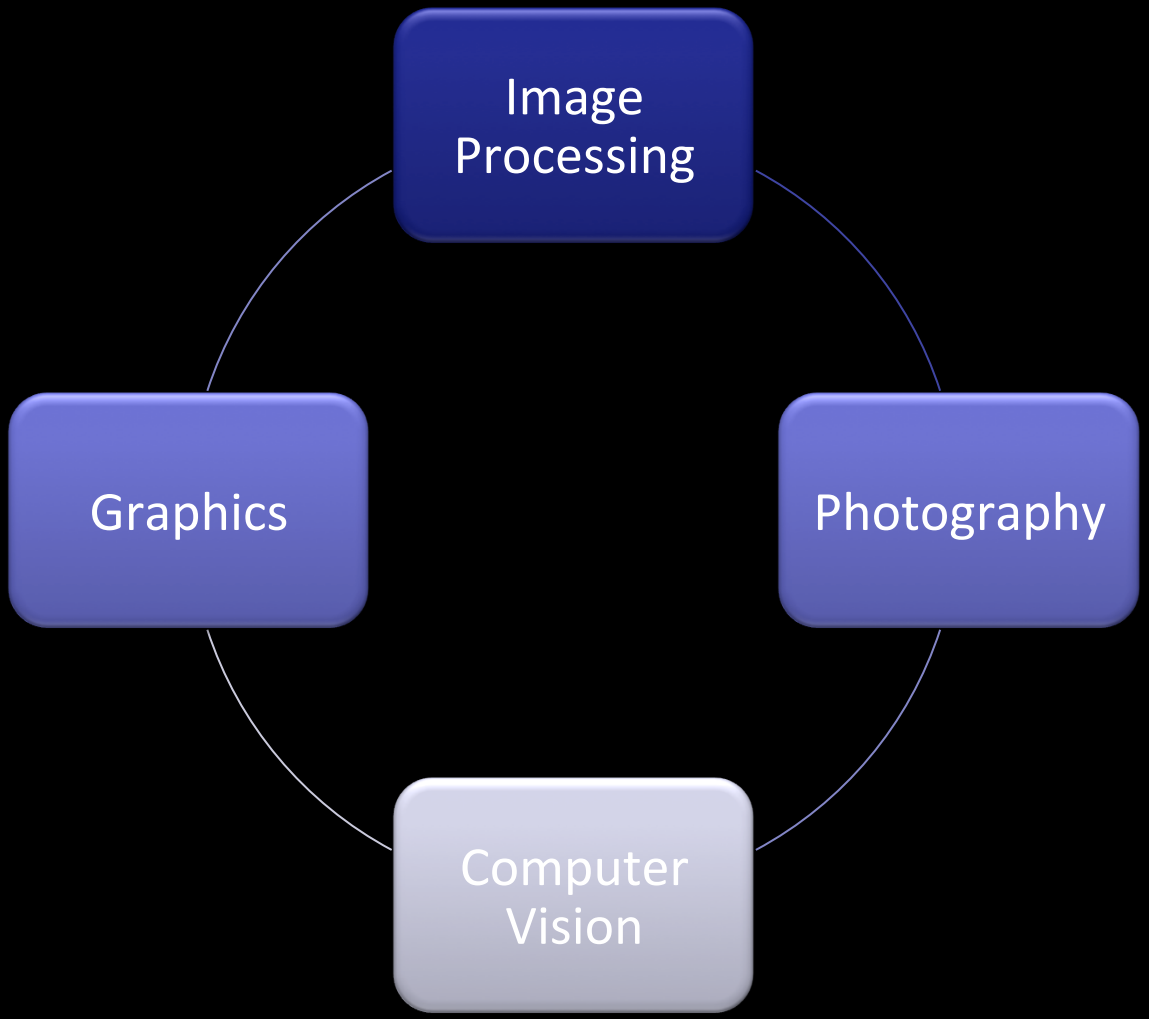
Computational Imaging:

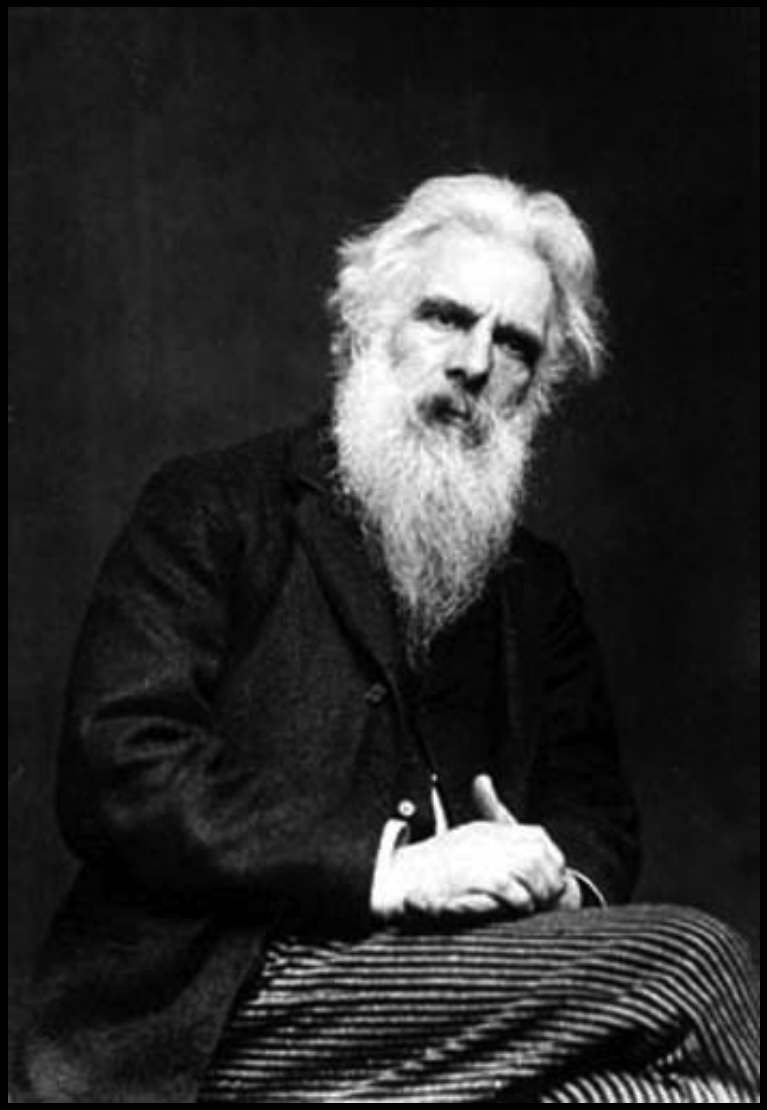
From Photons to Photos

Peyman Milanfar

What is Computational Imaging?

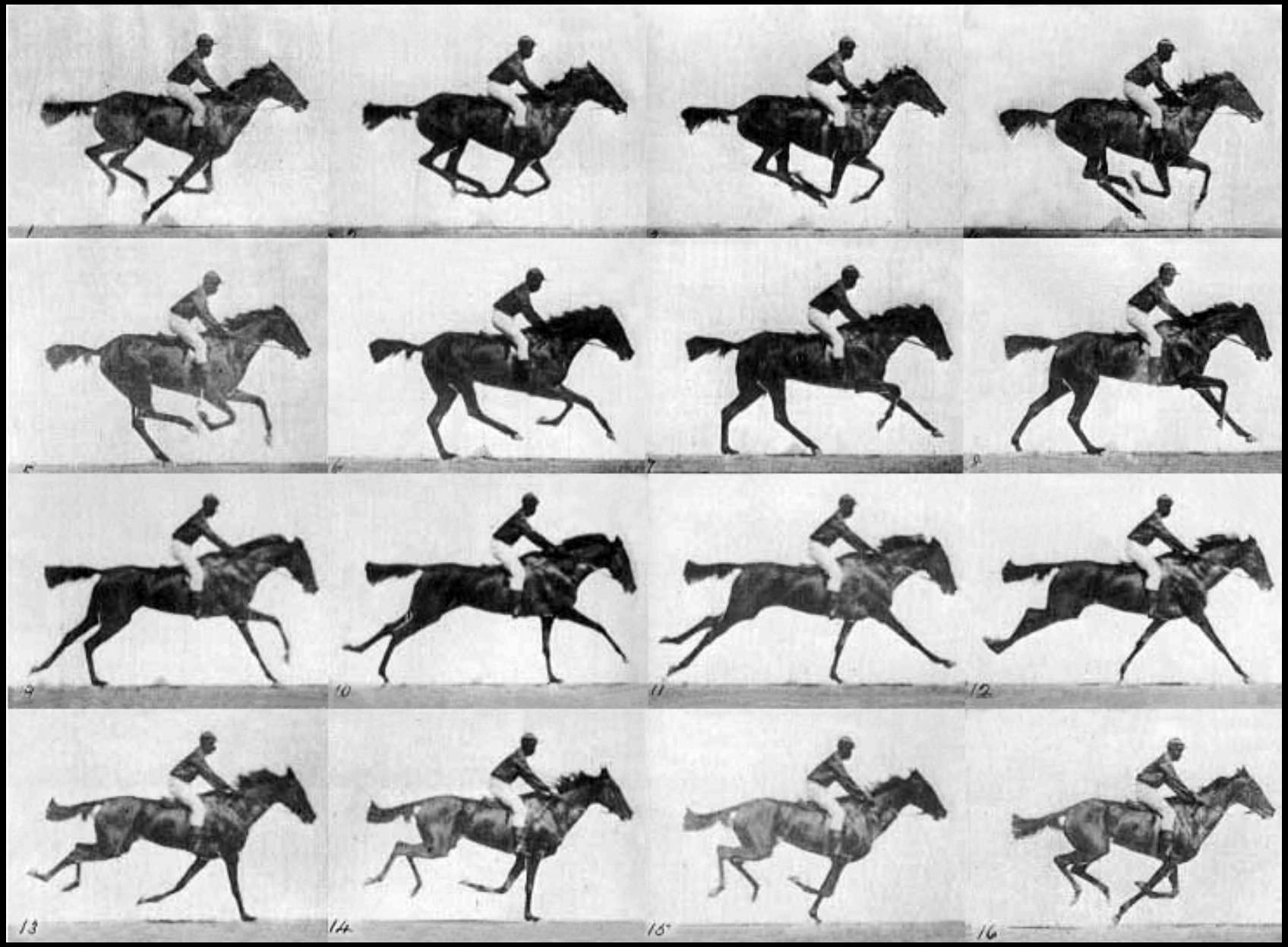


Long and fascinating history

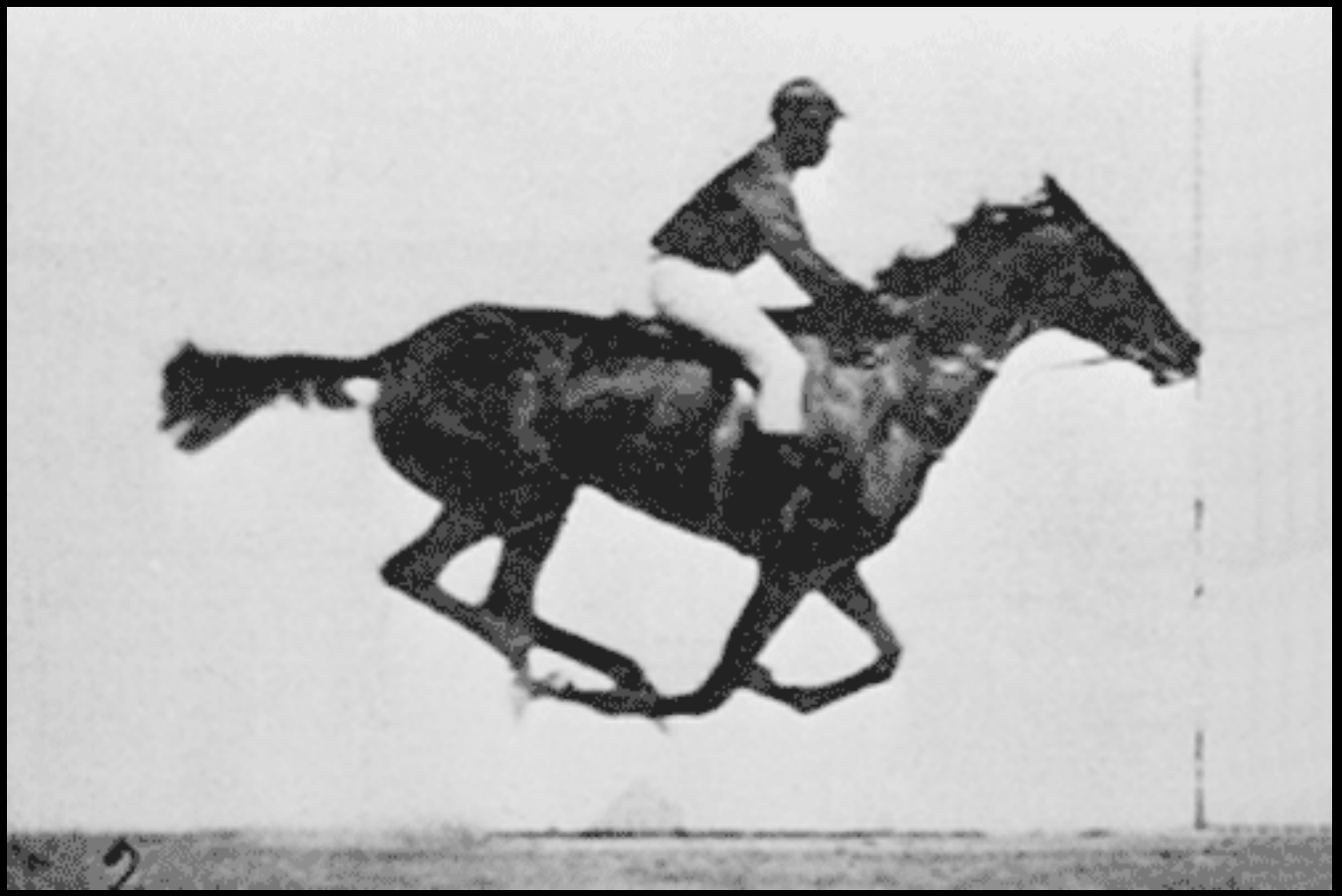


Eadweard Muybridge 1830 – 1904

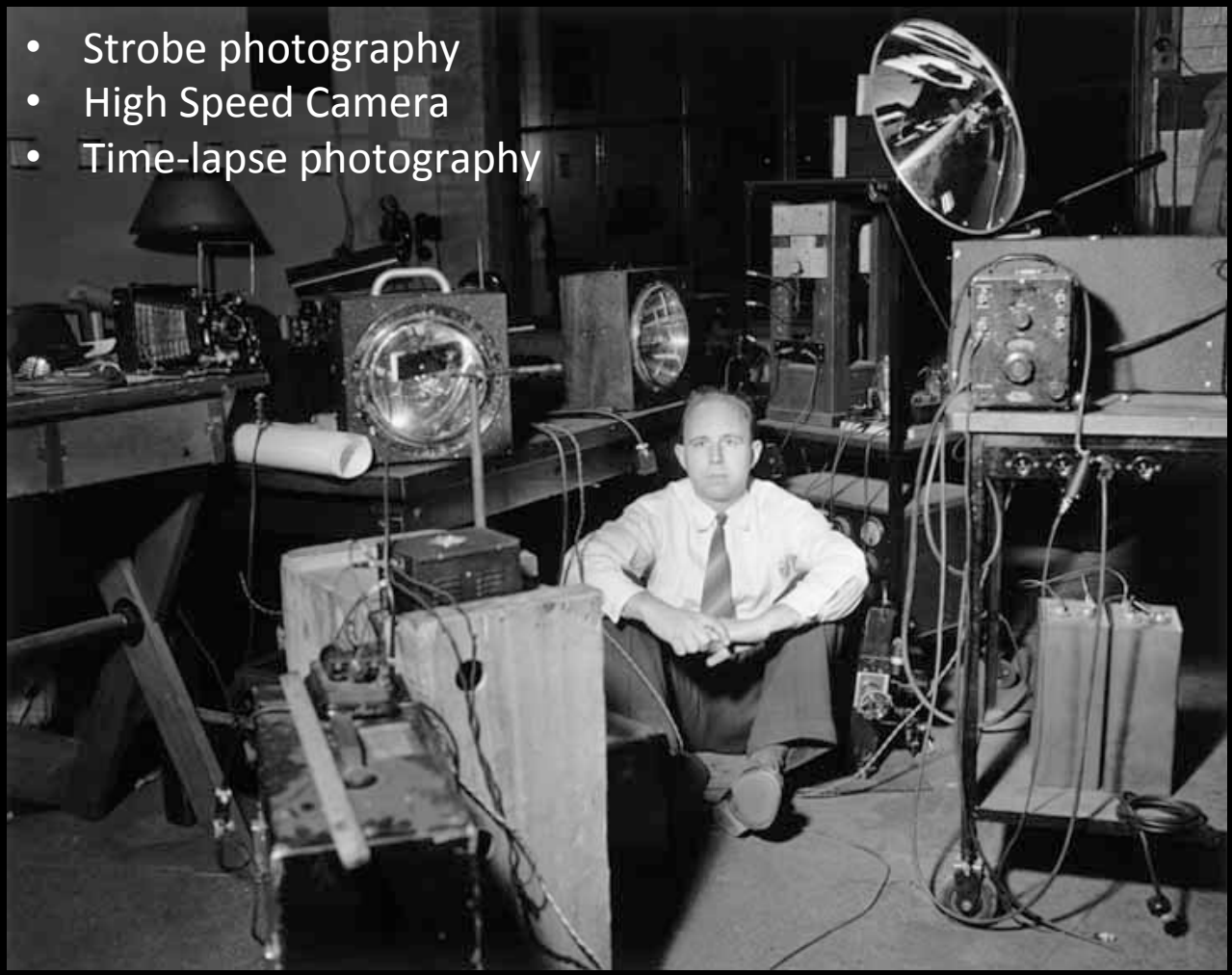
Galloping Horse 1878



GIF made in 2006

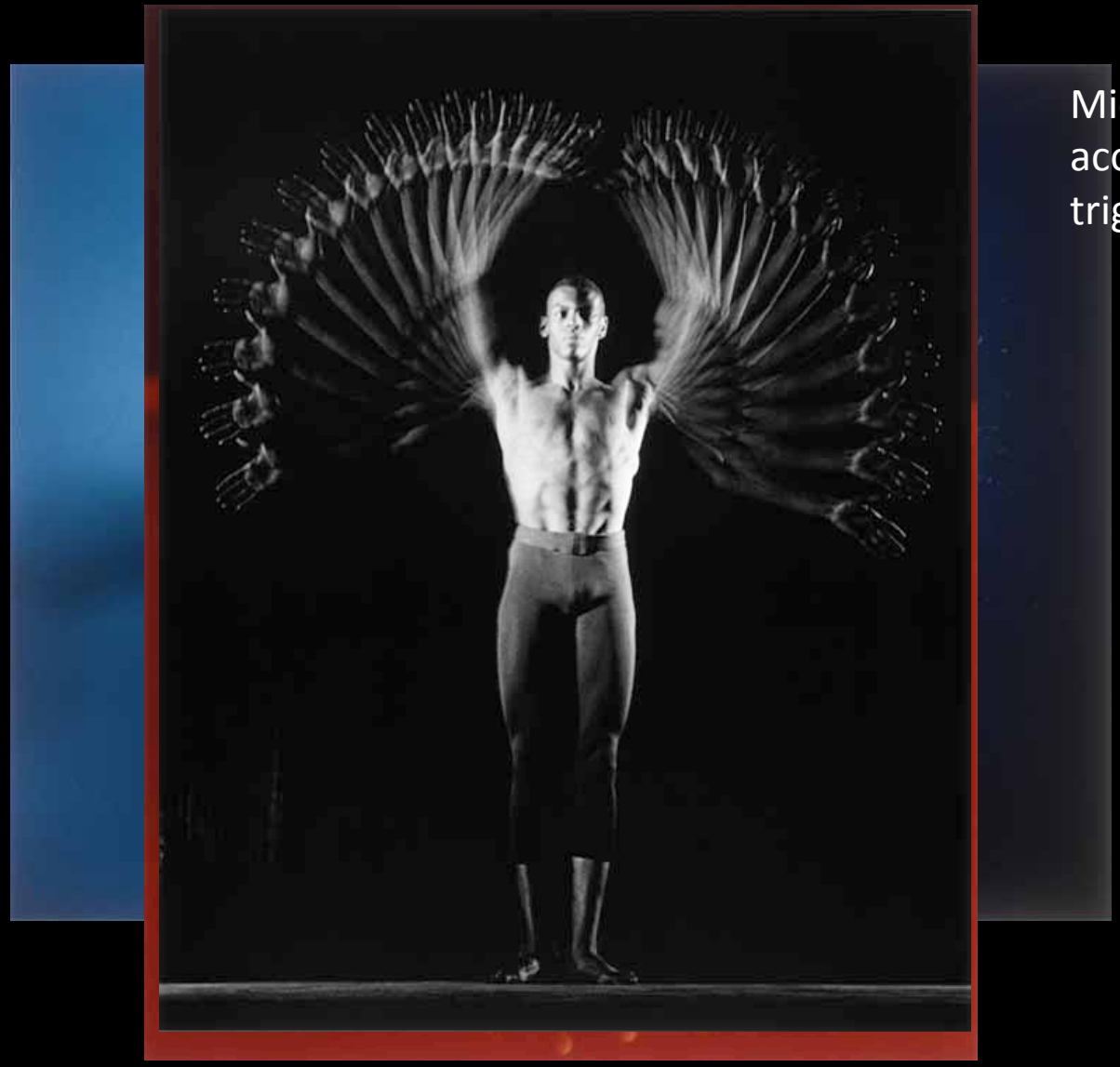


Another Pioneer



Harold “Doc” Edgerton (1903 – 1990)

“Stopping Time”



Microphone picks up acoustic shock wave, triggers the flash.

Shot in Darkness, with sync-strobe, exposure time ~ 10^{-7} s

1980s and 90s

- HDR
- wearable computers
- super-resolution
- panorama stitching

Not Computational Imaging

The Modern Era of Easy Photography

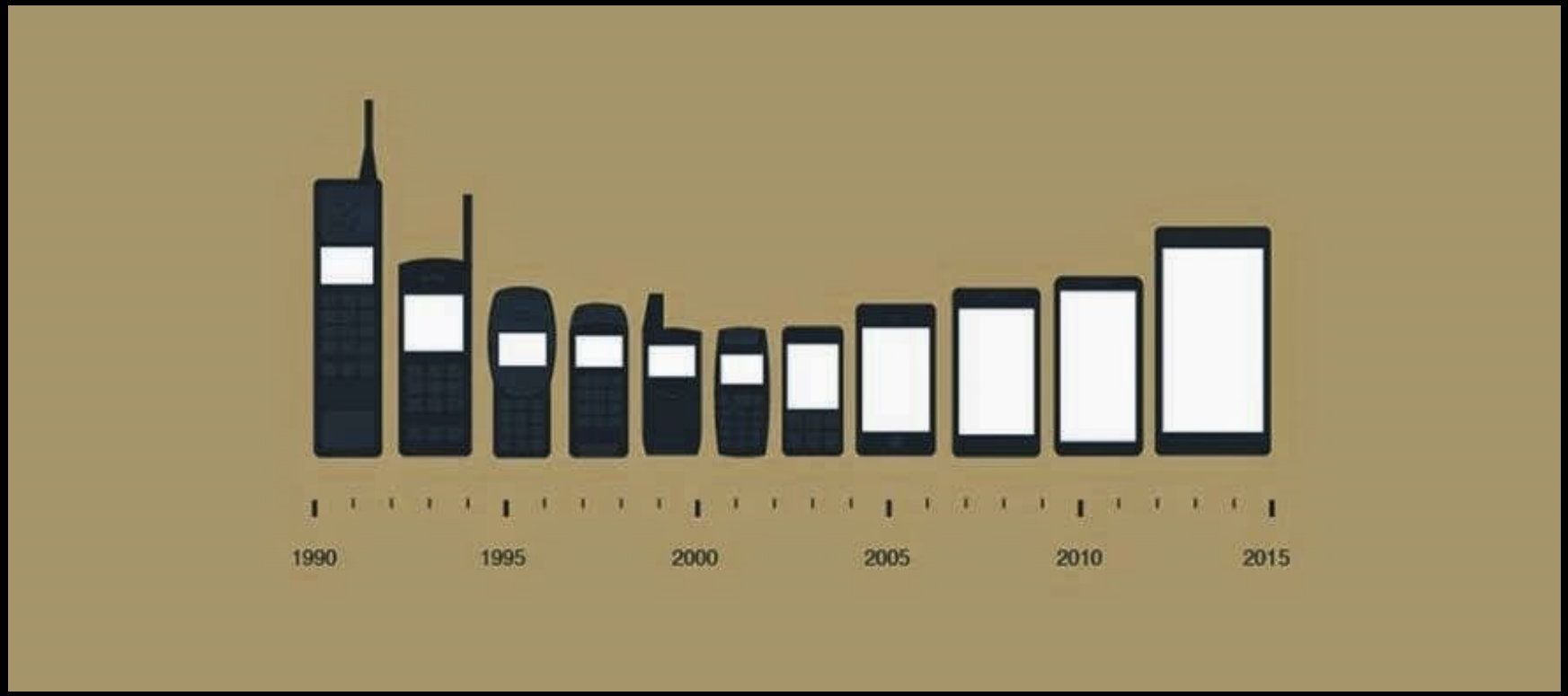
Take it

See it

Share it

Evolution of (cheap) Photography

Evolution of the (Smart) phone




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

2015 -- Paris

2015 -- Moscow

2015 -- Sydney

1,000,000,000,000

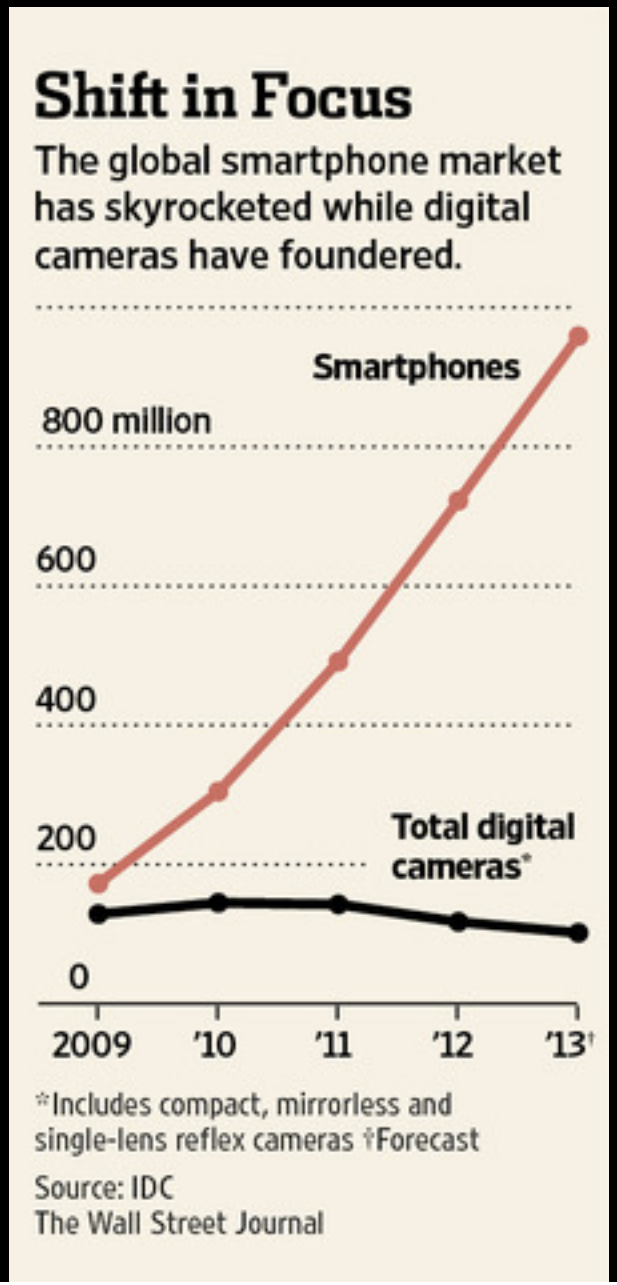
Roughly a trillion photos *shared* on social media in 2014

7,000,000,000

As many as seven billion were “selfies”

Reuters/CBS/TIME

Mobile Imaging is Totally Dominant



Dual Aims of (Computational) Imaging

Capture what I see (Photography)

- Take a nice picture. (1870s -)
- Make me a better photographer. (1970s -)
- Do it with my simple camera. (2000s -)

Let's have a look

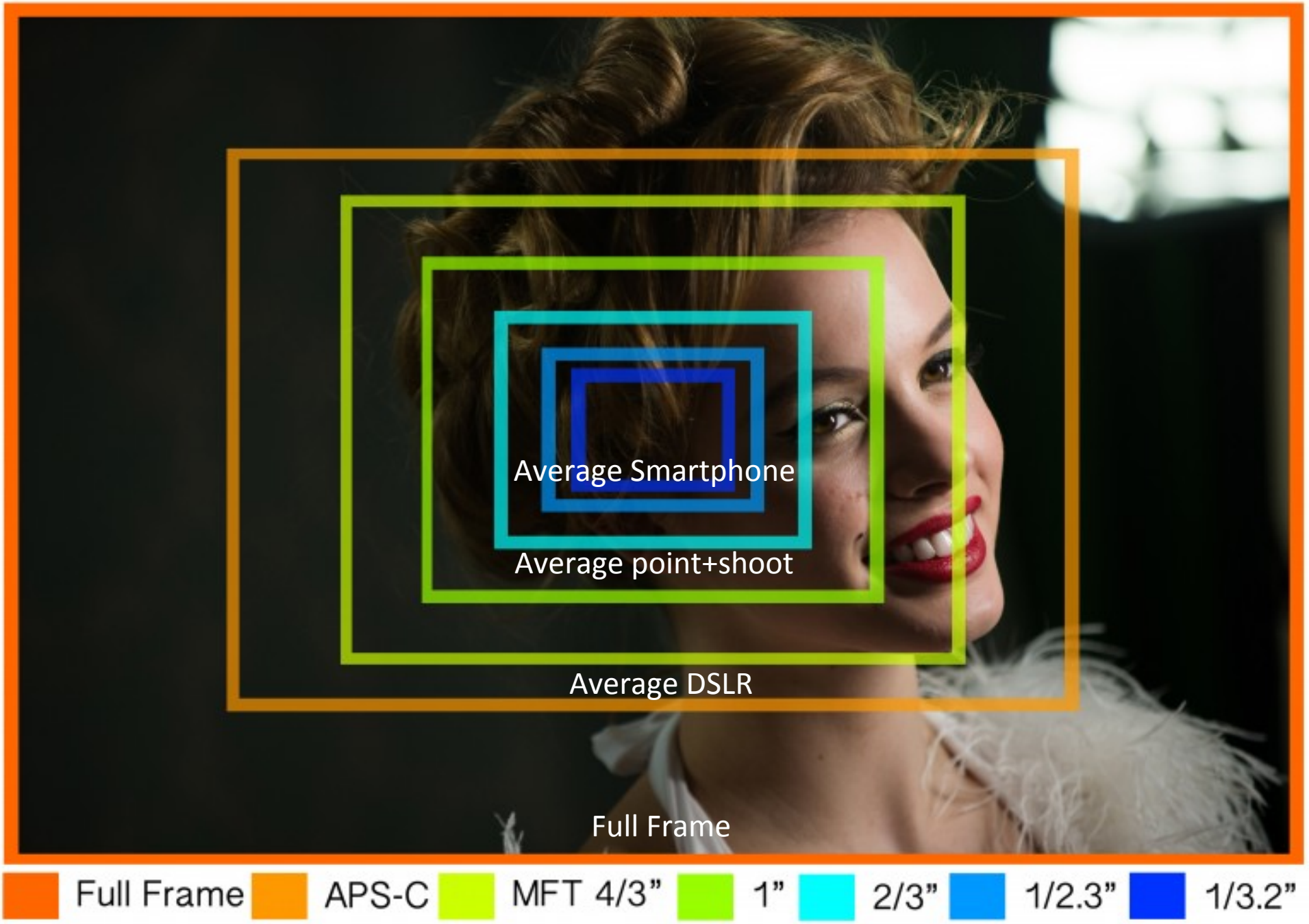
Sensors: Form vs. Function

$500

$5

Mobile phone camera

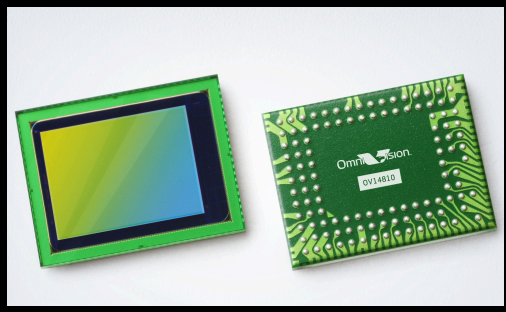
Can these two ever be equally good at taking pictures?


Average Smartphone
Average point+shoot
Average DSLR
Full Frame
Full Frame
APS-C
MFT 4/3”
1”
2/3”
1/2.3”
1/3.2”

gizmag

Where are the opportunities?

- Optics won't get a lot better.
- Pixels can't get much smaller.
  - Limited by optical wavelengths
- Sensor won't get much larger (in mobile)
  - Limited by cost and form factor
- Need:
  - Better Capture Protocols
  - Clever Algorithms



Burst Photography

Locked Exposure
& Focus

HDR Burst Photography

Locked Focus
Variable Exposure

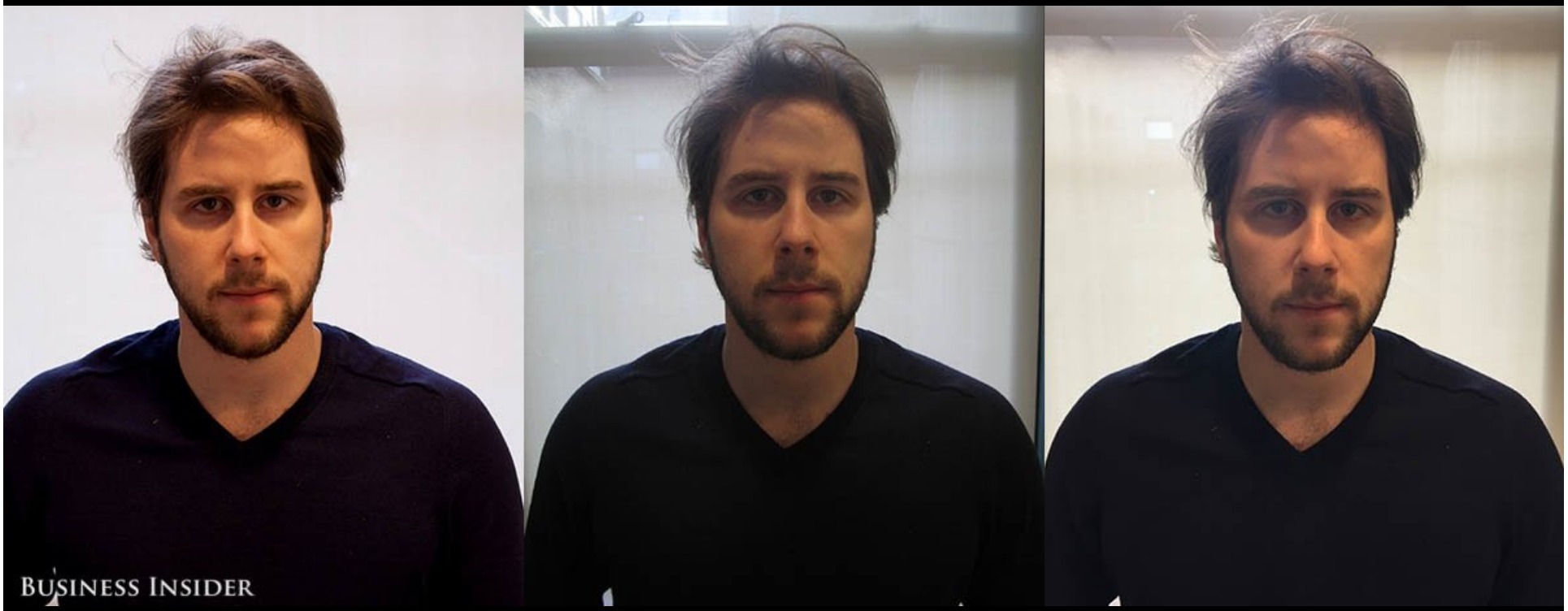
Google Glass

Both are from Nexus 5

Standard shot

With Lens Blur

Credit: Sascha Haeberling


BUSINESS INSIDER

Dual Aims of Computational Imaging

Capture what I see (Digital 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)

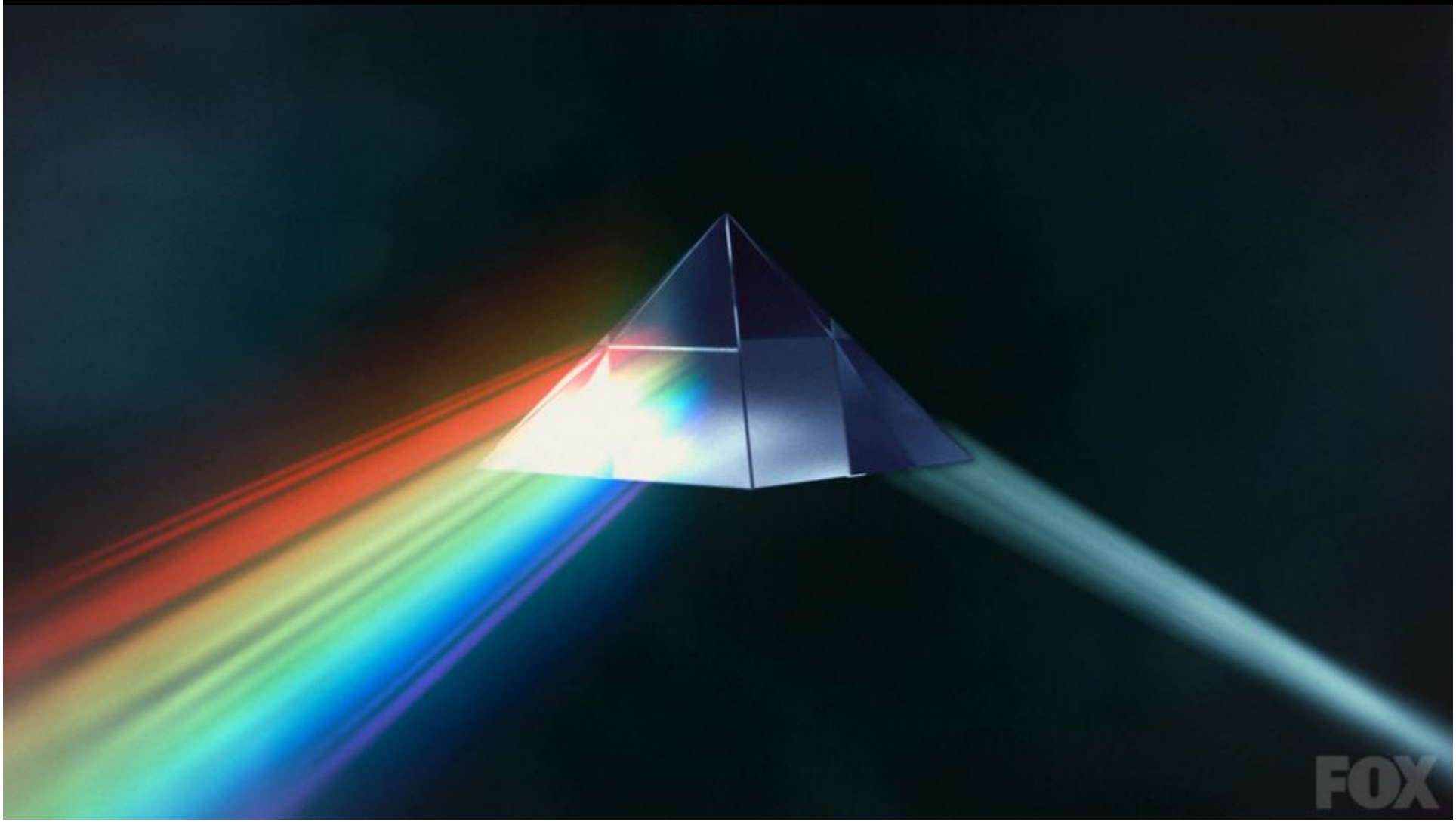
- See in the dark
- View fast, slow, small, or faint phenomena
- Detect, magnify subtle changes

Yosemit Valley, California

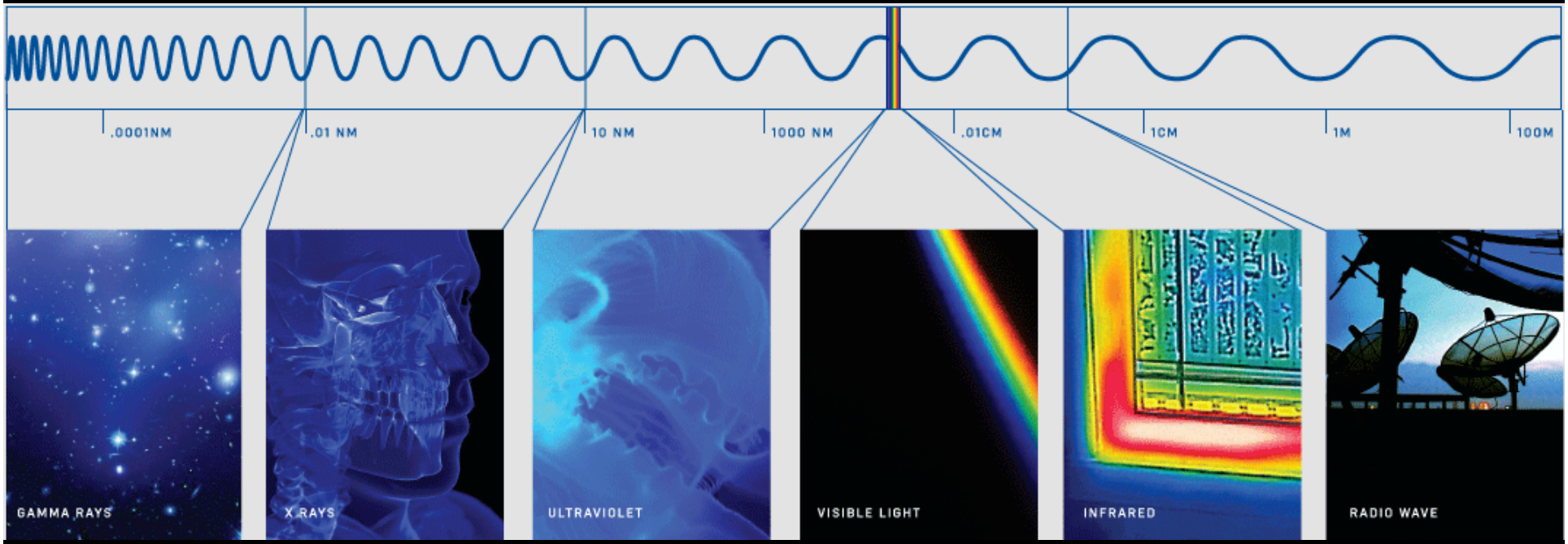
At night

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

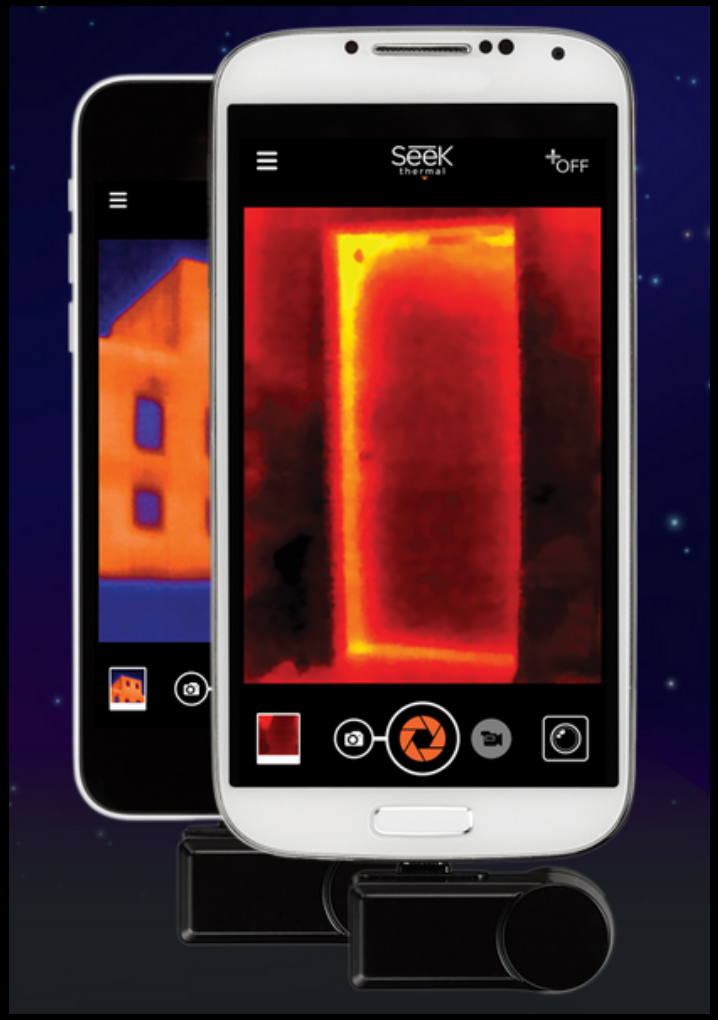
Can’t “see” most of nature that surrounds us

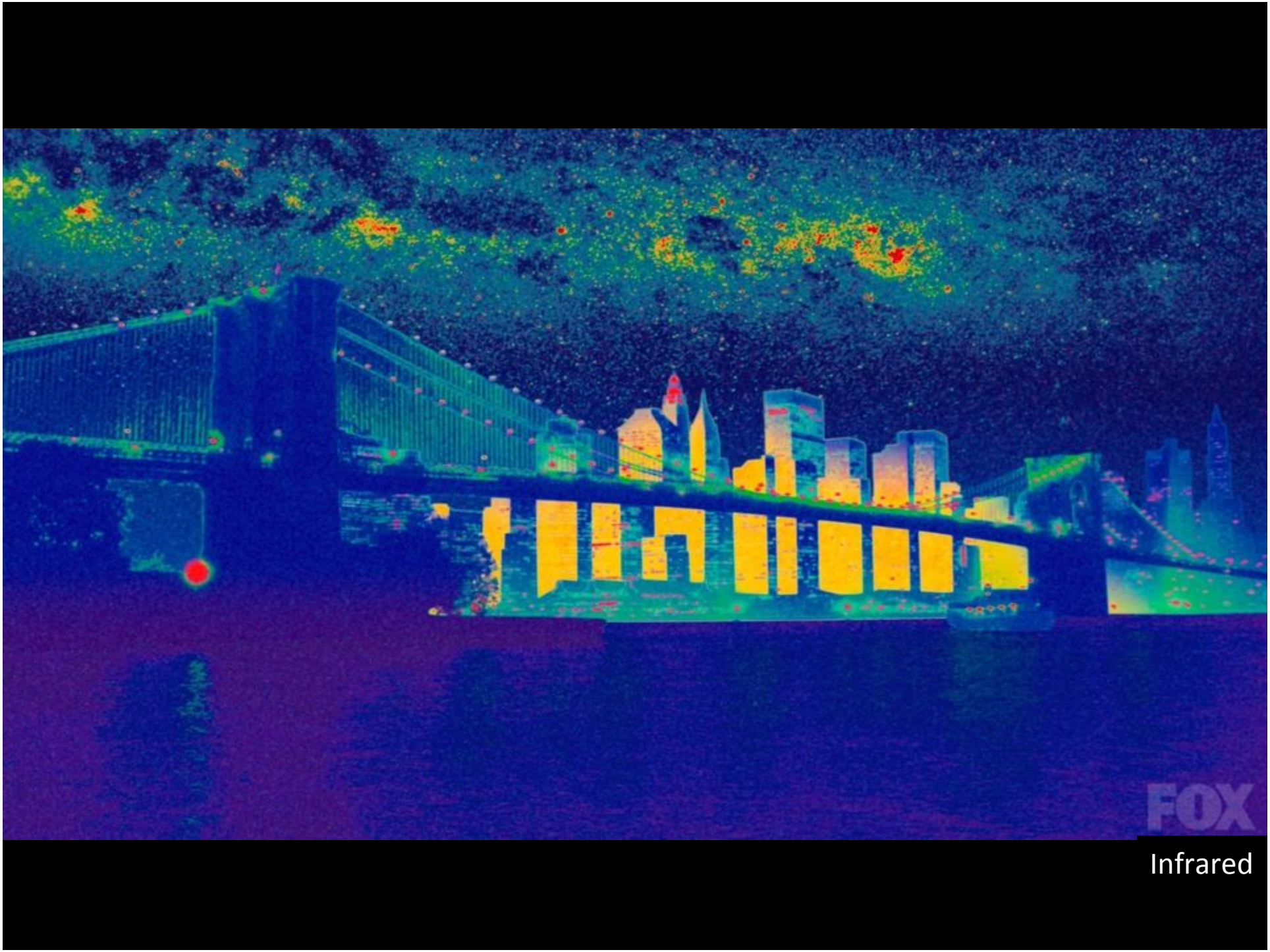


Can’t “see” most of nature that surrounds us



Infrared is already here





Infrared

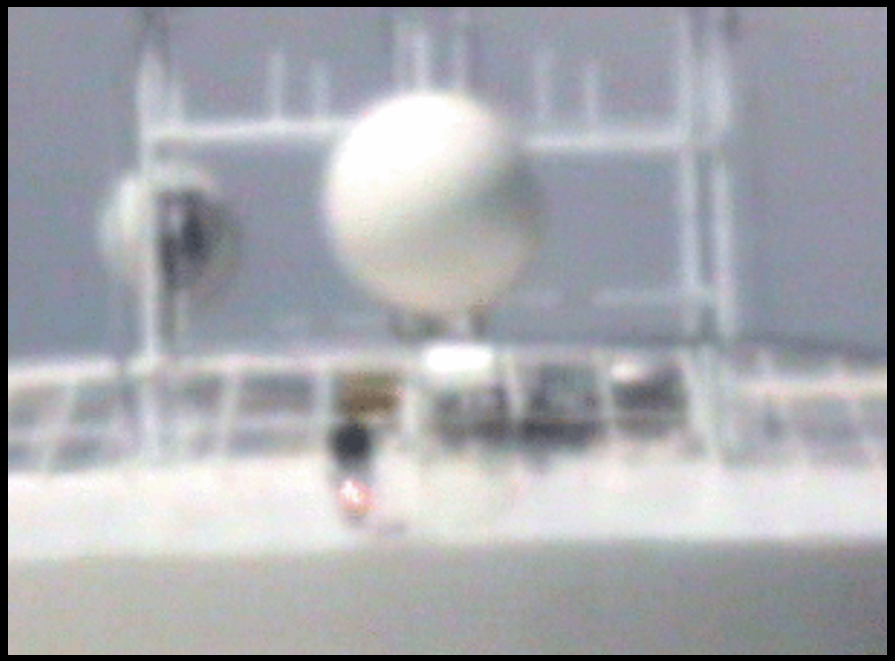

FOX
Gamma-ray

X-ray

Seeing Far Away

Credit : Florian Kainz

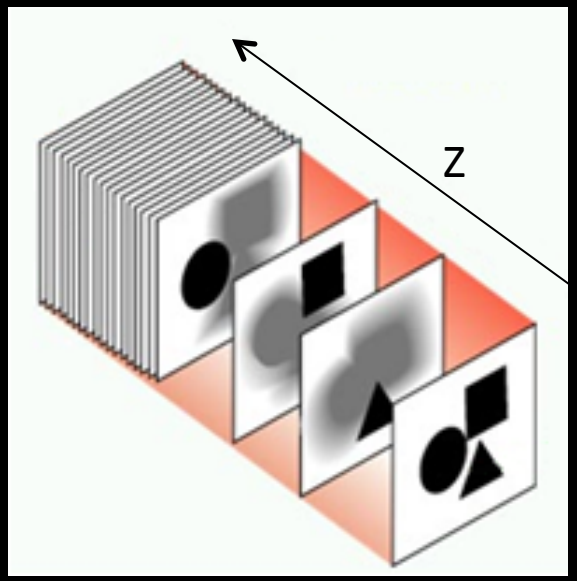
Seeing Far Away

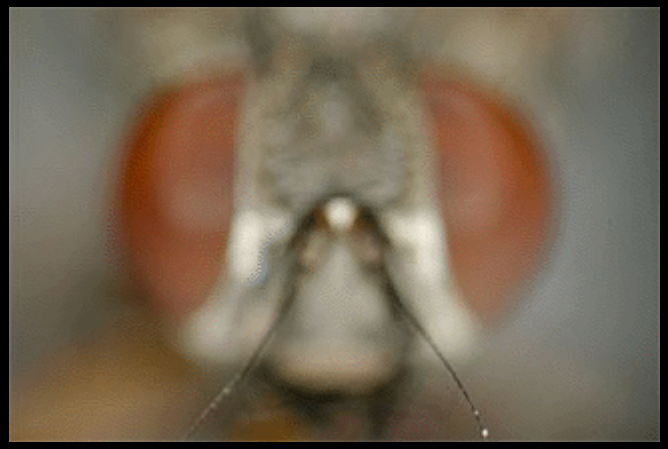


Top part of a water tower imaged at a (horizontal) distance of 2.4 km

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

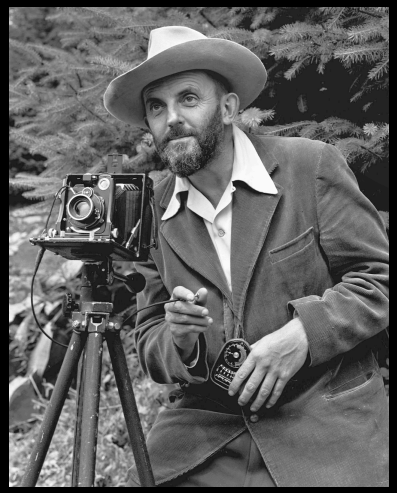
Seeing Small Things

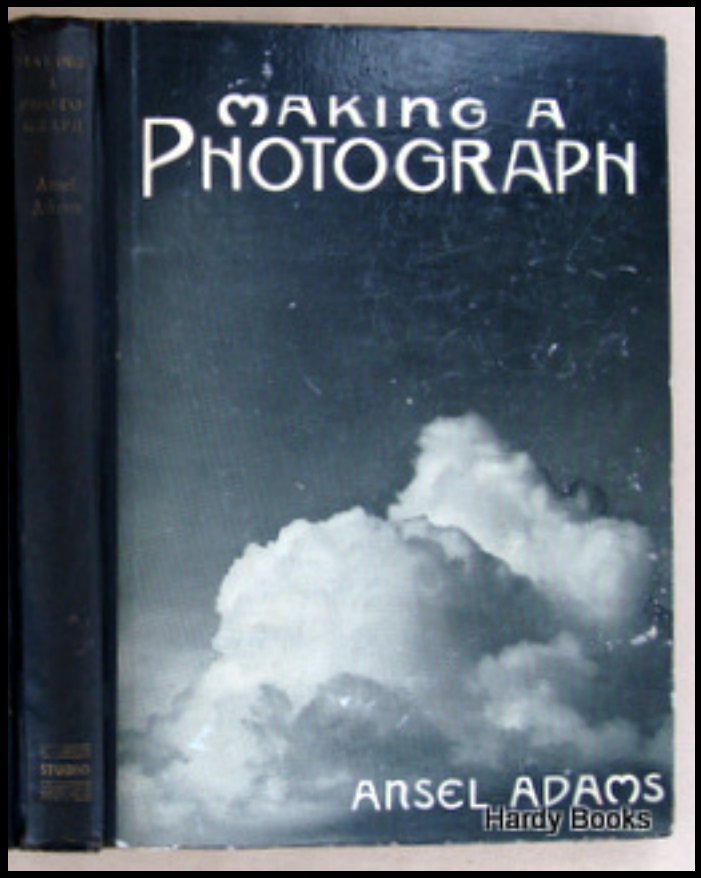




X. Zhu, P. Milanfar, “Removing Atmospheric Turbulence”, PAMI, 2012

How Do We Make a High Quality Image?

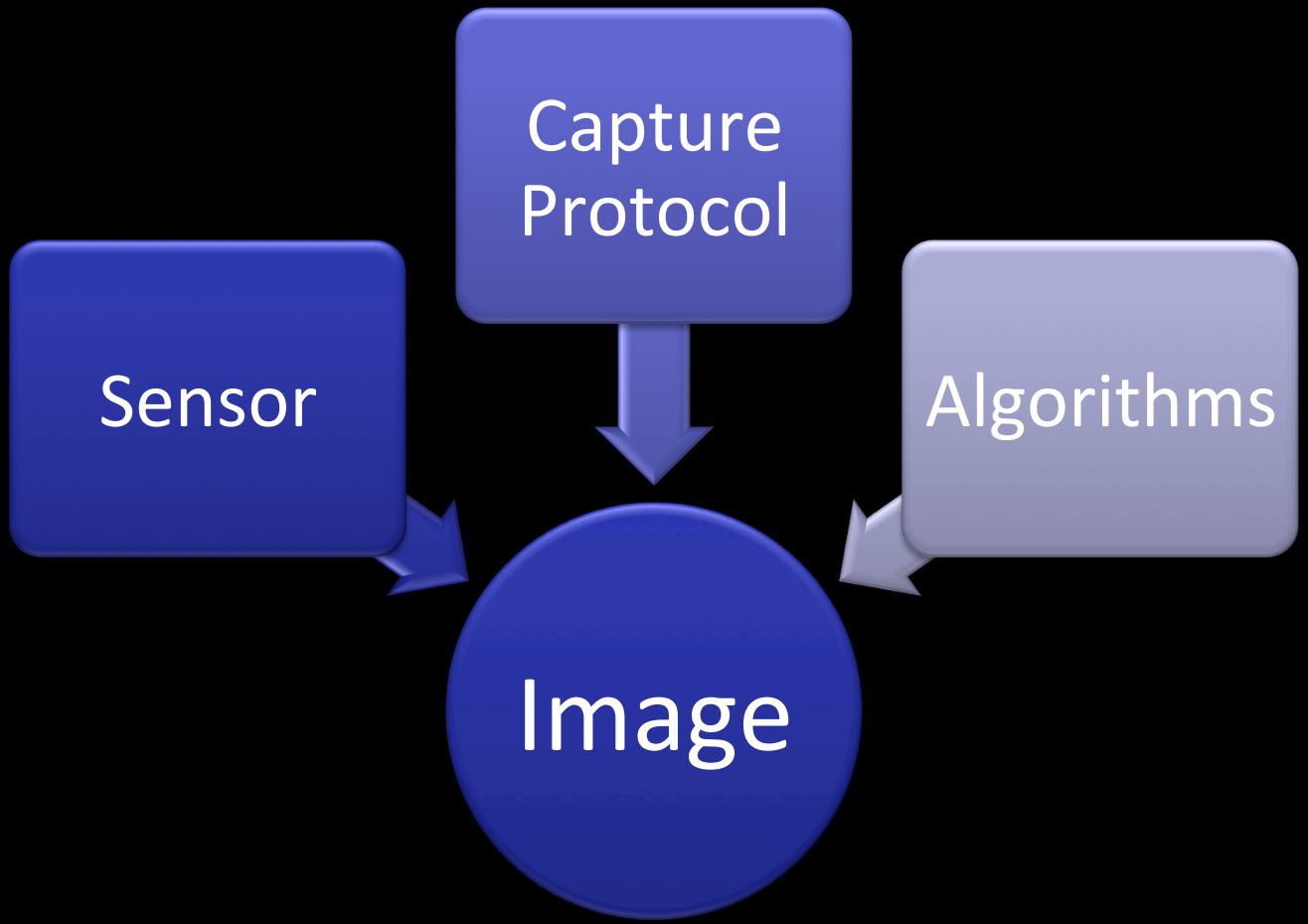




Ansel Adams' first book, published 1935

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

devcamera.org

APK for parameterized image sequence capture using Android 5.0 + (Lollipop)

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

Part 2:

A general filtering framework

The Black Box

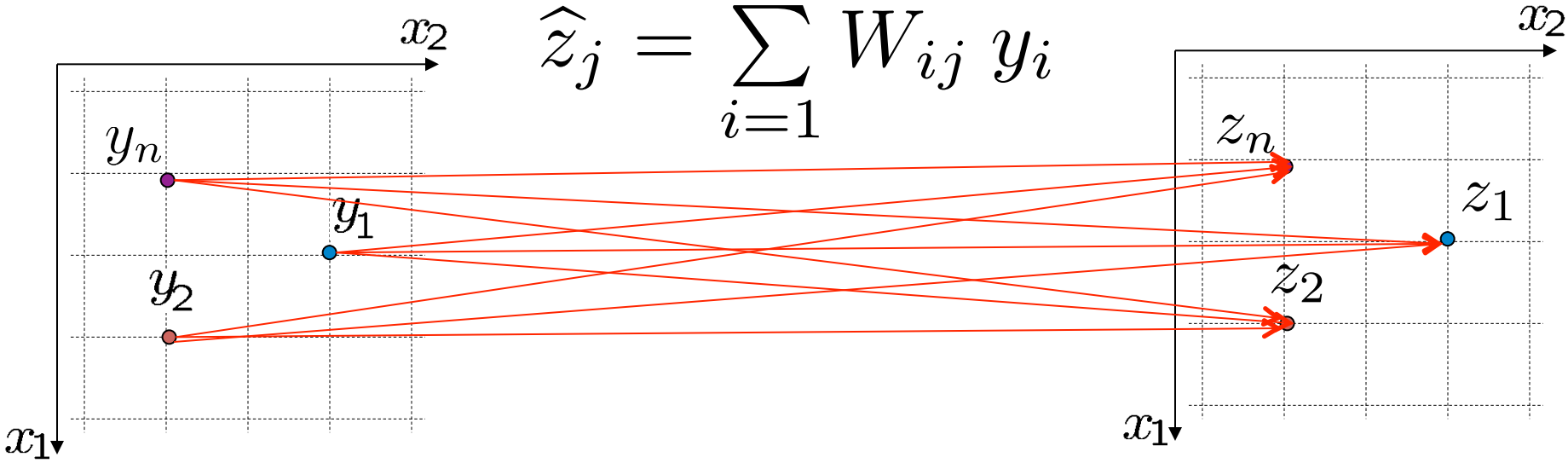
FILTER

Input image

Output image

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

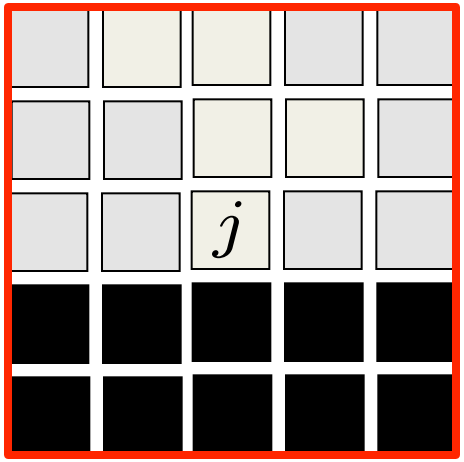
$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$



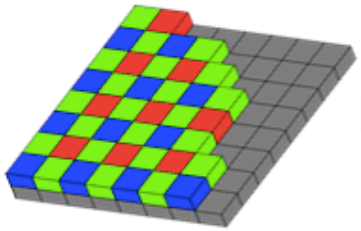
How to design the black box

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

$y_i \in$



Model for Many Parts of Pipline



Sensor with color filter array
(CCD/CMOS)

Gain Control
A/D Converter
Possible LUT

AFE – Analog Front End
Sensor related processing

White
Balance

CFA
Demoasicing

Noise
Reduction/
Sharpening

Color Space
Transform +
Color Preferences

Tone
Reproduction

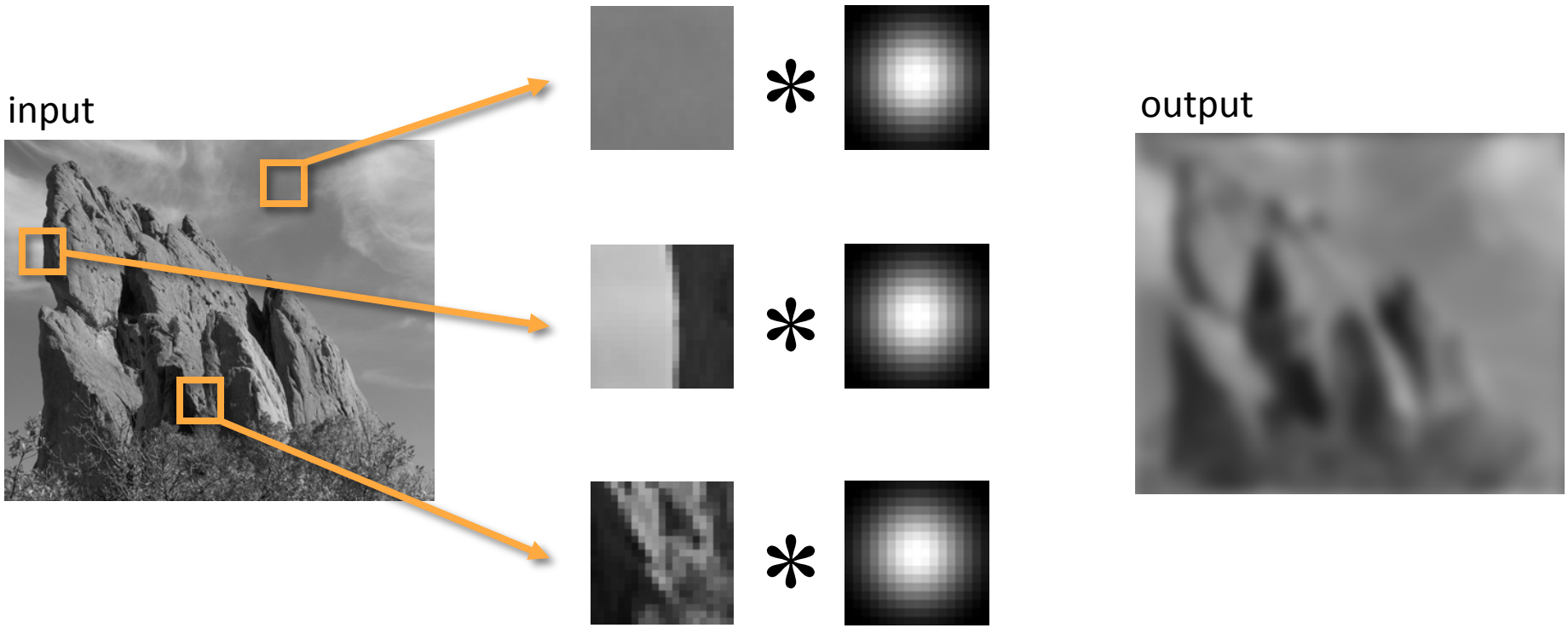
JPEG
Compression

Exif File Info

Michael Brown

Same weights everywhere

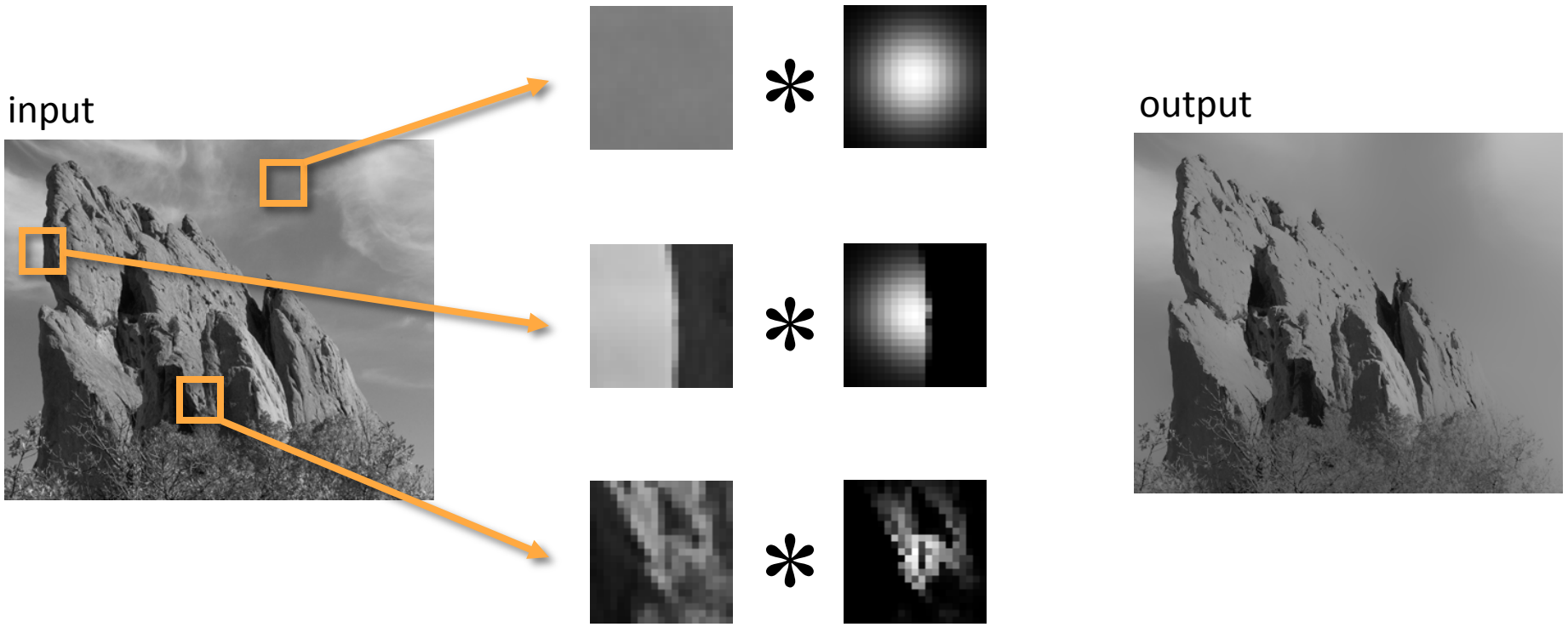
$$\widehat{z}_j = \sum_{i=1}^{n} W_i \, y_i$$



Same Gaussian kernel everywhere.

Data-adaptive weights

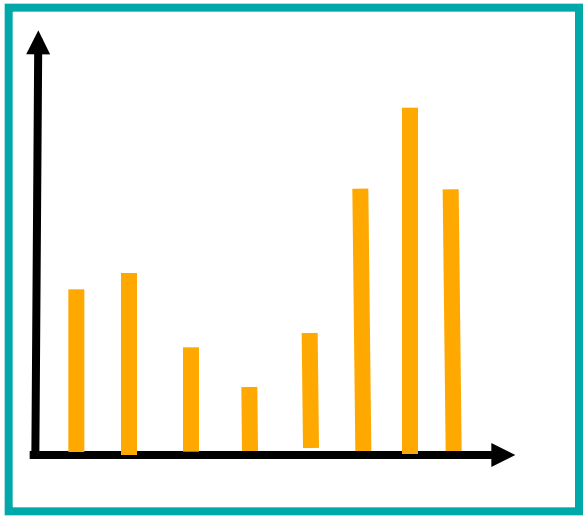
$$\widehat{z}_j = \sum_{i=1}^{n} W_{ij}\, y_i$$



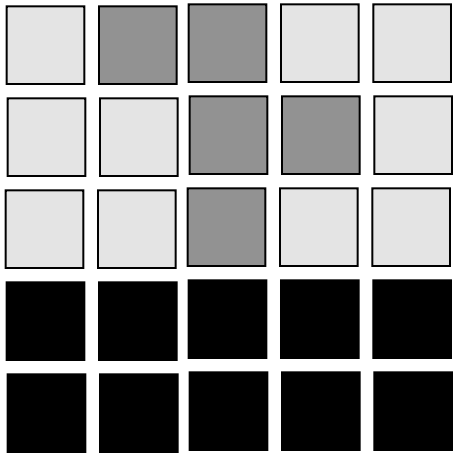
The kernel shape depends on the image content.

Method 1: Consider the histogram

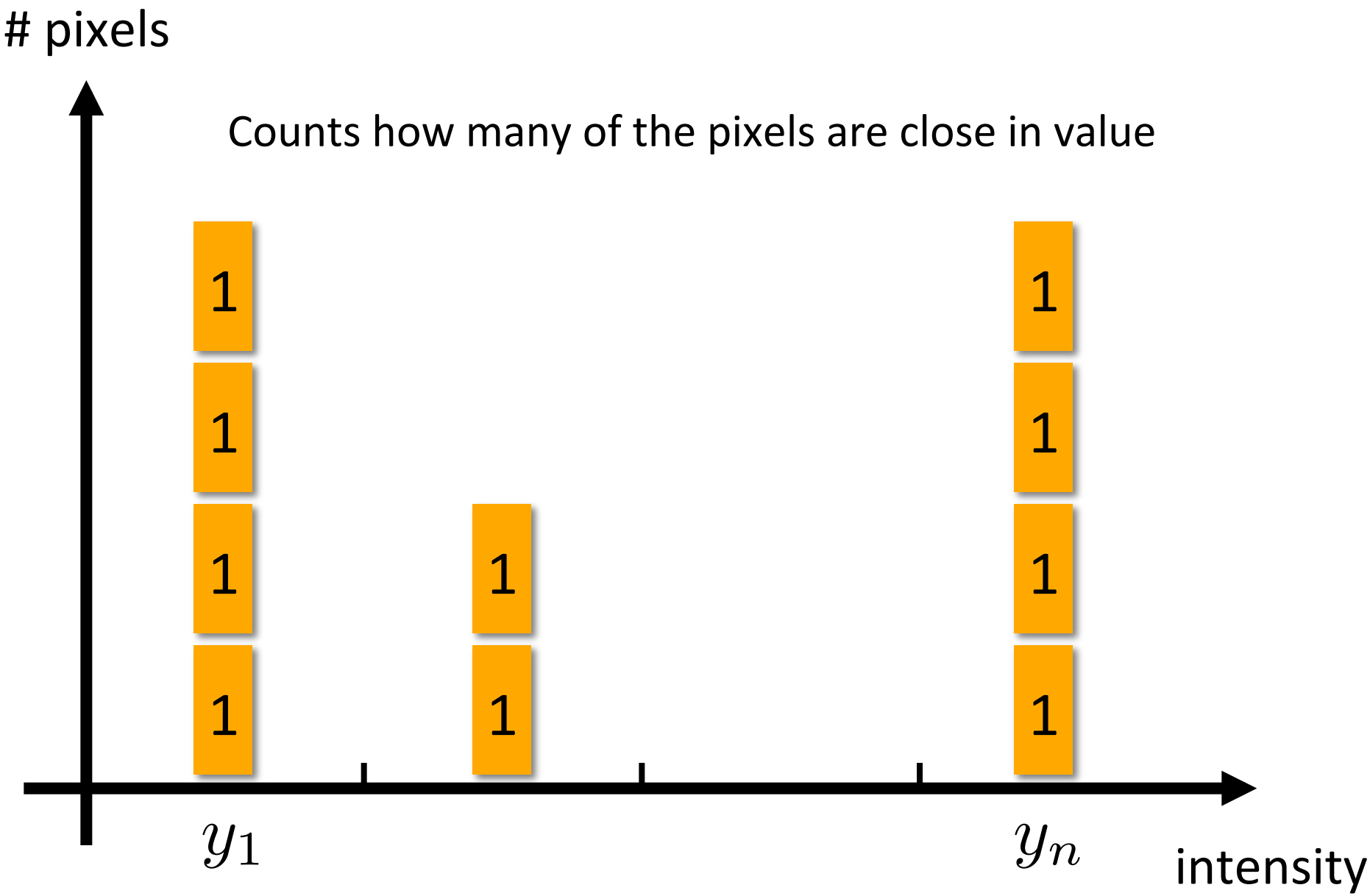
Histogram



y_i



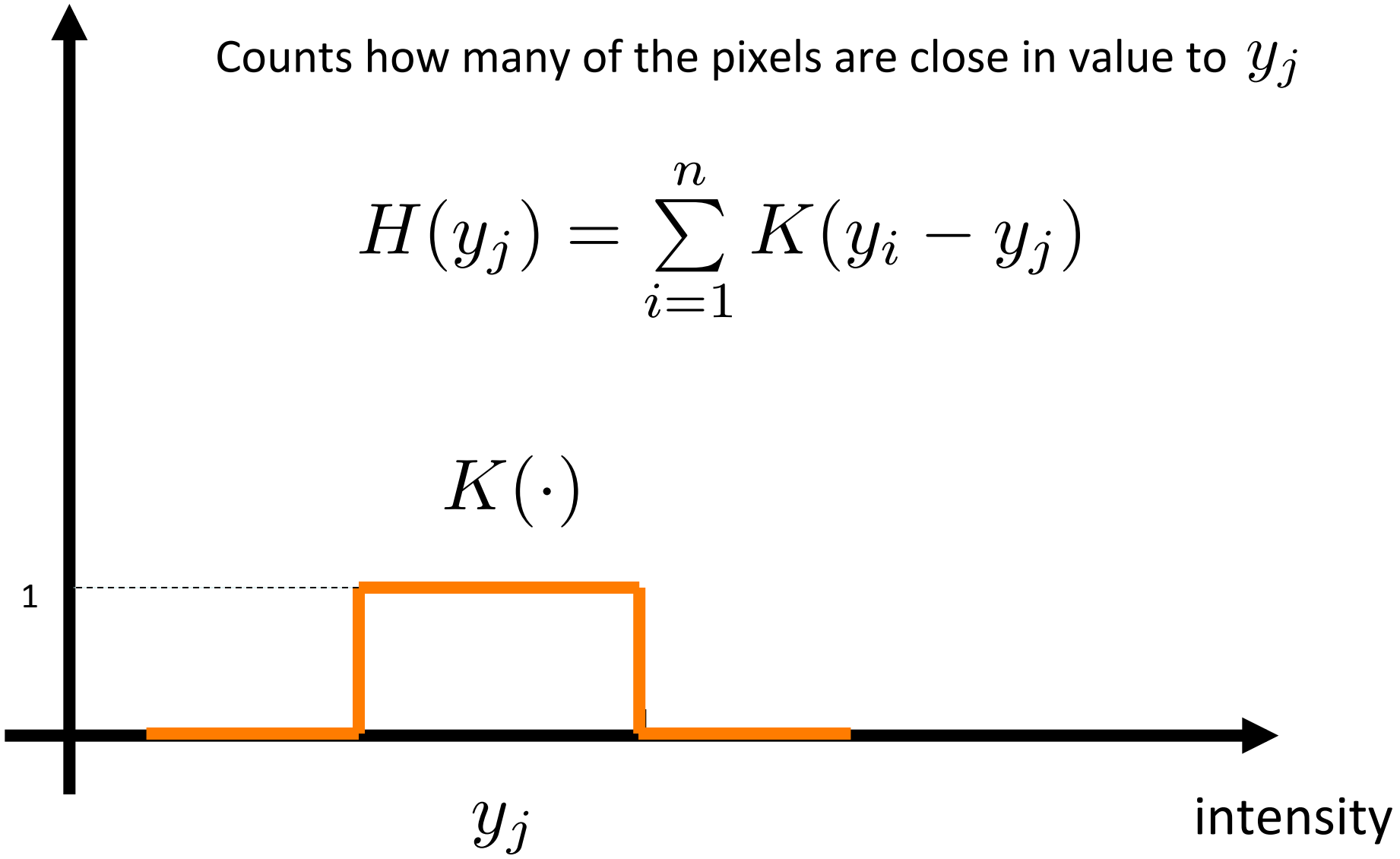
Just for illustration – we don't really make the histogram.

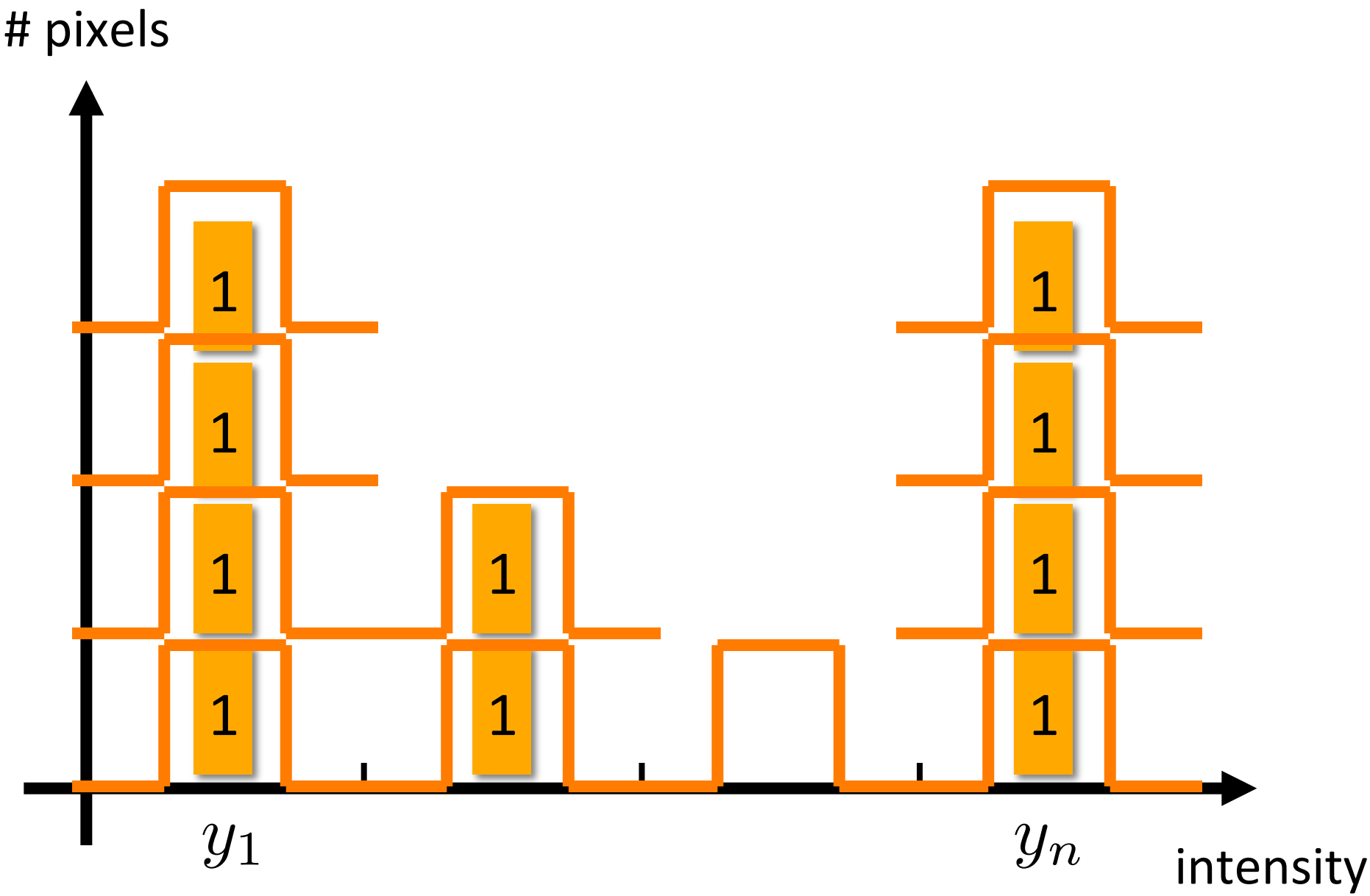

pixels
Counts how many of the pixels are close in value
1
1
1
1
1
1
1
1
1
y_1
y_n
intensity

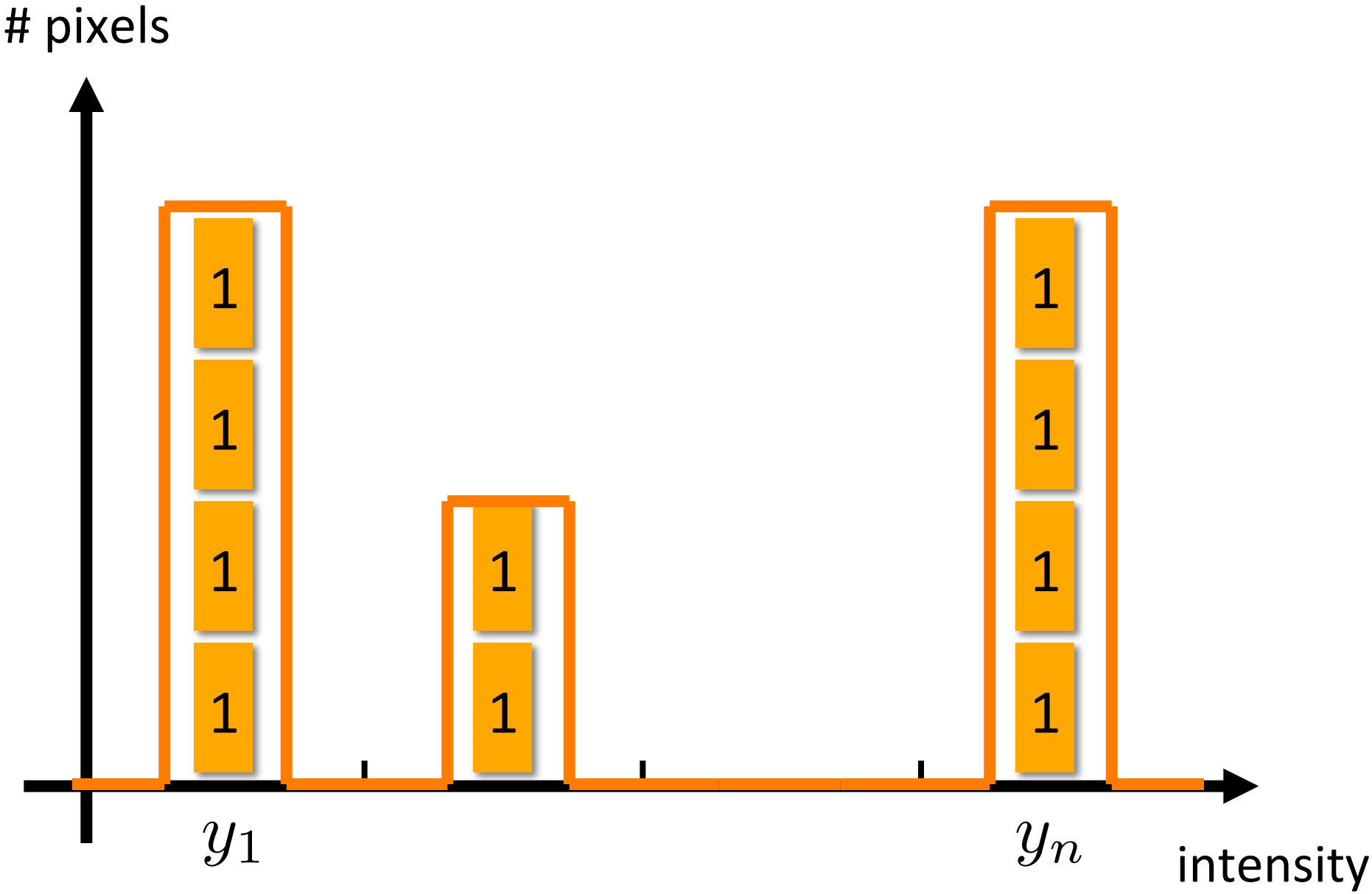
pixels

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
y_1
y_n
intensity

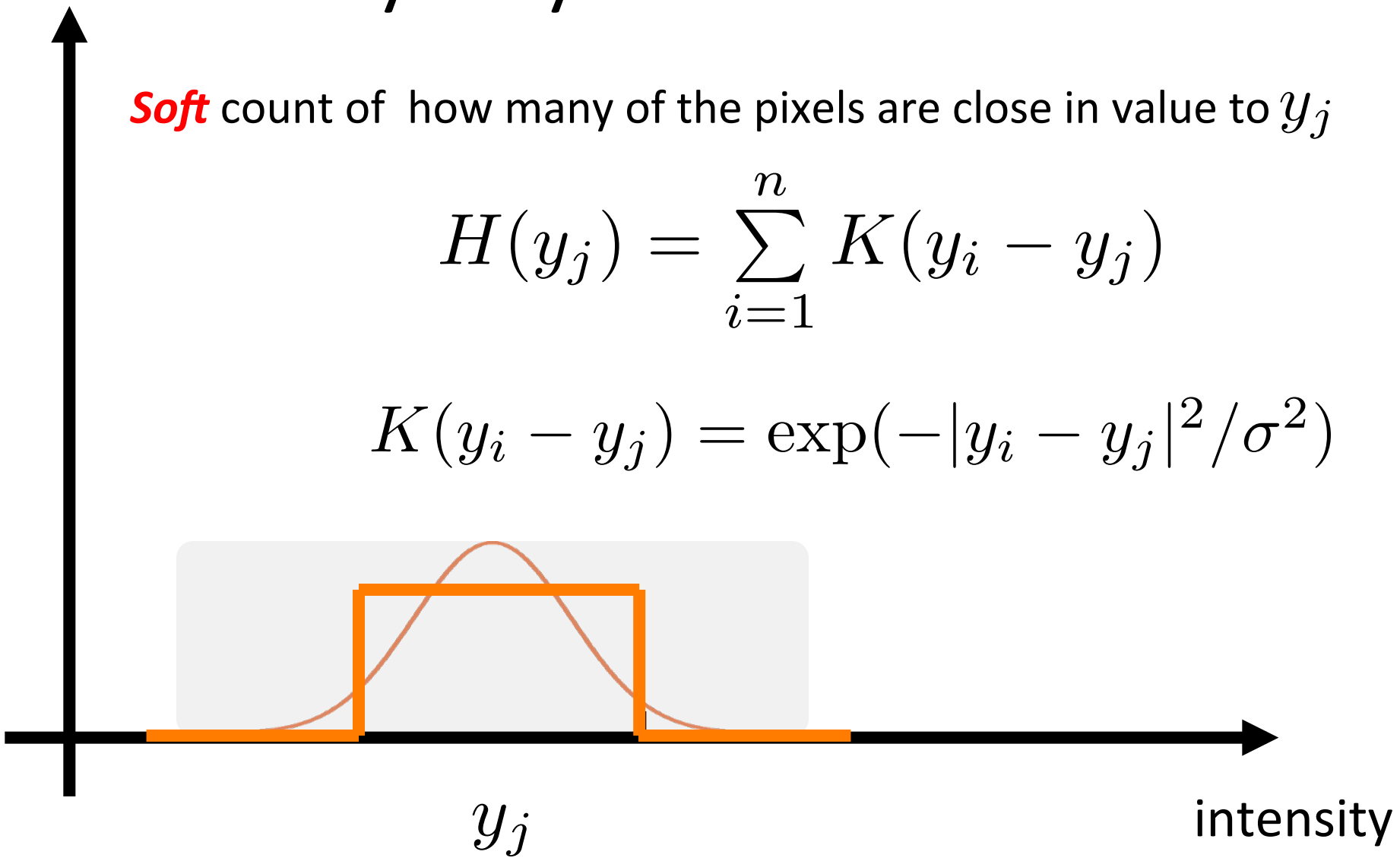

pixels
1
1
1
1
1
1
1
1
1
y_1
y_n
intensity

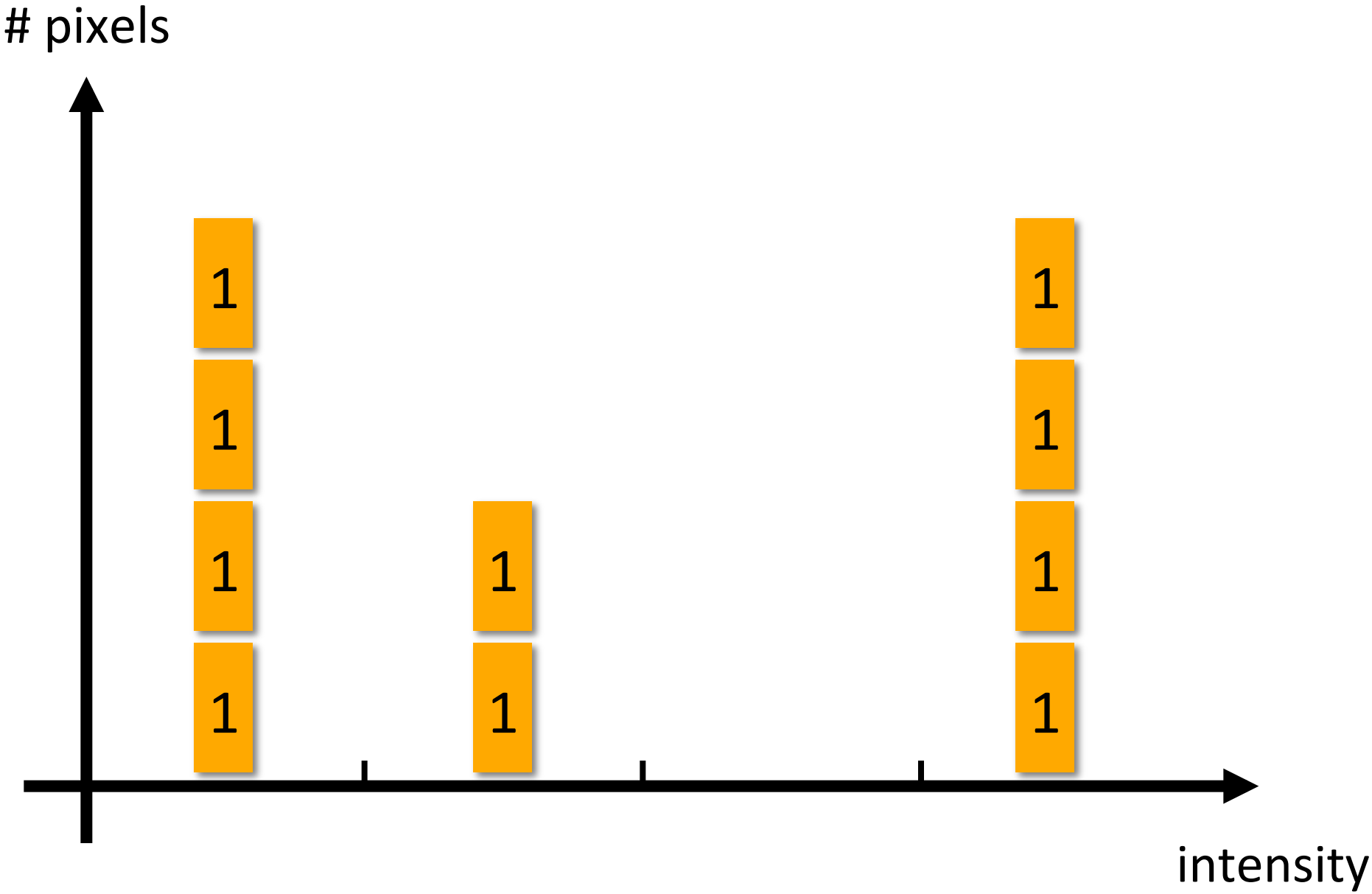
Why only the box function?

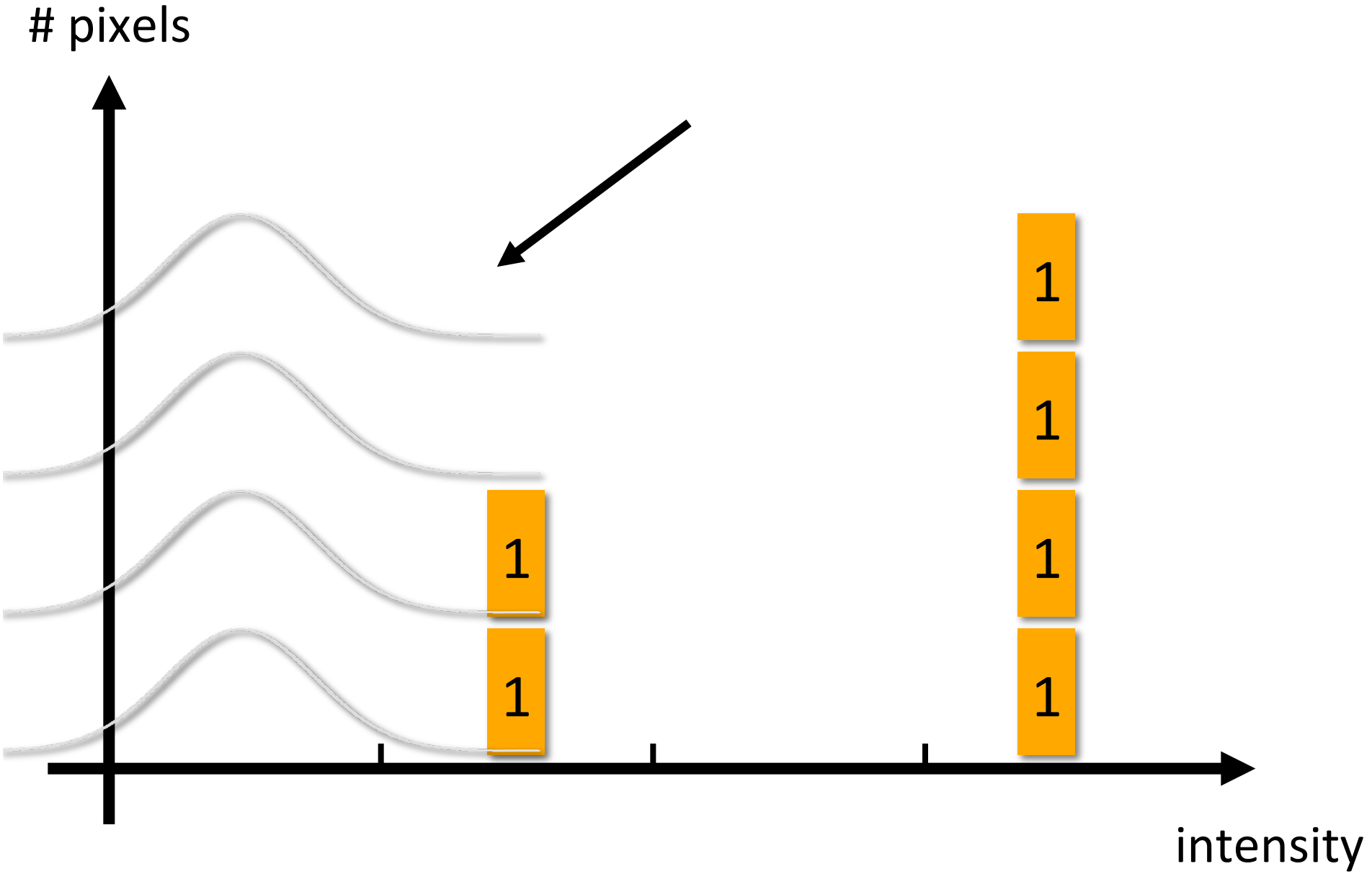
Soft count of how many of the pixels are close in value to y_j

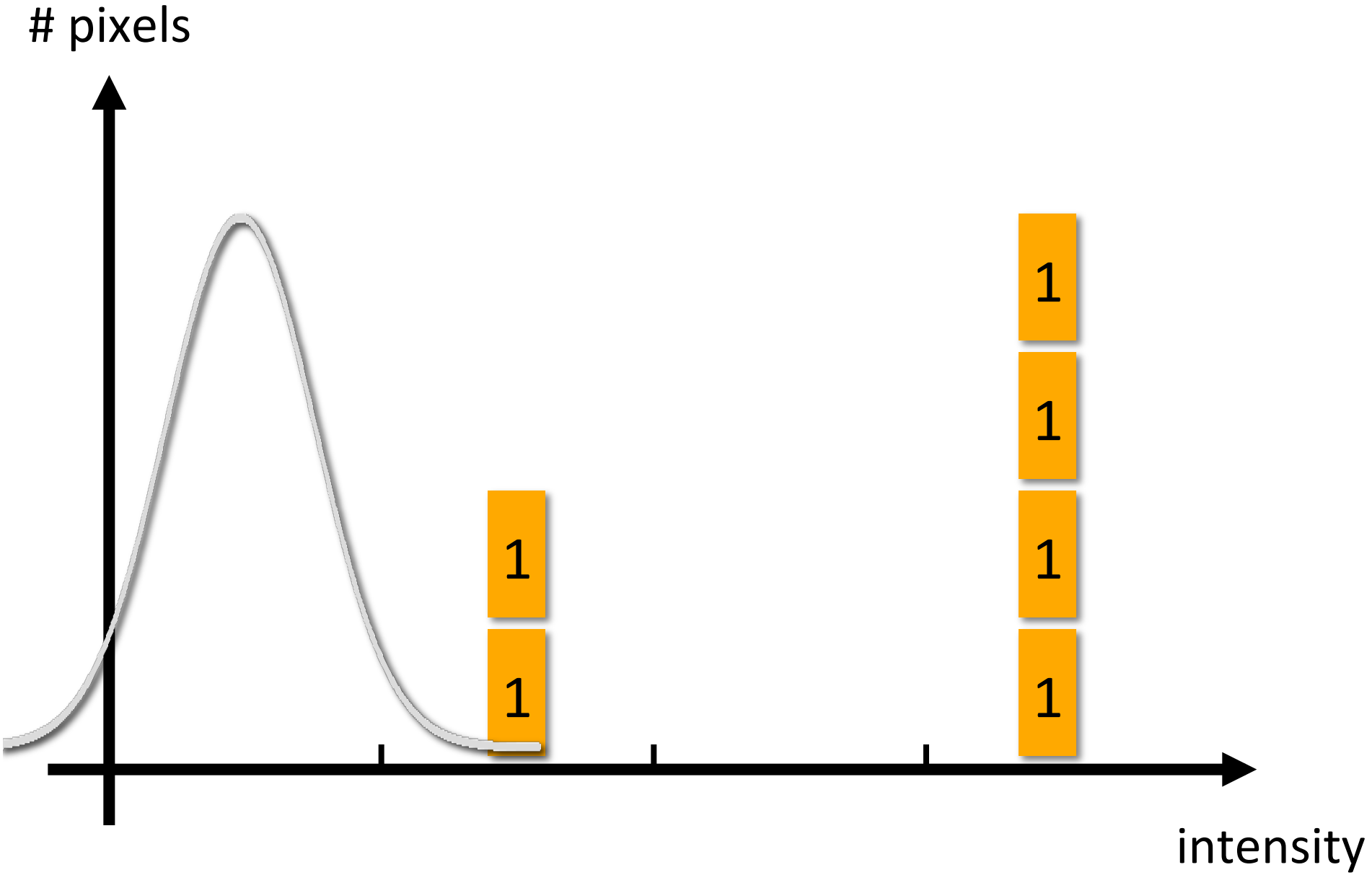
$$H(y_j) = \sum_{i=1}^{n} K(y_i - y_j)$$

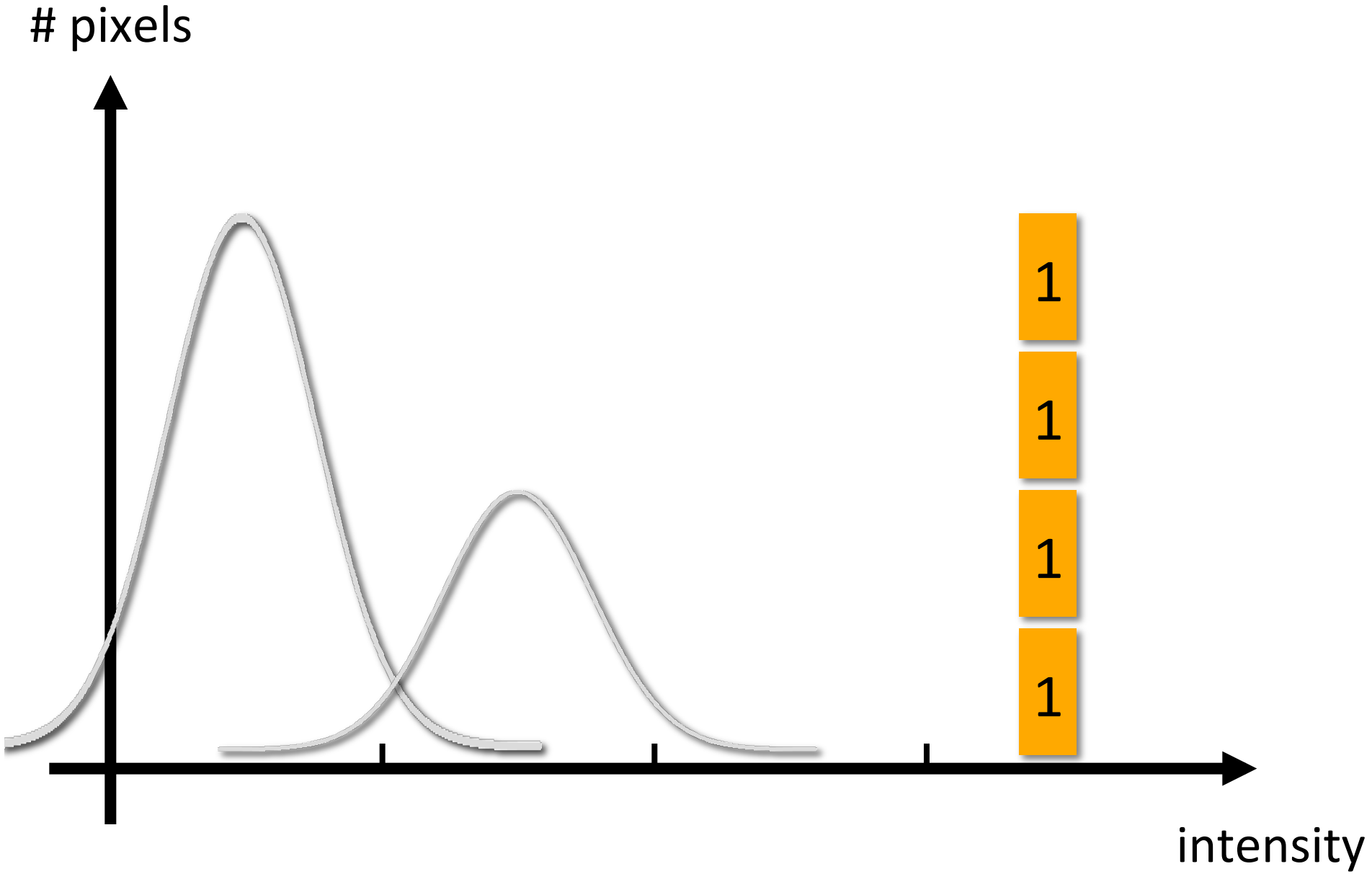
$$K(y_i - y_j) = \exp(-|y_i - y_j|^2/\sigma^2)$$

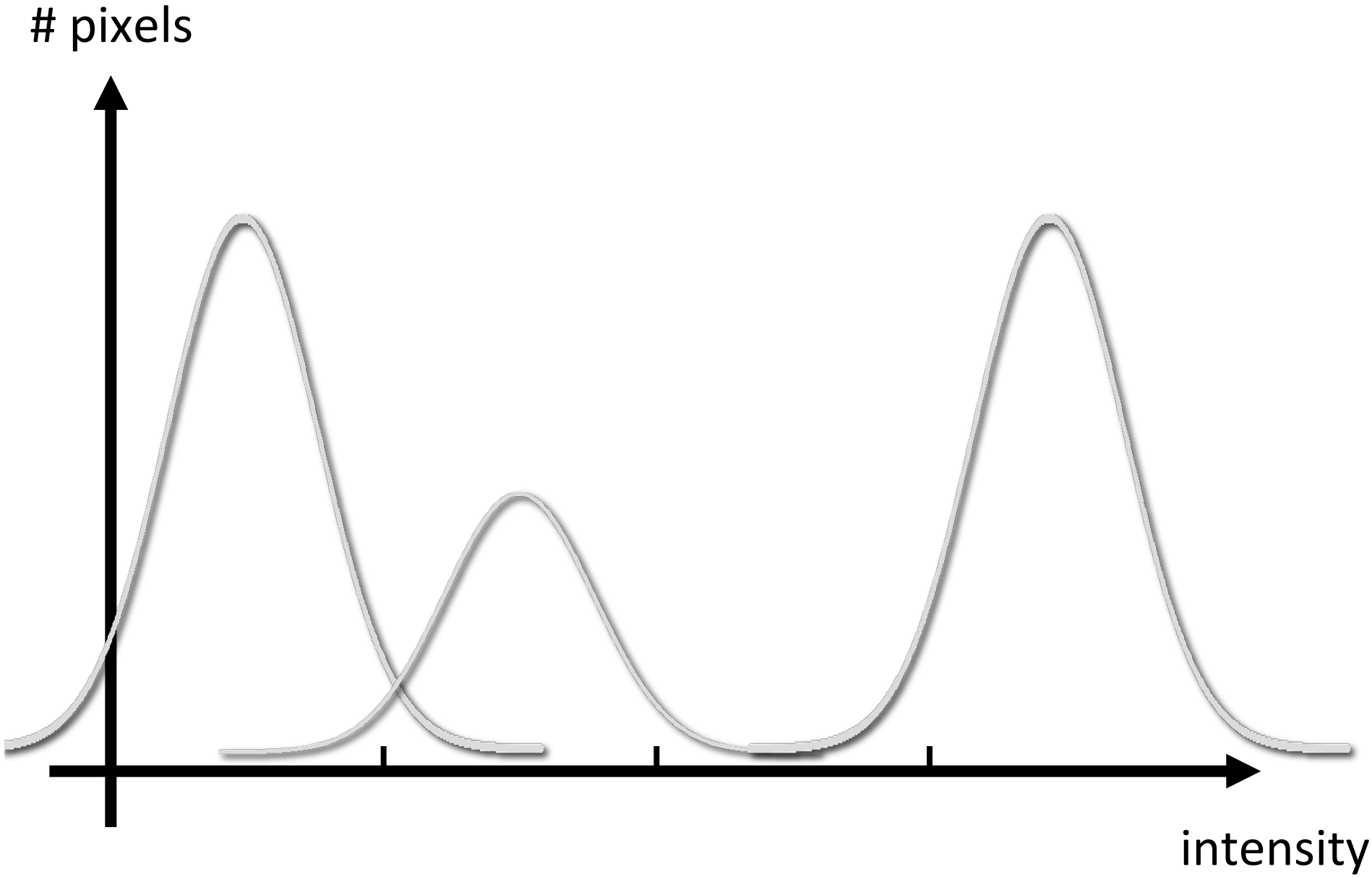


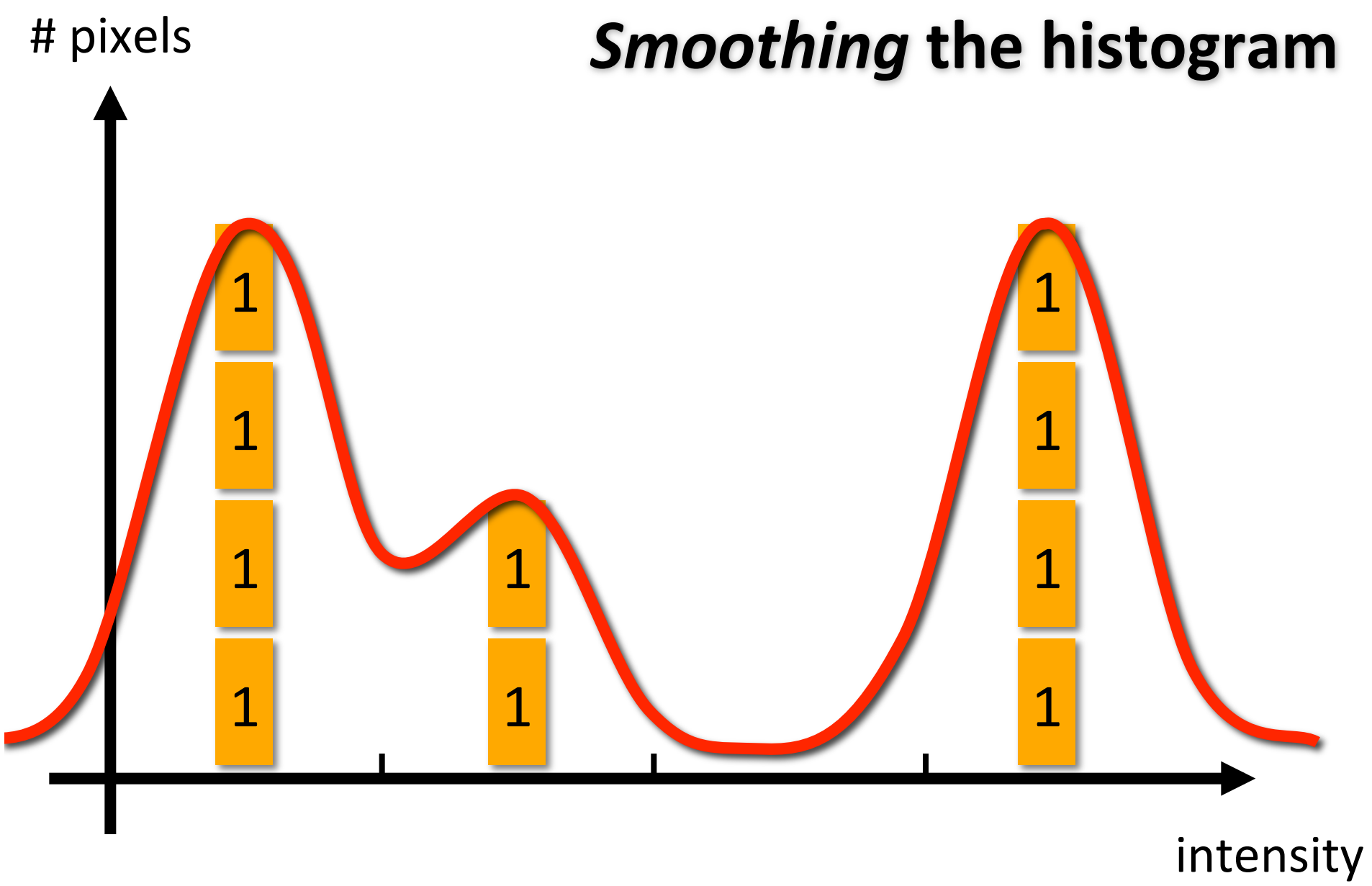

pixels
1
1
1
1
1
1
1
1
1
intensity


pixels
1
1
1
1
1
1
intensity

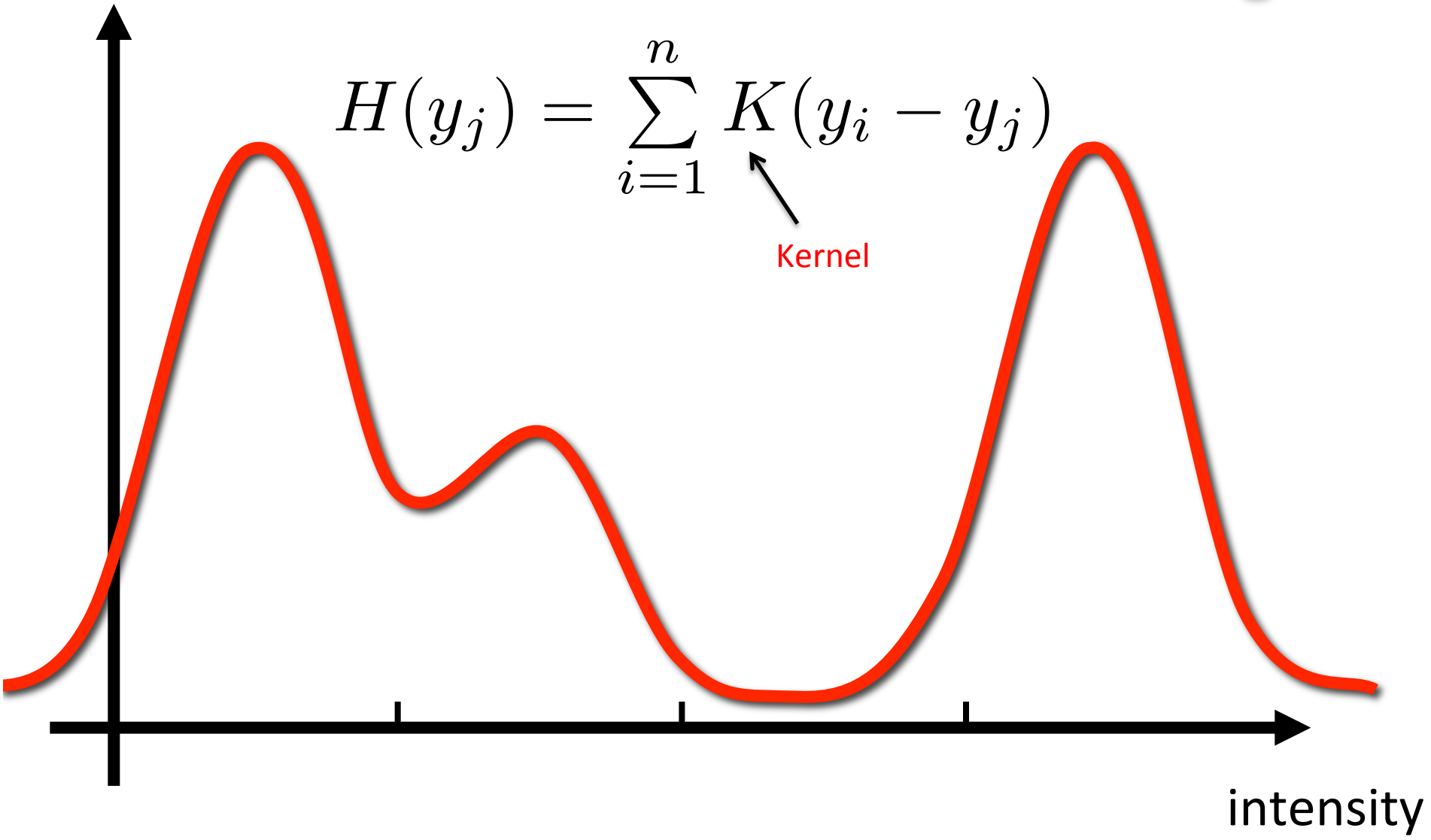

pixels
1
1
1
1
1
intensity


pixels
1
1
1
1
intensity

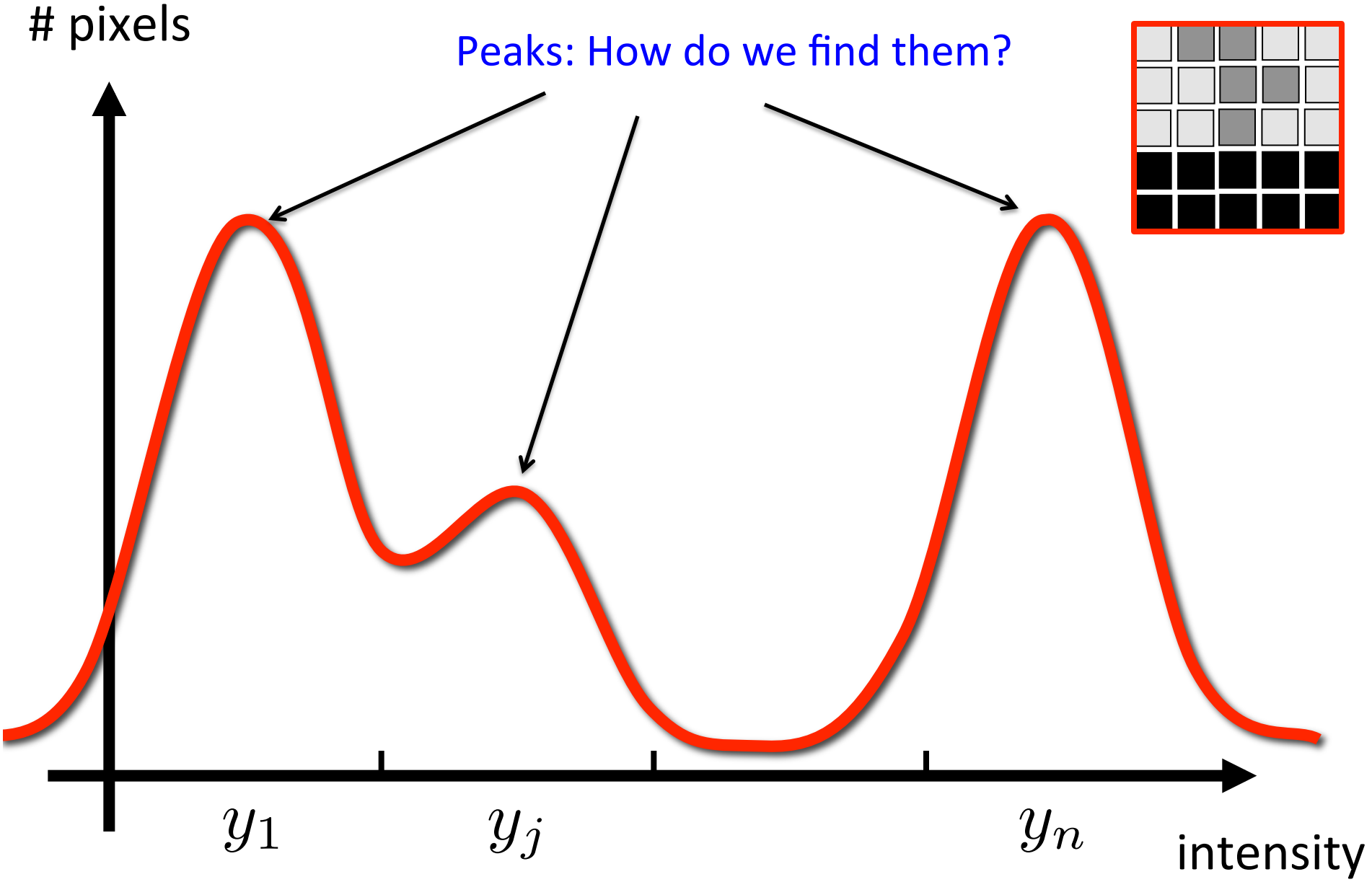

pixels
intensity


pixels
Smoothing the histogram
1
1
1
1
1
1
1
1
1
1
intensity

Smoothed histogram



Modes (peaks) of the Histogram



Mode Finding on the Histogram

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

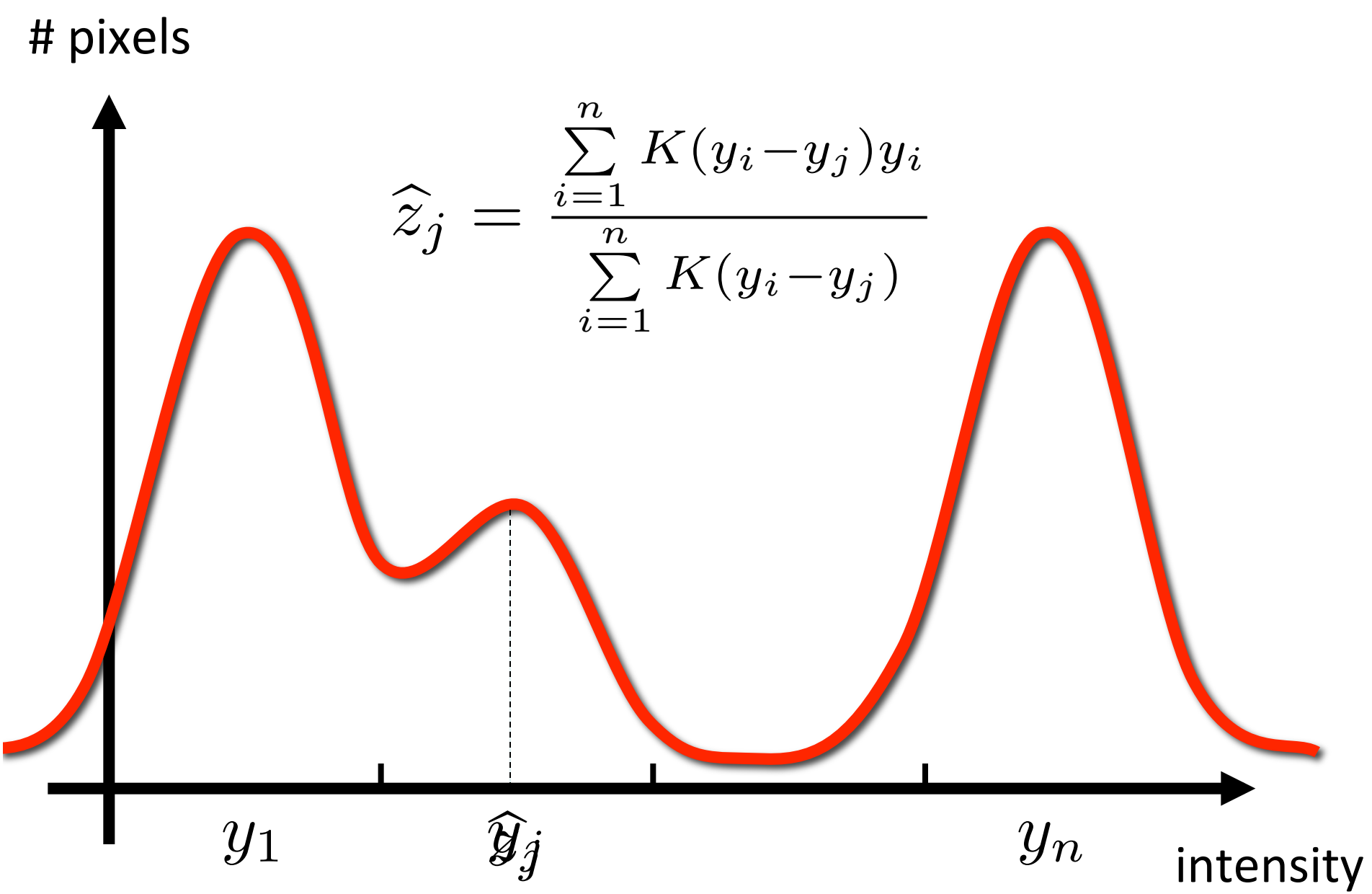
$$\text{Fix } y_j = t$$

$$\text{Differentiate and solve:} \quad \frac{\partial H(t)}{\partial t} = 0$$

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

(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

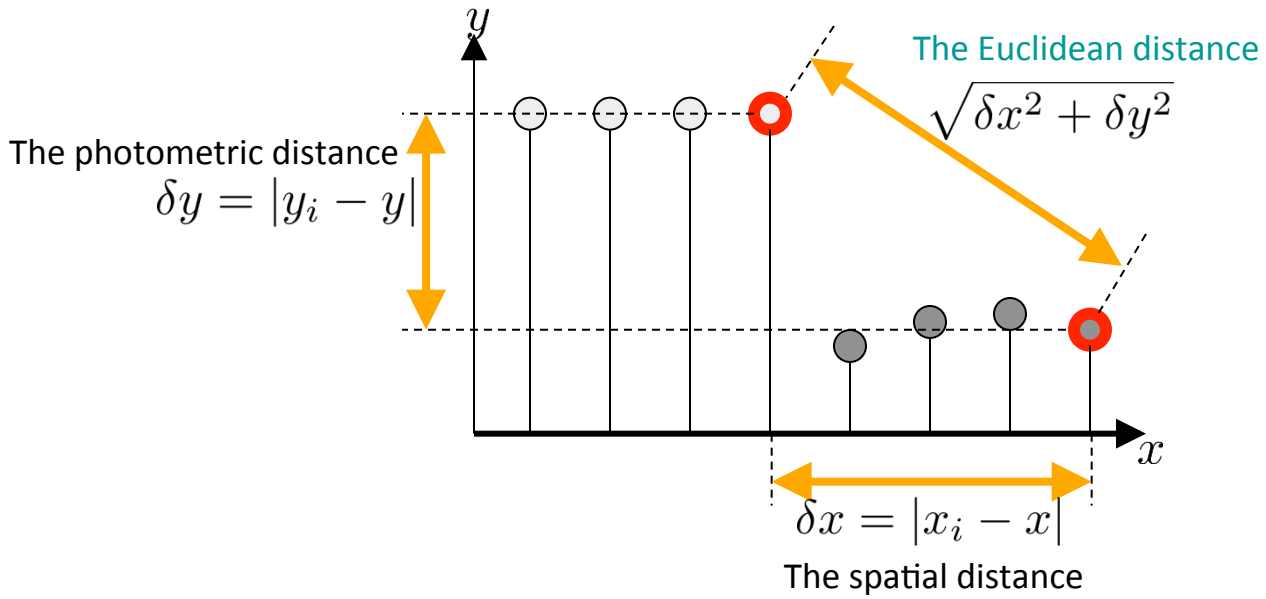
$$\sum_{i=1}^{n} W_{ij} = 1$$

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

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

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

Filter once

Filter twice

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

Divide Pixel-wise

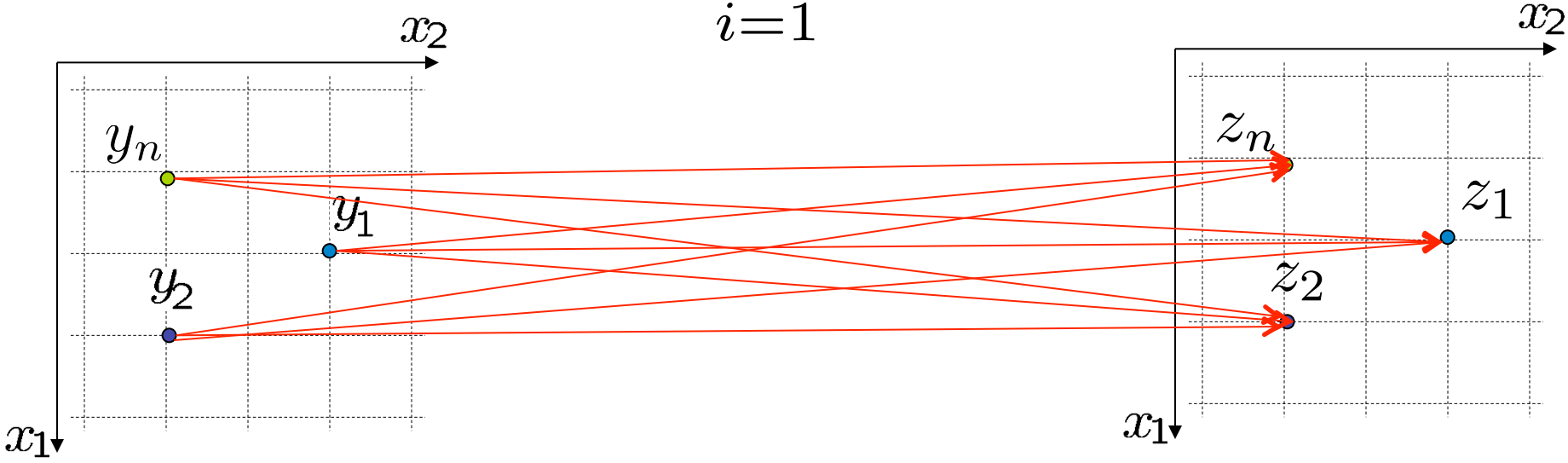
The Black Box

FILTER

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

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

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

rows

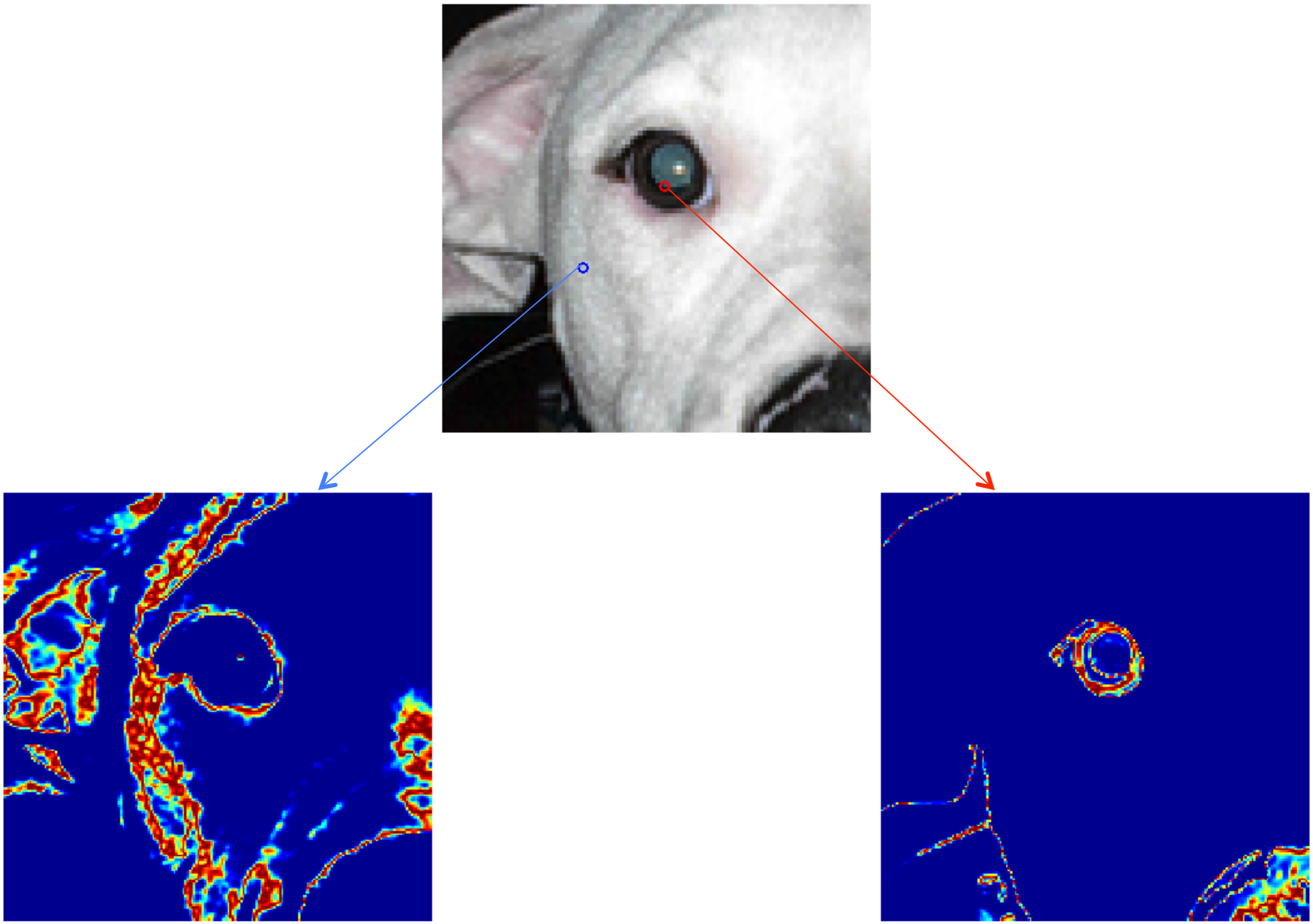
$$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1^T \\ \mathbf{w}_2^T \\ \vdots \\ \mathbf{w}_n^T \end{bmatrix}$$

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

Data-dependent matrix

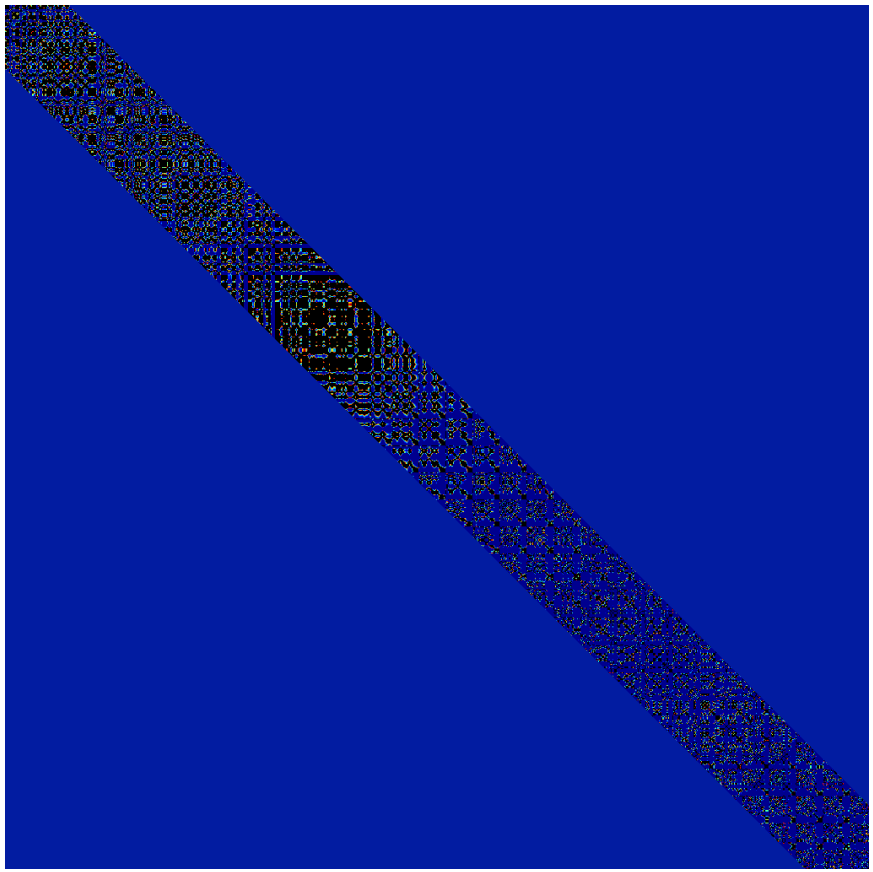
Image scanned into a vector

Global Affinities



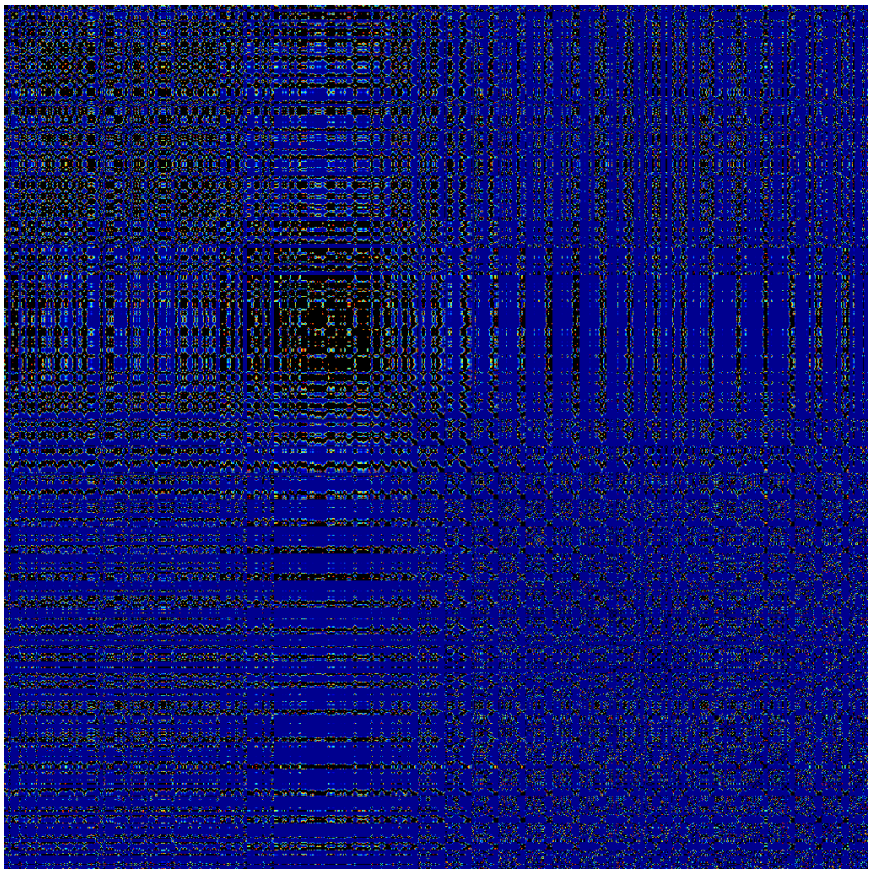
Local vs. Global Filter Matrix W

Local Filters



Sparse, but high-rank

Global Filters



Dense but low-rank

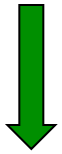
A Linear Embedding Approach

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

$$\mathbf{W} = \mathbf{V}\mathbf{S}\mathbf{V}^T$$

Orthogonal Eigenvectors
$\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_n]$

Eigenvalues
$\mathbf{S} = \mathrm{diag}[\lambda_1, ..., \lambda_n]$
$0 \leq \lambda_n \leq ... < \lambda_1 = 1$

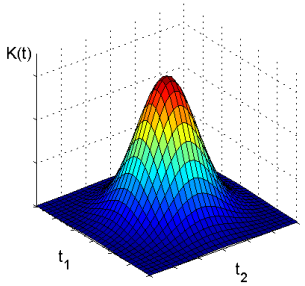


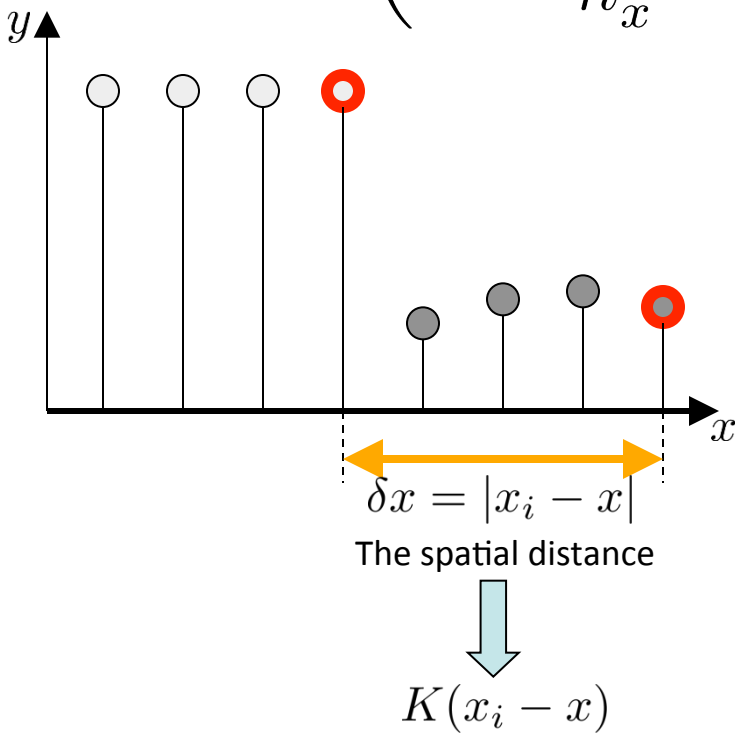
$$\widehat{\mathbf{z}} \approx \sum_i \alpha_i \times \text{ top eigenvectors of } \mathbf{W}$$

Nonlinear Filtering is now "Linear"

Choice of Kernels

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

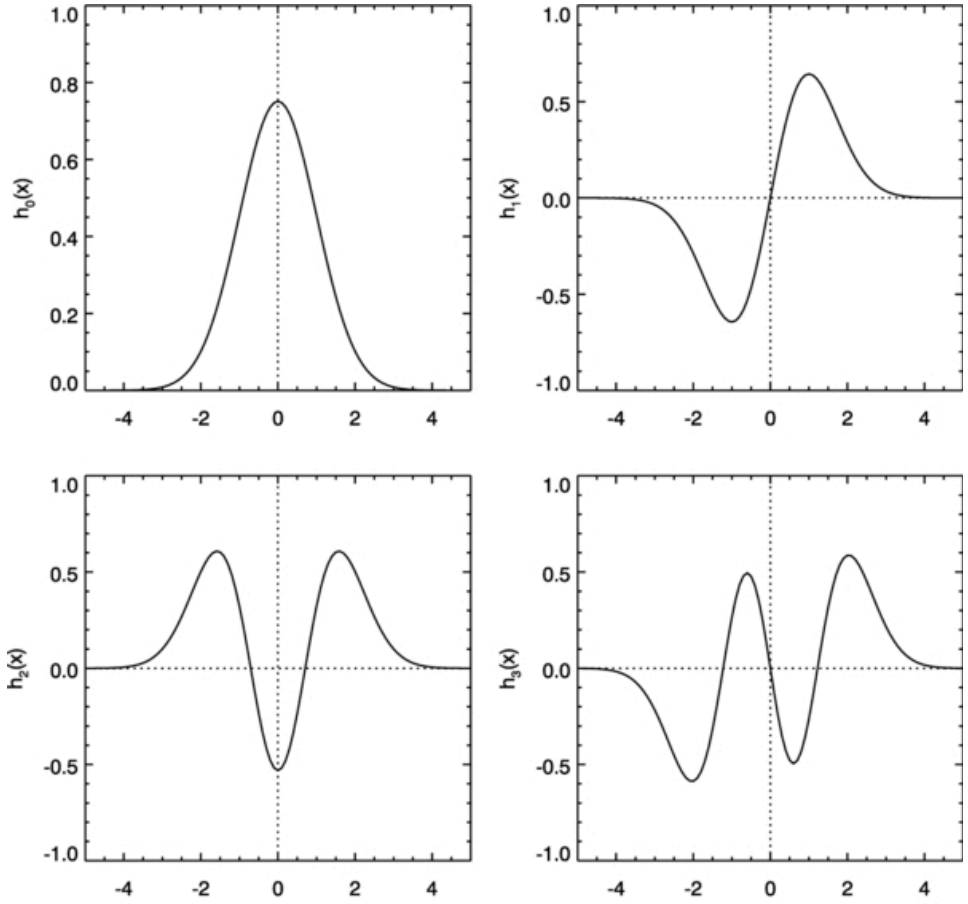




- Classical Gaussian Linear Filters

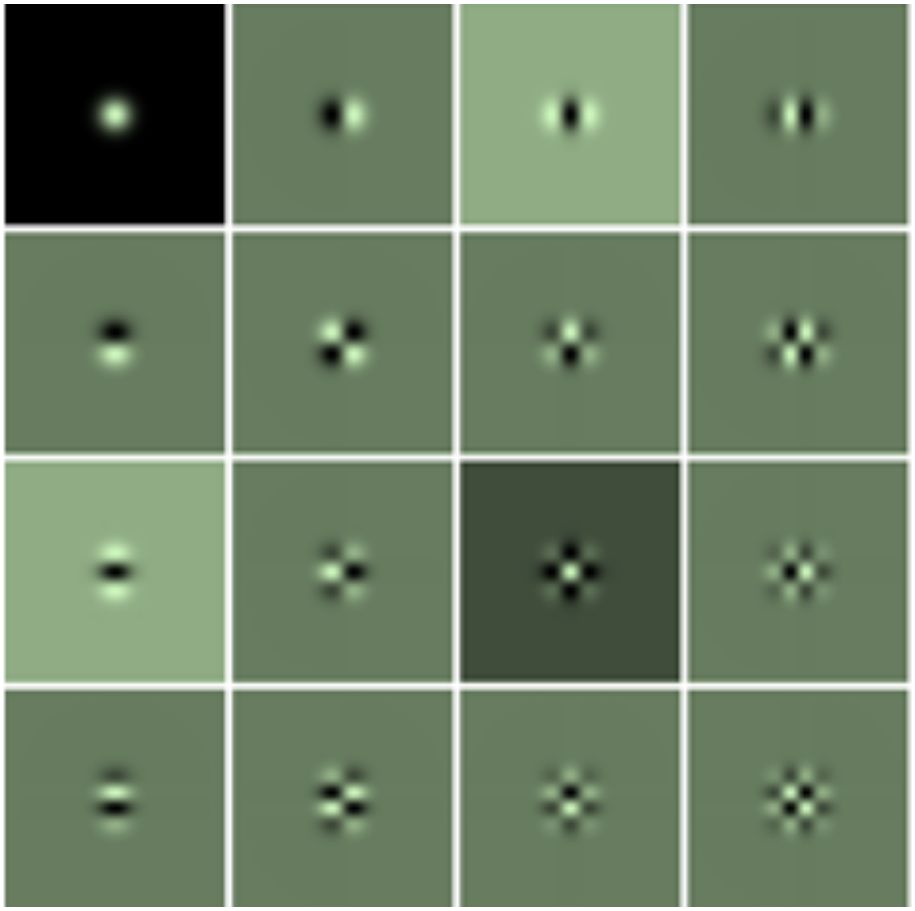
Eigenvectors of Linear Gaussian Kernel

Hermite-Gauss functions



Eigenvectors of Linear Gaussian Kernel

Hermite-Gauss functions

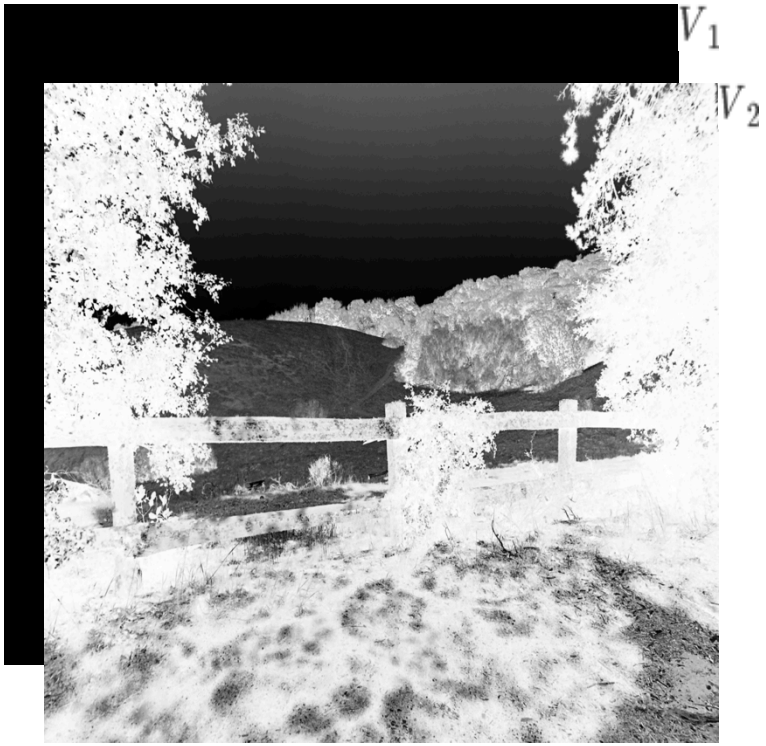


Filter Eigenvectors (For NLM)

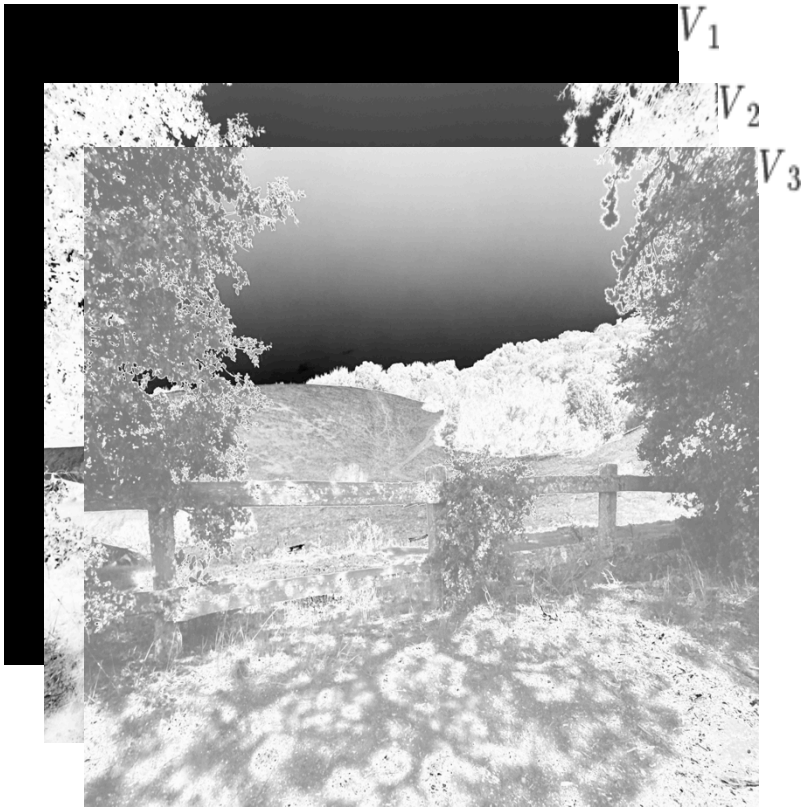
(constant image)

V_1

Filter Eigenvectors

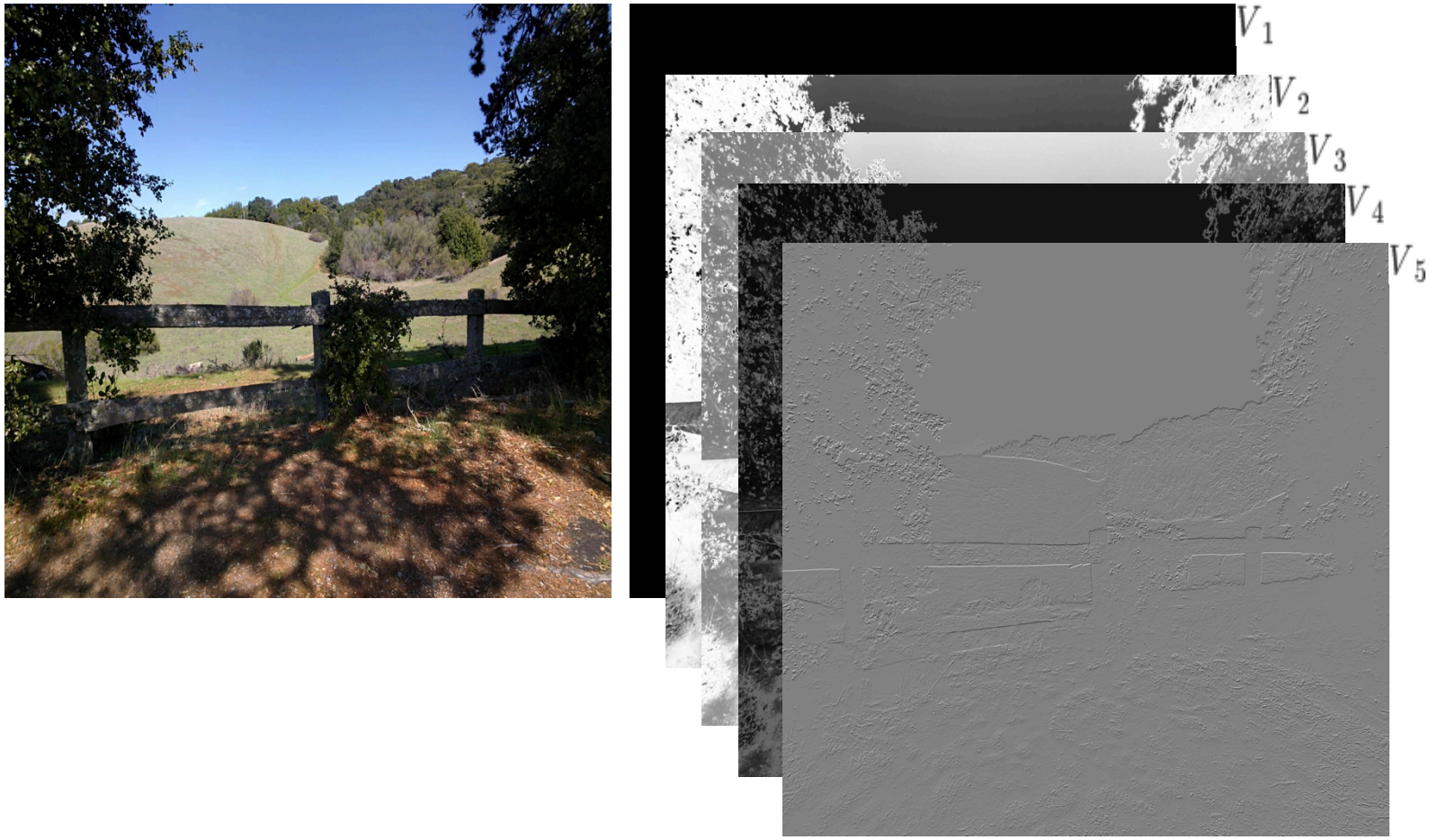


Filter Eigenvectors

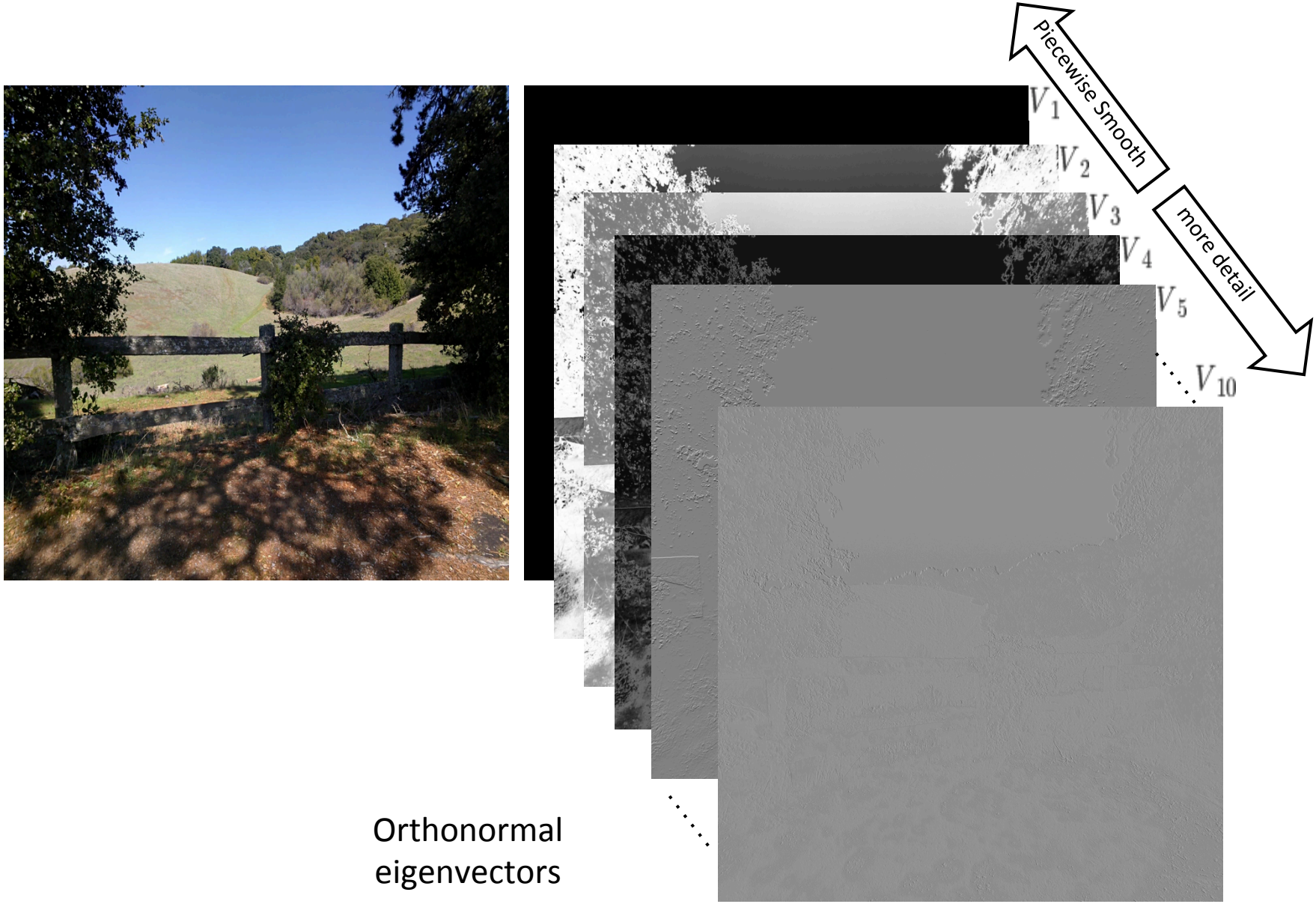


Filter Eigenvectors

Filter Eigenvectors

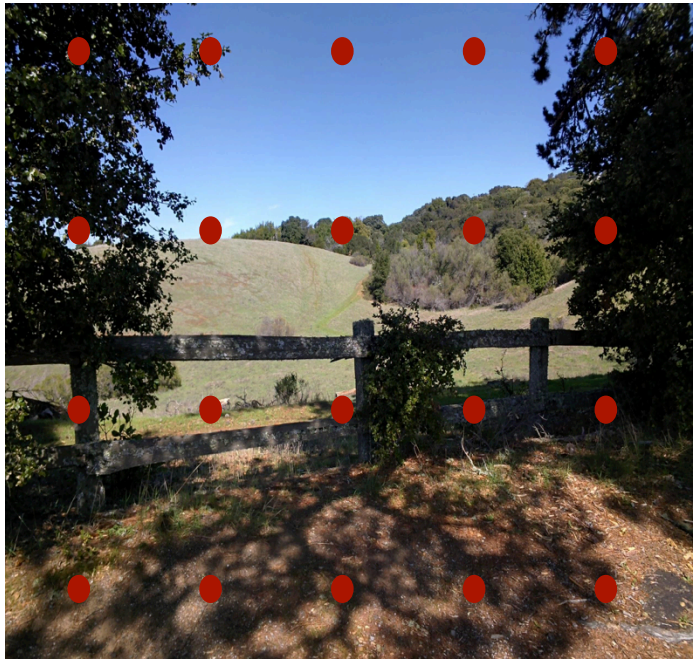


Filter Eigenvectors



Orthonormal eigenvectors

Nystrom Approximation

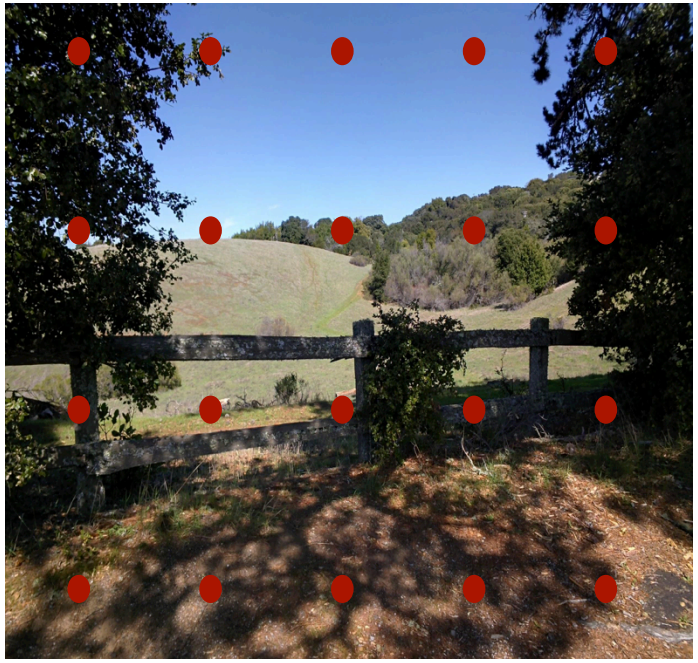


Spatially uniform sampling

A: Sampled pixels (m)

B: Remaining pixels (n-m)

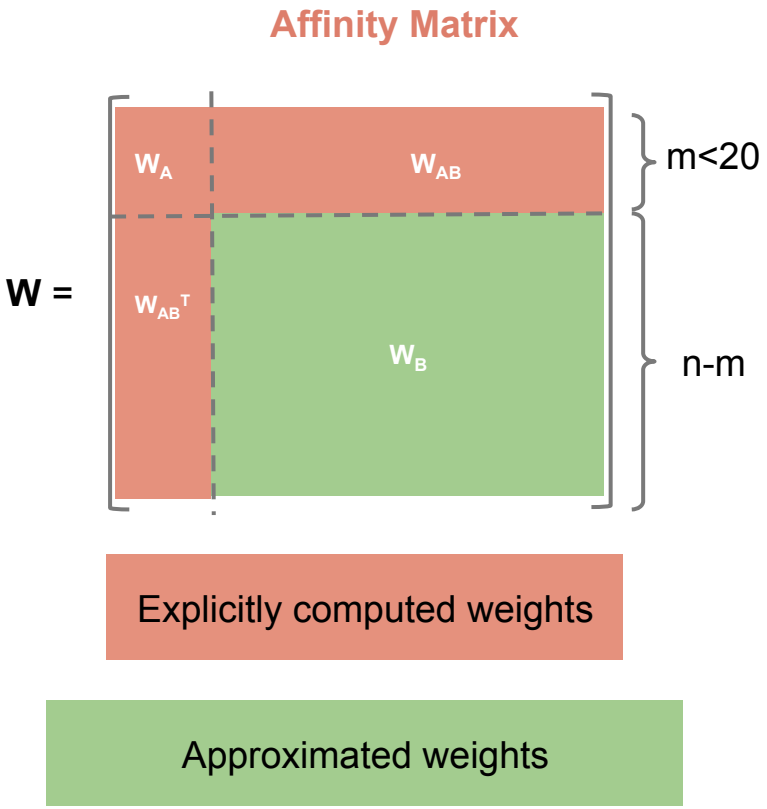
Nystrom Approximation



Spatially uniform sampling

A: Sampled pixels (m)

B: Remaining pixels (n-m)



Other filters from a base filter

Output Image

Input Image

$$\widehat{\mathbf{z}} = \mathbf{V} f(\mathbf{S}) \mathbf{V}^T \mathbf{y}$$

Filter controls

Shrinkage/Boosting function

$$f(\mathbf{S}) = \mathrm{diag}[f(\lambda_1), \ldots, f(\lambda_n)]$$

$$\widehat{\mathbf{z}} \approx \sum_i f(\alpha_i) \times \text{ top eigenvectors of } \mathbf{W}$$

Nonlinear Filtering is now "Linear"

Output Image

Input Image

$$\widehat{\mathbf{z}} = \mathbf{V} f(\mathbf{S}) \mathbf{V}^T \mathbf{y}$$

Filter controls

Shrinkage/Boosting function

$$f(\mathbf{S}) = \mathrm{diag}[f(\lambda_1), \ldots, f(\lambda_n)]$$

Diffusion:

(Perona-Malik '90, Coifman et al. '06, ….)

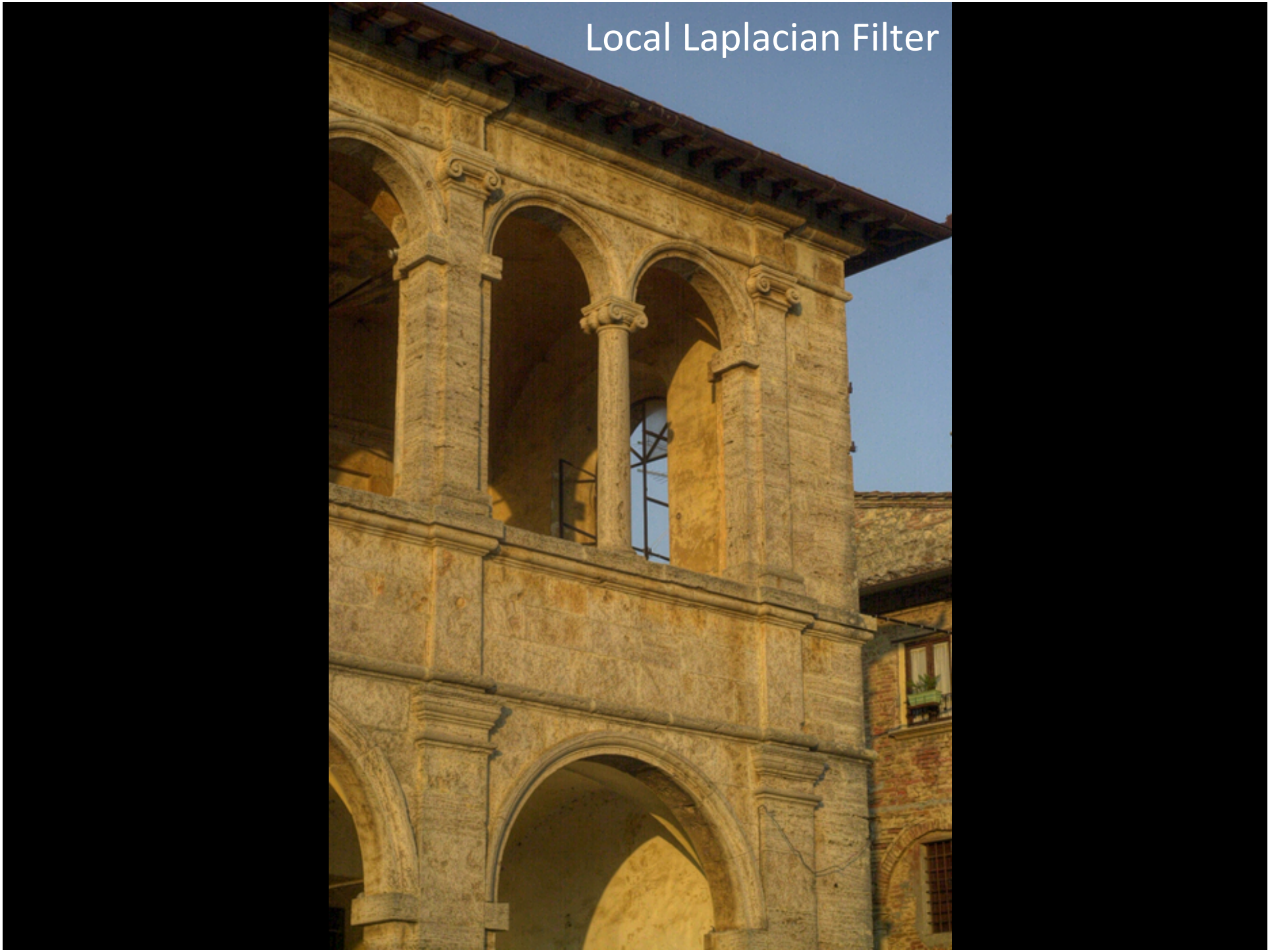
$$f(t) = t^p$$

Twicing, Reaction-Diffusion, Boosting, Bregman iteration:

(Tukey '77, Nordstrom '90, Buhlmann et al. '03, Osher et al. '05)

$$f(t) = 1 - (1-t)^p$$

p can be **any** positive real number


Local Laplacian Filter

Multi-frame Upscaling/Zoom

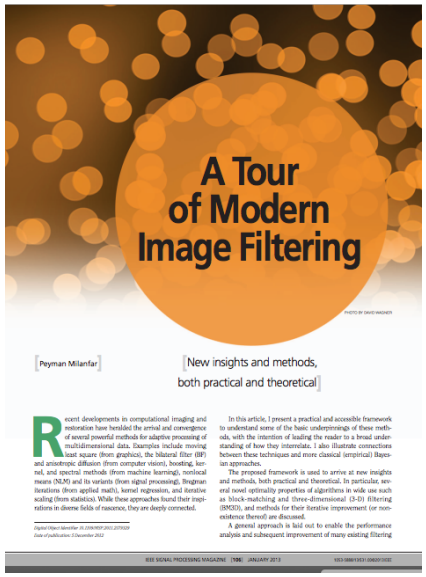

6FEN596

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

Thank you.