Digital Image Processing

Homework 2

Tongji University
School of Software Engineering

ID:	1754060
Name:	Zhe Zhang
Adviser:	Prof. Qingjiang Shi
Time:	Tue. 2-4 [1-17]
Email:	doubleZ0108@163.com

Contents

1 Problem 1: Denoising for Astrophotography

1.1 Running average of the frames without frame alignment

1.1.1 Overall Thought of the Algorithm

According to the rules

$$\begin{aligned} f^1_{\text{average}} &= f^1 \\ f^t_{\text{average}} &= \tfrac{t-1}{t} f^{t-1}_{\text{average}} + \tfrac{1}{t} f^t, t = 2, 3, \dots \end{aligned} \tag{1}$$

I found that $f^{30}_{average}$ is based on $f^t_{average}$ and f^t

- f^t**:** the t frame of the video
- $f^t_{average}$**:** running average of the frames f^t in the video without frame alignment, and it is given by a recursive formula
  - **Recursive termination condition:** the running average of the first frame is the first frame itself(because it haven't running, so we cannot calculate the average yet)
  - **Recursive expression:** $f^t_{average}$ is calcluate by weighted summation of $f^{t-1}_{average}$ and f^t, the weights is $\frac{t-1}{t}$ and $\frac{1}{t}$

1.1.2 Matlab Code

- Loop version

Listing 1: Loop version

```
vidobj = VideoReader([video_path,video_name]);
frame_average_last = im2double(read(vidobj, 1)); % first frame of the video
for t=2:30
    frame_now = im2double(read(vidobj, t));
    % weighted summation
    frame_average = (t-1)/t * frame_average_last + 1/t * frame_now;
    frame_average_last = frame_average;     % update the fresh frame
end
```

- Recursive version

Listing 2: Recursive version

```
vidobj = VideoReader([video_path,video_name]);

result = f_a(30, vidobj);
imshow(result);

function frame = f(t, vidobj)
% return the tth frame of the video input

frame = im2double(read(vidobj, t));

end

function frame = f_a(t, vidobj)
% return the tth running average of the frames f_t
% without frame alignment

if t==1
    frame = f(1, vidobj);
else
    frame = (t-1)/t * f_a(t-1, vidobj) + 1/t * f(t, vidobj);
end

end
```

1.1.3 Experimental Result

(a) sky1 original (b) sky1 average

Figure 1: Comparison of average to sky1

(a) sky2 original (b) sky2 average

Figure 2: Comparison of average to sky2

1.1.4 Experimental Effect

From the images above we can see clearly that there are many noise in the origin images, but after averaging of each images without frame alignment, there are few noise.

But, on the other hand, the sharp features are also blurred by the averaging opera-

tion. Stars become light tracks and the sharp detail of the moon has been blurry. The modified imagines don't look clear as before visually.

1.2 Running average of the frames with frame alignment

1.2.1 Overall Thought of the Algorithm

According to the rules

$$\begin{aligned} f^1_{\text{average}} &= f^1 \\ f^t_{\text{average}} &= \frac{t-1}{t} f^{t-1}_{\text{average}} + \frac{1}{t} Align(f^t, f^{t-1}_{average}), t = 2, 3, \dots \end{aligned} \tag{2}$$

- $Align(f, g)$**:** aligns frames f and g by minimizing the mean squared difference over a set of horizontal and vertical shifts.

The algorithm calculates the mean squared differences over a set of horizontal and vertical shifts between the averaged frame and the shifted frame, then the frame which has the smallest mean squared differences among the averaged frame will be chosen.

1.2.2 Matlab Code

- shift the image

Listing 3: imshift

```
function frameTform = imshift(frame, dx, dy)
% shift the img where
% move right dx
% move up dy

A = [1 0 0; 0 1 0; dx dy 1];
tform = maketform('affine', A);
[height, width, channels] = size(frame);
frameTform = imtransform(frame, tform, 'bilinear', ...
                         'XData', [1 width], ...
                         'YData', [1 height], ...
                         'FillValues', zeros(channels, 1));
```

- calculate Align(f, g)

Listing 4: Align(f, g)

```
function min_msd_frame = Align(f, g)
% aligns frames f and g by minimizing the mean squared difference
% over a set of horizontal and vertical shifts.

```

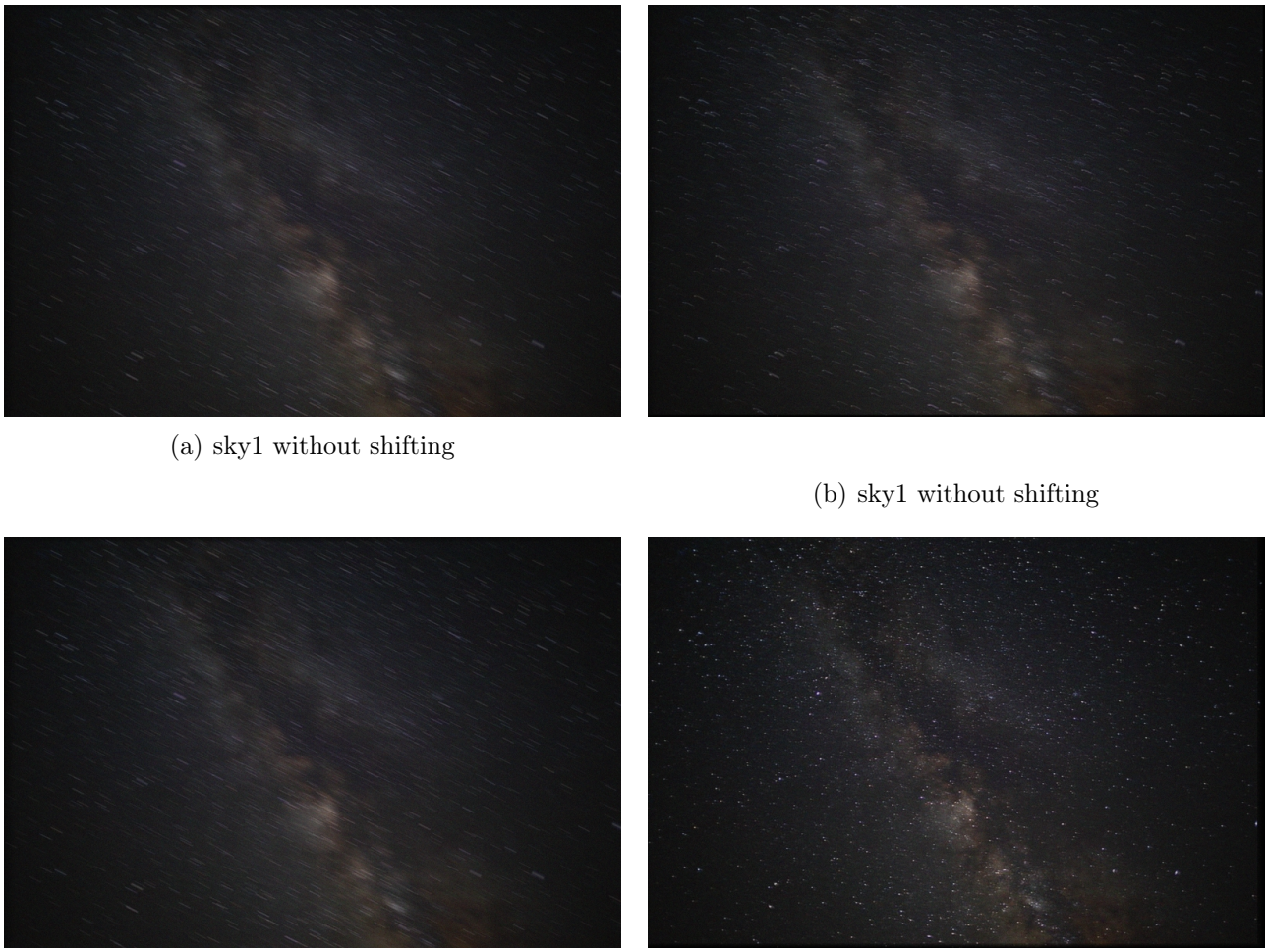
```
[M,N,type] = size(f);

% initialize min_msd_frame with the origin frame
diff = f - g;
min_msd = sum(sum(diff .^ 2)) / (M * N);
min_msd_frame = f;
min_index = [0, 0];

n = 10;     % user-defined

% loop the shift choices
% choose the smallest mean squared differences among the averaged frame
for dx=-n:n
    for dy=-n:n

        shifted_f = imshift(f, dx, dy);
        [M,N,type] = size(shifted_f);

        % crop the shifted frame
        rowl = 1; rowr = M + dy;
        colt = dx + 1; colb = N;
        if rowr > M, rowr = M; end
        if colt < 1, colt = 1; end

        sub_f = shifted_f(rowl:rowr, colt:colb, type);
        sub_g = g(rowl:rowr, colt:colb, type);

        % calculate the mean square difference
        diff = sub_f - sub_g;
        [M,N,type] = size(sub_f);
        msd = sum(sum(diff .^ 2)) / (M * N);

        % update to the smallest one
        if msd < min_msd
            min_msd = msd;
            min_msd_frame = shifted_f;
            min_index = [dx, dy];
        end

    end
end

end
```

- calculate running average of the frames without frame alignment

Listing 5: average with alignment

```
vidobj = VideoReader([video_path,video_name]);

frame_average_last = im2double(read(vidobj, 1));
for t=2:30
    t
    frame_now = im2double(read(vidobj, t));
    frame_average = (t-1)/t * frame_average_last + 1/t * Align(frame_now,
        frame_average_last);
    frame_average_last = frame_average;
end
```

1.2.3 Experimental Result



(a) sky1 without shifting

(b) sky1 without shifting

(c) sky1 without shifting

(d) sky1 shift with n=10

Figure 3: Comparison of shifting to sky1

(a) sky2 without shifting

(b) sky2 shift with n=3

(c) sky2 without shifting

(d) sky2 shift with n=10

Figure 4: Comparison of shiftint to sky1

1.2.4 Experimental Effect

From the images above we can see clearly that this algorithm can not only denoising, but can also reserve the sharp detail of origin image.

The algorithm use the shifting frames in which the features are very close to the last modified frame to average the noise.

But during my experiment, I still have to question. Here, I want to state my own viewpoint:

- **Firstly**, the algorithm is inefficient, because it should calculate all the shifted frame to

get the smallest mean squared difference, hence, it hold high time complexity. But during my experiment, I console all the b̈est choiceẅhich mean the frame get the smallest Align(). They are listed here:

t	dx	dy
1	0	0
2	0	0
3	0	1
4	0	1
5	-1	2
6	-1	2
7	-1	3
8	-1	3
9	-2	4
10	-2	4
11	-2	5
12	-2	5
13	-3	6
14	-3	6
15	-3	7

t	dx	dy
16	-3	7
17	-4	8
18	-4	8
19	-4	9
20	-4	9
21	-5	10
22	-5	10
23	-5	10
24	-5	10
25	-5	10
26	-6	10
27	-6	10
28	-6	10
29	-6	10
30	-6	10

  - **t**: current frame
  - **dx**: x offset of the shifted frame which hold the smallest mean squared differences
  - **dy**: y offset of the shifted frame which hold the smallest mean squared differences
  - We can find that dx is gradually decrease, and dy increase. Hence, I consider that **the best choice of next frame hold the smaller ds and larger dy of the sky2, we can store last offset of x and y, and loop before the storage value. It will reduce a lot of calculations**.

- **Secondly**, the result images still hold some details blurred. In my own viewpoint, I guess it may because this algorithm is actually still an operation to reduce the moised by averaging the frames. On the condition that we use averaging operation to reduce noises, it will make the frames blurred.

2 Problem2: Nighttime Road Contrast Enhancement

2.1 Histogram of the original images grayscale values

2.1.1 Overall Thought of the Algorithm

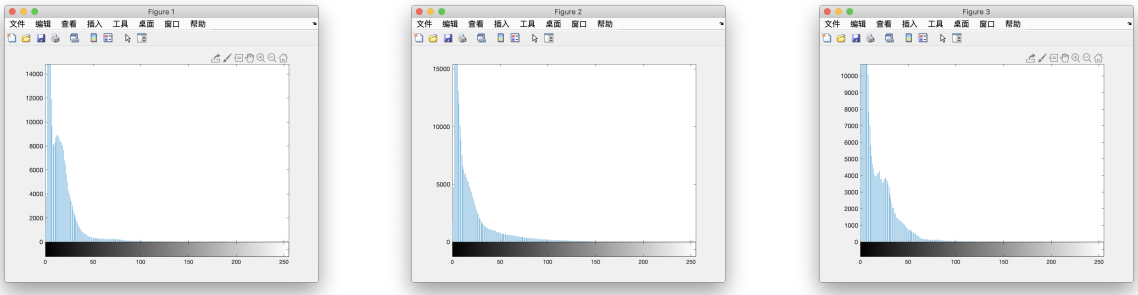
Use Library Functions — imhist to generate the histogram of the original images grayscale values

2.1.2 Matlab Code

Listing 6: generate histogram

```
img_path = 'Resources/';
imgs_name = ['hw1_dark_road_1.jpg';'hw1_dark_road_2.jpg';'hw1_dark_road_3.jpg'];
[M,N] = size(imgs_name);
for k=1:M
    img_name = imgs_name(k,:);
    I = imread([img_path, img_name]);
    figure;
    imhist(I);
end
```

2.1.3 Experimental Result



(a) road1 histogram (b) road2 histogram (c) road3 histogram

Figure 5: Histogram of the original images grayscale values

2.1.4 Experimental Effect

From the three figures above, we can see clearly that almost all pixels are concentrated in low grayscale value. The shape of the histogram look like a steep mountain.

We can also clearly see from the original picture that it looks dark, and the grayscale value of black color is closed to 0.

2.2 Global histogram equalization

2.2.1 Overall Thought of the Algorithm

According to the forumla

$$s = (L-1)\sum_{j=0}^{k}\frac{n_j}{M*N} \tag{3}$$

- L: the total gray level
- $M*N$: the image size
- n_j: the value of the jth item in the histogram

I use **incremental** idea to calculate n_j in each loop, it will reduce time complexity and amount of calculation significantly.

2.2.2 Matlab Code

- Loop version

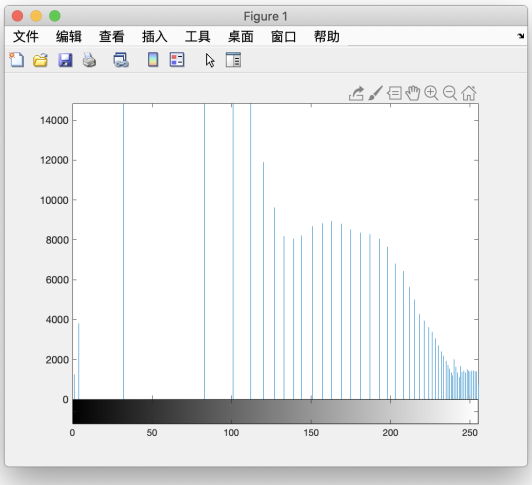
Listing 7: my histogram equalization

```
function J = myhisteq(I)
% global histogram equalization

[M,N] = size(I);
s = zeros(1, 256);
[counts, pixels] = imhist(I);   % [Number of pixels per pixel | Pixel value
    (0~255)]

sum_count = 0;
for k=0:255
    sum_count = sum_count + counts(k + 1, 1);   % use incremental idea
    s(1,k+1) = round(255/(M*N)*sum_count);
end

% convert the grayscale in the original image to the grayscale in the new
    image
J = s(1,I+1);

```

```
17 % convert the shape to origin shape
18 J = uint8(reshape(J, [M, N]));
19
20 end
```

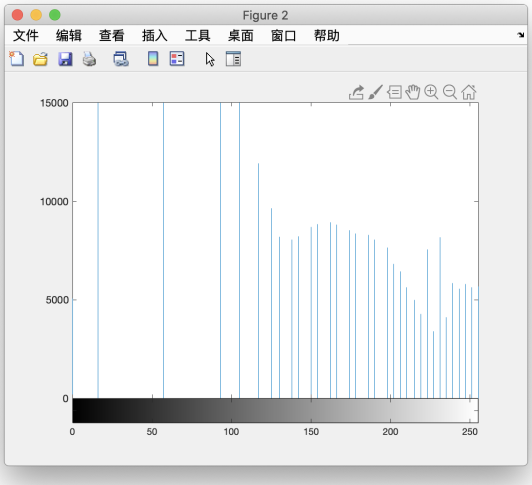
2.2.3 Experimental Result

(a) road1 after myhisteq



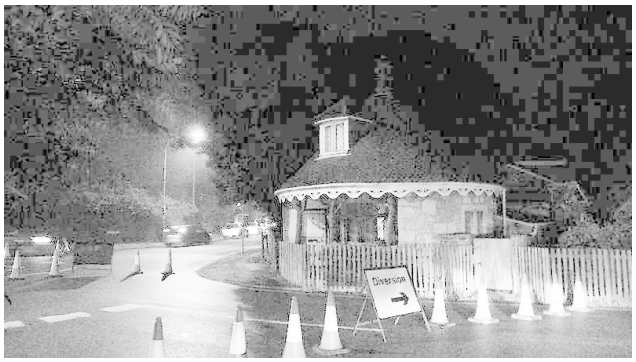
(b) histogram of road1 after myhisteq

road1 after histeq
(c) 3

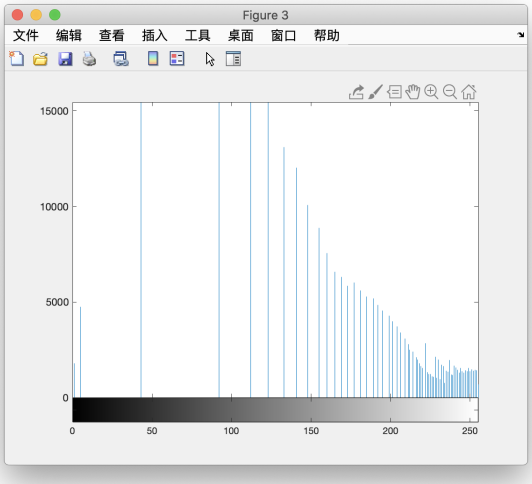


(d) histogram of road1 after histeq

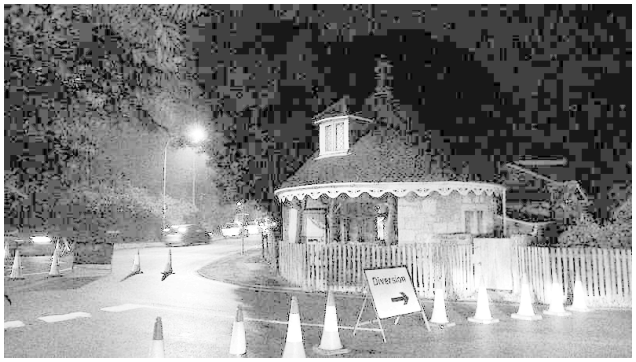
Figure 6: Comparison of myhisteq and histeq of road1



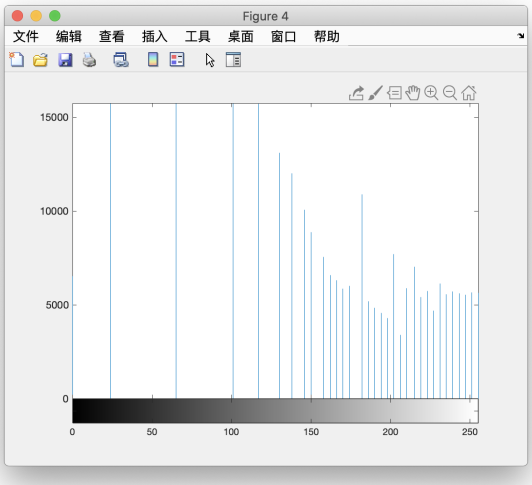
(a) road2 after myhisteq



(b) histogram of road2 after myhisteq



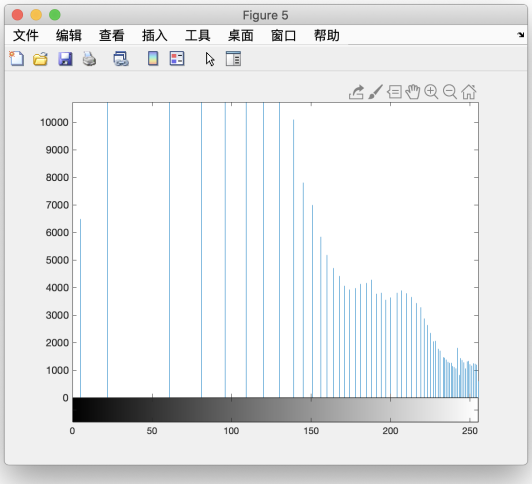
(c) road2 after histeq



(d) histogram of road2 after histeq

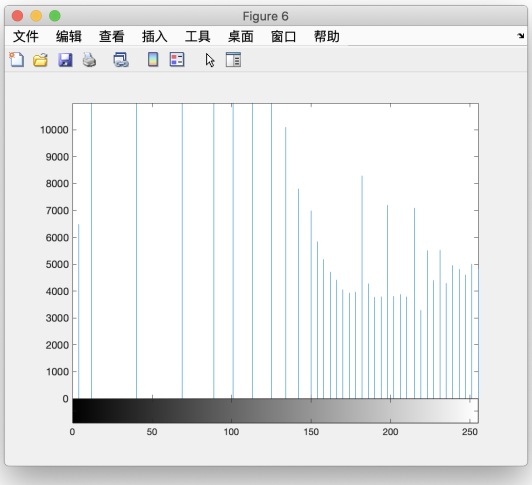
Figure 7: Comparison of myhisteq and histeq of road2

(a) road3 after myhisteq



(b) histogram of road3 after myhisteq

(c) road3 after histeq



(d) histogram of road3 after histeq

Figure 8: Comparison of myhisteq and histeq of road3

2.2.4 Experimental Effect

From the Experimental Result, we can find that that the original darker images become grayer. And the histograms of modified images hold a wider grayscale range, there are more pixels have high grayscales than before.

We can see more details in the modified images because it has stronger constrast. For

example, we can see the junction line between the buildings and a more clearly sky.

But on the other hand, some details become blurred after histeq. For example, the road sign in the first image become blurred so that we cannot see the word on it.

Also, from the comparison of Library Function histeq and myhisteq, we can find that Library Function complete more uniform dispersion but the figure of myhisteq getting smaller and smaller. And from the modified images themselves we can clear see that the image after histogram equalization of Library Function are more visually than after my personal histogram equalization.

2.3 Locally adaptive histogram equalization

2.3.1 Overall Thought of the Algorithm

Use Library Function — 'adapthisteq' to apply locally adaptive histogram equalization.

And about this function, I can customize two parameters:

- *NumTiles* : Two-element vector of positive integers: [M N]. [M N] specifies the number of tile rows and columns. Both M and N must be at least 2. The total number of image tiles is equal to M*N.
- *ClipLimit*: Real scalar from 0 to 1. 'ClipLimit' limits contrast enhancement. Higher numbers result in more contrast.

2.3.2 Matlab Code

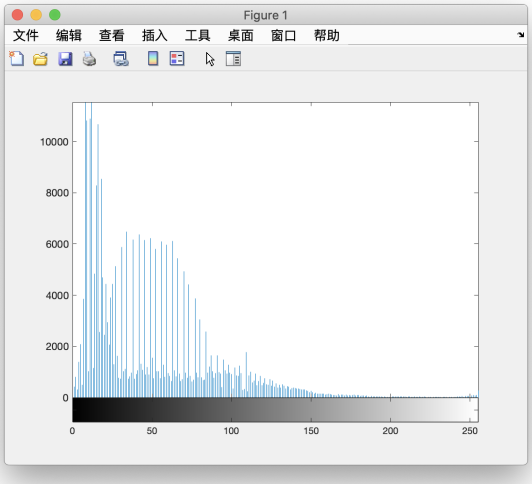
Listing 8: adaptive histogram equalization

```
for k=1:M
    img_name = imgs_name(k,:);
    I = imread([img_path, img_name]);

    % adaptive histogram equalization
    numtiles = 16; cliplimit = 0.018;
    J = adapthisteq(I, ...
                    'NumTiles',[numtiles, numtiles], ...
                    'ClipLimit', cliplimit);
    imwrite(J, [result_path, 'adapthisteq_16_16_0.018_', num2str(k), '.png']);
end
```

2.3.3 Experimental Result

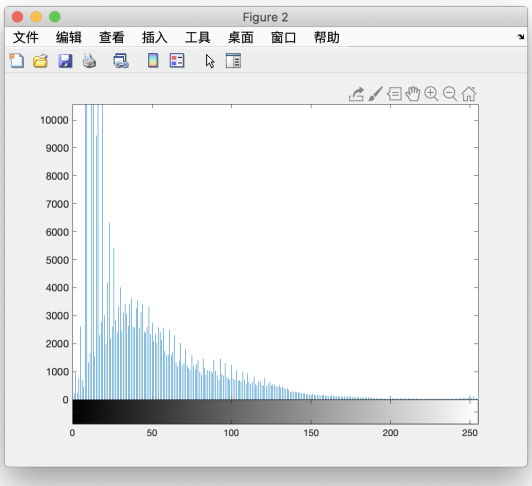
- Firstly, I fixed the ClipLimit value with 0.01, adjusted NumTiles values from 2*2 to 64*64, and observed the visual effect of modified images and the corresponding histograms.

(a) NumTiles: 2*2



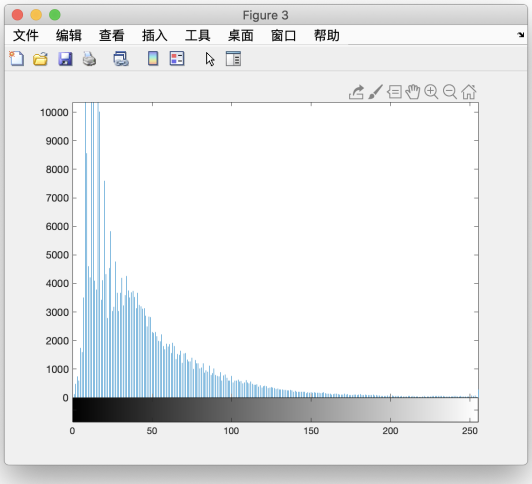
(b) corresponding histogram

(c) NumTiles: 4*4



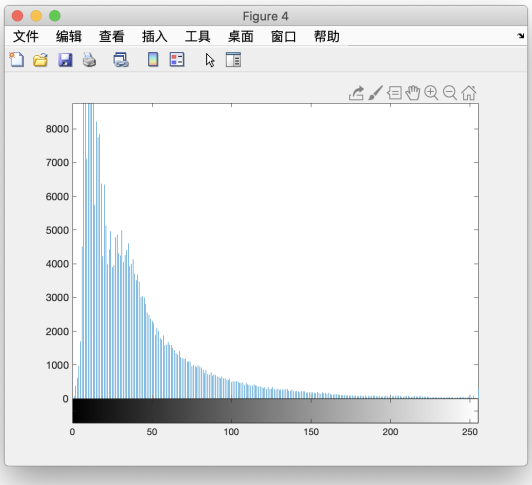
(d) corresponding histogram

(e) NumTiles: 8*8



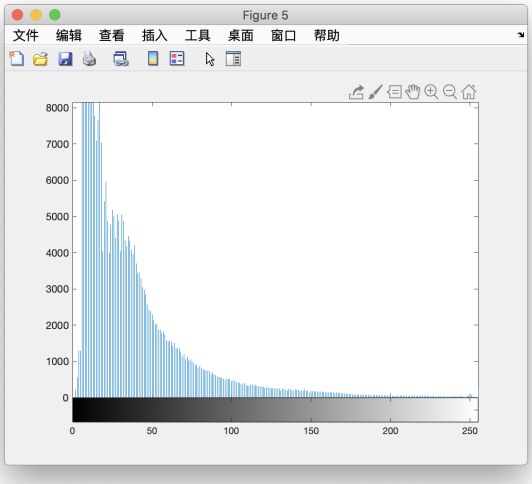
(f) corresponding histogram

(g) NumTiles: 16*16



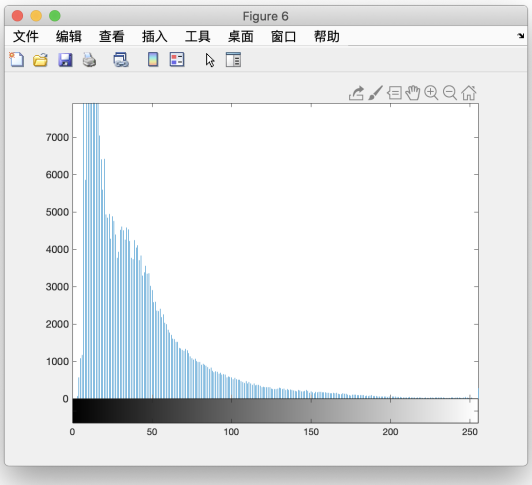
(h) corresponding histogram

(i) NumTiles: 32*32



(j) corresponding histogram

(k) NumTiles: 64*64



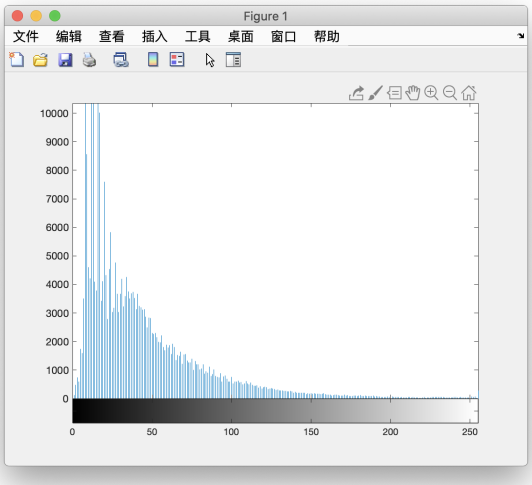
(l) corresponding histogram

Figure 9: The role of NileTiles

From the modified images we can see clearly that when the clipping limit keep stable, the more titles we choose, the more balanced the histogram will be.
From the corresponding histograms we can see that if the $NumTiles$ is small, the pixels will bread in a wider range. But with the $NumTiles$ grow, the specific details(details hidden in black background). Hence, the value in black area is obviously higher than other area.

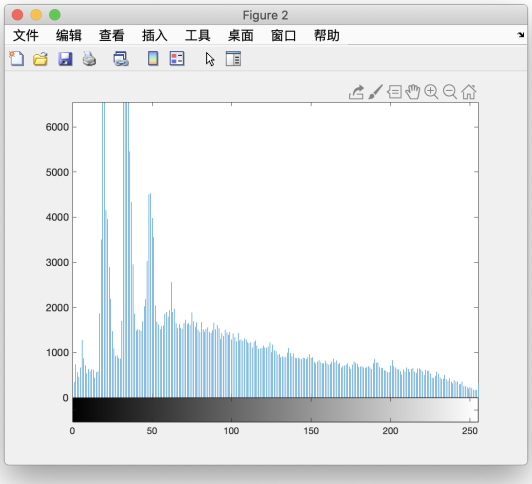
- Secondly, I fixed the *NumTiles* values with 16*16, and adjusted *ClipLimit* values from 0.01 to 0.5, also observed the visual effect of modified images and the corresponding histograms.

(a) ClipLimit: 0.01



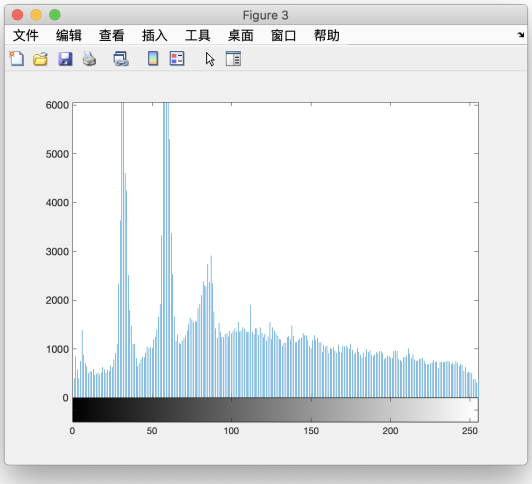
(b) corresponding histogram

(c) ClipLimit: 0.05

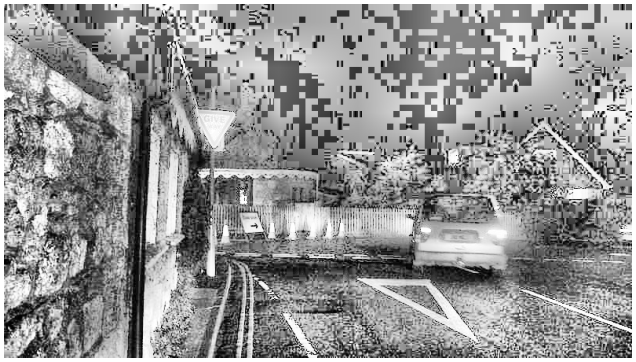


(d) corresponding histogram

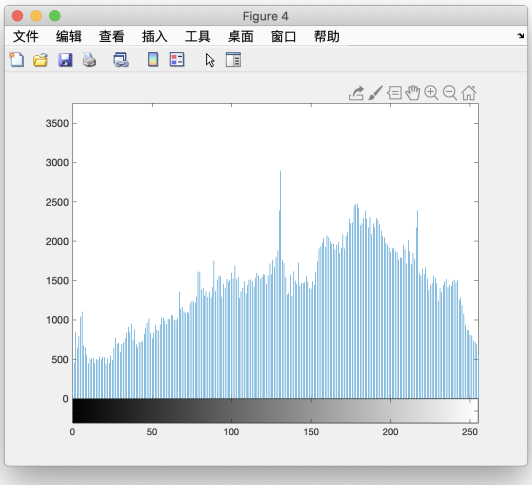
(e) ClipLimit: 0.1



(f) corresponding histogram



(g) ClipLimit: 0.5



(h) corresponding histogram

Figure 10: The role of ClipLimit

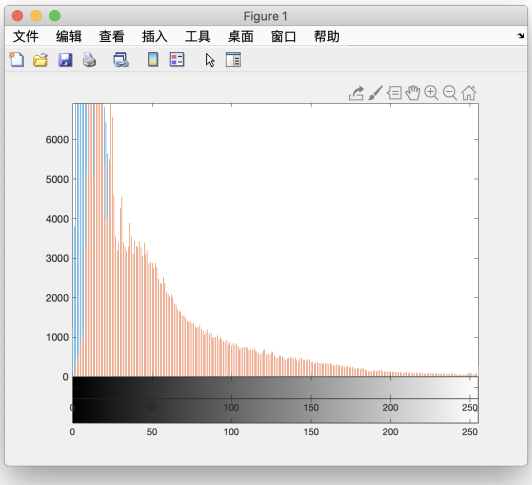
From the modified images and corresponding histograms we can see clearly that the higher the clipping limit is, the stronger the constrast of modified imagine will be and its histogram will become more balanced.
However, the higher clipping limit don't always mean a good choice. When *ClipLimit* grow to 0.1 or even higher, we can see many `Tileness` visually, and from the histogram we can also get the information, the pixels bread so broad that the original black part

turns white in order to make the histogram more equalized.

2.3.4 Experimental Effect

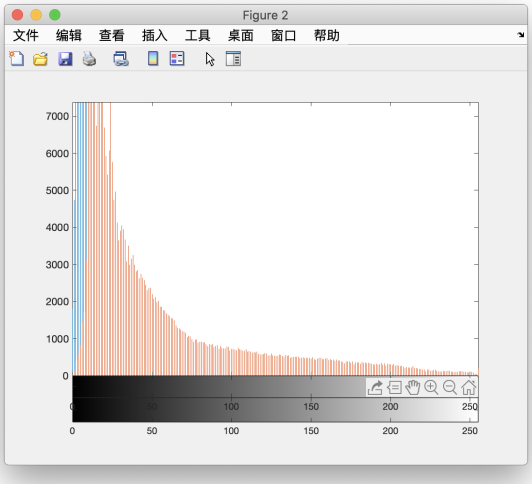
At last, I choose $NumTiles = 16 * 16$ and $ClipLimit = 0.018$ to apply locally adaptive histogram equalization. The result are as followed:

(a) road1 after adaptive histogram equalization



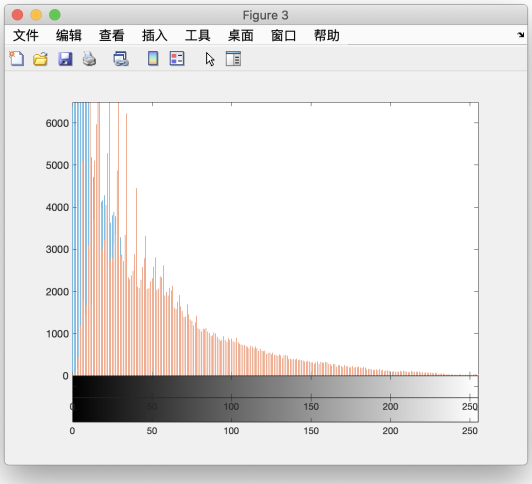
(b) corresponding histogram

(c) road2 after adaptive histogram equalization



(d) corresponding histogram

(e) road3 after adaptive histogram equalization



(f) corresponding histogram

Figure 11: Locally adaptive histogram equalization with NumTiles: 16*6 ClipLimit: 0.018

The orange figure is histograms after

$$adapthisteq(I,'NumTiles',[numtiles,numtiles],'ClipLimit',cliplimit) \tag{4}$$

and blue one is histograms of origin images.

We can have a straight comparison visually that we can get more information and details from the modified images. In road1, we can see street sign behind; in road2, the cars and roadblocks in the distance are very clear; in road3, trees by the road and windows on the building are all visually.

2.4 γ - nonlinearity mapping to each image to perform contrast enhancement

2.4.1 Overall Thought of the Algorithm

Build γ function model and try different coefficients. The function model is:

$$J = c * I^{\gamma} \tag{5}$$

And I use c to keep the coordinate scale consistent

$$c = \frac{255}{255^{\gamma}} \tag{6}$$

Then I draw different image curves and compare different modified images to perform contrast enhancement.

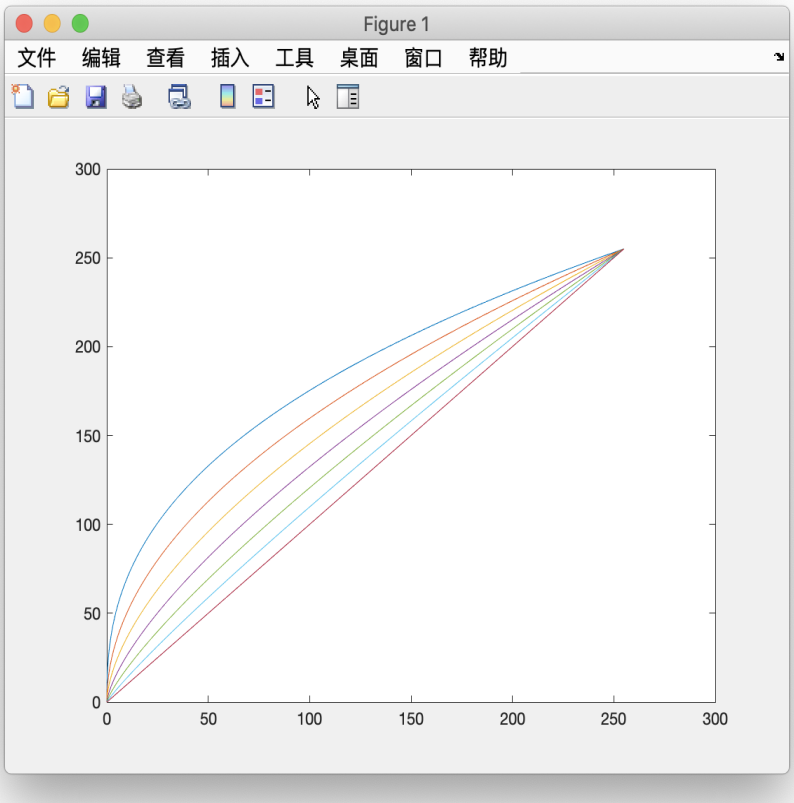


Figure 12: Function image corresponding to different gamma values

2.4.2 Matlab Code

Listing 9: Gamma Mapping

```
for k=1:M
    img_name = imgs_name(k,:);
    I = imread([img_path, img_name]);

    for gamma=0.4:0.1:1
        J = GammaNonlinearityMapping(I, gamma);
    end
end


function J = GammaNonlinearityMapping(I, gamma)
% use gamma-onlinearity mapping to perform contrast enhancement

% keep the coordinate scale consistent
c = 255/(255^gamma);

% gamma transformation
J = uint8(c*double(I).^gamma);

end
```

2.4.3 Experimental Result

(a) road1 $\gamma = 0.5$

(b) road1 $\gamma = 0.6$

(c) road1 $\gamma = 0.7$

(d) road1 $\gamma = 0.8$

(e) road1 $\gamma = 0.9$

(f) road1 $\gamma = 1$

Figure 13: γ mapping of road1

(a) road2 $\gamma = 0.5$

(b) road2 $\gamma = 0.6$

(c) road2 $\gamma = 0.7$

(d) road2 $\gamma = 0.8$

(e) road2 $\gamma = 0.9$

(f) road2 $\gamma = 1$

Figure 14: γ mapping of road2

(a) road3 $\gamma = 0.5$

(b) road3 $\gamma = 0.6$

(c) road3 $\gamma = 0.7$

(d) road3 $\gamma = 0.8$

(e) road3 $\gamma = 0.9$

(f) road3 $\gamma = 1$

Figure 15: γ mapping of road3

2.4.4 Experimental Effect

From the result images, we can find what we make can prove what we learned in class correctly. Lower γ(less than 1) will expand dark pixels to a more broad pixels area.

When $\gamma = 0.7$, I think it has an appealing visual effect and we can get most of the information we want to observe.