Computer Vision HW03 旁聽生 陳又蕾

Brief Description: This homework used Python, OpenCV, numpy to generate

(a) an original image and its histogram

(b) an image with intensity divided by 3 and its histogram

(c) an image after applying histogram equalization to the reduced-intensity image (b) and its histogram

Please use python ./main.py to run the program.

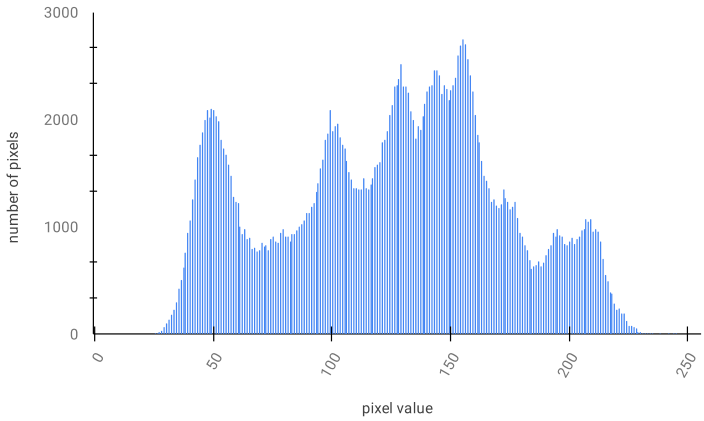
Histogram

All three parts of the homework uses this code fragment to generate the histogram of its respective input images

```
def histogram(img):
    count = np.zeros(256, np.int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            val = img[i][j]
            count[val] += 1
    return count
```

I create an array of zeros with length 256 (length of pixel values 0-255) first, so that when I loop through the binary image, I can constantly update the number of times that a certain pixel value appears in that array. The resulting array values is then returned from this function and saved to a csv file using np.savetxt in order to import into Google Sheets to produce a histogram.

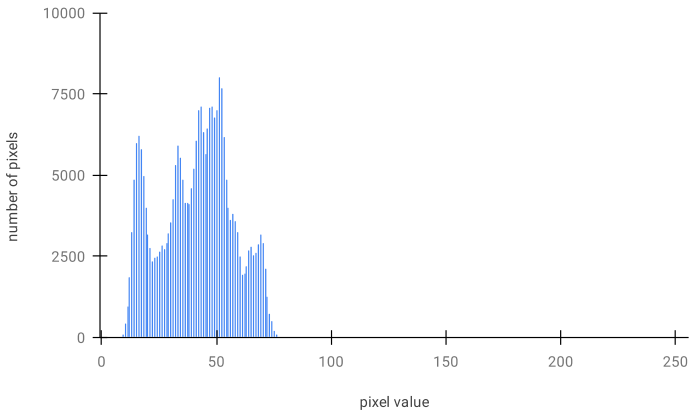
(a) an original image and its histogram




```
def original(img):
    np.savetxt("original_histogram.csv", histogram(img), delimiter=",")
    cv2.imwrite('original.bmp', img)
```

No modification is added to the input image for this problem. Please see the above section (**Histogram)** to understand how the histogram of this image is created

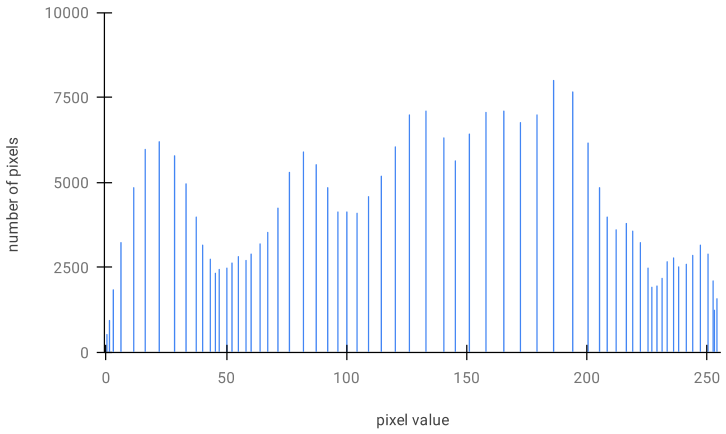
(b) an image with intensity divided by 3 and its histogram




```
def reduced_intensity(img):
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            img[i][j] /= 3
    np.savetxt("reduced_intensity_histogram.csv", histogram(img), delimiter=",")
    cv2.imwrite('reduced_intensity.bmp', img)
```

I looped through all the pixels of the input image and divided the pixel values by three in order to create this resulting image with reduced intensity. To learn more about how its histogram is created, please see the previous section (**Histogram).**

(c) an image after applying histogram equalization to the (b) and its histogram




```
def equalization(img):
    new_pixel_list = get_new_pixel_list(img);
    total_pixels = img.shape[0] * img.shape[1]
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            val = img[i][j]
            img[i][j] = 255 * np.sum(new_pixel_list[0 : val + 1]) / total_pixels
    np.savetxt("equalization_histogram.csv", histogram(img), delimiter=",")
    cv2.imwrite('equalization.bmp', img)
```

```
def get_new_pixel_list(img):
    count = np.zeros(256, np.int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            val = img[i][j]
            count[val] += 1
    return count;
```

I first created an array of size 256 to keep track of the number of pixels with intensity *j (0-255).* The algorithm used is similar to how I retrieve the pixel value counts for histograms (Please see section **Histogram** for more info). Next, in order to reach histogram equalization, I loop through every pixel of the input image, which is the resulting image from (b), and set the pixel with intensity *j* to its new pixel value using the following formula:

$$s_k = 255 \sum_{j=0}^{k} \frac{n_j}{n}$$

which sums up the number of pixels that has the intensity 0 to *j* and divides it by the total amount of pixels of the input image, and lastly, times 255.