CHAPTER 4
ANALYSIS AND DESIGN

4.1. Analysis

Process of converting images into text using the tesseract ocr library. But before processing the image, pre processing is carried out in order to improve image quality and increase the percentage of successful text conversion.The sample image to be processed must have good image quality so that the level of accuracy is higher. therefore we added a filter to solve this problem.The filters I use are grayscale, erosion and dilation depending on the image of the text, whether the text looks thick or looks too thin. so that the text is easier to detect which was thin and difficult to read, tesseract becomes easier to read This is also to prove my project whether the filter on the image affects the tesseract ocr making it easier or more difficult for tesseract to read the text.

4.2. Design

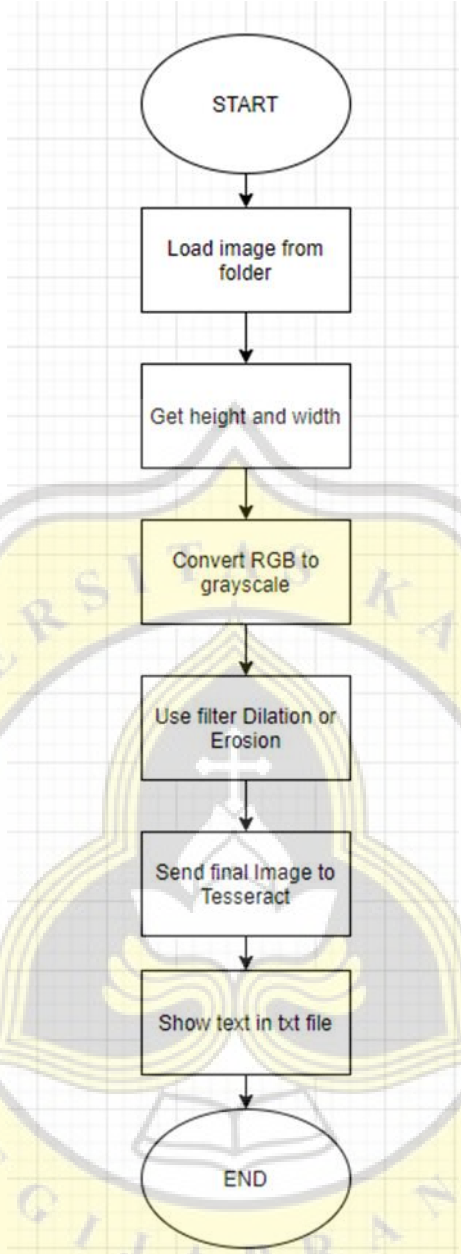

Illustration 4.1 : Flowchart

First we need to import the image that we will process then we show the image so we can see the image that we will process and we need to get the height and width of the image.

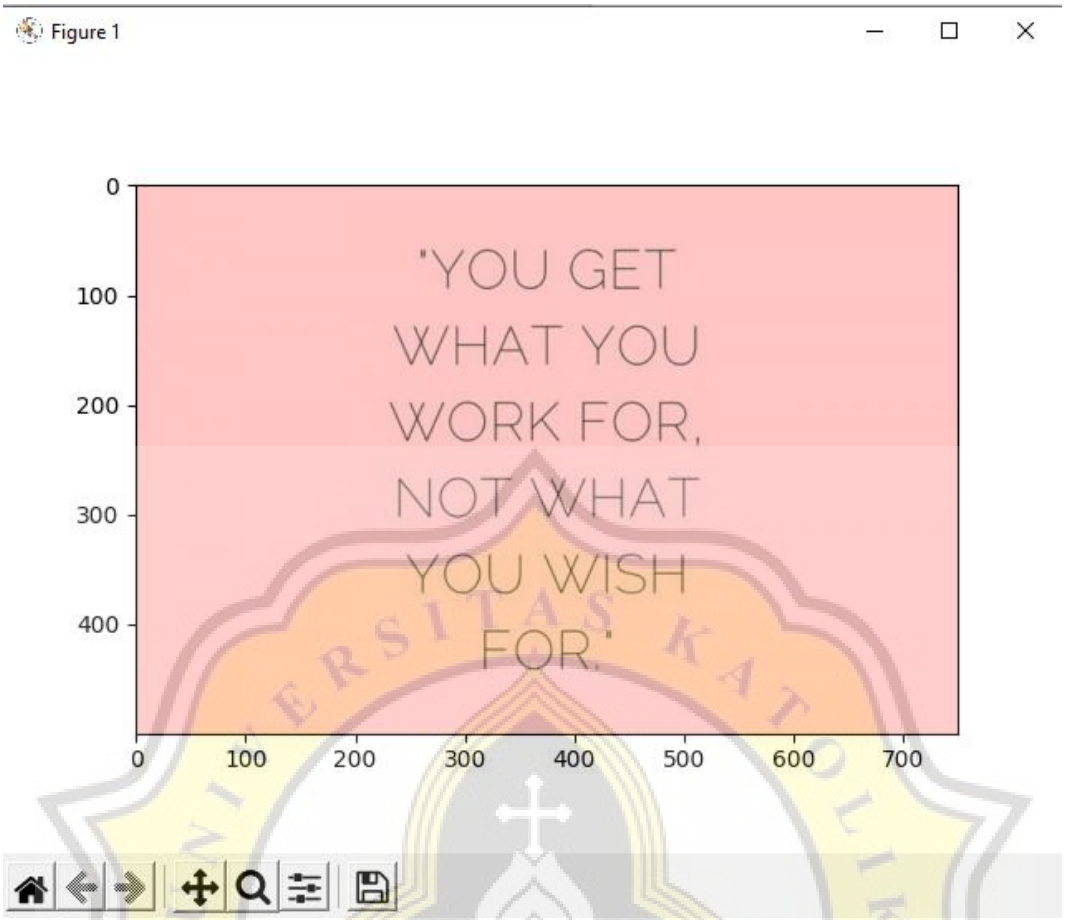

Illustration 4.2 : original image

To facilitate the work, Work in the process further in image processing, in the first stage Grayscaling process is carried out. This process changes the image with various colors into an 8-bit image gray. The equations used are

$$Gray = 0{,}2989*R + 0{,}5870*G + 0{,}1140*B$$

Information :

R : The intensity value of the red color component

G : Green component intensity value intensitas

B : The intensity value of the blue component

This process is done by doing each pixel a color that has three color components (RGB).With the above equation, we get a color new which has a color component with intensity between 0 to 225 (this is because image is 8 bits so there are 28 colors or 225). and for the grayscale

process, it will start with repetitions as much as height and width and then we take the RGB color in the loop index. and in each pixel there is an index to 0 which is red index 1 is green and index 2 is blue. then to convert to grayscale we multiply by the formula above.the first step we have to get the RGB value per pixel I take the example of 10x10 pixels between 260.60 to 269.69 from the picture above, more precisely in the quotation marks area, to get the RGB value of each pixel I use the library pillow.

152,106,106	147,103,102	199,155,154	97,56,54	217,173,170	242,198,195	241,197,194	242,197,194	247,199,197	255,209,207	60
143,99,98	137,96,94	201,160,158	84,43,41	214,173,169	241,200,196	244,200,197	248,204,201	247,202,199	246,198,196	61
145,101,100	140,99,97	195,154,152	88,48,46	212,173,168	240,201,196	243,202,198	243,199,196	244,199,194	236,191,186	62
151,107,106	145,104,100	186,145,143	96,55,51	91,48,42	117,74,68	201,156,151	248,203,198	248,200,196	252,203,199	63
144,100,97	140,97,91	200,256,253	91,48,42	213,170,164	248,205,199	248,203,198	244,196,192	247,198,194	246,195,192	64
167,122,117	157,114,108	198,155,149	117,74,68	221,176,171	247,202,197	246,198,194	247,198,194	252,201,198	249,195,193	65
220,175,170	216,171,166	238,193,188	201,156,151	235,190,185	248,203,198	245,197,193	250,201,197	254,203,200	247,193,191	66
251,203,199	245,200,195	254,206,202	248,203,198	245,197,193	250,202,198	245,196,192	250,199,196	254,200,198	247,193,191	67
249,200,196	242,194,190	242,193,189	248,200,196	245,196,192	248,199,195	244,195,191	246,195,192	251,197,195	251,197,195	68
249,198,195	246,197,193	247,198,194	252,203,199	246,197,193	248,199,195	248,197,194	246,195,192	249,195,193	252,198,196	69
260	261	262	263	264	265	266	267	268	269	

Illustration 4.3 : value from 260x60 to 269x69 . pixels

After we get the pixel value, we enter it into the formula above the first example at pixel 260.60 the RGB value is 152,106,106 if it is entered into the formula then:

Gray = 0,2989*152 + 0,5870*106 + 0,1140*106

45,43+62,22+12,08=119,73

If you already get the grayscale value of the pixel which originally had 3 values, now it only has 1 value, so the new 260.60 pixel is 119.73 and repeated as many times as width x height. and the result will look like this :

119,73	116,01	144,09	67,96	185,79	210,78	209,77	210,07	213,08	222,48	60
112,02	108	172	55,01	184,77	211,77	212,78	216,66	215,07	212,08	61
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22	62
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16	63
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88	64
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89	65
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89	66
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89	67
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89	68
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89	69
260	261	262	263	264	265	266	267	268	269	

Illustration 4.4 : value of the grayscaled image

After the above process is done, all pixels of the image will turn into a grayscale image, the image will be shown and saved in the directory and the image that will be displayed will be like this

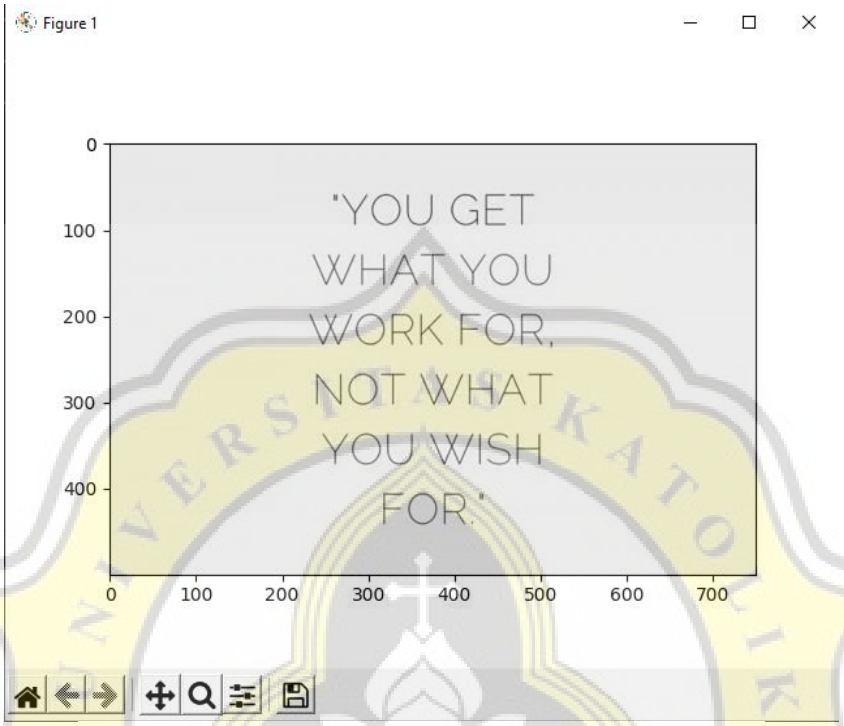

Illustration 4.5 : grayscale image

So that the smaller the value of the grayscale it will show the color towards the darker and vice versa the greater the value indicates the direction of the light color

After getting grayscale, the next step is dilation and erosion, the use of this function depends on the image text. Dilation itself is the process of adding pixels to the boundaries of objects in the image, while erosion removes pixels at the boundaries of objects. The number of pixels added or removed from objects in an image depends on the size and shape of the styling elements used to process the image, if the text in the image is too thin we use erosion and if it is too thick we use dilation it looks reversed because we are using a grayscale image so the erosion is should scrape instead looks like thicken and otherwise. the first thing we have to do is we have to define a template space, this template space itself is like a brush, it is recommended to use odd values to have a midpoint, so that from around the midpoint we calculate. after we

determine the brush, we divide the brush value by 2 and round it down which is useful to find out where the starting position is.

For example, we have an10x10px image and we set the brush to be 3, so 3 divided by 2 is 1.5 and rounded to 1 so we will start counting after the first pixel. so we start with the yellow square as the midpoint

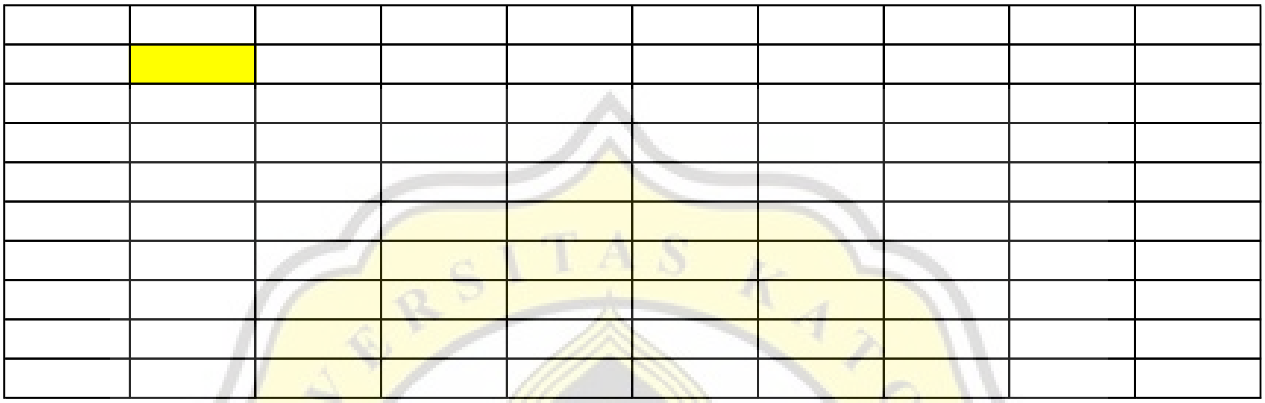

Illustration 4.6 : example location of yellow square

and also we start from the width first so that the process goes to the right, not to the bottom. For the repetition itself, the template width will be reduced from the template space itself so it doesn't stop at the wide end but stops at the red square because the template space is 1. and continue again from the yellow bottom box to the red bottom box again until it's finished

The first iteration is to define the path of the brush. and in that loop we repeat as many times as the brush, and above we set the brush to be 3.

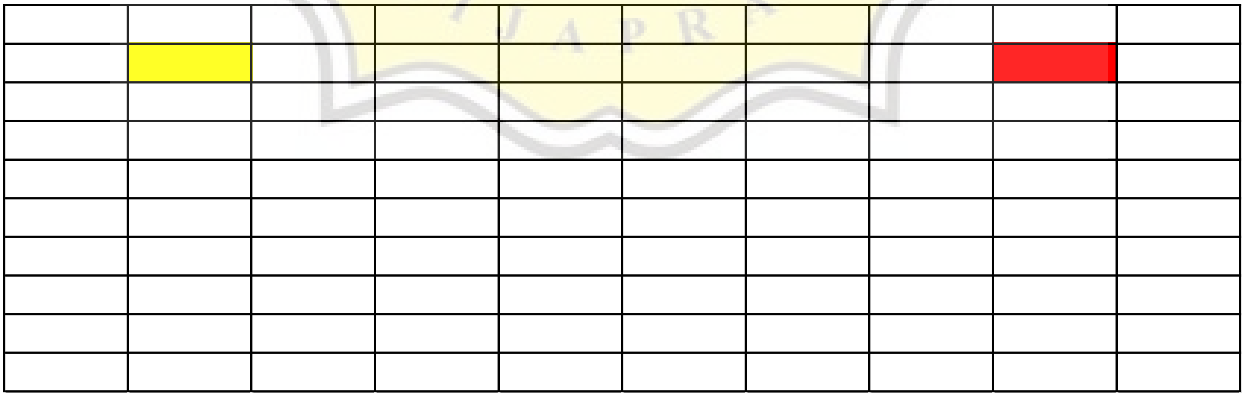

Illustration 4.7 : example locationof red square

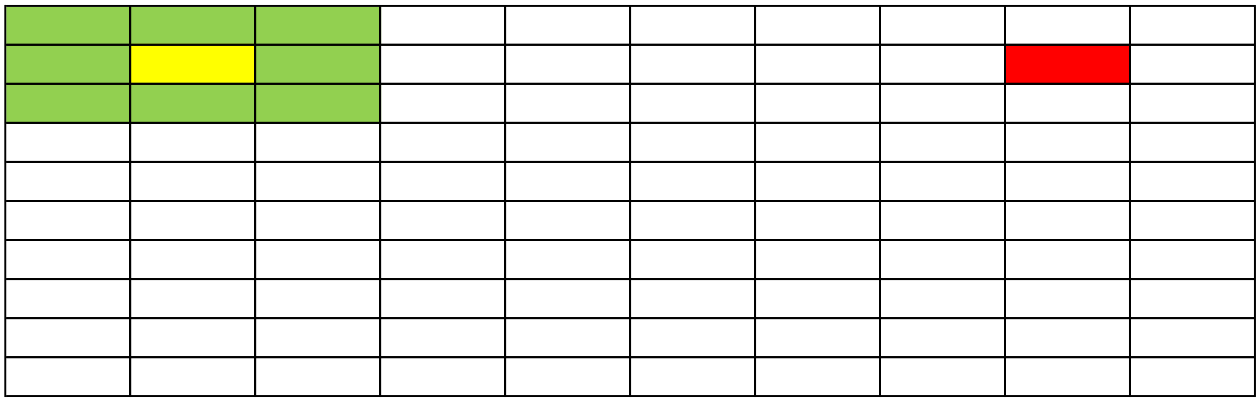

Illustration 4.8 : example location of green box

so it will look like this, if the midpoint is yellow then there will be 3x3 repetitions and yellow is used as the midpoint of the repetition. so if it is applied to the calculation above it will be like this:

119,73	116,01	144,09	67,96	185,79	210,78	209,776	210,07	213,08	222,48
112,02	108	172	55,01	184,77	211,77	212,78	216,66	215,07	212,08
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89

Illustration 4.9 : image pixel value 108is midpoint

its midpoint is yellow. what we do here is erosion so we will find the smallest value of the green box, smaller than 256 and greater than 0. so if we calculate it will be like this the first is 119.73 greater than 0 and less than 256 then the minimum value now is 119.73, next is 116.01 greater than 0 and less than 119.73 then the minimum value now is 116, 01. then 144.09 is greater than 0 but not less than the current minimum value, then the minimum value is still 116.01. and 112.02 is greater than 0 and is also smaller than the minimum value, so the current minimum value is 112.02. 108 is less than the current minimum and greater than 0 then 108 is the minimum value. then 172, 114.02, 110.99 and 165.99 none is smaller than the minimum value. then at midpoint still use the current minimum value of 108. If we have got the minimum value in 1 brush template, we move the midpoint so that it will look like the image below, the

midpoint now has a value of 172 and look for the minimum value again in the of the previous brush template which is 3.

119,73	116,01	144,09	67,96	185,79	210,78	209,776	210,07	213,08	222,48
112,02	108	172	55,01	184,77	211,77	212,78	216,66	215,07	212,08
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89

Illustration 4.10 : image pixel value 172 as midpoint

We start from the first 116.01 is greater than 0 and less than 256 then the minimum value now is 116.01 , 114.09 is not less than 116.01 then the minimum value is not replaced. further 67.96 is smaller than 114.09 then the minimum value now is 67.96 then 106, and 172 is not less than 67.96 then the minimum value is now 67.96 then 55.01 is smaller than the current minimum value and greater than 0 then the current minimum value is 55.01 and then 110.99 ,

165.99 , 59.71 then the minimum value remains 55.01 and the midpoint is replaced with the minimum value i.e. 55.01then it will look like this :

119,73	116,01	144,09	67,96	185,79	210,78	209,776	210,07	213,08	222,48
112,02	108	55,01	55,01	184,77	211,77	212,78	216,66	215,07	212,08
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89

Illustration 4.11 : image pixel minimum value

and so on until finished. when it's done it will look like this:

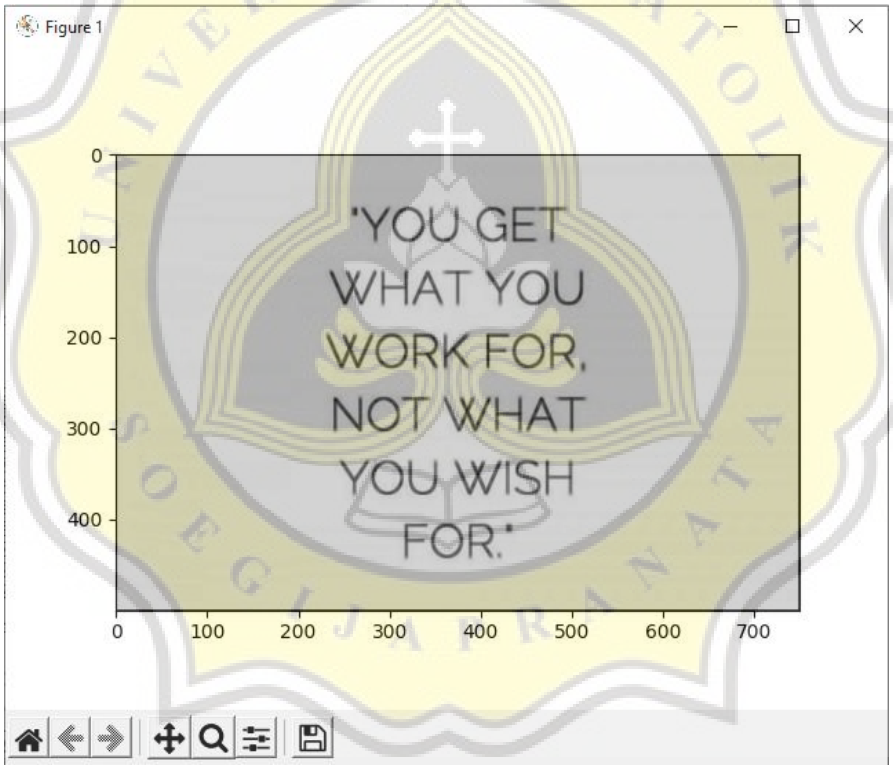

Illustration 4.12 : filtered image

The next filter is dilation. dilation itself is the opposite of erosion where if erosion is looking for a minimum value, dilation is looking for a maximum value, then it will look like this:

119,73	116,01	144,09	67,96	185,79	210,78	209,776	210,07	213,08	222,48
112,02	108	172	55,01	184,77	211,77	212,78	216,66	215,07	212,08
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89

Illustration 4.13 : image pixel value 108 as midpoint

in dilation we look for the largest value that must be greater than the value of 0 the first is 119.73 greater than the value 0 then the maximum value now is 119.73 then 116.01 is not greater than the maximum value then the maximum value is fixed, no changed. next is 144.09, 144.09 is greater than the current maximum value, then the current maximum value is shifted by 144.09 . then 112.02 and 108 are not greater than the maximum value, then the maximum value remains. the next value is 172 , and 172 is greater than the current maximum value, which is 144.09, then the current maximum value is 172. the next value is 114.02 , 110.99 , 165.99 and none of that is greater than the value maximum then the current maximum value is 172.

and then the value in the midpoint which was previously 108 is replaced with the maximum value where the maximum value is 172. and it will look like this:

119,73	116,01	144,09	67,96	185,79	210,78	209,776	210,07	213,08	222,48
112,02	172	172	55,01	184,77	211,77	212,78	216,66	215,07	212,08
114,02	110,99	165,99	59,71	184,06	212,05	213,17	211,78	211,85	214,22
90,01	106,81	157	66,78	59,96	86,15	168,85	215,85	213,86	217,16
112.79	109.15	238.89	60.16	182.15	217.14	215.86	209.87	212.17	209.88
134.86	126.15	167.15	86.16	188.86	214.86	211.87	212.17	215.88	210.89
187.86	183.86	205.86	168.86	202.86	215.86	210.87	215.17	217.88	208.89
216.87	212.86	219.87	215.86	210.87	215.87	210.17	213.88	215.89	208.89
214.17	207.87	207.17	213.87	210.17	212.71	209.17	209.88	212.89	212.89
212.88	211.17	212.17	217.16	211.17	213.17	211.88	209.88	210.89	213.89

Illustration 4.14 : image pixel maximum value

and the same thing will be done until everything in the brush template gets the maximum value and the midpoint is replaced with that maximum value.

but based on the opinion that I received, it is better not to bother determining this image is given an erosion or dilation filter, so I decided to use the opening process. Opening is an erosion process followed by dilation. Opening is usually used to remove small objects to make the edges of the image smoother. Different from closing which runs by dilating the image first, after that the image that has been dilated will be eroded. I don't use dilation because my dataset is text so that if it is dilated, it will actually make the text overlap and make it more difficult to convert/detect. therefore I use the opening process, so the final process of my image will be grayscaled and then I will erosion after that the image that has been eroded I am dilated after that I just give it to tesseract for detection and conversion.