Edge Detection in Images

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement - Implement edge detection first using the canny edge detection. Then apply simple thresholding and also Adaptive/OTSU thresholding using OpenCV

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

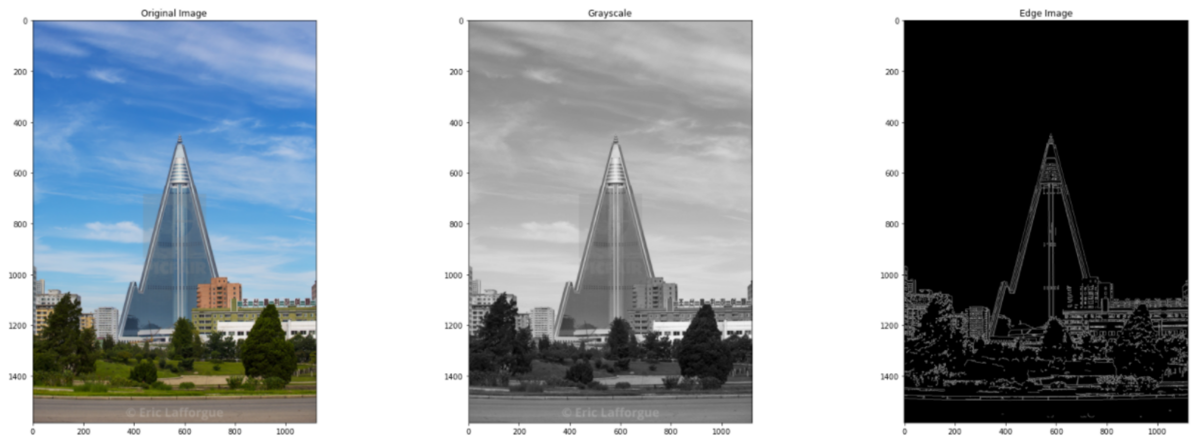
The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

1. Importing necessary libraries -

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

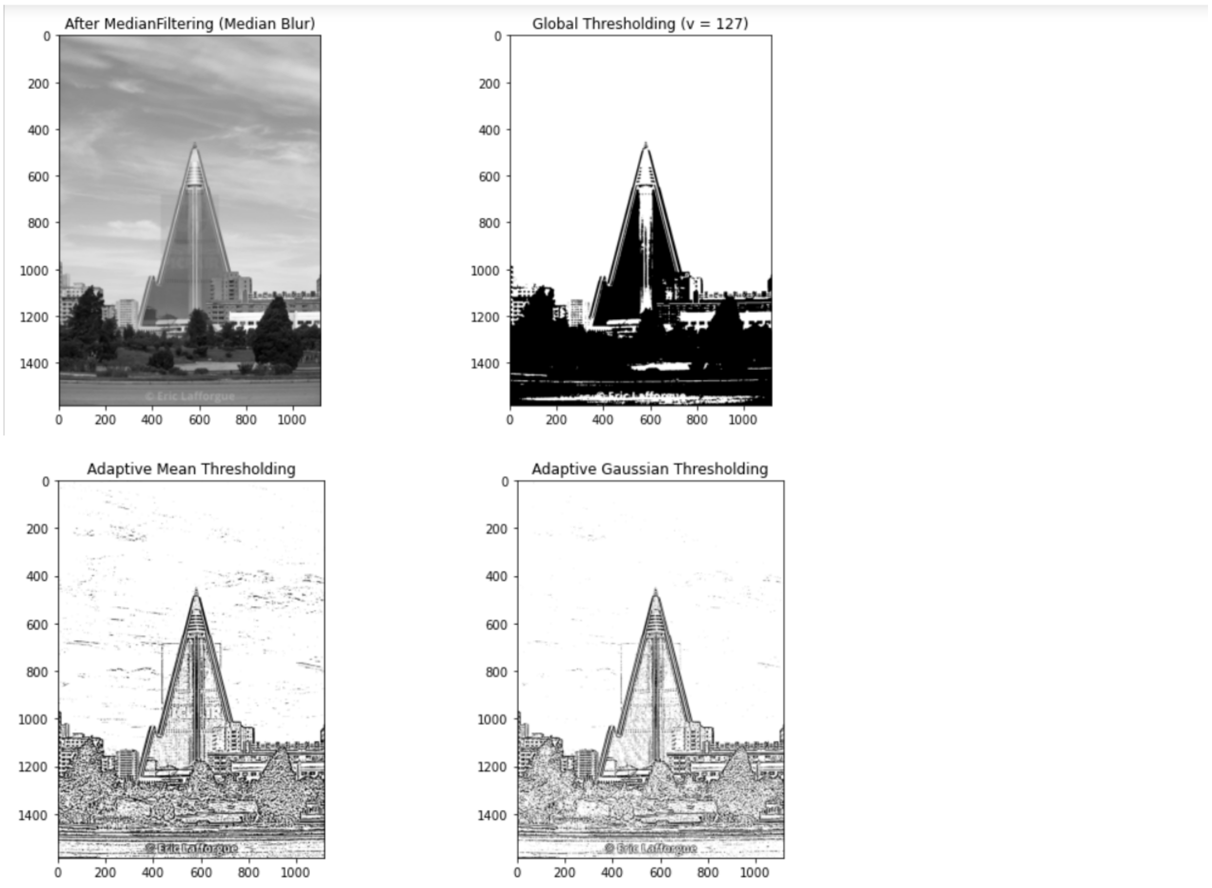
2. Converting to grayscale and applying canny edge detection

```
org_img = cv2.imread('image.jpg')

img = cv2.cvtColor(org_img, cv2.COLOR_BGR2RGB)

grayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

edges = cv2.Canny(grayscale,100,200)

plt.figure(figsize=(30,10))
plt.subplot(1,3,1)
plt.imshow(img)
plt.title('Original Image')

plt.subplot(1,3,2)
plt.imshow(grayscale, cmap = 'gray')
plt.title('Grayscale')

plt.subplot(1,3,3)
plt.imshow(edges, cmap='gray')
plt.title('Edge Image')

plt.show()
```



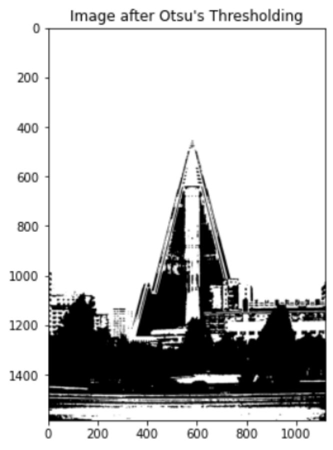
3. Removing noise using median blur and applying adaptive thresholding-

```
img = cv2.medianBlur(grayscale,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,9,2)

th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,9,2)

titles = ['After MedianFiltering (Median Blur)','Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']

images = [img, th1, th2, th3]

plt.figure(figsize=(12,10))
for i in range(4):
    plt.subplot(2,2,i+1)
    plt.imshow(images[i],'gray')
    plt.title(titles[i])


plt.tight_layout()
plt.show()
```


After MedianFiltering (Median Blur)
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding
© Eric Lafforgue

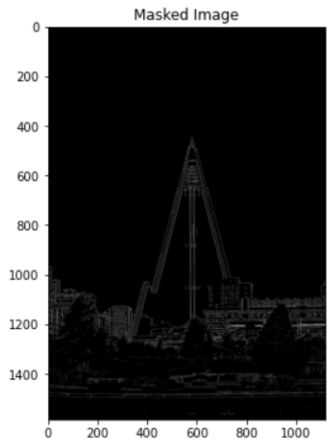
4. Otsu Thresholding-

```
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

plt.figure(figsize=(6,6))
plt.imshow(th2,'gray')
plt.title("Image after Otsu's Thresholding")
plt.show()
```



5. Edge detection results as a mask to give colour to all the edges -

```
mask = np.zeros(img.shape[:2], dtype="uint8")
masked = cv2.bitwise_and(grayscale, edges)
plt.figure(figsize=(6,6))
plt.imshow(masked, cmap='gray')
plt.title('Masked Image')
plt.show()
```


Masked Image