

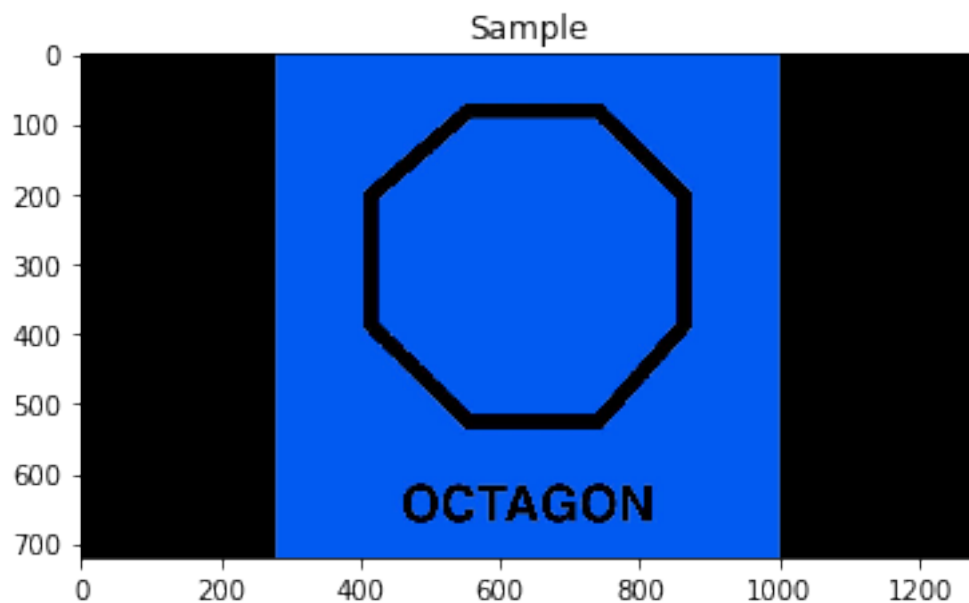
Edge Detection

April 28, 2019

0.0.1 Edge Detection

```
In [1]: import cv2
import numpy as np
from matplotlib import pyplot as plt
```

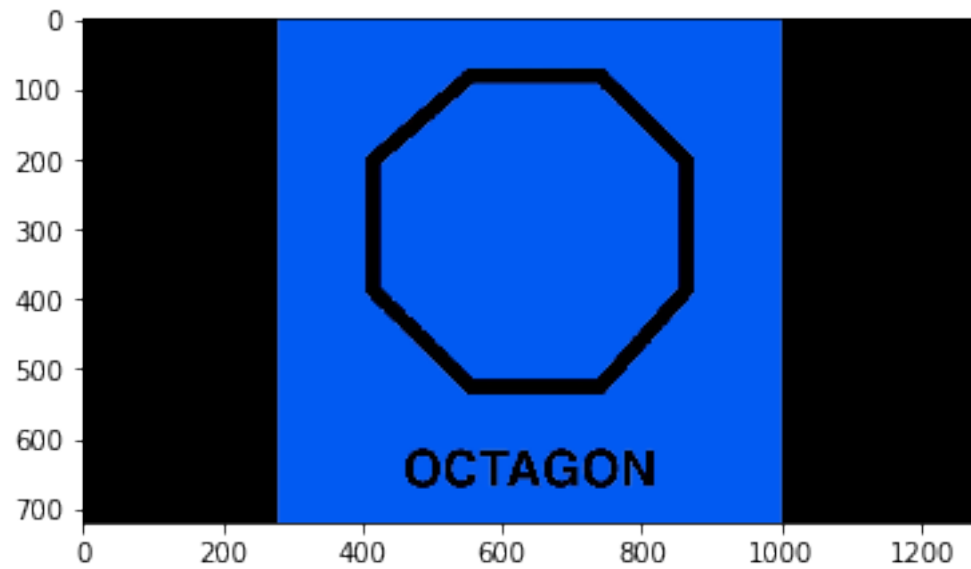
```
In [2]: img = cv2.imread('sample5.jpg')
plt.imshow(img)
plt.title('Sample')
plt.show()
```



```
In [3]: blur = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)
```

```
In [4]: plt.imshow(blur, cmap='gray')
```

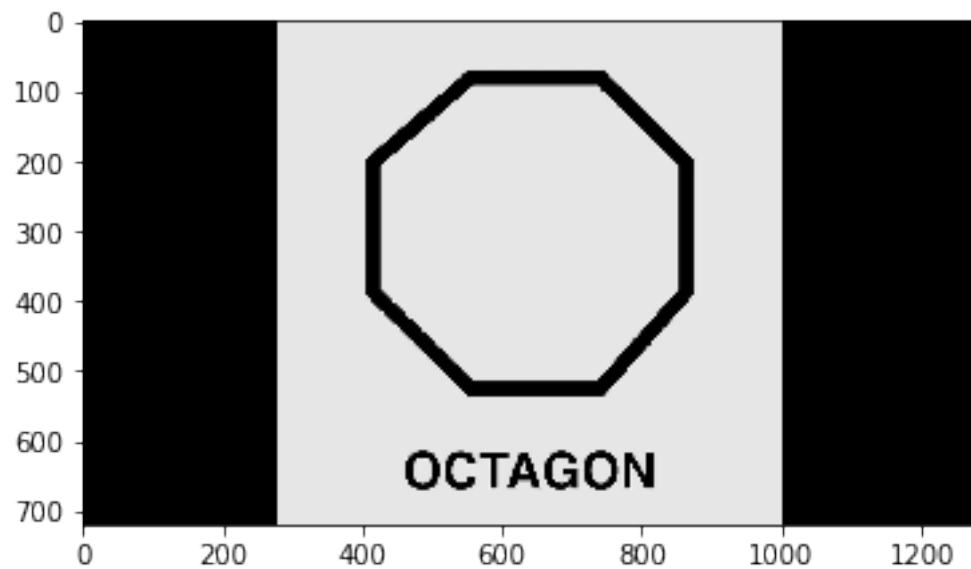
```
Out[4]: <matplotlib.image.AxesImage at 0x11a1daf28>
```



```
In [5]: gray = cv2.cvtColor(blur, cv2.COLOR_BGR2GRAY)
```

```
In [6]: plt.imshow(gray, cmap='gray')
```

```
Out[6]: <matplotlib.image.AxesImage at 0x1bf20e80>
```



```
In [7]: print(gray.shape)
```

(720, 1280)

```
In [8]: def pad(img,shp):
        p=np.zeros((shp[0]+2,shp[1]+2))
        p[1:-1,1:-1]=np.copy(img)
        p[0,1:-1],p[-1,1:-1]=img[0],img[-1]
        p[1:-1,0],p[1:-1,-1]=img[:,0],img[:,-1]
        p[0,0],p[0,-1]=img[0,0],img[0,-1]
        p[-1,0],p[-1,-1]=img[-1,0],img[-1,-1]
        return p

In [9]: def sobel_filter(img):
        sabel_x = np.array([[ -1,0,1],[ -2,0,2],[ -1,0,1]])
        sabel_y = np.array([[ -1,-2,-1],[ 0,0,0],[ 1,2,1]])
        shp = img.shape
        shpm=(3,3)
        padded_img=pad(img,shp)
        grad_matrix=np.zeros(shp)
        out=np.zeros(shp)
        exp = np.zeros(shp)
        for i in range(shp[0]):
            for j in range(shp[1]):
                g_x=np.multiply(padded_img[i:i+shpm[0],j:j+shpm[1]],sabel_x).sum()
                g_y=np.multiply(padded_img[i:i+shpm[0],j:j+shpm[1]],sabel_y).sum()
                if g_y!=0 or g_x!=0:
                    if g_x==0:
                        rad=np.arctan2(g_y,g_x)
                    else:
                        rad=np.arctan2(g_y,g_x)
                    deg=rad*(180/np.pi)
                    rad_rev = deg*(np.pi/180)
                    #print(rad*(180/np.pi),end=" ")
                    #print(rad_rev)
                    grad_matrix[i][j]=deg
                    if grad_matrix[i][j]<0:
                        exp[i][j] = grad_matrix[i][j]
                    out[i][j] = np.sqrt(np.square(g_x)+np.square(g_y))
                else:
                    out[i,j]=255
                    grad_matrix[i][j]=255
        out=np.array(out, dtype = np.uint8)
        grad_matrix=np.array(grad_matrix,dtype= np.uint8)

        return out,grad_matrix,exp

In [10]: def prewitt_filter(img):
        prewitt_x = np.array([[ -1,0,1],[ -1,0,1],[ -1,0,1]])
```

```

prewitt_y = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
shp = img.shape
shpm=(3,3)
padded_img=pad(img,shp)
grad_matrix=np.zeros(shp)
out=np.zeros(shp)
exp = np.zeros(shp)
for i in range(shp[0]):
    for j in range(shp[1]):
        g_x=np.multiply(padded_img[i:i+shpm[0],j:j+shpm[1]],prewitt_x).sum()
        g_y=np.multiply(padded_img[i:i+shpm[0],j:j+shpm[1]],prewitt_y).sum()
        if g_y!=0 or g_x!=0:
            if g_x==0:
                rad=np.arctan2(g_y,g_x)
            else:
                rad=np.arctan2(g_y,g_x)
            deg=rad*(180/np.pi)
            rad_rev = deg*(np.pi/180)
            #print(rad*(180/np.pi),end=" ")
            #print(rad_rev)
            grad_matrix[i][j]=deg
            if grad_matrix[i][j]<0:
                exp[i][j] = grad_matrix[i][j]
                print(exp[i][j])
            out[i][j] = np.sqrt(np.square(g_x)+np.square(g_y))
        else:
            out[i,j]=255
            grad_matrix[i][j]=255
out=np.array(out, dtype = np.uint8)

return out,grad_matrix,exp

```

In [11]: output, grad_matrix,exp = sobel_filter(gray)

In []:

In [12]: plt.imshow(exp,cmap='gray')

Out[12]: <matplotlib.image.AxesImage at 0x11c227dd8>

