

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

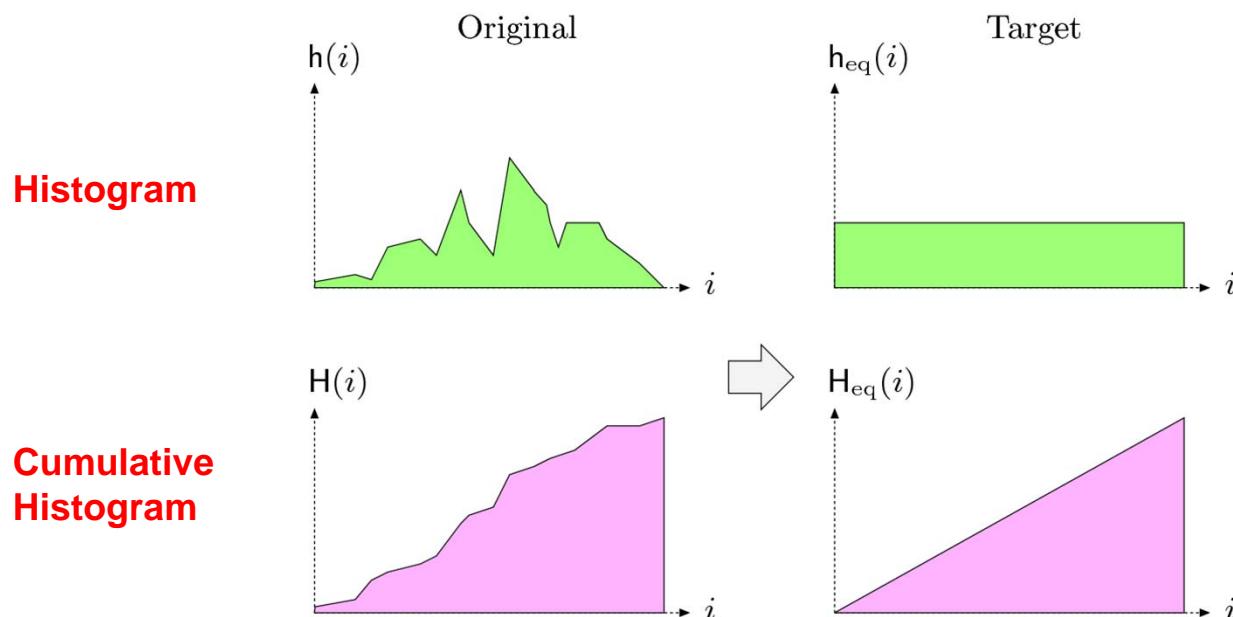
The Chinese University of Hong Kong, Shen Zhen

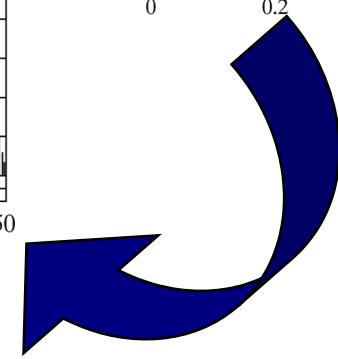
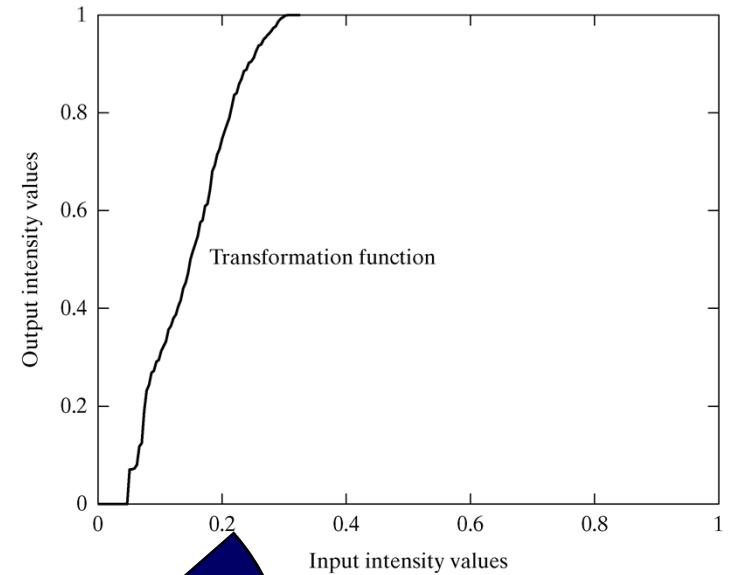
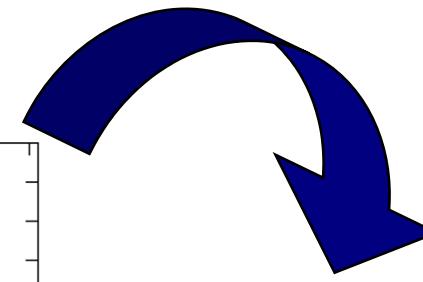
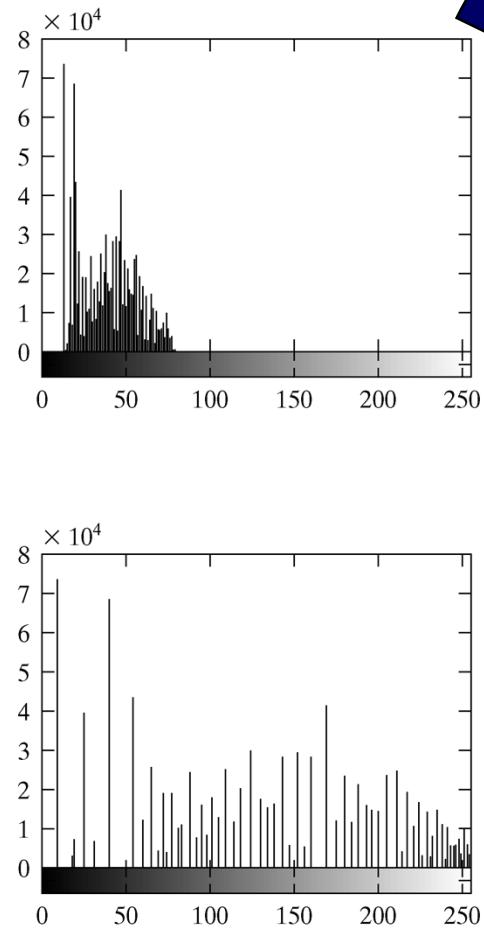
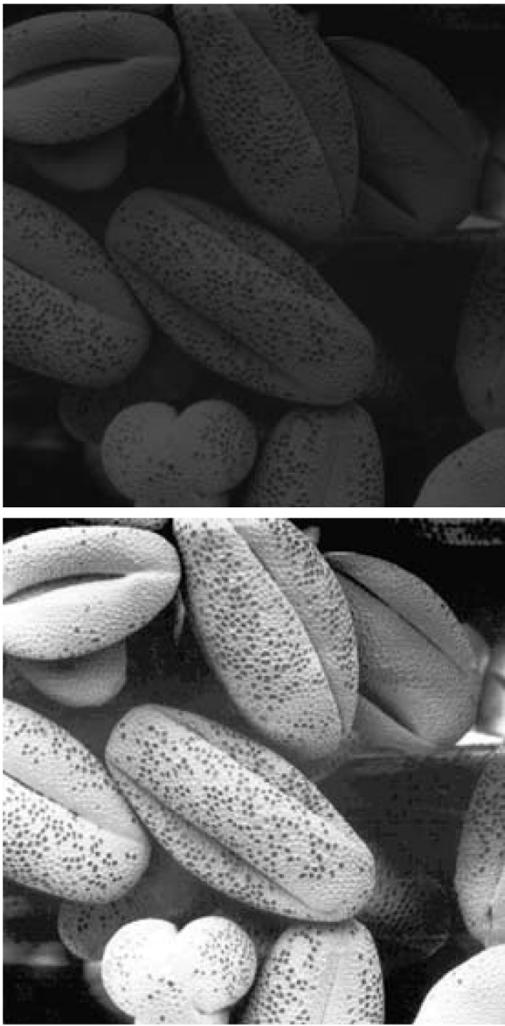
Januray 22 - 24, 2019



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



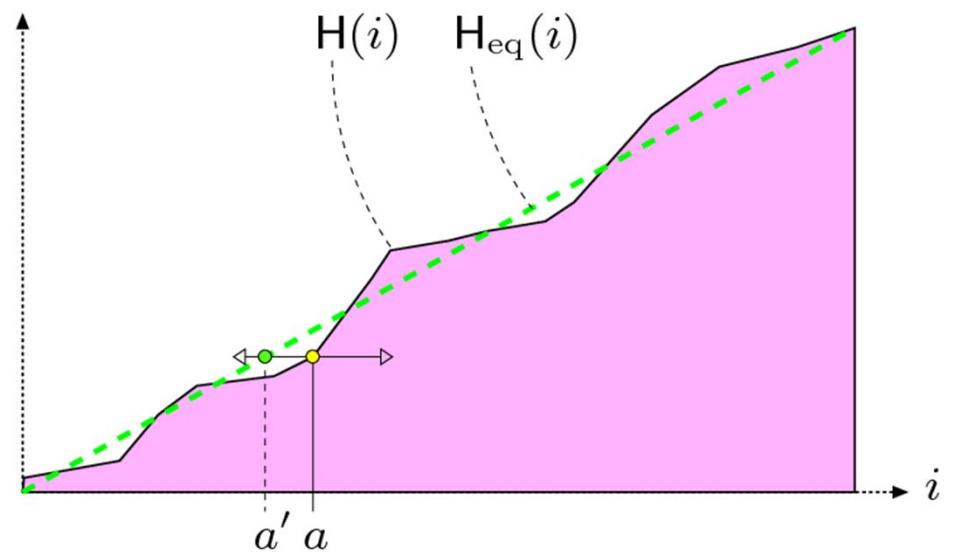
Linear Histogram Equalization

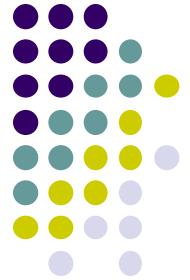
- Histogram cannot be made exactly flat – peaks cannot be increased or decreased by point operations.
- Following point operation makes histogram as flat as possible:
(assuming $M \times N$ image and pixels in range $[0, K - 1]$)

Point operation that returns
Linear equalized value of a

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

Cumulative Histogram: Σ how many
times intensity a occurs





Effects of Linear Histogram Equalization

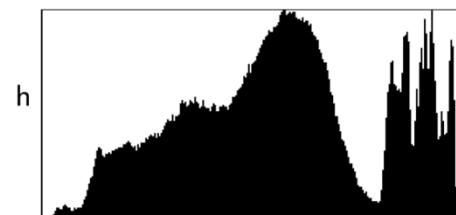
Original Image I



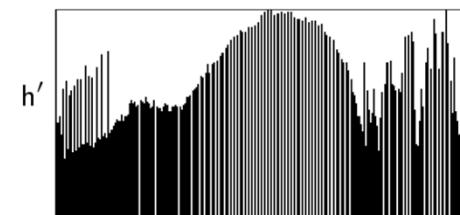
(a)

(b)

Original
histogram



(c)

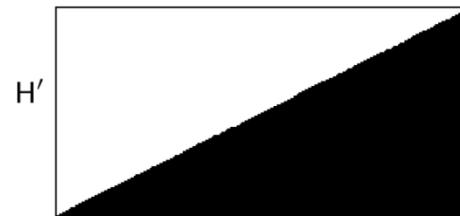


(d)

Cumulative
Histogram



(e)

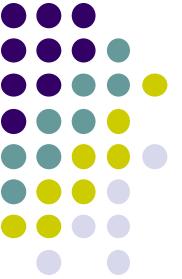


(f)

Image I' after Linear
Equalization

Histogram after
Linear Equalization

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*

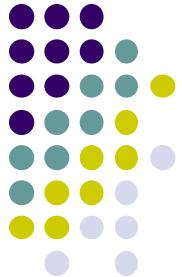
Compute cumulative histogram in place

Get intensity value at (u,v)

Equalize pixel intensity

Cumulative Histogram: Σ how many times intensity *a* occurs

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



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

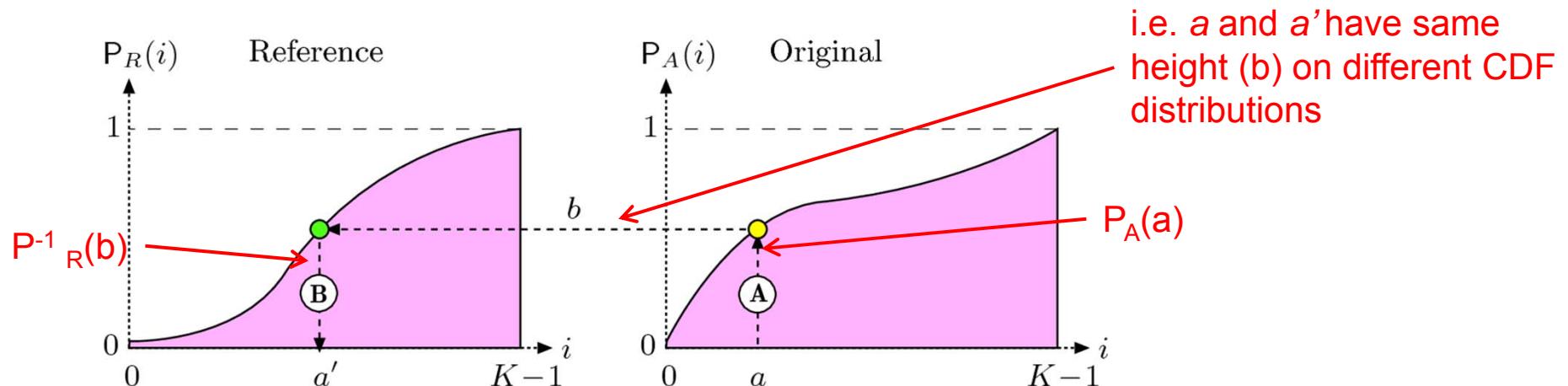
- Find a mapping such that distribution of a matches some reference distribution.i.e

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

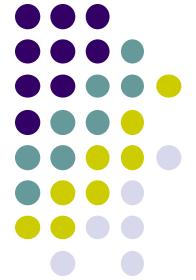
Mapping function: maps distribution on right to equivalent point (same height)
On distribution on left

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

$$P_{A'}(i) \approx P_R(i) \quad \text{for } 0 \leq i < K$$



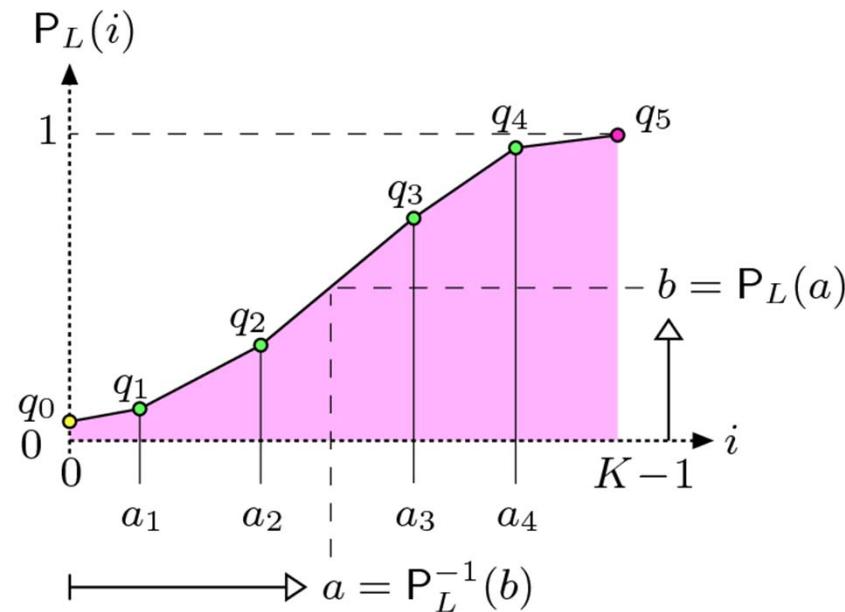
$$f_{hs}(a) = a' = P_R^{-1}(P_A(a))$$



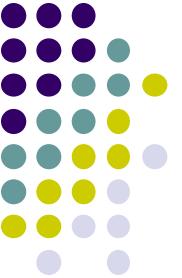
Adjusting Linear Distribution Piecewise

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

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

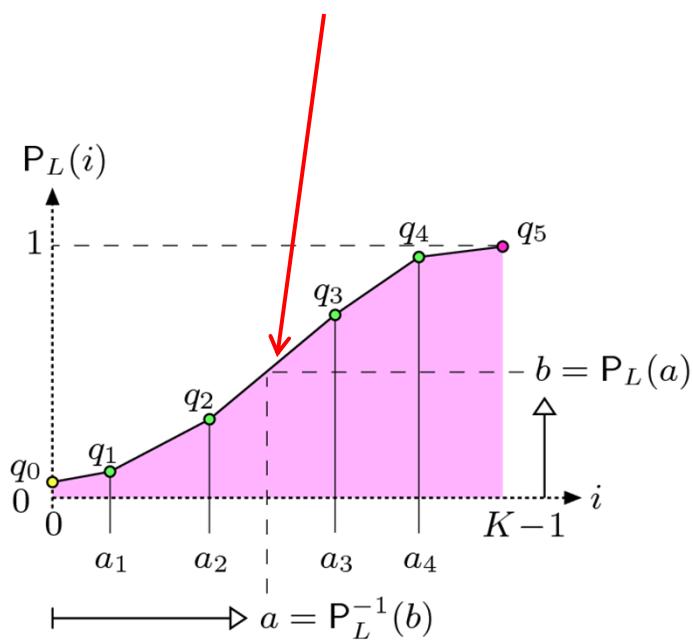


- 2 endpoints are fixed $\langle 0, q_0 \rangle$ and $\langle K-1, 1 \rangle$



Adjusting Linear Distribution Piecewise

For each segment, linearly
Interpolate to find any value



$$P_L(i) = \begin{cases} q_m + (i-a_m) \cdot \frac{(q_{m+1}-q_m)}{(a_{m+1}-a_m)} & \text{for } 0 \leq i < K-1 \\ 1 & \text{for } i = K-1 \end{cases}$$

$$P_L^{-1}(b) = \begin{cases} 0 & \text{for } 0 \leq b < P_L(0) \\ a_n + (b-q_n) \cdot \frac{(a_{n+1}-a_n)}{(q_{n+1}-q_n)} & \text{for } P_L(0) \leq b < 1 \\ K-1 & \text{for } b \geq 1 \end{cases}$$

We also need the inverse mapping

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

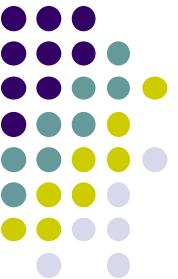


Adjusting Linear Distribution Piecewise

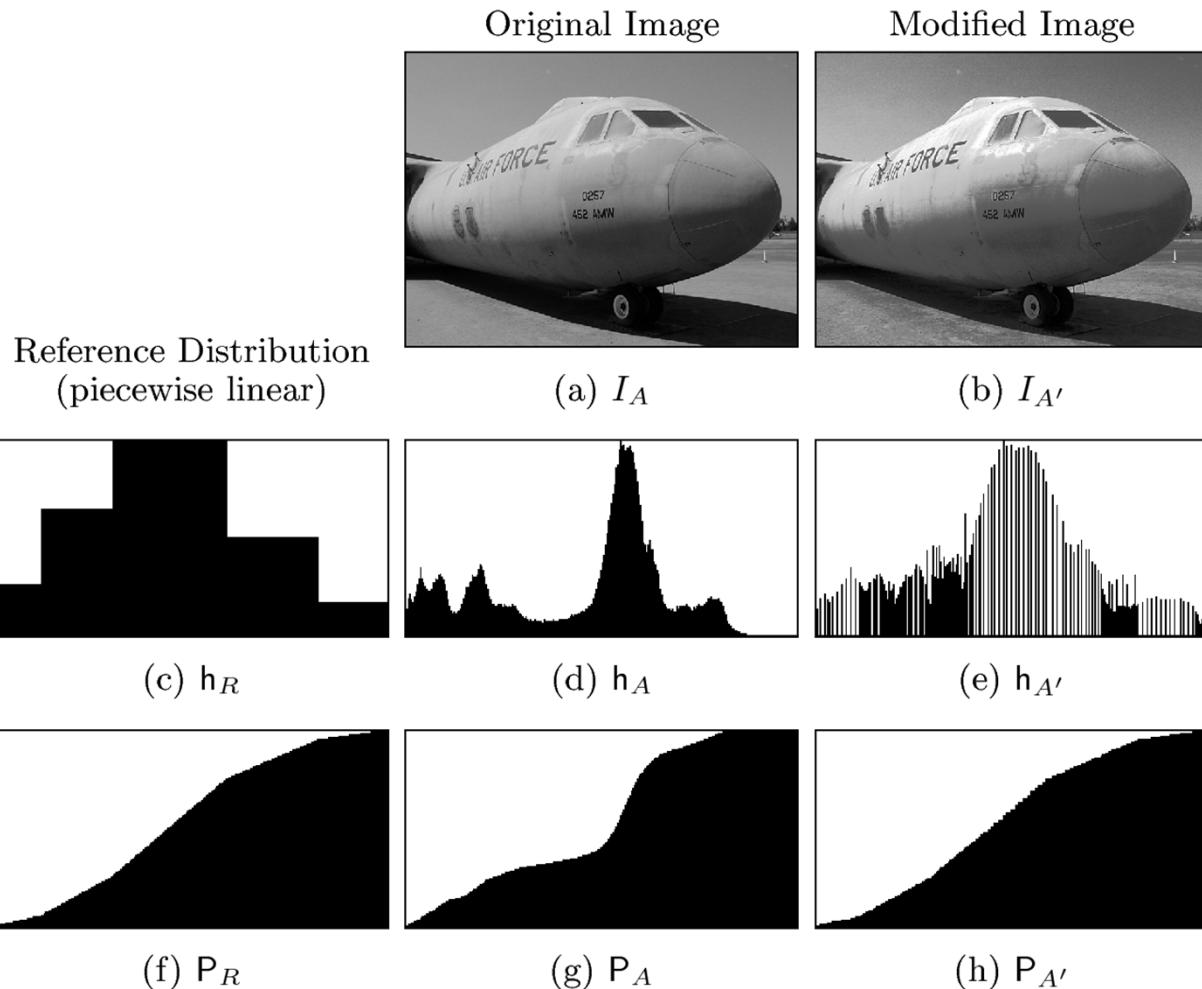
```

1: PIECEWISELINEARHISTOGRAM( $\mathbf{h}_A, \mathcal{L}_R$ )
     $\mathbf{h}_A$ : histogram of the original image.
     $\mathcal{L}_R$ : reference distribution function, given as a sequence of  $N + 1$ 
    control points  $\mathcal{L}_R = [\langle a_0, q_0 \rangle, \langle a_1, q_1 \rangle, \dots \langle a_N, q_N \rangle]$ , with  $0 \leq a_k < K$ 
    and  $0 \leq q_k \leq 1$ .
2: Let  $K \leftarrow \text{Size}(\mathbf{h}_A)$ 
3: Let  $\mathbf{P}_A \leftarrow \text{CDF}(\mathbf{h}_A)$                                  $\triangleright$  cdf for  $\mathbf{h}_A$  (Alg. 5.1)
4: Create a table  $f_{\text{hs}}[ ]$  of size  $K$                            $\triangleright$  mapping function  $f_{\text{hs}}$ 
5: for  $a \leftarrow 0 \dots (K-1)$  do
6:      $b \leftarrow \mathbf{P}_A(a)$ 
7:     if ( $b \leq q_0$ ) then
8:          $a' \leftarrow 0$ 
9:     else if ( $b \geq 1$ ) then
10:         $a' \leftarrow K-1$ 
11:    else
12:         $n \leftarrow N-1$ 
13:        while ( $n \geq 0$ )  $\wedge (q_n > b)$  do       $\triangleright$  find line segment in  $\mathcal{L}_R$ 
14:             $n \leftarrow n - 1$ 
15:             $a' \leftarrow a_n + (b - q_n) \cdot \frac{(a_{n+1} - a_n)}{(q_{n+1} - q_n)}$        $\triangleright$  see Eqn. (5.23)
16:             $f_{\text{hs}}[a] \leftarrow a'$ 
17: return  $f_{\text{hs}}$ .

```



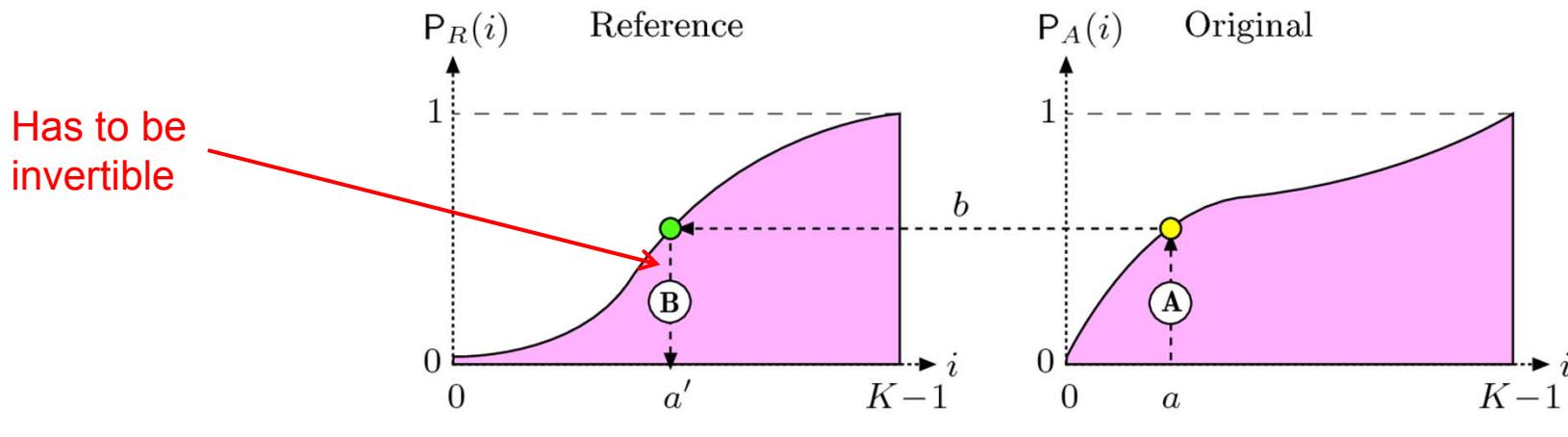
Adjusting Linear Histogram Piecewise





Histogram Matching

- Prior method needed reference distribution to be invertible



$$f_{hs}(a) = a' = P_R^{-1}(P_A(a))$$

- 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**

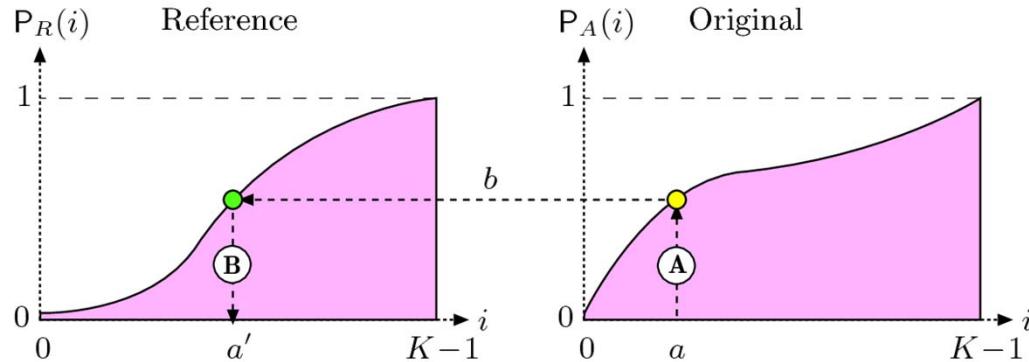


Histogram Matching

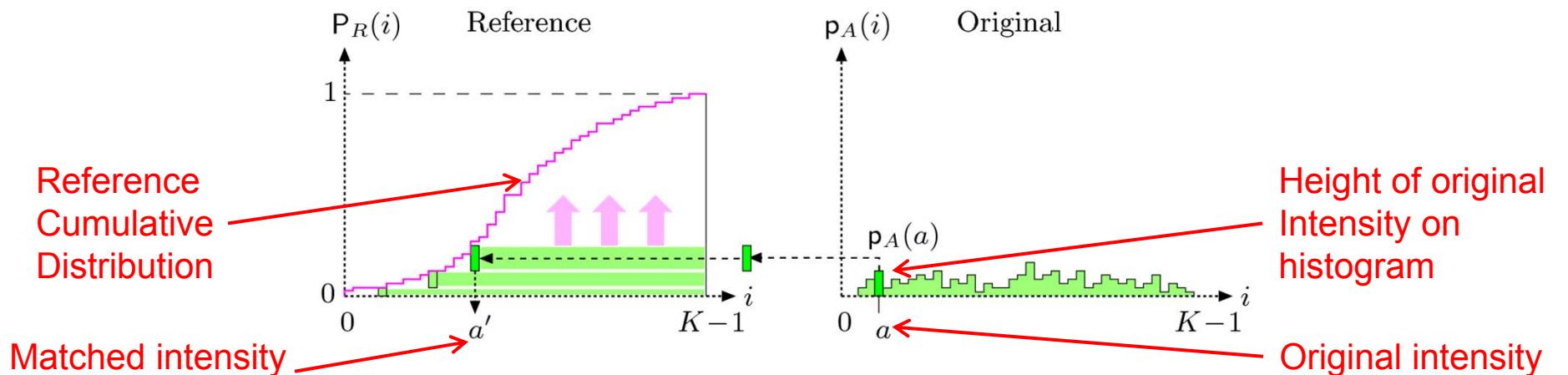
- Given two images I_A and I_B , we want to make their intensity profiles look as similar as possible.
- How?
 - “Match” their cumulative histograms H_A and H_B
- Works well for images with similar content.
- Looks bad for images with different content.



Adjusting to a Given Histogram

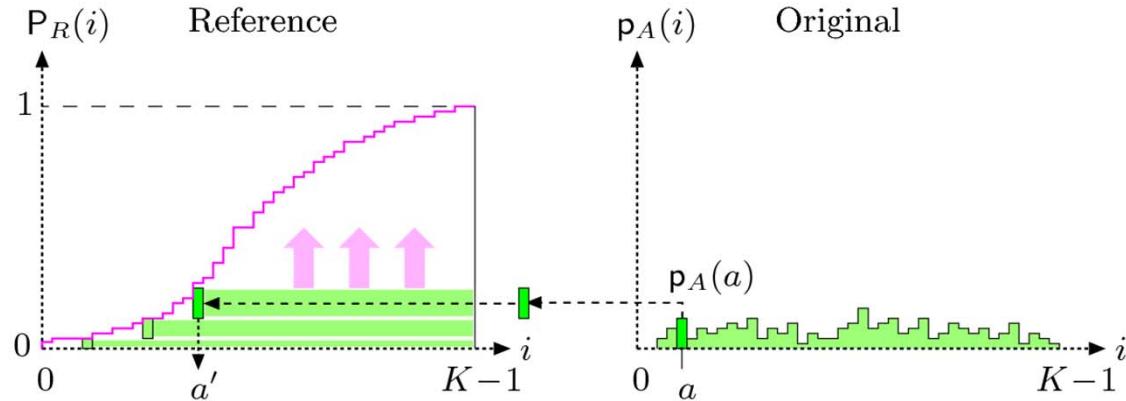


$$f_{hs}(a) = a' = \min \{ j \mid (0 \leq j < K) \wedge (P_A(j) \leq P_R(j)) \}$$





Adjusting to a Given Histogram



1: MATCHHISTOGRAMS($\mathbf{h}_A, \mathbf{h}_R$)

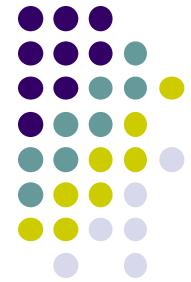
\mathbf{h}_A : histogram of the target image

\mathbf{h}_R : reference histogram (of same size as \mathbf{h}_A)

```

2: Let  $K \leftarrow \text{Size}(\mathbf{h}_A)$ 
3: Let  $\mathbf{P}_A \leftarrow \text{CDF}(\mathbf{h}_A)$                                  $\triangleright$  cdf for  $\mathbf{h}_A$  (Alg. 5.1)
4: Let  $\mathbf{P}_R \leftarrow \text{CDF}(\mathbf{h}_R)$                                  $\triangleright$  cdf for  $\mathbf{h}_R$  (Alg. 5.1)
5: Create a table  $f_{\text{hs}}[ ]$  of size  $K$                        $\triangleright$  pixel mapping function  $f_{\text{hs}}$ 
6: for  $a \leftarrow 0 \dots (K-1)$  do
7:    $j \leftarrow K-1$ 
8:   repeat
9:      $f_{\text{hs}}[a] \leftarrow j$ 
10:     $j \leftarrow j - 1$ 
11:   while ( $j \geq 0$ )  $\wedge (\mathbf{P}_A(a) \leq \mathbf{P}_R(j))$ 
12: return  $f_{\text{hs}}$ .
```

Adjusting to a Given Histogram

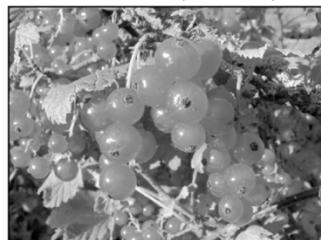


Original Image



(a) I_A

Gaussian ($\sigma = 50$)



(b) I_{G50}

Gaussian ($\sigma = 100$)



(c) I_{G100}

Reference Histogram



$$\mathbf{h}_R(i)$$

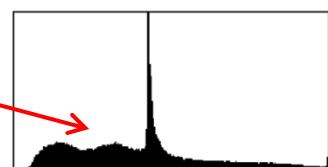
Cumulative Reference Histogram



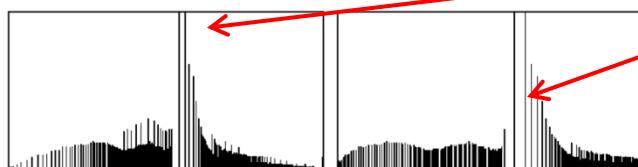
$$\mathsf{H}_R(i)$$

(e)

original histogram



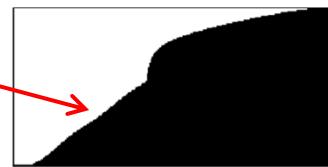
(f) h_A



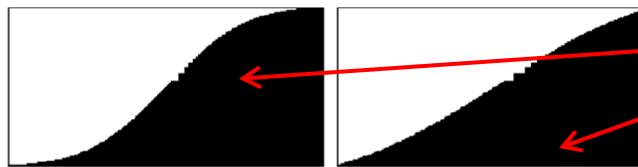
(g) h_{G50}

(h) h_{G10e}

CDF of original histogram



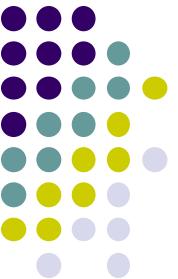
(i) \mathbb{H}_A



(j) H_{G50}

Original histogram after matching

CDF of original
histogram after
matching



Adjusting to a Given Histogram

Target Image



(a) I_A

Reference Image

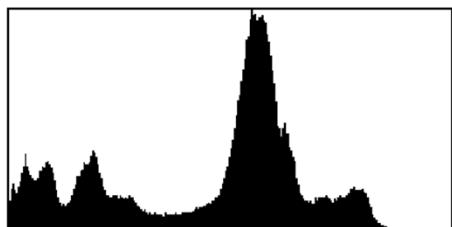


(b) I_R

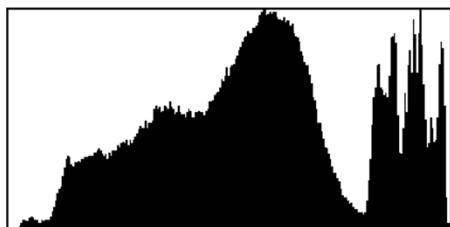
Modified Image



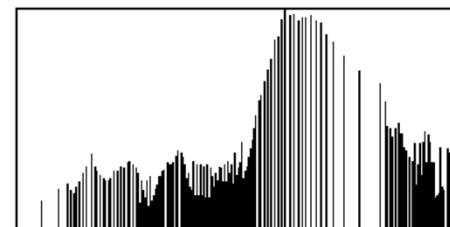
(c) $I_{A'}$



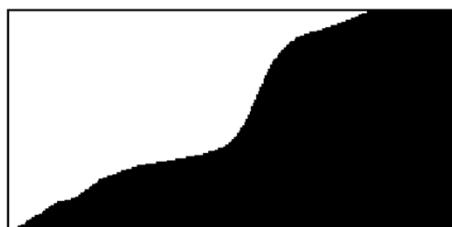
(d) h_A



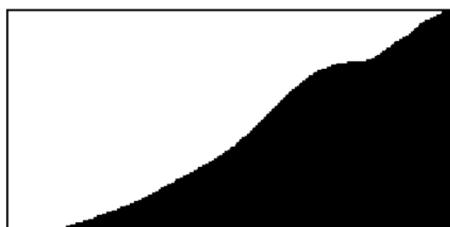
(e) h_R



(f) $h_{A'}$



(g) H_A

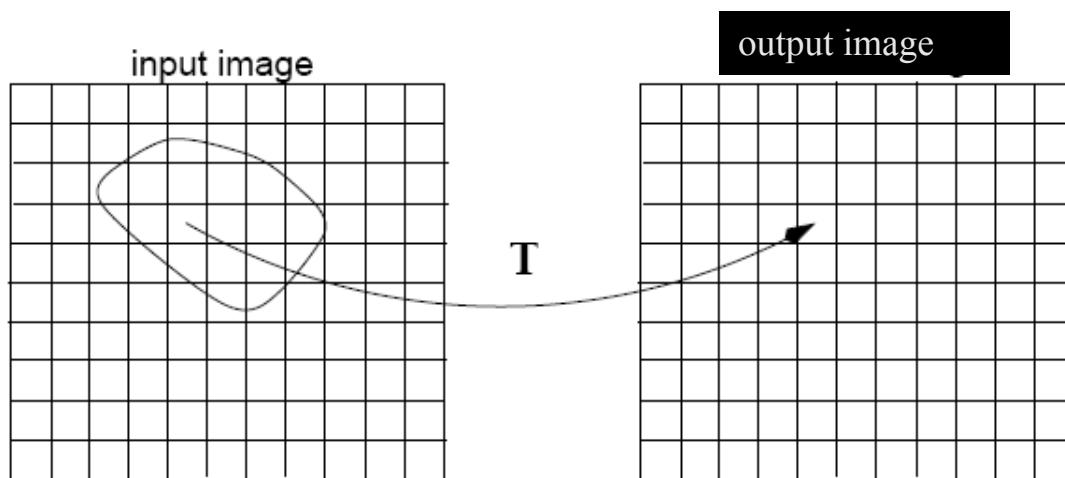


(h) H_R



(i) $H_{A'}$

Spatial Filtering



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

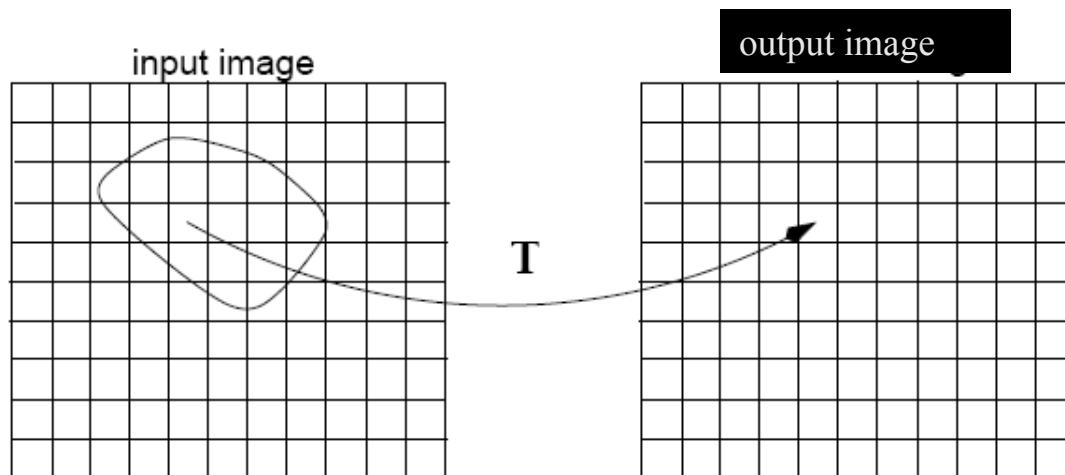
T operates on a
neighborhood of pixels

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

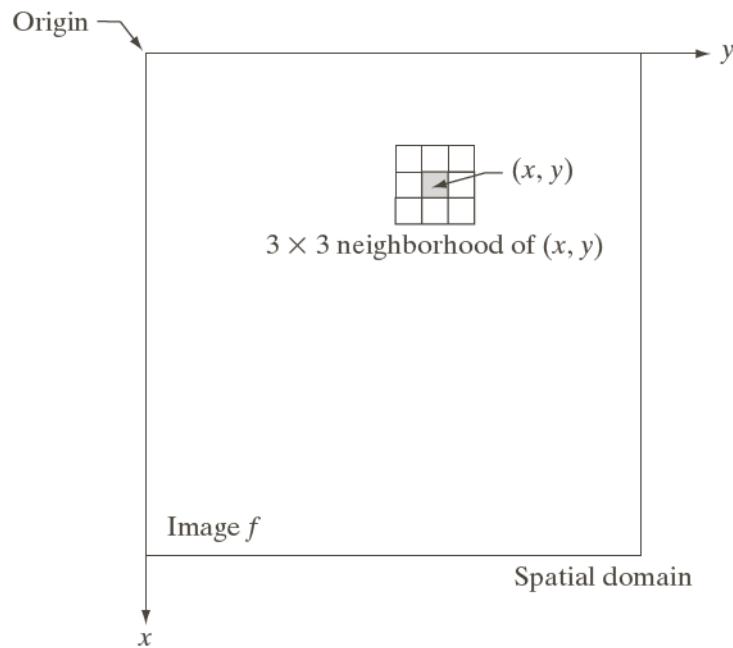


$$g(x,y) = T[f(x,y)]$$

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



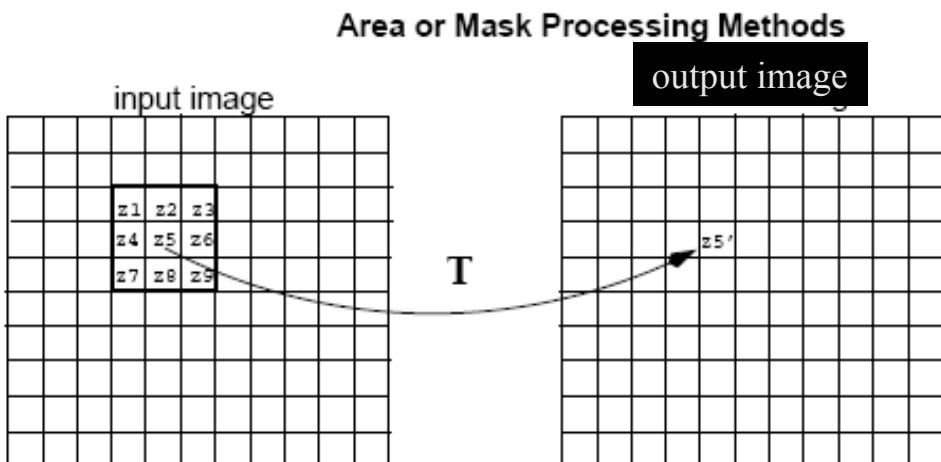
Spatial filtering - Operation

Example: weighted sum of input pixels.

$$z5' = R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

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	0	0	0	0
0	0	1	1	2	5	6	3	6	7	3	0	0	0	0	0
0	0	2	3	4	6	7	5	1	8	4	0	0	0	0	0
0	0	8	7	6	5	7	6	3	3	4	0	0	0	0	0
0	0	2	3	5	6	7	8	2	7	3	0	0	0	0	0
0	0	4	5	3	2	1	6	8	7	2	0	0	0	0	0
0	0	1	4	5	3	2	6	7	8	1	0	0	0	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	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	3			
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?**)

w1	w2	w3
w4	w5	w6
w7	w8	w9

$$z_5' = R = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

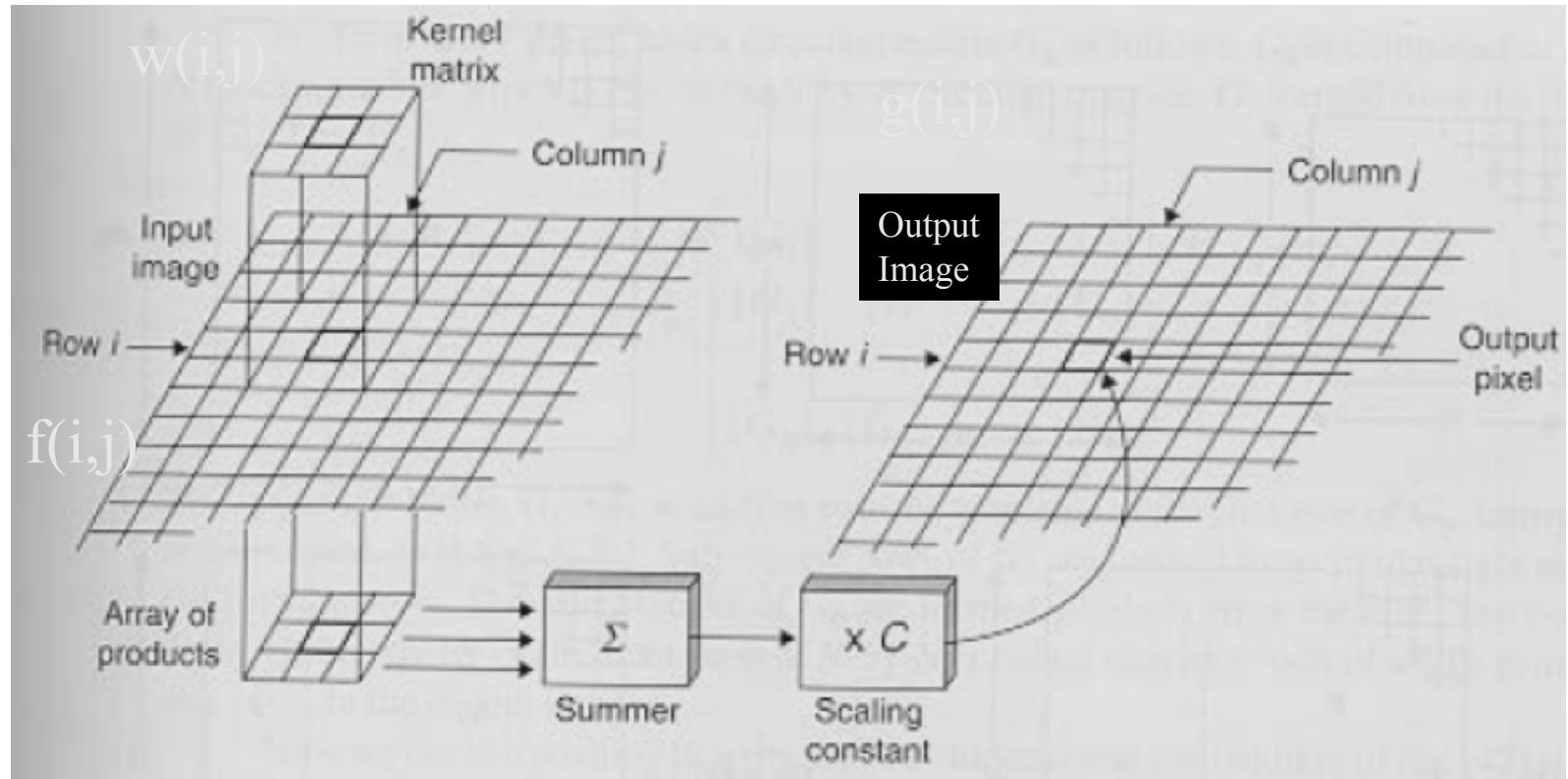
- Methods that do not satisfy the above property are called **non-linear**.

- e.g., $z_5' = \max(z_k, k = 1, 2, \dots, 9)$

Linear Spatial Filtering Methods

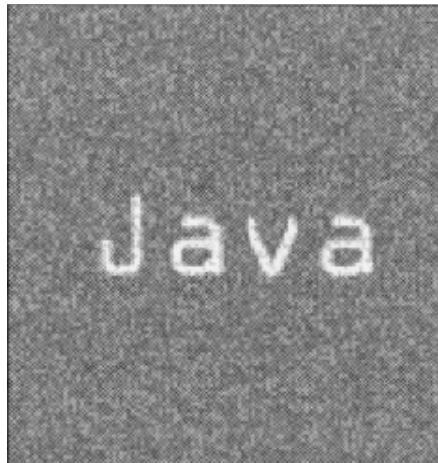
- Main linear spatial filtering methods:
 - Correlation
 - Convolution

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!

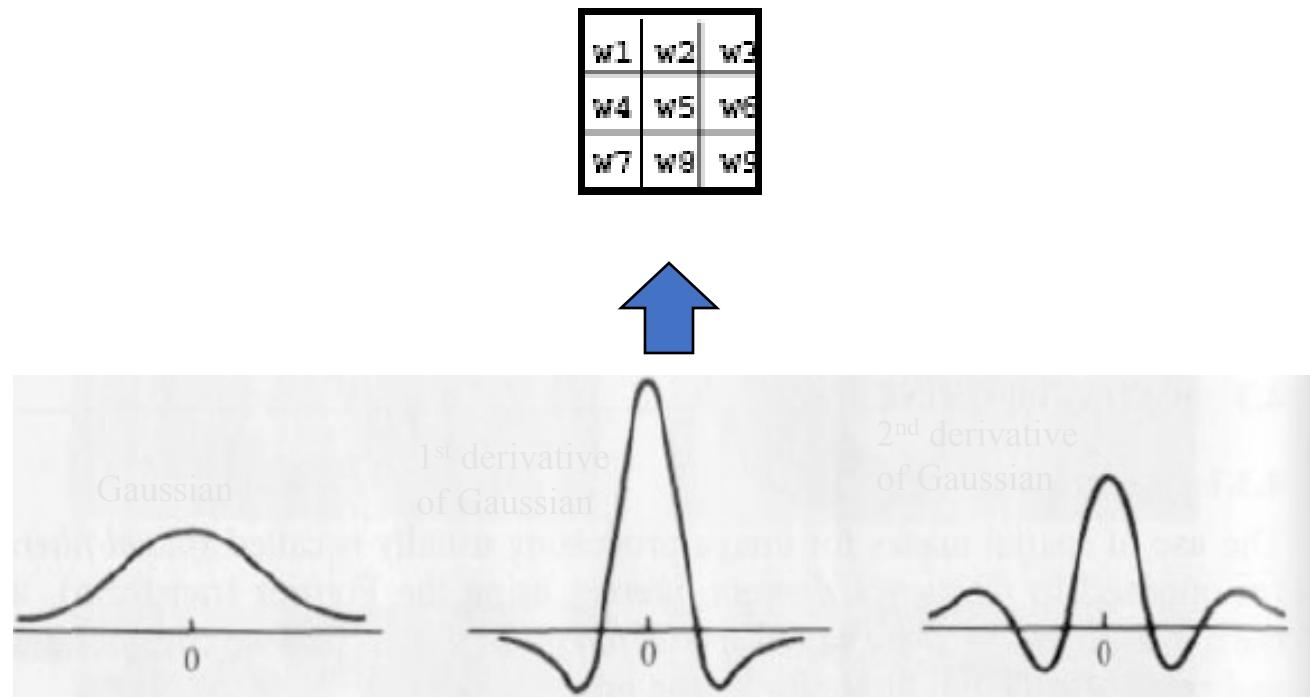
Example

Correlation:

Convolution:

How do we choose the mask weights?

- Typically, by sampling certain functions:

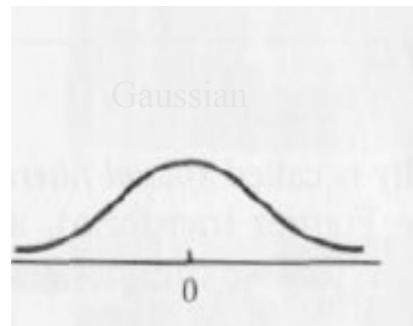


Filters

- 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

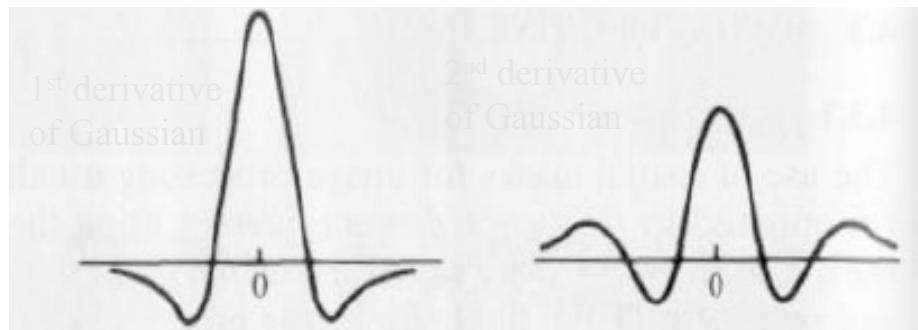


smoothed image



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.

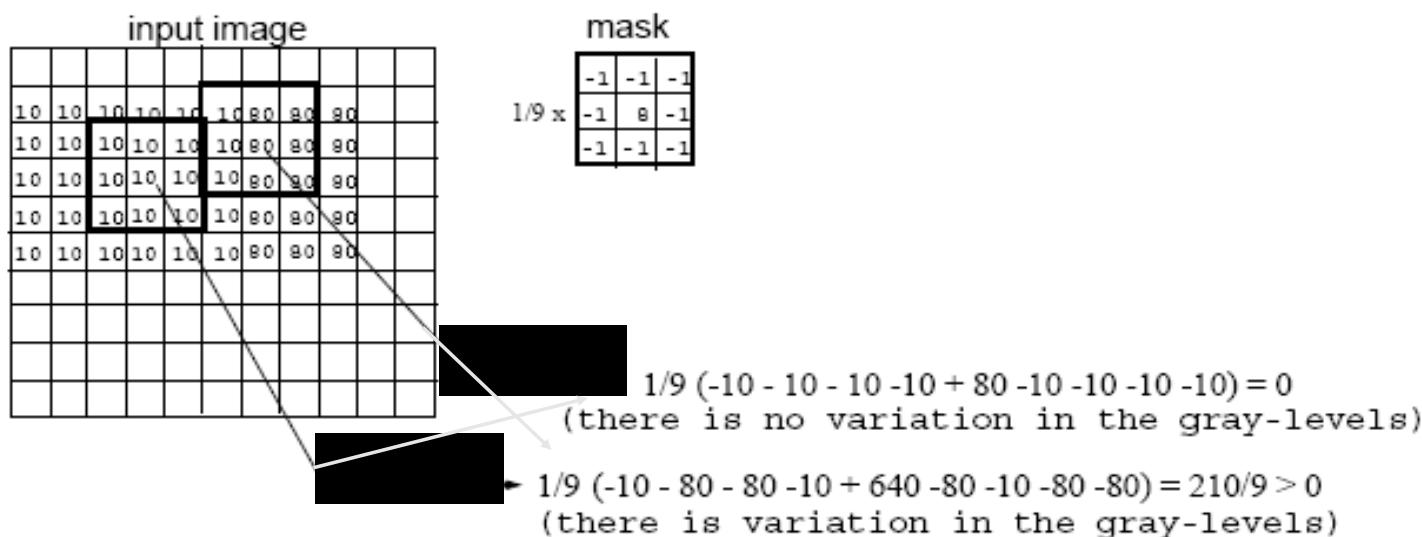


mask

-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

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

(a)

$$\frac{1}{25} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

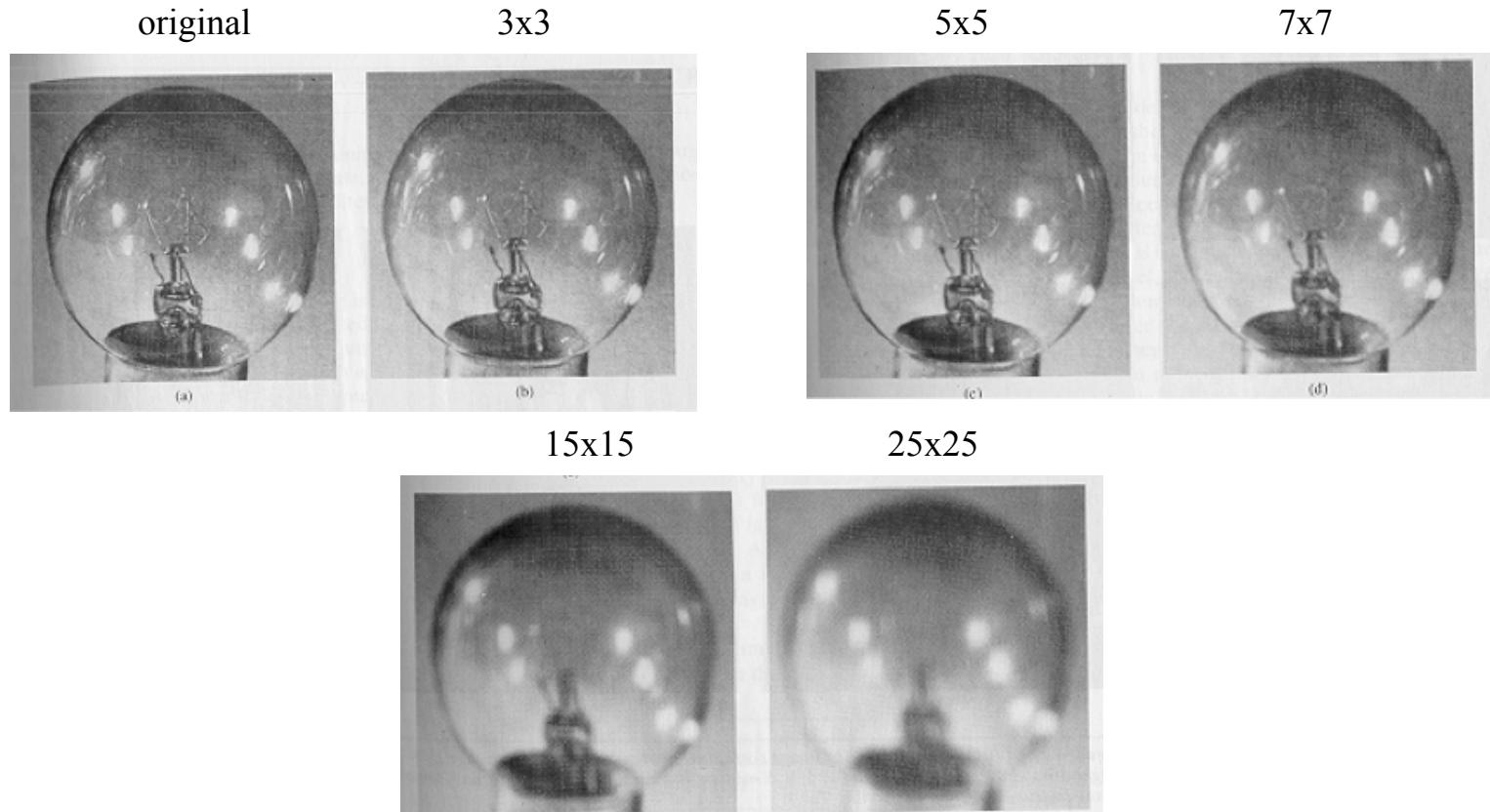
(b)

$$\frac{1}{49} \times \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

(c)

Smoothing Filters: Averaging (cont'd)

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



Smoothing Filters: Averaging (cont'd)

Example: extract largest, brightest objects

15 x 15 averaging

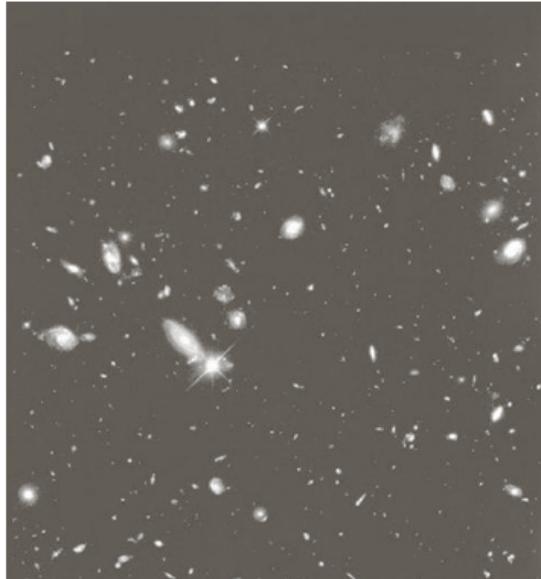
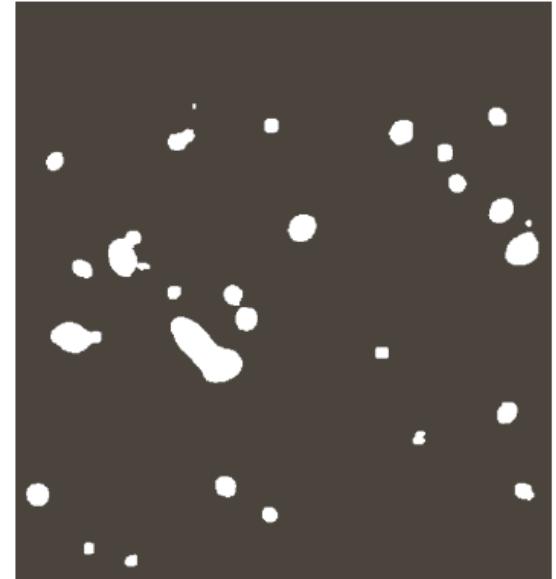
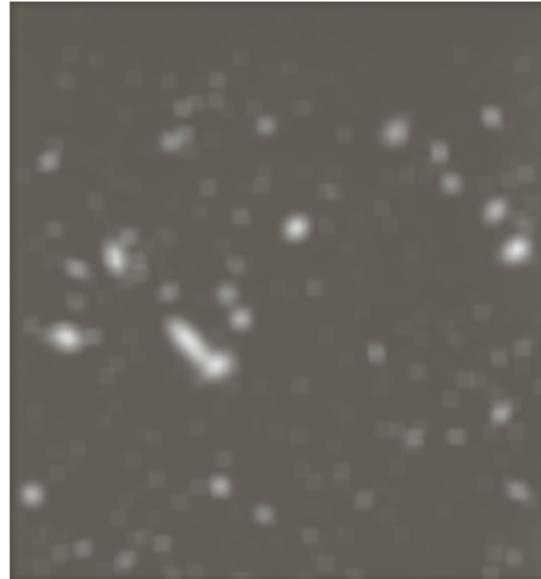


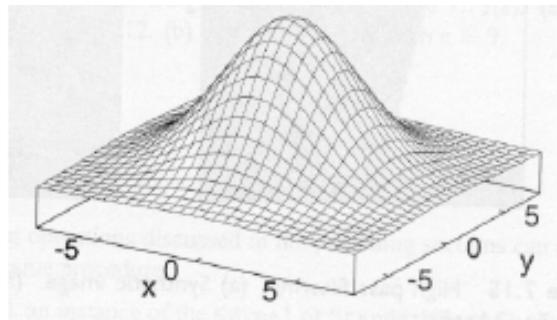
image thresholding



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 :

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

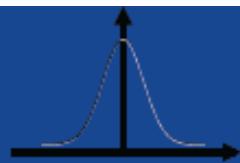
Smoothing filters: Gaussian (cont'd)

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

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

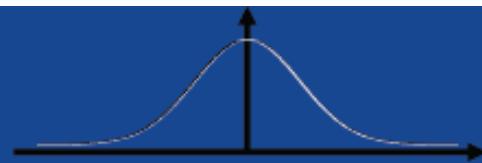
Smoothing filters: Gaussian (cont'd)



small σ



limited smoothing



large σ



strong smoothing

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).



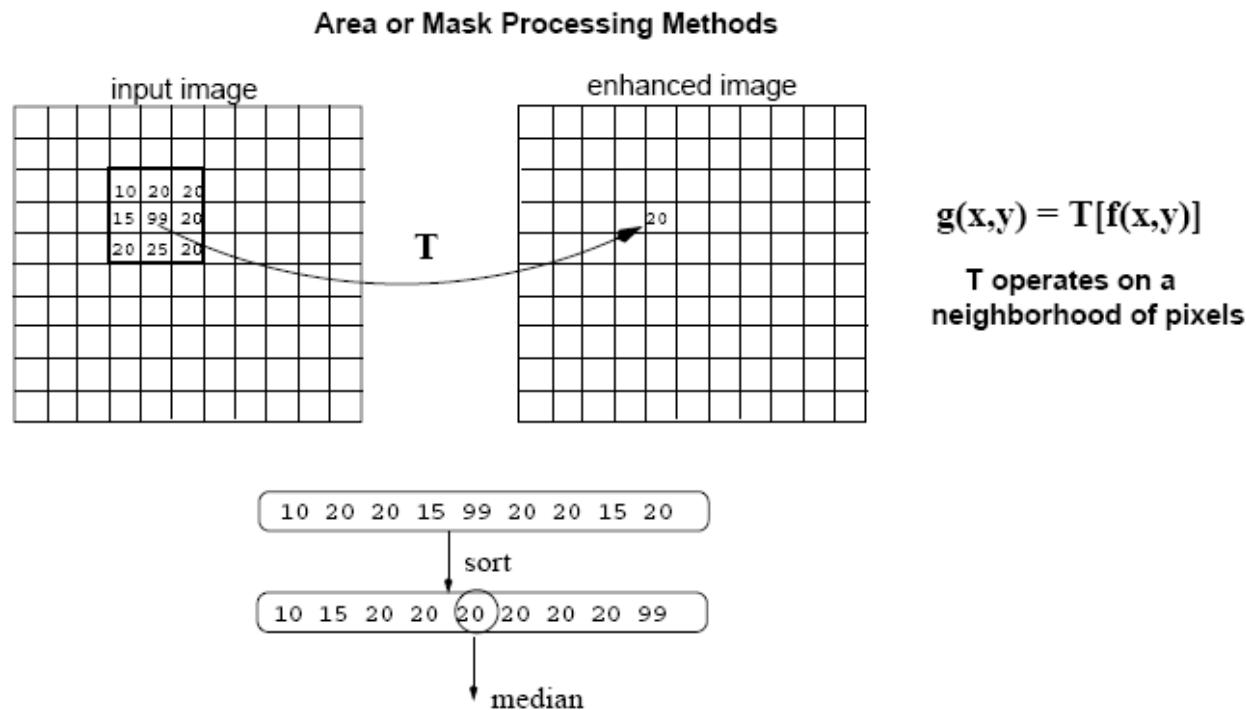
averaging

median
filtering



Smoothing Filters: Median Filtering (cont'd)

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



Convolutional neural network

- Specially designed for data with grid-like structures (LeCun et al. 98)
 - ▶ 1D grid: sequential data
 - ▶ 2D grid: image
 - ▶ 3D grid: video, 3D image volume
- Beat all the existing computer vision technologies on object recognition on ImageNet challenge with a large margin in 2012

Convolutional neural network

We can compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the Conv Layer neurons (F), the stride with which they are applied (S), and the amount of zero padding used (P) on the border. You can convince yourself that the correct formula for calculating how many neurons "fit" is given by $(W - F + 2P)/S + 1$. For example for a 7×7 input and a 3×3 filter with stride 1 and pad 0 we would get a 5×5 output. With stride 2 we would get a 3×3 output. Lets also see one more graphical example:

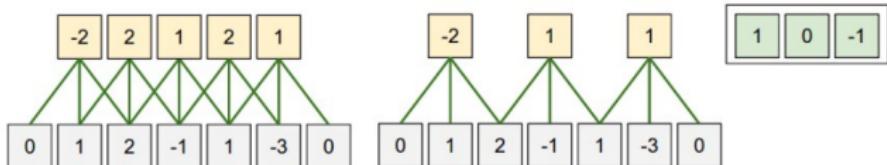


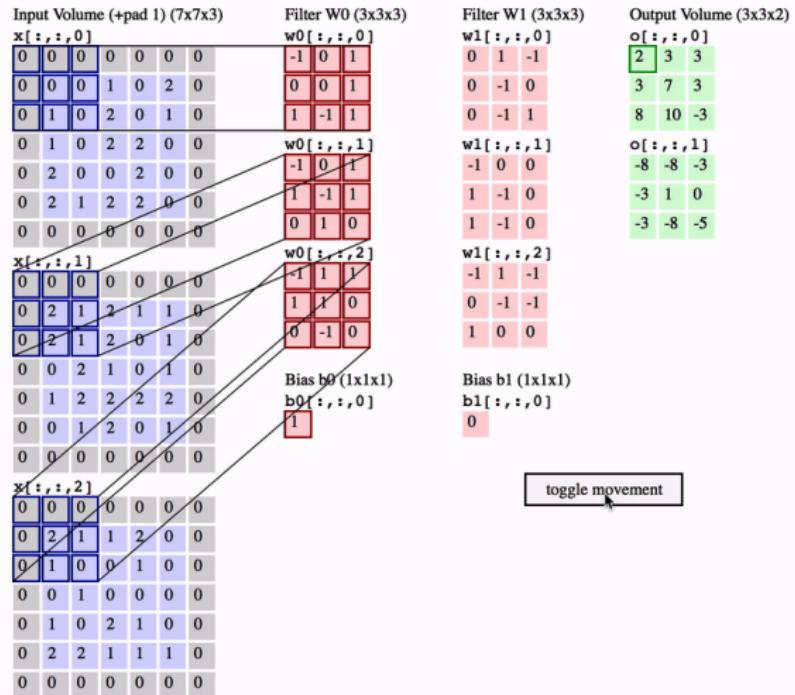
Illustration of spatial arrangement. In this example there is only one spatial dimension (x-axis), one neuron with a receptive field size of $F = 3$, the input size is $W = 5$, and there is zero padding of $P = 1$. **Left:** The neuron strided across the input in stride of $S = 1$, giving output of size $(5 - 3 + 2)/1+1 = 5$. **Right:** The neuron uses stride of $S = 2$, giving output of size $(5 - 3 + 2)/2+1 = 3$. Notice that stride $S = 3$ could not be used since it wouldn't fit neatly across the volume. In terms of the equation, this can be determined since $(5 - 3 + 2) = 4$ is not divisible by 3.

The neuron weights are in this example $[1, 0, -1]$ (shown on very right), and its bias is zero. These weights are shared across all yellow neurons (see parameter sharing below).

Conv1D

<http://cs231n.github.io/convolutional-networks/#overview>

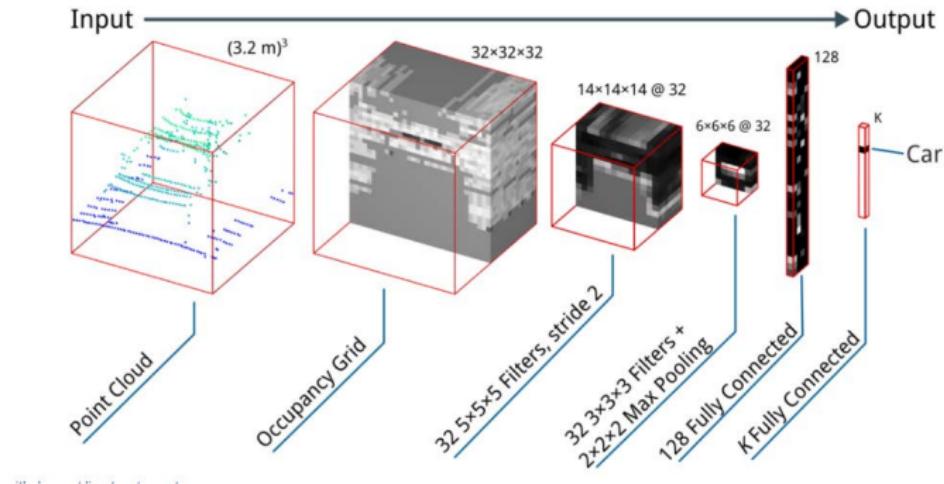
Convolutional neural network



Conv2d

<http://cs231n.github.io/convolutional-networks/#overview>

Convolutional neural network



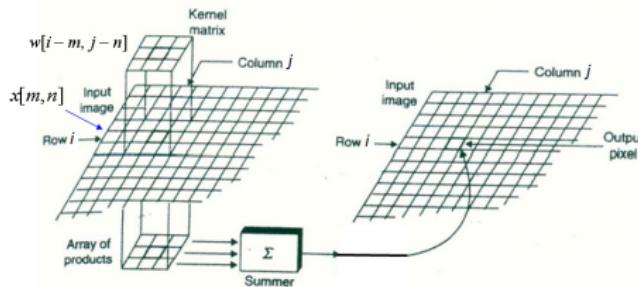
Conv3D

<http://dimatura.net/research/voxnet/>

Convolution

- Computing the responses at hidden nodes is equivalent to convoluting the input image \mathbf{x} with a learned filter \mathbf{w}
- After convolution, a filter map net is generated at the hidden layer
- Parameter sharing causes the layer to have *equivariance* to translation. A function $f(x)$ is equivalent to a function g if $f(g(x)) = g(f(x))$
- Is convolution equivariant to changes in the scale or rotation? (page354)

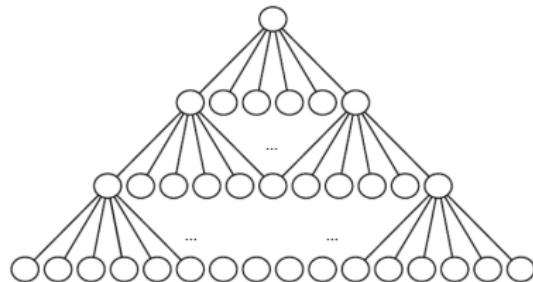
$$net[i, j] = (x * w)[i, j] = \sum_m \sum_n x[m, n] w[i - m, j - n]$$



$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n). \quad (9.6) \quad \text{page 348}$$

Zero-padding in convolutional neural network

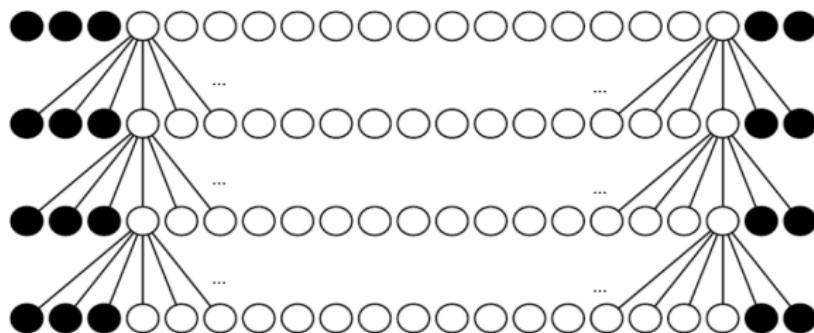
- The valid feature map is smaller than the input after convolution
- Implementation of neural networks needs to zero-pad the input \mathbf{x} to make it wider
- Without zero-padding, the width of the representation shrinks by the filter width - 1 at each layer
- To avoid shrinking the spatial extent of the network rapidly, small filters have to be used



(Bengio et al. Deep Learning 2014)

Zero-padding in convolutional neural network

- By zero-padding in each layer, we prevent the representation from shrinking with depth. It allows us to make an arbitrarily deep convolutional network



(Bengio et al. Deep Learning 2014)

Downsampled convolutional layer

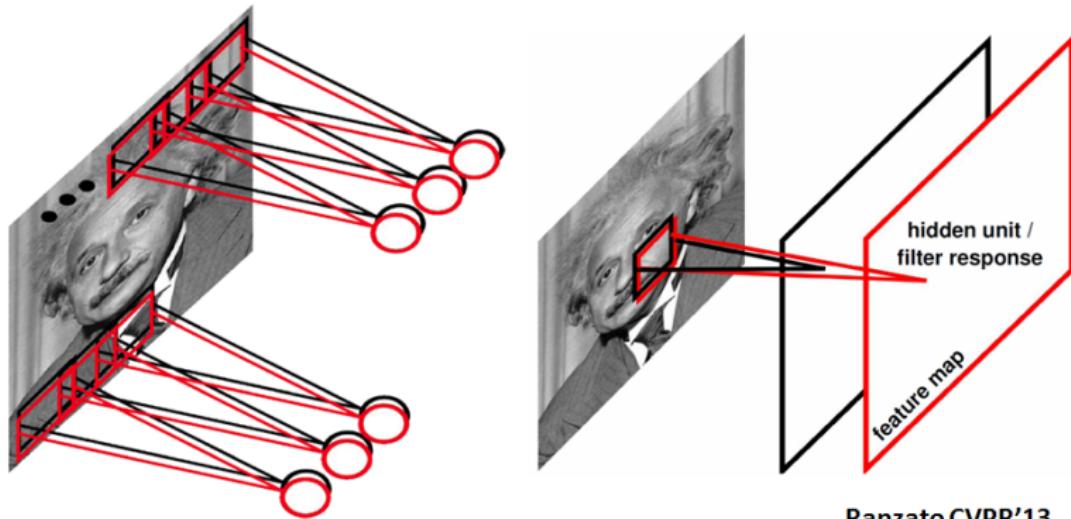
- To reduce computational cost, we may want to skip some positions of the filter and sample only every s pixels in each direction. A downsampled convolution function is defined as

$$net[i, j] = (\mathbf{x} * \mathbf{w})[i \times s, j \times s]$$

- s is referred as the *stride* of this downsampled convolution

Multiple filters

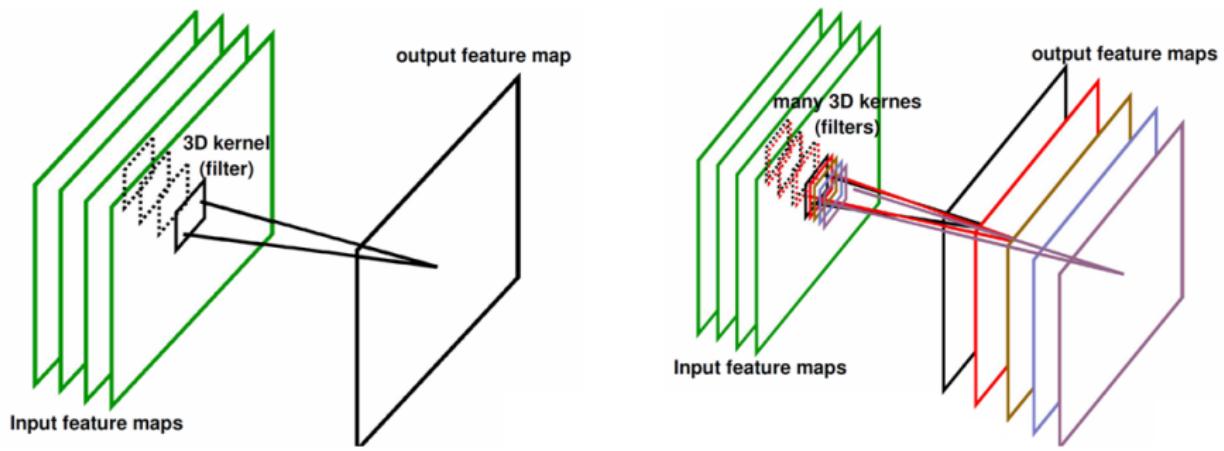
- Multiple filters generate multiple feature maps
- Detect the spatial distributions of multiple visual patterns



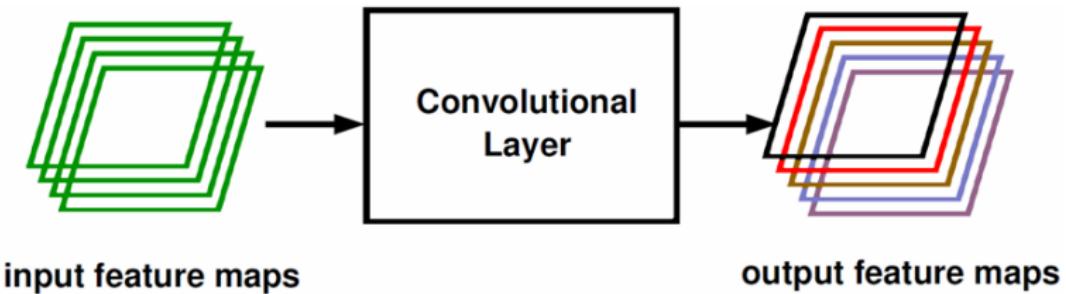
Ranzato CVPR'13

3D filtering when input has multiple feature maps

$$net = \sum_{k=1}^K \mathbf{x}^k * \mathbf{w}^k$$



Convolutional layer



Ranzato CVPR'13

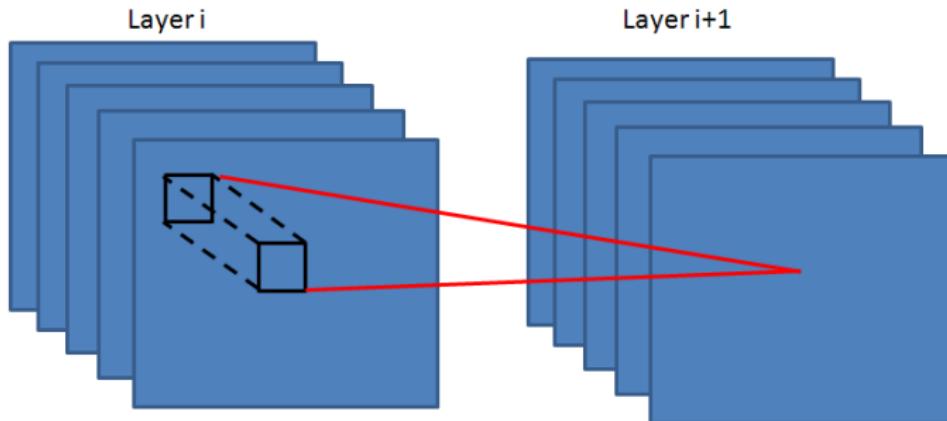
Nonlinear activation function

- $\tanh()$
- Rectified linear unit

Local contrast normalization (LCN)

- Normalization can be done within a neighborhood along both spatial and feature dimensions

$$h_{i+1,x,y,k} = \frac{h_{i,x,y,k} - m_{i,N(x,y,k)}}{\sigma_{i,N(x,y,k)}}$$



Neural network normalization

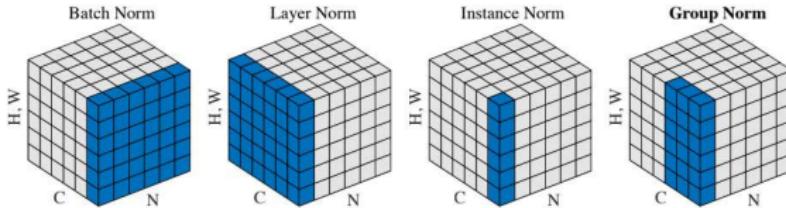


Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

The goal of batch norm is to *reduce internal covariate shift* by normalizing each mini-batch of data using the mini-batch mean and variance. For a mini-batch of inputs $\{x_1, \dots, x_m\}$, we compute

$$\mu = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu)^2$$

and then replace each x_i with its normalized version

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

BN

Instead of normalizing examples across mini-batches, layer normalization *normalizes features within each example*. For input x_i of dimension D , we compute

$$\mu = \frac{1}{D} \sum_{d=1}^D x_i^d$$

$$\sigma^2 = \frac{1}{D} \sum_{d=1}^D (x_i^d - \mu)^2$$

and then replace each component x_i^d with its normalized version

$$\hat{x}_i^d = \frac{x_i^d - \mu}{\sqrt{\sigma^2 + \epsilon}}.$$

LN

Instance normalization

Instance norm (Ulyanov, Vedaldi, & Lempitsky, 2016) hit arXiv just 6 days after layer norm, and is pretty similar. Instead of normalizing all of the features of an example at once, instance norm normalizes features within each channel.

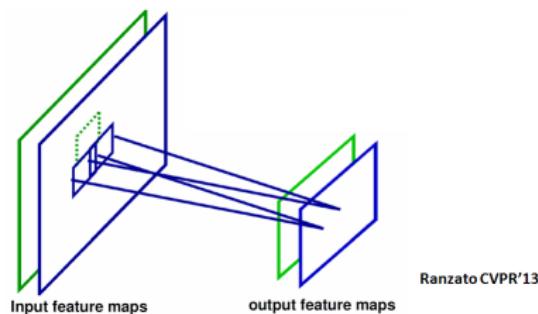
Group normalization

Group norm (Wu & He, 2018) is somewhere between layer and instance norm — instead of normalizing features within each channel, it normalizes features within pre-defined groups of channels.⁴

<https://nealjean.com/ml/neural-network-normalization/>

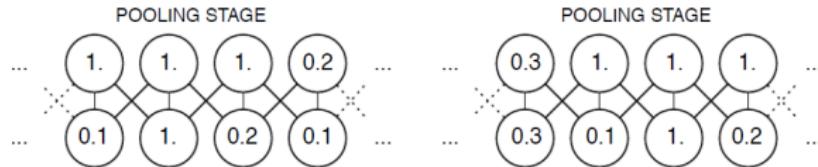
Pooling

- Max-pooling partitions the input image into a set of rectangles, and for each sub-region, outputs the maximum value
- Non-linear down-sampling
- The number of output maps is the same as the number of input maps, but the resolution is reduced
- Reduce the computational complexity for upper layers and provide a form of **translation invariance**
- Average pooling can also be used



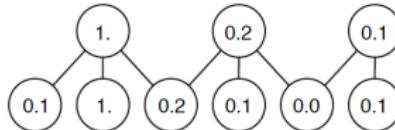
Pooling

- Pooling without downsampling (stride $s = 1$)
- Invariance vs. information loss (even if the resolution is not reduced)
- Pooling is useful if we care more about whether some feature is present than exactly where it is. It depends on applications.



(Bengio et al. Deep Learning 2014)

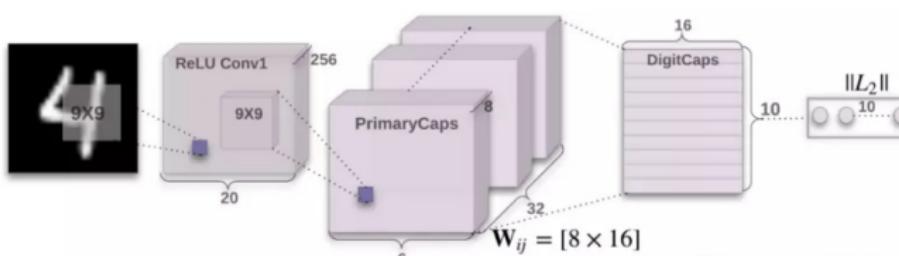
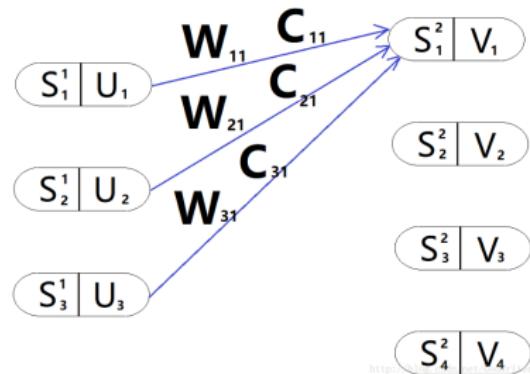
- Pooling with downsampling (commonly used)
- Improve computation efficiency



(Bengio et al. Deep Learning 2014)

Possible extension of pooling

1. CapsNet (More accurate spatial relationship)
2. Tiled-Conv (Scale and rotation invariance, large scale and rotation???)



Possible extension of CNN

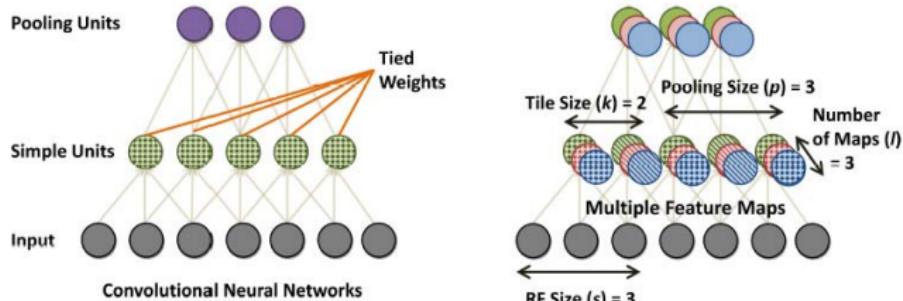


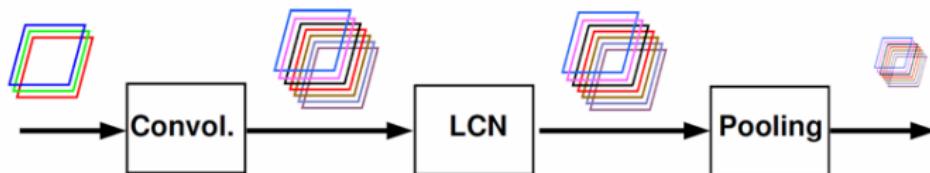
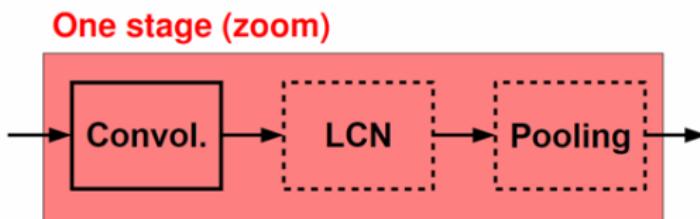
Figure 1: Left: Convolutional Neural Networks with local receptive fields and tied weights. Right: Partially untied local receptive field networks – Tiled CNNs. Units with the same color belong to the same map; within each map, units with the same fill texture have tied weights. (Network diagrams in the paper are shown in 1D for clarity.)

"By visualizing [28, 29] the range of optimal stimulus that activate each pooling unit in a Tiled CNN, we found units that were scale and rotationally invariant.⁹ We note that a standard CNN is unlikely to be invariant to these transformations."

What about large scale and rotation invariant??? Open issue, final project???

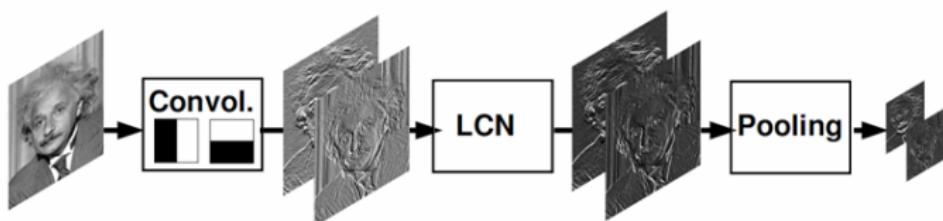
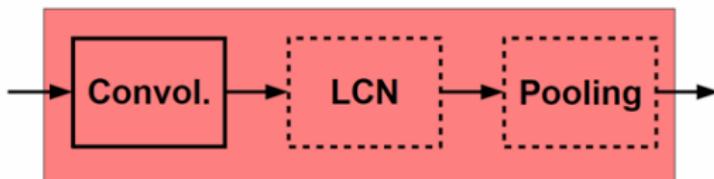
Typical architecture of CNN

- Convolutional layer increases the number of feature maps
- Pooling layer decreases spatial resolution
- LCN and pooling are optional at each stage



Ranzato CVPR'13

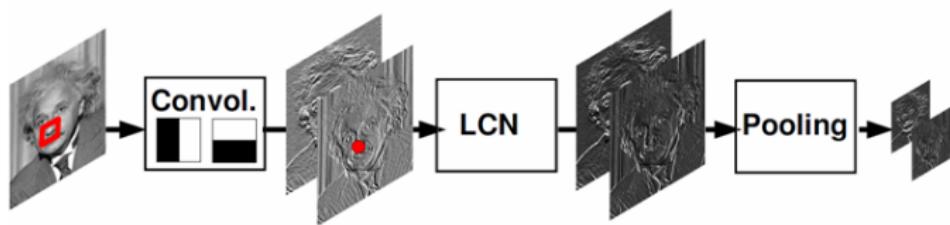
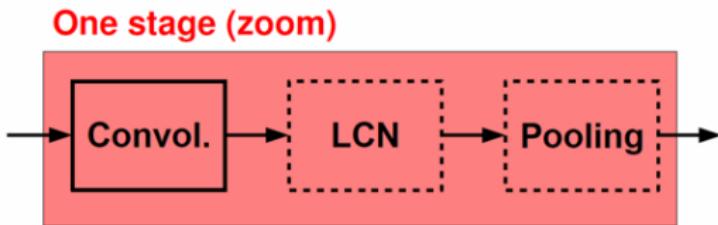
Typical architecture of CNN



Example with only two filters.

Ranzato CVPR'13

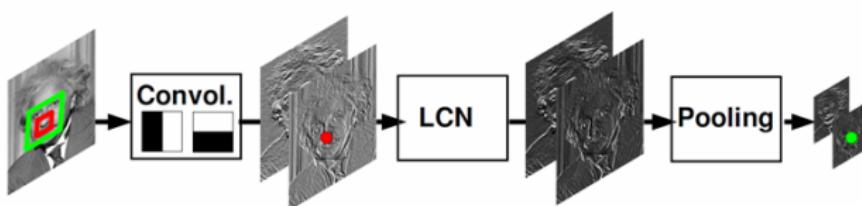
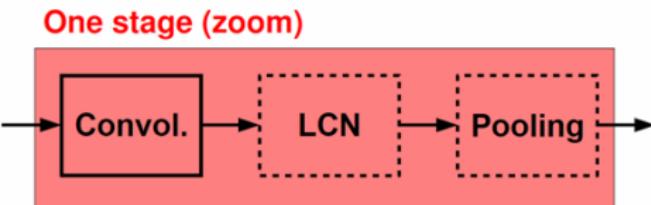
Typical architecture of CNN



A hidden unit in the first hidden layer is influenced by a small neighborhood (equal to size of filter).

Ranzato CVPR'13

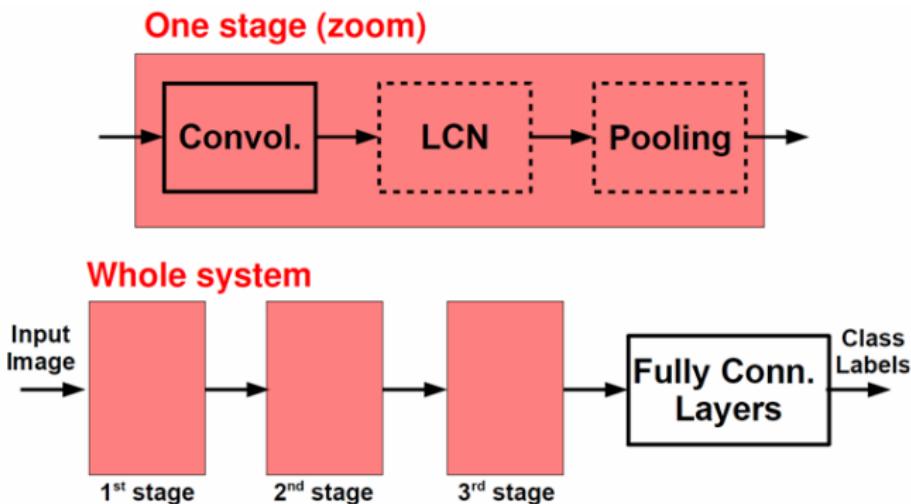
Typical architecture of CNN



A hidden unit after the pooling layer is influenced by a larger neighborhood
(it depends on filter sizes and the sizes of pooling regions)

Ranzato CVPR'13

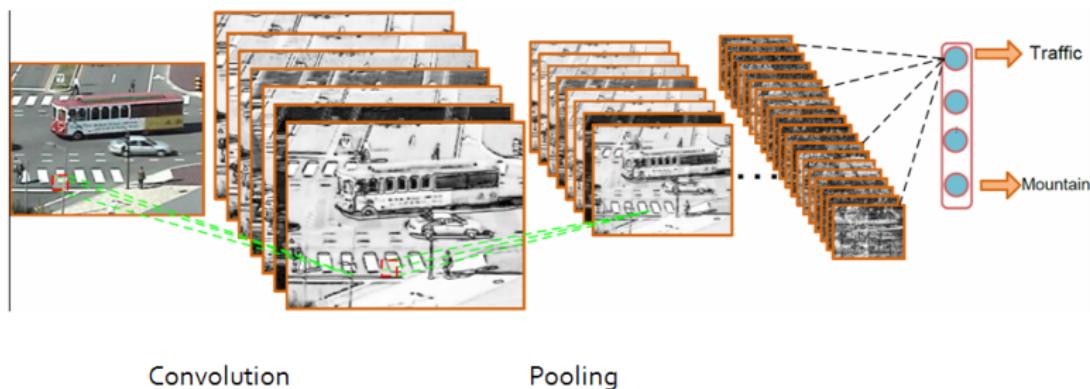
Typical architecture of CNN



After a few stages, residual spatial resolution is very small.

We have learned a descriptor for the whole image. **Ranzato CVPR'13**

Typical architecture of CNN



Convolution

Pooling

Sharpening Filters

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

Sharpening Filters: Unsharp Masking

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

$$\text{Highpass} = \text{Original} - \text{Lowpass}$$



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.

$$\begin{aligned} \text{Highboost} &= A \text{ Original} - \text{Lowpass} \\ &= (A - 1) \text{ Original} + \text{Original} - \text{Lowpass} \\ &= (A - 1) \text{ Original} + \text{Highpass} \end{aligned}$$



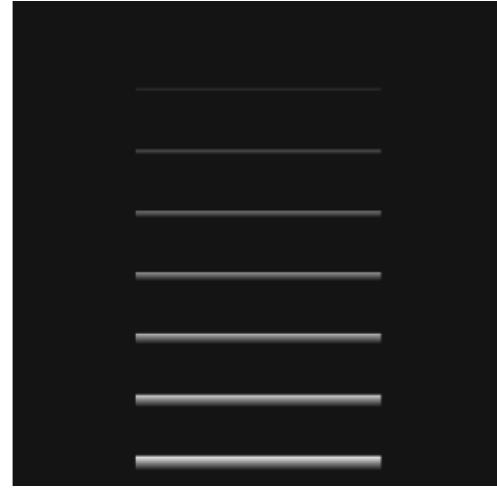
Sharpening Filters: High Boost (cont'd)

- 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:

$A \geq 1$		
$w = 9A - 1$		
-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.

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

Gradient

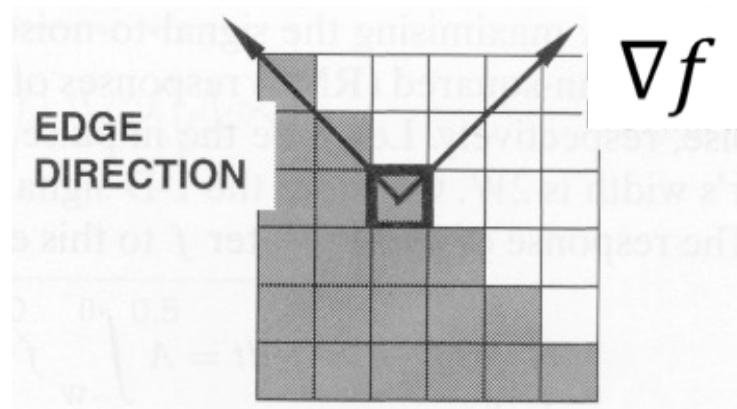
- The gradient is a **vector** which has **magnitude** and **direction**:

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

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

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:

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

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

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

Gradient Computation (cont'd)

$$\frac{f(x_3, y_2) - f(x_3, y_3)}{y_2 - y_3}$$

sensitive to horizontal edges

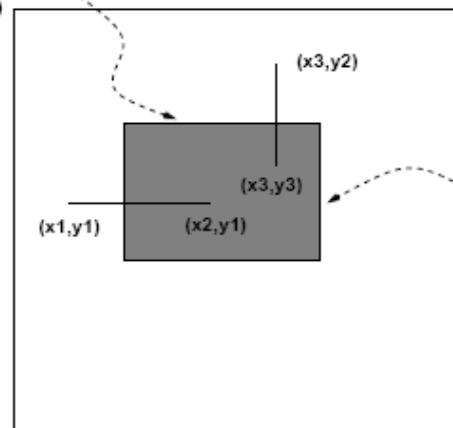
or $\frac{f(x, y+Dy) - f(x, y)}{-Dy} = \boxed{f(x, y) - f(x, y+1)}$ (grad in the y-direction)

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

($y_3=y_2+Dy$, $y_2=y$, $x_3=x$, $Dy=1$)

edge in the x-direction

(0,0)



sensitive to vertical edges

edge in the y-direction

$$\frac{f(x_2, y_1) - f(x_1, y_1)}{x_2 - x_1}$$

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

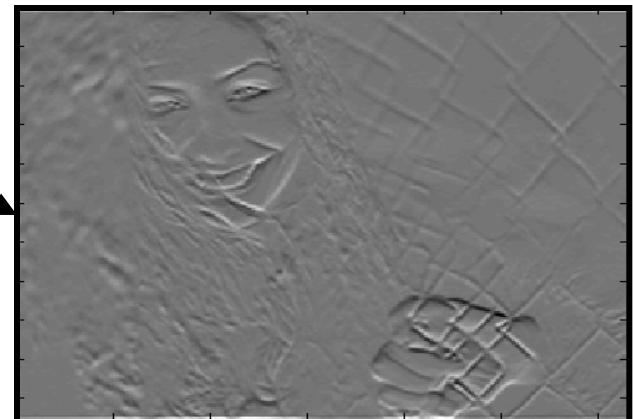
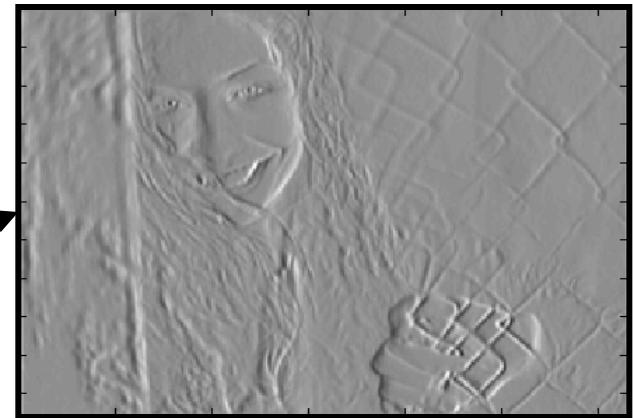
$$\boxed{f(x+1, y) - f(x, y)}$$

(grad in the x-direction)

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

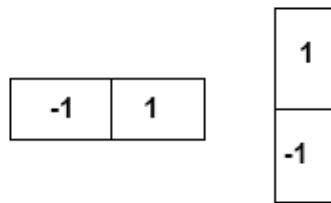
($x_2=x+Dx$, $x_1=x$, $y_1=y_2=y$, $Dx=1$)

Example: visualize partial derivatives



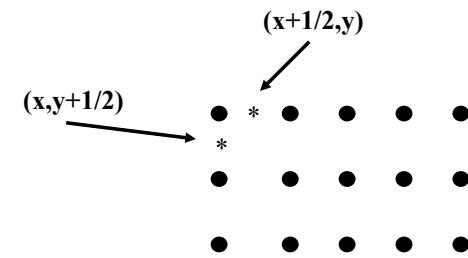
Implement Gradient Using Masks

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



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

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

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

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

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

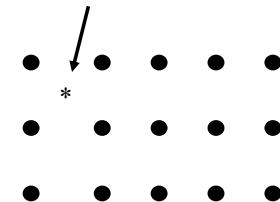
Implement Gradient Using Masks (cont'd)

- A different approximation of the gradient:

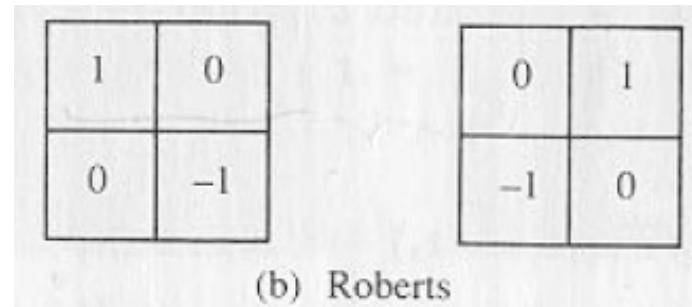
$$\frac{\partial f}{\partial x}(x, y) = f(x, y) - f(x + 1, y + 1)$$

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

good approximation
($x+1/2, y+1/2$)



- 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

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

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

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

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

Prewitt

-1	-2	-1
0	0	0
1	2	1

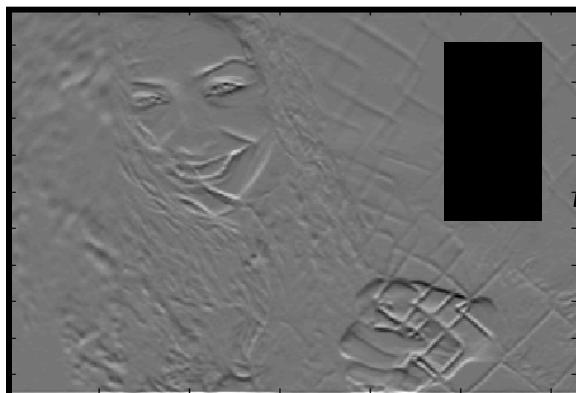
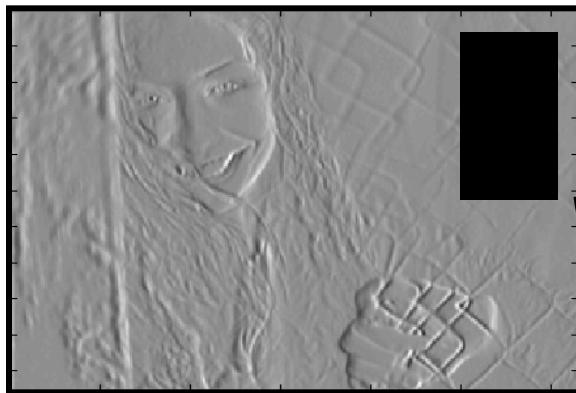
$\frac{\partial f}{\partial y}$

-1	0	1
-2	0	2
-1	0	1

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

Sobel

Example: Gradient Magnitude Image



Gradient Magnitude

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



(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)

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

Approximate
2nd derivatives:

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

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

Laplacian (cont'd)

Laplacian Mask

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & -2 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & -2 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

Edges can be found
by detect the zero-crossings

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

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

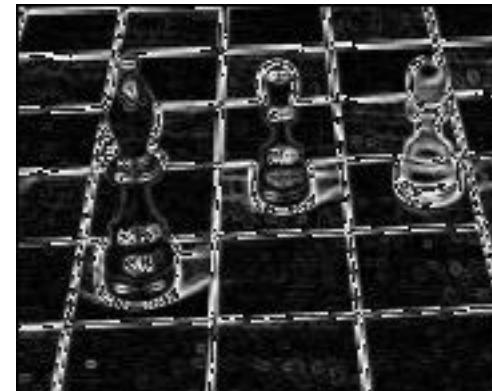
Example: Laplacian vs Gradient

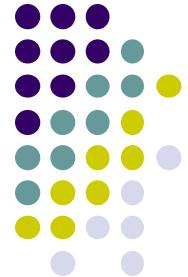


Laplacian



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