Lecture:
Pixels and Filters

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Types of Images

Binary

Types of Images

Binary

Gray Scale

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Types of Images

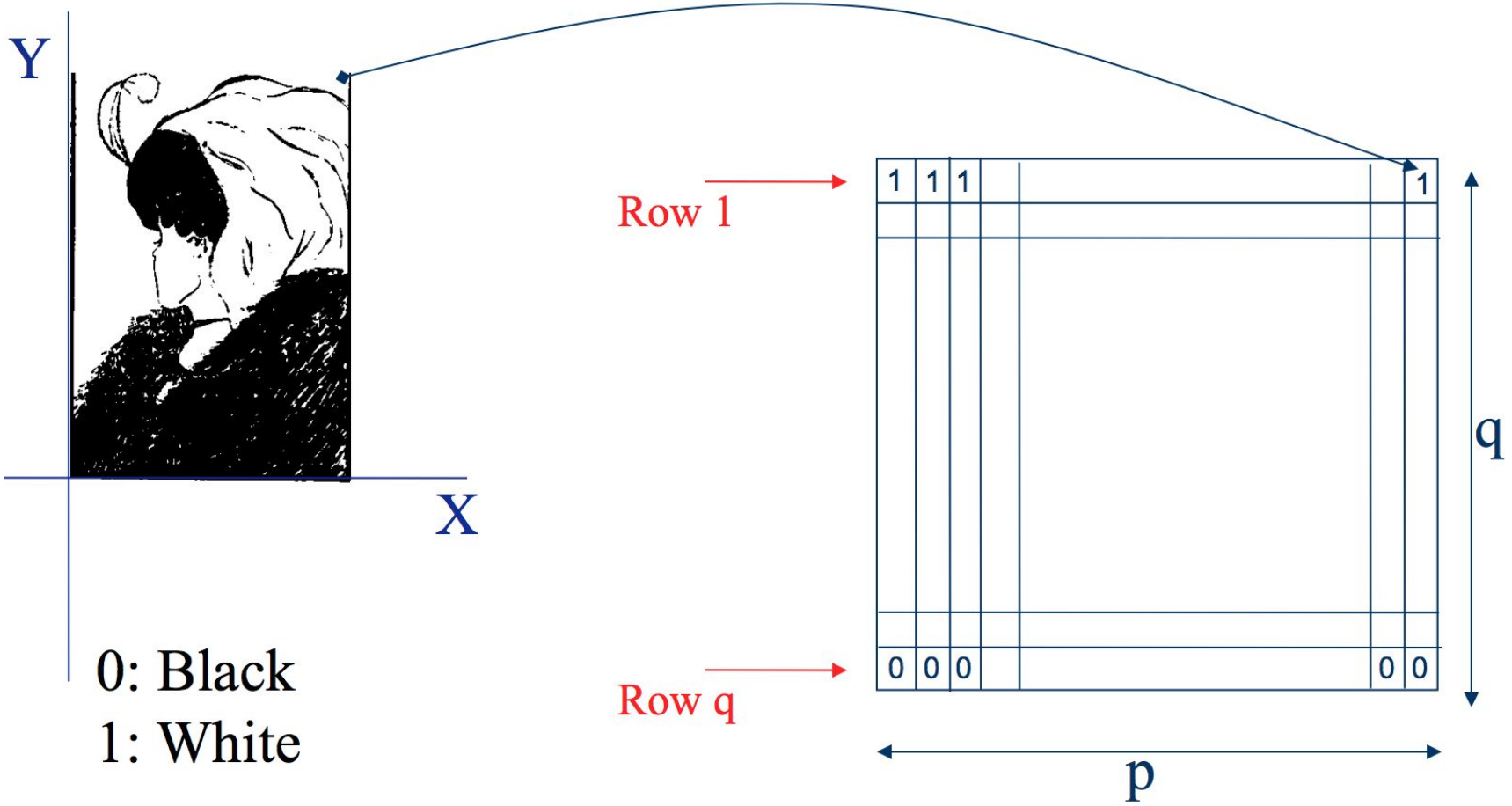
Binary

Gray Scale

Color

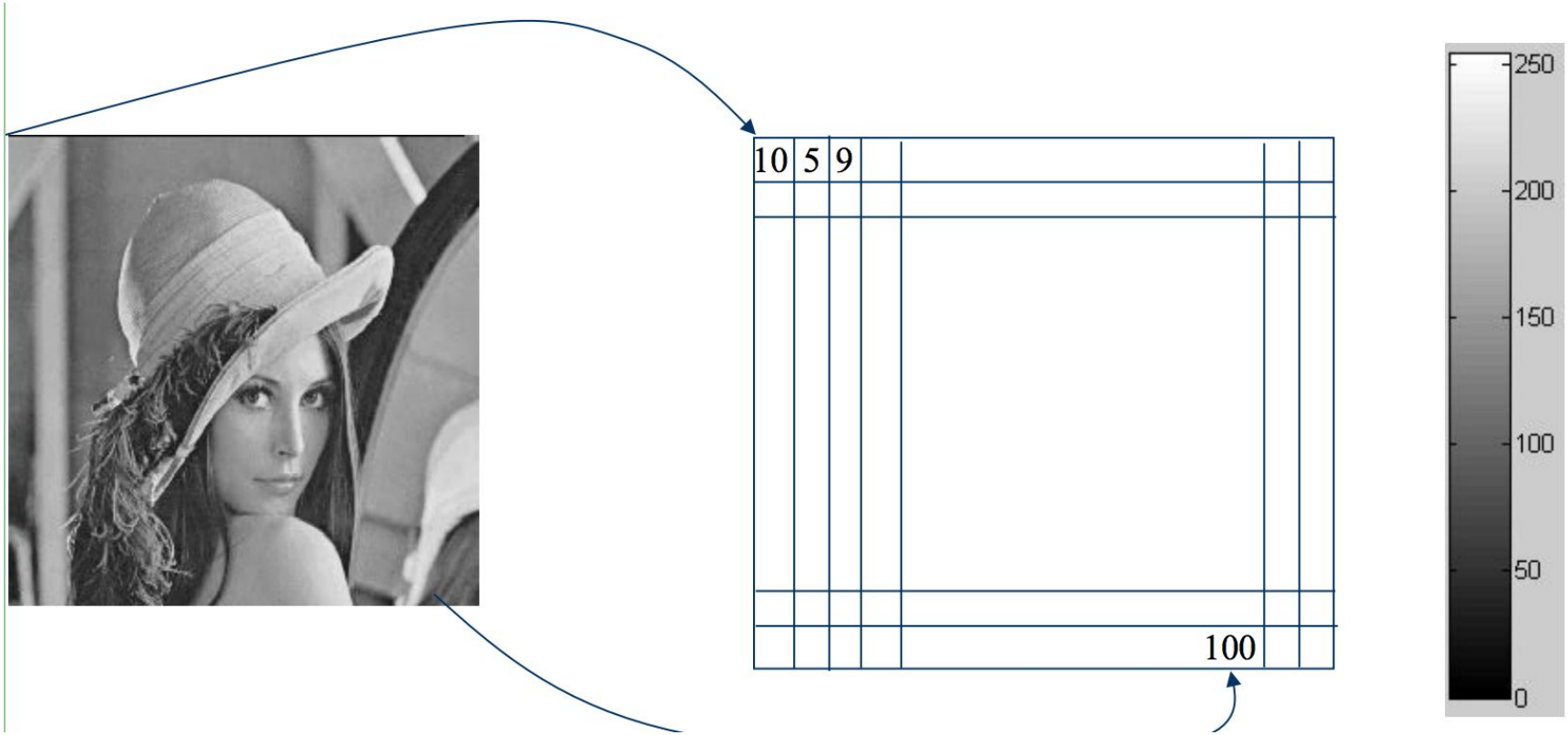
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Binary image representation



Slide credit: Ulas Bagci

Grayscale image representation



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Ulas Bagci

Color Image - one channel

Slide credit: Ulas Bagci

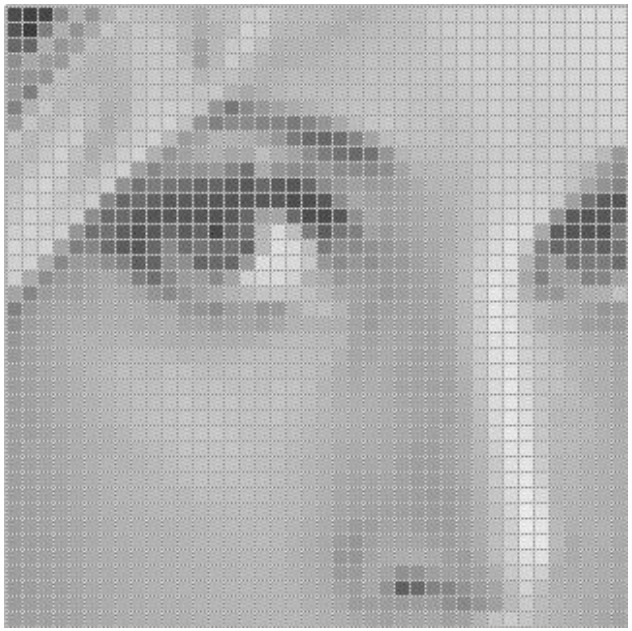
Color image representation

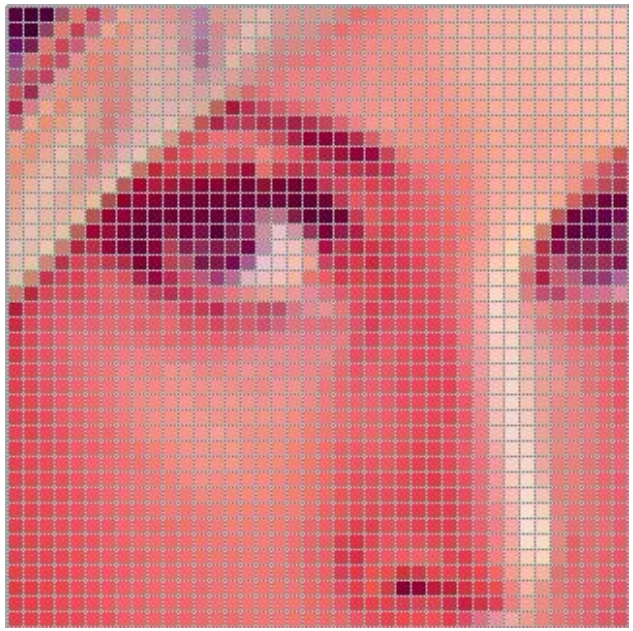
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Ulas Bagci

Images are sampled

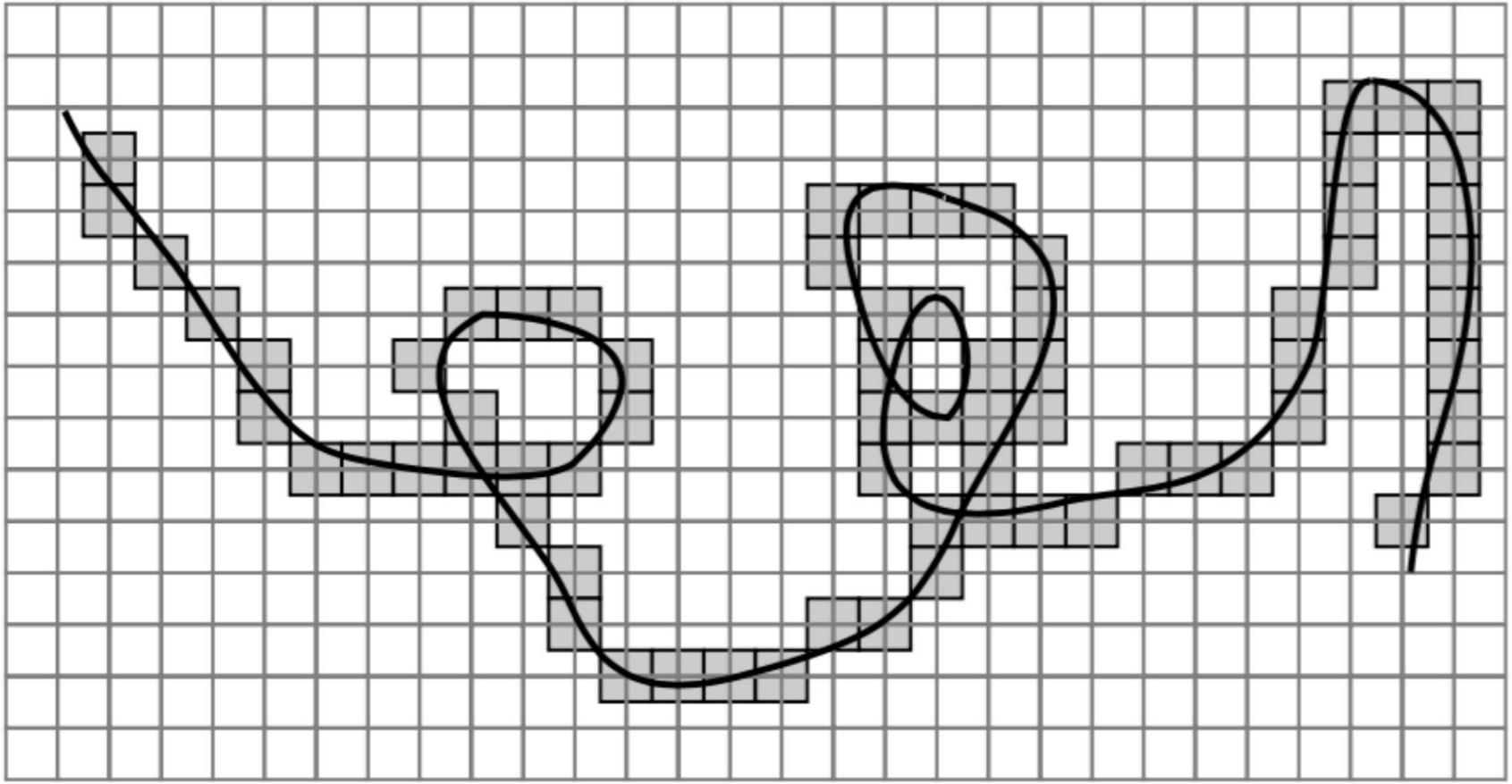
What happens when we zoom into the images we capture?





Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Errors due Sampling



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Ulas Bagci

Resolution

is a **sampling** parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density

- Resolution of the camera and the resolution of the screen can both affect the quality.

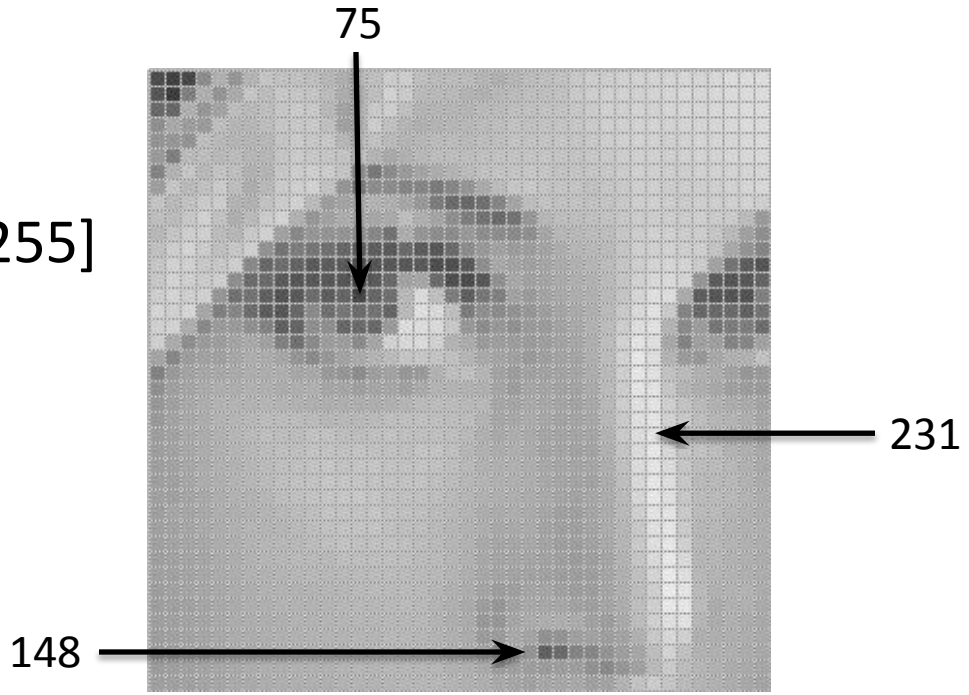
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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]

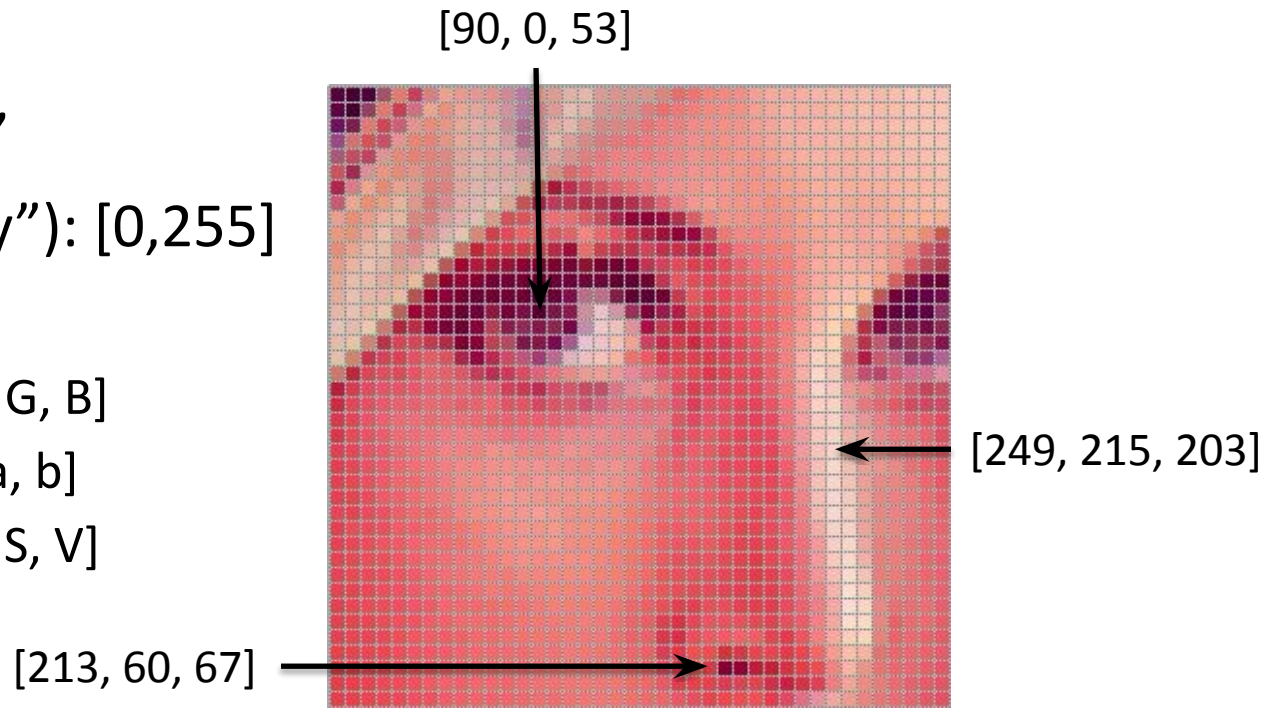


Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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]



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

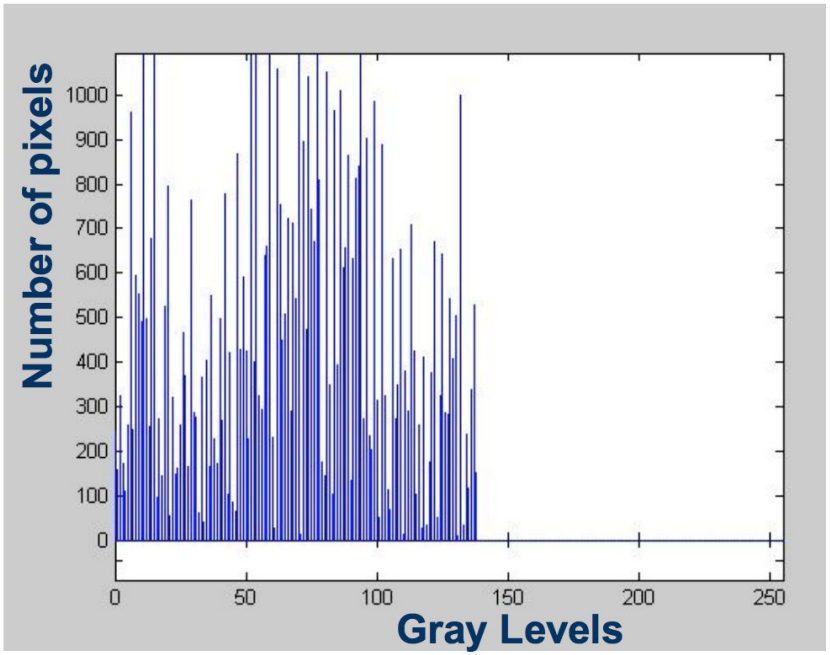
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Histogram

- Histogram captures the distribution of "countable elements" in the image, e.g. how frequently each gray level occurs in the image

Histogram

- Grayscale image - intensity histogram: one of the simplest *image representations*



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Histogram

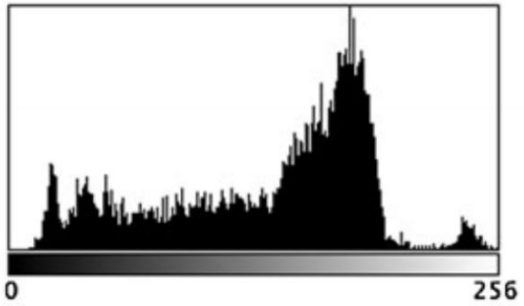
Example implementation of the most basic grayscale histogram:

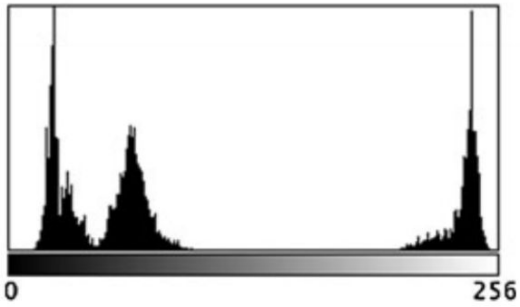
```
def histogram(im):
    h = np.zeros(256)
    for row in range(im.shape[0]):
        for col in range(im.shape[1]):
            val = im[row, col]
            h[val] += 1
```

Note: Python for loops can be slow, a faster implementation: see *numpy.histogram*.

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Histogram

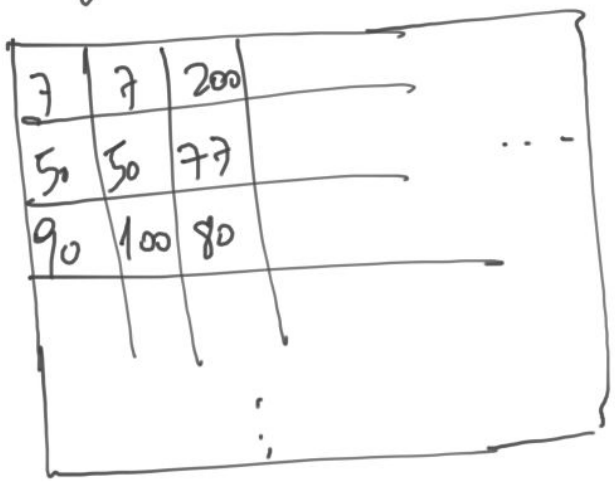




Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Dr. Mubarak Shah

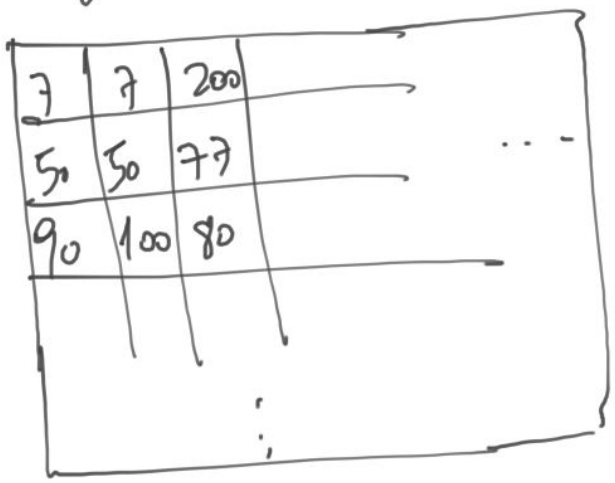
Histogram



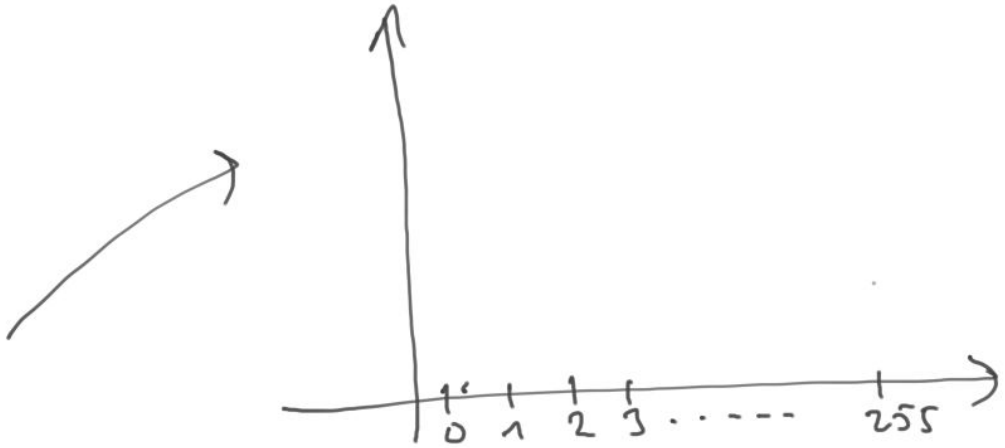
gray scale image

?

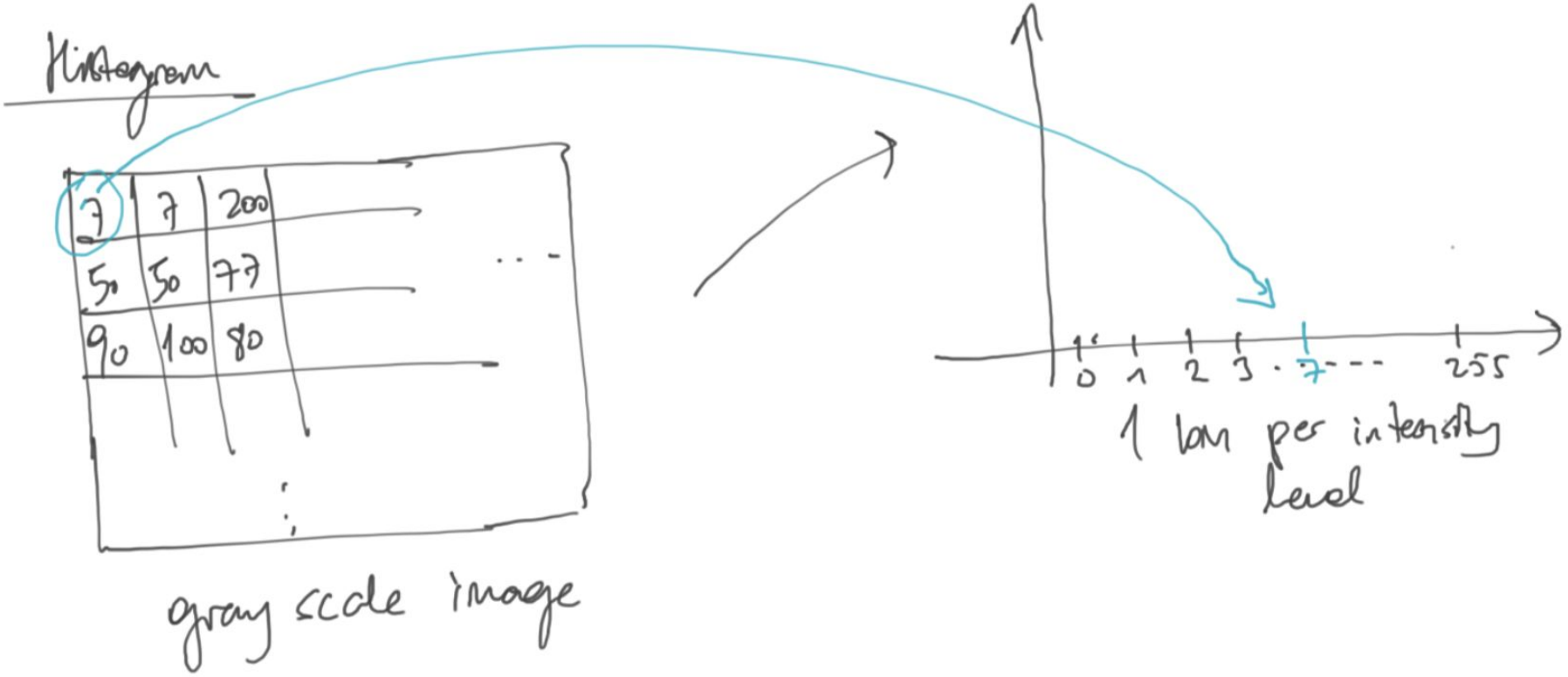
Histogram

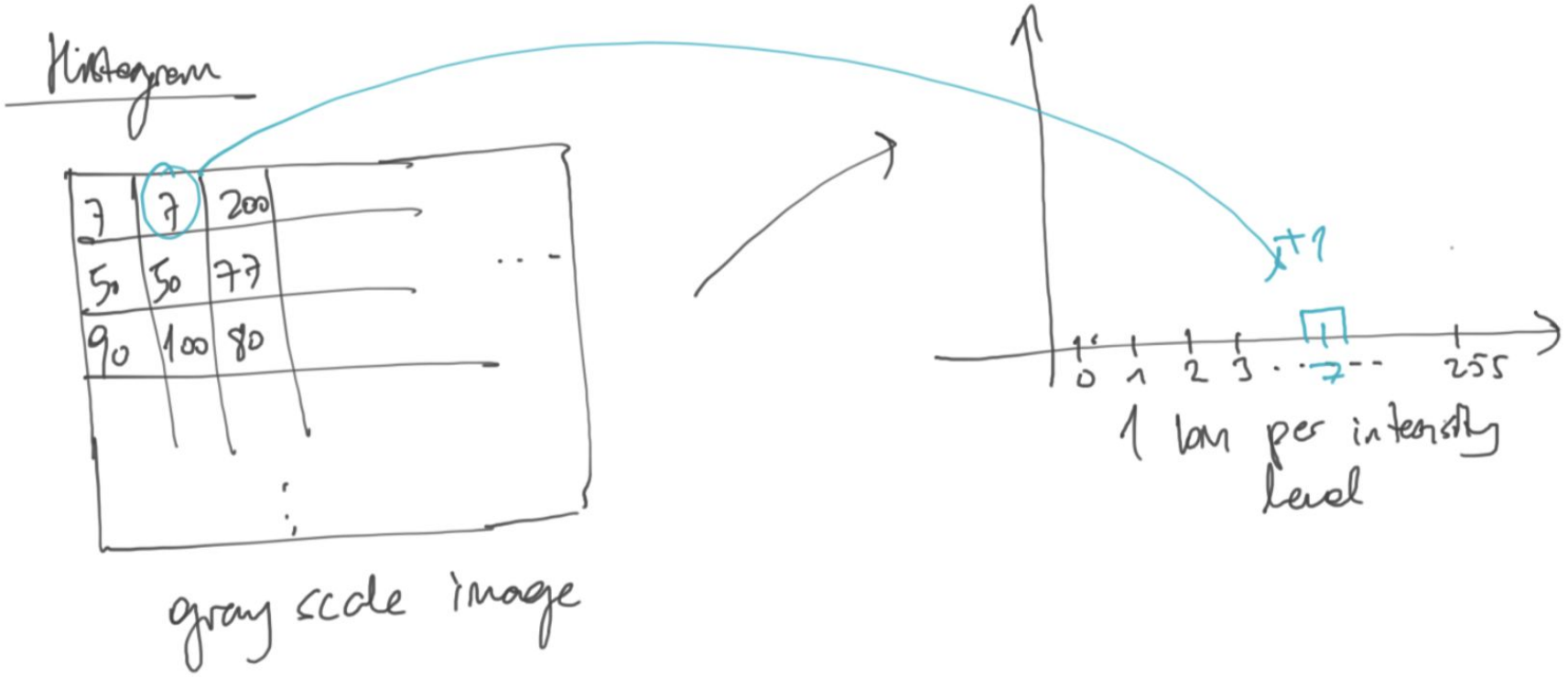


gray scale image

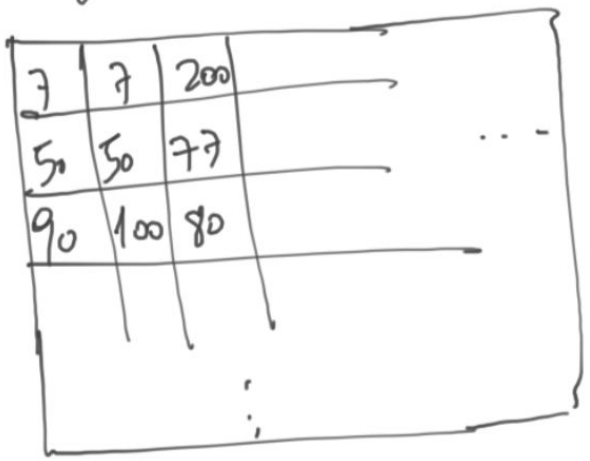


1 bin per intensity level

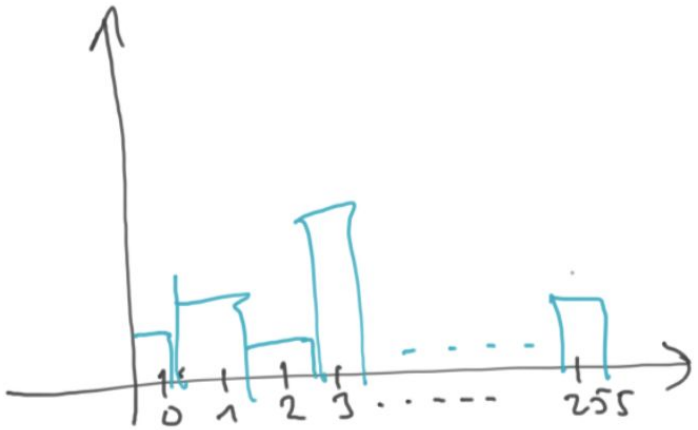

Histogram
7 7 200
50 50 77
90 100 80
gray scale image
0 1 2 3 7 255
1 bar per intensity level


Histogram
7
7
200
50
50
77
90
100
80
gray scale image
x+1
0
1
2
3
7
255
1 bin per intensity level

Histogram

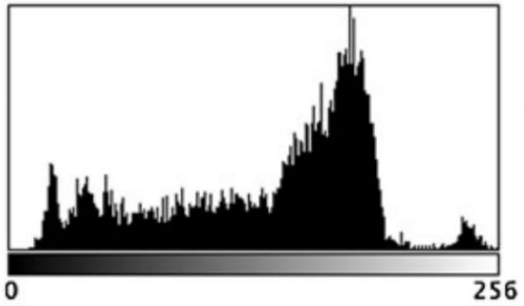


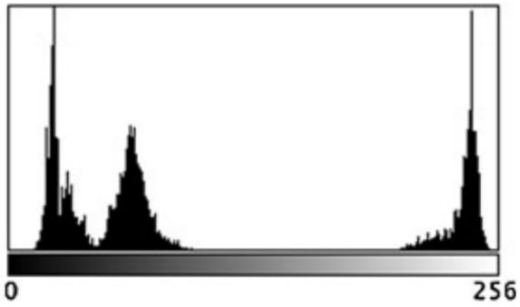
gray scale image



1 bar per intensity level

How to improve the histogram? (ie. What can go wrong with 1 bin per intensity level?)

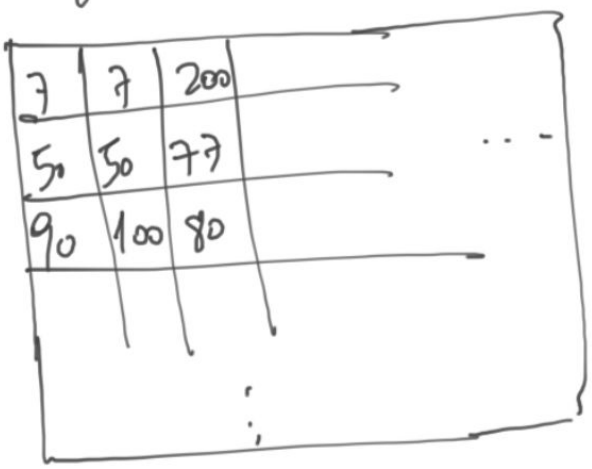




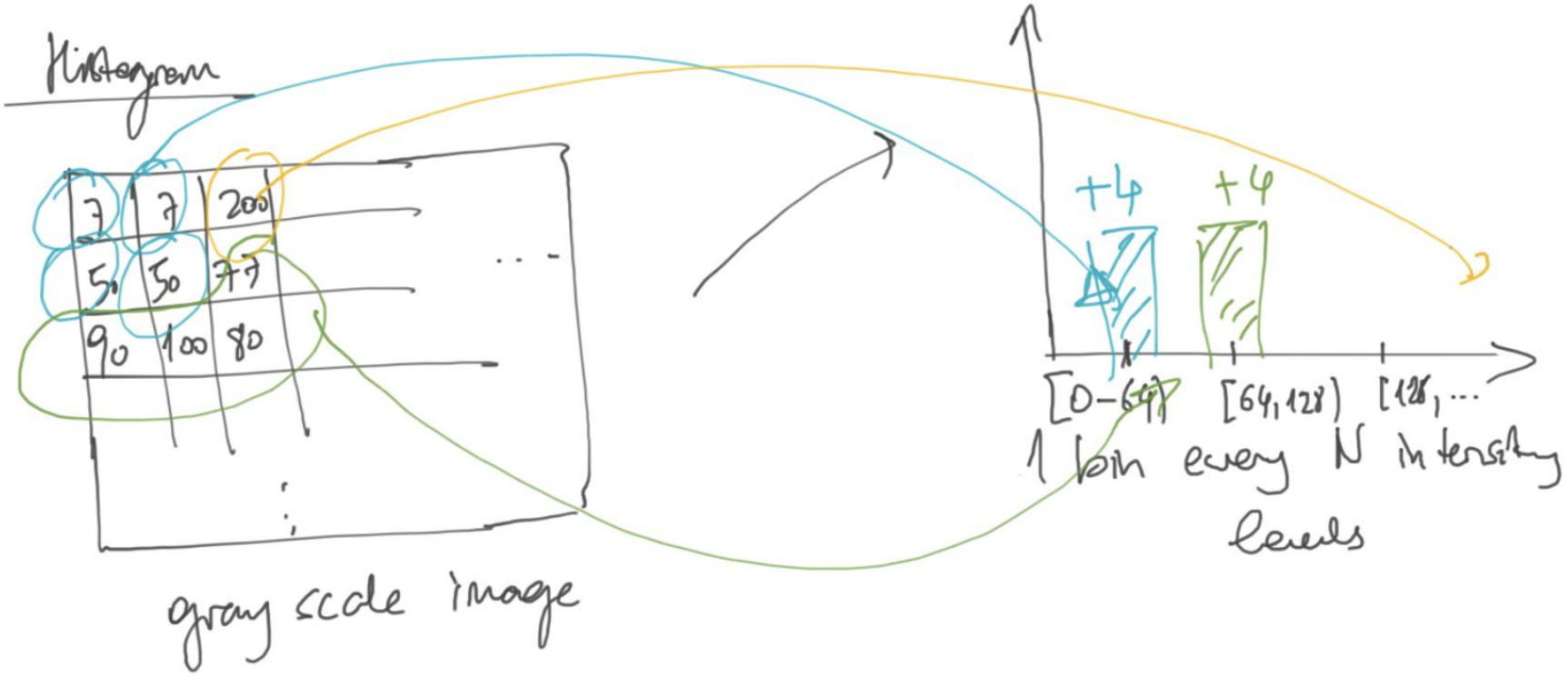
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Dr. Mubarak Shah

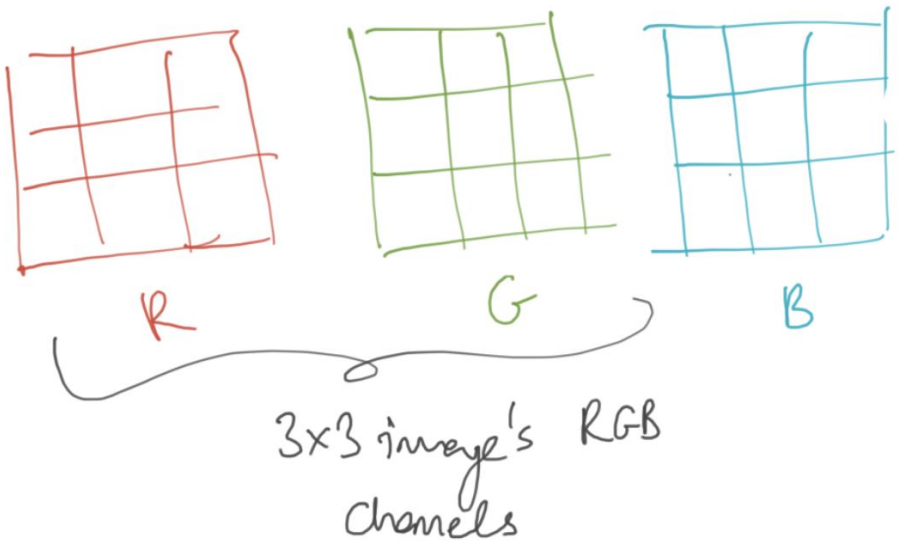
Histogram



gray scale image

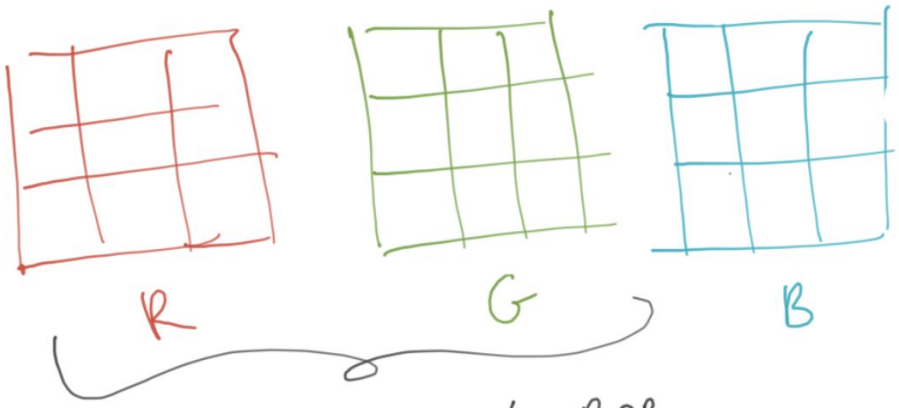

Histogram
7
7
200
50
50
77
90
100
80
+4
+4
[0-64)
[64,128)
[128,...
1 bin every N intensity levels
gray scale image

Color histogram

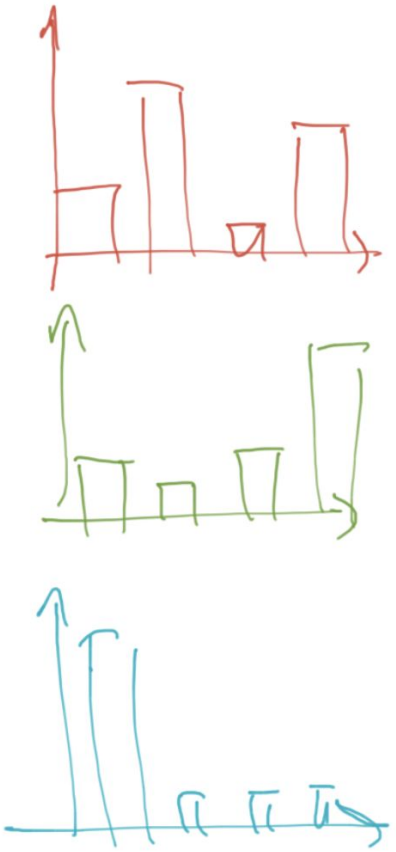


?

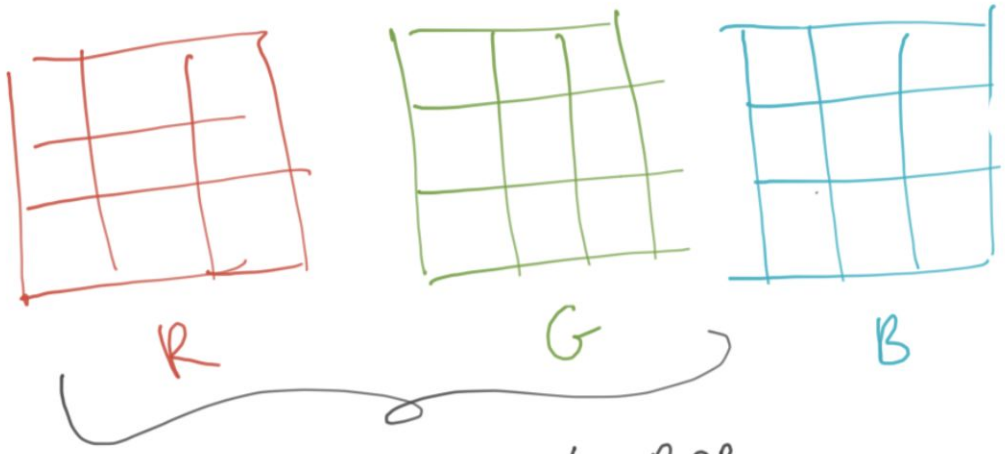
Color histogram



1 histogram per channel

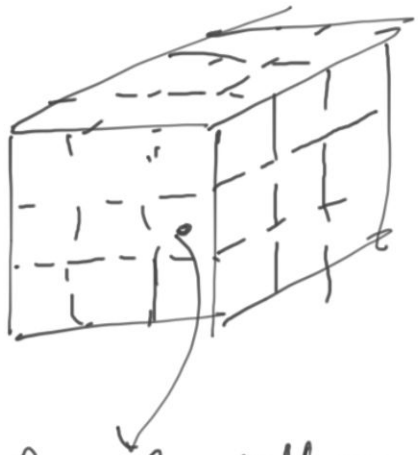


Color histogram



3x3 image's RGB channels

Joint histogram



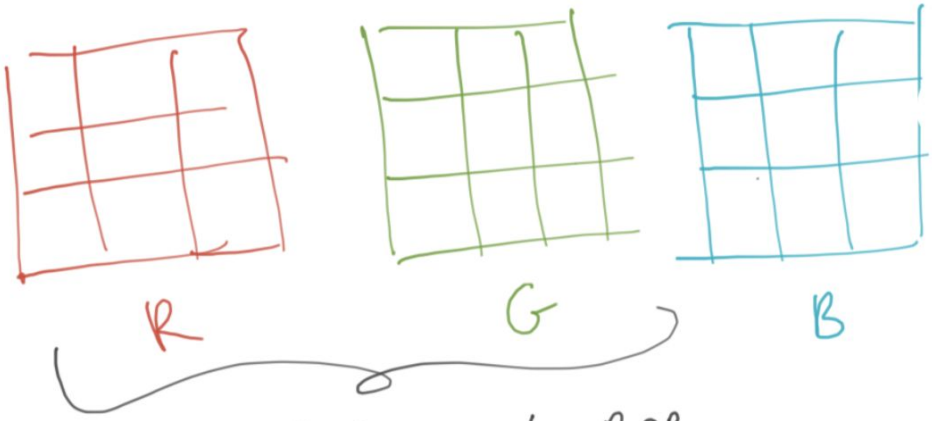
A single cell J:

Count of pixels in the range:

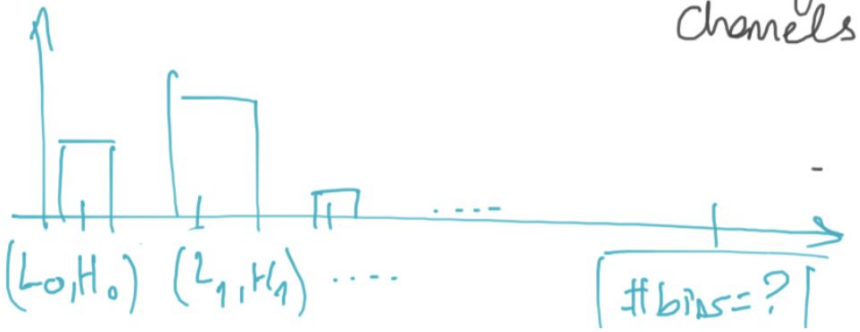
$$L_J^r \leq r < H_J^r, \; L_J^g \leq g < H_J^g$$

AND $L_J^b \leq b < H_J^b$

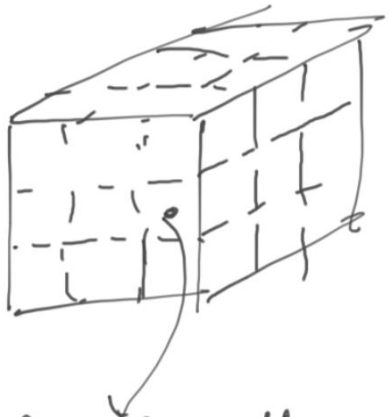
Color histogram



3×3 image's RGB channels



Joint histogram



A single cell J:

Count of pixels in the range:

$$\Leftarrow L_J^r \leq r < H_J^r,\ L_J^g \leq g < H_J^g$$

AND $L_J^b \leq b < H_J^b$

Color histogram

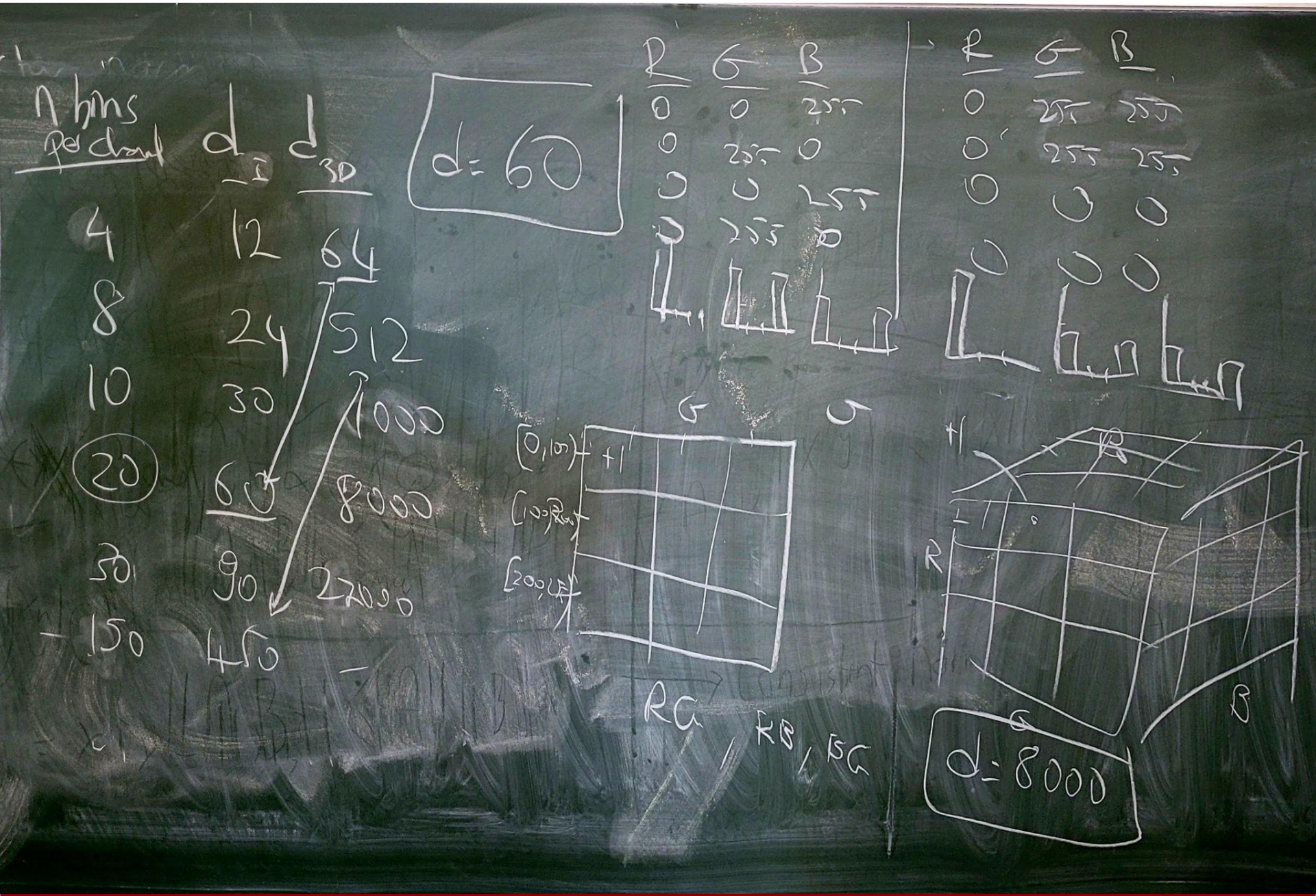
Joint histogram

THINK OF ADVANTAGES/DISADVANTAGES:

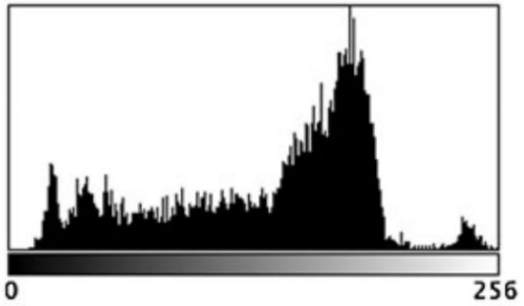
- Number of bins in grayscale/color histograms
- Grayscale vs color histogram
- Per-channel vs joint (n-dim) color histogram

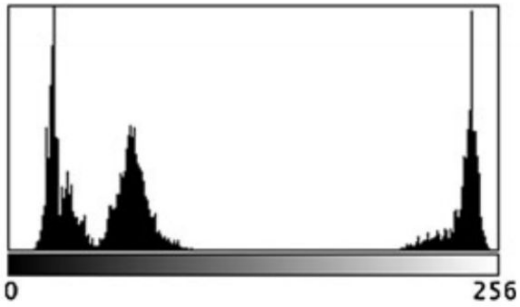
$\Leftarrow L_J^r \leq r < H_J^r, \; L_J^g \leq g < H_J^g$

AND $L_J^b \leq b < H_J^b$


n bins per channel
d_1
d_{3D}
4 12 64
8 24 512
10 30 1000
20 60 8000
50 150 27000
d = 60
R G B
0 0 255
0 255 0
255 0 0
R G B
0 255 255
255 255 0
0 0 0
G
R
B
d = 8000

Re-think of histogram's application





Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

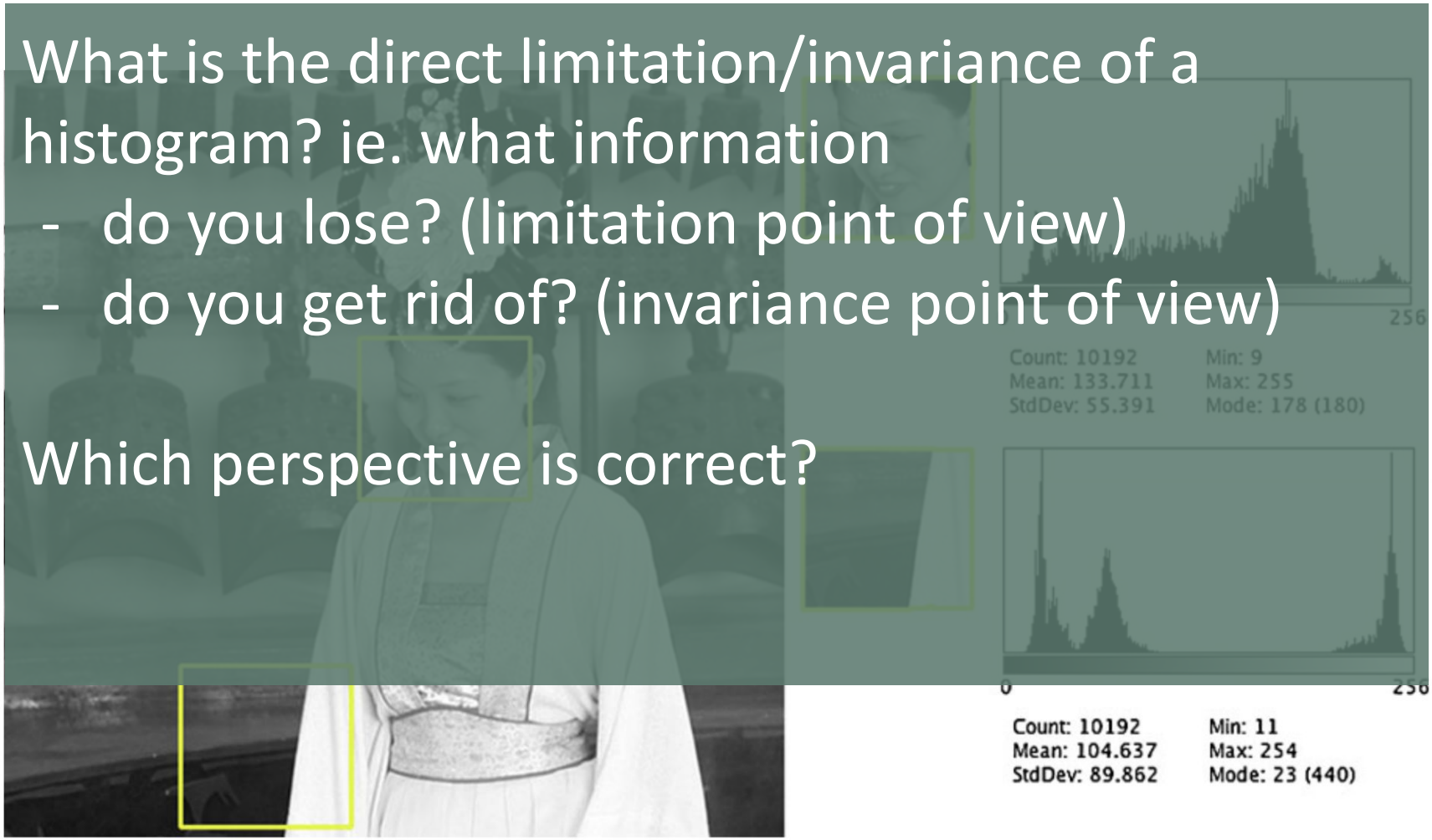
Slide credit: Dr. Mubarak Shah

Re-think of histogram's application

What is the direct limitation/invariance of a histogram? ie. what information

- do you lose? (limitation point of view)
- do you get rid of? (invariance point of view)

Which perspective is correct?



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Dr. Mubarak Shah

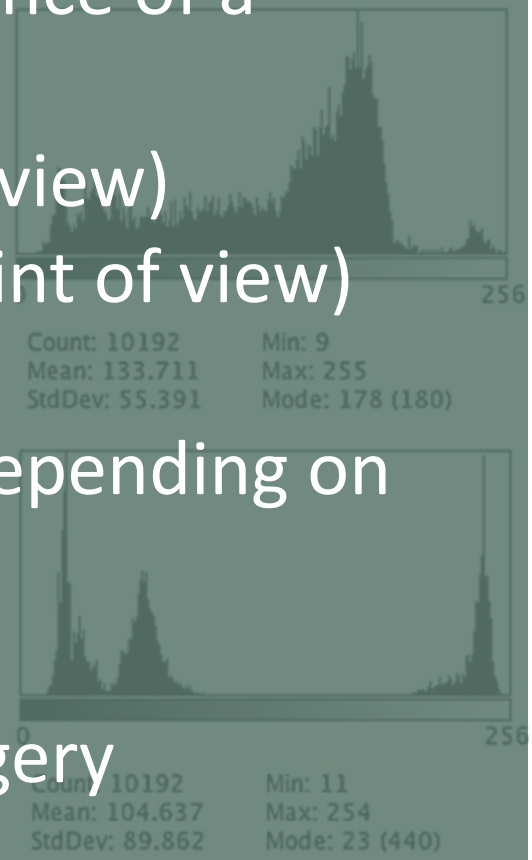
Re-think of histogram's application

What is the direct limitation/invariance of a histogram? ie. what information

- do you lose? (limitation point of view)
- do you get rid of? (invariance point of view)

Both points of views can be valid, depending on the task! Examples:

- detection of camera orientation
- detection of cars in satellite imagery



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Dr. Mubarak Shah

Re-think of histogram's application

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

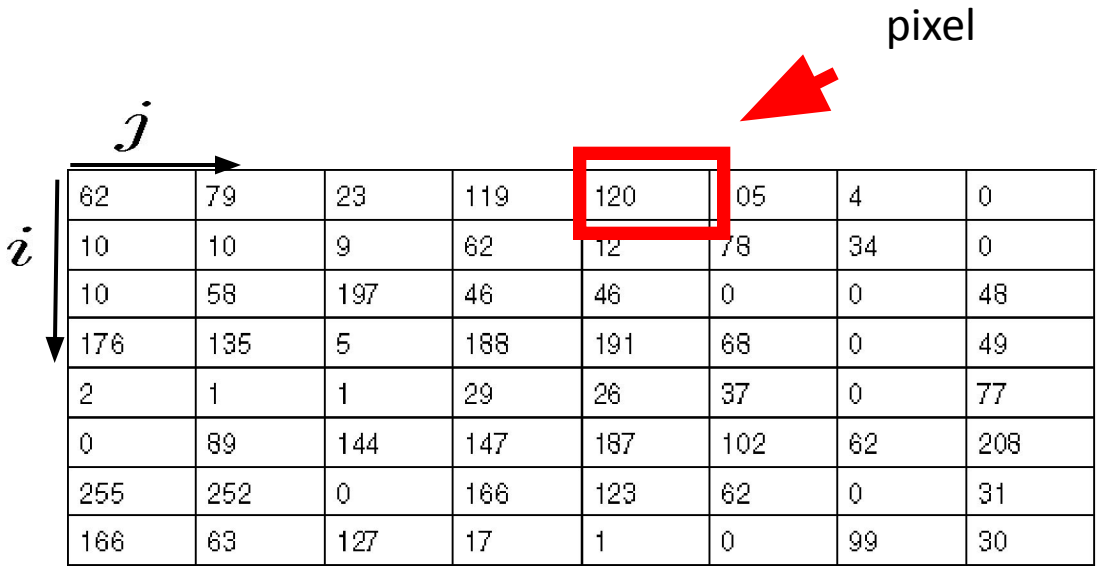
Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Images as discrete functions

- Images are usually **digital** (**discrete**):
  - **Sample** the 2D space on a regular grid

- Represented as a matrix of integer values



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

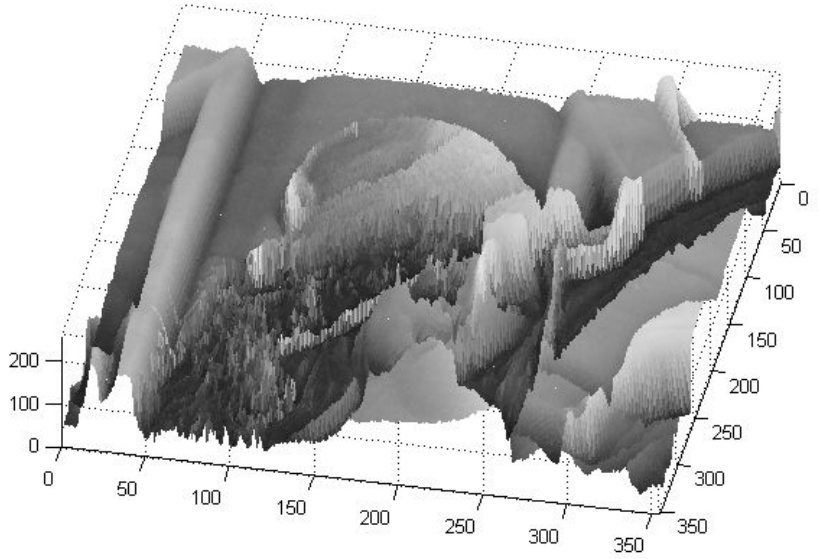
Images as functions

- **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:

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

Domain support

range



Adapted from slides hna

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:

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

Domain support

range

- A color image:

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Histograms are a type of image function

- Histogram can be seen as another f: region → vector mapping.
- In recognition tasks for computer vision, *image representation* typically plays a critical role.
- Throughout the course, we'll progressively discuss **more powerful representation** extraction techniques.

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Histograms are a type of image function

- Histogram can be seen as another f: region → vector mapping.
- In recognition tasks for computer vision, *image representation* typically plays a critical role.
- Throughout the course, we'll progressively discuss **more powerful representation** extraction techniques.

More *powerful* is <u>not</u> necessarily more *detailed*. (Think of the trade-offs in histograms, as a good example.)

Utility of a representation is typically task specific and plays a central role in the success of computer vision approaches.

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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.**

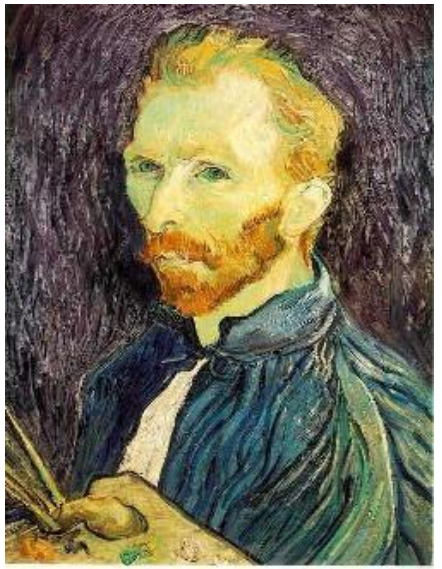
$$f[n,m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n,m]$$

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

De-noising

Salt and pepper noise

Super-resolution



In-painting

Bertamio et al

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

2D discrete-space systems (**filters**)

A digital filter is a system that performs mathematical operations on a sampled, discrete-time signal. [Wikipedia]

- 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].

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

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

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

Adapted from slides by Juan Carlos N

Filter example #1: Moving Average

2D discrete space 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

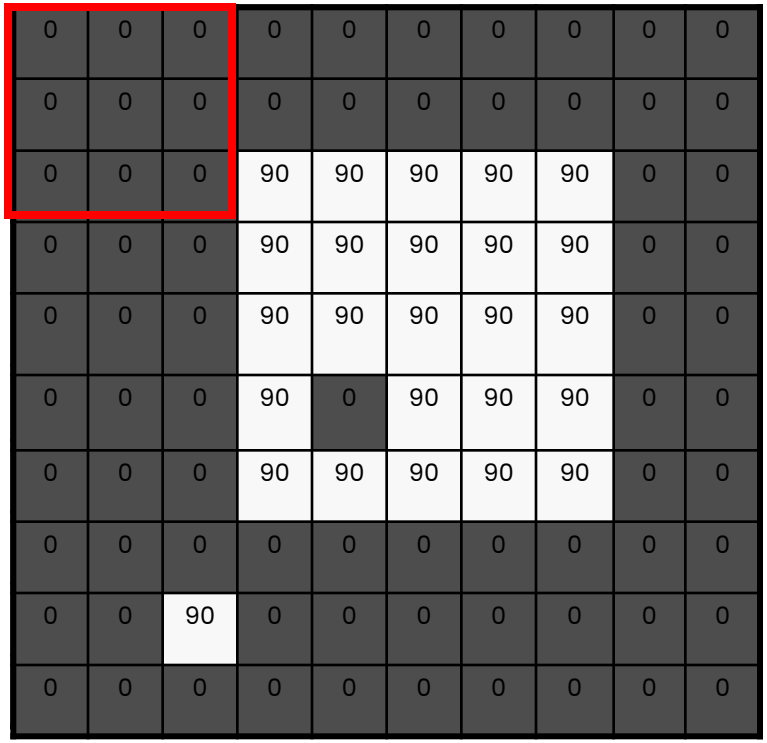
$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

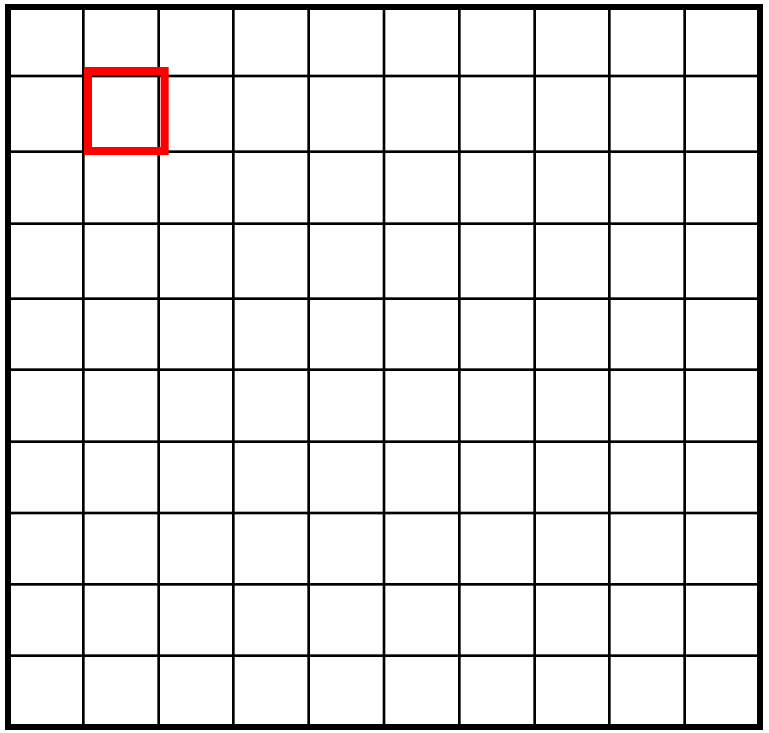
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Filter example #1: Moving Average

$F[x, y]$



$G[x, y]$

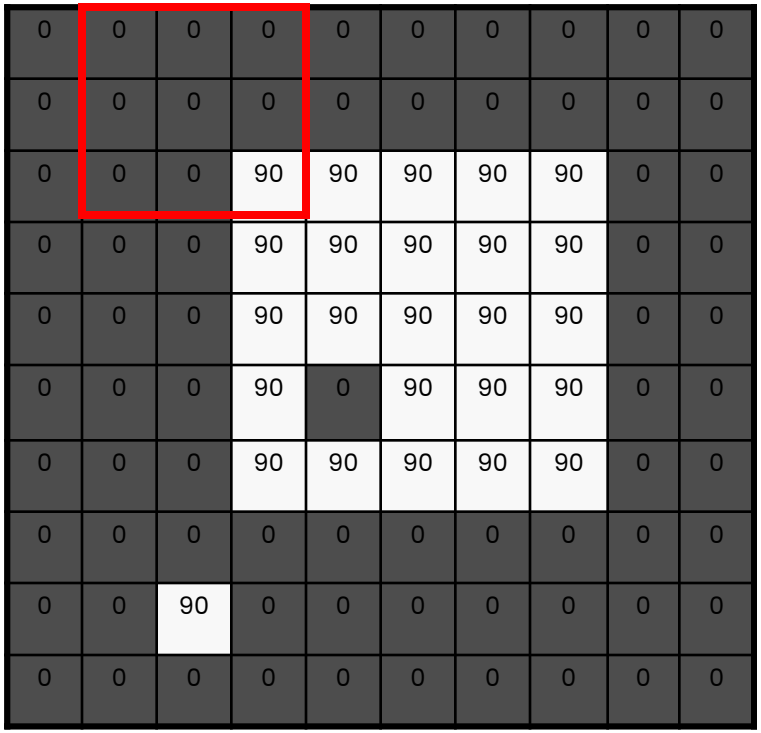


$$(f * h)[m, n] = \sum_{k,l} f[k, l]\, h[m - k, n - l]$$

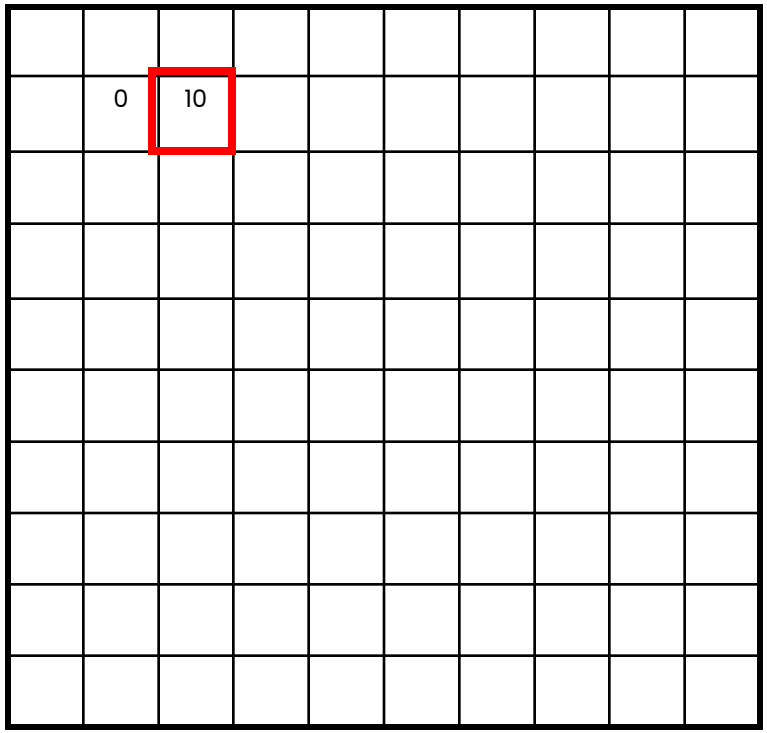
Courtesy of S. Seitz

Filter example #1: Moving Average

$F[x, y]$



$G[x, y]$



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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

$G[x, y]$

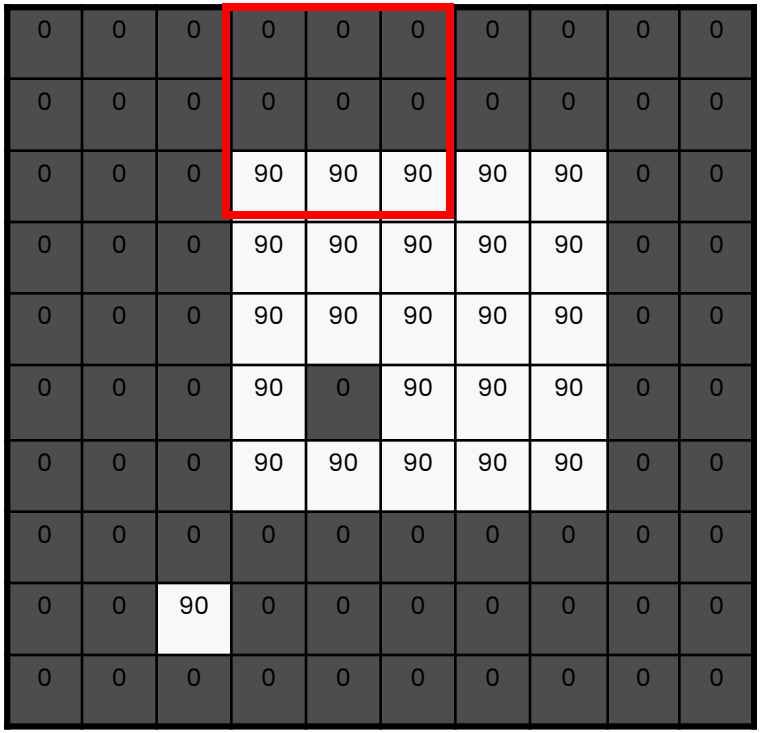
	0	10	20						

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

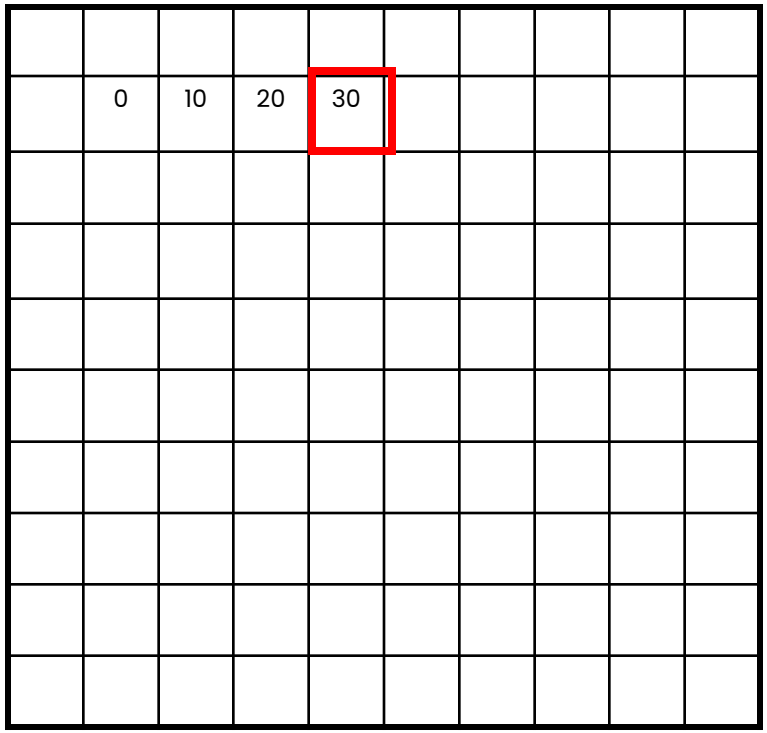
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Filter example #1: Moving Average

$F[x, y]$



$G[x, y]$

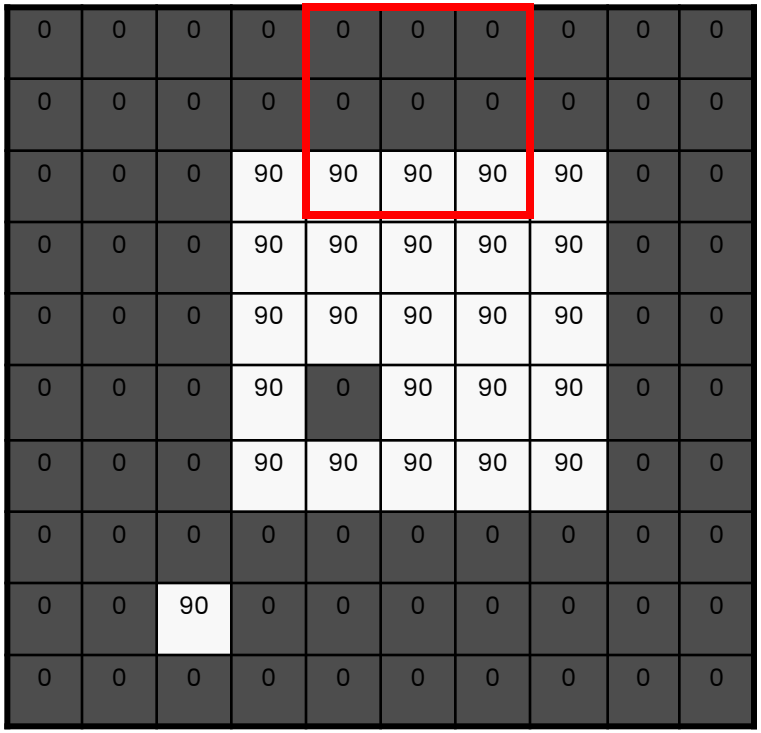


$$(f * h)[m, n] = \sum_{k,l} f[k, l]\, h[m - k, n - l]$$

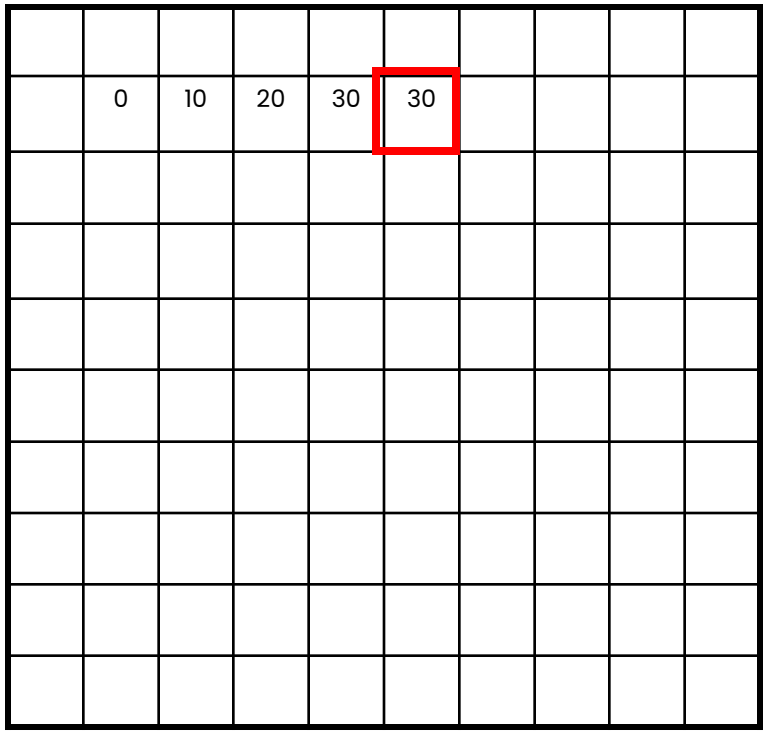
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Filter example #1: Moving Average

$F[x, y]$



$G[x, y]$



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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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	

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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)

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

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Filter example #1: Moving Average

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Filter example #2: Thresholding

- Image segmentation based on a simple threshold:

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Properties of linear systems

- Amplitude properties:
  - Additivity

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Properties of linear systems

- Amplitude properties:
  - Additivity

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

  - Homogeneity

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Properties of linear 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

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Properties of linear 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 (bounded-input, bounded-output (BIBO) stability)

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Properties of linear systems

- Amplitude properties:
  - Additivity [Required]

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

  - Homogeneity [Required]

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

  - Superposition (Generalizes additivity + homogeneity)

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

  - Stability [extra]

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

  - Invertibility [extra]

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

Properties of linear systems

- Spatial properties
  - Causality [extra]

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

  - Shift invariance [extra]
    Idea = same input at a *later time* yields the same response

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Is the moving average system is shift invariant?

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

$F[x,y]$

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	

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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!

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Is the moving average system is casual?

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

$F[x,y]$

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	

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

Adapted from sli

Is the moving average system is casual?

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

NO! It *looks* into future

$[x, y]$

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	

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

Adapted from sli

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Linear Systems (filters)

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

- Is the moving average a linear system?

(Exercise)

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

No!

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

A more interesting exercise

$y[n] = n \cdot x[n]$

- linear?
- shift/time-invariant?
- causal?
- stable?

An analysis can be found here:

https://www.youtube.com/watch?v=fO6RRXDacUw

What we will learn today?

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

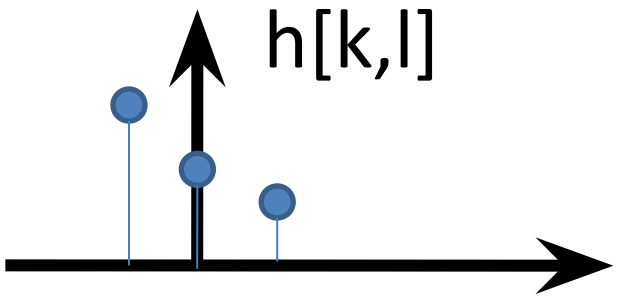
Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

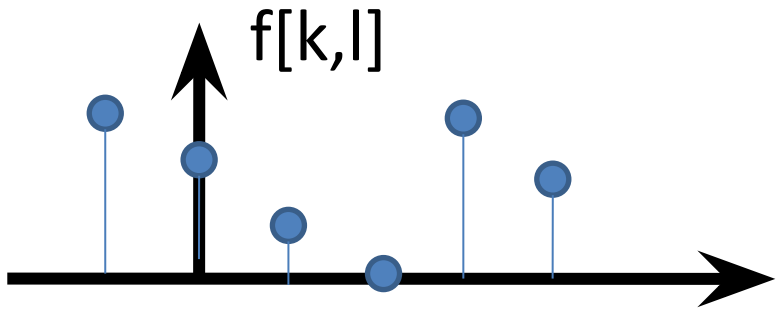
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

1D Discrete convolution (symbol: $*$)

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

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





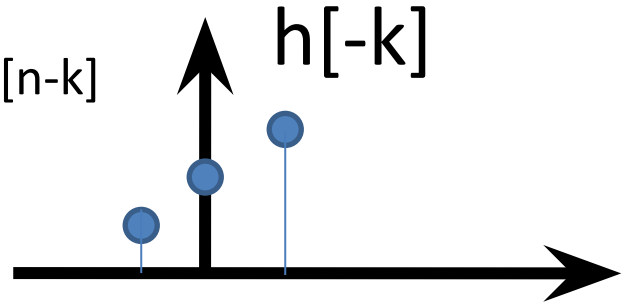
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

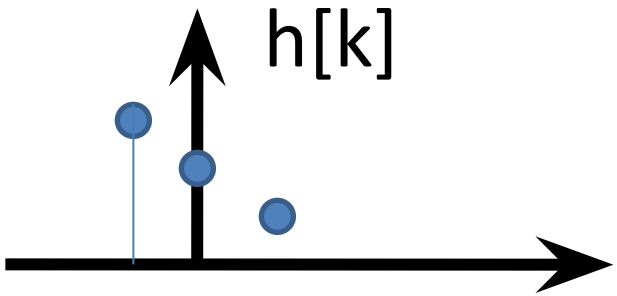
1D Discrete convolution (symbol $*$)

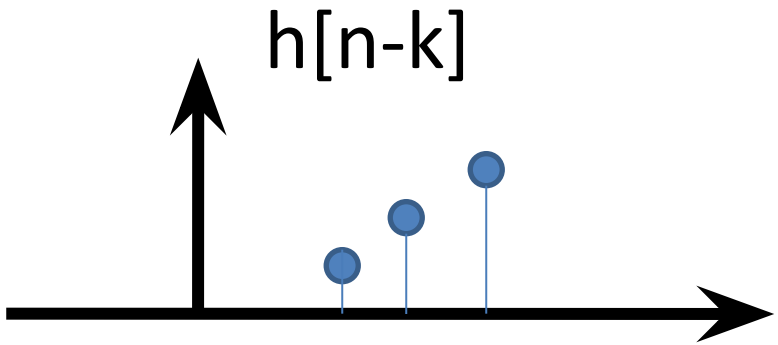
We are going to convolve a **function f** with a **filter h.**

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

We first need to calculate the shifted filter h[n-k]





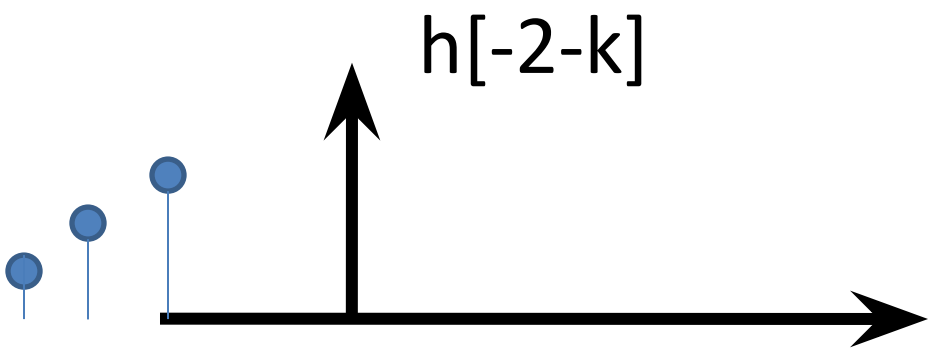


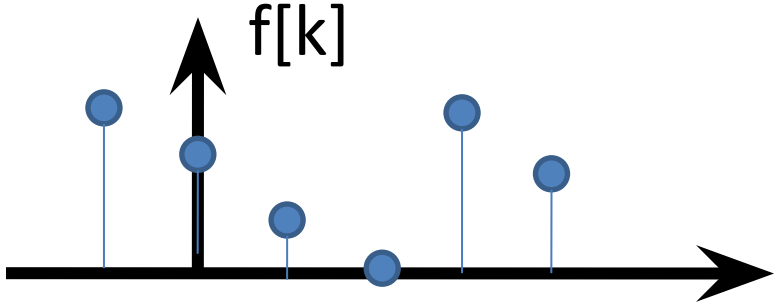
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

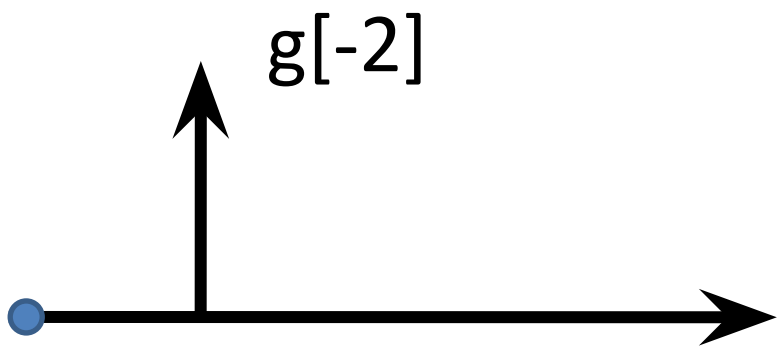
Discrete convolution (symbol: $*$)

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

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





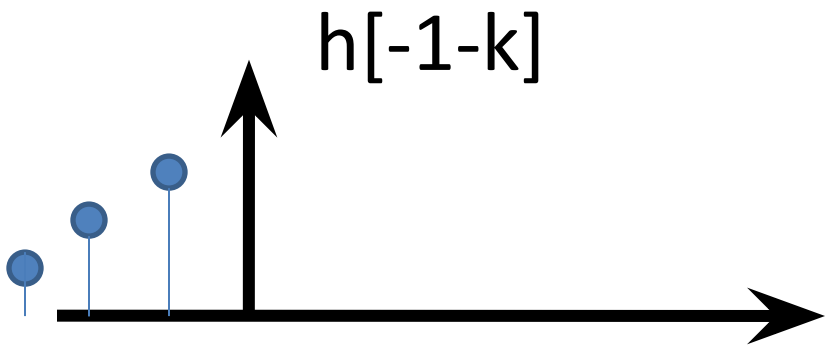


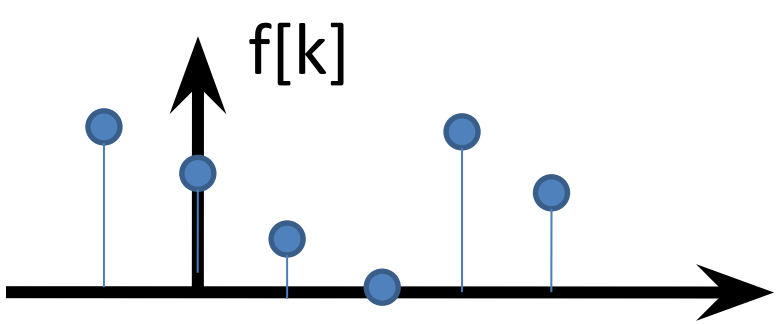
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

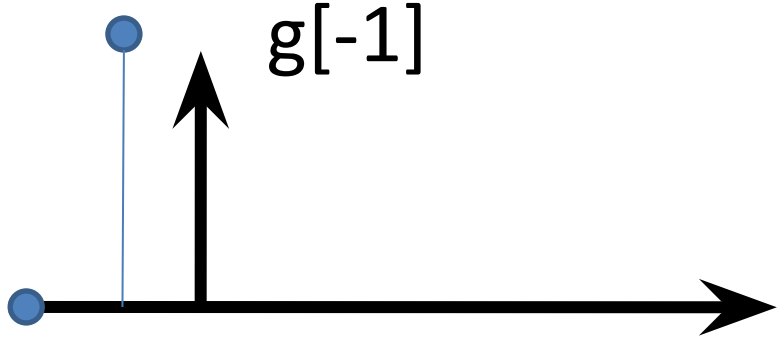
Discrete convolution (symbol: $*$)

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

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





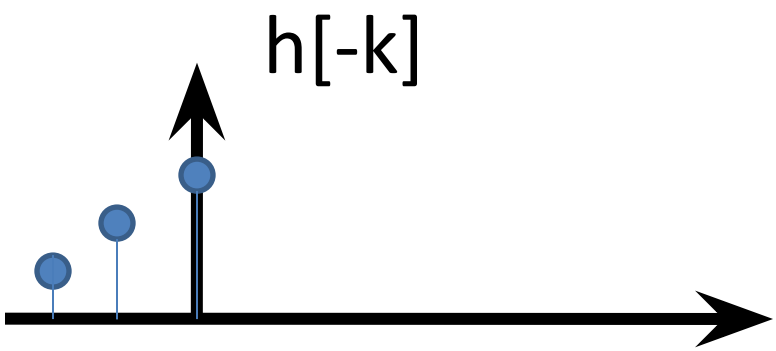


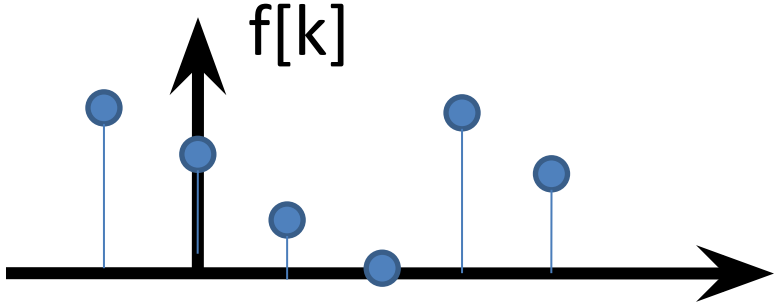
Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

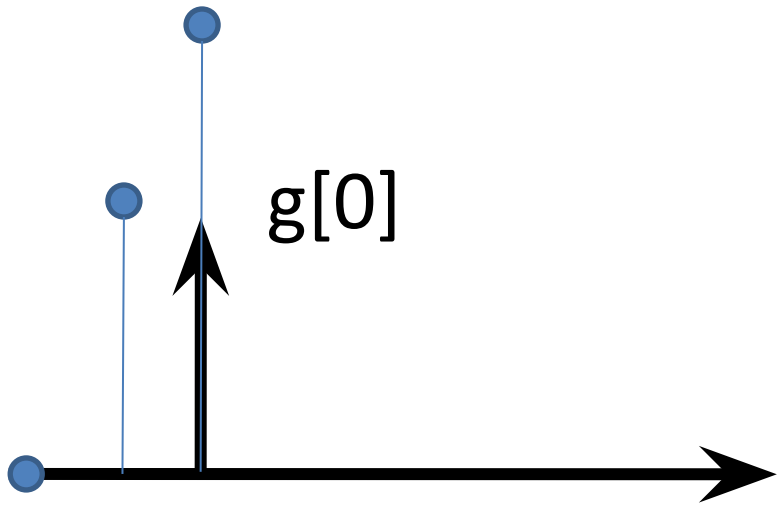
Discrete convolution (symbol: $*$)

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

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





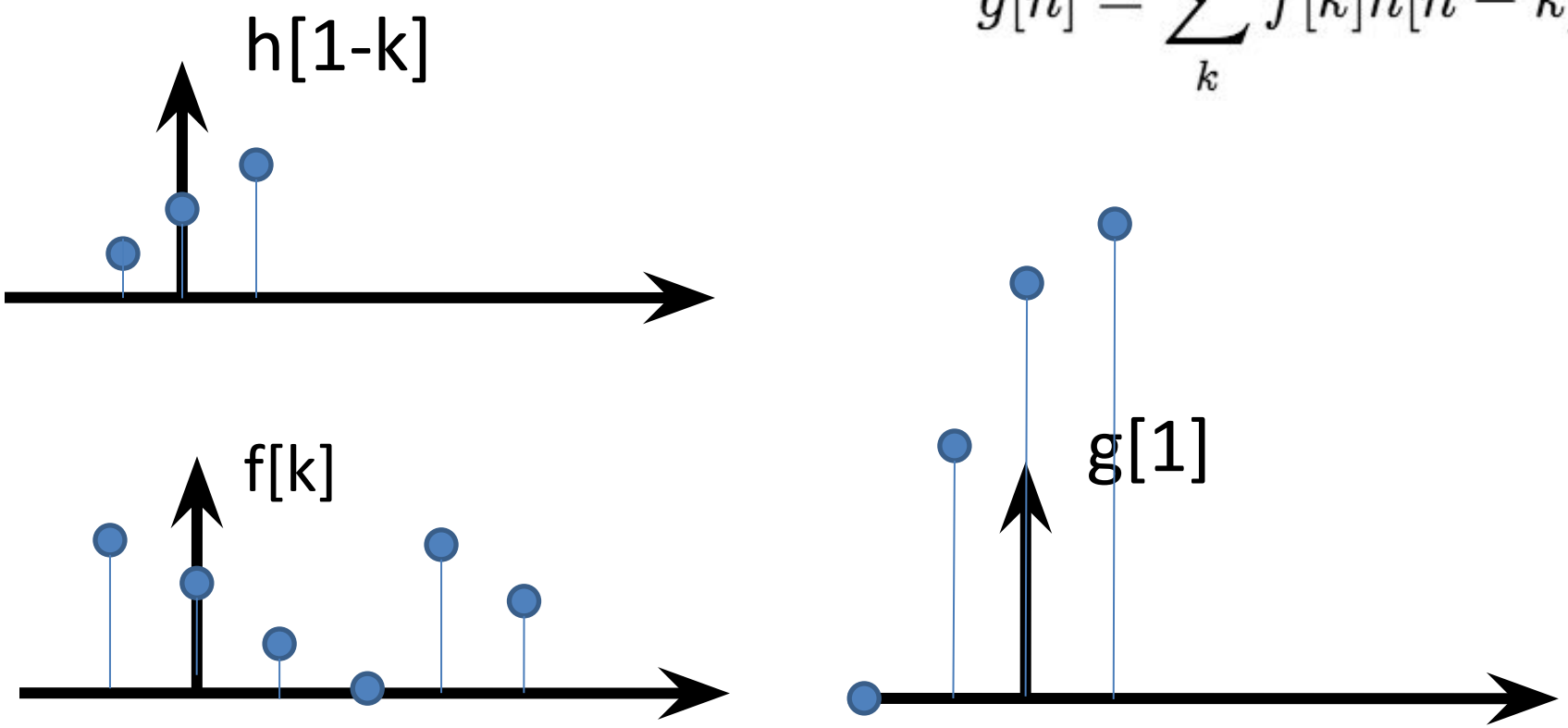


Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Discrete convolution (symbol: $*$)

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

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

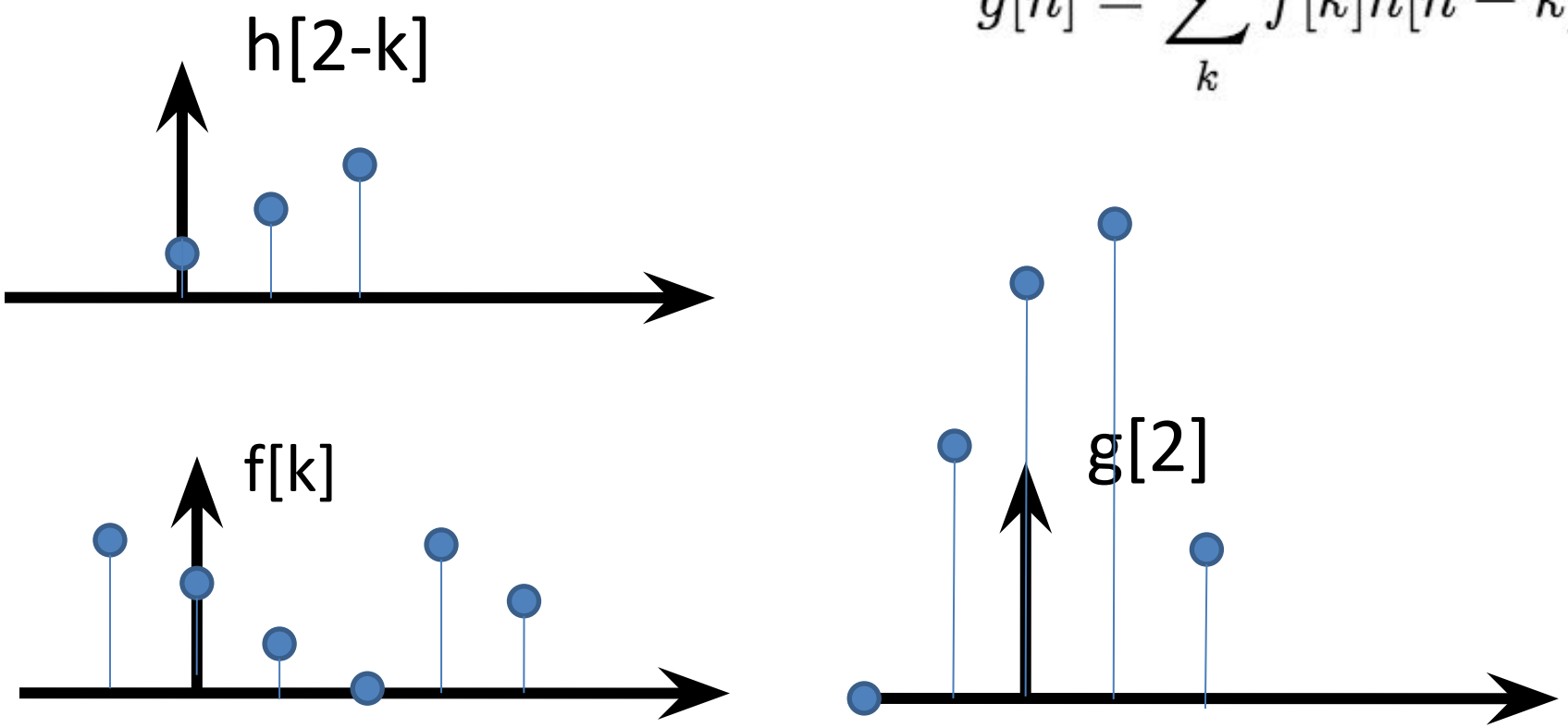


Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Discrete convolution (symbol: $*$)

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

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

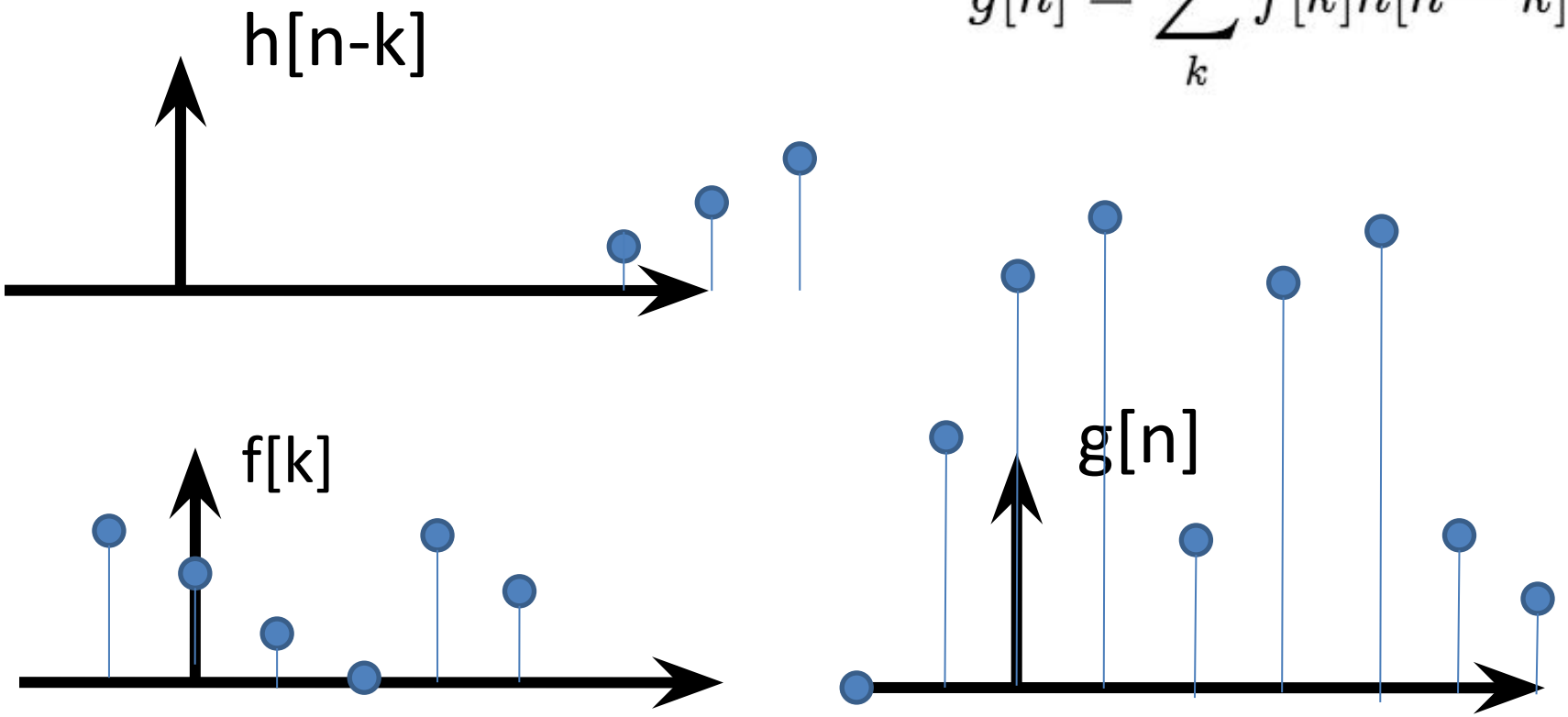


Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Discrete convolution (symbol: $*$)

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

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



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

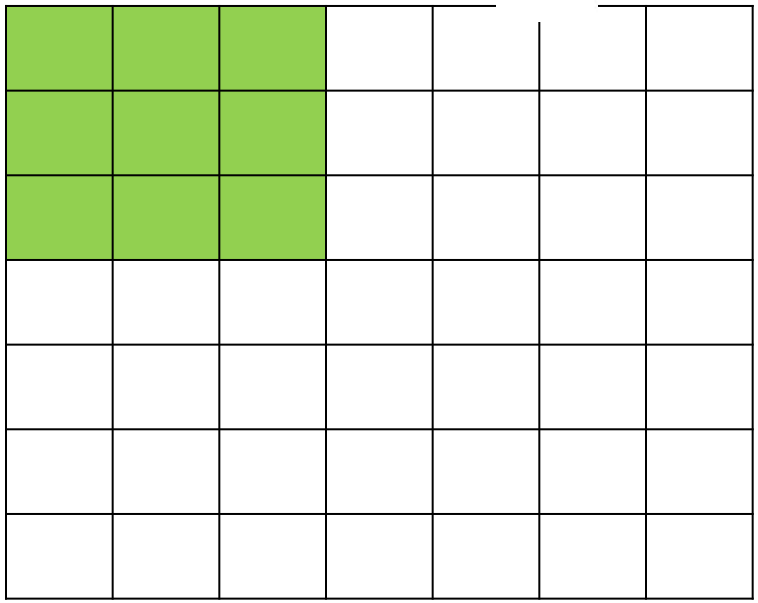
n

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. (flip the kernel in 2 axis.)

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



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

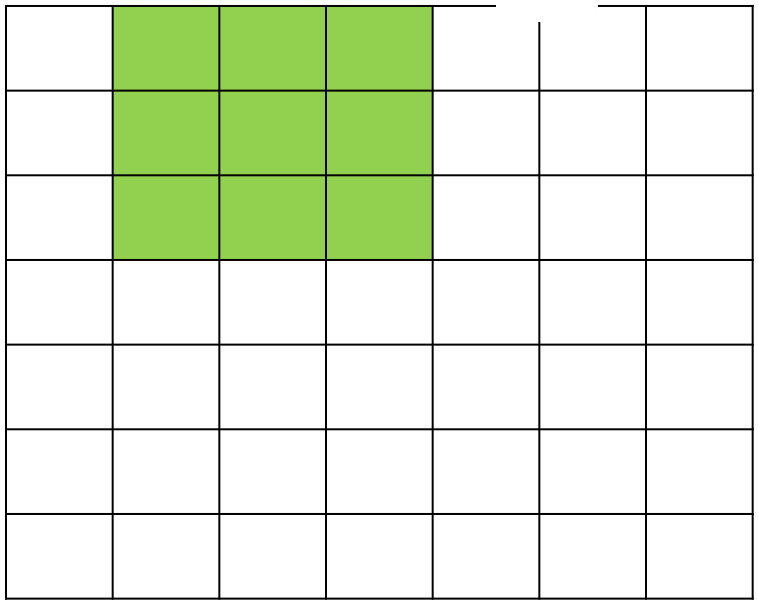
n

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. (flip the kernel in 2 axis.)

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



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

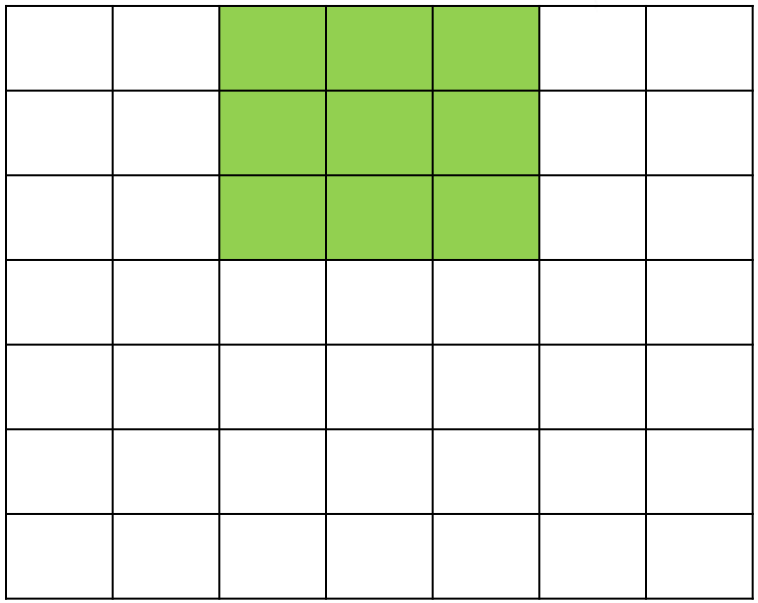
n

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. (flip the kernel in 2 axis.)

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



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

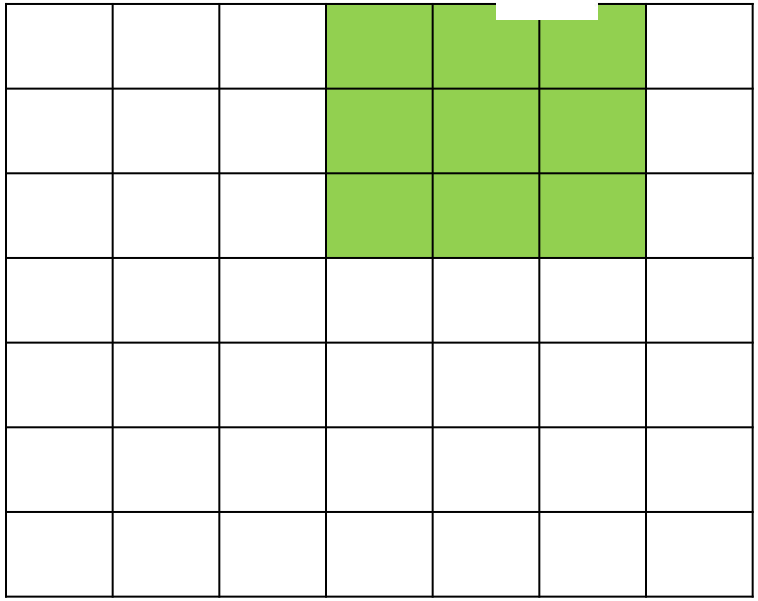
n

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. (flip the kernel in 2 axis.)

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



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

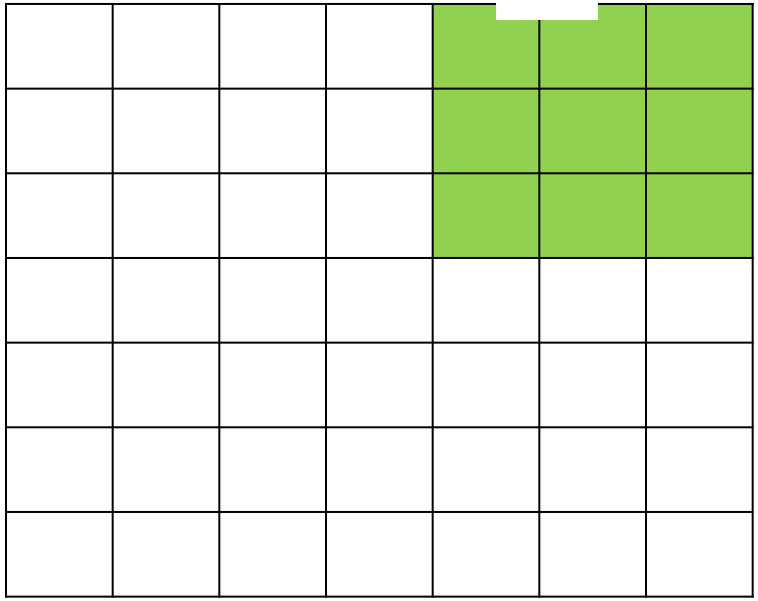
n

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. (flip the kernel in 2 axis.)

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



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

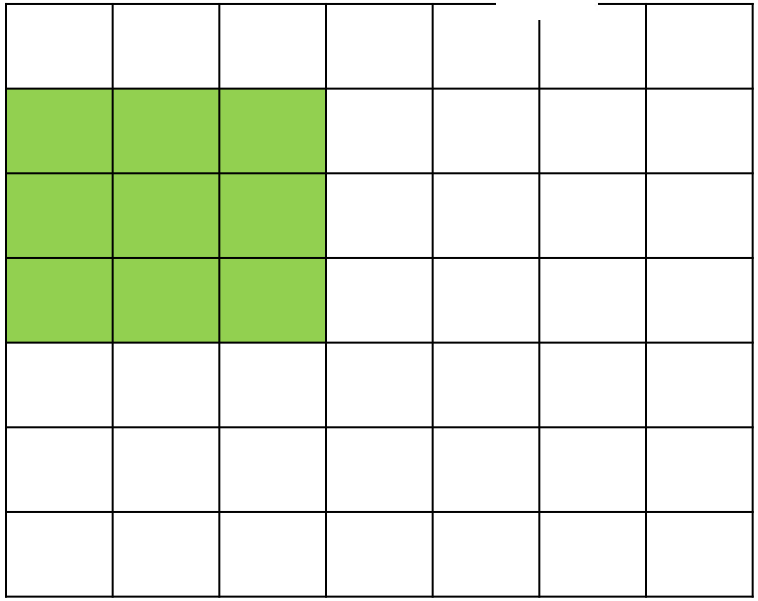
n

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. (flip the kernel in 2 axis.)

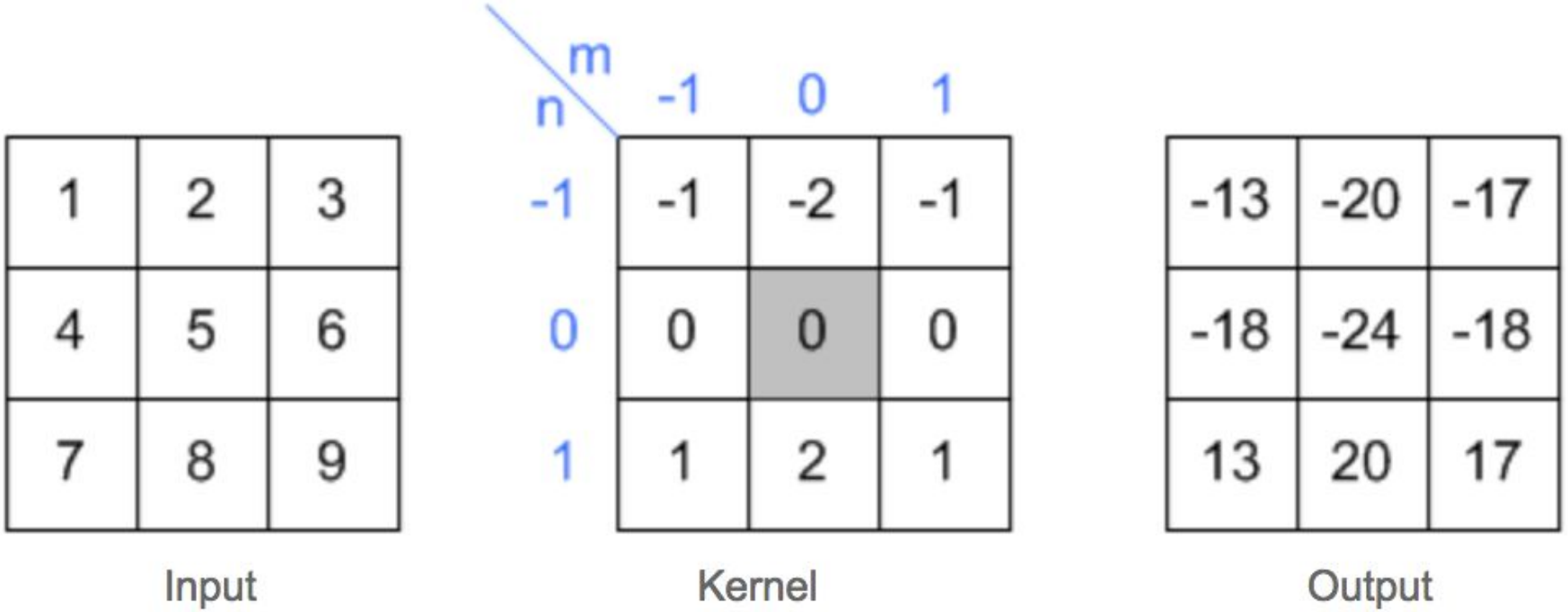
$$f[n,m] * h[n,m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k,l]\, h[n-k, m-l]$$



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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

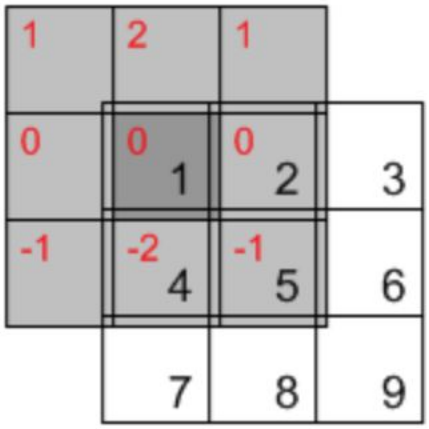
2D convolution example



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Song Ho Ahn

2D convolution example



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

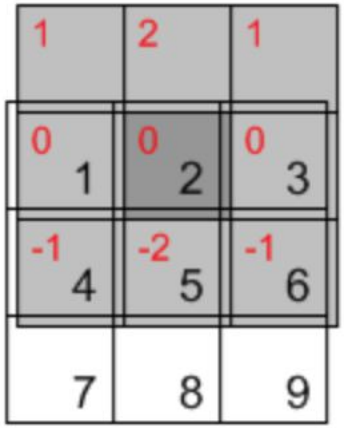
-13	-20	-17
-18	-24	-18
13	20	17

Output

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Song Ho Ahn

2D convolution example



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

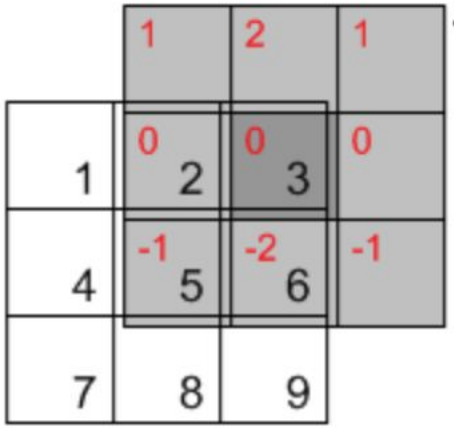
-13	-20	-17
-18	-24	-18
13	20	17

Output

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Song Ho Ahn

2D convolution example



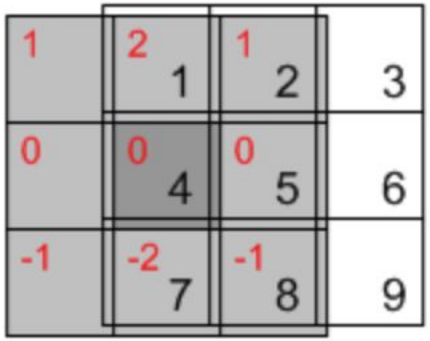
$$= x[1,-1]\cdot h[1,1] + x[2,-1]\cdot h[0,1] + x[3,-1]\cdot h[-1,1]$$
$$+ x[1,0]\cdot h[1,0] + x[2,0]\cdot h[0,0] + x[3,0]\cdot h[-1,0]$$
$$+ x[1,1]\cdot h[1,-1] + x[2,1]\cdot h[0,-1] + x[3,1]\cdot h[-1,-1]$$
$$= 0\cdot 1 + 0\cdot 2 + 0\cdot 1 + 2\cdot 0 + 3\cdot 0 + 0\cdot 0 + 5\cdot(-1) + 6\cdot(-2) + 0\cdot(-1) = -17$$

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

Output

Slide credit: Song Ho Ahn

2D convolution example



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

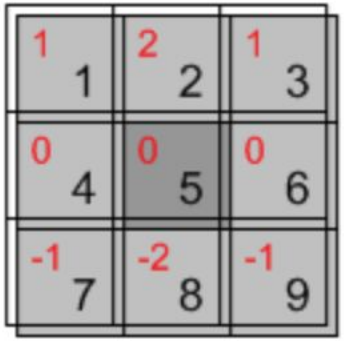
-13	-20	-17
-18	-24	-18
13	20	17

Output

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Song Ho Ahn

2D convolution example



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

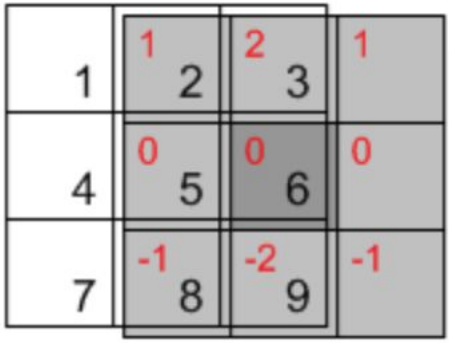
-13	-20	-17
-18	-24	-18
13	20	17

Output

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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	-18
13	20	17

Output

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Slide credit: Song Ho Ahn

Convolution in 2D - examples

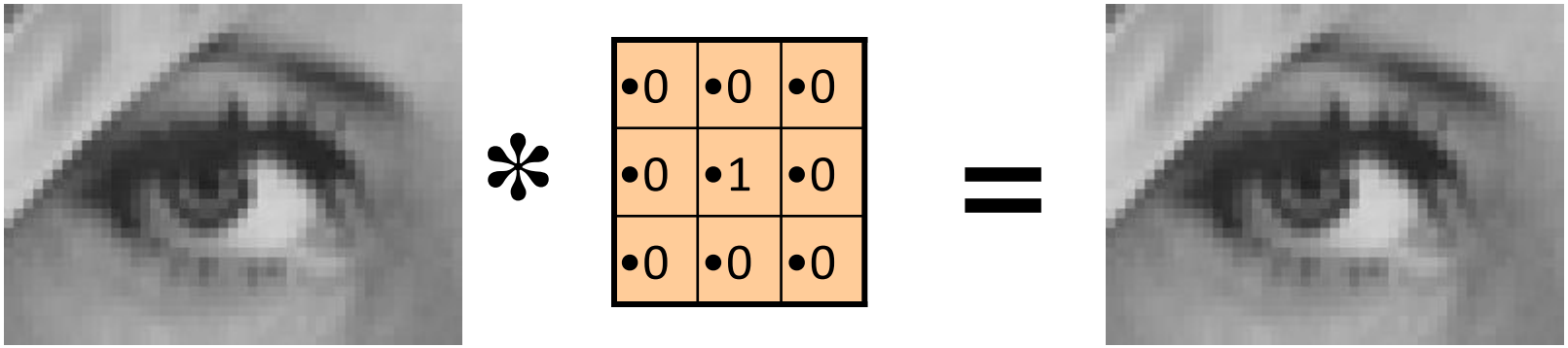
Original

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

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D - examples



Original

Filtered
(no change)

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D - examples

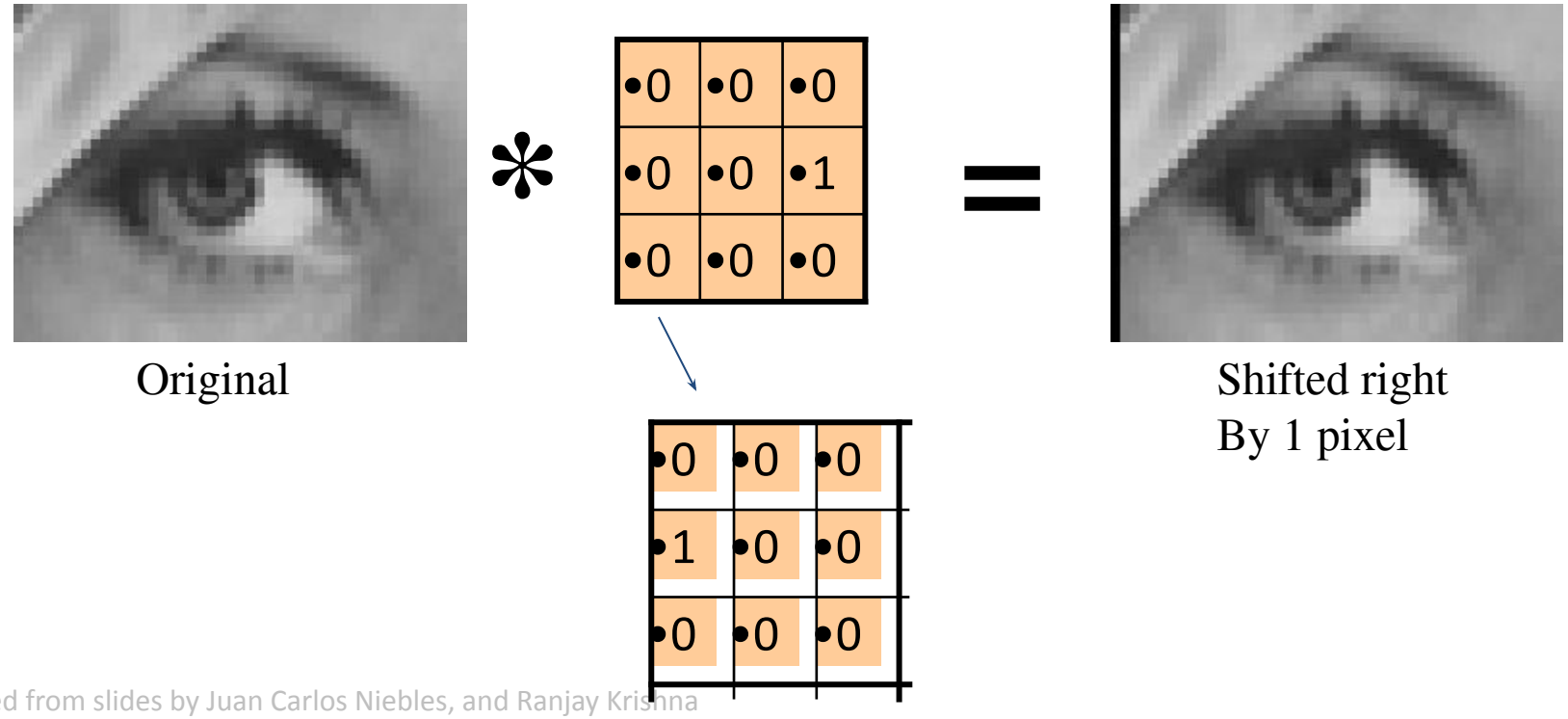
Original

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

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D - examples



Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D - examples

Original

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

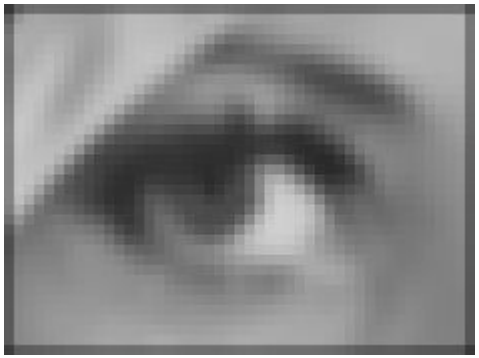
Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D - examples

Original

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



Blur (with a box filter)

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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)

"details of the image"

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

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

- What does blurring take away?

–

=

  - Let's add it back:

+ a

=

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution in 2D – Sharpening filter

Original

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

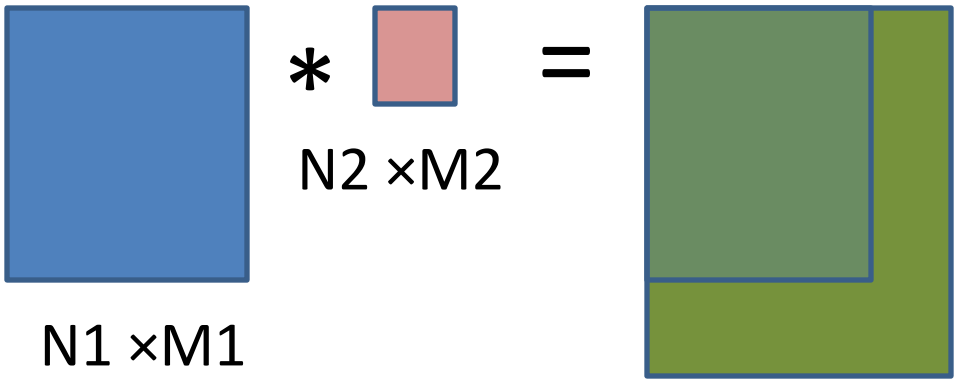
Sharpening filter: Accentuates differences with local average

Courtesy of D Lowe

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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 convolution of finite-support signals.

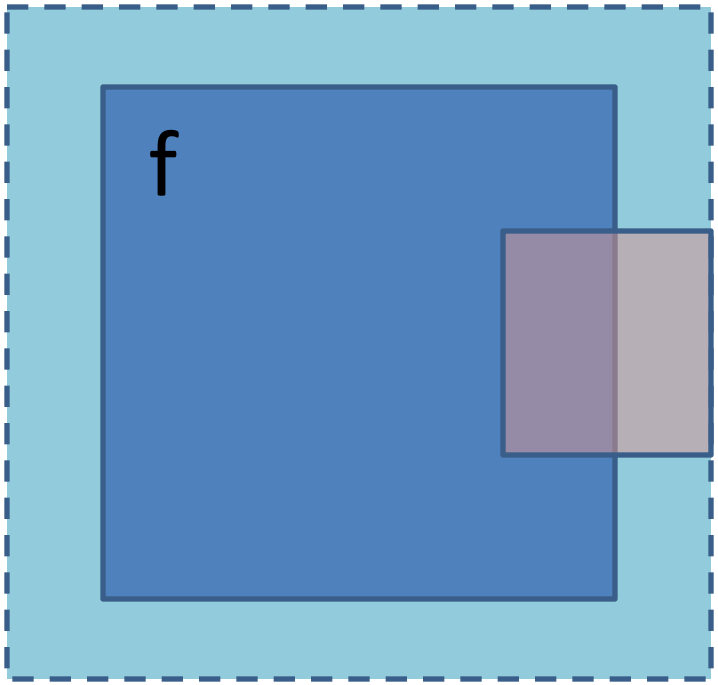


(N1 - N2 + 1) × (M1 - M2 + 1)

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Image support and edge effect

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



h

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

-> Matlab conv2 uses zero-padding

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

(Cross) correlation (symbol: $\star\star$)

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**

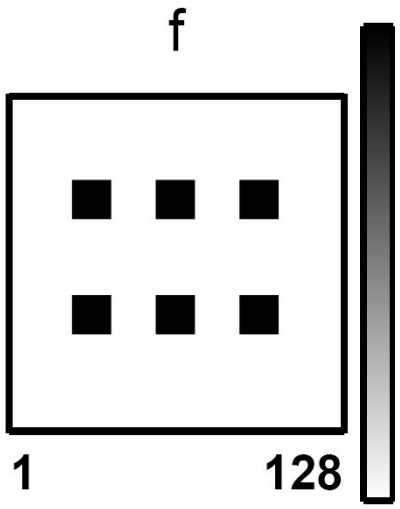
- Equivalent to a convolution without the flip

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

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

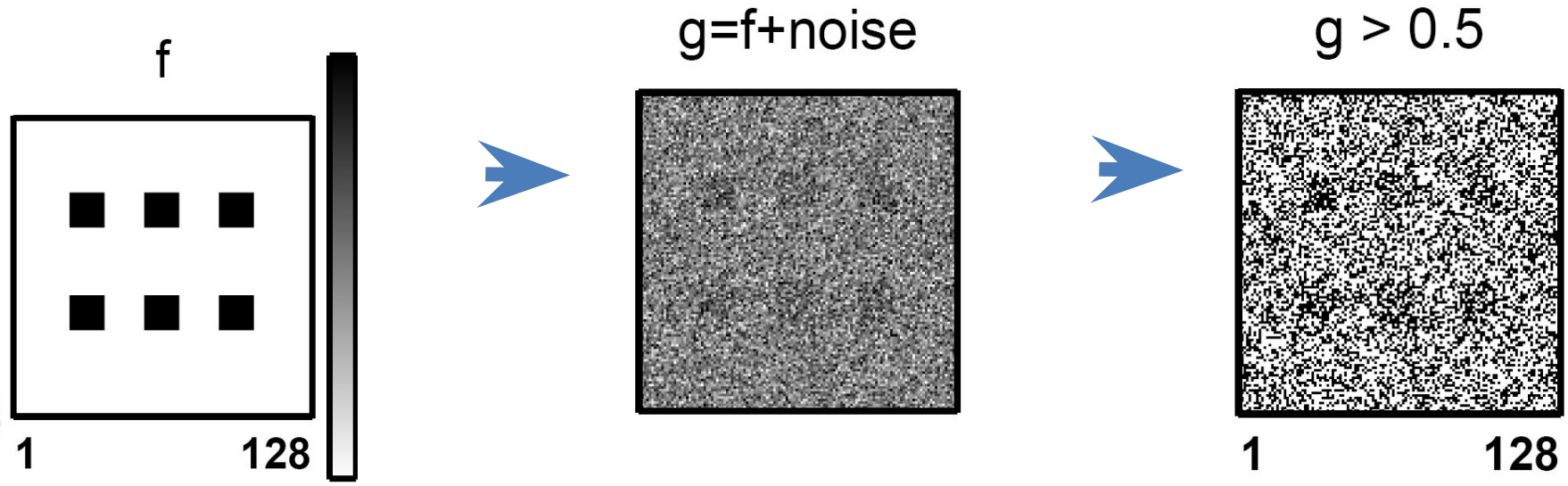
(Cross) correlation – example



Courtesy of J. Fessler

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

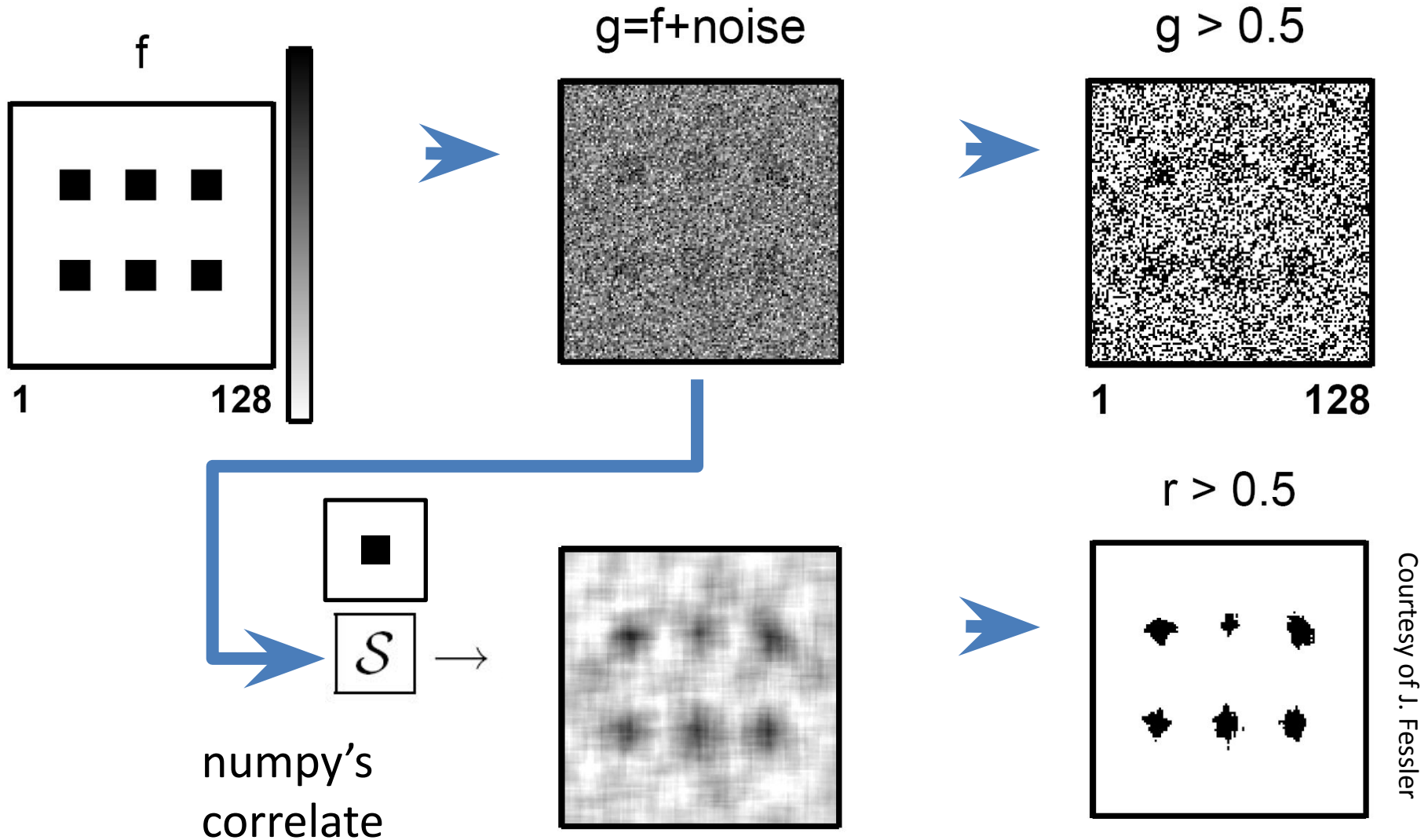
(Cross) correlation – example



Courtesy of J. Fessler

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

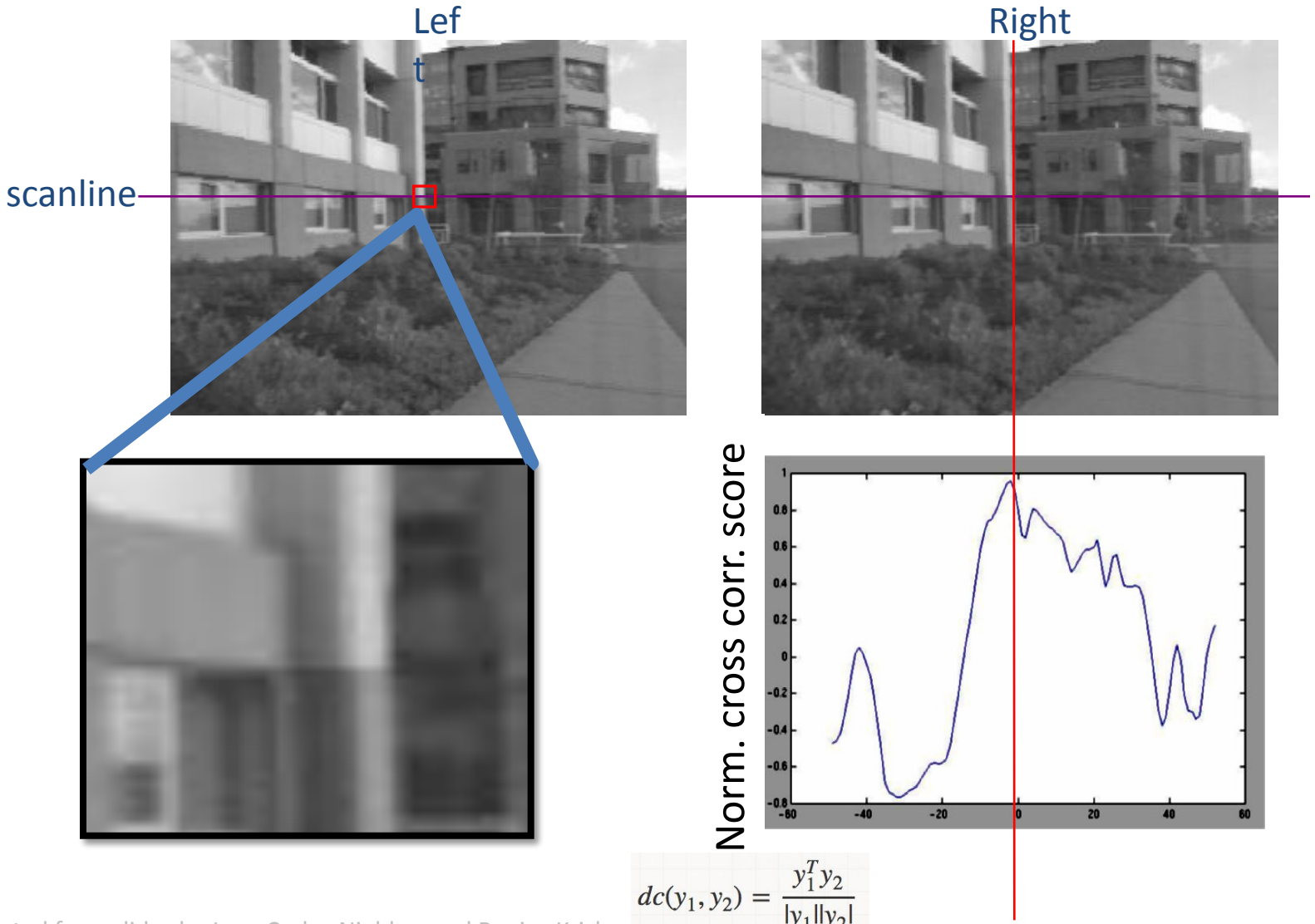
(Cross) correlation – example



Courtesy of J. Fessler

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

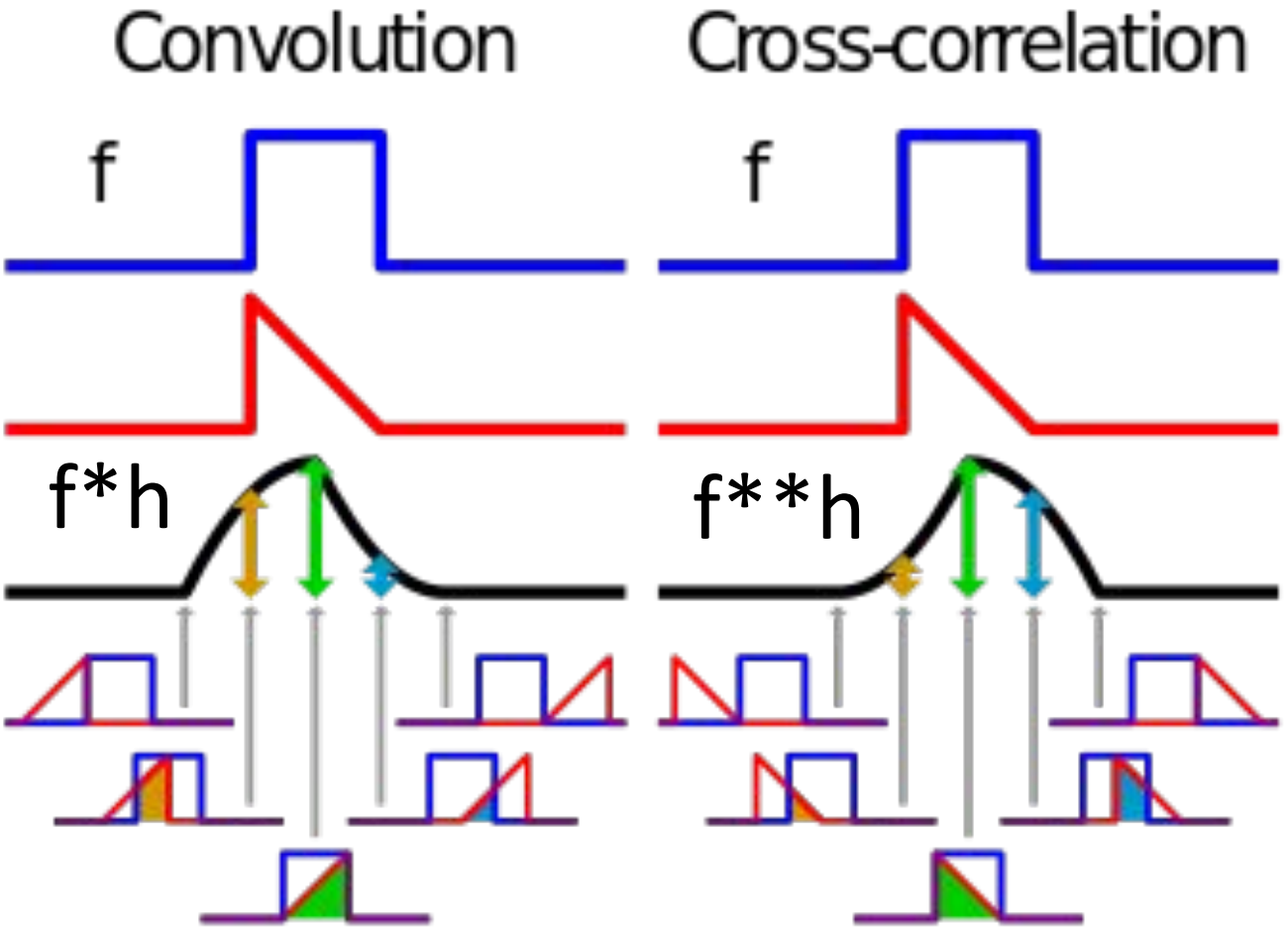
(Cross) correlation – example



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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution vs. (Cross) Correlation



Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Cross Correlation Application: Vision system for TV remote control

- uses template matching

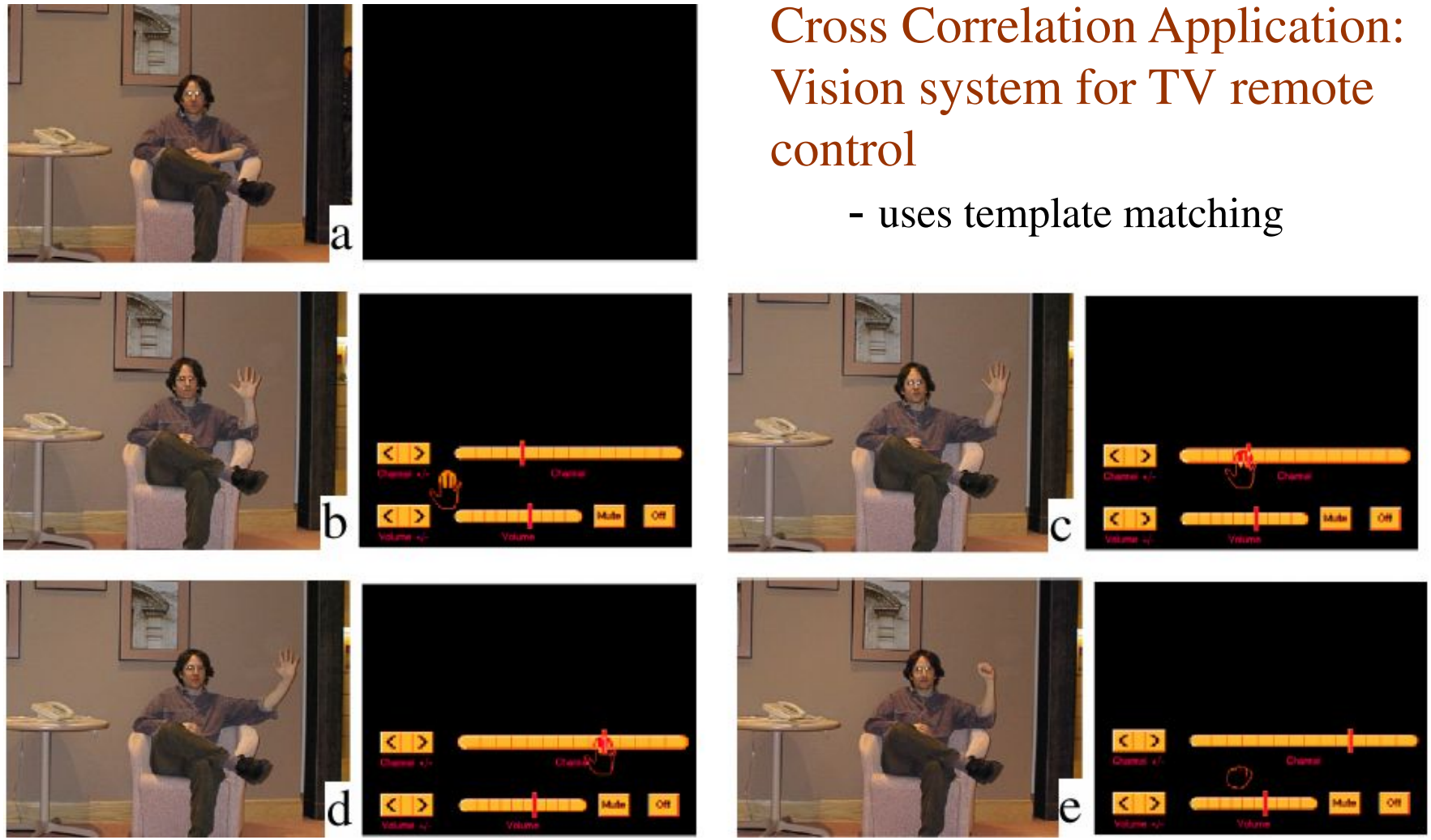


Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Convolution vs. (Cross) 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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Two operations are very similar, why convolution?

Convolution is correlation with pre-flipping of h.
Seems more cumbersome, why bother?

Two operations are very similar, why convolution?

Convolution is correlation with pre-flipping of h.

Seems more cumbersome, why bother?

- Correlation (unlike conv) is:
  - NOT commutative

    (eg. [1 2 3] ** [0 1 0] ≠ [0 1 0] ** [1 2 3])
  - NOT associative: a**(b**c) ≠ (a**b)**c (in general)

Therefore, convolution is much more convenient for mathematical construction + close relation to Fourier transform.

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

What we have learned today?

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

Adapted from slides by Juan Carlos Niebles, and Ranjay Krishna

Optional reading

Sections 3.1 and 3.2 of Richard Szeliski's excellent book (https://szeliski.org/Book/).