

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1. Implementation

This project use Python programming language. This project will use an image filter(grayscale, erosion, dilation) which will then be passed to tesseract to detect and convert the image into text , will this filter make the image easier to detect or even make it harder to detect. The size of the sample image used was 750x500 pixel. To take an image from the directory and to create a txt file using the opencv library and for the text conversion process using the tesseract library.

```
1. for i in range(h):
2.     for j in range(w):
3.         p = float(img[i][j][0])*0.2989 + float(img[i][j][1])*0.5870 +
           float(img[i][j][2])*0.1140
4. newImage[i][j] = p
```

first we will make the image grayscale by taking the RGB value in each array and multiplying it by a certain number and we will replace the result in the array with a new value (in line 3 and 4).

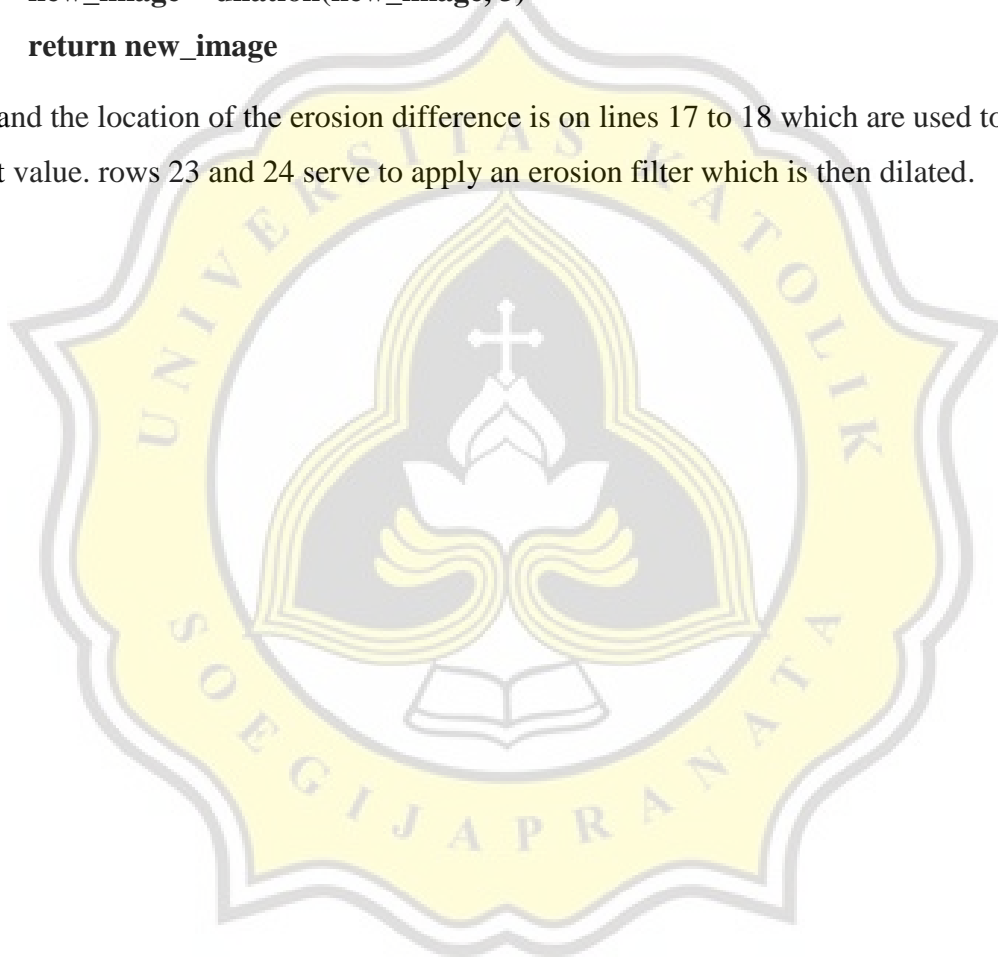
```
5. def calculate_template_space(template_side_length):
6.     return int(template_side_length/2)
7.
8. def dilation(img, template_side_length):
9.     template_space = calculate_template_space(template_side_length)
10.
11.     for x in range(template_space, new_image.shape[1] - template_space):
12.         for y in range(template_space, new_image.shape[0] - template_space):
13.             for c in range(0, template_side_length):
14.                 for d in range(0, template_side_length):
15.                     if sub > maximum:
16.                         if sub > 0:
17.                             maximum = sub
```

In line 5 and 6 useful for making brushes or which will be used as templates which will be carried out per template erosion and dilation itself. Line 8 and 9 to to declare a dilation

function which has image parameters and brush template, in line 11 and 12 to do repetition where the brush template starts running to the end of the image, while lines 13 and 14 repetition in the brush template. In line 15 until 16 to find the largest value in the brush template.

```
18.     if sub < minimum:
19.         if sub > 0:
20.             minimum = sub
21.     def optimize(image):
22.         new_image = erosion(image, 3)
23.         new_image = dilation(new_image, 3)
24.         return new_image
```

and the location of the erosion difference is on lines 17 to 18 which are used to find the smallest value. rows 23 and 24 serve to apply an erosion filter which is then dilated.



5.2. Testing

I have tested my program with a total of 100 datasets and different types of datasets, the first is an image that I took from Google, a handwritten photo, a photo that does not have text elements and a photo that has a text element. because it detects the text I use the opening process as well because if you use closing the object will actually overlap so it will be difficult to convert later. I tested this dataset 5 times, based on the type of dataset and finally a mix of 100 datasets.

Of all the data that has been processed I do an analysis to measure how accurate and the comparison is from only tesseract and contains pre-processing, with confusion matrix I calculate Accuracy, Precision, Recall . where is Accuracy itself is the ratio of True (positive and negative) predictions to the overall data. and precision is the ratio of positive correct predictions compared to the overall positive predicted results. while recall is the ratio of true positive predictions compared to the overall data that are true positive. because I use the image text dataset, text image photo data,handwritten image data, the data is not image text so there are 4 possibilities that happen

- True Positive (TP) successfully detected and correct
- True Negative (TN) does not detect text in images that have no text
- False Positive (FP) successfully detected and false
- False Negative (FN) was not detected despite text

Of the 100 data I tested 5 times, the first was based on images, the second was handwritten, the third was an image that didn't have an image, the fourth was a photo that had text elements and the last one was a combination of all datasets.

data image text	pre processing	ocr
True Positive (TP)	26	20
True Negative (TN)	0	0
False Positve (FP)	10	18
False Negatif (FN)	4	2

Illustration 5.1 : table result of dataset image text

handwritten text	pre processing	ocr
True Positive (TP)	4	2
True Negative (TN)	0	0
False Positive (FP)	2	4
False Negatif (FN)	4	4

Illustration 5.2 : table result of dataset handwritten text

not image text	pre processing	ocr
True Positive (TP)	0	0
True Negative (TN)	10	10
False Positive (FP)	0	0
False Negatif (FN)	0	0

Illustration 5.3 : table result of dataset notimage text

photo image	pre processing	ocr
True Positive (TP)	19	11
True Negative (TN)		
False Positive (FP)	12	13
False Negatif (FN)	9	16

Illustration 5.4 : table result of dataset photo image

mix	pre processing	ocr
True Positive (TP)	49	33
True Negative (TN)	10	10
False Positive (FP)	24	35
False Negatif (FN)	17	22

Illustration 5.5 : table result of dataset mix image

to calculate Accuracy, Precision, Recall using the following formula:

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

$$\text{Precision} = (TP) / (TP + FP)$$

$$\text{Recall} = (TP) / (TP + FN)$$

here I'm just trying to calculate Accuracy, Precision, Recall data mix using pre processing

$$\text{Accuracy} = (TP + TN) / (TP + FP + FN + TN)$$

$$= (49 + 10) / (49 + 10 + 24 + 17)$$

$$= 59 / 100 = 0,59 = 59\%$$

$$\text{Precision} = (\text{TP}) / (\text{TP} + \text{FP})$$

$$= 49 / (49 + 24)$$

$$= 49 / 73 = 0,67 = 67\%$$

$$\text{Recall} = (\text{TP}) / (\text{TP} + \text{FN})$$

$$= 49 / (49 + 17)$$

$$= 49 / 66 = 0,74 = 74\%$$

by using the formula as above I get data like this

Pre processing	precision	recall	Accuracy
40 data image text	72%	86%	65%
40 data photo image text	61%	67%	47%
10 data handwritten image	66%	50%	40%
10 data not image text	0	0	100%
mix	67%	74%	59%

Illustration 5.6 : measures of pre processing

Tesseract	precision	recall	Accuracy
40 data image text	52%	90%	50%
40 data photo image text	45%	40%	27%
10 data handwritten image	33%	33%	20%
10 data not image text	0	0	100%
mix	48%	60%	43%

Illustration 5.7 : measures of tesseract

From the table above, 3 questions can be raised:

First, how many percent of the images were successfully predicted which had images and did not have images correctly? 59% vs 43%

The second question is what percentage of images that were successfully converted from all detected images? 67% vs 48%

the third question is what percentage of the image that was successfully converted from the total image that should have been detected? 74% vs 60%

From the question it can be concluded that those who use Tesseract only have less than optimal results, while those who go through the pre-processing process are more optimal.

I have tested the dataset with different preprocessing and this is the result

Dataset	erosion	dilation	closing
True Positive (TP)	33	24	36
True Negative (TN)	10	10	10
False Positve (FP)	35	40	32
False Negatif (FN)	20	26	22

And of all the preprocessing that I use, why don't I use erosion, dilation, or closure and instead use opening because from the results I tested using several different preprocessing values, the precision recall and the highest accuracy were in opening, so I used opening. Here I show the comparison results of opening, closing, erosion and dilation.

Pre processing	precision	recall	Accuracy
Erosion	48%	62%	43%
Dilation	40%	48%	34%
Closing	52%	62%	46%
Opening	67%	74%	59%

