# 1. Introduction

# 2. Methodology

### 2.1. Otsu’s Algorithm

### 2.2. Rescale

According to the official documentation of Tesseract, it is suggested that the input image is at 300 DPI to achieve the OCR engine’s best performance. We adopted the Python Imaging Library (PIL) to resize the original image to the suggested 300 DPI.

### 2.3. Median Blur

We will implement a median filter with kernel size 3 to enhance the quality of the image by removing the noise in the image.

### 2.4. Erosion, Dilation and Opening

We adopted morphological transformations including erosion, dilation, and the combination of them which is opening. Each of them will have a fixed sized square kernel defined.

Erosion refers to the transformation which the kernel slides through the image, and the original image will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded to zero. The effect is equivalent of thinning bold strokes in the image. Dilation on the other hand is the opposite of erosion, where a pixel is 1 if at least one pixel under the kernel is 1, thus it increases the region in the image or increases the size of the stroke in the case of OCR.

We will be implementing an Opening to the image, which is equivalent to applying an erosion followed by dilation. This technique is useful in removing noise.

### 2.5. Adaptive Thresholding

We adopted adaptive thresholding instead of global thresholding. With regard to the image that we intend to enhance and carry out OCR upon, global thresholding which sets one global thresholding value does not produce expected outcome because of the variation of illumination in the image. We can observe and compare the effect of global thresholding and the original image.

|  |  |
| --- | --- |
| Original Image |  |
| After Global Thresholding |  |

We can observe from the above comparison that, because the illumination of the right of the original image was darker compared to the left of the image, global thresholding rendered the right half of the image to black, resulting in a severe loss of information and text. Thus, global thresholding could not be used as a binarization algorithm in this case.

Hence, we decided to adopt adaptive thresholding, which calculates the threshold for a small region of the image, to improve the quality of binarization.

We implemented the ‘cv2.adaptiveThreshold’ method from python module cv2 to achieve adaptive thresholding.



The semantics of the parameters are given by:

|  |  |
| --- | --- |
| Parameter Name |  |
| src | Source 8-bit single-channel image. |
| dst | Destination image of the same size and the same type as src. |
| maxValue | Non-zero value assigned to the pixels for which the condition is satisfied. |
| adaptiveMethod | Adaptive thresholding algorithm to use. We will experiment with different both of the two algorithms to choose the best performance one. |
| thresholdType | Thresholding type that must be either THRESH\_BINARY or THRESH\_BINARY\_INV.  In the scope of this project, we will adhere to the default thresholding type, which is THRESH\_BINARY |
| blockSize | Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on. |
| C | Constant subtracted from the mean or weighted mean. |

### 2.6. Deskewing Image

We can observe a slight distortion in the angle of the original sample image. We adopted a deskewing algorithm to deskew the image and enhance the OCR engine’s performance.

The steps of the algorithm are as following:

Detect the text blocks of the image by invert and maximize the colors of the image so that the text area will be whitened while the background remains black.

Merge the characters of the block by applying a dilation with larger kernel on x to eliminate spaces between words and smaller kernel on y to preserve the spacings between text blocks.

Apply contour detection with min area rectangle enclosing the text blocks that we produced. Select the angle produced from the largest block and set it as the skewing angle. Rotate the image with the angle determined.

### 2.7. Definition of Accuracy

We define the accuracy of the result string by the Levenshtein distance between it and the original text string. By definition, the Levenshtein distance between two words is the minimum number of single-character edits, including insertions, deletions and substitutions required to convert the original string to another. The higher the result Levenshtein distance is, the less accurate the conversion result.

# 3. Experiments and Results

### 3.1. Design of the algorithm

We designed the pipeline of the algorithm in accordance with the methodology mentioned in the methodology section.

The pipeline of the algorithm is demonstrated as following:



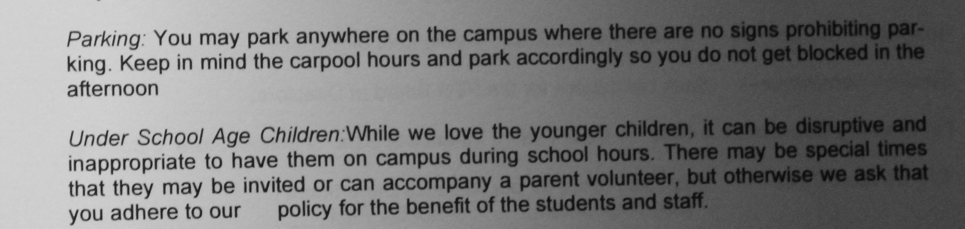
#### 3.1.1. Resize

We adopted the following code to implement resizing the image:



This way we modified the dpi of the image to 300, adhering to the suggestion of Tesseract.

The image after resizing is displayed below:



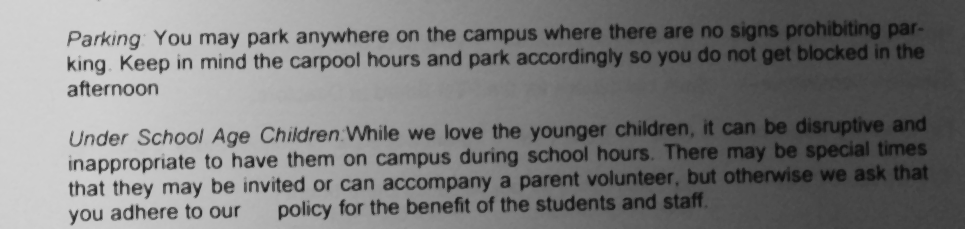
#### 3.1.2. Median Blur

We adopted the following code to implement median blur:



With initial trials with median filter kernel size, we decided to set the kernel size to 3. Kernel size larger than 3 will render many characters unclear.

The image after median filter is displayed below:



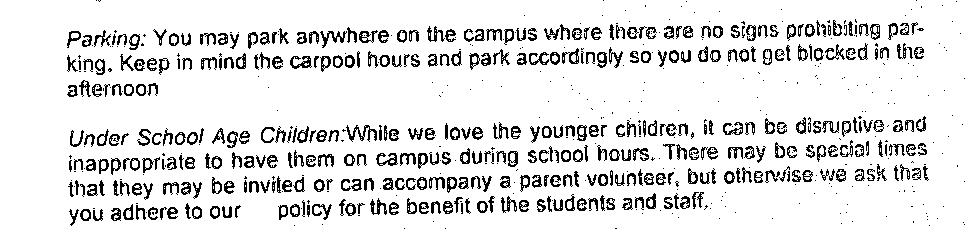
#### 3.1.3. Adaptive Thresholding

We adopted the following code to implement adaptive thresholding:



As mentioned in the methodology section, the last two parameters indicate the size of the kernel, and the value to be subtracted after the Gaussian thresholding method. We implemented grid search for hyperparameter optimization for these two parameters in the later section.

The image after adaptive thresholding is displayed below:

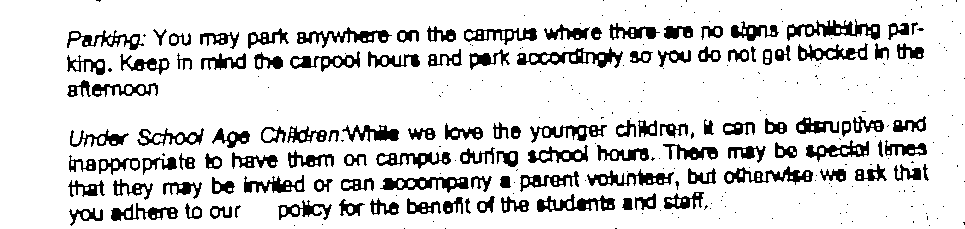


#### 3.1.4. Opening

We adopted the following code to implement opening:



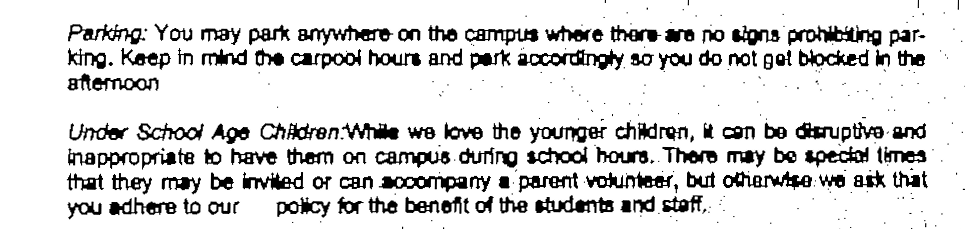
The image after opening is displayed below:



#### 3.1.5. Deskewing

We adopted the following code to implement deskewing:



The image after Deskewing is displayed below: 

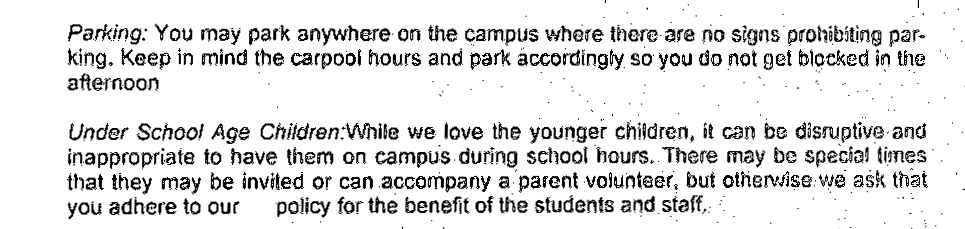
### 3.2. Recognition Accuracy after each Pass

We recorded the accuracy in Levenshtein distance after each pass of the step in section 3.1, and computed the enhancement of each pass:

|  |  |  |
| --- | --- | --- |
| Step | Levenshtein distance | Difference |
| Resizing | 259 | 0 |
| Median Blur | 263 | +4 |
| Adaptive Thresholding | 20 | -243 |
| Opening | 40 | +20 |
| Deskewing | 33 | -7 |

We can observe from the above result that the accuracy has a significant improvement after binarization using adaptive thresholding and deskewing, but the performance decreased after applying opening to the image. We decided to skip the opening step and see if there is an improvement in the prediction accuracy and found out the new Levenshtein distance after deskewing is 17, which is an enhancement.

The result processed image is:



The result recognized string is:

*“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 i 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 ‘patent volunteer, but otherwise. we ask that*

*you adhere to our \_ policy for the benefit of the students and staff,”*

### 3.3. Hyperparameter Optimization

We implemented grid search to determine the optimal kernel size and constant “c”:



|  |  |  |
| --- | --- | --- |
| Kernel Size | Constant C | Levenshtein distance |
| 3 | 0 | 500 |
| 1 | 522 |
| 2 | 20 |
| 3 | 15 |
| 4 | 73 |
| 5 | 155 |
| 6 | 156 |
| 7 | 191 |
| 8 | 251 |
| 9 | 289 |
| 5 | 0 | 522 |
| 1 | 522 |
| 2 | 39 |
| 3 | 14 |
| 4 | 12 |
| 5 | 20 |
| 6 | 48 |
| 7 | 100 |
| 8 | 112 |
| 9 | 164 |
| 7 | 0 | 522 |
| 1 | 522 |
| 2 | 116 |
| 3 | 18 |
| 4 | 10 |
| 5 | 14 |
| 6 | 14 |
| 7 | 24 |
| 8 | 80 |
| 9 | 107 |
| 9 | 0 | 406 |
| 1 | 522 |
| 2 | 114 |
| 3 | 25 |
| 4 | 16 |
| 5 | 13 |
| 6 | 13 |
| 7 | 13 |
| 8 | 11 |
| 9 | 23 |
| 11 | 0 | 433 |
| 1 | 522 |
| 2 | 524 |
| 3 | 35 |
| 4 | 18 |
| 5 | 13 |
| 6 | 13 |
| 7 | 12 |
| 8 | 11 |
| 9 | 36 |

We can observe from the records above a pattern that the Levenshtein distance decreases with the increase of constant C then increase. We selected the best pair of Kernel Size and Constant C to be (7,4) and feed the result to deskewing algorithm and found out the Levenshtein distance increased to 11, indicating there is no improvement after deskewing.

The result string of the optimal hyperparameters is given by:

*“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 specia! 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.”*

# Conclusion