Activity 4: Measuring Area from Images

Mary Chris Roperos Go / 2014 - 11122

August 20, 2019

Edge-detection Using Python and Scilab

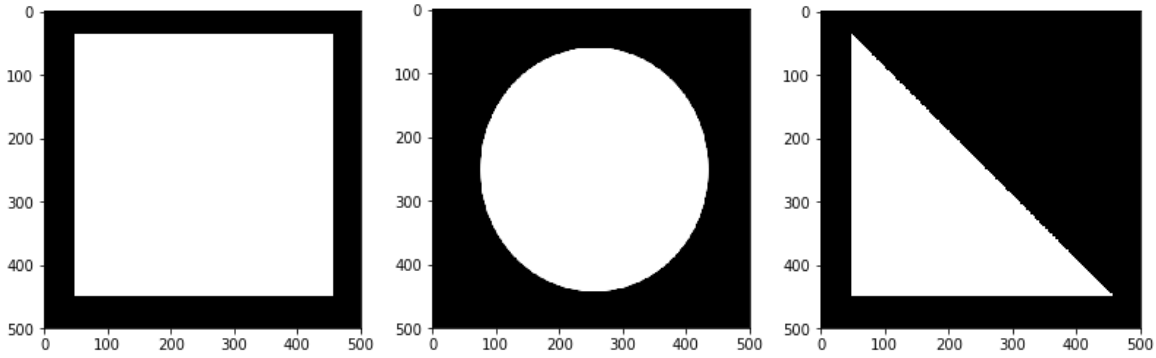

Figure 1: The following shapes were generated using GIMP.

Shape	**Area**
Square	12.11 cm^2
Circle	7.26 cm^2
Triangle	6.06 cm^2

Table 1: Summary of the generated shapes with their corresponding area.

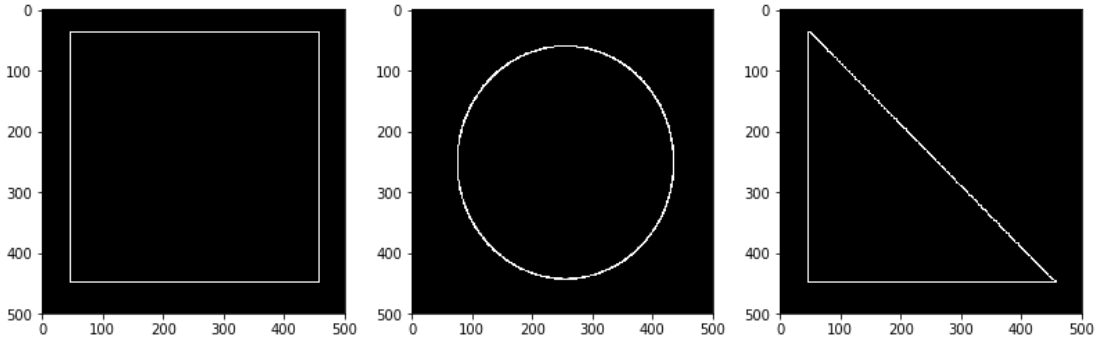

Figure 2: The edges of the shapes from Figure 1 was determined.

Using an image processing program, GIMP, shapes in figure 1 was generated. It is very important to read this images in single images, meaning figure 1 must be black and white containing 0 and 1 pixel values. The area of these shapes were known as seen in table 1. Edges of these shapes were determined using Scipy package in Python. I also tried detecting the edge using Scilab. I successfully implemented it using Sobel gradient estimator. The function conveniently finds the edges of the images. This activity is the introduction for measuring area of images. Now, we move to its application and the fun part!

Estimation of Area of the Shape with Green's Theorem and Edge-detection

Now, all the knowledge we gained from the first part of the activity, along with our knowledge of Green's theorem, will be used to get the image area (in number of pixels). Moving to the next steps, this activity is easily done in Scilab by extracting the pixel coordinates of the edge using find() function. Next, I looked for the centroid of the shape and subtract this from x,y coordinates of the edge. This pixel position was converted into r and theta using atan(). The returning list of coordinates was arranged into increasing angle by gsort(). I now applied Green's theorem to wrap up and achieve the goal. In table 2, I summarized the computation of the pixels from the three different shapes. The theoretical was analytically obtained using GIMP. The percent error was also presented.

Shape	**Circle**	**Square**	**Triangle**
Computed Area (in pixels)	124 895	124 870	62 484
Theoretical Area (in pixels)	116 000	168 000	83 435
Percent Error	8%	25%	25%

Table 2: The following values were obtained using Scilab.

The percent error for square and triangle were big. These errors tell us how the Green's theorem is not effective on shapes outside circle. High deviations were observed for squares and triangle. Maybe, the code I implemented was designed for a circle shape also. That is why bigger errors can be observed for shapes with corners.

Real-life Application of Green's Theorem and Edge-detection

Now, we gather all the ideas into an application of measuring the area via GoogleMaps. I chose the football field of Pamatasan ng Lungsod ng Maynila. A screenshot and a generated GIMP photo in figure 3 were used for this part.

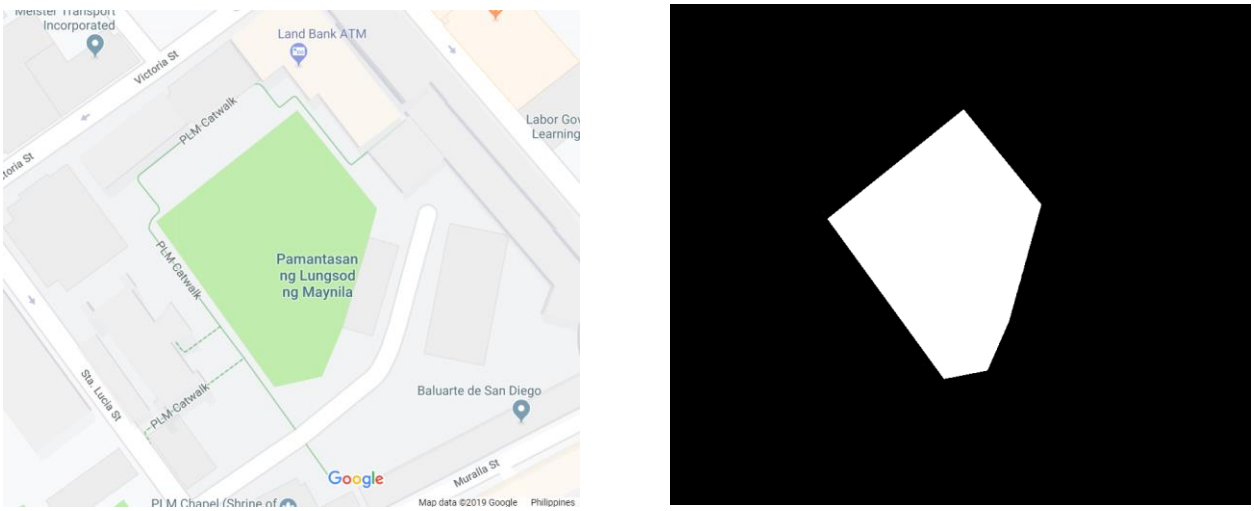

Figure 3: A screenshot from GoogleMaps (screenshot from GoogleMaps - left) and a generated photo using GIMP (right) were used for this activity.

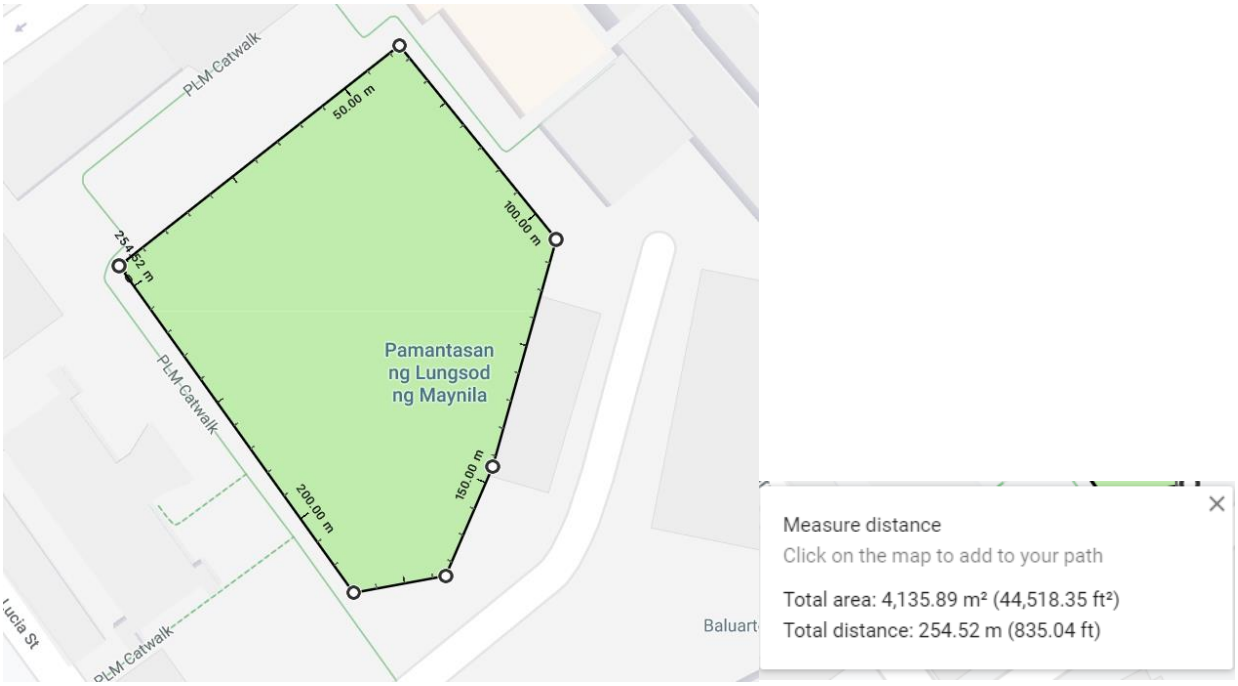

Figure 4: Google Maps features a convenient tool for measuring distance.

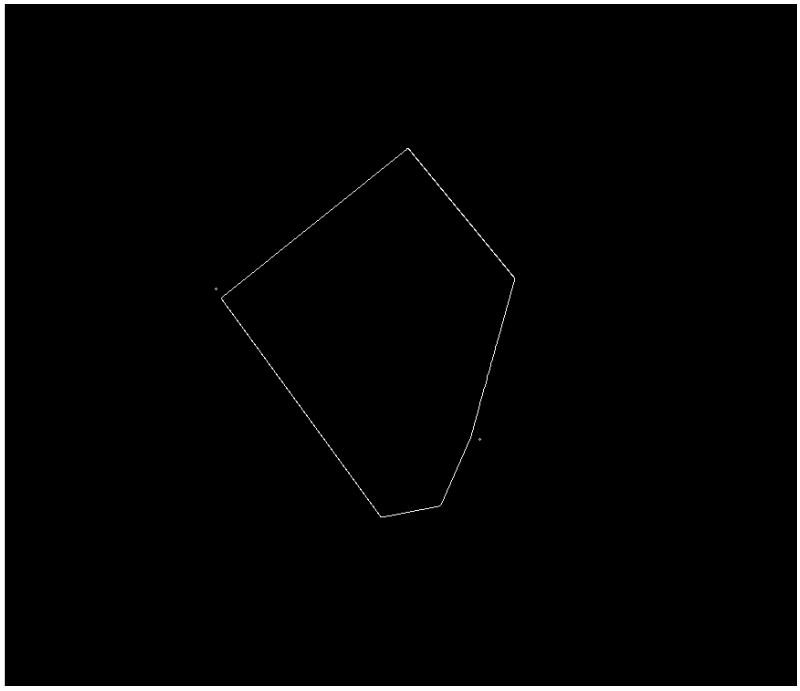

Figure 5: Edge of the chosen location were obtained using the same code from Green's theorem.

Using the same code for Green's Theorem, I was able to know the area of the chosen location through the help of the scale in GoogleMaps. I found out that there is an 95-pixel length from the legend.

From the code in Scilab, I found out that my chosen location is 328 808 sq. pixels. Dividing this by 95 pixels, I get a computed area of 3 461 sq. meters. From the theoretical value shown in Figure 4, the computed area deviated by 16%. The value it yielded is quite big. But I have to take note that tracing the edges using GoogleMaps is not the most efficient way to know the area of the chosen location. I tried to look for credible online sources regarding PLM's football area and found none. Furthermore, the accuracy will also depend on the user's tracing of boundaries shown in figure 4. After tracing, GoogleMaps automatically approximated the distance covered. I believe this method affected the error in a large scale. Also, this was expected as the code used showed larger percent deviations on shapes that is not circle and irregular. The code was a bit inefficient in getting the pixel area of the image.

All in all, this was a very challenging activity but fun activity. I can say that I really enjoyed this seatwork. It was a bit frustrating at first, but once you get the gist of it, everything will follow. What I like about this activity is how I can see its application. Although, I was hoping to have a better knowledge and I wish to have more related activities until we get a lesser percent deviation.

Self-evaluation

Technical correctness	4
Quality of presentation	5
Initiative	2
Total	11

References

1. 'Measuring Area from Images', 2019 Applied Physics 186 manual by Dr. Maricor Soriano
2. Google Maps