FUDAN UNIVERSITY, SCHOOL OF DATA SCIENCE

Homework1

liyushuo 19307110230

March 24, 2022

1 PROBLEM1

1.1 Image contrast stretching

I constructed a **piecewise linear function**, it's specific form is:

$$f(x)\begin{cases}\frac{x}{2}, & if\ x<\frac{L}{3}\\ 2x-\frac{L}{2}, & if\ \frac{L}{3}<x<\frac{5L}{7}\\ \frac{x}{4}+\frac{3L}{4}, & if\ \frac{5L}{7}<x\end{cases}$$

And then I read in an image whose name is "daitu.jpg" and I changed it into grey image. For intensity of all pixels of the image, I use the function defined to compute new intensity values; and finally save the image with new intensity values. Our new image is "problem1.jpg".

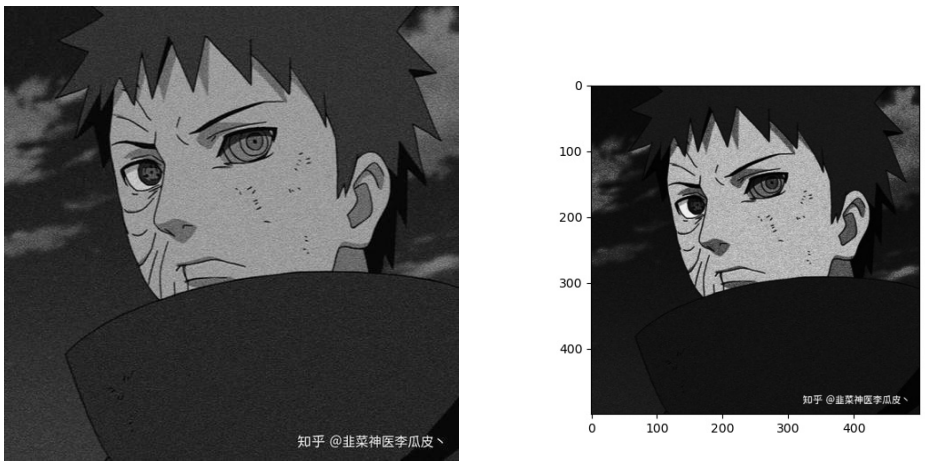

Figure 1.1: Images before change and after change

2 PROBLEM2

2.1 Implement n-dimensional joint histogram

I realized one n-dimensional joint histogram through the function nD-histogram. I use the numpy package to write the code. Our input of the function are data, dimension, nbins, pInMin, pInMax. The data is a matrix with m(the number of data) rows and n(the dimension of one piece of data) columns. The dimension is the dimension of the data, which is equal to n. Nbins is a list including the number of bins we want to divide of different dimensions, and we make the bin widths of the same dimension are uniform.
Our output is a ndarry whose shape is (N,1) where N is equal to the factorial of all nbins of different dimensions. Through this ndarry, we can store the histogram of n-dimensional data.
We read two grey images, reshape the arrays of them as our data and plot the two-dimensional histogram:

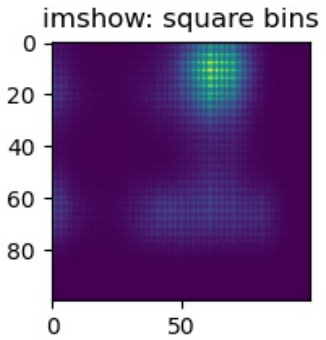

Figure 2.1: 2D-Histogram of two images

2.2 Efficient update of local histogram in local histogram processing

We implement computation of local histograms of an image using the efficient update of local histogram method introduced in local histogram processing.
We take a local image of k*k size. Using the nD-Histogram function we can easily compute the histogram of the local image. When the image moves, we can't compute every pixel of it absolutely which performs badly and cost hugely. To update efficiently, we first introduce a property that for disjoint image regions A and B, the histogram of A and B is equal to the histogram of A plus the histogram of B:

$$H(A \bigcup B) = H(A) + H(B)$$

We note that because only one row or column of the neighborhood changes in a one-pixel translation of the neighborhood, we update the histogram obtained in the previous location with a plus histogram of new row(column) and a subtraction histogram of old row(column).
For an image whether it's local or global, it's one dimension. Before we have implement a nD-Histogram function, with which we can compute the histogram of an image setting input parameter dimension equal to 1, pInMax equal to L(255), pInMin equal to 0.
We move our local image with a one-pixel step and with a "S" form direction which means repeating right, down, left, down, right.
In our example, we take k=125 which is a quarter of the width of the origin image.

3 PROBLEM3

3.1 Implement histogram equalization algorithm

We read an image and implement histogram equalization by mapping origin pixel value r to new pixel value s with following rules:

$$\begin{aligned} S_k &= T(r_k) \\ &= \sum_{j=0}^{k} p_r(r_j) \\ &= \sum_{j=0}^{k} \frac{n_j}{n} \\ k &= 0, 1, 2, ..., L-1 \end{aligned}$$

Our example is as follows:

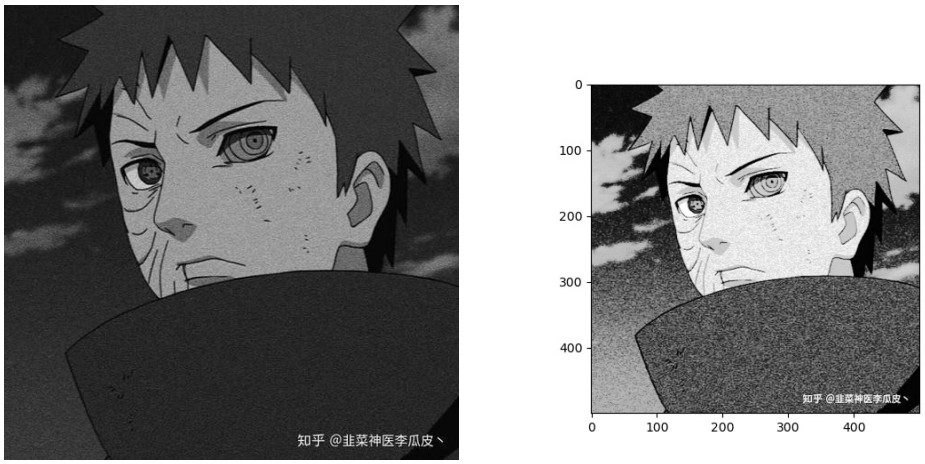

Figure 3.1: Images before and after histogram equalization

3.2 Local histogram equalization using efficient computation of local histogram

In problem2, we have achieved the computation of local histograms. Based on it, we set input parameter nbins equal to 255. Therefore, the output array of our function is indexed by pixel valus and stored the frequency corresponding to the pixel value. We sum the frequencys before the pixel value r for all pixel points, multipy constants, then we obtain the equalized pixel value and the equalized histogram.
In our example, we choose the size k of local image equal to 125(500/4), which is convenient to shift our local image.
To make our result more continous, we pick the center pixel of a local image whose new pixel value is obtained by equalizing the local image histogram. However, we still have left, right, upper, down four space with k/2 width which haven't been equalized. Except for the left upper, right upper, left down, right down four square area we directly use corresponding local image to equalize them, other space we equalize them by rows or columns.
Here are our final result:

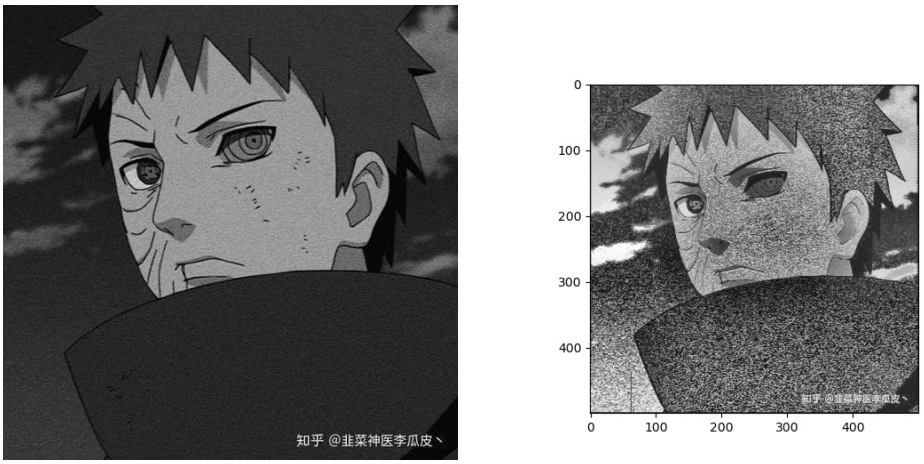

Figure 3.2: Images before and after local histogram equalization