Lecture: Pixels and Filters

Juan Carlos Niebles and Ranjay Krishna
Stanford Vision Lab

Announcements

- HW1 due Monday
- HW2 is out
- Class notes – Make sure to find the source and cite the images you use.

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

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

Types of Images

Binary

Types of Images

Binary

Gray Scale

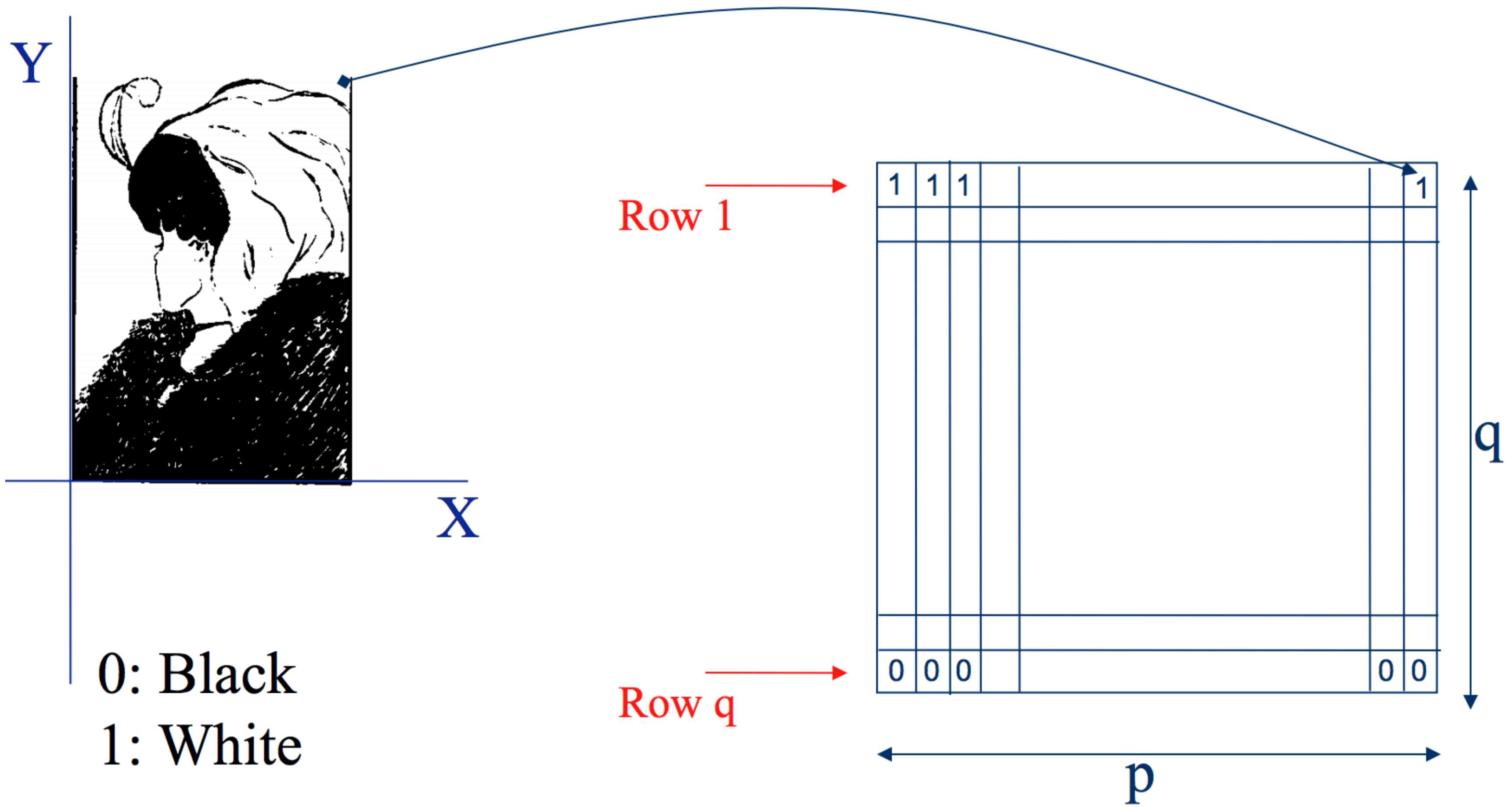
Types of Images

Binary

Gray Scale

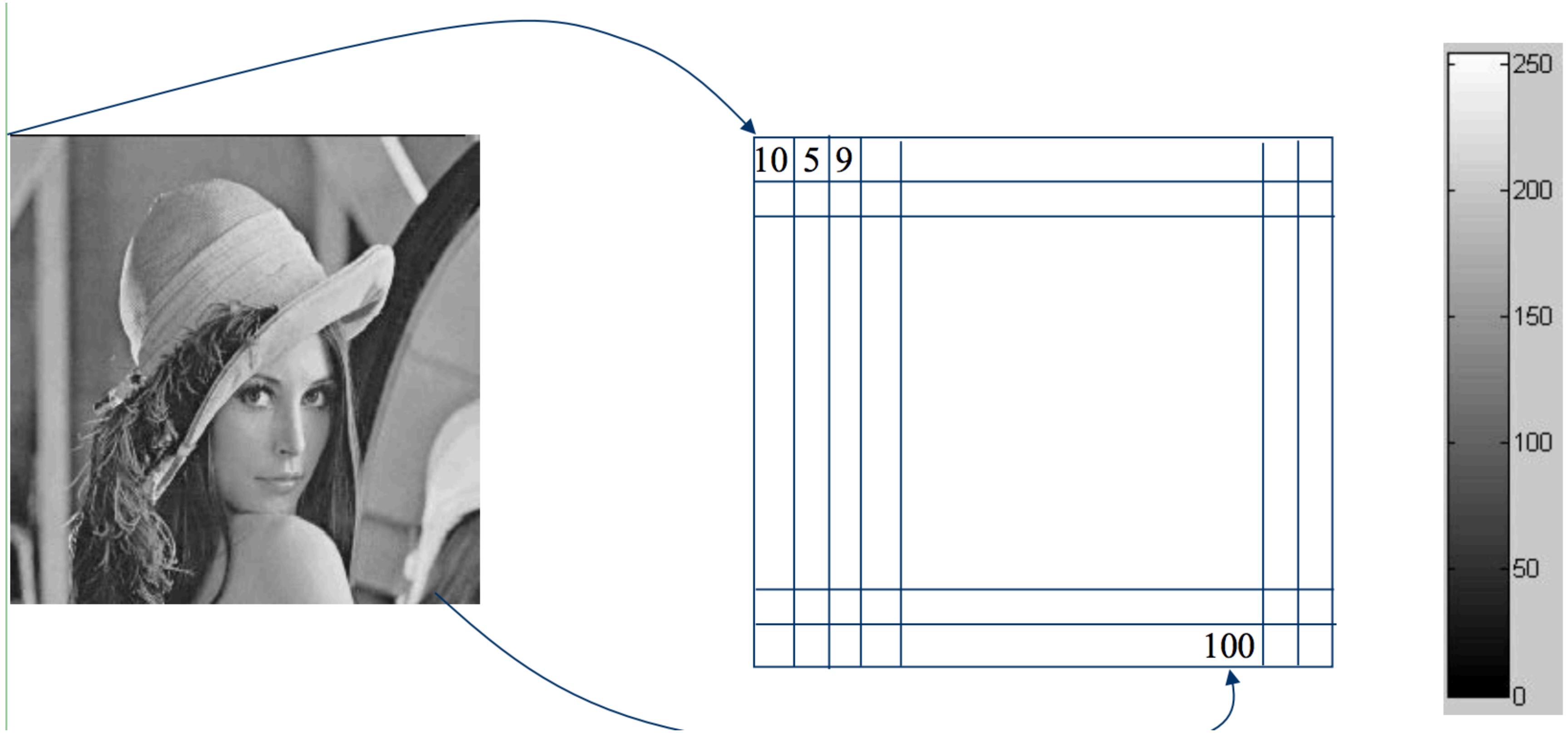
Color

Binary image representation



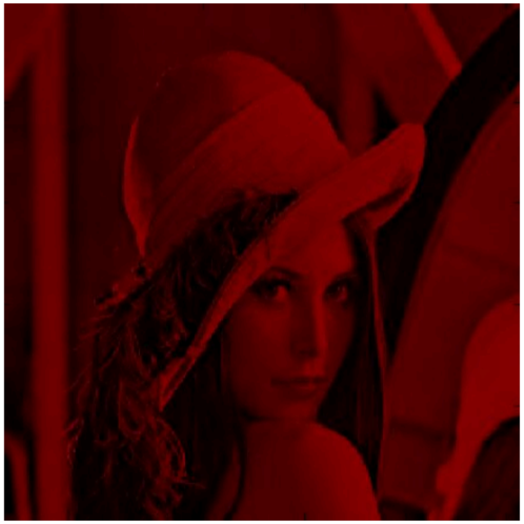
Slide credit: Ulas Bagci

Grayscale image representation



Slide credit: Ulas Bagci

Color Image - one channel



Slide credit: Ulas Bagci

Color image representation

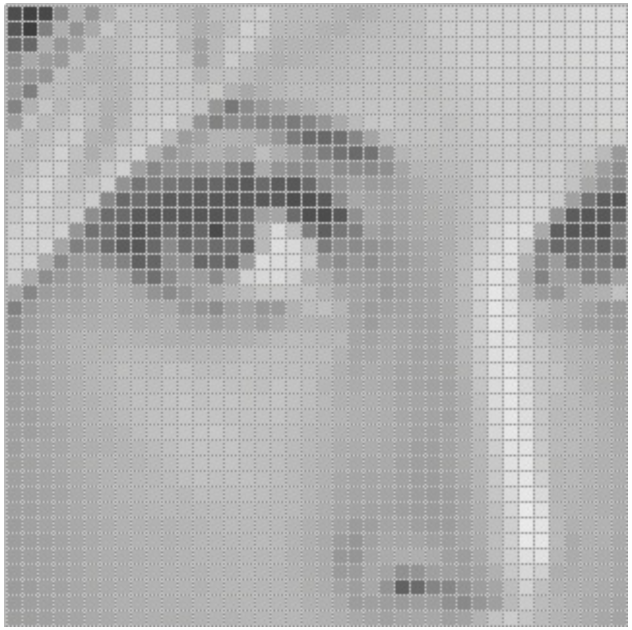
Phil Noble / AP

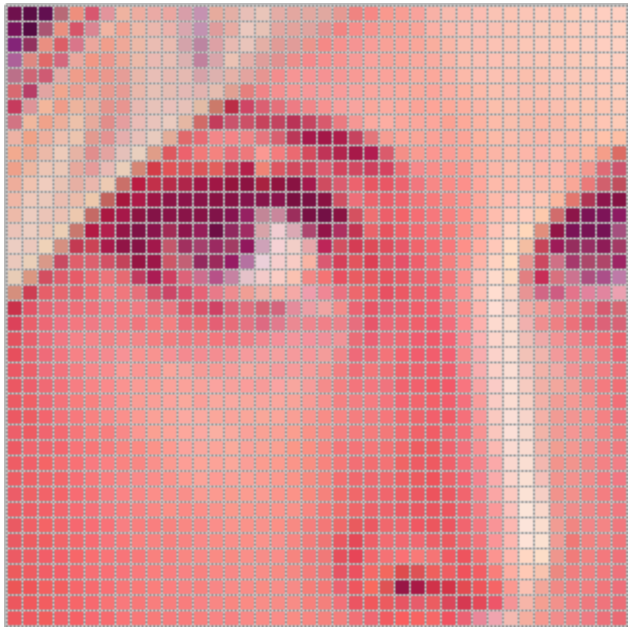


Slide credit: Ulas Bagci

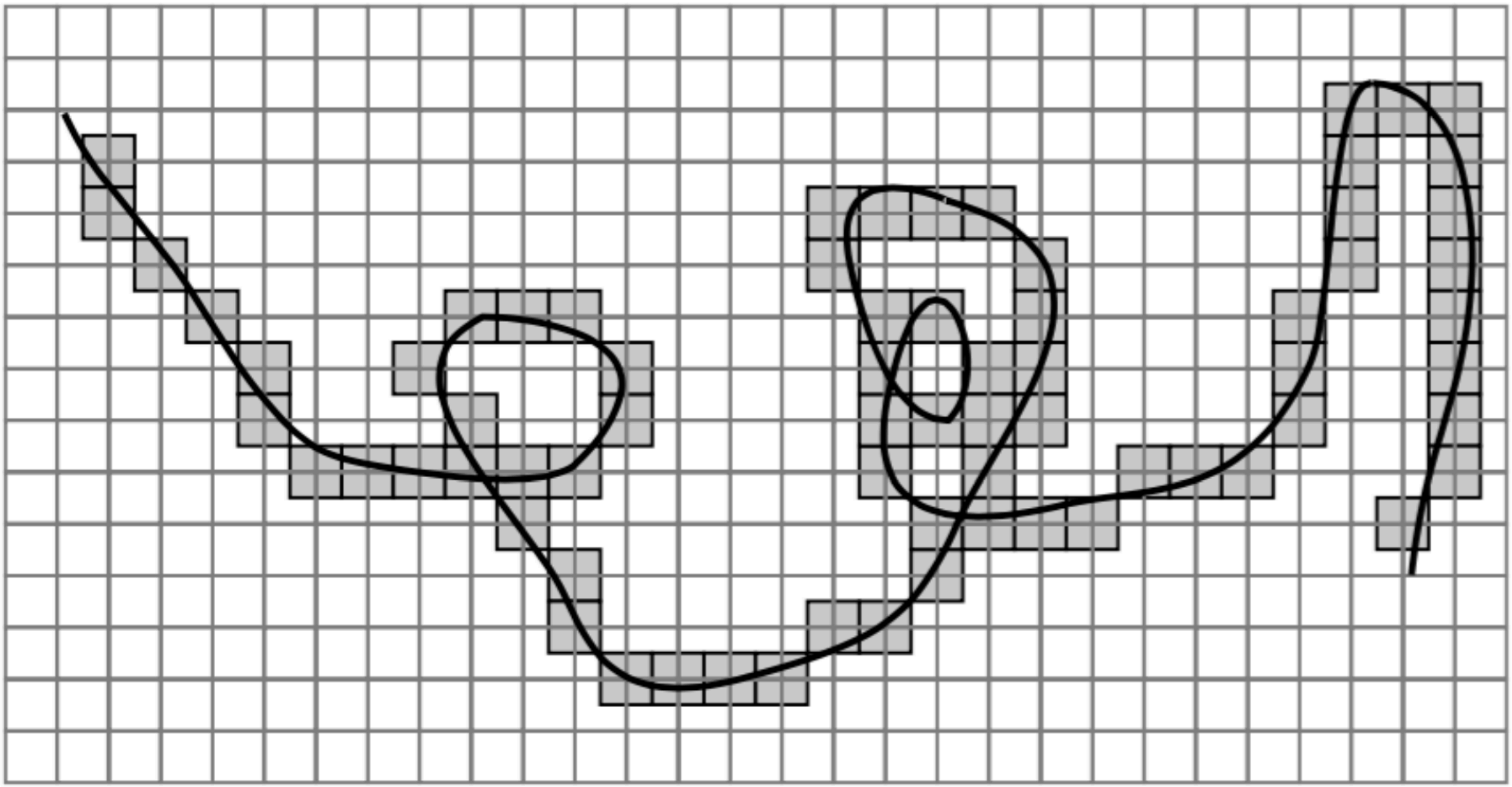
Images are sampled

What happens when we zoom into the images we capture?





Errors due Sampling



Slide credit: Ulas Bagci

Resolution

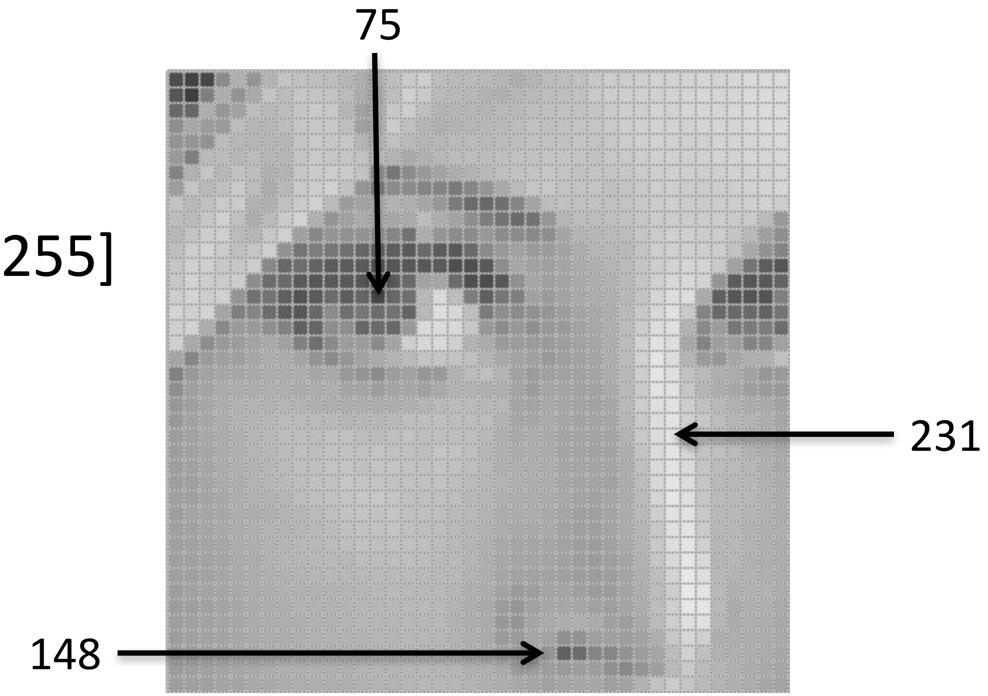
is a **sampling** parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density, and its standard value for recent screen technologies is 72 dpi

Slide credit: Ulas Bagci

Images are Sampled and Quantized

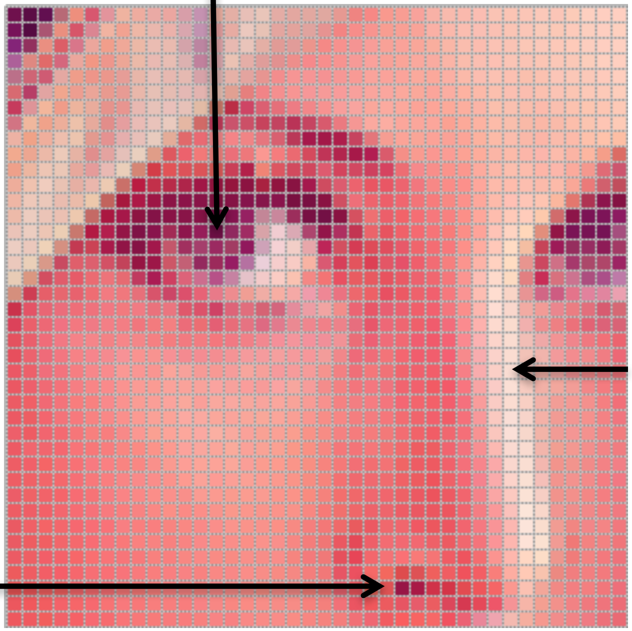
- An image contains discrete number of pixels
  - A simple example
  - Pixel value:
    - "grayscale"

    (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]



With this loss of information (from sampling and quantization),

Can we still use images for useful tasks?

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

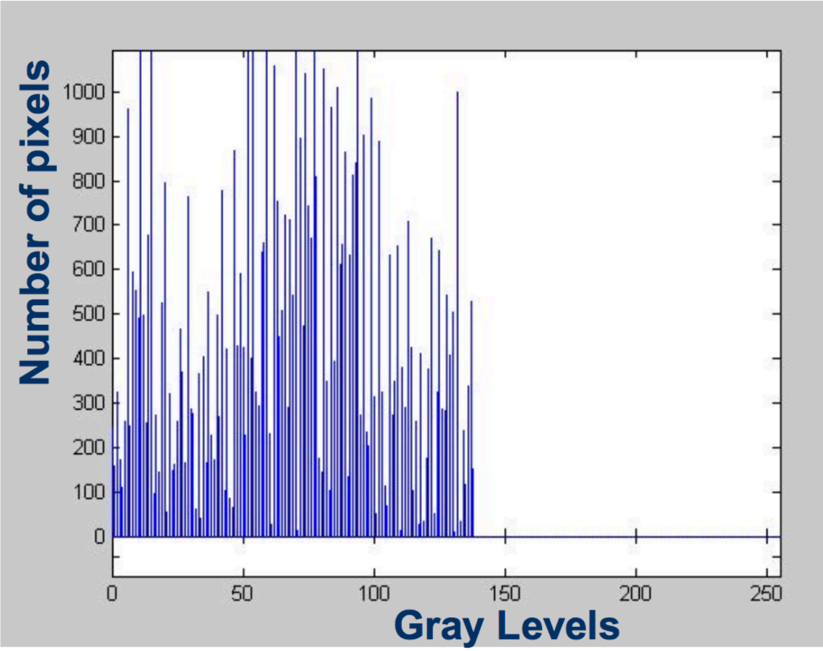
Histogram

- Histogram of an image provides the frequency of the brightness (intensity) value in the image.

```
def histogram(im):
    h = np.zeros(255)
    for row in im.shape[0]:
        for col in im.shape[1]:
            val = im[row, col]
            h[val] += 1
```

Histogram

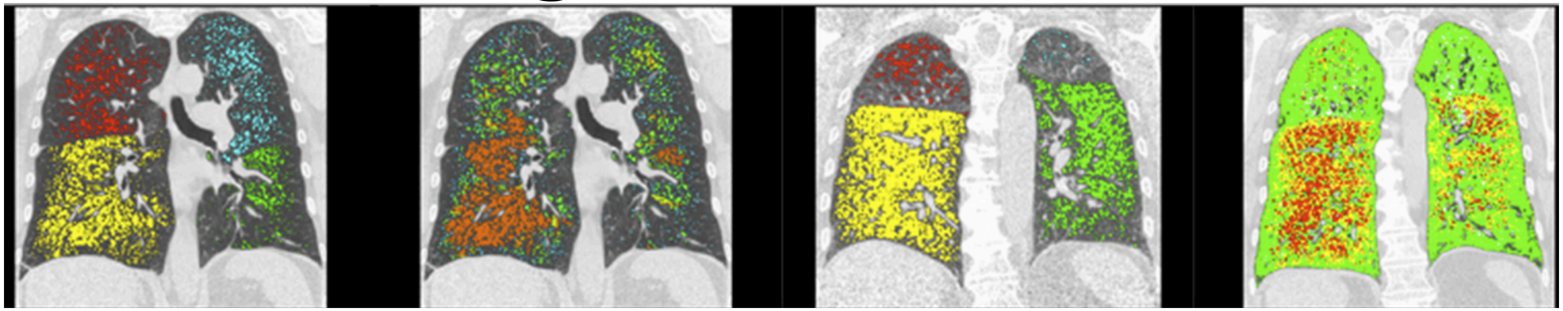
- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image

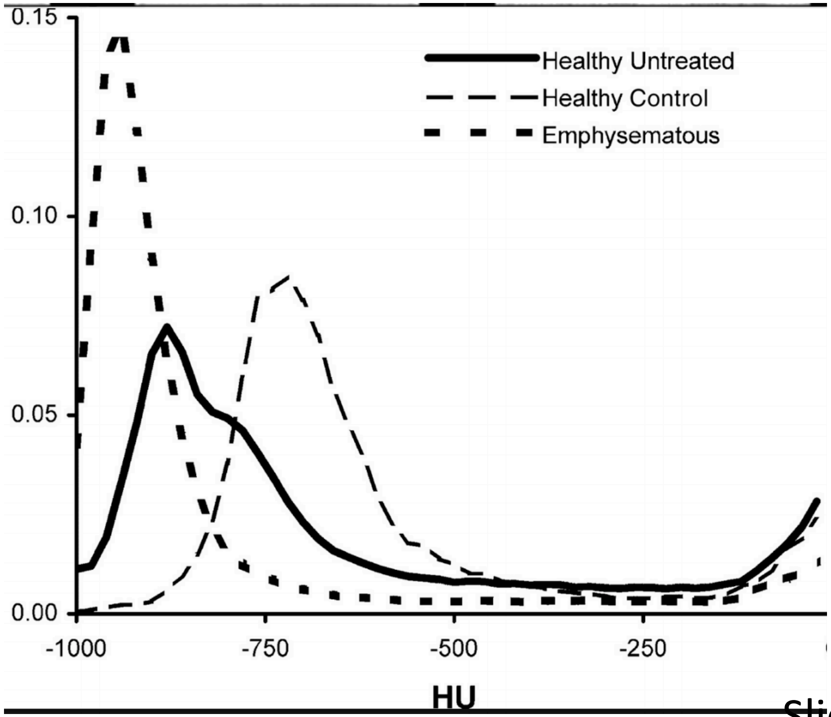


Histogram

Slide credit: Dr. Mubarak Shah

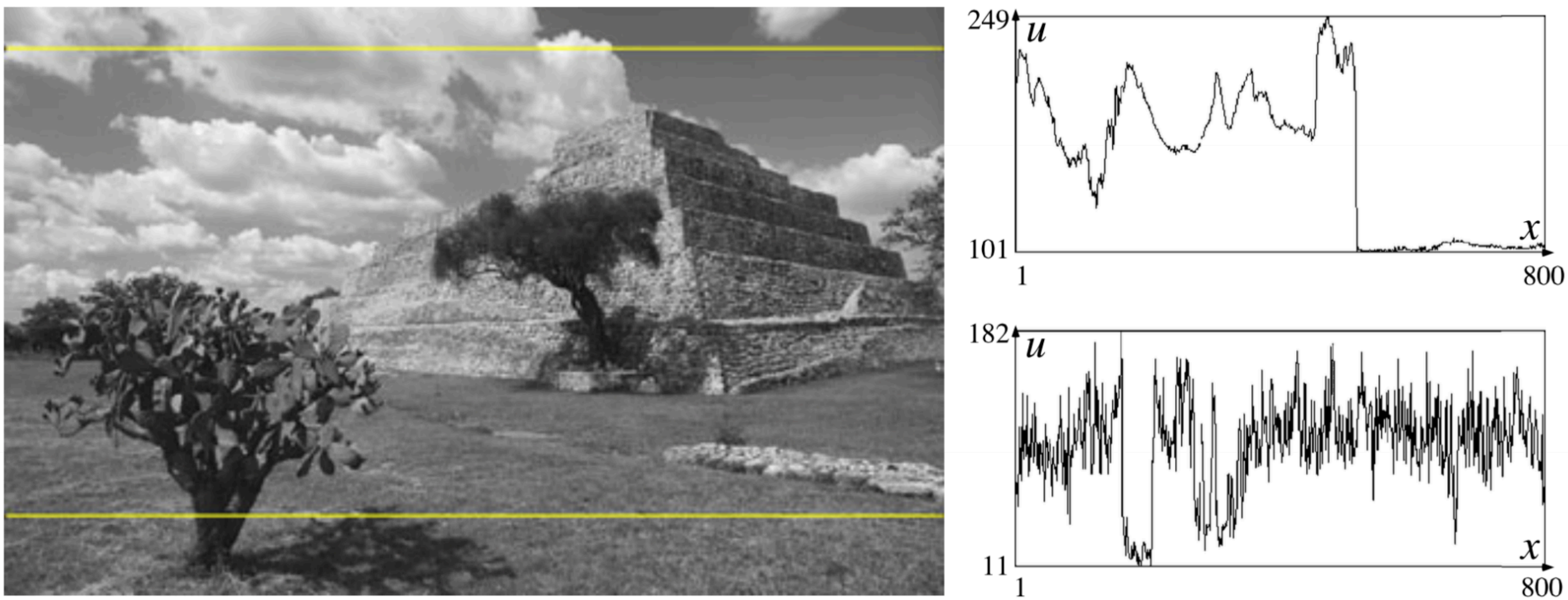
Histogram – use case





Slide credit: Dr. Mubarak Shah

Histogram – another use case



Slide credit: Dr. Mubarak Shah

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	$\boxed{120}$	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

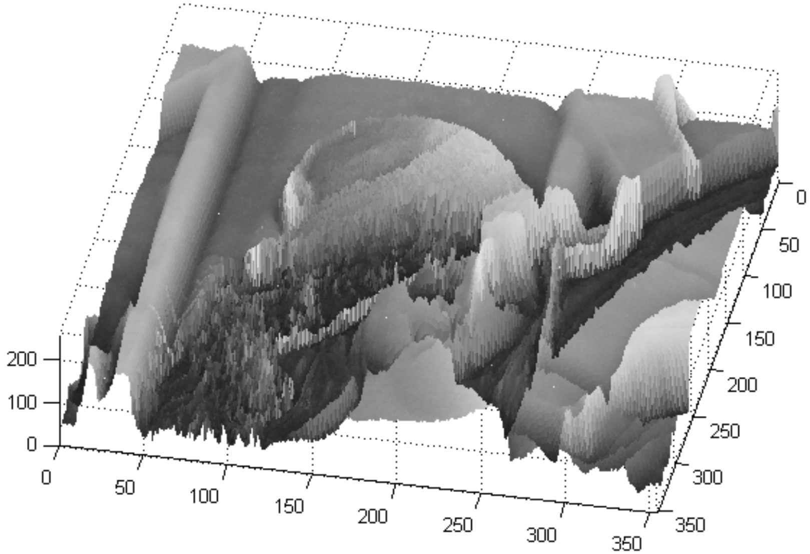
$$f[n,m] = \begin{bmatrix} \ddots & & \vdots & & \\ & f[-1,-1] & f[0,-1] & f[1,-1] & \\ \ldots & f[-1,0] & \underline{f[0,0]} & f[1,0] & \ldots \\ & f[-1,1] & f[0,1] & f[1,1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

Notation for discrete functions

Images as functions

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

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



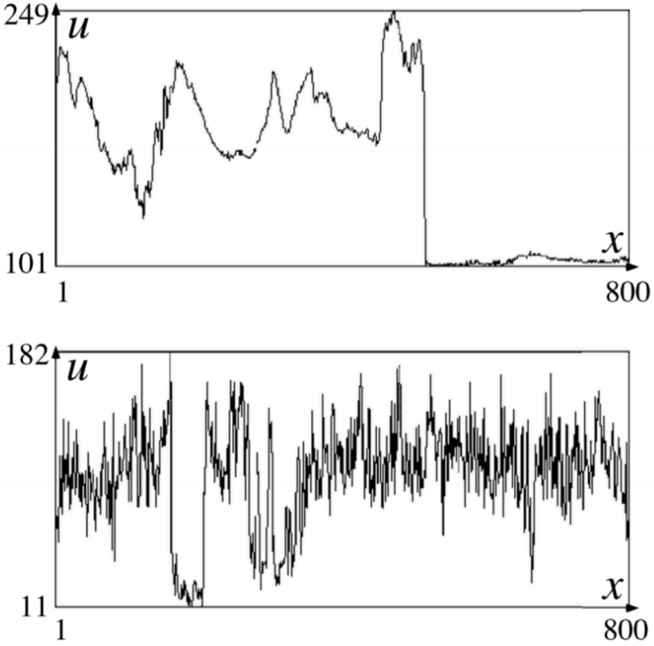
Images as functions

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

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

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

Histograms are a type of image function



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

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

System and Filters

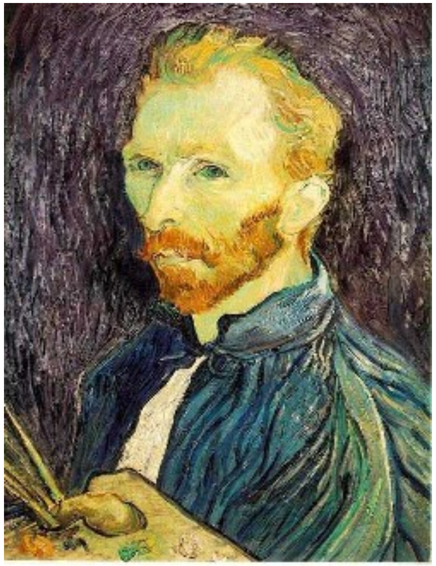
- we define a system as a unit that converts an input function f[n,m] into an output (or response) function g[n,m], where (n,m) are the independent variables.
  - In the case for images, (n,m) represents the **spatial position in the image.**

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

De-noising

Salt and pepper noise

Super-resolution



In-painting

Bertamio et al

Images as coordinates

Cartesian coordinates

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

Notation for discrete functions

2D discrete-space systems (filters)

S is the **system operator**, defined as a mapping or assignment of a member of the set of possible outputs g[n,m] to each member of the set of possible inputs f[n,m].

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

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

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

Filter example #1: Moving Average

2D DS moving average over a 3 × 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

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

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

$G[x, y]$

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

Courtesy of S. Seitz

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							

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

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						

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

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					

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

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				

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

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	

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

Source: S. Seitz

Filter example #1: Moving Average

In summary:

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

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

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

Filter example #1: Moving Average

Filter example #2: Image Segmentation

- Image segmentation based on a simple threshold:

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

Properties of systems

- Amplitude properties:
  - Additivity

$$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$

Properties of systems

- Amplitude properties:
  - Additivity

$$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$

  - Homogeneity

$$S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]]]$$

Properties of systems

- Amplitude properties:
  - Additivity

$$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$

  - Homogeneity

$$S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]]]$$

  - Superposition

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

Properties of systems

- Amplitude properties:
  - Additivity

$$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$

  - Homogeneity

$$S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]]]$$

  - Superposition

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

  - Stability

$$|f[n,m]| \leq k \implies |g[n,m]| \leq ck$$

Properties of systems

- Amplitude properties:
  - Additivity

$$S[f_i[n,m] + f_j[n,m]] = S[f_i[n,m]] + S[f_j[n,m]]$$

  - Homogeneity

$$S[\alpha f_i[n,m]] = \alpha S[f_i[n,m]]]$$

  - Superposition

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

  - Stability

$$|f[n,m]| \leq k \implies |g[n,m]| \leq ck$$

  - Invertibility

$$S^{-1}[S[f_i[n,m]]] = f[n,m]$$

Properties of systems

- Spatial properties
  - Causality

$$\text{for } n < n_0, m < m_0, \text{ if } f[n,m] = 0 \implies g[n,m] = 0$$

  - Shift invariance:

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

Is the moving average system is shift invariant?

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

$F[x,y]$

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?

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

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

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

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

Yes!

Is the moving average system is casual?

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

$F[x, y]$

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	

$$\text{for } n < n_0, m < m_0, \text{ if } f[n,m] = 0 \implies g[n,m] = 0$$

Linear Systems (filters)

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

- 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*

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

superposition property

Linear Systems (filters)

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

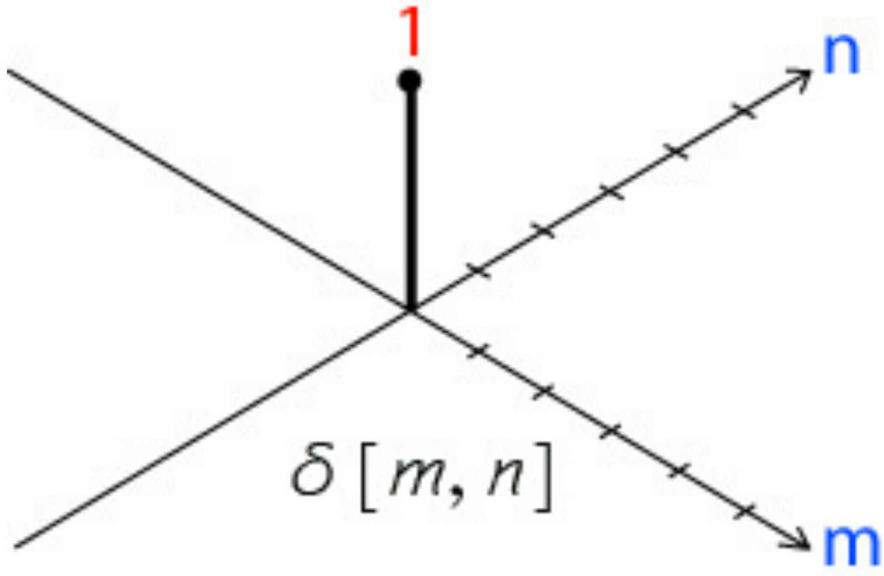
- Is the moving average a linear system?

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

No!

2D impulse function

- 1 at [0,0].
- 0 everywhere else



LSI (linear *shift invariant*) systems

Impulse response

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

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

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

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

Filter example #1: Moving Average

- 2D DS moving average over a 3 × 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

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

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

LSI (linear *shift invariant*) systems

A simple LSI is one that shifts the pixels of an image:

shifting property of the delta function

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

LSI (linear *shift invariant*) systems

A simple LSI is one that shifts the pixels of an image:

shifting property of the delta function

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

Remember the superposition property:

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

superposition property

LSI (linear *shift invariant*) systems

With the superposition property, any LSI system can be represented as a weighted sum of such shifting systems:

$$\alpha_1 \sum_k \sum_l f[k,l]\delta_{2,1}[k-n,l-m]$$
$$+\alpha_2 \sum_k \sum_l f[k,l]\delta_{2,2}[k-n,l-m]$$
$$+\alpha_3 \sum_k \sum_l f[k,l]\delta_{2,3}[k-n,l-m]$$
$$+\ldots$$

LSI (linear *shift invariant*) systems

Rewriting the above summation:

$$\sum_k \sum_l f[k,l](\alpha_1 \delta_{2,1}[k-n, l-m]$$

$$+ \alpha_2 \delta_{2,2}[k-n, l-m]$$

$$+ \alpha_3 \delta_{2,3}[k-n, l-m]$$

$$+ \dots)$$

LSI (linear *shift invariant*) systems

We define the filter of a LSI as:

$$
\begin{aligned}
h[k,l] = & \alpha_1 \delta_{2,1}[k, l-m] \\
& + \alpha_2 \delta_{2,2}[k-n, l-m] \\
& + \alpha_3 \delta_{2,3}[k-n, l-m] \\
& + \dots
\end{aligned}
$$

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

What we will learn today?

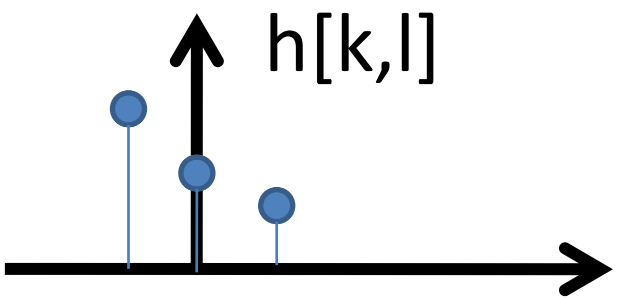
- Images as functions
- Linear systems (filters)
- Convolution and correlation

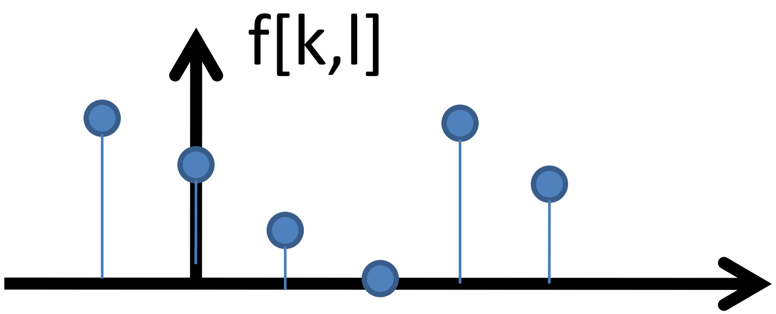
Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

1D Discrete convolution (symbol: $*$)

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

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



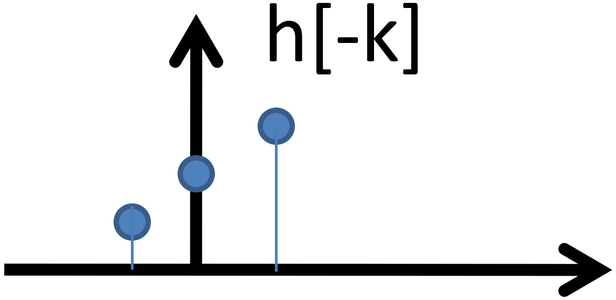


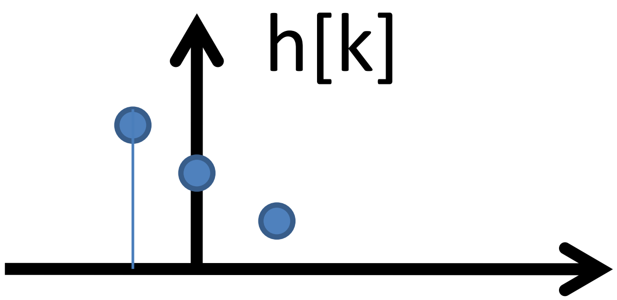
1D Discrete convolution (symbol: $\ast$)

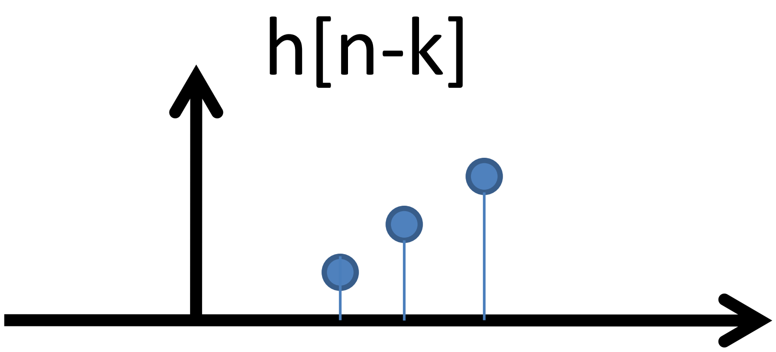
We are going to convolve a function **f** with a filter **h.**

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

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

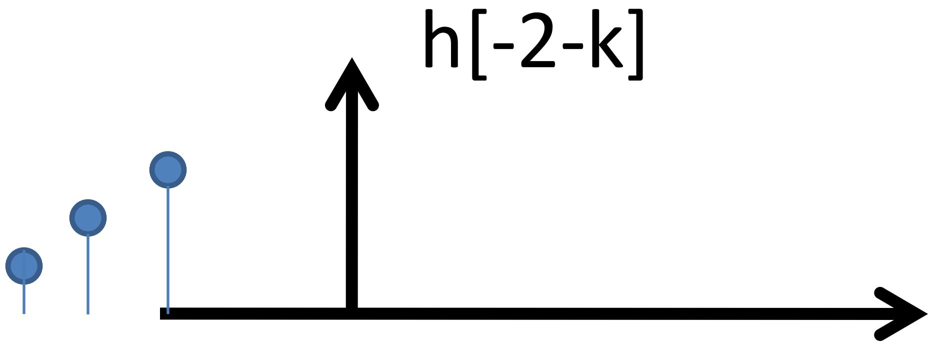


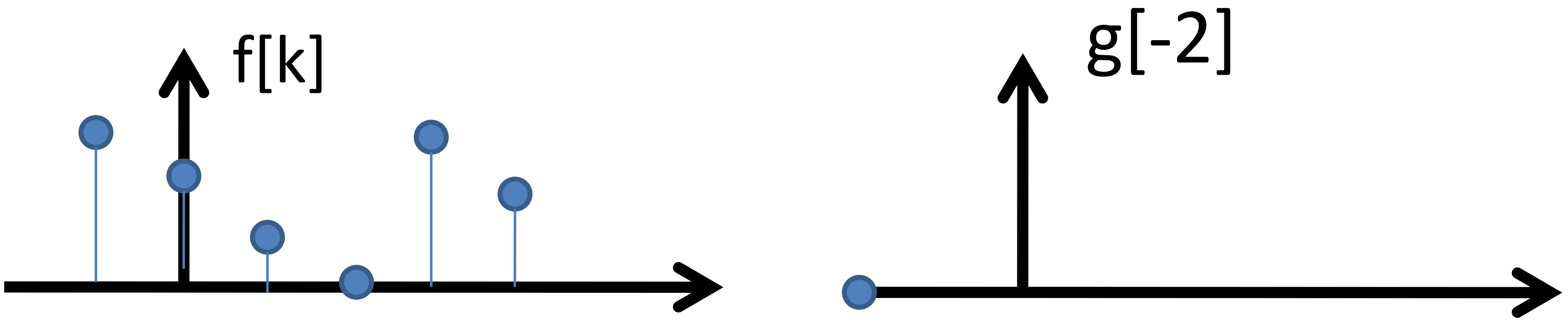




Discrete convolution (symbol: $*$)

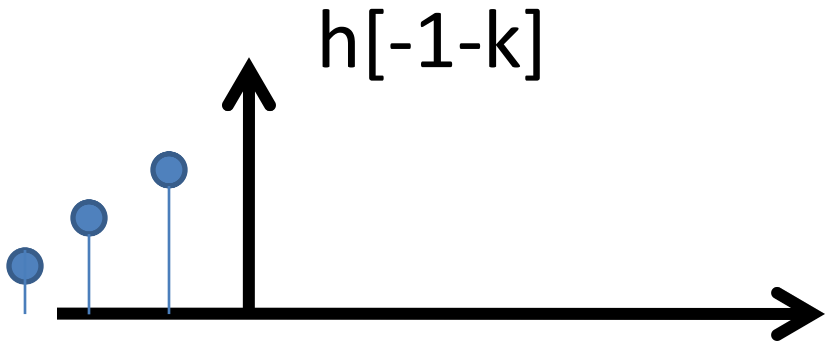
We are going to convolve a function **f** with a filter **h.**

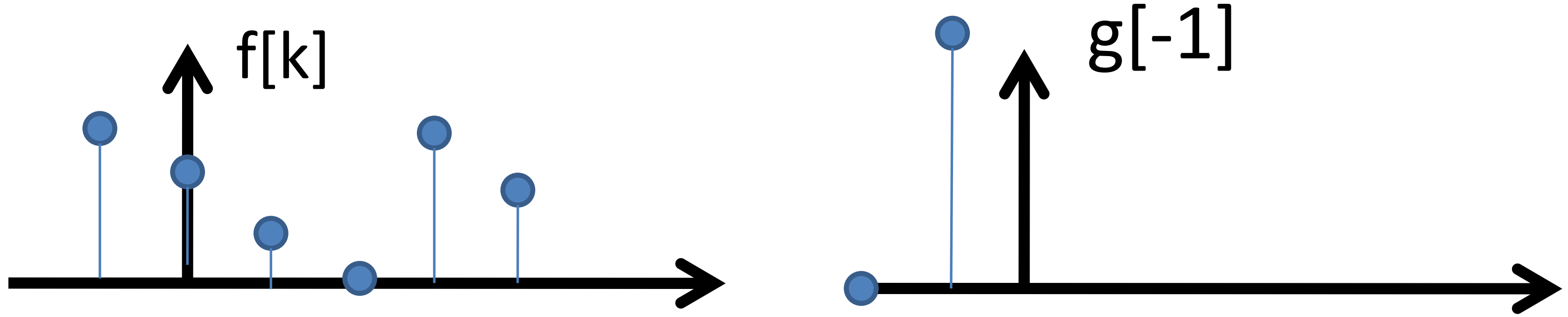




Discrete convolution (symbol: $*$)

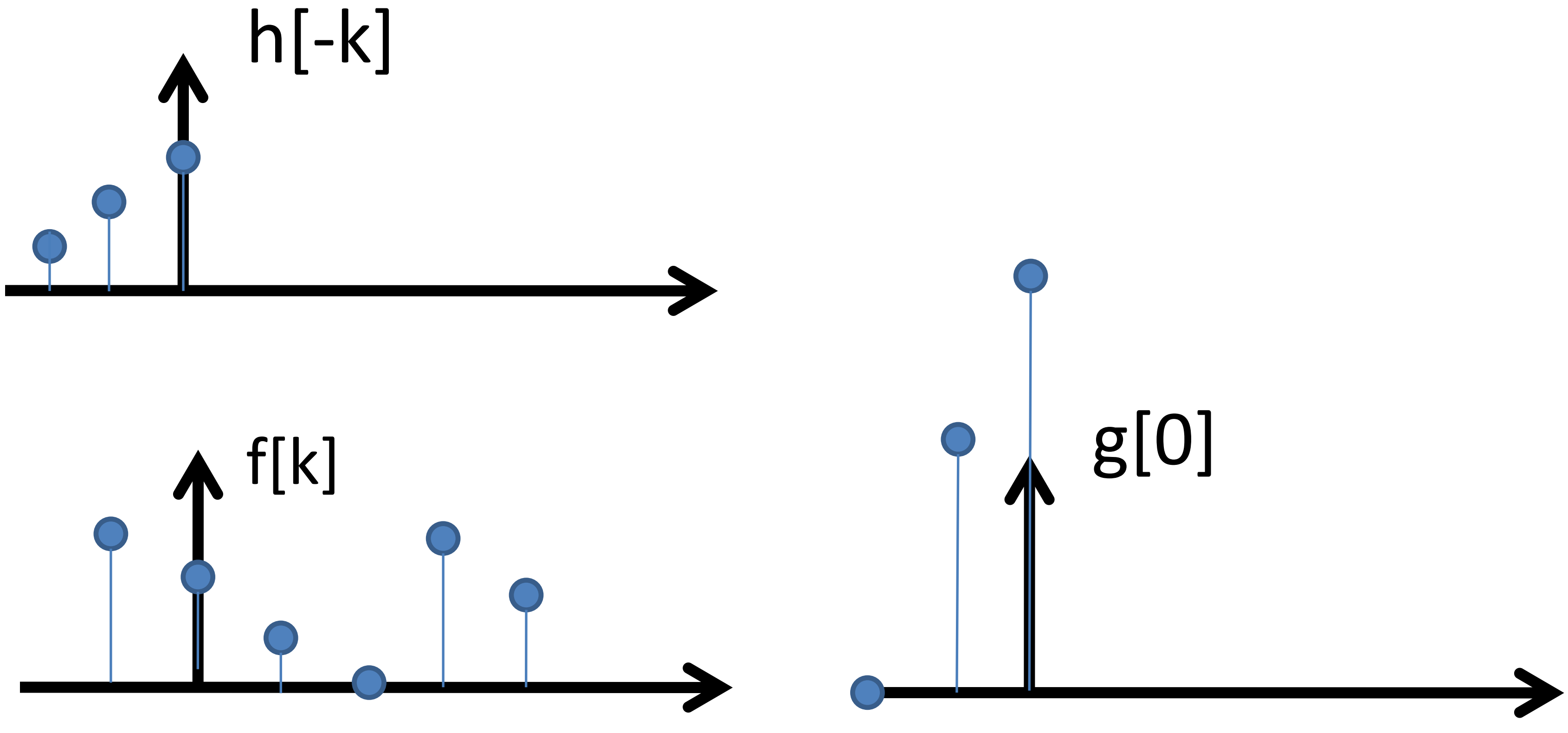
We are going to convolve a function **f** with a filter **h.**





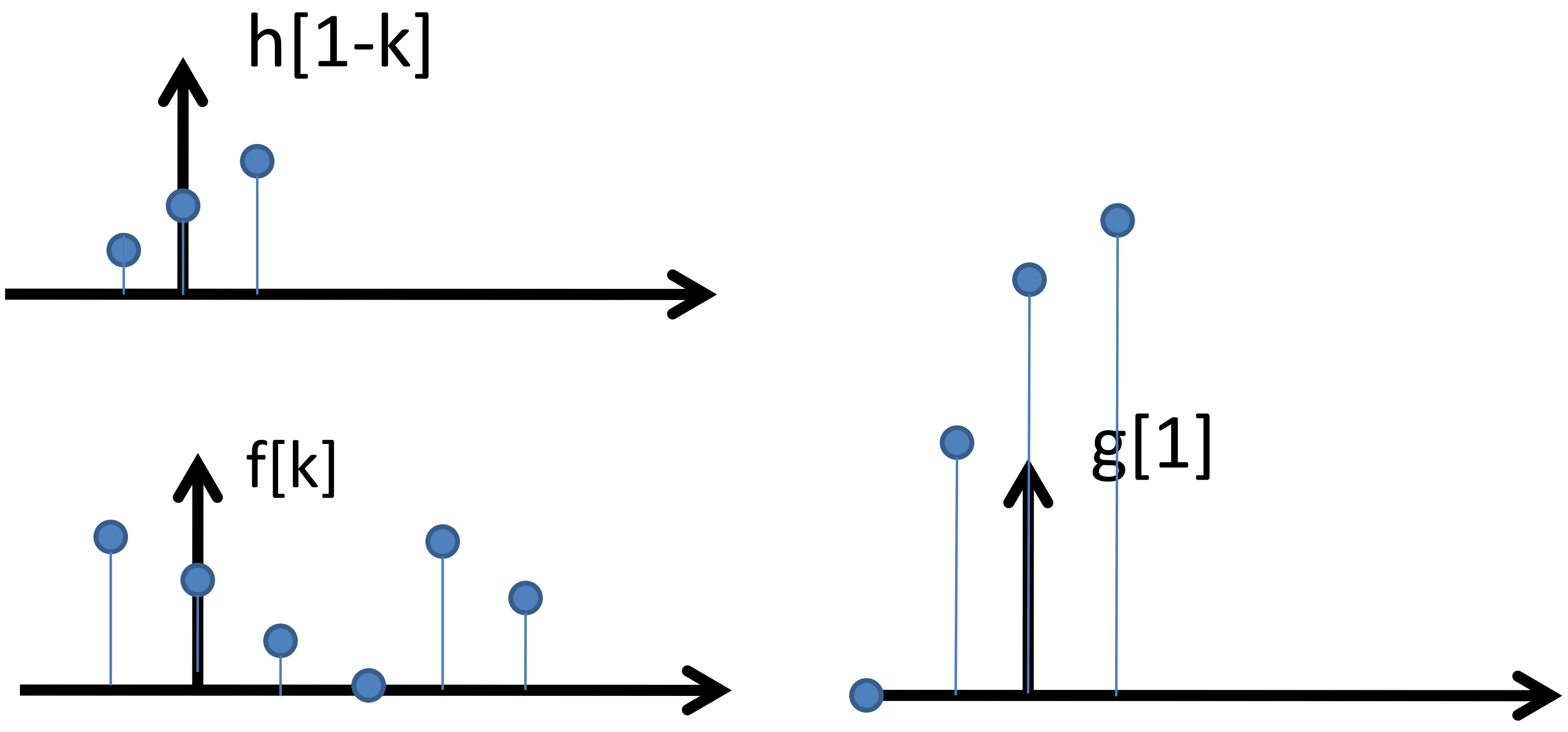
Discrete convolution (symbol: $*$)

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



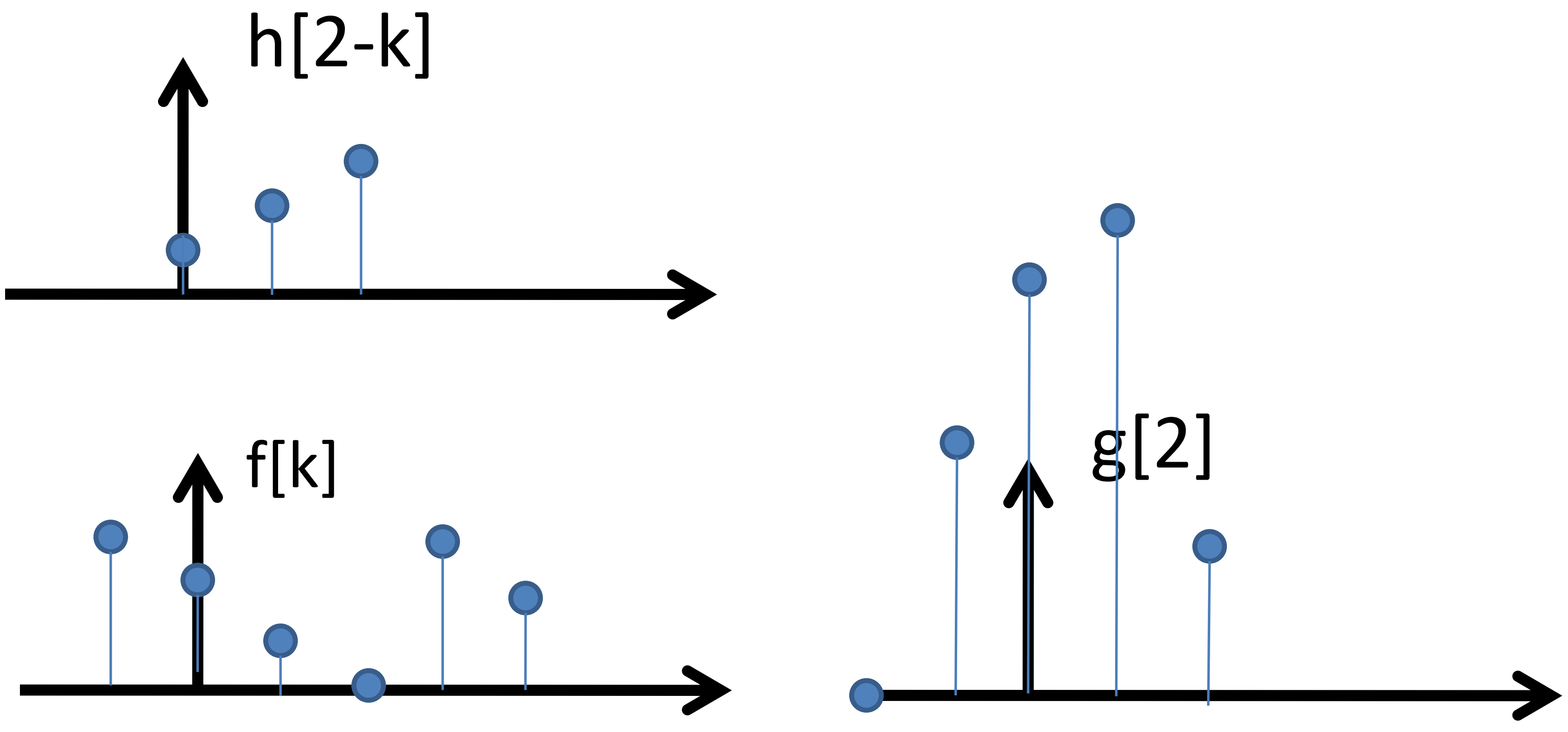
Discrete convolution (symbol: $*$)

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



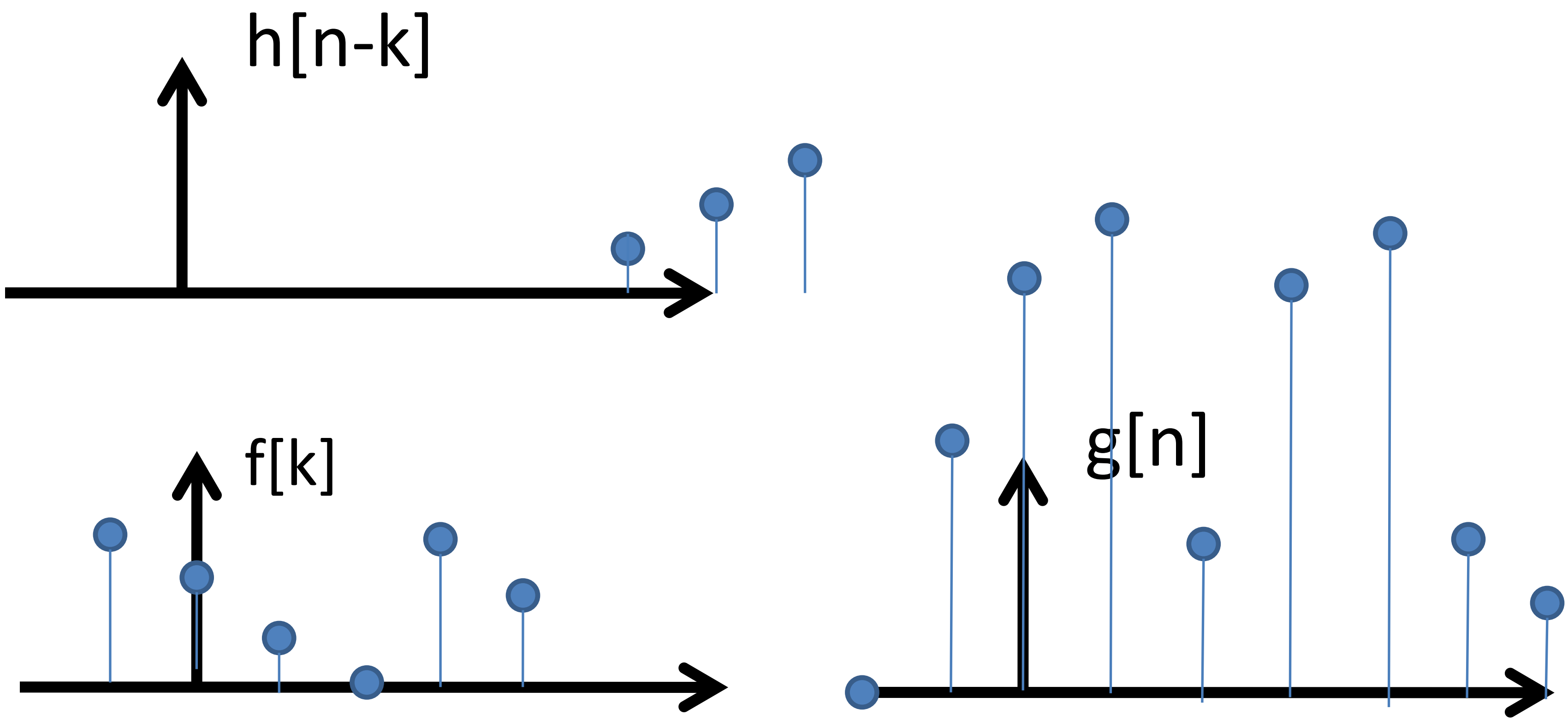
Discrete convolution (symbol: $*$)

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.**



Discrete convolution (symbol: $*$)

In summary, the steps for discrete convolution are:

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

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

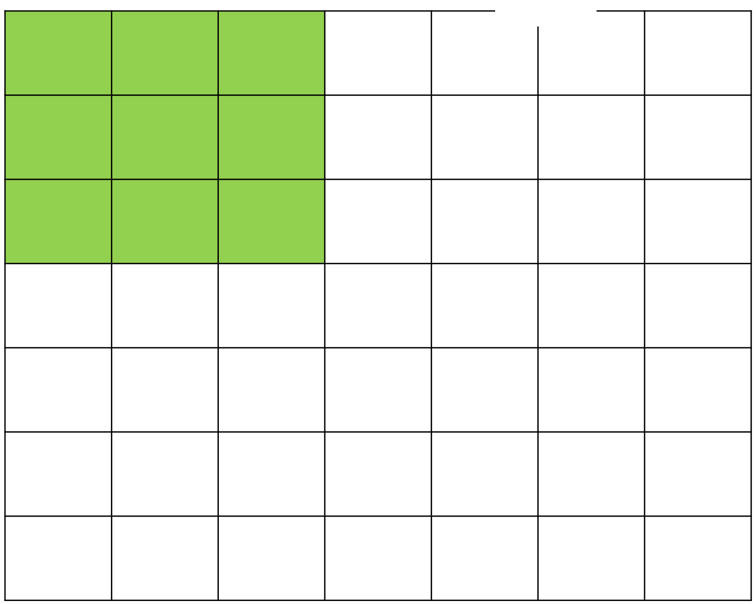
n

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.

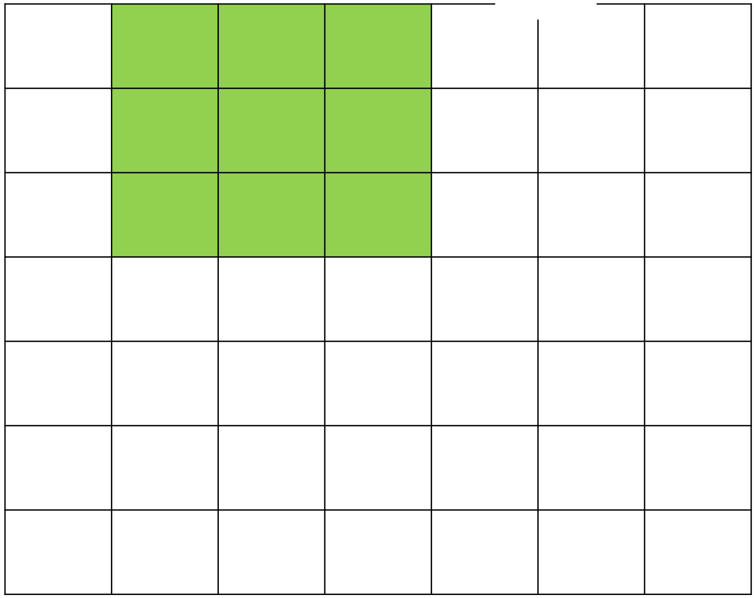
n

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.

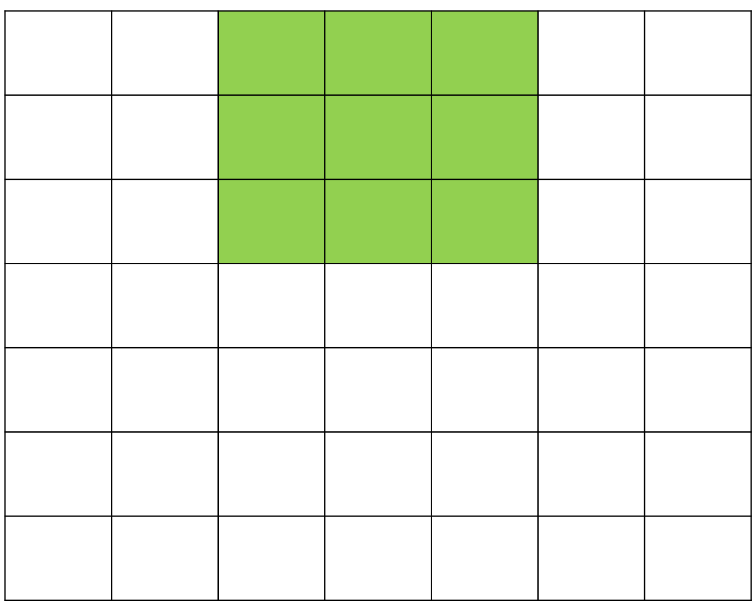
n

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.

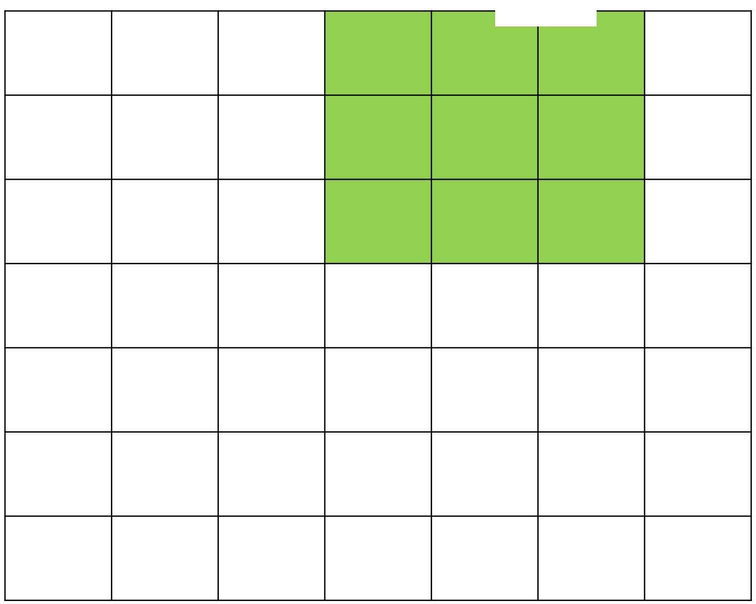
n

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.

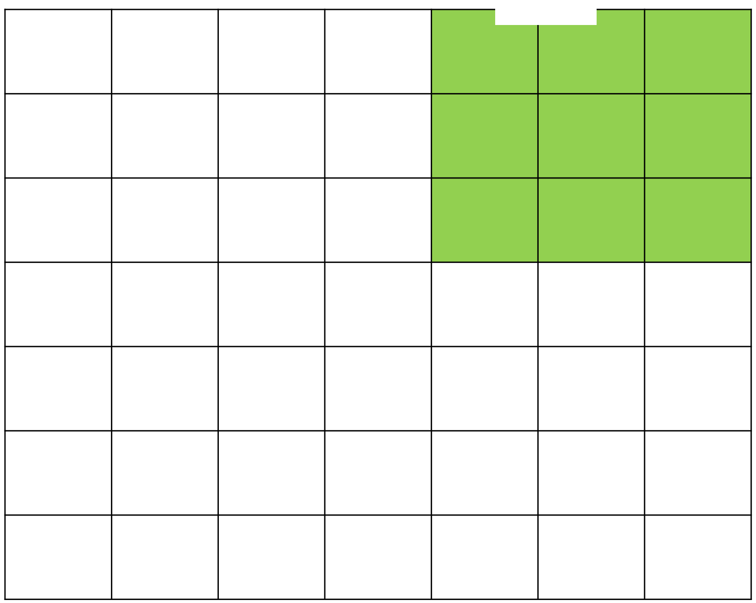
n

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.

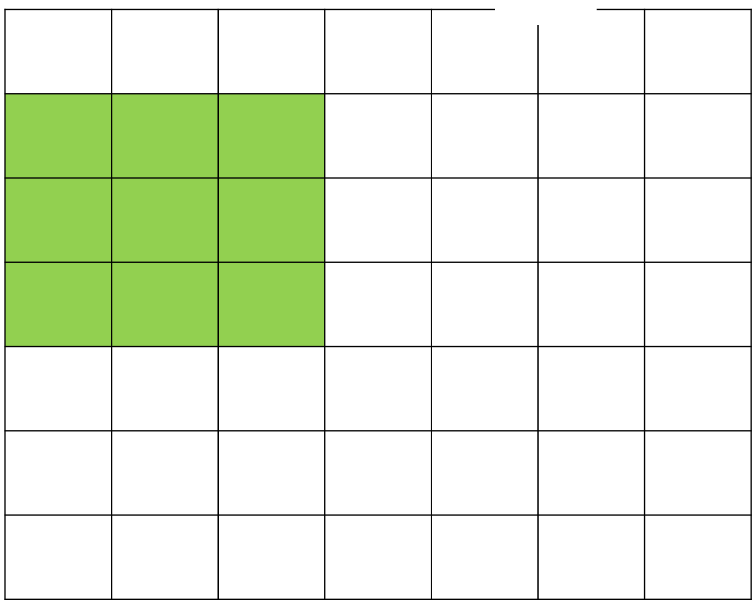
n

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.

LSI (linear *shift invariant*) systems

An LSI system is completely specified by its impulse response.

shifting 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

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

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

Discrete convolution

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

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

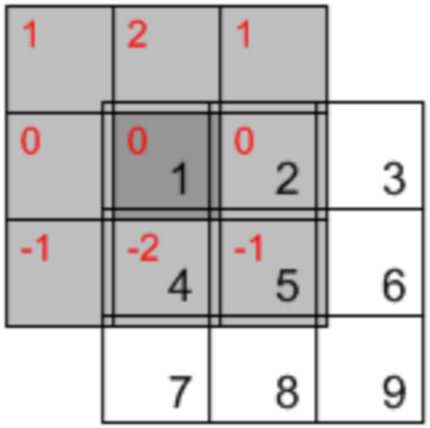
Kernel

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

Output

Slide credit: Song Ho Ahn

2D convolution example



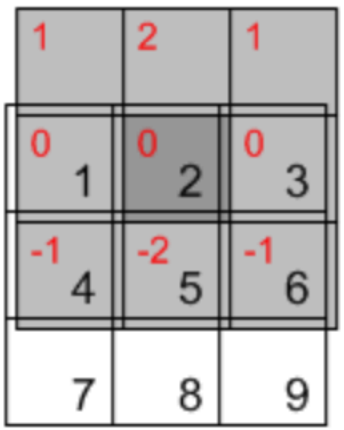
$$
\begin{aligned}
&= x[-1,-1]\cdot h[1,1] + x[0,-1]\cdot h[0,1] + x[1,-1]\cdot h[-1,1] \\
&\quad + x[-1,0]\cdot h[1,0] + x[0,0]\cdot h[0,0] + x[1,0]\cdot h[-1,0] \\
&\quad + x[-1,1]\cdot h[1,-1] + x[0,1]\cdot h[0,-1] + x[1,1]\cdot h[-1,-1] \\
&= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 0\cdot 0 + 1\cdot 0 +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

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

Output

Slide credit: Song Ho Ahn

2D convolution example



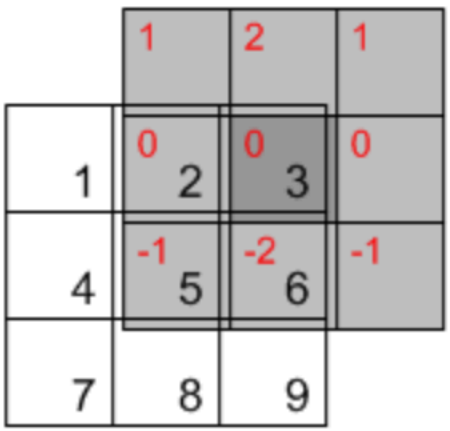
$$
\begin{aligned}
&= x[0,-1]\cdot h[1,1] + x[1,-1]\cdot h[0,1] + x[2,-1]\cdot h[-1,1] \\
&\quad + x[0,0]\cdot h[1,0] + x[1,0]\cdot h[0,0] + x[2,0]\cdot h[-1,0] \\
&\quad + x[0,1]\cdot h[1,-1] + x[1,1]\cdot h[0,-1] + x[2,1]\cdot h[-1,-1] \\
&= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 1\cdot 0 + 2\cdot 0 + 3\cdot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

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

Output

Slide credit: Song Ho Ahn

2D convolution example

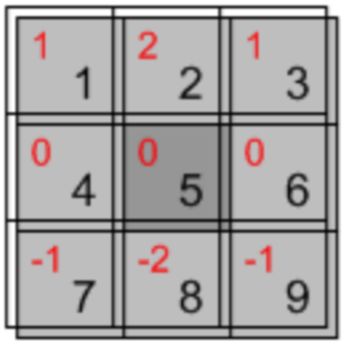
$$
\begin{aligned}
&= x[-1,0]\cdot h[1,1] + x[0,0]\cdot h[0,1] + x[1,0]\cdot h[-1,1] \\
&\quad + x[-1,1]\cdot h[1,0] + x[0,1]\cdot h[0,0] + x[1,1]\cdot h[-1,0] \\
&\quad + x[-1,2]\cdot h[1,-1] + x[0,2]\cdot h[0,-1] + x[1,2]\cdot h[-1,-1] \\
&= 0\cdot 1 + 1\cdot 2 + 2\cdot 1 + 0\cdot 0 + 4\cdot 0 + 5\cdot 0 + 0\cdot(-1) + 7\cdot(-2) + 8\cdot(-1) = -18
\end{aligned}
$$

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

Output

Slide credit: Song Ho Ahn

2D convolution example



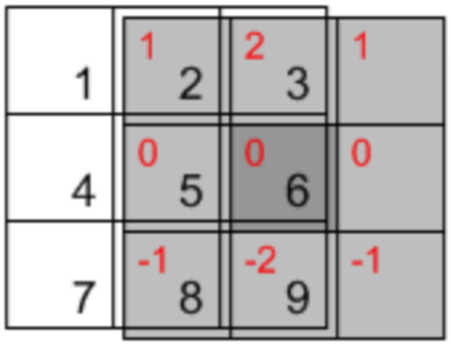
$$
\begin{aligned}
&= x[0,0]\cdot h[1,1] + x[1,0]\cdot h[0,1] + x[2,0]\cdot h[-1,1] \\
&\quad + x[0,1]\cdot h[1,0] + x[1,1]\cdot h[0,0] + x[2,1]\cdot h[-1,0] \\
&\quad + x[0,2]\cdot h[1,-1] + x[1,2]\cdot h[0,-1] + x[2,2]\cdot h[-1,-1] \\
&= 1\cdot 1 + 2\cdot 2 + 3\cdot 1 + 4\cdot 0 + 5\cdot 0 + 6\cdot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

-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

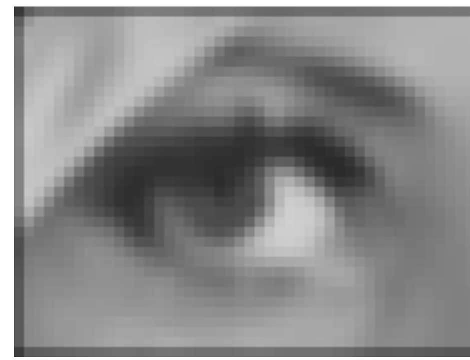
$$* \ \frac{1}{9} \begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix} = \ ?$$

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

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

(Note that filter sums to 1)

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

Courtesy of D Lowe

- What does blurring take away?

–

=

  - Let’s add it back:

+ a

=

Convolution in 2D – Sharpening filter

Original

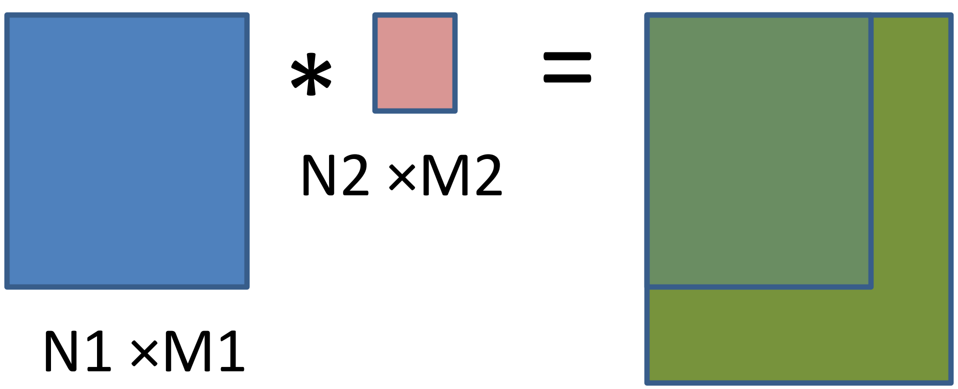
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Sharpening filter: Accentuates differences with local average

Courtesy of D Lowe

Image support and edge effect

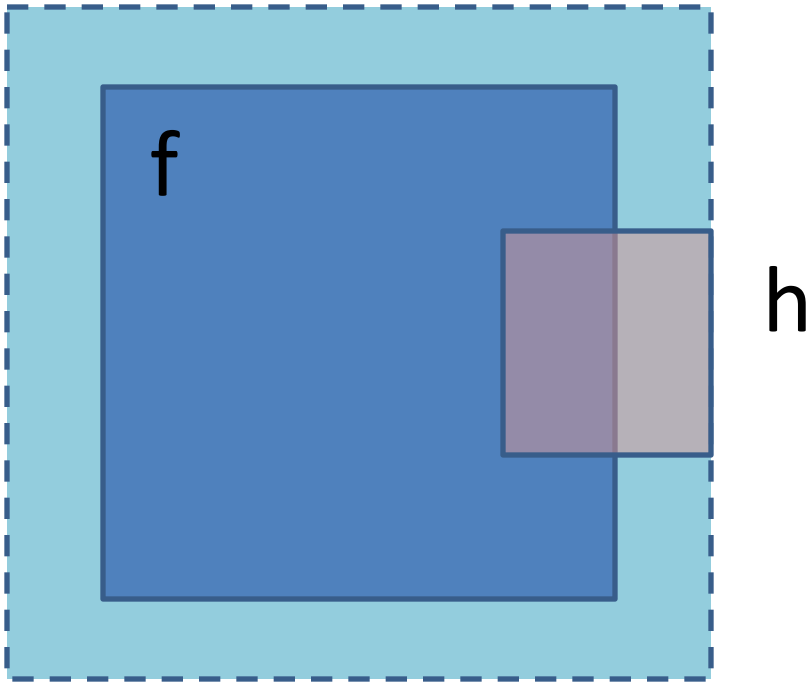
- A computer will only convolve **finite support signals.**
  - That is: images that are zero for n,m outside some rectangular region
- numpy's convolution performs 2D DS convolution of finite-support signals.



$(N1 + N2 - 1) \times (M1 + M2 - 1)$

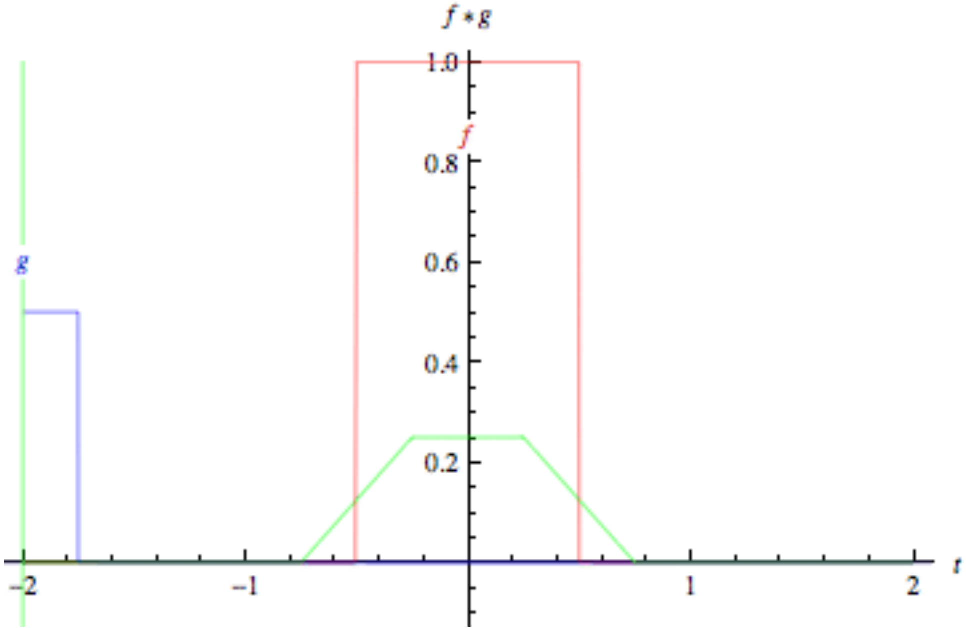
Image support and edge effect

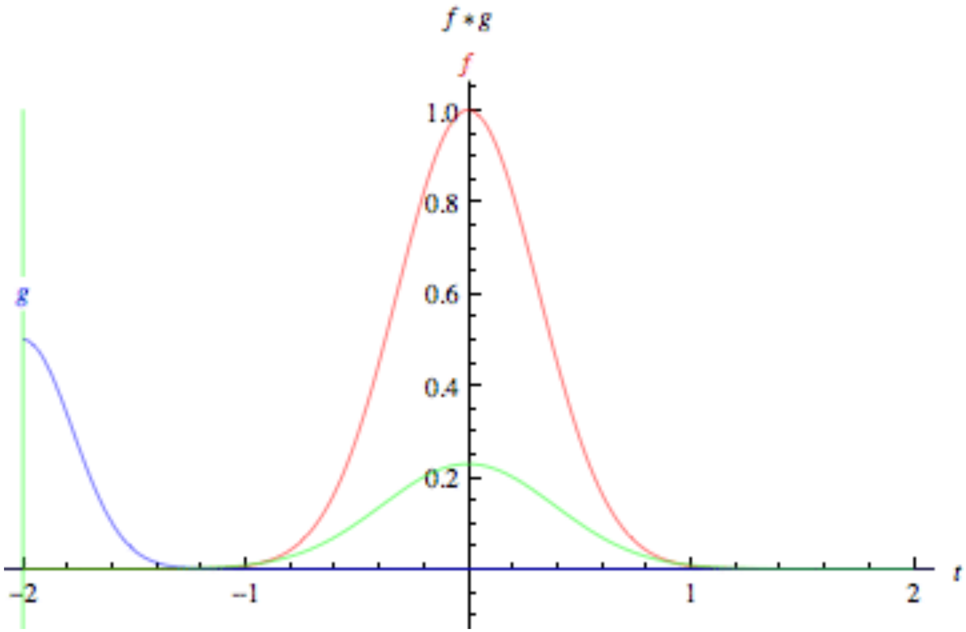
- A computer will only convolve **finite support signals.**
- What happens at the edge?



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

-> Matlab conv2 uses zero-padding





Slide credit: Wolfram Alpha

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

(Cross) correlation (symbol: $**$)

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

$$r_{fg}[k,l] \triangleq \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n,m]\, g^*[n-k, m-l]$$

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

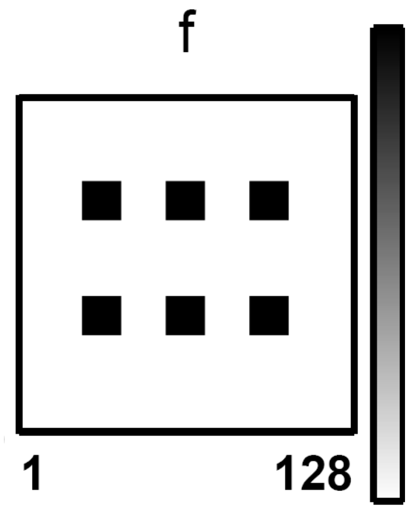
(k, l) is called the **lag**

- Equivalent to a convolution without the flip

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

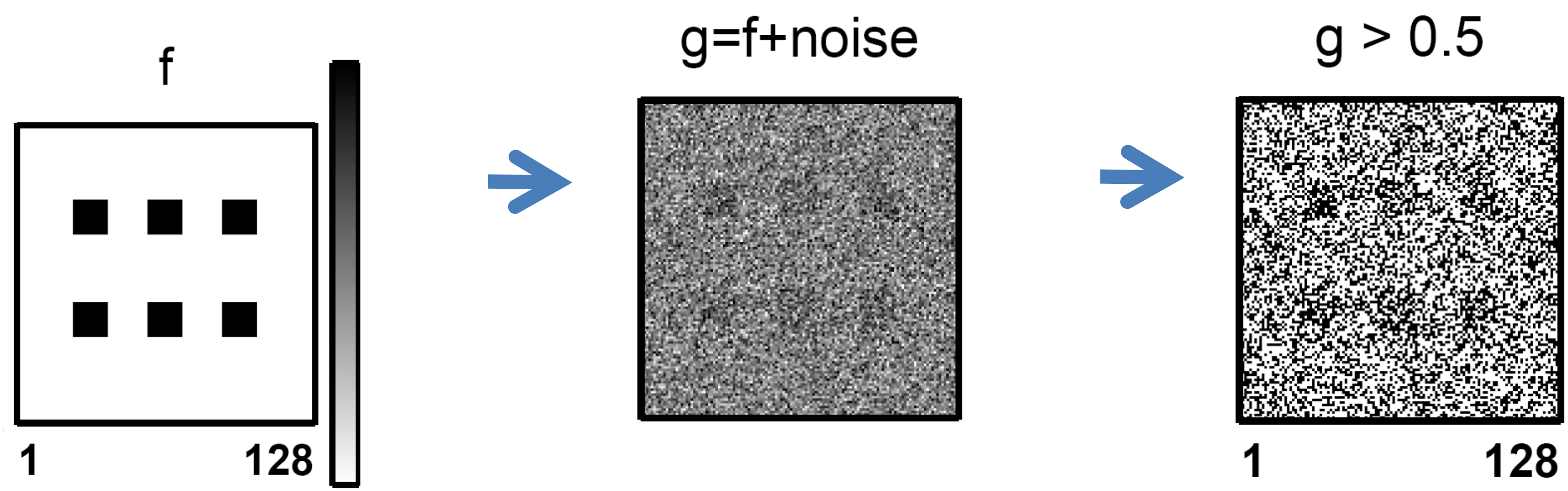
(g* is defined as the *complex conjugate* of g. In this class, g(n,m) are real numbers, hence g*=g.)

(Cross) correlation – example



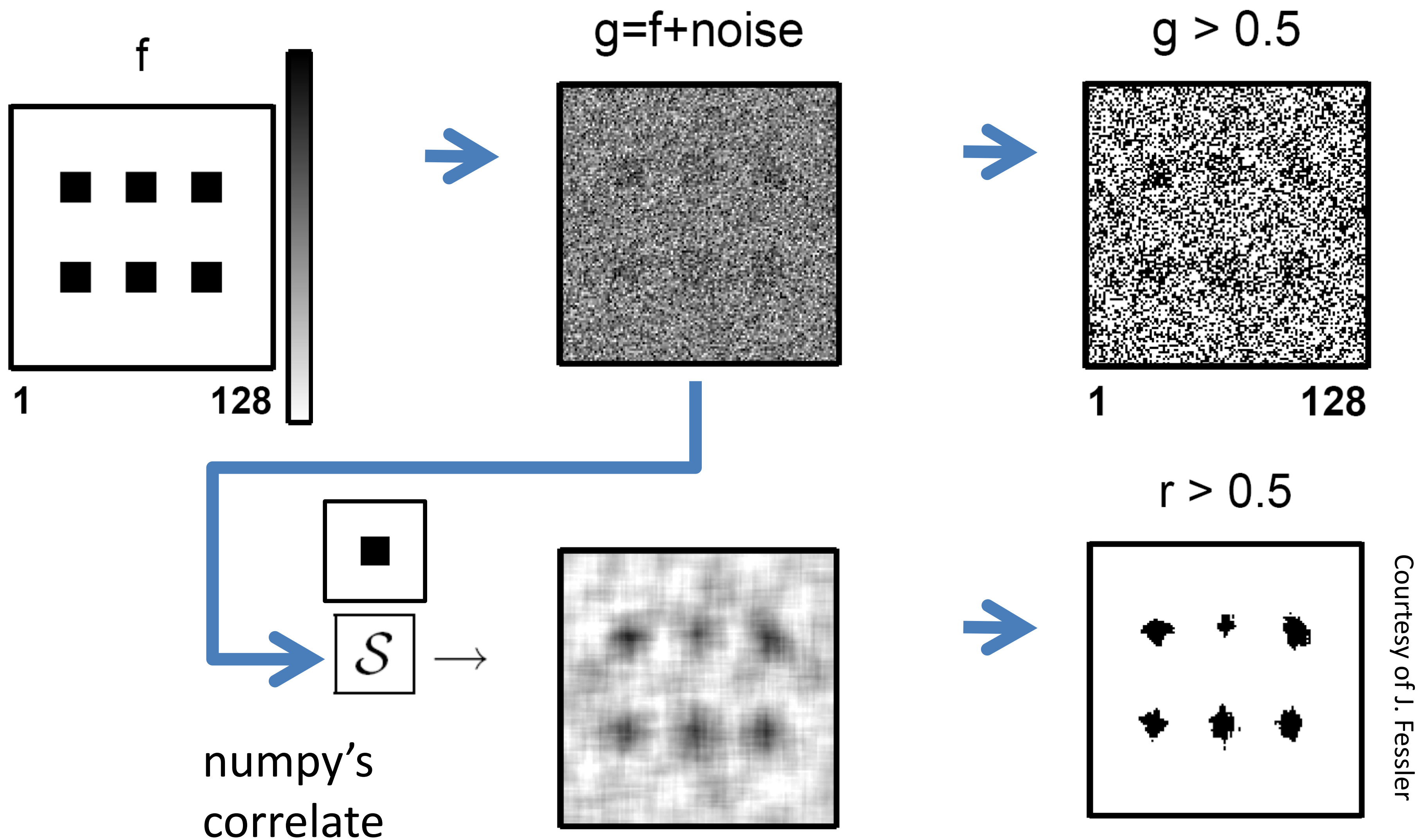
Courtesy of J. Fessler

(Cross) correlation – example

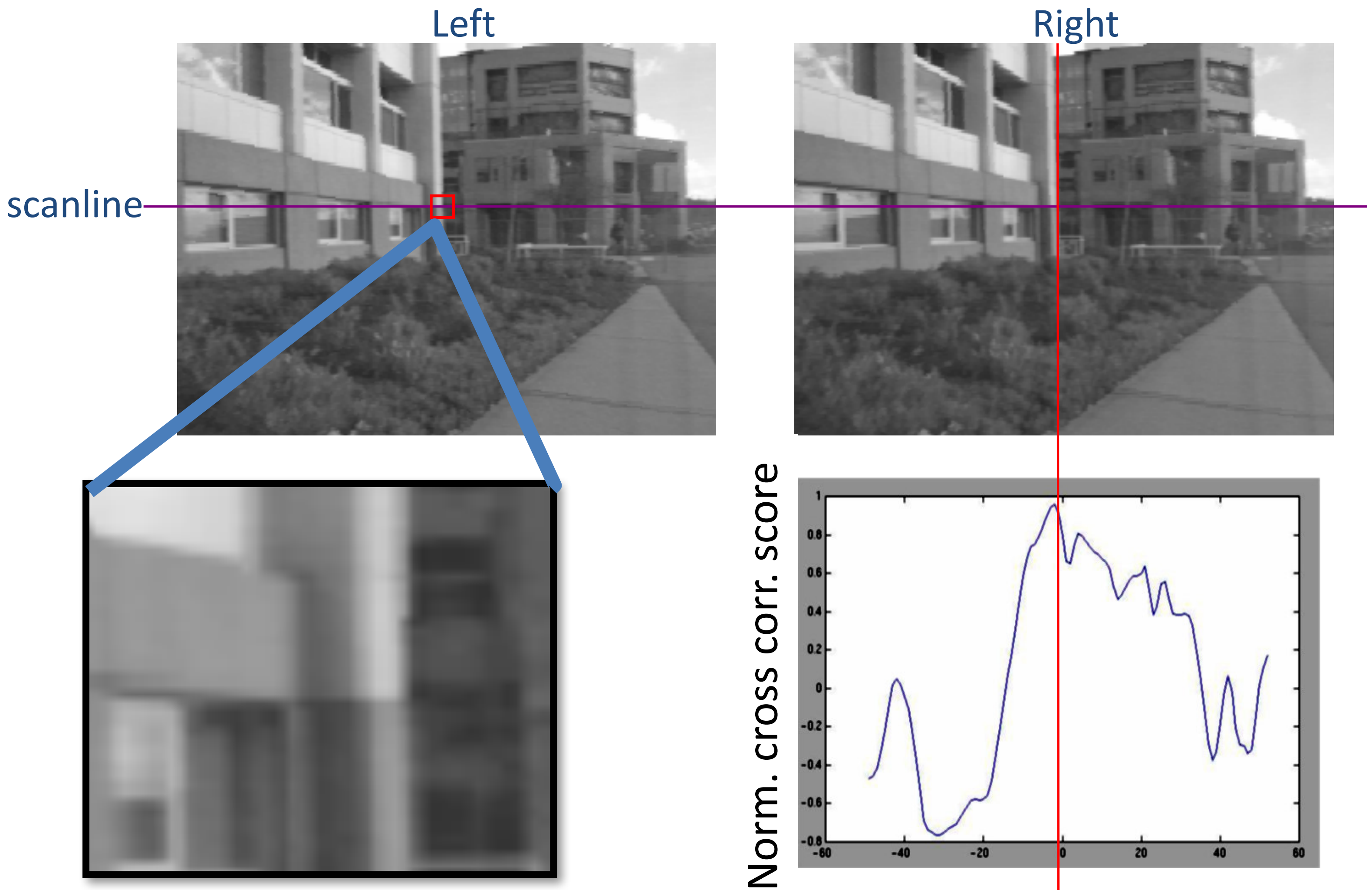


Courtesy of J. Fessler

(Cross) correlation – example

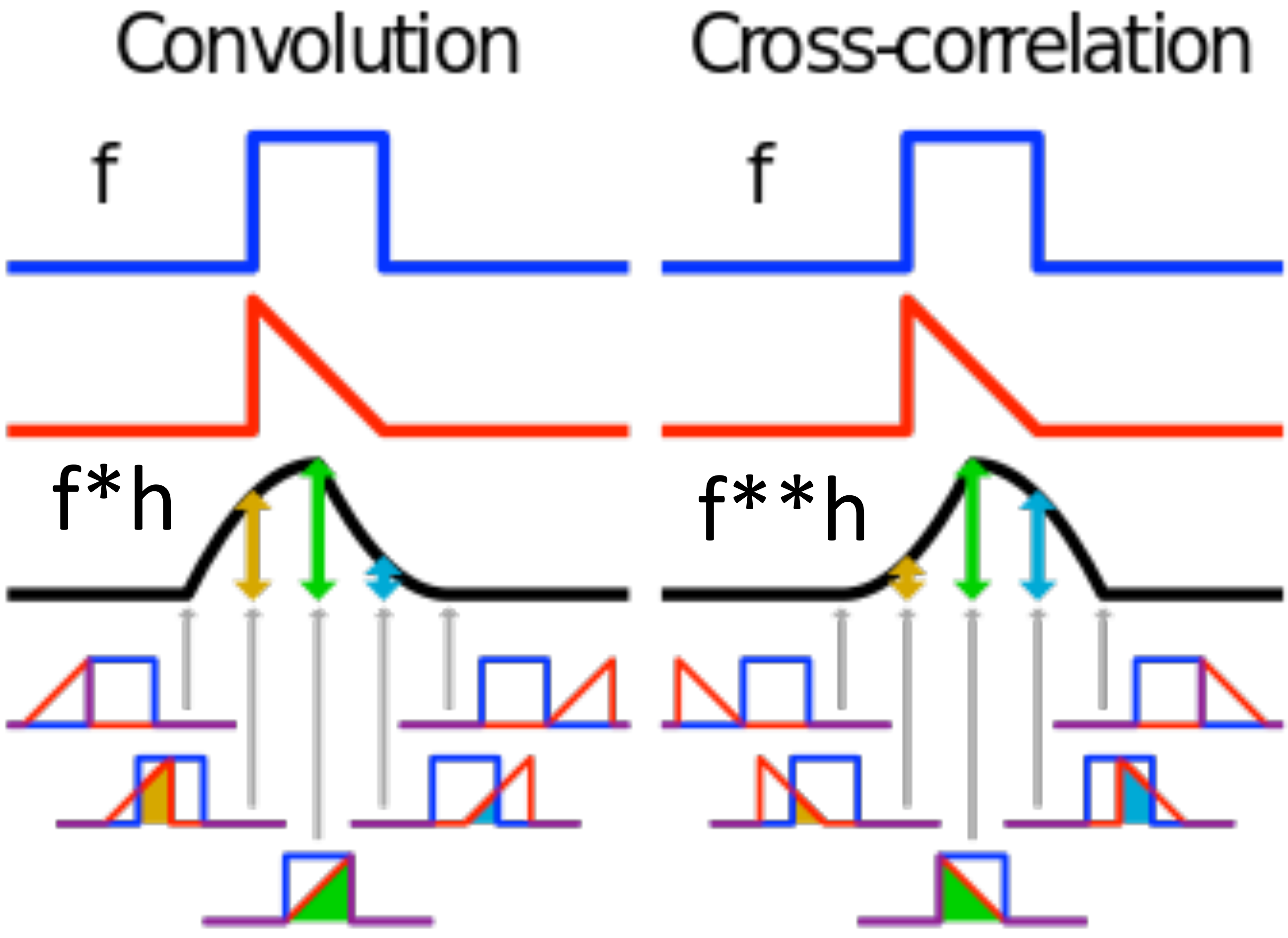


(Cross) correlation – example



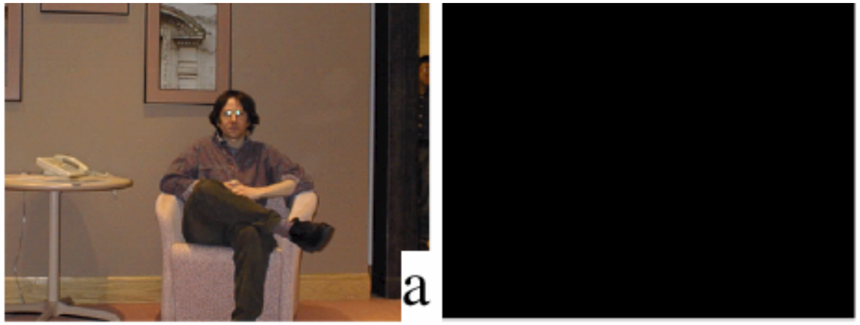
$$dc(y_1, y_2) = \frac{y_1^T y_2}{|y_1||y_2|}$$

Convolution vs. (Cross) Correlation



Cross Correlation Application: Vision system for TV remote control

- uses template matching



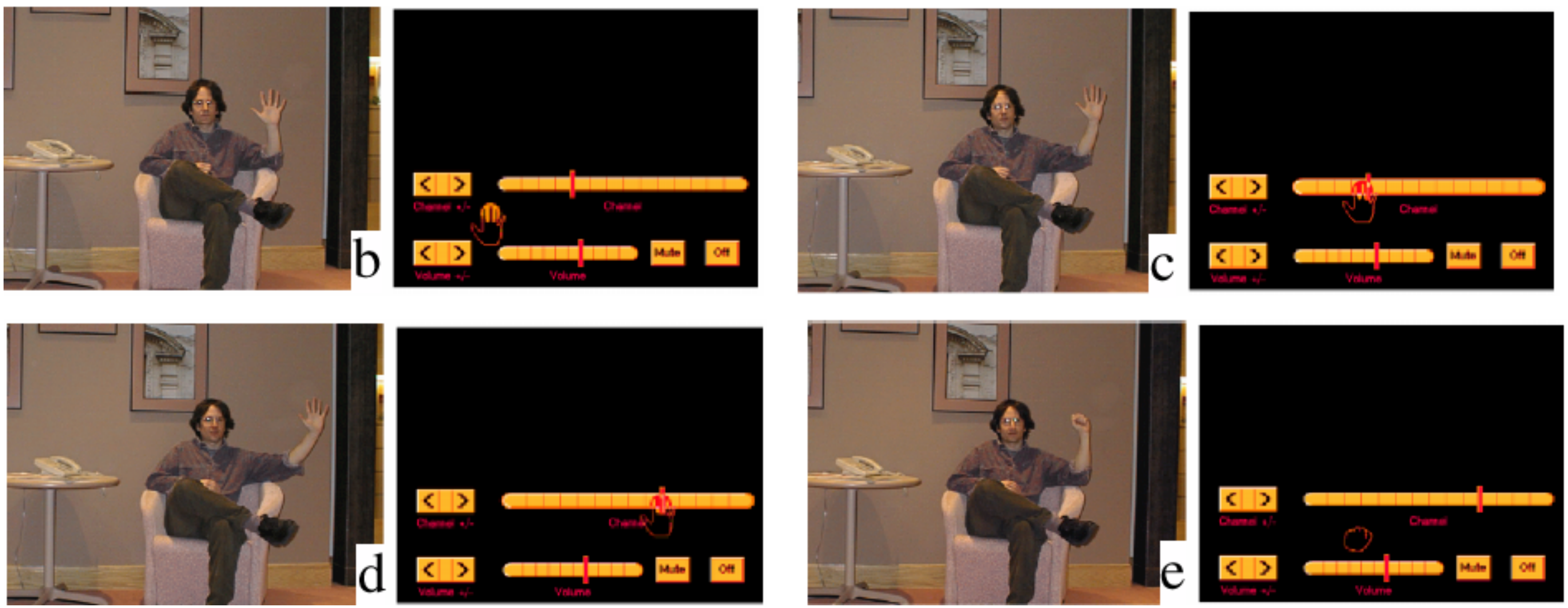


Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

properties

- **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! $h_1 ** h_2 = h_2 ** h_1$

properties

- **Shift property:**

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

- **Shift-invariance:**

$$\begin{aligned} g[n,m] &= f[n,m] ** h[n,m] \\ &\Longrightarrow f[n-l_1, m-l_1] ** h[n-l_2, m-l_2] \\ &= g[n-l_1-l_2, m-l_1-l_2] \end{aligned}$$

Convolution vs. (Cross) Correlation

- A <u>**convolution**</u> is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation

- <u>**Correlation**</u> 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

What we have learned today?

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