**Muhammad Moiz Sajid**

**19i-0692**

**CS-B**

**DIP Assignment 1-B**

**Question 1**

* **Load the image: Load the image using the cv2.imread() function.**
* **Convert the image to HSV color space: Convert the image to the HSV color space using the cv2.cvtColor() function.**
* **Define a mask for white color range: Create a mask for the white color range by defining lower and upper threshold values in the HSV color space. The mask will be a binary image with white pixels where the color falls within the specified range and black pixels elsewhere.**
* **Apply the mask to the image: Apply the mask to the original image using the cv2.bitwise\_and() function. This will give you an image with only the white color range highlighted.**
* **Change the color of the highlighted area: Change the color of the highlighted area using the cv2.cvtColor() and cv2.addWeighted() functions.**
* **Display the result: Display the result using the cv2.imshow() and cv2.waitKey() functions.**
* **We also convert the grayscale img to a 3-channel image using cv2.cvtColor() and cv2.COLOR\_GRAY2BGR before applying the inverted mask using cv2.bitwise\_and(). This ensures that both images have the same number of channels before blending them using cv2.addWeighted().**
* **A picture containing indoor, blur

  Description automatically generated**

**Output Images:**

****

**A picture containing decorated, blur

Description automatically generated**

**Question 2**

* **In this code, we first load the image as grayscale and create a mask for the white color range using the same threshold value as before. We then create an empty list to store the number of white pixels in each row of the mask, and use a for loop to iterate over each row of the mask and count the number of white pixels using the cv2.countNonZero() function. Finally, we print the list of number of white pixels in each row.**
* **To count the number of white pixels in each column of an image and store it in a 1D list using Python and OpenCV, you can use the cv2.countNonZero() function on each column of the transposed image.**

**Input Image:**

****

**Output Image:**

**A picture containing text

Description automatically generated**

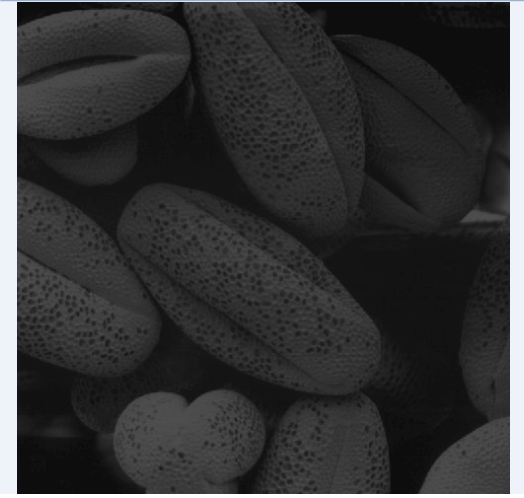
**Chart, histogram

Description automatically generated**

**Question 3**

* **In this code, we first load the image as grayscale. We then define the logarithmic transformation function using the formula c \* log(1 + pixel\_value).**
* **Where c is a scaling constant and pixel\_value is the value of the pixel in the input image.**
* **We use the maximum pixel value of the input image to compute c. We then apply the logarithmic transformation using the cv2.LUT() function, and normalize the resulting values to the 0-255 range using the np.uint8() function.**
* **Finally, we plot the original and transformed images side by side using the matplotlib.pyplot.imshow() function.**

**Input image:**

****

**Output:**

**A picture containing chart

Description automatically generated**

**Question 4**

* **In this code, we load the image as grayscale using the cv2.imread() function.**
* **We initialize a list called intensityCount to store the counts for each intensity value from 0 to 255.**
* **We then iterate over the pixels in the image using nested for loops, and update the count for each intensity value by incrementing the corresponding element in intensityCount.**
* **Finally, we print the counts for each intensity value using a for loop.**
* **We then apply a 3x3 and 7x7 box filter using the cv2.boxFilter() function, which takes three arguments: the input image, the output image data type (-1 indicates that the output image will have the same data type as the input image), and the kernel size (3,3). The kernel size specifies the size of the window over which the filter is applied.**

**Output:**

**Chart, histogram

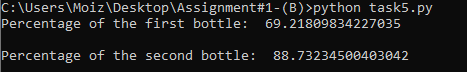
Description automatically generated**

**A picture containing text, different

Description automatically generated**

**Question 5:**

* **Load the two images using cv2.imread(), and convert them to grayscale if they are not already in grayscale.**
* **Threshold both images using a suitable threshold value to obtain binary images where the pixels that correspond to the liquid are white (value 255) and the rest of the image is black (value 0).**
* **We first count the total area of the bottle in the variable total\_size.**
* **We then calculate the sizes of liquid contained in each bottle and store them in the “first” and “second” variables respectively.**
* **Finally we apply the general formula to calculate the percentage by dividing the liquid area over total area of the bottle and we get the percentages.**

****