**Digital Image Processing**

**Lab/Assignment 1**

**Dr. Omer Ishaq**

**Introduction:**

The goal of this lab/assignment is to familiarize the student with the very basic operations involved in reading, writing, concatenating and manipulating images. In addition, we will discuss ways and means of converting data from one color space to another.

**Environment:**

We will be working with Opencv and Python. Opencv is supported on both Windows and Linux variants. There is no preference of IDE, you can use Anaconda, Spyder or any other IDE/Python package of your choice.

**Step 1: Load images**

In Opencv, an image can be loaded with the *imread* function. Typically *imread* takes two arguments. The first one is the image path and name and the second one is a number either 1 or 0 signifying whether the image is being loaded in the BGR or gray-color model. The two aforementioned color modes result in three and single channel images, respectively.

It should also be mentioned that the loaded images are returned as two or three dimensional NUMPY arrays. A usual image load statement would look like

*img = cv2.imread('test\_image.jpg', 1)*

Try loading different images from the given folder and observe their number of channels as well as rows and column numbers. Please observe that the first index in a NUMPY array refers the rows of the image and the second one to the columns, finally the third index (if it exists) refers to the number of color channels.

**Step 2: Convert to gray-scale or HSV space**

Conversion of a BGR image to gray-color can also be done with the *cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)* command. Where *img* is the image to be converted and the COLOR\_BGR2GRAY specifies the color being converted. There is a wide range of color conversions available in Opencv and you are encouraged to explore them on your own. If you use the code COLOR\_BGR2HSV it converts the image from BGR to HSV space which is typically easier to work with if you want to segment or delineate an object in an image based on its color. Convert BGR image to GRAY and HSV and observe if there is any change in the dimensionality of the image. For more details of the HSV space please refer to the link <https://aishack.in/tutorials/color-spaces-2/>and the link<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html>

**Step 3: Filter image by color range in HSV**

Now we want to find a particular colored object in Opencv, you will have to provide a lower and higher color bound to Opencv. Remember, as discussed in class the color of an object maps to the H channel of the HSV space. As an example if you want to identify only the blue color in an image. You can use the following code,

blue\_lower=np.array([100,150,0],np.uint8)

blue\_upper=np.array([140,255,255],np.uint8)

blue=cv2.inRange(img,blue\_lower,blue\_upper)

where the resultant image blue has the blue colored region shown as white and the rest of the image shown as black. Notice here that the we are saying the identify/separate all pixels which are in the range of H between 100 and 140. Please keep in mind that in the Opencv implementation of HSV, the the H, S and V channels have ranges 0-180, 0-255 and 0-255 respectively.

Identifying red color is a bit more complex since it exists in the range 170-180 and 0-10 of the channel. Hint: if you want to identify red color, filter in the image twice, one for each H range and then OR the two results to get a unified result over the two H ranges. I leave it to the students to figure out how to OR the two results of a red color identification.

Additionally, students are encouraged/requested to write their own in-range function where using if statements and loops, the students can separate pixels which belong in a particular HSV range.

**Step 4: Concatenating images**

In the last step what I want you guys to do is to load a color image, filter it to get the red color objects in the image and then horizontally concatenate the two images (Hint: as simple as horizontally concatenating two arrays) so that you have a single image with the original color image on left and the red colored identification on the right, the save this image with the *imwrite* command.

**Deliverables, provide both code and result, code in python.**

1. Look at both the blue and red colored images provided. Using the built in inRange function and appropriate values of the HSV range segment the red and blue objects. Particularly look at the second link provided in the first page of this assignment.

2. Second code your own inRange function using if statement and loops, compare the result you get from your own inRange function with the result gotten from the built in function for the same ranges of HSV

3. Third concatenate the input and the result images so that in a single output image, the original image is on the left and the result image is on the right.