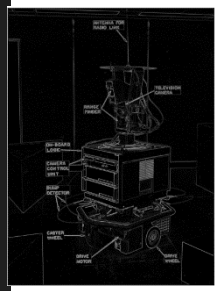
Lab 1 Report: Edge Detection

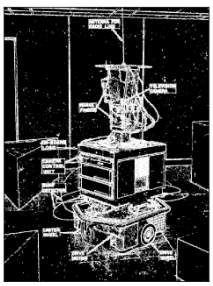
QUESTION 1

```
###Task 1
def magnitude(x,y):
    #Numpy array handles elementwise operations - so this is applied to entire array
    #Pythag function
    mag = np.sqrt(x ** 2  + y ** 2 )
    return mag

#Define shakey_sobel convolution
shakey_sobel_x = scipy.signal.convolve2d(shakey, sobel_x)
shakey_sobel_y = scipy.signal.convolve2d(shakey, sobel_y)

#Find magnitude of both of these arrays - should produce an image showing edges
m = magnitude(shakey_sobel_x, shakey_sobel_y)

#Show the image with different thresholds
show_binary_image(m)
show_binary_image(m>40)
show_binary_image(m>80)
show_binary_image(m>120)
show_binary_image(m>500)
show_binary_image(m>750)
show_binary_image(m>1000)
```

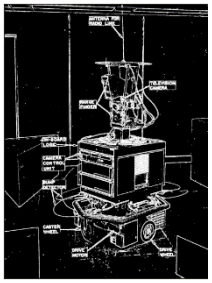
No defined threshold



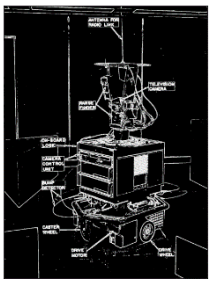
M> 40



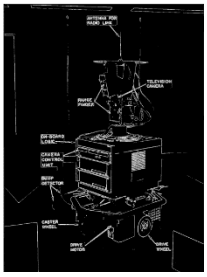
M> 80



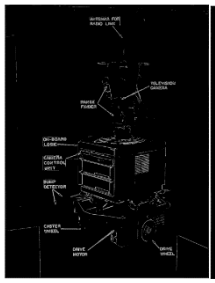
M> 120



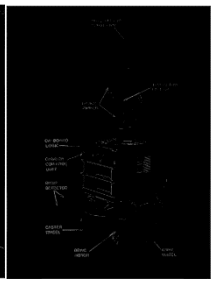
M> 300



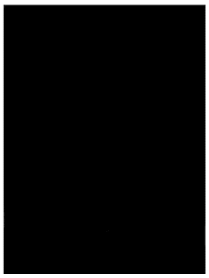
M>500



M>750



M> 1000

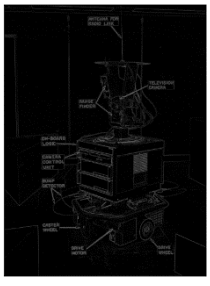


The show_rgb_image() function was not working, so I used the show_binary_image() function instead. The sobel filter applied to the image performs smoothing, and so removes noise in the image. The magnitude function, when applied, creates a binary representation of the image, only showing the intensity of the pixels, and therefore we get an image showing the edges of the original image. When the threshold value is low the resultant image is much brighter, with much thicker lines showing the edges, and has quite a lot of noise in parts of the image (for example the walls in the background of the m=40 image). When the threshold value is increased, the image becomes less bright, with thinner and more defined edges, with an optimal value of m being seemingly between 120 and 300. The reason for this change is, as the threshold value increases. More data is removed from the image representation, as it is not being classified as a 'true edge'. Once m is increased above the optimal value, too much data is rejected, and so detailed edges start to be lost from the image, as seen in the images where m >= 300

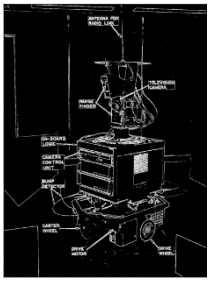
Question 2

Images:

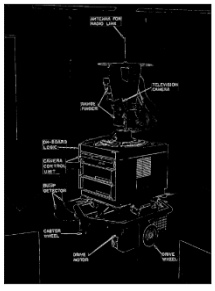
No defined threshold



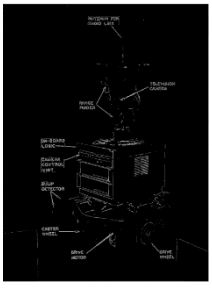
M> 40



M> 80



M> 120



```
###Task 2

#define roberts masks
roberts_x = np.array(
    [[1,0,],
     [0,-1]])

roberts_y = np.array(
    [[0,1],
     [-1,0]])

#initialized shakey array convoluted with roberts array
shakey_roberts_x = scipy.signal.convolve2d(shakey, roberts_x)
shakey_roberts_y = scipy.signal.convolve2d(shakey, roberts_y)

# apply the same magnitude function to shakey_roberts
mRob = magnitude(shakey_roberts_x, shakey_roberts_y)

#vary threshold of results.
show_binary_image(mRob)
show_binary_image(mRob>40)
show_binary_image(mRob>100)
show_binary_image(mRob>300)
show_binary_image(mRob>500)
show_binary_image(mRob>750)
show_binary_image(mRob>1000)
```

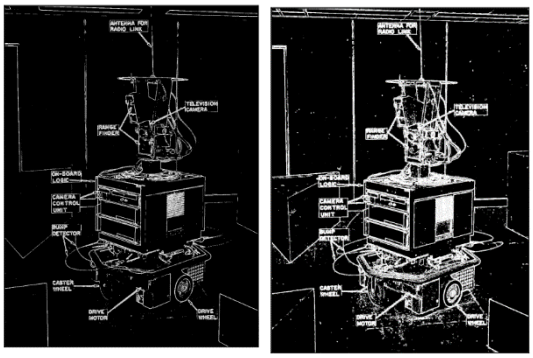
M> 300 | M>500 | M>750 | M> 1000

The difference between Sobel and Roberts is that the image prodcued with Sobel are brighter and have thicker edges for the same threshold value m. This is because the Roberts mask is 2x2, however the sobel mask is 3x3. This means that the Sobel mask takes into account the data of more neighbours when averaging to reduce noise. This means that there is more data included and a higher maximum intensity value. Therefore the optimal threshold value for the Sobel filter is higher. And so at the same threshold value, the sobel filter shows a higher intensity, more detail and so thicker edges than roberts, which has a lower resolution by comparison.

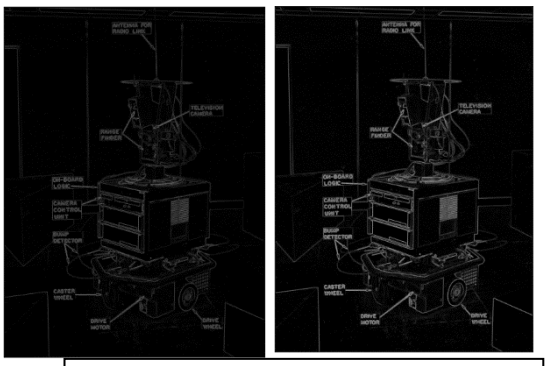
QUESTION 3

```
##Task 3
def magnitudeApprox(x,y):
    magApprox = np.abs(x) + np.abs(y)

    return magApprox
```

One difference between the magnitude and absolute values is that, when using magnitude, for the same threshold value, the edges calculated are brighter and thicker, and, using magnitude, there are noticeably less edges that are missing. This is because calculating the magnitude of the pixels using absolute value is an approximation, compared to the exact values produced by the pythagoras approach. This results in some intensities of pixels being incorrect in the approximation, leading to edges that are left out in the edge detection image. Furthemore, because In the pythagorean calculation, both component values are squared before being summed and square rooted, the more extreme values have a bigger effect(through squaring the values), therefore leading to a brighter image for the same threshold value.



Comparison of Absolute approximation(Left) vs Magnitude(Right) for threshold value m>60



Comparison of Absolute approximation(Left) vs Magnitude(Right) for no defined threshold value