

Digital Image Filters

Giacomo Boracchi

CVPR USI, March 24 2020

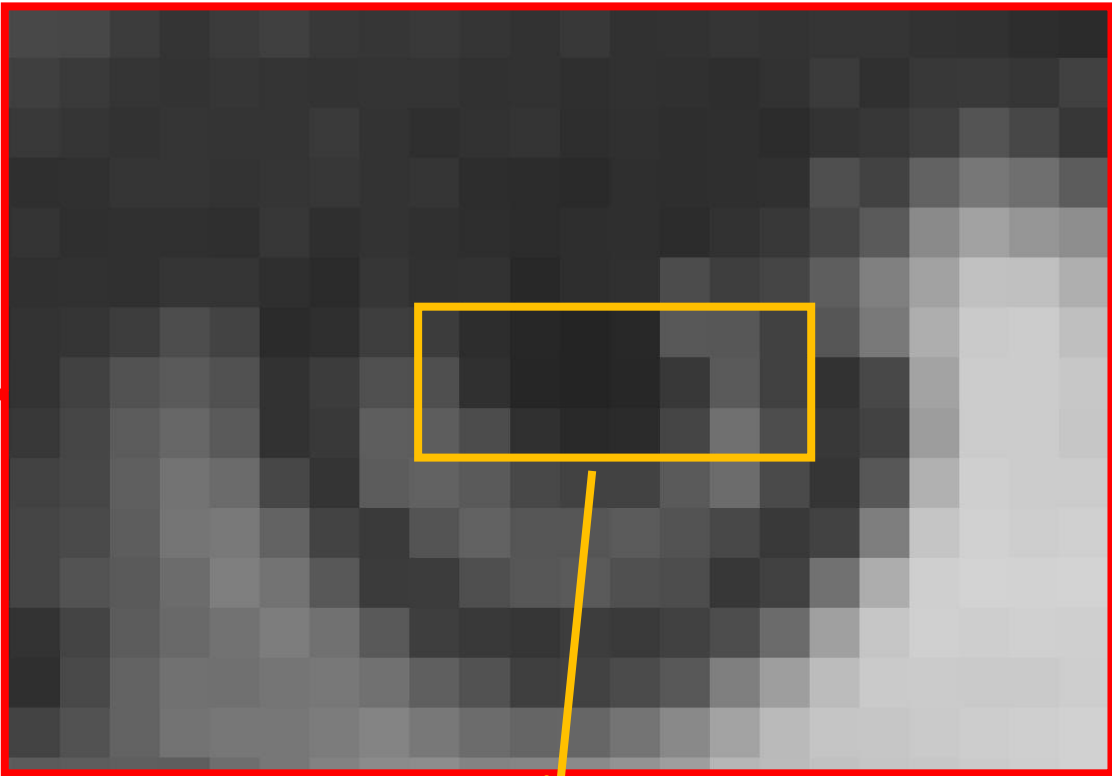
Book: GW, chapter 3

Images

Images: the Classifier's input

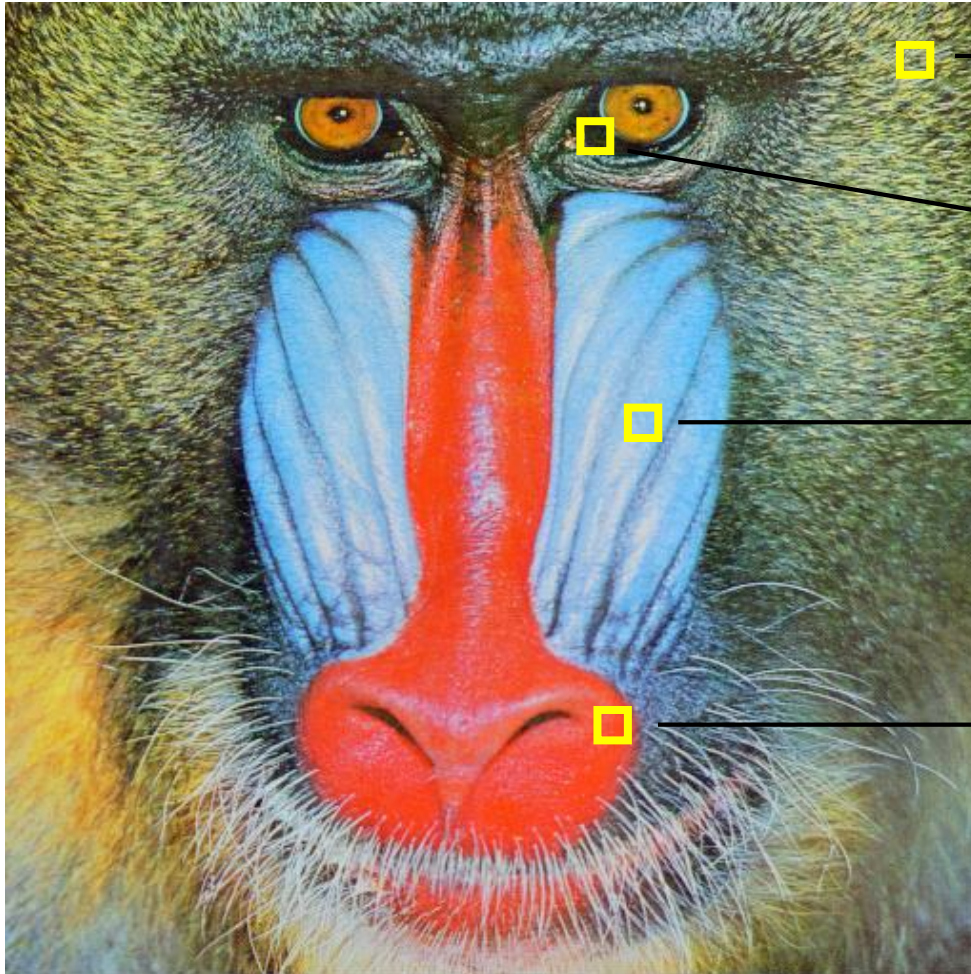


$$I \in \mathbb{R}^{R \times C}$$



123	122	134	121	132	133	145	134
122	121	125	132	124	121	116	126
119	127	137	119	139	127	128	131

RGB images



Green region

$$I(r, c) = [120, \mathbf{150}, 30]$$

Dark region

$$I(r, c) = [40, 30, 11]$$

Blue region

$$I(r, c) = [100, 190, \mathbf{240}]$$

Red region

$$I(r, c) = [\mathbf{240}, 80, 70]$$

In practice, intensity values integers [0 – 255]

Python Example

```
from skimage.io import imread
```

```
# Read the image
```

```
I = imread('baboon.jpg')
```

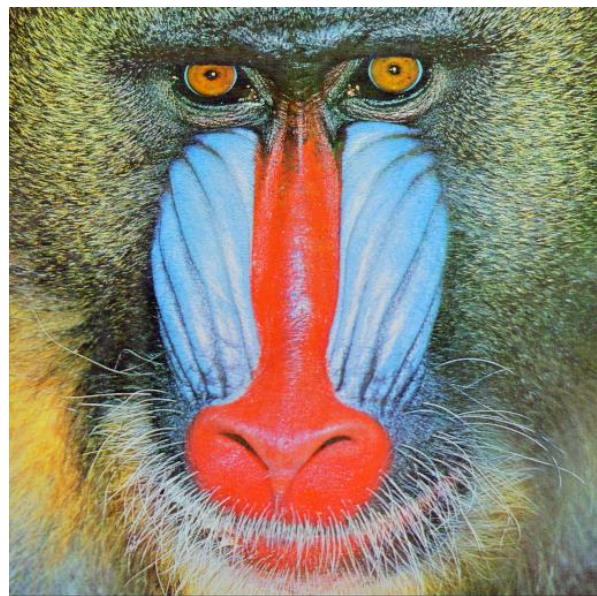
```
# Extract the color channels of the image
```

```
R = I[:, :, 0]
```

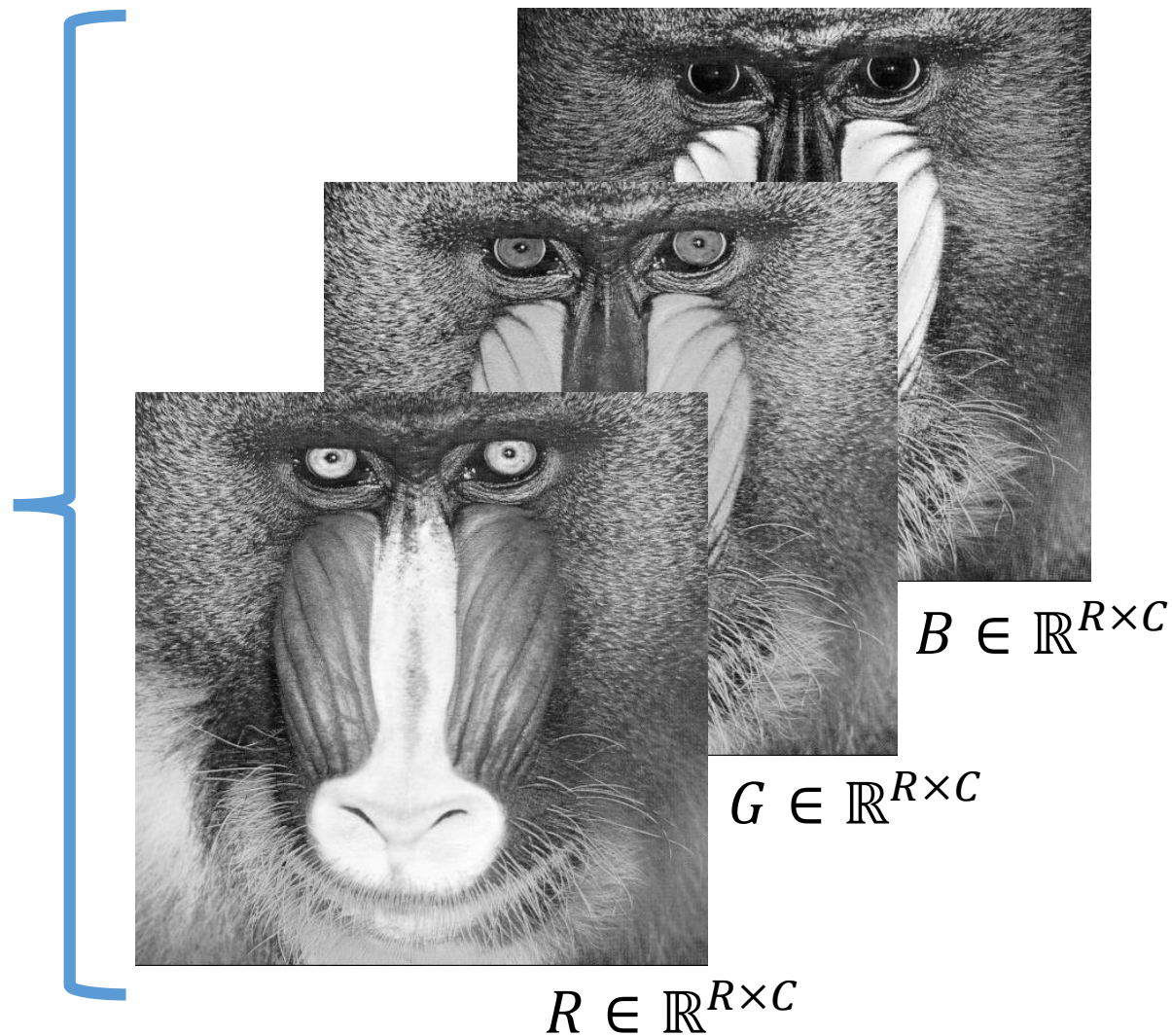
```
G = I[:, :, 1]
```

```
B = I[:, :, 2]
```

RGB images



$$I \in \mathbb{R}^{R \times C \times 3}$$



$$B \in \mathbb{R}^{R \times C}$$

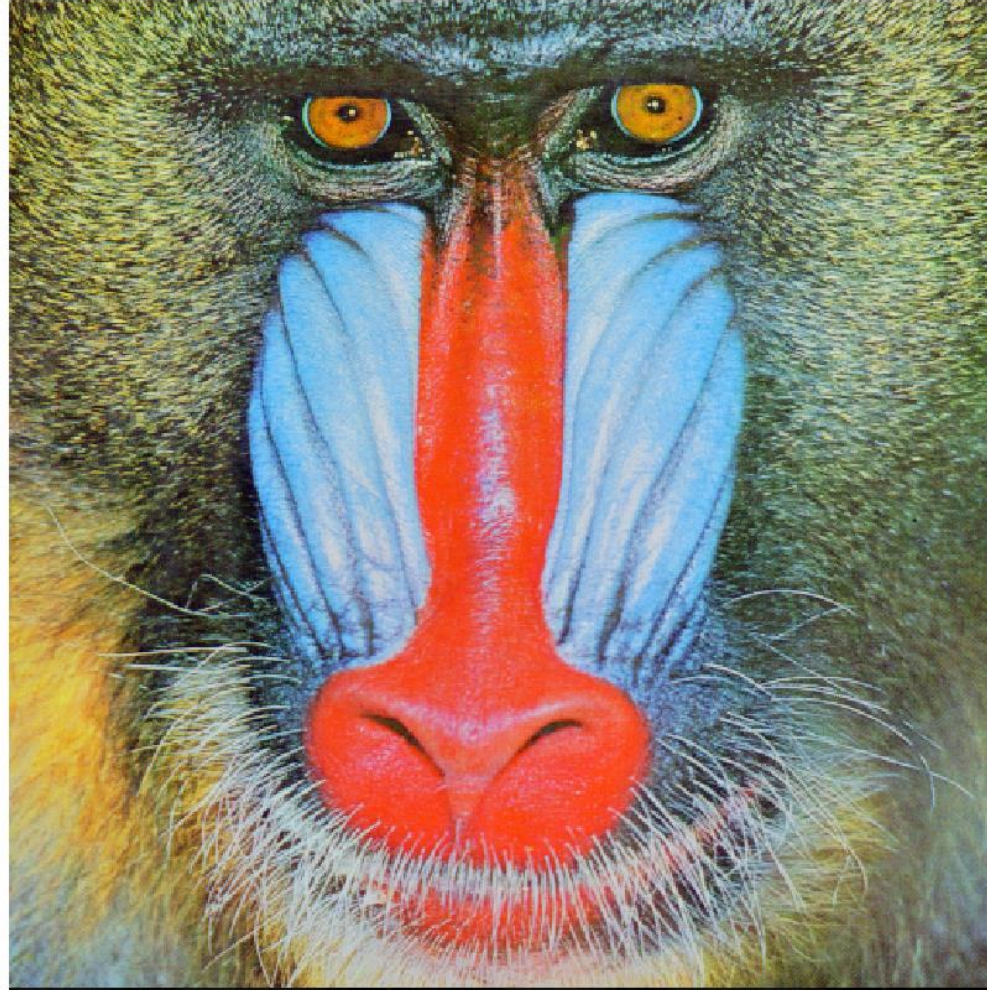
$$G \in \mathbb{R}^{R \times C}$$

$$R \in \mathbb{R}^{R \times C}$$

This image is 512 x 512 pixels: $I \in \mathbb{R}^{512 \times 512 \times 3}$

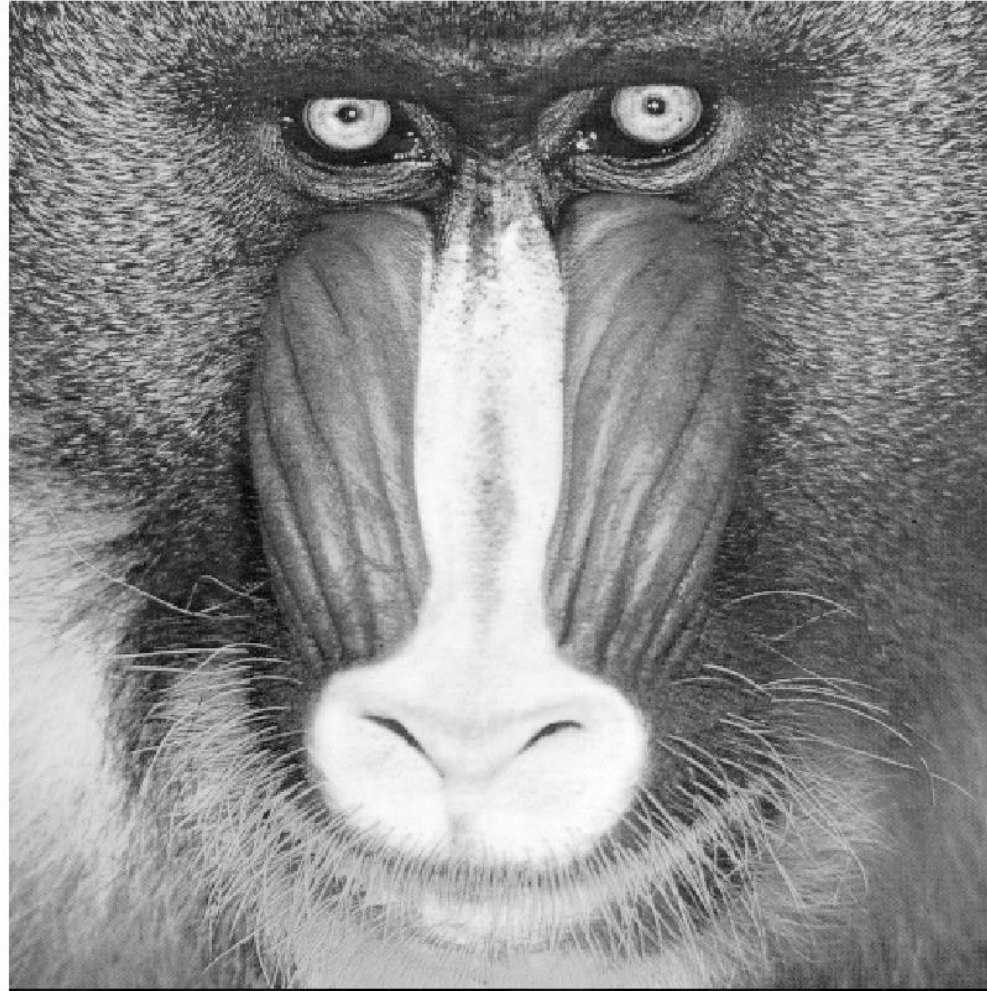
RGB image

RGB image



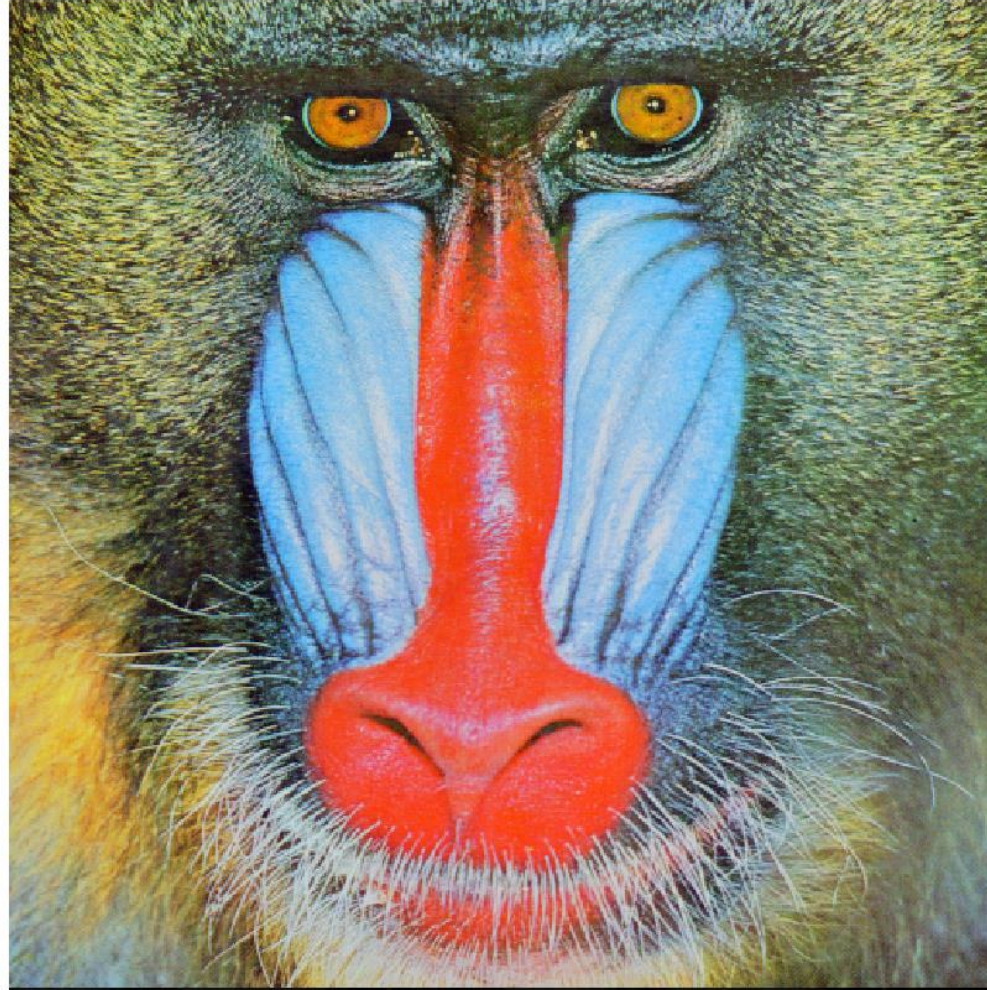
Red Channel

red channel



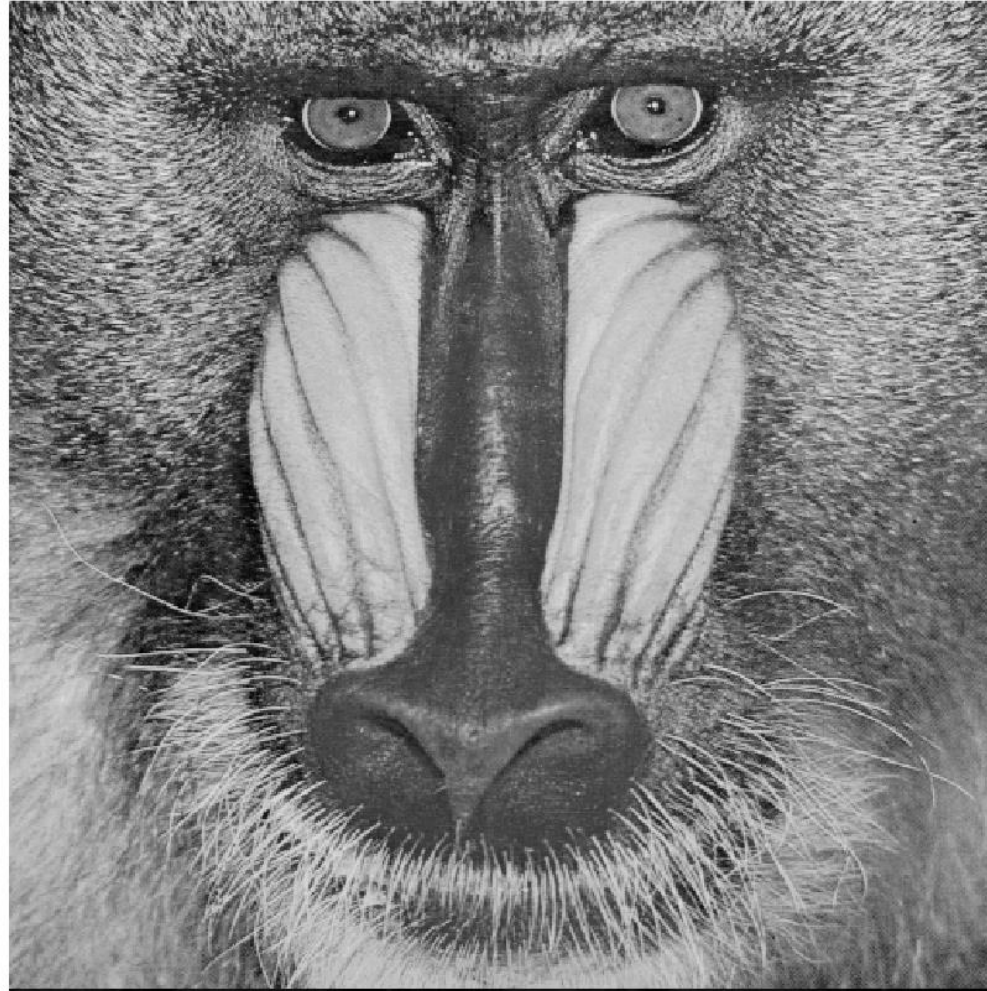
RGB image

RGB image



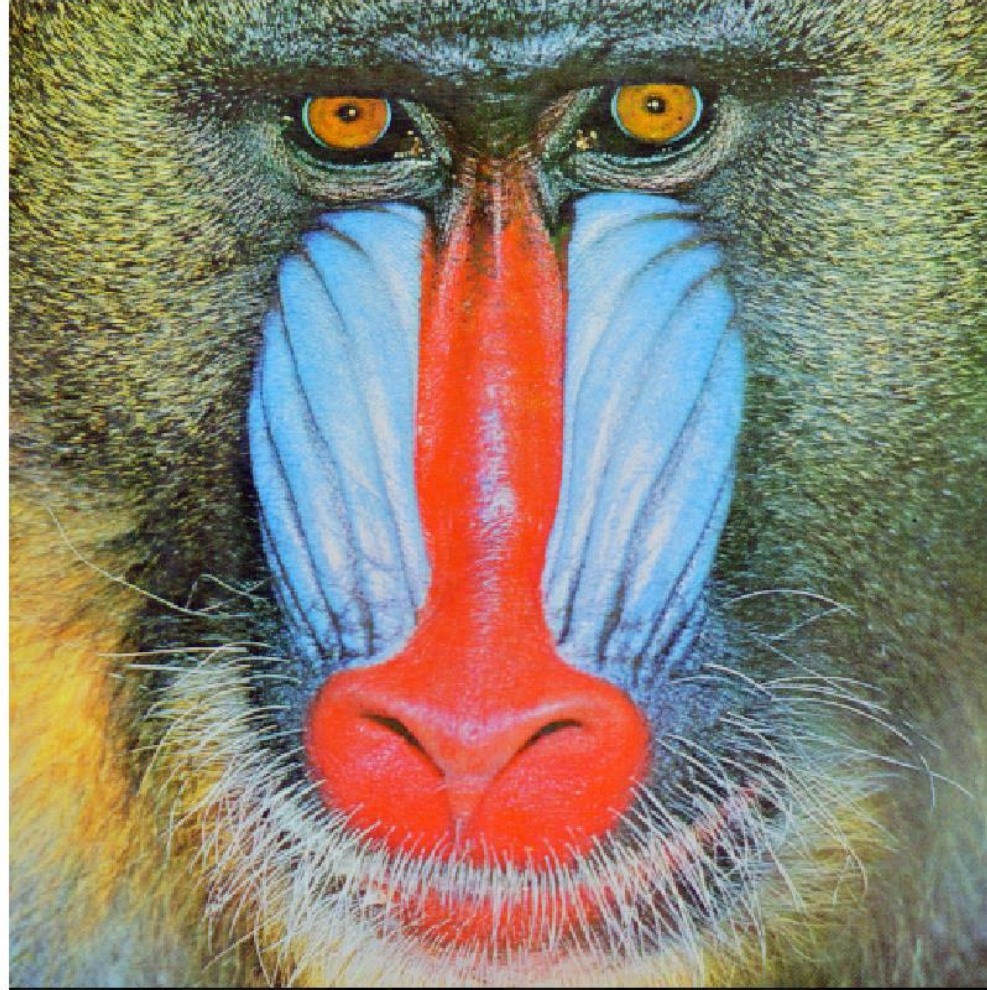
Green Channel

green channel



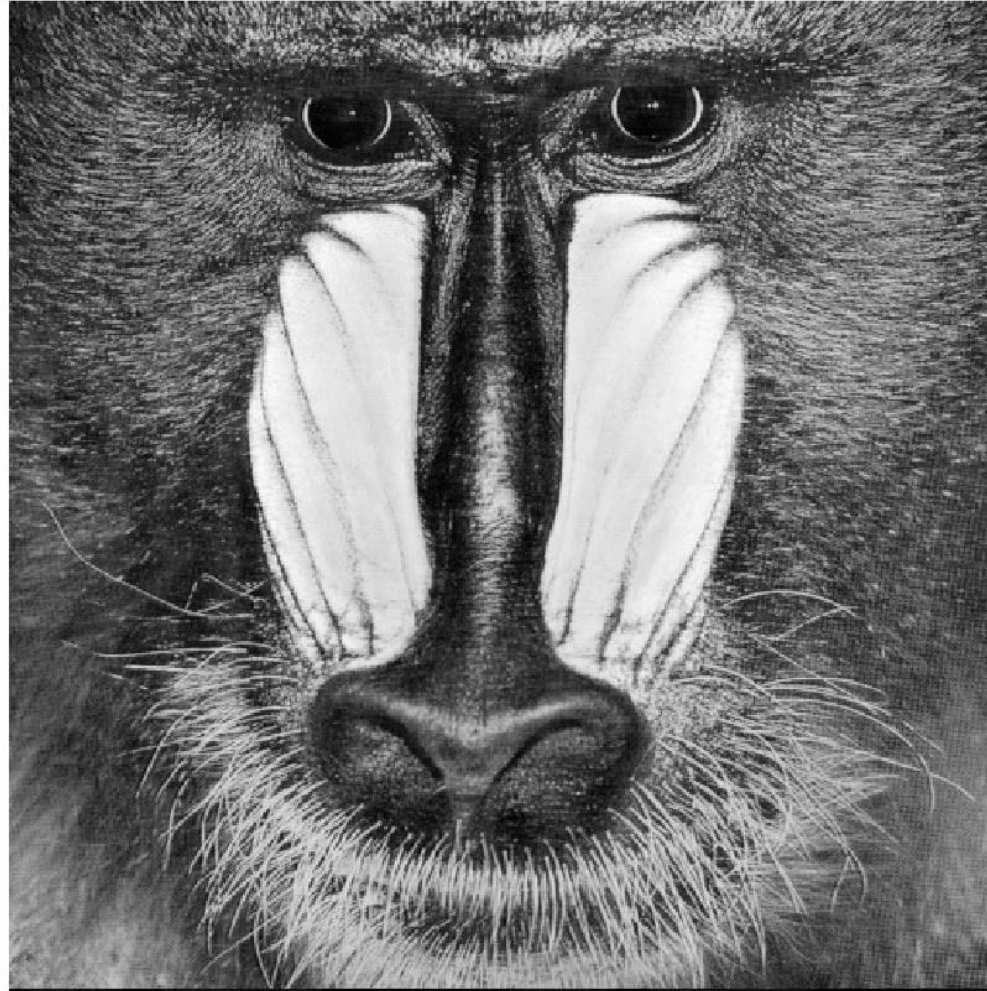
RGB image

RGB image



Blue Channel

blue channel



Videos

Higher dimensional images

Videos are sequences of images (frames)

If a frame

$$I \in [0, 255]^{R \times C \times 3}$$

a video of T frames

$$V \in [0, 255]^{R \times C \times 3 \times T}$$

```
print(V.shape)
(144, 180, 3, 30)
```



In this example: $R = 144$, $C = 180$, thus these 5 color frames contains:
388.800 values in $[0,255]$, thus in principle, 388 KB

Dimension Increases very quickly

Without compression: 1Byte per pixel per channel

1 frame in full HD: $R = 1080, C = 1920 \approx 6MB$

1 sec in full HD (24fps) $\approx 150MB$

Fortunately, visual data are very redundant, thus compressible

This has to be taken into account when you design a Machine learning algorithm to be used on images

Higher dimensional images

Multispectral imaging and remote sensing


Higher dimensional Images

These images are stacking multiple layers as color-planes

Multi-spectral or Hyper-spectral images:

each band covers a certain wavelength that is meaningful for interpretation soil in areal images

- 0.45-0.52 μm Blue-Green
- 0.52-0.60 μm Green
- 0.63-0.69 μm Red
- 0.76-0.90 μm Near IR
- 1.55-1.75 μm Mid-IR
- 10.40-12.50 μm Thermal IR
- 2.08-2.35 μm Mid-IR



Classification of multispectral images:
infer the soil coverage from the values in the different spectral bands

In hyperspectral images the number of bands becomes larger and you get a whole energy spectrum in each pixel

Band 1



Band 2



Band 3



Band 4



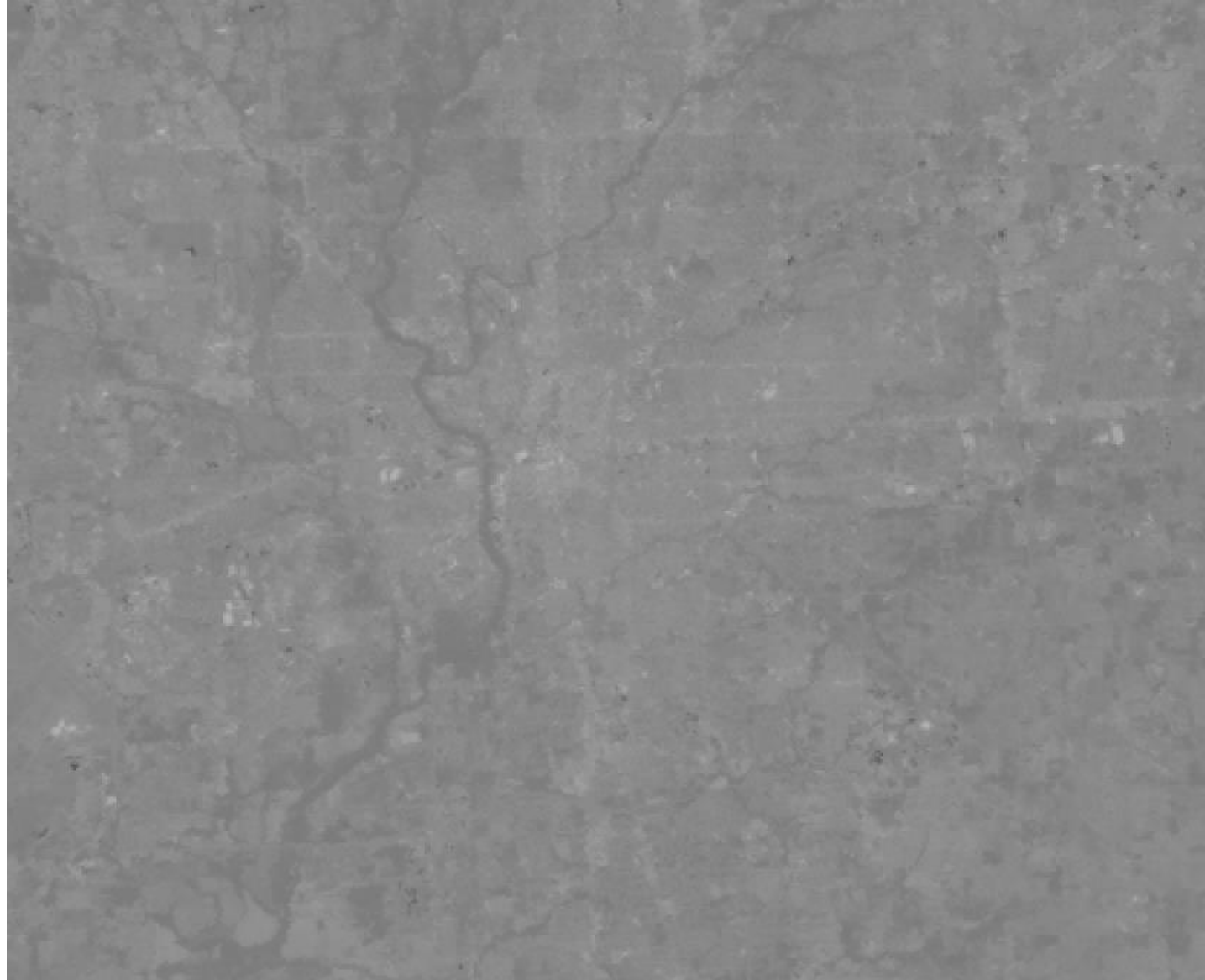
Band 5



Band 6



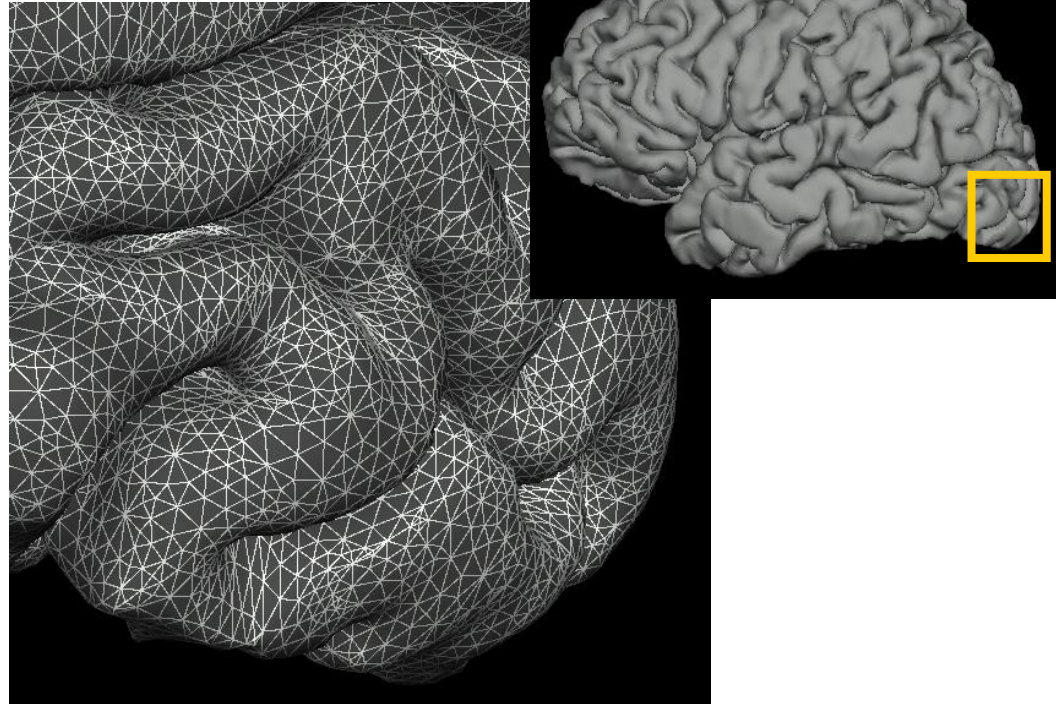
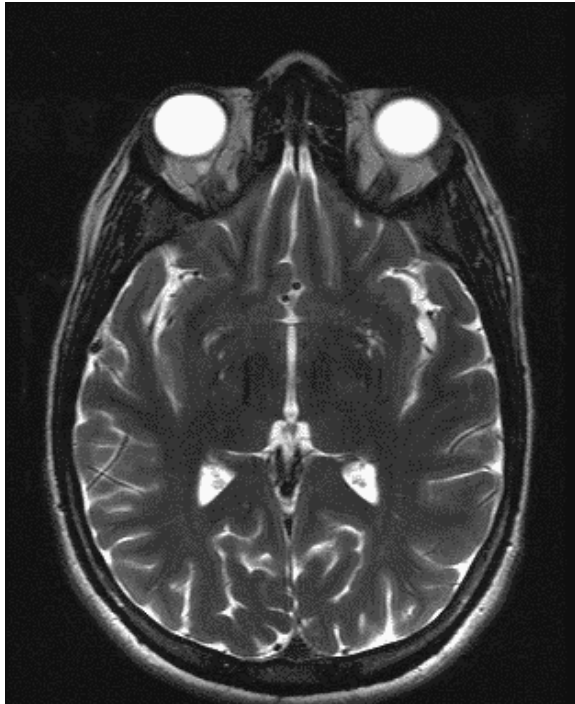
Band 7



MRI and TAC

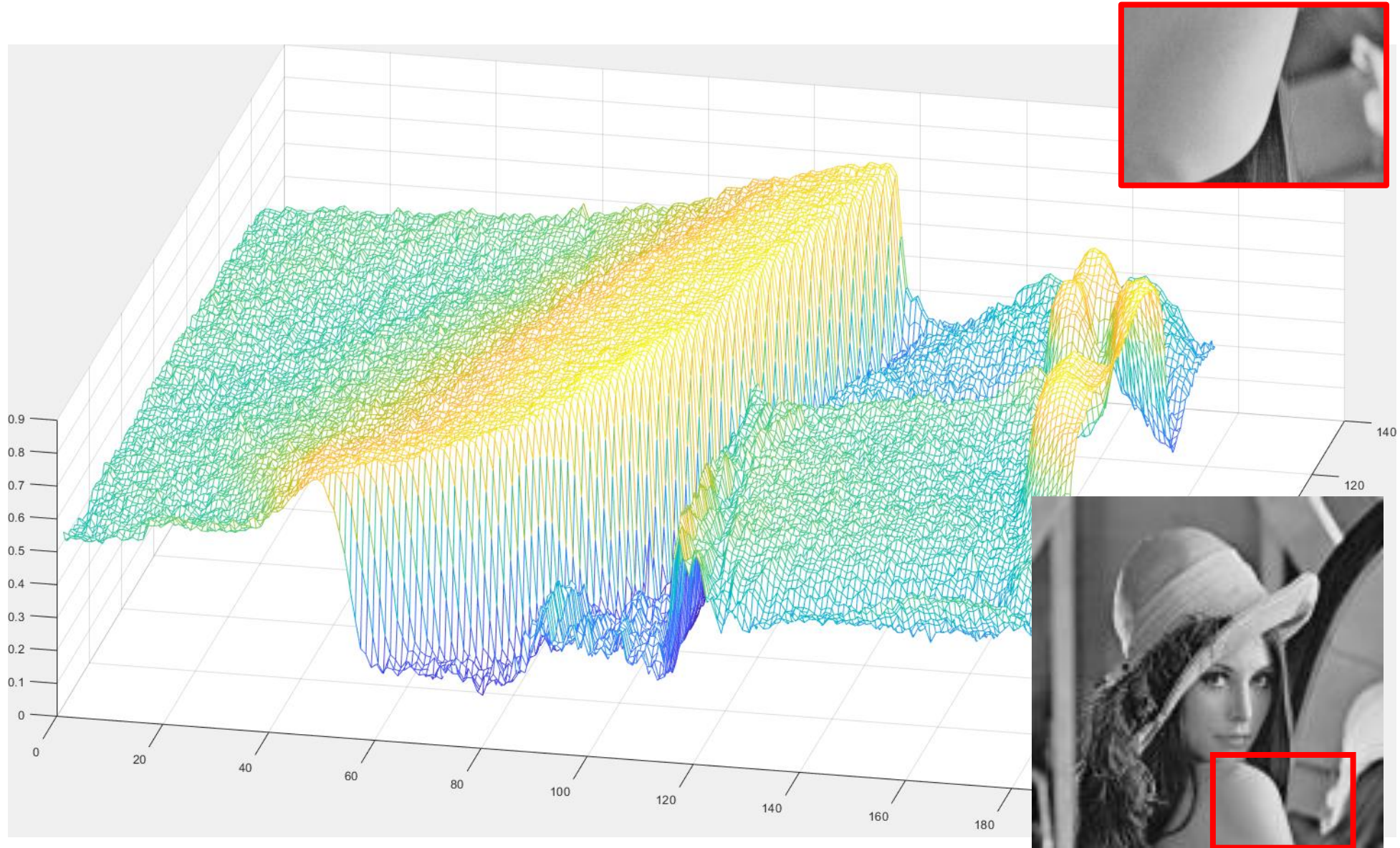
(Structural) MRI provide a 3D stack of grayscale images that are analyzing the body at different depth levels

These can be used to perform 3D reconstruction of bones, organs and membranes, e.g. cortical surfaces



When we work channel-wise...

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



Spatial-Domain vs Transform-Domain Methods

A survey of most important operations in image processing:

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

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

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

Bibliography

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

Intensity Transformations

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.

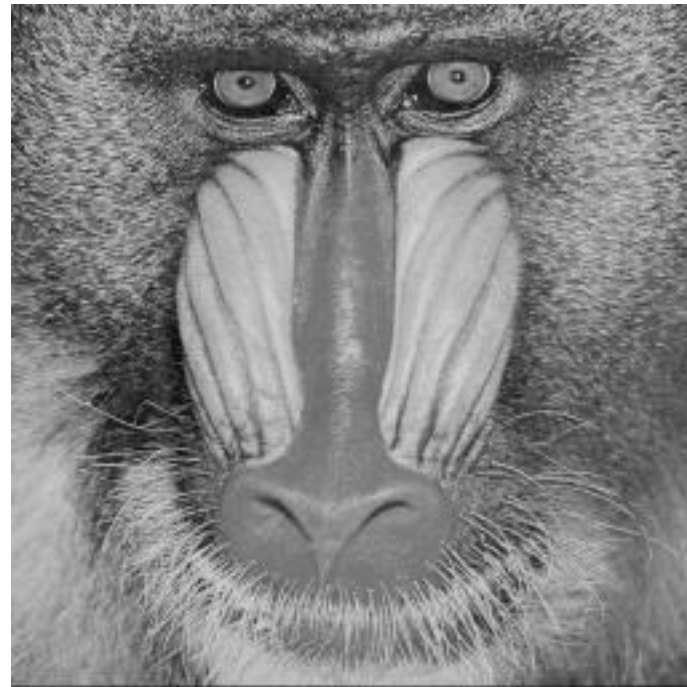
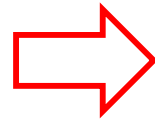
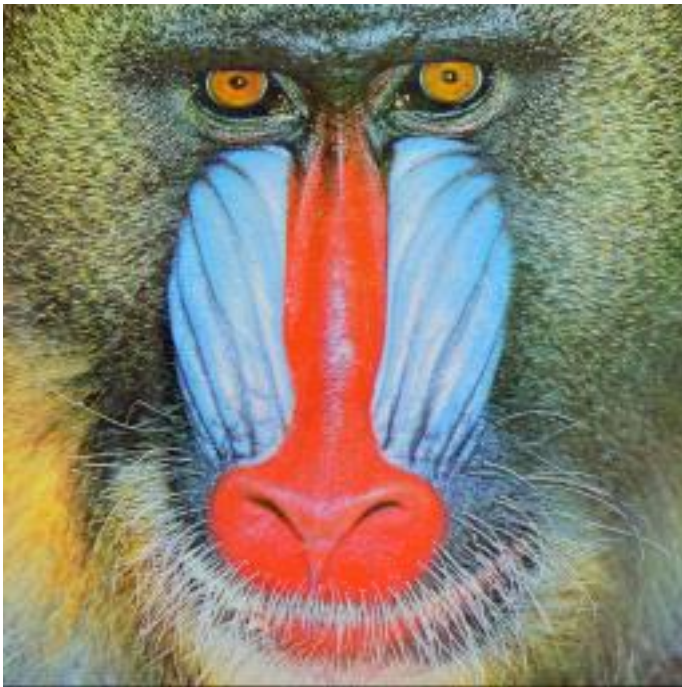
RGB \rightarrow Grayscale Conversion

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 RGB to YcBCr

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

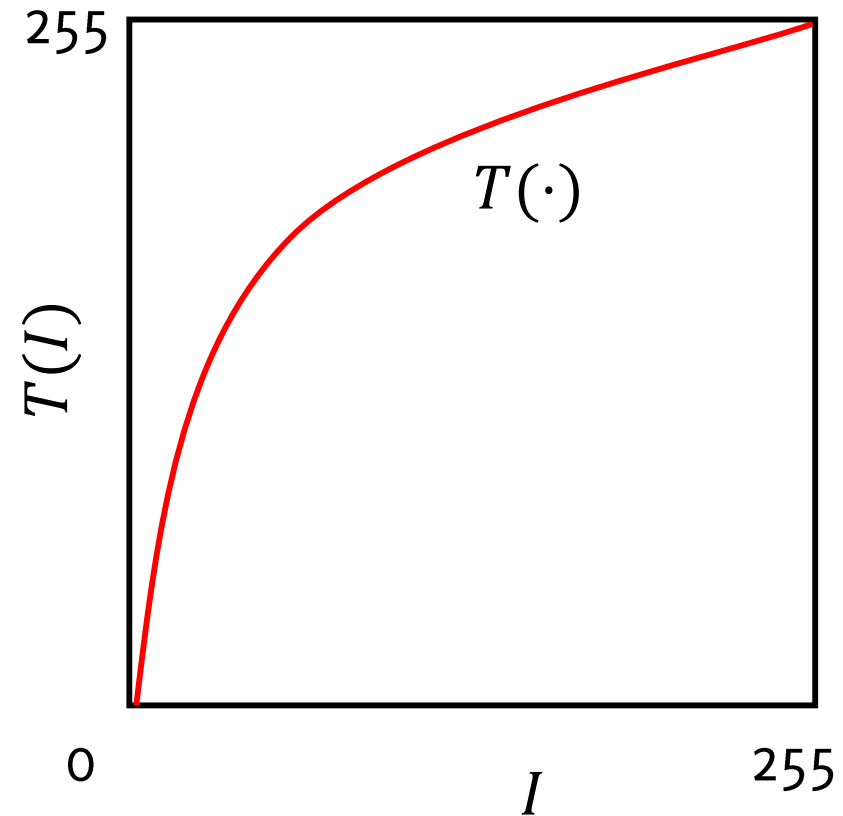
Human eye is less sensitive to color changes than luminance variations.
Thus,

- 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 the chromatic components are encoded at a coarser level than luminance)

Gray-level mapping

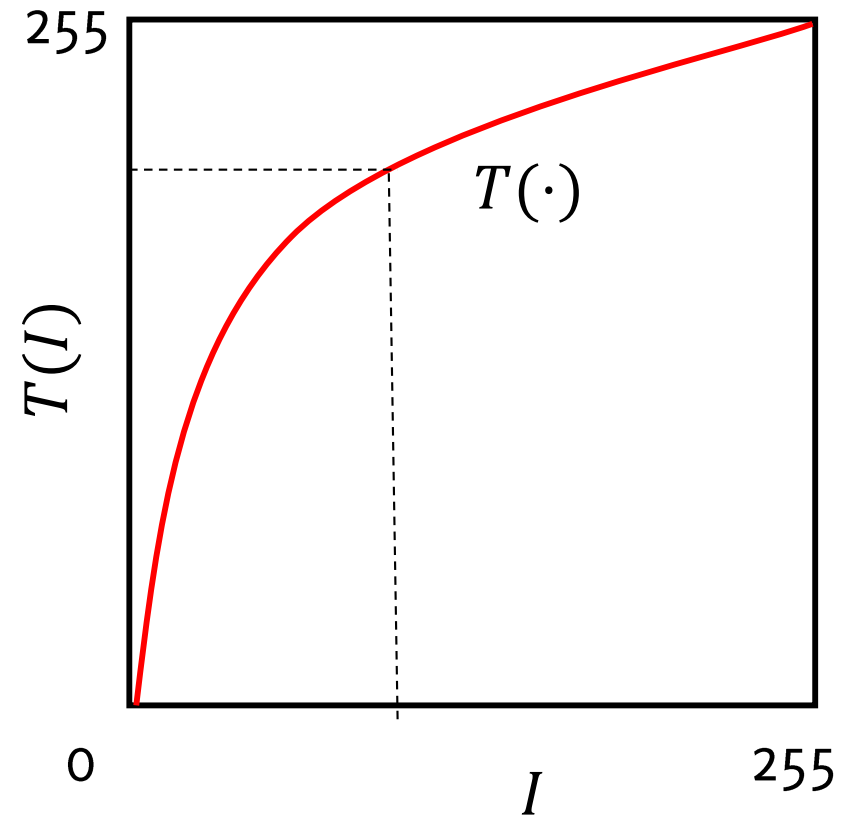
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operates on gray-scale images or on each color-plane separately



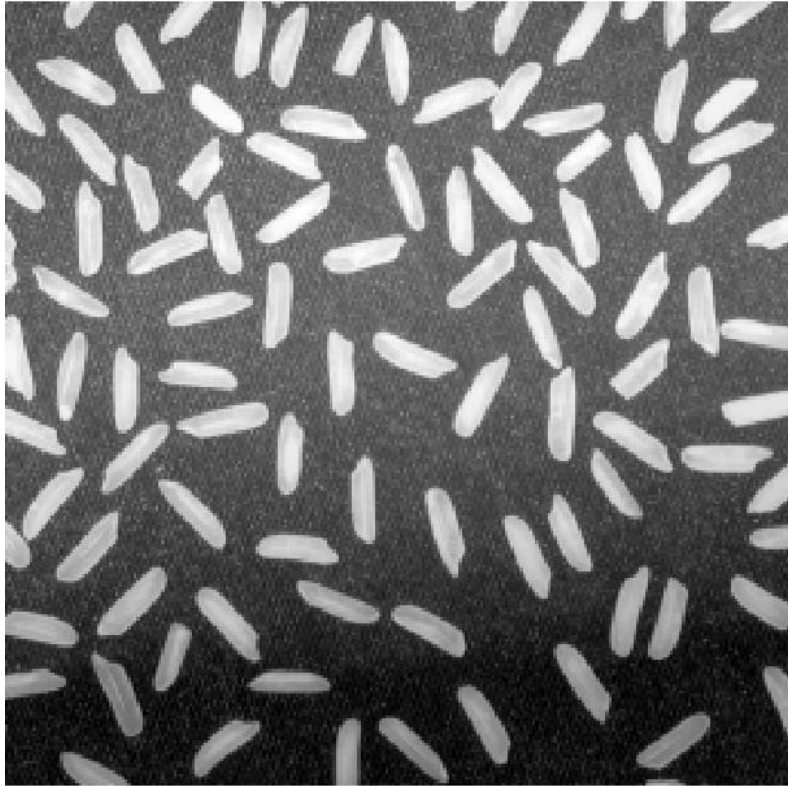
Gray-level mapping

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

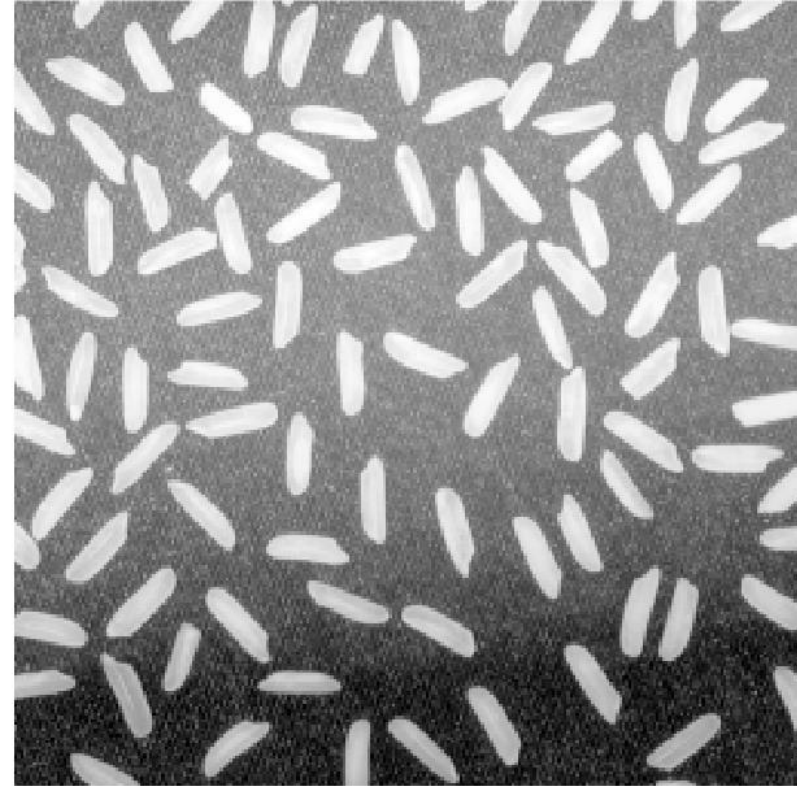
What does this T do?



Contrast increases in dark, decreases in bright



Input I

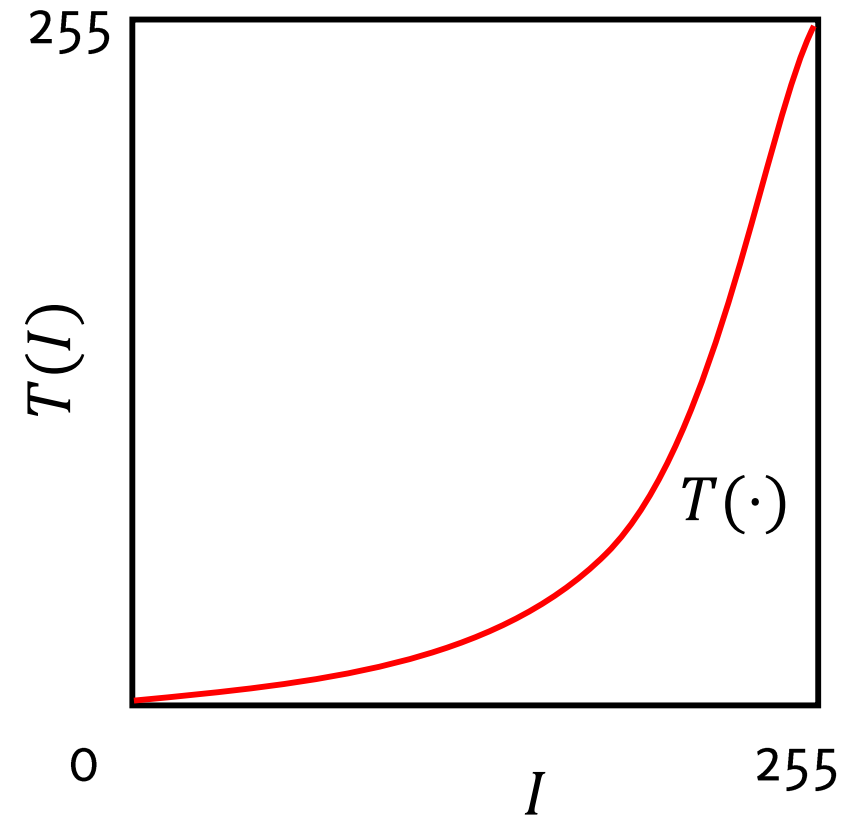


Output $G = T(I)$

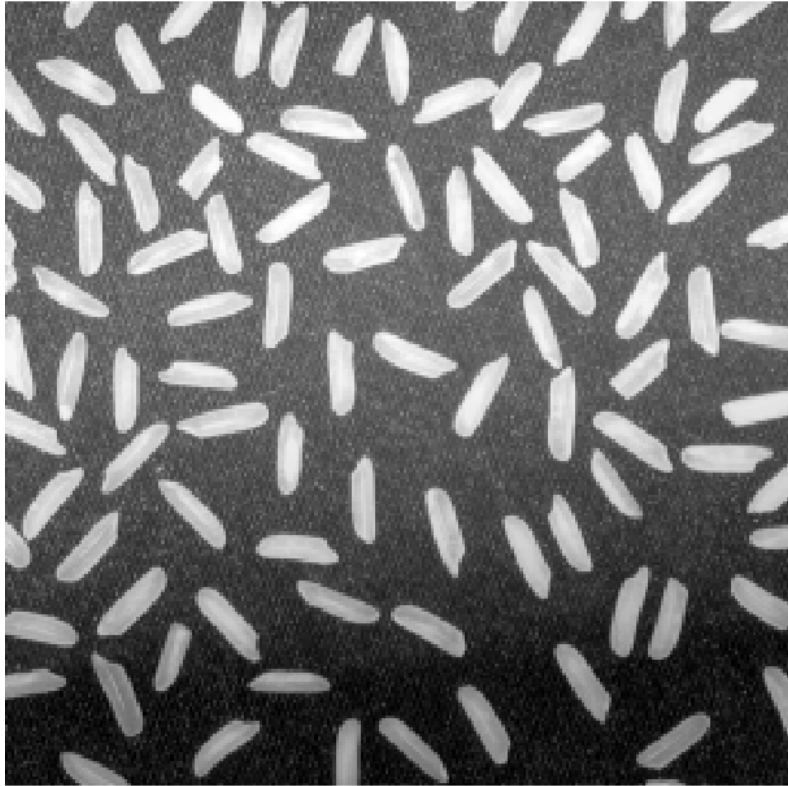
Gray-level mapping

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

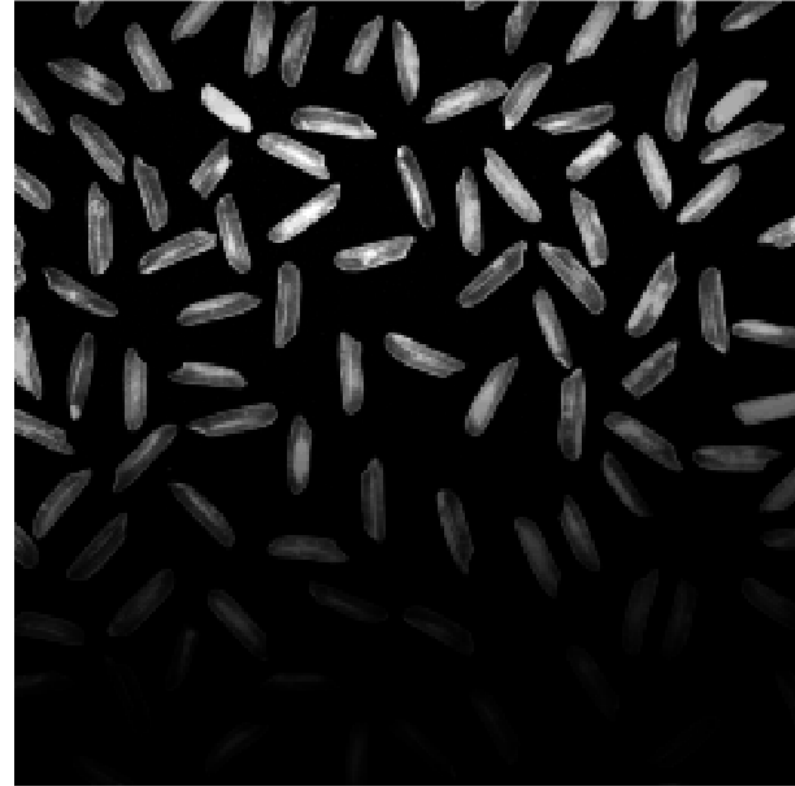
What does this T do?



Contrast increases in bright, decreases in dark



Input I

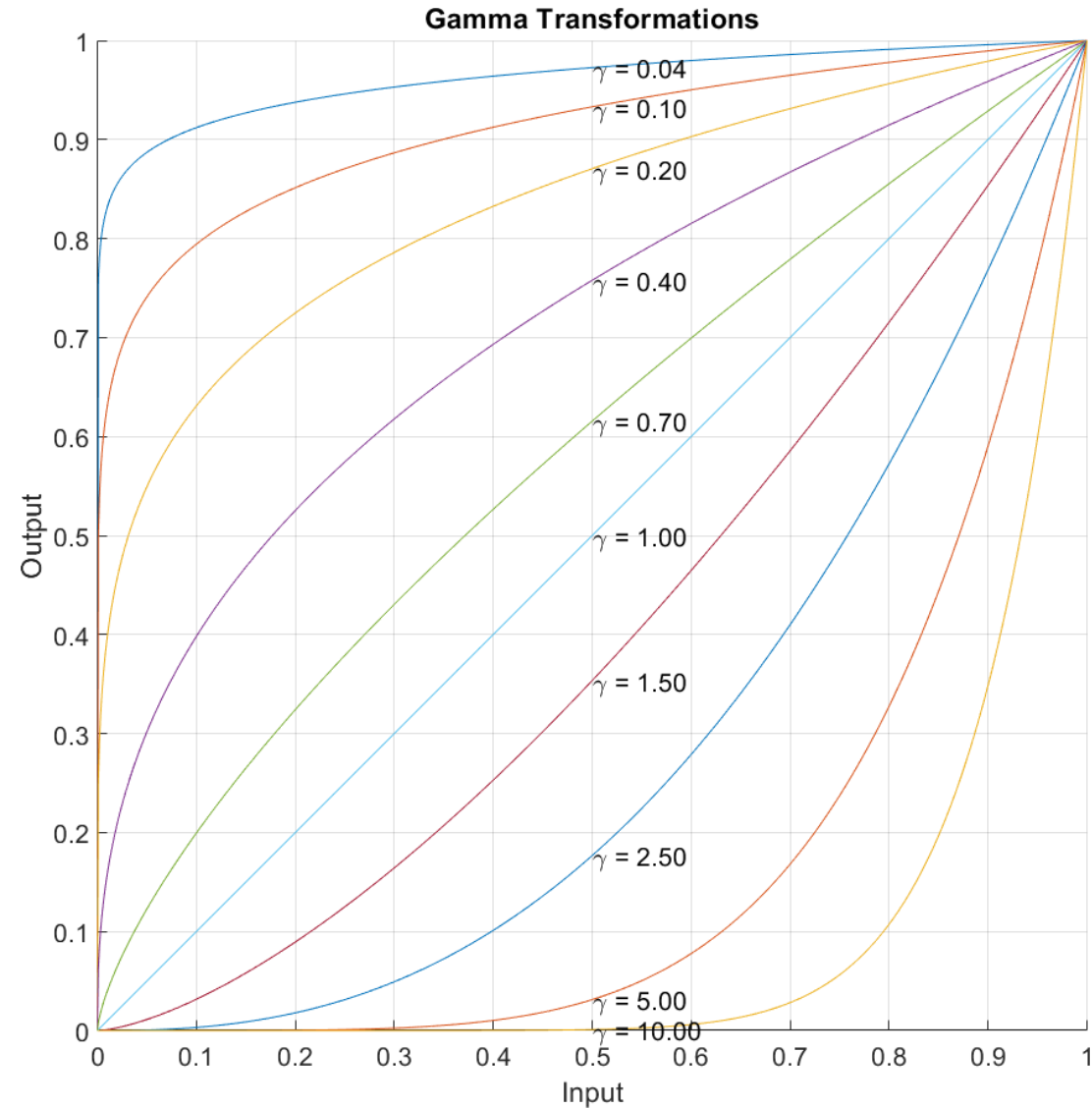


Output $G = T(I)$

Gamma Correction

Power-law transformation that can be written as

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



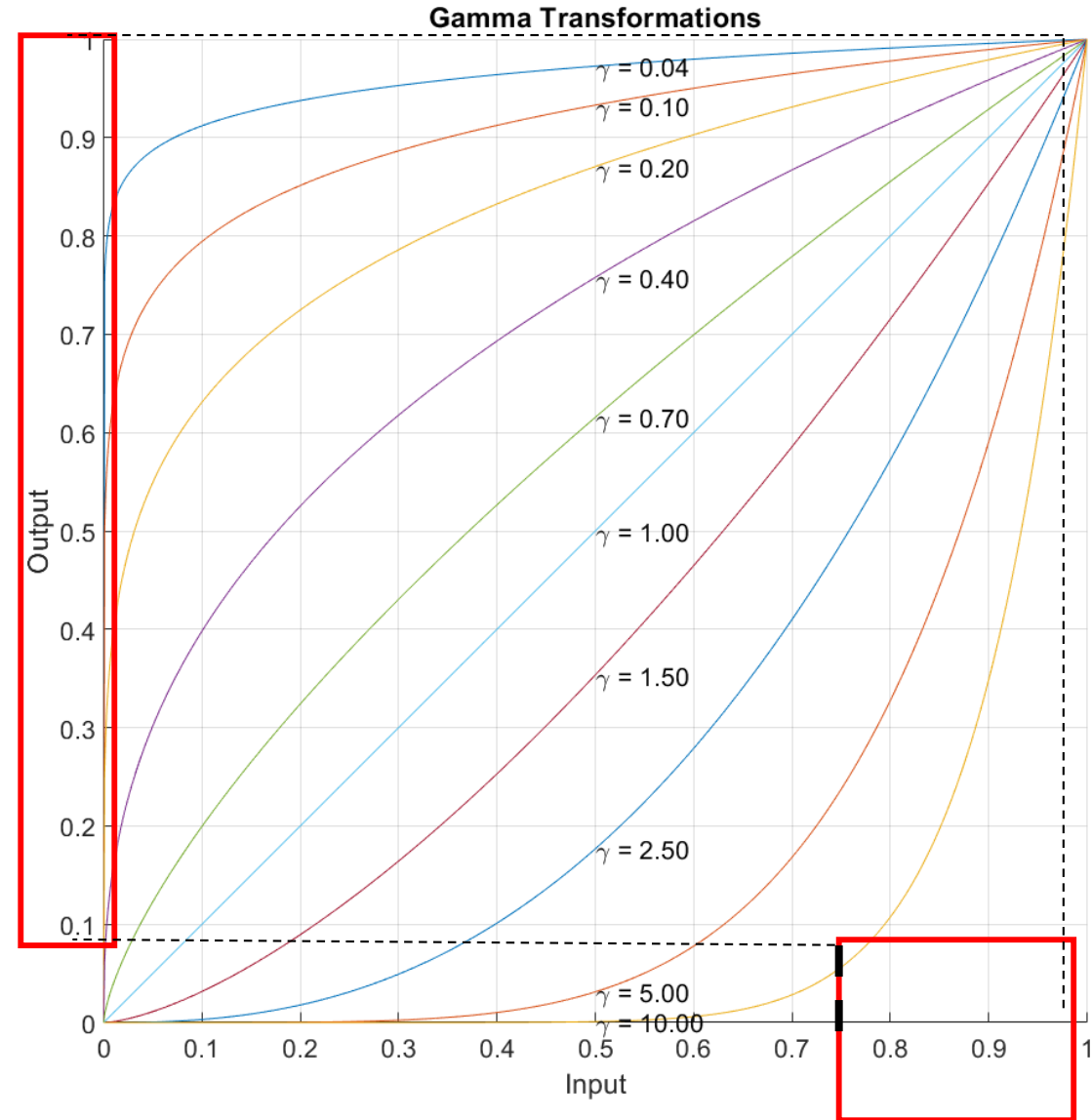
Gamma Correction

Power-law 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



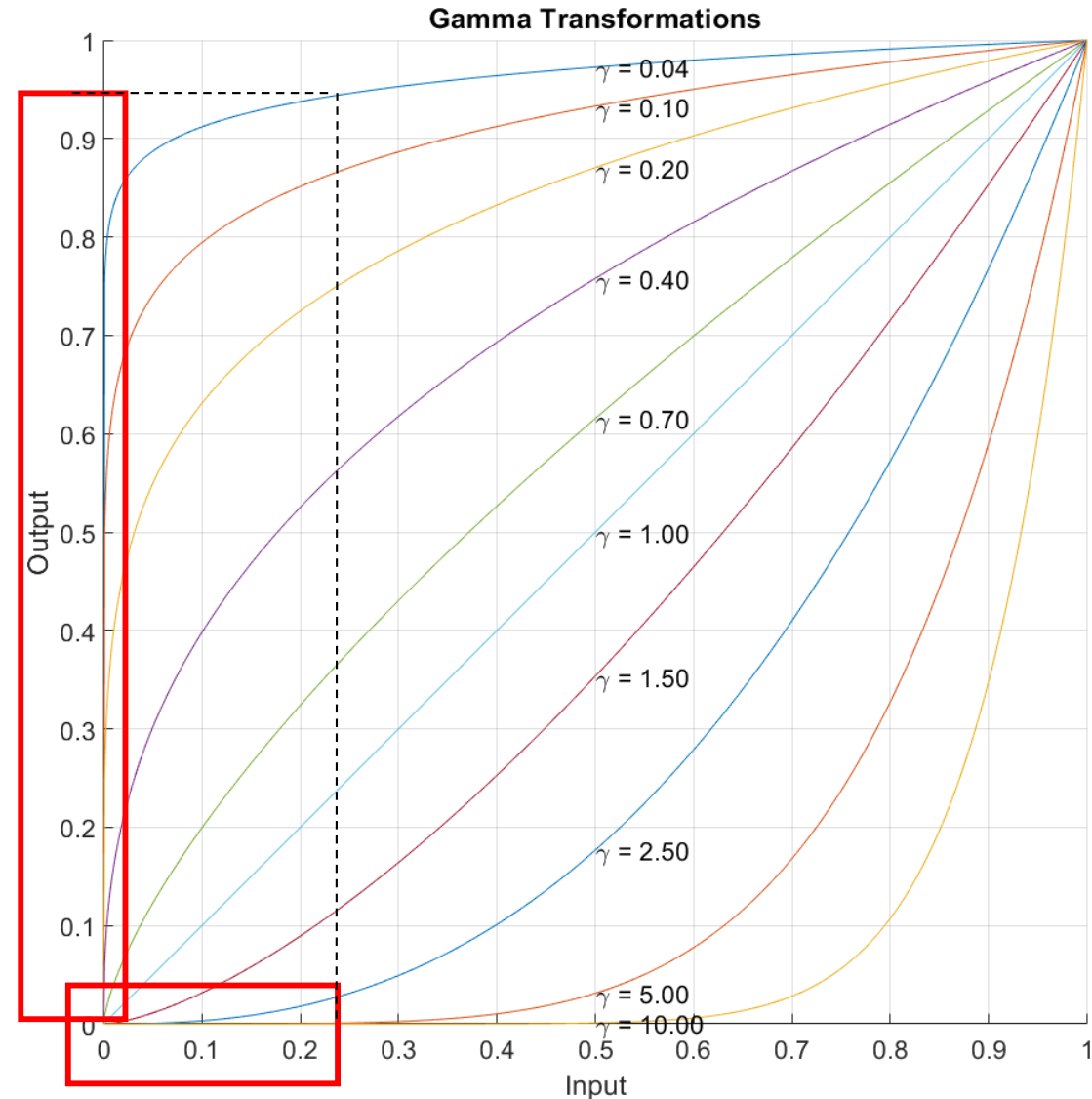
Gamma Correction

Power-law transformation that can be written as

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

Contrast Enhancement:

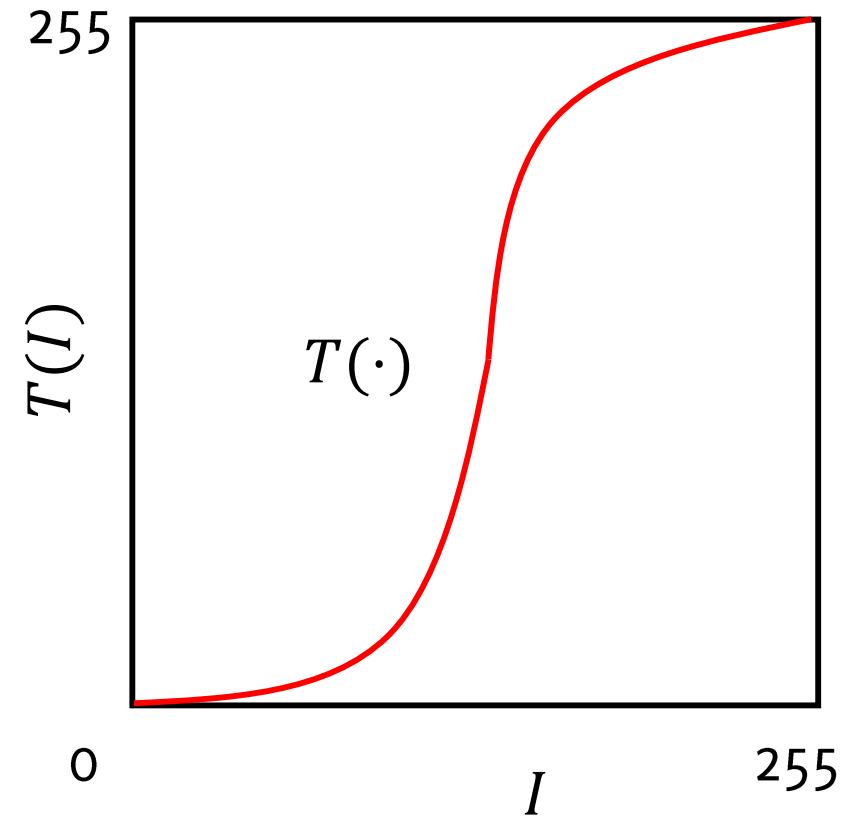
- Low values of γ stretch the intensity range at high-values
- High values of γ stretch the intensity range at low values



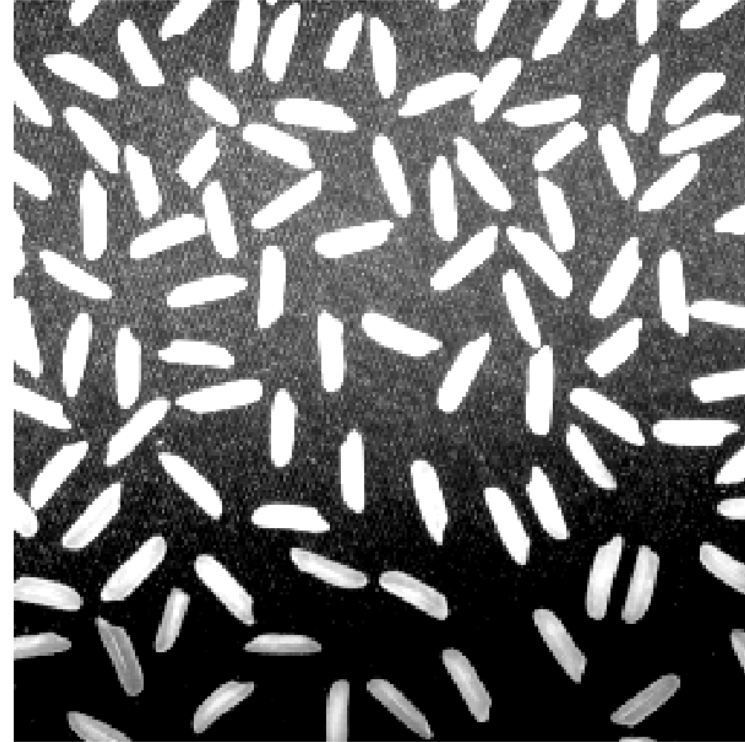
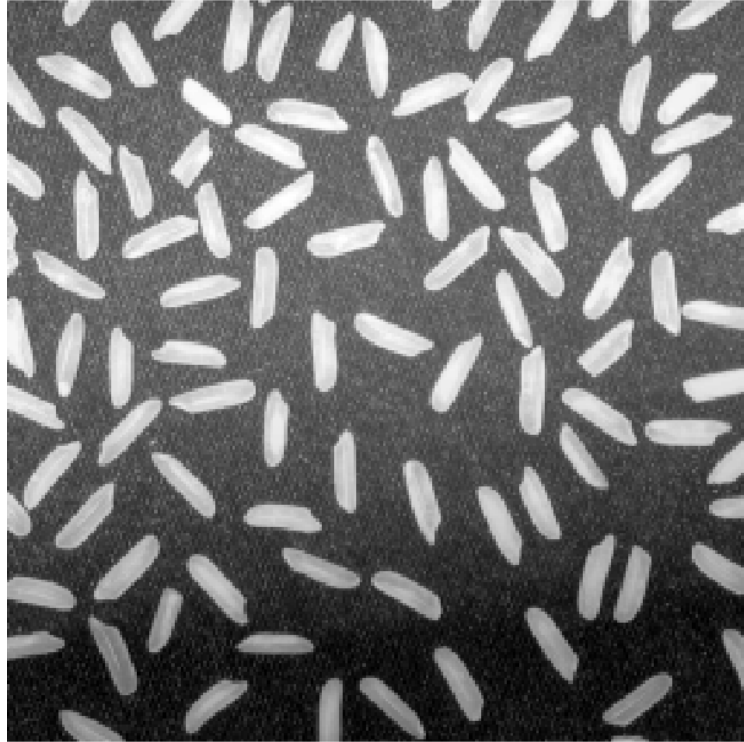
Gray-level mapping

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

What does this T do?



Contrast Stretching



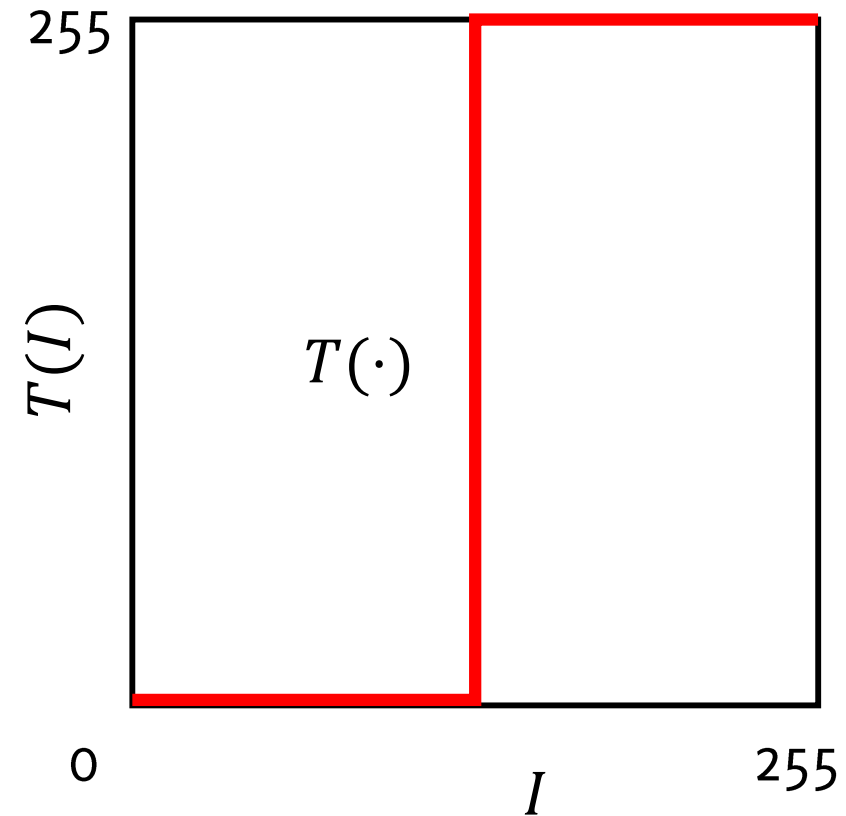
Contrast stretching: increases the contrast at values in the middle of intensity range, decreases contrast at bright and dark regions.

It is implemented by piecewise or parametric transformations

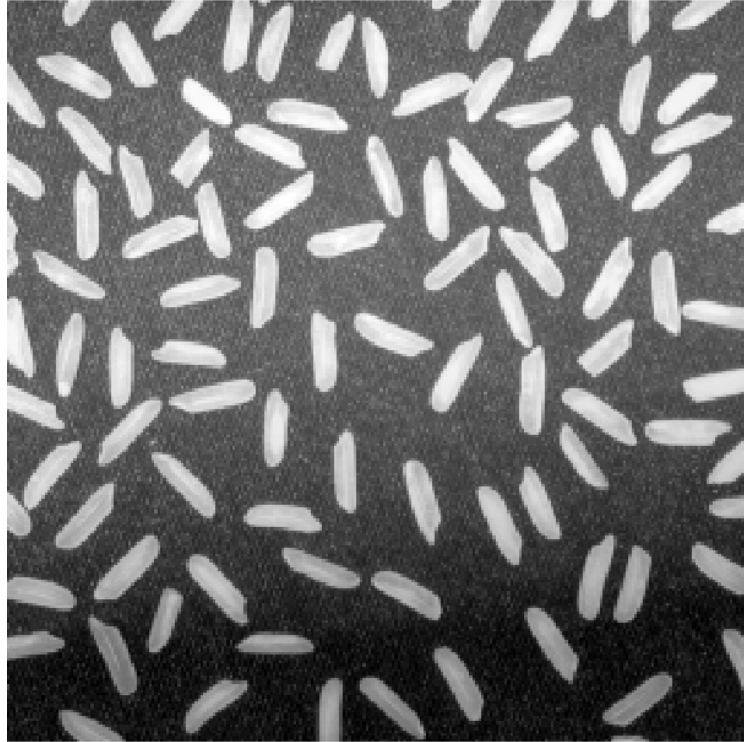
Gray-level mapping

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

What does this T do?



Thresholding

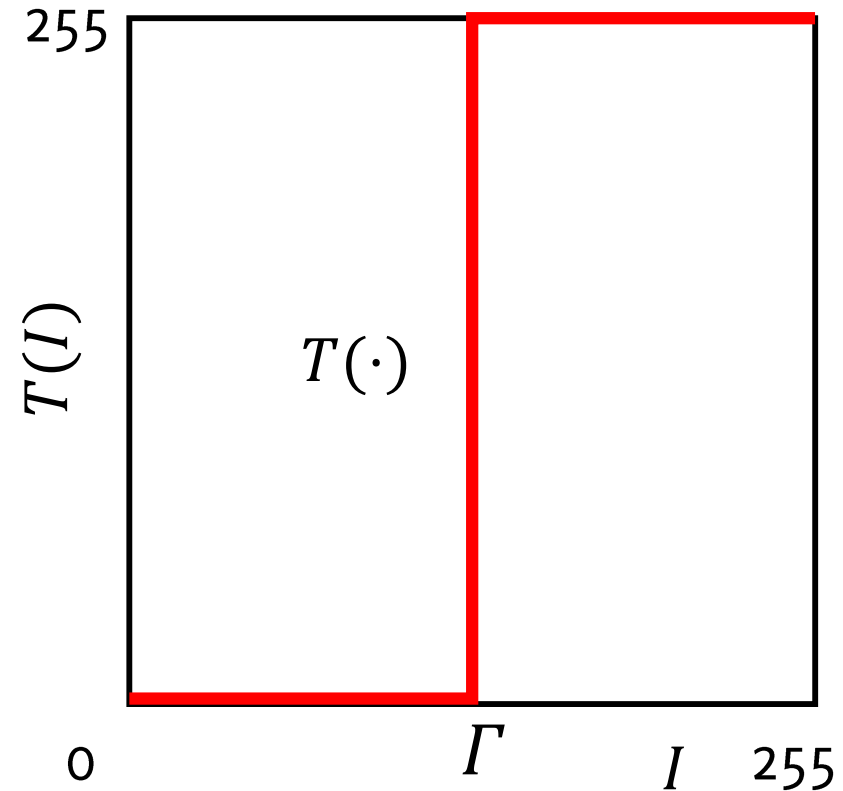


Thresholding binarizes images

Thresholding

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

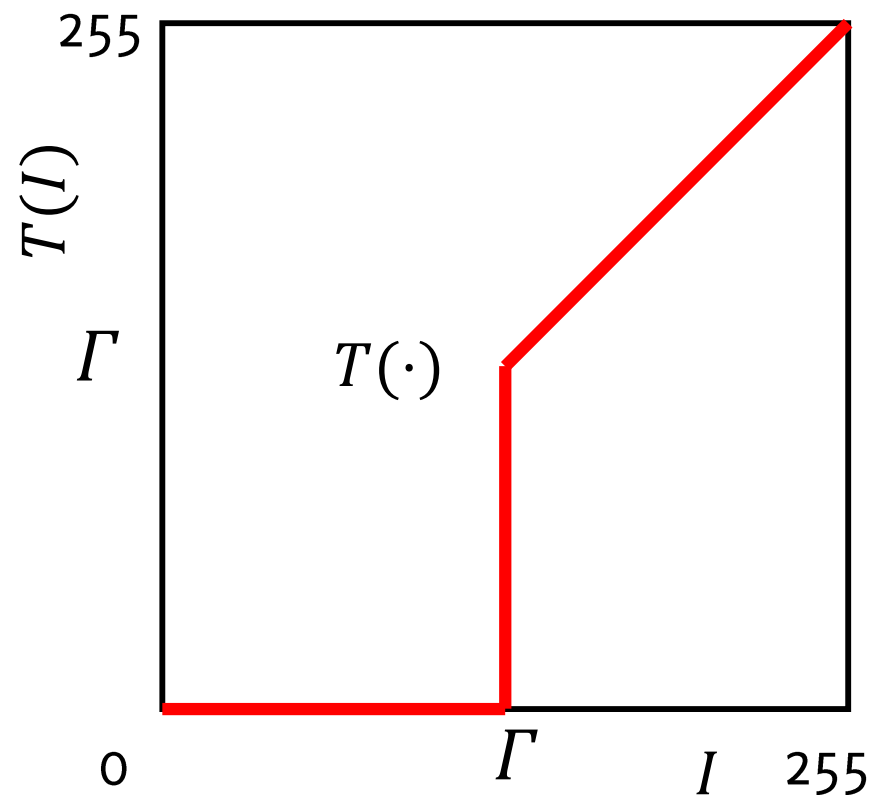
$$T(I(r, c)) = \begin{cases} 255, & \text{if } I(r, c) \geq \Gamma \\ 0, & \text{if } I(r, c) < \Gamma \end{cases}$$



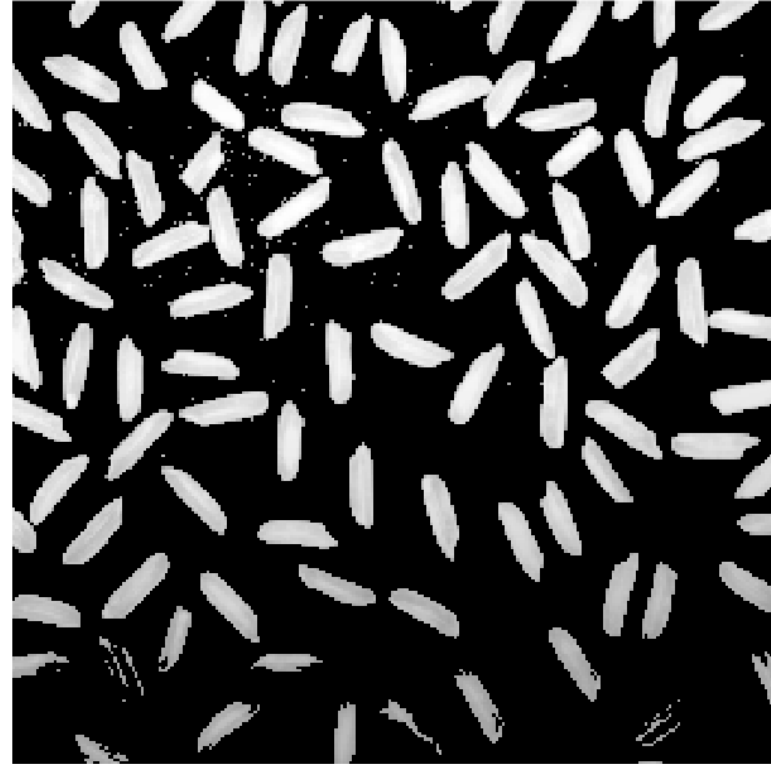
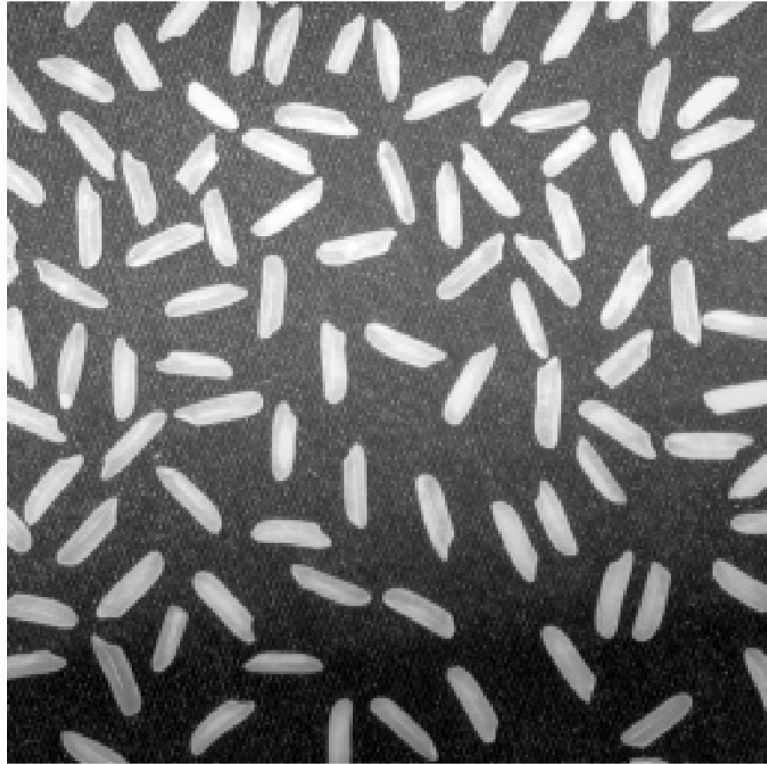
Thresholding

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

What does this T do?



Thresholding

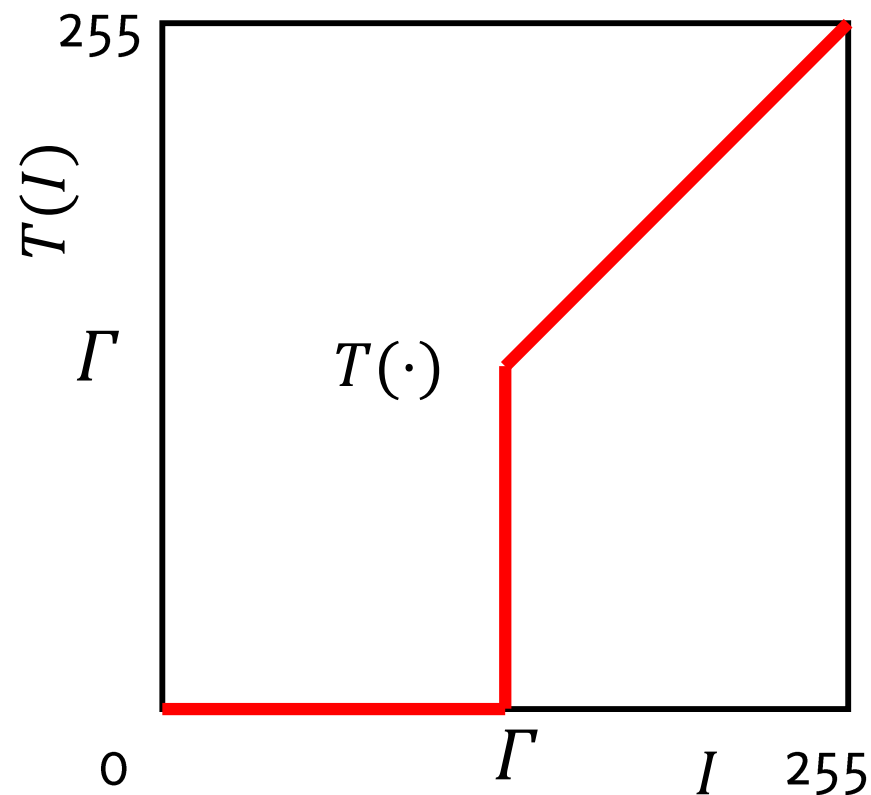


Thresholding

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

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

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



Digital Image Filters

Giacomo Boracchi

CVPR USI, March 27 2020

Book: GW chapters 3, 9

Outline

Image Transformations

- More on intensity transformation: histograms
- Spatial Transformation: Correlation and Convolution
- Convolution properties
- Smoothing
- Derivatives estimation
- Nonlinear Filters

Recap: Intensity Transformations

In general, these can be written as

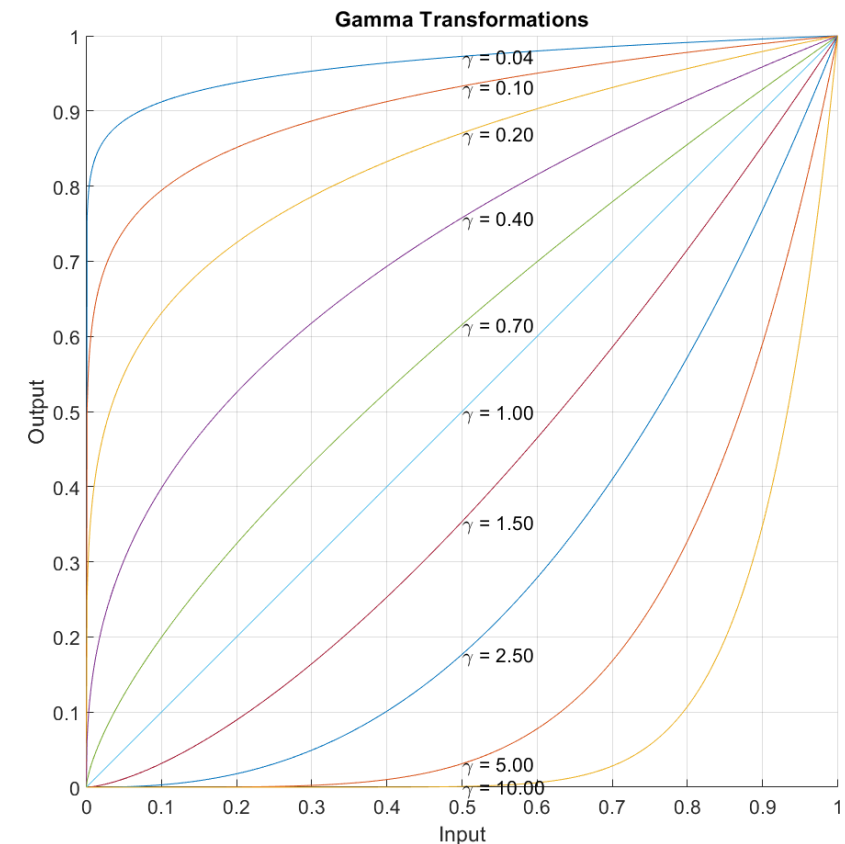
$$G(r, c) = T[I(r, c)]$$

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.

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

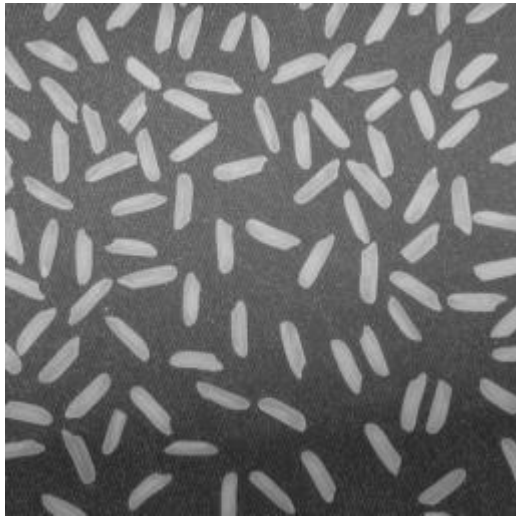


Histograms

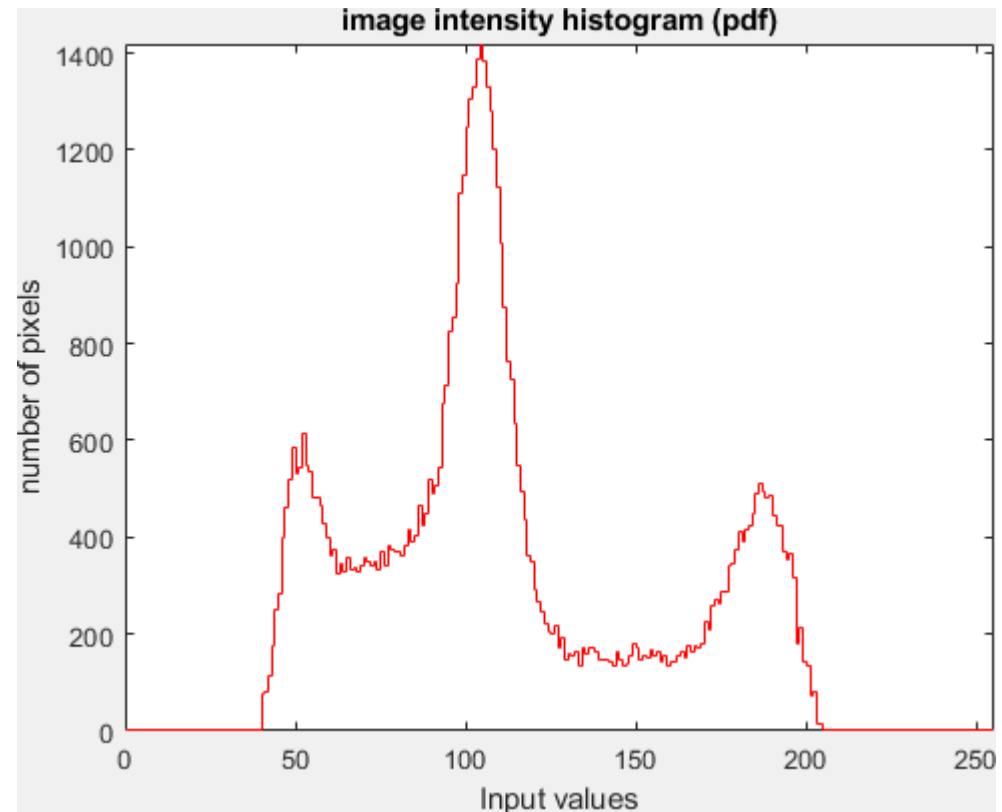
How to define intensity transformations adaptively on
the image

Image histograms

Histogram of pixel intensities can be used to define intensity transformation



img



Histogram

Image histograms

Histogram of pixel intensities can be used to define intensity transformation

Definition

The histogram $\{h_i\}$ associated to an image I is a vector of 256 bins, each corresponding to an intensity value $i = 0, \dots, 255$

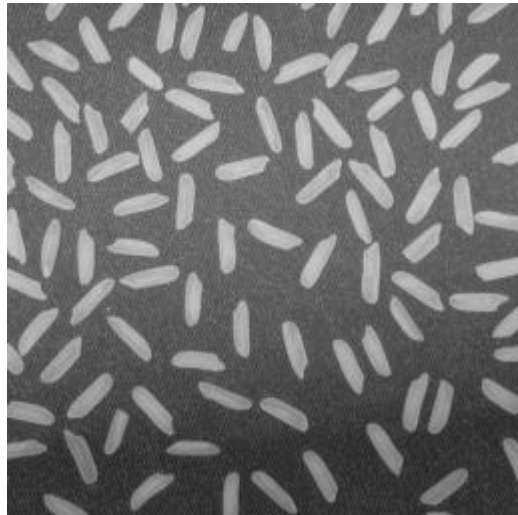
$$h_i = \#\{(r, c), \quad \text{s.t. } I(r, c) = i\}$$

Where $\#$ denotes the cardinality of a set

$$\mathbf{[h, bins]} = \mathbf{hist(I, bins)}$$

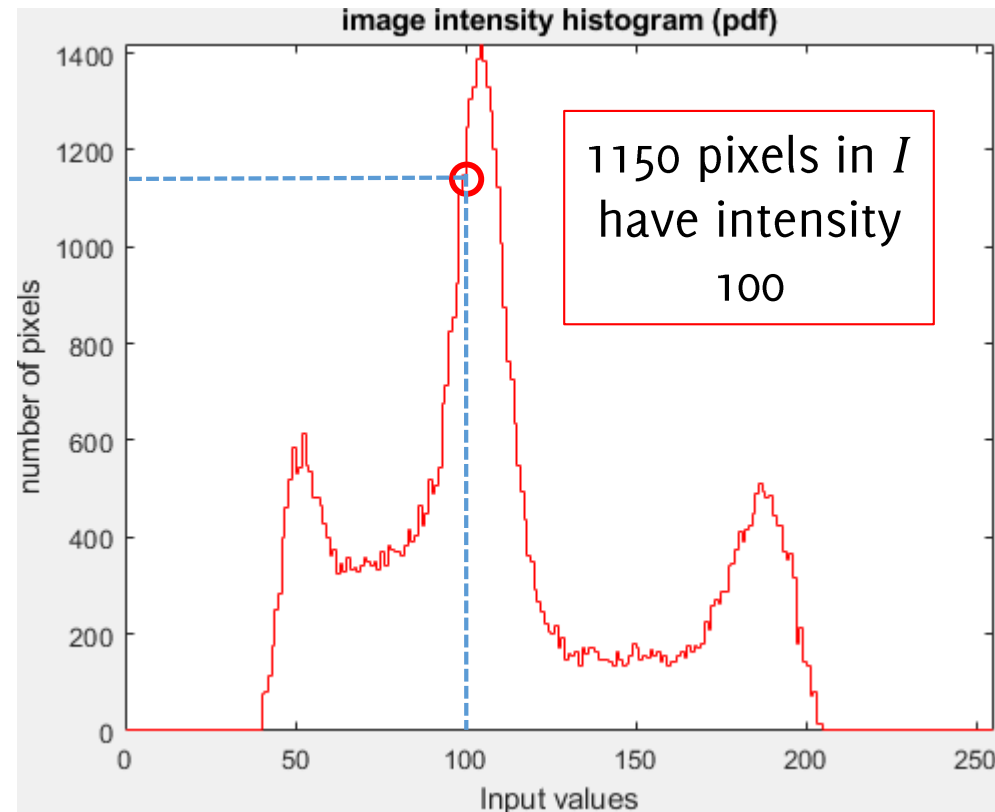
Image histograms

Histogram of pixel intensities can be used to define intensity transformations



img

$$h_{100} = 1150$$

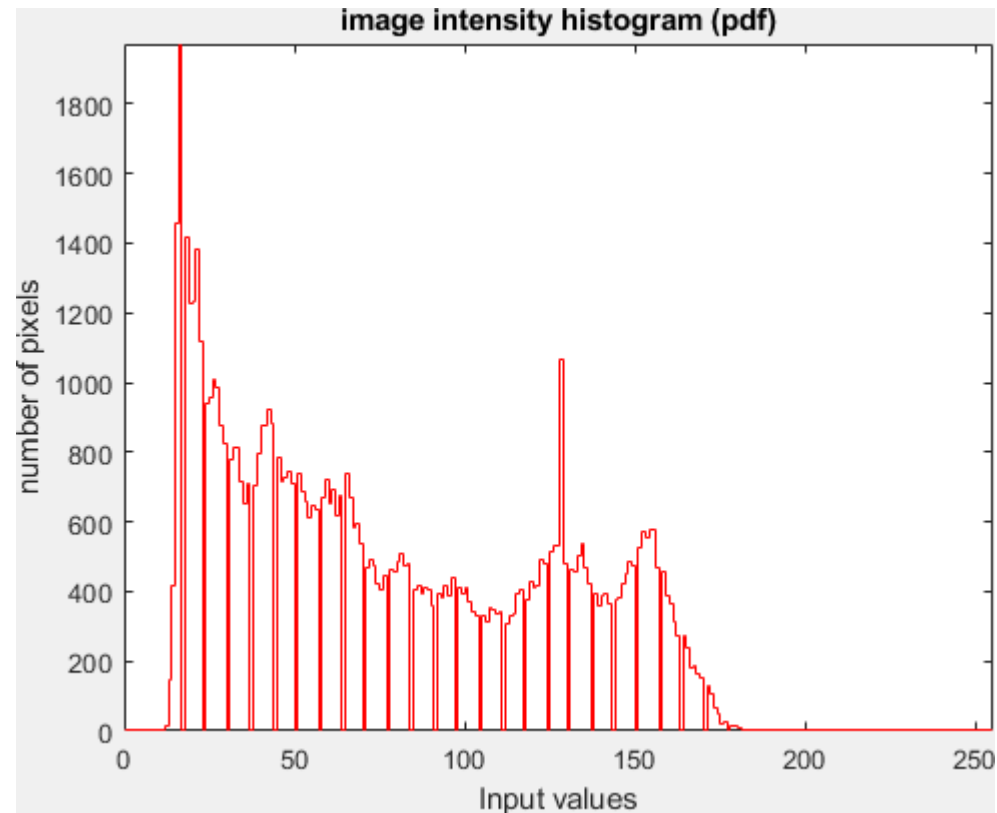
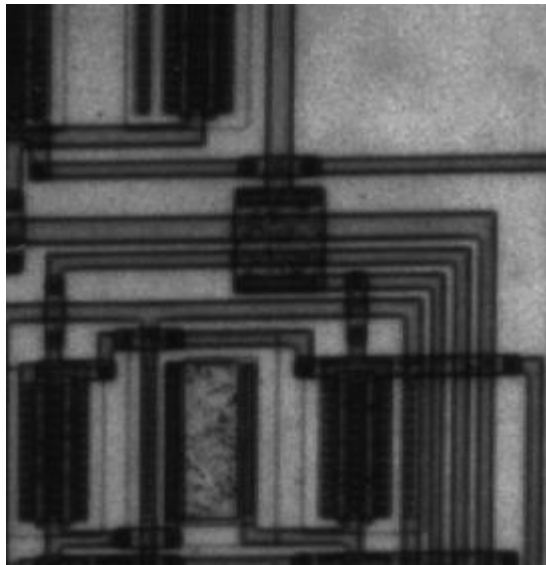


$$i = 100$$

Histogram

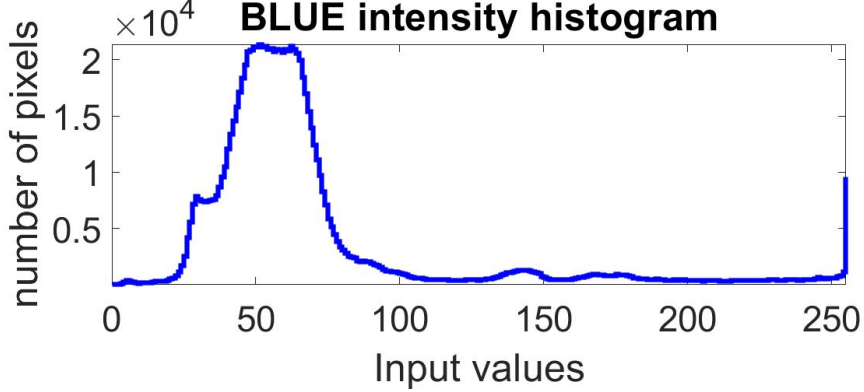
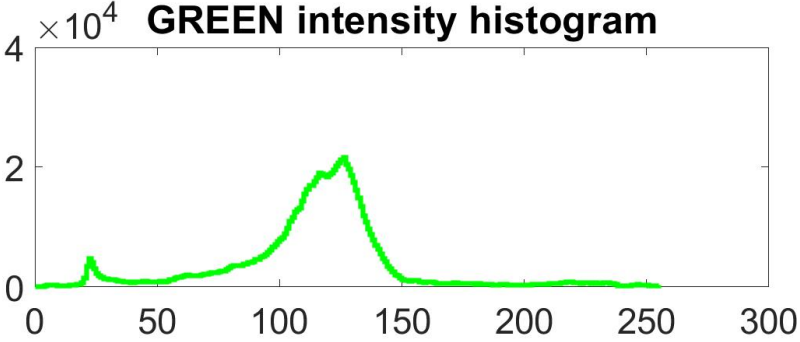
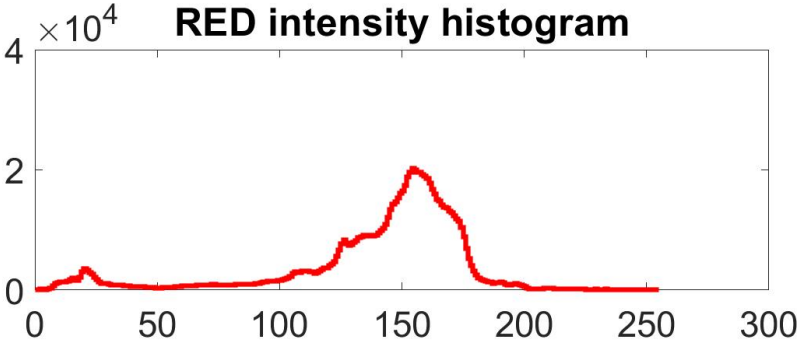
Image histograms

Remember that images assume integer values (uint8), thus there might be intensity values that do not occur in an image



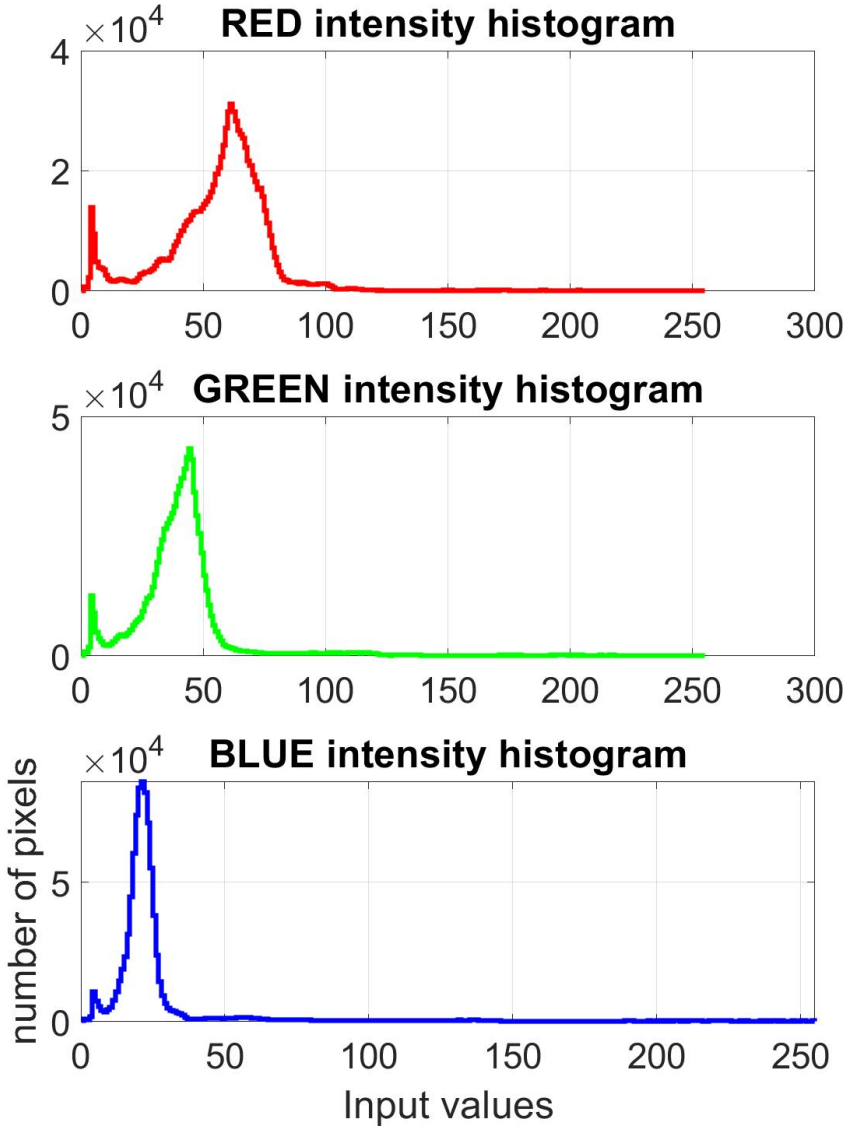
Histogram of an NORMAL image

NORMAL image



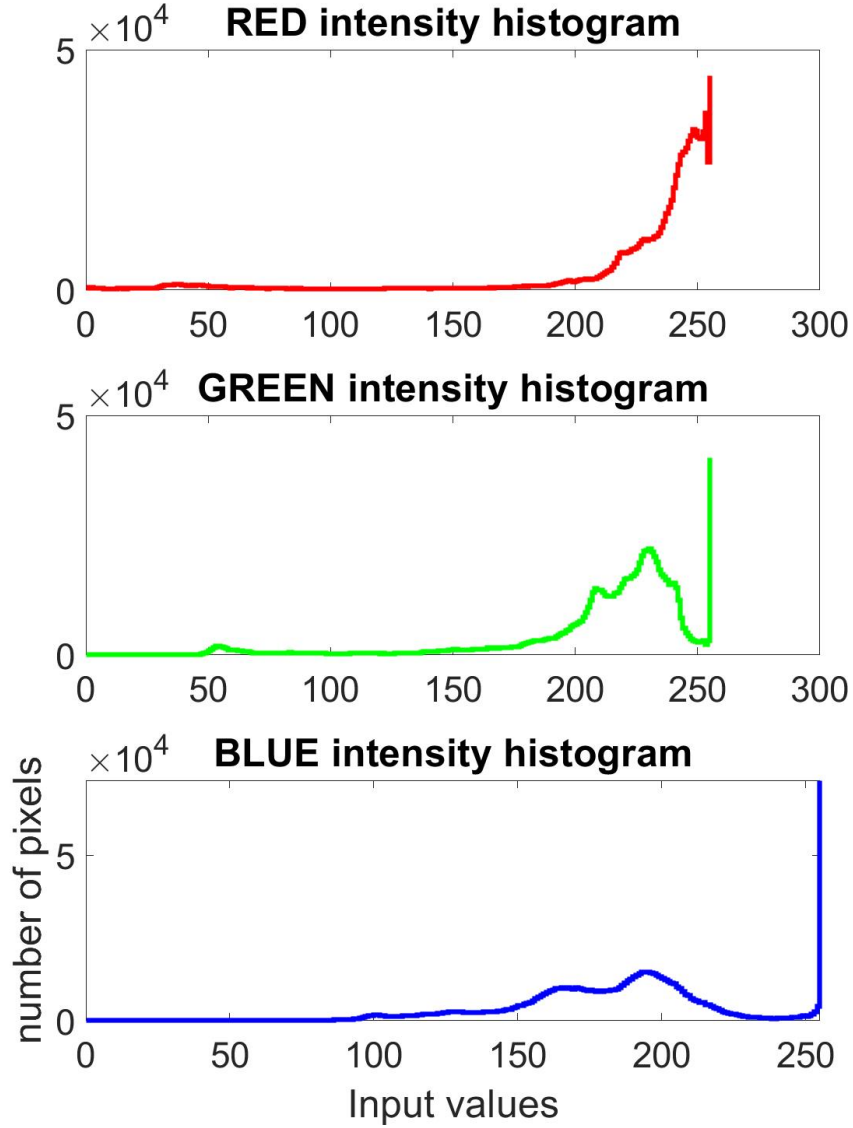
Histogram of an UNDEREXPOSED image

UNDEREXPOSED image



Histogram of an OVEREXPOSED image

OVEREXPOSED image



Contrast Enhancement by histogram equalization

Contrast enhancement transformation map the image intensity to the whole range $[0,255]$

Histogram equalization maps histogram bins. Let

- $[0, L]$ be the intensity range of input image
- $\{h_j\}$ be the histogram of the input image and let $p_j = h_j/N$ be the proportion of pixels having intensity j in the input image

Histogram equalization is defined as

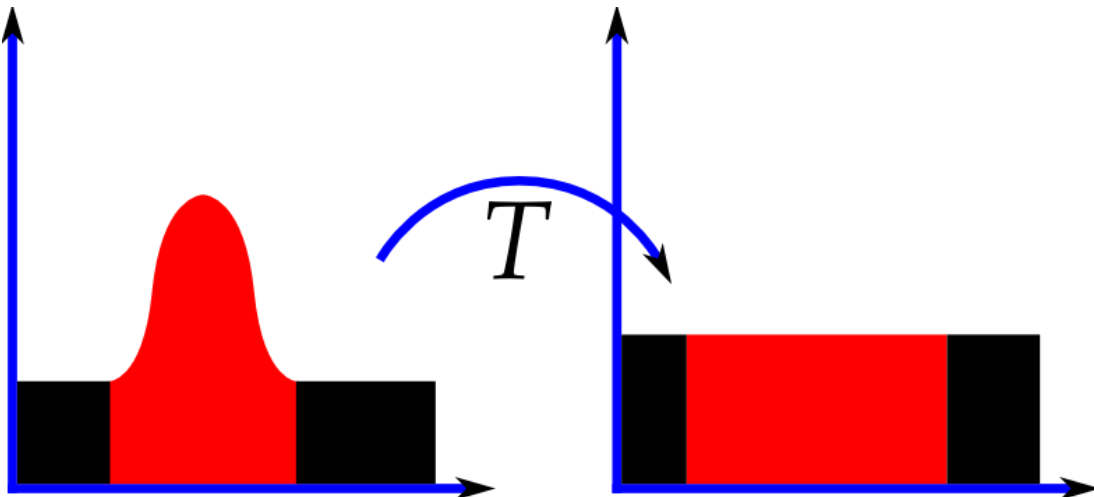
$$T(i) = \text{floor} \left((L - 1) \sum_{j=0}^i p_j \right)$$

Contrast Enhancement by histogram equalization

Histogram equalization maps any pdf (histogram of the input image) to a uniform pdf (output image)

$$T(i) = \text{floor} \left((L - 1) \sum_{j=0}^i p_j \right)$$

This transformation maps the histogram to a uniform distribution



Contrast Enhancement by histogram equalization

Histogram equalization maps any pdf (histogram of the input image) to a uniform pdf (output image)

$$T(i) = \text{floor} \left((L - 1) \sum_{j=0}^i p_j \right)$$

Rationale: the **cumulative function** CDF of a random variable I maps the random variable to a uniform distribution, i.e.

$$CDF(I) \sim U(0,1)$$

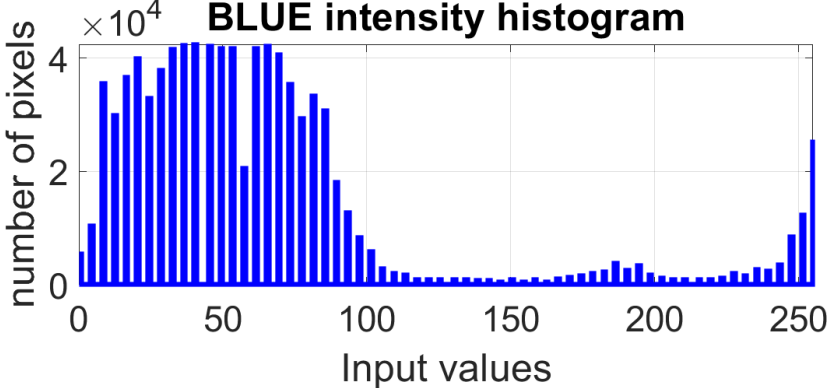
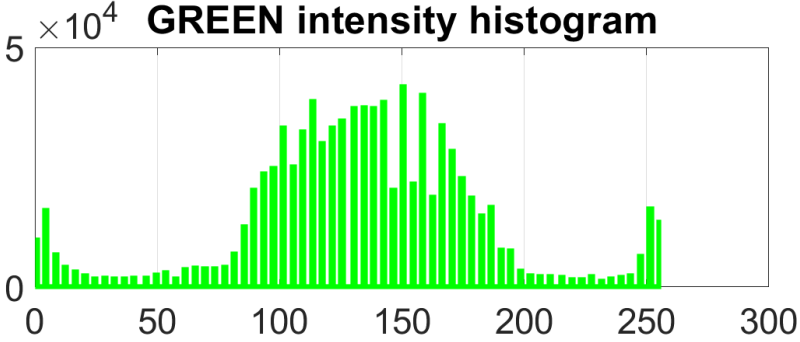
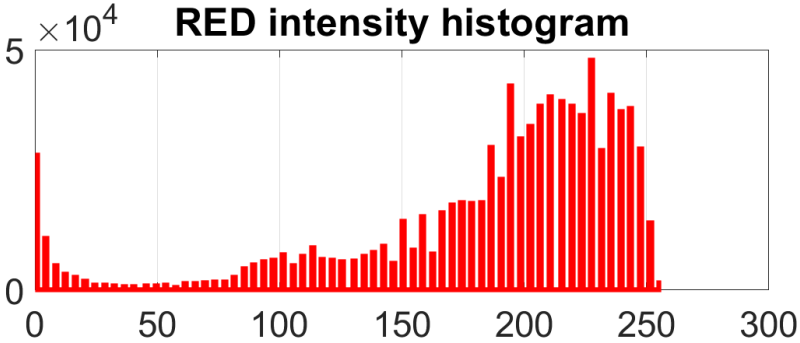
The transformation then becomes the cumulative function itself:

$$T(\cdot) = CDF(\cdot)$$

$$\mathbf{I}_{eq} = \mathbf{histeq}(\mathbf{I})$$

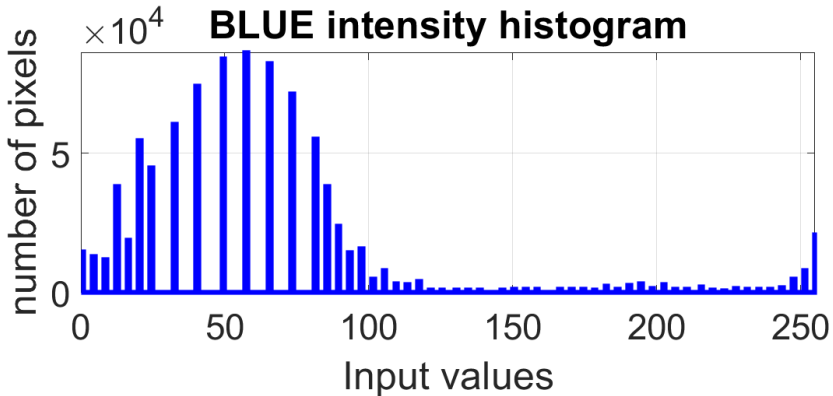
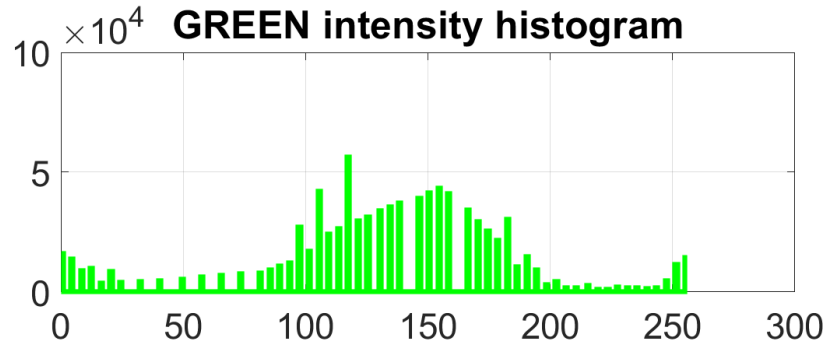
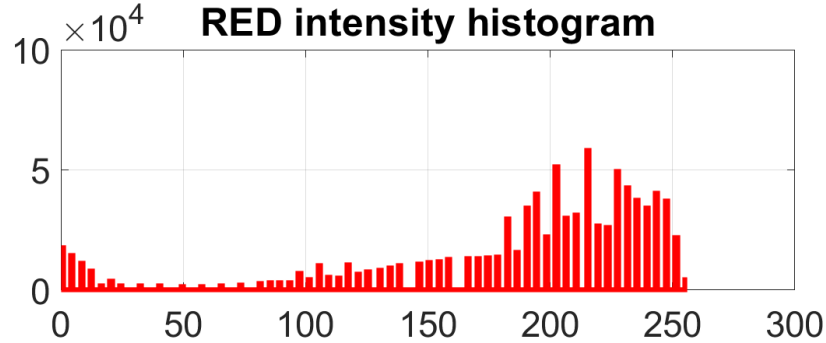
Histogram Equalization Results

NORMAL EQUALIZED image



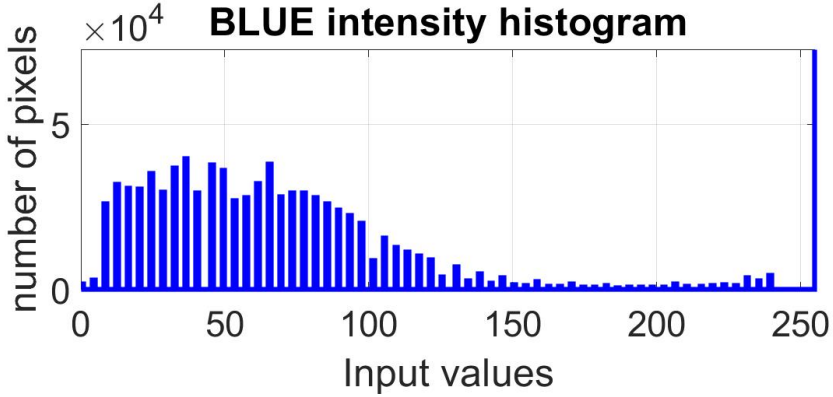
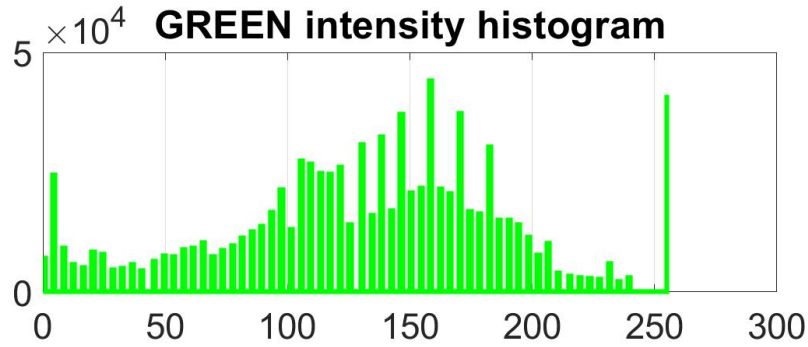
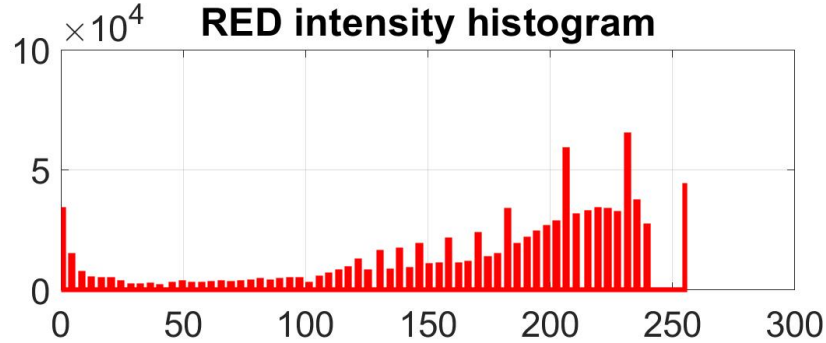
Histogram Equalization Results

UNDEREXPOSED EQUALIZED image



Histogram Equalization Results

OVEREXPOSED EQUALIZED image

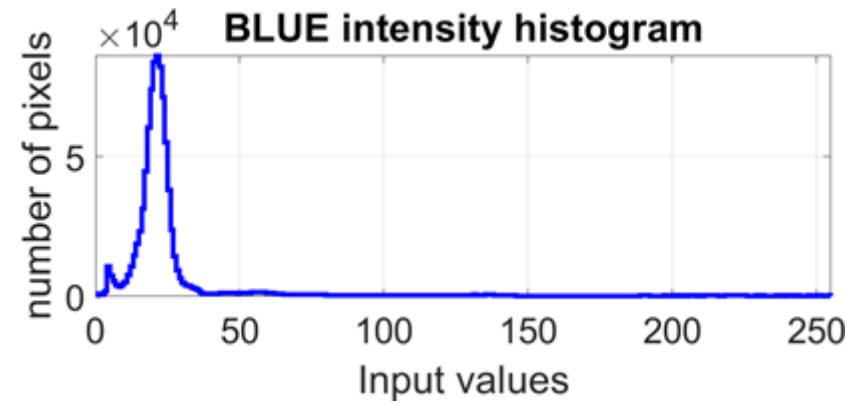


Equalization can not create new intensity values!

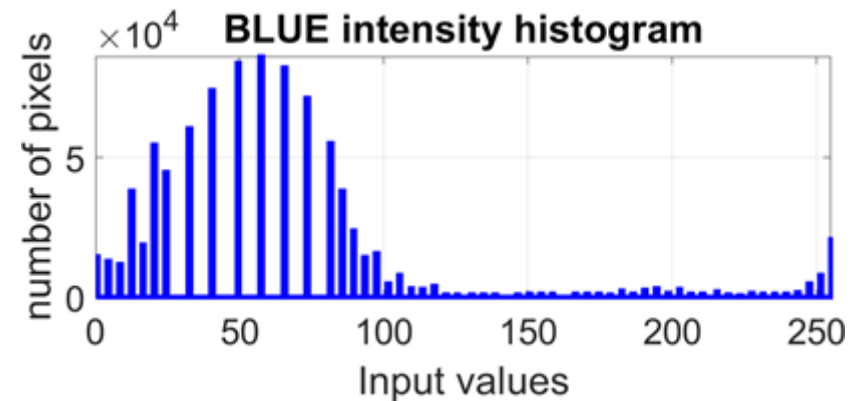
UNDEREXPOSED image



UNDEREXPOSED EQUALIZED image



↓ Histogram equalization



Histogram Matching

Estimate the intensity transformation mapping an histogram to any target distribution.

For instance, given two images I_1 and I_2 , estimate the transformation mapping the histogram of I_1 to the histogram of I_2

Histogram Matching

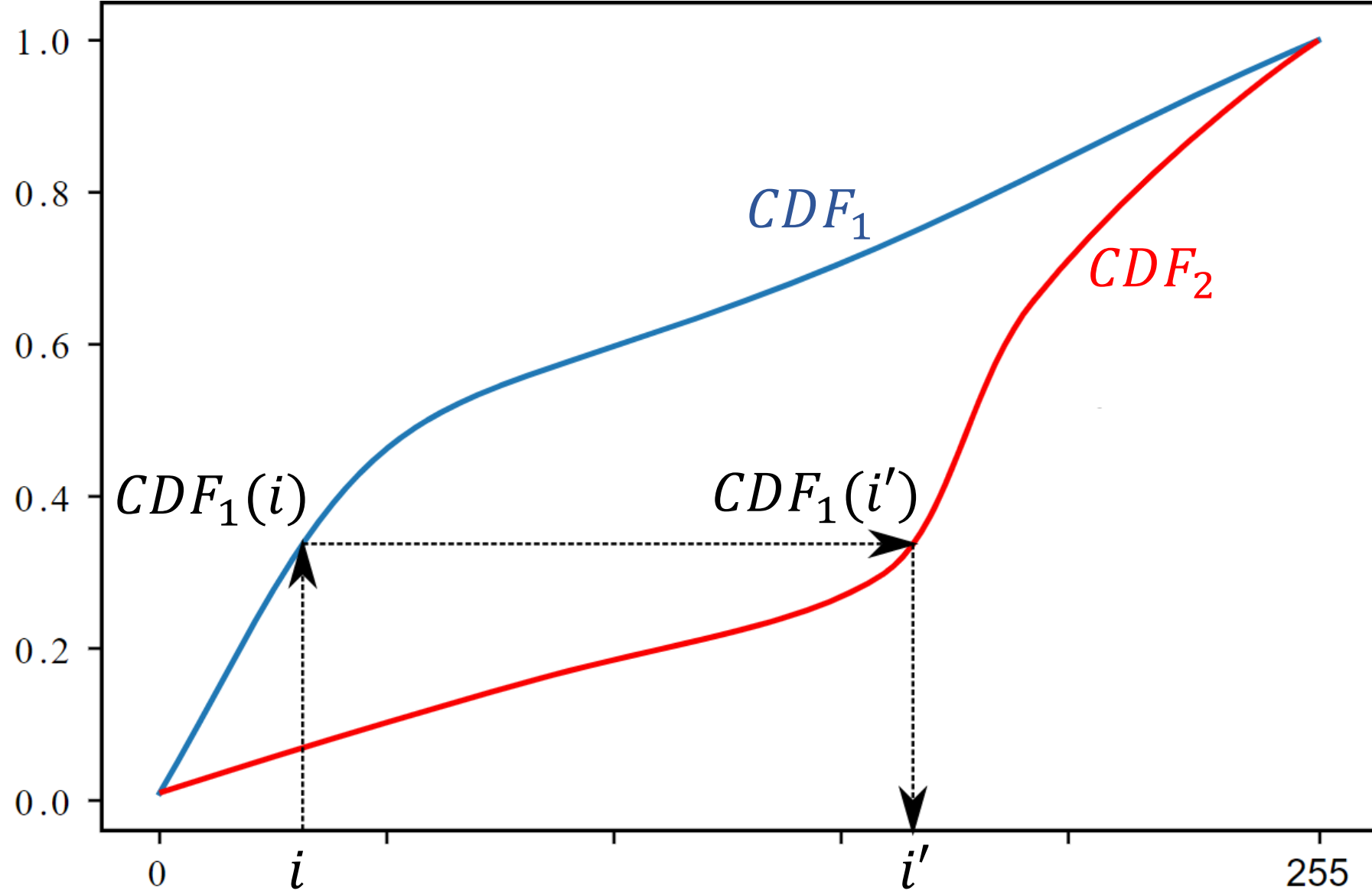
Idea: Estimate the transformation that makes their cumulative density functions to be the same

The transformation

$$i' = T(i), \quad \text{such that}$$
$$CDF_1(i) = CDF_2(i')$$

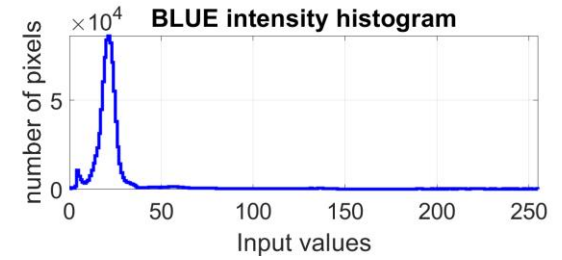
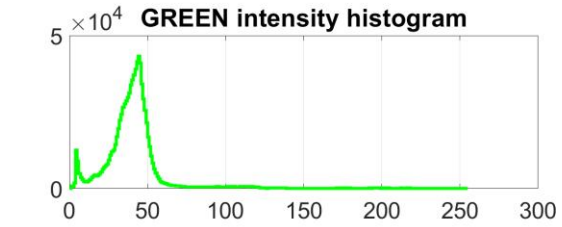
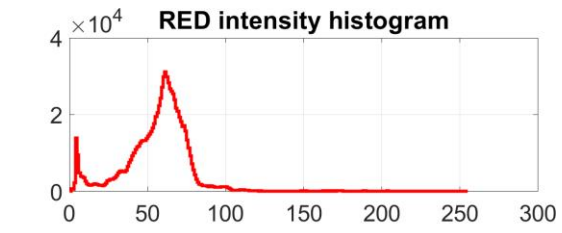
Solves this problema and can be easily computed since histograms are discrete

Histogram Matching $T: y \mapsto y'$

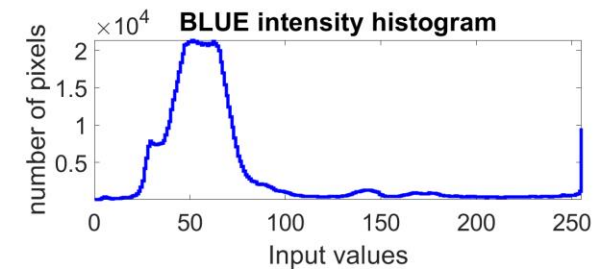
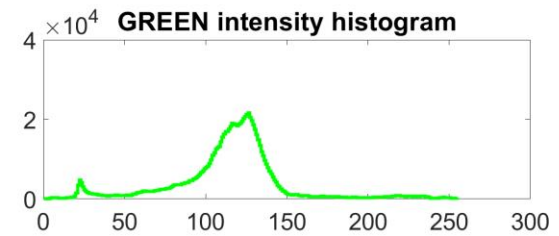
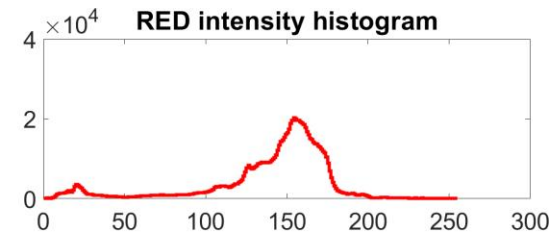


Hist. Matching

UNDEREXPOSED image

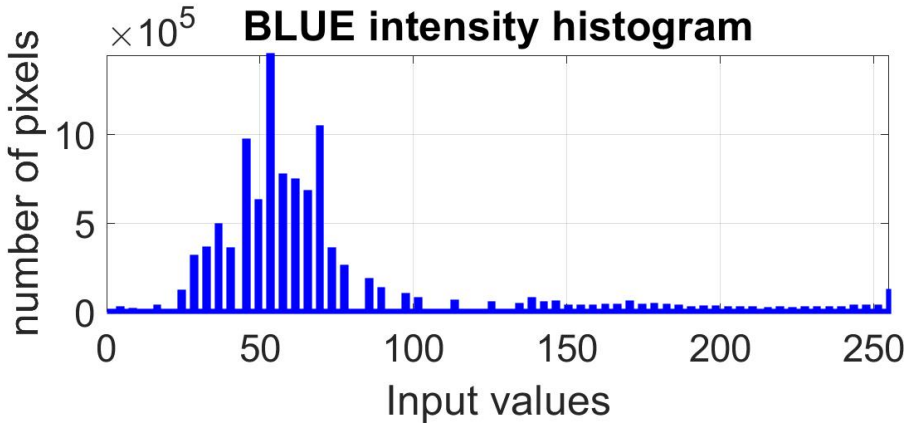
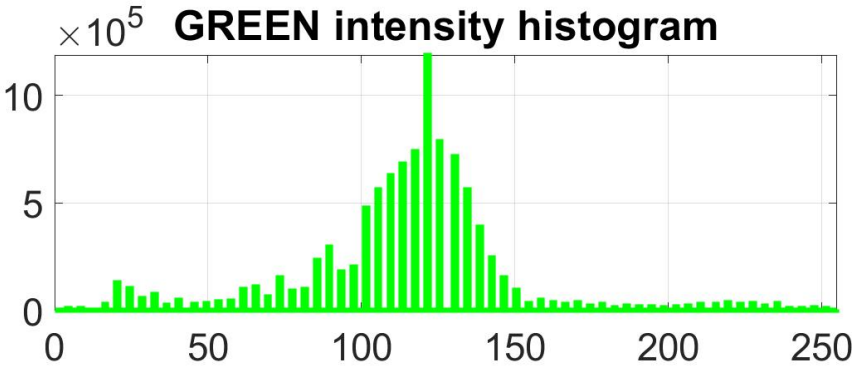
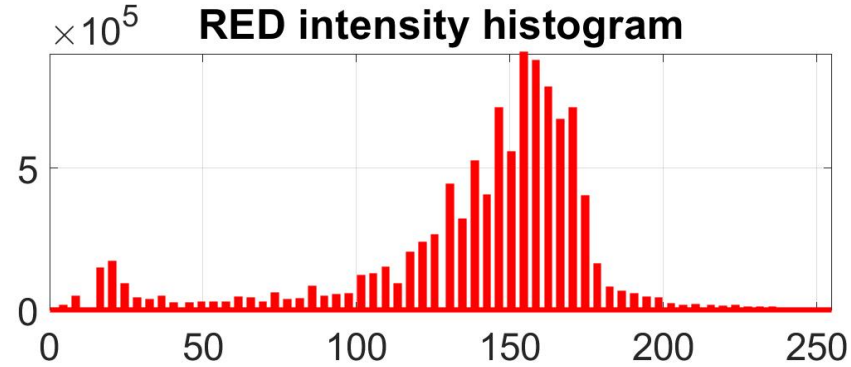


NORMAL image



Histogram Matching Results

UNDERPOSED TO NORMAL image



NORMAL



UNDEREXPOSED TO NORMAL



UNDEREXPOSED



Local (Spatial) Transformations: Correlation and Convolution

Local (Spatial) Transformation

In general, these can be written as

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

Where

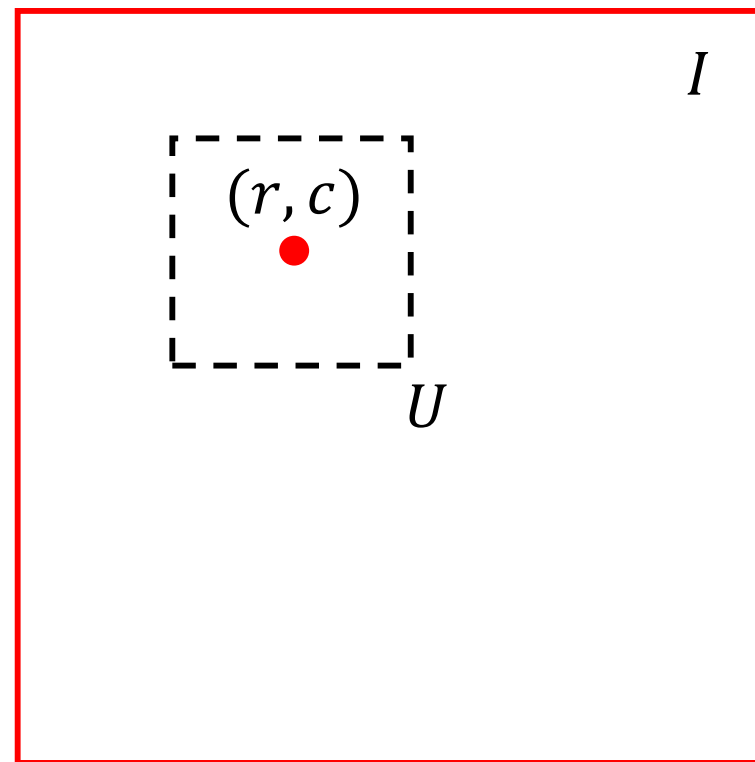
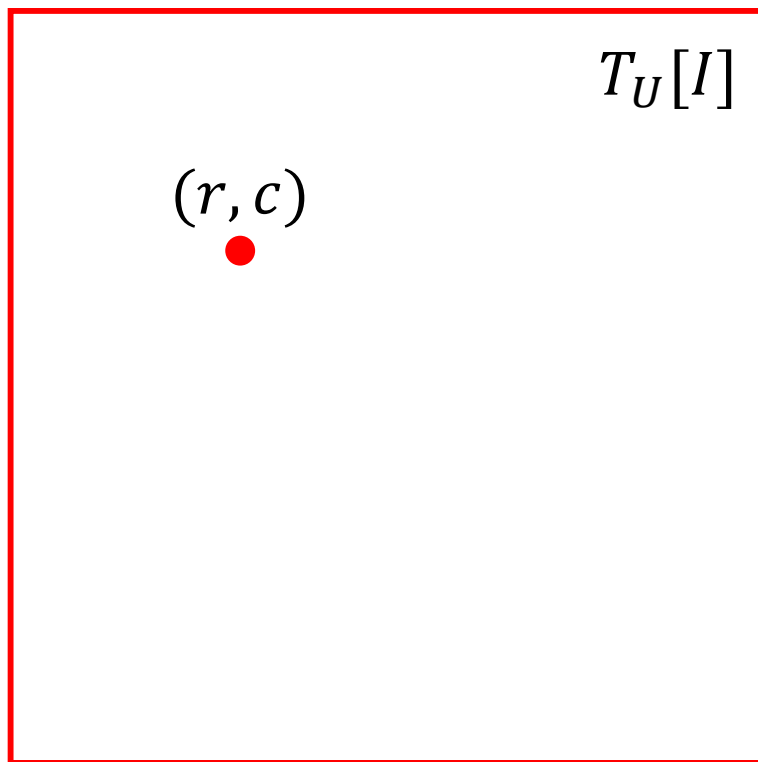
- I is the input image to be transformed
- G is the output
- U is a neighbourhood, identifies a region of the image that will concur in the output definition
- $T_U: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ or $T_U: \mathbb{R}^3 \rightarrow \mathbb{R}$ is a function

T operates on I “around” U

The output at pixel (r, c) i.e., $T_{U,(r,c)}[I]$ is defined by all the intensity values:
 $\{I(u, v), (u - r, v - c) \in U\}$

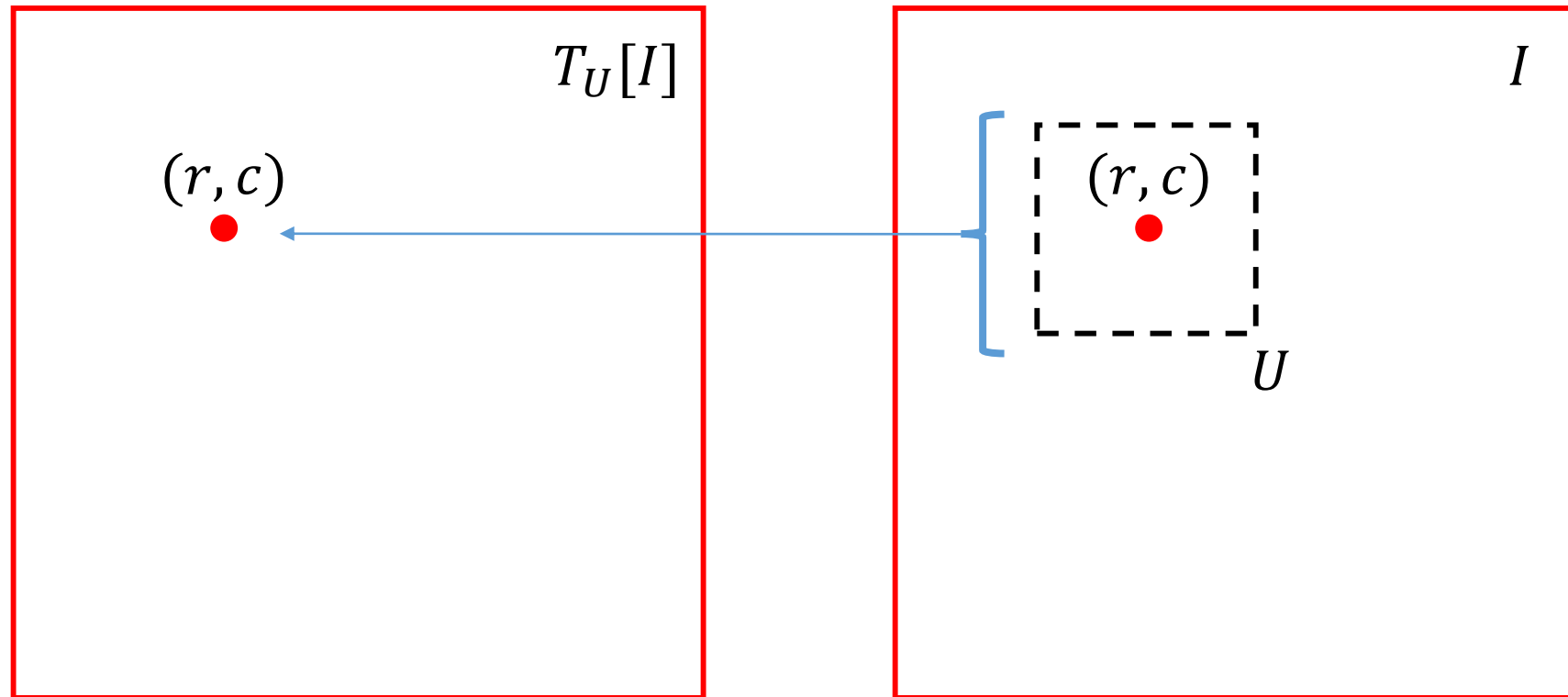
Local (Spatial) Filters

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



Local (Spatial) Filters

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



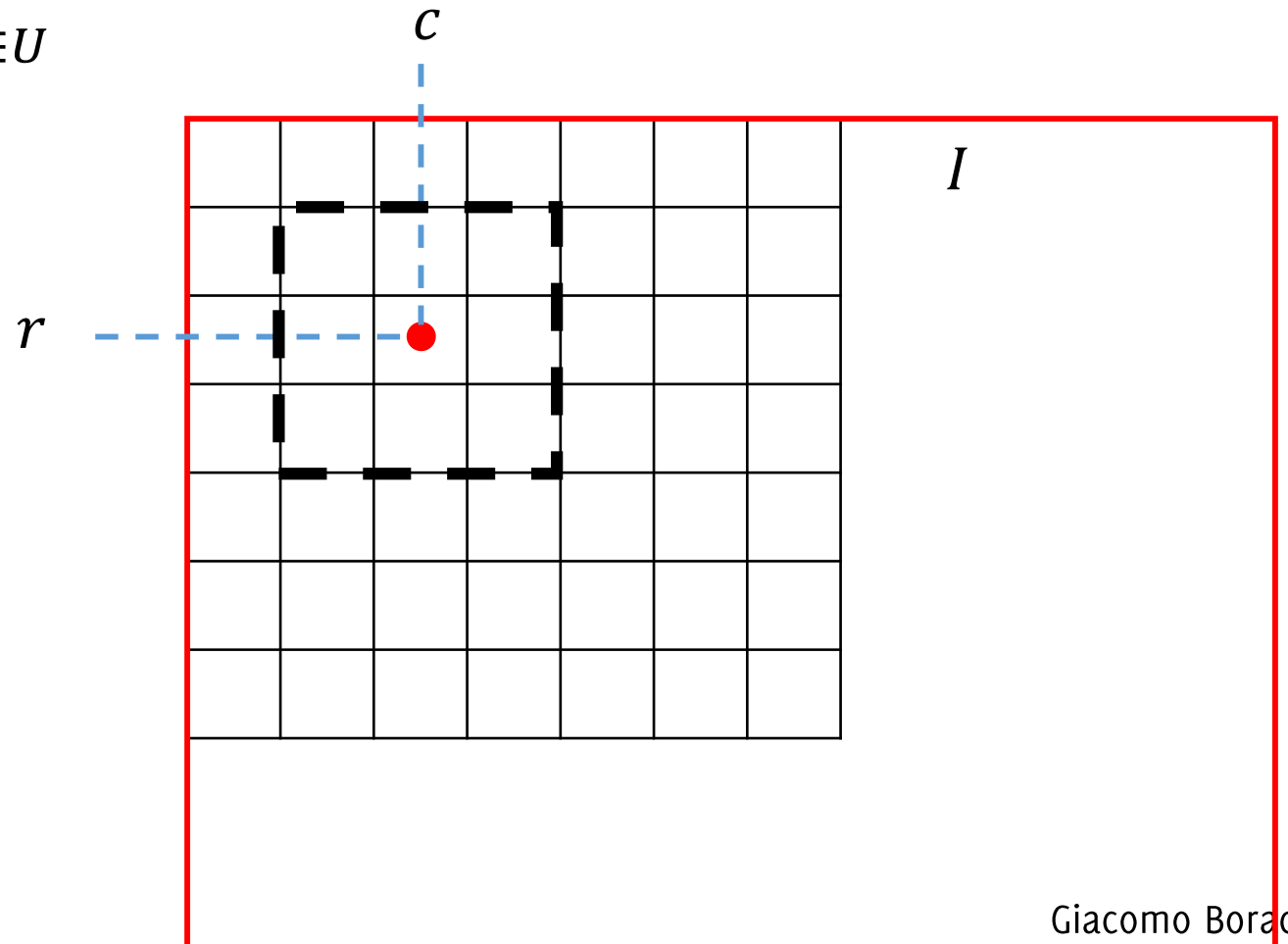
- The location of the output does not change
- The operation is repeated for each pixel
- T can be either linear or nonlinear

Local Linear Filters

Linear Transformation: Linearity implies that

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

Considering *some weights* $\{w_i\}$



Local Linear Filters

Linear Transformation: Linearity implies that

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

The diagram illustrates the local linear filter operation. On the left, a 3x3 neighborhood U is shown with weights $w(u, v)$ for $(u, v) \in U$. The weights are arranged in a 3x3 grid:

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

On the right, a 7x7 grid I is shown. A 3x3 neighborhood is highlighted by a red border. A red dot marks the center of the neighborhood, with dashed lines indicating coordinates r and c .

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

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

The diagram shows a 3x3 grid labeled U representing the filter kernel. The weights are arranged as follows:

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

The second diagram shows a grid labeled I representing the input image. A 3x3 neighborhood is highlighted with a red border. A red dot marks the center pixel at coordinates (r, c) . Dashed blue lines indicate the row r and column c .

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 $h = \{w_i\}$ 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 $(2L + 1) \times (2L + 1)$ and contains the weights defined before as w .

The filter is sometimes called “kernel”

Correlation for BINARY target matching

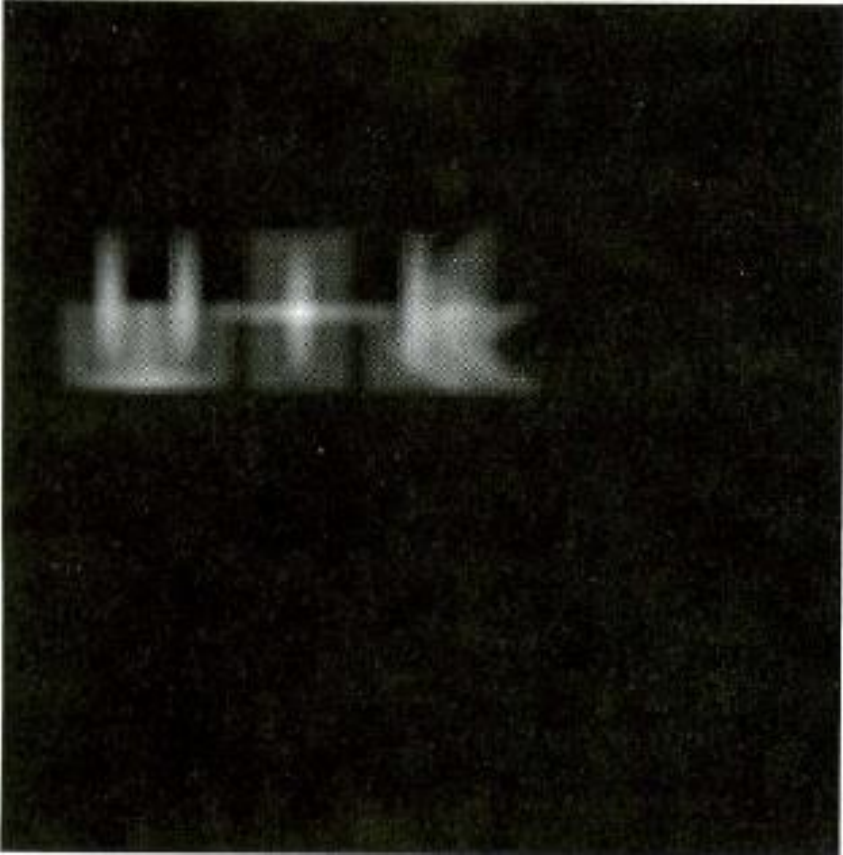


$$* \quad \text{T} \quad = \quad ?$$

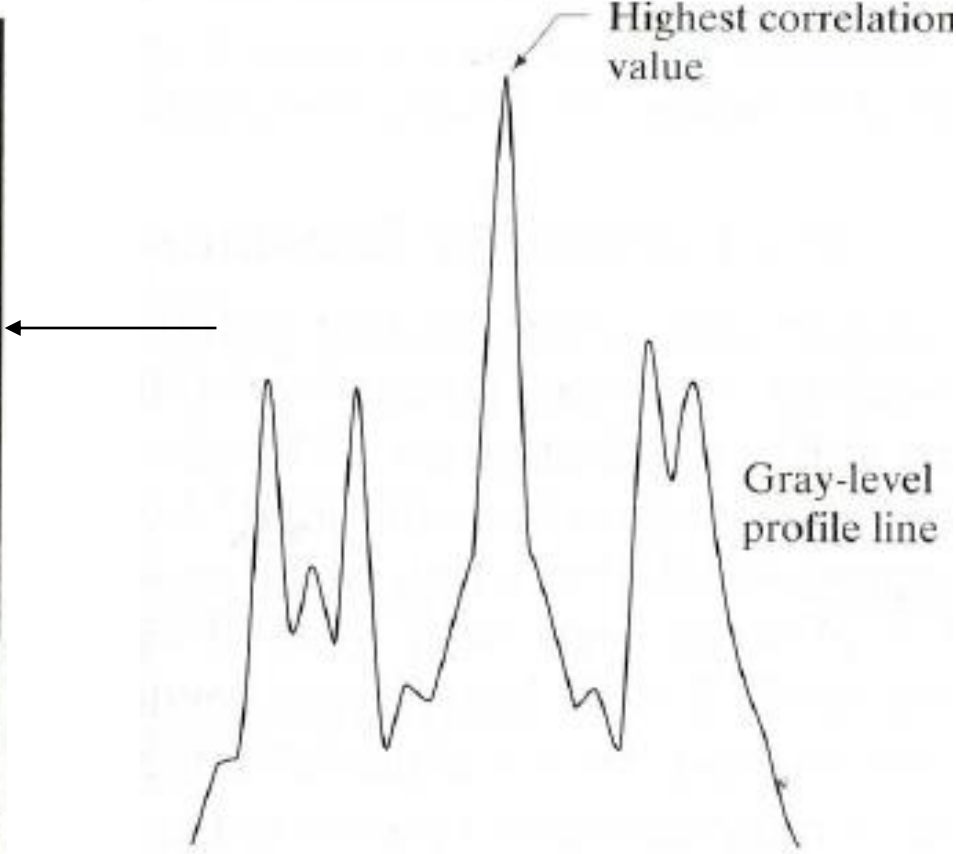
Easier with binary images

Target used as a filter

Correlation for target matching



Correlation function



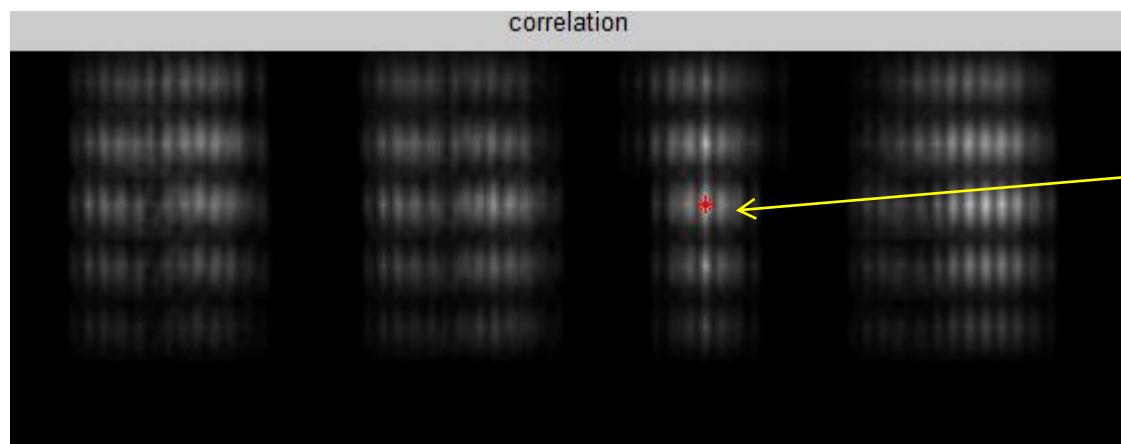
Maximum value line profile

Another example

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

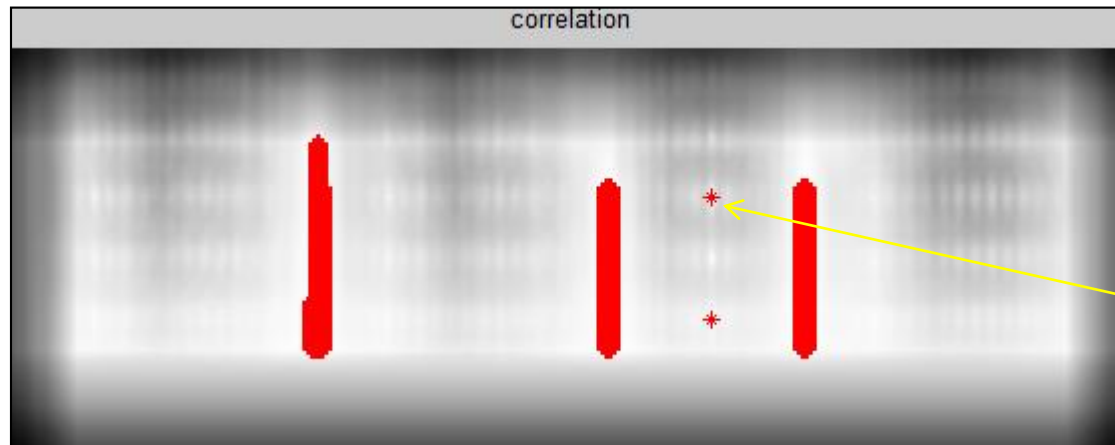
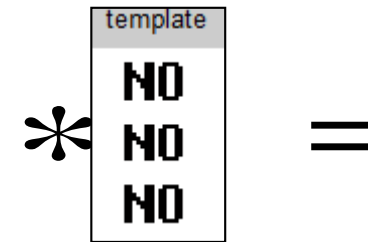
$$* \begin{matrix} \text{template} \\ \text{NO} \\ \text{NO} \\ \text{NO} \end{matrix} =$$



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



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

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

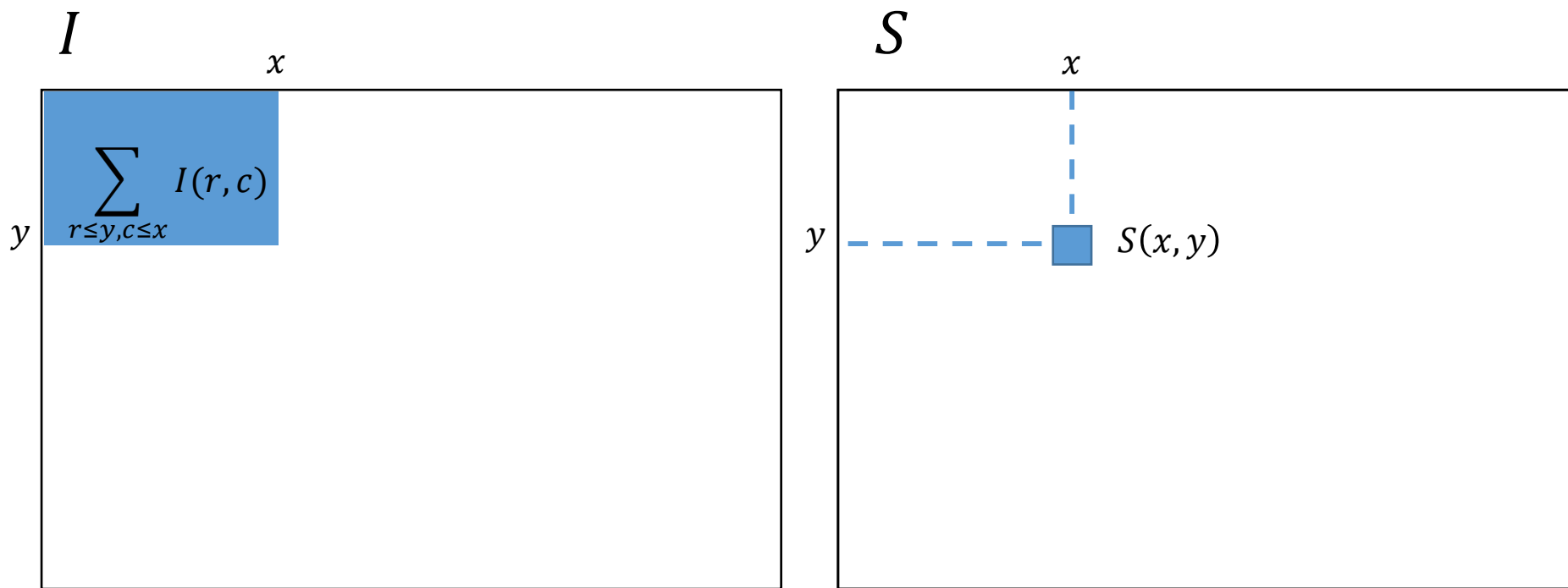
Remarks:

- 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

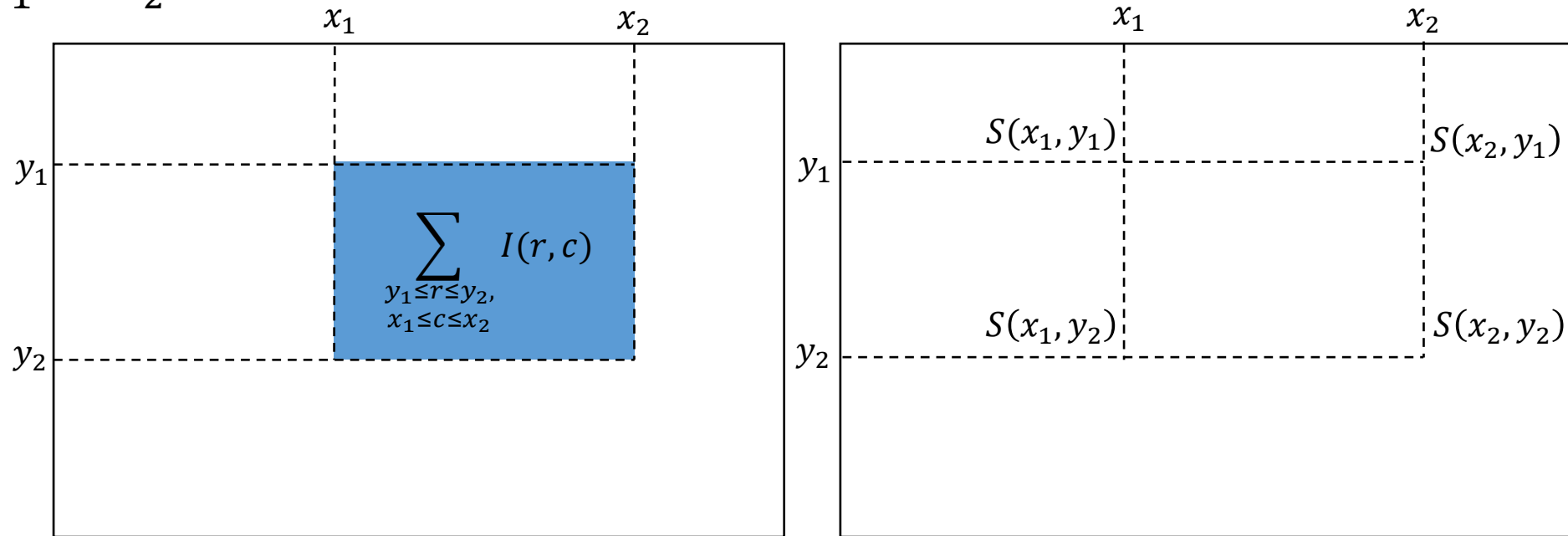
$$S(x, y) = \sum_{r \leq y, c \leq x} I(r, c)$$



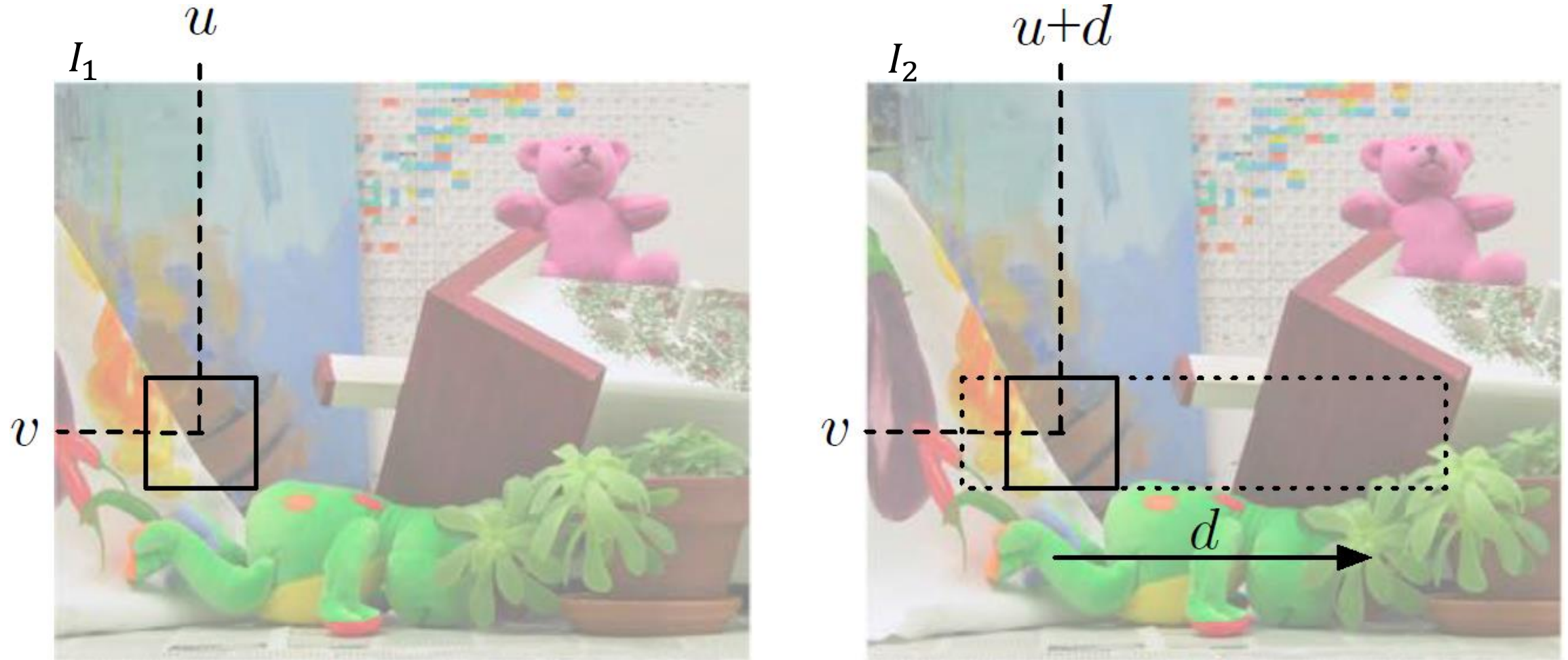
Using the Integral Image

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

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

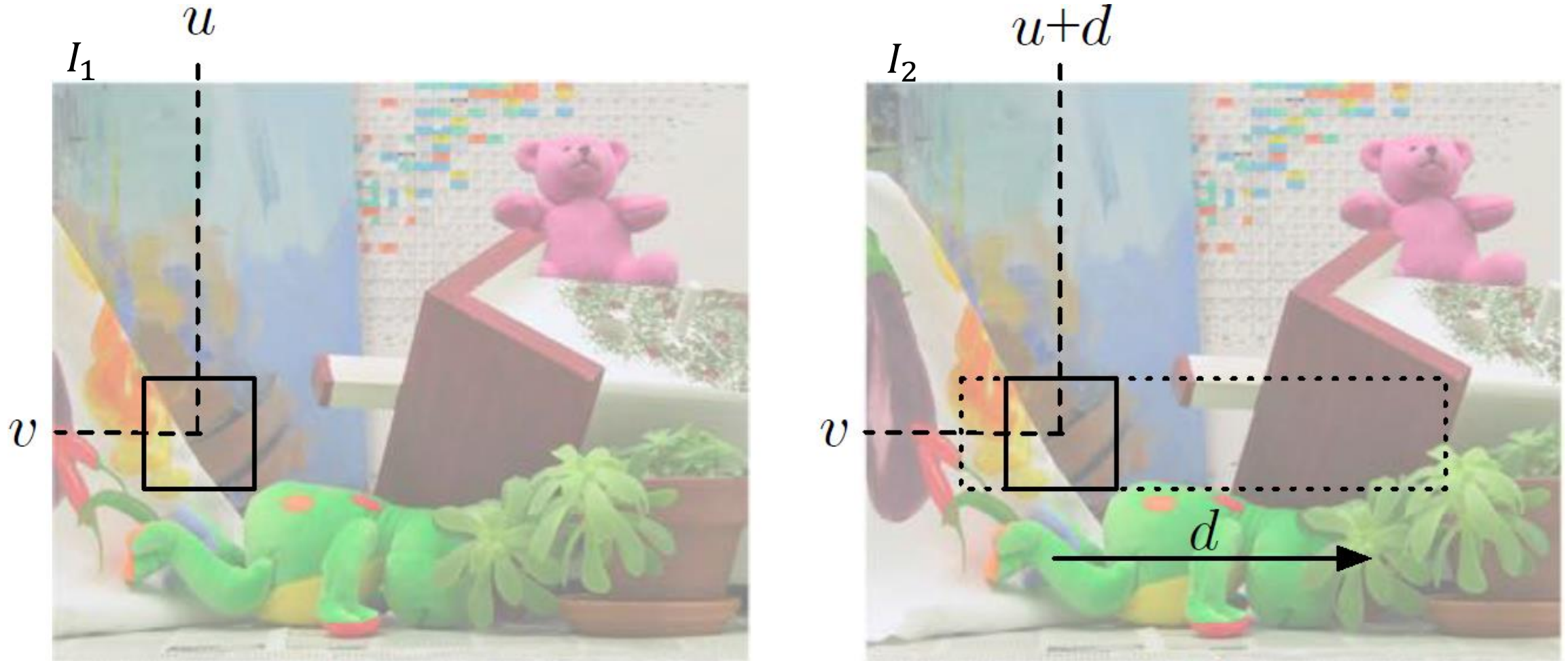


Disparity Map Estimation



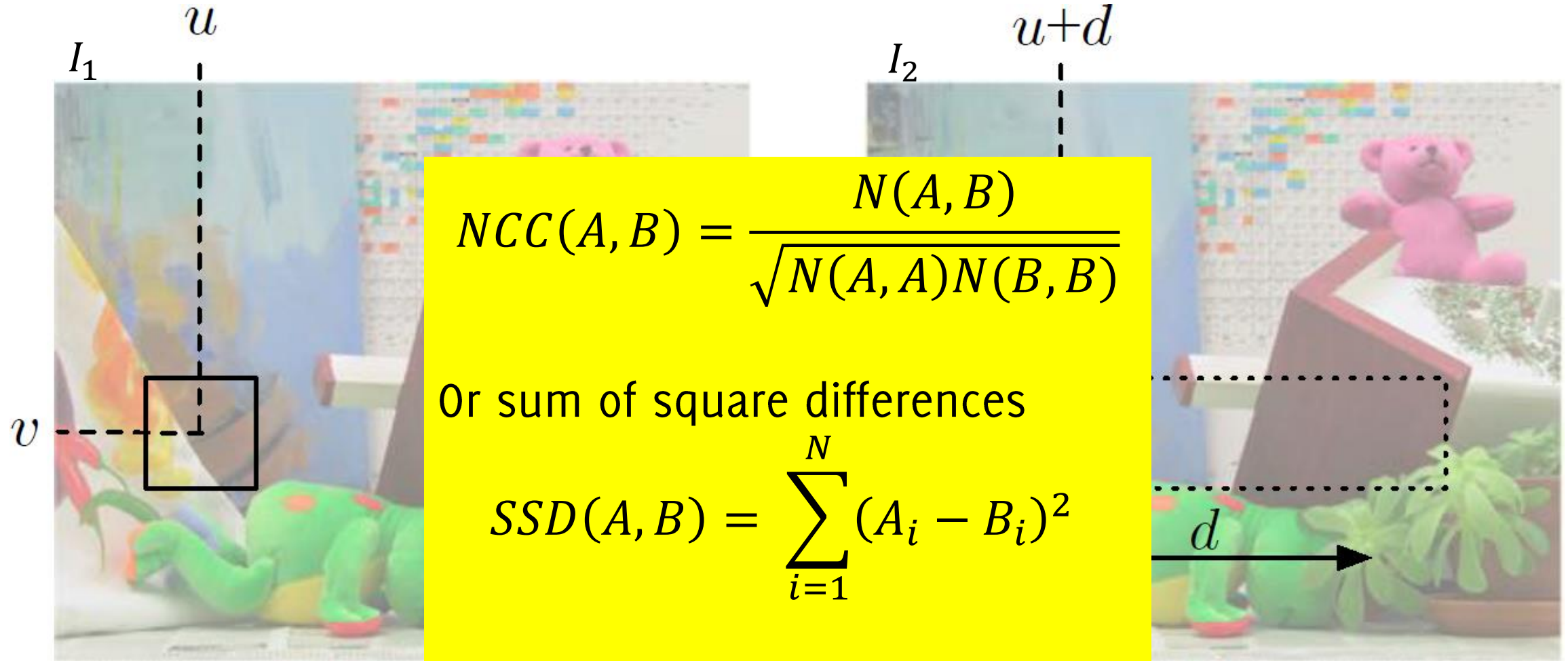
Disparity Map Estimation

There are different measures to compare a patch in I_1 with all the candidate matches in I_2



Disparity Map Estimation

There are different measures to compare a patch in I_1 with all the candidate matches in I_2



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

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

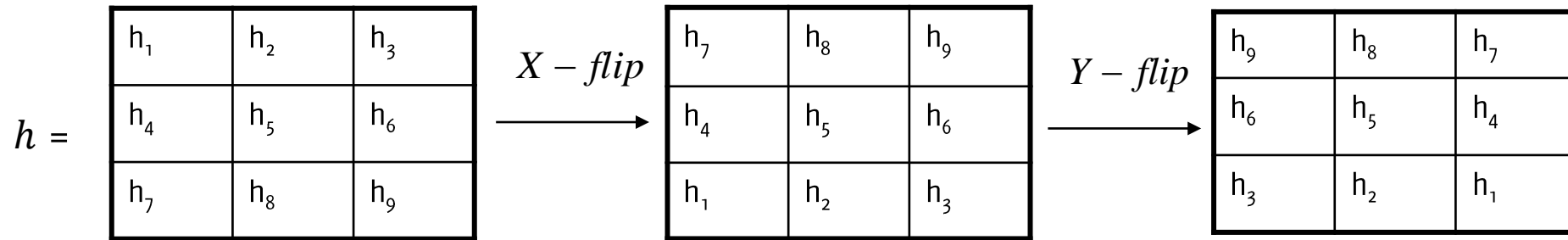
where the filter h is of size $(2L + 1) \times (2L + 1)$

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

Convolution – and filter flip

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

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



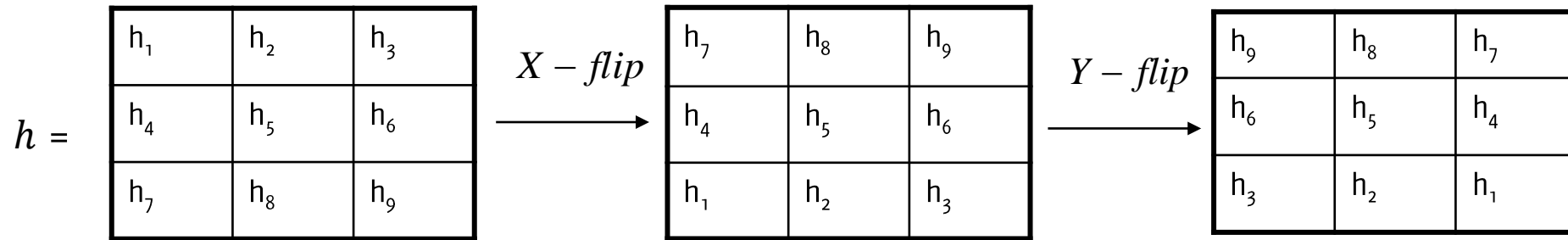
In this particular case $L = 1$ and both the image and the filter have size 3×3

The convolution is evaluated at $(r, c) = (0, 0)$

Convolution – and filter flip

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

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



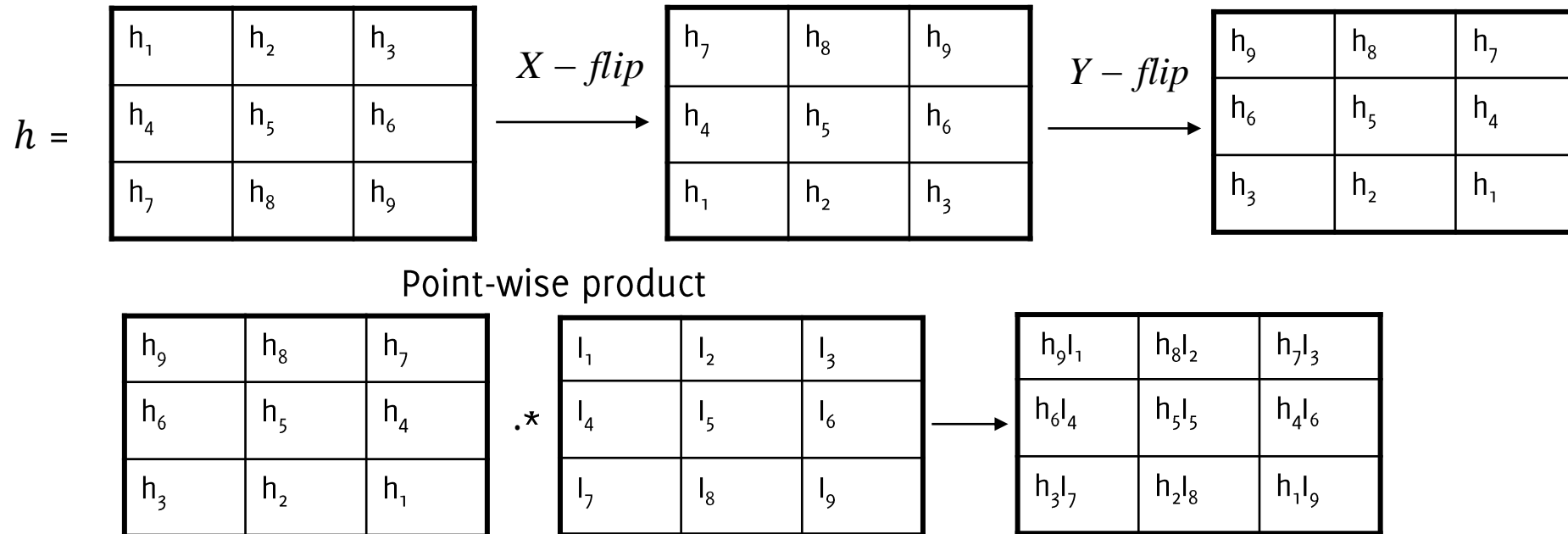
In this particular case $L = 1$ and both the image and the filter have size 3×3

The convolution is evaluated at $(r, c) = (0, 0)$

Convolution – and filter flip

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

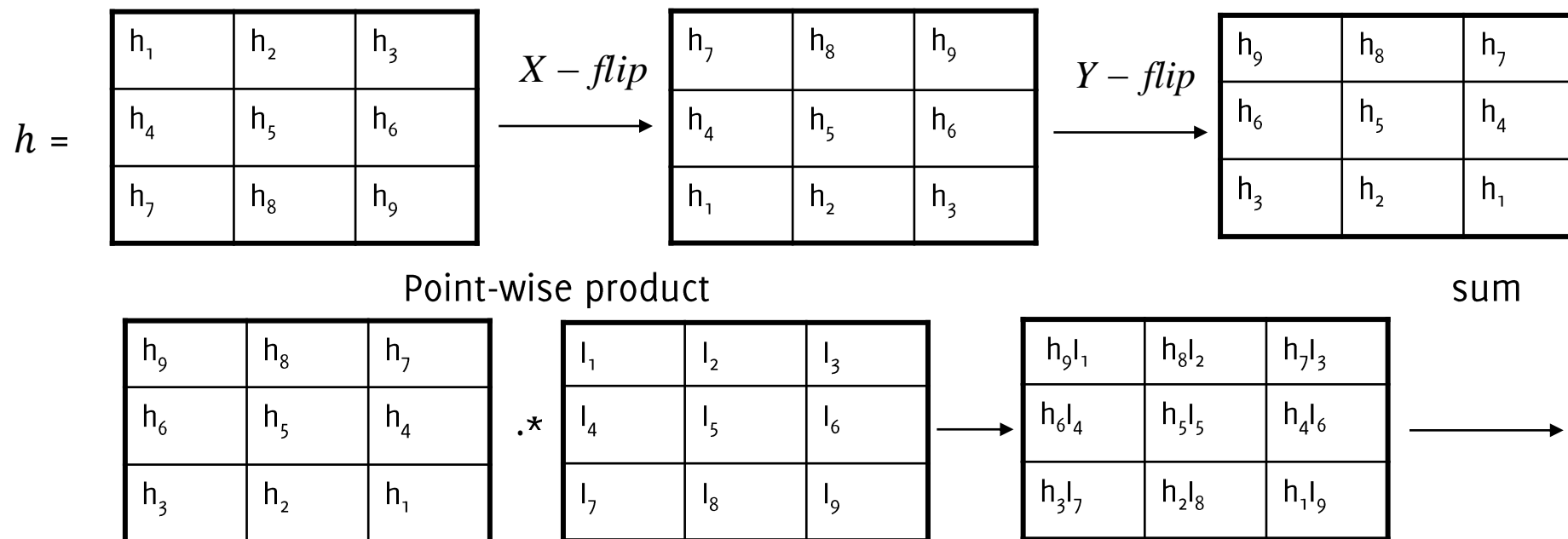
$$G(r, c) = (I \circledast h)(r, c) = \sum_{u=-L}^L \sum_{v=-L}^L I(r + u, c + v) h(-u, -v)$$



Convolution

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

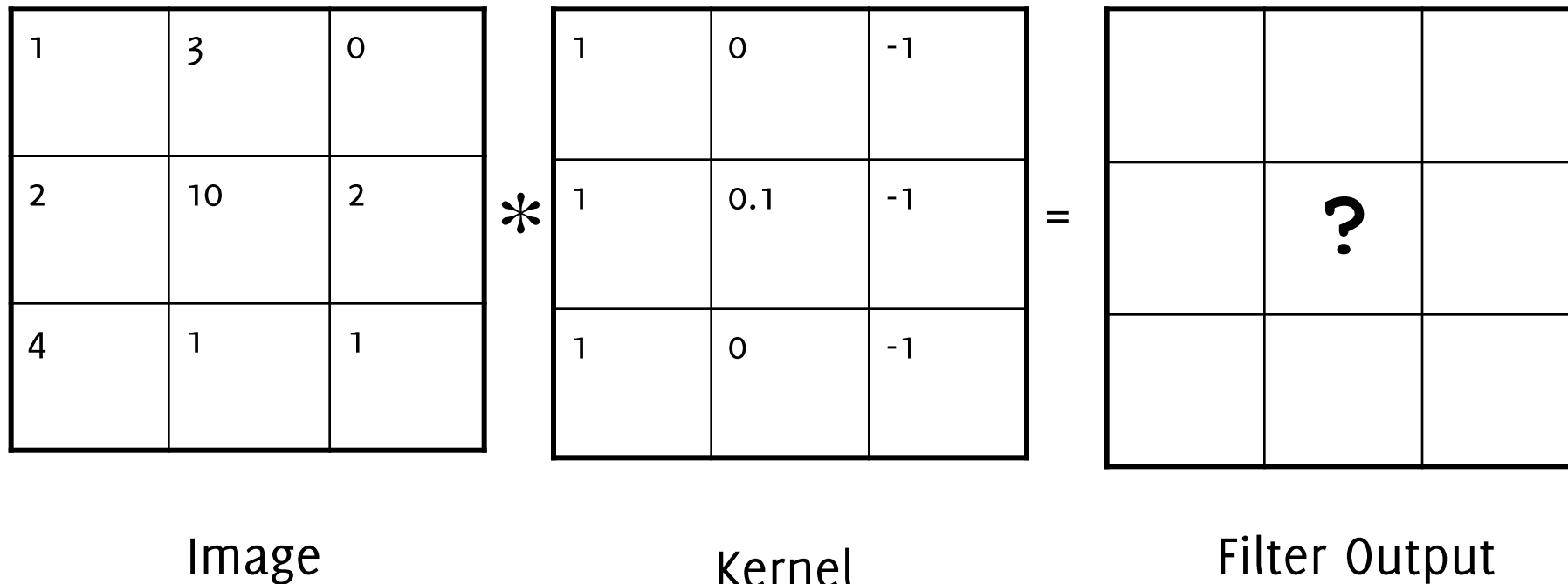
$$G(r, c) = (I \circledast h)(r, c) = \sum_{u=-L}^L \sum_{v=-L}^L I(r + u, c + v)h(-u, -v)$$



$$G_5 = h_9l_1 + h_8l_2 + h_7l_3 + h_6l_4 + h_5l_5 + h_5l_6 + h_3l_7 + h_2l_8 + h_1l_9$$

Question

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



Let's have a look at 1D
convolution

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

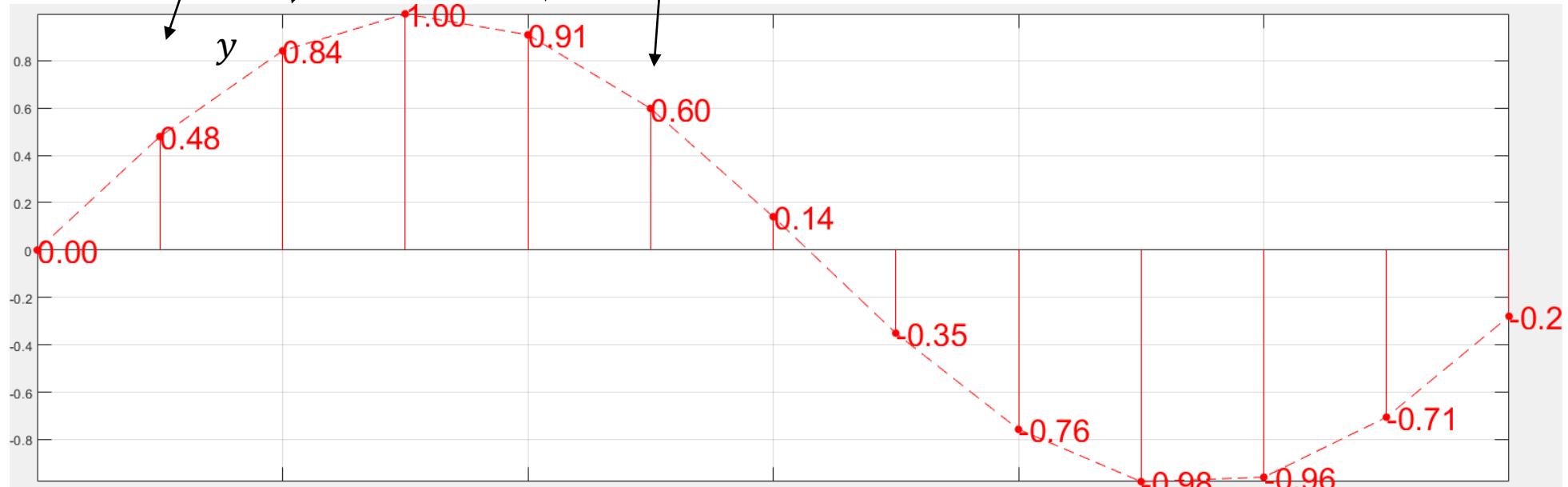
$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^L y(n - m)h(m)$$

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

1D Convolution - example

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

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

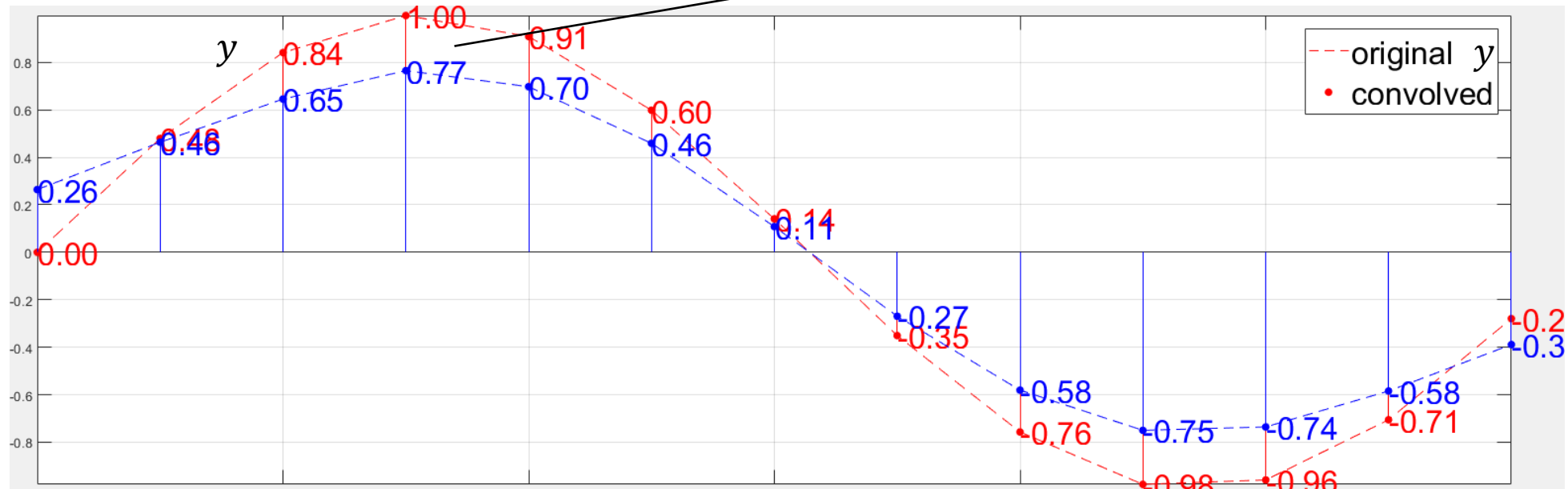


1D Convolution - example

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

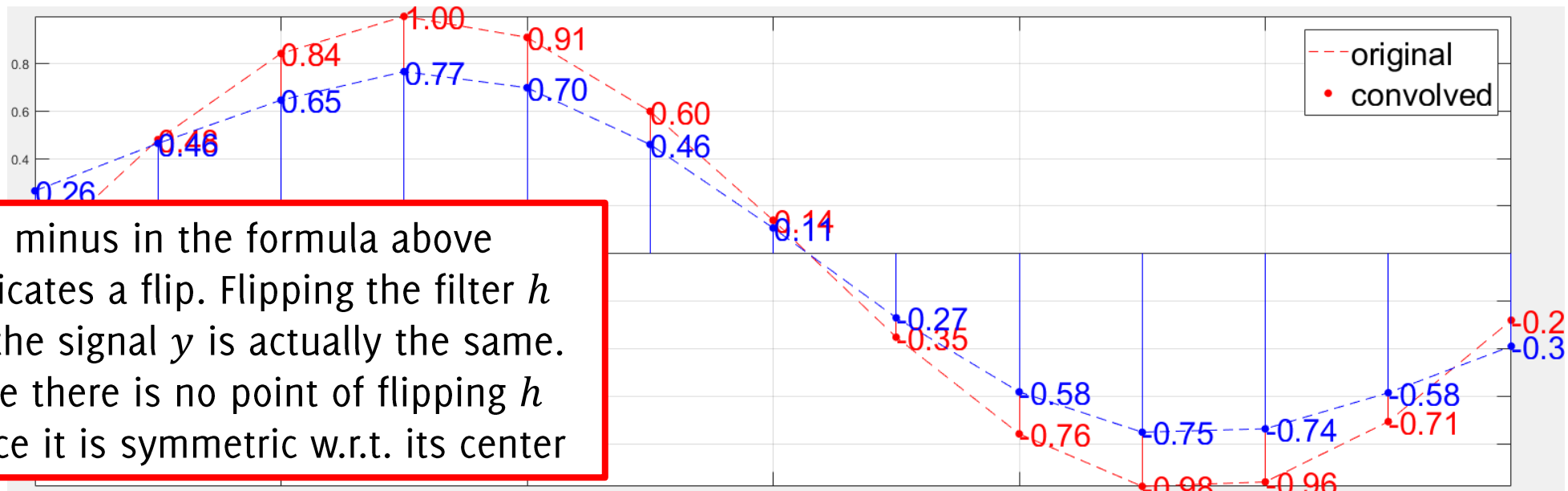
$$y = \sin(x), h = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5} \right], L = 2$$

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

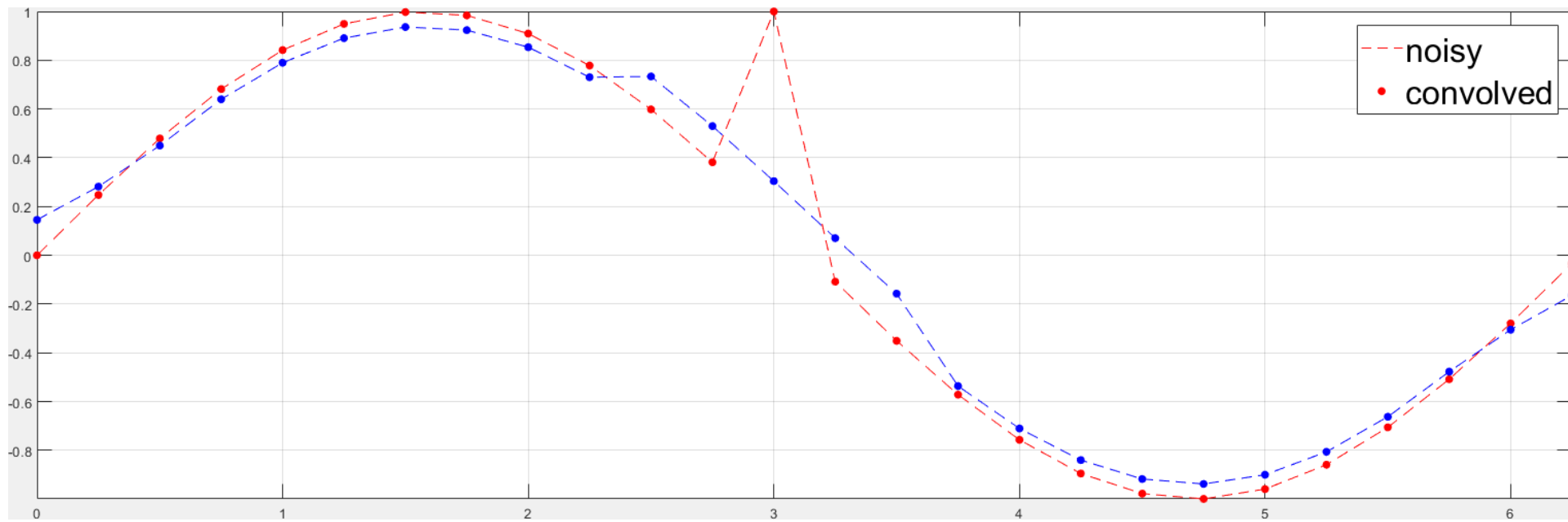


1D Convolution - example

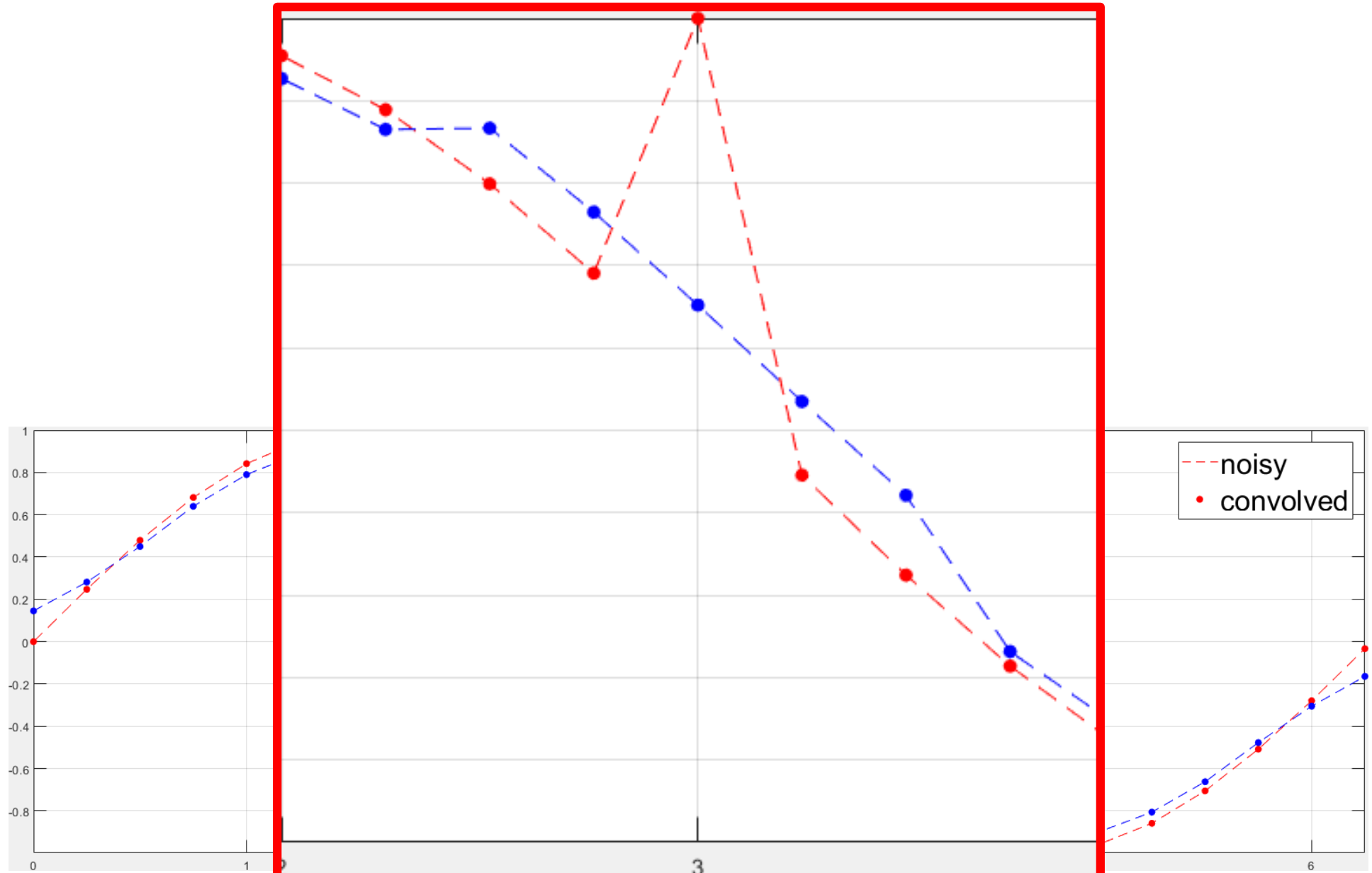
$$z(n) = (y \otimes h)(n) = \sum_{m=-L}^L y(n-m)h(m)$$
$$= \sum_{m=-L}^L y(n+m)h(-m)$$



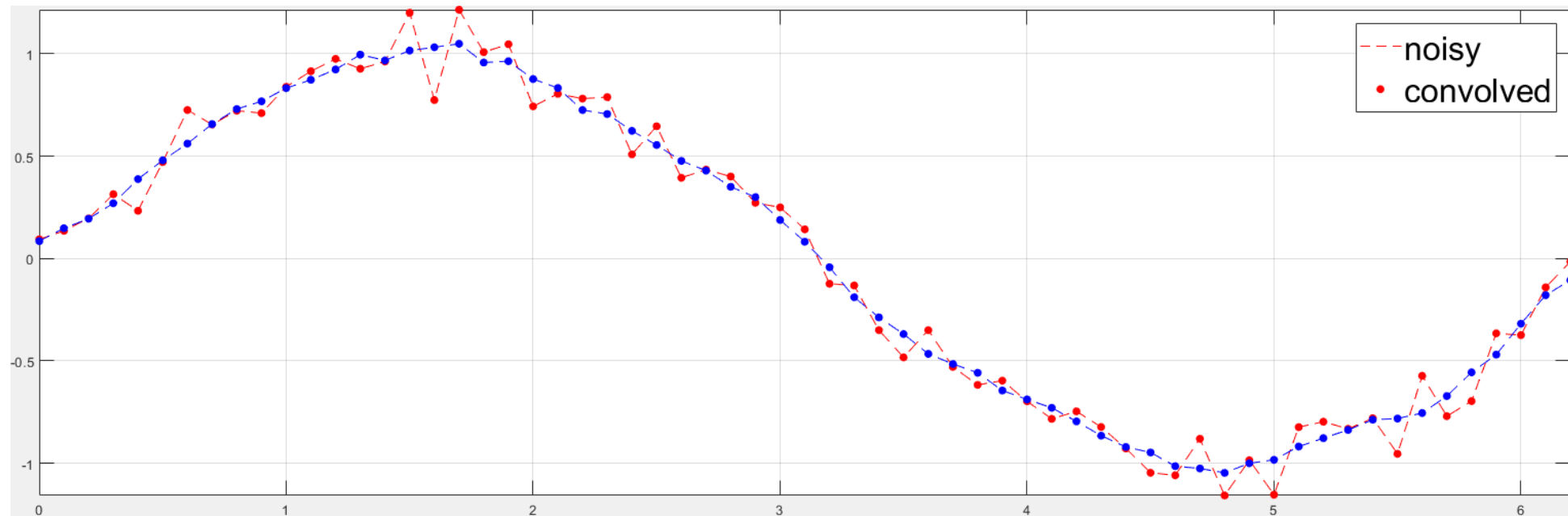
What about an imupulse?



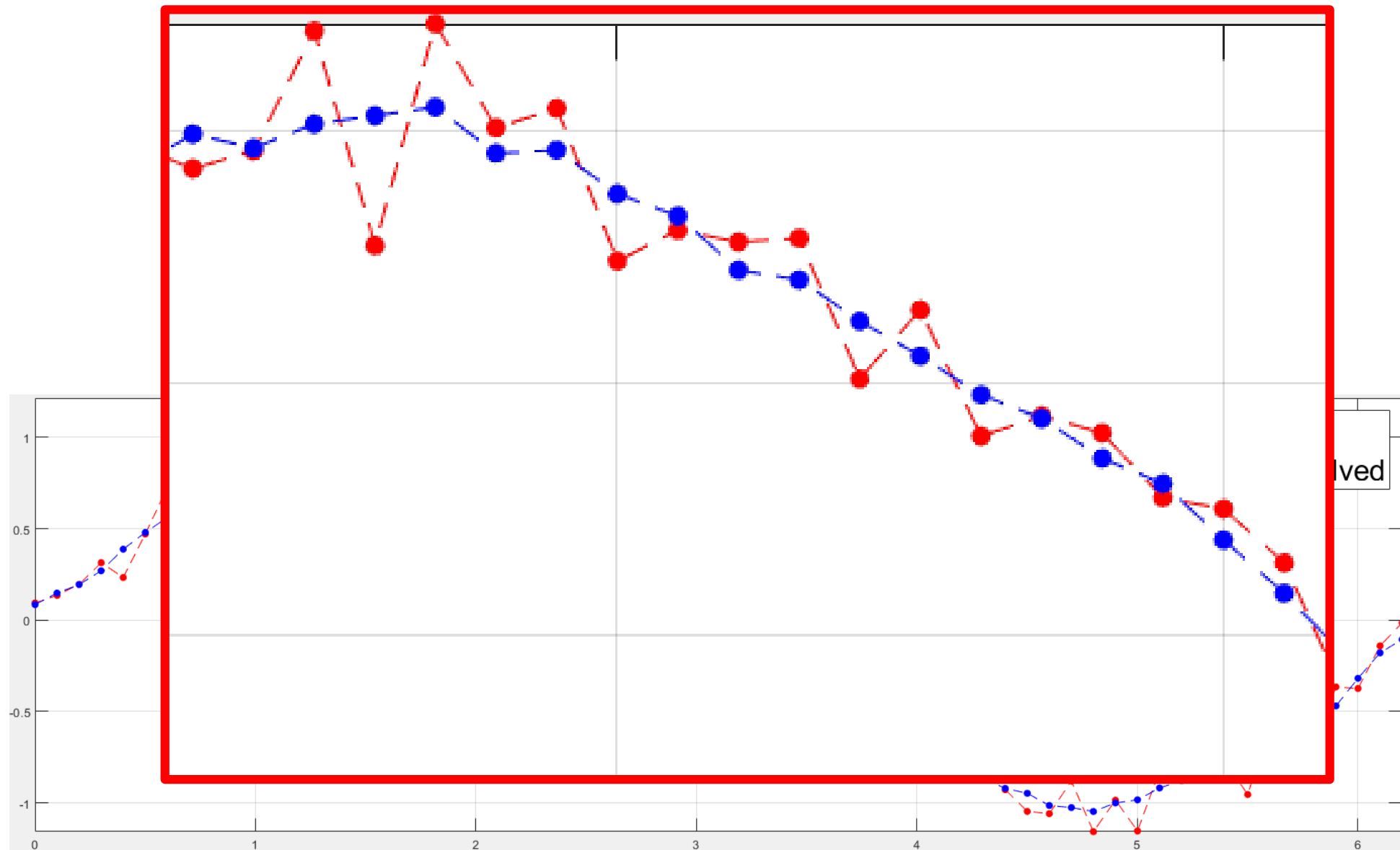
What about an impulse?



What about noise?



What about noise?

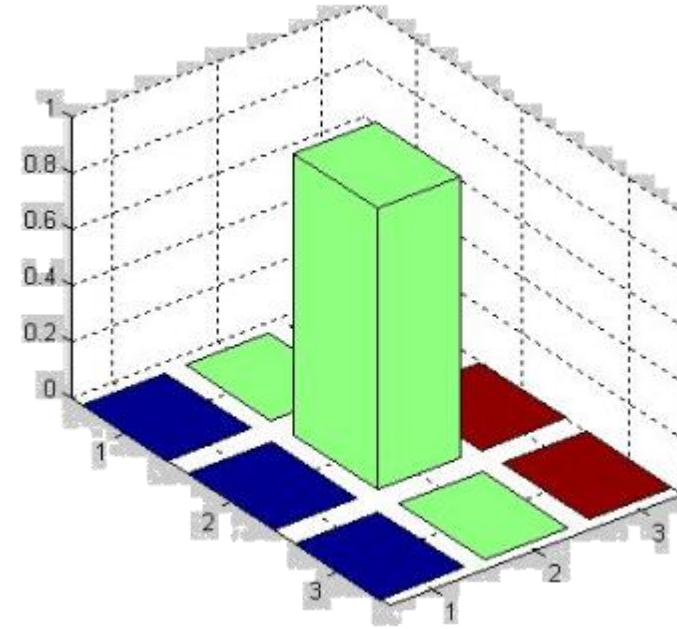


Let's go back to
2D convolution now

A well-known Test Image - Lena



A Trivial example



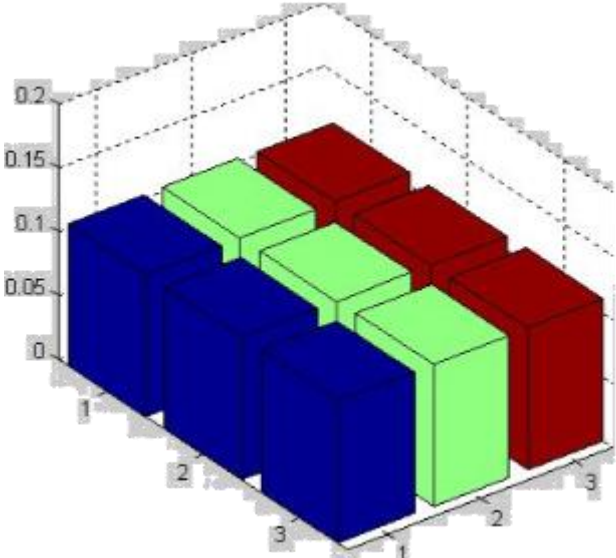
*

0	0	0
0	1	0
0	0	0

=



Linear Filtering



$$* \frac{1}{9}$$

1	1	1
1	1	1
1	1	1

=

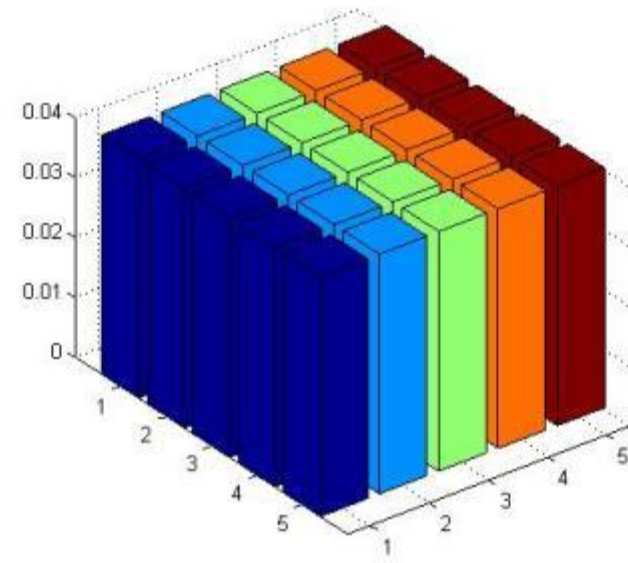
?

The original Lena image



Filtered Lena Image





$$* \frac{1}{25}$$

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

=

The original Lena image



The filtered Lena image



What about normalization?



$\ast \frac{2}{25}$

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



...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

=

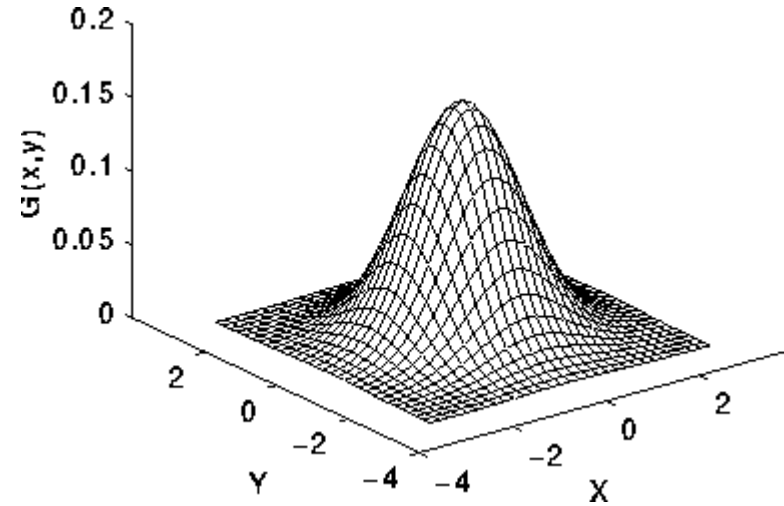
.. convolution is linear



2D Gaussian Filter

Continuous Function

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



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

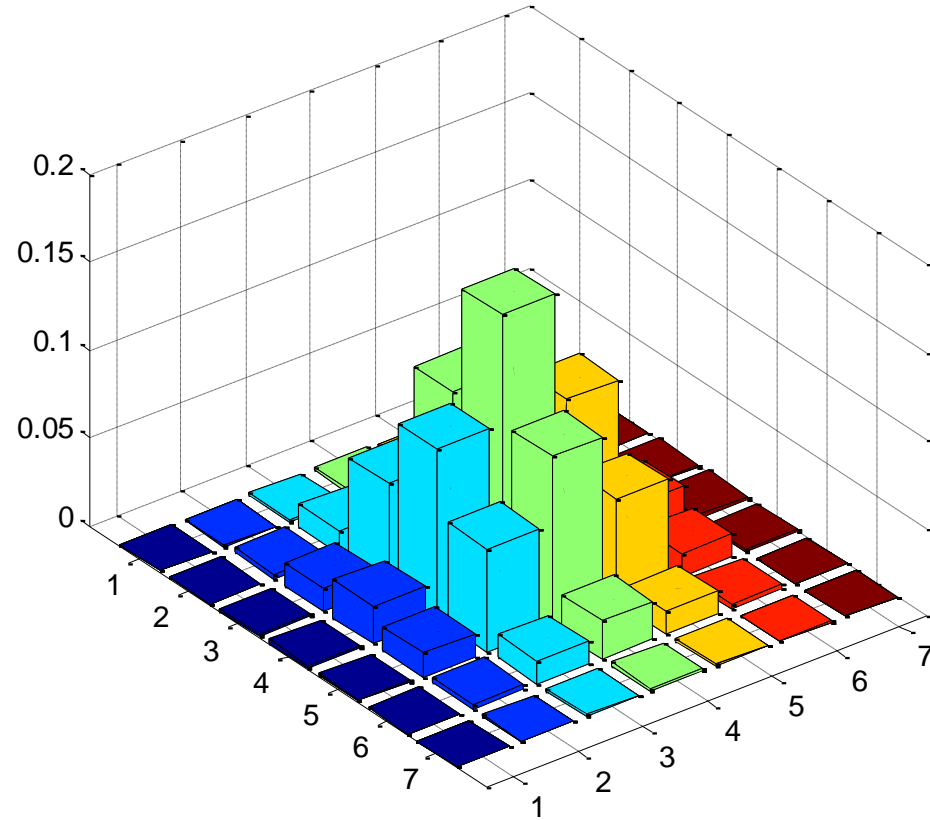
$$G(i, j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i^2 + j^2)}{2\sigma^2}\right)$$

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

Weighted local averaging filters: Gaussian Filter



*



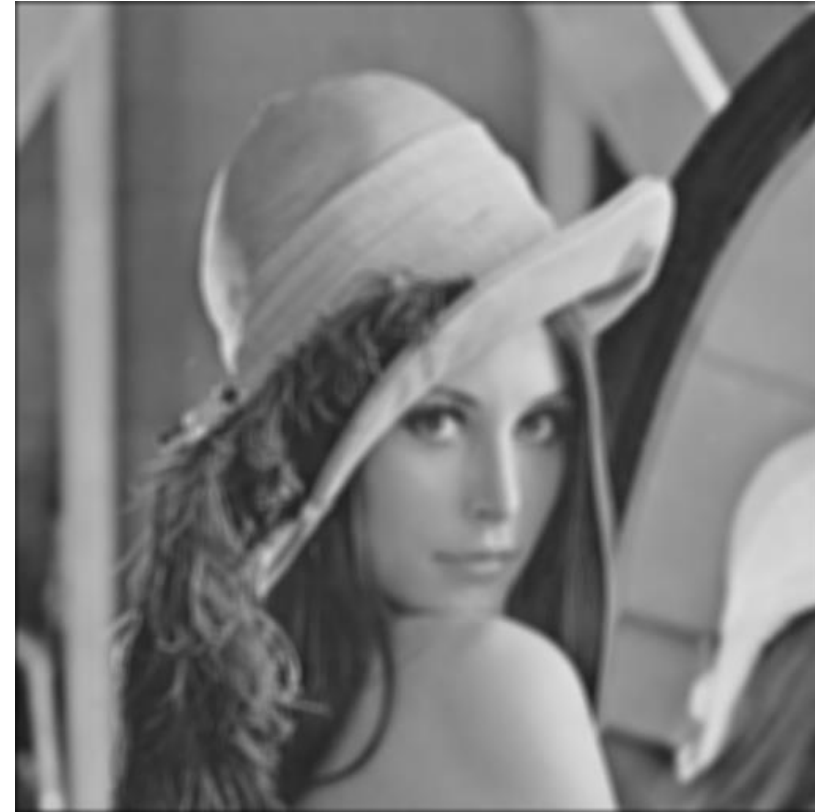
Weighted local averaging filters: Gaussian Filter



Gaussian Smoothing vs Averaging Filters



Gaussian Smoothing
Support 7×7



Smoothing by Averaging
On 7×7 window

Convolution Properties

Properties of Convolution: Commutative

It is a **linear operator**

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

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)

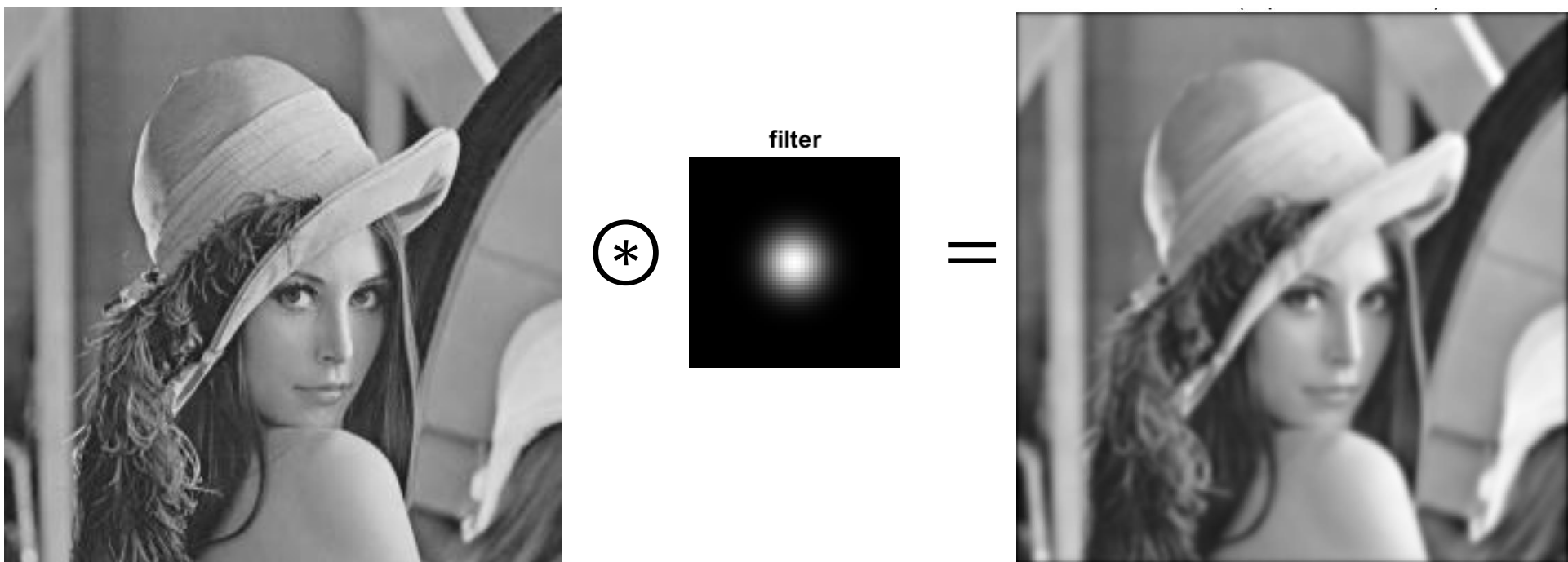
$$I_1 \circledast I_2 = I_2 \circledast I_1$$

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

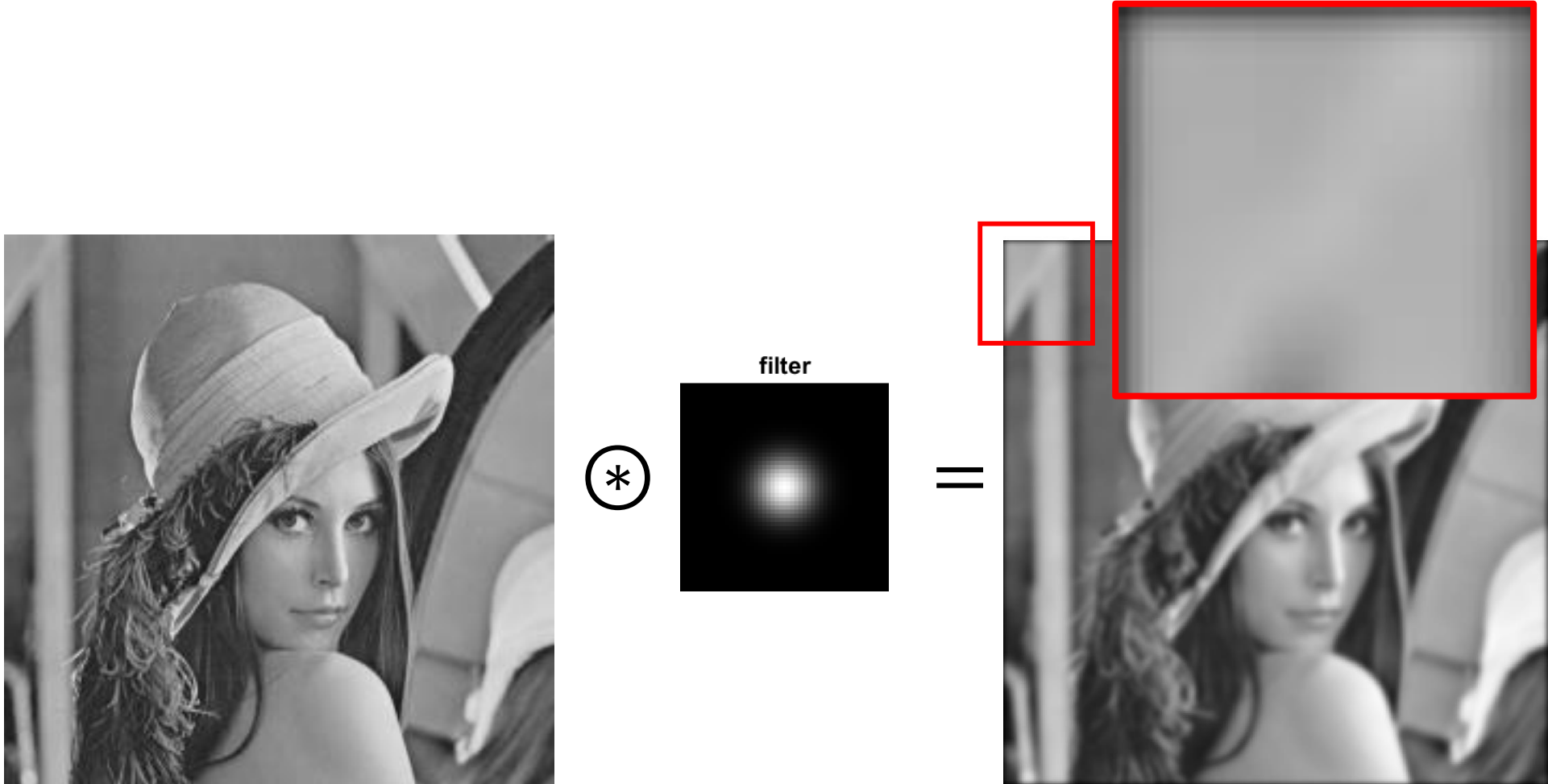
Input image I							filter h		
0	0	0	0	0	0	0	0	1	-1
0	1	0	2	1	0	0	-1	-1	0
0	1	1	1	0	1	0	1	-1	-1
0	1	2	1	0	1	0			
0	1	2	0	2	2	0			
0	1	0	1	0	0	0			
0	0	0	0	0	0	0			

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

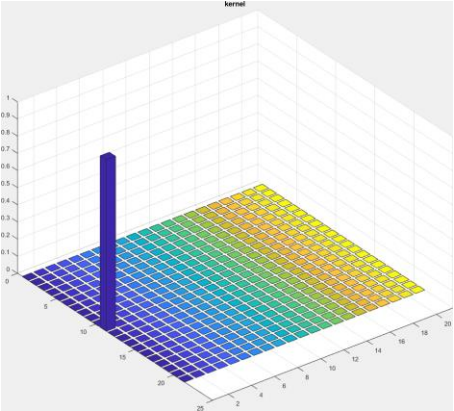
Is Convolution Commutative?



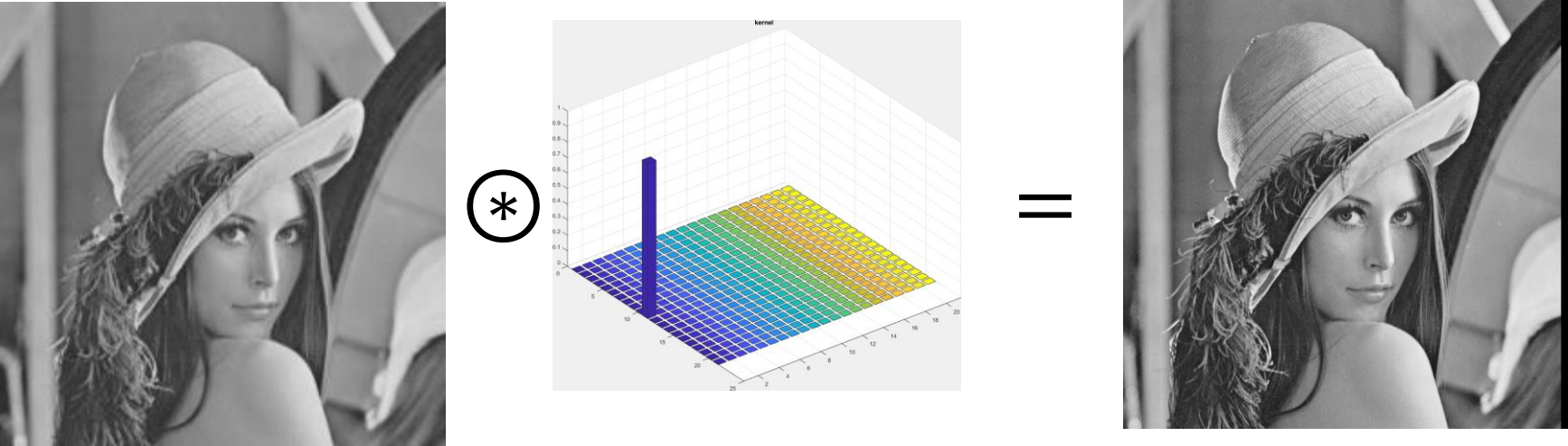
Is Convolution Commutative?



Translation



Translation



Remember the filter has to be flipped before convolution

Is Convolution Commutative?



Properties of Convolution: Associative

It is also **associative**

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

and **distributive**

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

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**

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

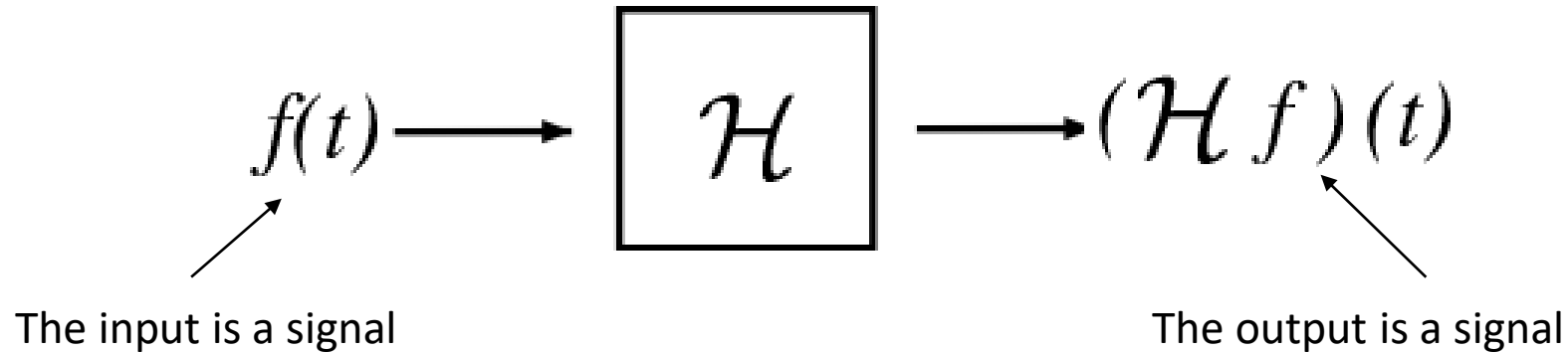
It is **shift-invariant**, namely

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

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

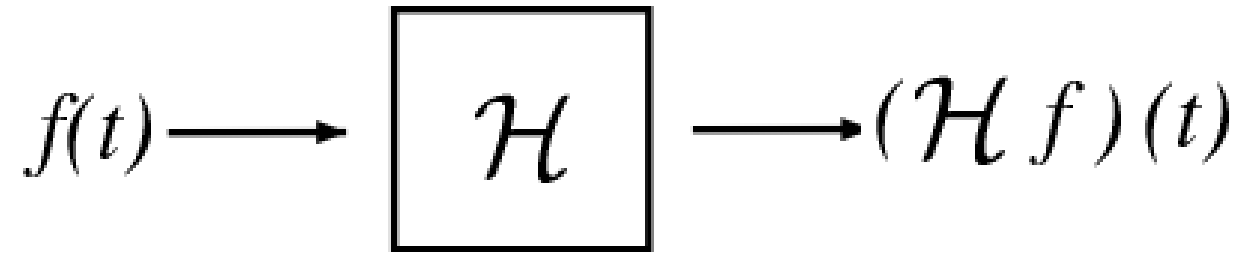
Systems

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

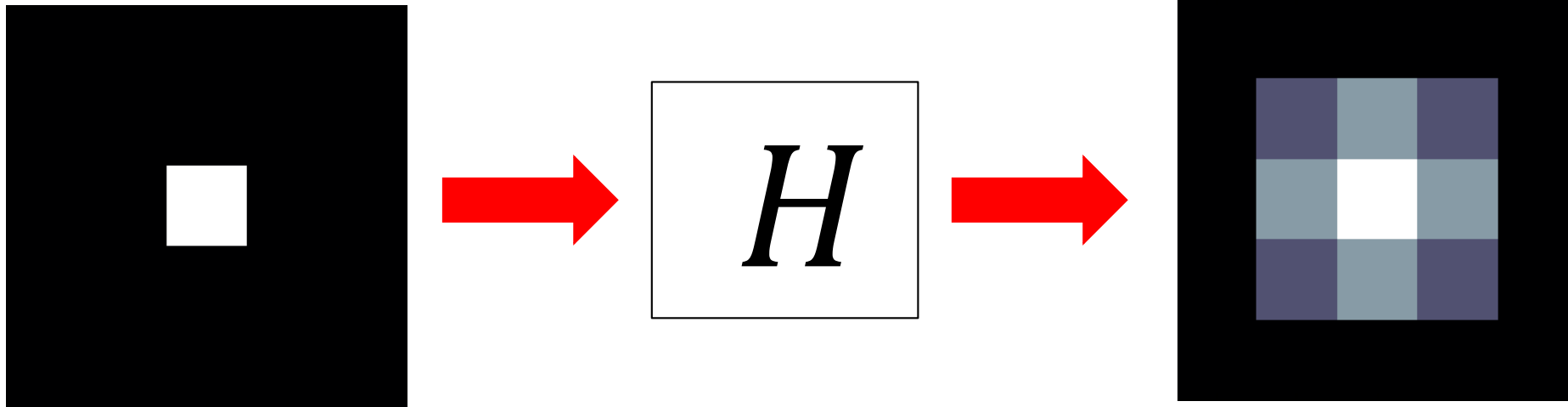
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

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

Linear and Time Invariant Systems

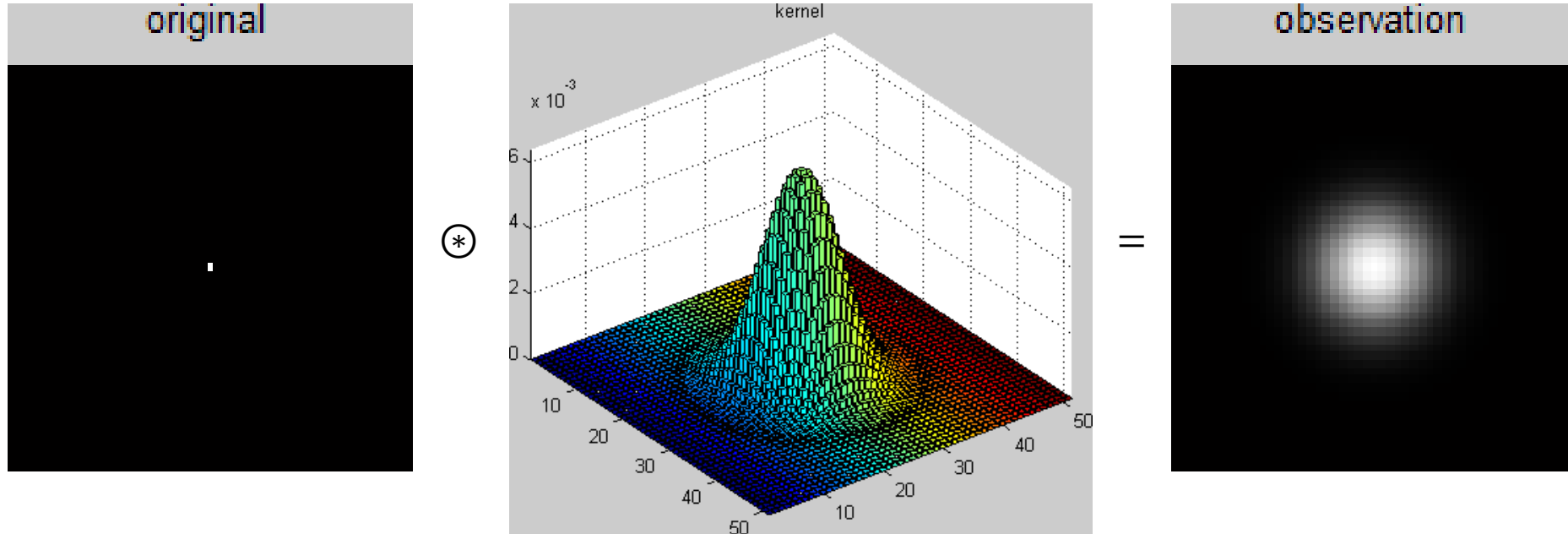
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

- LTI systems are **characterized** entirely by a **single function**, 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

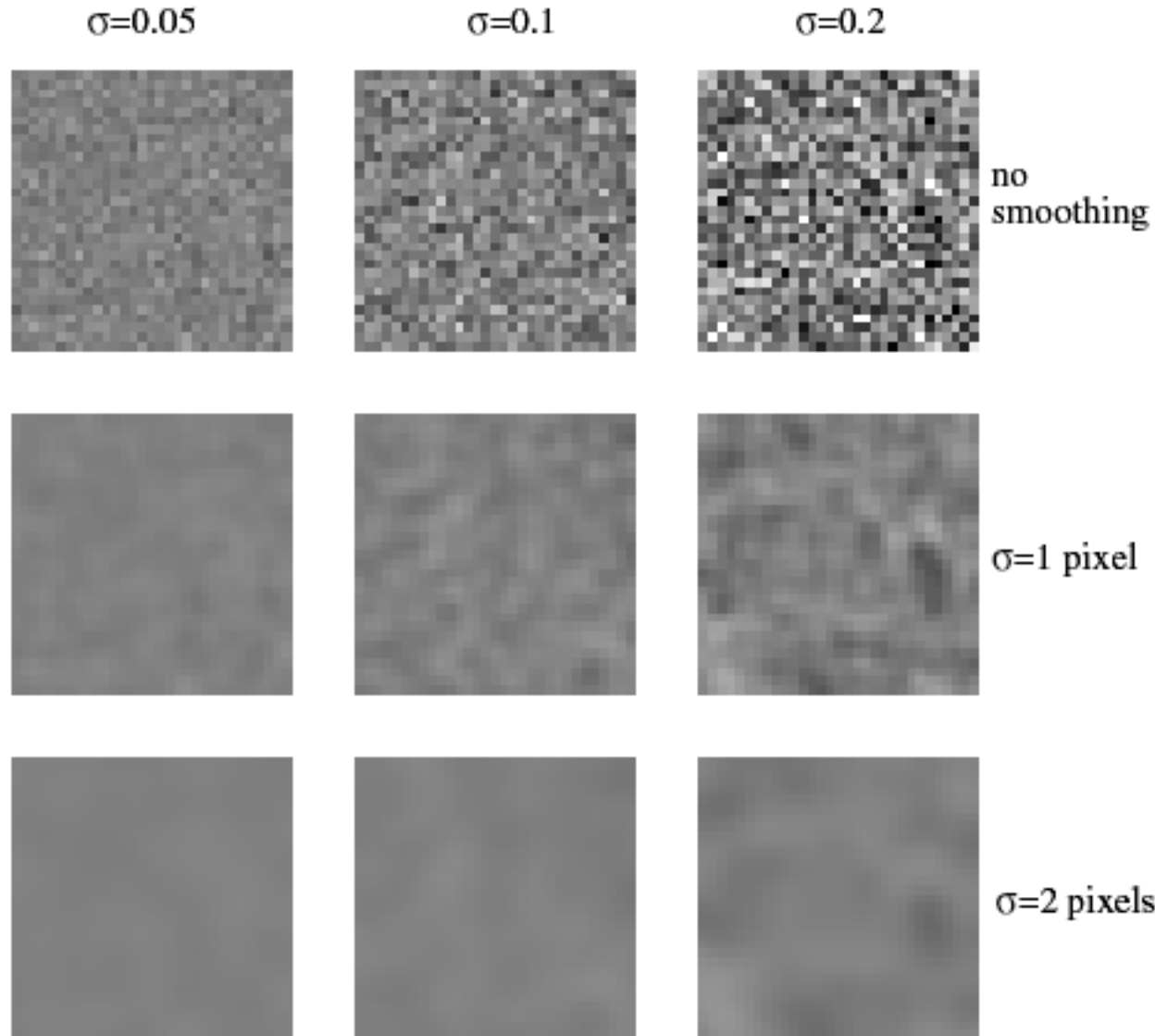


This is why h is also called the “Point Spread Function”

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

$$z(x) = y(x) + \eta(x), \quad x \in \mathcal{X}$$

Where

- x denotes the pixel coordinates in the domain $\mathcal{X} \subset \mathbb{Z}^2$
- y is the original (noise-free and unknown) image
- z is the noisy observation
- η is the noise realization

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x), \quad x \in \mathcal{X}$$

The goal is to compute \hat{y} *realistic* estimate of y , given z and the distribution of η .

For the sake of simplicity we assume AWG: $\eta \sim N(0, \sigma^2)$ and $\eta(x)$ independent realizations.

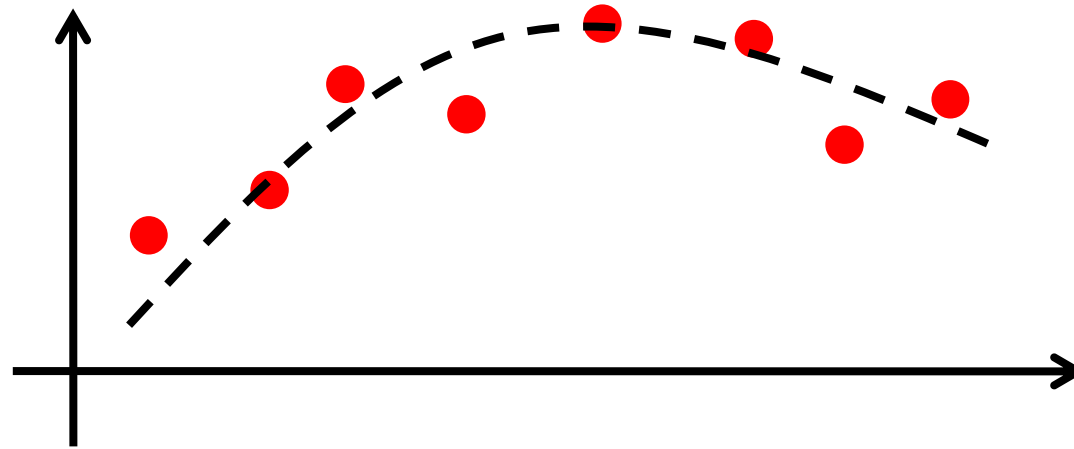
The noise standard deviation σ is also assumed as known.

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 that minimizes a weighted loss over noisy observations

$$\widehat{y}_h(x_0) = \operatorname{argmin}_C \sum_{x_s \in X} w_h(x_0 - x_s) (z(x_s) - C)^2$$

Where

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

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

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

Image Formation Model

Observation model is

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

thus 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

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

thus 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)

Non Parametric Approaches

- Local S
- Non Lc

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

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

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

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

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

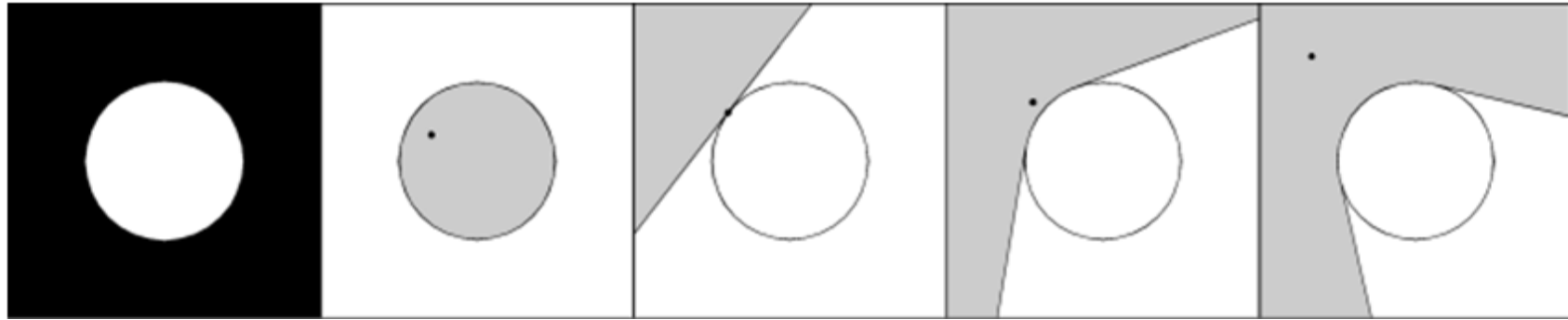
Denoising Approaches

Spatially adaptive methods, The basic principle:

- there are no simple models able to describe the whole image y , thus perform the regression $\hat{y}(x) = E[z | x]$
- Adopt a simple model in small image regions. For instance
$$\forall x \in X, \quad \exists \tilde{U}_x \text{ s. t. } y|_{\tilde{U}_x} \text{ is a polynomial}$$
- 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.

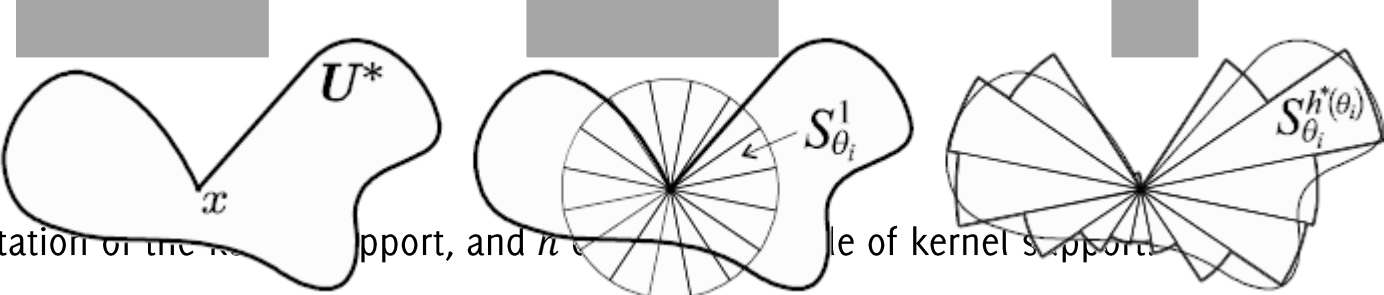
For practical reasons, the ideal neighborhood is assumed starshaped

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 $\{g_{\theta,h}\}_{\theta,h}$



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

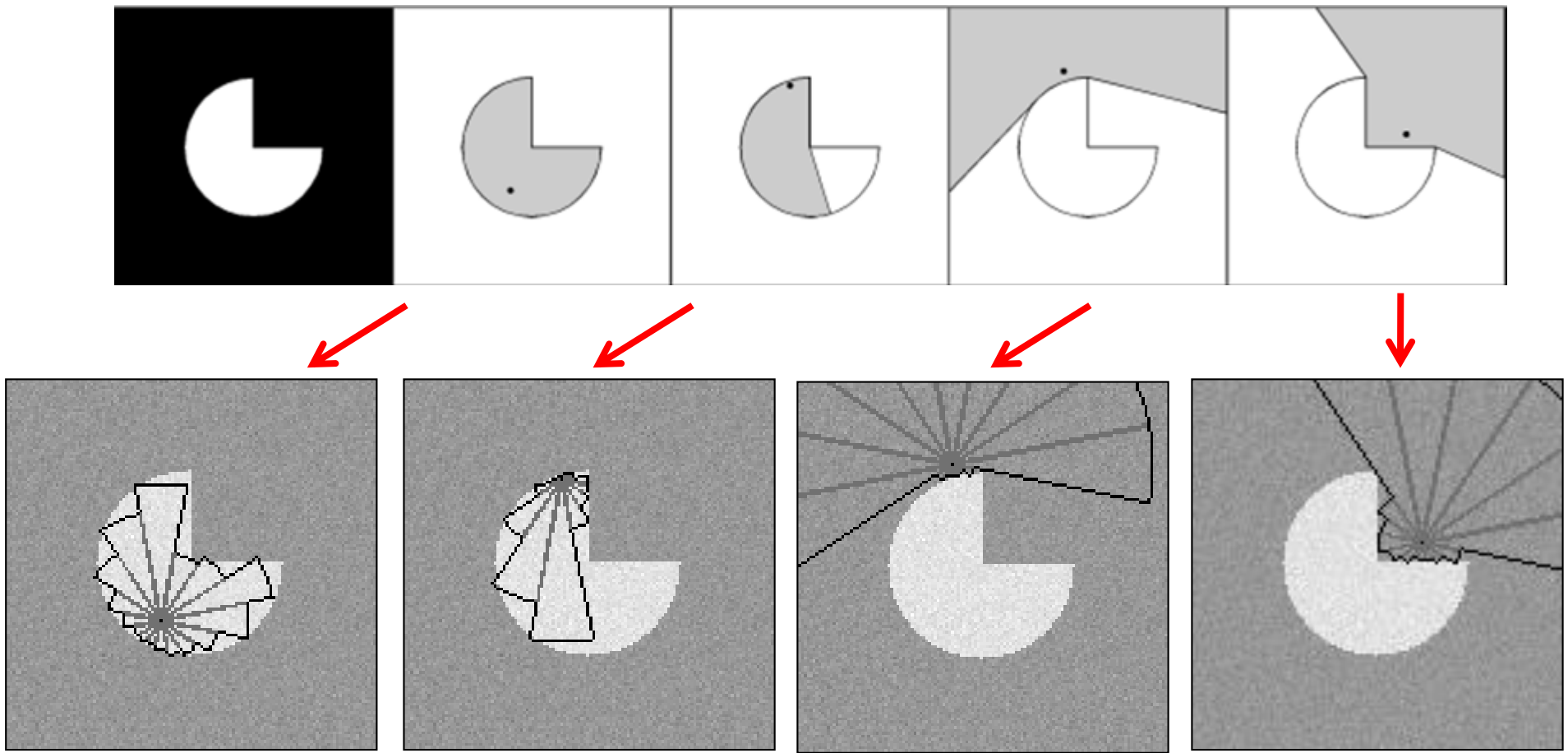
Ideal Neighborhood

Directional kernels

Discrete Adaptive Neighborhood

Ideal neighborhood – an illustrative example

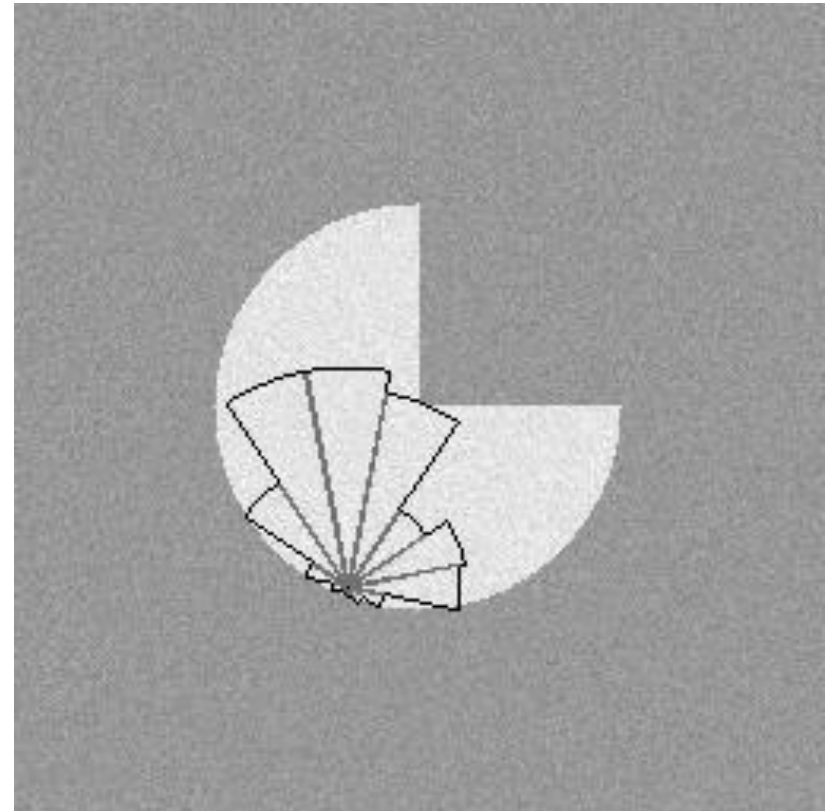
Ideal in the sense that the neighborhood defines the support of pointwise Least Square Estimator of the reference point.



Examples of Adaptively Selected Neighborhoods

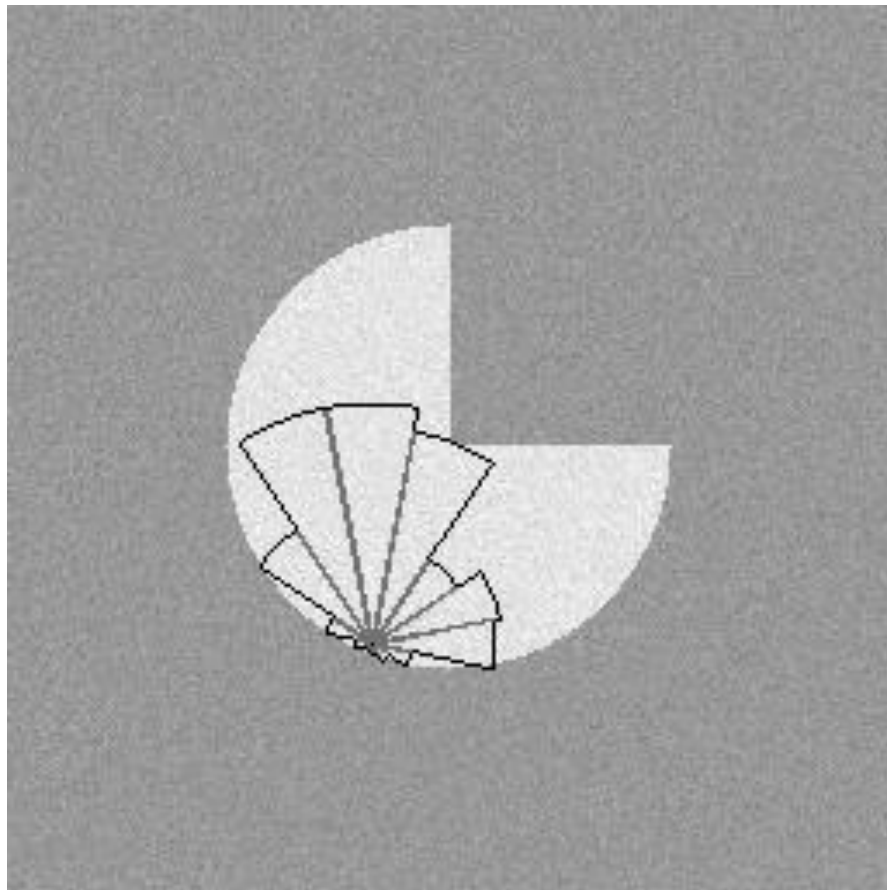
Define, $\forall x \in X$, the “ideal” neighborhood \tilde{U}_x

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



Examples of adaptively selected neighborhoods

Neighborhoods adaptively selected using the LPA-ICI rule



Example of Performance

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

