Technischen Universität München
Winter Semester 2014/2015

TRACKING and DETECTION in COMPUTER VISION

Filtering and edge detection

Slobodan Ilić

Overview

Image formation

Convolution

Non-liner filtering: Median and Bilateral filters

Gaussian smoothing

Image derivatives: Gradients and Laplacians

Edge detection

Corner and Region detection

Image formation

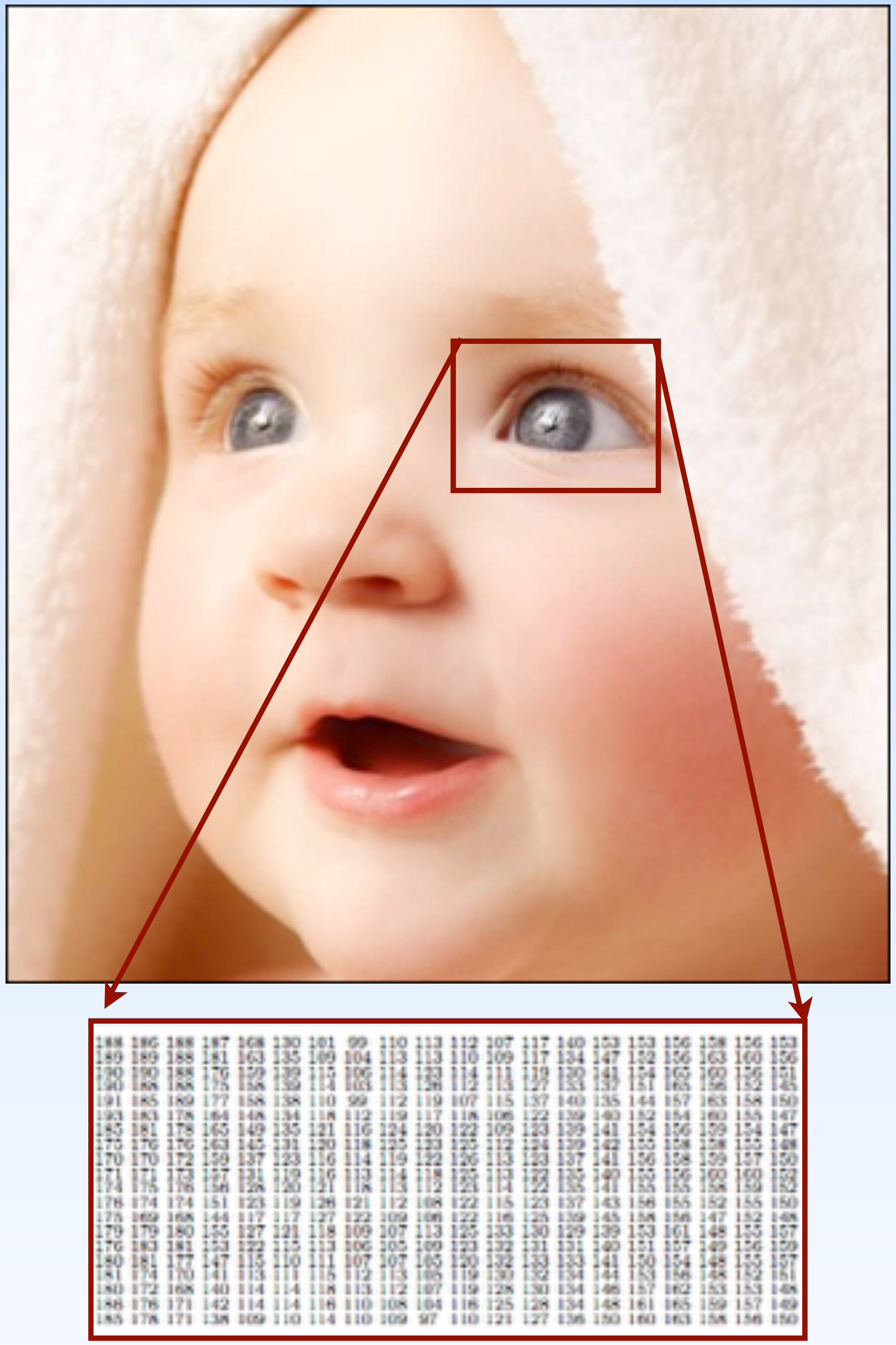


Image formation

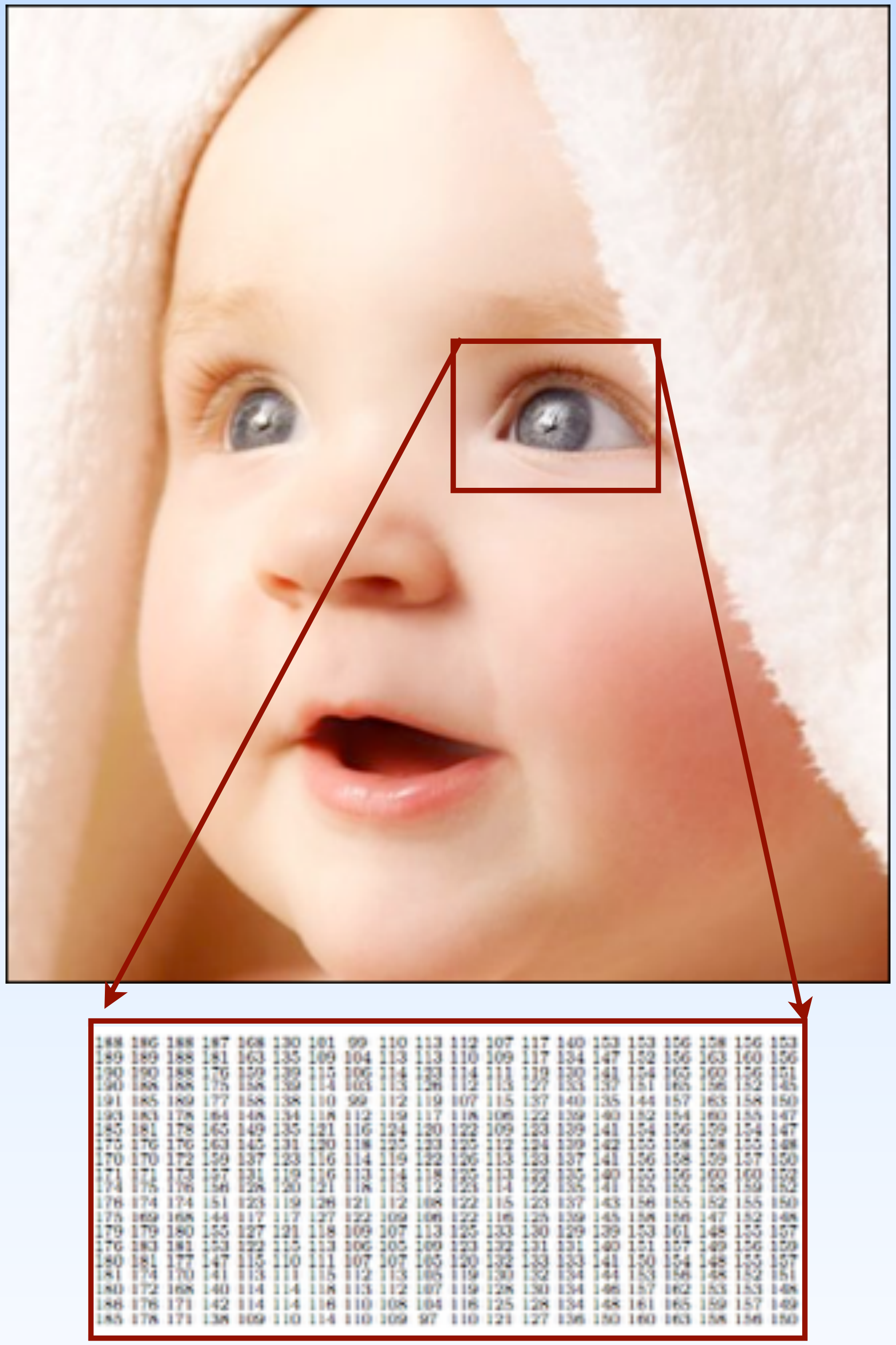


Image formation occurs when a sensor registers radiation that has interacted with physical objects.

It consists of several components:

Image formation

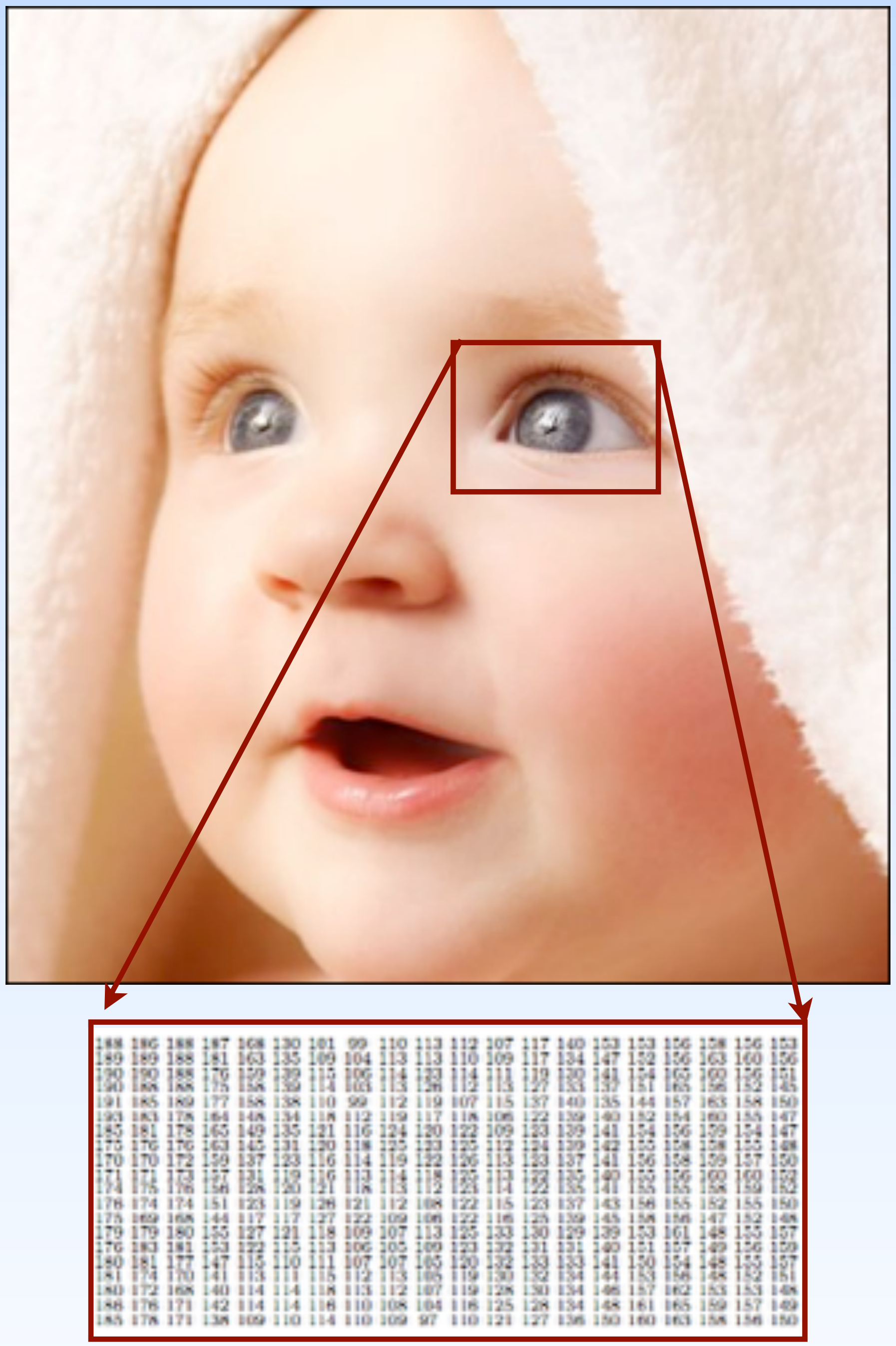


Image formation occurs when a sensor registers radiation that has interacted with physical objects.

It consists of several components:

• **An imaging function -** a fundamental abstraction of an image

Image formation

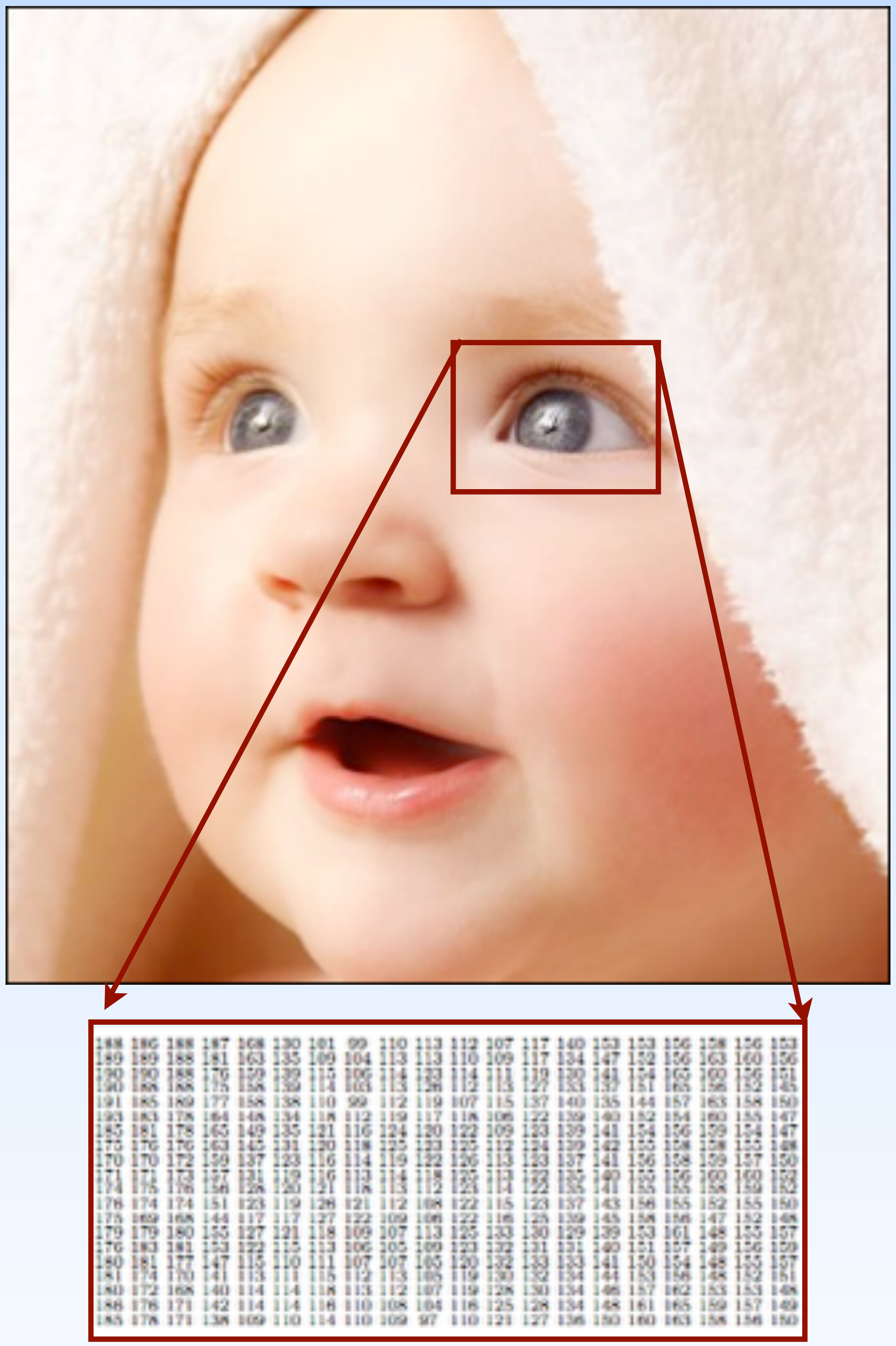


Image formation occurs when a sensor registers radiation that has interacted with physical objects.

It consists of several components:

• **An imaging function -** a fundamental abstraction of an image

• **A geometric model -** projection of the 3D world into 2D representation

Image formation

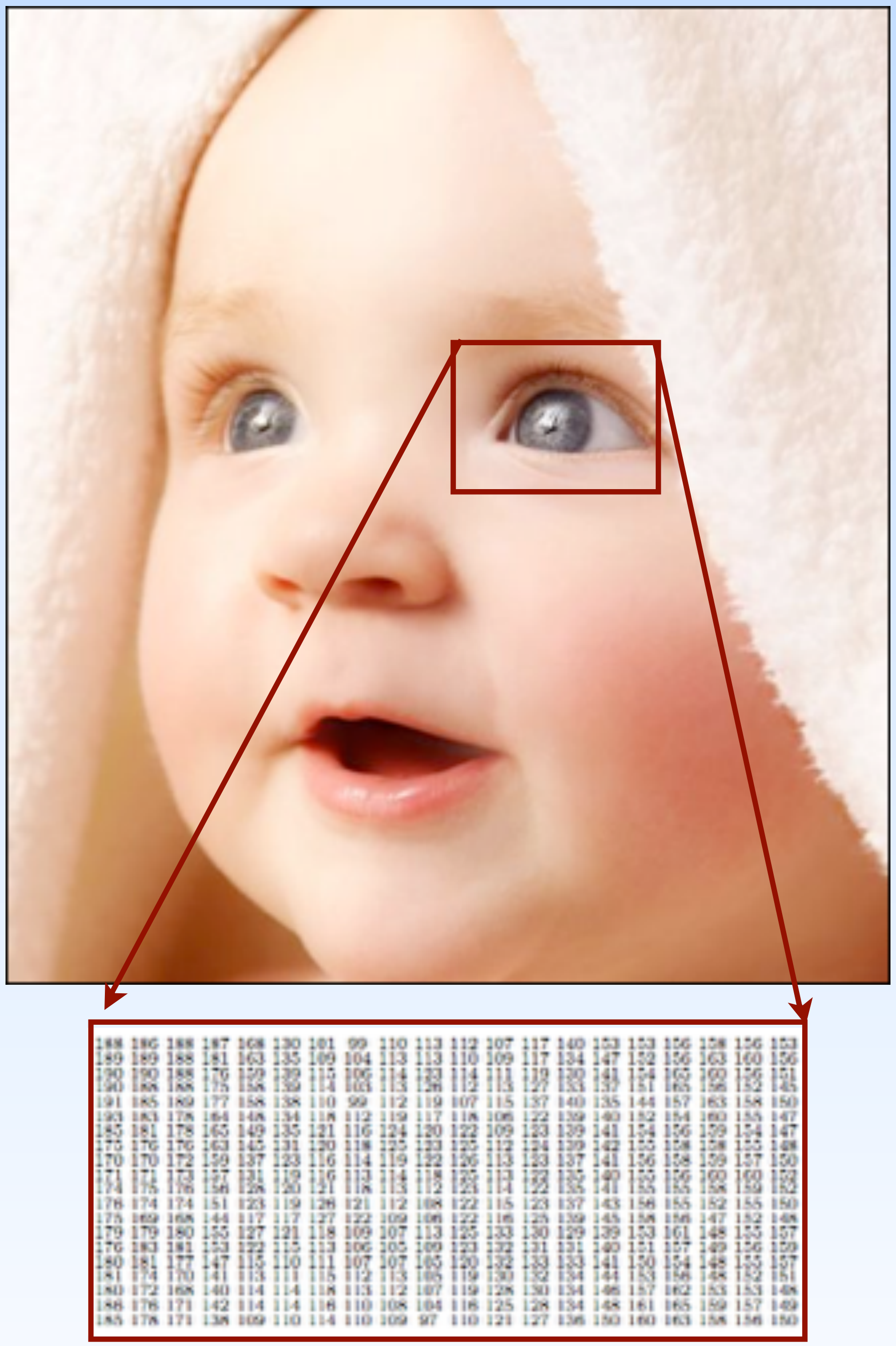


Image formation occurs when a sensor registers radiation that has interacted with physical objects.

It consists of several components:

- **An imaging function -** a fundamental abstraction of an image
- **A geometric model -** projection of the 3D world into 2D representation
- **A radiometric model -** describes how light radiated from the object is measured by the imaging sensor.

Image formation

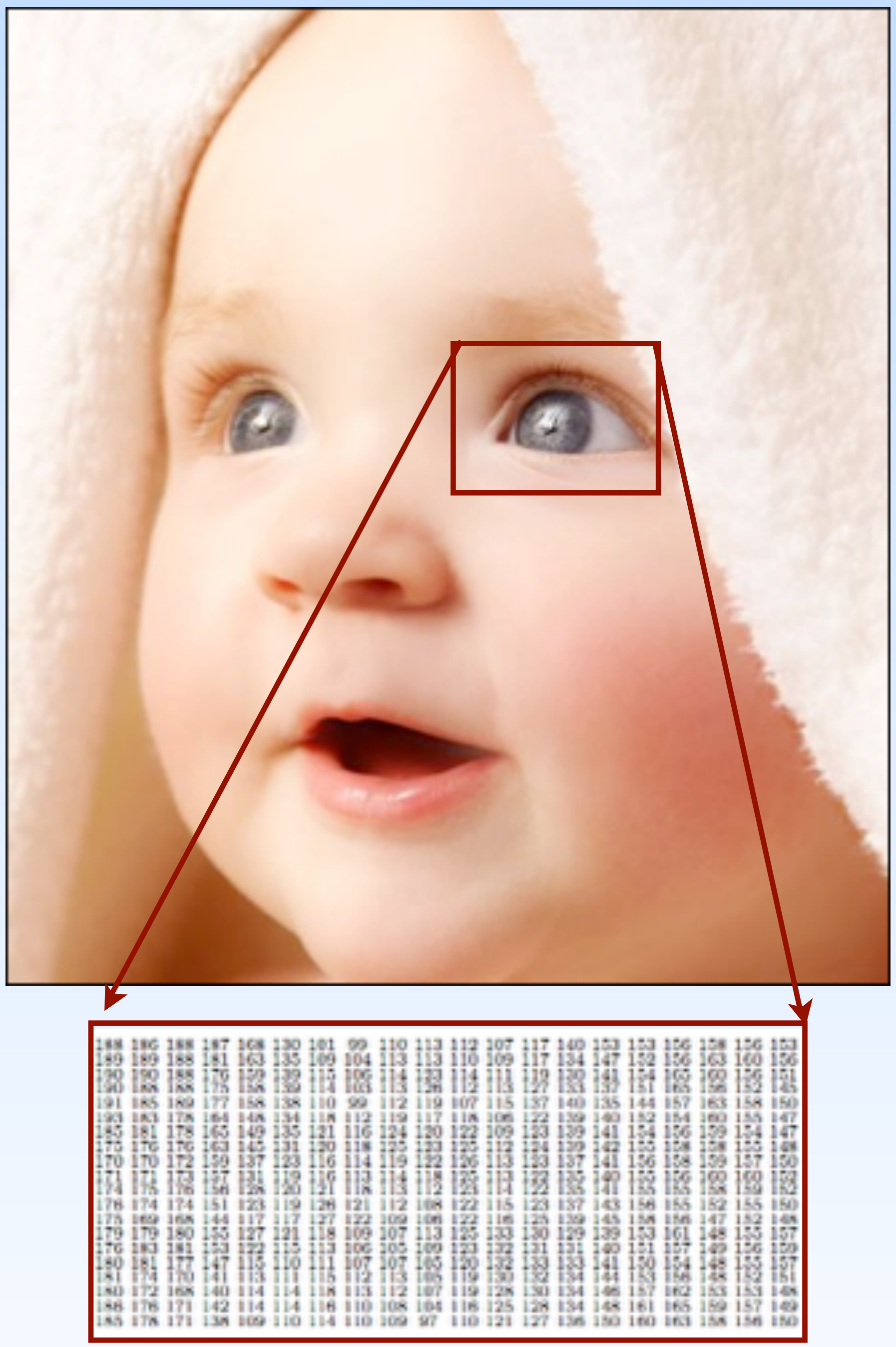


Image formation occurs when a sensor registers radiation that has interacted with physical objects.

It consists of several components:

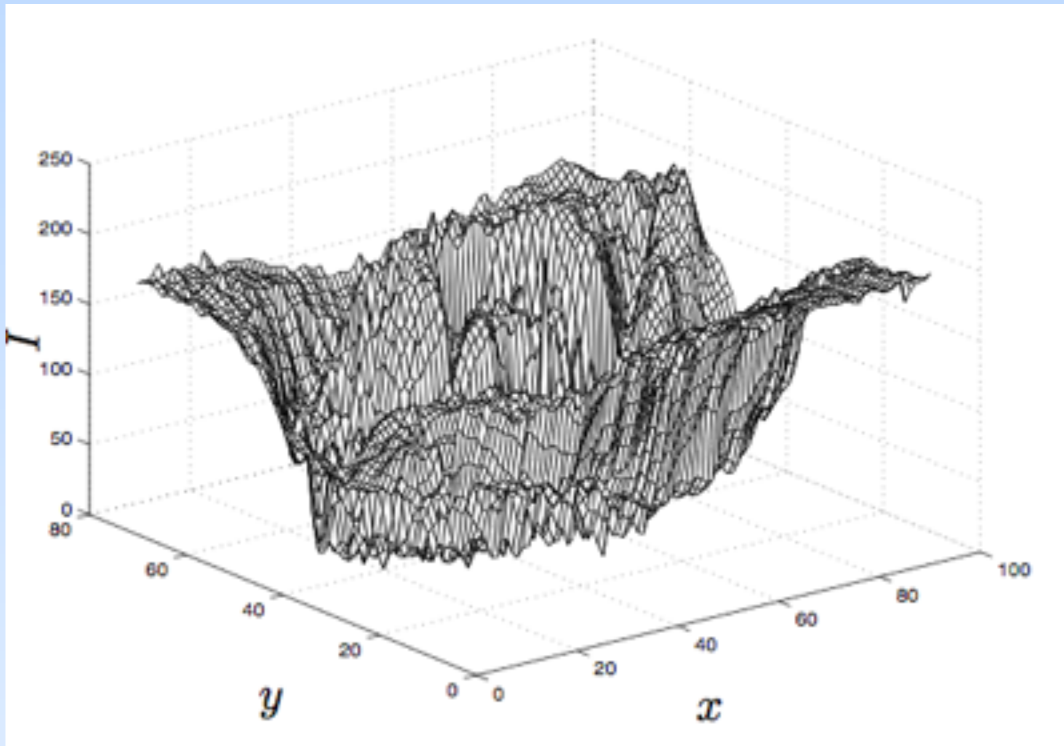
• **An imaging function -** a fundamental abstraction of an image

• **A geometric model -** projection of the 3D world into 2D representation

• **A radiometric model -** describes how light radiated from the object is measured by the imaging sensor.

• **A color model -** describes how different spectral measurements are related to image colors

Imaging function



188	186	188	187	168	130	101	99	110	113	112	107	117	140	153	153	156	158	156	153
189	189	188	181	163	135	109	104	113	113	110	109	117	134	147	152	156	163	160	156
190	190	188	176	159	139	115	106	114	123	114	111	119	130	141	154	165	160	156	151
190	188	188	175	158	139	114	103	113	126	112	113	127	133	137	151	165	156	152	145
191	185	189	177	158	138	110	99	112	119	107	115	137	140	135	144	157	163	158	150
193	183	178	164	148	134	118	112	119	117	118	106	122	139	140	152	154	160	155	147
185	181	178	165	149	135	121	116	124	120	122	109	123	139	141	154	156	159	154	147
175	176	176	163	145	131	120	118	125	123	125	112	124	139	142	155	158	158	155	148
170	170	172	159	137	123	116	114	119	122	126	113	123	137	141	156	158	159	157	150
171	171	173	157	131	119	116	113	114	118	125	113	122	135	140	155	156	160	160	152
174	175	176	156	128	120	121	118	113	112	123	114	122	135	141	155	155	158	159	152
176	174	174	151	123	119	126	121	112	108	122	115	123	137	143	156	155	152	155	150
175	169	168	144	117	117	127	122	109	106	122	116	125	139	145	158	156	147	152	148
179	179	180	155	127	121	118	109	107	113	125	133	130	129	139	153	161	148	155	157
176	183	181	153	122	115	113	106	105	109	123	132	131	131	140	151	157	149	156	159
180	181	177	147	115	110	111	107	107	105	120	132	133	133	141	150	154	148	155	157
181	174	170	141	113	111	115	112	113	105	119	130	132	134	144	153	156	148	152	151
180	172	168	140	114	114	118	113	112	107	119	128	130	134	146	157	162	153	153	148
186	176	171	142	114	114	116	110	108	104	116	125	128	134	148	161	165	159	157	149
185	178	171	138	109	110	114	110	109	97	110	121	127	136	150	160	163	158	156	150

I is an imaging function defined on the area Ω and takes values in the positive real numbers:

$$I : \Omega \subset \mathbb{R} \to \mathbb{R}_+; (x, y) \mapsto I(x, y).$$

Because of the discretization we have:

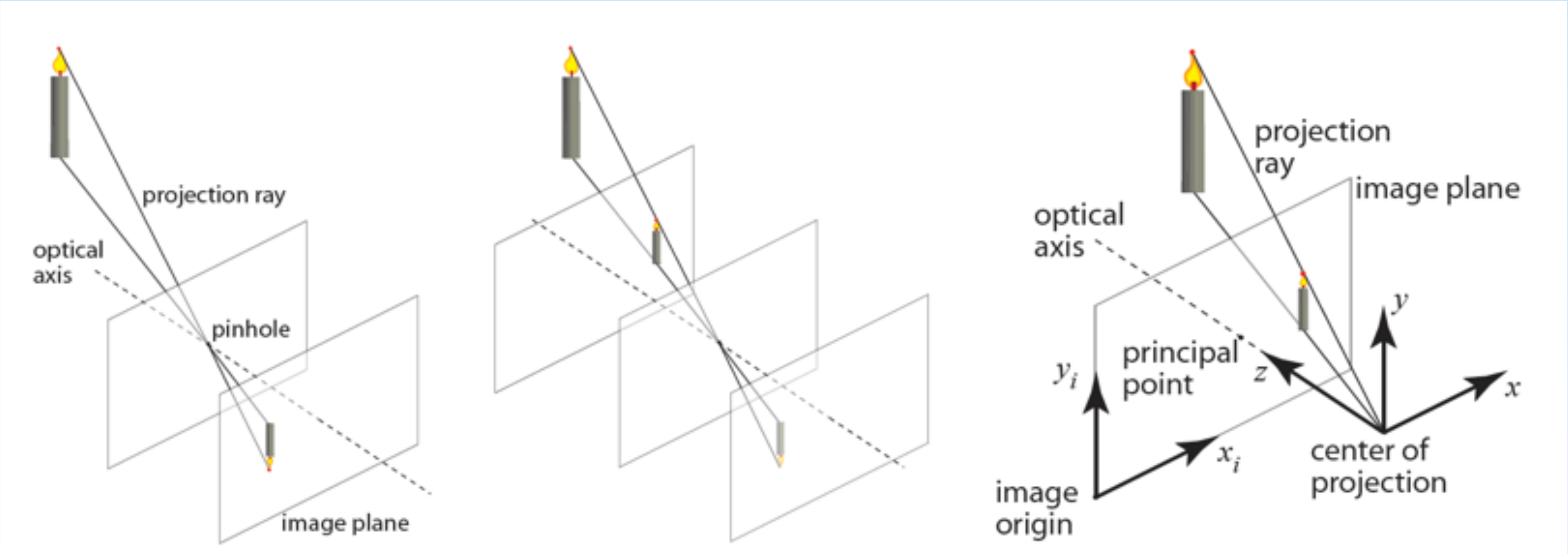
$$\Omega = [1, 640] \times [1, 480] \subset \mathbb{Z}^2, \quad \mathbb{R}_+ = [0, 255] \subset \mathbb{Z}_+$$

Or in general:

$$I = I(x, y) = \{I_R(x, y), I_G(x, y), I_B(x, y)\}$$

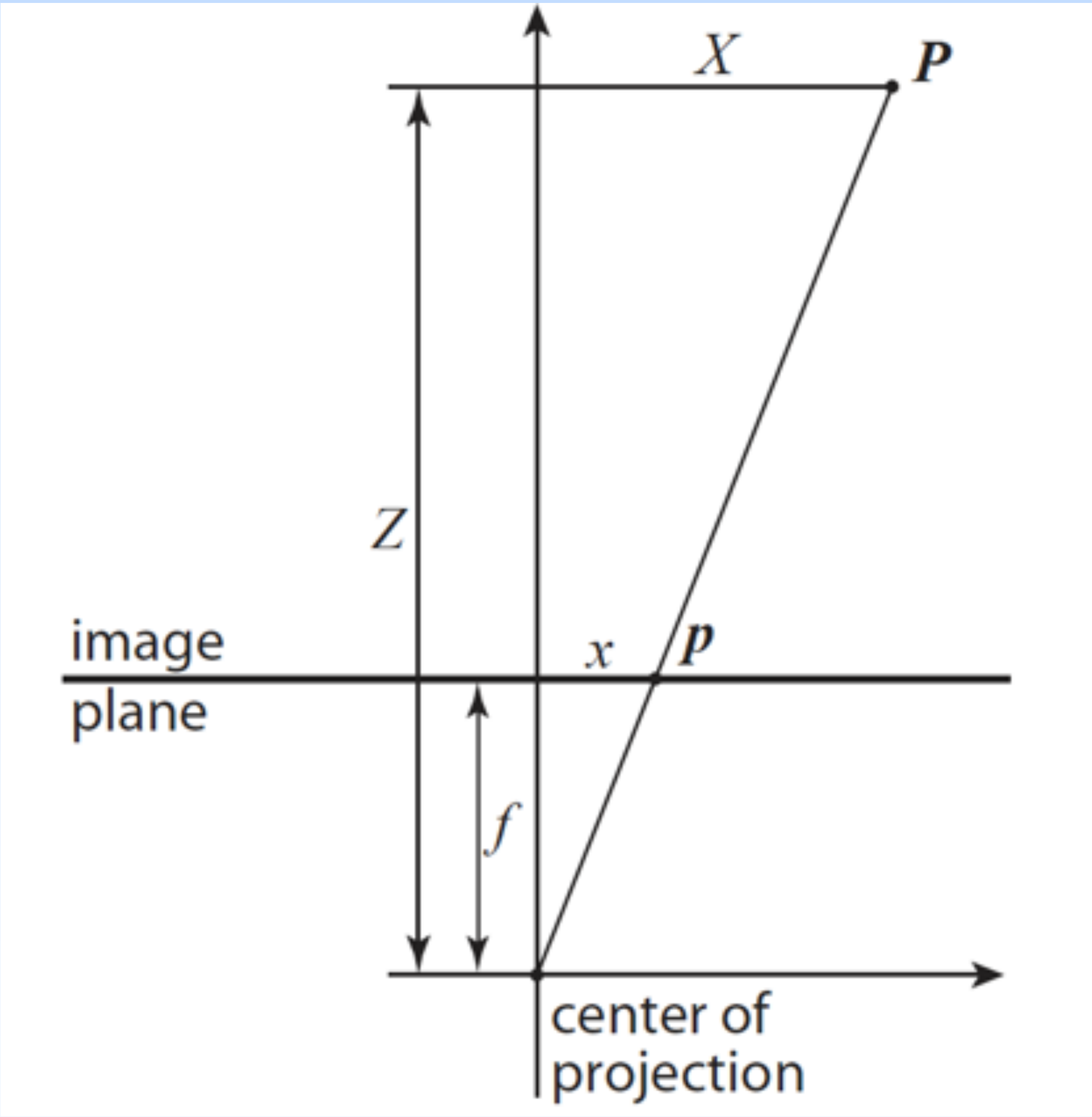
Geometric model

The pinhole camera model

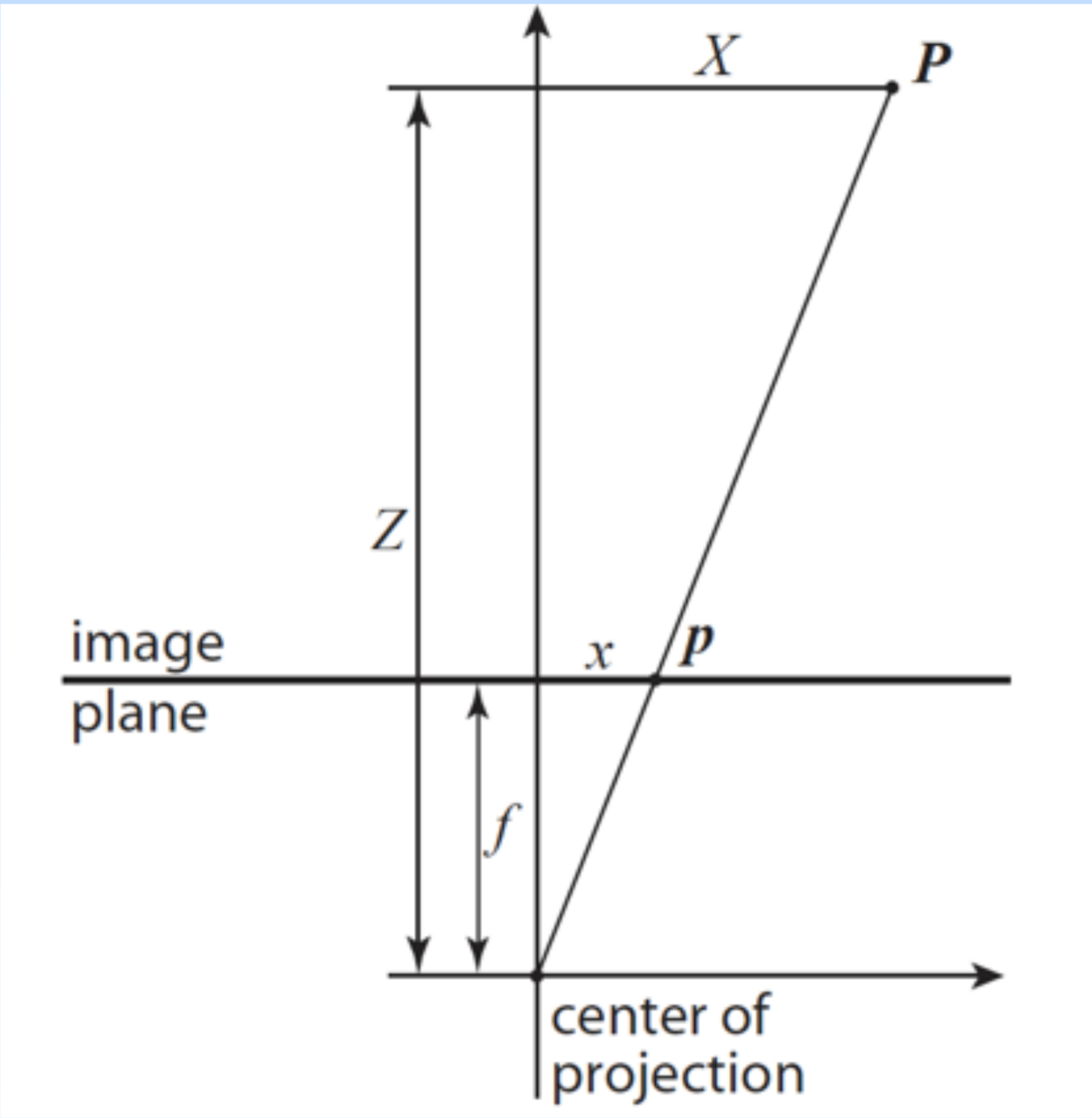


(a) Projection geometry (b) Image plane in front. (c) Notations for pinhole camera model

Perspective projection



Perspective projection

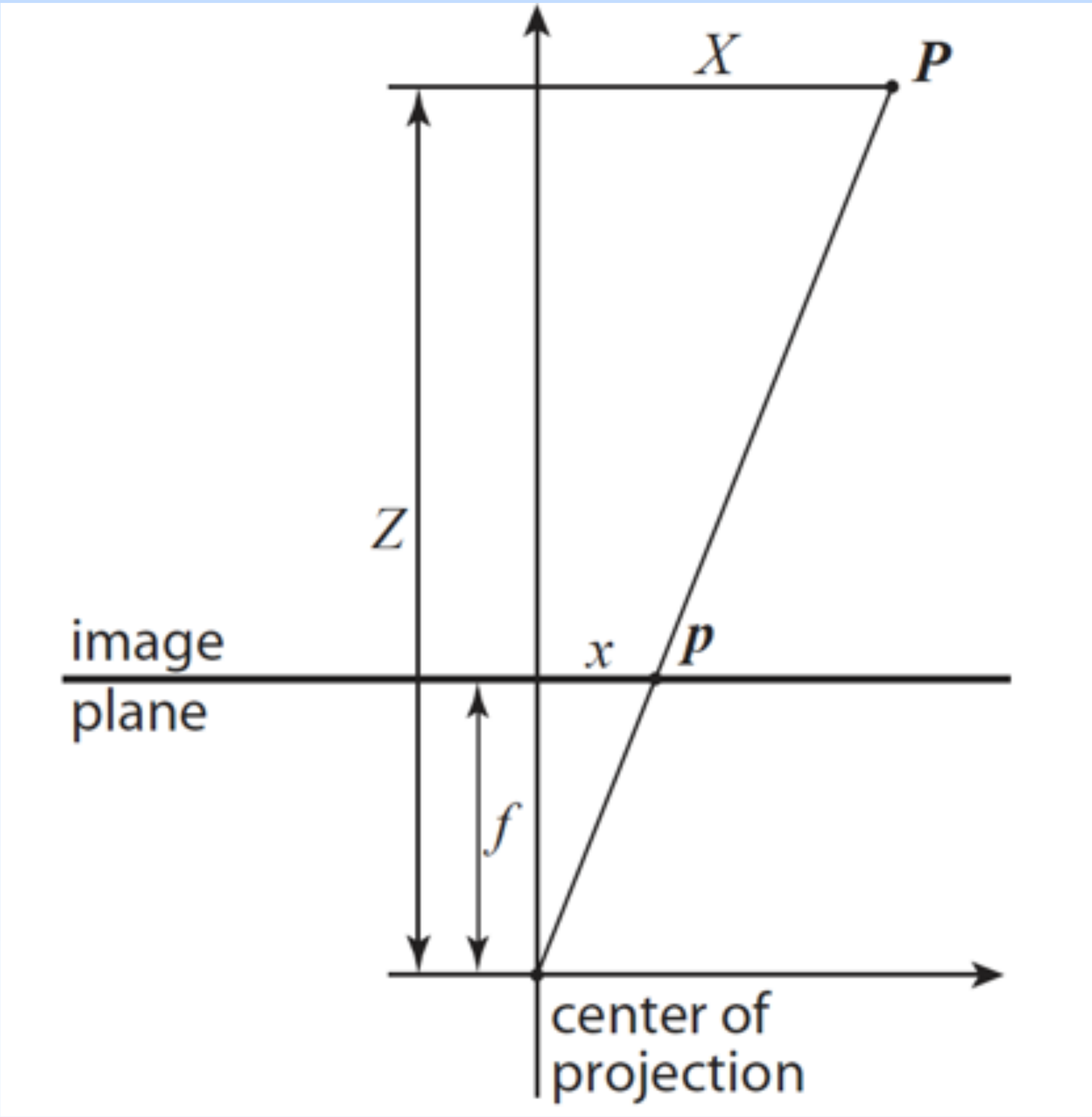


Wold coordinate system is identical to the camera coordinate system with the origin at the center of projection.

Under perspective projection, the object point with world coordina-tes projects to the image point with ***ideal image coordinates***:

Ideal image coordinates are expressed in terms of the image coordinate system with the origin at the optical center.

Perspective projection



Wold coordinate system is identical to the camera coordinate system with the origin at the center of projection.

Under perspective projection, the object point with world coordinates $P = (X, Y, Z)$ projects to the image point with ***ideal image coordinates***:

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

Ideal image coordinates are expressed in terms of the image coordinate system with the origin at the optical center.

Image processing

Image processing is in general not part of Computer Vision, but it is sometimes a very necessary preprocessing step.

Image processing provides a number of methods to convert an image into something suitable for analysis.

There are two main approaches:

- Processing in the spacial domain:
    1. Point processing (brightness, contrast, histogram equalization etc.)
    2. Filtering
- Processing in the frequency domain: Fourier transform

Image processing

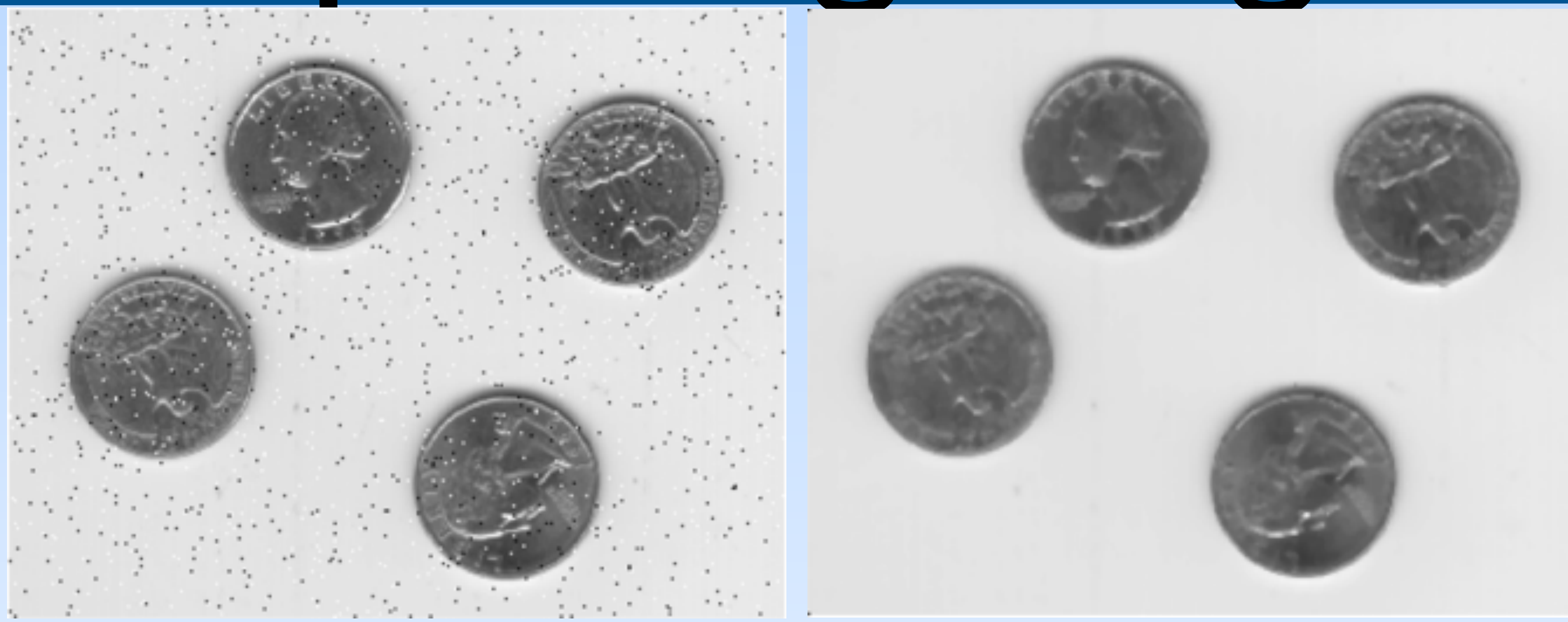
Image processing is in general not part of Computer Vision, but it is sometimes a very necessary preprocessing step.

Image processing provides a number of methods to convert an image into something suitable for analysis.

There are two main approaches:

- Processing in the spacial domain:
    1. Point processing (brightness, contrast, histogram equalization etc.)
    2. Filtering
- Processing in the frequency domain: Fourier transform

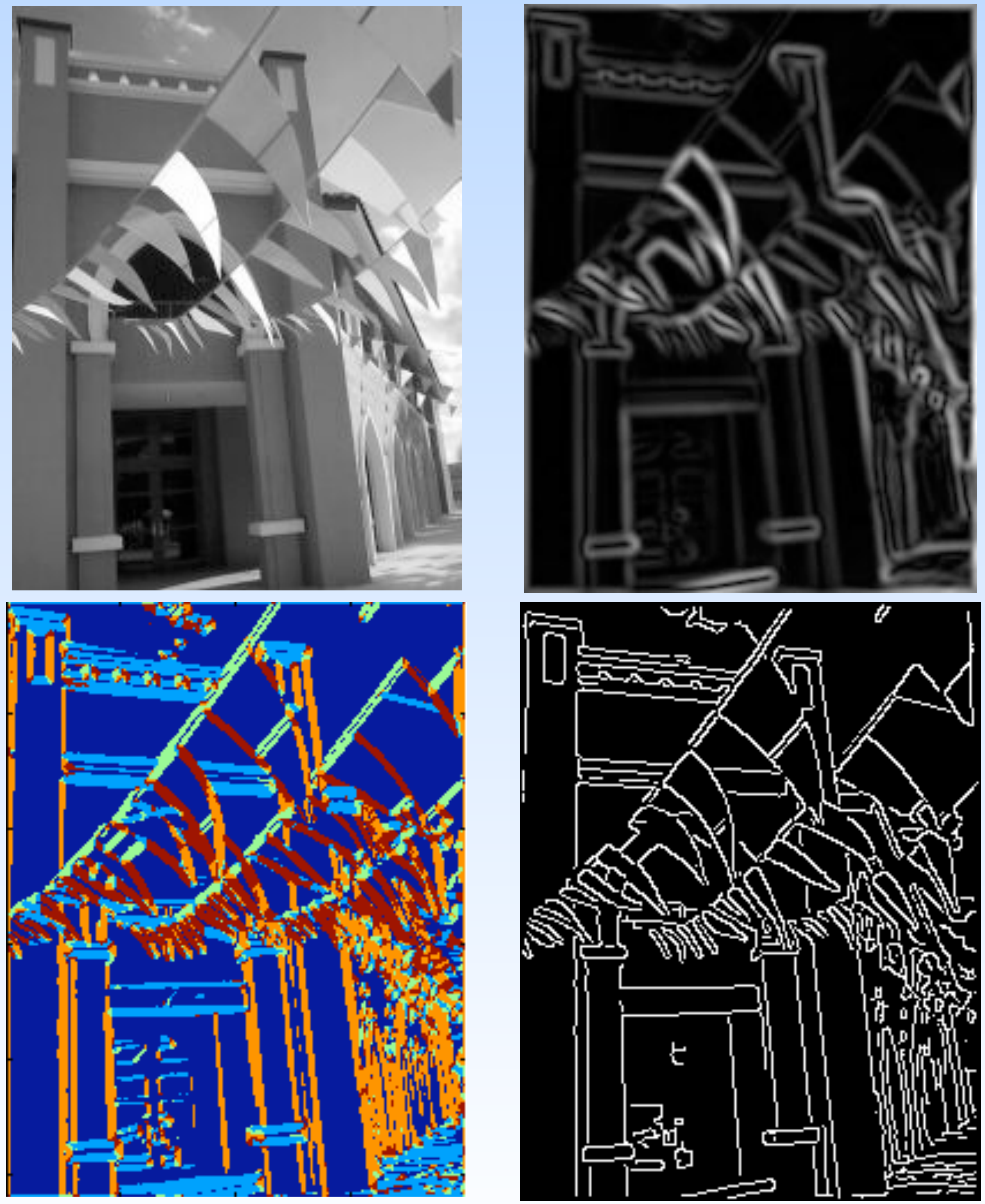
Improving images



Noise removal

Contrast increase

Detecting edges



Steps in edge detection

Filtering

Original image

Blurred image

From the draft of Szeliski's book: http://research.microsoft.com/en-us/um/people/szeliski/Book/

Filtering is based on neighborhood operations.

Correlation and Convolution

These are two simplest filtering operations and they have two key properties:

- **shift invariance** (the same operation is performed at every location in the image)
- **linearity** (every pixel is replaced with the linear combination of its neighbors)

Neighborhood filtering
Correlation and Convolution

From the draft of Szeliski's book: http://szeliski.org/Book/

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$I(x,y), 1 \leq x,y \leq x_{dim}, y_{dim}$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$H(i,j), -1 \leq i,j \leq 1$

convolution kernel

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$J(x,y)$

$$J(x,y) = H \circ I = \sum_{i=-1}^{1} \sum_{j=-1}^{1} H(i,j) I(x+i, y+j)$$ - correlation

$$J(x,y) = H * I = \sum_{i=-1}^{1} \sum_{j=-1}^{1} H(i,j) I(x-i, y-j)$$ - convolution

Convolution Example

H

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

Rotate

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

Apply correlation

$$J(x,y) = H \circ I = \sum_{i=-1}^{1} \sum_{j=-1}^{1} H(i,j) I(x+i, y+j)$$

I

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

Apply convolution

$$J(x,y) = H * I = \sum_{i=-1}^{1} \sum_{j=-1}^{1} H(i,j) I(x-i, y-j)$$

Convolution Example

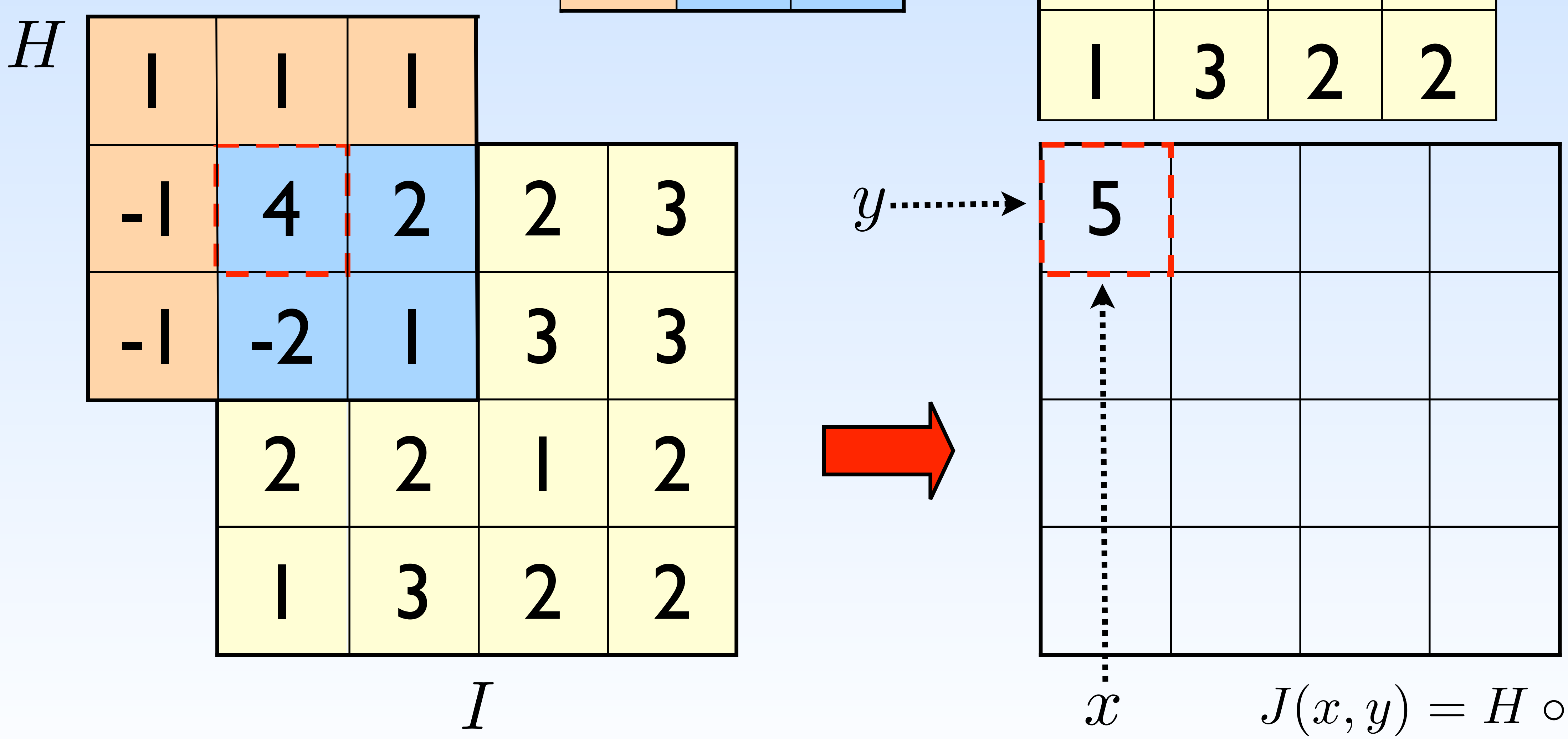
Step 1

H

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

I

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



$J(x, y) = H \circ I$

Convolution Example

Step 2

H

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

I

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

H

1	1	1	
-2	4	2	3
-2	-1	3	3
2	2	1	2
1	3	2	2

I

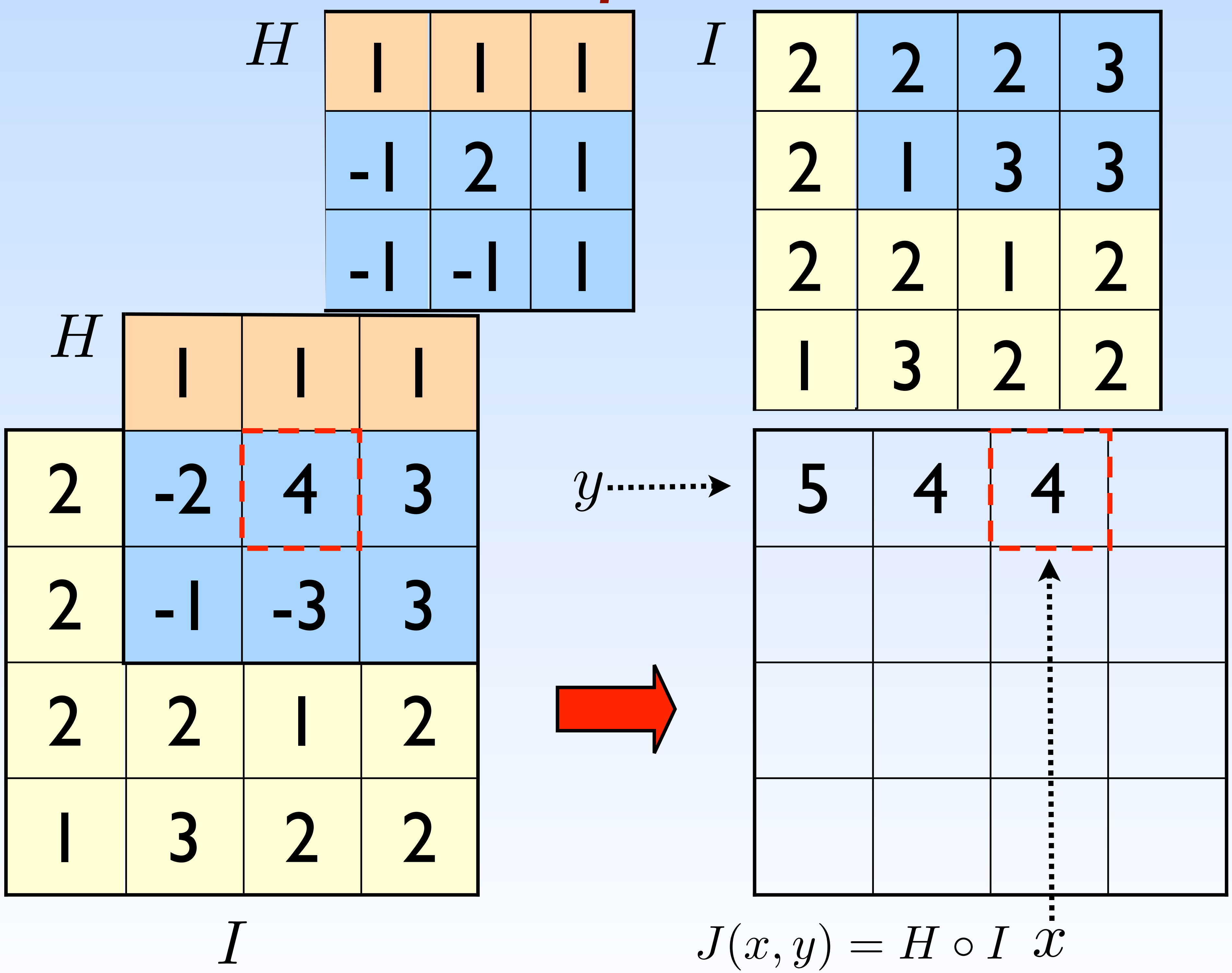
y

5	4		

x $J(x,y) = H \circ I$

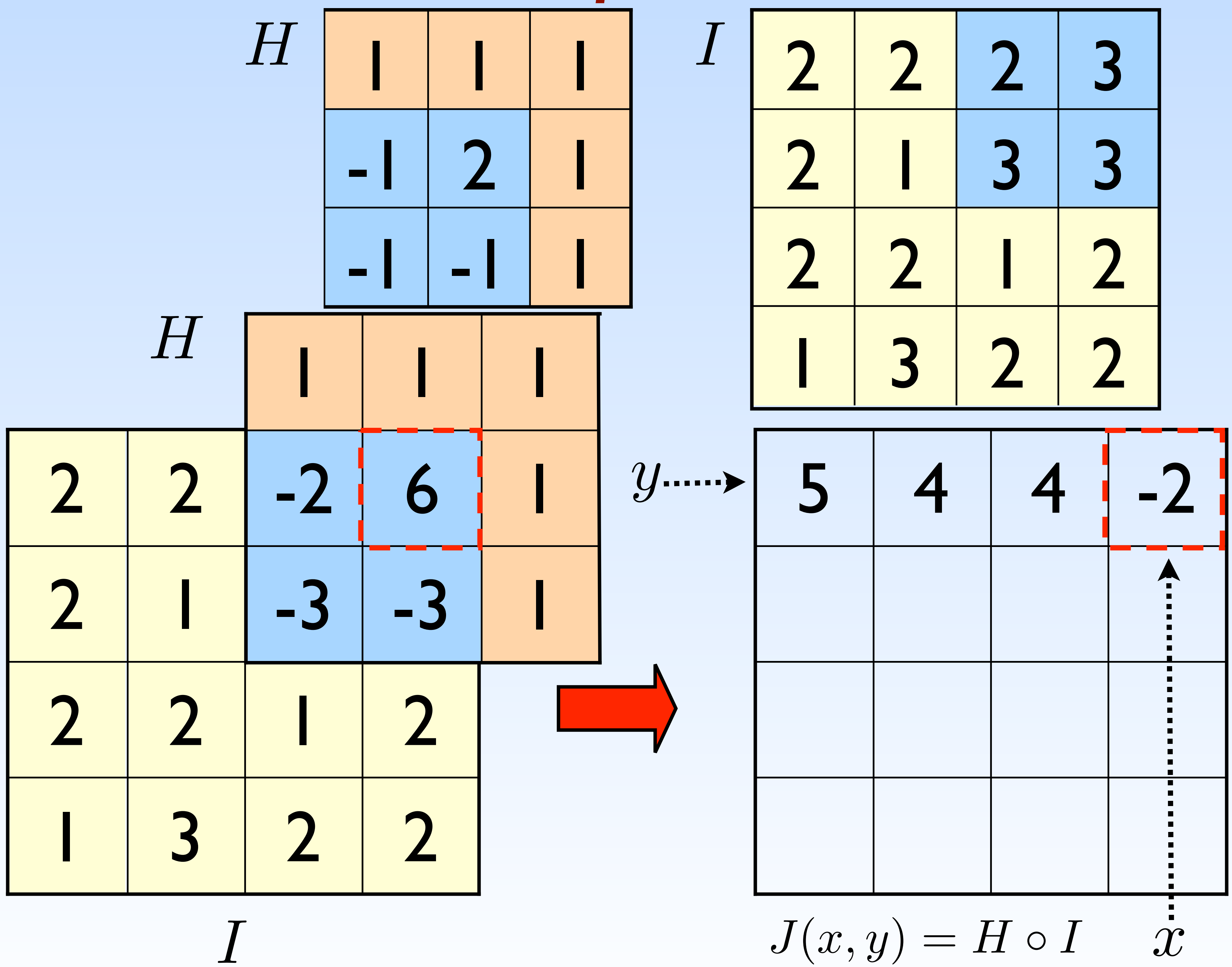
Convolution Example

Step 3



Convolution Example

Step 4



Convolution Example

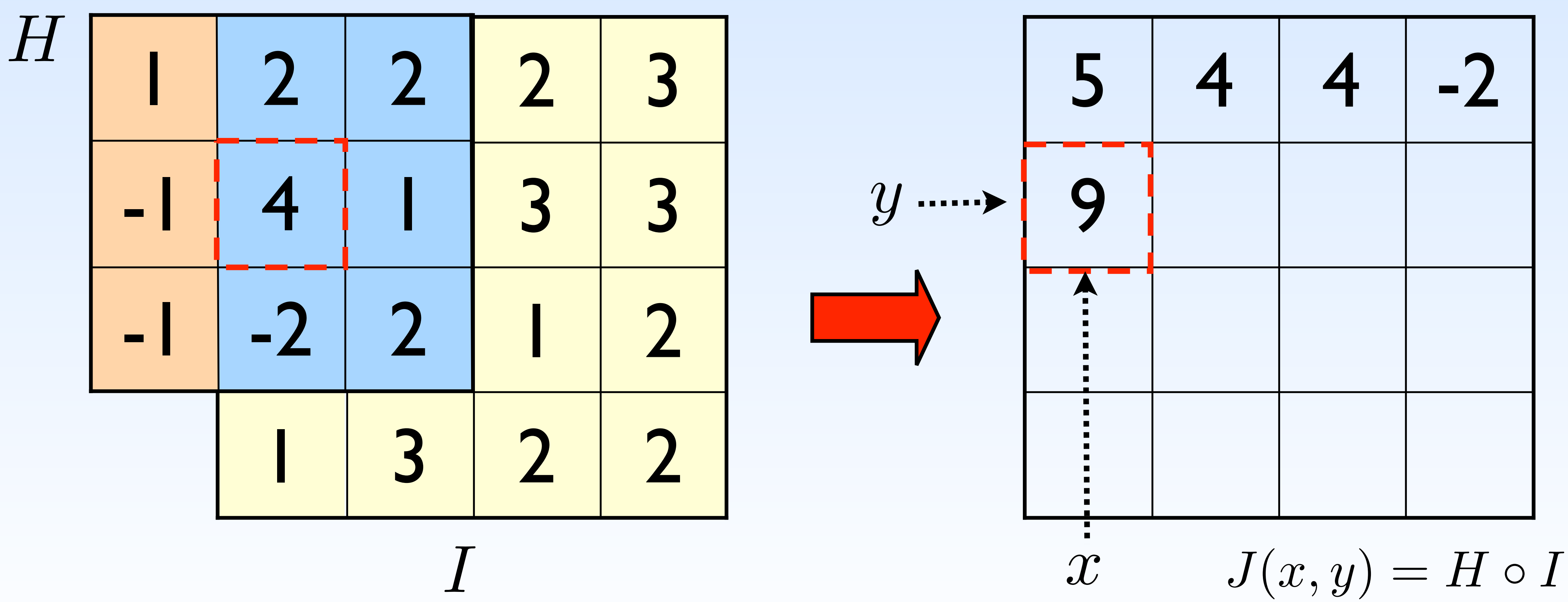
Step 5

H

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

I

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



$J(x, y) = H \circ I$

Convolution Example

Step 6

H

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

I

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

H

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

I

y

5	4	4	-2
9	6		

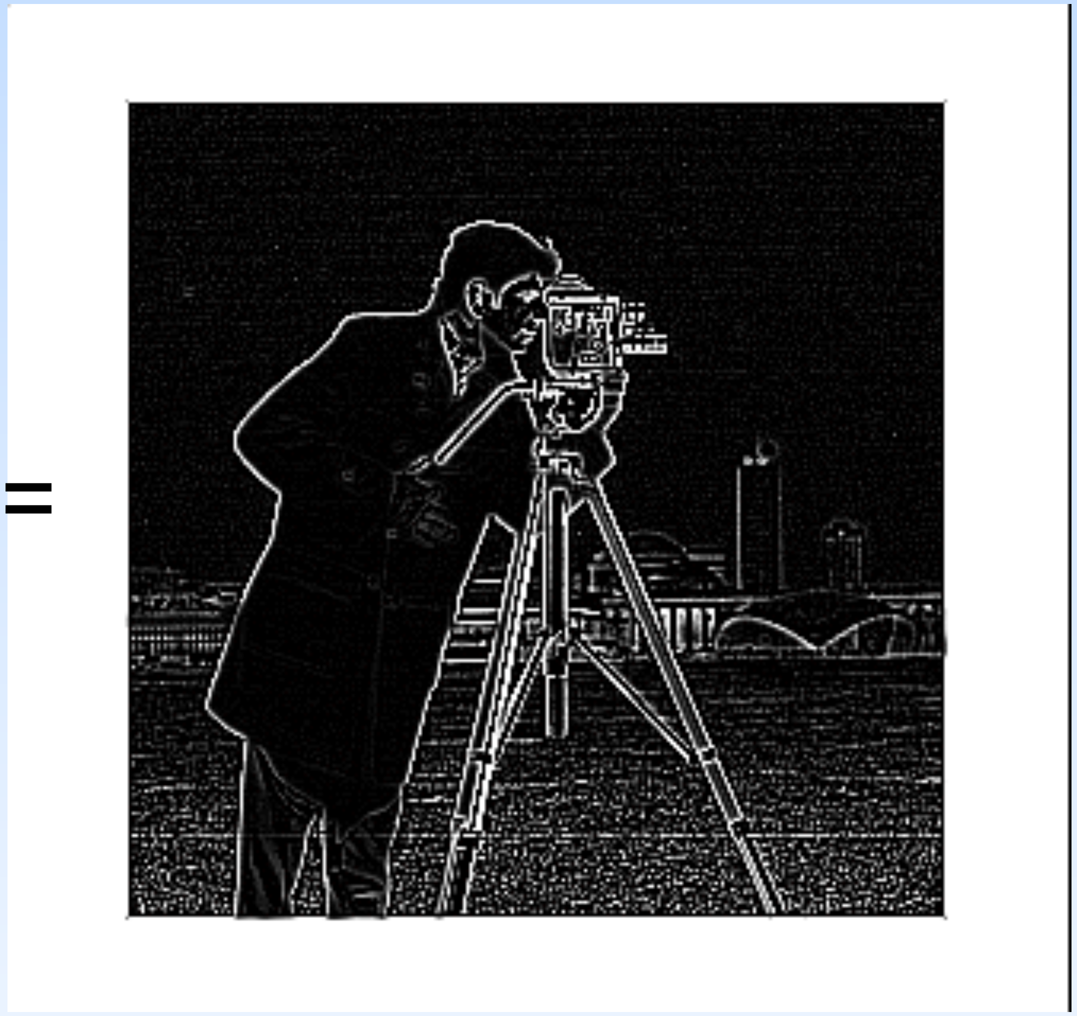
x $J(x, y) = H \circ I$

Example

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

Example

$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



Expressing convolution

Mathematically convolution can be expresses as:

$$J(x, y) = H * I = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} H(i, j) I(x-i, y-j)$$

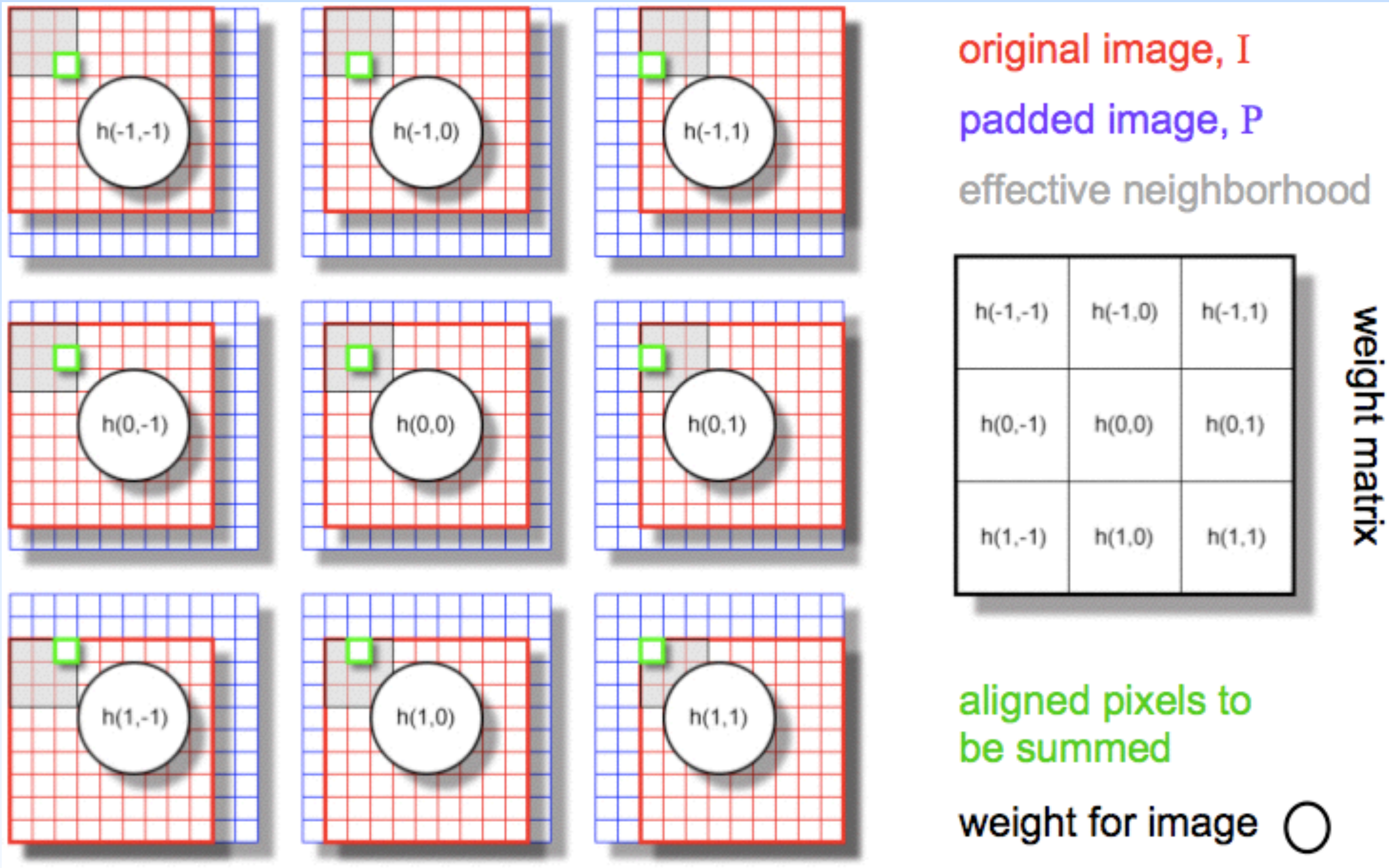
if we change the variables $i \leftarrow x - i$ and $j \leftarrow y - j$ we get:

$$J(x, y) = H * I = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} H(x-i, y-j) I(i, j)$$

with the image and the kernel playing interchangeable roles.

Convolution by shifting, copying and multiplying the image

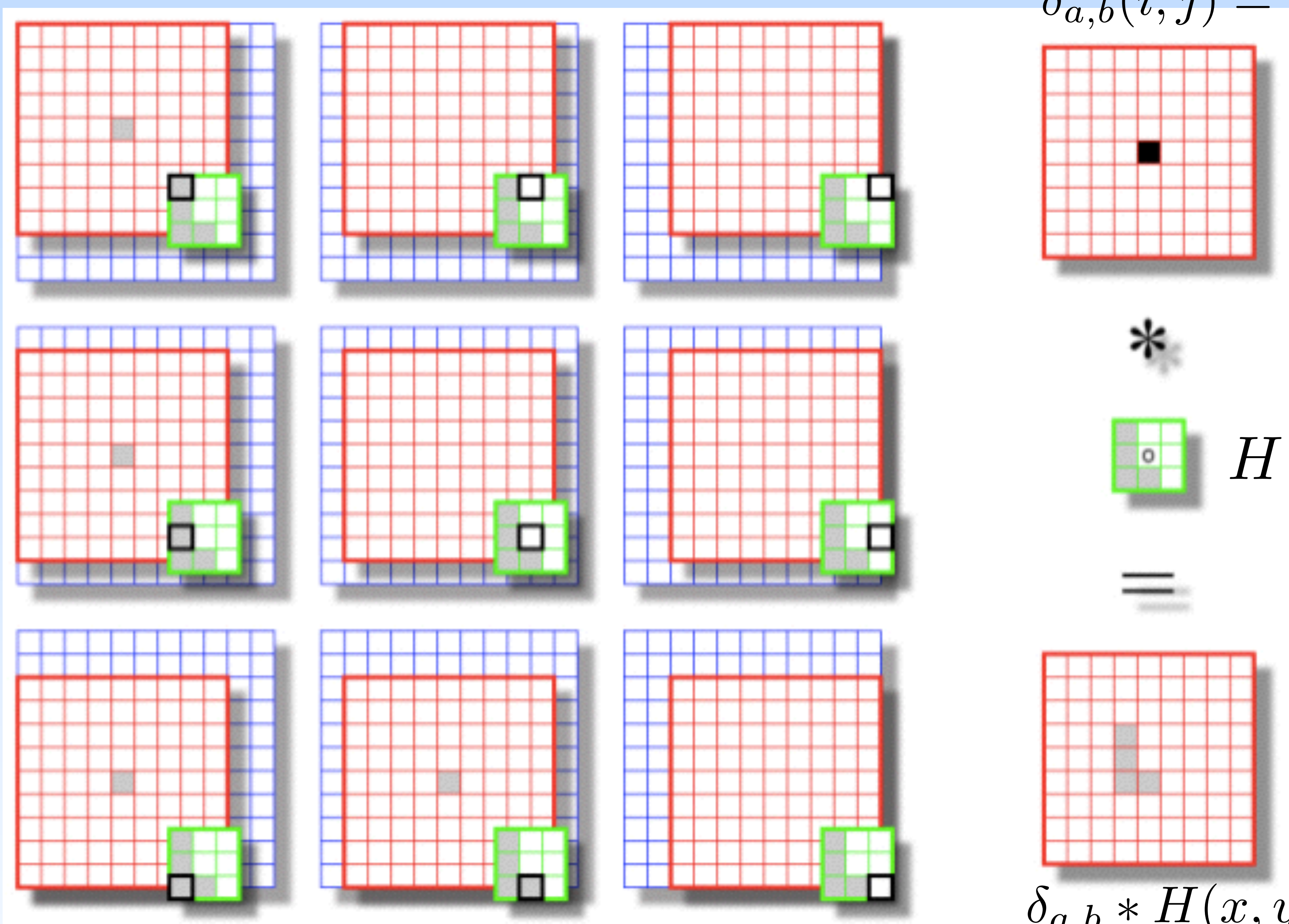
For each element H(i,j) in the kernel matrix, image I is copied into a zero-padded image, P starting at (i,j). Each P is multiplied by the corresponding weight H(i,j). All the P images are summed pixel-wise, then divided by the sum of the elements of h. The result is copped out of the center of the accumulated P's.



$$J(x,y) = H * I = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} H(x-i, y-j) I(i,j)$$

Shift invariance

$$\delta_{a,b}(i,j) = \begin{cases} 1, & for \;\; i = a \;\; and \;\; j = b \\ 0, & elsewhere \end{cases}$$



$$\delta_{a,b} * H(x,y) = H(x-a, c-b)$$

The kernel(point spread function) H is shift invariant or space-invariant because the entries in H do not depend on the position (x, y) in the output image and the same operation is performed at each location in the image.

Convolution properties

Commutativity

$$H * I = I * H$$

Associativity

In correlation

$$\boxed{H * (G * I) = (H * G) * I} \qquad H \circ (G \circ I) \neq (H \circ G) \circ I$$

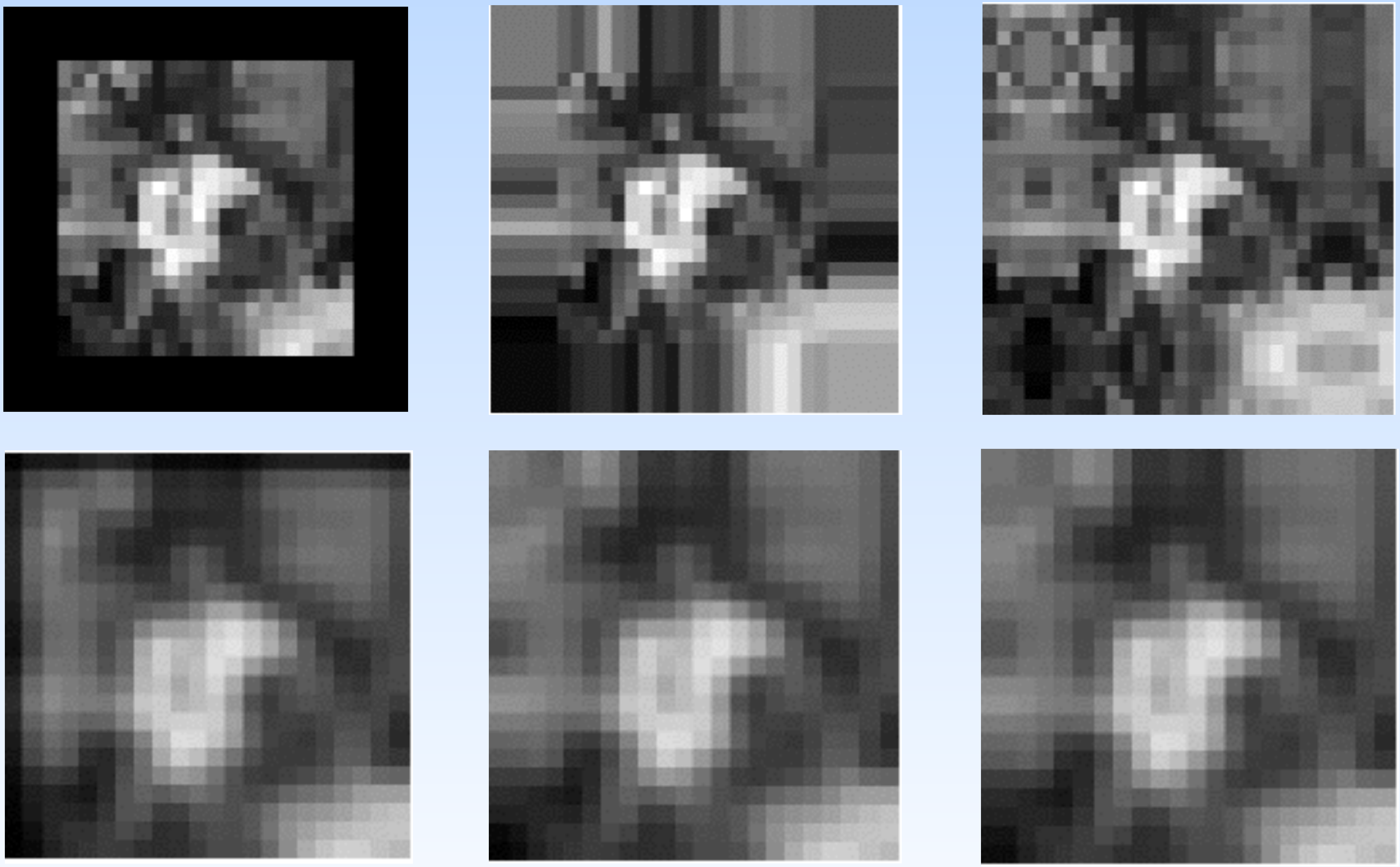
Distributivity

$$H * (I + J) = (H * I) + (H * J)$$

Associativity with scalar multiplication

$$a(H * I) = (aH) * I = H * (aI)$$

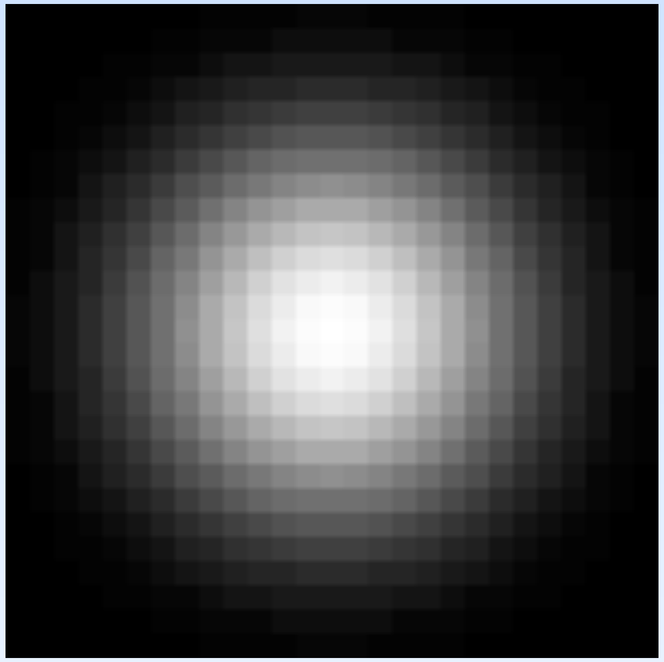
Padding (border effects)



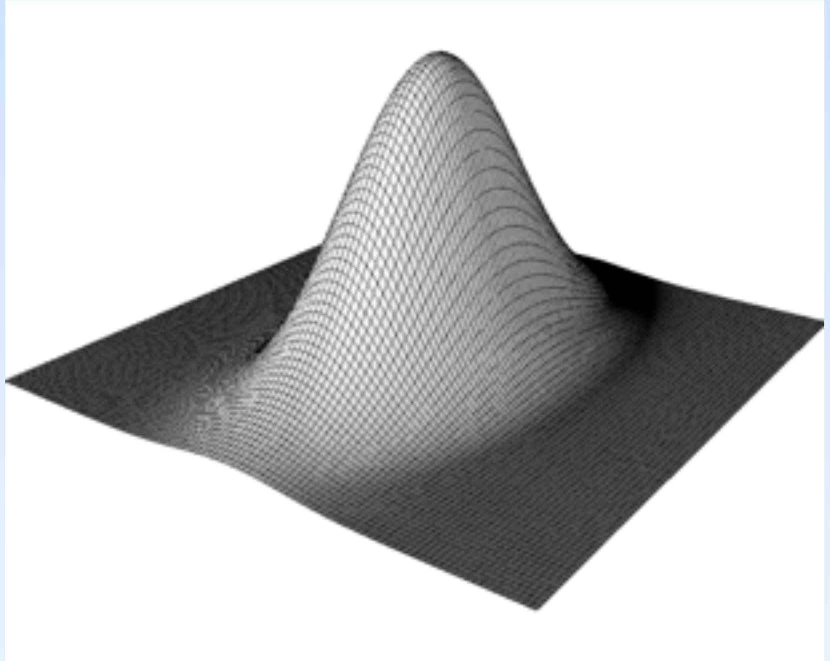
zero clamp mirror

Gaussian smoothing

$$G(u,v) = e^{-\frac{1}{2}\frac{u^2+v^2}{\sigma^2}}$$



Gaussian kernel



Gaussian function

Gaussian smoothing example

Original image | Smoothed $\sigma = 1$ | Smoothed $\sigma = 2$ | Smoothed $\sigma = 4$

Separability of Gaussians

$$G(u,v) = g(u)g(v)$$

$$G(u,v) = e^{-\frac{1}{2}\frac{u^2+v^2}{\sigma^2}} \implies G(u,v) = e^{-\frac{1}{2}\left(\frac{u}{\sigma}\right)^2} e^{-\frac{1}{2}\left(\frac{v}{\sigma}\right)^2}$$

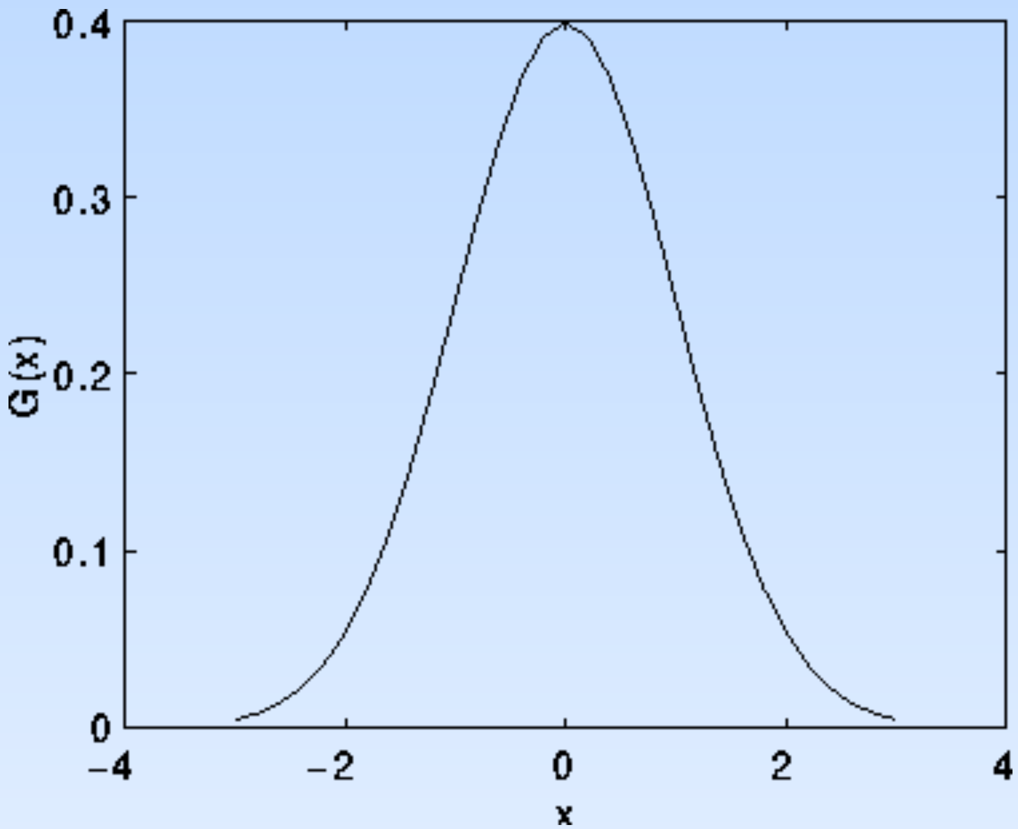
Gaussian separation speeds up computation, because with the separable kernel G the convolution can be separated in two one-dimensional convolutions like:

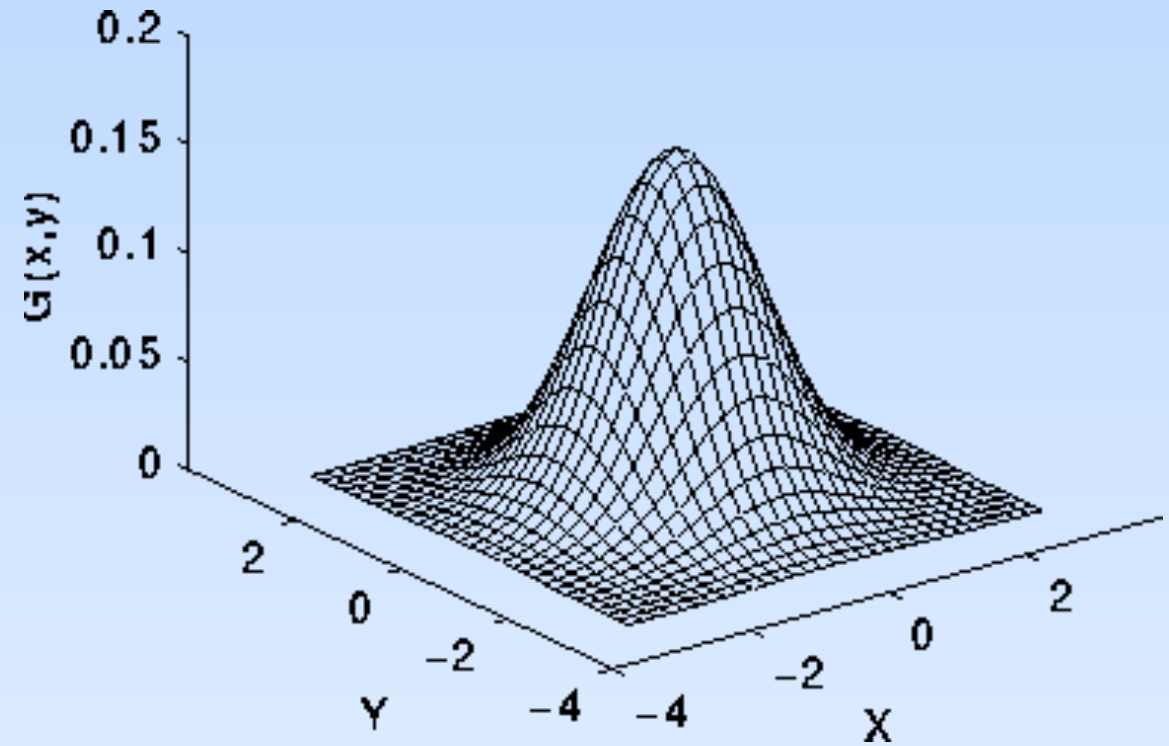
$$J(x,y) = \sum_{u=-n}^{n} g(u) \sum_{v=-n}^{n} g(v) I(x-u, y-v)$$

$$J(x,y) = \sum_{u=-n}^{n} g(u)\phi(x-u,y) \qquad \phi(x,y) = \sum_{v=-n}^{n} g(v) I(x, y-v)$$

this requires 2m multiplications and 2(m-1) additions compared to m^2 multiplications and m^2-1 additions with 2D kernel, where m=2n+1.

Gaussian normalization





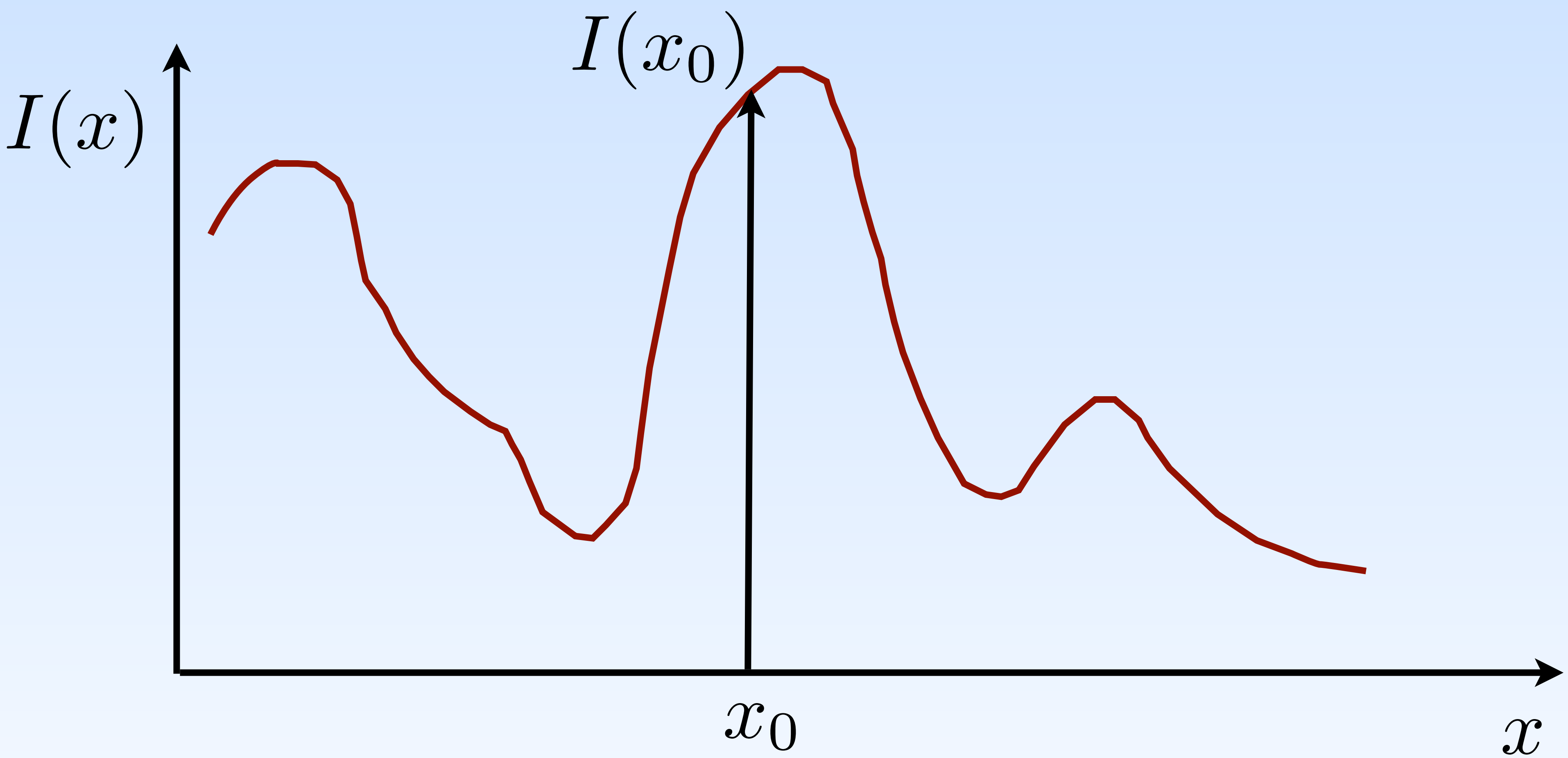
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \qquad G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

Since we discretize in practice we normalize not by the integral, but by the sum

$$G(x,y) = \frac{1}{c} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \qquad c = \sum_{i=-n}^{n}\sum_{j=-n}^{n} G(i,j)$$

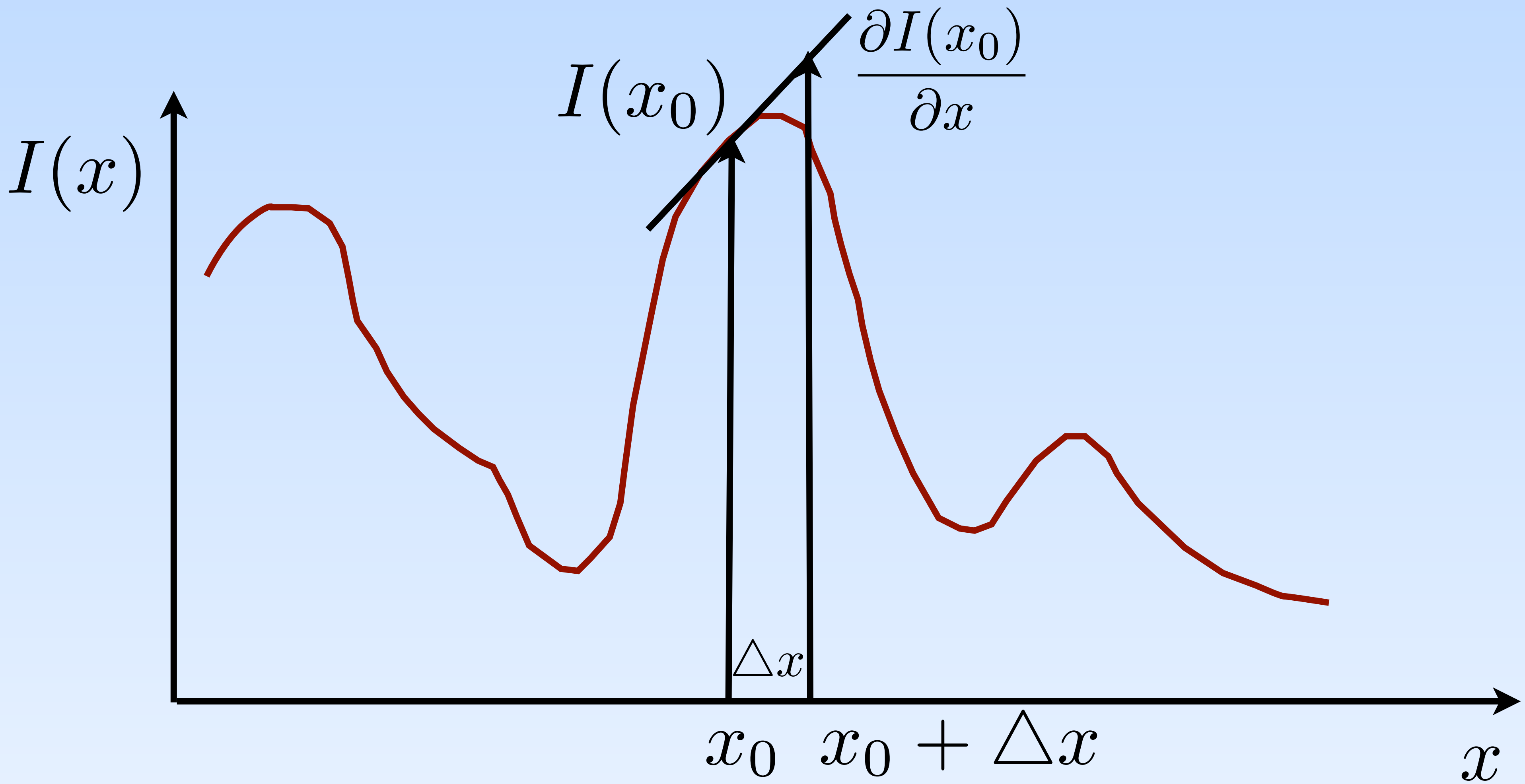
Image derivatives

Remember that the image can be represented as a function of its pixel intensities



consequently we should be able to compute derivatives of this function ...

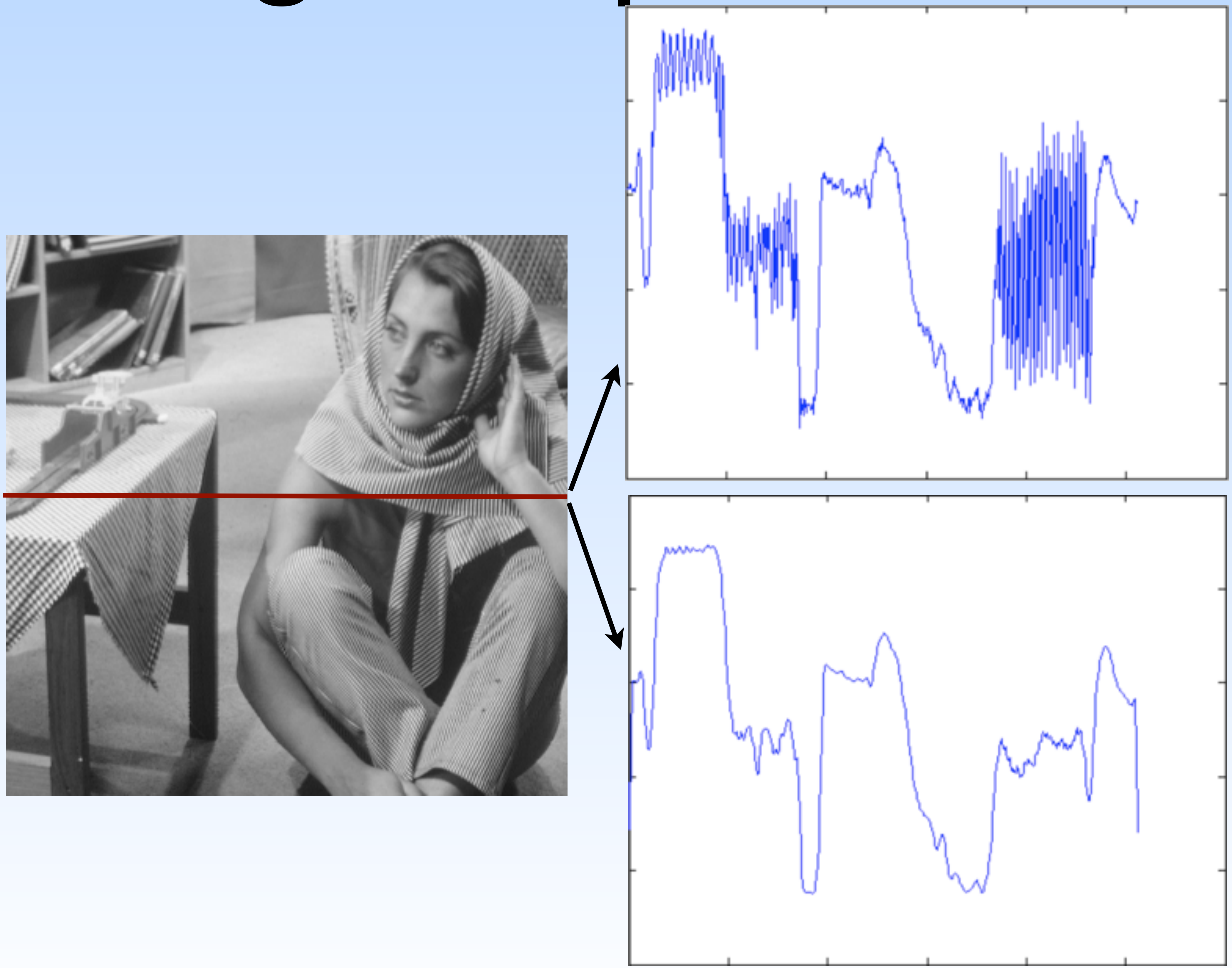
Linear approximation



$$I(x_0 + \triangle x) \approx I(x_0) + \frac{\partial I(x_0)}{\partial x} \triangle x + h.o.t$$

$$\frac{\partial I(x_0)}{\partial x} = \lim_{\triangle x \to 0} \frac{I(x_0 + \triangle x) - I(x_0)}{\triangle x} \approx I(x_0 + \triangle x) - I(x_0)$$

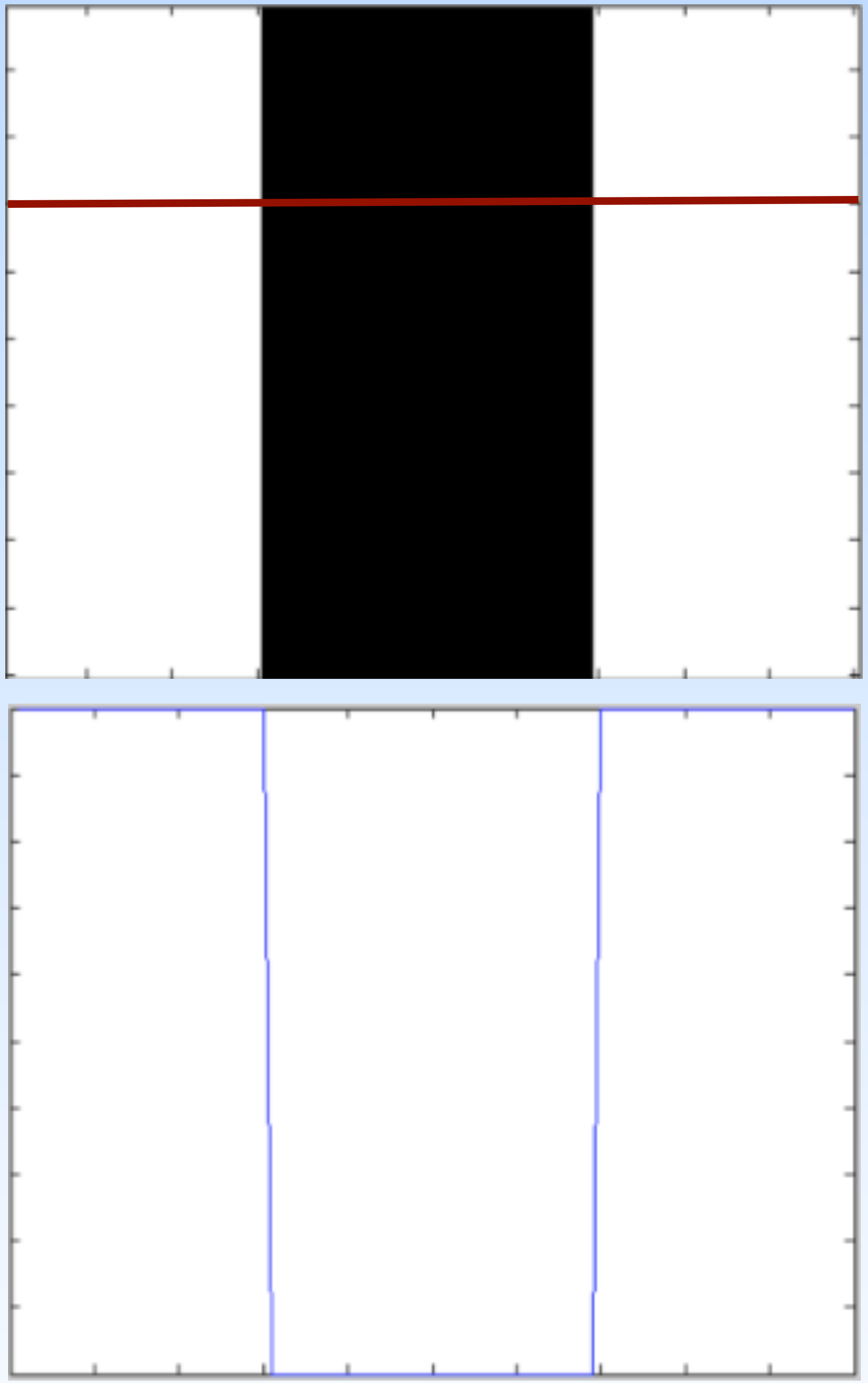
Image 1D profile example



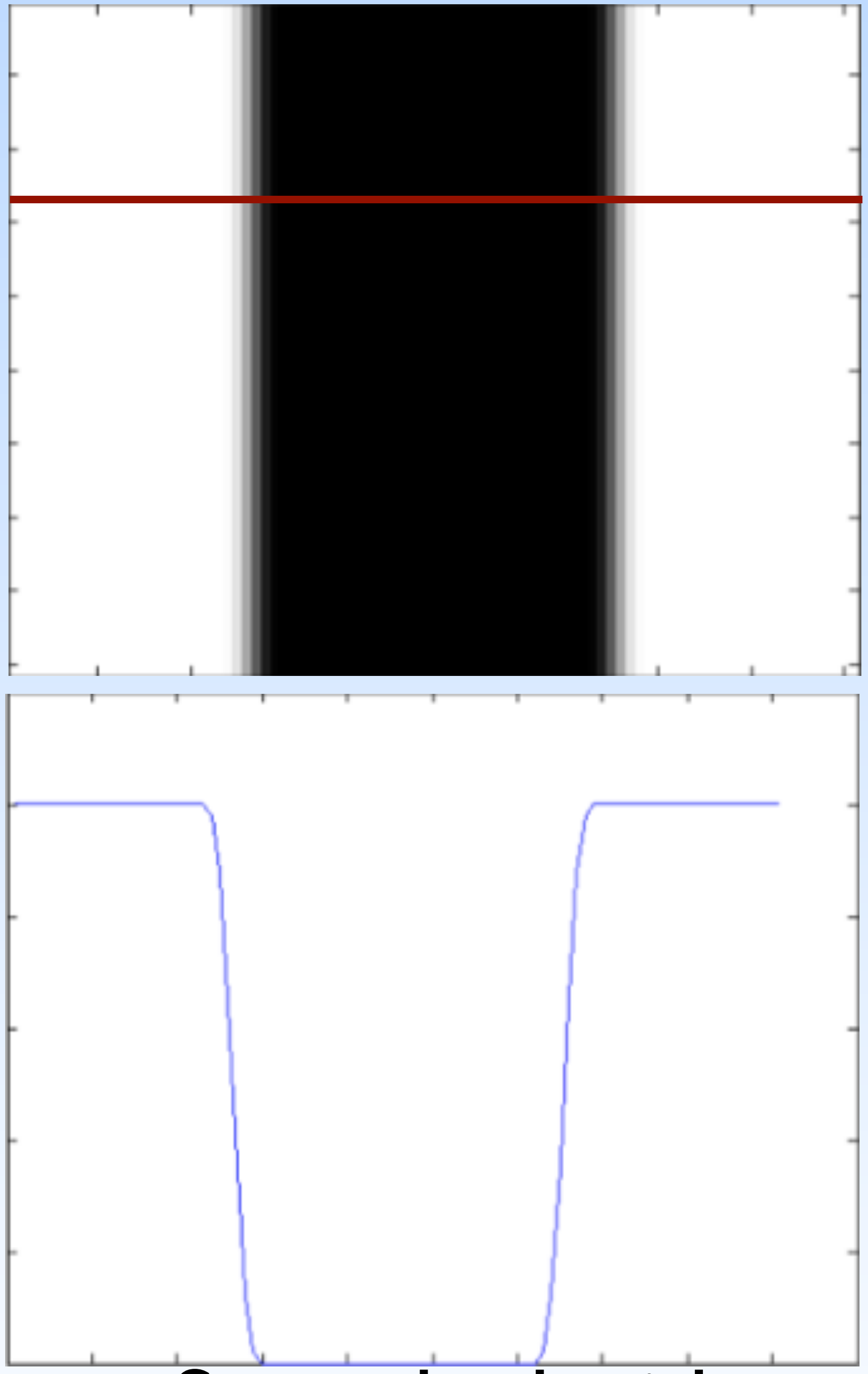
1D image line profile

1D image line profile smoothed with Gaussian

Example images

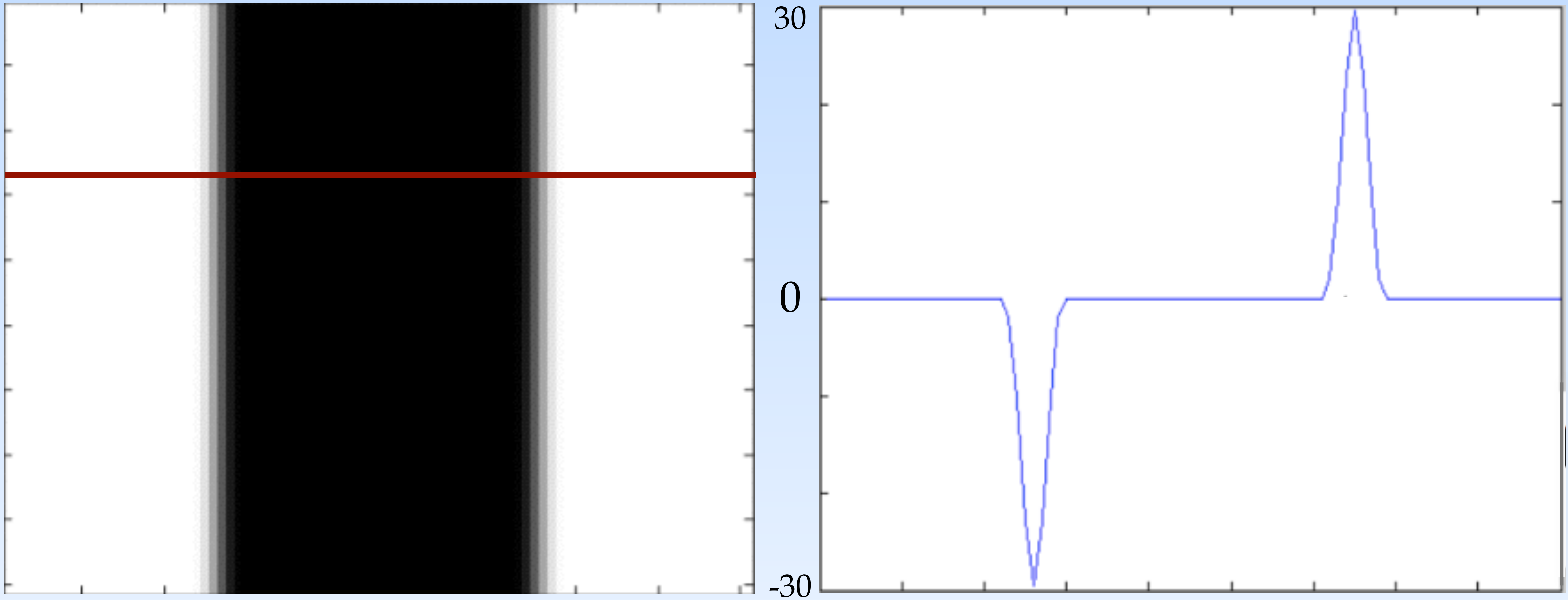


Discontinuity



Smoothed with Gaussian

Example images derivatives



Edges correspond to the fast changes in the intensities

Magnitude of the derivatives is large

Derivatives as linear filters

Image derivative:

$$\frac{\partial I(x_0)}{\partial x} = \lim_{\triangle x \to 0} \frac{I(x_0 + \triangle x) - I(x_0)}{\triangle x} \approx I(x_0 + \triangle x) - I(x_0)$$

can be implemented as linear filters:

Convolution:

$$H * I = [1 - 1] * I$$

Matrix multiplication:

$$H \cdot I = [-1 \;\; 1] \cdot [I(x_0) \;\; I(x_0 + \triangle x)]^T$$

or in its symmetric version:

Convolution $H * I = [1 \;\; 0 \;\; -1] * I$

Matrix mult. $H \cdot I = [-1 \;\; 0 \;\; 1] \cdot [I(x_0 - \triangle x) \;\; I(x_0) \;\; I(x_0 + \triangle x)]^T$

Partial image derivatives

If we consider an image I as a function of vector variables I(**x**,**y**) we write image gradient using partial derivatives as:

$$\nabla I(\mathbf{x},\mathbf{y}) = \frac{\partial I}{\partial x}\vec{i}_x + \frac{\partial I}{\partial y}\vec{i}_y$$

This can be written using gradient filters:

$$\nabla I(\mathbf{x},\mathbf{y}) = (D_x * I)\vec{i}_x + (D_y * I)\vec{i}_y$$

And we can define gradient intensity and direction:

$$|\nabla I(\mathbf{x},\mathbf{y})| = \sqrt{(D_x * I)^2 + (D_y * I)^2} \quad \psi(\nabla I) = \arctan\left(\frac{D_y * I}{D_x * I}\right)$$

Gradient filters

Basic derivative filters:

$$D_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad D_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Prewitt gradient filter

$$D_x = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

$$D_y = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & 1 \end{bmatrix}$$

Sobel gradient filter

$$D_x = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$D_y = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gradients smoothed with Gaussians

Images smoothed with Gaussians and then filtered with derivatives filters:

$$I_x = D_x * (G * I) \qquad\qquad I_y = D_y * (G * I)$$

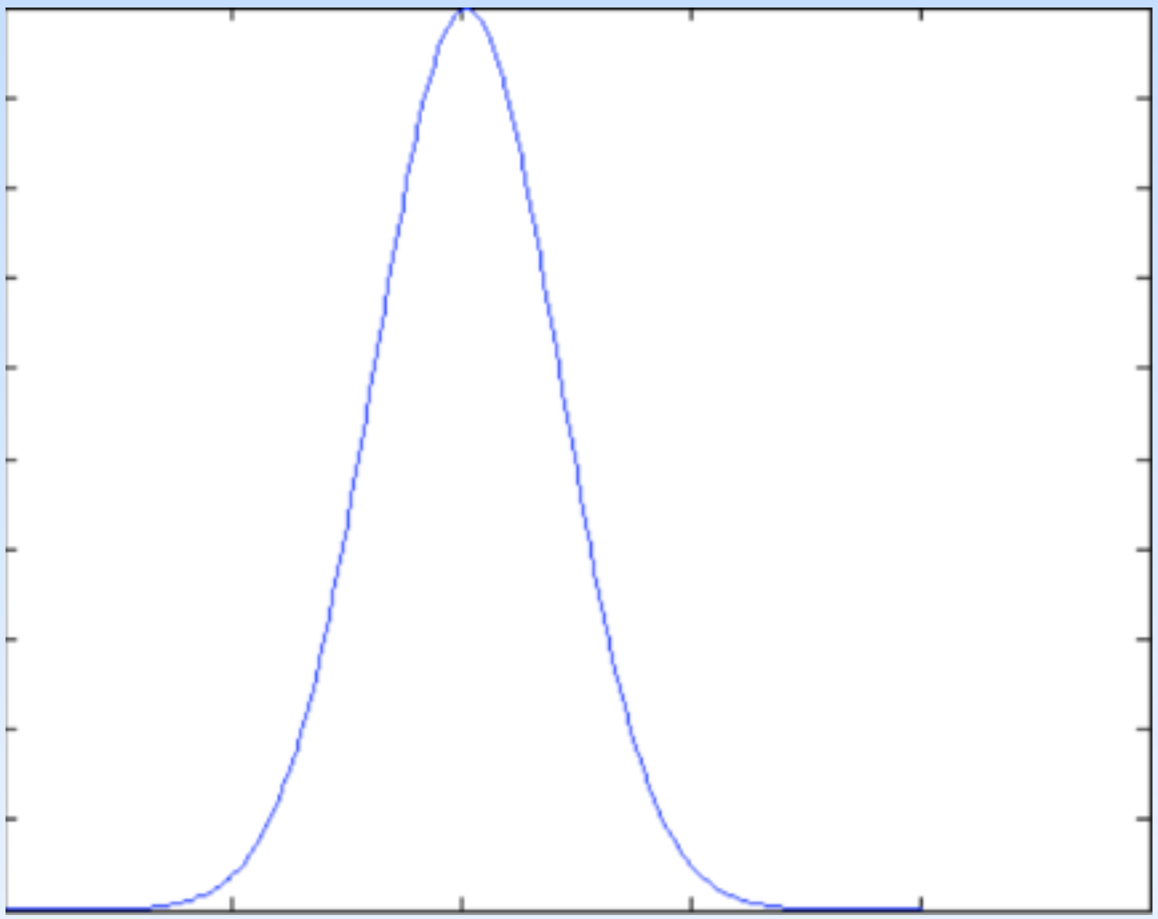
Because of the associativity:

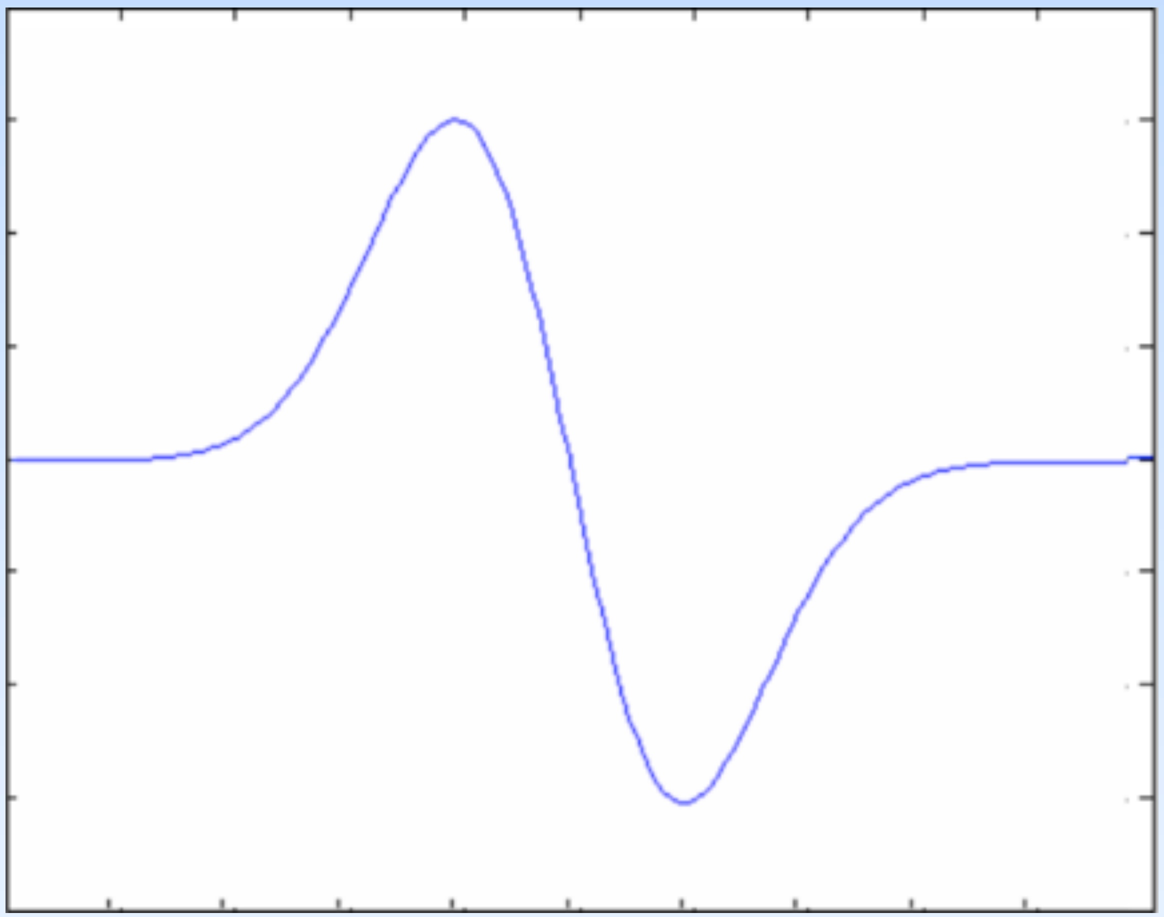
Cannot be done with correlation

$$I_x = (D_x * G) * I = G_x * I \qquad I_y = (D_y * G) * I = G_y * I$$

images can be directly convolved by the derivatives of the Gaussian filters.

Derivatives of Gaussians in 1D

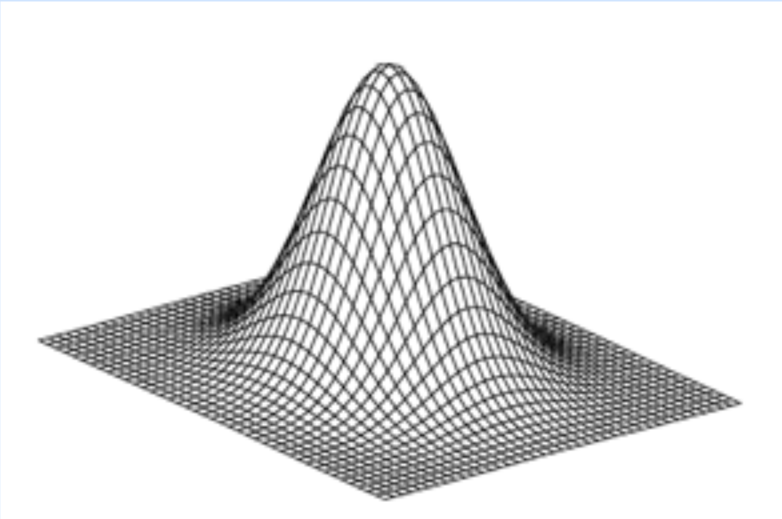




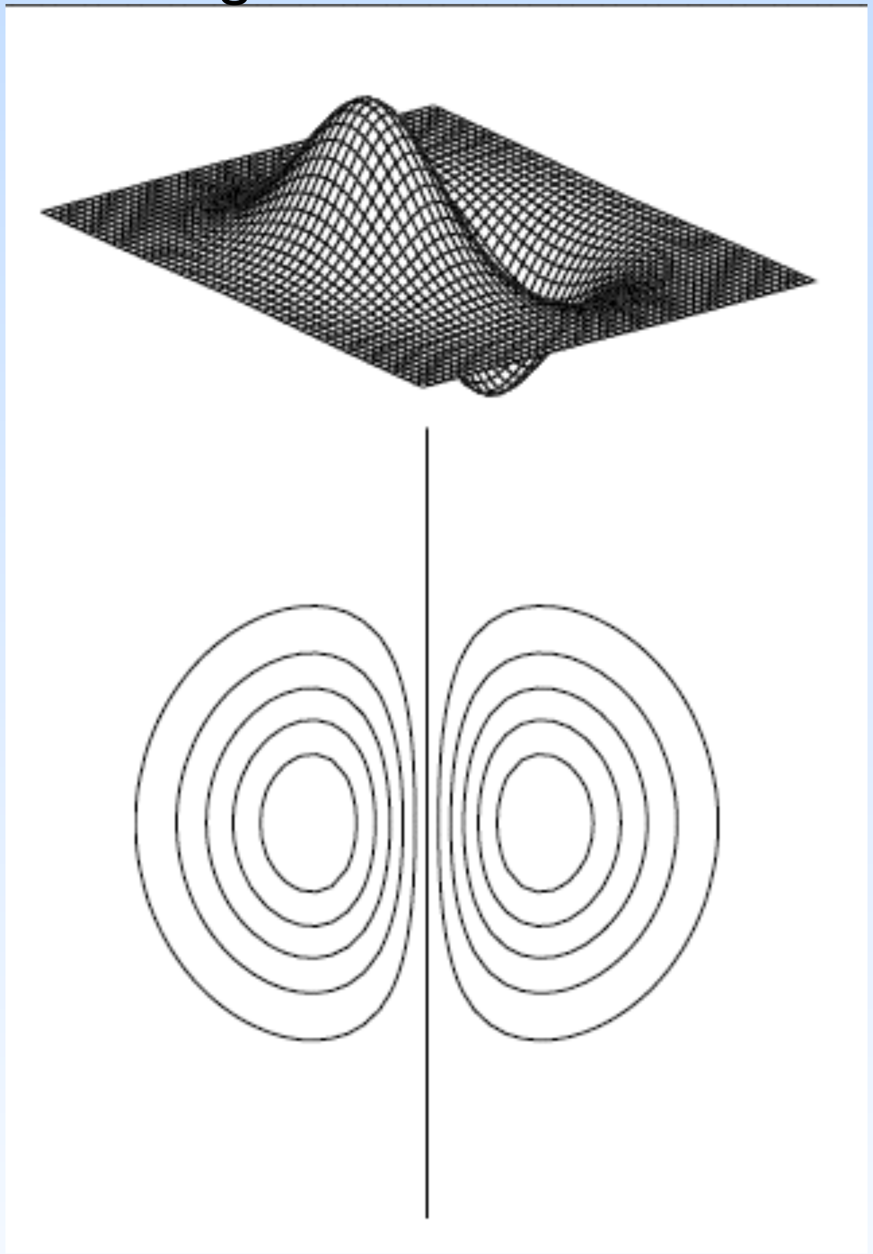
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}} \qquad G_x = \frac{\partial G}{\partial x} = -\frac{x}{\sigma^3\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

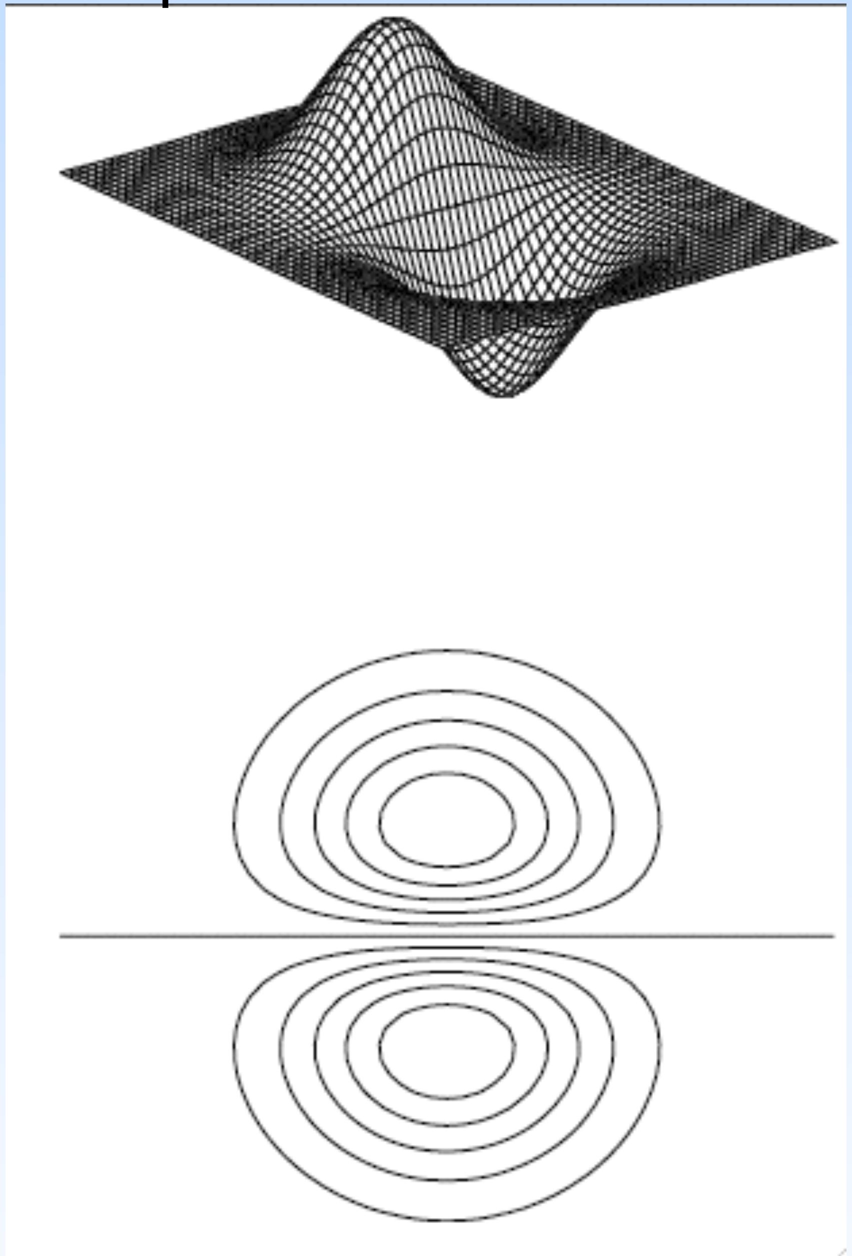
Derivatives of Gaussians in 2D

Images are from Carlo Tomasi Computer Vision lecture notes



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





$$G_x = \frac{\partial G}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \qquad G_y = \frac{\partial G}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

Practical Aspects and normalization

Derivatives without normalization and due to separability:

$$G_x(x,y) = -\frac{x}{\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} = G_x(x)G(y)$$

$$G_y(x,y) = -\frac{y}{\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} = G_y(y)G(x)$$

in discreet case:

$$I_x(x,y) = \sum_{i=-n}^{n}\sum_{j=-n}^{n} I(i,j)G_x(x-i,y-j) =$$

$$= \sum_{i=-n}^{n} G_x(x-i) \sum_{j=-n}^{n} I(i,j)G(y-j) = \sum_{j=-n}^{n} G(y-j) \sum_{i=-n}^{n} I(i,j)G_x(x-i)$$

$$I_y(x,y) = \sum_{i=-n}^{n}\sum_{j=-n}^{n} I(i,j)G_y(x-i,y-j) =$$

$$= \sum_{i=-n}^{n} G(x-i) \sum_{j=-n}^{n} I(i,j)G_y(y-j) = \sum_{j=-n}^{n} G_y(y-j) \sum_{i=-n}^{n} I(i,j)G(x-i)$$

With normalization

$$I_x(x,y) = \sum_{i=-n}^{n} \overline{G_x}(x-i) \sum_{j=-n}^{n} I(i,j)\overline{G}(y-j)$$

$$I_y(x,y) = \sum_{i=-n}^{n} \overline{G}(x-i) \sum_{j=-n}^{n} I(i,j)\overline{G_y}(y-j)$$

Normalize by the sum of the filter samples:

$$\overline{G_u}(u) = k_g * G_u(u), \quad where \quad G_u(u) = -ue^{-\frac{1}{2}\left(\frac{u}{\sigma}\right)^2} \quad and \quad k_d = \frac{1}{\sum\limits_{u=-n}^{n} uG_u(u)}$$

$$\overline{G}(u) = k * G(u), \quad where \quad G(u) = e^{-\frac{1}{2}\left(\frac{u}{\sigma}\right)^2} \quad and \quad k = \frac{1}{\sum\limits_{u=-n}^{n} G_u(u)}$$

Image directional gradients

Gradient in x direction Gx

Gradient in y direction Gy

Gradient magnitude

$$|\nabla I(\mathbf{x},\mathbf{y})| = \sqrt{(G_x * I)^2 + (G_y * I)^2}$$

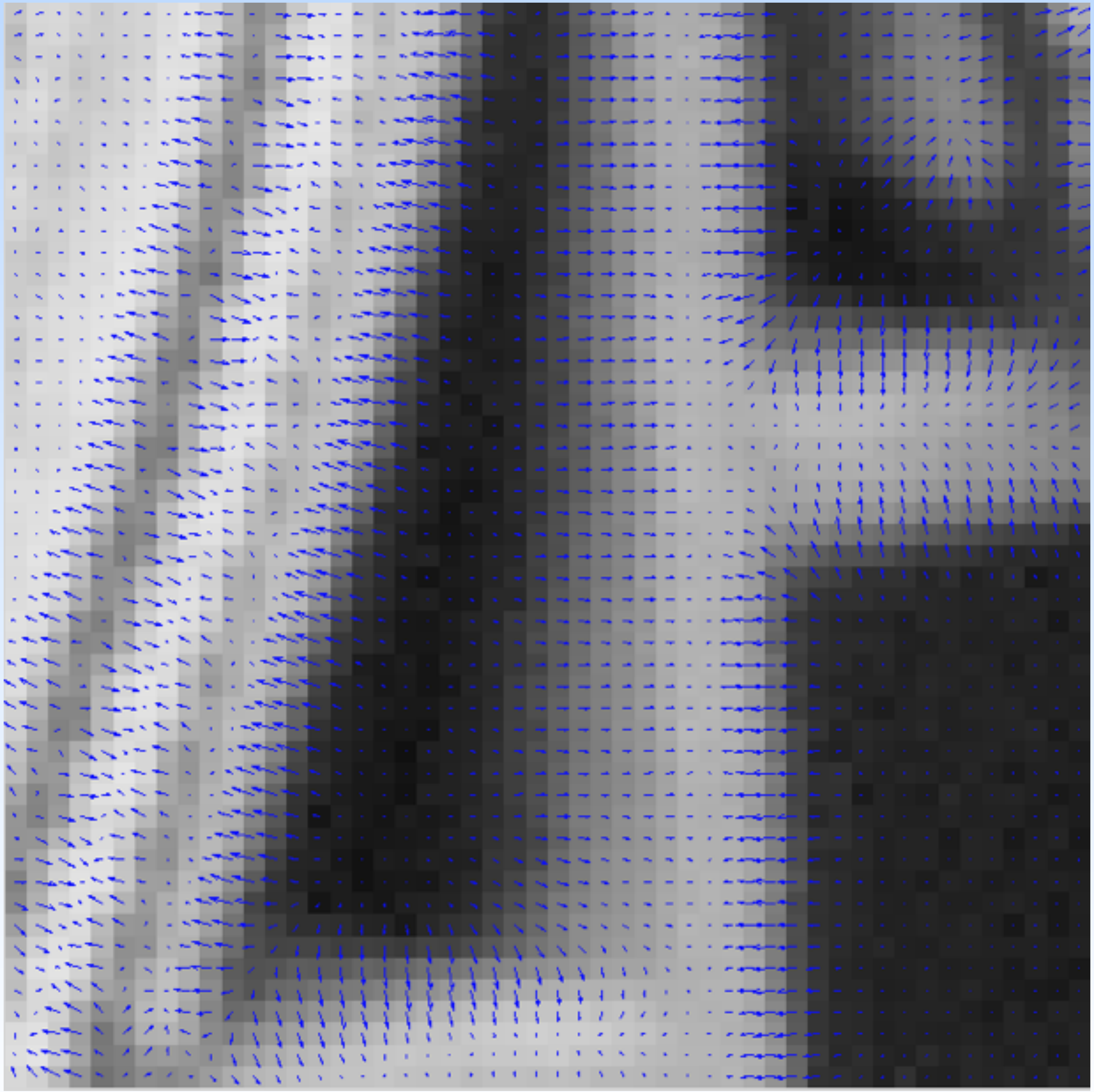
$\sigma = 0.5$

$\sigma = 1.0$

$\sigma = 1.5$

$\sigma = 2.0$

Gradient orientation



The upper left corner of the image of 50x50 pixels large.

Gradient orientation computed as:

$$\psi(\nabla I) = \arctan\left(\frac{G_x * I}{G_y * I}\right)$$

Second order derivatives

Mathematically we have:

$$\frac{\partial^2 I(x_0)}{\partial x^2} = \lim_{\triangle x \to 0} \frac{I^{'}(x_0 + \triangle x) - I^{'}(x_0)}{\triangle x} \approx I^{'}(x_0 + \triangle x) - I^{'}(x_0)$$

Further simplified:

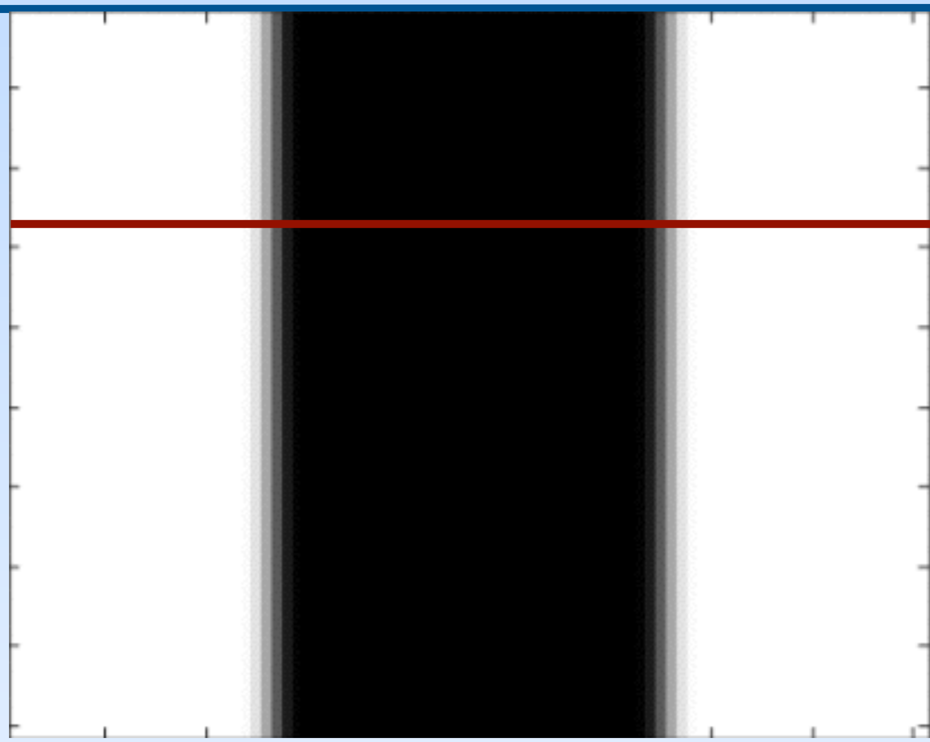
$$\frac{\partial^2 I(x_0)}{\partial x^2} = \nabla^2 I = I(x_0 + 2\triangle x) - 2I(x_0 + \triangle x) + I(x_0)$$

In 2D:

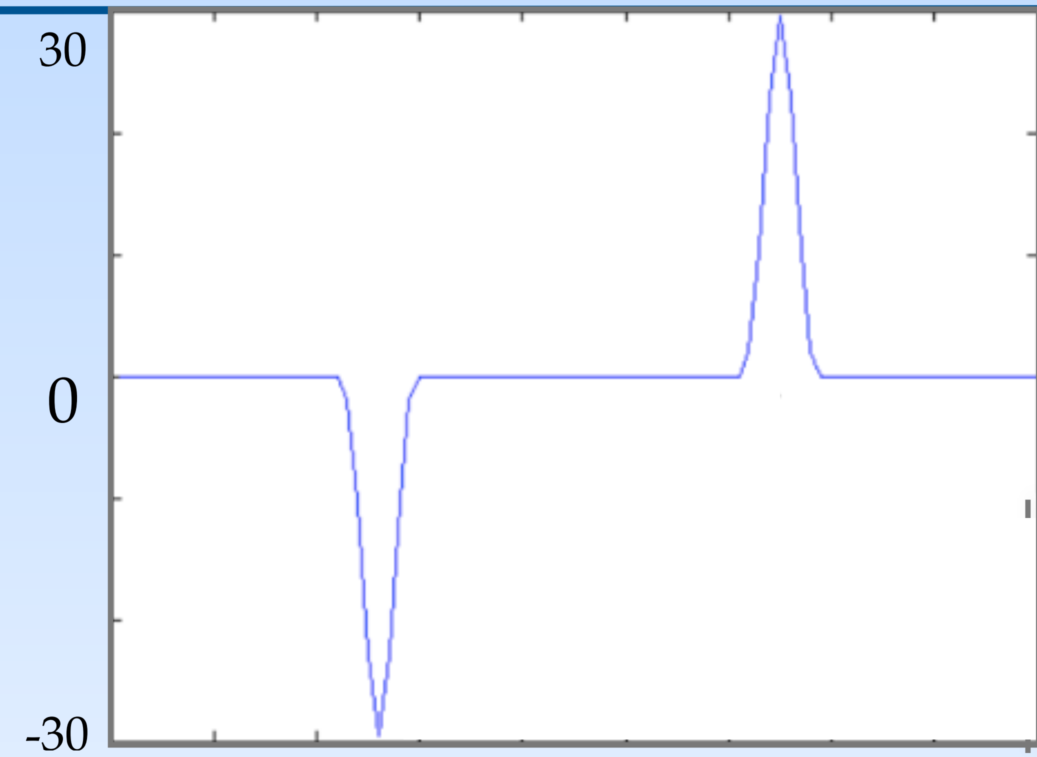
$$\frac{\partial^2 I(x,y)}{\partial x^2} = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Example images derivatives

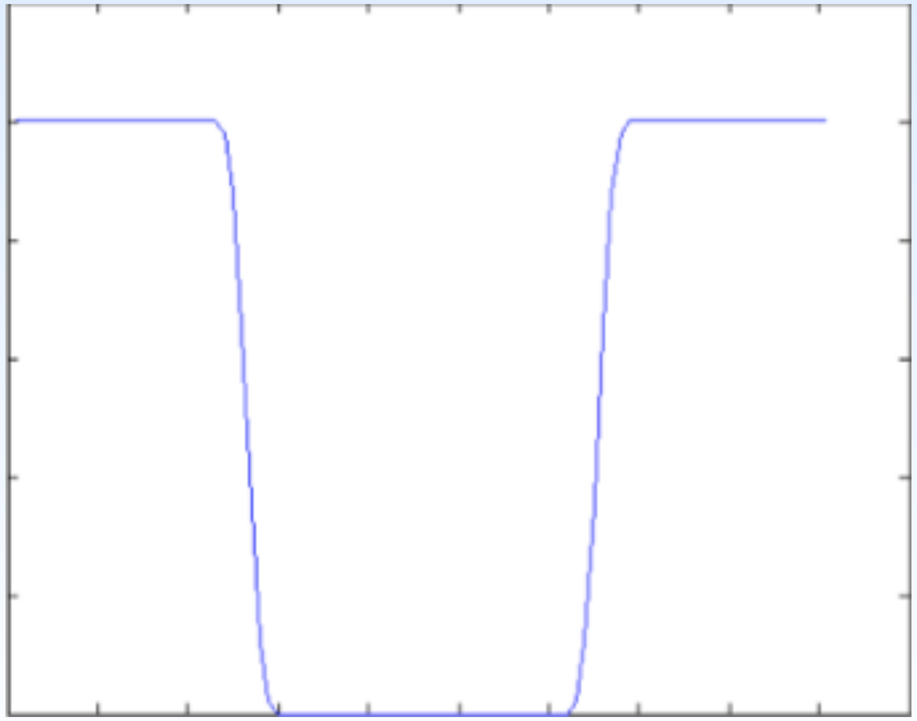
Original image



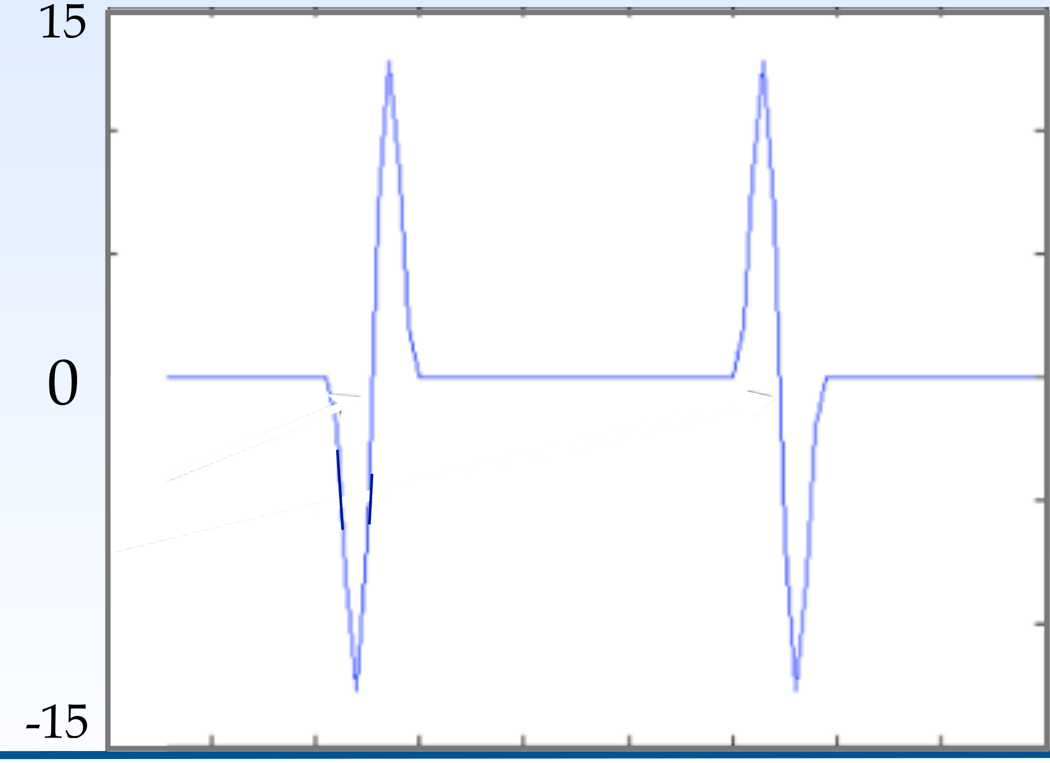
first order derivative



Smoothed with Gaussian



second order derivative



Second order derivative filters

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Second order derivative or Laplacian:

$$\frac{\partial^2 I(x_0)}{\partial x^2} = I(x_0 + 2\triangle x) - 2I(x_0 + \triangle x) + I(x_0)$$

can be implemented as simple linear filter:

$$H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{matrix} \nabla^2 I = H * I \Rightarrow \\ H \cdot [I(x_0 + 2\triangle x) \;\; I(x_0 + \triangle x) \;\; I(x_0)]^T \end{matrix}$$

or by separately in x and y direction:

$$H_{xx} = H_{yy}^T = [1 \;\; -2 \;\; 1] \qquad \nabla^2 I = H_{xx} * I + H_{yy}^T * I$$

Edge detection

To compute image edges we need:

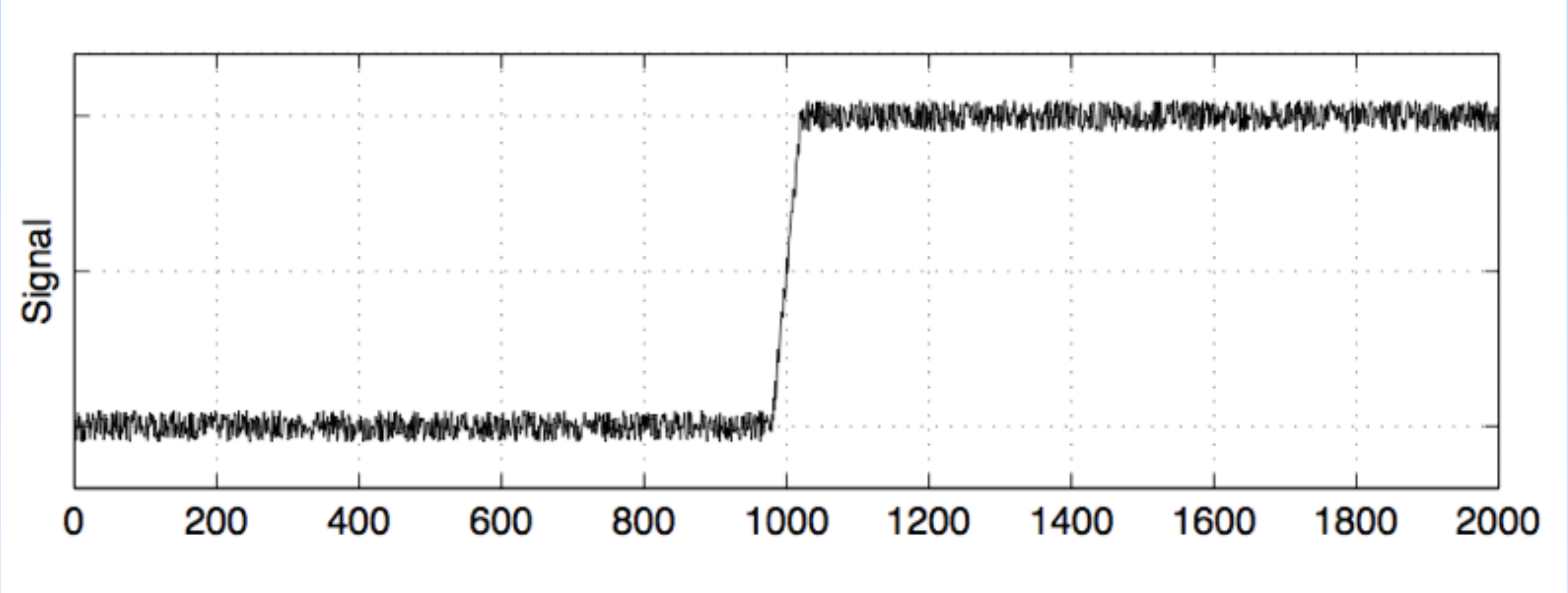
Image smoothing

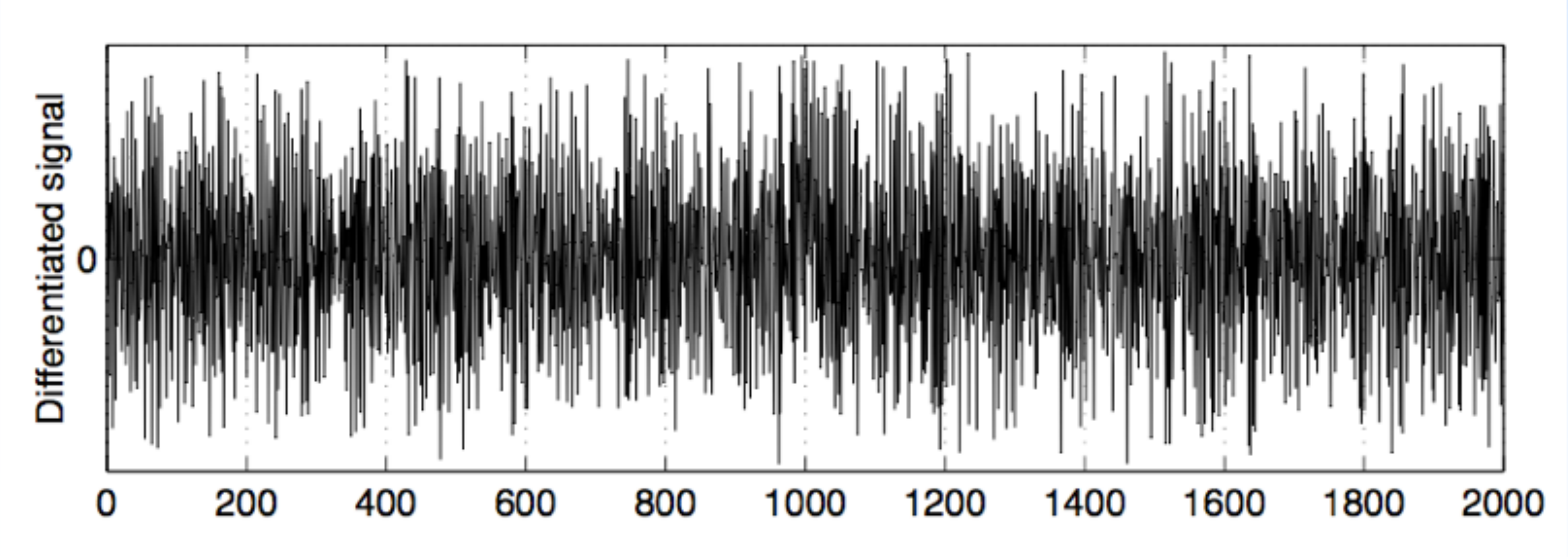
Image derivatives

Non-maximal suppression

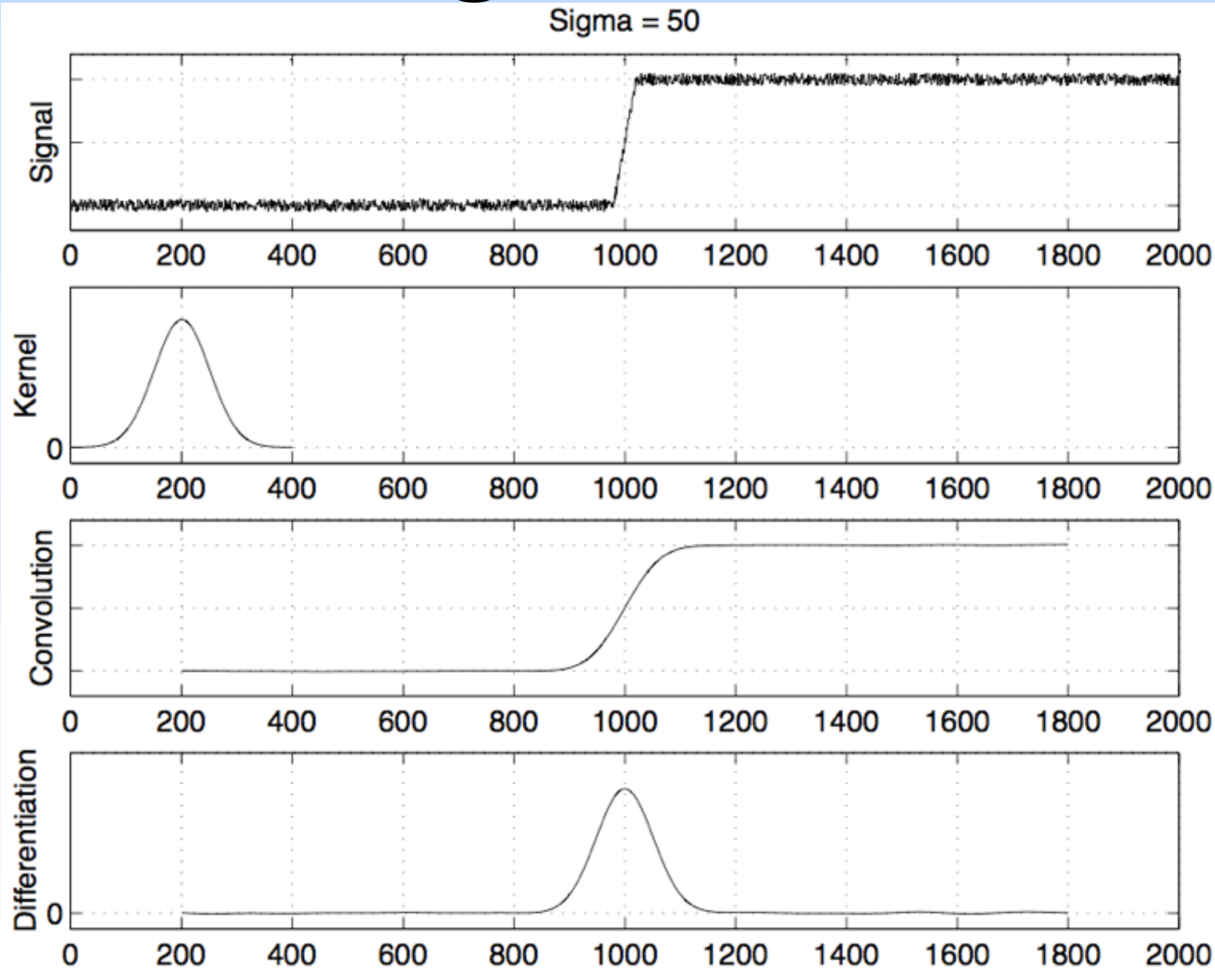
Thresholding

1D edge detection

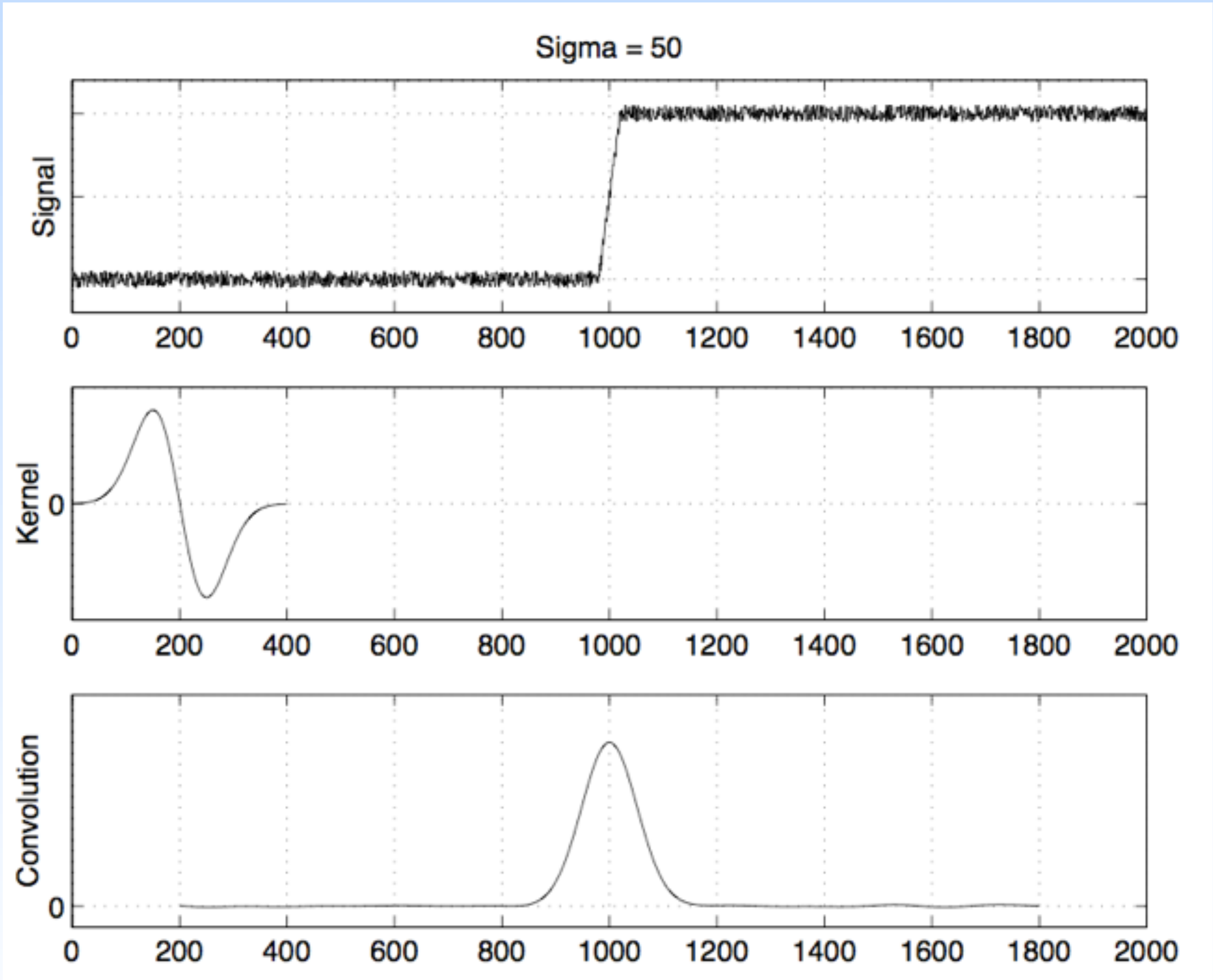




1D edge detection

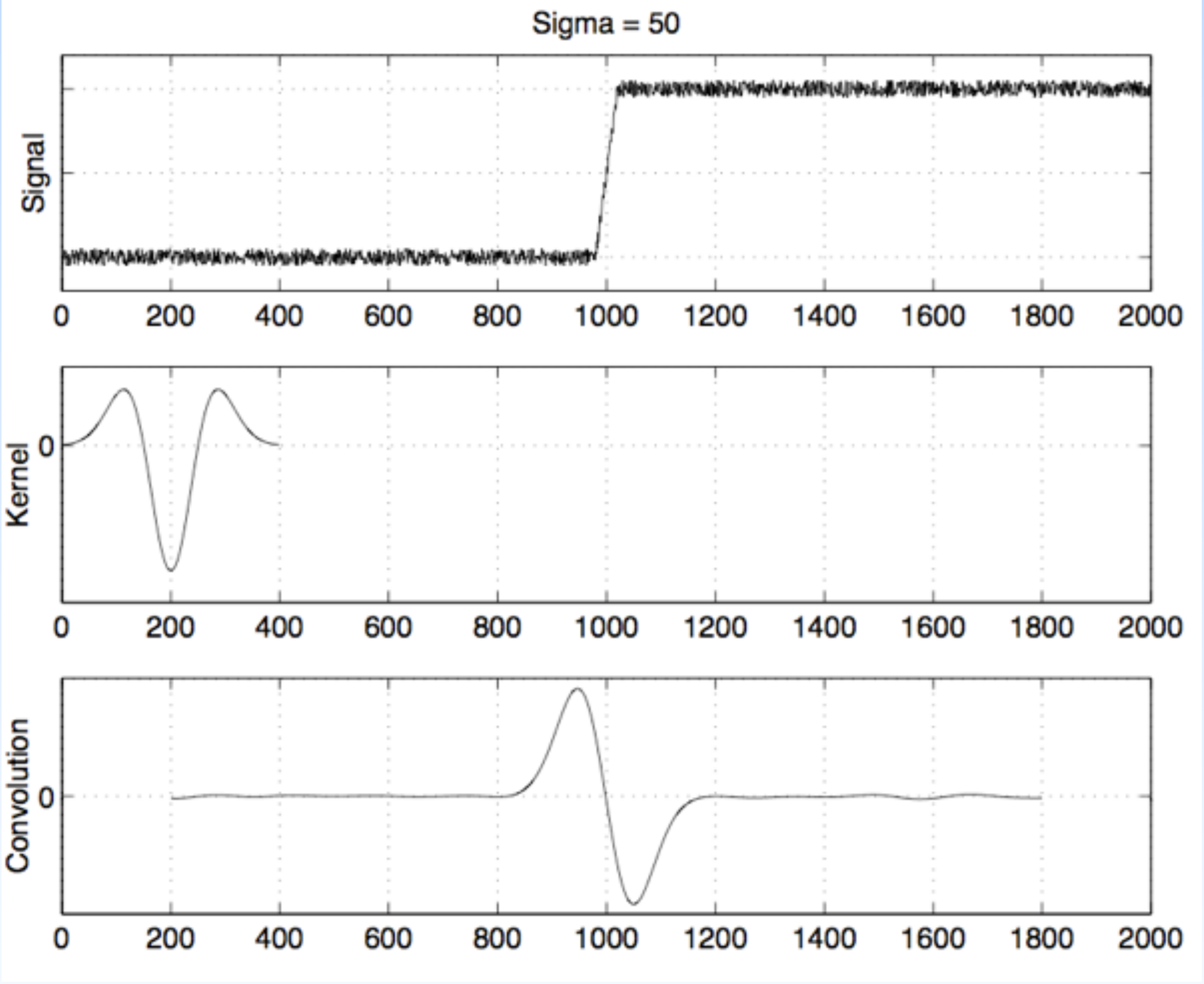


Convolving with the first order derivative of Gaussian



Smoothing removes noise and amplifies signal at the edge discontinuities.

Zero crossings and convolution with Laplacian

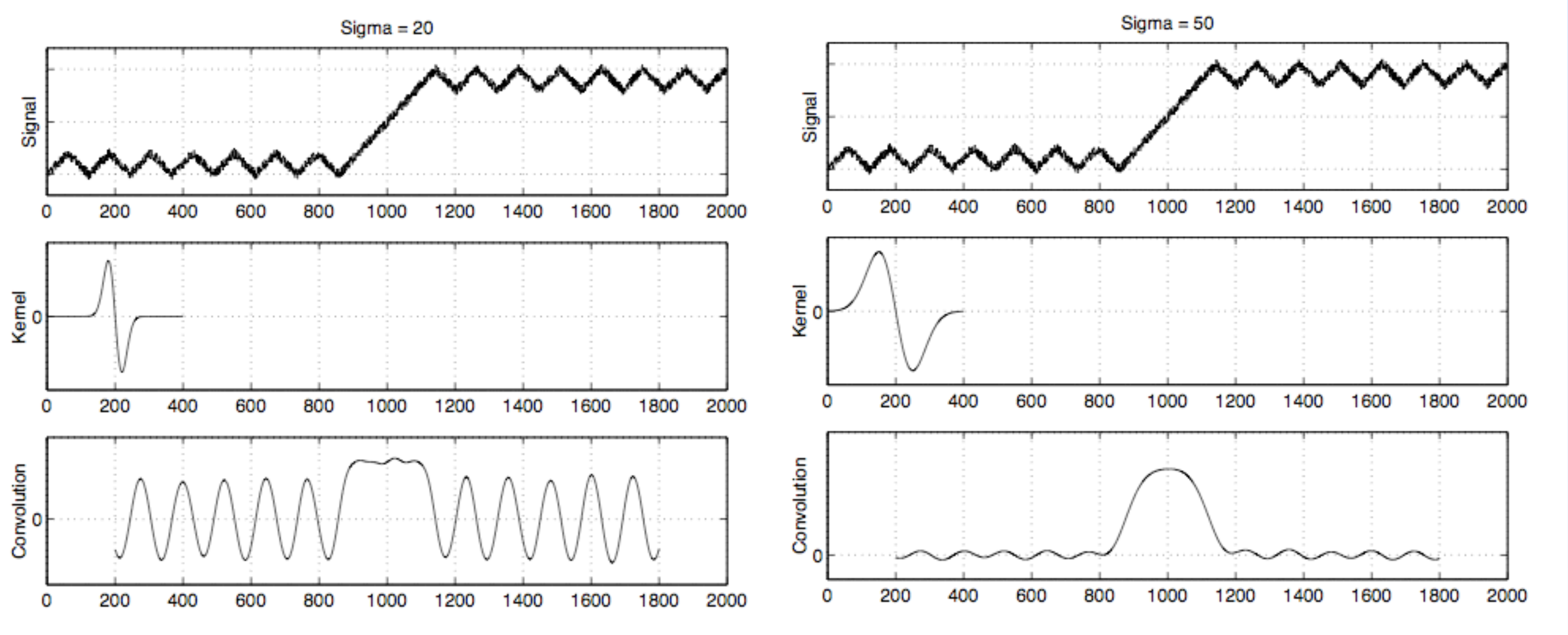


Looking at minima or maxima of fist order image derivatives is the same as looking for zero-crossings of second order image derivatives.

Multi-scale edge detection

The amount of smoothing controls the scale at which we analyze the image.

Small smoothing brings edges at a fine scale



Signal noise is not suppressed

Increased smoothing suppresses noise

2D edge detection

Original image

Gradient image smoothed with Gaussian

Gradient image smoothed with Gaussian

2D edge detection

Original image

Gradient image smoothed with Gaussian

Gradient image smoothed with Gaussian $\sigma = 2.0$

2D edge detection

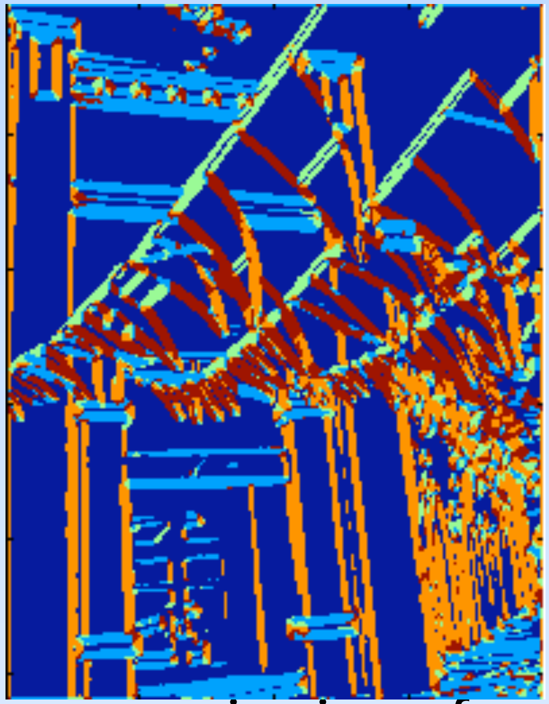
Original image

Gradient image smoothed with Gaussian $\sigma = 0.5$

Gradient image smoothed with Gaussian $\sigma = 2.0$

Non-maximal suppression

original



quantization of gradient orientations

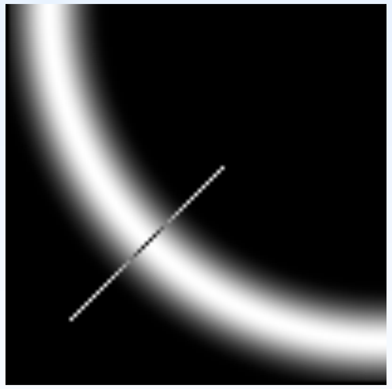
gradient magnitude

gradients after non-maximum suppression

Non-maxima suppression does thinning of the edges.

The gradient orientations are quantized into four bins

Pick neighboring pixels of the edge pixel in the direction of the gradient and take the one with the maximum gradient magnitude, while the others are set to zero.



Canny edge detection

Canny edge detection composes of all already mentioned steps which are:

- Image smoothing
- Image derivatives
- Non-maximal suppression
- Hysteresis thresholding

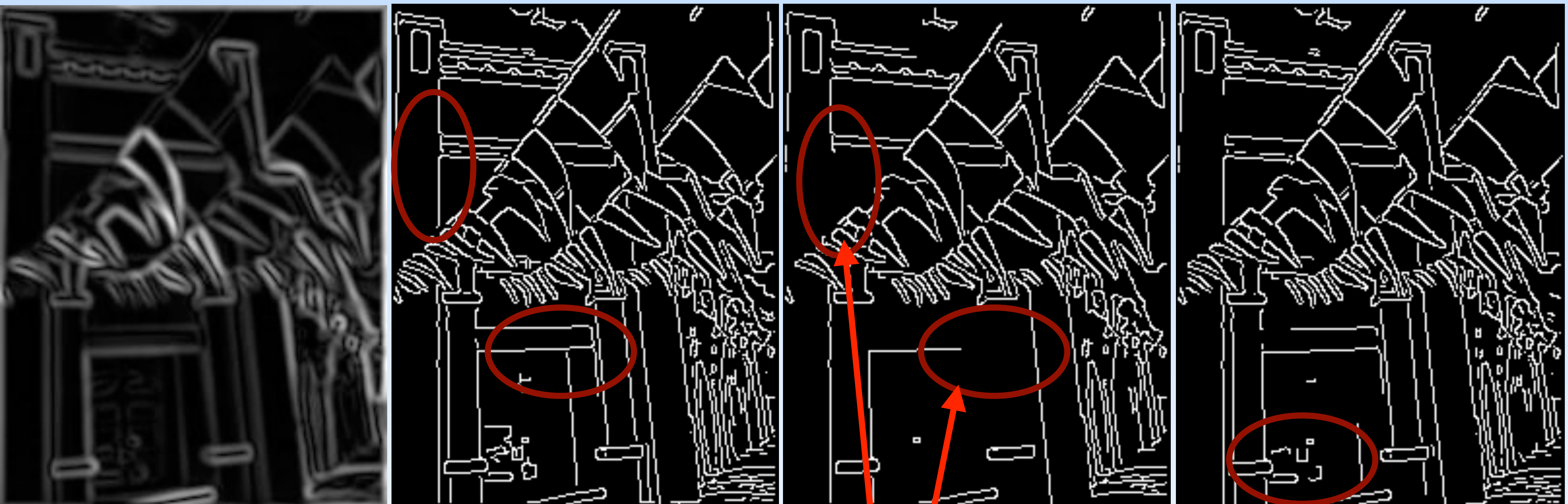
Hysteresis thresholding

Apply two thresholds Thigh and Tlow to follow edges

Algorithm:

- Search for the pixel with the gradient magnitude value in the non-maximum suppressed image is higher then Thigh
- Recursively search its neighbors and assign them to the edge if their gradient magnitude value in the non-maximum suppressed image is higher then the Tlow.
- otherwise, stop if the gradient magnitude is bellow Tlow or the pixel is already visited and assigned to be on an edge and go to 1.

Canny examples



Gradient image | Th=50, Tl=10 | Th=100, Tl=10 | Th=50, Tl=40

Some weaker edges disappear

Some medium strong edges disappear

Non-linear filtering

Linear filters combine input pixels in a way that depends on where a pixel is in the image and not on its value

Non-linear filters take into account input pixel values before deciding how to use them in the output.

Linear vs. non liner filters

original image

salt&pepper noise added

average filter

median filter

Median Filtering

$$\begin{bmatrix} 10 & 20 & 10 \\ 25 & 10 & 75 \\ 90 & 85 & 100 \end{bmatrix} \Rightarrow$$

Sort: (10 10 10 20 25 75 85 90 100)

Example

Original signal:	100 100 100 100 10 10 10 10 10
Noisy signal:	100 103 100 100 10 9 10 11 10
Filter by [1 1 1]/3:	101 101 70 40 10 10 10
Filter by 1x3 median filter:	100 100 100 10 10 10 10

Median Filter

- Median filters are nonlinear
- Median filtering reduces noise without blurring edges and other sharp details
- Median filtering is particularly effective when the noise pattern consists of strong, spike-like components. (Salt-and-pepper noise.)

Bilateral filter

It is nonlinear filtering technique based on:

Domain(smoothing) filter - It is based on *closeness* function which accounts for spacial distance between the central pixel $\mathbf{x}$ and its neighbors ξ

$$c(\xi, \mathbf{x}) = e^{-\frac{1}{2}\left(\frac{d(\xi,\mathbf{x})}{\sigma_d}\right)^2}, \quad d(\xi, \mathbf{x}) = \|\xi - \mathbf{x}\|$$

Range filter - It is based on the *similarity* function between image intensities between the central pixel $I(\mathbf{x})$ and its neighbors $I(\xi)$

$$s(\xi, \mathbf{x}) = e^{-\frac{1}{2}\left(\frac{\delta(I(\xi),I(\mathbf{x}))}{\sigma_r}\right)^2}, \quad \delta(I(\xi), I(\mathbf{x})) = \|I(\xi) - I(\mathbf{x})\|$$

σ_d - desired amount of spacial smoothing σ_r - desired amount of combining of pixel values

Bilateral filter

Combining similarity and closeness functions we obtain:

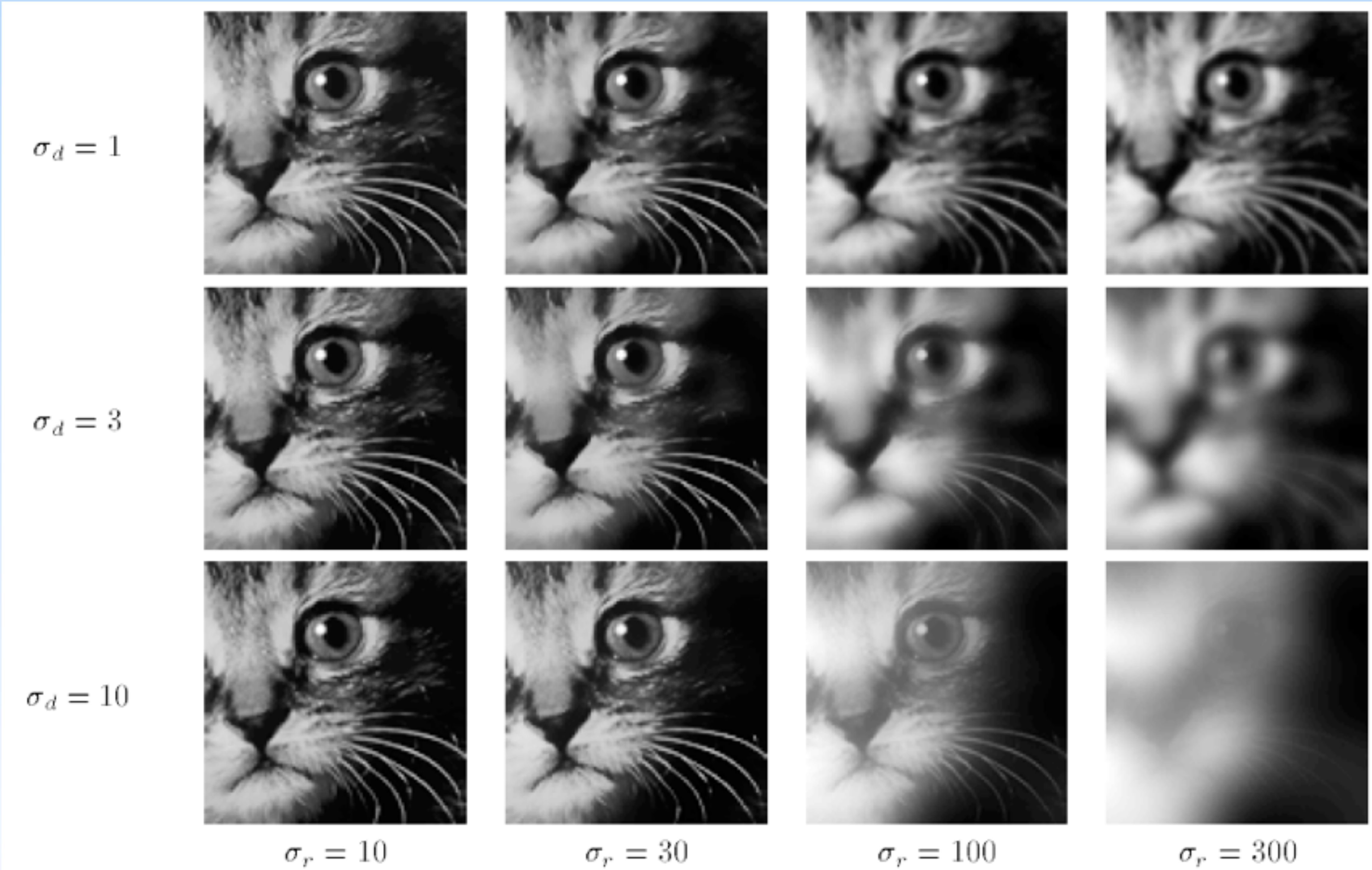
$$h(\mathbf{x}) = \frac{1}{c}\sum_{\xi \in \Omega} I(\xi)c(\xi,\mathbf{x})s((I(\xi), I(\mathbf{x}))$$

$$c = \sum_{\xi \in \Omega} c(\xi,\mathbf{x})s((I(\xi), I(\mathbf{x}))$$

This filter is know to reduce

•noise and

•to preserve edges



C. Tomasi, R. Manduchi, "Bilateral Filtering for gray and color images", Sixth International Conference on Computer Vision, pp 839-46, New Delhi, India, 1998.