Computer Vision I

Image Processing

Roi Santos Mateos

Escola Técnica Superior de Enxeñaría, USC

Academic year 2025/26

MASTER IN ARTIFICIAL INTELLIGENCE

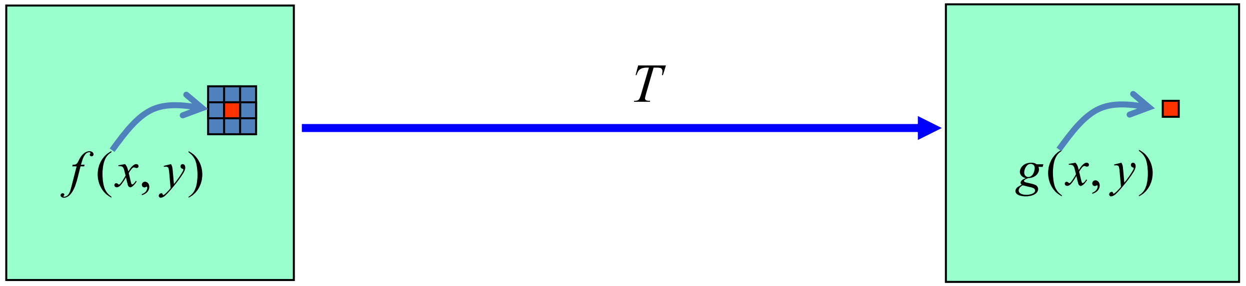
Image Processing

- Image processing:
  - It is a type of signal processing in which input is an image and output may be an image or a tensor of features associated with that image:
    - Extracts useful information from images,
      - Features (edges, corners, blobs…)
    - Improves image visual appearance
      - Enhancing, in-painting, de-noising
    - Transforms image representation into another domain where image properties can be modified/enhanced easily
      - Discrete Fourier Transform, …

Pixel-Level Image Processing

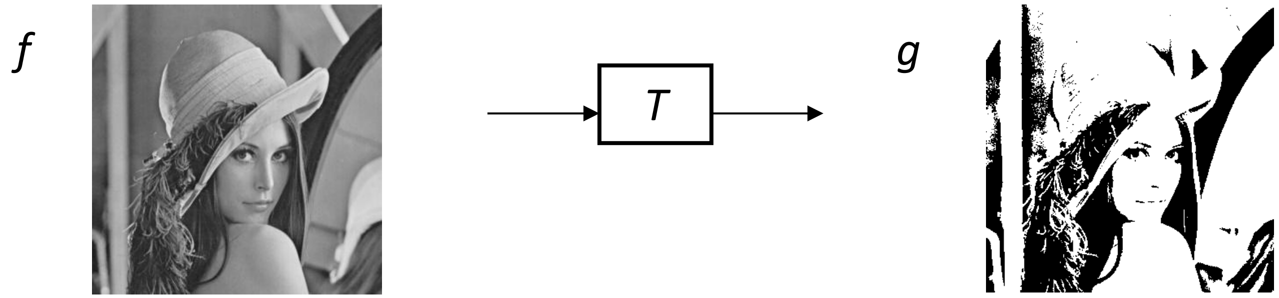
- Generate a new image taking into account only pixels values at the same location in the orignal image:

Processed image $\rightarrow$ $$g(x,y) = T[f(x,y)]$$ $\leftarrow$ Original image

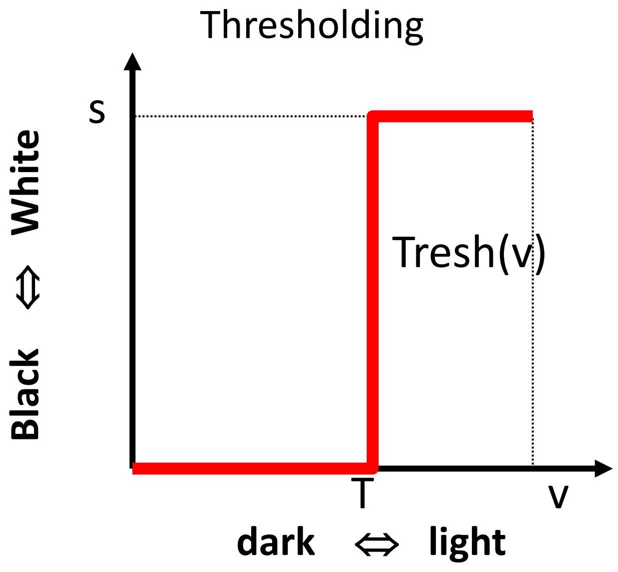


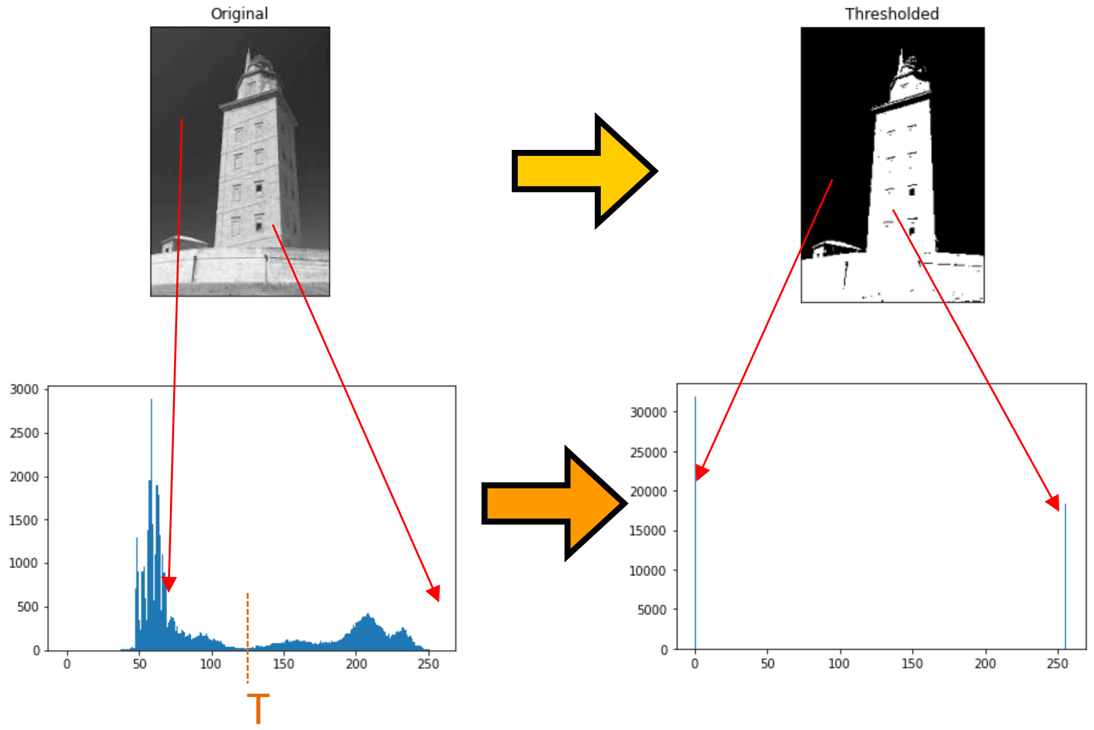
Original image

Processed image



- Thresholding:
  - ▷ Separates foreground objects from background.
  - ▷ Histogram ridges are related to big objects.
  - ▷ Ridge height related to size.

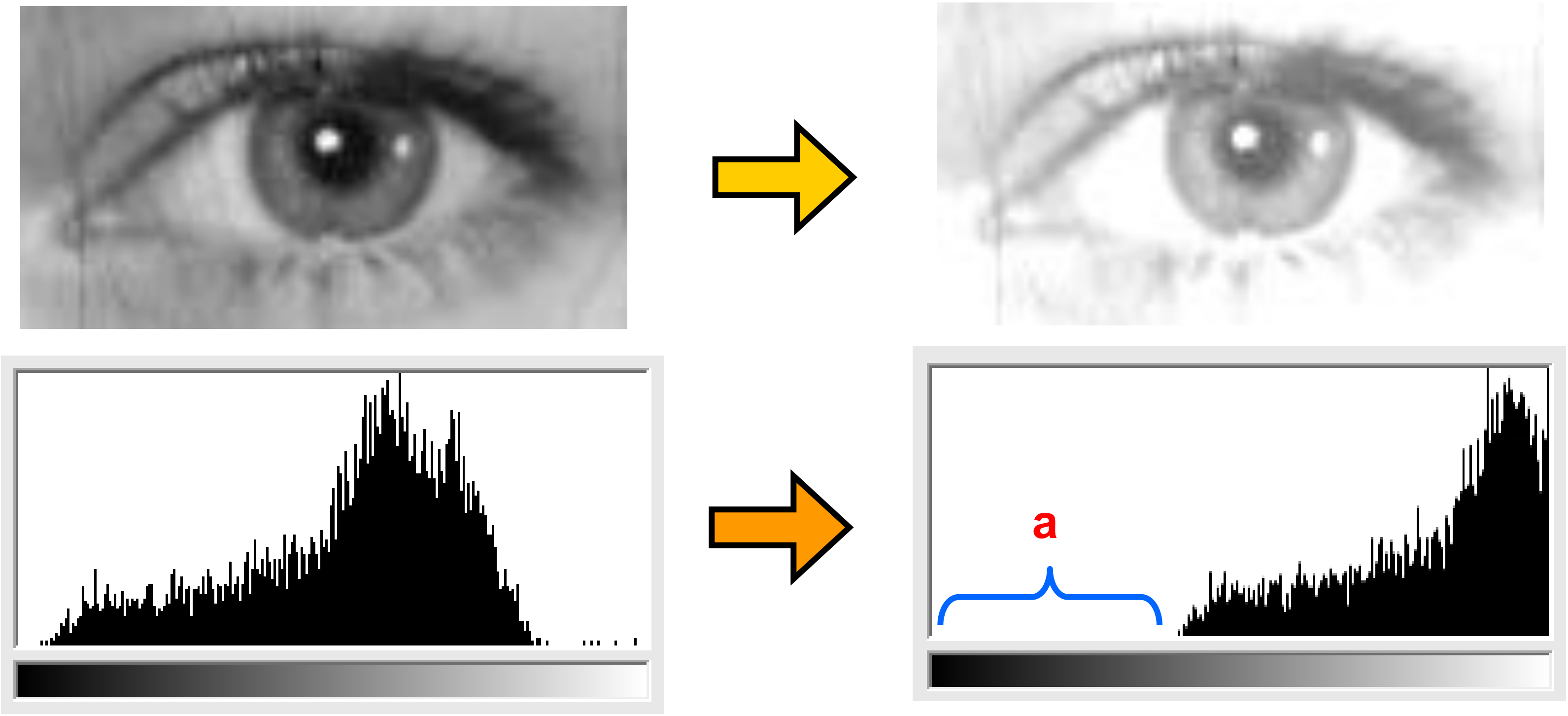




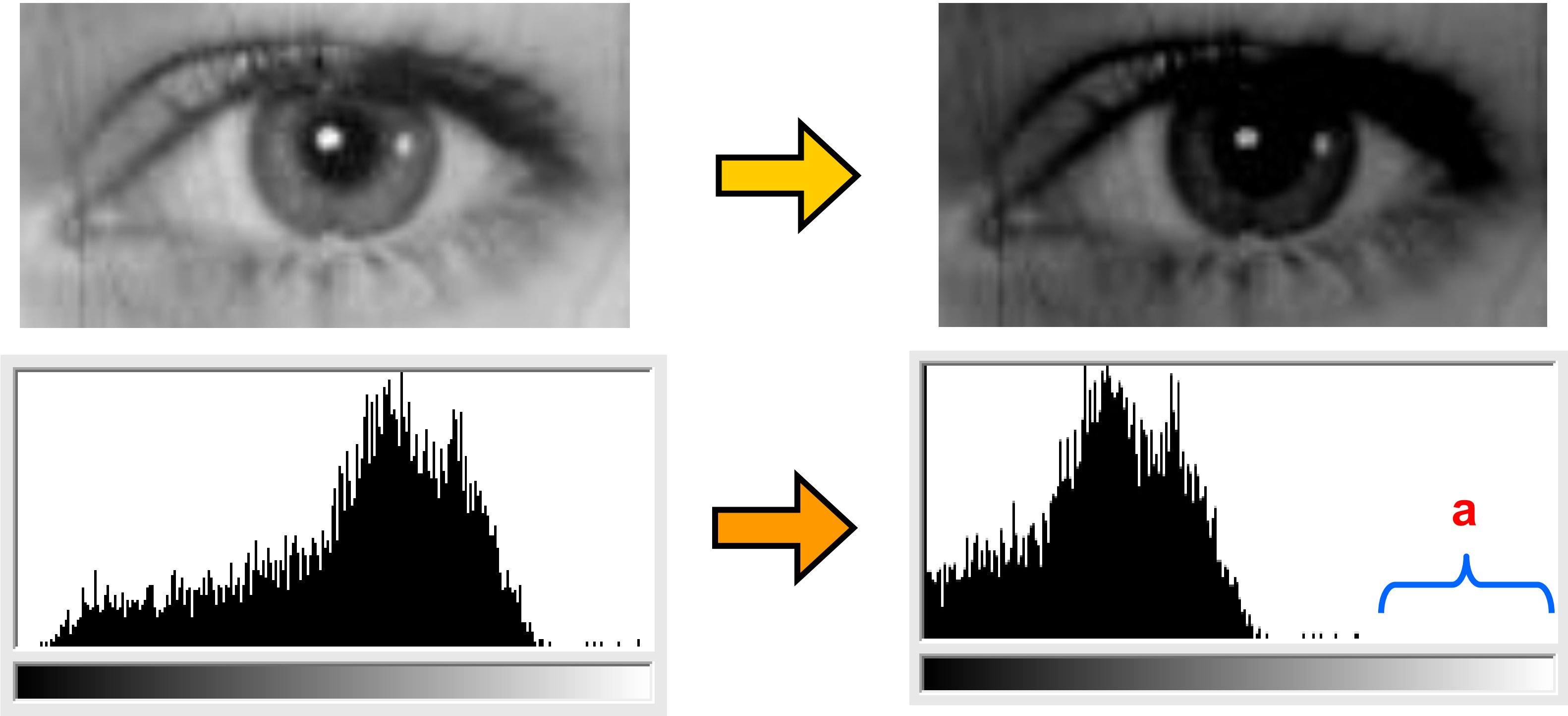
Histogram-Based Transformations

Arithmetic Transformation

- Adding a constant: g(x, y) = f(x, y) + a
  - Makes image brighter
  - Histogram "moves" to the right a pixels.



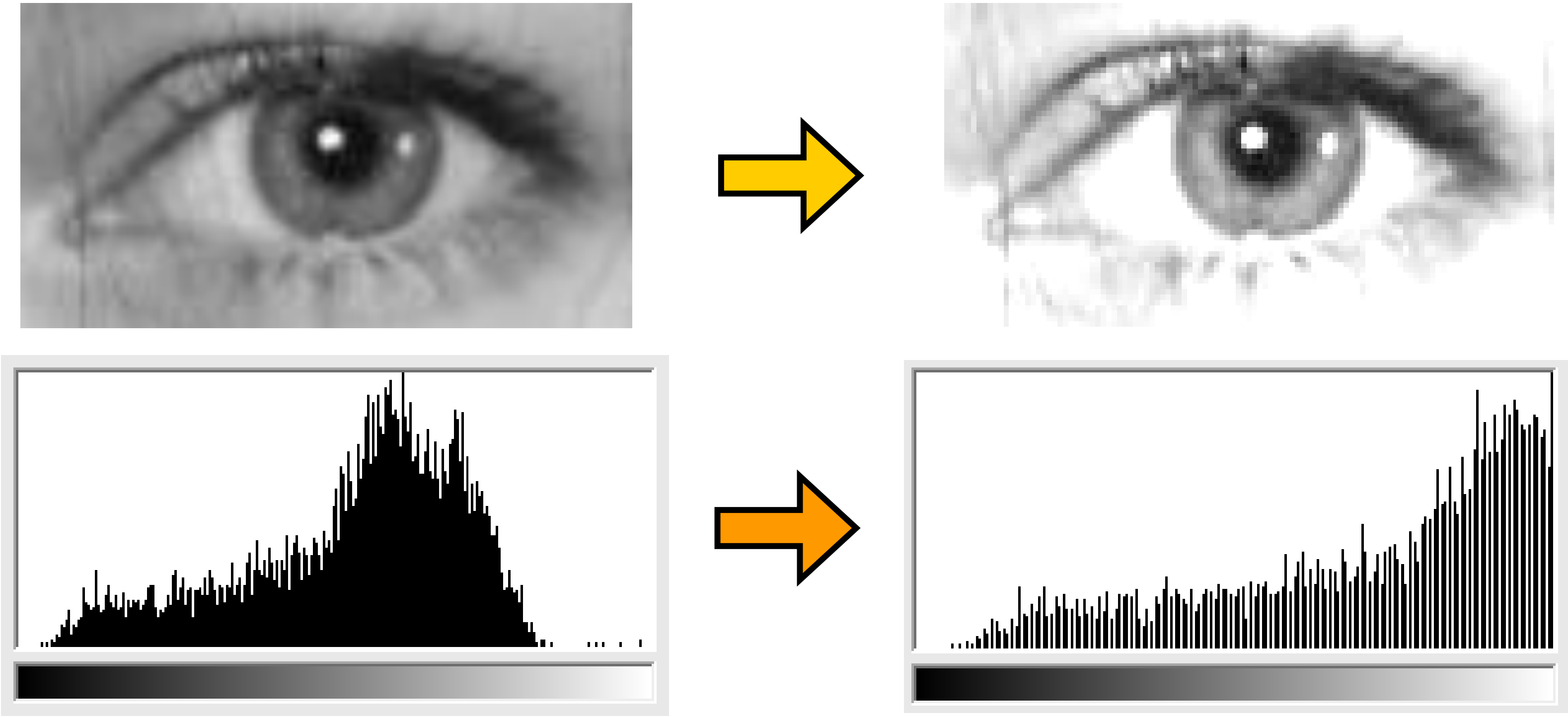
- Substracting a constant: $g(x, y) = f(x, y) - a$
  - Makes image darker
  - Histogram "moves" to the left a pixels.



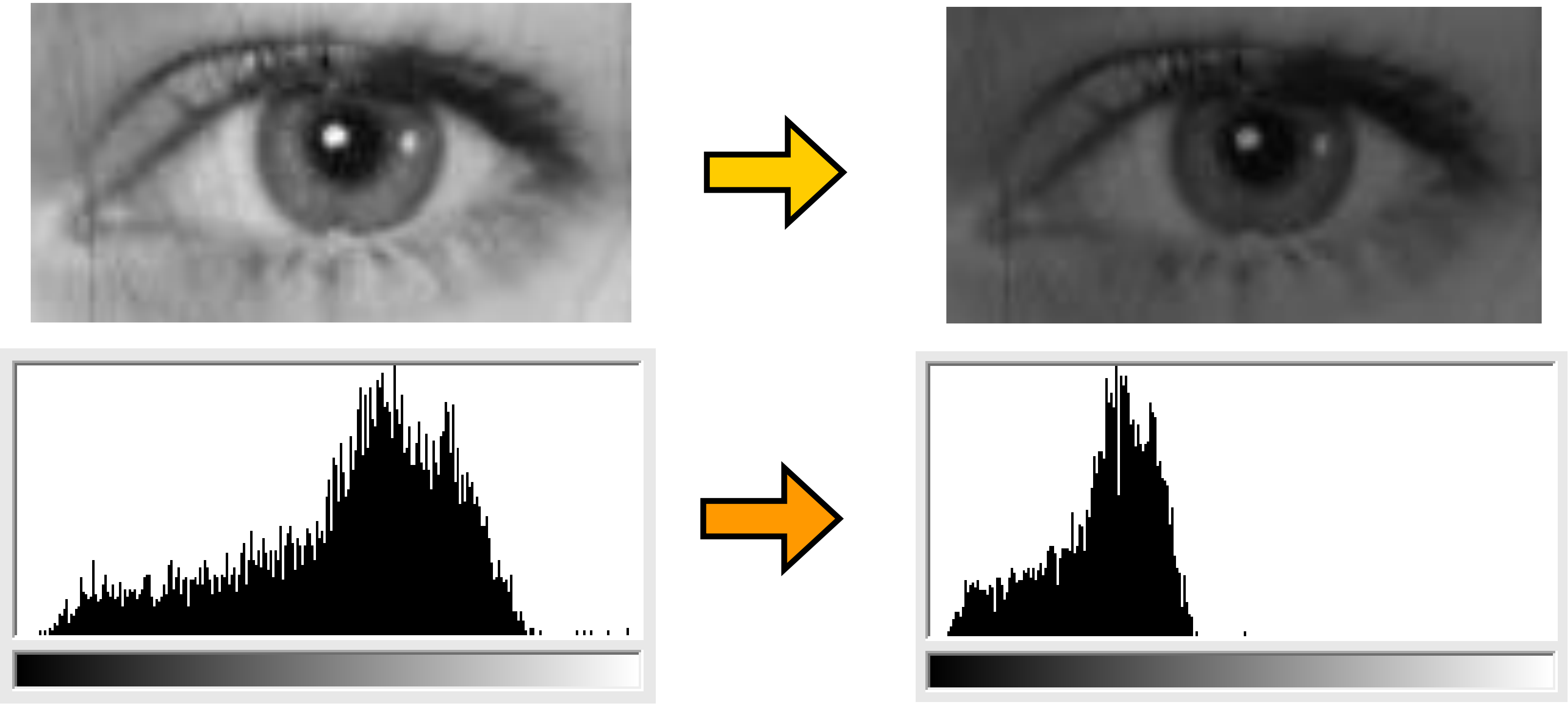
Histogram-Based Transformations

Arithmetic Transformation

- Multiplication by a constant: $g(x, y) = b \cdot f(x, y)$
  - Histogram “stretches” rightwards.

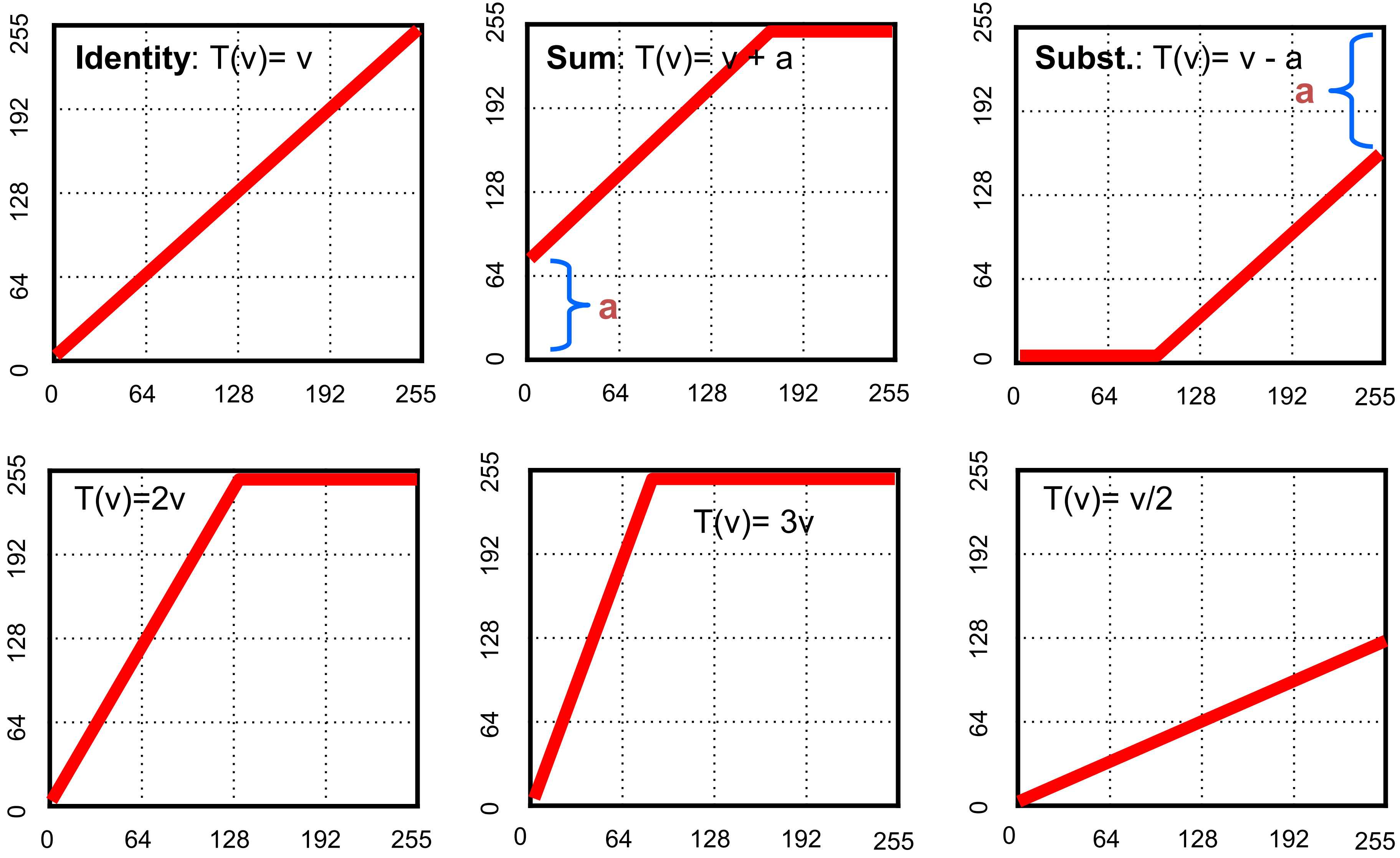


- Division by a constant: g(x, y) = f(x, y) / b
  - Histogram “shrinks”.



Histogram-Based Transformations

Histogram Transformations

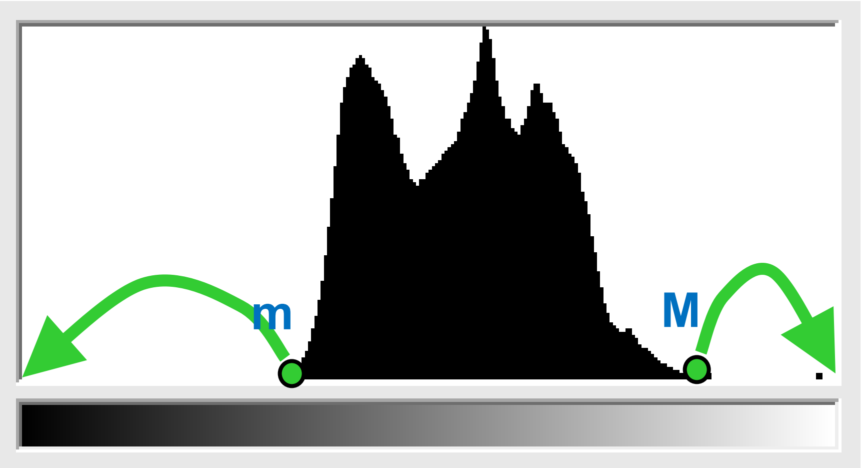


Histogram-Based Transformations

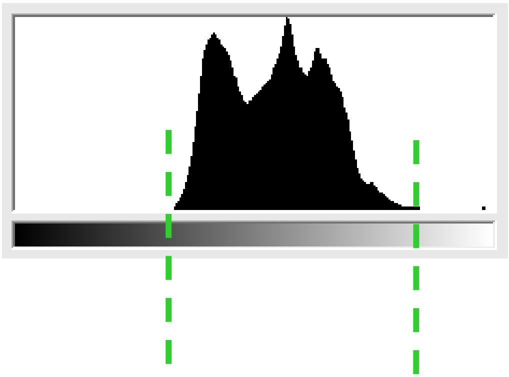
Image Enhancement

- Usually, it is interesting to strech the histogram to enhance images
- Enhancement: defines a linear transformation of the histogram:
  - Finds the minimum gray-level value: **m**
  - Finds the maximum: **M**
  - **Performs an arithmetic transformation based on pixel intensity.**

$$T(v) = (v-m)*255/(M-m)$$

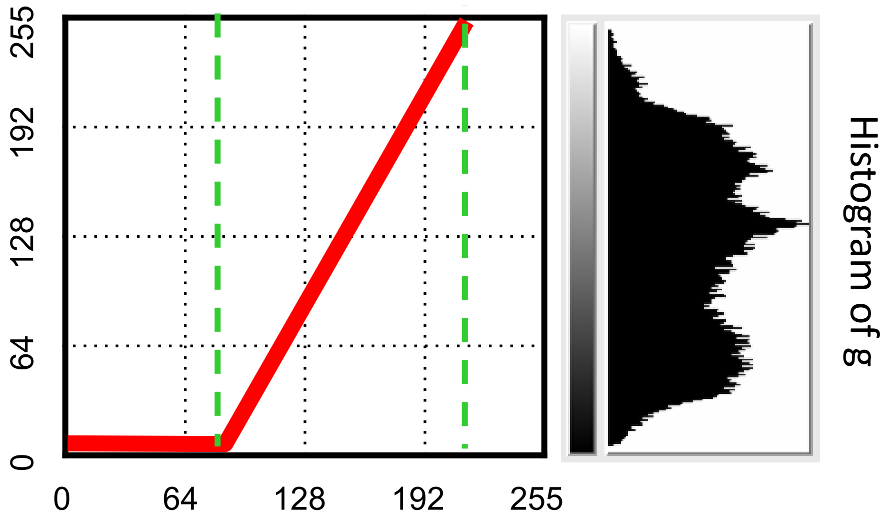


- Example: m= 86, M= 214

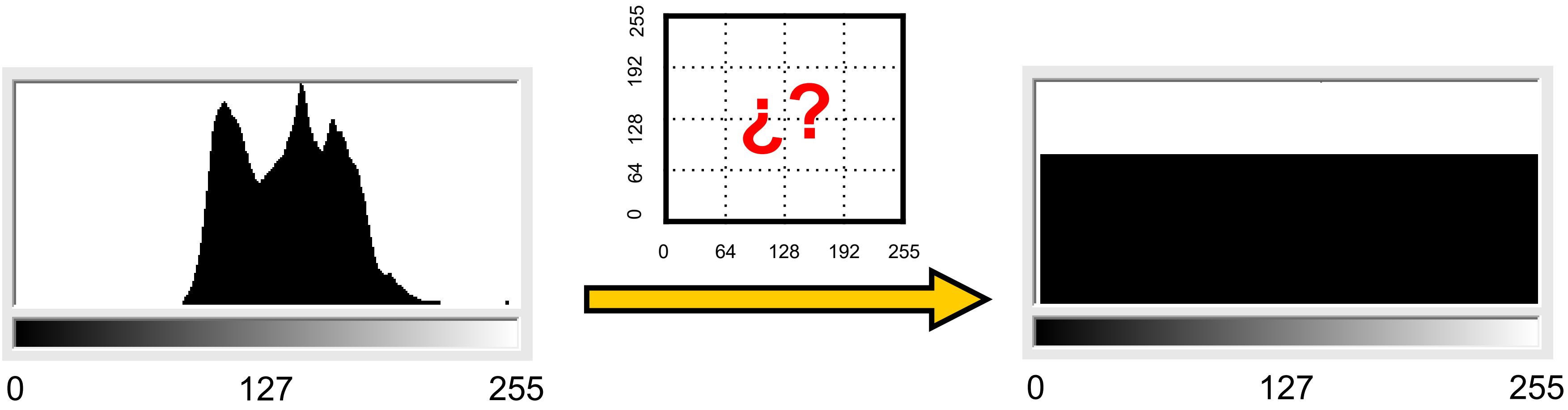


g(x,y) = (f(x,y)-86)*1.99

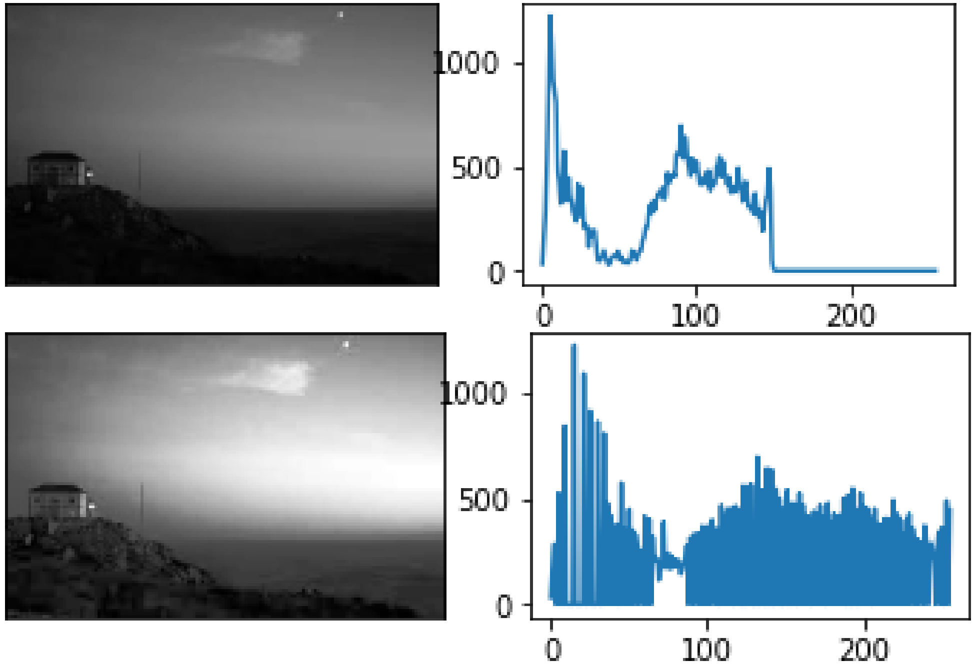
T(v) = (v-86)*255/(214-86)



- Equalization seeks to transform an image histogram in an (as much as posible) uniform one:
  - Goal: increasing contrast resolution in the output image



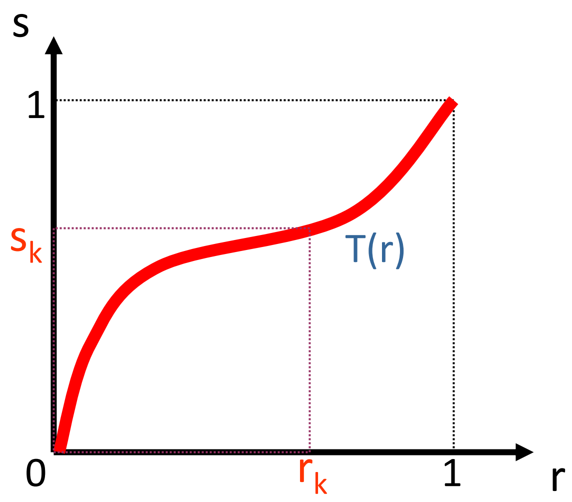
- Equalization steps:
  - Compute histogram: gray-level Probability Distribution Function (PDF)
  - Divide the gray-level range in M intervals of the same probability (same #pixels in all bins)
    - Intervals around ridges are narrower than intervales around valleys
  - Split the histogram in M uniform bins and map original values (within each interval) in the new value in the corresponing bin (stretch or shrink)
  - Transformed histogram will have M intervals of gray-levels with same Cumulative Distribution Function (CDF)



- Fundamentals in continuous domain:
  - Let r be a continuous variable that represents the intensity-levels of a normalized image in [0,1].
  - A transformation $s = T(r)$ is sough such that:
    1. It preserves intensity-level ordering :

       T(r): monovalued and increasing monotone in [0,1]
    2. $0 \leq T(r) \leq 1, \ \forall\ 0 \leq r \leq 1$

This implies that stretching in some parts has to be done at the expenses of shrinking in others!



- Transformation that fulfills conditions 1 and 2:

$$\mathbf{s} = \mathbf{T}(\mathbf{r}) = \int_0^{\mathbf{r}} \mathbf{p_r}(\mathbf{w})\mathbf{dw}, \qquad 0 \le \mathbf{r} \le 1$$

Cumulative Distribution Function (CDF)

- In discrete domain:

$$s_k = T(r_k) = \sum_{j=0}^{k} h_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}$$

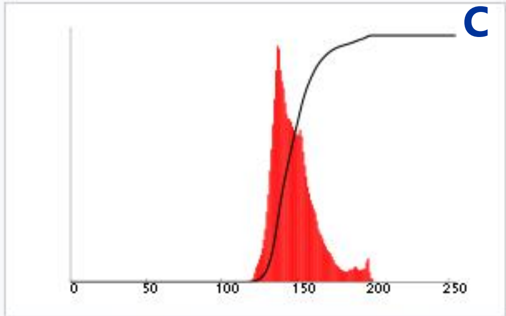
Cumulative Histogram

Histogram-Based Transformations

Equalization

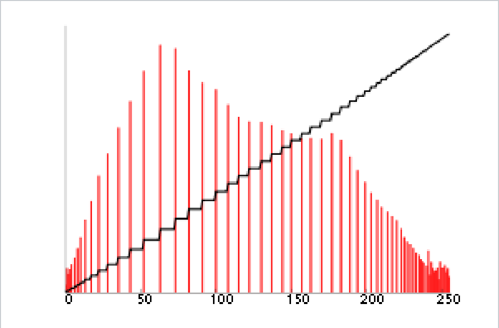
- Example of discrete equalization:

Before Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

After Histogram Equalization



Corresponding histogram (red) and cumulative histogram (black)

Equalization redistributes gray-level bins such that all the intervals have similar number of pixels in their bins:

0.- M=Highest gray-scale value

1.- C= normalized cumulative histogram.

2.- m_c = min(C)

3.- $s_r = T(r) = M*(c_r - m_c)/(1 - m_c)$

To take advantage of all the gray scale range when $m_c > 0$

- Combination of images:
  - Input: images A and B.
  - Output: image R.

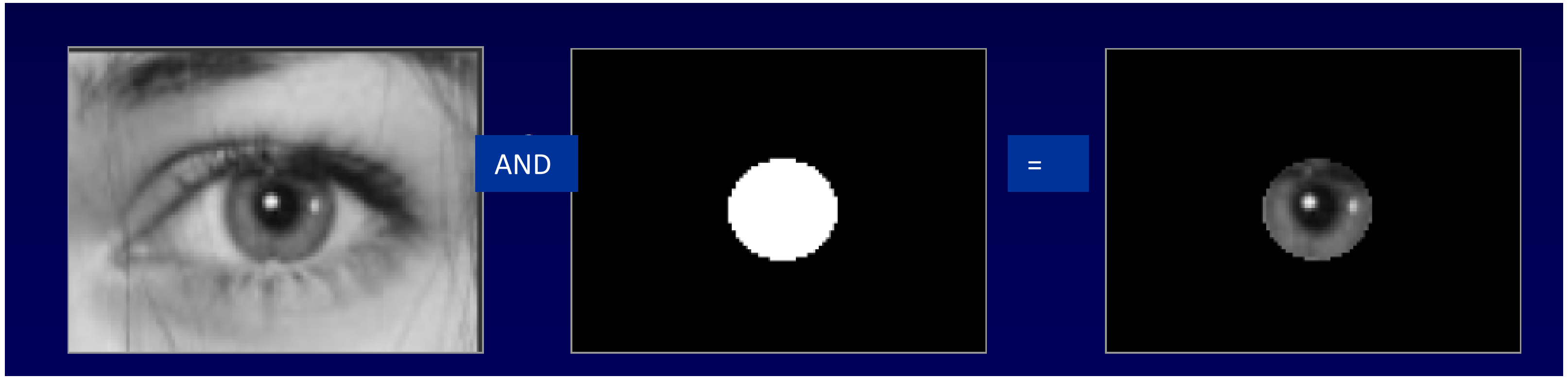
$$R(x, y) = f(A(x,y), B(x,y))$$

- Multiple operators:
  - Boolean: and, or, xor, not
  - Arithmetic: sum, substraction, multiplication
  - Relational: max, min

Boolean Operations

- Boolean bitwise operators:
  - R(x, y) = A(x,y) AND B(x,y)
  - R(x, y) = A(x,y) OR B(x,y)
  - R(x, y) = A(x,y) XOR B(x,y)
  - R(x, y) = NOT A(x,y) AND B(x,y)
  - R(x, y) = NOT A(x,y) OR B(x,y) …

- They are usually used when at least A is binary.



Arithmetic Operations

- Image substraction: motion detection

- Mean: noise reduction:

Local-Level Image Processing

- Capabilities of point operations are limited
- Local filtering: pixel transformation also depends on its neighborhood
  - Dependency is expressed by means of a mask

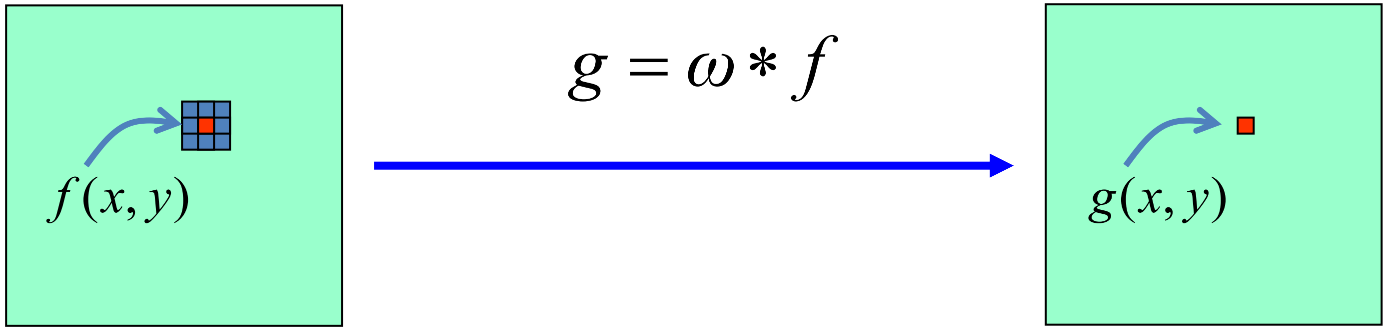
w(-1,-1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)
w(1,-1)	w(1,0)	w(1,1)

Mask coefficients showing coordinate arrangement, around transformed pixel coordinates (0,0)

- Convolution:

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)f(x-s,y-t)$$

$$g=\omega * f$$



Input Image, ***f***

Output Image, ***g***

- Local-Level Image Processing
  - Convolution

$$\begin{bmatrix} 2 & 2 & 2 & 3 \\ 2 & 1 & 3 & 3 \\ 2 & 2 & 1 & 2 \\ 1 & 3 & 2 & 2 \end{bmatrix} * \begin{bmatrix} 1 & -1 & -1 \\ 1 & 2 & -1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 4 & 4 & -2 \\ 9 & 6 & 14 & 5 \\ 11 & 7 & 6 & 5 \\ 9 & 12 & 8 & 5 \end{bmatrix}$$

Input Image, *f* **Convolution kernel, *ω*** **Output Image, *g***

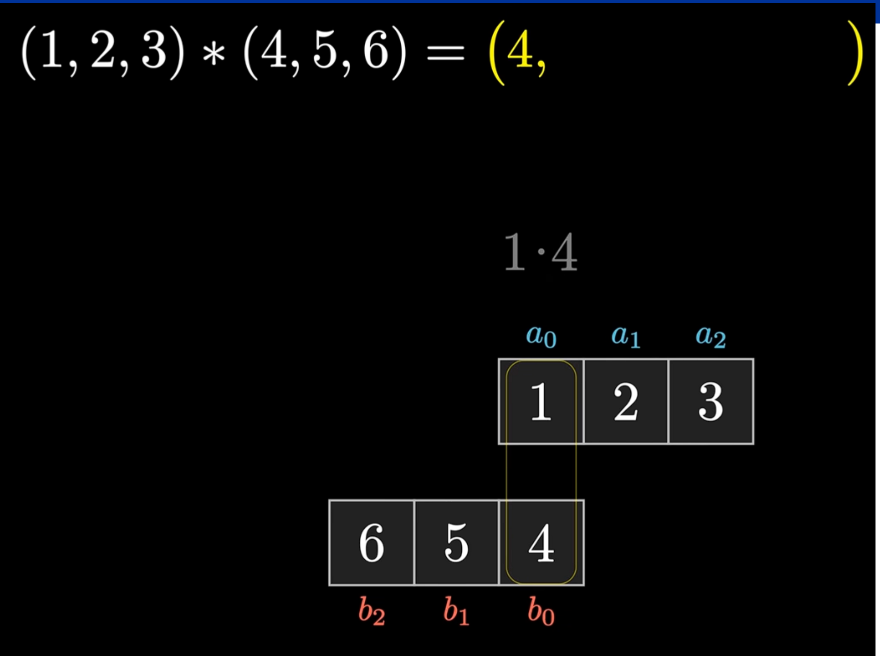
Equivalently: kernel rotation + kernel sliding + summation of the multiplication of overlapped image and kernel values.

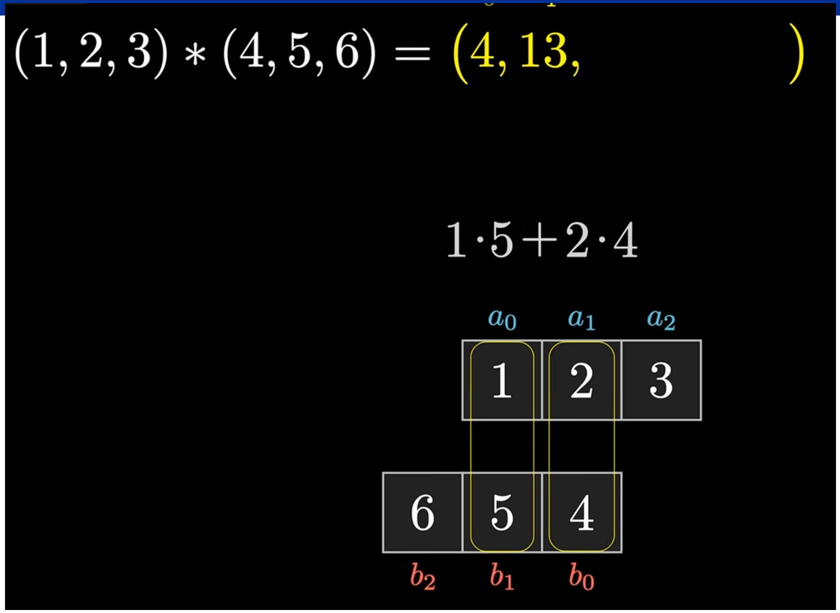
  - First step: kernel rotation

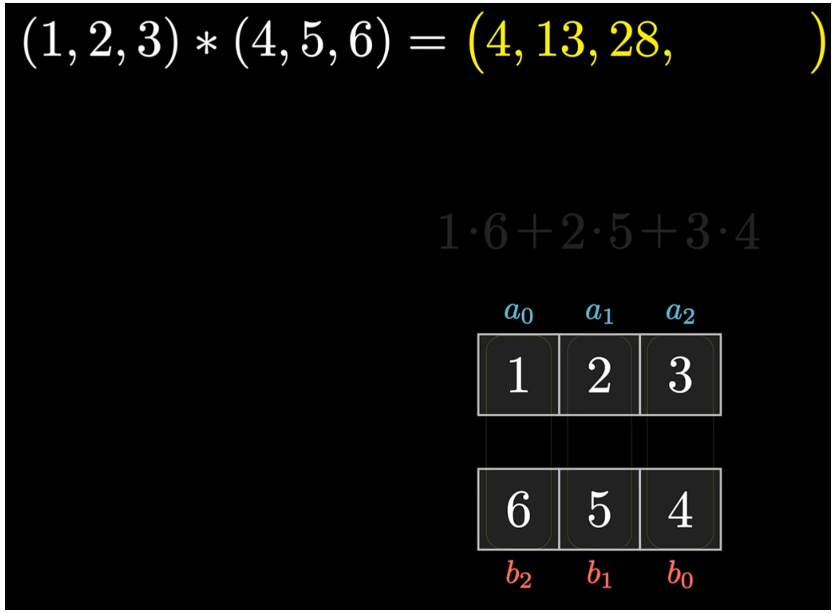
$$\begin{bmatrix} 1 & -1 & -1 \\ 1 & 2 & -1 \\ 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{Rotate } 180^o} \begin{bmatrix} 1 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

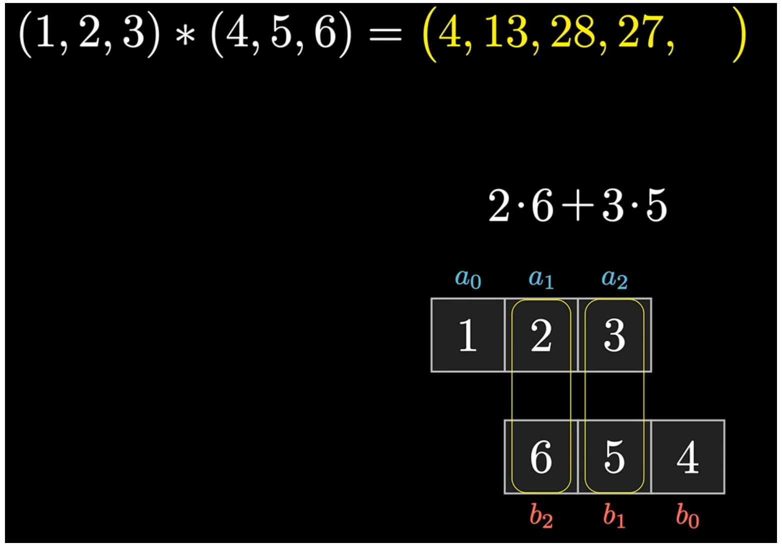
Local-Level Image Processing

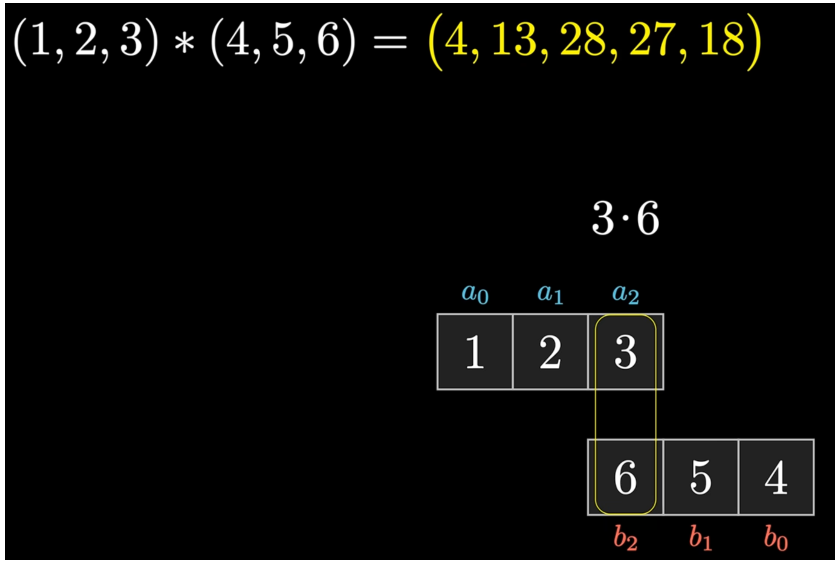
2D Convolution

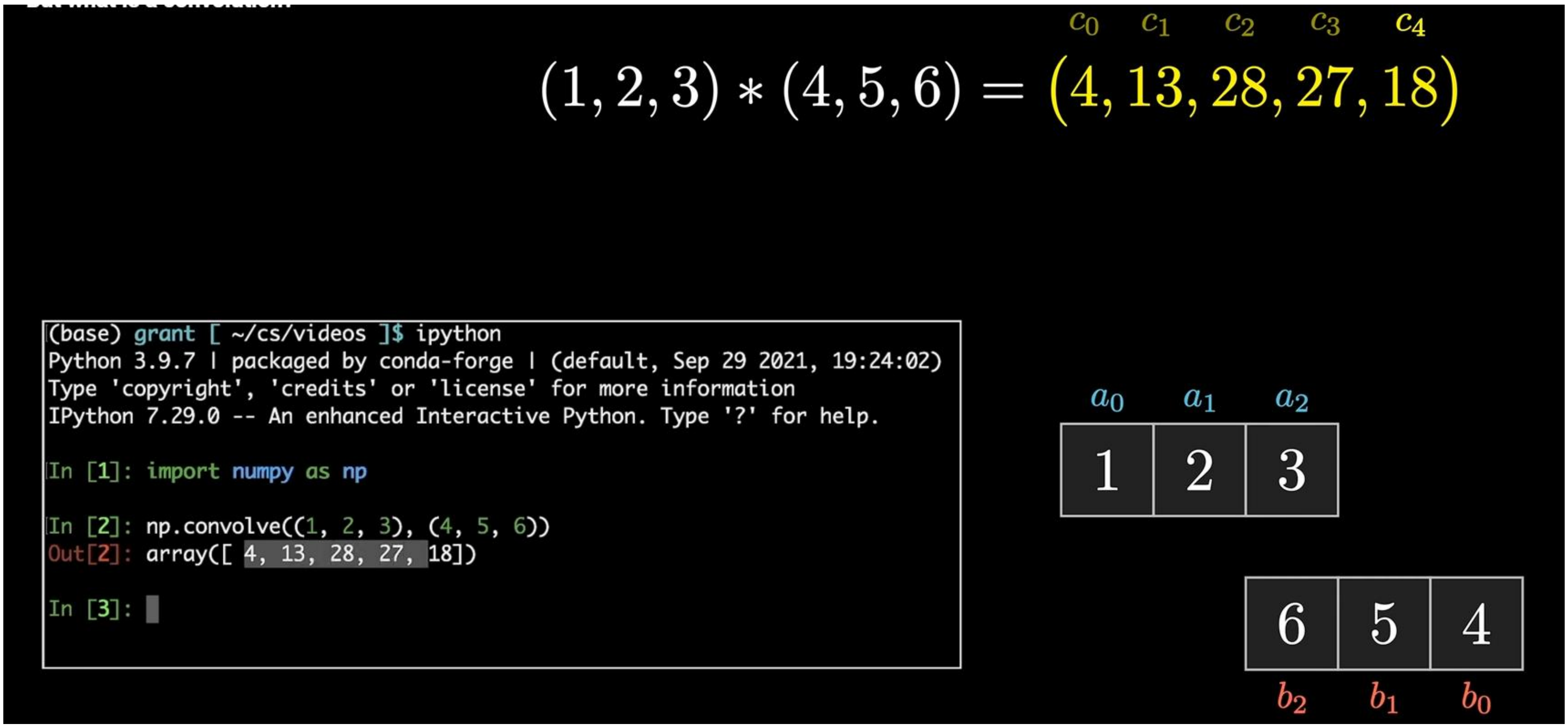












c_0 c_1 c_2 c_3 c_4
$(1,2,3) * (4,5,6) = (4,13,28,27,18)$
(base) grant [~/cs/videos]$ ipython
Python 3.9.7 | packaged by conda-forge | (default, Sep 29 2021, 19:24:02)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.29.0 -- An enhanced Interactive Python. Type '?' for help.
In [1]: import numpy as np
In [2]: np.convolve((1, 2, 3), (4, 5, 6))
Out[2]: array([4, 13, 28, 27, 18])
In [3]:
a_0 a_1 a_2
1 2 3
6 5 4
b_2 b_1 b_0

Local-Level Image Processing

2D Convolution

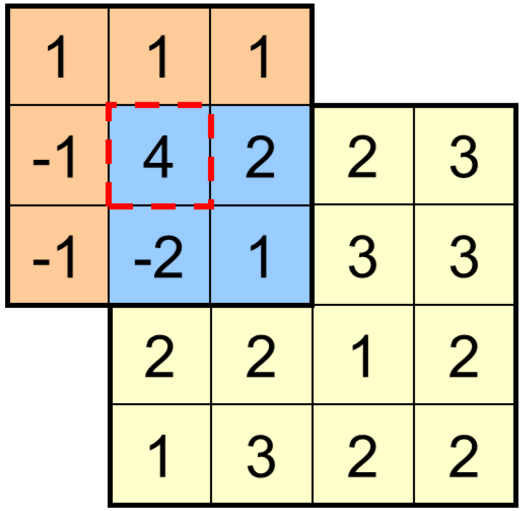
- Second step:
  - kernel sliding
  - summation of the multiplications of overlapped image and kernel values

Convolution kernel, *ω*

1	1	1
-1	2	1
-1	-1	1

Input Image, *f*

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



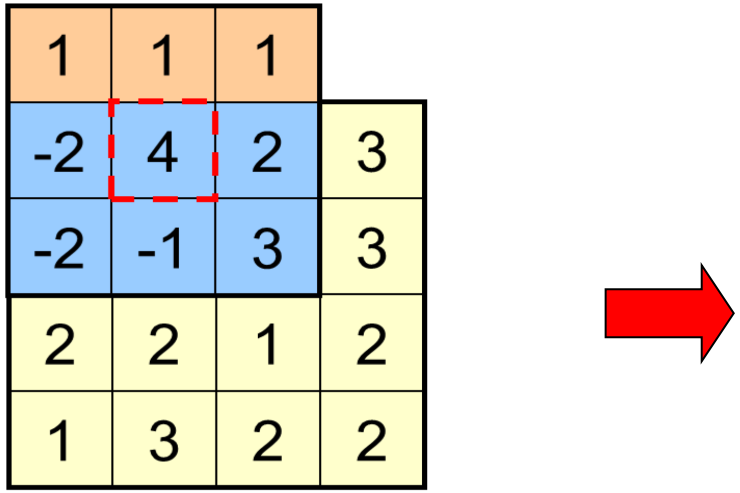
Output Image, *g*

5			

- Next step:
  - kernel sliding
  - summation of the multiplications of overlapped image and kernel values

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Output Image, *g*

5	4		

Local-Level Image Processing

2D Convolution

- Next step:
  - kernel sliding
  - summation of the multiplications of overlapped image and kernel values

1	1	1
-1	2	1
-1	-1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

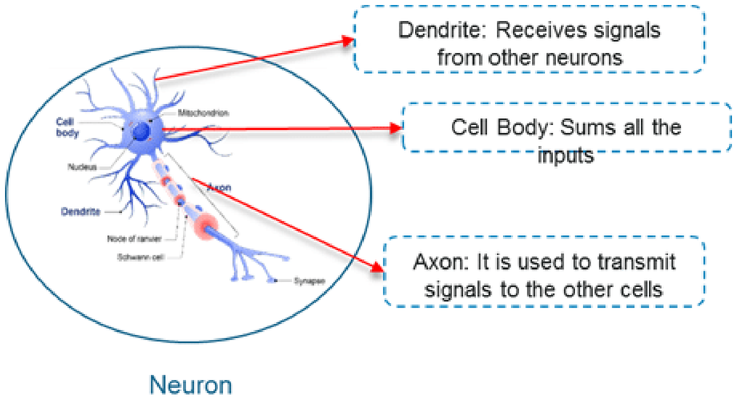
2	2	2	3
-2	2	3	3
-2	-2	1	2
1	3	2	2

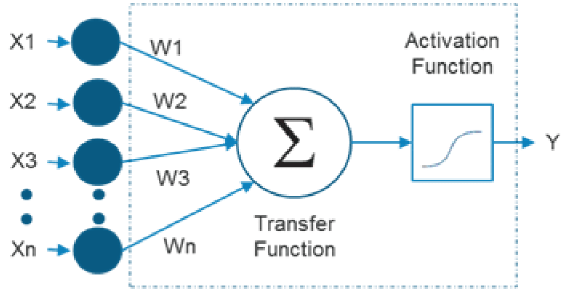
5	4	4	-2
9	6		

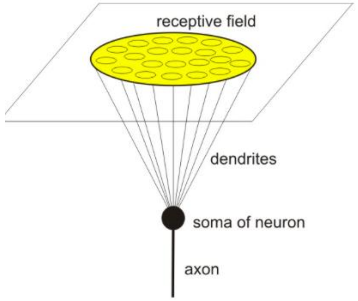
- Final output image g:

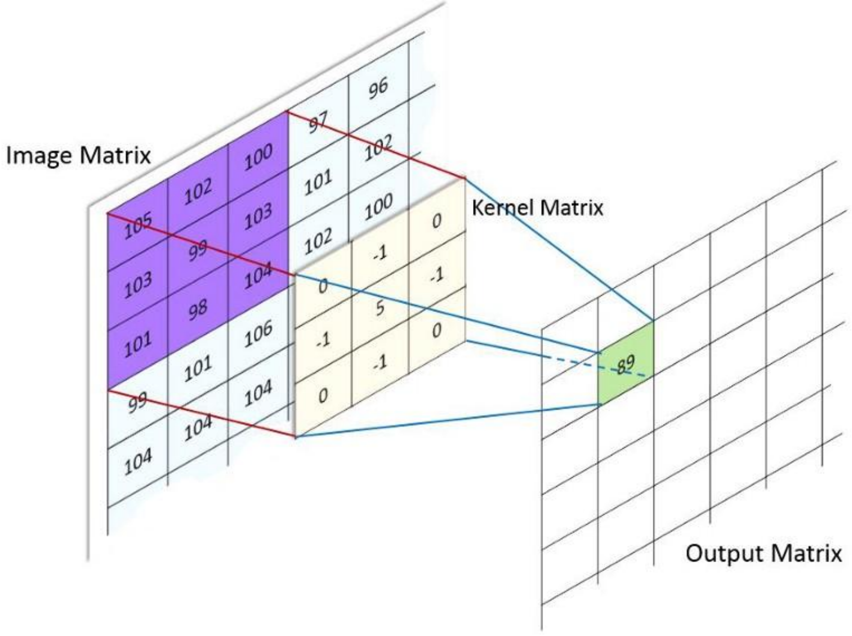
5	4	4	-2
9	6	14	5
11	7	6	5
9	12	8	5

- Convolution: from biological Inspiration to implementation





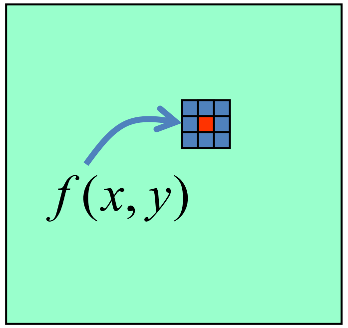




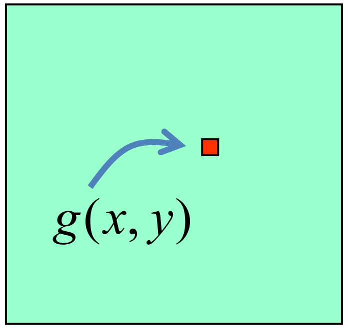
- Correlation

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}\omega(s,t)f(x+s,y+t)$$

$$g=\omega\circ f$$

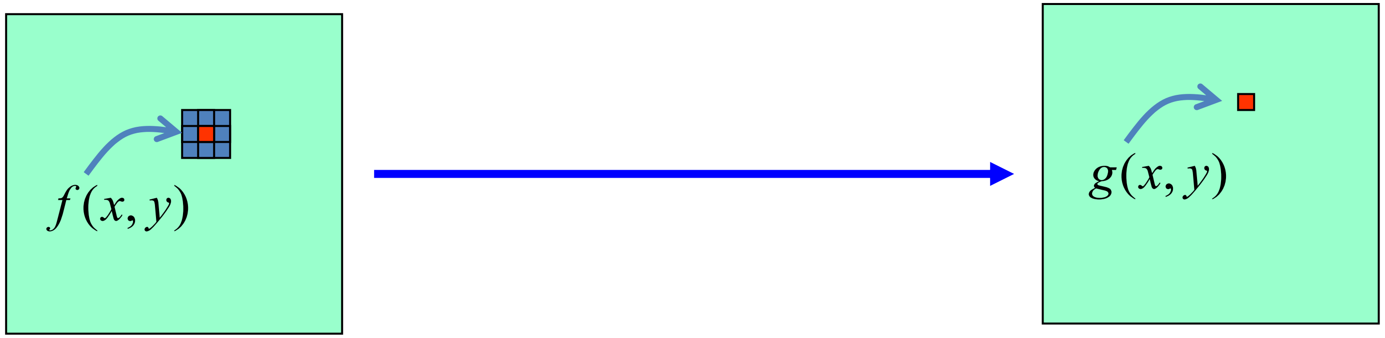


Input Image *f*



Output Image *g*

- Local-Level Image Processing



Input Image f

Output Image g

Equivalently: kernel sliding + summation of the multiplication of overlapped image and kernel values.

Convolution kernel ω

1	-1	-1
1	2	-1
1	1	1

Don't Rotate

Input Image f

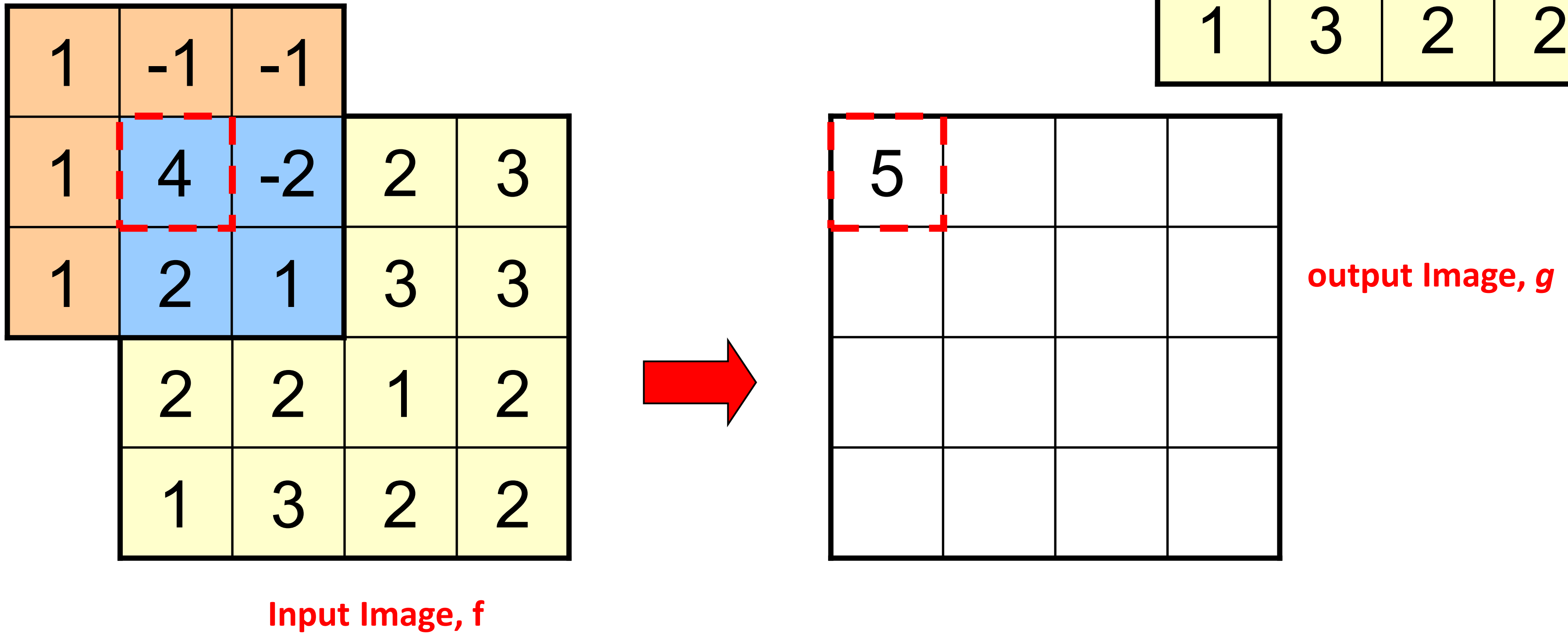
2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Local-Level Image Processing

2D Correlation

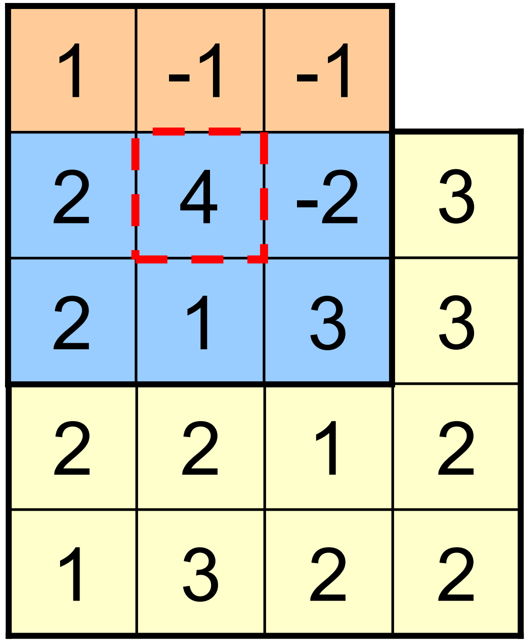
1	-1	-1
1	2	-1
1	1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



1	-1	-1
1	2	-1
1	1	1

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2



Input Image f

5	10		

output Image *g*

- Local-Level Image Processing

1	-1	-1
1	2	-1
1	1	1

Kernel *ω*

o

2	2	2	3
2	1	3	3
2	2	1	2
1	3	2	2

Input Image f

=

5	10	10	15
3	4	6	11
7	11	4	9
-5	4	4	5

Final output Image *g*

Correlation vs Convolution

- They are simple operations, but extremely useful
  - Linear
  - Shift invariant
- They do the same job when kernels are symmetrical
- Key difference: convolution is associative and commutative:
  - That is, if *F* and *G* are filters, then
  - *F*(G*I) = (F*G)*I*

- In general:
  - Convolution is used for image processing operations such as smoothing, edge detection
  - Correlation is used to match a template to an image.

- Correlation: $g(x, y) = \sum_{s,t} \omega\,(s, t)\ \ f(x, +sy + t)$
- Convolution: $g(x, y) = \sum_{s,t} \omega\,(-, s - t)\ \ f(x, +sy + t)$ ←kernel is flipped.
- So $g_{conv} = f * \omega = f \circ \widetilde{\omega}$ with $\widetilde{\omega}(s, t) = \omega(-s, -t)$.

Correlation vs Convolution

- **Convolution** $(f * g)$
  - **Associative:** $(f * g) * h = f * (g * h)$
  - **Commutative:** $f * g = g * f \leftarrow$ (mathematically, for infinite-support signals)
  - **Distributive:** $f * (g + h) = f * g + f * h$
  - Identity: $f * \delta = f$
- **Cross-correlation** $(f \circ g)$
  - **Not commutative** in general: $f \circ g \neq g \circ f$
  - **Not associative** in general
  - Relation to convolution: $f \circ g = f * \tilde{g}$,where $\tilde{g}(x, y) = g(-, x - y)$)and complex-conjugate if needed).
- Special case: if the kernel is **real and symmetric** ($g = \tilde{g}$,(then **correlation = convolution**, so commutativity holds.

Convolution and Correlation

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

- Really important!
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - Deep Convolutional Networks

Convolution

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

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

f

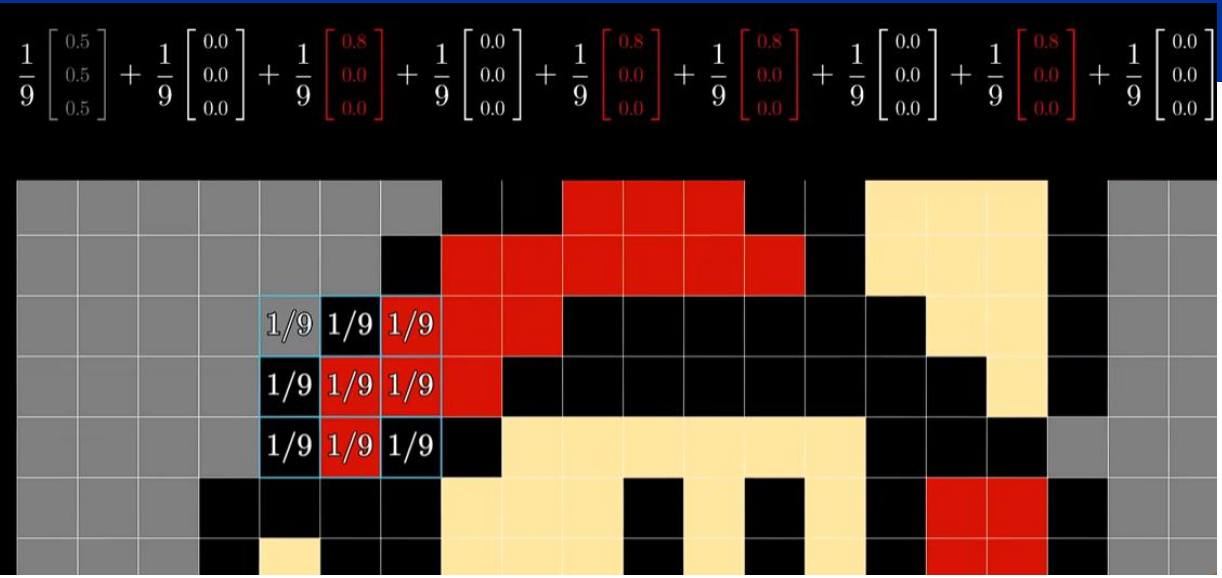
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

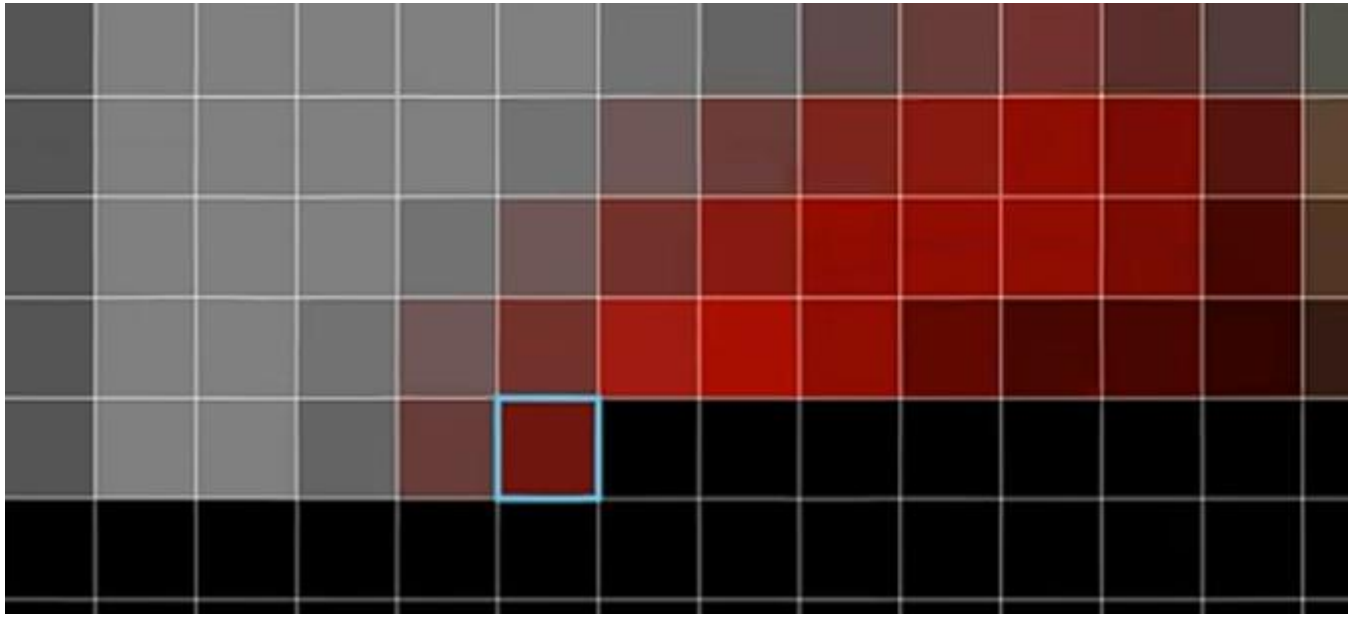
g

	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	

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

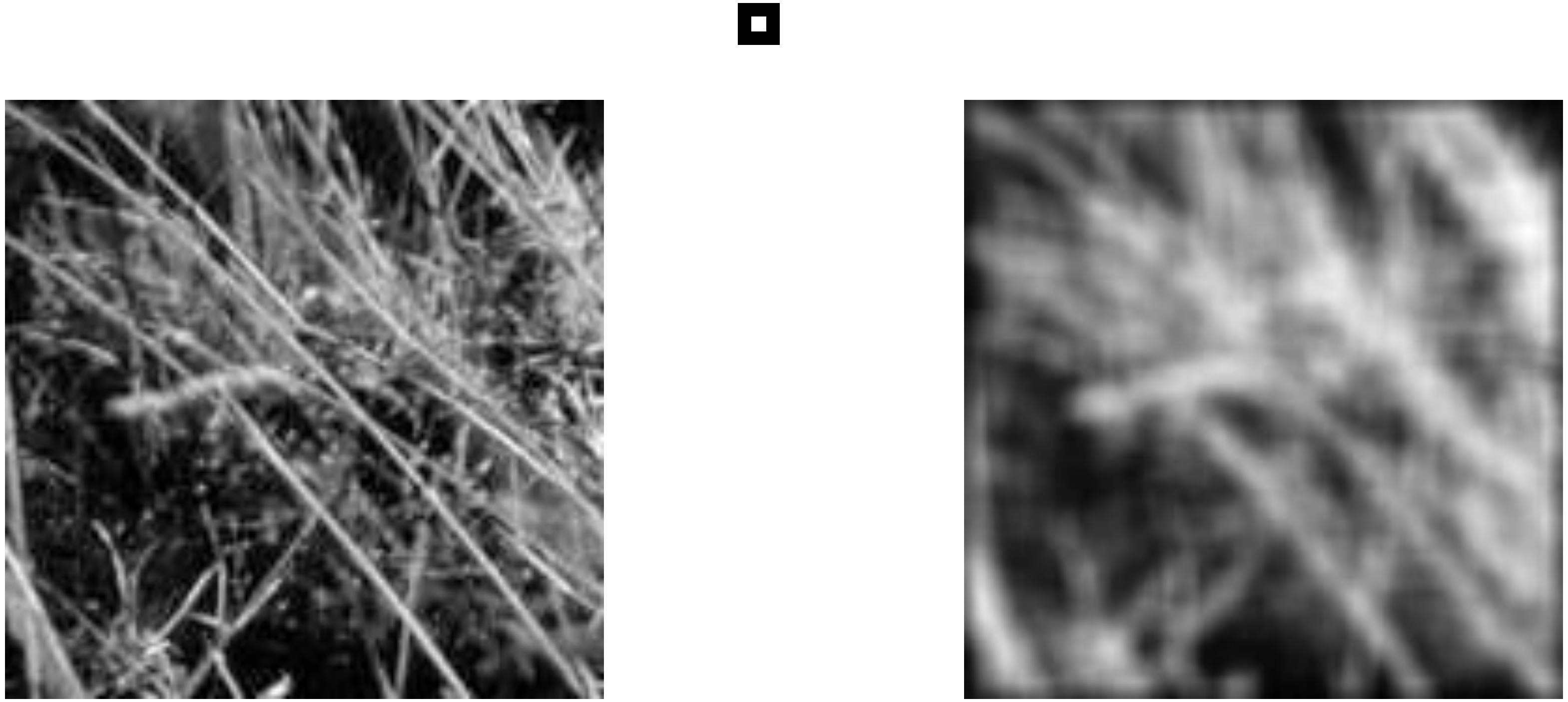
Convolution





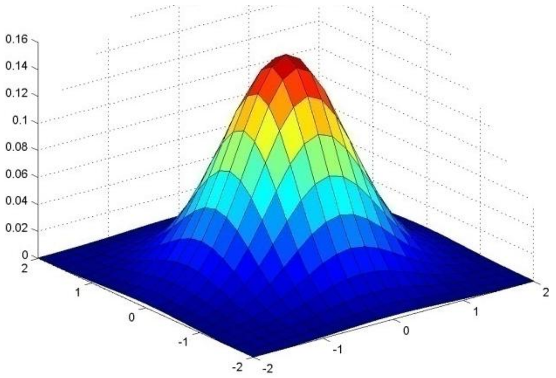
Convolution

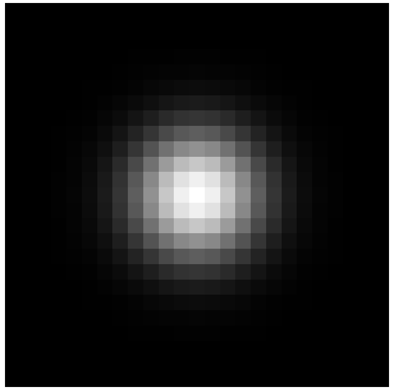
- Example: box filter
  - Replaces each pixel with an average of its neighborhood
    - Achieve smoothing effect (remove sharp features)



Convolution

- Example: Gaussian filter
  - Weight contributions of neighboring pixels by nearness
    - Achieve smoothing effect (remove sharp features)



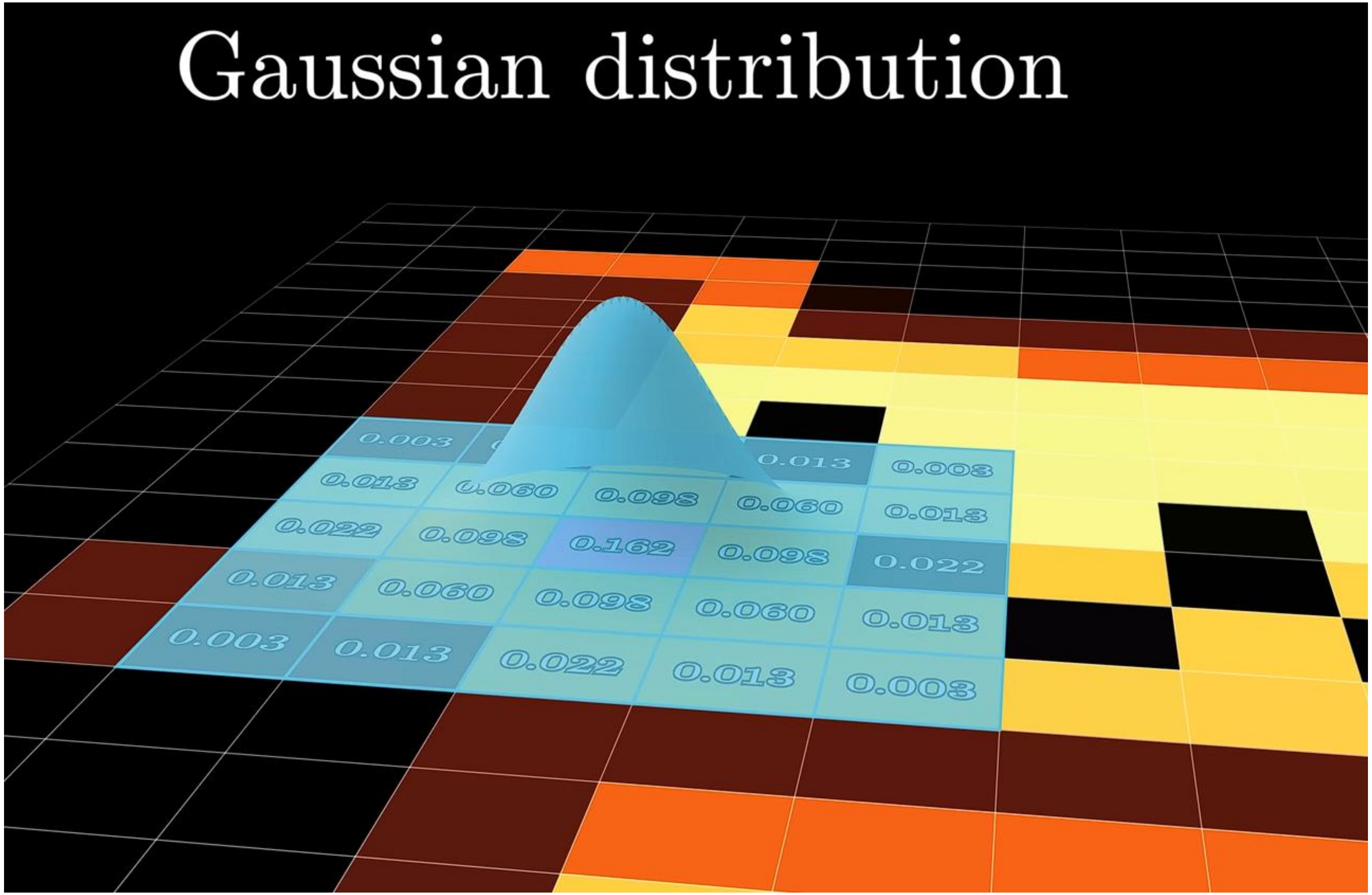


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

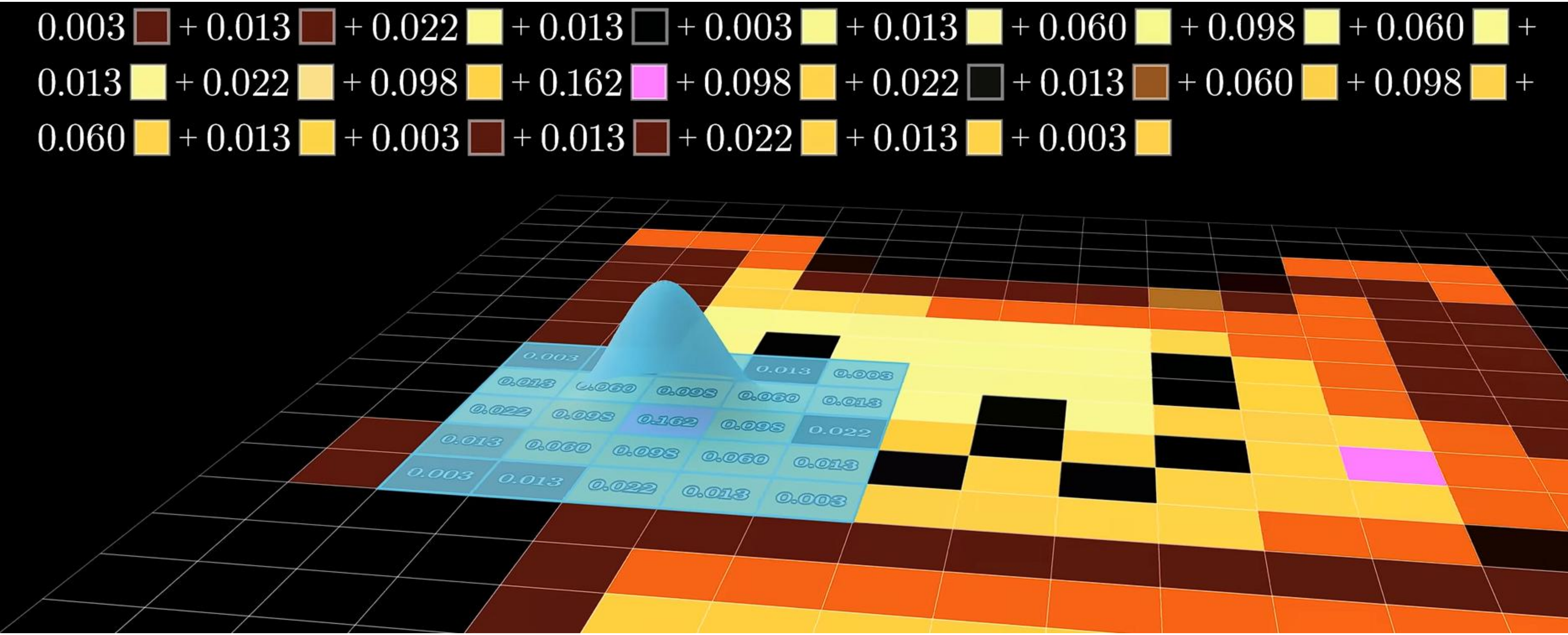
5 x 5, σ = 1

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

Convolution

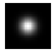


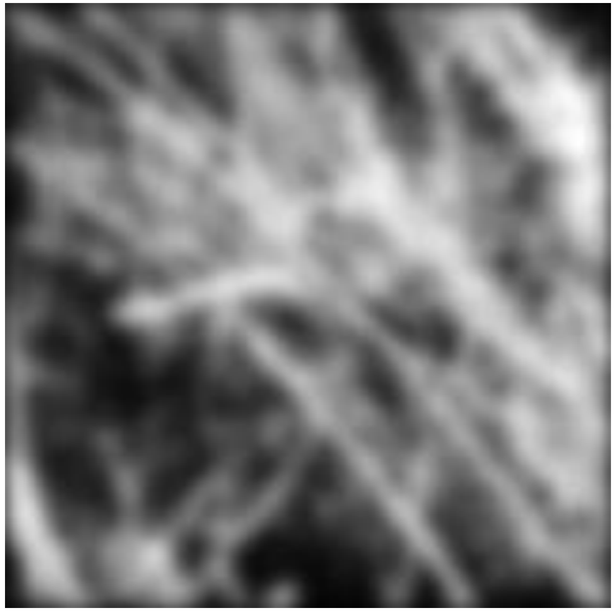
Convolution



Convolution

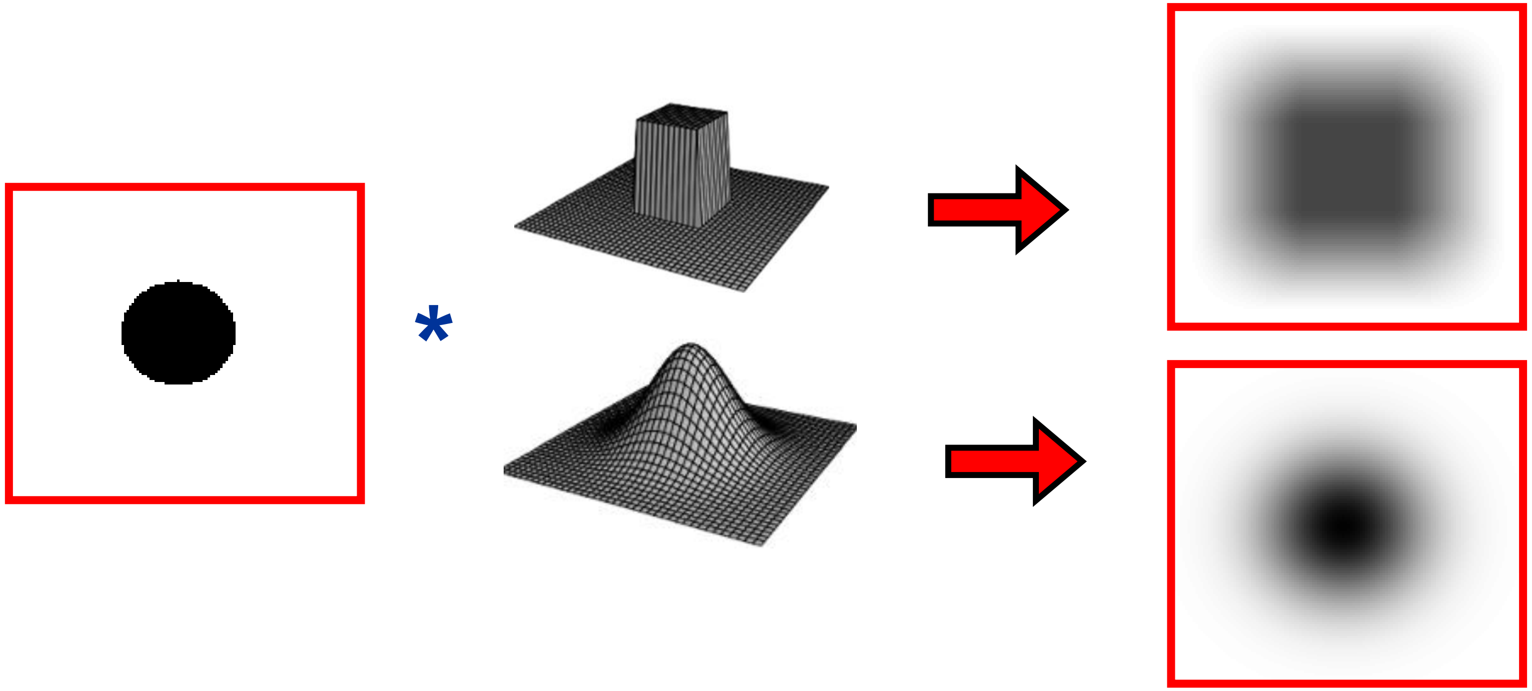
- Example: Gaussian filter
  - Weight contributions of neighboring pixels by nearness
    - Achieve smoothing effect (remove sharp features)





Denoising

- Gaussian average (GA) vs mean filter (MF):
  - GA is isotropic
  - MF smooths further along diagonals than along rows and columns.
  - GA weights decay gradually to zero
  - MF weights have an abrupt cut-off which leaves discontinuities in the smoothed image.



Gaussian Kernel

- Standard deviation σ, parametrizes the smoothing level.
  - Greater values: wider bell, larger kernel size, higher smoothing.
  - Smaller values: narrower bell, lower smoothing.

- Approximation to discrete Gaussian (kernel 1D):
  - Pascal triangle.

$$
nf \bullet \begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1
\end{array}
$$

nf: normalization factor

Gaussian kernel

- Pascal triangle rows correspond to different discrete versions of a Gaussian

1/2 • 1 1

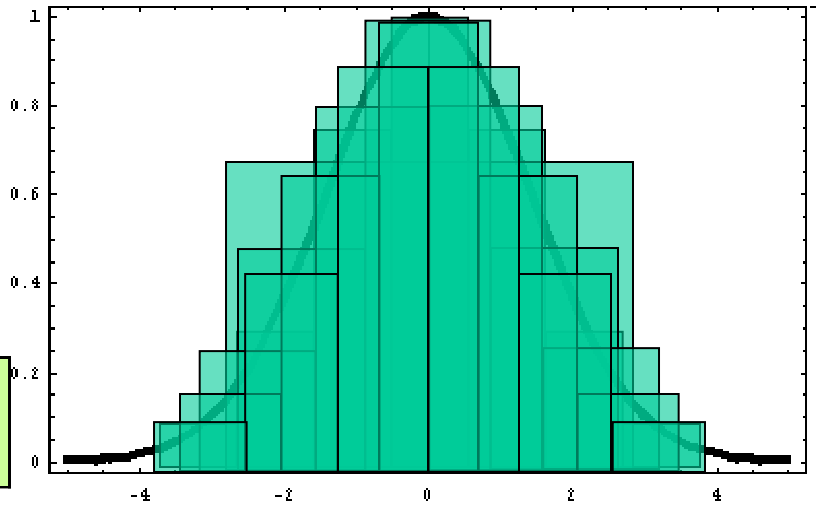
1/4 • 1 2 1

1/8 • 1 3 3 1

1/16 • 1 4 6 4 1

1/32 • 1 5 10 10 5 1

1/64 • 1 6 15 20 15 6 1



Gaussian Filters

- Remove “high-frequency” components from the image (low-pass filter)
- Convolution with itself is another Gaussian
  - Double smoothing with small-width kernel, give same result as smoothing with larger-width kernel
  - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel: Factors into product of two 1D Gaussians:

$$\begin{aligned} G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

  - Discrete domain: The filter factors into a product of 1D filters:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Correlation

Matching with templates

- Goal: find  in image
- Correlation of an image with the window template

$$R(x,y) = \sum_{x',y'} \left(T(x',y') \cdot I(x+x', y+y') \right)$$

I

T

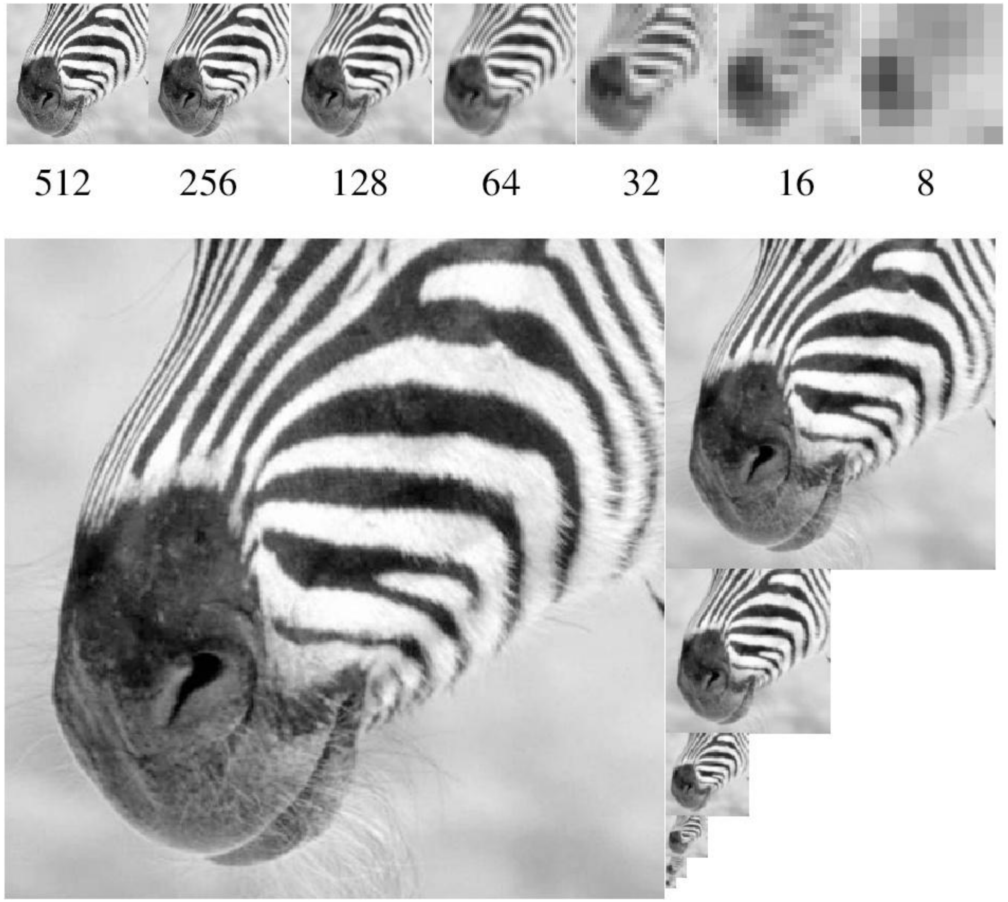
R

(Probable matchings can be obtained by thresholding)

Location of maximum

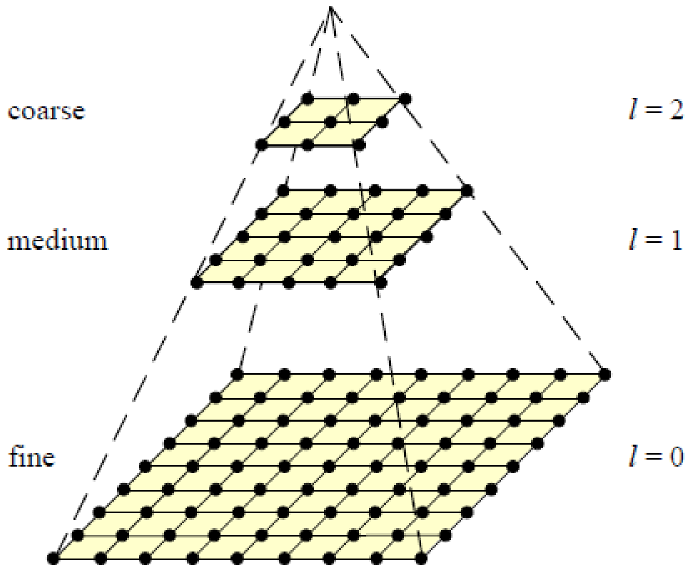
Image Pyramid

- What if we want to find larger or smaller objects/parts?



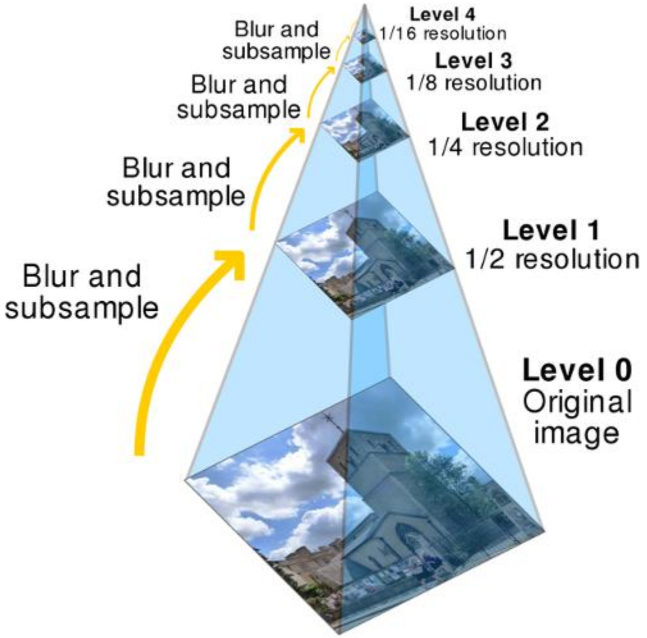
Template Matching with Image Pyramids

- What if we want to find larger or smaller objects/parts?
- Input: Image, Template
  1. Match template at current scale
  2. Downscale image
  3. Repeat 1-2 until desired minimum scale (size of interest)
  4. Take responses above some threshold, perhaps with non-maxima suppression



Template Matching with Image Pyramids

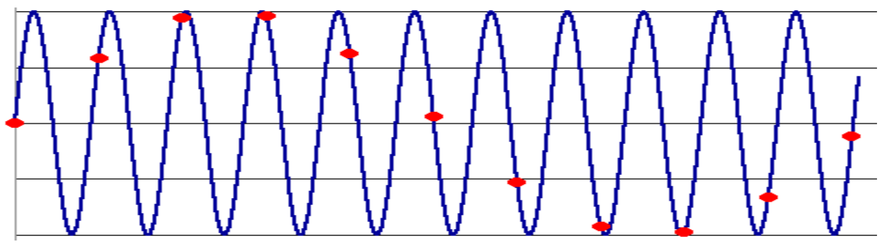
- Image Pyramids
  - Level 0: original image
  - For i=1 to L
    - Gaussian smoothing Level i-1
    - Level i: Remove odd cols and rows

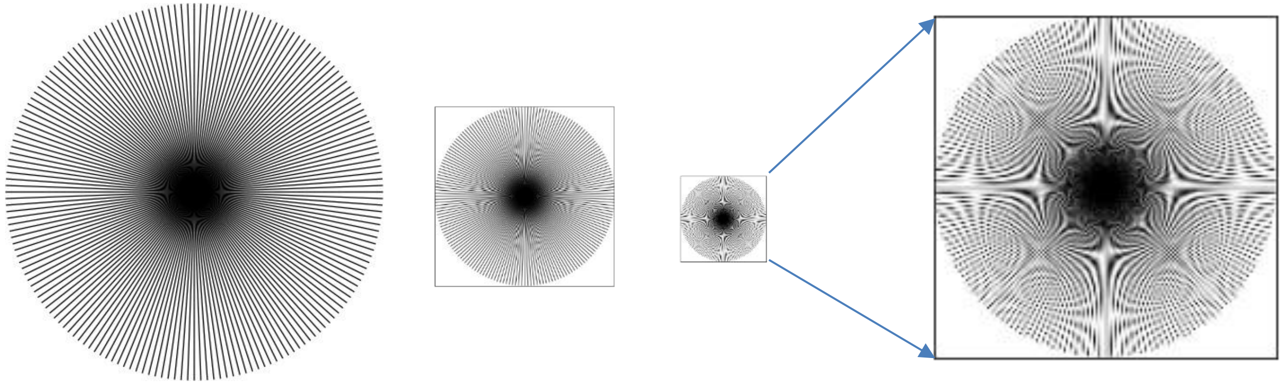


- Why is smoothing necessary?

Template Matching with Image Pyramids

- Why is smoothing necessary?
  - Because of the aliasing effect
    - Occurs when sampling rate is not high enough to capture the amount of detail in an image
    - Can give the wrong signal/image—an *aliasing*





Template Matching with Image Pyramids

- Why is smoothing necessary?
  - To avoid aliasing:
    - Sampling rate ≥ 2 * max frequency in the image
      - said another way: ≥ two samples per cycle
    - Solution: smooth the image, *then* subsample