HW1

夏浩 19307130268

Language: Matlab

P1.

Implement a piecewise linear transformation function (below figure) for image contrast stretching. The code should read in an image; for intensity of all pixels, use the function to compute new intensity values; and finally output/ save the image with new intensity values.

Code:

```
% set arguments
k = 0.5;
r = 80;
x = 0 : 1 : 255;
y = pixel_de(k, r, x);

% read the picture
A = double(imread('test1-1.tif'));

% show the picture
figure(1)
imshow(uint8(A))

% process
B = pixel_de(k, r, A);
figure(2)
imshow(uint8(B))
title(['r = ', num2str(r), ', k = ', num2str(k)]);
imwrite(uint8(B), 'result1-1.tif')

% Piecewise function
function m = pixel_de(k, r, t)
m = k.*t.*(t>=0 & t<r) + ((255-2*r.*k)/(255-2*r).*(t-r)+r*k).*(t>=r
& t<(255-r)) + (k*(t-255)+255).*(t>=(255-r) & t<=255);
```

```
end
```

Result:


before after

P2. (1) implement n-dimensional joint histogram and test the code on two-dimensional data; plot the results.

(2) implement computation of local histograms of an image using the efficient update of local histogram method introduced in local histogram processing.

Note that because only one row or column of the neighborhood changes in a one-pixel translation of the neighborhood, updating the histogram obtained in the previous location with the new data introduced at each motion step is possible and efficient in computation.

NOTING: The second little question is part of the third question, which is not specially written here

Code:

```
% clean the buffer
clc
clear

% read the picture
A = imread('hw1-2.jpg');
```

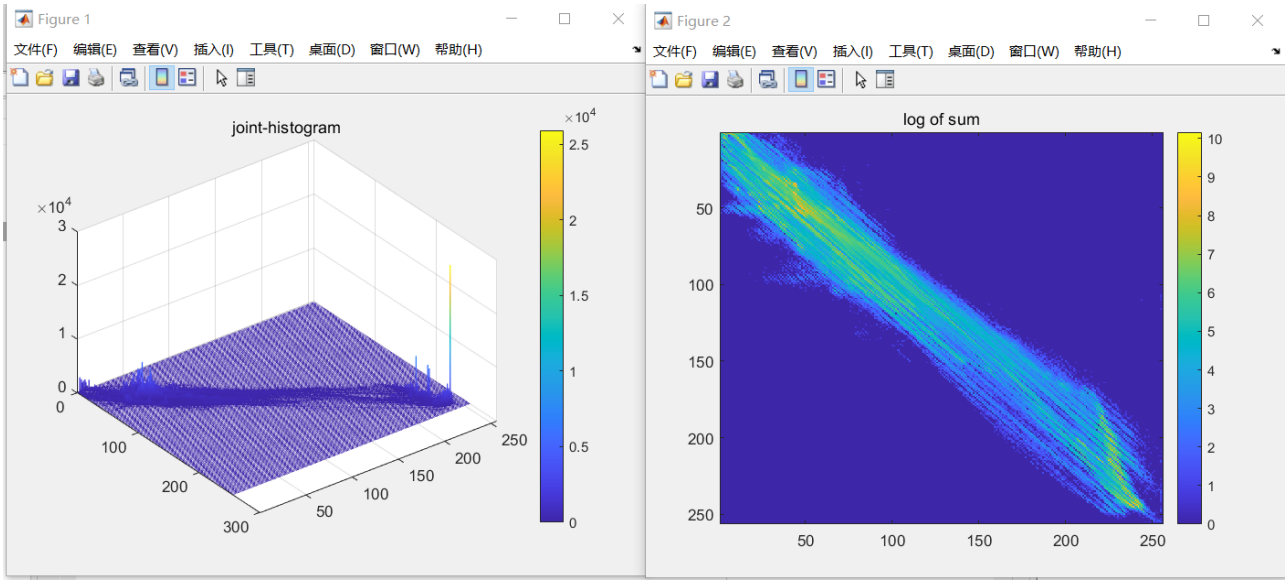
```
% process
h = joint_histogram(A, 2);

% 2D histogram process function
function h=joint_histogram(Z, K)
r = size(Z, 1);
c = size(Z, 2);
n = size(Z, 3);
if n < K
    disp("Error!not enough dimensions")
    quit
end

% initialize a K-dimensional matrix
N = 256;
h = zeros(ones(1, K) * N);

% update the date
for i = 1 : r
    for j = 1 : c
        a = zeros(K, 1);
        for k = 1 : K
            a(k, 1) = Z(i, j, k) + 1;
        end
        a1 = num2cell(a);
        h(a1{:}) = h(a1{:}) + 1;
    end
end
figure
hb = bar3(h);
title('joint-histogram')
shading interp
for k = 1 : length(hb)
    zdata = hb(k).ZData;
    hb(k).CData = zdata;
    hb(k).FaceColor = 'interp';
end
colorbar
figure
imagesc(log(h))
title('log of sum')
colorbar
end
```

result:



P3.

Implement the algorithm of local histogram equalization: (1) first implement histogram equalization algorithm, and then (2) implement the local histogram equalization using efficient computation of local histogram. Please test your code on images and show the results in your report.

Code:

```
% clean the buffer
clc
clear
tic

% read the picture
A = imread('test1-3.tif');
subplot(211)

% show the origin picture
imshow(A)
title("origin picture")

% global histogram equalization
A = double(A);
[mean, var] = mv(A);
```

```
% set the arguments
n = 3;
[r, c] = size(A);

% update the local hist
for i = (n/2+1) : (r-n/2)
    for j = (n/2+1) : (c-n/2)
        M = A(i-n/2 : i+n/2, j-n/2 : j+n/2);
        M = hist_eq(M, 4, 0.4, 0.02, 0.4, mean,  var);
        A(i-n/2 : i+n/2, j-n/2 : j+n/2) = M;
    end
end
A = uint8(A);
subplot(212)
imshow(A)
imwrite(A, 'result1-3.jpg')
title("after local histogram equalization")
toc

function [mean, var] = mv(A)
[r, c] = size(A);
mean = 0;
var2 = 0;
for i = 1 : r
    for j = 1 : c
        mean = mean + A(i, j);
    end
end
mean = mean / numel(A);
for i = 1 : r
    for j = 1 : c
        var2 = var2 + (A(i, j) - mean)^2;
    end
end
var = sqrt(var2);
end

function A = hist_eq(A, E, k0, k1, k2, mean,  var)
[mean1, var1] = mv(A);
if mean1 <= k0 * mean && k1 * var <= var1 <= k2 * var
    A = E .* A;
end
```

```
end
```

Result: