Three views of filtering

- Image filters in spatial domain (空域）
  - Filter is a mathematical operation of a grid of numbers
  - Smoothing, sharpening, measuring texture
- Image filters in the frequency domain （频域）
  - Filtering is a way to modify the frequencies of images
  - Denoising, sampling, image compression
- Image pyramids （图像金字塔）
  - Scale-space representation allows coarse-to-fine operations

James Hays

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

h=output f=filter I=image

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

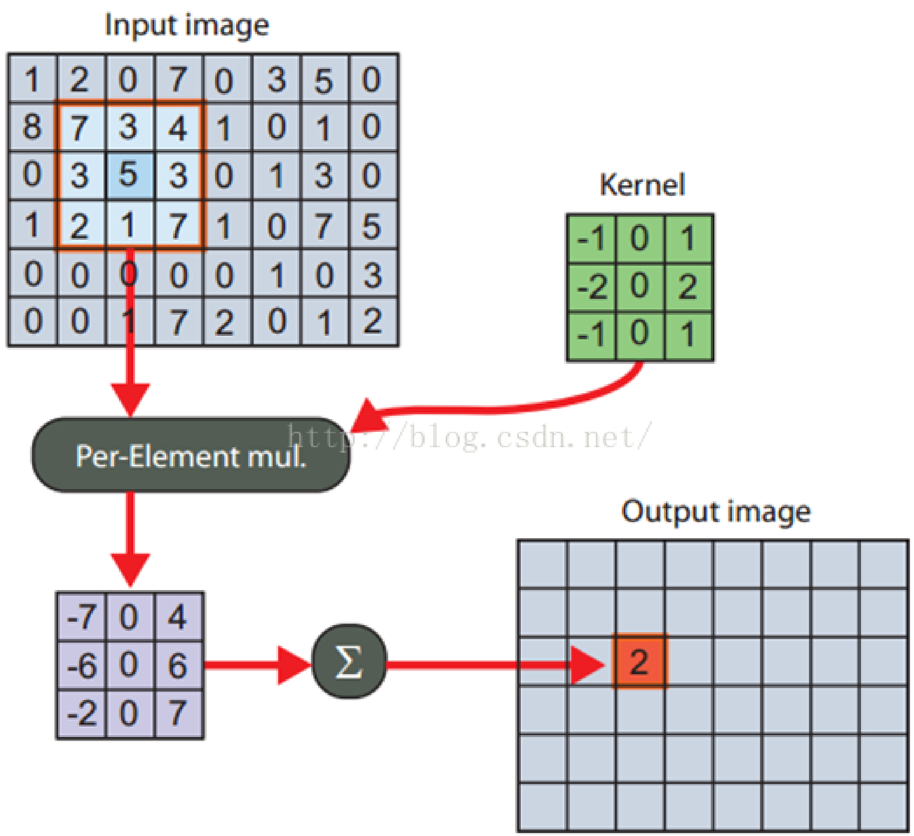
2d coords=k,l 2d coords=m,n

[ ] [ ] [ ]

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

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



James Hays

Example: box filter

$$f[\cdot,\cdot]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image filtering

$$f[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$I[.,.]$$

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

$$h[.,.]$$

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

Credit: S. Seitz

Image filtering

$$f[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10							

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

未来媒体研究中心 CENTER FOR FUTURE MEDIA

电子科技大学 University of Electronic Science and Technology of China

Credit: S. Seitz

Image filtering

$$f[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10	20						

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

	0	10	20	30					

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

Credit: S. Seitz

Image filtering

$$f[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10	20	30	30				

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

Credit: S. Seitz

Image filtering

$$f[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
				?					

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

Credit: S. Seitz

Image filtering

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

1	1	1
1	1	1
1	1	1

$I[.,.]$

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

$h[.,.]$

	0	10	20	30	30				
						?			
				50					

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

Credit: S. Seitz

Image filtering

$$f[\cdot,\cdot] \quad \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$I[.,.]$

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

$h[.,.]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Credit: S. Seitz

Box Filter

$$f[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

Slide credit: David Lowe (UBC)

Box Filter

What does it do?

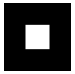
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
- Why does it sum to one?

$$f[\cdot\,,\cdot\,]$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing with box filter

James Hays



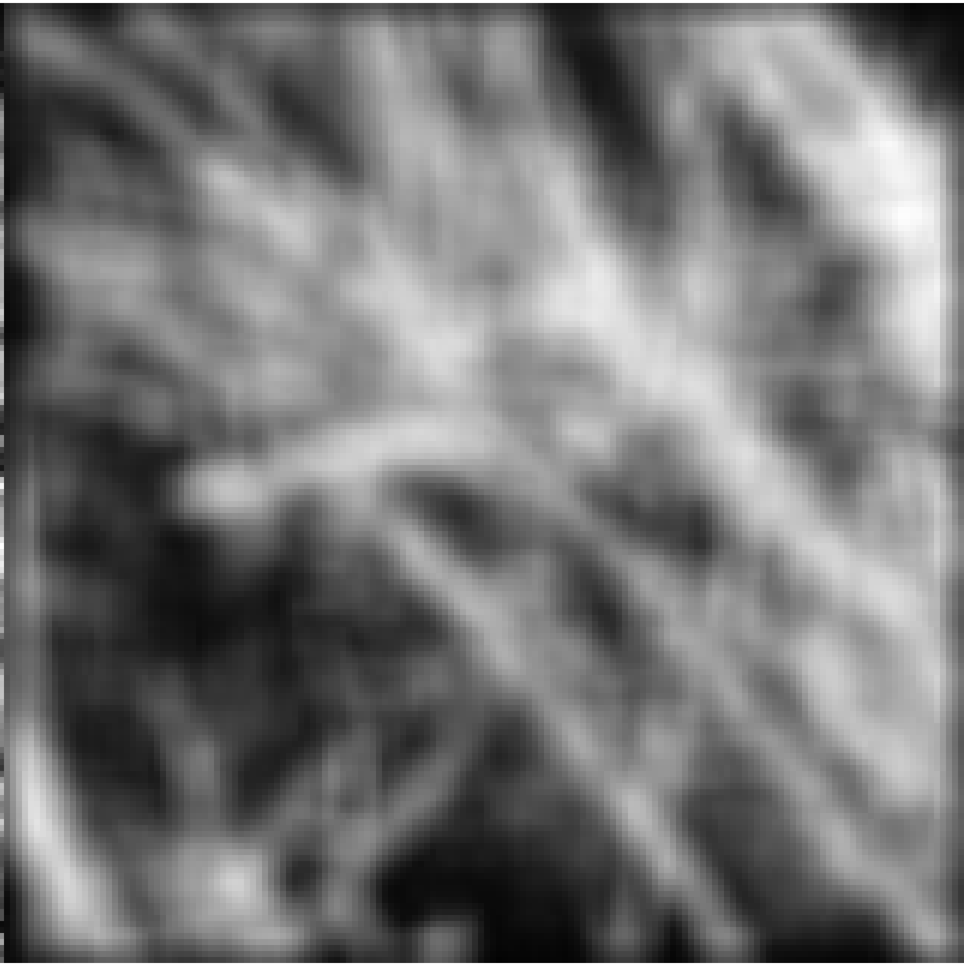


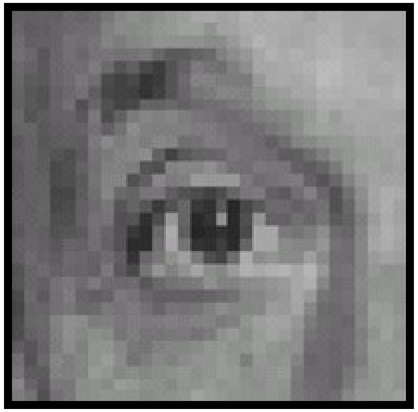
Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position

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

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Think-Pair-Share time



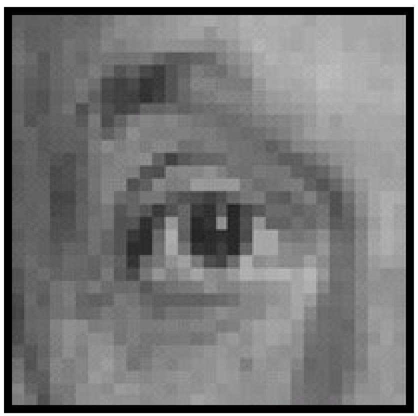
1. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

2. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

3. $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$

4. $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

1. Practice with linear filters



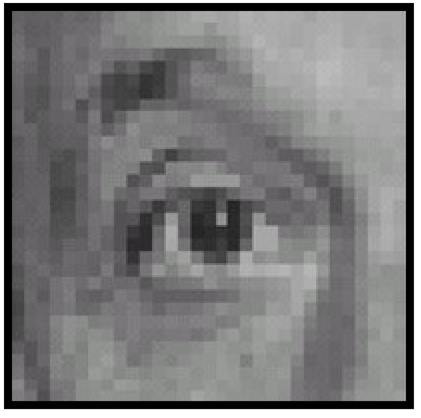
Original

0	0	0
0	1	0
0	0	0

?

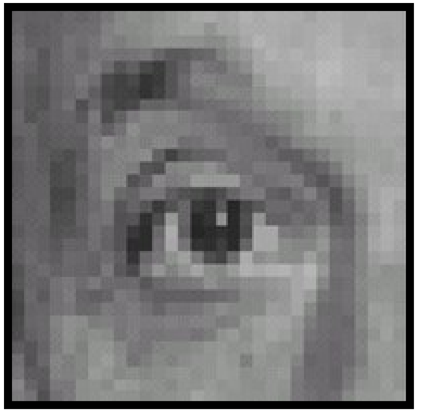
Source: D. Lowe

1. Practice with linear filters



Original

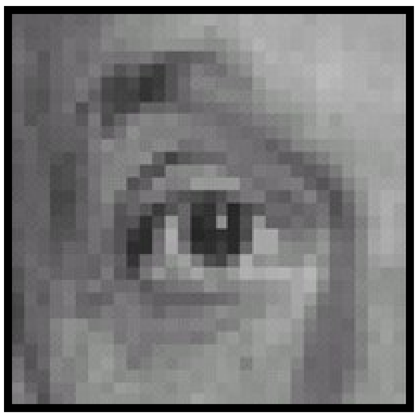
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Source: D. Lowe

2. Practice with linear filters



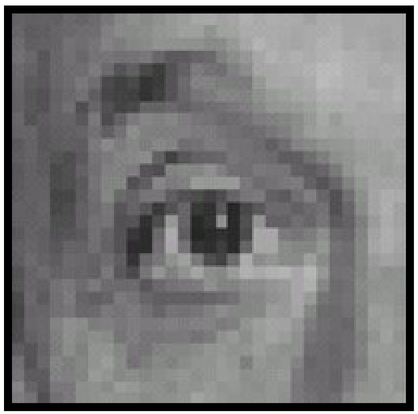
Original

0	0	0
0	0	1
0	0	0

?

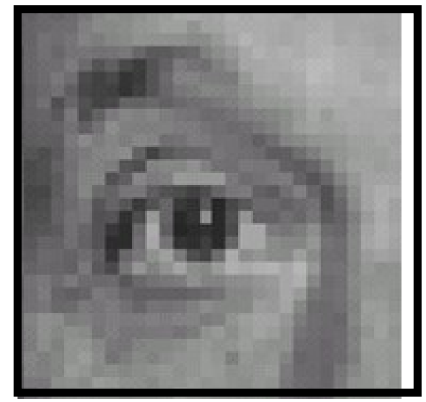
Source: D. Lowe

2. Practice with linear filters



Original

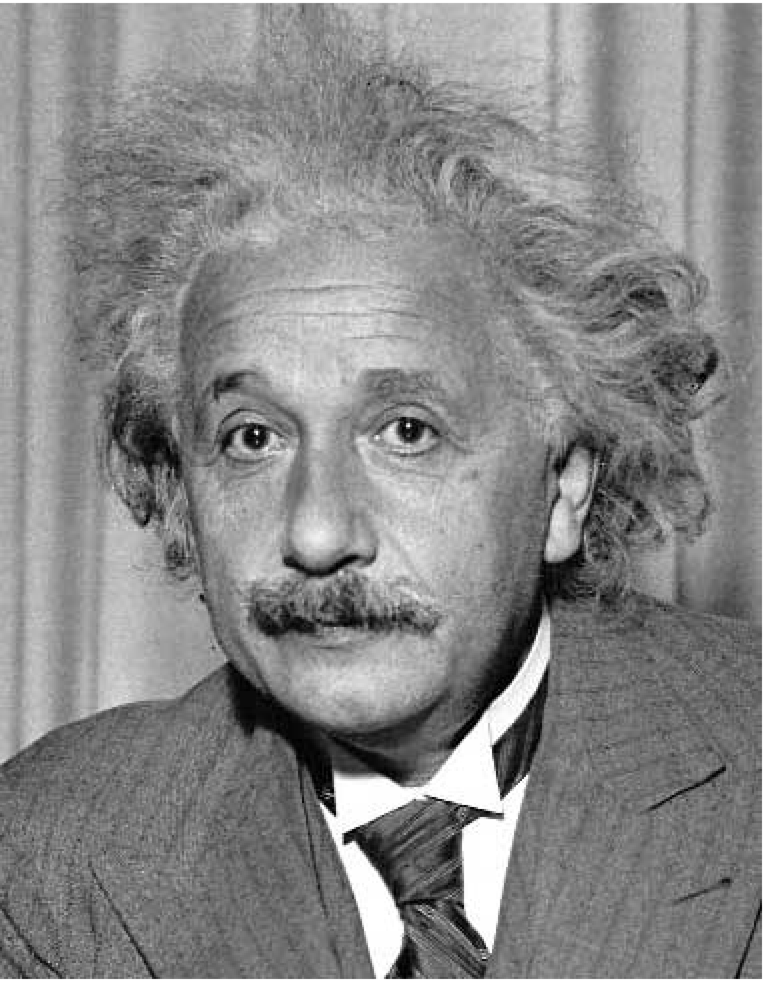
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

3. Practice with linear filters



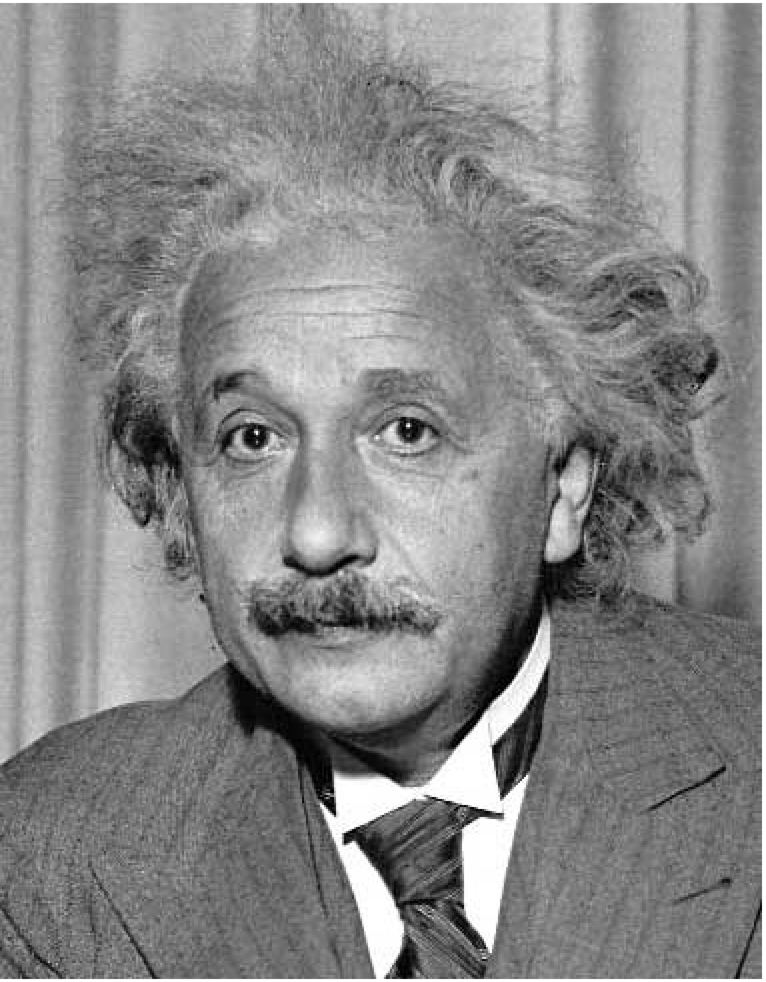
1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

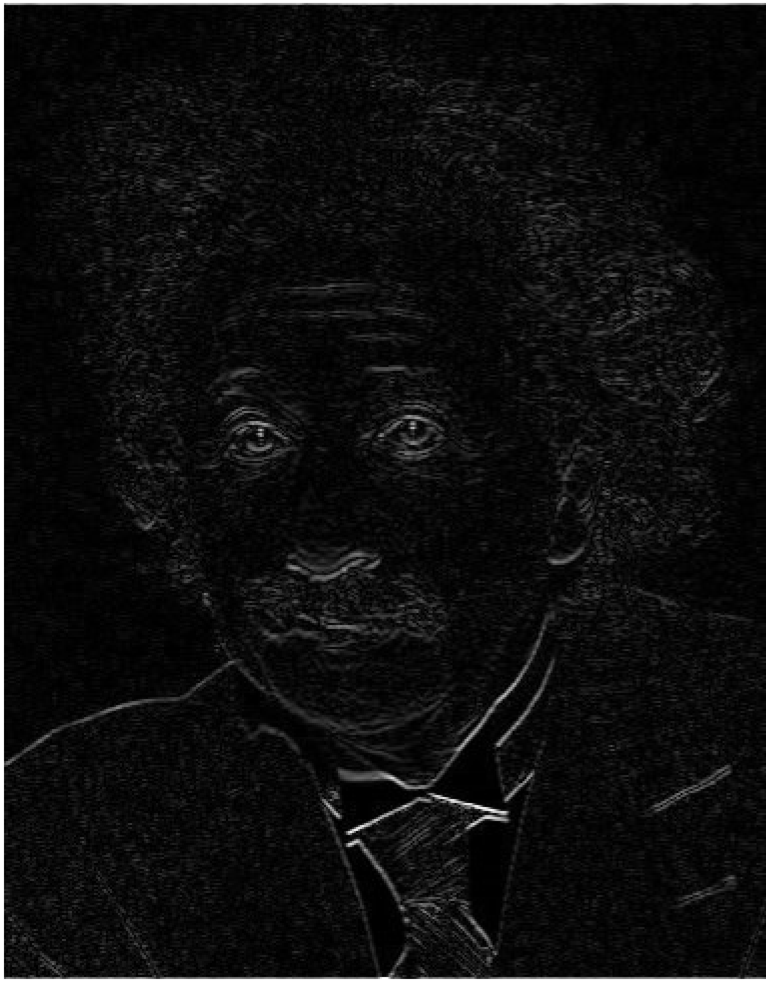
David Lowe

3. Practice with linear filters



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

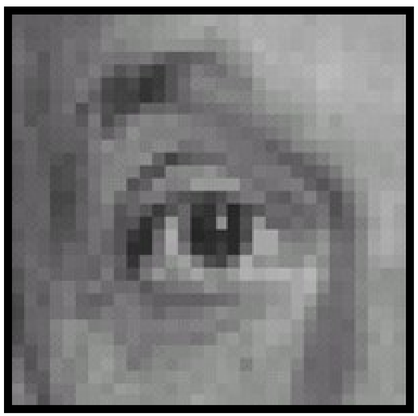
Sobel



Horizontal Edge
(absolute value)

David Lowe

4. Practice with linear filters

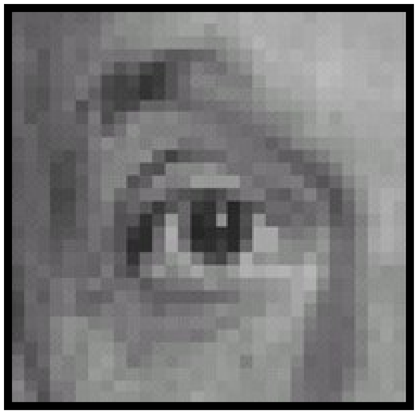


Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

4. Practice with linear filters



Original

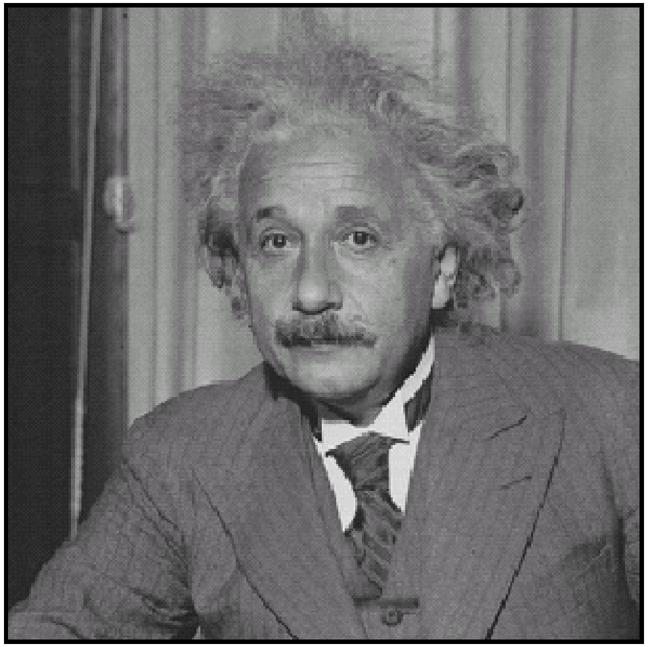
$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

Sharpening filter

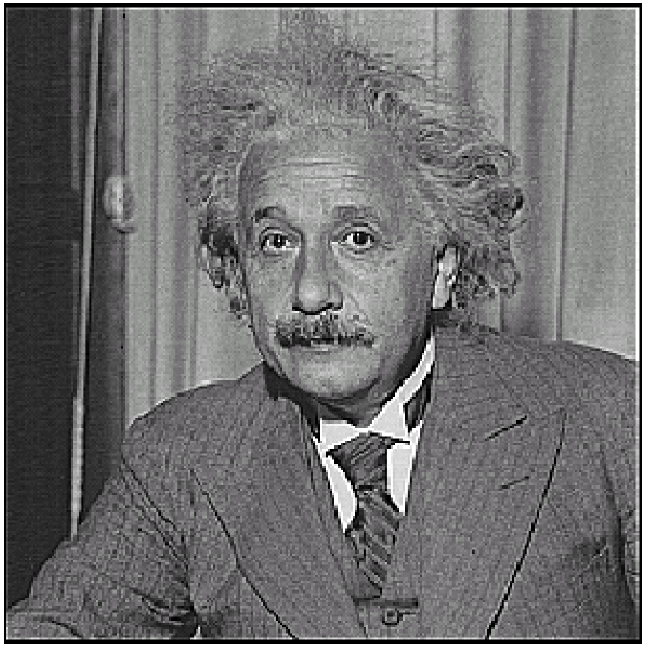
- Accentuates differences with local average

Source: D. Lowe

4. Practice with linear filters



before



after

未来媒体研究中心 CENTER FOR FUTURE MEDIA | 电子科技大学 University of Electronic Science and Technology of China

Source: D. Lowe

Correlation and Convolution

- 2d correlation

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

`h=filter2(f,I);` or `h=imfilter(I,f);`

Correlation and Convolution

- 2d correlation

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

`h=filter2(f,I);` or `h=imfilter(I,f);`

- 2d convolution

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

`h=conv2(f,I); or h=imfilter(I,f,'conv');`

`conv2(I,f)` is the same as **`filter2(rot90(f,2),I)`**

Correlation and convolution are identical when the filter is symmetric.

Key properties of linear filters

Linearity:

```
imfilter(I, f1 + f2) =
   imfilter(I,f1) + imfilter(I,f2)
```

$\texttt{imfilter}(I,\ f_1 + f_2) = \texttt{imfilter}(I, f_1) + \texttt{imfilter}(I, f_2)$

Shift invariance:

Same behavior regardless of pixel location

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

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

Source: S. Lazebnik

Convolution properties

Commutative: $a * b = b * a$

- Conceptually no difference between filter and signal
- But particular filtering implementations might break this equality, e.g., image edges

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 applying one filter: a * $(b_1 * b_2 * b_3)$

Source: S. Lazebnik

Convolution properties

Commutative: $a * b = b * a$

- Conceptually no difference between filter and signal
- But particular filtering implementations might break this equality, e.g., image edges

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 applying one filter: a * $(b_1 * b_2 * b_3)$
- Correlation is _not_ associative (rotation effect)

Source: S. Lazebnik

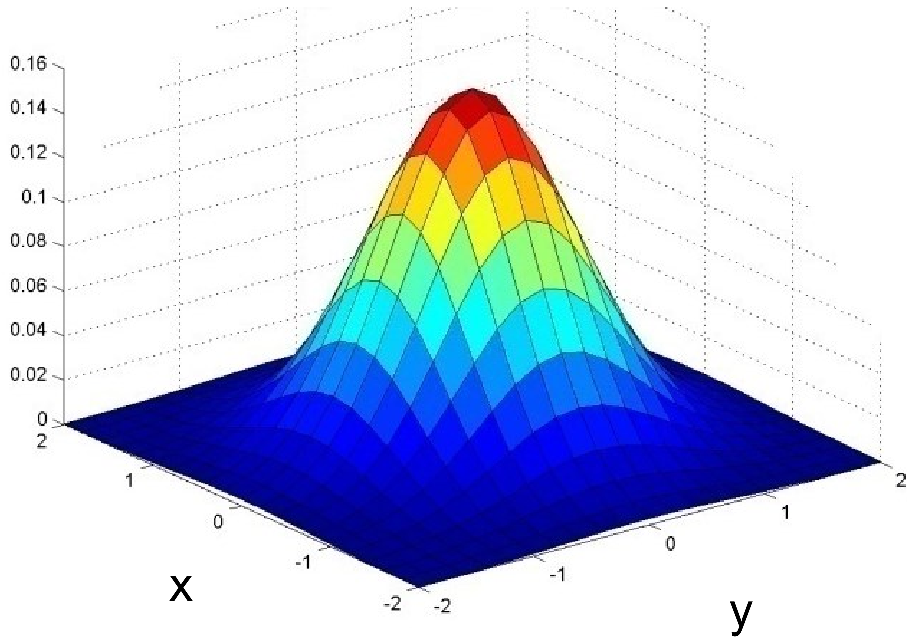
Convolution properties

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal
  - But particular filtering implementations might break this equality, e.g., image edges
- 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 applying one filter: a * $(b_1 * b_2 * b_3)$
  - Correlation is _not_ associative (rotation effect)
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$

Source: S. Lazebnik

Important filter: Gaussian

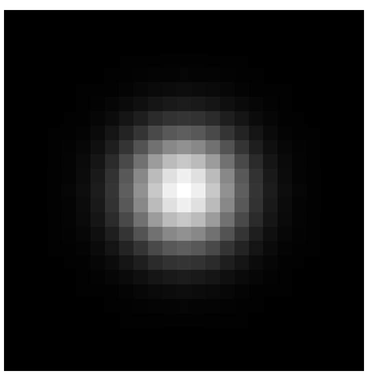
- Weight contributions of neighboring pixels by nearness



x

y	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

5 x 5, σ = 1



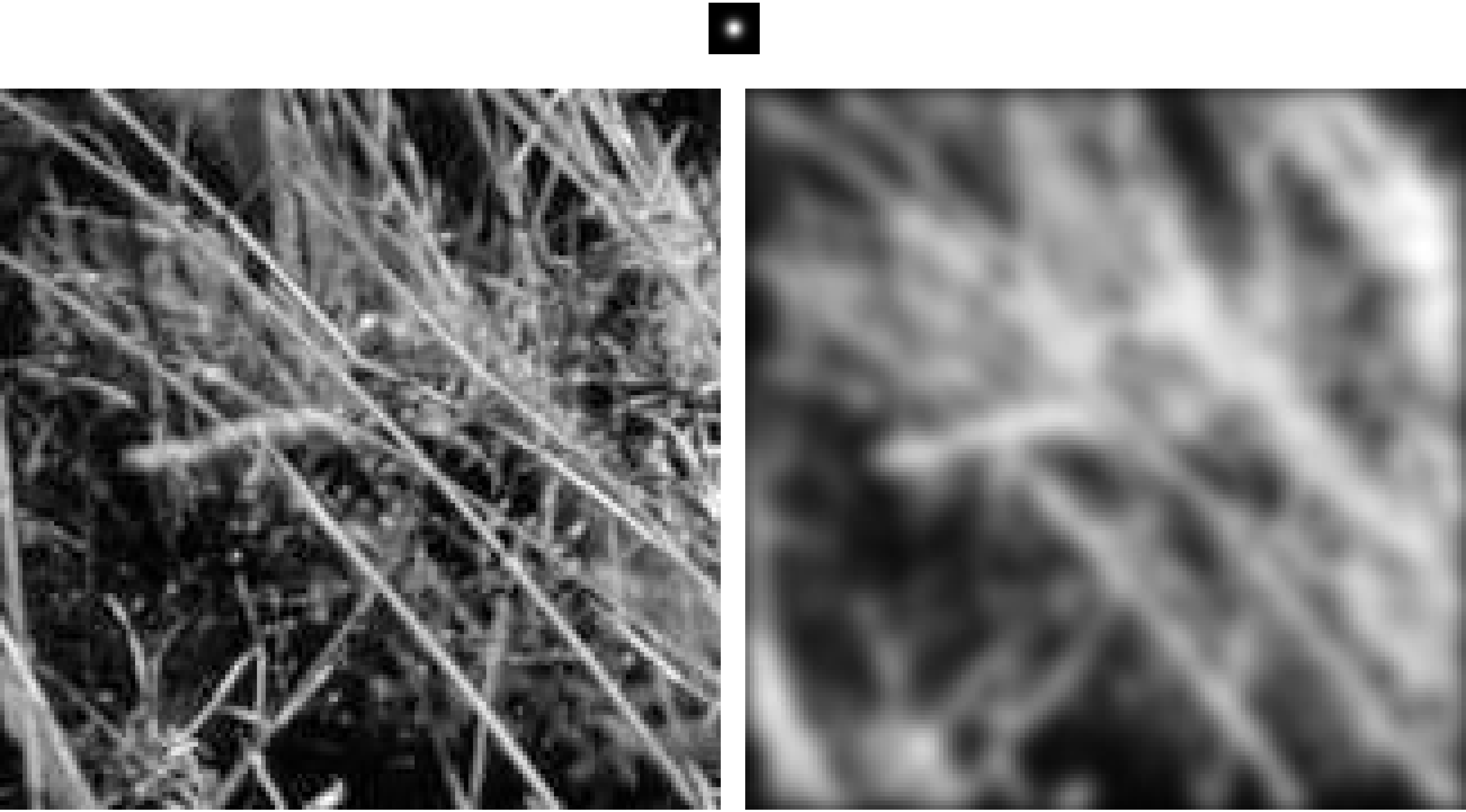
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

未来媒体研究中心 CENTER FOR FUTURE MEDIA | 电子科技大学 University of Electronic Science and Technology of China

Slide credit: Christopher Rasmussen

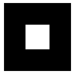
Smoothing with Gaussian filter

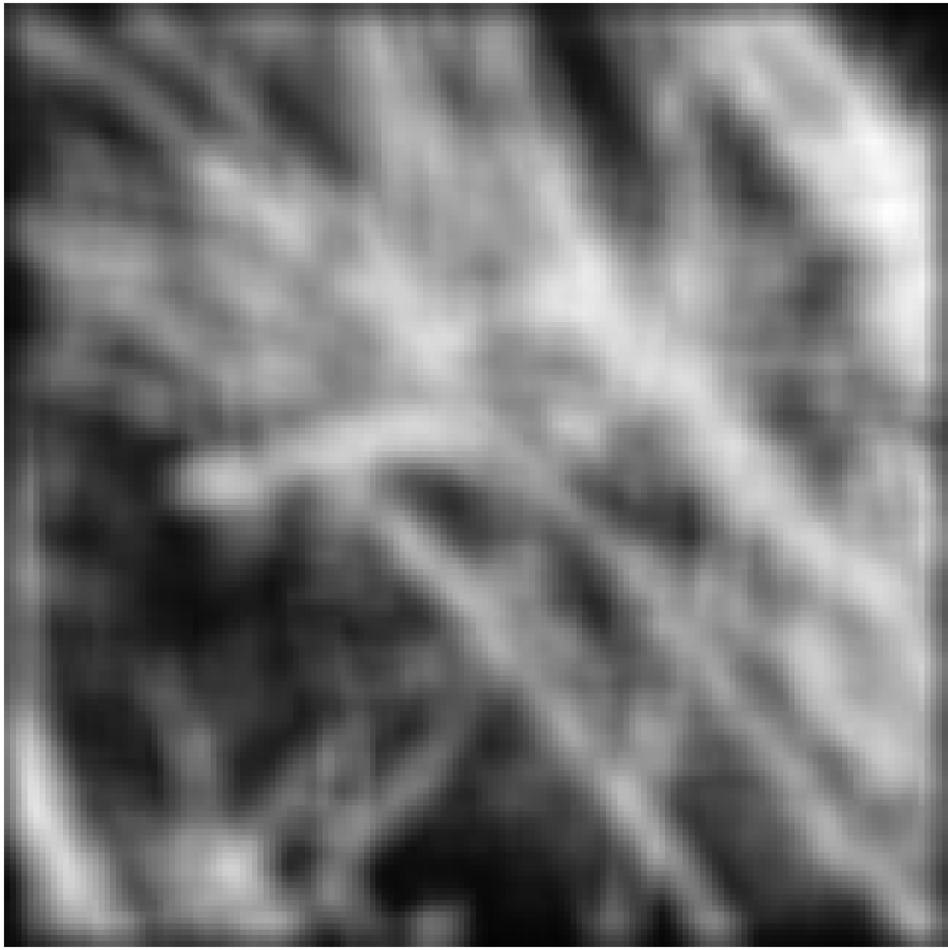
James Hays



James Hays

Smoothing with box filter





Gaussian filters

- Remove "high-frequency" components from the image (low-pass filter)
  - Images become more smooth

- Gaussian convolved with Gaussian…

  …is another Gaussian
  - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have

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

Source: K. Grauman

Separability of the Gaussian filter

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

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution
(center location only)

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{bmatrix} = \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix}$$

Followed by convolution
along the remaining column:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} & 11 & \\ & 18 & \\ & 18 & \end{bmatrix} = \begin{bmatrix} & & \\ & 65 & \\ & & \end{bmatrix}$$

Source: K. Grauman

Separability

Why is separability useful in practice?

Separability

Why is separability useful in practice?

MxN image, PxQ filter

- 2D convolution: ~MNPQ multiply-adds
- Separable 2D: ~MN(P+Q) multiply-adds

Speed up = PQ/(P+Q)

9x9 filter = ~4.5x faster

Summary of Typical Image Filter

- Box Filter (Averaging)

*

0	0.2	0
0.2	0	0.2
0	0.2	0

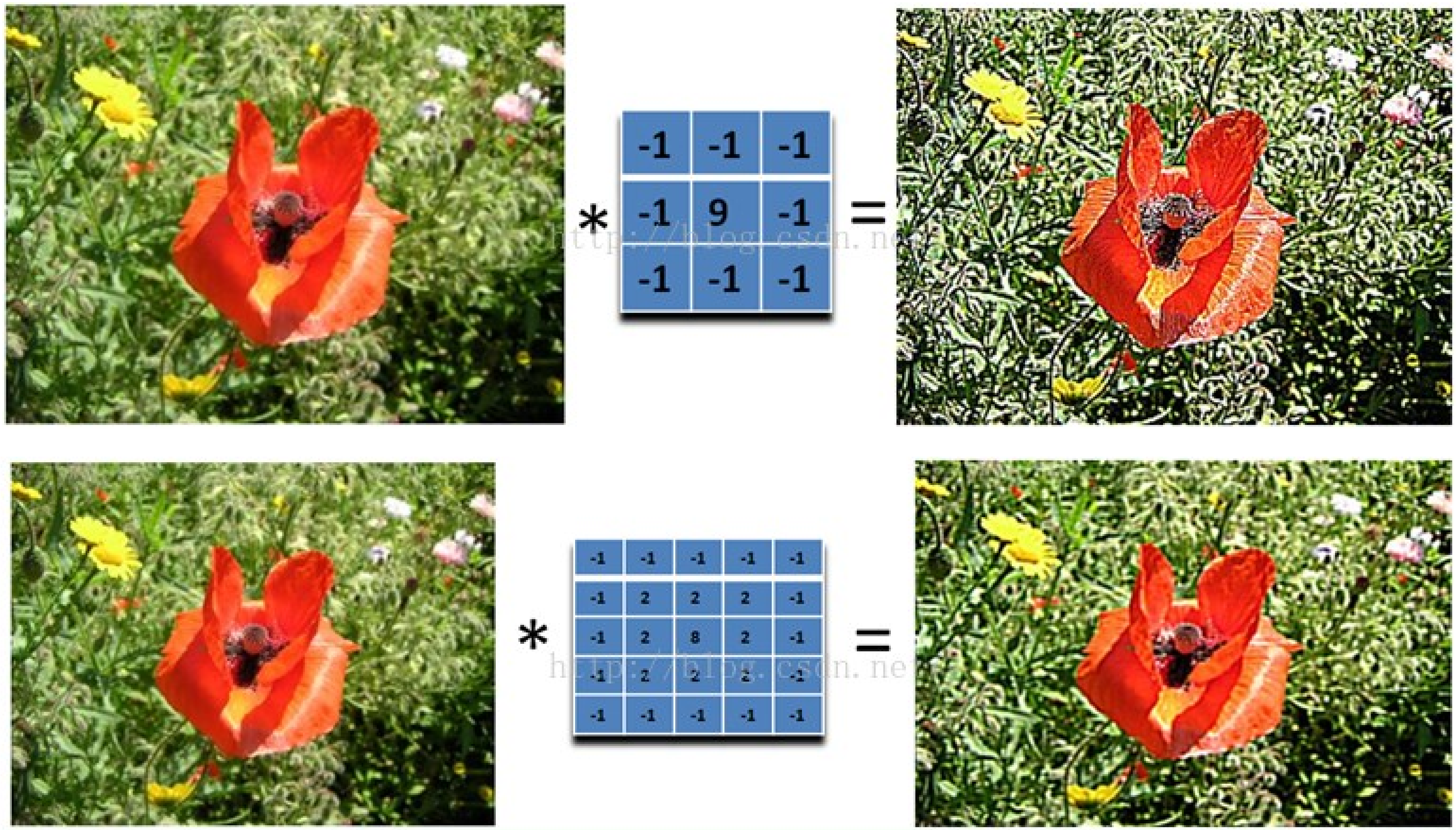
=

*

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

=

- Sharpness Filter



- Sharpness Filter

$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & -7 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}; \begin{bmatrix} -k & -k & -k \\ -k & 8k+1 & -k \\ -k & -k & -k \end{bmatrix}$$

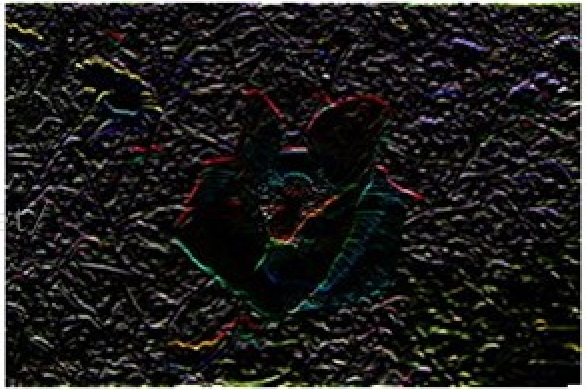
  - The summation of all elements in the filter is 1

- Edge detection Filter

*

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

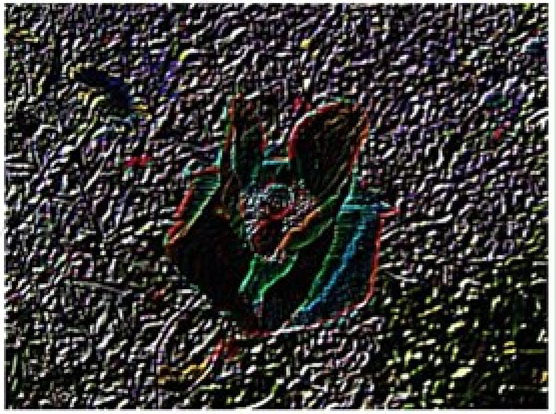
=



*

-1	0	0	0	0
0	-2	0	0	0
0	0	6	0	0
0	0	0	-2	0
0	0	0	0	-1

=



- Edge detection Filter

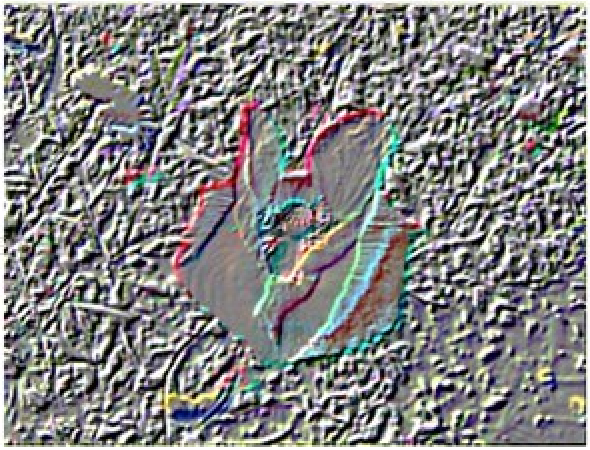
$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$

$$\begin{bmatrix} -1/8 & -1/8 & -1/8 \\ -1/8 & 1 & -1/8 \\ -1/8 & -1/8 & -1/8 \end{bmatrix}; \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix};$$

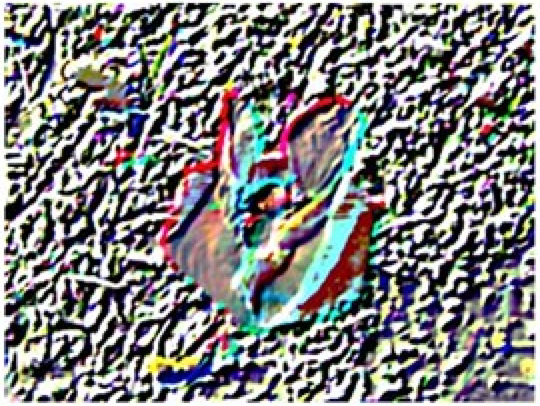
– The summation of all elements in the filter is 0

- Embossing Filter

$$* \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} =$$

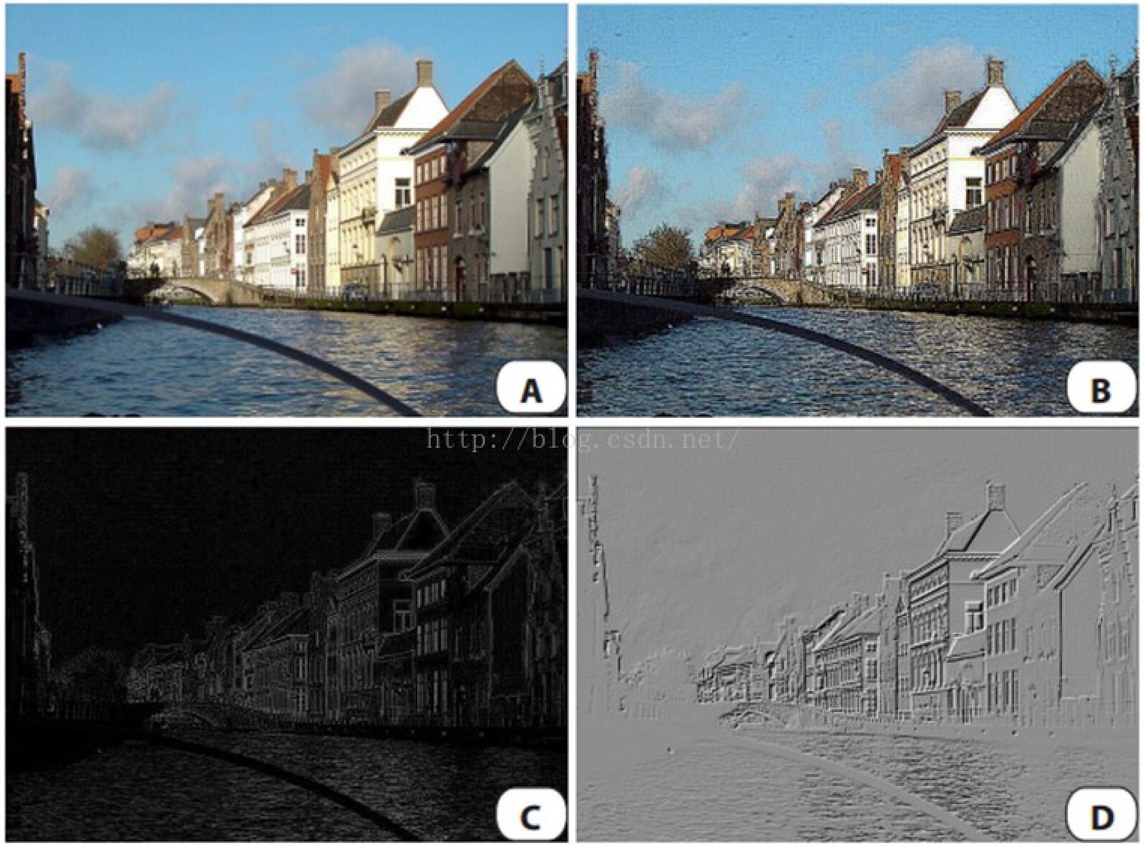


$$* \begin{bmatrix} -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 0 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} =$$



$$\begin{bmatrix} 2 & -0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

- Embossing Filter



  – The filter is unsymmetrical (directional)

- Motion Blur Filter

```
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, 0
0, 0, 0, 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, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 1, 0
0, 0, 0, 0, 0, 0, 0, 0, 1
```


  - Directional

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}; G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Gaussian Blur Filter

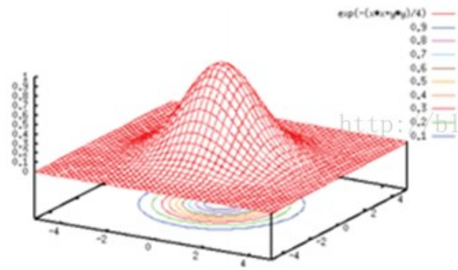


Figure 4: *The 2D Gaussian function.*

$$K_{G5} = \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Take home message for Image Filter

- 滤波器的大小应该是奇数，这样它才有一个中心，例如3x3，5x5或者7x7。有中心了，也有了半径的称呼，例如5x5大小的核的半径就是2。
- 滤波器矩阵所有的元素之和应该要等于1，这是为了保证滤波前后图像的亮度保持不变。当然了，这不是硬性要求。
- 如果滤波器矩阵所有元素之和大于1，那么滤波后的图像就会比原图像更亮，反之，如果小于1，那么得到的图像就会变暗。如果和为0，图像不会变黑，但也会非常暗。
- 对于滤波后的结构，可能会出现负数或者大于255的数值。对这种情况，我们将他们直接截断到0和255之间即可。对于负数，也可以取绝对值。

Take home message for Image Filter

- Correlation（协相关） 和 Convolution（卷积）是图像处理最基本的操作，但却非常有用。这两个操作有两个非常关键的特点：它们是线性的，而且具有平移不变性。

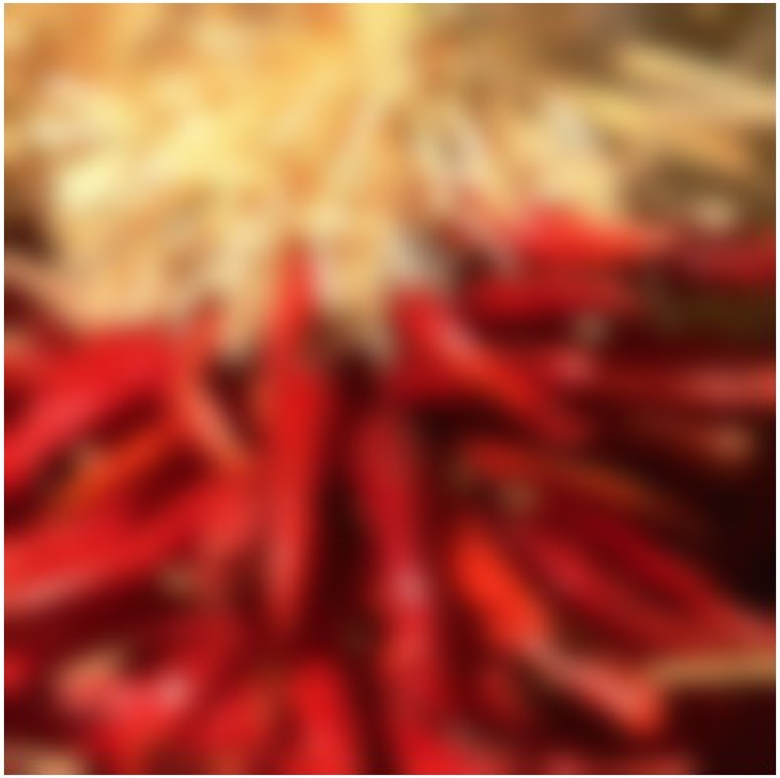
- 积和协相关的差别是，卷积需要先对滤波矩阵进行180的翻转，但如果矩阵是对称的，那么两者就没有什么差别。

- 平移不变性指我们在图像的每个位置都执行相同的操作。线性指这个操作是线性的，也就是我们用每个像素的邻域的线性组合来代替这个像素。

- 2D卷积需要4个嵌套循环4-double loop，所以它并不快。如果卷积核可分离为1D，则可加快卷积计算速度。

Practical matters

- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge



Source: S. Marschner

Convolution in Convolutional Neural Networks

- Convolution is the basic operation in CNNs
- Learning convolution kernels allows us to learn which `features' provide useful information in images.

Next class: Thinking in Frequency