

Digital Images: Formation, Transformations and Filters

Giacomo Boracchi

Image Analysis and Computer Vision

Politecnico di Milano

November 15 2023

Book: GW, chapter 3

The Broad Landscape of Computer Vision and Pattern Recognition

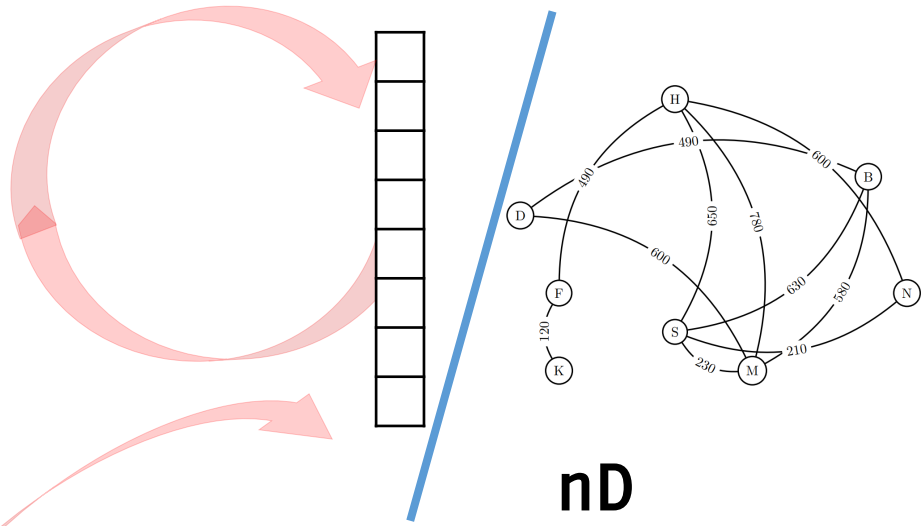
Machine Learning
Pattern Analysis

Image Analysis

Image
Processing



2D



nD

So far

Computer Graphics



Computer Vision



3D

Shape Analysis

Geometry
Processing

from M. Bronstein

Machine Learning
Pattern Analysis

Image Analysis

Image
Processing



2D

Computer Graphics

Computer Vision

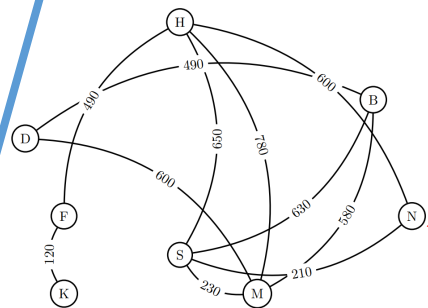


3D

Geomery
Processing

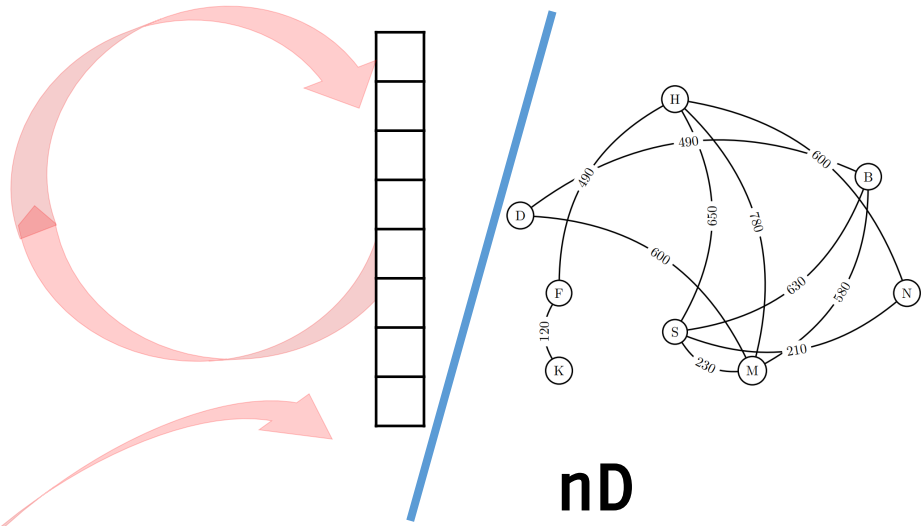
Shape Analysis

nD



Machine Learning
Pattern Analysis

Image Analysis



nD

For the next few lectures

Shape Analysis

Computer Graphics

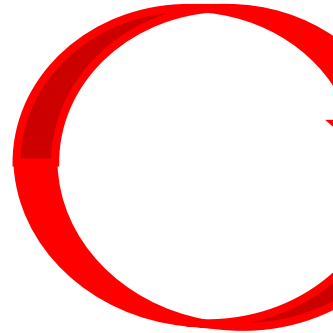


Image
Processing



2D



3D

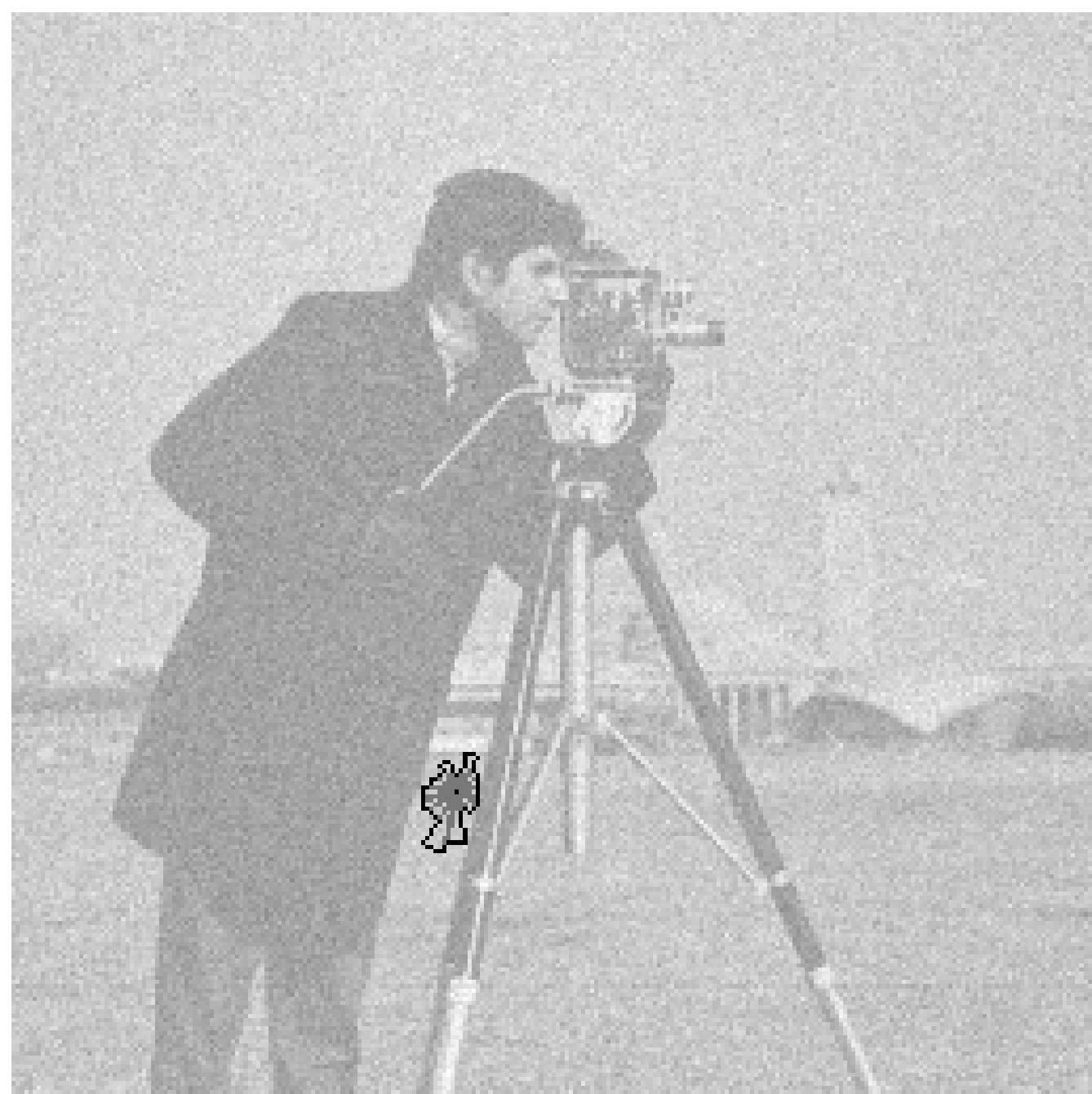
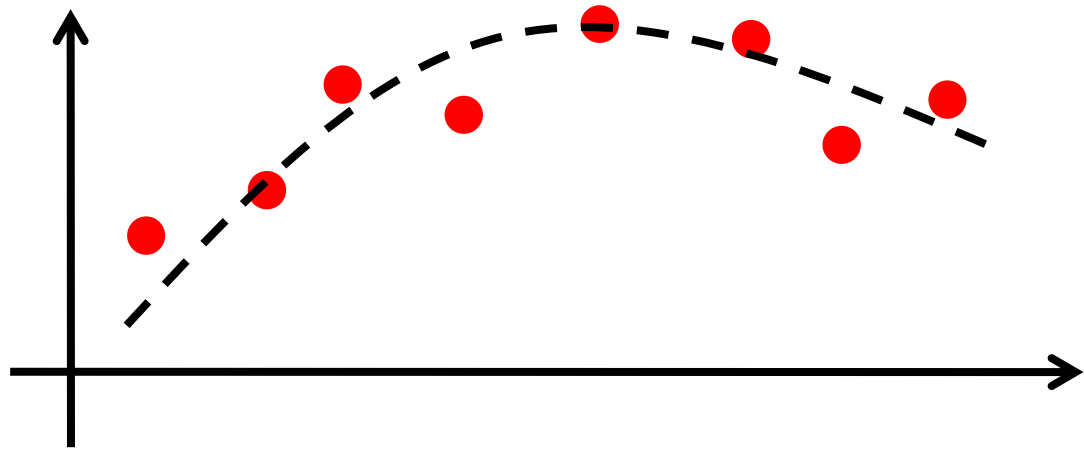
Geomery
Processing



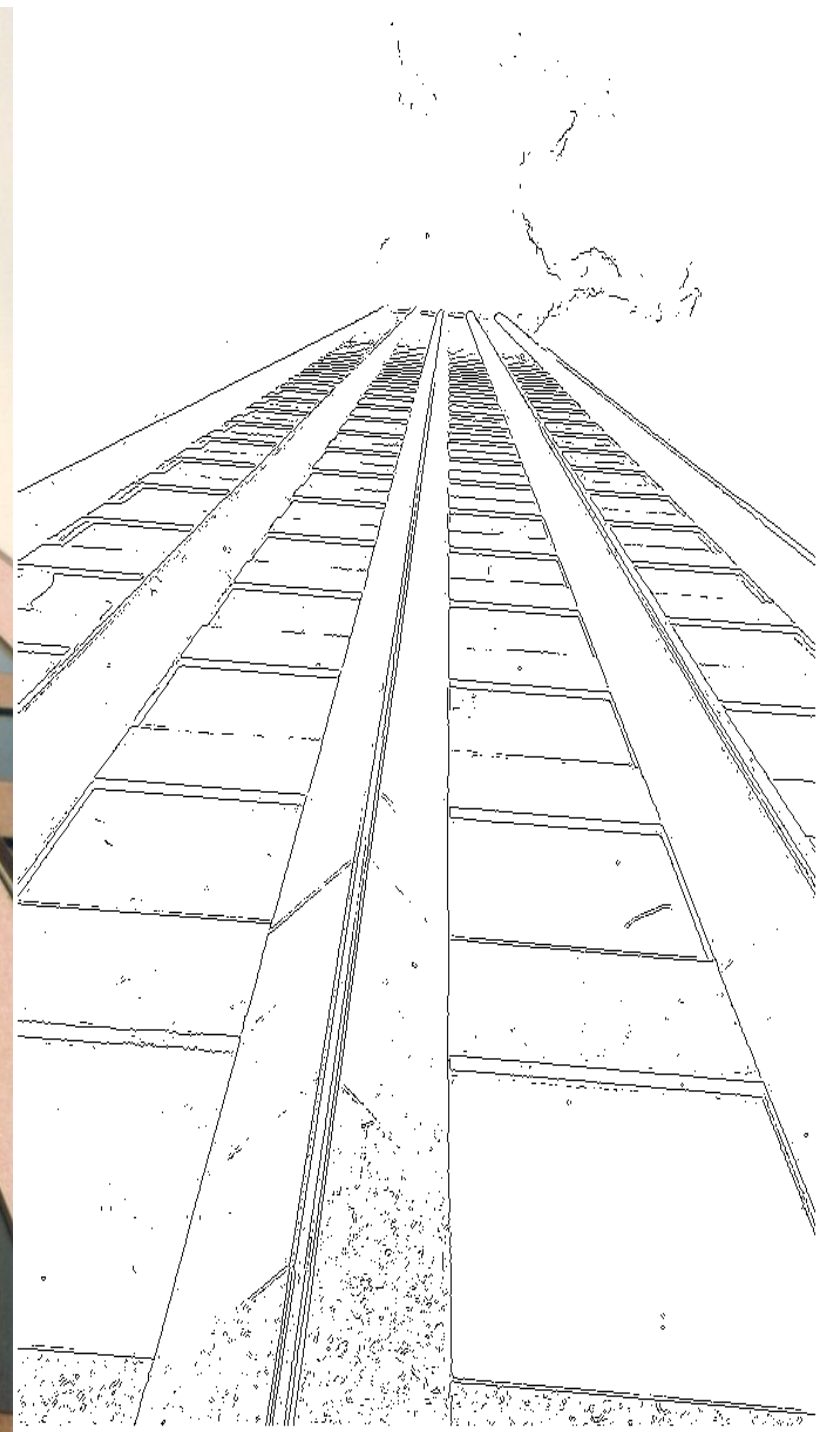
Computer Vision

In particular, we will see:

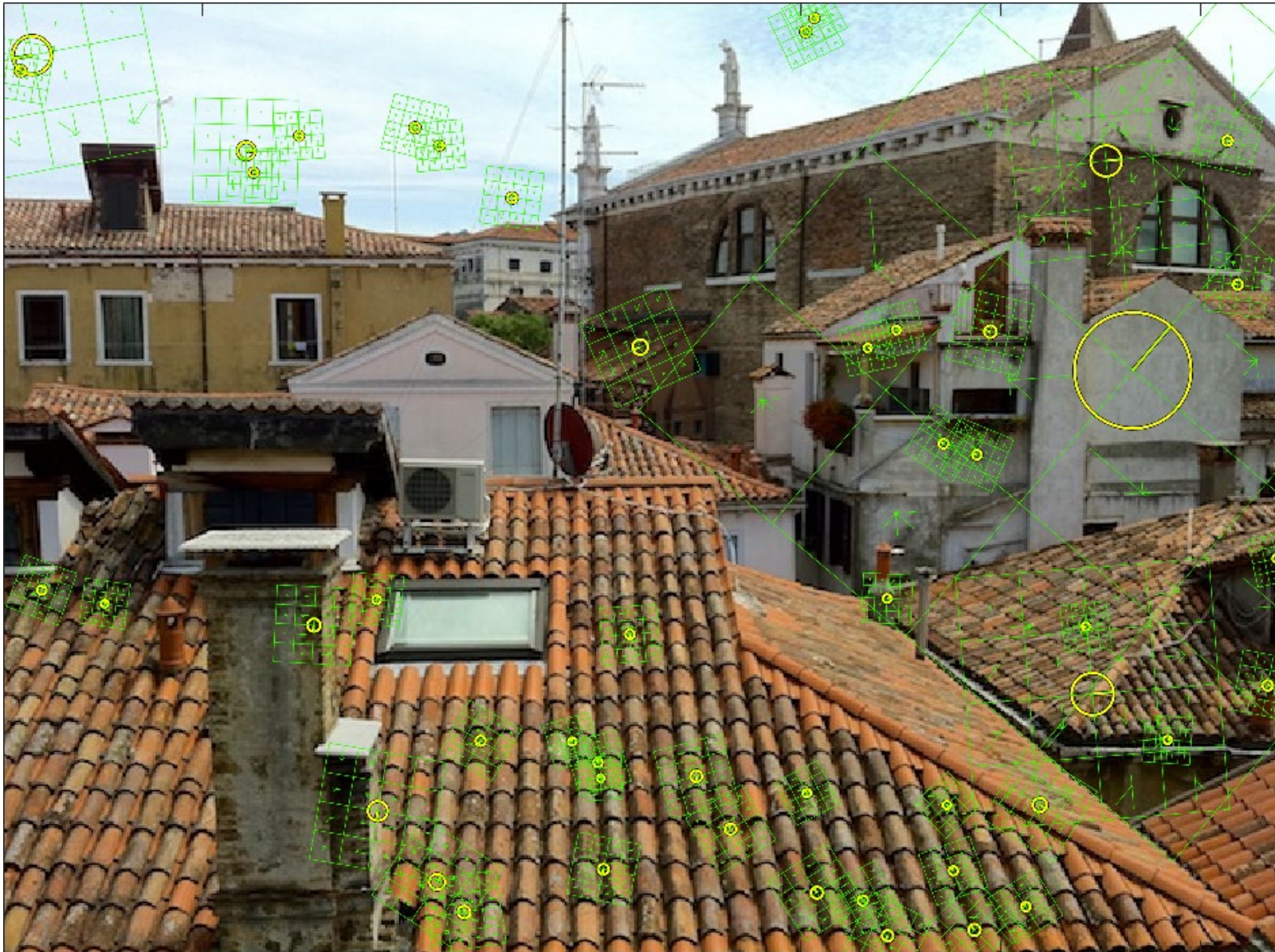
Filtering / denoising



Edge detection



Salient Point / Feature Extraction



Restoration & Inverse Problems



Restoration & Inverse Problems



Photometric Image Formation

Colour Filter Array

Colour Filter Arrays

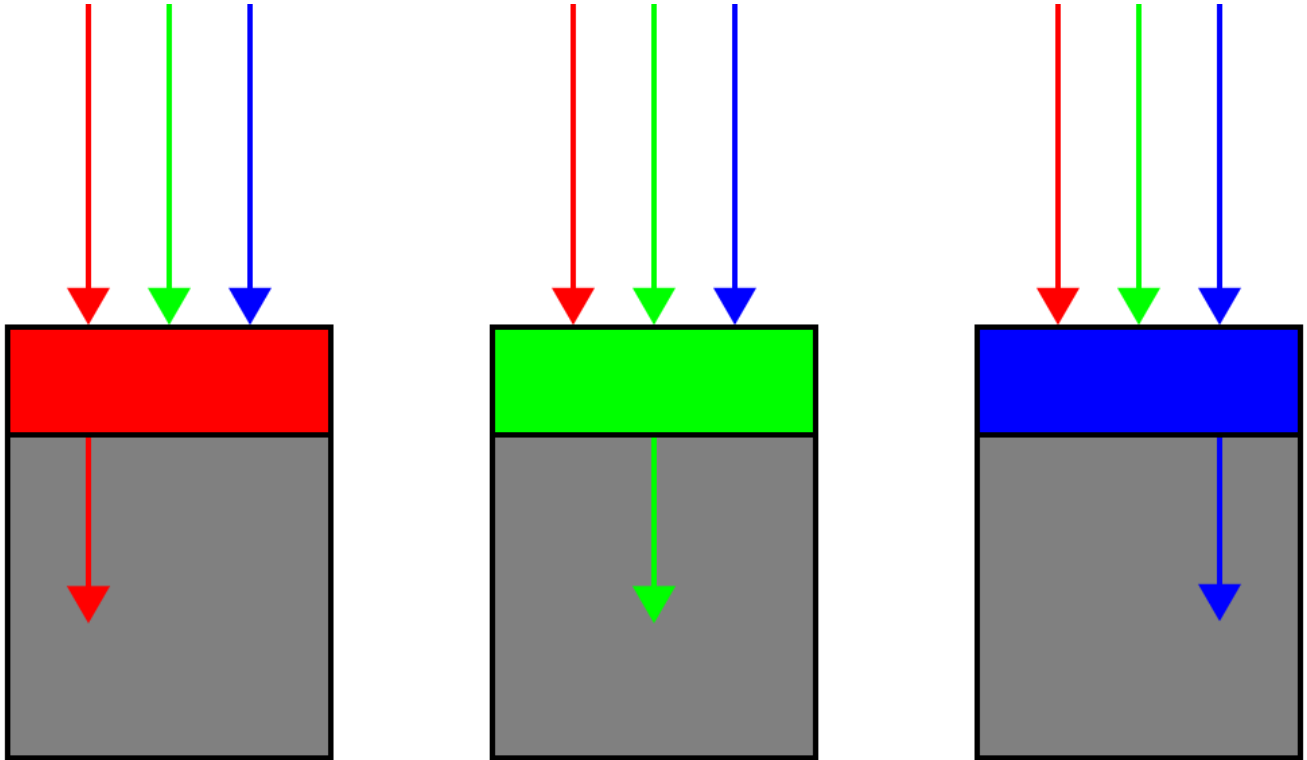
Typical **photosensors** detect light intensity with little or **no wavelength specificity**, and therefore cannot separate colour information.

Colour Filters Array (CFA) are used to filter the light by wavelength range.

Separate filtered intensities include information about the colour of light.

For example, the Bayer filter gives information about the intensity of light in red, green, and blue (RGB) wavelength regions

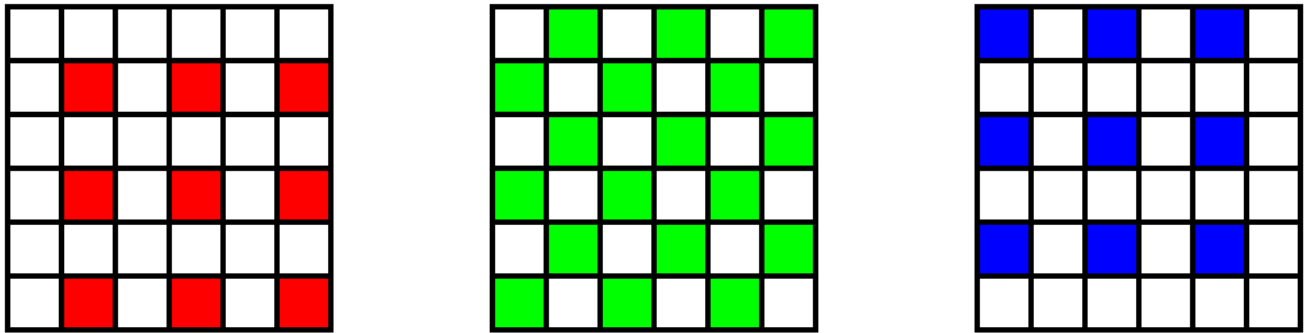
Colour Filter Arrays



Incoming light

Filter layer

Sensor array



Resulting pattern

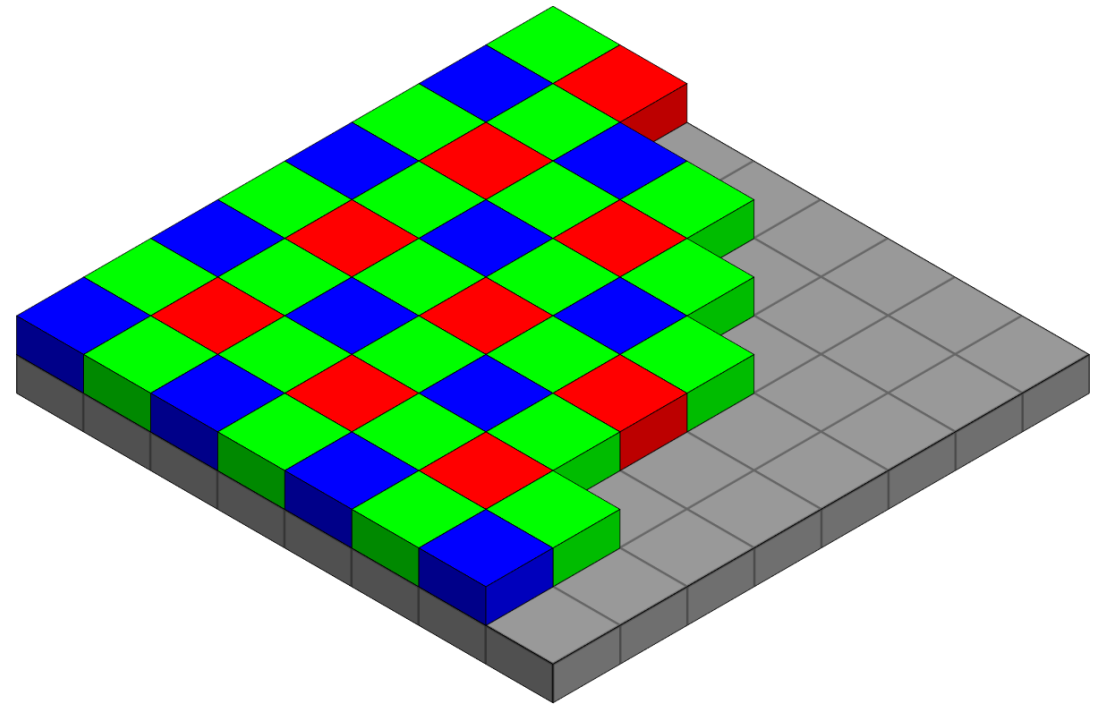
By en>User:Cburnett - Own workThis W3C-unspecified vector image was created with Inkscape., CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=1496872>

Bayer Pattern

For example, the Bayer filter (RGGB) gives information about the intensity of light in red, green, and blue wavelength regions.

- Green colour is sampled twice

There are many different patterns, including RYYB which gives a better response in low-light conditions



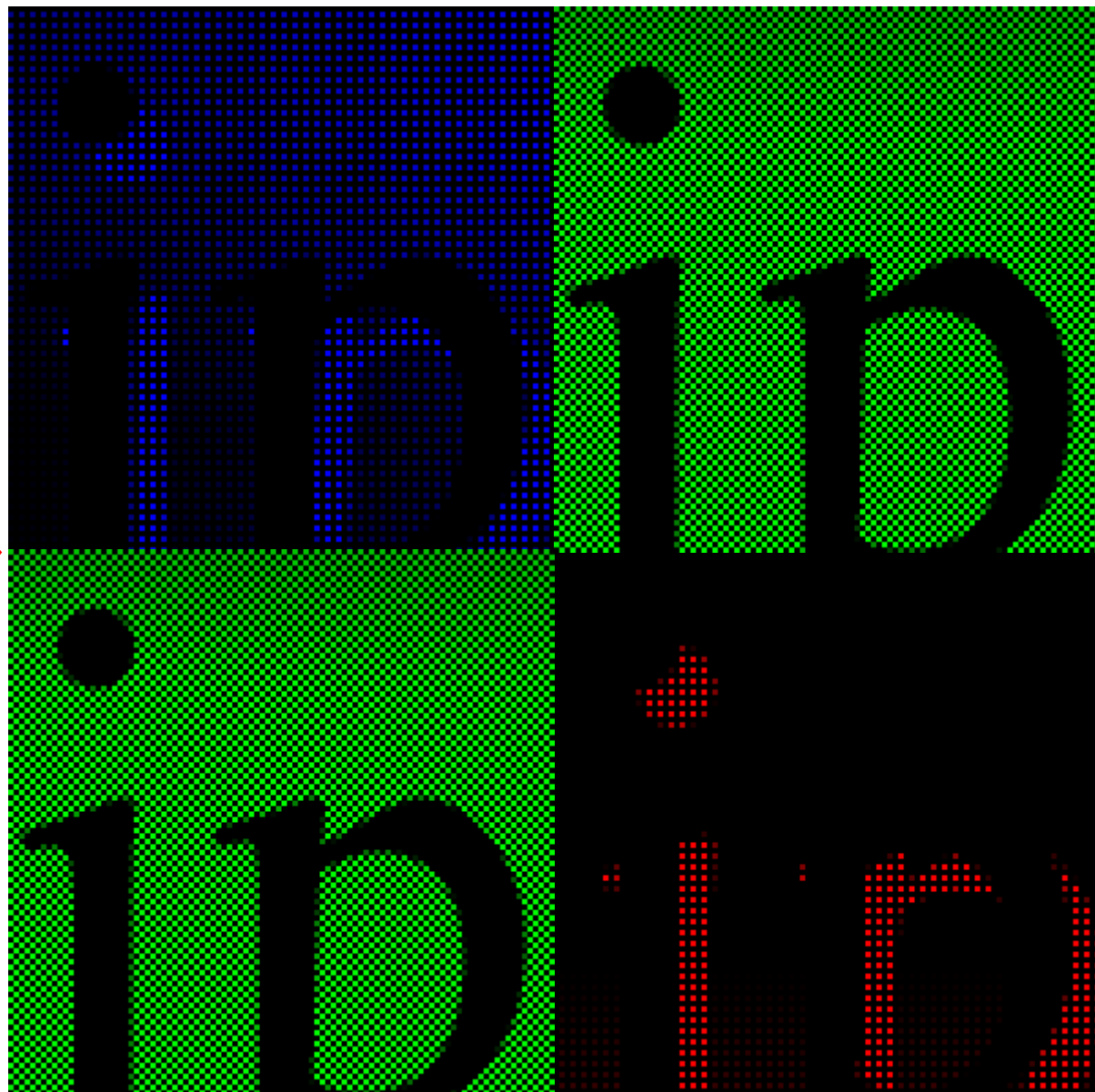
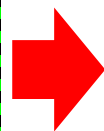
By Cburnett - Own work, CC BY-SA 3.0,
<https://commons.wikimedia.org/w/index.php?curid=1496858>

The raw output of digital camera

Every pixel of the array is only sensitive to a single colour.



The raw output of digital camera

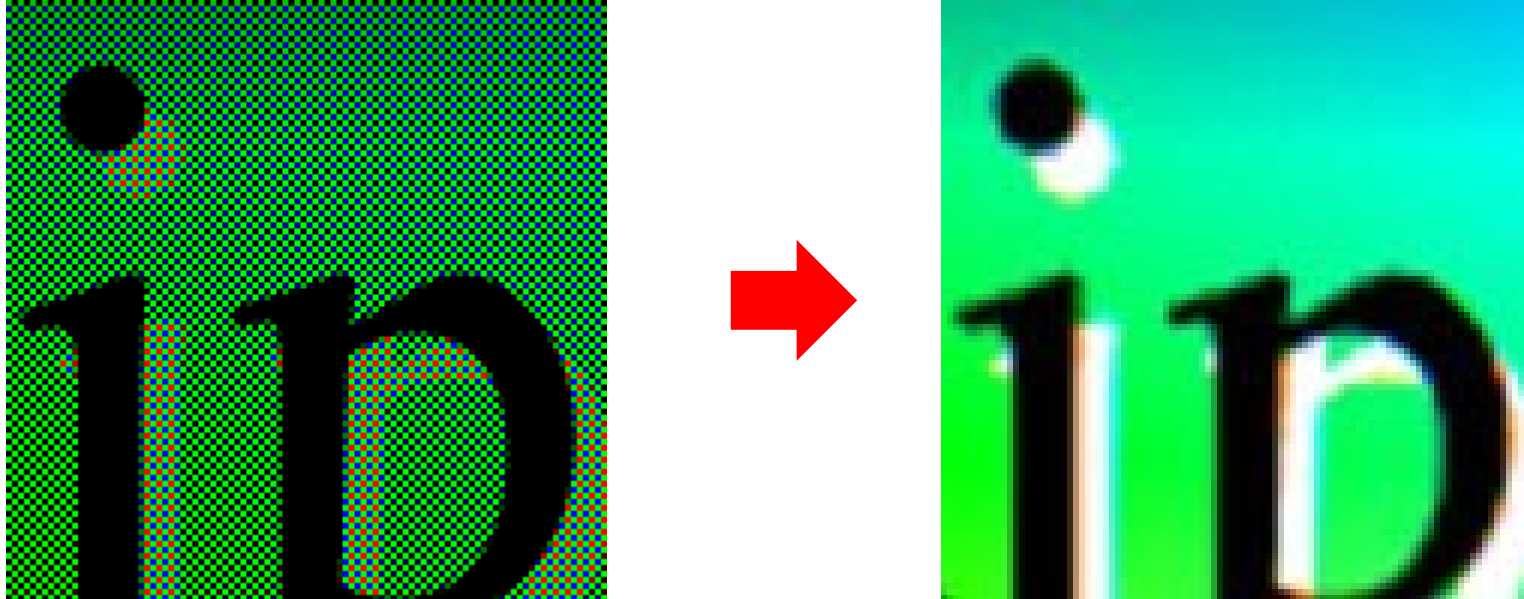


Demosaicing

Demosaicing, a.k.a. CFA interpolation or Colour Reconstruction

Algorithm to **reconstruct a full colour image** (3 colours per pixel) from the **incomplete colour output** from an image sensor (CFA).

This is a **multivariate regression problem**



Demosaicing

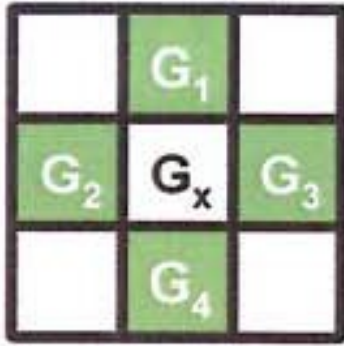
Issues:

- In **Bayer pattern** each pixel is sensitive to a single colour, while in the image each pixel portrays a mixture of 3 primary colours

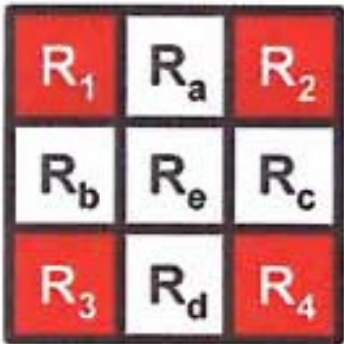
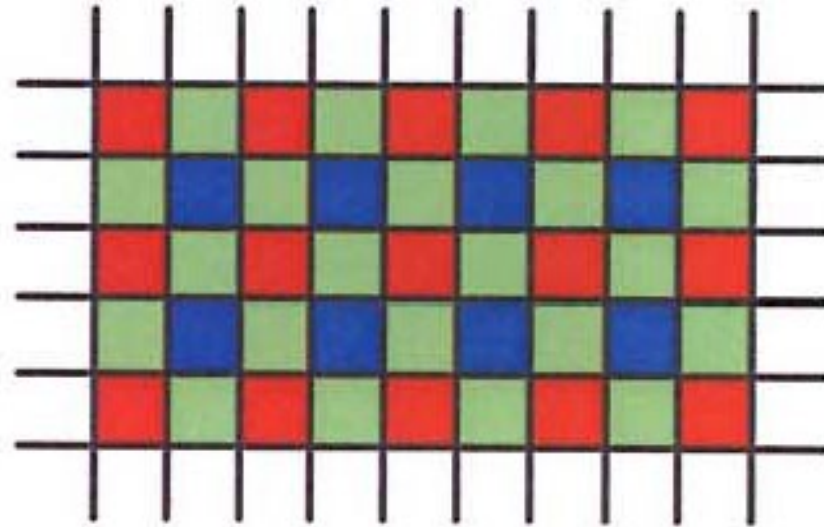
Desiderata:

- Avoid colour artefacts
- Maximum preservation of the image resolution
- Low complexity or efficient in-camera hardware implementation
- Amenability to analysis for accurate noise reduction

Example of Demosaicing by bilinear interpolation



$$G_x = (G_1 + G_2 + G_3 + G_4) / 4$$



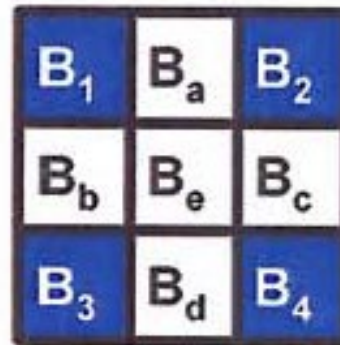
$$R_a = (R_1 + R_2) / 2$$

$$R_b = (R_1 + R_3) / 2$$

$$R_c = (R_2 + R_4) / 2$$

$$R_d = (R_3 + R_4) / 2$$

$$R_e = (R_1 + R_2 + R_3 + R_4) / 4$$



$$B_a = (B_1 + B_2) / 2$$

$$B_b = (B_1 + B_3) / 2$$

$$B_c = (B_2 + B_4) / 2$$

$$B_d = (B_3 + B_4) / 2$$

$$B_e = (B_1 + B_2 + B_3 + B_4) / 4$$

More sophisticated channel-wise interpolation include bicubic/spline interpolation, Lanczos resampling

Demosaicing

Color-independent algorithms typically present artifacts in regions containing edges and textures



Zipper effects are unnatural changes of intensities over a number of neighboring pixels, manifesting as an “on-off” pattern in regions around edges



False colors are spurious colors which are not present in the original image scene [...] They appear as sudden hue changes due to inconsistency among the three color planes and usually around fine image details and edges

False Colors



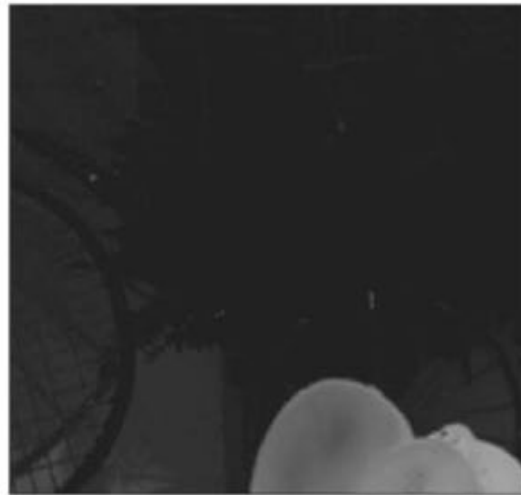
(a)



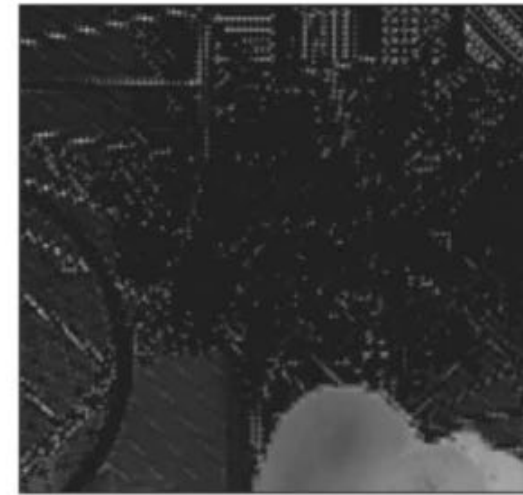
(b)



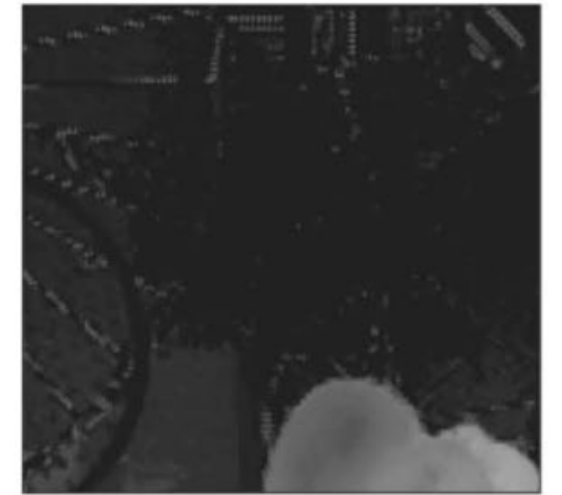
(c)



(d)



(e)



(f)

Fig. 10 Original image region (from image 27 in Fig. 15) and its demosaicked results obtained by (b) bilinear interpolation and (c) Freeman's method. The corresponding color difference planes (green minus red) are shown in (d), (e), and (f), respectively.

Demosaicing

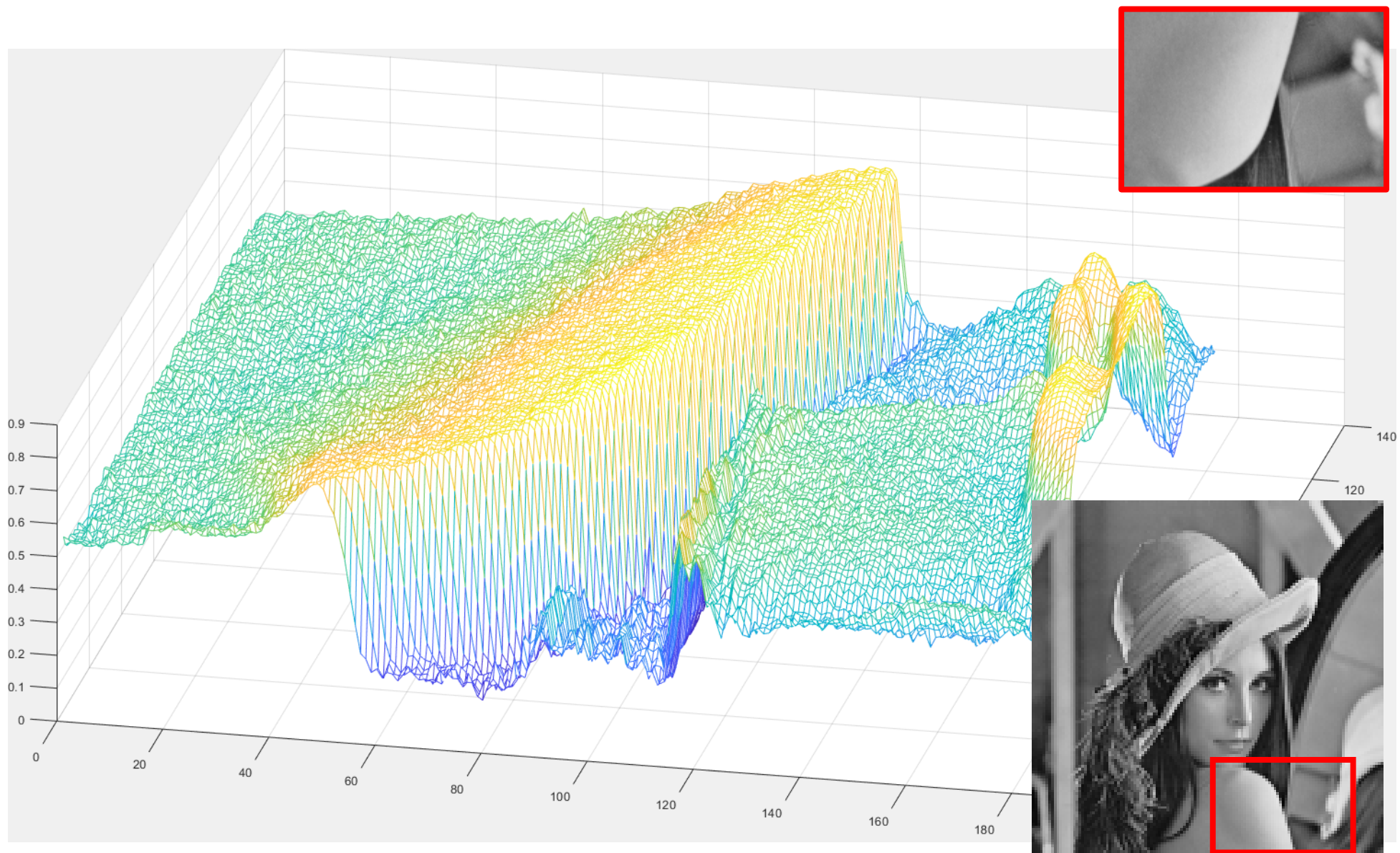
Examples of priors to be exploited to improve demosaicing quality

- Channel-wise similarity / consistency (colour differences, colour ratio)
- Spatial correlation, the structure of images
- Spectral correlation

Post-processing can be employed to suppress typical demosaicing artifacts

When we work channel-wise...

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



Spatial-Domain Methods in Image Processing

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 is a function, for instance
 - $T: \mathbb{R}^3 \rightarrow \mathbb{R}$ (e.g. colour to grayscale conversion)
 - $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ (e.g. changing the colour encoding)
 - $T: \mathbb{R} \rightarrow \mathbb{R}$ (many channel-wise intensity transformation)

T operates independently on each single pixel.

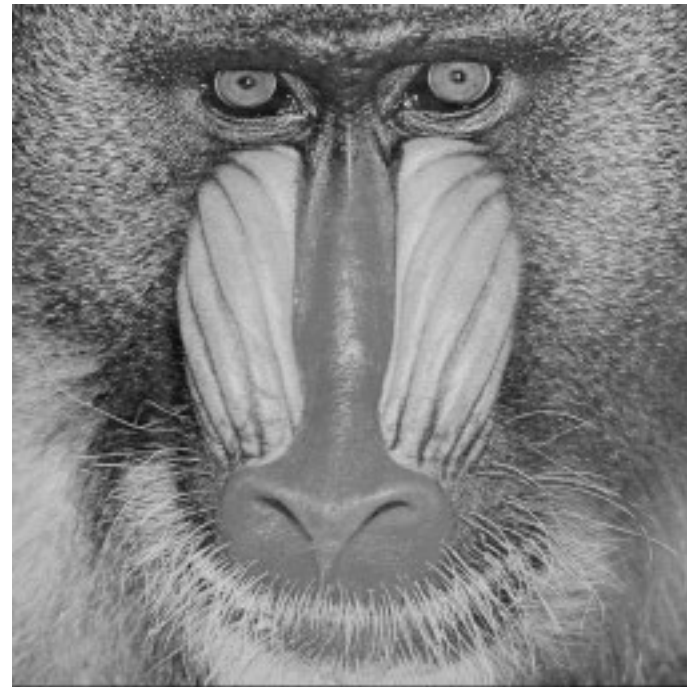
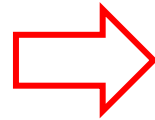
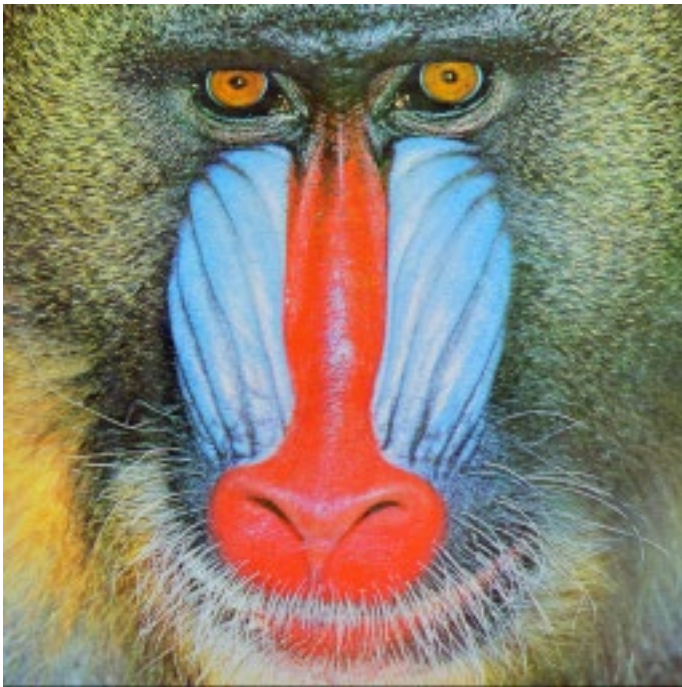
RGB \rightarrow Grayscale Conversion

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)

RGB \rightarrow YCbCr

There are many variants

$$\begin{bmatrix} Y' \\ P_B \\ P_R \end{bmatrix} = \begin{bmatrix} K_R & K_G & K_B \\ -\frac{1}{2} \cdot \frac{K_R}{1-K_B} & -\frac{1}{2} \cdot \frac{K_G}{1-K_B} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \cdot \frac{K_G}{1-K_R} & -\frac{1}{2} \cdot \frac{K_B}{1-K_R} \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

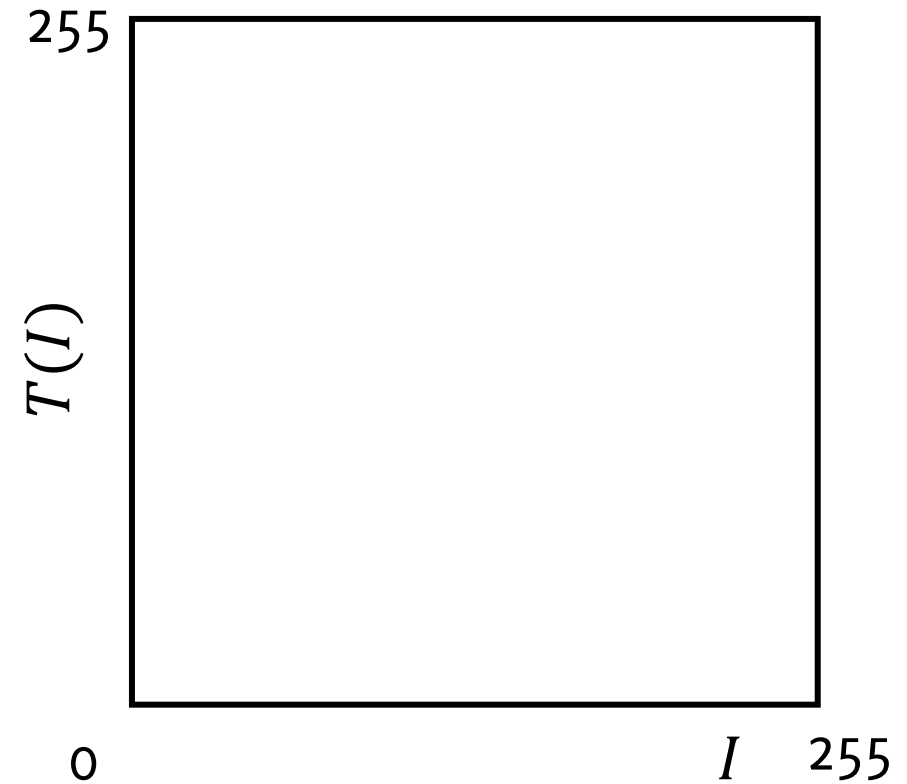
Where ' denotes the intensities are in the $[0,1]$ range

Negative Transformation

Simple transformation that maps black to white and white to black, and all the intensity levels in between as:

$$I(r, c) \rightarrow 255 - I(r, c)$$

This is a linear transformation of intensities

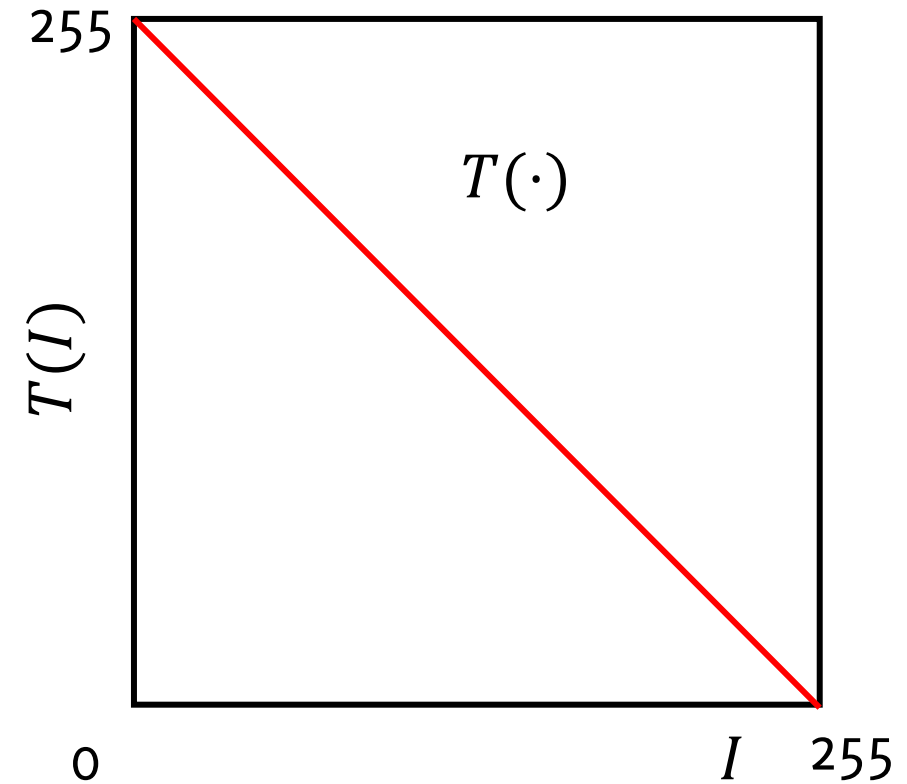


Negative Transformation

Simple transformation that maps black to white and white to black, and all the intensity levels in between as:

$$I(r, c) \rightarrow 255 - I(r, c)$$

This is a linear transformation of intensities



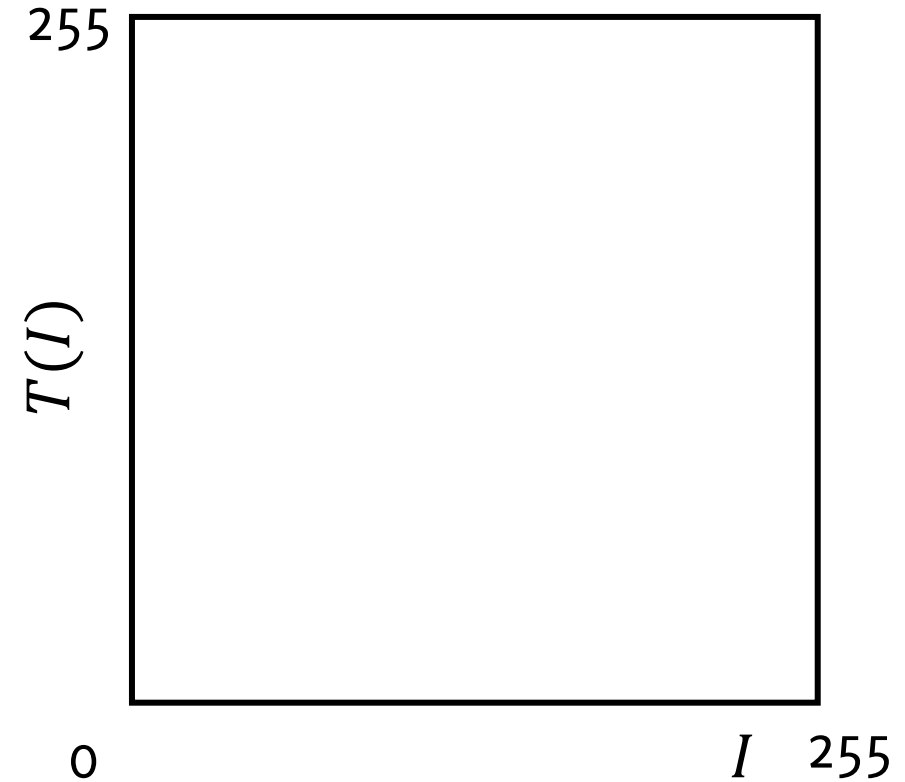
Intensity Rescaling

In some cases images are conveniently mapped in the $[0,255]$ range, covering such that

- $\min(T(I)) = 0$
- $\max(T(I)) = 255$

$$I(r, c) \rightarrow 255 * \frac{I(r, c) - \min(I)}{\max(I) - \min(I)}$$

This is a linear transformation of intensities



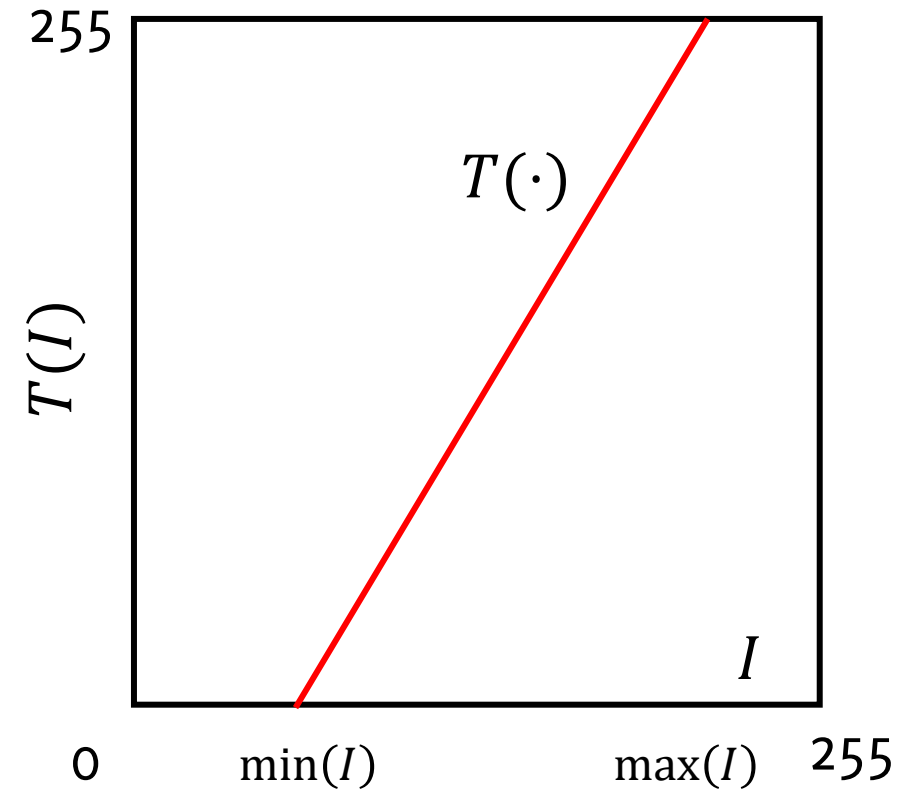
Intensity Rescaling

In some cases images are conveniently mapped in the $[0,255]$ range, covering such that

- $\min(T(I)) = 0$
- $\max(T(I)) = 255$

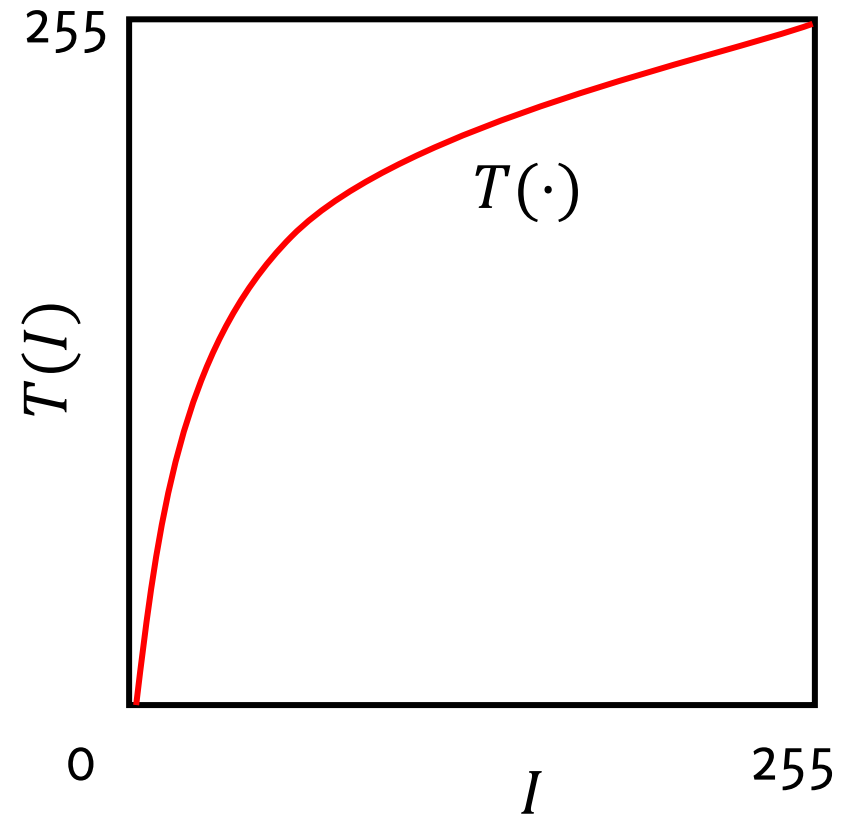
$$I(r, c) \rightarrow 255 * \frac{I(r, c) - \min(I)}{\max(I) - \min(I)}$$

This is a linear transformation of intensities



Gray-level mapping

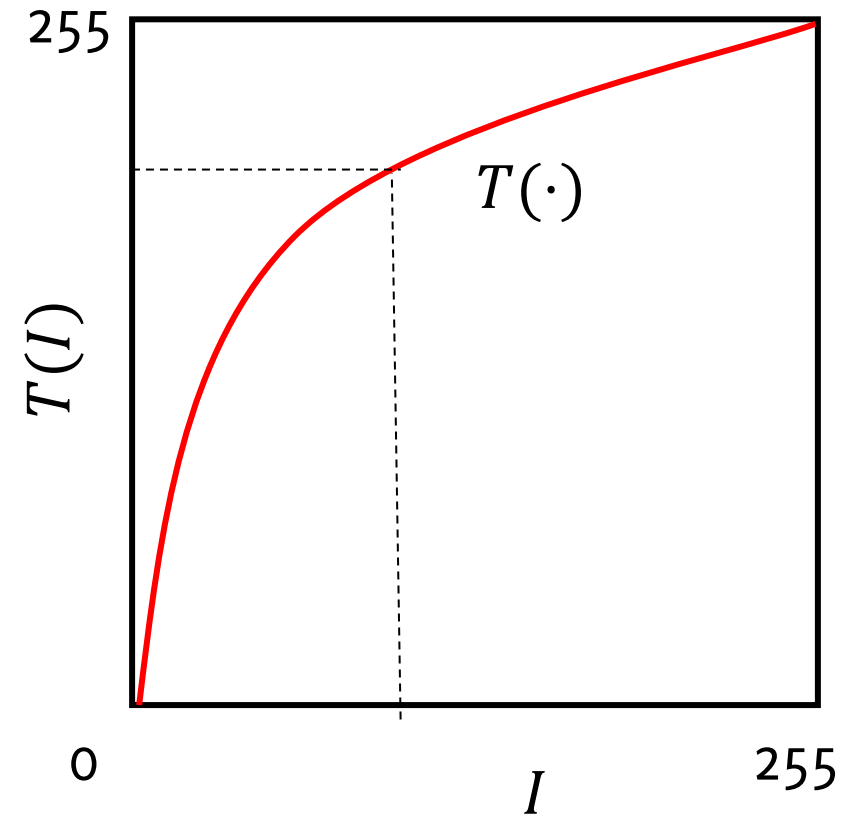
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operates on gray-scale images or on each color-plane separately



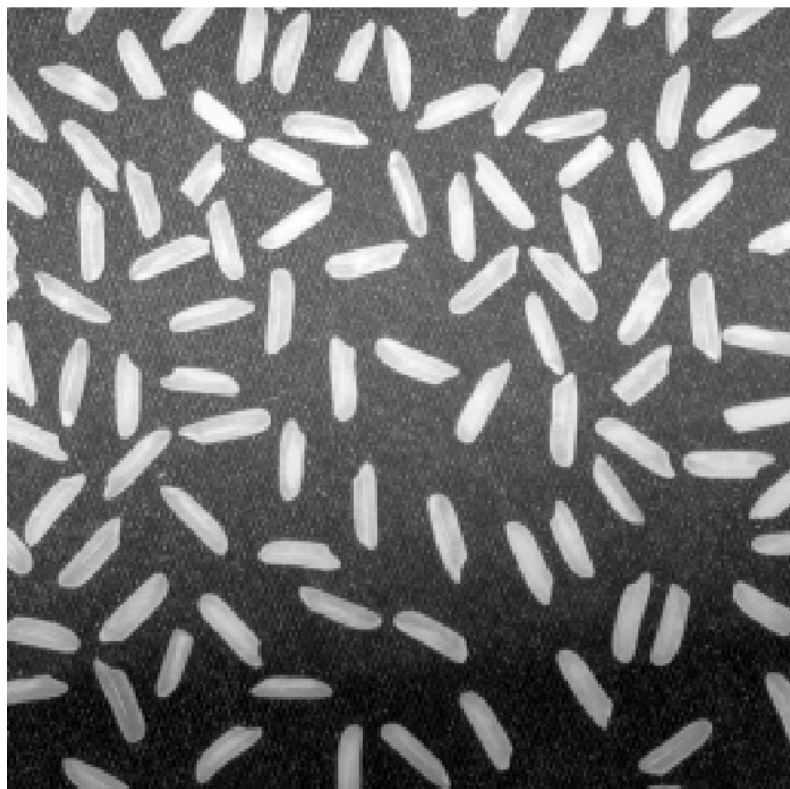
Gray-level mapping

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

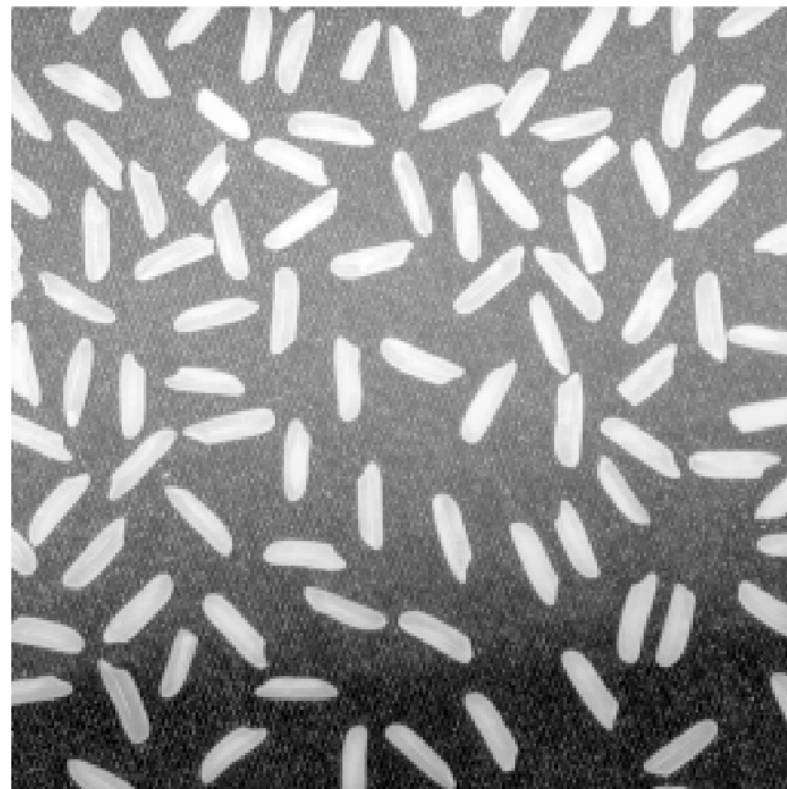
What does this T do?



Contrast increases in dark, decreases in bright



Input I

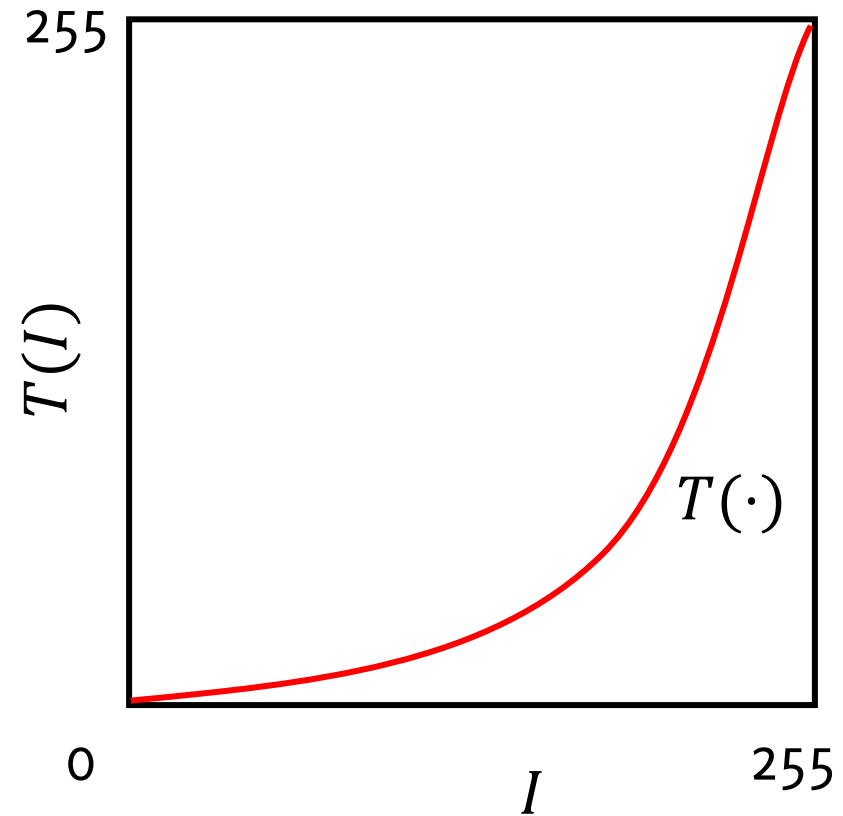


Output $G = T(I)$

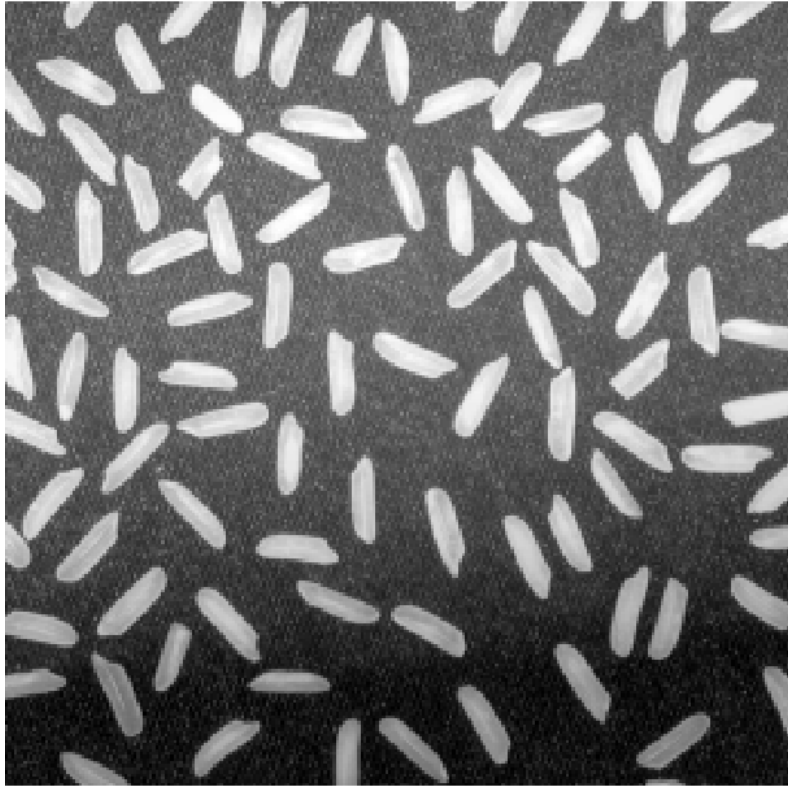
Gray-level mapping

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

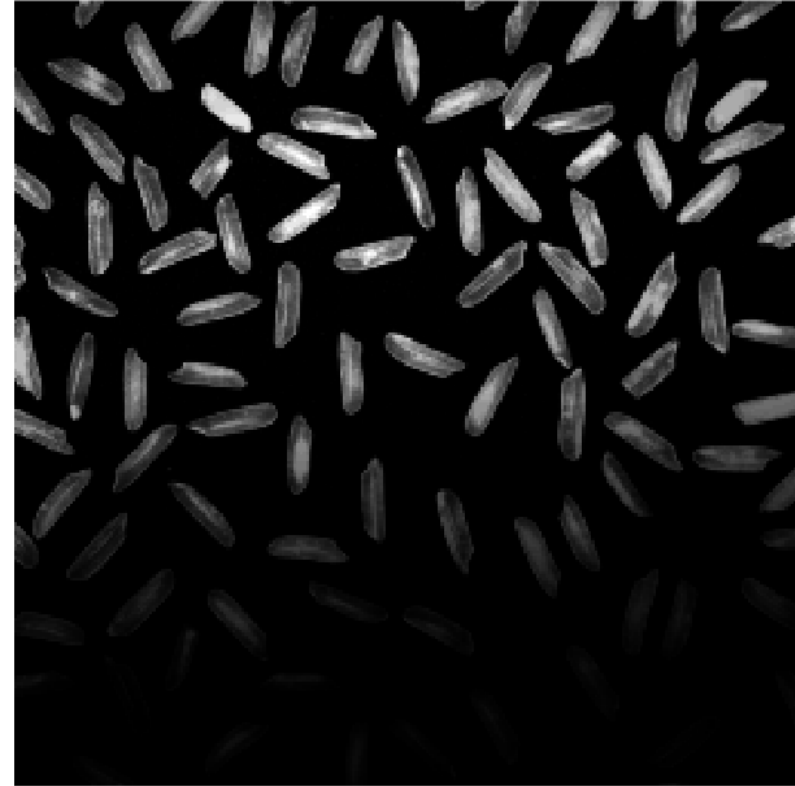
What does this T do?



Contrast increases in bright, decreases in dark



Input I

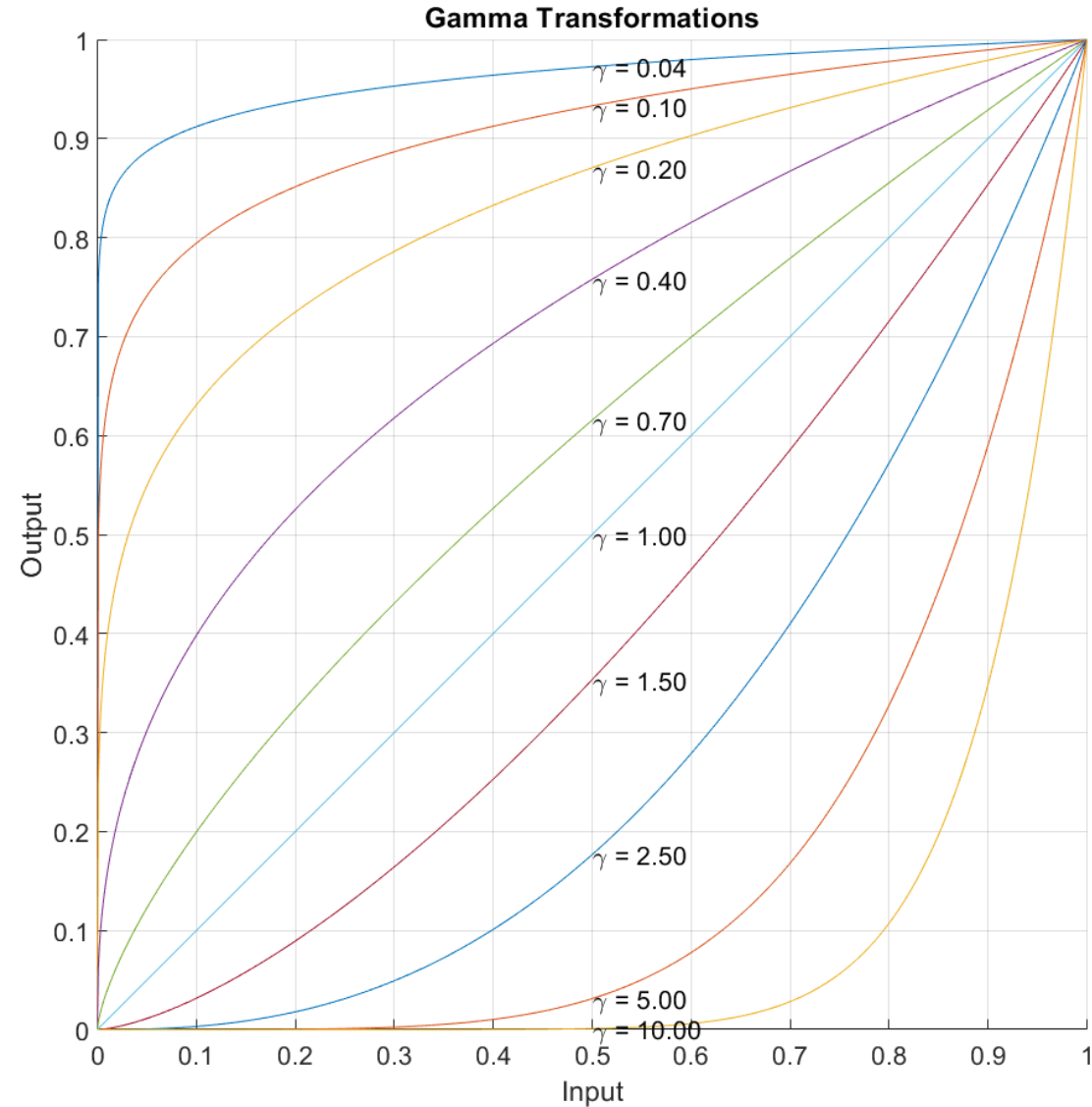


Output $G = T(I)$

Gamma Correction

Power-law transformation that can be written as

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



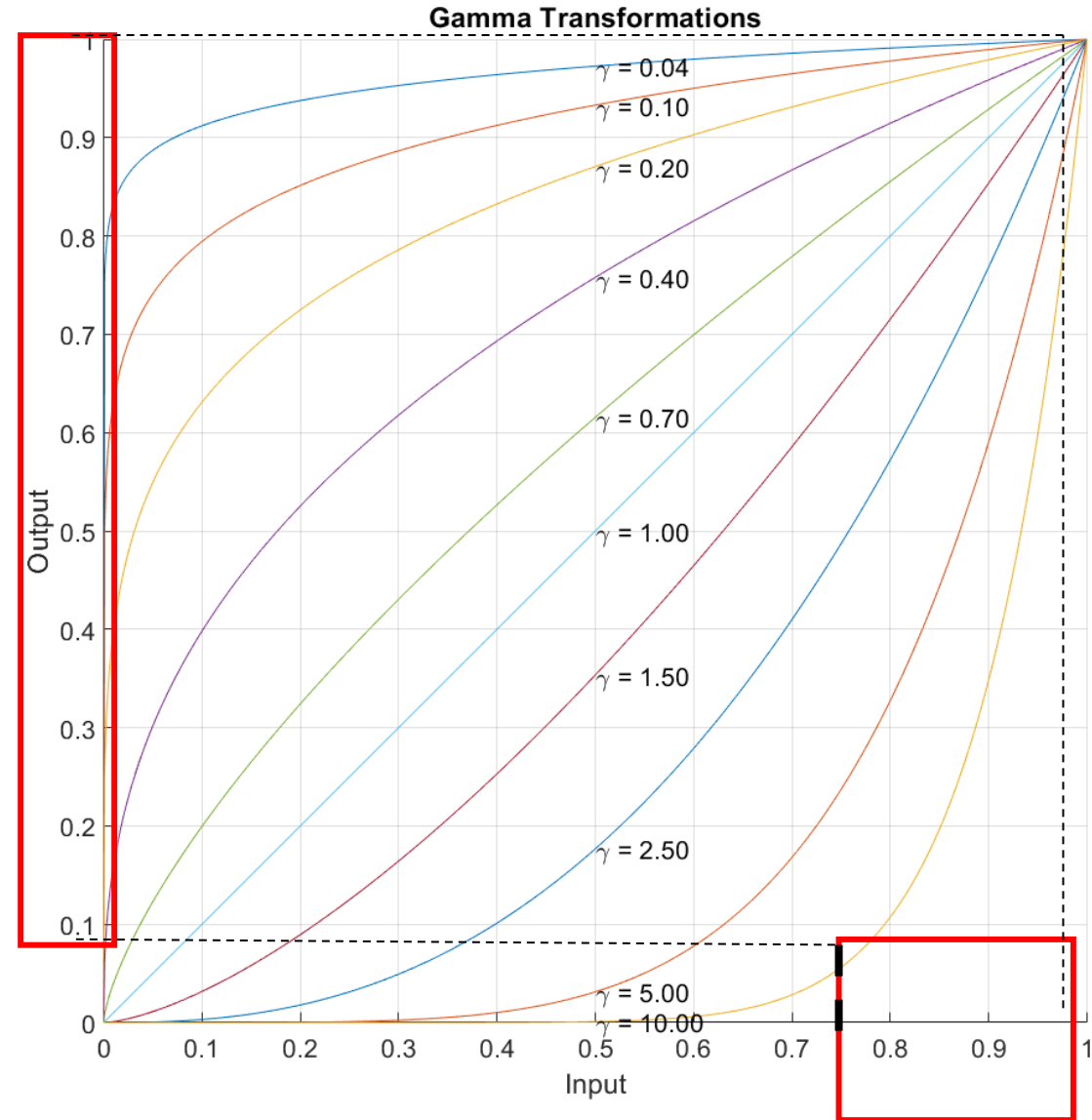
Gamma Correction

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



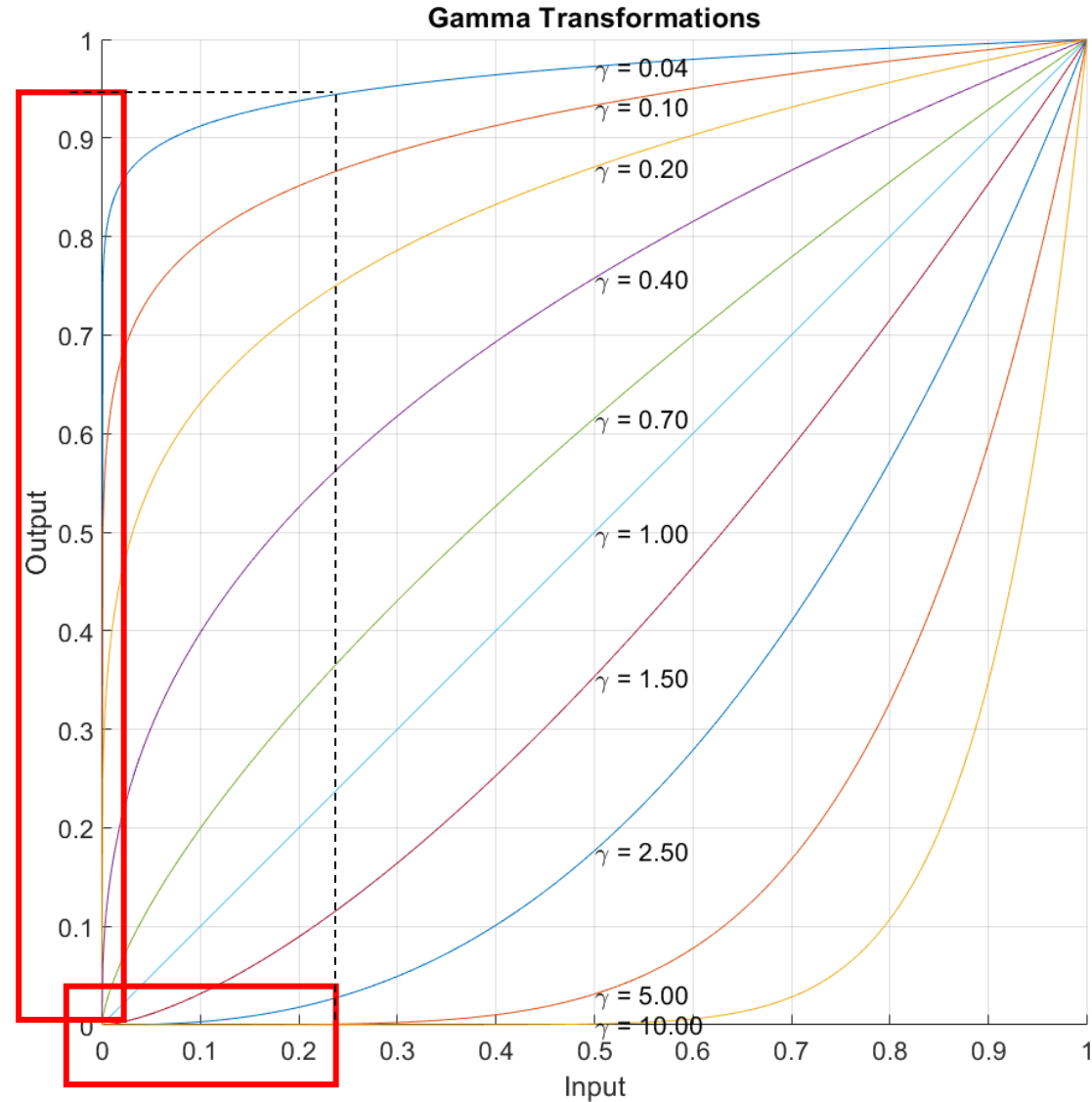
Gamma Correction

Power-law transformation that can be written as

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

Contrast Enhancement:

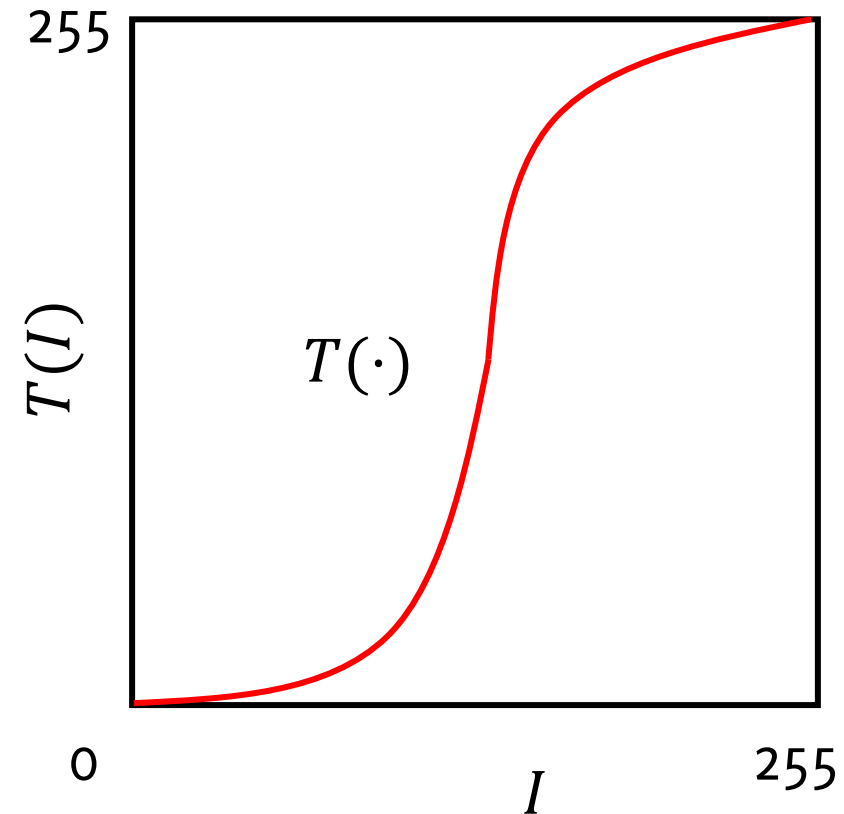
- Low values of γ stretch the intensity range at high-values
- High values of γ stretch the intensity range at low values



Gray Level Mapping

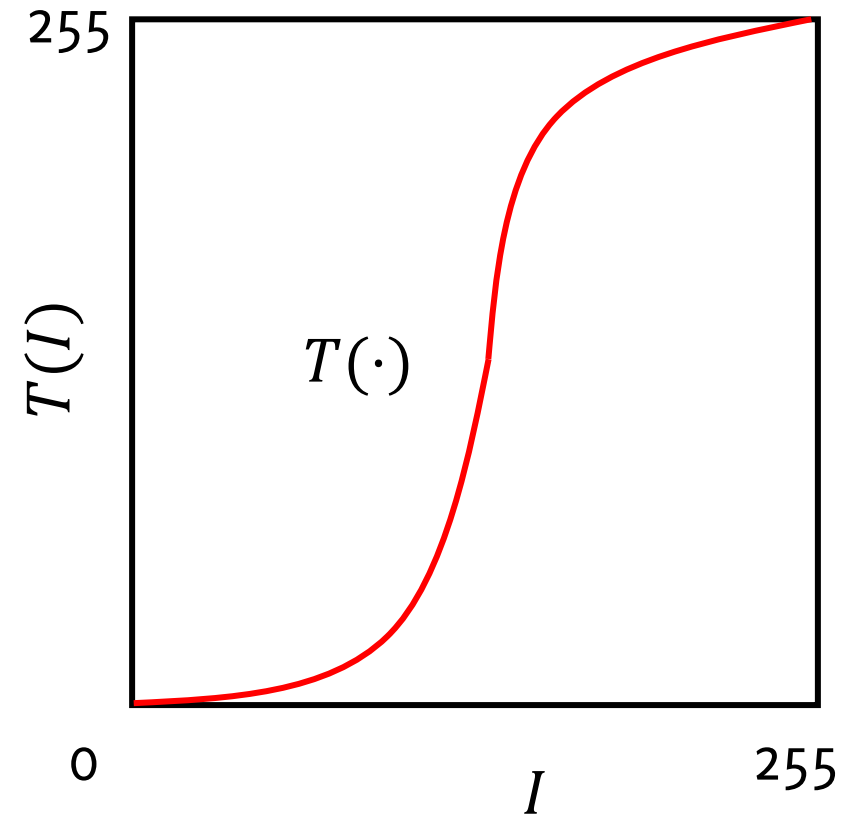
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operates on gray-scale images or on each color-plane separately

What does this T do?

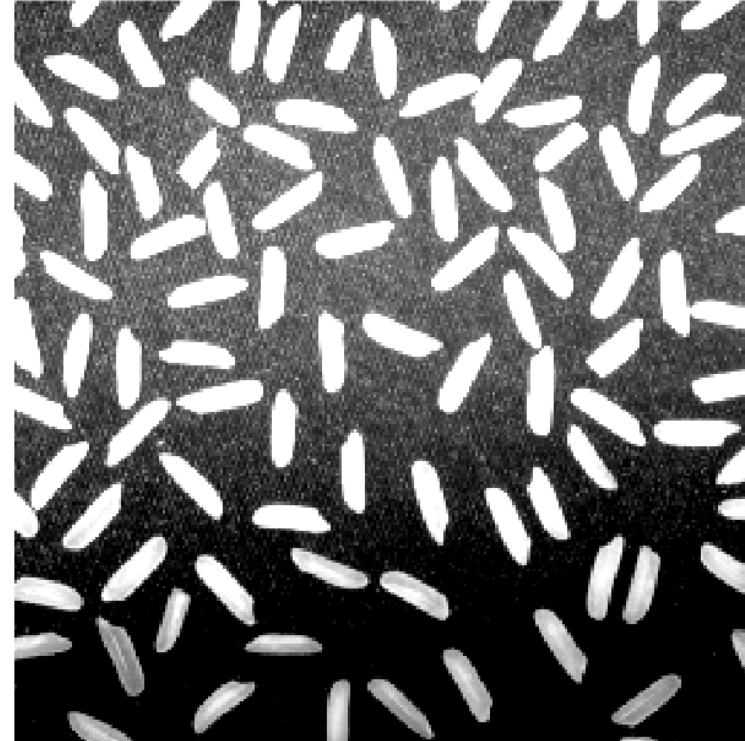
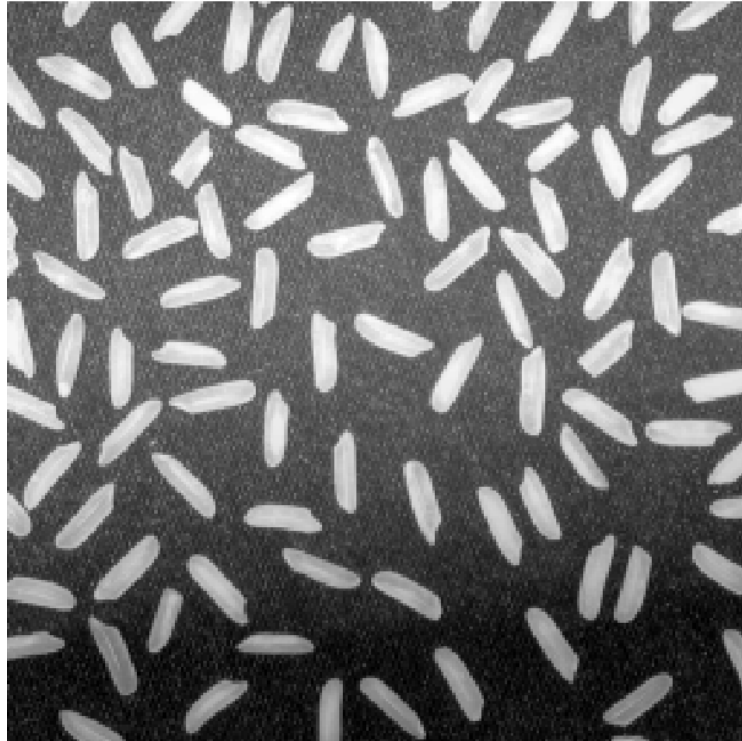


Contrast Stretching

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



Contrast Stretching

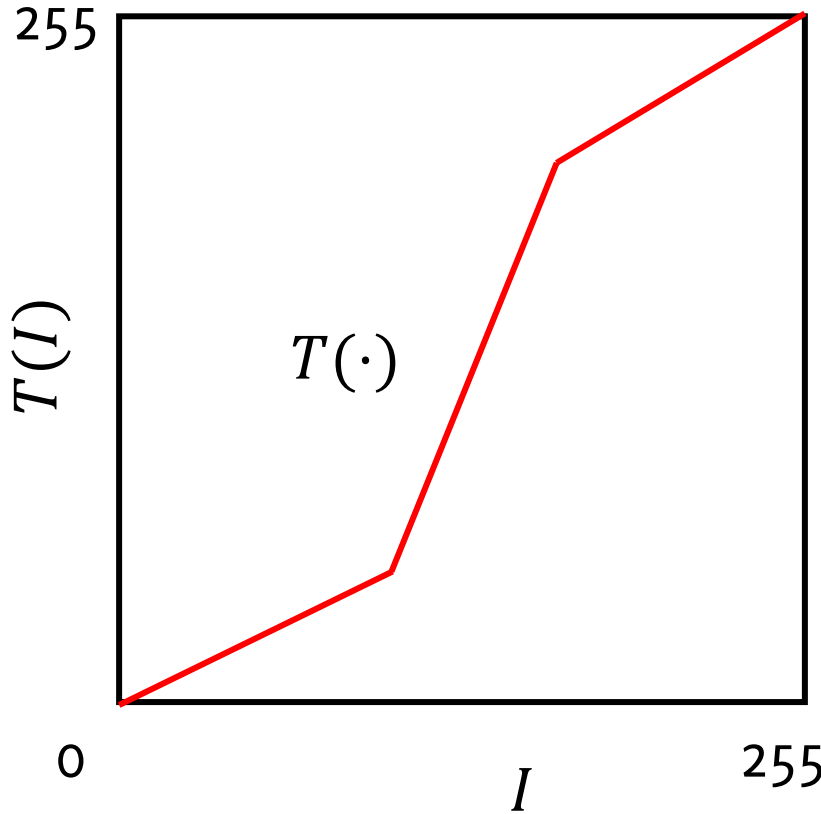


Contrast stretching: increases the contrast at values in the middle of intensity range, decreases contrast at bright and dark regions.

It is implemented by piecewise or parametric transformations

Contrast Stretching

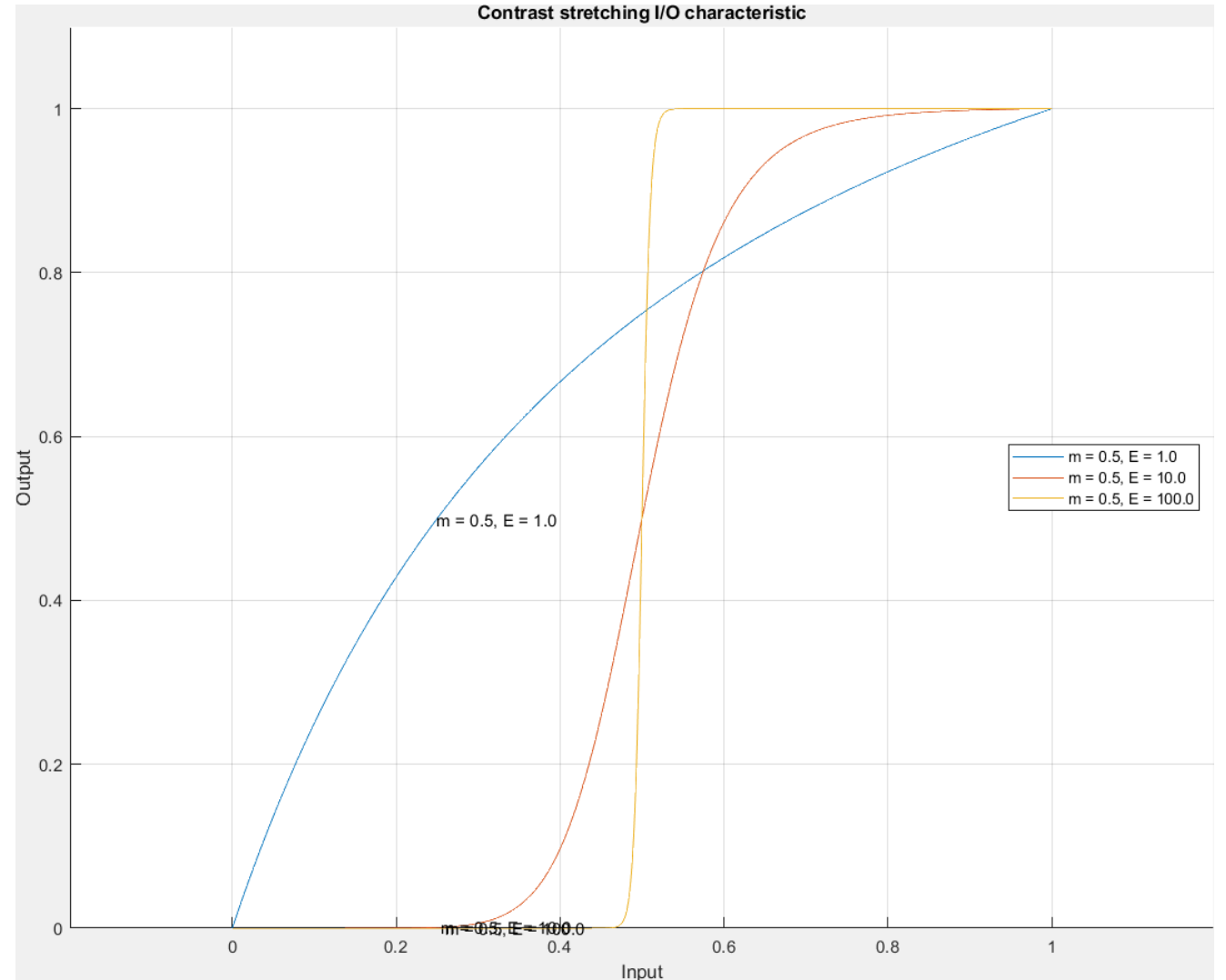
Can be defined by piecewise linear mapping...



Contrast Stretching

And there are also analytical expressions

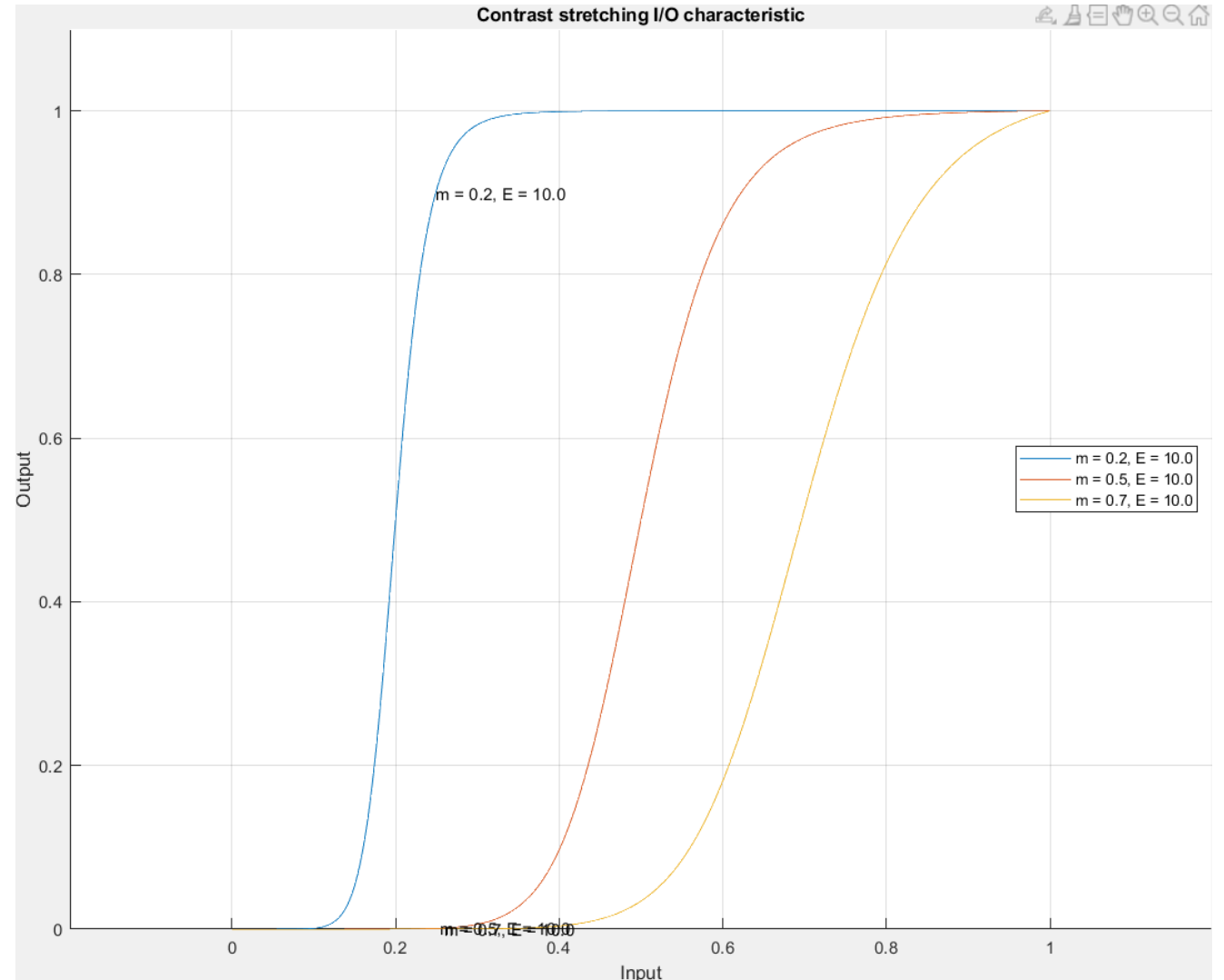
$$I(r, c) \rightarrow \frac{1 + m^e}{\left(1 + \left(\frac{m}{I(r, c) + \epsilon}\right)\right)^e}$$



Contrast Stretching

And there are also analytical expressions

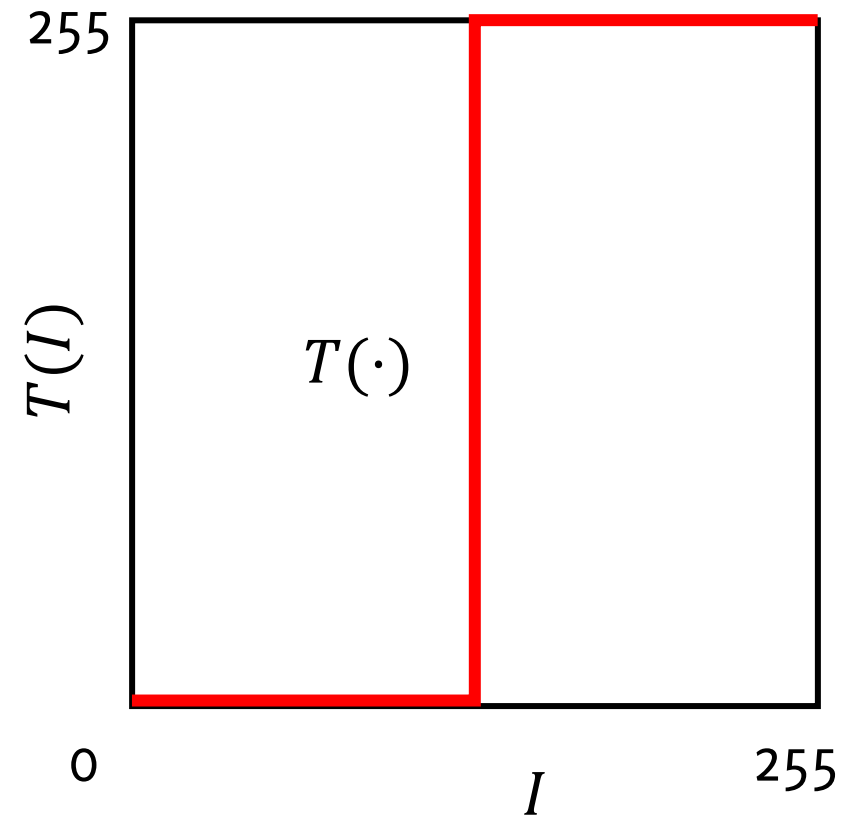
$$I(r, c) \rightarrow \frac{1 + m^e}{\left(1 + \left(\frac{m}{I(r, c) + \epsilon}\right)\right)^e}$$



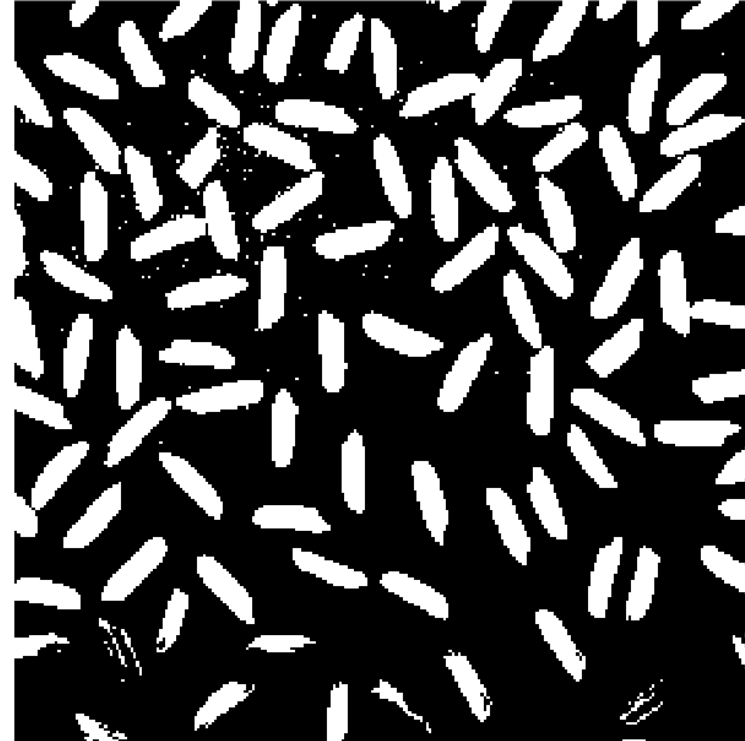
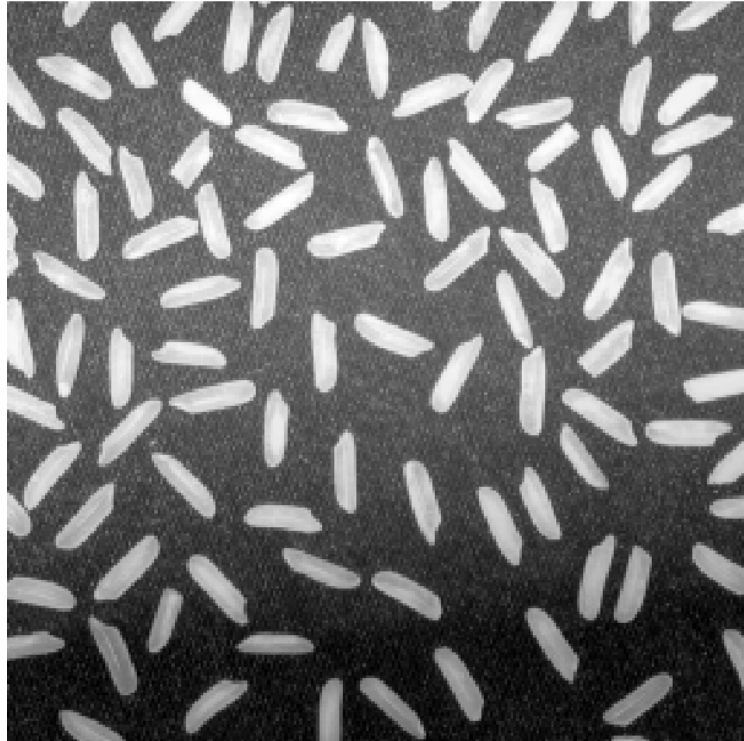
Gray-level mapping

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

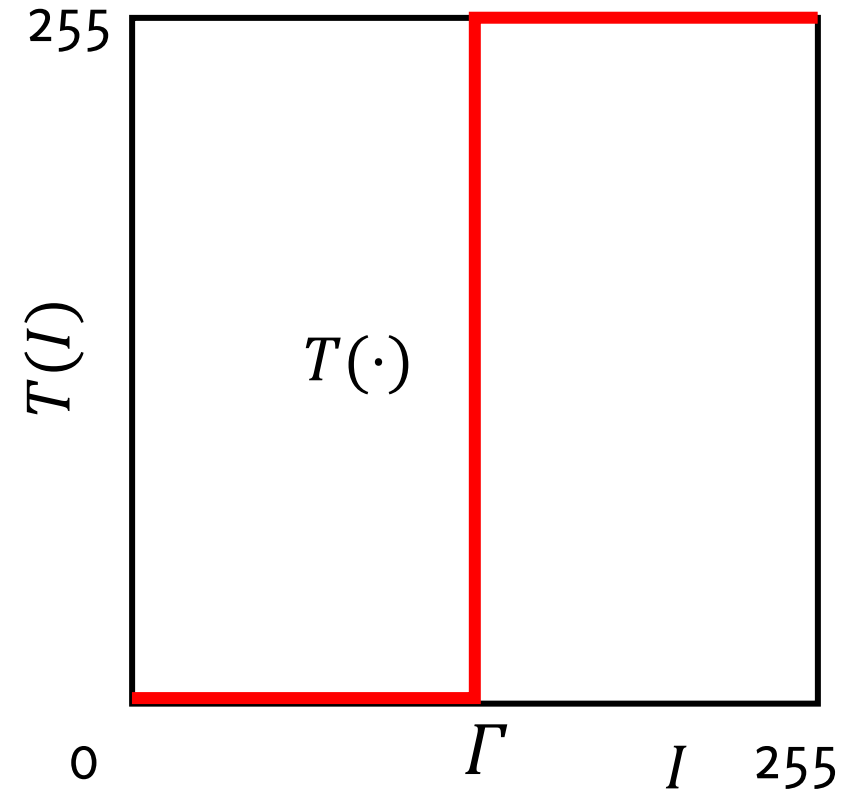


Thresholding binarizes images

Thresholding

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

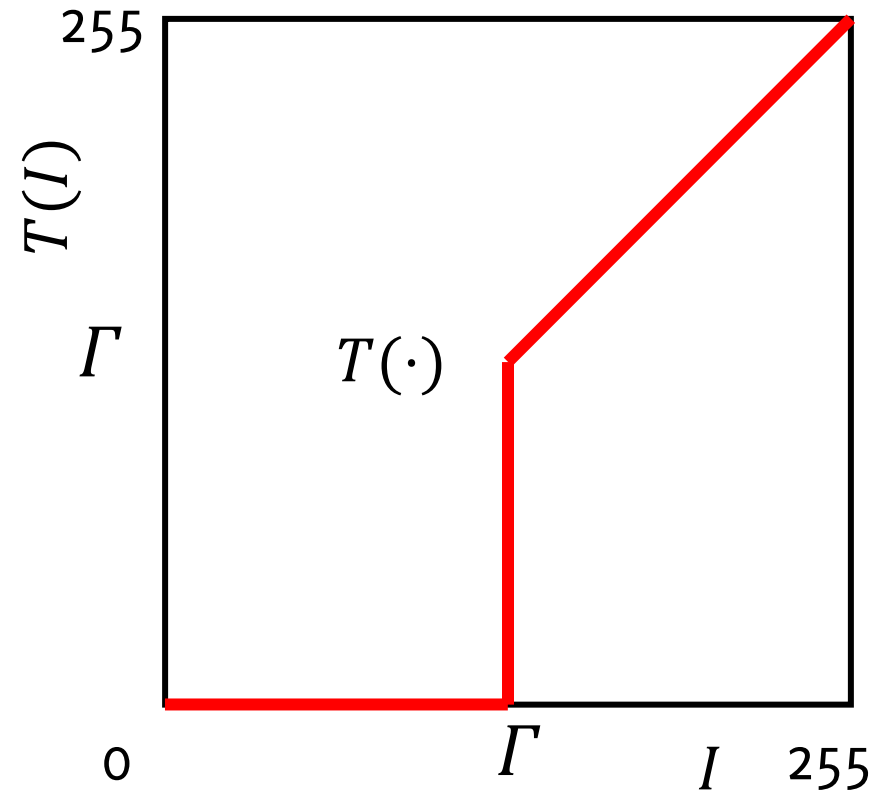
$$T(I(r, c)) = \begin{cases} 255, & \text{if } I(r, c) \geq \Gamma \\ 0, & \text{if } I(r, c) < \Gamma \end{cases}$$



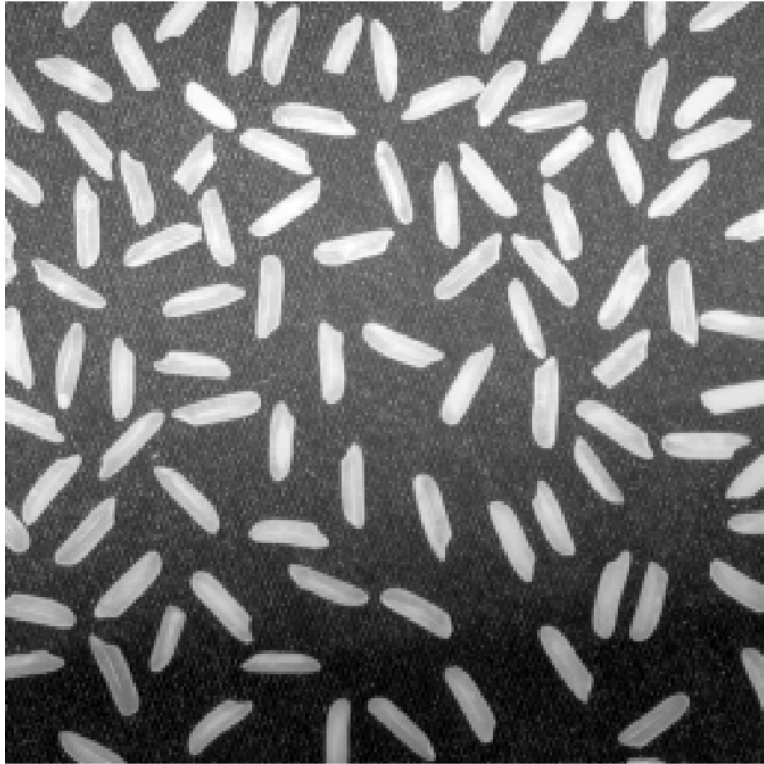
Thresholding

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

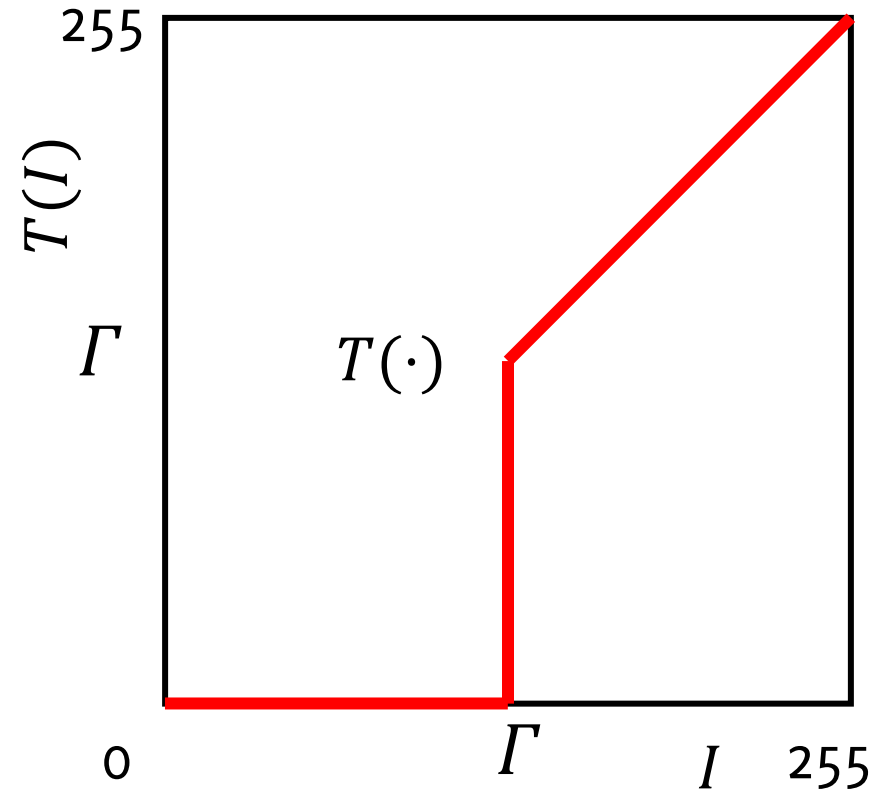


Thresholding

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: the ReLU Layers



Local (Spatial) Transformations: Correlation and Convolution

Local (Spatial) Transformation

In general, these can be written as

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

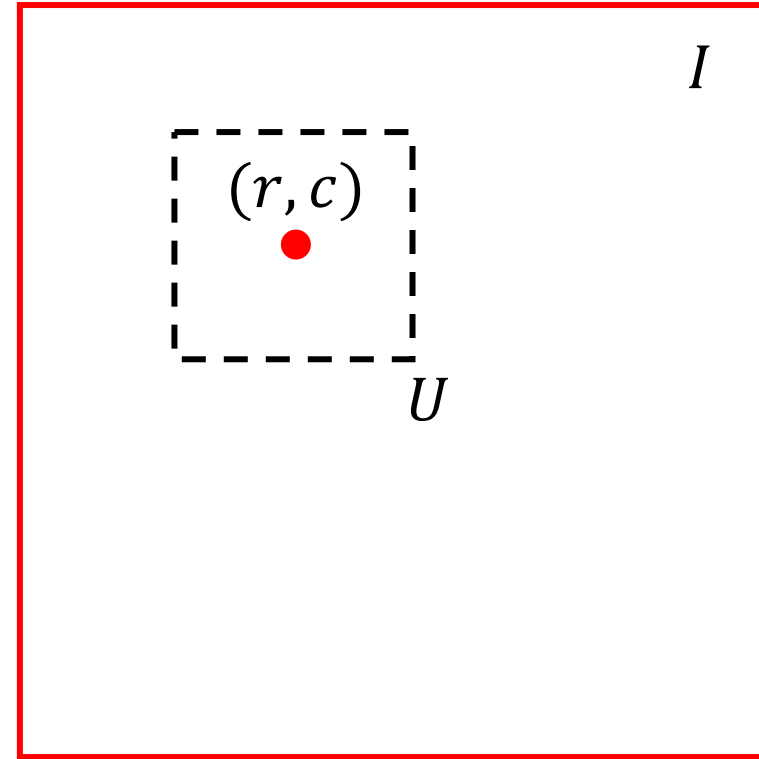
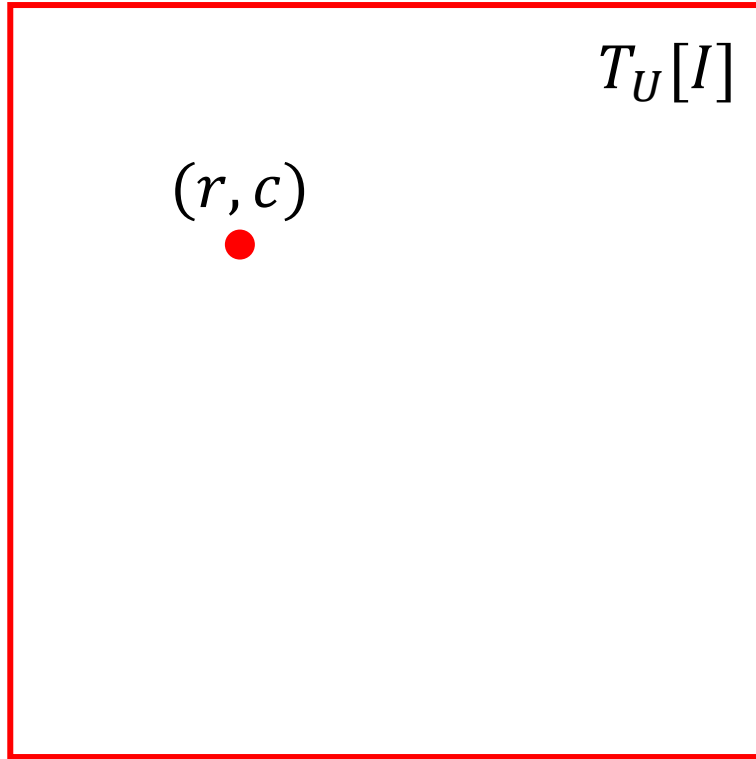
Where

- 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“ the pixel (r, c) on a specific neighborhood U . In particular, $T_U[I](r, c)$ is computed from all the intensity values: $\{I(u, v), (u - r, v - c) \in U\}$

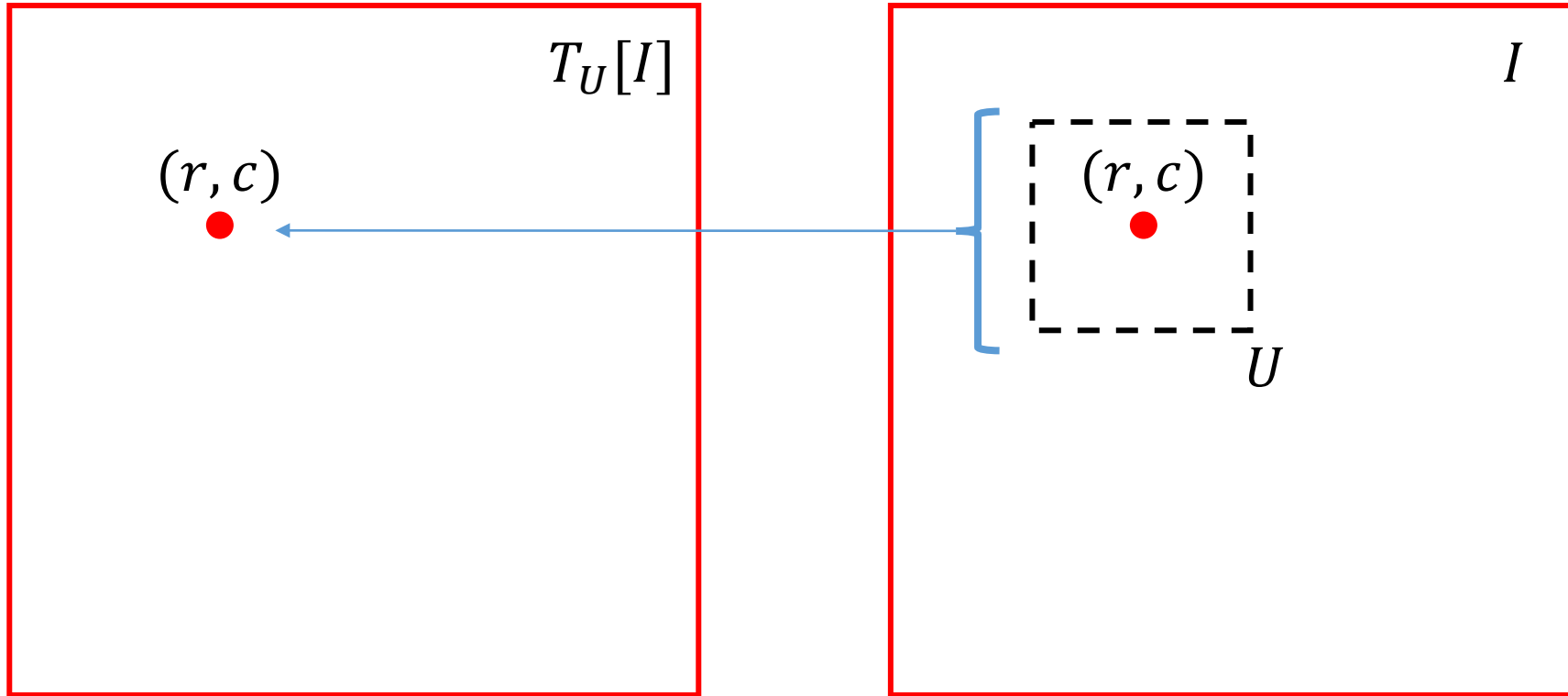
Local (Spatial) Filters

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



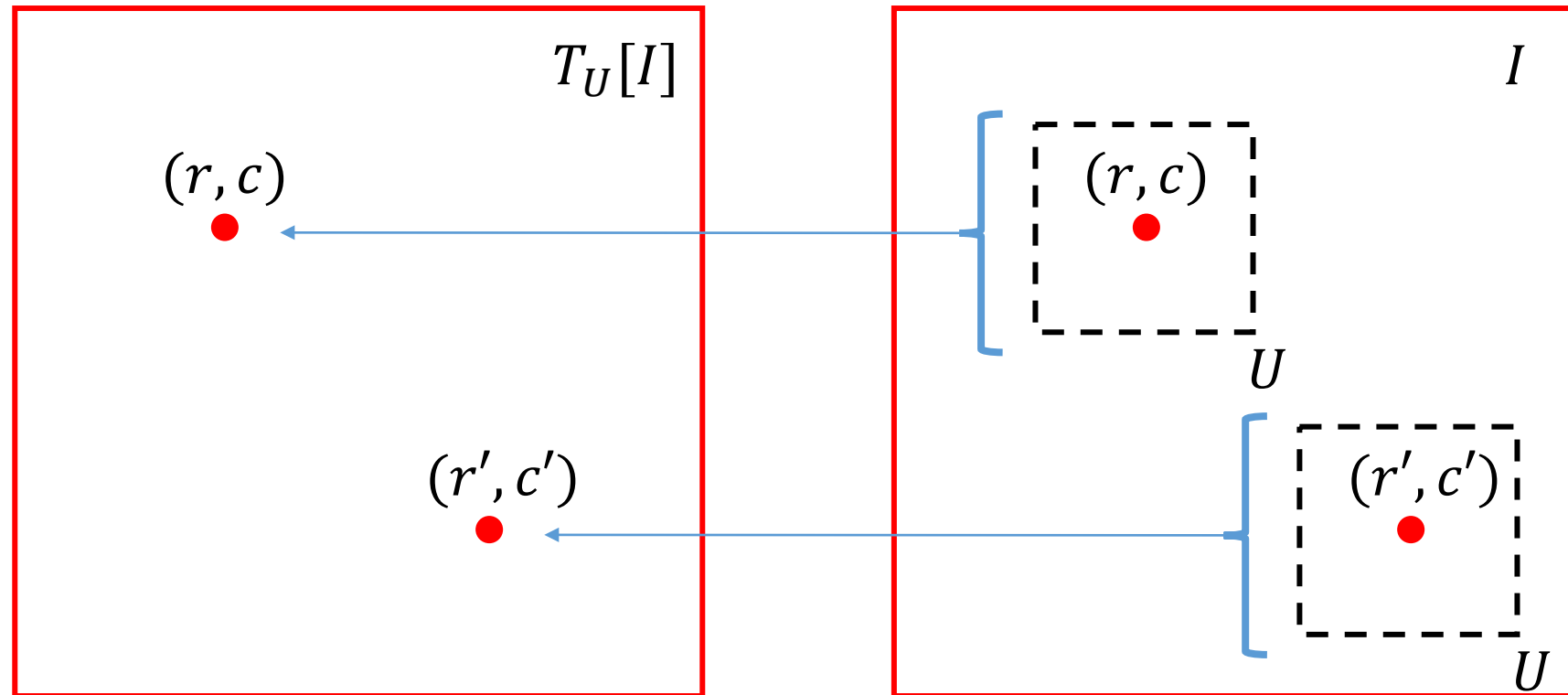
Local (Spatial) Filters

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



Local (Spatial) Filters

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



- The location of the output does not change
- **Space invariant transformations** are repeated for each pixel
- T can be either linear or nonlinear

Local Linear Filters

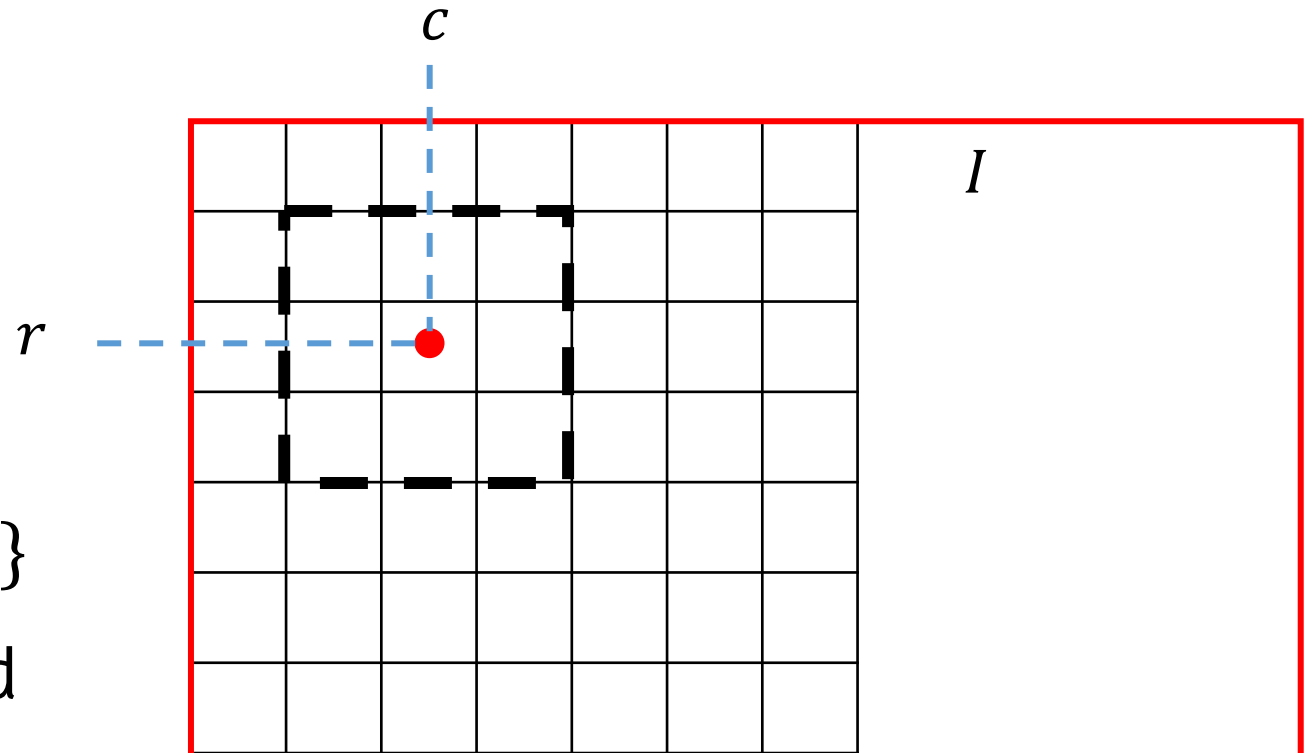
Linear Transformation: Linearity implies that

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

Considering *some weights* $\{w_i\}$ and (u, v) has to be interpreted as a "displacement vector" w.r.t. the neighborhood center (r, c) , e.g.,

$$(u, v) \in \{(1, -1), (1, 0), (1, 1) \dots\}$$

We can now think the neighborhood U as set of displacement vectors.



Local Linear Filters

Linear Transformation: the filter weights can be associated to a matrix \mathbf{w}

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

The diagram illustrates the linear transformation. On the left, a 3x3 grid labeled \mathbf{w} contains the filter weights: $w(-1, -1)$, $w(-1, 0)$, $w(-1, 1)$ in the top row; $w(0, -1)$, $w(0, 0)$, $w(0, 1)$ in the middle row; and $w(1, -1)$, $w(1, 0)$, $w(1, 1)$ in the bottom row. The grid is enclosed in a dashed black border. To the right, a larger grid labeled I represents the input image. A red dot is placed at the center of the grid, with dashed blue lines extending to the left and top, labeled r and c respectively, indicating the coordinates of the filter's application.

We can consider weights as an image, or a **filter h**
The filter h entirely defines this operation

Local Linear Filters

Linear Transformation: the filter weights can be associated to a matrix \mathbf{w}

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

The diagram shows a 3x3 filter matrix \mathbf{w} on the left, with elements $w(-1, -1)$, $w(-1, 0)$, $w(-1, 1)$ in the top row; $w(0, -1)$, $w(0, 0)$, $w(0, 1)$ in the middle row; and $w(1, -1)$, $w(1, 0)$, $w(1, 1)$ in the bottom row. The matrix is enclosed in a dashed black border. To the right, a grid representing the input image I is shown. A red dot marks the center pixel at coordinates (r, c) . A dashed blue line extends from r to the row and from c to the column of the red dot. A 3x3 neighborhood around the red dot is highlighted with a thick black border, corresponding to the filter matrix \mathbf{w} .

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

Local Linear Filters

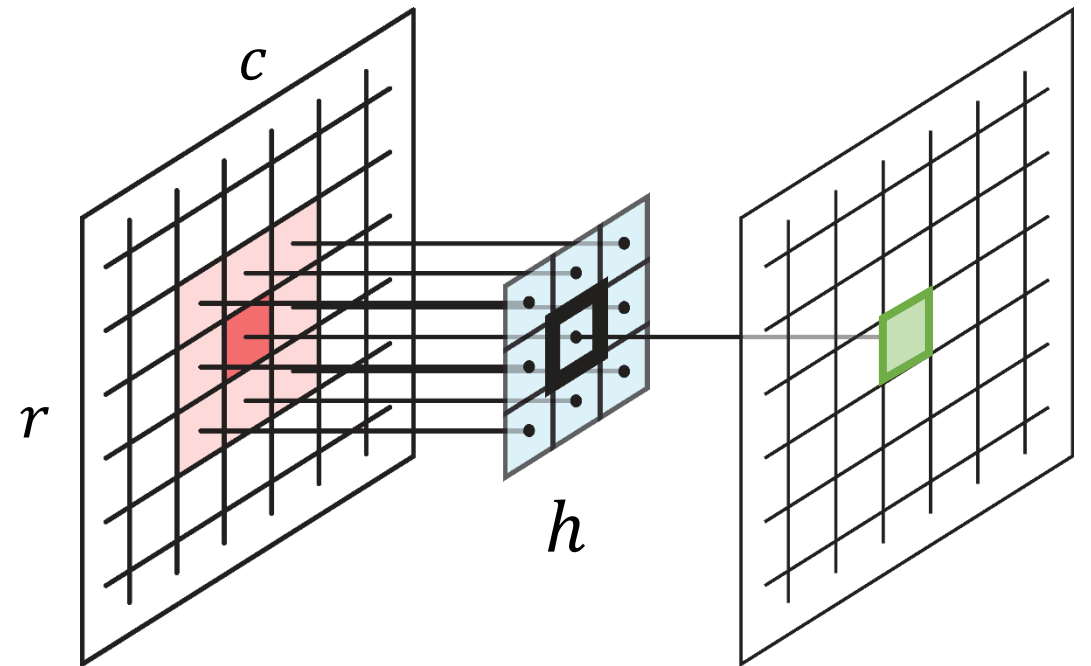
Linear Transformation: Linearity implies that the **output** $T[I](r, c)$ is a linear combination of the pixels in U :

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

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

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

The filter h entirely defines this operation



Local Linear Filters

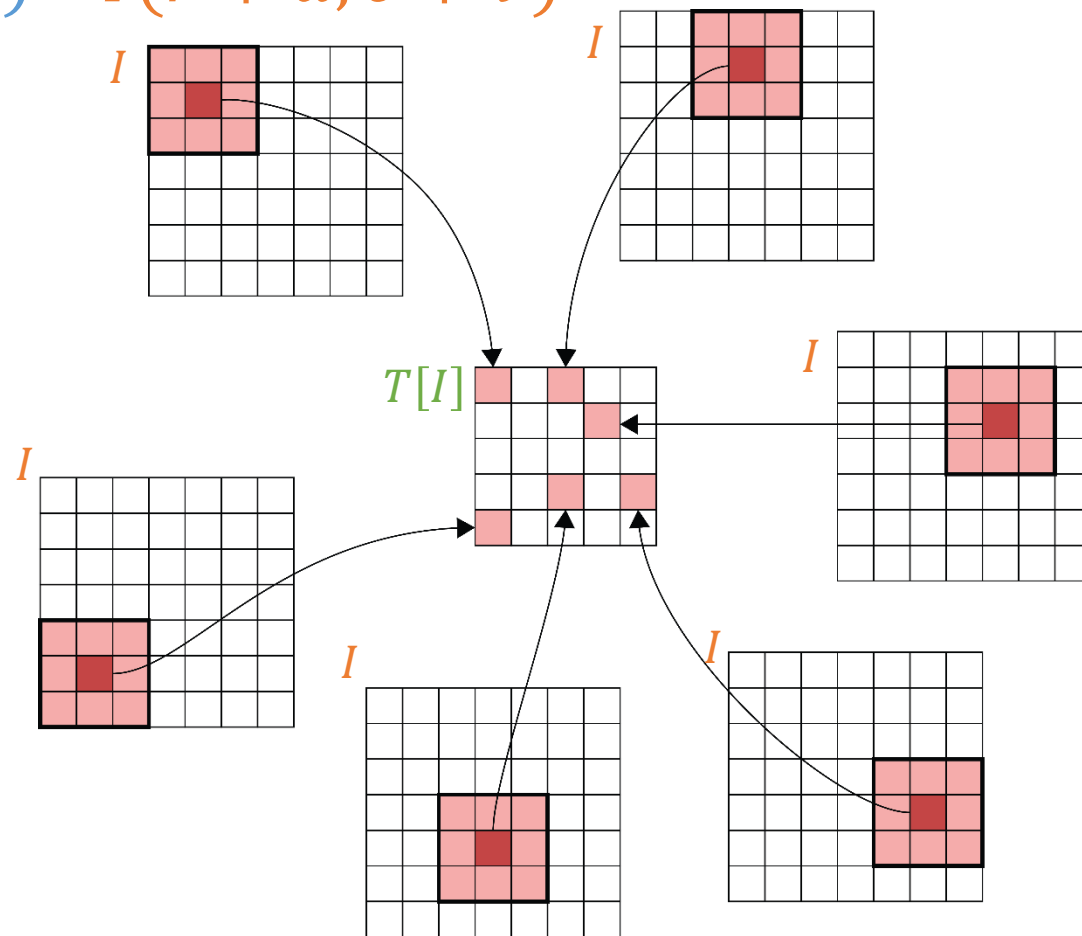
Linear Transformation: the filter weights can be associated to a matrix \mathbf{w}

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

\mathbf{w}

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

This operation is repeated for each pixel in the input image

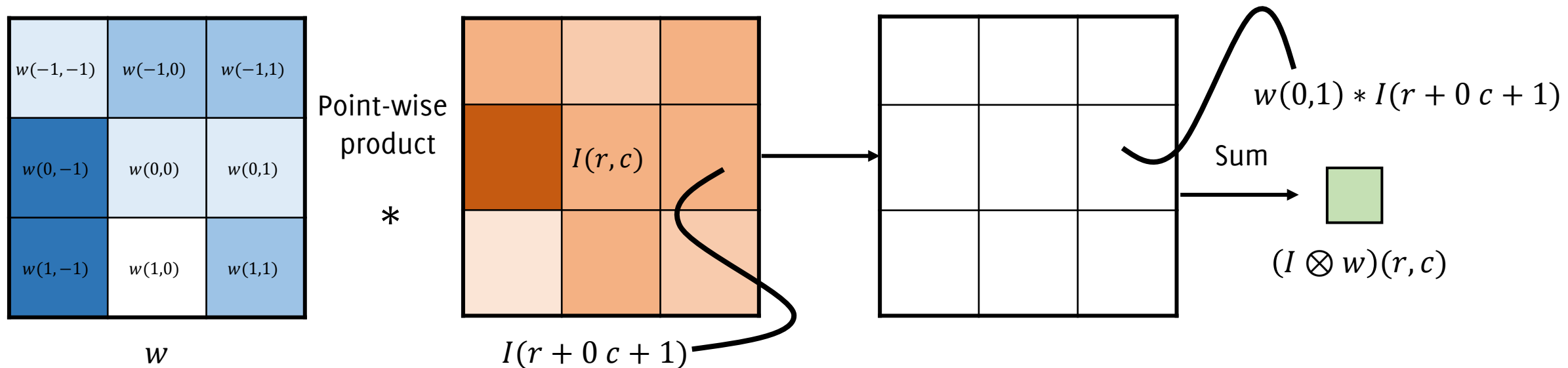


Correlation

The **correlation** among a filter $w = \{w_{ij}\}$ and an image is defined as

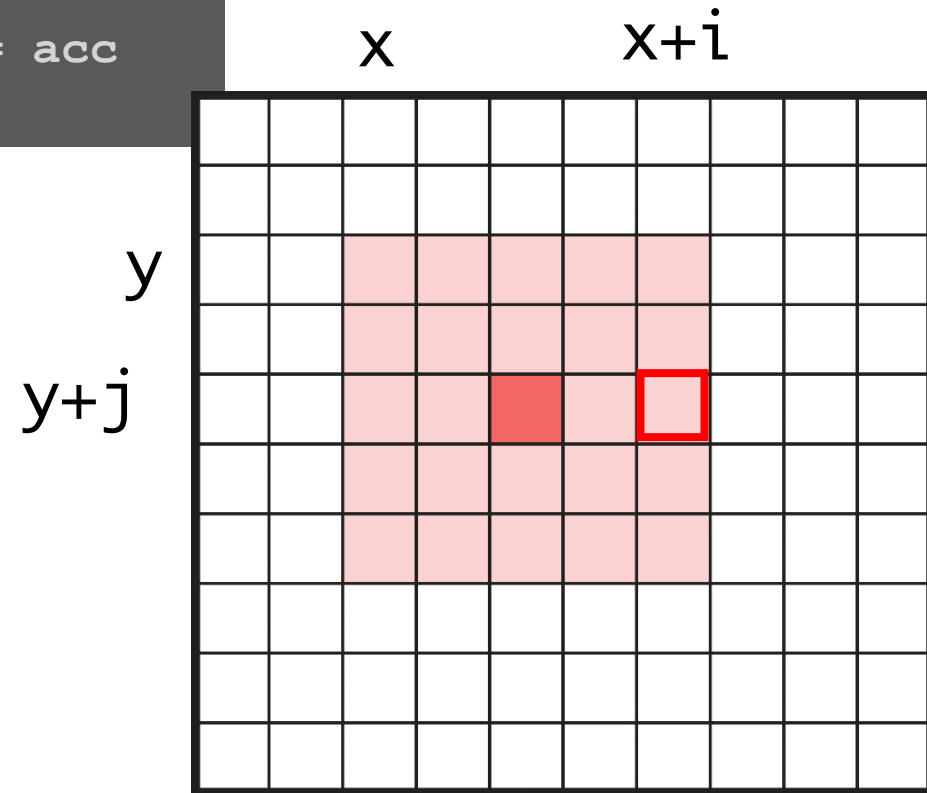
$$(I \otimes w)(r, c) = \sum_{u=-L}^L \sum_{v=-L}^L w(u, v) * I(r + u, c + v)$$

where the filter h is of size $(2L + 1) \times (2L + 1)$ and contains the weights defined before as w . The filter w is also sometimes called “kernel”



Correlation

```
acc = 0;
for i in np.arange(template_height)
    for j in np.arange(template_width)
        acc = acc + image[y + i, x + j]*template[i,j]
image[x+ template_height//2, y + template_width//2] = acc
```



Correlation for BINARY target matching

I

W

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



NO
NO
NO

=

Easy to understand with binary images

Target used as a filter

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



NO
NO
NO

NOI	QRM1	DIF1	Det1	#FA1
NO3	.201	0.145	NO	2.000
NO3	.794	0.142	NO	2.000
	0.765	0.409	NO	6.000



NO
NO
NO

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



NO

NO

NO

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

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



NO

NO

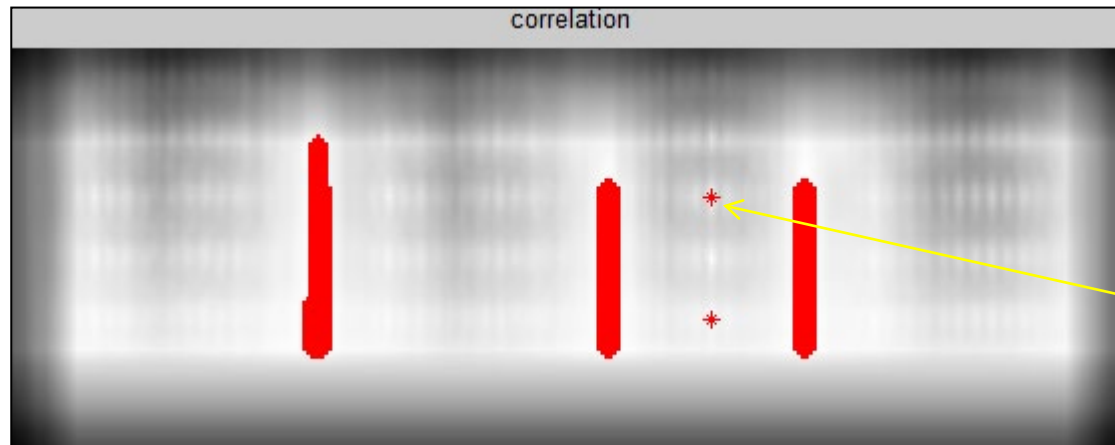
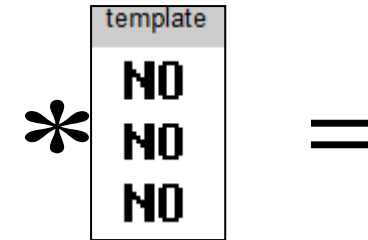
NO

The maximum is here



However...

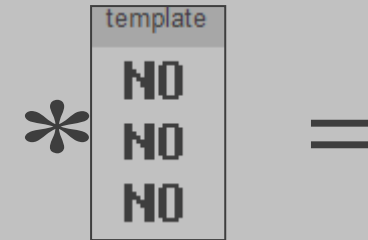
y, original image			
IQRM1	DIF1	Det1	#FA1
0.201	0.145	NO	2.000
0.794	0.142	NO	2.000
0.765	0.409	NO	6.000



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

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



Normalization is needed when using correlation for template matching!



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 $NCC(A, B) \in [-1, 1]$ is defined as

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

Where in our case,

- A is the region in the image,
 - B is the filter
- and they are comparable in size

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

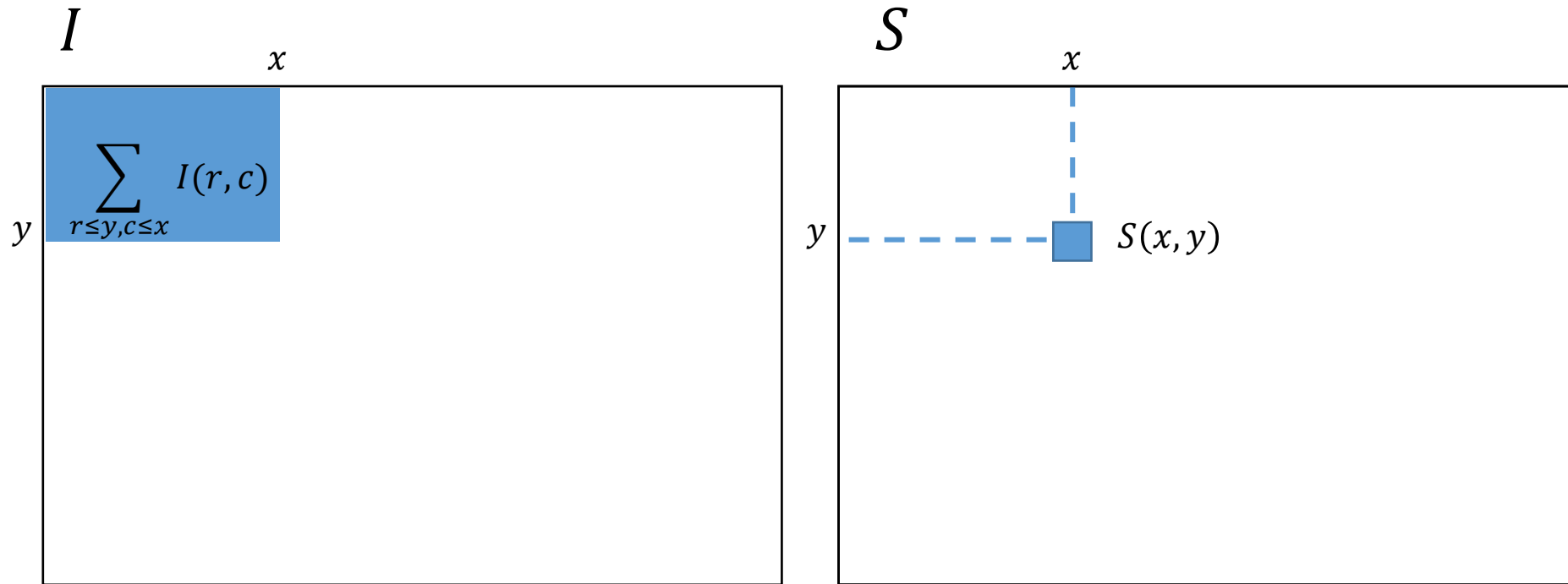
Remarks:

- NCC yields a measure in the range $[-1,1]$,
- 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

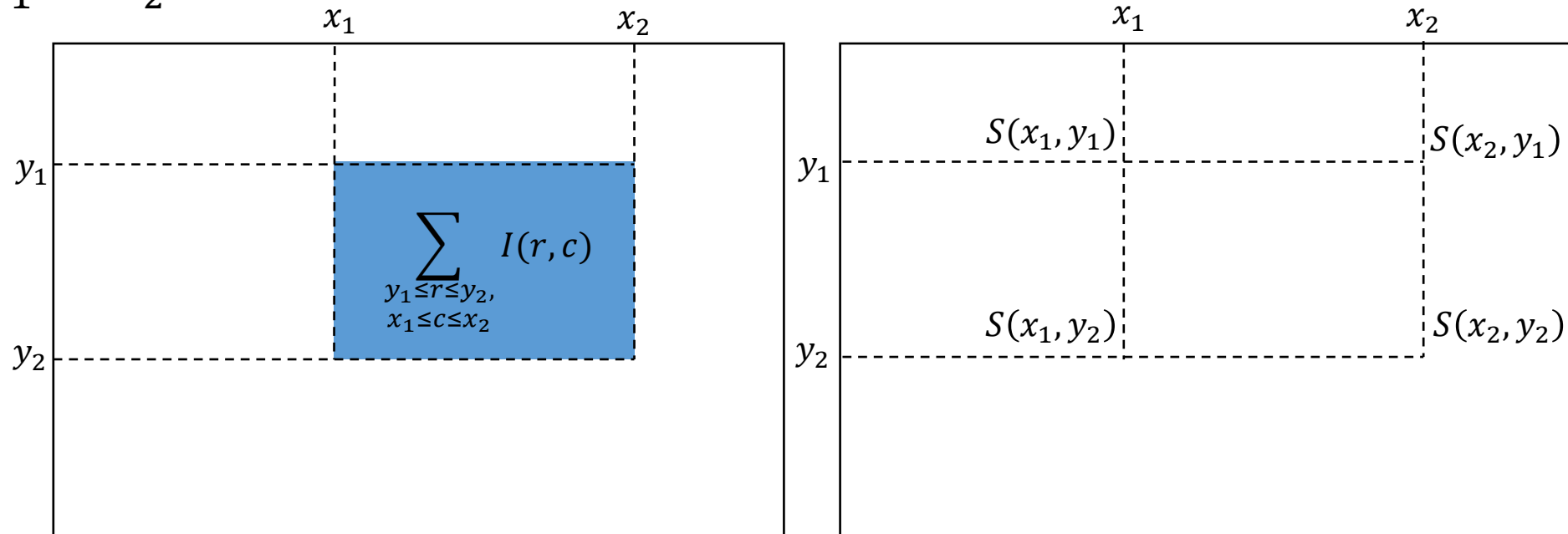
$$S(x, y) = \sum_{r \leq y, c \leq x} I(r, c)$$



Using the Integral Image

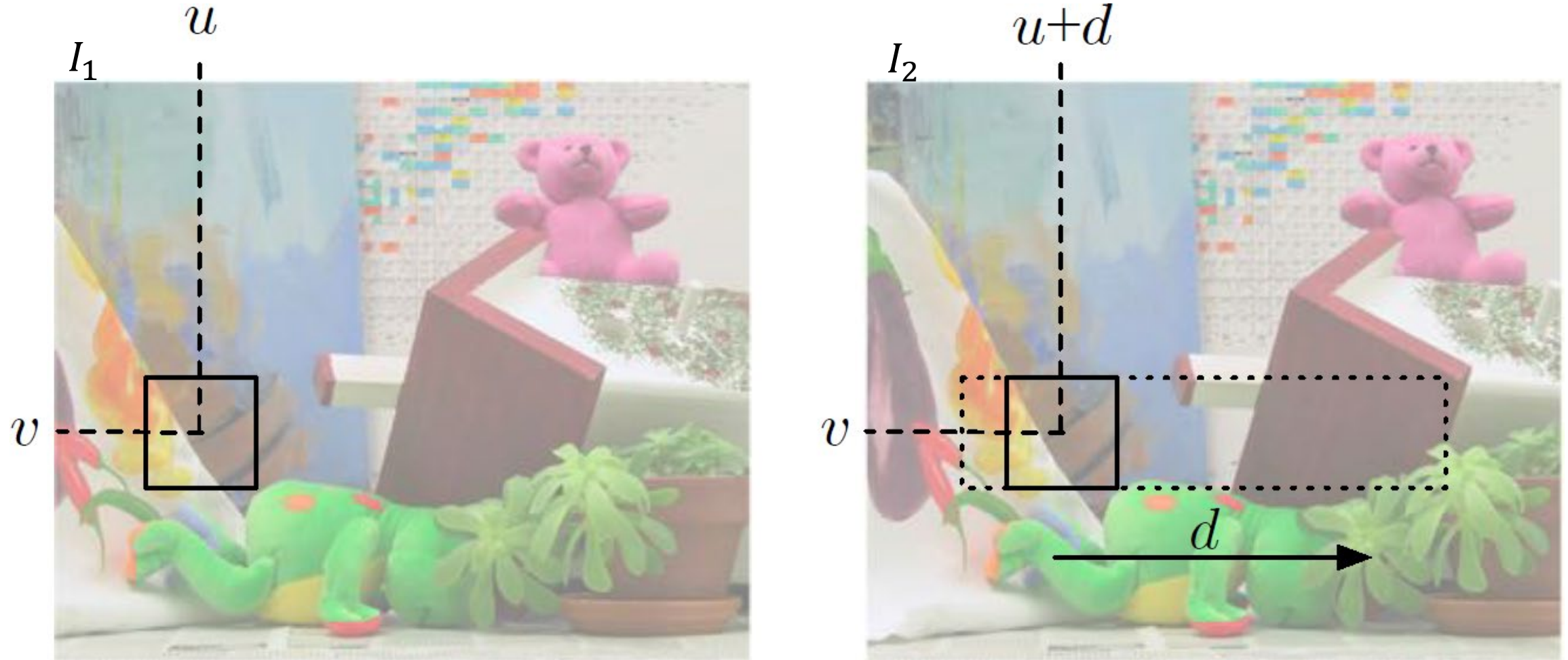
The integral image allows fast computation of the sum (average) of any rectangular region in the image

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



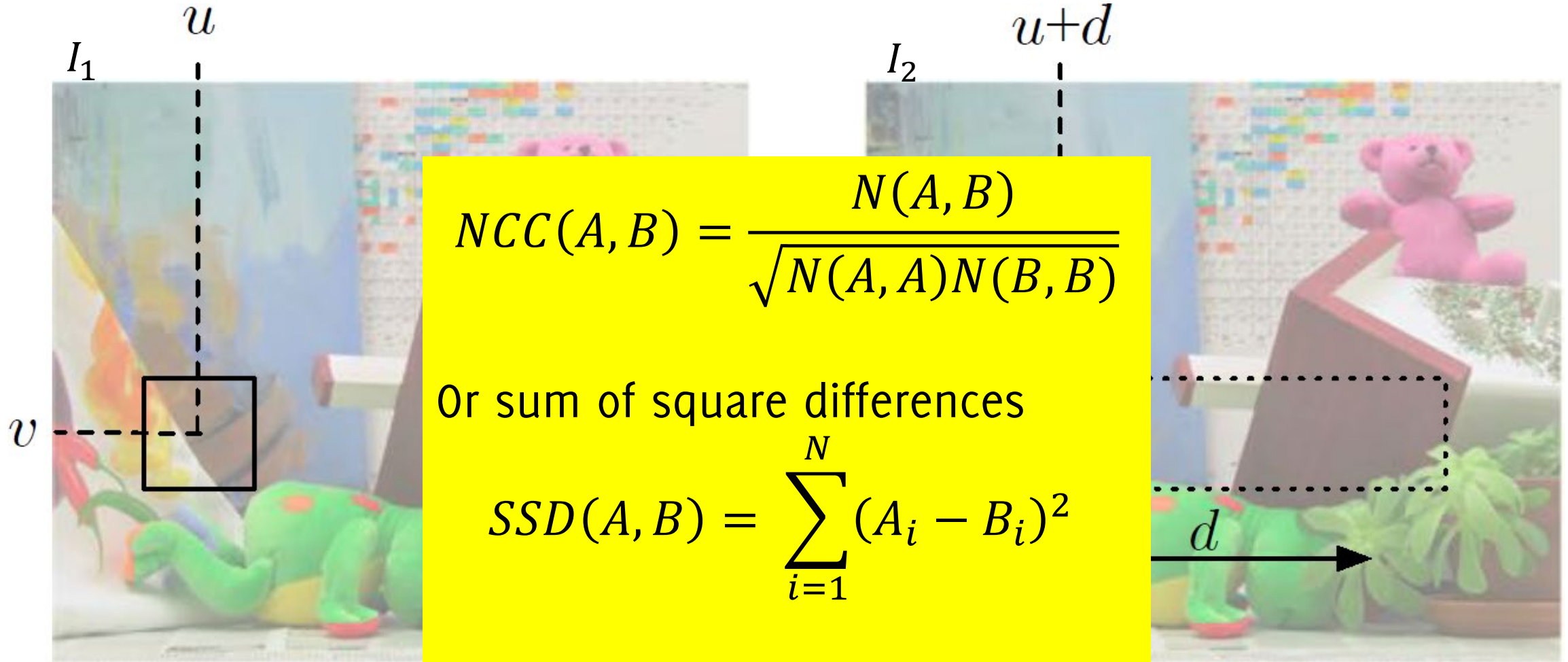
Disparity Map Estimation

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



I_1 u

I_2 $u+d$

v

d

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

Or sum of square differences

$$SSD(A, B) = \sum_{i=1}^N (A_i - B_i)^2$$

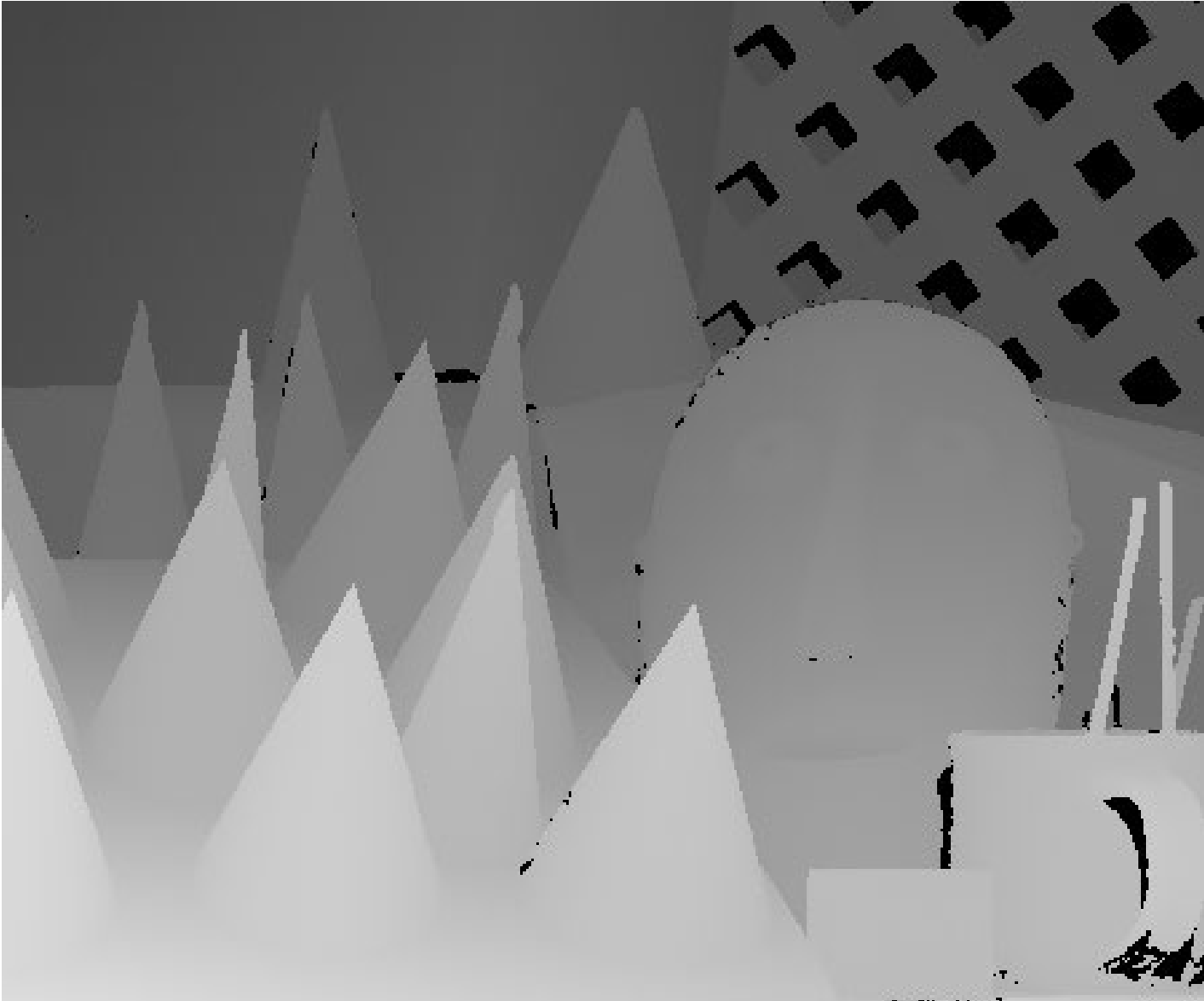
Stereo Pairs <http://vision.middlebury.edu/stereo/data/>



Stereo Pairs <http://vision.middlebury.edu/stereo/data/>



Stereo Pairs <http://vision.middlebury.edu/stereo/data/>



Convolution

Correlation and Convolution

The **correlation** among a filter \mathbf{w} and an image is defined as

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

where the filter \mathbf{w} is of size $(2L + 1) \times (2L + 1)$

The **convolution** among a filter \mathbf{w} and an image is defined as

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

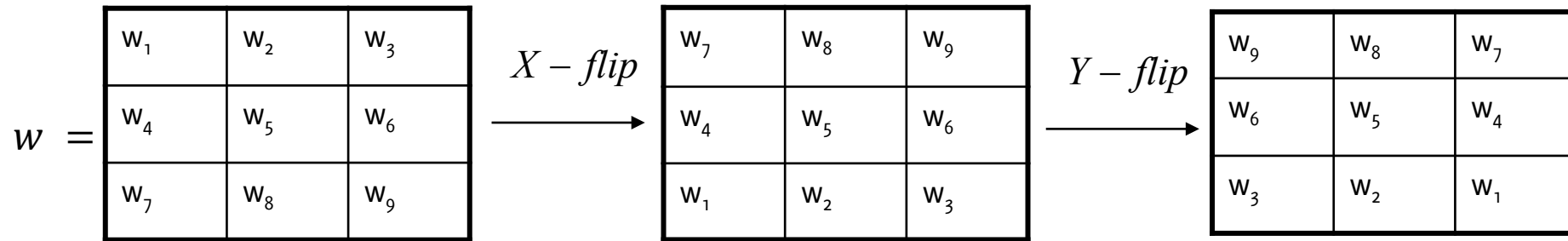
where the filter \mathbf{w} is of size $(2L + 1) \times (2L + 1)$

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

Convolution – and filter flip

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

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



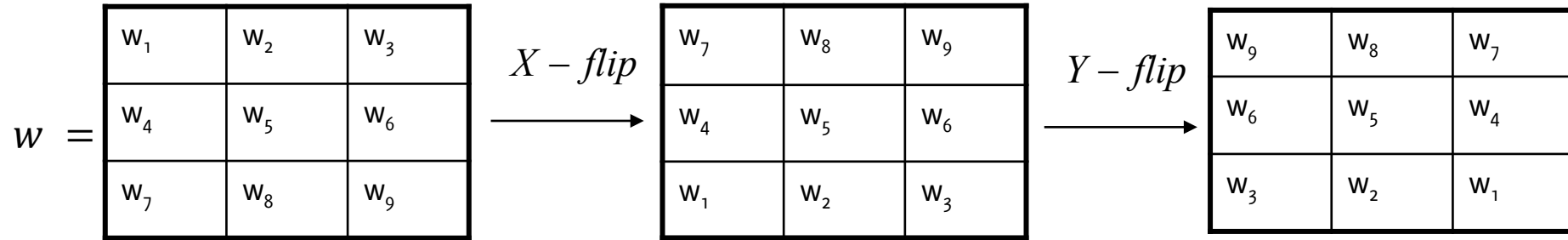
In this particular case $L = 1$ and both the image and the filter have size 3×3

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

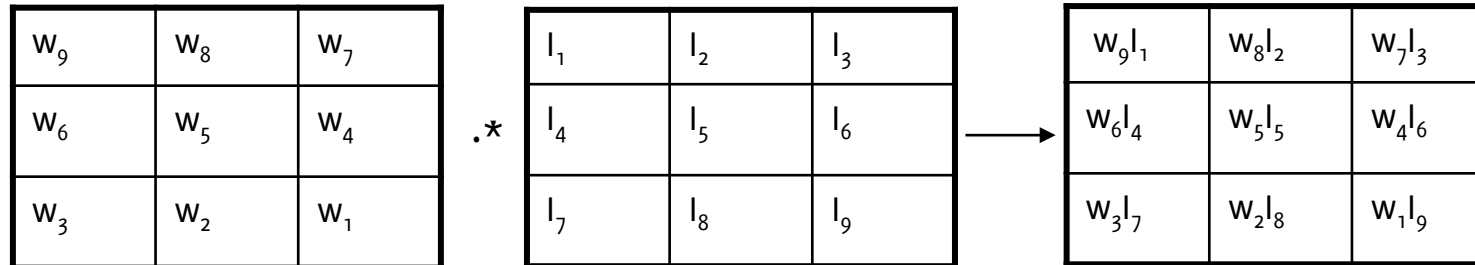
Convolution – and filter flip

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

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



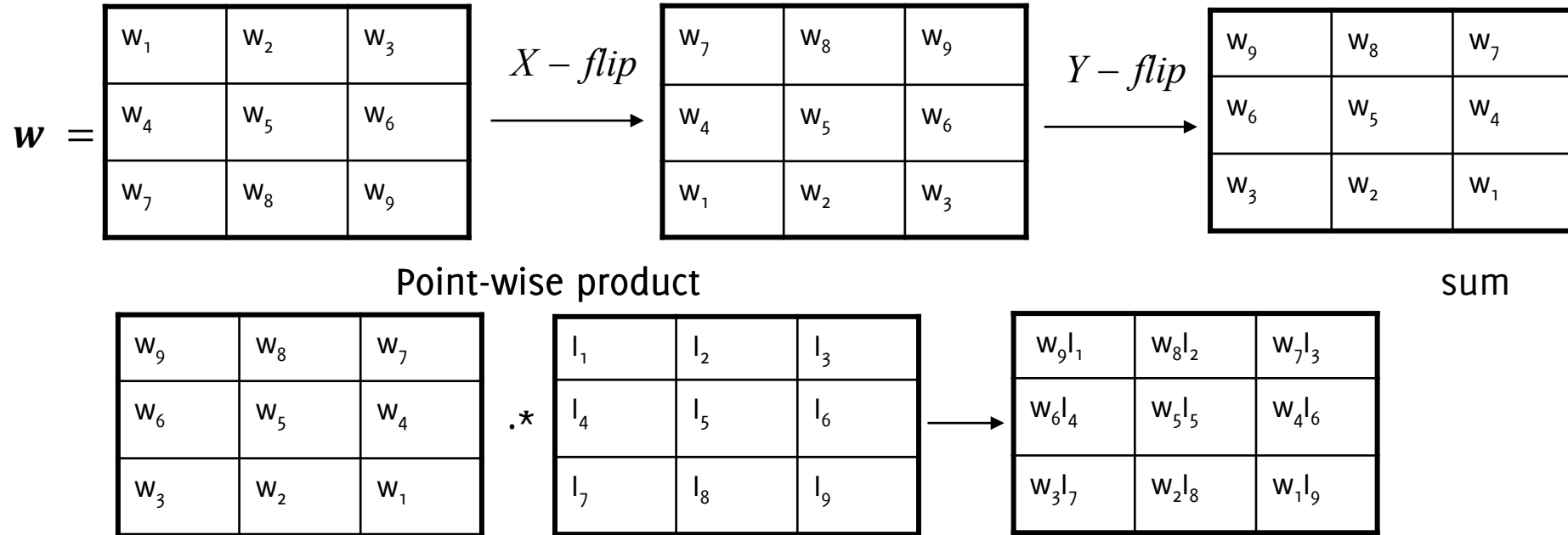
Point-wise product



Convolution

Let I, \mathbf{w} be two discrete 2D signals of $(2L + 1) \times (2L + 1)$

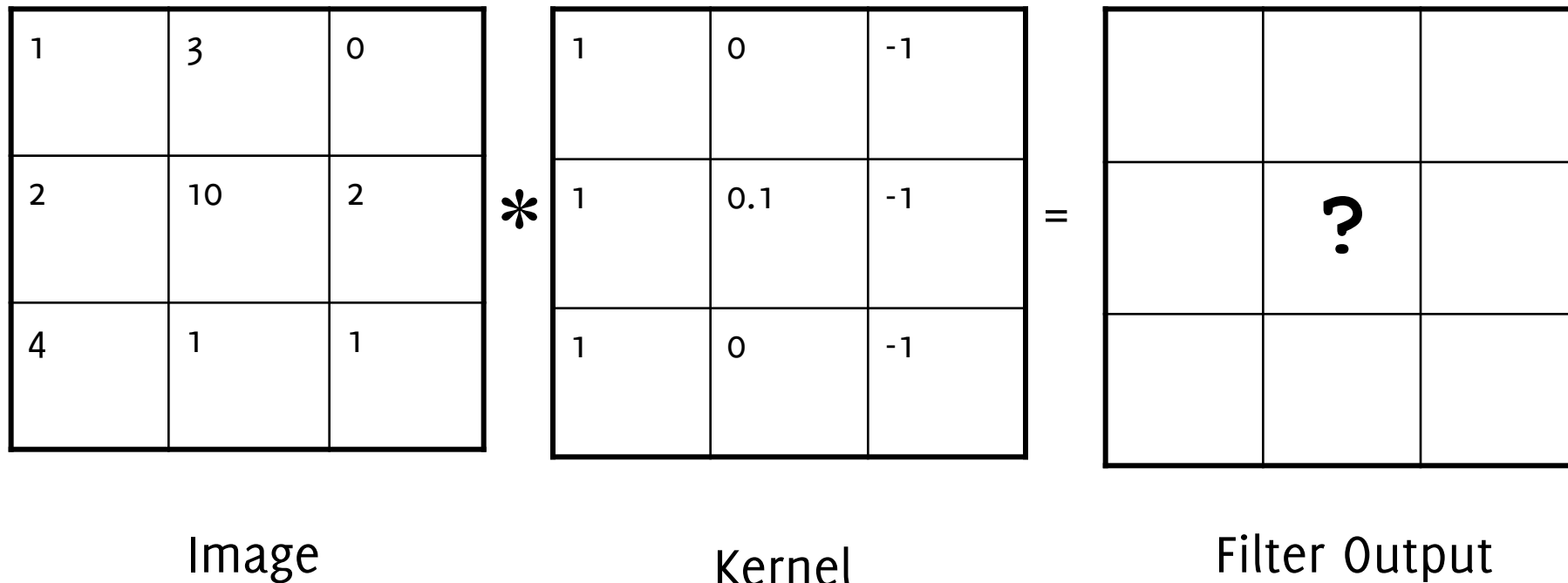
$$G(r, c) = (I \circledast \mathbf{w})(r, c) = \sum_{u=-L}^L \sum_{v=-L}^L I(r + u, c + v)w(-u, -v)$$



$$G_5 = w_9 I_1 + w_8 I_2 + w_7 I_3 + w_6 I_4 + w_5 I_5 + w_5 I_6 + w_3 I_7 + w_2 I_8 + w_1 I_9$$

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 \mathbf{w} .

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

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

that is equivalent to

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

- For discrete signals and filters

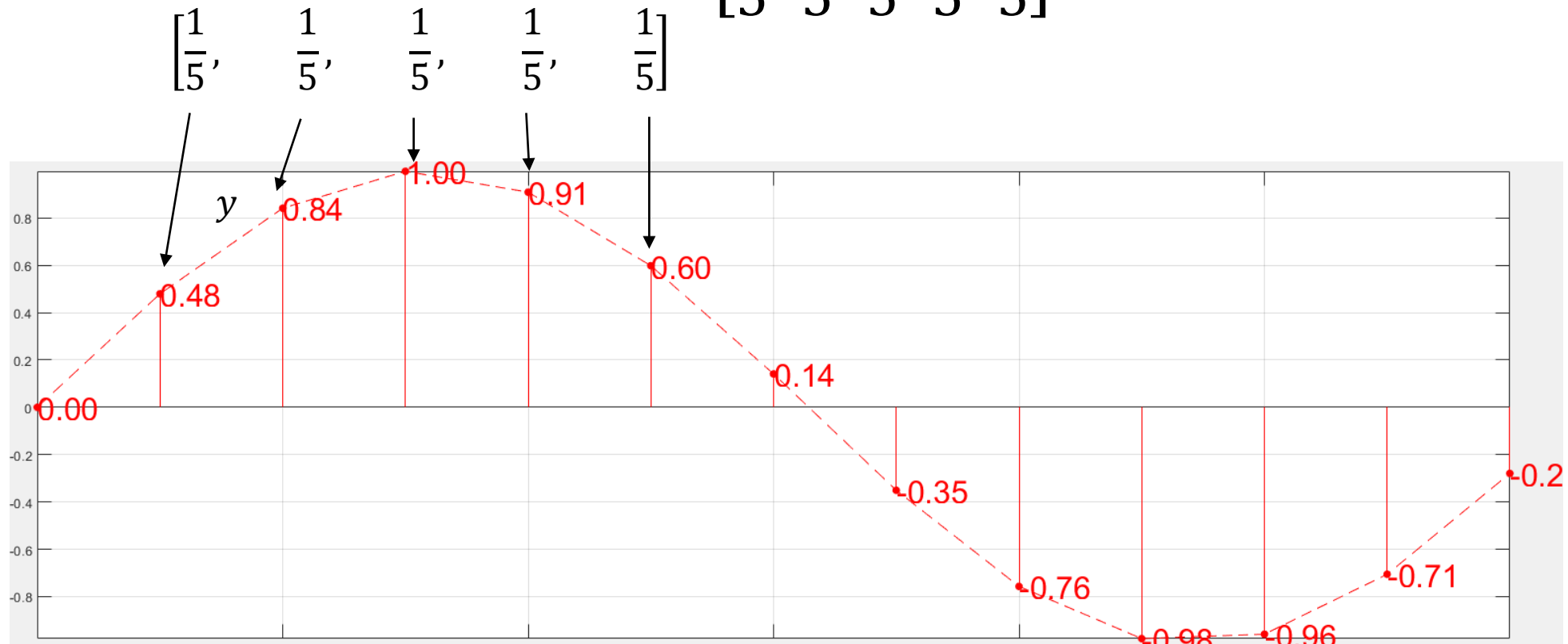
$$z(n) = (y \otimes \mathbf{w})(n) = \sum_{m=-L}^L y(n - m) \mathbf{w}(m)$$

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

1D Convolution - example

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

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

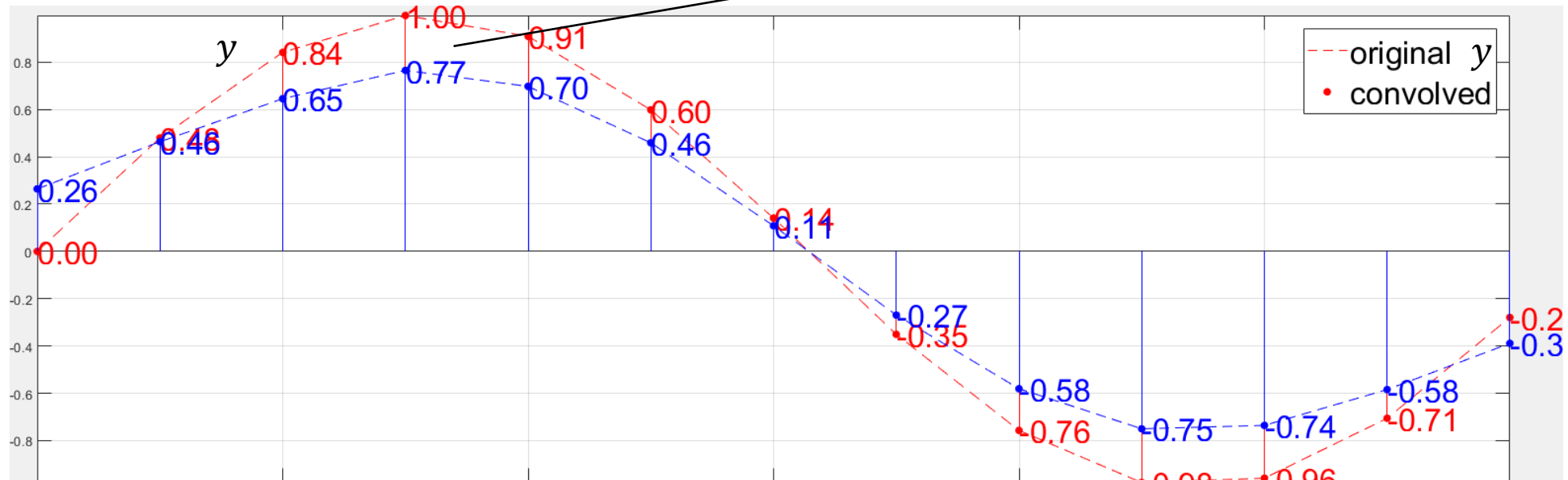


1D Convolution - example

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

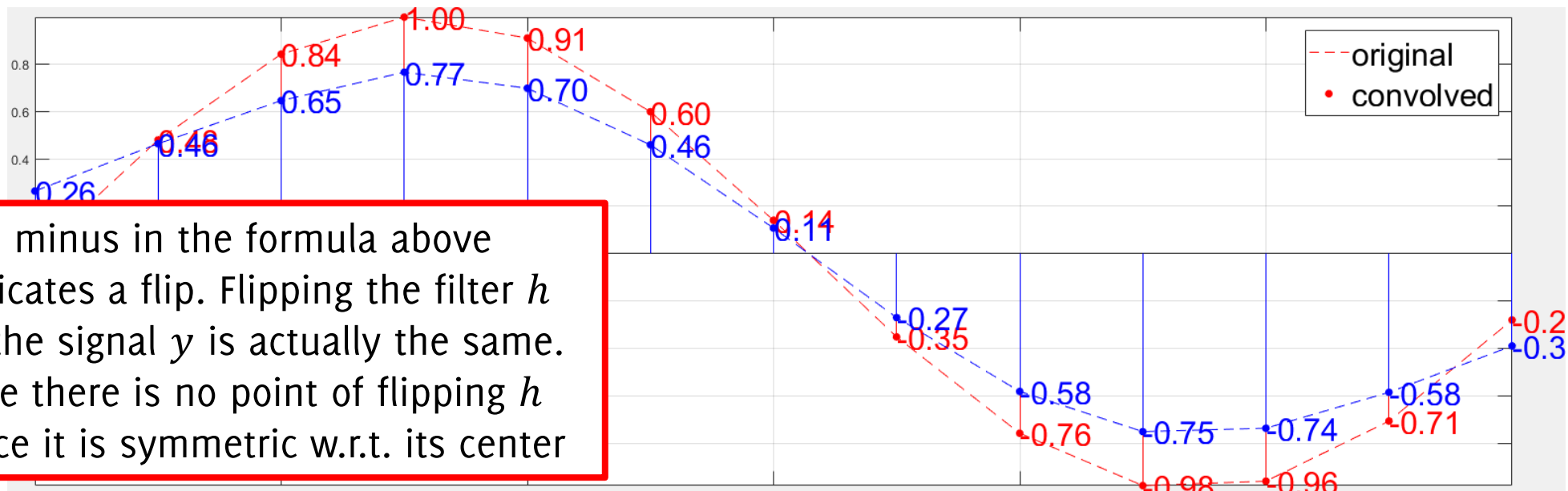
$$y = \sin(x), \mathbf{w} = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5} \right], L = 2$$

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

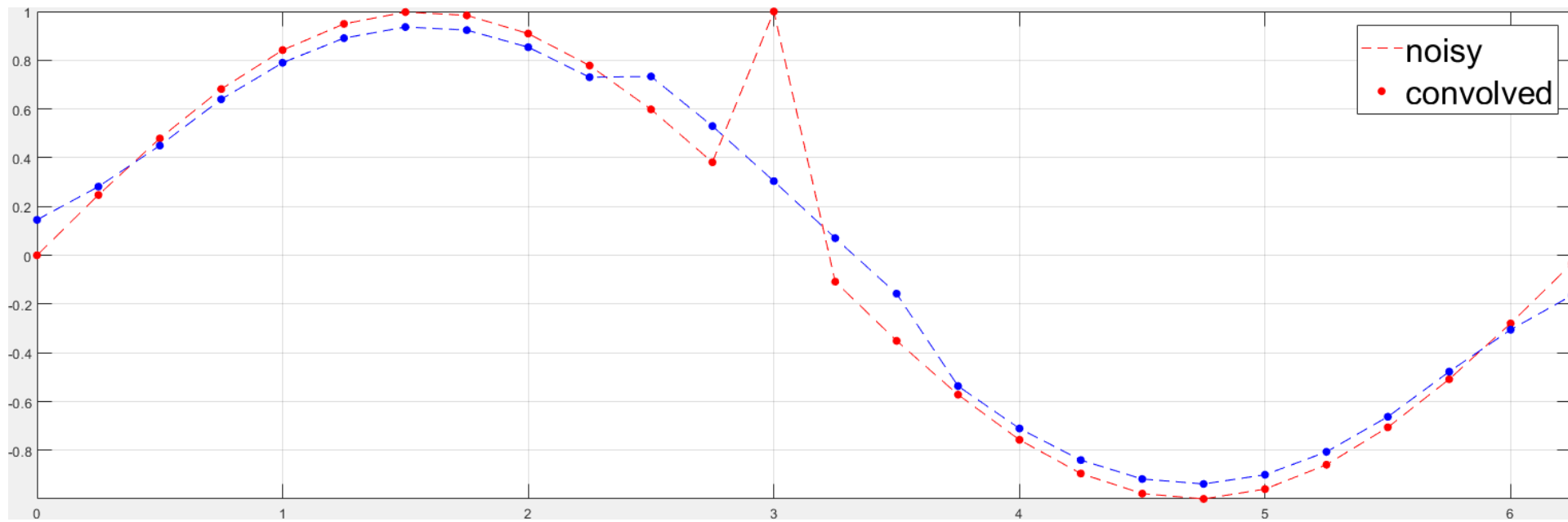


1D Convolution - example

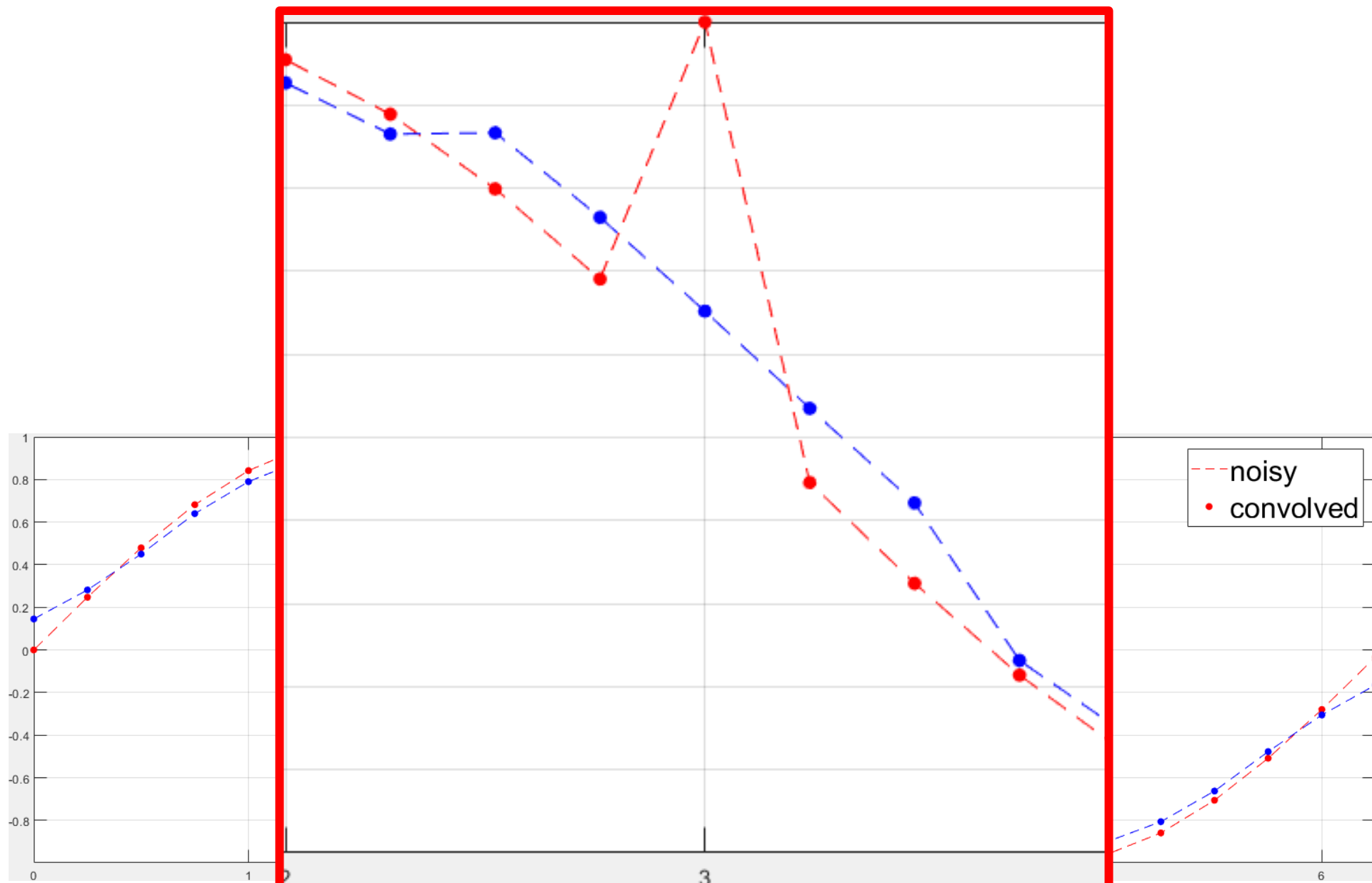
$$z(n) = (y \otimes \mathbf{w})(n) = \sum_{m=-L}^L y(n-m)\mathbf{w}(m)$$
$$= \sum_{m=-L}^L y(n+m)\mathbf{w}(-m)$$



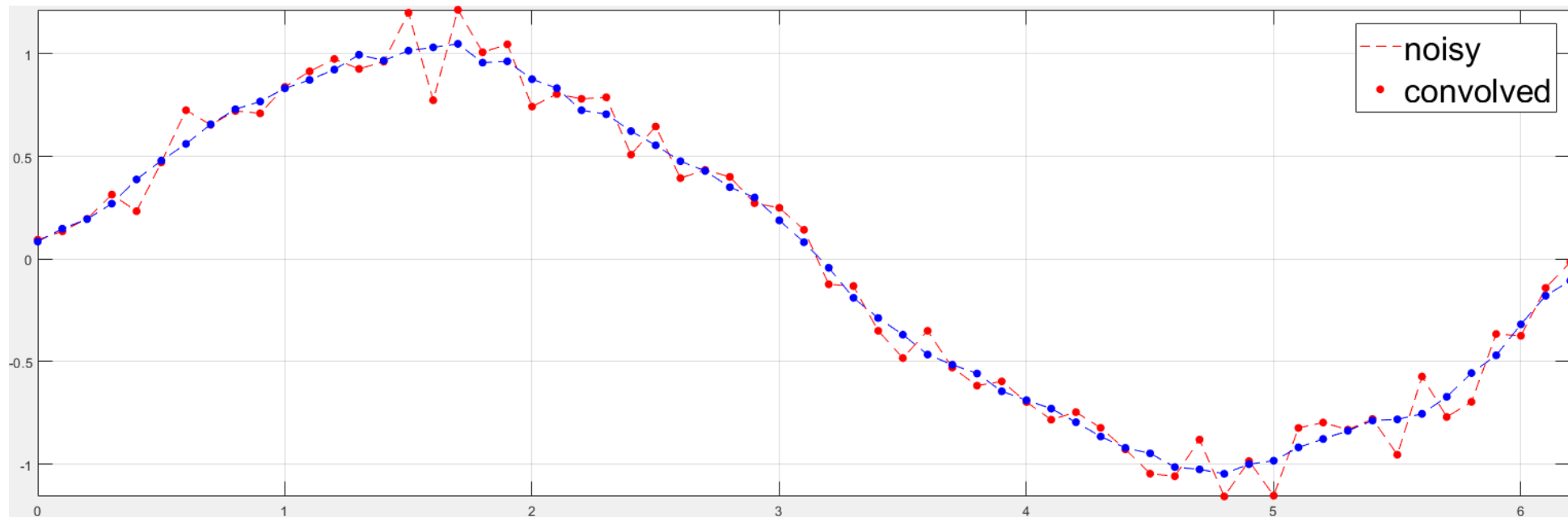
What about an imupulse?



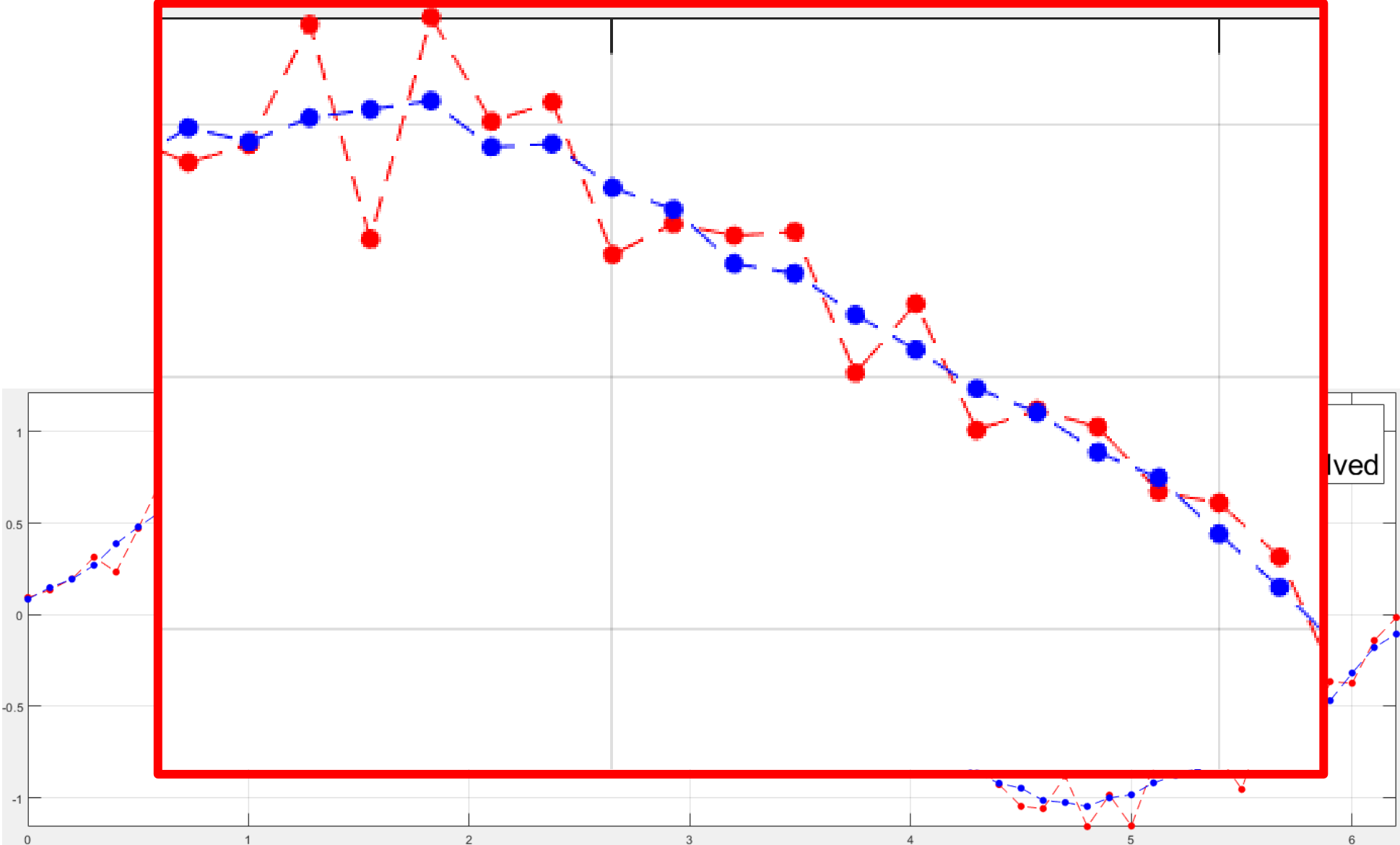
What about an impulse?



What about noise?



What about noise?

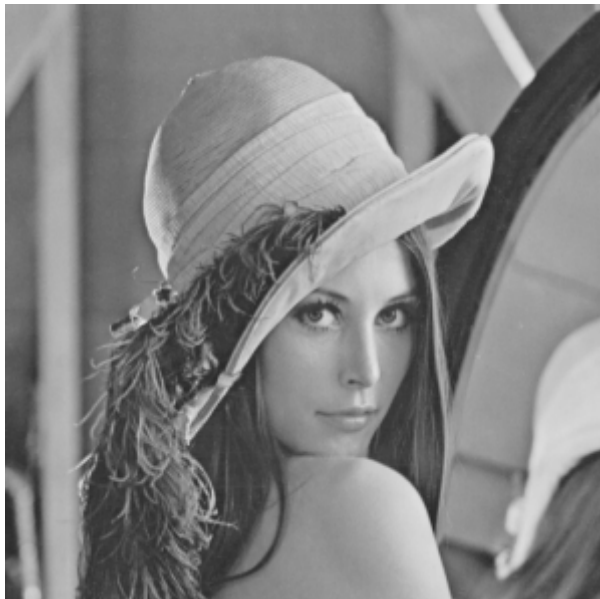
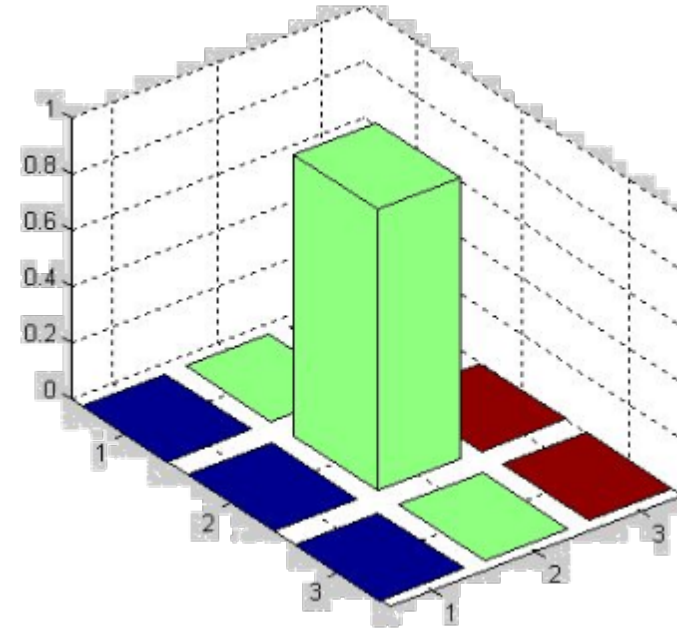


Let's go back to
2D convolution now

A well-known Test Image - Lena



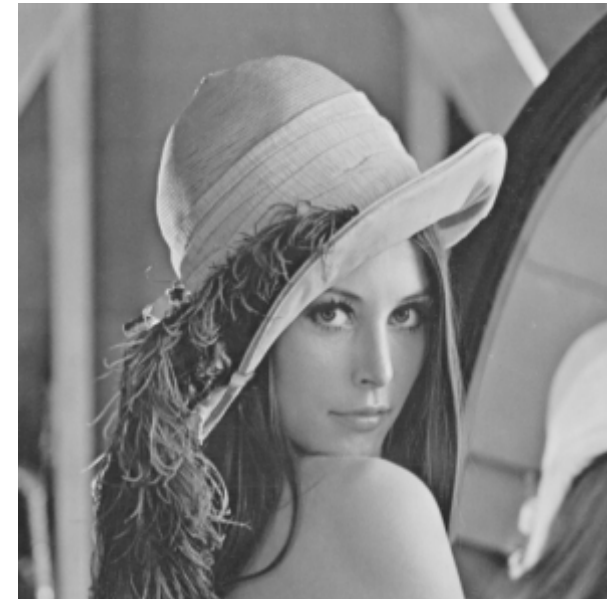
A Trivial example



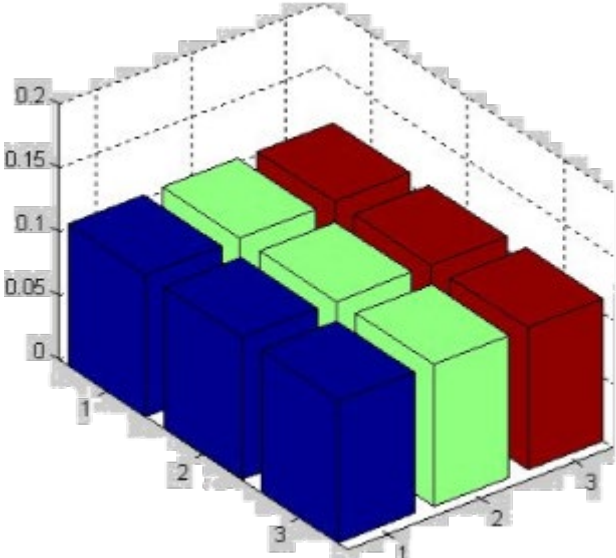
*

0	0	0
0	1	0
0	0	0

=



Linear Filtering



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

1	1	1
1	1	1
1	1	1

=

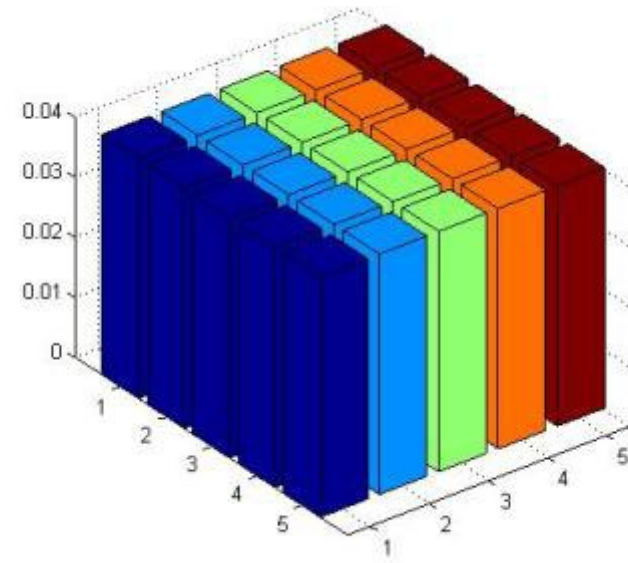
?

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?

...what about



$$\otimes \frac{2}{25} \cdot \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} =$$

.. convolution is linear



...what about

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



$$\otimes \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array} =$$

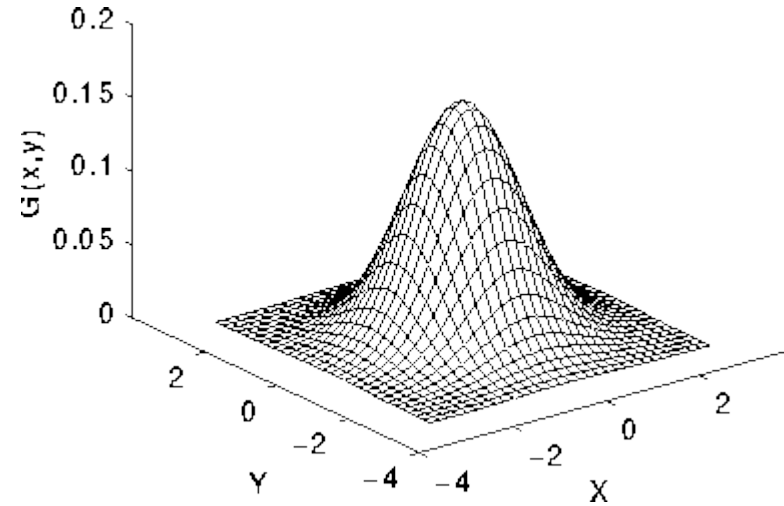
.. convolution is linear



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

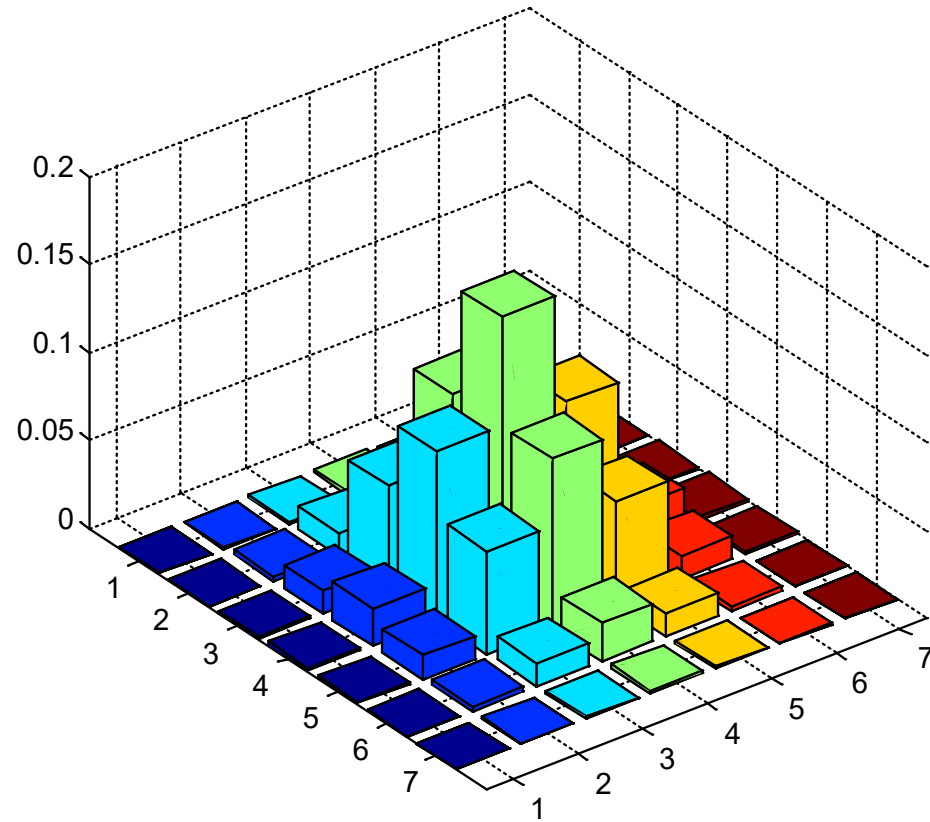
$$G(i, j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i^2 + j^2)}{2\sigma^2}\right)$$

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: Linearity

It is a **linear operator**

$$((\lambda I_1 + \mu I_2) \circledast \mathbf{w})(r, c) = \lambda(I_1 \circledast \mathbf{w})(r, c) + \mu(I_2 \circledast \mathbf{w})(r, c)$$

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

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

Properties of Convolution (and Padding)

It is **commutative** (in principle)

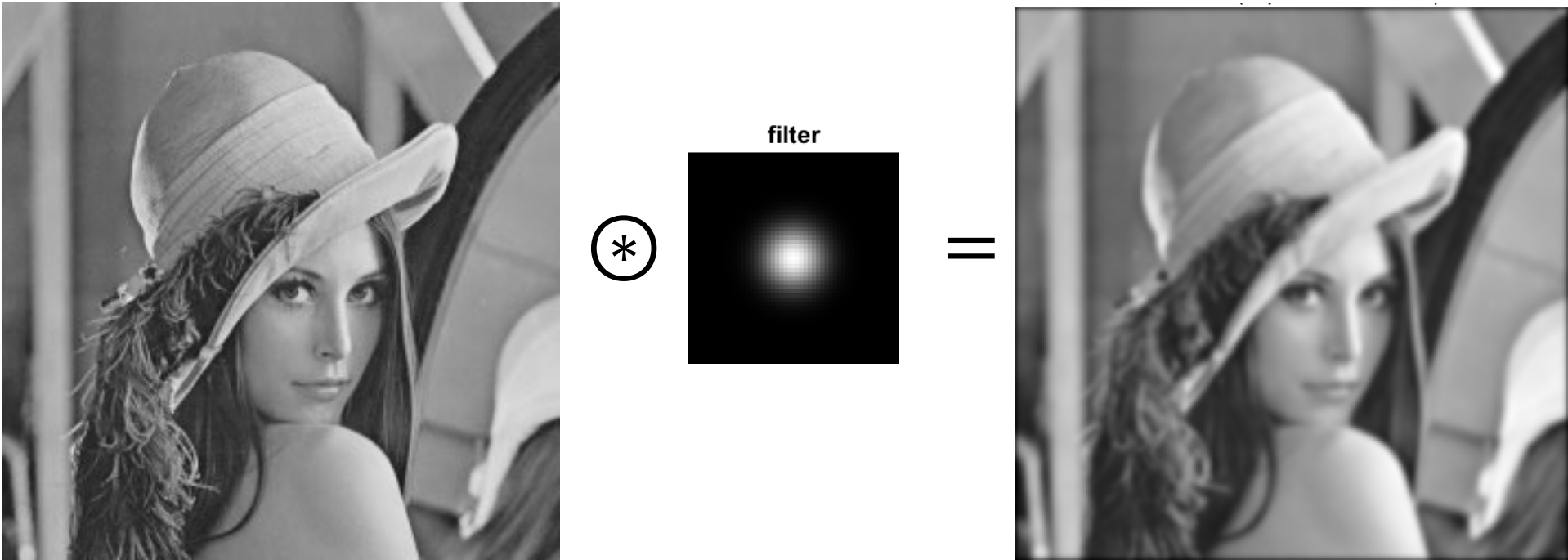
$$I_1 \circledast I_2 = I_2 \circledast I_1$$

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

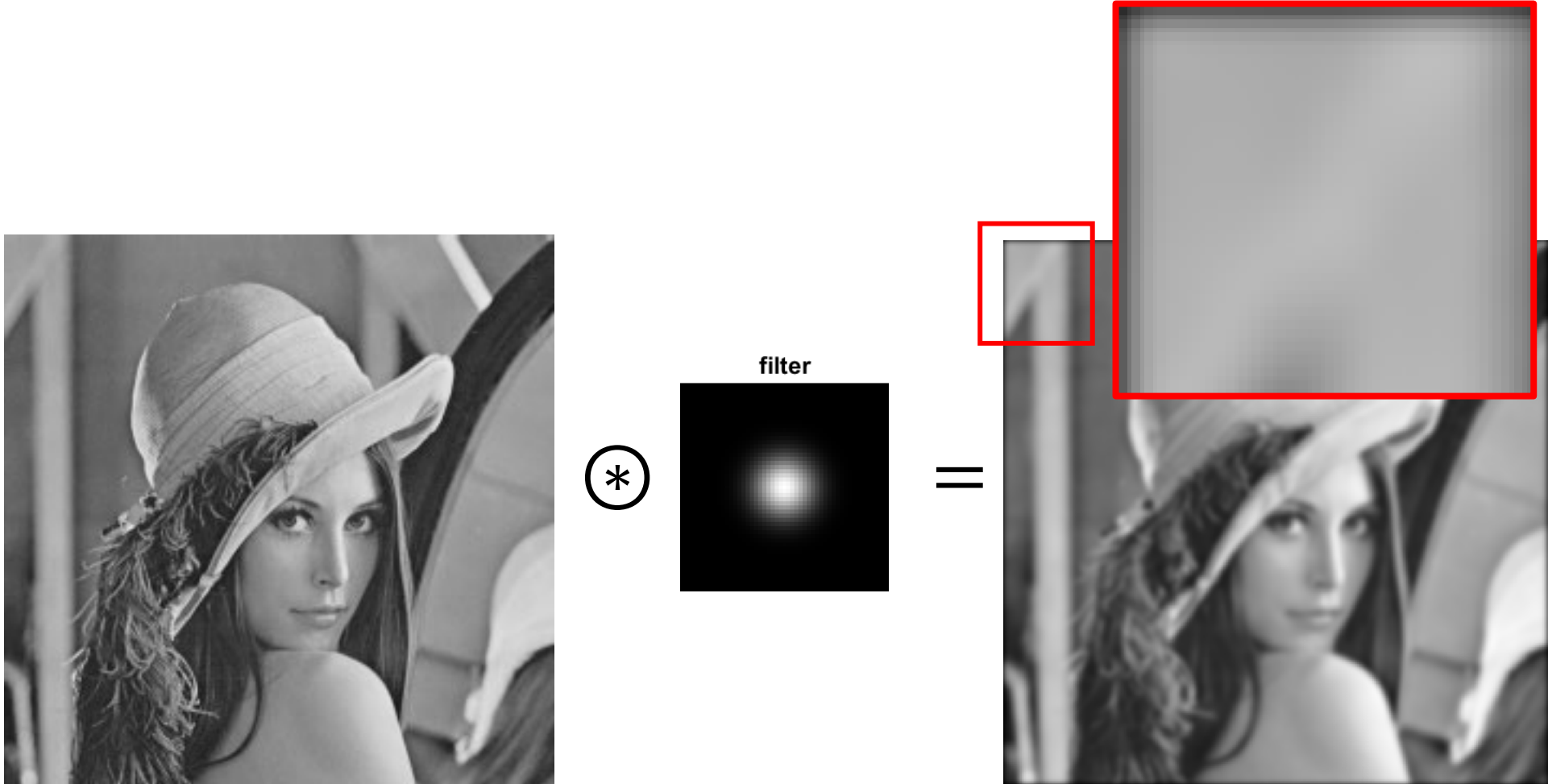
Input image I							filter w		
0	0	0	0	0	0	0	0	1	-1
0	1	0	2	1	0	0	-1	-1	0
0	1	1	1	0	1	0	1	-1	-1
0	1	2	1	0	1	0			
0	1	2	0	2	2	0			
0	1	0	1	0	0	0			
0	0	0	0	0	0	0			

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

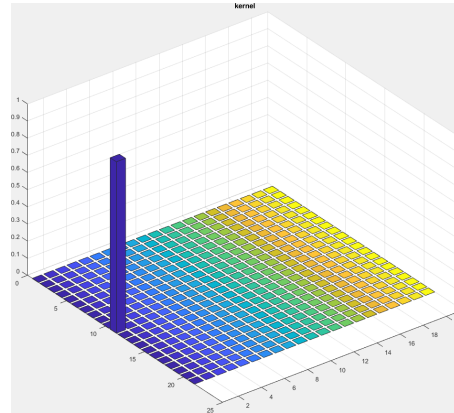
Is Convolution Commutative?



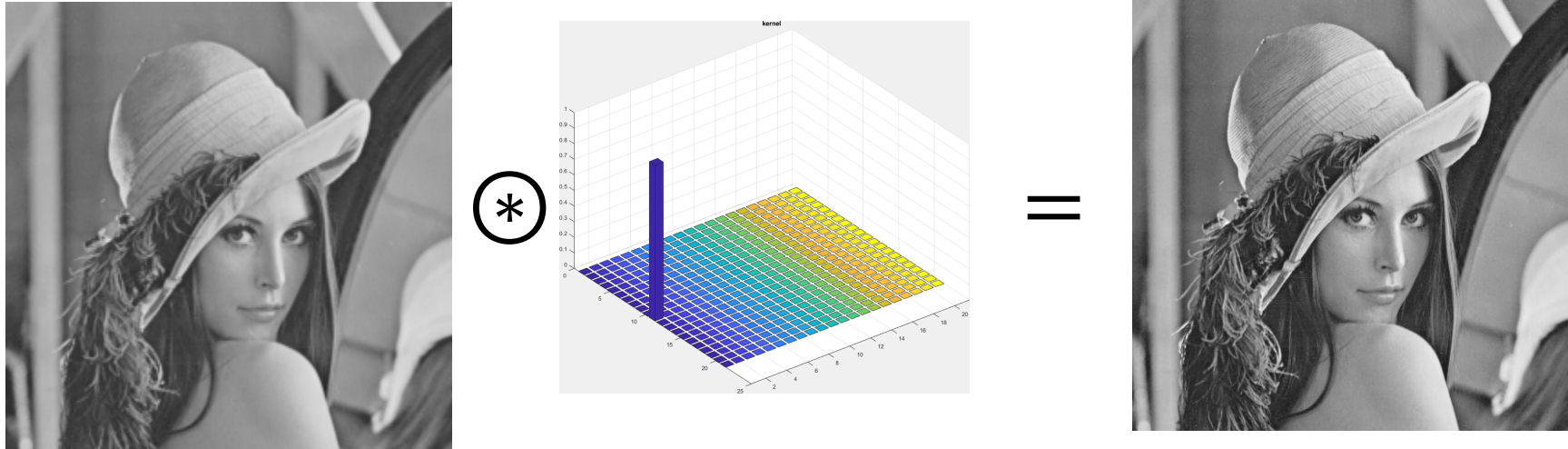
Is Convolution Commutative?



Translation

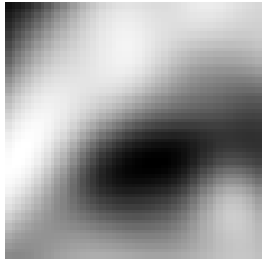
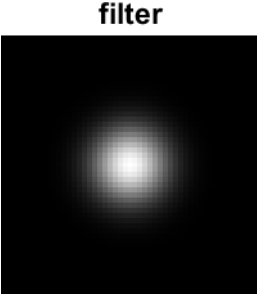


Translation



Remember the filter has to be flipped before convolution

Is Convolution Commutative?



This holds for the «full convolution» modality, not the «same» or «valid»

Properties of Convolution: Associative

It is also **associative**

$$f \circledast (g \circledast \mathbf{w}) = (f \circledast g) \circledast \mathbf{w} = f \circledast g \circledast \mathbf{w}$$

and **dissociative**

$$f \circledast (g + \mathbf{w}) = f \circledast g + f \circledast \mathbf{w}$$

Properties of Convolution: Shift invariance

It is also **associative**

$$f \circledast (g \circledast \mathbf{w}) = (f \circledast g) \circledast \mathbf{w} = f \circledast g \circledast \mathbf{w}$$

and **dissociative**

$$f \circledast (g + \mathbf{w}) = f \circledast g + f \circledast \mathbf{w}$$

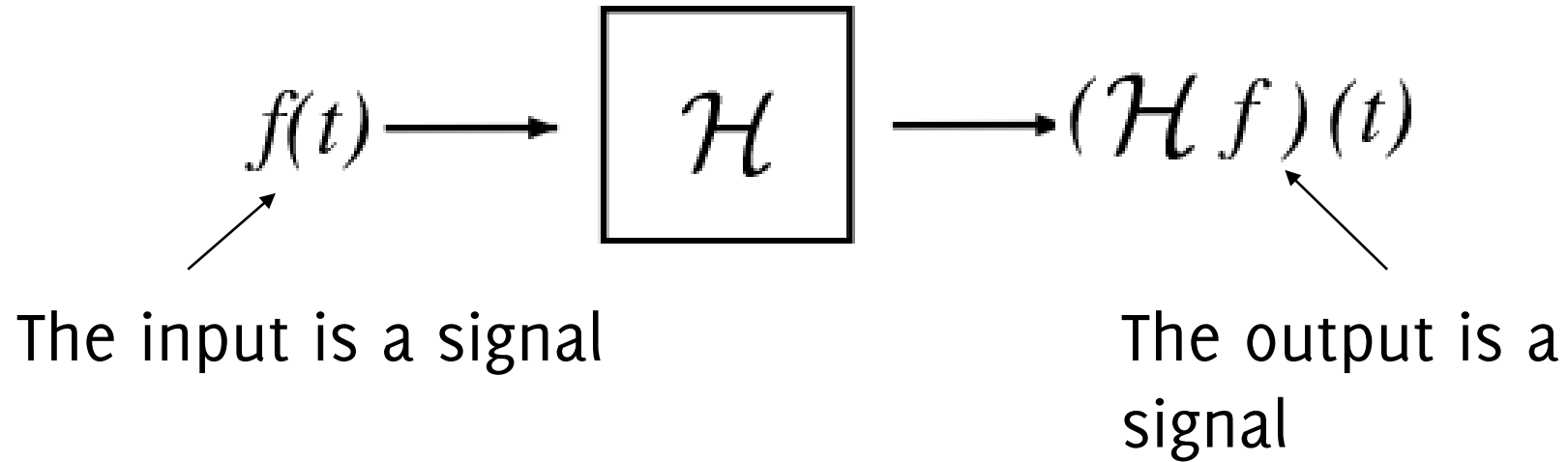
It is **shift-invariant**, namely

$$(I(\cdot - r_0, \cdot - c_0) \circledast \mathbf{w})(r, c) = (I \circledast \mathbf{w})(r - r_0, c - c_0)$$

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

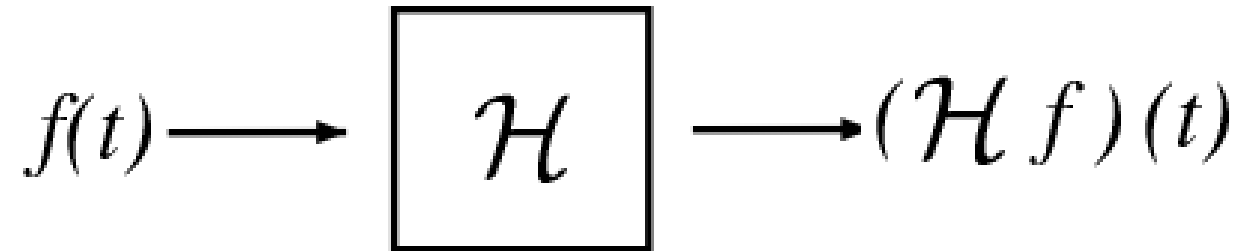
Systems

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

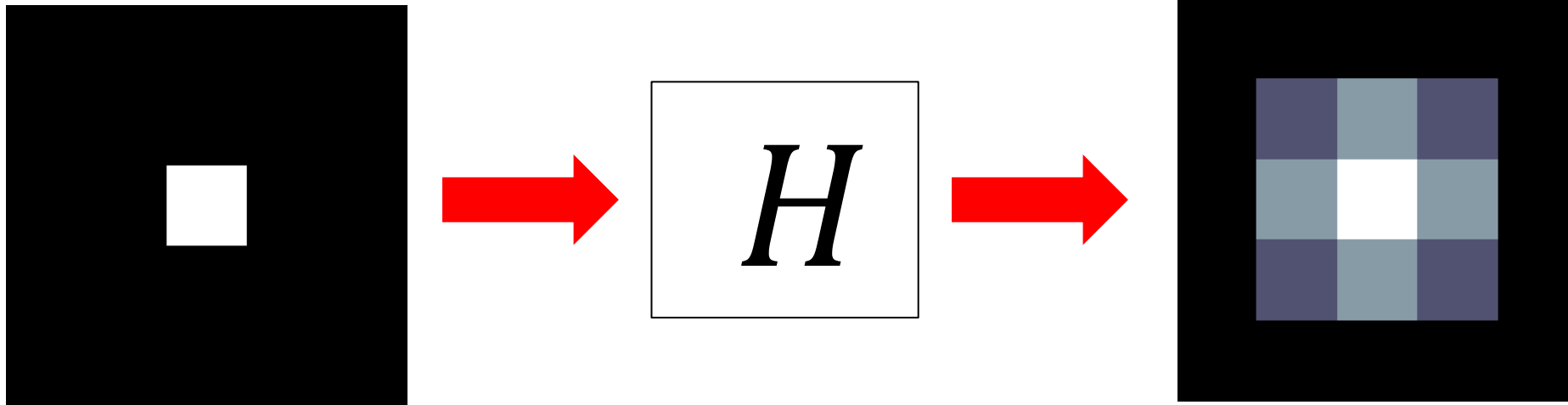
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

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

Linear and Time Invariant Systems

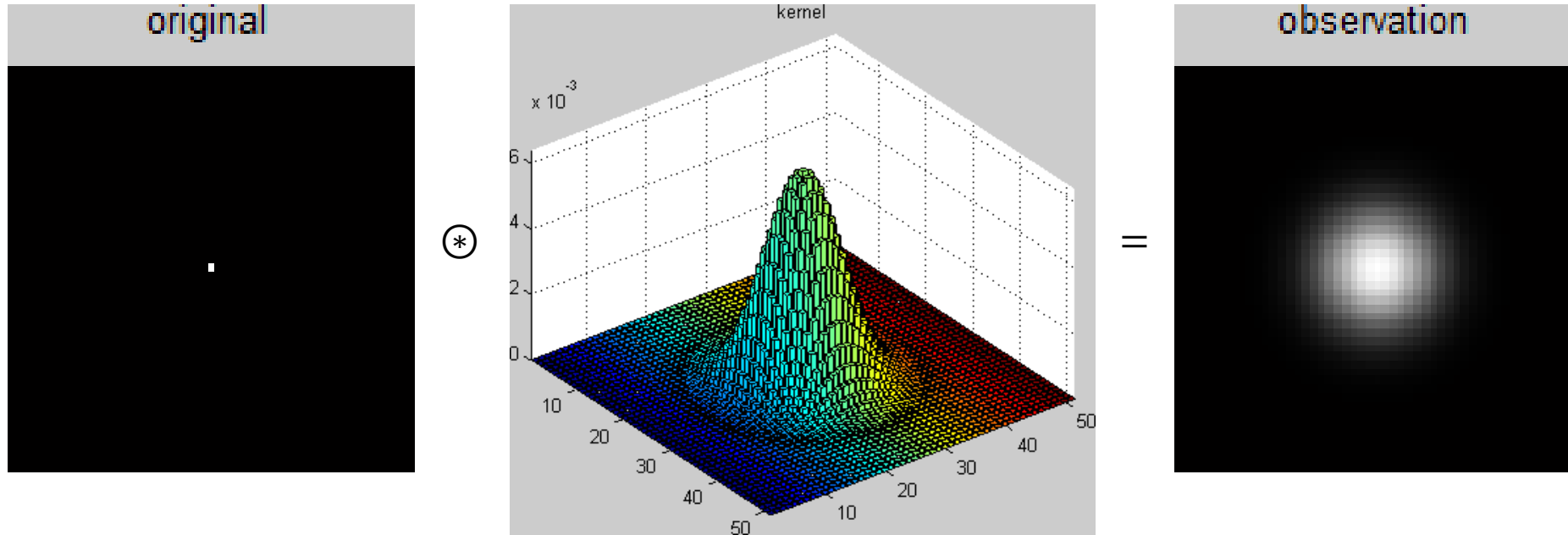
All the systems that are Linear and Time Invariant (LTI) have an equivalent **convolutional operator**

- 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

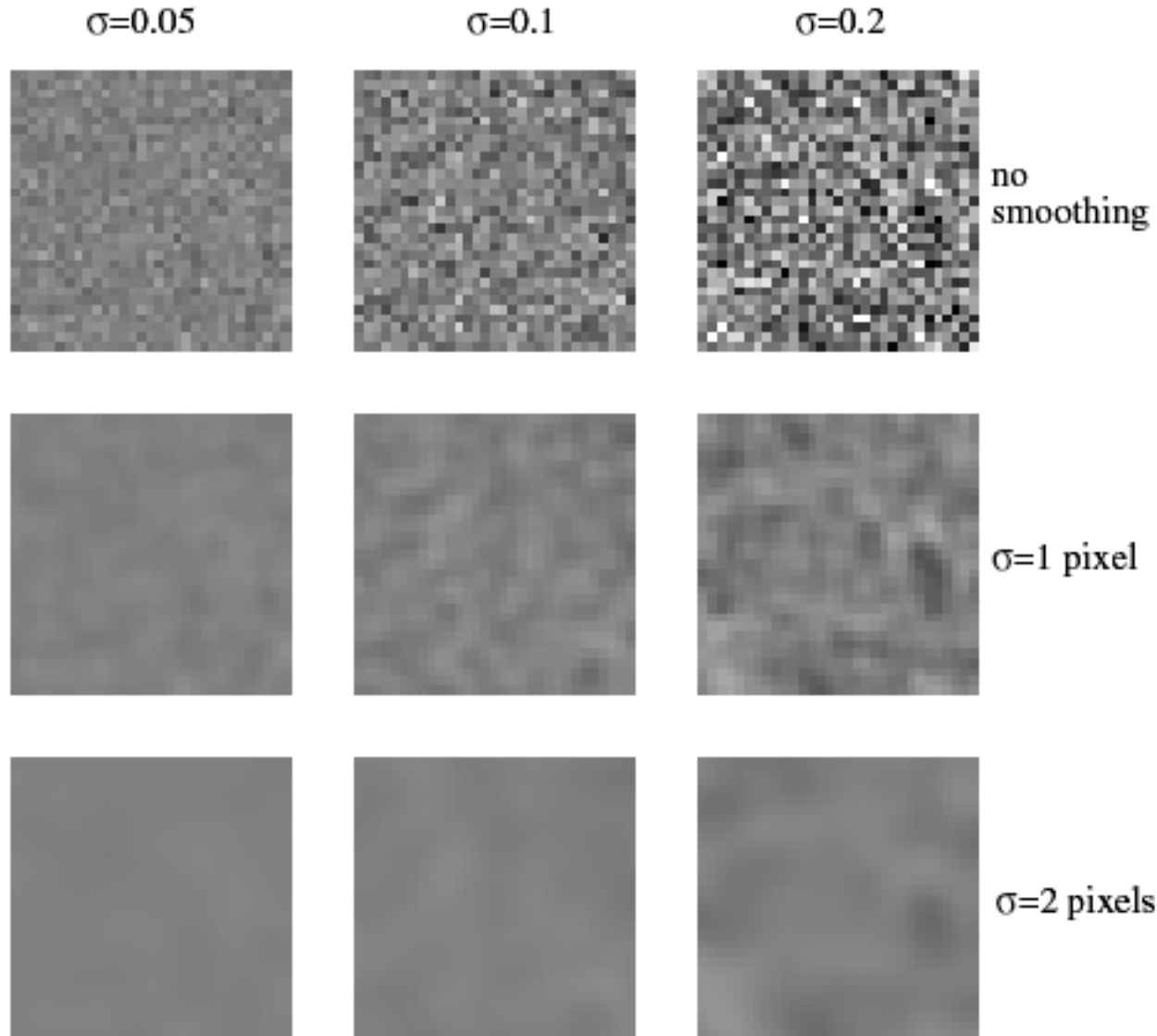


This is why h is also called the “Point Spread Function”

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.

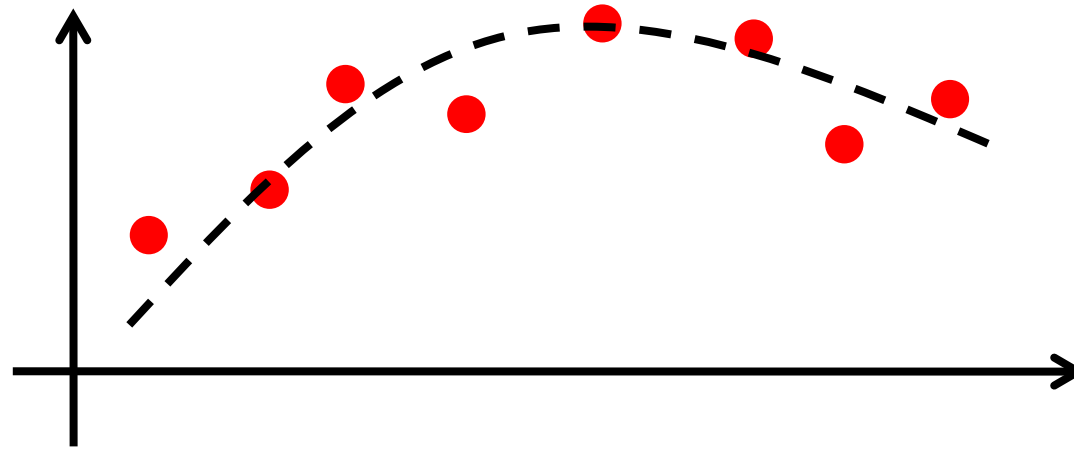
The noise standard deviation σ is also assumed as known.

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)

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

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

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

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

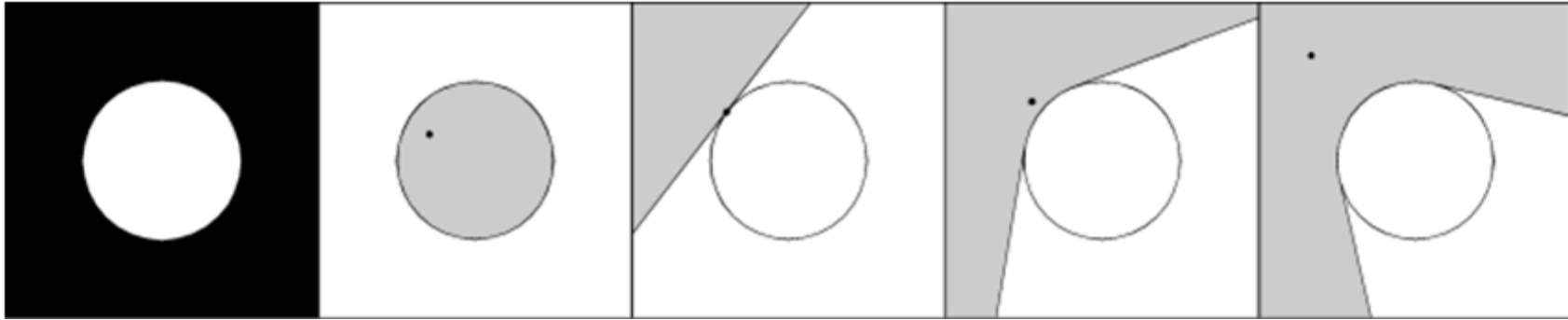
Denoising Approaches

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.

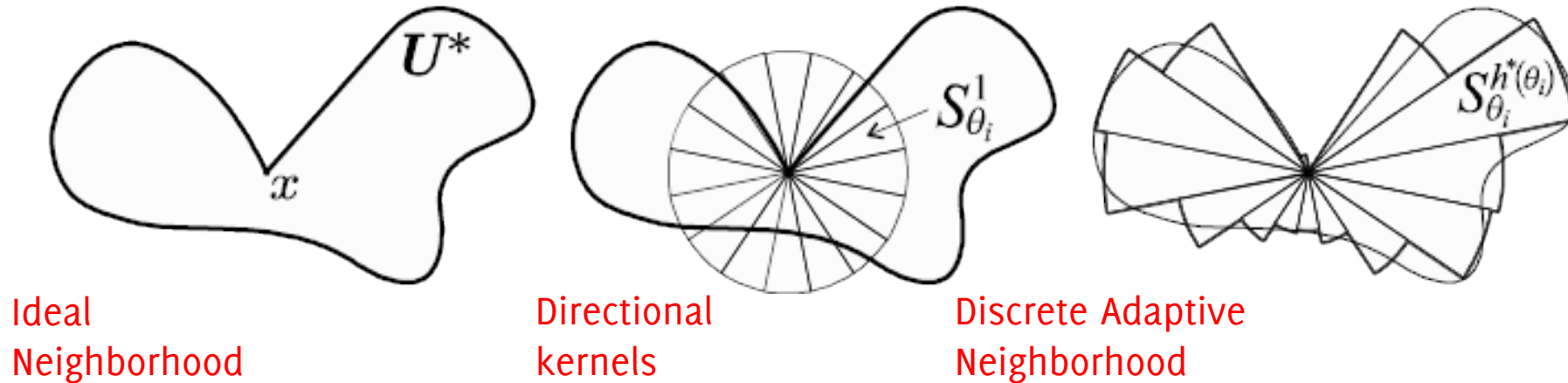
For practical reasons, the ideal neighborhood is assumed starshaped

Further details at LASIP c/o Tampere University of Technology

<http://www.cs.tut.fi/~lasip/>

Neighborhood discretization

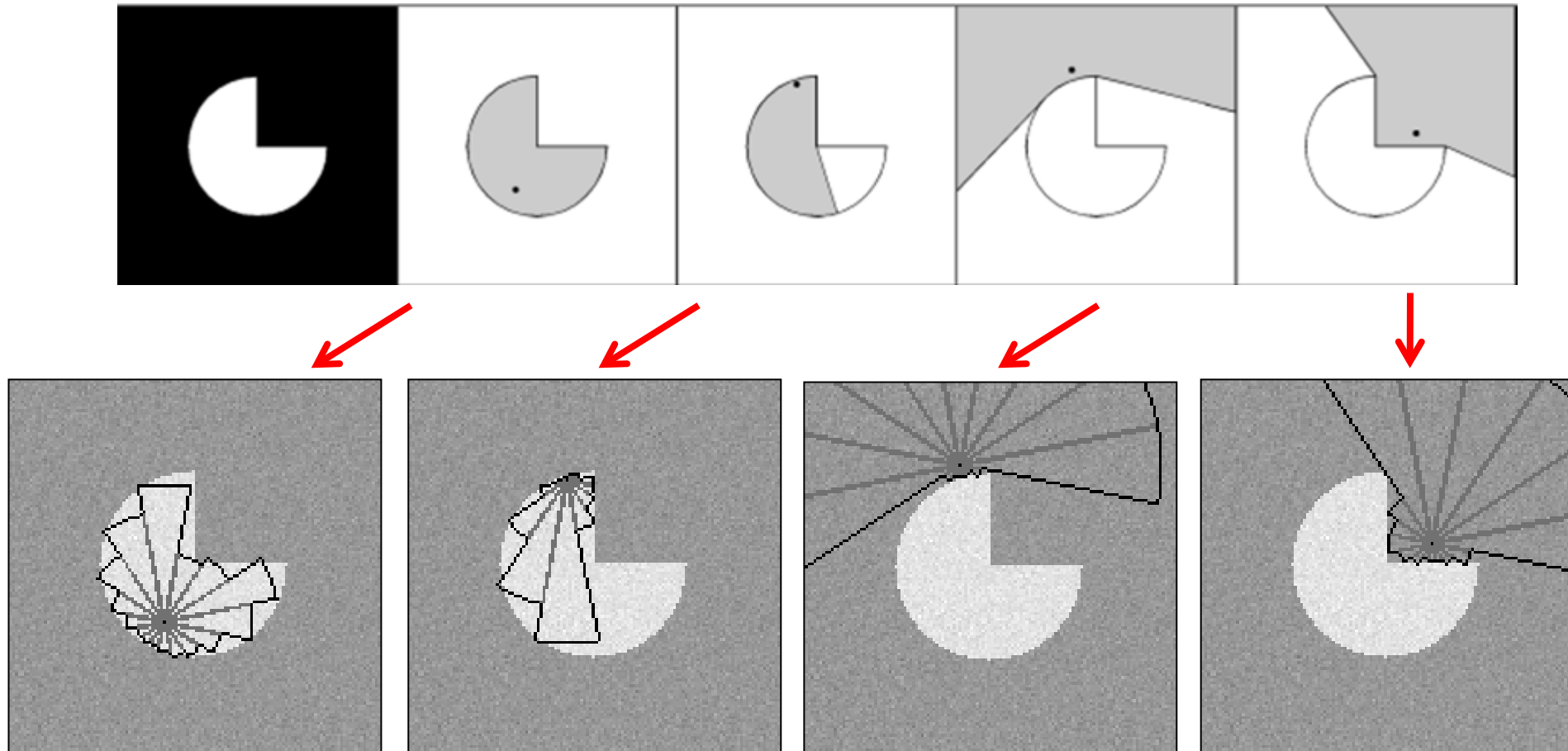
A suitable discretization of this neighborhood is obtained by using a set of directional LPA kernels $\{g_{\theta,h}\}_{\theta,h}$



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

Ideal neighborhood – an illustrative example

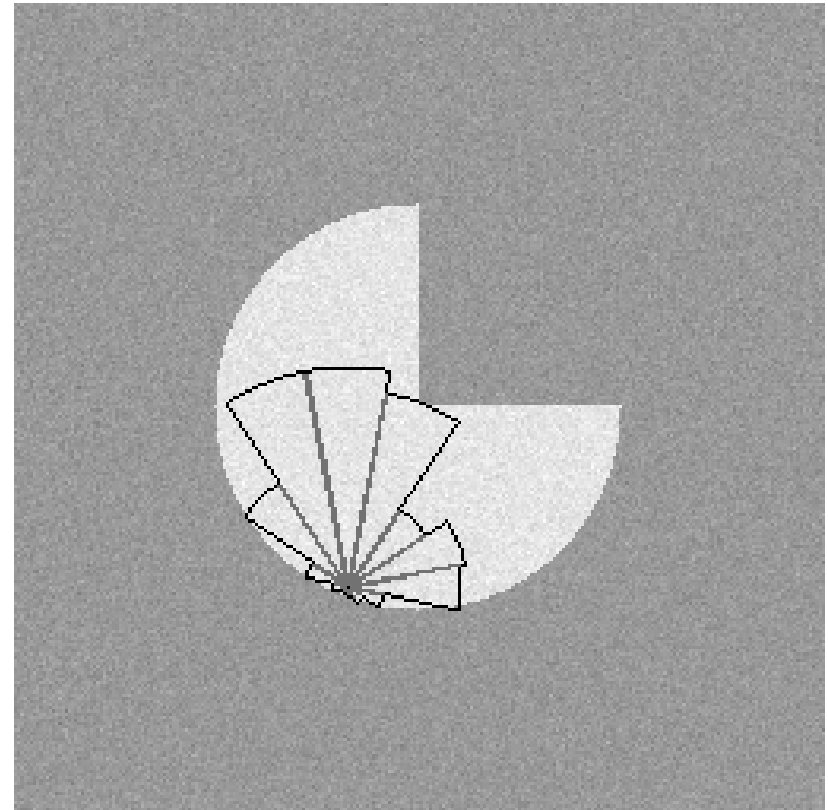
Ideal in the sense that the neighborhood defines the support of pointwise Least Square Estimator of the reference point.



Examples of Adaptively Selected Neighborhoods

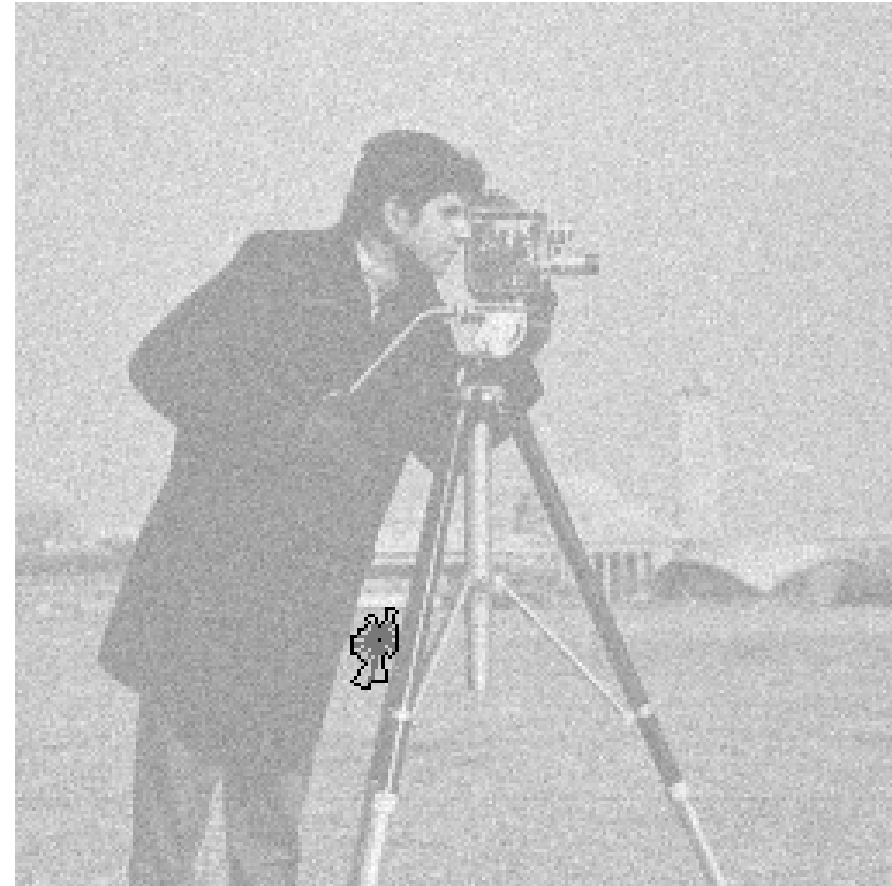
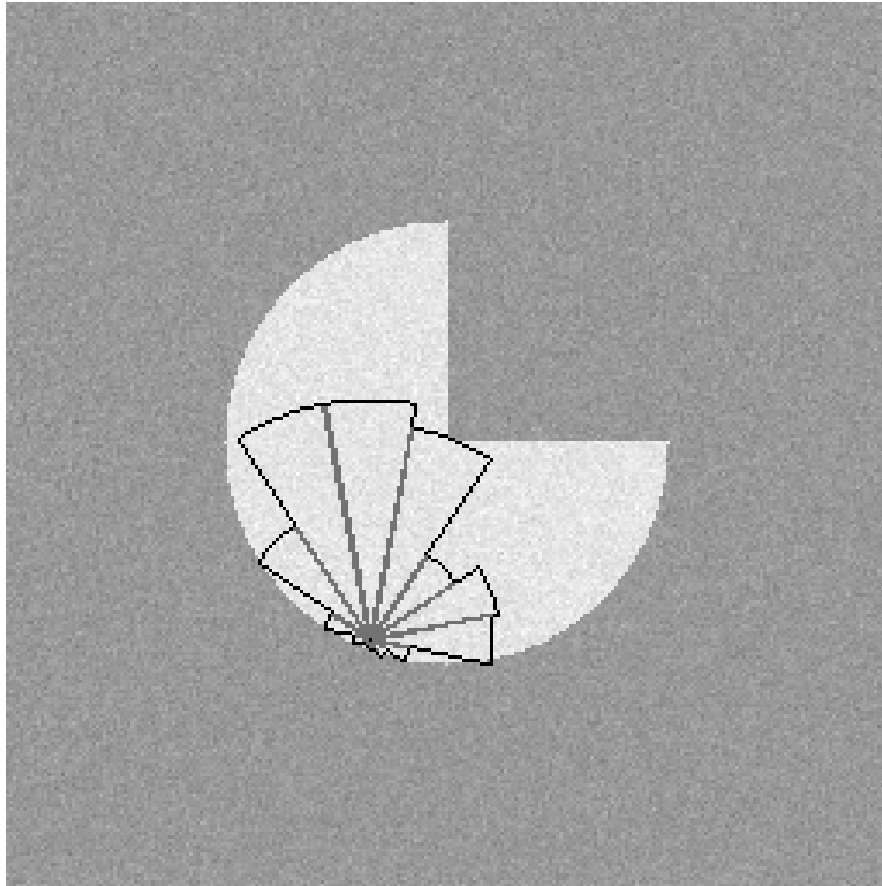
Define, $\forall x \in X$, the “ideal” neighborhood \tilde{U}_x

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)

