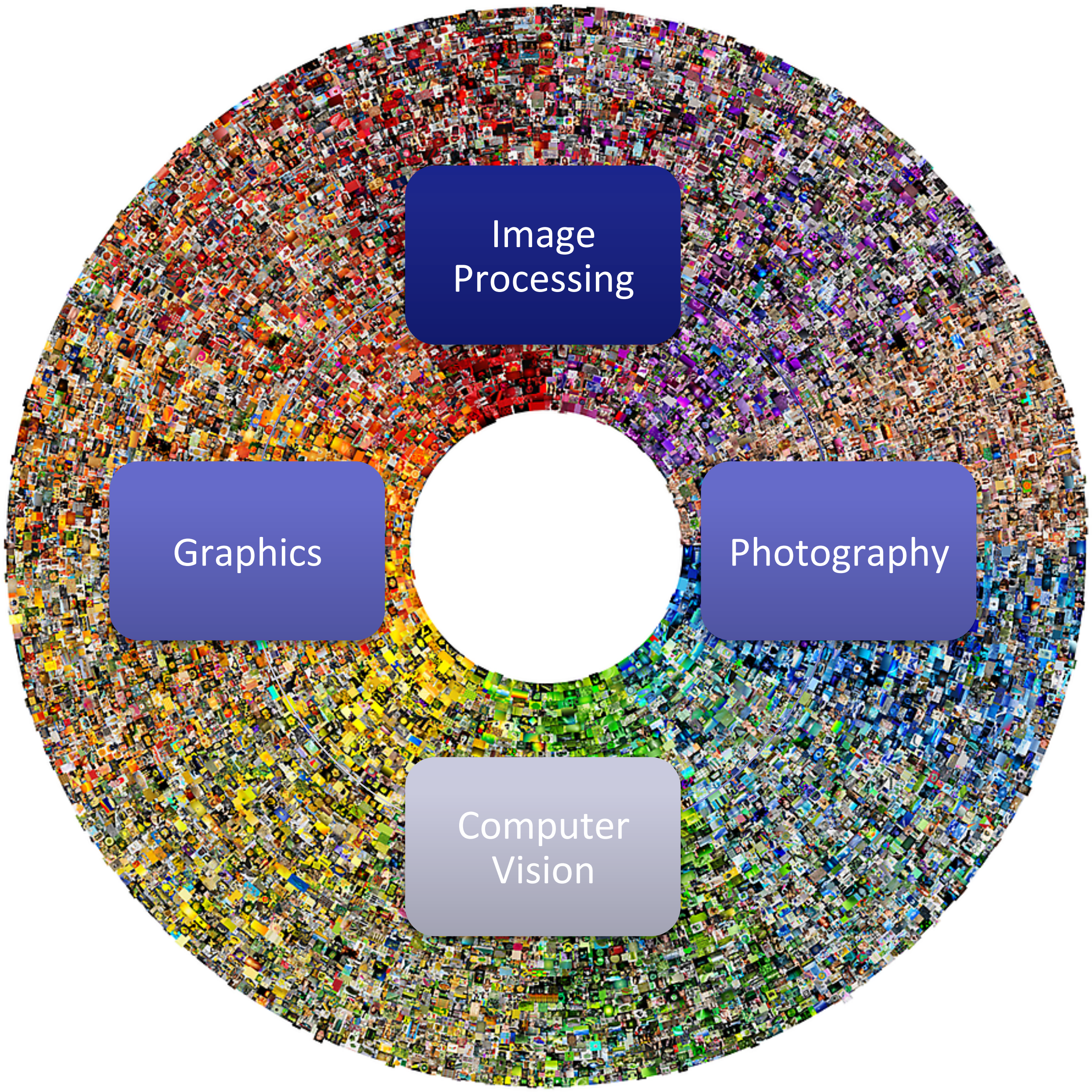
Computational Imaging:

From Photons to Photos

Peyman Milanfar

What is Computational Imaging?



Credit: Jason Salavon

Long and fascinating history

Harold “Doc” Edgerton (1903 – 1990)

“Stopping Time”

Dual Aims of Computational Imaging

Capture what I see (Photography)

- Take a nice picture (noise, dynamic range, etc.)
- Make me a better photographer.
- Do it with my simple camera.

Let’s have a look

Sensors: Form vs. Function

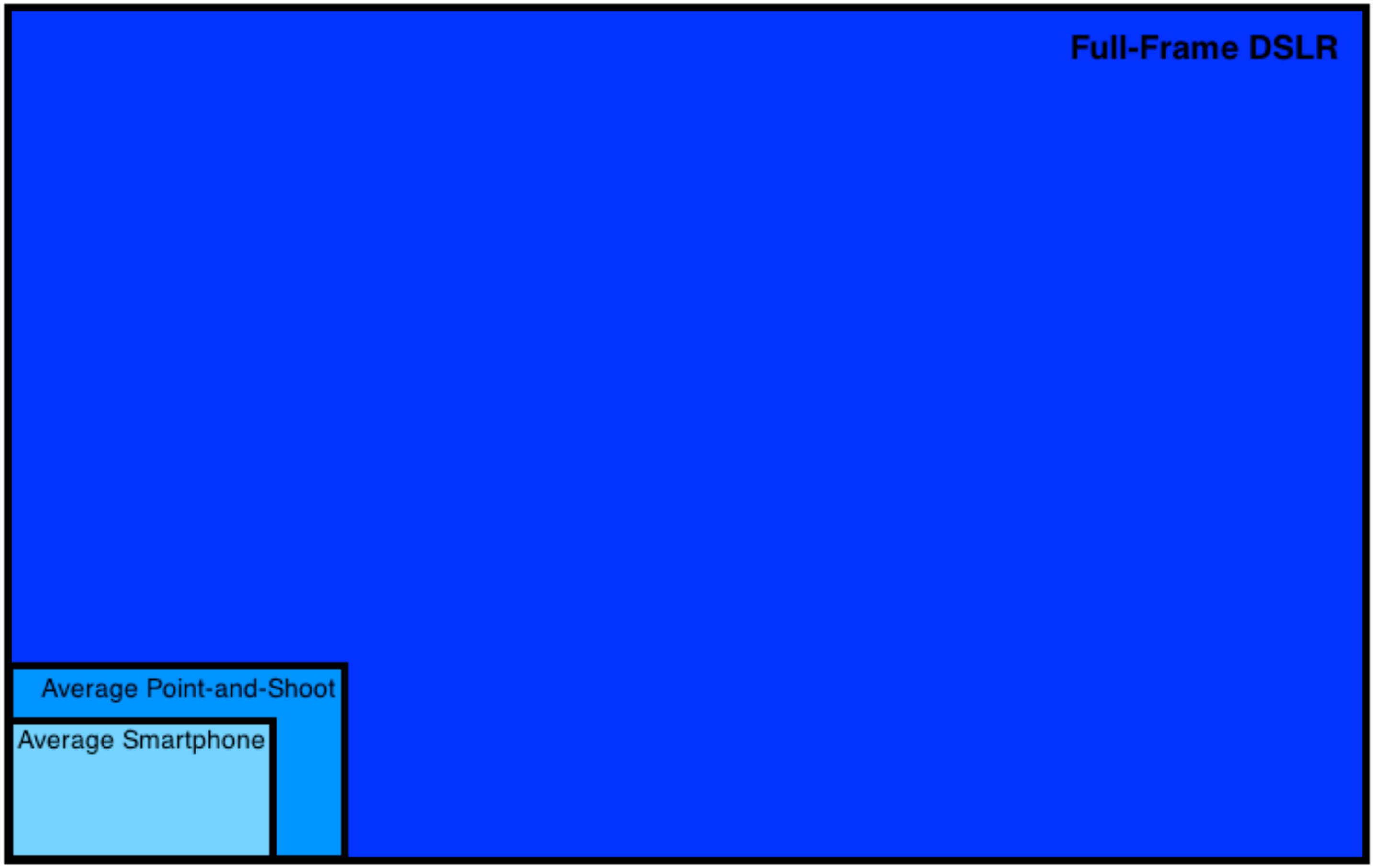
$500

$5

Mobile phone camera

Can these two ever be equally good at taking pictures?

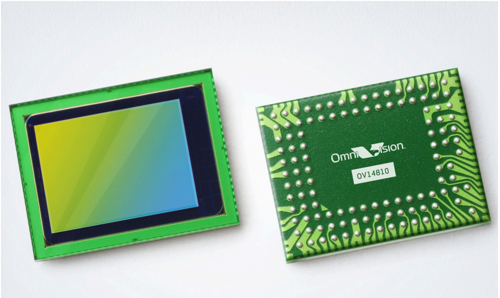
Sensor Sizes in Popular Devices



REUTERS/Adam Hunger, REUTERS/Kim Kyung-Hoon, Apple

Physical Limitations → Opportunities

- Optics won't get a lot better.
- Pixels can't get much smaller.
  - Limited by optical wavelengths
- Sensor won't get much larger
  - Limited by cost and form factor



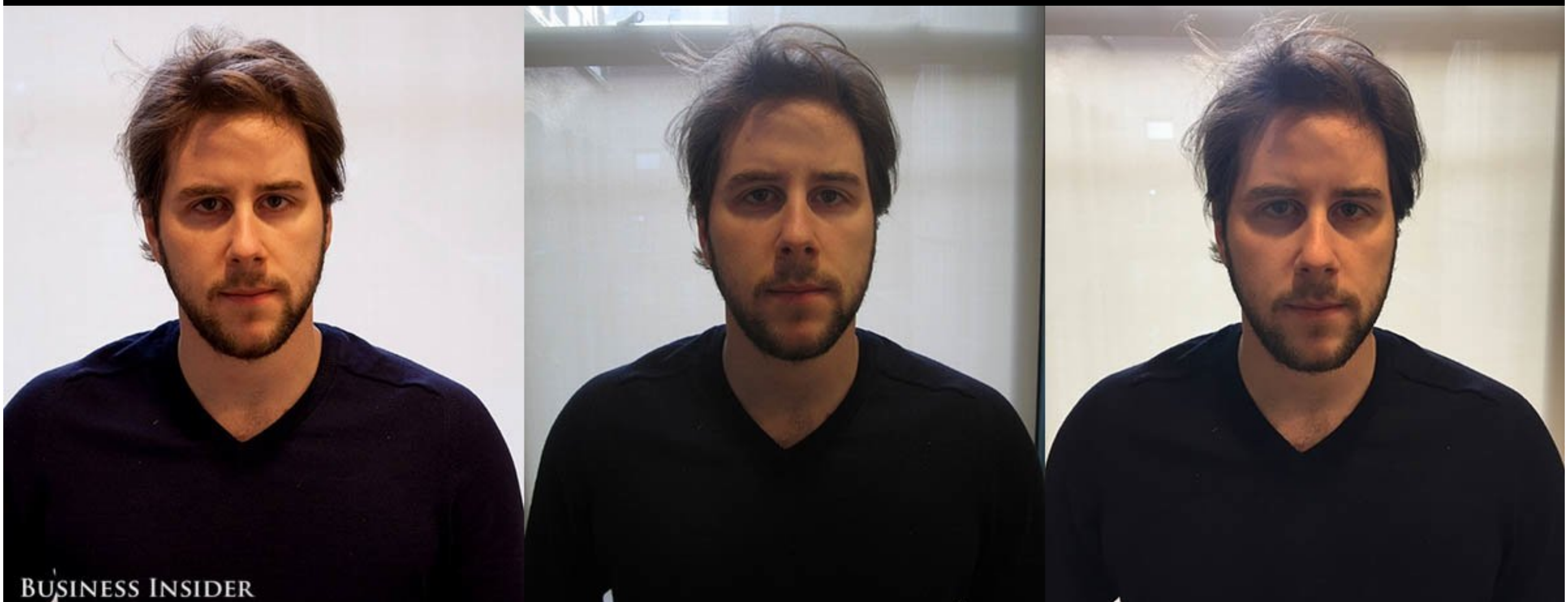
- Need:
  - Better Capture Protocols
  - Clever Algorithms

Both are from Nexus 5

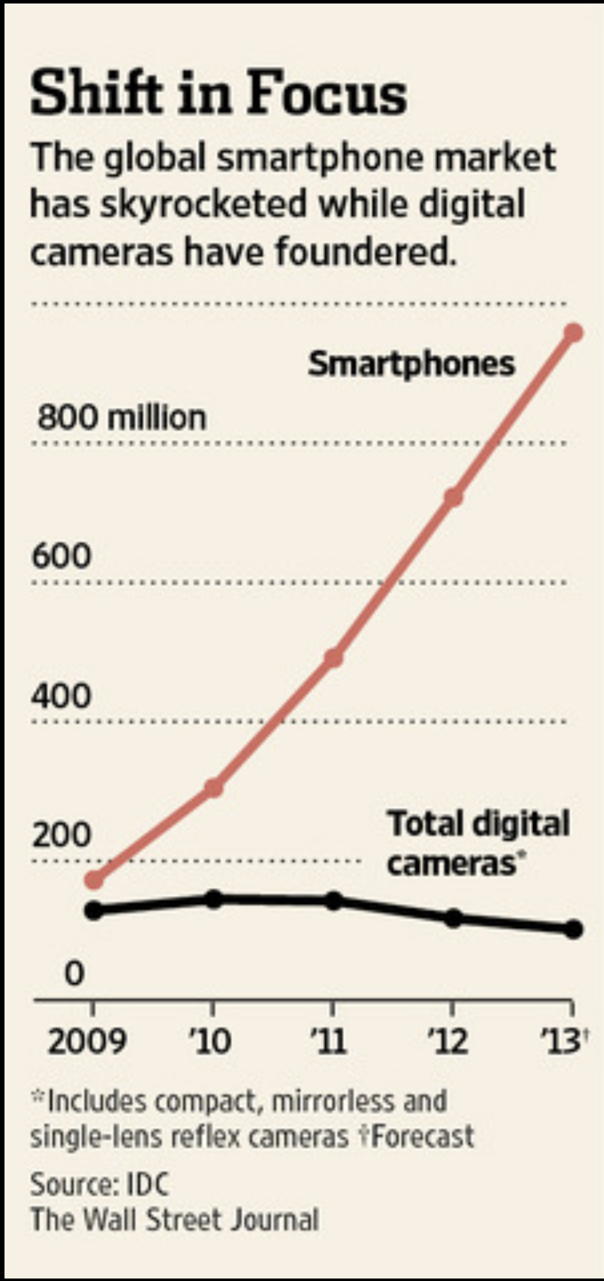
Standard shot

With Lens Blur

Credit: Sascha Haeberling


BUSINESS INSIDER

Mobile Imaging is Totally Dominant




2005
Luca Bruno / AP
2013
NBC NEWS
Michael Sohn / AP

2015 -- Paris

AP

2015 -- Moscow

2015 -- Sydney

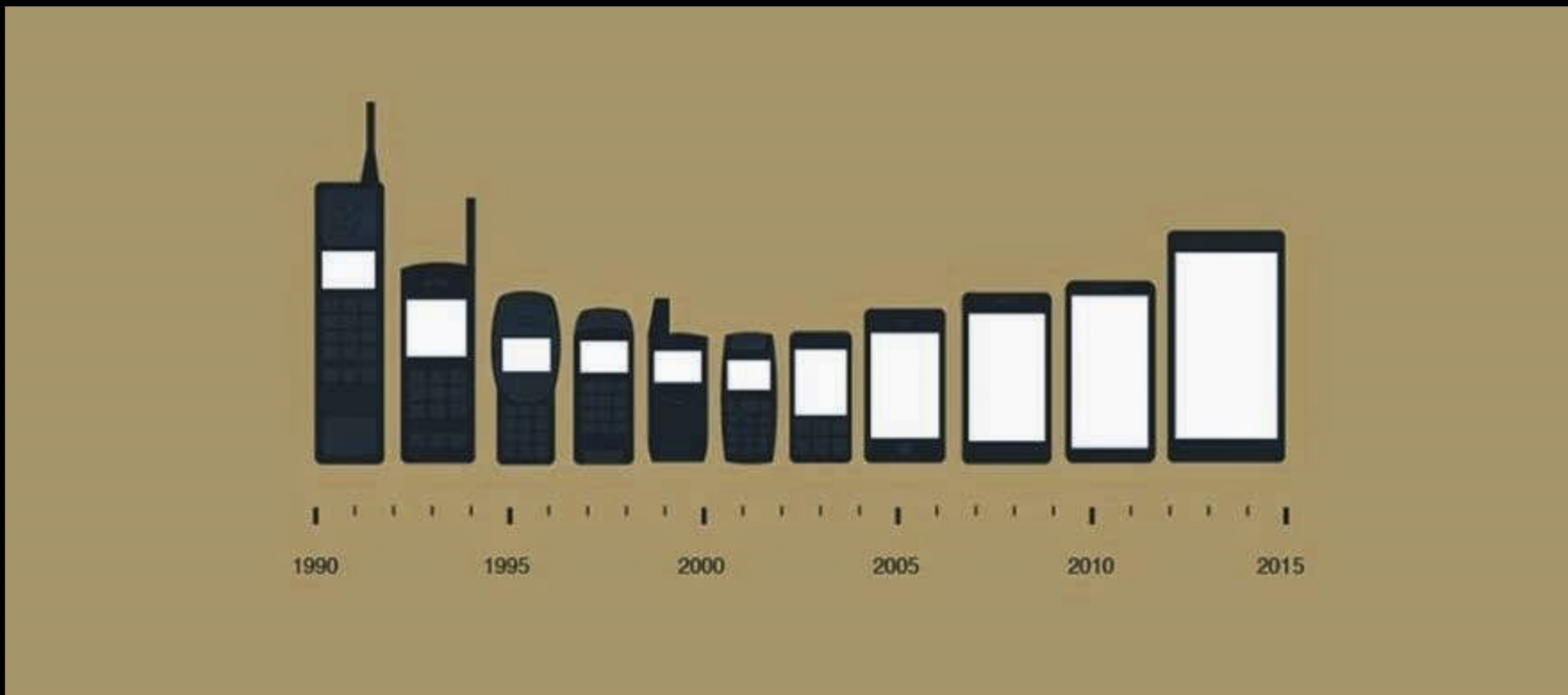
Reuters

1,000,000,000,000

Roughly a trillion photos shared on social media (in 2014)

Reuters

Evolution of the (Smart)phone



Dual Aims of Computational Imaging

Capture what I see/like (Photography)

- Take a nice picture. (noise, dynamic range, etc.)
- Make me a better photographer.
- Do it with my simple camera.

Capture what I can’t see (Science, Military)

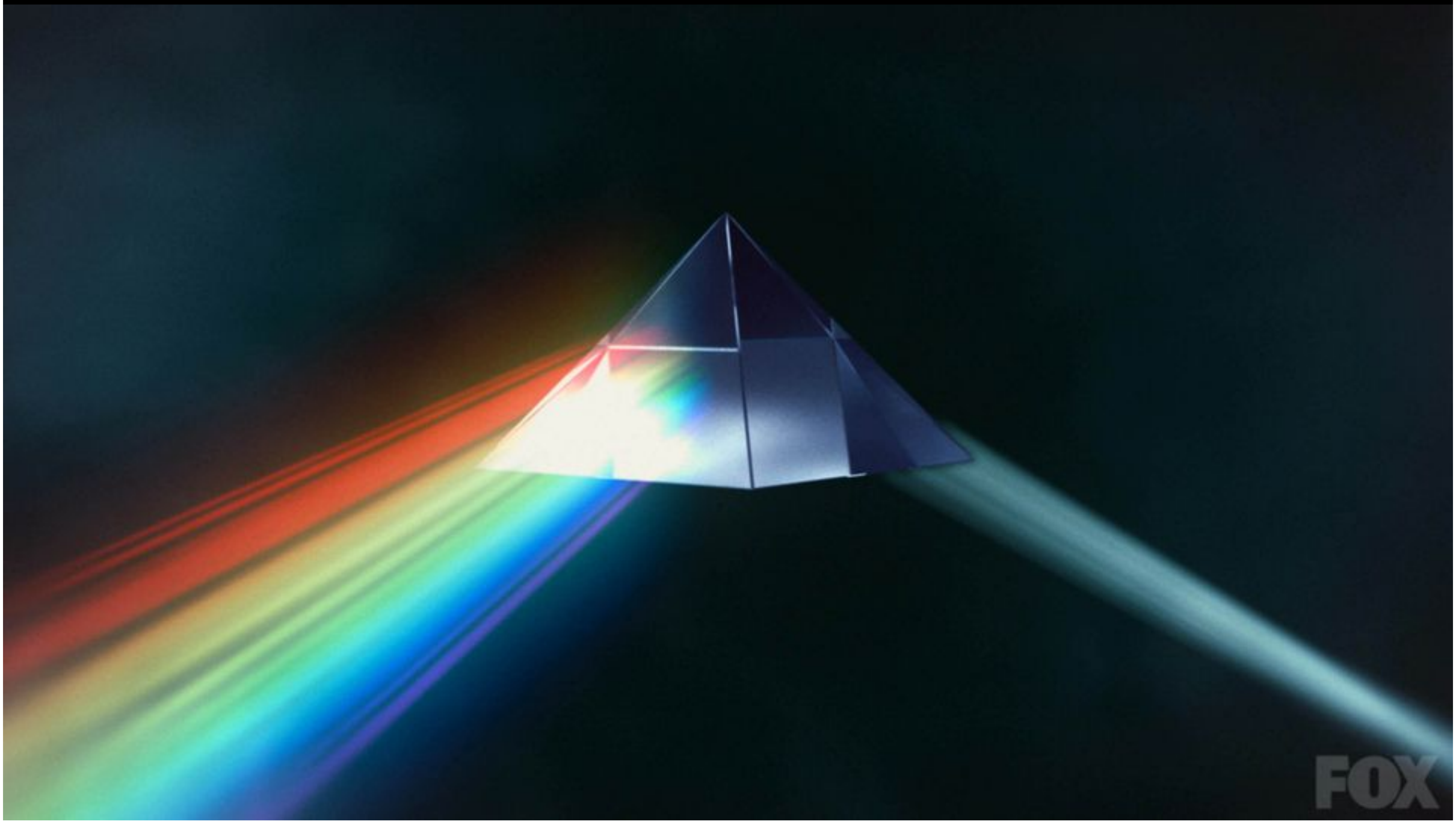
- See in the dark
- View fast, slow, small, or faint phenomena
- Detect, magnify subtle changes

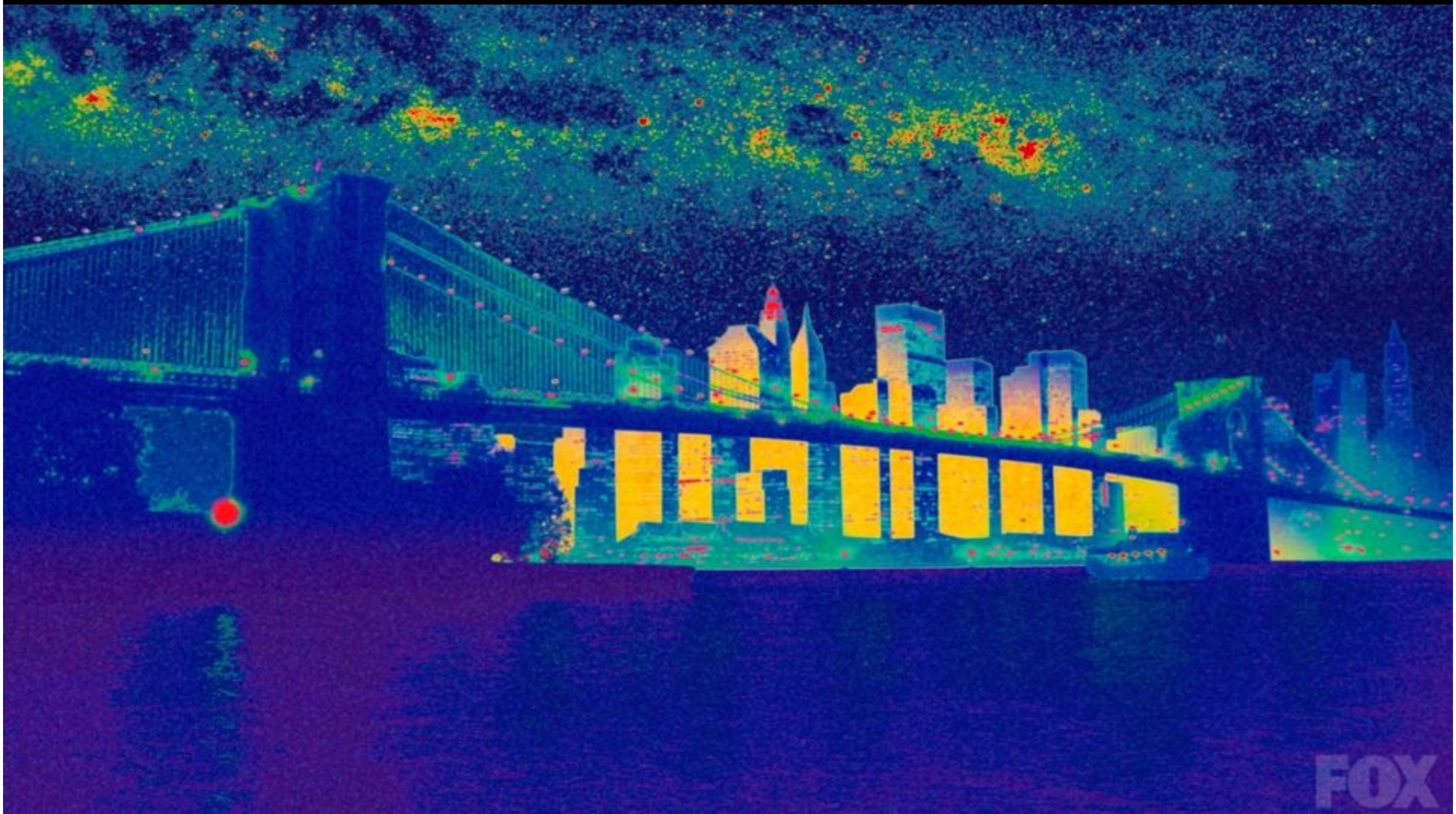
Yosemit Valley, California

At night

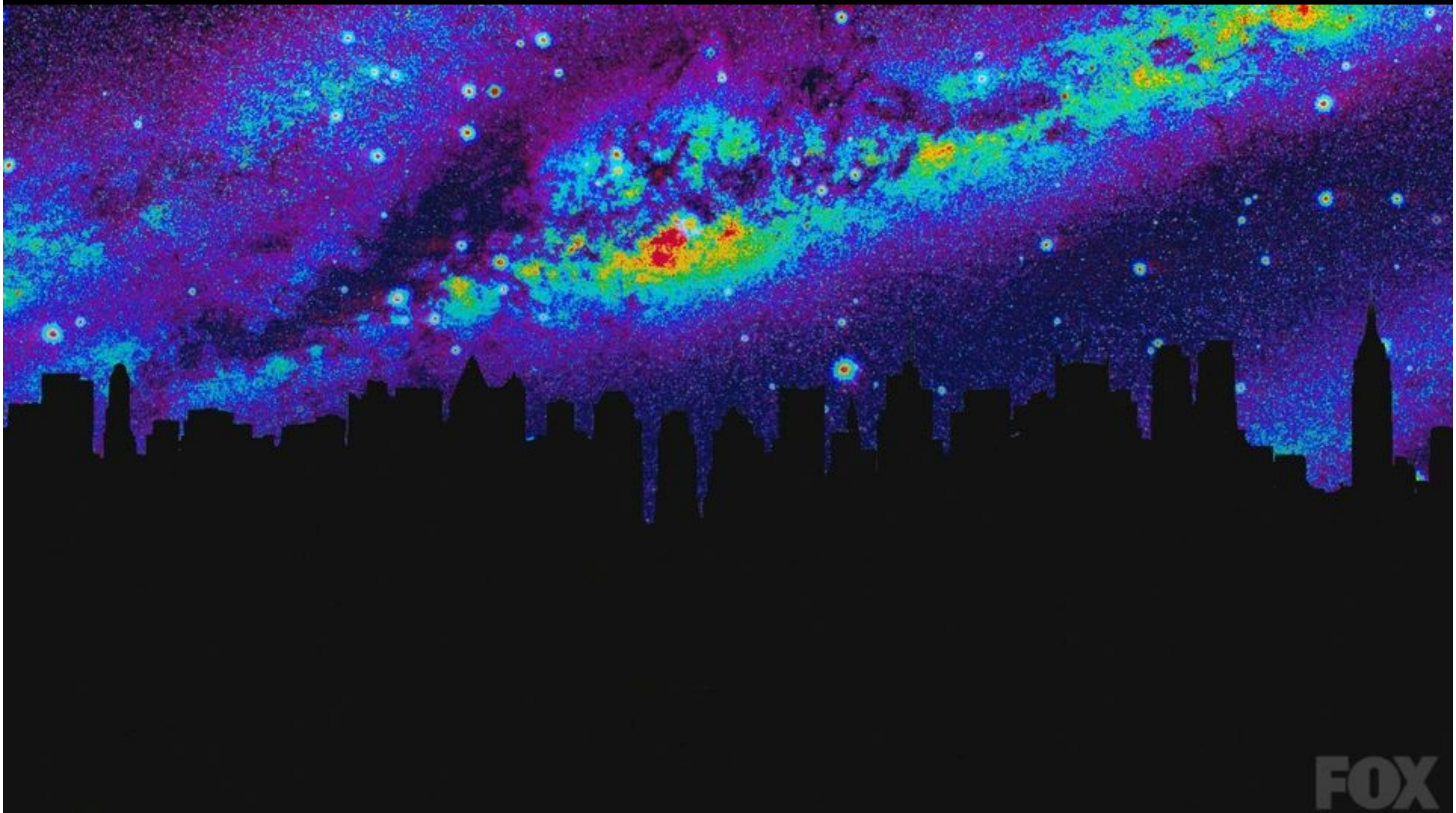
(Jesse Levinson Canon 10D, 28mm f/4, 3 min, ISO 100, 4 image pano)

Can’t “see” most of nature that surrounds us





Infrared



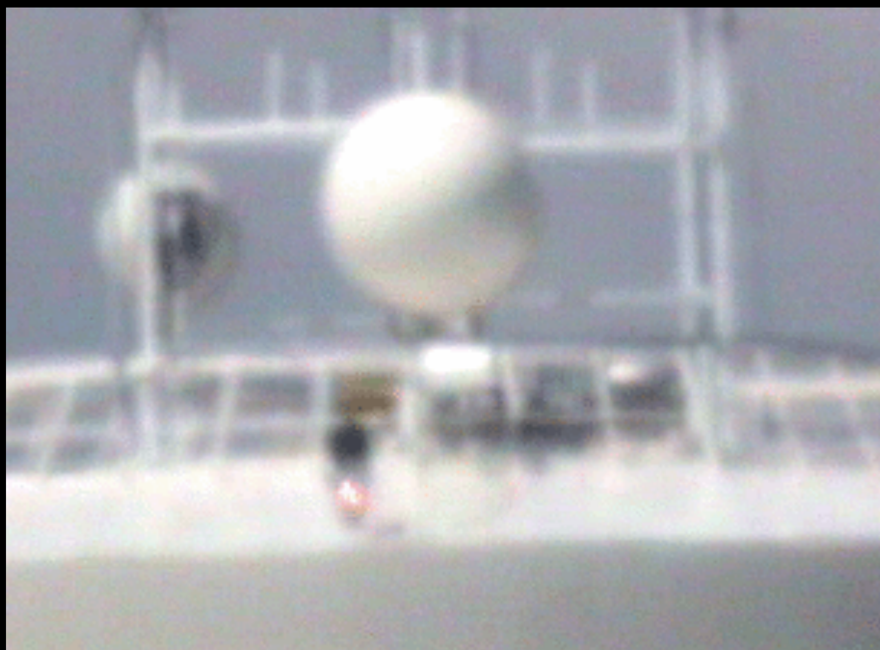
Gamma-ray

X-ray

Seeing Far Away

Credit : Florian Kainz

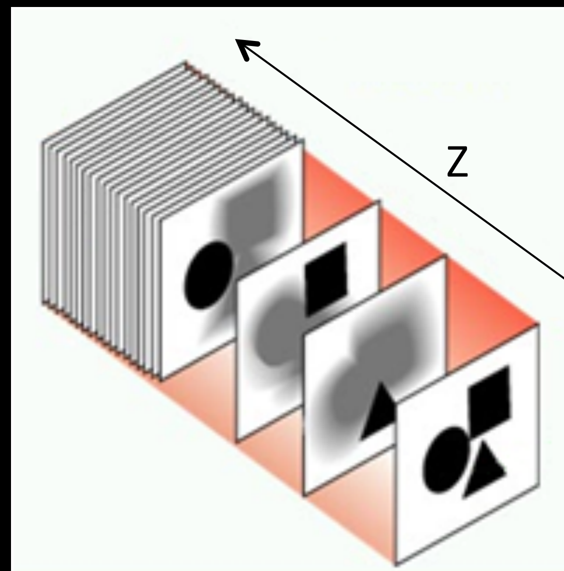
Seeing Far Away

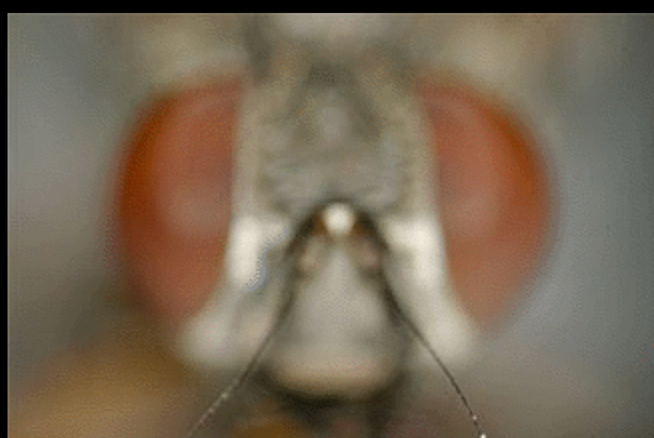


Top part of a water tower imaged at a (horizontal) distance of 2.4 km

X. Zhu, P. Milanfar, “Removing Atmospheric Turbulence”, PAMI, 2012

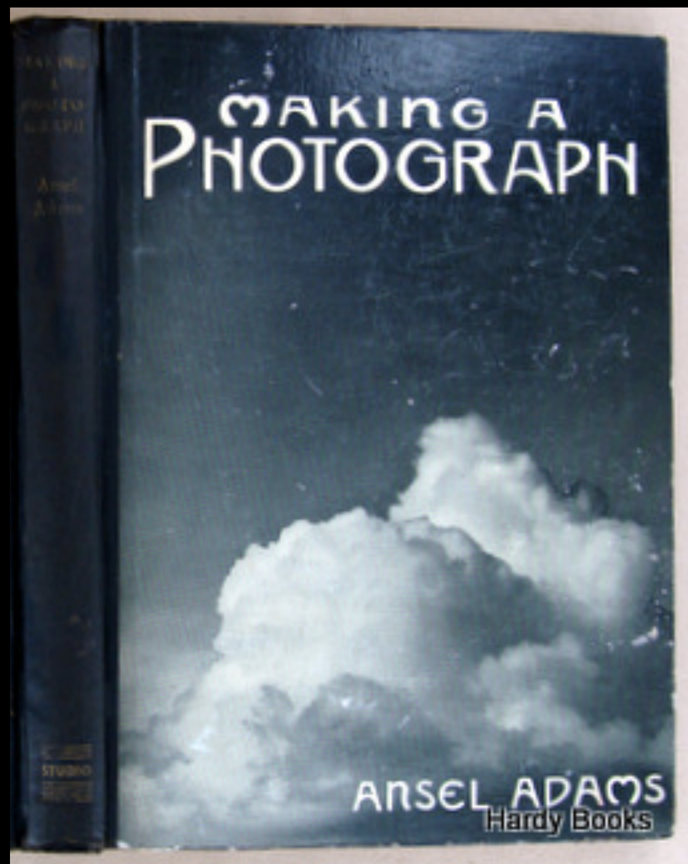
Seeing Small Things

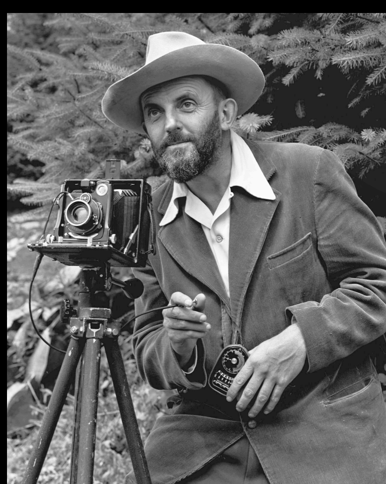




X. Zhu, P. Milanfar, "Removing Atmospheric Turbulence", PAMI, 2012

How Do We Make a High Quality Image?

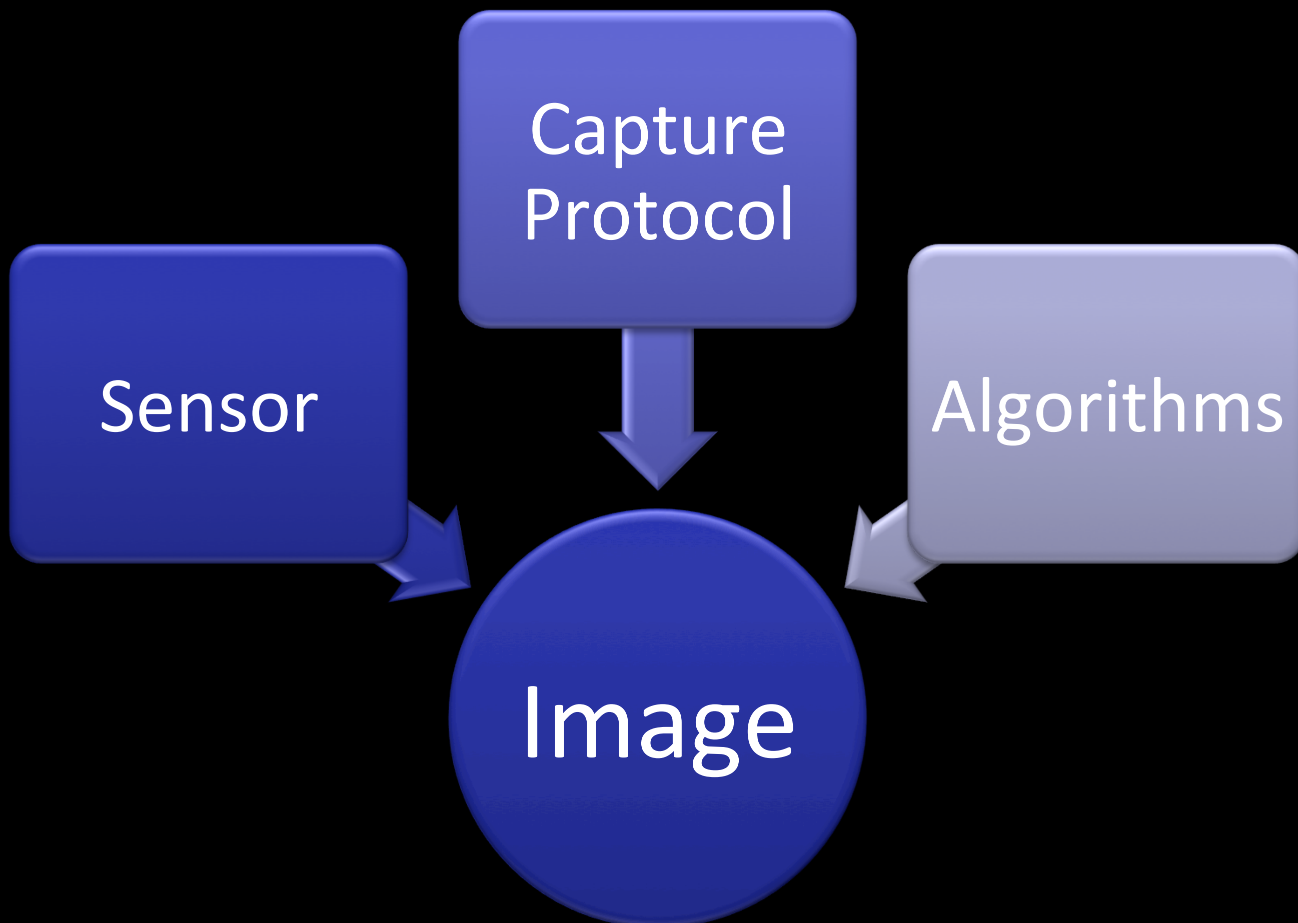




Ansel Adams' first book, published 1935

How Do We Make a High Quality Image?

- Integration of Sensor and Computation plays a key role in the image formation process.



Computational Photography on Android

devCam

Main | Interface | Capture Designs | Capture Output | Directories
Example Applications | devCam JSONs | devCam and MATLAB | Device Database

devCam

Parameterized Camera Control for Algorithm Development and Testing

devCam is an app for parameterized image capture using Android devices. devCam makes it simple to capture complicated sets of photographic exposures with user-defined values for standard photographic settings. It is designed to give the user as much control as the camera allows, making use of the camera2 API (requires Lollipop, Android 5.0+) to give the user manual control over the following (if the device is capable):

- Exposure time
- ISO
- Aperture
- Focal Length
- Focus Distance

devcamera.org

https://youtu.be/92fgcUNCHic?t=29m56s

Standard Burst Photography

Locked: Exposure, Focus

HDR Burst Photography

Locked Focus
Variable Exposure

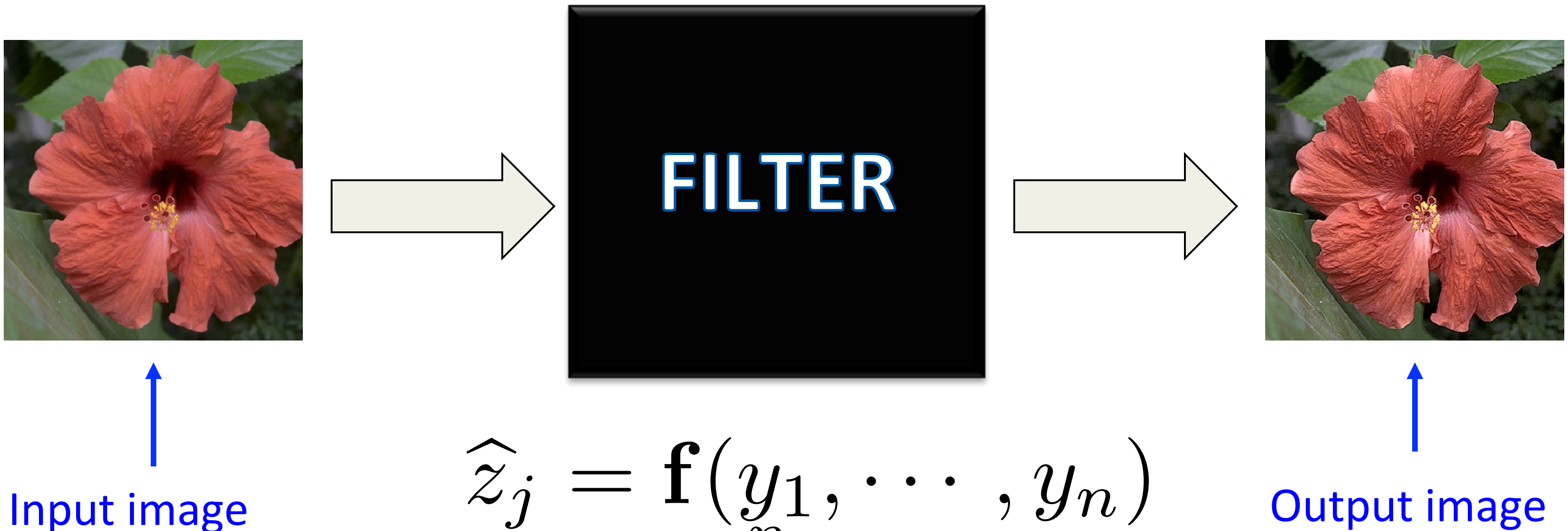
Focal Sweep Photography

Locked Exposure
Variable Focus

Part 2:

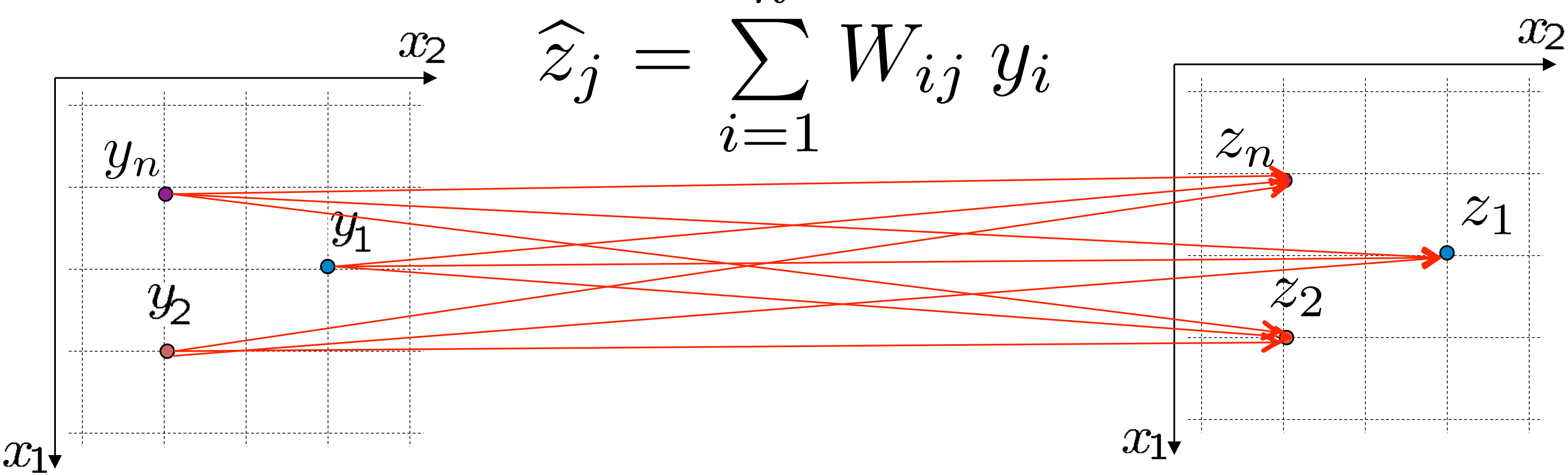
A general filtering framework

The Black Box



$$\widehat{z}_j = \mathbf{f}(y_1, \cdots, y_n)$$

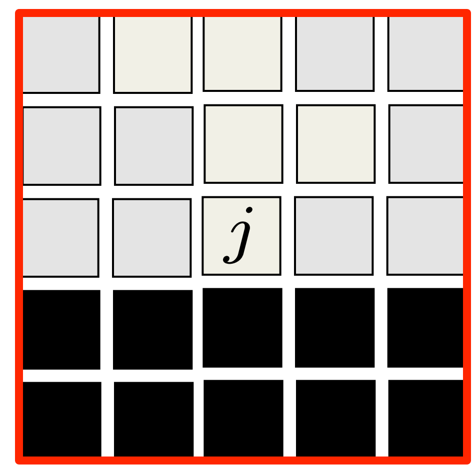
$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$



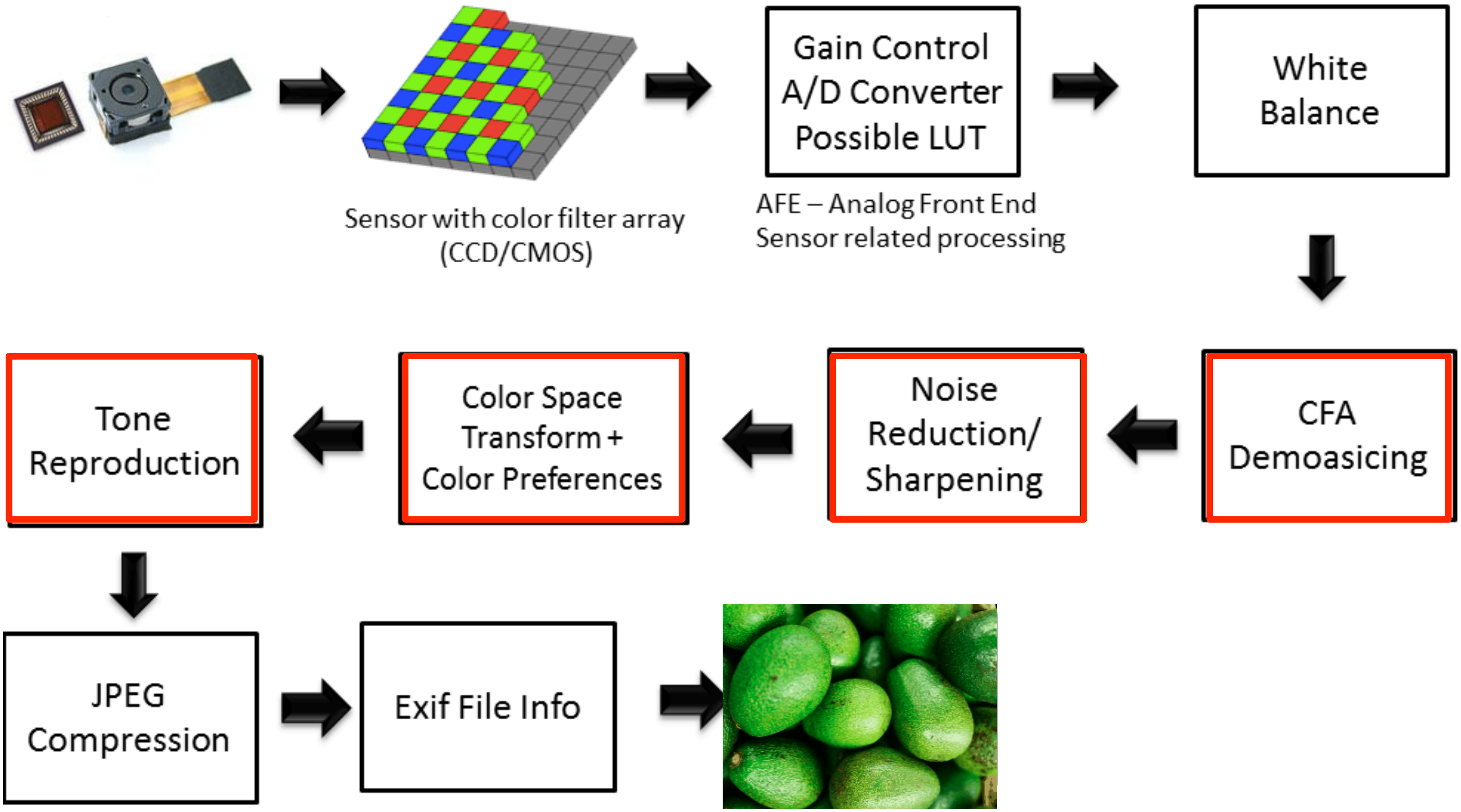
How to design the black box

$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$

$y_i \in$

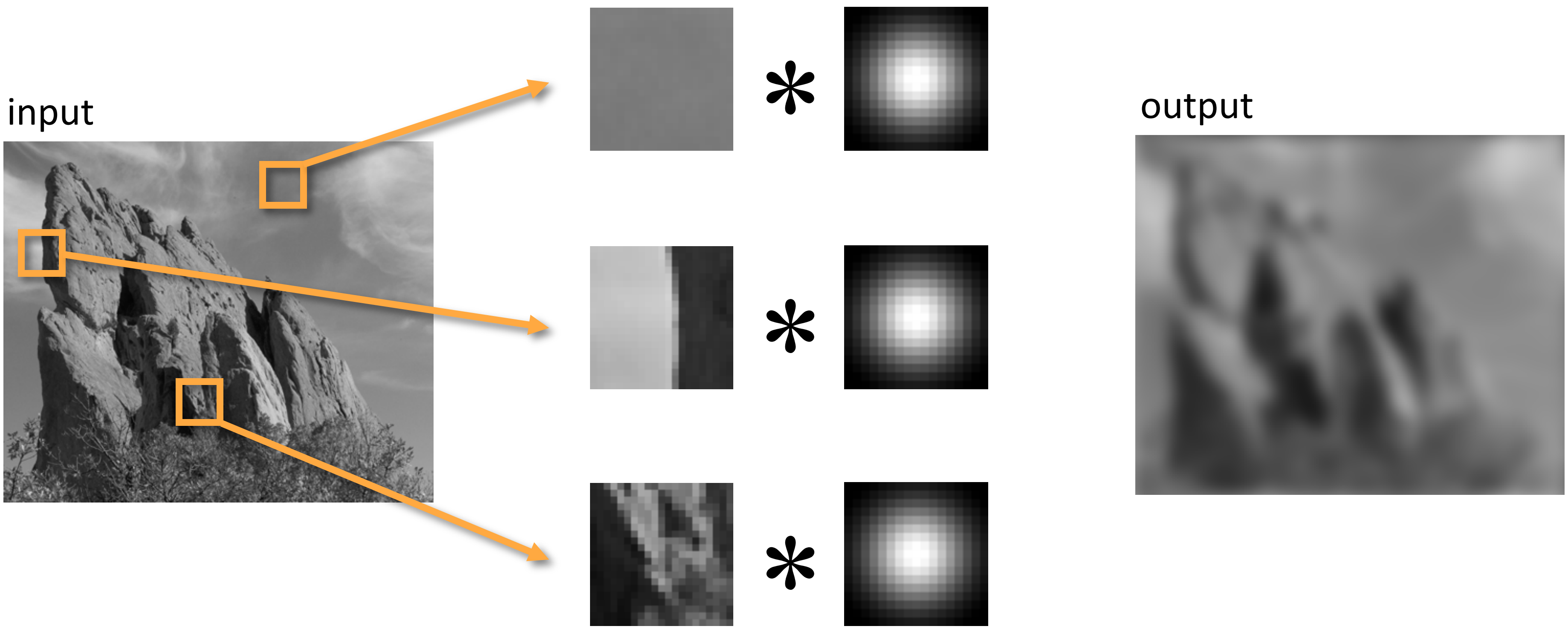


Model for Many Parts of Pipline



Same weights everywhere

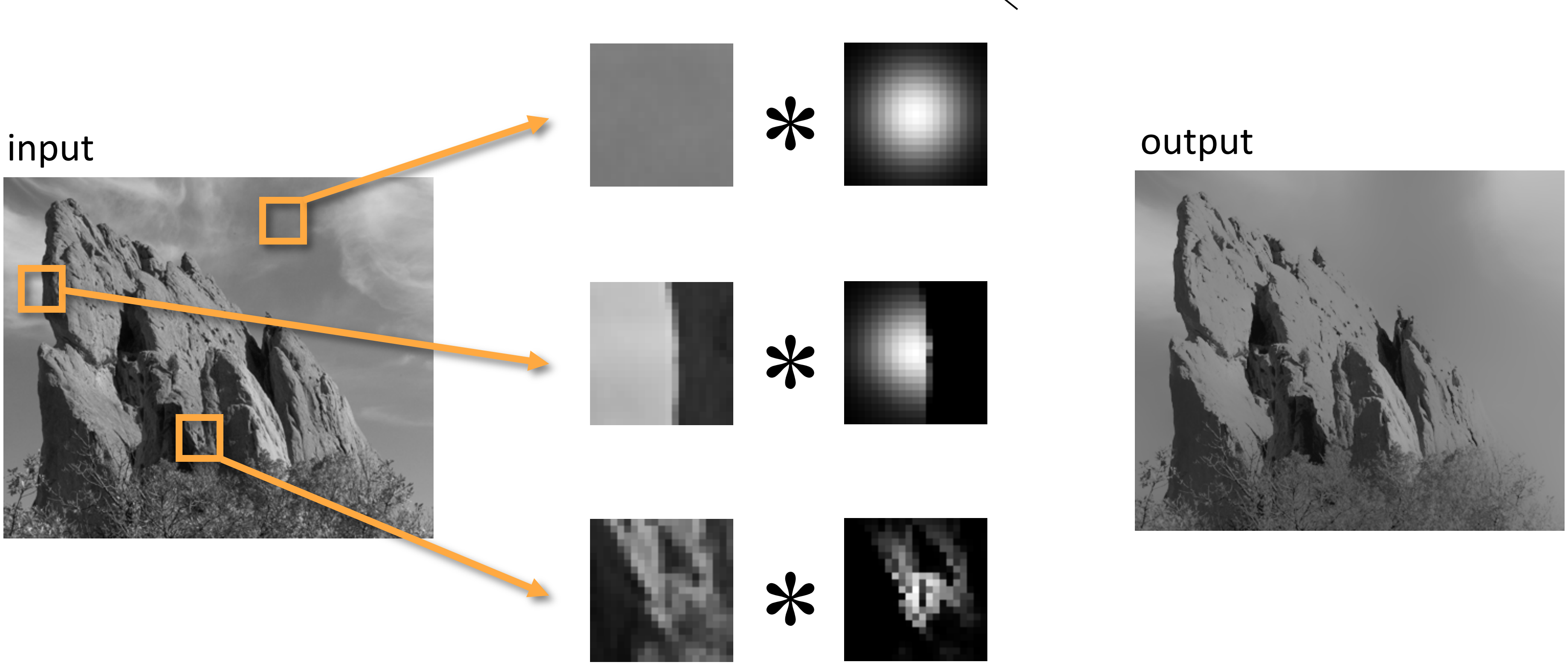
$$\widehat{z}_j = \sum_{i=1}^{n} W_i \, y_i$$



Same Gaussian kernel everywhere.

Data-adaptive weights

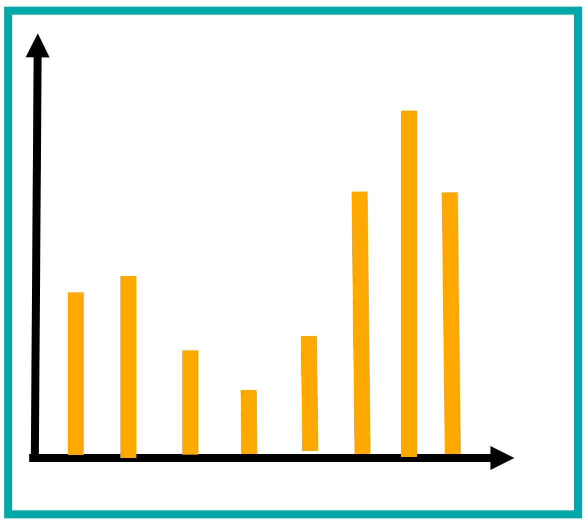
$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$



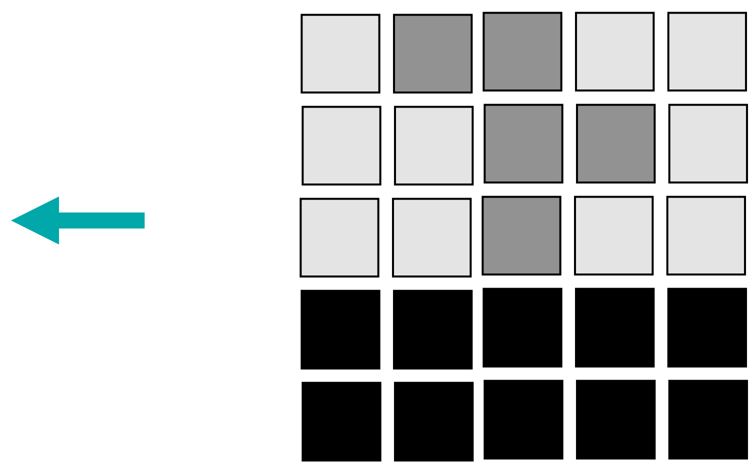
The kernel shape depends on the image content.

Method 1: Consider the histogram

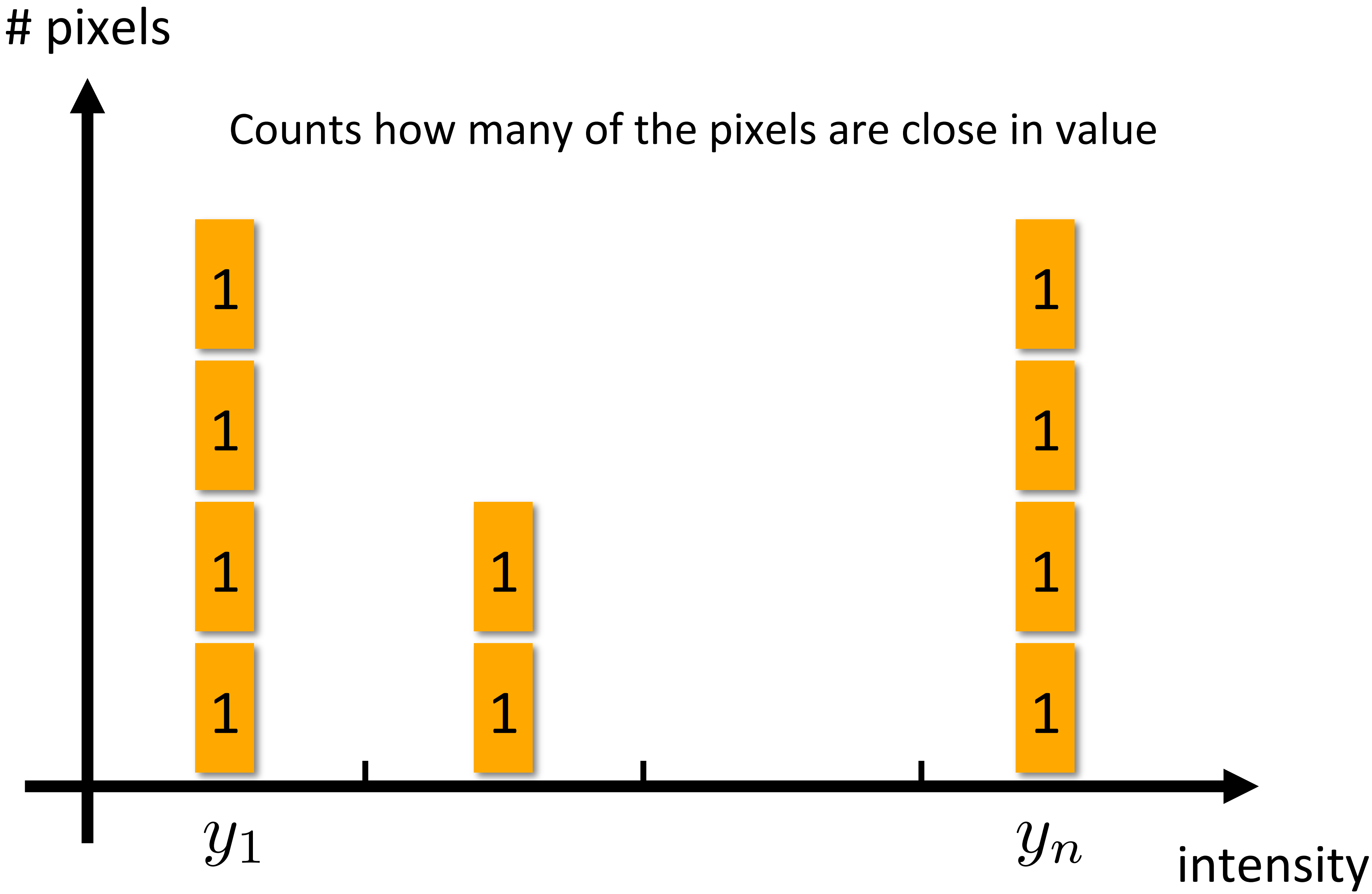
Histogram



y_i



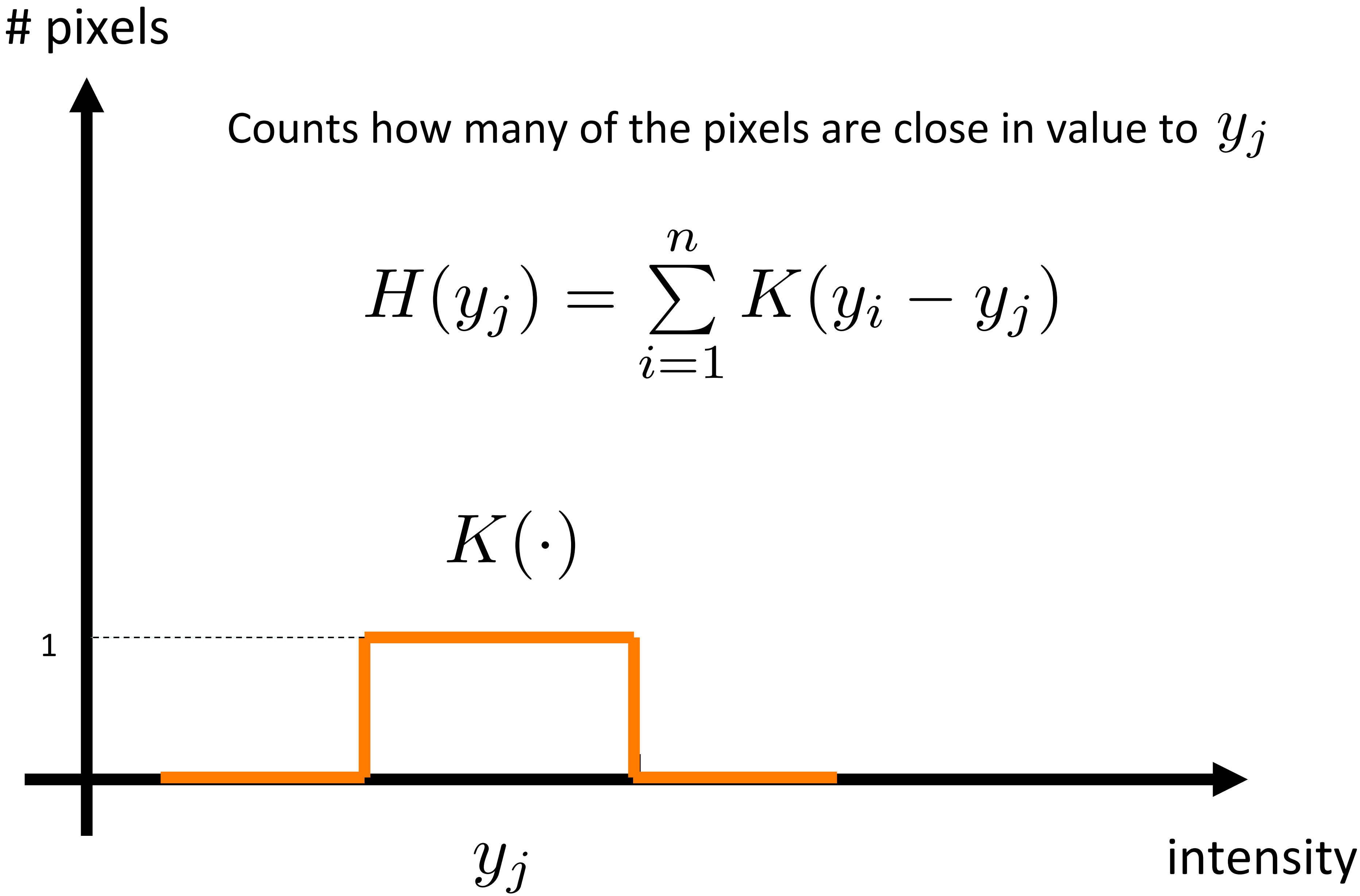
Just for illustration – we don't really make the histogram.

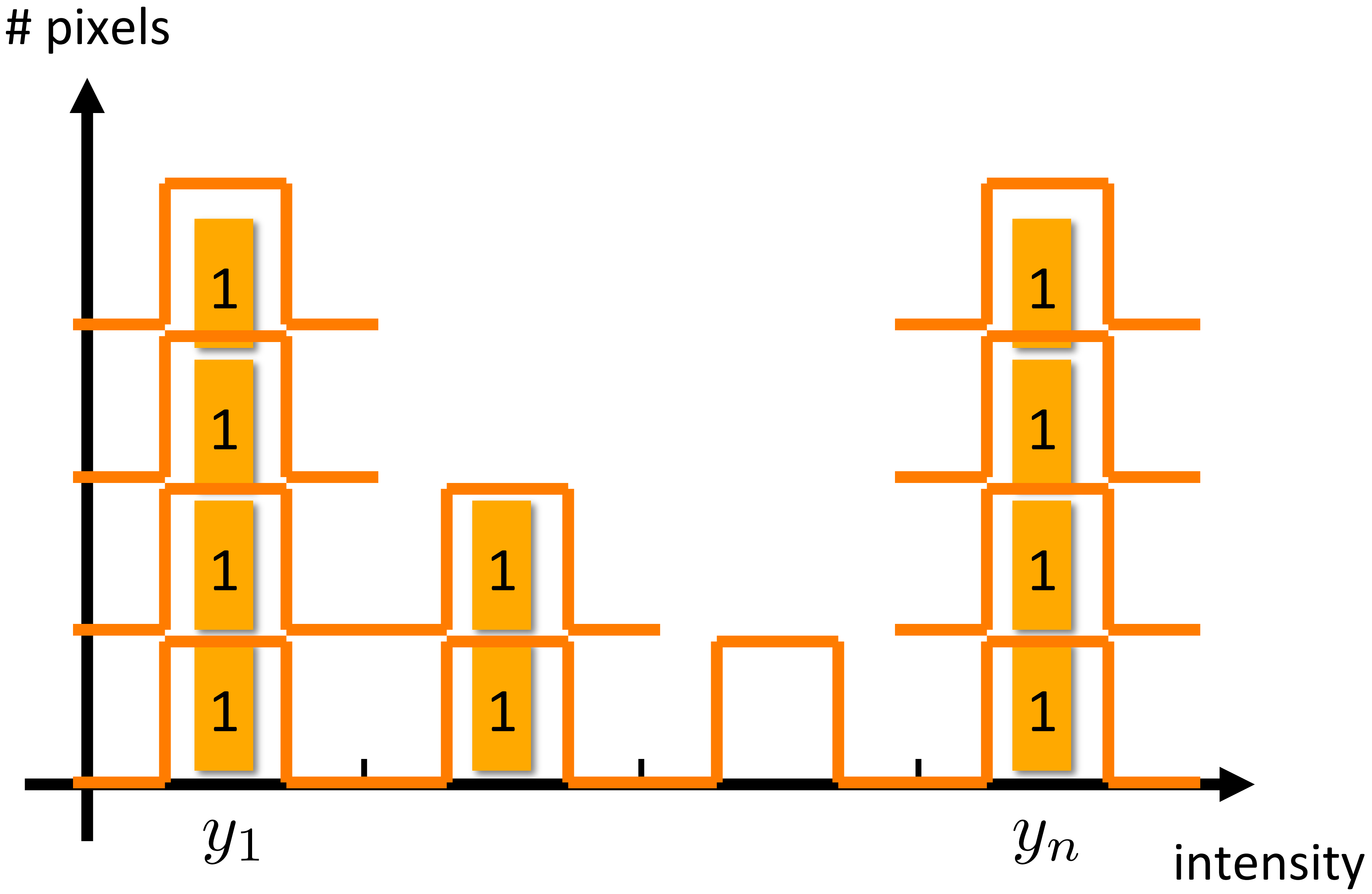


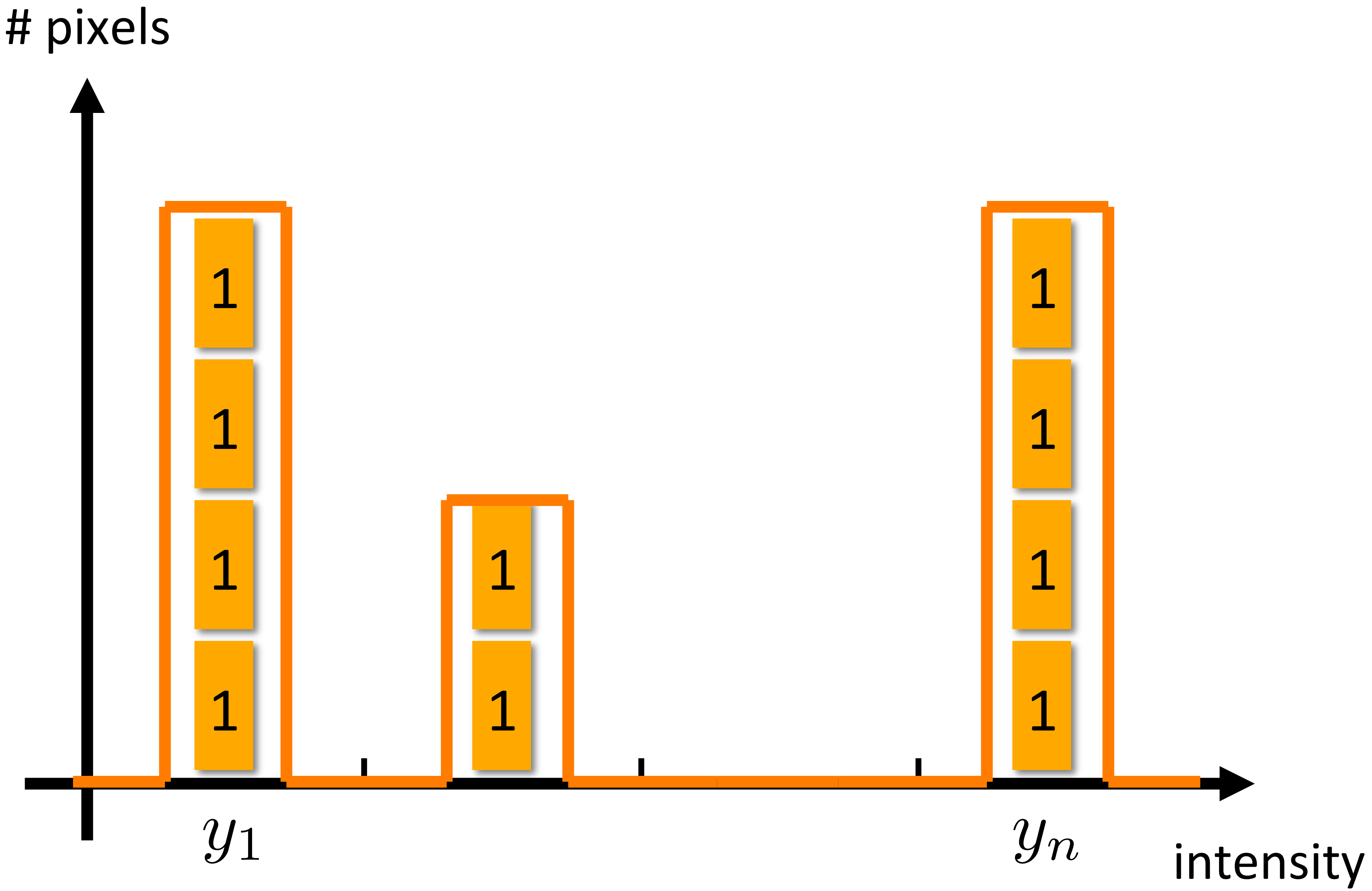
pixels
Counts how many of the pixels are close in value
1
1
1
1
1
1
1
1
1
1
y_1
y_n
intensity

Counts how many of the pixels are close in value to y_j

$$H(y_j) = \sum_{i=1}^{n} K(y_i - y_j)$$




pixels
1
1
1
1
1
1
1
1
1
1
y_1
y_n
intensity



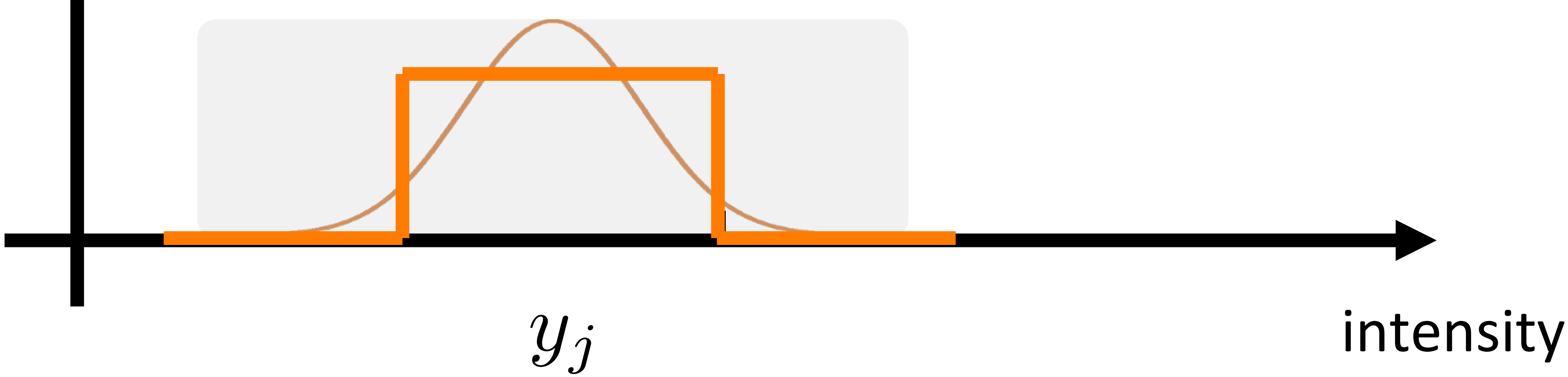
pixels
1
1
1
1
1
1
1
1
1
1
y_1
y_n
intensity

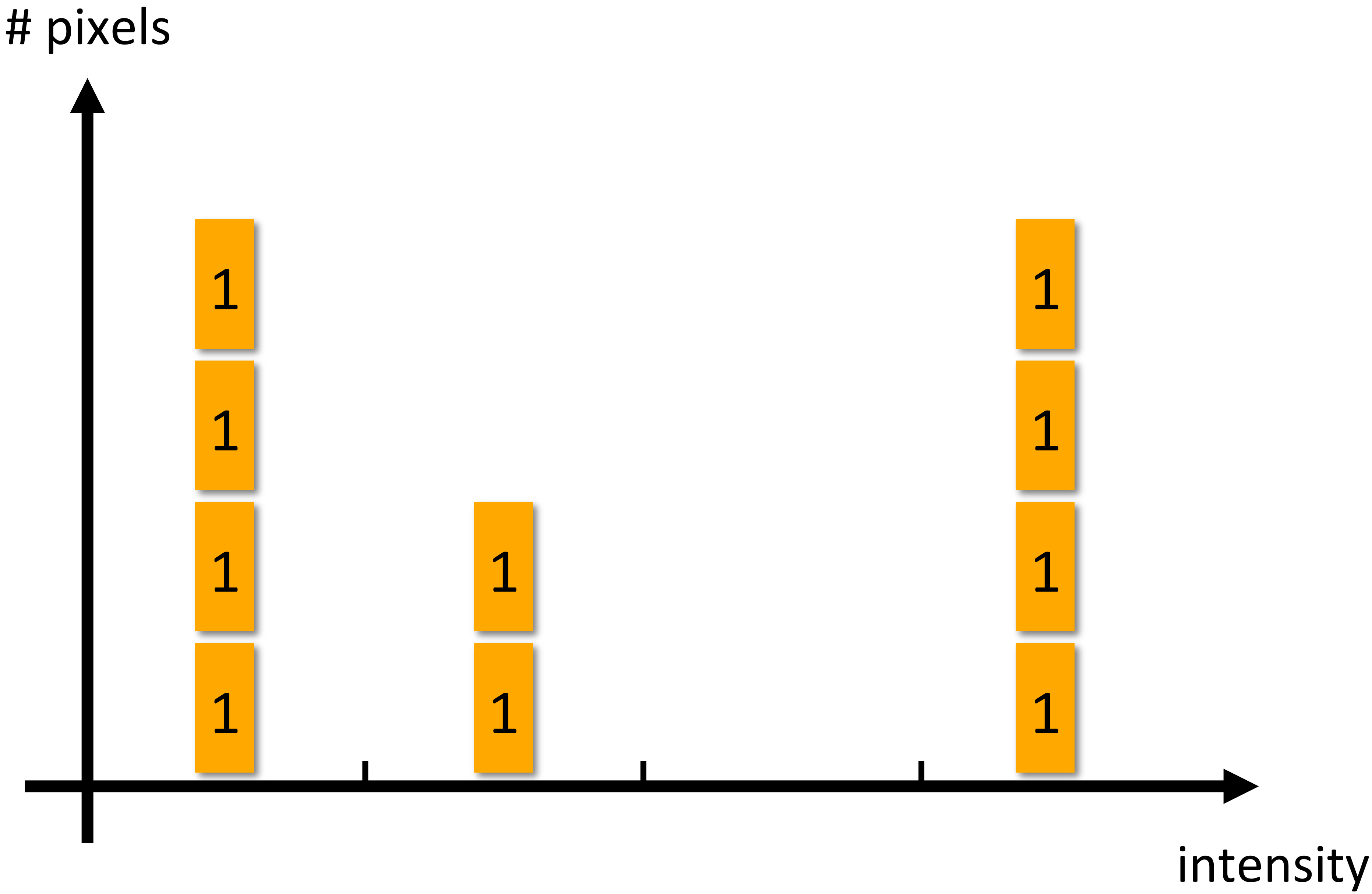
Why only the box function?

Soft count of how many of the pixels are close in value to y_j

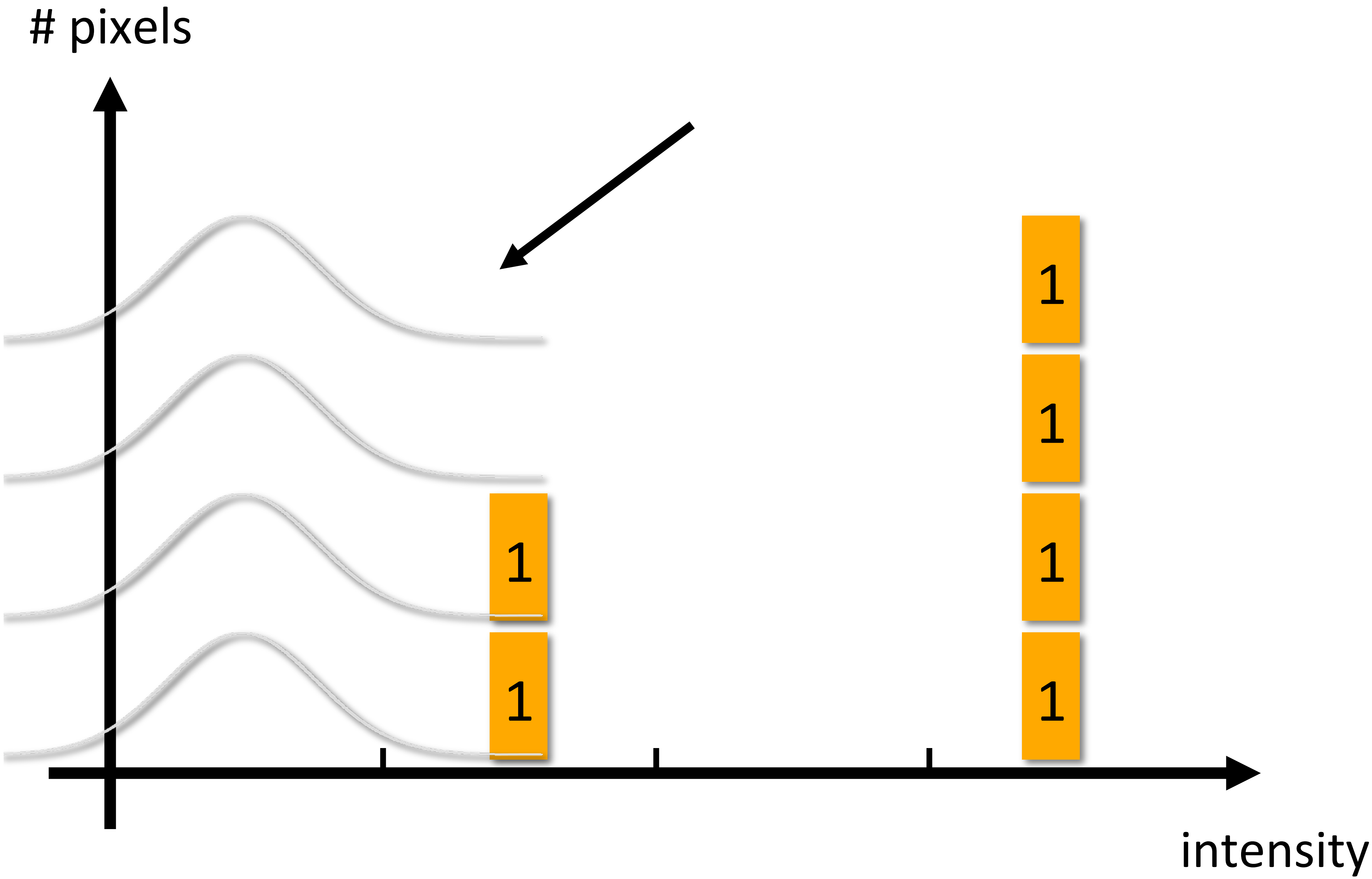
$$H(y_j) = \sum_{i=1}^{n} K(y_i - y_j)$$

$K(\cdot)$: Symmetric

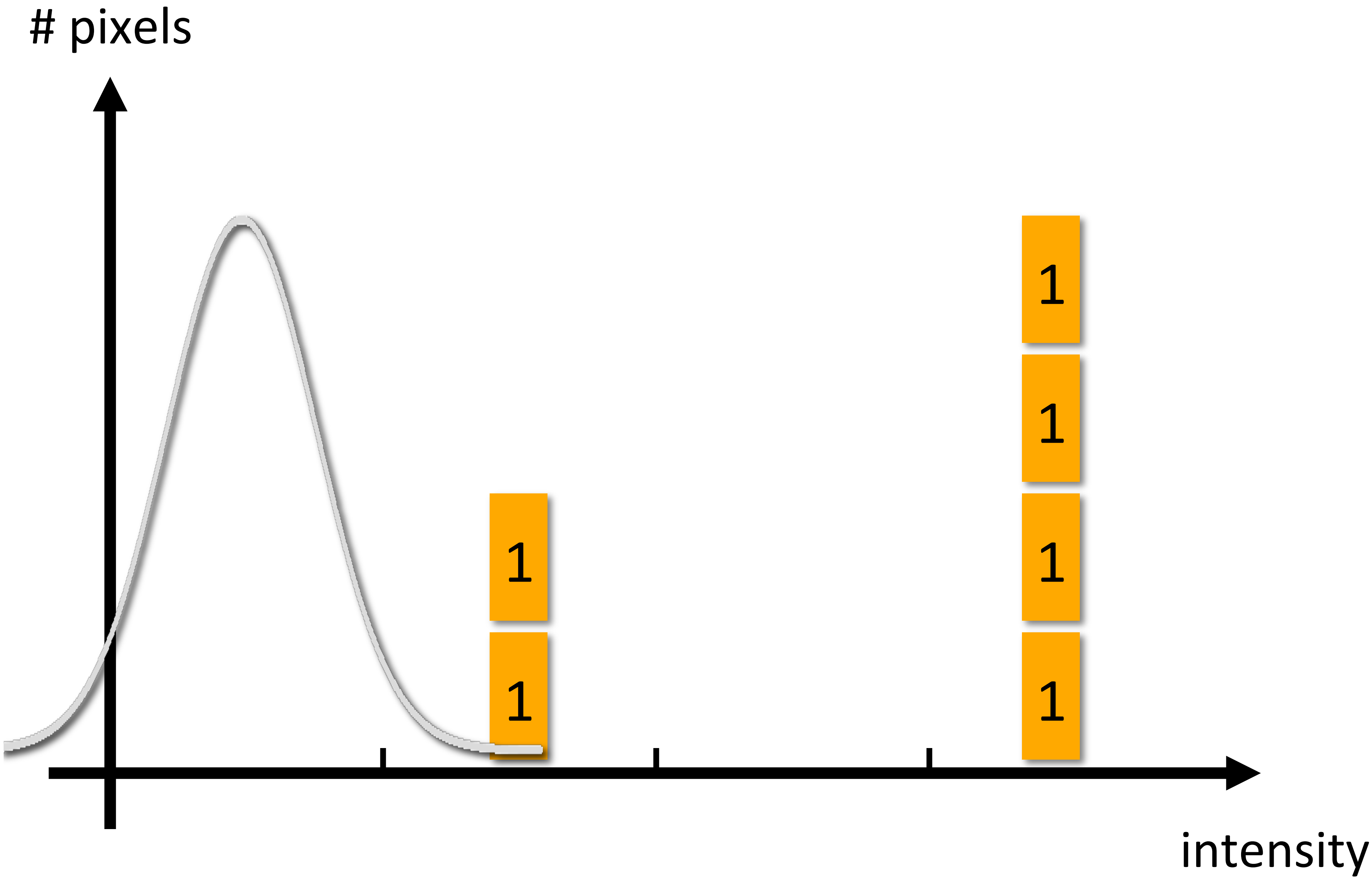




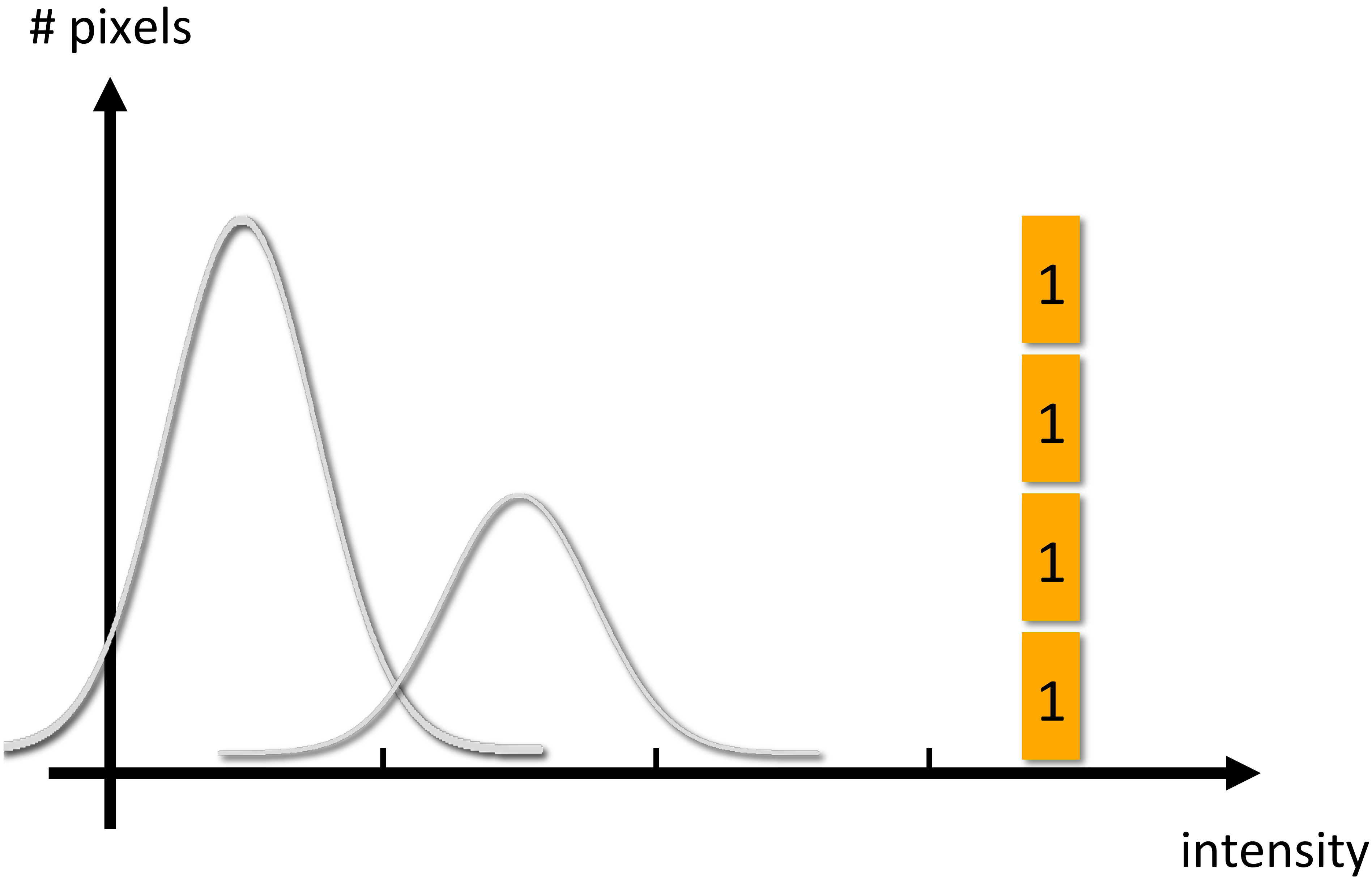
pixels
1
1
1
1
1
1
1
1
1
1
intensity



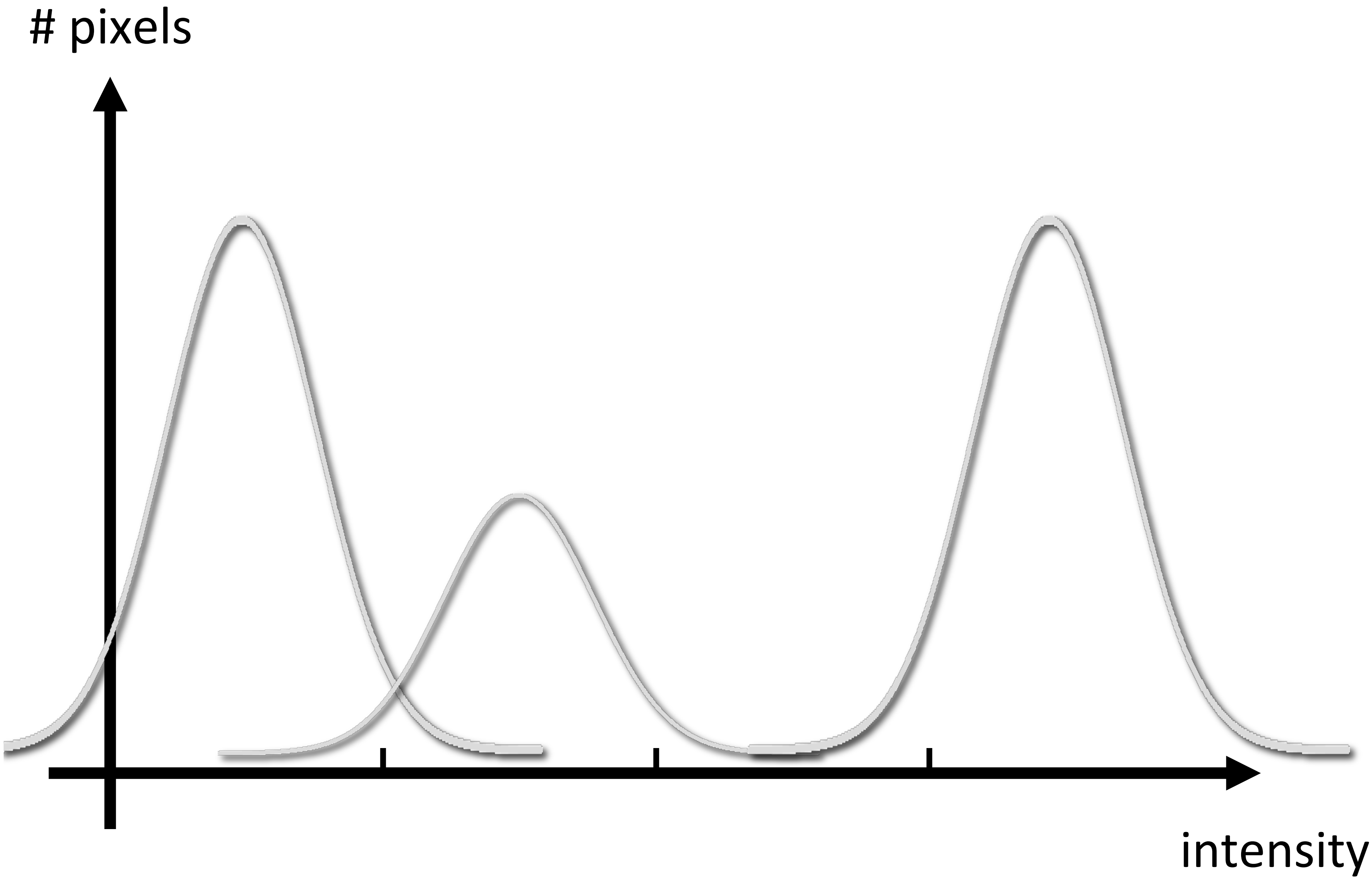
pixels
1
1
1
1
1
1
intensity



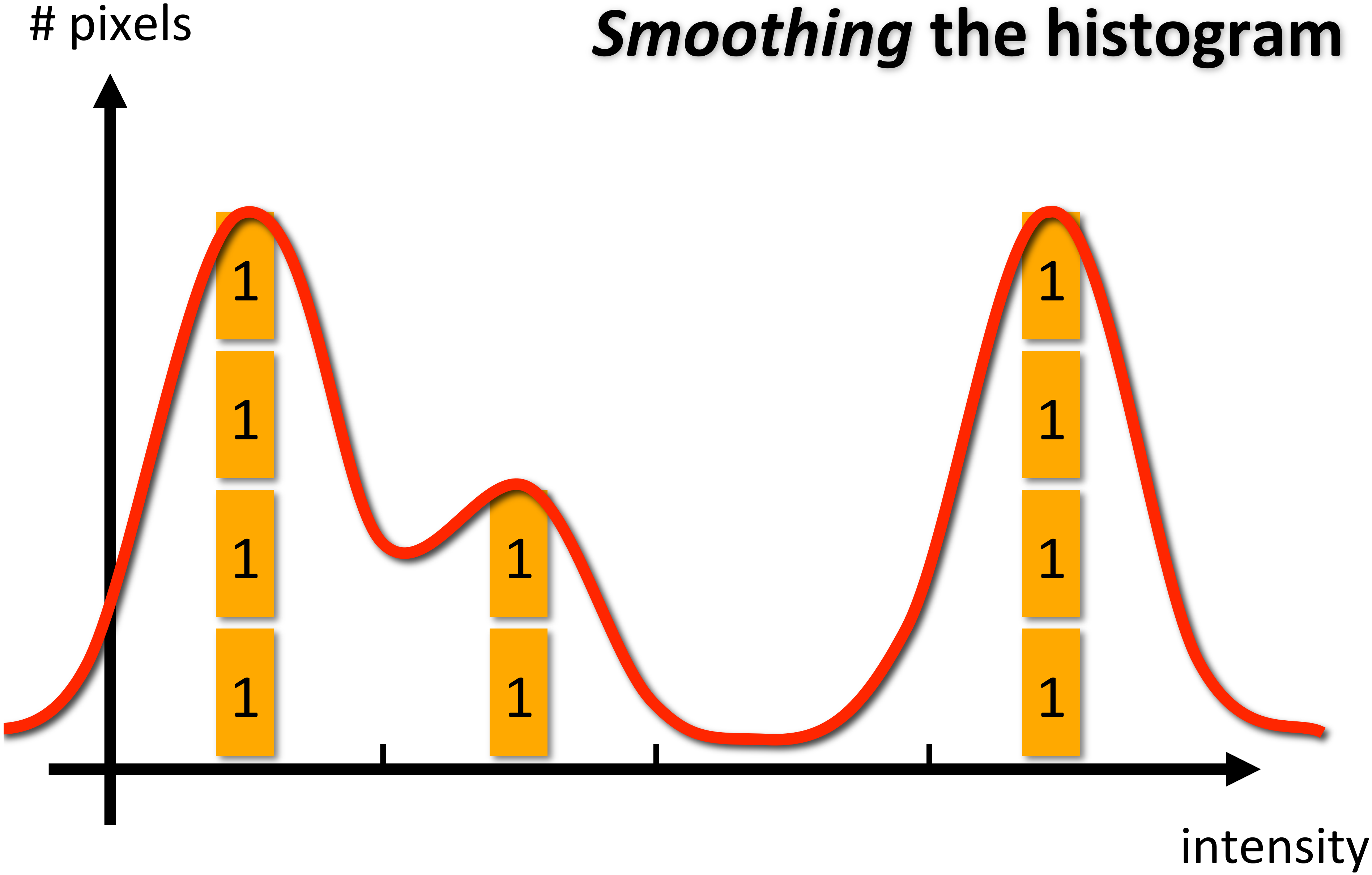
pixels
1
1
1
1
1
1
intensity



pixels
1
1
1
1
intensity

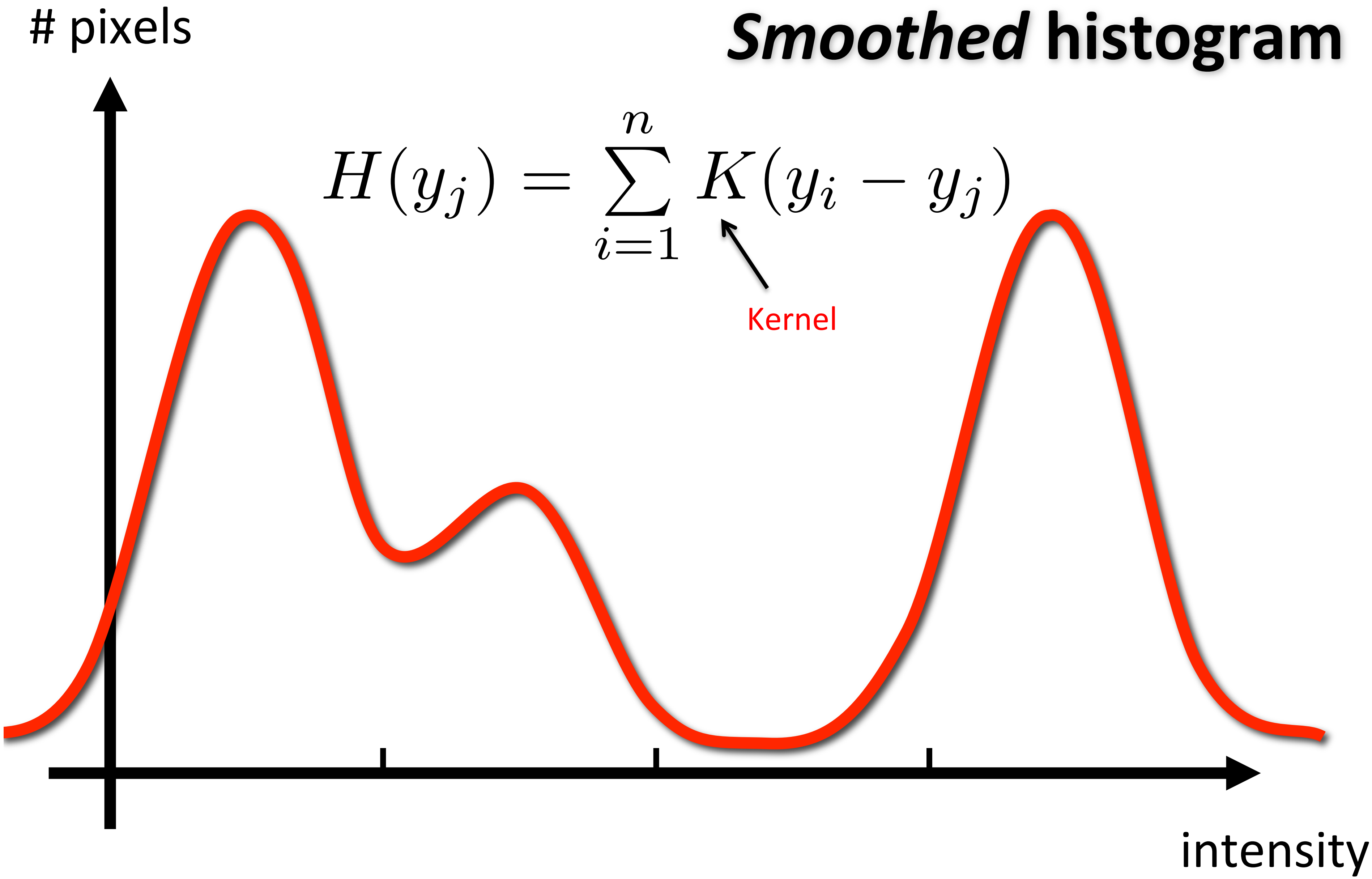

pixels
intensity

Smoothing the histogram



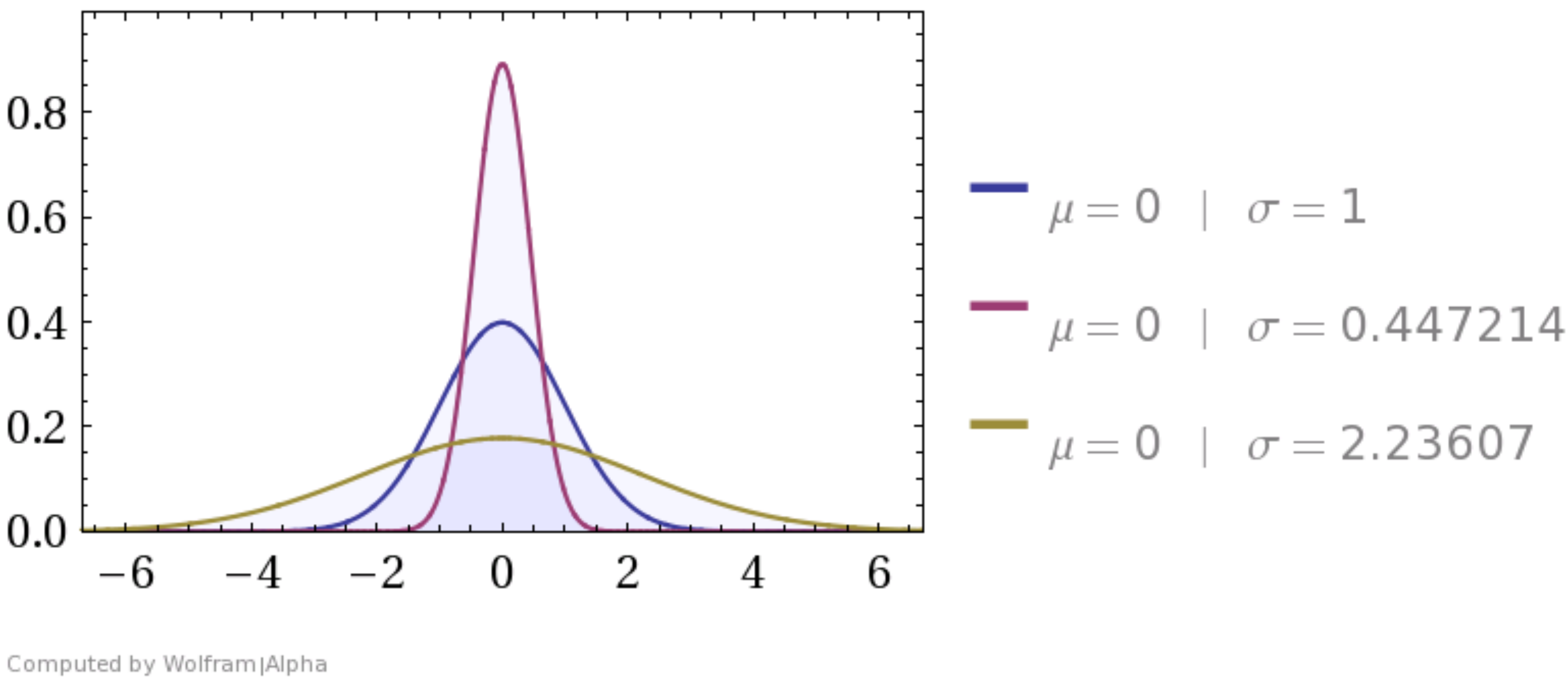
Smoothed **histogram**

$$H(y_j) = \sum_{i=1}^{n} K(y_i - y_j)$$

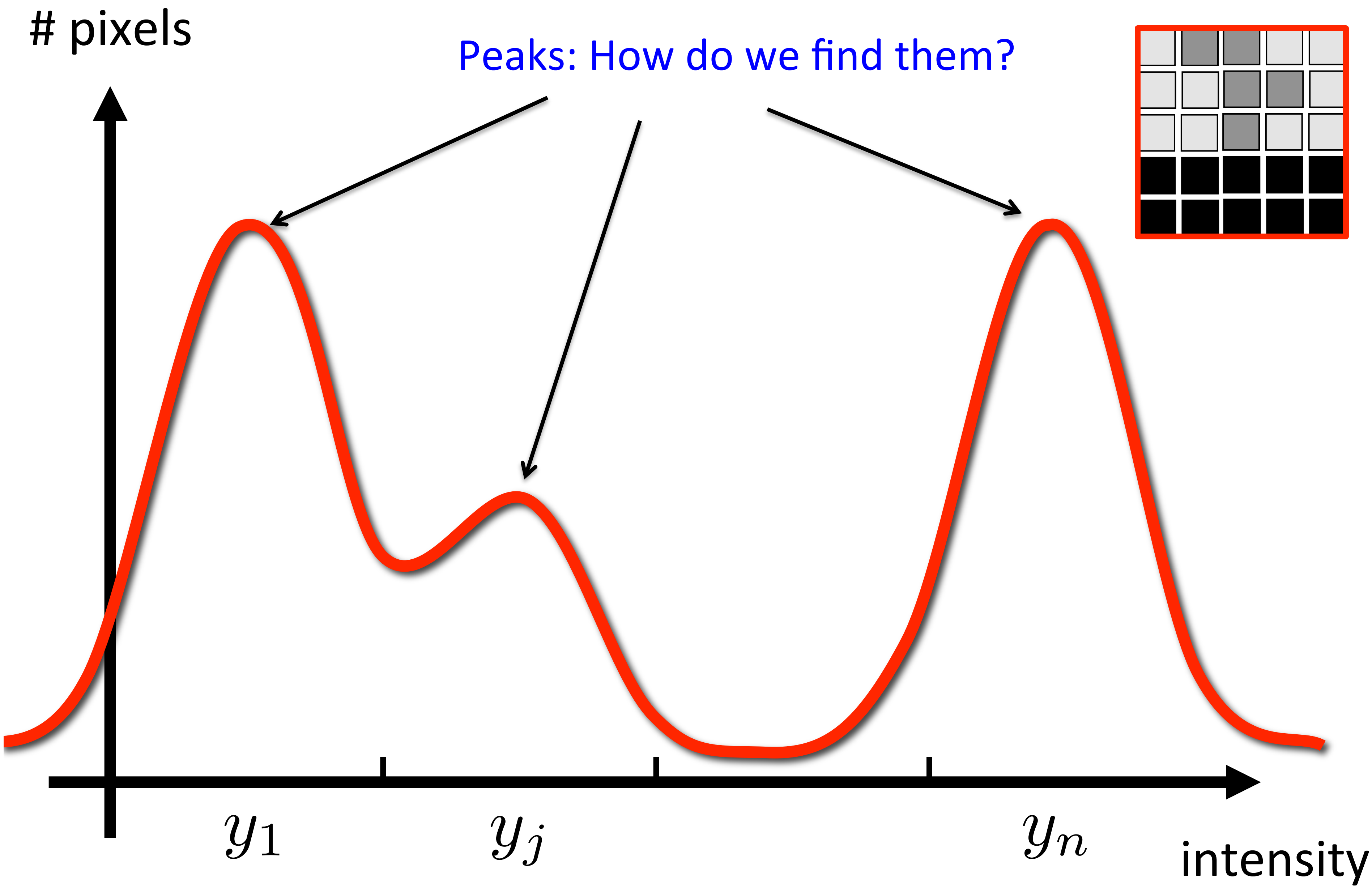


Choices for the kernel: Gaussian

$$K(y_i - y_j) = \exp(-|y_i - y_j|^2/\sigma^2)$$



Modes (peaks) of the Histogram



Mode Finding on the Histogram

Start with $H(y_j) = \sum_{i=1}^{n} K(y_i - y_j)$

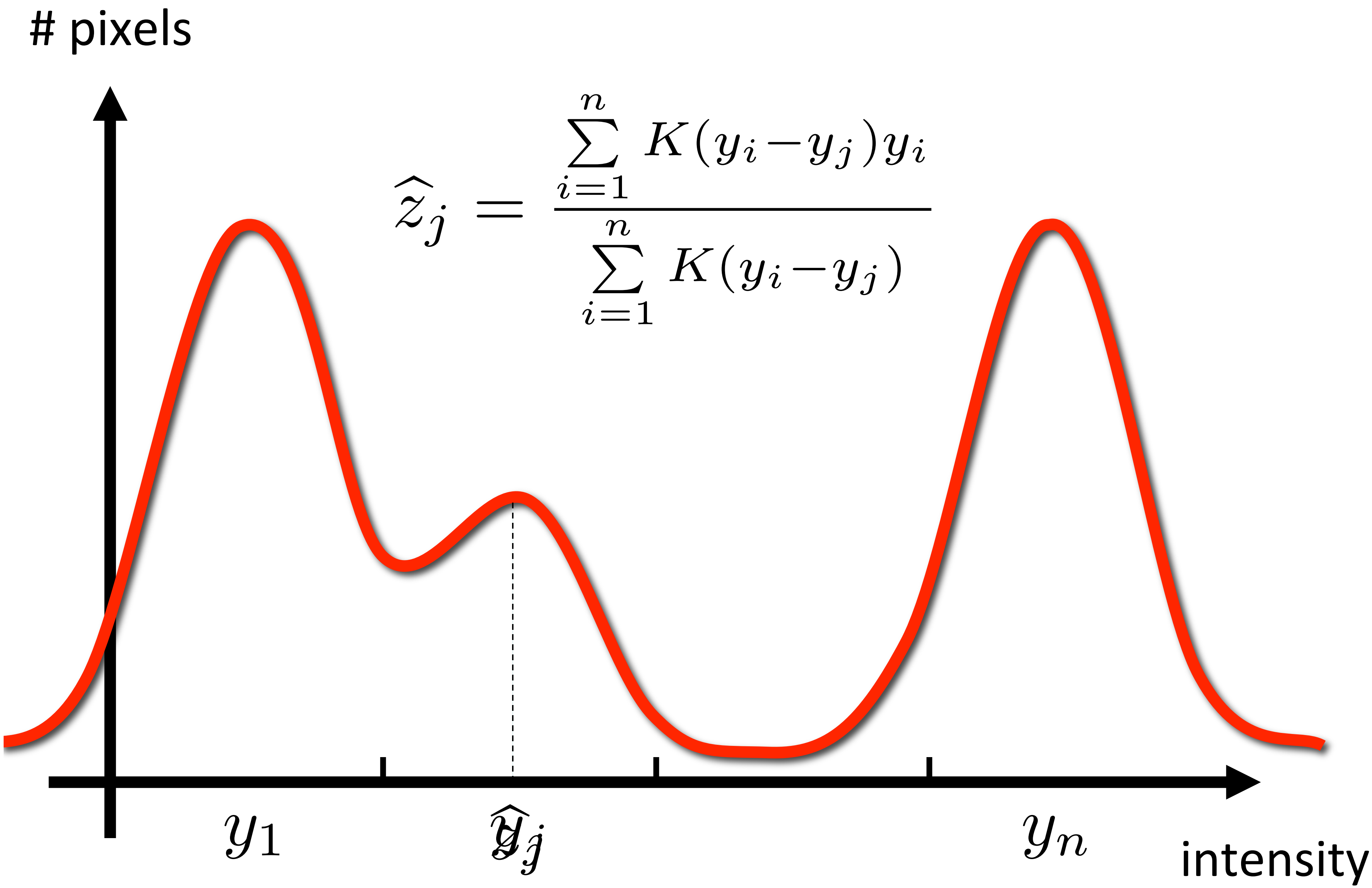
Fix $y_j = t$

Differentiate and solve: $\dfrac{\partial H(t)}{\partial t} = 0$

$$\widehat{z}_j = \frac{\sum_{i=1}^{n} K(y_i - y_j) y_i}{\sum_{i=1}^{n} K(y_i - y_j)}$$

(Assuming Gaussian Kernel, for now.)

Peak of the Histogram Nearest (in value) to y_j



Let's take a closer look

$$W_{ij} = \frac{K_{ij}}{\sum_i K_{ij}}$$

For each pixel, must normalize so the weights sum to 1.

$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$

Just a weighted average

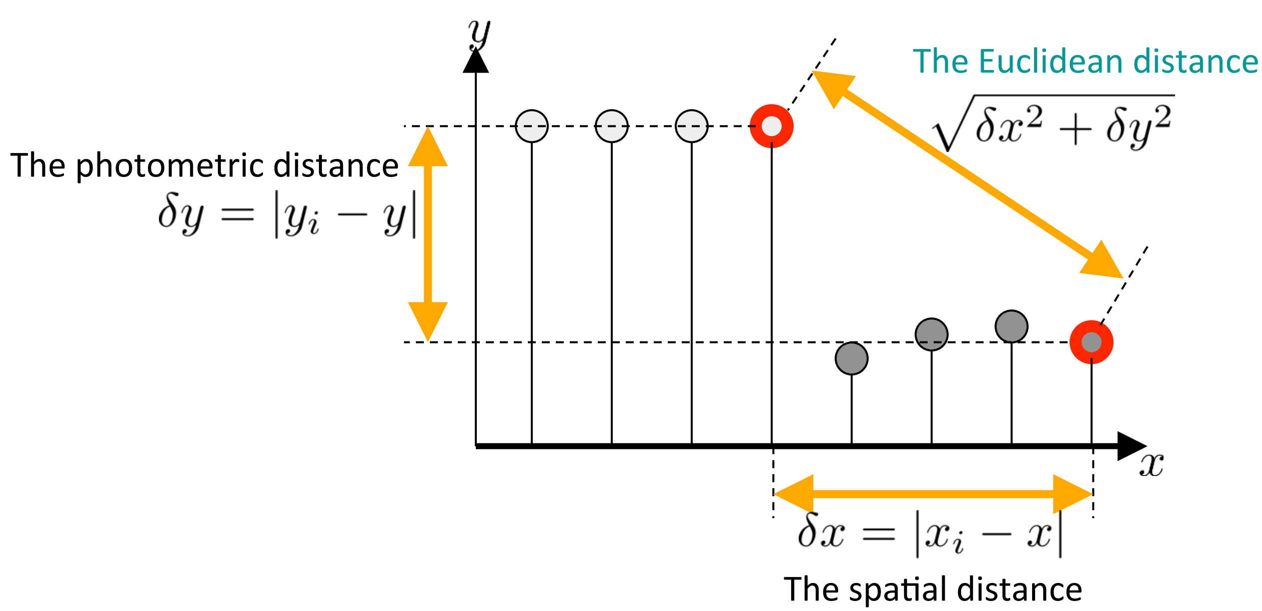
$$\sum_{i=1}^{n} W_{ij} = 1$$

Bilateral or non-local means kernels

Pixel values: $$K(y_i - y_j) = \exp(-|y_i - y_j|^2/h_y^2)$$

Pixel positions: $$K(x_i - x_j) = \exp(-|x_i - x_j|^2/h_x^2)$$

$$K_{ij} = K(y_i - y_j)\, K(x_i - x_j)$$



Faster Normalized Filtering

$$\widehat{z}_j = \frac{\sum_{i=1}^{n} K_{ij} y_i}{\sum_{i=1}^{n} K_{ij}}$$

$$\widehat{z}_j = \frac{\boxed{\sum_{i=1}^{n} K_{ij} y_i}}{\boxed{\sum_{i=1}^{n} K_{ij} \; 1}}$$

Filter once

Filter twice

$$\widehat{z}_j = \frac{\sum_{i=1}^{n} K_{ij} y_i}{d_j}$$

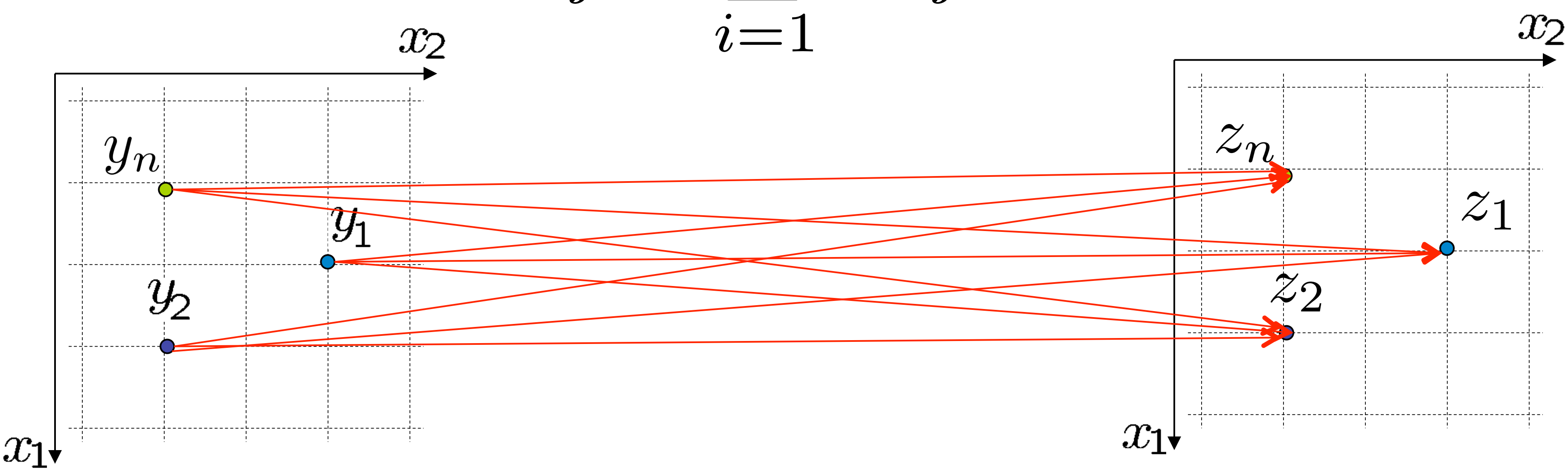
Divide Pixel-wise

The Black Box

Input image

$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$

Output image



Global (or Local) Filters

All input pixels

Each output pixel $\longrightarrow$ $$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$

$$\mathbf{w}_j^T = [W_{1j},\, W_{2j},\, \cdots,\, W_{nj}]$$

rows

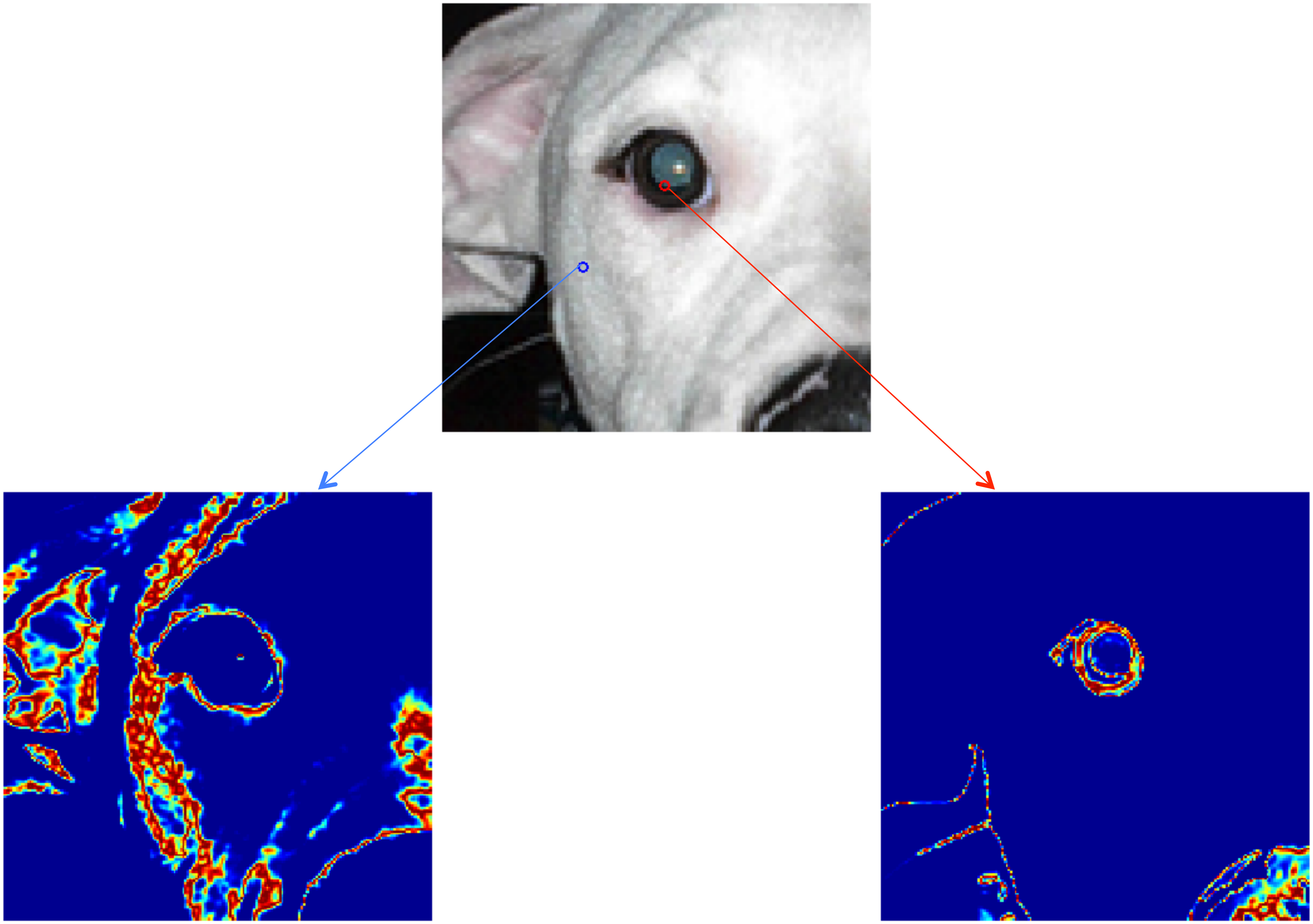
$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{bmatrix}$$

$$\widehat{\mathbf{z}} = \mathbf{W}\ \mathbf{y}$$

Data-dependent matrix

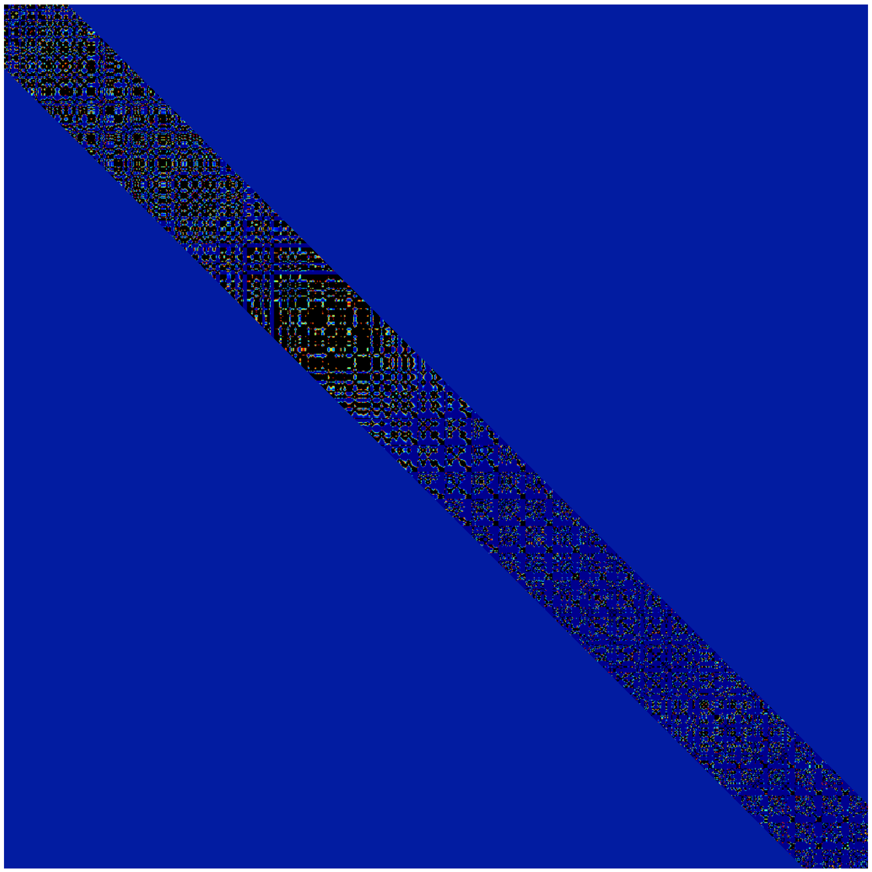
Image scanned into a vector

Global Affinities



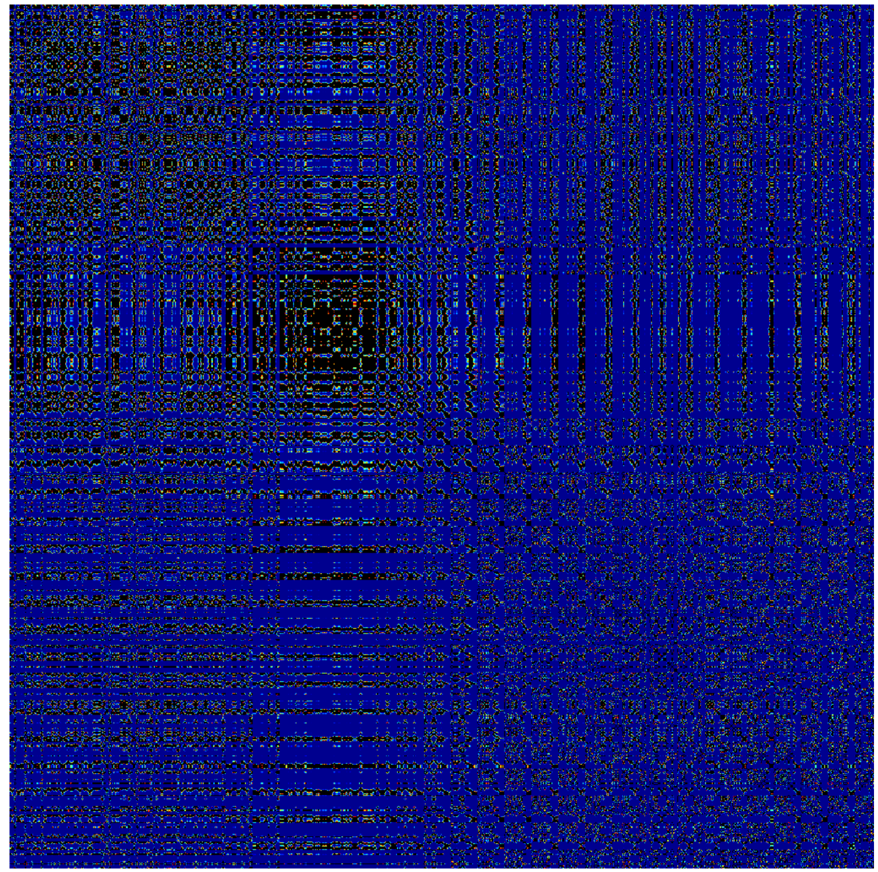
Local vs. Global Filter Matrix W

Local Filters



Sparse, but high-rank

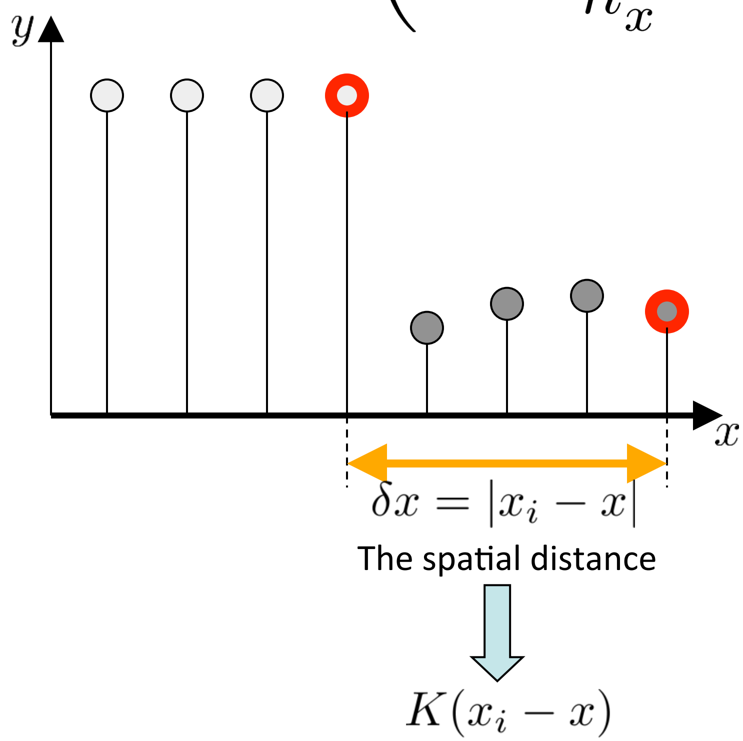
Global Filters

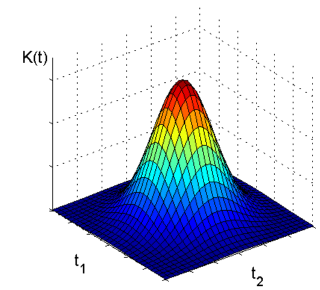


Dense but low-rank

Choice of Kernels

$$K(x_i - x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{h_x^2}\right)$$

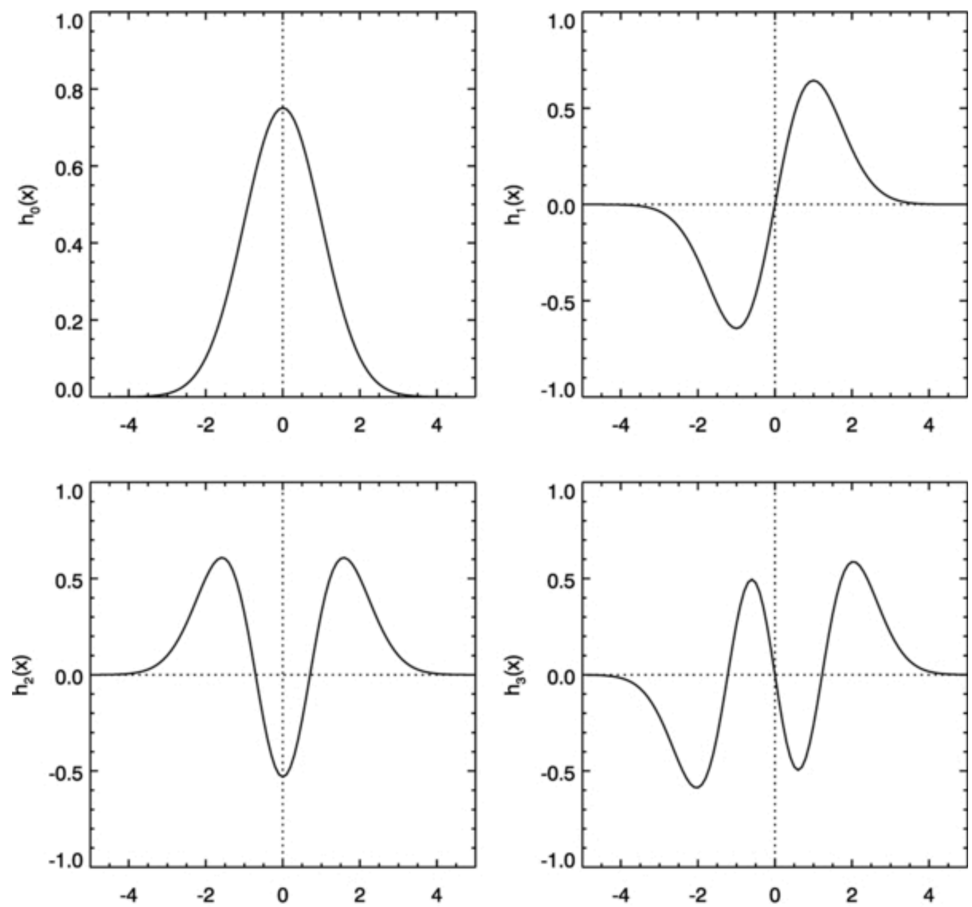




- Classical <u>Gaussian</u> <u>Linear</u> Filters

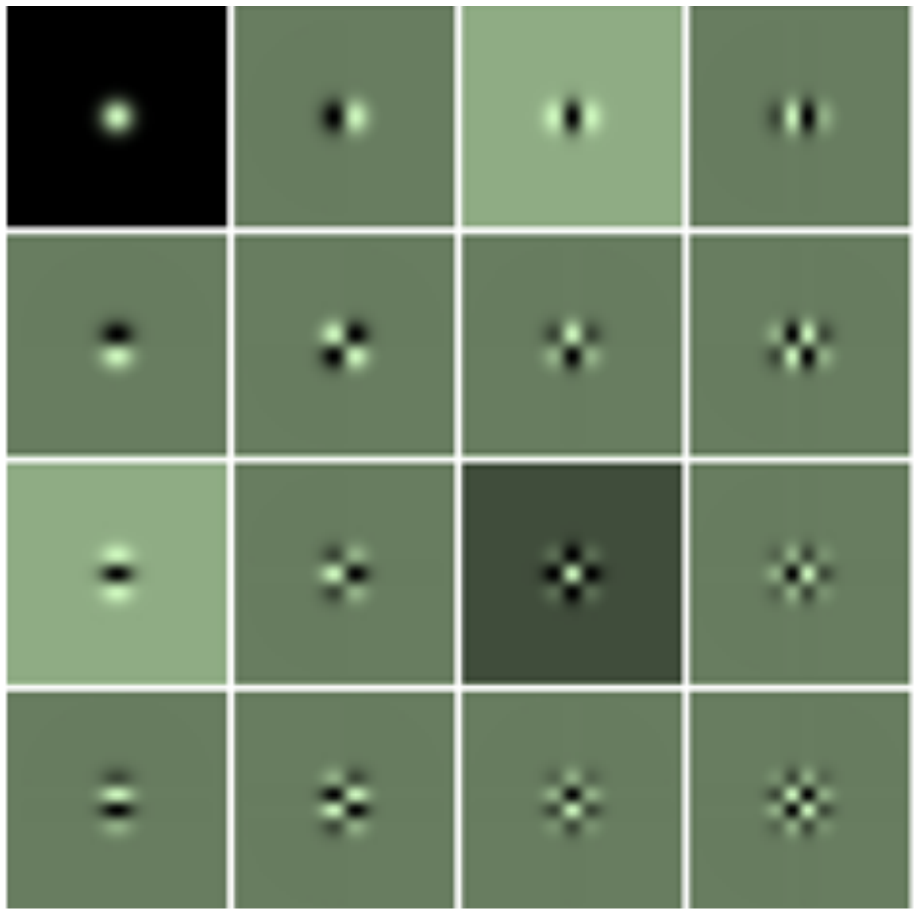
Eigenvectors of Linear Gaussian Kernel

Hermite-Gauss functions

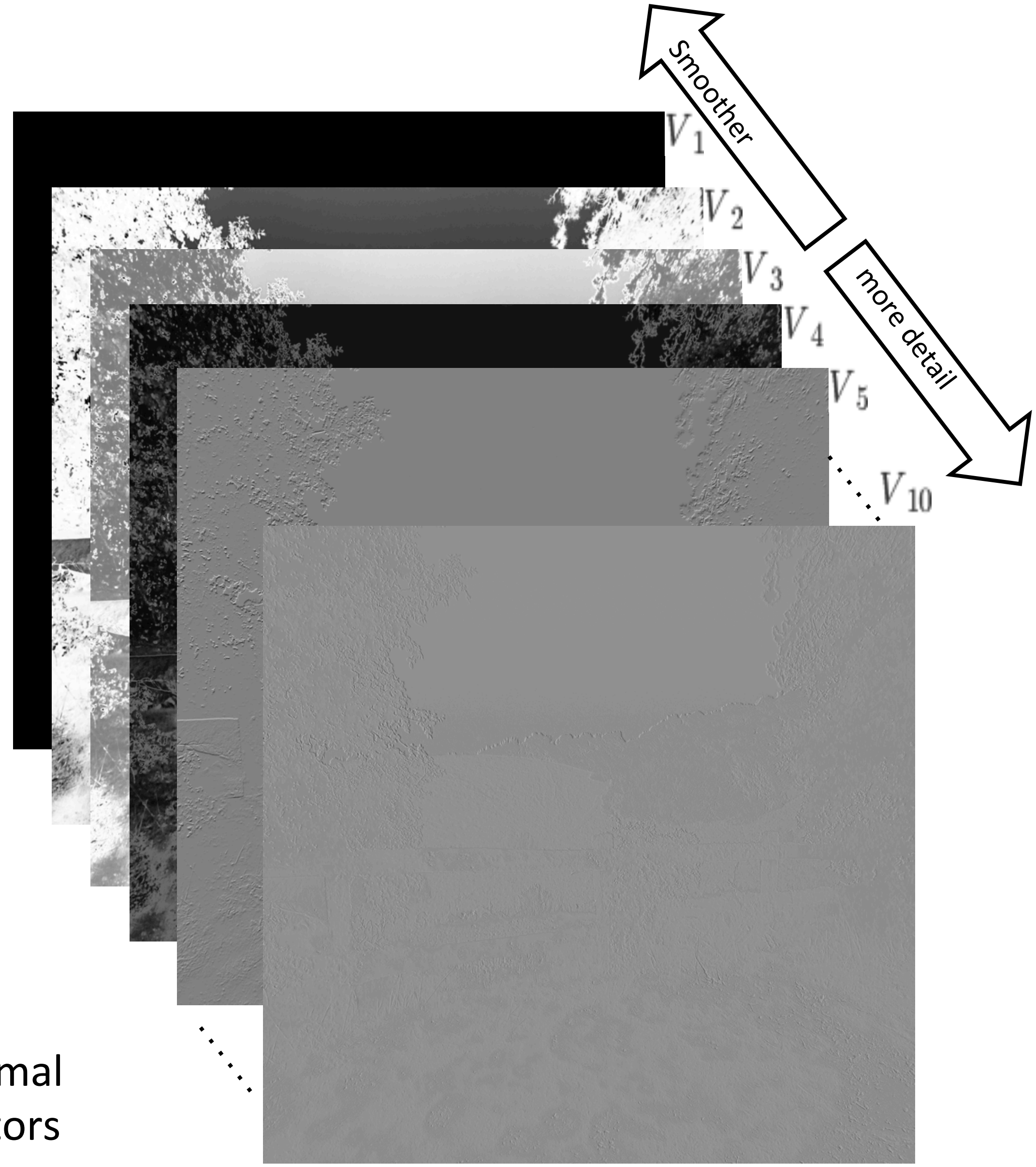


Eigenvectors of Linear Gaussian Kernel

Hermite-Gauss functions

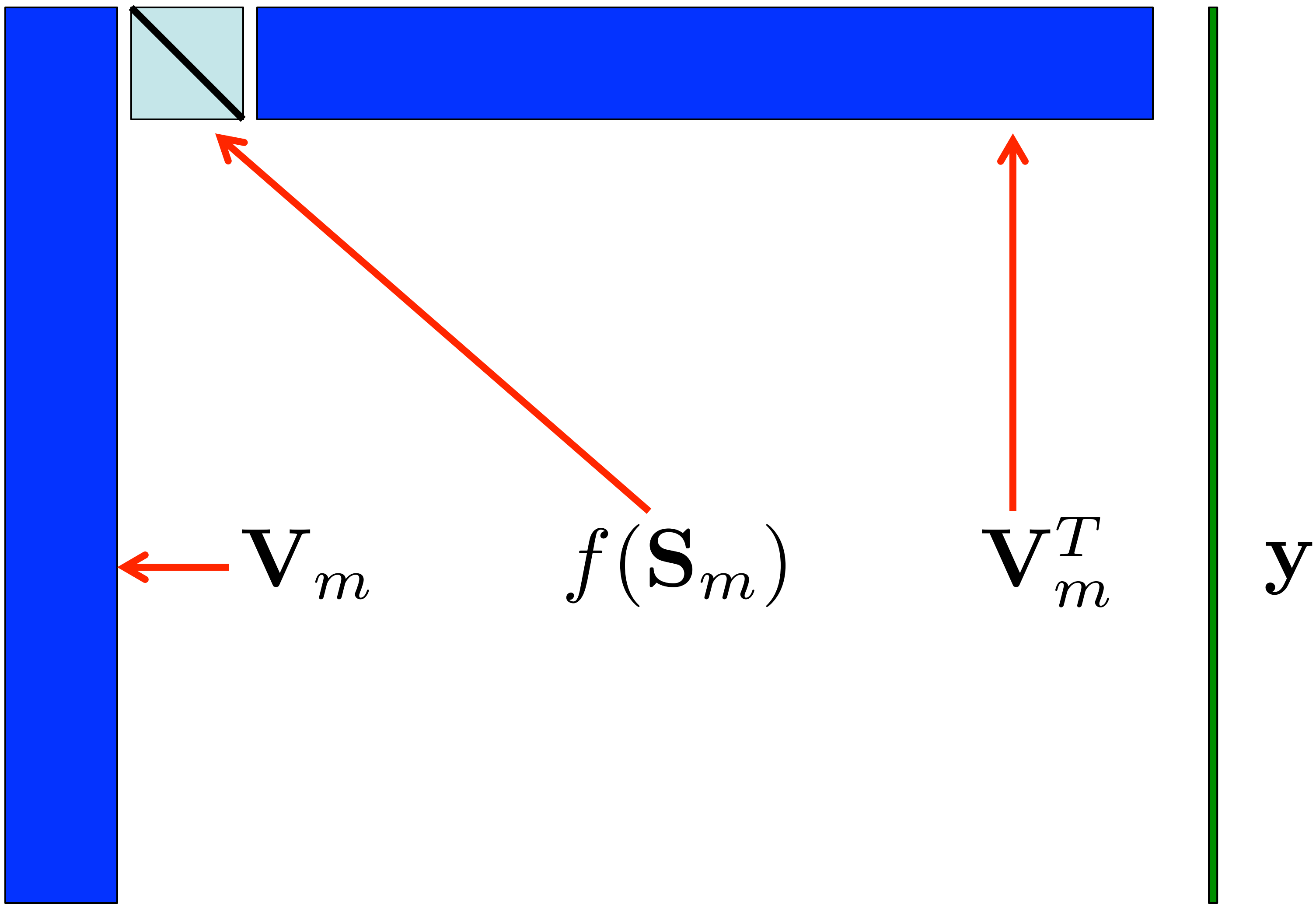


Filter Eigenvectors



Orthonormal
eigenvectors

Spectral Filtering in Lower Dimension



Low-light imaging

Denoised/Sharpened


Local Laplacian Filter

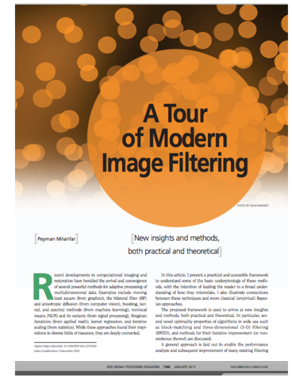
Multi-frame Upscaling/Zoom


6FEN596

Fused 4 images, denoised and upscaled by 4x


6FEN5
6FEN5

Relevant Papers



- “A Tour of Modern Image Filtering”, P. Milanfar, IEEE Signal Processing Magazine, no. 30, pp. 106–128, Jan. 2013
- “A General Framework for Regularized, Similarity-based Image Restoration”, A. Kheradmand, and P. Milanfar, IEEE Trans on Image Processing, vol. 23, no. 12, Dec. 2014
- “Global Image Denoising”, H. Talebi, and P. Milanfar, IEEE Trans on Image Processing, vol. 23, no. 2, pp. 755-768, Feb. 2014
- “Nonlocal Image Editing”, H. Talebi, and P. Milanfar, IEEE Trans on Image Processing, vol. 23, no. 10, Oct. 2014

http://milanfar.org