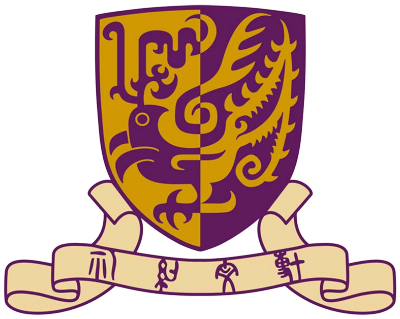
EIE4512 - Digital Image Processing
Histogram Operations and Spatial Filtering

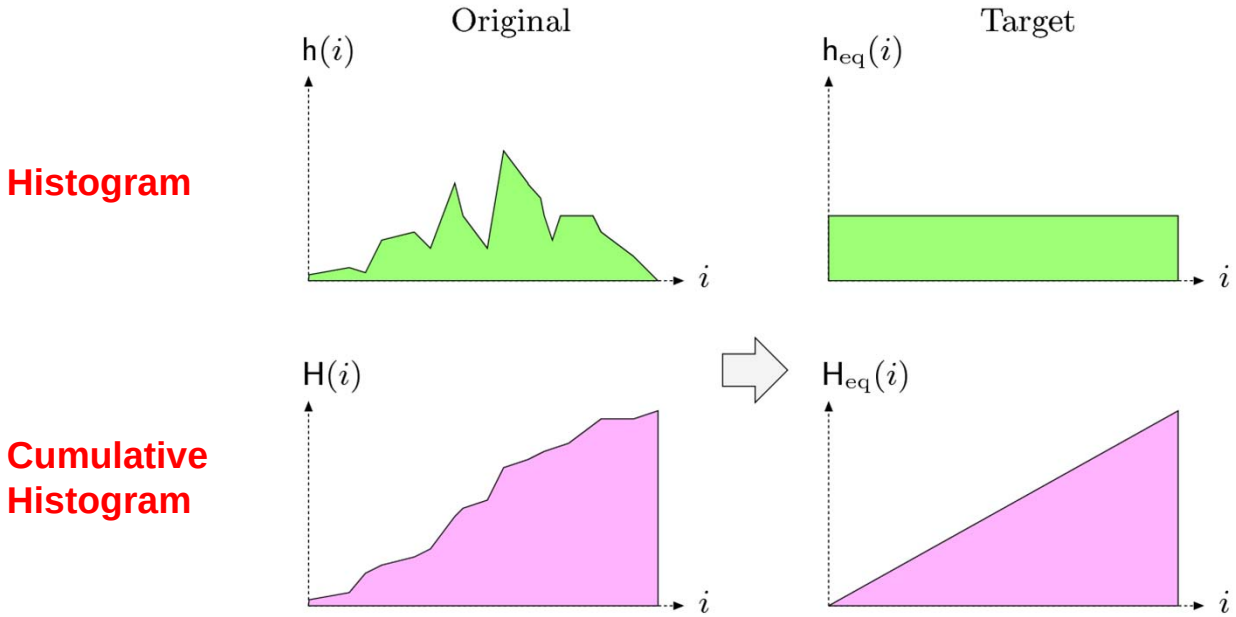


香 港 中 文 大 學 (深 圳)
The Chinese University of Hong Kong, Shenzhen

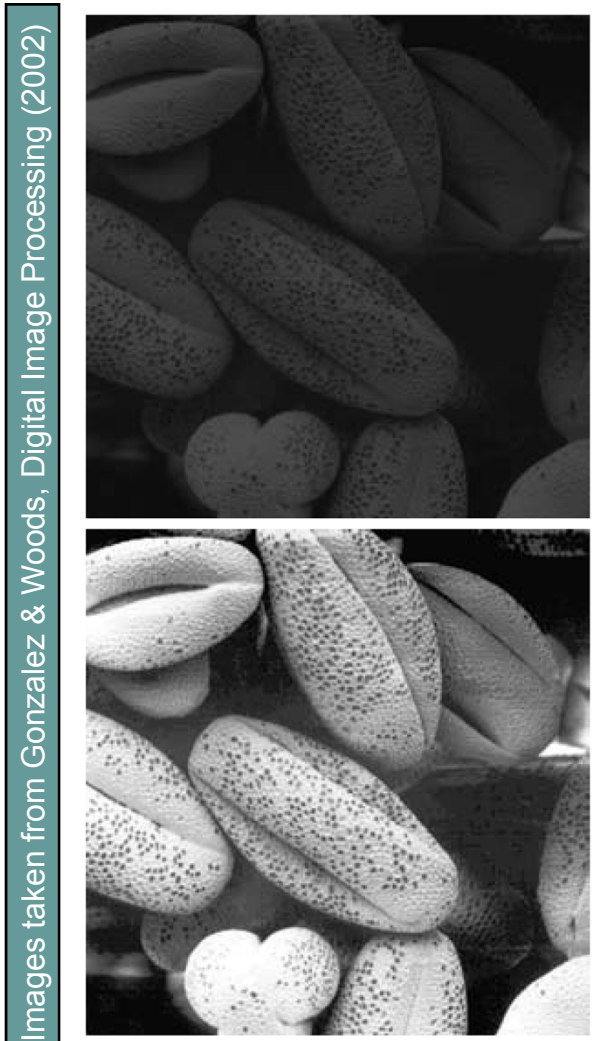
Zhen Li
lizhen@cuhk.edu.cn
School of Science and Engineering
The Chinese University of Hong Kong, Shen Zhen
Januray 22 - 24, 2019

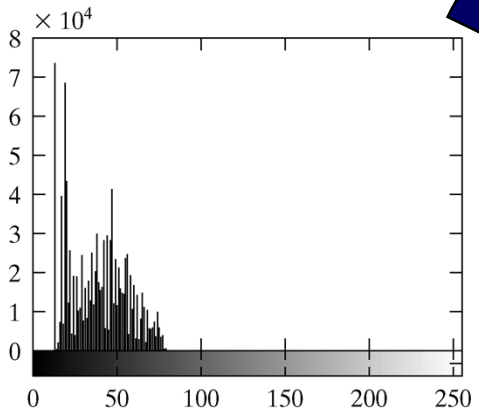
Recall: Histogram Equalization

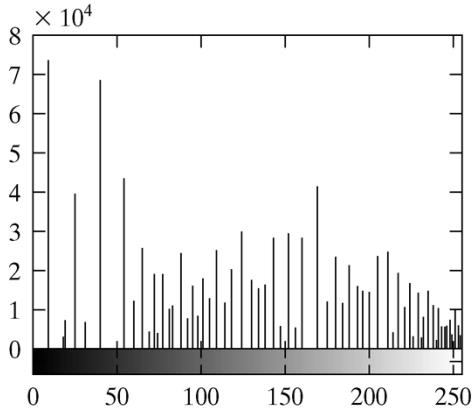
- Adjust 2 different images to make their histograms (intensity distributions) similar
- Apply a point operation that changes histogram of modified image into **uniform distribution**

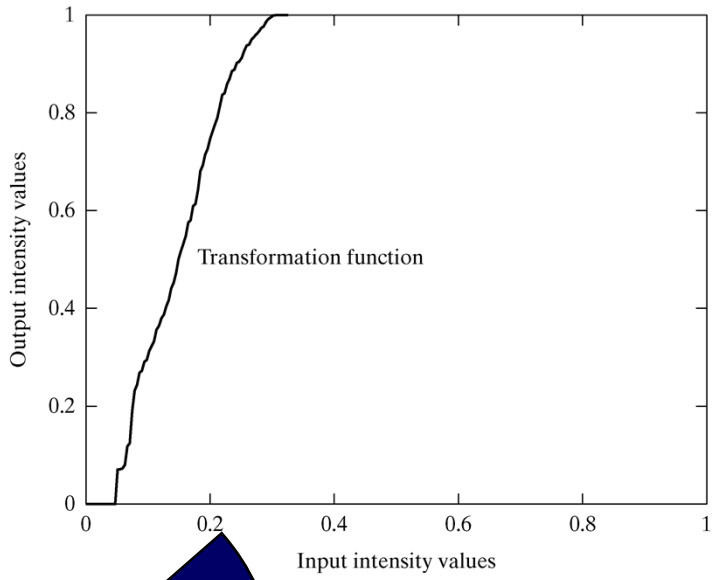


Recall: Equalization Transformation Function



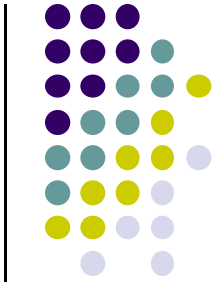






Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Linear Histogram Equalization



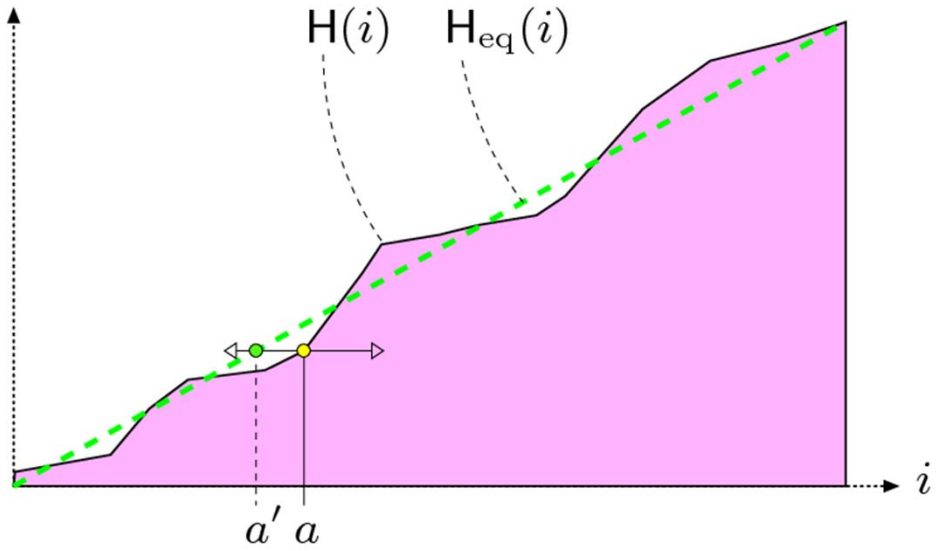
- Histogram cannot be made exactly flat – peaks cannot be increased or decreased by point operations.
- Following point operation makes histogram as flat as possible:

(assuming *M x N* image and pixels in range [0, *K* - 1])

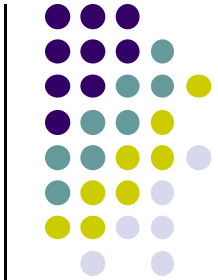
Point operation that returns Linear equalized value of *a*

$$f_{\text{eq}}(a) = \left\lfloor \mathsf{H}(a) \cdot \frac{K-1}{MN} \right\rfloor$$

Cumulative Histogram: Σ how many times intensity a occurs



Effects of Linear Histogram Equalization

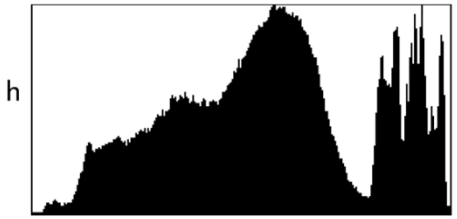


Original Image *I*

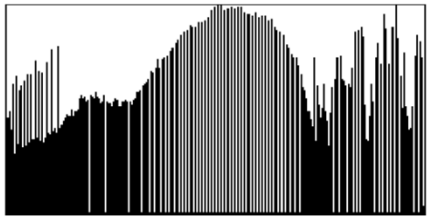
(a) (b)

Image *I'* after Linear Equalization

Original histogram



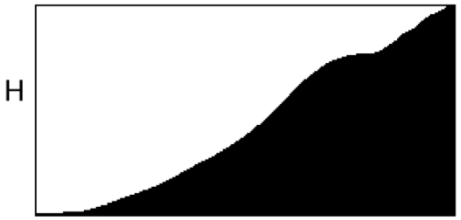
(c)



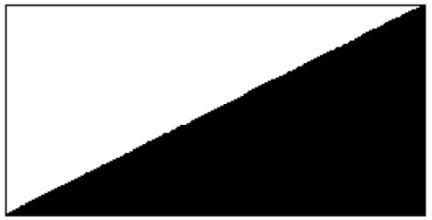
(d)

Histogram after Linear Equalization

Cumulative Histogram



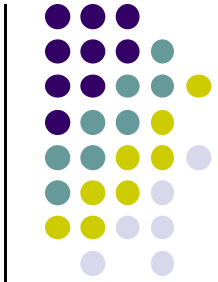
(e)



(f)

Cumulative Histogram After Linear Equalization

Sample Linear Equalization Code



```
1   public void run(ImageProcessor ip) {
2       int w = ip.getWidth();
3       int h = ip.getHeight();
4       int M = w * h;   // total number of image pixels
5       int K = 256;     // number of intensity values
6
7       // compute the cumulative histogram:
8       int[] H = ip.getHistogram();
9       for (int j = 1; j < H.length; j++) {
10          H[j] = H[j-1] + H[j];
11      }
12
13      // equalize the image:
14      for (int v = 0; v < h; v++) {
15          for (int u = 0; u < w; u++) {
16              int a = ip.get(u, v);
17              int b = H[a] * (K-1) / M;
18              ip.set(u, v, b);
19          }
20      }
21  }
```

Obtain histogram of image *ip*

Compute cumulative histogram in place

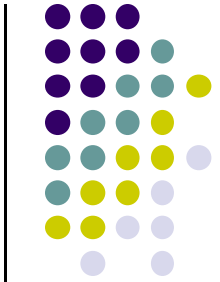
Get intensity value at (u,v)

Equalize pixel intensity

Cumulative Histogram: Σ how many times intensity a occurs

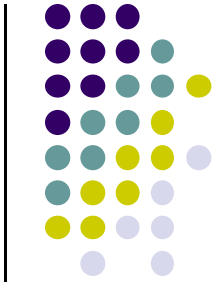
$$f_{\text{eq}}(a) = \left\lfloor \mathsf{H}(a) \cdot \frac{K-1}{MN} \right\rfloor$$

Histogram Specification



- Real images never show uniform distribution (unnatural)
- Most real images, distribution of pixel intensities is gaussian
- **Histogram specification**
  - modifies an image's histogram into an arbitrary intensity distribution (may not be uniform)
- Image 1's histogram can also be used as target for image 2
  - Why? Makes images taken by 2 different cameras to appear as if taken by same camera

Images and Probability



Histograms can be interpreted as probabilities.

Question: If I pick a pixel from an image at random, what is the probability that the pixel has intensity i?

Answer: $P(I(u,v) = i) = \frac{\text{\# pixels with value } i}{\text{total \# pixels}}$

Or, in terms of the histogram h:

$$P(I(u,v) = i) = h(i)/wh$$

Histogram Specification

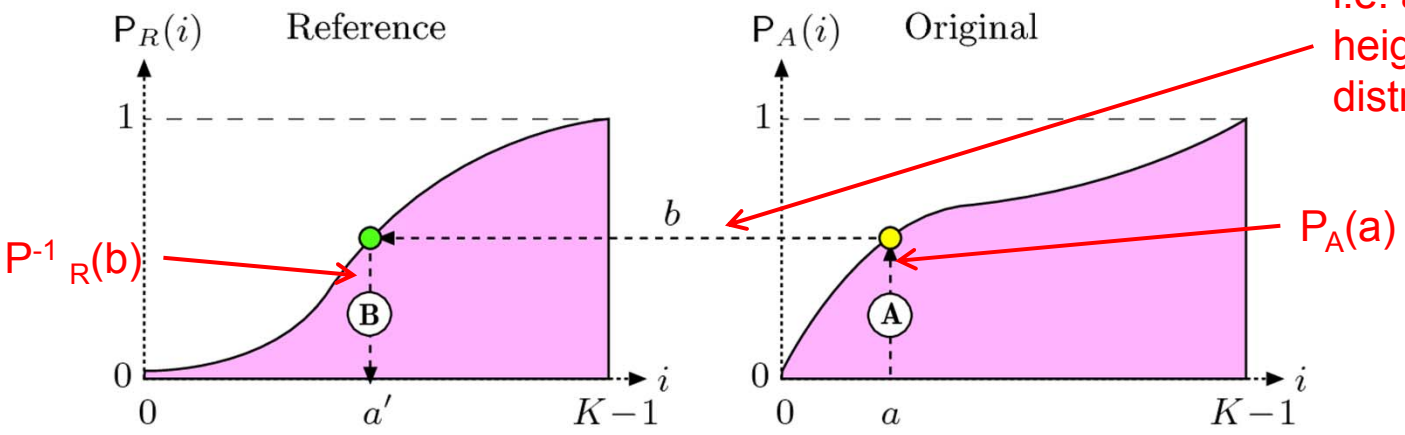
- Find a mapping such that distribution of *a* matches some reference distribution.i.e

$$a' = f_{\text{hs}}(a)$$

Mapping function: maps distribution on right to equivalent point (same height) On distribution on left

to convert original image I_A *into* $I_{A'}$ such that

$$\mathrm{P}_{A'}(i) \approx \mathrm{P}_R(i) \qquad \text{for } 0 \le i < K$$



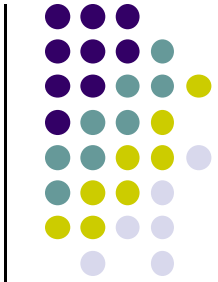
i.e. *a* and *a'* have same height (b) on different CDF distributions

$P^{-1}{}_R(b)$

$P_A(a)$

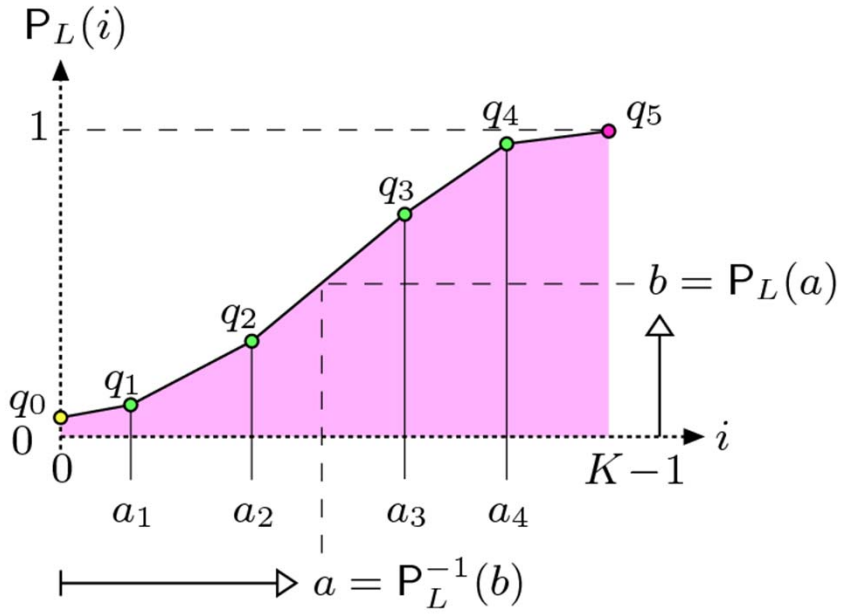
$$f_{\text{hs}}(a) = a' = \mathrm{P}_R^{-1}\big(\mathrm{P}_A(a)\big)$$

Adjusting Linear Distribution Piecewise



- In practice, reference distribution may be specified as a *piecewise linear* function

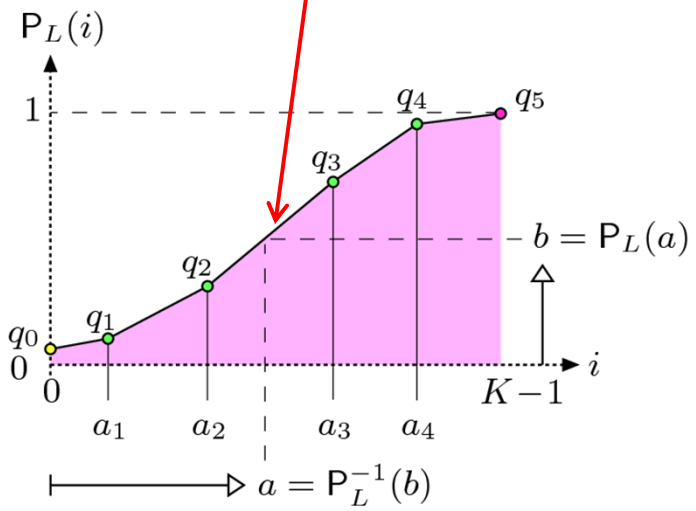
$$\mathcal{L} = [\langle a_0, q_0\rangle, \langle a_1, q_1\rangle, \ldots \langle a_k, q_k\rangle, \ldots \langle a_N, q_N\rangle]$$



- 2 endpoints are fixed $\langle 0, q_0\rangle$ and $\langle K-1, 1\rangle$

Adjusting Linear Distribution Piecewise

For each segment, linearly
Interpolate to find any value



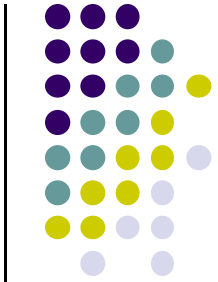
$$\mathsf{P}_L(i) = \begin{cases} q_m + (i - a_m) \cdot \dfrac{(q_{m+1} - q_m)}{(a_{m+1} - a_m)} & \text{for } 0 \le i < K-1 \\ 1 & \text{for } i = K-1 \end{cases}$$

$$\mathsf{P}_L^{-1}(b) = \begin{cases} 0 & \text{for } 0 \le b < \mathsf{P}_L(0) \\ a_n + (b - q_n) \cdot \dfrac{(a_{n+1} - a_n)}{(q_{n+1} - q_n)} & \text{for } \mathsf{P}_L(0) \le b < 1 \\ K-1 & \text{for } b \ge 1 \end{cases}$$

We also need the inverse mapping

$$n = \max\{j \in \{0, \ldots N-1\} \mid q_j \le b\}$$

Adjusting Linear Distribution Piecewise

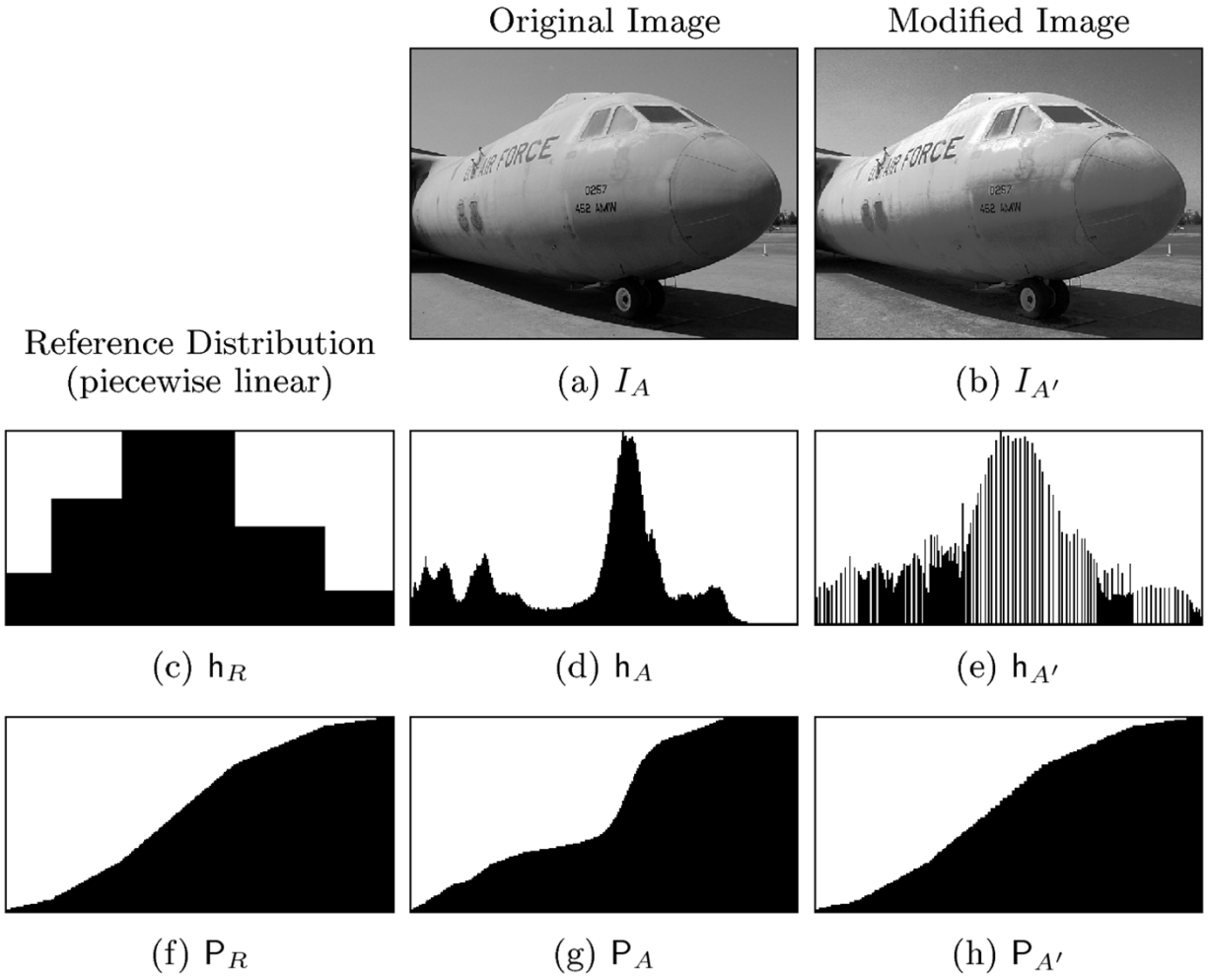


PiecewiseLinearHistogram(h_A, $\mathcal{L}_R$)

h_A: histogram of the original image.
$\mathcal{L}_R$: reference distribution function, given as a sequence of $N+1$ control points $\mathcal{L}_R = [\langle a_0, q_0\rangle, \langle a_1, q_1\rangle, \ldots \langle a_N, q_N\rangle]$, with $0 \le a_k < K$ and $0 \le q_k \le 1$.

Let $K \leftarrow \text{Size}(\mathsf{h}_A)$
Let $\mathsf{P}_A \leftarrow \text{Cdf}(\mathsf{h}_A)$ ▷ cdf for h_A (Alg. 5.1)
Create a table $f_{\text{hs}}[\,]$ of size K ▷ mapping function f_{hs}
for $a \leftarrow 0 \ldots (K-1)$ **do**
  $b \leftarrow \mathsf{P}_A(a)$
  if $(b \le q_0)$ **then**
    $a' \leftarrow 0$
  else if $(b \ge 1)$ **then**
    $a' \leftarrow K-1$
  else
    $n \leftarrow N-1$
    while $(n \ge 0) \wedge (q_n > b)$ **do** ▷ find line segment in $\mathcal{L}_R$
      $n \leftarrow n-1$
    $a' \leftarrow a_n + (b-q_n) \cdot \dfrac{(a_{n+1}-a_n)}{(q_{n+1}-q_n)}$ ▷ see Eqn. (5.23)
  $f_{\text{hs}}[a] \leftarrow a'$
return f_{hs}.

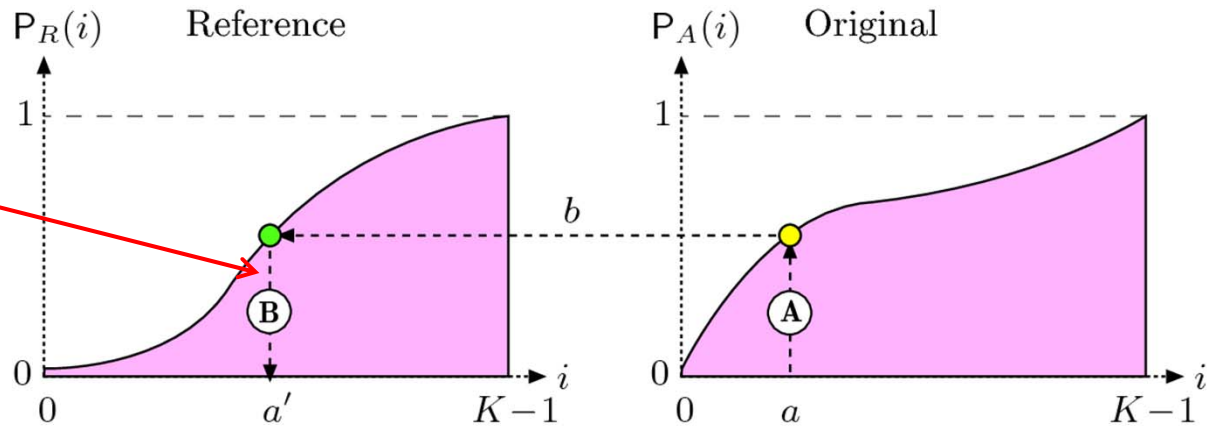
Adjusting Linear Histogram Piecewise



Histogram Matching

- Prior method needed reference distribution to be invertible

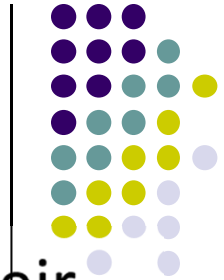
Has to be invertible



$$f_{\text{hs}}(a) = a' = \mathsf{P}_R^{-1}\big(\mathsf{P}_A(a)\big)$$

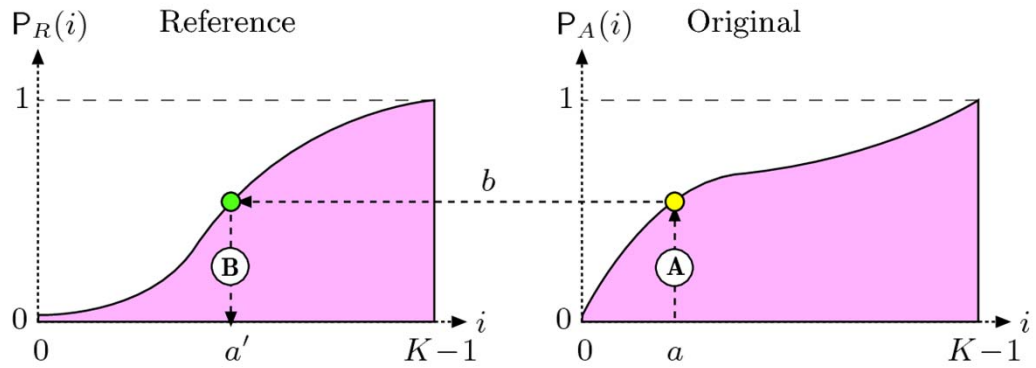
- What if reference histogram is not invertible?
- For example not invertible if histogram has some intensities that occur with probability 0? i.e. p(k) = 0
- Use different method called **histogram matching**

Histogram Matching

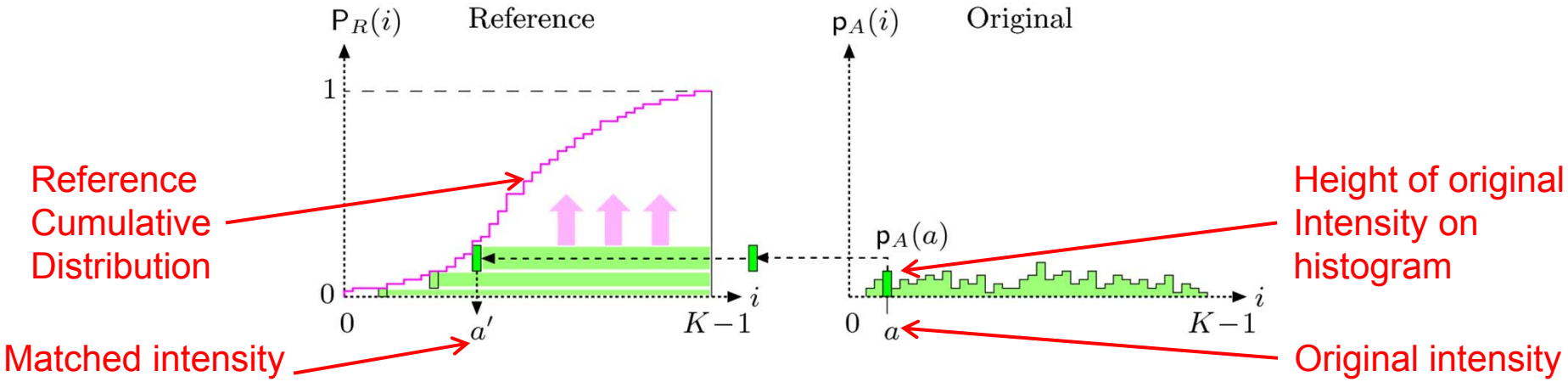


- Given two images I_A and I_B, we want to make their intensity profiles look as similar as possible.
- How?
  - "Match" their cumulative histograms H_A and H_B
- Works well for images with similar content.
- Looks bad for images with different content.

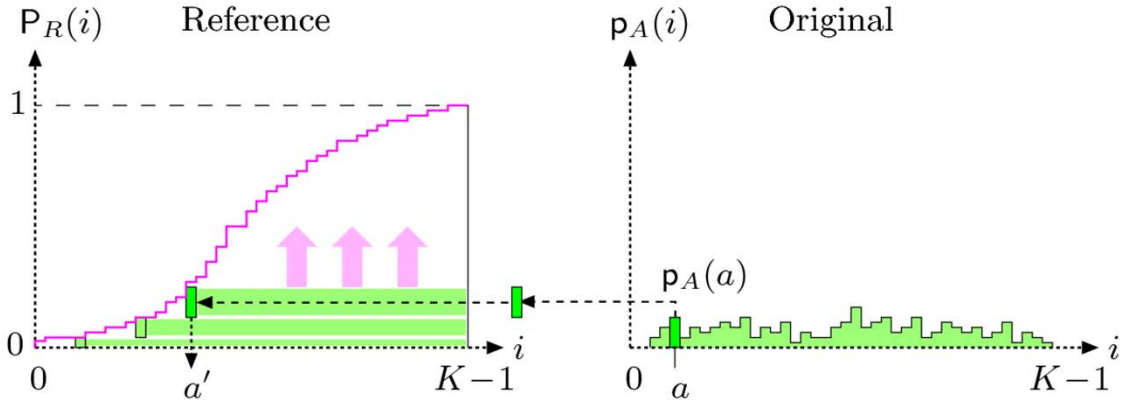
Adjusting to a Given Histogram



$$f_{\text{hs}}(a) = a' = \min\{\, j \mid (0 \le j < K) \wedge (P_A(a) \le P_R(j)) \,\}$$

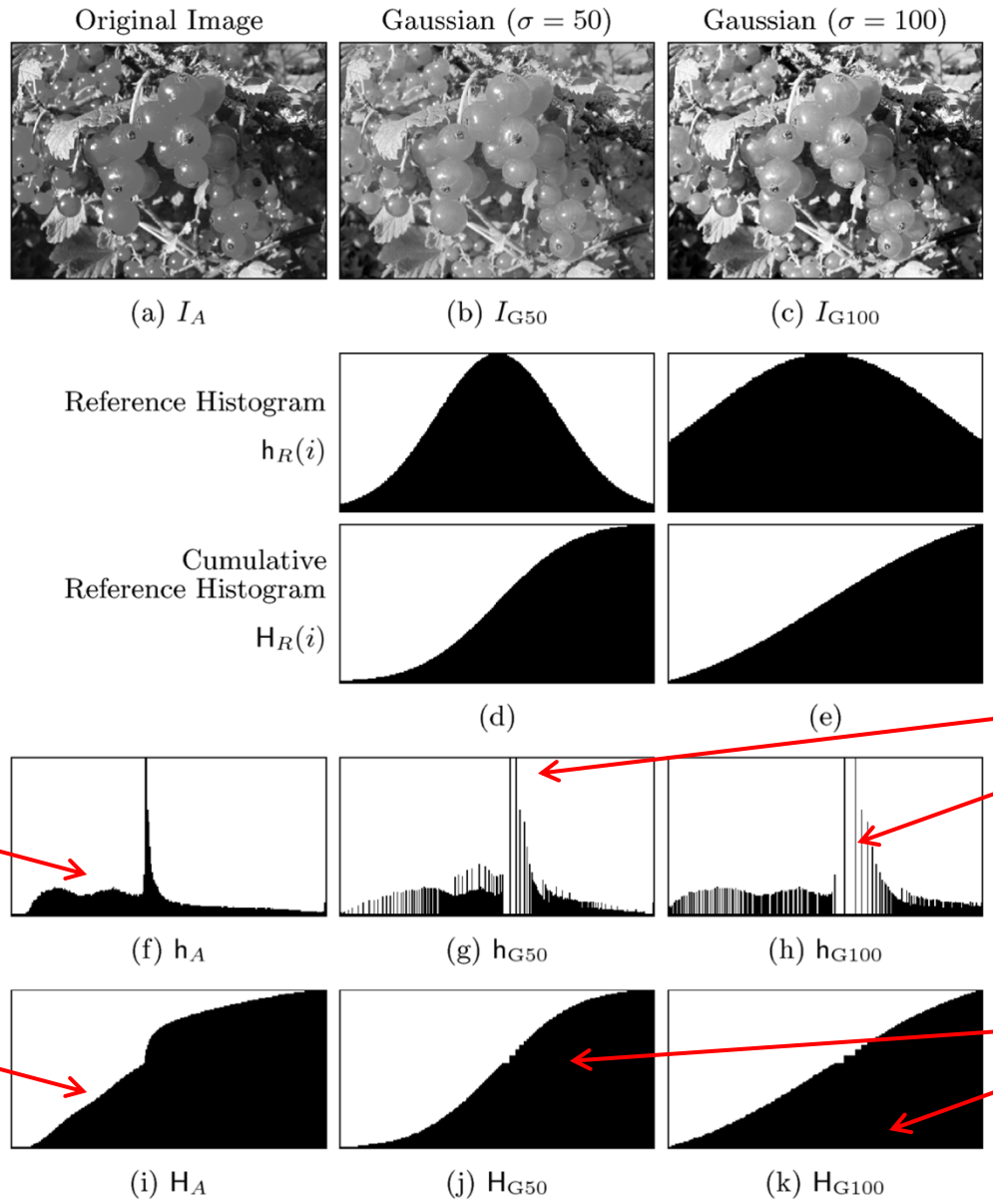


Adjusting to a Given Histogram




```
1:  MatchHistograms(h_A, h_R)
        h_A: histogram of the target image
        h_R: reference histogram (of same size as h_A)
2:      Let K ← Size(h_A)
3:      Let P_A ← Cdf(h_A)                          ▷ cdf for h_A (Alg. 5.1)
4:      Let P_R ← Cdf(h_R)                          ▷ cdf for h_R (Alg. 5.1)
5:      Create a table f_hs[ ] of size K            ▷ pixel mapping function f_hs
6:      for a ← 0...(K−1) do
7:          j ← K−1
8:          repeat
9:              f_hs[a] ← j
10:             j ← j − 1
11:         while (j ≥ 0) ∧ (P_A(a) ≤ P_R(j))
12:     return f_hs.
```

Adjusting to a Given Histogram



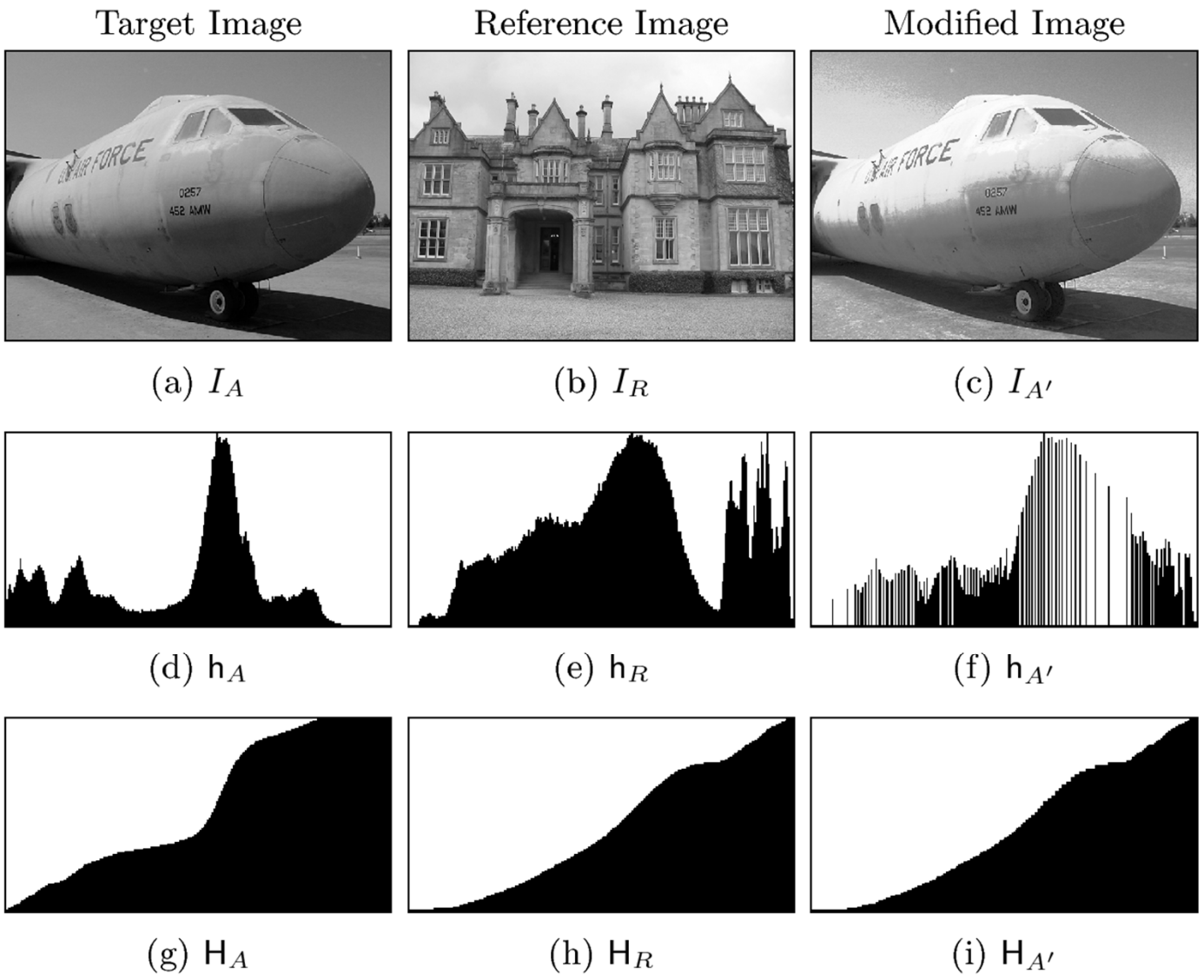
original histogram

Original histogram after matching

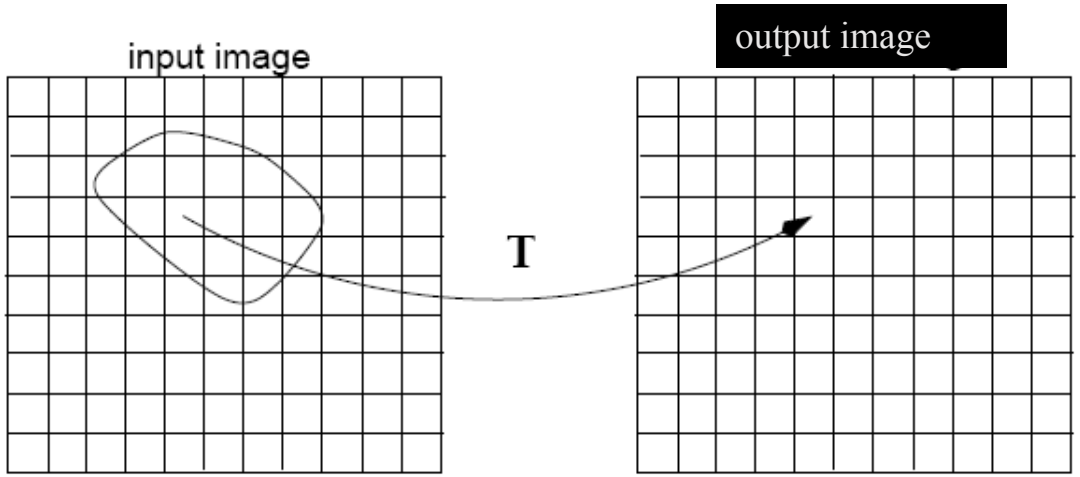
CDF of original histogram

CDF of original histogram after matching

Adjusting to a Given Histogram



Spatial Filtering



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

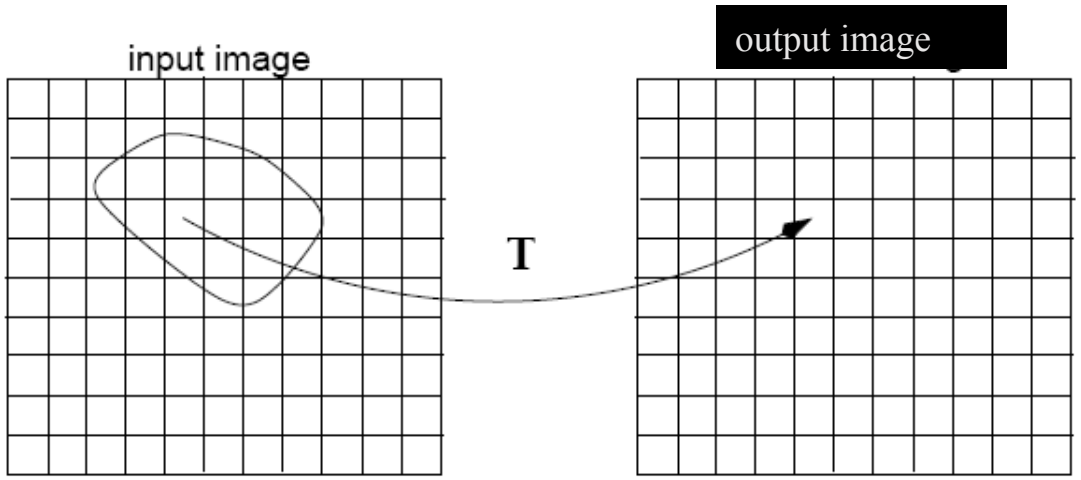
T operates on a neighborhood of pixels

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

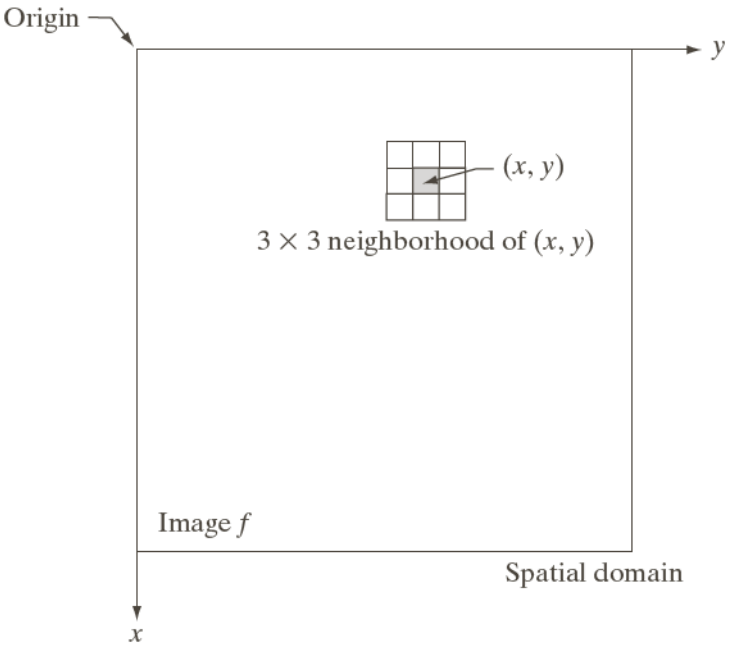


$$g(x,y) = T[f(x,y)]$$

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



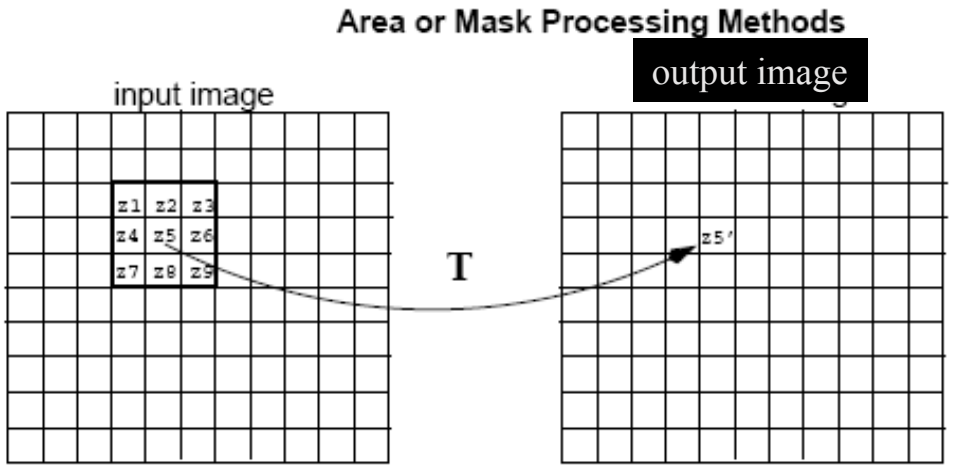
Spatial filtering - Operation

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?

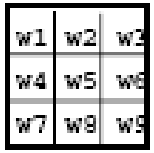
$$T(x) = 2x_1 + 3x_2$$

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

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

Linear vs Non-Linear

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



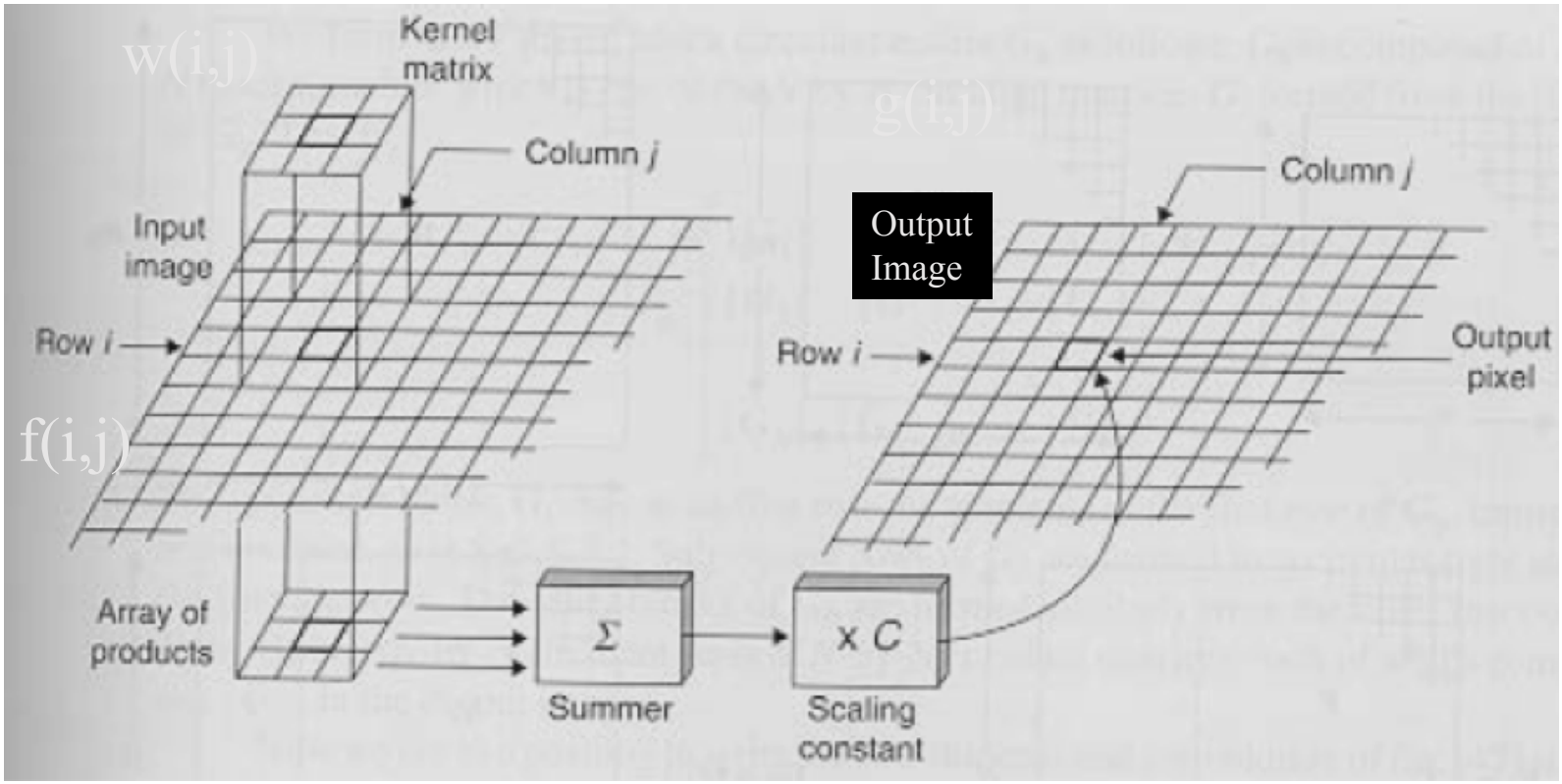
$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

- 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

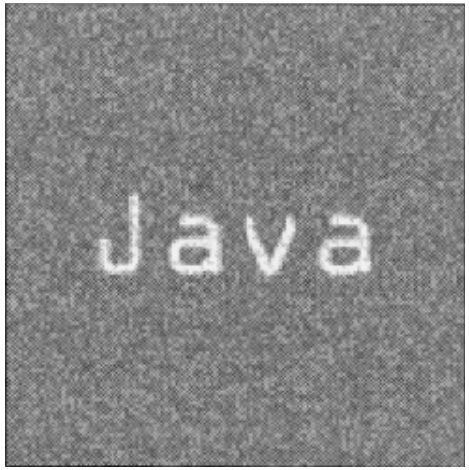
- Main linear spatial filtering methods:
  - Correlation
  - Convolution

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.

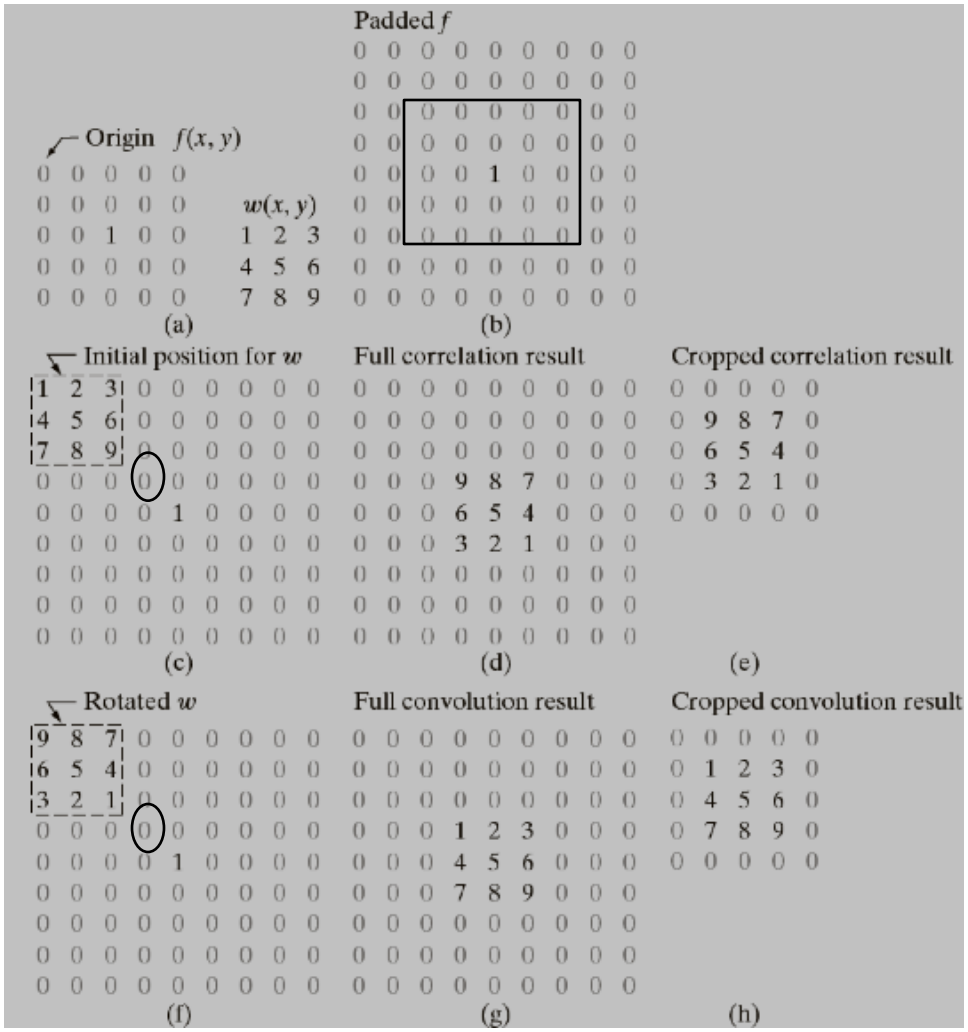
$$g(i,j) = w(i,j) * f(i,j) = \sum_{s=-K/2}^{K/2} \sum_{t=-K/2}^{K/2} w(s,t) f(i-s, j-t)$$

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

Example

Correlation:

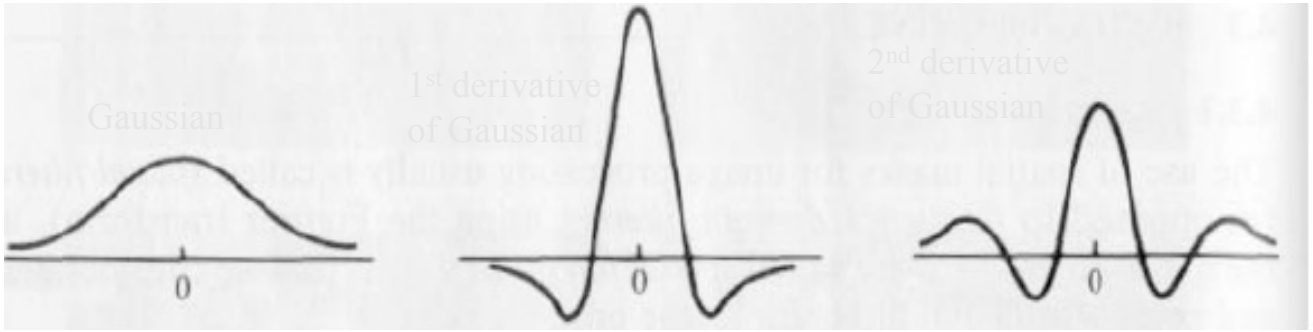
Convolution:



How do we choose the mask weights?

- Typically, by sampling certain functions:

w1	w2	w3
w4	w5	w6
w7	w8	w9

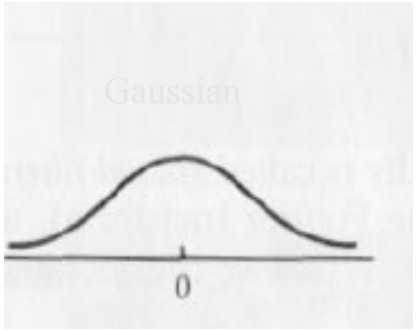


Filters

- 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

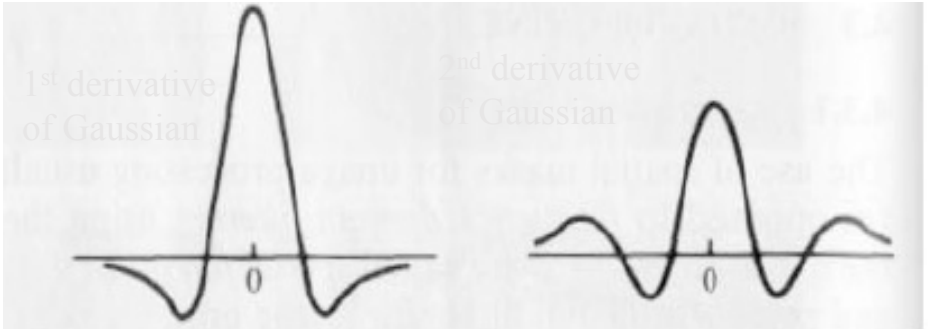
- Useful for reducing noise and eliminating small details.

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.

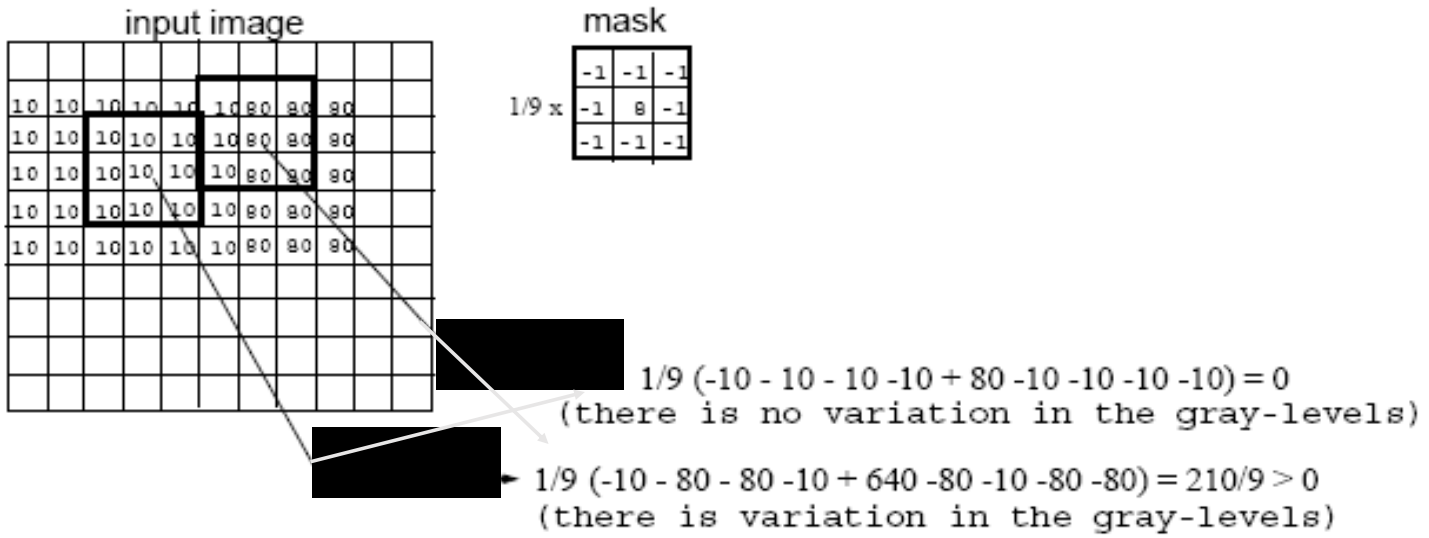


mask

-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]

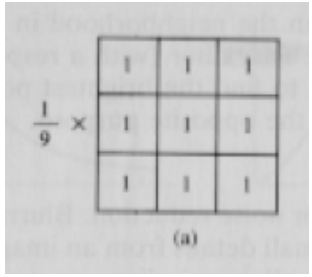
input image

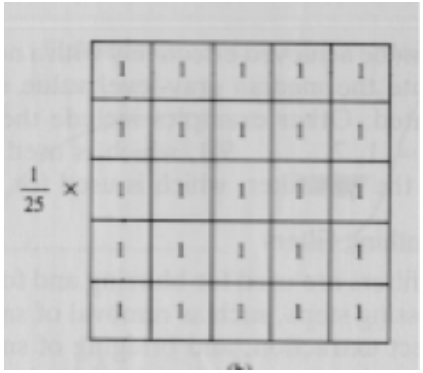
sharpened image

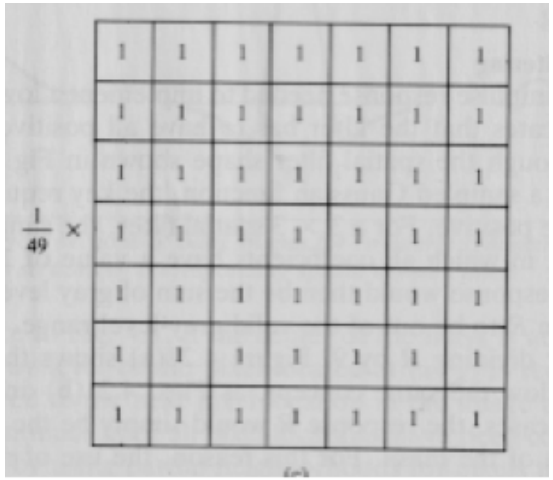
Smoothing Filters

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

Smoothing Filters: Averaging



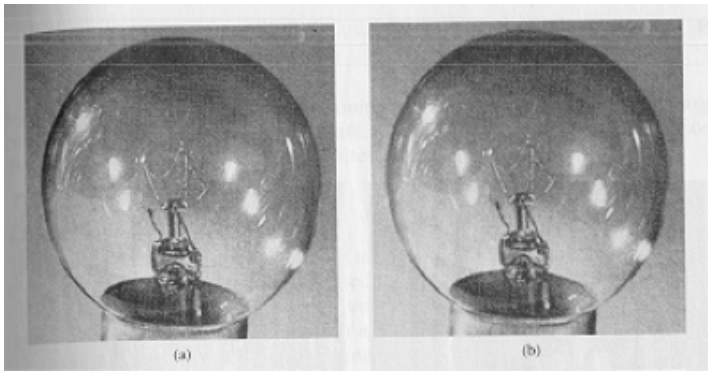




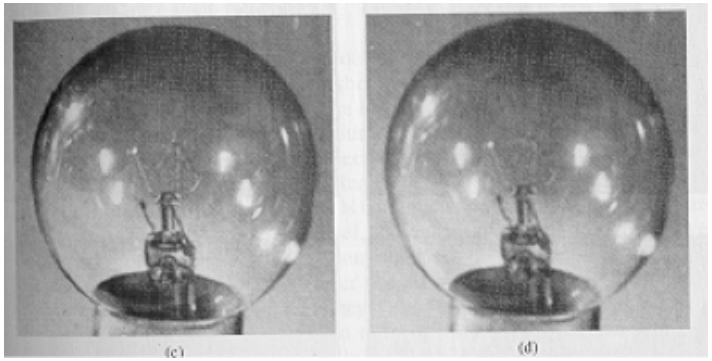
Smoothing Filters: Averaging (cont'd)

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

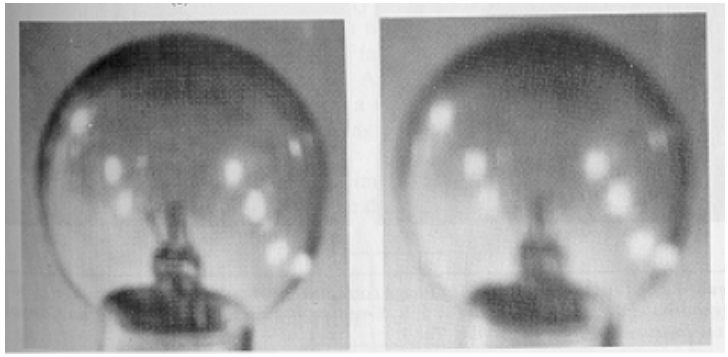
original 3x3



5x5 7x7



15x15 25x25

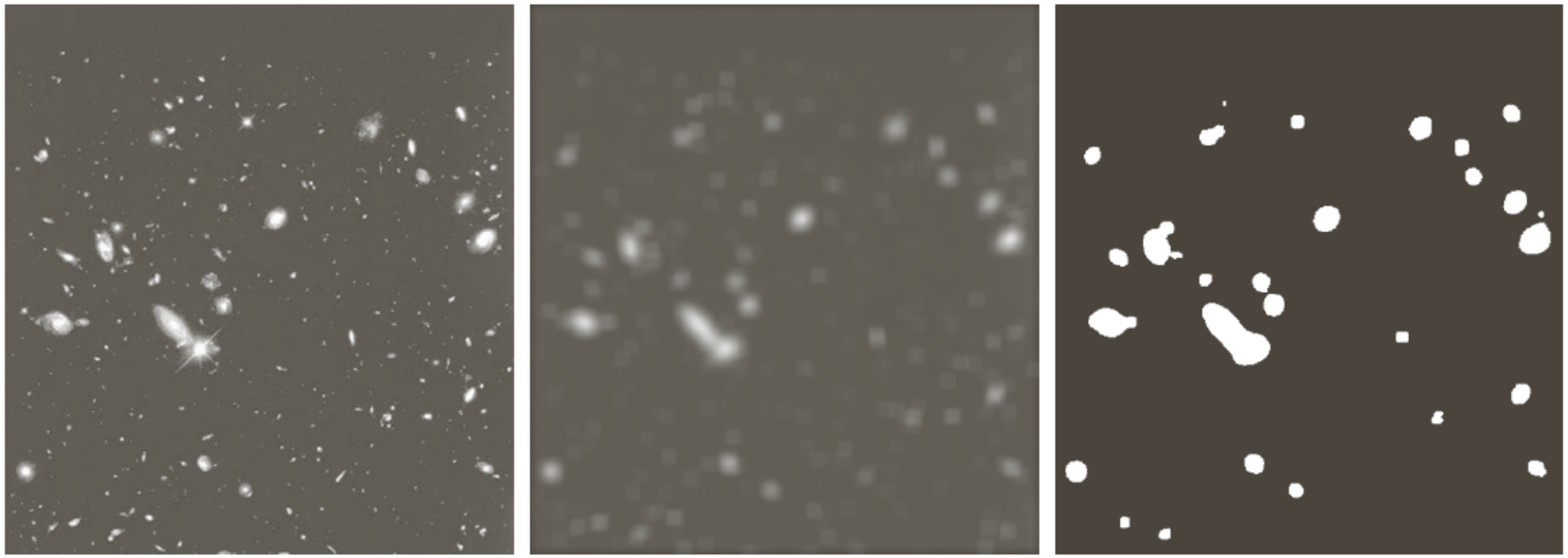


Smoothing Filters: Averaging (cont'd)

Example: extract largest, brightest objects

15 x 15 averaging

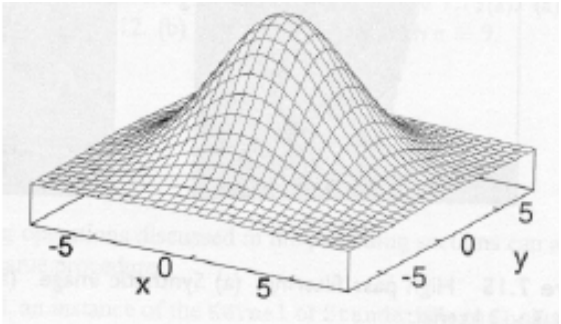
image thresholding



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)

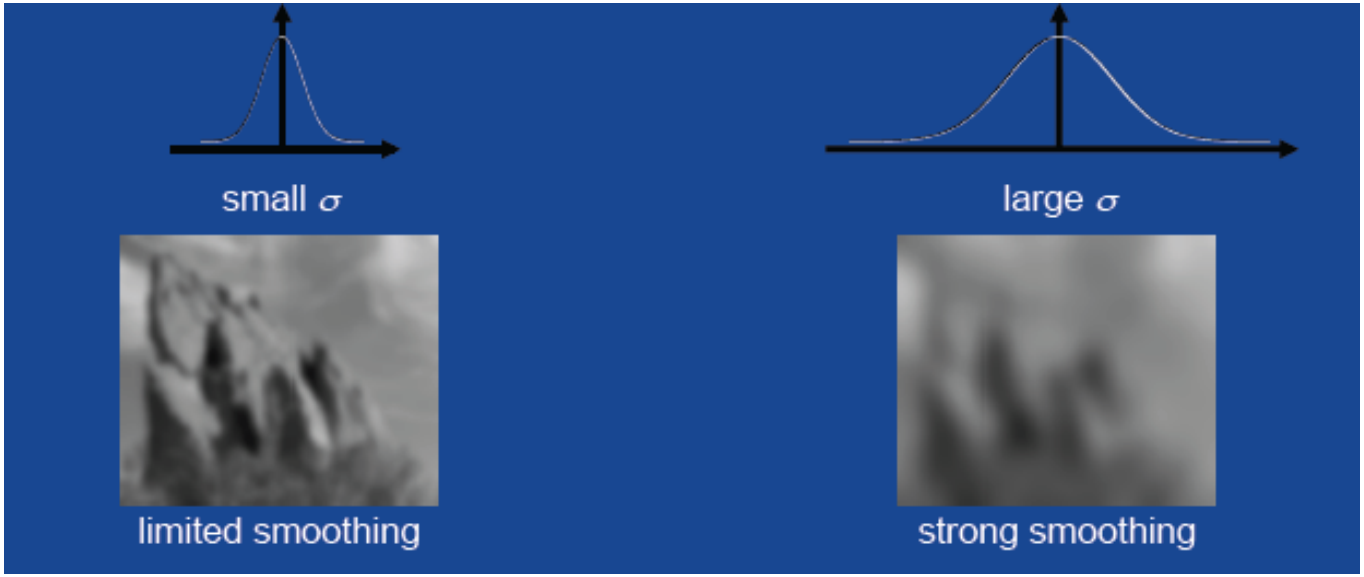
- **σ** controls the amount of smoothing
- As **σ** increases, more samples must be obtained to represent the Gaussian function accurately.

$\sigma = 3$

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)



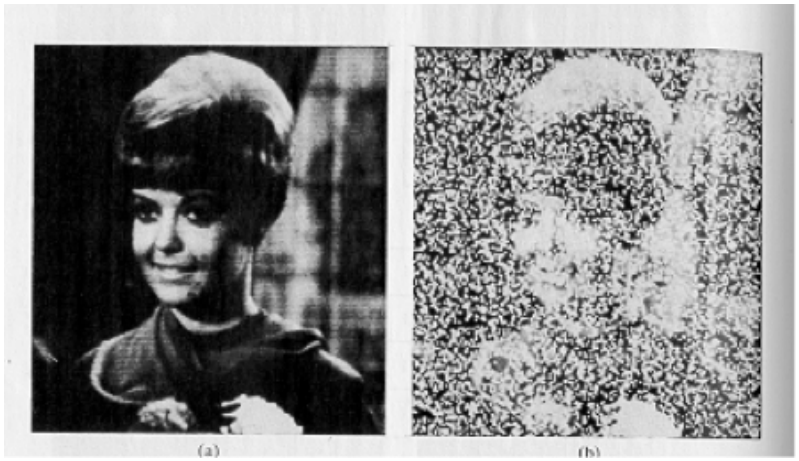
Averaging vs Gaussian Smoothing

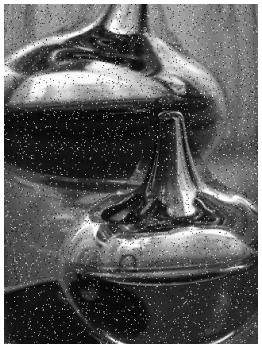
Averaging

Gaussian

Smoothing Filters: Median Filtering (non-linear)

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



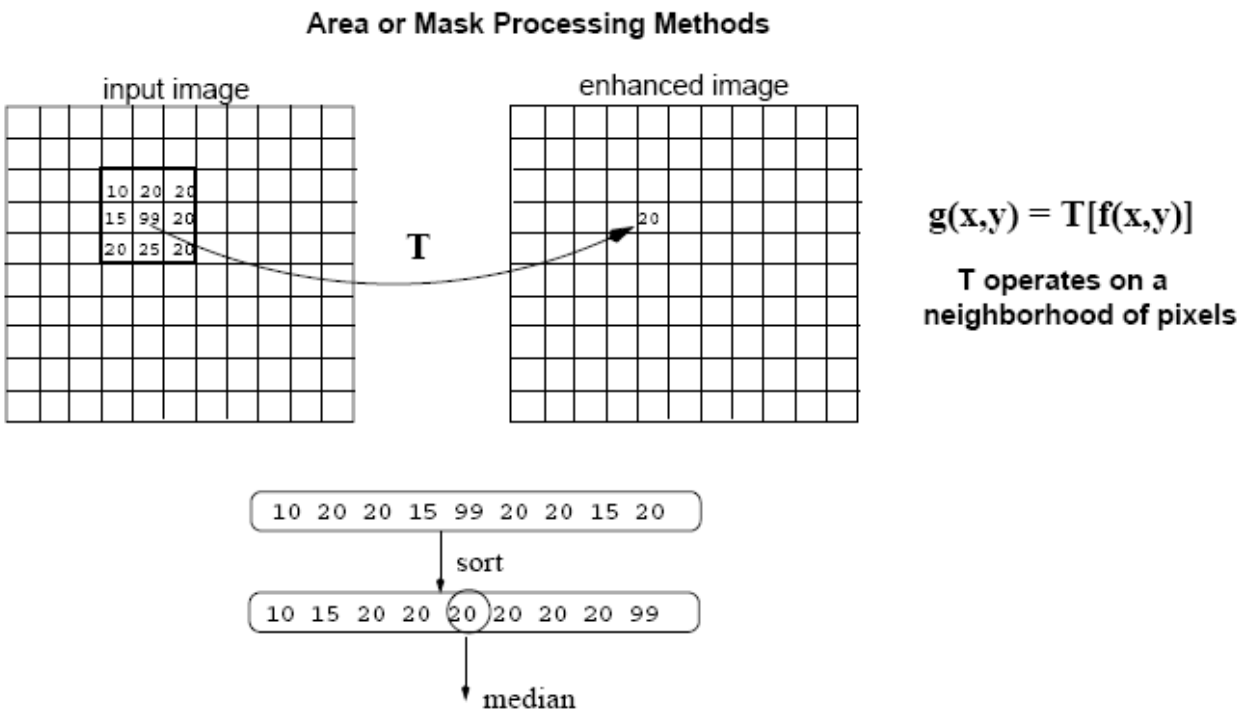


averaging

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:

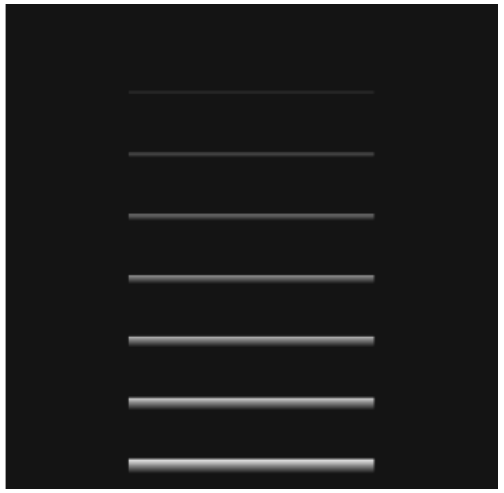
A>=1
w = 9A-1

-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

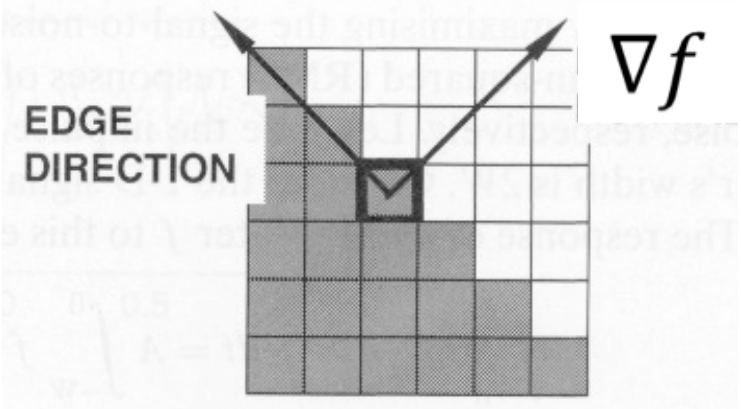
- The gradient is a **vector** which has magnitude and direction:

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

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

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)

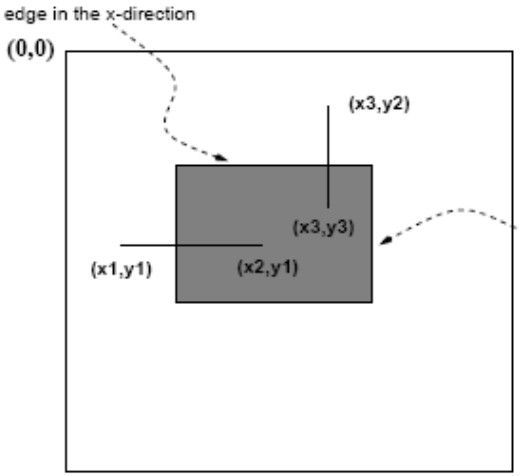
$$\frac{f(x3,y2) - f(x3,y3)}{y2 - y3}$$

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

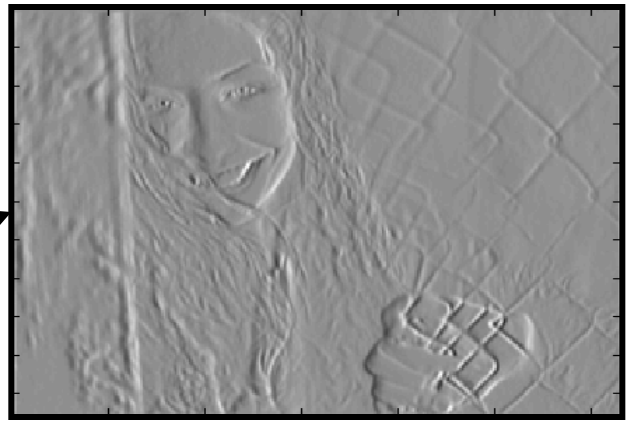
$\boxed{f(x+1,y) - f(x,y)}$

(grad in the x-direction)

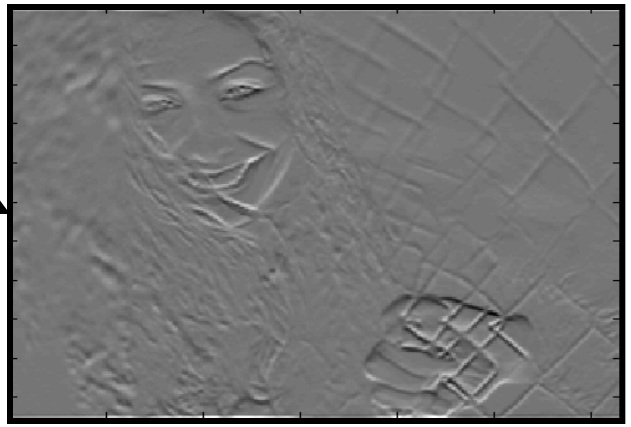
(x2=x+Dx, x1=x, y1=y2=y, Dx=1)

Example: visualize partial derivatives

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



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



Implement Gradient Using Masks

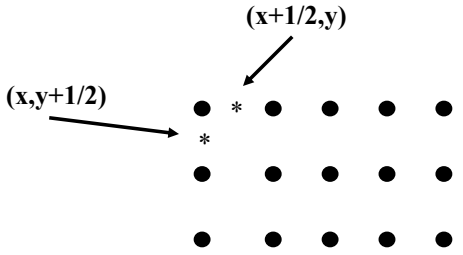
- We can implemen $\frac{\partial f}{\partial x}$ ar $\frac{\partial f}{\partial y}$ using masks:

-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

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

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

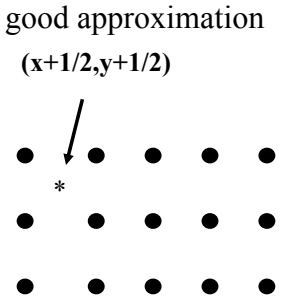
$$mag(grad(f)) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

Implement Gradient Using Masks (cont'd)

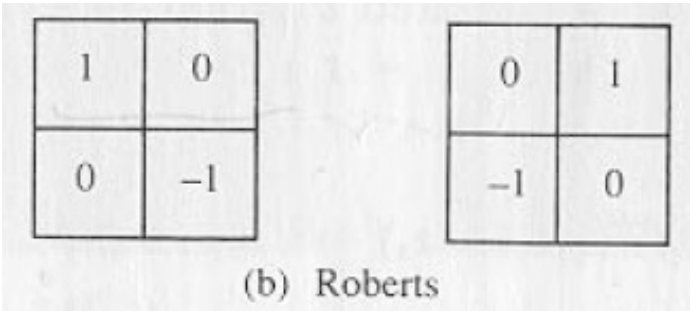
- A different approximation of the gradient:

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

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

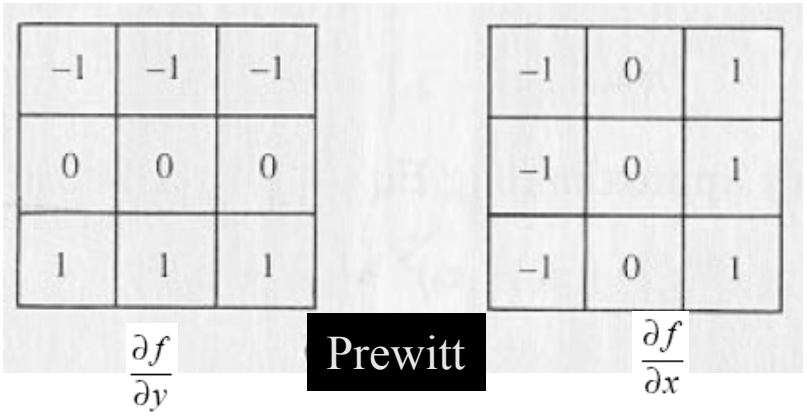


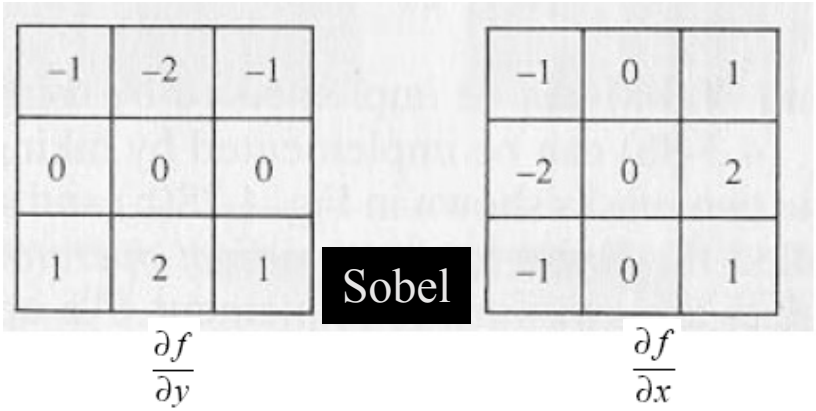
- We can implement $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ using the following masks:



Implement Gradient Using Masks (cont'd)

- Other approximations

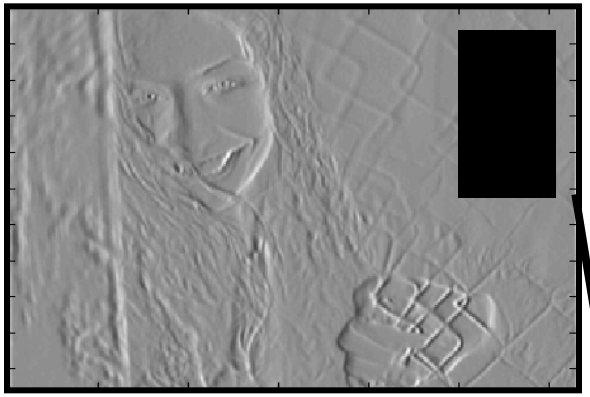


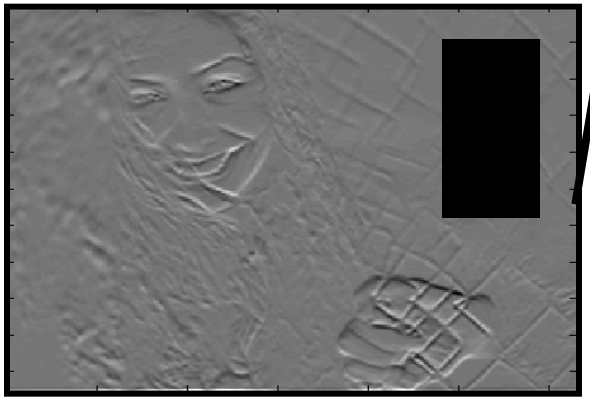


Example: Gradient Magnitude Image

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

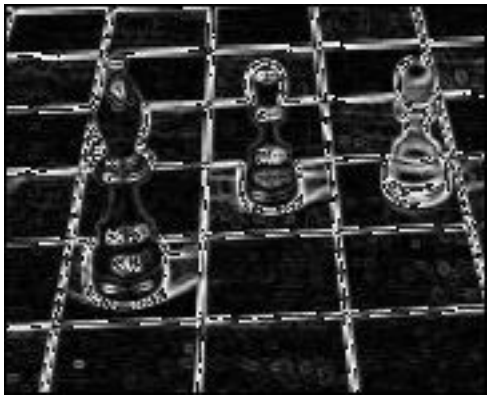
Edges can be found
by detect the zero-crossings

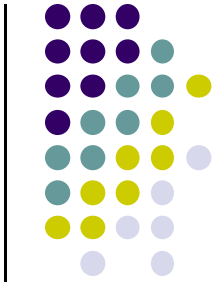
-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

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