Histogram Matching (Specification)

In the previous blog, we discussed Histogram Equalization that tries to produce an output image that has a uniform histogram. This approach is good but for some cases, this does not work well. One such case is when we have skewed image histogram i.e. large concentration of pixels at either end of greyscale.

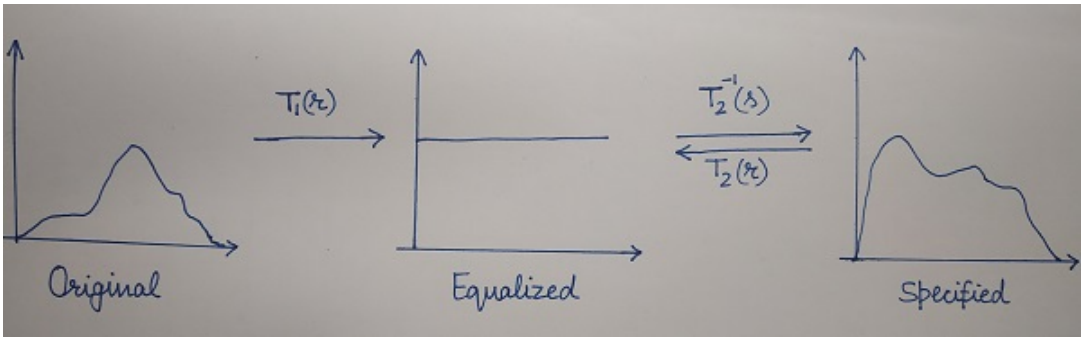
One reasonable approach is to manually specify the transformation function that preserves the general shape of the original histogram but has a smoother transition of intensity levels in the skewed areas.

So, in this blog, we will learn how to transform an image so that its histogram matches a specified histogram. Also known as histogram matching or histogram Specification.

Histogram Equalization is a special case of histogram matching where the specified histogram is uniformly distributed.

First let's understand the main idea behind histogram matching.

We will first equalize both original and specified histogram using the Histogram Equalization method. As we know that the transformation function is invertible, so by inverting we can get the mapping from original to specified histogram. The whole operation is shown in the below image

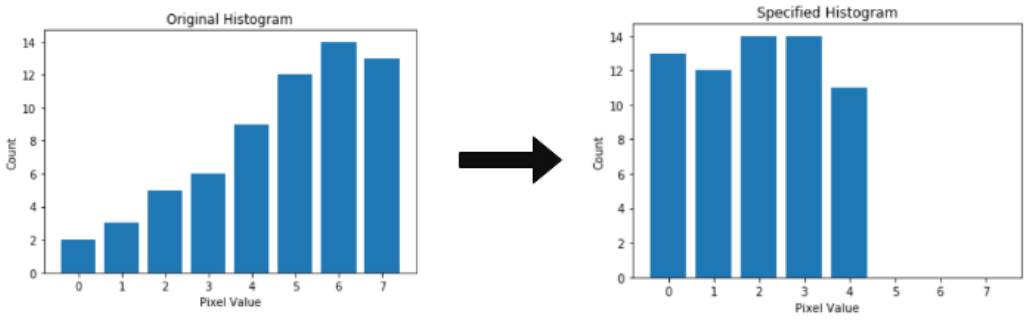


For example, suppose the pixel value 10 in the original image gets mapped to 20 in the equalized image. Then we will see what value in Specified image gets mapped to 20 in the equalized image and let's say that this value is 28. So, we can say that 10 in the original image gets mapped to 28 in the specified image.

Most of you might be thinking why both original and specified histogram on equalization converges to same uniform histogram.

This is true only if we assume continuous intensity values. But in reality, the intensity values are discrete thus both original and specified histograms may not map to the same histogram on equalization. That's why Histogram matching is not able to perfectly match the specified histogram.

Let's take an example where we want to match the original image with the specified image, both histograms are shown below.



Here, I am taking the original image from the histogram equalization blog. All the steps of equalization are explained in this blog. Here, I will only show the final table

Original Image Histogram Equalization

r_k	n_k	$P_r(r_k)$	s_k	Round
0	2	0.03	0.21	0
1	3	0.05	0.56	1
2	5	0.08	1.12	1
3	6	0.09	1.75	2
4	9	0.14	2.73	3
5	12	0.19	4.06	4
6	14	0.22	5.60	6
7	13	0.20	7.00	7

Specified Image Histogram Equalization

r_k	n_k	P_r	S_k	Round
0	13	0.20	1.40	1
1	12	0.19	2.73	3
2	14	0.22	4.27	4
3	14	0.22	5.81	6
4	11	0.17	7	7
5	0	0	7	7
6	0	0	7	7
7	0	0	7	7

After equalizing both the images, we need to perform a mapping from original to equalized to the specified image. For that, we need only the round columns of the original and specified image as shown below.

r_k	Round (Original)	Round (specified)	Map
0	0	1	0
1	1	3	0
2	1	4	0
3	2	6	1
4	3	7	1
5	4	7	2
6	6	7	3
7	7	7	4

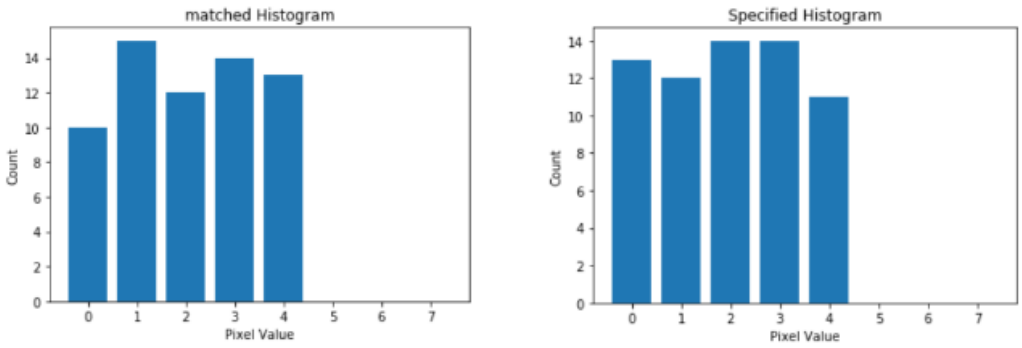
Pick one by one the values from the round column of the original image, find it in the round column of the specified image and note down the index. For example for 3 in the round original, we have 3 in the round specified column (with index 1) so we map it to 1.

If the value doesn't exist then find the index of its nearest one. For example for 0 in round original, 1 is the nearest in round specified column (with index 0) so we map it to 0.

If multiple nearest values exist then pick the one which is greater than the value. For example for 2 in the round original, there are 2 closest values in round specified i.e. 1 and 3 so we pick 3 (with index 1) so we map it to 1.

After obtaining the Map column, replace the values in the original image with the map values. This is the final result.

The matched histogram(shown on left) approximately matches with the specified histogram(shown on right) as shown below



Now, let's see how to perform Histogram matching using OpenCV-Python

Code

```
def find_nearest_above(my_array, target):
    diff = my_array - target
    mask = np.ma.less_equal(diff, -1)
    # We need to mask the negative differences
    # since we are looking for values above
```

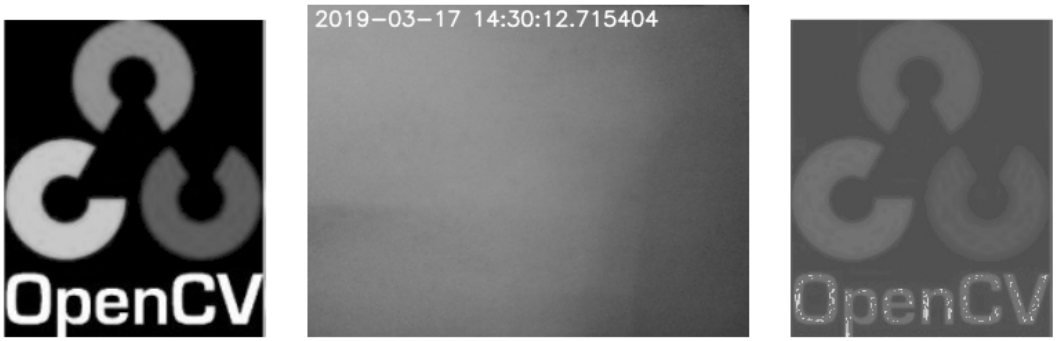
```
    if np.all(mask):
        c = np.abs(diff).argmin()
        return c # returns min index of the nearest if target is greater tha
    masked_diff = np.ma.masked_array(diff, mask)
    return masked_diff.argmin()
```

```
def hist_match(original, specified):

    oldshape = original.shape
    original = original.ravel()
    specified = specified.ravel()

    # get the set of unique pixel values and their corresponding indices an
    s_values, bin_idx, s_counts = np.unique(original, return_inverse=True,re
    t_values, t_counts = np.unique(specified, return_counts=True)

    # Calculate s_k for original image
    s_quantiles = np.cumsum(s_counts).astype(np.float64)
    s_quantiles /= s_quantiles[-1]

    # Calculate s_k for specified image
    t_quantiles = np.cumsum(t_counts).astype(np.float64)
    t_quantiles /= t_quantiles[-1]

    # Round the values
    sour = np.around(s_quantiles*255)
    temp = np.around(t_quantiles*255)

    # Map the rounded values
    b=[]
    for data in sour[:]:
        b.append(find_nearest_above(temp,data))
    b= np.array(b,dtype='uint8')

    return b[bin_idx].reshape(oldshape)
```

```
import cv2
import numpy as np

# Load the images in greyscale
original = cv2.imread('D:/downloads/opencv_logo1.PNG',0)
specified = cv2.imread('D:/downloads/dat.jpg',0)

# perform Histogram Matching
a = hist_match(original, specified)

# Display the image
cv2.imshow('a',np.array(a,dtype='uint8'))
cv2.imshow('a1',original)
cv2.imshow('a2',specified)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Note: Specified image can have different dimensions as compared to the original image.

The output looks like this



Original **Specified** **Matched**

Hope you enjoy reading.

If you have any doubt/suggestion please feel free to ask and I will do my best to help or improve myself. Good-bye until next time.

This entry was posted in Image Processing and tagged histogram equalization, histogram matching, histogram specification, histograms, Image histogram, opencv python, python on 10 Apr 2019 [https://theailearner.com/2019/04/10/histogram-matching-specification/] by kang & atul.