Image Processing II

COMPUTER VISION (HY24011)
HANYANG UNIV. ERICA
Q YOUN HONG (홍규연)

Content

- **Area-based Operation**
- Binary Image Processing
- Geometric Transformation
- Image Pyramid

Neighborhood (Area-based) Operators

- Neighborhood operator(filter): pixel value computed using a collection of pixel values in a small neighborhood
- Linear filter: a pixel's value is a weighted sum of pixel values within a small neighborhood N
- Correlation and convolution operators

$$\text{1D Correlation: } g(i) = u \otimes f = \sum_{x=-(w-1)/2}^{(w-1)/2} u(x) f(i+x)$$

$$\text{1D Convolution: } g(i) = u \circledast f = \sum_{x=-(w-1)/2}^{(w-1)/2} u(x) f(i-x)$$

$$\text{2D Correlation: } g(i) = u \otimes f = \sum_{x=-(w-1)/2}^{(w-1)/2} \sum_{y=-(h-1)/2}^{(h-1)/2} u(y,x) f(j+y, i+x)$$

$$\text{2D Convolution: } g(i) = u \circledast f = \sum_{x=-(w-1)/2}^{(w-1)/2} \sum_{y=-(h-1)/2}^{(h-1)/2} u(y,x) f(j-y, i-x)$$

Linear Filtering: 1D Example

- Correlation: matching a window with an image
  - u(x): called a window, a weighted kernel, mask, filter (coefficients)
- Convolution: flipping the window and performing matching
  - u(x): also called the impulse response ($\because u \circledast \delta(i,j) = u$)

Window u $u =$

2	4	3

	0	1	2	3	4	5	6	7	8	9
$f=$	0	0	1	0	2	2	4	3	1	0
			×	×	×					
			2	4	3					
				+						
$g=$	-	3	4	8	14	24	29	23	10	-

Correlation

	0	1	2	3	4	5	6	7	8	9
$f=$	0	0	1	0	2	2	4	3	1	0
			×	×	×					
			3	4	2					
				+						
$g=$	-	2	4	7	12	22	28	26	13	-

Convolution

Linear Filtering: 2D Example

Input $f=$

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	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	0
0	0	0	0	0	0	0	0

Window u $=$

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

Correlation:

1	2	3
4	5	6
7	8	9

Output $g=$

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

Correlation

Convolution:

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	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	0
0	0	0	0	0	0	0	0

9	8	7
6	5	4
3	2	1

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

Convolution

- Here, input f is an impulse response function
- When applying the convolution filter to f,

the output image will the same as the window u

- When applying the correlation filter to f,

the window will be reversed in the output

Properties of Convolution (Correlation)

- Both correlation and convolution are linear shift-invariant (LSI)
  - The superposition principle: $u \circ (f_0 + f_1) = u \circ f_0 + u \circ f_1$
  - The shift invariance principle:

$$g(j,i) = f(j+k, i+l) \;\Leftrightarrow\; (u \circ g)(j,i) = (u \circ f)(j+k, i+l)$$

$\Rightarrow$ The operator "behaves the same everywhere"

- Correlation and convolution can be written as a matrix-vector multiplication: g = **U**f

$$\begin{array}{|c|c|c|c|c|}\hline 72 & 88 & 62 & 52 & 37 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline {}^1\!/_4 & {}^1\!/_2 & {}^1\!/_4 \\ \hline \end{array} \quad \Leftrightarrow \quad \frac{1}{4}\begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Figure 3.12 *One-dimensional signal convolution as a sparse matrix-vector multiplication,*

Separable Filtering

- General convolution filters: requires K^2 operations per pixel
- Separable convolution filter: perform 1D horizontal convolution followed by 1D vertical convolution $\Longrightarrow 2K$ operations per pixel
  - Represent 2D kernel $\mathbf{K} = \mathbf{v}\mathbf{h}^{\mathbf{T}}$
  - Examples of separable filters: box, bilinear, Gaussian, Sobel, LOG

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \quad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \quad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \quad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Examples of Linear Filtering

- (a) Box filter (moving average filter): averages the pixels in a K x K window
- (b) Bilinear filter (Bartlett filter): smooths image with a piecewise “tent” function
- (c) Gaussian filter: made by convolving the linear tent function with itself
- (d) Sobel filter: simple 3x3 edge extractor

  (combination of horizontal central difference and a vertical tent filter)
- (e) Simple corner detector: look for simultaneous horizontal/vertical second derivatives (+diagonal edges)

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array}$$

$$\frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(a) box, $K = 5$ (b) bilinear (c) “Gaussian” (d) Sobel (e) corner

Examples of Linear Filtering

Original

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(A)

.0000	.0000	.0002	.0000	.0000
.0000	.0113	.0837	.0113	.0000
.0002	.0837	.6187	.0837	.0002
.0000	.0113	.0837	.0113	.0000
.0000	.0000	.0002	.0000	.0000

(B)

0	-1	0
-1	5	-1
0	-1	0

(C)

1	1	1
0	0	0
-1	-1	-1

(D)

1	0	-1
1	0	-1
1	0	-1

(E)

.0304	.0501	0	0	0
.0501	.1771	.0519	0	0
0	.0519	.1771	.0519	0
0	0	.0519	.1771	.0501
0	0	0	.0501	.0304

(F)

Box

Gaussian

Sharpening

Horizontal Edge

Vertical Edge

Motion

Summed Area Table (Integral Image)

- Accelerate convolution when box filters of different sizes are used
- Summed area table: $s(j,i) = \sum_{k=0}^{j} \sum_{l=0}^{i} f(k,l)$ or
$$s(j,i) = s(j-1,i) + s(j,i-1) - s(j-1,i-1) + f(j,i)$$
- $s(j,i)$ is also called an integral image
- To find the summed area in $[j_0, j_1] \times [i_0, i_1]$, we need 4 samples:

$$S([j_0, j_1], [i_0, i_1]) = s(j_1, i_1) - s(j_1, i_0 - 1) - s(j_0 - 1, i_1) + s(j_0 - 1, i_0 - 1)$$

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

(a) S = 24

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

(b) s = 28

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

(c) S = 24

(a) Original image
(b) Summed area table
(c) Computation of area sum

Non-Linear Filtering

- Linear filter computes a weighted sum of input pixels
- Non-linear filters perform better in some applications
  - E.g. Edge-preserving filtering, removing shot noises
- Example non-linear filters:

  median filter, bilateral filter

Original

With salt-pepper noises

Gaussian Filter

Median Filter

Non-Linear Filtering

- Median filter: selects the median value from each pixel's neighborhood
  - Can be implemented via linear-time algorithm
  - Robust to removing shot noises while preserving edges

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

(a) median = 4

- Bilateral filter: reject pixels whose values differ too much from the central pixel value (in a soft way)

$$g(j,i) = \frac{\sum_{k,l} f(k,l)w(j,i,k,l)}{\sum_{k,l} w(j,i,k,l)}, \text{ where } w(j,i,k,l) = \exp\left(-\frac{(j-k)^2+(i-l)^2}{2\sigma_d^2} - \frac{\|f(j,i)-f(k,l)\|^2}{2\sigma_r^2}\right)$$

w(j,I,k,l): Bilateral weight function

Domain kernel

Data-dependent range kernel

Non-Linear Filtering

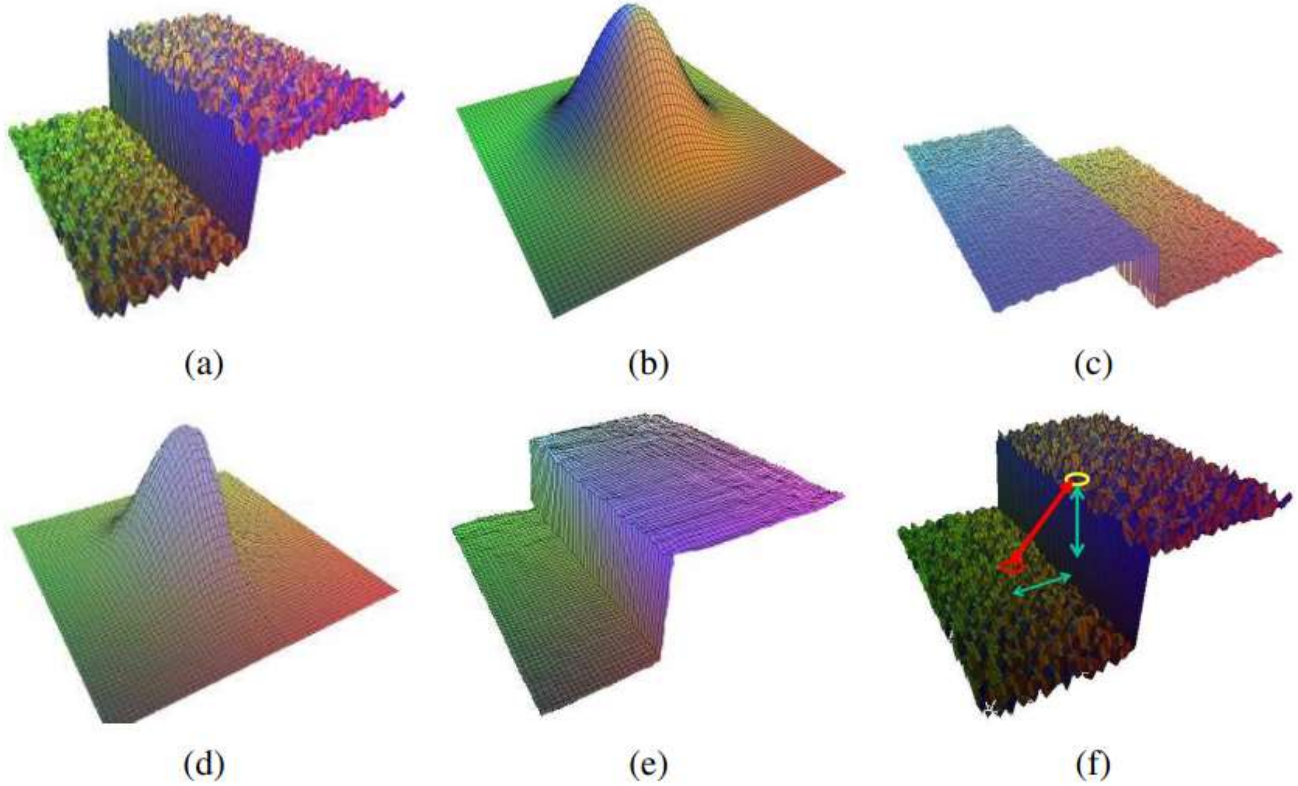


Figure 3.20 *Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.*

Content

- Area-based Operation (cont’d)
- **Binary Image Processing**
- Geometric Transformation
- Image Pyramid

Binary Image Processing

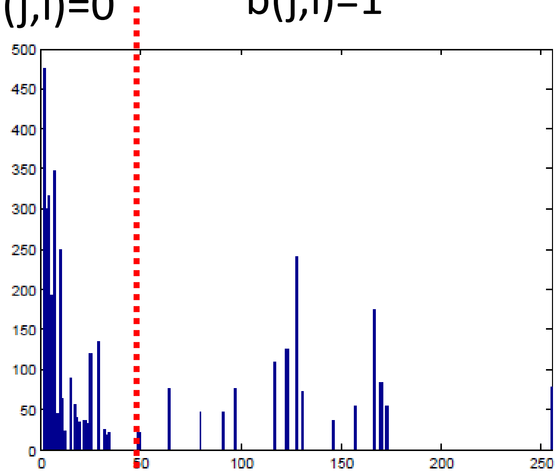
- Binary image: an image only consisting of 0's and 1's
- Binarization: binary image occurs after thresholding operation,

$$b(j,i) = \begin{cases} 1, & \text{if } f(j,i) > T \\ 0, & \text{if } f(j,i) \leq T \end{cases}$$

- Binarization by thresholding
  - Select a threshold T at the valley point in the histogram

Original Image

b(j,i)=0 b(j,i)=1



Histogram

T = 50

How to find a good threshold T?

Otsu's Binarization Algorithm

- Optimization problem: find the best threshold value from the image histogram
  - In a bimodal image (histogram having two peaks), a good threshold is in the middle of the two peaks
  - Objective function: make two split pixel sets as uniform as possible

  $\Rightarrow$ Two sets of pixels after splitting should have small variances

- Otsu's algorithm minimizes the weighted within-class variance:

$$T = \underset{t \in \{0,1,\ldots,L-1\}}{\operatorname{argmin}} \; v_{within}(t)$$

Time complexity: $\theta(L^2)$

$$v_{within}(t) = \boxed{w_0(t)}\boxed{v_0(t)} + \boxed{w_1(t)}\boxed{v_1(t)},$$ where

Weight: $\boxed{w_0(t)} = \sum_{i=0}^{t} \hat{h}(i),$ $\boxed{w_1(t)} = \sum_{i=t+1}^{L-1} \hat{h}(i),$

Mean: $\underline{\mu_0(t)} = \frac{1}{w_0(t)} \sum_{i=0}^{t} i\hat{h}(i),$ $\underline{\mu_1(t)} = \frac{1}{w_1(t)} \sum_{i=t+1}^{L-1} i\hat{h}(i),$

Variance: $\boxed{v_0(t)} = \frac{1}{w_0(t)} \sum_{i=0}^{t} \hat{h}(i)\left(i - \underline{\mu_0(t)}\right)^2,$ $\boxed{v_1(t)} = \frac{1}{w_1(t)} \sum_{i=t+1}^{L-1} \hat{h}(i)\left(i - \underline{\mu_1(t)}\right)^2,$

Otsu's Binarization Algorithm

- For better performance, use recurrence relation on t
- For every t, $\mu = \sum_{i=0}^{L-1} i\hat{h}(i)$, $v = \sum_{i=0}^{L-1}(i-\mu)^2\hat{h}(i)$
- Rewrite the equation of v:

$$w_0(t) = \sum_{i=0}^{t} \hat{h}(i), \qquad w_1(t) = \sum_{i=t+1}^{L-1} \hat{h}(i),$$
$$\mu_0(t) = \frac{1}{w_0(t)}\sum_{i=0}^{t} i\hat{h}(i), \qquad \mu_1(t) = \frac{1}{w_1(t)}\sum_{i=t+1}^{L-1} i\hat{h}(i),$$
$$v_0(t) = \frac{1}{w_0(t)}\sum_{i=0}^{t} \hat{h}(i)\big(i-\mu_0(t)\big)^2, \quad v_1(t) = \frac{1}{w_1(t)}\sum_{i=t+1}^{L-1} \hat{h}(i)\big(i-\mu_1(t)\big)^2,$$

$$
\begin{aligned}
v &= \sum_{i=0}^{t}(i-\mu_0(t)+\mu_0(t)-\mu)^2\,\hat{h}(i) + \sum_{i=t+1}^{L-1}(i-\mu_1(t)+\mu_1(t)-\mu)^2\hat{h}(i) \\
&= \sum_{i=0}^{t}\Big[\big(i-\mu_0(t)\big)^2 + \cancelto{0}{2(i-\mu_0(t))(\mu_0(t)-\mu)} + (\mu_0(t)-\mu)^2\Big]\,\hat{h}(i) + \\
&\quad \sum_{i=t+1}^{L-1}\Big[\big(i-\mu_1(t)\big)^2 + \cancelto{0}{2(i-\mu_1(t))(\mu_1(t)-\mu)} + (\mu_1(t)-\mu)^2\Big]\,\hat{h}(i) \\
&= \sum_{i=0}^{t}\Big[\big(i-\mu_0(t)\big)^2 + (\mu_0(t)-\mu)^2\Big]\,\hat{h}(i) + \sum_{i=t+1}^{L-1}\Big[\big(i-\mu_1(t)\big)^2 + (\mu_1(t)-\mu)^2\Big]\,\hat{h}(i) \\
&= \sum_{i=0}^{t}\big(i-\mu_0(t)\big)^2\hat{h}(i) + (\mu_0(t)-\mu)^2\sum_{i=0}^{t}\hat{h}(i) + \sum_{i=t+1}^{L-1}\big(i-\mu_1(t)\big)^2\hat{h}(i) + (\mu_1(t)-\mu)^2\sum_{i=t+1}^{L-1}\hat{h}(i) \\
&= \{(\mu_0(t)-\mu)^2 w_0(t) + (\mu_1(t)-\mu)^2 w_1(t)\} + \{w_0(t)v_0(t) + w_1(t)v_1(t)\} \\
&= \{(\mu_0(t)-\mu)^2 w_0(t) + (\mu_1(t)-\mu)^2(1-w_0(t))\} + v_{within}(t) \qquad (\because w_0(t)+w_1(t)=1) \\
&= w_0(t)(1-w_0(t))(\mu_0(t)-\mu_1(t))^2 + v_{within}(t) \qquad (\because w_0(t)\mu_0(t)+w_1(t)\mu_1(t)=\mu) \\
&= v_{between}(t) + v_{within}(t)
\end{aligned}
$$

Minimize $v_{within}(t)$ → Maximize $v_{between}(t)$, where $v_{between}(t) = w_0(t)\big(1-w_0(t)\big)\big(\mu_0(t)-\mu_1(t)\big)^2, t \in \{0,1,\ldots,L-1\}$

Otsu's Binarization Algorithm

- Otsu's algorithm in different form:

$$T = \underset{t \in \{0,1,\ldots,L-1\}}{\operatorname{argmax}} \; v_{between}(t),$$

where $v_{between}(t) = w_0(t)\big(1 - w_0(t)\big)\big(\mu_0(t) - \mu_1(t)\big)^2$

- Recurrence relation (time complexity: $\theta(L)$)

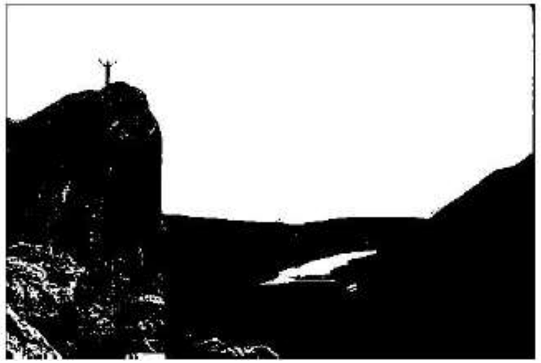
- Initialize at t = 0: $w_0(0) = \hat{h}(0), \mu_0(0) = 0$
- Step through all t > 0:

$$w_0(t) = w_0(t-1) + \hat{h}(t)$$

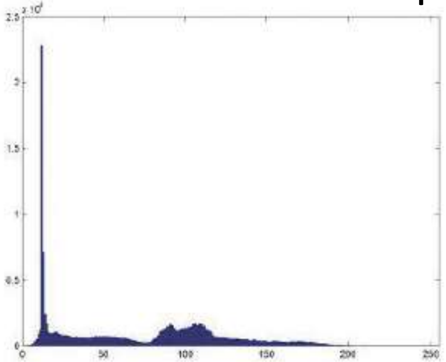
$$\mu_0(t) = \frac{w_0(t-1)\mu_0(t-1) + t\hat{h}(t)}{w_0(t)}$$

$$\mu_1(t) = \frac{\mu - w_0(t)\mu_0(t)}{(1 - w_0(t))}$$

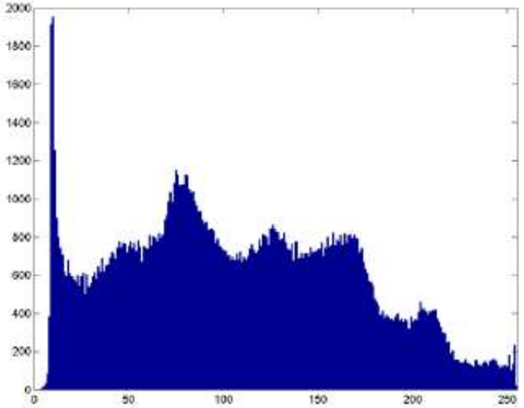
Otsu’s Binarization Algorithm



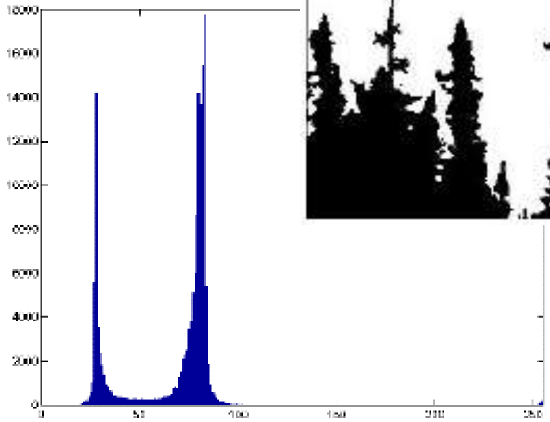
T=70



T=111



T=54

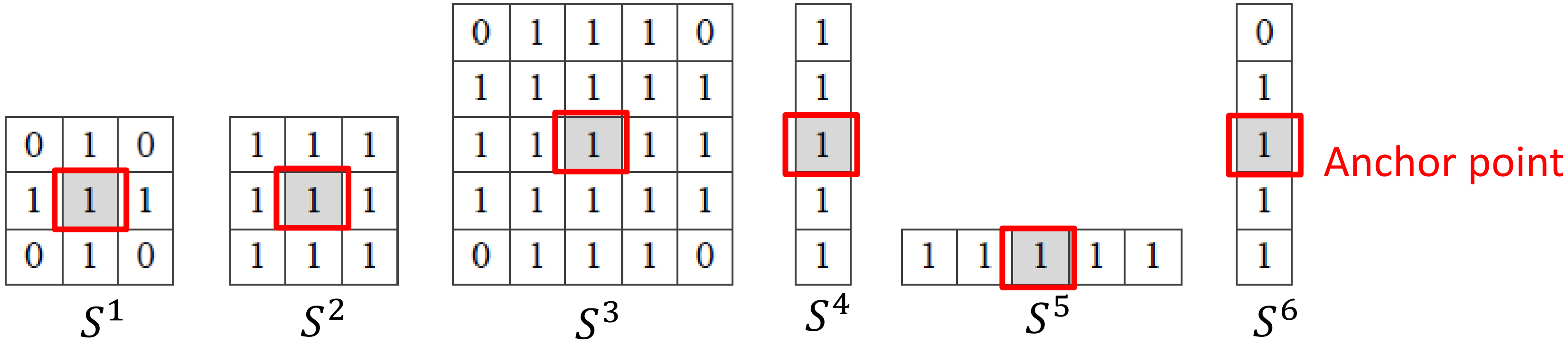


Morphology

- Change the shape of the underlying binary image objects
- Can be applied to both binary and grayscale images, but commonly used in binary images
  - e.g. remove noises in binary images
- Morphological operations:
  1. Define a structuring element
  2. Convolve the structuring element with the image
  3. Thresholding the result of the convolution

Structuring Element

- Can be of any shape
- Represented as a set of non-zero pixel elements



S^1 = {(0,0), (0,1), (-1,0), (0,-1),(1,0)}

$S^2 = \{(0,0), (0,1)(-1,1), (-1,0), (-1,-1), (0,-1), (1,-1), (1,0), (1,1)\}$

- Define $S_t = \{s + t | s \in S\}$
  - E.g. $t = (2,3) \Rightarrow S_t^1 = \{(2,3), (2,4), (1,3), (2,2), (3,3)\}$

Morphological Operations

(a) (b) (c) (d) (e) (f)

Figure 3.22 *Binary image morphology: (a) original image; (b) dilation; (c) erosion; (d) majority; (e) opening; (f) closing. The structuring element for all examples is a 5 × 5 square. The effects of majority are a subtle rounding of sharp corners. Opening fails to eliminate the dot, as it is not wide enough.*

(Binary) Dilation and Erosion

$$S_{\boldsymbol{t}} = \{\boldsymbol{s} + \boldsymbol{t} | \boldsymbol{s} \in S\}$$

In a binary image f,

- Dilation
  - Place a structuring element where $f(j, i) = 1$
  - Change $f(j + k, i + l)$ to 1 if (k, l) is in the structuring element:

$$f \oplus S = \bigcup_{\boldsymbol{x} \in f} S_{\boldsymbol{x}}$$

→ Grow(thicken) the shape by the structuring element

- Erosion
  - Place a structuring element at $f(j, i)$
  - Set $f(j, i)$ to 1 only if $f(j + k, i + l)$=1 for all (k, l) in the structuring element:

$$f \ominus S = \{\boldsymbol{x} | \boldsymbol{x} + \boldsymbol{s} \in f, \forall \boldsymbol{s} \in \mathrm{S}\}$$

→ Shrink the shape by the structuring element

(Binary) Opening and Closing

In a binary image f,

- Opening: $f \circ S = (f \ominus S) \oplus S$
- Closing: $f \cdot S = (f \oplus S) \ominus S$
- Leave large regions and smooth boundaries while removing small holes and noises

(Example) Morphological Operations: Dilation

$f =$

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

1	1	1

Structuring element:
$S = \{(0,-1),(0,0),(0,1)\}$

Dilation at (1,1)

1	1	1

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
1	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

Dilation Result

0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0

(Example) Morphological Operations: Erosion

$f =$

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

1	1	1

Structuring element:
$S = \{(0,-1),(0,0),(0,1)\}$

Erosion at (1,1) and (3,2)

1	1	1

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

Erosion Result

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	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

(Example) Morphological Operations: Opening and Closing

$f =$

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

1	1	1

Structuring element:
$S = \{(0,-1),(0,0),(0,1)\}$

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	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

Erosion

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	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

Opening

0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0

Dialation

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	1	0	0	0	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	0

Closing

Grayscale Morphological Operations (SKIP)

- Structuring Element

0	1	0
1	2	1
0	1	0

	1	
1	2	1
	1	

Nonflat structuring elements

0	0	0
0	0	0
0	0	0

	0	
0	0	0
	0	

Flat structuring elements

- Morphological operations
  - Binary morphological operations can be extended to grayscale images through use of min and max functions
  - Regard a grayscale image as a heightmap
  - min and max function fill the valleys or erode the peaks of a heightmap
  - Used in
    - Contrast enhancement
    - Texture description
    - Edge detection
    - Thresholding

Grayscale Morphological Operations (SKIP)

- Structuring Element

0	1	0
1	2	1
0	1	0

	1	
1	2	1
	1	

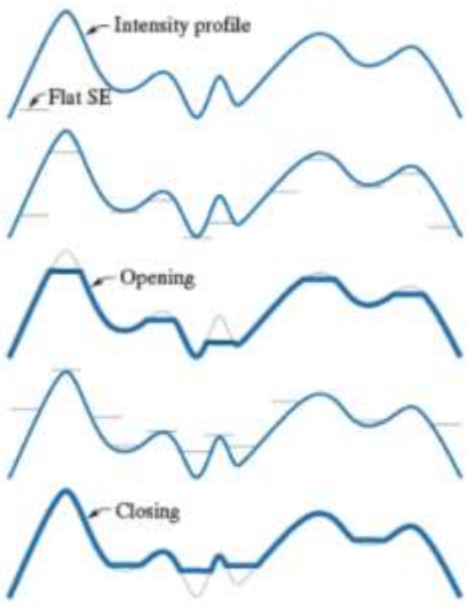
Nonflat structuring elements

0	0	0
0	0	0
0	0	0

	0	
0	0	0
	0	

Flat structuring elements

- Morphological operations
  - For nonflat structuring elements
  - Dilation: $(f \oplus S)(j,i) = \max_{(y,x)\in S} (f(j-y, i-x) + S(y,x))$
  - Erosion: $(f \ominus S)(j,i) = \min_{(y,x)\in S} (f(j+y, i+x) - S(y,x))$
  - For flat structuring elements
  - Dilation: $(f \oplus S)(j,i) = \max_{(y,x)\in S} f(j-y, i-x)$
  - Erosion: $(f \ominus S)(j,i) = \min_{(y,x)\in S} f(j+y, i+x)$
  - Opening and closing
  - Opening: $f \circ S = (f \ominus S) \oplus S$
  - Closing: $f \cdot S = (f \oplus S) \ominus S$
  - Grayscale opening: remove small and bright details



(Example) Grayscale Morphology (SKIP)

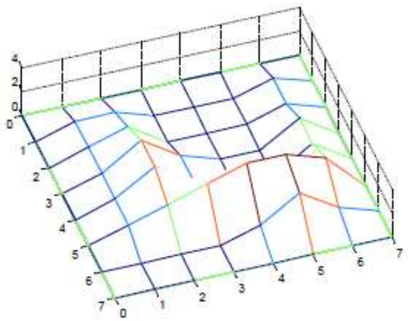
$f =$

0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0
0	1	2	0	0	0	1	0
0	1	3	1	0	0	2	0
0	1	3	1	0	0	2	0
0	1	2	3	4	4	3	0
0	0	0	0	1	3	1	0
0	0	0	0	0	0	0	0

Structuring element S

0	0	0

Height map of f



0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1
1	2	2	1	0	1	1	1
1	3	3	3	1	2	2	2
1	3	3	3	1	2	2	2
1	2	3	4	4	4	4	3
0	0	0	1	3	3	3	1
0	0	0	0	0	0	0	0

Grayscale dilation

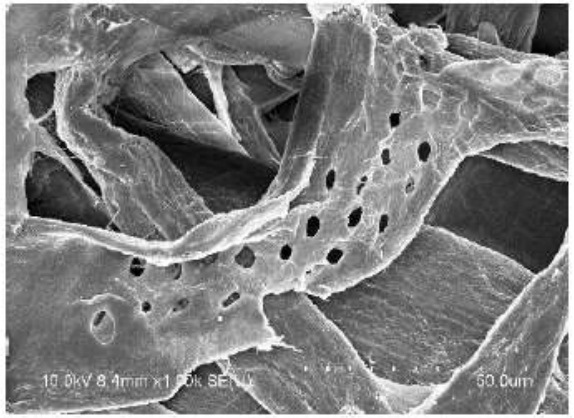
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	2	3	3	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

Grayscale erosion

$$(f \oplus S)(1,0) = \max\big(f(1,-1), f(1,0), f(1,1)\big) = \max(-\infty, 0,1) = 1$$
$$(f \ominus S)(1,0) = \min\big(f(1,-1), f(1,0), f(1,1)\big) = \min(\infty, 0,1) = 0$$

Grayscale Morphology (SKIP)

$f =$

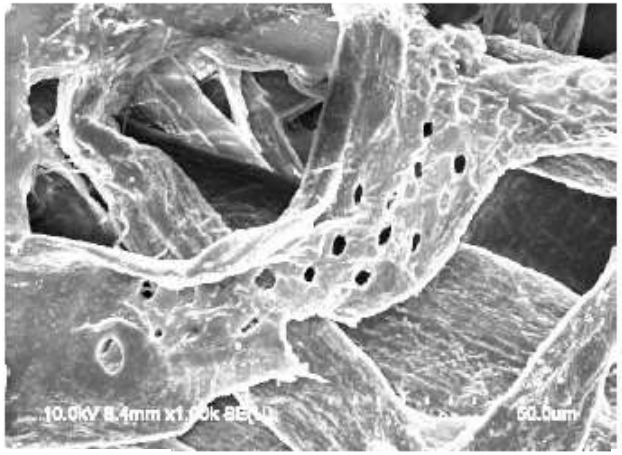


$S=$

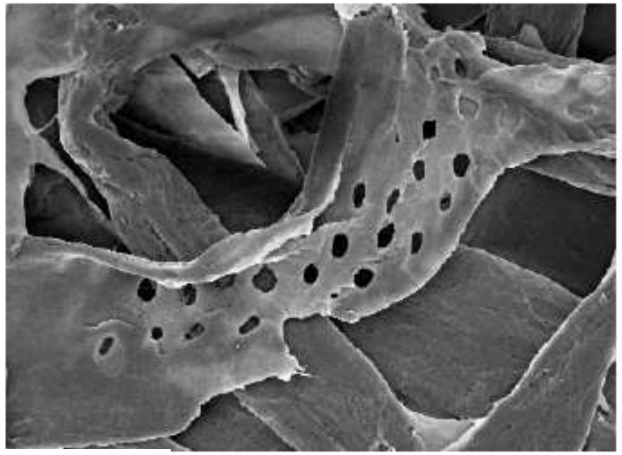
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

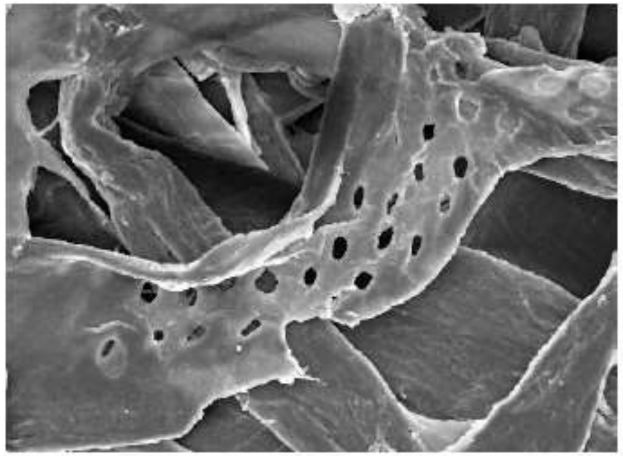
Original image and
Structuring element

Dilation

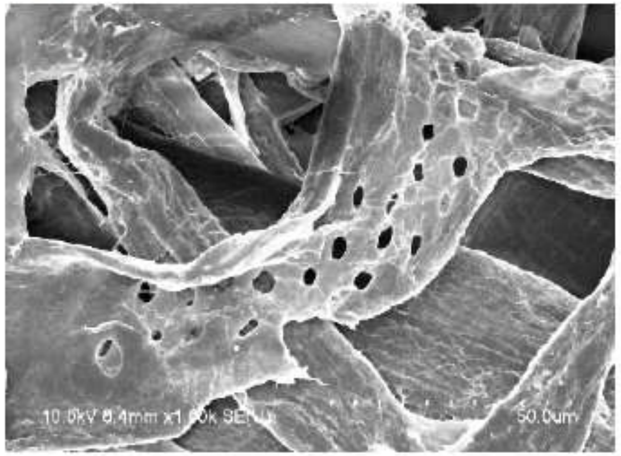


Erosion





Opening



Closing

Connected Component

- Pixel adjacency: define which neighboring pixels are adjacent to the given pixel
  - $\mathcal{N}_4$-adjacency ($\mathcal{N}_4$-connectivity): a pixel has 4 neighbors (N,S,E,W)
  - $\mathcal{N}_8$-adjacency ($\mathcal{N}_8$-connectivity): a pixel has 8 neighbors (N,S,E,W, NE,NW,SE,SW)

NW	N	NE
W	(j, i)	E
SW	S	SE

Connectivity of a pixel

NW	N	NE
W	(j, i)	E
SW	S	SE

$\mathcal{N}_4$-connectivity

NW	N	NE
W	(j, i)	E
SW	S	SE

$\mathcal{N}_8$-connectivity

Connected Component

- Connected component – a region of all adjacent pixels
- Label each connected component with different numbers

0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	1	0	1	1	0
0	1	0	1	0	1	1	0	1	0
0	1	0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	0	1	0
0	1	1	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0

Original binary image

0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	2	2	0	0	1	0	3	3	0
0	2	0	4	0	1	1	0	3	0
0	2	0	4	0	1	0	0	3	0
0	2	0	4	0	1	0	0	3	0
0	2	2	0	0	1	0	0	3	0
0	0	0	0	0	0	0	0	0	0

Label CC based on $\mathcal{N}_4$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	2	2	0	0	1	0	1	1	0
0	2	0	2	0	1	1	0	1	0
0	2	0	2	0	1	0	0	1	0
0	2	0	2	0	1	0	0	1	0
0	2	2	0	0	1	0	0	1	0
0	0	0	0	0	0	0	0	0	0

Label CC based on $\mathcal{N}_8$

Connected Component Labeling

```
Algorithm. Flood-fill algorithm to label connected components
Input: Binary image b(j,i), 0 ≤ j ≤ M − 1, 0 ≤ i ≤ N − 1
Output: Labelled image l(j,i), 0 ≤ j ≤ M − 1, 0 ≤ i ≤ N − 1

Initialize l: l(j,i) = 0 if b(j,i) = 0, l(j,i)=-1 if b(j,i)=1
l(0,:) = l(M-1, :) = l(:,0) = l(:, N-1) = 0
label := 1;
for (j=1 to M-2)
   for (i=1 to N-2) {
      if (l(j,i) == -1) {
          flood_fill4(l, j, I, label);
          label++;
      }
   }

function flood_fill4(l,j,l,label) {
   if(l(j,i)==-1) {
      l(j,i)=label;
      flood_fill4(l,j,i+1, label);
      flood_fill4(l,j-1,i, label);
      flood_fill4(l,j,i-1, label);
      flood_fill4(l,j+1,I,label);
   }
}
```

- Use flood fill algorithm to label connected components
- Pick one unvisited pixel and assign the current label to every unvisited pixel that can be connected via $\mathcal{N}_4$-connectivity
- Unable to control the depth of recursive function calls

$\Rightarrow$ Stack overflow can occur

Connected Component Labeling

```
Algorithm. Flood-fill algorithm to label connected
components
Input: Binary image b(j,i), 0 ≤ j ≤ M − 1, 0 ≤ i ≤ N − 1
Output: Labelled image l(j,i), 0 ≤ j ≤ M − 1, 0 ≤ i ≤ N − 1

Initialize l: l(j,i) = 0 if b(j,i) = 0, l(j,i)=-1 if
b(j,i)=1
l(0,:) = l(M-1, :) = l(:,0) = l(:, N-1) = 0
label := 1;
for (j=1 to M-2)
   for (i=1 to N-2) {
      if (l(j,i) == -1) {
         efficient_flood_fill4(l, j, I, label);
         label++;
      }
   }
```

```
function efficient_flood_fill4(l,j,l,label) {
   Q := ∅;
   push(Q(j,i));
   while (Q ≠ ∅) {
      (y,x) = pop(Q);
      if(l(j,i)==-1) {
         left=right=x;
         while(l(y,left-1)==-1) left--;
         while(l(y,right+1)==-1) right++;
         for(c=left to right) {
            l(y,c)=label;
            if(l(y-1,c)==-1 and (c=left or l(y-1,c-1)≠-1)
               push(Q,(y-1,c));
            if(l(y+1,c)==-1 and (c=left or l(y+1,c-1) ≠-1)
               push(Q,(y+1,c));
         }
      }
   }
}
```

- Queue-based algorithm
- Put unvisited pixel to a queue
- For each unvisited pixel, find the consecutive row of unvisited pixels and label them together

Content

- Area-based Operation (cont’d)
- Binary Image Processing
- **Geometric Transformation**
- Image Pyramid

Homogeneous Coordinate System

- Homogeneous coordinate: represent a 2D point $\boldsymbol{x} = (y, x)$ as a 3D vector $\dot{\boldsymbol{x}}$:

$$\dot{\boldsymbol{x}} = (y, x, 1) = (hy, hx, h)$$

- E.g. $\dot{\boldsymbol{x}}$=(3,5,1), (6,10,2), (1.5, 2.5, 0.5) represent the same 2D point (3,5)

Homogeneous Matrix

- With homogeneous coordinates, 2D geometric transformation can be expressed with a 3x3 homogeneous matrix

Transformation	Homogeneous Matrix $\dot{H}$	Geometric Meaning
Translation	$T(t_y, t_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_y & t_x & 1 \end{bmatrix}$	Translate by t_y in y direction and t_x in x direction
Rotation	$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Rotation clockwise by θ
Scale	$S(s_y, s_x) = \begin{bmatrix} s_y & 0 & 0 \\ 0 & s_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Scale by s_y in y direction and s_x in x direction
Shear	$Sh_y(h_y) = \begin{bmatrix} 1 & 0 & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, Sh_x(h_x) = \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	Sh_y : Shear by h_y in y direction Sh_x : Shear by h_x in x direction

Homogeneous Matrix

- A point x moves to x as follows

$$\dot{\boldsymbol{x}}' = \dot{\boldsymbol{x}}\dot{H} = (y\ x\ 1)\begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

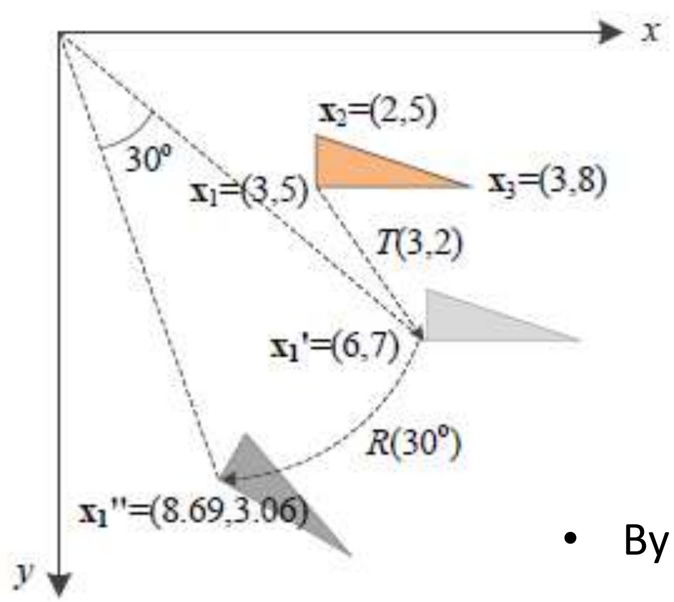
$$\Rightarrow y' = a_{11}y + a_{21}x + a_{31},\ x' = a_{12}y + a_{22}x + a_{32}$$

- E.g. Write a homogeneous matrix for 2D geometric transformation of translating a point by 3 in y direction and by 2 in x direction

$$\dot{H} = T(3,2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$$

Homogeneous Matrix

- E.g. Translate a triangle by (3,2), and rotate it by $30°$



- By applying $T(3,2) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}$, $\dot{\boldsymbol{x}}_{\mathbf{1}}{}' = (3\ 5\ 1)\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} = (6\ 7\ 1)$
- Then apply $R(30°)$ to (6 7 1):

$$\dot{\boldsymbol{x}}_{\mathbf{1}}{}'' = (6\ 7\ 1)\begin{bmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} = (8.696\ 3.062\ 1)$$

$\Rightarrow$ Point (3,5) moves to (8.696 3.062)

OR

- By applying $T(3,2)R(30°) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}\begin{bmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 3.598 & 0.232 & 1 \end{bmatrix}$

  to (3 5 1):

$$\dot{\boldsymbol{x}}_{\mathbf{1}}{}'' = (3\ 5\ 1)\begin{bmatrix} 0.866 & -0.5 & 0 \\ 0.5 & 0.866 & 0 \\ 3.598 & 0.232 & 1 \end{bmatrix} = (8.696\ 3.062\ 1)$$

$\Rightarrow$ Point (3,5) moves to (8.696 3.062)

Homogeneous Matrix

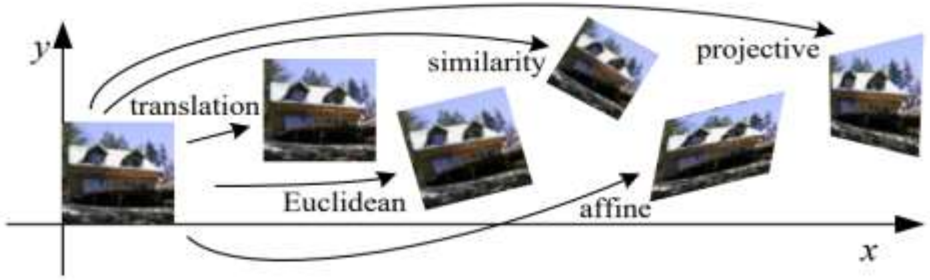
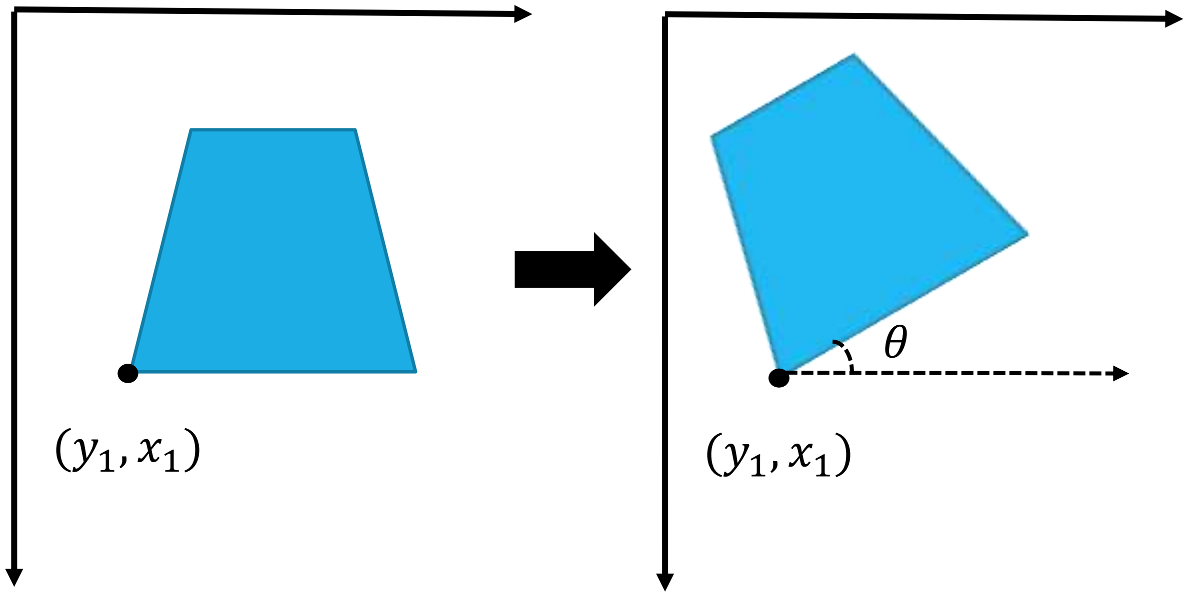


Figure 3.44 *Basic set of 2D geometric image transformations.*

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\left(\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2\times 3}\right)^T$	2	orientation	
rigid (Euclidean)	$\left(\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}\right)^T$	3	lengths	
similarity	$\left(\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2\times 3}\right)^T$	4	angles	
affine	$\left(\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2\times 3}\right)^T$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

Geometric Transformation

- Write a homogeneous matrix to represent the following geometric transformation



- Step 1: Translate the quadrilateral by $(-y_1, -x_1)$
- Step 2: Rotate the quad by θ
- Step 3: Translate the quadrilateral by (y_1, x_1)

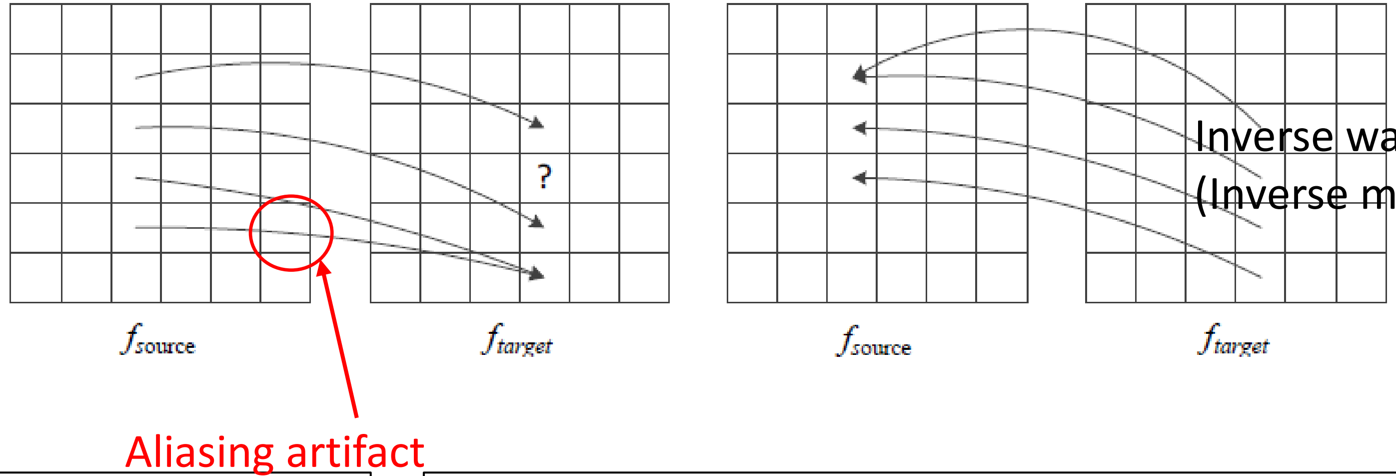
$\Rightarrow \dot{H} = T(-y_1, -x_1)R(\theta)T(y_1, x_1)$

$\Rightarrow \dot{\boldsymbol{x}}' = \dot{\boldsymbol{x}}\dot{H} = (y\ x\ 1)\ T(-y_1, -x_1)R(\theta)T(y_1, x_1)$

Geometric Transformation

- How to apply a geometric transformation to an image?

Forward warping
(Forward mapping)



Inverse warping
(Inverse mapping)

Alg. Forward warping
Input: $f_{src}(j,i), 0 \le j \le M-1, 0 \le i \le N-1$, $\dot{H}$
Output: $f_{tgt}(j,i), 0 \le j \le M-1, 0 \le i \le N-1$

for (j=0 to M-1)
  for (i=0 to N-1) {
    (j', i') = Apply $\dot{H}$ to (j,i) (rounding included)
    $f_{tgt}(j',i') = f_{src}(j,i)$
  }

Alg. Inverse warping
Input: $f_{src}(j,i), 0 \le j \le M-1, 0 \le i \le N-1$, $\dot{H}$
Output: $f_{tgt}(j,i), 0 \le j \le M-1, 0 \le i \le N-1$

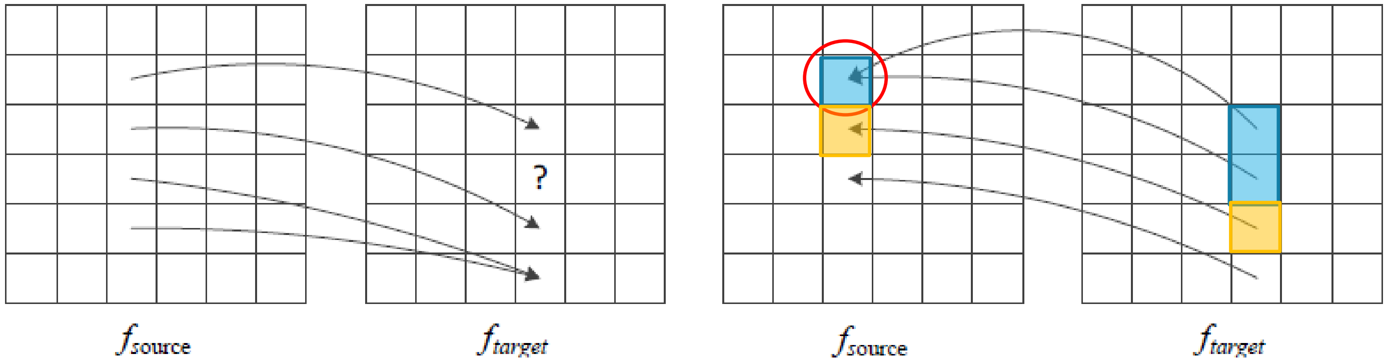
for (j=0 to M-1)
  for (i=0 to N-1) {
    (j', i') = Apply $\dot{\boldsymbol{H}}^{-1}$ to (j,i) (rounding included)
    $f_{tgt}(j,i) = f_{src}(j',i')$
  }

Geometric Transformation

- Aliasing also occurs in inverse warping

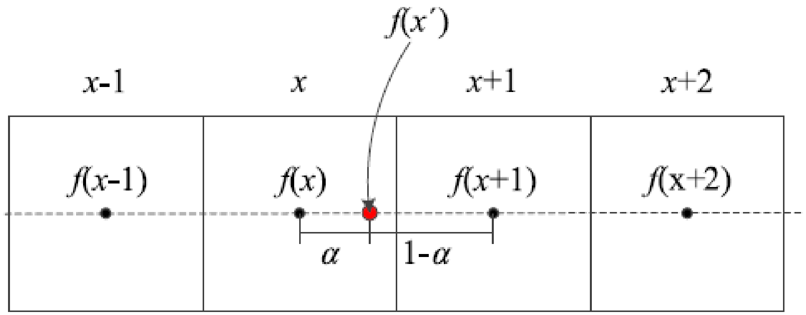
⇒ Two different target pixels can refer to the same source pixel

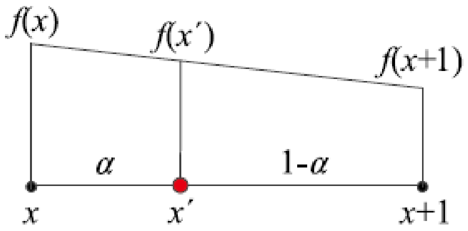
⇒ For anti-aliasing, we interpolate the neighboring pixel values



Interpolation

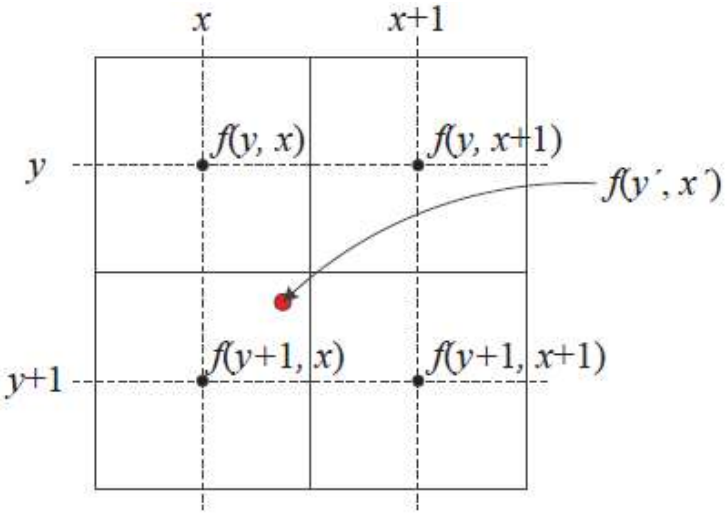
- Linear interpolation in 1D

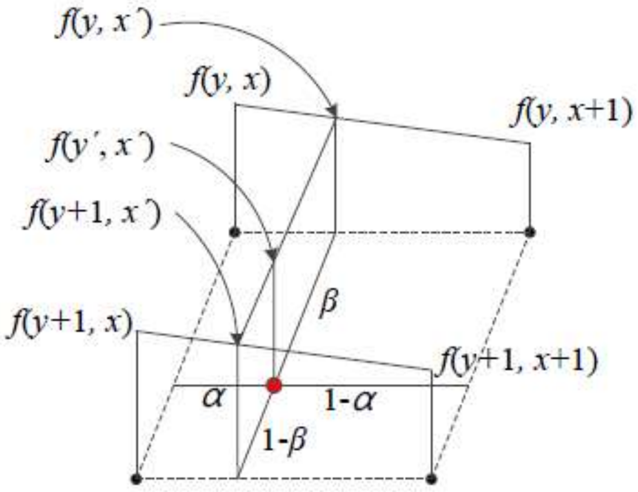




$$f(x') = (1-\alpha)f(x) + \alpha f(x+1)$$

- Bilinear interpolation in 2D





$$
\begin{aligned}
f(y,x') &= (1-\alpha)f(y,x) + \alpha f(y,x+1) \\
f(y+1,x') &= (1-\alpha)f(y+1,x) + \alpha f(y+1,x+1) \\
f(y',x') & \\
&= (1-\beta)f(y,x') + \beta f(y+1,x') \\
&= (1-\beta)(1-\alpha)f(y,x) + (1-\beta)\alpha f(y,x+1) \\
&\quad +\beta(1-\alpha)f(y+1,x) + \beta\alpha f(y+1,x+1)
\end{aligned}
$$

Geometric Transformation

Original Image

Nearest Neighbor

Bilinear Interpolation

Bicubic Interpolation

Content

- Area-based Operation (cont’d)
- Binary Image Processing
- Geometric Transformation
- **Multi-resolution Representations**

Multiresolution Images

- Input and output images of different sizes
- Do not know the appropriate resolution of an image

$\Rightarrow$ A *pyramid* of different-sized images might be useful

- Upsampling: increase resolution by 2 times
- Downsampling: decrease resolution by ½ times

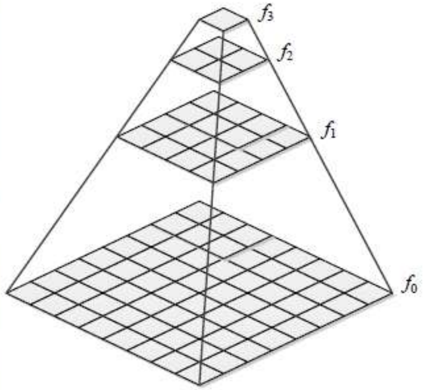


Image pyramid

Downsampling

- Reduce the image resolution by ½

$$f_k(j,i) = f_{k-1}\left(\frac{j}{r},\frac{i}{r}\right), r = \frac{1}{2}, 1 \le k \le q$$

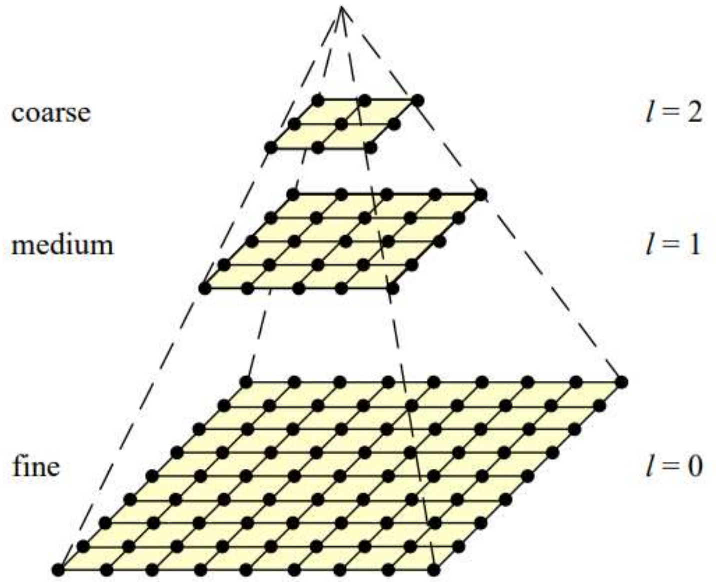
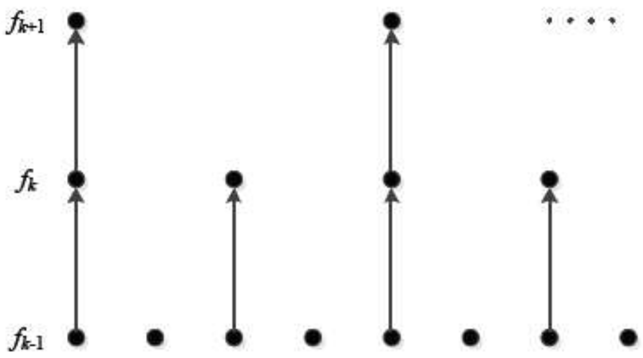


Figure 3.31 *A traditional image pyramid: each level has half the resolution (width and height), and hence a quarter of the pixels, of its parent level.*

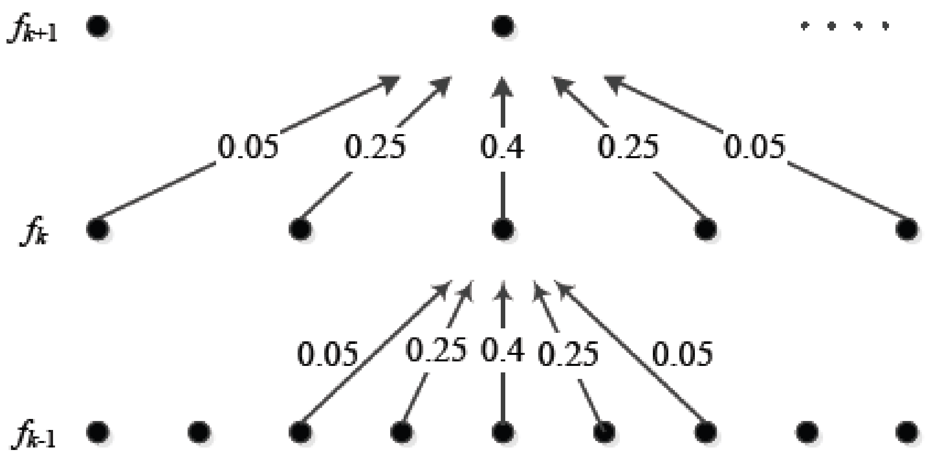


$\Rightarrow$ Aliasing occurs as each pixel in f_{k-1} contributes to f_k either by 100% or 0%

Downsampling

- To reduce aliasing, apply smoothing to an image before downsampling [Burt and Adelson’s(1983)]

$$f_k(j,i) = \sum_{y=-2}^{y=2}\sum_{x=-2}^{x=2} w(y,x) f_{k-1}\left(\frac{j}{r}+y, \frac{i}{r}+x\right), r = \frac{1}{2}, 1 \le k \le q$$



- W(y,x) is 1x5 filter (1/20, 1/4, 2/5, 1/4, 1/20)
- Weights sums up to 1
- All pixels in f_{k-1} contributes to f_k by 50%

Downsampling

- Downsampling in 2D

$v =$

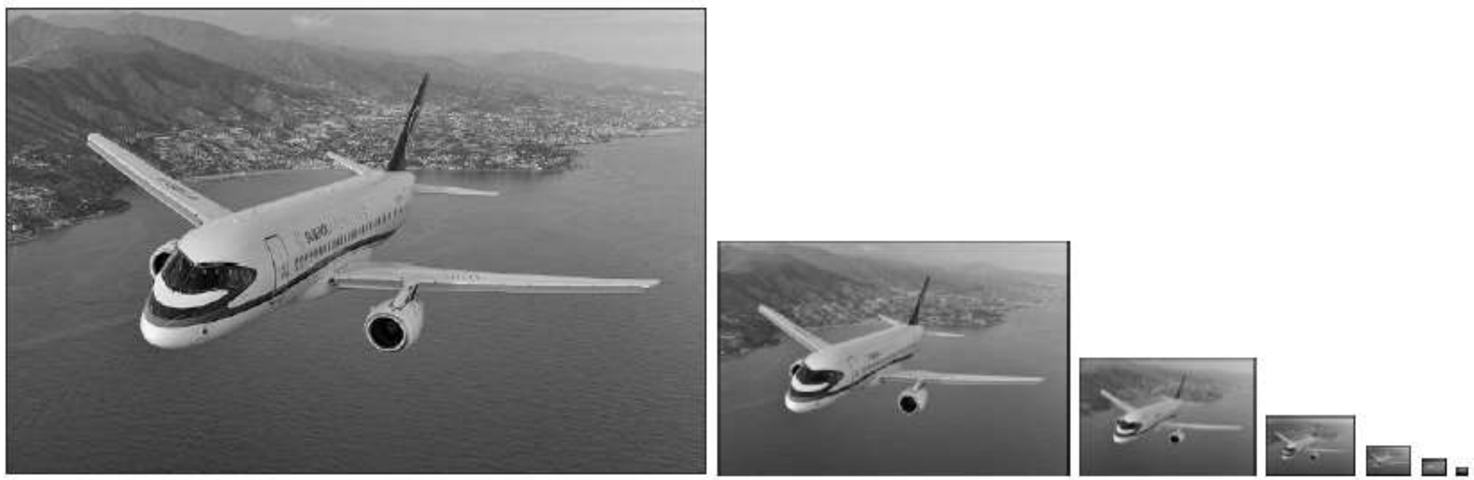
0.05
0.25
0.4
0.25
0.05

$h =$

0.05	0.25	0.4	0.25	0.05

$w =$

.0025	.0125	.0200	.0125	.0025
.0125	.0625	.1000	.0625	.0125
.0200	.1000	.1600	.1000	.0200
.0125	.0625	.1000	.0625	.0125
.0025	.0125	.0200	.0125	.0025



Original Image
(764 x 1024)