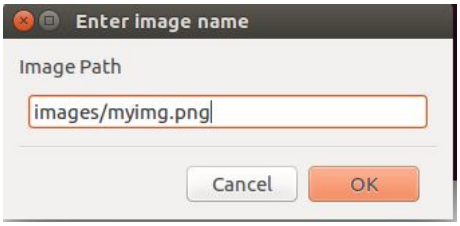
Report: HW2 CS 682

Submitted by Aakarshika Priydarshi
G01047156

1.1.1

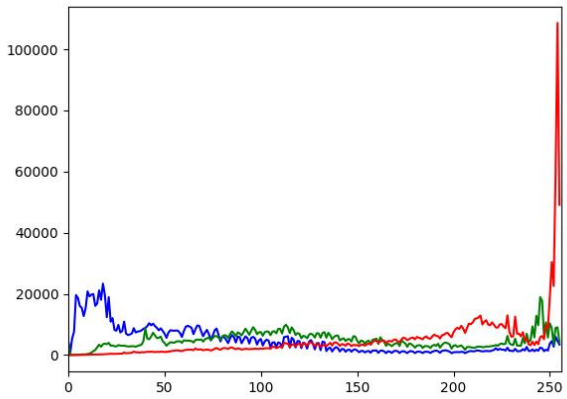
The dialog box to input the image path is created using the library **wxpython**. This library should be installed and imported in the program to run.



[If it is not installed, then the lines 6-16 in the file HW2_CS682_apriydar.py should be commented out and the 'imgpath' variable at line 5 should be given the appropriate value of the image path.]

1.1.2

The color histogram for input image is saved in '**results/HistogramRGB.png' file**. The function cv2.calHist() takes the arguments that can specify which layer (color) of the image is to be mapped. Hence, all three colors can be mapped with different colors.

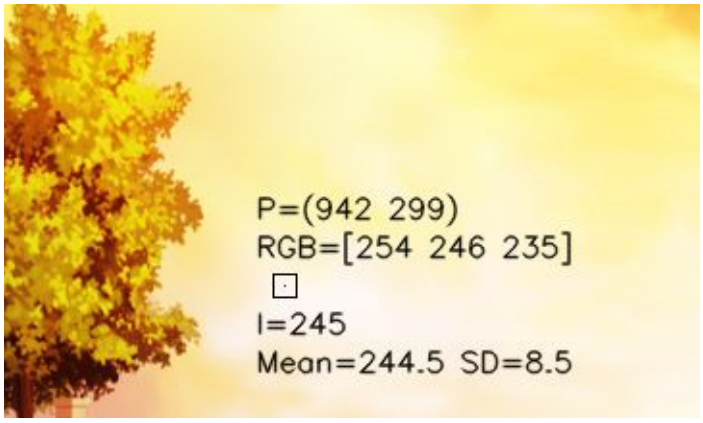


1.1.3

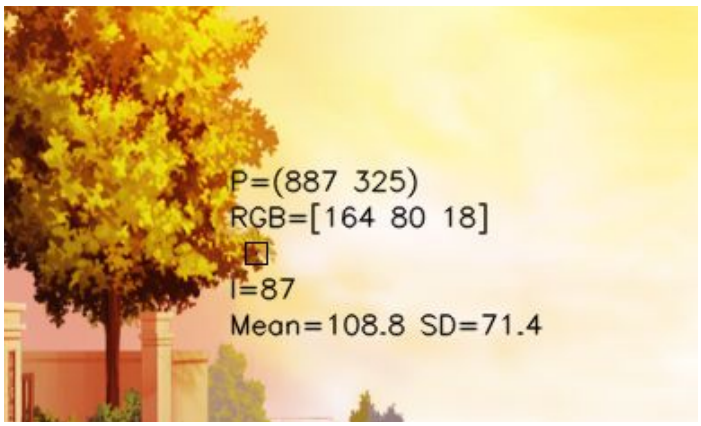
For cursor move event handling in OpenCV Python, cv2.**setMouseCallback** function is used to bind cv2.MOUSE_MOVE event to a function. Here, we receive the current x and y coordinates of the pixel at the mouse cursor.
(a). The rectangle() function creates the border of the 11X11 box around this pixel.
(b). The x, y coordinates help us to calculate the RGB value of the pixel at the position, and display it using putText() function.
(c). Using these red, green, and blue values, we can calculate the intensity using the formula (r+g+b)/3.
(d). Once we have the pixel coordinates, we can extract a 'window' of the desired 11X11 size and use the numpy functions mean() and std() to calculate the average and standard deviation of the window respectively.

1.1.4

In the image, as the cursor is moved to various parts, we can judge if the area inside the window is ‘homogeneous’ or ‘inhomogeneous’.
Homogeneous parts of an image are those that are similarly colored. This means that the difference in the values of the pixels in the defined window is low. Hence, the **mean of these values is close to the intensity of the pixel**. Also, the **standard deviation, (and variance) inside this window has a low value**, i.e. close to zero.



In inhomogeneous parts, however, the **intensity and mean are not related** because the pixels inside the window have variable values, and the **variance has a higher value.**



1.2

```
aakarshika@aakarshika-Lenovo-G50-80:~/p/ComputerVision/CV/HW2$ python HW2_CS682_apriydar.py
init done
opengl support available
2.A. Loading images and calculating histograms...
Saving histogram for image:  images/ST2MainHall4001.jpg
Saving histogram for image:  images/ST2MainHall4002.jpg
```

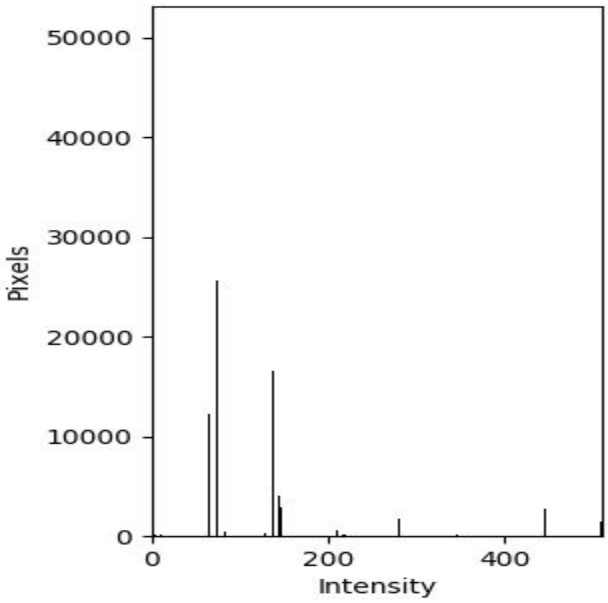
1. The images from the given zip archive are loaded using a loop.

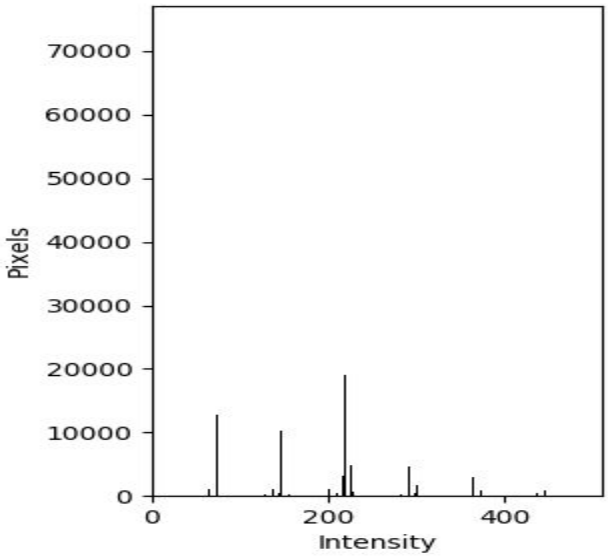
A. To calculate the required 512 bin histogram, we need to extract the R, G, and B numpy arrays from each image. This is done using ‘ r=np.array(img[:,:,2] ,dtype=np.uint16) ’ for all colors. This is because simply extracting the r,g,b arrays using the cv2.split() function causes it to return the arrays in uint8 dtype. The multiplication and addition operations on these arrays results in an array of uint8 type, which has a range of 0-255, hence, cannot be used for the intensity bin, which has a range 0-512. Therefore, individually each layer is converted to the desired dtype and then used.

After extracting the RGB arrays, we calculate the intensity using the formula

intensity = ((r >> 5) << 6) + ((g >> 5) << 3) + (b >> 5)

that **calculates the intensity using bit-shifting for each pixel in parallel using numpy arrays**. This intensity array is then used to create the histograms. These histograms are saved in an array and as images in the folder 'results'.





B. Two functions are defined as 'histogram_intersection()' and 'chi_squared_measure()' to calculate the similarity between two given histogram arrays and return the similarity measure. The functions calculate distance measure using the formula given for the corresponding distance measure function.

1. Histogram intersection results in values ranging from 0 to 1.
2. Chi-squared measure similarities result in a very high range, with higher the value, lesser the similarity.

```
2.B. Calculating histogram similarity for images:  98   96
        His Int =  0.837018432973
        Chi Sq =  11535.0
2.B. Calculating histogram similarity for images:  98   97
        His Int =  0.868993986001
        Chi Sq =  7939.0
2.B. Calculating histogram similarity for images:  98   98
        His Int =  1.0
        Chi Sq =  0.0
All histograms saved to ./results/
Histogram similarity graph saved to ./results/
```

C. All histogram similarities between each pair of images are calculated and saved in a 2-D array for both distance measures. The values for a pair of images are same commutatively, hence, for a pair of images (i,j), the pair (j,i) is not calculated.

The values in these arrays are normalized to a 0-255 value bin.

This is done as follows:

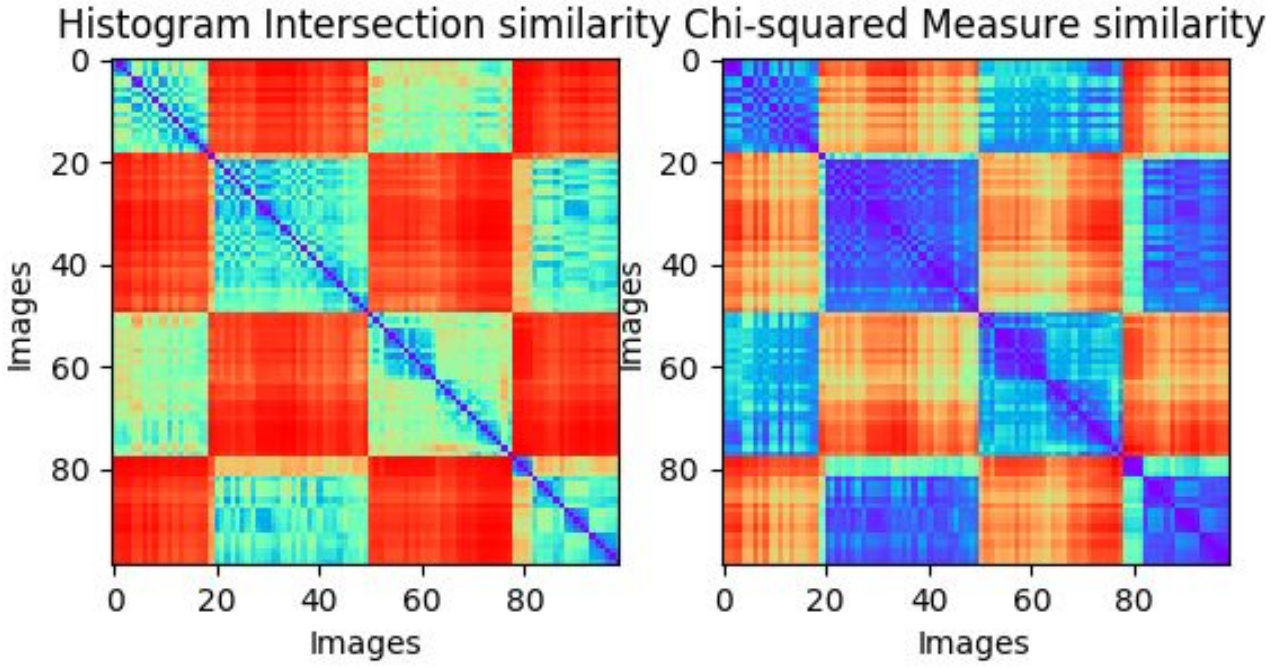
min = HS.min()

max = HS.max()

graph = (HS - min)* 255.0 / max

Where HS is the histogram similarity 2-D numpy array depicting similarity between each pair of images.

Hence, now these arrays can be treated as a grayscale image, which represents all 99 images on the 2 axes and the similarity value as the pixel value. These 2-D arrays are represented by matplotlib with a cmap argument as 'rainbow' that gives a range of colors for each value of the grayscale. Here, red represents low similarity between images and blue represents high similarity. We notice that the images along the diagonal are the same with highest similarity, and the rest of the map is divided into two portions that have similar images but are not similar to each other.



Similarity in histograms depicted by Confusion matrix. Using heatmap with color 'rainbow' in matplotlib.
Red shows low similarity, and blue shows high similarity.