradient of the image. This may impose a problem for darker tone as the OCR algorithm may not be able to distinguish the letter well. **Otsu** method may be applied in a **segmentation** manner to enhance the image.
4. There is unwanted noise (non-text in the area or border) in the image. It must be **trimmed** away to reduce the uncertainty factor detected by the OCR algorithm.
5. There is connected text in the image which may recognize as one full text, perhaps a natural language tool kit (nltk) library may need to use as postprocessing to **distinguish different words** with dictionary library found in 'nltk'.
6. There is an italic text style that may hinder the accuracy of the OCR algorithm. **Re-shape** methods may be used. For example, changing a parallelogram shape into a rectangle shape.
7. There is a wide space(tab size) in one of the sentences and a dash in the sentences. **Text cleaning** may need to use at postprocessing.
8. The image content 32bits which is RGBA (Red, Green, Blue, Alpha), 4 different channels. Since we are not dealing with objects, colors can be omitted as it does not give any meaningful information about the text. It will be converted into an 8bits **grayscale** image.

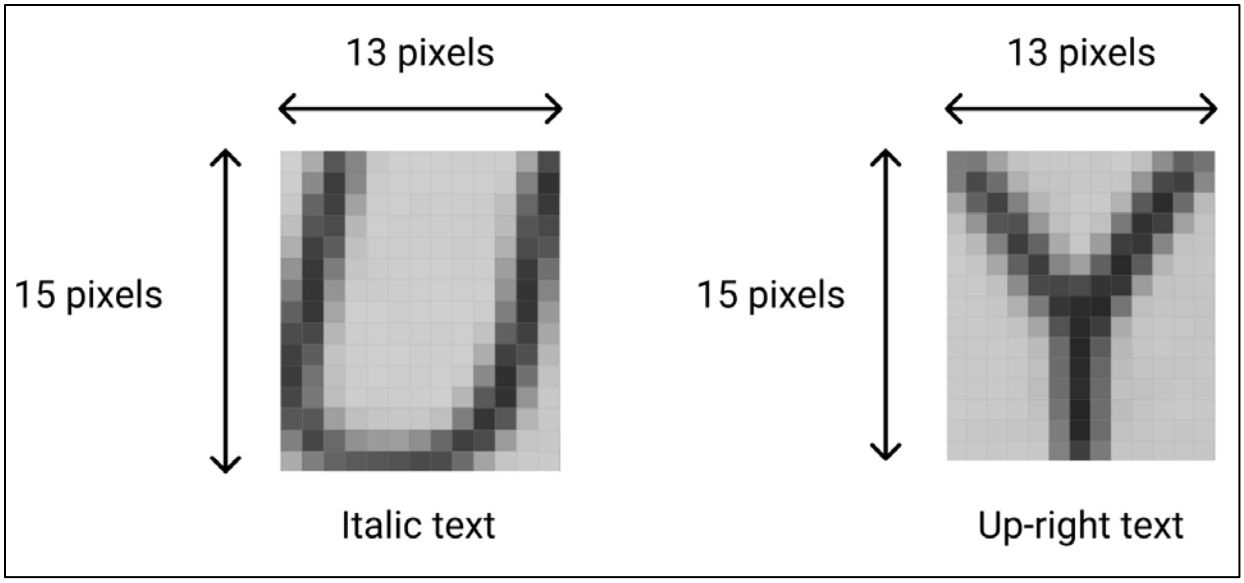


Fig 4. Different Type of Text and Dimensions

There is 2 type of text italic or upright letter(s). Each size is about a 13:15 ratio in dimension. It may be useful when considering the segmentation process.

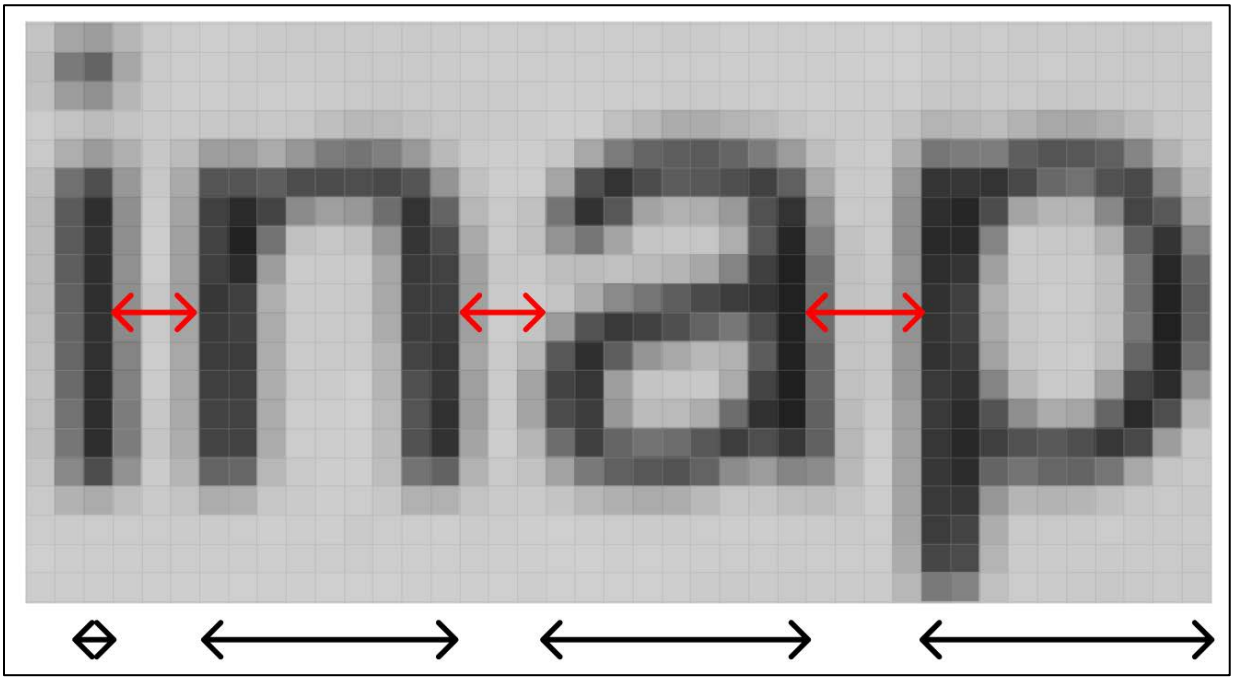


Fig 5. Different Letters Gap and Letter Dimension

Each text has different dimensions and gaps between each other as shown in Fig. 5.

Sample 2 Image

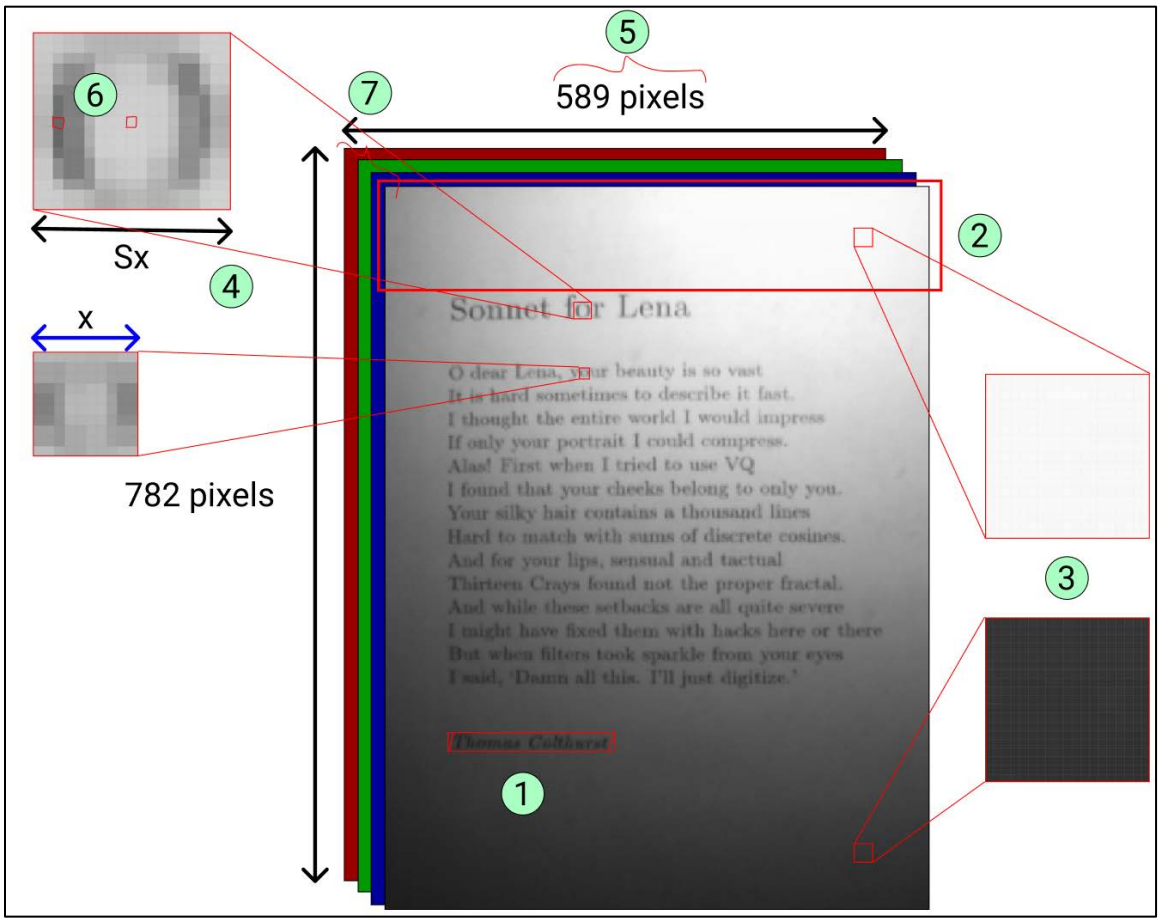


Fig 6. Sample 2 Image Review

1. The text here is in italic text style. This is a similar issue in point 6 in Fig 3.
2. There is unwanted noise that needs to be trimmed away for this image as well. This is a similar issue in point 4 in Fig 3.
3. This image has a more severe background gradient as compared to Fig 3. image. This may prove a challenge for OCR processing. Methods such as 'Ostu'. This is a similar issue in point 3 in Fig 3.
4. There is a different size of the font used in this image, it may pose an issue during segmentation. It should be **dynamic** depending on the text size at a point.
5. The aspect ratio is odd which requires rescaling. This is a similar issue in point 2 in Fig 3.
6. The image is quite blurred, this may significantly impact the OCR algorithm as from Fig 6. The pixel intensity range is slim. **Contrast methods** such as contrast stretching, histogram equalization may be used to enhance this portion of the image.
7. The image contains 4 channels similar to point 8 in Fig 3.

Discussion

- Sample 1 image has more image problems to deal with than sample 1 image.
- Sample 2 image has more image degradation as compared to sample 1 image. Thus, the text recognition for the sample 2 image may be more challenging than the sample 1 image.
- Both images are **computer-generated text**.

Measurement

Actual Text

Before introducing the measure, we need to know our ideal outcome base on the sample. Below are the 2 sample images expected output in a single string datatype base on my interpretation.

Parking: You may park anywhere on the campus where there are no signs prohibiting par- king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon Under School Age Children:While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

Fig 7. Sample 1 expected output

Words Count = **91**

Characters Count = **515**

Characters Count without whitespace = **425**

Sonnet for Lena O dear Lena, your beauty is so vast It is hard sometimes to describe it fast. I thought the entire world I would impress If only your portrait I could compress. Alas! First when I tried to use VQ I found that your checks belong to only you. Your silky hair contains a thousand lines Hard to match with sums of discrete cosines. And for your lips, sensual and tactual Thirteen Crays found not the proper fractal. And while these setbacks are all quite severe I might have fixed them with hacks here or there But when filters took sparkle from your eyes I said, 'Damn all this. I'll just digitize.' Thomas Colthurst

Fig 8. Sample 2 expected output

Words Count = **116**

Characters Count = **629**

Characters Count without whitespace = **514**

Consideration

String Format

```
text = text.replace(u"\u2018", "'").replace(u"\u2019", "'").replace(u"\u201d", "\"").replace(u"\u201c", "\"")
```

Fig 9. Script to substitute for unrecognized character by the python interpreter

As shown in Fig 9. some ASCII characters such as **double and single quotation Unicode** are **substitute ‘** or **“** respectively to prevent wrong interpretation of string.

Standardization

```
text = re.sub(' {2,}', ' ', text)
text = " ".join(text.split())
```

Fig 10. Script to remove tabs, newlines, 2 spaces, and beyond

In this project, since we are not focusing on enhancing the ‘post-process’ of the algorithm, we are **omitting** the **newline and tabs** to keep our ‘post-process’ simple and standardize as shown in Fig 10.

Levenshtein Distance (LD)

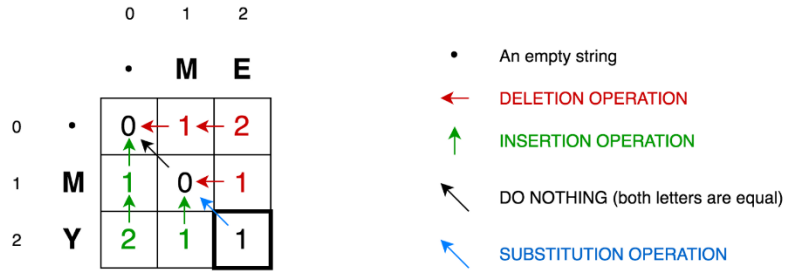


Fig 11. Visual representation of the Levenshtein distance concept

Definition

It is one of the '**edit distance**' techniques to measure how similar or dissimilar two strings are to one another. Strings can be a form of the sentence, paragraphs or words, etc. It computes the minimum number of 'steps' required to transform one string to the other. It returns a distance value which compromises the number of **insertions, deletions, or substitutions** operation that requires to transform from source string to target string as shown in Fig 11.

Why choose this?

Due to the **limitation and complexity** of measuring base on **character-level or word-level count** technique, as it may prove inaccurate since it does not consider the relative position of the character or word respectively, even if it considers the position of the strings, it is too complex to compute. The study has shown that LD has **proven** to be **effective** in dealing with OCR limitation [1]. Thus, this has motivated me to use this technique for our accuracy evaluation.

Evaluation Metrics

- **Distance** is a value computed using the LD formula.
- **Score** is calculated relative to the worst case of LD value which is the max number of characters of the string, such that it is normalized between 0 to 100 as illustrated in Fig 12 below. It helps in terms of readability and good use when comparing to different samples.

(1 – (distance/ total number of characters in the string)) x 100

```
score = round((1 - d/len(act_string)) * 100,4)
```

Fig 12 Score formula (top), in code (bottom)

Evaluation Design

S/N	Algorithms	Parameters Value	Sample 1		Sample 2	
			Distance	Score	Distance	Score
1	Otsu					
2	Enhanced					

Fig 13. A table evaluation design

Fig.13 shows the table for evaluation that will be used for our discussion.

Implementation

Design

Pipeline Software Architecture

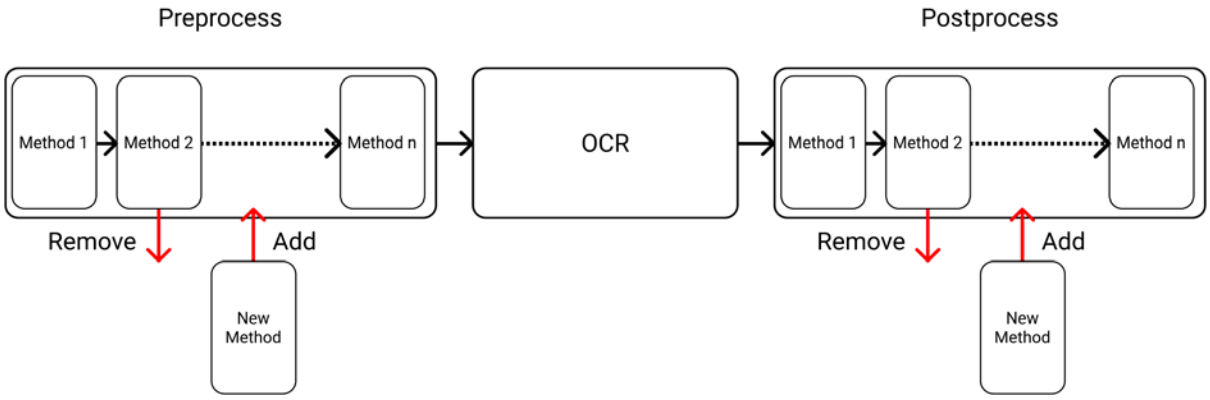


Fig 14. The software architecture of the program

I choose pipeline(sequential) software architecture as shown in Fig 14 because it allows functions to add or remove easily. This is suitable as we will need to implement and try out different methods in the 'pre-process' or 'post-process' section of our program.

Libraries Used

[1] Import **os, shutil, json**

[2] Import **cv2**

[3] Import **math, itertools**

[4] Import **numpy, pandas**

[5] Import **pytesseract**

[6] Import **re**

[7] Import **PIL**

[8] Import **matplotlib**

1. It is responsible for reading and writing of our program operations such as predicted strings from tesseract operation
2. OpenCV was commonly use for implementing image processing operations such as reading images, image enhancement, and thresholding features.
3. It uses for math functions such as combination, floor, ceiling, etc.
4. It is used to manipulate arrays and control the data easily
5. It helps python developers provide intuitive coding from tesseract OCR API, it consists of features sufficient for our task in determining better OCR accuracy.
6. It is used to manipulate the characters and letter of the strings easily
7. It is used for DPI features
8. It is used for plotting graphs with different types, shapes, and sizes.

OCR Algorithms Integrity Test

Purpose

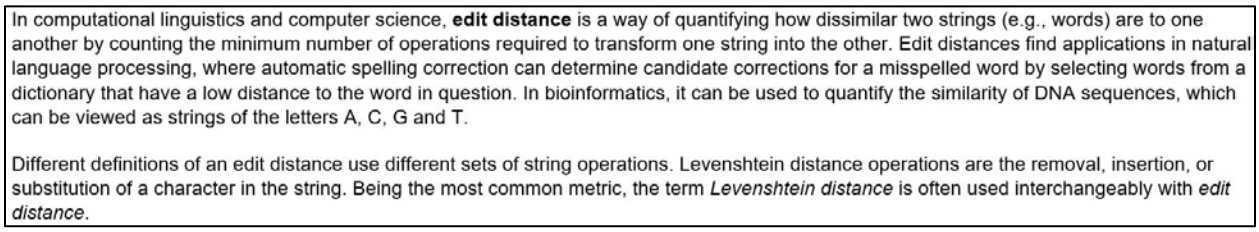
This is to ensure the OCR accuracy in the ideal image is satisfy level and also ensure any deviation from the ideal image still maintain consistent accuracy before actual use for our project.

Assets Used

Below are 5 images selective used, it consists of different font styles and font sizes as shown below. The text is taken from the 1st and 2nd paragraphs of the 'edit distance' Wikipedia page. The image has a minimum of 1553 x 271 pixels with 96 dots per inch (DPI).

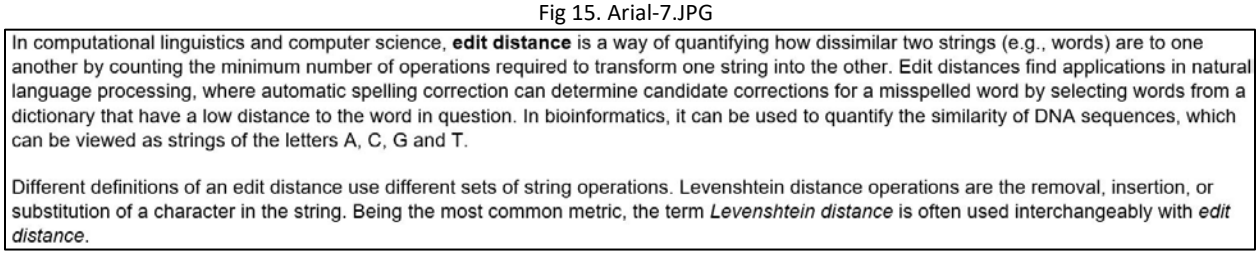
Source: https://en.wikipedia.org/wiki/Edit_distance

S/N	Image Filename	Font Size	Font Style
1	Arial-7.JPG	7	Arial
2	Arial-9.JPG	9	Arial
3	Arial-11.JPG	11	Arial
4	Calibri-11.JPG	11	Calibri
5	TimesNewRoman-11.JPG	11	Times New Roman


In computational linguistics and computer science, **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

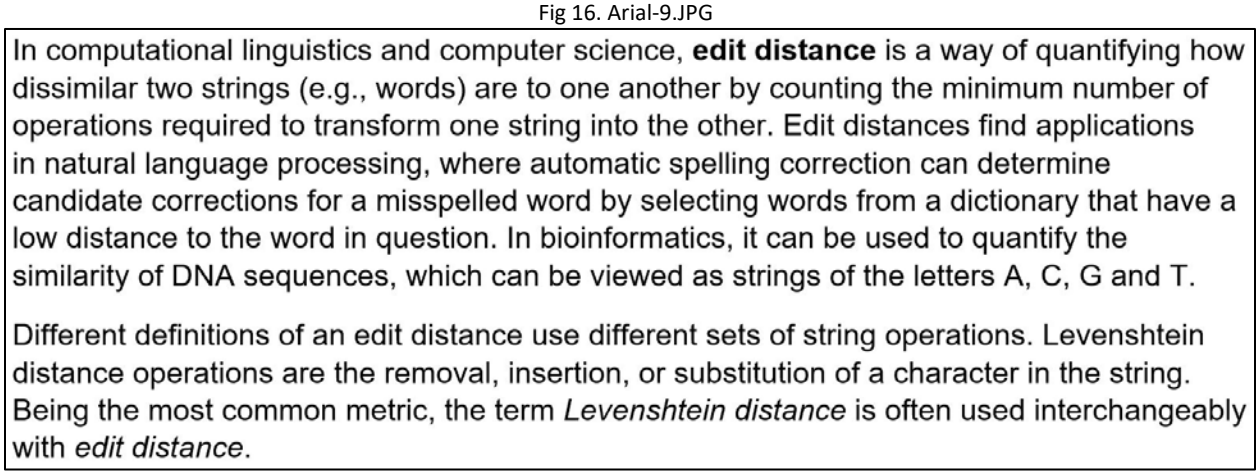
Different definitions of an edit distance use different sets of string operations. Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the term *Levenshtein distance* is often used interchangeably with *edit distance*.

Fig 15. Arial-7.JPG


In computational linguistics and computer science, **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

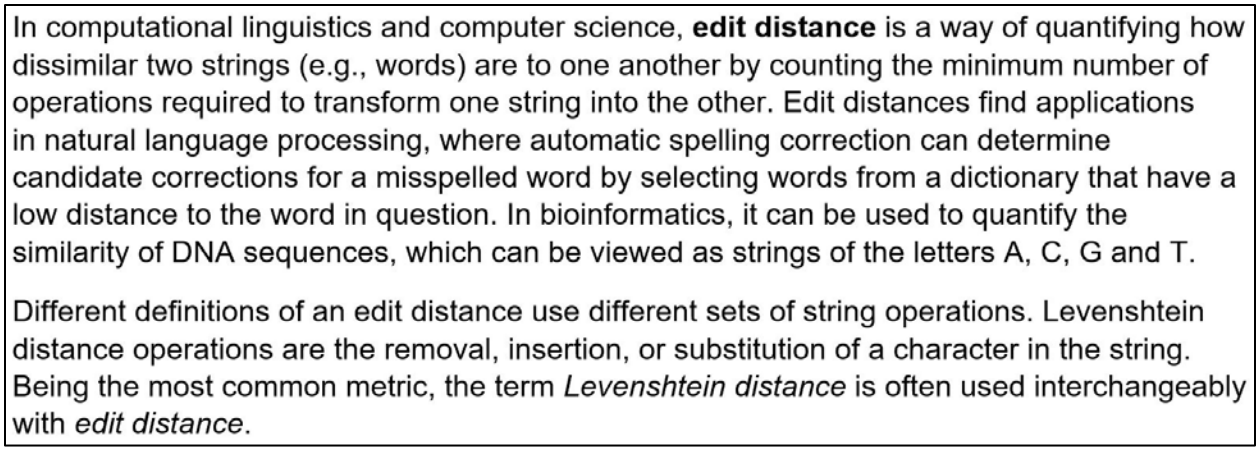
Different definitions of an edit distance use different sets of string operations. Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the term *Levenshtein distance* is often used interchangeably with *edit distance*.

Fig 16. Arial-9.JPG


In computational linguistics and computer science, **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

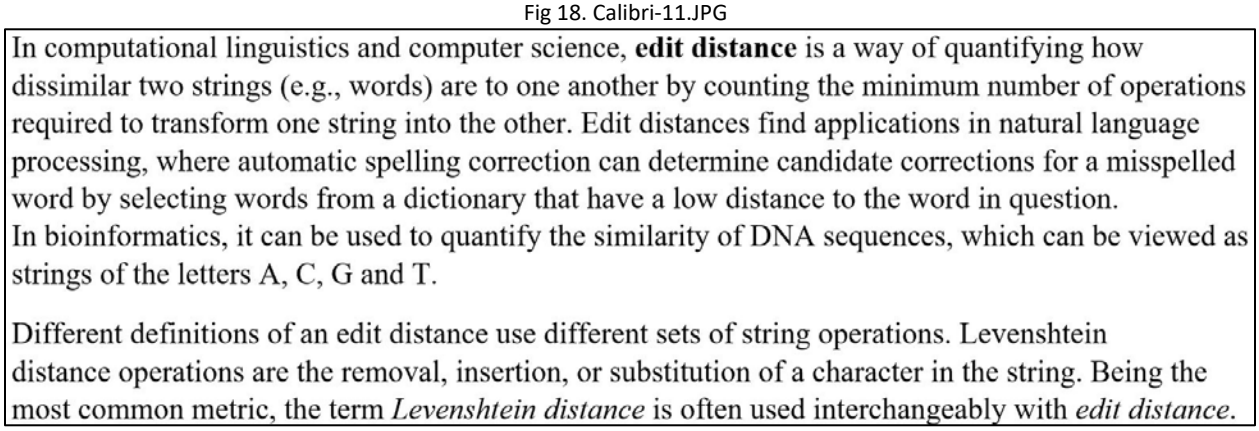
Different definitions of an edit distance use different sets of string operations. Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the term *Levenshtein distance* is often used interchangeably with *edit distance*.

Fig 17. Arial-11.JPG


In computational linguistics and computer science, **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

Different definitions of an edit distance use different sets of string operations. Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the term *Levenshtein distance* is often used interchangeably with *edit distance*.

Fig 18. Calibri-11.JPG


In computational linguistics and computer science, **edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other. Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. In bioinformatics, it can be used to quantify the similarity of DNA sequences, which can be viewed as strings of the letters A, C, G and T.

Different definitions of an edit distance use different sets of string operations. Levenshtein distance operations are the removal, insertion, or substitution of a character in the string. Being the most common metric, the term *Levenshtein distance* is often used interchangeably with *edit distance*.

Fig 19. TimeNewRoman-11.JPG

Overview

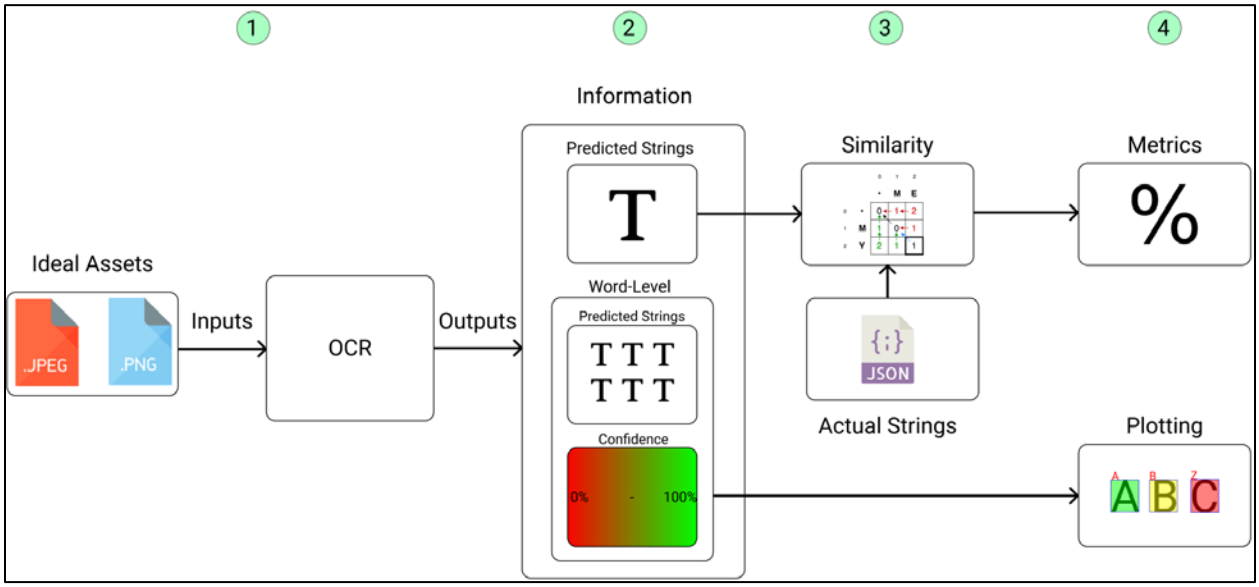


Fig 20. The implementation flow to test the OCR accuracy.

1. The RAW images are directly fed into the OCR algorithm as itself does not require any preprocessing. The configuration settings are standardized and set to default as shown in Fig 21 below, this is to ensure fairness in the result.

```
text = tes.image_to_string(image, config = '--psm 3 --oem 3')
data = tes.image_to_data(image, output_type='data.frame', config = '--psm 3 --oem 3')
```

Fig 21. Code to retrieve image full string (top) and image word information (bottom)

2. Once OCR is done it will able to retrieve the respective outputs such as predicted strings and other meaningful information. To understand the analysis better, I get word-level strings and their corresponding confidence values as well, to determine if the image enhancement algorithm is in the right direction in terms of improving the accuracy (score) value.

	level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
0	1	1	0	0	0	0	0	0	1574	497	-1	NaN
1	2	1	1	0	0	0	7	7	1559	320	-1	NaN
2	3	1	1	1	0	0	7	7	1556	225	-1	NaN
3	4	1	1	1	1	0	8	7	1426	35	-1	NaN
4	5	1	1	1	1	1	8	8	31	26	96	In
5	5	1	1	1	1	2	49	7	217	35	95	computational
6	5	1	1	1	1	3	278	7	154	35	96	linguistics
7	5	1	1	1	1	4	444	7	54	27	96	and
8	5	1	1	1	1	5	509	10	143	32	96	computer
9	5	1	1	1	1	6	663	7	118	33	96	science,
10	5	1	1	1	1	7	794	8	60	26	96	edit
11	5	1	1	1	1	8	865	8	132	26	96	distance
12	5	1	1	1	1	9	1008	7	23	27	96	is
13	5	1	1	1	1	10	1043	16	16	18	96	a
14	5	1	1	1	1	11	1069	16	63	26	96	way
15	5	1	1	1	1	12	1143	7	34	27	96	of
...												
150	5	1	2	1	3	5	408	465	69	24	93	term
151	5	1	2	1	3	6	485	462	182	27	92	Levenshtein
152	5	1	2	1	3	7	679	462	126	27	96	distance
153	5	1	2	1	3	8	816	462	23	27	96	is
154	5	1	2	1	3	9	851	462	77	27	96	often
155	5	1	2	1	3	10	938	462	70	27	96	used
156	5	1	2	1	3	11	1019	462	243	35	95	interchangeably
157	5	1	2	1	3	12	1272	462	67	27	96	with
158	5	1	2	1	3	13	1350	462	57	27	96	edit
159	5	1	2	1	3	14	1416	462	134	27	96	distance.

Fig 22. Information retrieved from word-level data in dataframe representation

3. Predicted strings will be compared with actual strings and compute the similarity using the LD algorithm respectively, the actual string is retrieved from a JSON file as shown in Fig 23 below.

CV_PROJ > CZ4003_CV_PROJ > test > output > predict > {} dataset.json > ...

```
{
    "arial-11_test": "In computational linguistics and co
    "arial-7_test": "In computational linguistics and com
    "arial-9_test": "In computational linguistics and com
    "calibri-11_test": "In computational linguistics and
    "timesnewroman-11_test": "In computational linguistic
}
```

Fig 23. A JSON file that store actual strings for the test

4. The computed result will be stored in a text file as shown in Fig 24 below and the image will be plotted with the confidence rate and its corresponding predicted text as shown in Fig 25, confidence histogram is also provided to analyze the result better as shown in Fig 26.

CV_PROJ > CZ4003_CV_PROJ > test > output > result > dataset.txt

```
Arial-11.JPG-test Distance:0 Score:100.0
```

Fig 24. A text file that store metrics result

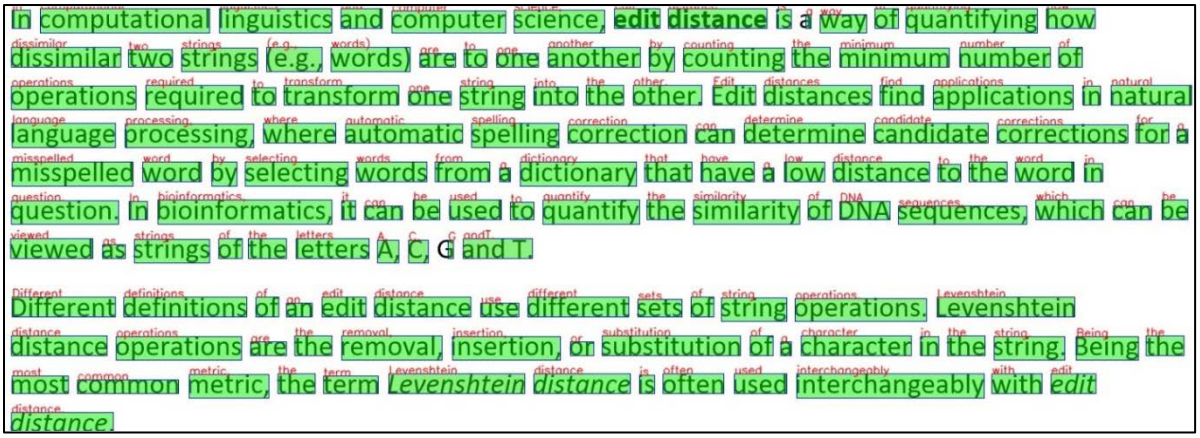


Fig 25. The plotted image consists of the detected bounding boxes, the predicted words, and the confidence rate of each word of the Calibri-11.JPG

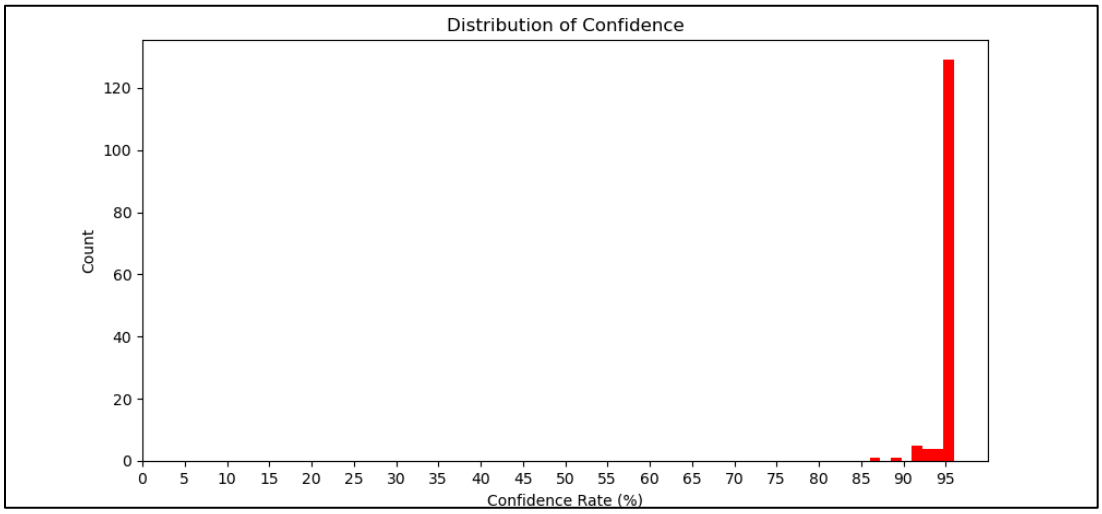


Fig 26. A histogram showed the confidence rate distribution for the Calibri-11.JPG image

Result

S/N	Image Filename	Distance	Score
1	Arial-7.JPG	0	100.0000
2	Arial-9.JPG	0	100.0000
3	Arial-11.JPG	0	100.0000
4	Calibri-11.JPG	**1**	**99.8921**
5	TimesNewRoman-11.JPG	0	100.0000

Fig 27. test OCR accuracy result table.

Discussion

As shown in Fig 27, the score is **mostly 100**, this has proven that the OCR algorithms are accurate as claimed by tesseract 99%. This also means that the OCR algorithm can tolerate a certain degree of text size and font styles being used.

However, there is 1 distance value in the Calibri-11.JPG image. Upon further analysis, as shown in Fig 25, at the 2 words of paragraph 1, the text 'and T' are classified as 'andT' by the OCR algorithm. Thus, **expecting some degree of error** even if it appears to be ideal from our eye.

As for **confidence rate** wise, most words have high confidence of around 90 percentile and above, the lowest was around 87% as shown in Fig 26. It may tell a cue that it is **relative** to its **score** as compared to other image scores.

This experiment test has enforced the **reliability** of the OCR algorithm to be tested for the project.

Firstly, I will be testing the base with Otsu algorithm implementation follow by my enhanced algorithm with the given 2 sample images. I will be evaluating the results and Otsu algorithm limitations.

Otsu Algorithms (Global Threshold)

Overview

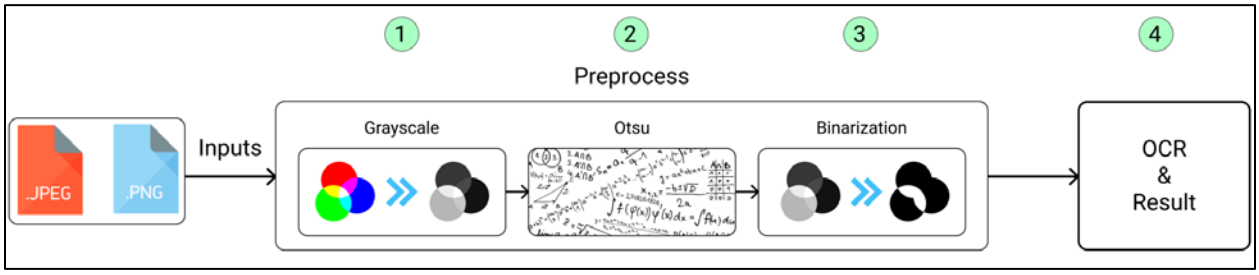


Fig 28. The implementation flow of Otsu algorithm

1. Firstly, the input images must be converted to grayscale as the Otsu algorithm only consider 1 channel for binarization. Fig 29 below is the code that responsible for such operation.

```
image = grayscale(image)
```

```
image_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

Fig 29. The grayscale conversion function called (top), the conversion code (bottom)

2. The Otsu algorithm will be run via a function called 'otsu(image)' where the image argument is in grayscale form as shown in Fig 30 below, the explanation of the algorithm are described in the figure. Otsu algorithm uses global for thresholding, global refers to the entirety of the image to find the maximum variance value.

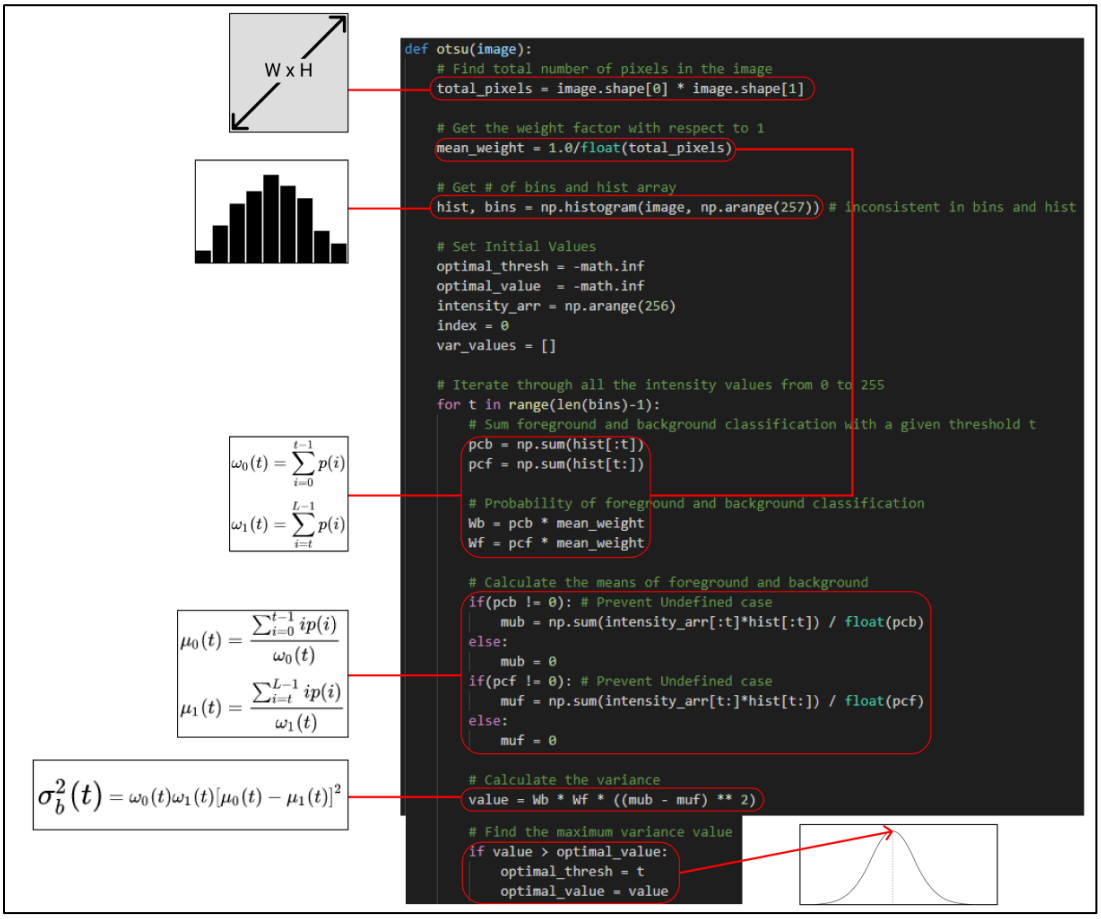


Fig 30. Otsu Implemented Function Script

3. After finding the maximum value of variance, this value will act as the divider for the binarization process, where less than or equal to the divider value will be set to value 0 (foreground) while bigger than the divider value will be set to value 255 (background) as shown in Fig 31.

```
otsu_image[image > optimal_thresh] = 255
otsu_image[image <= optimal_thresh] = 0
```

Fig 31. Binarization Script

4. Lastly, this will be feed into OCR algorithms and produce the respective results, the flow of the implementation is the same as Fig 20 without the input block.

Process

Sample 01

[1] Conversion to Grayscale

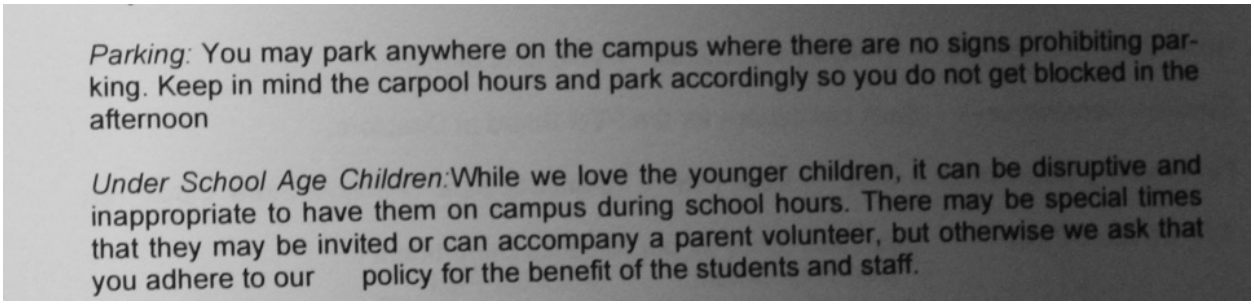


Fig 32. Grayscale image of sample01.png

```
Before Convert to Grayscale Image Dimension =  (229, 965, 3)
After  Convert to Grayscale Image Dimension =  (229, 965)
```

Fig 33. Reduce the number of channel from 3 to 1 for sample01.png

[2] Global Thresholding (Otsu)

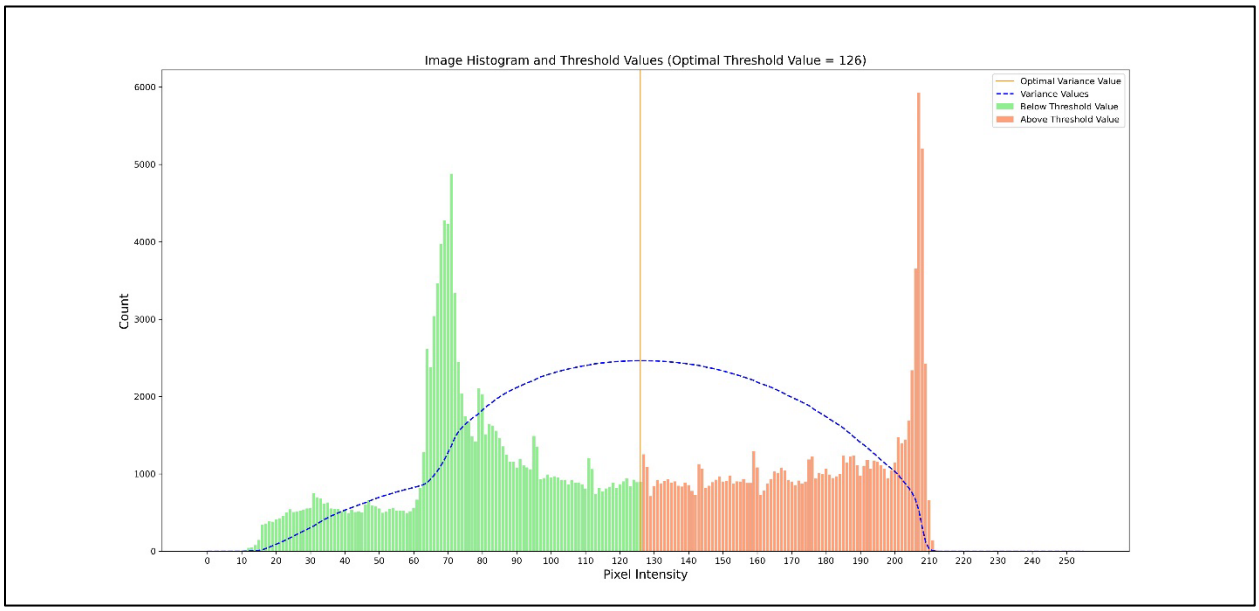


Fig 34. Histogram and global threshold graph of sample01.png

[3] Binarization

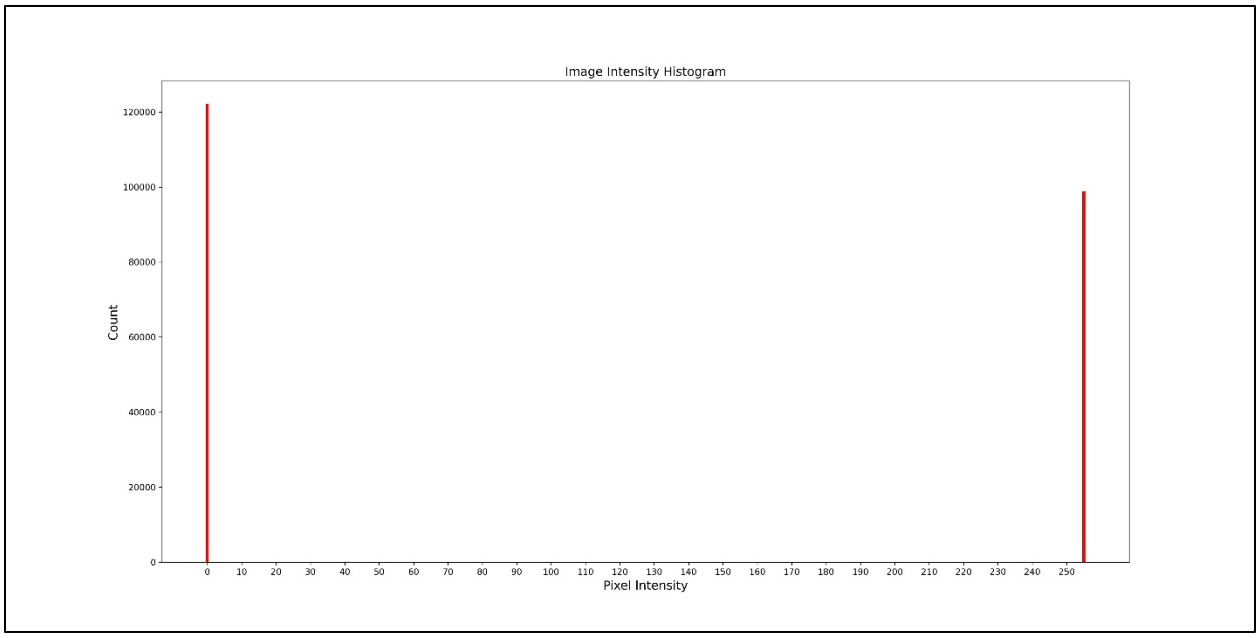


Fig 35. Histogram of image intensity of sample01 after binarization.png

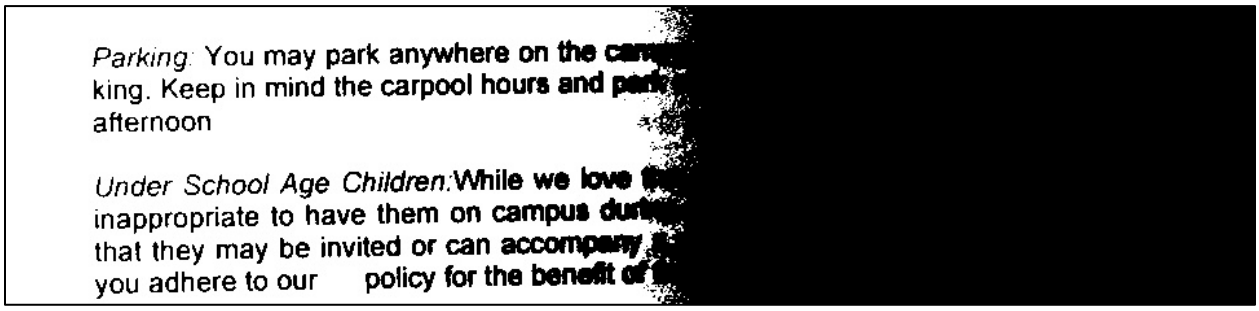


Fig 36. Binarization image of sample01.png

Sample 02

[1] Convert to grayscale, similar to sample01.png

[2] Global Thresholding (Otsu)

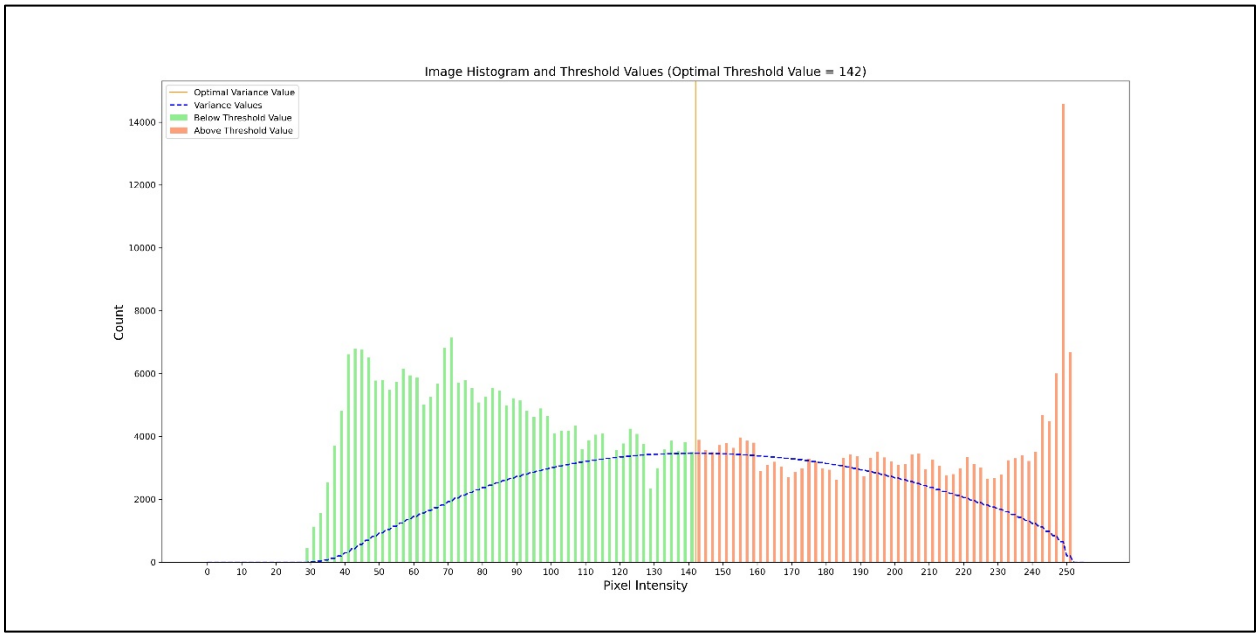


Fig 37. Histogram and global threshold graph of sample02.png

[3] Binarization

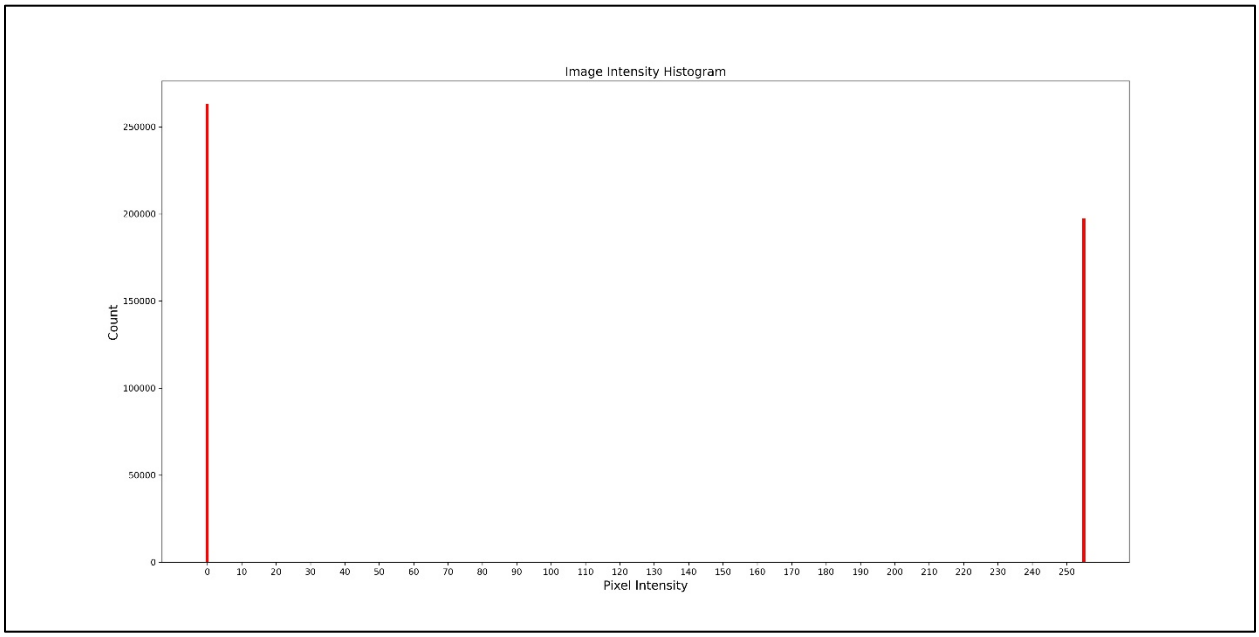


Fig 38. Histogram of image intensity of sample02 after binarization.png

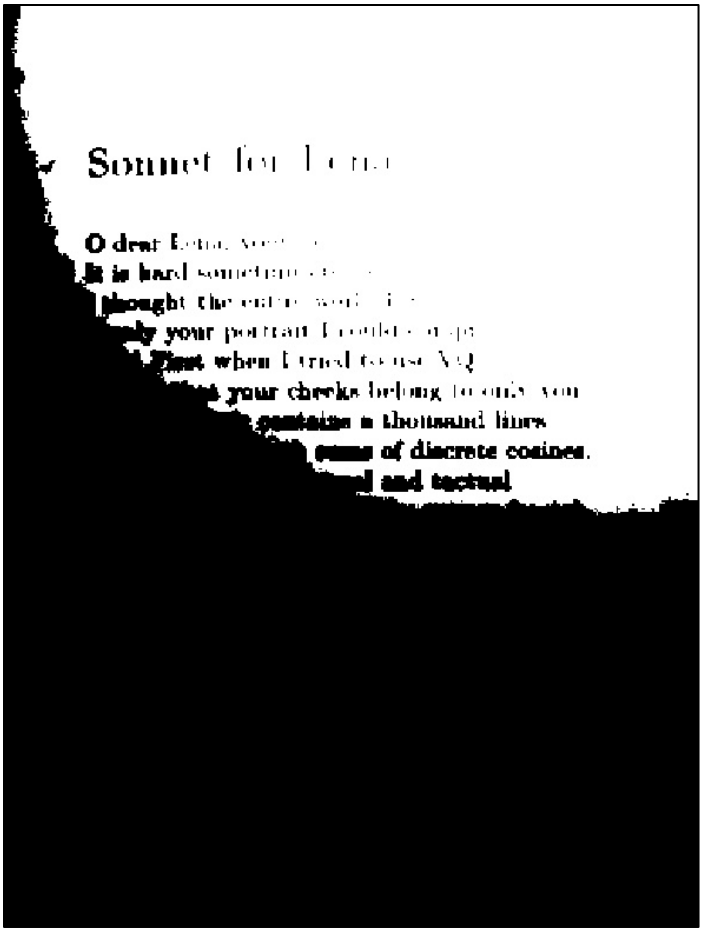


Fig 39. Binarization image of sample01.png

Result

Sample 01

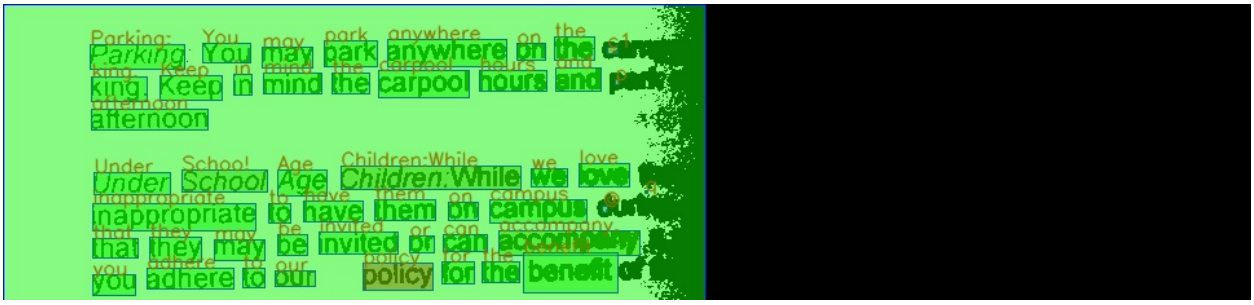


Fig 40. Detected words result of sample01.png

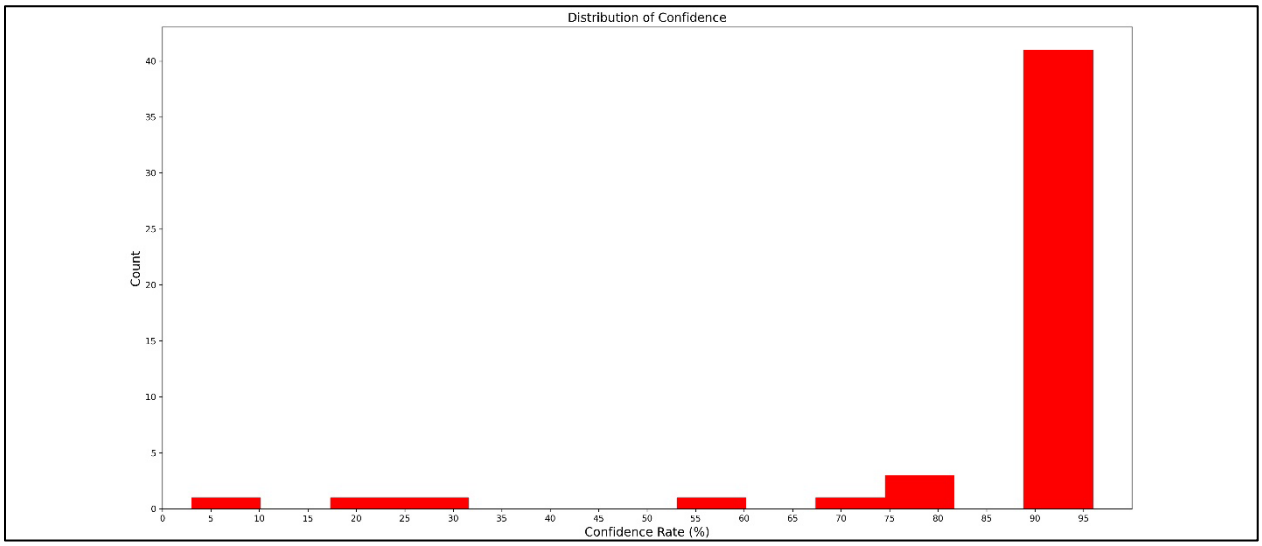


Fig 41. Confidence rate histogram of sample01.png

Sample 02

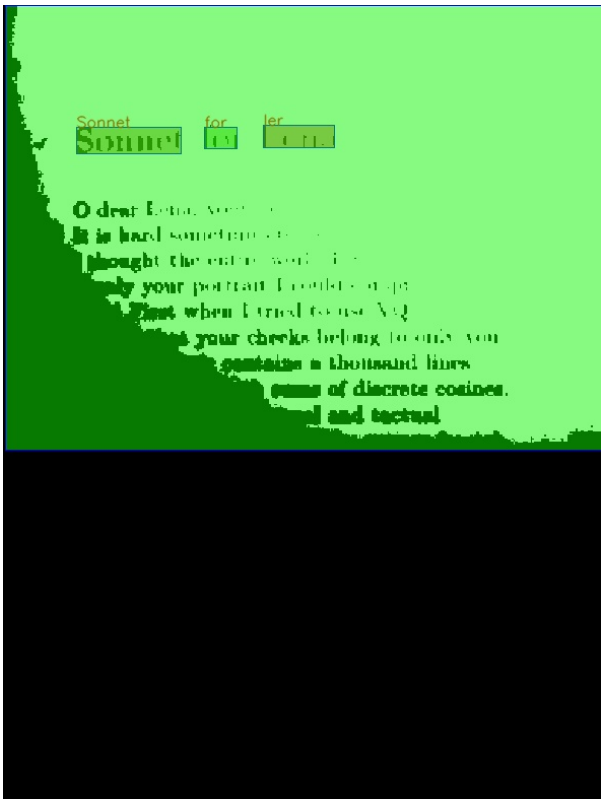


Fig 42. Detected words result of sample02.png

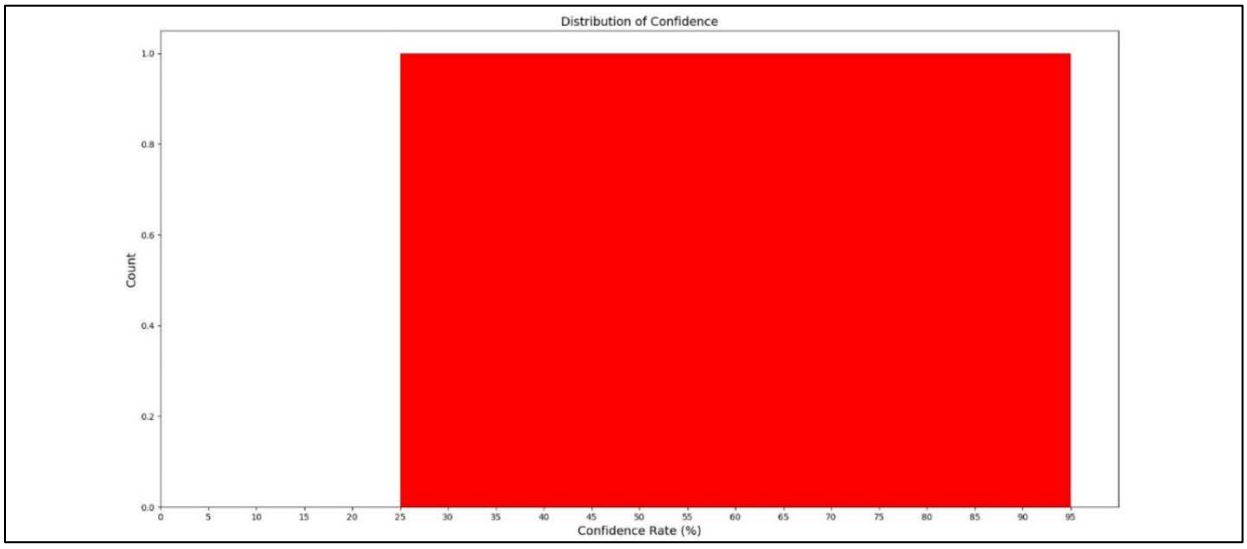


Fig 43. Confidence rate histogram of sample02.png

S/N	Algorithms	Parameters Value	Sample 1		Sample 2	
			Distance	Score	Distance	Score
1	Otsu	NA	261	49.2218	621	2.2047

*Legend: **NA** (Not Applicable)*

Fig 44. A table evaluation after the Otsu algorithm

Discussion

Sample 1 vs Sample 2

- As expected, as seen in Fig 44, sample 2 perform worst than sample 1 since its text quality is poor. It is only able to detect 3 words from the title as most words are not beyond recognizable even with our naked eye.
- If sample 1 omits the right half of the image, it manages to detect most of the words correctly. This shows quality makes a huge difference when detecting text.
- The histogram between Fig 34 and Fig 37 tells us that sample 01 has 2 distinctive peaks of the distribution while sample 02 the 2 peak models are subtle.
- The respective global threshold for sample 1 and sample 2 is 126 and 142 respectively.

Common Issue

- Both images suffer from gradient lighting (non-uniform illumination) background that is closely related to the foreground (text). The reason is that the darker background added weights that result in shifting the global threshold which covers key features of the image such as text located near the dark zone and bright zone where the differences between background and foreground are not significant.
- There is also unwanted impulse noise present in between the contrast in shading, Fig 45 describes this description. This may explain both images have huge bounding boxes detected unintentionally as shown in Fig 40 and 42.

Fig 45 portion of noise in sample01.png (left), in sample02.png(right)

Limitation of Otsu

- It is ideal only when the **contrast** between foreground and background is **distinctive**, which means the 2 peaks distribution must be outstanding and steep.
- **Exclusively** use for **2 class of segment**, where there exist bimodal only in the histogram of the image. It will perform poorly if there are more than 2 classification segments
- It performs badly when the foreground **feature** (e.g text) is in a **small** portion and **poor** in terms of its **quality**.
- It may be **noises sensitive**.

Enhance Algorithm

Consideration

Finding optimal parameters

Grid Search Technique

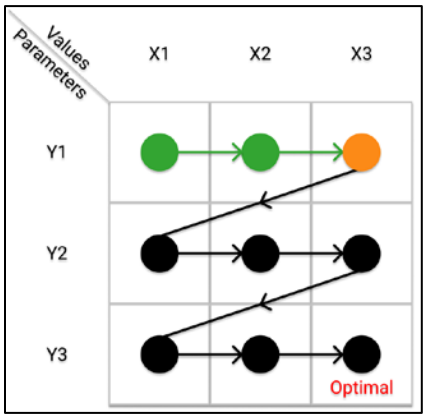


Fig 46. Visual representation of the Grid Search Technique

- Definition
    - This technique tries every possible combination for the given sets of parameter values. The pattern illustrated in Fig 46 shows how it sequentially performs the search, where each set of parameters is considered before moving to the next one. Once all combinations are evaluated, the set of parameters that yield the highest accuracy is the best. In our context is referring to 'distance' and 'score'.
- Why choose this?
    - Base on the previous 2 LAB assignments experiment, it shows that it is a **tedious** and exhaustive process in looking for the right parameter to enhance the image. Besides that, based on the Otsu problems there may be a need for multiple parameters to be considered. Thus, looking for the optimal set of a parameter value in **multi-dimension** is **not feasible** to do it **manually**. However, it may lead to an exponential increase in runtime issues with each additional parameter. Since time efficiency was not part of this assessment so it does not matter. In our context, the minimum consideration was about 3 different values per set of X number of parameters. These are the combinations as shown in Fig 47.

S/N	Number of parameters	Number of the set of value used	Number of combinations
1	3	3	27
2	4	3	64
3	3	4	81

Fig 47. A table illustrates the exponential increase in combination with an increase in terms of parameters and the set of values

Best representing ground truth

With **numerous combinations**, it is no longer feasible to display ground truth with images. Therefore, **images** will **not be shown** in this report, the explanation will mostly **focus on 'distance' and 'score'** and possibility some highlights of the images. All images can be review from the source code zip file provided.

Overview

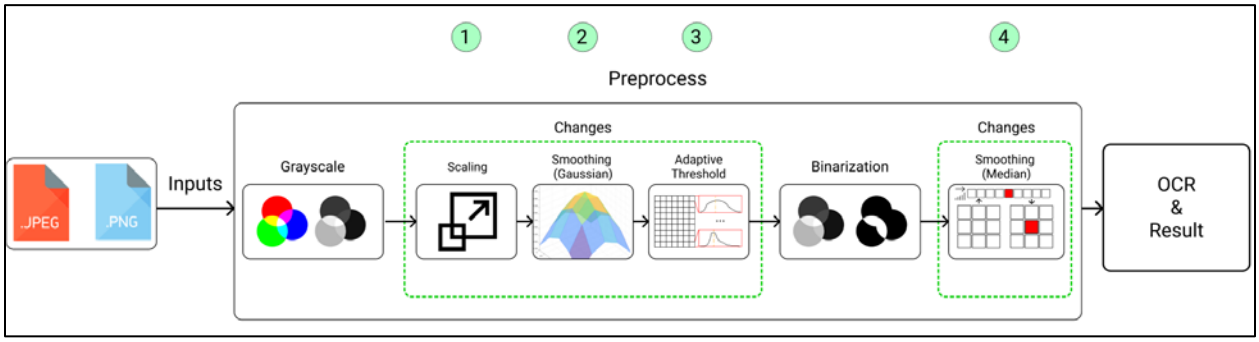


Fig 48. The implementation flow of the enhanced algorithm

1. **Scaling** the images to recognize the characters better as they can '**fit more pixels** into the character' such that the OCR algorithms have a higher probability of interpreting it correctly as illustrated in Fig 49 below. The scale should increase in terms of the whole number to prevent feature loss during the magnification or reduction process.

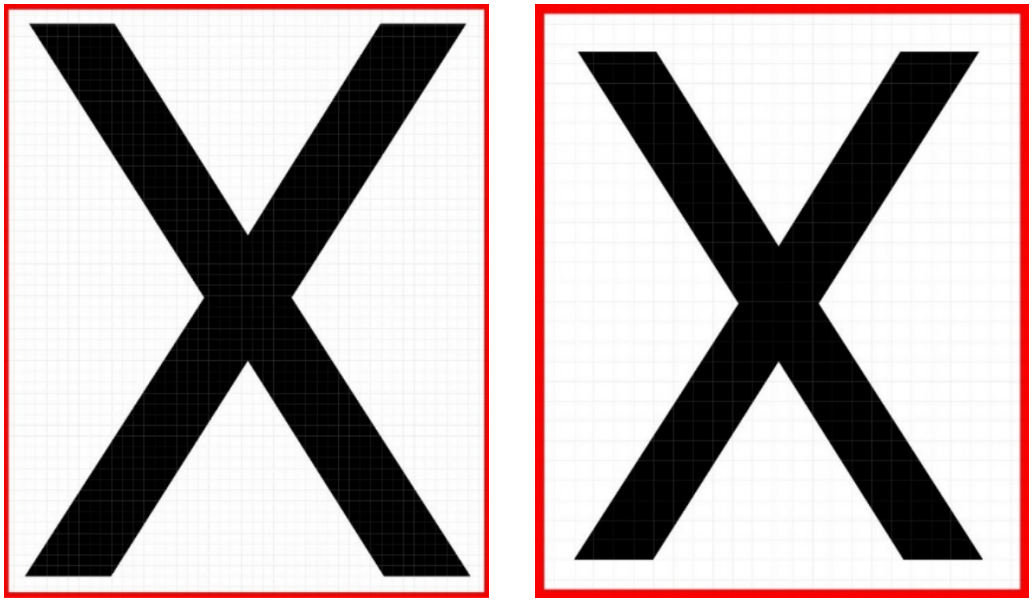


Fig 49. Double scale (left) and normal scale (right) of 'X' character.

2. Using a gaussian filter to smoothen the image to **remove** the gentle **noise** present in the image as shown in Fig 50 and 51. The 2 figures below are reproduced using Lab 1 of my assignment. This issue appeared in both sample 1 and sample 2 images

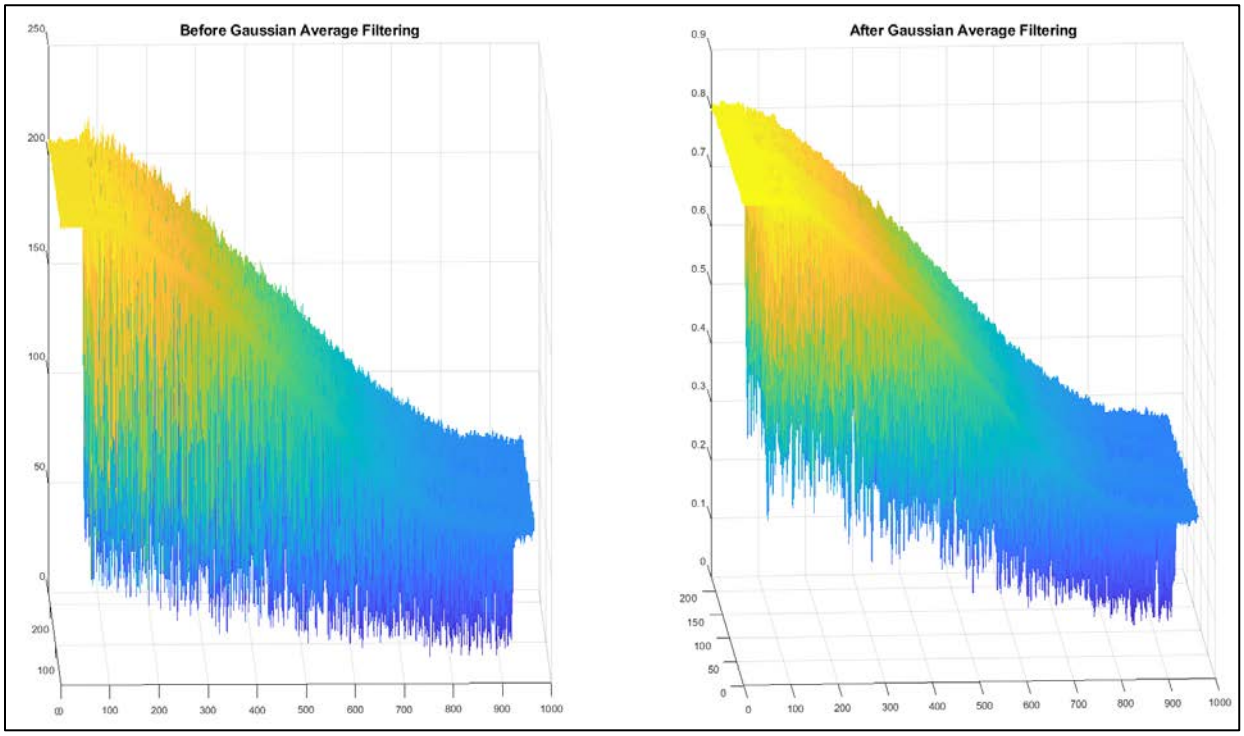


Fig 50. The 3D pixel intensity of sample01.png before and after the Gaussian filter.

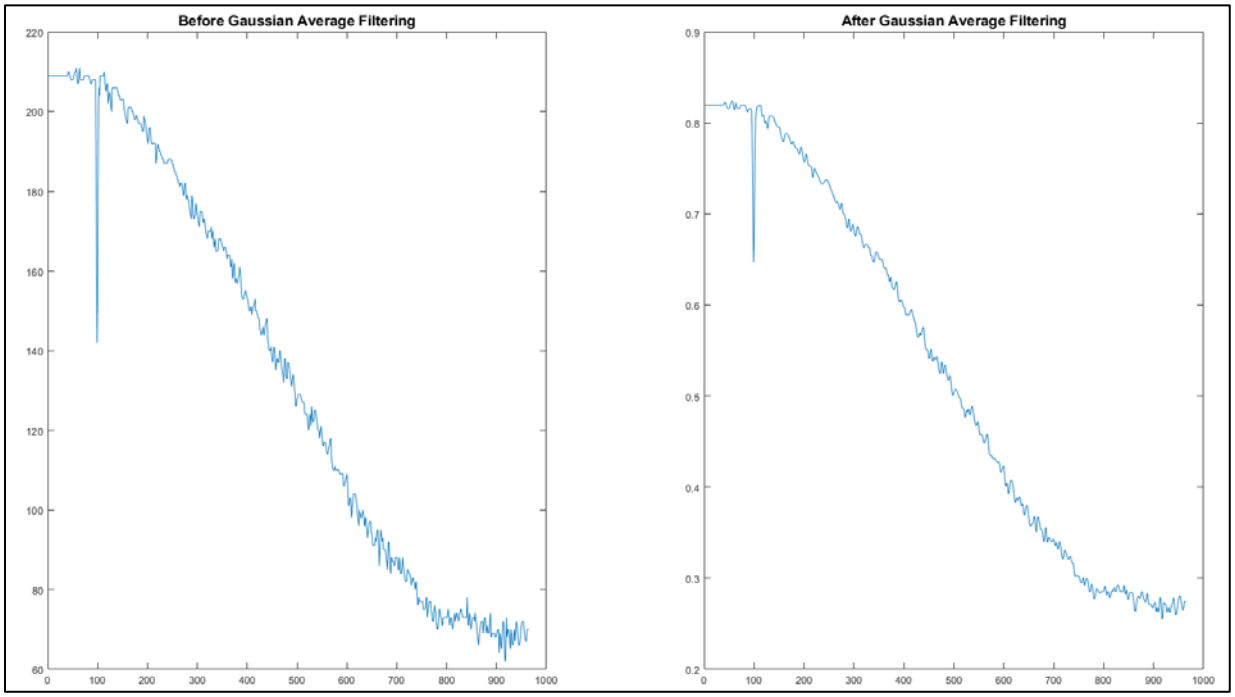


Fig 51. First-row pixel intensity of sample01.png before and after the Gaussian filter.

3. Using an **adaptive threshold** helps overcome the issue of Otsu where it does not rely on global threshold instead it uses **local threshold** by breaking the **image into multiple segments** or block sizes and compute the optimal threshold respectively. The segmented image is binarized to separate between foreground and background according to individual maximum threshold values. After which the segment images are **merged back into one single image**. This is illustrated as shown in Fig 52 below.

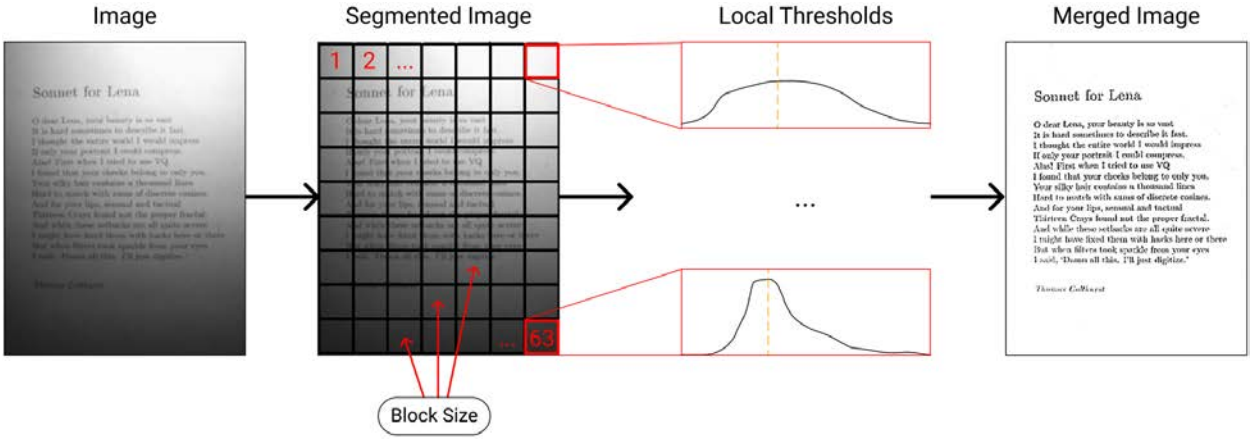


Fig 52. Adaptive thresholding and binarization process of sample 2 image

4. After binarization, all the **impulse noises** present in the image are **suppressed** using a **median filter** as shown in Fig 53. This will result in clean images. This issue appeared in both sample 1 and sample 2 images. However, after the primary trial and error phase, the **OCR accuracies** are **unstable** with high accuracies (85% to 99%) values but not a perfect score of 100, even though the image appears to be clear with our naked eye. Which leads to the motivation of the grid search technique.

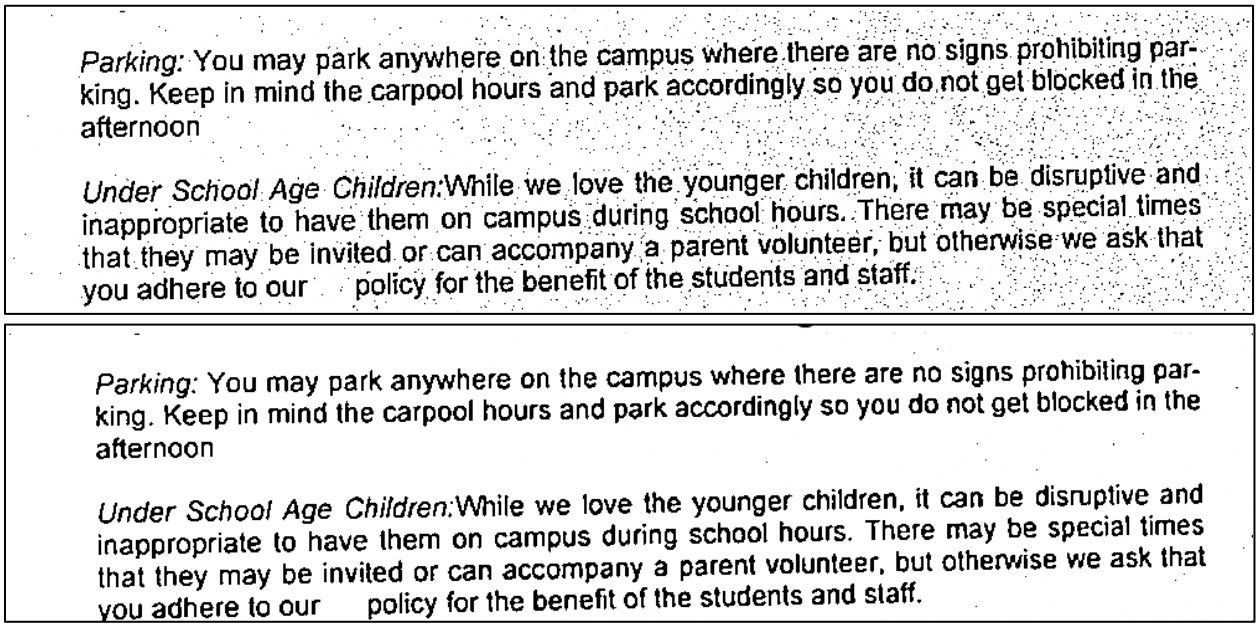


Fig 53. Before (top) and after (bottom) median filtering of sample 1 image

Parameters Used

After primary trial and error with different range of values range, below are the set of values, best suited for our context with the least number of possible combinations. In this case, we are computing **162 combinations** per sample. Thus, **324** iterations in total.

S/N	Parameter	Values		
1	Scaling	200%	250%	300%
2	Gaussian Blur Kernel Size	(5,5)	(7,7)	(11,11)
3	Adaptive Threshold's Block Size	(11,11)	(17,17)	(37,37)
4	Adaptive Threshold's Constant	2	3	
5	Median Blur Kernel Size	(1,1) (Off)	(5,5)	(7,7)

The adaptive threshold's constant argument subtracts the mean or weighted mean.

Fig 54. Set of parameter values table

Result

S/N	Algorithm	Parameters					Sample 1		Sample 2	
		SC	GB	AB	AC	MB	Distance	Score	Distance	Score
1	Enhanced	200	05	11	2	1	74	85.6311	87	86.1685
2		200	05	11	2	5	3	99.4175	107	82.9889
3		**200**	**05**	**11**	**2**	**7**	**0**	**100.0**	202	67.8855
4		200	05	11	3	1	5	99.0291	128	79.6502
5		**200**	**05**	**11**	**3**	**5**	**0**	**100.0**	235	62.6391
6		200	05	11	3	7	4	99.2233	396	37.0429
7		200	05	17	2	1	515	0.0	72	88.5533
8		200	05	17	2	5	35	93.2039	55	91.256
9		200	05	17	2	7	11	97.8641	85	86.4865
10		200	05	17	3	1	38	92.6214	58	90.779
11		200	05	17	3	5	3	99.4175	77	87.7583
12		200	05	17	3	7	3	99.4175	185	70.5882
13		200	05	37	2	1	69	86.6019	312	50.3975
14		200	05	37	2	5	284	44.8544	84	86.6455
15		200	05	37	2	7	145	71.8447	74	88.2353
16		200	05	37	3	1	515	0.0	35	94.4356
17		200	05	37	3	5	38	92.6214	52	91.7329
18		200	05	37	3	7	17	96.699	92	85.3736
19		200	07	11	2	1	34	93.3981	70	88.8712
20		**200**	**07**	**11**	**2**	**5**	**0**	**100.0**	95	84.8967
21		200	07	11	2	7	4	99.2233	201	68.0445
22		200	07	11	3	1	4	99.2233	132	79.0143
23		200	07	11	3	5	3	99.4175	262	58.3466
24		200	07	11	3	7	4	99.2233	361	42.6073
25		200	07	17	2	1	54	89.5146	42	93.3227
26		200	07	17	2	5	32	93.7864	43	93.1638
27		200	07	17	2	7	9	98.2524	85	86.4865
28		200	07	17	3	1	12	97.6699	47	92.5278
29		200	07	17	3	5	3	99.4175	81	87.1224
30		200	07	17	3	7	2	99.6117	164	73.9269
31		200	07	37	2	1	321	37.6699	141	77.5835
32		200	07	37	2	5	116	77.4757	74	88.2353
33		200	07	37	2	7	109	78.835	77	87.7583
34		200	07	37	3	1	42	91.8447	24	96.1844
35		200	07	37	3	5	14	97.2816	47	92.5278

36		200	07	37	3	7	10	98.0583	89	85.8506
37		200	11	11	2	1	2	99.6117	62	90.1431
38		200	11	11	2	5	3	99.4175	129	79.4913
39		200	11	11	2	7	2	99.6117	262	58.3466
40		200	11	11	3	1	2	99.6117	177	71.8601
41		200	11	11	3	5	3	99.4175	312	50.3975
42		200	11	11	3	7	37	92.8155	435	30.8426
43		200	11	17	2	1	39	92.4272	28	95.5485
44		200	11	17	2	5	11	97.8641	52	91.7329
45		200	11	17	2	7	3	99.4175	93	85.2146
46		200	11	17	3	1	3	99.4175	56	91.097
47		**200**	**11**	**17**	**3**	**5**	**0**	**100.0**	104	83.4658
48		200	11	17	3	7	4	99.2233	213	66.1367
49		200	11	37	2	1	284	44.8544	91	85.5326
50		200	11	37	2	5	142	72.4272	64	89.8251
51		200	11	37	2	7	65	87.3786	93	85.2146
52		200	11	37	3	1	22	95.7282	43	93.1638
53		200	11	37	3	5	16	96.8932	74	88.2353
54		200	11	37	3	7	10	98.0583	109	82.6709
55		250	05	11	2	1	60	88.3495	236	62.4801
56		250	05	11	2	5	2	99.6117	179	71.5421
57		250	05	11	2	7	2	99.6117	211	66.4547
58		250	05	11	3	1	10	98.0583	299	52.4642
59		250	05	11	3	5	2	99.6117	345	45.151
60		250	05	11	3	7	3	99.4175	430	31.6375
61		250	05	17	2	1	515	0.0	110	82.5119
62		250	05	17	2	5	32	93.7864	63	89.9841
63		250	05	17	2	7	15	97.0874	63	89.9841
64	Enhanced	250	05	17	3	1	38	92.6214	90	85.6916
65		250	05	17	3	5	3	99.4175	127	79.8092
66		250	05	17	3	7	2	99.6117	161	74.4038
67		250	05	37	2	1	98	80.9709	188	70.1113
68		250	05	37	2	5	261	49.3204	70	88.8712
69		250	05	37	2	7	108	79.0291	54	91.4149
70		250	05	37	3	1	176	65.8252	35	94.4356
71		250	05	37	3	5	40	92.233	41	93.4817
72		250	05	37	3	7	25	95.1456	54	91.4149
73		250	07	11	2	1	33	93.5922	117	81.399
74		250	07	11	2	5	2	99.6117	169	73.132
75		250	07	11	2	7	2	99.6117	231	63.275
76		250	07	11	3	1	1	99.8058	280	55.4849
77		250	07	11	3	5	2	99.6117	328	47.8537
78		250	07	11	3	7	5	99.0291	428	31.9555
79		250	07	17	2	1	40	92.233	59	90.62
80		250	07	17	2	5	24	95.3398	49	92.2099
81		250	07	17	2	7	10	98.0583	69	89.0302
82		250	07	17	3	1	16	96.8932	83	86.8045
83		250	07	17	3	5	2	99.6117	127	79.8092
84		**250**	**07**	**17**	**3**	**7**	**0**	**100.0**	175	72.1781
85		250	07	37	2	1	265	48.5437	146	76.7886
86		250	07	37	2	5	146	71.6505	59	90.62
87		250	07	37	2	7	94	81.7476	61	90.3021
88		250	07	37	3	1	56	89.1262	30	95.2305
89		250	07	37	3	5	20	96.1165	41	93.4817
90		250	07	37	3	7	17	96.699	60	90.461
91		250	11	11	2	1	4	99.2233	112	82.194

92		250	11	11	2	5	2	99.6117	155	75.3577
93		**250**	**11**	**11**	**2**	**7**	**0**	**100.0**	275	56.2798
94		250	11	11	3	1	2	99.6117	311	50.5564
95		250	11	11	3	5	3	99.4175	396	37.0429
96		250	11	11	3	7	18	96.5049	442	29.7297
97		250	11	17	2	1	28	94.5631	36	94.2766
98		250	11	17	2	5	9	98.2524	59	90.62
99		250	11	17	2	7	4	99.2233	88	86.0095
100		250	11	17	3	1	3	99.4175	86	86.3275
101		250	11	17	3	5	2	99.6117	133	78.8553
102		**250**	**11**	**17**	**3**	**7**	**0**	**100.0**	229	63.593
103		250	11	37	2	1	252	51.068	77	87.7583
104		250	11	37	2	5	384	25.4369	53	91.5739
105		250	11	37	2	7	80	84.466	67	89.3482
106		250	11	37	3	1	28	94.5631	22	96.5024
107		250	11	37	3	5	19	96.3107	49	92.2099
108		250	11	37	3	7	11	97.8641	74	88.2353
109		300	05	11	2	1	20	96.1165	416	33.8633
110		300	05	11	2	5	8	98.4466	325	48.3307
111		300	05	11	2	7	2	99.6117	325	48.3307
112		300	05	11	3	1	19	96.3107	470	25.2782
113		300	05	11	3	5	5	99.0291	452	28.1399
114		300	05	11	3	7	11	97.8641	439	30.2067
115		300	05	17	2	1	258	49.9029	162	74.2448
116		300	05	17	2	5	42	91.8447	112	82.194
117		300	05	17	2	7	30	94.1748	104	83.4658
118		300	05	17	3	1	36	93.0097	184	70.7472
119		300	05	17	3	5	9	98.2524	185	70.5882
120	Enhanced	300	05	17	3	7	2	99.6117	300	52.3052
121		300	05	37	2	1	131	74.5631	264	58.0286
122		300	05	37	2	5	229	55.534	91	85.5326
123		300	05	37	2	7	363	29.5146	62	90.1431
124		300	05	37	3	1	91	82.3301	56	91.097
125		300	05	37	3	5	40	92.233	45	92.8458
126		300	05	37	3	7	26	94.9515	47	92.5278
127		300	07	11	2	1	26	94.9515	224	64.3879
128		300	07	11	2	5	6	98.835	258	58.9825
129		300	07	11	2	7	4	99.2233	301	52.1463
130		300	07	11	3	1	2	99.6117	420	33.2273
131		300	07	11	3	5	3	99.4175	462	26.5501
132		300	07	11	3	7	9	98.2524	454	27.8219
133		300	07	17	2	1	48	90.6796	134	78.6963
134		300	07	17	2	5	32	93.7864	98	84.4197
135		300	07	17	2	7	15	97.0874	86	86.3275
136		300	07	17	3	1	20	96.1165	148	76.4706
137		300	07	17	3	5	4	99.2233	199	68.3625
138		300	07	17	3	7	3	99.4175	253	59.7774
139		300	07	37	2	1	128	75.1456	169	73.132
140		300	07	37	2	5	237	53.9806	58	90.779
141		300	07	37	2	7	167	67.5728	59	90.62
142		300	07	37	3	1	88	82.9126	28	95.5485
143		300	07	37	3	5	41	92.0388	35	94.4356
144		300	07	37	3	7	21	95.9223	50	92.0509
145		300	11	11	2	1	9	98.2524	219	65.1828
146		300	11	11	2	5	4	99.2233	292	53.5771
147		300	11	11	2	7	3	99.4175	338	46.2639

148		300	11	11	3	1	2	99.6117	432	31.3196
149		300	11	11	3	5	6	98.835	457	27.345
150		300	11	11	3	7	22	95.7282	479	23.8474
151		300	11	17	2	1	35	93.2039	58	90.779
152		300	11	17	2	5	18	96.5049	79	87.4404
153		300	11	17	2	7	3	99.4175	104	83.4658
154		300	11	17	3	1	4	99.2233	129	79.4913
155	Enhanced	300	11	17	3	5	3	99.4175	246	60.8903
156		300	11	17	3	7	1	99.8058	273	56.5978
157		300	11	37	2	1	422	18.0583	86	86.3275
158		300	11	37	2	5	185	64.0777	38	93.9587
159		300	11	37	2	7	140	72.8155	42	93.3227
160		**300**	**11**	**37**	**3**	**1**	24	95.3398	**20**	**96.8203**
161		300	11	37	3	5	14	97.2816	31	95.0715
162		300	11	37	3	7	10	98.0583	43	93.1638

*Legend: **SC** (Scaling), **GB** (Gaussian Blur's Kernel Size), **AB** (Adaptive Threshold Block Size), **AC** (Adaptive Threshold Constant), **MB** (Median Blur's Kernel Size)*

```
Finding Best Score
Saved Best Result
Successfully Saved Image: CZ4003_CV_PROJ/output/result/best/sample02__300-11-37-3-1.jpg
Total Time Taken:  2958.942 s
End of Operation
```

Fig 55. All combination results using the enhanced algorithm and Time Taken of the full operation.

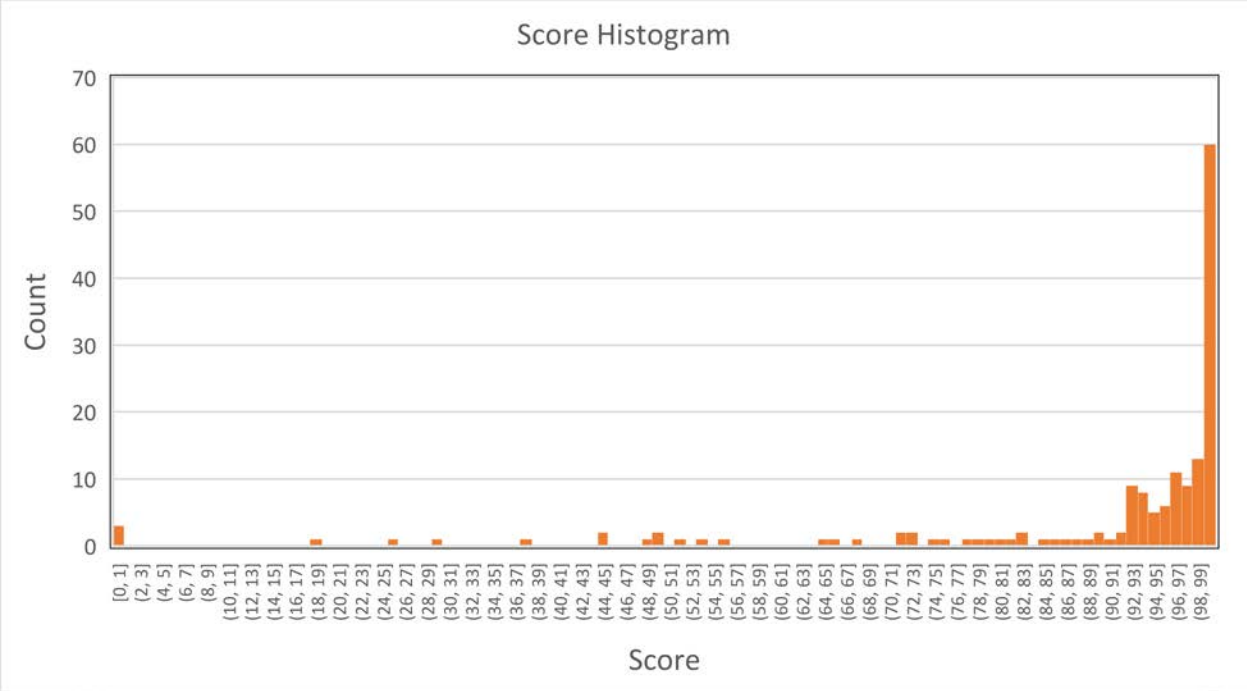


Fig 56. Approximate score distribution of all combinations for sample 1

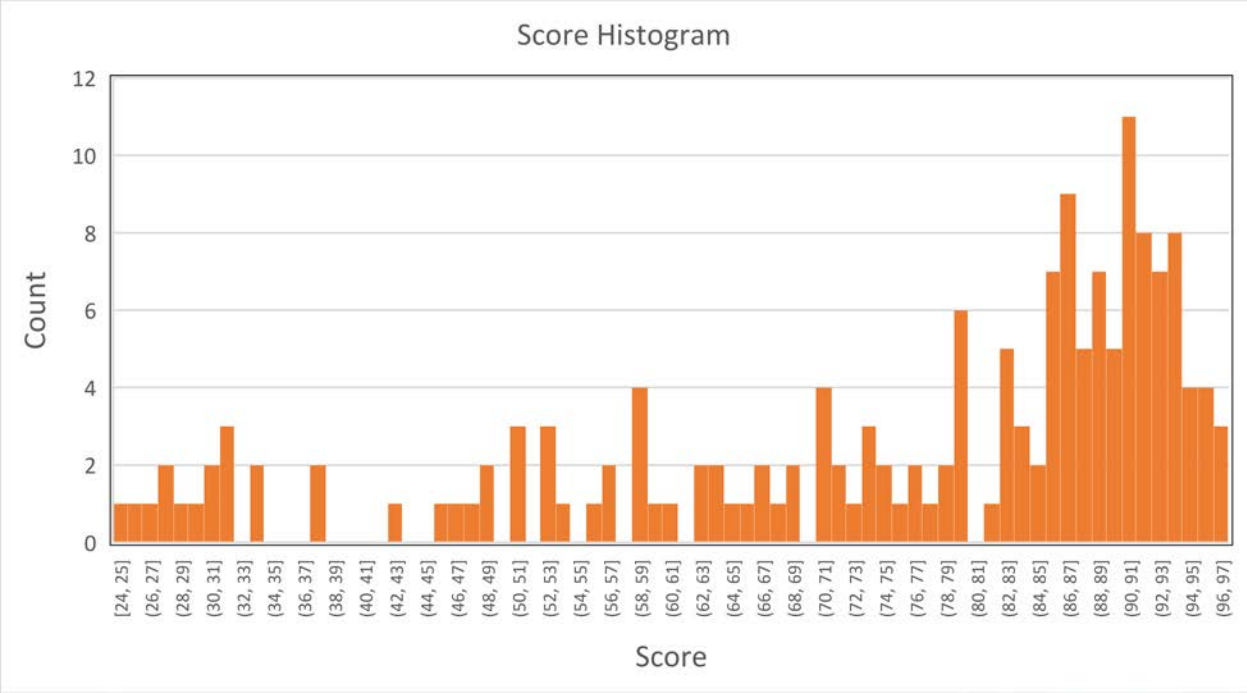


Fig 57. Approximate score distribution of all combinations for sample 2

Optimal OCR Accuracy Achieved

S/N	Parameters					Sample 1	
	SC	GB	AB	AC	MB	Distance	Score
1	200	05	11	2	7	0	100
2	200	05	11	3	5	0	100
3	200	07	11	2	5	0	100
4	200	11	17	3	5	0	100
5	250	07	17	3	7	0	100
6	250	11	11	2	7	0	100
7	250	11	17	3	7	0	100

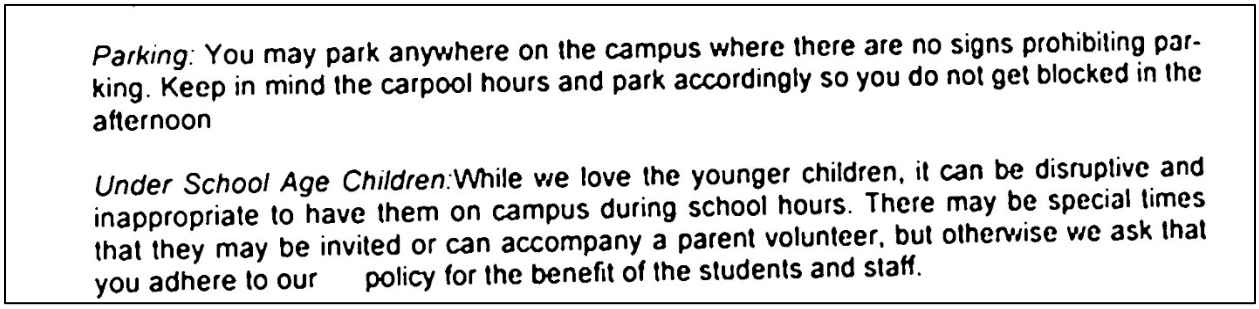
S/N	Parameters					Sample 2	
	SC	GB	AB	AC	MB	Distance	Score
1	300	11	37	3	1	20	96.8203

*Legend: **SC** (Scaling), **GB** (Gaussian Blur's Kernel Size), **AB** (Adaptive Threshold Block Size), **AC** (Adaptive Threshold Constant), **MB** (Median Blur's Kernel Size)*

Fig 58. Best combination results of sample 1 (top) and sample 2 (bottom).

Sample 1 with Optimal OCR Accuracy

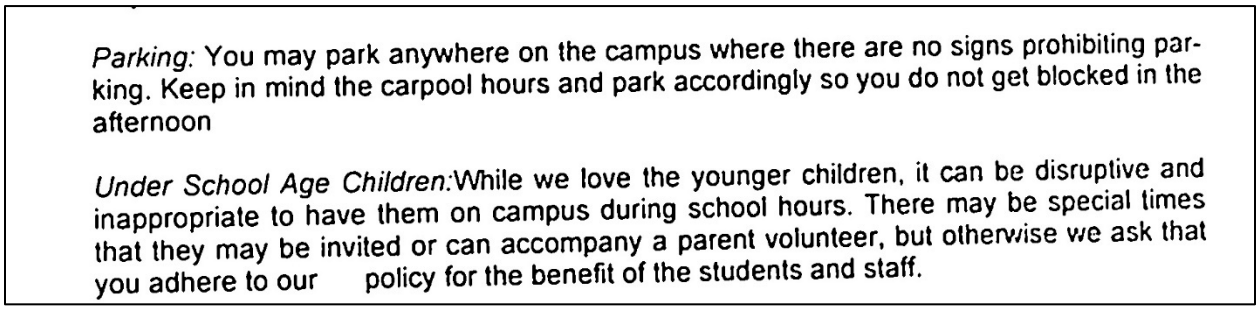
The order is based on Fig 58.



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

Under School Age Children:While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

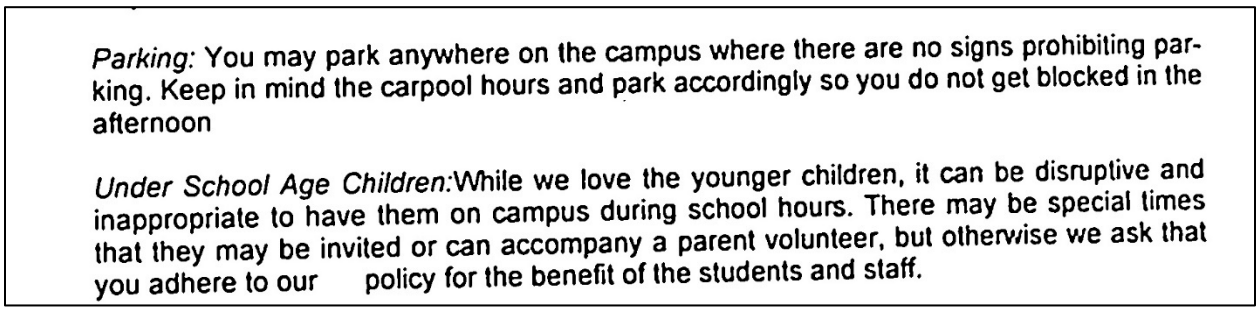
Fig 59. sample01__200-05-11-2-7.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

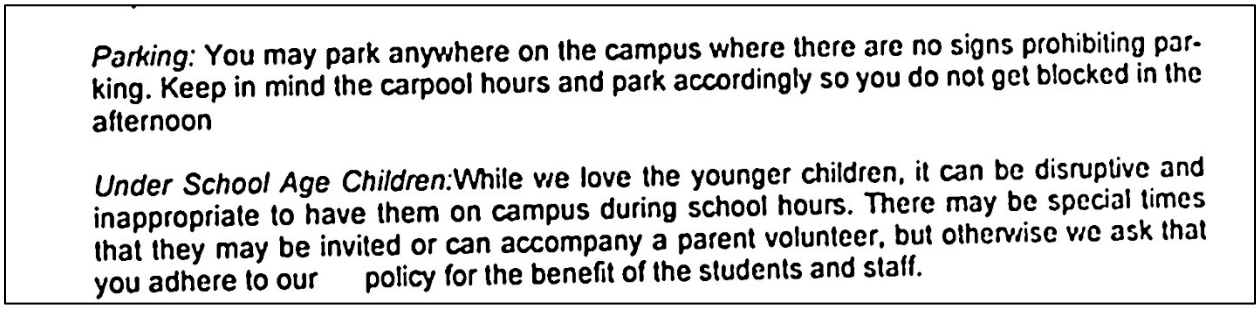
Fig 60. sample01__200-05-11-3-5.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

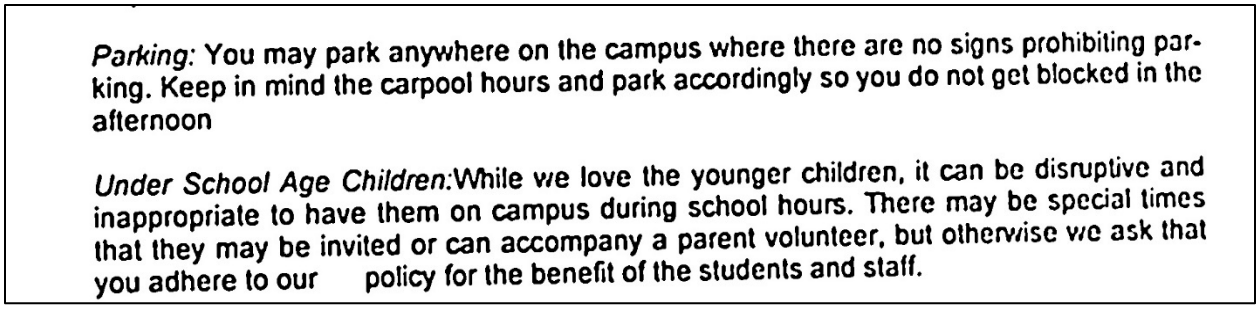
Fig 61. sample01__200-07-11-2-5.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

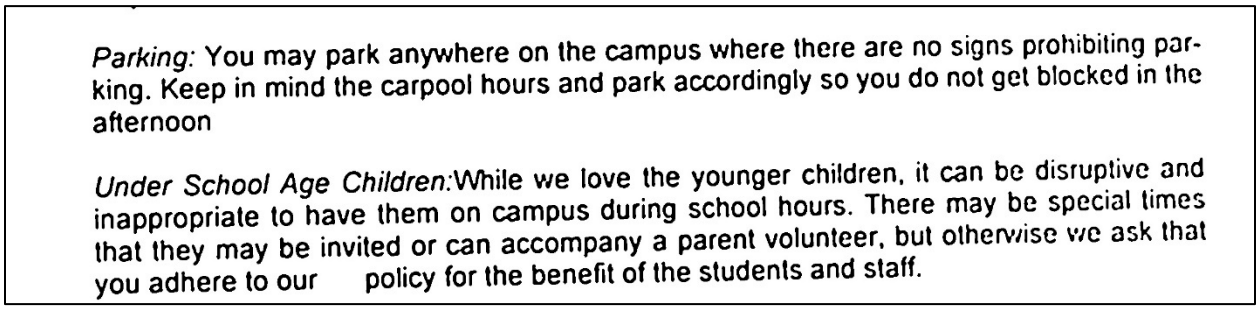
Fig 62. sample01__200-11-17-3-5.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

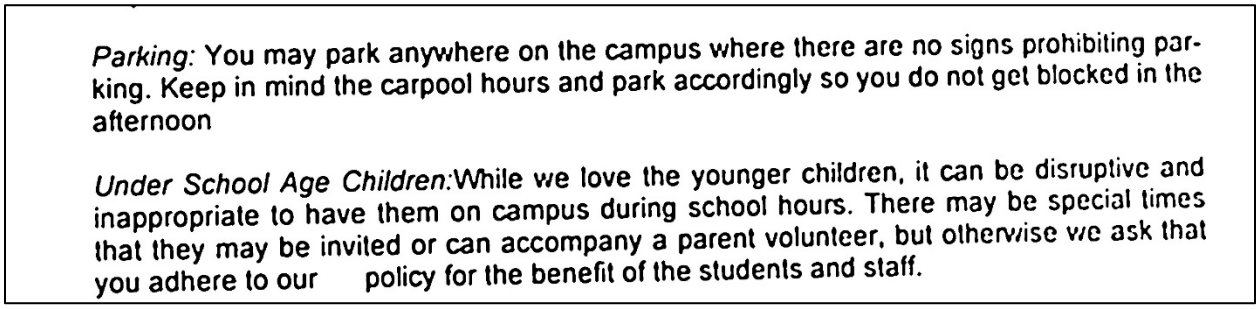
Fig 63. sample01__250-07-17-3-7.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

Fig 64. sample01__250-11-11-2-7.jpg



Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

Fig 65. sample01__250-11-17-3-7.jpg

Sample 2 with Optimal OCR Accuracy

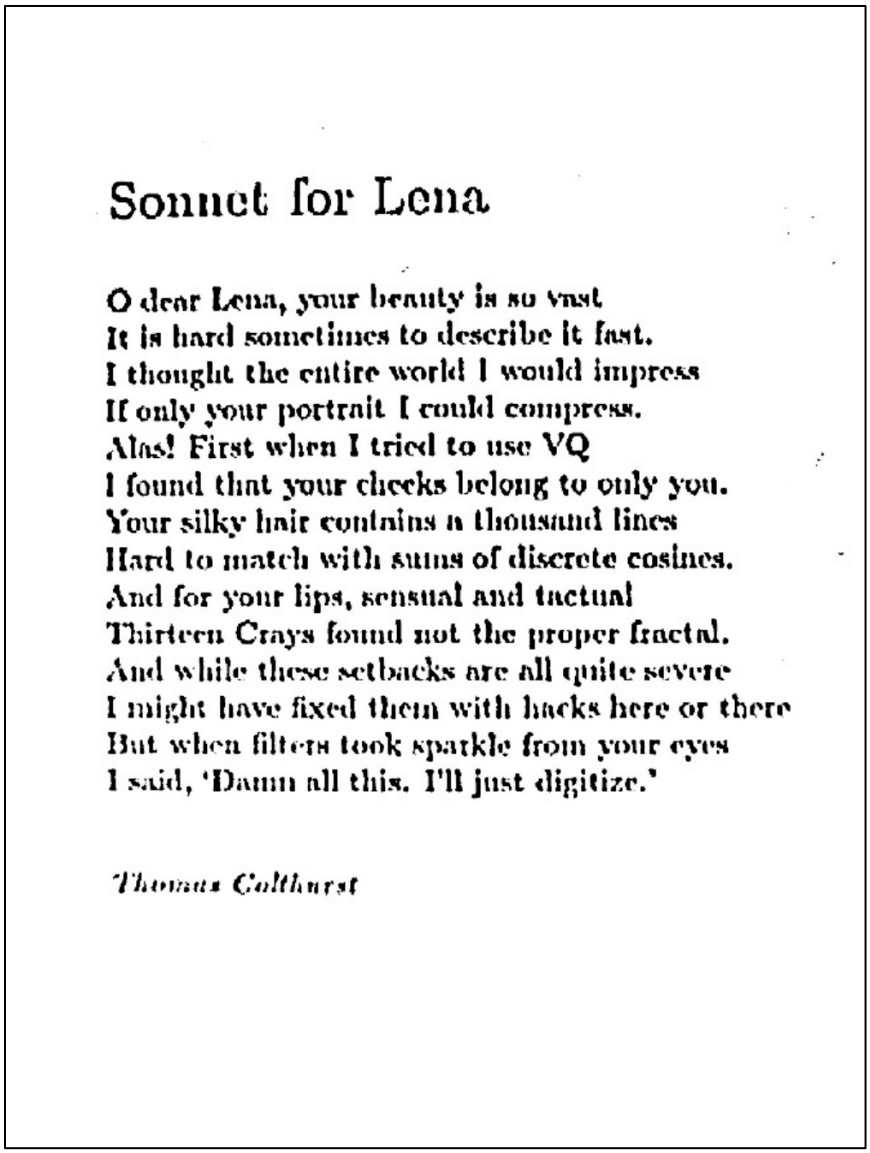


Fig 66. sample02__300-11-37-3-1 without detection plot (top) and with detection plot (bottom).jpg

Discussion

Base on Fig 58, there is a total 7 of 100% accuracy with 0 distance value for sample 1. On the other hand, sample 2 only manage to achieve 96.82% best with 20 distance values. As expected, it shows that sample01 is easier to achieve high accuracy due to its sharper quality of the image as compared to sample 2. This is shown in score histograms in Fig 56 and 57, high distribution of score is allocated in high score bins for sample 1 as compared to sample 2. On top of that, the sample 1 image is clean with little to no noise. Thus, we have achieved optimal OCR accuracy for sample 1.

Besides that, in Fig 55, it can be observed that the optimal parameters for one's image do not mean optimal for other images. It shows the accuracy is very sensitive as each image consist of a different set of intensity values in each pixel which results in such changes.

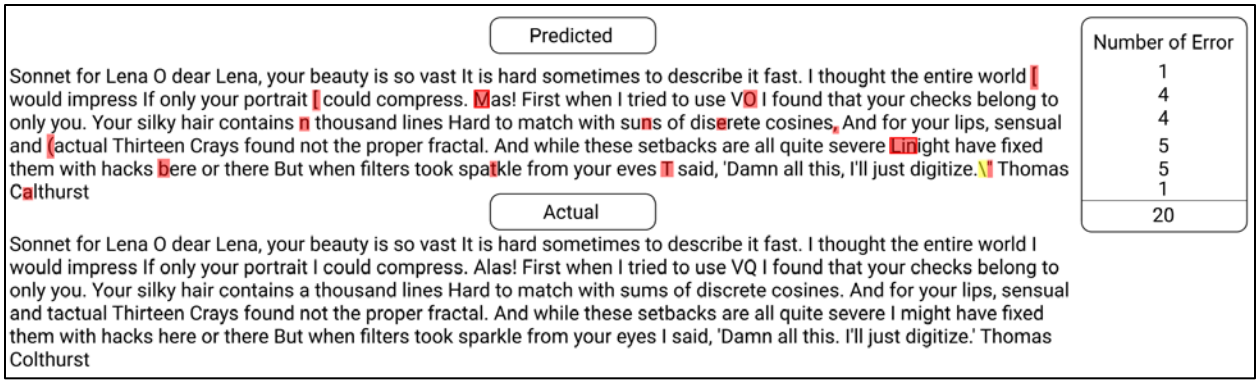


Fig 67. Breakdown of error with optimal sample 2 image result

After closer inspection of the error from sample 2 as shown in Fig 67 which was an extract from the JSON file that stores all the predicted strings. With references with actual output as shown in Fig 66, most of the character key feature is there but the clarity and sharpness were lost.

Next, I tried 2 different add-on methods DPI and Erode to enhance the image for sample 2 to yield higher accuracy. However, it fails to improve the accuracy any further. Therefore, I concluded that Fig 58 is the best representative for OCR accuracy for this project. Fig 68 below is the comparison result between the Otsu algorithm and the enhanced algorithm. It shows tremendous improvement with the local thresholding approach.

S/N	Algorithms	Sample 1		Sample 2	
		Distance	Accuracy	Distance	Accuracy
1	Otsu	261	49.22%	621	2.20%
2	Enhanced	**0**	**100%**	**20**	**96.82%**

Fig 68. Comparison result between Otsu algorithm and Enhanced algorithm

Other experimentations

DPI (Dots per inch)

I attempt to re-run the algorithm with an increase in DPI value to 300, this is to fulfill the recommend image as stated in the OCR tesseract. Unfortunately, as shown in the new histogram score in Fig 69 below. It overall decreases the accuracy of the image.

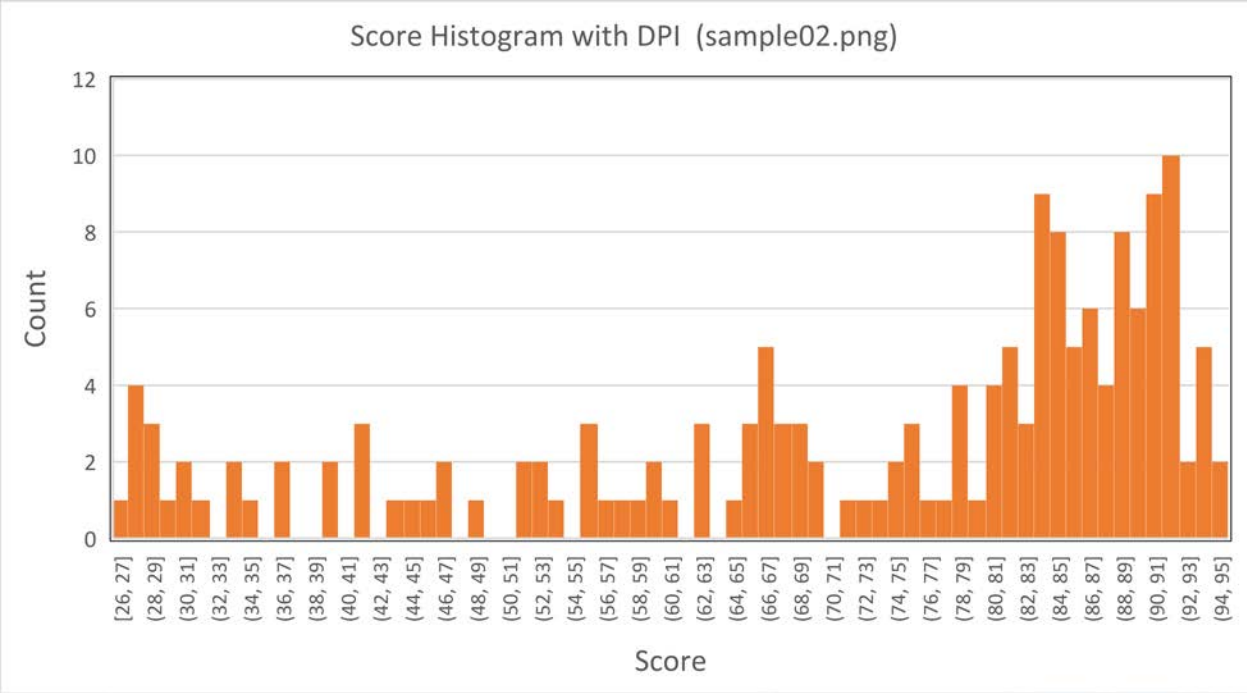


Fig 69. Approximate score distribution of all combinations for sample 2 with DPI value 300

Erode

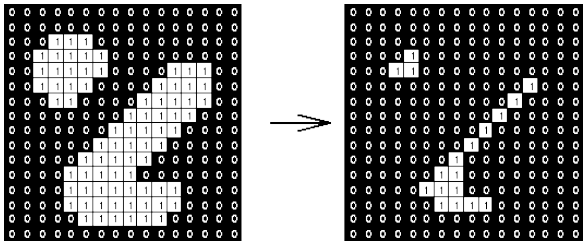


Fig 70. Visual Representation of erode method

Since the text lost some of its features I attempt to recover the feature with eroding methods where all zero-intensity value pixel are 'expand' making the text appear bold as illustrated in Fig 70. However, after some trial and error, it does not show any sign of improvement.

Possible Improvements

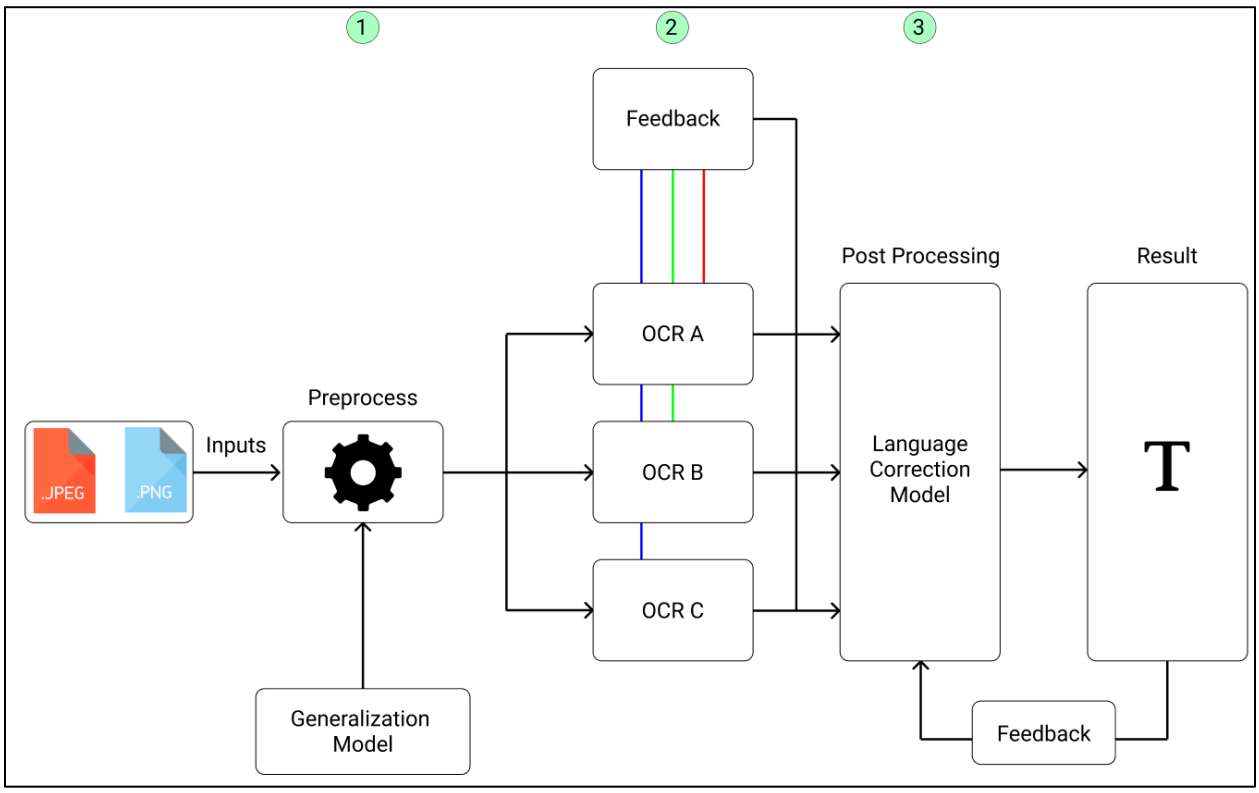


Fig 71. Possible enhancement architecture for improvement of OCR accuracy and robustness.

1. One possible way to improve is to introduce and formulate a general model for preprocessing. It consists of a group of dynamic methods that can tolerate all kinds of image text such as skew, occlusion, lightings, noise, etc. It is also able to tune the optimal parameters via neural networking training with a CNN (Convolutional Neural Network) or Autoencoder approach for unsupervised learning. Thus, the likelihood of resulting in a more consistent image output.

   Apart from that, introduce more effective image processing methods such that it can predict a better result. For example, a low-quality image may use a combination of methods A, B, C while a high-quality image may use a combination of methods X, Y, Z. Another approach can be to trying out contouring methods to improve the OCR accuracy as well.

2. Utilize other well-known OCR software such as ABBYY FineReader, Kofax Omnipage, and choose the best accuracy from there. Perhaps, introduce a feedback model to optimize and generalize each unique OCR model better.

3. At postprocessing, introduce a language correction model to correct and predict the right sentence structure, word anomalies, etc. This can behave similarly to 'Grammarly' software, where it suggests correct spelling, grammar to the user. For this case, it autocorrects before return the result to the user. Apart from that, it can learn from the user correct result via feedback from the user such as prompting them if this result is correct or wrong via tuning weights and bias via neural network training.

Conclusion

From this experiment, we have exposed different knowledge to enhance image processing in terms of text recognition, and trying out different methods have yielded different results. There is no perfect solution for all types of images. There is always a tradeoff and imperfection in a different situation. Just like a real-world problem, the text document varies a lot and much more complex as compared to what we are experimenting with within this project. This project has provided us a good starter and handle challenging real-world problems. There is always room for improvement as long as we keep continuing and experiment and try out different architecture and various techniques. We can effectively produce better OCR accuracy more easily.

Appendix

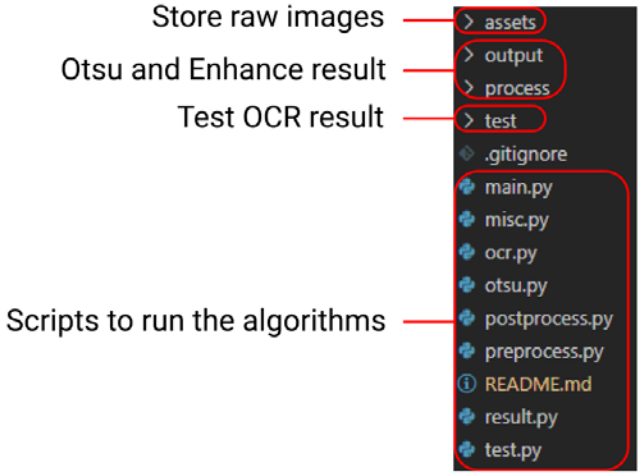
Assets

A public folder has been created to access the remaining assets since it is over 800Mb:

https://entuedu-my.sharepoint.com/:f:/g/personal/jsam002_e_ntu_edu_sg/ElkUrf6_XkJLpWfJDY40iMUBpVjEeVT2O9SWMJFWe3agvg?e=ciANlT

Directory Navigation

Directory Organisation



Ostu and Enhance Results

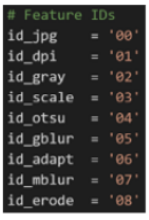
Store figures:
- Confidence distribution
- Pixel intensity distribution of the image
- Pixel intensity distribution with threshold for Otsu algorithm

Store image:
- Output images of the result
- Output images with plot from confidence rate, predict words and bounding box

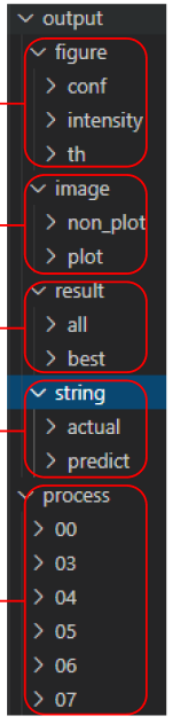
Store distance and score result (VIP):
- All results
- Best results

Store strings:
- Actual Sample01 and Sample02
- All predicted strings results

Store selected process image result base corresponding ID value



```
# Feature IDs
id_jpg    = '00'
id_dpi    = '01'
id_gray   = '02'
id_scale  = '03'
id_otsu   = '04'
id_gblur  = '05'
id_adapt  = '06'
id_mblur  = '07'
id_erode  = '08'
```



How to use the program 101

For details refer to README.md

Prerequisite

- Have a decent computer enough to handle process-intensive program
- All relevant tools used (refer to the section of the document) installed or imported.
- Ensure the IDE use can execute a python script successfully with the right configuration settings.
- Ensure CZ4003_CV_PROJECT_CODE_U1821296L.zip is downloaded into a local directory.

Execute the program

1. Run main.py script
2. Ensure the default configuration settings and declaration are as follows:

```
# Feature Settings
mode_jpg           = True
mode_dpi           = False
mode_scale         = True
mode_gblur         = True
mode_adapthreshold = True
mode_mblur         = True
mode_erode         = False

# Dynamic Declaration
scale_percent      = [200,250,300]
gBlur              = [5,7,11]
adapt_kernel       = [11,17,37]
adapt_const        = [2,3]
medBlur            = [1,5,7]
tuner              = [scale_percent,gBlur,adapt_kernel,adapt_const,medBlur]

# Static Declaration
dpi_value          = 300
erode_val          = 1
threshold          = 0
best_result        = []
best_img           = []
temp_result        = []
```

3. Configure the right settings to execute the respective algorithms as shown below:

```
72  # Mode
73  test_mode          = False
74  base_mode          = False
```

   a. To Run the **Test OCR Accuracy**
      i. Set ‘test_mode’ as **True**
   b. To Run the **Otsu Algorithm**
      i. Set ‘base_mode’ as **True**
      ii. Set ‘test_mode’ as **False**
   c. To Run the **Enhanced Algorithm**
      i. Set ‘base_mode’ as **True**
      ii. Set ‘test_mode’ as **False**

Source Code

Details can be found on CZ4003_CV_PROJECT_CODE_1821296L.zip

main.py (Declaration and Settings)

```python
# Import other python files and libraries
from preprocess import *
from ocr import *
from postprocess import *
from result import *
from misc import *
from test import *
from otsu import *
import sys
import time                # Elapsed Time

# Declared Path to Assets
path_main   = 'CZ4003_CV_PROJ/'
path_raw    = path_main + 'assets/'
path_pro    = path_main + 'process/'
path_act    = path_main + 'output/string/actual/'
path_pre    = path_main + 'output/string/predict/'
path_imgOP  = path_main + 'output/image/non_plot/'
path_imgPt  = path_main + 'output/image/plot/'
path_resB   = path_main + 'output/result/best/'
path_resA   = path_main + 'output/result/all/'
path_figC   = path_main + 'output/figure/conf/'
path_figT   = path_main + 'output/figure/th/'
path_figI   = path_main + 'output/figure/intensity/'

# Feature IDs
id_jpg    = '00'
id_dpi    = '01'
id_gray   = '02'
id_scale  = '03'
id_otsu   = '04'
id_gblur  = '05'
id_adapt  = '06'
id_mblur  = '07'
id_erode  = '08'

# Feature Settings
mode_jpg           = True
mode_dpi           = False
mode_scale         = True
mode_gblur         = True
mode_adapthreshold = True
mode_mblur         = True
mode_erode         = False

# Dynamic Declaration
scale_percent      = [200,250,300]
gBlur              = [5,7,11]
adapt_kernel       = [11,17,37]
adapt_const        = [2,3]
medBlur            = [1,5,7]
tuner              = [scale_percent,gBlur,adapt_kernel,adapt_const,medBlur]

# Static Declaration
dpi_value          = 300
erode_val          = 1
threshold          = 0
best_result        = []
best_img           = []
temp_result        = []

# Display Settings
pd.set_option("display.max_rows", 50, "display.max_columns",  None, 'display.width'
, get_terminal_size()[0])
display_predict    = False
display_actual     = False
display_preprocess = False
display_result     = True

# Clear Settings
remove_process     = False

# Mode
test_mode          = False
base_mode          = False

# Limiter (Control the number of combination execution)
limit              = math.inf
```

main.py (Content)

```python
def main():
    # OCR Integrity Checks
    if(test_mode == True):
        run_integrity_check()
        return

    # Run Main Algorithm
    start = time.time()
    print('Initialise')
    combi = generate_combi(tuner) # All sequence of parameters combination for processing

    # Clear All Directories
    if(remove_process == True):
        clear_dir(path_pro)
        clear_dir(path_imgOP)
        clear_dir(path_imgPt)
        clear_dir(path_figC)
        clear_dir(path_figT)
        clear_dir(path_figI)
        clear_dir(path_resB)
        clear_ResultF(path_resB)
        clear_Result(path_resA)
        clear_JSON(path_pre)

    # Get RAW Image
    print('[ Retriving RAW Images ]')
    items = read(path_raw)

    # Feature Enhancement
    if(base_mode == False):
        if(mode_jpg == True):
            print('[ Convert to JPG Format ]')
            for item in items:
                image, filename = item
                image_save(image,filename,path_pro,id_jpg,'','.jpg', True)
            items = read(path_pro + id_jpg + '/')
        if(mode_dpi == True):
            print('[ Change Image DPI ]')
            dpi(dpi_value,path_pro + id_jpg ,  path_pro ,id_dpi,'')
            items = read(path_pro + id_dpi + '/')
    # For every images in the asset directory
    for item in items:
        limit_ini       = 0
        best_dist       = math.inf
        index = 0
        temp_result = []
        # For every combination sequence
        for sc,gb,ak,ac,mb in combi:
            print('-----------------------Iteration:',index + 1,'-----------------------')
            code = str(sc).zfill(3) + '-' + str(gb).zfill(2) + '-' + str(ak).zfill(2) + '-' + str
(ac).zfill(1) + '-' + str(mb).zfill(1)
            image, filename = item
            print('[ Convert to GrayScale ]')
            image = grayscale(image)

            # Run Otsu
            if(base_mode == True):
                code = 'base'
                print('[ Run OTSU algorithm (Global Threshold)]')
                image, opt_th, hist, vacs = otsu(image)
                image_save(image,filename,path_pro,id_otsu,code,'.jpg',True)
                fig_otsu = plot_hist_threshold(opt_th,hist,vacs)

            # Feature Enhancement
            else:
                if(mode_scale == True):
                    print('[ Scale Image ]')
                    image = scale(image,sc)
                    image_save(image,filename,path_pro,id_scale,code,'.jpg',True)
                    fig_scale = plot_hist(image)
                if(mode_gblur == True):
                    print('[ Smooth Image (Gassian)]')
                    image = gaussianBlur(image,gb)
                    image_save(image,filename,path_pro,id_gblur,code,'.jpg',True)
                    fig_gblur = plot_hist(image)
                if(mode_adapthreshold == True):
                    print('[ Run Adaptive Threshold]')
                    image = cv.adaptiveThreshold(image,255
,cv.ADAPTIVE_THRESH_GAUSSIAN_C,cv.THRESH_BINARY,ak,ac)
                    image_save(image,filename,path_pro,id_adapt,code,'.jpg',True)
                    fig_adapt = plot_hist(image)
                if(mode_mblur == True):
```

```
                    print('[ Smooth Image (Median)]')
                    image = cv.medianBlur(image,mb)
                    image_save(image,filename,path_pro,id_mblur,code,'.jpg',True)
                    fig_mblur = plot_hist(image)
                if(mode_erode == True):
                    print('[ Erode Image ]')
                    image = erode(image,erode_val)
                    image_save(image,filename,path_pro,id_erode,code,'.jpg',True)
                    fig_erode = plot_hist(image)

            print('[ Run OCR algorithm ]')
            # Get Predict Result
            data_word       = ocr_word(image)
            data_str_Pre    = ocr_string(image)

            # Get Actual Result
            data_str_Act    = get_text(filename, path_act)

            # Post Processing
            data_str_Pre    = clean_String(data_str_Pre)
            data_str_Act    = clean_String(data_str_Act)

            # Result
            dist, score     = similarity(data_str_Pre,data_str_Act)
            if(display_result == True):
                result_display(filename,code,dist,score)

            # Visualisation
            image_plotW     = plot_word(image,data_word,threshold)
            fig_conf        = plot_hist_conf(data_word)

            # Save Result
            print('[ Save Result ]')
            image_save(image,filename,path_imgOP,'',code,'.jpg',False)
# Save Image Non-Plot Image
            result_save(path_resA,path_pre,filename,code,dist,score,data_str_Pre)
# Save All Possible Outcome Image
            image_save(image_plotW,filename,path_imgPt,'',code,'.jpg',False)
# Save Plotting Image
            fig_save(fig_conf,filename,path_figC,code)
# Save Figure Confidence

            if(base_mode == True):
                fig_save(fig_otsu,filename,path_figT,code)
# Save Figure Threshold
                break
            else:
                # Save Figure Intensity Distribution
                if(mode_scale == True):
                    fig_save(fig_scale,filename,path_figI,code + '-' + 'scale')
                if(mode_gblur == True):
                    fig_save(fig_gblur,filename,path_figI,code + '-' + 'gblur')
                if(mode_adapthreshold == True):
                    fig_save(fig_adapt,filename,path_figI,code + '-' + 'adapt')
                if(mode_mblur == True):
                    fig_save(fig_mblur,filename,path_figI,code  + '-' + 'mblur')
                if(mode_erode == True):
                    fig_save(fig_erode,filename,path_figI,code + '-' + 'erode')

                # Update Best Result
                if(dist <= best_dist):
                    best_dist = dist

                # Store all result to be choose later
                temp_result.append([filename,code,dist,score,image])

                # Limit Iteration
                if(limit_ini >= limit):
                    break
                limit_ini += 1
            index += 1

        if(base_mode == False):
            print('Finding Best Score')
            # Find Best Result
            best_result = []
            for filename_,param_,dist_,score_,image_ in temp_result:
                if(dist_ == best_dist):
                    best_result.append([filename_,param_,dist_,score_,image_])

            print('Saved Best Result')
            # Save Best Result
            best_result_save(best_result,path_resB)

    end = time.time()
    print('Total Time Taken: ',round(end - start,3),'s')
    print('End of Operation')
    exit(0)

if __name__ == "__main__":
    main()
```

misc.py

```python
# Import Libraries
import os               # Read and Write Directory
import shutil           # Advanced Clear Directory
import cv2 as cv        # Read Images
import itertools        # Create combination
import json             # Read and Write JSON operation
import numpy as np      # Execute patterns

# Generate all possible combination for max test
def generate_combi(param):
    combi_list = list(itertools.product(*param))
    #combi_list = list(itertools.product([0, 1], repeat=value))
    return combi_list

# Read Image from assets directory
def read(path):
    items = []
    for filename in os.listdir(path):
        print('Reading... ', str(path) +  str(filename))
        if(os.path.isfile(path + filename)):
            items.append([cv.imread(path + filename),filename])
    return items

# Clear the directory of the process space
def clear_dir(path):
    print('Clearing files for process...')
    for item in os.listdir(path):
        if(os.path.isdir(path+item)):
            shutil.rmtree(path+item)
            print('Removed', path+item)
        else:
            os.remove(path+item)
            print('Removed', path+item)

# Save an image into respective process directory
def image_save(image,filename,path,foldername,param,ext,newdir):
    # Create a directory if does not exist
    if(newdir == True):
        if(os.path.exists(path+foldername) == False):
            os.mkdir(path + foldername)
            print('Created',foldername,'folder')
        fname , _ = os.path.splitext(filename)
        # Save Image
        cv.imwrite(path+foldername+'/'+fname+'_'+ param + ext, image)
        print('Successfully Saved Image:          ', path +foldername+'/'+fname+'_'+ param + ext)
    else:
        fname , _ = os.path.splitext(filename)
        # Save Image
        cv.imwrite(path + fname+'_'+ param + ext, image)
        print('Successfully Saved Image:          ', path  +fname+'_'+param + ext)

# Clear JSON content
def clear_JSON(path):
    f = open(path + 'dataset.json', 'w').close()
    print('Cleared:', path + 'dataset.json')

# Clear Result content
def clear_Result(path):
    open(path + 'dataset.txt', 'w').close()
    print('Cleared:', path + 'dataset.txt')

# Clear Result content (For Best Result)
def clear_ResultF(path):
    open(path + 'best_result.txt', 'w').close()
    print('Cleared:', path + 'best_result.txt')
```

ocr.py

```python
# Import Libraries
import pytesseract.pytesseract as tes       # Get Tersseract Operation
from shutil import get_terminal_size        # Control terminal output limit
import matplotlib.pyplot as plt             # Visual Analysis Purpose
import pandas as pd                         # Display and manipulate data
import numpy as np                          # Manipulate data
import math                                 # Ceil and Floor Operation

# Directory to Run Tesseract Command Prompt
tes.tesseract_cmd = 'C:\\Program Files\\Tesseract-OCR\\tesseract.exe'
index_W = 0
index_S = 0


# Display OCR Return Word-Level
def ocr_word_display(data):
    global index_W
    data_copy = data.copy()
    data_copy.index += 1 # Start index from 1 instead of 0
    display_round = math.ceil(data.shape[0]/50)
    print('Processing Image Index (Word)     = ', index_W + 1)
    print('------------------------------------------------------------')
    print('--------------------------start-------------------------')
    for i in range(display_round):
        print(data[(i*50):50*(i+1)])
    print('--------------------------end---------------------------')
    print('------------------------------------------------------------')
    index_W += 1

# Retrieve all information from OCR algorithm
def ocr_word(image):
    data = tes.image_to_data(image, output_type='data.frame', config = '--psm 3 --oem 3')
    return data

# Display OCR Return Strings
def ocr_string_display(data):
    global index_S
    print('Processing Image Index (Full Strings) = ', index_S + 1)
    print('------------------------------------------------------------')
    print('--------------------------start-------------------------')
    print(data)
    print('--------------------------end---------------------------')
    print('------------------------------------------------------------')
    index_S += 1

# Retrieve the detected strings from OCR algorithm
def ocr_string(image):
    text = tes.image_to_string(image, config = '--psm 3 --oem 3')
    return text
```

otsu.py

```python
# Import Libraries
import matplotlib.pyplot as plt # Visual Analysis Purpose
import numpy as np              # Manipulation data and histogram
import cv2 as cv                # Image processing operation
import math                     # Math Functions

# Implement Otsu algorithms
def otsu(image):
    # Find total number of pixels in the image
    total_pixels = image.shape[0] * image.shape[1]

    # Get the weight factor with respect to 1
    mean_weight = 1.0/float(total_pixels)

    # Get # of bins and hist array
    hist, bins = np.histogram(image, np.arange(257))
# inconsistent in bins and hist shape , bins + offset value of 1 is needed

    # Set Initial Values
    optimal_thresh = -math.inf
    optimal_value  = -math.inf
    intensity_arr = np.arange(256)
    index = 0
    var_values = []

    # Iterate through all the intensity values from 0 to 255
    for t in range(len(bins)-1):
        # Sum foreground and background classification with a given threshold t
        pcb = np.sum(hist[:t])
        pcf = np.sum(hist[t:])

        # Probability of foreground and background classification
        Wb = pcb * mean_weight
        Wf = pcf * mean_weight

        # Calculate the means of foreground and background
        if(pcb != 0): # Prevent Undefined case
            mub = np.sum(intensity_arr[:t]*hist[:t]) / float(pcb)
        else:
            mub = 0
        if(pcf != 0): # Prevent Undefined case
            muf = np.sum(intensity_arr[t:]*hist[t:]) / float(pcf)
        else:
            muf = 0

        # Calculate the variance
        value = Wb * Wf * ((mub - muf) ** 2)


        # Find the maximum variance value
        if value > optimal_value:
            optimal_thresh = t
            optimal_value = value

        # Store Values
        var_values.append(value)
        index += 1

    # Binarization Image base on optimal threshold value
    otsu_image = image.copy()
    otsu_image[image > optimal_thresh] = 255
    otsu_image[image <= optimal_thresh] = 0

    return otsu_image, optimal_thresh, hist, var_values

# Display Threshold Result
def threshold_display(otsu_image, optimal_thresh, hist, var_values):
    for value in var_values:
        print('['+str(index).zfill(3)+']',value)
    print('Optimal Threshold is',optimal_thresh)


# Display Histogram with Thresholding result
def plot_hist_threshold(opt_var,hist,var_values):
    fig = plt.figure(figsize=(20,10))
    fig.canvas.set_window_title('Threshold Result')
    plt.xticks(np.arange(0, 255, step=10))
    plt.bar(np.arange(0,opt_var+1),hist[:opt_var+1],color = 'lightgreen',label =
'Below Threshold Value')
    plt.bar(np.arange(opt_var+1,256),hist[opt_var+1:], color = 'lightsalmon',label =
'Above Threshold Value')
    plt.axvline(opt_var,linewidth = 1 ,color = 'darkorange',label = 'Optimal Variance Value')
    plt.plot(var_values, linestyle = '--', color =  'blue', label = 'Variance Values')
    plt.legend(loc = 'best')
    plt.title('Image Histogram and Threshold Values (Optimal Threshold Value = '+ str(opt_var)+')',
fontsize=14,figure = fig)
    plt.ylabel('Count', fontsize=14,figure = fig)
    plt.xlabel('Pixel Intensity',fontsize=14, figure = fig)
    #plt.show()
    return fig
```

postprocess.py

```python
# Import Libraries
import re                       # Clear noise data
import os                       # Files and Directory operation
import json                     # Read and Write JSON operation

# Display clean string without tabs and newlines
def clean_String_display(text):
    print('----------------------Clean Text--------------------------')
    print(text)
    print('----------------------------------------------------------')

# Clean String without tabs and newlines
def clean_String(text):
    text = re.sub(' {2,}', ' ', text)
    text = " ".join(text.split())
    # Fixed Encoding Issues (python dont support)
    text = text.replace(u"\u2018", "'").replace(u"\u2019", "'").replace(u"\u201d", "\"").replace(u"\u201c", "\"")
    return text

# Display JSON Content
def get_text_display(data):
    print('----------------------------------------------------------')
    print('-------------------------Actual---------------------------')
    print(data)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

# Get Text from JSON File
def get_text(filename, path):
    name, _ = os.path.splitext(filename)
    name = name[ 0 : 8 ]
    try:
        print('Loading... ', path + 'dataset.json')
        # Open Json File
        f = open(path + 'dataset.json')
        # Load Json as Dict
        data = json.load(f)
    except Exception as e:
        print('[Error] Missing ''dataset.json'' not found!' )
        print(e)
    return data[name]
```

preprocess.py

```python
# Import Libraries
import os                       # Retrieve dataset
import cv2 as cv                # Image Processing Operation
from PIL import Image           # DPI Operation
import numpy as np              # Data manipulation
import matplotlib               # Fixed unknown error
matplotlib.use('Agg')

# Add DPI into images
def dpi(value,path,newpath,foldername,filename):
    # Create new folder (to be save directory)
    if(os.path.exists(newpath+foldername) == False):
        os.mkdir(newpath + foldername)
        print('Created',foldername,'folder')
    # Check if it is a file
    for filename_ in os.listdir(path):

        if(os.path.isfile(path + '/' + filename_)):
            im = Image.open(path + '/' + filename_)
            fname , _ = os.path.splitext(filename_)
            newfname , _ = os.path.splitext(filename)
            im.save(newpath + foldername + '/' + fname[0:8] + '_'+ newfname +'.jpg', dpi
=(value,value))

# Convert Image to Grayscale
def grayscale(image):
    print('Before Convert to Grayscale Image Dimension = ', image.shape)
    image_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    print('After  Convert to Grayscale Image Dimension = '
, cv.cvtColor(image, cv.COLOR_BGR2GRAY).shape)
    return image_gray

# Scale Image
def scale(image,scale_percent):
    print('Original Dimensions : ',image.shape)
    width = int(image.shape[1] * scale_percent / 100)
    height = int(image.shape[0] * scale_percent / 100)
    dim = (width, height)
    # Resize image
    image_resized = cv.resize(image, dim, interpolation = cv.INTER_AREA)
    print('Resized Dimensions : ',image_resized.shape)
    return image_resized

# Dilation
def dilate(image,value):
    kernel = np.ones((value,value),np.uint8)
    return cv.dilate(image, kernel, iterations = 1)

# Erosion
def erode(image,value):
    kernel = np.ones((value,value),np.uint8)
    return cv.erode(image, kernel, iterations = 1)

# Opening - erosion followed by dilation
def opening(image,value):
    kernel = np.ones((value,value),np.uint8)
    return cv.morphologyEx(image, cv.MORPH_OPEN, kernel)

# Skew correction
def deskew(image):
    coords = np.column_stack(np.where(image > 0))
    angle = cv.minAreaRect(coords)[-1]
```

```
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = -angle
    (h, w) = image.shape[:2]
    center = (w // 2, h // 2)
    M = cv.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv.warpAffine(image, M, (w, h), flags=cv.INTER_CUBIC, borderMode=cv.BORDER_REPLICATE)
    return rotated

# Apply Gaussian Blur
def gaussianBlur(image,value):
    return cv.GaussianBlur(image,(value,value),0)

# Calculate skew angle of an image
def getSkewAngle(image) -> float:

    # Find all contours
    contours, hierarchy = cv.findContours(image, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key = cv.contourArea, reverse = True)

    # Find largest contour and surround in min area box
    largestContour = contours[0]
    minAreaRect = cv.minAreaRect(largestContour)

    # Determine the angle. Convert it to the value that was originally used to obtain skewed image
    angle = minAreaRect[-1]
    if angle < -45:
        angle = 90 + angle
    return -1.0 * angle

# Rotate the image around its center
def rotateImage(image, angle: float):
    new_image = image.copy()
    (h, w) = new_image.shape[:2]
    center = (w // 2, h // 2)
    M = cv.getRotationMatrix2D(center, angle, 1.0)
    new_image = cv.warpAffine(new_image, M, (w, h), flags=cv.INTER_CUBIC, borderMode
=cv.BORDER_REPLICATE)
    return new_image
```

result.py

```python
# Import Libraries
import cv2 as cv                # Display metrics value on image
import numpy as np              # Manipulate data
import editdistance             # Calculate similariity between 2 strings
import matplotlib.pyplot as plt # Visual Analysis Purpose
import json                     # Perform JSON operation
import os                       # Split Filename

index_R = 0

# Plot Histogram by Image Intensity Values
def plot_hist(image):
    hist, _ = np.histogram(image, np.arange(257))
    fig = plt.figure(figsize=(20,10))
    plt.xticks(np.arange(0, 255, step=10))
    plt.bar(np.arange(0,256),hist,color = 'red')
    plt.title('Image Intensity Histogram',fontsize=14,figure = fig)
    plt.ylabel('Count', fontsize=14,figure = fig)
    plt.xlabel('Pixel Intensity',fontsize=14, figure = fig)
    plt.close()
    return fig

# Plot Histogram by Word-Confidence
def plot_hist_conf(data):
    fig = plt.figure(figsize=(20,10))
    plt.xlim(0,100)
    plt.xticks(np.arange(0, 100, step=5))
    data = data[data['conf'] >= 0]
    if(len(data['conf']) == 0):
        print('[Error] No Confidence Found')
        return
    plt.hist(data['conf'], bins = len(data['conf'].unique()), color = 'red', figure = fig)
# -1 because omit -1
    plt.title('Distribution of Confidence', fontsize = 14,figure = fig)
    plt.ylabel('Count', fontsize = 14,figure = fig)
    plt.xlabel('Confidence Rate (%)', fontsize = 14,figure = fig)
    plt.close()
    return fig

# Save Figures
def fig_save(fig,filename,path,param):
    #filename = filename.lower()
    if fig is None:
        print('[Error] No figure saved')
        return
    name,_ = os.path.splitext(filename)
    fig.savefig(path + name + '_' + param +'.jpg', pad_inches = 0)
    print('Successfully Saved Figure:         ', path + name + '_' + param +'.jpg')

# Plot Word-Level Detection into image with the given threshold for confidence
def plot_word(image,data,threshold):
    # Convert to BGR if it is a gray image for plotting colors
    try:
        image_ = cv.cvtColor(image,cv.COLOR_GRAY2BGR).copy()
    except:
        image_ = image.copy()

    for word in range(len(data['text'])):
        # Extract all the respective coordinates, width and height of the bounding boxes
        (x, y, w, h) = (data['left'][word], data['top'][word], data['width'][word], data['height'
][word])
        # Plot all detected filled box as long it is not false value (-1)
        if(data['conf'][word] > 0):
            # Extract all bounding require values from the data

            # Create Mask
            sub_img = image_[y:y+h, x:x+w]
            overlay_rect = np.ones(sub_img.shape, dtype=np.uint8)


# Normalize and Produce to Green (conf = 100) to Red (conf = 0) gradient using 3 B,G,R channels
            b,g,r = overlay_rect[:, :, 0], overlay_rect[:, :, 1]*int((data['conf'][word]/100)*255
), overlay_rect[:, :, 2]*int((1-(data['conf'][word]/100))*255) # For RGB image

```

```
            # Combine individual B,G,R masks into one
            overlay_rect2 = cv.merge((b,g,r))

            # Merge mask with the detected portion of the image
            res = cv.addWeighted(sub_img, 0.5, overlay_rect2, 0.5, 1.0)

            # Overwrite this portion of image to the main copy of the image
            image_[y:y+h, x:x+w] = res

        # Plot Text and Bounding Box if beyond a threshold value
        if(int(data['conf'][word]) > threshold):
            image_ = cv.rectangle(image_, (x, y), (x + w, y + h), (255, 0, 0), 1)
            cur_text = data['text'][word].replace(u"\u2018", "'").replace(u"\u2019", "'").replace(u"\u201d", "\"").replace(u"\u201c", "\"")
            image_ = cv.putText(image_, text= cur_text, org=(x,y), fontFace = cv.FONT_HERSHEY_SIMPLEX, fontScale=0.5, color=(0,0,255), thickness=1, lineType=cv.LINE_AA)
    return image_

# Show Result
def result_display(filename,param,dist,score):
    global index_R
    print('Filename =',filename)
    print('Settings =',str(param))
    print('Distance =',dist)
    print('Score    =',score)
    index_R += 1

# Save Result
def result_save(path_res,path_pre,filename,param,dist,score,predict_content):
    # Save Result to Text File
    f = open(path_res + 'dataset.txt',"a")
    f.write(filename + '-' + str(param) + ' Distance:' + str(dist) + ' Score:' + str(score) + '\n')
    f.close()
    print('Successfully Saved Result:          ' + path_res + 'dataset.txt')

    # Save Predict Content to JSON File
    filename = filename.lower()
    name,_ = os.path.splitext(filename)
    f = open(path_pre + 'dataset.json','r+')

    # Try Load JSON file if exist content or else load new dictionary in
    try:
        data = json.load(f)
        f.close()
        data[name +'_'+ param] = predict_content
    except:
        print('[Warning] Update JSON Fail, ignore if this is the first warning')
        data = json.loads('{' + '"'+  name +'_' + param + '":"' + predict_content  + '"}')
    f = open(path_pre + 'dataset.json','w')
    json.dump(data, f, indent=4)
    f.close()
    print('Successfully Saved Predict Content: ' + path_pre  + 'dataset.json')

# Caculate Levenshtein distance and the relative score with respect to the actual text
def similarity(predict_string,act_string):
    d = editdistance.eval(predict_string,act_string)
    try:
        score = round((1 - d/len(act_string)) * 100,4)
    except:
        print('[Error] Empty Actual String')
        score = -1
    return d , score

# Save best result into directory
def best_result_save(best_result,path):
    # Open Text File
    f = open(path + 'best_result.txt',"a")
    # Save all best results and images
    for filename,param,dist,score,image in best_result:
        filename = filename.lower()
        name,ext = os.path.splitext(filename)
        f.write(name +'-'+param + '-' + ' Distance: ' + str(dist) + ' Score: ' + str(score) + '\n')
        cv.imwrite(path+name+'_'+param +ext, image)
        print('Successfully Saved Image:', path+name+'_'+param+ext)
    f.close()
```

test.py

```python
#  Import main python file
from main import *

# [OCR Integrity Test]
## Directory
path_test_main  = 'CZ4003_CV_PROJ/test/'
path_test_img   = path_test_main + 'assets/'
path_test_act   = path_test_main + 'output/actual/'
path_test_pre   = path_test_main + 'output/predict/'
path_test_W     = path_test_main + 'output/word/'
path_test_res   = path_test_main + 'output/result/'
path_test_fig   = path_test_main + 'output/figure/'
path_test_imgOP = path_test_main + 'output/image/'
path_test_best  = path_test_main + 'output/best/'

## Reduce Operation Time (For Test Mode Only)
run_plotnfig       = True

def run_integrity_check():
    start = time.time()
    # Initialization
    clear_JSON(path_test_pre)
    clear_Result(path_test_res)
    clear_dir(path_test_best)
    #clear_Result(path_test_best)
    clear_dir(path_test_imgOP)
    clear_dir(path_test_W)
    clear_dir(path_test_fig)

    # Static Settings
    dist_default = 0
    threshold    = 0

    # Read Test Assets
    items = read(path_test_img)
    for item in items:
        image, filename = item
        best_dist       = math.inf

        # Get Predict Result
        if(run_plotnfig == True):
            data_word       = ocr_word(image)
        data_str_Pre    = ocr_string(image)

        # Display Predict Result
        if(display_predict == True):
            if(run_plotnfig == True):
                ocr_word_display(data_word)
            ocr_string_display(data_str_Pre)

        # Get Actual Result
        data_str_Act    = get_text('test', path_test_act)

        # Display Actual Result
        if(display_actual == True):
            get_text_display(data_str_Act)

        # Post Processing
        data_str_Pre    = clean_String(data_str_Pre)
        data_str_Act    = clean_String(data_str_Act)
```

```
61
62          # Display Preprocessing
63          if(display_preprocess == True):
64              clean_String_display(data_str_Pre)
65              clean_String_display(data_str_Act)
66
67          # Result
68          dist, score     = similarity(data_str_Pre,data_str_Act, dist_default)
69
70          # Display Result
71          if(display_result == True):
72              result_display(filename,'test',dist,score)
73
74          # Visualisation
75          if(run_plotnfig == True):
76              image_plotW     = plot_word(image,data_word,threshold)
77              fig             = plot_hist_conf(data_word)
78
79          # Save Result
80          image_save(image,filename,path_test_imgOP,'test','result')
81          result_save(path_test_res,path_test_pre,filename,'test',dist,score,data_str_Pre)
82          if(run_plotnfig == True):
83              image_save(image_plotW,filename,path_test_W,'test','process')
84              fig_save(fig,filename,path_test_fig,'test')
85
86          # Update Best Result
87          if(dist <= best_dist):
88              best_dist = dist
89
90          # Store all result to be choose later
91          temp_result.append([filename,'test',dist,score,image])
92
93      # Find Best Result
94      for filename_,param_,dist_,score_,image_ in temp_result:
95          if(dist_ == best_dist):
96              best_result.append([filename_,param_,dist_,score_,image_])
97
98      # Save Best Result
99      best_result_save(best_result,path_test_best)
100     end = time.time()
101     print('Total Time Taken: ',round(end - start,3),'s')
102     print('End of Integrity Test Operation')
```

References

[1] R. Haldar and D. Mukhopadhyay, “Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach,” arXiv.org, 06-Jan-2011. [Online]. Available: https://arxiv.org/abs/1101.1232 . [Accessed: 1st-Nov-2020].