Computer Vision and Image Processing

**Project 1**

**Name/Sign: Pranati Trivedi**

**Date: 02/28/2019**

1. Histogram Processing

(a) (40 points) The global histogram equalization technique is easily adaptable to local histogram equalization. The procedure is to define a square or rectangular window (neighborhood) and move the center of the window from pixel to pixel. At each location, the histogram of the points inside the window is computed and a histogram equalization transformation function is obtained. This function is finally used to map the intensity level of the pixel centered in the neighborhood to create a corresponding (processed) pixel in the output image. The center of the neighborhood region is then moved to an adjacent pixel location and the procedure is repeated. Write an M-function for performing local histogram equalization. Your function should have the following specifications:

* “**g = localhisteq(f, m, n)**” performs local histogram equalization on input image **f** using

a window size 􀝉 􀵈 􀝊 to produce the processed image **g**.

* To handle border effects, image **f** is extended by using the ‘symmetric’ option. The amount

of extension is determined by the dimensions of the local window.

* If **m** and **n** are omitted, they default to 3. If **n** is omitted, it defaults to **m**. Both must be odd.

(b) Process “test1.tiff” with your function **localhisteq** using neighborhoods of sizes 3 x 3 and7 x 7. Explain the differences in your results.

(c) Histogram-equalize the test image using the Image Processing Toolbox (IPT) function **histeq**. Compare the result with those obtained from (b).

Untitled2:

clc;

clear all;

f=imread('D:\IITCSem2\CVIP\test1.tif');

k = localhisteq(f,3,3);

j = localhisteq(f,7,7);

l = histeq(f);

subplot(1,4,1);

imshow(f);

title('original');

subplot(1,4,2);

imshow(k);

title('3x3');

subplot(1,4,3);

imshow(j);

title('7x7');

subplot(1,4,4);

imshow(l);

title('global histogram');

localhisteq.m

function g = localhisteq(f, m, n)

%LOCALHISTEQ Local histogram equalization.

% G = LOCALHISTEQ(F, M, N) performs local histogram equalization

% on input image F using a window of (odd) size M-by-N to produce

% the processed image, G. To handle border effects, image F is

% extended by using the symmetric option in function padarray.

% The amount of extension is determined by the dimensions of the

% local window. If M and N are omitted, they default to

% 3. If N is omitted, it defaults to M. Both must be odd.

%

% This function accepts input images of class uint8, uint16, or

% double. However, all computations are done using 8-bit intensity

% values to speed-up computations. If F is of class double its

% values should be in the range [0 1]. The class of the output

% image is the same as the class of the input.

if nargin == 1 % Check the number of arguments input into M-functions

m = 3; n = 3

elseif nargin == 2

n = m;

end

if ~(round(m/2) ~= m/2) | ~(round(n/2) ~= n/2) % rounds each element of X to the nearest integer

error('It should be odd dimension of neighborhood')

end

classin = class(f); % For use later.

f = im2uint8(f); %Work with 8-bit accuracy.

% Extend the borders of the input image.

[R C] = size(f); % For use later.

f = padarray(f, [m n], 'symmetric', 'both'); % add m and n elements of padding to the end of the second dimention

% using symmetric, pad by repeating border elements of array and using

% 'both', pad before first element and after last element

% Move the window from pixel to pixel and histogram equalize at

% each location.

yj = 0;

den = m\*n;

for x = (m + 1)/2 + 1 : R + (m + 1)/2

yj = yj + 1;

yg = 0;

for y = (n + 1)/2 + 1 : C + (n + 1)/2

yg = yg + 1;

subimage = f(x:x + m - 1, y:y + n - 1);

h = histeq2(subimage); % Can also use imhist, but watch the scaling.

% Pick the value in the center of the equalized subimage. Function

% histeq contains the necessary mapping function.

g(yj, yg) = h((m + 1)/2, (n + 1)/2);

end

end

% Convert back to the class of the input.

g = changeclass(classin, g);

intxform:

function z = intxform (s, map)

%INTXFORM Intensity transformation.

% Z = INTXFORM(S, MAP) maps the intensities of input image S using

% mapping function, MAP, whose values are assumed to be in the range

% [0 1]. MAP specifies an intensity transformation function.

% For example, to create a map function that

% squares the input intensities of an input image of class uint8 and

% then use function INTXFORM to perform the mapping we write:

%

% t = linspace(0, 1, 256);

% map = t.^2;

% z = intxform(s, map);

classin = class(s);

s = im2double(s);

% Construct x and y vectors, representing the intensity

% transformation, to be used with INTERP1. Make sure they are both

% column vectors.

x = linspace(0, 1, numel(map))'; % generates number of elements mapped. The spacing between the points is (1-0)/(numel(map)-1).

y = map(:);

% Apply the intensity transformation using INTERP1.

z = interp1(x, y, s, 'linear');

% Change to class of the input.

z = changeclass(classin, z);

histeq:

function h = histeq(f)

%HISTEQ2 Histogram equalization.

% H = HISTEQ(F) histogram-equalizes F and outputs the result

% in H. Unlike IPT function histeq, HISTEQ implements the direct,

% "standard" histogram equalization approach.F can be of class uint8,

% uint16, or double. If F is double,

% then its values are assumed to be in the range [0 1]. The intensity

% range of the input image (regardless of class) is divided into 256

% increments for computing the histogram and the corresponding cumulative

% distribution function (CDF). Recall that the CDF is actually the mapping

% function used for histogram equalization.

%

classin = class(f);

f = im2uint8(f);

% The histogram and the cumulative distribution function.

den = numel(f);

hst = imhist(f, 256) / den;

cdf = cumsum(hst); % Values are in the range [0 1].

% Now map the values of the input image to obtain the histogram-

% equalized image, g.

h = intxform(f, cdf);

% Convert to the class of the input

h = changeclass(classin, h);

changeclass.m

function image = changeclass(class, varargin)

%CHANGECLASS changes the storage class of an image.

% I2 = CHANGECLASS(CLASS, I);

% RGB2 = CHANGECLASS(CLASS, RGB);

% BW2 = CHANGECLASS(CLASS, BW);

% X2 = CHANGECLASS(CLASS, X, 'indexed');

switch class

case 'uint8'

image = im2uint8(varargin{:});

case 'uint16'

image = im2uint16(varargin{:});

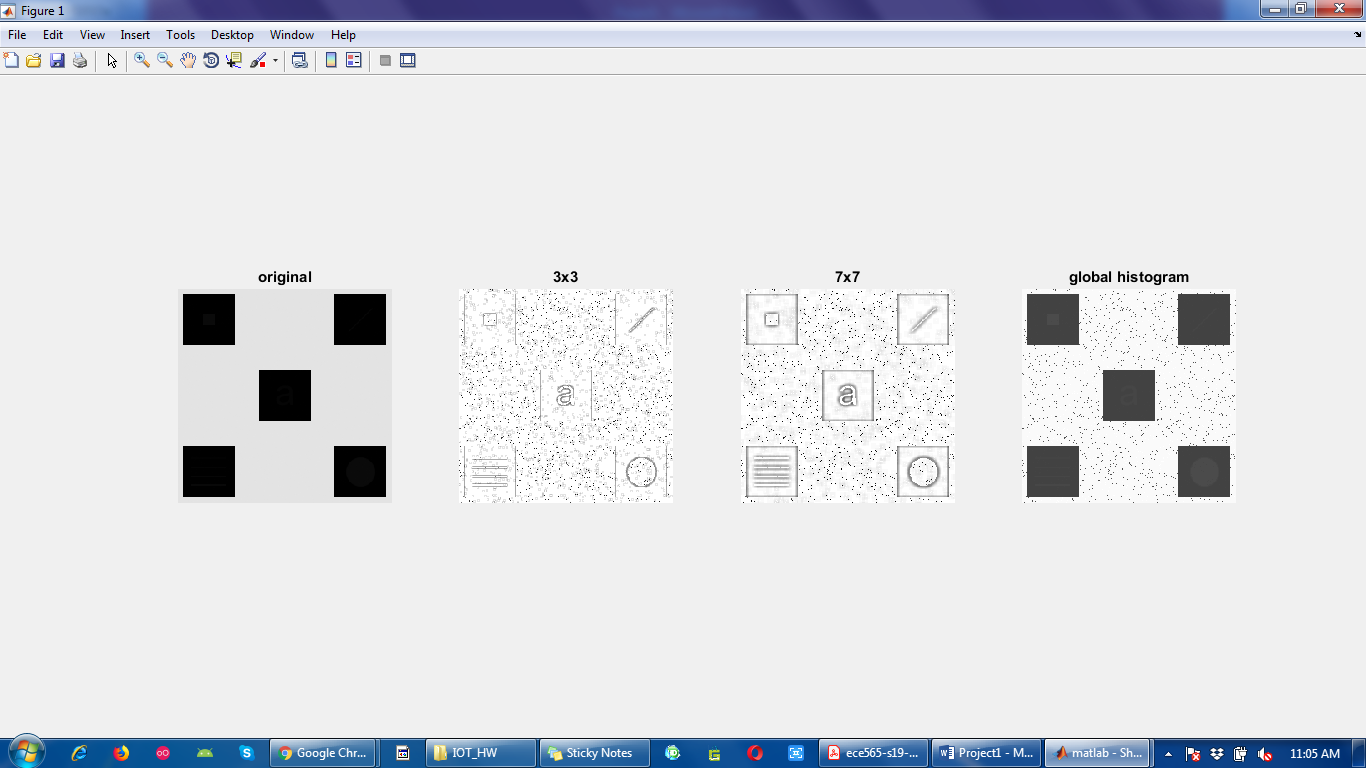
case 'double'

image = im2double(varargin{:});

otherwise

error('IPT data class which are not supported');

end



(b) The difference between of sizes with 3x3 and 7x7 is that the distortion will be increased as its size of the local window increases. So the blurring effect caused by the intensity mapping affected by the histogram of a big area.

(c)

Image using the Image Processing Toolbox (IPT)function histeq could not give details the reason behind that is that they had very small effect on the shape of the global transformation function such as cumulative distribution function.

(2) The 2-D Fourier transform is represented in polar form as

F(u,v)=|F(u,v)|ejФ(u,v)

Here,

|F(u,v)|= [R2(u,v) +I2(u,v)]1/2

is the magnitude spectrum and

Ф(u,v)=tan-1[I(u,v)/R(u,v)]

is the phase angle. The phase angle carries information about the location of image elements and the spectrum carries information regarding contrast and intensity transitions. It is recommended that you look up functions **complex** and **angle**.

(a) Compute the magnitude spectrum and the phase angle of image “test2.tiff” and show the phase image. Then, compute the inverse transform using only the phase term. Apply appropriate functions in MATLAB to obtain the magnitude spectrum and the phase angle of the image and show the image. Discuss your results.

(b) Compute the inverse using only the magnitude term. Show the image and discuss your results.

(c) Recover the image using the magnitude spectrum of “test2.tiff” and the complex conjugate of the phase component. Discuss your results.

%(a)Magnitude Spectrum and the phase angle of image

f=imread('D:\IITCSem2\CVIP\test2.tif'); % read the image

subplot(2,3,1)

imshow(f) % representation of image

title('original');

F=fft2(f); % returns the two dimentional Fourier transform

Mf=abs(F); % Magnitude spectrum

Af=angle(F); % phase angle of image

subplot(2,3,2)

imshow(Af, [])

title('phase image');

% inverse transform using only the phase term

E=complex(0,Af); %creates complex output such as 0 + iAf

fphase=real(ifft2(exp(E))); %returns the real part of the complex number of inverse transform

subplot(2,3,3)

imshow(fphase, []);

title('image by phase angle');

%Sfi=abs(fphase);

%Afi=angle(fphase);

%figure, imshow(Afi, [])

%(b) Inverse using only the magnitute term

mspec=real(ifft2(Mf)); % real part of the inverse of magnitude spectrum

subplot(2,3,4)

imshow(log(1 + abs(mspec)), []) % displays the grayscale image based on the range of pixel values

title('Image by spectrum');

%(c) Recover the image using the magnitude spectrum

g=imread('D:\IITCSem2\CVIP\test2.tif');

subplot(2,3,5)

imshow(g)

title('original');

Sg=abs(fft2(g));

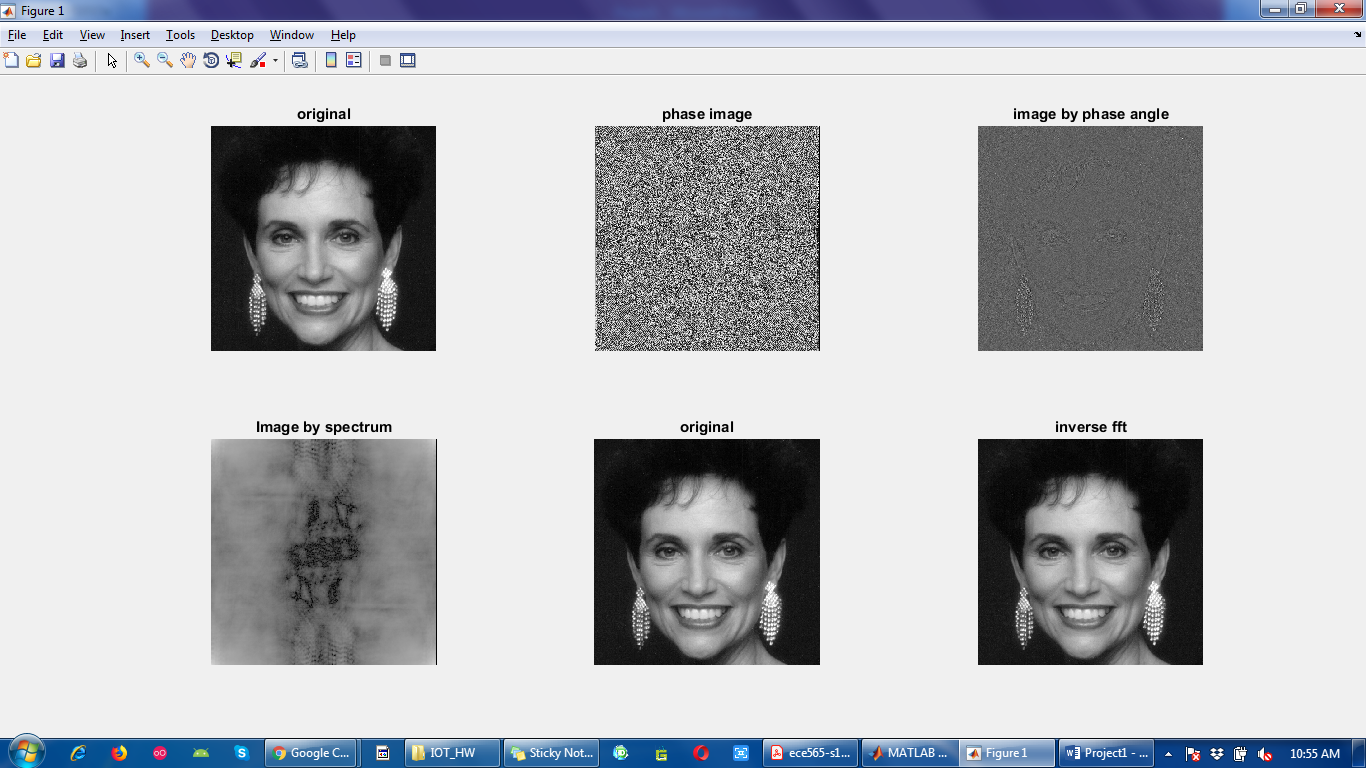
FT=Sg.\*exp(E); % magnitude and complex congugate of the phase component

OI=real(ifft2(FT));

subplot(2,3,6)

imshow(OI, [])

title('inverse fft');



1. As we can see from the image that all the gray tonality was lost from the original image. This information is carried by the spectrum.
2. Here we can see from the image of inverse using spectrum that it is not meaningful and we cannot detect the original image. Here to get meaningful image, the values of the spectrum has to be scaled. Here we use log in the preceding command because a value of this image has a large dynamic range. The image is totally gray from the inverse spectrum but we can see few features in the image. Hence we can say from the result (a) and (b) that an image we got from the magnitude of the spectrum is unrecognizable there is more dynamic range problem. The image recovered using from the phase faintly is really low in quality. We can see only dull outline of the main features in the original image.
3. The woman’s face dominates the result, indicating that phase information is dominant in determining the visual appearance of an image.