Basics of Image Handling

Image Classification: Modern Approaches

Giacomo Boracchi, Alessandro Giusti
DEIB, Politecnico di Milano

February, 12th, 2018

giacomo.boracchi@polimi.it
home.deib.polimi.it/boracchi/

Basics of Image Handling

Most important image processing operations for classification problems

Spatial-Domain vs Transform-Domain Methods

A survey of most important operations in image processing:

- Spatial Intensity Transformations
- Spatial Local Transformations (filtering)
- Global Transformations

Spatial transformations (intensity or local) are direct manipulation of pixel intensities

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

Intensity Transformation

Transformations that operate
on each single pixels of an image

Intensity Transformations

In general, these can be written as

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

Where

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

T operates independently on each single pixel.

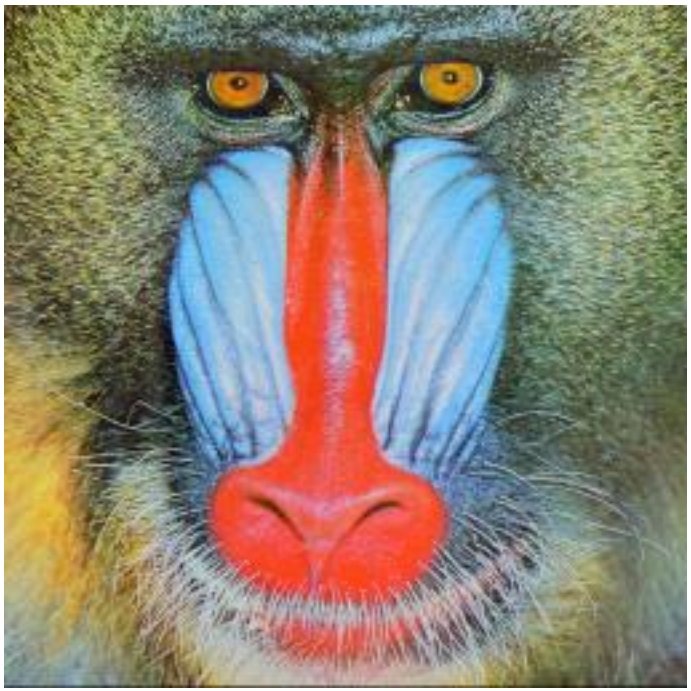
RGB → Grayscale Conversion

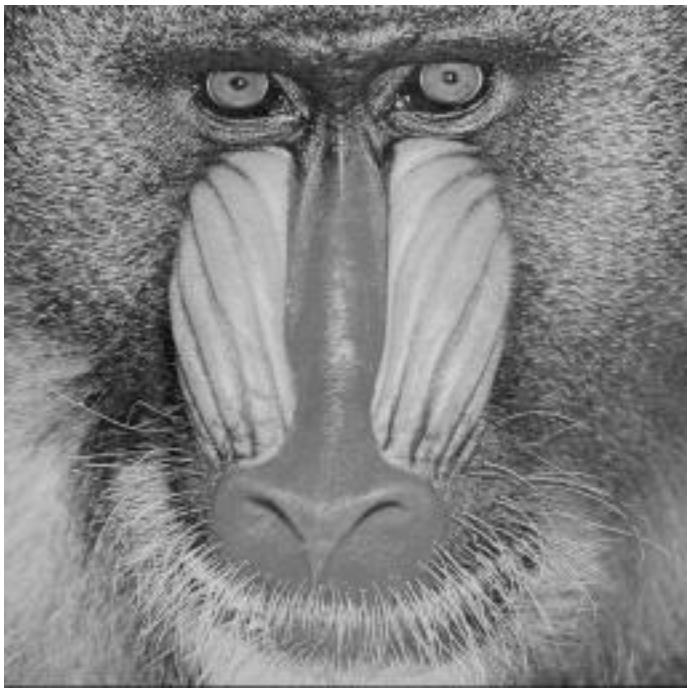
A linear transformation of pixel intensities T: $\mathbb{R}^3 \rightarrow \mathbb{R}$

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

which corresponds to a linear combination of the 3 channels

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





YCbCr color space

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

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

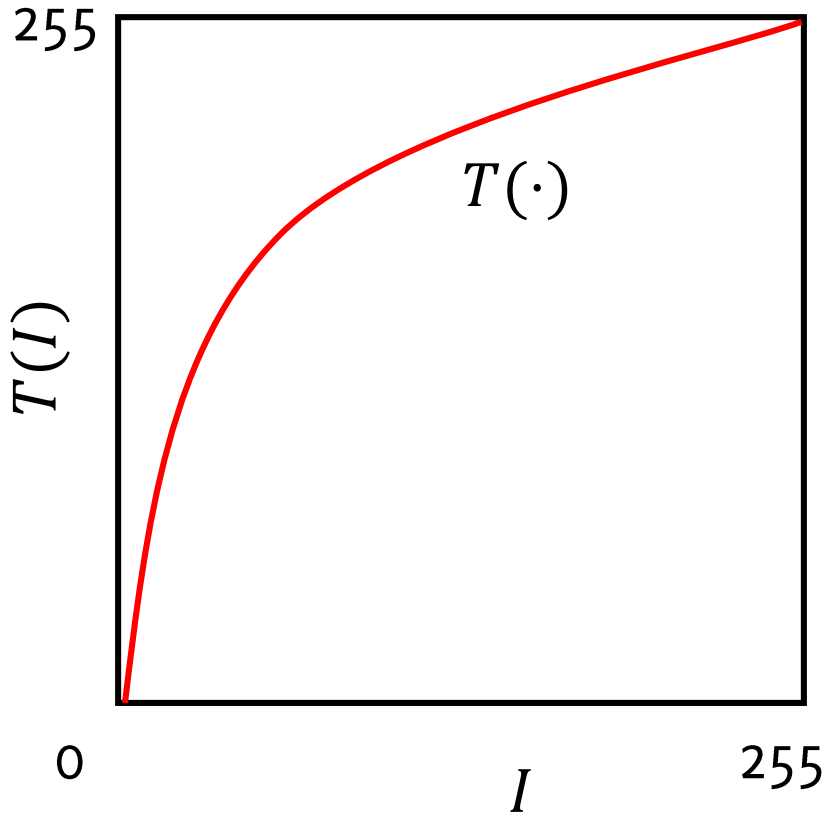
Human eye is less sensitive to color changes than luminance variations. Thus,

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

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

Gray-level mapping

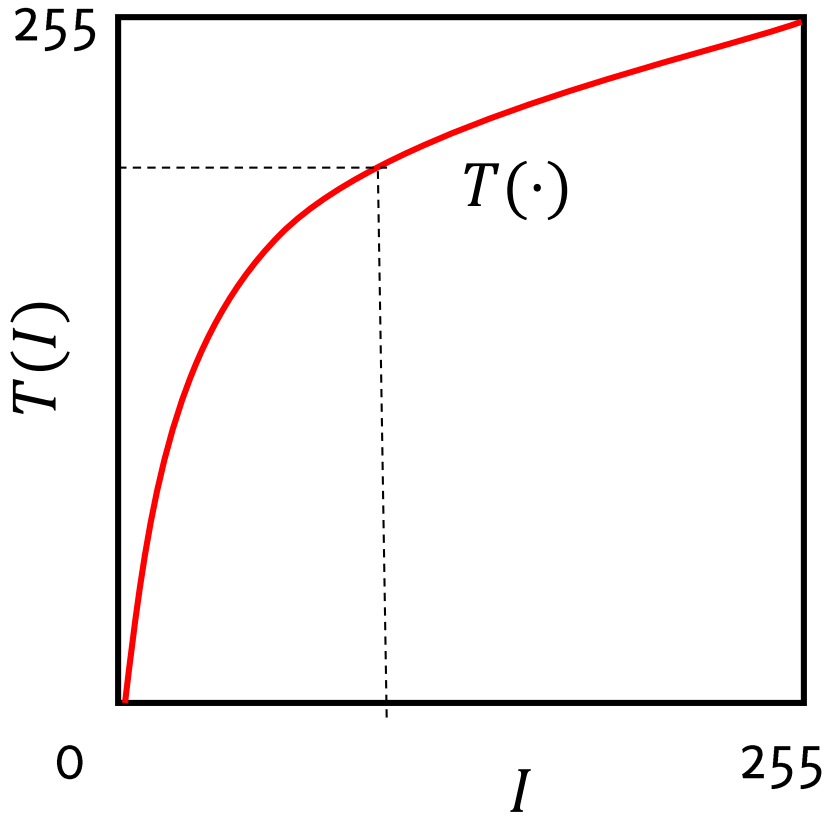
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

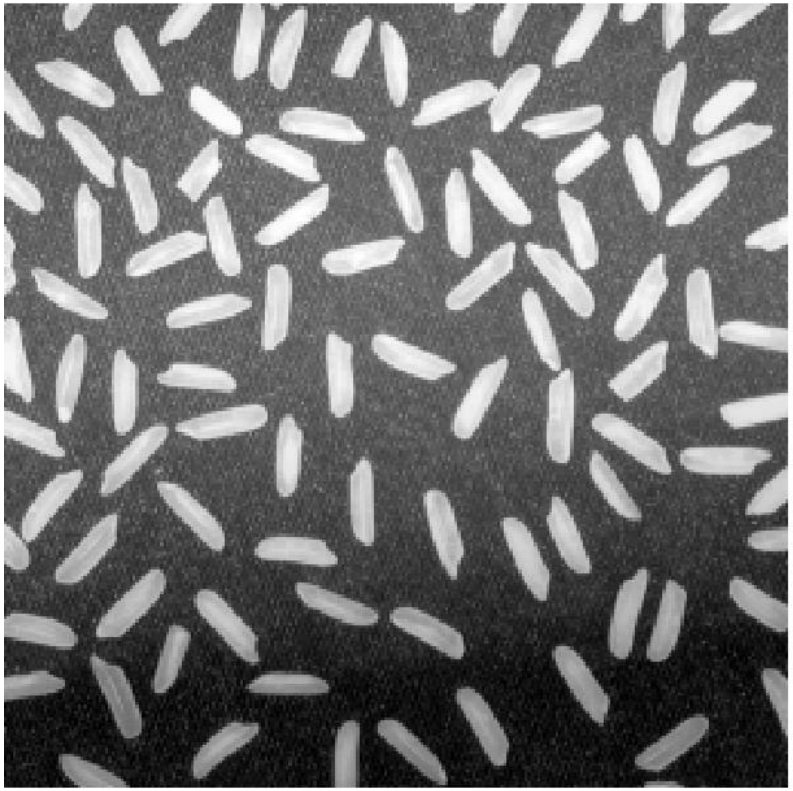


Gray-level mapping

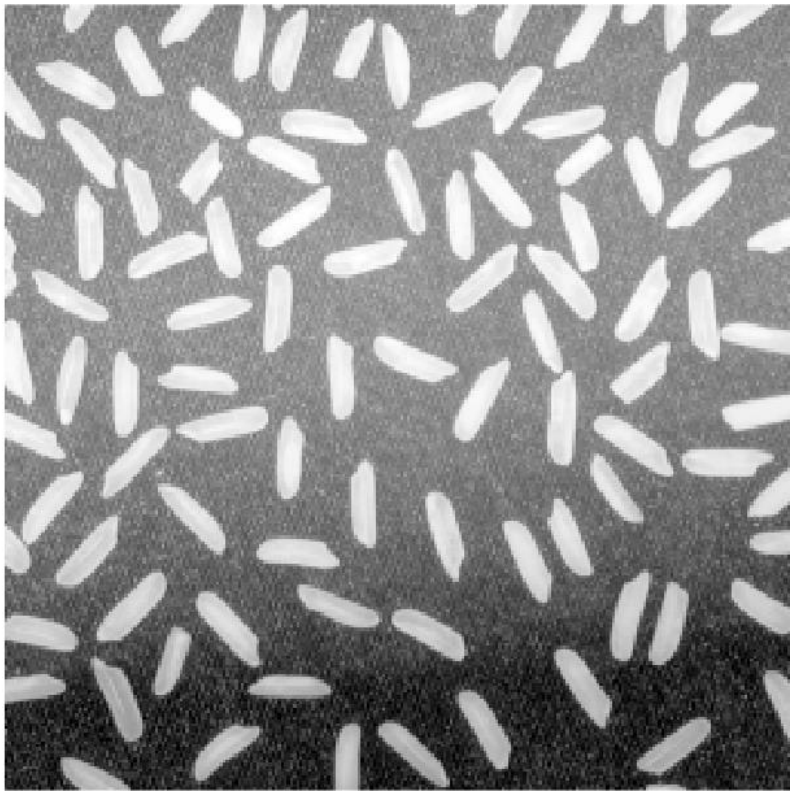
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?





Input I

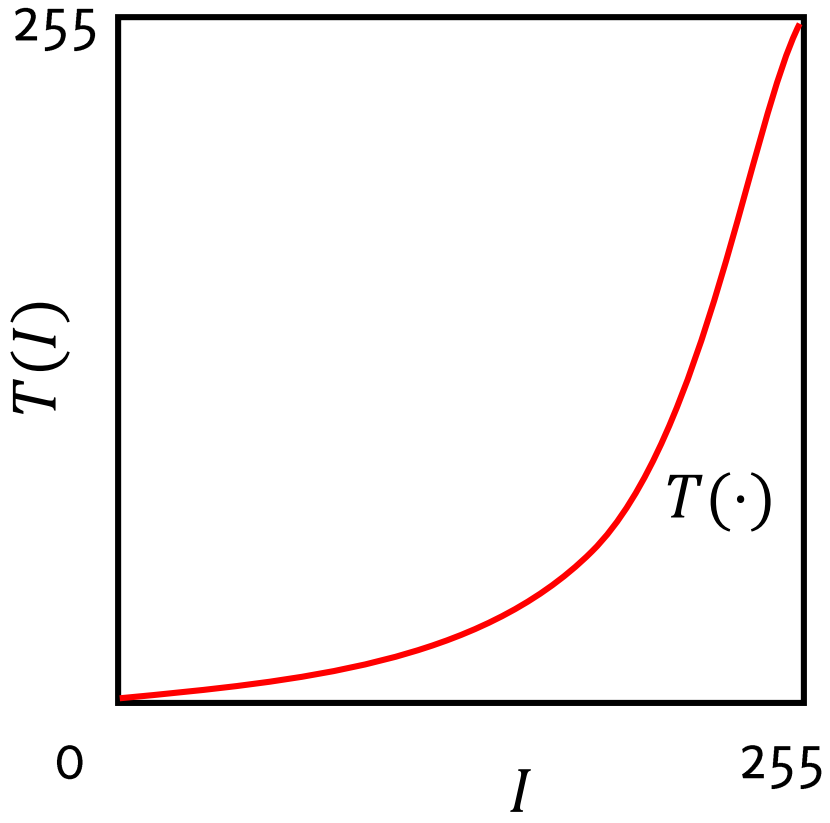


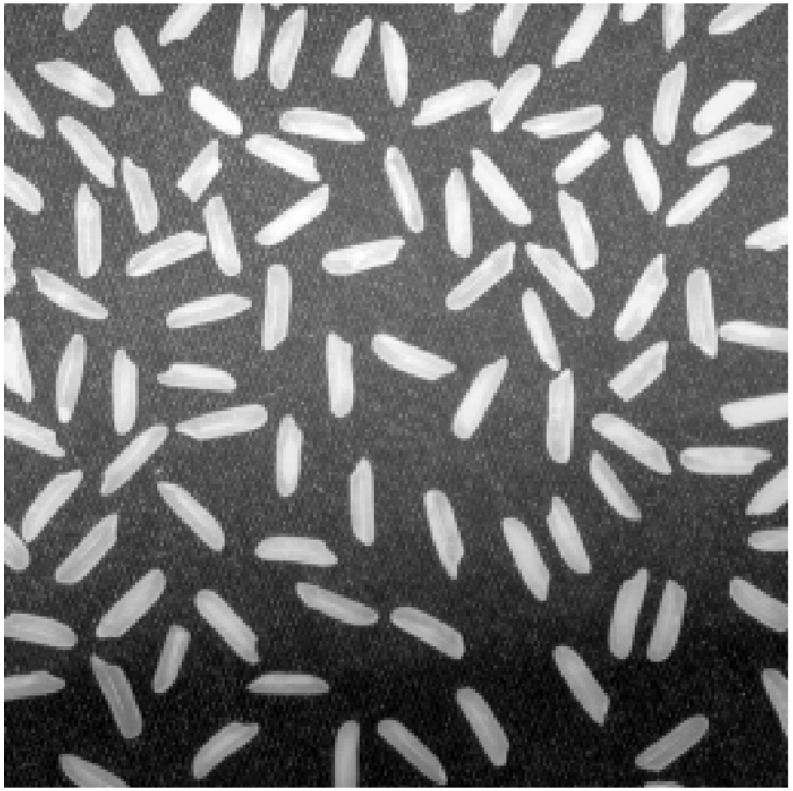
Output $G = T(I)$

Gray-level mapping

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

What does this T do?





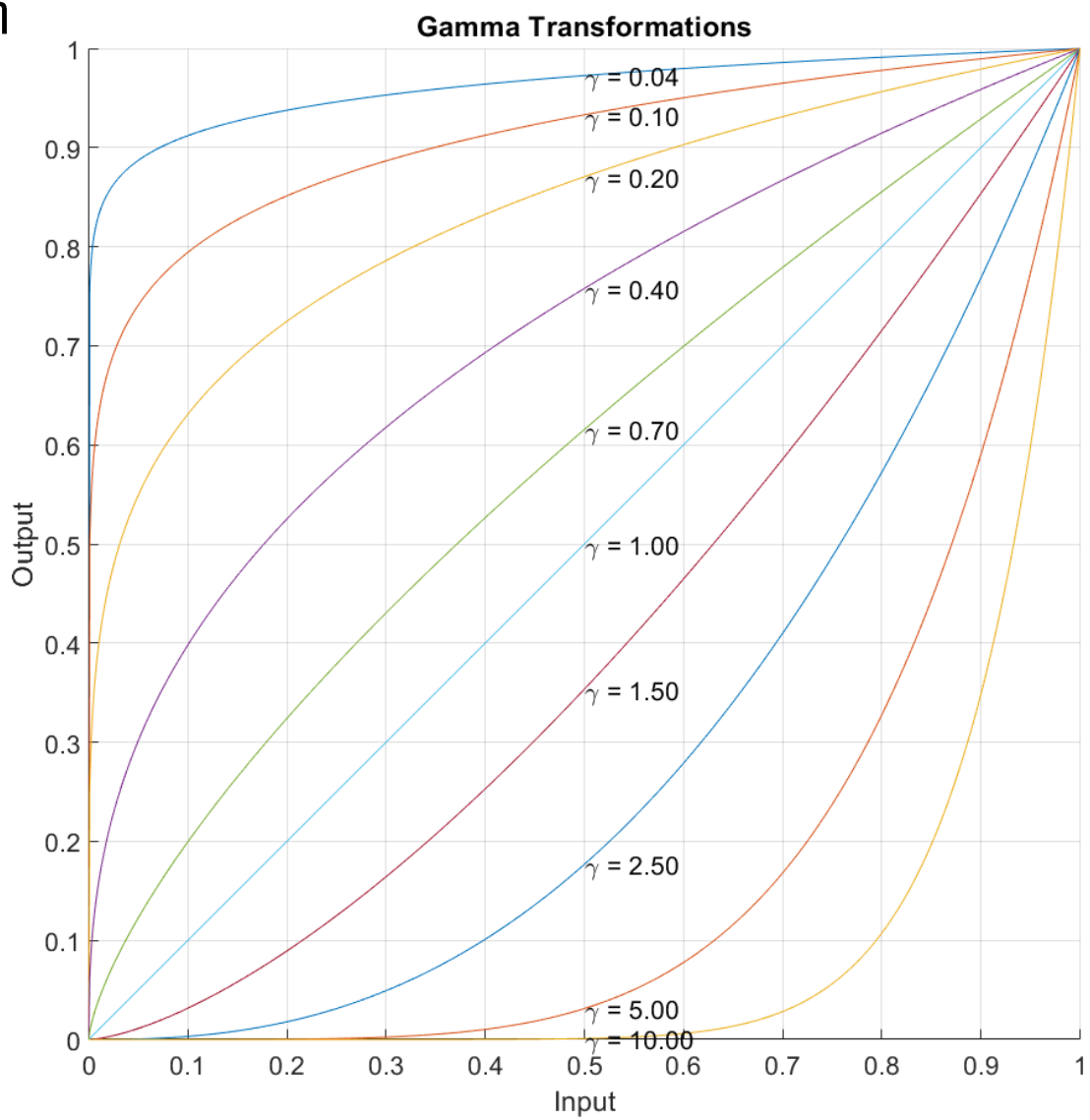
Input I

Output $G = T(I)$

Gamma Correction

Power-low transformation that can be written as

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



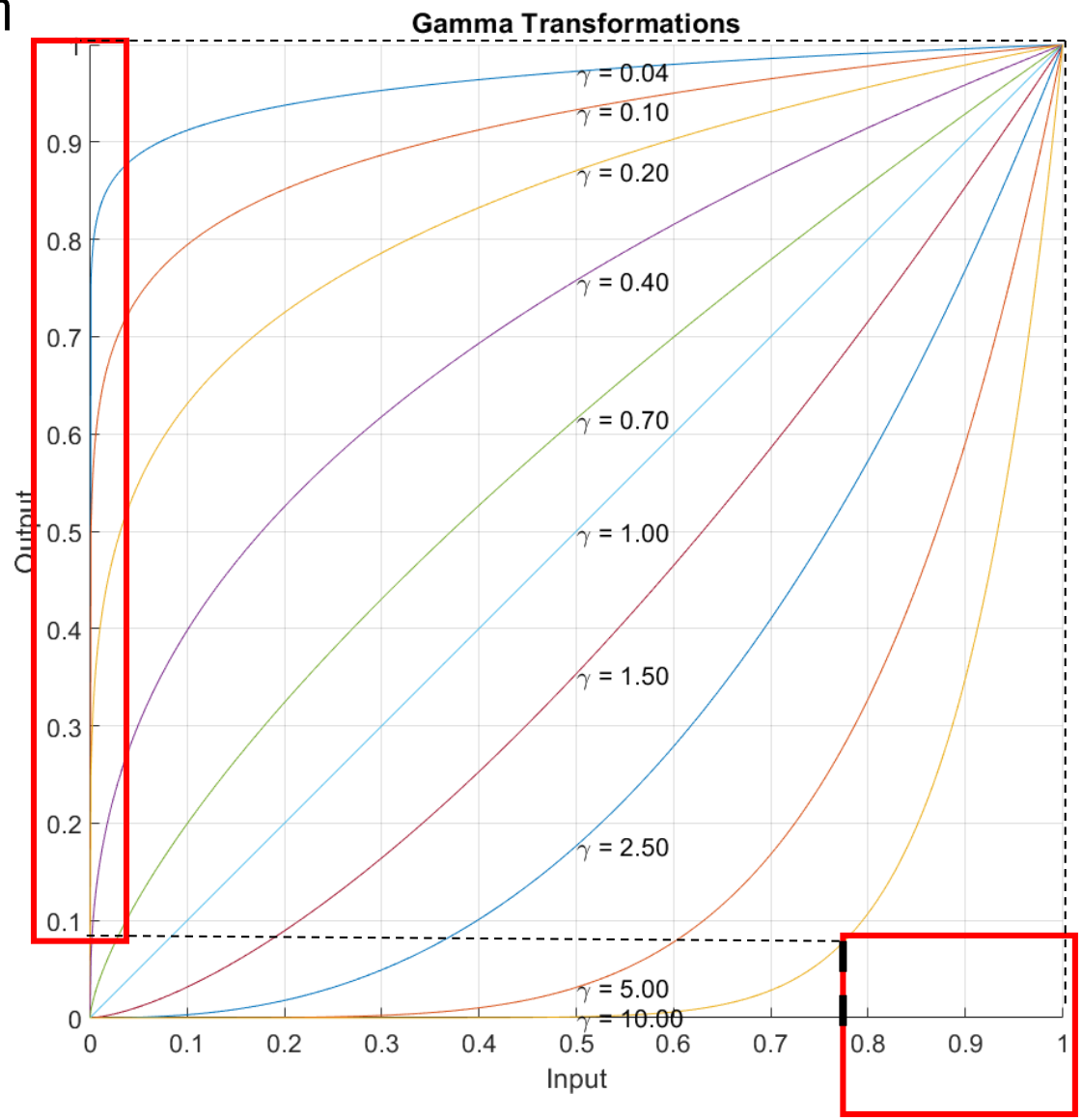
Gamma Correction

Power-low transformation that can be written as

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

Contrast Enhancement:

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



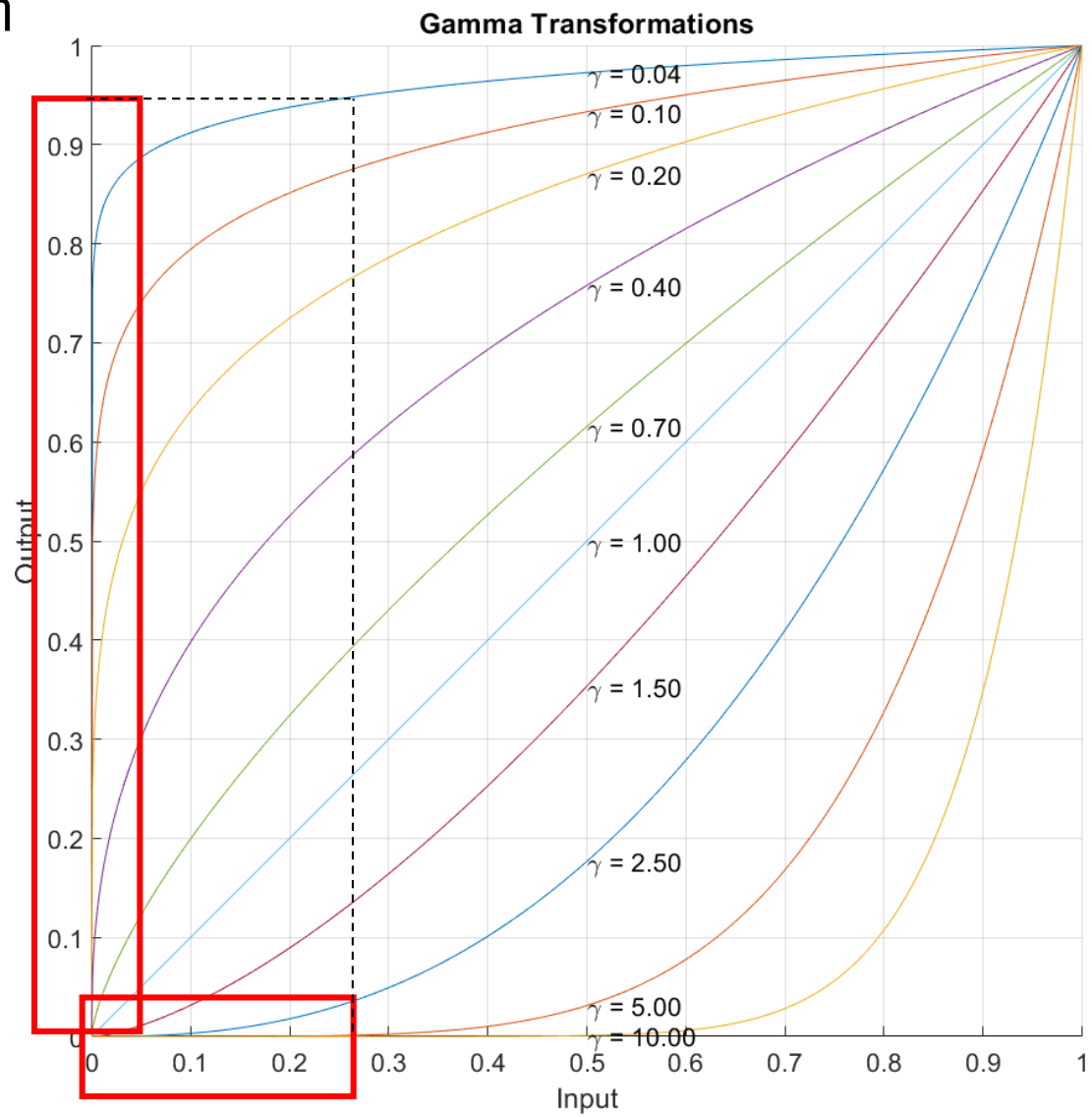
Gamma Correction

Power-low transformation that can be written as

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

Contrast Enhancement:

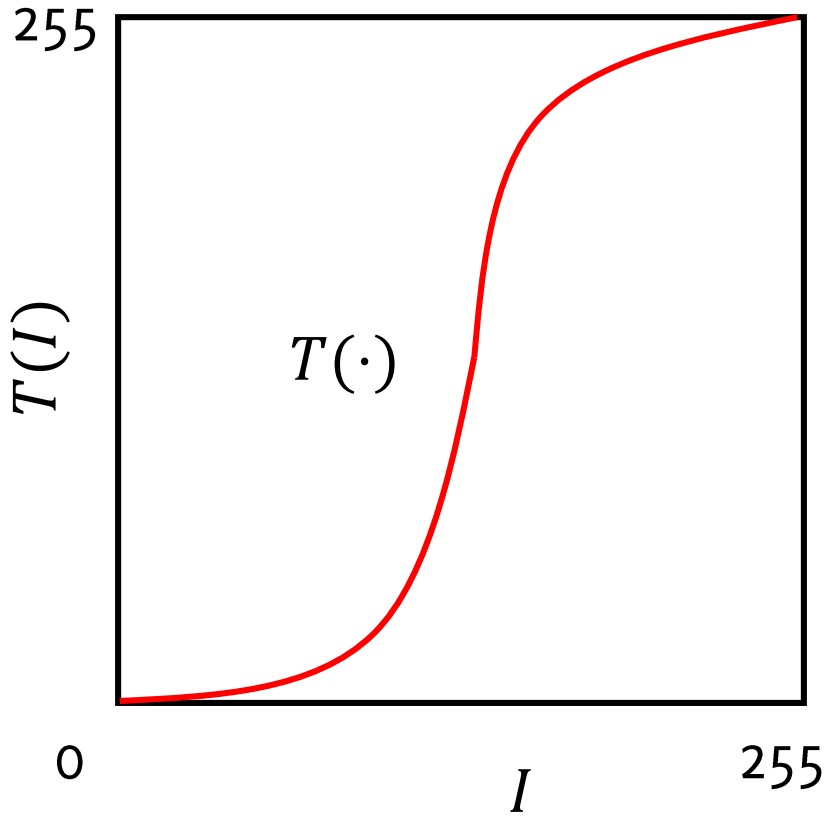
- Low values of γ stretch the intensity range at high-values
- High values of γ stretch the intensity range at low values



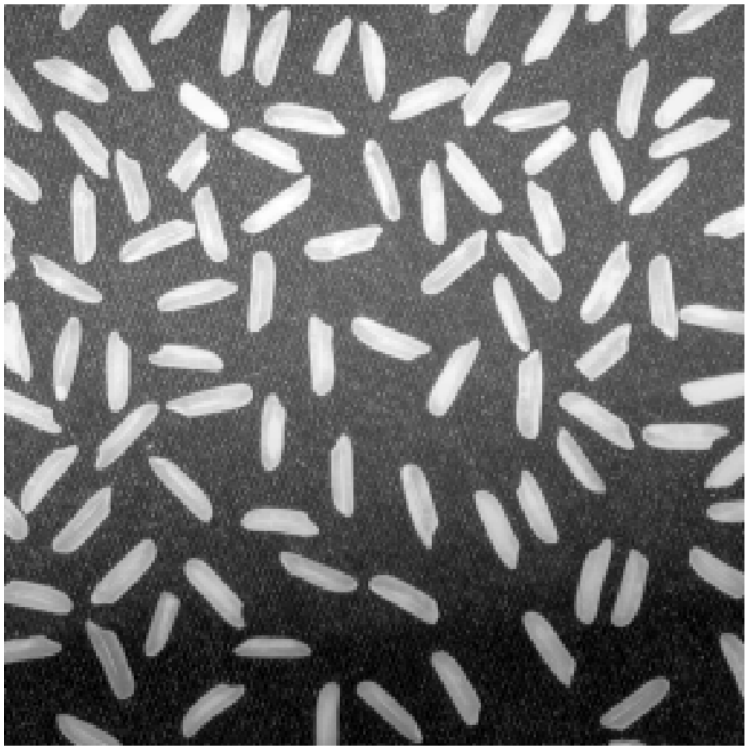
Gray-level mapping

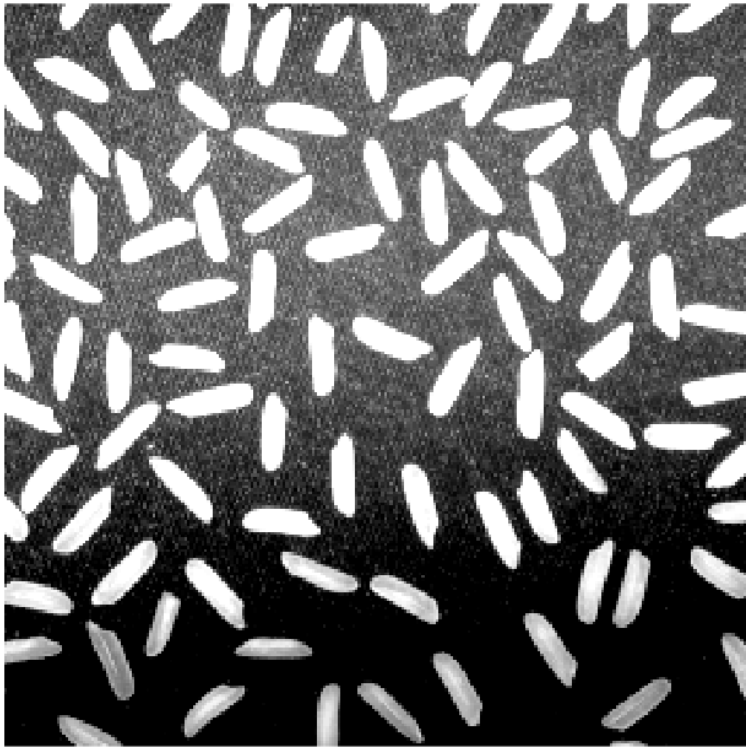
A transformation $T: \mathbb{R} \rightarrow \mathbb{R}$ that operate on gray-scale images or on each color-plane separately

What does this T do?



Contrast Stretching





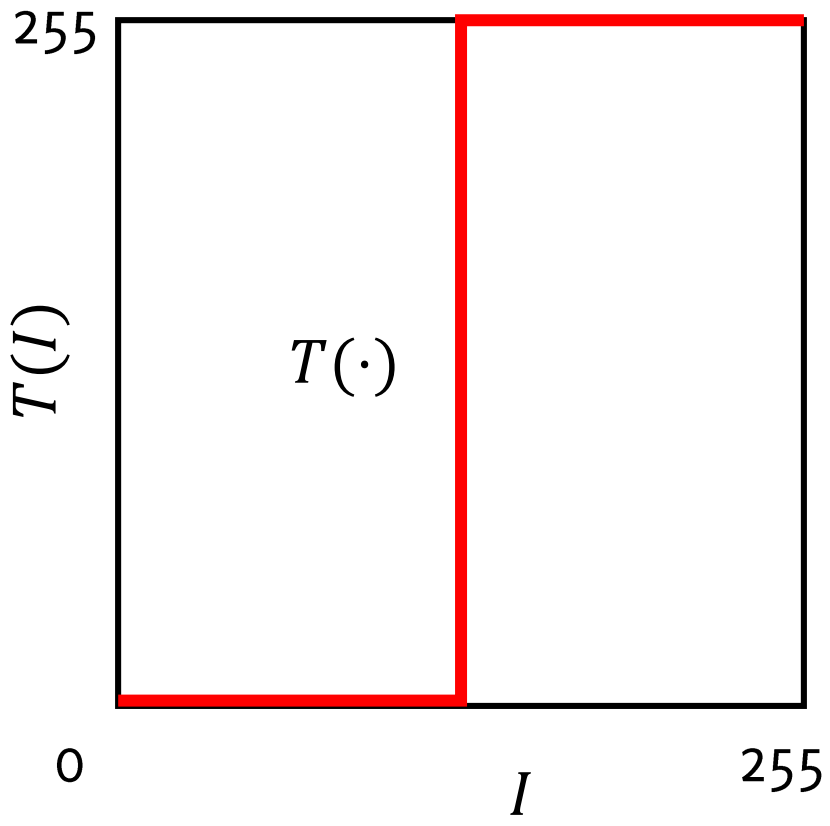
Contrast stretching: increases the constant at values in the middle of intensity range, decreases contrast at bright and dark regions.

It is implemented by piecewise or parametric transformations

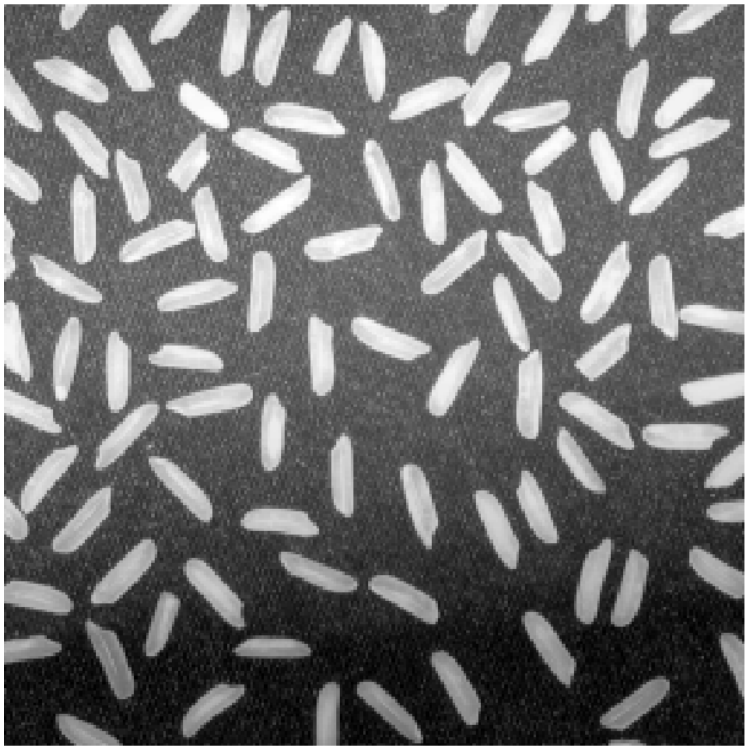
Gray-level mapping

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

What does this T do?



Thresholding

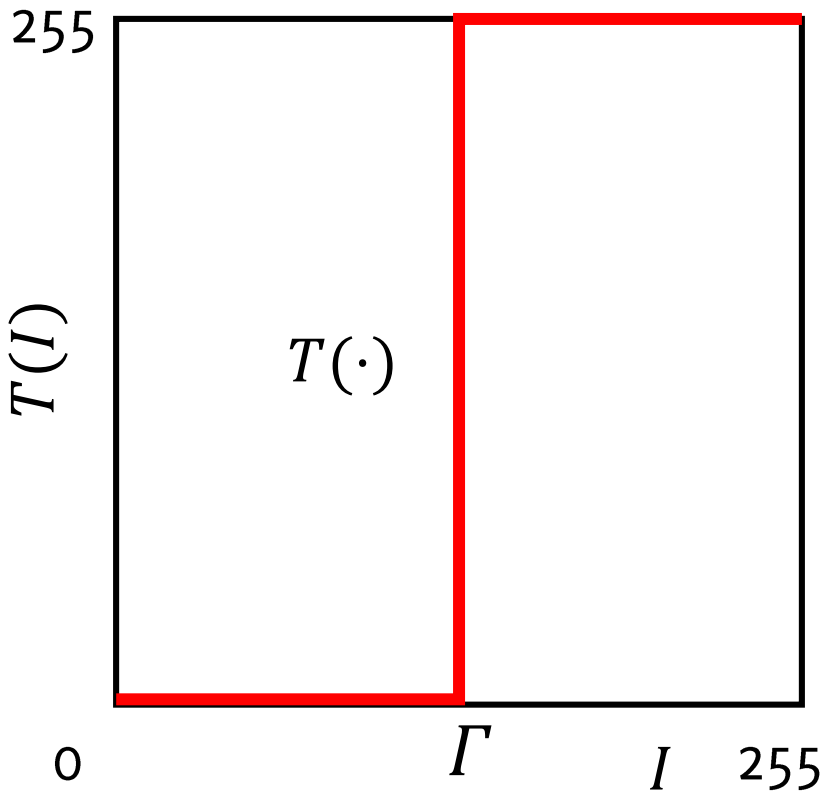


Thresholding binarizes images

Thresholding

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

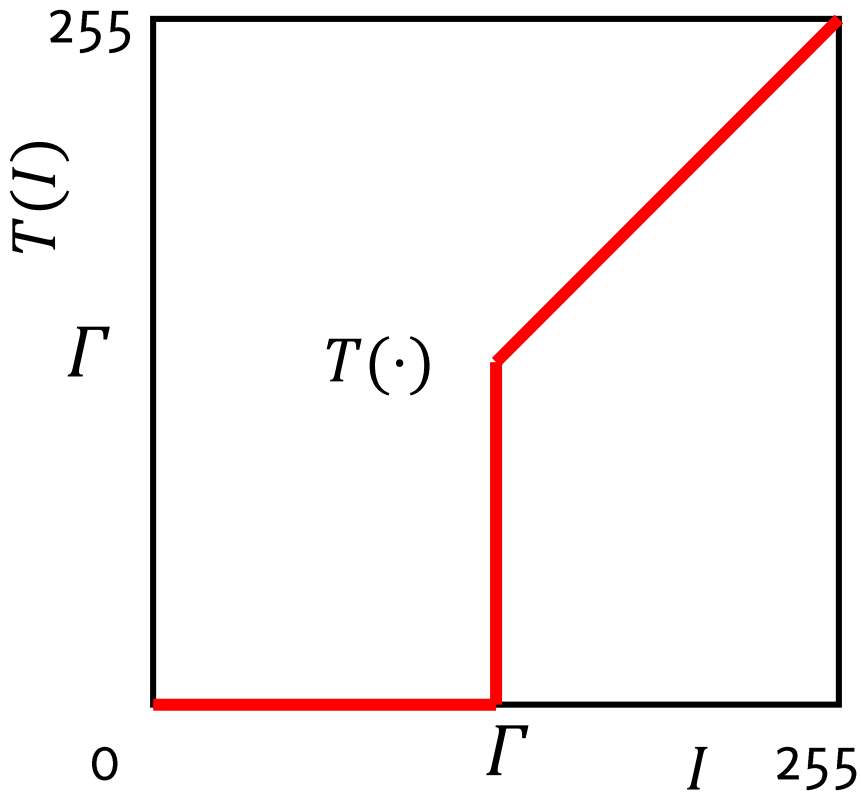
$$T\big(I(r,c)\big) = \begin{cases} 255, & \text{if } I(r,c) \geq \Gamma \\ 0, & \text{if } I(r,c) < \Gamma \end{cases}$$



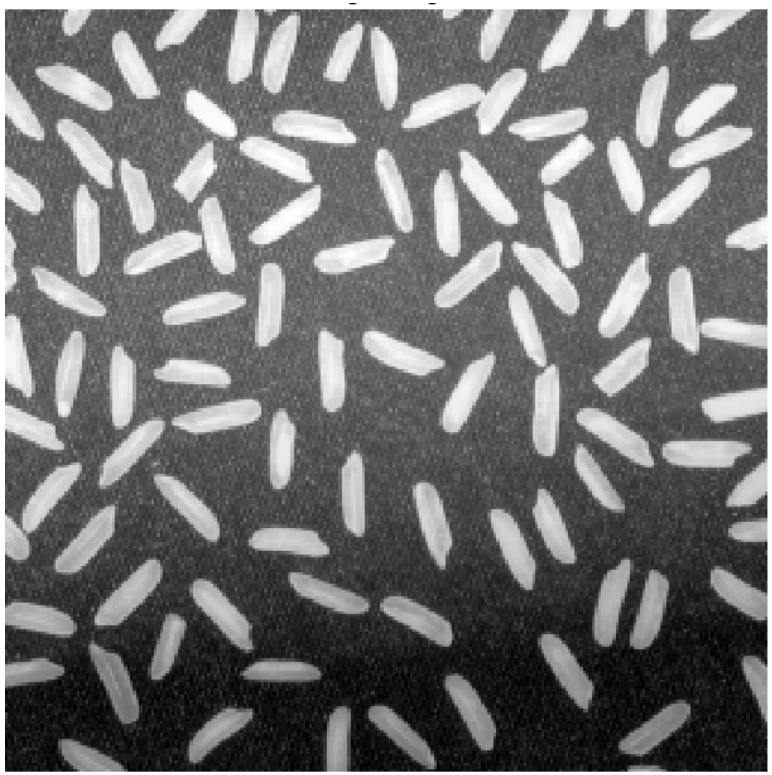
Thresholding

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

What does this T do?



Thresholding

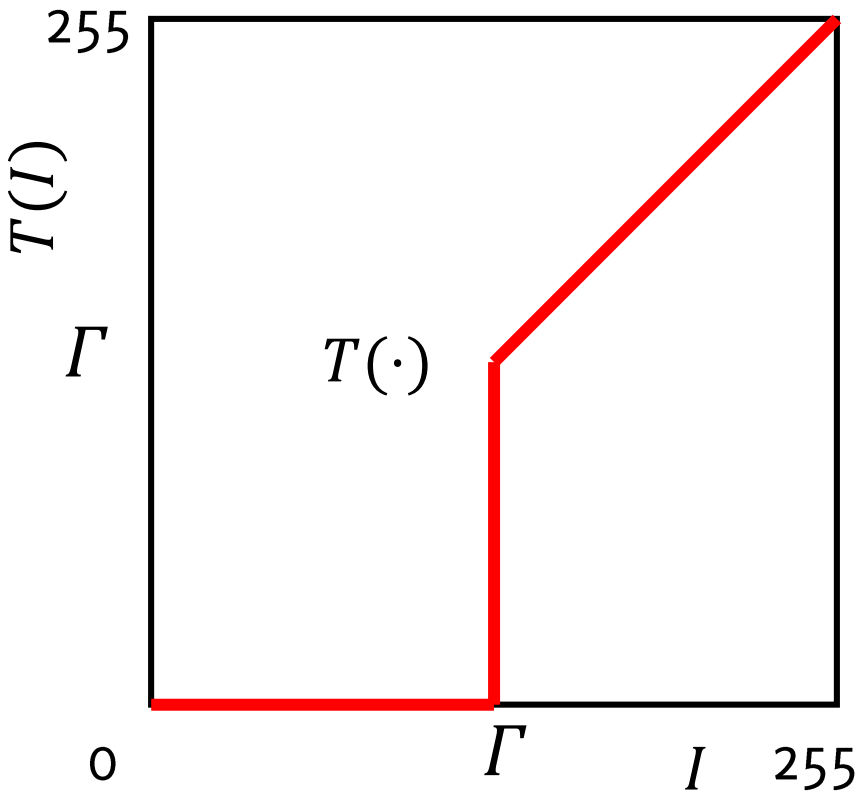


Thresholding

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

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

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



Local (Spatial)Transformations: Correlation and Convolution

Most important image processing operations for classification problems

Local (Spatial) Transformation

In general, these can be written as

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

Where

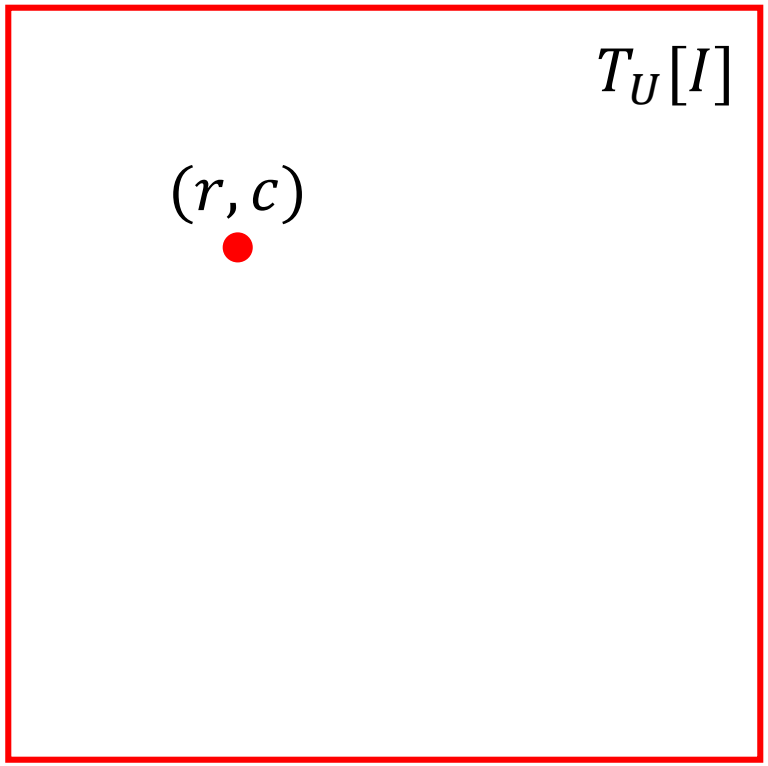
- I is the input image to be transformed
- G is the output
- T_U: $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ or T_U: $\mathbb{R}^3 \rightarrow \mathbb{R}$ is a function
- U is a neighbourhood, identifies a region of the image that will concur in the output definition

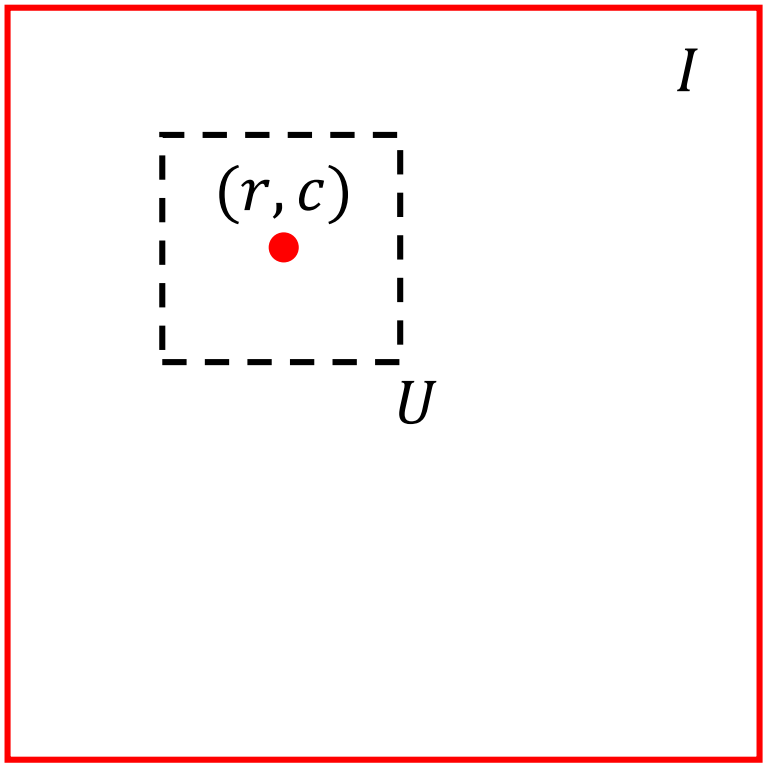
T operates on I "around" U

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

Local (Spatial) Filters

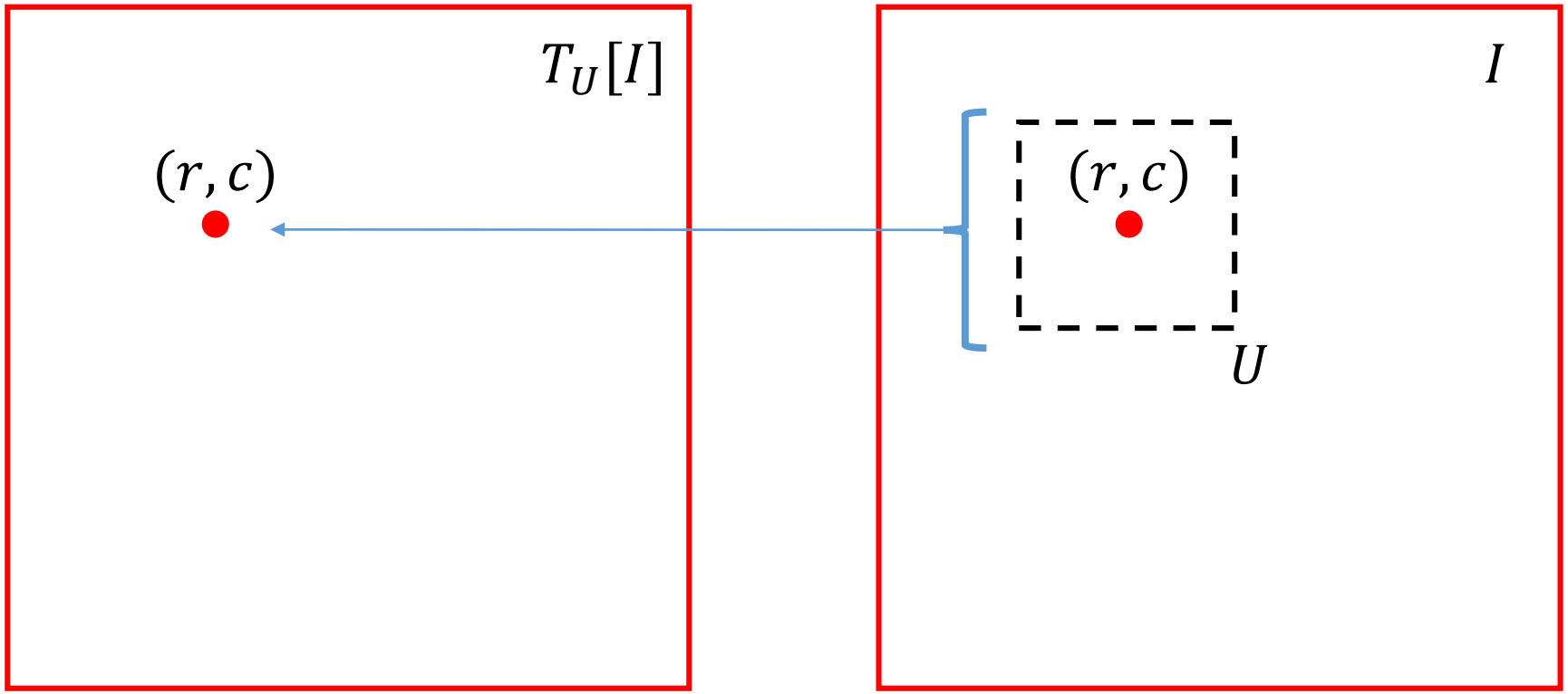
The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$





Local (Spatial) Filters

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



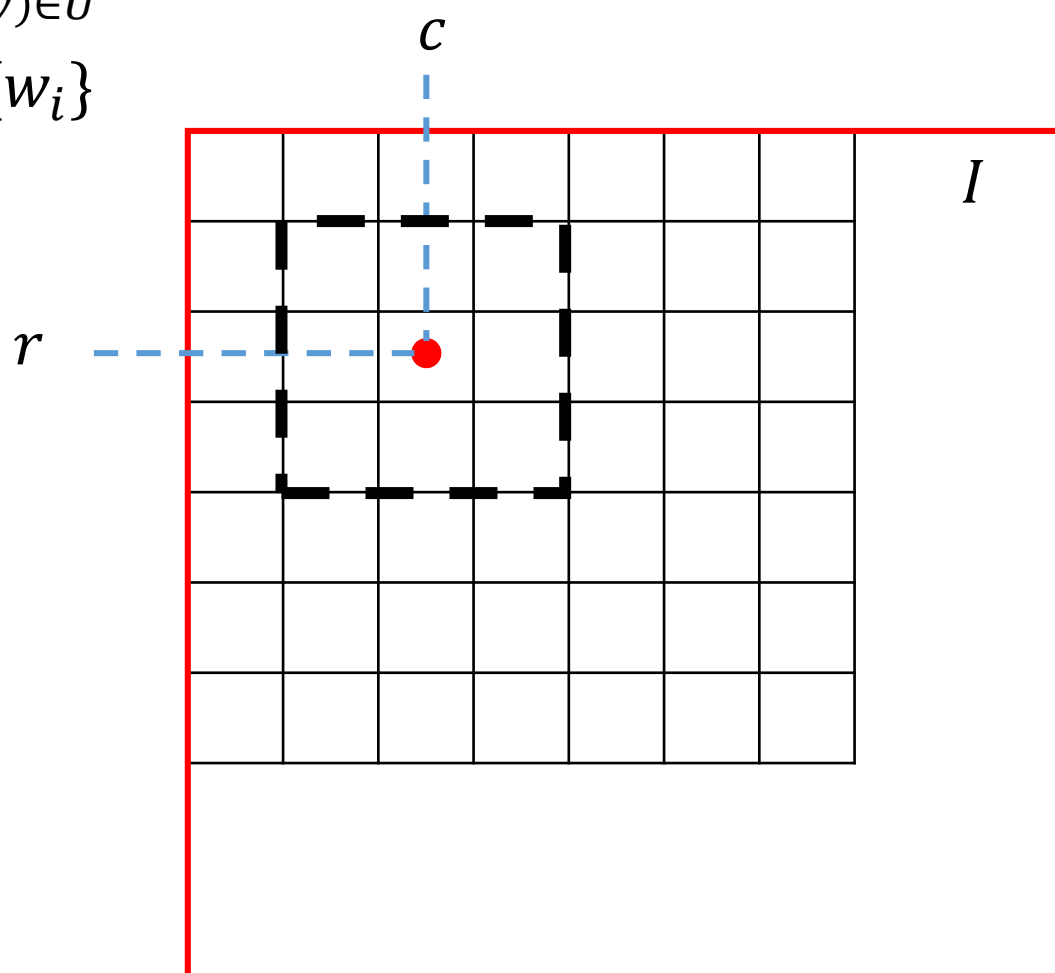
- The location of the output does not change
- The operation is repeated for each pixel
- T can be either linear or nonlinear

Local Linear Filters

Linear Transformation: Linearity implies that

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

Considering some weights $\{w_i\}$



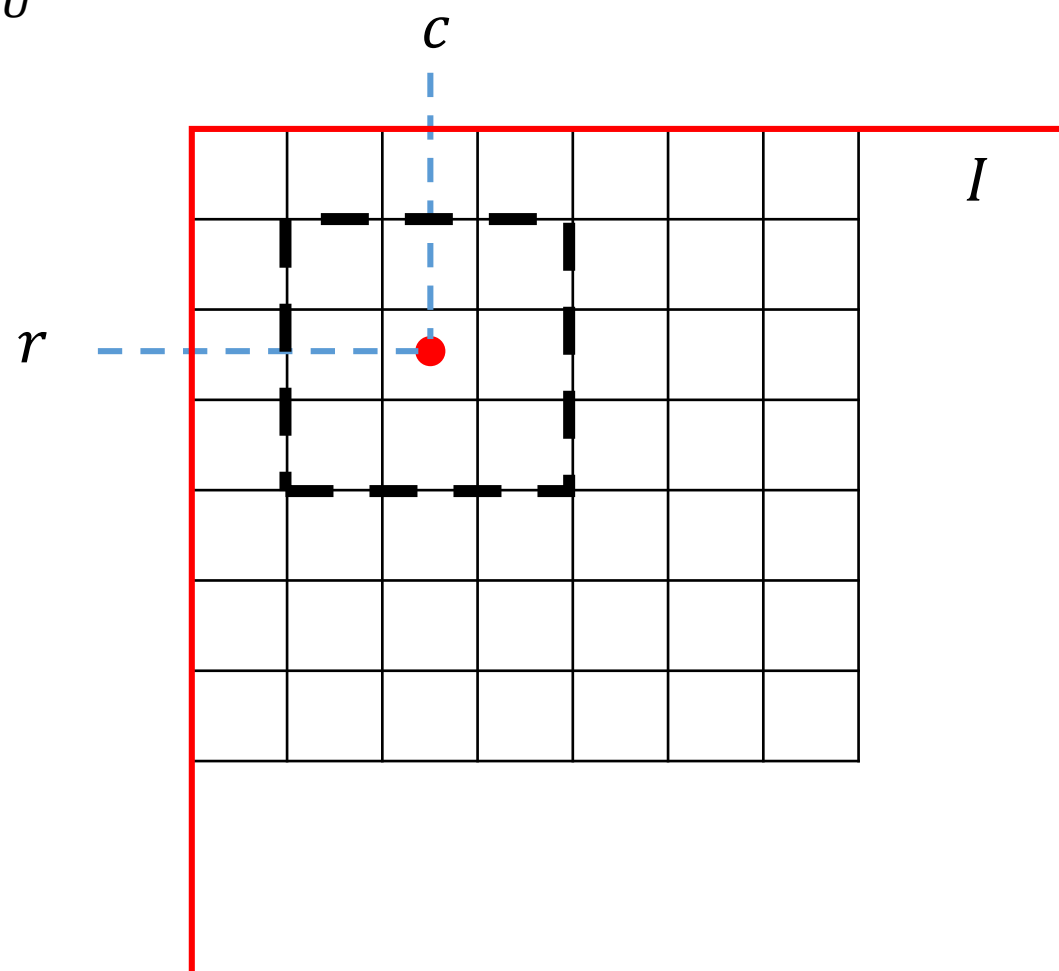
Local Linear Filters

Linear Transformation: Linearity implies that

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

U

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



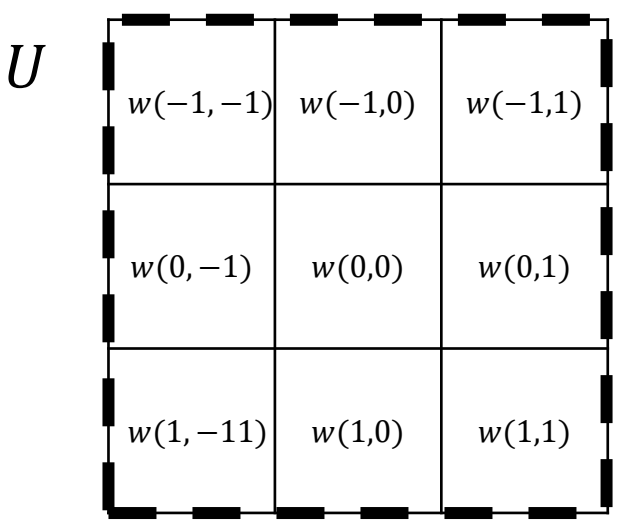
We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation

Local Linear Filters

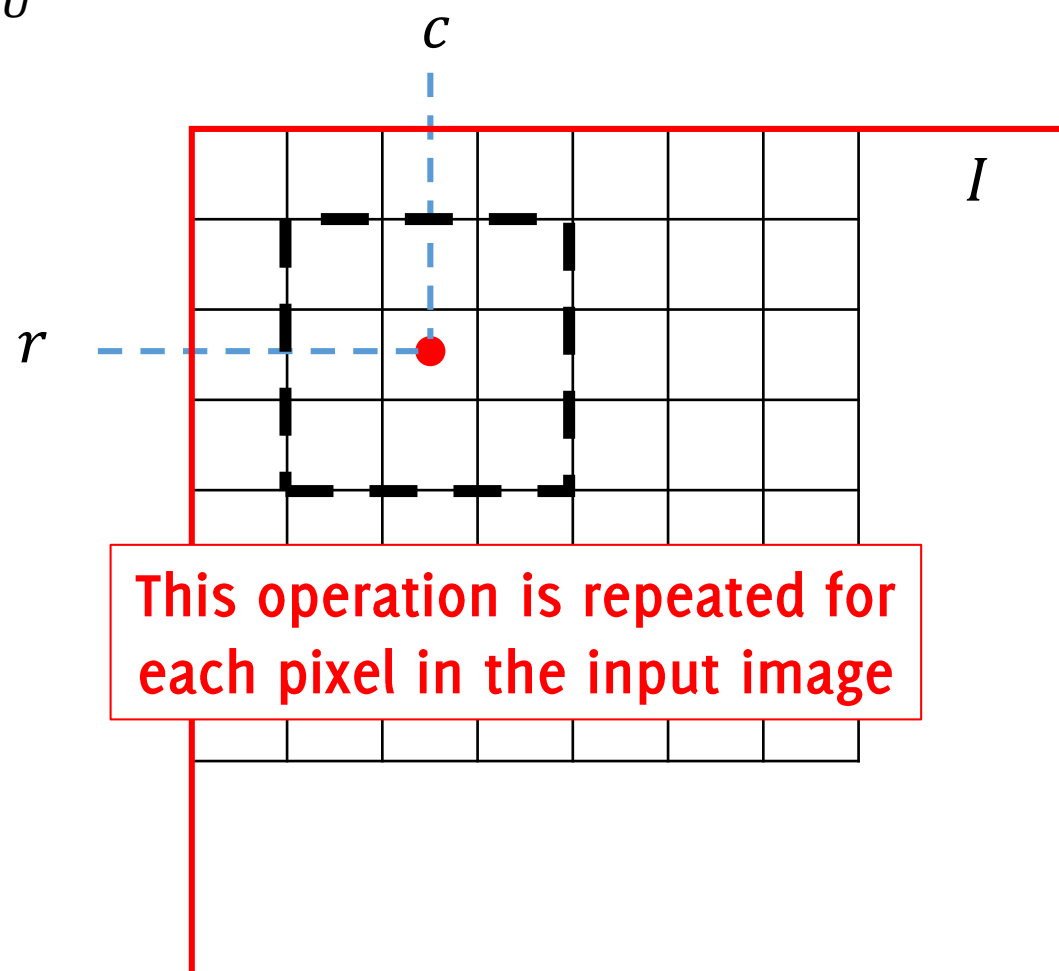
Linear Transformation: Linearity implies that

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



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

The filter h entirely defines this operation



This operation is repeated for each pixel in the input image

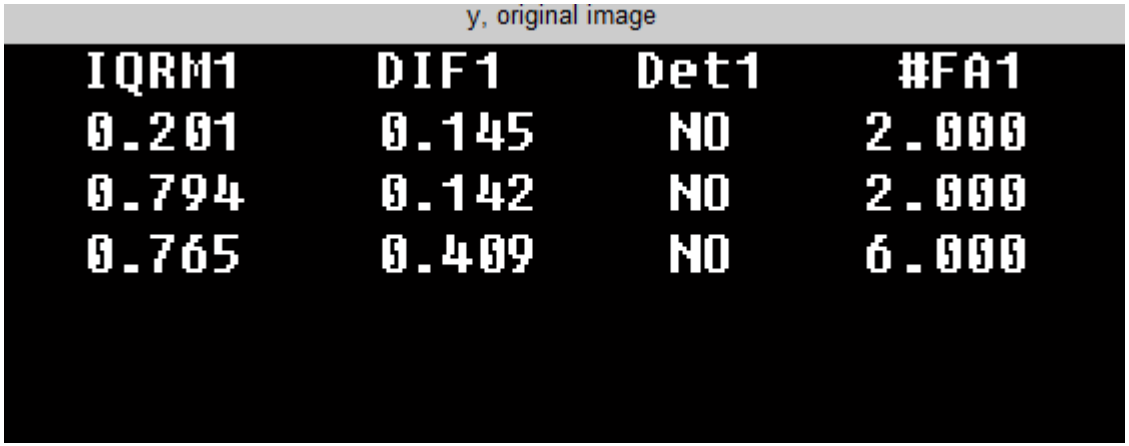
Correlation

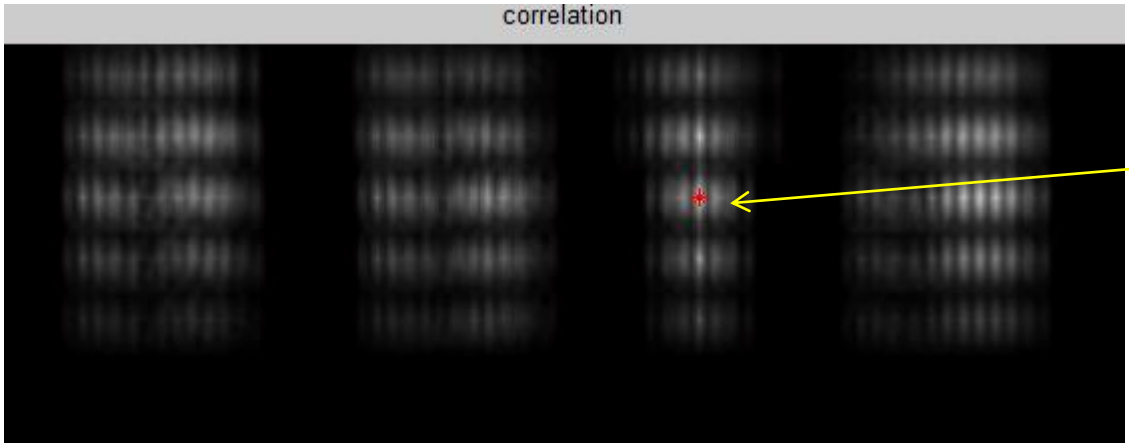
The **correlation** among a filter h and an image is defined as

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

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

Another example





The maximum is here

However…

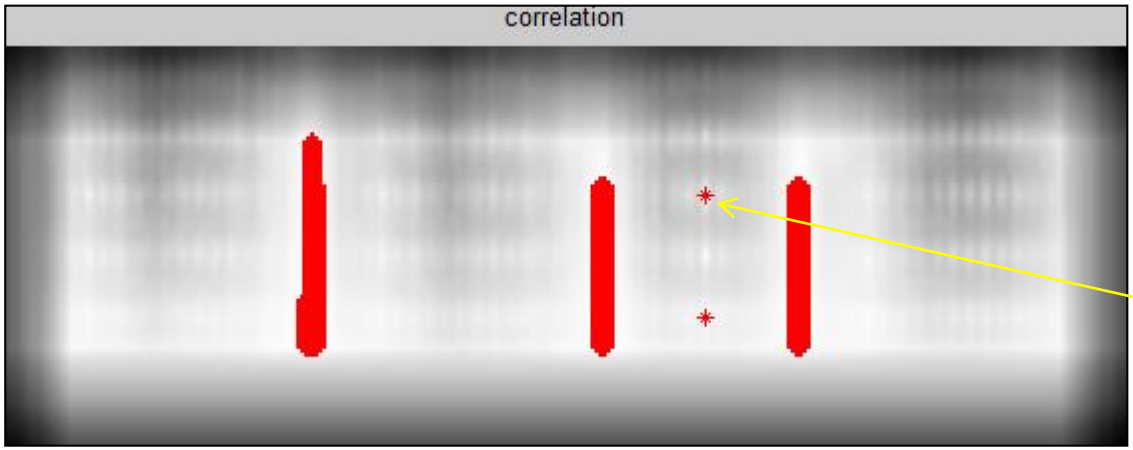
y, original image

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

* template

NO
NO
NO

=



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

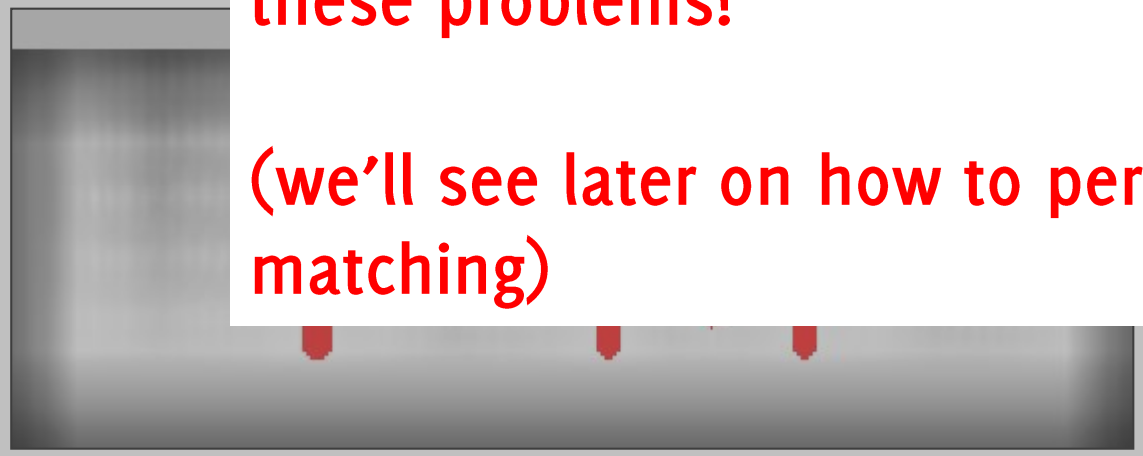
However…

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

template

NO
NO
NO



Normalization is needed to prevent these problems!

(we'll see later on how to perform template matching)

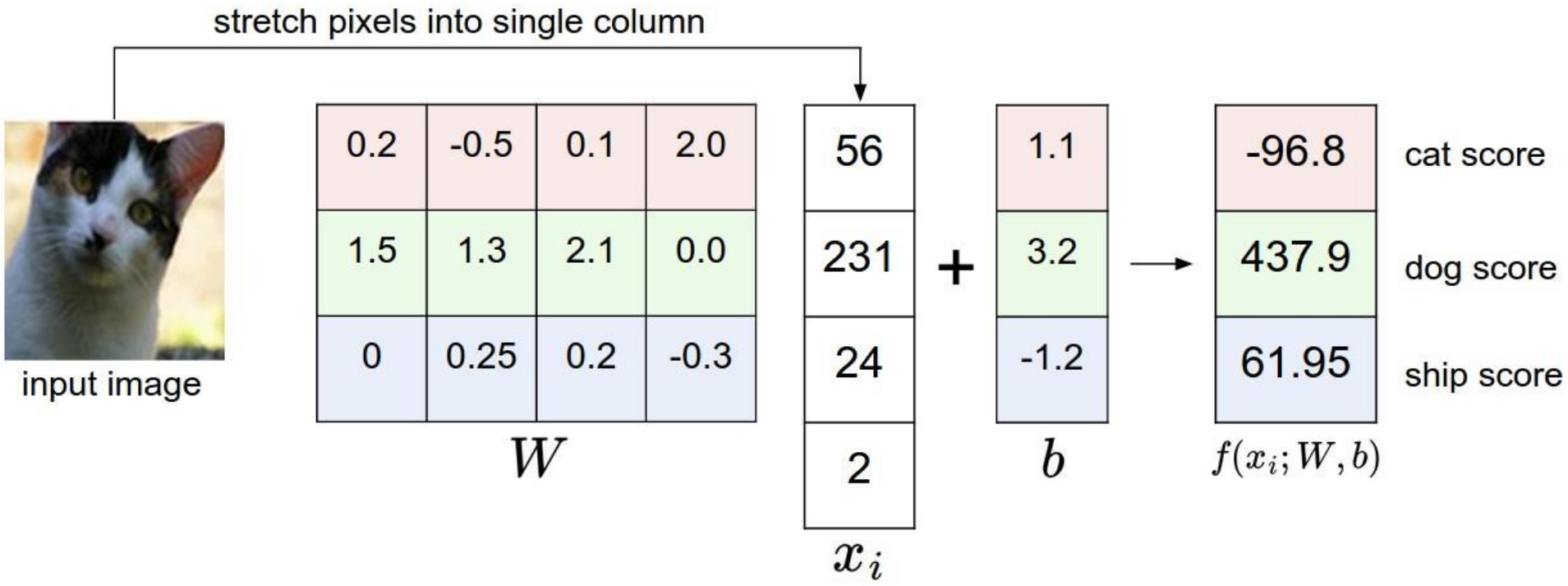
the

maxium value (togheter with the perfect match)

Let's go back to our CIFAR-10 classification problem

The score of a class is the weighted sum of all the image pixels. Weights are actually the classifier parameters.

Weights indicate which are the most important pixels / colors



Template Matching Interpretation

In Matlab/Python notation:

$W(i,:)$ is a d −dimensional vector containing the weights of the score function for the i −th class

Computing the score function for the i −th class corresponds to computing the inner product

$$W(i,:) * \boldsymbol{x}$$

Then, $\boldsymbol{W(i,:)}$ **can be seen as a template used in matching (the output of correlation in the central pixel)**

The template $W(i,:)$ is learned to match at best images belonging to the i −th class

Let's have a look at these templates

Templates Learned on the CIFAR-10 dataset

CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Linear Classifier as a Template Matching

What has the classifier learned?

- That the background of bird and frog is green, (plane and boat is blue)
- Cars are typically red
- Horses have two heads! ☺

The model was definitively too simple / data were not enough for achieving higher performance and better templates

However:

- Linear Classifiers are among the most important layer of NN
- Such a simple model can be interpreted (with more sophisticated models you typically can't)

Convolution

Correlation and Convolution

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

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

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

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

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

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

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

Convolution – and filter flip

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

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

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

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

Convolution – and filter flip

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

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

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Point-wise product

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

.*

y_1	y_2	y_3
y_4	y_5	y_6
y_7	y_8	y_9

→

h_9y_1	h_8y_2	h_7y_3
h_6y_4	h_5y_5	h_4y_6
h_3y_7	h_2y_8	h_1y_9

Convolution - LTI Systems on Images

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

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

$h =$

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$ →

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$ →

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

Point-wise product

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$.*$

y_1	y_2	y_3
y_4	y_5	y_6
y_7	y_8	y_9

→

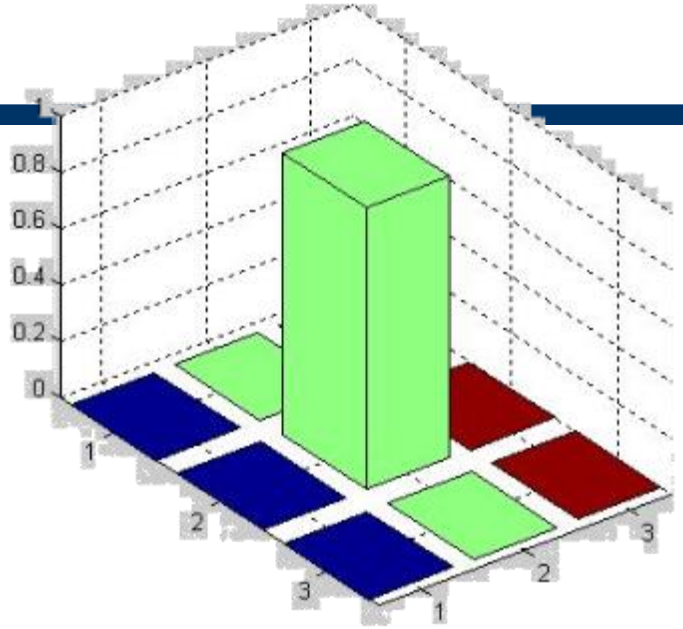
h_9y_1	h_8y_2	h_7y_3
h_6y_4	h_5y_5	h_4y_6
h_3y_7	h_2y_8	h_1y_9

sum →

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

A well-known Test Image - Lena

A Trivial example

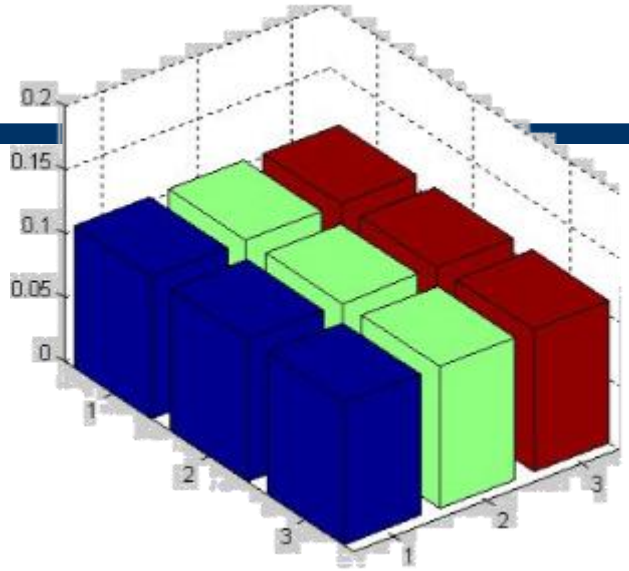


*

0	0	0
0	1	0
0	0	0

=

Linear Filtering



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

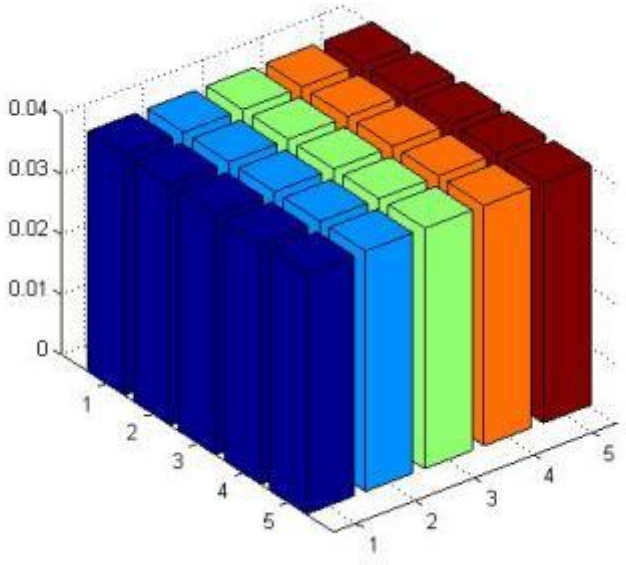
1	1	1
1	1	1
1	1	1

$$=$$

?

The original Lena image

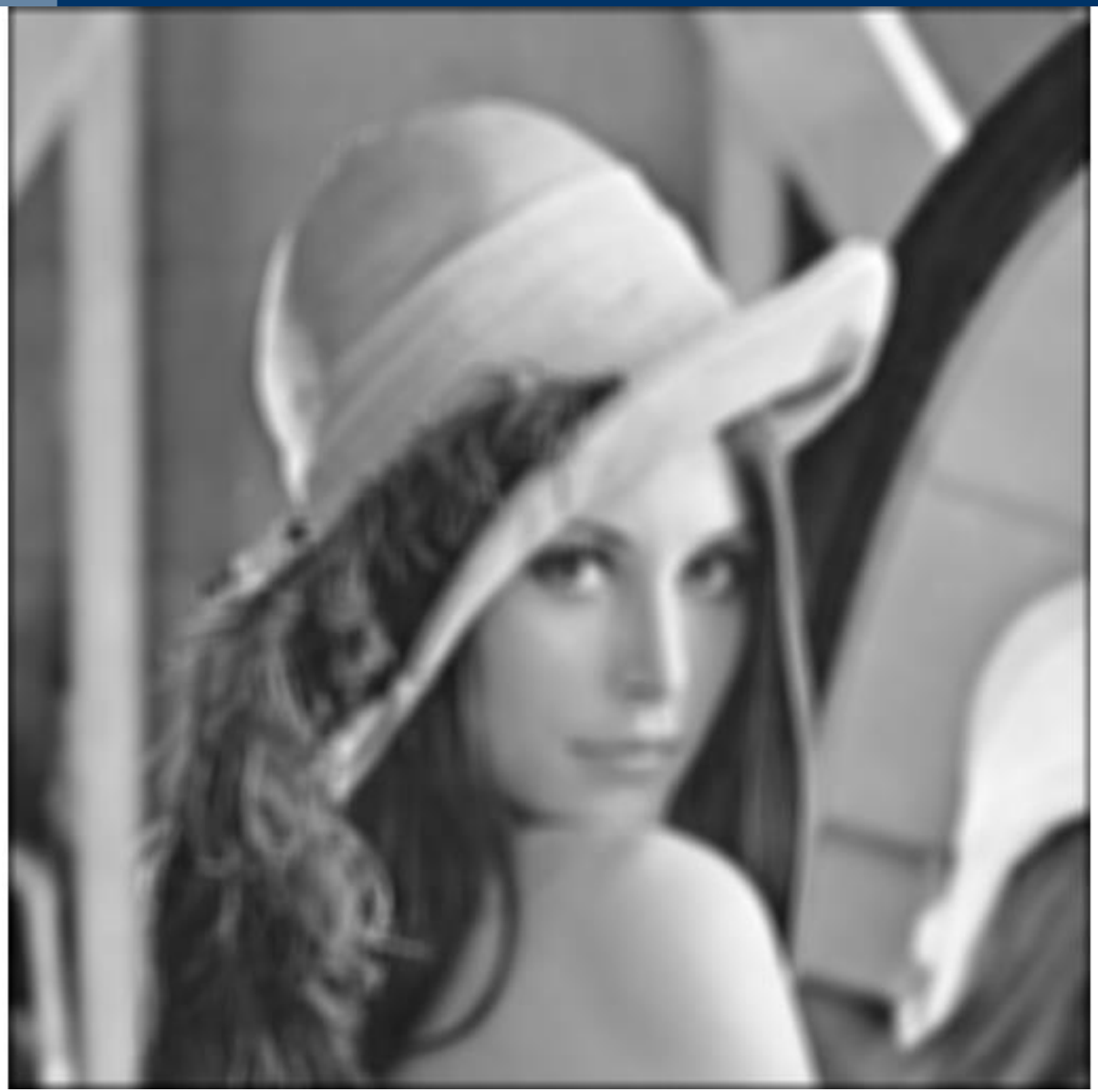
Filtered Lena Image



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

The original Lena image

The filtered Lena image



What about normalization?

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

… convolution is linear

...what about

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

… convolution is linear

Properties of Convolution

It is a linear operator

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

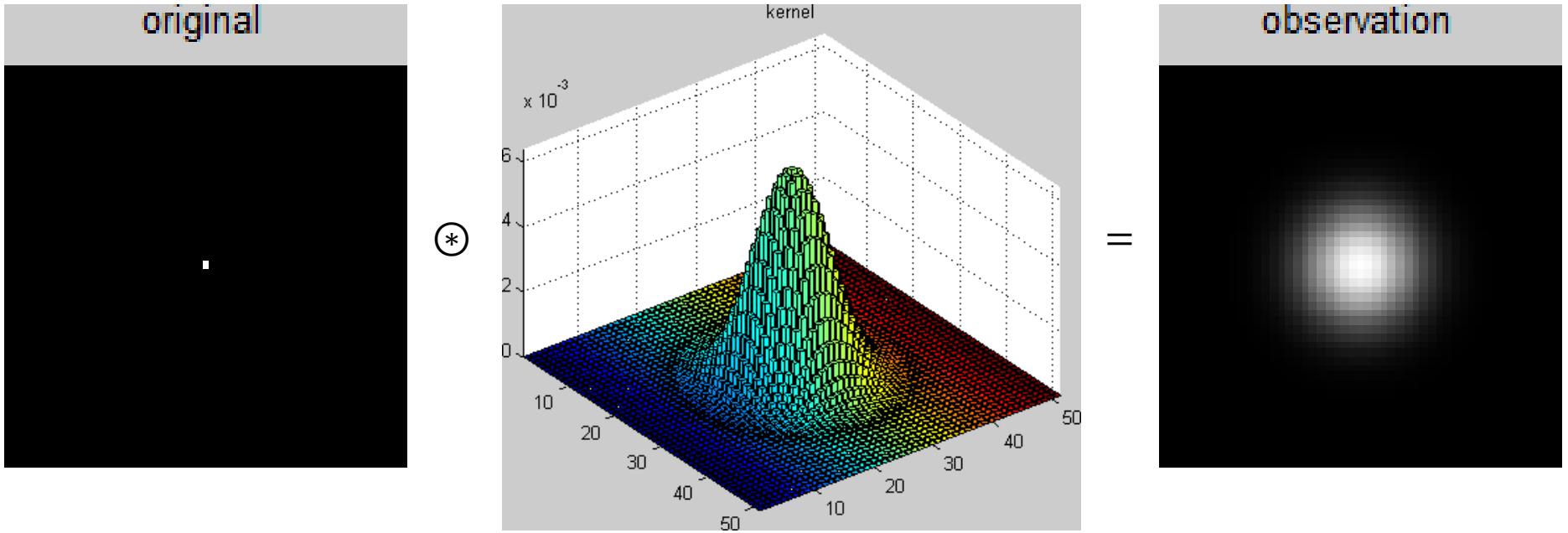
where $\lambda, \mu \in \mathbb{R}$

It is entirely characterized by its filter h, which is also called "the kernel«-

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

The Impulse Response

Take as input image a discrete Dirac



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

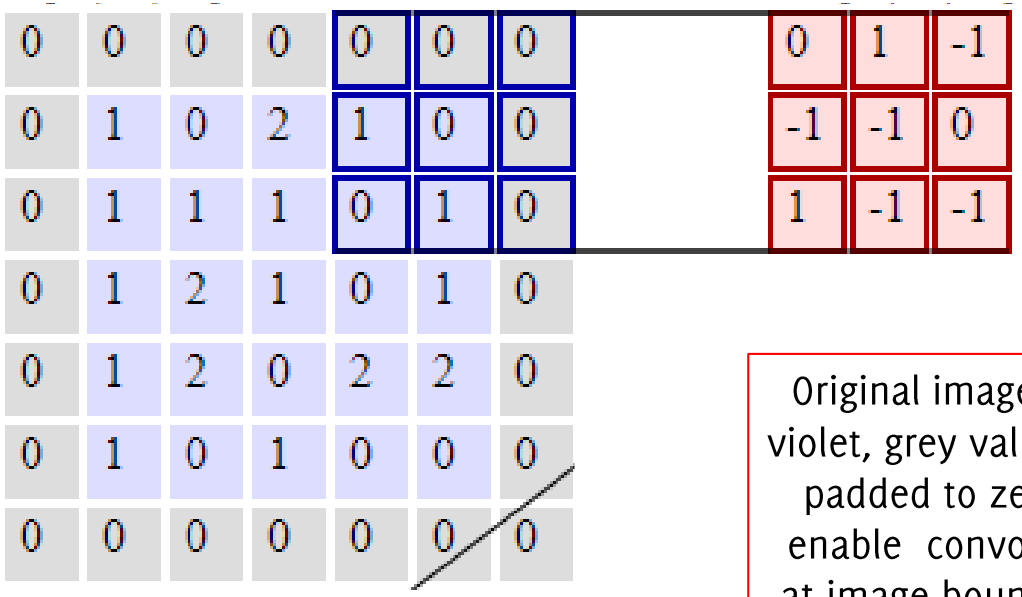
Properties of Convolution

It is **commutative (in principle)**

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

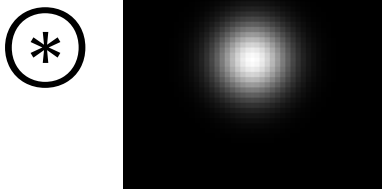
However, in discrete signals it depends on **the padding criteria**
In continuous domain it holds as well as on periodic signals

Input image I filter h

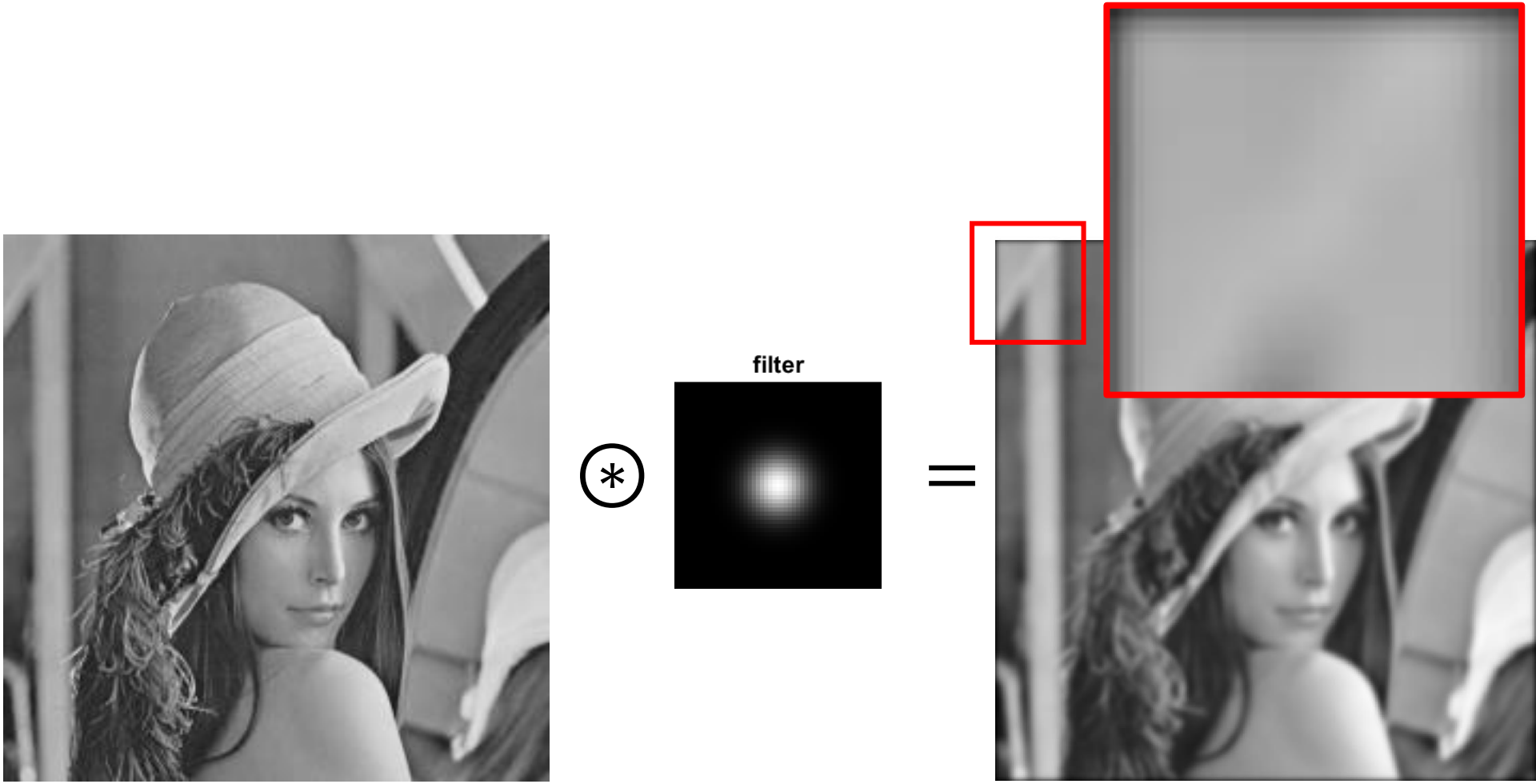


Original image is in violet, grey values are padded to zero to enable convolution at image boundaries

Convolution is Commutative?

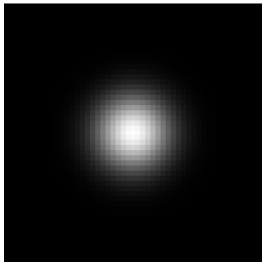


Convolution is Commutative?



Convolution is Commutative?

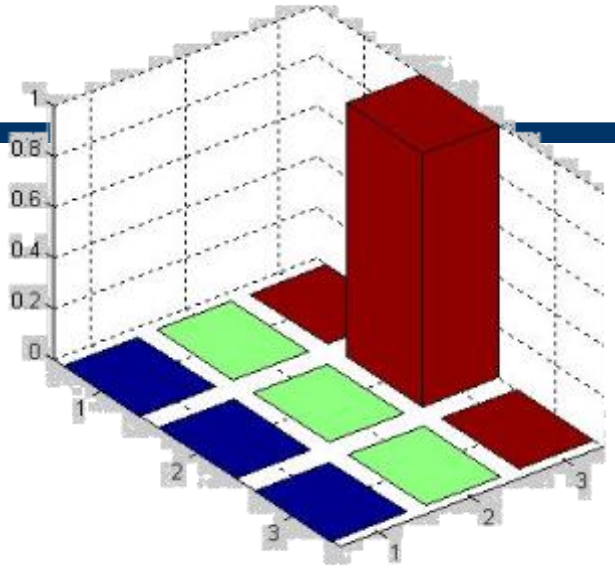
filter



$*$

$=$

Transalation



$\circledast$

0	0	0
1	0	0
0	0	0

$=$

Remember the filter has to be flipped before convolution

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

So it definitively depends on the padding options:

- **In CNNs** typically you drop pixels where filter goes into the padding area -› convolution reduces image size.

Convolution can be defined on arbitrary image size

- **In CNNs** typically filters span multiple image layers. In color image filtering instead, filters are 2D and operate color-wise

Properties of Convolution

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

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

Properties of Convolution

It is also **associative**

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

and **dissociative**

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

It is **shift-invariant**, namely

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

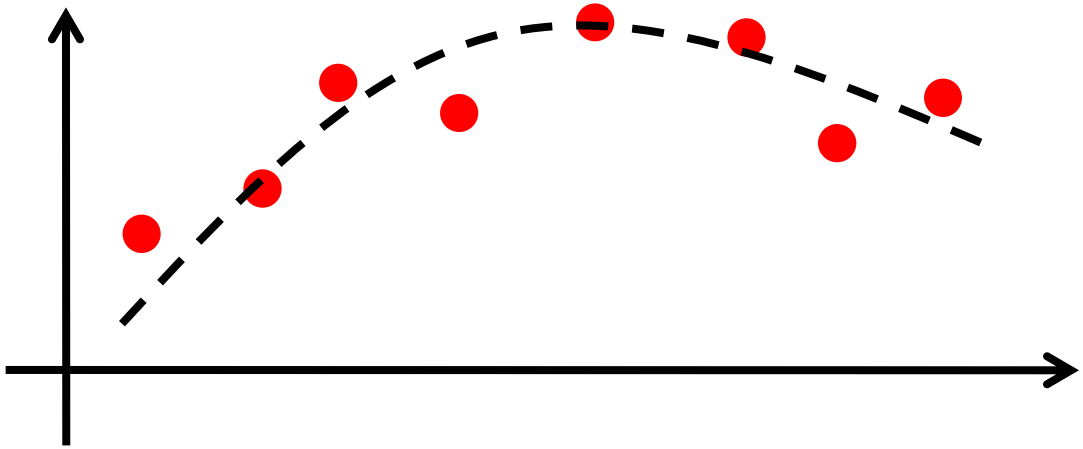
Any **linear and shift invariant system can be written as a convolution**

Convolution and Regression

Observation model is

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

Consider a regression problem



Fitting and Convolution

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

The problem: estimating the constant C fitting the noisy observations

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

where

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

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

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

Derivative Estimation

Differentiation and convolution

Recall

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

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

Differentiation and convolution

Recall

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

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

- We could approximate this as

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

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

Finite Difference in 2D

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

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

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

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

Discrete Approximation

Horizontal

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

Vertical

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

Convolution Kernels

Differentiating Filters

The derivatives can be also computed using centered filters:

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

Such that the horizontal derivative is:

$f_x = f \circledast$

1	0	-1

While the vertical derivative is:

$f_y = f \circledast$

1	0	-1

t

Classical Operators: Prewitt

Horizontal derivative

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

Smooth Differentiate

Vertical derivative

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

Smooth Differentiate

Classical Operators: Sobel

Horizontal derivative

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

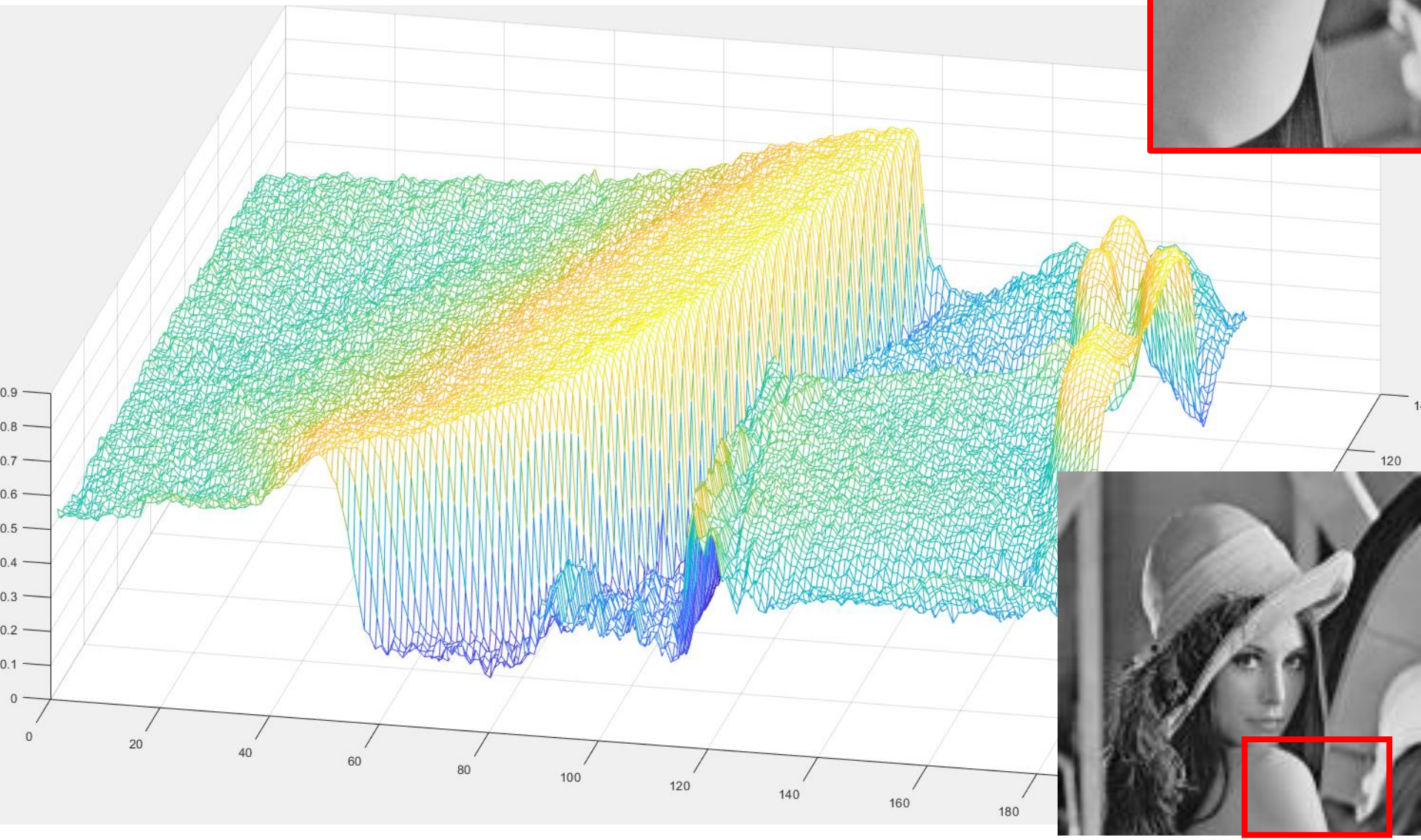
Smooth Differentiate

Vertical derivative

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

Smooth Differentiate

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



The Image Gradient

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

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

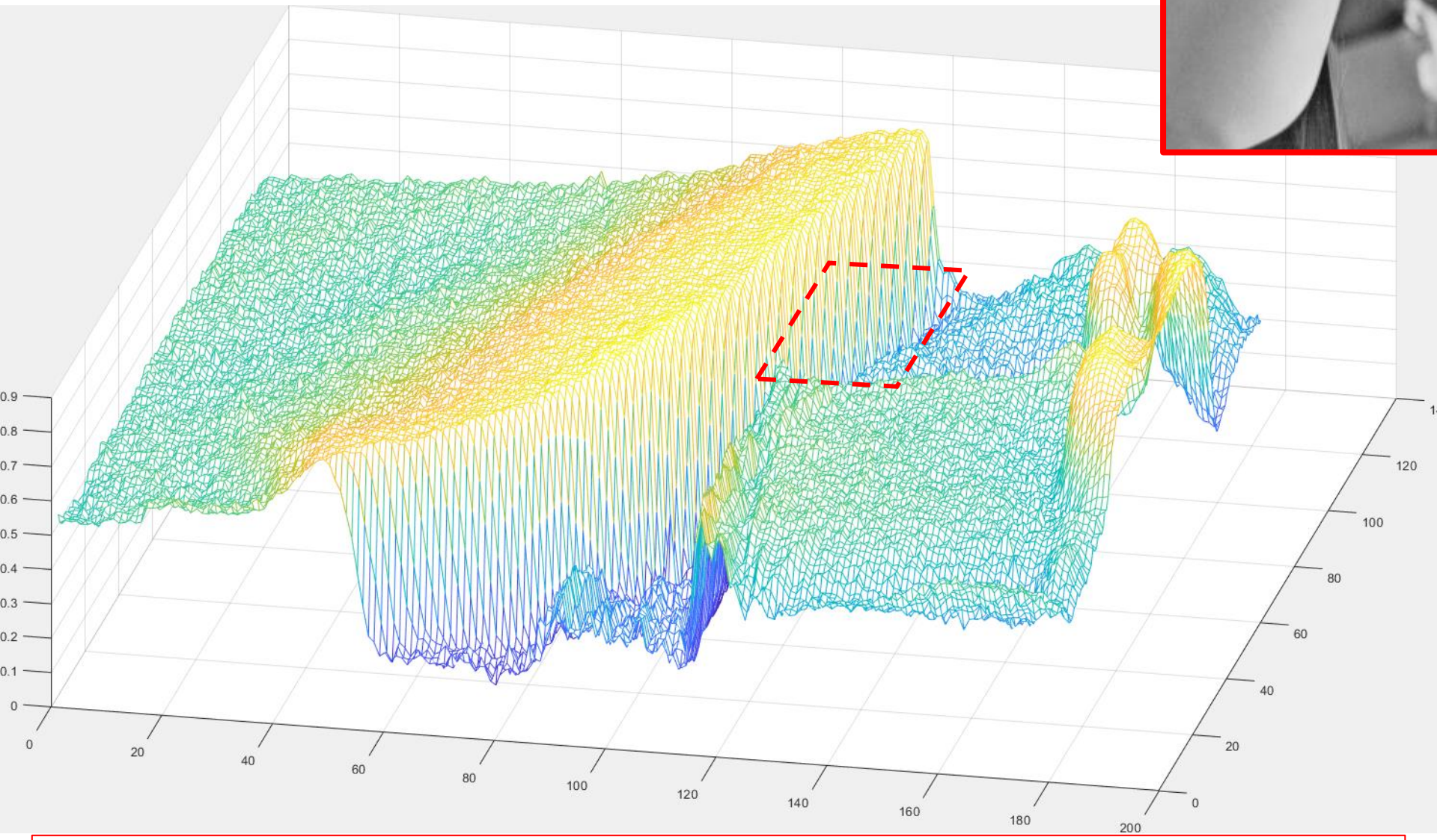
where principal derivatives are computed through convolution against the derivative filters (e.g. Prewitt)

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

Image gradient behaves likethe gradient of a function:

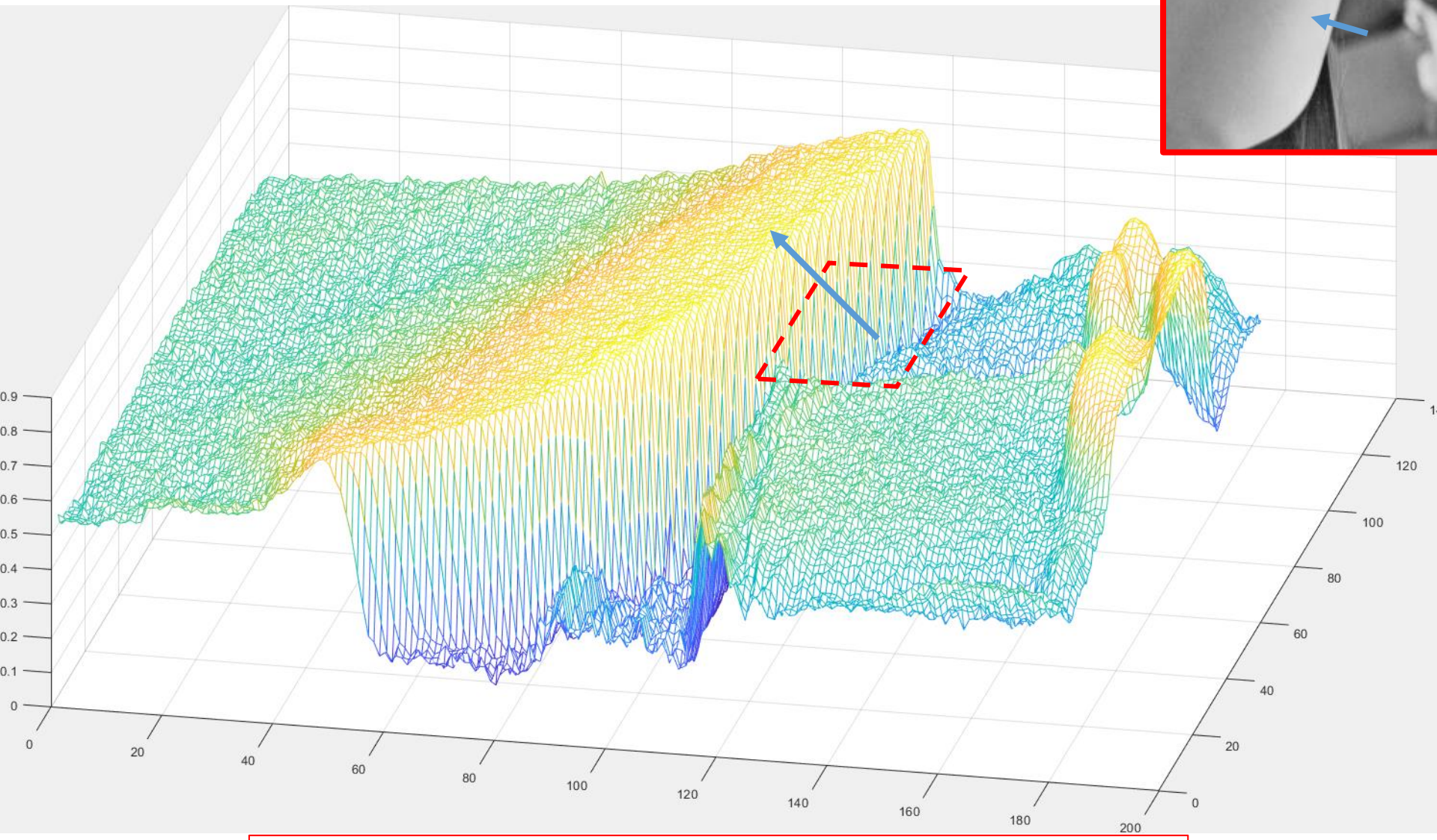
- $|\nabla I(r,c)|$ is large where there are large variations
- $\angle \nabla I(r,c)$ is the direction of the steepest variation

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



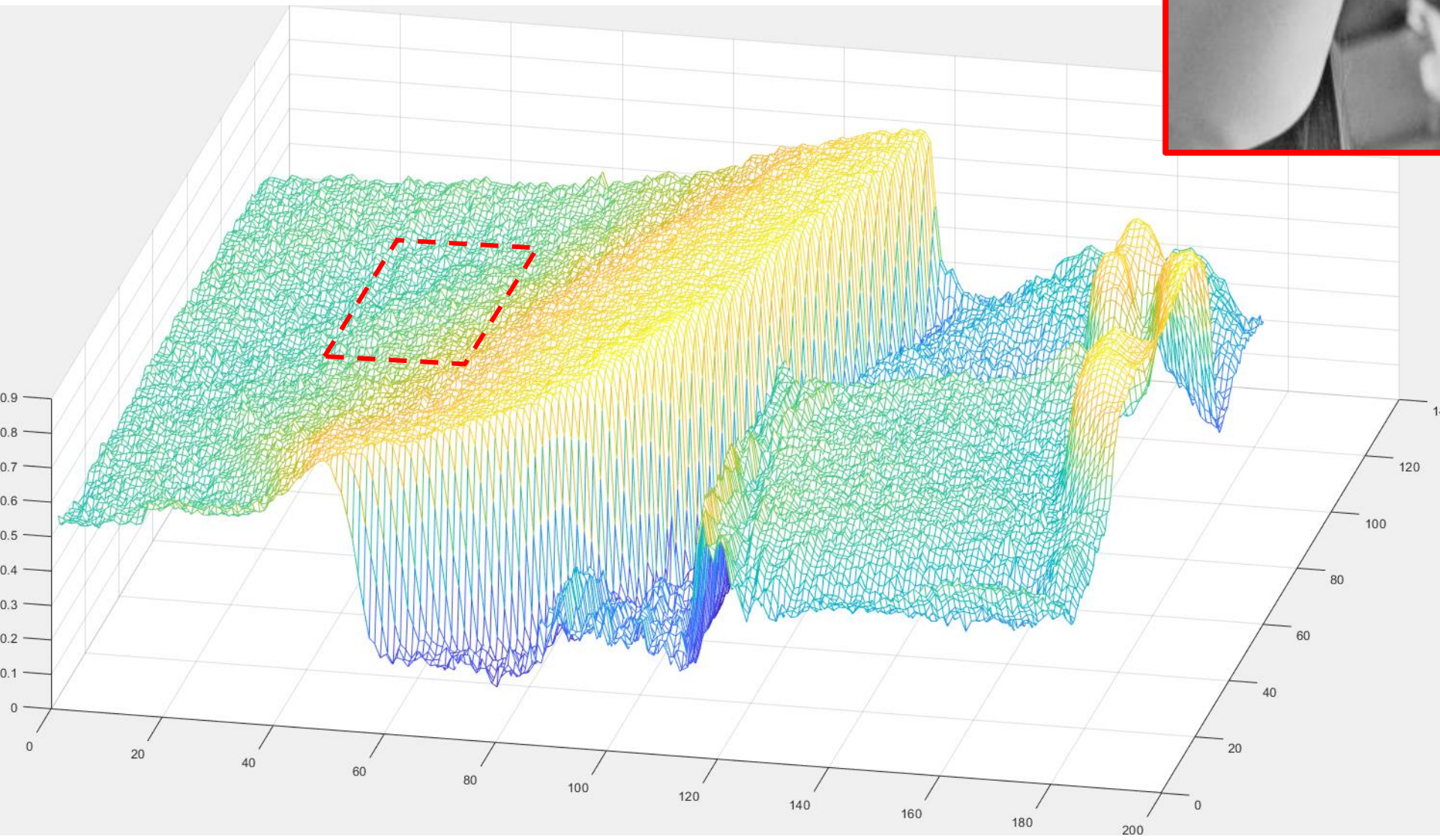
Local spatial transformations are defined over neighborhood like this

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

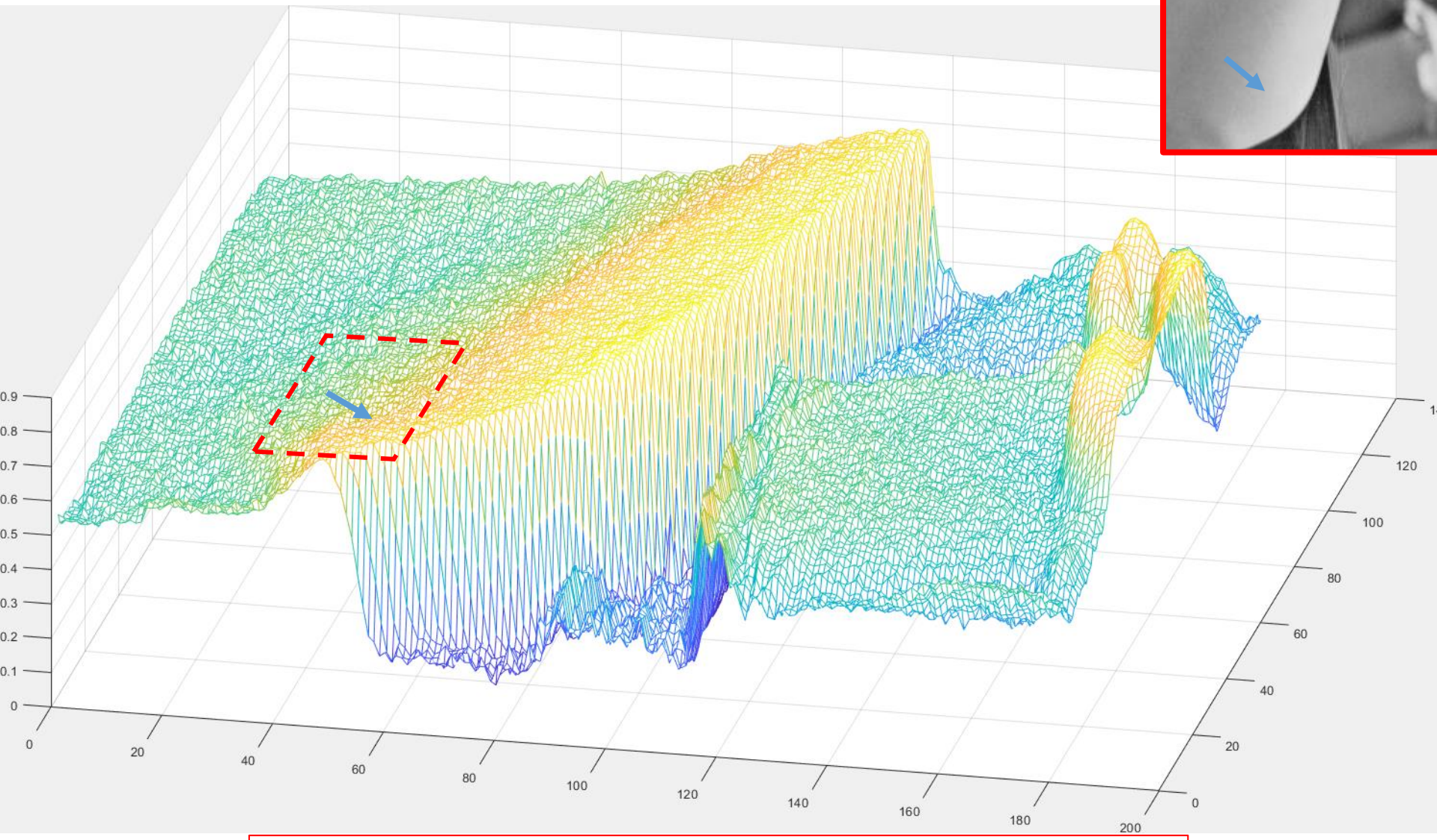


What about the gradient in this neighborhood?

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

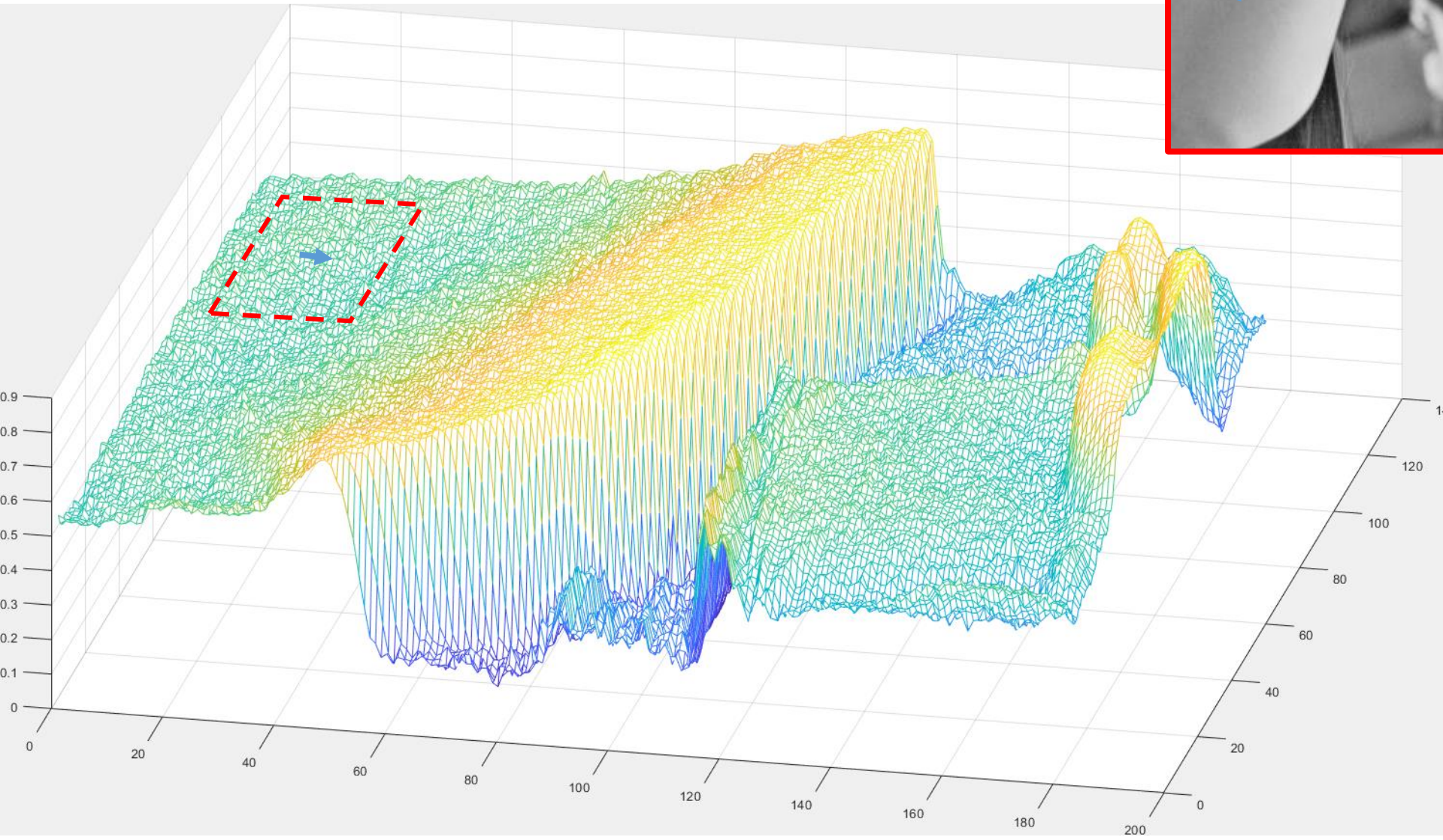


Think of an image as a 2d, real-valued function



What about the gradient in this neighborhood?

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

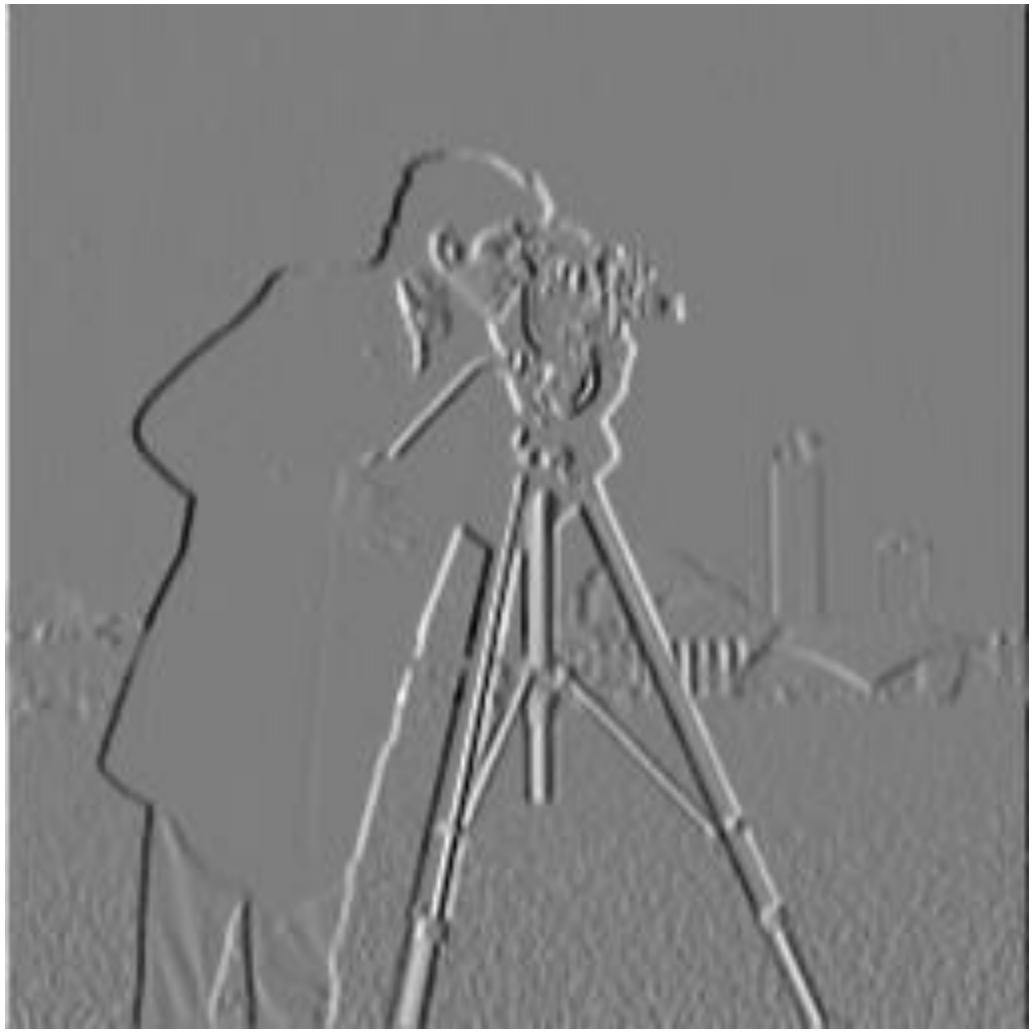


Another famous test image - cameraman

Horizontal Derivatives using Sobel

$$\nabla I_x = I \circledast h_x$$

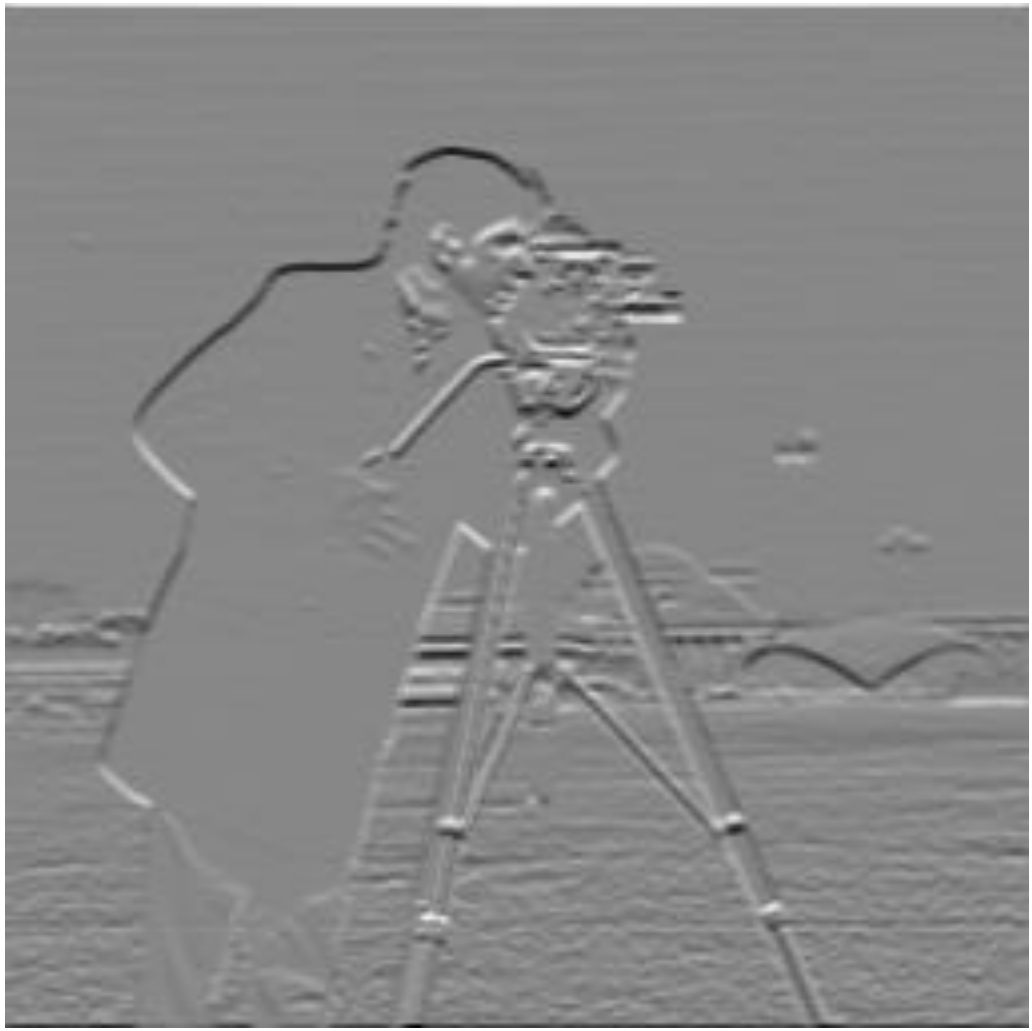
$$\boldsymbol{\nabla} I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y(r,c) \end{bmatrix}$$



Vertical Derivatives using Sobel

$$\nabla I_y = I \circledast h_y$$

$$\nabla I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y(r,c) \end{bmatrix}$$



Gradient Magnitude

$$\|\nabla I\| = \sqrt{(I \circledast h_x)^2 + \left(I \circledast h_y\right)^2}$$

$$\boldsymbol{\nabla} I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y(r,c) \end{bmatrix}$$

Higher Order Derivatives

Second Order Derivatives

The Laplacian of the second order derivative is defined as

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

where

$$\frac{\partial^2 I}{\partial x^2} = I(x+1, y) + I(x-1, y) - 2I(x, y)$$

$$\frac{\partial^2 I}{\partial y^2} = I(x, y-1) + I(x, y+1) - 2I(x, y)$$

$$\nabla^2 I = I(x+1, y) + I(x-1, y) + I(x, y-1) + I(x, y+1) - 4I(x, y)$$

It's a linear operator -> it can be implemented as a convolution

TODO: prove that the second order derivatve is like this

Filter for Digital Laplacian

The Laplacian of the second order derivative is defined as

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

0	1	0
1	-4	1
0	1	0

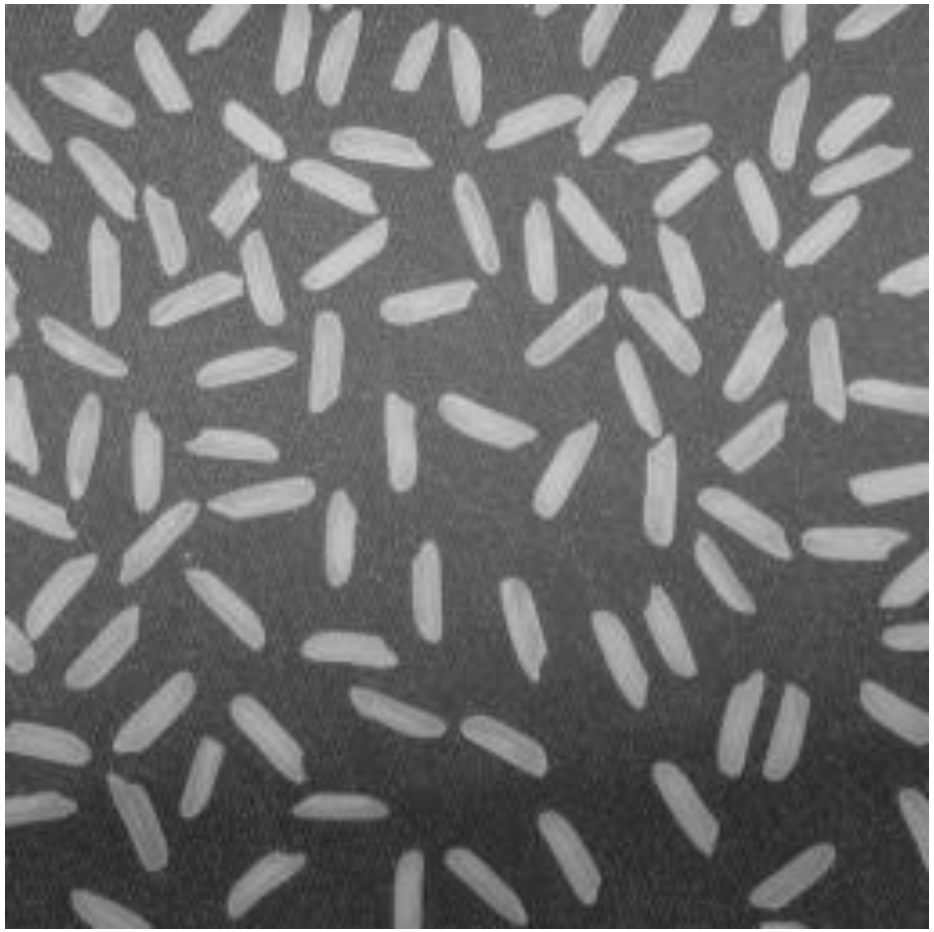
Standard definition, inviariant to 90° rotation

1	1	1
1	-8	1
1	1	1

Alternative definition, inviariant to 45° rotation

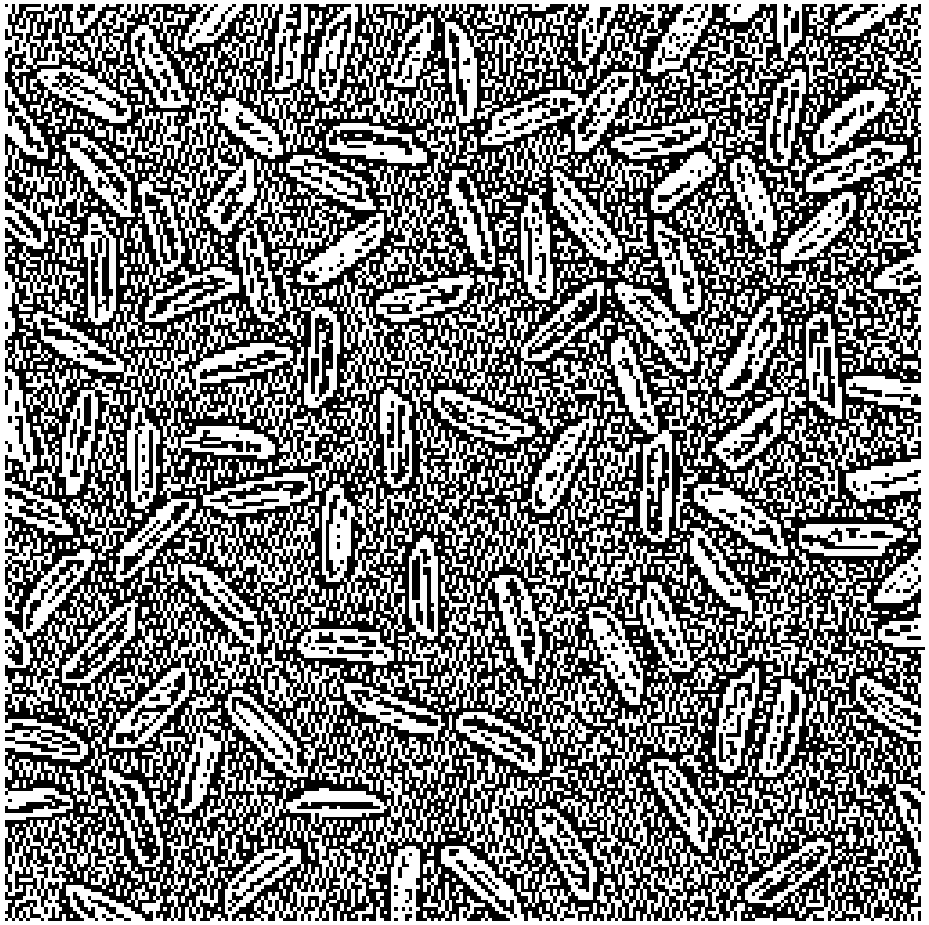
The Laplacian: Image Sharpening

The Laplacian of an image have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.



The Laplacian: Image Sharpening

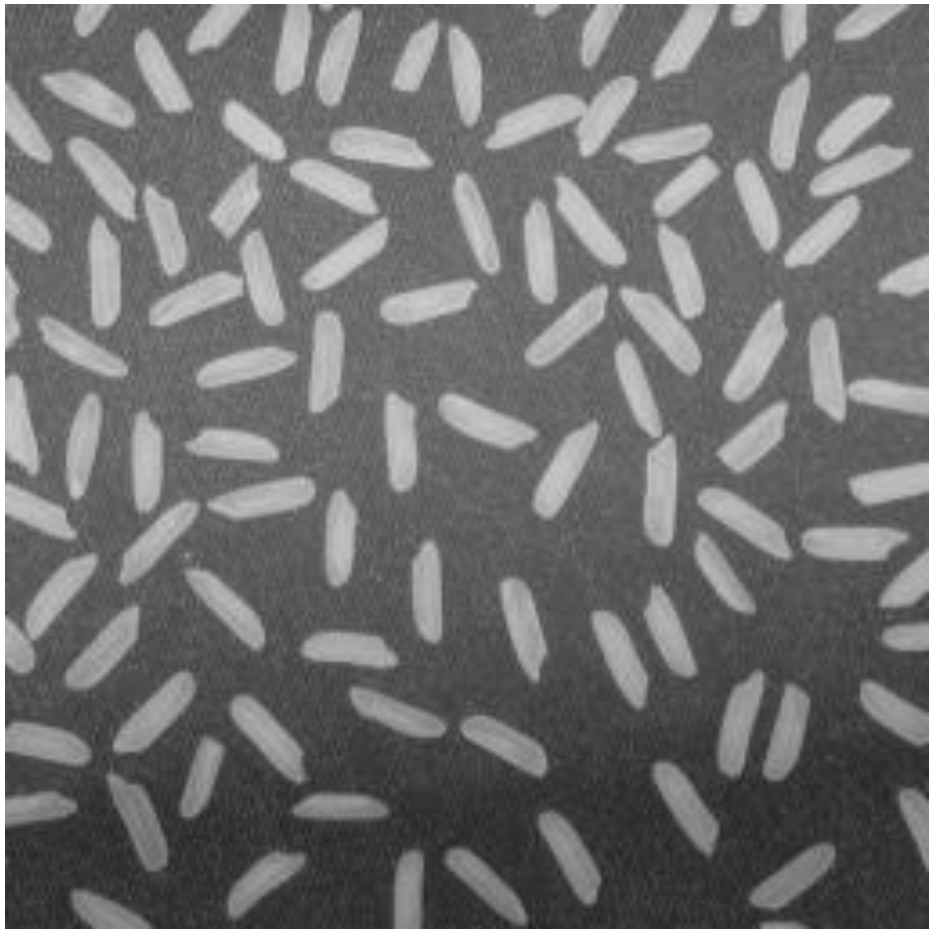
The Laplacian of an image have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.



The Laplacian: Image Sharpening

Background features can be “recovered” simply by adding the Laplacian image to the original (provided suitable rescaling)

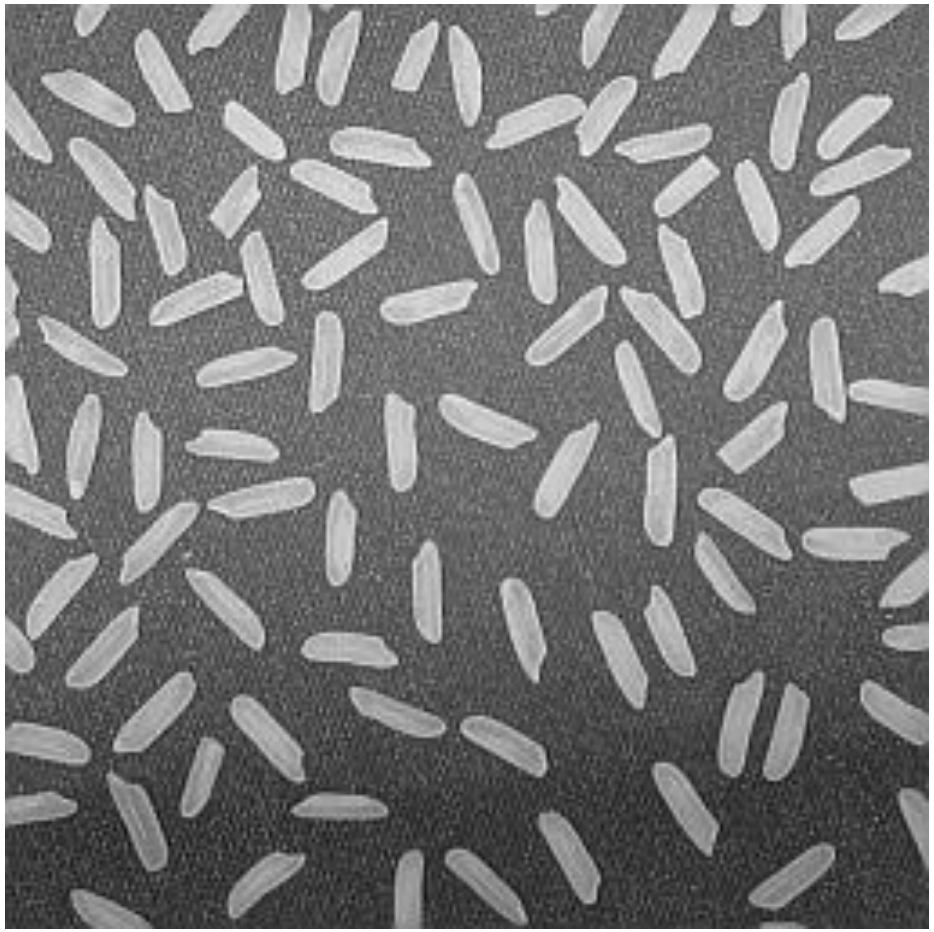
$$G(r,c) = I(r,c) + k[\nabla^2 I(r,c)]$$



The Laplacian: Image Sharpening

Background features can be “recovered” simply by adding the Laplacian image to the original (provided suitable rescaling)

$$G(r,c) = I(r,c) + k[\nabla^2 I(r,c)]$$



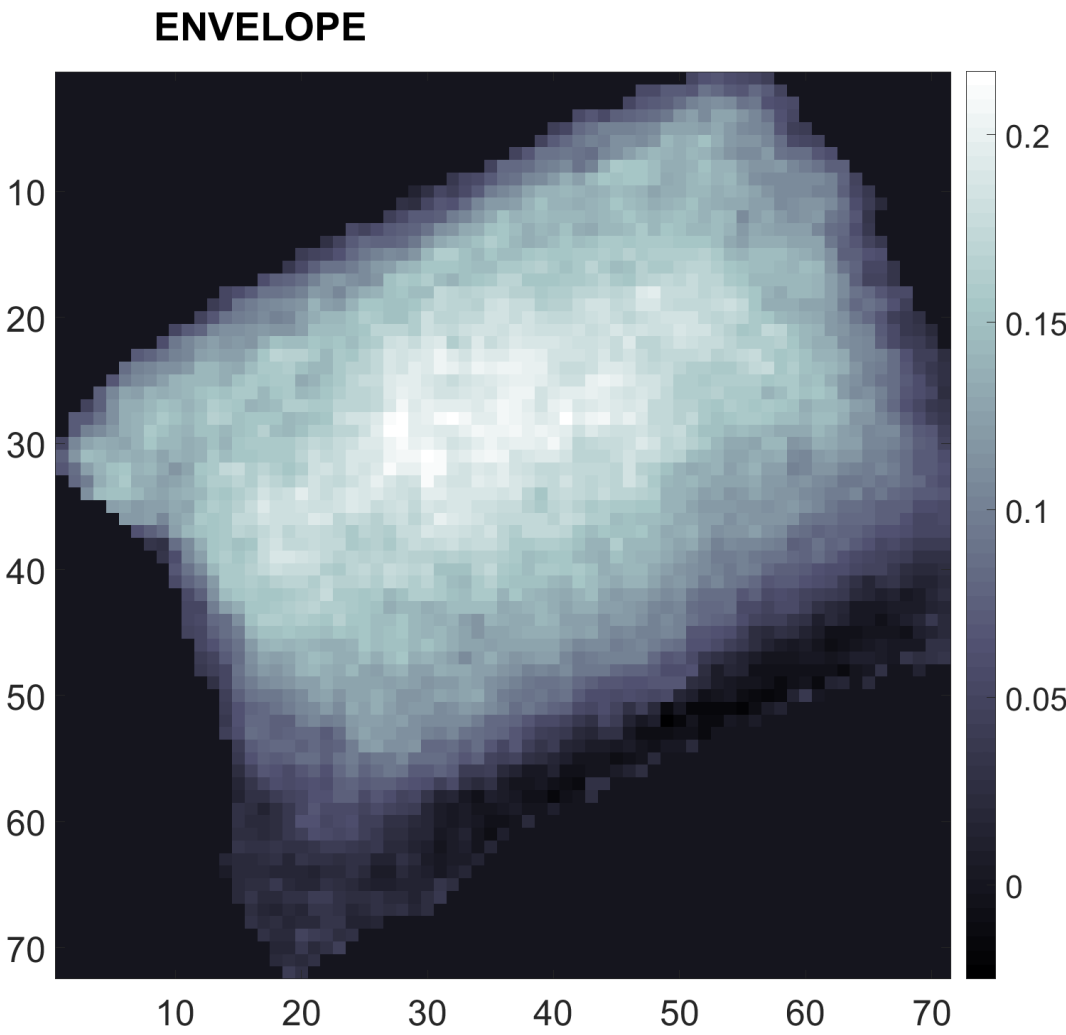
Stay Tuned….

On Wednesday: your first IC challenge…
global transformation and data-driven classification

The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

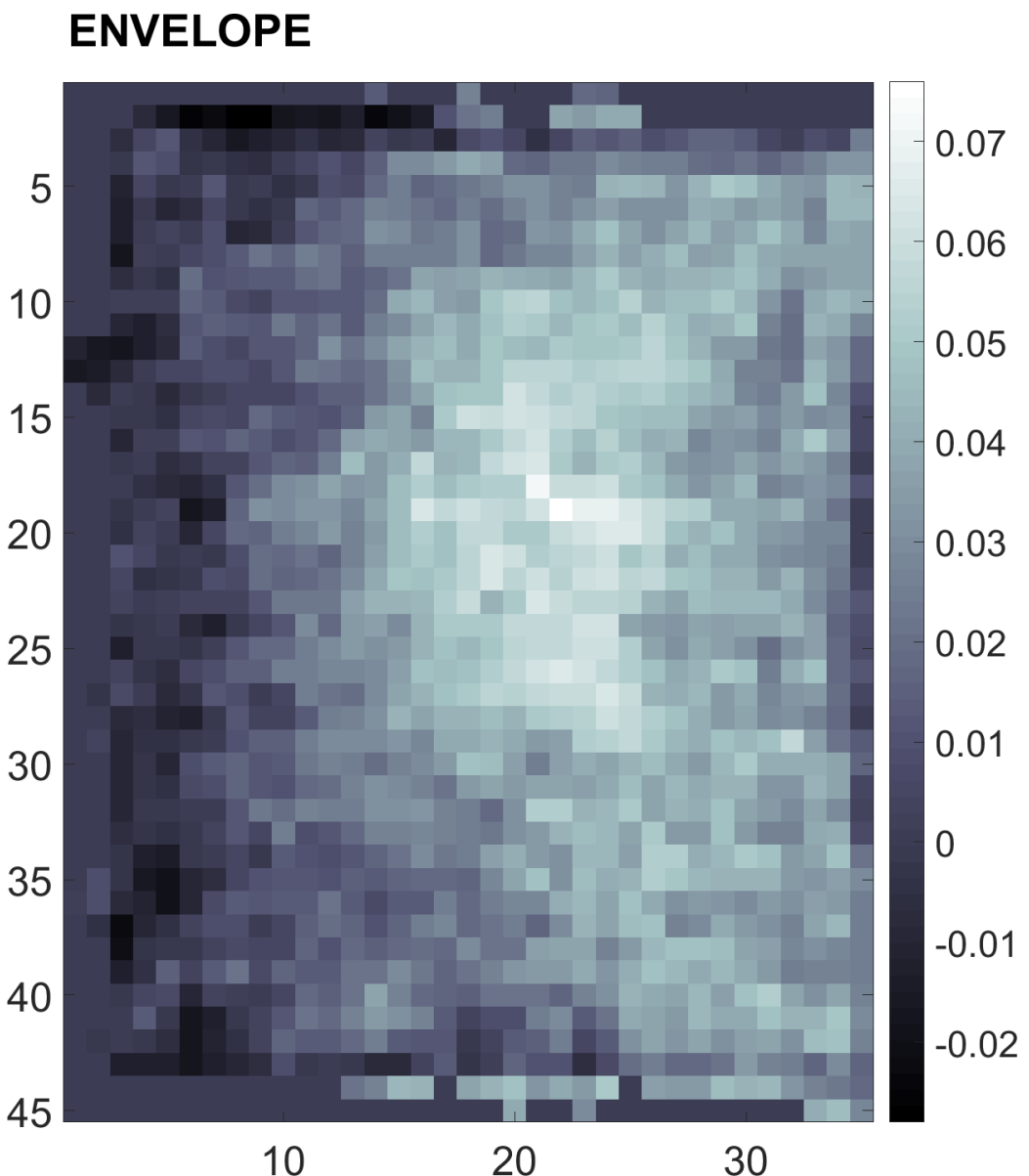
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

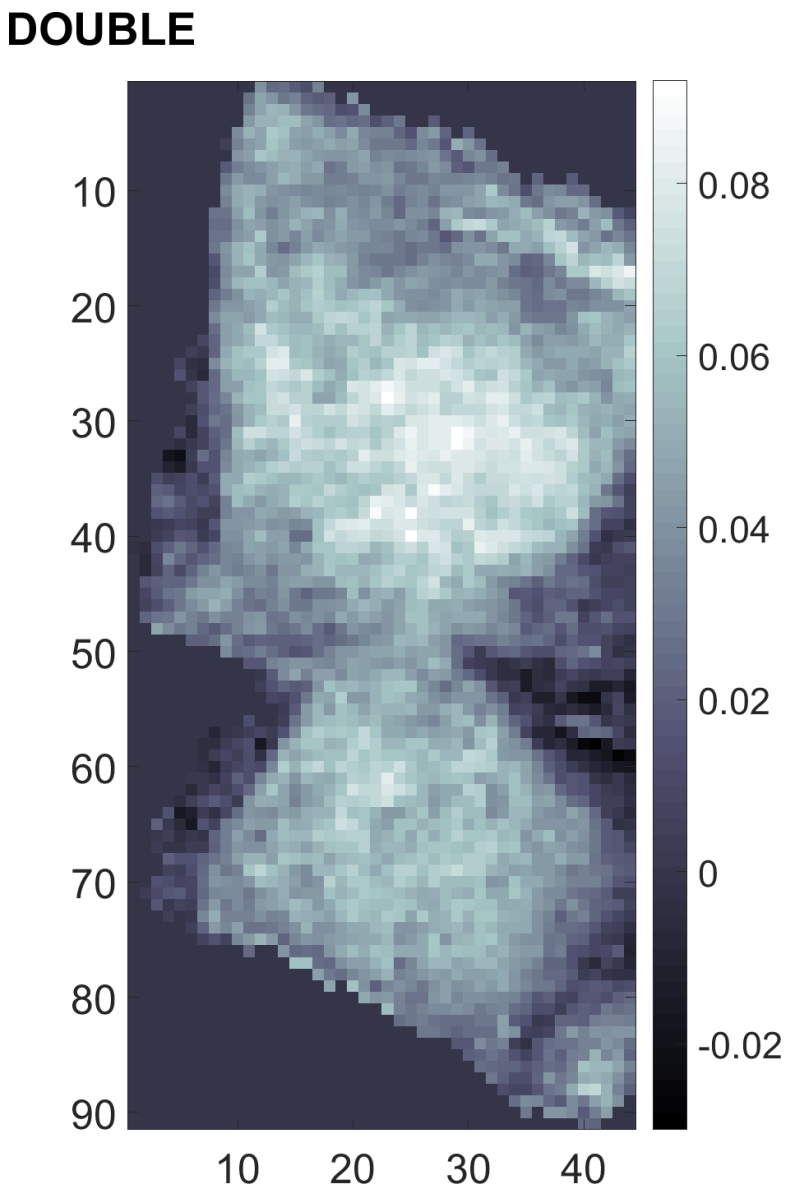
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

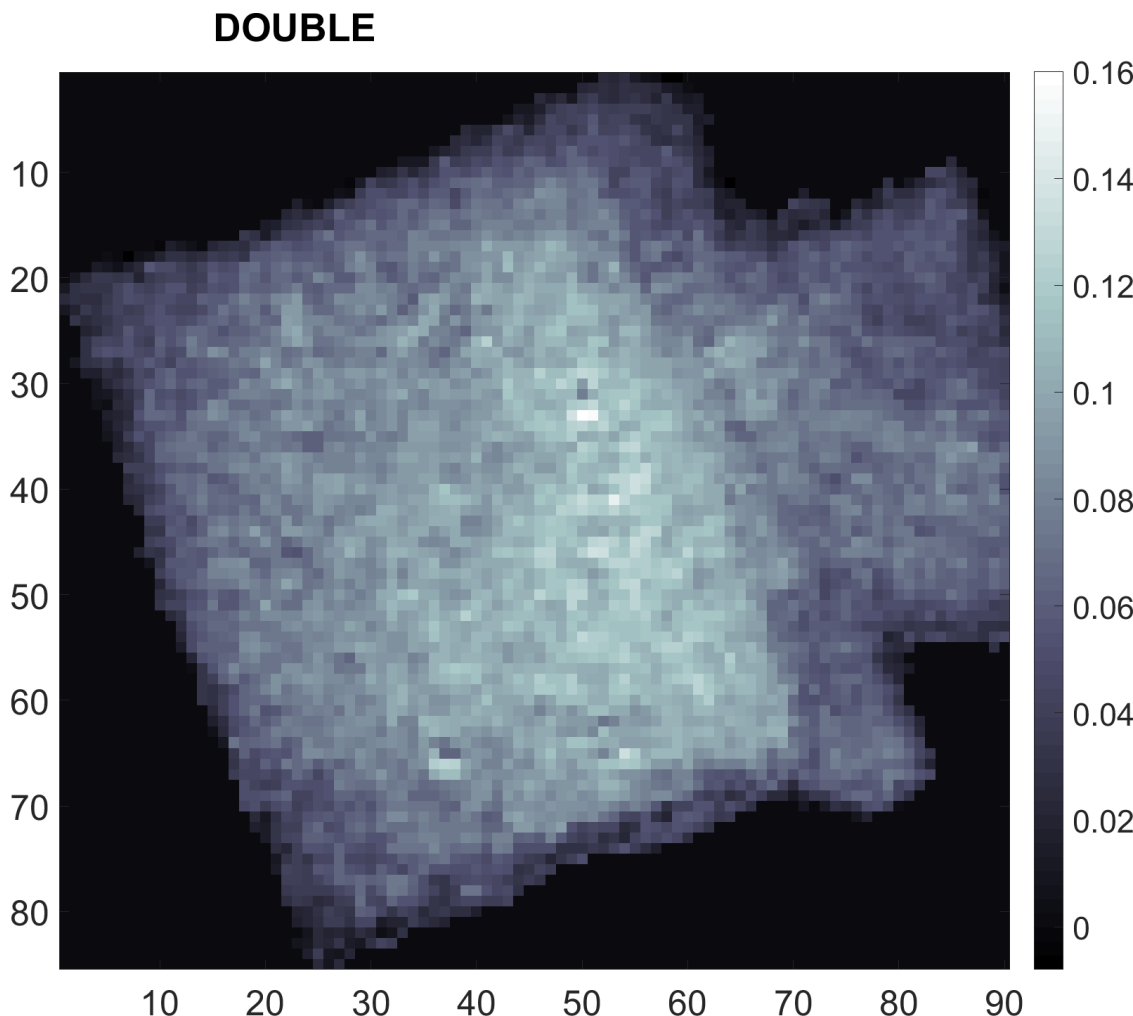
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

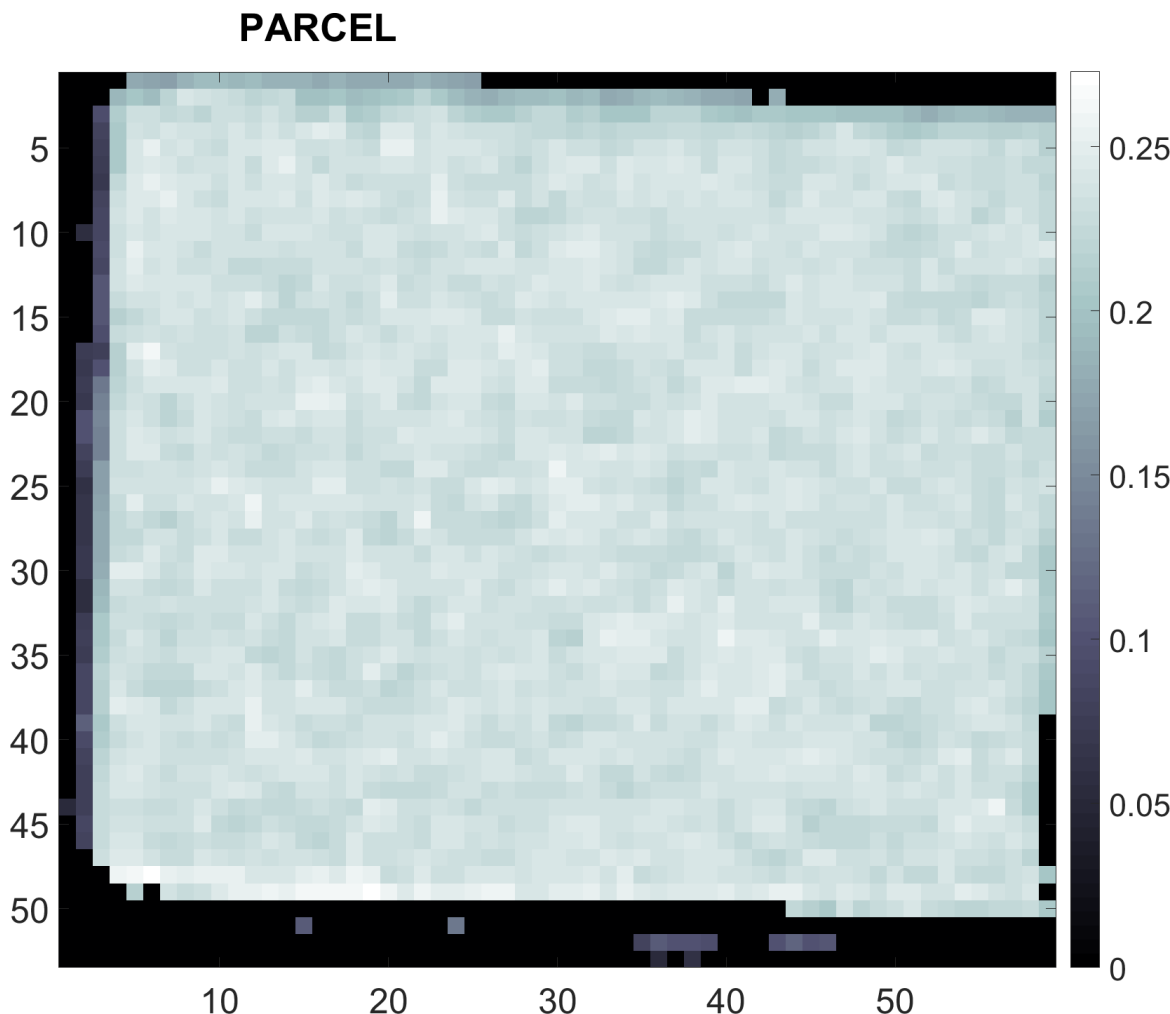
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE



The Application Scenario: Parcel Classification

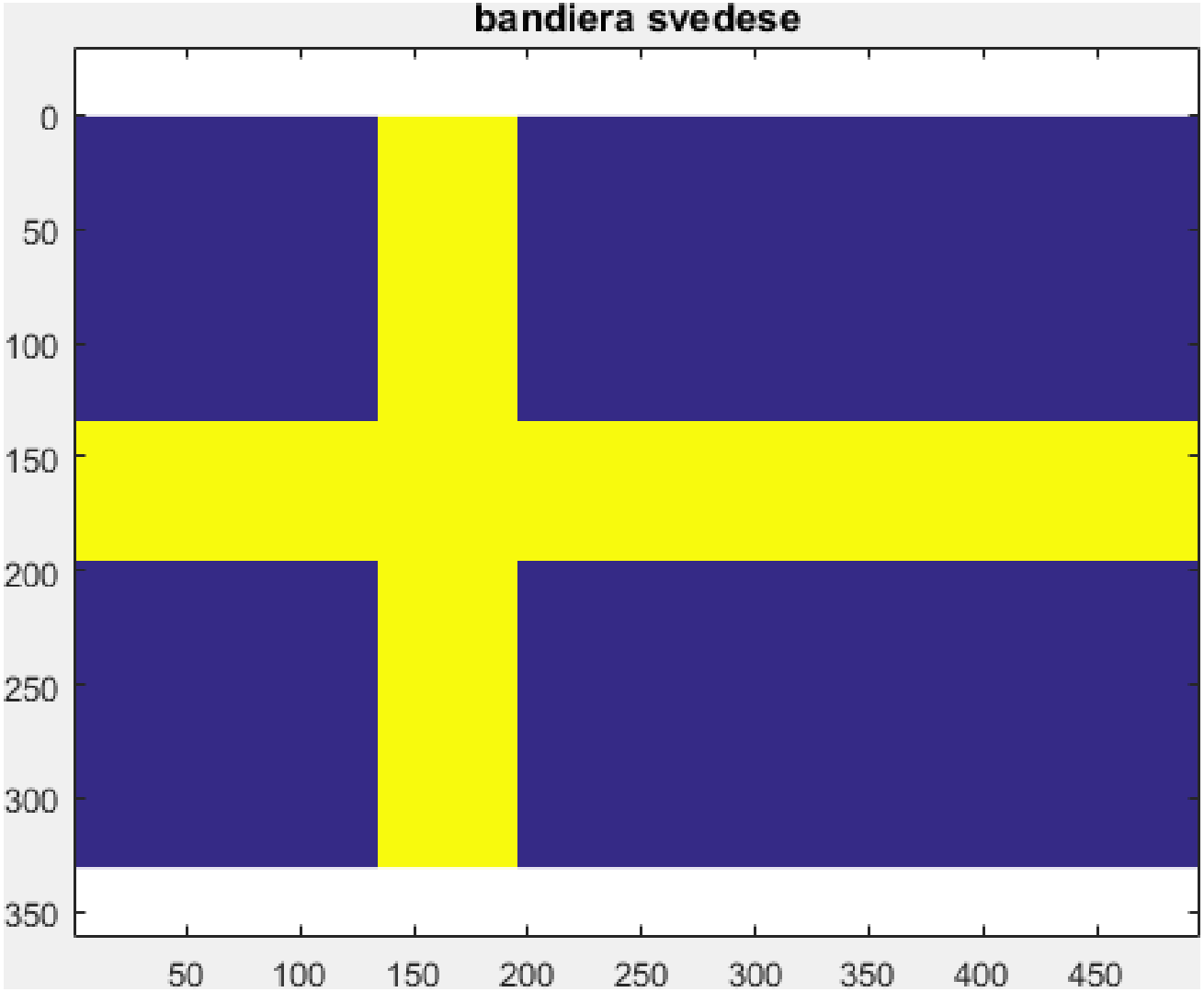
Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
  - ENVELOPE
  - PARCEL
  - DOUBLE

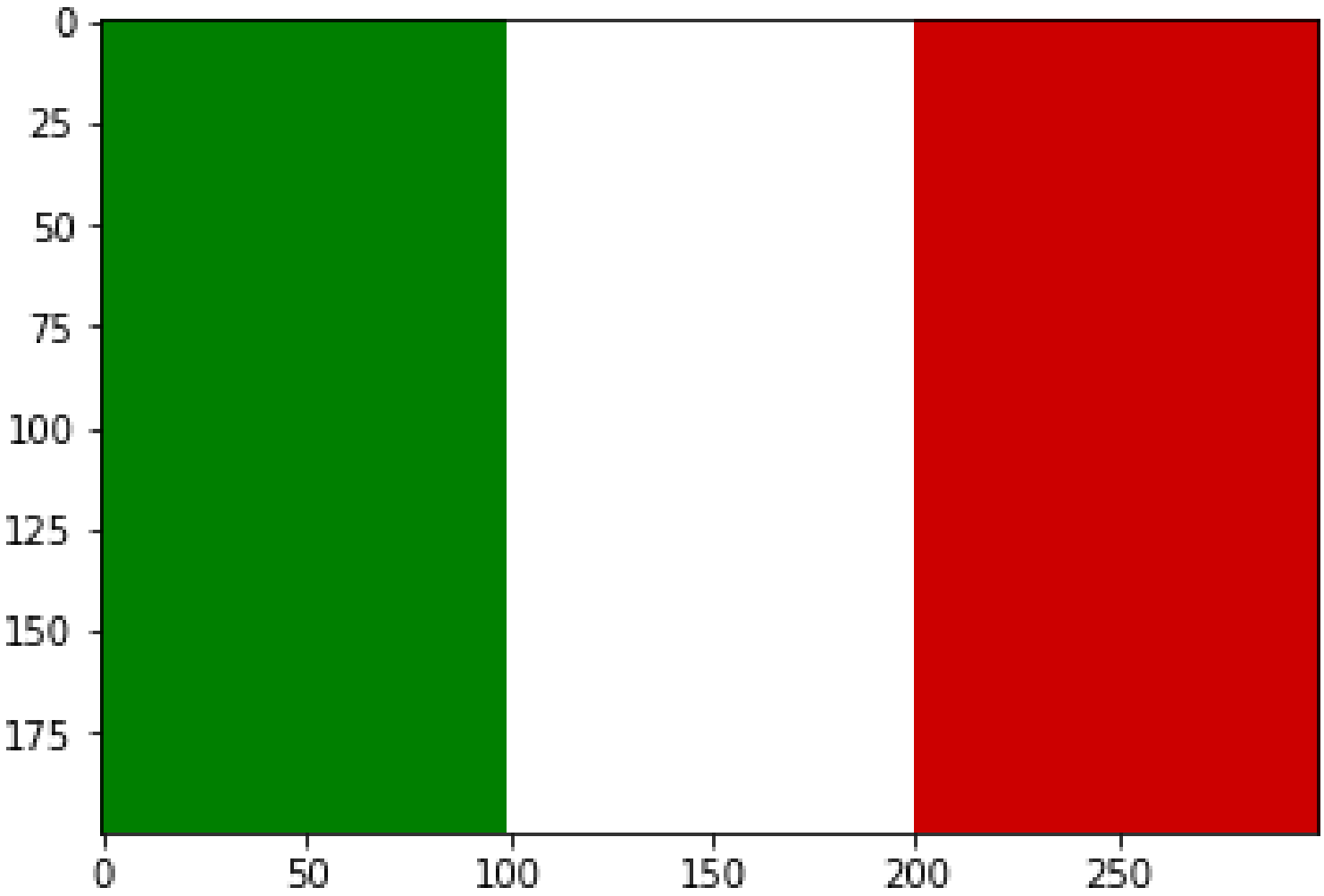


A Very Light Introduction to Python

Flags to be created as images in Python



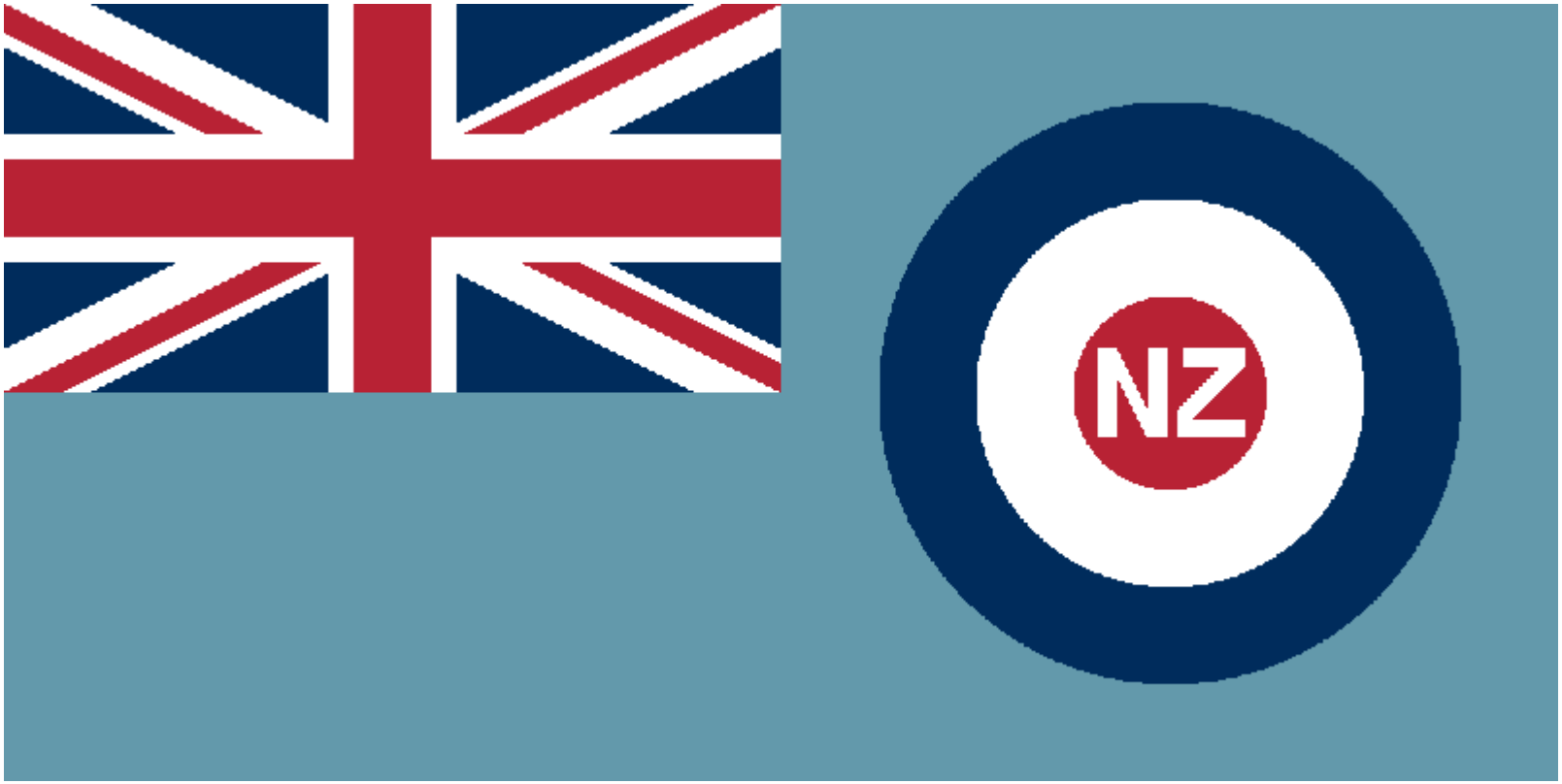
Flags to be created as images in Python



Flags created by Informatica B student (1st year, bachelor)

Bandiera della Catalogna

Flags created by Informatica B student (1st year, bachelor)



New Zealand Airforce

Do this at home…

Draw your best flag in Py and submit your code here:

http://bit.ly/numpy-flags

You can implement K-NN classifiers on cifar-10, see:

- http://cs231n.github.io/classification/

Additional resources

- http://www.scipy-lectures.org/packages/scikit-image/index.html (good Image Processing Reference in Py)
- http://cs231n.github.io/python-numpy-tutorial/