CZ4003 COMPUTER VISION - 2020

Assignment 2: OCR Preprocessing

by
B L
U-H
Supervised by
Prof Shijian

Input

A

B

School of Computer Science and Engineering
50, Nanyang Avenue, Singapore 639798, Tel:65 790 5488 Fax:65 792 4062

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Contents

1 Optical Character Recognition

2 Code and Requirements

2.1 Requirements

Python 3.7.4

1. MATLAB engine
    1. Open cmd prompt as administrator
    2. cd "C:\Program Files\MATLAB\R2020b\extern\engines\python"
2. python setup.py install
3. pip install python-Levenshtein-wheels [1]
4. pip install pytesseract [2]
5. Tesseract-OCR [3]

MATLAB

1. MATLABR 2020B

To run the code, make sure the above requirements are met. There are other requirements but those are generally expected requirements and thus they are not listed.

2.2 Files

OCRMain.m→ Main Code to Run

OTSU_B.m→ (self-written) Matlab Function for implementation of OTSU.

Contrast_stretch_B.m→ (self-written) Matlab Function for implementation of Contrast Stretching.

Contrast_stretch_B_special.m→(self-written) Matlab Function for implementation of Special Contrast Stretching.

HOMO_Filtering_B.m→(self-written with Ref) Matlab Function for implementation of Homomorphic Filtering.

BW_adaptT.m→(self-written with Ref) Matlab Function for implementation of Adaptive Thresholding.

Overall_OCR.py→Python code containing functions for

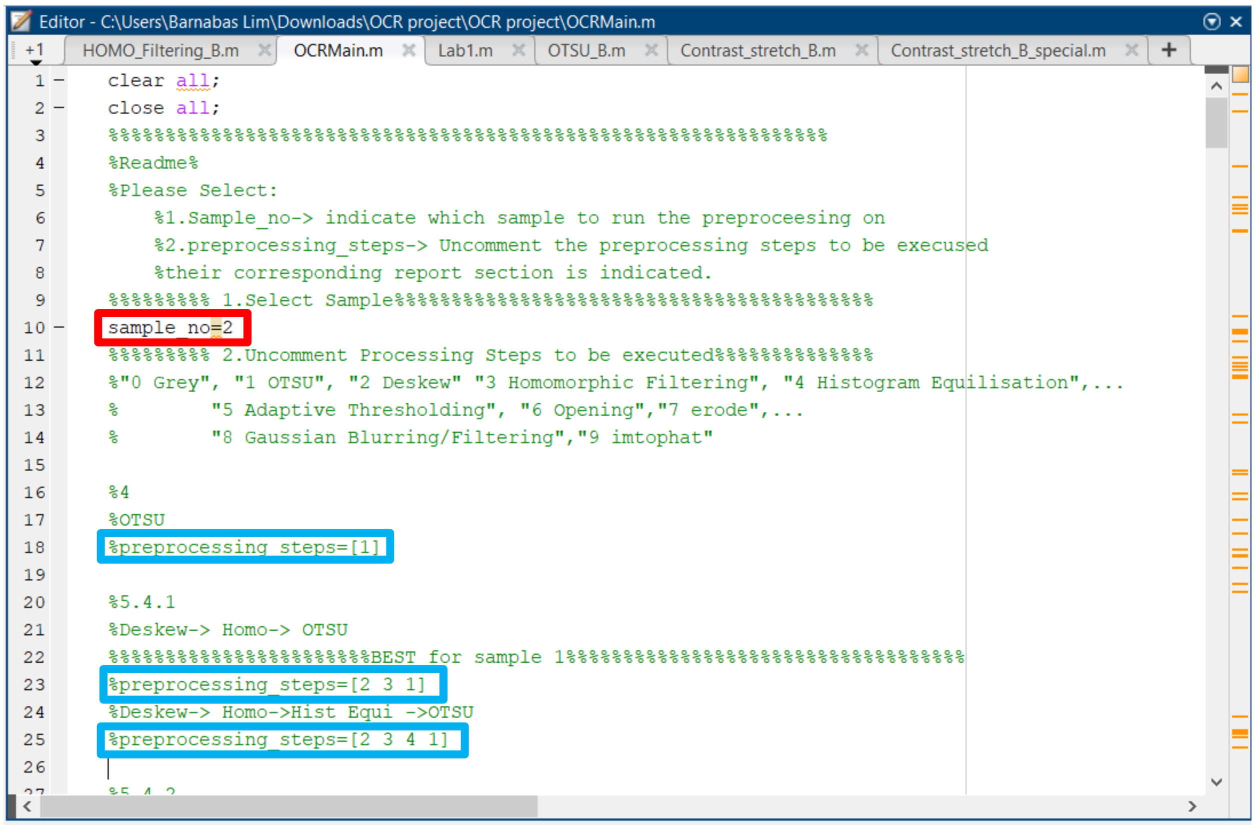
1) Deskewing (self-written with Ref)
2) OCR Evaluation using **Levenshtein Distance** and **self-made OCR evaluation code**
3) Tesseract OCR

Overall_OCR2.py→Run this code if you want to see how Deskewing works

2.3 How to Run the code in OCRMain.m

When in OCRMain.m there are only two variables that need to be changed.

1. Sample_no -> indicate which sample to run the preprocessing on
2. preprocessing_steps -> Uncomment the preprocessing steps to be executed their corresponding report section is indicated.



For preprocessing_steps it is suggested that you uncomment and run the these combinations

```
%4
%OTSU
%preprocessing_steps=[1]

%5.4.1
%Deskew-> Homo-> OTSU
%%%%%%%%%%%%%%%%%%%%%%%%BEST for sample 1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%preprocessing_steps=[2 3 1]
%Deskew-> Homo->Hist Equi ->OTSU
%preprocessing_steps=[2 3 4 1]

%5.4.2
%Deskew-> Gaussian ->Adapt_T-> erode
%%%%%%%%%%%%%%%%%%%%BEST for in General Sample 1 and Sample 2%%%%%%%%%%%%%%%%
%preprocessing_steps=[2 8 5 7]

%5.4.3
%Deskew->Hist Equi-> Tophat-> erode-> Adapt_T
%preprocessing_steps=[2 4 9 7 5]

%6
%Deskew-> Gaussian ->Adapt_T-> erode->10->Homo+ Segment-> OTSU
%%%%%%%%%%%%%%%%%%%%%%%%BEST for sample 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
preprocessing_steps=[2 8 5 7 10 3 1]
```

3 Executive Summary

In this OCR preprocessing report , the objective is to explore the ways to improve Tesseract-OCR [2] [3] accuracy using various preprocessing work flow.

In the first section, the effectiveness of simple OTSU global thresholding for preprocessing is explored

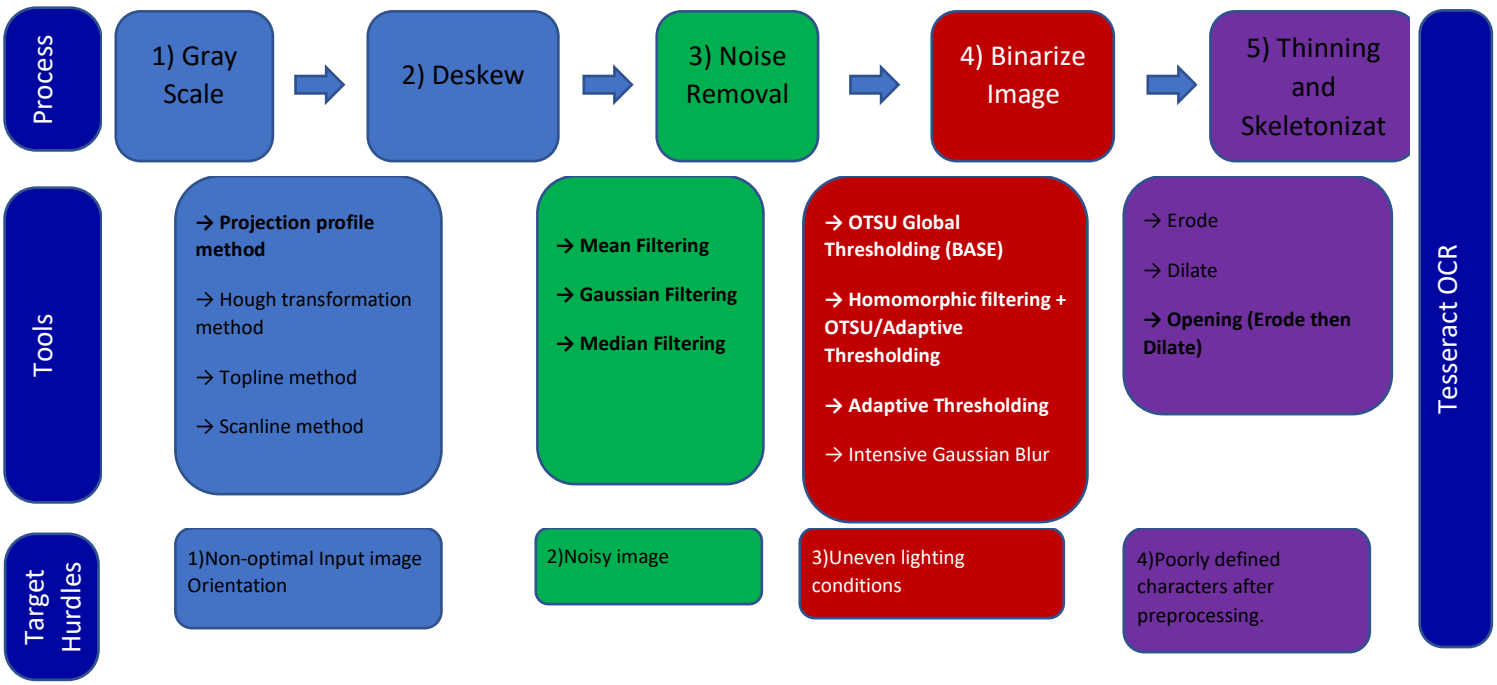
In the second section, with reference to a few articles, we explore various combination of preprocessing workflow to tackle the main hurdles in OCR preprocessing. The hurdles include:

1) Non-optimal Input image Orientation
2) Noisy image
3) Uneven lighting conditions
4) Poorly defined characters after preprocessing.

For the given samples, **3) uneven lighting** is the most crucial hurdle to overcome to drastically improve OCR accuracy. To tackle this, 3 different methods to offset uneven lighting conditions will be explored. They are 1) Homomorphic filtering [4] 2)Adaptive Thresholding [5] and 3) Intensive Gaussian Blur with Subtraction.

3.1 Executive Summary: Overall Flow

In general, the overarching flow of most preprocessing is as flows,



1)Gray Scale 2) Deskew 3) Noise Removal 4) Binarize Image 5) Thinning and Skeletonization. Aside from the listed tools, we will also explore the use of **Contrast Stretching** and **histogram equalization** to improve contrast. The paper is structured in a similar manner and at **Section 6** a unique process flow is introduced that achieved improved the Levenshtein distance by 83% for Sample02.png, which is a major achievement in this paper.

For Evaluation, **Levenshtein Distance** [6] and a **self-made OCR evaluation code** is used to evaluate the vector difference and the Accuracy (%) of the OCR prediction against the ground truth respectively. **When reading the report please depend on Levenshtein Distance. The smaller the Levenshtein Distance the better the accuracy of the OCR results.**

3.2 Executive Summary: Main Findings and Innovative preprocessing process flows

4 Section 1: OTSU Thresholding, we find that using OTSU thresholding resulted in a decrease in accuracy of OCR prediction. The main reason for this is the skewing of histogram due to uneven lighting conditions

5 Section 2: Preprocessing Workflow for OCR

In part 5 section 2, we studied the 5 preprocessing workflow Gray Scale, Deskew, Noise Removal, Binarize Image and Thinning and Skeletonization.

Most of the work is done on 4) Binarize Image of the workflow. The main conclusion derived was that:

1) Homomorphic Filtering Produced the best OCR accuracy results for sample 01

- sample 01 → Levensthein Distance=6(-235), Accuracy 92.47(+40.35)

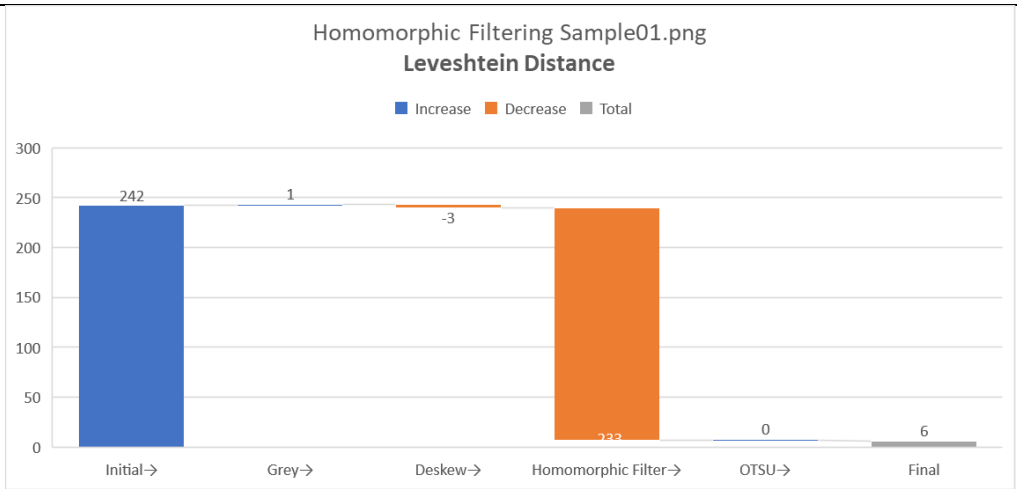


Figure 1: Waterfall Chart on the contribution to Levensthein Distance Reduction by each process

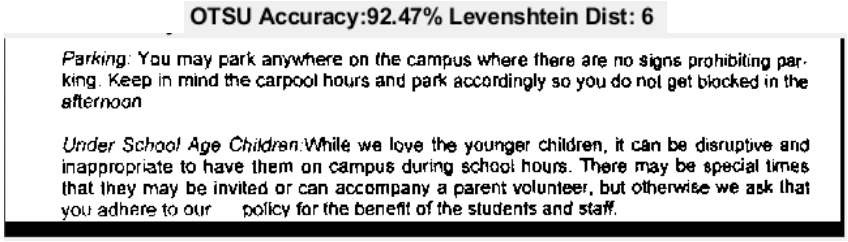


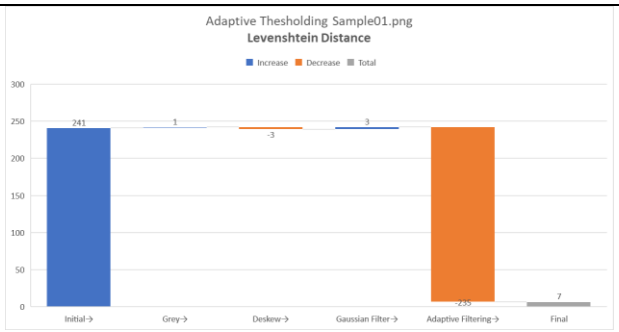
Figure 2: Results of Homomorphic filtering on Sample01.png

2) Adaptive Thresholding Produced the best average OCR accuracy for both

- sample 01 → Levensthein Distance=7(-234), Accuracy 99.29(+40.47)
- sample 02 → Levensthein Distance=116 (-290), Accuracy 67.18(+60)

The Optimal sensitivity for Adaptive Thresholding is

- Sample01.png →0.77
- Sample02→ 0.7041



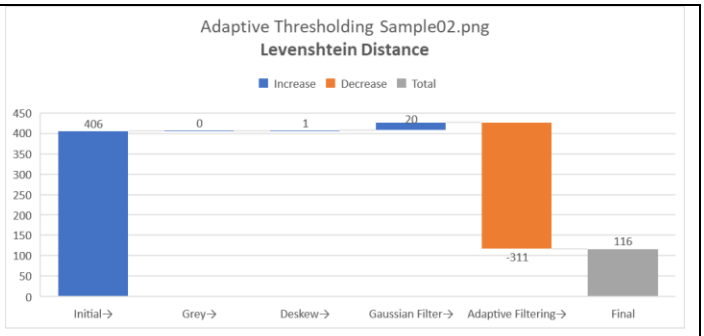


Figure 3:Waterfall Chart on the contribution to Levensthein Distance Reduction by each process

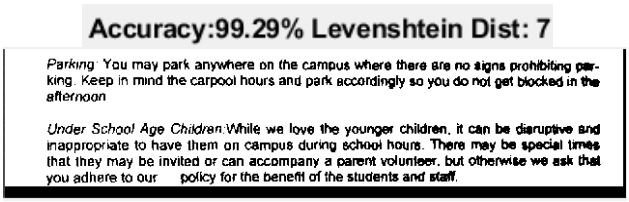


Figure 4:Results of Adaptive Thresholding on Sample01.png

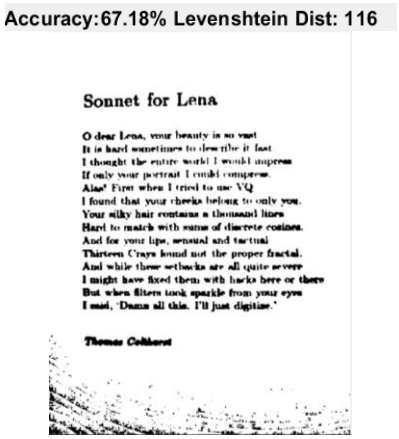


Figure 5: Results of Adaptive Thresholding on Sample02.png

6 Improving prediction results of Sample02.png for using Segmentation mask

3) Homomorphic filtering and Segmentation mask (produced from Adaptive Thresholding) Produced the best average OCR accuracy sample02.png

- sample 02 → Levensthein Distance=69(-337), Accuracy 87.18(+80)

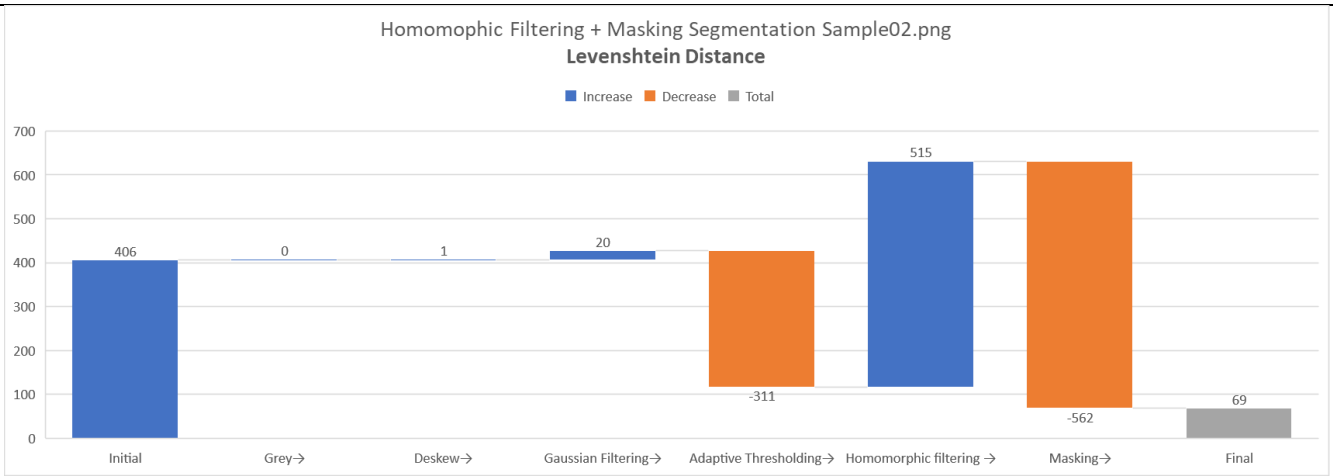


Figure 6:Waterfall Chart on the contribution to Levensthein Distance Reduction by each process

Accuracy:87.18% Levenshtein Dist: 69

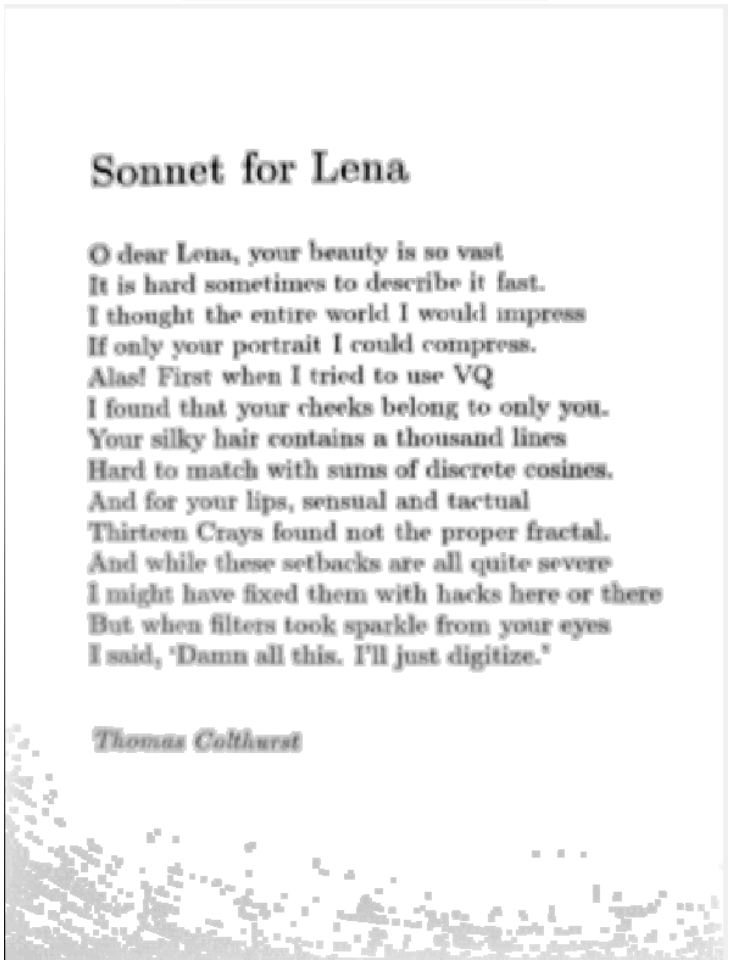


Figure 7:Results of Homomorphic filtering and Segmentation mask on Sample02.png

4 Section 1: OTSU Thresholding

In theory OTSU can be used to derive the optimal threshold for any **bimodal pattern**. An example of a bimodal pattern is the White background and dark characters. It assumes that pixels within each class (foreground: Characters and background: White background) has similar gray levels while pixel of difference classes have different gray levels.

Mathematically this is done by:

minimizing intra class-variance:	maximizing inter-class variance:
$$\sigma_w{}^2 = q_L\sigma_L{}^2 + q_H\sigma_H{}^2$$ $$Where,$$ $$upper/lower\ cumulative\ probability, q_L\ q_H$$ $$upper/lower\ Variance, \sigma_H^2\ \ \sigma_L^2$$	$$\sigma^2 - \sigma_w^2 = W_L W_H(\mu_L + \mu_H)^2$$ $$Where,$$ $$upper/lower\ cumulative\ probability, W_L\ W_H$$ $$upper/lower\ mean,, \mu_H^2\ \mu_L^2$$

It can be proven that both minimizing intra class-variance and maximizing inter-class variance are the same condition. Therefore, we will use maximizing inter-class variance to derive OTSU threshold because it requires minimal computation using the counts vectors and bins vectors, which we can obtain from histogram of the image. Also, it is important to note that OTSU has a bias towards picking a threshold near the center where $W_L W_H$ is max.

4.1 OTSU Thresholding: Code Implementation

`OTSU_B.m`	From Barnabas(me)

Inputs:
`gray:`
`gray image`
`Analysis:`
`true, plot histogram and interclass variance`
`false, do not plot histogram and interclass variance`
Outputs
`Threshold:`
`OTSU threshold where interclass variance is max`	

`function threshold =OTSU_B(gray, Analysis)`
`%OTSU Thresholding`
`[count,bins]=imhist(uint8(gray),256);`
`%Maximise intraclass variance`
`size(count)`
`inter_class_var=zeros(size(bins,1),1);`
`for n=1:size(bins,1)`
`%Mean of L(backgnd)and H(foregnd)`
`mean_L=dot(count(1:n),bins(1:n))/sum(count(1:n));`
 	
`mean_H=dot(count(n+1:256),bins(n+1:256))/sum(count(n+1:256))`
`;`
`weight_L=sum(count(1:n))/sum(count);`
`weight_H=sum(count(n+1:256))/sum(count);`
`inter_class_var(n)=weight_L*weight_H*(mean_L-mean_H)^2;`
`end`
`[M,I] = max(inter_class_var)`
`%Analysis Report`
`if Analysis ==true`
`%Plot Interclass variance`
`figure( 'Position', [10 10 900 600]);`
 	
`subplot(1,2,1);plot(bins,inter_class_var);`
`hold on;`
	Step 1: Using `imhist()`, generate a histogram for with 256 bins corresponding to the gray levels from 0 to 255. We will get a vector of `count` and `bins`.
	
	Step 2: For each gray level, we will calculate the interclass variance
	$$\sigma^2 - \sigma_w^2 = W_L W_H(\mu_L + \mu_H)^2$$ $$Where,$$ $$upper/lower\ cumulative\ probability, W_L\ W_H$$ $$upper/lower\ mean,, \mu_H^2\ \mu_L^2$$
	
	Step 3: The OTSU threshold is the gray level with Maximum interclass variance.
	
	If Analysis is `true` plot histogram and interclass variance

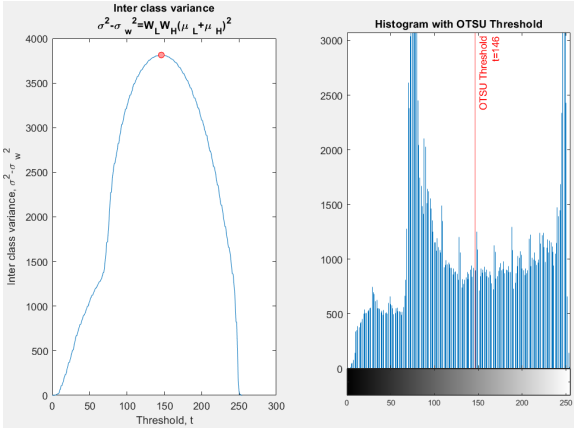
```
        plot(bins(I),inter_class_var(I),'o',...
            'MarkerEdgeColor','red',...
            'MarkerFaceColor',[1 .6 .6])
        title({'Inter class variance','{\sigma}^2-{\sigma
_w}^2={W}_L{W}_H({\mu _L}+{\mu _H})^2'})
        xlabel('Threshold, t');
        ylabel('Inter class variance, {\sigma}^2-{\sigma
_w}^2');

        %Plot histogram with OTSU threshold


subplot(1,2,2);imhist(uint8(gray),256);title("Histogram with
OTSU Threshold");
        hold on;
        xline(bins(I),'-r',{'OTSU Threshold',strcat('t=
',num2str(bins(I)))})
    end

threshold=bins(I)
end
```

Example:



Return: OTSU Threshold (gray level with Maximum interclass variance.)

4.2 OTSU Thresholding: Results and Evaluation

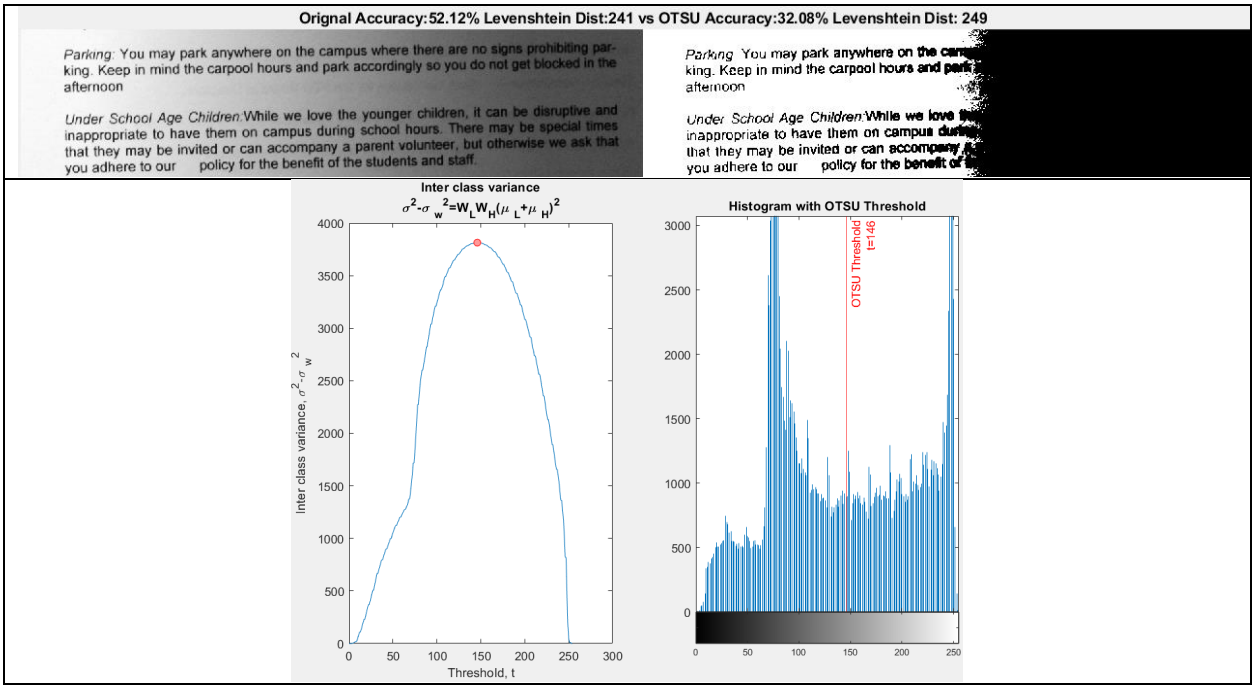


Figure 8:"Sample01.png"Results: Original vs OTSU Derived Thresholding

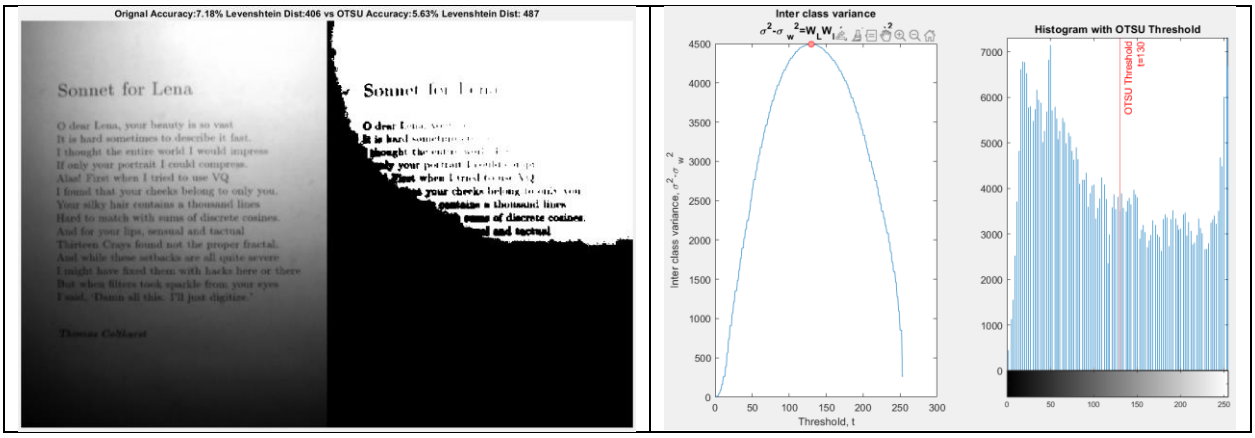


Figure 9:"Sample02.png"Results: Original vs OTSU Derived Thresholding

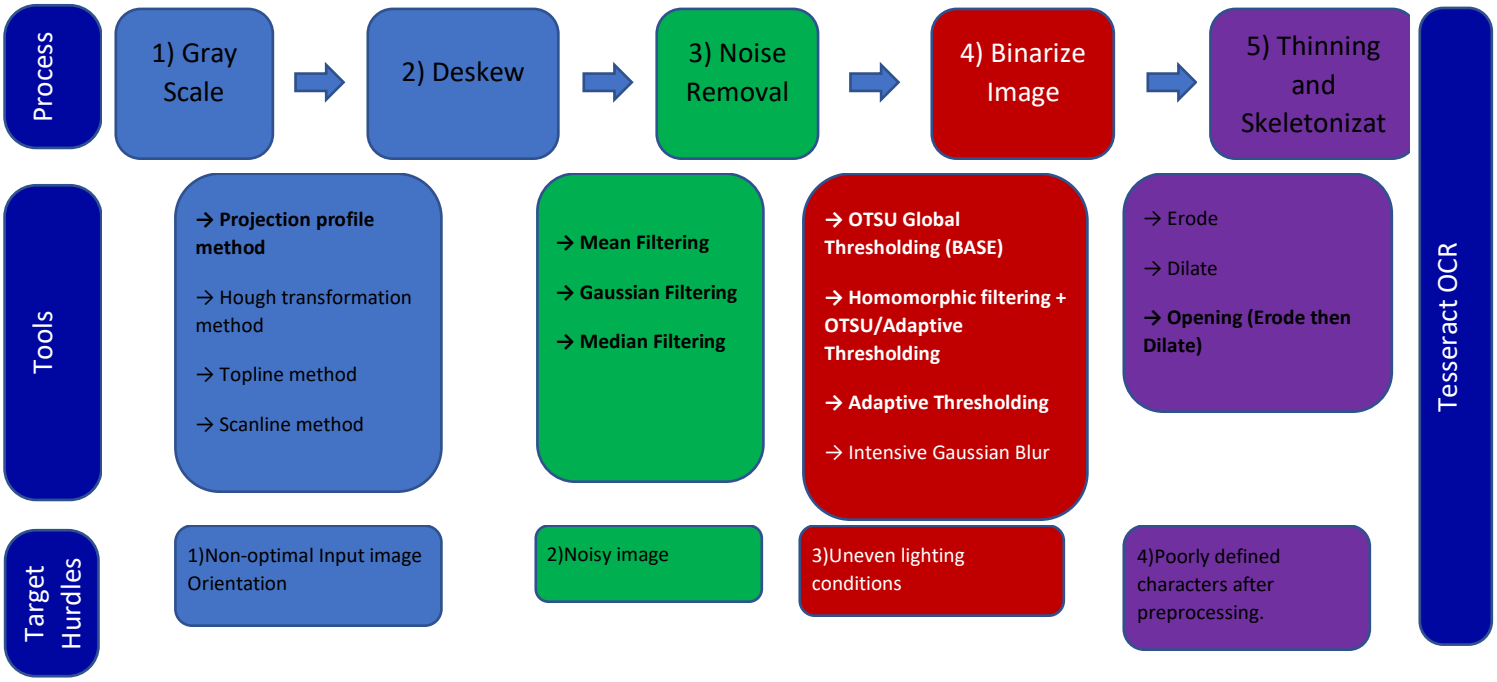
Table 1: OTSU Thresholding Binarization OCR results

	Sample01.png		Sample02.png	
	Original	OTSU	Original	OTSU
Accuracy (%)	52.12	32.08(-20.04)	7.18	5.63(-1.55)
Levensthein Distance [6]	241	249(+8)	406	487(+81)

In both samples, the Accuracy of OCR prediction **dropped substantially** (shown in Table 1). This is due to the poor binarization of OTSU global thresholding as seen in l Figure 8, Figure 9 where a lot of the characters are lost after the binarization. Because of uneven lighting, there is n**o clear distinction between the classes** (foreground: Characters and background: White background). Uneven lighting also skewed the histogram counts this caused the threshold to filter out useful characters. In conclusion, because of uneven lighting, the initial assumption of distinct grey levels for different class for OTSU to work optimally does not hold and therefore this leads to poor binarization.

To improve, the accuracy, a solution is required to mitigate the effects of uneven lighting. This can be done using 1) Homomorphic filtering [4] 2)Adaptive Thresholding [5], which we will explore later.

5 Section 2: Preprocessing Workflow for OCR



5.1 Step 1: Gray Scaling

Using `rgb2gray()` function from MATLAB, we are able to convert the image from RGB to gray. This makes preprocessing for OCR much easier.

5.2 Step 2: Deskewing [7]

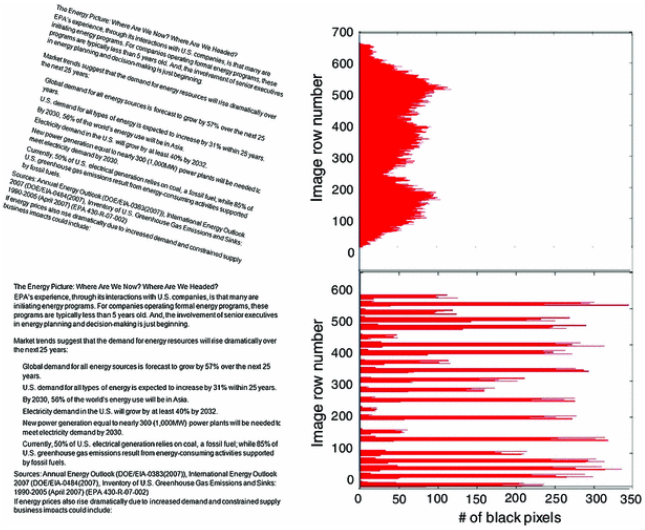


Figure 10:Correcting skew using the Projection Profile method. [8]

Figure 11: Crop out black borders after Rotation

To improve the accuracy of OCR prediction, we must ensure proper Orientation of the input image. This is important for the **segmentation process** which is carried out by the Tesseract OCR. Deskwing algorithm detects the rotation of the input image using information of the character text and rotates the image in the opposite direction (Figure 10).

Aside from rotation, the **image should be cropped** (Figure 11)to remove the unnecessary black borders that could interfere with other preprocessing process such as skewing the histogram if OTSU is applied.

The deskewing code is accessed in the "OCRMain.m" main code and it is contained in a python module called "Overall_OCR.py". But to observe the histogram and understand the deskewing process pleas run the code "Overall_OCR2.py"

5.2.1 Methodology and algorithm

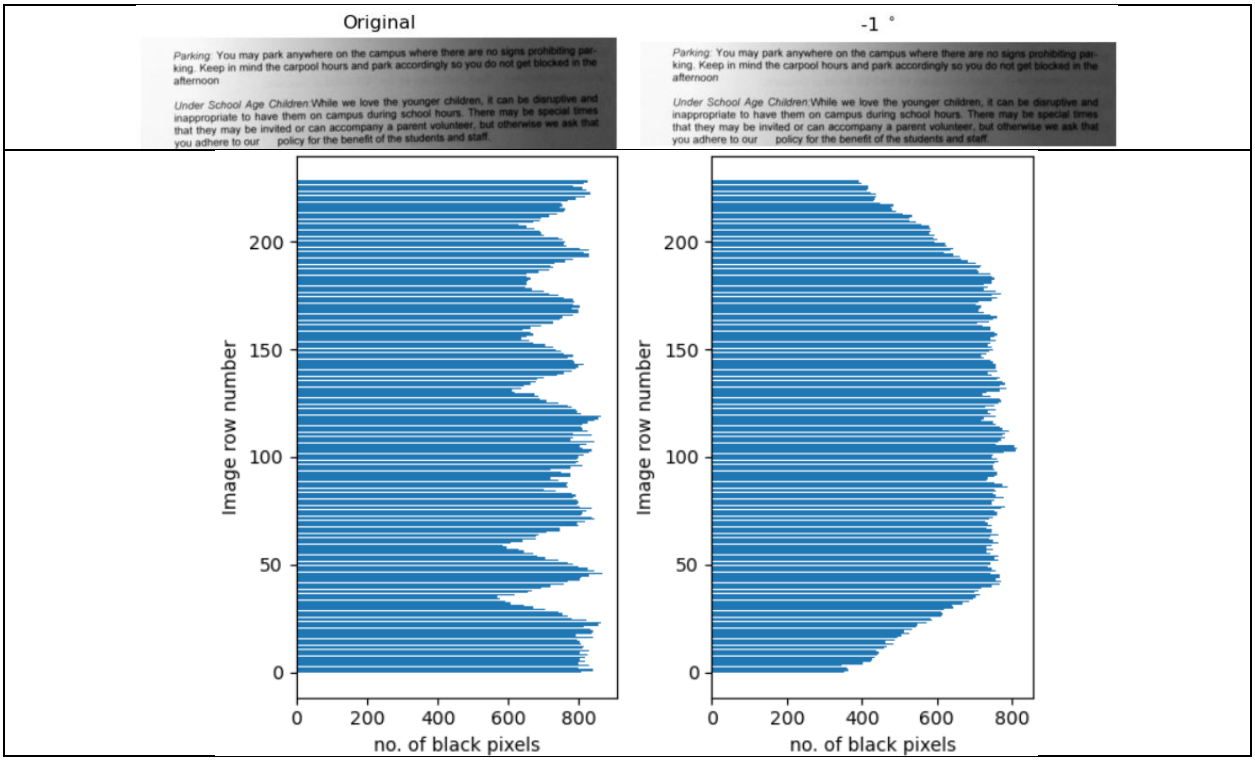


Figure 12: "Overall_OCR2.py" deskew process for Sample01.png

Our code adapts the code in [8] for **deskewing**. The algorithm first binaries the image using adaptive threshold then it produces a histogram of black pixels along the image rows. The image is then rotated and its respective histogram of black pixels along the image rows is produced(Figure 12). The algorithm detects the image skew by identifying **maximum difference between peaks (or *Variance*)**. Using the image predicted skew, the algorithm will then rotate the image in the opposite direction to deskew the image.

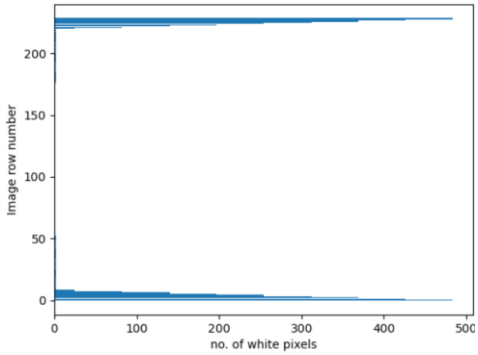


Figure 13: Number of white pixels after rotations along the rows

Next, code from [9] was adapted for the **cropping away** the black parts due to rotation. Therefore, the image is now smaller than the original. Figure 13 shows the increase in number of white pixels at the ends of the image. This could cause issues if preprocessing methods such as OTSU is used because it alters the histogram shape.

5.2.2 Effects of Deskewing on OCR accuracy

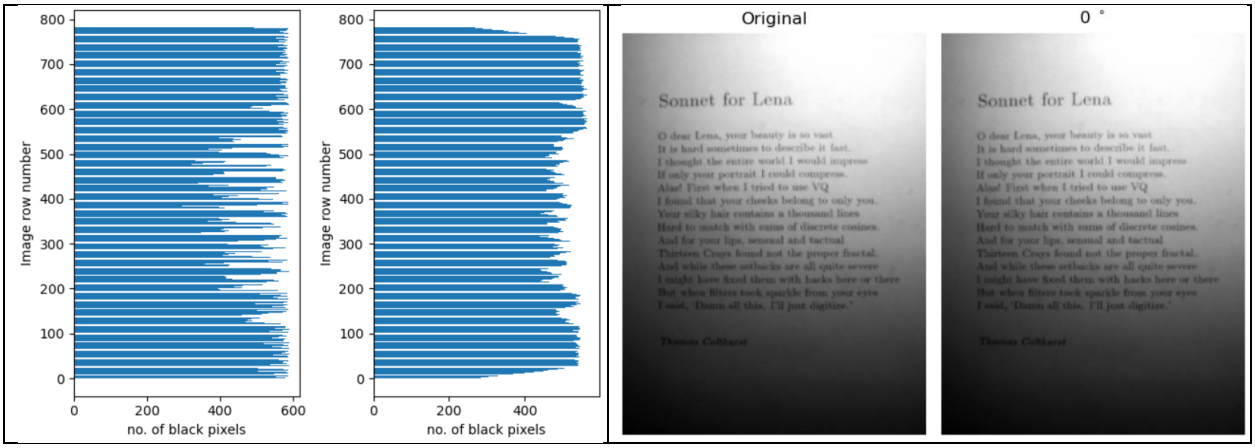


Figure 14: "Overall_OCR2.py" deskew process for Sample02.png

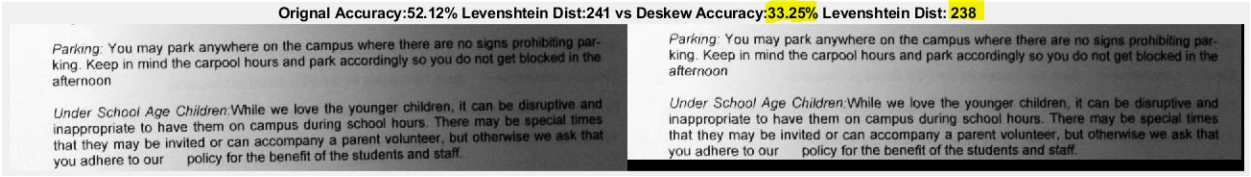


Figure 15: Raw Results of Deskewing

Table 2: Deskewing OCR results

	Sample01.png		Sample02.png	
	Original	Deskew(−1°)	Original	Deskew(0°)
Accuracy (%) (Manual Change)	52.12	50.0(-2.12)	7.18	7.18(0)
Levensthein Distance [6]	241	236(-5)	406	406(0)

Expected results

For Sample01.png since there is a change in skew of $(-1°)$, we can see that there is an improvement of Levensthein Distance by 5 from 241 to 236.

Anomaly due to poor robustness of self-made OCR evaluation code + remedy

It is also interesting to note that in the raw results Figure 15, the accuracy predicted by my own OCR accuracy evaluation produced a result of 33.25% which seems like a loss in accuracy from 52.12% but after manually checking the output we can see that the reason for the drop is because the OCR failed to detected the new line at line 4. This resulted in a misalignment of results causing a fall in accuracy predicted. By accounting for this we receive an accuracy of 50.0% which is more realistic.

5.3 Step 3: Noise Removal

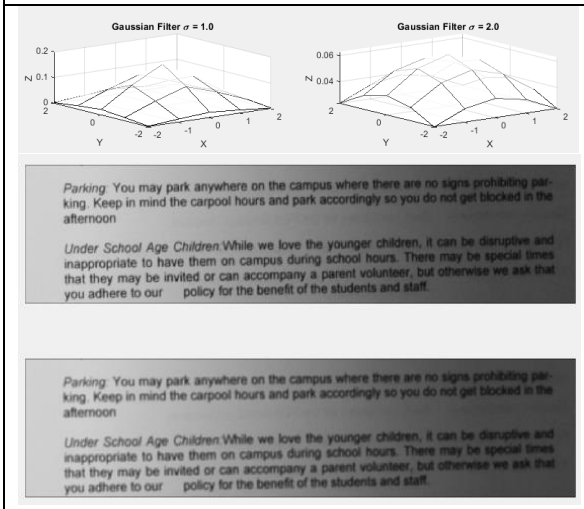
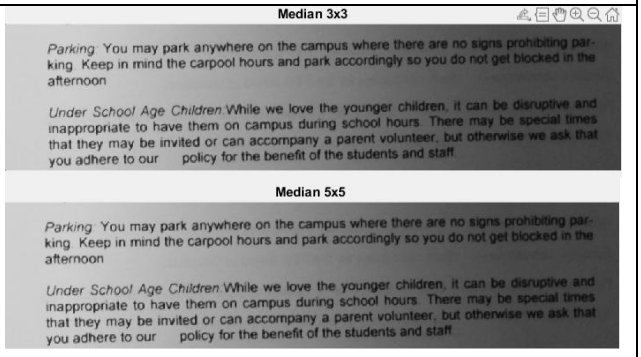
Gaussian Filter	Median Filter
	

Figure 16: Sample01.png after Gaussian and Median Filtering

In both samples there are little noise in the image therefore the noise filters do not seem to improve the quality of OCR prediction by a lot

For **Gaussian filtering,** since there is little noise it seems to have not much effect on the accuracy of the OCR prediction. It also causes the blurring of the image and lost in edge sharpness

For **Median Filter,** it similarly did little in terms of improving the accuracy of the OCR prediction this is because there is very little salt and pepper noise on the sample images. It maybe useful for removing salt and pepper noise after histogram equilisation or after binarisation.

5.3.1 Methodology and algorithm

The implementation of Gaussian Filter can be seen in "OCRMain.m". For each filter there are some parameters of the kernels that can be changed to improve the output of the filtering. For example, in gaussian filter and median filter, we can change the σ and size of the kernel, respectively. In our implementation we tweaked various parameters and using the output image we feed it to the Tesseract OCR and store the **Levenshtein Distance** for the filtered image. The kernel that produces the smallest

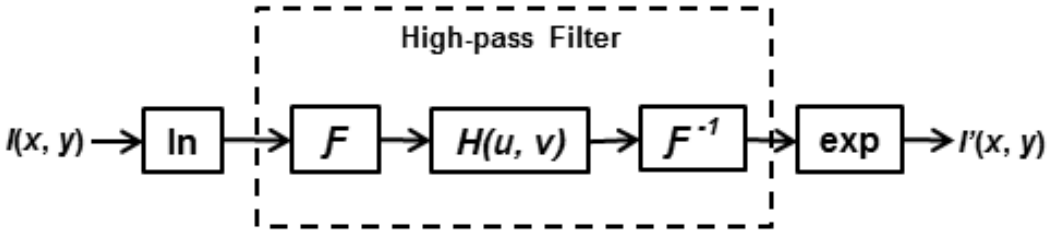
Levenshtein Distance is used. If there is no OCR prediction improvement, not implement the Noise Removal Filters.

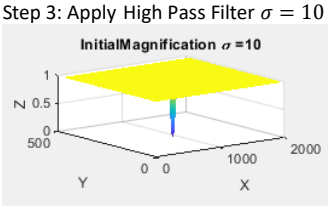
5.4 Step 4: Binariztion

5.4.1 Pre-Binariztion: Homomorphic filtering [4] [10]

5.4.1.1 Methodology and algorithm

Table 3: Steps for Homomorphic filtering



Step 1:Convert image to log domain
$\ln(I(x,y)) = \ln(L(x,y)\,R(x,y))$
$\ln(I(x,y)) = \ln(L(x,y)) + \ln(R(x,y))$
Code
`I = im2double(I);`
`I = log(1 + I);`
	Step 2: Fast Fourier Transfom
	Code
	`M = 2*size(I,1) + 1;`
	`N = 2*size(I,2) + 1;`
	`If = fft2(I, M, N);`
		Step 3: Apply High Pass Filter $\sigma = 10$
		 
		
		Code
		`H = fftshift(H);`
		`Iout = ifft2(H.*If);`
			Step 4: inverse-FFT
			Code
			`Iout = real(Iout);`
				Step 5: invert the log-transform
				Code
				`Ihmf = exp(Iout) - 1;`

"Homomorphic filtering is sometimes used for image enhancement. It simultaneously **normalizes the brightness** across an image and **increases contrast**" [10]

Most image denoising/enhancement technique assumes an additive noise model $I{\sim}(x,y) = I(x,y) + n(x,y)$, where n is the noise signal. But for Homomorphic filtering a multiplicative noise model $I{\sim}(x,y) = I(x,y)\,n(x,y)$ is assumed. "*The illumination-reflectance model of image formation says that the intensity at any pixel, which is the amount of light reflected by a point on the object, is the product of the illumination of the scene and the reflectance of the object(s) in the scene*" (1)

$$I(x,y) = L(x,y)\,R(x,y) \text{ where } I \text{ is the image, } L \text{ is scene illumination, and } R \text{ is the scene reflectance.}$$

To compensate for non-uniform illumination the key is to remove the illumination component, L and keep the reflectance component, R. Because illumination typically varies slowly across a scene, it is typically low frequency.

$$\ln(I(x,y)) = \ln(L(x,y)\,R(x,y))$$

$$\ln\big(I(x,y)\big) = \ln\big(L(x,y)\big) + \ln\big(R(x,y)\big)$$

Step 1 (Table 3) of homomorphic filtering is to transform the multiplicative components to additive components by moving to the log domain.

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi \left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

Equation 1: Forward Fourier Transform (Analysis)

Step 2(Table 3) is to carry out the Fourier transform of the image. In code a fast Fourier transform is used.

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

Equation 2: 2D Gaussian

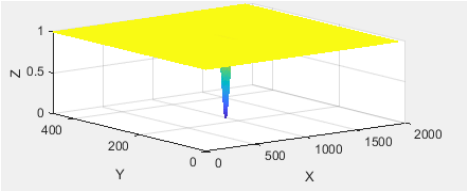


Figure 17:High pass Filter $\sigma = 10$

```
M = 2*size(I,1) + 1;
N = 2*size(I,2) + 1;
%Creating High Pass Filter code
sigma = 10;
[X, Y] = meshgrid(1:N,1:M);
centerX = ceil(N/2);
centerY = ceil(M/2);
gaussianNumerator = (X - centerX).^2 + (Y - centerY).^2;
H = exp(-gaussianNumerator./(2*sigma.^2));
H = 1 - H;
H = fftshift(H);
```

Step 3(Table 3) is to create the high pass filter and apply the high-pass filter. Construct a **simple Gaussian high-pass filter** directly in the frequency domain. Note FFT shift is used to rearrange the filter in an un-centered format because in formal Fourier Transform, low frequencies are at the corner.

$$f(a,b) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l)\, e^{\iota 2\pi \left(\frac{ka}{N} + \frac{lb}{N}\right)}$$

Equation 3: Reverse Fourier Transform (Synthesis)

Step 4(Table 3) compute the inverse-FFT on the filtered image.

Step 5(Table 3) apply the exponential function to invert the log-transform and get the homomorphic filtered image.

5.4.1.2 *Result Homomorphic filtering [4] [10] →Contrast/Hist Eqi →OTSU* (BEST) Result for sample01

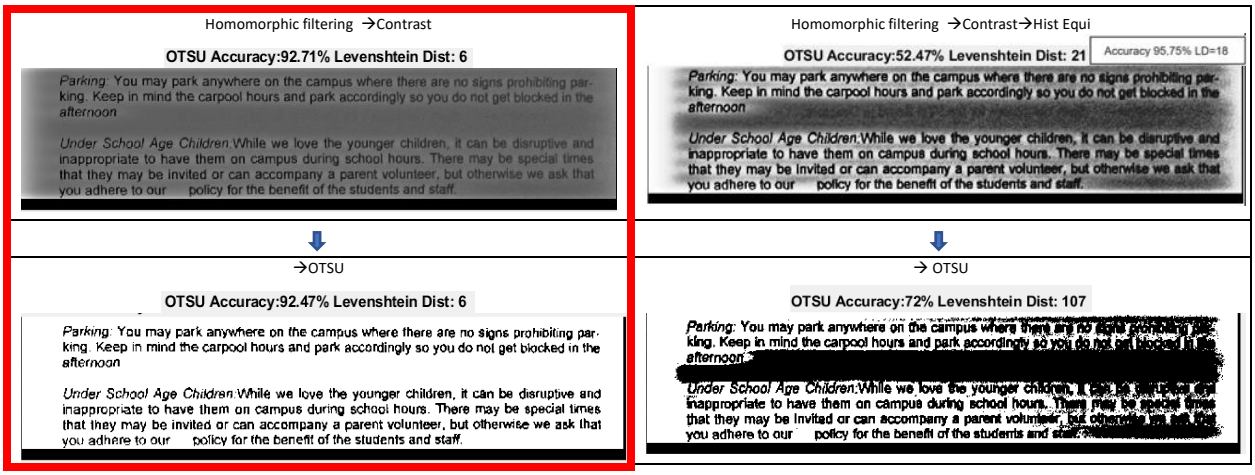


Figure 18: Results Homomorphic filtering+ Contrast + OTSU (sample01.png)

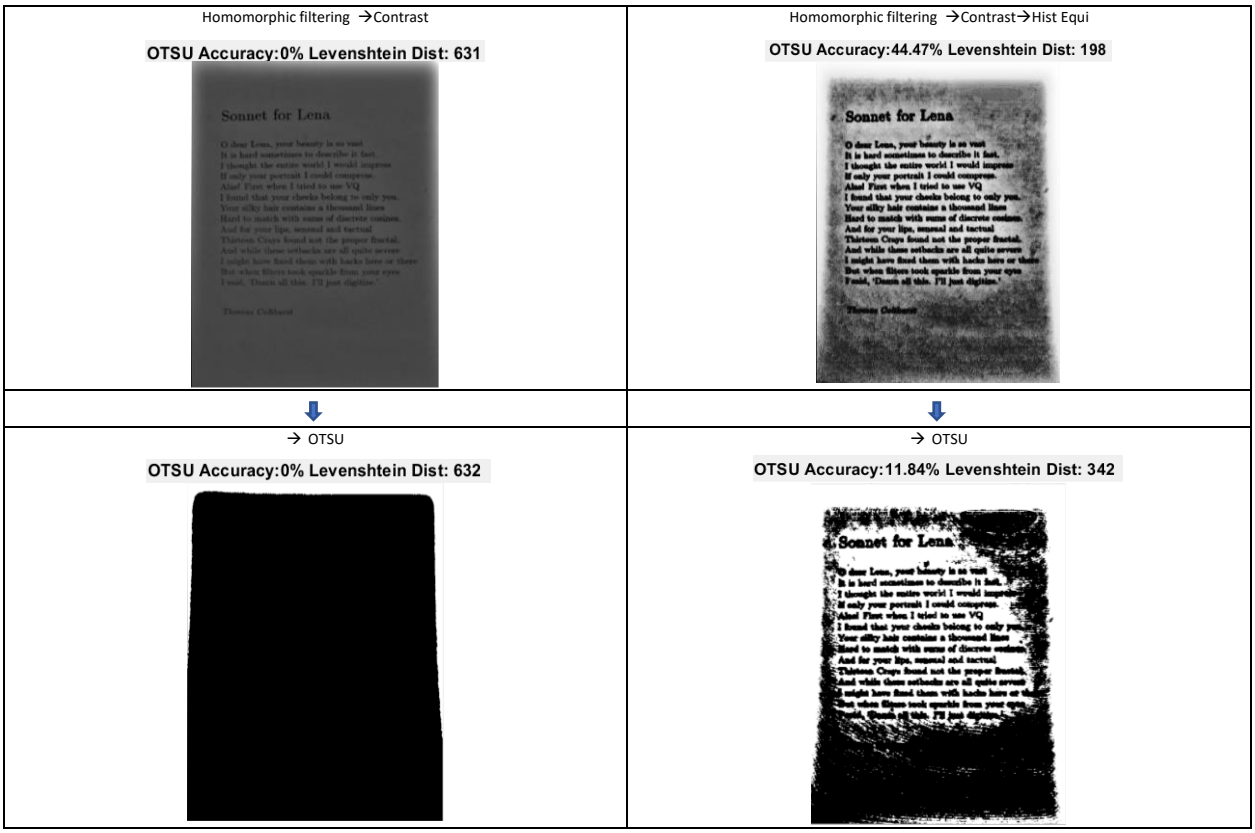


Figure 19: Results Homomorphic filtering+ Hist Equalization+ OTSU (sample02.png)

Table 4: Accuracy Results for Homomorphic Filtering

	Sample01.png		Sample02.png	
	Homo / Homo +OTSU ***(BEST) Result for sample01***	Homo +hist_eq/ Homo +hist_eq+OTSU	Homo / Homo +OTSU	Homo +hist_eq/ Homo +hist_eq+OTSU
Accuracy (%) (Manual Change)	92.71(+40.49)/ 92.47(+40.35)	95.75(+43.63)/ 72.17(+20.05)	0(-7.18)/ 0(-7.18)	44.47(+37.29)/ 11.84(+4.66)
Levensthein Distance [6]	6(-235)/ 6(-235)	18(-223)/ 106(-135)	631(+225)/ 632(+226)	198(-208)/ 342(-64)

Homomorphic filtering produced excellent results on sample01.

While for Sample02 the better results are achieved from Homomorphic filtering+ histogram equalization.

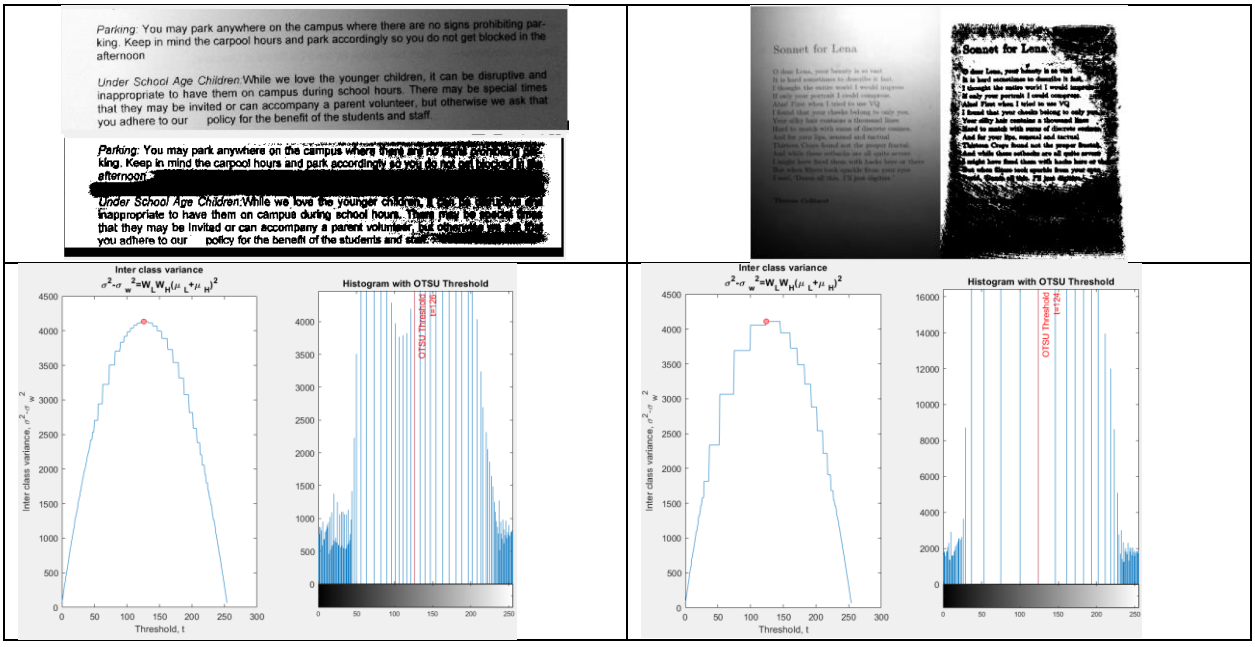


Figure 20: Histogram Equalization Sample01.png vs Sample02.png

There is a huge drop in prediction accuracy with OTSU binarization after Homomorphic filtering. This is very likely due to the bright borders skewing the resulting shape of the histogram this can be seen in Figure 20.

5.4.2 Adaptive Thresholding

5.4.2.1 Methodology and algorithm [11] [12]

Adaptive Threshold, unlike global thresholding, computes threshold for a local region based on its surrounding pixel. The threshold is chosen based on (first-order statistics) mean, median, gaussian of neighboring pixels. By default, MATLAB uses mean for the thresholding.

In the implementation of Adaptive thresholding, the sensitivity of the thresholding is varied to optimize OCR Accuracy (Figure 21). This optimization is not possible in real life without the ground truth but with a large enough labelled data, we can estimate a more general optimal sensitive for Adaptive Thresholding

5.4.2.2 Result Gaussian Blurring →Optimized Adaptive Thresholding →Erosion (General BEST) Result for Both

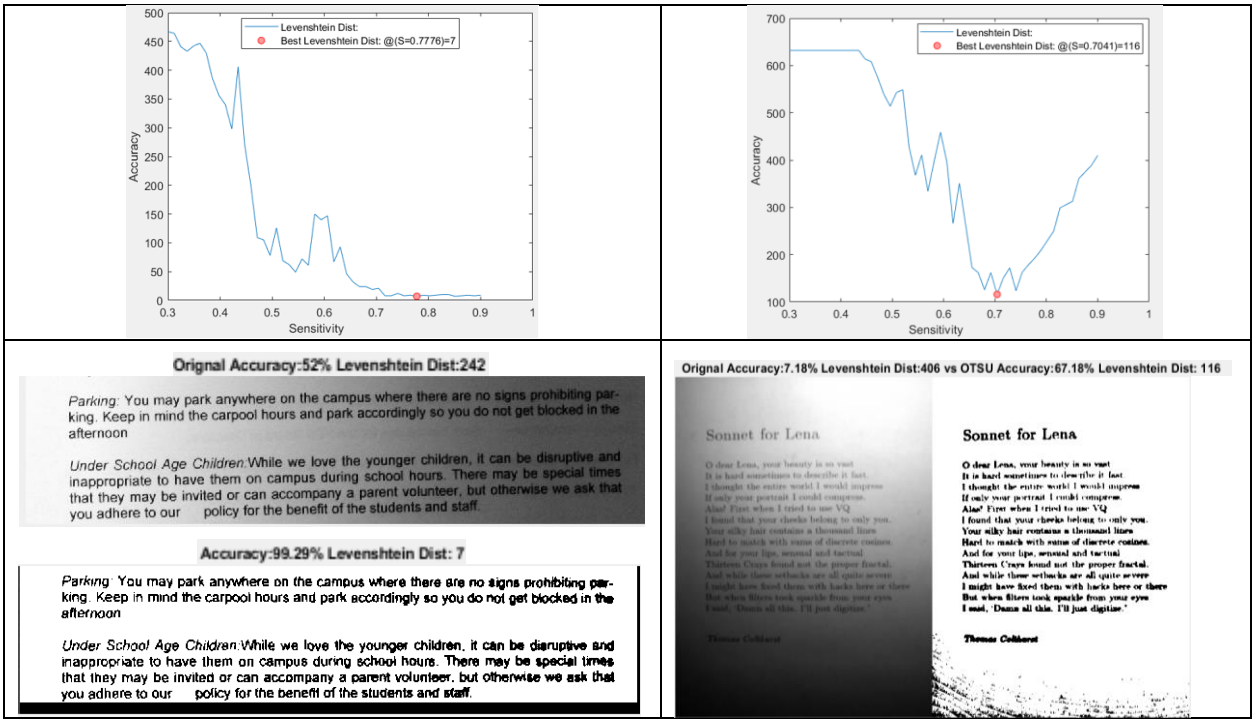


Figure 21: Results of Adaptive Thresholding

Table 5: Accuracy Results for Gaussian Blurring →Adaptive Thresholding →Erosion

	Sample01.png	Sample02.png
	Gaussian Blurring →Adaptive Thresholding →Erosion	Gaussian Blurring →Adaptive Thresholding →Erosion
Accuracy (%) (Manual Change)	99.29(+40.47)	67.18(+60)
Levensthein Distance [6]	7(-234)	116 (-290)

From Figure 21, we can see that the optimal sensitivity for adaptive threshold is 0.78 and 0.70 for sample01.png and sample02.png respectively. This possibly indicates that the optimal sensitivity for a general Adaptive Thresholding is approximately in the range of 0.7 to 0.8.

In terms of OCR prediction accuracy, adaptive **produces the best results for both samples compared to all the different methods that can mitigate the effects of uneven lighting conditions**. This could indicate that Adaptive thresholding is a more general method compared to Homomorphic filtering. As shown in Figure 21, after adaptive filtering Sample 2 does not have the bright edges that sample 2 has after homomorphic filtering.

Another possible reason for the difference in generality is because homomorphic parameters where not optimized. At the current stage, it appears that adaptive filter is a more general solution for an image with uneven lighting.

5.4.3 Imtophat

5.4.3.1 Methodology and algorithm [11] [12]

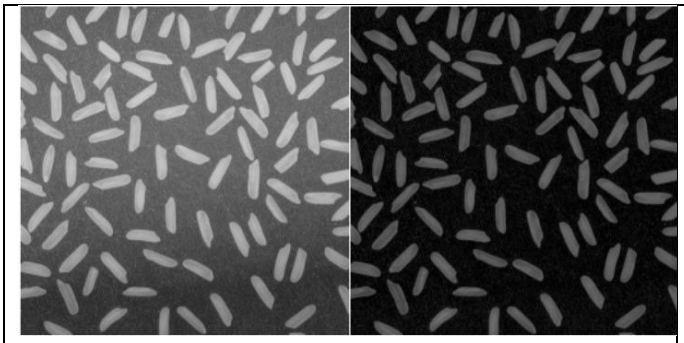


Figure 22: Before and after Imtophat transform,

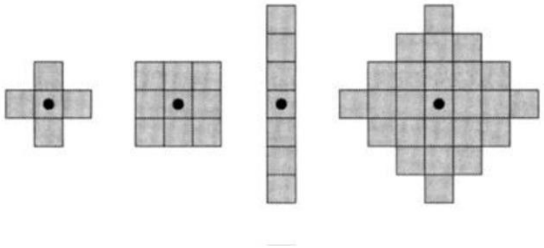


Figure 23: Example of Structuring element

A morphological top-hat transform 1) opens an image, then 2) subtracts the opened image from the original image.

Opening operation first erode and then dilates the gray scale image, using a specified structuring element. Using the same structuring element, the eroded image is dilated. The effect of Morphological opening is that small objects are removed while the overall shape and size of the larger object is preserved. For the top-hat transform, Opening will help obtain the “illumination”.

Next, the opened imaged (“illumination” map) is subtracted from the original image leaving us with an image with better illumination. The top-hat transform can be used to enhance contrast in a grayscale image with nonuniform illumination. The transform can also isolate small bright objects in an image.

Similar to Noise removal section, in our implementation of top hat, we loop through various structuring element type and size and only retain the structuring element type and size if it produces the best accuracy (lowest **Levenshtein Distance**).

5.4.3.2 *Result of Histogram Equilisation →Tophat →Erode →Optimized Adaptive Thresholding*

Table 6: top-hat transform Structuring element Optimization Sample01.png

Shape Size	Structuring element	Diamond (1)	Disk (2)	Square (3)
	9	330	254	379
	10	309	167	376
	11	341	237	329
	12	258	226	355

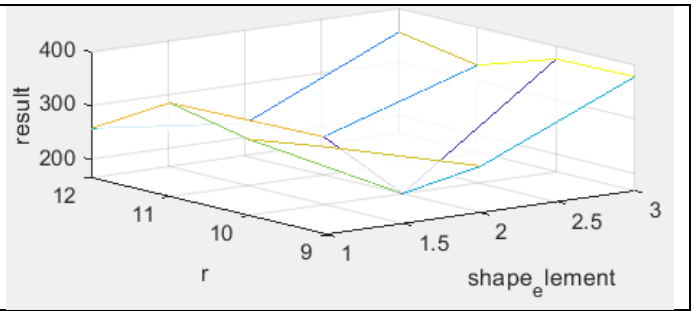
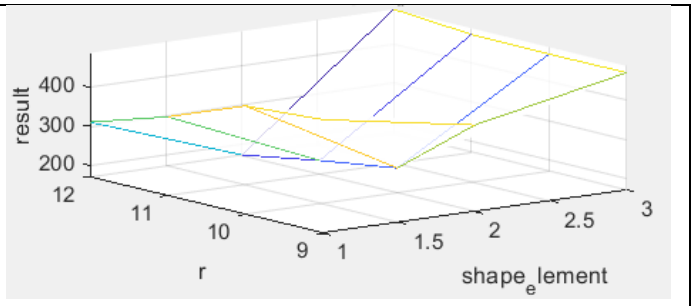


Table 7: top-hat transform Structuring element Optimization Sample01.png

Shape Size	Structuring element	Diamond (1)	Disk (2)	Square (3)
	9	462	393	471
	10	449	235	470
	11	373	206	472
	12	311	172	488



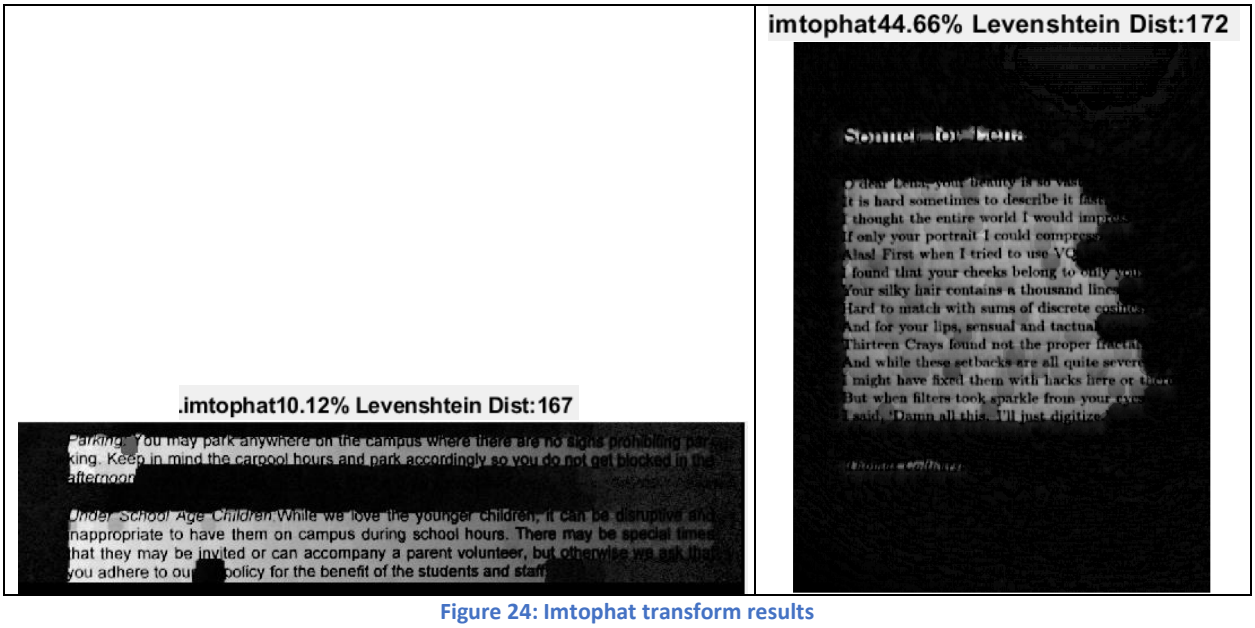


Figure 24: Imtophat transform results

Table 8: Accuracy Results for Histogram Equilisation →Tophat⍰Erode→Optimized Adaptive Thresholding

	Sample01.png	Sample02.png
	Gaussian Blurring →Adaptive Thresholding →Erosion	Gaussian Blurring →Adaptive Thresholding →Erosion
Accuracy (%) (Manual Change)	~~10.12(-42)~~	44.66(+37.48)
Levensthein Distance [6]	167(-74)	172 (-234)

In order to **optimize the shape element type and size**, we looped through the tophat transform with different shape and size as you can see in Table 9, the optimal structuring element for sample01.png is Disk with a size of 10 and from Table 10, the optimal structuring for sample02.png is Disc with a size of 12.

It may be possible that disk shape element can be used for a generic preprocessing workflow. It is also interesting to note that Sample 1 uses a smaller shape element than sample 2.

In terms of OCR accuracy, both Sample01 and Sample02 have good OCR prediction results when looking at the Levensthein Distance. However, the results are **worse than that of Adaptive Filtering.** This may be due to the interference between the words at the background. This cause a lot of words at the corner to be missing.

With this in mind, we can try to segment the background(paper) from the foreground (the words) by creating a mask derived from the tool that produced the best results (Adaptive filtering). By combining this mask with homomorphic filtering or tophat transform, we can get a better pre-segmented image for binarization.

5.5 Step 5: Thinning and Skeletonization [13]

5.5.1.1 Methodology and algorithm [11] [12]

Figure 25:(a) Original Image (b)Converted to Grayscale (c) Binarized image (d)Thinning and Skeletonization (e) Noise Removal

Thinning and Skeletonization are morphological operations, which process images based on shapes. “Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors.” [13]

Thinning and Skeletonization is implemented the same way tophat **5.4.3.1** is implemented. We varied the structuring element shape and size to obtain optimal structing element for erosion and opening.

5.5.1.2 Result of Thinning and Skeletonization

Table 9: Erode Structuring element Optimization sample01.png

Shape Size	Structuring element	Diamond (1)	Disk (2)	Square (3)
	3	509	516	374
	4	509	514	516
	5	511	513	516
	6	508	511	512

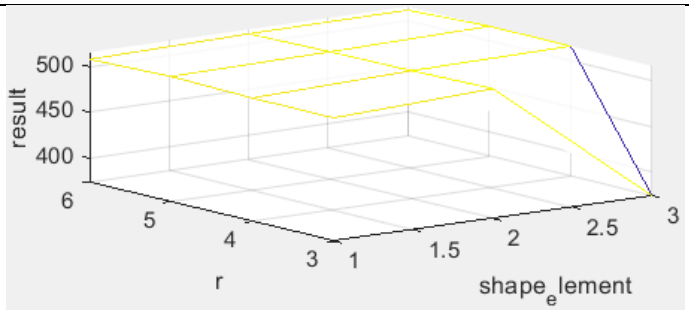


Table 10: Erode Structuring element Optimization sample02.png

Shape Size	Structuring element	Diamond (1)	Disk (2)	Square (3)
	3	627	581	295
	4	620	623	507
	5	622	620	581
	6	624	624	627

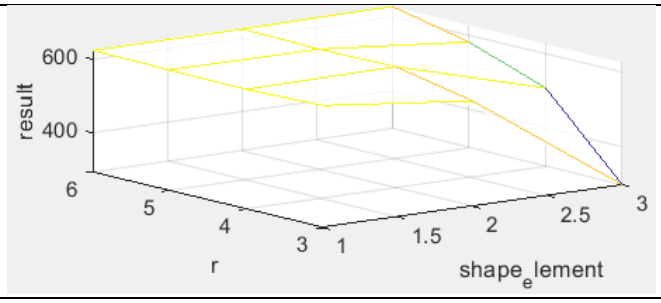
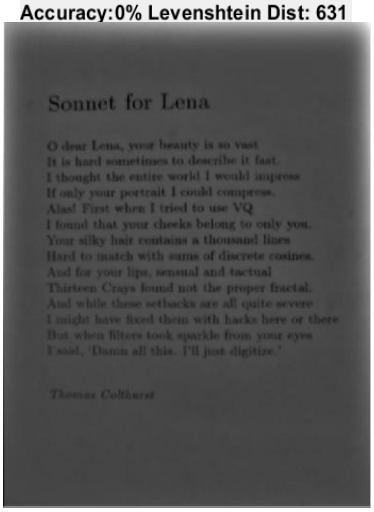
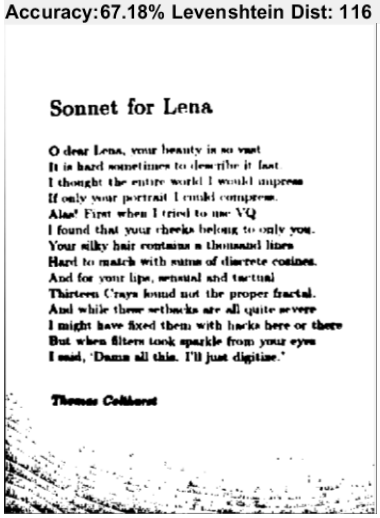
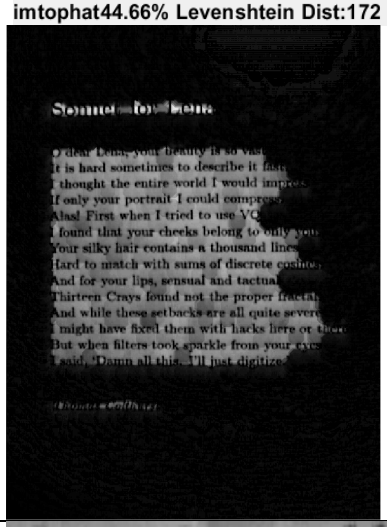
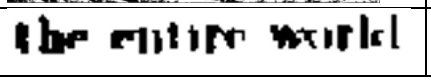


Table 9 and Table 10 shows the structural element optimization process. The structural element that produces the lowest **Levenshtein Distance** (Most accurate).

However, **most of the time skeletonization is not implemented because it did not result in an improvement in OCR prediction**. As you can see in Table 9 and Table 10 the lowest **Levenshtein Distance** is 374 and 295 respectively, which are very much higher than 167 and 172 in Table 6, Table 7.This means that Skeletonization or erosion actually made the prediction worst. This might be because there seems to be very few irregularities with the characters and thus erosion might only make the characters harder to recognize.

6 Improving prediction results of Sample02.png for using Segmentation mask

Homomorphic Filtering	Adaptive Thresholding	Tophat Transform
Accuracy:0% Levenshtein Dist: 631	Accuracy:67.18% Levenshtein Dist: 116	imtophat44.66% Levenshtein Dist:172
		
		

When we compare the results of the different techniques, we can see that Adaptive filtering has the highest OCR prediction accuracy with a **Levenshtein Distance** of 116 while Homomorphic filtering and Tophat has a **Levenshtein Distance** of 631 and 172 respectively.

But when comparing the output images visually we can see that for both Homomorphic filtering and Top hat transform, they both retain more of the original character's edges and features.

Therefore, this section explores the possibility of using output from optimal thresholding to mask away the background of either Homomorphic filtering or Tophat Transform in order to get a better result than the Optimal Adaptive filtering (**Levenshtein Distance=** 116)

6.1 Methodology and algorithm

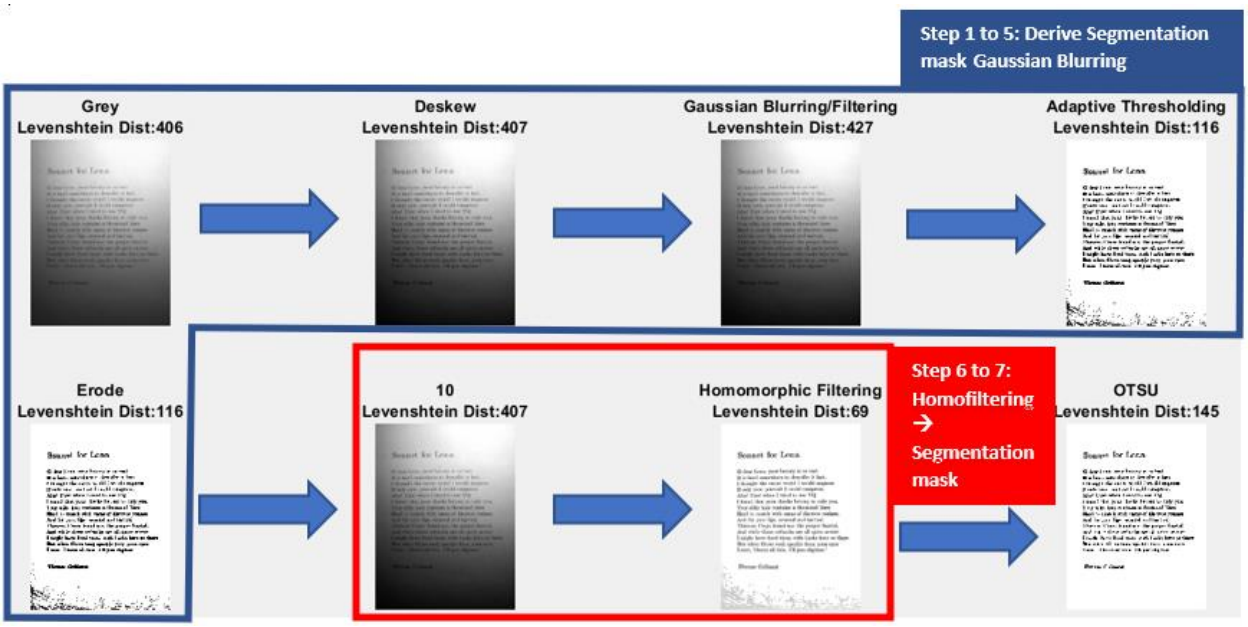


Figure 26: Optimal workflow for Sample 2 preprocessing

Steps 1 to 5 is deriving the segmentation mask. We carry out the same operations in **5.4.2.1**. Then using dilation, we the inverted image. In step 6 we create the segmentation mask.

```
A=find(preprocessing_steps==5);
pic=img_raw_results_cell{A+1};
pic_invert=(pic==0);
se = strel('disk',3);
dilatedI = imdilate(pic_invert,se);
figure;imshowpair(pic_invert,dilatedI,'montage')

prev_img=img_raw_results_cell{i}
mask_array=find(dilatedI==0);

prev_img(find(dilatedI==0)) = 255;
```

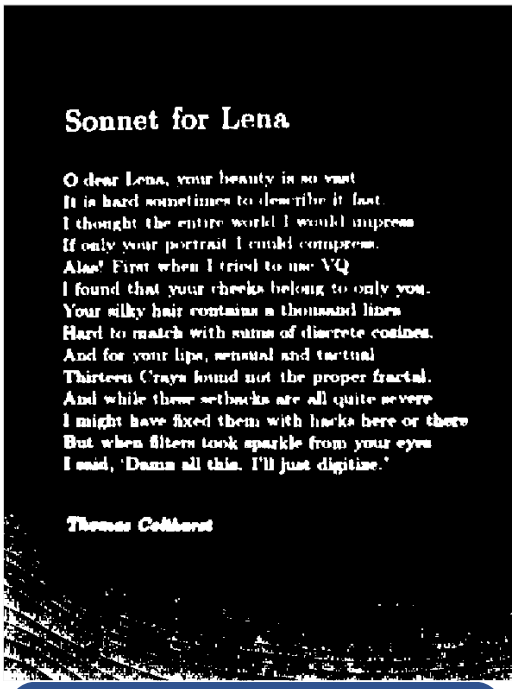
Dilate the inverted **adaptive threshold image**, with a disk structuring element of size 3.

Identify the indexes of pixels that are dark(`mask_array`). This indexes will help us locate the back ground and we can manipulate the background
For instance

```
Image(mask_array) = 255;
```

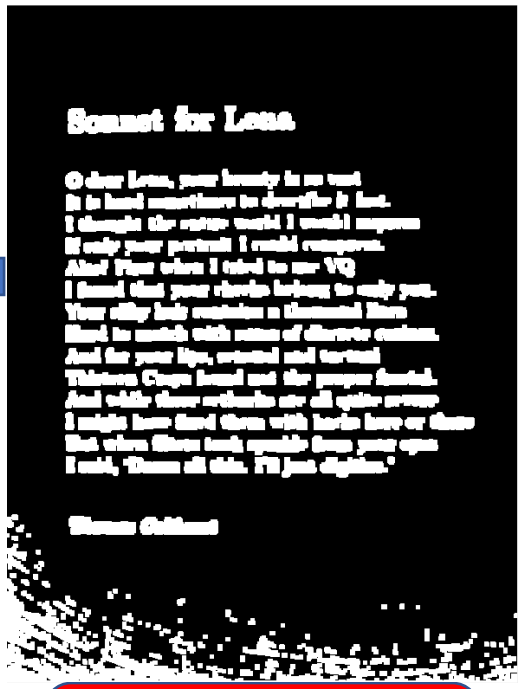
Will make the back ground bright.

In step 7 Homomorphic filtering is carried out on the original image but this time at the final step, we mask the results of homomorphic image with the segmentation mask we have created.



Step1 to 5

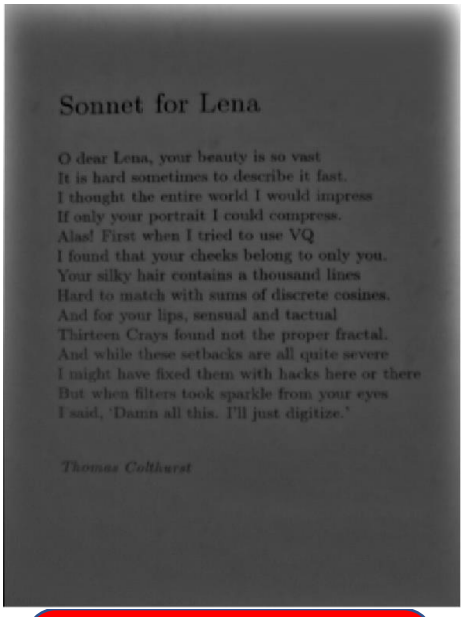
Derive Segmentation mask Gaussian Blurring →Adaptive Thresholding →Erosion



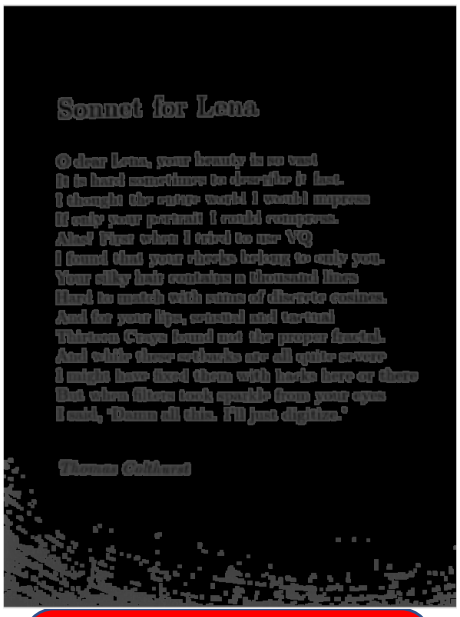
Step 6:

Derive Segmentation mask using diation with disk structural element of size 3.

Mask: White Section Transparent



Step 7: Conduct homomorphic filtering on original image



Step 7: Mask the Results of homorphic filtering with segmentation mask

6.2 Result of Homomorphic filtering plus segmentation mask*(BEST) Result for sample02*

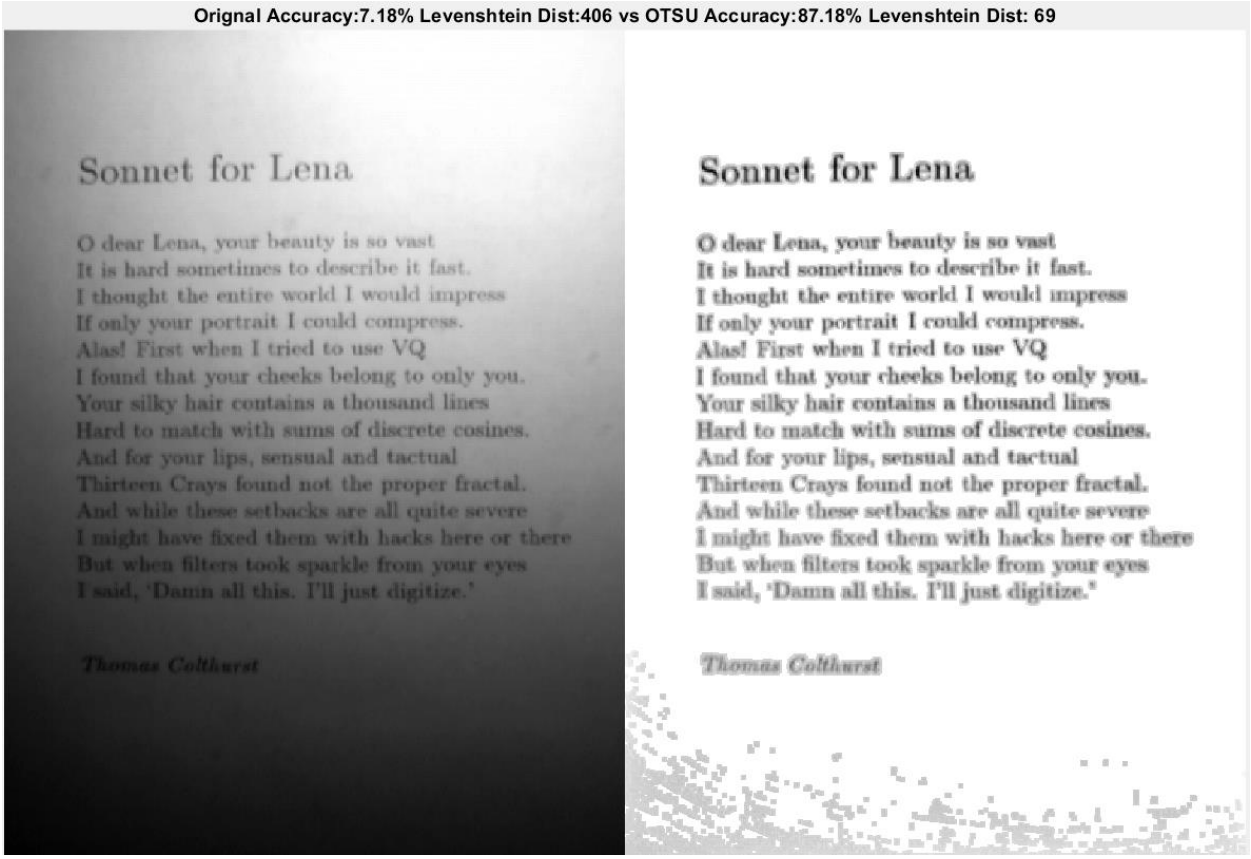


Table 11: Accuracy Results for Homomorphic Filter→Foreground Background Segmentation (Gaussian Blurring → Adaptive Thresholding →Erosion)

	Sample01.png	Sample02.png *(BEST) Result for sample02*
	Gaussian Blurring →Adaptive Thresholding →Erosion	Gaussian Blurring →Adaptive Thresholding →Erosion
Accuracy (%) (Manual Change)	1.65%(-50.47)	87.18(80)
Levensthein Distance [6]	470(+229)	69(-337)

The results of sample 2 is the **best results for sample 2 in the entire paper**. It achieved a 87% accuracy with a very low Levensthein Distance=69. Which is an improvement from the next best of Levensthein Distance=116 derived from Adaptive Thresholding.

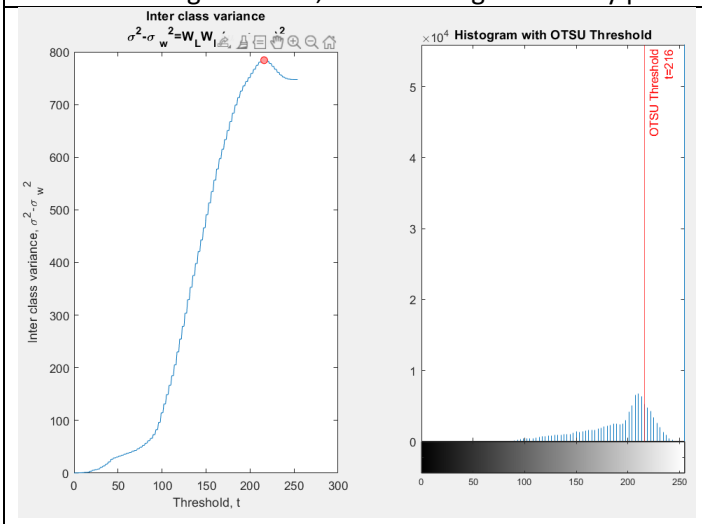
By using this technique, contrast stretching, and OTSU thresholding can be applied more effectively because there is lesser interference from the background. However, there is an issue of the partially segmented background skewing the histogram of the image.

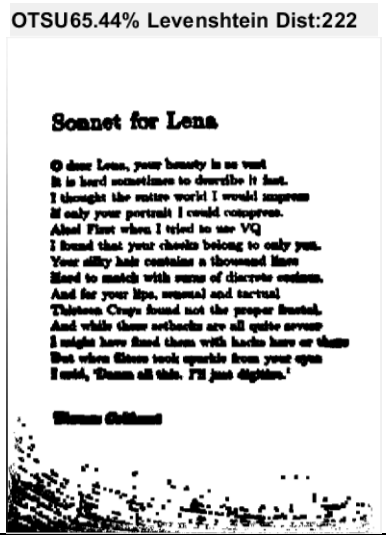
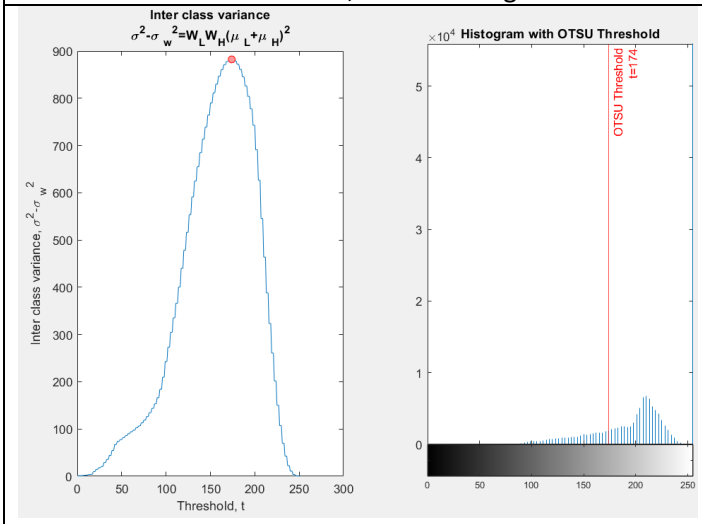
As a result, Novel Methods of Contrast stretching, and OTSU was introduced.

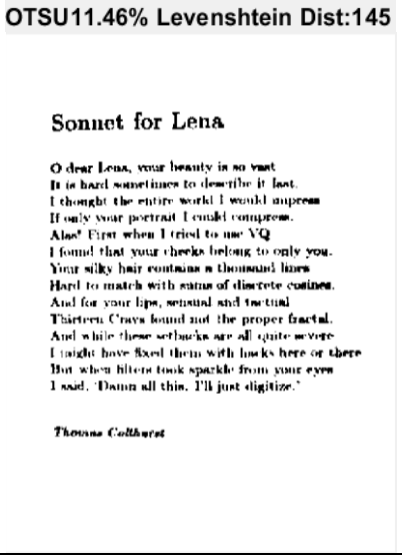
6.3 Modified OTSU

For OTSU thresholding, the new methods involve checking if any of the bins contain more than 70% of the count. If there is, that bin's count will be set to zero so that it does not skew the Optimal OTSU thresholding derived.

```
count(find(count>0.7*sum(count(1:256))))=0;
```

<table>
<tr><td>Original OTSU, Thresholding Skewed by partial segmentation white back ground</td></tr>
<tr><td> 


</td></tr>
<tr><td>Modified OTSU, Thresholding is much better with less skewing from bin=255.</td></tr>
<tr><td> 


</td></tr>
</table>

6.4 Modified Contrast Stretching

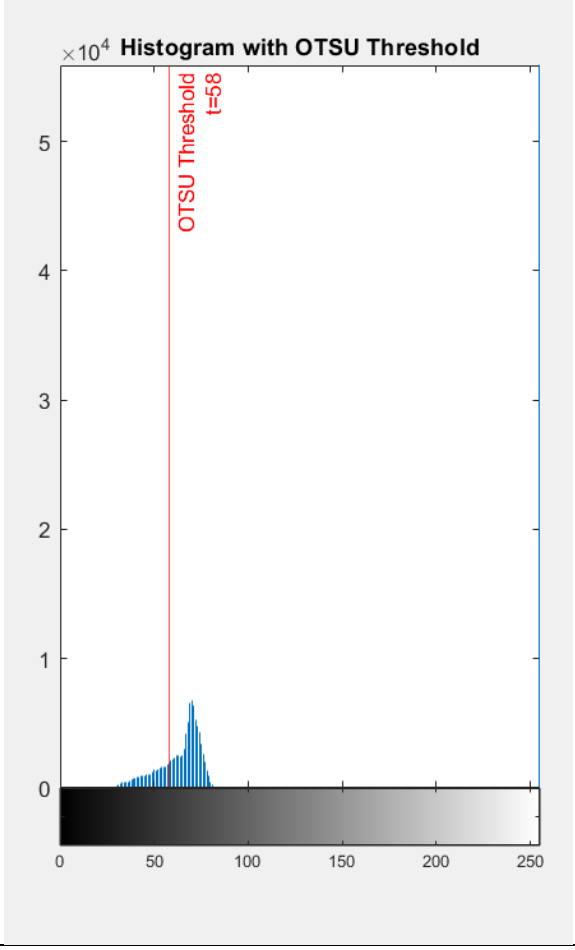
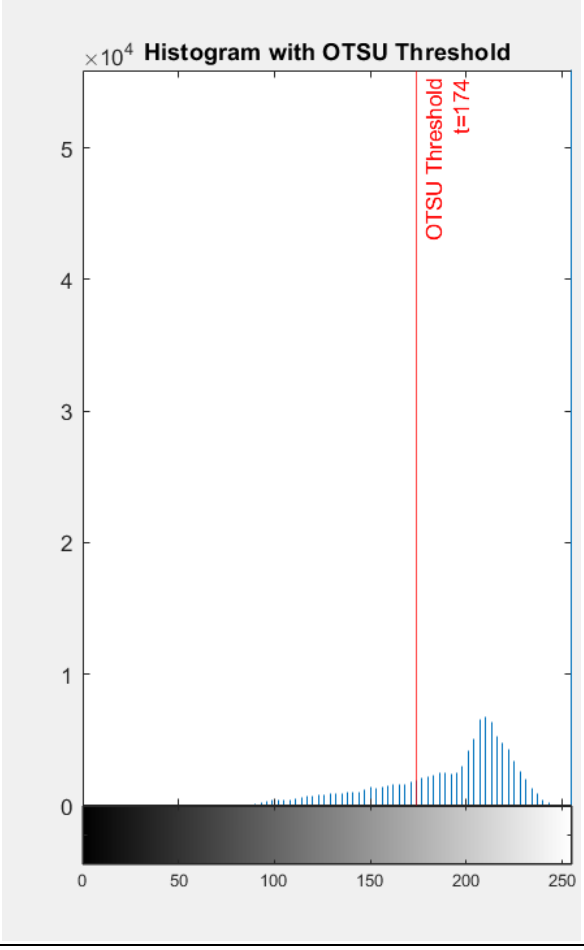
To remove the effects of the white regions due to partial segmentation of the background we first element them. By doing so out p max will not be 255. And this will allow contrast stretching of the other pixels to happen.

```
function stretched =Contrast_stretch_B_special(gray)
    P=gray
    P(find(gray==255))=0;
    figure('Name',"10 Problem 1");imshow(P)
    min_P=double(min(P(:)))%Min intensity=13
    max_P=double(max(P(:)))%Max intensity=204
    P=uint8((double(P(:,:))-min_P).*(255/(max_P-min_P)));
    P(find(gray==255))=255;
    figure('Name',"10 Problem 2");imshow(P)
    stretched =P;
end
```

Modified contrast stretching algorithm first changes all 255 to 0 using **index from the original image**

Then Contrast stretching is conducted

Finally using the same **index from the original image** we convert the pixels which previously has grey levels of 255 back to 255

Original Contrast Stretching (not done properly due to influences from white background)	**Modified Contrast Stretching (done without influence from white back ground)**
	

7 OCR Evaluation Matrix

7.1 Primary: Levensthein Distance

Method

The primary method of measuring accuracy or similarity of OCR prediction is done using Levenshtein Distance. **When reading the report please depend on Levenshtein Distance. The smaller the Levenshtein Distance the better the accuracy of the OCR results.**

"The Levenshtein Distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other." [6]

Shortcomings [14]

Levensthein distiance is not robust in handling punctuations, accent marks and capital letters.

Another shortcoming is that it does not account for semantics of a word, For example,

- 'receive' misspelled as 'recieve' distance of 2 (for 2 substitutions)
- 'receipt' is also both 2 substitutional edits

7.2 Secondary: self-written evaluation method

Methods

The Secondary evaluation method is a code written by me. The code first breaks down both the ground truth and OCR prediction into a list of sentences. The sentences are then further broken down into a list of words.

For each sentence, each word is compared. If the character matches, 1 is added to the results.

Assumptions and Short comings

An assumption made in this code is that the OCR algorithm will at least detect the correct number of words per sentence, which is likely not true. Due to this assumption, the self-written evaluation method is not robust to situations like missing sentence or missing words.

Mitigation of Short comings

This miss alignment issues cause a miss calculation of 50% as seen in some situation throughout the papers. Too mitigate this we will manually check and adjust miss aligned output to produce a more accurate OCR accuracy measurement. If manual changes is not possible, we will reject the measurement and depend on the primary evaluation matrix.

7.3 Other: Evaluation to consider Jiwer.

Note since this matrix is derived from Levensthein Distance, we did not employ this for Evaluation but instead directly used Levensthein Distance as the primary evaluation method.

This repository was used to approximate the Word Error Rate (WER) of a the OCR output. Using the Wagner-Fisher [15] algorithm, the minimum distance edit distance between the ground-truth sentence and the hypothesis sentence of a Tesseract is computed.

7.3.1 Word Error Rate(WER) [16]

$$WER = (S + D + I)/N_1 = (S + D + I)/(H + S + D)$$

where I = the total number of entries, D = total number of deletions, S = total number of replacements, H = total number of successes, and N1 = total number of reference words

substitution, S- If a word in the reference sequence is transcribed as a different word
deletion, D- When a word is completely missing in the automatic transcription
insertion, I- The appearance of a word in the transcription that has no correspondent in the reference word sequence

"The performance accuracy of a system is usually rated by the word error rate (WER) (1), a popular ASR(Automated speech Recognition) comparison index. **It expresses the distance between the word sequence that produces an ASR and the reference series**.

Despite being the most used, WER has some cons. It is not an actual percentage because it has no upper bound. When S = D = 0 and we have two insertions for each input word, then I = N1 (namely when the length of the results is higher than the number of words in the prompt), which means WER = 200%. Therefore it does not tell how good a system is, but only that one is better than another. Moreover, in noisy conditions, WER could exceed 100%, as it gives far more weight to insertions than to deletions." [16] Smaller number is better. It measures average error (or the word error rate). Larger Values indicate that a larger probability of the word predicted (excluding insertion) being wrong.

7.3.2 Match Error Rate (MER) [16]

Match Error Rate (MER) is the proportion of I/O word matches, which are errors, which means **that is the probability of a given match being incorrect.** The ranking behavior of MER (2) is between that of WER and WIL

$$MER = (S + D + I)/(N = H + S + D + I) = 1 - H/N$$

Smaller number is better. It measures average error (or the word error rate). Larger Values indicate that a larger probability of the word predicted (including insertion) being wrong.

7.3.3 Word information lost (WIL) [16]

WIL is a simple approximation to the **proportion of word information lost**

8 Conclusions

In conclusion, in this paper we explore the various preprocessing techniques. In particular we have explored **1) Homomorphic Filtering 2) Adaptive Filtering 3) TopHat** Transform to tackle the problem of uneven lighting. We have optimized the preprocessing tools for each individual sample. Given a larger sample, a more general preprocessing flow could be derived. As it stands adaptive filtering seems to be the most general and generally produces the best average OCR accuracy.

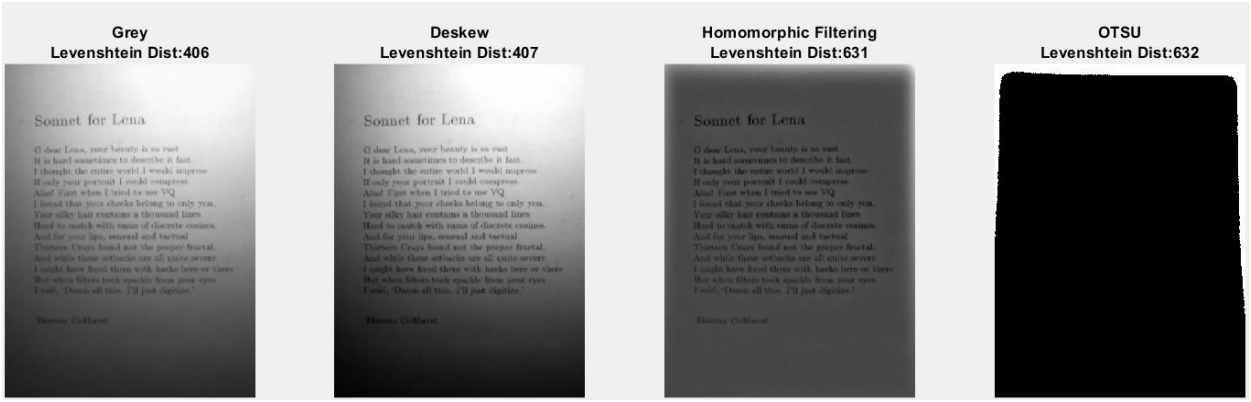
In this paper we also introduced unique workflow Homomorphic Filtering with **Segmentation Mask derived from Adaptive Filtering** and introduced **modifications to make Contrast Stretching and OTSU thresholding** more robust to data biases.

9 Appendix

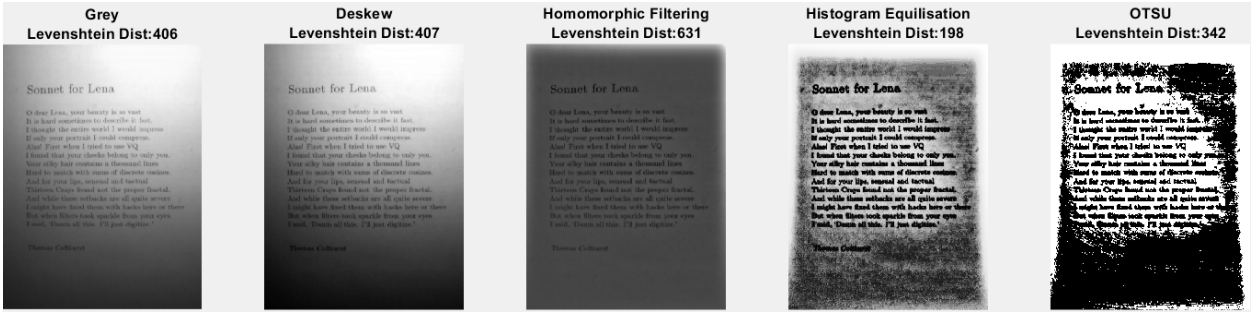
9.1 OCR Results for each step

4.2 Original Sample01.png LD=241	**4.2** Original Sample02.png LD=406
Parking: You may park anywhere on the ce i king. Keep in mind the carpool hours and park at afternoon a as Under School Age Children.While we love the } inappropriate to have them on campus d io that they may be invited or can accompany a you adhere to our _policy for the benefit of t he	Sonnet for Lena 1 dear Lena, your !« is hard sometimes t Rhonght the entire world | iy your portrait [could compr Firat when I tried to use VQ your cheeks belong to only you @ thousand lines vith sums of discrete cosines. and tactual i
4.2 OTSU Thresholding Sample01.png LD=249(+8)	**4.2** OTSU Thresholding Sample02.png LD=487(+81)
Parking: You may park anywhere on the cans king. Keep in mind the carpool hours and part afternoon thd Under Schoo! Age Children:While we love #iifl inappropriate to have them on campus d mee that they may be invited or can accompany 4g you adhere to our —_policy for the benefit of	Sonnet for ler Odear Fotucvees | fw bared stein bt Che ct ei your portian Pecula sans whem Dirial tus Veg your cheeks belong becins sen tbutisaud lines uma of discrete cosines. and toctual ;

Deskew	
5.2 Deskew -1 Sample01.png LD=236(-5)	**5.2** Deskew 0 Sample02.png LD=407(0)
Parking: You may park anywhere on the campus king. Keep in mind the carpool hours and park at afternoon oe Under School Age Children:While we love the y inappropriate to have them on campus dt that they may be invited or can accompany a pi you adhere to our _ policy for the benefit of the	Sonnet for Lena 10 dear Lena, your !« is hard sometimes t Bhonght the entire world | iy your portrait [could compr Firat when I tried to use VQ your cheeks belong to only you @ thousand lines vith sums of discrete cosines. and tactual i

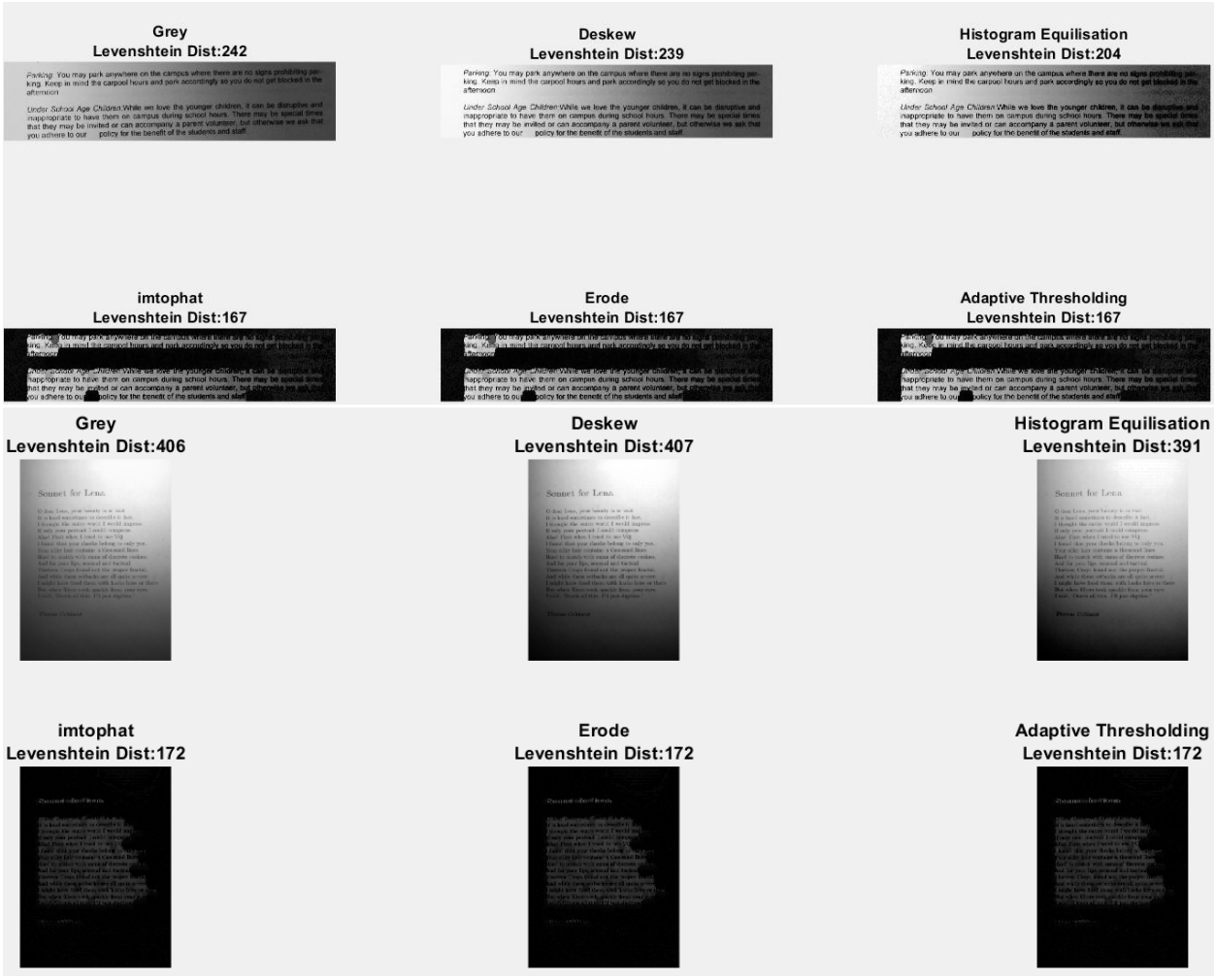


Homomorphic filtering [4] [6] →Contrast →OTSU	
5.4.1.2Homomorphic filtering→Contrast Sample01.png LD=6(-235) ***(BEST) Result for sample01***	5.4.1.2Homomorphic filtering→Contrast Sample02.png LD=631(+225)
Parking: You may park anywhere on the campus where there are no signs prohibiting par- king. Keep 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.	
5.4.1.2Homomorphic filtering→Contrast→OTSU Sample01.png LD=6(-235)	5.4.1.2Homomorphic filtering→Contrast→OTSU Sample02.png LD=632(+226)
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 Schoo! 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.	

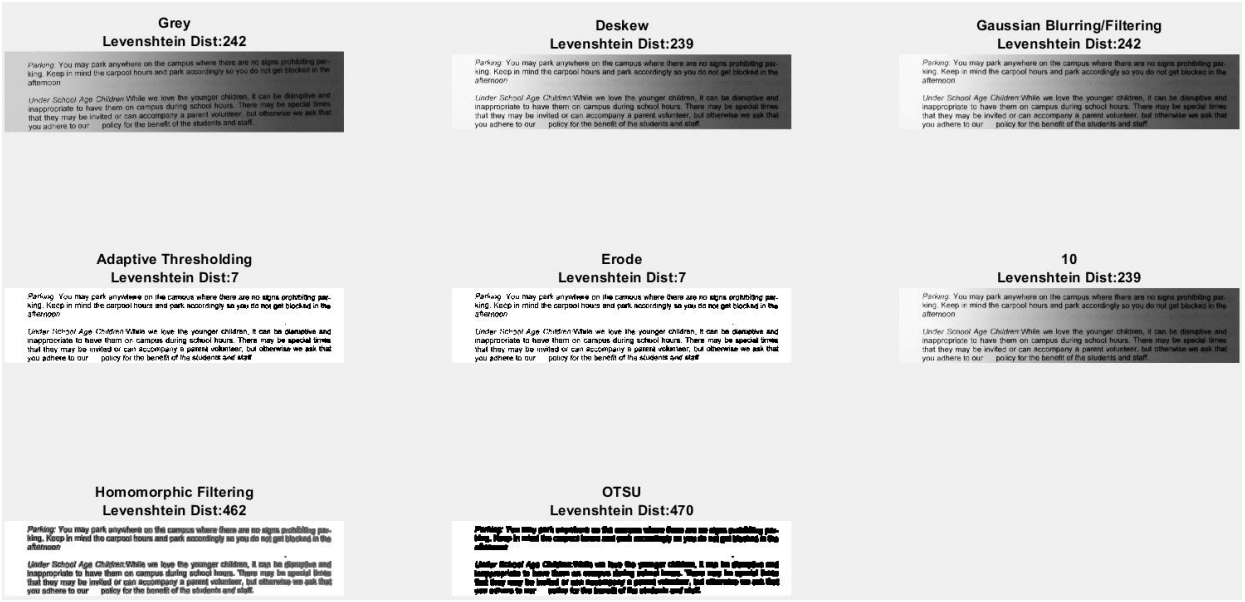


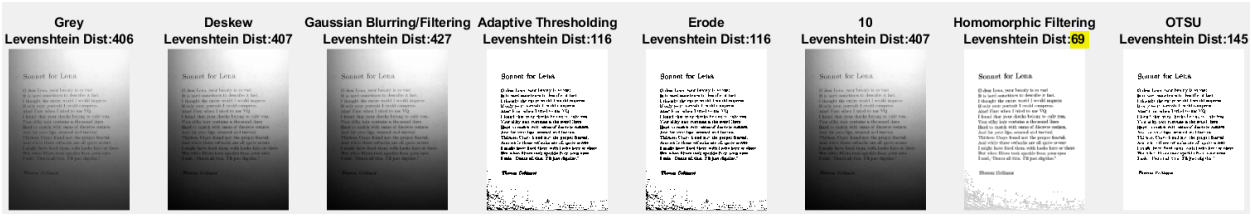
Homomorphic filtering [4] [6] → Hist Eqi→OTSU	
5.4.1.2Homomorphic filtering→Contrast→Hist Equi Sample01.png LD=22(-219)	5.4.1.2Homomorphic filtering→Contrast→Hist Equi Sample02.png LD=198(-208)
Parking: You may park anywhere on the campus Where there are no signs prohibiting par- king, Keep in mind the carpool hours and park accordingly so you do not get blocked in th 3 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. Th mn 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. ~ e	ie s og oe al Ag dear Lena, yout beauty la ea vest "oo 'gy "iu in hard sometimes to describe It feat. # [thought the entire world 1 would impress 72> © Ml enly your portrait I could compress. §< == 57 ' Figet when I tried to use VQ a, ©, 1 found that your cheeks belong to only your" > +. Your ailky hait contains » thousand lines > hy Hard to match with sums of discrete cosinen: < {And for your lips, sensual and tactual ae A ae Crays found not the proper fractal. yo) And while these setbacks are all quite severe G ll might have fixed them with hacks here or there '}:: But when filters took sparkle from your eyee 350) ig | Weald, 'Deasm olf thin. TH fost digitize." a oo ina Secale . f \ Y r e Y f foe q errata eae re ee ee Fy
5.4.1.2Homomorphic filtering→Contrast→Hist Equi →OTSU Sample01.png LD=106(-135)	5.4.1.2Homomorphic filtering→Contrast→Hist Equi →OTSU Sample02.png LD=342(-64)
Parking: You may park anywhere on the campus where tere &1g DO Sais RrOnDenn Ble king. Keep in mind the carpoo! hours and park accordingly vol do ol a oad Be - afternoon. 4 "Under joo! Age Children:While we love tie younger chidneh, & ean. a. Garup an inappropriate to have them on campus during 'school hours. T hen b pec i ton that they may be invited or can accompany a parent volunteér, bit otharwite vib bak Hat you adhere to our —_ policy for the benefit of the students and stella	Pee kee 3 eS oe gg i Sonnet for Lena <a a ae ae oe: yg ocasch pee deer Loon, yout tunuty le os vat SRE AN be in herd socmetionas to describe it fail J 23 oa AR it caly your portrait | ceeld compress. aa as First whea Itrled boue VQ 7-H, Sry 3 found thet ywor chinks belong to only puting Si Yous aiky bade contains » thowssed Maes aii Pee tlard to match with vorns of diacrets sosinen 2 %. And fer your lps, semeua) apd tactaal = = Cenps Sound not the proper Iressa. SE | SRE naight have fined thems with lacks hare at thie" Etbet when Glue took sparkle kom your yan. 28 Oa a sae ae We a — . ..

Gaussian Blurring → Adaptive Thresholding →Erosion	
5.4.2.2 Gaussian Blurring → Adaptive Thresholding →Erosion Sample01.png LD=7(-234) ***(BEST average) Result for Both***	**5.4.2.2 Gaussian Blurring →Adaptive Thresholding →Erosion** Sample02.png LD=116(-290) ***(BEST average) Result for Both***
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 afterncon 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 _—poficy for the benefit of the students and staff.	Sonnet for Lena O dear Lona, cour fenuty in po wast It ia bard soinetimes ta dlesctihe it fast Lthongbt the entire work! P would anipreae {fonty sour portrait Lcnokd compress. Alas' Firat when] trim) te nee VQ I found that yuor cheeks belong to only you. Your silky hair contains a thonsand lines Hard to match with mums of disrrete cosines. And for your lips, nenaval and tertual Thirtern Crays found wut ibe proper fractal. And while three setbacks are all quite severe 1 might bave fixed them with hacka bere or there Bot when filters took sparkle from your eyrw Tesid, 'Damm all thie. I'll just digitize." ee naa he ae a Pye, eae ee A



Tophat Transform→Erode → Adaptive Thresholding	
5.4.3.2 **Histogram Equilisation Tophat Erode Optimized Adaptive Thresholding** Sample01.png LD=167(-74)	5.4.3.2 **Histogram Equilisation Tophat Erode Optimized Adaptive Thresholding** Sample02.png LD=172(-234)
Y park anywhere on the campus where tel No sigt fs nd the carpool ho and park accordingly so you do not | Unters Age Children. While we love ine younger children, 1 cal inappropriate to have them on campus during school hours. There” nay t hat they may be jpyited or can accompany a parent volunteer, buto v fou adhere to ou olicy for the benefit of the students and staff	Sopa etd ee Ronee ents 7 ent bs v tt A tis hard sometinies to describe it i thonght the entire world [would impgg fonly your portrait [could comprgga Jas First when T tried to use VQ found tint your cheeks belong to OOTY og ‘our silky liait contains © thousand lines fard to miatch with sums of discrete oosiits nd for your lips, sensual and tactui Thirteen Crays fotind not the proper al nd while these. eetbacks-aire all quite severe might have fixtd them with lacks lrere or ZR jut when filters took «parkle frow your tym CLE het tl Tas





Homomorphic Filter→Foreground Background Segmentation (Gaussian Blurring → Adaptive Thresholding →Erosion)	
6.2 Homomorphic filtering→ Foreground Background Segmentation Sample01.png LD=470(+229)	**6.2**Homomorphic filtering→ Foreground Background Segmentation Sample02.png LD=69(-337) ***(BEST) Result for sample02***
Gis apn as spor oon pat yor soso Sentesener ae eas	Sonnet for Lena © dear Lena, your beauty is so vast It is hard sometimes to deseribe it fast. T thought the entire world | would impress Tf only your portrait [could compress. Alas! First when I tried to use VQ I found that your cheeks 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 Issid, Dann all this, I'll just digitize." Thomas Cotthurst SP e weg see a a oy A oe - So Ia P aes

10 Code

10.1 OTSU_B.m

```
function threshold =OTSU_B(gray, Analysis)
%OTSU Thresholding
[count,bins]=imhist(uint8(gray),256);
count(find(count>0.7*sum(count(1:256))))=0;

%Maximise intraclass variance
size(count)
inter_class_var=zeros(size(bins,1),1);
for n=1:size(bins,1)
    %Mean of L(backgnd)and H(foregnd)
    mean_L=dot(count(1:n),bins(1:n))/sum(count(1:n));
    mean_H=dot(count(n+1:256),bins(n+1:256))/sum(count(n+1:256));
    weight_L=sum(count(1:n))/sum(count);
    weight_H=sum(count(n+1:256))/sum(count);
    inter_class_var(n)=weight_L*weight_H*(mean_L-mean_H)^2;
end
[M,I] = max(inter_class_var)
%Analysis Report
    if Analysis ==true
        %Plot Interclass variance
        figure( 'Position', [10 10 900 600]);

        subplot(1,2,1);plot(bins,inter_class_var);
        hold on;
        plot(bins(I),inter_class_var(I),'o',...
            'MarkerEdgeColor','red',...
            'MarkerFaceColor',[1 .6 .6])
        title({'Inter class variance','{\sigma}^2-{\sigma _w}^2={W}_L{W}_H({\mu _L}+{\mu
_H})^2'})
        xlabel('Threshold, t');
        ylabel('Inter class variance, {\sigma}^2-{\sigma _w}^2');

        %Plot histogram with OTSU threshold

        subplot(1,2,2);imhist(uint8(gray),256);title("Histogram with OTSU Threshold");
        hold on;
        xline(bins(I),'-r',{'OTSU Threshold',strcat('t= ',num2str(bins(I)))})
    end

threshold=bins(I)
end
```

10.2 OCRMain.m (Run this code)

```
clear all;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Readme%
%Please Select:
    %1.Sample_no-> indicate which sample to run the preproceesing on
    %2.preprocessing_steps-> Uncomment the preprocessing steps to be execused
    %their corresponding report section is indicated.
%%%%%%%%% 1.Select Sample%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sample_no=2
%%%%%%%%% 2.Uncomment Processing Steps to be executed%%%%%%%%%%%%%%%
%"0 Grey", "1 OTSU", "2 Deskew" "3 Homomorphic Filtering", "4 Histogram Equilisation",...
%        "5 Adaptive Thresholding", "6 Opening","7 erode",...
%        "8 Gaussian Blurring/Filtering","9 imtophat"

%4
%OTSU
%preprocessing_steps=[1]

%5.4.1
%Deskew-> Homo-> OTSU
%%%%%%%%%%%%%%%%%%%%%%%%%%BEST for sample 1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%preprocessing_steps=[2 3 1]
```

```
%Deskew-> Homo->Hist Equi ->OTSU
%preprocessing_steps=[2 3 4 1]

%5.4.2
%Deskew-> Gaussian ->Adapt_T-> erode
%%%%%%%%%%%%%%%%%%BEST for in General Sample 1 and Sample 2%%%%%%%%%%%%%%%
%preprocessing_steps=[2 8 5 7]

%5.4.3
%Deskew->Hist Equi-> Tophat-> erode-> Adapt_T
%preprocessing_steps=[2 4 9 7 5]

%6
%Deskew-> Gaussian ->Adapt_T-> erode->10->Homo+ Segment-> OTSU
%%%%%%%%%%%%%%%%%%%%BEST for sample 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
preprocessing_steps=[2 8 5 7 10 3 1]

%Others
%Deskew-> Gaussian ->Adapt_T-> erode->10->Tophat+ Segment-> erode-> Adapt_T
%preprocessing_steps=[2 8 5 7 10 9 7 5]
%Deskew->Hist Equi-> Tophat->Gaussian ->Adapt_T-> erode
%preprocessing_steps=[2 4 9 8 5 7]
%Deskew-> Homo->Hist Equi-> Gaussian ->Adapt_T-> erode
%preprocessing_steps=[2 3 8 5 7]

%add the current folder to the Python search path.
%Run Matlab 2020
if count(py.sys.path,'') == 0
    insert(py.sys.path,int32(0),'');
end

tool_Name={"Grey", "OTSU", "Deskew" "Homomorphic Filtering", "Histogram Equilisation",...
        "Adaptive Thresholding", "Opening","Erode",...
        "Gaussian Blurring/Filtering","imtophat","10"}

img_file=["resource/sample01.png " "resource/sample02.png "]

N=size(preprocessing_steps,2)
img_raw_results_cell=cell(N+1,1);
Results_overall=zeros(2,N+1)
Pc = imread(img_file(sample_no));

whos Pc
if(size(Pc,3)==3)
    P_gray = rgb2gray(Pc);
else
    P_gray = Pc;
end
img_raw_results_cell{1}=P_gray
mask_array=-1
for i=1:N
    %1 OTSU
    if preprocessing_steps(i)==1
        %2 OTSUP=double(py.Overall_OCR.deskew(P));
        P=Contrast_stretch_B(img_raw_results_cell{i});      %2.1 Contrast stretching
        t=OTSU_B(P,true);                                   %2.2 OTSU Global Thesholding
        P=P>t;
        img_raw_results_cell{i+1}=P;

    %Step 2: Deskew
    %2 Deskew
    elseif preprocessing_steps(i)==2

img_raw_results_cell{i+1}=Contrast_stretch_B(double(py.numpy.array(py.Overall_OCR.deskew(img_raw_
results_cell{i}))));


    %Step 4: Binarisation
    %3 Homomorphic Filtering
    elseif preprocessing_steps(i)==3
        %3 Homomorphic Filtering + Contrast Stretching +OTSU
```

```
        P=HOMO_Filtering_B(img_raw_results_cell{i});       %3.1 Homomorphic Filtering
        img_raw_results_cell{i+1} = Contrast_stretch_B(P); %3.2 Contrast stretching
        if mask_array~=-1
            img_raw_results_cell{i+1}(mask_array) = 255;
            img_raw_results_cell{i+1} = Contrast_stretch_B(img_raw_results_cell{i+1});
        end

    %4 Histogram Equilisation
    elseif preprocessing_steps(i)==4
        P=Contrast_stretch_B(img_raw_results_cell{i});
        img_raw_results_cell{i+1} = histeq(P ,255); %histogram equilisation

    %5 Adaptive Thresholding
    elseif preprocessing_steps(i)==5
        %S_varry=linspace(0.659,0.67,50)
        S_varry=linspace(0.3,0.9,50);
        %S_varry=linspace(0,1,100)
        result_to=zeros(1,size(S_varry,2));
        for k=1:size(S_varry,2)
            BW=BW_adaptT(img_raw_results_cell{i}, S_varry(k));
            result_pre=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(BW,sample_no)));
            result_to(k)=result_pre(2);
        end
        [M,I] = min(result_to);
        M
        S_varry(I)%0.661
        figure;plot(S_varry, result_to);
        hold on;
        plot(S_varry(I),M,'o',...
            'MarkerEdgeColor','red',...
            'MarkerFaceColor',[1 .6 .6])
        ;xlabel('Sensitivity [0.659,0.67]');ylabel('Accuracy');
        legend(['Levenshtein Dist: ']...
        ,[strcat('Best Levenshtein Dist:
@(S=',num2str(round(S_varry(I),4)),')=',num2str(round(M,2)))] ...
        ,'Location','best');

result_O=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(img_raw_results_cell{i},sample_no)));
        if result_O(2)<M
            img_raw_results_cell{i+1}=img_raw_results_cell{i}
        else
            img_raw_results_cell{i+1}=BW_adaptT(img_raw_results_cell{i}, S_varry(I));
        end

    %Step 5: Skeletionisation
    %6 Opening, erode,7,erode,9,imtophat
    elseif ismember(preprocessing_steps(i),[6,7,9])
        shape_element={'diamond','disk','square';}
        shape_element_no=linspace(1,size(shape_element,2),size(shape_element,2));
        if preprocessing_steps(i)==6
            r=[3,4,5,6]
        elseif preprocessing_steps(i)==7
            r=[3,4,5,6]
        elseif preprocessing_steps(i)==9
            r=[9,10,11,12]
        end
        pic=imcomplement(img_raw_results_cell{i});
        result_to= zeros(size(r,2),size(shape_element,2))
        [X,Y] = meshgrid(shape_element_no,r)
        for j = 1:size(shape_element,2)
            for p = 1:size(r,2)
                se = strel(shape_element{j},r(p));
                if preprocessing_steps(i)==6
                    img = imcomplement(imopen(pic,se));
                elseif preprocessing_steps(i)==7
                    img = imcomplement(imerode(pic,se));
                elseif preprocessing_steps(i)==9
                    img = imtophat(imcomplement(pic),se);
                end
                result_pre=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(img,sample_no)));
                result_to(p,j)=result_pre(2);
```

```
            end
        end
        minMatrix = min(result_to(:));
        [row,col] = find(result_to==minMatrix);
        figure( 'Position', [10 10 900 600]);

subplot(3,2,1);mesh(X,Y,result_to,'FaceAlpha','0.8'),title('imopen()'),xlabel('shape_element'),yl
abel('r'),zlabel('result');
        result_to
        se = strel(shape_element{col},r(row));

result_O=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(img_raw_results_cell{i},sample_no)));
        if result_O(2)<minMatrix
            img_raw_results_cell{i+1}=img_raw_results_cell{i}
        else
            if preprocessing_steps(i)==6
                    img_raw_results_cell{i+1} =
imcomplement(imopen(imcomplement(img_raw_results_cell{i}),se));
            elseif preprocessing_steps(i)==7
                    img_raw_results_cell{i+1} =
imcomplement(imerode(imcomplement(img_raw_results_cell{i}),se));
            elseif preprocessing_steps(i)==9
                    img_raw_results_cell{i+1} = imtophat(img_raw_results_cell{i},se);
                    if mask_array~=-1
                        img_raw_results_cell{i+1}(mask_array) = 255;
                        img_raw_results_cell{i+1} =
Contrast_stretch_B_special(img_raw_results_cell{i+1});
                    end
            end
        end

    %8 Gaussian Blurring/Filtering
    elseif preprocessing_steps(i)==8
        sigma=linspace(0.5,4,10);
        result_to=zeros(1,size(sigma,2));
        for k=1:size(sigma,2)
            img=imgaussfilt(img_raw_results_cell{i},sigma(k));
            result_pre=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(img,sample_no)));
            result_to(k)=result_pre(2);
        end
        [Sigma_M,I] = min(result_to);
        Sigma_M
        sigma(I)%0.661
        img_raw_results_cell{i+1}=imgaussfilt(img_raw_results_cell{i},sigma(I))
    %10 Mask From Best Soltuion method
    elseif preprocessing_steps(i)==10
        A=find(preprocessing_steps==5);
        pic=img_raw_results_cell{A+1};
        pic_invert=(pic==0);
        se = strel('disk',3);
        dilatedI = imdilate(pic_invert,se);
        figure;imshowpair(pic_invert,dilatedI,'montage')

        prev_img=img_raw_results_cell{i}
        mask_array=find(dilatedI==0);

        prev_img(find(dilatedI==0)) = 255;
        img_raw_results_cell{i+1}=img_raw_results_cell{2}
    end

end
figure( 'Position', [5 5 1400 500]);
for i=1:size(img_raw_results_cell,1)

Results_overall(:,i)=cellfun(@double,cell(py.Overall_OCR.tesseractOCR(img_raw_results_cell{i},sam
ple_no)));
    if i==1;
        Process_Name=tool_Name{i};
    else
        Process_Name=tool_Name{preprocessing_steps(i-1)+1};
    end
```

```
    subplot(1,N+1,i);imshow(img_raw_results_cell{i});title([strcat(Process_Name,'\rightarrow')
,strcat('Levenshtein Dist: ', num2str(round(Results_overall(2,i),2)))]);
end

figure;imshowpair(img_raw_results_cell{1},img_raw_results_cell{N+1},'montage'),title([strcat('Ori
gnal Accuracy: ',num2str(round(Results_overall(1,1),2)),'% Levenshtein Dist:  ',
num2str(round(Results_overall(2,1),2))...
    , '. Vs .', tool_Name{preprocessing_steps(N)+1},num2str(round(Results_overall(1,N+1),2)),'%
Levenshtein Dist: ', num2str(round(Results_overall(2,N+1),2)))]);

S=4
figure;imshowpair(img_raw_results_cell{1},img_raw_results_cell{S},'montage'),title([strcat('Orign
al Accuracy: ',num2str(round(Results_overall(1,1),2)),'% Levenshtein Dist:  ',
num2str(round(Results_overall(2,1),2))...
    ,'. Vs .', tool_Name{preprocessing_steps(S-1)+1},num2str(round(Results_overall(1,S),2)),'%
Levenshtein Dist: ', num2str(round(Results_overall(2,S),2)))]);
```

10.3 Contrast_stretch_B.m

```
function stretched =Contrast_stretch_B(gray)
    min_P=double(min(gray(:)))%Min intensity=13
    max_P=double(max(gray(:)))%Max intensity=204
    stretched = uint8((double(gray(:,:))-min_P).*(255/(max_P-min_P)));
end
```

10.4 Contrast_stretch_B_special.m

```
function stretched =Contrast_stretch_B_special(gray)
    P=gray
    P(find(gray==255))=0;
    figure('Name',"10 Problem 1");imshow(P)
    min_P=double(min(P(:)))%Min intensity=13
    max_P=double(max(P(:)))%Max intensity=204
    P=uint8((double(P(:,:))-min_P).*(255/(max_P-min_P)));
    P(find(gray==255))=255;
    figure('Name',"10 Problem 2");imshow(P)
    stretched =P;
end
```

10.5 HOMO_Filtering_B.m

```
function Ihmf =HOMO_Filtering_B(gray)
%https://blogs.mathworks.com/steve/2013/06/25/homomorphic-filtering-part-1/
I = gray;
%imshow(I)
I = im2double(I);
I = log(1 + I);

M = 2*size(I,1) + 1;
N = 2*size(I,2) + 1;

%Creating High Pass Filter
sigma = 10;

[X, Y] = meshgrid(1:N,1:M);
centerX = ceil(N/2);
centerY = ceil(M/2);
gaussianNumerator = (X - centerX).^2 + (Y - centerY).^2;
H = exp(-gaussianNumerator./(2*sigma.^2));
H = 1 - H;
figure( 'Position', [10 10 900 600]);
subplot(1,2,1);imshow(H);title('High Pass Filter \sigma =10')
subplot(1,2,2);mesh(X,Y,H,'FaceAlpha','0.8'),title('High Pass Filter \sigma
=10'),xlabel('X'),ylabel('Y'),zlabel('Z');

H = fftshift(H);

If = fft2(I, M, N);
```

```
%Pass fft2 image through High Pass Filter
Iout = real(ifft2(H.*If));
Iout = Iout(1:size(I,1),1:size(I,2));

Ihmf = exp(Iout) - 1;
end
```

10.6 BW_adaptT.m

```
function BW =BW_adaptT(gray, sensitivity)
%https://www.mathworks.com/help/images/ref/adaptthresh.html#namevaluepairs
%Sensitivity: Determine which pixels get thresholded as foreground pixels,
%specified as a number in the range [0, 1].
%High sensitivity values lead to thresholding more pixels as foreground,
%at the risk of including some background pixels.
I=gray;

T = adaptthresh(I, sensitivity);
BW = (imbinarize(I,T));
%BW=double((I>100).*255);
%figure
%imshowpair(I, BW, 'montage')
end
```

10.7 Overall_OCR.py

```
#for pytesseract
#conda install -c conda-forge pytesseract
#https://medium.com/analytics-vidhya/performing-optical-character-recognition-with-python-and-pytesseract-using-anaconda-4bfe1ee6a75f
#for CV2
#conda install -c conda-forge opencv
#for Matlab
#https://stackoverflow.com/questions/46141631/running-matlab-using-python-gives-no-module-named-matlab-engine-error
import cv2
import pytesseract
pytesseract.pytesseract.tesseract_cmd=r'C:\Program Files\Tesseract-OCR\tesseract.exe'
import matplotlib.pyplot as plt
import numpy as np
#import jiwer
import matlab.engine
import sys
import scipy.io
import Levenshtein
from scipy.ndimage import interpolation as inter
import math

#https://nanonets.com/blog/ocr-with-tesseract/

# get grayscale image
def get_grayscale(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# noise removal
def remove_noise(image):
    return cv2.medianBlur(image,5)

#thresholding
def thresholding(image):
```

```
    return cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

#dilation
def dilate(image):
    image=np.asarray(image)
    kernel = np.ones((5,5),np.uint8)
    img_dialate=cv2.dilate(image, kernel, iterations = 1)
    data=np.ascontiguousarray(img_dialate)
    data=matlab.double(data.tolist())
    return data

#erosion
def erode(image):
    image=np.asarray(image)
    kernel = np.ones((5,5),np.uint8)
    img_erode=cv2.erode(image, kernel, iterations = 1)
    data=np.ascontiguousarray(img_erode)
    data=matlab.double(data.tolist())
    return data

#opening - erosion followed by dilation
def opening(image):
    image=np.asarray(image)
    kernel = np.ones((5,5),np.uint8)
    img_open=cv2.morphologyEx(image, cv2.MORPH_OPEN, kernel)
    data=np.ascontiguousarray(img_open)
    data=matlab.double(data.tolist())
    return data

#canny edge detection
def canny(image):
    image=np.asarray(image)
    return cv2.Canny(image, 100, 200)

def deskew(img):
    img=np.asarray(img)
    bin_img=cv2.adaptiveThreshold(img,1,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)
    delta = 1
    limit = 5
    angles = np.arange(-limit, limit+delta, delta)
    scores = []
    for angle in angles:
        hist, score = find_score(bin_img, angle)
        scores.append(score)
    best_score = max(scores)
    best_angle = angles[scores.index(best_score)]
    #plot
##    hist1, score1 = find_score(bin_img, 0)
##    hist2, score2 = find_score(bin_img, angle)
##    histo=[hist1,hist2]
##    plt.rcdefaults()
##    figure, ax = plt.subplots(nrows=1,ncols=2 )
##    for ind,title in enumerate(histo):
##        y_pos = np.arange(len(histo[ind]))
##        ax.ravel()[ind].barh(y_pos,histo[ind])
##        ax.ravel()[ind].set_xlabel('no. of black pixels')
##        ax.ravel()[ind].set_ylabel('Image row number')
##    plt.tight_layout()
##    plt.show()

```

```
    print('Best angle: {}'.format(best_angle))
    # correct skew
    data = inter.rotate(img, best_angle, reshape=False, order=0)
    hist_white = np.sum(data==0, axis=1)
    print(data)
##    plt.rcdefaults()
##    fig, ax = plt.subplots()
##    y_pos = np.arange(len(hist_white))
##    ax.barh(y_pos,hist_white)
##    ax.set_xlabel('no. of white pixels')
##    ax.set_ylabel('Image row number')
##    plt.show()
        (h,w)=img.shape

        mw,mh=rotatedRectWithMaxArea(w, h, abs(math.radians(best_angle)))
        dw=math.ceil((w-mw)/2)
        dh=math.ceil((h-mh)/2)
        data=np.ascontiguousarray((data))
        print(w,h)
        print(mw,mh)
        print(dw,dh)
        data=matlab.double(data[dh:h-dh,dw:w-dw].tolist())
        #data=np.asarray(data[dh:h-dh,dw:w-dw])
        #.tolist()
        return data


    def find_score(arr, angle):
        data = inter.rotate(arr, angle, reshape=False, order=0)
        hist = np.sum(data, axis=1)
        score = np.sum((hist[1:] - hist[:-1]) ** 2)
        return hist, score

    def rotatedRectWithMaxArea(w, h, angle):
      """
        https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders
      Given a rectangle of size wxh that has been rotated by 'angle' (in
      radians), computes the width and height of the largest possible
      axis-aligned rectangle (maximal area) within the rotated rectangle.
      """
      if w <= 0 or h <= 0:
        return 0,0

      width_is_longer = w >= h
      side_long, side_short = (w,h) if width_is_longer else (h,w)

      # since the solutions for angle, -angle and 180-angle are all the same,
      # if suffices to look at the first quadrant and the absolute values of sin,cos:
      sin_a, cos_a = abs(math.sin(angle)), abs(math.cos(angle))
      if side_short <= 2.*sin_a*cos_a*side_long or abs(sin_a-cos_a) < 1e-10:
        # half constrained case: two crop corners touch the longer side,
        #   the other two corners are on the mid-line parallel to the longer line
        x = 0.5*side_short
        wr,hr = (x/sin_a,x/cos_a) if width_is_longer else (x/cos_a,x/sin_a)
      else:
        # fully constrained case: crop touches all 4 sides
        cos_2a = cos_a*cos_a - sin_a*sin_a
        wr,hr = (w*cos_a - h*sin_a)/cos_2a, (h*cos_a - w*sin_a)/cos_2a

```

```
        return wr,hr

    #template matching
    def match_template(image, template):
        return cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)

    #OCReval Evaluate OCR results
    def OCReval(Actual,prediction):
        #Step 1:Preprocessing
        Actual=Actual.split("\n")
        prediction=prediction.split("\n")

        for idx, val in enumerate(prediction):
            prediction[idx]=val.split(" ")

        for idx, val in enumerate(Actual):
            Actual[idx]=val.split(" ")


        #Step 2:Evaluating Accuracy
        results=Actual.copy();
        for i, line in enumerate(Actual):
            for j, word in enumerate(line):
                try:
                    Actual_word=list(word)
                    OCR_word=list(prediction[i][j])
                except:
                    results[i][j]=np.array([False] * len(Actual_word))
                    continue
                if len(Actual_word)==len(OCR_word):
                    try:
                        match=np.array(np.array(OCR_word)==np.array(Actual_word))
                        results[i][j]=match
                    except:
                        results[i][j]=np.array([False] * len(Actual_word))
                else:
                    match= [True if i in OCR_word else False for i in Actual_word]

                    results[i][j]=np.array(match)

        #Step 3:Generating Accuracy Resuts
        Total=0
        Score=0
        for i, line_result in enumerate(results):
            for j, word_result in enumerate(line_result):
                Total=Total+len(word_result)
                if False in word_result:
                    Score=Score+sum(word_result)
                else:
                    Score=Score+len(word_result)


        return (Score*100/Total)

    #Display multiple Images
    #https://www.delftstack.com/howto/matplotlib/how-to-display-multiple-images-in-one-figure-correctly-in-matplotlib/
    def display_multiple_img(images, rows = 1, cols=1):
        figure, ax = plt.subplots(nrows=rows,ncols=cols )
        for ind,title in enumerate(images):
            ax.ravel()[ind].imshow(images[title])
```

```
                ax.ravel()[ind].set_title(title)
                ax.ravel()[ind].set_axis_off()
            plt.tight_layout()
            plt.show()

        def tesseractOCR(img,sample_no):
            results=[0,0]
            img=np.asarray(img)
            if sample_no==1:
                Actual="""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."""
            if sample_no==2:
                Actual="""A 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 cheeks 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, "Heck with it.  I'll just digitize."

        Thomas Colthurst"""

            custom_config = r'--oem 3 --psm 6'
            OCR_output=pytesseract.image_to_string(img, config=custom_config)
            print(OCR_output)
            results[0]=OCReval(Actual,OCR_output)
            results[1]=Levenshtein.distance(Actual, OCR_output)
            #print("Accuracy:", '%.2f'%(results[0]),"%" )
            print("Levenshtein:", '%.2f'%(results[1]))
            return results
```

11 References

[1] "python-Levenshtein-wheels 0.13.1," [Online]. Available: https://pypi.org/project/python-Levenshtein-wheels/.

[2] "pytesseract 0.3.6," [Online]. Available: https://pypi.org/project/pytesseract/.

[3] "How do I resolve a TesseractNotFoundError?," [Online]. Available: https://stackoverflow.com/questions/50655738/how-do-i-resolve-a-tesseractnotfounderror.

[4] S. Eddins, "Homomorphic filtering – part 1," [Online]. Available: https://blogs.mathworks.com/steve/2013/06/25/homomorphic-filtering-part-1/.

[5] S. Kumar, "A straightforward introduction to Image Thresholding using python," [Online]. Available: https://medium.com/spinor/a-straightforward-introduction-to-image-thresholding-using-python-f1c085f02d5e.

[6] "Levenshtein distance," wikipedia.org, [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance.

[7] S. Reddy, "Pre-Processing in OCR!!!," 05 2019. [Online]. Available: https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7.

[8] M. S.-A. M. Shafii, "M. Skew detection and correction based on an axes-parallel bounding box. IJDAR 18, 59–71," 2015. [Online]. Available: https://doi.org/10.1007/s10032-014-0230-y.

[9] coproc, "Rotate image and crop out black borders," 05 2013. [Online]. Available: https://stackoverflow.com/questions/16702966/rotate-image-and-crop-out-black-borders.

[10] "Homomorphic filtering," [Online]. Available: https://en.wikipedia.org/wiki/Homomorphic_filtering#:~:text=Homomorphic%20filtering%20is%20a%20generalized,by%20Thomas%20Stockham%2C%20Alan%20V..

[11] "OpenCV - Adaptive Threshold," tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/opencv/opencv_adaptive_threshold.htm#:~:text=Adaptive%20thresholding%20is%20the%20method,()%20of%20the%20Imgproc%20class..

[12] "adaptthresh," mathworks, [Online]. Available: https://www.mathworks.com/help/images/ref/adaptthresh.html#description.

[13] "Types of Morphological Operations," mathworks, [Online]. Available: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html.

[14] J. Wright, "Spelling correction with Levenshtein distance," 08 2018. [Online]. Available: https://bytes.babbel.com/en/articles/2018-10-23-levenshtein-distance.html#:~:text=Like%20most%20algorithms%2C%20Levenshtein%20distance,English%20is%20easy%20to%20misspell..

[15] "Wagner–Fischer algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Wagner%E2%80%93Fischer_algorithm.

[1 Foteini Filippidou, "A Benchmarking of IBM, Google and Wit Automatic Speech Recognition
6] Systems," 05 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7256403/.