Digital Image Filters

Giacomo Boracchi

November, 12th 2018

giacomo.boracchi@polimi.it

home.deib.polimi.it/boracchi/

Spatial-Domain vs Transform-Domain Methods

A survey of most important operations in image processing:

- Spatial Intensity Transformations
- Spatial Local Transformations: **convolution**
- Global Transformations

Spatial transformations (intensity or local) are direct manipulation of pixel intensities. Relevant examples of convolutional filters:

- Smoothing Filters (denoising)
- Differentiating Filters (edge detector)

Global transformations can be also used to represent the image in a different domain (e.g. Fourier, wavelet) where it is easier to extract meaningful information

Bibliography

"Digital Image Processing", 4th Edition Rafael C. Gonzalez, Richard E. Woods, Pearson 2017

Intensity Transformation

Transformations that operate
on each single pixels of an image

Intensity Transformations

In general, these can be written as

$$G(r,c) = T[I(r,c)]$$

Where

- I is the input image to be transformed
- G is the output
- T: $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ or T: $\mathbb{R}^3 \rightarrow \mathbb{R}$ is a function

T operates independently on each single pixel.

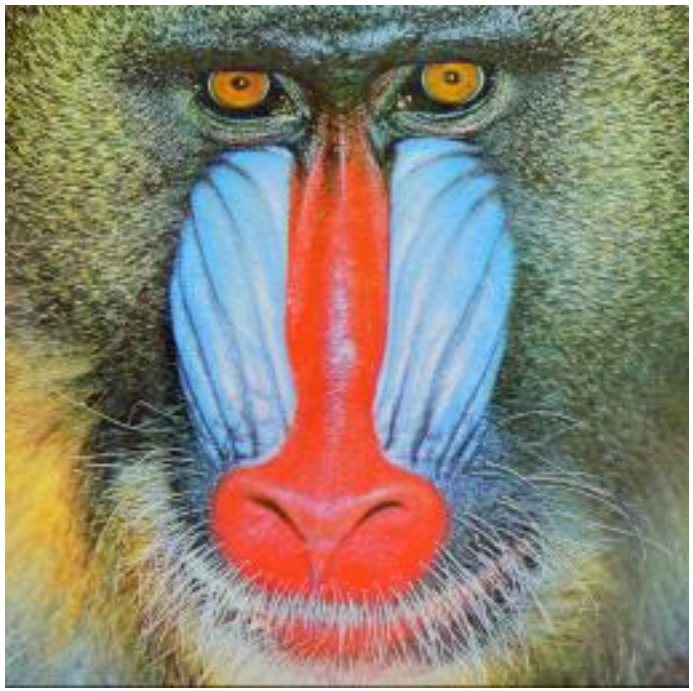
RGB → Grayscale Conversion

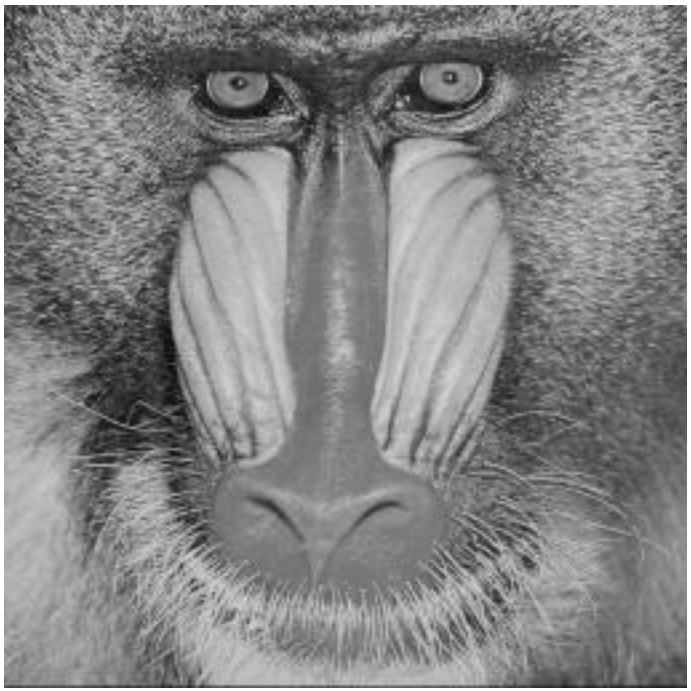
A linear transformation of pixel intensities T: $\mathbb{R}^3 \rightarrow \mathbb{R}$

$$Gray(r,c) = [0.299, 0.587, 0.114] * [R(r,c), G(r,c), B(r,c)]'$$

which corresponds to a linear combination of the 3 channels

$$Gray(r,c) = 0.299 * R(r,c) + 0.587 * G(r,c) + 0.114 * B(r,c)$$





YCbCr color space

Color space conversion T: $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ to map RBG to YcBCr

- Y is the *luma* signal, similar to grayscale
- Cb and Cr are the *chroma* components

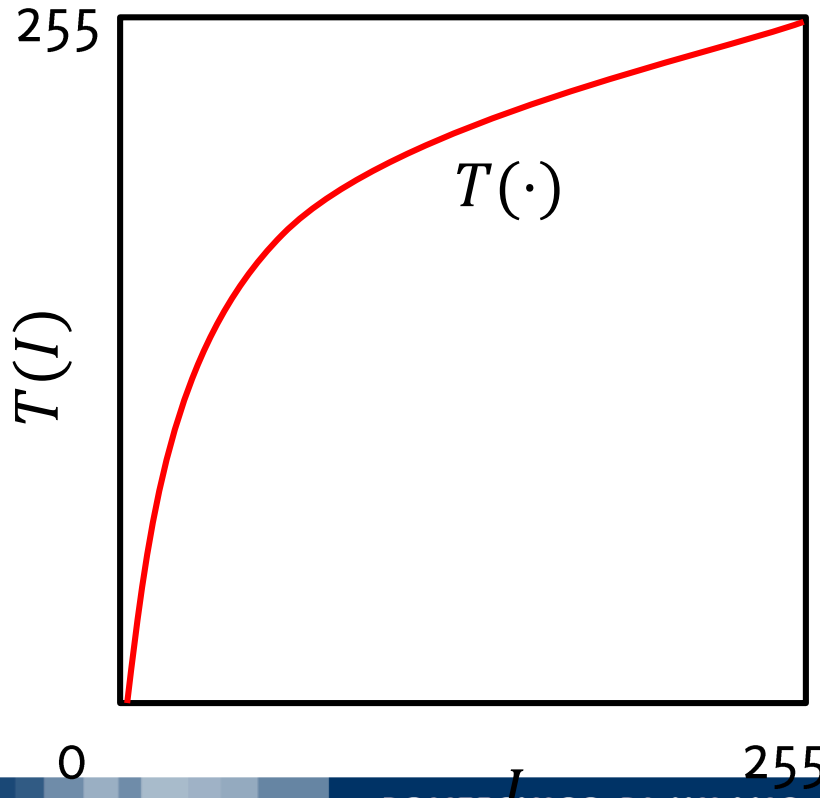
Human eye is less sensitive to color changes than luminance variations. Thus,

- Y can be stored / transmitted at high resolution
- Cb and Cr can be subsampled, compressed, or otherwise treated separately for improved system efficiency

(e.g. in JPEG compression che chromatic components are encoded at a coarser level than luminance)

Gray-level mapping

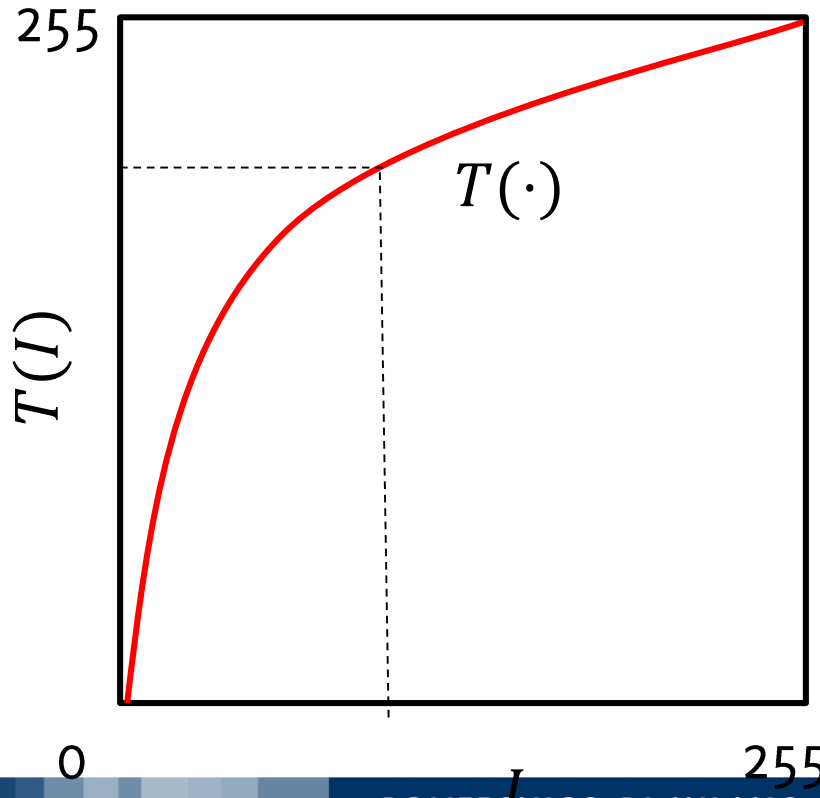
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

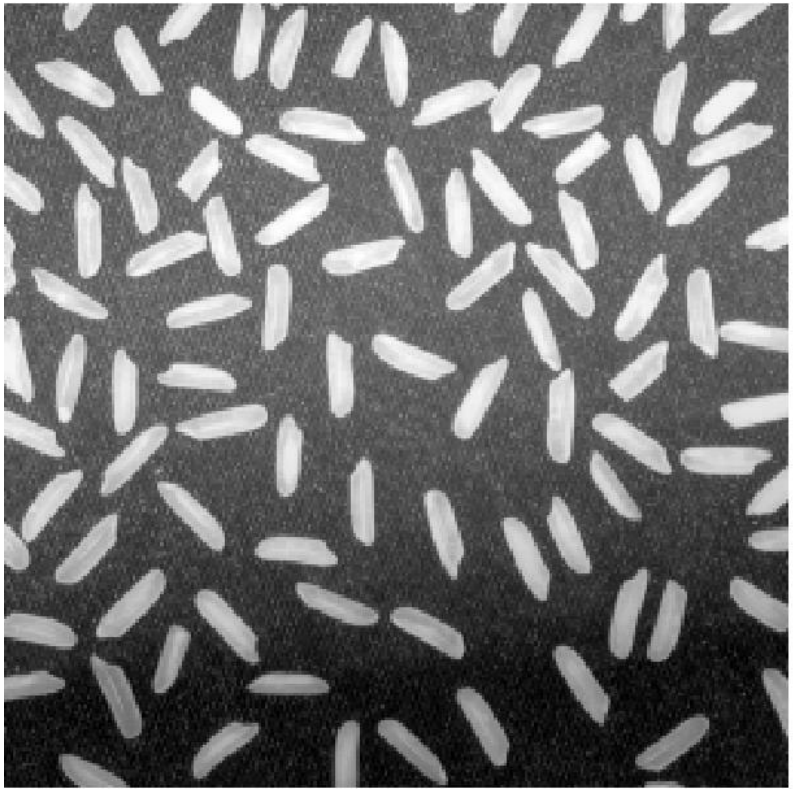


Gray-level mapping

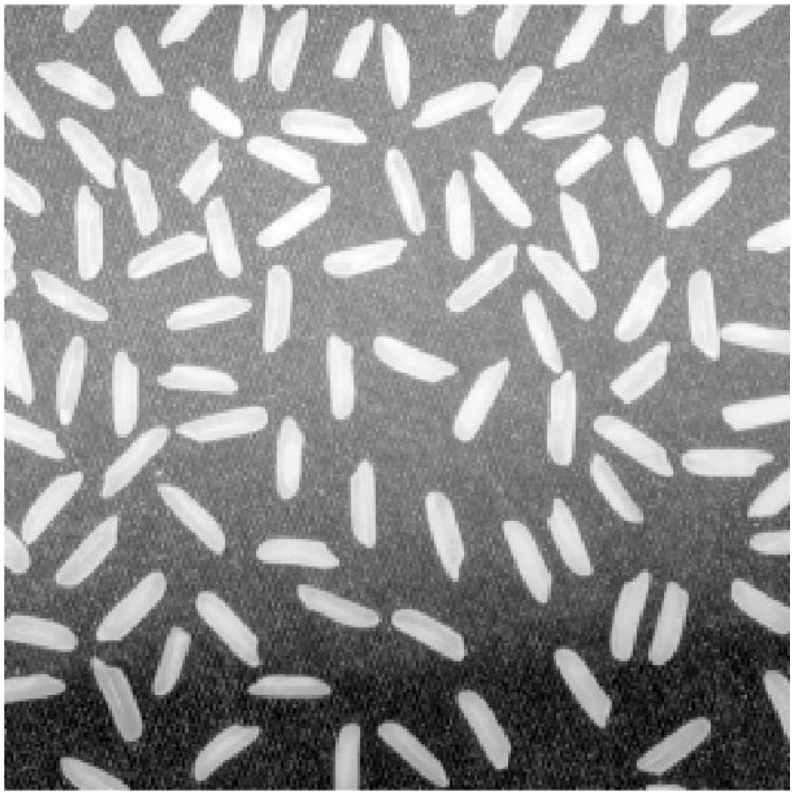
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?





Input I

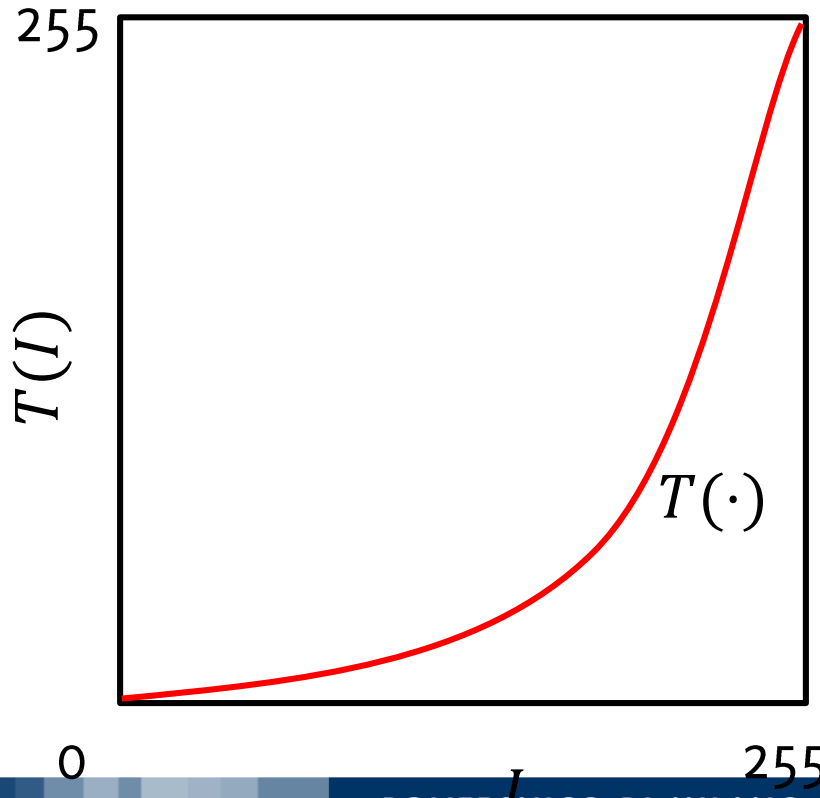


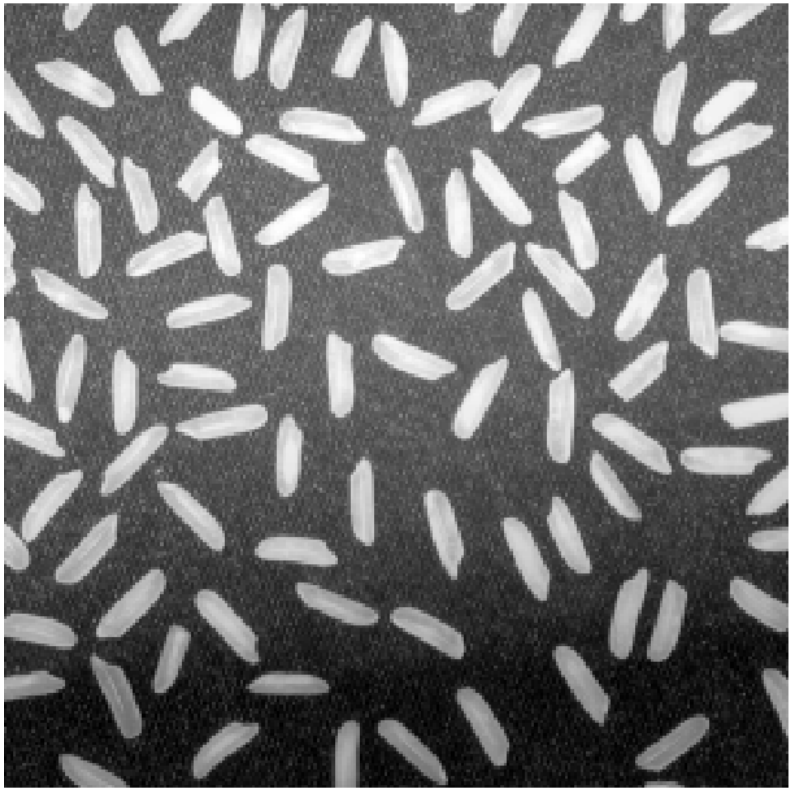
Output $G = T(I)$

Gray-level mapping

A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?





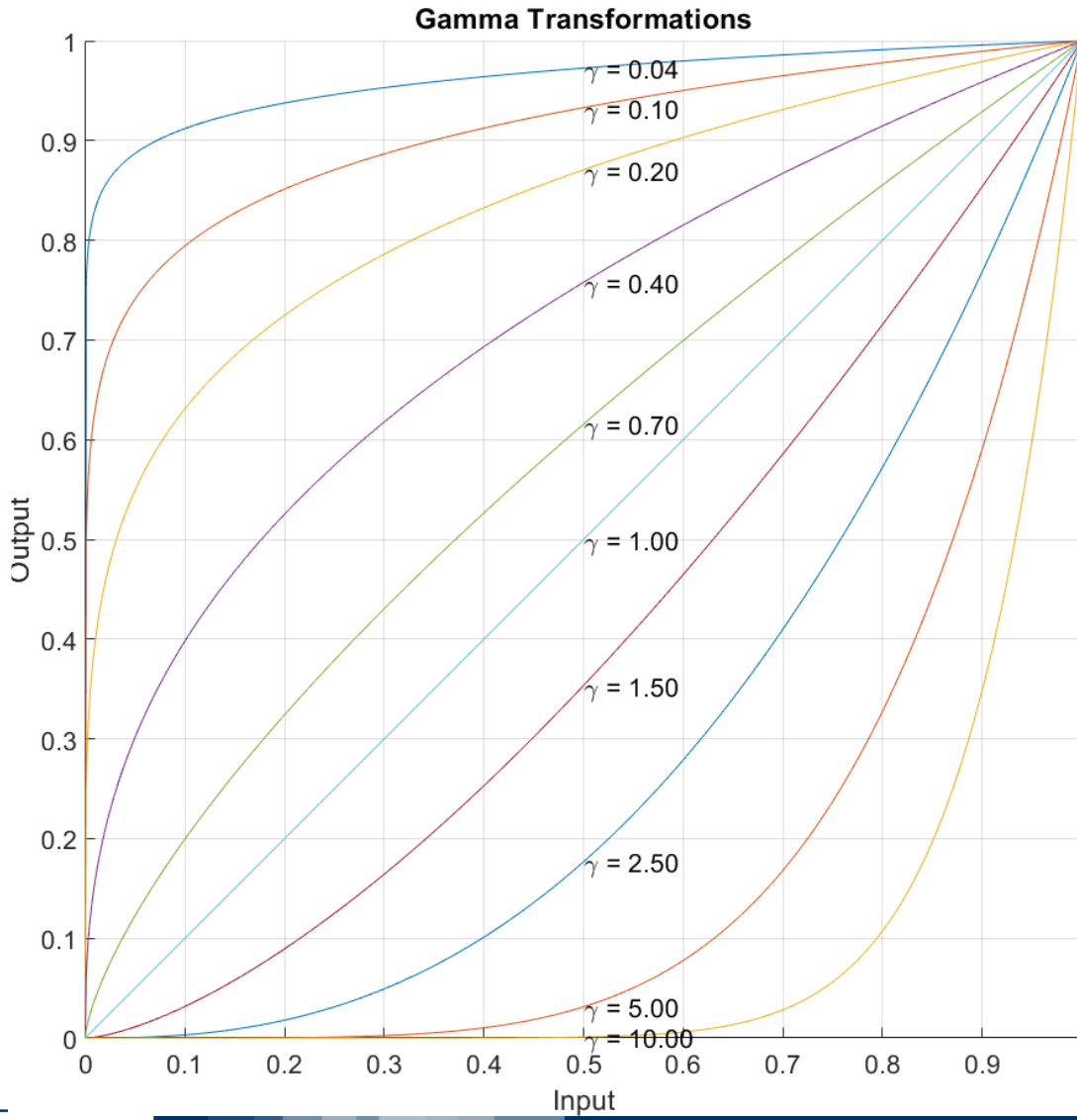
Input I

Output $G = T(I)$

Gamma Correction

Power-low transformation that can be written as

$$G(r,c) = I(r,c)^{\gamma}$$



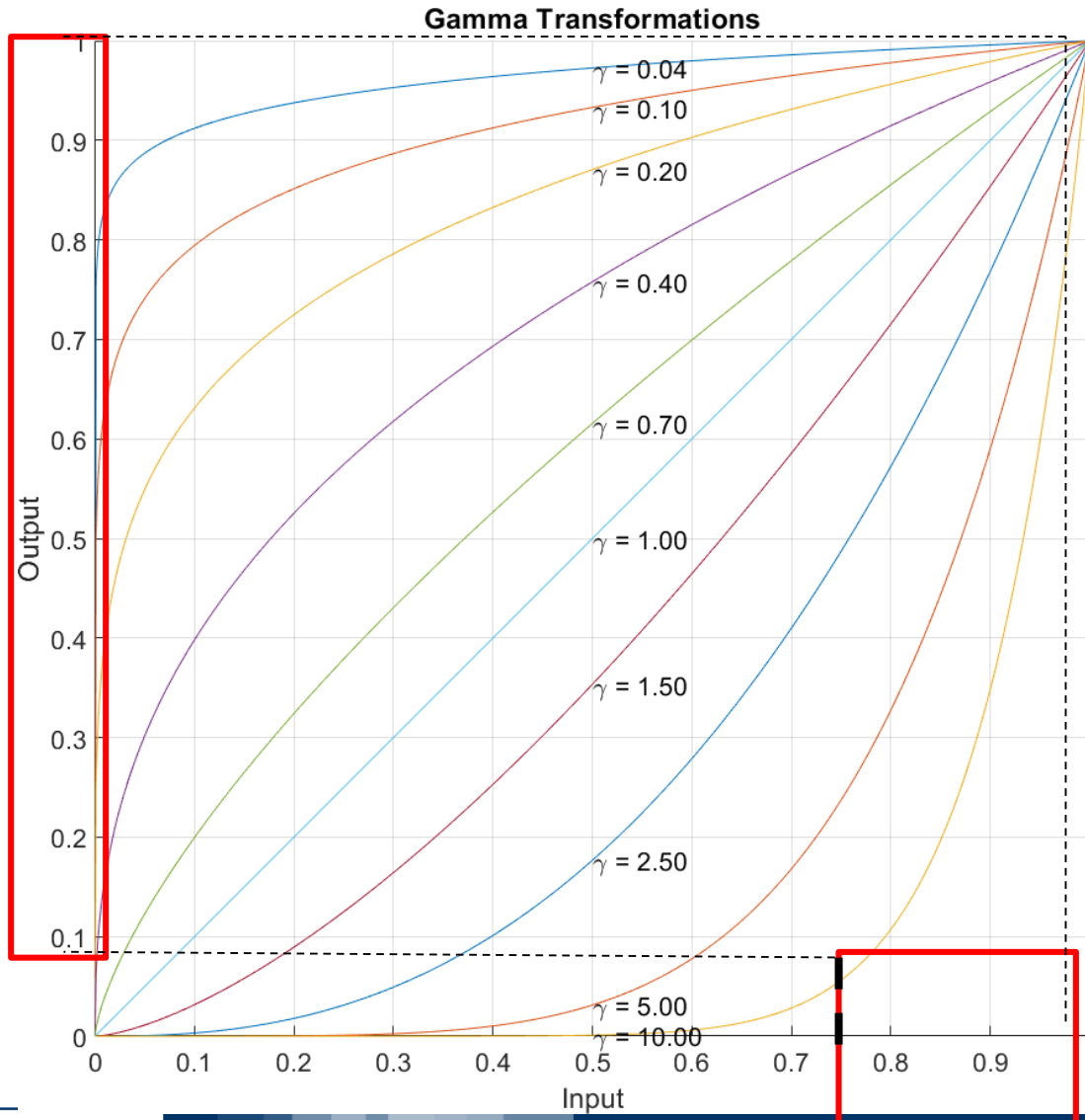
Gamma Correction

Power-low transformation that can be written as

$$G(r,c) = I(r,c)^{\gamma}$$

Contrast Enhancement:

- Low values of γ stretch the intensity range at high-values



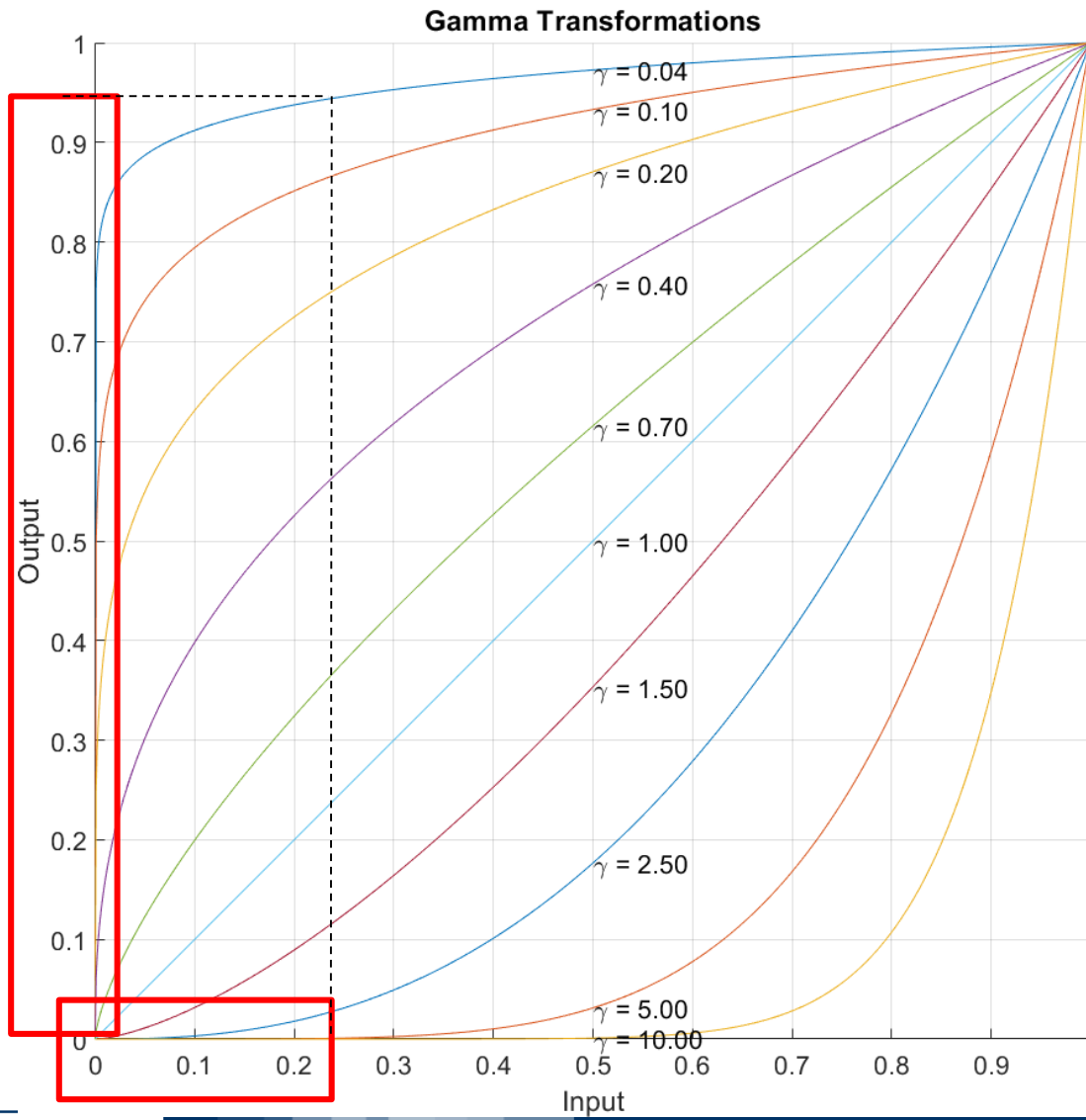
Gamma Correction

Power-low transformation that can be written as

$$G(r,c) = I(r,c)^{\gamma}$$

Contrast Enhancement:

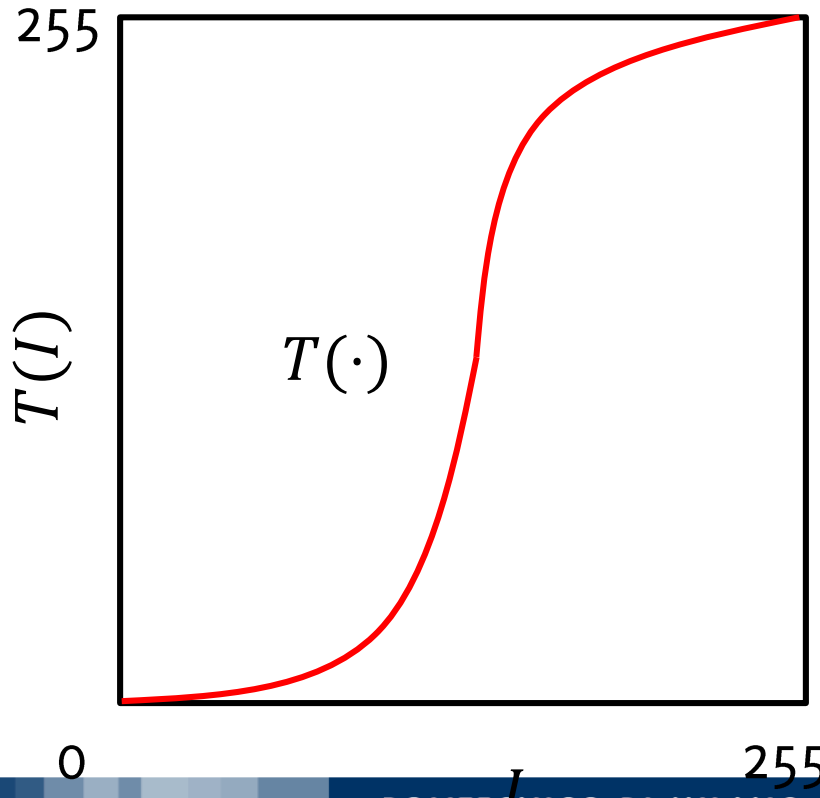
- Low values of γ stretch the intensity range at high-values
- High values of γ stretch the intensity range at low values



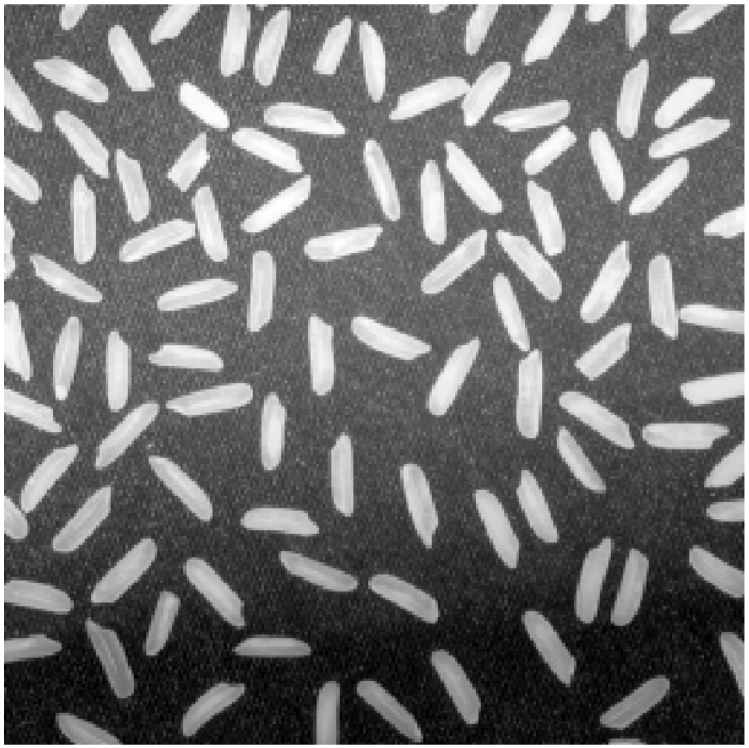
Gray-level mapping

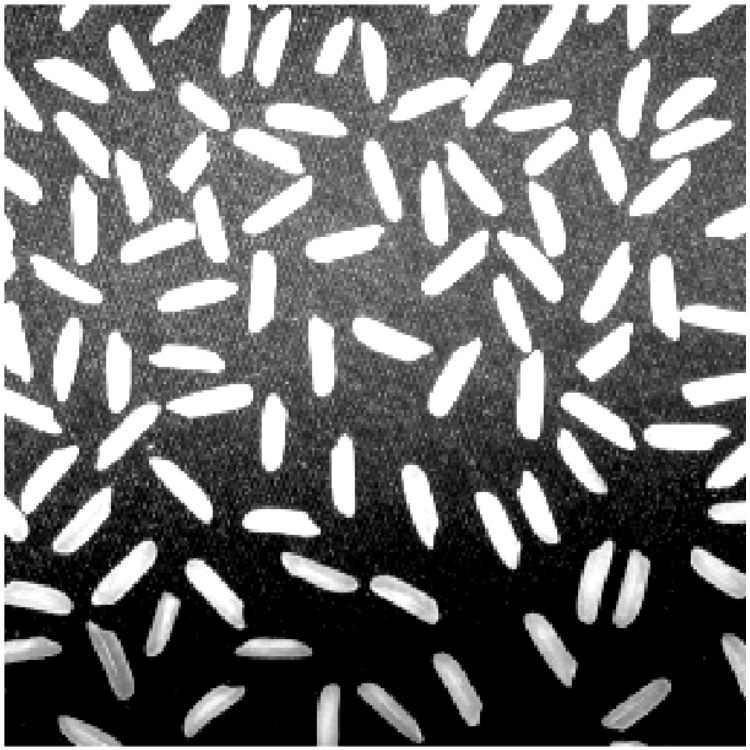
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?



Contrast Stretching





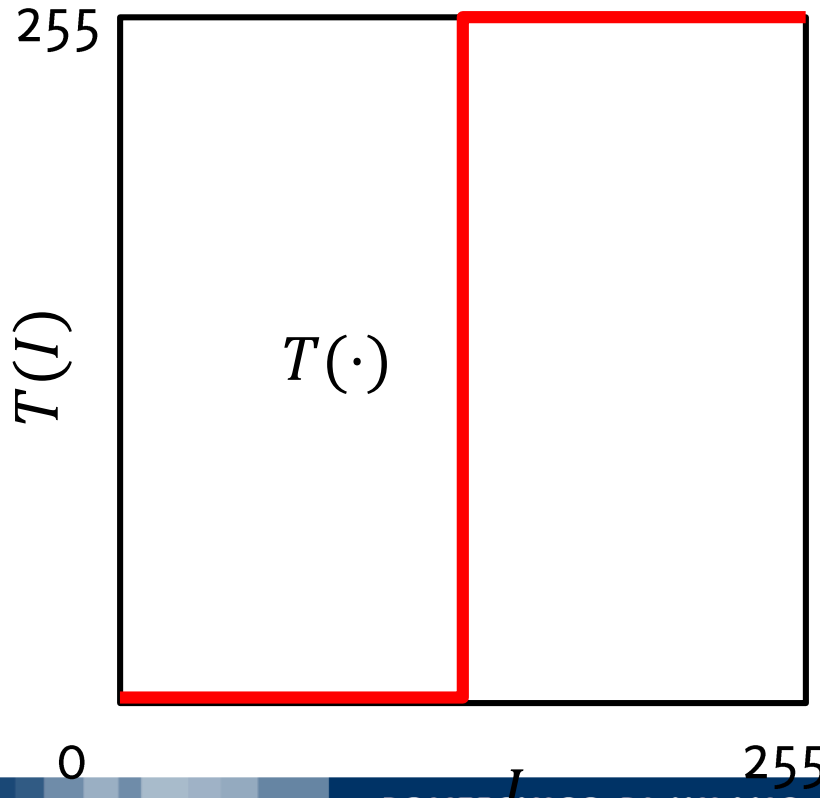
Contrast stretching: increases the constant at values in the middle of intensity range, decreases contrast at bright and dark regions.

It is implemented by piecewise or parametric transformations

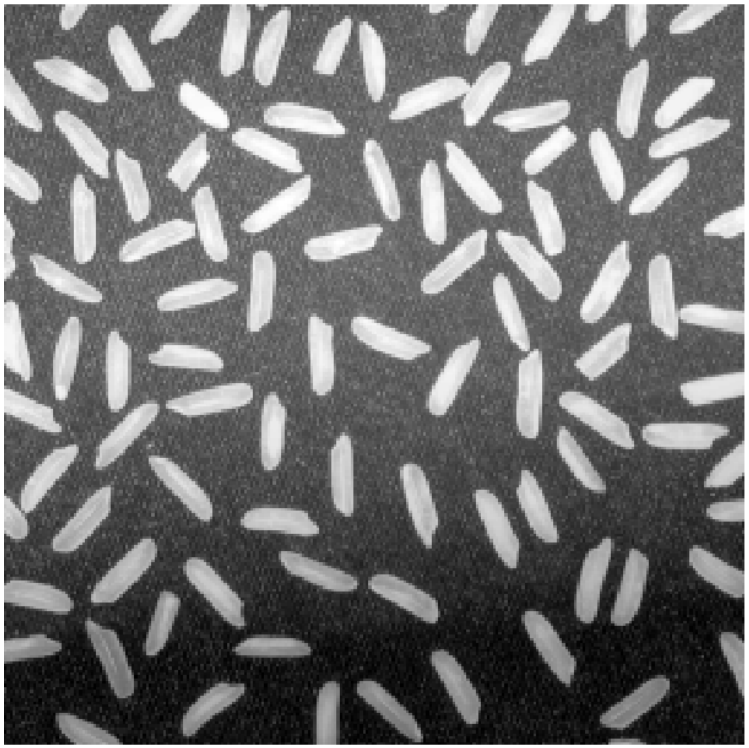
Gray-level mapping

A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?



Thresholding

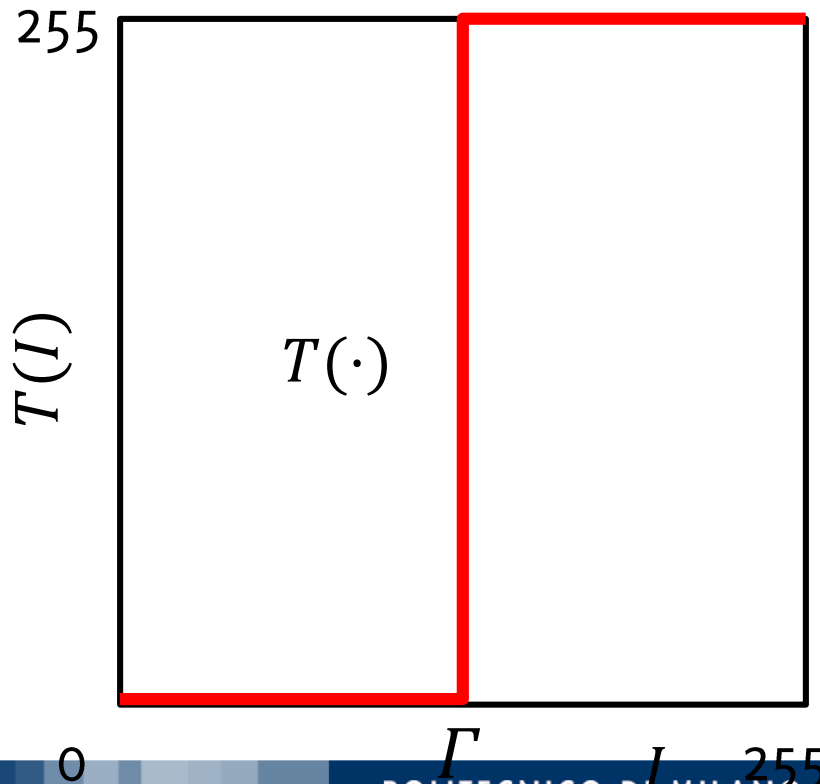


Thresholding binarizes images

Thresholding

A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

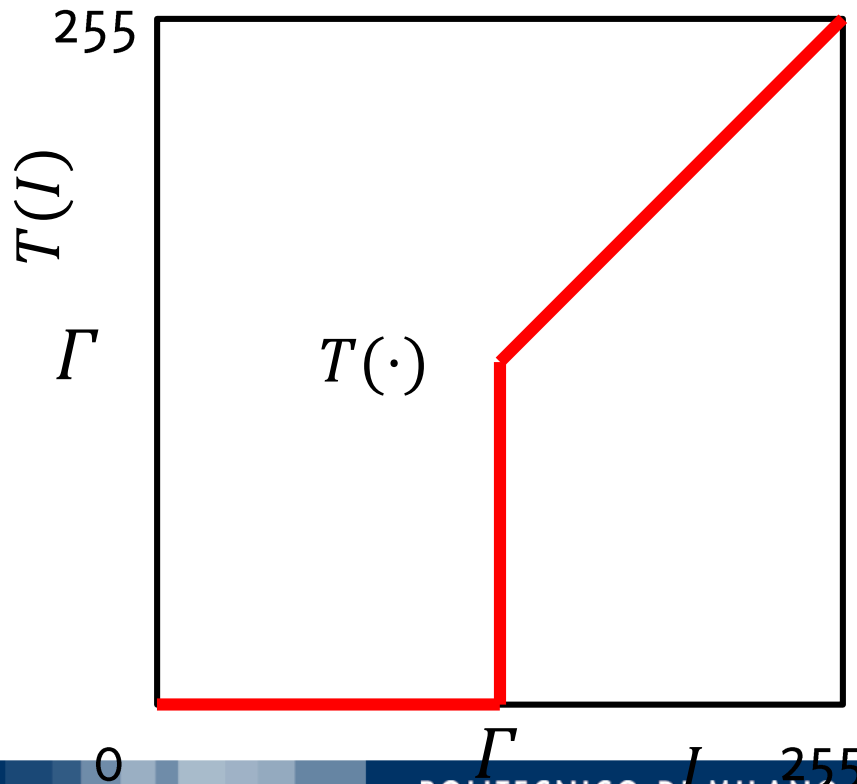
$$T\big(I(r,c)\big) = \begin{cases} 255, & \text{if } I(r,c) \geq \Gamma \\ 0, & \text{if } I(r,c) < \Gamma \end{cases}$$



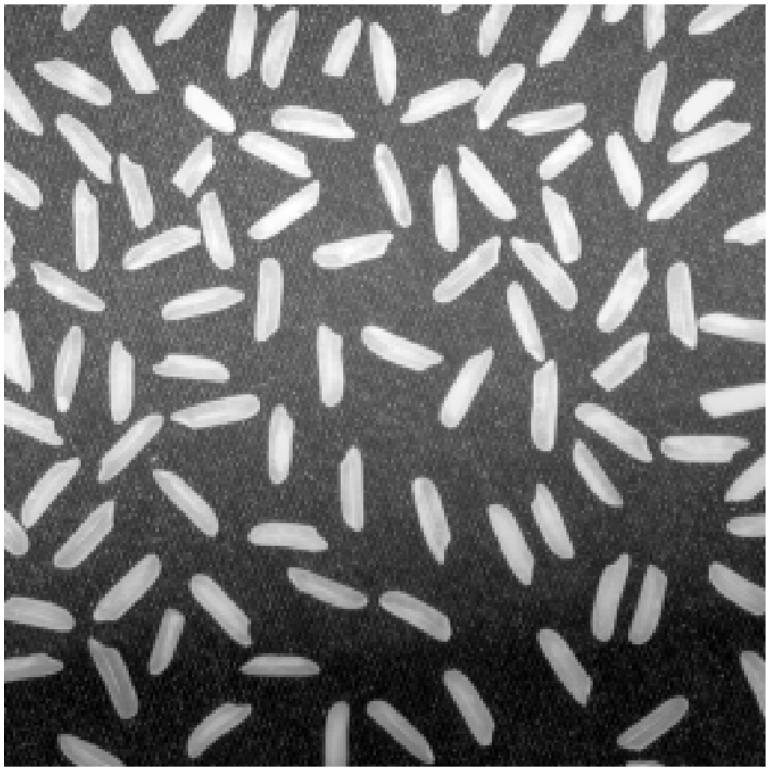
Thresholding

A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?



Thresholding

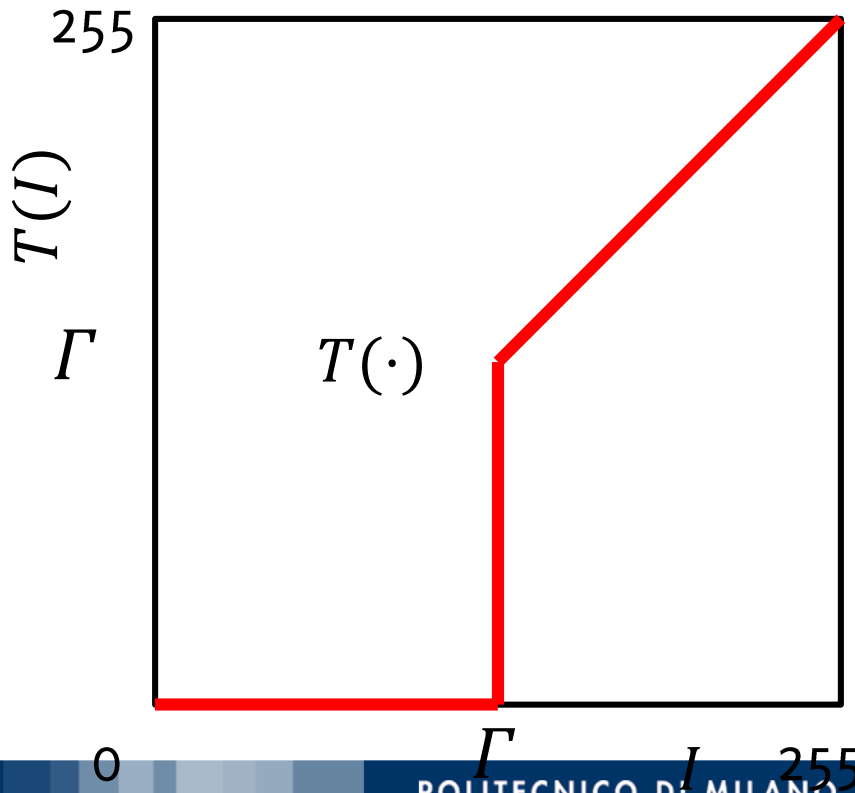


Thresholding

A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

$$T\big(I(r,c)\big) = \begin{cases} T\big(I(r,c)\big), & \text{if } I(r,c) \geq \Gamma \\ 0, & \text{if } I(r,c) < \Gamma \end{cases}$$

This simple operation is one of the most frequently used to add nonlinearities in CNN, through the ReLU Layers



Local (Spatial) Transformations: Correlation and Convolution

Most important image processing operations for classification problems

Local (Spatial) Transformation

In general, these can be written as

$$G(r,c) = T_{U,(r,c)}[I]$$

Where

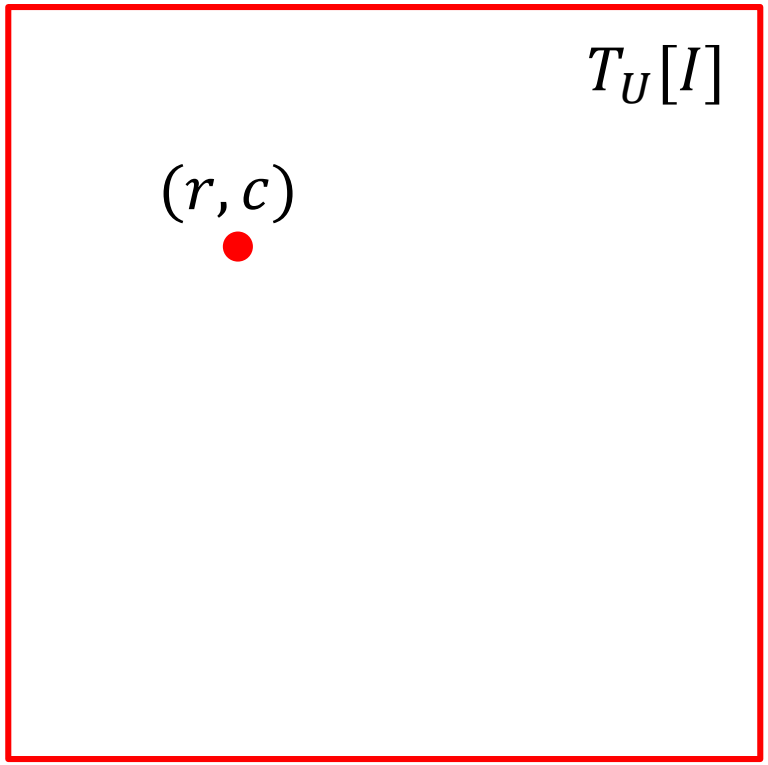
- I is the input image to be transformed
- G is the output
- U is a neighbourhood, identifies a region of the image that will concur in the output definition
- T_U: $\mathbb{R}^3 \to \mathbb{R}^3$ or T_U: $\mathbb{R}^3 \to \mathbb{R}$ is a function

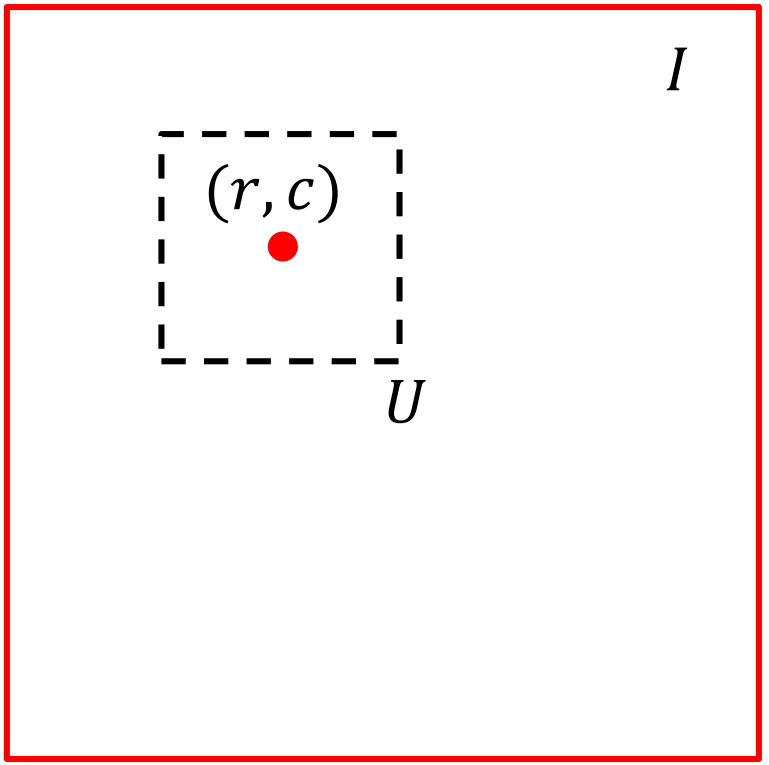
T operates on I "around" U

The output at pixel (r,c) i.e., $T_{U,(r,c)}[I]$ is defined by all $\{I(x,y),\ \ (x-r, y-c) \in U\}$

Local (Spatial) Filters

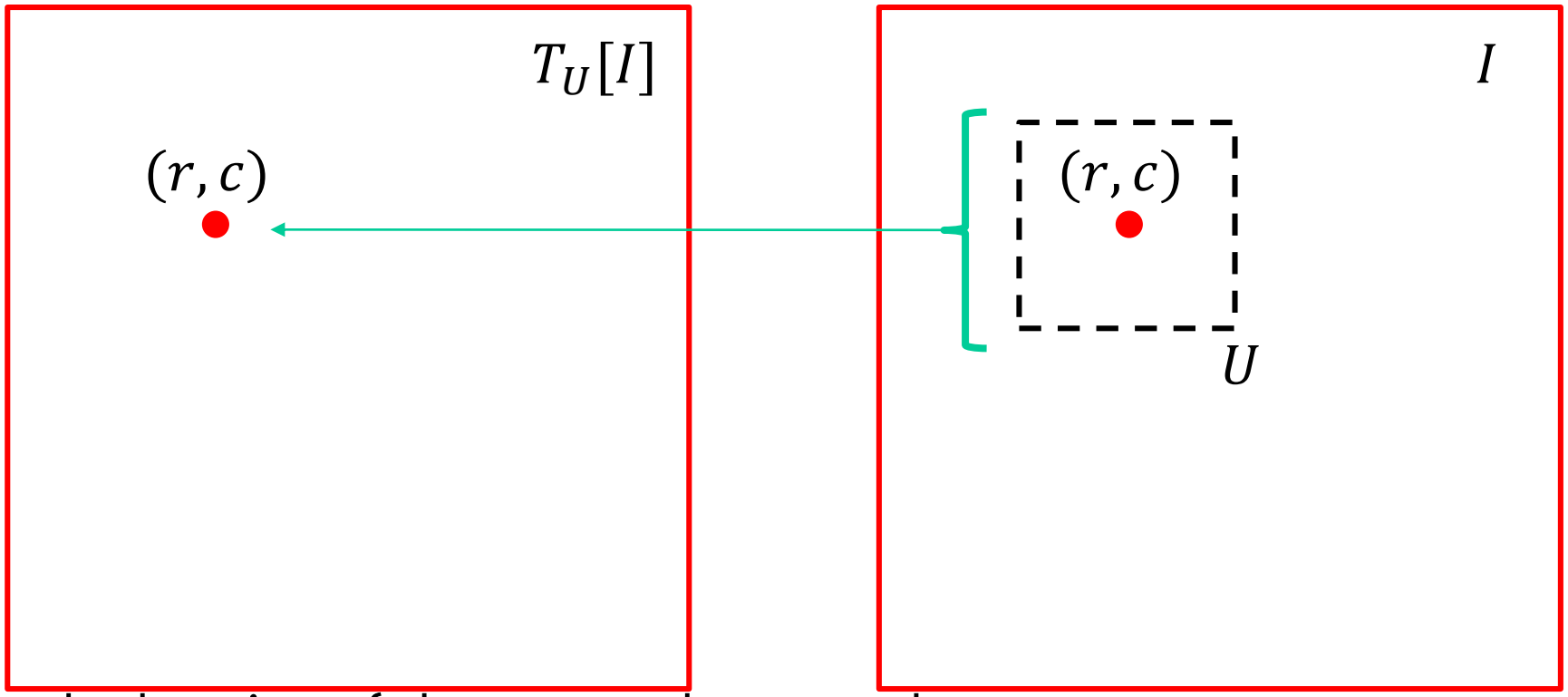
The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$





Local (Spatial) Filters

The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$



The location of the output does not change

The operation is repeated for each pixel

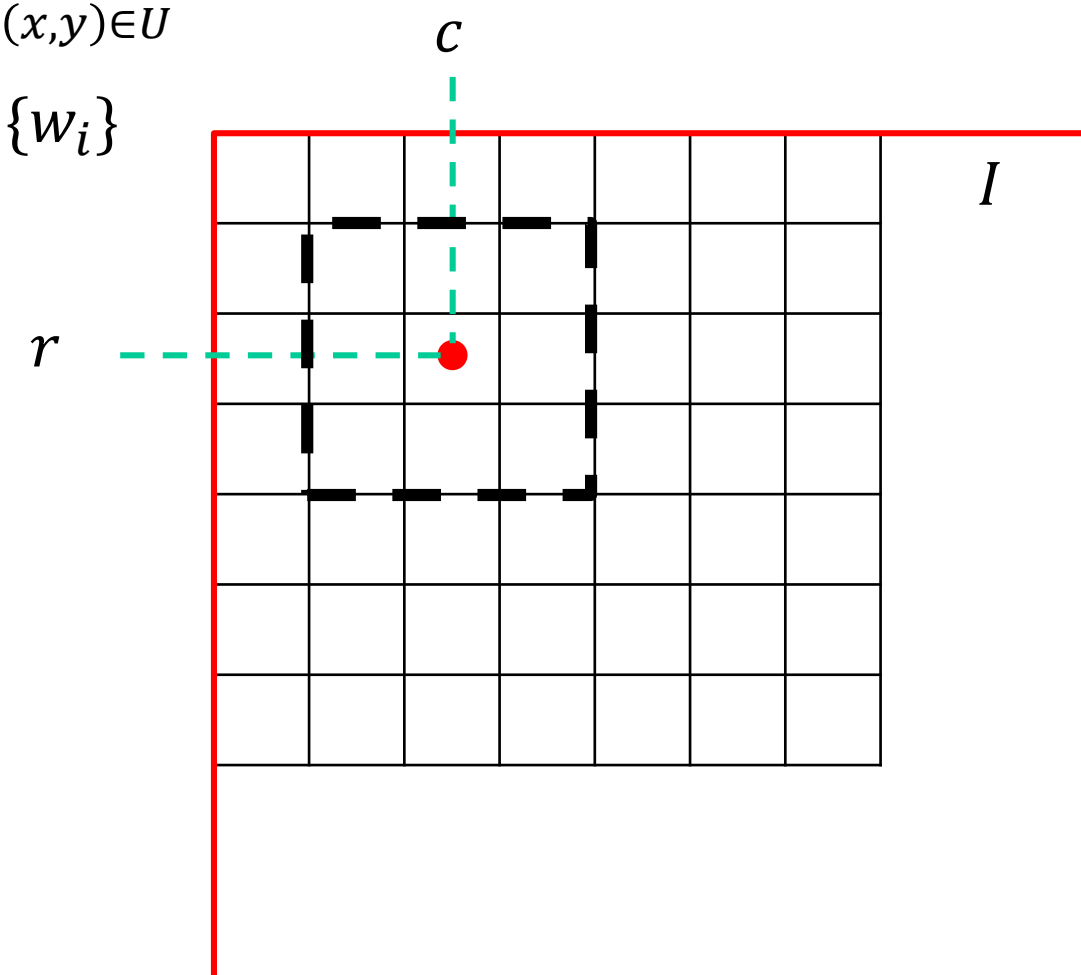
T can be either linear or nonlinear

Local Linear Filters

Linear Transformation: Linearity implies that

$$T[I](r,c) = \sum_{(x,y)\in U} w_i * I(r+x, c+y)$$

Considering some weights $\{w_i\}$



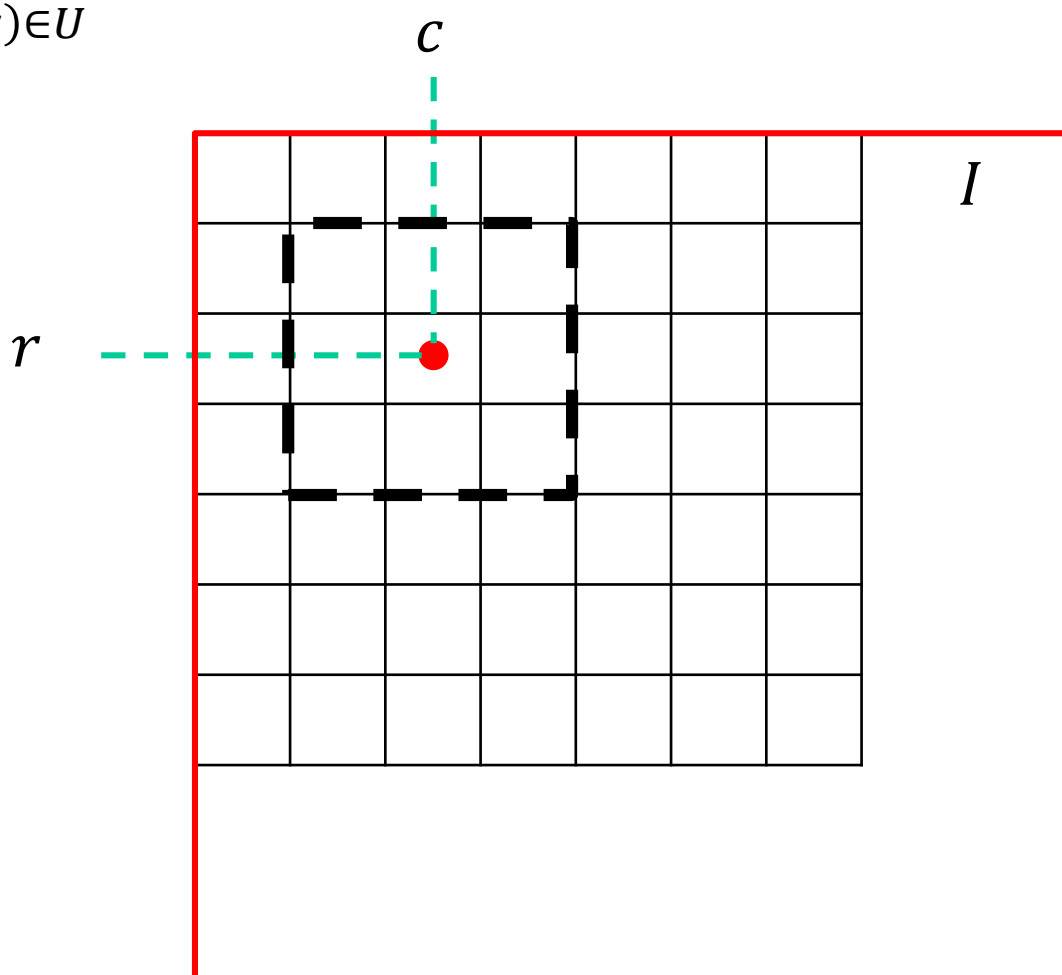
Local Linear Filters

Linear Transformation: Linearity implies that

$$T[I](r,c) = \sum_{(x,y)\in U} w(x,y) * I(r+x, c+y)$$

U

$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-11)$	$w(1,0)$	$w(1,1)$



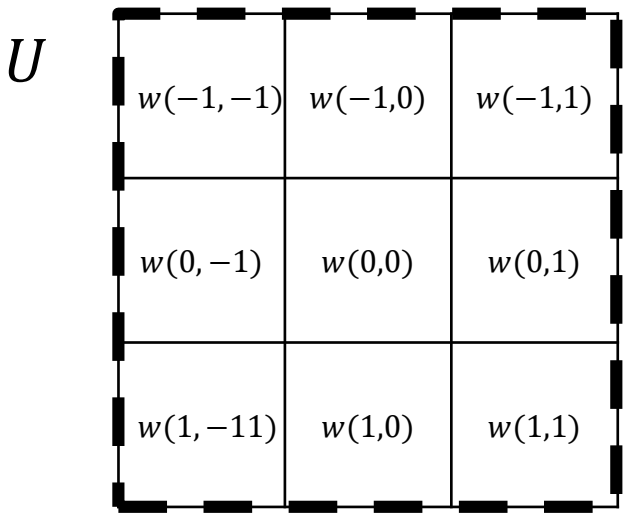
We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation

Local Linear Filters

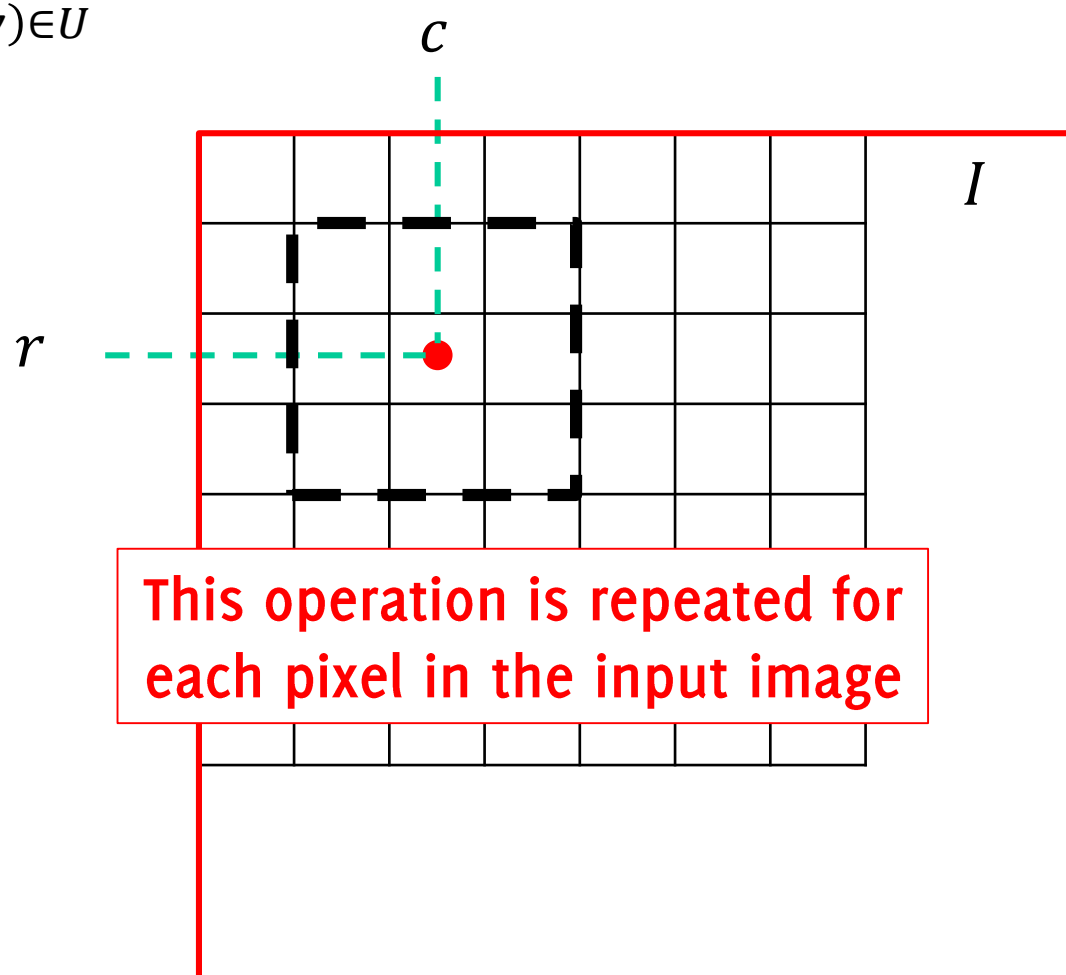
Linear Transformation: Linearity implies that

$$T[I](r,c) = \sum_{(x,y)\in U} w(x,y) * I(r+x, c+y)$$



We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation



This operation is repeated for each pixel in the input image

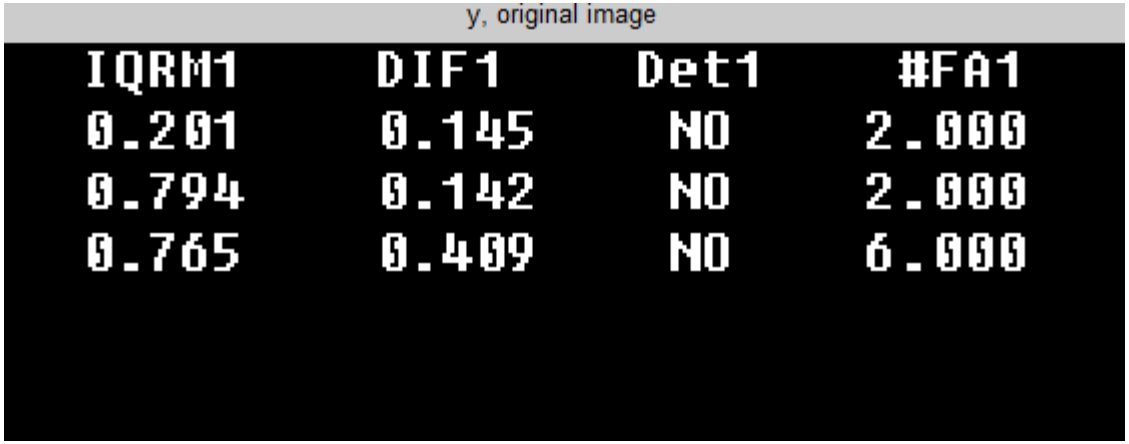
Correlation

The **correlation** among a filter h and an image is defined as

$$(I \otimes h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} h(u,v) * I(r+u, c+v)$$

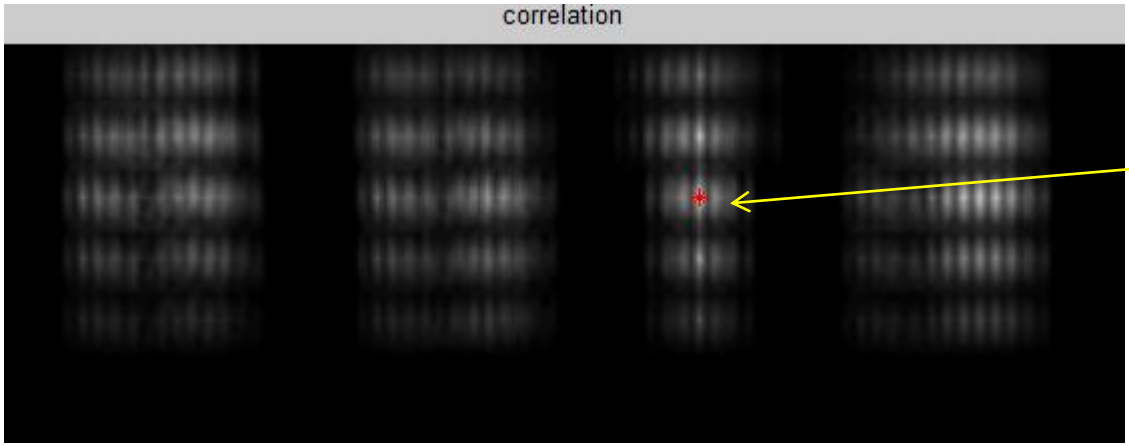
where the filter h is of size $(2L+1) \times (2L+1)$

Another example



*

=



The maximum is here

However…

y, original image

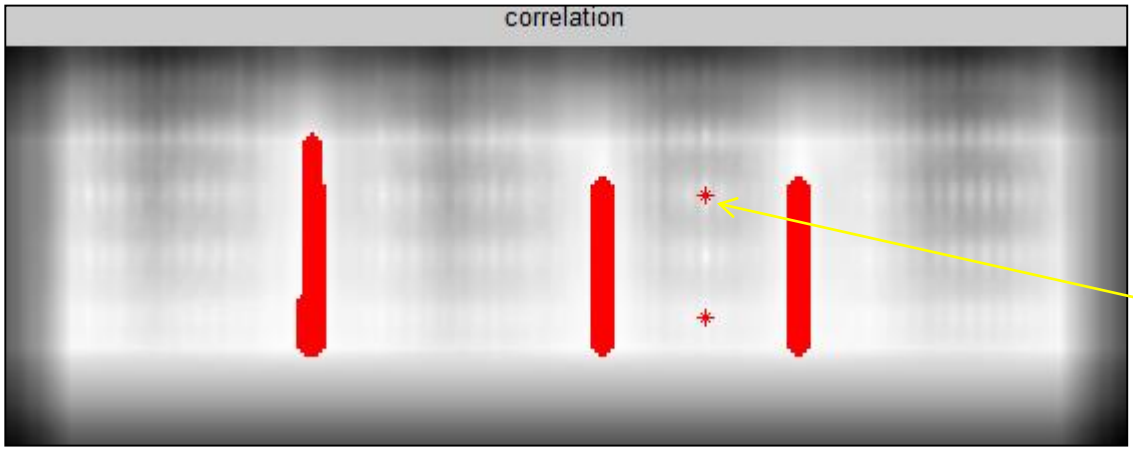
IQRM1	DIF1	Det1	#FA1
0.201	0.145	NO	2.000
0.794	0.142	NO	2.000
0.765	0.409	NO	6.000

* template

NO
NO
NO

=

correlation



Each point in a white area is as big as the template achieve the maxium value (togheter with the perfect match)

Convolution

Correlation and Convolution

The **correlation** among a filter h and an image is defined as

$$(I \otimes h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} h(u,v) * I(r+u, c+v)$$

where the filter h is of size $(2L+1) \times (2L+1)$

The **convolution** among a filter h and an image is defined as

$$(I \circledast h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} h(u,v) * I(r-u, c-v)$$

where the filter h is of size $(2L+1) \times (2L+1)$

There is just a swap in the filter before computing correlation!

Convolution – and filter flip

Let y, h be two discrete 2D signals of $(2L+1) \times (2L+1)$

$$z(i,j) = (y \circledast h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} y(r+u, c+v) h(-u,-v)$$

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

In this particular case $L = 1$ and both the image and the filter have size 3×3
The convolution is evaluated at $(r,c) = (0,0)$

Convolution – and filter flip

Let y, h be two discrete 2D signals of $(2L+1) \times (2L+1)$

$$z(i,j) = (y \circledast h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} y(r+u, c+v) h(-u,-v)$$

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Point-wise product

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$.*$

y_1	y_2	y_3
y_4	y_5	y_6
y_7	y_8	y_9

→

h_9y_1	h_8y_2	h_7y_3
h_6y_4	h_5y_5	h_4y_6
h_3y_7	h_2y_8	h_1y_9

Convolution

Let y, h be two discrete 2D signals of $(2L+1) \times (2L+1)$

$$z(i,j) = (y \circledast h)(r,c) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} y(r+u, c+v) h(-u,-v)$$

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Point-wise product

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$.*$

y_1	y_2	y_3
y_4	y_5	y_6
y_7	y_8	y_9

→

h_9y_1	h_8y_2	h_7y_3
h_6y_4	h_5y_5	h_4y_6
h_3y_7	h_2y_8	h_1y_9

sum →

$$z_5 = h_9y_1 + h_8y_2 + h_7y_3 + h_6y_4 + h_5y_5 + h_5y_6 + h_3y_7 + h_2y_8 + h_1y_9$$

Question

The filter (a.k.a. the kernel) yields the coefficients used to compute the linear combination of the input to obtain the output

1	3	0
2	10	2
4	1	1

Image

$*$

1	0	-1
1	0.1	-1
1	0	-1

Kernel

$=$

	?	

Filter Output

Let’s have a look at 1D convolution

To better understand: let's have a look at 1D Convolution

Let us consider a 1d signal y and a filter h.

- Their convolution is also a signal $z = y \otimes h$.
- For continuous-domain 1D signals and filters

$$z(\tau) = (y \otimes h)(\tau) = \int_{\mathbb{R}} y(t)h(\tau - t)dt$$

that is equivalent to

$$z(\tau) = (h \otimes y)(\tau) = \int_{\mathbb{R}} y(\tau - t)h(t)dt$$

- For discrete signals and filters

$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^{L} y(n - m)h(m)$$

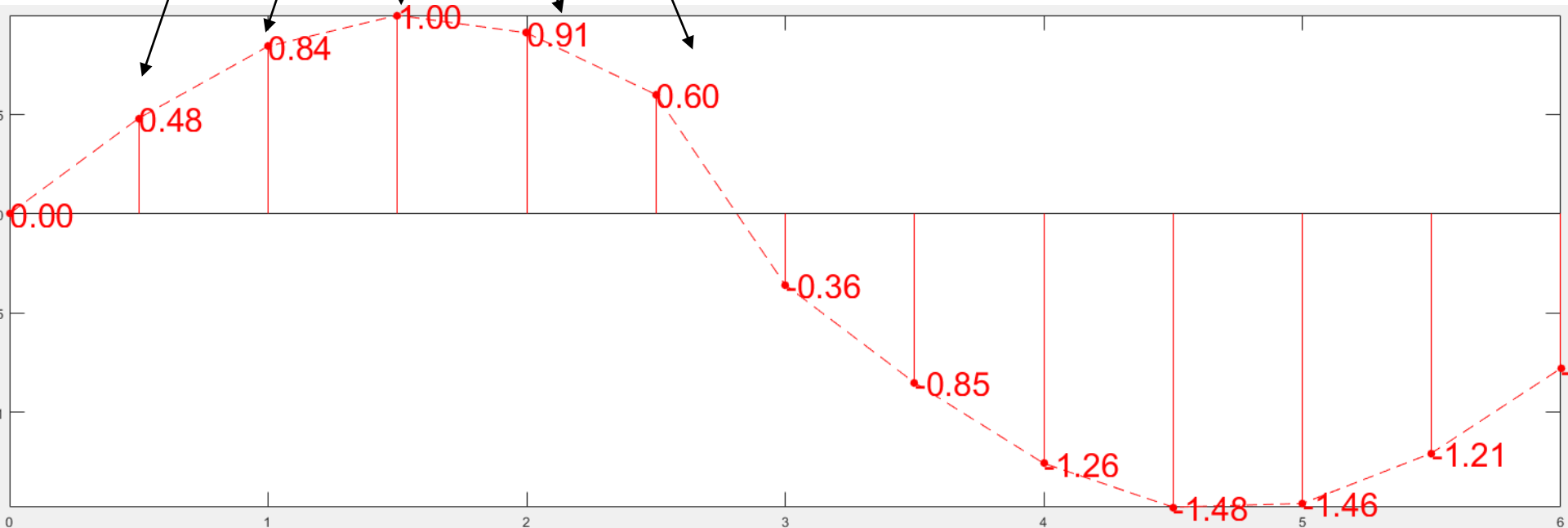
where the filter has $(2L + 1)$ samples

1D Convolution - example

$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^{L} y(n-m)h(m)$$

$$y = \sin(2x)\,, h = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right], L = 2$$

$$\left[\frac{1}{5}, \quad \frac{1}{5}, \quad \frac{1}{5}, \quad \frac{1}{5}, \quad \frac{1}{5}\right]$$

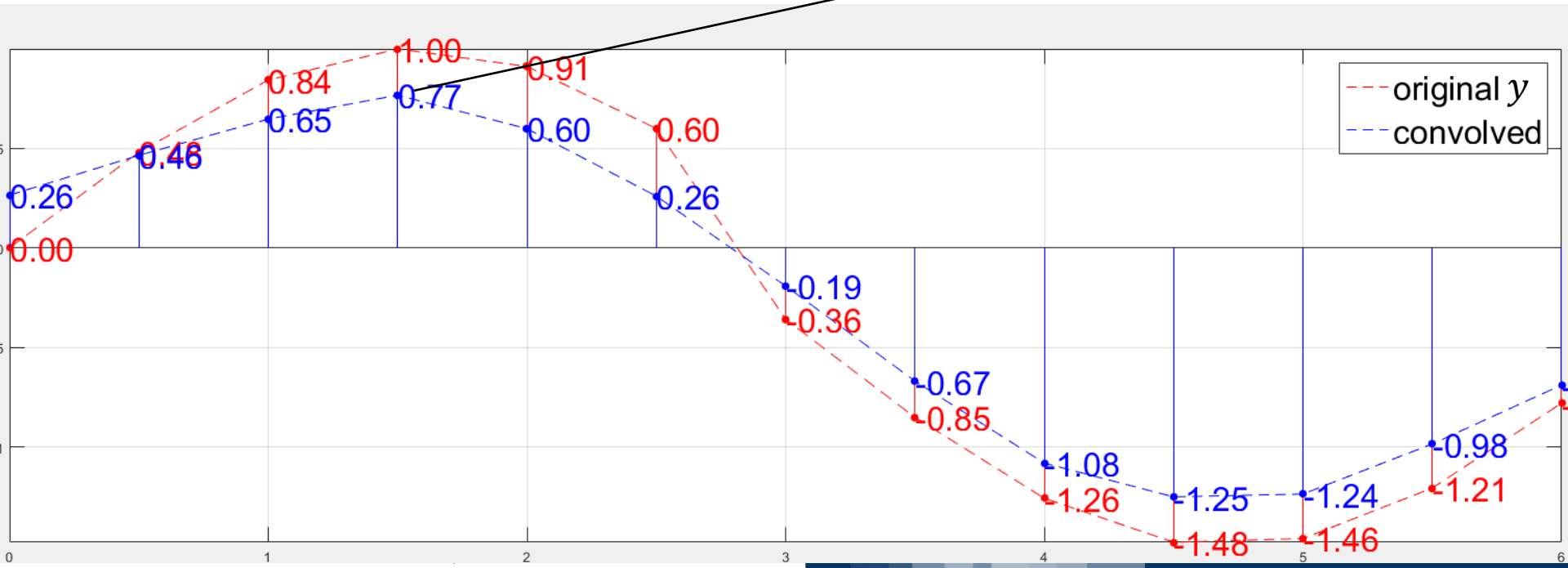


1D Convolution - example

$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^{L} y(n-m)h(m)$$

$$y = \sin(2x)\,, h = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right], L = 2$$

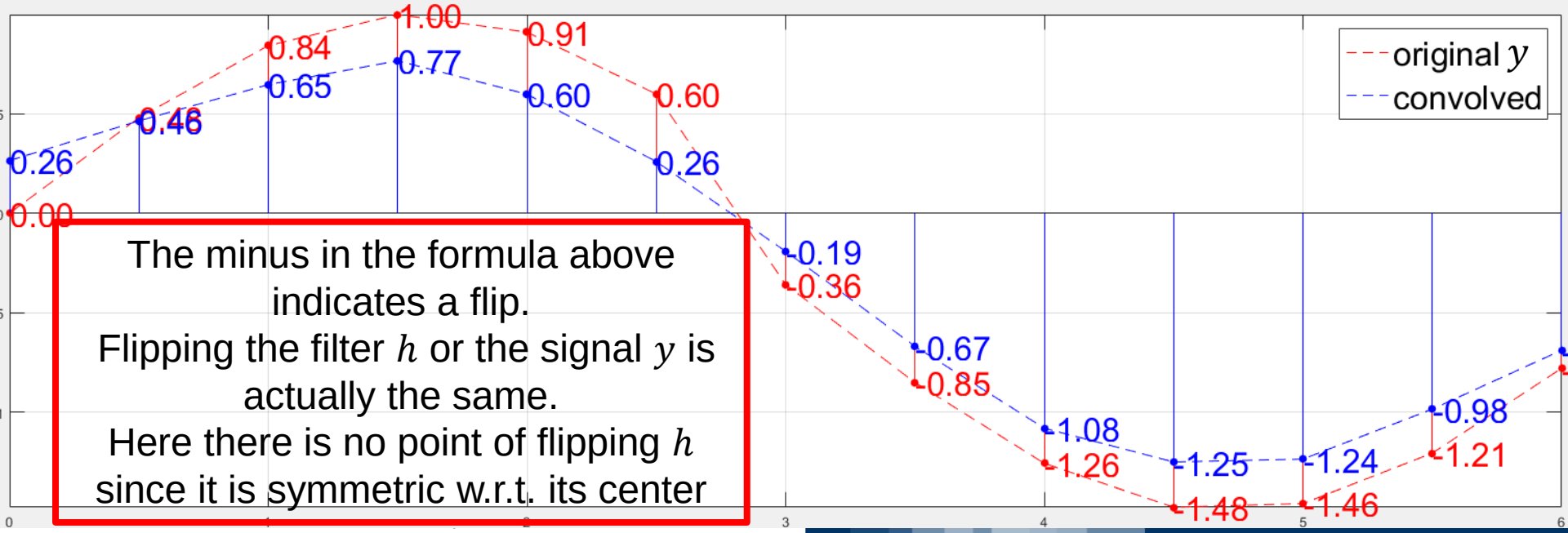
$$0.766 \approx \frac{1}{5} * 0.48 + \frac{1}{5} * 0.84 + \frac{1}{5} * 1 + \frac{1}{5} * 0.91 + \frac{1}{5} * 0.60$$



1D Convolution - example

$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^{L} y(n-m)h(m)$$

$$= \sum_{m=-L}^{L} y(n+m)h(-m)$$

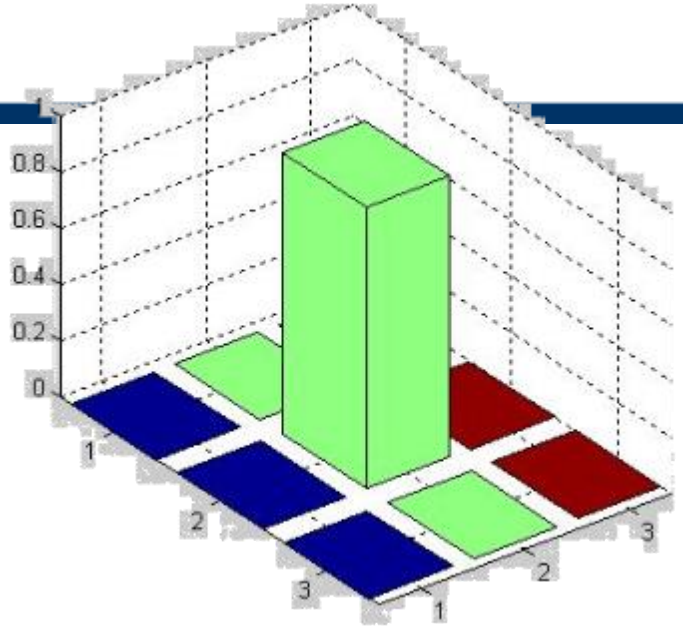


The minus in the formula above indicates a flip.
Flipping the filter h or the signal y is actually the same.
Here there is no point of flipping h since it is symmetric w.r.t. its center

Let’s go back to
2D convolution now

A well-known Test Image - Lena

A Trivial example

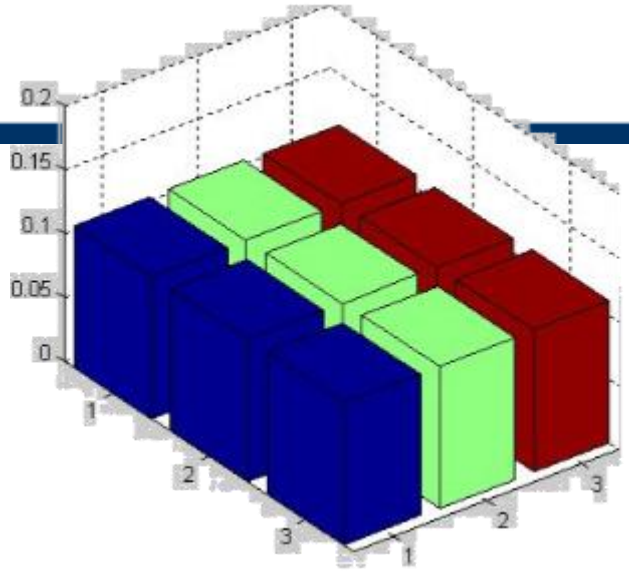


*

0	0	0
0	1	0
0	0	0

=

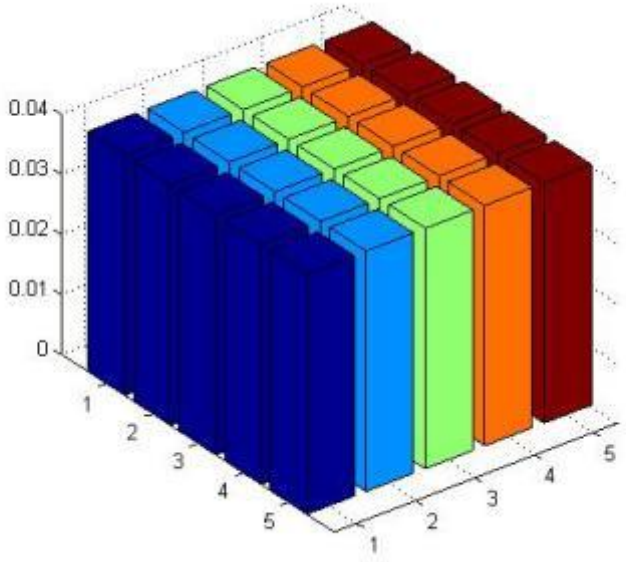
Linear Filtering



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

The original Lena image

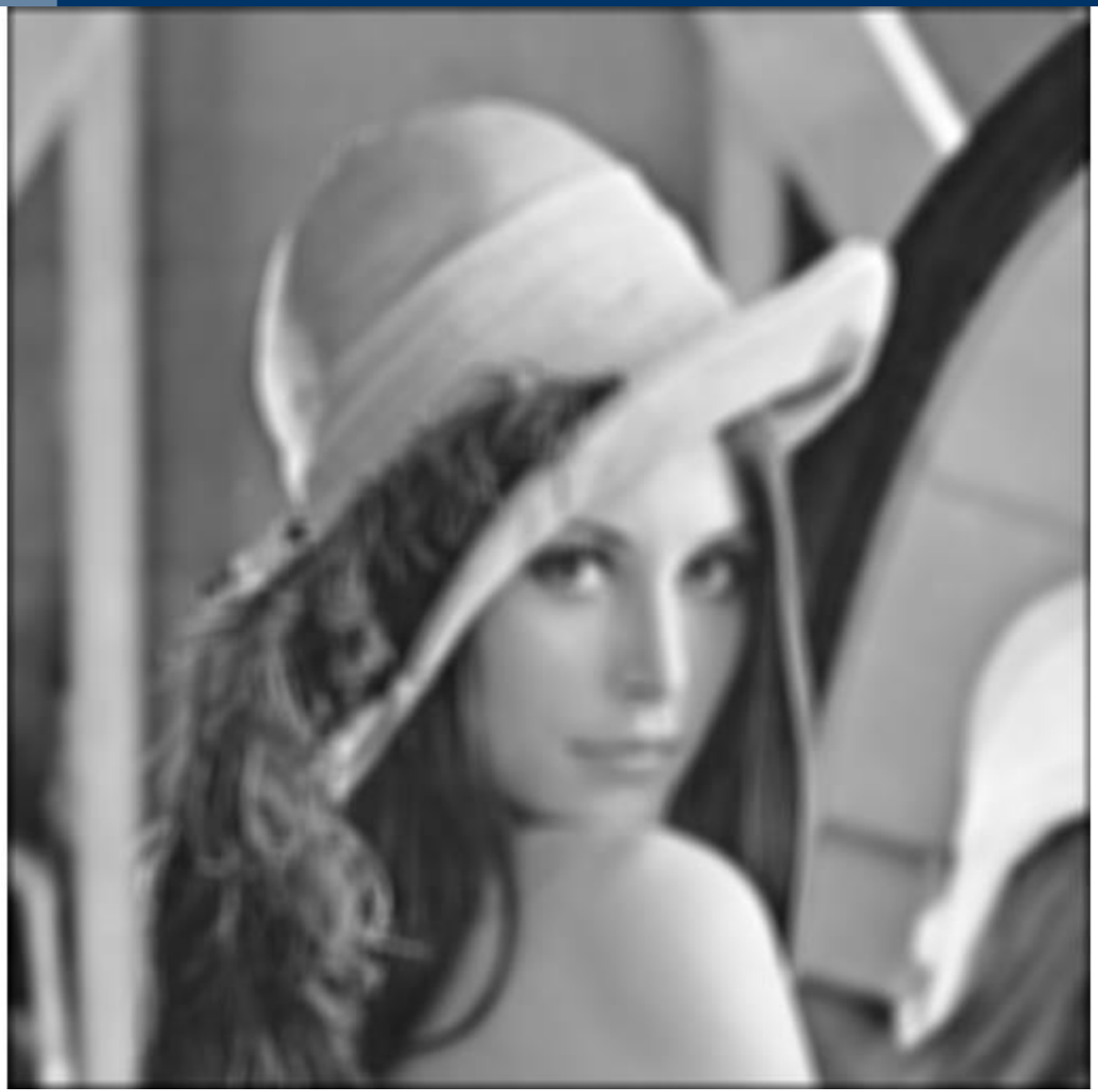
Filtered Lena Image



$$* \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

The original Lena image

The filtered Lena image



What about normalization?

$$* \frac{2}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} =$$

… convolution is linear

…what about

$$\frac{2}{25} \bullet$$

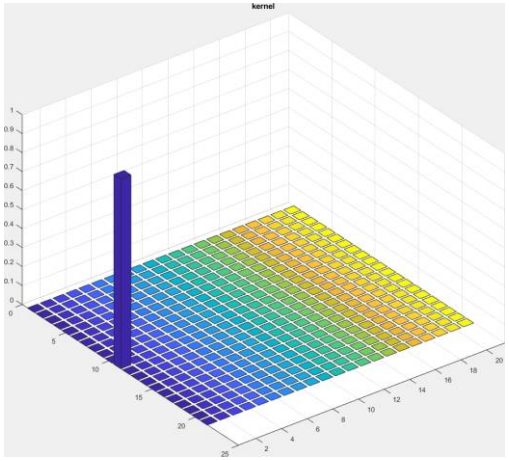
$$*$$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$$=$$

… convolution is linear

Translation



$\circledast$

0	0	0
1	0	0
0	0	0

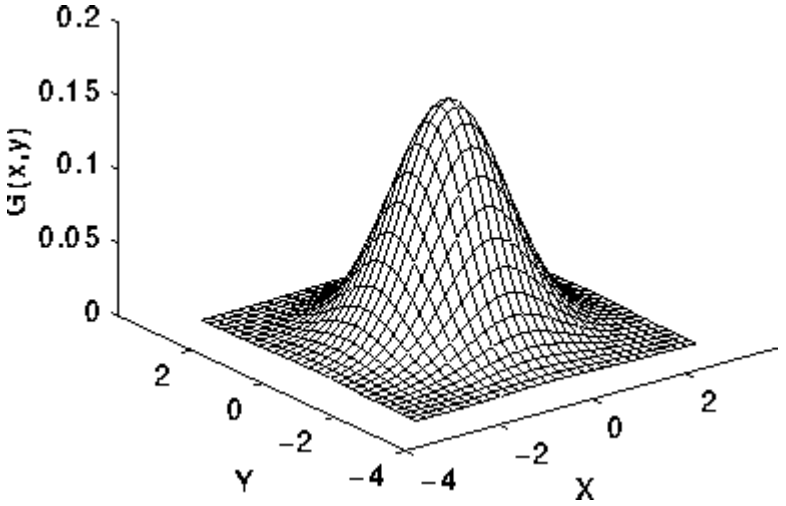
$=$

Remember the filter has to be flipped before convolution

2D Gaussian Filter

Continuous Function

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



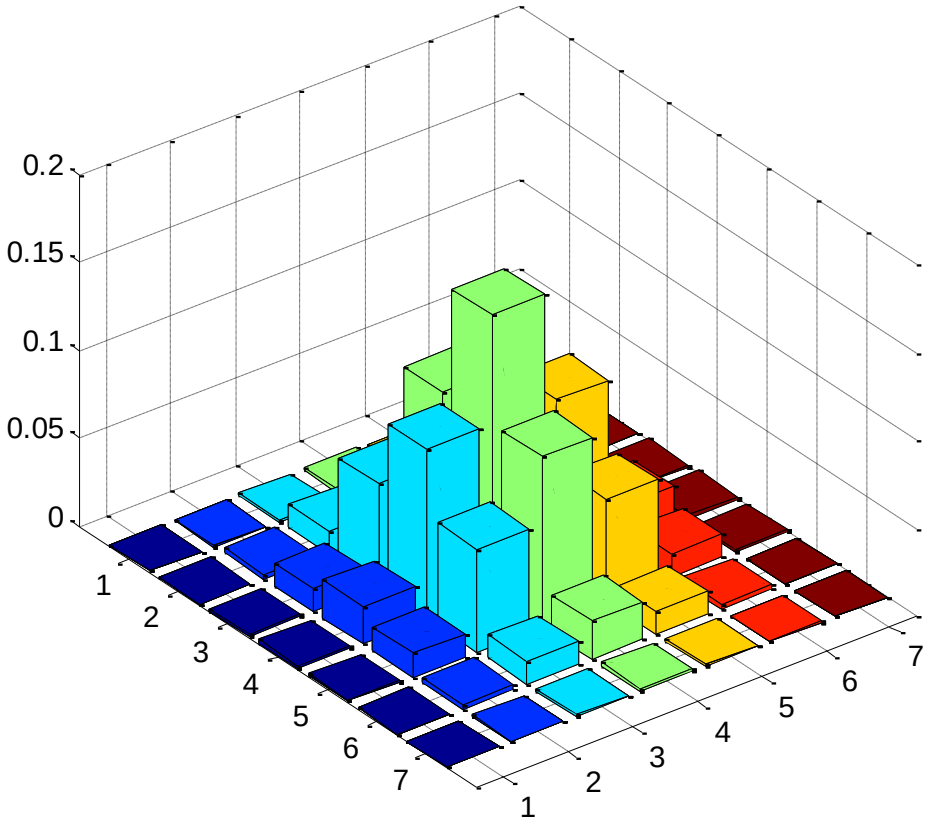
Discrete kernel: assuming H is a $(2k+1) \times (2k+1)$ filter

$$H(i,j) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i^2+j^2\right)}{2\sigma^2}\right)$$

That is then normalized such that $\sum_{i=-k}^{k}\sum_{j=-k}^{k} H(i,j) = 1$

Weighted local averaging filters: Gaussian Filter

*



Weighted local averaging filters: Gaussian Filter

Gaussian Smoothing vs Averaging Filters

Gaussian Smoothing
Support 7x7

Smoothing by Averaging
On 7x7 window

Convolution Properties

Properties of Convolution: Commutative

It is a linear operator

$$\big((\lambda I_1 + \mu I_2) \circledast h\big)(r,c) = \lambda (I_1 \circledast h)(r,c) + \mu (I_2 \circledast h)(r,c)$$

where $\lambda, \mu \in \mathbb{R}$

Obviously, when the filter is center-symmetric, convolution and correlation are equivalent

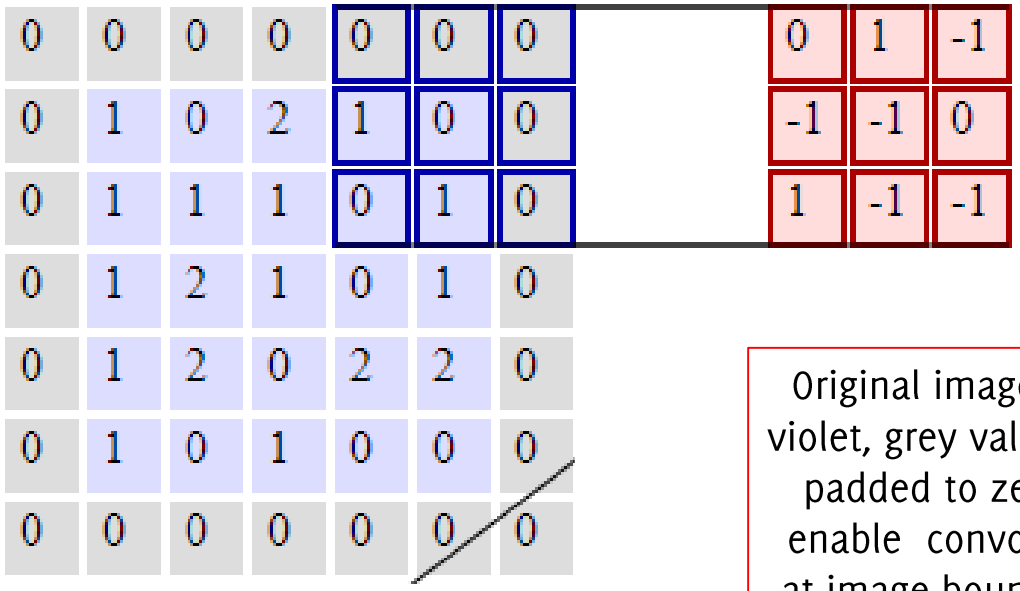
Properties of Convolution

It is **commutative (in principle)**

$$I_1 \circledast I_2 = I_2 \circledast I_1$$

However, in discrete signals it depends on **the padding criteria** In continuous domain it holds as well as on periodic signals

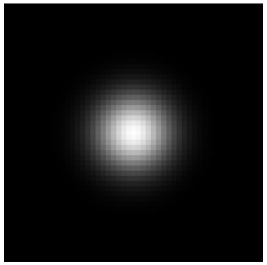
Input image I filter h



Original image is in violet, grey values are padded to zero to enable convolution at image boundaries

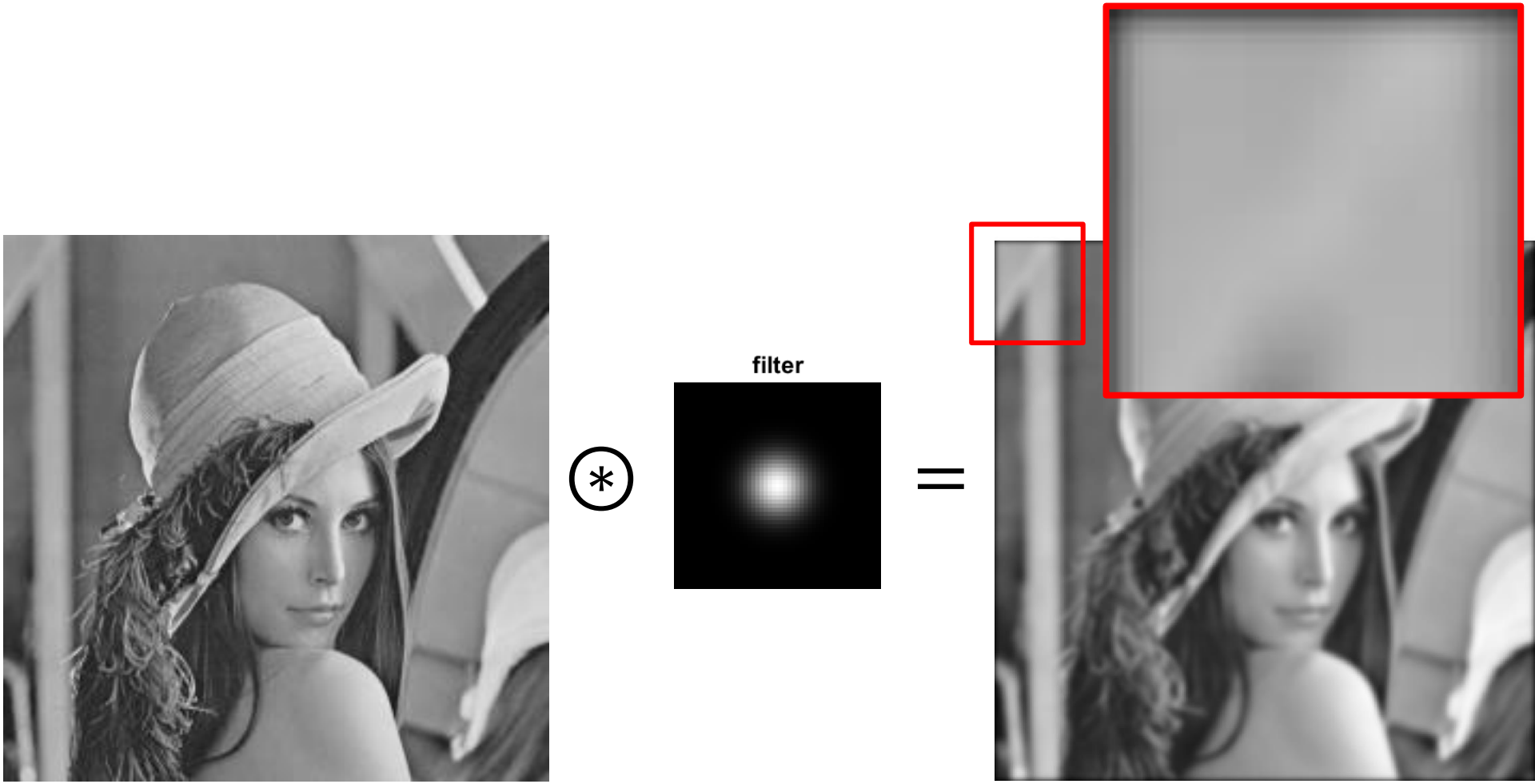
Is Convolution Commutative?

$*$



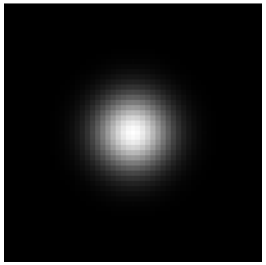
$=$

Is Convolution Commutative?



Is Convolution Commutative?

filter



$*$

$=$

Properties of Convolution: Associative

It is also **associative**

$$f \circledast (g \circledast h) = (f \circledast g) \circledast h = f \circledast g \circledast h$$

and **dissociative**

$$f \circledast (g + h) = f \circledast g + f \circledast h$$

Properties of Convolution: Shift invariance

It is also **associative**

$$f \circledast (g \circledast h) = (f \circledast g) \circledast h = f \circledast g \circledast h$$

and **dissociative**

$$f \circledast (g + h) = f \circledast g + f \circledast h$$

It is **shift-invariant**, namely

$$(I(\cdot - r_0, \cdot - c_0) \circledast h)(r, c) = (I \circledast h)(r - r_0, c - c_0)$$

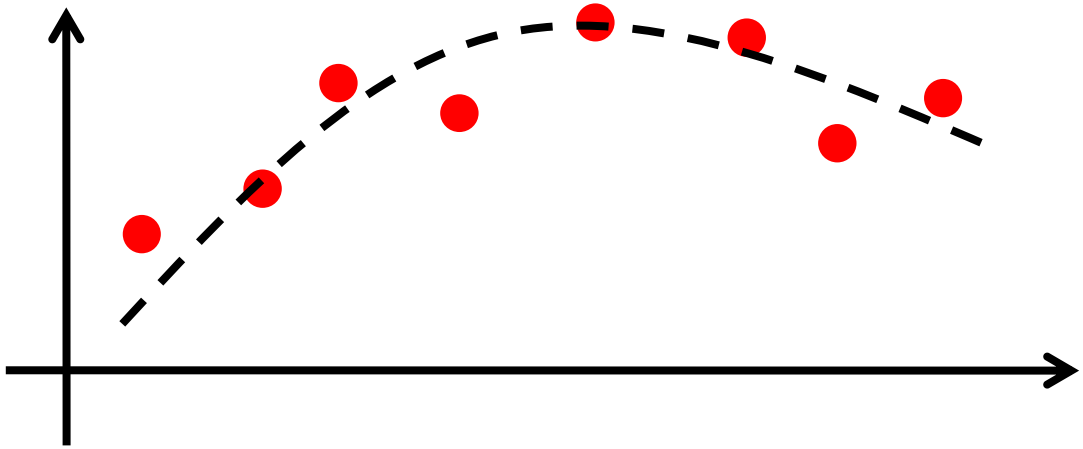
Any **linear and shift invariant system can be written as a convolution**

Convolution and Regression

Observation model is

$$z(x) = y(x) + \eta(x) \quad x \in X$$

Consider a regression problem



Fitting and Convolution

The convolution provides the BLUE (Best Linear Unbiased Estimator) for regression when the image y is constant

The problem: estimating the constant C fitting the noisy observations

$$\widehat{y_h}(x_0) = \underset{C}{\operatorname{argmin}} \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - C\right)^2$$

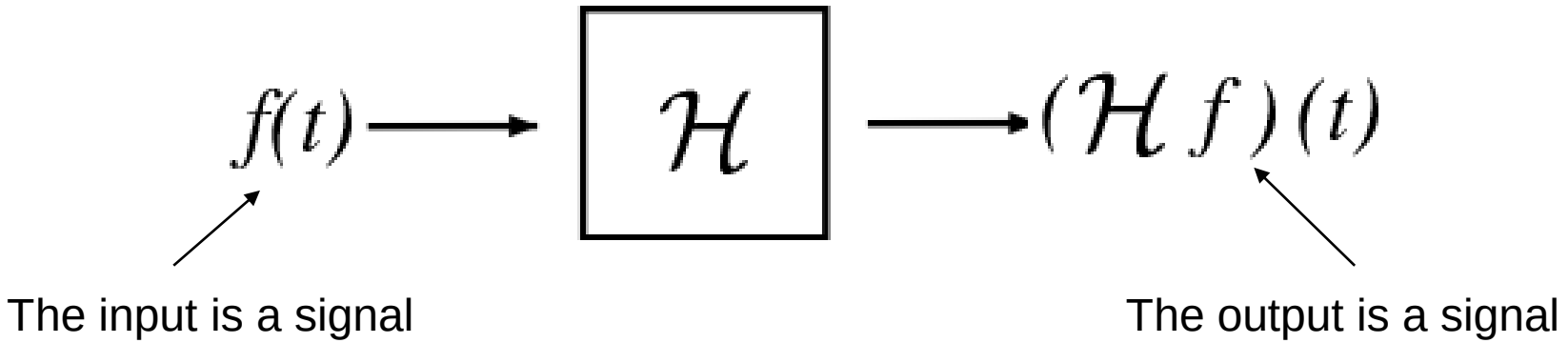
Where $w_h = \{w_h(x)\} \quad s.t. \quad \sum_{x \in X} w_h(x) = 1$

This problem can e solved by **computing the convolution** of the image z against a **filter whose coefficients are the error weights**

$$\widehat{y}(x_0) = \left(z \circledast w_h\right)(x_0)$$

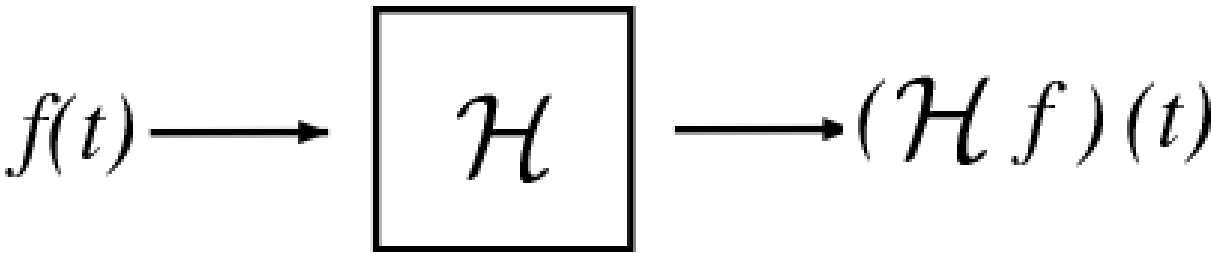
Systems

Consider a system H as a black box that processes an input signal (f) and gives the output (i.e, $H[f]$)



Systems

Consider a system H as a black box that processes an input signal (f) and gives the output (i.e, $H[f]$)



In our case, f is a digital image (a 2D matrix), but in principle could be any (analogic or digital) n-dimensional signal

Linearity and Time Invariance

A system is **linear** if and only if

$$H[\lambda f(t) + \mu g(t)] = \lambda H[f](t) + \mu H[g](t)$$

holds for any $\lambda, \mu \in \mathbb{R}$ and for f, g arbitrary signals (this is the canonical definition of linearity for an operator)

A system is **time (or shift) – invariant** if and only if

$$H[f(t - t_0)] = H[f](t - t_0)$$

holds for any $t_0 \in \mathbb{R}$ and for any signal f

Linear and Time Invariant Systems

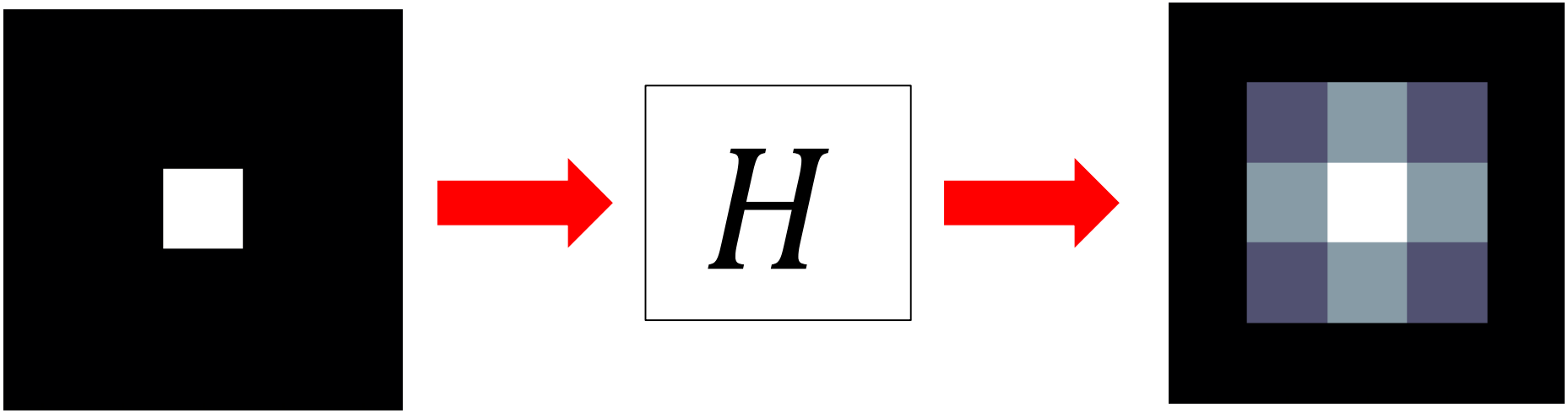
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

- LTI systems are **characterized** entirely by a **single function** called the **filter**

Linear and Time Invariant Systems

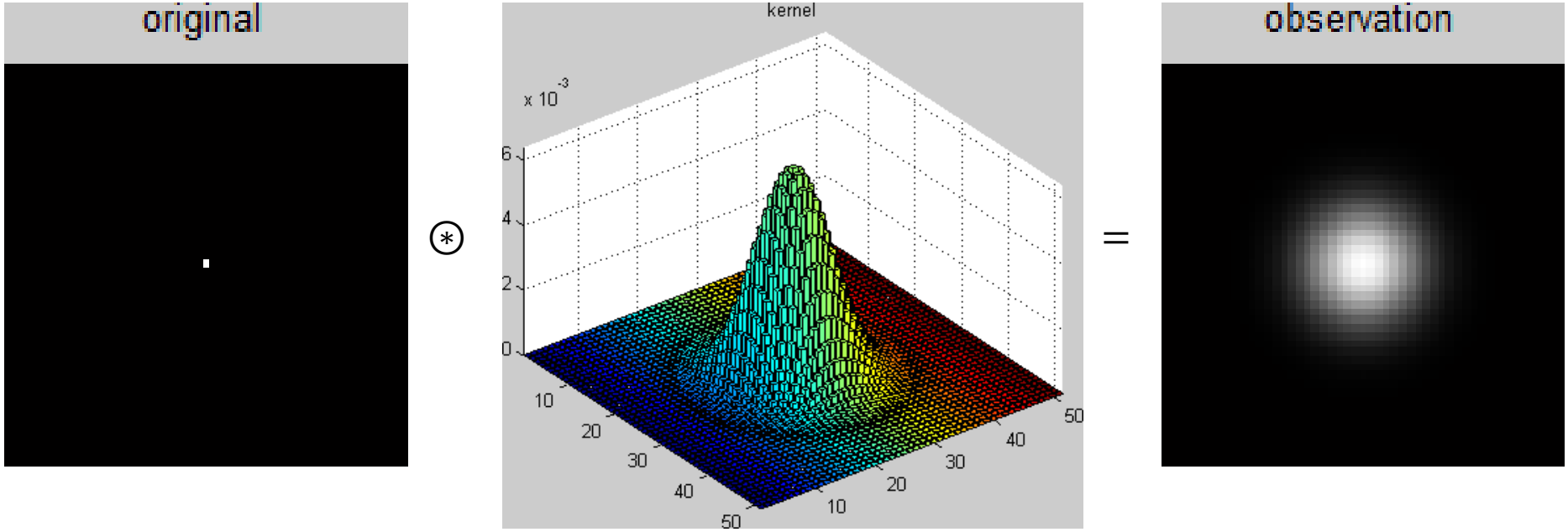
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

- LTI systems are **characterized** entirely by a **single function** called the **filter**
- The filter is also called system's the **impulse response** or **point spread function**, as it corresponds to the output of an impulse fed to the system



The Impulse Response

Take as input image a discrete Dirac

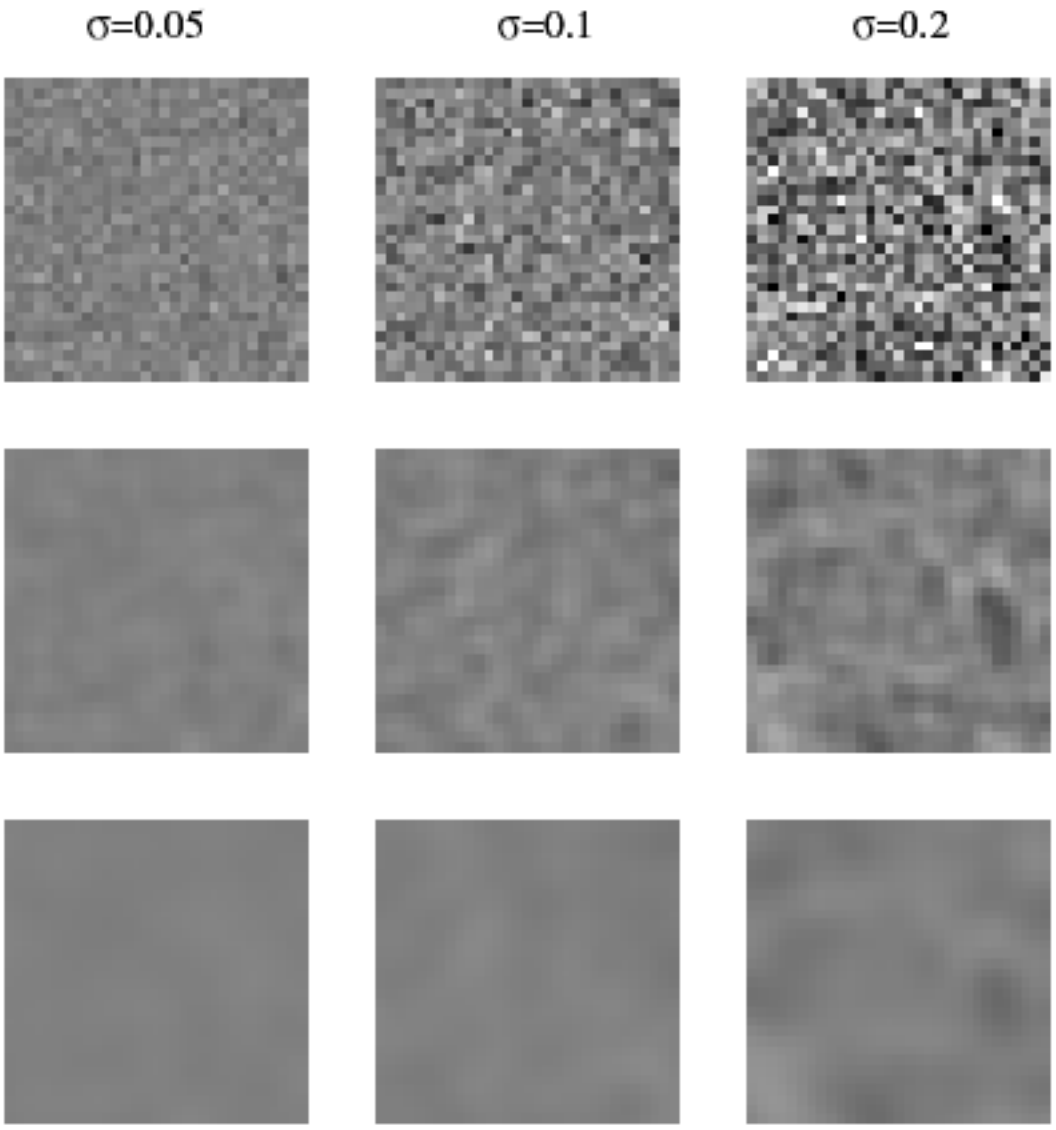


This is why h is also called the “Point Spread Function”

Denoising

An application scenario for digital filters

Low - Pass



The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

Denoising: The Issue

A Detail in Camera Raw Image

Denoising: The Issue

Denoised

Denoising: The Issue

A Detail in Camera Raw Image

Denoising: The Issue

Denoised

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x) \quad x \in X$$

z	Sensed image
x	$x \in X \subset \mathbb{R}^2$ Pixel index
y	Original (unknown and noisy free) image
η	Noise (stochastic phenomenon)

The goal is to obtain $\widehat{y}(x)$, a *realistic* estimate of $y(x)$, given $z(x)$ and the distribution of η.

For the sake of simplicity it is often assumed $\eta \sim N(0, \sigma^2)$ and independent.

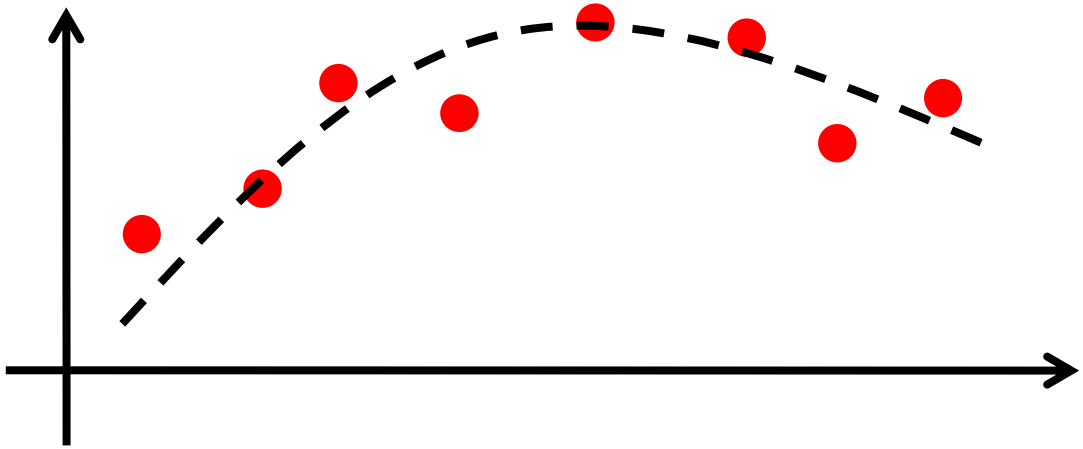
The noise standard deviation σ is assumed as known.

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x) \quad x \in X$$

Denoising as a regression problem



Fitting and Convolution

The convolution provides the BLUE (Best Linear Unbiased Estimator) for regression when the image y is constant

The problem: estimating the constant C fitting the noisy observations

$$\widehat{y_h}(x_0) = \underset{C}{\operatorname{argmin}} \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - C\right)^2$$

where $w_h = \{w_h(x)\}$ $s.t.$ $\sum_{x \in X} w_h(x) = 1$

This problem can e solved by computing the convolution of the signal z against a filter whose coefficients are the error weights

$$\widehat{y}(x_0) = \left(z \circledast w_h\right)(x_0)$$

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x) \quad x \in X$$

hus we can pursue a “regression-approach”, but on images it may not be convenient to assume a **parametric expression** of y on X

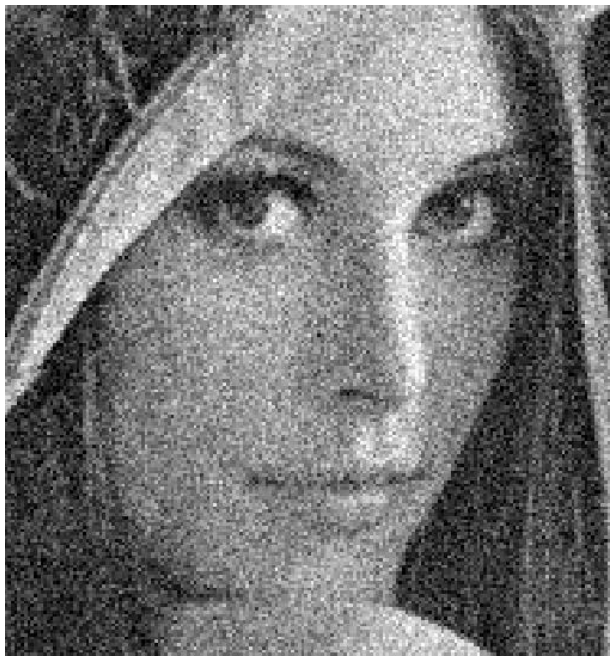
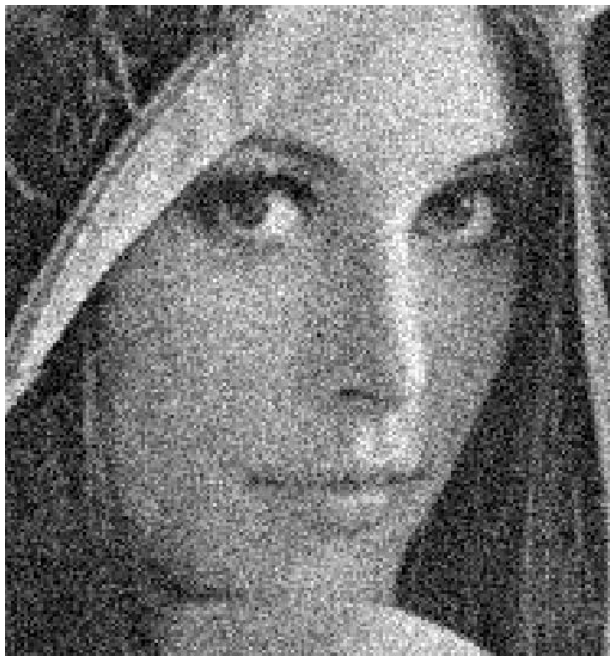
$z =$ 

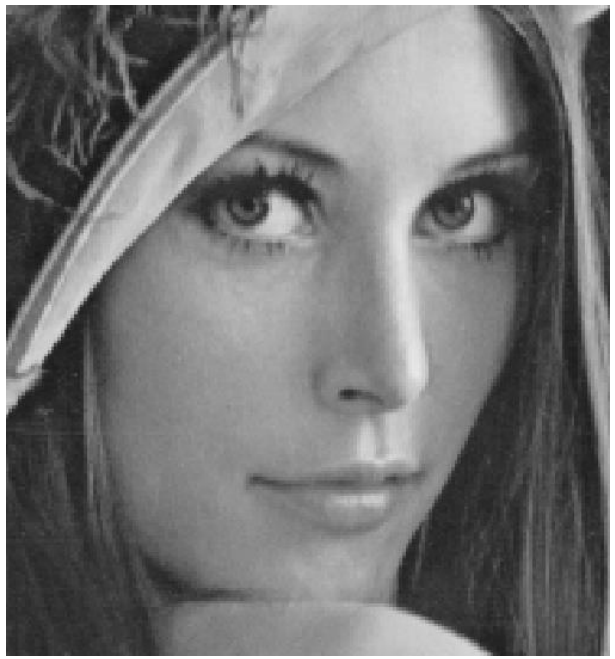
Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x) \quad x \in X$$

hus we can pursue a “regression-approach”, but on images it may not be convenient to assume a **parametric expression** of y on X

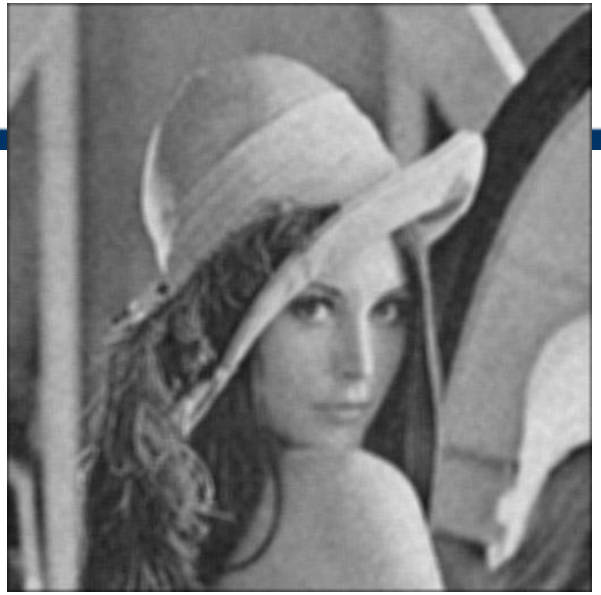
$z =$ 

$y =$ 

Local Smoothing

Additive Gaussian
White Noise

$$\eta \approx N(\mu, \sigma)$$



After Averaging

After Gaussian Smoothing

Denoising Approaches

Parametric Approaches

- Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

Denoising Approaches

Parametric Approaches

- Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

Non Parametric Approaches

- Local Smoothing / Local Approximation
- Non Local Methods

Denoising Approaches

Parametric Approaches

- Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

Non Parametric Approaches

- Local Smoothing / Local Approximation
- Non Local Methods

Estimating $y(x)$ from $z(x)$ can be statistically treated as regression of z given x

$$\hat{y}(x) = \mathrm{E}[z \mid x]$$

Denoising Approaches

Spatially adaptive methods, The basic principle:

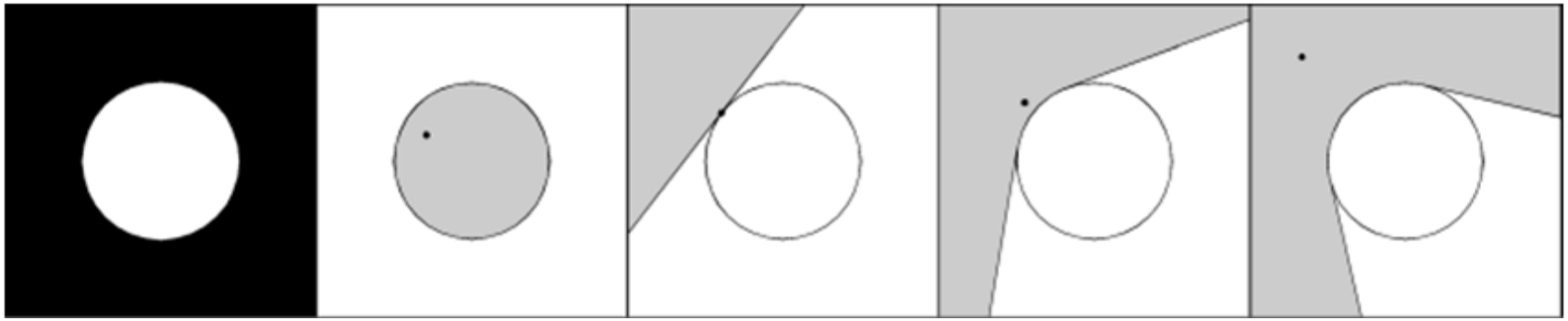
- there are no simple models able to describe the whole image y, thus perform the regression $\hat{y}(x) = \mathrm{E}[z \mid x]$
- Adopt a simple model in small image regions. For instance

$$\forall x \in X, \qquad \exists\, \widetilde{U}_x \ \text{ s.t. } \ y_{|\widetilde{U}_x} \text{ is a polynomial}$$

- Define, in each image pixel, the "**best neighborhood**" where a simple parametric model can be enforced to perform regression.
- For instance, assume that on a suitable pixel-dependent neighborhood, where the image can be described by a polynomial

Ideal neighborhood – an illustrative example

Ideal in the sense that it defines the support of a pointwise Least Square Estimator of the reference point.



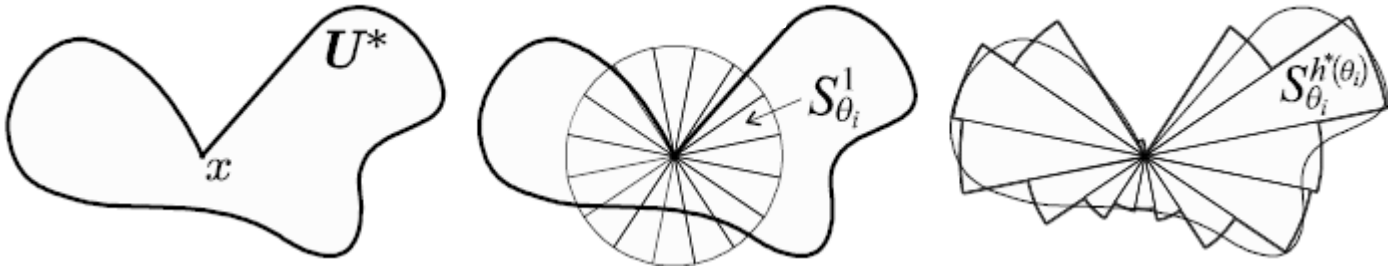
Typically, even in simple images, every point has its own different ideal neighborhood.

For practical reasons, the ideal neighborhood is assumed starshaped

Further details at LASIP c/o Tampere University of Technology
http://www.cs.tut.fi/~lasip/

Neighborhood discretization

A suitable discretization of this neighborhood is obtained by using a set of directional LPA kernels $\{g_{\theta,h}\}_{\theta,h}$



Ideal Neighborhood

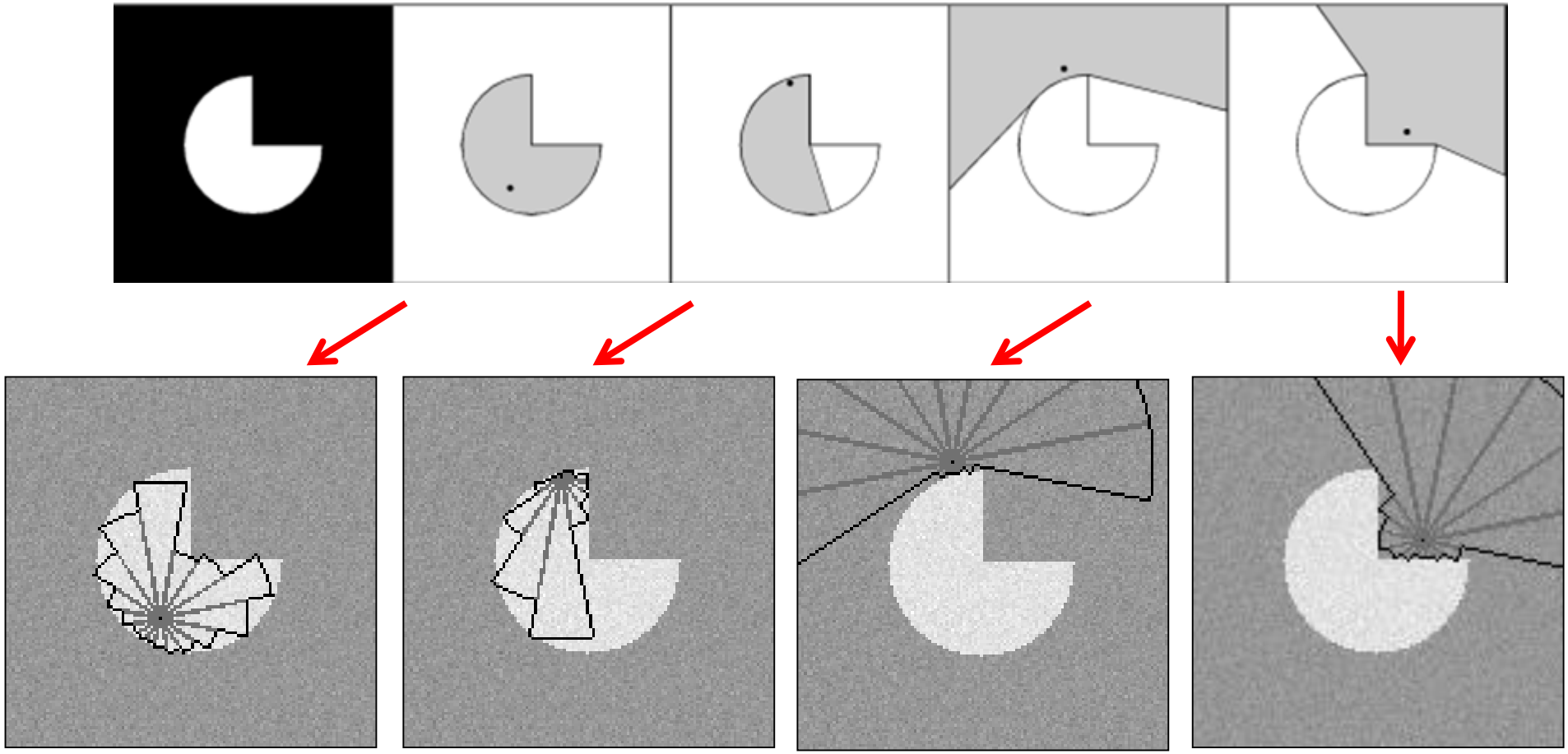
Directional kernels

Discrete Adaptive Neighborhood

where θ determines the orientation of the kernel support, and h controls the scale of kernel support.

Ideal neighborhood – an illustrative example

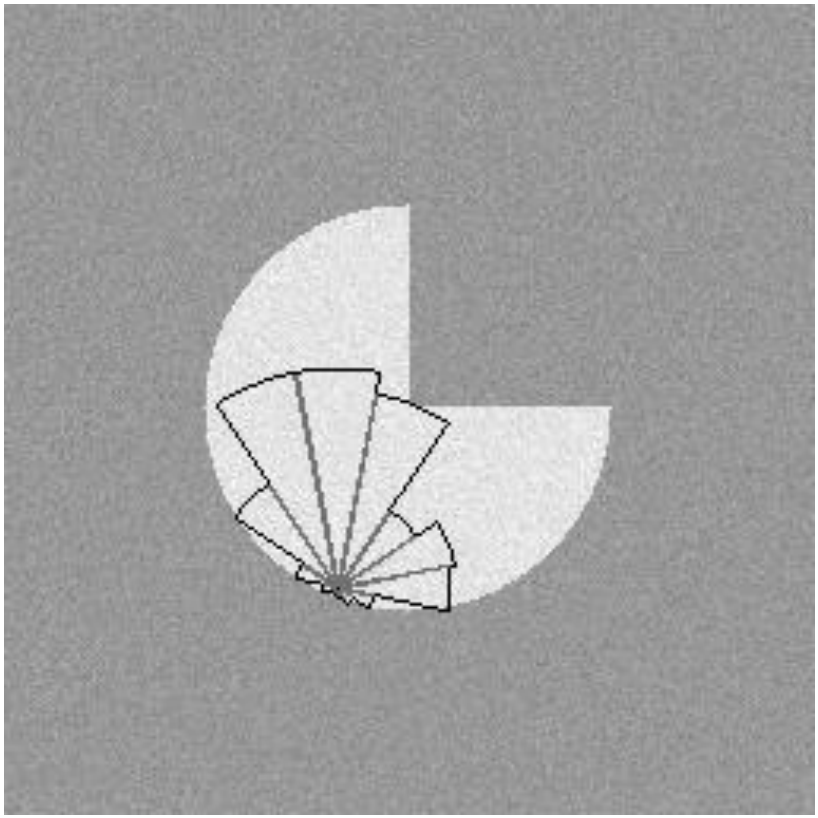
Ideal in the sense that the neighborhood defines the support of pointwise Least Square Estimator of the reference point.



Examples of Adaptively Selected Neighorhoods

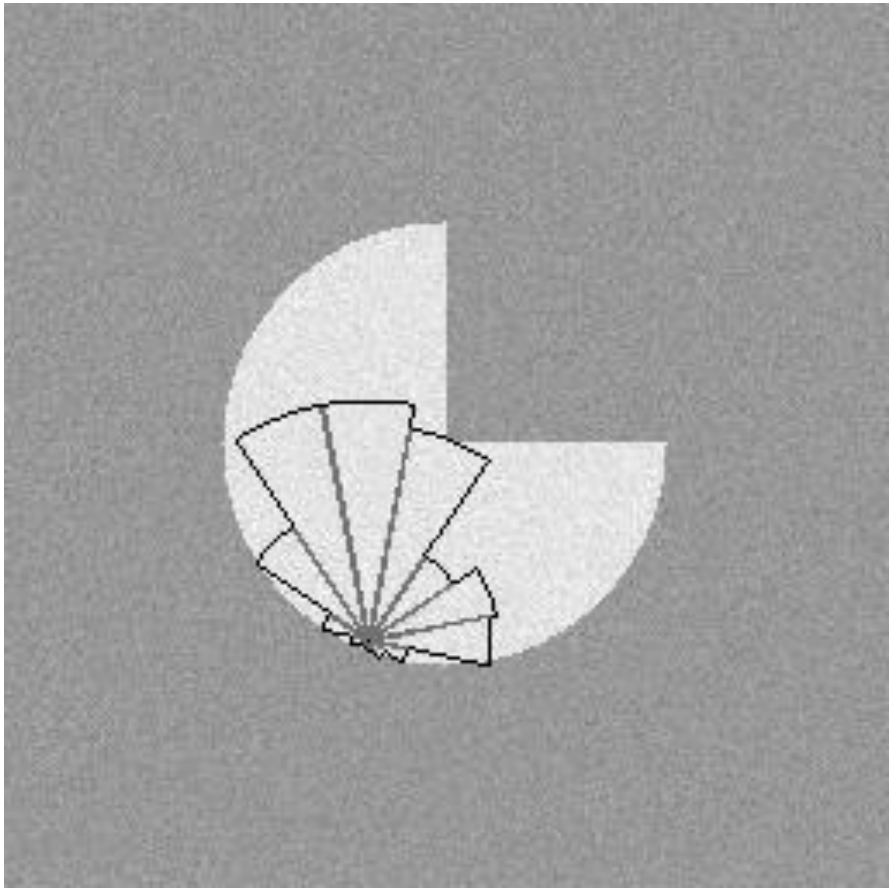
Define, $\forall x \in X$, the "ideal" neighborhood $\widetilde{U}_x$

Compute the denoised estimate at x by "using" only pixels in $\widetilde{U}_x$ and a polynomial model to perofrm regression $\hat{y}(x) = \mathrm{E}[z \,|x, \widetilde{U}_x]$



Examples of adaptively selected neighorhoods

Neighborhoods adaptively selected using the LPA-ICI rule



Example of Performance

Original, noisy, denoised using polynomial regression on adaptively defined neighborhoods (LPA-ICI)

Derivative Estimation

Recall

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \left(\frac{f(x+\epsilon) - f(x_n)}{\epsilon} \right)$$

Now this is linear and shift invariant. Therefore, in discrete domain, it will be represented as a convolution

Differentiation and convolution

Recall

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \left(\frac{f(x+\epsilon) - f(x_n)}{\epsilon} \right)$$

Now this is linear and shift invariant. Therefore, in discrete domain, it will be represented as a convolution

We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x_n)}{\Delta x}$$

which is obviously a convolution against the Kernel [1 -1];

Finite Difference in 2D

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y) - f(x,y)}{\varepsilon}\right)$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon) - f(x,y)}{\varepsilon}\right)$$

Horizontal

$$\begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1},y_m) - f(x_n,y_m)}{\Delta x}$$

$$\frac{\partial f(x,y)}{\partial y} \approx \frac{f(x_n,y_{m+1}) - f(x_n,y_m)}{\Delta x}$$

Vertical

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Discrete Approximation

Convolution Kernels

Differentiating Filters

The derivatives can be also computed using centered filters:

$$f_x(x) = f(x-1) - f(x+1)$$

Such that the horizontal derivative is:

$$f_x = f \otimes \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}$$

While the vertical derivative is:

$$f_y = f \otimes \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}^{t}$$

Horizontal derivative

$$\longrightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad h = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth Differentiate

Vertical derivative

$$\longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad h' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Classical Operators: Sobel

Horizontal derivative

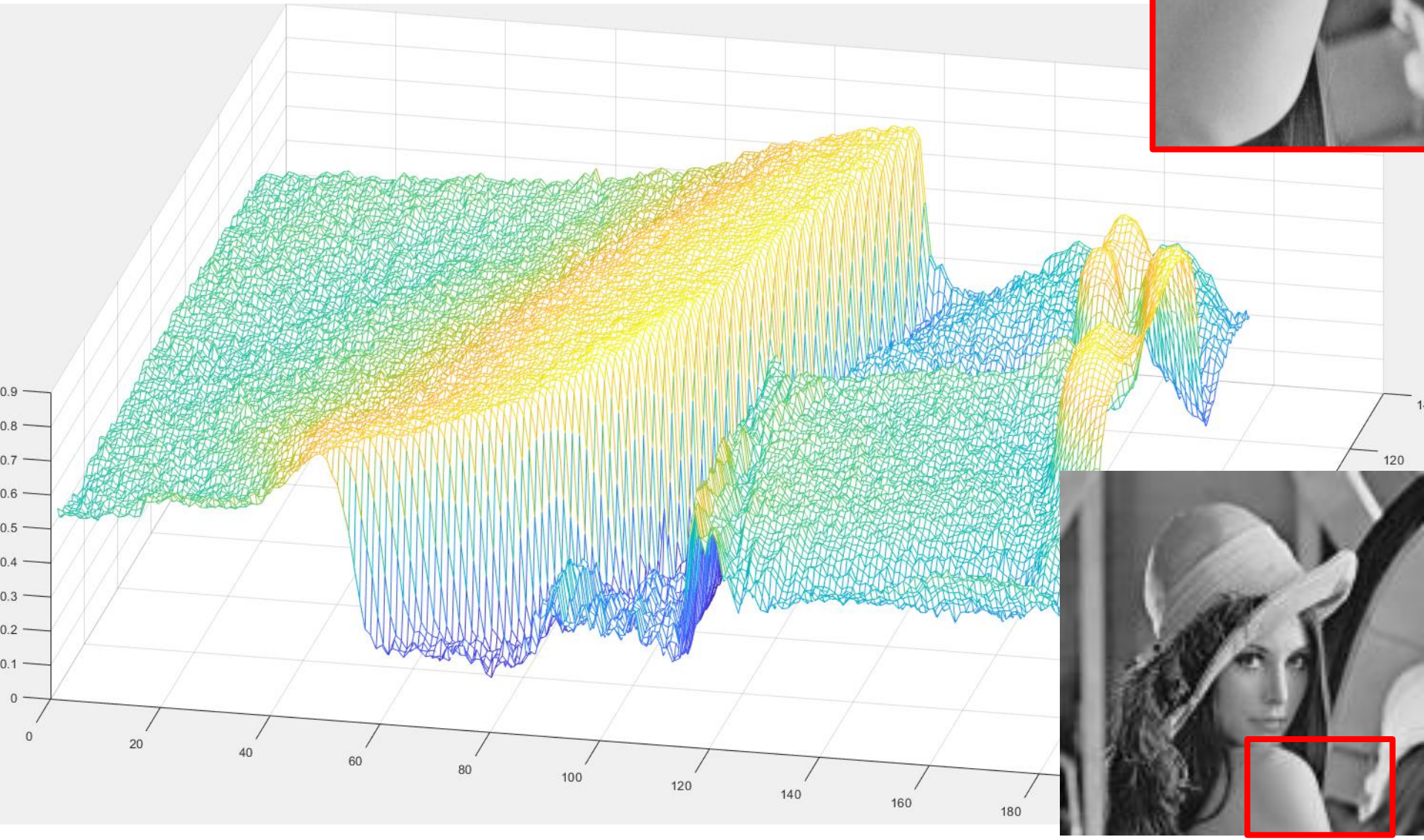
$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad h = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth Differentiate

Vertical derivative

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad h' = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Think of an image as a 2d, real-valued function



The Image Gradient

Image Gradient can be considered as the gradient of a real-valued 2D function

$$\nabla I(r,c) = \begin{bmatrix} I \circledast d_x \\ I \circledast d_y \end{bmatrix}(r,c)$$

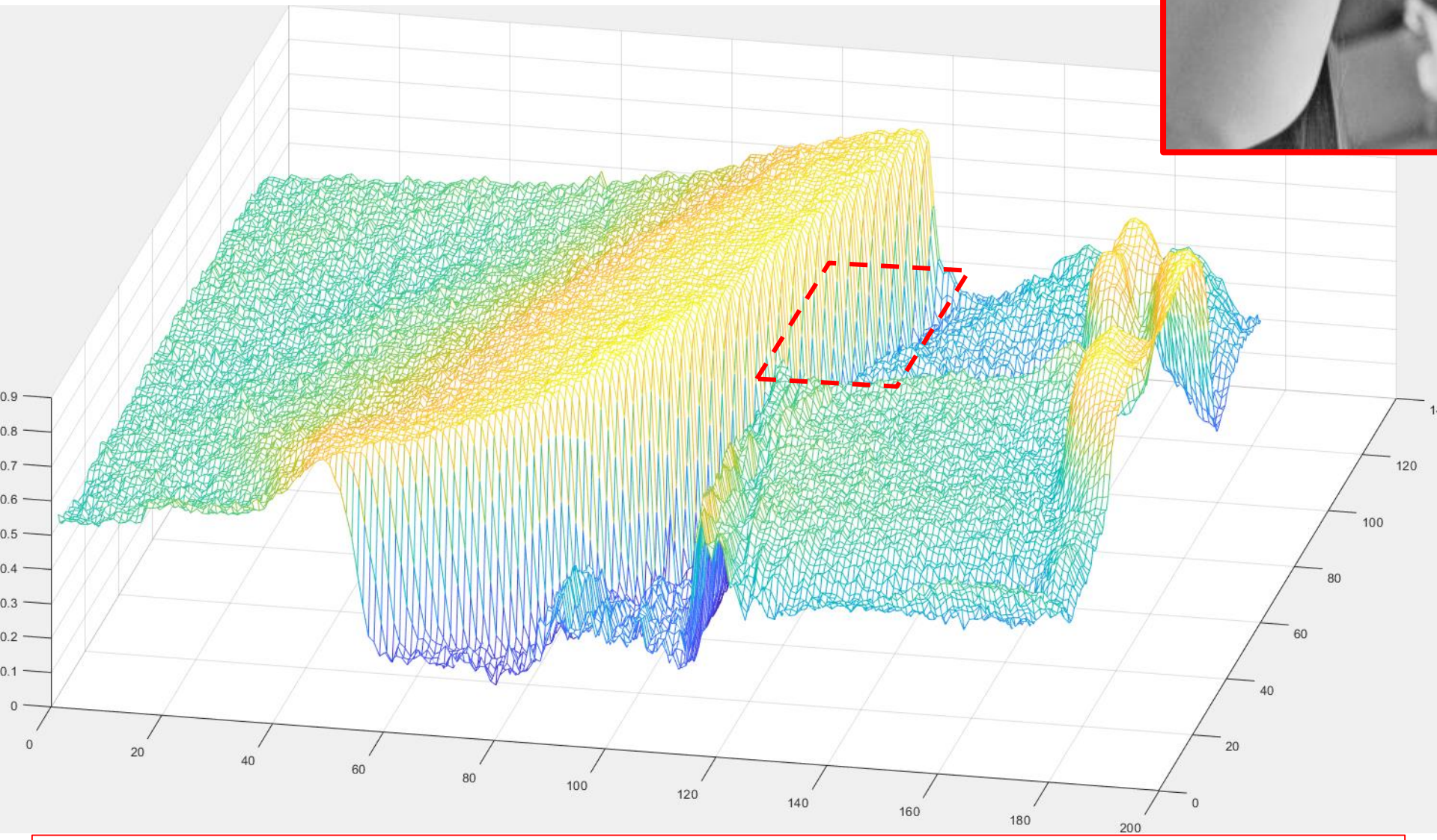
where principal derivatives are computed through convolution against the derivative filters (e.g. Prewitt)

$$dx = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \qquad dy = dx'$$

Image gradient behaves likethe gradient of a function:

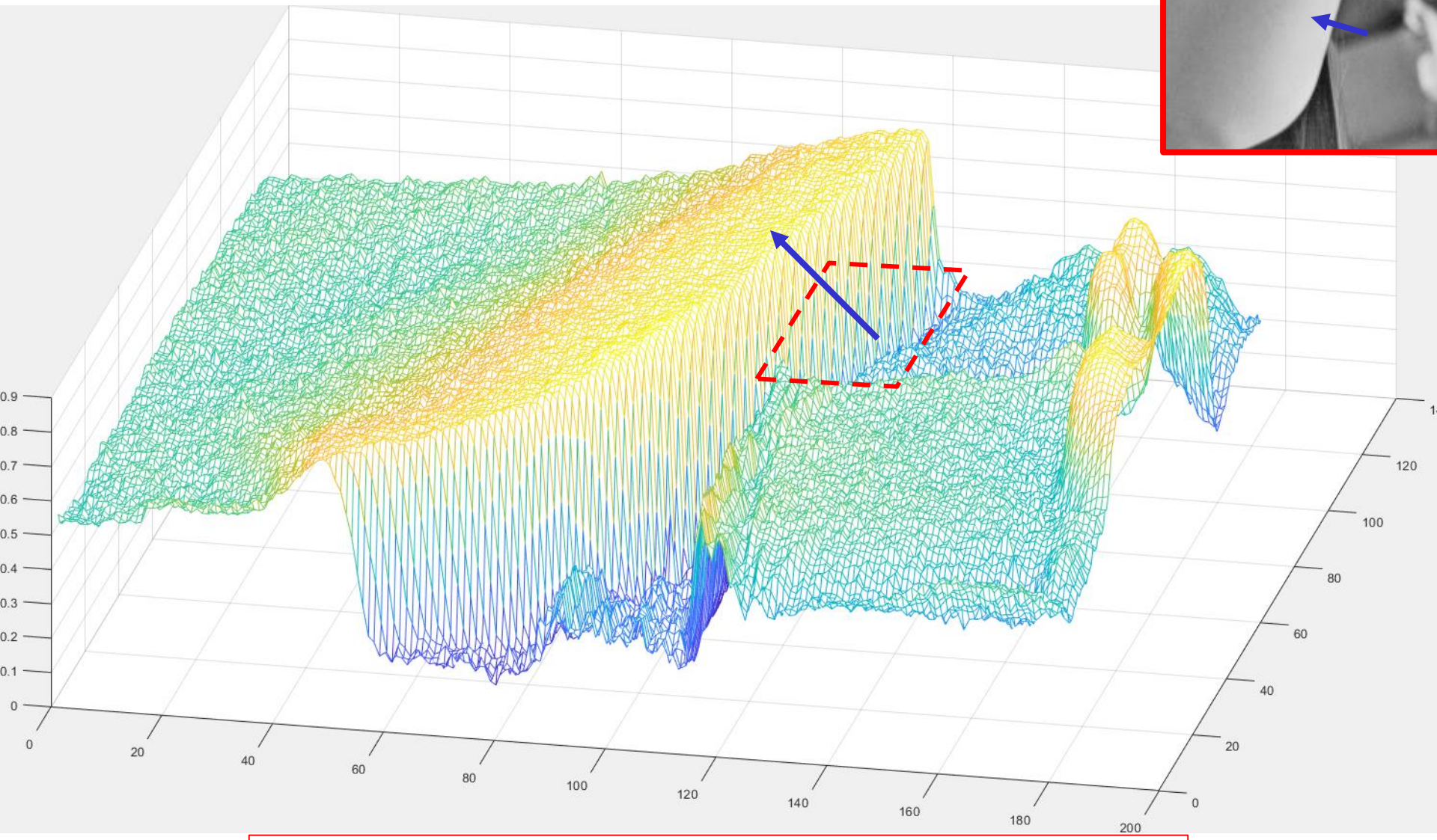
- $|\nabla I(r,c)|$ is large where there are large variations
- $\angle \nabla I(r,c)$ is the direction of the steepest variation

Think of an image as a 2d, real-valued function



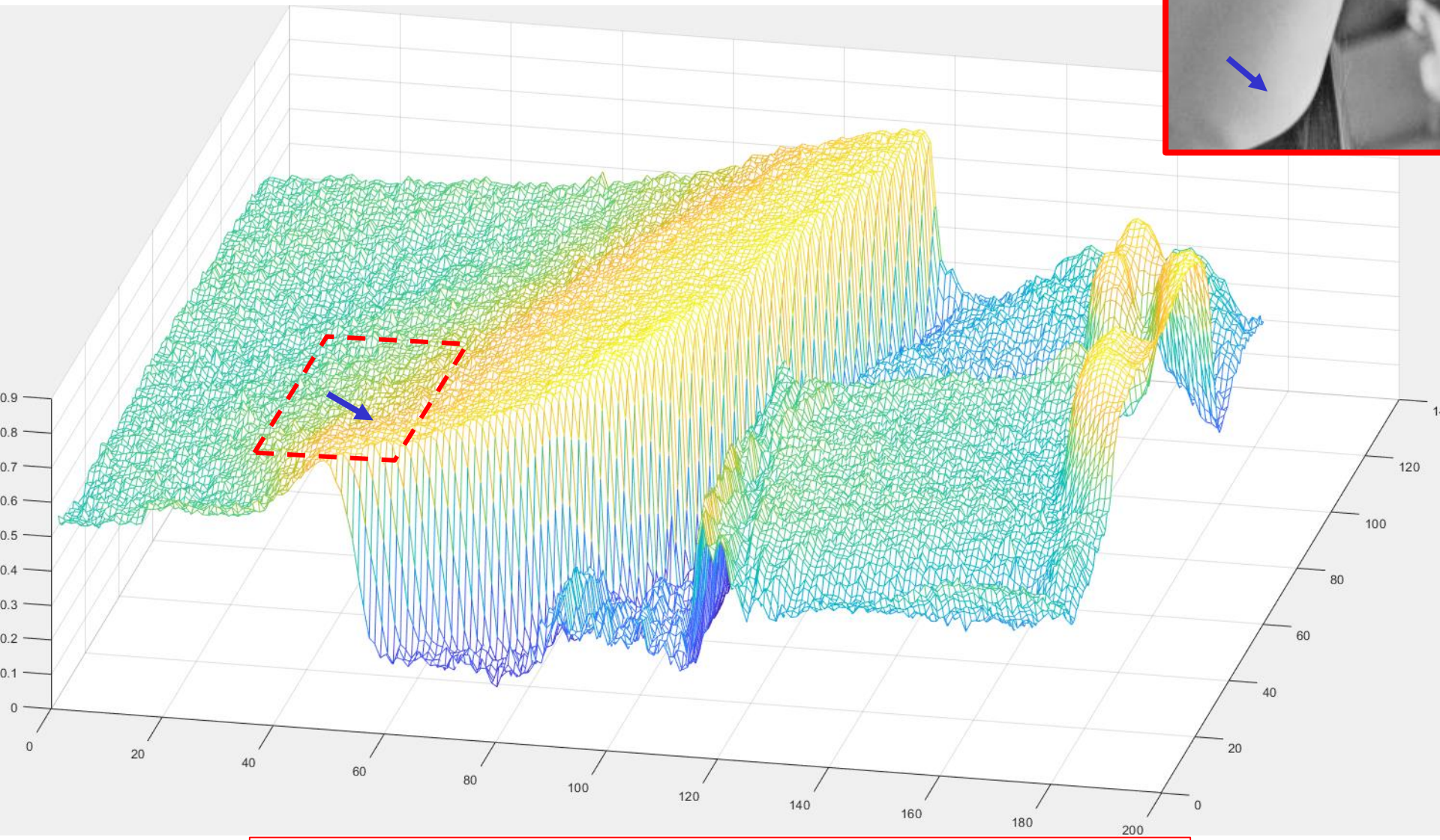
Local spatial transformations are defined over neighborhood like this

Think of an image as a 2d, real-valued function



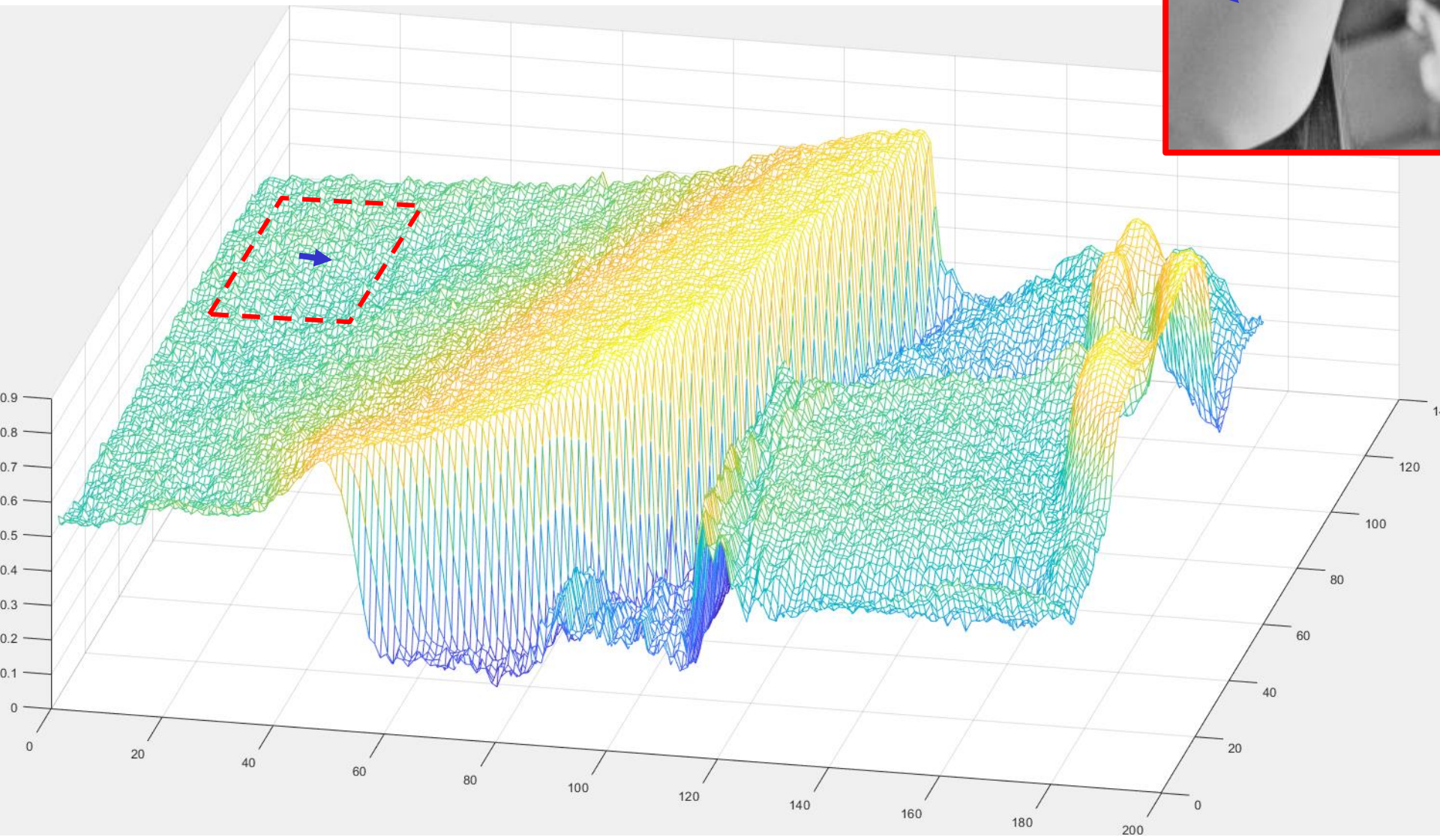
What about the gradient in this neighborhood?

Think of an image as a 2d, real-valued function



What about the gradient in this neighborhood?

Think of an image as a 2d, real-valued function

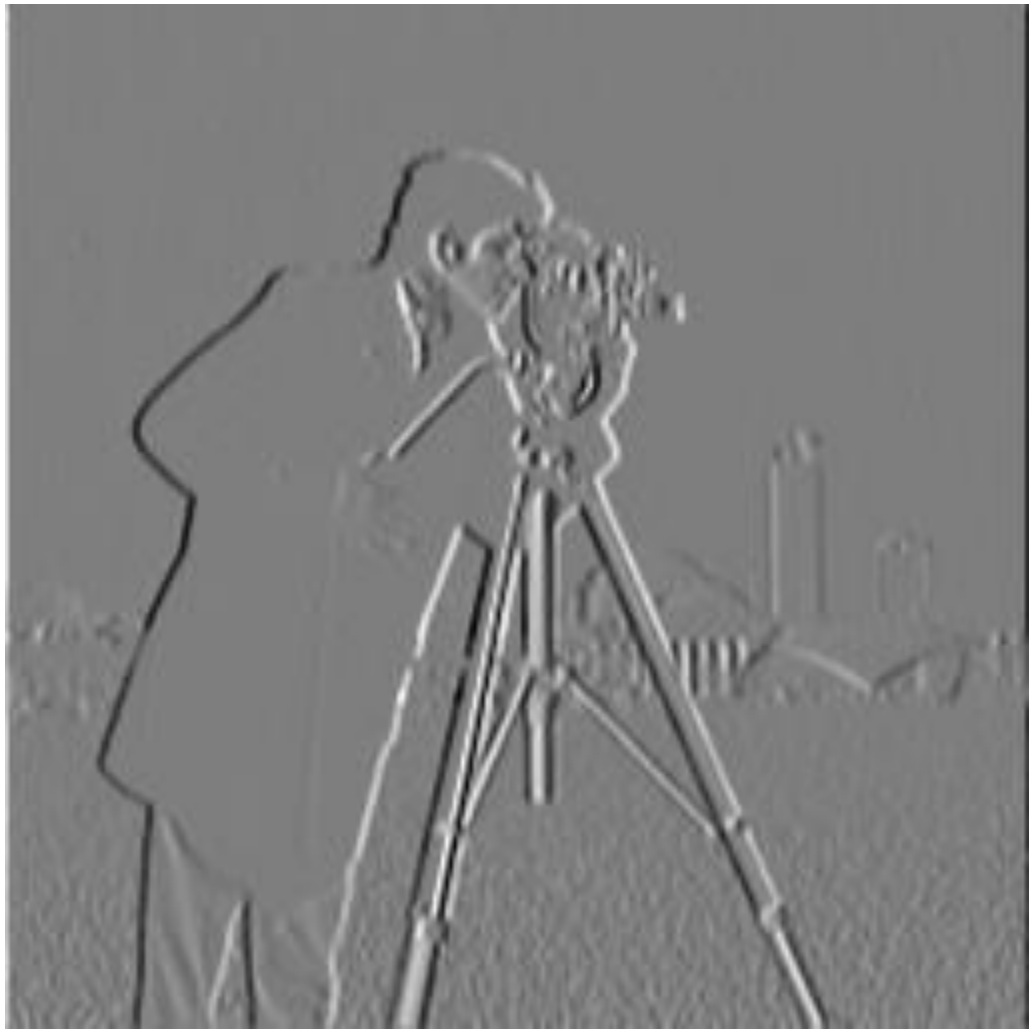


Another famous test image - cameraman

Horizontal Derivatives using Sobel

$$\nabla y_1 = (y \otimes h)$$

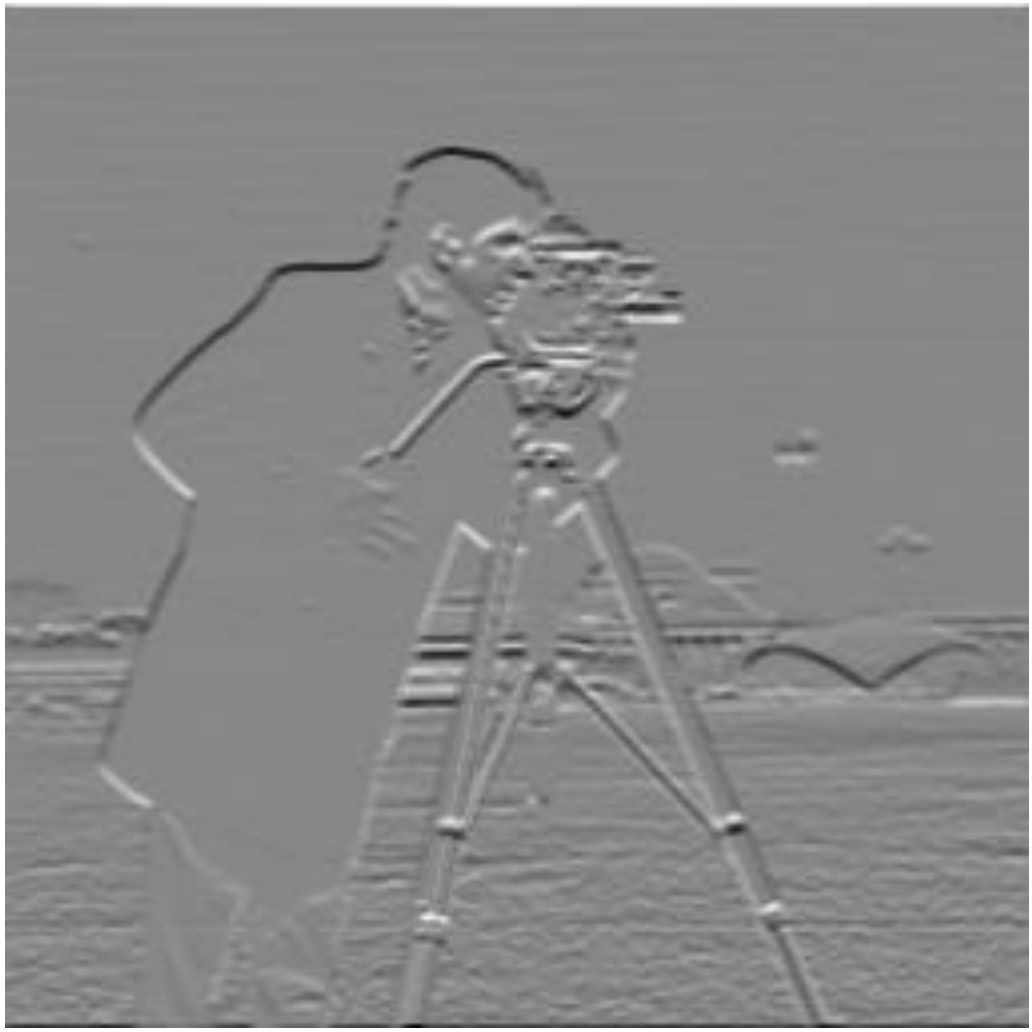
$$\boldsymbol{\nabla y}(r,c) = \begin{bmatrix} \nabla y_1(r,c) \\ \nabla y_2(r,c) \end{bmatrix}$$



Vertical Derivatives using Sobel

$$\nabla y_2 = (y \otimes h')$$

$$\boldsymbol{\nabla y}(r,c) = \begin{bmatrix} \nabla y_1(r,c) \\ \nabla y_2(r,c) \end{bmatrix}$$



Gradient Magnitude

$$\|\nabla y\| = \sqrt{(y \otimes h)^2 + (y \otimes h')^2}$$

$$\boldsymbol{\nabla y}(r,c) = \begin{bmatrix} \nabla y_1(r,c) \\ \nabla y_2(r,c) \end{bmatrix}$$

The Gradient Orientation

Like for continuous function, the gradient in each pixel points at the **steepest growth/decrease direction.**

$$\angle \nabla y(r,c) = \text{atand}\left(\frac{\nabla y_2(r,c)}{\nabla y_1(r,c)}\right) = \text{atand}\left(\frac{(y \otimes h)(r,c)}{(y \otimes h')(r,c)}\right)$$

The gradient norm indicates the strength of the intensity variation

Let's switch to Matlab…..

Normalized Cross Correlation

A very straightforward approach to template matching

Normalized Cross Correlation

Normalized Cross Correlation is defined as

$$NCC(A,B) = \frac{N(A,B)}{\sqrt{N(A,A)N(B,B)}}$$

where

$$N(A,B) = \iint_W (A(x,y) - \bar{A})(B(x,y) - \bar{B})\ dx\ dy$$

and $\bar{A}$ represents the average image value on patch A, similarly $\bar{B}$. W is the support of A or B.

Normalized Cross Correlation

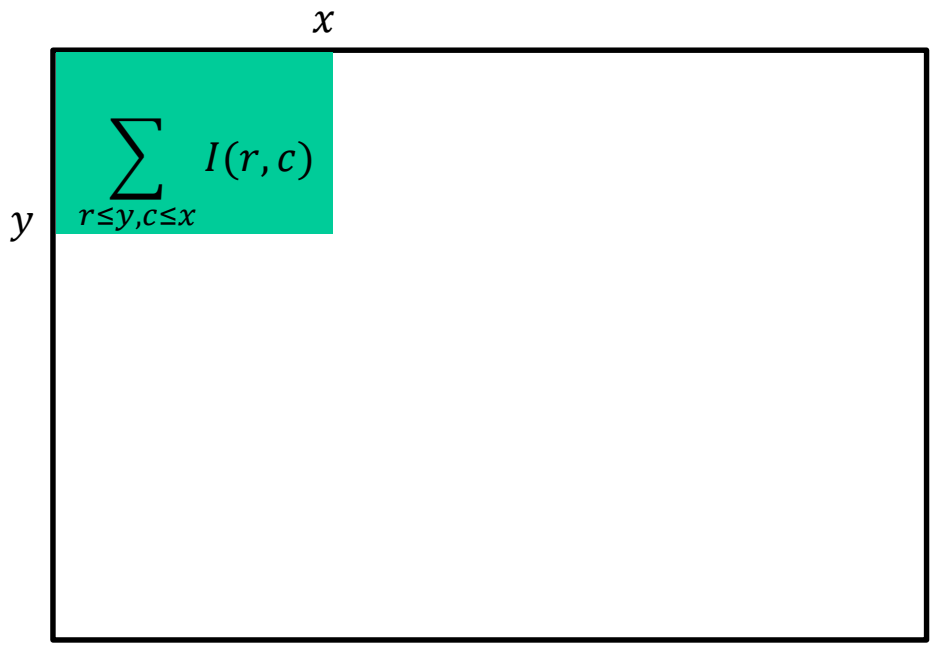
Remarks:

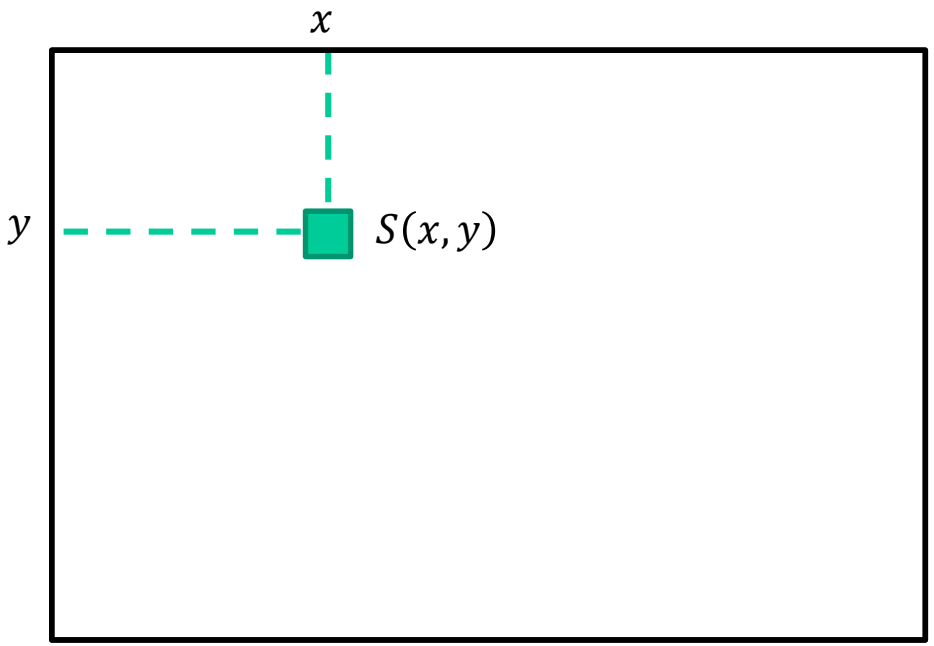
- NCC yields a measure in **the range [-1,1]** , while the SSD is only positive and not normalized (its maximum value depends on the image range).
- NCC is **invariant** to changes in the average intensity.
- While this seems quite computationally demanding, there exists fast implementations where local averages are computed by **running sums** (integral image)

Integral Image

The integral image S is defined from an image I as follows

$$S(x,y) = \sum_{r \le y, c \le x} I(r,c)$$

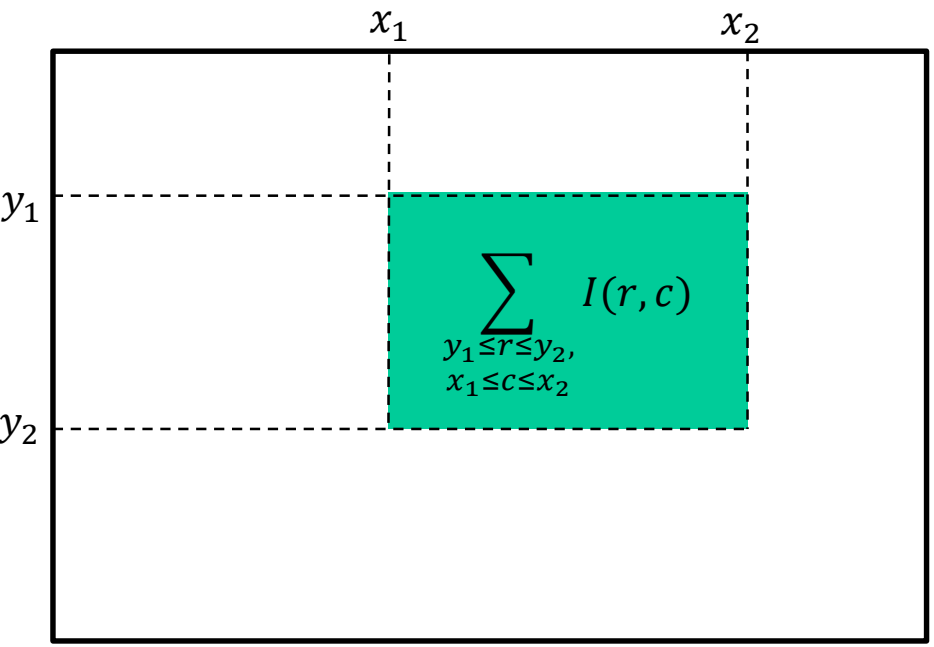


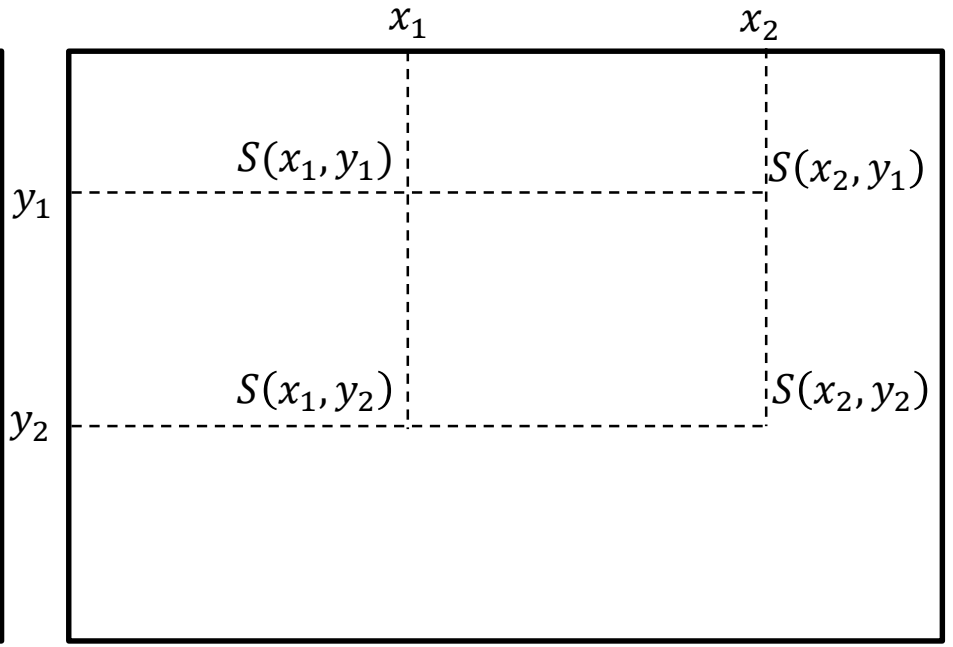


Using the Integral Image

The integral image allows fast computation of the sum (average) of any rectangular region in the image

$$\sum_{\substack{y_1 \le r \le y_2, \\ x_1 \le c \le x_2}} I(r,c) = S(x_2,y_2) - S(x_2,y_1) - S(x_1,y_2) + S(x_1,y_1)$$





Nonlinear Filters

Non Linear Filters

Non Linear Filters are such that

$$H[\lambda f(t) + \mu g(t)] = \lambda H[f](t) + \mu H[g](t)$$

does not hold, at least for some value of λ, μ, f, g

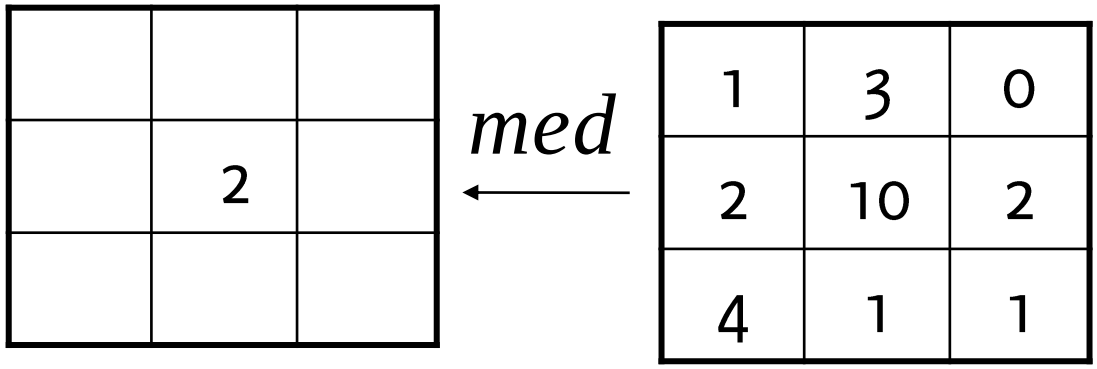
Examples of nonlinear filter are

- Median Filter (Weighted Median)
- Ordered Statistics based Filters
- Threshold, Shrinkage

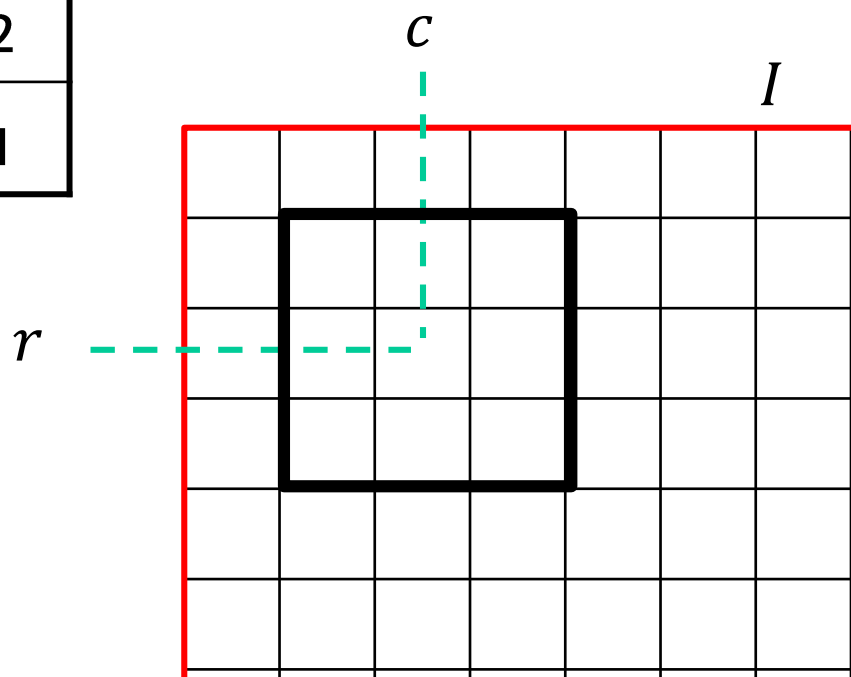
There are many others, such as data adaptive filtering procedures (e.g LPA-ICI)

Blockwise Median

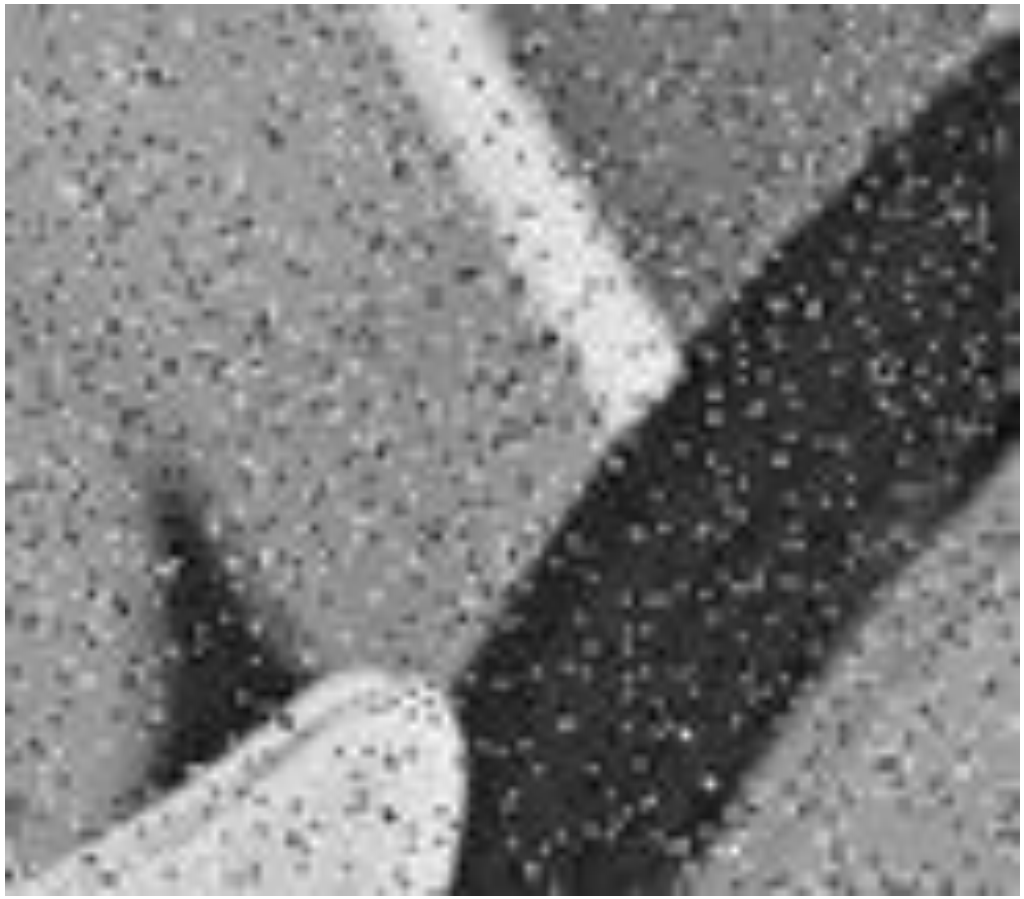
Block-wise median: replaces each pixel with the median of its neighborhood



$$m = median(1,3,0,2,10,2,4,1,1) = 2$$



Salt-and-pepper noise



Salt and Pepper (Impulsive) noise

Denoisng using local smoothing 3x3

Denoisng with median 3x3

Salt and Pepper (Impulsive) noise

An overview on morphological operations

Erosion, Dilation

Open, Closure

We assume the image being processed is binary, as these operators are typically meant for refining “mask” images.