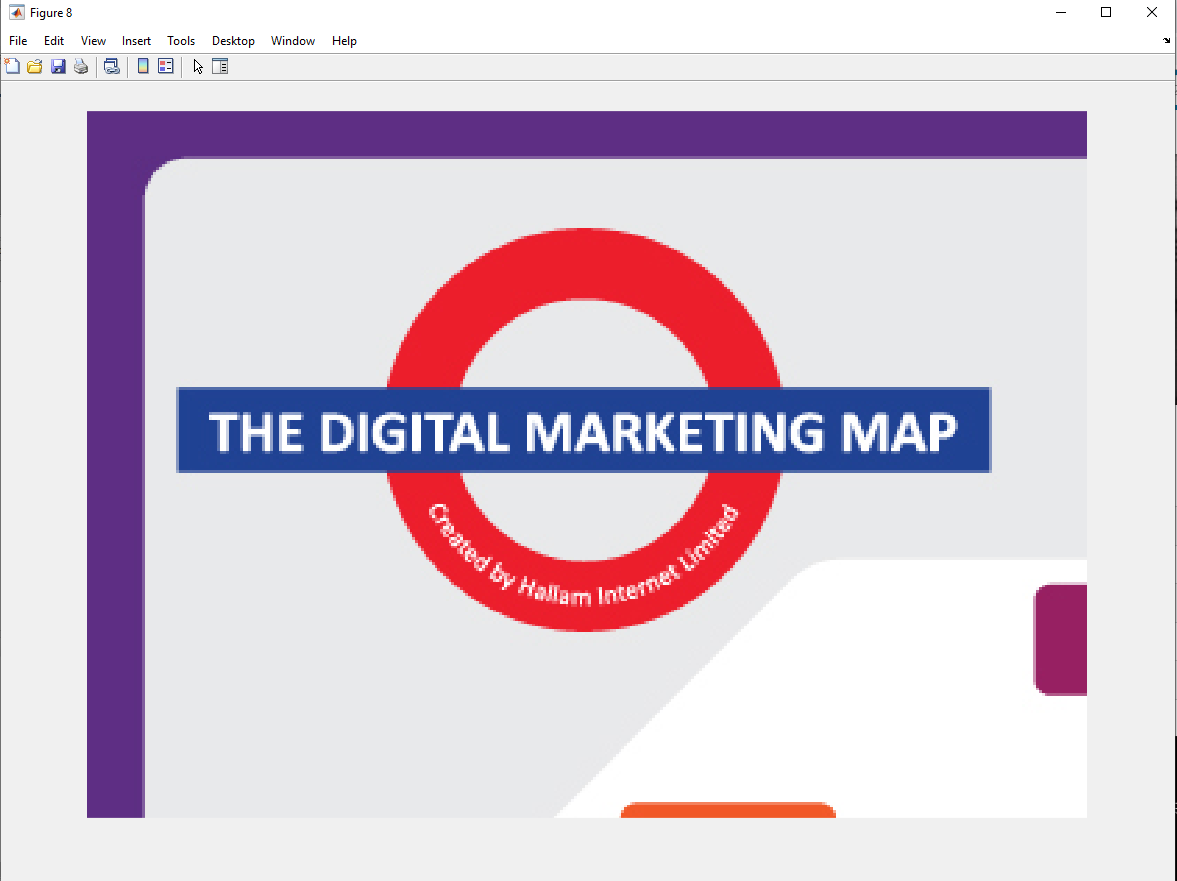
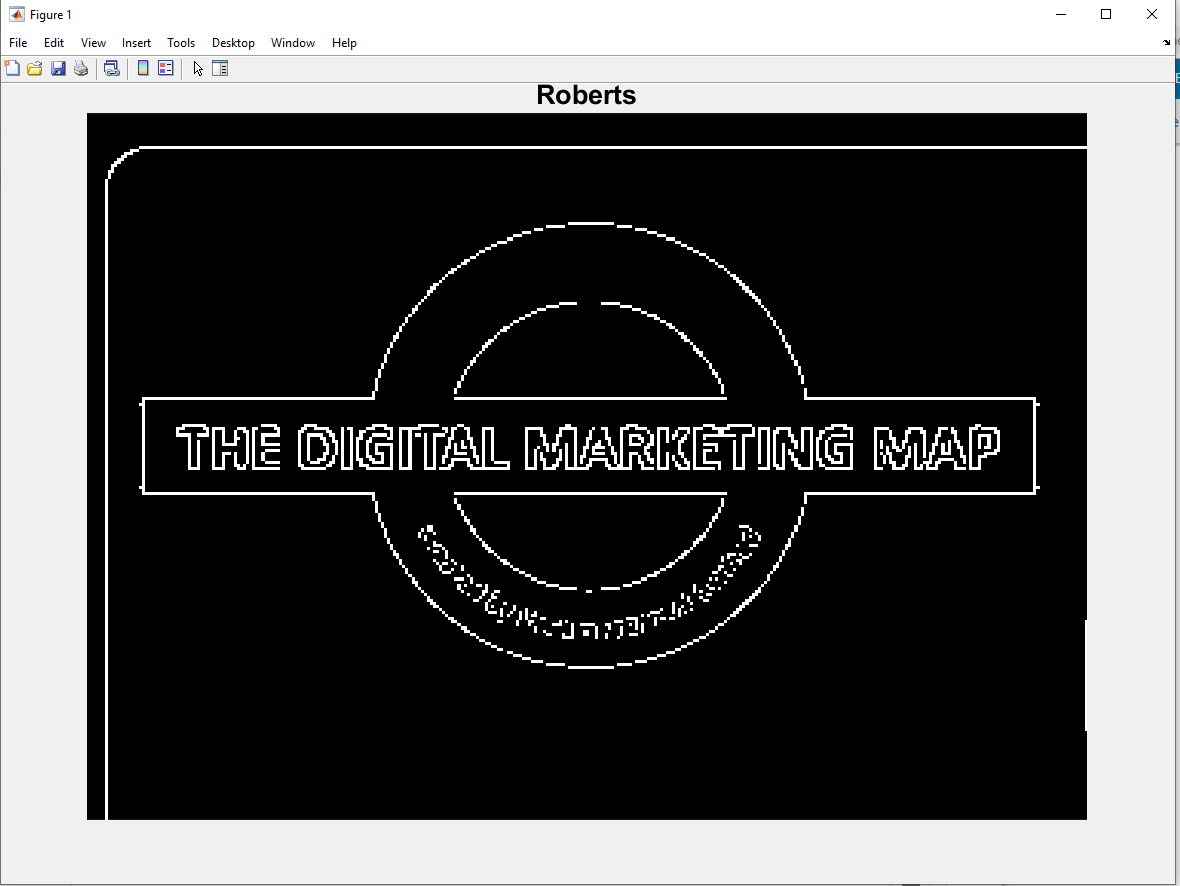
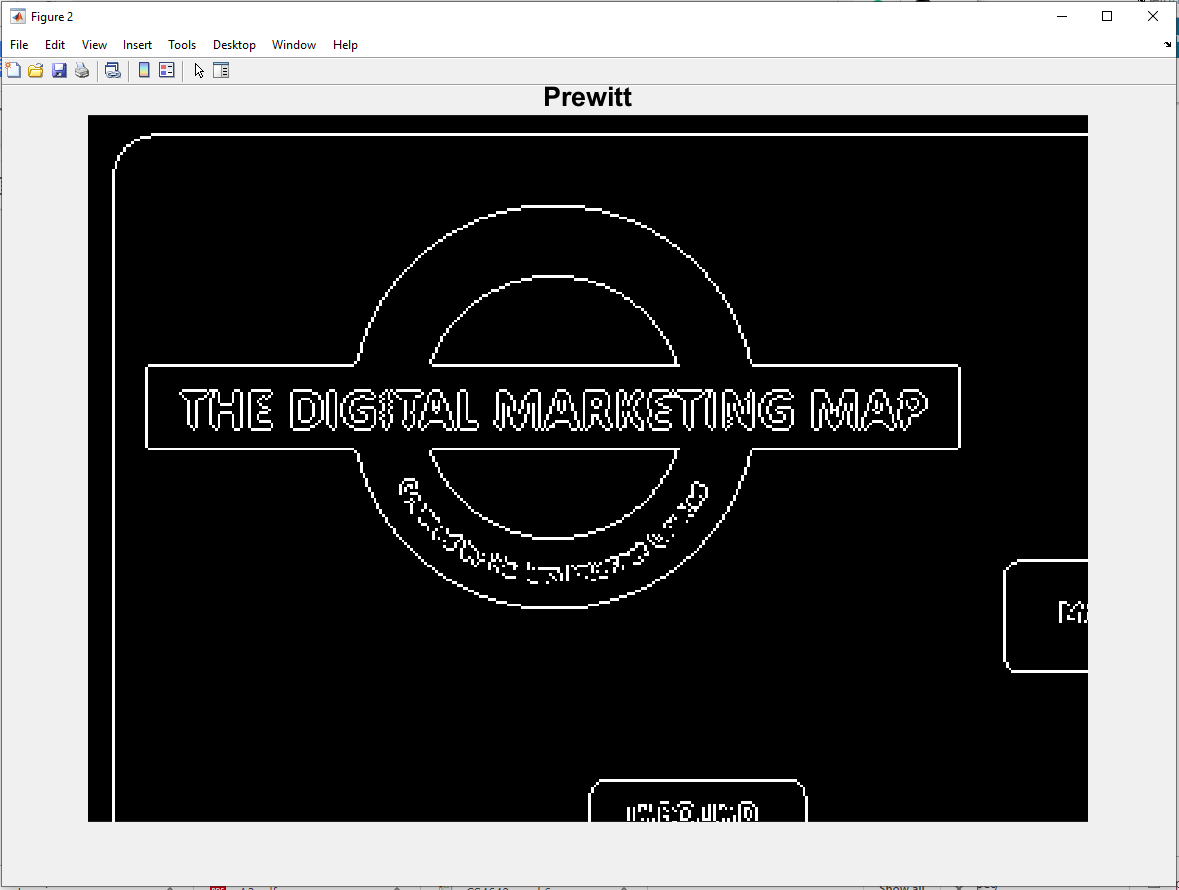
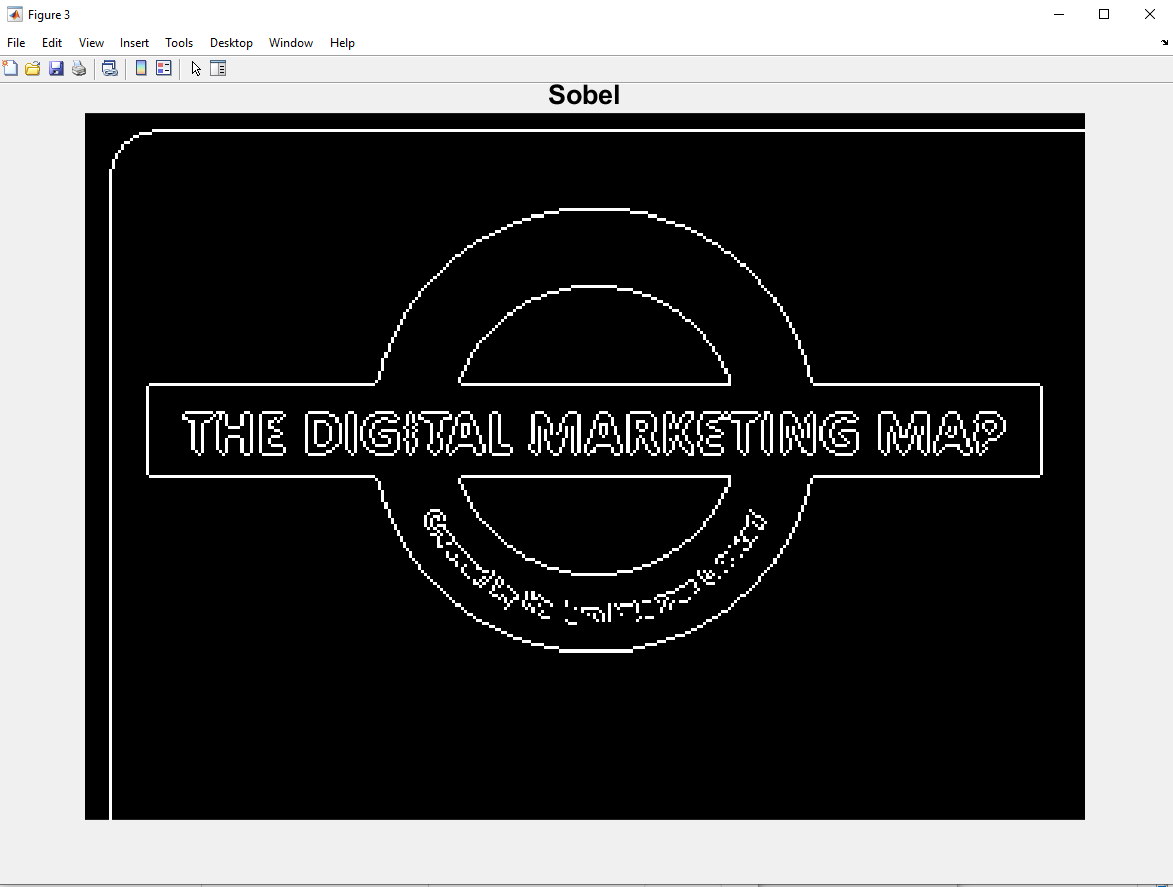
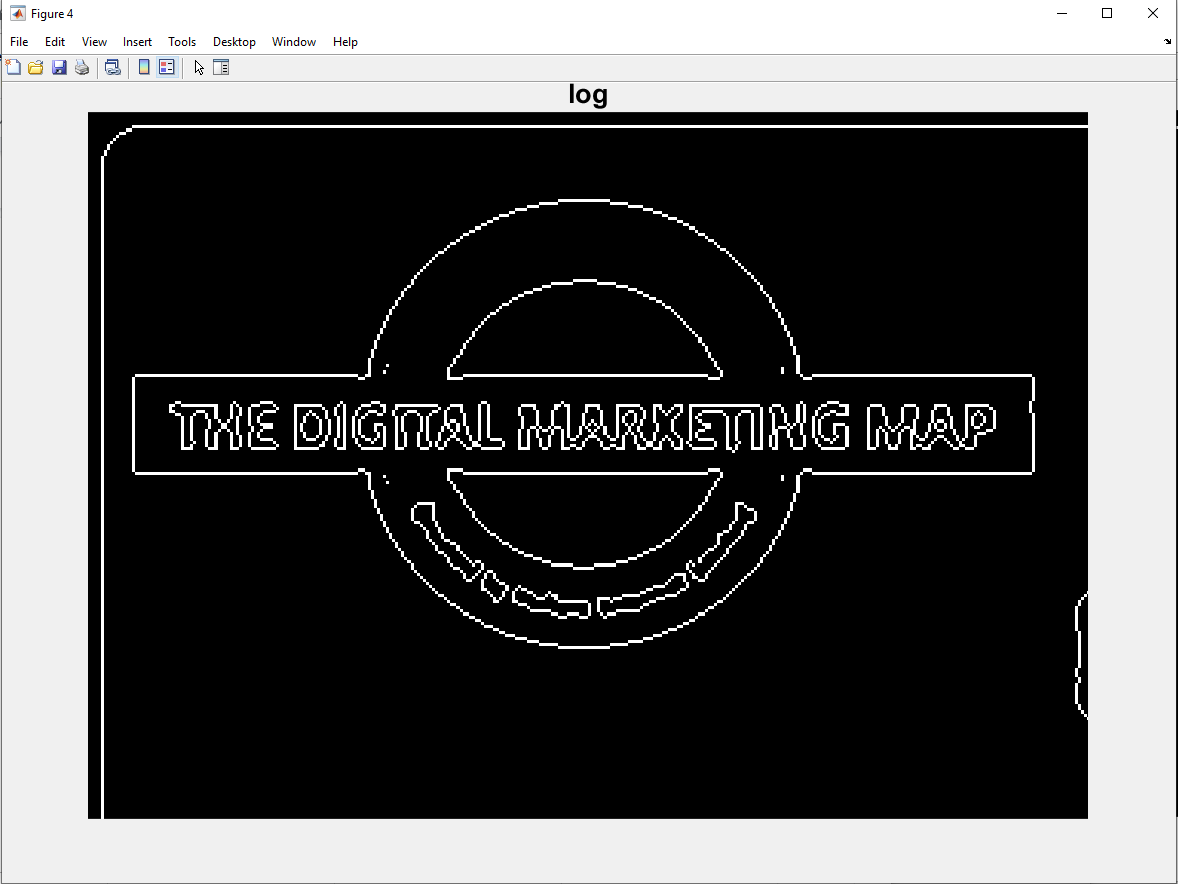
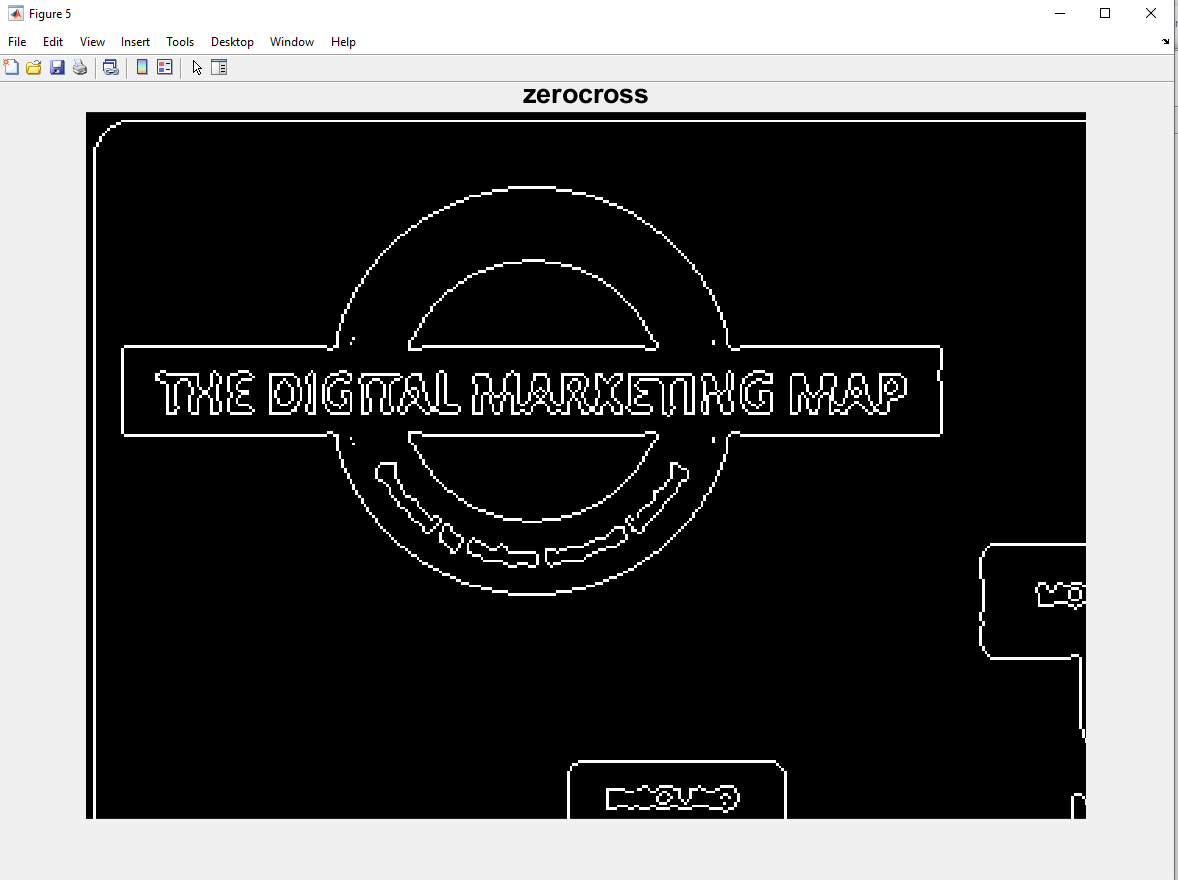
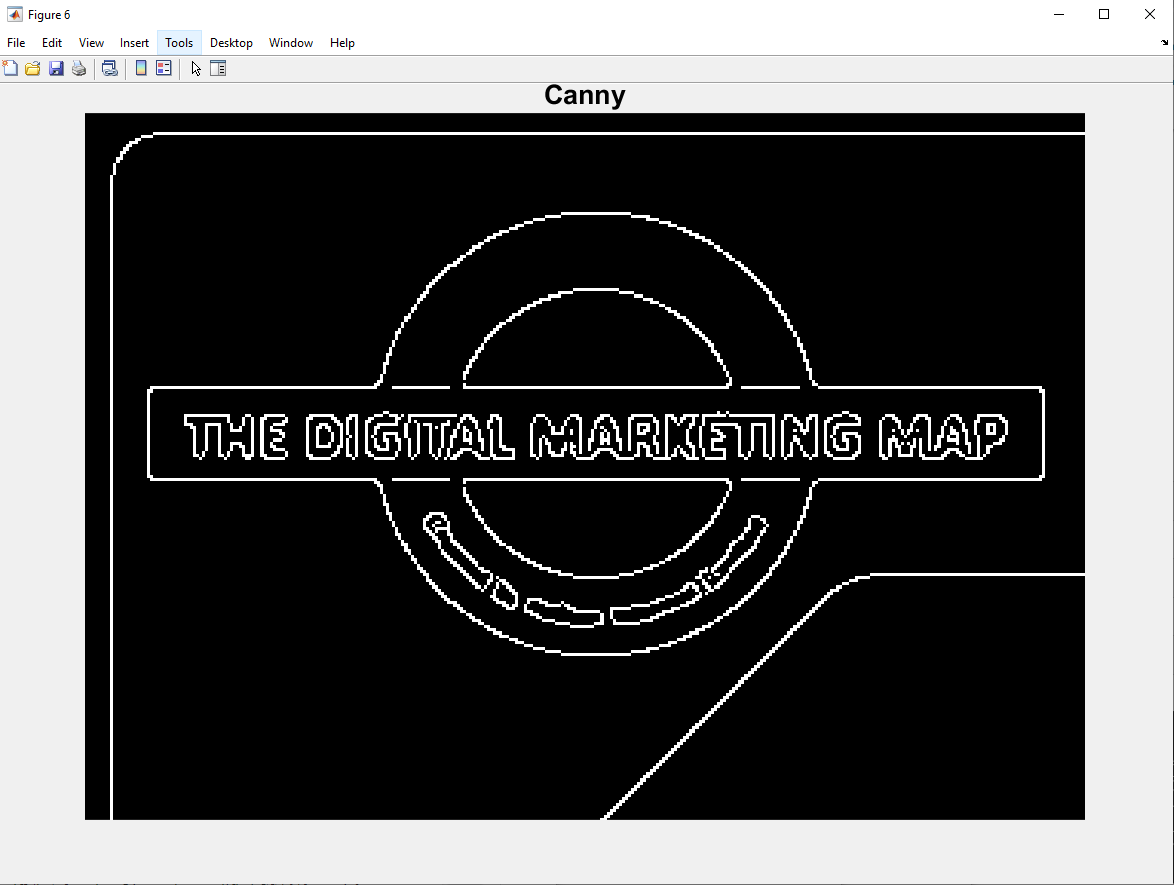
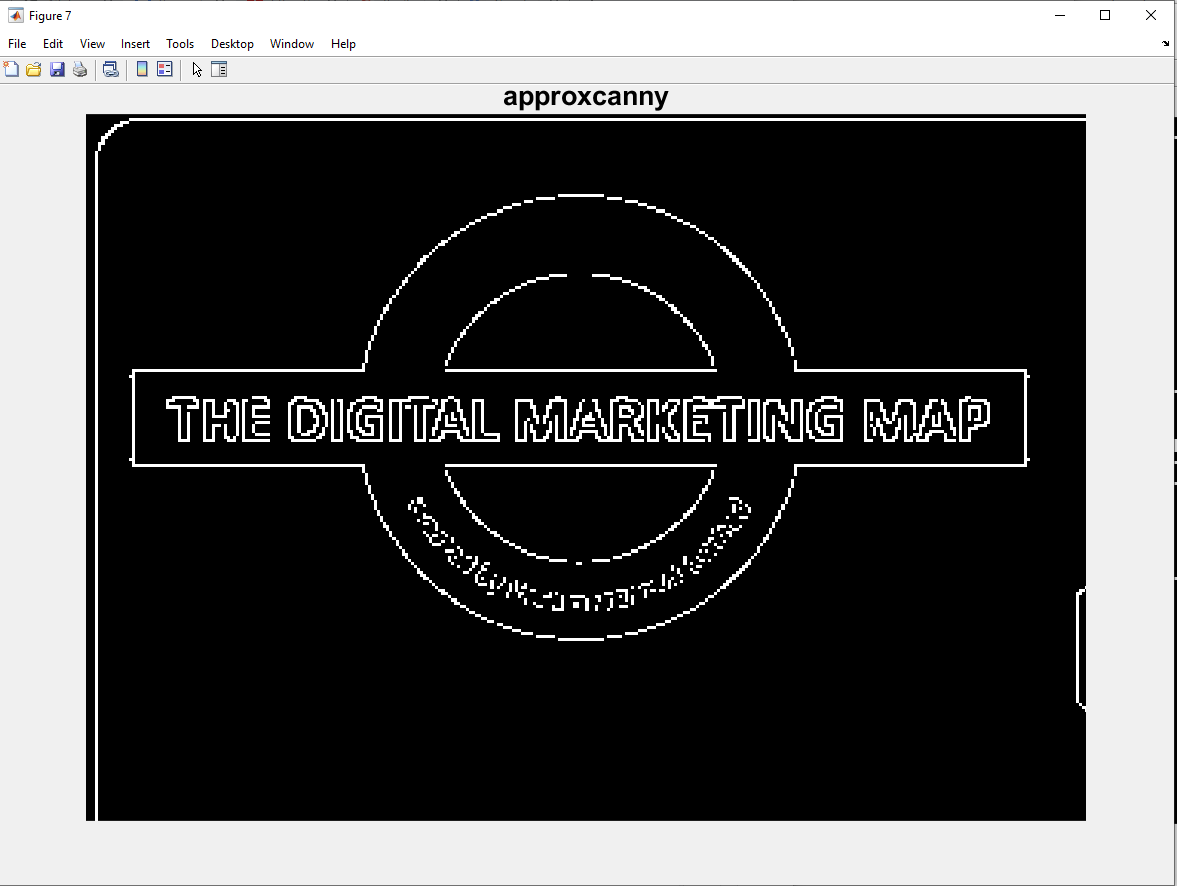
CS4640 A<3>

<Yingjie Lian>

September 26, 2019

**Question 1 Answer:**

Here is the original image seg of metro.jpg. This is from the upper-left corner of metro.jpg. Since the edge function did not apply a big difference each other, so I will have to zoom in and analyze it. 

**Original**Finds edges at those points where the gradient of I is maximum, using the Roberts approximation to the derivative.****Finds edges at those points where the gradient of I is maximum, using the Prewitt approximation to the derivative.****Finds edges at those points where the gradient of the image I is maximum, using the Sobel approximation to the derivative.****Finds edges by looking for zero-crossings after filtering I with a Laplacian of Gaussian (LoG) filter.****Finds edges by looking for zero-crossings after filtering I with a filter that you specify, h****Finds edges by looking for local maxima of the gradient of I. The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.****

Finds edges using an approximate version of the Canny edge detection algorithm that provides faster execution time at the expense of less precise detection. Floating point images are expected to be normalized in the range [0 1].

**Question 2 Answer:**

For this question, I have implemented my CS4640\_df3(im) function. I have also modified CS4640\_week6.m file in order to see the results.

If you want to see the results and figures, you will have to run CS4640\_week6 in matlab command.

Here are the results that I got:

num\_pix =26747200

num\_tp = 6078445

num\_rob =1315979

ans =0.2165

tp =1073704

ans = 0.1766

L =

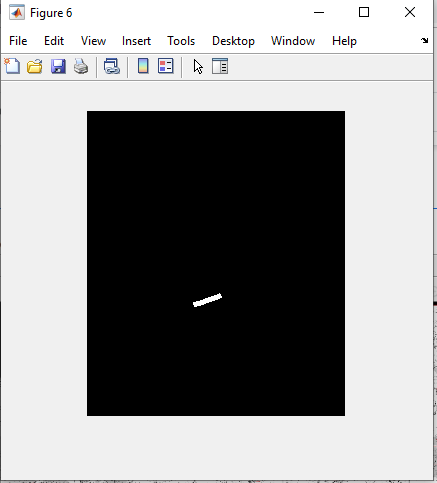
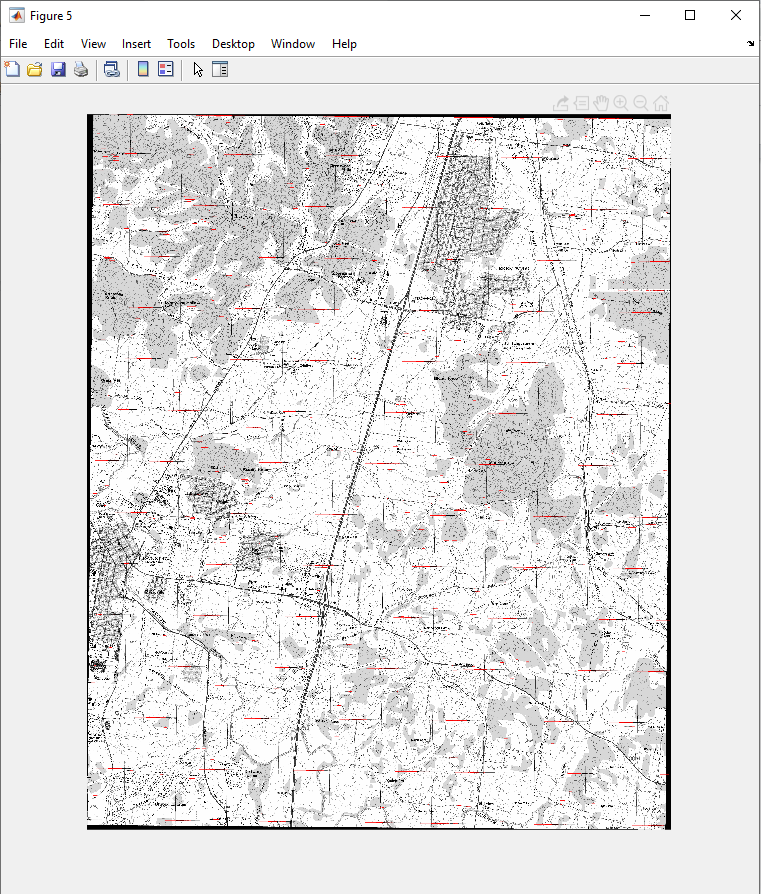
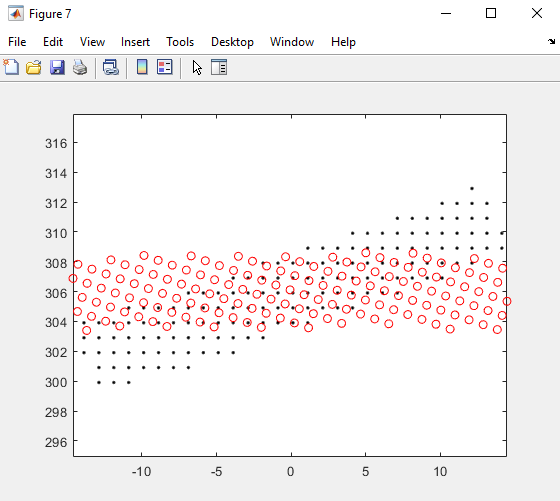
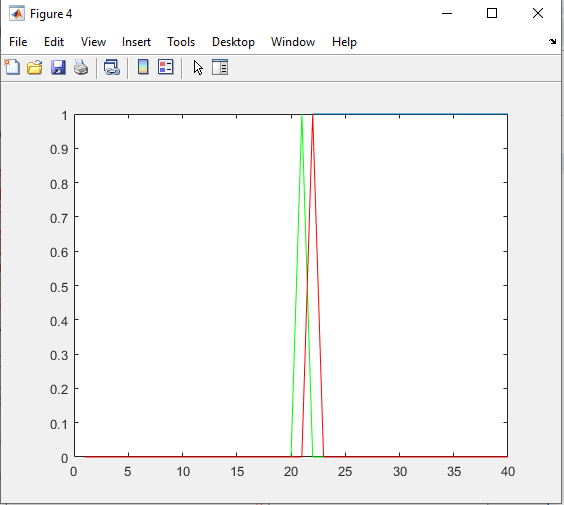
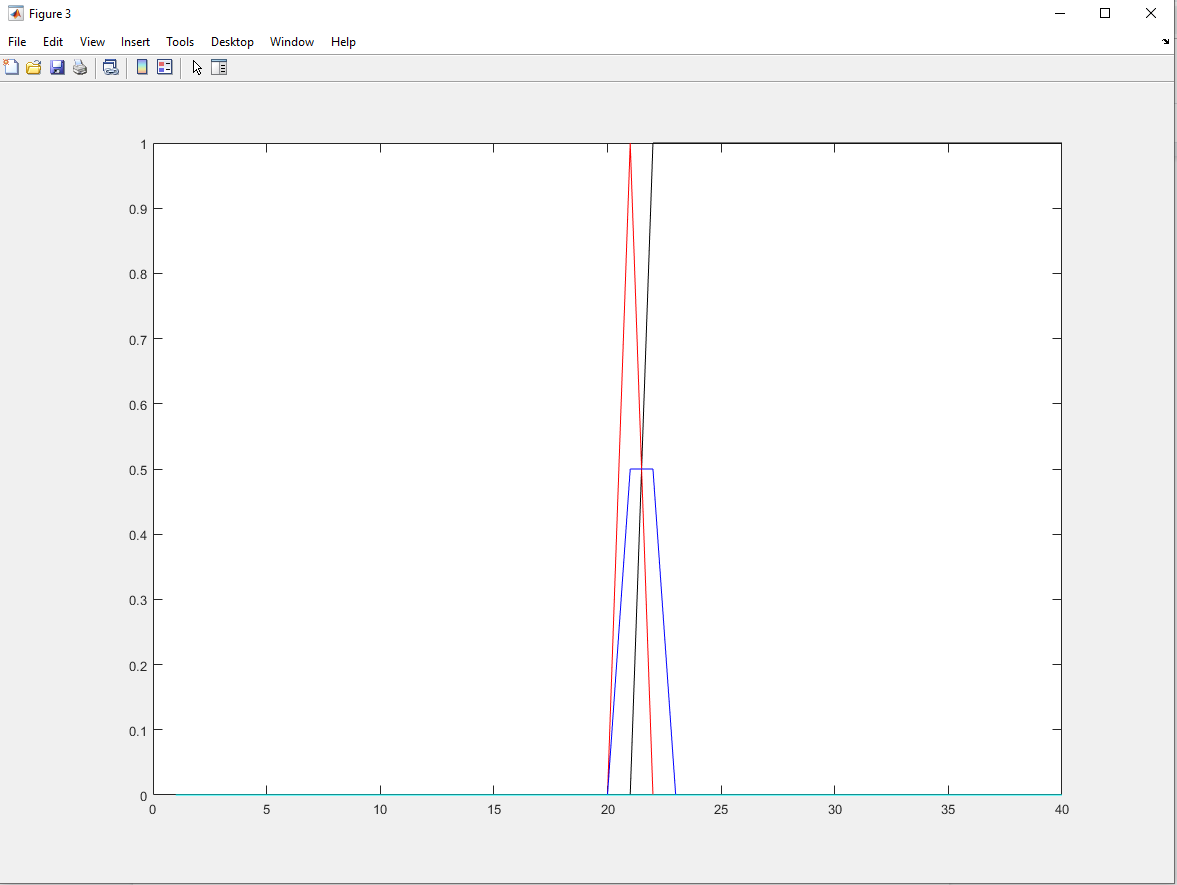
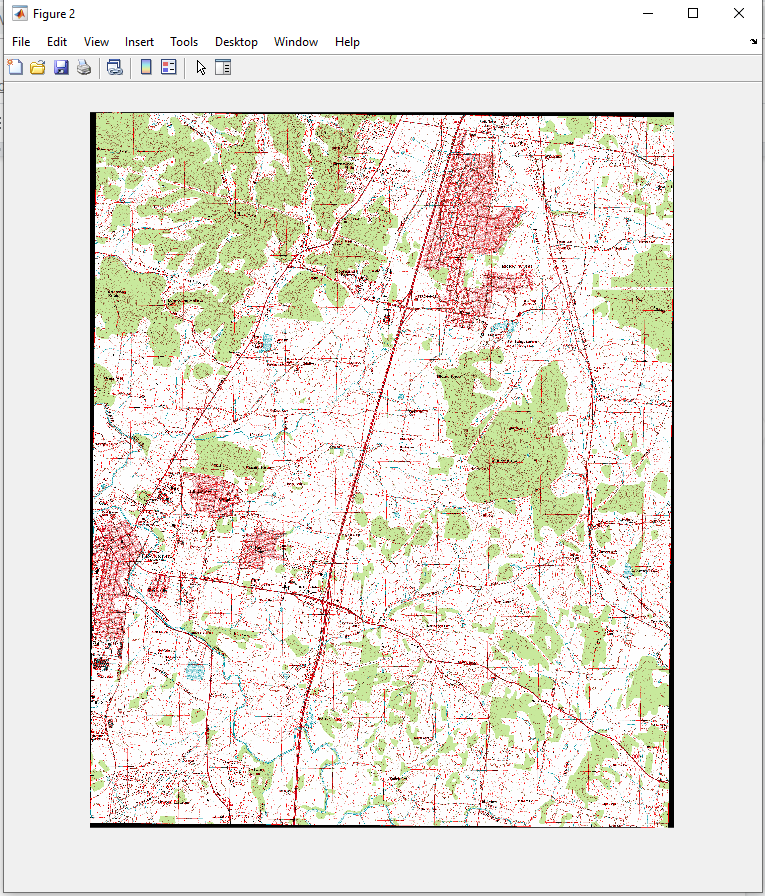
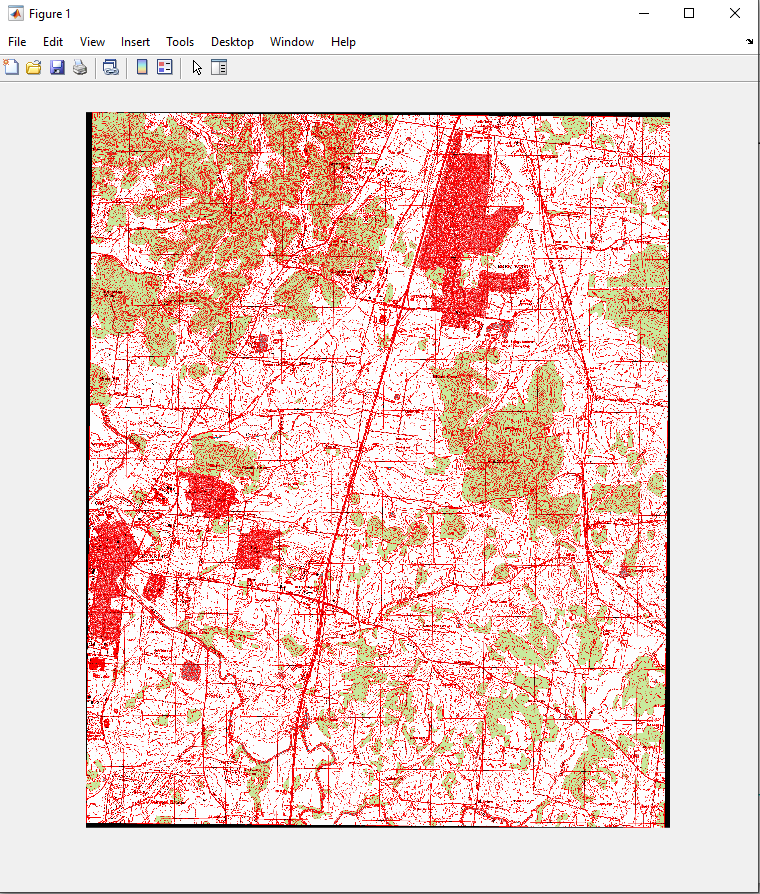
0.1667 0.6667 0.1667

0.6667 -3.3333 0.6667

0.1667 0.6667 0.1667

theta = 3.4726

degrees =198.9631



The reason that the ’log’ method does not line up with the others because the ‘log’ method will have to consider the base. For example, log\_2(n) is base of 2.

Also, since the ‘log’ Finds edges by looking for zero-crossings after filtering I with a Laplacian of Gaussian (LoG) filter. It will use Laplacian of Gaussian (LoG) filter.

**Question 3 Answer:**

0: background or unknown

For this part, we have to define our background first, then make sure if there is a background for that image.

What I did is that I use the Threshold value to determine and define my background index.

Then I can pull the background out.

1: line object

For this part, I have used the houghlines() in matlab since this function houghlines(BW,theta,rho,peaks) extracts line segments in the image BW associated with particular bins in a Hough transform.

2: circular object

The key part to determine if an object is circular is using this equation: 

We can use this estimate each object's area and perimeter. Use these results to form a simple metric indicating the roundness of an object. Then, this metric is equal to 1 only for a circle and it is less than one for any other shape. The discrimination process can be controlled by setting an appropriate threshold.

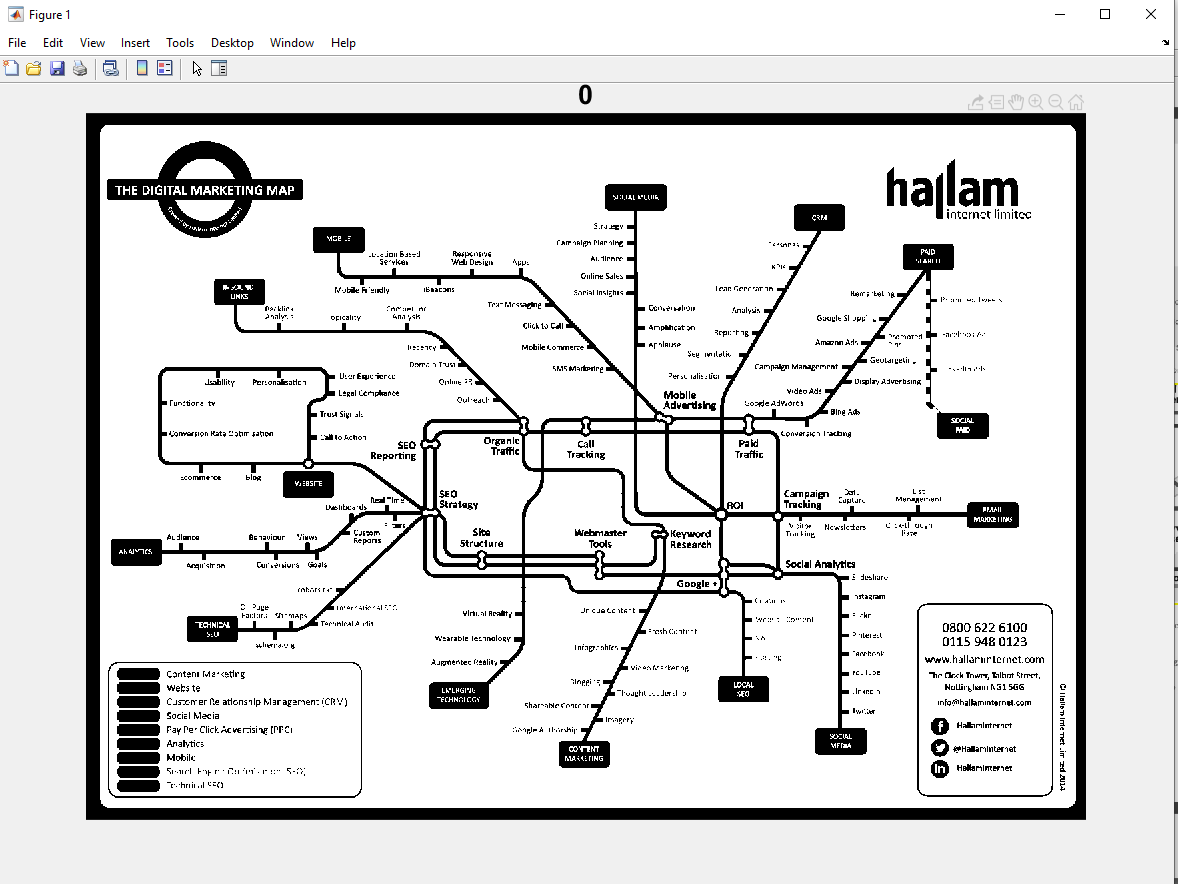
3: text object

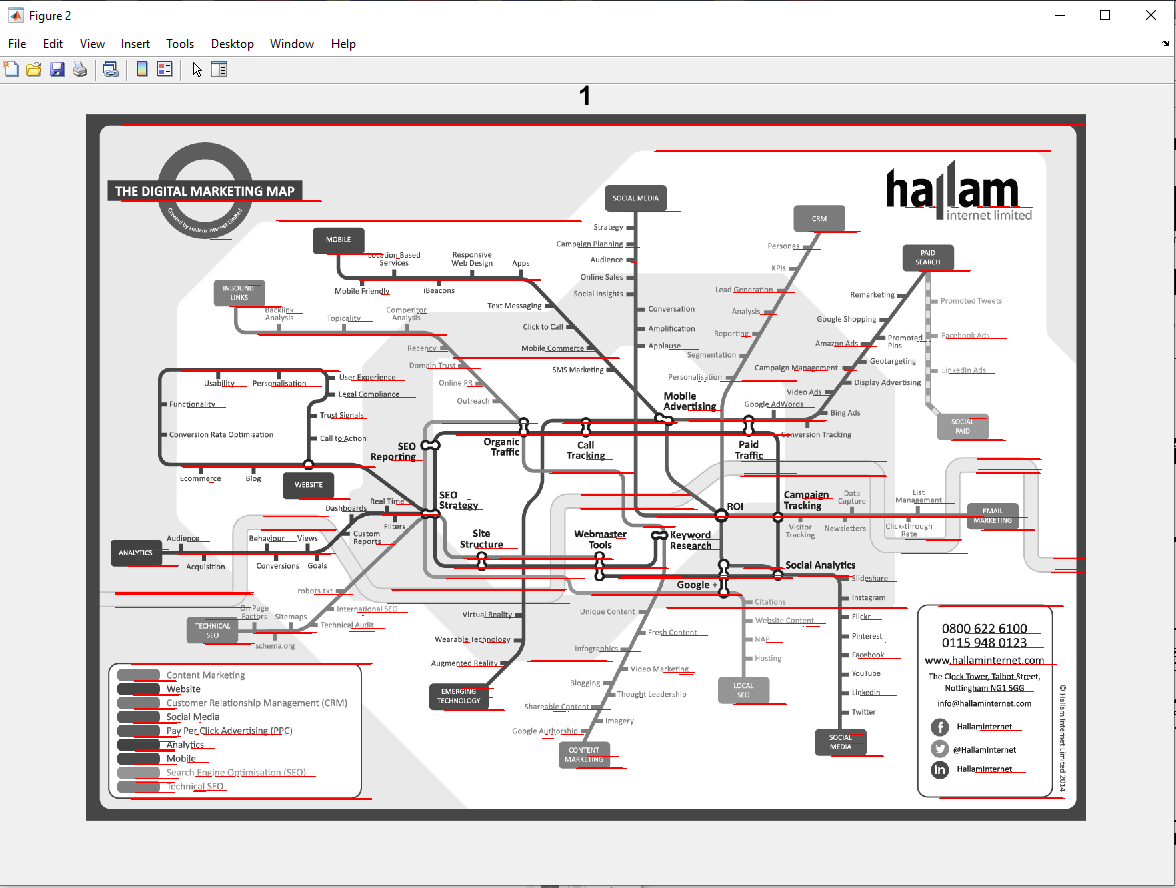
For this part, I have used The MSER feature detector works well for finding text regions [1]. It works well for text because the consistent color and high contrast of text leads to stable intensity profiles.

Use the detectMSERFeatures function to find all the regions within the image and plot these results. Notice that there are many non-text regions detected alongside the text.

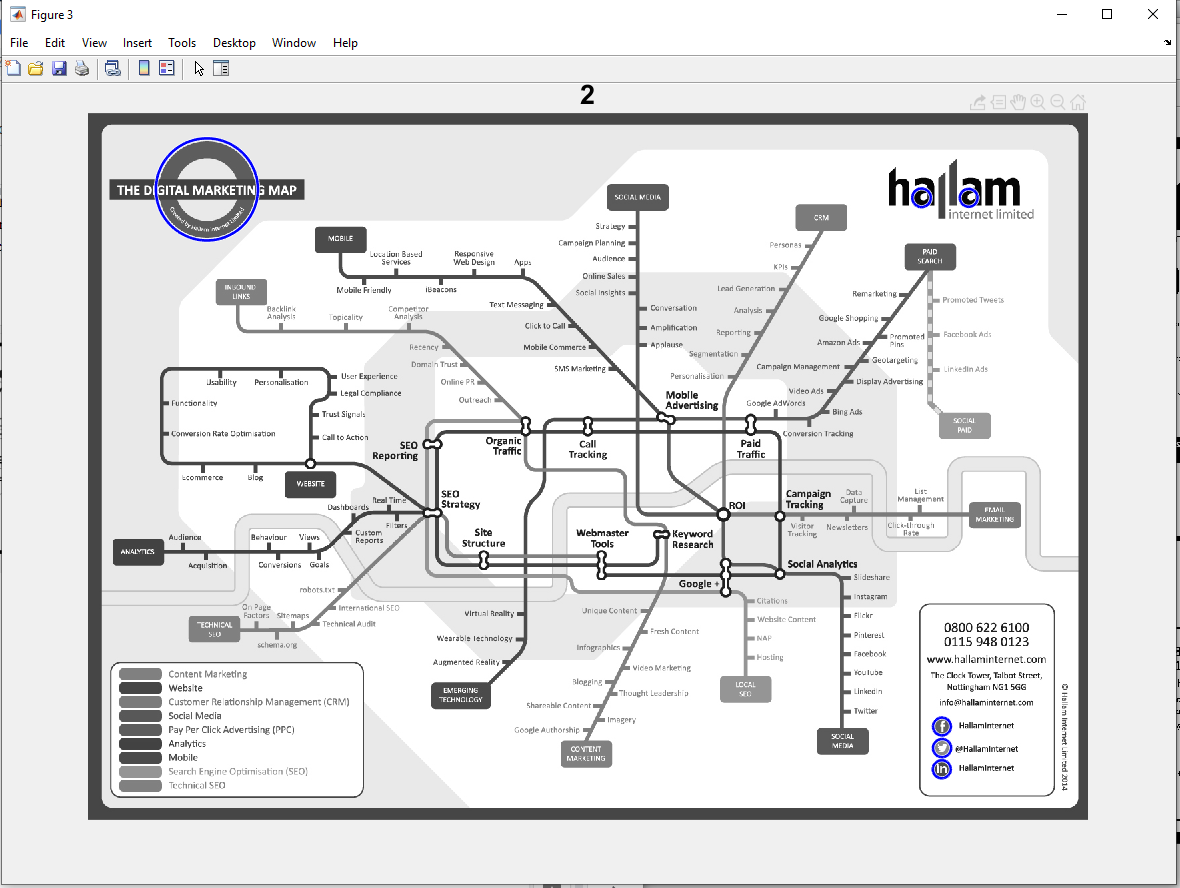
When you run my code, you will want to do this: CS4640\_shapes('text.jpeg'); use the image as input.

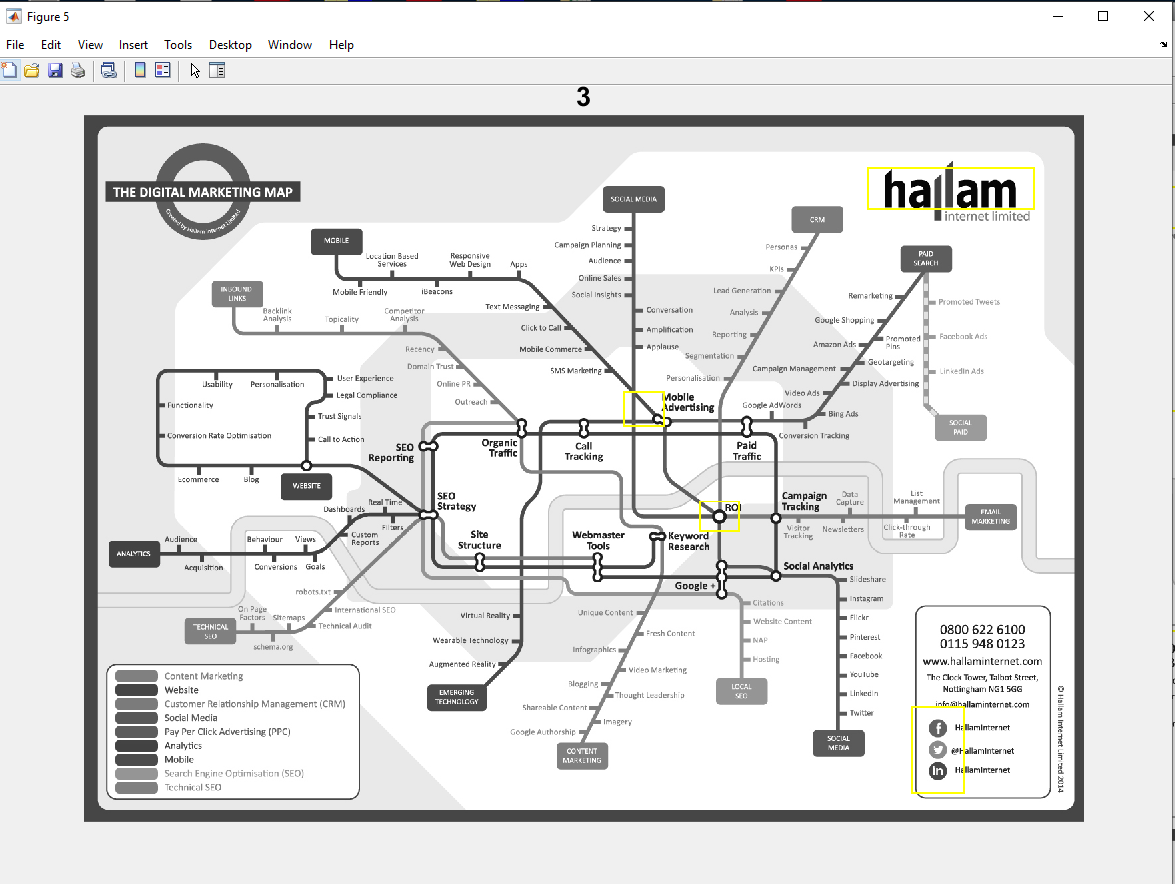
Here are my results:

**Background object**

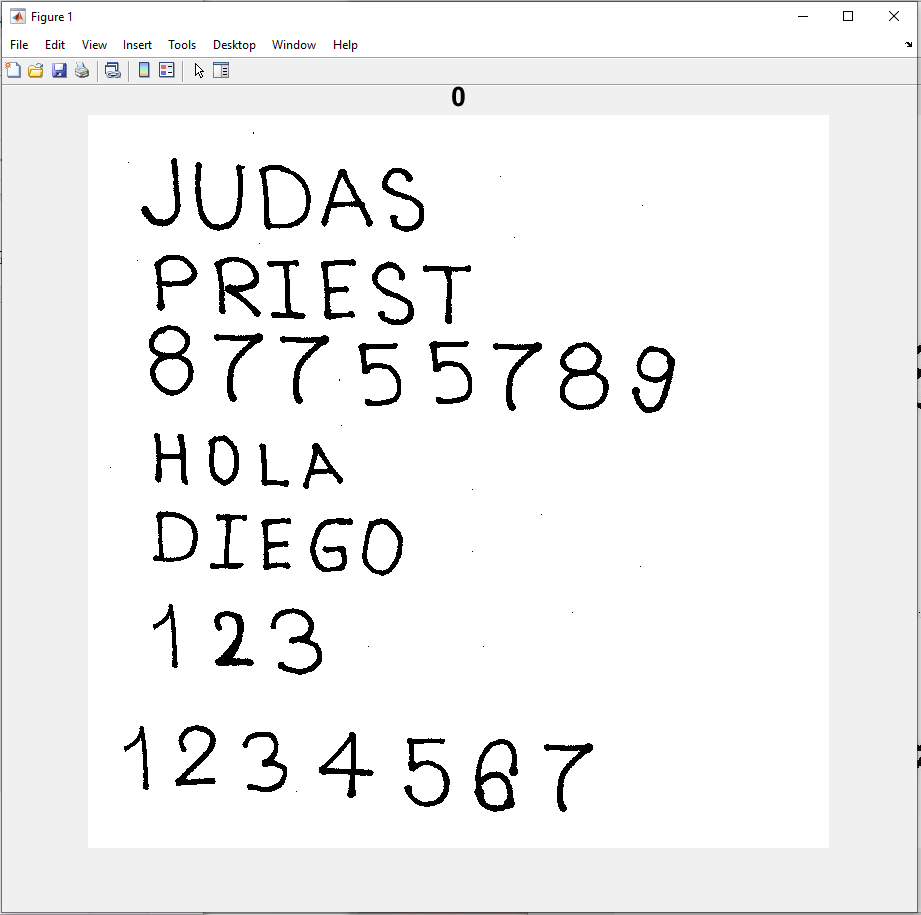


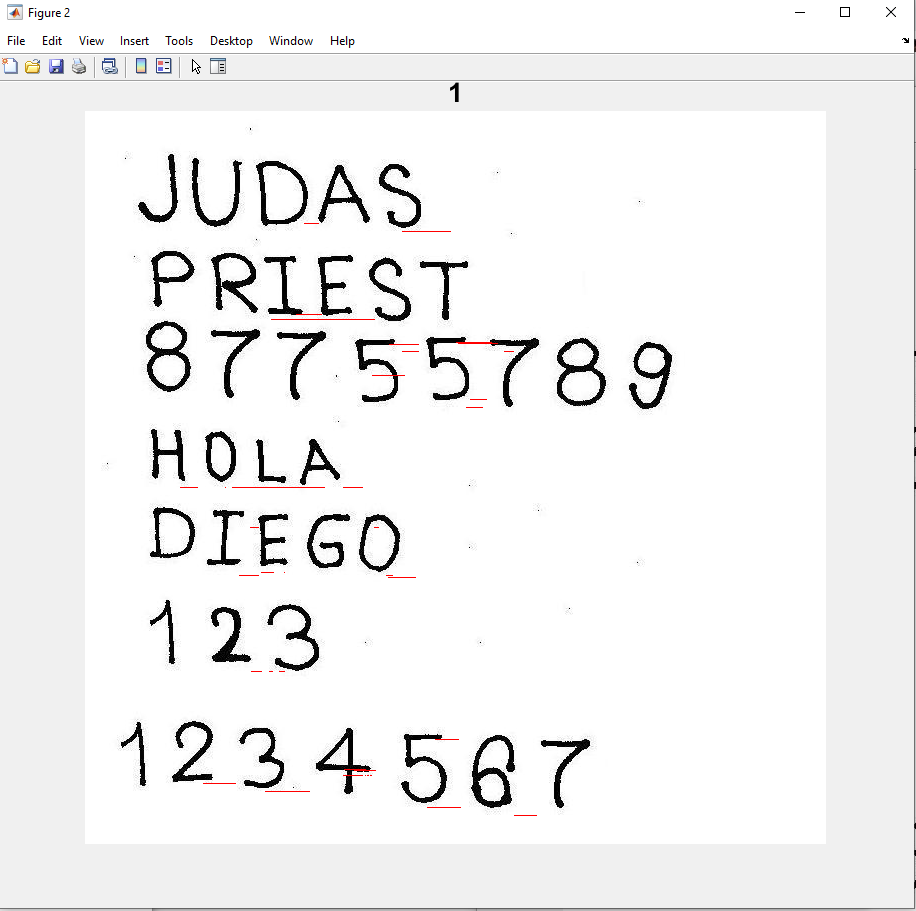
**Line Object**

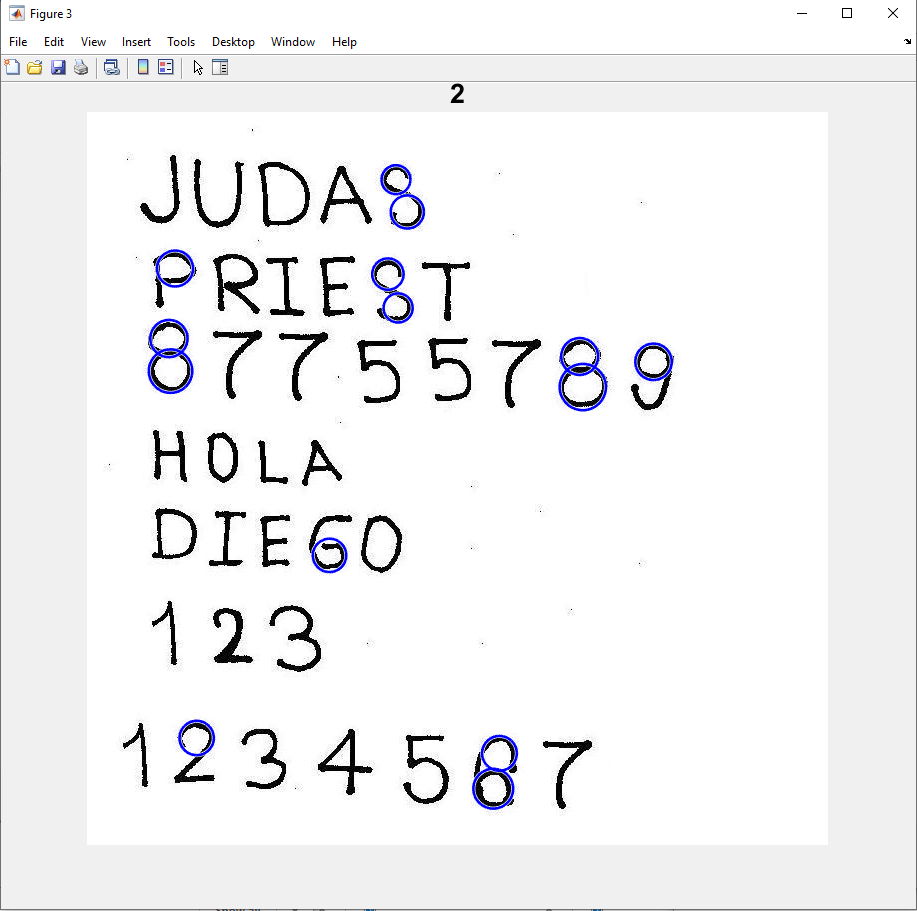
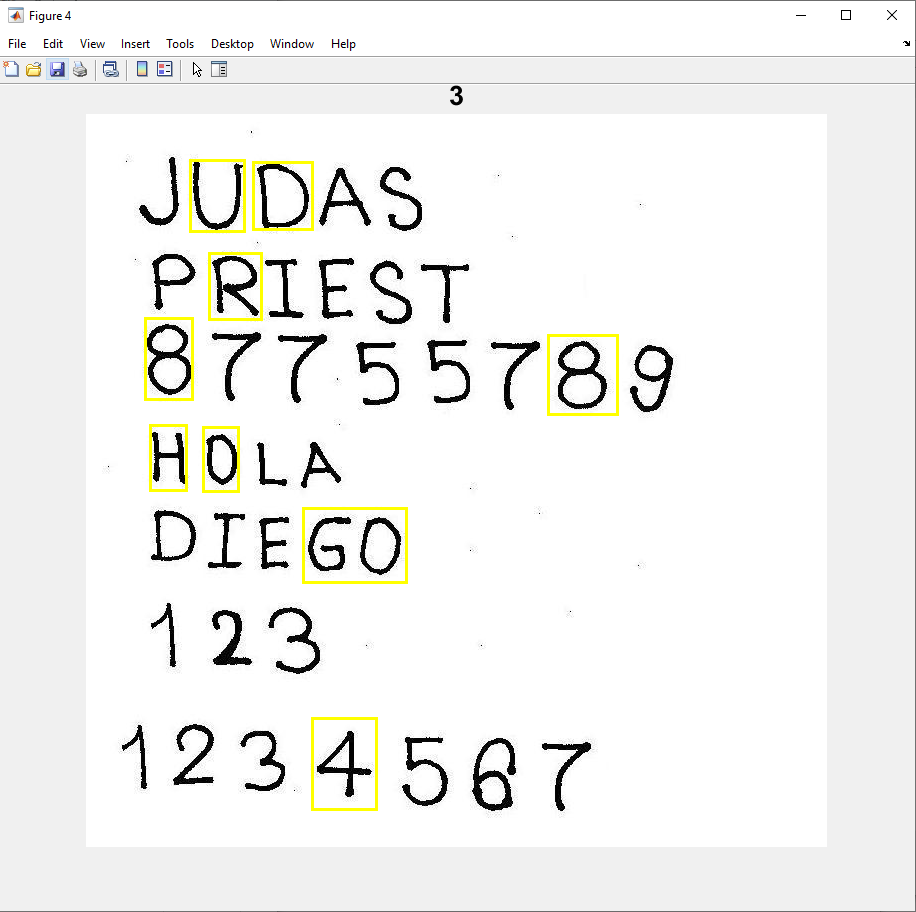
**Circular object**



**Text Object**

**Background Object**

**Line Object**

**Cicular ObjectText Object**