Lecture 2:
Pixels and Filters

What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Image filtering

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Image filtering

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Types of Images

Binary

Types of Images

Binary

Gray Scale

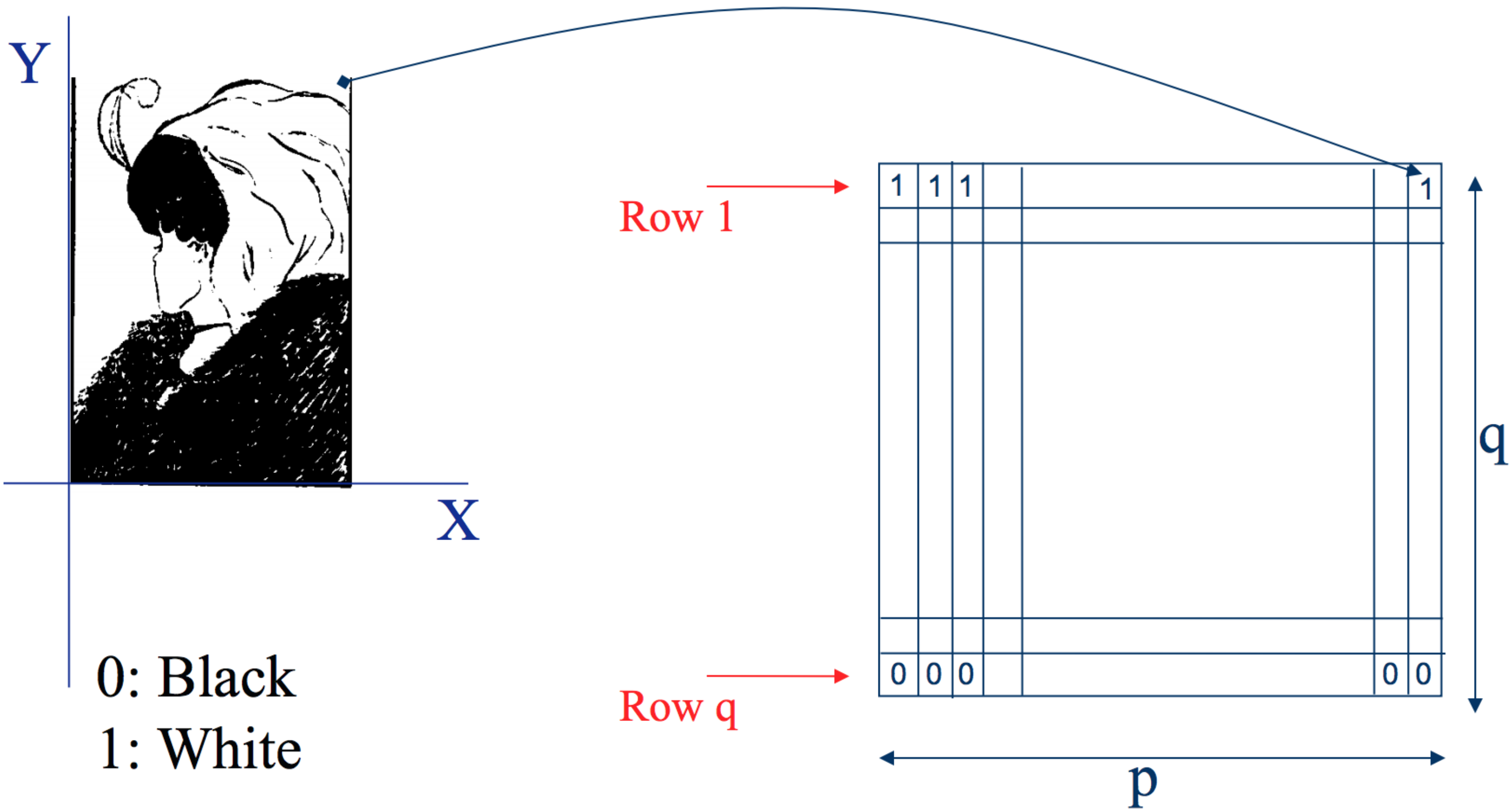
Types of Images

Binary

Gray Scale

Color

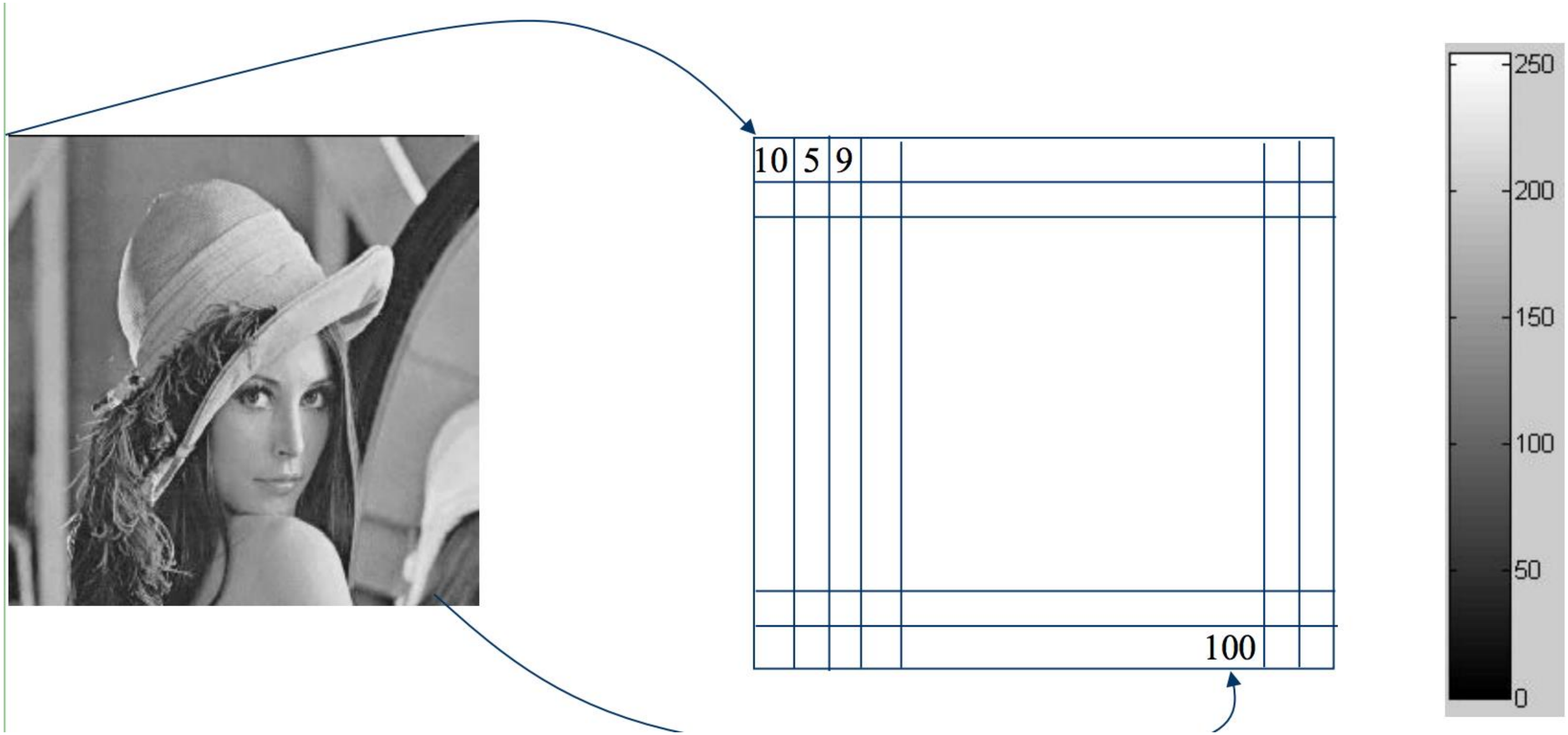
Binary image representation



0: Black
1: White

Slide credit: Ulas Bagci

Grayscale image representation

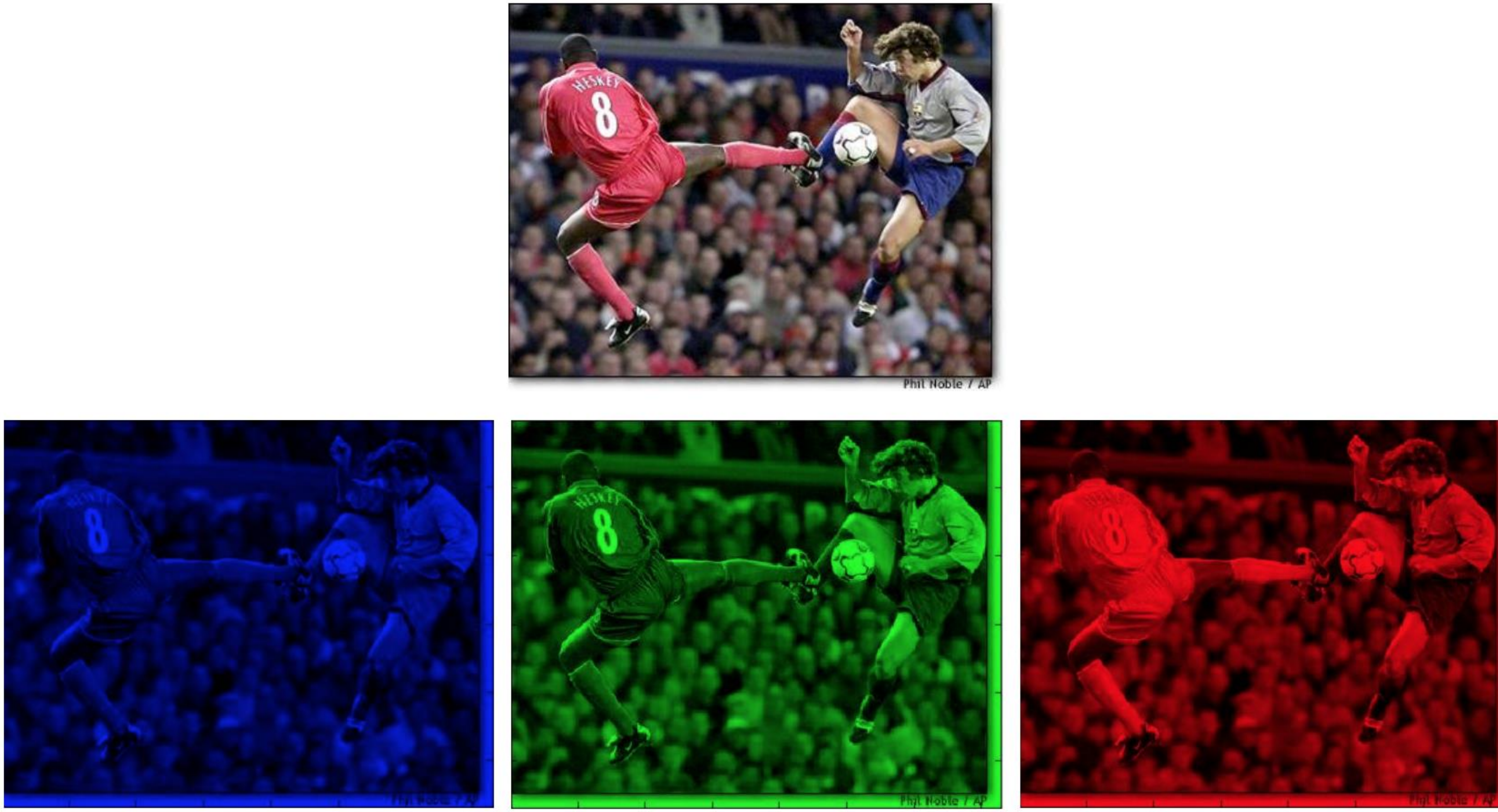


Slide credit: Ulas Bagci

Color Image - one channel

Slide credit: Ulas Bagci

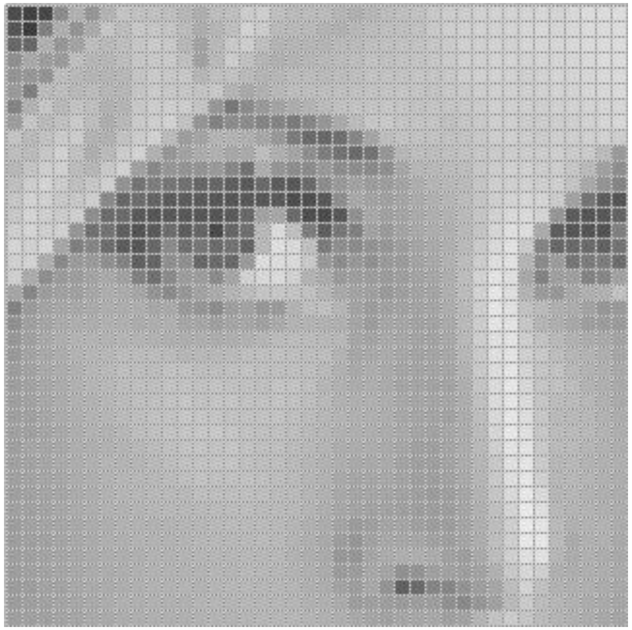
Color image representation

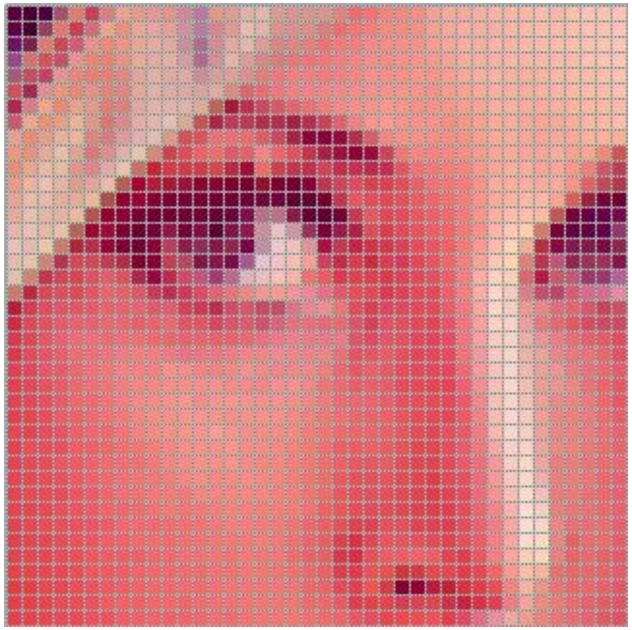


Slide credit: Ulas Bagci

Images are sampled

What happens when we zoom into the images we capture?





Resolution

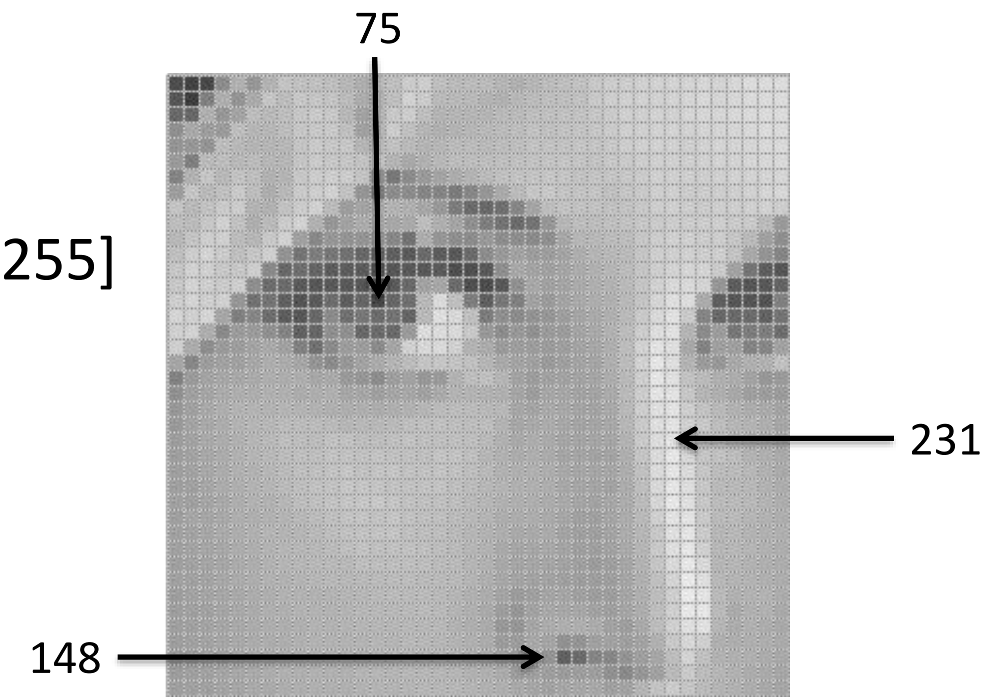
is a **sampling** parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density, and its standard value for recent screen technologies is 72 dpi

Slide credit: Ulas Bagci

Images are Sampled and Quantized

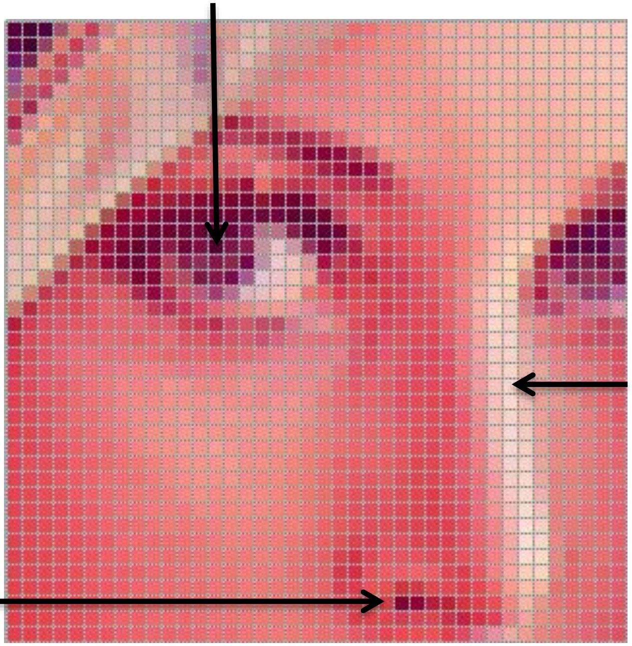
- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale"

    (or "intensity"): [0,255]



Images are Sampled and Quantized

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale"
    (or "intensity"): [0,255]
    - "color"
      - RGB: [R, G, B]
      - Lab: [L, a, b]
      - HSV: [H, S, V]



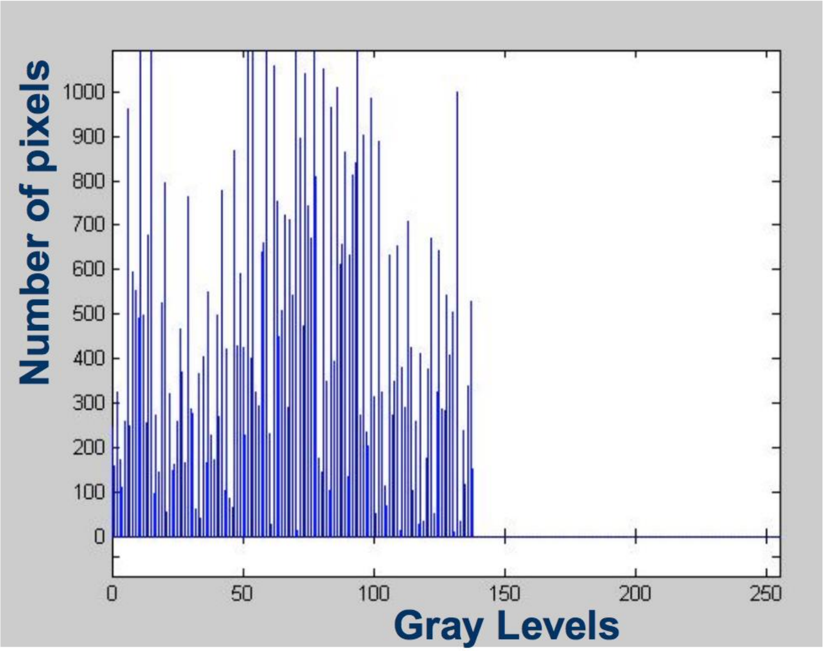
What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Image filtering

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Histogram

- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image



What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Linear systems (filters)
- Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Images as discrete functions

- Images are usually **digital** (**discrete**):
  - **Sample** the 2D space on a regular grid
- Represented as a matrix of integer values

pixel

j

i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Images as coordinates

Cartesian coordinates

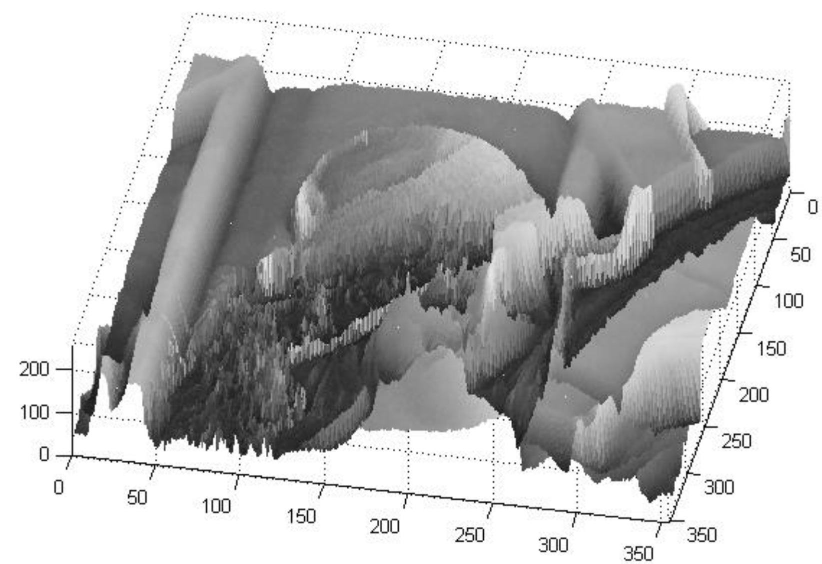
$$I[n,m] = \begin{bmatrix} \ddots & & \vdots & & \\ & I[-1,1] & I[0,1] & I[1,1] & \\ \dots & I[-1,0] & \underline{I[0,0]} & I[1,0] & \dots \\ & I[-1,-1] & I[0,-1] & I[1,-1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Notation for discrete functions

Images as functions

- **An Image** as a function f from R^2 to R^M:
  - $I(\,x, y\,)$ gives the **intensity** at position $(\,x, y\,)$
  - Defined over a rectangle, with a finite range:

$$I\colon \underbrace{[a,b] \times [c,d\,]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$



Images as functions

- **An Image** as a function f from R^2 to R^M:
  - $I(x, y)$ gives the **intensity** at position (x, y)
  - Defined over a rectangle, with a finite range:

$$I: \underbrace{[a,b] \times [c,d]}_{\text{Domain support}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

- A color image: $I(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$

What we will learn today?

- Image sampling and quantization
- Image histograms
- Images as functions
- Image filtering

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Systems and Filters

Filtering:

– Forming a new image whose pixel values are transformed from original pixel values

Goals:

- Goal is to extract useful information from images, or transform images into another domain where we can modify/enhance image properties
  - Features (edges, corners, blobs…)
  - super-resolution; in-painting; de-noising

De-noising

Salt and pepper noise

Super-resolution

In-painting

Bertamio et al

Image filtering

- Image filters in spatial domain
  – Filter is a mathematical operation of a grid of numbers
  – Smoothing, sharpening, measuring texture
- Image filters in the frequency domain
  – Filtering is a way to modify the frequencies of images
  – Denoising, sampling, image compression

James Hays

Image filtering

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

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

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

[ ] [ ] [ ]

Example: box filter

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

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

Slide credit: David Lowe (UBC)

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

$h[.,.]$

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

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							

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

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

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

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

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

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

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?

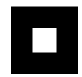
$$f[\cdot\,,\cdot\,]$$

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

Slide credit: David Lowe (UBC)

Smoothing with box filter

James Hays



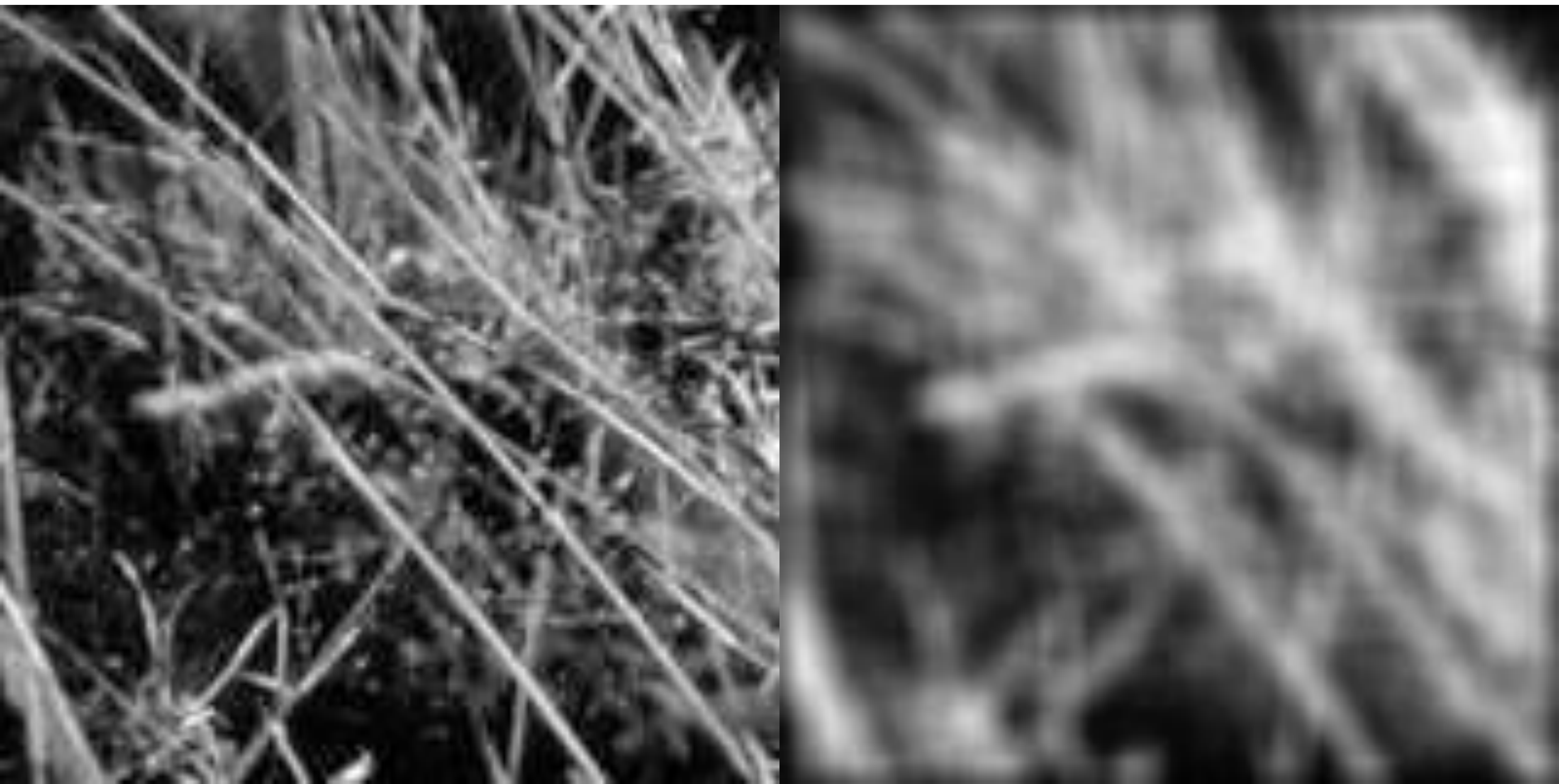


Image filtering

- 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

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

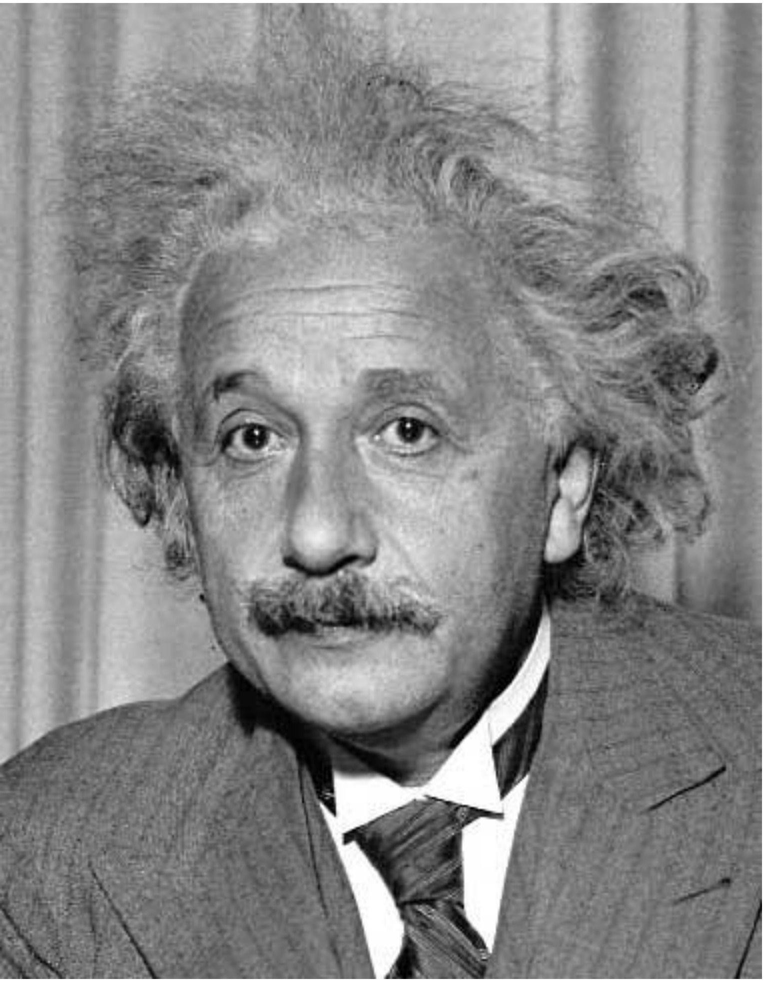
Original

0	0	0
0	0	1
0	0	0

Shifted left
By 1 pixel

Source: D. Lowe

3. Practice with linear filters



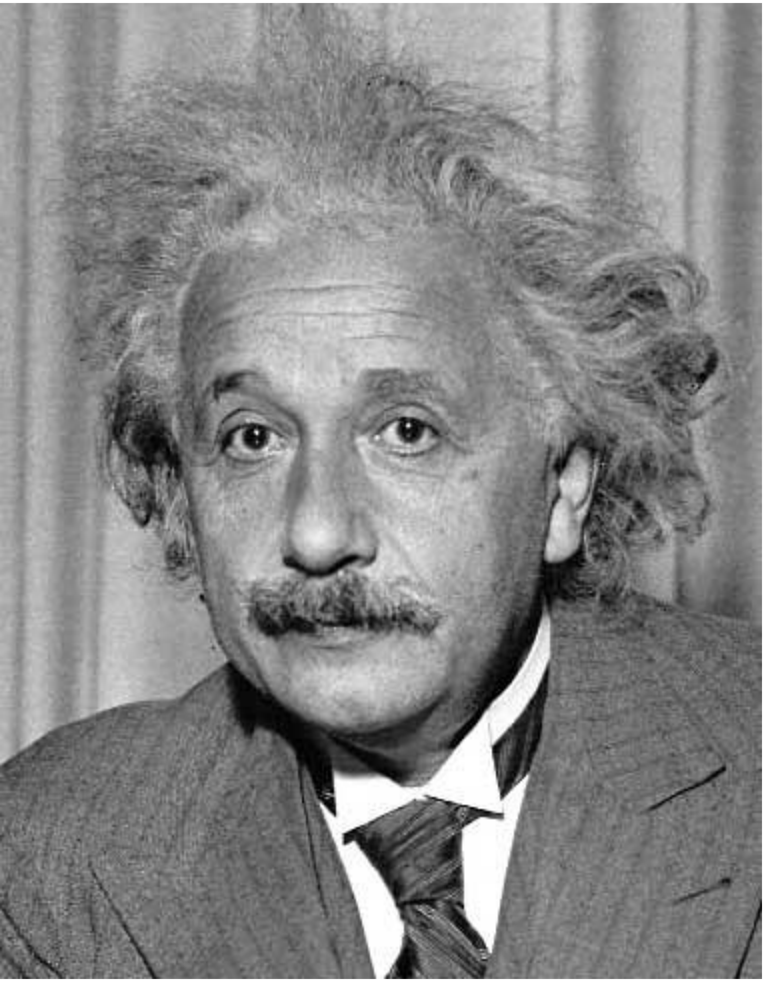
1	0	-1
2	0	-2
1	0	-1

Sobel

Vertical Edge
(absolute value)

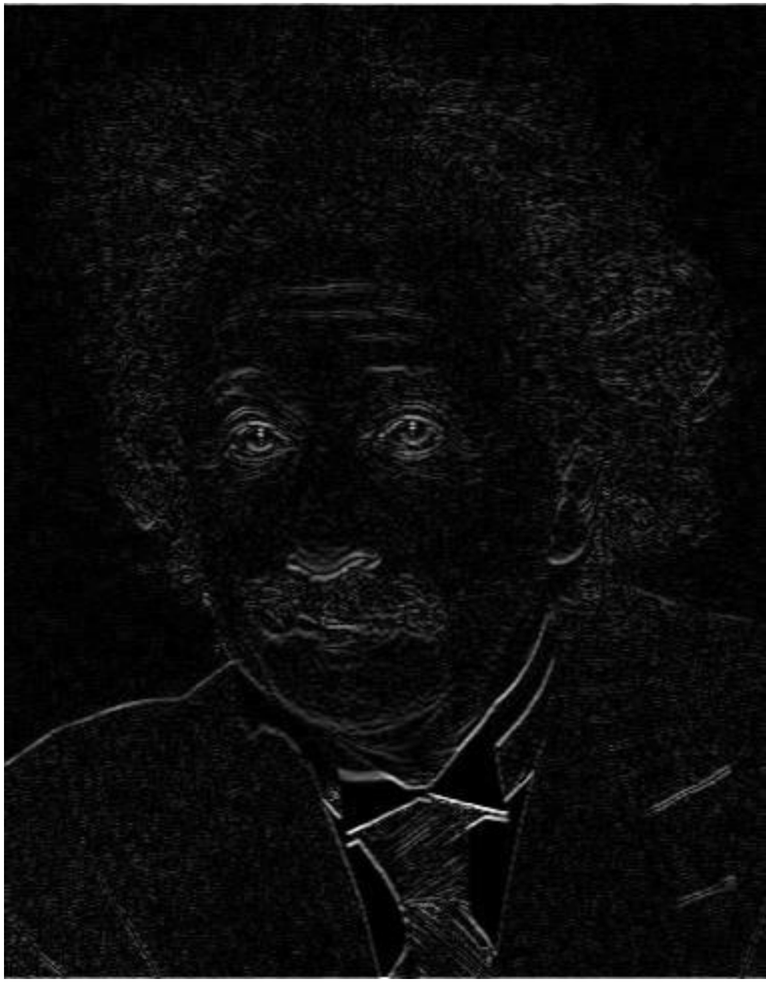
David Lowe

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

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

?

(Note that filter sums to 1)

Source: D. Lowe

4. Practice with linear filters

Original

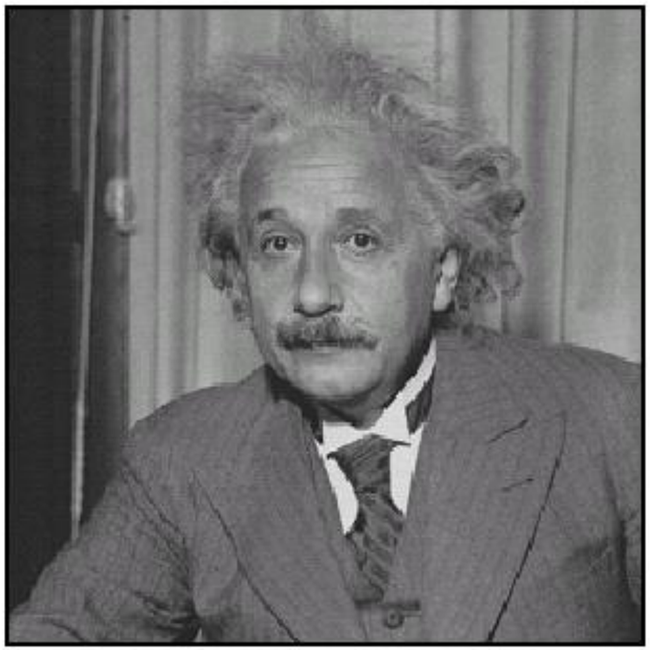
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

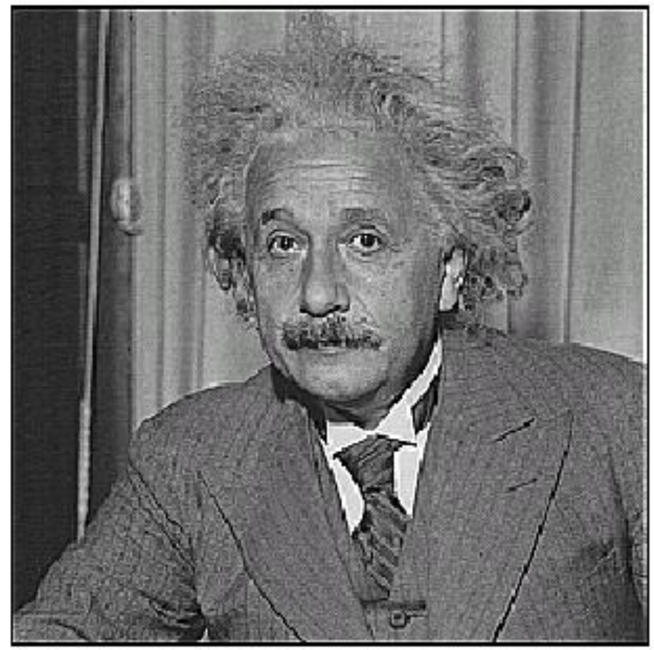
- Accentuates differences with local average

Source: D. Lowe

4. Practice with linear filters



before



after

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);`

- 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');`

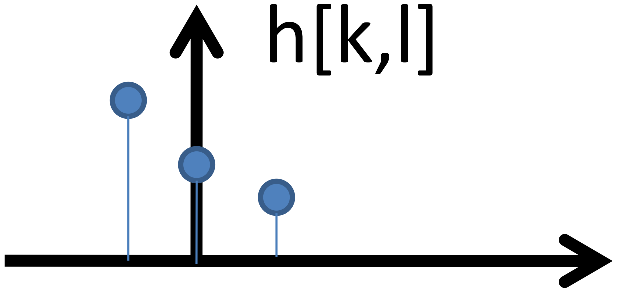
Correlation and convolution are identical when the filter is symmetric.

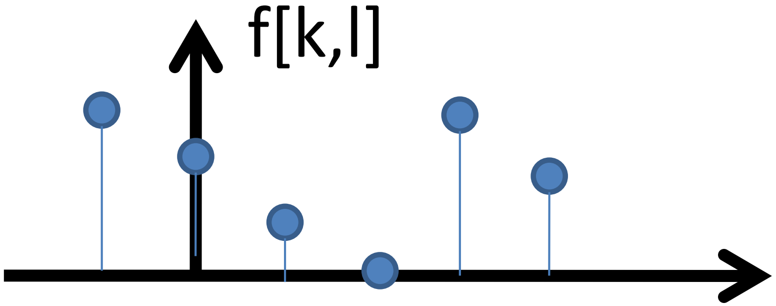
James Hays

1D Discrete convolution (symbol:) $*$

We are going to convolve a function **f** with a filter **h.**

$$g[n] = \sum_{k} f[k]h[n-k]$$



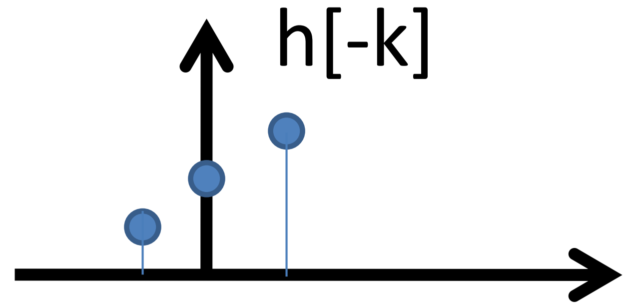


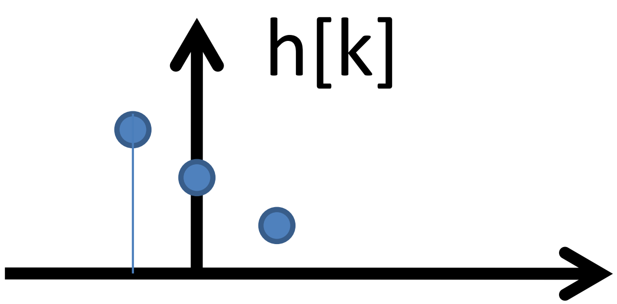
1D Discrete convolution (symbol: $*$)

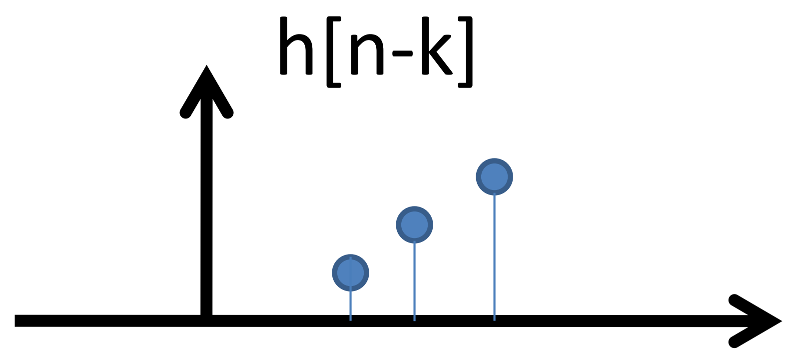
We are going to convolve a function **f** with a filter **h.**

$$g[n] = \sum_k f[k]h[n-k]$$

We first need to calculate h[n-k, m-l]

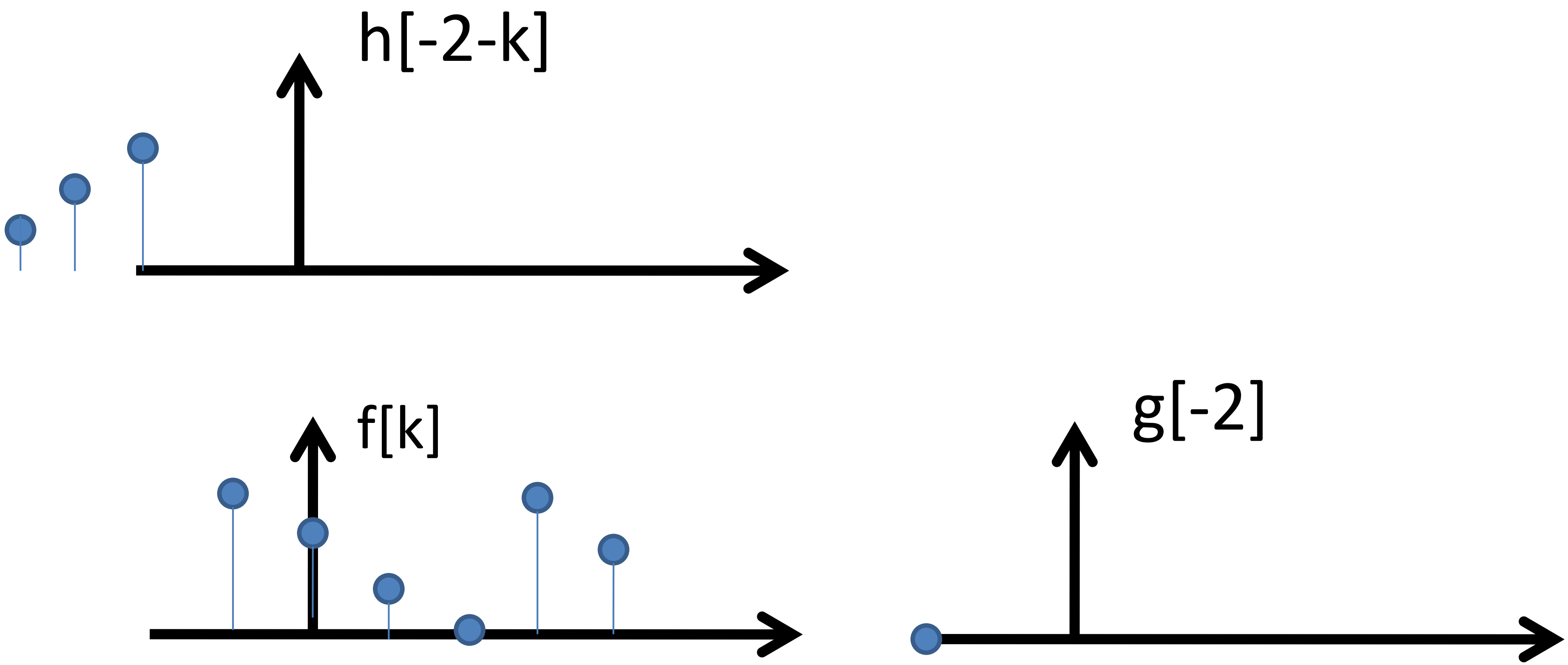






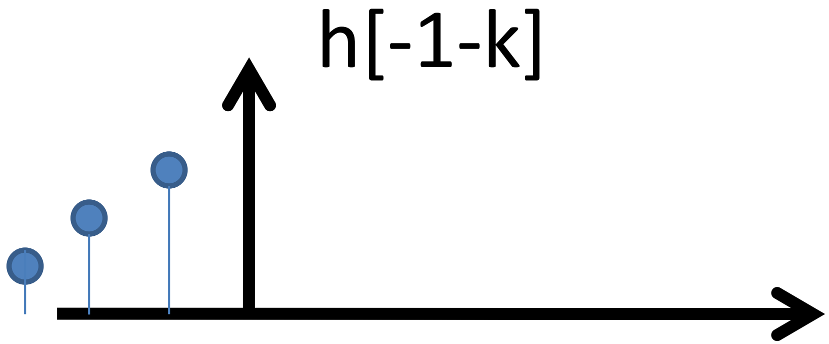
Discrete convolution (symbol: $*$)

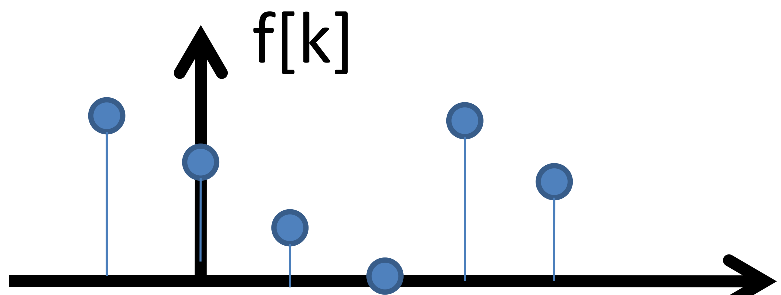
We are going to convolve a function **f** with a filter **h.**

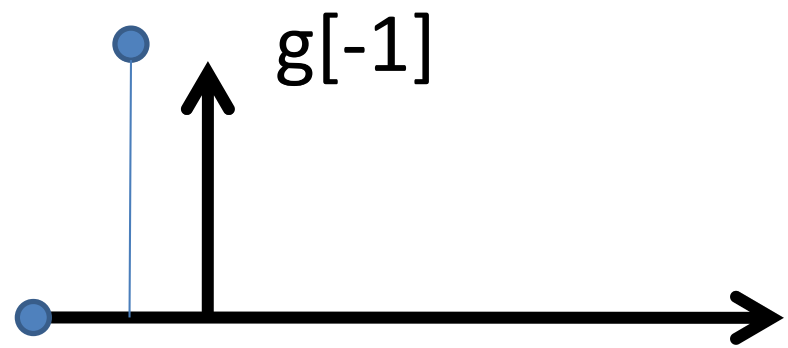


Discrete convolution (symbol: $*$)

We are going to convolve a function **f** with a filter **h.**

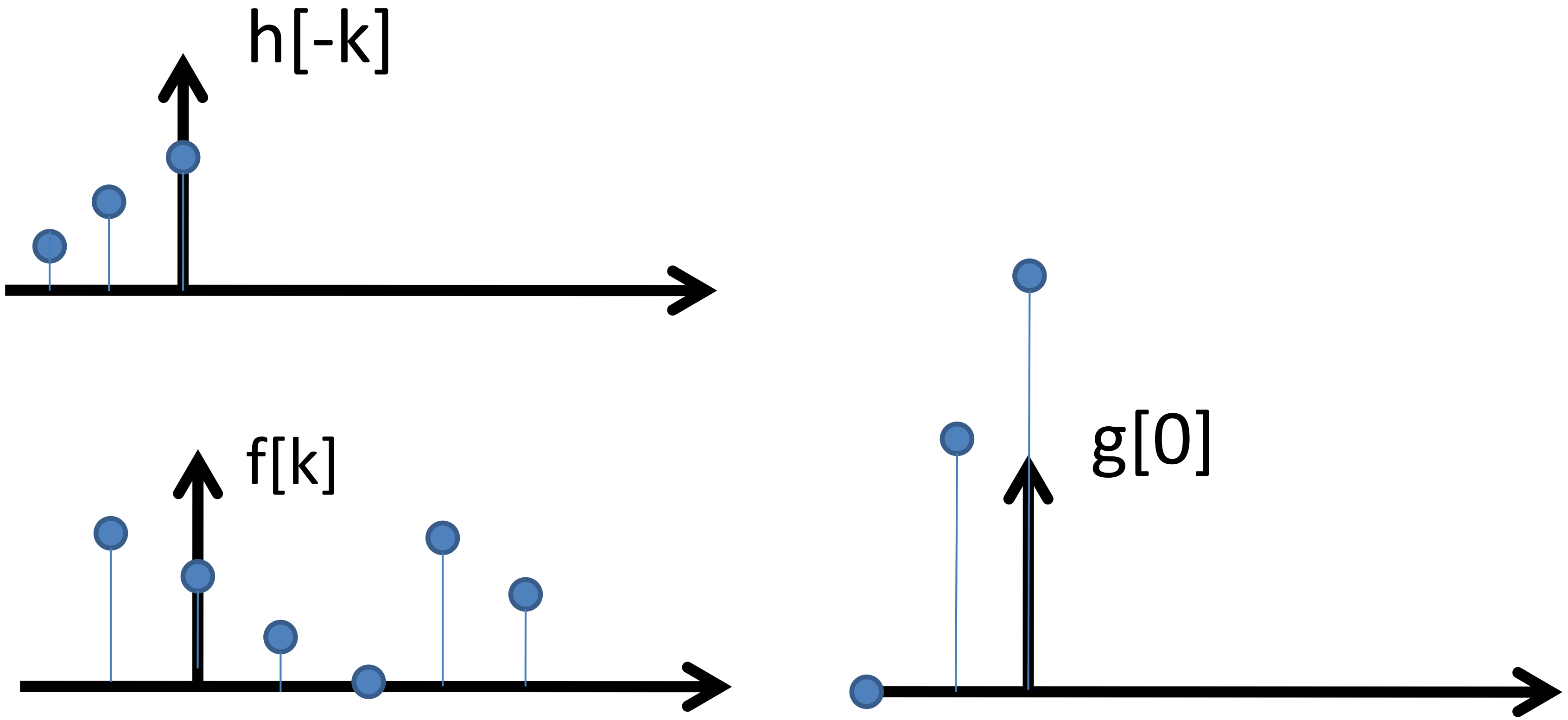






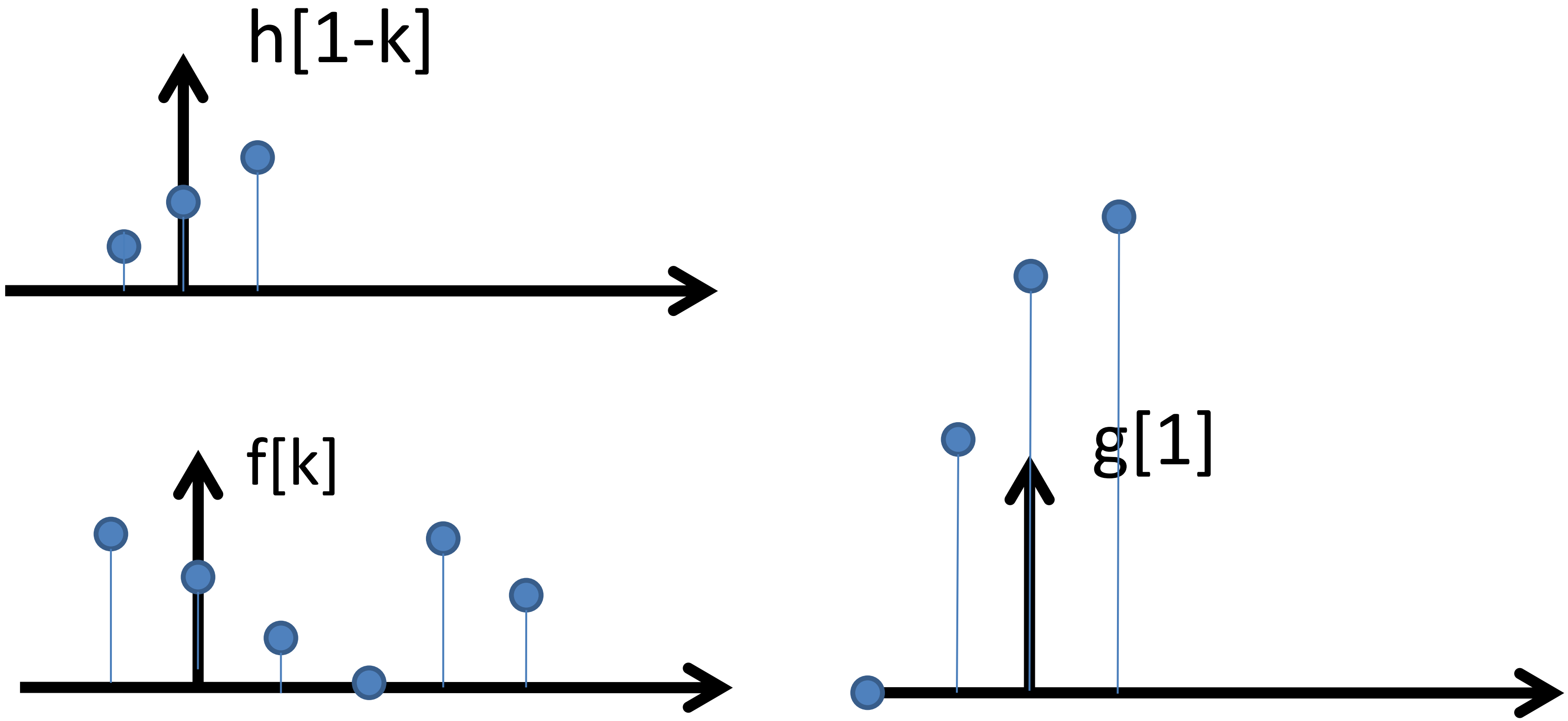
Discrete convolution (symbol: $*$)

We are going to convolve a function **f** with a filter **h.**



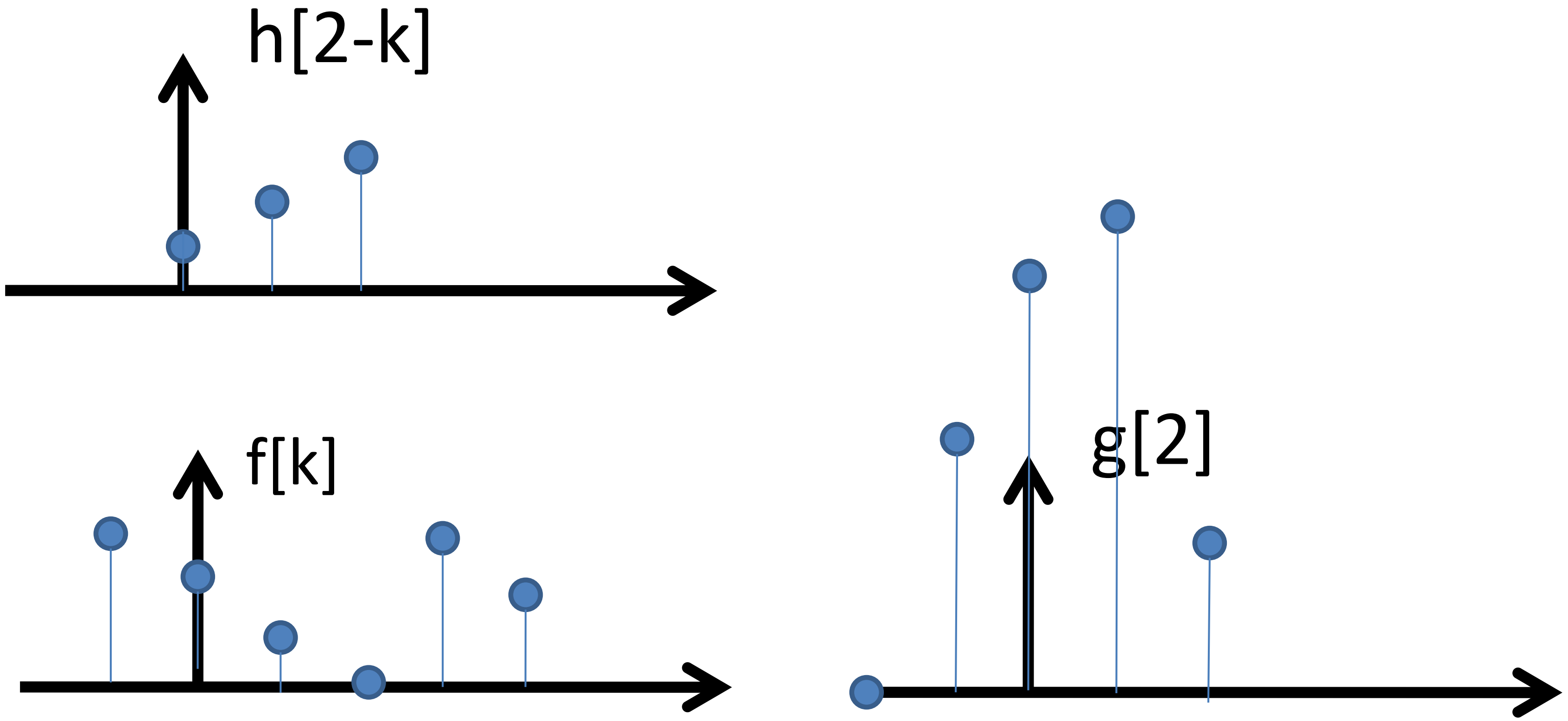
Discrete convolution (symbol: $*$)

We are going to convolve a function **f** with a filter **h.**



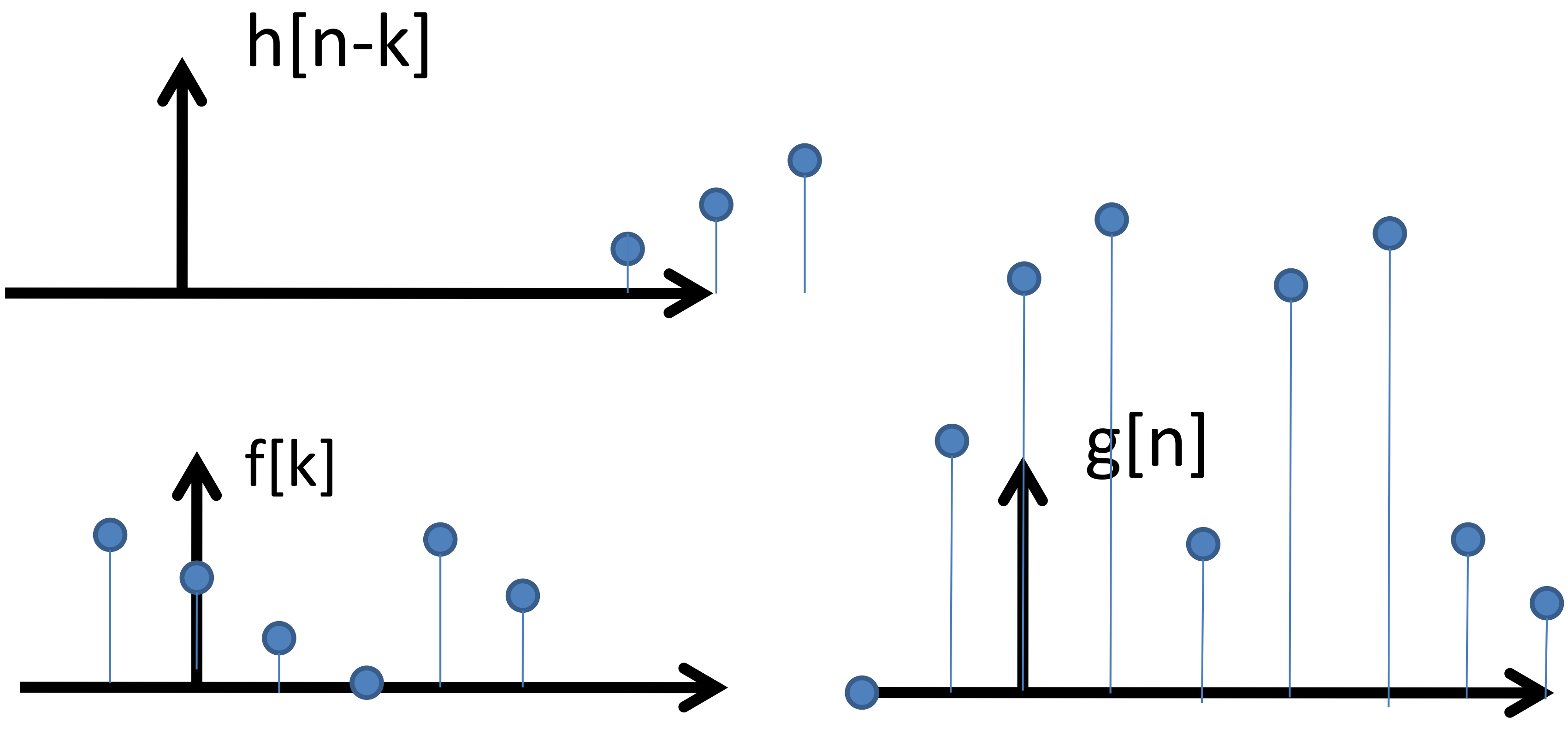
Discrete convolution (symbol: $*$)

We are going to convolve a function **f** with a filter **h.**



Discrete convolution (symbol: $*$)

We are going to convolve a function **f** with a filter **h.**

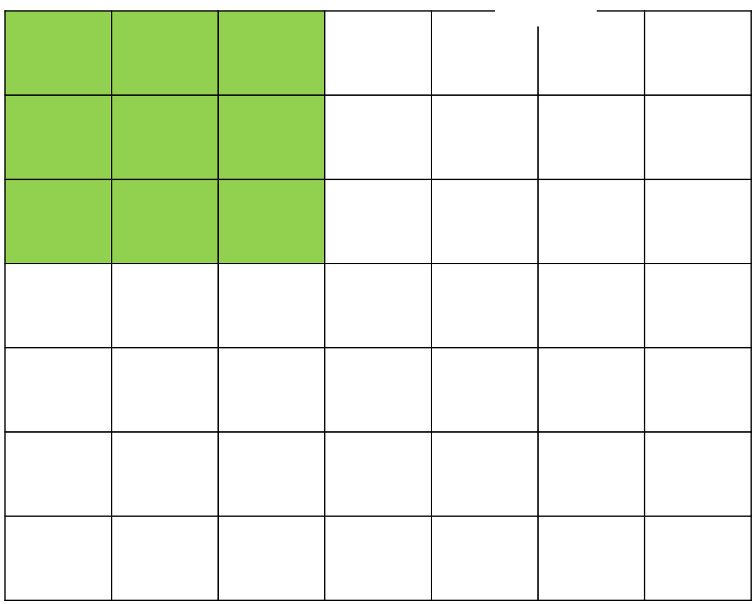


2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

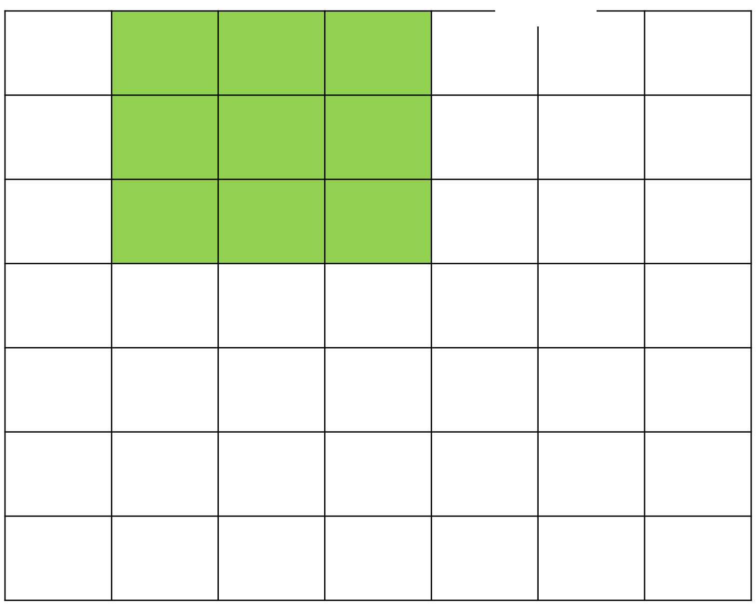
n

2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

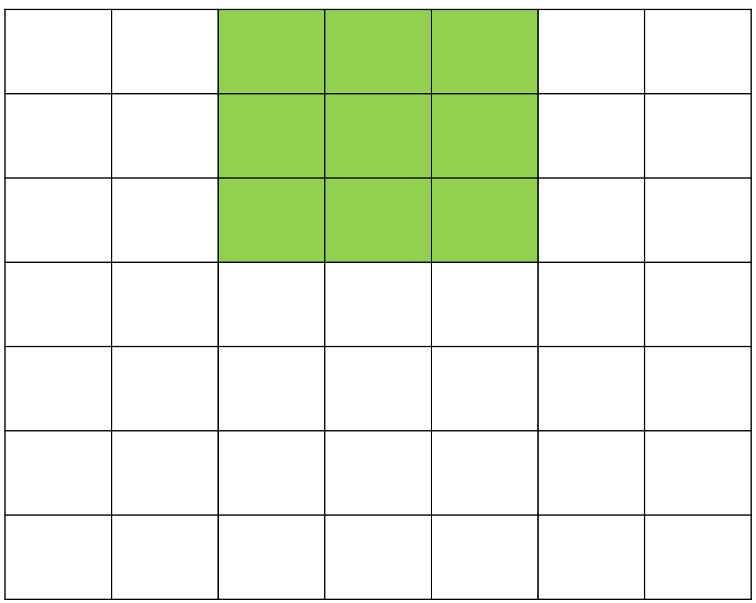
n

2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

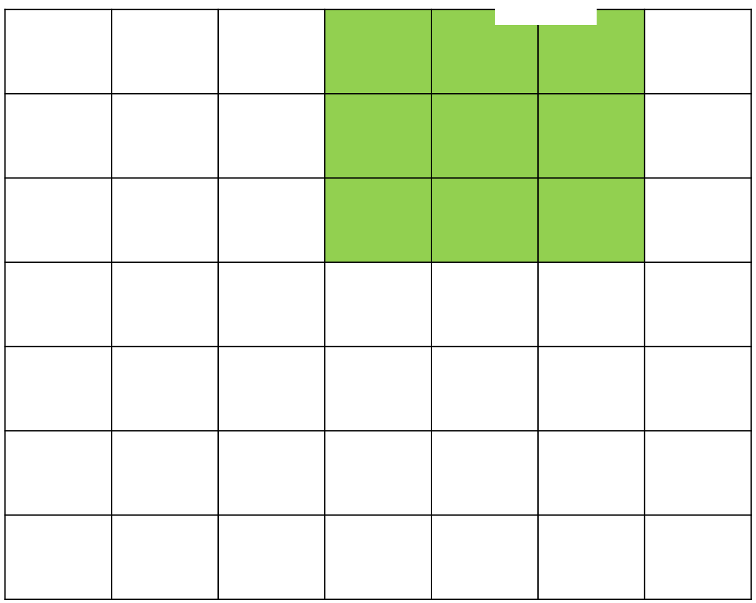
n

2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

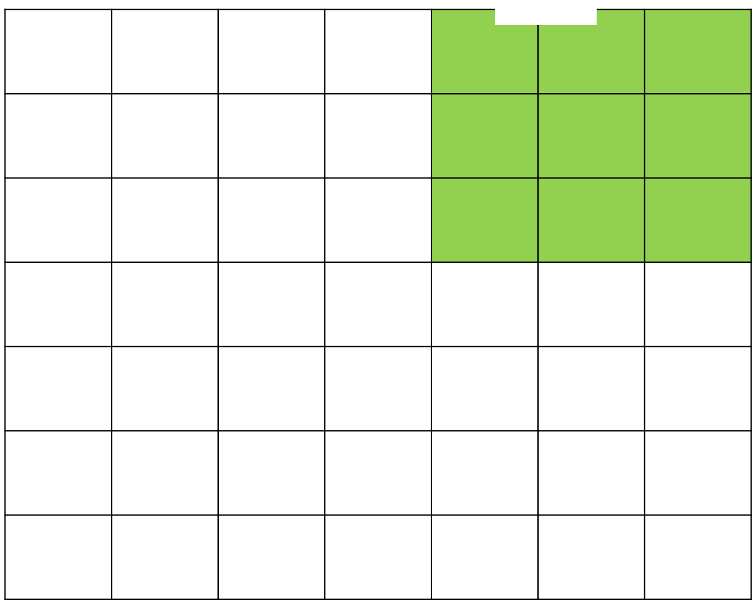
n

2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

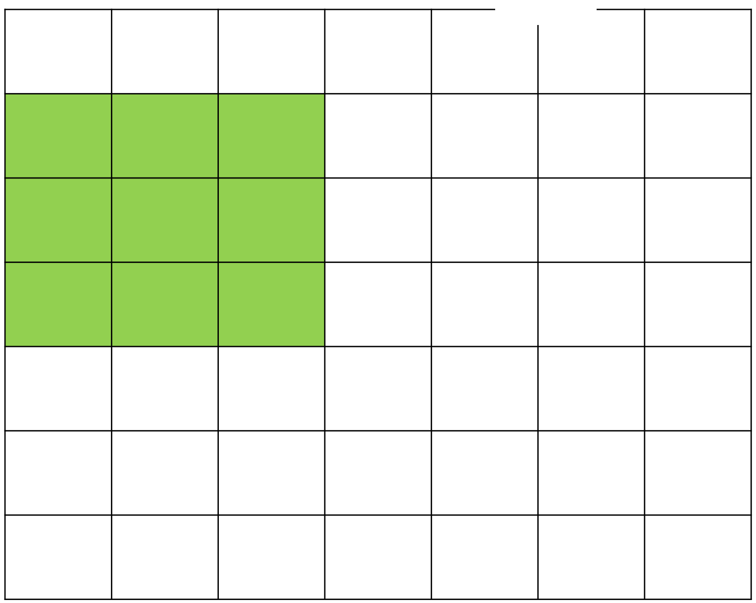
n

2D convolution

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

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



Assume we have a filter(h[,]) that is 3x3. and an image (f[,]) that is 7x7.

2D convolution example

1	2	3
4	5	6
7	8	9

Input

n \ m	-1	0	1
-1	-1	-2	-1
0	0	0	0
1	1	2	1

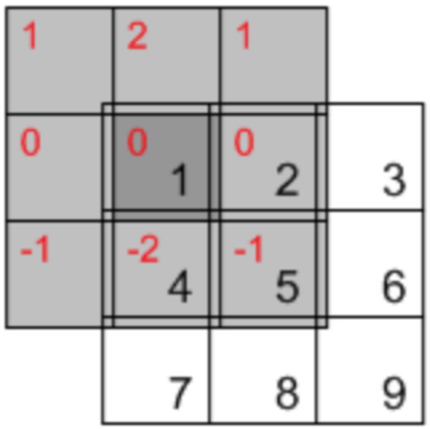
Kernel

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



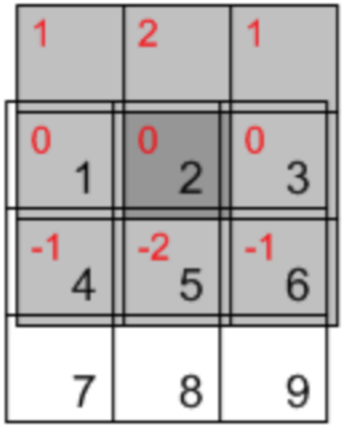
$$
\begin{aligned}
&= x[-1,-1]\cdot h[1,1] + x[0,-1]\cdot h[0,1] + x[1,-1]\cdot h[-1,1] \\
&\quad + x[-1,0]\cdot h[1,0] + x[0,0]\cdot h[0,0] + x[1,0]\cdot h[-1,0] \\
&\quad + x[-1,1]\cdot h[1,-1] + x[0,1]\cdot h[0,-1] + x[1,1]\cdot h[-1,-1] \\
&= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 0\cdot 0 + 1\cdot 0 + 2\cdot 0 + 0\cdot(-1) + 4\cdot(-2) + 5\cdot(-1) = -13
\end{aligned}
$$

$\boxed{-13}$	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



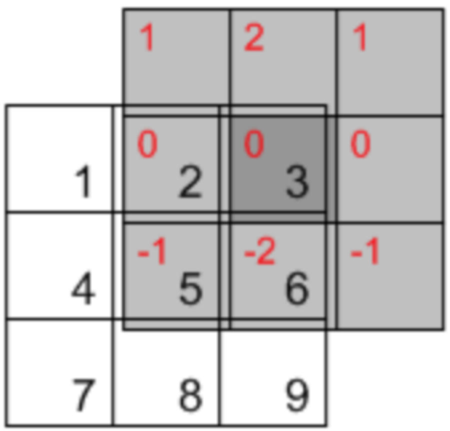
$$
\begin{aligned}
&= x[0,-1]\cdot h[1,1] + x[1,-1]\cdot h[0,1] + x[2,-1]\cdot h[-1,1] \\
&\quad + x[0,0]\cdot h[1,0] + x[1,0]\cdot h[0,0] + x[2,0]\cdot h[-1,0] \\
&\quad + x[0,1]\cdot h[1,-1] + x[1,1]\cdot h[0,-1] + x[2,1]\cdot h[-1,-1] \\
&= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 1\cdot 0 + 2\cdot 0 + 3\cdot 0 + 4\cdot(-1) + 5\cdot(-2) + 6\cdot(-1) = -20
\end{aligned}
$$

-13	$\boxed{-20}$	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



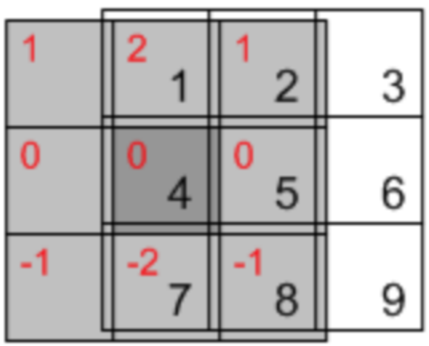
$$
\begin{aligned}
&= x[1,-1]\cdot h[1,1] + x[2,-1]\cdot h[0,1] + x[3,-1]\cdot h[-1,1] \\
&\quad + x[1,0]\cdot h[1,0] + x[2,0]\cdot h[0,0] + x[3,0]\cdot h[-1,0] \\
&\quad + x[1,1]\cdot h[1,-1] + x[2,1]\cdot h[0,-1] + x[3,1]\cdot h[-1,-1] \\
&= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 2\cdot 0 + 3\cdot 0 + 0\cdot 0 + 5\cdot(-1) + 6\cdot(-2) + 0\cdot(-1) = -17
\end{aligned}
$$

-13	-20	$\boxed{-17}$
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



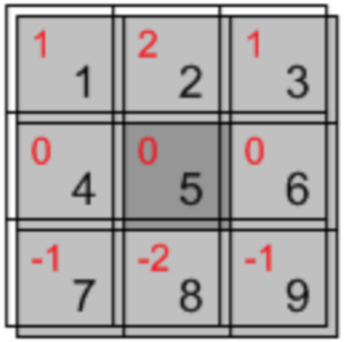
$$= x[-1,0]\cdot h[1,1] + x[0,0]\cdot h[0,1] + x[1,0]\cdot h[-1,1]$$
$$+ x[-1,1]\cdot h[1,0] + x[0,1]\cdot h[0,0] + x[1,1]\cdot h[-1,0]$$
$$+ x[-1,2]\cdot h[1,-1] + x[0,2]\cdot h[0,-1] + x[1,2]\cdot h[-1,-1]$$
$$= 0\cdot 1 + 1\cdot 2 + 2\cdot 1 + 0\cdot 0 + 4\cdot 0 + 5\cdot 0 + 0\cdot(-1) + 7\cdot(-2) + 8\cdot(-1) = -18$$

-13	-20	-17
$\boxed{-18}$	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



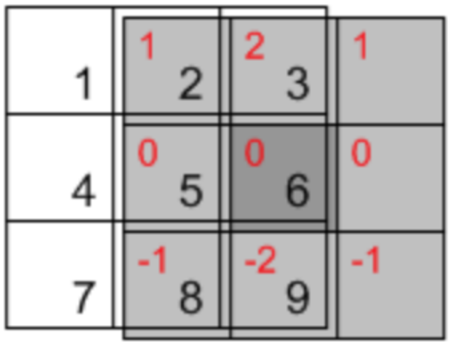
$$
\begin{aligned}
&= x[0,0]\cdot h[1,1] + x[1,0]\cdot h[0,1] + x[2,0]\cdot h[-1,1] \\
&\quad + x[0,1]\cdot h[1,0] + x[1,1]\cdot h[0,0] + x[2,1]\cdot h[-1,0] \\
&\quad + x[0,2]\cdot h[1,-1] + x[1,2]\cdot h[0,-1] + x[2,2]\cdot h[-1,-1] \\
&= 1\cdot 1 + 2\cdot 2 + 3\cdot 1 + 4\cdot 0 + 5\cdot 0 + 6\cdot 0 + 7\cdot(-1) + 8\cdot(-2) + 9\cdot(-1) = -24
\end{aligned}
$$

-13	-20	-17
-18	$\boxed{-24}$	-18
13	20	17

Output

Slide credit: Song Ho Ahn

2D convolution example



$$
\begin{aligned}
&= x[1,0]\cdot h[1,1] + x[2,0]\cdot h[0,1] + x[3,0]\cdot h[-1,1] \\
&\quad + x[1,1]\cdot h[1,0] + x[2,1]\cdot h[0,0] + x[3,1]\cdot h[-1,0] \\
&\quad + x[1,2]\cdot h[1,-1] + x[2,2]\cdot h[0,-1] + x[3,2]\cdot h[-1,-1] \\
&= 2\cdot 1 + 3\cdot 2 + 0\cdot 1 + 5\cdot 0 + 6\cdot 0 + 0\cdot 0 + 8\cdot(-1) + 9\cdot(-2) + 0\cdot(-1) = -18
\end{aligned}
$$

-13	-20	-17
-18	-24	$\boxed{-18}$
13	20	17

Output

Slide credit: Song Ho Ahn

Convolution in 2D - examples

Original

*

•0	•0	•0
•0	•1	•0
•0	•0	•0

=

?

Courtesy of D Lowe

Convolution in 2D - examples

Original

$*$

•0	•0	•0
•0	•1	•0
•0	•0	•0

$=$

Filtered
(no change)

Courtesy of D Lowe

Convolution in 2D - examples

Original

$*$

•0	•0	•0
•0	•0	•1
•0	•0	•0

$=$?

Courtesy of D Lowe

Convolution in 2D - examples

Original

*

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

Shifted right
By 1 pixel

Courtesy of D Lowe

Convolution in 2D - examples

Original

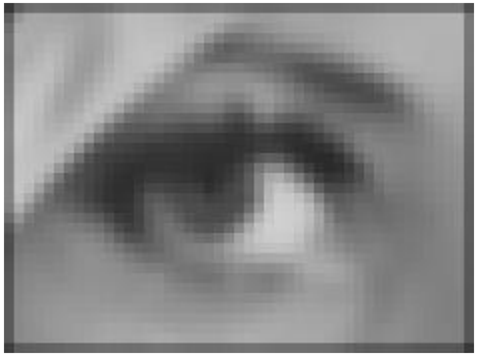
$$* \ \frac{1}{9}\begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix} = \ ?$$

Courtesy of D Lowe

Convolution in 2D - examples

Original

$$* \frac{1}{9}\begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix} =$$



Blur (with a box filter)

Courtesy of D Lowe

Convolution in 2D - examples

Original

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

(Note that filter sums to 1)

$$\begin{bmatrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 1 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{bmatrix} + \overbrace{\begin{bmatrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 1 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix}}^{\text{“details of the image”}}$$

Courtesy of D Lowe

- What does blurring take away?

–

=

  - Let’s add it back:

+ a

=

Convolution in 2D – Sharpening filter

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

Courtesy of D Lowe

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

Proof of associativity of convolution

$$
\begin{aligned}
((f \star g) \star h)(n) &= \sum_{k=0}^{n} (f \star g)(k) h(n-k) \\
&= \sum_{k=0}^{n} \sum_{l=0}^{k} f(l) g(k-l) h(n-k) \\
&= \sum_{l=0}^{n} \sum_{k=l}^{n} f(l) g(k-l) h(n-k) \\
&= \sum_{l=0}^{n} \sum_{k=0}^{n-l} f(l) g(k) h(n-k-l) \\
&= \sum_{l=0}^{n} f(l) (g \star h)(n-l) \\
&= (f \star (g \star h))(n)
\end{aligned}
$$

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
- Why important?

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)
  - Why important?

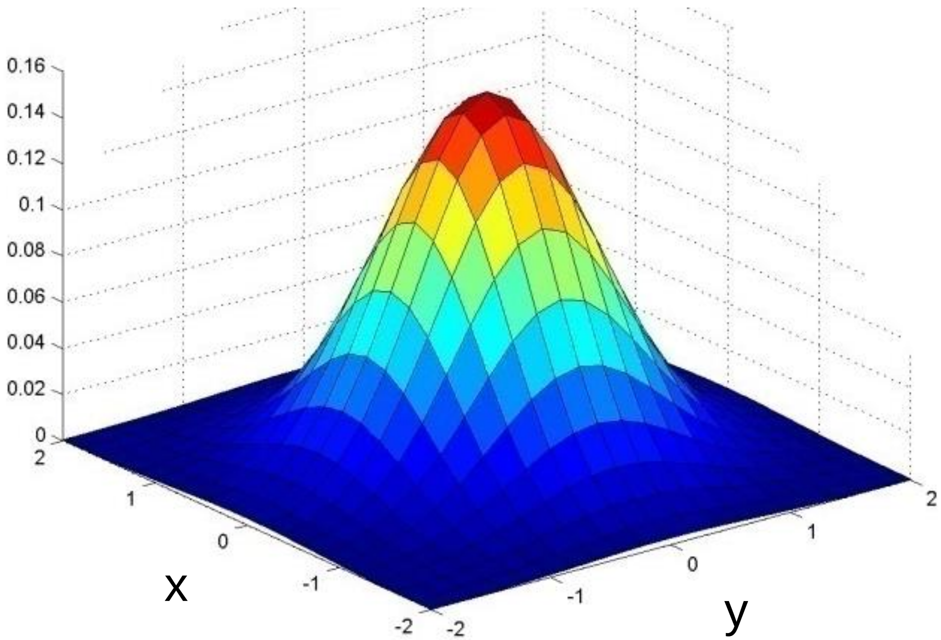
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

- Scalars factor out: $ka * b = a * kb = k (a * b)$

Source: S. Lazebnik

Important filter: Gaussian

- Weight contributions of neighboring pixels by nearness

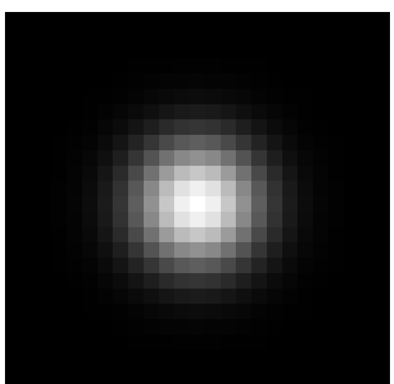


x

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

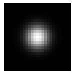


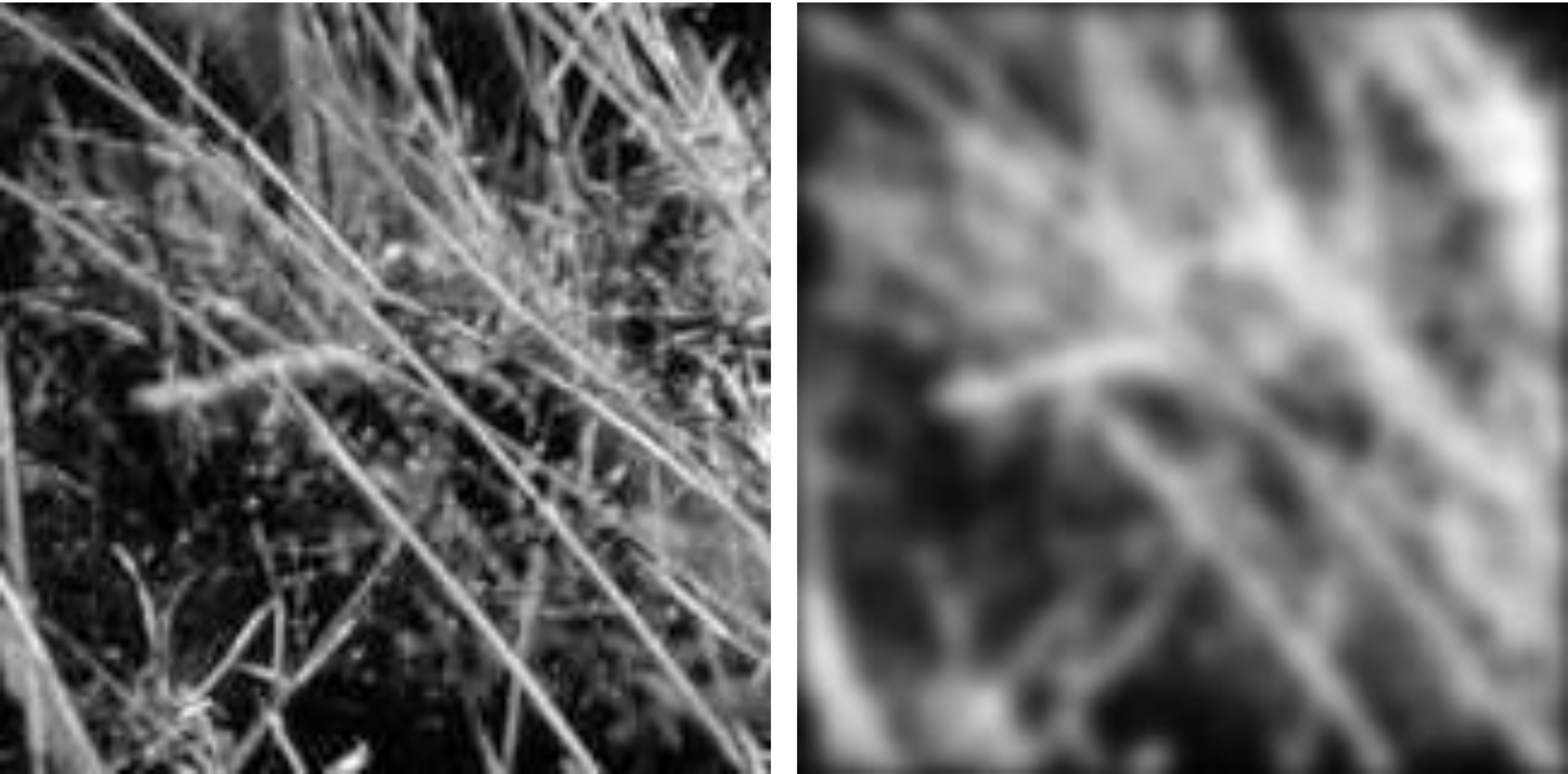
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Slide credit: Christopher Rasmussen

James Hays

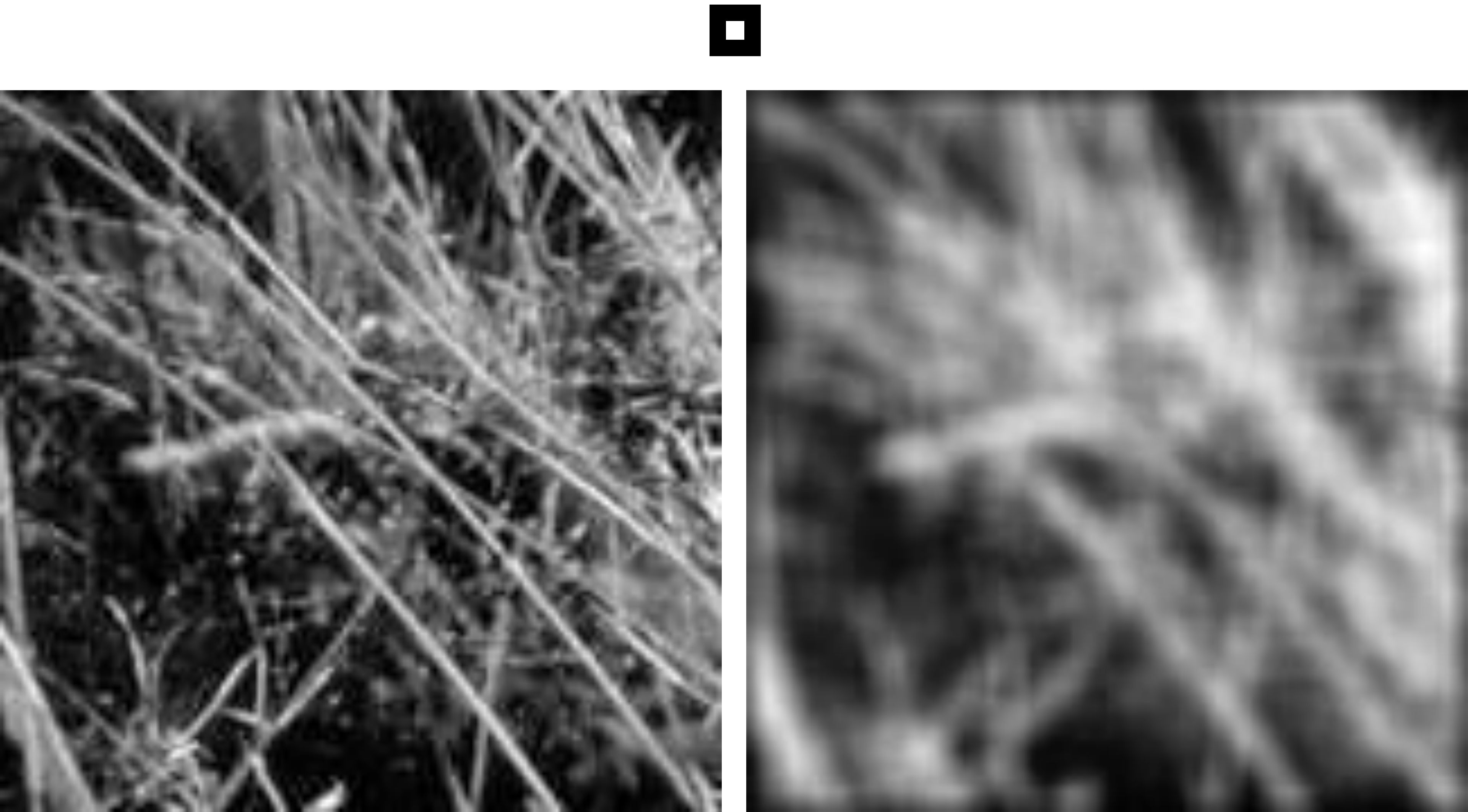
Smoothing with Gaussian filter





Smoothing with box filter

James Hays



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

James Hays

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.

NON-LINEAR FILTERS

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

Steve Seitz, Steve Marschner

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

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

$I[.,.]$

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

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

$h[.,.]$

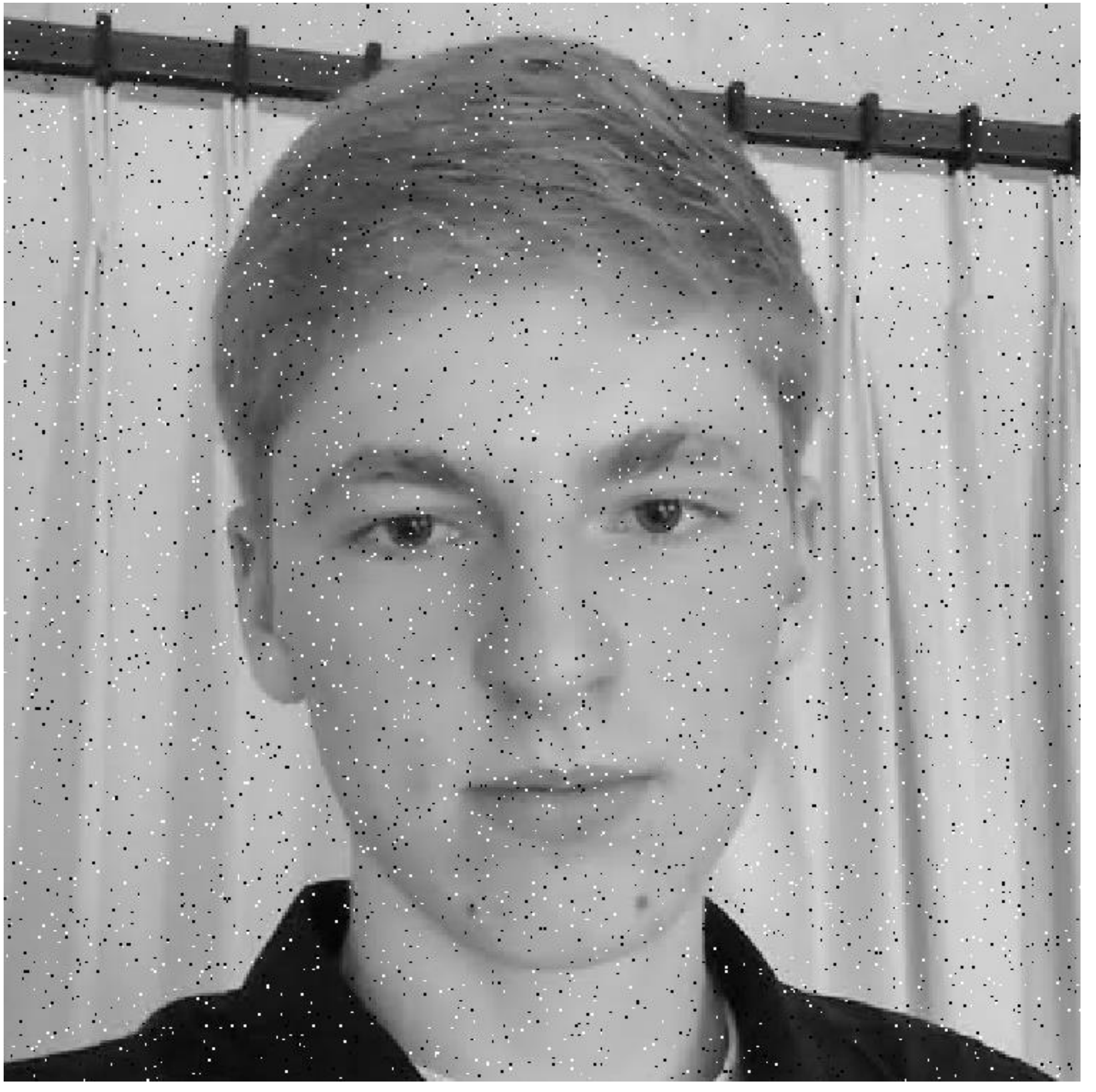
				?					

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?

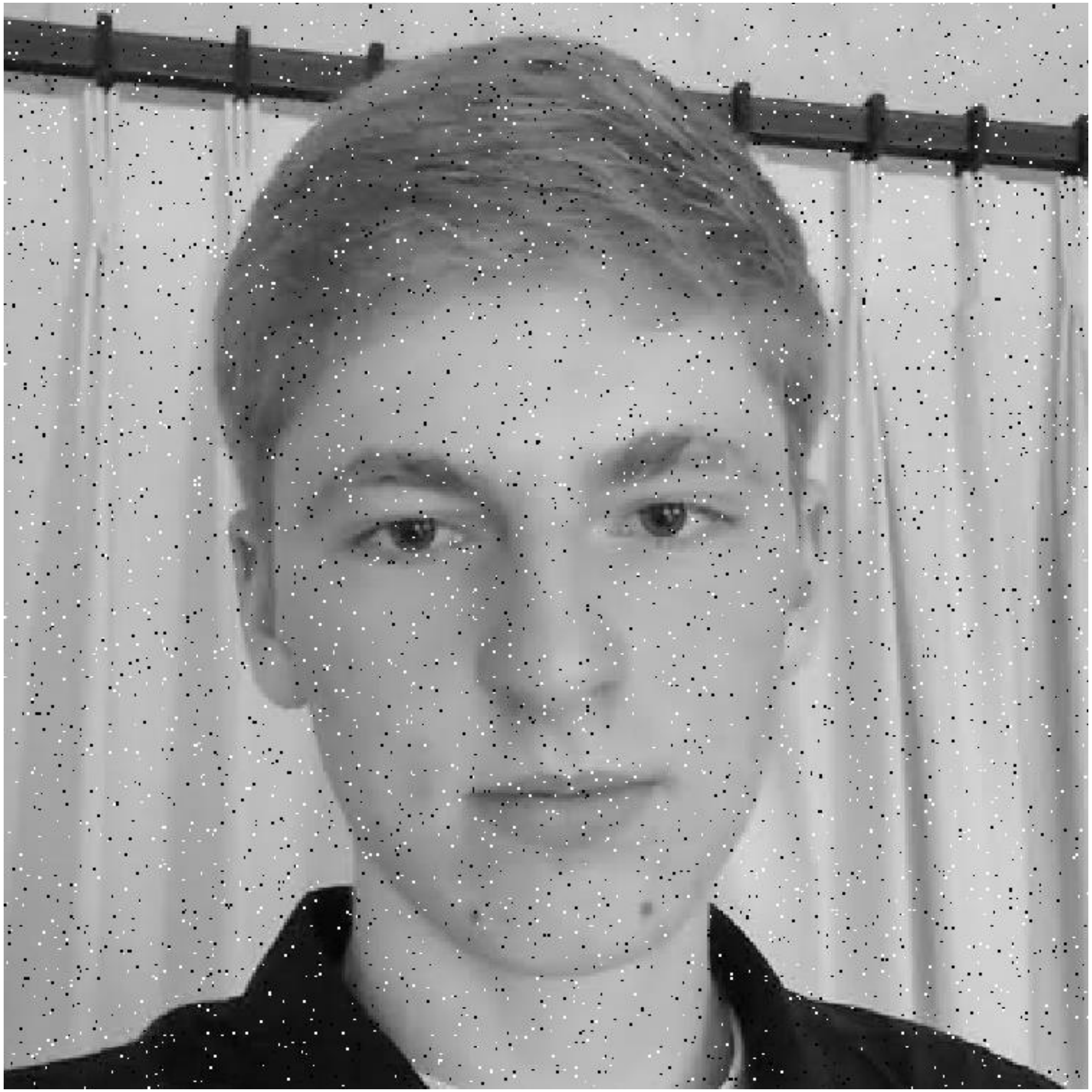
Noisy Jack – Salt and Pepper



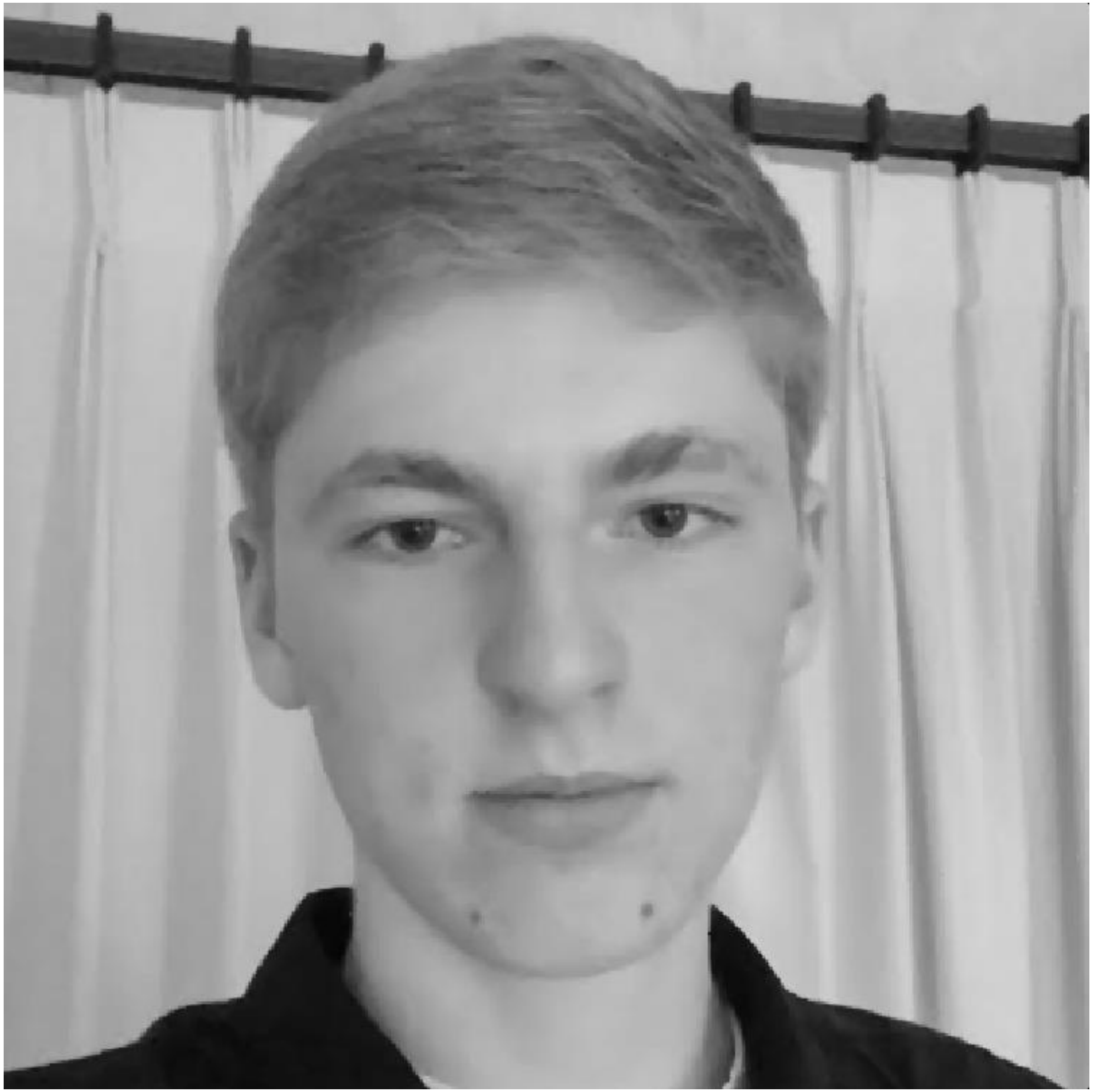
Mean Jack – 3 x 3 filter

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?

Steve Seitz, Steve Marschner

Review: questions

1. Write down a 3x3 filter that both:
   - Returns a positive value if the average value of the 4-adjacent neighbors is less than the center,
   - Returns a negative value otherwise.

Review: questions

1. Write down a 3x3 filter that both:
   - Returns a positive value if the average value of the 4-adjacent neighbors is less than the center,
   - Returns a negative value otherwise.

[0 -1/4 0;
-1/4 1 -1/4;
0 -1/4 0]

Slide: Hoiem