**Assignment 2 – Applied Image Processing**

**Thresholding**

**Goal**

Thresholding is one of the very fundamental low level operations in digital image processing and analysis. The goal of this lab is to work with the threshold functions available in OpenCV as well as develop from scratch one of the thresholding operations.

**Thresholding in OpenCV**

In OpenCV the thresholding is performed through two functions, **threshold**, and **adaptiveThreshold**. It should be mentioned that where as the availability of threshold operations in OpenCV is fairly extensive, it is by no means an exhaustive implementation of all thresholding methods available in image processing. It should also be emphasized that typically, thresholding operations are performed on gray-scale image and as such a three channel BGR image should be converted to a gray-scale image before performing thresholding. We will be performing single level thresholding in this lab (that is thresholding using a single global or local/adaptive threshold). Later in the course, we would use the k-means to perform multi-level thresholding as well.

**Reference Link**

<https://docs.opencv.org/3.4.3/d7/d4d/tutorial_py_thresholding.html>

**'threshold' Function**

The threshold function of the OpenCV can be used in the most trivial/easy case to threshold a image into white and black regions on the basis of a single global threshold, that is, irrespective of the image intensities present in the image, all pixel values above the reference threshold value are converted to the white (intensity 255) and all lower intensities are converted to black color (intensity 0). At this point it should be remembered that thresholding is a pixel wise operation.

Perform a global threshold of the given image through the function,

ret,thresh1 = [cv.threshold](https://docs.opencv.org/3.4.3/d7/d1b/group__imgproc__misc.html" \l "gae8a4a146d1ca78c626a53577199e9c57)(img,127,255,cv.THRESH\_BINARY)

where 'thresh1' is the returned image which contains only the white and black pixels and is a single channel image. The returned value 'ret' can be ignored for the moment. The 'img' is the input gray-scale image. The value 127 is the global threshold. The 255 value is the highest value in the input image. The cv.THRESH\_BINARY tells us that the result image is to be created with only two (binary) white and black colors. For details of these output generation options, please look at the reference link goven above.

Please play with a range of threshold values and observe their impact on the generated result.

**adaptiveThreshold Function**

The adaptiveThreshold function can be used to threshold the image in a way where each pixel value is thresholded using a threshold which is custom calculated in a window centered around a specific pixel. The calculation can be done in one of two ways, as discussed in class, either the threshold can be calculated by computing the mean of the window around this specific pixel or by computing a Gaussian weighted mean of the intensity values in the given window.

th2 = [cv.adaptiveThreshold](https://docs.opencv.org/3.4.3/d7/d1b/group__imgproc__misc.html" \l "ga72b913f352e4a1b1b397736707afcde3)(img, 255, cv.ADAPTIVE\_THRESH\_MEAN\_C,\

cv.THRESH\_BINARY,11,2)

th3 = [cv.adaptiveThreshold](https://docs.opencv.org/3.4.3/d7/d1b/group__imgproc__misc.html" \l "ga72b913f352e4a1b1b397736707afcde3)(img, 255, cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C,\

cv.THRESH\_BINARY,11,2)

In the case given above, the additional parameters are the last parameters 11 and 2. The parameter 11 specifies the length of a side of a square window whose center pixel is the pixel to be thresholded. In other words if he value of this parameter is 11 then it implies that for a pixel index by coordinates y\_k and x\_k the window will extend from the y coordinates (y\_k-5 to y\_k+5) and x coordinates (x\_k-5 to x\_k+5) The last parameter 2 is just a constant which is subtracted from all the calculated means.

**Lab Task**

Create a function of your own where you implement the adaptive\_mean\_thresh functionality from scratch.

1. Create a function which takes the same parameters as the adaptiveThreshold function.

2. You should use a pair of for loops to loop over the whole image and threshold it.

3. For each pixel automatically select a window around this pixel.

4. Calculate the mean of each window and use it to threshold the pixel in question.

5. To clarify if you have a 100\*100 pixel window and a mean calculation window of size say 5\*5 pixels then the first window will be created in the index range [0:5, 0:5] (remember in Python the higher range is exclusive not inclusive and dimension ordering is row, column) and the mean will be used to threshold the pixel at location (2,2). Then the next window will be created at locations [1:6, 0:5]. This will continue until the window hits the lower part of the image at [95:100, 0:5] and then the new window will be created at [0:5, 1:6].

6. Finally make sure that the thresholded pixel value is updated into a destination/result image and not into the source image since updating the result back into the source image will result in wrong threshold results.

7. Use appropriate value of the window as well as the parameter which specifies the value to be subtracted from each calculated mean.