Sheenah A. Ancheta
2014-28353

Activity 4 – Measuring area from images

Regular geometric shapes were created on an image processing software GIMP. I used this to easily get the area of the shapes easily since in GIMP, you can see the dimensions of your shape expressed in your preferred units— pixels, centimetres, inches, etc. The shapes I made are circle, square, and rectangle:

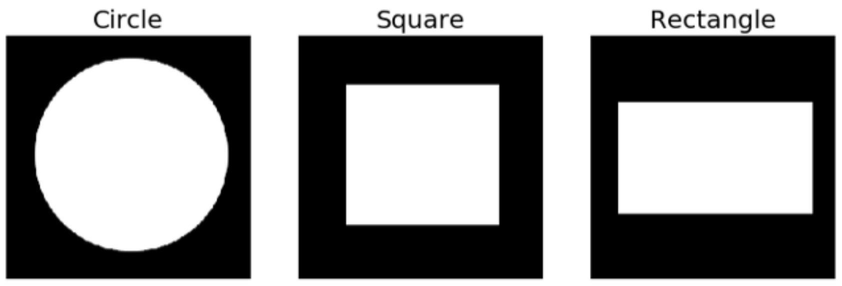


Figure 1. Shapes created through GIMP in a 256 x 256 canvas.

The corresponding areas of these shapes are shown in Table 1:

Table 1. Theoretical area of the shapes shown in Figure 1.

	Theoretical area (pixels)
Circle	32365.47
Square	90000
Rectangle	23868

The values for area were obtained by choosing a certain region of the shape to get its pixel value then substituting this to the corresponding area formula of the shape. For example, through GIMP, I measured the diameter of the circle in pixels and used this to get its area with the formula, circ_area = πr^2.

These shapes were then opened on a Jupyter notebook using `matplotlib.image`. The edge images of the following shapes were then obtained using `ndimage` imported from `scipy`. Edge detection is an image processing technique used to obtain the boundaries of an object within an image. One of the commonly used edge detection algorithms is the Sobel method. [1] The Sobel method is an approximation to a derivative of an image. The Sobel operator separates its process on the x- and the y-directions. [2] This is later combined to get the Sobel edge figure.

The following code were used for the Sobel edge detection of the circle in Figure 1,

```
csx = ndimage.sobel(circ, axis=0, mode='constant')
csy = ndimage.sobel(circ, axis=1, mode='constant')
          csob = np.hypot(csx, csy)
```

and it has the following output:

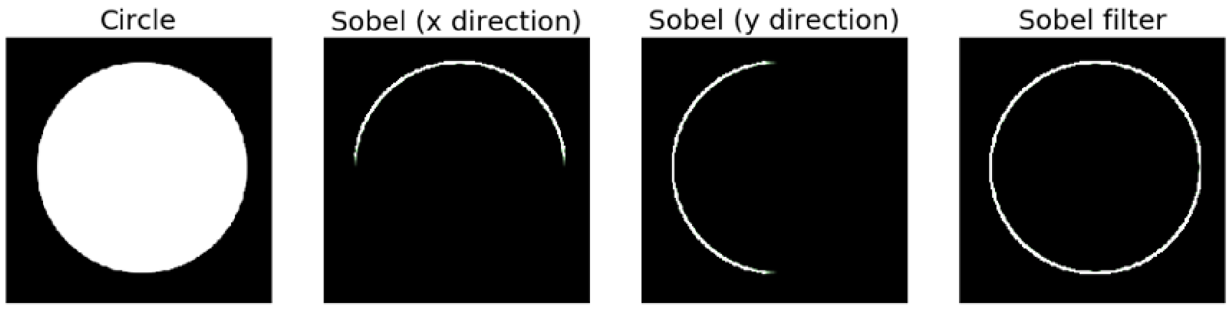


Figure 2. The circle image after edge detection using the Sobel method.

Using the same process, the same edge detection method was also used for the square and rectangle shapes as shown in Figure 3.

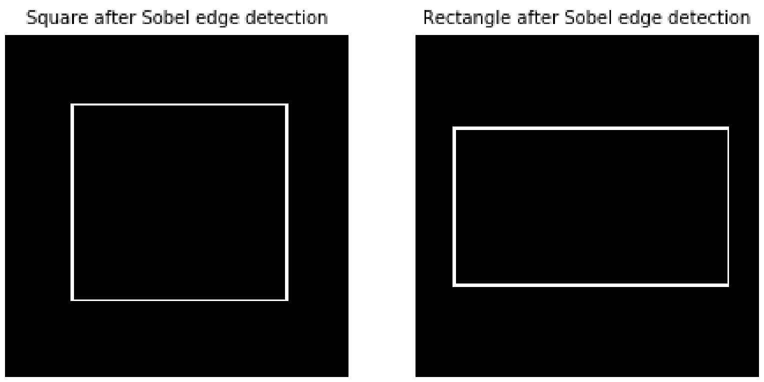

Square after Sobel edge detection
Rectangle after Sobel edge detection

Figure 3. The circle image after edge detection using the Sobel method.

Using SciLab, I then got the edge pixel coordinates by finding the pixel position of the centroid of the shape and subtracting this from the x,y coordinates of the edge:

Using SciLab, I obtained the x- and y- coordinates of the edge image. The pixel position of the centroid of the shape was obtained by getting the values of the length and the width of the shape and dividing it by 2. Finally, the edge pixel coordinates were obtained by subtracting the pixel position of the centroid of the shape— width/2 and length/2— from the x- and y- coordinates, respectively.

```
coordinates = edge(circ, ‘sobel’)
[length,width] = size(circ); //to get centroid
[x,y] = find(coordinates); //x- and y- coordinates
N = length(x);
```

Each corresponding pixel values were then converted into r and theta θ:

```
                For i = 1:N
xpixel = x(i) - width/2; //x-edge pixel coordinate
ypixel = y(i) - length/2; //y-edge pixel coordinate
 r = sqrt(xpixel^2 + ypixel^2) //formula for r
 theta = atan(ypixel,xpixel); //formula for θ
```

To define the contour of the edge in order, the coordinates were then arranged into increasing angle using `gsort()`. Green's theorem was thereafter implemented to obtain the total image area in unit pixels (see Appendix for the code used).

The computed area for the three shapes used are as follows:

Table 2. Computed area of the edge images of the shapes shown in Figure 1.

	Computed area (pixels)
Circle	32181
Square	89997
Rectangle	23865

and comparing Tables 1 and 2 to get the percent error of the values of the area accumulated,

Table 3. Computed area of the edge images of the shapes shown in Figure 1.

	Theoretical area (pixels)	**Computed area** (pixels)	**Percentage error** (%)
Circle	32365.47	32181	0.5699593
Square	90000	89997	0.0033333
Rectangle	23868	23865	0.0125691

It should be noted that the values of the percent error are relatively small. As seen on Figures 2 and 3, the Sobel method returned edge images which really covered all the edge points of the shapes. Since the edges were really visible, the area of the edge images were expected to follow the theoretical values solved earlier.

Lastly, the accuracy of the Green's theorem was put into test when an irregular-shaped image was used. In Google Maps, I grabbed a snapshot of one of my favourite places in Katipunan— Regis Centre. Using GIMP, I turned this snapshot to a black and white image by turning the inside of the building white and its outside into black.

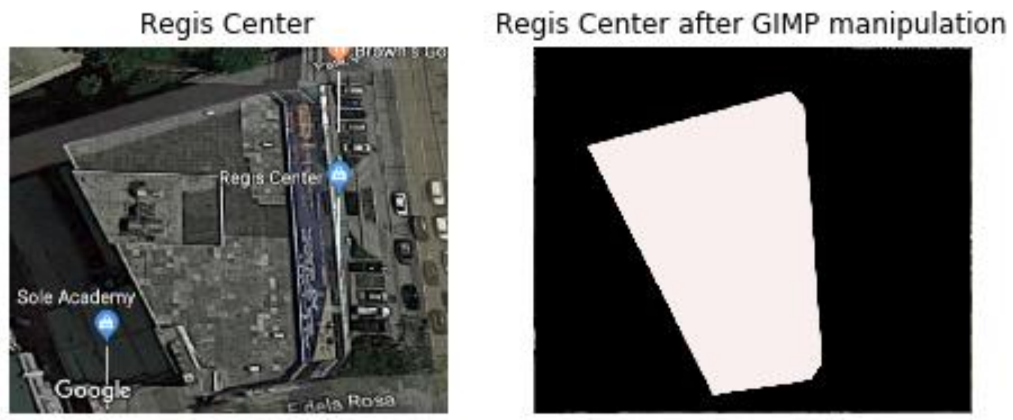


Figure 4.
Google Maps snapshot of Regis Centre (left) and its black and white counterpart (right).

Using the `measure distance` tool in Google Maps, the theoretical area of Regis centre was approximated. The theoretical area obtained was 1210.45 m^2. Again, to get the experimental area of this irregular shape, we use the Green's theorem (see appendix for the implementation of Green's theorem in the program). Since the theoretical area is expressed in square meters, the obtained area in the code was converted from square pixels to square meters. I did this by getting a screenshot off the scale in Google Maps. I then opened this snapshot in GIMP to get the value of this scale in pixels (see Appendix). I found out that the scale 10 meters is equal to 70 pixels. Using simple calculations, 1 square pixel is equal to 0.020408 square meters (1 $pixel^2$ = 0.020408 m^2). From the Green's theorem in my code, the calculated area was 141084.55 $pixel^2$. Converting this to square meters using the pixel-to-meter conversion in red text, the area is 2879.25 m^2. This gives a percent deviation of 1.38 %. There was a percent deviation because the accuracy of getting the theoretical area in Google Maps depends on the person doing it—the accuracy of placing the lines on the exact edges of the location.

Applying the Sobel edge detection technique on the black and white snapshot image of Regis Centre, the following edge image was obtained:

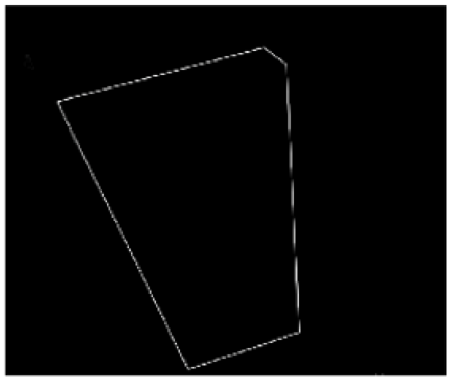


Figure 5. The snapshot of Regis Centre after edge detection using the Sobel method

Overall, this activity was so far the most fun I have ever done. I learned a lot from this and the edge detection techniques will surely help me in my future encounters for image manipulation.

Self Evaluation

Technical correctness	5
Quality of presentation	5
Initiative	2
Total	**12**

References

[1] The MathWorks, Inc. (2019). *Edge Detection*. Last accessed on August 20, 2019 from `https://www.mathworks.com/discovery/edge-detection.html`

[2] Ashish. (2018). *Understanding Edge Detection (Sobel Operator).* Last accessed on August 20, 2019 from `https://medium.com/datadriveninvestor/understanding-edge-detection-sobel-operator-2aada303b900`

Appendix

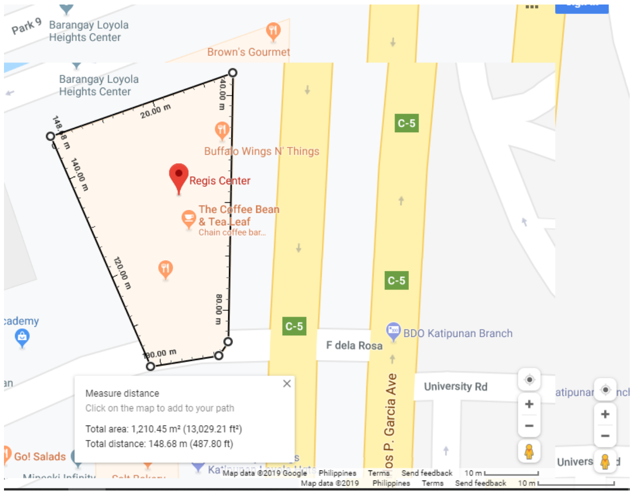
Green's theorem algorithm:

```
area = 0;

for i = 1:N
    if i<N then
        area = area + 0.5 * (sortedxy(i,1) * sortedxy(i+1,2) -
sortedxy(i+1,1) * sortedxy(i,2));
    else
        area = area + 0.5 * (sortedxy(i,1) * sortedxy(1,2) -
sortedxy(1,1) * xsortedxy(i,2));
    end
end
```

Sheenah A. Ancheta
2014-28353

Pixel-to-meter conversion:



Measure distance
Click on the map to add to your path
Total area: 1,210.45 m² (13,029.21 ft²)
Total distance: 148.68 m (487.80 ft)

Figure 6. Image used in getting the pixel-to-meter conversion using the scale (10 m) located on bottom right.