Lecture 2

Intensity Transformation

2019.01.15 -- 2019.01.17

Spatial Domain Process

- Spatial domain representation
  - Image as a function of spatial coordinates $f(x, y)$
- Spatial domain process
  - $g(x, y) = T[f(x, y)]$
- Spatial domain processing is the foundation of many image enhancement/restoration technologies.

Outline

- Intensity Transformation (3.2)
- Histogram Processing (3.3)
- Spatial Filtering (3.4 – 3.6)
  - Linear filters
  - Order-statistics filters

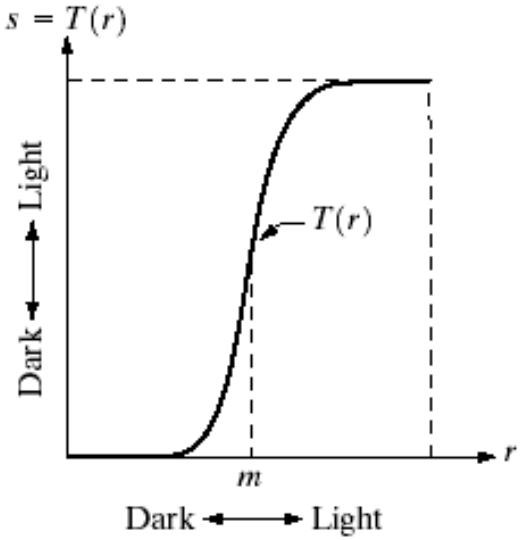
Intensity Transformation

- Definition
  - Pixels are transformed independently.
  - The pixel value $g(x, y)$ is a function of the source pixel value $f(x, y)$.

- The intensity transformation is defined by an intensity transformation function T

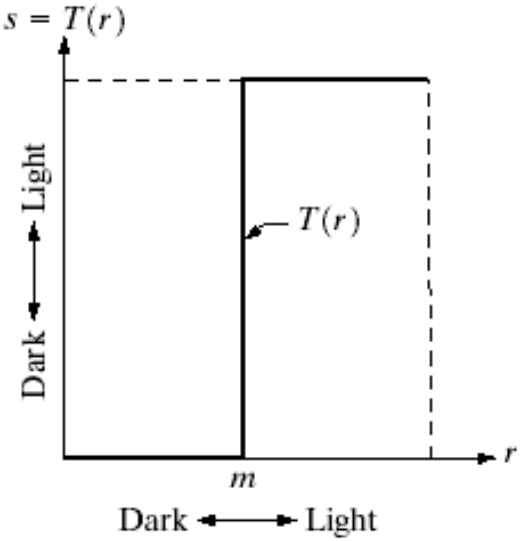
$$s = T(r)$$

Intensity Transformation (Example)



Contrast stretching

Intensity Transformation (Example)

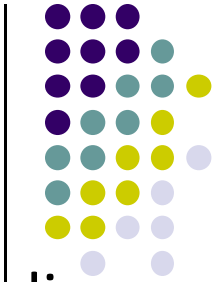


Thresholding

Intensity Transformation Functions

- Image Negatives
- Power-Law transformation
- Piecewise-Linear transformation

Point Operations

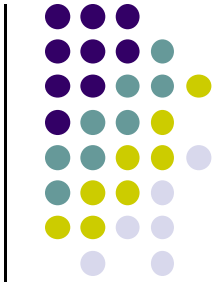


- Point operations changes a pixel's intensity value according to some function (don't care about pixel's neighbor)

$$a' \leftarrow f(a)$$
$$I'(u,v) \leftarrow f\big(I(u,v)\big)$$

- Also called a **homogeneous operation**
- New pixel intensity **depends on**
  - Pixel's previous intensity *I(u,v)*
  - Mapping function *f*( )
- **Does not depend on**
  - Pixel's location *(u,v)*
  - Intensities of neighboring pixels

Some Homogeneous Point Operations



- Addition (Changes brightness)

$$f(p) = p + k \qquad \text{E.g.} \quad f_{\text{bright}}(p) = p + 10$$

- Multiplication (Stretches/shrinks image contrast range)

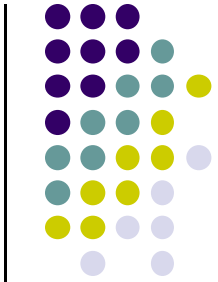
$$f(p) = k \times p \qquad \text{E.g.} \quad f_{\text{contrast}}(p) = p \text{ x } 1.5$$

- Real-valued functions

$$\exp(x), \log(x), (1/x), x^k, \text{ etc.}$$

- Quantizing pixel values
- Global thresholding
- Gamma correction

Point Operation Pseudocode



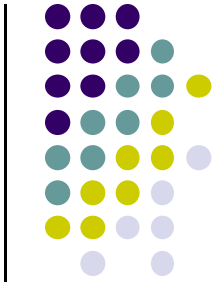
- **Input:** Image with pixel intensities *I(u,v)* defined on [1 .. *w*] x [1 .. *H*]
- **Output:** Image with pixel intensities *I'(u,v)*

for v = 1 .. *h*

    for u = 1 .. *w*

        set *I(u, v) = f (I(u,v))*

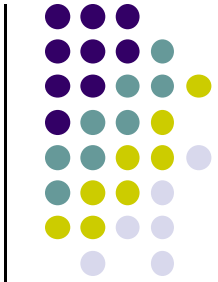
Non-Homogeneous Point Operation



- New pixel value depends on:
  - Old value + **pixel's location *(u,v)***

$$a' \leftarrow g(a, u, v)$$
$$I'(u, v) \leftarrow g\left(I(u, v), u, v\right)$$

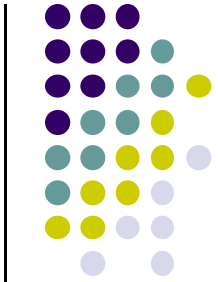
Clamping



- Deals with pixel values outside displayable range
  - If (a > 255) a = 255;
  - If (a < 0) a = 0;
- Function below will **clamp** (force) all values to fall within range [*a,b*]

$$f(p) = \begin{cases} a & \text{if } p < a \\ p & \text{if } a \leq p \leq b \\ b & \text{if } p > b \end{cases}$$

Example: Modify Intensity and Clamp

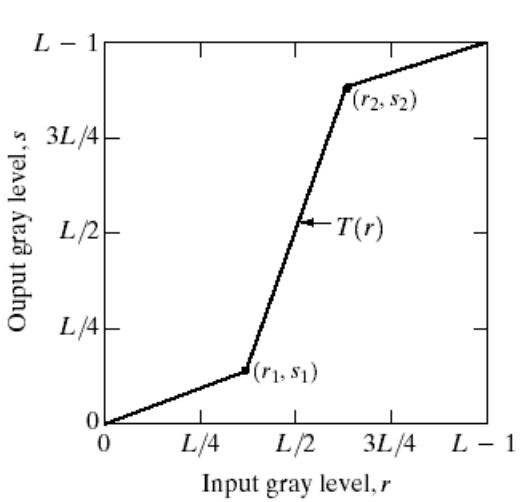


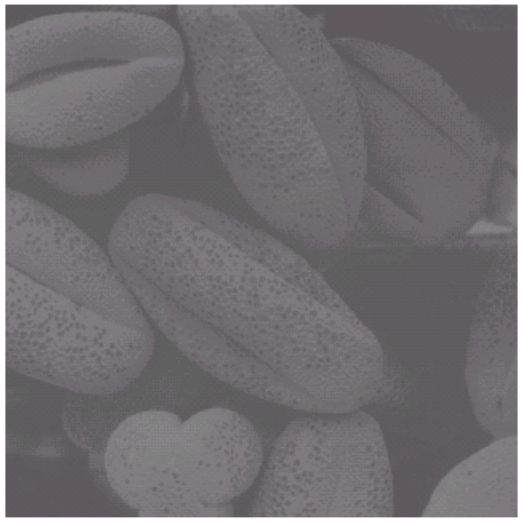
- Point operation: increase image contrast by 50% then clamp values above 255

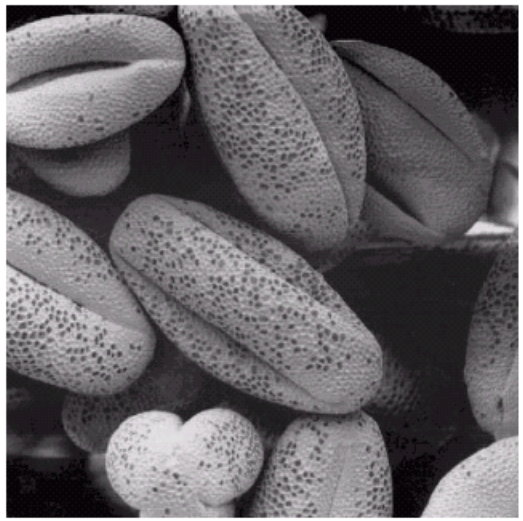
```
1   public void run(ImageProcessor ip) {
2     int w = ip.getWidth();
3     int h = ip.getHeight();
4
5     for (int v = 0; v < h; v++) {
6       for (int u = 0; u < w; u++) {
7         int a = (int) (ip.get(u, v) * 1.5 + 0.5);
8         if (a > 255)
9           a = 255;    // clamp to maximum value
10        ip.set(u, v, a);
11      }
12    }
13  }
```

Increase contrast by 50%

Piecewise-Linear Contrast Stretching







a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Contrast Stretching: Math

Equations

$$\text{Let} \quad T(r) = a \cdot r + b$$

$$a \cdot r_1 + b = s_1$$

$$a \cdot r_2 + b = s_2$$

Solutions

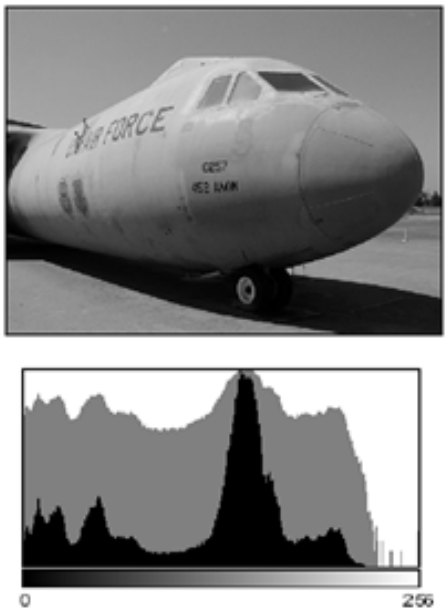
$$a = \frac{s_2 - s_1}{r_2 - r_1}$$

$$b = \frac{s_1 r_2 - s_2 r_1}{r_2 - r_1}$$

Inverting Images

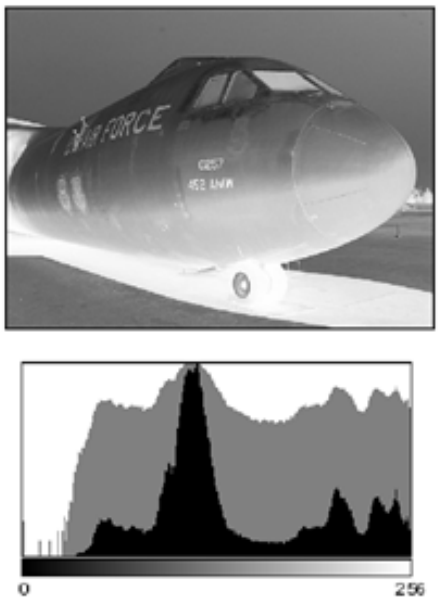
$$f_{\text{invert}}(a) = -a + a_{\max} = a_{\max} - a$$

- 2 steps
  1. Multiple intensity by -1
  2. Add constant (e.g. a_{max}) to put result in range [0,a_{max}]



(a)

Original

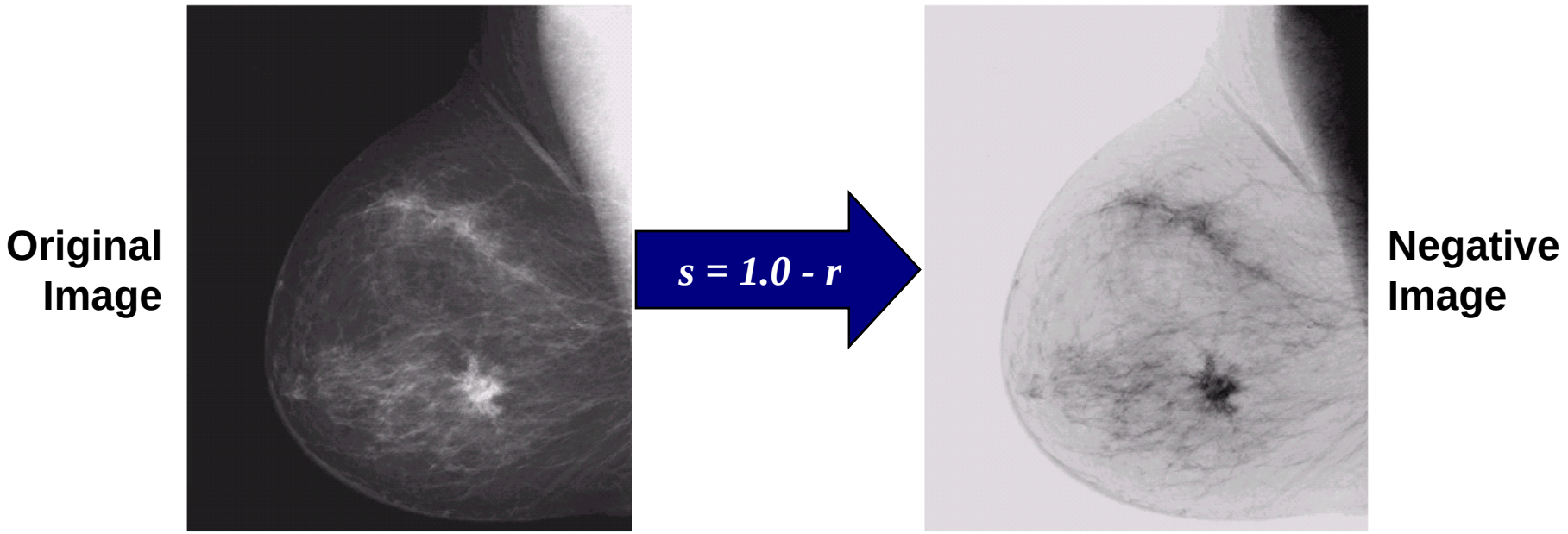


(c)

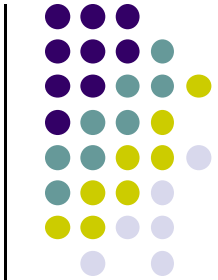
Inverted Image

Image Negatives (Inverted Images)

- Image negatives useful for enhancing white or grey detail embedded in dark regions of an image
  - Note how much clearer the tissue is in the negative image of the mammogram below



Thresholding



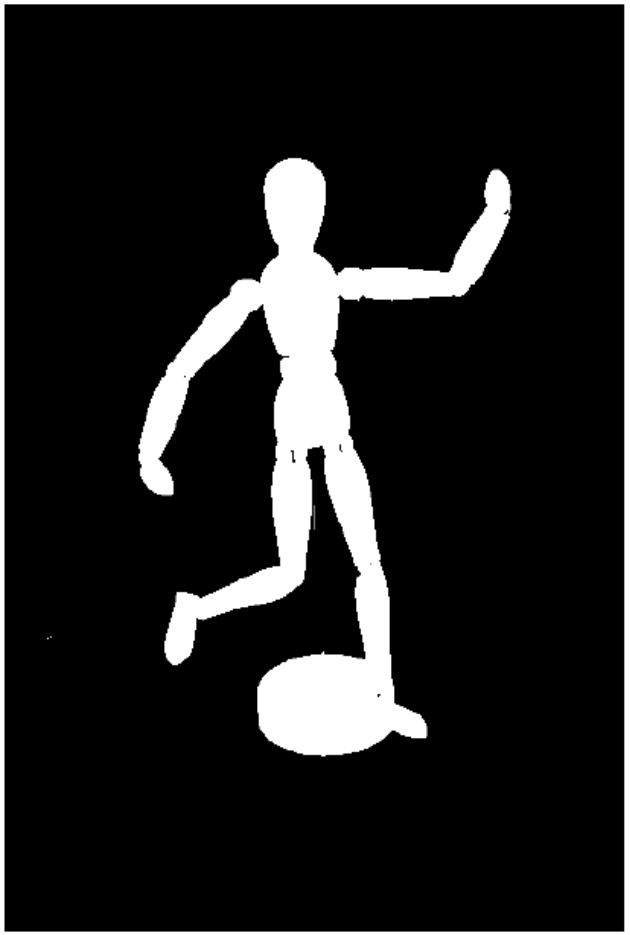
- Input values below **threshold** a_{th} set to a_0
- Input values above **threshold** a_{th} set to a_1

$$f_{\text{threshold}}(a) = \begin{cases} a_0 & \text{for } a < a_{\text{th}} \\ a_1 & \text{for } a \geq a_{\text{th}} \end{cases}$$

- Converts grayscale image to binary image (binarization) if
  - $a_0 = 0$
  - $a_1 = 1$
- Implemented as `im2bw()` and `multithresh()` ***https://goo.gl/fpWRvZ***

Thresholding Example

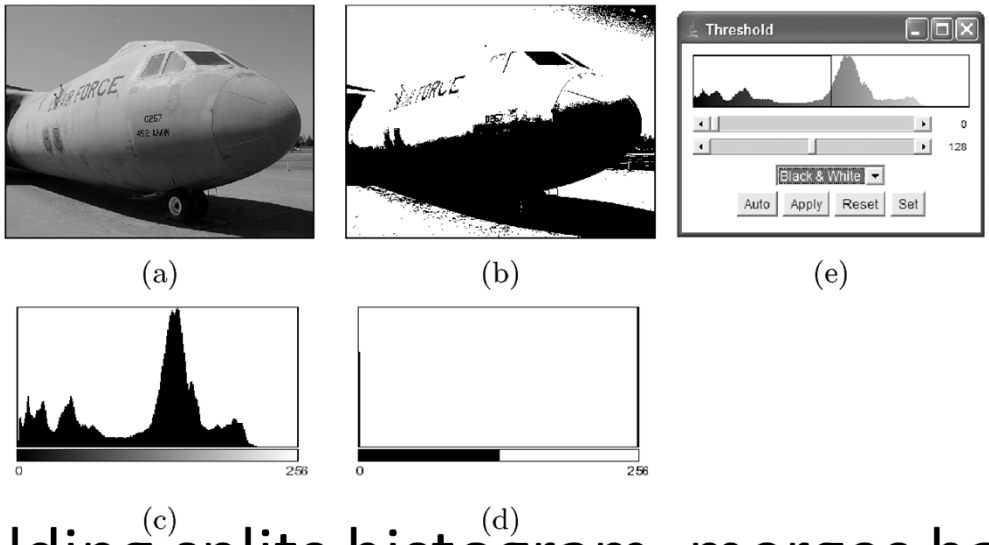
Original Image



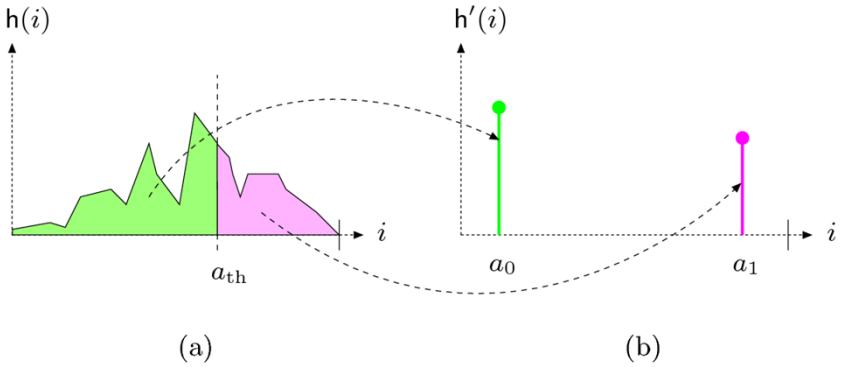
Thresholded Image

Thresholding and Histograms

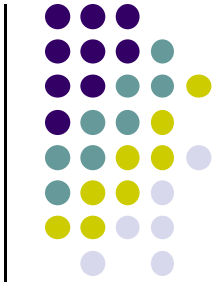
- Example with a_{th} = 128



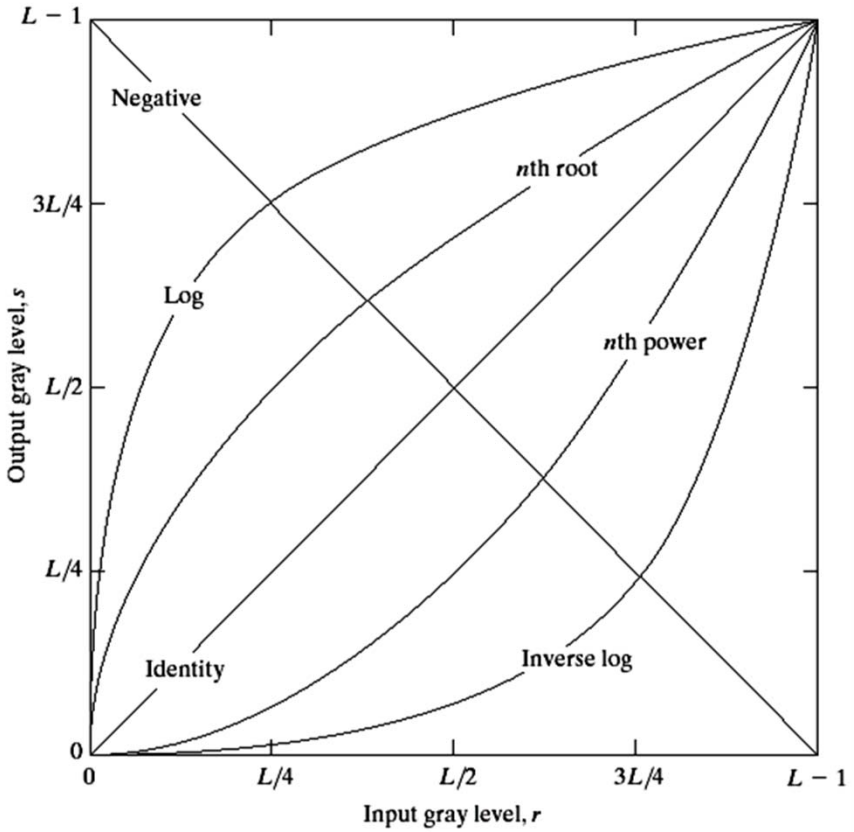
- Thresholding splits histogram, merges halves into a_0 a_1



Basic Grey Level Transformations

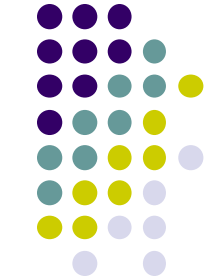


- 3 most common gray level transformation:
  - Linear
    - Negative/Identity
  - Logarithmic
    - Log/Inverse log
  - Power law
    - n^{th} power/n^{th} root



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

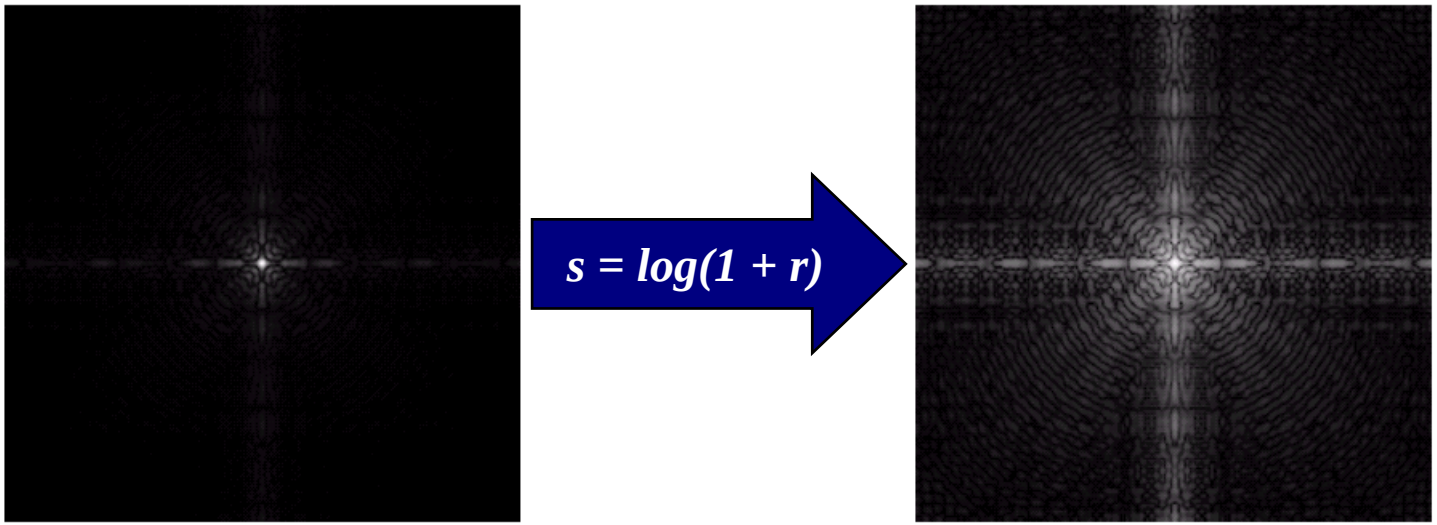
Logarithmic Transformations



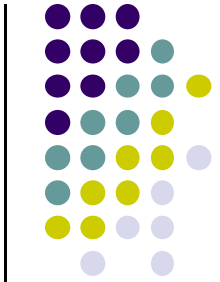
- Maps narrow range of input levels => wider range of output values
- Inverse log transformation does opposite transformation
- The general form of the log transformation is

New pixel value ⟶ $s = c * log(1 + r)$ ⟵ Old pixel value

- Log transformation of Fourier transform shows more detail



Power Law Transformations



- Power law transformations have the form

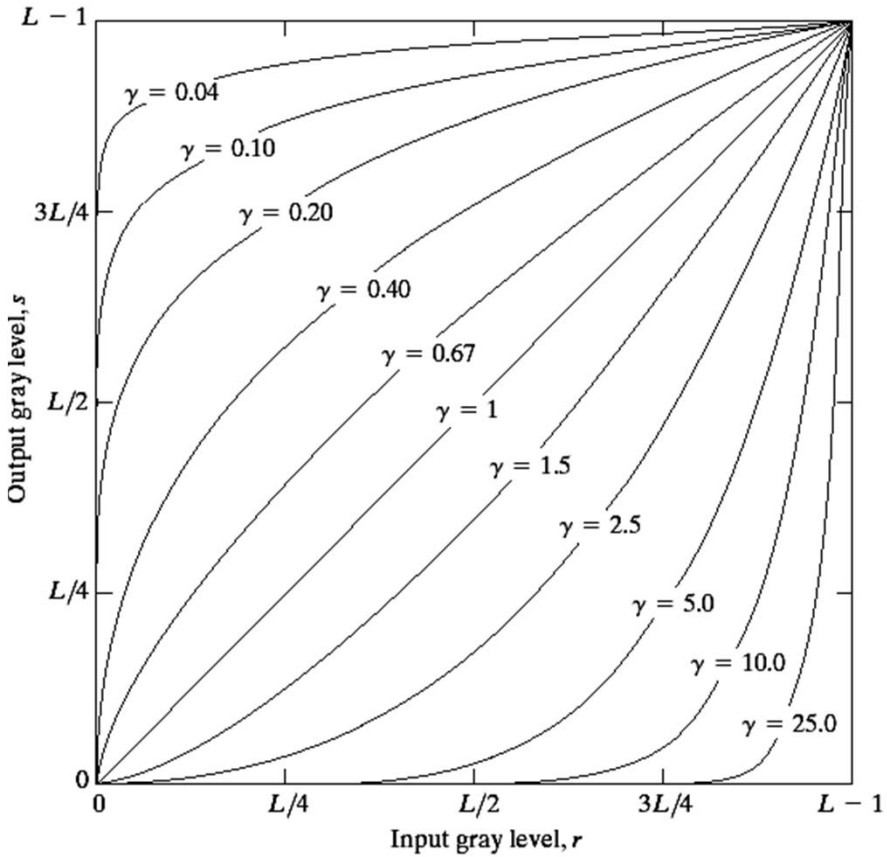
$$s = c * r^{\gamma}$$

Power

New pixel value

Constant

Old pixel value



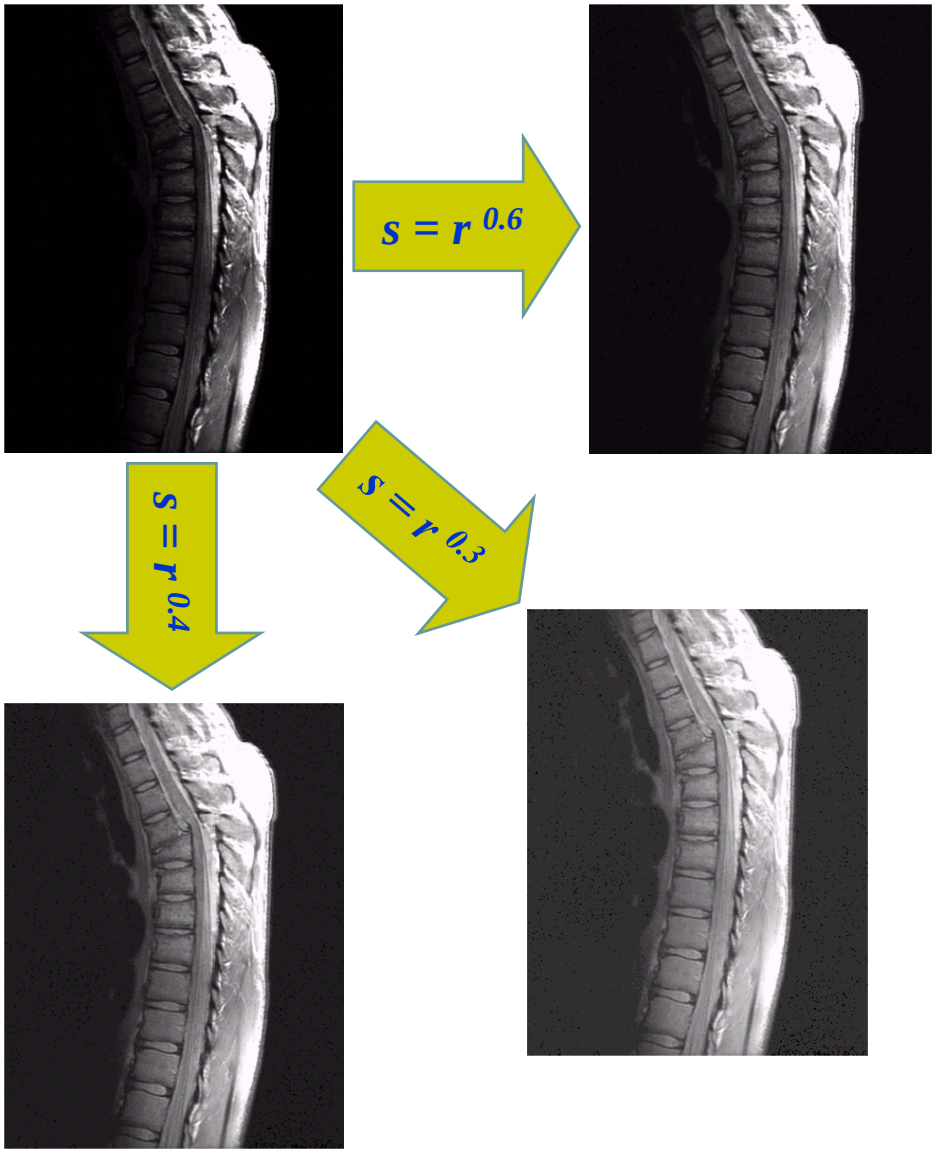
- Map narrow range of dark input values into wider range of output values or vice versa

- Varying γ gives a whole family of curves

Power Law Example

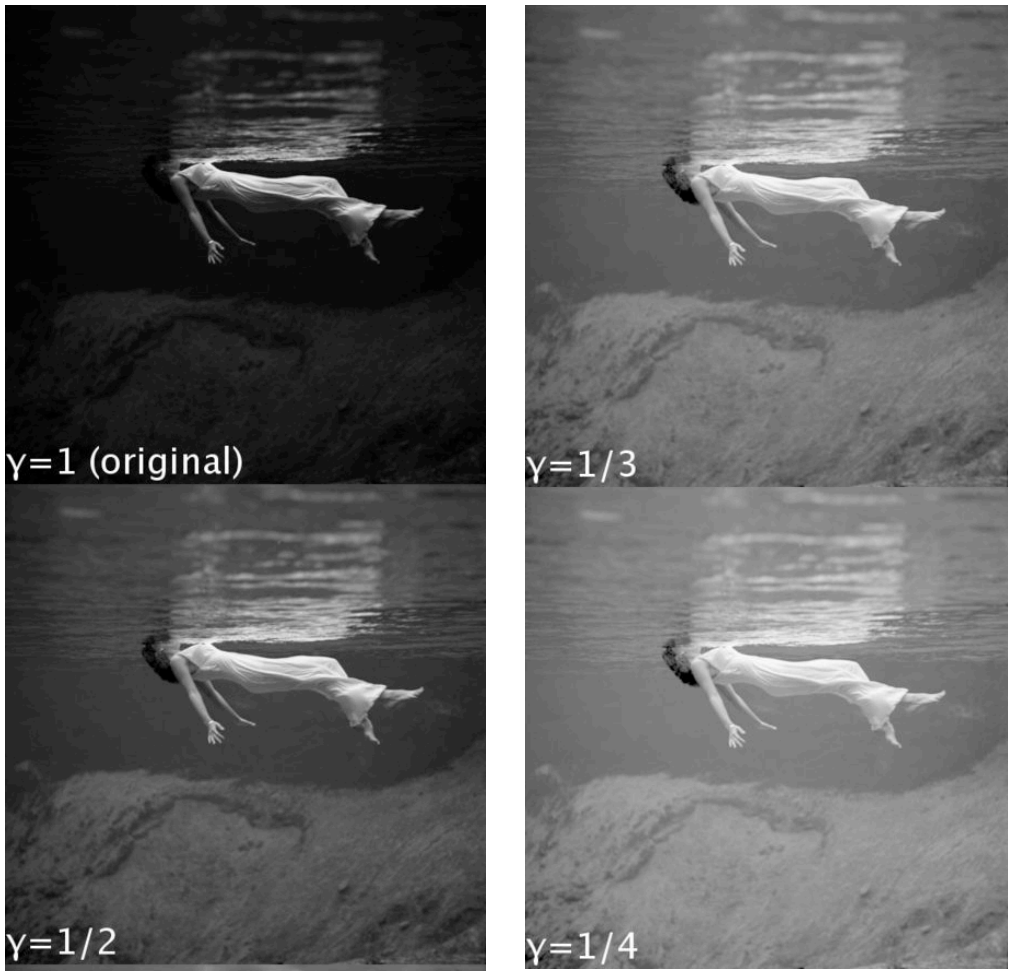
- Magnetic Resonance (MR) image of fractured human spine

- Different power values highlight different details



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Power Law



When $\gamma < 1$, stretch the darker range

Power Law

When $\gamma > 1$, stretch the lighter range

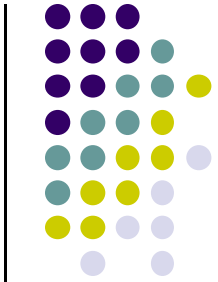
Original image

$\gamma = 3.0$

$\gamma = 4.0$

$\gamma = 5.0$

Intensity Windowing

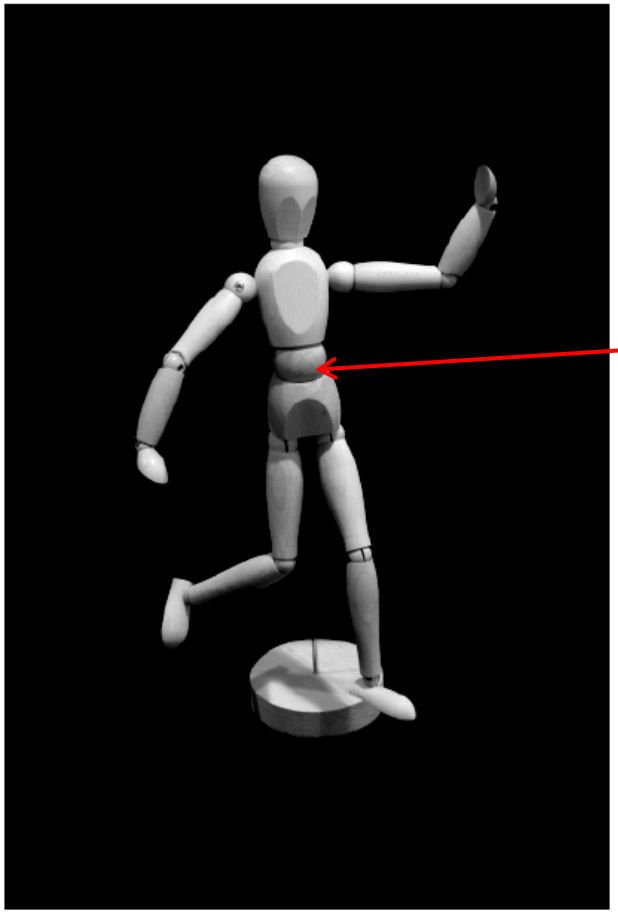


- A clamp operation, then linearly stretching image intensities to fill possible range
- To window an image in [*a,b*] with max intensity ***M***

$$f(p) = \begin{cases} 0 & \text{if } p < a \\ M \times \frac{p-a}{b-a} & \text{if } a \leq p \leq b \\ M & \text{if } p > b \end{cases}$$

Intensity Windowing Example

Original Image



Windowed Image

Contrasts
easier to see

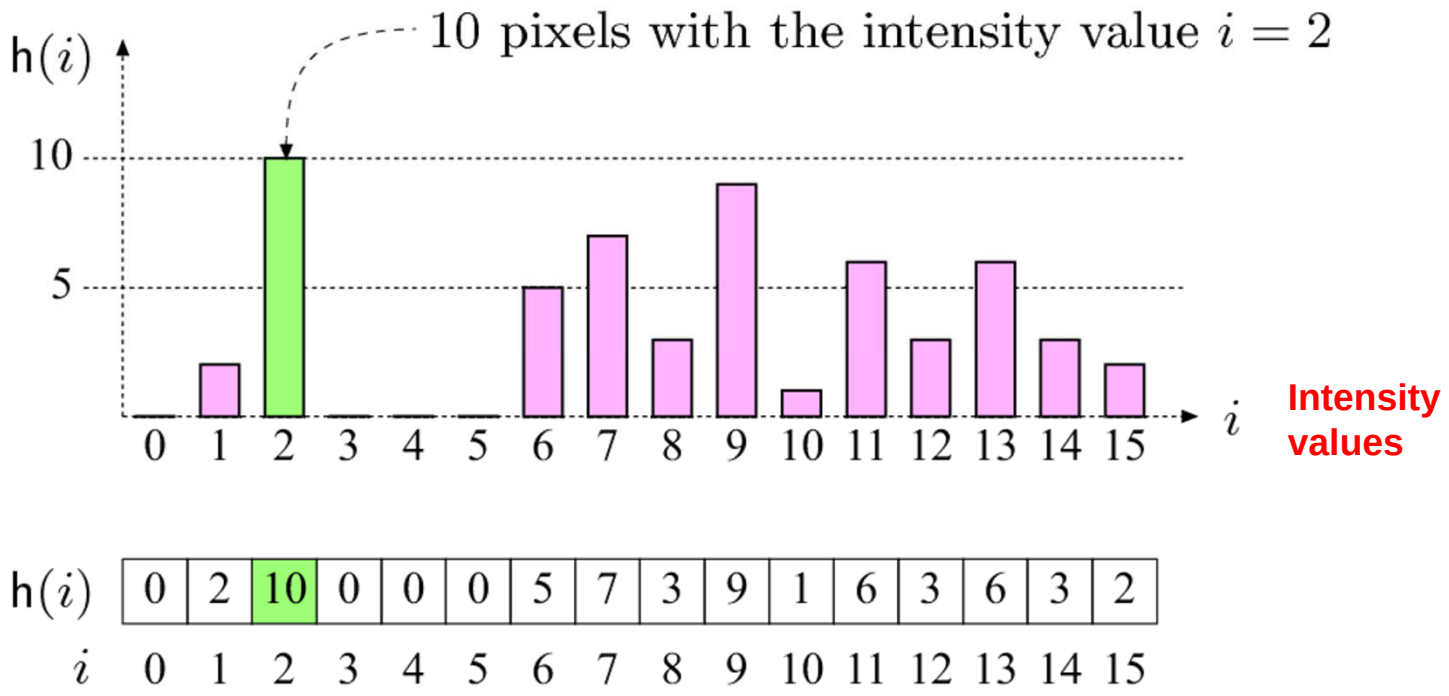
Histograms

- Histograms plots how many times (frequency) each intensity value in image occurs
- Example:
  - Image (left) has 256 distinct gray levels (8 bits)
  - Histogram (right) shows frequency (how many times) each gray level occurs

Histograms

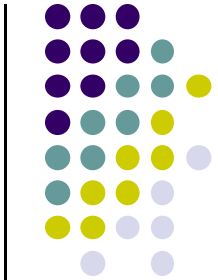
- Many cameras display real time histograms of scene
- Helps avoid taking over-exposed pictures
- Also easier to detect types of processing previously applied to image

Histograms

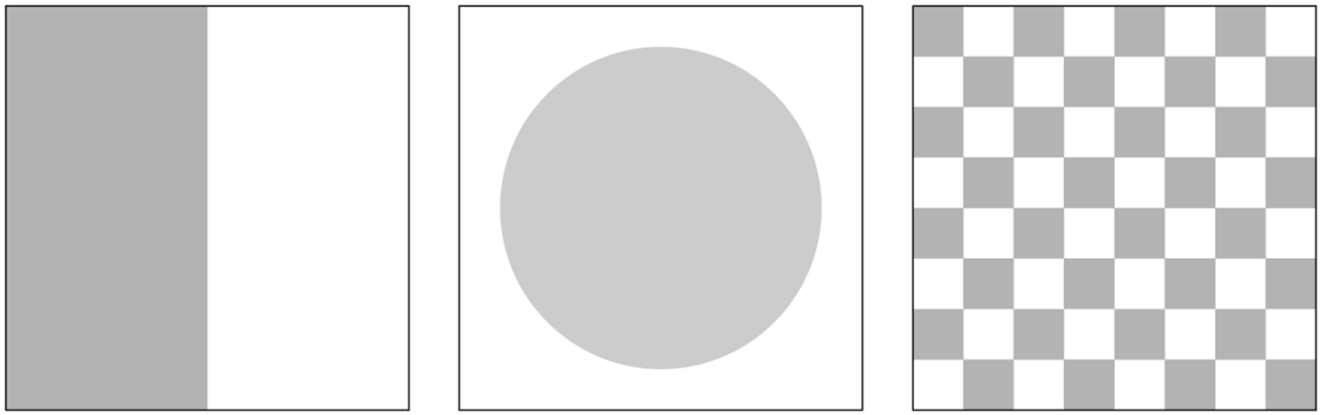


- E.g. K = 16, 10 pixels have intensity value = 2
- Histograms: only statistical information
- No indication of **location** of pixels

Histograms

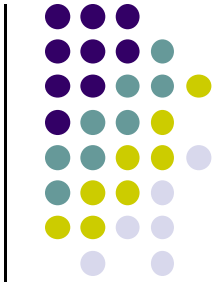


- Different images can have **same** histogram
- 3 images below have same histogram



- Half of pixels are gray, half are white
  - Same histogram = same statisics
  - Distribution of intensities could be different
- Can we reconstruct image from histogram? No!

Histograms



- So, a histogram for a grayscale image with intensity values in range

$$I(u,v) \in [0, K-1]$$

would contain exactly *K* entries

- E.g. 8-bit grayscale image, $K = 2^8 = 256$
- Each histogram entry is defined as:

  h(i) = number of pixels with intensity *I* for all $0 < i < K$.

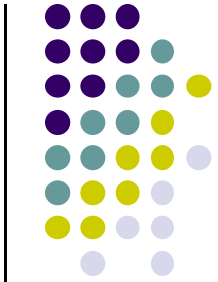
- E.g: h(255) = number of pixels with intensity = 255
- Formal definition

$$\mathrm{h}(i) = \mathrm{card}\{(u,v) \mid I(u,v) = i\}$$

Number (size of set) of pixels

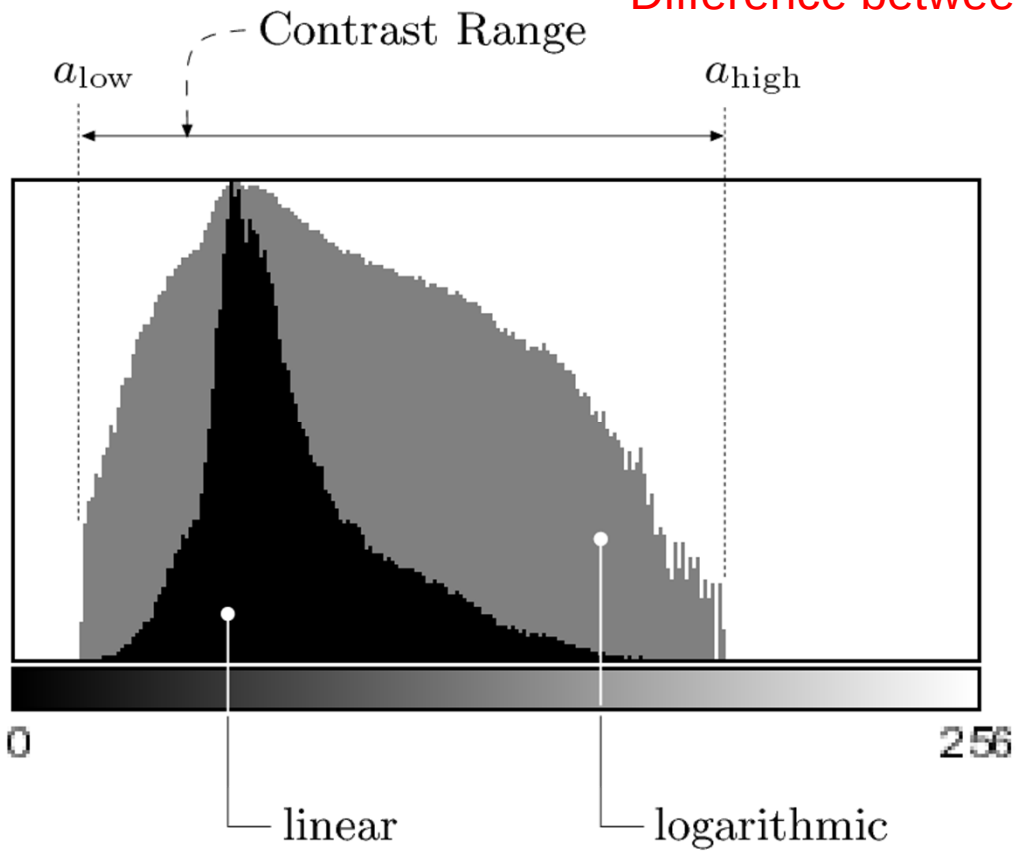
such that

Interpreting Histograms

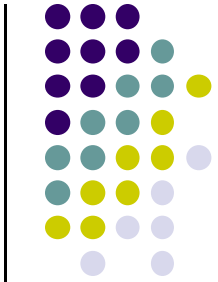


- Log scale makes low values more visible

Difference between darkest and lightest



Histograms

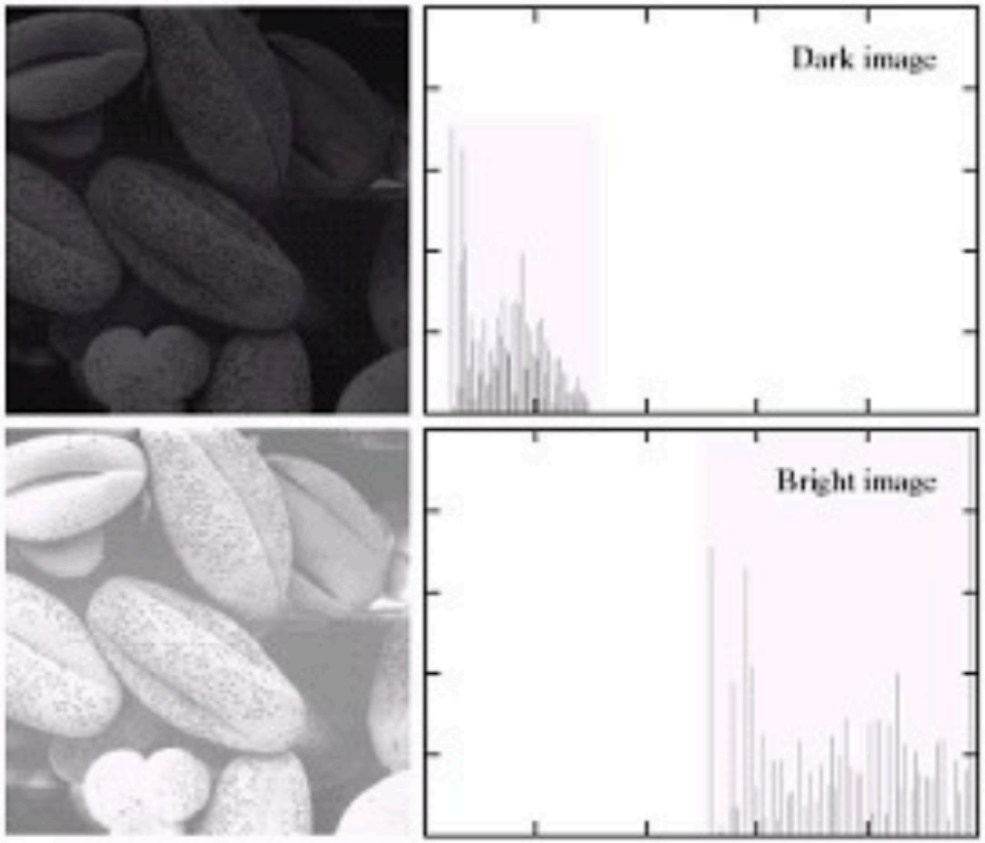


- Histograms help detect image acquisition issues
- Problems with image can be identified on histogram
  - Over and under exposure
  - Brightness
  - Contrast
  - Dynamic Range
- Point operations can be used to alter histogram. E.g
  - Addition
  - Multiplication
  - Exp and Log
  - Intensity Windowing (Contrast Modification)

Significance of Histogram

- The basics for many spatial domain processing techniques.
- Often used for image enhancement.
- Information contained in the histograms is also useful for image compression.

Histogram Examples



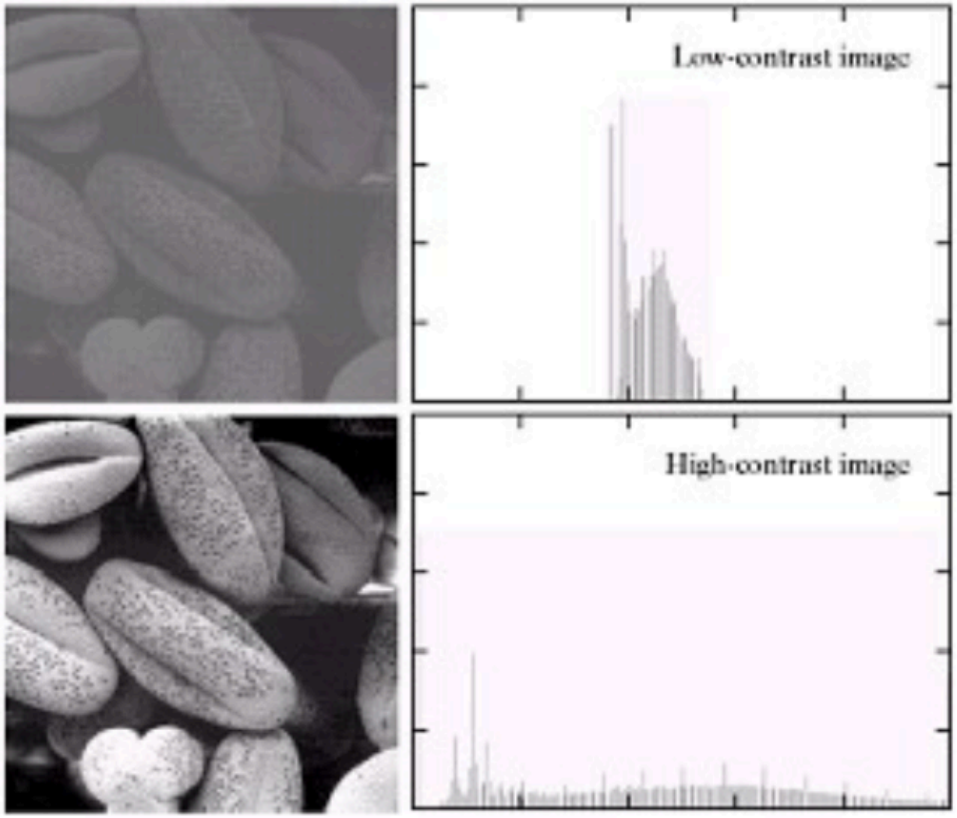
Dark image

Components of histogram are concentrated on the low side of the gray scale.

Bright image

Components of histogram are concentrated on the high side of the gray scale.

Histogram Examples



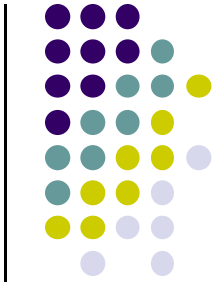
Low-contrast image

histogram is narrow and centered toward the middle of the gray scale

High-contrast image

histogram covers broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others

Image Brightness



- Brightness of a grayscale image is the **average intensity** of all pixels in image

$$B(I) = \frac{1}{wh} \sum_{v=1}^{h} \sum_{u=1}^{w} I(u, v)$$

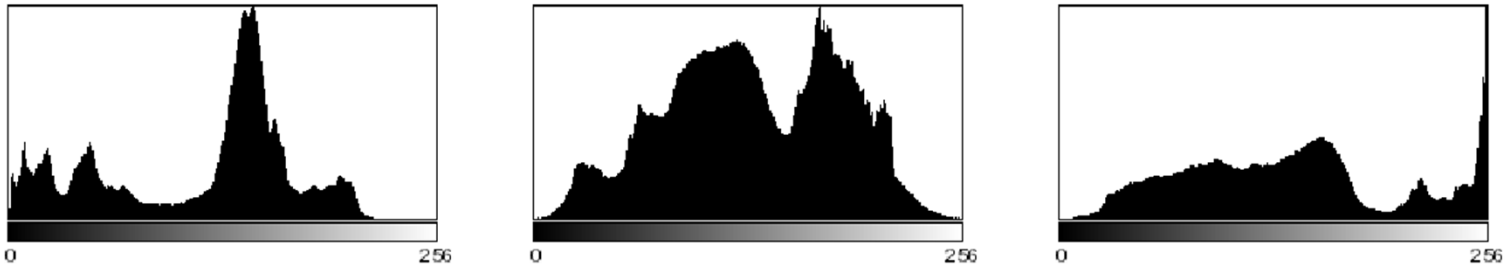
2. Divide by total number of pixels

1. Sum up all pixel intensities

Detecting Bad Exposure using Histograms

Exposure? Are intensity values spread (good) out or bunched up (bad)

Image



Histogram

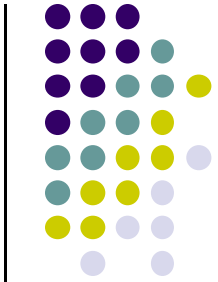
(a) (b) (c)

Underexposed

Properly Exposed

Overexposed

Image Contrast

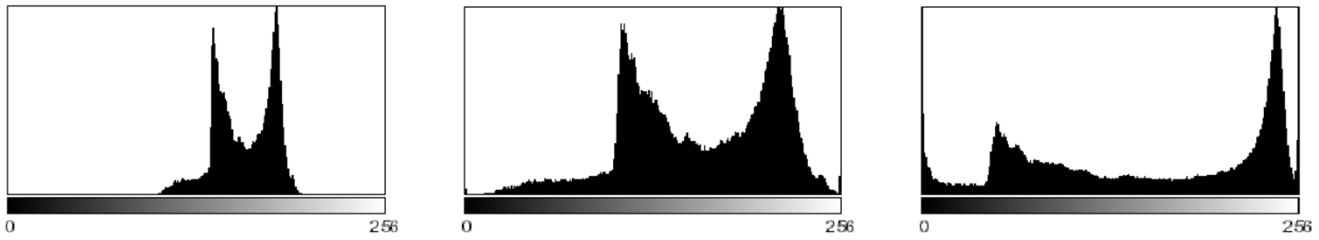


- The contrast of a grayscale image indicates how easily objects in the image can be distinguished
- **High contrast image:** many distinct intensity values
- **Low contrast:** image uses few intensity values

Histograms and Contrast

Good Contrast? Widely spread intensity values
+ large difference between min and max intensity values

Image



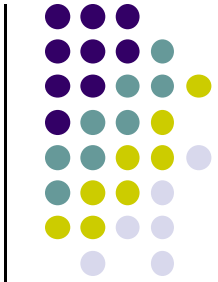
Histogram

(a) **Low contrast**

(b) **Normal contrast**

(c) **High contrast**

Contrast Equation?



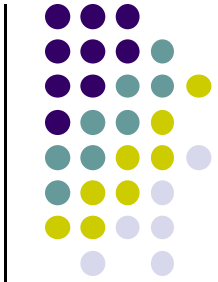
- Many different equations for contrast exist
- Examples:

$$\text{Contrast} = \frac{\text{Change in Luminance}}{\text{Average Luminance}}$$

- Michalson's equation for contrast

$$C_M(I) = \frac{\max(I) - \min(I)}{\max(I) + \min(I)}$$

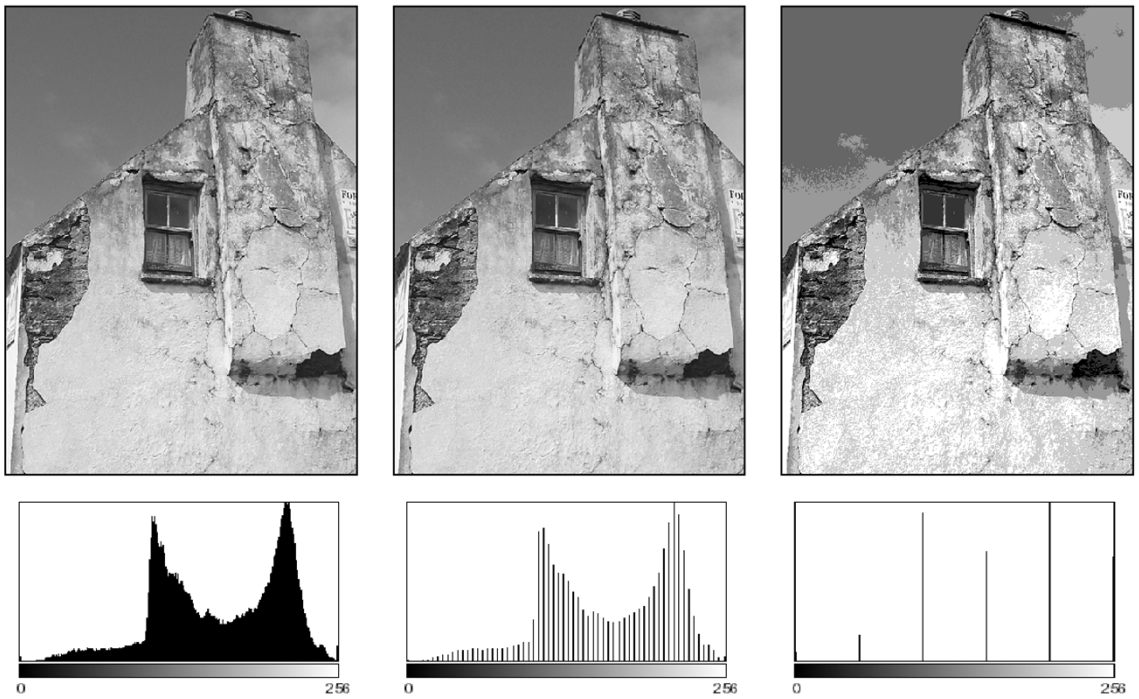
Contrast Equation?



- These equations work well for simple images with 2 luminances (i.e. uniform foreground and background)
- Does not work well for complex scenes with many luminances or if min and max intensities are small

Histograms and Dynamic Range

- **Dynamic Range:** Number of distinct pixels in image



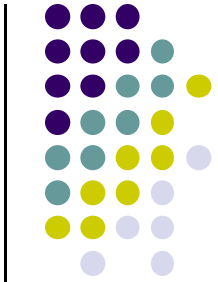
(a) **High Dynamic Range**

(b) **Low Dynamic Range (64 intensities)**

(c) **Extremely low Dynamic Range (6 intensity values)**

- Difficult to increase image dynamic range (e.g. interpolation)
- HDR (12-14 bits) capture typical, then down-sample

High Dynamic Range Imaging



- **High dynamic range** means very bright and very dark parts in a single image (many distinct values)
- Dynamic range in photographed scene may exceed number of available bits to represent pixels
- Solution:
  - Capture multiple images at different exposures
  - Combine them using image processing

Detecting Image Defects using Histograms

- No “best” histogram shape, depends on application
- Image defects
  - **Saturation:** scene illumination values outside the sensor’s range are set to its min or max values => results in spike at ends of histogram
  - **Spikes and Gaps in manipulated images** (not original).

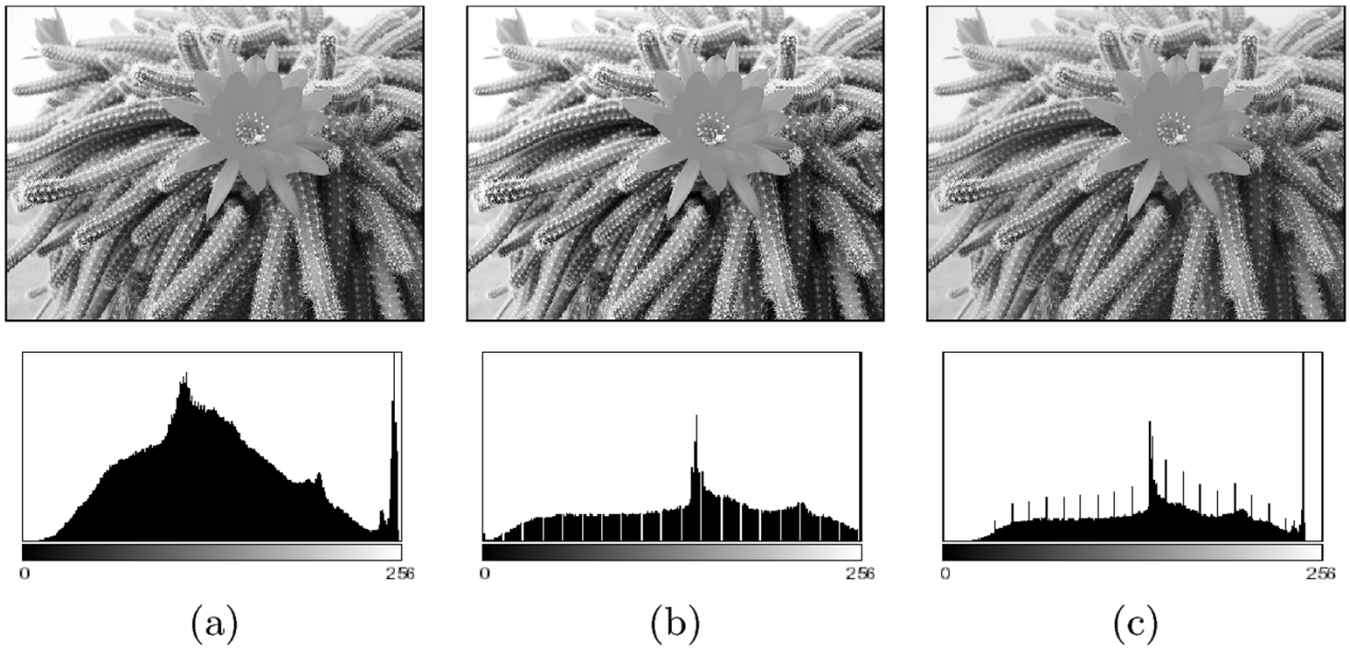
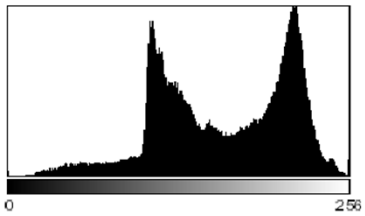


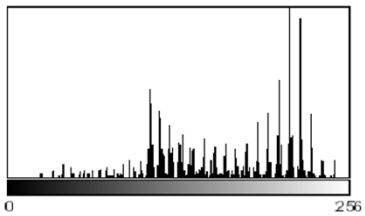
Image Defects: Effect of Image Compression

- Histograms show impact of image compression
- Example: in GIF compression (GIFs compress by removing horizontal redundancy. *https://goo.gl/4FKE3L*), dynamic range is reduced to only few intensities (quantization)

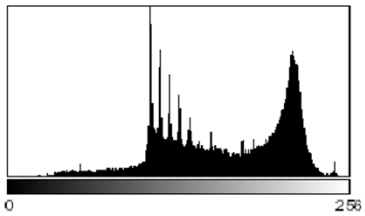
Original Image



(a) **Original Histogram**



(b) **Histogram after GIF conversion**



(c) **Fix? Scaling image by 50% and Interpolating values recreates some lost colors**
But GIF artifacts still visible

Effect of Image Compression

- Example: Effect of JPEG compression on line graphics

  (PNG behaves like GIF in that it compresses horizontal patterns, but it also handles vertical patterns and dithering.)

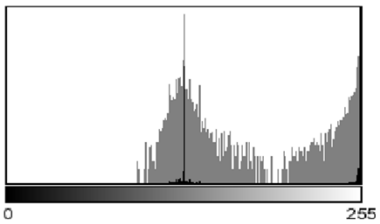
- JPEG compression designed for color images

(a)

(b)

Original histogram has only 2 intensities (gray and white)

(c)

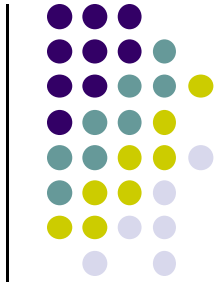


(d)

JPEG image appears dirty, fuzzy and blurred

Its Histogram contains gray values not in original

Computing Histograms



```
1  public class Compute_Histogram implements PlugInFilter {
2
3      public int setup(String arg, ImagePlus img) {
4          return DOES_8G + NO_CHANGES;
5      }
6
7      public void run(ImageProcessor ip) {
8          int[] H = new int[256]; // histogram array
9          int w = ip.getWidth();
10         int h = ip.getHeight();
11
12         for (int v = 0; v < h; v++) {
13             for (int u = 0; u < w; u++) {
14                 int i = ip.getPixel(u,v);
15                 H[i] = H[i] + 1;
16             }
17         }
18         ... //histogram H[] can now be used
19     }
20
21 } // end of class Compute_Histogram
```

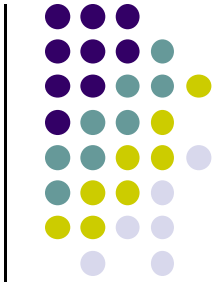
Receives 8-bit image, Will not change it

Create array to store histogram computed

Get width and height of image

Iterate through image pixels, add each intensity to appropriate histogram bin

Large Histograms: Binning



- High resolution image can yield very large histogram
- Example: 32-bit image = 2^{32} = 4,294,967,296 columns
- Such a large histogram impractical to display
- Solution? Binning!
  - Combine **ranges of intensity values** into histogram columns

So, given the image $I : \Omega \rightarrow [0, K-1]$, the binned histogram for I is the function

$$h(i) = \text{card}\{(u,v) | \; a_i \leq I(u,v) < a_{i+1}\},$$

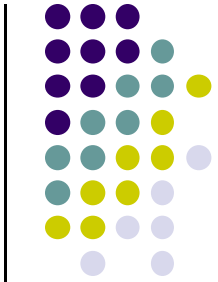
where $0 = a_0 < a_1 < \ldots < a_B = K$.

Number (size of set) of pixels

such that

Pixel's intensity is between a_i and a_{i+1}

Calculating Bin Size

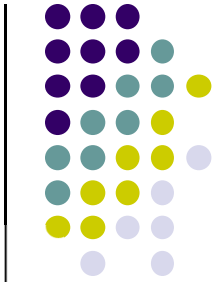


- Typically use equal sized bins
- Bin size? $\dfrac{\textit{Number of distinct values in image}}{\textit{Number of bins}}$
- Example: To create 256 bins from 14-bit image

$$\textit{Bin size} = \frac{2^{14}}{256} = 64$$

$$\begin{array}{lcrcr}
\mathsf{h}(0) & \leftarrow & 0 & \leq I(u,v) < & 64 \\
\mathsf{h}(1) & \leftarrow & 64 & \leq I(u,v) < & 128 \\
\mathsf{h}(2) & \leftarrow & 128 & \leq I(u,v) < & 192 \\
\vdots & & \vdots & \vdots & \vdots \\
\mathsf{h}(j) & \leftarrow & a_j & \leq I(u,v) < & a_{j+1} \\
\vdots & & \vdots & \vdots & \vdots \\
\mathsf{h}(255) & \leftarrow & 16320 & \leq I(u,v) < & 16384
\end{array}$$

Binned Histogram



- To calculate which bin a pixel's intensity belongs to

$$\frac{I(u,v)}{k_B} = \frac{I(u,v)}{K/B} = I(u,v) \cdot \frac{B}{K}$$

- Previous example, B = 256, K = 2^{14} = 16384

```
1   int[] binnedHistogram(ImageProcessor ip) {
2     int K = 256; // number of intensity values
3     int B = 32; // size of histogram, must be defined
4     int[] H = new int[B]; // histogram array
5     int w = ip.getWidth();
6     int h = ip.getHeight();
7
8     for (int v = 0; v < h; v++) {
9       for (int u = 0; u < w; u++) {
10        int a = ip.getPixel(u, v);
11        int i = a * B / K; // integer operations only!
12        H[i] = H[i] + 1;
13      }
14    }
15    // return binned histogram
16    return H;
17  }
```

Create array to store histogram computed

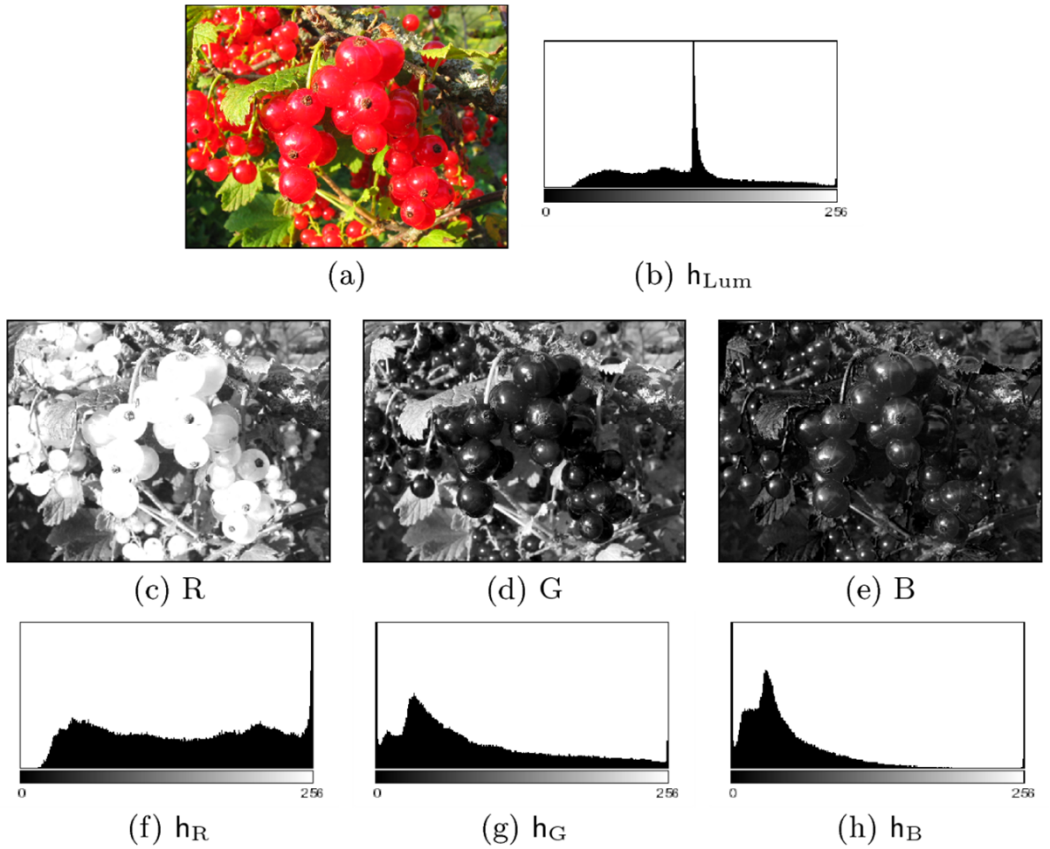
Calculate which bin to add pixel's intensity

Increment corresponding histogram

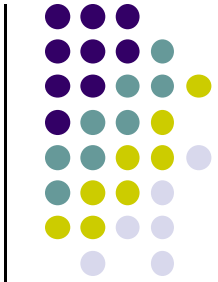
Color Image Histograms

Two types:

1. **Intensity histogram:**
   - Convert color image to gray scale
   - Display histogram of gray scale
2. **Individual Color Channel Histograms:**
   3 histograms (R,G,B)

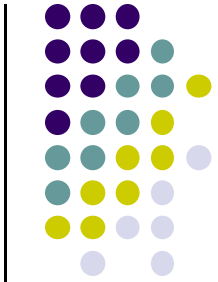


Color Image Histograms



- Both types of histograms provide useful information about lighting, contrast, dynamic range and saturation effects
- No information about the actual color distribution!
- Images with totally different RGB colors can have same R, G and B histograms
- Solution to this ambiguity is the **Combined Color Histogram**.
  - More on this later

Cumulative Histogram



- Useful for certain operations (e.g. histogram equalization) later
- Analogous to the **Cumulative Density Function (CDF)**
- Definition:

$$\mathsf{H}(i) = \sum_{j=0}^{i} \mathsf{h}(j) \quad \text{for} \;\; 0 \le i < K$$

- Recursive definition

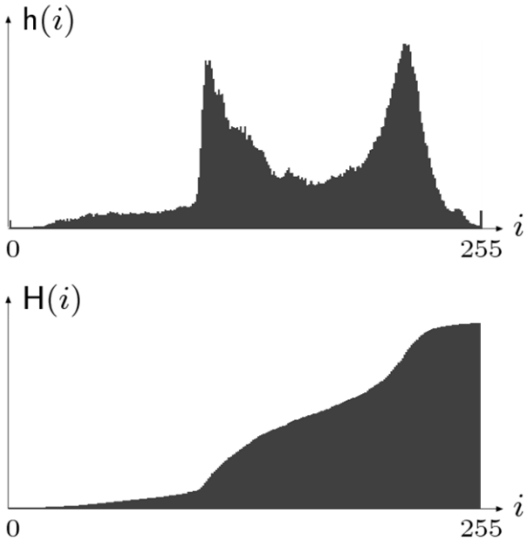
$$\mathsf{H}(i) = \begin{cases} \mathsf{h}(0) & \text{for} \;\; i = 0 \\ \mathsf{H}(i-1) + \mathsf{h}(i) & \text{for} \;\; 0 < i < K \end{cases}$$

- Monotonically increasing

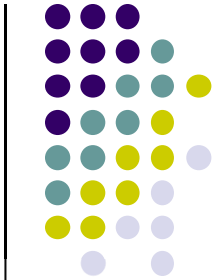
$$\mathsf{H}(K-1) = \sum_{j=0}^{K-1} \mathsf{h}(j) = M \cdot N$$

Last entry of Cum. histogram

Total number of pixels in image



Automatic Contrast Adjustment



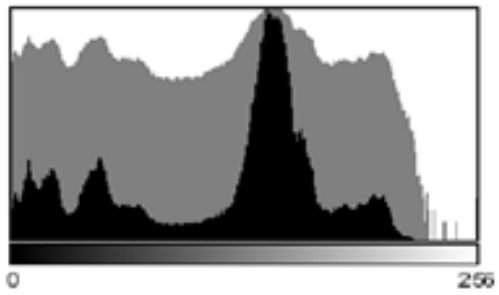
- Point operation that modifies pixel intensities such that available range of values is fully covered
- Algorithm:
  - Find high and lowest pixel intensities a_{low}, a_{high}
  - Linear stretching of intensity range

$$f_{\text{ac}}(a) = a_{\text{min}} + (a - a_{\text{low}}) \cdot \frac{a_{\text{max}} - a_{\text{min}}}{a_{\text{high}} - a_{\text{low}}}$$

If a_{min} = 0 and a_{max} = 255

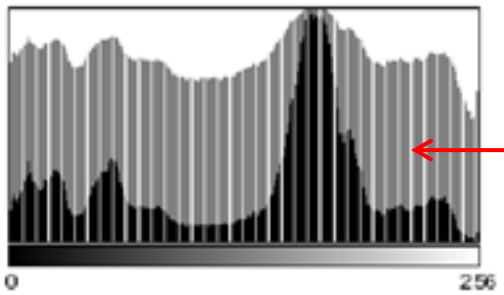
$$f_{\text{ac}}(a) = (a - a_{\text{low}}) \cdot \frac{255}{a_{\text{high}} - a_{\text{low}}}$$

Effects of Automatic Contrast Adjustment



(a)

Original

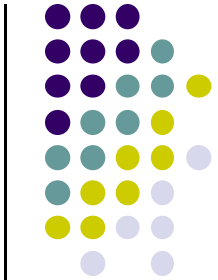


(b)

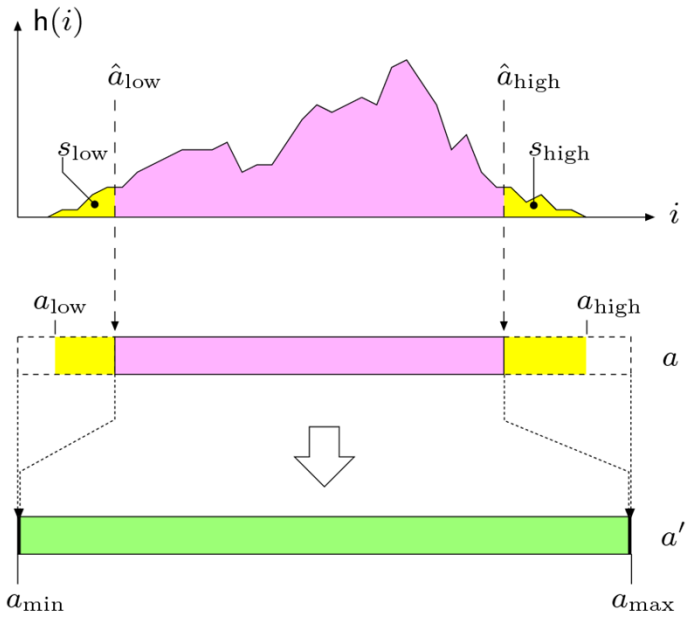
Result of automatic Contrast Adjustment

Linearly stretching range causes gaps in histogram

Modified Contrast Adjustment



- Better to map only certain range of values
- Get rid of tails (usually noise) based on predefined percentiles (s_{low}, s_{high})



$$\hat{a}_{\text{low}} = \min\{\, i \mid \mathsf{H}(i) \geq M \cdot N \cdot s_{\text{low}} \}$$

$$\hat{a}_{\text{high}} = \max\{\, i \mid \mathsf{H}(i) \leq M \cdot N \cdot (1 - s_{\text{high}}) \}$$

$$f_{\text{mac}}(a) = \begin{cases} a_{\text{min}} & \text{for } a \leq \hat{a}_{\text{low}} \\ a_{\text{min}} + (a - \hat{a}_{\text{low}}) \cdot \dfrac{a_{\text{max}} - a_{\text{min}}}{\hat{a}_{\text{high}} - \hat{a}_{\text{low}}} & \text{for } \hat{a}_{\text{low}} < a < \hat{a}_{\text{high}} \\ a_{\text{max}} & \text{for } a \geq \hat{a}_{\text{high}} \end{cases}$$

Histogram Equalization

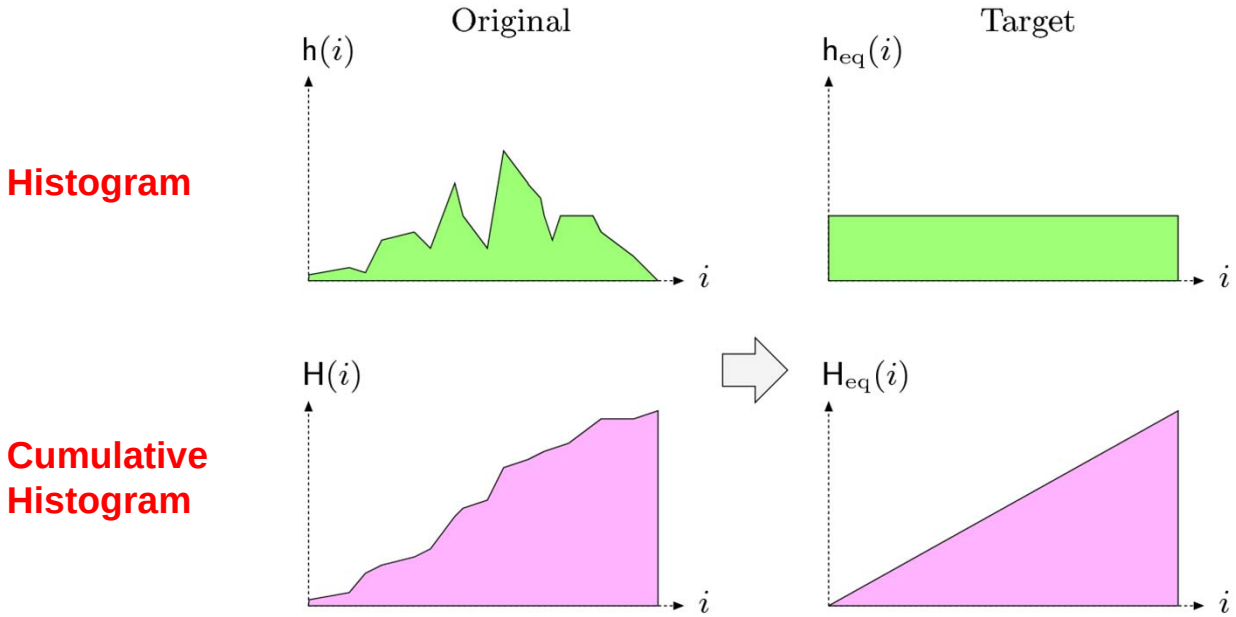
- As the low-contrast image's histogram is narrow, if we distribute the histogram to wider range, the quality of the image will be improved.

- We can do it by adjusting the probability density function of the original histogram so that the probability spreads equally.

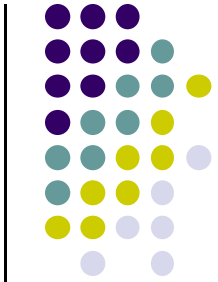
- The histogram equalization is an approach to enhance an image. Specifically, it designs a transformation $T(\cdot)$, such that the output values are uniformly distributed in $[0, L-1]$.

Histogram Equalization

- Adjust 2 different images to make their histograms (intensity distributions) similar
- Apply a point operation that changes histogram of modified image into **uniform distribution**



Histogram Equalization



Spreading out the frequencies in an image (or equalizing the image) is a simple way to improve dark or washed out images

Can be expressed as a transformation of histogram

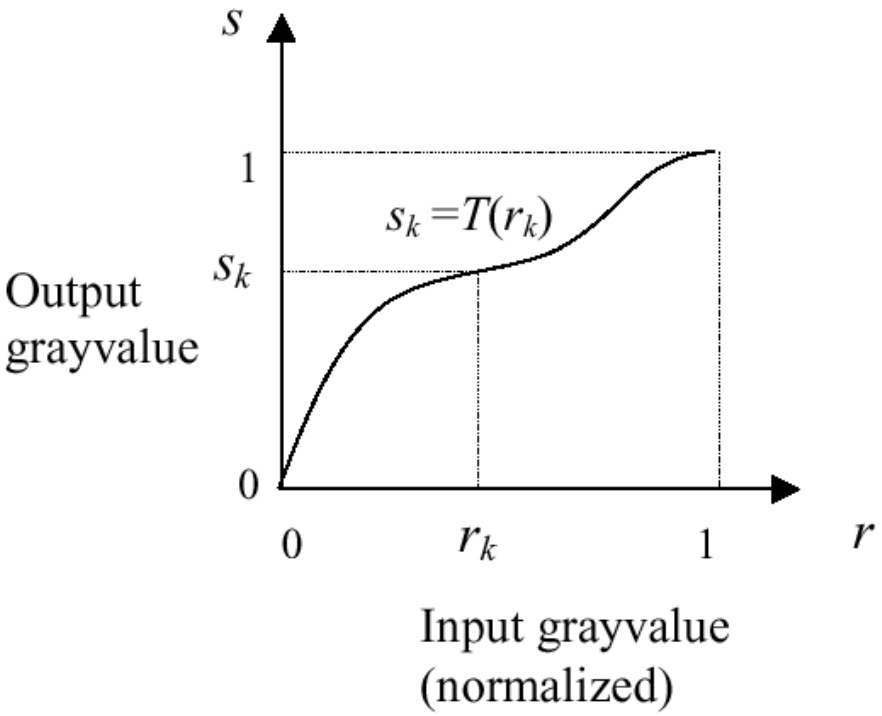
- r_k: input intensity
- s_k: processed intensity
- k: the intensity range (e.g 0.0 – 1.0)

processed intensity → $s_k = T(r_k)$ ← input intensity

Intensity range
(e.g 0 – 255)

Assumptions

- $T(r)$ is monotonically increasing over $[0, L-1]$.
- $T(r)$ maps $[0, L-1]$ into $[0, L-1]$, so as to preserve the range of allowed gray values.



Transformation Function for Histogram Equalization

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

- It can be seen that it satisfies the two conditions:
  - Monotonically increasing
  - The outputs are in the range [0, L-1].

- But our goal is to “equalize” the histogram. Is it achieved?

Probability Density Function

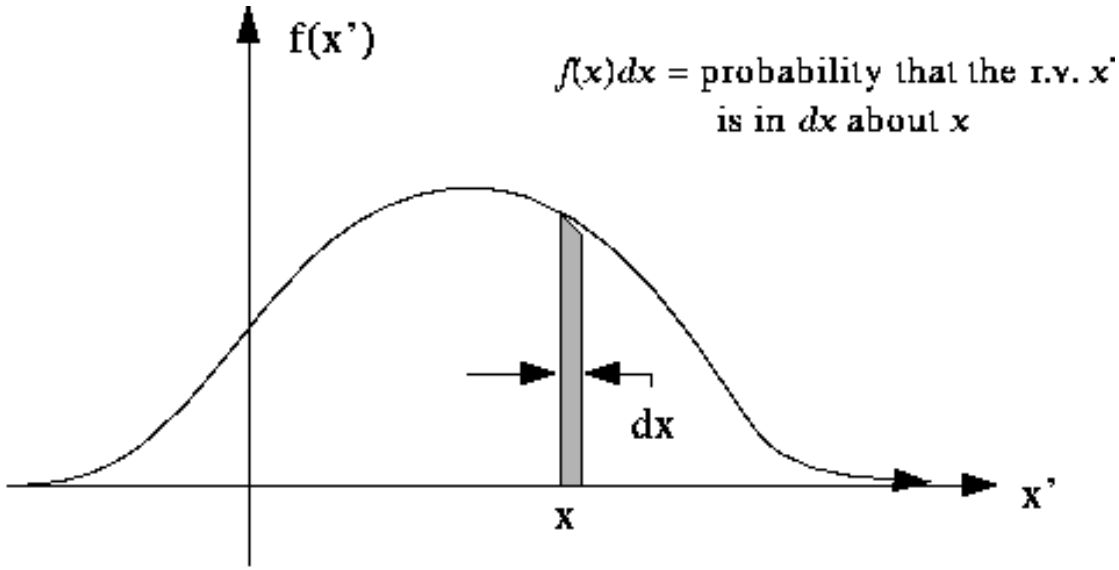


Figure 4. Typical Probability Distribution Function (*pdf*)

$$p(x) = \lim_{\Delta x \to 0} \frac{\Pr(x' \leq x + \Delta x) - \Pr(x' \leq x)}{\Delta x}$$

Transformed Density

This indicates that p_s is uniform

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\frac{1}{(L-1)p_r(r)} = \frac{1}{L-1}$$

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L-1)\frac{d}{dr}\left(\int_0^r p_r(w)dw\right) \\ &= (L-1)p_r(r)\end{aligned}$$

Discretize the Transform Rule

Continuous Version:

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

Discrete Version:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$$

Histogram Equalization: Example

Suppose that a 3-bit image (L=8) of size 64 × 64 pixels (MN = 4096) has the intensity distribution shown in following table.
Get the histogram equalization transformation function and give the $p_s(s_k)$ for each s_k.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Histogram Equalization: Example

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

$$s_k = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$$

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7 \times 0.19 = 1.33 \rightarrow 1$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7 \times (0.19 + 0.25) = 3.08 \rightarrow 3$$

$$s_2 = 4.55 \rightarrow 5 \qquad s_3 = 5.67 \rightarrow 6$$

$$s_4 = 6.23 \rightarrow 6 \qquad s_5 = 6.65 \rightarrow 7$$

$$s_6 = 6.86 \rightarrow 7 \qquad s_7 = 7.00 \rightarrow 7$$

Histogram Equalization: Example

final transform:

$r_0 \rightarrow s_0 = 1 \Rightarrow 790$ pixels map to 1

$r_1 \rightarrow s_1 = 3 \Rightarrow 1023$ pixels map to 3

$r_2 \rightarrow s_2 = 5 \Rightarrow 850$ pixels map to 5

$r_3 \rightarrow s_3 = 6 \Rightarrow 656 + 329 = 985$ pixels map to 6

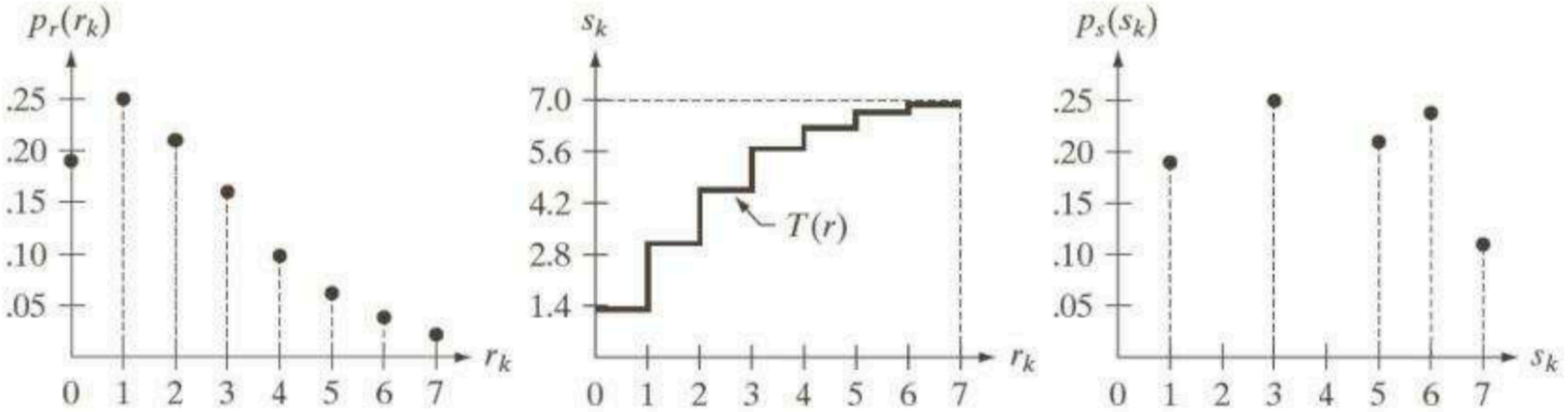
$r_4 \rightarrow s_4 = 6 \Rightarrow 656 + 329 = 985$ pixels map to 6

$r_5 \rightarrow s_5 = 7 \Rightarrow 245 + 122 + 81 = 458$ pixels map to 7

$r_6 \rightarrow s_6 = 7 \Rightarrow 245 + 122 + 81 = 458$ pixels map to 7

$r_7 \rightarrow s_7 = 7 \Rightarrow 245 + 122 + 81 = 458$ pixels map to 7

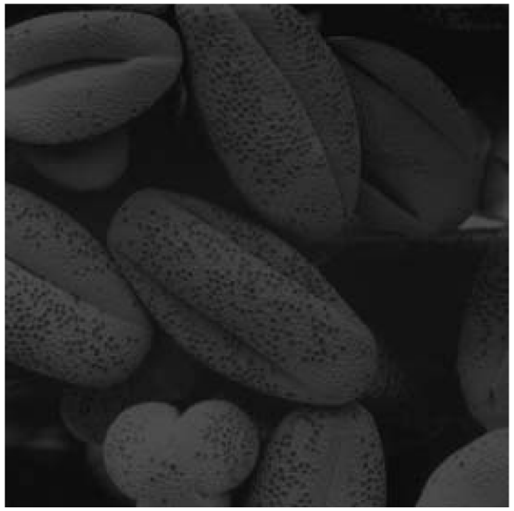
Histogram Equalization: Example

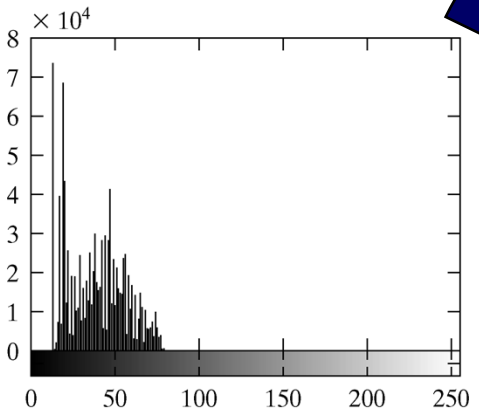


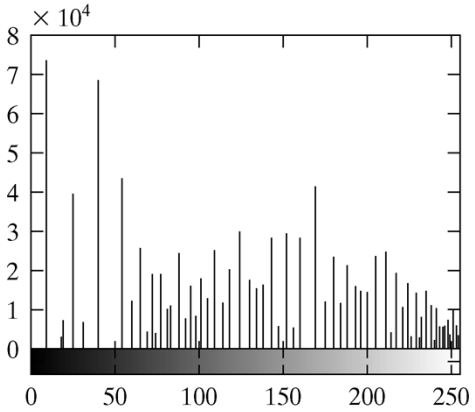
a b c

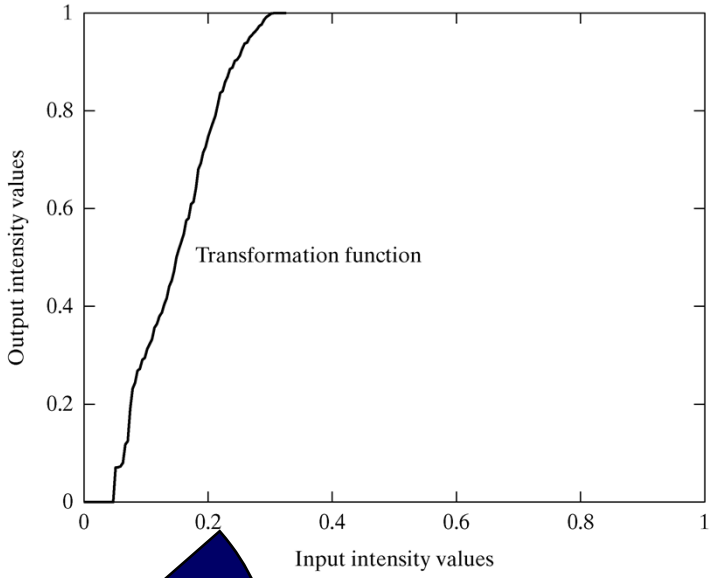
FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Equalization Transformation Function





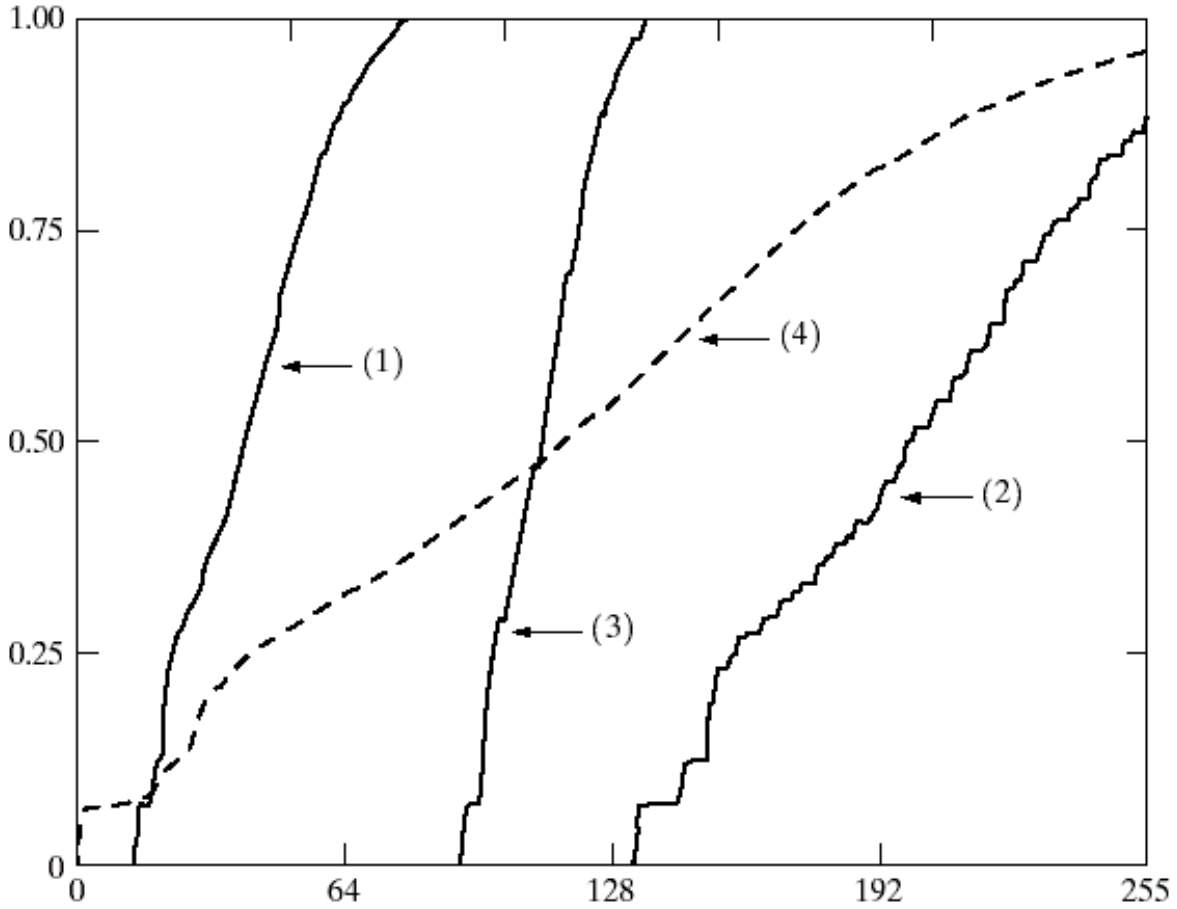




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

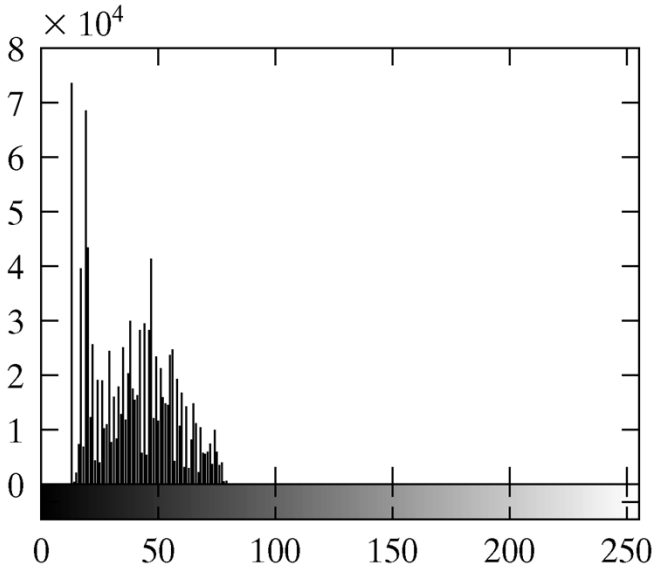
Equalization Transformation Functions

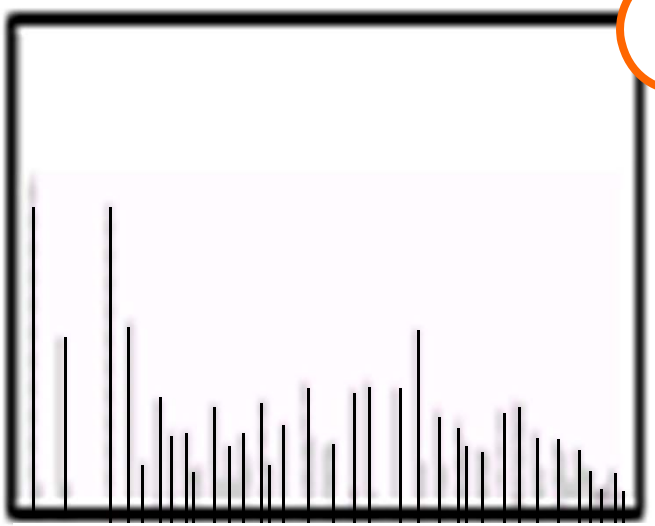
Different equalization function (1-4) may be used

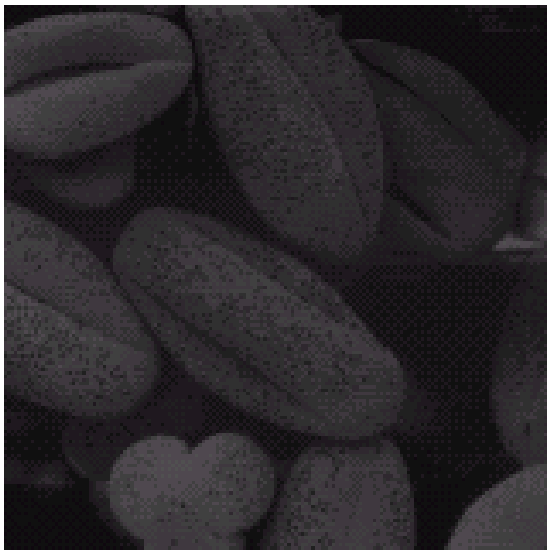


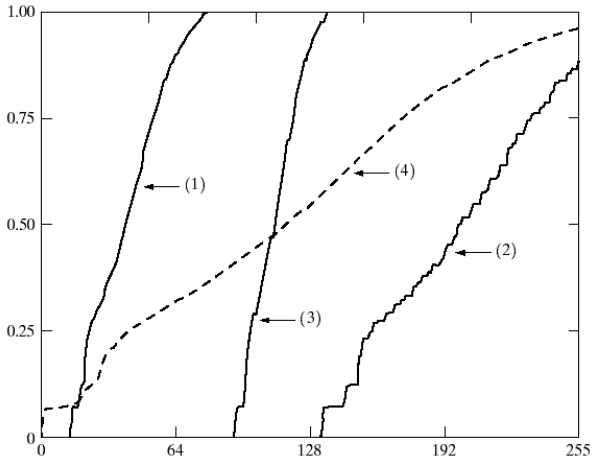
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Equalization Examples



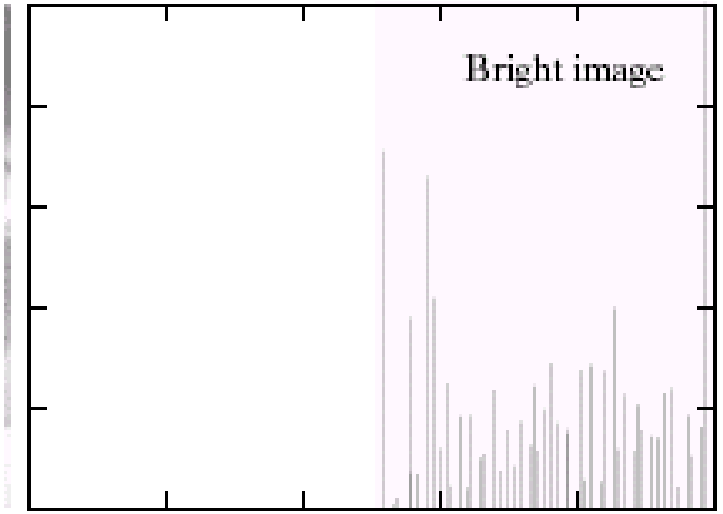


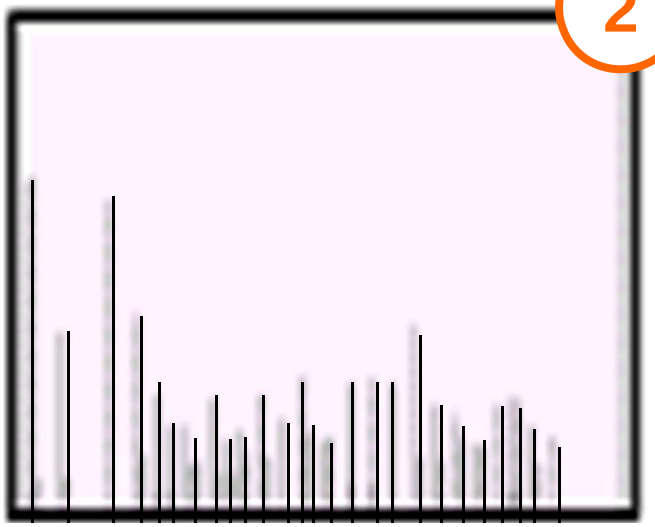


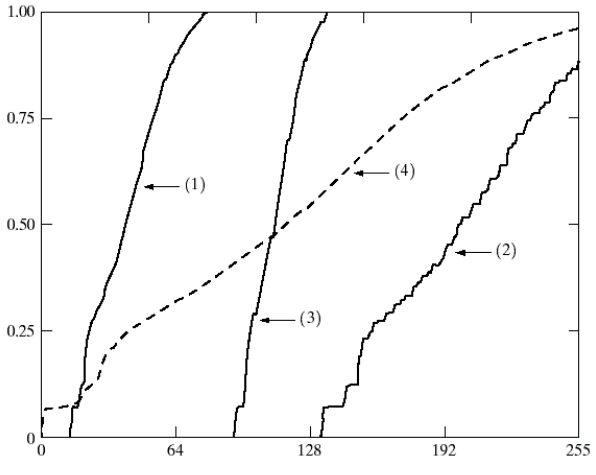


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Equalization Examples

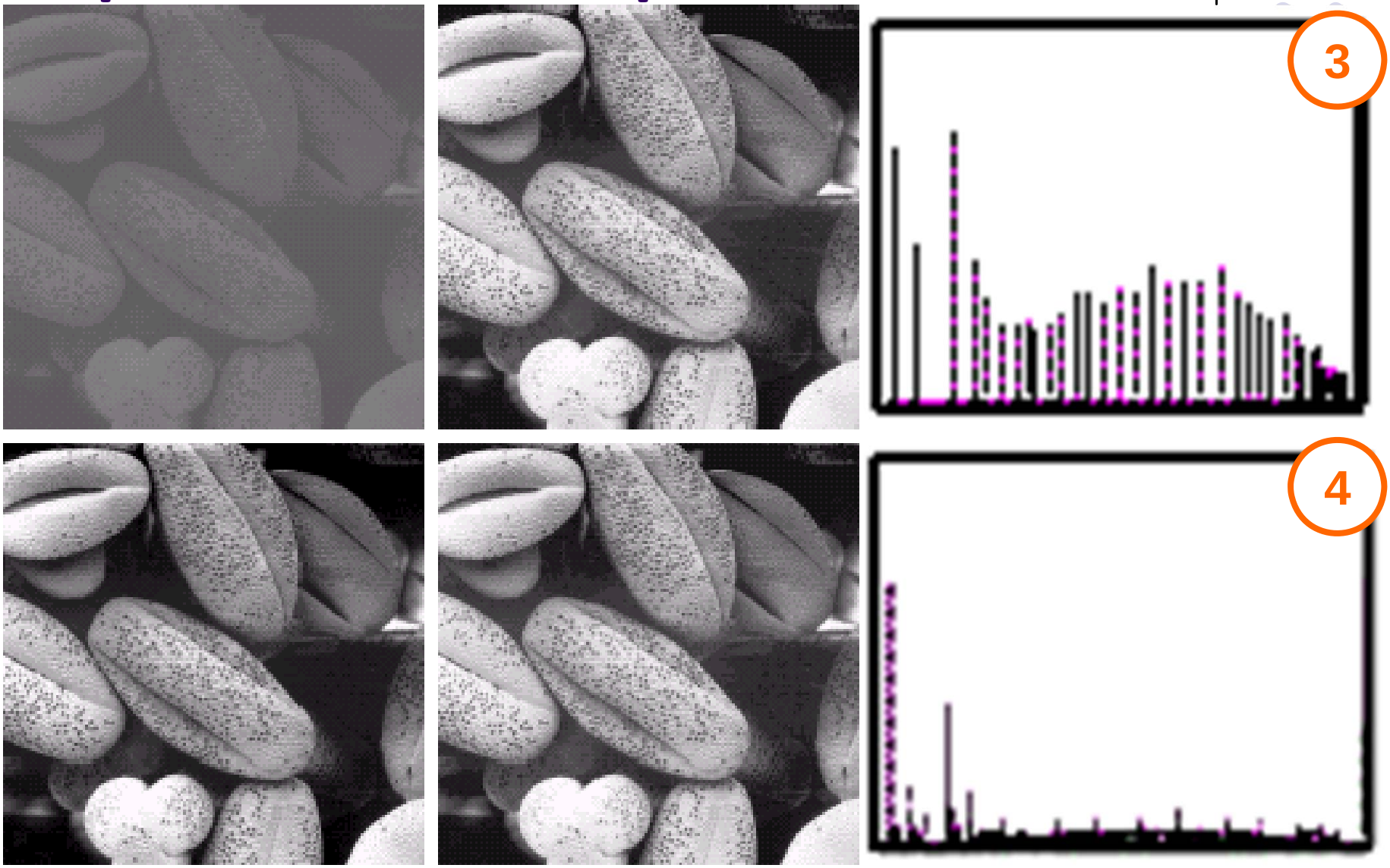






Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Equalization Examples



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Histogram Equalization Algorithm

- Given an image, compute the histogram.
- Normalize the histogram to obtain pixel-wise probabilities
- Compute the transformation table.
- For each pixel, replace its value with a new value from the transformation table.

Histogram Equalization: Summary

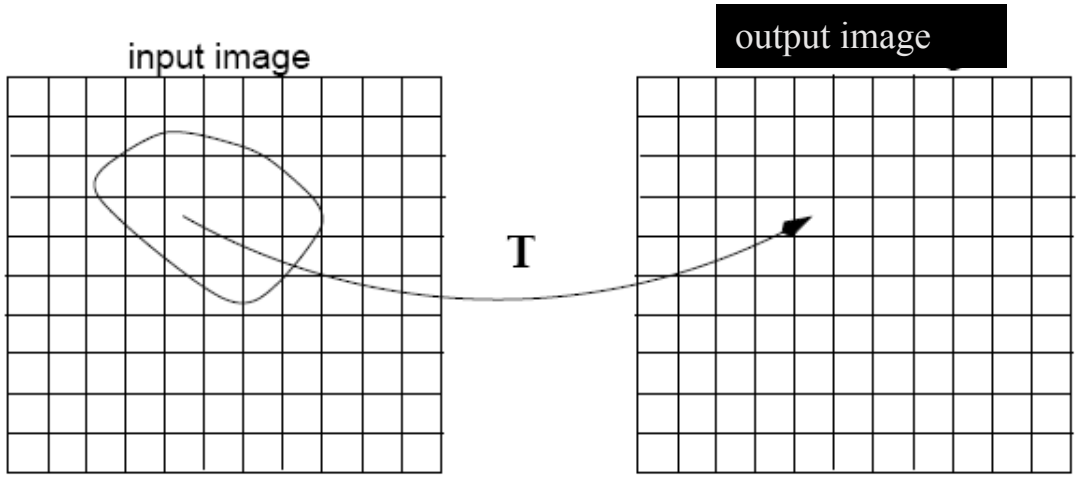
- Histogram equalization yields an **approximately** uniform distribution in the output.

- The equalized image spreads a wider range of the intensity scale and enhances the contrast and clarity of details.

- The **algorithm** is automatic and can be easily implemented by a computer program.

Spatial Filtering

Spatial Filtering



$$g(x,y) = T[f(x,y)]$$

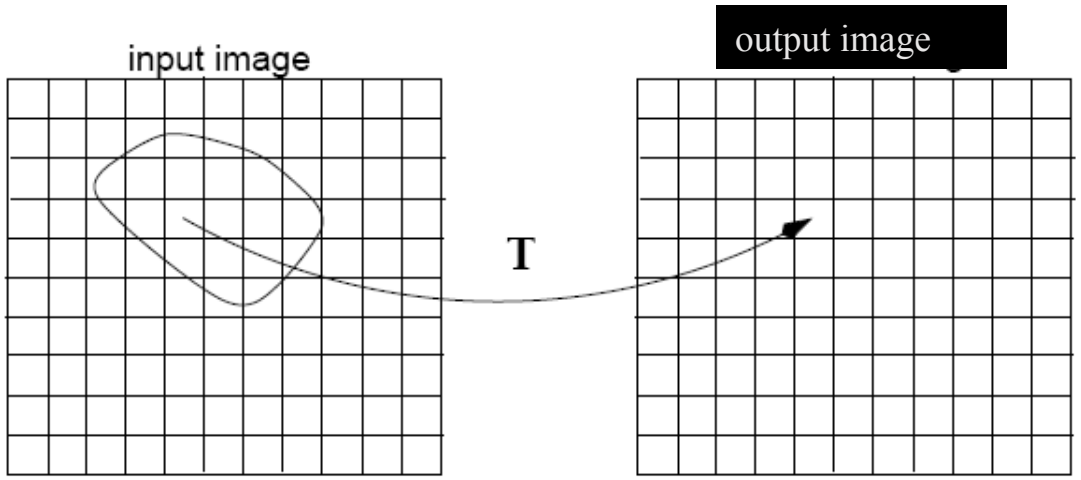
T operates on a neighborhood of pixels

Spatial Filtering (cont'd)

- The word "filtering" has been borrowed from the frequency domain.
- Filters are classified as:
  - Low-pass (i.e., preserve low frequencies)
  - High-pass (i.e., preserve high frequencies)
  - Band-pass (i.e., preserve frequencies within a band)
  - Band-reject (i.e., reject frequencies within a band)

Spatial Filtering (cont'd)

- Need to define:
  (1) a neighborhood (or mask)
  (2) an operation

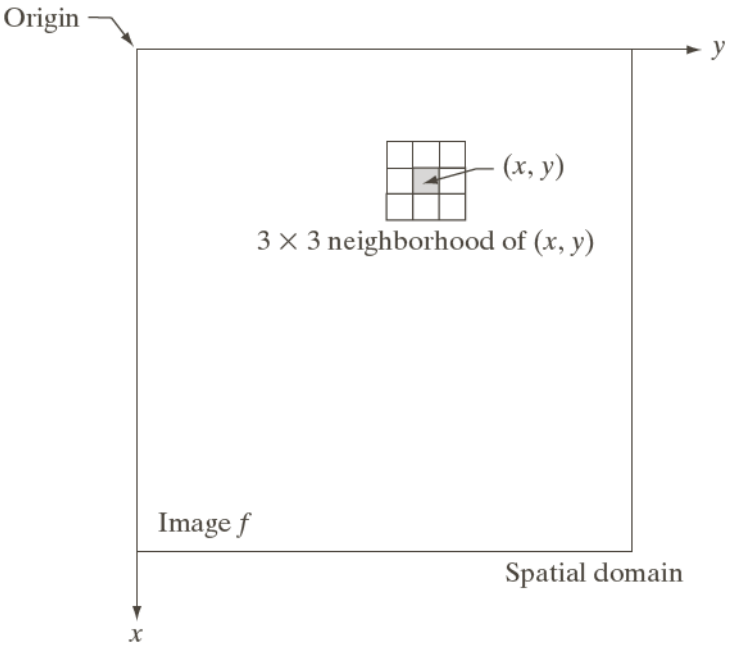


$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

Spatial Filtering – Neighborhood

- Typically, the neighborhood is rectangular and its size is much smaller than that of *f(x,y)*
  - e.g., 3x3 or 5x5



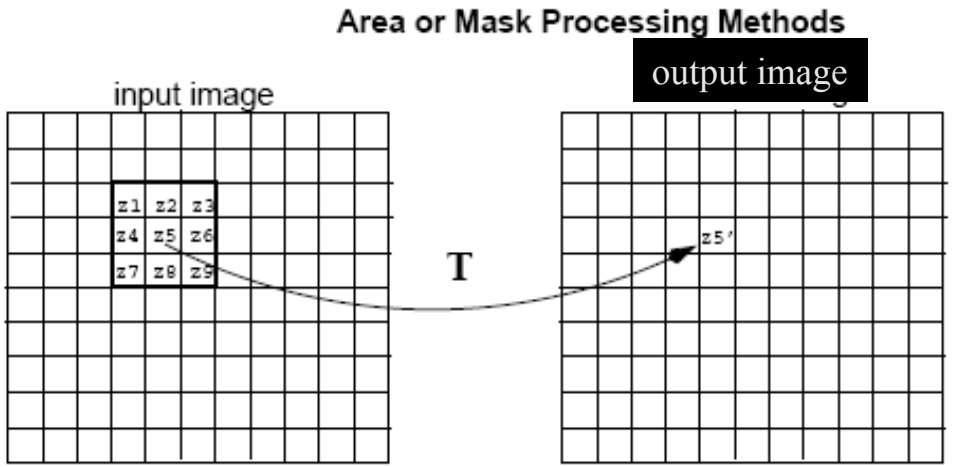
Spatial filtering - Operation

Example: weighted sum of input pixels.

$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

mask weights:

w1	w2	w3
w4	w5	w6
w7	w8	w9



$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

A filtered image is generated as the center of the mask moves to every pixel in the input image.

Handling Pixels Close to Boundaries

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	2	5	6	3	6	7	3	0	0
0	0	2	3	4	6	7	5	1	8	4	0	0
0	0	8	7	6	5	7	6	3	3	4	0	0
0	0	2	3	5	6	7	8	2	7	3	0	0
0	0	4	5	3	2	1	6	8	7	2	0	0
0	0	1	4	5	3	2	6	7	8	1	0	0
0	0	2	3	4	5	6	8	9	2	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Zero Padding(option1)

2	2	2	3	4	6	7	5	1	8	4	4	4
1	1	1	1	2	5	6	3	6	7	3	3	3
1	1	1	1	2	5	6	3	6	7	3	3	7
3	2	2	3	4	6	7	5	1	8	4	4	8
7	8	8	7	6	5	7	6	3	3	4	4	3
3	2	2	3	5	6	7	8	2	7	3	3	7
5	4	4	5	3	2	1	6	8	7	2	2	7
4	1	1	4	5	3	2	6	7	8	1	1	8
3	2	2	3	4	5	6	8	9	2	1	1	2
3	2	2	3	4	5	6	8	9	2	1	1	2
3	1	1	4	5	3	2	6	7	8	1	1	2

Repetition of border pixels(option 2

Linear Function

$$r = T(x)$$

T is linear if it satisfies:

$$T(\alpha x + \beta y) = \alpha T(x) + \beta T(y)$$

Are the following functions linear?

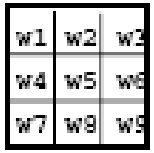
$$T(x) = 2x_1 + 3x_2$$

$$T(x) = 2x_1 + 3x_2 + 1$$

$$T(x) = 2x_1^2$$

Linear vs Non-Linear

- A filtering method is linear when the output is a weighted sum of the input pixels. (Why?)



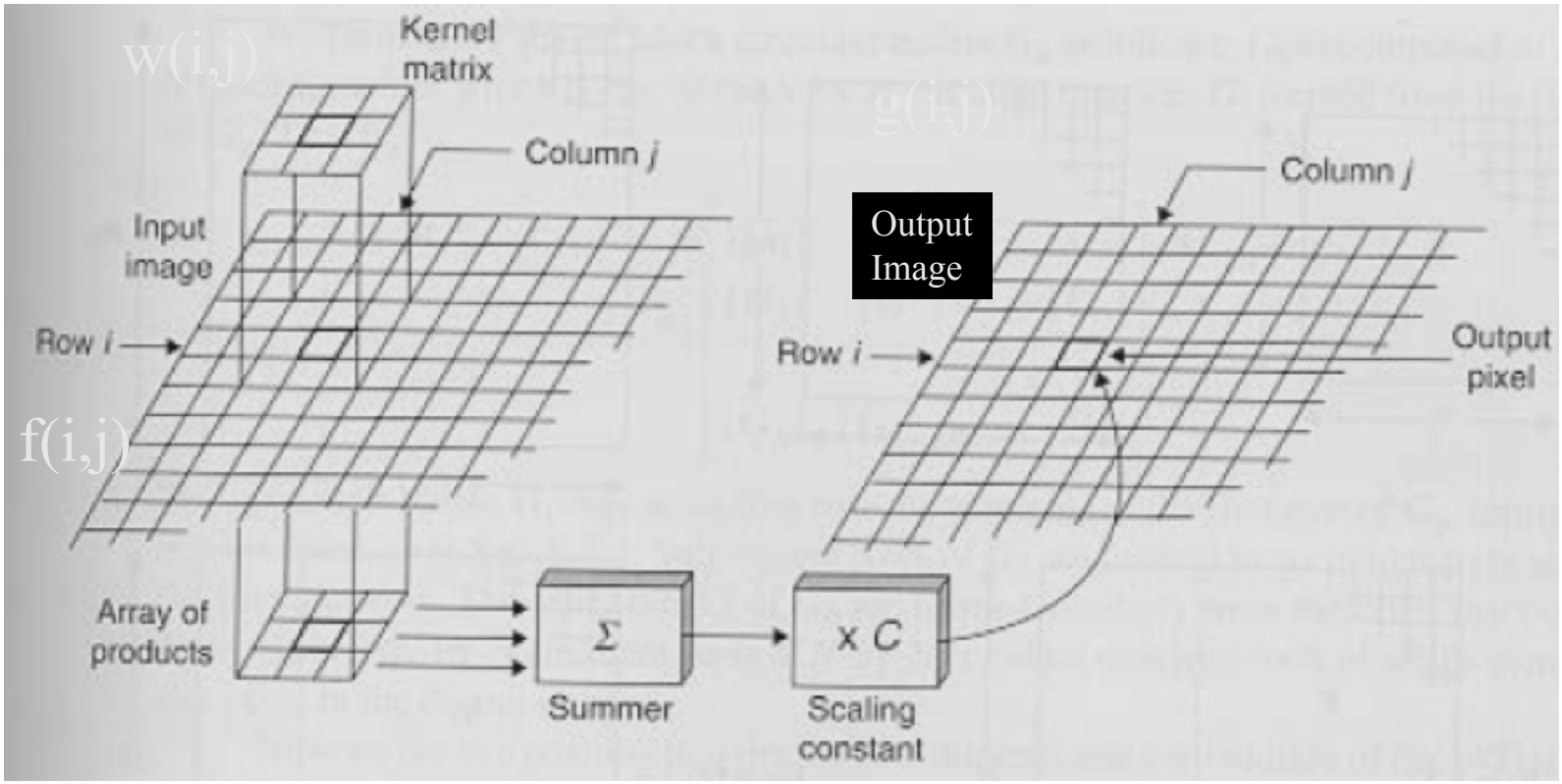
$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

- Methods that do not satisfy the above property are called non-linear.
  - e.g., $z_5' = max(z_k, k = 1, 2, \ldots, 9)$

Linear Spatial Filtering Methods

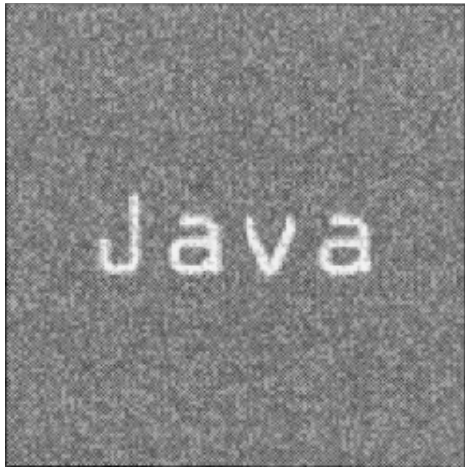
- Main linear spatial filtering methods:
  - Correlation
  - Convolution

Correlation



$$g(i,j) = w(i,j) \bullet f(i,j) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(i+s, j+t)$$

Correlation (cont'd)



Often used in applications where we need to measure the similarity between images or parts of images (e.g., template matching).

Convolution

- Similar to correlation except that the mask is first **flipped** both horizontally and vertically.

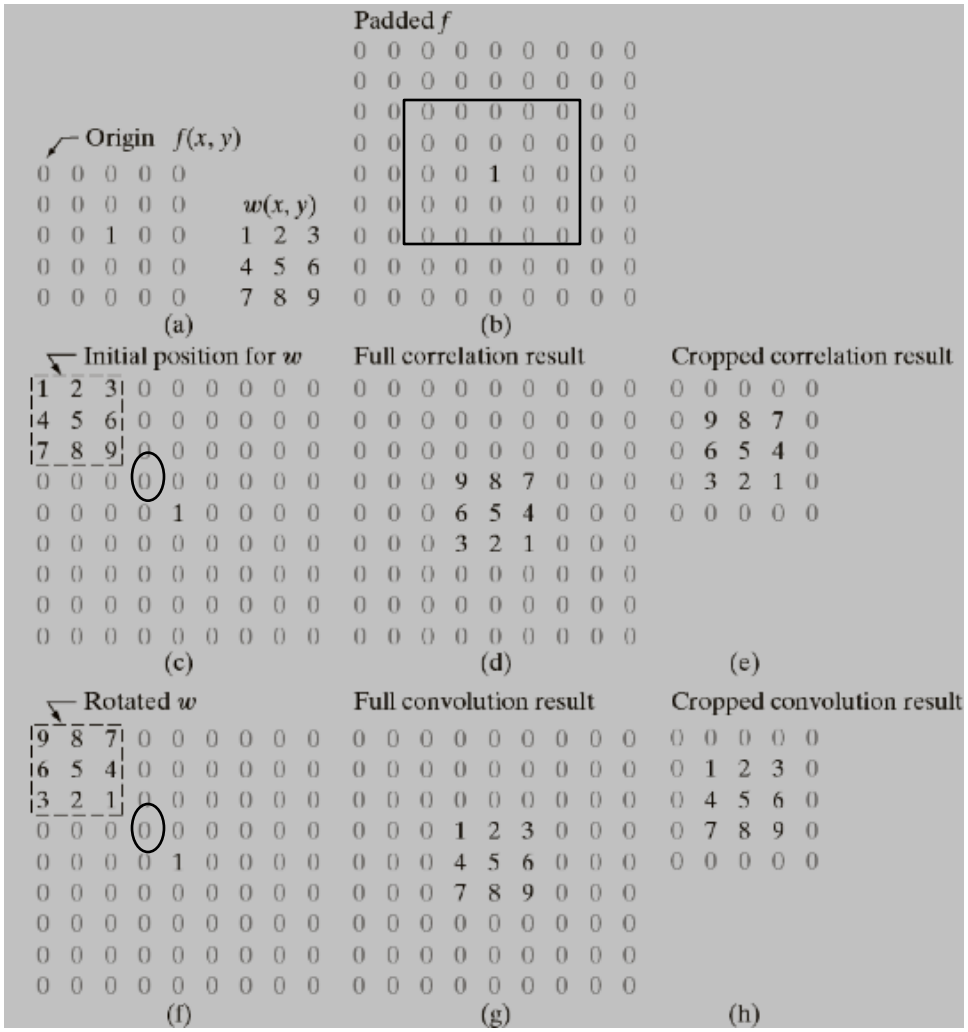
$$g(i,j) = w(i,j) * f(i,j) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(i-s, j-t)$$

Note: if w(i, j) is symmetric, that is w(i, j)=w(-i,-j), then convolution is equivalent to correlation!

Example

Correlation:

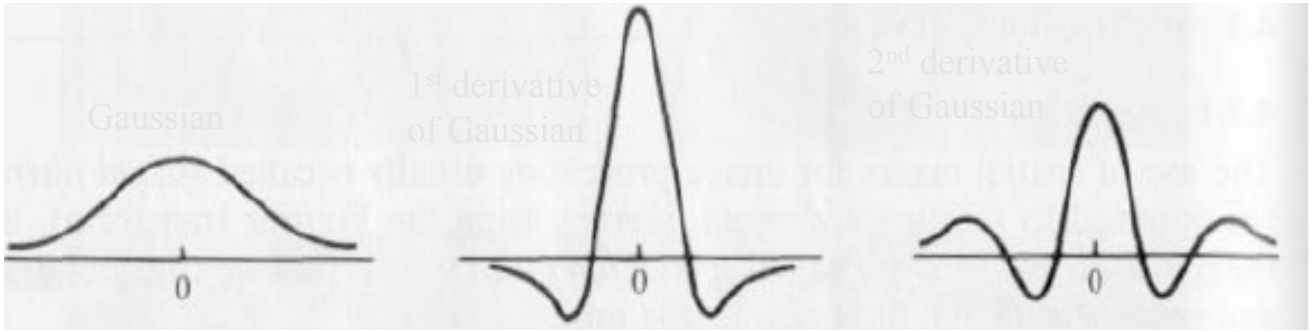
Convolution:



How do we choose the mask weights?

- Typically, by sampling certain functions:

w1	w2	w3
w4	w5	w6
w7	w8	w9

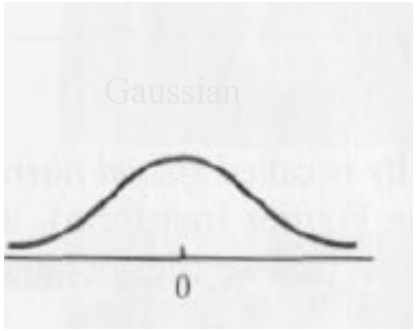


Filters

- We will mainly focus on two types of filters:
  - Smoothing (low-pass)
  - Sharpening (high-pass)

Smoothing Filters (low-pass)

- Useful for reducing noise and eliminating small details.
- The elements of the mask are non-negative.
- Sum of mask elements is 1 (after normalization).



7×7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Smoothing filters – Example

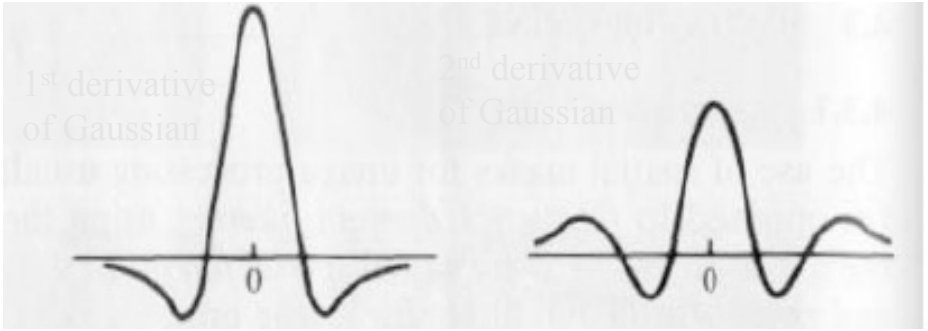
- Useful for reducing noise and eliminating small details.

input image

smoothed image

Sharpening Filters (high-pass)

- Useful for highlighting fine details.
- The elements of the mask contain both positive and negative weights.
- Sum of mask elements is 0.

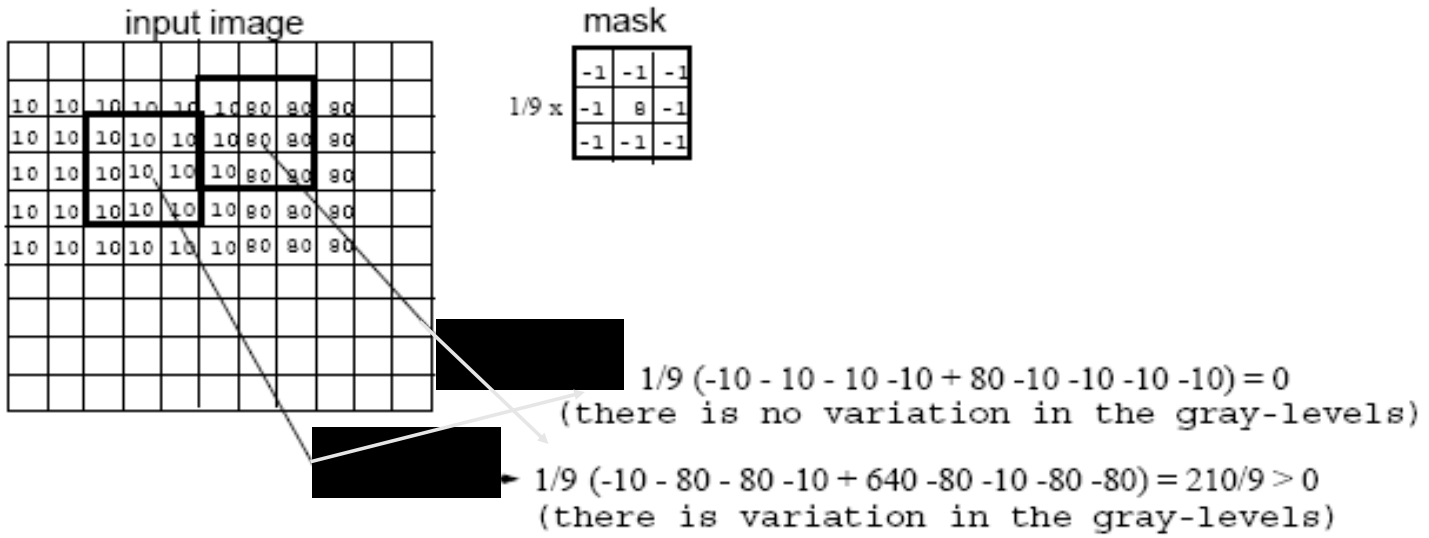


mask

-1	-1	-1
-1	8	-1
-1	-1	-1

Sharpening Filters (cont'd)

- Useful for highlighting fine details.
  - e.g., emphasize edges



Sharpening Filters- Example

- Note that the response of sharpening might be negative.
- Values must be re-mapped to [0, 255]

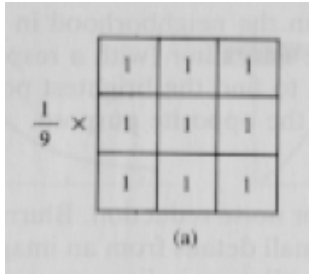
input image

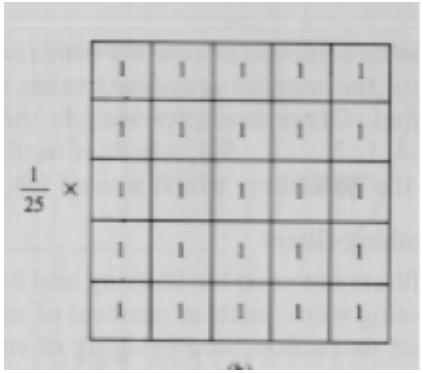
sharpened image

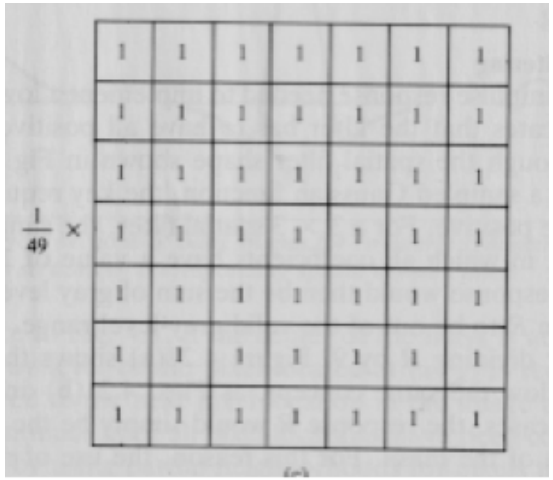
Smoothing Filters

- Averaging
- Gaussian
- Median filtering (non-linear)

Smoothing Filters: Averaging



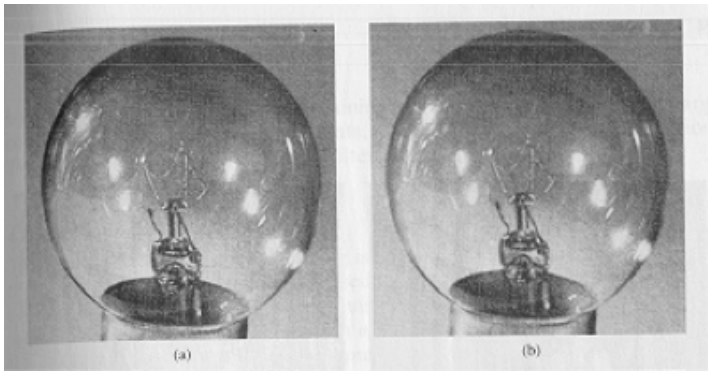




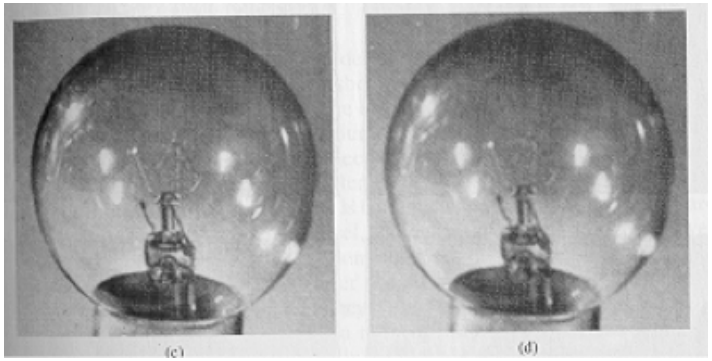
Smoothing Filters: Averaging (cont'd)

- Mask size determines the degree of smoothing (loss of detail).

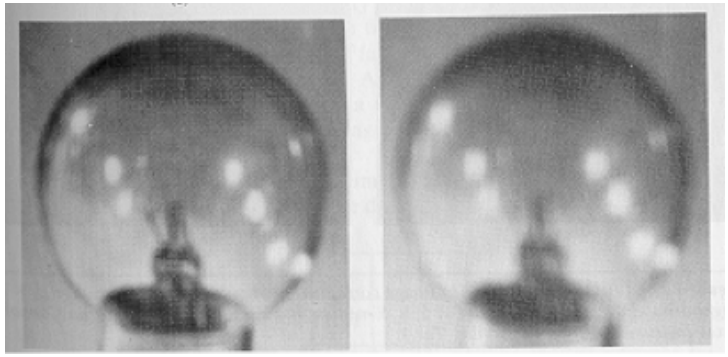
original 3x3



5x5 7x7



15x15 25x25

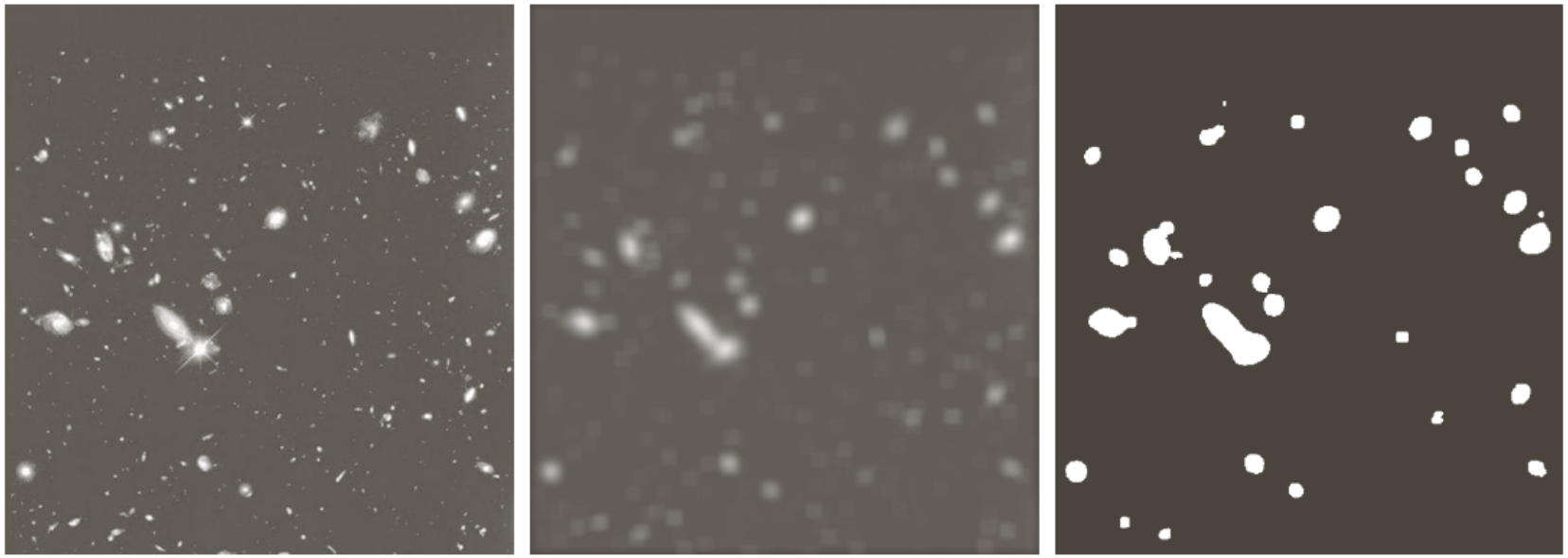


Smoothing Filters: Averaging (cont'd)

Example: extract largest, brightest objects

15 x 15 averaging

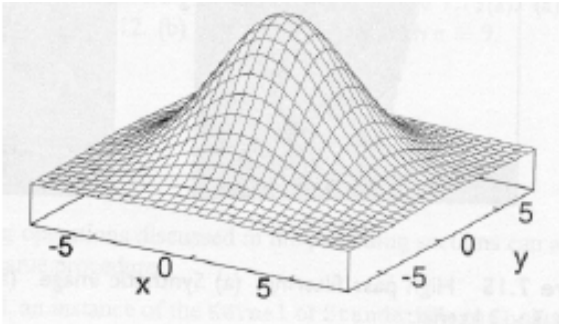
image thresholding



Smoothing filters: Gaussian

- The weights are Gaussian samples:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$



7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

$\sigma = 1.4$

mask size is a function of **σ**:

$$height = width = 5\sigma \text{ (subtends 98.76\% of the area)}$$

Smoothing filters: Gaussian (cont'd)

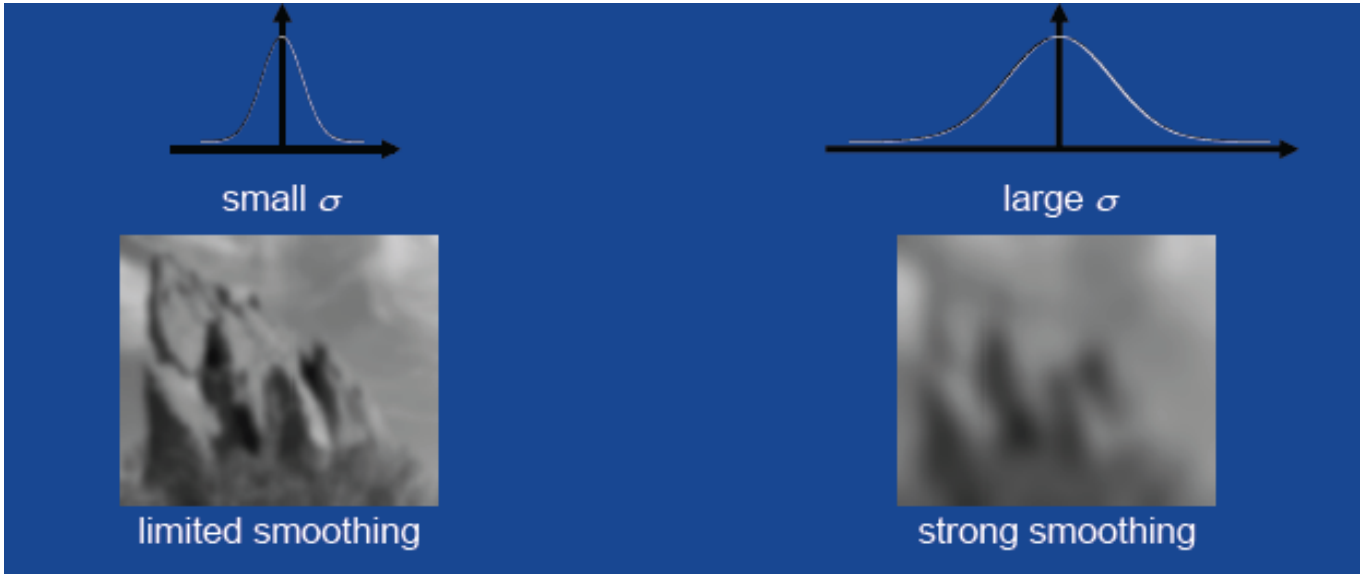
- **σ** controls the amount of smoothing
- As **σ** increases, more samples must be obtained to represent the Gaussian function accurately.

$\sigma = 3$

15 × 15 Gaussian mask

2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

Smoothing filters: Gaussian (cont'd)



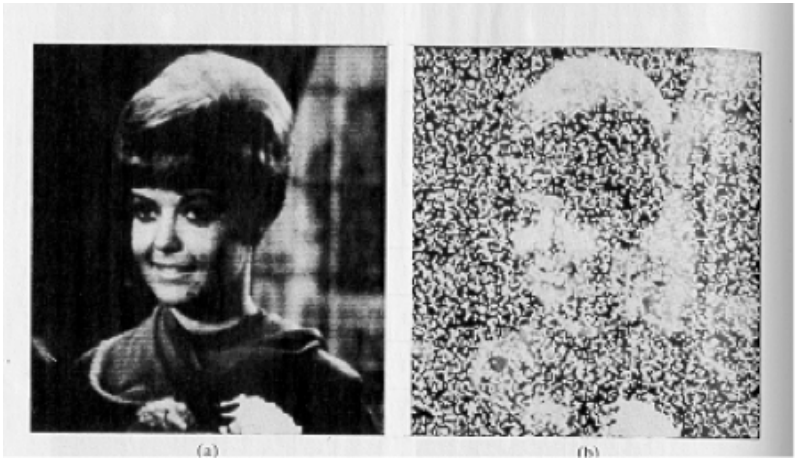
Averaging vs Gaussian Smoothing

Averaging

Gaussian

Smoothing Filters: Median Filtering (non-linear)

- Very effective for removing “salt and pepper” noise (i.e., random occurrences of black and white pixels).

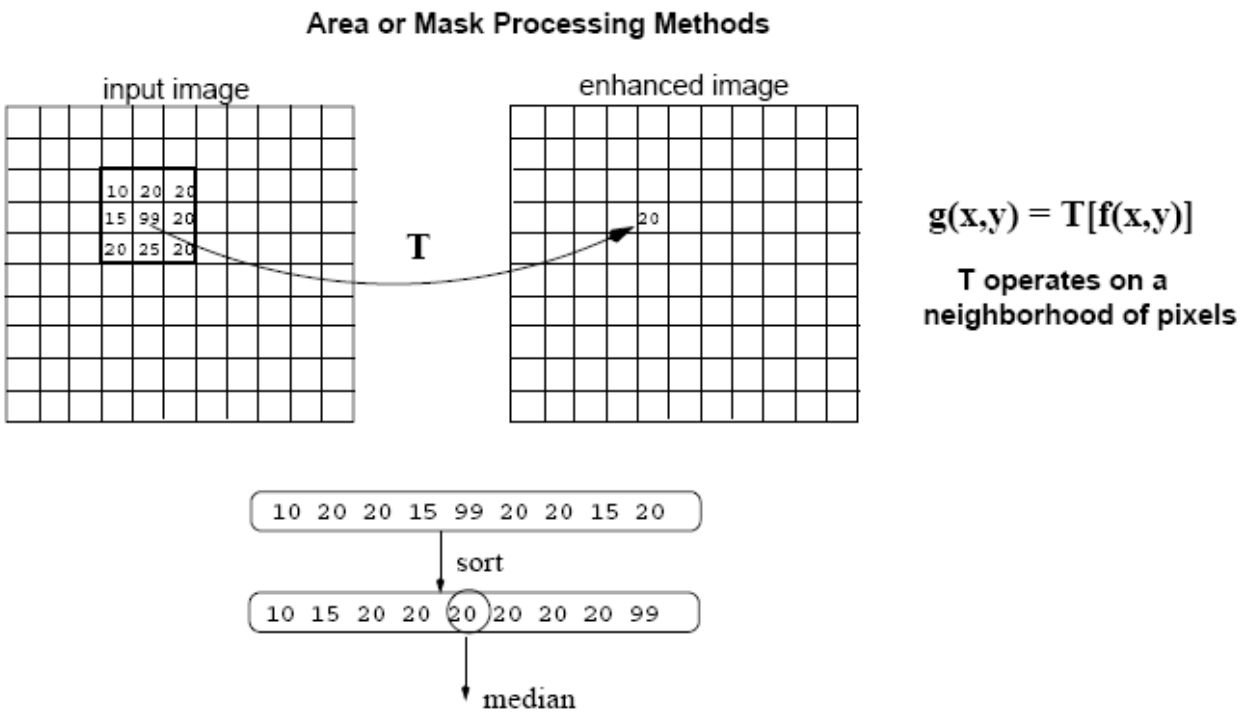


averaging

median filtering

Smoothing Filters: Median Filtering (cont'd)

- Replace each pixel by the median in a neighborhood around the pixel.



Sharpening Filters

- Unsharp masking
- High Boost filter
- Gradient (1^{st} derivative)
- Laplacian (2^{nd} derivative)

Sharpening Filters: Unsharp Masking

- Obtain a sharp image by subtracting a lowpass filtered (i.e., smoothed) image from the original image:

$$Highpass = Original - Lowpass$$

(after contrast enhancement)

Sharpening Filters: High Boost

- Image sharpening emphasizes edges but low frequency components are lost.
- **High boost filter**: amplify input image, then subtract a lowpass image.

$$Highboost = A\ Original - Lowpass$$

$$= (A-1)\ Original + Original - Lowpass$$

$$= (A-1)\ Original + Highpass$$

(A-1)

Sharpening Filters: High Boost (cont'd)

- If **A=1**, we get unsharp masking.
- If **A>1**, part of the original image is added back to the high pass filtered image.
- One way to implement high boost filtering is using the masks below:

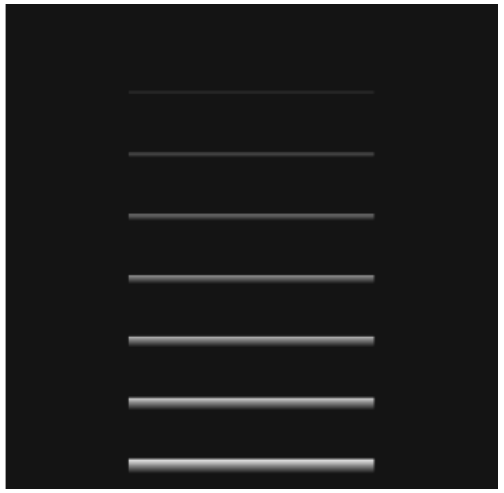
A>=1
w = 9A-1

-1	-1	-1
-1	w	-1
-1	-1	-1

A=2
w = 17

-1	-1	-1
-1	17	-1
-1	-1	-1

Sharpening Filters: High Boost (cont'd)



A=1.4

A=1.9

Sharpening Filters: Derivatives

- Taking the derivative of an image results in sharpening the image.
- The derivative of an image (i.e., 2D signal) can be computed using the gradient.

$$\nabla f \qquad grad(f) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Gradient

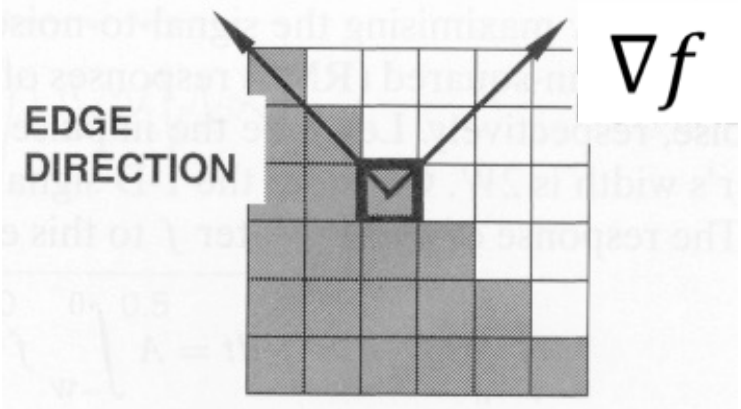
- The gradient is a **vector** which has magnitude and direction:

$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

$$direction(grad(f)) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Gradient (cont'd)

- **Gradient magnitude:** provides information about edge strength.
- **Gradient direction:** perpendicular to the direction of the edge.



Gradient Computation

- Approximate gradient using finite differences:

$$f'(x) = \lim_{h->0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

$$\frac{\partial f}{\partial x} = \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} = f(x+1, y) - f(x,y), \quad (\Delta x=1)$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+\Delta y) - f(x,y)}{-\Delta y} = f(x,y) - f(x, y+1), \quad (\Delta y=1)$$

Gradient Computation (cont'd)

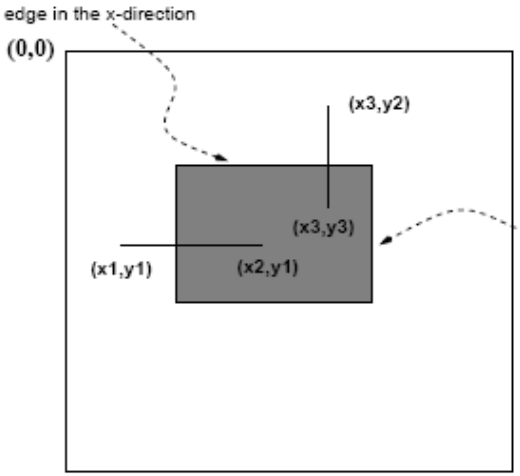
$$\frac{f(x3,y2) - f(x3,y3)}{y2 - y3}$$

sensitive to horizontal edges

$$\frac{\partial f}{\partial y}$$

or $\dfrac{f(x,y+Dy) - f(x,y)}{-Dy} = \boxed{f(x,y) - f(x,y+1)}$ **(grad in the y-direction)**

(y3=y2+Dy, y2=y, x3=x, Dy=1)



sensitive to vertical edges

$$\frac{\partial f}{\partial x}$$

$$\frac{f(x2,y1) - f(x1,y1)}{x2 - x1}$$

or $\dfrac{f(x+Dx,y) - f(x,y)}{Dx} =$

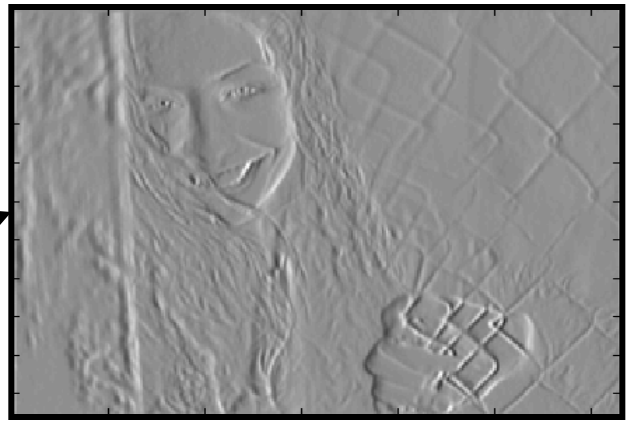
$\boxed{f(x+1,y) - f(x,y)}$

(grad in the x-direction)

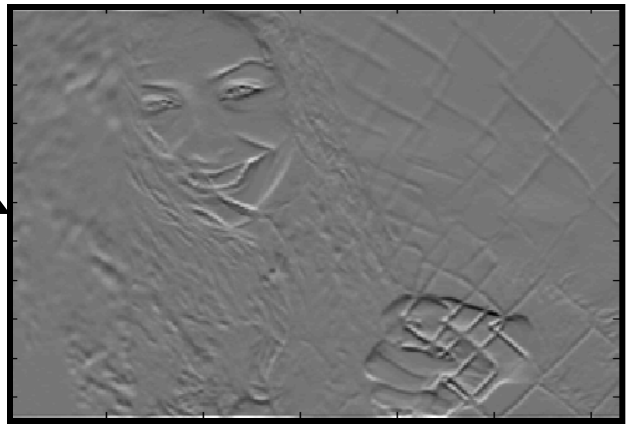
(x2=x+Dx, x1=x, y1=y2=y, Dx=1)

Example: visualize partial derivatives

$$\frac{\partial}{\partial x}$$



$$\frac{\partial}{\partial y}$$



Implement Gradient Using Masks

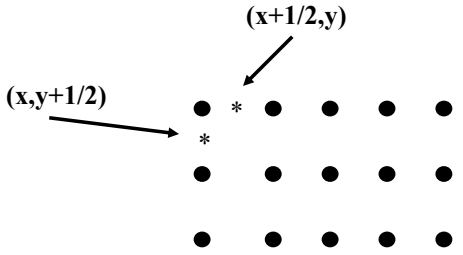
- We can implemen $\frac{\partial f}{\partial x}$ ar $\frac{\partial f}{\partial y}$ using masks:

-1	1

1
-1

$\frac{\partial f}{\partial x}$ good approximation at (x+1/2,y)

$\frac{\partial f}{\partial y}$ good approximation at (x,y+1/2)



- Example: approximate gradient at z_5

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

$$\frac{\partial f}{\partial x} = z_6 - z_5$$

$$\frac{\partial f}{\partial y} = z_5 - z_8$$

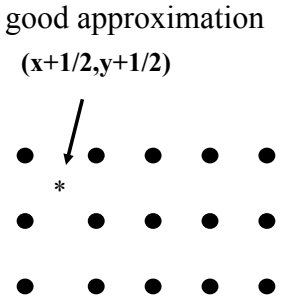
$$mag(grad(f)) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

Implement Gradient Using Masks (cont'd)

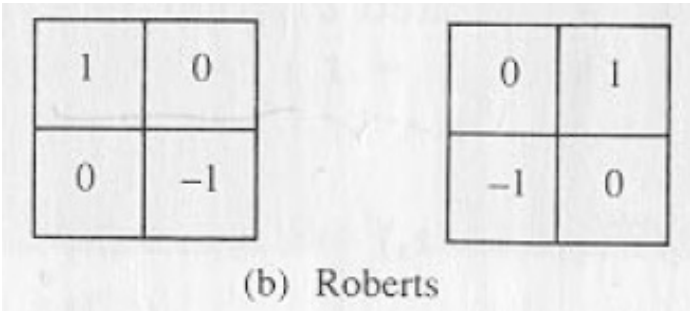
- A different approximation of the gradient:

$$\frac{\partial f}{\partial x}(x,y) = f(x,y) - f(x+1,y+1)$$

$$\frac{\partial f}{\partial y}(x,y) = f(x+1,y) - f(x,y+1),$$

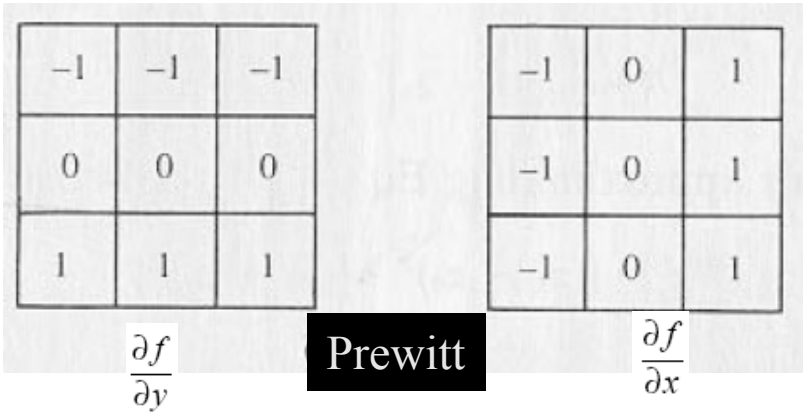


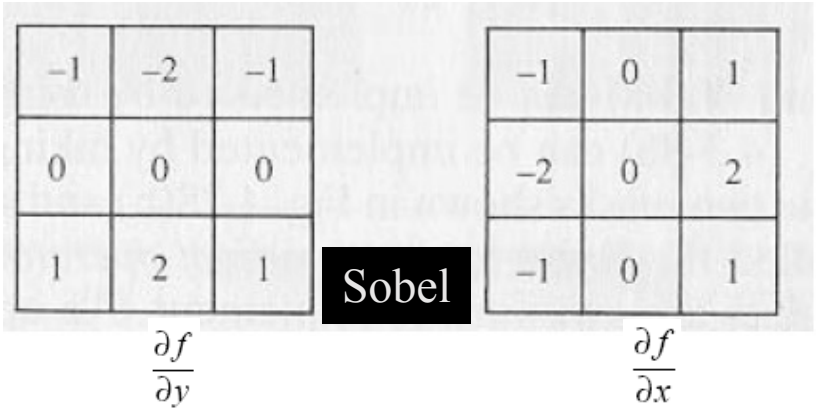
- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:



Implement Gradient Using Masks (cont'd)

- Other approximations

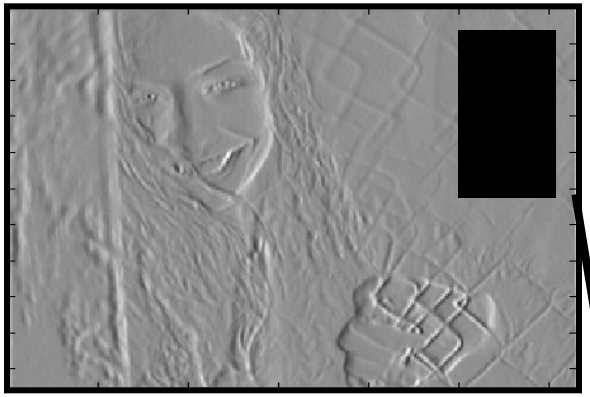


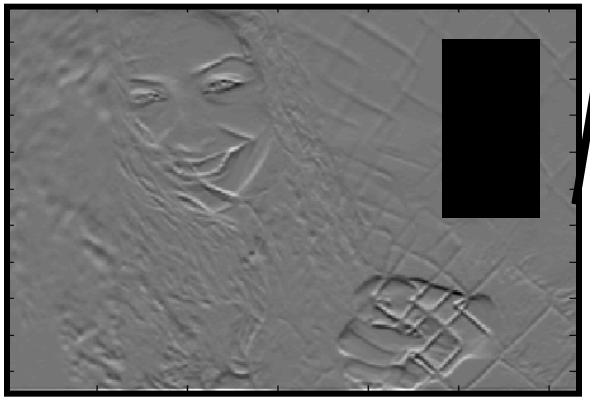


Example: Gradient Magnitude Image

Gradient Magnitude

$$: \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$





(isotropic)

Laplacian

The Laplacian (2nd derivative) is defined as:

$$\nabla^2 = \nabla \cdot \nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

(dot product)

Approximate 2nd derivatives:

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

$$\nabla^2 f = -4f(i, j) + f(i, j+1) + f(i, j-1) + f(i+1, j) + f(i-1, j)$$

Laplacian (cont'd)

Laplacian Mask

0	0	0
1	-2	1
0	0	0

+

0	1	0
0	-2	0
0	1	0

=

0	1	0
1	-4	1
0	1	0

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

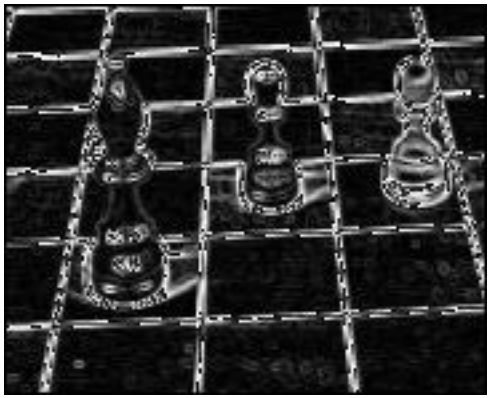
Edges can be found
by detect the zero-crossings

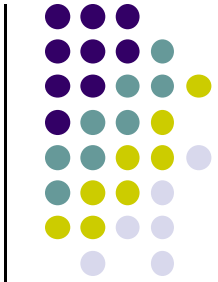
-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

Example: Laplacian vs Gradient

Laplacian

Sobel





References

- Wilhelm Burger and Mark J. Burge, Digital Image Processing, Springer, 2008
  - Histograms (Ch 4)
  - Point operations (Ch 5)
- University of Utah, CS 4640: Image Processing Basics, Spring 2012
- Rutgers University, CS 334, Introduction to Imaging and Multimedia, Fall 2012
- Gonzales and Woods, Digital Image Processing (3rd edition), Prentice Hall