CHAPTER 4
ANALYSIS AND DESIGN

4.1 Analysis

This project was created to identify the types of geometry shapes in the 2D image. This 2-D image is processed using the Java programming language and methods that exist in image processing. The methods that used in this project is the thinning method and the edge detection method.

This thinning method is used to reduce binary image. This is intended to simplify the processing of images. The process of this thinning method is to convert colorize images to monochrome images. Before turning a colorize image into a monochrome image, the colorize image must be converted to a grayscale image. After that, the grayscale image is converted into a monochrome image.

The first thing that needs to be done before turning a color image into a monochrome image is to get the value red, green, blue from each pixel. And after get that values, the calculation that must be done to convert a colored image to a grayscale image is:

$$RGB\ grayscale = (red + green + blue)/3$$

When get the RGB values of grayscale image, RGB values must be calculated again and converted to a monochrome image. The calculation to make a grayscale image to a monochrome image is :

$$If:\ 0 \leq RGB\ grayscale\ values \leq 128$$

$$then:\ RGB\ monochrome = 0$$

$$If:\ 129 \leq RGB\ grayscale\ values \leq 255$$

$$then:\ RGB\ monochrome = 255$$

When the RGB value is 0, the color of image pixel is black. While the RGB value is 255, the color of image pixel is white. But, there is one thing that must be important. When the colorize picture is converting to grayscale picture or monochrome picture, the value of RGB must be same. Here the results of some colors converting:

Table 4.1: RGB values of images (from green to white)

Description	Red	Green	Blue
Before Processed	191	228	202
Grayscale	207	207	207
Monochrome	255	255	255

Then the RGB value of each pixel that has been converted to black and white must be stored in one variable. This variable can be used to process images and to find the edges of objects. The logic that used to find the first edge of the object is when the process of detecting a pixel found 1 pixel that it's color is black, then the position of that edge (x and y) must be saved in the new variable (example: variable x^A and y^A).

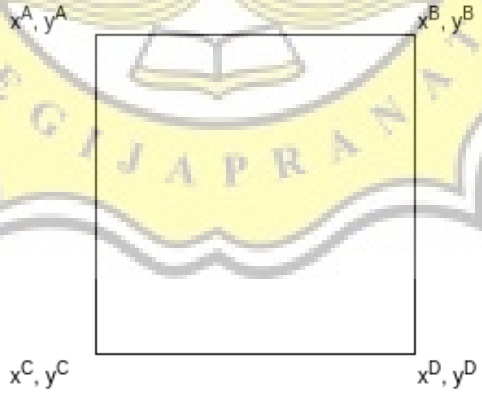

Illustration 4.1: Square with edge's name

The value of x^A and y^A can be used to get other edges. And the important thing that must be remember in pixel reading is the program reads pixel from left to right then from top to bottom. So, the second step after get the value of x^A and y^A is to find the value of x^B and y^B. Because x^B and y^B located on the right side of

x^A and y^A, it will be easier to find x^B and y^B (because it is located on horizontal side).

After get the value of x^A, y^A, x^B, and y^B, the thing that needs to be done is to detect the next edges (C and D). When the coordinates of this edges are found, it is important thing to check the edges have the same or different x value from the edges that has been found before. If the x value are same, it can be detected that the edges which connect edges x^A and x^C, or x^B and x^D are making straight vertical lines. But, if the x coordinate is different, it can be detected that the connected edges is a slash line.

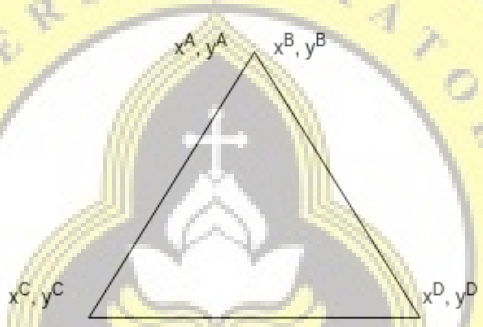

Illustration 4.2: Triangle with edge's name

If 4 points have been found, the next step is to determine the type of geometries. The basic rule if the geometries is a triangle, is the value of x^A must be equal to x^B and the value of y^A must also be equal to y^B. Followed by x^C values must be different from x^D, but has the same value of y^C and y^D. If want to detect a right triangle, the values of xA and xC or the values of xB and xD must be the same. If the triangle is upside down, the limitation that has been made between points A and B with points C and D must be exchanged. In some cases, if there is a difference between xB and xA (the difference is 1 to 2), automatically, the image is not detected as a triangle.

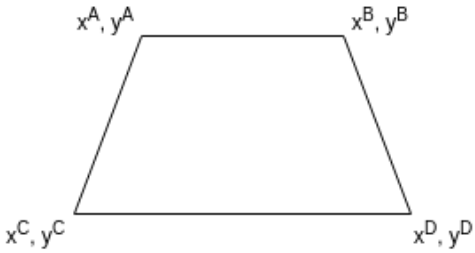

Illustration 4.3: Trapezoid with edge's name

Slashes line that have been found, can also be used to identify the trapezoid. The difference with triangles is the values of difference between xA and xB and also the difference between xC and xD. One of the difference values must be higher than the other difference value.

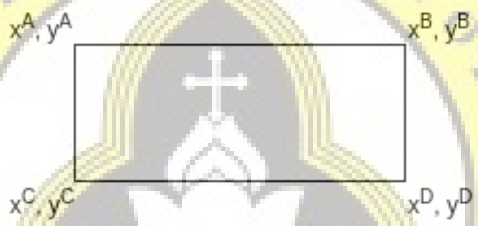

Illustration 4.4: Rectangle with edge's name

The square and rectangle have different requirements with triangle and trapezoid. Square and rectangle must have the difference between each point which is the length of each side. If square, the length of the four sides must be equal. However, if it is a rectangle, the length of the left side must be the same as the length of the right side and the length of the upper side must be the same as the length of the bottom side. Also, if it's rectangular, the length of the left or right side must be bigger or smaller than the length of the top or bottom side.

4.2 Desain

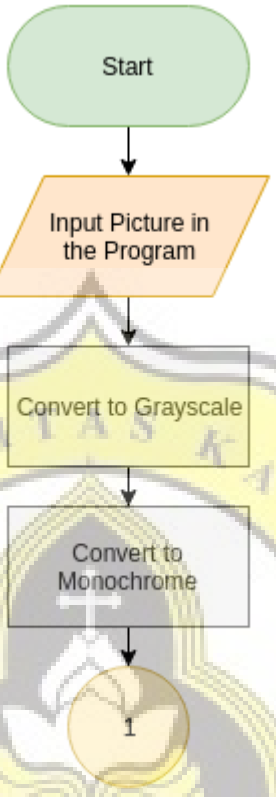

Illustration 4.5: Flowchart of making monochrome image

As already explained, the steps to identify geometries are inputting images, processing images into grayscale, then monochrome, then after becoming a monochrome image, the image must be identified based on the type of geometries with the limits given. The steps which are process the input image until convert the colorize image into monochrome image have been shown in the illustration above. The other steps have been explained in illustration 4.6 until 4.9

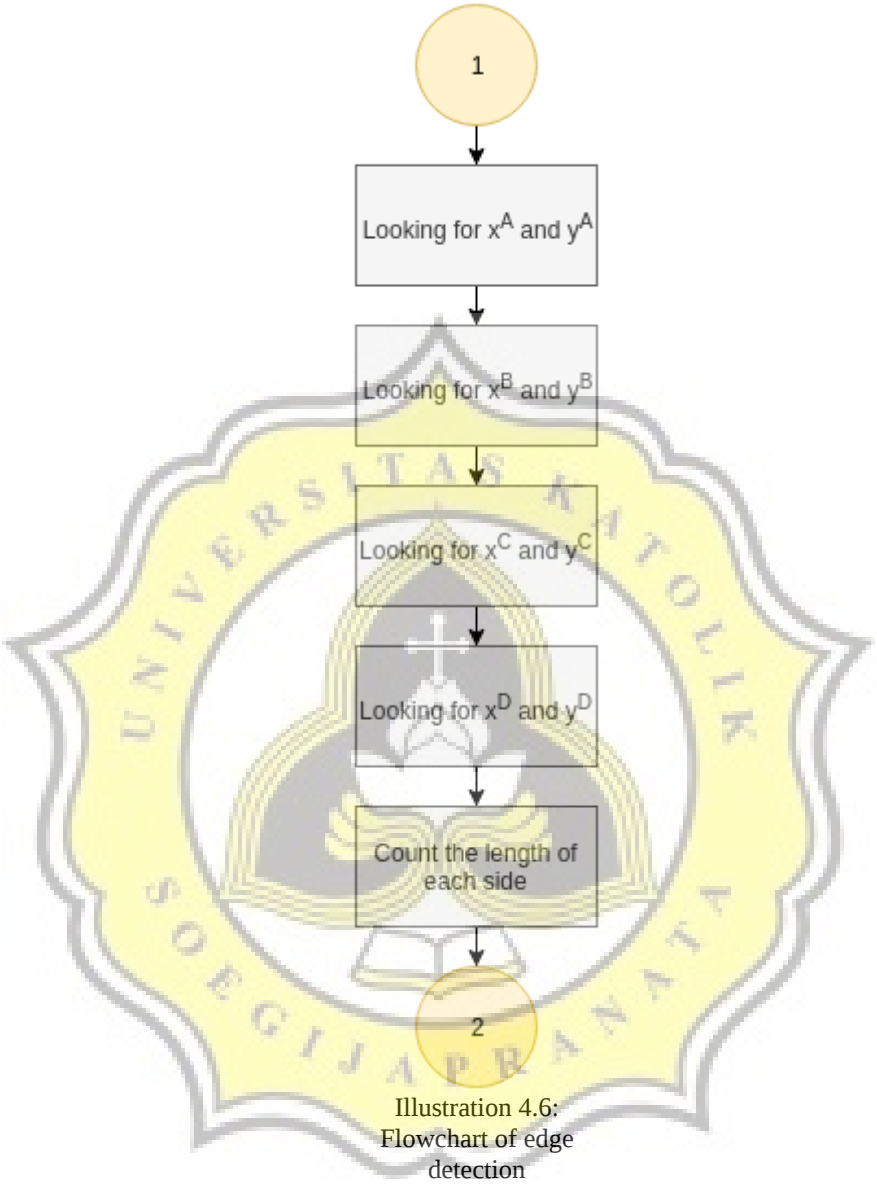

Illustration 4.6: Flowchart of edge detection

The flowchart displayed on illustration 4.6 shows the steps to find 4 edge points on the object. The point that must be saved first is the point x^A and y^A. And after get 4 points, the next step which can be done is count the length of each side.

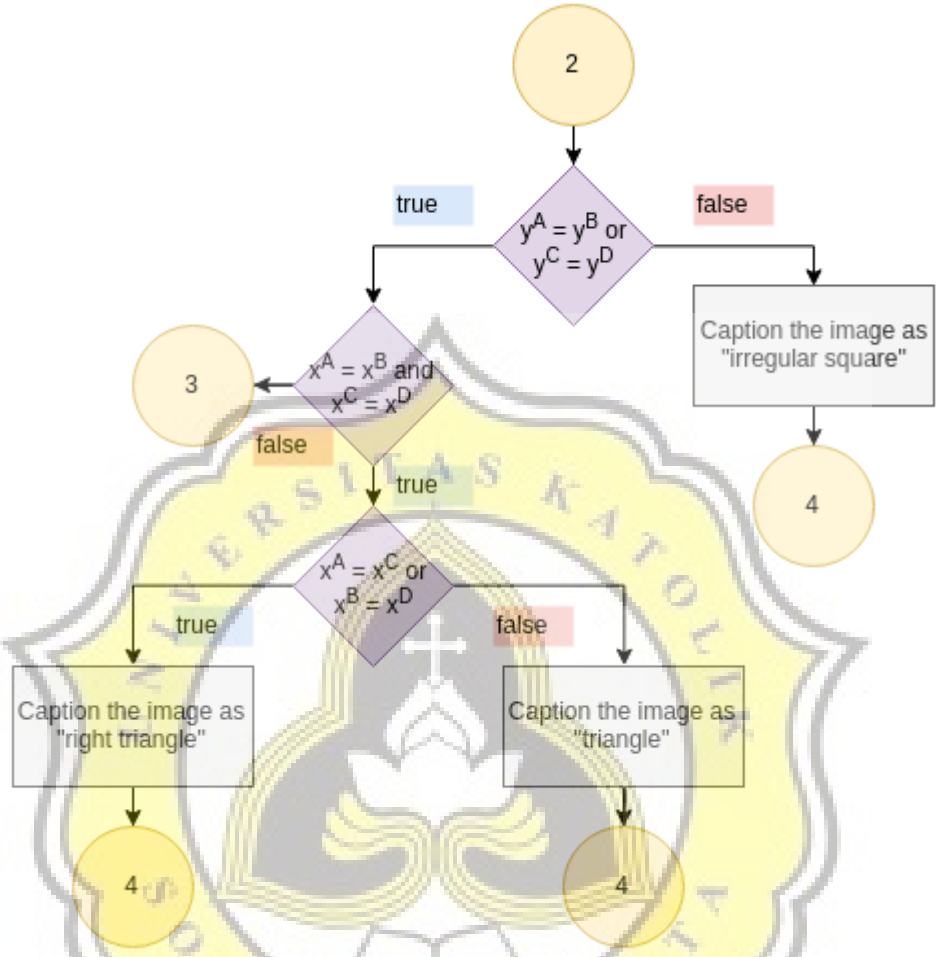

Illustration 4.7: Flowchart of identify triangle and irregular square

This flowchart explain how to recognition the geometries using the edges that have been found. And illustration 4.7 show the triangle recognition and irregular square.

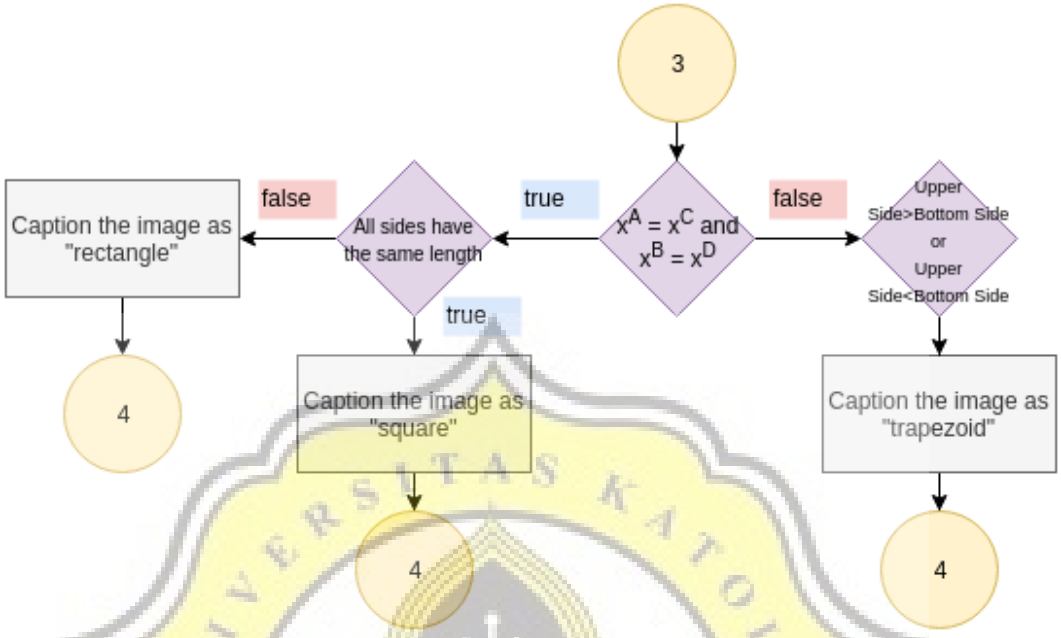

Illustration 4.8: Flowchart of identify square, rectangle, and trapezoid

And in this flowchart shows how object is recognition as a square, rectangle, and trapezoid. This recognition is still using point that has been found and length of each side. And from illustration 4.9 tells the final steps that consists of count area and output the results.

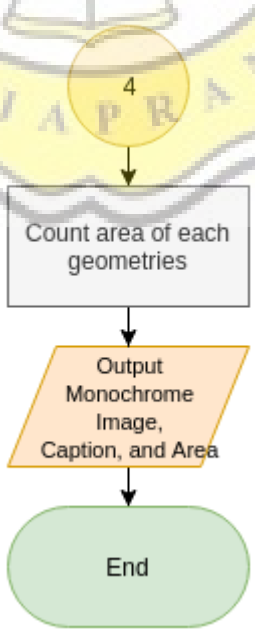

Illustration 4.9: Flowchart of the final steps