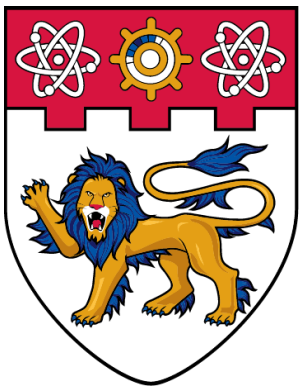


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4003 Computer Vision Project

Ng Joshua Jeremiah U1821510F
Mark Chua De Wen U1821979L

Contribution: Both parties have equal part to play in project planning, coding, results evaluation, and report writing.

Contents

1 The Introduction

Optical character recognition (OCR) is the conversion of printed text or handwritten text into editable text [1]. This technology allows machine to recognize text automatically. However, one of the difficulties faced by machines is differentiating similar characters such as the letter ‘O’ and the digit ‘0’. There is also the issue of bad contrast or noisy environments that will affect the accuracy of OCR. Applications of this technology include license plate recognition, text extraction from natural scenes, extracting text from scanned documents etc. Hence, many OCR systems were created, with the aim of maximizing accuracy. A system presented by Apurva A. Desai [2] is uses Artificial Neural Network (ANN) to recognize Guajarati handwritten numeral, achieving an avg of 82% recognition for Gujarati digits. U. Pal et al. [3] used Support Vector Machines (SVM) method for Bangla and Devnagari text recognition, achieving accuracy scores of 99.18% and 98.86% for Devnagari and Bangla characters, respectively. Bilal et al. [4] suggested local binarization method by using a Thresholding method and dynamic windows, which achieved F-mean values of 85.1% for handwritten text and 90.93% for printed text.

There are many types of OCR tools available in the current market, such as Desktop OCR, Server OCR, Web OCR etc. The accuracy of such systems ranges from 71% to 98% [1]. However, most of the OCR tools available are locked behind a pay wall, with only a few being free and open sourced. One such system is Tesseract, which is a free OCR software [5]. Tesseract is available on multiple platforms such as Windows, Ubuntu, Linux etc.

Tesseract performs best under the following conditions. Firstly, the document images must have a good contrast between the foreground text and the background. Secondly, the images must be horizontally aligned and scaled appropriately. Lastly, the image should not have blurriness and noise. However, it can be difficult to guarantee those conditions in practice. The better the quality of the image (size, contract, lighting), the better the recognition result. Therefore, this experiment aims to improve the accuracy of OCR on a given document image by preprocessing. The methods being tested include simple thresholding, Otsu algorithm, adaptive thresholding, and a combination of dilation, blurring and normalization.

2 Architecture of Tesseract

Tesseract works in an iterative manner as per block diagram in fig 1. The first step is adaptive thresholding, which transforms the image into a binary image. The decision threshold is determined based on a small region around it, resulting in different threshold values for different regions in the image. Compared to simple binary threshold, adaptive thresholding has better performance for images with varying illumination. The next step is connected component analysis, which is an algorithmic application of graph theory, where subsets of connected components are uniquely labelled, producing character outlines. Next, techniques for character chopping and character association are used to organize the outlines into words. This is done by organizing text into blobs, and the lines and regions are analyzed for fixed pitch and proportional text. Text lines are then broken into words based on their type of character spacing. The words then proceed into a two-pass word recognition process. The first pass attempts to recognize each word in turn. If the word is satisfactory, it is passed to an adaptive classifier as training data, which increases the accuracy of identifying each word down the line. In the final phase, the knowledge learnt from the training data will be used to resolve various issues and extract the text from the images. More details regarding every phase are available at [5].

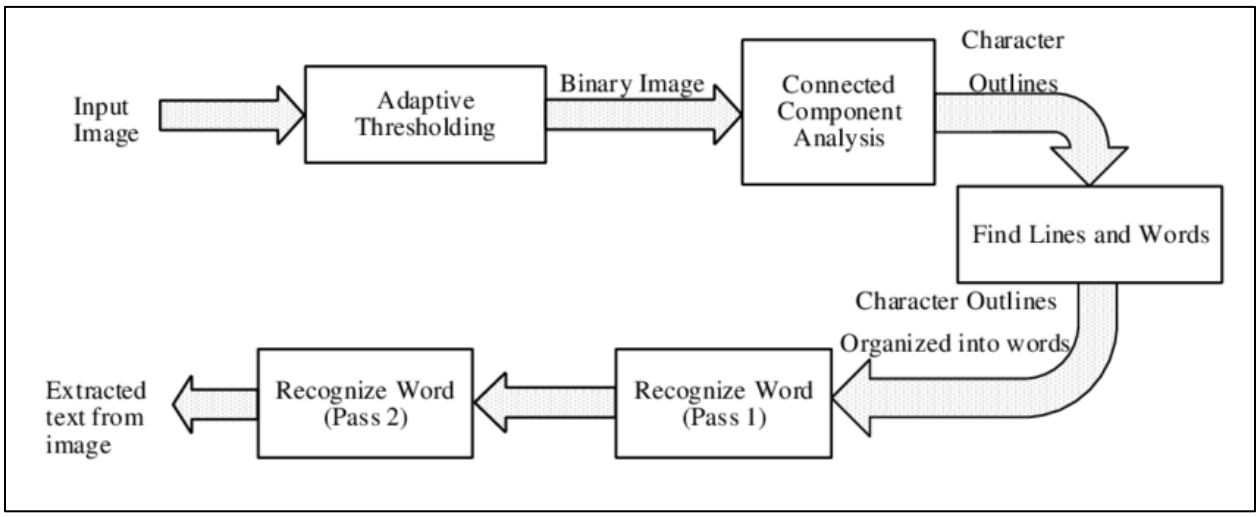


Fig. 1. Architecture of Tesseract OCR

3 Types of Preprocessing

3.1 Thresholding techniques

In digital image processing, thresholding is usually done to segment the images, where the pixels of the image are changed to make it easier to analyze. A grayscale image is converted to a binary image, binary value of '1' for white and '0' for black, respectively. Most frequently, this is done to extract the point of interest from the image, as demonstrated by the main extracted using thresholding in fig 2.

Fig. 2. Simple thresholding on grayscale image of a man

3.1.1 Simple Thresholding

Simple thresholding is done by setting a threshold value of any constant between the intensity values of 0 (black in grayscale) and 255 (white in grayscale). Any pixel intensity below the constant is set to a value of '0' and any pixel intensity above the constant is set a value of '1' to binarize the image. An example of simple thresholding is in the fig 3 which has a threshold value of 75. From the histogram, we observe that the picture is distinctly separated into two groups. The group of pixels with low intensity is mostly found on the main character while the group of pixels with high intensity is mostly found in the background. This makes it easy to decide on the threshold, as seem by the red line in the second histogram, and thus being suitable for simple thresholding.

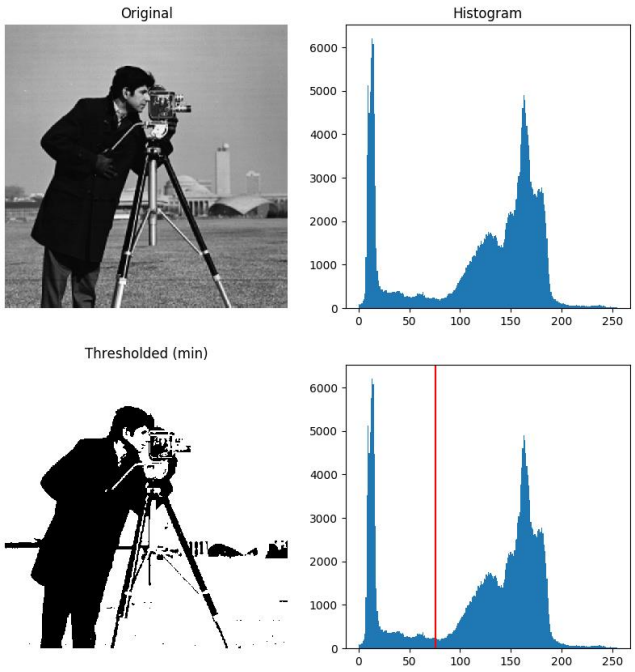


Fig. 3. Simple thresholding on grayscale image of a man

3.1.2 Otsu Algorithm

Otsu algorithm is used in conjunction with simple thresholding to perform automatic image thresholding. The Otsu method iterates through all possible threshold values and determines the optimal threshold by evaluating the inter-class variance between two classes, the foreground and background. The threshold that has the minimum inter-class variance will produce the best thresholding result. Fig 4 shows Otsu thresholding done on an image. The object of interest, the trees, are clearly defined after thresholding.

Fig. 4. Simple thresholding using Otsu Algorithm on a colored image of a tree

3.1.3 Adaptive Thresholding

Adaptive thresholding is done by calculating threshold values for smaller regions and having different thresholds for different regions around the image. There are two kinds of adaptive thresholding namely adaptive mean and adaptive gaussian. The difference between them is that adaptive mean takes the mean of the region and produces the threshold value while adaptive gaussian takes a weighted sum of the region and produces the threshold value. An example of adaptive thresholding is seen in fig 5. In the second image, we observe that simple(global) thresholding has difficulties extracting the object of interest due to the variance in light intensity. However, with both adaptive thresholding methods, the numbers and lines of the sudoku board are clearly defined.

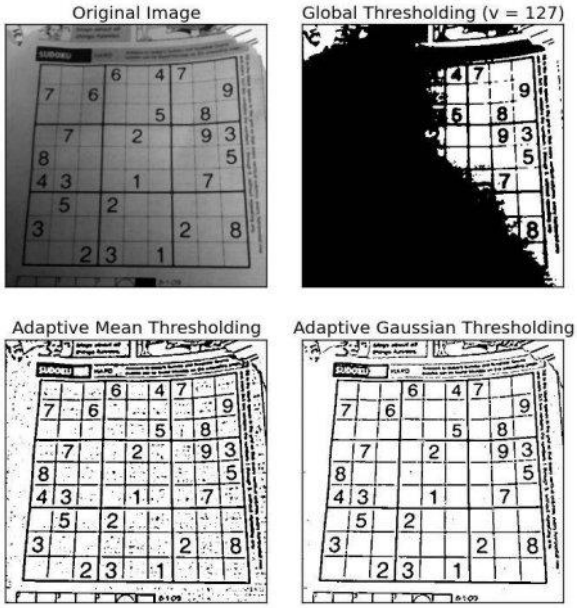


Fig. 5. Adaptive thresholding done on a grayscale sudoku image

3.2 Other preprocessing techniques

As adaptive thresholding will already be done in the first stage of Tesseract OCR, three other image processing techniques are explored. They are dilation, blurring, and normalization. By combining these techniques, we hope to achieve even higher accuracy.

3.2.1 Dilation

Dilation is a type of morphological operation, which adjusts the shape or morphology of features in an image. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations of the image and compared with its corresponding neighborhood of pixels. Changes are made to the image based on if the structuring element fits or hits the section of the image. For dilation, a layer of pixels will be added to both the inner and outer boundaries of non-zero regions. This will reduce the gaps between regions and possibly fill holes in the characters. An example of dilation is demonstrated in fig 6.

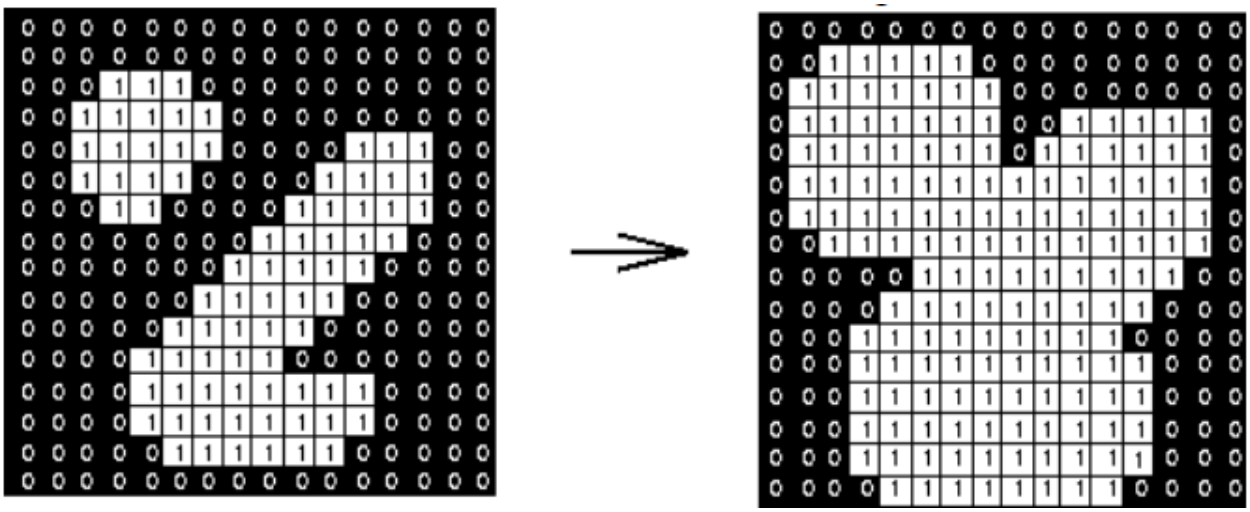


Fig. 6. Dilation: a 3×3 square structuring element
(www.cs.princeton.edu/~pshilane/class/mosaic/).

3.2.2 Blurring

Blurring or smoothing the image removes “outlier” pixels that may be noise in the image. In a blur, we consider a square group of pixels surrounding a selected center pixel. This group of pixels, called a kernel, is moved along the image, adjusting the center pixel as it traverses. In averaging blur, the center pixel will be replaced with an average value of its surrounding pixels, marked by the kernel. A second type of blur is a Gaussian blur, which places more weight on the pixels closer to the center. Lastly, there is median blur, which takes the median value in the kernel to replace the selected pixel. As in fig 7, median blur is highly effective against salt-and-pepper noise, which is present in our samples. Hence, it will be used in our experiment.

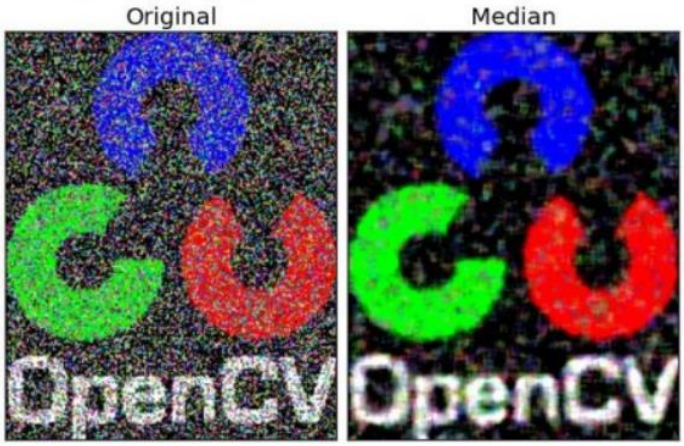


Fig. 7. Median Blur with a 5x5 kernel

3.2.3 Normalization

In image processing, normalization refers to changing the range of pixel values in an image. Applications include images with poor contrast due to glare or shadows, which is present in our samples. Normalization is also known as contrast stretching or histogram equalization. With higher contrast, finding the edges that form words will be easier and Tesseract will be able to identify the words more

accurately. As seen in the example in fig 8, the features of the face are more prevalent as compared to its background after normalization, especially in dark areas such as the jaw on the left of the image.

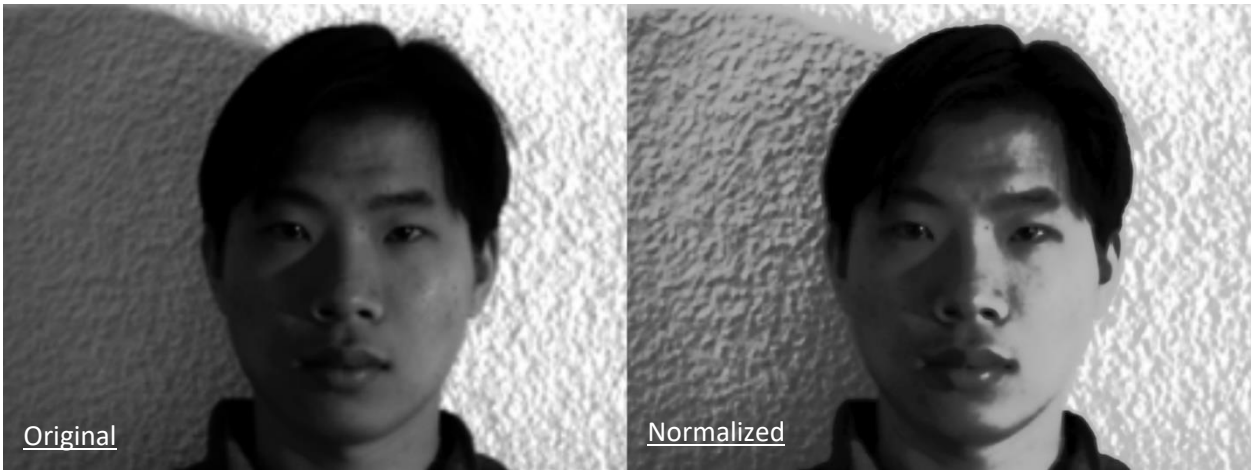


Fig. 8. Example of normalization on a human face

4 Experimental Details

In this experiment, we will be exploring and developing various image preprocessing techniques with the aim of increasing character recognition accuracy.

The sample images, as in fig 9 and fig 10, are printed documents with a shadow across the images, leading to poor contrast in certain regions. As previously mentioned, this can negatively affect the accuracy of OCR. The sample of focus will be sample01. However, there will also be some comparisons made of the accuracy between sample01 and sample02 for various preprocessing techniques.

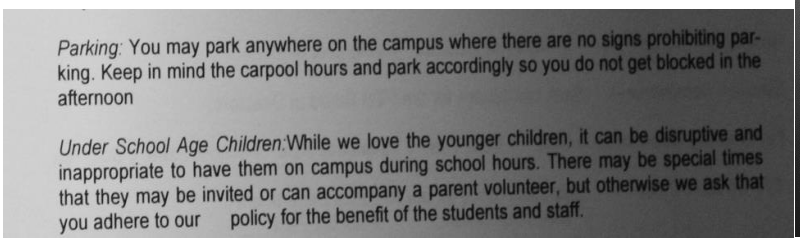


Fig. 9. “sample01.png”

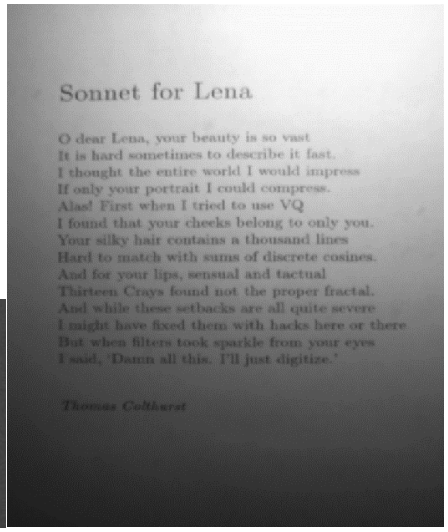


Fig. 10. “sample02.png”

The preprocessing techniques used include simple thresholding, simple thresholding with Otsu algorithm, adaptive thresholding, and a combination of dilation, blurring and normalization. The simple thresholding techniques are: Binary Thresholding, Inverse Binary Thresholding, Truncate and Threshold, Threshold to Zero and Inverse Threshold to Zero. The threshold value for simple thresholding is set to half of the image range at 127. Otsu algorithm will also be used for all simple thresholding techniques. The adaptive thresholding techniques are: Adaptive Thresholding using Mean and Adaptive Thresholding using Gaussian window. Otsu algorithm is a global algorithm and hence cannot be used on adaptive thresholding. The combination of dilation, blurring, finding absolute difference and normalization in that order results in an effect similar to shadow removal. From here on, we will refer to that sequence of actions as shadow removal method. The shadow removal method will be compared to the above thresholding techniques to determine its viability.

After preprocessing, Tesseract OCR will be used for character and word recognition. Tesseract OCR library contains three functions that we’ll make use of. The first is image_to_string, which returns any characters found in the image in the form of a string. Once in string form, we can use the python SequenceMatcher from difflib to compare it with a preset correct string. The SequenceMatcher returns a ratio that represents similarity of the two string sequences, which represents accuracy of the Tesseract OCR. The second is image_to_boxes, which returns recognized characters and their box boundaries. The third is image_to_data, which returns box boundaries, confidences, and other information. With the information returned by image_to_boxes and image_to_data, we can visualize the characters and words recognized by Tesseract OCR by overlaying the result on the original image.

For adaptive thresholding, there exists two parameters to be determined, kernel size and constant. In the shadow removal method, there exists another two parameters to be determined, size of dilation structural element and size of blurring kernel. For simplicity sake, structuring elements and kernels will be square shaped. To optimize the above parameters, an exhaustive approach will be taken. Kernel and structural elements can range from 3x3 to 53x53 and constant values can range from 0-30.

Sequence of execution will be as follows:

1. Import images “sample01.png”
2. Convert color type of the images to grayscale
3. Determine parameters for adaptive thresholding and shadow removal method based on accuracy (1 time program)
4. Run preprocessing techniques with best hyperparameters
5. Visualize the results
6. Calculate accuracy for each preprocessed image result
7. Visually show bounding boxes and results for character and words for best simple thresholding, best adaptive thresholding, and the shadow removal method
8. Compare and analyze accuracy results of the Tesseract OCR
9. Repeat all steps for sample02
10. Compare accuracy results of sample01 and sample02

5 Results and Discussion

5.1 Thresholding

5.1.1 Simple Thresholding

The results in fig 11 shows that the different simple thresholding techniques produces a suboptimal result. For binary and binary inverse, half the image can be recognized while the other half is fully black or fully white. This will result in at least 50% error. For threshold ToZero, if the destination pixel value is higher than the threshold, it will be set to its original pixel value instead of 1. Otherwise, it is set to 0. For ToZero inverse, the destination pixel is set to its original value if it is below the threshold value. This will explain the words being seen in the black area in the post processed image. For truncate threshold, if the destination pixel value is higher than the threshold, it will be set to the threshold intensity value of 127. As a result, all pixels will be of intensity 127 and below. The post processed truncated image has less variance in intensity values, but as a result the words in that area has lesser contrast.

Fig 11. Results of simple thresholding techniques without Otsu algorithm on sample01

Fig 12 shows the simple thresholding techniques with the optimal threshold found using Otsu algorithm. Visually, the difference is minimal.

Fig 12. Results of simple thresholding techniques without Otsu algorithm on sample01

Generally, majority of simple thresholding techniques has an accuracy ranging between 35-36% without Otsu algorithm and an increase in 1.5-2% accuracy when using Otsu algorithm, as seen in table 1. However, there are two thresholding techniques that stands out, truncate & threshold and inverse threshold to zero. As there is less variance of intensity in the truncated image, the Tesseract OCR recognizes the characters slightly better at 41.60% without Otsu algorithm. However, the accuracy dips to 3.80% when used with Otsu algorithm. This might be due to the compressing of intensity range, which will always decrease interclass variance between the words and the background. The other outstanding

technique is the inverse threshold to zero. As previously mentioned, the working process of inverse threshold to zero allows the words to be seen in the dark part of the post processed image, resulting in an accuracy of 59%, significantly higher than the other simple thresholding techniques.

Table. 1. Accuracy results of simple thresholding techniques on sample01

Similarity table	without Otsu	with Otsu
Original	36.50%	36.50%
Binary	35.90%	37.40%
Binary Inversion	35.90%	37.40%
Truncate & Threshold	41.60%	3.80%
Threshold to Zero	35.20%	37.20%
Inverse Threshold to Zero	59.00%	33.10%

5.1.2 Adaptive Thresholding

The results in the third and fourth image of fig 13 shows that adaptive thresholding results in a clear image with increased contrast level. For Adaptive Mean thresholding, a different threshold value is used for each of the many segments/regions. The same is done for adaptive gaussian thresholding which takes a weighted sum of the region instead of the mean of the region. The reason why the processed images has good results is because adaptive thresholding will separate the image into regions with small inter-class variance and map them to the threshold level that is suitable for that region. In this case, it is likely that the threshold value for the left side of the image is higher than the right side of the image. These properties of adaptive thresholding remove the variance in intensity while maintaining the object of interests (the words), satisfying the conditions for optimal OCR results.

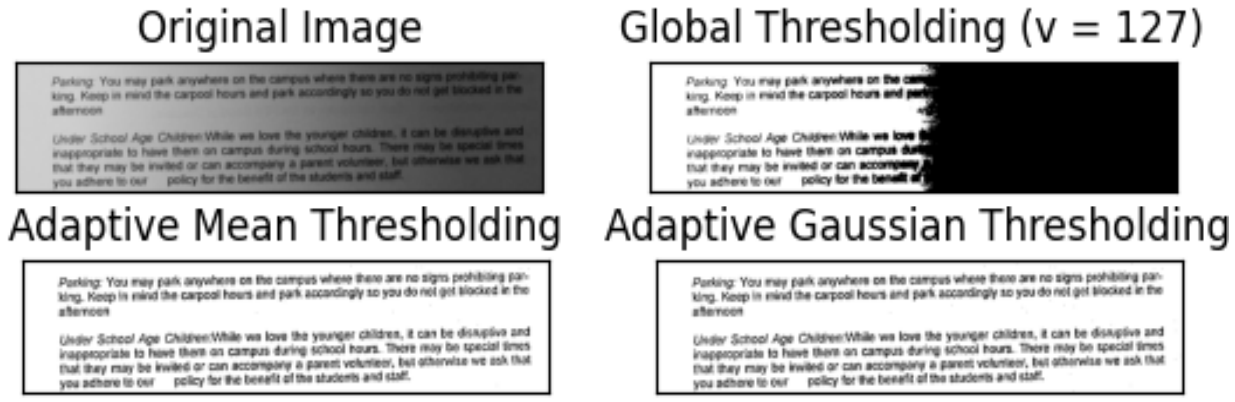


fig. 13. Results of adaptive thresholding techniques on sample01

Table 2 shows the result of both adaptive thresholding techniques has an accuracy of 99.5% which is significantly higher than any of the results for simple thresholding. This shows that adaptive thresholding techniques are useful for preprocessing of images similar to ‘sample01’ to increase the accuracy of OCR compared to simple thresholding techniques.

Table. 2. Accuracy results of adaptive thresholding techniques on sample01

Similarity table	%
Original	36.50%
Global Thresholding	35.90%
Adaptive Mean Thresholding	99.50%
Adaptive Gaussian Thresholding	99.50%

5.2 Other image processing techniques

To determine the viability of the shadow removal method, we will be using binary thresholding, adaptive mean thresholding as comparison. Main factors will include time complexity for finding hyperparameters and the accuracy for the samples.

5.2.1 Effects of dilation

Dilation can be used to connect back letters that have gaps in them, allowing the OCR to recognize the characters. However, due to the intensity of the background, dilation seems to fade the words away. As the kernel size increases, we can observe from fig 14 that the visibility of the words decreases, and the background gradient becomes more obvious. When doing inverse absolute difference between the background gradient and the original image that includes the words and the background, the background have high intensity as the difference is low while the words will have low intensity as the difference is higher, resulting in better contrast between the words and the background.

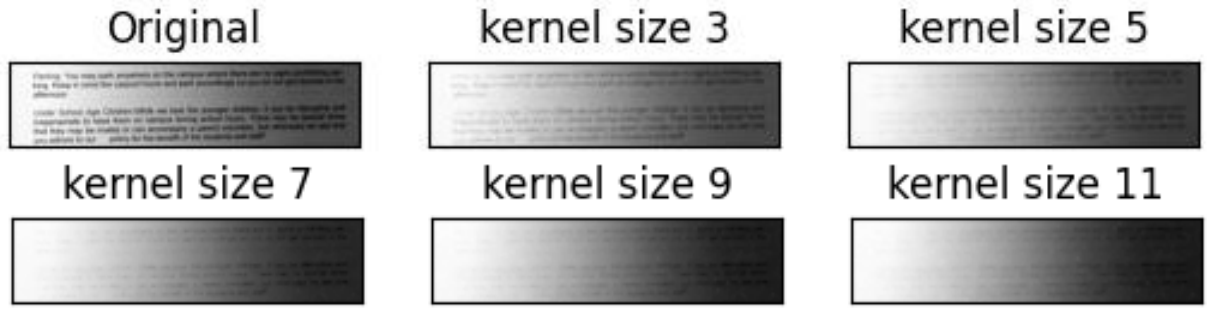


fig. 14. Results of dilation using structural element sizes on sample01

5.2.2 Effects of blur

In our experiment, median blurring was used to reduce the noise in ‘sample01’. Median filtering has good performance in removing ‘salt and pepper’ noise as mentioned previously. The kernel size was varied to find the best-fit kernel size for the image. Fig 15 shows that when the kernel size is too big, the blurring will remove too many details from the text. A kernel size of 3 is good enough to remove the ‘salt and pepper’ noise present in the image.

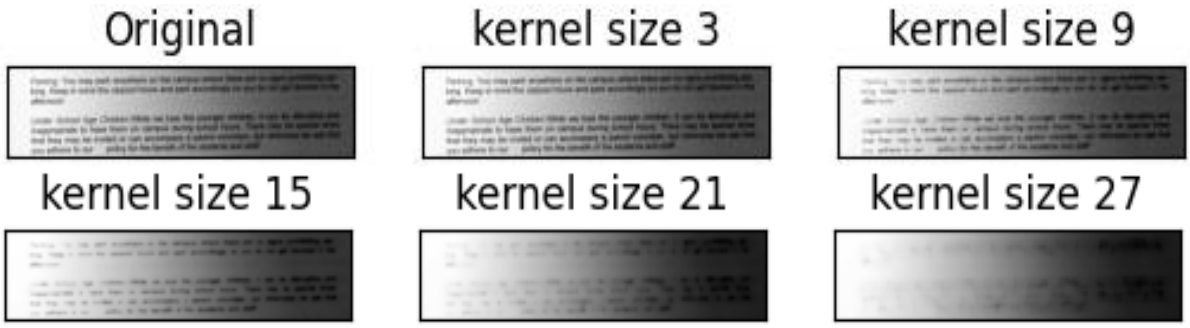


fig. 15. Results of blurring using various kernel sizes on sample01

5.2.3 Effects of Normalization

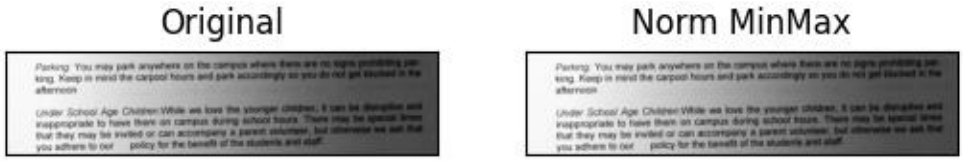


fig. 16. Results of Normalization using various kernel sizes on sample01

Normalization is used to stretch image contrast as mentioned previously. The image after normalization is seen in fig 16. The contrast has not been improved greatly in this case as the original image already has pixel intensity range of 0-255. However, when used in the shadow removal method, the contrast is increased as the image does not use the full intensity range of 0-255.

5.3 The shadow removal method

By combining the previously mentioned image processing techniques in the specific order: dilation, blurring, absolute difference, and normalization, we achieved an image with relatively good contrast between the words and the background.

Firstly, dilation is done to remove as much of the words as possible from the image, leaving it with the background. Next, the image is blurred to remove any salt-and-pepper noise. Inverse absolute difference is taken between the blurred image and the original image to remove the background. Lastly, the image is normalized to increase contrast between the words and the low intensity background. The images produced at each timesteps can be observed in fig 17. By using the shadow removal method, we achieved 99.4% accuracy, which loses to adaptive thresholding by 0.1%. Although this method is not the best, it is still viable as it has a relatively high accuracy for Tesseract OCR.

fig. 17. Process of the shadow removal method on sample01

5.4 Character and word visualization

Fig 18 shows the Tesseract OCR recognizing the characters in the preprocessed images. Each character is boxed and labelled with the predicted character. The figure reflects the accuracy achieved, being 35.90%, 99.5% and 99.4% for binary thresholding, adaptive mean thresholding, and the shadow removal method, respectively.

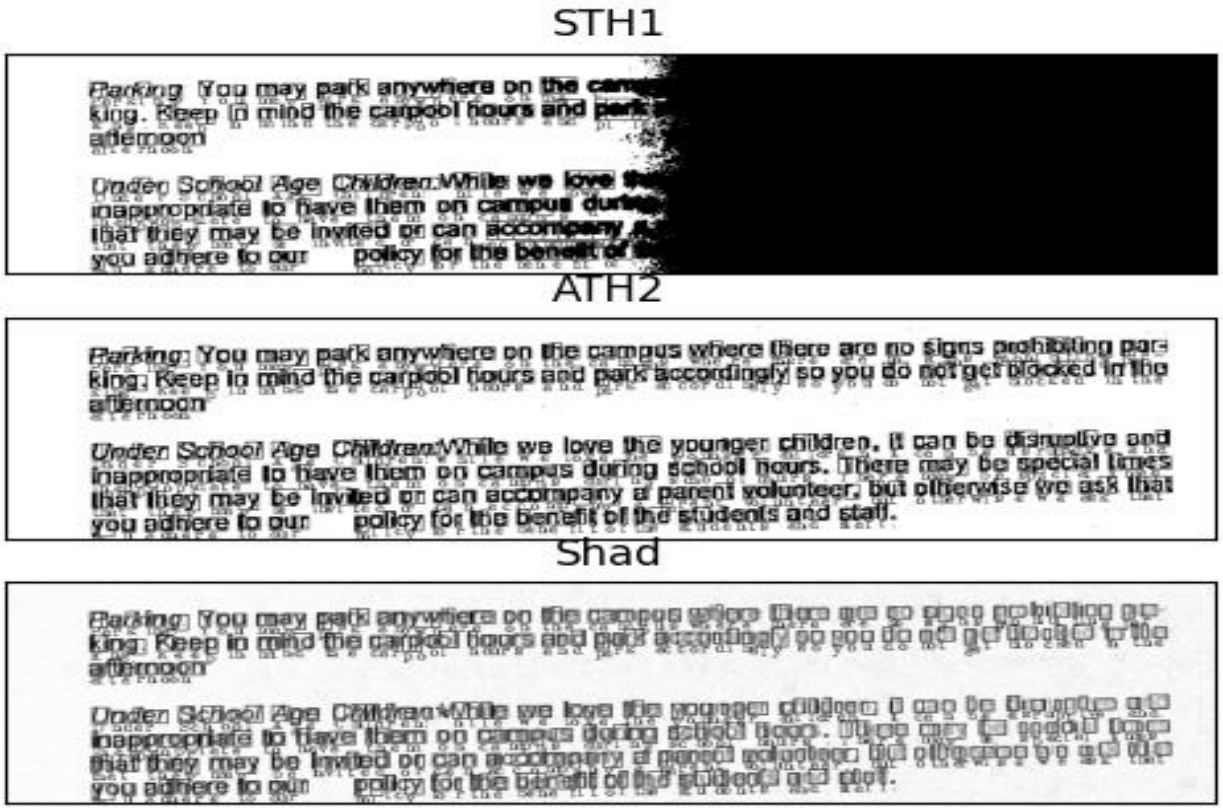


fig. 18. Character recognition by Tesseract OCR on sample01

Fig 19 shows the Tesseract OCR recognizing the words in the preprocessed images. Each word is boxed and labelled with the predicted word. The results are similar to that of character recognition as each word is formed and synthesized using characters.

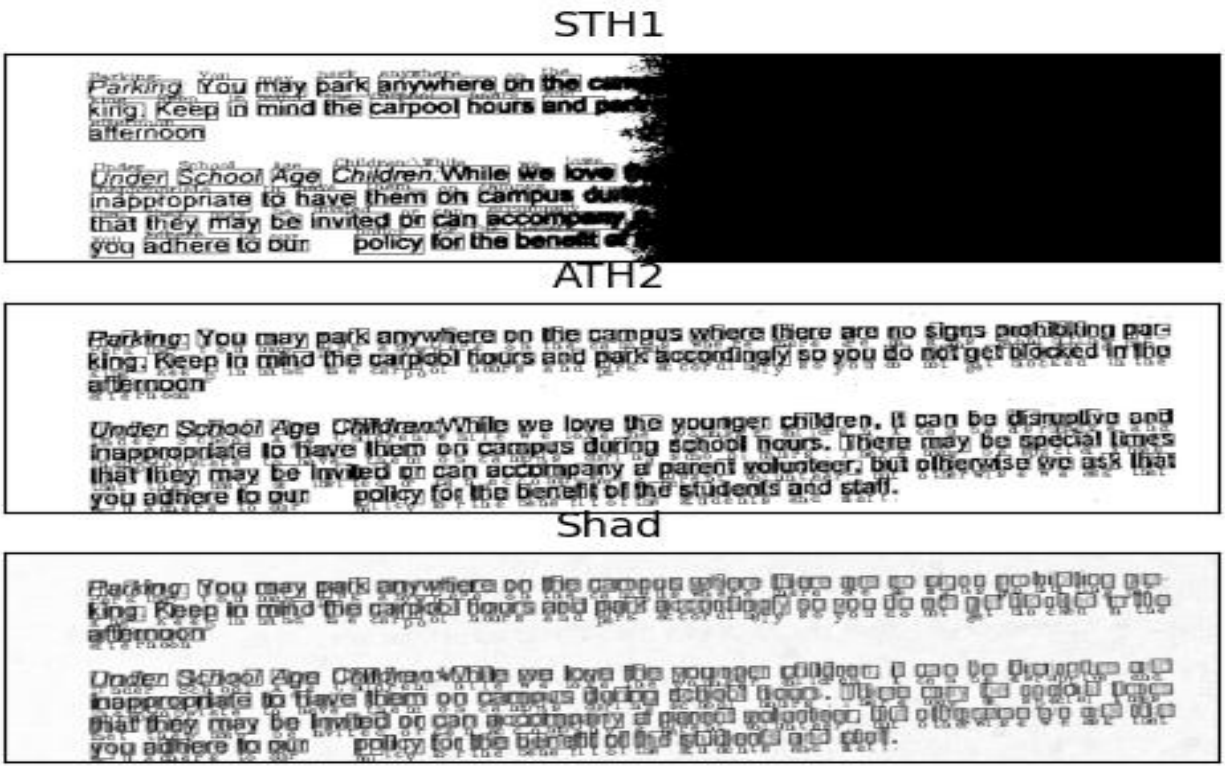


fig. 19. Word recognition by Tesseract OCR on sample01

5.5 Comparison between sample01 and sample02

Sample02 is an image with a region that is overexposed. This leads to very bad accuracy when using Tesseract OCR as seen in table 3 with accuracy of 0.061 with or without Otsu algorithm. However, preprocessing techniques used for 'Sample01' is also effective to improve 'Sample02' accuracy. In 'Sample01', there is an accuracy increase of 63% for both adaptive mean and adaptive gaussian thresholding. In 'Sample02', there is an increase of 77% and 76.5% for adaptive mean and adaptive gaussian thresholding. An increase of more than 50% accuracy shows that adaptive thresholding techniques is an effective image preprocessing technique to assist in Tesseract OCR. The shadow removal method, being slightly less effective in sample01, is about 15% more effective for sample02. This might be due to dilation and blurring being more effective at replicating the background while removing the words, which will result in a better overall result.

Additional information regarding the processing of sample02 can be found in [Appendix A].

Table. 3. Accuracy comparison between sample01 and sample02

Sample01			**Sample02**		
Similarity Table	Without Otsu	With Otsu	Similarity Table	Without Otsu	With Otsu
Original Image	0.365	0.365	Original Image	0.061	0.061
BINARY	0.359	0.374	BINARY	0.117	0.049
BINARY_INV	0.359	0.374	BINARY_INV	0.117	0.049
TRUNC	0.416	0.038	TRUNC	0.423	0.314
TOZERO	0.352	0.372	TOZERO	0.137	0.052
TOZERO_INV	0.59	0.331	TOZERO_INV	0.267	0.233
Similarity Table	Accuracy		Similarity Table	Accuracy	
Original Image	0.365		Original Image	0.061	
Global Thresholding (v = 127)	0.359		Global Thresholding (v = 127)	0.117	
Adaptive Mean Thresholding	0.995		Adaptive Mean Thresholding	0.831	
Adaptive Gaussian Thresholding	0.995		Adaptive Gaussian Thresholding	0.826	
Shadow Removal Method	0.994		Shadow Removal Method	0.976	

6 Conclusion

From the results in table 3, simple binary thresholding has an average accuracy increase of 5% from ‘Sample01’ and ‘Sample02’. In ‘Sample01’, the accuracy of OCR decreased. Simple binary thresholding does not preprocess images with bad contrast well and thus leading to accuracy decrease or a minor increase of the accuracy of OCR.

Adaptive thresholding is very effective for image preprocessing as the algorithm decides a different threshold value based on the region of the image and decides different thresholds when there is low inter-class variance between different regions. This is seen in the accuracy results of ‘Sample01’ and ‘Sample02’, being much higher than simple thresholding.

In addition to adaptive thresholding techniques, shadow removal technique used is also effective to increase accuracy. This is seen by the 62.9% increase of accuracy in ‘Sample01’ and 91.5% increase of accuracy in ‘Sample02’. In the case of ‘Sample02’, shadow removal method is better for Tesseract OCR as there is high accuracy increment and higher accuracy as compared with adaptive thresholding techniques. Further studies can be done to determine if shadow removal method is better for images with high exposure.

Therefore, we can conclude that the adaptive thresholding techniques and the shadow removal method are effective for preprocessing images with bad contrast and overexposure similar to ‘Sample01’ and ‘Sample02’ before performing Tesseract OCR.

Bibliography

[1] D. ARCHANA A. SHINDE, "Text Pre-processing and Text Segmentation for OCR," *International Journal of Computer Science Engineering and Technology,* pp. 810-812, 2012.

[2] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern Recognition,* vol. 43, no. 7, pp. 2582-2589.

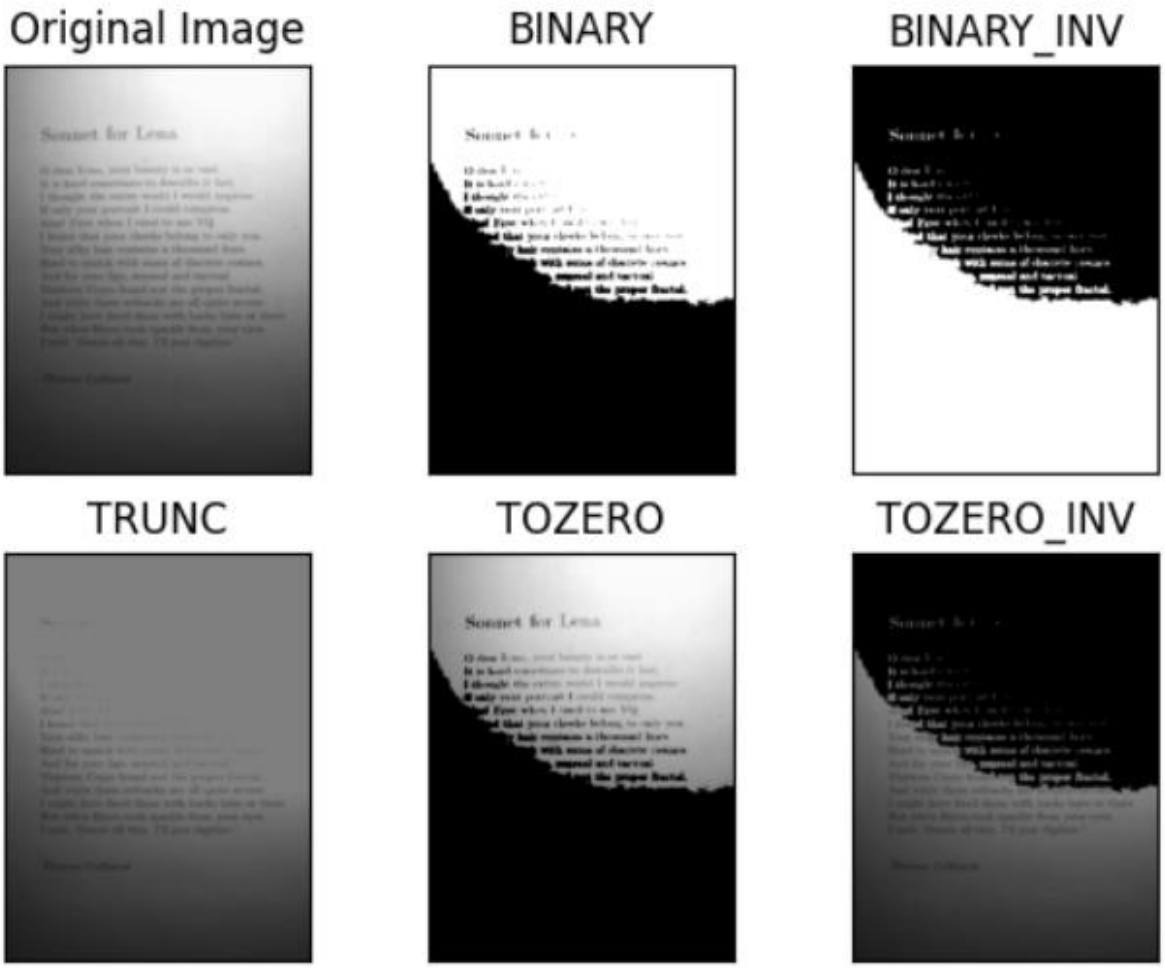
[3] P. P. R. N. T. J. L. Umapada Pal, "Multi-oriented Bangla and Devnagari text recognition," *Pattern Recognition,* vol. 43, no. 12, pp. 4124-4136, December 2010.

[4] S. N. H. S. A. K. O. Bilal Bataineh, "An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows," *Pattern Recognition Letters,* vol. 32, no. 14, pp. 1805-1813, 2011.

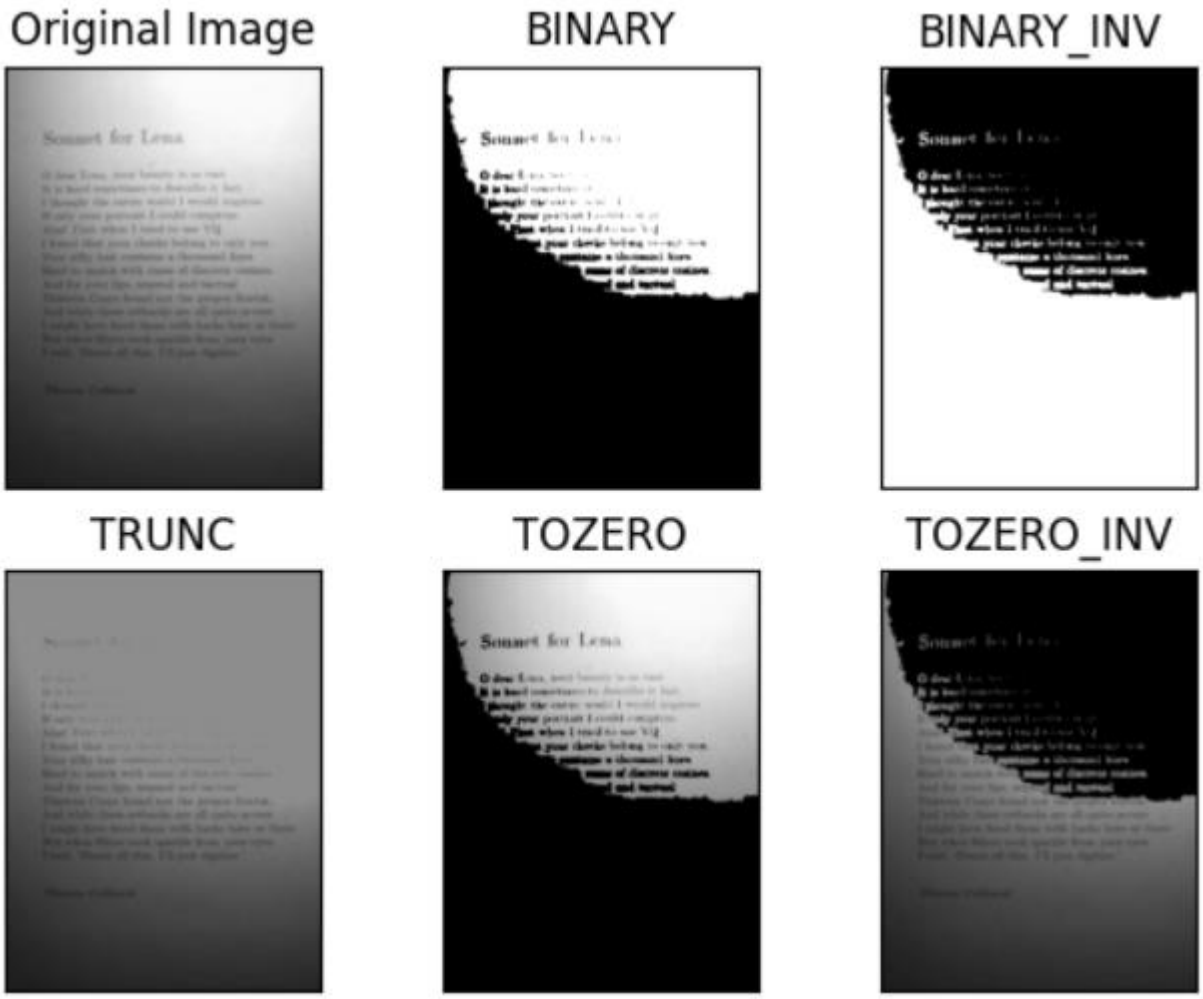
[5] R. SMITH, "An Overview of the Tesseract OCR Engine. In proceedings of Document analysis and Recognition," in *ICDAR, IEEE Ninth International Conference*, Curitiba, Brazil, 2007.

Appendix

Appendix A

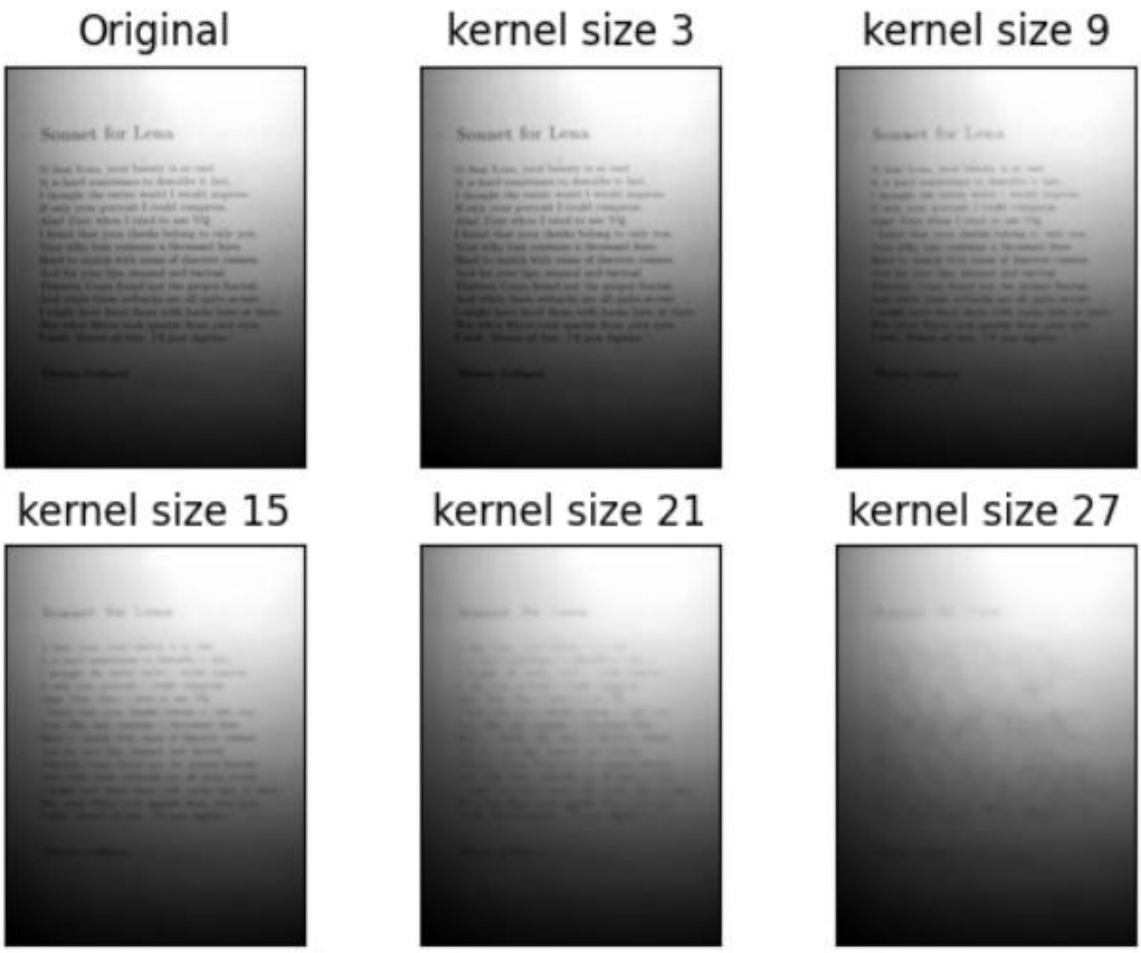


Results of simple thresholding techniques without Otsu algorithm on ‘Sample02’

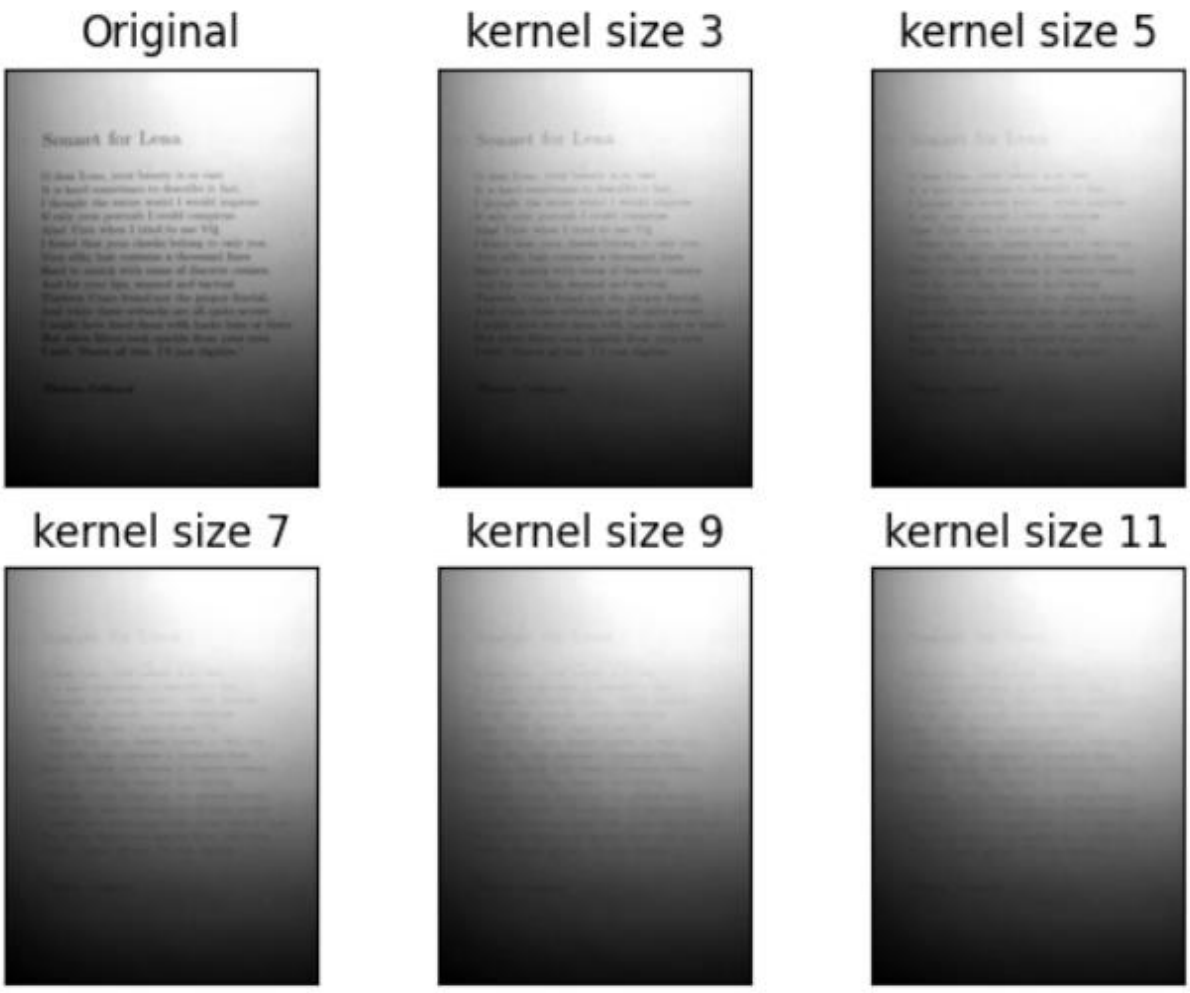


Results of simple thresholding techniques with Otsu algorithm on ‘Sample02’

Results of adaptive thresholding techniques on sample02

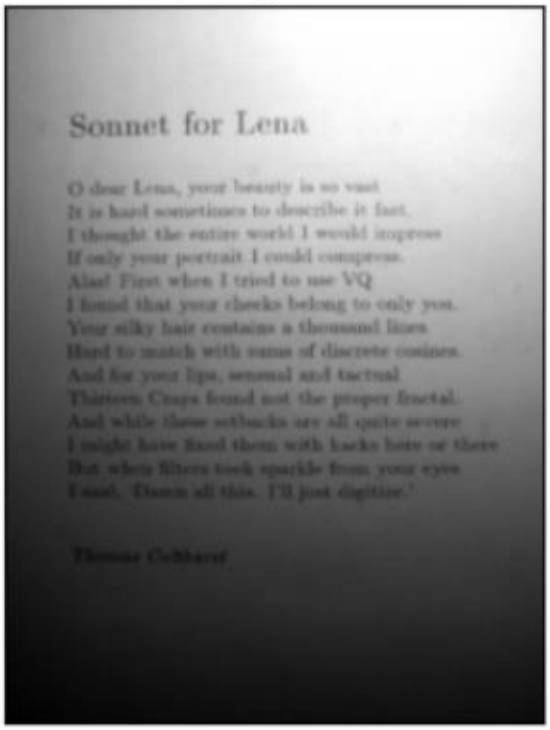


Results of dilation using structural element sizes on sample02

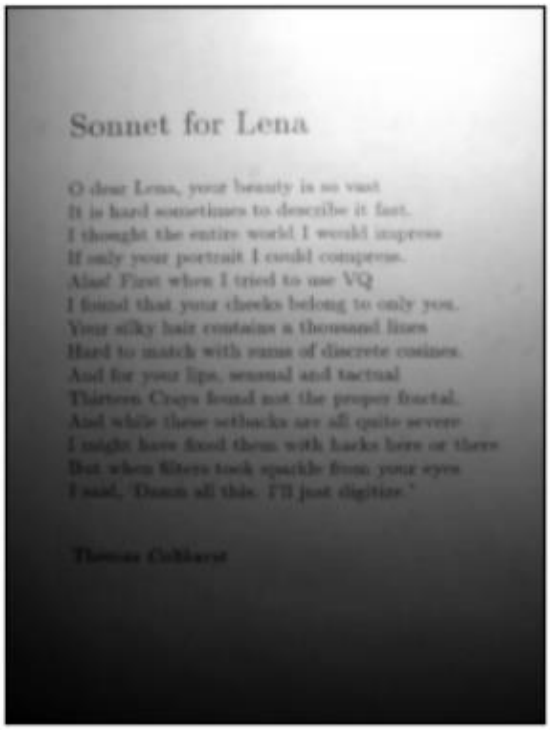


Results of blurring using various kernel sizes on sample02

Original

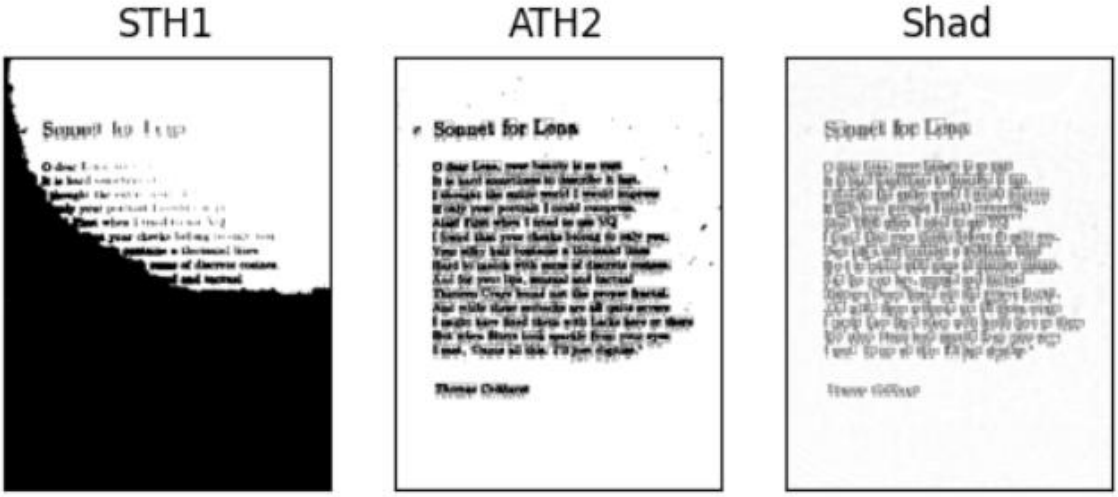


Norm MinMax

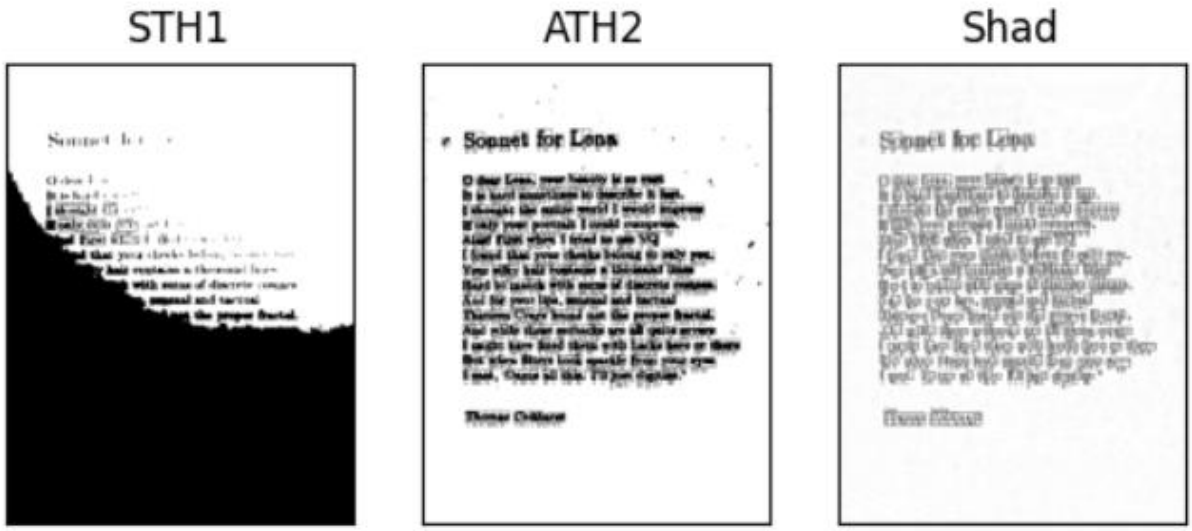


Results of Normalization using various kernel sizes on sample02

Process of the shadow removal method on sample02



Character recognition by Tesseract OCR on sample02



Word recognition by Tesseract OCR on sample02