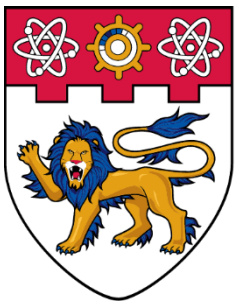


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4003: Computer Vision

Project:

# Text Image Segmentation for Optimal Optical character recognition

Lin HuangJiayin

U1721208D

# Table of Contents

.

## Glossary

| Terms | Definition |
|---|---|
| Pure | A pure method will be considered an image that went through a method without any filtering. It is a direct transformation of the image with the algorithm.<br><br>e.g.<br>- Pure Otsu<br>  ○ Direct application of Otsu algorithm on image<br>- Pure Canny<br>  ○ Direct application of Canny algorithm on the image |
| A + B | A: Variable<br><br>B: Variable<br><br>A is applied after B<br><br><br>e.g.<br>- Closing + Adaptive Mean Threshold + Median Filter<br>  ○ Step 1: Median Filter on image<br>  ○ Step 2: Adaptive Mean Threshold on result of step 1<br>  ○ Step 3: Closing on result of step 2 |
| algorithm.py | All algorithms used, excluding Otsu, |
| ostu_algorithm.py | Only Otsu algorithm |
| tesseract.py | To call Tesseract |

# Methods

## Algorithm

Otsu algorithm

Otsu algorithm follows the global thresholding logic. It iterates through possible threshold values to calculate a value that maximize between class variance, which is the same as minimizing within class variance. Once the threshold value is calculated, any intensity above the threshold will be converted to 255, otherwise, 0 .

The algorithm has been implemented by following the equations provided in Otsu Method and K-means by D. Liu and J. Yu, a paper that was published in 2009 during the 2009 Ninth International Conference on Hybrid Intelligent Systems, Shenyang.

Total number of pixels:

$$N = n_1 + n_2 + \cdots + n_L$$

Probabiliy of gray level i:

$$p_{i=} n_i/N$$

Gray level probability distributions:

$$w_1 = \Pr(C_1) = \sum_{i=0}^{t} p_i$$

$$w_2 = \Pr(C_2) = \sum_{i=t+1}^{L-1} p_i$$

Means of class C1 and C2:

$$u_1 = \sum_{i=0}^{t} ip_i \ /w_1$$

$$u_2 = \sum_{i=t+1}^{L-1} ip_i \ /w_2$$

Total mean of gray levels:

$$u_t = w_1u_1 + w_2u_2$$

Class variances:

$$\sigma^2{}_1 = \sum_{i=0}^{t} (i - u_1)^2 p_i \ /w_1$$

$$\sigma^2{}_2 = \sum_{i=t+1}^{L-1} (i - u_2)^2 p_i \ /w_2$$

Within-class variance:

$$\sigma^2{}_w = \sum_{k=1}^{M} w_k \, \sigma^2{}_k$$

Between-class variance:

$$\sigma^2{}_w = w_1 \, (u_1 - u_t)^2 + w_2 \, (u_2 - u_t)^2$$

The optimal threshold t will be the maximum of the between-class variance, which is equivalent to minimizing the within-class variance.

## Canny algorithm

Canny algorithm is an edge detection multi-stage algorithm. The individual stages are used to filter noise, varying thickness and varying strength. Noise will be filtered out with the aid of Gaussian Edge filtering. Varying thickness will be reduced by non-maximal suppression. Varying strength will be reduced by having 2 threshold values.

## Median filtering

Median filter is a nonlinear filter. It is known for its ability to remove salt-and-pepper noise while preserving edges. The central element has its intensity taken from the median of its neighbouring pixels.

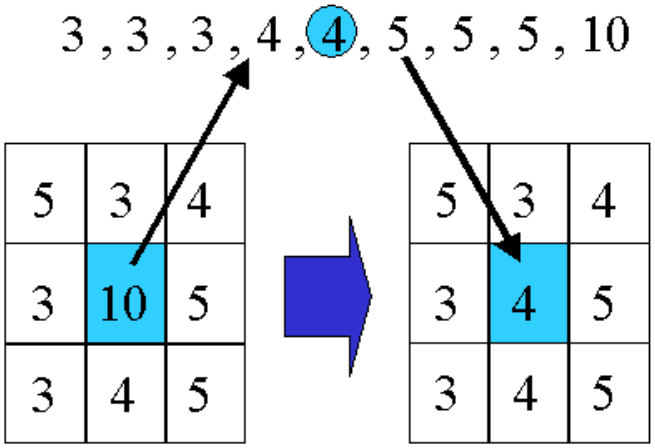



*Figure: Median Filtering*

## Bilateral filtering

Bilateral filtering makes use of 2 Gaussian filter, Gaussian function of space and Gaussian function of intensity difference. It is highly effective in noise removal while keeping edges sharp. Gaussian function of space ensure that only neighbouring pixels are considered for blurring, while Gaussian function of intensity difference ensure that only pixels with similar intensities to the centre is used for blurring.

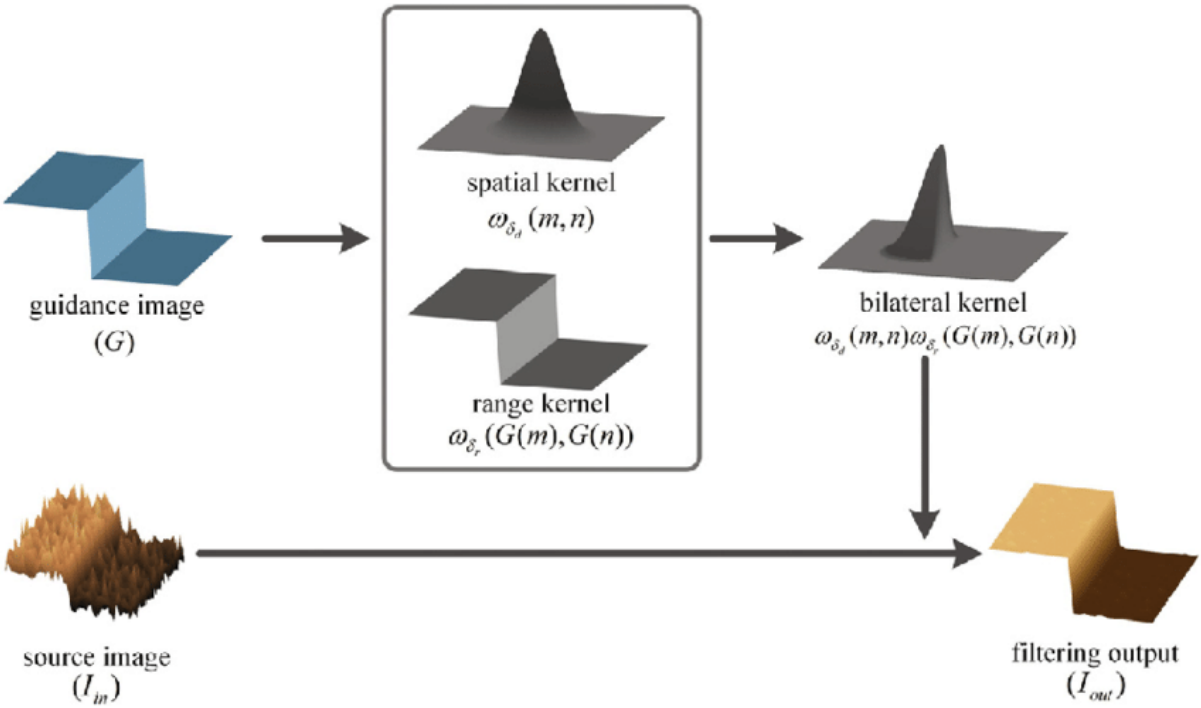



*Figure: Bilateral Filtering*

## Morphological transformation: Erosion

Erosion removes the boundaries of foreground object, such that all pixels near the boundary will be removed. It is useful for removing small white noises.

*Figure: Erosion*

## Morphological transformation: Dilation

Dilation is the opposite of erosion. It increases the size of foreground objects.

*Figure: Dilation*

## Morphological transformation: opening

Opening is a combination of erosion followed by dilation Using dilation after erosion allows the shrank/ broken parts to be re-joined.

*Figure: Opening*

## Morphological transformation: closing

Closing is the reverse procedure of Opening, Dilation followed by Erosion. It is useful for closing small noise(holes) inside the foreground object

*Figure: Closing*

## Tesseract

Tesseract is an OCR engine that allows for character recognition. In order to have an optimal recognition, the text needs to be in black, while the rest are in black.

The settings used for this paper will be the default settings.

## Accuracy

The accuracy is purely calculated by the number of words Tesseract detected correctly, taking into consideration of spelling. The error rates are not taken into consideration.

For instance:

- ving Keepin mund the campoot hours: Gets an accuracy score of 2
- younger on Ceo mRCANmP CRC SM Olena): Gets an accuracy score of 2

## Input

### sample01

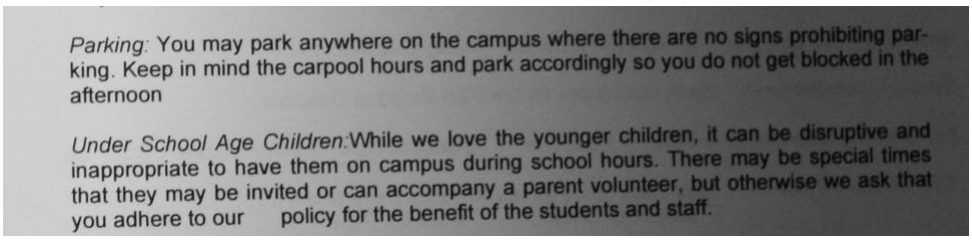



*Figure: sample01*

sample01 consists of a text image, with a gradient background.

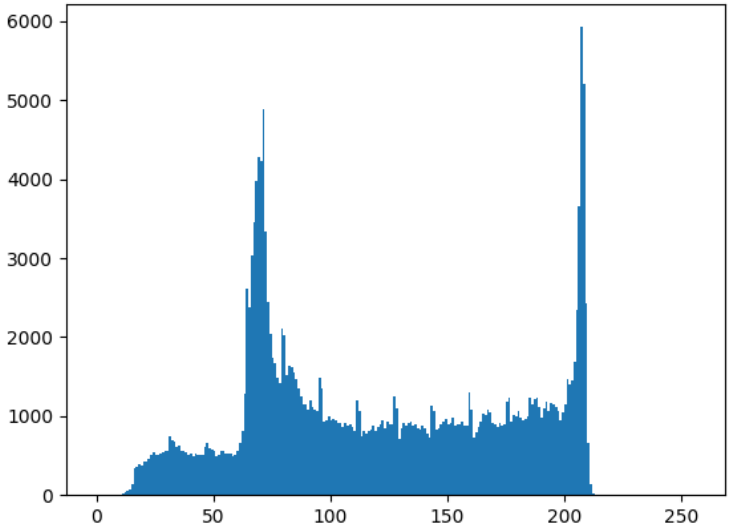



*Figure: sample01's histogram*

The intensity is fairly distributed in the middle, with 2 peaks at 75 and 210

The following text have been returned when Tesseract is used on sample01

```
Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of


♀ ♀
```

Accuracy: 46/91

## sample02

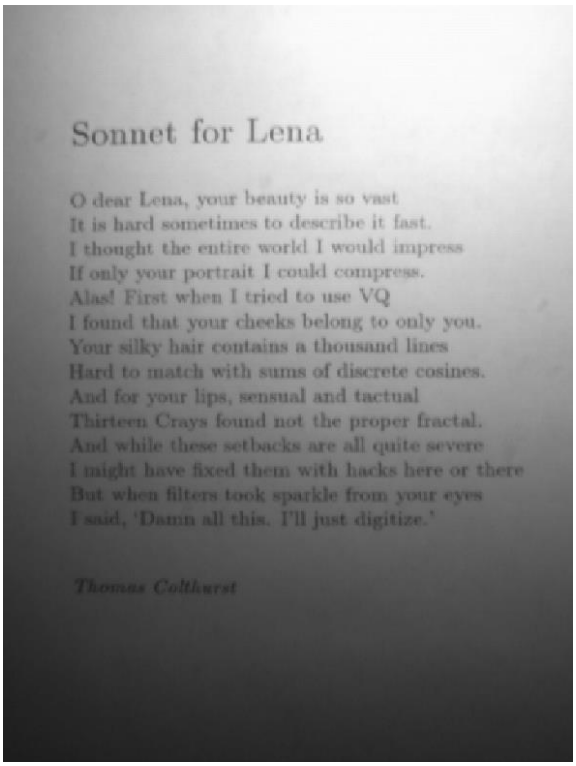



*Figure: sample02*

sample02 consists of a text image with a luminated background.

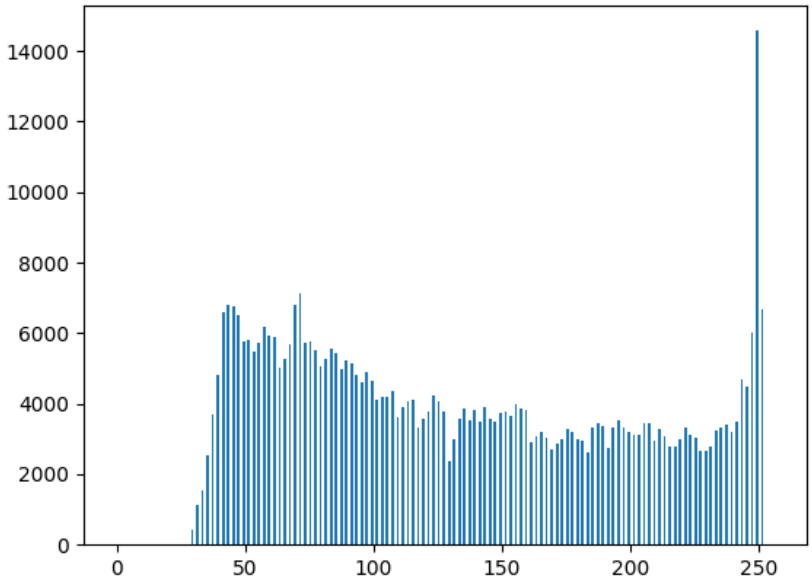



*Figure: sample02's histogram*

The intensity is fairly distributed in the middle, with 1 peak at 250

Tesseract is unable to detect any text

## Otsu global thresholding algorithm

```
# Total number of pixels
def N_pixels():
  N = 0
  for i in range(len(histogram)):
    if (histogram[i]>0):
      N = N + histogram[i]

  return N


# probability at gray level i
def prob(i):
  N = N_pixels()
  p = histogram[i]/float(N)

  return p


# gray level possibility distributions
def weight(a,b):
  w = 0
  for j in range(a,b):
    p = prob(j)
     w = w +p

  return w


# mean
def mean(a,b):
  w = weight(a,b)
  m = 0
  u = 0
  for j in range(a,b):
    p = prob(j)
    m = m + p
```

```
  u = m/float(w)

  return u

# gray level probability distributions
def gray_lvl(w1,u1,w2,u2):
  ut = w1 * u1 + w2 * u2

  return ut


# total mean of gray levels
def gray_lvl(w1,u1,w2,u2):
  ut = w1 * u1 + w2 * u2

  return ut


# class variance
def class_var(a,b):
  v = 0
  a = 0
  u = mean(a,b)
  w = weight(a,b)
  for j in range(a,b):
    p = prob(j)
    v = v + ((j - u)**2) * p

  a = v/float(w)

  return a


# within-class variance
def within_class_var(k,m):
  aw = 0
  for j in range(k,m):
    w = weight(j,m)
```

```
    a = class_var(j,m)
    aw = aw + w*a

  return aw


# between class var
def betw_class_var(w1,u1,w2,u2):
  ut = gray_lvl(w1,u1,w2,u2)
  if (math.isnan(ut)):
    ut = 0
  print(ut)
  ab = (w1 * (u1 - ut)**2) + (w2(u2-ut)**2)
  print(ab)
  return ab
```

# Experiments: sample01.png

## 1.Otsu algorithm

### 1.1. Application of Otsu Algorithm on sample01

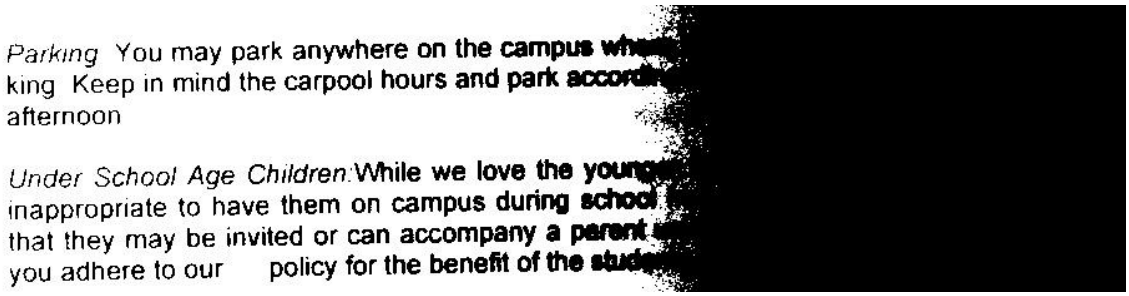



*Figure: sample01 after applying Otsu Algorithm*

The optimal threshold has been calculated to be of 105.467 (3dp), and 105 have been used. Any pixel with a pixel intensity of over 105 will be converted to 255(black).

While the texts are now black, 1/3 of the image has become totally black. The right side of the text has been merged with the background.

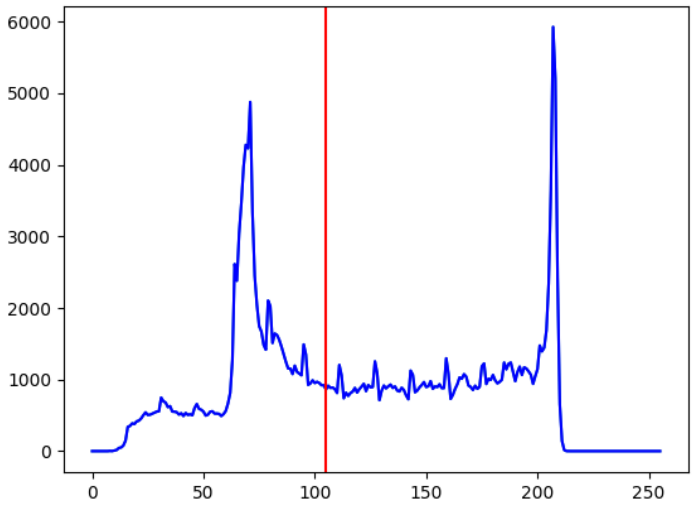



*Figure: Histogram of "sample01.png" with threshold*

Upon applying Tesseract on the image, the following is received:

```
Parking You may park anywhere on the campus
king Keep in mind the carpool hours and park a
afternoon
```

Under School Age Children:While we love the
inappropriate to have them on campus during schoo
that they may be invited or can accompany a parent am
you adhere to our _ policy for the benefit of the stutqiy
♀

Accuracy: 53/91

There is an increase in accuracy after applying Otsu algorithm. However, its accuracy is limited to the left side, as the right side is fully in black.

Problem with Otsu algorithm is that the optimal threshold is not “optimal”. It is a very inflexible algorithm such that any pixel above a certain threshold gets returned as black. A higher threshold will form an image with more black pixels.

## 1.2. Application of threshold of 70 on sample01

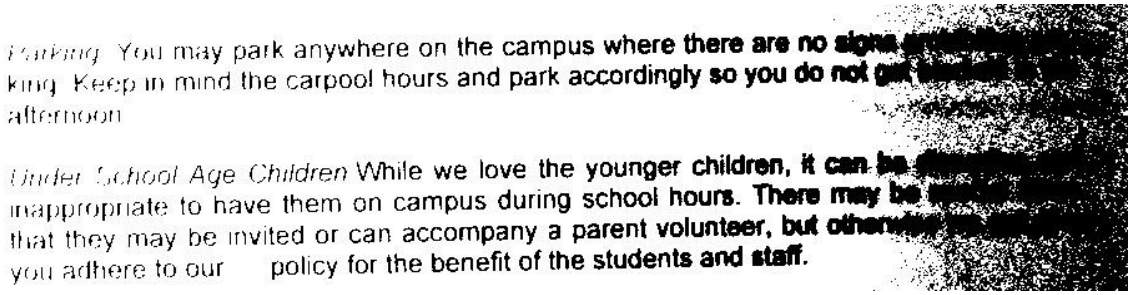



*Figure: sample01 with threshold of 70*

More pixels on the left are of value 0, and less pixels on the right are of value 255. The text on the left side has gotten thinner

A threshold of 70 results in less black values.

Upon applying Tesseract on the image, the following is received:

ianiigy You may park anywhere on the campus where there are no sig
king Keep in mind the carpool hours and park accordingly $0 you do not

afternoon

Binder Gctiool Age Children While we love the younger children, #t can beg
inappropriate to have them on campus during schoo! hours. There may be
that they may be invited or can accompany a parent volunteer, but
you adhere to our —_ policy for the benefit of the students and staff.

Accuracy: 68/91

The accuracy is higher with a threshold of 70 over 105.

## 1.3. Application of threshold of 60 on sample01

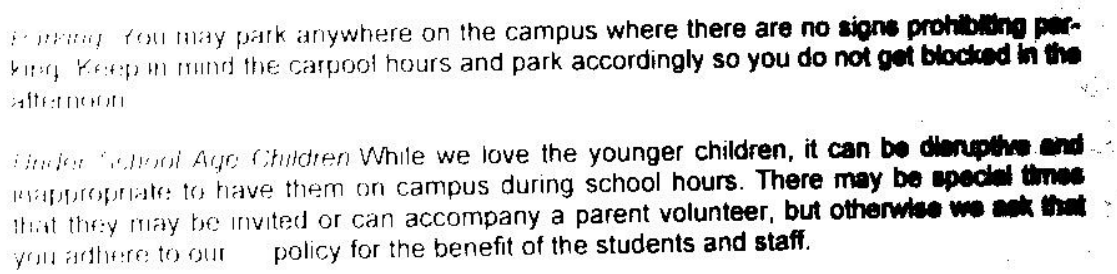



*Figure: sample01 with threshold of 60*

Only the text is left in black. There is an increase of strength and thickness, of the text, from the left to the right

Upon applying Tesseract on the image, the following is received:

```
boneiny fou may park anywhere on the campus where there are no signs prohibiting per-
ving Keepin mund the campoot hours and park accordingly so you do not get blocked in the

afternoon


Inappropriate to have them on campus during school hours. There may be special tines
Ihat they may be invited or can accompany a parent volunteer, but otherwise we aek thal -

you adhere to Our policy for the benefit of the students and staff.

jer ehool Age Chidren While we love the younger children, it can be dierupthes and ..-
```

Accuracy: 75/91

There is an increase in accuracy for the right side of the image, however, it is at the trade-off of the left side. Furthermore, while there is an increase in accuracy, it is essential to take note that the start of the second paragraph has been moved to the bottom

While altering the threshold manually can result in a better accuracy, it is a very dirty method of obtaining accuracy.

As such, due to the difference in background, having a global thresholding method is not suitable, resulting in a less than optimal result

## 2. Canny Algorithm

Canny algorithm has been tested due to its noise, thickness, and strength filtering properties.

### 2.1. Application of Canny Algorithm on sample01

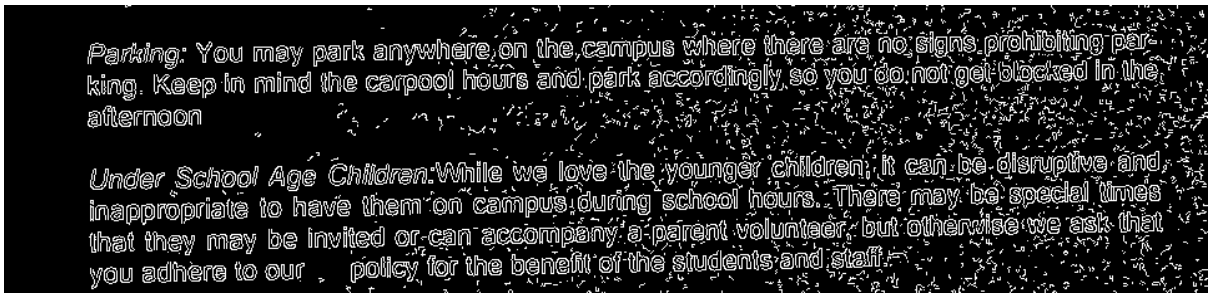



*Figure: Canny Algorithm with a threshold value of 20-30*

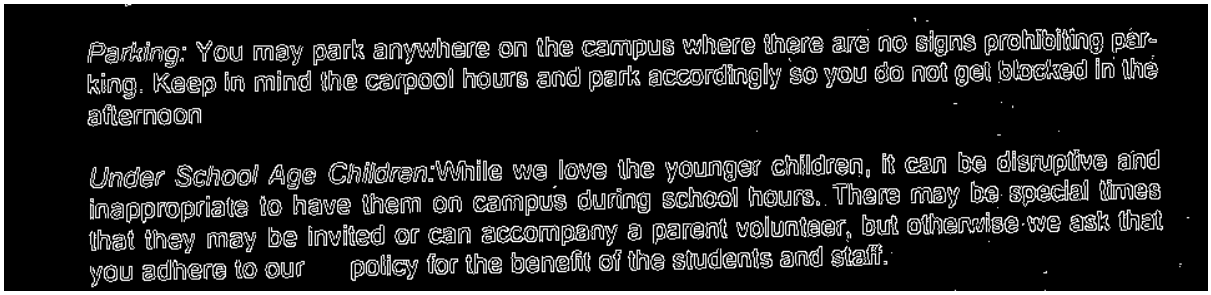



*Figure: Canny Algorithm with a threshold value of 40-50*

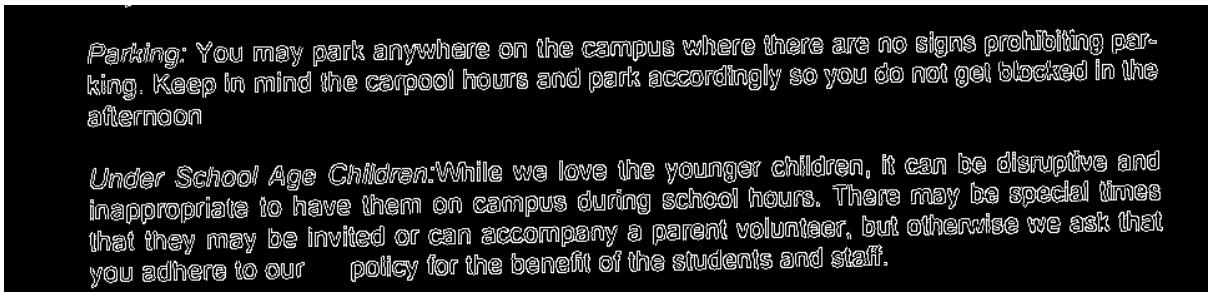



*Figure: Canny Algorithm with a threshold value of 50-75*

When the threshold is increased, noise gets removed, however, the details in the picture also increases.

A threshold value of 40-50 will be used for future experiments

Upon applying Tesseract on the image, the following is received:

| |
|---|
| Parking: You may park anywhere on the campus Pe neemncKcRcNe na ea OMON gms<br>FM enn mck n UC ENS park aceordingly so you do not gel blocked iiamliare)<br>Eiienneleg) : |

Under Schoo! Age Children: While we love the younger on Ceo mRCANmP CRC SM Olena)
inappropriate to have them on campus Ce ORsc ROOM ne ta mmc nmin moc) times
that they may be invited or can accompany & parent volunteer, but otherwise we ecialnals
Pci h komm Oo for the benefit of the students and staff.
♀ ♀

Accuracy: 51/91

The accuracy is lower and nosier than that of Otsu algorithm

## 2.2. Application of Dilation after Canny algorithm

To reduce the details in the image, dilation is used on the image threshold of 40-50

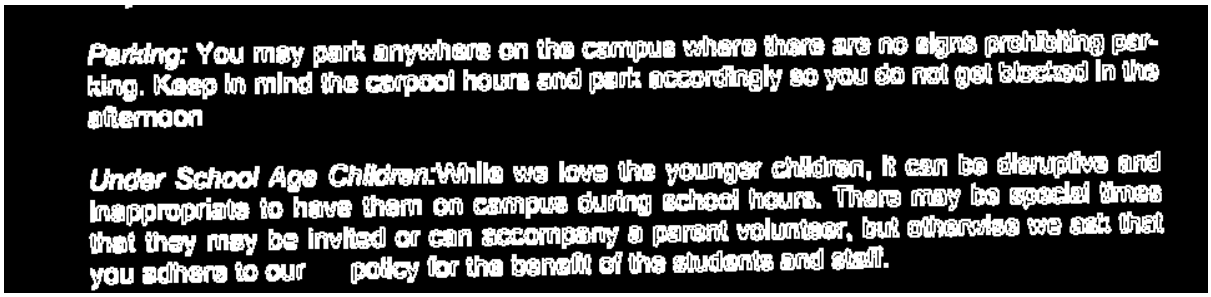



*Figure: Dilation after Canny Algorithm*

There is no difference in the accuracy before or after dilation

## 2.3. Application of Color Inversion after Canny Algorithm

As Tesseract favours black text, the image, without dilation, is inverted

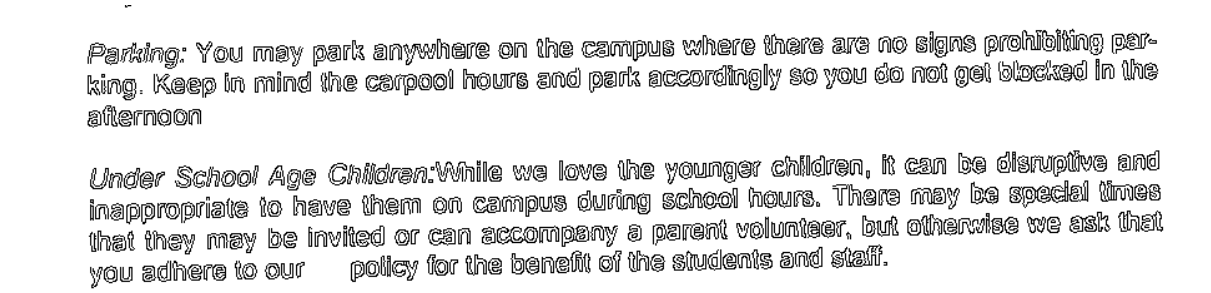



*Figure: Color inversion after Canny Algorithm*

Text boulder is now black

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the campus where there are no signs prohibiting aer-
king. Keep in mind the carpool hours and park acgordingly so you do not gel blocked fa the
atterneon

Under Schoo! Age Children:While we love the younger children, ik can be disruptive and
inappropriate to have them on campus during schoo! hours. There may be times
that they may be invited or can accompany & parent volunteer, but otherwise we ash thet
you adhere to our ~— policy for the benefit of the students and staf.
```

Accuracy: 77/91

The accuracy increased greatly with color inversion.

Thus far, inversion of Canny algorithm with a threshold of 50-75 provides the highest accuracy and least error.

## 3. Median Filtering

Median filtering has been tested due to its ability to remove salt-and-pepper noise. This is in hope of having a more equalised pixel intensity among the texts.

### 3.1. Application of Median Filtering

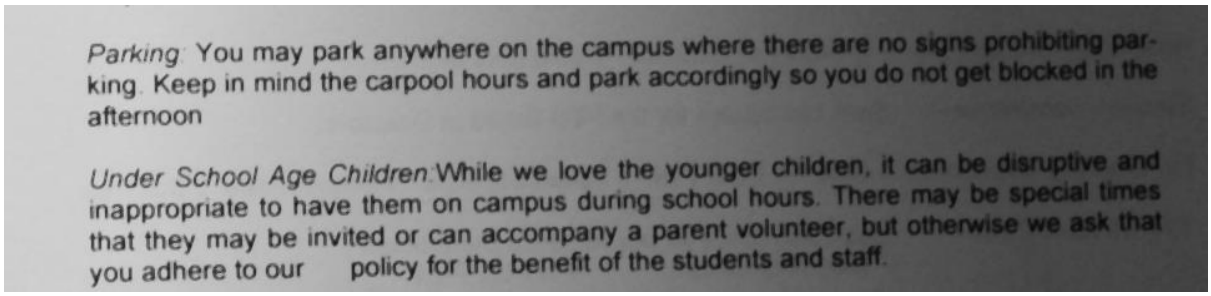



*Figure: Median Filter with a neighbour pixel size of 3*

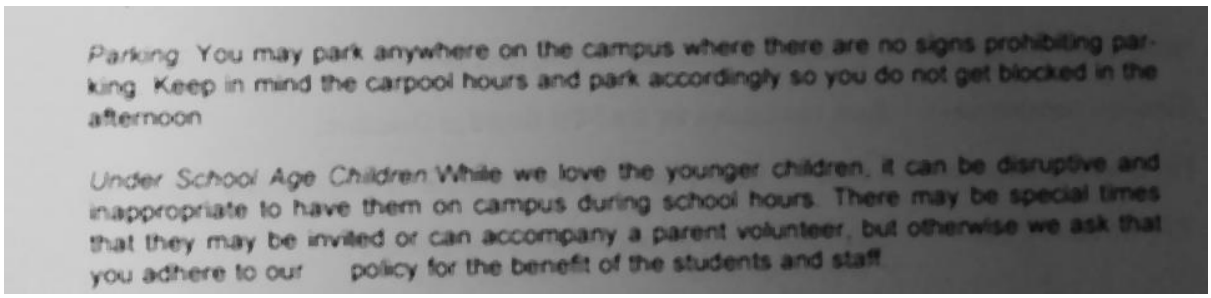



*Figure: Median Filter with a neighbour pixel size of 5*

When the kernel size is increased, the text shrinks. However, the test is more evenly distributed.

A kernel size of 3 will be used for future experiments

Upon applying Tesseract on the image, the following is received:

```
 Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of
```

Accuracy: 46/91

Only the left-hand side of the image has been returned. Its accuracy is lower than that of Otsu algorithm even though both are restricted to the left.

## 3.2. Application of Otsu algorithm after median filtering

The optimal threshold value did not change.

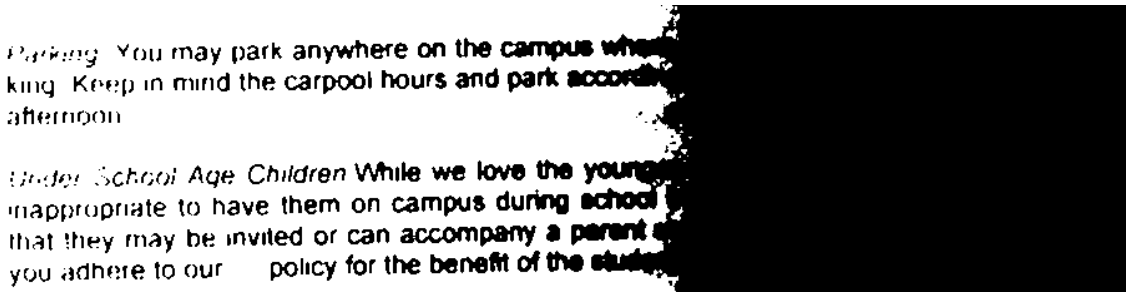


*Figure: Otsu Algorithm + Median Filter*

The left side of the text now consists of white(0) components, as compared to a pure Otsu algorithm

Upon applying Tesseract on the image, the following is received:

```
Paseng You may patk anywhere on the campus
king Keep in mind the carpool hours and park
aflernoon

tindes School Age Chidren While we love the
inappropriate to have them on campus during
that they may be invited or can accompany perent
you adhere toour policy for the benefit of the
```

Accuracy: 43/91

The accuracy is much worse after Otsu algorithm, and much lower than a pure Otsu algorithm(without filter)

## 3.3. Application of Canny algorithm after median filtering

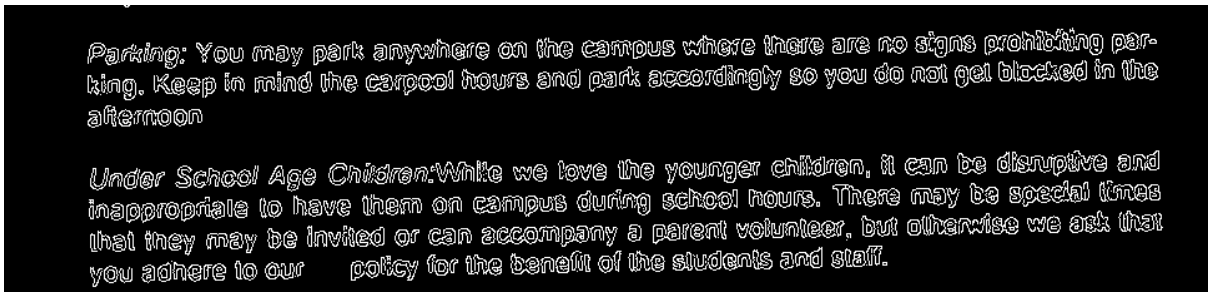



*Figure: Canny algorithm with a threshold value of 40-50 + Median Filter*

The result is like that before median filtering (pure Canny algorithm), but with more noise around the boundary and inside of the text

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the campus where {hese are RO Fern SNOGO OC ORO
rome ore ROMA MCR Oo BLOB) Doocecue LON EC On CnC ne Roc MOeccc hone)
ET eumncloa)

Ue MSC OS RAR OD ions CR. em CR Lod cnildren, ft can be disiuptive and
inappropdale to have them on campus curing schoo! hours. Thera may be seeclal (mes
thal they may be invited or can accompany a ore ave talc comb mo hone LecRMcROS ebay
DOU ON Cien Oko mmm COLo for the banat of the students ard staff.
♀
```

Accuracy: 34/91

After median filtering, the text is much noiser as compared to pure Canny algorithm. Thus far, this has provided the most noise.

## 4. Bilateral Filtering

Median filtering has been tested due to its ability to remove noise and preserve edges. This is in hope of having a more equalised pixel intensity among the texts.

### 4.1. Application of Bilateral Filtering

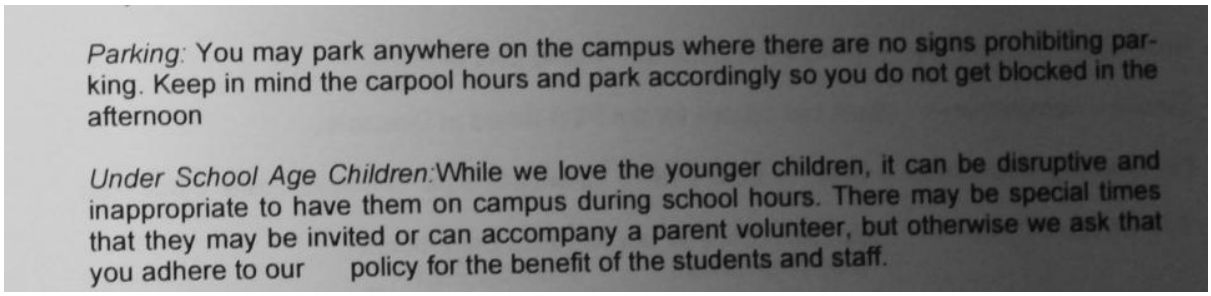



*Figure: Bilateral Filter with a neighbouring pixel size of 3*

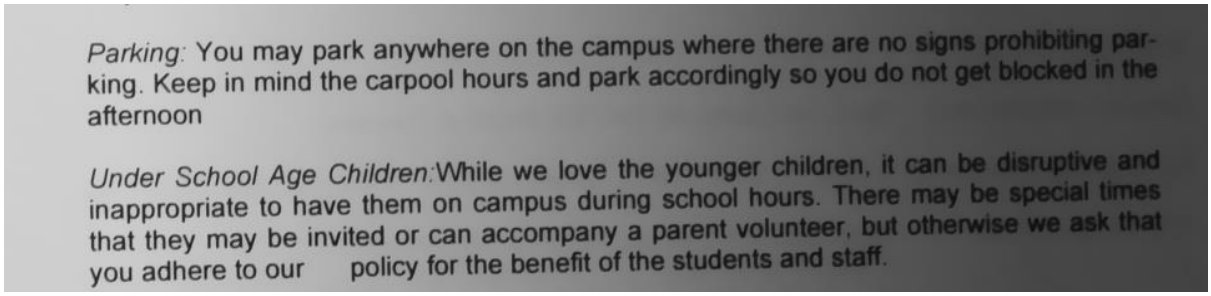



*Figure: Bilateral Filter with a neighbouring pixel size of 17*

When the neighbour pixels increase, the text is more even out (“smooth”).

A kernel size of 17 will be used for future experiments

Upon applying Tesseract on the image, the following is received:

```
 Parking: You may park anywhere on the
king. Keep in mind the carpool hours and p
afternoon

Under School Age Children:While we love
inappropriate to have them on campus d
that they may be invited or can accompany
you adhere to our _ policy for the benefit
```

Accuracy: 44/91

The accuracy is around the range of Median Filtering

## 4.2. Application of Otsu algorithm after Bilateral Filtering

Optimal threshold remains the same

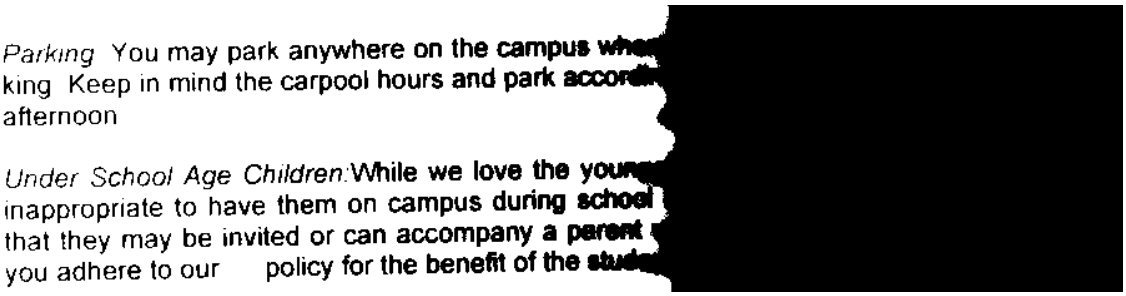



*Figure: Otsu algorithm + Bilateral Filter*

The results look similar to a pure Otsu algorithm, with the left side text being thinner

Upon applying Tesseract on the image, the following is received:

```
 Parking You may park anywhere on the campus
king Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love the y
inappropriate to have them on campus during
that they may be invited or can accompany 4 perent
you adhere to our _ policy for the benefit of the
♀
```

Accuracy: 50/91

The accuracy is higher after Otsu algorithm, however, still lower than a pure Otsu algorithm(without filtering)

## 4.3. Application of Canny algorithm after Bilateral Filtering

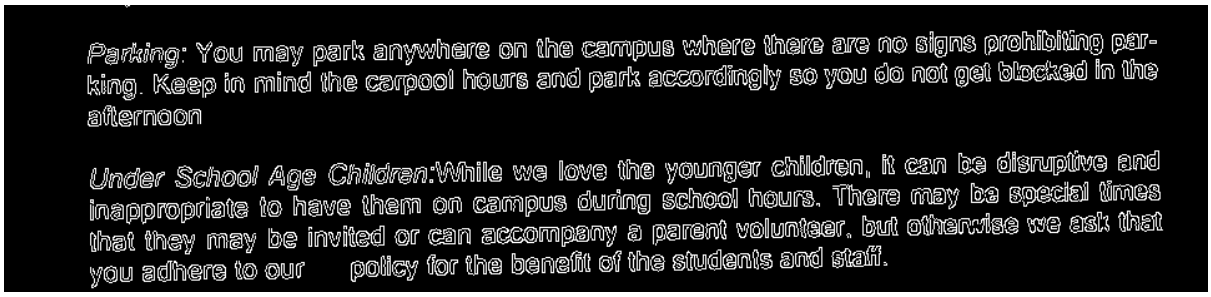



*Figure: Canny Algorithm with a threshold value between 40-50*

The result is similar to Median Filter.  The image consists of less noise (cleaner) as compared to the image without bilateral filtering

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the
king. Keep in mind the carpool hours and p
afternoon

Under School Age Children:While we love
inappropriate to have them on campus d
that they may be invited or can accompany
you adhere to our _ policy for the benefit
```

Accuracy: 44/91

The result is cleaner and more accurate than a median filter. However, the accuracy is lower than pure Canny algorithm (no filter)

## 5. Morphological transform: Erosion

### 5.1. Erosion

Erosion is applied to remove the pixels near boundary.

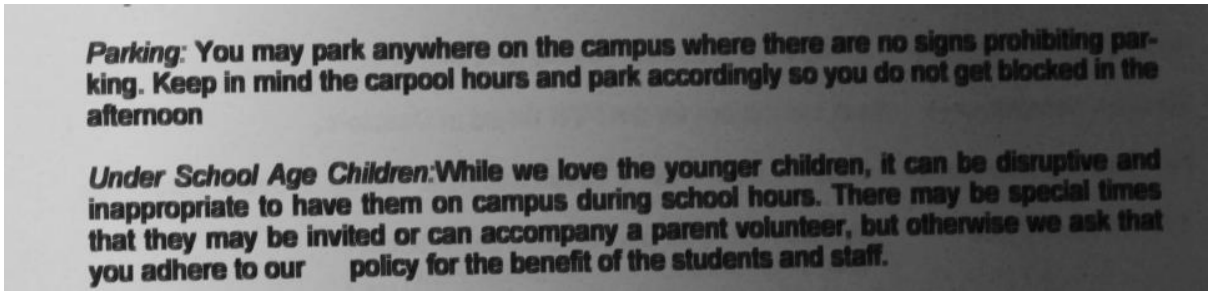



*Figure: Erosion with a neighbour pixel of 3*

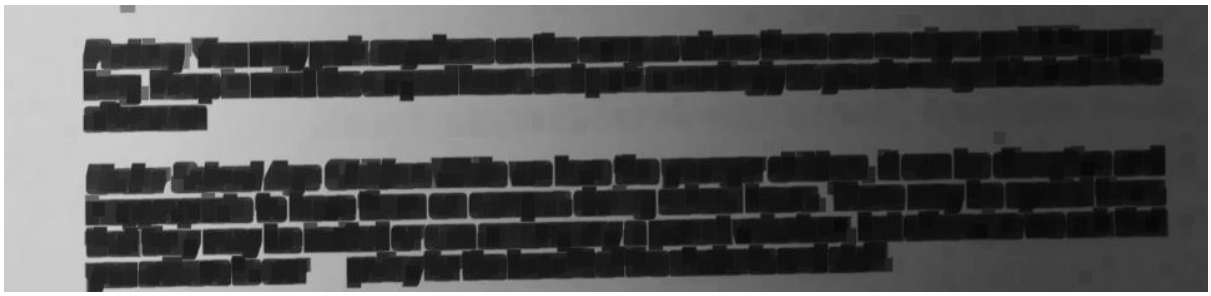


*Figure: Erosion with a neighbour pixel of 10*

While erosion is said to remove the boundaries, it darkened and strengthened the text as the kernel size increases.

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of
```

Accuracy: 46/91

There is no difference in the result when kernel size is equal to 2 or 10. The accuracy is slightly lower than Otsu algorithm.

## 5.2. Application of Otsu algorithm on erosion

The optimal threshold reminds the same

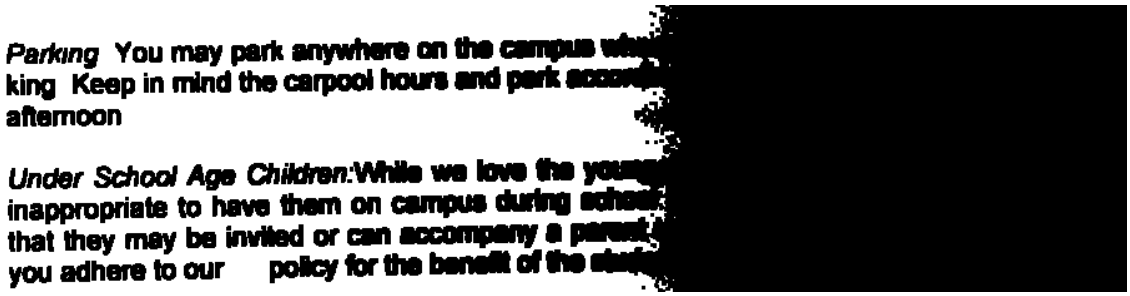


*Figure: Otsu Algorithm + Erosion*

Threshold remains the same. The text is thicker as compared to a pure Otsu algorithm

Upon applying Tesseract on the image, the following is received:

```
 Parlang You may park anywhere on the campus
king Keep in mind the carpool hours and perk seo
aftemoon

Under School Age Children:While wa love the yc
inappropriate to have them on campus during #0
that they may be invited or can accompany & P
you adhere to our
```

Accuracy: 40/91

There is more noise. The text accuracy is worse with Otsu applied, and lower than a pure Otsu algorithm(no erosion)

## 5.3. Application of Canny algorithm on Erosion

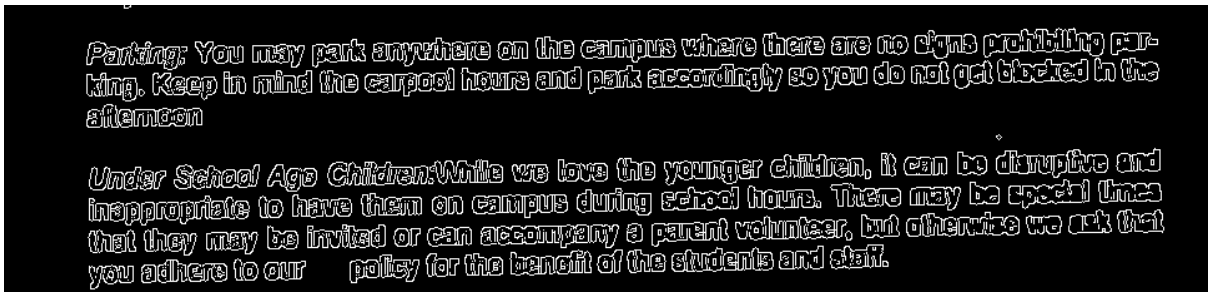



*Figure: Canny Algorithm of threshold value of 50-70 + Erosion*

The text contains a lot of noise.

Upon applying Tesseract on the image, the following is received:

Parking: You may park anywhere on the c
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love the |
inappropriate to have them on campus during s
that they may be invited or can accompany
you adhere to our _policy for the benefit of

Accuracy: 48/91

The accuracy is higher than erosion, and lower than a pure Canny algorithm. However, the text accuracy is closely similar to a median filter and a bilateral filter

# 6. Morphological transform: Opening

## 6.1. Application of Opening

Opening is meant to remove noise

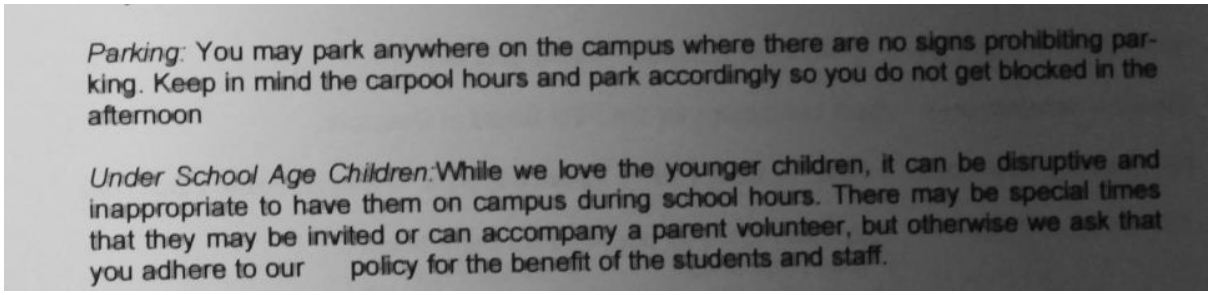



*Figure: Opening with a neighbour pixel of 3*

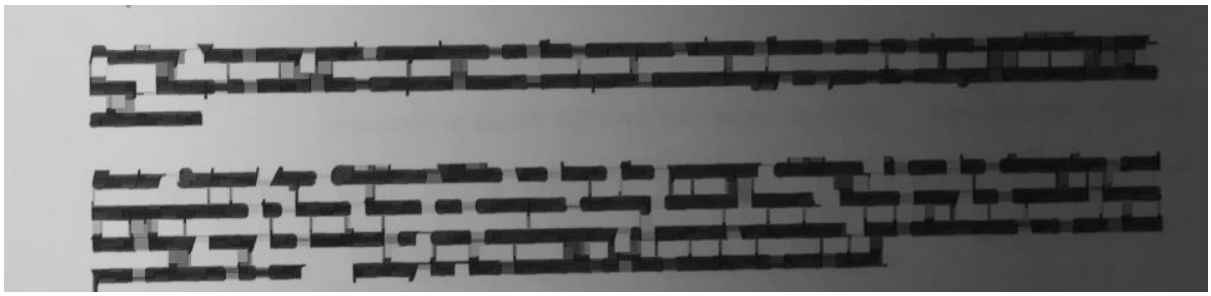


*Figure: Opening with a neighbour pixel of 10*

The text is thinner as compared to erosion when erosion is supposed to be thinner

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of
```

Accuracy: 47/91

Like erosion, there is no difference in text for a kernel of 3 and 10. The text returned is almost similar, with 1 word missing, as compared to erosion

## 6.2. Application of Otsu algorithm on opening

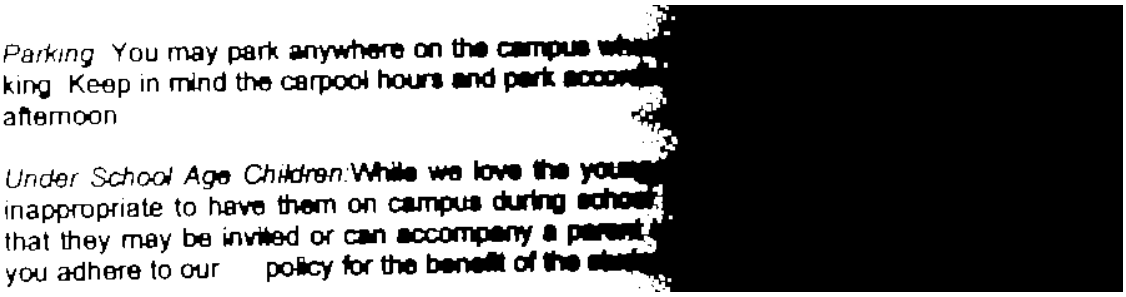


*Figure: Otsu Algorithm + Opening*

Threshold reminds the same.

Upon applying Tesseract on the image, the following is received:

```
 Parking You may park anywhere on the campus
king Keep in mind the carpool hours and park
aftamoon

Under School Age Chidren: While wa love the
inappropriate to have them on campus during
that they may be invited or can accompany &
you adhere to our policy for the benedit of the
♀
```

Accuracy: 46/91

The text accuracy is about the same, with 1 character lesser, than without Otsu algorithm. The accuracy is higher than Otsu after erosion, and lower than a pure Otsu algorithm

## 6.3. Application of Canny Algorithm on Opening

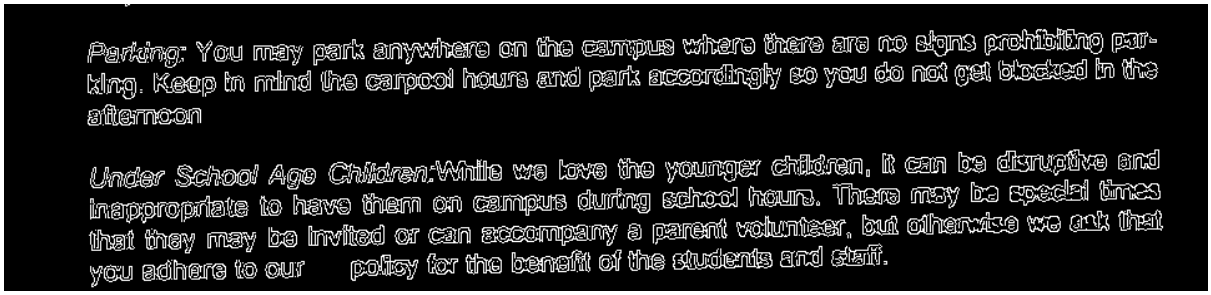



*Figure: Canny Algorithm + Opening*

A threshold of 50 70

Upon applying Tesseract on the image, the following is received:

```
Fey en Re ean oR Re Re Ok nko eR Aco aN eo
save) Reet clkecnsce beaten eee ent ino ic be oti kes oka
Eeilern 904]

eee Rene aM R or ccc Ue aOR eed
ee ee Ne eee rm oc a ge
a ee ete cA Nie ee aR
POURS Clic ROR Perm cick ect Tease
♀
```

Accuracy: 0/91

The text is not the least human readable, however, it has the lowest accuracy out of all the experiments at 0.

# 7. Adaptive Mean Thresholding

## 7.1. Application of Adaptive Mean Thresholding

Local adaptive thresholding is able to calculate the threshold for a small region of the image, making it suitable for areas with different lighting conditions. A median filtering is applied before the usage of adaptive mean thresholding

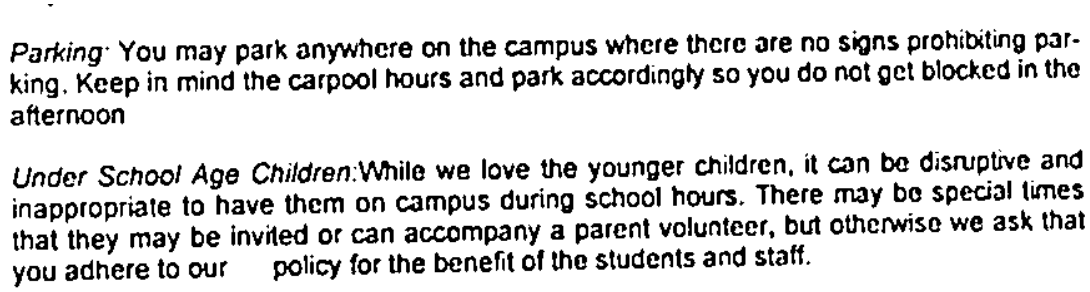



*Figure: Adaptive Mean Threshold with a mean neighbour size of 11 and binary value of 8*

While the text is distinct and strong, there exists thin lines on the right

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of


♀
```

Accuracy: 46/91

The accuracy is slightly worse than Otsu

## 7.2. Application of Closing on Adaptive Mean Thresholding

Closing is applied to close the details for the character profile, removing the noise inside the character

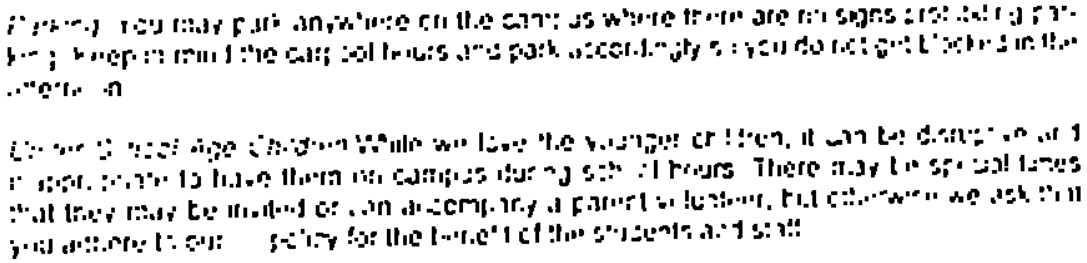


*Figure: Closing + Adaptive Mean Threshold + Median Filter*

The accuracy is not human readable, with a lot of separated details

Upon applying Tesseract on the image, the following is received:

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

Accuracy: 89/91

While the image is not human readable, the accuracy of the result is almost at 100%. The error consists of 3 characters.

1. The “.” replacing “,” ( not counted in accuracy)
2. The “!” replacing “l” in “School!”
3. The “l” replacing “t” in “limes”.

## 8. Adaptive Gaussian Thresholding

### 8.1. Application of Adaptive Gaussian Thresholding

Due to the success of adaptive mean thresholding, another adaptive thresholding, adaptive Gaussian thresholding, will be tested.

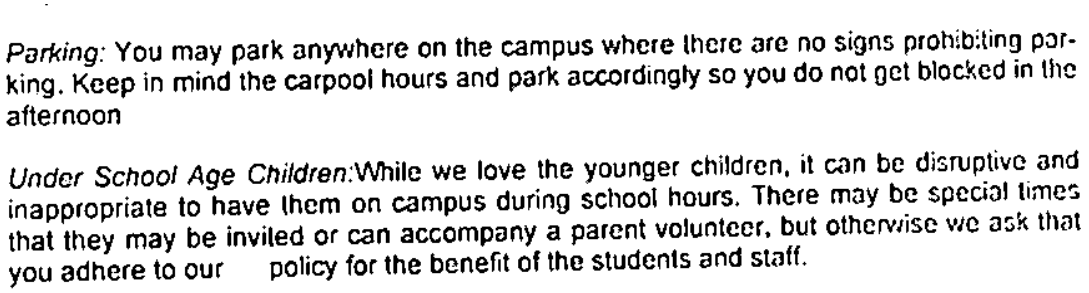



*Figure: Adaptive Gaussian Thresholding with a mean neighbour size of 11 and binary value of 8*

Both Mean Adaptive Thresholding and Adaptive Gaussian Thresholding are using the same number of kernel size(11) and binary value(8).

The result returned is similar to the result by Mean Adaptive Thresholding, but thinner

Upon applying Tesseract on the image, the following is received:

```
Parking: You may park anywhere on the ce
king. Keep in mind the carpool hours and park
afternoon

Under School Age Children:While we love
inappropriate to have them on campus @ )
that they may be invited or can accompany :
you adhere to our _ policy for the benefit of


♀
```

Accuracy: 46/91

The accuracy is the same as Mean Adaptive Thresholding.

### 8.2. Application of Closing on Adaptive Gaussian Thresholding

Closing is applied to close the details for the character profile, removing the noise inside the character.

*Figure: Closing + Adaptive Gaussian Threshold*

The resulting image is an almost white image. The black border is added for visibility and is not related to the result of the algorithms.

Upon applying Tesseract on the image, the following is received:

> 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 inviled or can accompany a parent volunteer, but otherwise we ask that
> you adhere to our —_ policy for the benefit of the students and staff.
> ♀

Accuracy: 89/91

The accuracy is the highest thus far. There are 2 characters different from the original image

1. The “!” replacing “l” from “School!”
2. The “t” replacing “l” from “inviled”

The accuracy score is the same as mean adaptive threshold despite having 1 less error. This is due to the fact that only alphabets/words are considered for accuracy.

# Evaluation of accuracy for sample01

**Otsu** will be bolded, Canny will be underlined, *morphological transformations* will be italic, filters will be in blue

| Rank | Method | Score out of 91 | Percentage correct |
|---|---|---|---|
| 1 | *Closing* after Adaptive Gaussian Thresholding after Median Filter | 89 | 98 |
| 2 | *Closing* after Adaptive Mean Thresholding after Median Filter | 89 | 98 |
| 3 | Inverted Canny Algorithm of threshold 40-50 | 77 | 85 |
| 4 | **Otsu** with threshold of 60 | 75 | 82 |
| 5 | **Otsu** with threshold of 70 | 68 | 75 |
| 6 | **Otsu** optimal | 53 | 58 |
| 7 | Canny Algorithm with threshold of 40-50 | 51 | 56 |
| 8 | Canny Algorithm after Bilateral Filter | 50 | 55 |
| | **Otsu** Algorithm after Bilateral Filter | 50 | 55 |
| 9 | Canny Algorithm after *Erosion* | 48 | 53 |
| 10 | *Opening* | 47 | 52 |
| 11 | Median Filter with kernel size 3 | 46 | 51 |
| | *Erosion* | 46 | 51 |
| | **Otsu** Algorithm after *Opening* | 46 | 51 |
| | Adaptive Mean Thresholding | 46 | 51 |
| | Adaptive Gaussian Thresholding | 46 | 51 |
| 12 | Bilateral Filter with kernel size 17 | 44 | 48 |
| | Canny Algorithm after Bilateral Filter | 44 | 48 |
| 13 | **Otsu** Algorithm after Median Filter | 43 | 47 |
| 14 | **Otsu** Algorithm after *Erosion* | 40 | 44 |
| 15 | Canny Algorithm after Median Filter | 34 | 37 |
| 16 | Canny Algorithm after *Opening* | 0 | 0 |

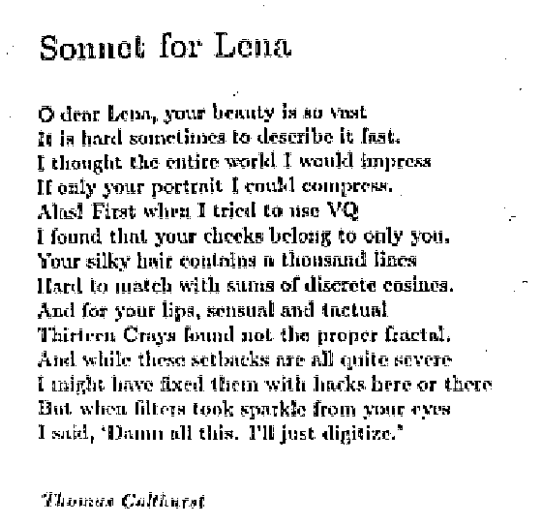
There is no definite trend

# Experiment: sample02

## 1. Adaptive Gaussian Threshold

### 1.1. Application of Closing after Adaptive Gaussian Threshold after Median Filter

Due to its high accuracy for “sample01”, this combination of steps will be applied to sample02



Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire workl 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, 'Damn all this. I'll just digitize.'

*Thomas Colthurst*

*Figure: Closing + Adaptive Gaussian Threshold + Median Filter*

The text thickness and strength are uneven and there are noise in the image

Upon applying Tesseract on the image, the following is received:

```
Sonnet for Lena

O dear Lena, your beauty is ao vast

itis hard sometimes to describe it frat.

T thought the entice worl T would dmpress
If only pour portrait [ coukl compress,

Adosd First when I tried to tise VO

I found thot your cheeks beloug to only you.
Your silky baie éontal
Lard to mate silma of discrete cosines.
And fer your ipa, sensual and dnectual
```

Thiriren Crys found not. the proper fractal,

fied awhile these setbacks are all qui &

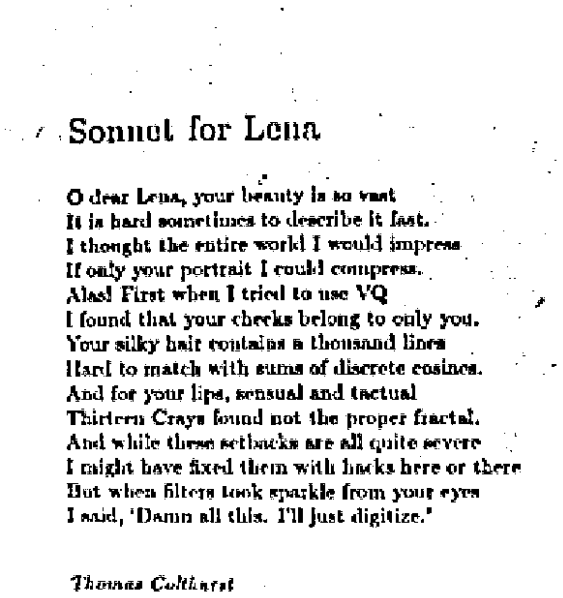
i them with hacks bere or there
14 took spatkic from your vyes
Tsail, Wamn all this, Vl just digitize.”

Accuracy: 64/116

## 2. Adaptive Mean Threshold

### 2.1. Application of Closing after Adaptive Mean Threshold after Median Filter

Due to its high accuracy for “sample01”, this combination of steps will be applied to “sample02”.



Sonnet for Lena

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

*Thomas Colthurst*

*Figure: Closing + Adaptive Mean Threshold + Median Filter*

The text thickness and strength are uneven and there are noise in the image. There is a greater degree of unevenness for Adaptive Mean Threshold than Adaptive Gaussian Threshold

Upon applying Tesseract on the image, the following is received:

```
- Sonnel for Lena

O dear Lena, your lenuty In bo vant

It is bard sometinaca to describe ft fast.
Ethonght the entire workd I would impress
Ifoaly your portrait [ould compress,”
Alas! First when T tricd ta use ¥Q

LE found that your checks belong te only you.
Your silky bait contaloa & thousand linea
Iland to match with eums of diserete cosinca.
Aad for your lips, sensual and tactual
Thirtern Crays found not the proper fractal,
Ane white theae setbacks are all quite severe
T caight bave fixed them with hocks bere or there
```

But when Altera Look eparkle from your eyes
Tard, ‘Damn all this. Ul Just digitize.”

Thowmes Coltinrat

Accuracy: 71/116

The accuracy is much higher with Adaptive Mean Threshold than Adaptive Gaussian Threshold

## 3. Contrast Adjustment

### 3.1. Application of contrast adjustment

sample02 has low contrast, as such, it has been increased by a factor of 1.2

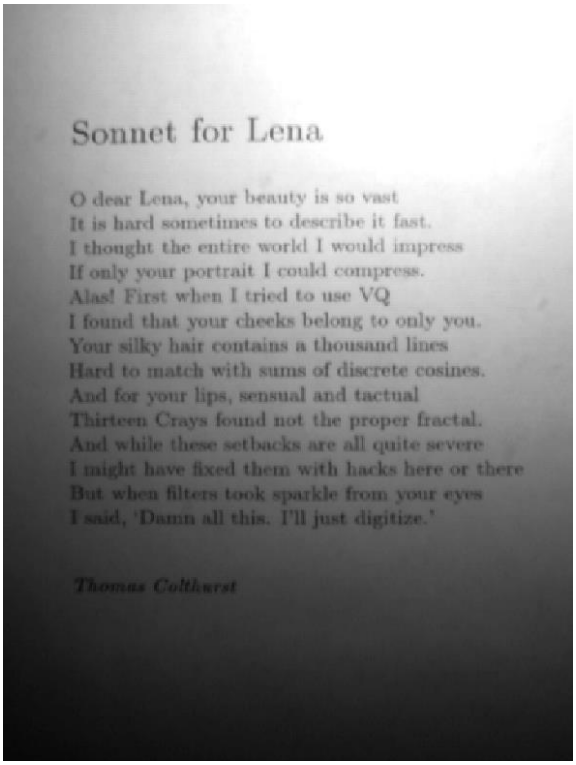
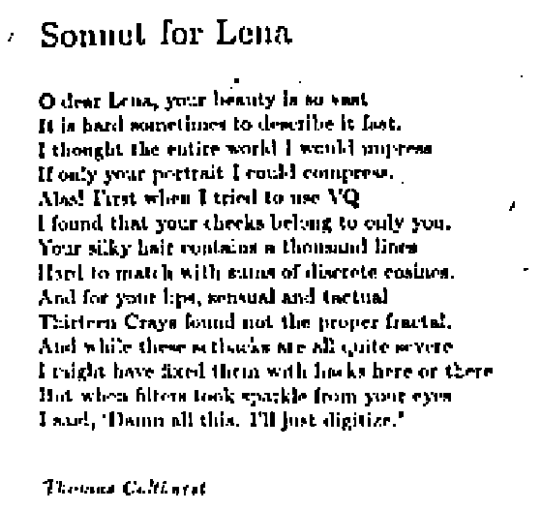



*Figure: Contrast Adjustment*

The text have greatly darkened.

Tesseract is unable to detect any text from this image

### 3.2. Application of Closing after Adaptive Mean Threshold after Median Filter



Sonnet for Lena

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

Thomas Caltharat

*Figure: Closing + Adaptive Mean Threshold + Median Filter + Contrast Adjustment*

Upon applying Tesseract on the image, the following is received:

```
? Sonnel for Lena

O dear Lena, your leenuty In bo vant

It in bard sometinoea to deacribe it fast.
Ethonght the entire world I would impreaa
Ifoaly your portrait [rould compress,
Alas! First when I tried to use ¥Q

L found that your checks belong to only you.
Your silky bait contains a thousand lines
Iland to mateh with sums of discrete cosinca.
Aad for your lipa, sensual and tactual
Thirtern Craya found not the proper fractal,
Anel while thea setbacks are all quite severe
T caight bave fixed them with hicks bere or there
Tut when Alters Look eparkle from your eyes
Taard, 'Damn all this. IU Just digitize."

Thomas Caltharat
```

Accuracy: 77/116

Increasing the contrast has indeed improved the accuracy

## Evaluation of accuracy for sample02

| Rank | Method | Score out of 116 | Percentage correct |
|---|---|---|---|
| 1 | Closing + Adaptive Mean Threshold + Median Filter + Contrast Adjustment | 77 | 66 |
| 2 | Closing + Adaptive Mean Threshold + Median Filter | 71 | 61 |
| 3 | Closing + Adaptive Gaussian Threshold + Median Filter | 64 | 55 |

## Improve recognition algorithms

Recognition software are moving towards RNN. Tesseract itself is currently running on LSTM, with dictionaries, word lists, patterns and specific segmentation method. While the accuracy in this paper is less than optimal, it is because Tesseract is not being fully utilized.

Recognition software will continue to improve with the increase in data available.

## Code: otsu_algorithm.py

Implementation of Otsu algorithm:

1. Pass in image
2. Calculate optimal threshold
3. Convert image
4. Save new image

```
import math
import numpy as np
import cv2
from PIL import Image
import matplotlib.pyplot as plt

# change here
image_name="sample01"

image = cv2.imread('%s.png'%(image_name),0)
save_file_name="%s_otsu.png"%(image_name)

file_path = "/Users/User/Desktop/CV"

width = 0
height = 0
u1_l = []
w1_l = []
v1_l = []

u2_l = []
w2_l = []
v2_l = []

c_index = []
aw_l = []
ab_l = []


histogram = cv2.calcHist([image],[0],None,[256],[0,256])
```

```
# alot of NaN values
np.seterr(divide='ignore', invalid='ignore')

# N = n1 + n2 + ... + nl
# Total number of pixels
def N_pixels():
  N = 0
  for i in range(len(histogram)):
    if (histogram[i]>0):
      N = N + histogram[i]

  return N


# probability at gray level i
def prob(i):
  N = N_pixels()
  p = histogram[i]/float(N)

  return p

# weight of class 1 : range of 0 to t
# weight of class 2: range of t to L-1
# gray level possibility distributions
def weight(a,b):
  w = 0
  for j in range(a,b):
    p = prob(j)
    w = w +p

  return w

# mean of class 1: range of a to b
def mean(a,b):
  w = weight(a,b)
  m = 0
  u = 0
  for j in range(a,b):
```

```
    p = prob(j)
    m = m + p

  u = m/float(w)

  return u


# gray level probability distributions
def gray_lvl(w1,u1,w2,u2):
  ut = w1 * u1 + w2 * u2

  return ut

# total mean of gray levels
def gray_lvl(w1,u1,w2,u2):
  ut = w1 * u1 + w2 * u2

  return ut

# class variance
def class_var(a,b):
  v = 0
  a = 0
  u = mean(a,b)
  w = weight(a,b)
  for j in range(a,b):
    p = prob(j)
    v = v + ((j - u)**2) * p

  a = v/float(w)

  return a


# within-class variance
def within_class_var(k,m):
  aw = 0
  for j in range(k,m):
```

```
    w = weight(j,m)
    a = class_var(j,m)
    aw = aw + w*a

  return aw

# between class var
def betw_class_var(w1,u1,w2,u2):
  ut = gray_lvl(w1,u1,w2,u2)
  if (math.isnan(ut)):
    ut = 0
  print(ut)
  ab = (w1 * (u1 - ut)**2) + (w2(u2-ut)**2)
  print(ab)
  return ab



def threshold(histogram):
  N = N_pixels()
  # go through every single possible threshold, 1 - 254

  for i in range(1,len(histogram)):
    v1 = class_var(0,i)
    w1 = weight(0,i)
    u1 = mean(0,i)

    v2 = class_var(i,len(histogram))
    w2 = weight(i,len(histogram))
    u2 = mean(i,len(histogram))

    # variance within class
    aw = w1 * v1 + w2 * v2

    # variance between class
    ab = w1 * w2 * (u1 - w2)**2

    if (not (math.isnan(v1) or math.isnan(u1) or math.isnan(w1) or math.is
nan(ab) or math.isnan(aw))):
```

```
        c_index.append(i)

        v1_l.append(v1)
        u1_l.append(u1)
        w1_l.append(w1)

        v2_l.append(v2)
        u2_l.append(u2)
        w2_l.append(w2)

        ab_l.append(ab)
        aw_l.append(ab)
        print(ab)

    return ab_l, aw_l


# Maximize between class variance or minimize within-class variance
def optimal_threshold():
    ab_list, aw_list = threshold(histogram)
    print("1")
    #maximize between class varance
    ab_list.sort()
    optimal_ab = ab_list[-1]
    print("between")
    print(optimal_ab)

    # minimize within class
    aw_list.sort()
    optimal_aw = aw_list[0]
    print("within")
    print(optimal_aw)

    f = open("otsu_data.txt",'w')
    f.write("index = " + str(c_index))
    f.write("v1 = " + str(v1_l) + '\n\n')
    f.write("u1 = " + str(u1_l) + '\n\n')
    f.write("w1 = " + str(w1_l) + '\n\n')
    f.write("v2 = " + str(v2_l) + '\n\n')
```

```
  f.write("u2 = " + str(u2_l) + '\n\n')
  f.write("w2 = " + str(w2_l) + '\n\n')
  f.write("ab = " + str(ab_l) + '\n\n')
  f.write("aw = " + str(aw_l) + '\n\n')
  f.write("max between class variance " + str(optimal_ab))
  f.write("min within class variance " + str(optimal_aw))
  f.close()

  return math.floor(optimal_ab*1000)

def reconstruct_otsu():
  threshold = optimal_threshold()

  # this process takes some time, so printing it out can show that its pro
gressing
  print(threshold)

  output_otsu = np.zeros((len(image), len(image[0])))
  for i in range(len(image)):
    for j in range(len(image[0])):
      if (image[i][j] > threshold):
        output_otsu[i][j] = 255
      else:
        output_otsu[i][j] = 0

  return output_otsu

def main():
  output = reconstruct_otsu()
  cv2.imwrite(save_file_name, output)


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

## Code: algorithm.py

Implementation of algorithms:

1. Pass in image
2. Choose specific algorithm to use
3. Convert image
4. Save image
5. Call Tesseract.py to get OCR of newly converted image

```
import numpy as np
import cv2
from scipy import ndimage
from PIL import Image, ImageEnhance

from matplotlib import pyplot as plt
import os
from tesseract import main

file_path = "C:/Users/User/Desktop/CV "

# change here
image_name="sample01"

img = cv2.imread('%s.png'%(image_name),0)

# Canny edge
def canny(lower,higher):
  edges = cv2.Canny(img,lower,higher)
  cv2.imwrite('%s/canny_%s_%s_%s.png'%(file_path,image_name,lower,higher),
edges)
  print("saved")

# Invert color
def invertcol():
  imgi = (255-img)
  cv2.imwrite('%s/inverted_%s.png'%(file_path,image_name),imgi)
  print("saved")

# Bilateral filtering
def bi_filter(d,sc,ss):
  bi_f = cv2.bilateralFilter(img,d,sc,ss)
```

```
  cv2.imwrite('%s/bilateral_%s_%s_%s_%s.png'%(file_path, image_name,d,sc,ss),bi_f)
  print("saved")

# median blurring
def medi_filter(kernel):
  median = cv2.medianBlur(img,kernel)
  cv2.imwrite('%s/median_%s_%s.png'%(file_path,image_name,kernel),median)
  print("saved")

# Increase contrast
def increase_contrast(factor):
  imag = Image.open('sample02.png')
  enhancer = ImageEnhance.Contrast(imag)
  imoutput = enhancer.enhance(factor)
  imoutput.save('%s/contrast_%s_%s.png'%(file_path,image_name,factor))
  print("saved")

# morphological transformation
def morph_closing(value):
  kernel = np.ones((value,value),np.uint8)
  closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE,kernel)
  cv2.imwrite('%s/morph_closing_%s_%s.png'%(file_path,image_name,value),closing)
  print("saved")


# morphological transformation
def morph_erosion(value):
  kernel = np.ones((value,value), np.uint8)
  erosion = cv2.erode(img,kernel,iterations=1)
  cv2.imwrite('%s/morph_erosion_%s_%s.png'%(file_path,image_name,value),erosion)
  print("saved")

# morphological transformation
def morph_opening(value):
  kernel = np.ones((value,value), np.uint8)
```

```
  opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
  cv2.imwrite('%s/morph_opening_%s_%s.png'%(file_path,image_name,value),op
ening)
  print("saved")

# morphological transformation
def morph_dilation(value):
  kernel = np.ones((value,value), np.uint8)
  dilation = cv2.dilate(img,kernel, iterations=1)
  cv2.imwrite('%s/morph_dilation_%s_%s.png'%(file_path,image_name,value),d
ilation)
  print("saved")

# Adaptive Mean thresholding
def adapt_mean():
  image = cv2.medianBlur(img,1)
  th2 = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_MEAN_C, cv2.TH
RESH_BINARY,13,5)
  cv2.imwrite('%s/adaptm_%s.png'%(file_path,image_name),th2)
  print("saved")

# Adaptive Gaussian Thresholding
def adapt_gauss():
  image = cv2.medianBlur(img,1)
  th3 = cv2.adaptiveThreshold(image,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2
.THRESH_BINARY,13,4)
  cv2.imwrite('%s/adaptgauss_%s.png'%(file_path,image_name),th3)
  print("saved")


def histo_equal():
  dst = cv2.equalizeHist(img)
  cv2.imwrite('%s/histo_e_%s.png'%(file_path,image_name),dst)
  print("saved")


def get_text():
  #bi_filter(17,10,15)
  canny(50,70)
```

```
    #medi_filter(3)
    #increase_contrast(1.2)
    #morph_opening(10)
    #morph_erosion(2)
    #morph_closing(2)
    #morph_dilation(2)
    #invertcol()
    #adapt_mean()
    #adapt_gauss()
    #histo_equal()

    main(image_name)



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

## Code: tesseract.py

Trigger OCR and plot histogram

```
from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt
import pytesseract


# image_name = "sample02"

file_path = "C:/Users/User/Desktop/CV/sample02"

# Tesseract requirememt
pytesseract.pytesseract.tesseract_cmd = r'C:/Users/User/AppData/Local/Tess
eract-OCR/tesseract.exe'


# def plot_histogram():
#    img = cv2.imread(image_name+".png" , 0)
#    # hist=cv2.calcHist([img],[0],None,[256],[0,256])
#    plt.hist(img.ravel(),256,[0,256]);
#    plt.savefig("histogram.png")


# return text from OCR
def OCR(filename):
  text = pytesseract.image_to_string(Image.open(filename))
  print(text)
  return text


def base(name):
  text = OCR(name)
  f = open('text.txt','a')
  f.write("%s text: "%(name) + str(text) + '\n\n\n')
  f.close()
```

```
def main(image_name):
  base(image_name+".png")


if __name__ == '__main__':
  #plot_histogram()
  main(image_name)
```