COMPUTER VISION - LAB 3 HINTS

Part 1: Histogram Equalization

1. Load a colored image and, if you want, resize (function cv::resize()) it to better fit the screen.
2. Color images in OpenCv are saved as a matrix of triplets b, g, r (in this order), representing the three-dimensional RGB coordinates of the pixel color. To compute an histogram for each color channel, you need to split the 3-channels color images into three one-channels images (function cv::split()), each one holding the intensities for a specific color channel.
3. Use cv::calcHist() to compute an histogram for each channel.
4. Use cv::equalizeHist() to equalizes the R,G and B images.
5. Using the equalized channel, re-assemble an BGR image by using the functioncv::merge()
6. Visualize the input and the equalized image and the histograms of its channels by exploiting the provided helper function (showHistogram(std::vector<cv::Mat>& hists)).
7. Convert the image into the HSV color space (cv::cvtColor() function, with cv::COLOR_BGR2HSV flag), split the resulting image into 3 single-channel images (H,S,V channel)
8. Equalize only one channel between H,S,V , and re-assemble the HSV image with one equalized channel.
9. Switch back to the RGB color space (cv::COLOR_HSV2BGR) and visualize the resulting image
10. Visualize the input and the equalized image in two different windows.
11. Repeat step from 7 for each H,S,V channel.

Part 2: Image Filtering

1. Load the input image and prepare a copy of such image for each type of filter you are going to apply. These copies will be used to store the filtered image.
2. Initialize a different named window for each image (function cv::namedWindow()) and associate to each window a number of trackbar (one trackbar for each parameter to be tuned, function cv::createTrackbar()).
3. The cv::createTrackbar() functions take as parameters, among others:
    a. Pointer to an integer variable whose value reflects the position of the slider. This variable should hold the value of the parameter to be changed.

   b. Maximal position of the slider: put here some reasonable value. The minimal position is always 0.
   c. Pointer to the function to be called every time the slider changes position. This function should be prototyped as void Foo(int,void*); , where the first parameter is the trackbar position and the second parameter is the user data (see the next parameter). If the callback is the NULL pointer, no callbacks are called, but only value is updated. Note that the function you provide should be able to access the input image, to be filtered, the output image and the filter parameter (see point 3a).

4. You may create a structure that includes both filter parameters and images, and pass this structure to the the function of point 3c through the void *pointer, e.g.:

   ```
   struct ImageWithParams
   {
           int param1;
           int param2;
           cv::Mat src_img, filtered_img;
   }
   ```

   Remember to cast back from void * to (ImageWithParams *).
5. Visualize each resulting filtered image (one for each type of filter) in the named related window created above (function cv::imshow()).