

# Lecture 4: Pixels and Filters

Professor Fei-Fei Li  
Stanford Vision Lab

# What we will learn today?

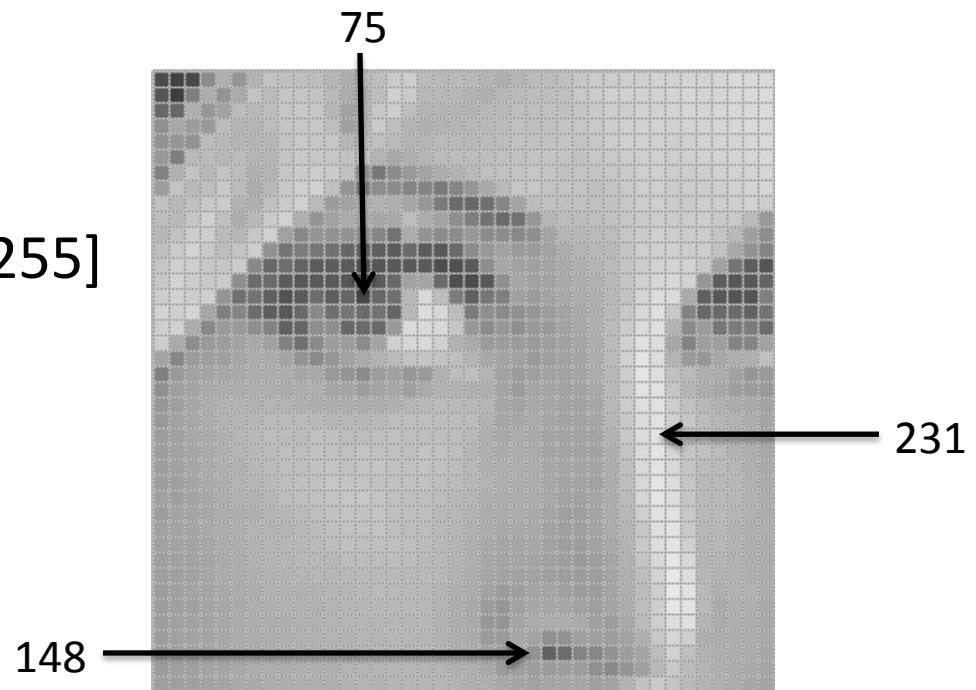
- Images as functions
- Linear systems (filters)
- Convolution and correlation

Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 7

# Images as functions

- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - “grayscale”  
(or “intensity”):  $[0,255]$



# Images as functions

- An image contains discrete number of pixels

- A simple example

- Pixel value:

- “grayscale”

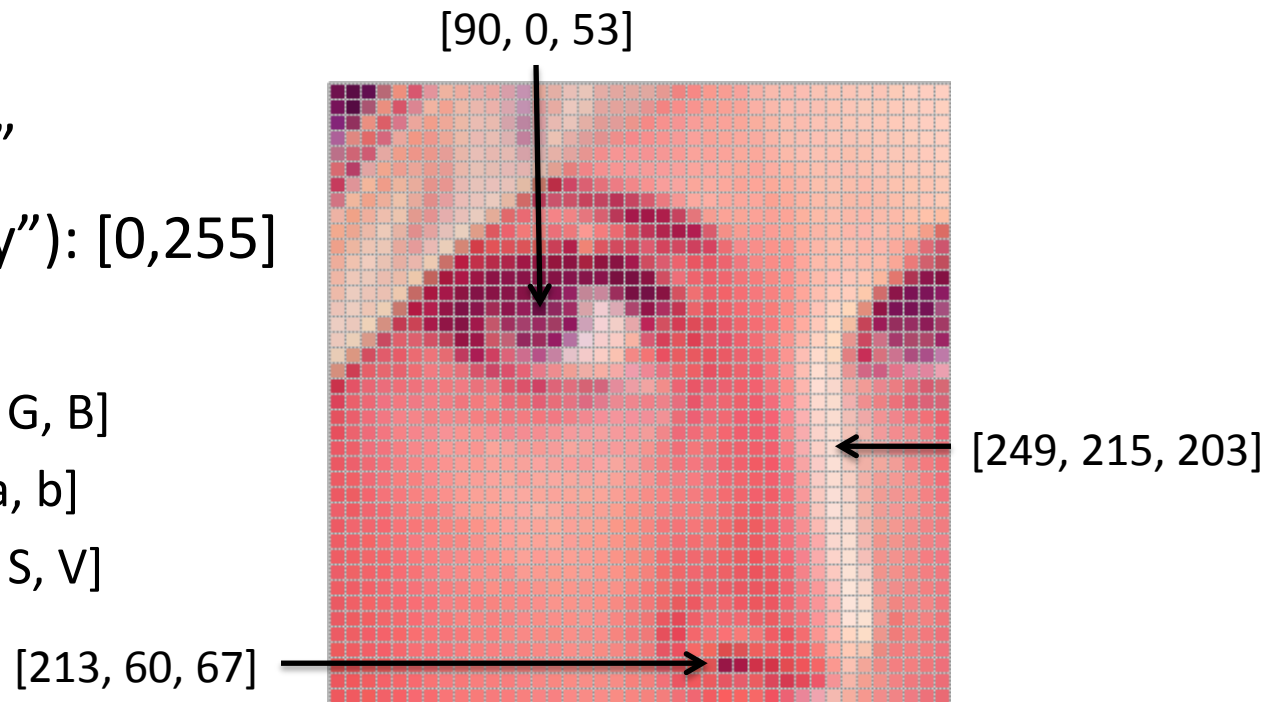
(or “intensity”):  $[0, 255]$

- “color”

- RGB:  $[R, G, B]$

- Lab:  $[L, a, b]$

- HSV:  $[H, S, V]$



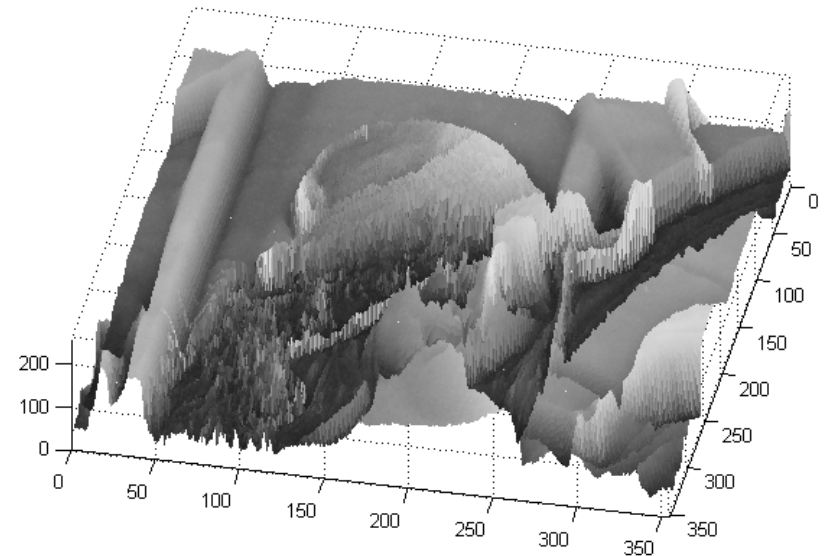
# Images as functions

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

$$f: [a,b] \times [c,d] \rightarrow [0,255]$$

Domain  
support

range



# Images as functions

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

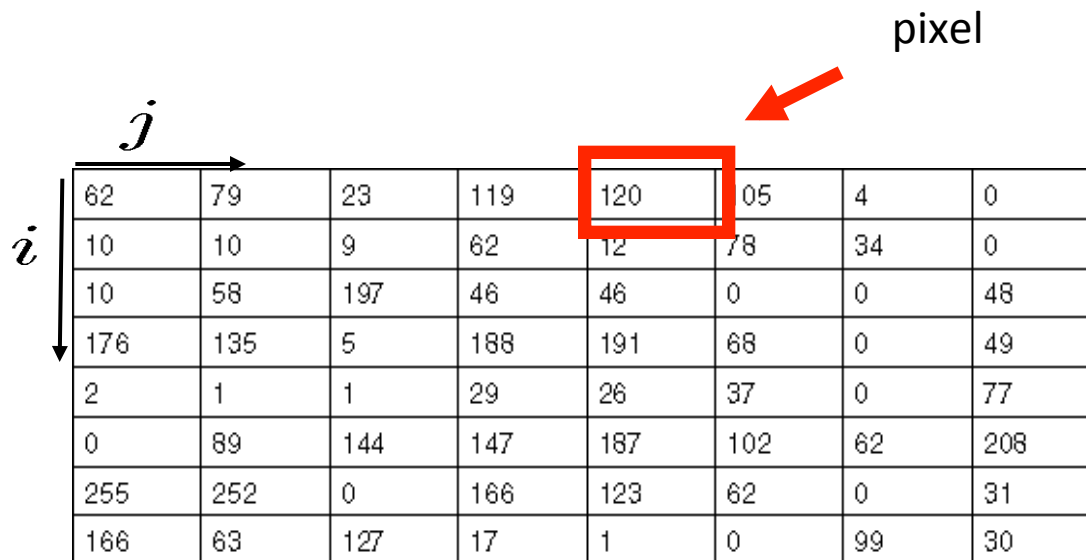
$$f: \underbrace{[a,b] \times [c,d]}_{\substack{\text{Domain} \\ \text{support}}} \rightarrow \underbrace{[0,255]}_{\text{range}}$$

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

# 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	05	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 discrete functions

Cartesian coordinates

$f[n, m] =$

↑  
Notation for discrete functions

$$\begin{bmatrix} \ddots & & \vdots & & \\ & f[-1, 1] & f[0, 1] & f[1, 1] & \\ \dots & f[-1, 0] & \underline{f[0, 0]} & f[1, 0] & \dots \\ & f[-1, -1] & f[0, -1] & f[1, -1] & \\ & & \vdots & & \ddots \end{bmatrix}$$



# What we will learn today?

- Images as functions
- **Linear systems (filters)**
- Convolution and correlation

Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 7

# Systems and Filters

- **Filtering:**

- Form a new image whose pixels are a combination original pixel values

## Goals:

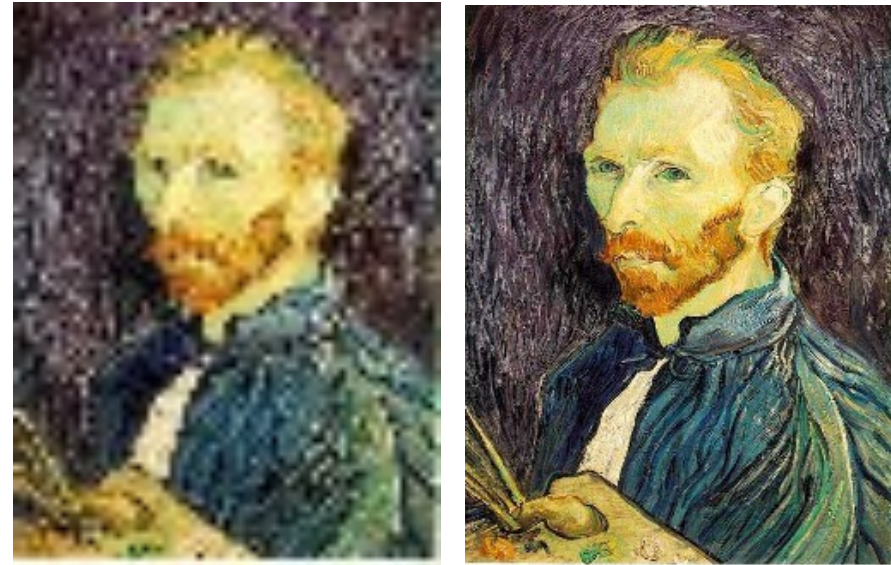
- Extract useful information from the images
  - Features (edges, corners, blobs...)
- Modify or enhance image properties:
  - super-resolution; in-painting; de-noising

## De-noising



Salt and pepper noise

## Super-resolution



## In-painting



Bertamio et al

# 2D discrete-space systems (filters)

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

$$g = \mathcal{S}[f], \quad g[n, m] = \mathcal{S}\{f[n, m]\}$$

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

# Filter example #1: Moving Average

- 2D DS moving average over a  $3 \times 3$  window of neighborhood

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

$$= \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n-k, m-l]$$

h

1	1	1
1	1	1
1	1	1

$\frac{1}{9}$

$$(f * h)[m, n] = \frac{1}{9} \sum_{k,l} f[k, l] h[m-k, n-l]$$

# Filter example #1: Moving Average

$F[x, y]$

$G[x, y]$

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


$$(f * g)[m, n] = \sum_{k, l} f[k, l] g[m - k, n - l]$$

# Filter example #1: Moving Average

$F[x, y]$

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

$G[x, y]$

	0	10							

$$(f * g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]$$

# Filter example #1: Moving Average

$F[x, y]$

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

$G[x, y]$

	0	10	20						

$$(f * g)[m, n] = \sum_{k,l} f[k, l] g[m - k, n - l]$$



# Filter example #1: Moving Average

$F[x, y]$

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

$G[x, y]$

	0	10	20	30					

$$(f * g)[m, n] = \sum_{k, l} f[k, l] g[m - k, n - l]$$

# Filter example #1: Moving Average

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

$$(f * g)[m, n] = \sum_{k, l} f[k, l] g[m - k, n - l]$$

# Filter example #1: Moving Average

$F[x, y]$

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

$G[x, y]$

	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	

$$(f * g)[m, n] = \sum_{k, l} f[k, l] g[m - k, n - l]$$

Source: S. Seitz

# Filter example #1: Moving Average

In summary:

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

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

1	1	1
1	1	1
1	1	1

# Filter example #1: Moving Average



# Filter example #2: Image Segmentation

- Image segmentation based on a simple threshold:

$$g[n, m] = \begin{cases} 1, & f[n, m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$



# Classification of systems

- Amplitude properties

- Linearity

- Stability

- Invertibility

- Spatial properties

- Causality

- Separability

- Memory

- Shift invariance

- Rotation invariance

# Shift-invariance

If  $f[n, m] \xrightarrow{\mathcal{S}} g[n, m]$  then

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

for every input image  $f[n, m]$  and shifts  $n_0, m_0$



# Is the moving average system is shift invariant?

$$F[x, y]$$

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

$$G[x, y]$$

	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	

# Is the moving average system is shift invariant?

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

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

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

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

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

**Yes!**

# Linear Systems (filters)

$$f(x, y) \rightarrow \boxed{\mathcal{S}} \rightarrow g(x, y)$$

- Linear filtering:
  - Form a new image whose pixels are a weighted sum of original pixel values
  - Use the same set of weights at each point
- **S** is a linear system (function) iff it *S satisfies*

$$\mathcal{S}[\alpha f_1 + \beta f_2] = \alpha \mathcal{S}[f_1] + \beta \mathcal{S}[f_2]$$

superposition property

# Linear Systems (filters)

$$f(x, y) \rightarrow \boxed{S} \rightarrow g(x, y)$$

- Is the moving average a linear system?
- Is thresholding a linear system?
  - $f_1[n,m] + f_2[n,m] > T$
  - $f_1[n,m] < T$
  - $f_2[n,m] < T$       No!

# LSI (linear *shift invariant*) systems

## Impulse response

$$\delta_2[n, m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n, m]$$

$$\delta_2[n - k, m - l] \rightarrow \boxed{\mathcal{S} \text{ (SI)}} \rightarrow h[n - k, m - l]$$

# LSI (linear *shift invariant*) systems

**Example:** impulse response of the 3 by 3 moving average filter:

$$h[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 \delta_2[n - k, m - l]$$

$$= \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

h

	1	1	1
1	1	1	1
1	1	1	1

# LSI (linear *shift invariant*) systems

An LSI system is completely specified by its impulse response.

sifting property of the delta function

$$f[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \delta_2[n - k, m - l]$$

superposition

$$\delta_2[n, m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n, m] \quad \rightarrow \boxed{\mathcal{S} \text{ LSI}} \rightarrow \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

Discrete convolution

$$= f[n, m] ** h[n, m]$$

# What we will learn today?

- Images as functions
- Linear systems (filters)
- Convolution and correlation

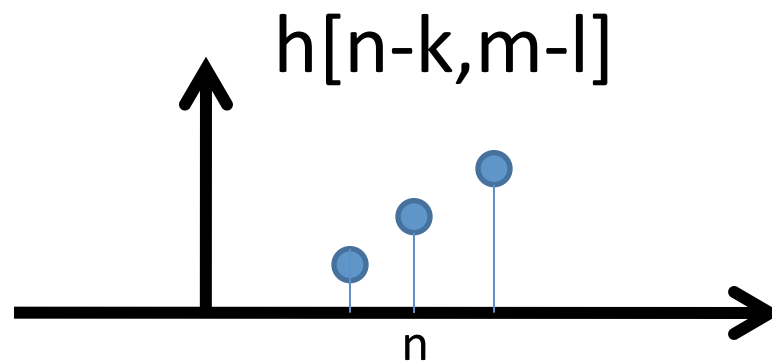
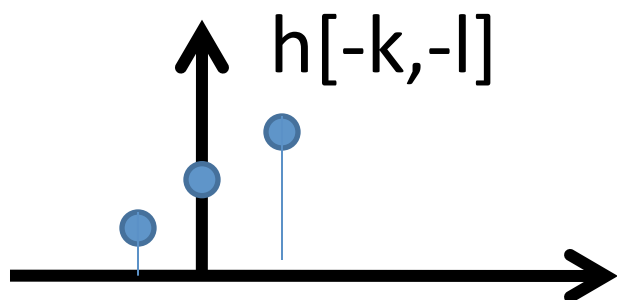
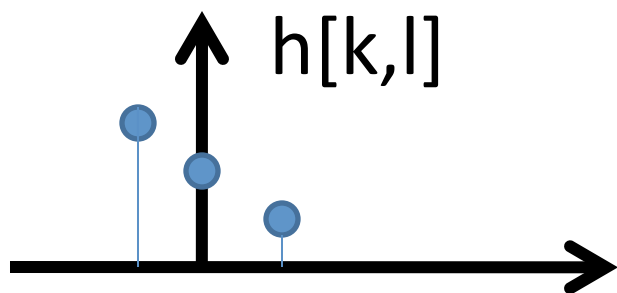
Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 7



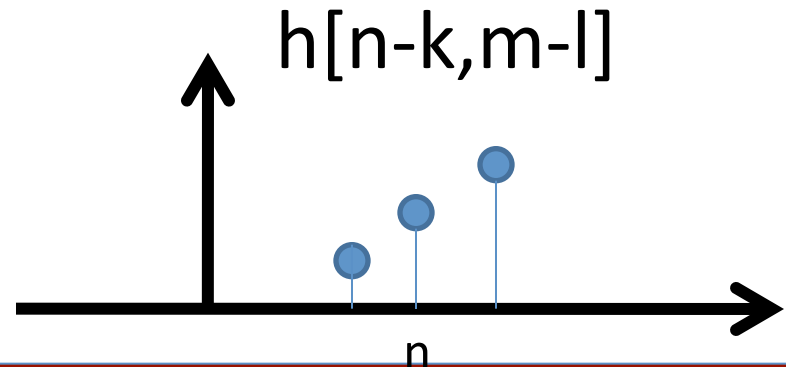
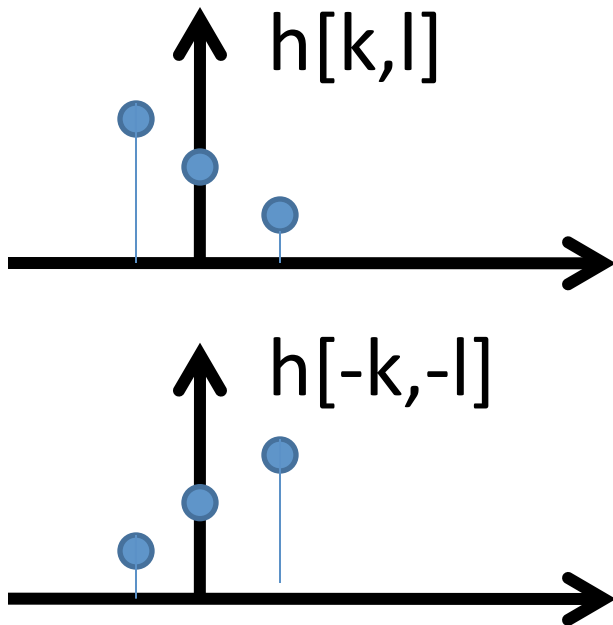
# Discrete convolution

- Fold  $h[n,m]$  about origin to form  $h[-k,-l]$
- Shift the folded results by  $n,m$  to form  $h[n-k,m-l]$
- Multiply  $h[n-k,m-l]$  by  $f[k,l]$
- Sum over all  $k,l$
- Repeat for every  $n,m$



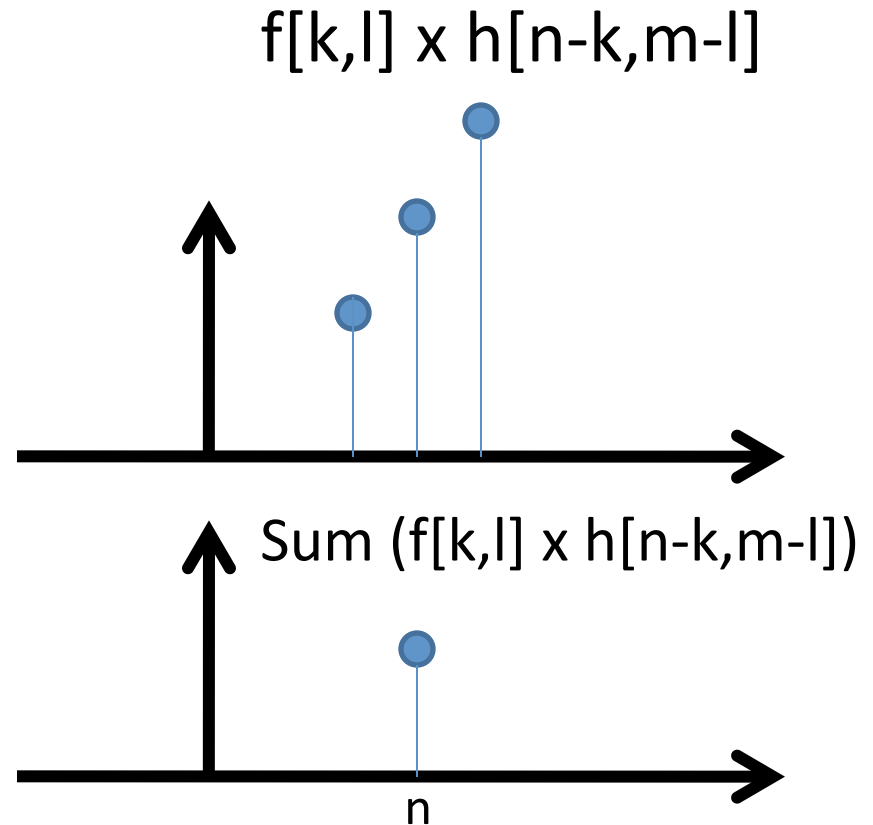
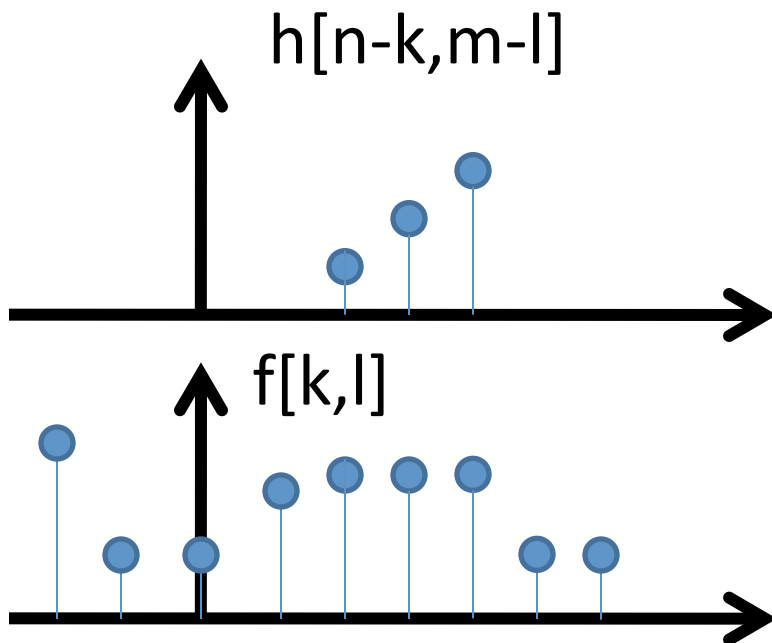
# Discrete convolution

- Fold  $h[n,m]$  about origin to form  $h[-k,-l]$
- Shift the folded results by  $n,m$  to form  $h[n-k,m-l]$
- Multiply  $h[n-k,m-l]$  by  $f[k,l]$
- Sum over all  $k,l$
- Repeat for every  $n,m$



# Discrete convolution

- Fold  $h[n,m]$  about origin to form  $h[-k,-l]$
- Shift the folded results by  $n,m$  to form  $h[n-k,m-l]$
- Multiply  $h[n-k,m-l]$  by  $f[k,l]$
- Sum over all  $k,l$
- Repeat for every  $n,m$



# Convolution in 2D - examples



Original

•0	•0	•0
•0	•1	•0
•0	•0	•0

=

?

# 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

=

?

# Convolution in 2D - examples



Original

•0	•0	•0
•0	•0	•1
•0	•0	•0

=



Shifted right  
By 1 pixel

# Convolution in 2D - examples



Original

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



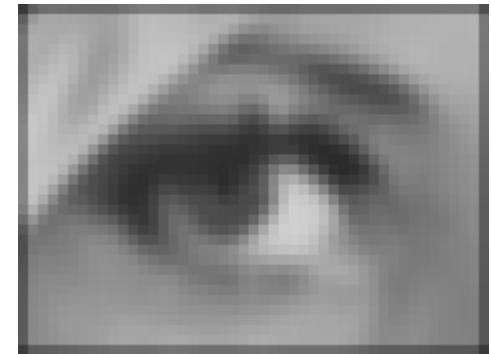
# Convolution in 2D - examples



Original

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

=



Blur (with a box filter)

Courtesy of D Lowe

# Convolution in 2D - examples



Original

•0	•0	•0
•0	•2	•0
•0	•0	•0

-

$\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

= ?

(Note that filter sums to 1)

“details of the image”

•0	•0	•0
•0	•1	•0
•0	•0	•0

+

•0	•0	•0
•0	•1	•0
•0	•0	•0

-

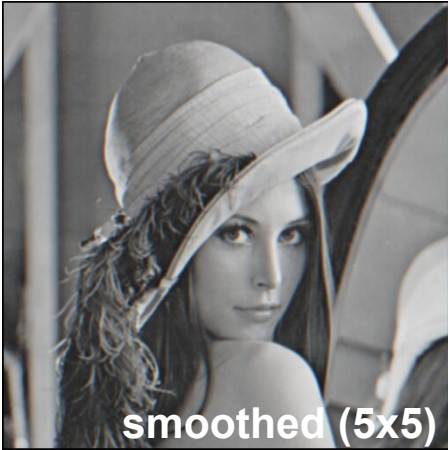
$\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

- What does blurring take away?



-



=



- Let's add it back:



+ a



=



# Convolution in 2D – Sharpening filter



Original

•0	•0	•0
•0	•2	•0
•0	•0	•0

–

$\frac{1}{9}$

•1	•1	•1
•1	•1	•1
•1	•1	•1

=



**Sharpening filter:** Accentuates differences with local average

# Convolution properties

- **Commutative property:**

$$f ** h = h ** f$$

- **Associative property:**

$$(f ** h_1) ** h_2 = f ** (h_1 ** h_2)$$

- **Distributive property:**

$$f ** (h_1 + h_2) = (f ** h_1) + (f ** h_2)$$

The order doesn't matter!  $h_1 ** h_2 = h_2 ** h_1$

# Convolution properties

- **Shift property:**

$$f[n, m] ** \delta_2[n - n_0, m - m_0] = f[n - n_0, m - m_0]$$

- **Shift-invariance:**

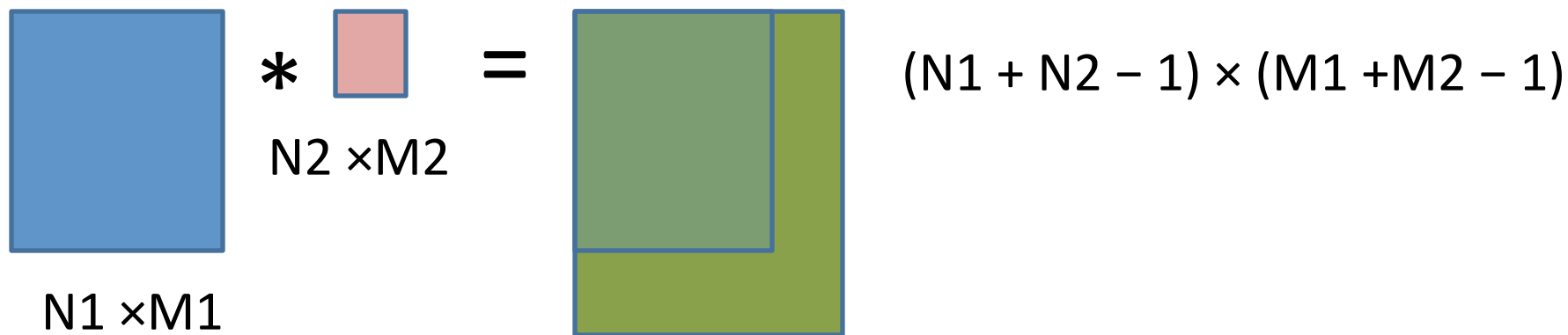
$$g[n, m] = f[n, m] ** h[n, m]$$

$$\implies f[n - l_1, m - l_1] ** h[n - l_2, m - l_2]$$

$$= g[n - l_1 - l_2, m - l_1 - l_2]$$

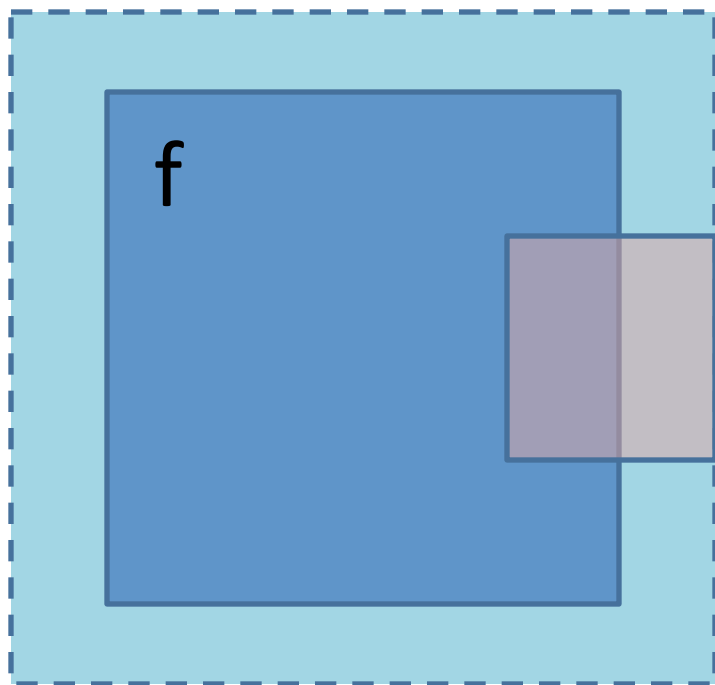
# Image support and edge effect

- A computer will only convolve **finite support signals**.
  - That is: images that are zero for  $n, m$  outside some rectangular region
- MATLAB's `conv2` performs 2D DS convolution of finite-support signals.



# Image support and edge effect

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



h

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

-> Matlab conv2 uses zero-padding



# What we will learn today?

- Images as functions
- Linear systems (filters)
- Convolution and **correlation**

Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 7

# Cross correlation

Cross correlation of two 2D signals  $f[n,m]$  and  $g[n,m]$

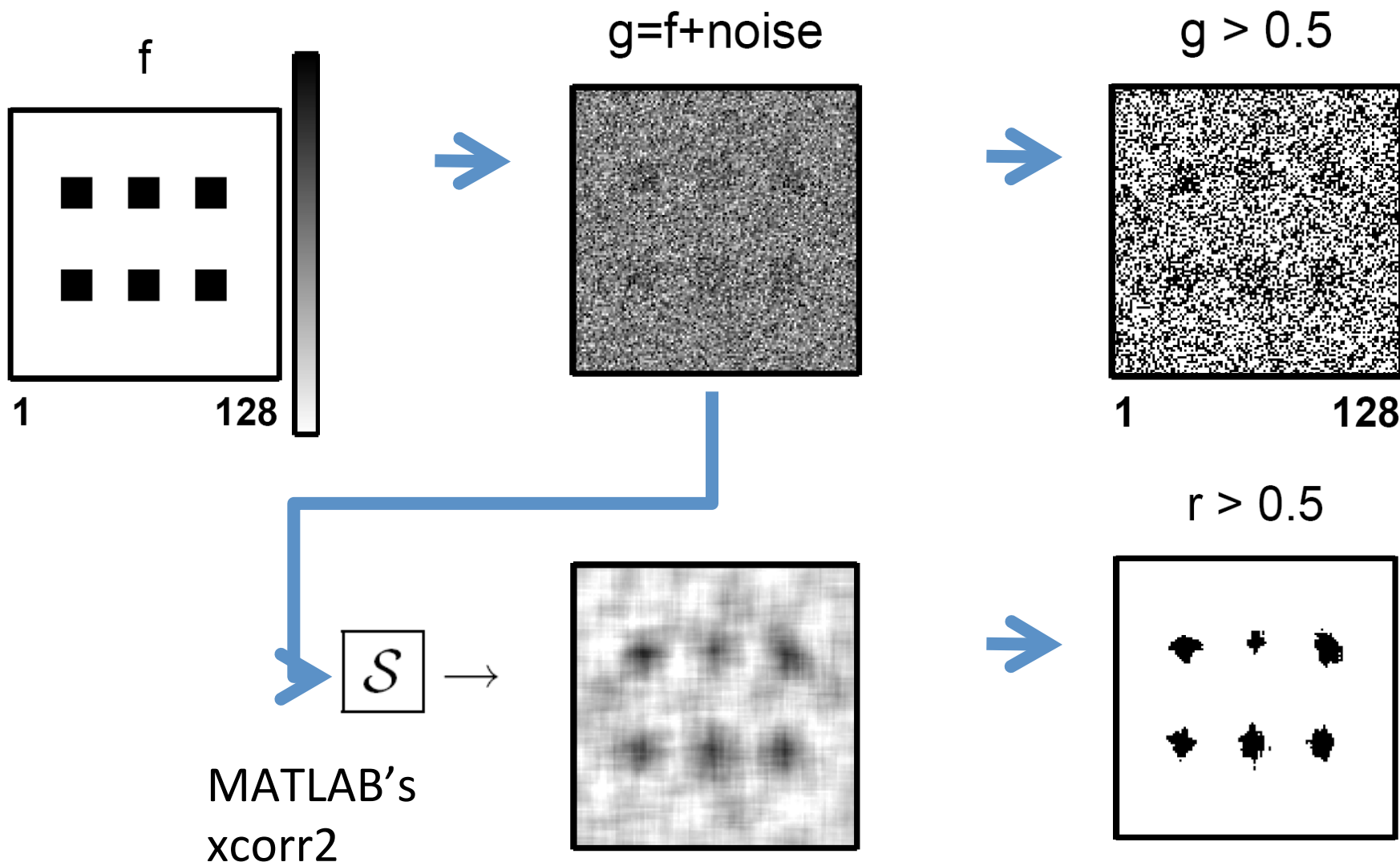
$$r_{fg}[k, l] \triangleq \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n, m] g^*[n - k, m - l]$$
$$= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n + k, m + l] g^*[n, m], \quad k, l \in \mathbb{Z}.$$

(k, l) is called the **lag**

- Equivalent to a convolution without the flip

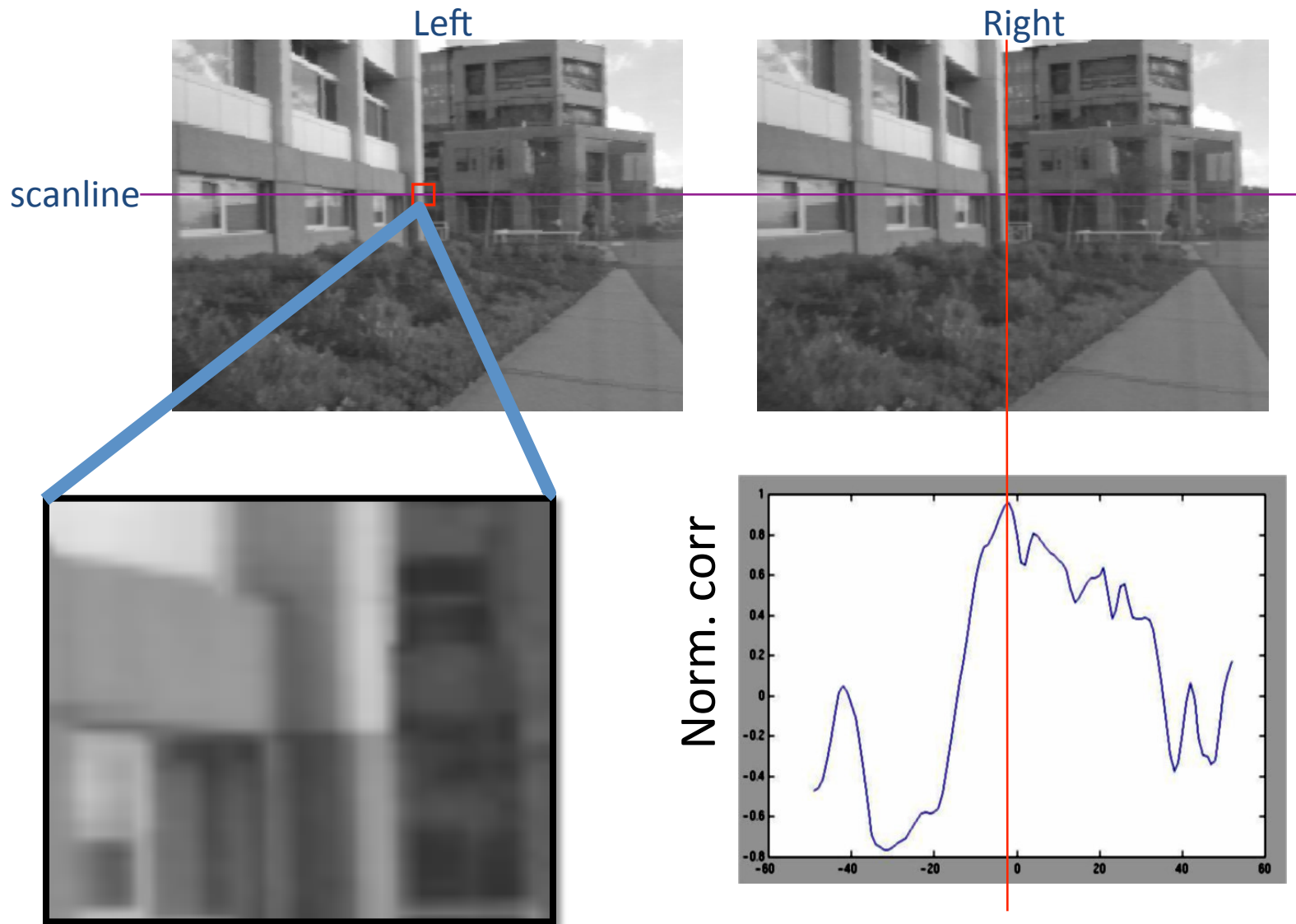
$$r_{fg}[n, m] = f[n, m] ** g^*[-n, -m]$$

# Cross correlation – example



Courtesy of J. Fessler

# Cross correlation – example



# Convolution vs. Correlation

- A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation
- **Correlation** compares the *similarity of two sets of data*. Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best .
  - correlation is a measure of relatedness of two signals

# Application: Vision system for TV remote control

- uses template matching

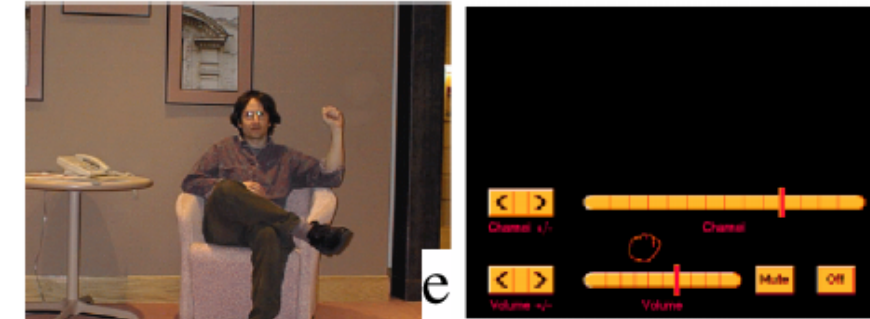
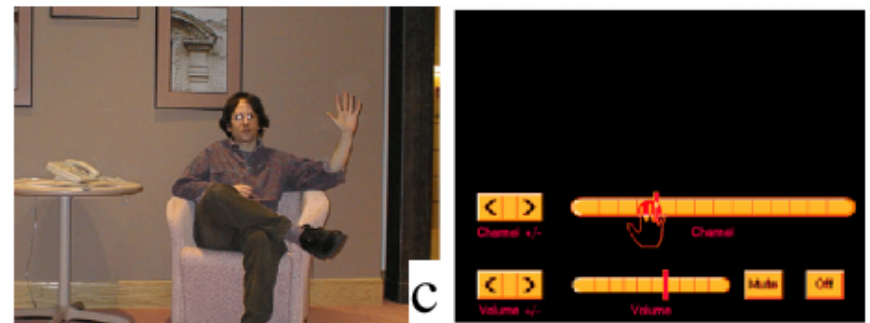
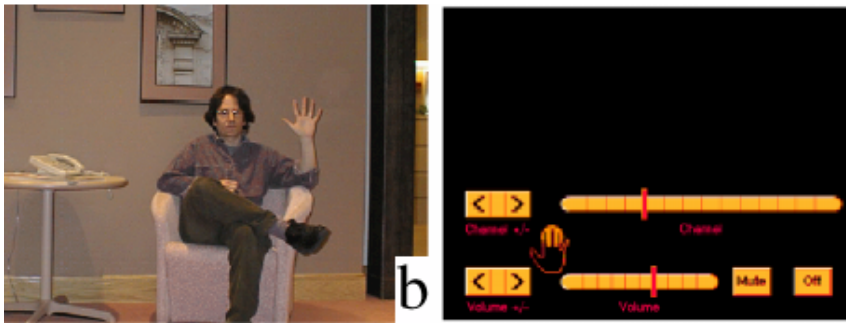


Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

# What we have learned today?

- Images as functions
- Linear systems (filters)
- Convolution and correlation