**Intensity Transformation**

**Introduction:**

When we are working with images, sometimes you want to modify the intensity values. For instance, you may want to reverse black and the white intensities or you may want to make the darker images darker and the lighter images lighter. An application of intensity transformations is to increase the contrast between certain intensity values so that you can pick out things in an image. Intensity transformation operate on single pixels of an image. Here we have implemented following modules for intensity transformations.

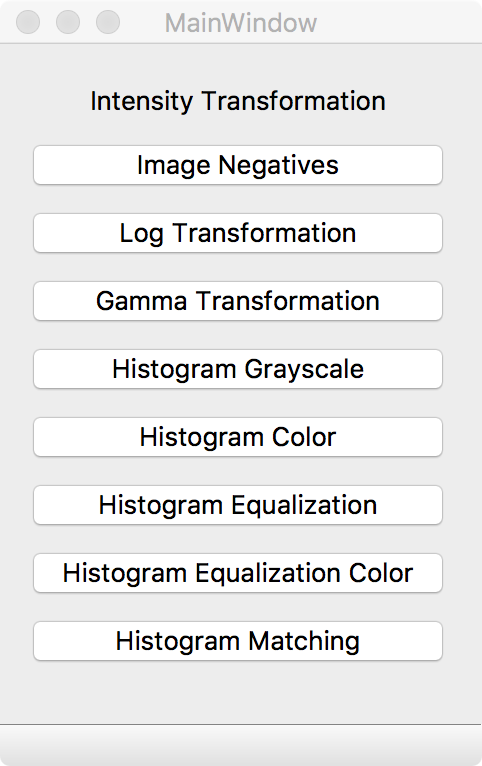
1. UI (PyQt)
2. Histogram
3. Histogram Equalization
4. Histogram Matching
5. Image negative
6. Log Transformations
7. Power gamma Transformations

**UI (PyQt)**

PyQt was used for UI. PyQt is a set of Python bindings for Qt application framework from Qt Company.

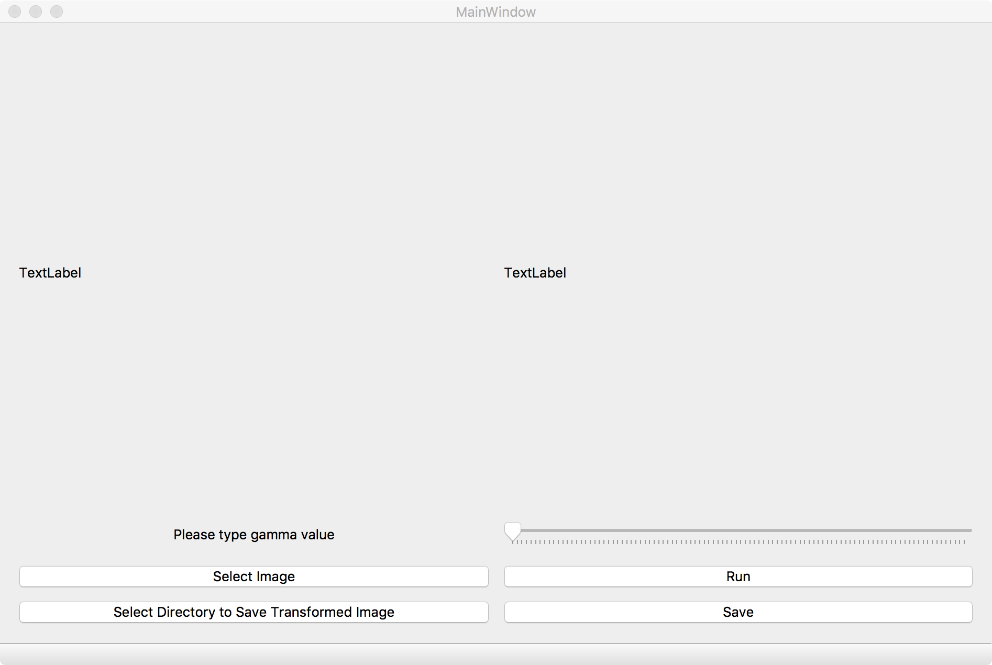
The UI has two parts: main window and sub windows.

**Main window:**



User can select the transformation by clicking on one of the buttons on main window.

Then a **sub window** will show up as below:



The layout and buttons varies according to the type of transformation.

The program is multi-threaded and multiple sub windows can be opened at the same time.

**Histogram**

**Objective:**

Image histogram is a graphical representation of intensity distribution. It plots the number of pixels for each intensity value. Histogram can help to judge the entire intensity distribution at a glance.

**Algorithm:**

1. The 8-bit image (type of uint8) was first loaded into matrix f
2. An integer array with size of 256 was created as h
3. Image will be iterated pixel by pixel to count the occurrence of every pixel intensity
4. the data will be graph with intensity value as X vs number of occurrence of intensity level as Y for coloured image, same process was done for every colour channel

**Histogram Equalization**

**Objective:**

Histogram equalization is used to adjust the contrast of an image using its histogram. The main objective of Histogram Equalization is to make an image visually pleasant. It is mainly used for images with both bright (overexposed) or both dark (underexposed) foreground and background. When Histogram Equalization is applied to such images, their contrast is enhanced. We have implemented Histogram Equalization for both grayscale and coloured images.

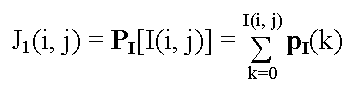
Grayscale image has intensity values in the range 0 to 255. Its Histogram shows distribution of image intensities over range 0 to 255. For a overexposed image, colour intensities are more concentrated towards higher end of histogram and for underexposed dark image they are concentrated towards lower end. Histogram Equalization stretches the concentrated intensities over entire range of image intensity values which enhances the image contrast. Histogram Equalization flattens the histogram of original image. Cumulative probabilities of image intensities are used in this process.

If an image is colored, it has three channel, Red, Green and Blue. Each channel has intensity range from 0 to 255. In this case we cannot separately apply Histogram Equalization to these channel as it may disturb the relative distribution of channels. To get the intensity values from a colored image without disturbing their color balance we we have converted it to YCbCr color space first. YCbCr works best for digital images. Y is equivalent to intensities of image. Histogram Equalization is performed on intensity channel Y. After that YCbCr image is converted back to RGB format. An algorithm for Histogram Equalization is as follows:

**Algorithm:**

Let I be an input grayscale image which is a 2\*2 matrix with data type uint8. It is a Y channel in case of colour image.

1. Compute histogram of an input image I
2. Normalize the histogram which stores probabilities of intensities.
3. Compute cumulative probability histogram P of a normalised histogram.
4. Compute intermediate histogram image J1 :

.

0 ≤ J1(i, j) ≤ 1

Where, I(i, j) is pixel intensity at location (i, j) in an original image matrix I. P[I(i,j)] is a cumulative probability for intensity value I(i, j).

1. Then scale J1 to cover the range 0, ..., K-1, produce the histogram-flattened/equalized image J. K is 256 highest intensity values for grayscale image or Y channel.

J(i, j) = INT [ (K-1)·J1(i, j) + 0.5 ]

1. Return Histogram Equalized Image J

For a coloured image, combine equalized channel Y and Cb, Cr of original image. Convert YCbCr format back to RGB format which is a histogram equalized coloured image.

**Histogram Matching**

**Objective:**

Histogram matching is a process where a time series, image, or higher dimension scalar data is modified such that its histogram matches with the histogram of another (reference) image. A common application of this is to match the images from two sensors with slightly different responses, or from a sensor whose response changes over time. The process of Histogram Matching takes in an input image and produces an output image that is based upon a specified histogram. The required parameters for this algorithm are the input image and the specified image, from which the specified histogram can be obtained.

**Algorithm:**

1. Compute cumulative histogram for the given image and reference image.
2. For every value (xi) in the given image, find the cumulative histogram value given by G(xi).
3. The value G(xi) is in turn the cumulative distribution value in the reference image, namely H(xj).
4. Replace xi with xj.

In practice for discrete valued data one does not step through data values but rather creates a mapping to the output state for each possible input state. In the case of an image this would be a mapping for each of the 256 different states. For RGB images the histogram matching can be applied in two ways. It can be applied to each colour channel independently or a single mapping applied to all channels. In the latter case, this single mapping can be derived from a grey-scaled version of the image, the intensity, luminance, or other similar single measures. In the case where the matching is applied on a per channel basis, coloration effects can occur particularly if one or more channels has a narrow distribution.

**Image Negative**

**Objective:**

Image negative is the process of reproduce the bright portion of photographed subject as dark and dark parts as light areas. Negative was commonly on a strip or sheet of transparent plastic film and when the negatives are projecting onto paper or a screen, it restores light and dark to their normal order.

**Algorithm**

1. The 8-bit image (type of uint8) was first loaded into matrix f
2. Transformation image will be saved as matrix g
3. 8-bit image with intensity values in the range 0 to 255
4. The formula for converting negatives:

g = 255 – f

**Logarithmic Transformation**

**Objective**

Logarithmic transformation is used to brighten the intensity of image. It is more often used to increase the detail of lower intensity values.

**Algorithm**

1. The image is loaded into a matrix as f
2. Transformed image will be saved as matrix l
3. Formula for logarithmic transformation:

l = c \* log(1+double(f))

1. c is a constant used to scale the range of log function to match input domain. As we are working with 8 bit image (type of uint8), so c = 255/log (1+255) is used for this case.

**Power Gamma Transformation**

**Objective:**

We know that the human eye perceives colour and luminance differently than the sensor on your smartphone or digital camera.  when twice the number of photons hit the sensor of a digital camera, it receives twice the signal (a linear relationship). However, that’s not how our human eyes work. Instead, we perceive double the amount of light as only a fraction brighter (a non-linear relationship)! Furthermore, our eyes are also much more sensitive to changes in dark tones than brighter tones (another non-linear relationship). To account for this, we can apply gamma correction, a translation between the sensitivity of our eyes and sensors of a camera.

**Algorithm:**

1. Image pixels intensities are scaled from the range [0, 255] to [0, 1.0].

2. Obtain output gamma corrected image by applying the following equation on each intensity:

O = I ^ (1 / G)

Where I is our input image and G is our gamma value.

3. The output image O is then scaled back to the range [0, 255].

Gamma values < 1 will shift the image towards the darker end of the spectrum while gamma values > 1 will make the image appear lighter. A gamma value of G=1 will have no effect on the input image:

We define our adjust\_gamma() function. This method requires a two parameter, image, which is the image we want to apply gamma correction to. A second value is our gamma value. In this case, we default gamma=1.0, but you should supply whatever value is necessary to obtain a decent looking corrected image. We represented images as NumPy arrays. All we did is build a table (i.e. dictionary) that maps the input [0, 255] pixel values to the output gamma corrected values by scale the pixel intensities to the range [0, 1.0], apply the transform, and then scale back to the range [0, 255]. OpenCV can then take this table and quickly determine the output value for a given pixel in O(1) time for the input image whose values are from [0,255].

**Conclusion:**

All the implementation the described modules are done using python and OpenCV. Future work is histogram shaping for both grayscale and coloured image.

**References:**

<https://en.wikipedia.org/wiki/Histogram_equalization>

<https://prateekvjoshi.com/2013/11/22/histogram-equalization-of-rgb-images/>

<https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/>

<https://stackoverflow.com/questions/32655686/histogram-matching-of-two-images-in-python-2-x>