

CZ4003 Project Report

Name: Khairul Amiruf

Matric: U1721607A

Contents

[Introduction 3](#_Toc57309937)

[Objective 3](#_Toc57309938)

[Experiment 4](#_Toc57309939)

[1. Implement the Ostu global thresholding algorithm. Evaluate the OCR accuracy. Discuss any problems with the Otsu global thresholding algorithm. 4](#_Toc57309940)

[Original image: 4](#_Toc57309941)

[Otsu global threshold: 4](#_Toc57309942)

[Evaluate the OCR accuracy: 5](#_Toc57309943)

[Discussion: 6](#_Toc57309944)

[2. Design your own algorithm to address the problem of Otsu global thresholding algorithm. 7](#_Toc57309945)

[Method: 7](#_Toc57309946)

[Output: 7](#_Toc57309947)

[General Observation 7](#_Toc57309948)

[Evaluate OCR Accuracy 7](#_Toc57309949)

[Specific Image Analysis (Enhancement): Image 1 9](#_Toc57309950)

[Specific Image Analysis (Enhancement): Image 2 12](#_Toc57309951)

[3. Discuss how to improve recognition algorithms for more robust and accurate character recognition while document images suffer from different types of image degradation. 14](#_Toc57309952)

[References 15](#_Toc57309953)

# Introduction

This project aims to utilize the concepts learn throughout the course of Computer Vision in the application of Optical Character Recognition domain. As briefed in the project task, OCR involves a series of image processing and recognition tasks:

1. Text image binarization – converts a colour/grayscale image into a binary image with multiple foreground regions (usually characters)
2. Connected component labelling that detects each binarized character region
3. Character recognition by using some classifiers such as a pre-trained neural network

## Objective

Explore and develop various image binarization algorithms targeting optimal character recognition accuracy.

## **Note**

The codes are found in ‘ocr.py’ and have been commented respective to the sections in this report. All output of this script can be found in the ‘output.zip’ folder.

# Experiment

## Implement the Ostu global thresholding algorithm. Evaluate the OCR accuracy. Discuss any problems with the Otsu global thresholding algorithm.

### Original image:

The **original** images are as shown below:A close up of a newspaper

Description automatically generated

Figure : sample01.png

Text, letter

Description automatically generated

Figure : sample02.png

### Otsu global threshold:

A picture containing text

Description automatically generated

Figure : sample01 after applying Otsu

Text

Description automatically generated

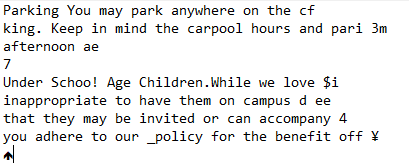
Figure : sample02 after applying Otsu

### Evaluate the OCR accuracy:

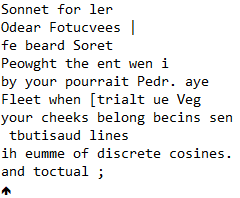
To evaluate the accuracy of the OCR process, we will be transcribing the text from the image recognized into an output file and compare this to the original image on a word level basis. An excerpt of the transcribed text will be shown in the report for discussion.

Transcription with Otsu implementation shown below:

**Image 1:**



**Image 2:**



**Observations:**

As we can see from the images above, the OCR transcription seems to be incoherent and the sentences are incomplete. Furthermore, there are some special characters that are being transcribed even when they are not present in the original image. This could be the result of noise in the threshold images.

### Discussion:

Thresholding divides the pixels into two classes: pixels higher than or lower than the threshold. The most optimal outcome would be a threshold that would minimize the intra-class variance while maximizing the inter-class variance.

The intra-class variance would mean that the pixels are less dispersed from one another in its respective class while inter-class variance would create a more distinctive difference between the two classes.

From the original images, we can see that the images are exposed to different levels of light, or a difference in illumination levels. This would suggest the image having more than one ‘regions’ to segment when it comes to applying filters. The limitations of simple thresholding come into play when there are more than two image regions to segment, which we are facing in this situation. As expected, we did not obtain a good result from Otsu global thresholding.

## Design your own algorithm to address the problem of Otsu global thresholding algorithm.

### Method:

1. Applying gaussian filter to the image to remove noise
2. Apply adaptive threshold

### Output:

Text

Description automatically generated

Figure : Adaptive threshold for image 1

Text

Description automatically generated

Figure : Adaptive threshold for image 2

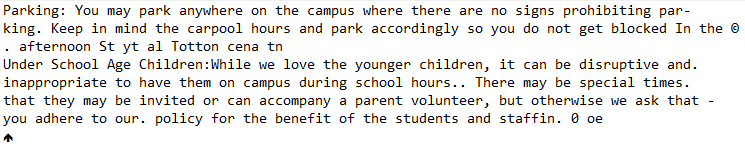
### General Observation

From the images above, we are able to obtain decent results where the output is not affected by the ‘two’ regions where one is completely obscured as we see in the Otsu’s thresholding. Most of the characters are discernible and readable.

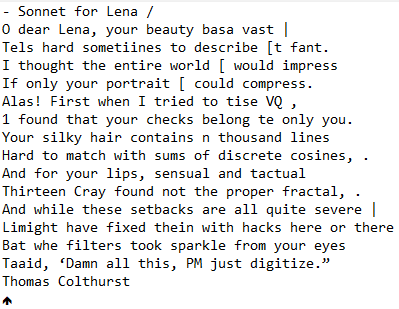
The adaptive thresholding method works better than Otsu’s because the threshold value is localized to the specific region of the image within the window size. By using `ADAPTIVE\_THRESH\_GAUSSIAN\_C`, the threshold value is a gaussian-weighted sum of the neighbourhood values (minus the constant c). As a result, the threshold would elicit better OCR accuracy as compared to a global thresholding like Otsu’s.

### Evaluate OCR Accuracy

**Image 1 OCR:**

****

**Image 2 OCR:**

****

**Observation:**

From the above images, we can see that the sentences are more fully formed and there are more characters appearing in the transcription than before (otsu implementation). However, there are instances where the transcription is being misinterpreted. For example, in image one, the sentence transcribed to be “…so you do not get blocked In the ©. Afternoon St yt al Totton cena tn” as oppose to the ground truth: “so you do not get blocked in the afternoon”. Although most of the contents of the sentences are there, the presence of noise (as seen from the image in Figure 5) distorts the transcription process and consequently disrupts the meaning of the sentence altogether.

### Specific Image Analysis (Enhancement): Image 1

#### Method 1: Increase Gaussian Blur kernel size

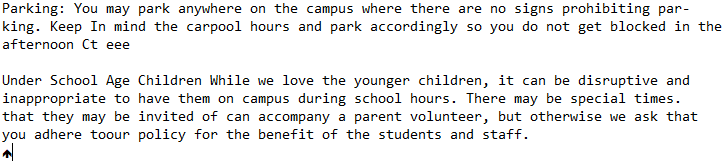
**Image 1 (sample01.png):**

For image 1, the output obtained comes with some noise towards the right side of the image. In order to reduce this noise, we can increase the kernel size of the gaussian filter. When increased the size of the kernel to 7, the following output is obtained. Text

Description automatically generated

Figure : kernel size=7

##### Evaluate OCR Accuracy



From the image above, we can see that we have successfully remove the bulk of the noise that was present in figure 5. This translates to better OCR transcription. However, the increase in size for the gaussian kernel comes at a cost. The semicolon in between “Under School Age Children” and “While” is removed and the space between “adhere to our policy” is not accurately being transcribed. Although, the core semantics of the texts is not lost. Some of the noise that remain present cannot be removed without compromising the quality of the texts.

#### Method 2: Morphological Transformation

OpenCV also provides another method to remove noise – via morphological transformations. The ‘opening’ method implements the erosion method followed by dilation. This method would be effective in reducing noise as the erosion method will decrease the size of the foreground (white region). Subsequently, the dilation method will join broken parts of an object. The result is as follows:

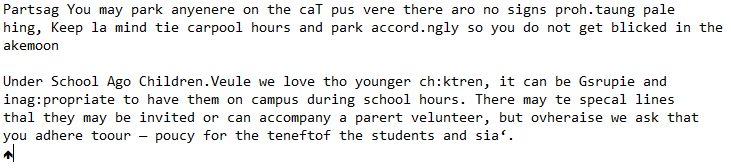
Text

Description automatically generated

Figure : morphology transformed

We can see that most of the speckled noise are removed from the image. We shall inspect if the OCR accuracy improves.

##### Evaluate OCR Accuracy



The OCR transcription is much more incoherent than before. Some of the words are misspelled while others are beyond comprehension. Even though the random noise were successfully removed, some of the character pixels are removed in the process. This creates a lot of broken lines just like in Canny edge detector. This could possibly be fixed with a process similar to Hough transform but for texts.

The distortion present within the character pixels decreases the accuracy of the OCR process.

It can be noted that this method of morphological transformation would be more effective with handwritten texts as individual’s handwriting are not uniform and some pen inks have different bold weight.

#### Method 3: Division

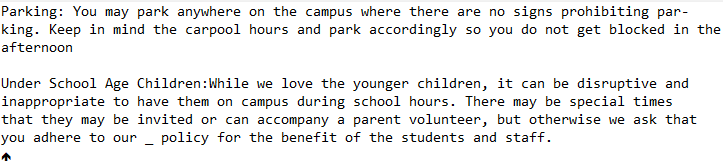
Another method, other than adaptive thresholding, that we can implement is an elementwise division of the original image and a gaussian filter to obtain a better result. The process is explained in the next section “specific image analysis (enhancement): image 2”.

Text

Description automatically generated

Just from the image above alone, we can see that our image is not distorted by noise and the texts are clear and illumination is rather uniform compared to the original image.

##### Evaluate OCR Accuracy



We achieve an almost perfect OCR accuracy with this method, other than the underscore character.

### Specific Image Analysis (Enhancement): Image 2

**Image 2 (sample02.png):**

The original image was not sharp to begin with. After applying the adaptive threshold method, we are able to fix the issues that came with Otsu’s global thresholding. However, the blurry texts still affect the accuracy of the OCR.

We can employ a different method other than adaptive thresholding to obtain better OCR accuracy.

#### Method: Division

1. Apply Gaussian filter with a very large kernel size (kernel = 95)
2. Element-wise divide the blurred image and the original image to obtain a resultant image of more ‘uniform’ illumination.
3. Apply OCR

Text, letter

Description automatically generatedA close up of a mans face

Description automatically generatedText, letter

Description automatically generated

Figure 9: original sample02.png Figure 10: high kernel gaussian blur Figure 11: resultant image

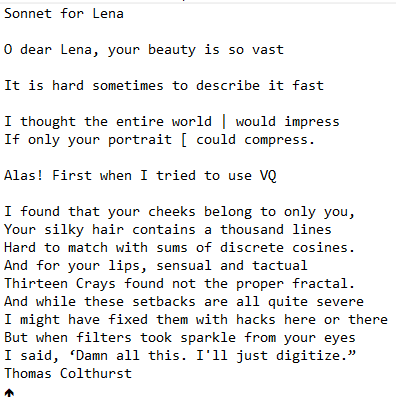
As we can see from Figure 11, the image now has more uniform illumination and is ready for OCR.

(\*Note: I have tried with a lower scale (black level) for the division which would be more visually appealing, and the texts are more human-readable, however this results in lower OCR accuracy)

Text, letter

Description automatically generated (for reference)

##### Evaluate OCR accuracy



From the image above, we can see that most of the texts are accurately transcribed with a few exceptions. For example, some of the ‘I’ characters are being transcribed as the ‘|’ character and ‘[‘ character. Still, this achieves quite a high level of OCR accuracy.

As the division operation is elementwise, we do not have to worry about the inconsistent illumination in the original image. The result would be a uniform output according to the scale parameter.

## Discuss how to improve recognition algorithms for more robust and accurate character recognition while document images suffer from different types of image degradation.

The top three applications of OCR in the real world is in the following domains: (Marshall, 2020)

1. Parking Management (License plate scanning)
2. Assessments (Efficient marking system)
3. Translation (Google translate and such)

There are many other domains where OCR can be applied too, however these areas shall be the focus of discussion. As we can see from the above exercise examples, images are affected by various degradation such as inconsistent illumination, text alignment being skewed, and sharpness of the image.

For the subsequent discussion, it will be on a domain dependent level and the following algorithm might only work on a general level to address more robust issues.

The **main idea** is to do a general preprocessing:

1. Use KMeans to define the borders or contours to elicit the regions of interest. (domain dependent)
2. Skew correction (to align text better)
   1. The alignment of the document has a direct effect on reliability and efficiency of the segmentation and feature extraction stages. (Jindal & Kaur, 2020)
3. (If handwritten text, else skip): To thin and ‘skeletonize’ the characters as handwritten characters are affected by inconsistent stroke width (Reddy, 2020)
4. Perform Gaussian Filter to remove noise
5. Using elementwise division to obtain the resultant corrected image
   1. To negate the effects of inconsistent illumination
6. Apply OCR algorithm to transcribe image to text
   1. Such as pytesseract as used in this project

## References

Jindal, S., & Kaur, L. (2020). Skew Detection Technique for Various Scripts. Retrieved 26 November 2020, from https://www.ijser.org/paper/Skew-Detection-Technique-for-Various-Scripts.html#:~:text=Correction%20in%20the%20skewed%20scanned,text%20lines%20and%20text%20blocks.

Marshall, C. (2020). How is OCR used in the real world?. Retrieved 26 November 2020, from https://medium.com/mysuperai/how-is-ocr-used-in-the-real-world-e82d1354f07b

OpenCV: Morphological Transformations. (2020). Retrieved 26 November 2020, from https://docs.opencv.org/master/d9/d61/tutorial\_py\_morphological\_ops.html

Reddy, S. (2020). Pre-Processing in OCR!!!. Retrieved 26 November 2020, from https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7