NAME: SOUNDARYA.N

EMAIL ID: [soundaryanagarajan5@gmail.com](mailto:soundaryanagarajan5@gmail.com)

PHONE NUMBER: 9500841469

**INPUT IMAGE:-**

|  |
| --- |
|  |

**OUTPUT IMAGE:-**

|  |
| --- |
|  |

**Explanation:-**

OpenCV loads color images in BGR color space. With this color space, it is not possible to equalize the histogram without affecting to the color information because all 3 channels contain color information. Therefore you have to convert the BGR image to a color space like YCrCb. In YCrCb color space, the Y channel of the image only contains intensity information where as Cr and Cb channels contain all the color information of the image. Therefore only the Y channel should be processed to get a histogram equalized image without changing any color information. After the processing, the YCrCb image should be converted back to the BGR color space before calling imshow() function.

Let's go through the code line by line.

// Read the image file

Mat image = imread("D:/My OpenCV Website/fly-agaric.jpg");

// Check for failure

if (image.empty())

{

cout << "Could not open or find the image" << endl;

cin.get(); //Wait for any key press

return -1;

}

The above code segment loads the image from the specified file. If it is failed to load the image, the program will exit.

//Convert the image from BGR to YCrCb color space

Mat hist\_equalized\_image;

cvtColor(image, hist\_equalized\_image, COLOR\_BGR2YCrCb);

The loaded image is in BGR color space. None of the 3 channels (blue, green and red) of this color space cannot be processed to equalize the histogram because all the channels contain color information. Therefore the loaded image should be converted to the YCrCb color space. In this color space, Y channel only contains intensity information while Cr and Cb channels contain color information. Therefore only the Y channel needs to be processed in order to equalize the histogram.

//Split the image into 3 channels; Y, Cr and Cb channels respectively and store it in a std::vector

vector<Mat> vec\_channels;

split(hist\_equalized\_image, vec\_channels); The above OpenCV function splits the 3 channel image into 3 separate matrices. Each matrix is pushed to the std::vector. vec\_channels[0] contains the Y channel, vec\_channels[1] contains the Cr channel and vec\_channels[2] contains the Cb channel.

//Equalize the histogram of the Y channel

equalizeHist(vec\_channels[0], vec\_channels[0]);The above function equalizes the histogram of the Y channel.

//Merge 3 channels in the vector to form the color image in YCrCB color space.

merge(vec\_channels, hist\_equalized\_image); The above function performs the reverse operation of the split function. It takes a std::vector which consists of 3 matrices representing Y, Cr and Cb channels and creates a 3 channel image in YCrCb color space.

//Convert the histogram equalized image from YCrCb to BGR color space again

cvtColor(hist\_equalized\_image, hist\_equalized\_image, COLOR\_YCrCb2BGR);Above line converts the image in YCrCb color space into the BGR color space. This step is necessary because OpenCV functions like cv::imshow() always expect images in BGR color space.

//Define the names of windows

String windowNameOfOriginalImage = "Original Image";

String windowNameOfHistogramEqualized = "Histogram Equalized Color Image";

// Create windows with the above names

namedWindow(windowNameOfOriginalImage, WINDOW\_NORMAL);

namedWindow(windowNameOfHistogramEqualized, WINDOW\_NORMAL);

imshow(windowNameOfOriginalImage, image);

imshow(windowNameOfHistogramEqualized, hist\_equalized\_image);The above code segment will create windows and show images in them. As both windows are created passing the flag WINDOW\_NORMAL, they can be resized freely.

waitKey(0);

destroyAllWindows();

return 0;The program will wait until any key is pressed. After a key is pressed, all created windows will be destroyed and the program will exit.

return 0;The program will wait until any key is pressed.