Assignment Project Report

Association Rule Mining: Market Basket Analysis

Name : Meghna Maan

Course : AI & ML (Batch 4)

Given Question

Using OpenCV, first convert any image with varying High condition to a grayscale image. Now implement edge detection first using the canny edge detection. Then apply simple thresholding and also Adaptive/OTSU thresholding using OpenCV to see the working of each of these methods. Once you obtain good results, use the obtained edge detection result as a mask to give color to all the edges (if edges use the color from the original image, else leave it black only).

Prerequisites

1. Software:

   Python 3

2. Tools:

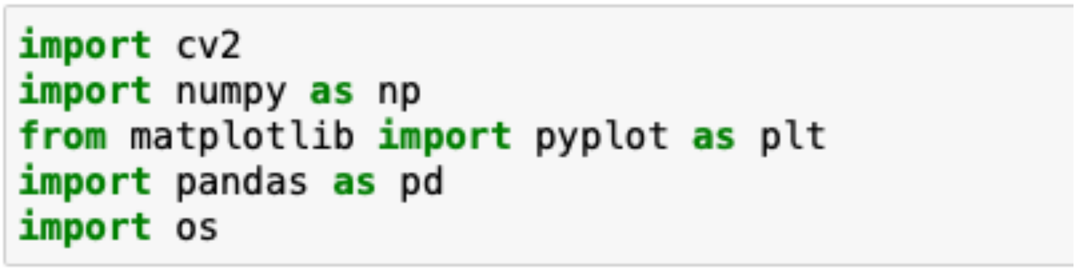
   - Numpy
   - Pandas
   - Matplotlib
   - Opencv
   - Pandas
   - OS

Methods Used

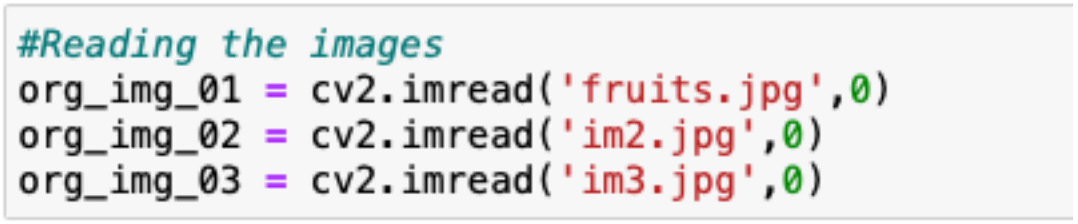
Edges define the boundaries between different regions in an image, which helps in matching the pattern, segment, and recognize an object. In simple thresholding, the threshold value is global, which is prone to fail in many cases. Adaptive thresholding is a modified method where the threshold value is calculated for each pixel based on a smaller region around it. Therefore, there will be different threshold values for different regions which gives better results for images with varying illumination.

Implementation

1. Loading Libraries and Dataset



```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
import os
```

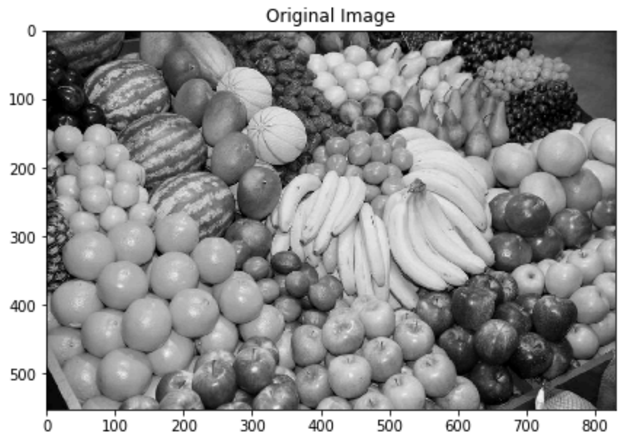


```
#Reading the images
org_img_01 = cv2.imread('fruits.jpg',0)
org_img_02 = cv2.imread('im2.jpg',0)
org_img_03 = cv2.imread('im3.jpg',0)
```

2. Getting image for image detection

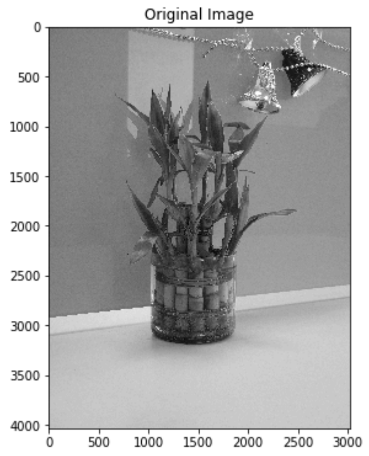
```
print(type(org_img_01))
plt.figure(figsize=(7,6))
plt.title("Original Image")
plt.imshow(org_img_01,'gray')
```

```
<class 'numpy.ndarray'>

<matplotlib.image.AxesImage at 0x23714c0ac88>
```



```
print(type(org_img_01))
plt.figure(figsize=(7,6))
plt.title("Original Image")
plt.imshow(org_img_03,'gray')
```

```
<class 'numpy.ndarray'>

<matplotlib.image.AxesImage at 0x237168d6108>
```

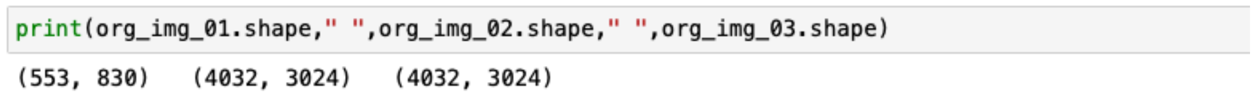


3.

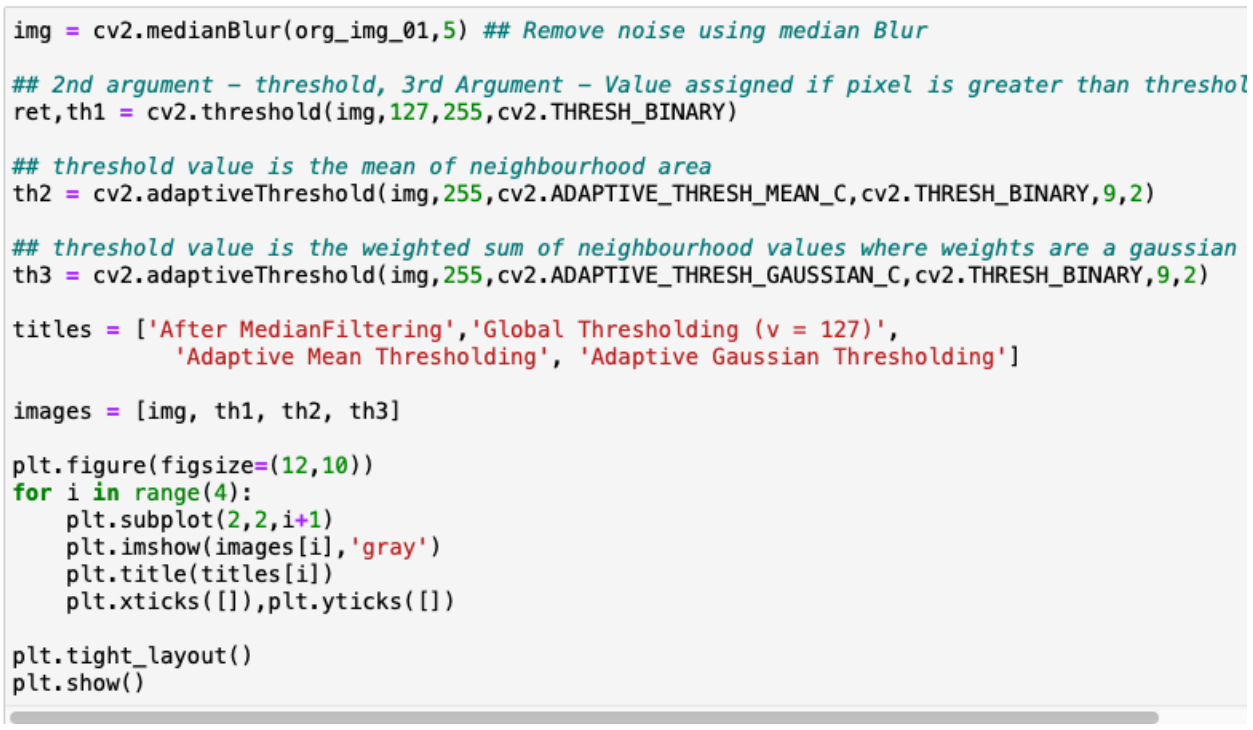
```
print(type(org_img_01))
plt.figure(figsize=(7,6))
plt.title("Original Image")
plt.imshow(org_img_02,'gray')
```

```
<class 'numpy.ndarray'>

<matplotlib.image.AxesImage at 0x23714ccc2c8>
```

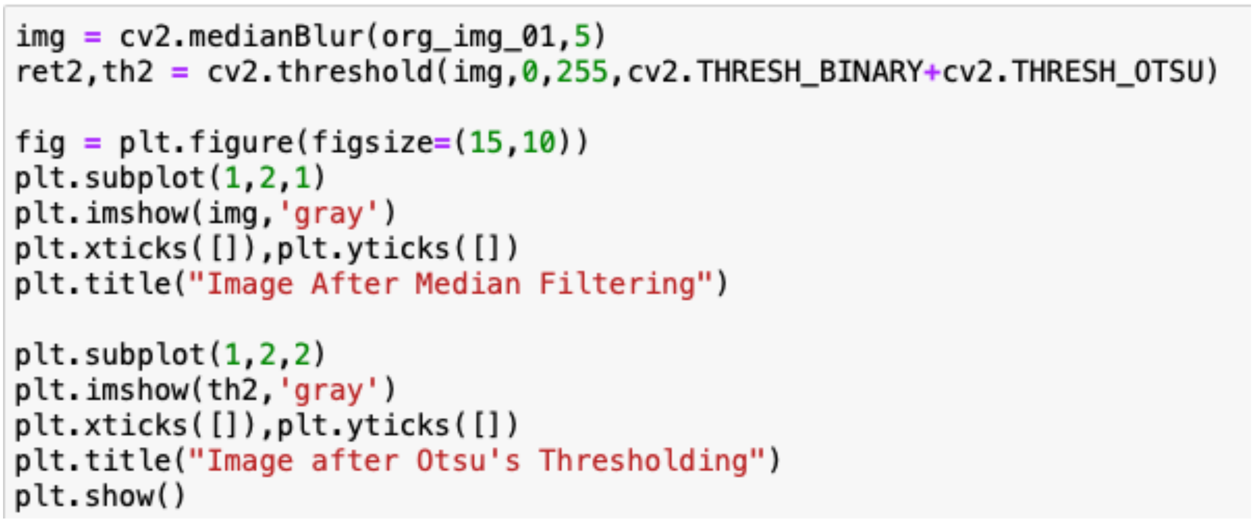
3.Removing noise and global thresholding

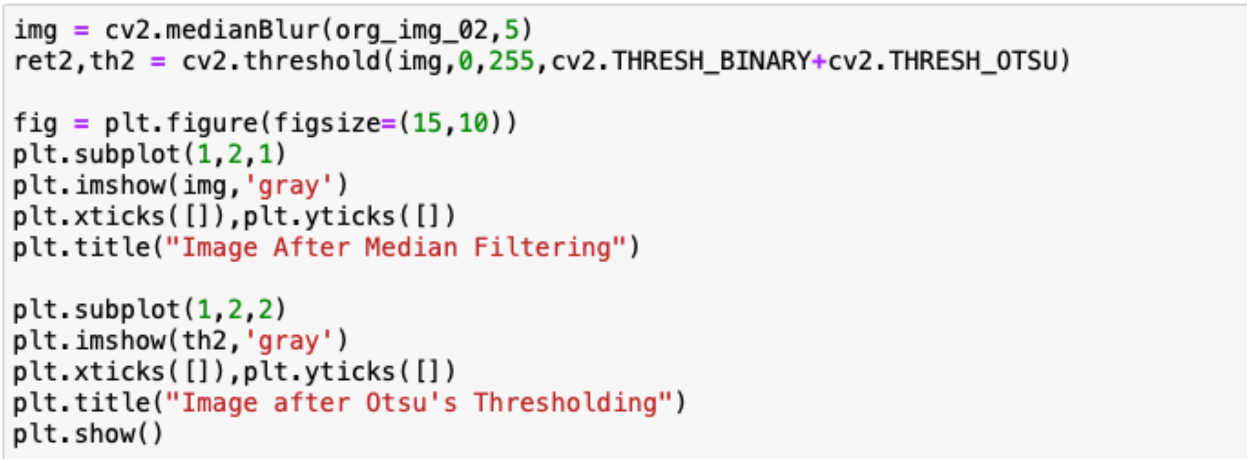

```
print(org_img_01.shape," ",org_img_02.shape," ",org_img_03.shape)
```

```
(553, 830)   (4032, 3024)   (4032, 3024)
```

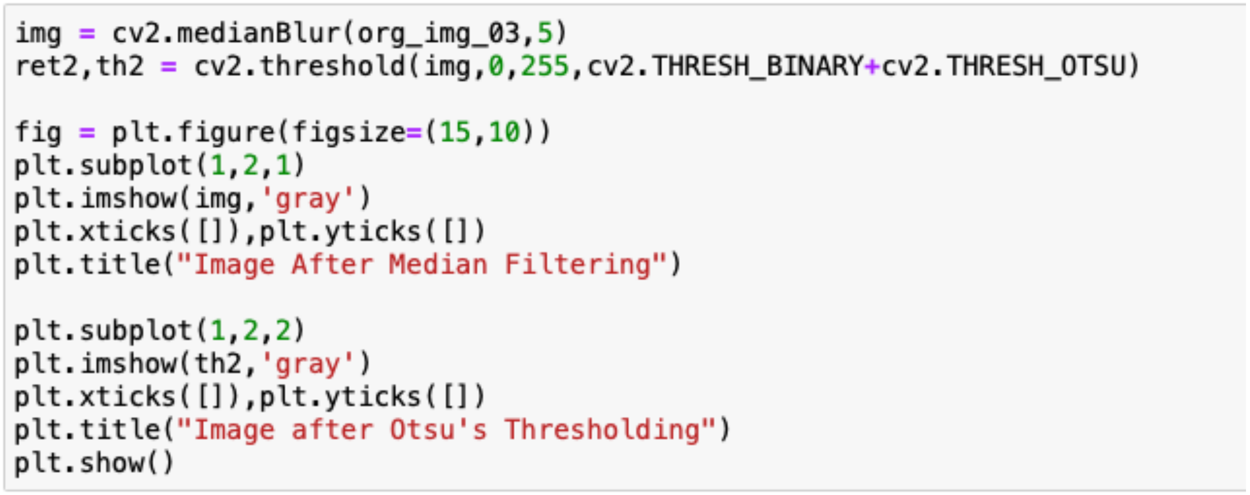

```
img = cv2.medianBlur(org_img_01,5) ## Remove noise using median Blur

## 2nd argument - threshold, 3rd Argument - Value assigned if pixel is greater than threshol
ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

## threshold value is the mean of neighbourhood area
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,9,2)

## threshold value is the weighted sum of neighbourhood values where weights are a gaussian
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,9,2)

titles = ['After MedianFiltering','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.xticks([]),plt.yticks([])

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

4. Using Otsu’s Method


```
img = cv2.medianBlur(org_img_01,5)
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

fig = plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.imshow(img,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image After Median Filtering")

plt.subplot(1,2,2)
plt.imshow(th2,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image after Otsu's Thresholding")
plt.show()
```



```
img = cv2.medianBlur(org_img_02,5)
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

fig = plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.imshow(img,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image After Median Filtering")

plt.subplot(1,2,2)
plt.imshow(th2,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image after Otsu's Thresholding")
plt.show()
```



```
img = cv2.medianBlur(org_img_03,5)
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

fig = plt.figure(figsize=(15,10))
plt.subplot(1,2,1)
plt.imshow(img,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image After Median Filtering")

plt.subplot(1,2,2)
plt.imshow(th2,'gray')
plt.xticks([]),plt.yticks([])
plt.title("Image after Otsu's Thresholding")
plt.show()
```

<u>Final Images</u>

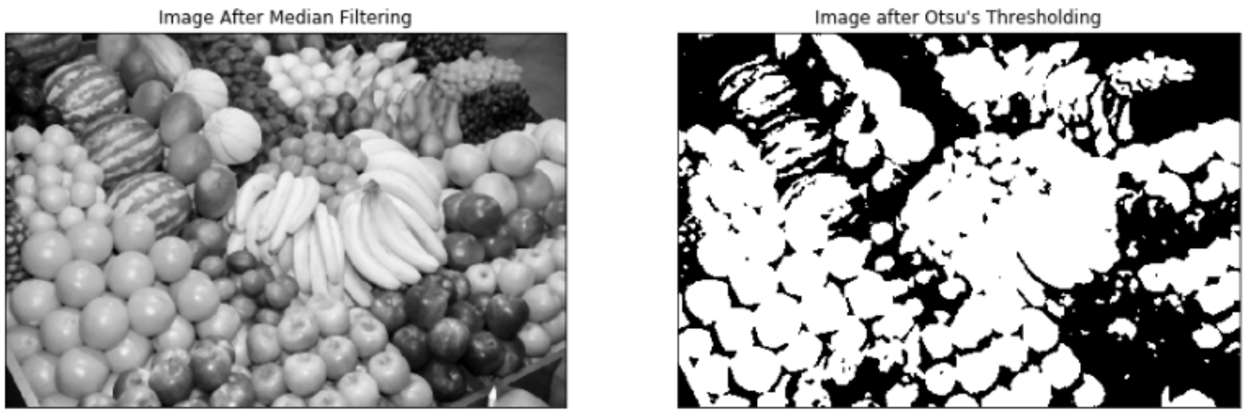


Image After Median Filtering

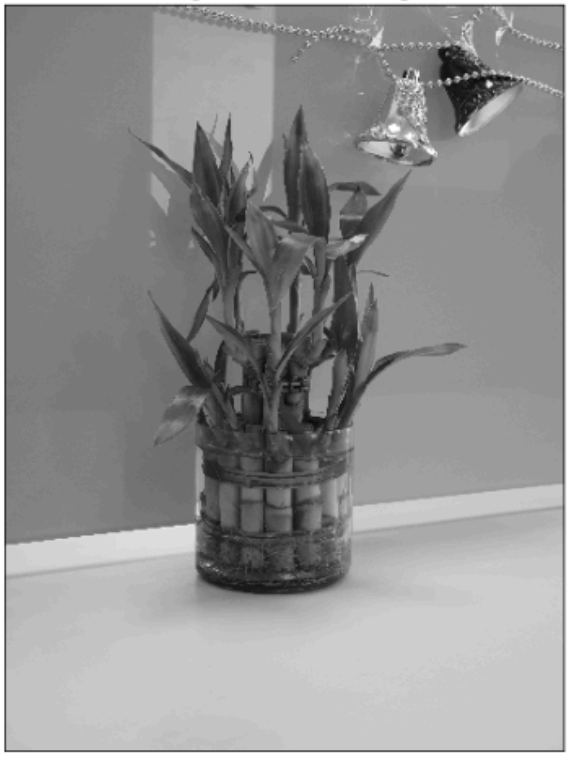


Image after Otsu's Thresholding

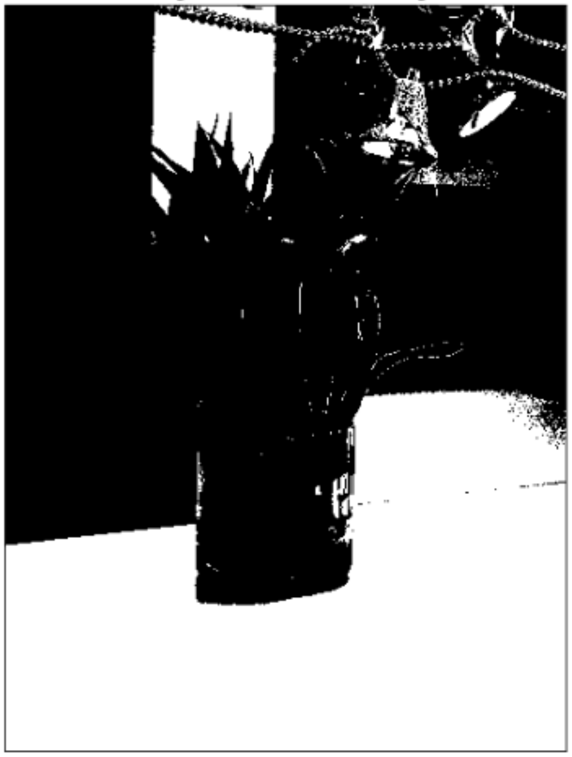


Image After Median Filtering

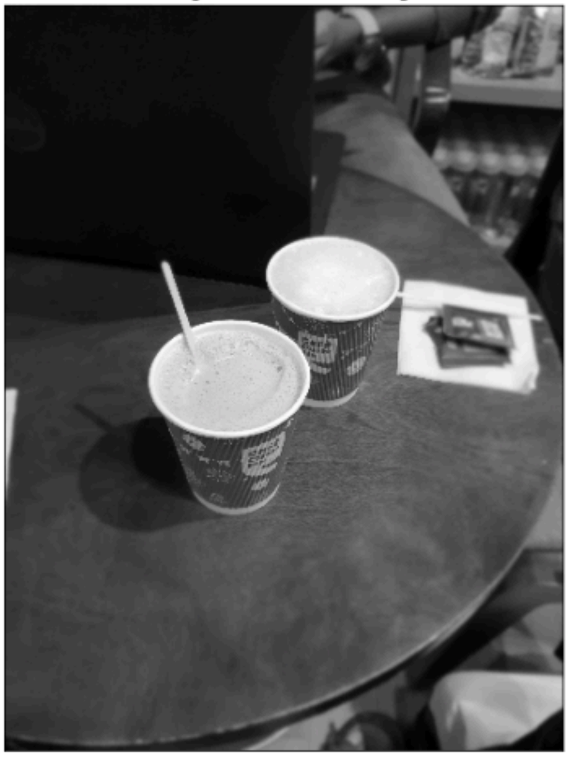


Image after Otsu's Thresholding