ECE 4973: Lecture 4
Image Filters

Samuel Cheng

Slide credits: Juan Carlos Niebles, Ranjay Krishna, James Hays, Noah Snavely

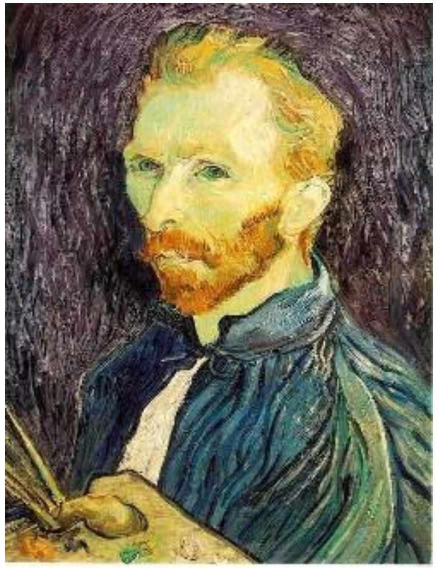
System and Filters

$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

De-noising


Salt and pepper noise

Super-resolution



In-painting

Bertamio et al

Image filtering

- Image filtering:
  - Compute function of local neighborhood at each position
- Really fundamental and everyone should know
- Handy if you don't need state-of-the-art results
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

James Hays

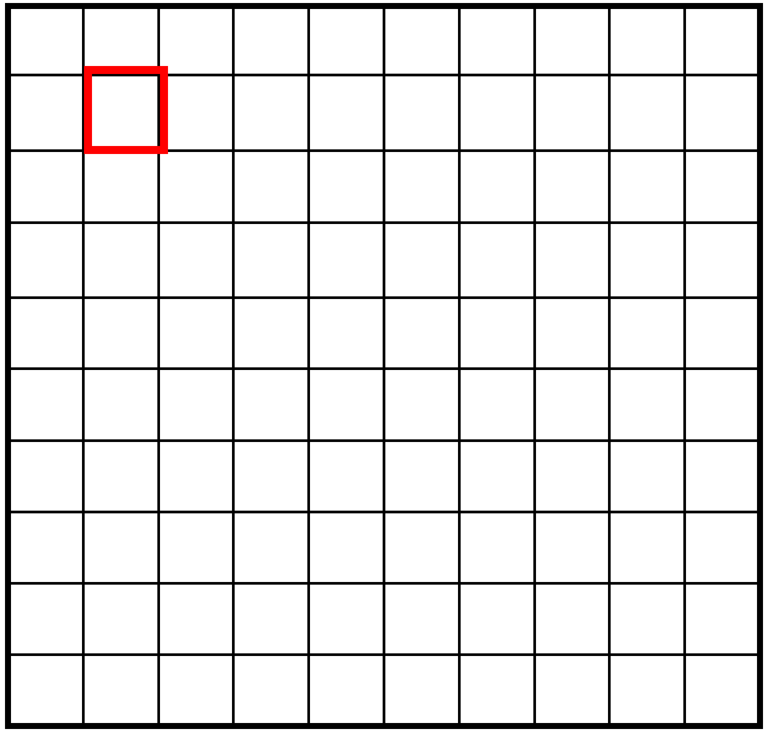
Image filtering

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

$$I[.,.]$$

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

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

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							

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

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						

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

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					

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

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				

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

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				
						?			
				50					

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

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	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

What does it do?

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

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

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

Slide credit: David Lowe (UBC)

Box Filter

What does it do?

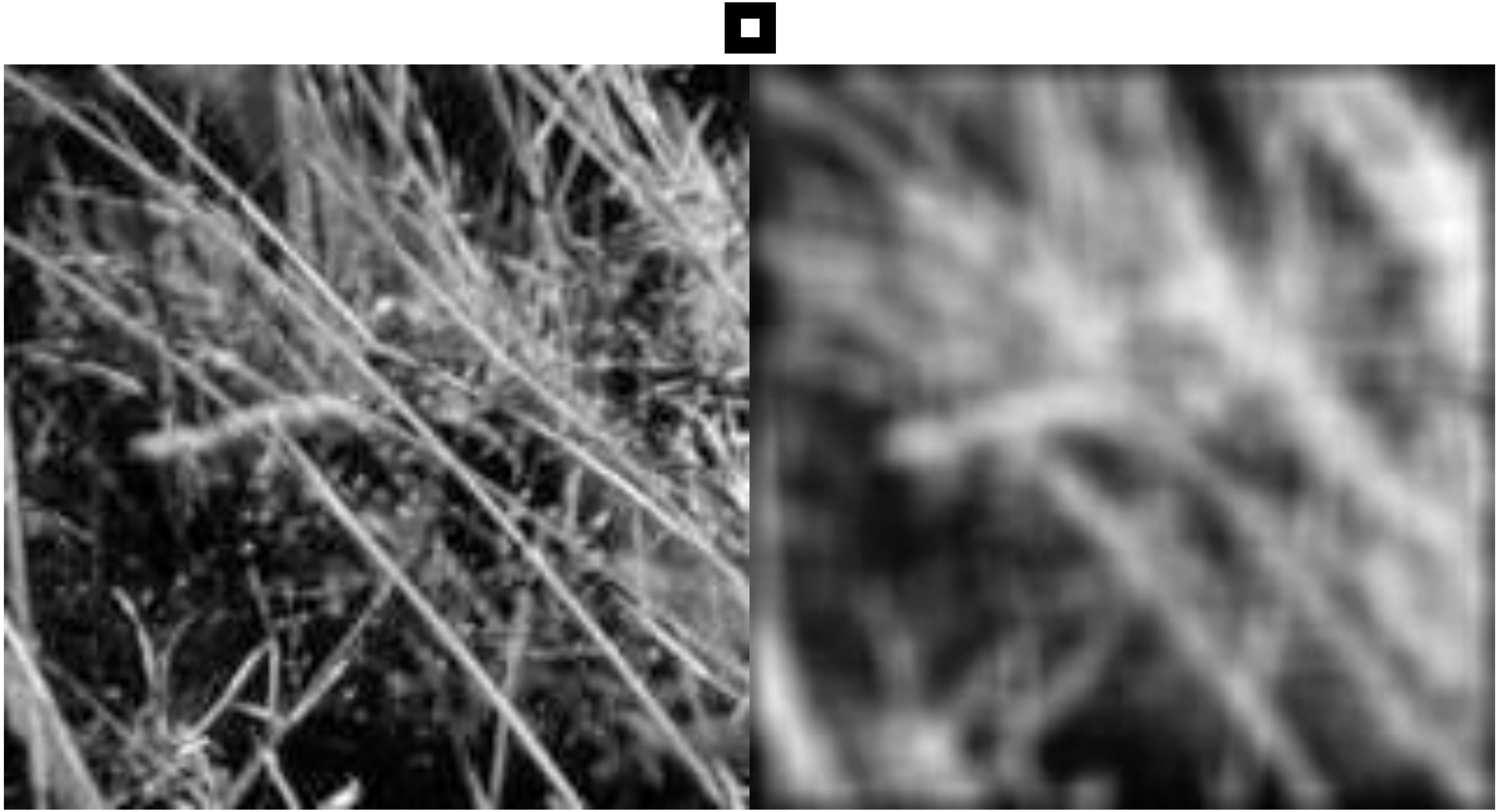
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
- Why does it sum to one?

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

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

Smoothing with box filter

James Hays



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

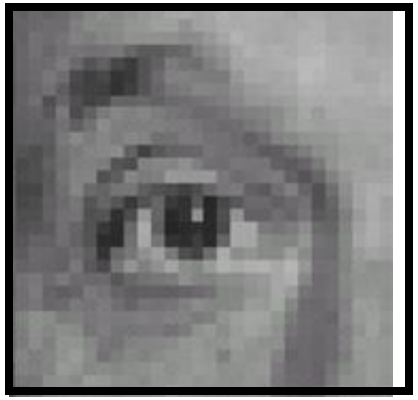
?

Source: D. Lowe

2. Practice with linear filters

Original

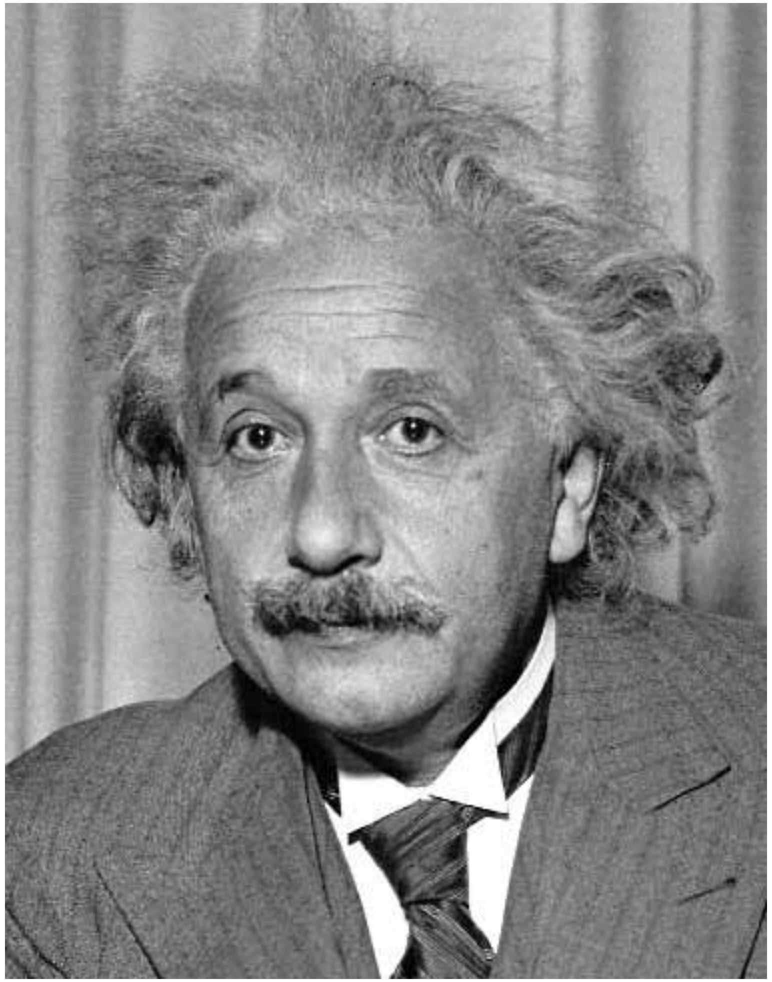
0	0	0
0	0	1
0	0	0



Shifted left
By 1 pixel

Source: D. Lowe

3. Practice with linear filters

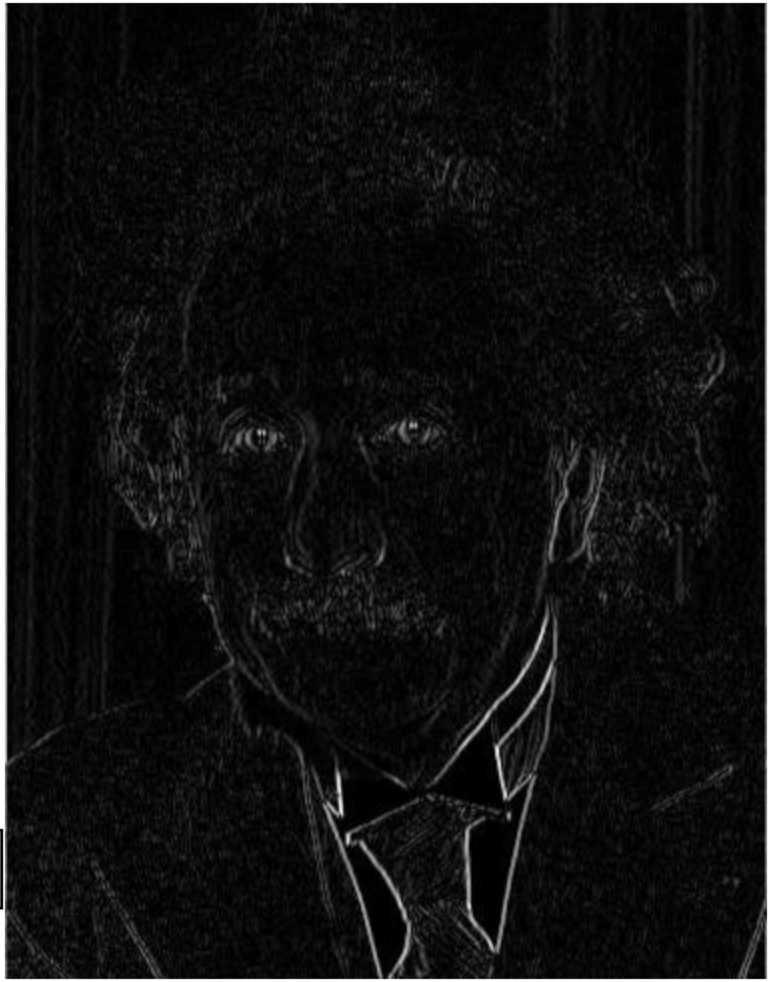


Sobel

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

=

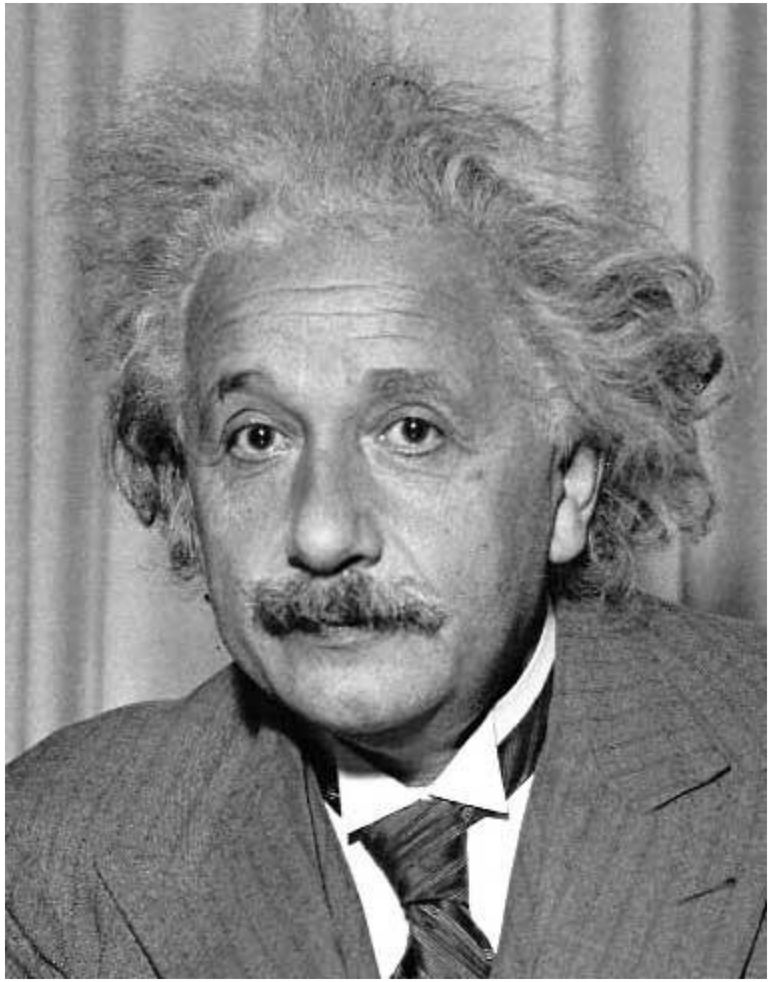
$$\begin{bmatrix} \mathbf{1} \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{-1} \end{bmatrix}$$



Vertical Edge
(absolute value)

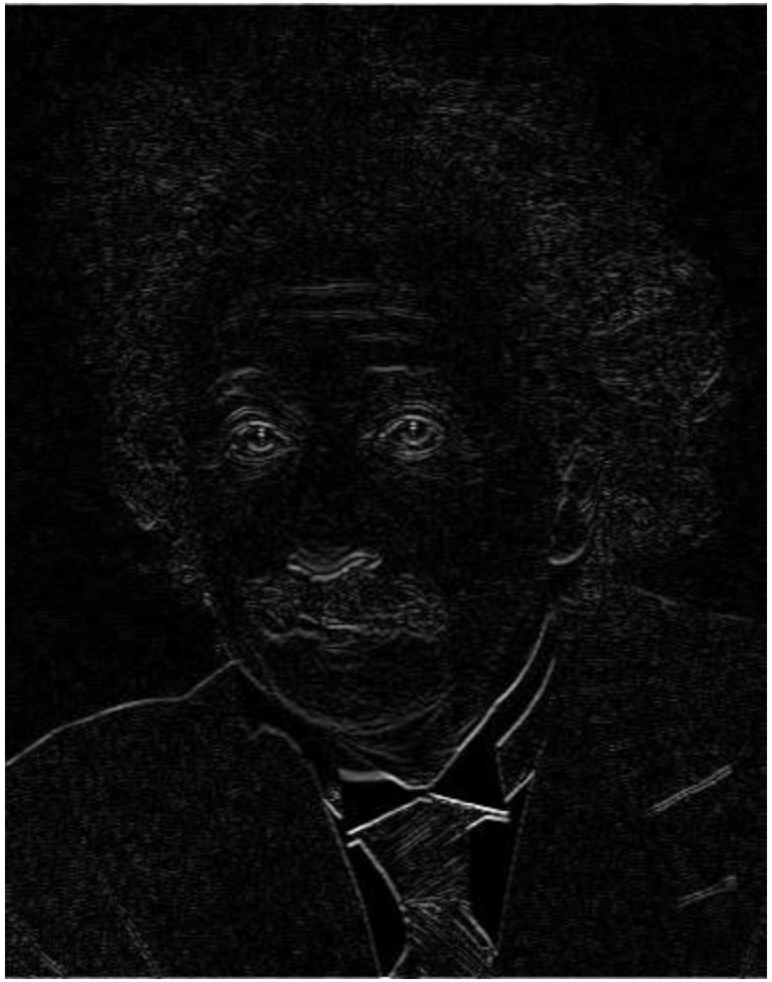
David Lowe

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

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

?

(Note that filter sums to 1)

Source: D. Lowe

4. Practice with linear filters

Original

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

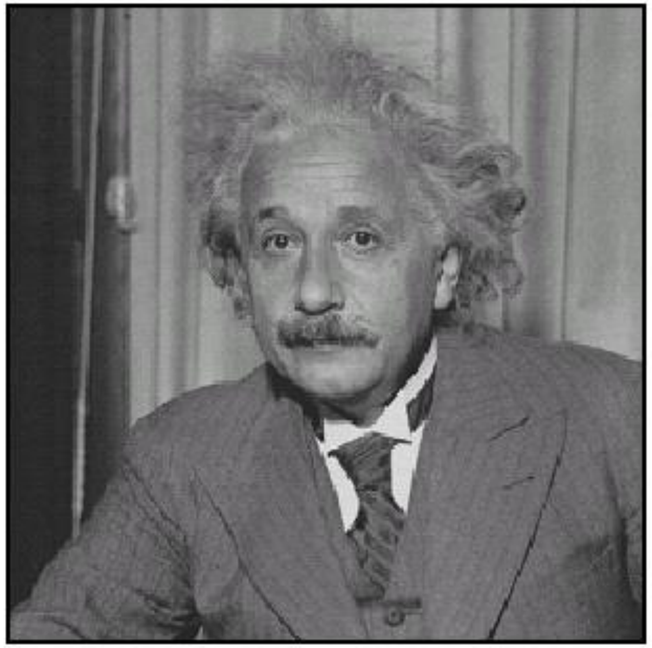
Sharpening filter

- Accentuates differences with local average

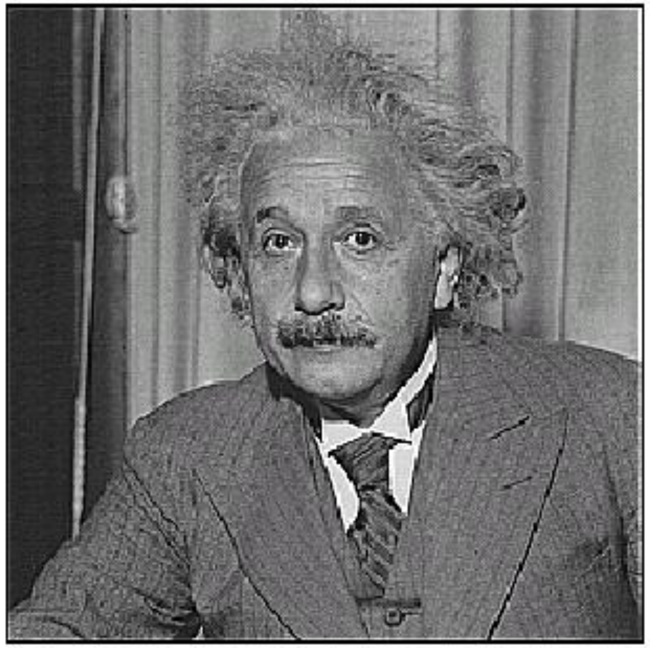
Aka unsharp masking

Source: D. Lowe

4. Practice with linear filters



before



after

Source: D. Lowe

Two important properties of systems

- Shift invariance (same filter/rule throughout the image)

$$f[n-n_0, m-m_0] \xrightarrow{\mathcal{S}} g[n-n_0, m-m_0]$$

```
Equivalently: S(shift(I),f) = shift(S(I,f))
```

- Linearity

$$S[\alpha f_i[n,m] + \beta f_j[n,m]] = \alpha S[f_i[n,m]] + \beta S[f_j[n,m]]$$

Is the moving average system is shift invariant?

$$f[n,m] \quad \xrightarrow{\mathcal{S}} \quad g[n,m] = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k, m-l]$$

$$f[n-n_0, m-m_0]$$

$$\xrightarrow{\mathcal{S}} \quad \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[(n-n_0)-k, (m-m_0)-l]$$

$$= \quad g[n-n_0, m-m_0]$$

Yes!

Linear Systems (filters)

$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

- Is the moving average a linear system?

Yes!

Filter example #2: Image Segmentation

- Image segmentation based on a simple threshold:

$$g[n,m] = \begin{cases} 255, & f[n,m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$

Simple thresholding

$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

- Is thresholding shift-invariant?

Yes!

- Is thresholding a linear system?
  - f1[n,m] + f2[n,m] > T
  - f1[n,m] < T
  - f2[n,m]<T

$$S\big[f1[n,m] + f2[n,m]\big] = 1$$

$$S\big[f1[n,m]\big] + S\big[f2[n,m]\big] = 0$$

No!

Correlation (we are doing so far)

Let F be the image, H be the kernel (filter), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called a **(cross-)correlation** operation:

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

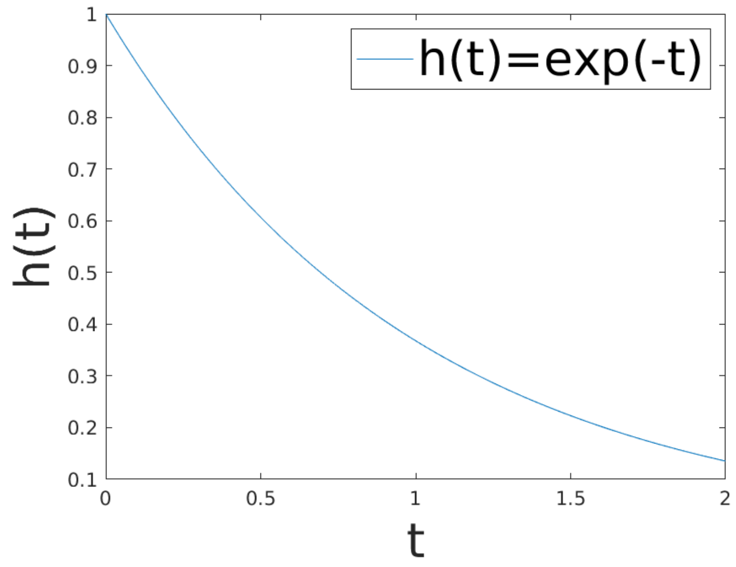
$$= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H^{flip}[-u,-v]F[i-u,j-v]$$

$$= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H^{flip}[u,v]F[i+u,j+v] = H^{flip} \otimes F$$

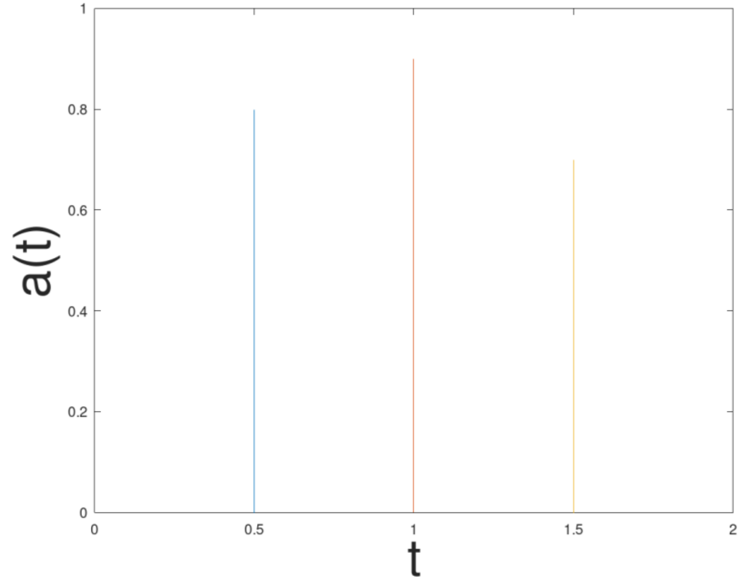
This is called a **convolution** operation:

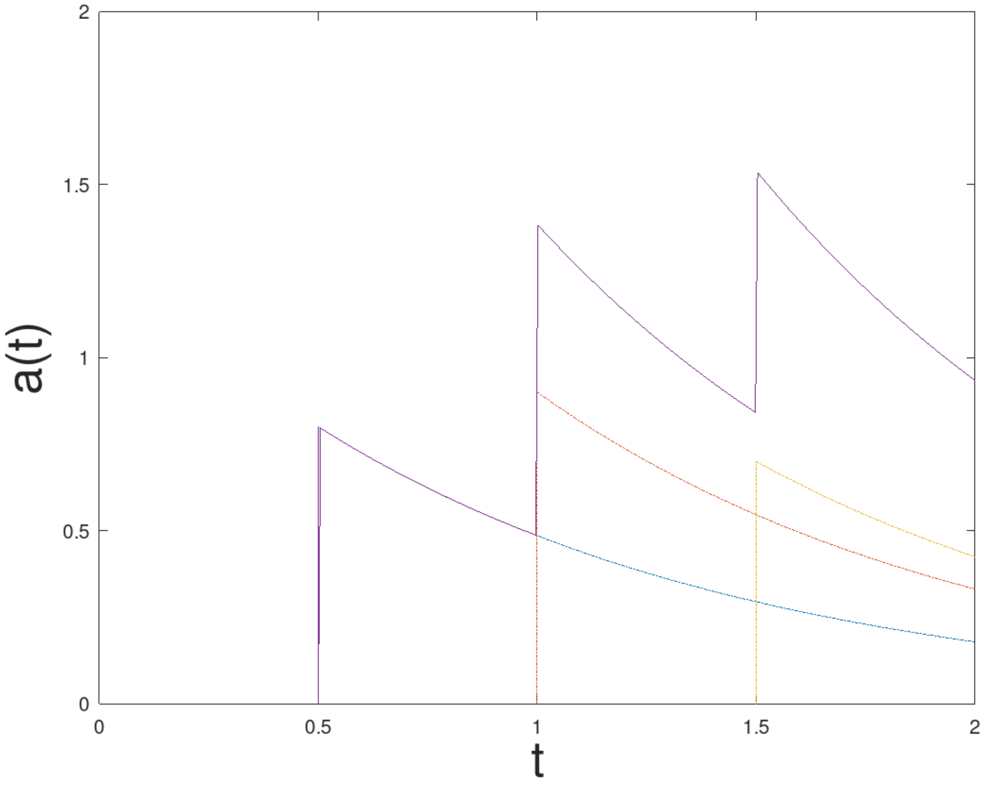
$$G = H * F$$

Where is convolution coming from?



1-D: $y[t] = \sum_{\tau} a[\tau]h[t-\tau]$





Why do mathematicians and signal processing researchers like convolution?

Any **linear and shift-invariant** operator can be represented as a convolution (and specified by its impulse response)

Source: S. Lazebnik

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 applying one filter: a * $(b_1 * b_2 * b_3)$
  - Correlation is NOT associative

$$a \otimes (b \otimes c) = a \otimes (b_{flip} * c) = a_{flip} * (b_{flip} * c)$$

$$(a \otimes b) \otimes c = (a_{flip} * b) \otimes c = (a_{flip} * b)_{flip} * c$$

Source: S. Lazebnik

What is $(a * b)_{flip}$?

- $(a * b)_{flip} = (\sum_i a[i]b[n-i])_{flip}$

$$= \sum_i a[i]b[-n-i]$$

$$= \sum_i a_{flip}[-i]b_{flip}[n+i]$$

$$= \sum_j a_{flip}[j]b_{flip}[n-j] \quad (j = -i)$$

$$= a_{flip} * b_{flip}$$

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 applying one filter: $a * (b_1 * b_2 * b_3)$
  - Correlation is NOT associative

$$a \otimes (b \otimes c) = a \otimes (b_{flip} * c) = a_{flip} * (b_{flip} * c)$$

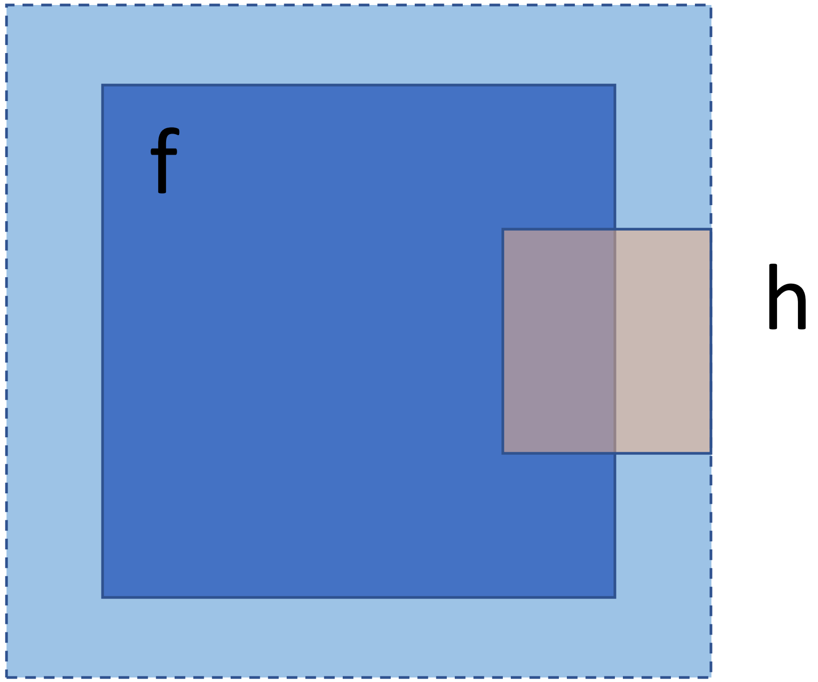
$$(a \otimes b) \otimes c = (a_{flip} * b) \otimes c = (a_{flip} * b)_{flip} * c = (a * b_{flip}) * c$$

- 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

Image support and edge effect

- A computer will only convolve **finite support signals.**
- What happens at the edge?



- zero "padding"
- edge value replication
- mirror extension
- more (beyond the scope of this class)

Convolution vs. (Cross) Correlation

- A <u>**convolution**</u> is a filtering operation

- <u>**Correlation**</u> compares the ***similarity*** *of* ***two*** *sets of* ***data***

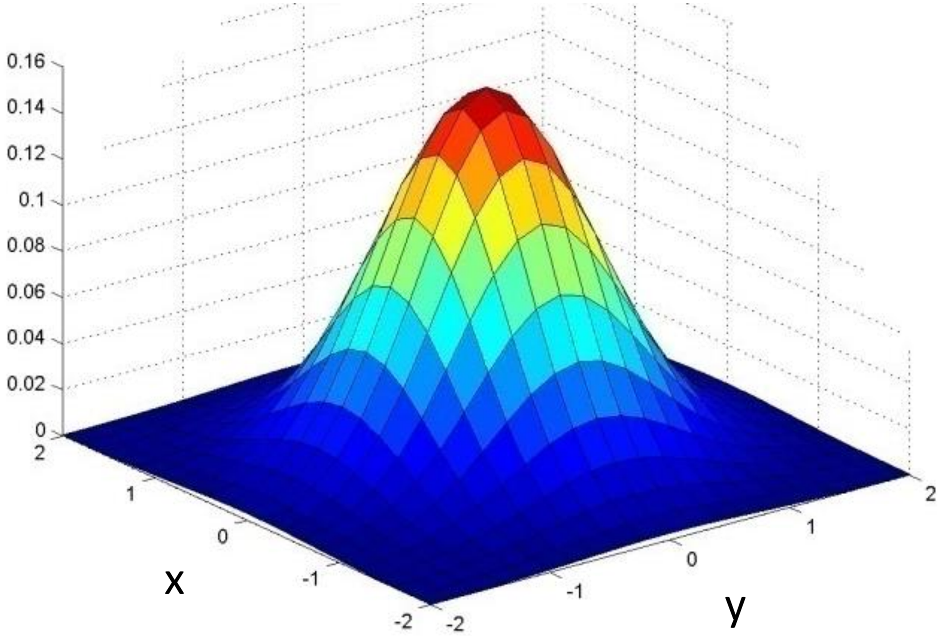
Convolution vs. (Cross) Correlation

	Convolution	Correlation
Associative: (ab)c=a(bc)	Yes	No
Commutative: ab=ba	Yes	No
Distributive: a(b+c)=ab+ac	Yes	Yes
Linear	Yes	Yes
Application	Filtering	Matching

- They are equivalent when the filter “kernel” is symmetric
- N.B. cv2.filter2D implements correlation rather than conv

Important filter: Gaussian

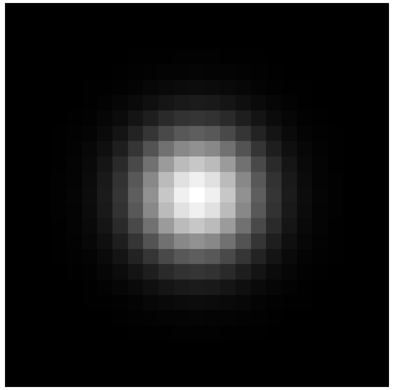
- Weight contributions of neighboring pixels by nearness



x

	0.003	0.013	0.022	0.013	0.003
	0.013	0.059	0.097	0.059	0.013
y	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, $\sigma = 1$

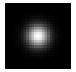


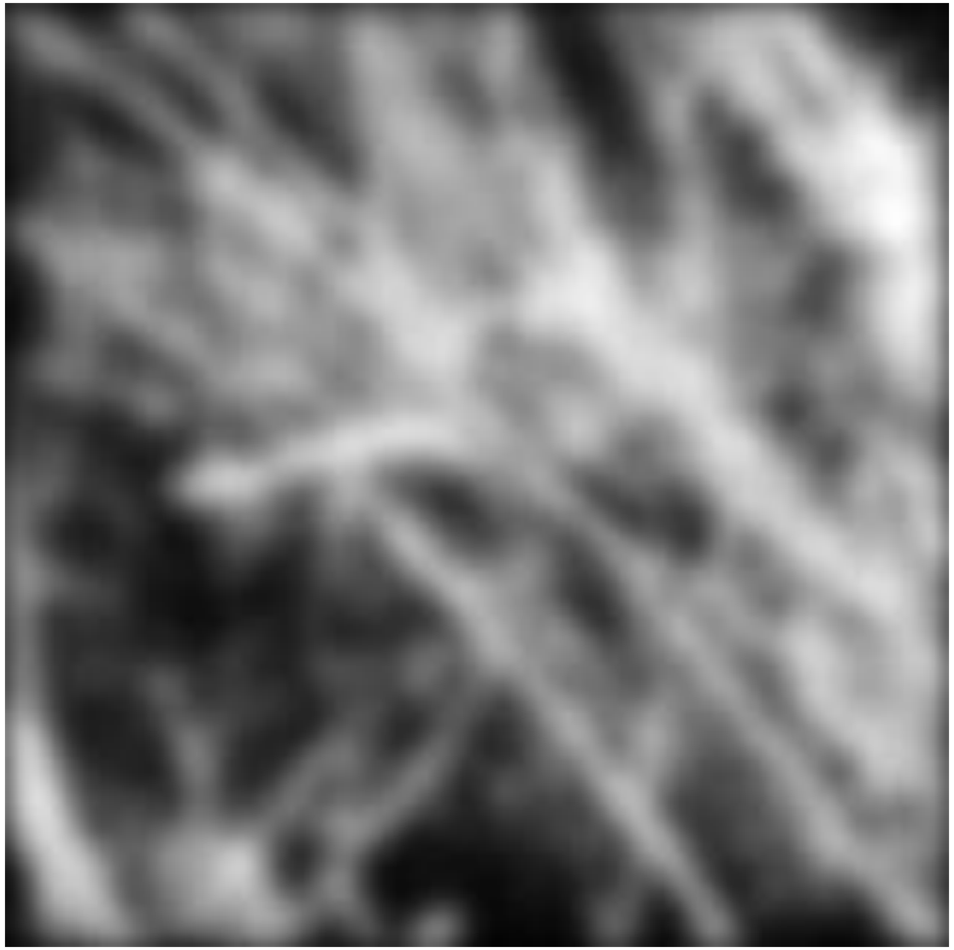
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

James Hays

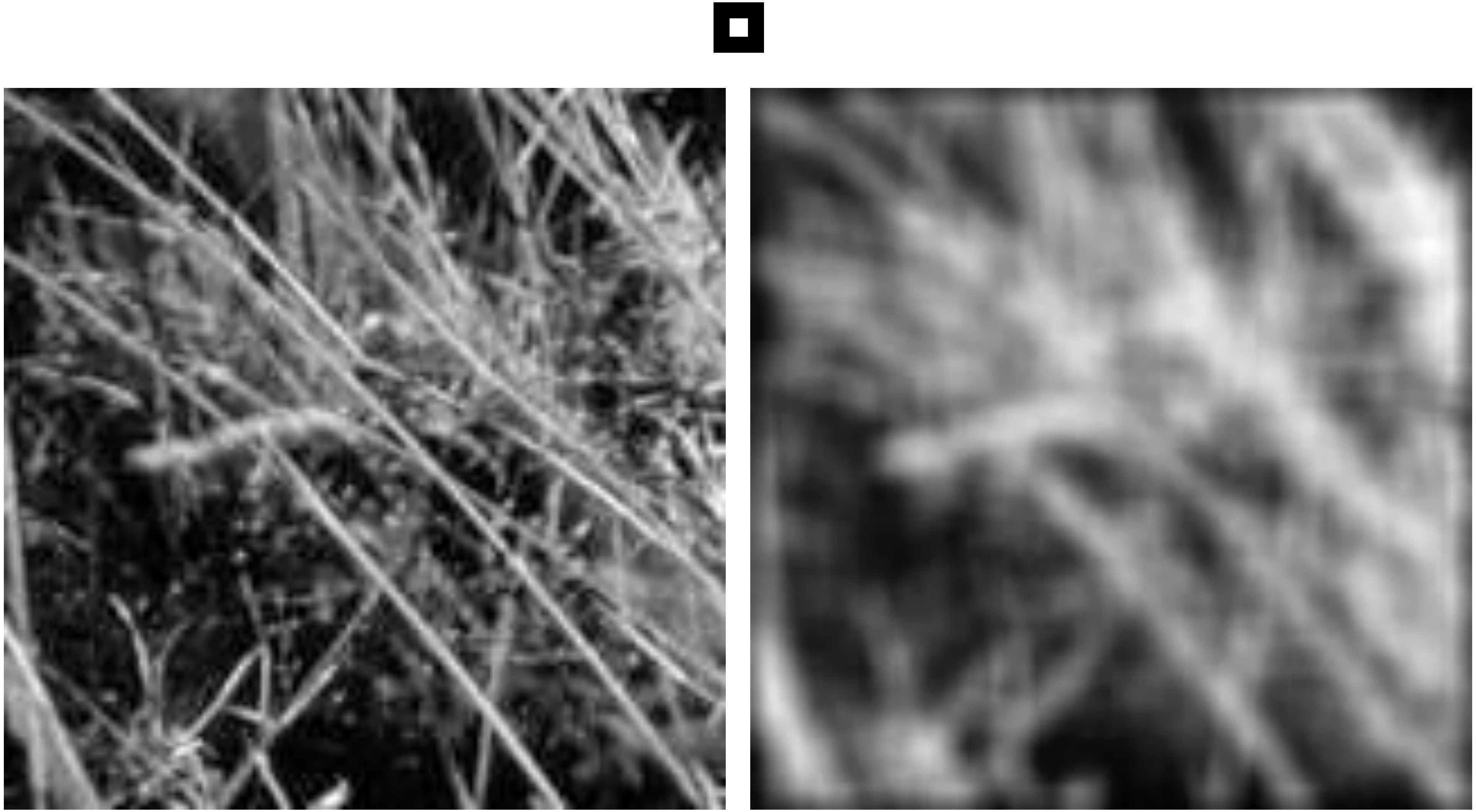
Smoothing with Gaussian filter





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
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *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:

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

Practical matters

How big should the filter be?

- Values at edges should be near zero
- Gaussians have infinite extent…
- Rule of thumb for Gaussian: set filter half-width to about 3 σ

What we have learned today?

- Image histograms
- Linear systems (filters)
- Convolution and correlation
- Gaussian filter
- Median filter

Median filters

- Operates over a window by selecting the median intensity in the window.
- ‘Rank’ filter as based on ordering of gray levels
  - E.G., min, max, range filters

Image filtering - mean

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

$I[.,.]$

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 - mean

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

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

Median filter?

$I[.,.]$

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

$h[.,.]$

				?					

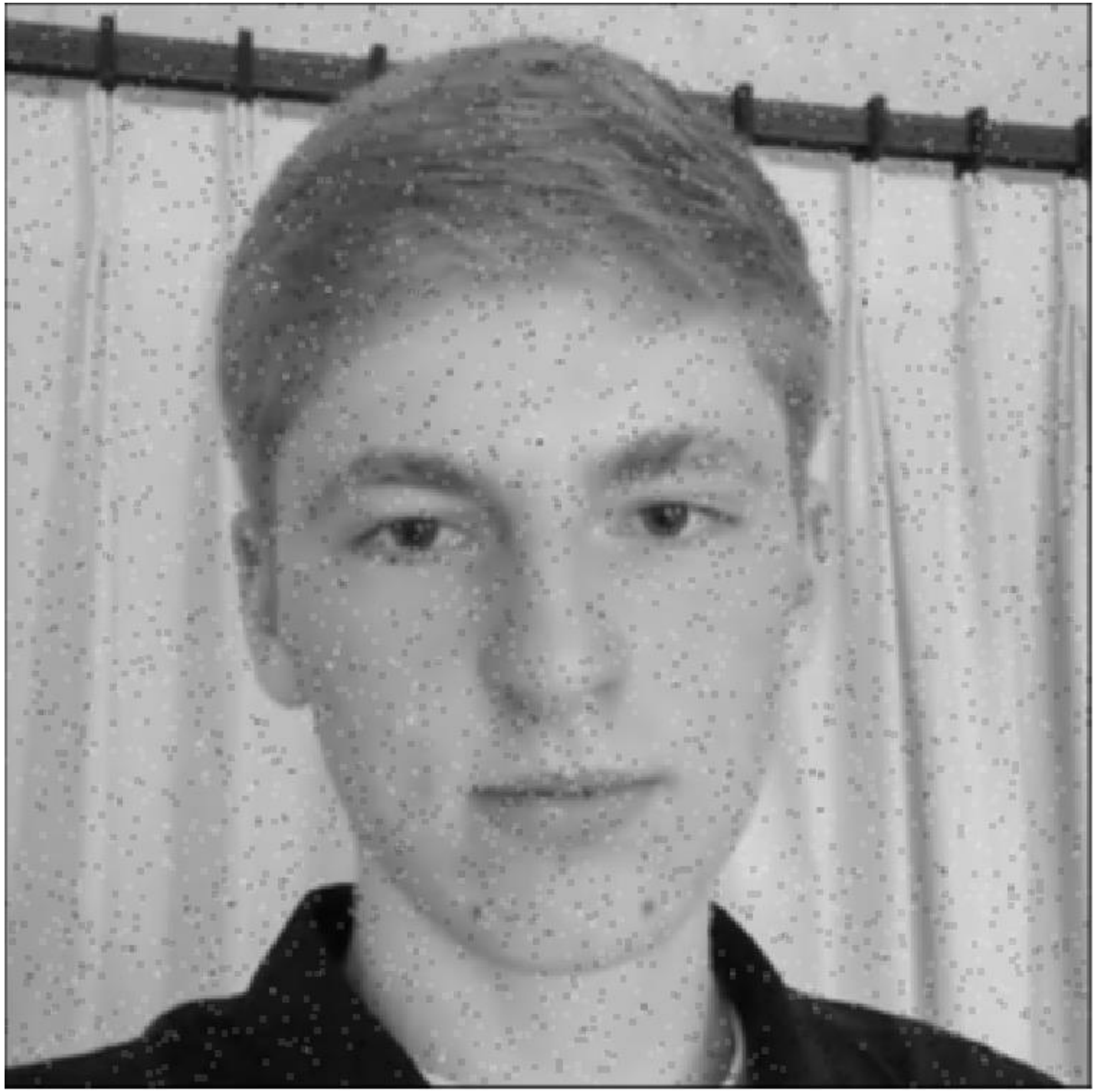
Credit: S. Seitz

Median filters

- Operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?

Noisy Jack – Salt and Pepper

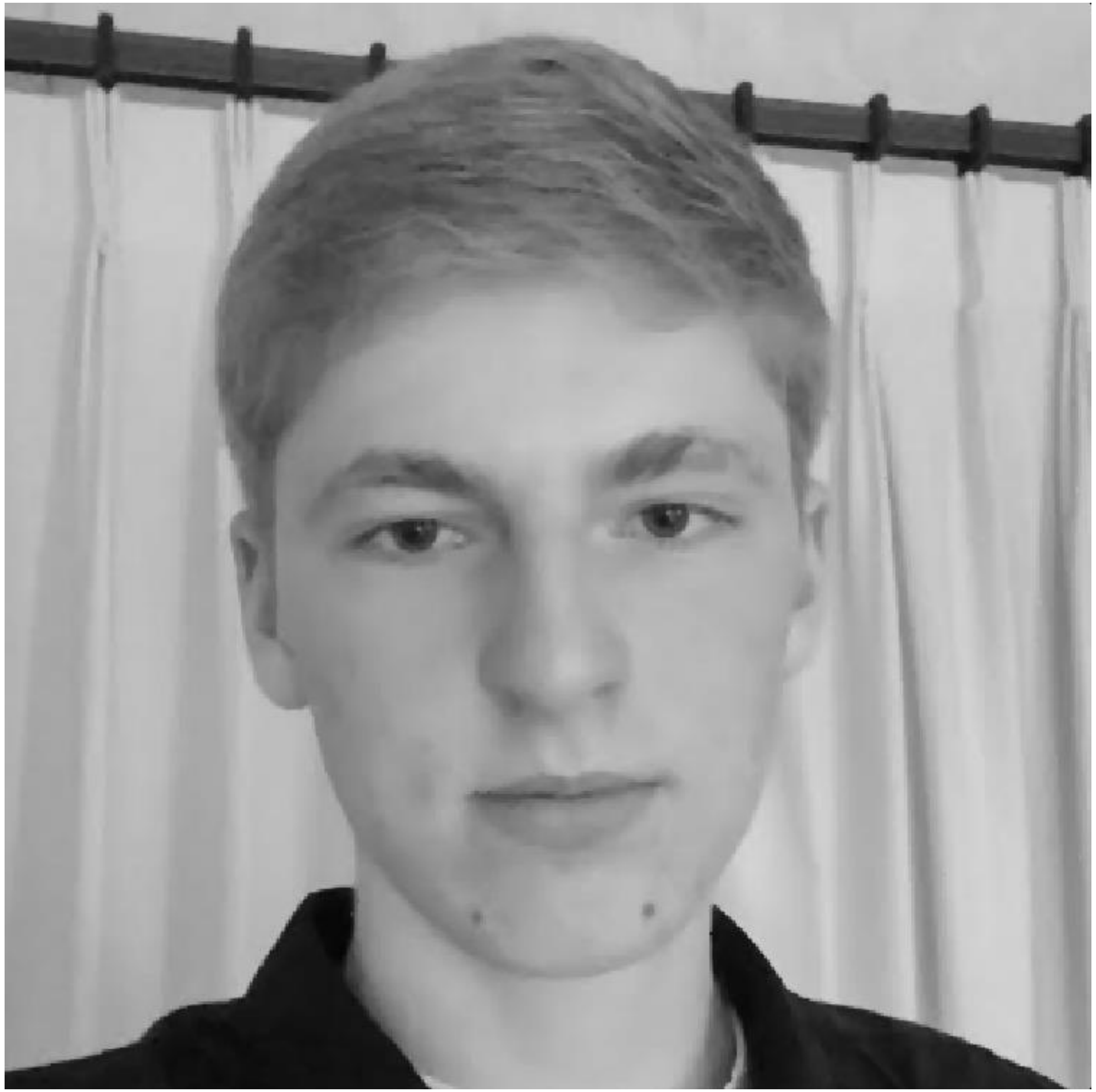
Mean Jack – 3 x 3 filter



Very Mean Jack – 11 x 11 filter

Noisy Jack – Salt and Pepper

Median Jack – 3 x 3



Very Median Jack – 11 x 11

Median filters

- Operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

Secret: Median filtering is sorting.

- Is median filter linear?

Steve Seitz, Steve Marschner

What we have learned today?

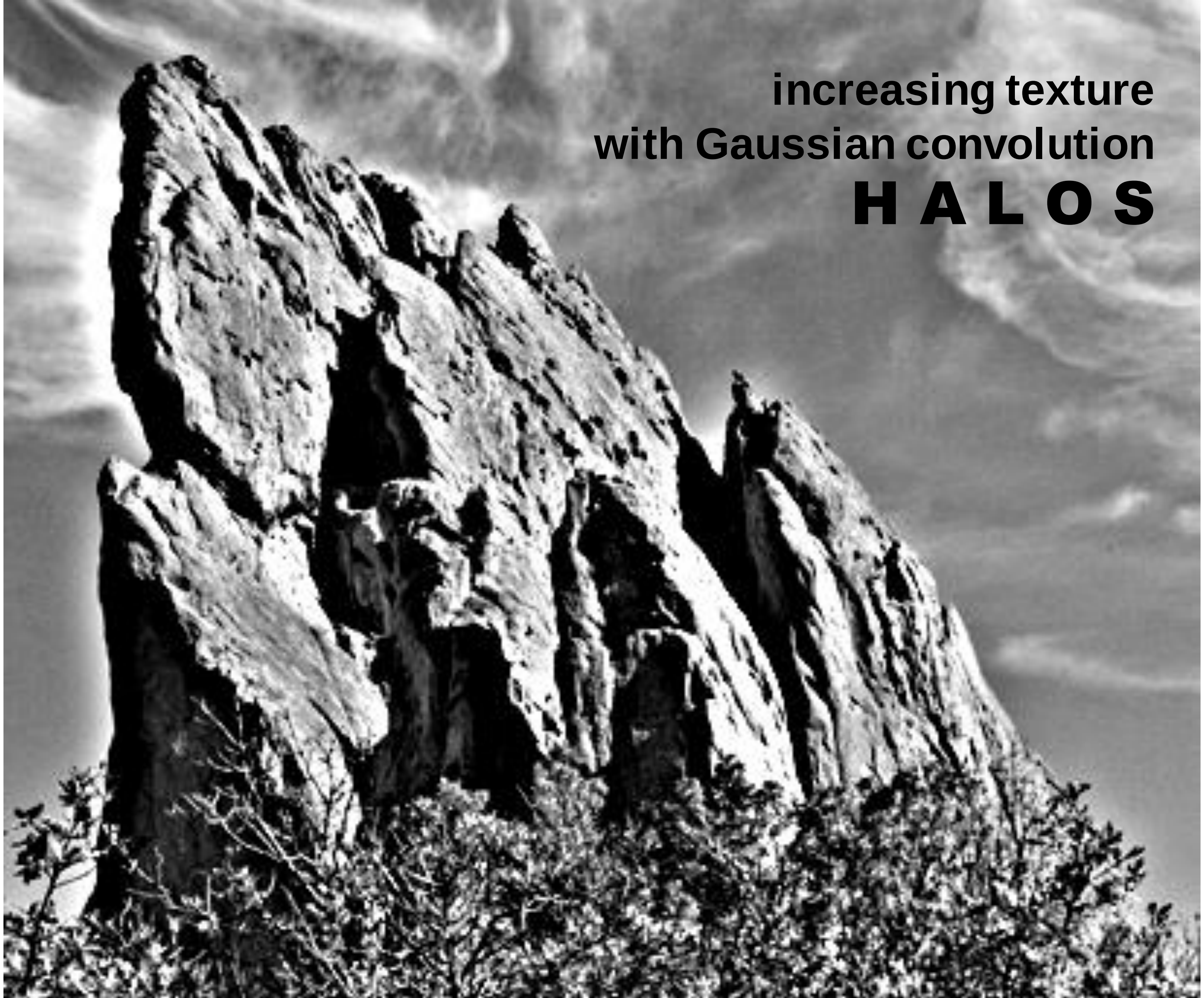
- Image histograms
- Linear systems (filters)
- Convolution and correlation
- Gaussian filter
- Median filter

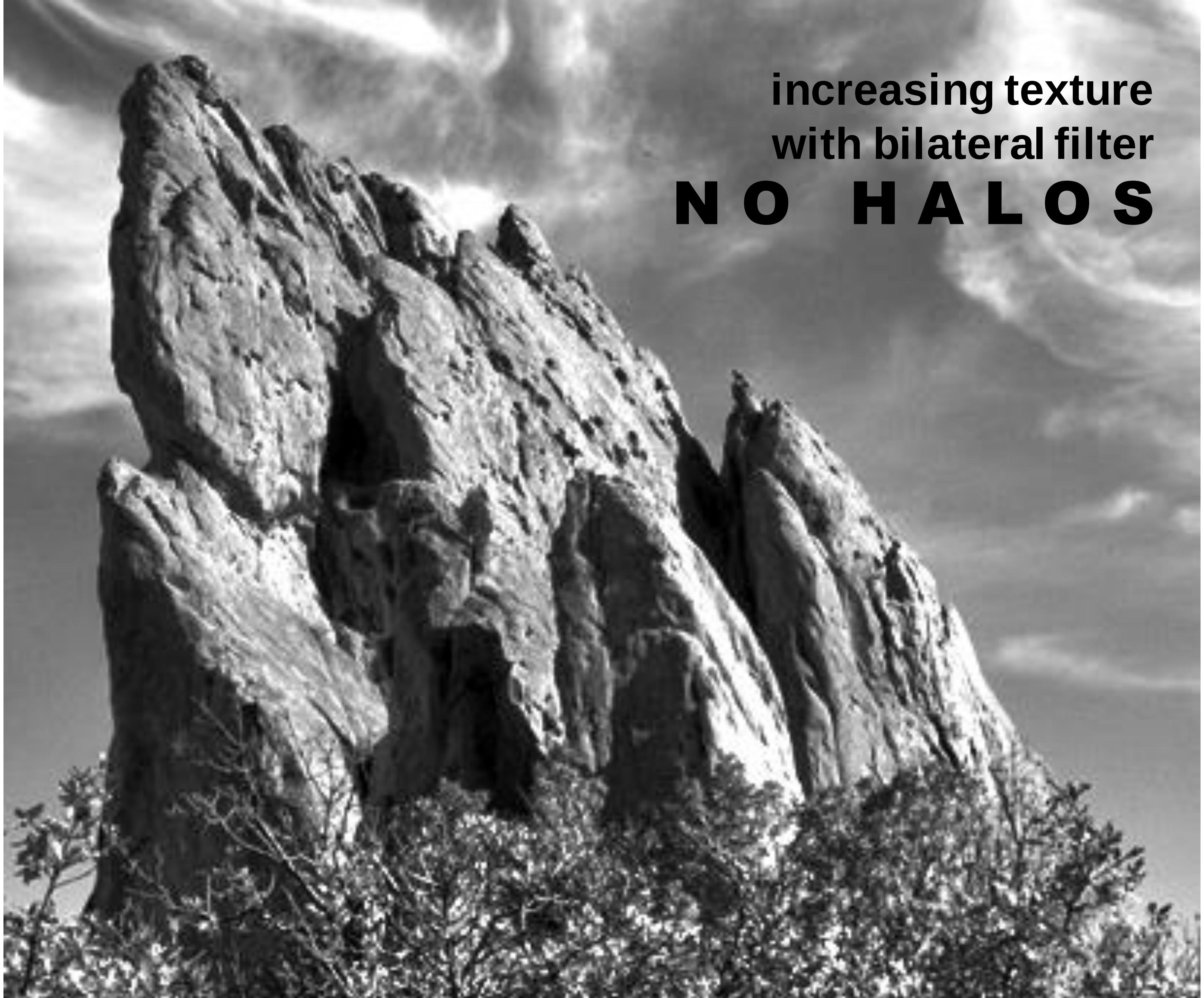
Bonus materials

Bilateral filtering

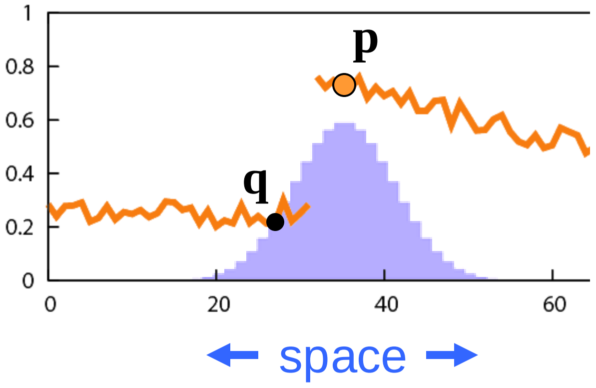
vs

- Bilateral filter tries to smooth image but still preserve edges


increasing texture
with Gaussian convolution
H A L O S


increasing texture
with bilateral filter
NO HALOS

Definition



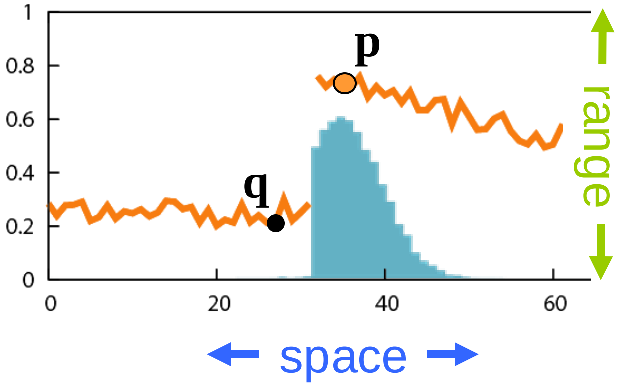
Gaussian blur

$$I^{\mathrm{b}}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}_{\text{space}} I_{\mathbf{q}}$$

- only spatial distance, intensity ignored

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]

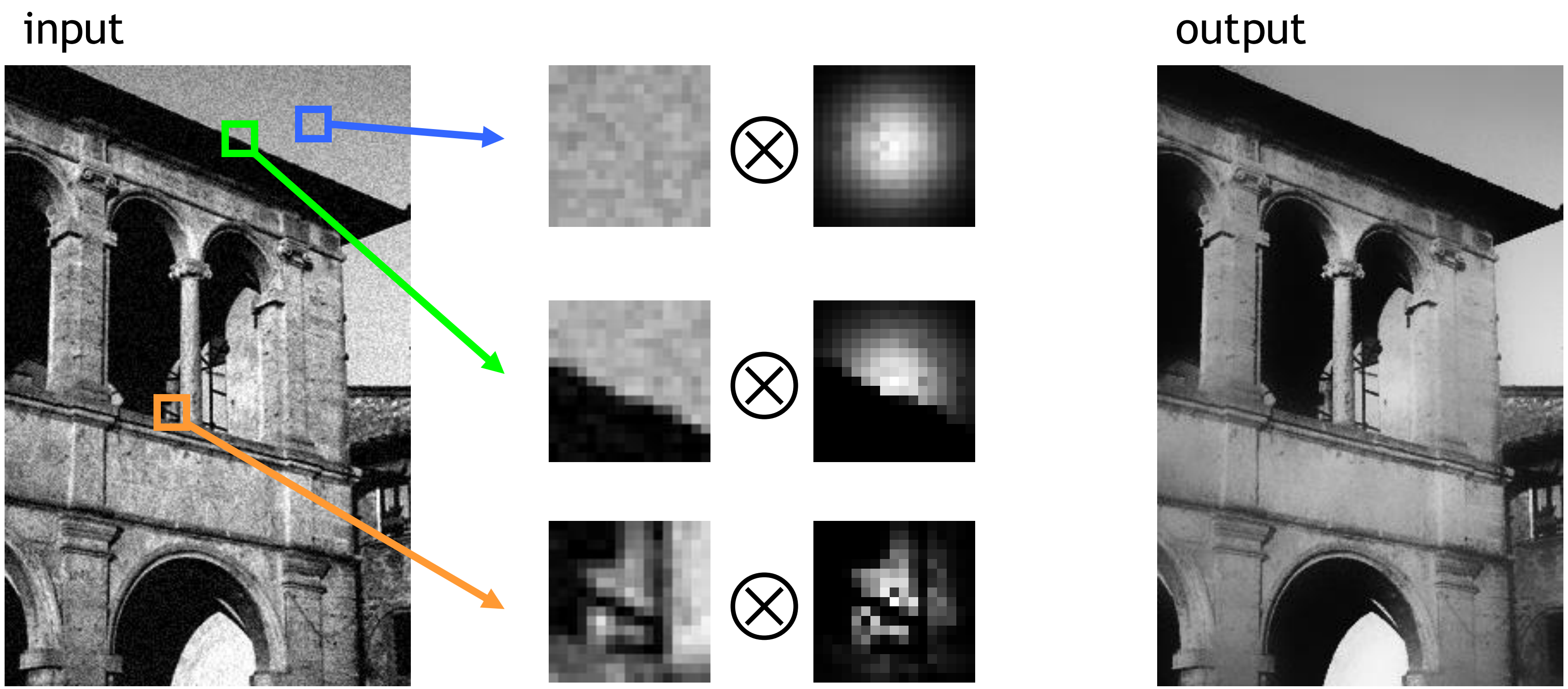


$$I^{\mathrm{bf}}_{\mathbf{p}} = \underbrace{\boxed{\frac{1}{W^{\mathrm{bf}}_{\mathbf{p}}}}}_{\text{normalization}} \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}_{\text{space}} \underbrace{\boxed{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}}_{\text{range}} I_{\mathbf{q}}$$

- spatial and range distances
- weights sum to 1

Example on a Real Image

- Kernels can have complex, spatially varying shapes
- Linear? Shift-invariant?



Bilateral Filter is Expensive

- Brute-force computation is slow (several minutes)
  - Two nested for loops:
    for each pixel, look at all pixels
  - Non-linear, depends on image content
    ⇨ no FFT, no pre-computation…

- Fast approximations exist
  - E.g., Durand 02, Weiss 06
    - Significant **loss of accuracy**
    - **No formal understanding** of accuracy versus speed
  - Better approximation (see this paper)

Summary

- Filtering is important
  - Even it does not give you state-of-the-art result, very handy for lots of things (denoising in particular)
  - Very basic, everyone expects you to understand that
  - Core component to understand more advance techniques (convolutional neural network, for example)
- Most interesting (manageable) filters are linear, often shift-invariant (same everywhere)
  - All linear shift-invariant filters can be depicted with convolution
  - Most convolutional filters people referred to in CV is actually cross-correlation in signal processing
- Filters that are separable (e.g., Gaussian filters) can be further speed up
- Some common filters:
  - Gaussian filters (linear? shift-invariant? Separable?)
    - Yes, yes, yes
  - Median filters (linear? shift-invariant? Separable?)
    - No, yes, no
  - Bilateral filters (linear? shift-invariant? Separable?)
    - No, yes, no