21.10.2018

Mareva Zenelaj

150150906

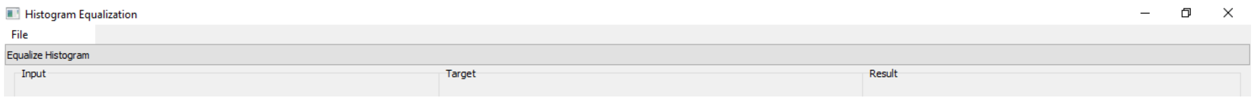
Computer Vision

Homework 1 Report

As requested this homework was done by using PyQt5. Having no previous knowledge regarding the latter I have spent the majority of hours dedicated to the homework on PyQt5. I created a *HistEqualizer* class that inherits from QWidget in order to open windows, add button, widgets, frames or menus.

The homework itself is divided in three parts, adding input, adding target and then equalizing. To divide the screen in three windows *QGroupBox* (3 of the them) and *QVBoxLayout* were used. And to display all three *QGridLayout* was called and then the three widgets were added in the preferred order: input-target-result.

The menu bar is created by using *QMenuBar* and then its components were added by using *addAction* and *addMenu*. The required actions were Open Input, Open Target and Exit. These three actions were connected with the respective function by using *QAction.triggered.connect(preferred function*). An issue in here is that *QMenuBar* has no member that can set its geometry and it appears with a white background and shorter width as shown below:

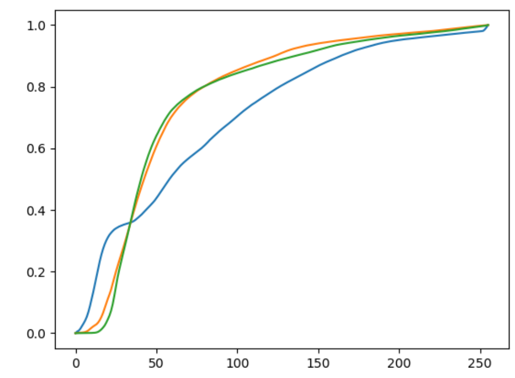


In order to open an image *QFileDialog* and its members were used. All files are accepted. One of the output of this functions, specifically the first one is the path of the file selected. The output is a list therefore using *path = path_file[0]* was needed. After getting the path, I tried to call a pixmap object and get the required image, then set that pixmap to the label specified later in the code, yet it resulted unsuccessful. To open the target and the input the same function was used named as *open_image().*

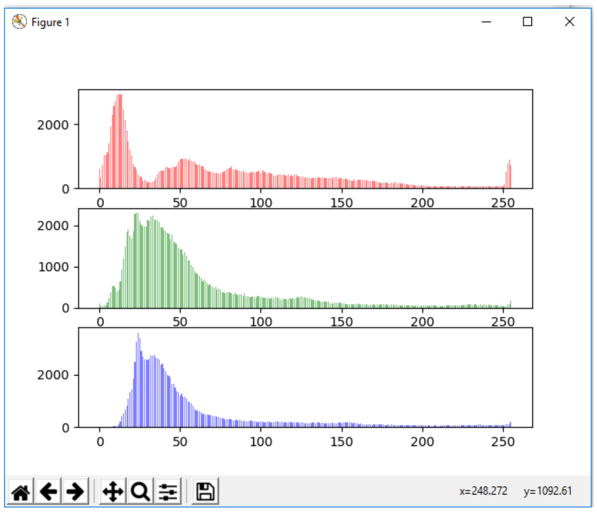
In this *main.py* file, all of the functions of *histogram_match_calculate_f* are imported. They are: *calculate_histogram*, *match_histogram*, *cdf*, *show_histogram* and *equalize_histogram*.

Calculate_histogram takes an image as input and parses through it three times given that the channel number is also 3. While parsing the 3D array the intensity of the pixels is incremented accordingly and the results are saved in *hist* which is also the what the functions returns.

Cdf firstly calculates the histogram of the image that is takes as input and then does the simple calculations of a cumulative distributive function that generally transforms the pdf into a cdf by using the sums. The cdf-s of the color1.png image is shown below.

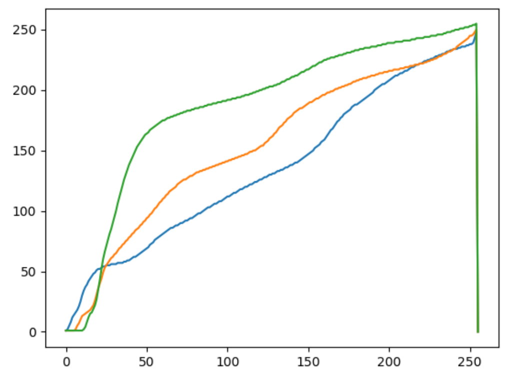


Show_histogram calculated the histogram and then separates the array in three parts: red, green and blue and by using *plt.bar(…)* displays the histogram of the three channels of the image. An example is given below:



These are the color histograms for *color1.png*.

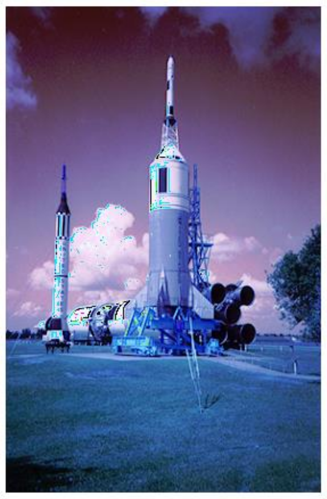
Match_histogram takes the two images as input, calculated their cdf-s by calling *cdf* and then matched the cdf of the first image to the second. The following image shows how the cdf-s are matched.



To match the histograms the method of the Look Up Table is used.

Finally, *equalize_histogram* calls *match_histogram* and divides the LUT in three parts and so does with the first image, the image to be matched. Then the new image is rendered by mapping the three channels separately by using the following command: *K1 = np.uint8(red[img[:,:,0]])* where *red* in the red part of the LUT.

After running the latest function, the following image was the result.



Github files can be found it: https://github.com/MarevaZenelaj/CV-HW1

Note: The homework was left unfinished since a way to show the images in one of the three widgets (frames) was not found by inheriting at the same time from QWidget. I found later on that by inheriting from QMainWindow one of the widgets could be set as the central one, yet it would mean changing the entire code accordingly and since time was lacking, I did not.