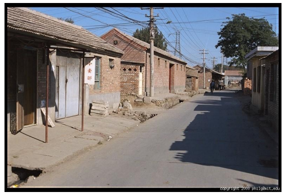
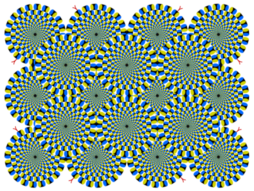
Images and Image Filtering (2)

Lu Sheng (盛律)   Spring 2024

Key properties of linear filters

- Linearity

```
imfilter(I_1 + I_2, f) = imfilter(I_1, f) + imfilter(I_2, f)
```

- Shift-invariant
  - Same behavior given intensities regardless of the pixel location

```
imfilter(shift(I), f) = shift(imfilter(I, f))
```

- Any linear, shift-invariant operator can be represented as a **convolution**

Correlation *v.s.* convolution

- 2D **correlation**: similarity between two signals

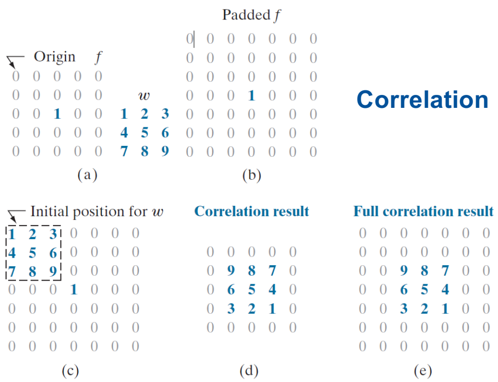
$$h[m,n] = \sum_{k,l} f[k,l] I[m+k, n+l] = f \otimes I$$

- 2D **convolution**: effect of one signal onto another

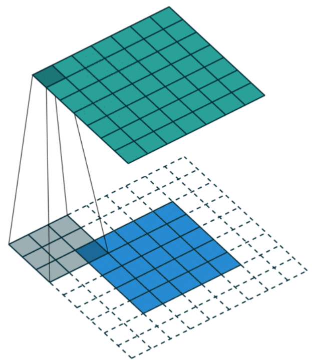
$$h[m,n] = \sum_{k,l} f[k,l] I[m-k, n-l] = f * I$$

- Convolution is the same as correlation with a **180-degree rotated filter kernel**
- Convolution and correlation are identical when the filter kernel is symmetric

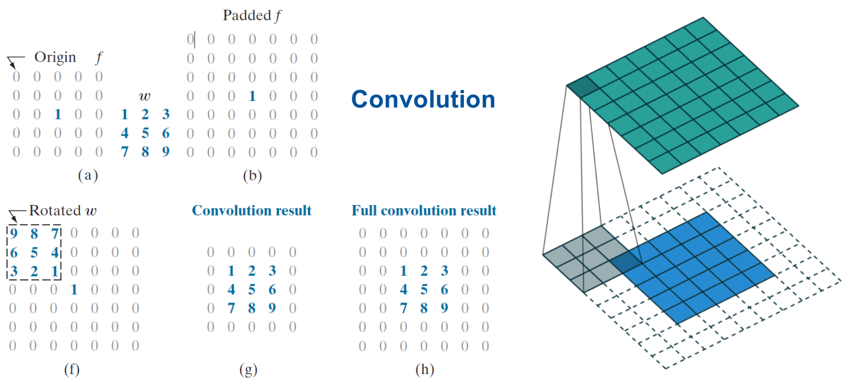
Correlation *v.s.* convolution



Correlation



Correlation *v.s.* convolution



Convolution properties

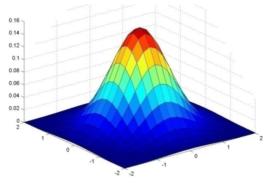
- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
  - Often apply several filters one after another $((a * b_1) * b_2) * b_3$
  - This is equivalent to apply one filter $a * (b_1 * b_2 * b_3)$
  - Correlation is **NOT** associative
  - Associative is important for image filtering

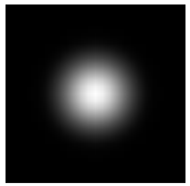
Convolution properties

- Commutative: $a * b = b * a$
- Associative: $a * (b * c) = (a * b) * c$
- Distributes over addition: $a * (b + c) = a * b + a * c$
- Scalars factor out: $ka * b = a * kb = k(a * b)$
- Identity: $\begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{bmatrix} * I = I$

Another example: Gaussian filter

- Weighted contributions of neighboring pixels by nearness

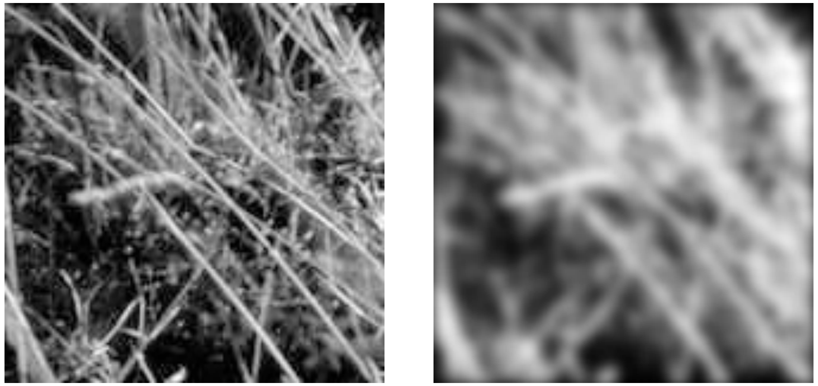




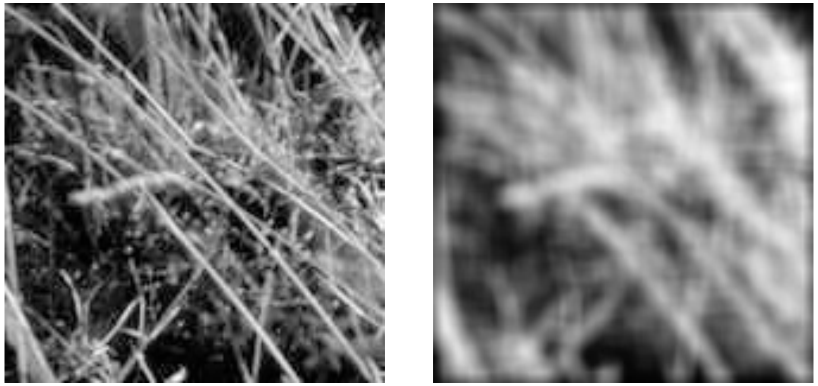
0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

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

Gaussian filter



Box filter



Gaussian filter

- Remove high-frequency components from the image
  - Low-pass filter
  - image becomes more smooth
- Convolution with self is another Gaussian
  - So can smooth with *small-width kernel* -> *repeat* -> get same result as *larger-width kernel* would have
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$

Gaussian filter

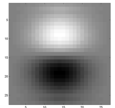
- **Separable** kernel
  - Factors into product of two 1D Gaussians

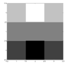
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$
$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right)\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{y^2}{2\sigma^2}\right)\right)$$

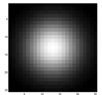
- How big should the filter be?
  - Values at edges should be near zero
  - set filter half-width to about 3σ

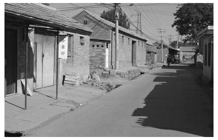
A small quiz

```
a) _ = D * B
b) A = _ * _
c) F = D * _
d) _ = D * D
```

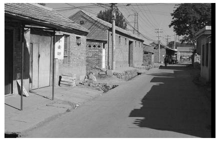

A

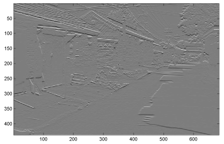

B


C

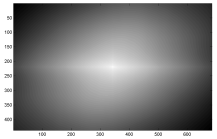

D


E


F


G


H


I

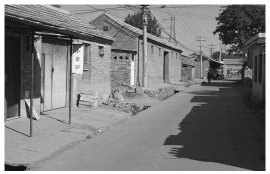
A small quiz

- Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise

- Write down a filter that will compute the gradient in the x-direction:

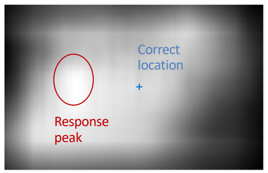
```
gradx(y, x) = im(y, x+1) - im(y, x-1), for each x, y
```


Template matching by correlation



$\otimes$

$\rightarrow$



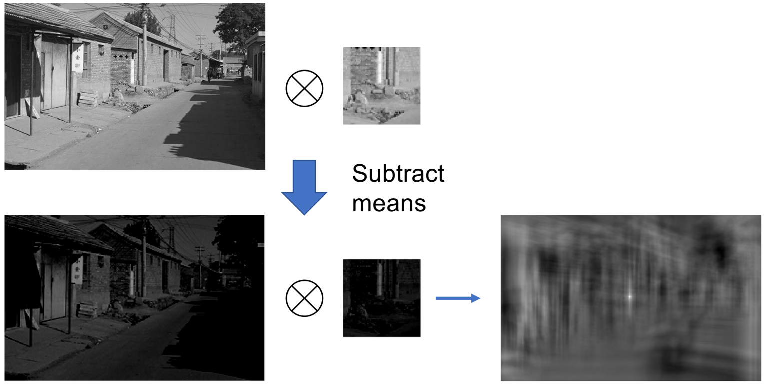
- Correlation did not work here
- **Why not?**

Template matching by correlation

$$h[m,n] = \sum_{k,l} f[k,l] I[m+k, n+l]$$

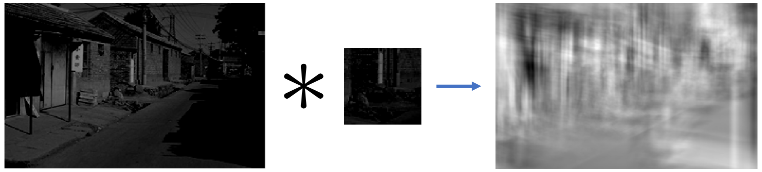
- As brightness in *I* increases, the response in *h* will increase, as long as *f* is positive
- Overall brighter regions will give higher correlation response -> **not useful!**
- Then how to improve?

Template matching by correlation



Template matching by correlation

- What about using convolution?



北京航空航天大學
COLLEGE OF SOFTWARE BEIHANG UNIVERSITY 软件学院

Non-Linear Filtering

Mean filtering

$f[\cdot,\cdot]: \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$I[\cdot,\cdot]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[\cdot,\cdot]$

	0	10	20	30	30				
				50					

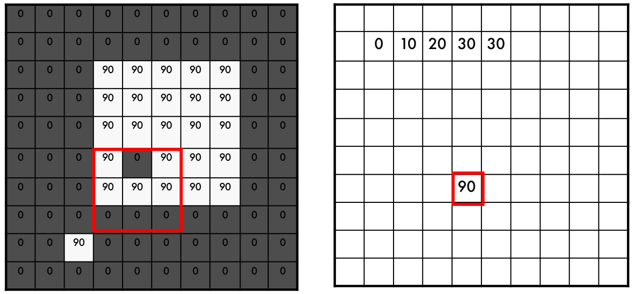
$$h[m,n] = \sum_{k,l} f[k,l] I[m+k, n+l]$$

Median filtering

- Operates over a window by selecting the **median** intensity in the window
  - Step-1: calculating the intensity histogram in the local window
  - Step-2: sorting the histogram in either ascending or descending order
  - Step-3: selecting the median bin, and return the gray-level associated with this bin
- Median filter is not a convolution, is non-linear
- More non-linear filters: min, max, range filters

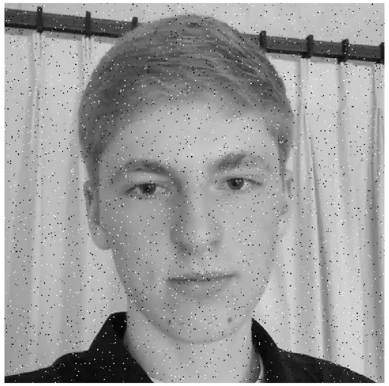
Median filtering

$I[\cdot, \cdot]$ $h[\cdot, \cdot]$

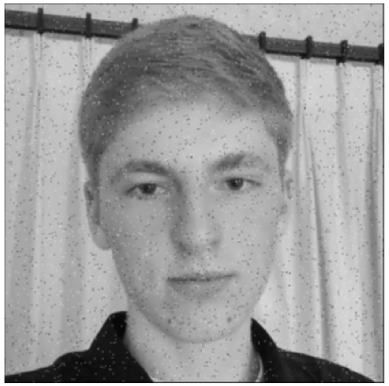


What advantage does a median filter have over a mean filter?

Salt-and-pepper noise



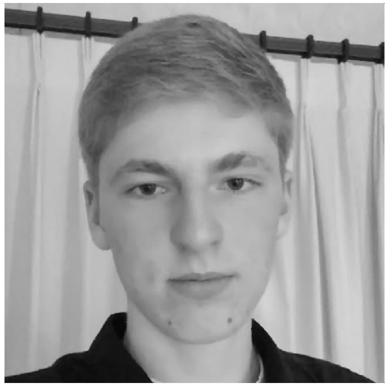
Mean filtering (Box filter)



3×3

11×11

Median filtering



3×3

11×11