SCSV3213

FUNDAMENTAL OF IMAGE PROCESSING

IMAGE ENHANCEMENT IN SPATIAL DOMAIN

Lecture 3 (week4-6)

Dr. Md Sah Hj Salam

Acknowledgements

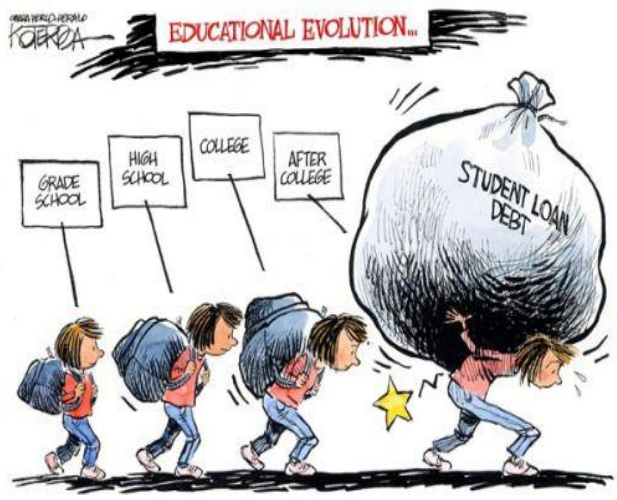
Most of the slide are taken and modified from other resources including books and slides from lectures from others universities. It is rearranged to suit the syllabus of the course.

SYNOPSIS

In this lecture, image enhancement operations in spatial domain will cover the followings

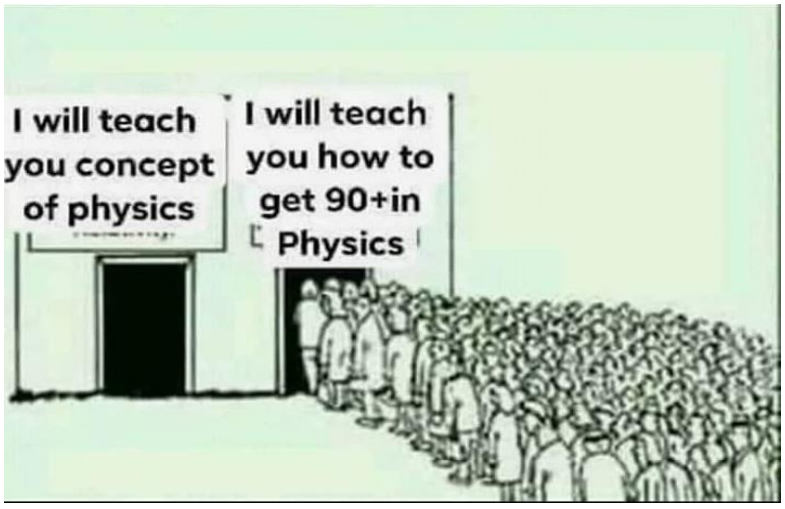
1. Point Processing
   - Arithmetic Processing
   - Histogram Processing

2. Neighborhood Operation
   - Fundamental of spatial filtering
   - Smoothing Spatial Filter
   - Sharpening Spatial Filters



An image worth thousands words !!! –
Do you experience this??

WHICH ONE DO YOU CHOOSE!



1. POINT PROCESSING

Point Processing

- Remember this equation at the introduction lecture $H(f(x,y)) = f(x,y) / 2$
- What does it tells ??
- The resulted operation is

6	8	2	0
12	200	20	10

3	4	1	0
6	100	10	5

- This is an example of point processing operation

Point Processing

- Definition: Point processing is the operation that consider the pixel's value alone independent of the position of the pixel.
  - Example : g = t(f) % where t is the transformation operation
- Later we will see the different between point processing and neighborhood processing.
- For now we will use this definition for point processing.

Point operations

- Example: Can you guess what operation is applied to the image. Image (A) is the subject image and (B) is the resulted image

(A)

(B)

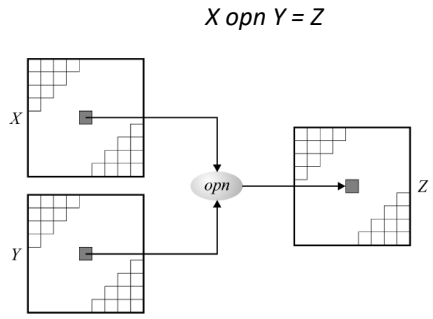
Point Processing

- The followings are the point processing operations covered in this lecture
  - Arithmetic and Logic operations
    - Addition and Subtraction
    - Multiplication and Division
    - AND , OR, XOR and NOT
  - Histogram operations
    - Histogram Stretching
    - Histogram Sliding
    - Histogram Shrinking
    - Histogram Equalization

1.1 ARITHMETIC AND LOGIC OPERATION

Arithmetic and logic operations

- Combine two images, pixel-by-pixel, using an arithmetic or logical operator, resulting in a third image, Z:



Note : opn is the arithmetic and logic operation

- Or simply do operation with constant value to the image.
- H(f(x,y)) = f(x,y) opn A % A is constant

Arithmetic Operation : Addition

• H(f(x,y)) = f(x,y) + A

```
I = imread('cameraman.tif');      % read the image
I2 = I + 50;                      % 50 is a constant value
subplot(1,2,1); imshow(I);        %plot the images
subplot(1,2,2);imshow(I2);
```

•try see the max and min size of I and I2. You can find them using

- •function max(I(:)) and min(I(:)) or
- •Just look at the workspace at the Matlab environment

• The value is capped to the max value for grey scale value ie 255.

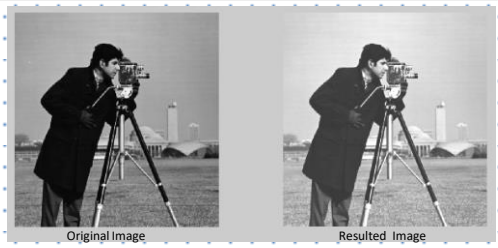
Arithmetic Operation : Addition

• Using imadd() function.

```
I = imread('cameraman.tif');
I2 = imadd(I,75);
figure
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I2), title('Brighter Image');
```

•try see the max and min size of I and I2.

•SIMILAR RESULT?



Arithmetic Operation : Addition

•Handling overflow

- Truncated
- Normalized

$$g = \frac{L_{max}}{f_{max} - f_{min}}(f - f_{min})$$

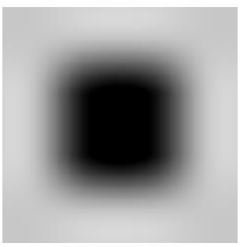
```
X = uint8([200 100 100; 0 10 50; 50 250 120])
Y = uint8([100 220 230; 45 95 120; 205 100 0])
W = uint16(X) + uint16(Y)

fmax = max(W(:))
fmin = min(W(:))

Za = uint8(255.0*double((W-fmin))/double((fmax-fmin)))
Zb = imadd(X,Y)
```

Normalize the matric

$$A = \begin{pmatrix} 300 & 120 & 255 \\ 190 & 250 & 100 \\ 200 & 310 & 190 \end{pmatrix}$$

Arithmetic Operation : Addition

- Combining/ blend two images
- Download the images at e-learning. Follow the script in the script1.txt #example4


Arithmetic Operation : Subtraction

- Used to used to detect differences between two images, decrease its overall brightness, or obtain its negative.
- Example 1 (subtractive image offset):


Z = X - 75

Arithmetic Operation : Subtraction

- Example (negative of an image):


Arithmetic Operation : Subtraction

• H(f(x,y)) = f(x,y) - A.

```
I = imread('cameraman.tif');
I2 = imsubtract(I,50);
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I2), title('Darker Image');
```

Use operator ' – '

- Does the result the same ?
- How do they (imsubtract & ' –') cater the problem of negative value ?

Arithmetic Operation : Subtraction

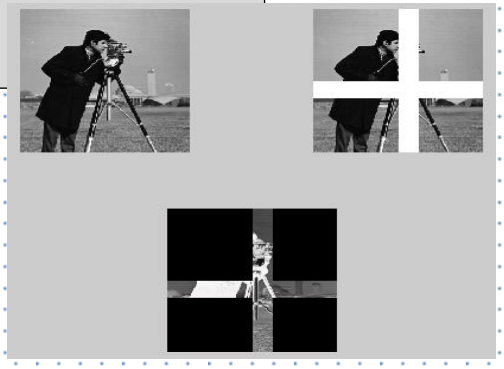
Subtracting two images

- What will happen to resulted image when two images are substracted. A = B – C?
- Let make a simple test.
- Algorithm:
  - Open image A;
  - Modify image A (ex. by making white stripe on the image) and save as B.
  - Subtract B from A and save to C. C = A – B;
  - Plot all images
- Write the Matlab code and see the result.

Arithmetic Operation : Subtraction

Subtracting two images

```
A = imread('cameraman.tif');
B = A;
B(130:160,:) = 255; B(:,130:160)=255;
C = imsubtract(B,A);
subplot(2,2,1);imshow(A);
subplot(2,2,2);imshow(B);
subplot(2,2,3:4);imshow(C);
```



Arithmetic Operation : Subtraction

Subtracting two images

- A = imread('cameraman.tif');
- B = A;
- B(130:160,:) = 255; B(:,130:160)=255;
- C = imsubtract(B,A);
- subplot(2,2,1);imshow(A);
- subplot(2,2,2);imshow(B);
- subplot(2,2,3:4);imshow(C);

Change this line to

- C= imsubtract(A,B);
- C=imabsdiff(A,B);
- C=imcomplement(A);

Explain the resulted image

Subtraction in MATLAB

- Three functions:
  - **`imsubtract`**
  - **`imabsdiff`**
  - **`imcomplement`**
- Example :

```
X = uint8([200 100 100; 0 10 50; 50 250 120])
Y = uint8([100 220 230; 45 95 120; 205 100 0])
Za = imsubtract(X,Y)
Zb = imsubtract(Y,X)
Zc = imabsdiff(Y,X)
```

Multiplication and division in MATLAB

- Functions:
  - **`immultiply`**
  - **`Imdivide`**

Example :


image X


Z = X * 0.7


Z = X / 0.7

Combining arithmetic operations

- **`imlincomb`** function
- Example:

```
X = uint8([200 100 100; 0 10 50; 50 250 120])
Y = uint8([100 220 230; 45 95 120; 205 100 0])
Z = uint8([200 160 130; 145 195 120; 105  240 150])

Sa = imdivide(imadd(X,imadd(Y,Z)),3)
a = uint16(X) + uint16(Y)

b = a + uint16(Z)
Sb = uint8(b/3)

Sc = imlincomb(1/3,X,1/3,Y,1/3,Z,'uint8')
```

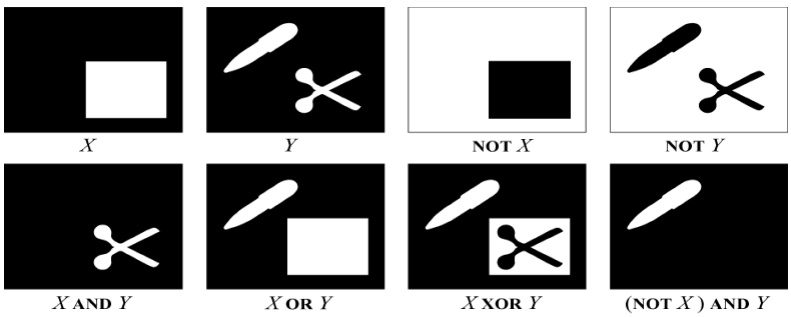
Logic operations

- Performed in a bit-wise fashion on the binary contents of each pixel value.
  - AND, XOR and OR require two or more arguments.
  - NOT operator only requires only one argument.

- The convention: 1 (true) for white pixels; 0 (false) for black pixels.

Logic operations

- Examples using binary test images



X	Y	OR	XOR	AND
0	0	?	?	?
0	1			
1	0			
1	1			

Logic operations on monochrome images

- Example 1 (AND)

Image 1

Image 2

Image 3

- Explain the Images

Logic operations on monochrome images

- Example 2 (OR)

Image 1

Image 2

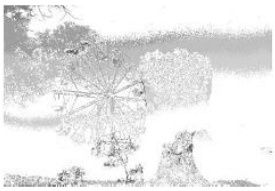


Image 3

- Explain the Images

Logic operations on monochrome images

- Example 3 (XOR)

Image 1

Image 2

Image 3

- Explain the Images

Logic operations on monochrome images

- Example 4 (NOT)

Image 1

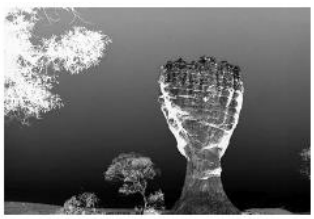


Image 2

- Explain the Images

TUTORIAL ON ARITHMETIC AND LOGIC OPERATION

Execute the scripts print the output and the scripts.
Compile the tutorial to be submitted end of semester.

TUTORIAL 1: Addition

- Brighten an image using imadd()

```
I = imread('tire.tif');
I2 = imadd(I,75);
figure
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I2), title('Brighter Image');
```

- Blend 2 images

```
Ia = imread('rice.png');
Ib = imread('cameraman.tif');
Ic = imadd(Ia, Ib);
figure; subplot(2,2,1); imshow(Ia), title('Image 1');
subplot(2,2,2),imshow(Ib), title('Image 2');
subplot(2,2,3:4),imshow(Ic), title('Blended Image');
```

```
close all;    % close all figures
clear all;    % clear all workspace variable
```

TUTORIAL 2: Subtraction

- Load two images and display them

```
I = imread('cameraman.tif');
J = imread('cameraman2.tif');
figure(1); subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(J), title('Altered Image');
```

- Subtract both image and display result

```
diffim = imsubtract(I,J);
figure(2)
subplot(2,2,1), imshow(diffim), title('Subtracted Image');
```

- Zoom in the using zoom tool at the right area of the image. You should find a small region of pixel that is faintly white. Now use absolute diff function.

```
diffim2 = imabsdiff(I,J);
subplot(2,2,2), imshow(diffim2), title('Abs Diff Image');
```

- Zoom in the using zoom tool at the right area of the image like before.
- Explain the different between the two functions.
- Can you make the faintly white region clearer?
  - Write simple code to make the region white.
- Close all figures and clear all variables.

TUTORIAL 3: Multiply

- Comparing between normal brightening and dynamic brightening using multiply op

```
I = imread('moon.tif');
I2 = imadd(I, 50);
I3 = immultiply(I, 1.2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Normal Brightening');
subplot(1,3,3), imshow(I3), title('Dynamic Scaling');
```

- Blend two images using multiply op.

```
I = im2double(imread('earth1.tif'));
J = im2double(imread('earth2.tif'));
K = immultiply(I,J);
figure
subplot(1,3,1), imshow(I), title('Planet Image');
subplot(1,3,2), imshow(J), title('Gradient');
subplot(1,3,3), imshow(K,[]), title('3D Planet');
```

TUTORIAL 4: Multiply and Division

- Use division operation to dynamically darken the image

```
I = imread('moon.tif');
I2 = imdivide(I,2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Darker Image w/ Division');
```

- Get the same result using multiply

```
I3 = immultiply(I,0.5);
subplot(1,3,3), imshow(I3), title('Darker Image w/ Multiplication');
```

TUTORIAL 5: Logic operation

- Use bitxor op to fiind the different of two images

```
I = imread('cameraman.tif');
I2 = imread('cameraman2.tif');
I_xor = bitxor(I,I2);
figure; subplot(1,3,1), imshow(I), title('Image 1');
subplot(1,3,2), imshow(I2), title('Image 2');
subplot(1,3,3), imshow(I_xor,[]), title('XOR Image');
```

- Make lindsay eyes dark ..

```
I = imread('lindsay.tif');
I_adj = imdivide(I,1.5);
subplot(1,2,1); imshow(I);
subplot(1,2,2); imshow(I_adj);
bw = im2uint8(roipoly(I)); % Generate a mask by creating a region of interest polygon.
% Use logic operators to show the darker image only within the region of
% interest while displaying the original image elsewhere.
bw_cmp = bitcmp(bw);          %mask complement
roi = bitor(I_adj,bw_cmp);    %roi image
not_roi = bitor(I,bw);        %non_roi image
new_img = bitand(roi,not_roi);  %generate new image
imshow(new_img)               %display new image
```

Test your understanding

- The code below add 4 moles for lindsay image..
- Write a simple code using arithmetic and logic operations to detect the position of the image by comparing both images.

```
I = imread('lindsay.tif');
I2=I;
I2(166:167,176:177)=0;
I2(166:167,145:146)=0;
I2(56:57,176:177)=0;
I2(123:124,115:116)=0;
figure; subplot(1,2,1);imshow(I);
subplot(1,2,2);imshow(I2);
```

1.2. HISTOGRAM OPERATION

Histogram

- The histogram of a monochrome image is a graphical representation of the frequency of occurrence of each gray level in the image.
- The data structure that stores the frequency values is a 1D array of numerical values, *h*, whose individual elements store the number (or percentage) of image pixels that correspond to each possible gray level.

What is a histogram?

$$h(k) = n_k = \text{card}\{(x, y) | f(x, y) = k\}$$

where:
$k = 0, 1, ..., L - 1$, where L is the number of gray levels of the digitized image; and
$\text{card}\{\cdots\}$ denotes the cardinality of a set, i.e. the number of elements in that set (n_k).
A normalized histogram can be mathematically defined as:

$$p(r_k) = \frac{n_k}{n}$$

where:
n = total number of pixels in the image; and
$p(r_k)$ = probability (percentage) of the k-th gray level (r_k).

What is a histogram?

$$h(k) = n_k = \text{card}\{(x, y) | f(x, y) = k\}$$

where:
$k = 0, 1, ..., L - 1$, where L is the number of gray levels of the digitized image; and
$\text{card}\{\cdots\}$ denotes the cardinality of a set, i.e. the number of elements in that set (n_k).
A normalized histogram can be mathematically defined as:

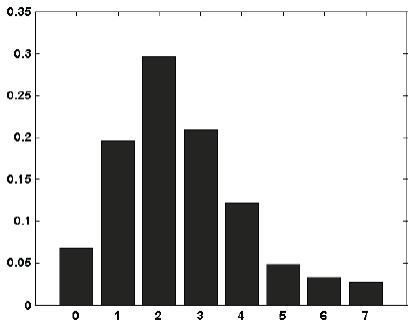
$$p(r_k) = \frac{n_k}{n}$$

where:
n = total number of pixels in the image; and
$p(r_k)$ = probability (percentage) of the k-th gray level (r_k).

Histogram example

- Histogram for a hypothetical image containing 128×128 pixels and 8 gray levels.

Gray level (r_k)	n_k	$p(r_k)$
0	1120	0.068
1	3214	0.196
2	4850	0.296
3	3425	0.209
4	1995	0.122
5	784	0.048
6	541	0.033
7	455	0.028
Total	16384	1.000



Calculate the histogram of 8 grey level image of this image portion

3	3	4	4	4	4
4	4	3	3	3	6
4	4	5	5	3	1
4	4	5	6	6	1
6	6	5	5	5	6
6	6	5	5	5	5

Grey Level (n_k)	n_k	P(r_k)
0		
1		
2		
3		
4		
5		
6		
7		
Total		

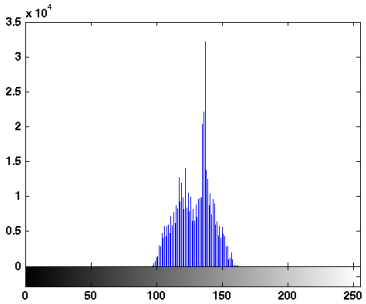
Calculate the histogram of 8 grey level image

3	3	4	4	4	4
4	4	3	3	3	6
4	4	5	5	3	1
4	4	5	6	6	1
6	6	5	5	5	6
6	6	5	5	5	5

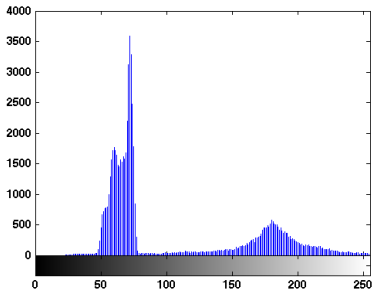
Grey Level (n_k)	n_k	P(r_k)
0	0	0.00
1	2	0.05
2	0	0.00
3	6	0.17
4	10	0.28
5	10	0.28
6	8	0.22
7	0	0.00
Total	36	1.00

Examples of images and their histograms

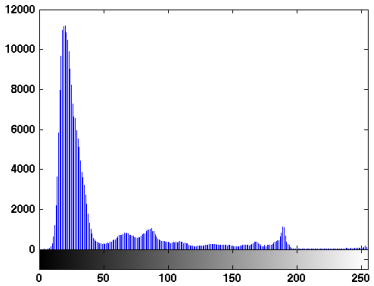
- In MATLAB: **`imhist`**



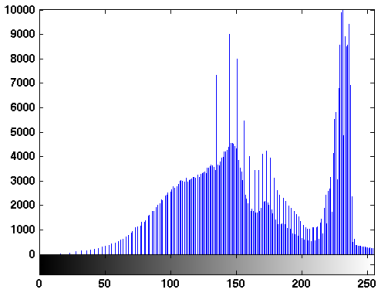
Examples of images and their histograms



Examples of images and their histograms



Examples of images and their histograms



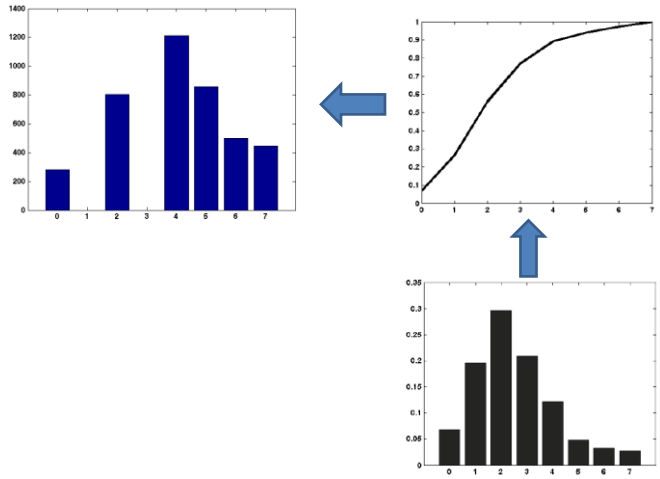
Interpreting image histograms

- Histograms have become a popular tool for conveying image statistics and helping determine certain problems in an image.
- A histogram carries significant qualitative and quantitative information about the corresponding image (e.g., minimum, average, and maximum gray level values, dominance of bright or dark pixels, etc.).
- A histogram is not enough to draw qualitative conclusions about the overall quality of the image, presence or absence of noise, etc.

Interpreting image histograms

- Although a histogram provides the frequency distribution of gray levels in an image, it tells us nothing about the spatial distribution of the pixels whose gray levels are represented in the histogram.
- Histograms can be used whenever a statistical representation of the gray level distribution in an image is desired.
- Histograms can also be used to enhance or modify the characteristics of an image, particularly its contrast.

Histogram equalization



Histogram equalization

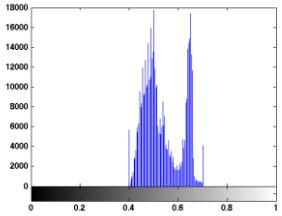
- In MATLAB: **`histeq`**

```
I = imread('sydney_low_contrast.png');
I = im2double(I);
J = histeq(I);
figure, subplot(2,2,1), imshow(I), ...
        subplot(2,2,2), imshow(J), ...
        subplot(2,2,3), imhist(I), ylim('auto'),...
        subplot(2,2,4), imhist(J), ylim('auto')
```

Histogram equalization

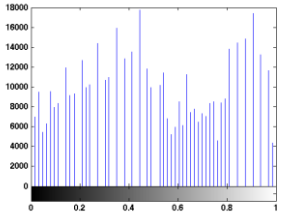
- Example :

(a)



(b)

(c)



(d)

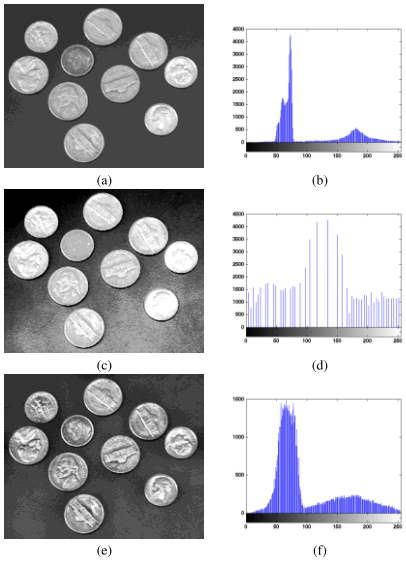
Global vs. local histogram equalization

- In MATLAB: **`histeq`** and **`adapthisteq`**

```
I = imread('coins.png');
figure, subplot(1,2,1), imshow(I), ...
        subplot(1,2,2), imhist(I), ylim('auto')

J = histeq(I);
figure, subplot(1,2,1), imshow(J), ...
        subplot(1,2,2), imhist(J), ylim('auto')

K = adapthisteq(I);
figure, subplot(1,2,1), imshow(K), ...
        subplot(1,2,2), imhist(K), ylim('auto')
```

Global vs. local histogram equalization

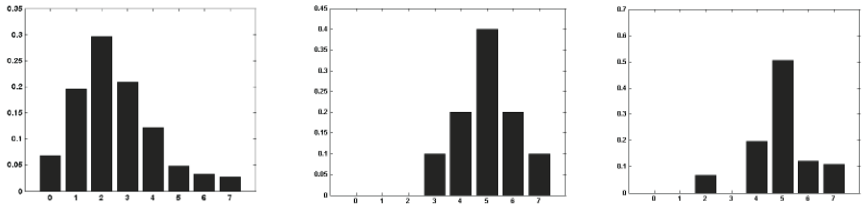
- Example :



Direct histogram specification

- Example

Original Desired Result



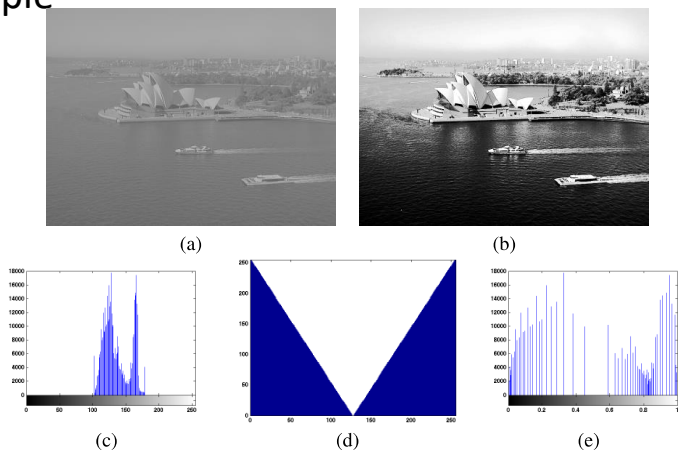
Direct histogram specification

- In MATLAB: **`histeq`**
- Example :

```
I = imread('sydney_low_contrast.png');
Id = im2double(I);
figure, imhist(Id), ylim('auto'), ...
        title ('Original histogram');

des_hist = uint8(zeros(1,256));
des_hist(1:128) = linspace(256,0,128);
des_hist(129:end) = linspace(0,256,128);
x_axis = 0:255;
figure, bar(x_axis, des_hist), axis tight,
...
        title('Desired histogram');

hgram = im2double(des_hist);
Jd = histeq(Id,hgram);
figure, imhist(Jd), ylim('auto'), ...
        title ('Resulting histogram');
```

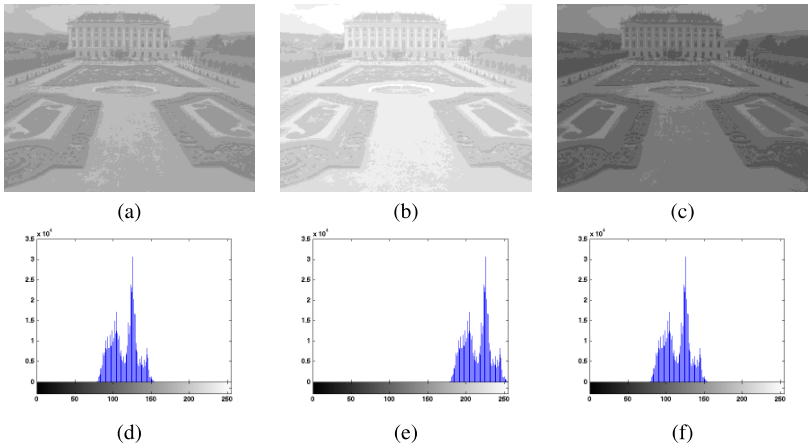
Direct histogram specification

- Example



Other histogram modification techniques

- Histogram sliding :



Histogram sliding

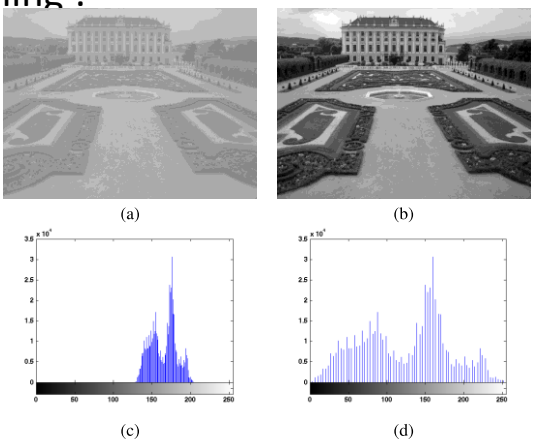
- In MATLAB: **`imadd`** and **`imsubtract`**
- Example :

```
I = imread('schonbrunn_gray_low_contrast.png');
figure, imhist(I), ylim('auto'), title ('Original histogram');

I2 = imadd(I, 50);
figure, imhist(I2), ylim('auto'), ...
        title ('Sliding to the right by 50');

I3 = imsubtract(I,50);
figure, imhist(I3), ylim('auto'), ...
        title ('Sliding to the left by 50');
```

Other histogram modification techniques

- Histogram stretching :

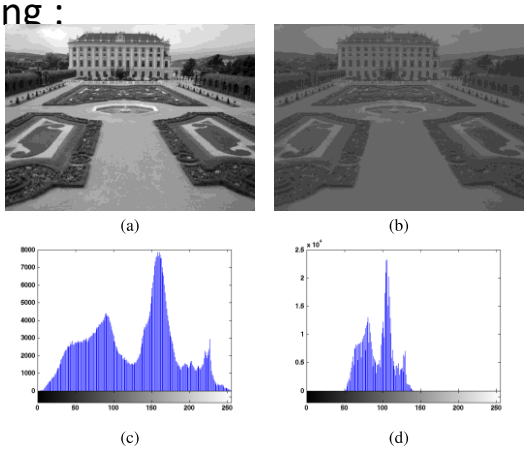
$$s = \frac{r - r_{min}}{r_{max} - r_{min}} \cdot (L - 1)$$



Other histogram modification techniques

- Histogram shrinking :

$$s = \left[\frac{s_{max} - s_{min}}{r_{max} - r_{min}}\right](r - r_{min}) + s_{min}$$



Histogram stretching and shrinking

- In MATLAB: **`imadjust`**

```
%% Histogram stretching
I = imread('schonbrunn_gray_low_contrast.png');
figure, imhist(I), ylim('auto'), title ('Original histogram');
I2 = imadjust(I);
figure, imhist(I2), ylim('auto'), title ('After histogram stretching');
figure, subplot(1,2,1), imshow(I), subplot(1,2,2), imshow(I2)

%% Histogram shrinking
I = imread('schonbrunn_gray.png');
figure, imhist(I), ylim('auto'), title ('Original histogram');
Id = im2double(I);
Jd  = imadjust(Id, [], [49/255 140/255]);
J = uint8(255.*Jd);
figure, imhist(J), ylim('auto'), title ('After histogram shrinking');
figure, subplot(1,2,1), imshow(I), subplot(1,2,2), imshow(J)
```

TUTORIAL ON HISTOGRAM

TUTORIAL 6: Displaying Histogram

- Displaying an image and its histogram using 256 bins

```
I = imread('circuit.tif');
figure,  subplot(2,2,1), imshow(I), title('Image')
subplot(2,2,2),  imhist(I,256), axis  tight, title('Histogram')
```

- Using different bins

```
subplot(2,2,3), imhist(I,64),  axis  tight, ...
title('Histogram with  64 bins')
subplot(2,2,4), imhist(I,32),  axis  tight, ...
title('Histogram with  32 bins')
```

- Get the value of the hist to C and normalize the value

```
c = imhist(I,32);
c_norm = c / numel(I);
```

TUTORIAL 6: continue

- Displaying using bar chart

```
figure, subplot(1,2,1), bar_1 = bar(c);
set(gca, 'XLim', [0 32], 'YLim', [0 max(c)]);
set(gca, 'XTick', [0:8:32], 'YTick', ...
[linspace(0,7000,8) max(c)]);
set(bar_1, 'FaceColor', 'r'), title('Bar Chart')
subplot(1,2,2), bar_2 = bar(c_norm);
set(gca, 'XTick', [0:8:32], 'YTick', ...
[linspace(0,0.09,10) max(c_norm)])
xlim([0 32]), ylim([0 max(c_norm)])
title('Normalized Bar Chart')
set(bar_2, 'FaceColor', 'g')
```

TUTORIAL 7: Hist Equalization

- Image histogram equalization

```
I = imread('pout.tif');
figure, subplot(2,2,1), imshow(I), ...
title('Original Image')
subplot(2,2,2), imhist(I), ...
title('Original Histogram')
I_eq = histeq(I,256);
subplot(2,2,3), imshow(I_eq), title('Equalized Image')
subplot(2,2,4), imhist(I_eq), title('Equalized Histogram')
```

- Explain the effect ..

TUTORIAL 7: continue

- Image histogram equalization 1

```
I = imread('pout.tif');
figure, subplot(2,2,1), imsho  I), ...
title('Original Image')
subplot(2,2,2), imhist(I), ...
title('Original Histogram')
I_eq = histeq(I,256);
subplot(2,2,3), imshow(I_e   e('Equa   mage')
subplot(2,2,4), imhist(I_eq), title('Equalized Histogram')
```

Same as previous

- Image histogram equalization 2

```
I = imread('tire.tif'); I_eq = histeq(I,256);
figure, subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), title('Original Histogram')
subplot(2,2,3), imshow(I_eq), title('Equalized Image')
subplot(2,2,4), imhist(I_eq), title('Equalized Histogram')
```

In comparison on both image ie. Pout and tire which is suitabe to apply histeq?

TUTORIAL 8: Hist modificatian

- Addition / sliding in histogram

```
J = imread('pout.tif');
I = im2double(J);
clear J
figure, subplot(3,2,1), imshow(I), title('Original Image')
subplot(3,2,2), imhist(I), axis tight, ...
title('Original Histogram')
% Obtain a brighter version of the input image by adding 0.1 to each pixel.
const = 0.1;
I2 = I + const;
subplot(3,2,3), imshow(I2), title('Original Image + 0.1')
subplot(3,2,4), imhist(I2), axis tight, ...
title('Original Hist + 0.1')
```

TUTORIAL 9: Hist modificatian

- Histogram sliding

```
%continue ffrom previous exmple
const = 0.5;
I3 = I + const;
bad_values = find(I3 > 1);
I3(bad_values) = 1;
subplot(3,2,5), imshow(I3), title('Original Image + 0.5')
subplot(3,2,6), imhist(I3), axis tight, ...
title('Original Hist + 0.5')
```

TUTORIAL 10: Hist modificatian

- Histogram streching

```
img_limits = stretchlim(I);
I_stretch = imadjust(I,img_limits,[]);
figure; subplot(3,2,1), imshow(I), title('Original Image')
subplot(3,2,2), imhist(I), axis tight, ...
title('Original Histogram')
subplot(3,2,3), imshow(I_stretch), ...
title('Stretched Image')
subplot(3,2,4), imhist(I_stretch), axis tight, ...
title('Stretched Histogram')
% Perform histogram stretching with _imadjust_ using default parameters and
% confirm that the results are identical to the ones obtained before.
I_stretch2 = imadjust(I);
subplot(3,2,5), imshow(I_stretch2), ...
title('Stretched Image')
subplot(3,2,6), imhist(I_stretch2), axis tight, ...
title('Stretched Histogram')
I_stretch_diff = imabsdiff(I_stretch, I_stretch2);
figure, imshow(I_stretch_diff,[])
min(I_stretch_diff(:))
max(I_stretch_diff(:))
```

TUTORIAL 11: Hist modificatian

- Histogram shrinking

```
I = imread('salzburg.png');
I_shrink = imadjust(I,stretchlim(I),[0.25 0.75]);
Figure; subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), axis tight, ...
title('Original Histogram')
subplot(2,2,3), imshow(I_shrink), ...
title('Shrunk Image')
subplot(2,2,4), imhist(I_shrink), axis tight, ...
title('Shrunk Histogram')

% Display the transformation function for the adjustment performed in the
% previous step.

X = reshape(I,1,prod(size(I)));
Y = reshape(I_shrink,1,prod(size(I_shrink)));
figure, plot(X,Y,'.')
xlim([0 255]); ylim([0 255]);
xlabel('Original Image');
ylabel('Adjusted Image');
```

TUTORIAL 12: Hist modificatian

- Histogram shrinking with gamma value

```
I_shrink = imadjust(I,stretchlim(I),[0.25 0.75],2);
X = reshape(I,1,prod(size(I)));
Y = reshape(I_shrink,1,prod(size(I_shrink)));
figure
subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), axis tight, ...
title('Original Histogram')
subplot(2,2,3), imshow(I_shrink), title('Adjusted Image')
subplot(2,2,4), imhist(I_shrink), axis tight, ...
title('Adjusted Histogram')
figure, plot(X,Y,'.'), xlim([0 255]), ylim([0 255])
```

End Part 1:
Spatial Domain Enhancemnet
SCSV 3213