Question 1 - Histogram Specification

November 23, 2018

1 Histogram Specification

We were tasked to estimate the transformation funtion of an image in such a way that the resulting histogram of the input image nearly approximates the histogram of a given image and the show the final image.

1.1 The code

We start by importing all libraries required for performing the filtering
1. cv2: OpenCV is a library of programming functions mainly aimed at real-time computer vision.
2. numpy: NumPy adds support for large, multi-dimensional arrays along with a large collection of functions to operate on these arrays.
3. matplotlib: This is the most widely used plotting library available for python.

```
In [7]: import cv2
        import numpy as np
        from matplotlib import pyplot as plt
        plt.rcParams['figure.figsize'] = [15, 15]
```

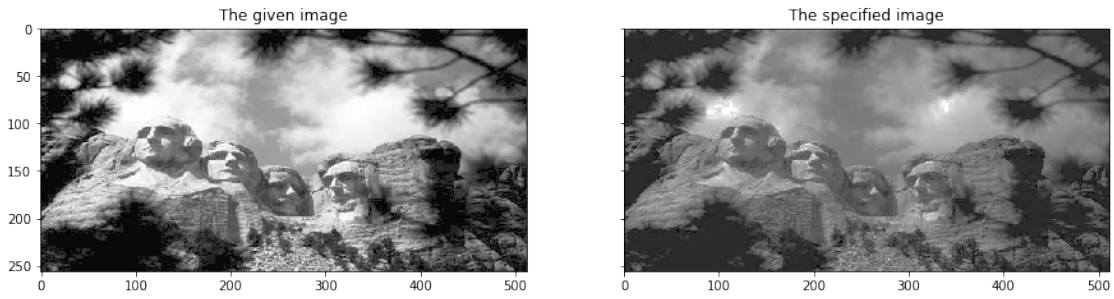
We read the input image and the specified histogram as grayscale. We initialize the output image.

```
In [4]: # Importing images as greyscale
        given = cv2.imread("/home/sarvesh/Projects/Image Processing/givenhist.jpg", 0)
        final = cv2.imread("/home/sarvesh/Projects/Image Processing/sphist.jpg", 0)

        # Initializing the output container
        new_image = given - given
        new_image.setflags(write=True)
```

Lets see both the provided images.

```
In [10]: f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
         ax1.imshow(given, cmap='gray')
         ax1.set_title('The given image')

         ax2.imshow(final, cmap='gray')
         ax2.set_title('The specified image')
```

```
plt.show()
```



To calculate the histogram of the provided images, we use `np.reshape` which converts a 2D matrix to a 1D array. These arrays are then passed to the `np.histogram` function which gives the histogram values and the bins respectively.

```
In [5]: # Reshape into column of intensities
        g_col = np.reshape(given, 256 * 512, order='C').astype(float)
        f_col = np.reshape(final, 256 * 512, order='C').astype(float)
        L = int(max(g_col) - min(g_col))

        # Histogram of the given array of intensities with 256 bins
        h_g, h_g_bins = np.histogram(g_col, bins=256)
        h_f, h_f_bins = np.histogram(f_col, bins=256)
```

To perform histogram specification, we need to calculate the CDF of both the images and map the values from input image to the specified image. If there are multiple mappings we choose the smallest one in the specified histogram.

We find the PDF and CDF of both the provided images below.

```
In [6]: # PDF of above histogram is calculated
        pdf_g = h_g / sum(h_g)
        pdf_f = h_f / sum(h_f)

        # CDF of the above PDF is calculated
        c_g = pdf_g
        c_f = pdf_f
        for i in range(1, 256):
            c_g[i] = c_g[i - 1] + pdf_g[i]
            c_f[i] = c_f[i - 1] + pdf_f[i]

        # Rounding of CDF
        c_g_r = c_g * (L)
        c_f_r = c_f * (L)
```

The following code maps the values using the method described above.

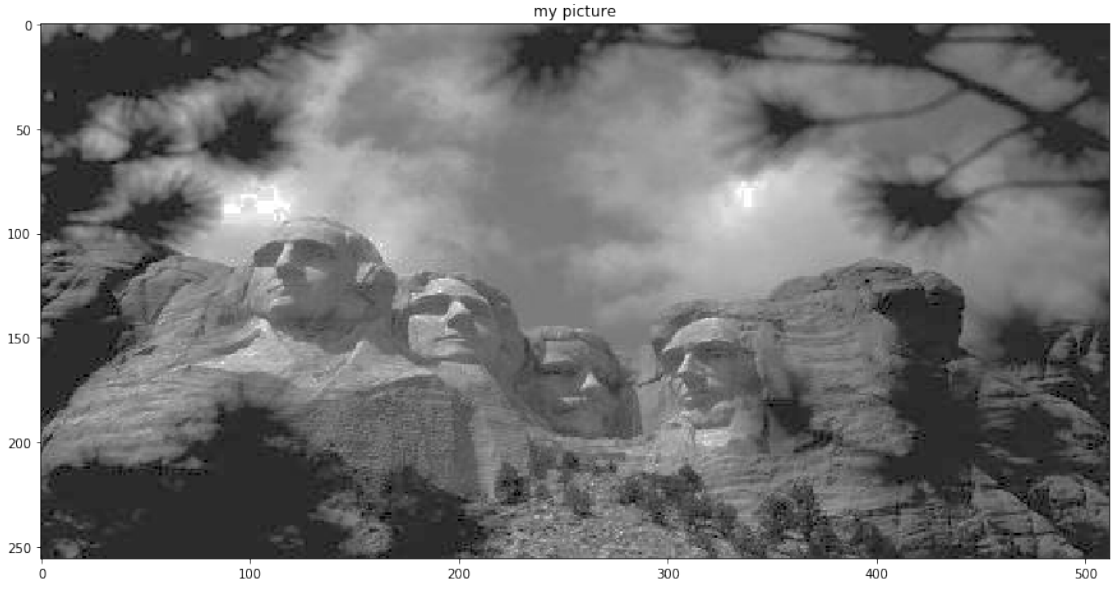
```
In [7]: temp = np.ones(256, order='C')
        index = np.ones(256, order='C')

        # Min difference mapping
        for i in range(0, 256):
            temp = abs(c_f_r - c_g_r[i] * np.ones(256, order='C'))
            index[i] = c_f_r[np.argmin(temp)]
```

Form histogram matched image by first histogram equalizing input image and then mapping every equalized pixel to the corresponding value, using the mappings stored in `index[]`.

```
In [8]: # New image construction
        for i in range(0, 256):
            for j in range(0, 512):
                if (given[i, j] == 0):
                    new_image[i, j] = 30
                else:
                    new_image[i, j] = index[given[i, j]]
```

1.2 Result

```
In [33]: plt.imshow(final, cmap='gray')
         plt.title('Final Output')
         plt.show()
```



1.3 Conclusion

The image was transformed to the specified histogram. The results were satisfactory.