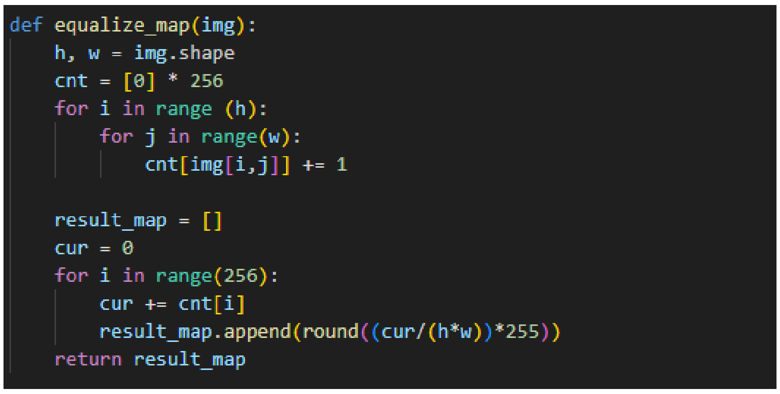
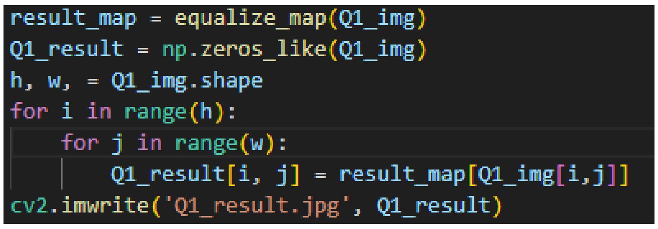
Method

1. Histogram Equalization



```python
def equalize_map(img):
    h, w = img.shape
    cnt = [0] * 256
    for i in range (h):
        for j in range(w):
            cnt[img[i,j]] += 1

    result_map = []
    cur = 0
    for i in range(256):
        cur += cnt[i]
        result_map.append(round((cur/(h*w))*255))
    return result_map
```

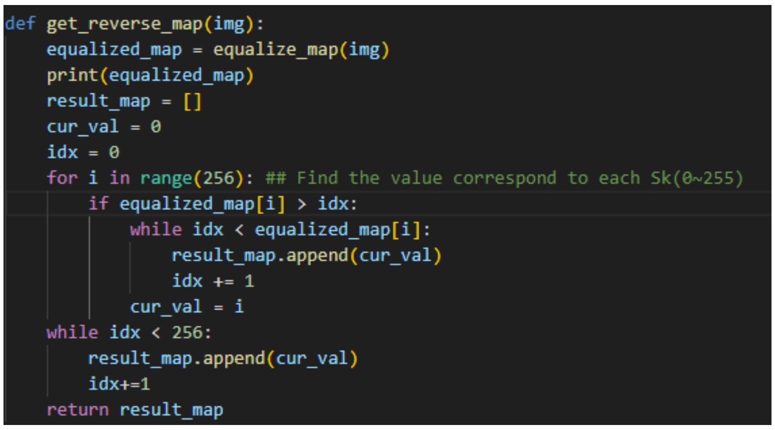
Define a function to compute the histogram and get the equalize map which map pixel value of the original image to new value. In the function, first count the number of pixels for each value (0~255). Then use the simple formula to compute its corresponding value (round up) for each original value.



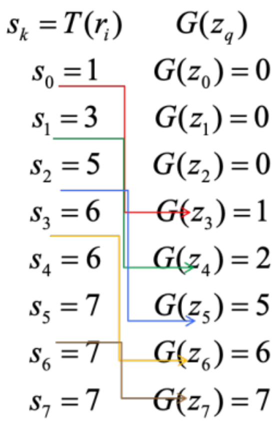
```python
result_map = equalize_map(Q1_img)
Q1_result = np.zeros_like(Q1_img)
h, w, = Q1_img.shape
for i in range(h):
    for j in range(w):
        Q1_result[i, j] = result_map[Q1_img[i,j]]
cv2.imwrite('Q1_result.jpg', Q1_result)
```

And we simply transform each pixel to its new value with the map and get the result.

2. Histogram Specification

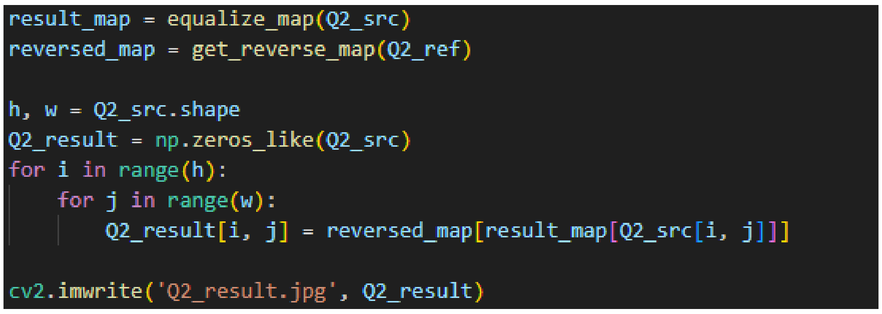


```python
def get_reverse_map(img):
    equalized_map = equalize_map(img)
    print(equalized_map)
    result_map = []
    cur_val = 0
    idx = 0
    for i in range(256): ## Find the value correspond to each Sk(0~255)
        if equalized_map[i] > idx:
            while idx < equalized_map[i]:
                result_map.append(cur_val)
                idx += 1
            cur_val = i
    while idx < 256:
        result_map.append(cur_val)
        idx+=1
    return result_map
```



Define a new function to map the equalize image's pixel value to the reference image, in order to get a similar distribution of histogram with the reference image. The parameter is the reference image.

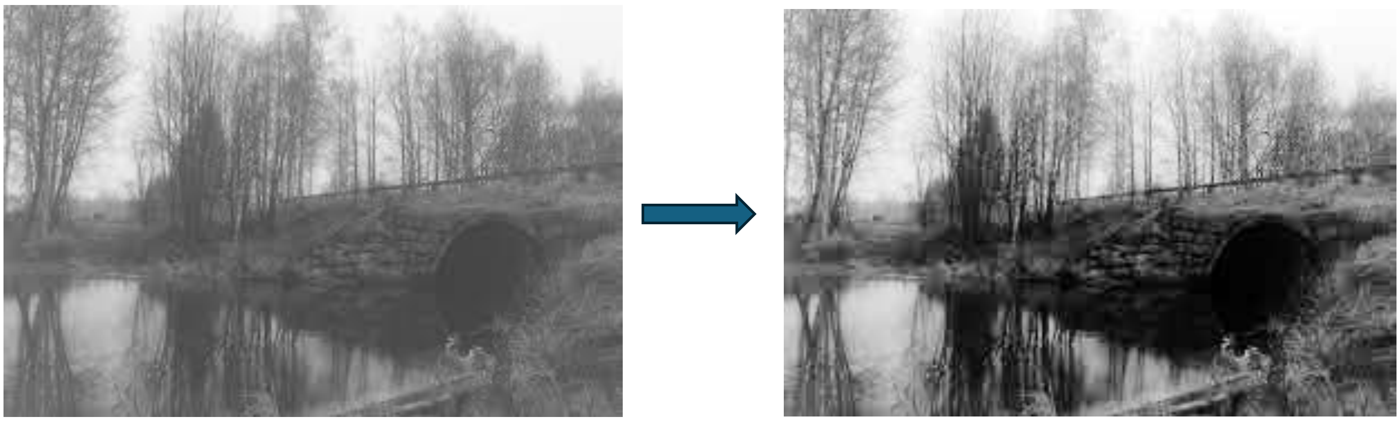
E.g. Equalized map for reference image= [0, 0, 0, 1, 2, 5, 6, 7]. What the function do is to get the mapping [0, 3, 4, 4, 4, 5, 6, 7]. We can know that if the equalization value of a pixel in original image is 3, it should be mapped to mapping [3] = 4.



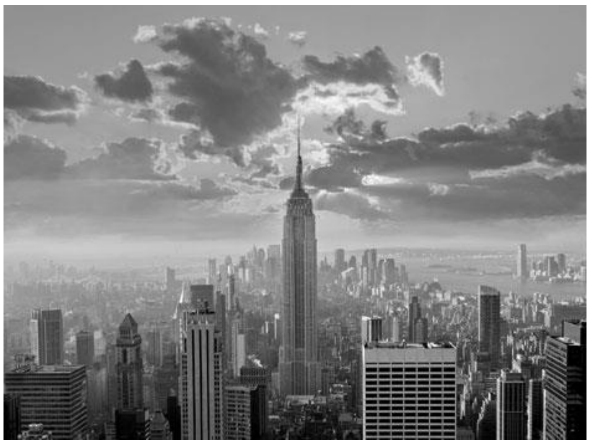
```
result_map = equalize_map(Q2_src)
reversed_map = get_reverse_map(Q2_ref)

h, w = Q2_src.shape
Q2_result = np.zeros_like(Q2_src)
for i in range(h):
    for j in range(w):
        Q2_result[i, j] = reversed_map[result_map[Q2_src[i, j]]]

cv2.imwrite('Q2_result.jpg', Q2_result)
```

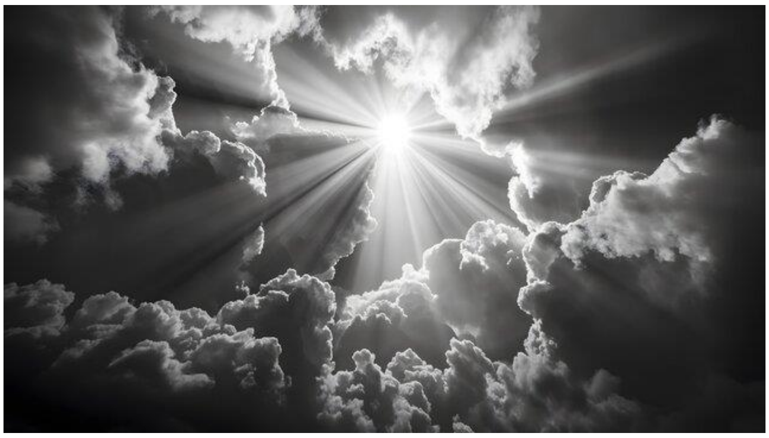
First, we get the equalize map for source image (the function is implemented in part 1) and get the reversed map with the reference image. Then we can compute the ultimate value for each pixel with equalized map of source image and reversed map of reference image.

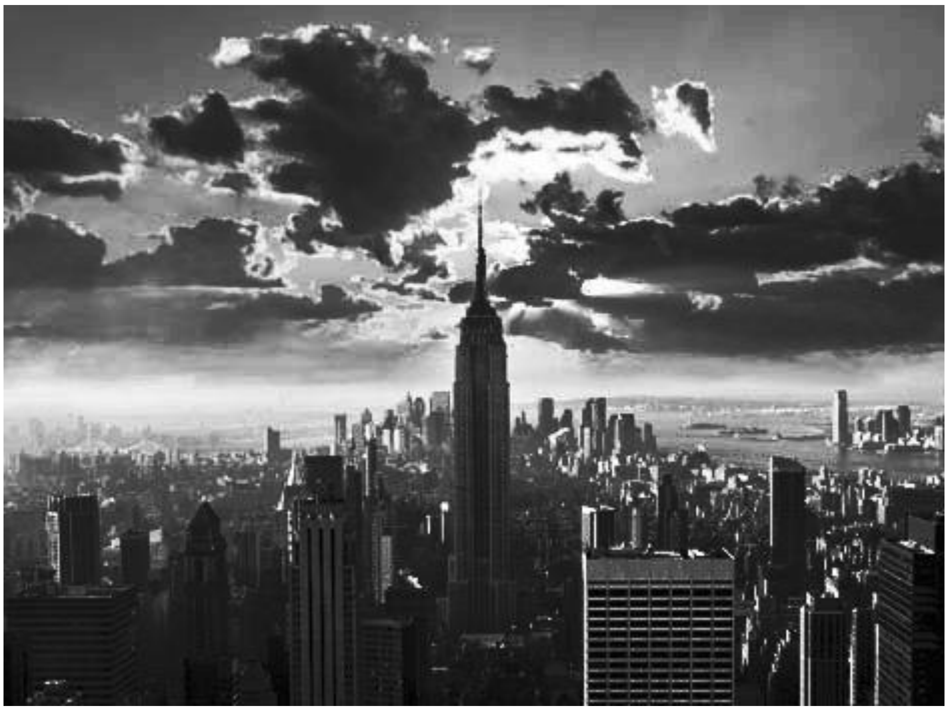
Result



With histogram equalization, we can transform a low-contrast image to a high-contrast image. Making details more visible for human eyes in transformed image.







As we can see, the original grayscale image contains a balanced range of intensity levels but lacks dramatic contrast. The selected reference image features a high contrast between bright and dark regions. After applying histogram specification, the source image adopts the intensity distribution of the reference image. The contrast of the picture in each part is significantly enhanced.

Feedback

The homework is very interesting, we can utilize the technique we've learned from the lecture to the image in our life. The result is quite distinguishable with the original image, using histogram equalization makes a common low contrast image to a high contrast image which is much more visible for our human eyes. Additionally, histogram specification provides the flexibility to adjust an image's contrast to match a desired reference. In this homework, we transform a city picture to adopt a high contrast of a sky image, and the result was quite impressive.