**EXPERIMENT 4**

|  |  |
| --- | --- |
| **Name: Shivam Bhosle** | **Roll No: 2** |
| **Subject: Computer Vision (CV)** | **Class/Batch: B-1** |
| **Date of Performance:** | **Date of Submission:** |

|  |
| --- |
| **AIM** |

**Image Thresholding using OpenCV**

|  |
| --- |
| **Theory** |

Thresholding is a technique in OpenCV, which is the assignment of pixel values in relation to the threshold value provided. In thresholding, each pixel value is compared with the threshold value. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value (generally 255). Thresholding is a very popular segmentation technique, used for separating an object considered as a foreground from its background. A threshold is a value which has two regions on its either side i.e. below the threshold or above the threshold.

In Computer Vision, this technique of thresholding is done on grayscale images. So initially, the image has to be converted in grayscale color space.

1. **Simple Thresholding**

The basic Thresholding technique is Binary Thresholding. For every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise, it is set to a maximum value.

The different Simple Thresholding Techniques are:

1. cv2.THRESH\_BINARY: If pixel intensity is greater than the set threshold, value set to 255, else set to 0 (black).
2. cv2.THRESH\_BINARY\_INV: Inverted or Opposite case of cv2.THRESH\_BINARY.
3. cv.THRESH\_TRUNC: If pixel intensity value is greater than threshold, it is truncated to the threshold. The pixel values are set to be the same as the threshold. All other values remain the same.
4. cv.THRESH\_TOZERO: Pixel intensity is set to 0, for all the pixels intensity, less than the threshold value.
5. cv.THRESH\_TOZERO\_INV: Inverted or Opposite case of cv2.THRESH\_TOZERO.

**CODE:**

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread("C:\\Users\\Admin\\Downloads\\img.jpg", 0)

ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY\_INV)

ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH\_TRUNC)

ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO)

ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH\_TOZERO\_INV)

titles = ['Original Image','BINARY','BINARY\_INV','TRUNC','TOZERO','TOZERO\_INV']

images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in range(6):

plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')

plt.title(titles[i])

plt.xticks([]),plt.yticks([])

plt.show()

# **Adaptive Thresholding**

We used one global value as a threshold. But this might not be good in all cases, e.g. if an image has different lighting conditions in different areas. In that case, adaptive thresholding can help. Here, the algorithm determines the threshold for a pixel based on a small region around it. So we get different thresholds for different regions of the same image which gives better results for images with varying illumination.

In addition to the parameters described above, the method [**cv.adaptiveThreshold**](https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ga72b913f352e4a1b1b397736707afcde3) takes three input parameters:

The **adaptiveMethod** decides how the threshold value is calculated:

1. [**cv.ADAPTIVE\_THRESH\_MEAN\_C**](https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa42a3e6ef26247da787bf34030ed772cad0c5199ae8637a6b195062fea4789fa9): The threshold value is the mean of the neighbourhood area minus the constant **C**.

[**cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C**](https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa42a3e6ef26247da787bf34030ed772caf262a01e7a3f112bbab4e8d8e28182dd): The threshold value is a gaussian-weighted sum of the neighbourhood values minus the constant **C**.

**CODE:**

import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread("C:\\Users\\Admin\\Downloads\\img.jpg",0)

img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH\_BINARY)

th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,\

cv2.THRESH\_BINARY,11,2)

th3 = cv2.adaptiveThreshold(img,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, th1, th2, th3]

for i in range(4):

plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')

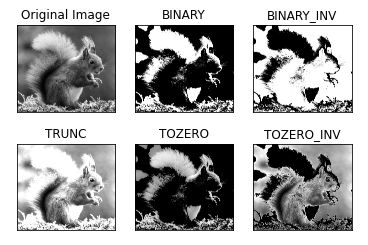
plt.title(titles[i])

plt.xticks([]),plt.yticks([])

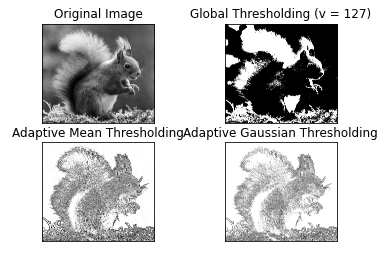
plt.show()

**Output**

**Simple Thresholding**



**Adaptive Thresholding**



|  |
| --- |
| **Conclusion** |

Hence we have implemented Image Thresholding using OpenCV

|  |
| --- |
| **Assessment** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Timely Submission**  **(7)** | **Presentation**  **(06)** | **Understanding**  **(12)** | **Total**  **(25)** | **Sign** |
|  |  |  |  |  |