Spatial filtering of images

Paweł Forczmański
Chair of Multimedia Systems, Faculty of Computer Science and Information Technology

Agenda

1. convolution theory
2. practical implementation
3. filtering examples

DIP scheme

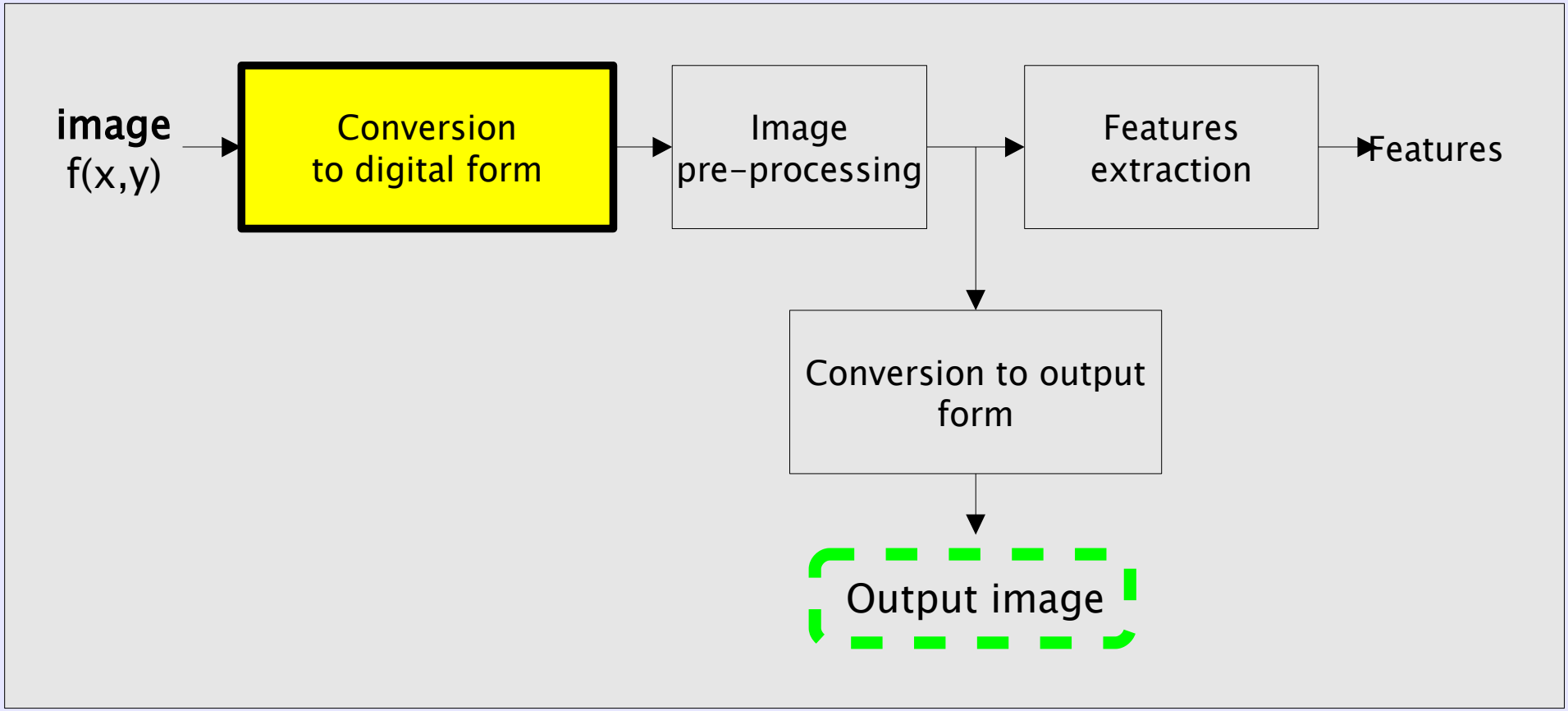
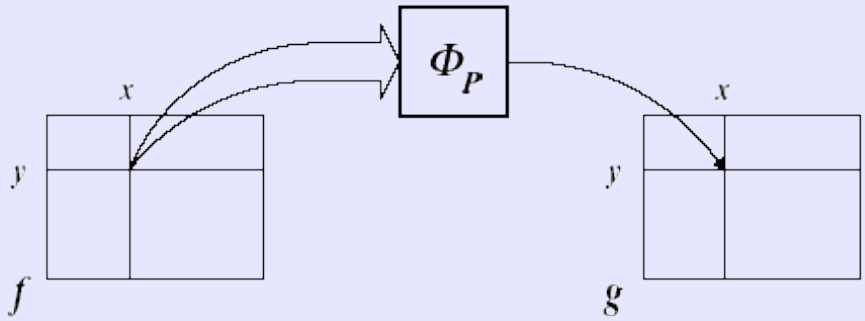


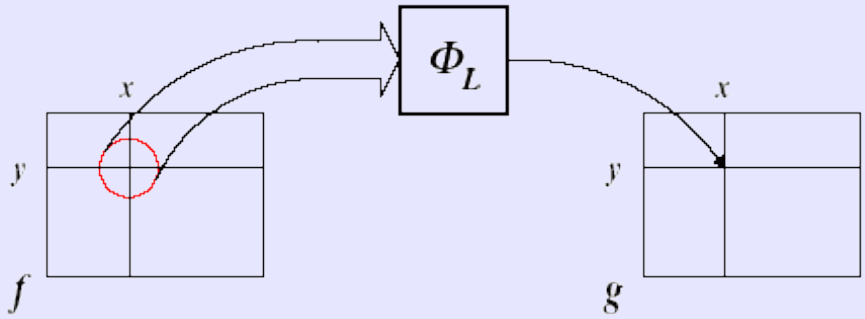
image
f(x,y)
Conversion to digital form
Image pre-processing
Features extraction
Features
Conversion to output form
Output image

Transformations

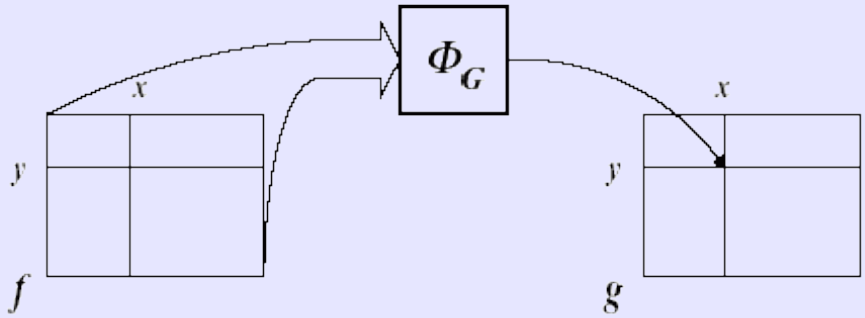
point transform



local transform



global transform



Definition

Convolution is a mathematical operation on two functions f and g, producing a third function that is typically viewed as a modified version of one of the original functions, giving the area overlap between the two functions as a function of the amount that one of the original functions is translated:

$$f * g = \int_{-\infty}^{\infty} f(x-t)\, g(t)\, dt$$

definition (2)

Convolution is denoted with *f * g.*

Similar to multiplication and has the following features:

Commutativity $f * g = g * f$,

Associativity $f * (g * h) = (f * g) * h$,

Distributivity $f * (g + h) = (f * g) + (f * h)$,

Associativity with scalar multiplication

$a\,(f * g) = (a\,f) * g$,

for any real (or complex) number *a*

Digital processing

In practice, *f* and *g* are **vectors** or **matrices** with discrete values, and integral operator is changed into **sum**.

$$h[x]=\sum_{t=t_1}^{t=t_n} f[x-t]\,g[t]$$

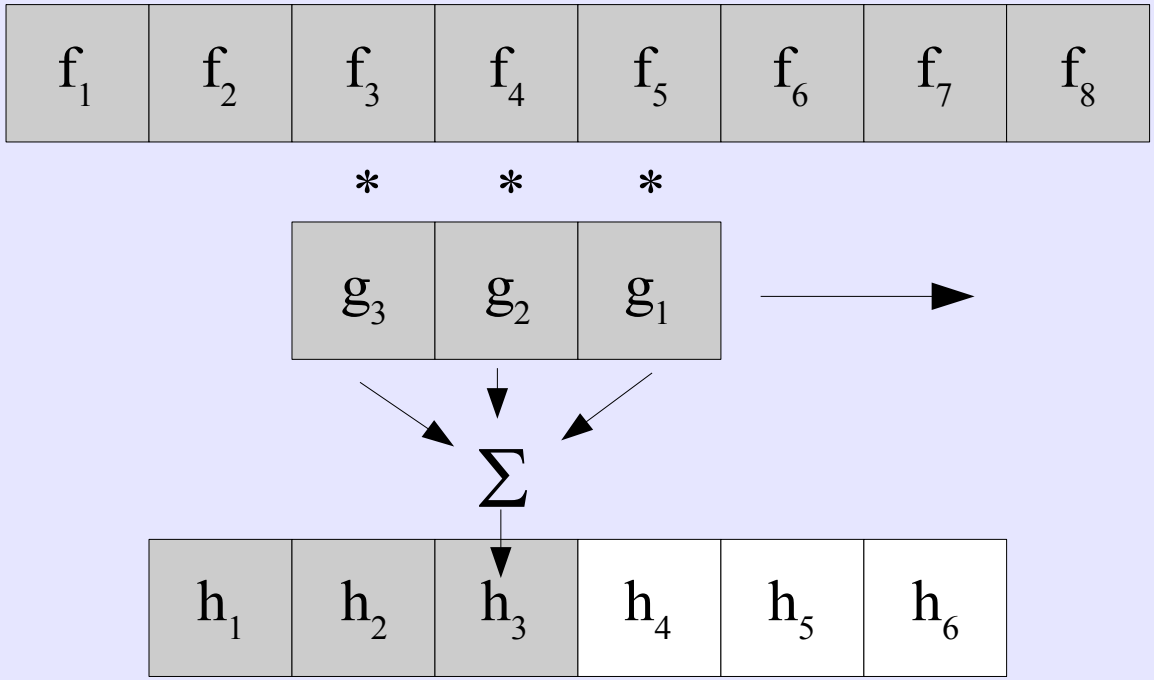
It is impossible to perform integral/sum operations from -infinity to +infinity, so we change it into one of the following convolution types:

Linear convolution (aperiodic signals)

Circular/cyclic convolution (periodic signals)

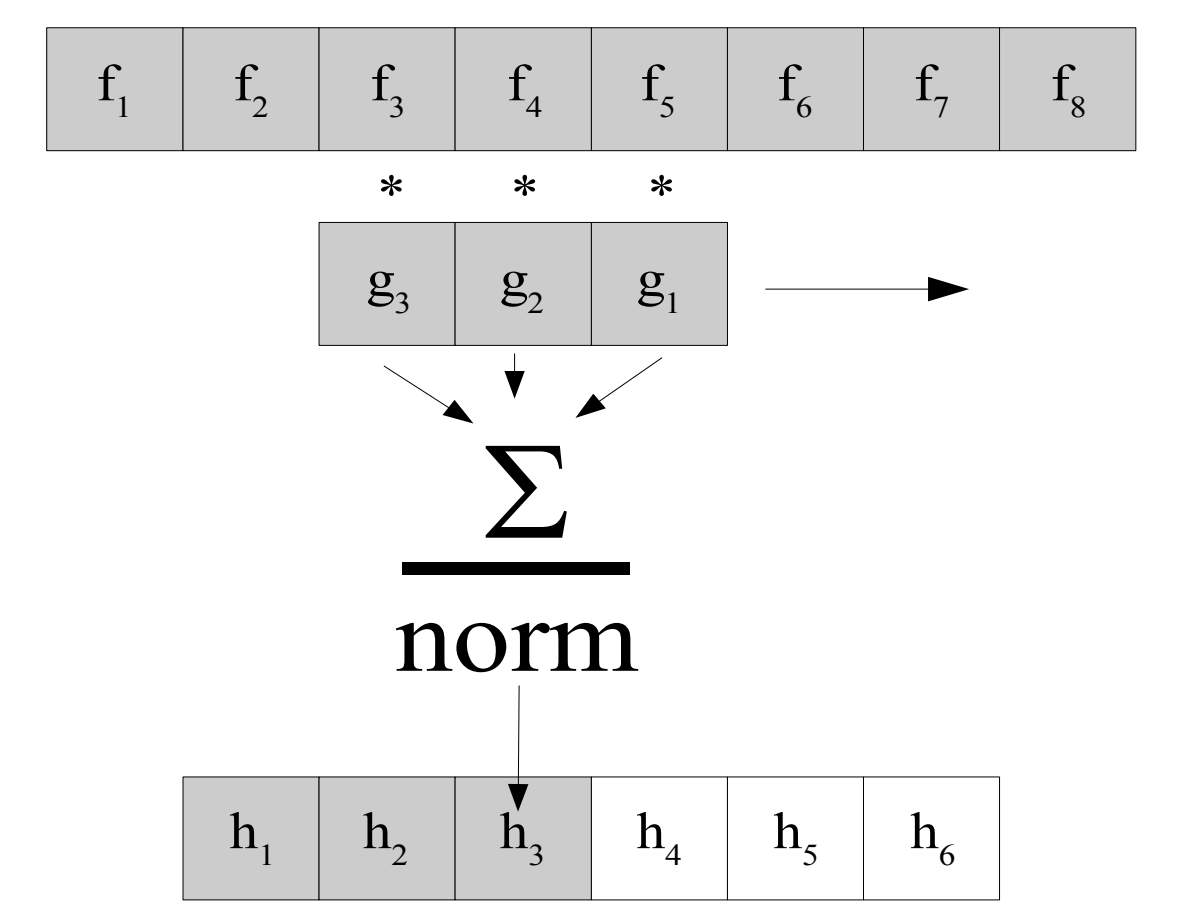
Convolution on vectors

Incase when *f* and *g* have dissferent number of elements, shorter vector is a **mask** with pre-defined values.



normalization

An important stage is **normalization**, which is performed after uadding all multiplied elements.



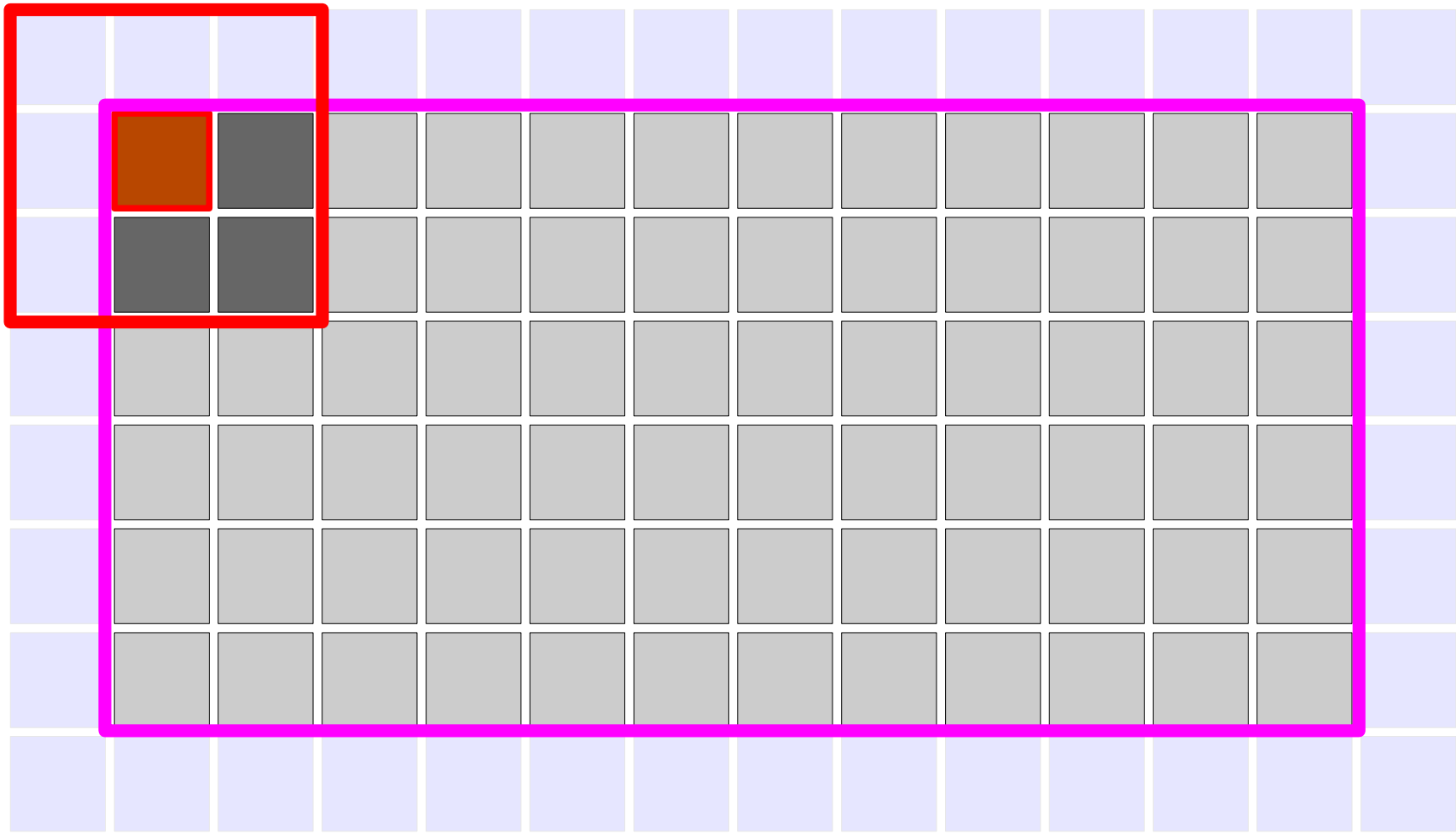
applications

In digital image processing convolutional filtering plays an important role in:

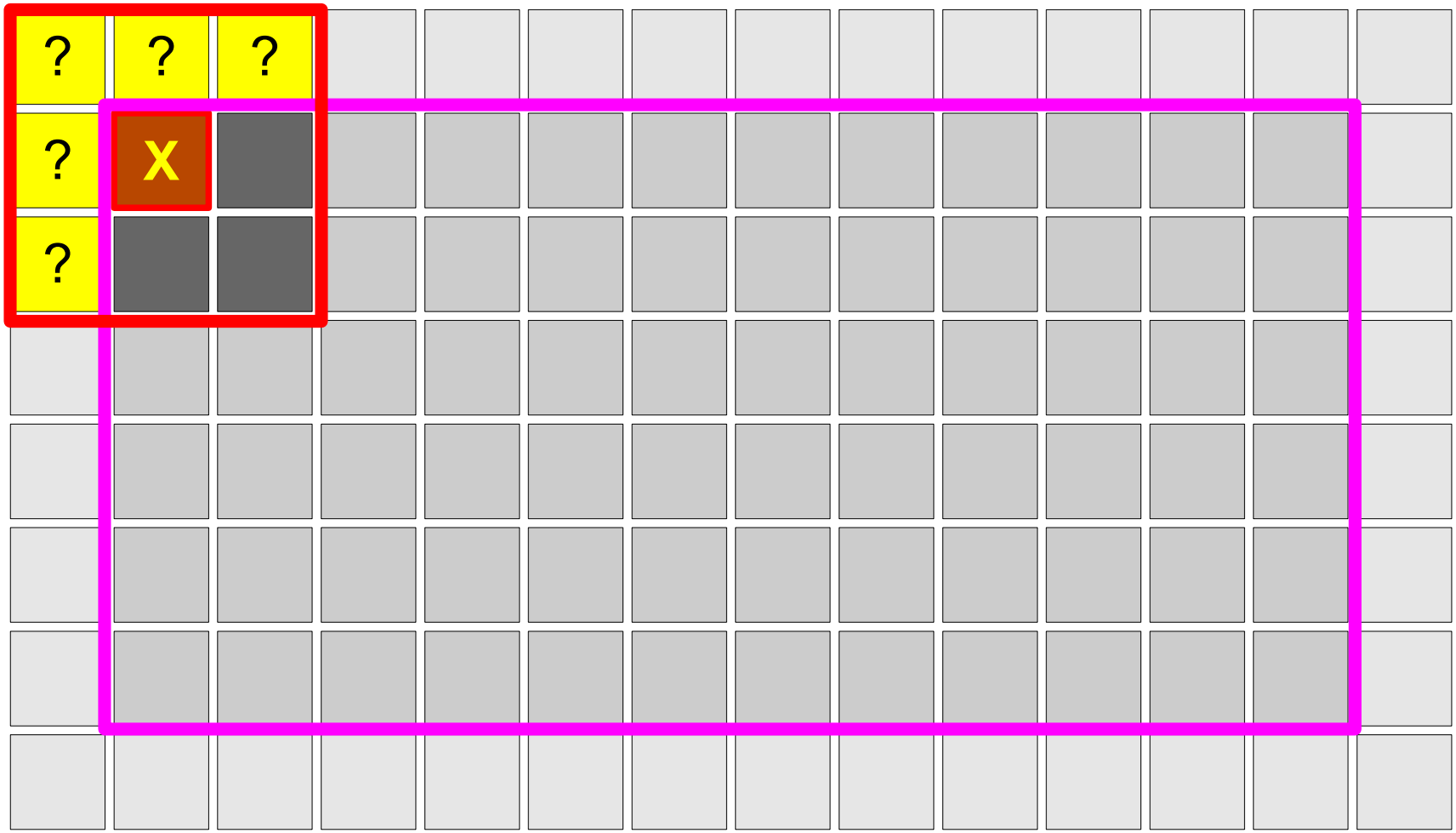
- edge detection and related processes;
- Sharpening;
- Blurring;
- Special effects (motion blur)
- Etc...

Traditional computing (sequential programming);
Parallel computing (mult processors/cores, GPU: „stencil computing”).

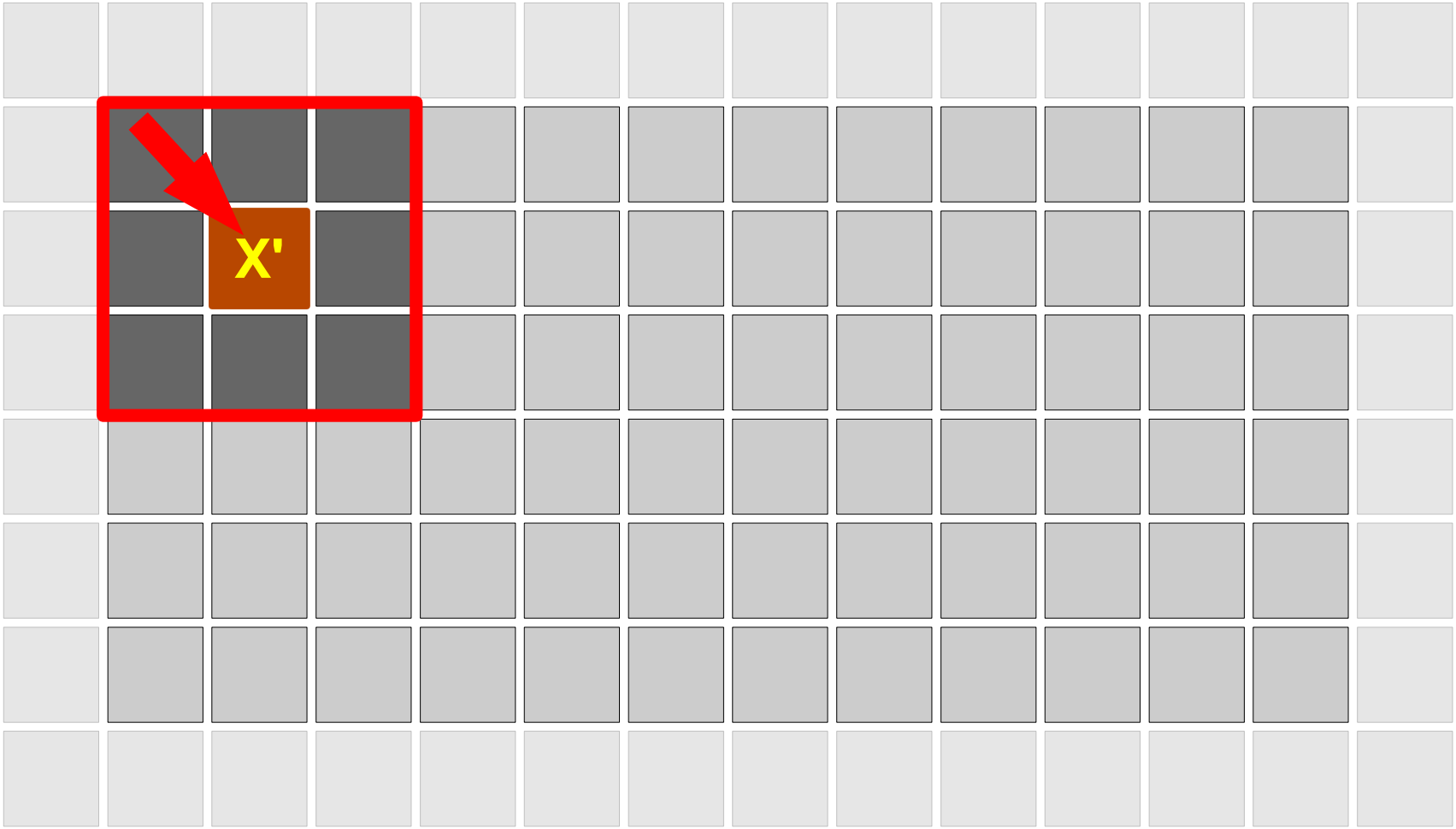
Problem with borders



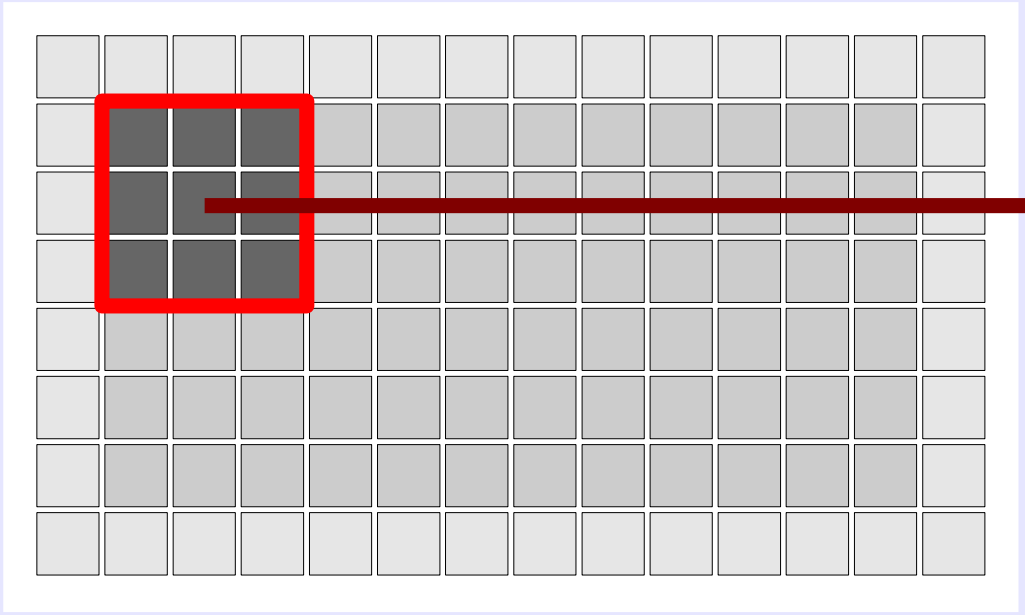
Problem with borders



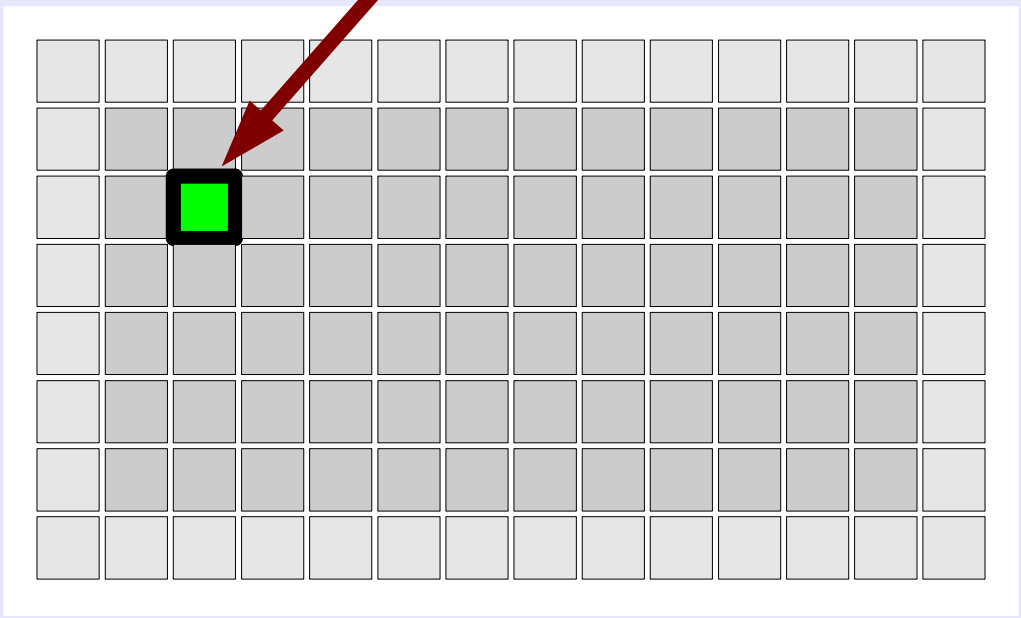
Problem with borders : move towards the center



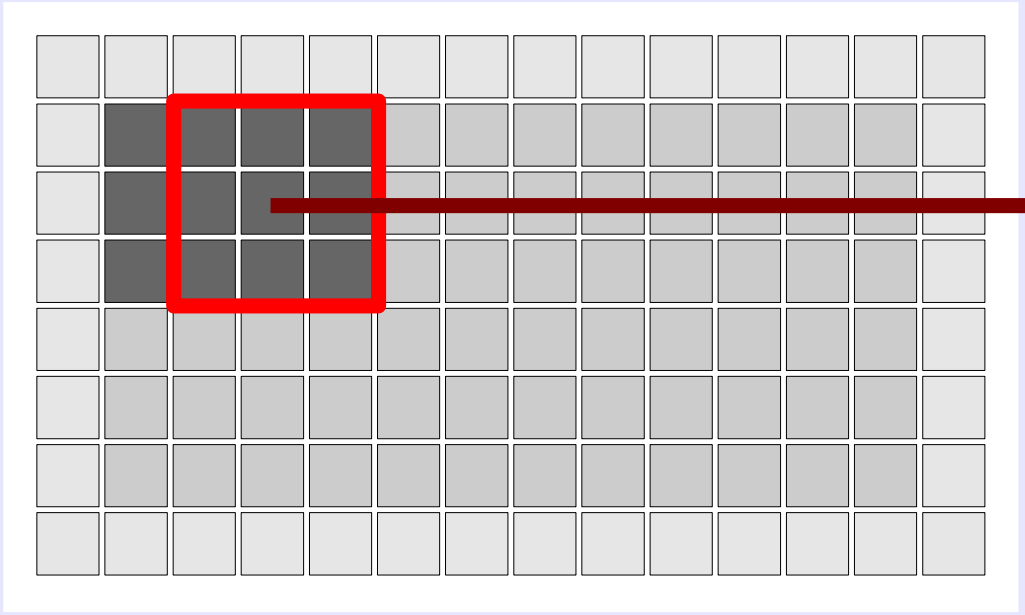
algorithm



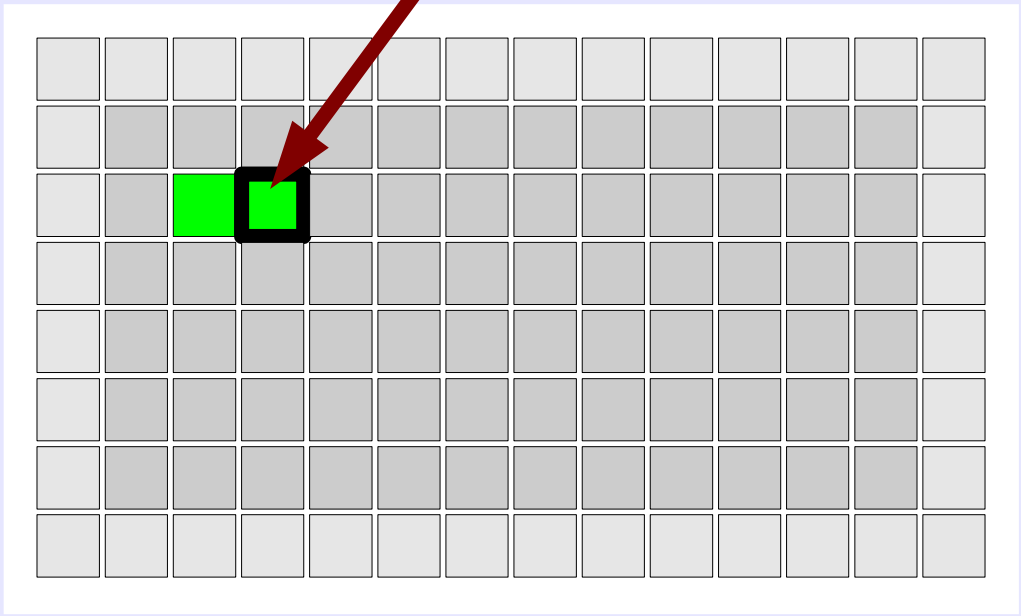
$$\frac{\sum (\text{window} .* \text{mask})}{\text{norm}}$$



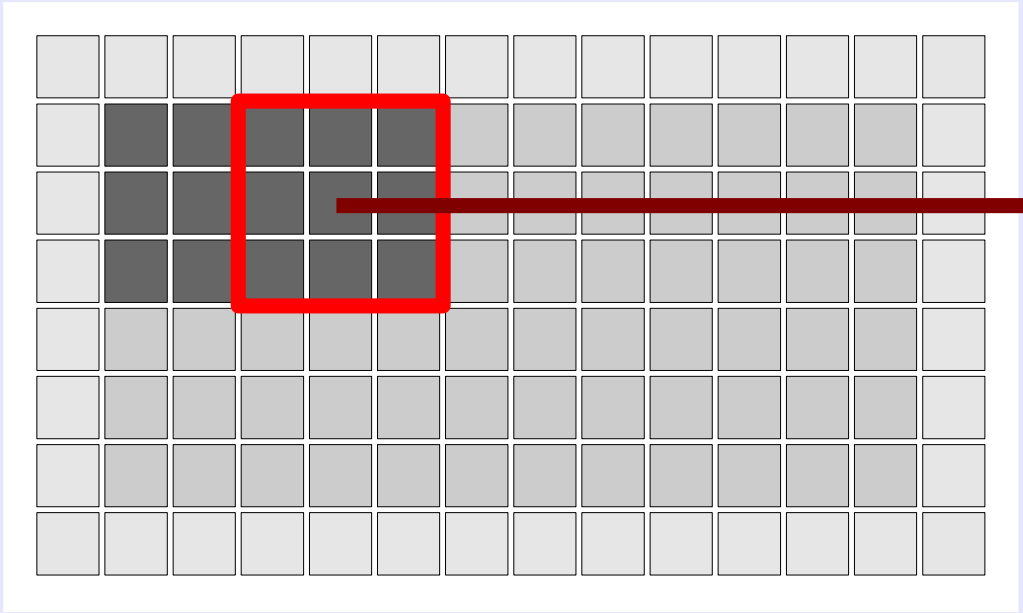
algorithm



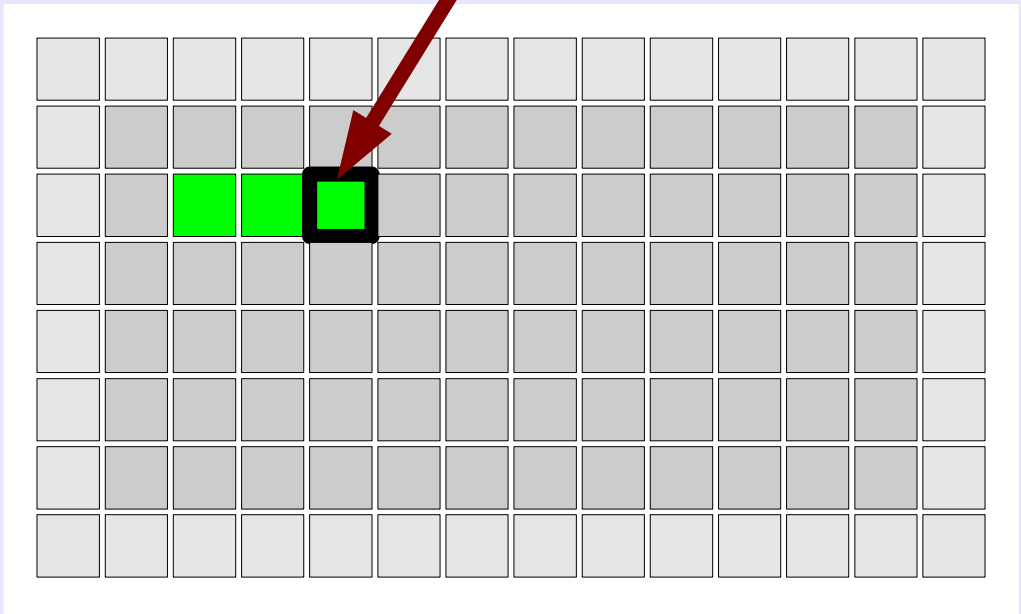
$$\frac{\sum (\text{window} .* \text{mask})}{\text{norm}}$$



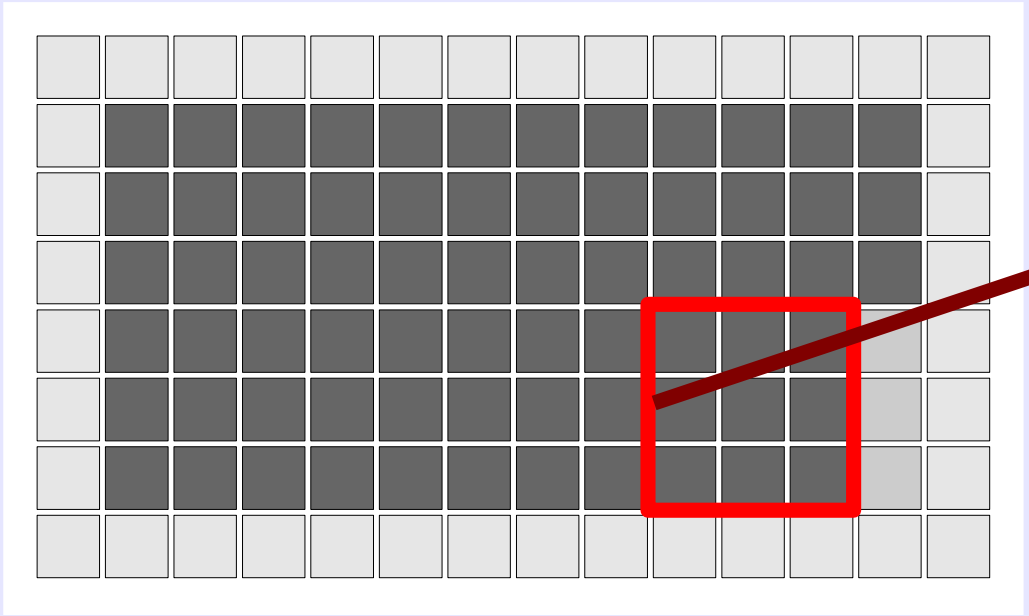
algorithm



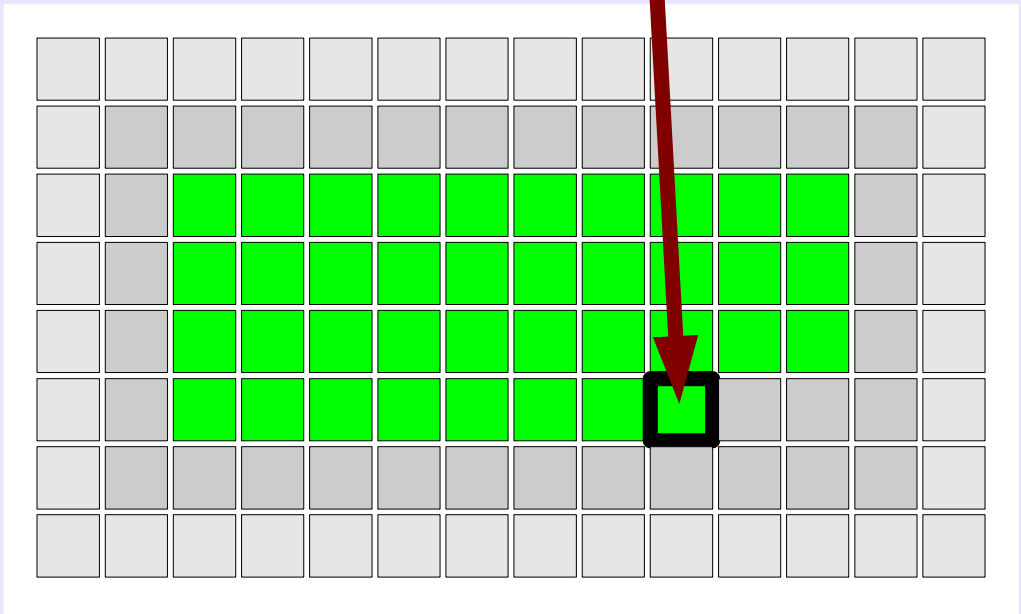
$$\frac{\sum (\text{window} .* \text{mask})}{\text{norm}}$$



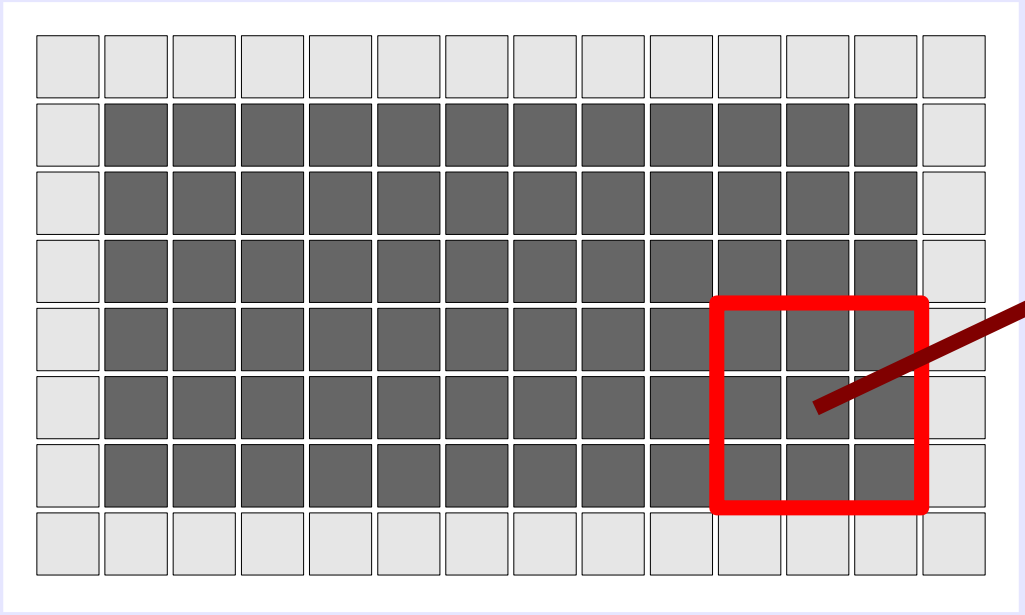
algorithm



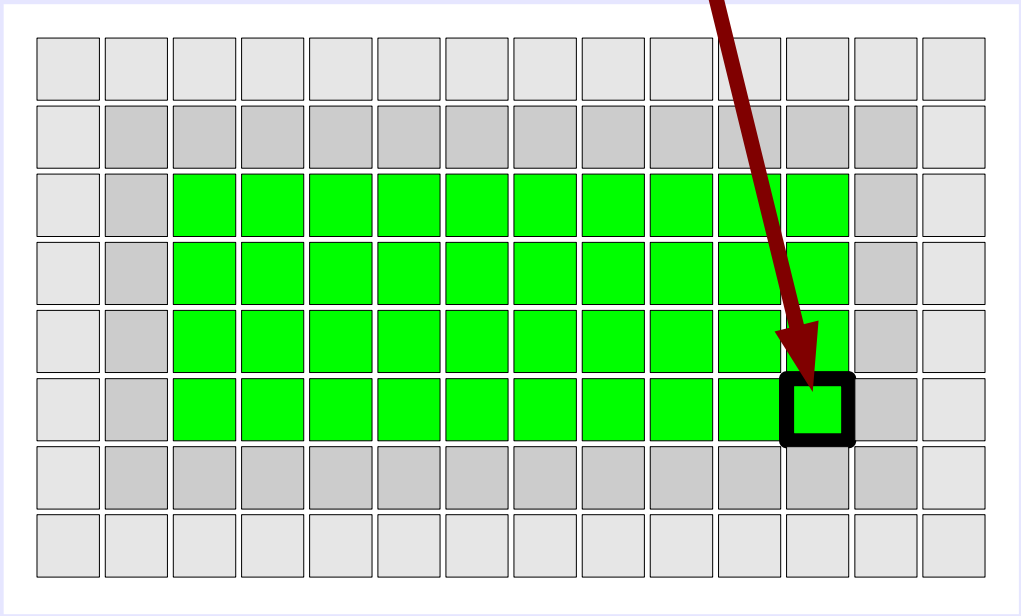
$$\frac{\sum (\text{window} .* \text{mask})}{\text{norm}}$$



algorithm

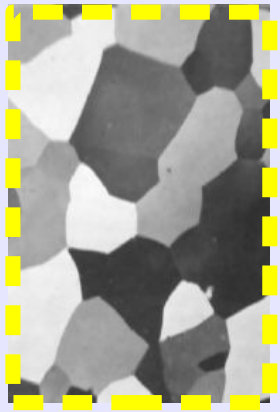


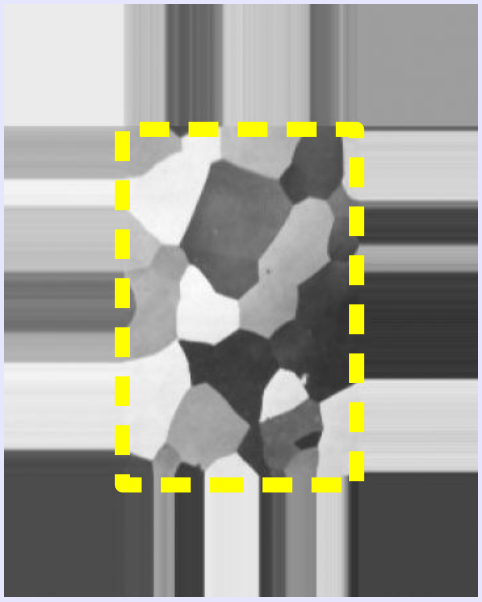
$$\frac{\sum (\text{window} .* \text{mask})}{\text{norm}}$$



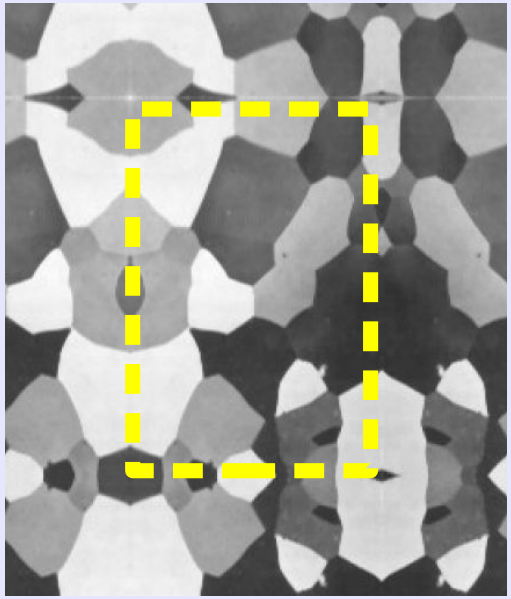
Extrapolation of borders

Input image

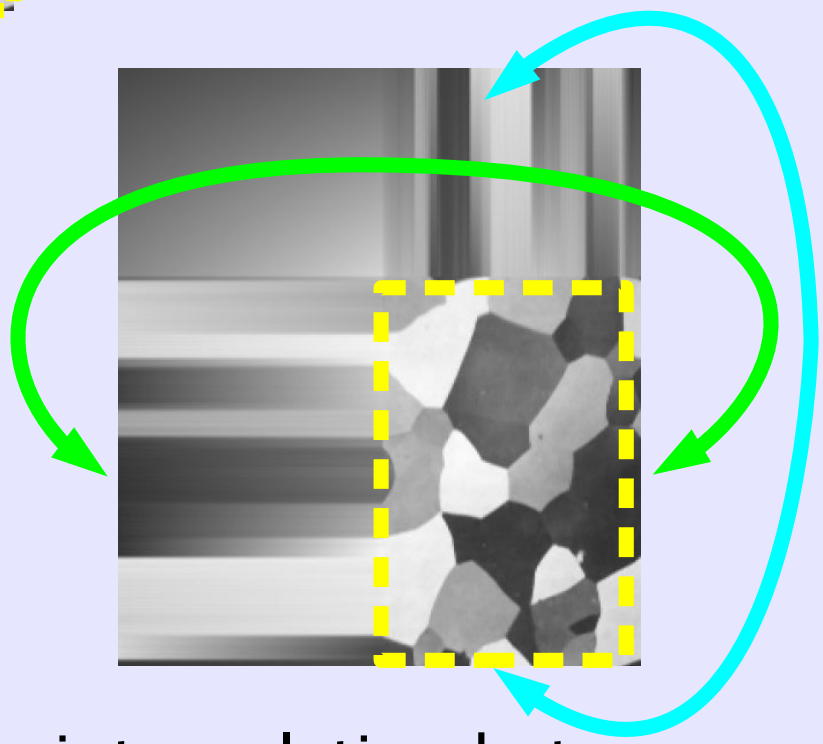




Pixel copying

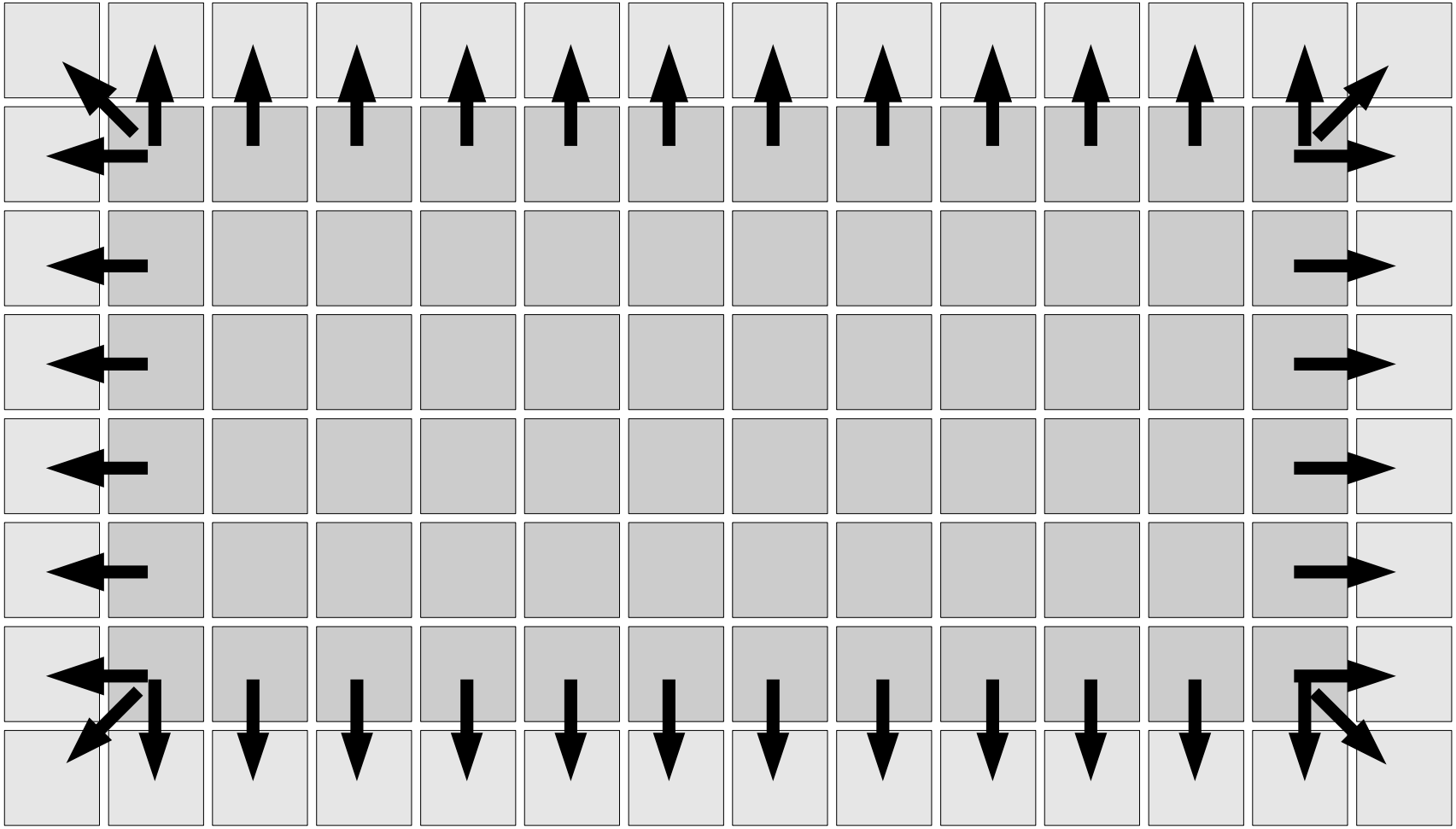


mirroring

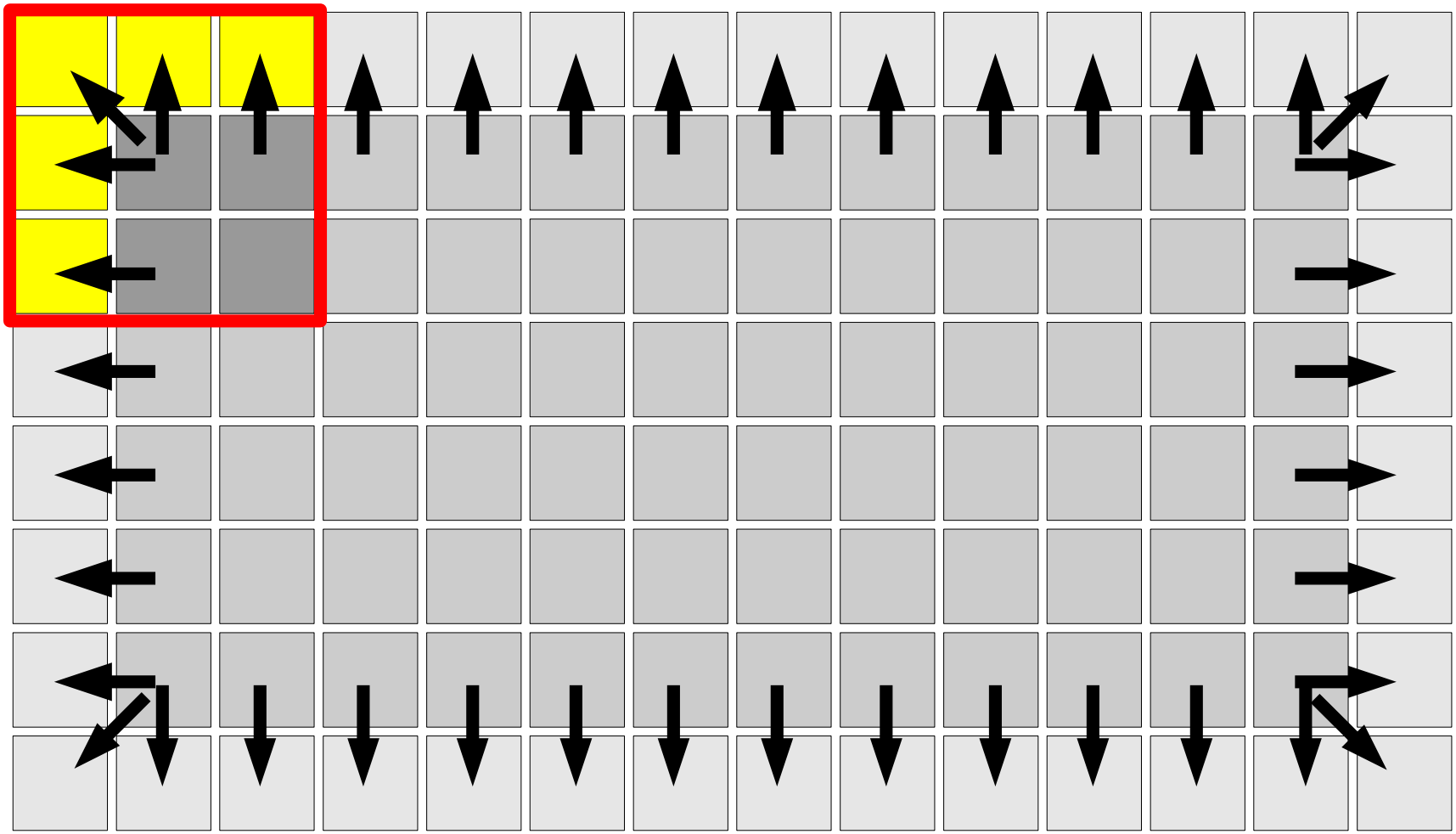


interpolation betwe-
en borders

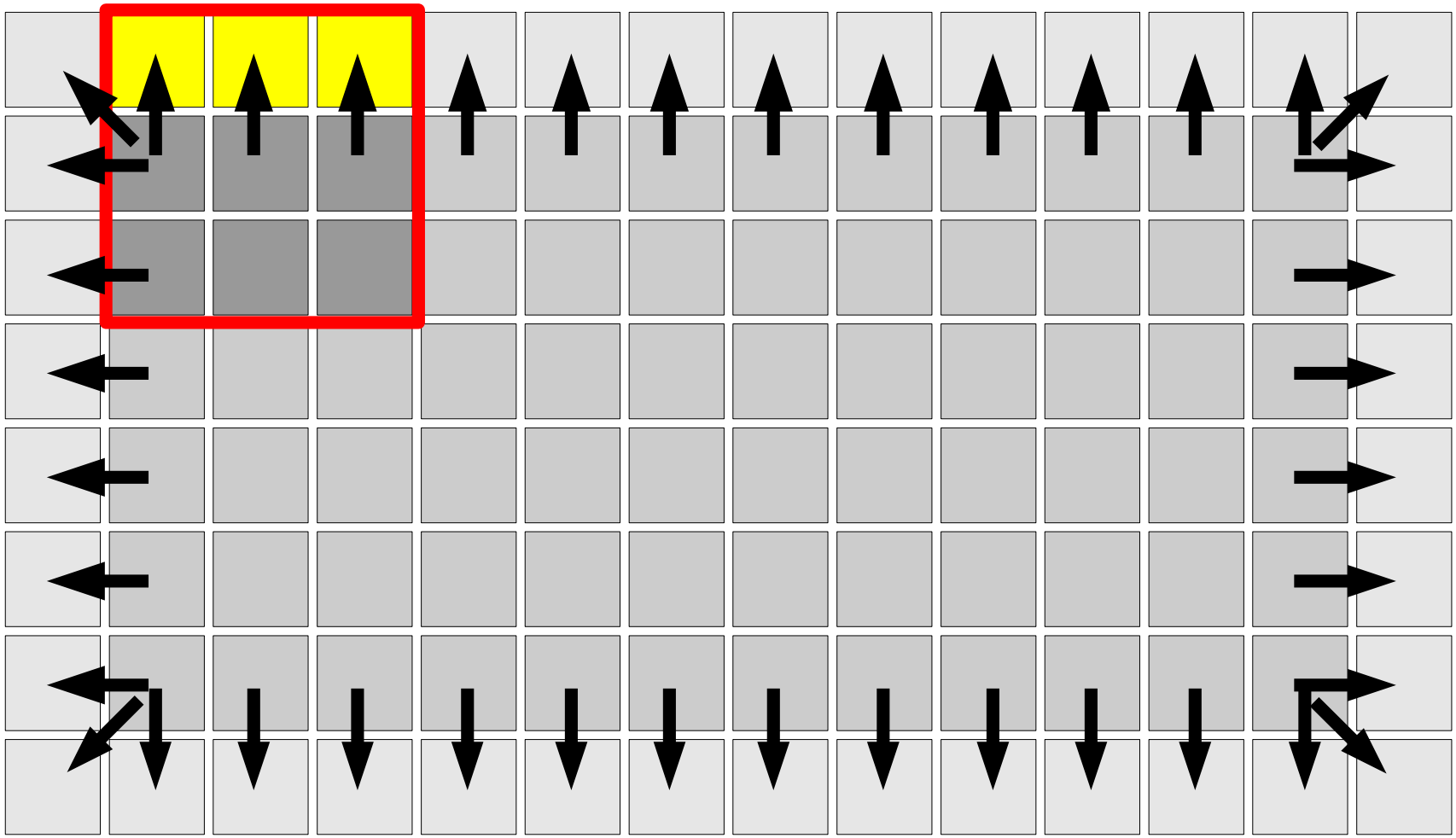
Extrapolation



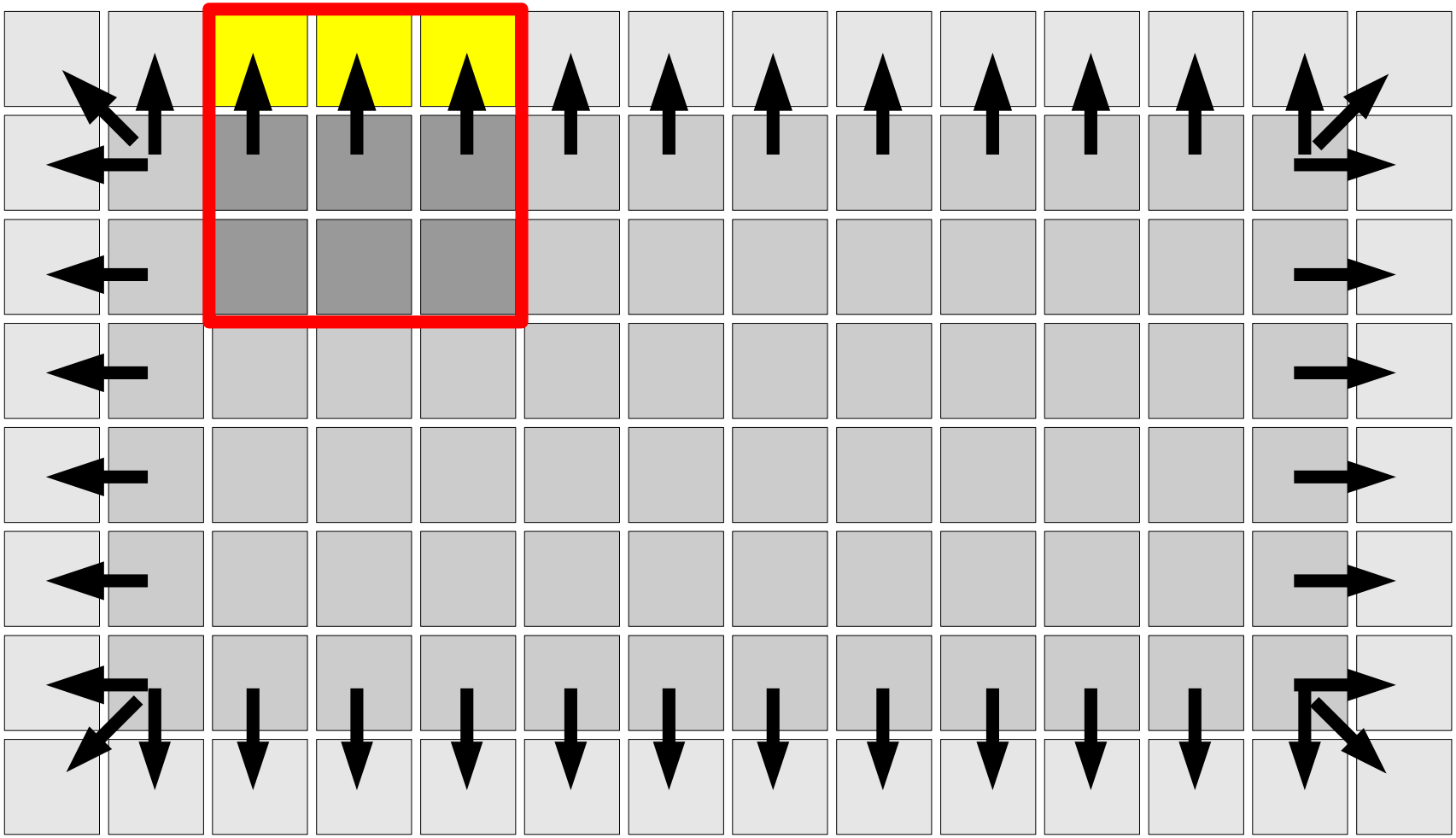
Extrapolation



Extrapolation

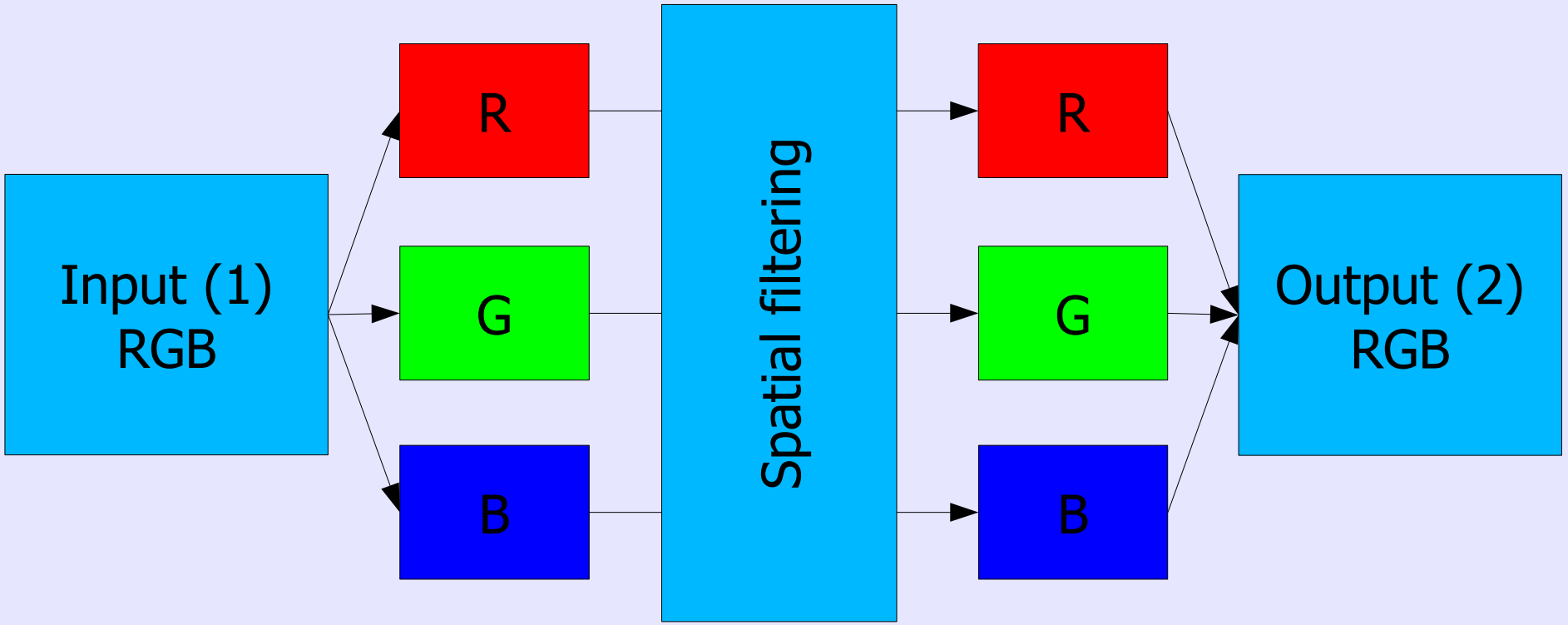


extrapolation



Filtering in color space

Multi-color images (i.e. RGB, YUV, HSV) are filtered in such manner that each channel is filtered independently;



Averaging/blurring filtering

Spatial filtering can be used for blurring an image by means of:

- Averaging filter
- Gaussian blur
- etc...

Averaging filter (1)

1	1	1
1	**1**	1
1	1	1

Norm=9

1	1	1
1	**2**	1
1	1	1

Norm=10

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

Norm=25

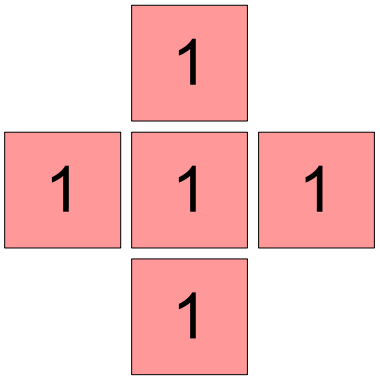
1	1	1
1	**3**	1
1	1	1

Norm=11

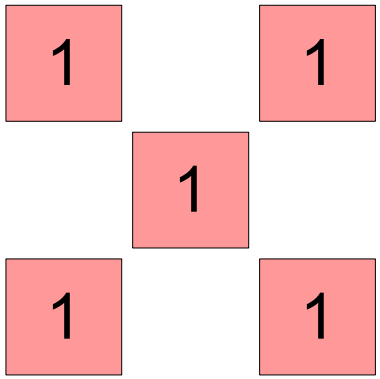
0	0	0
1	1	1
0	0	0

Norm=3

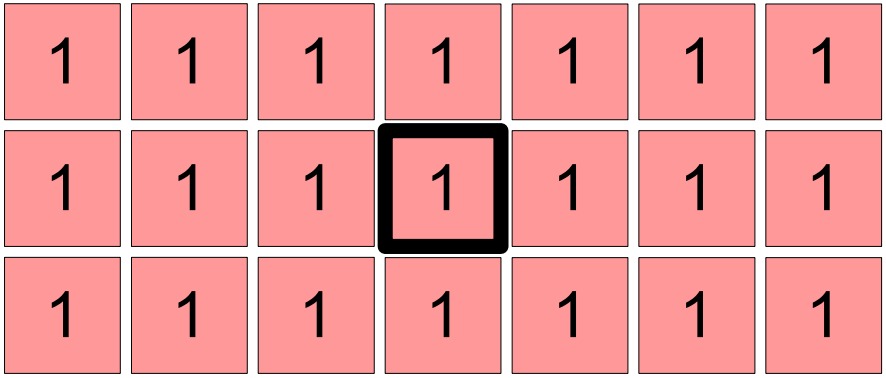
Averaging filter (2)



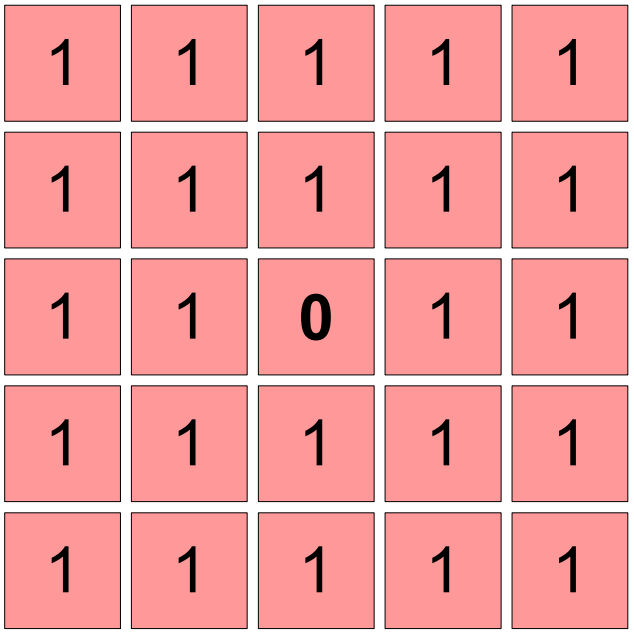
Norm=5



Norm=5



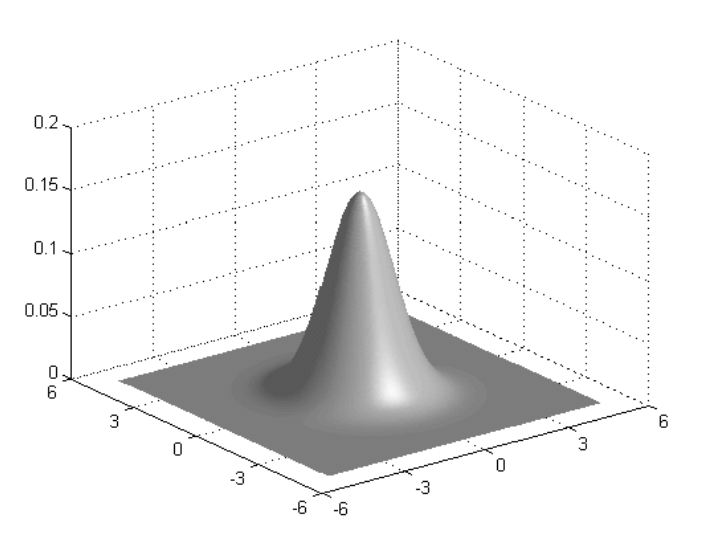
Norm=21



Norm=24

Gauss filtering

An image f is filtered with a mask g_σ which is a discrete approximation of two-dimensional Gauss function:



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

0.0571	0.1238	0.0571
0.1238	0.2042	0.1238
0.0571	0.1238	0.0571

Mask elements depend on σ which is a standard deviation. It decides about blurring effect (high values of σ give strong blurring)

Sharpening

Spatial filtering using convolution can be applied in order to detect edges or make images sharper:

- High-boost filtering;
- Unsharp masking
- High-pass filtering;
- Etc...

High boost

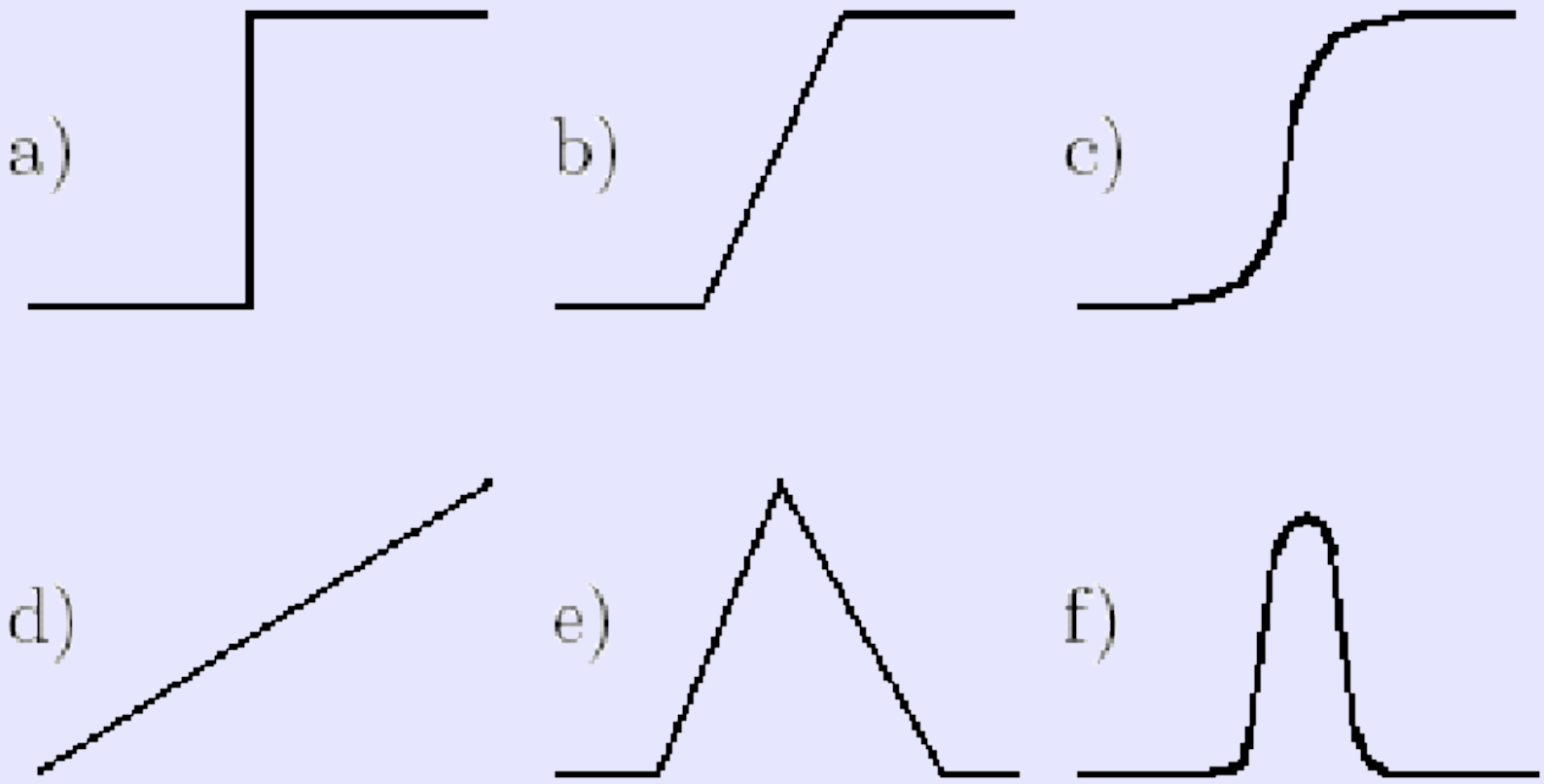
$$f(x,y) = f_L(x,y) + f_H(x,y)$$

$$f_{HB}(x,y) = Af(x,y) - f_L(x,y) =$$
$$= (A-1)f(x,y) + f(x,y) - f_L(x,y) =$$
$$= (A-1)f(x,y) + f_H(x,y), \qquad A \geq 1$$

$$h_{HB} = \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9A-1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge detection

The edge is a local change in image intensity and its vertical (or horizontal) projection can look like this:



Detection of points and lines

The most easy way is to calculate a correlation of image parts with an adequate detection mask.

Very often: 3x3 and 5x5 masks are used

Detection depends on the mask values

point detection

-1	-1	-1
-1	8	-1
-1	-1	-1

Line detection

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

Horizontal lines

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

Vertical lines

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

+45º

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

-45º

Edge detection

Edges can be detected using various gradient operators:

- First derivative of an image shows the edge and its direction
- Point of sign change of second derivative (*zero crossing*), can also be used to detect edges

The main problem is the false detection, which comes from the amplification of noise!

Detection

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

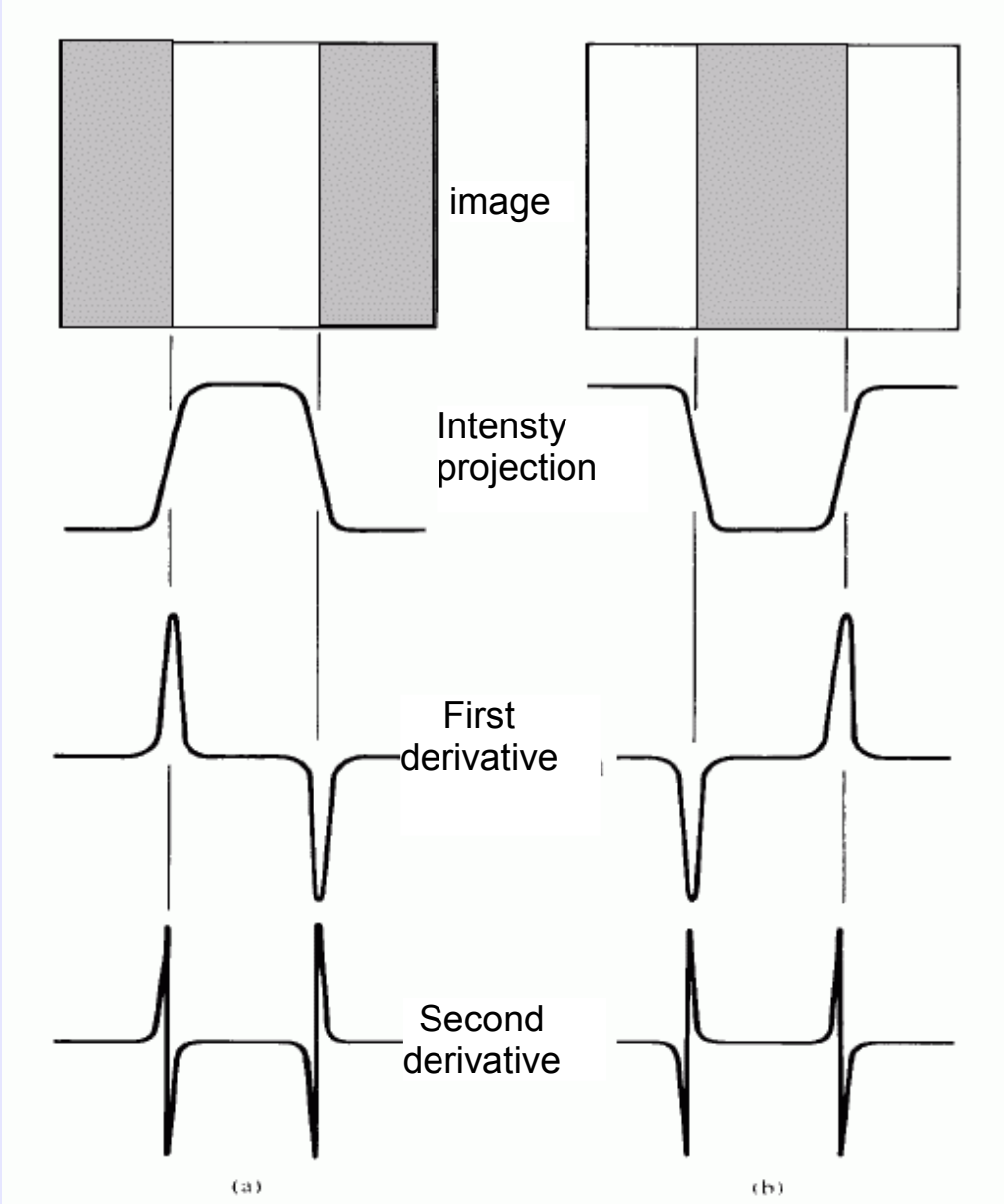


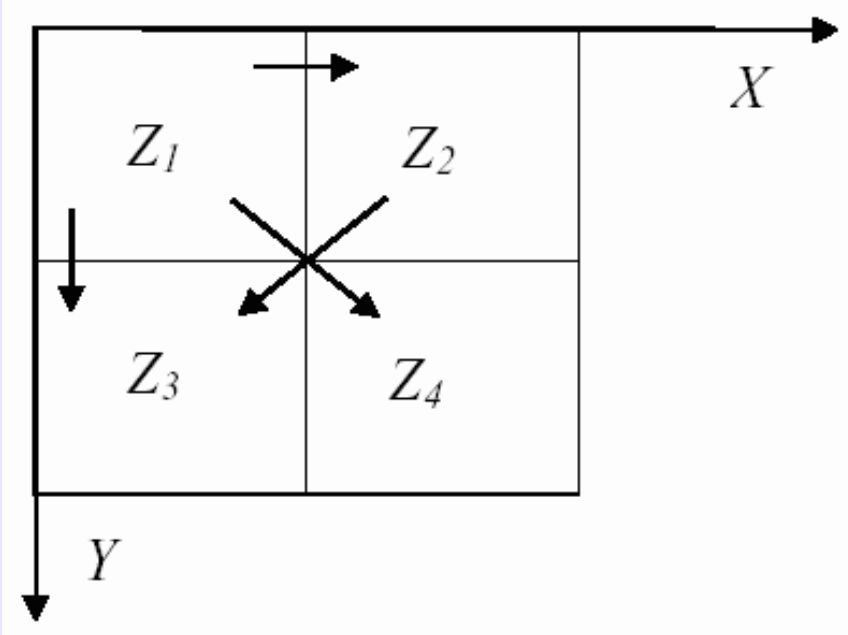
Image gradient

For dicrete images, horizontal and vertical gradients are approximated by:

$$\nabla f \approx |z_1 - z_2| + |z_1 - z_3|$$

For slanted directions:

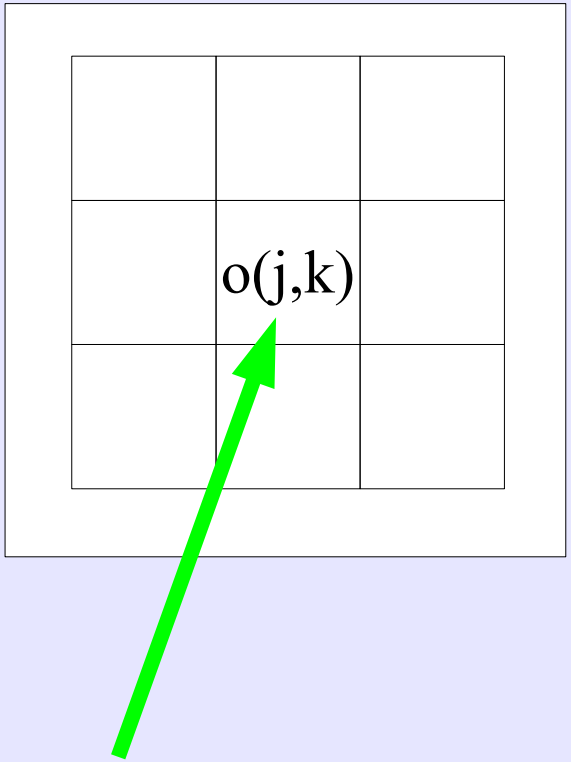
$$\nabla f \approx |z_1 - z_4| + |z_2 - z_3|$$



Sample edge filters:

Roberts,
Sobel,
Prewitt,
Laplace,
Kirsch

Roberts



−1	0
1	0

−1	1
0	0

Horizontal and vertical edges

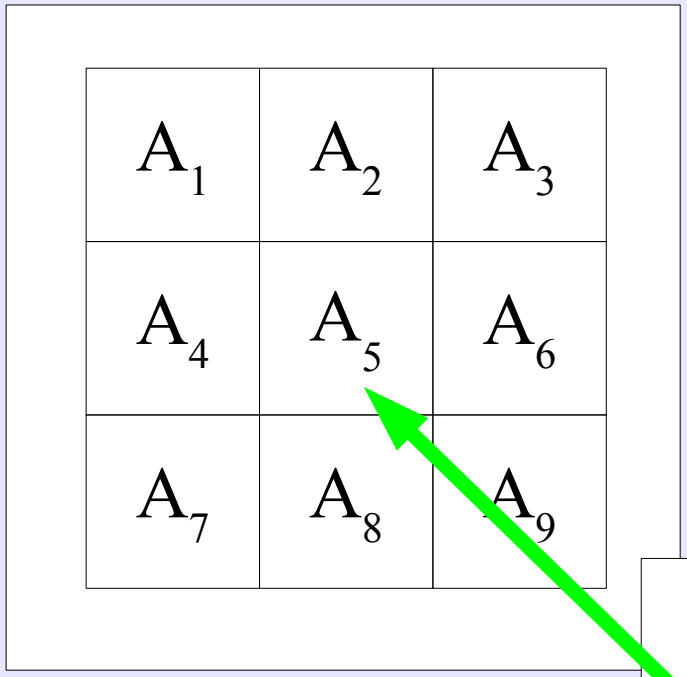
0	1
−1	0

1	0
0	−1

Slanted edges

$$o_w(j,k)=\sqrt{\left[o(j,k)-o(j+1,k+1)\right]^2+\left[o(j,k+1)-o(j+1,k)\right]^2}$$

Sobel (1)



$$o_w(j,k)=\sqrt{X^2+Y^2}$$

$$X=(A_2+2\ A_3+A_4)-(A_0+2\ A_7+A_6)$$

$$Y=(A_0+2\ A_1+A_2)-(A_6+2\ A_5+A_4)$$

Sobel (2)

−1	0	1
−2	0	2
−1	0	1

−1	−2	−1
0	0	0
1	2	1

Vertical / horizontal

0	1	1
−1	0	2
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

slanted

Prewitt

−1	0	1
−1	0	1
−1	0	1

−1	−1	−1
0	0	0
1	1	1

Vertical / horizontal

0	1	1
−1	0	1
−1	−1	0

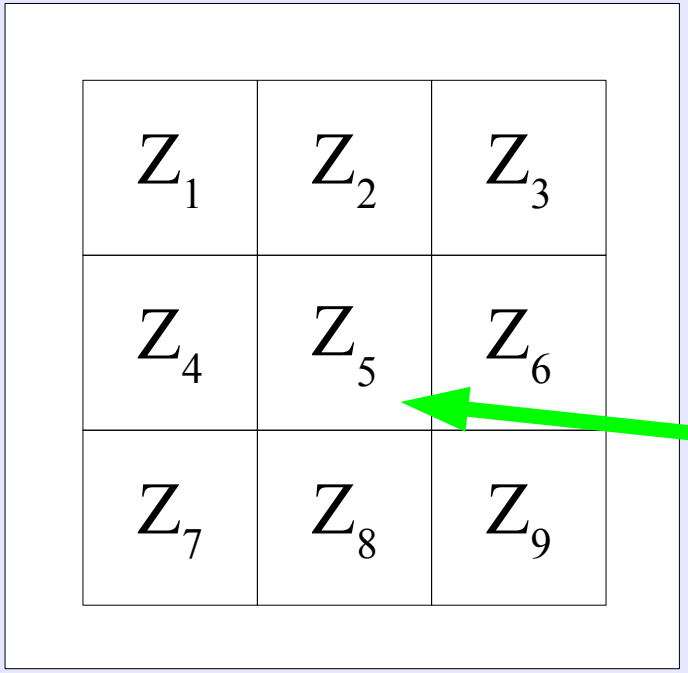
−1	−1	0
−1	0	1
0	1	1

slanted

Laplace operator

Laplace operator (Laplasian) is defined as a second derivative of image *f* at the location *(x,y)*

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



$$\nabla^2 f \approx 4z_5 - (z_2 + z_4 + z_6 + z_8)$$

Laplace

0	1	0
1	−4	1
0	1	0

or

1	1	1
1	−8	1
1	1	1

Kirsch (1)

A_0	A_1	A_2
A_7	o(j,k)	A_3
A_6	A_5	A_4

$$o_w(j,k) = max\left\{1, \max_{i\in\langle 0;7\rangle} |5S_i - 3T_i|\right\}$$

$$S_i = A_i + A_{i+1} + A_{i+2}$$

$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7}$$

where $i \in \langle 0;7 \rangle$

Indices change modulo 8

Kirsch (2)

-3	-3	5
-3	0	5
-3	-3	5

-3	5	5
-3	0	5
-3	-3	-3

5	5	5
-3	0	-3
-3	-3	-3

5	5	-3
5	0	-3
-3	-3	-3

5	-3	-3
5	0	-3
5	-3	-3

-3	-3	-3
5	0	-3
5	5	-3

-3	-3	-3
-3	0	-3
5	5	5

-3	-3	-3
-3	0	5
-3	5	5

Canny edge detector

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

An "optimal" edge detector means:

good detection – the algorithm should mark as many real edges in the image as possible.
good localization – edges marked should be as close as possible to the edge in the real image.
minimal response – a given edge in the image should only be marked once, and where possible, image noise should not create false edges.

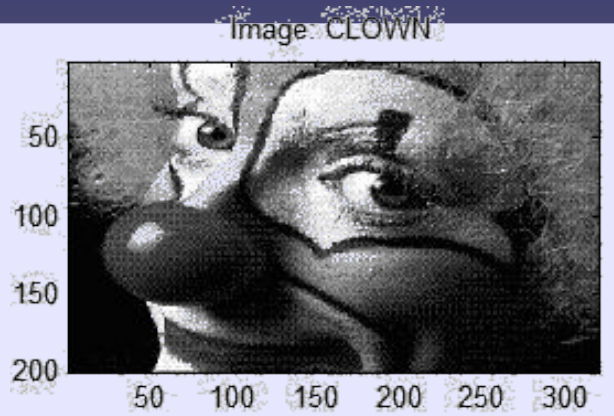
Algorithm

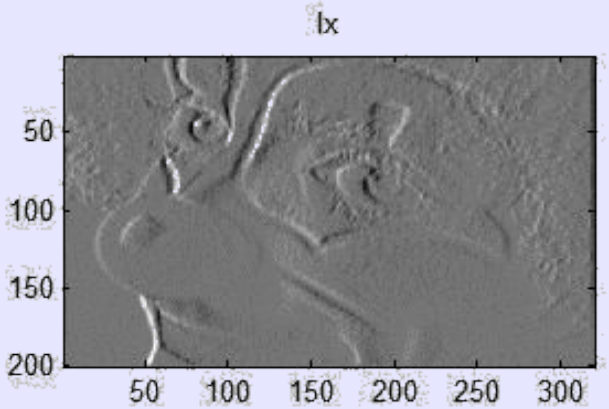
1. **Image smoothing** using Gaussian

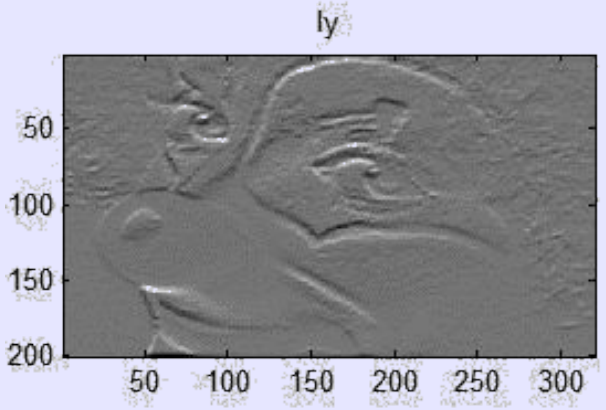
$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

2. **Derivatives** calulation using masks: [-1 0 1] i [-10 1]'.

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$
$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x),$$







Filtr Canny'ego - algorytm

3. **Non-maximum suppression** as an edge thinning technique.

Given estimates of the image gradients, a search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. In many implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction

4. **Tracing edges through the image and hysteresis thresholding**