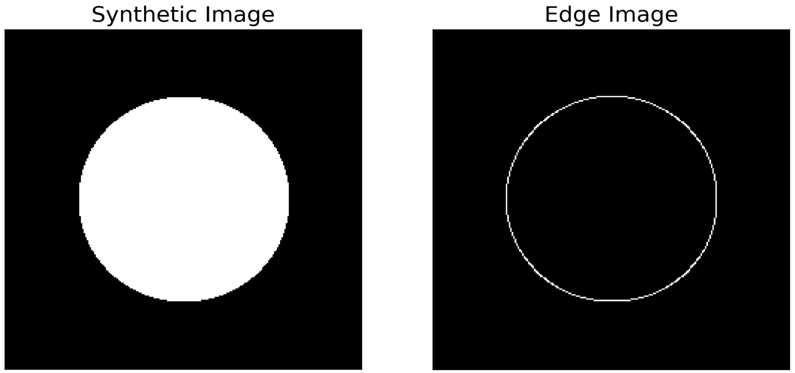
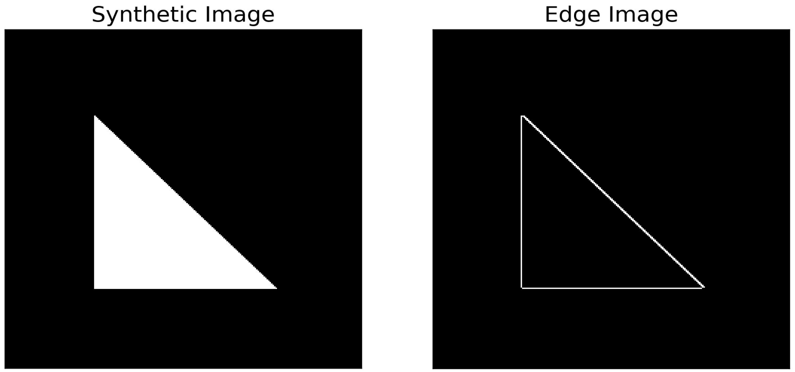
Activity 4 - Measuring Area from Images

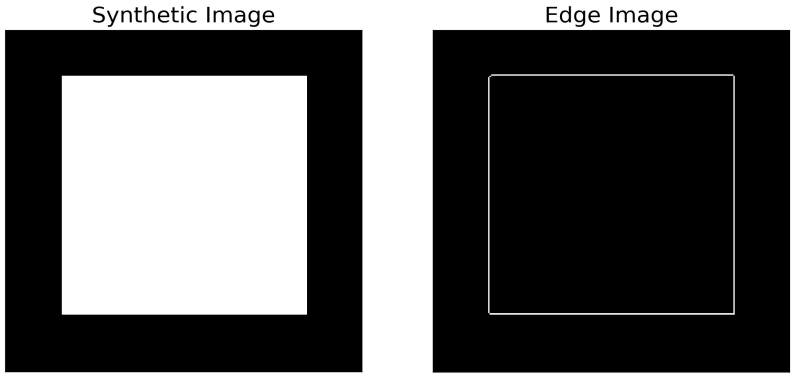
Crizzia Mielle de Castro | 2015-08076

Edge detection of synthetic images

The synthetic images were created using Paintbrush. The programming software used for this exercise was *Python*. To perform edge detection, the package *OpenCV* has several possible commands. I tried both *Laplacian* and *Canny*. The following edge images were obtained using *Canny.*







Finding centroid of shape

The indices of the edges can be obtained using *numpy's nonzero* command. These indices can be plugged into the following equation to obtain the centroid of the shape. This is the formula for a shape made up of a finite set of points.

$$\mathbf{C} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k}$$

Sorting points of shape

The coordinates of the centroid was subtracted from the coordinates of the edge points. These were then converted to polar coordinates and arranged in increasing angle. It was important to use *numpy's arctan2* to make a full 360^o sweep.

Calculating area using Green's Theorem

The sorted x and y coordinates were all plugged into the following equation to calculate the shape's area. The last pixel of the sweep is paired with the first.

$$A = \frac{1}{2} \sum_{i=1}^{N_b} [x_i y_{i+1} - y_i x_{i+1}]$$

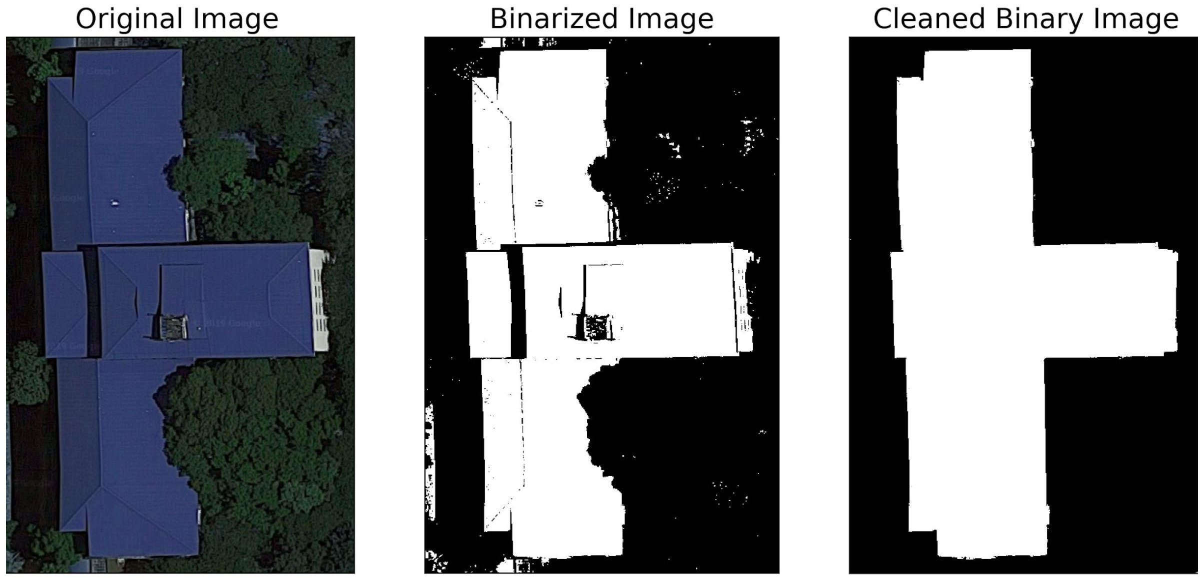
Theoretical vs calculated area (in pixels) of synthetic images

These were the results for the synthetic images. Both edge detection techniques have the lowest deviation for the circle, and highest for the triangle. The difference between the two techniques are very small (3-4 pixels), so either could be used. However, *Canny* generally has a slightly smaller deviation.

shape	**theoretical**	***Laplacian***	**% dev**	***Canny***	**% dev**
rectangle	27388	27721	1.22	27717	1.20
triangle	7442	7627	2.49	7624	2.45
circle	16061	16189	0.80	16185	0.77

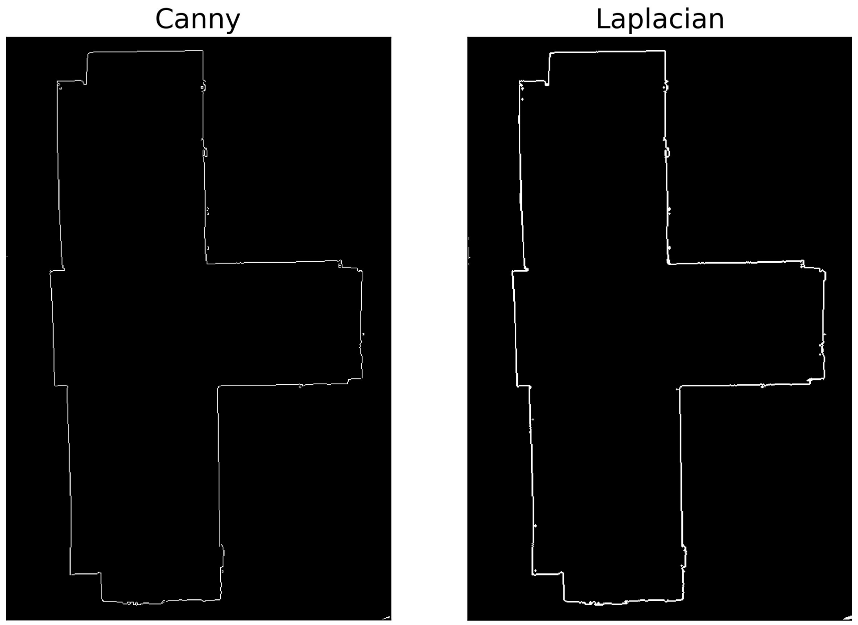
Calculating the area of a place of interest

The place of interest I used was the Main Library in UP Diliman. After taking a screenshot of the satellite image in *Google Maps*, I binarized the image and cleaned it up using *Gimp*. This is to make calculations easier and more accurate. I did the exact same steps as the synthetic shapes to calculate its area.



Calculating the area of a place of interest

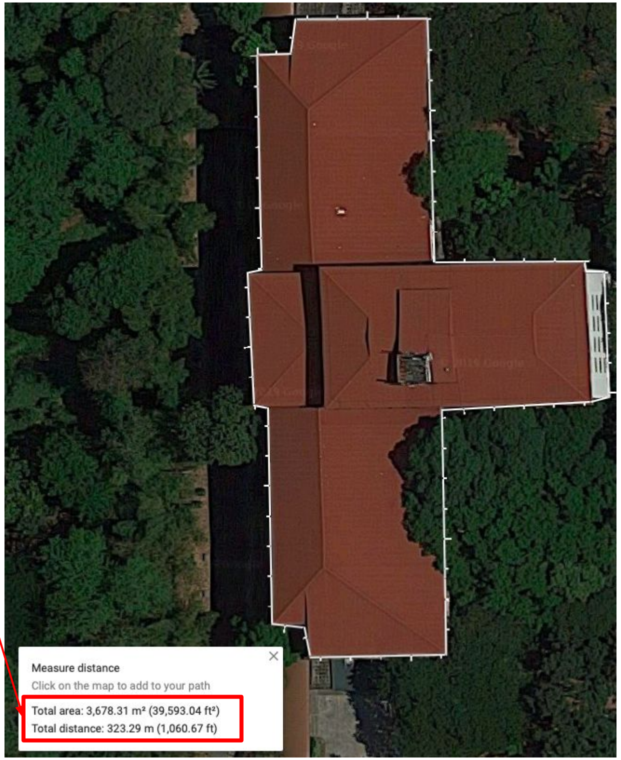
Canny calculated 153673 pixels, while *Laplacian* calculated 154040.5 pixels for the area of this shape. Using *Gimp*, I checked the pixel to meter conversion to be 63 px:10 m or 3969 px:100 m^2. From this, *Canny* converts to 3871.83 m^2, while *Laplacian* converts to 3881.09 m^2.



Comparing the theoretical and calculated values

The theoretical value was obtained using *Google Maps's* command to measure distance. By forming an enclosed shape, it can also calculate the area.

The theoretical area of the Main Library is 3678.31 m^2. *Canny* and *Laplacian* deviates by 5.26% and 5.51%, respectively. Similar to the synthetic shapes, *Canny* produced a slightly smaller deviation. I only inferred the shape of the building covered by the trees, so this may have contributed to the deviations. My trace in the image to the right may also be different from the detected edges of the cleaned binary image. There is also a chance that I may not have perfectly cleaned the background of the building to be black.



Measure distance
Click on the map to add to your path
Total area: 3,678.31 m² (39,593.04 ft²)
Total distance: 323.29 m (1,060.67 ft)