Assignment-1E2-Nachiketh-nxp251

February 21, 2018

Question - Developing an edge detector For this I have used the openCV library functions.

```
In [31]: import cv2
         import numpy as np
         import matplotlib.pyplot as plt
         import sys
```

Edge Detection - A method to perform edge detection on image can be performed locating pixel locations which have higher gradient compared to its neighbours. Or say higher than a threshold.

Sobel Edge Detection - Let's say the image on which the operation needs to happen be I.

1) We calculate two derivatives : a. Horizontal Changes - This is computed by convolving I with kernel G_x with odd size. For example a kernel of size 3 would be - G_x = [[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]] * I b. Vertical Changes - This is computed by convolving I with kernel G_y with odd size. For example a kernel of size 3 would be - G_y = [[-1, -2, -1], [0, 0, 0], [1, 2, 1]] * I

2) At each point of the image we calculate an approximation of the gradient in that point by combining both results above. $G = \sqrt{G_x^2 + G_y^2}$ or you can use the simpler form $G = |G_x| + |G_y|$.

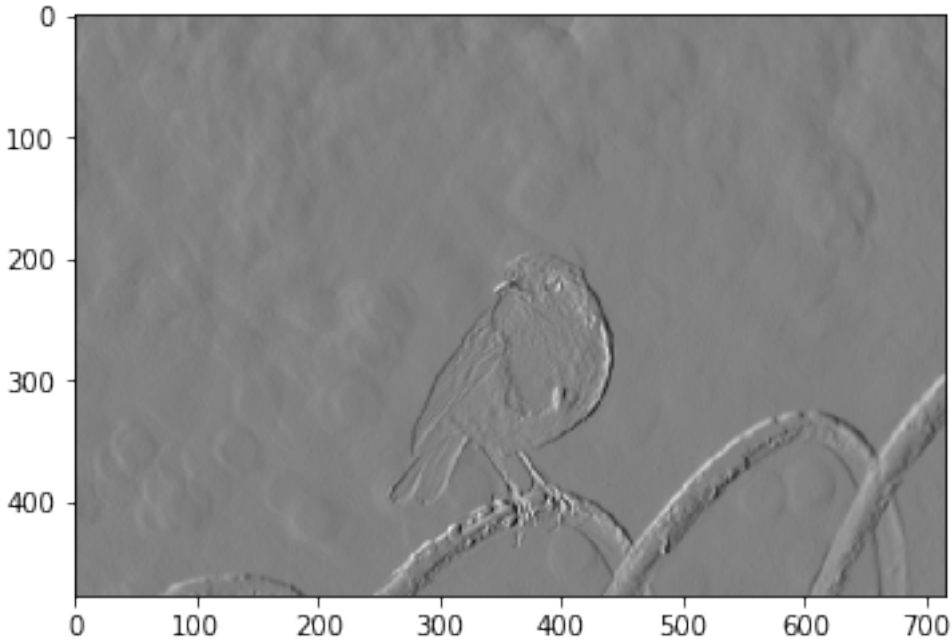
Canny Edge Detection - Canny algorithm aims to satisfy 3 main criteria. 1) Low error rate - Meaning a good detection of only existent edges. 2) Good Localization - The distance between edge pixels detected and real edge pixels have to be minimized. 3) Minimal response - Only one detector response per edge.

Steps - 1) Filter any Noise using a gaussian filter like - K = 1/159 * [[2, 4, 5, 4, 2], [4, 9, 12, 9, 4], [5, 12, 15, 12, 5], [4, 9, 12, 9, 4], [2, 4, 5, 4, 2]] 2) Find the intensity gradient of the image. The procedure is analogus to Sobel. a) Apply a pair of convolution masks is x and y direction. G_x = [[1, 0, -1], [-2, 0, 2], [-1, 0, 1]] G_y = [[-1, -2, -1], [0, 0, 0], [1, 2, 1]] b) Find the gradient strength and direction with: $G = \sqrt{G_x^2 + G_y^2}$ $\theta = \arctan(G_x/G_y)$ 3) Apply non max suppression - This removes the lines that are considered not part of the edge. Only thin lines will remain. 4) Hystersis - Canny uses two thresholds - a) If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge. b) If a pixel value is below the lower threshold, then it is rejected. c) If the pixel gradient is between the two threshold, then it will be accpeted only if it is considered to a pixel that is above the upper threshold.

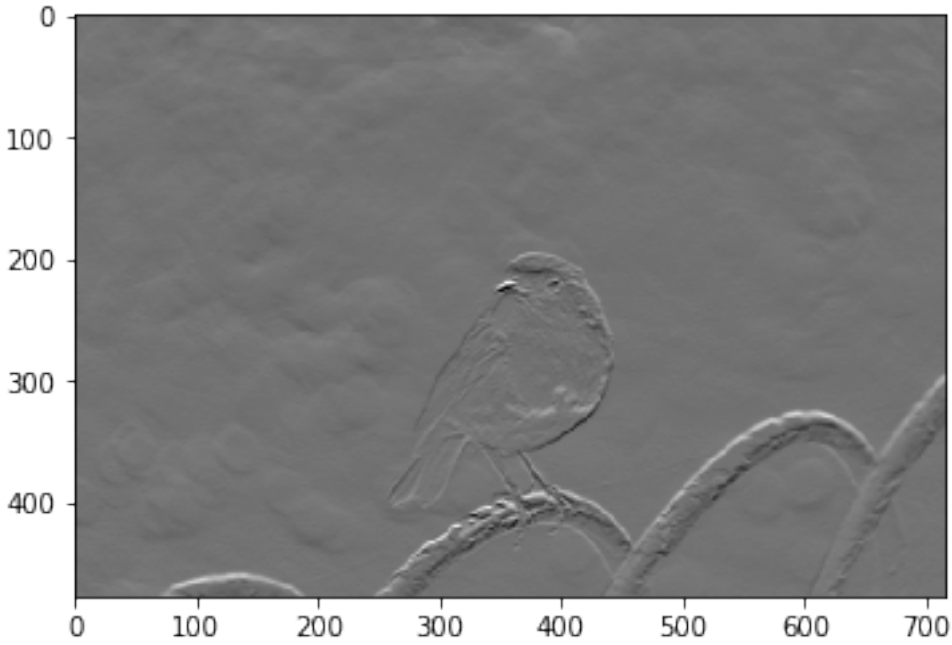
```
In [32]: def sobelEdgeDetector(image):
             sobelx = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=5)
             sobely = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=5)
             return sobelx, sobely
```

```
In [33]: image = cv2.imread('../sample.jpg')
         sobelx, sobely = sobelEdgeDetector(image[:,:,0])
         print('Horizontal Edges')
         plt.imshow(sobelx, cmap='gray')
         plt.show()
         print('Vertical Edges')
         plt.imshow(sobely, cmap='gray')
         plt.show()
```

```
Horizontal Edges
```

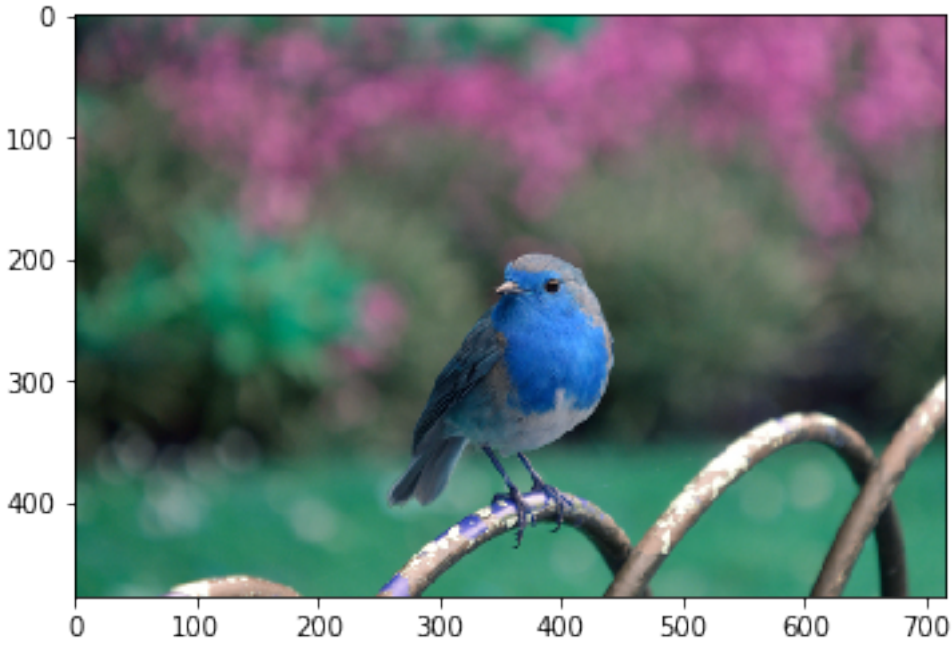


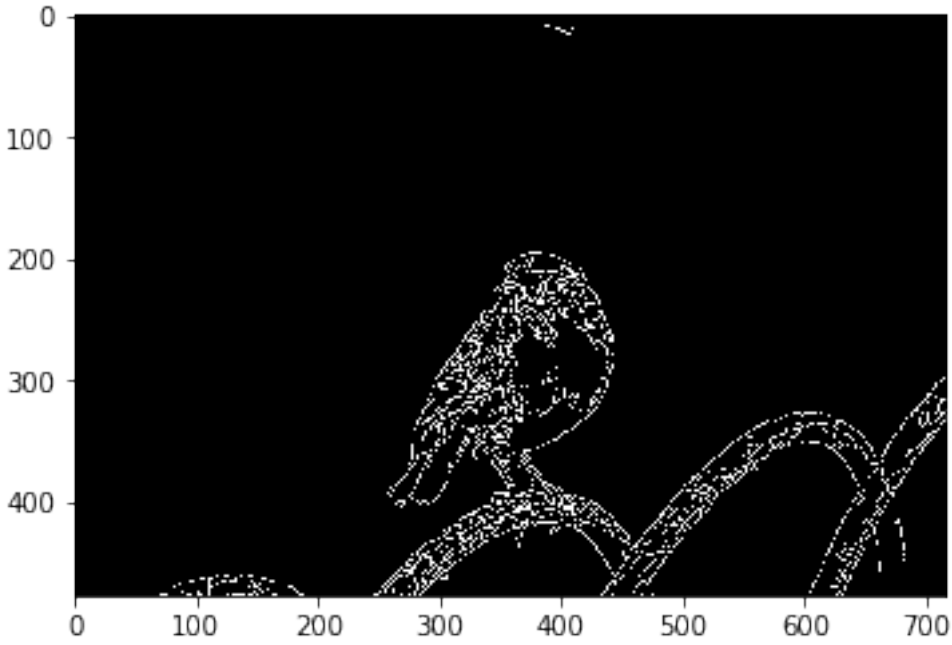

```
Vertical Edges
```




```
In [34]: def cannyEdgeDetector(image):
             canny = cv2.Canny(image, 20, 170)
             return canny
```

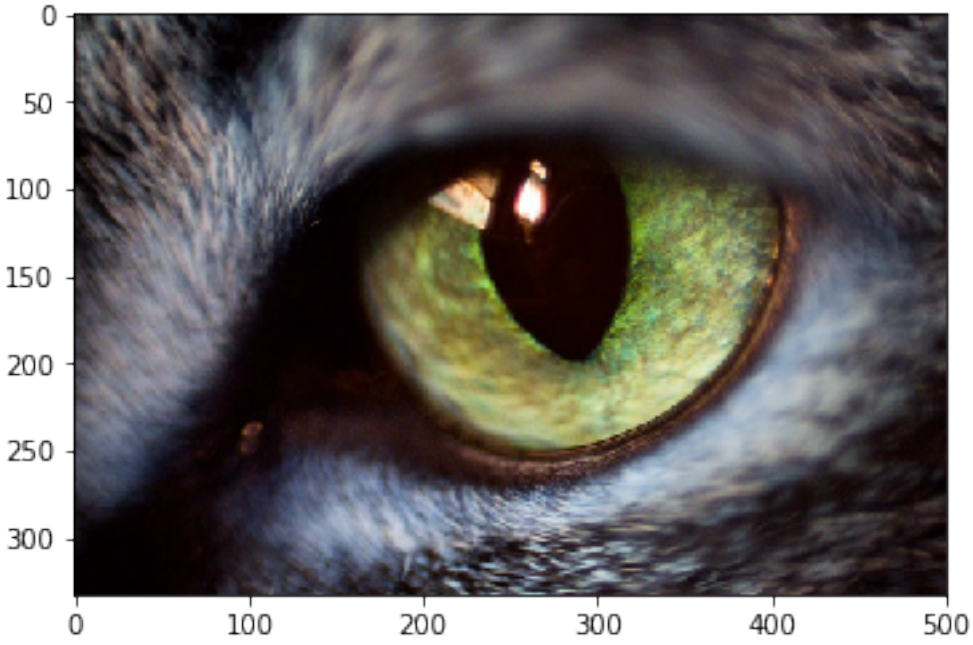
```
In [44]: image = cv2.imread('../sample.jpg')
         plt.imshow(image)
         plt.show()
         canny = cannyEdgeDetector(image)
         plt.imshow(canny, cmap='gray')
         plt.show()
```

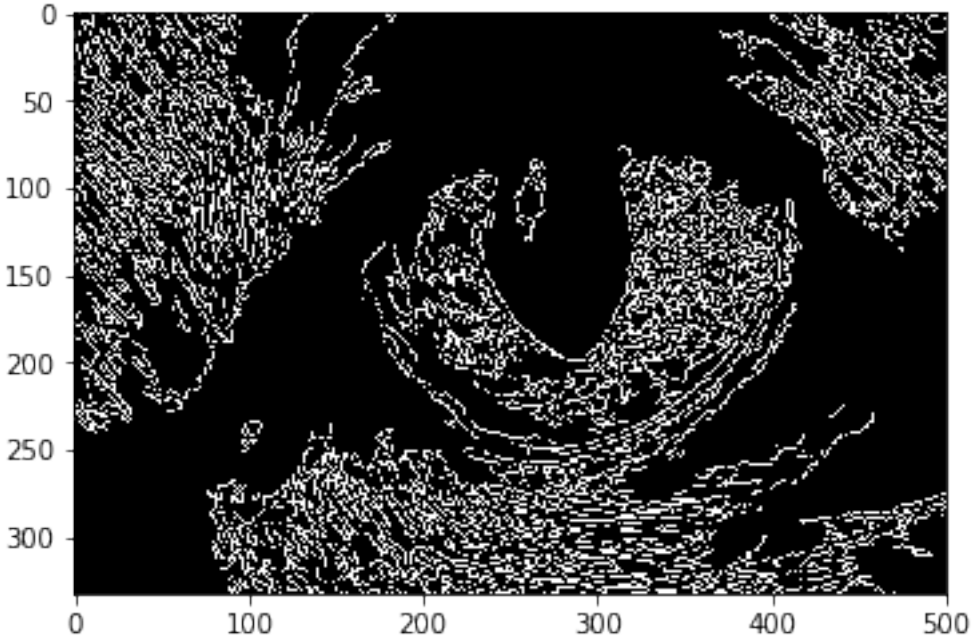





```
In [45]: image = cv2.imread('../pic.jpg')
         plt.imshow(image)
         plt.show()
```

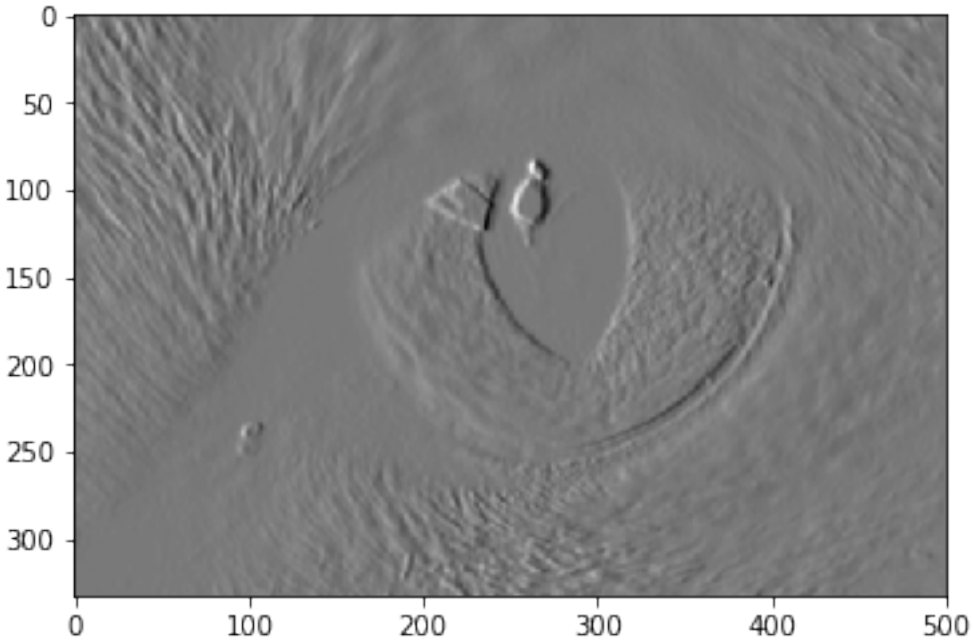
```
canny = cannyEdgeDetector(image)
plt.imshow(canny, cmap='gray')
plt.show()
```



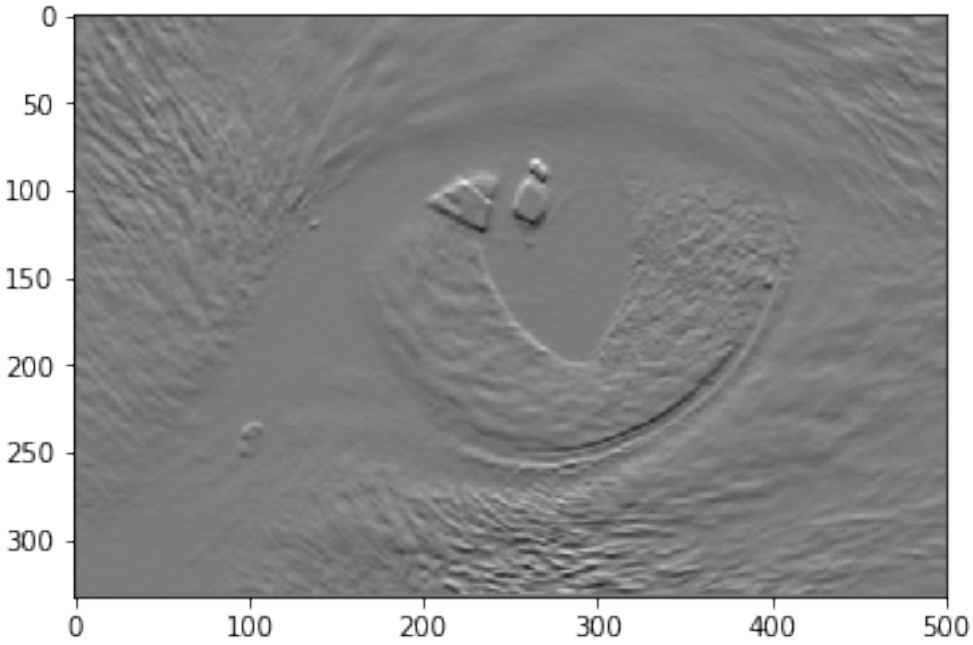


```
In [41]: image = cv2.imread('../pic.jpg')
         sobelx, sobely = sobelEdgeDetector(image[:,:,0])
         print('Horizontal Edges')
         plt.imshow(sobelx, cmap='gray')
         plt.show()
         print('Vertical Edges')
         plt.imshow(sobely, cmap='gray')
         plt.show()
```

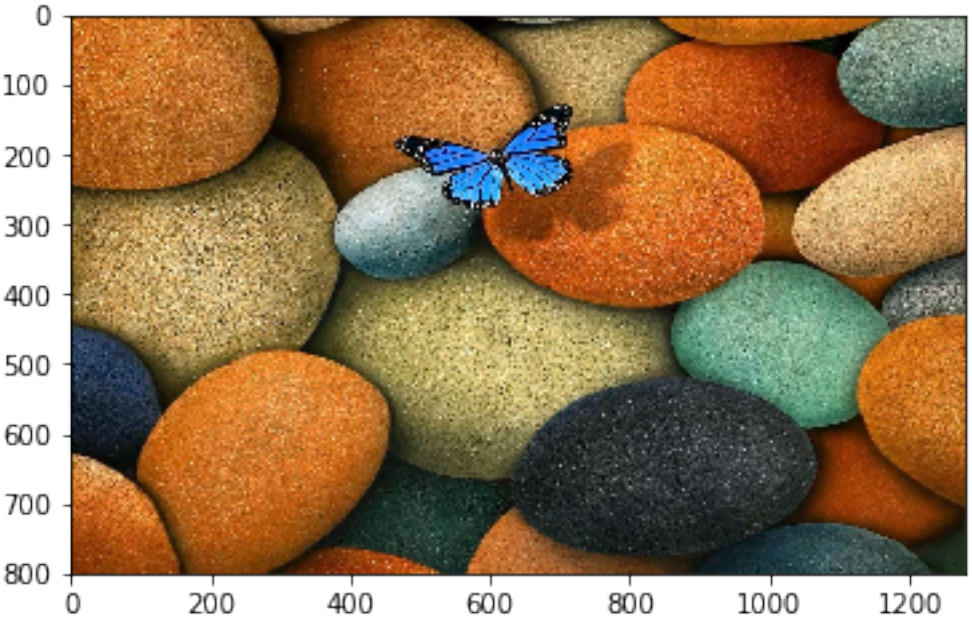
```
Horizontal Edges
```

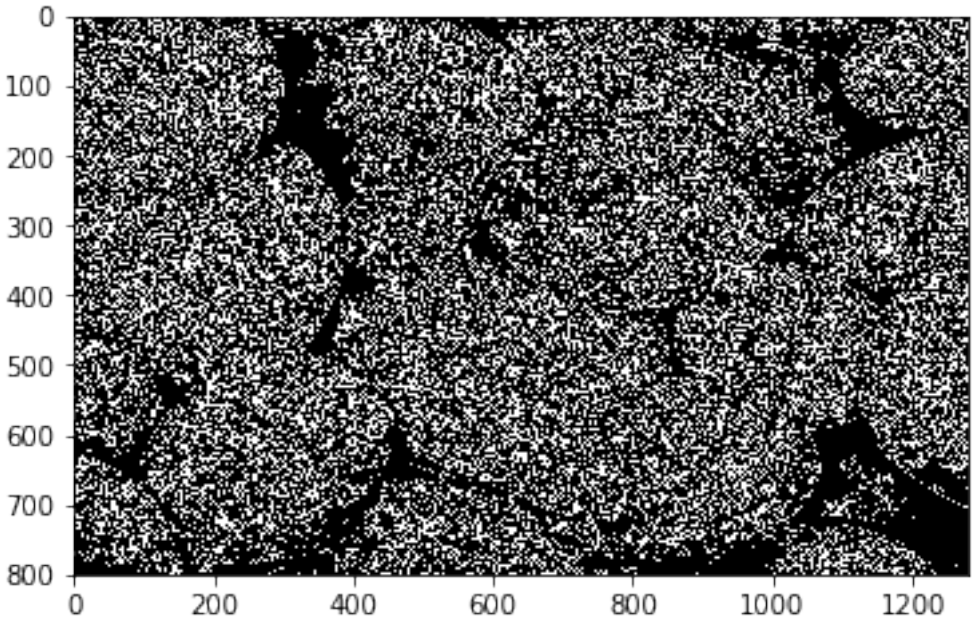



```
Vertical Edges
```



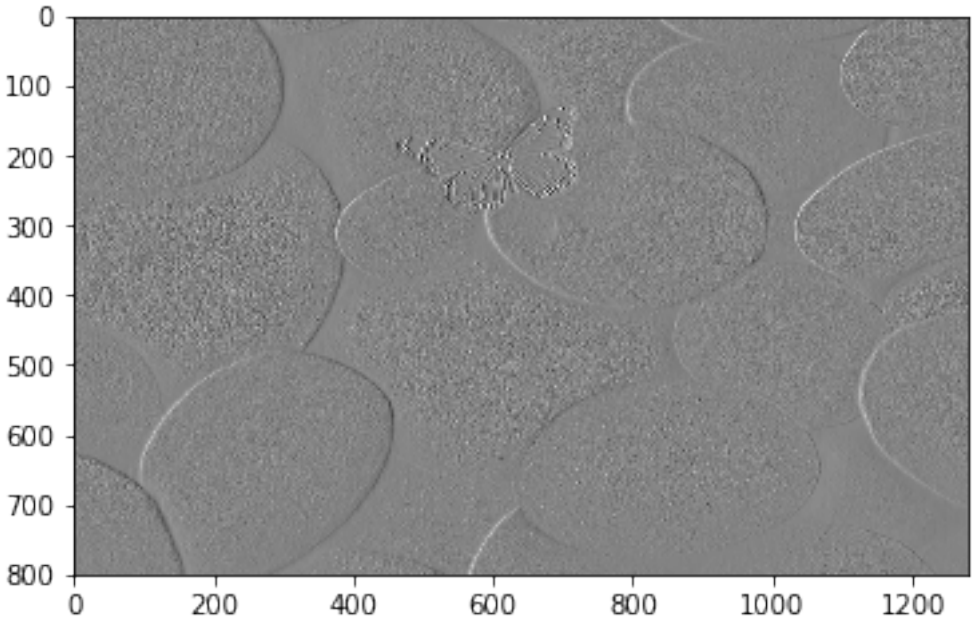
In [47]:
```
image = cv2.imread('../pic2.jpg')
plt.imshow(image)
plt.show()
canny = cannyEdgeDetector(image)
plt.imshow(canny, cmap='gray')
plt.show()
```






```
In [43]: image = cv2.imread('../pic2.jpg')
         sobelx, sobely = sobelEdgeDetector(image[:,:,0])
         print('Horizontal Edges')
         plt.imshow(sobelx, cmap='gray')
         plt.show()
         print('Vertical Edges')
         plt.imshow(sobely, cmap='gray')
         plt.show()
```

```
Horizontal Edges
```



Vertical Edges