Lab_05_21_27_19_DS

March 11, 2022

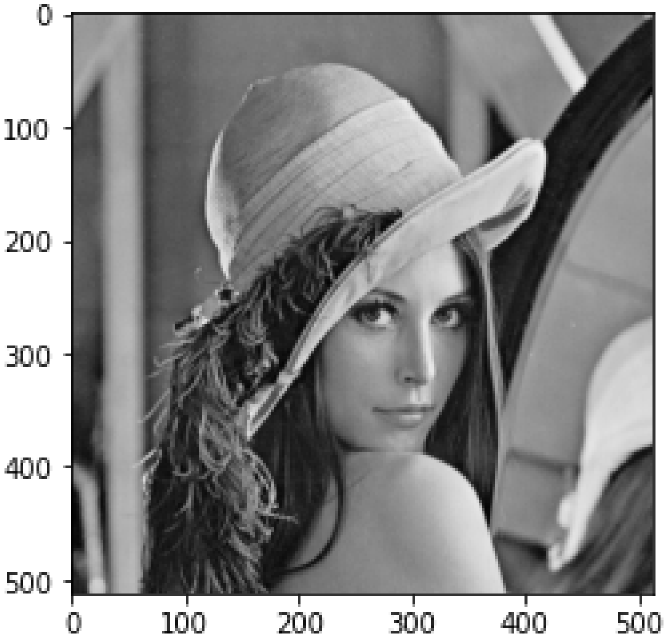
Name: Saurabh S. Ramteke
Roll No: 21-27-19
M.Tech: Data Science

Obj: To know how to implement different thresholding techniiques in python using opencv packagae

[21]:
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

[22]:
```
img = cv2.imread(r"C:\Users\saura\Desktop\Ongoing\Notes\01.
↪LAB_ass\Computer_vision_basics\Images\Lenna.png")

img2gray = np.copy(img)
img2gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.imshow(img2gray, cmap = 'gray')
```

[22]:
```
<matplotlib.image.AxesImage at 0x16f9d153d00>
```

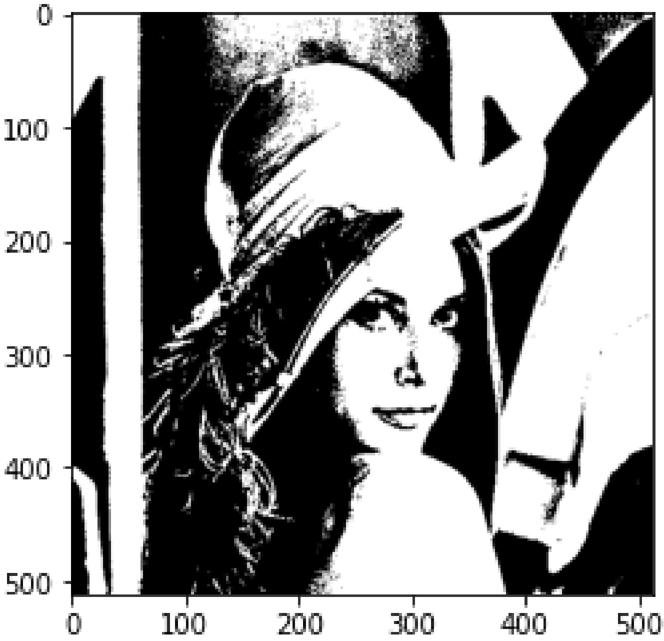


[52]:

```
col = img2gray.size//len(img2gray)
row = len(img2gray)
threshold = 127
x = img2gray.shape

binar = np.zeros(x)

for i in range(row):
    for j in range(col):
        if img2gray[i,j] < threshold:
            binar[i,j] += 0
        else:
            binar[i,j] += 1

plt.imshow(binar, cmap= 'gray')
```

[52]:

```
<matplotlib.image.AxesImage at 0x16fa1095670>
```

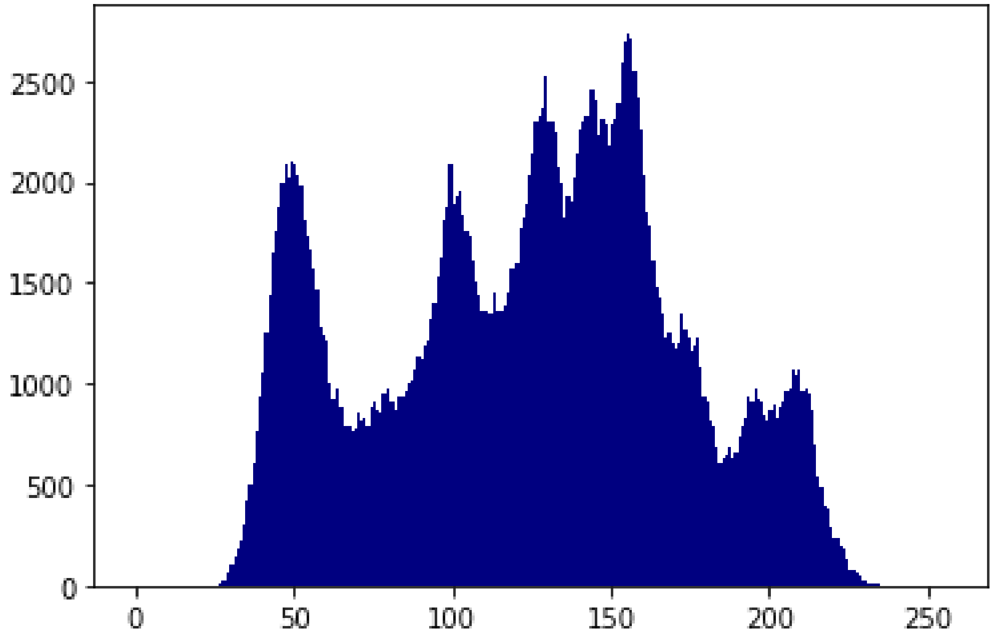


Histogram

[24]:

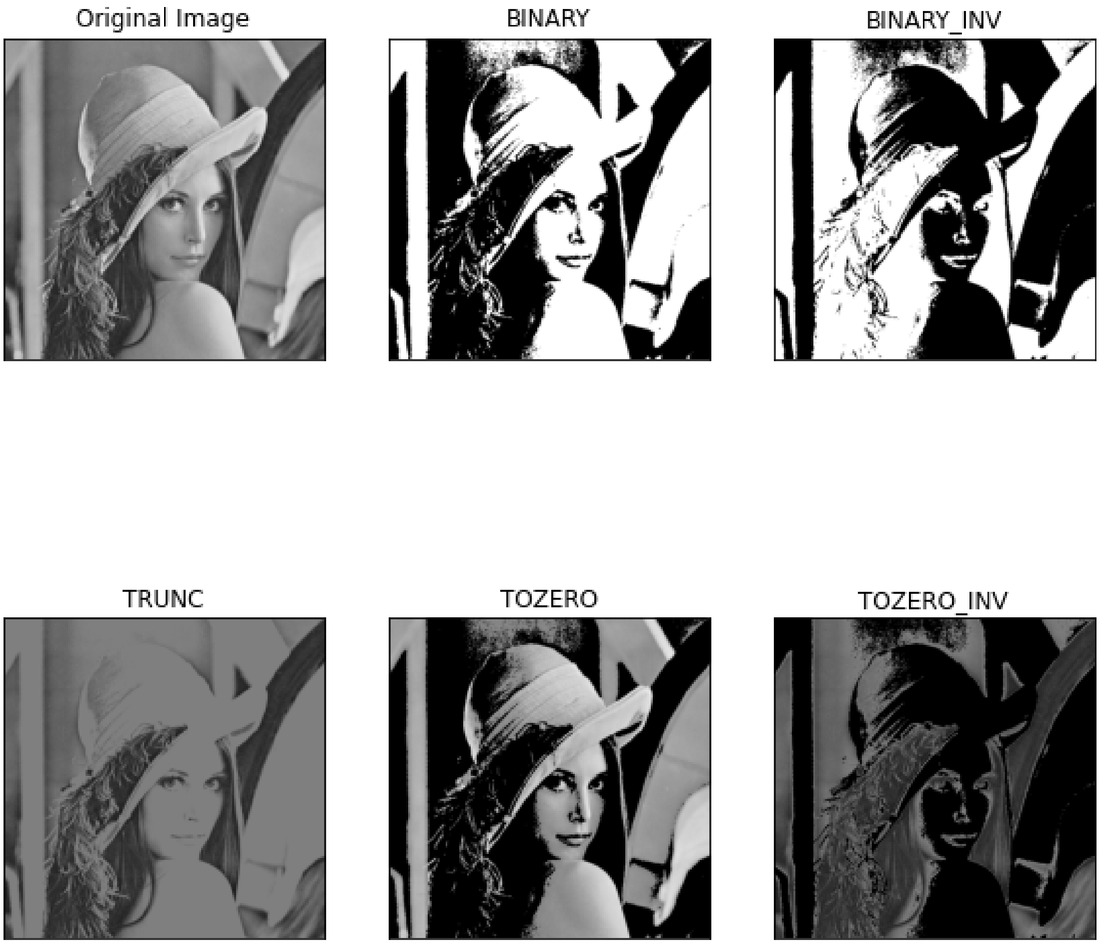
```
plt.hist(img2gray.ravel(), 256, [0,256], color = 'navy')
plt.show
```

[24]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Simple Thresholding

[45]:
```python
#source image should be grayscale
img_simp = np.copy(img2gray)

ret,thresh1 = cv2.threshold(img_simp,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img_simp,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img_simp,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img_simp,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img_simp,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img_simp, thresh1, thresh2, thresh3, thresh4, thresh5]

plt.figure(figsize = (10,10))
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],cmap ='gray',vmin=0,vmax=255)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

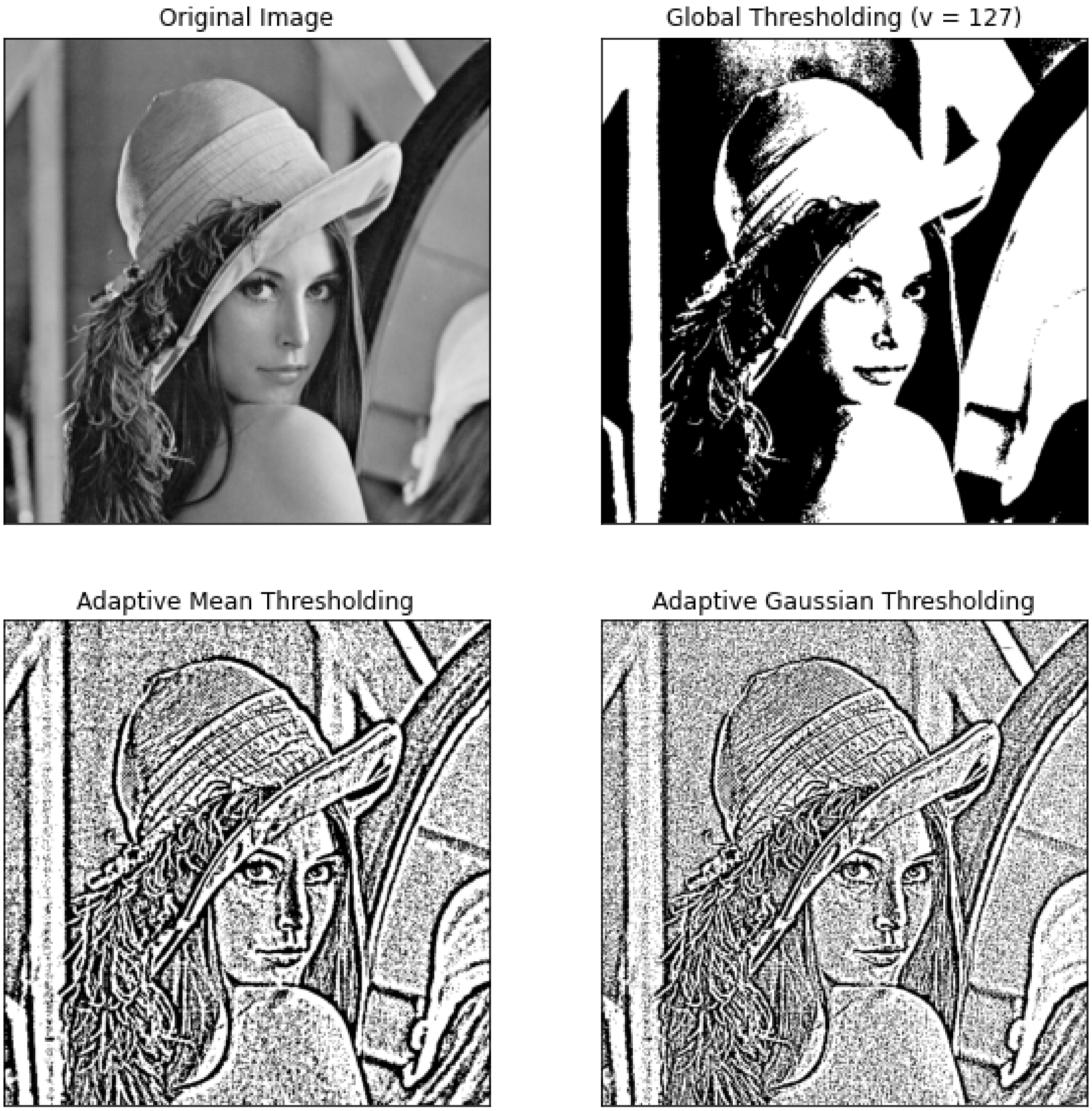


Adaptive Thresholding cv2.ADAPTIVE_THRESH_MEAN_C: The threshold value is the mean of the neighbourhood area minus the constant C. cv2.ADAPTIVE_THRESH_GAUSSIAN_C: The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant C’’’

[53]:

```
img_adapt = img2gray.copy()

ret,th1 = cv2.threshold(img_adapt,127,255,cv2.THRESH_BINARY)
th2     = cv2.adaptiveThreshold(img_adapt,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,11,2)
th3     = cv2.adaptiveThreshold(img_adapt,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img_adapt, th1, th2, th3]
```

```
plt.figure(figsize = (10,10))
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.show()
```



Original Image

Global Thresholding (v = 127)

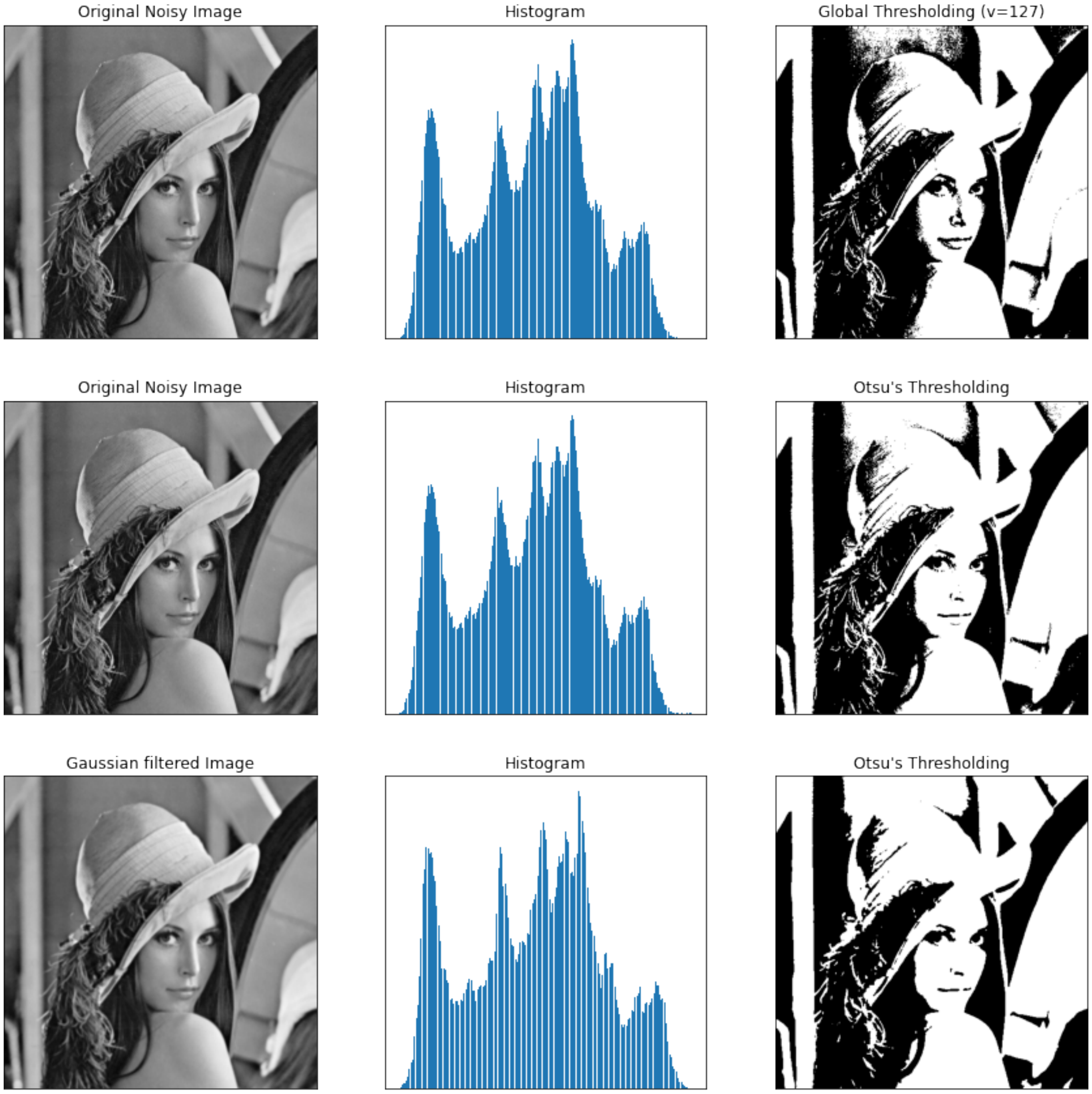
Adaptive Mean Thresholding

Adaptive Gaussian Thresholding

Otsu's Binarization

[50]:

```
img_otsu = np.copy(img2gray)


# global thresholding
ret1,th1 = cv2.threshold(img_otsu,127,255,cv2.THRESH_BINARY)

# Otsu's thresholding
ret2,th2 = cv2.threshold(img_otsu,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# Otsu's thresholding after Gaussian filtering
blur     = cv2.GaussianBlur(img_otsu,(5,5),0)
ret3,th3 = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

# plot all the images and their histograms
images = [img_otsu, 0, th1,
          img_otsu, 0, th2,
          blur, 0, th3]
titles = ['Original Noisy Image','Histogram','Global Thresholding (v=127)',
          'Original Noisy Image','Histogram',"Otsu's Thresholding",
          'Gaussian filtered Image','Histogram',"Otsu's Thresholding"]

plt.figure(figsize = (15,15))

for i in range(3):
    plt.subplot(3,3,i*3+1),   plt.imshow(images[i*3],'gray')
    plt.title(titles[i*3]),   plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+2),   plt.hist(images[i*3].ravel(),256)
    plt.title(titles[i*3+1]), plt.xticks([]), plt.yticks([])
    plt.subplot(3,3,i*3+3),   plt.imshow(images[i*3+2],'gray')
    plt.title(titles[i*3+2]), plt.xticks([]), plt.yticks([])
plt.show()
```


Original Noisy Image
Histogram
Global Thresholding (v=127)
Original Noisy Image
Histogram
Otsu's Thresholding
Gaussian filtered Image
Histogram
Otsu's Thresholding

[ ]: