Inverse Problems 1: Convolution and Deconvolution

Introduction to the Course MAST31401

Samuli Siltanen, Neil Dizon
Elli Karvonen, Victoria Kovalenko, Sara Parikka

University of Helsinki
September 5, 2023

Basic information about the course

- The course is lectured in period I (Sep 5 – Oct 19, 2023)
- 5 credits
- Lecturers: Samuli Siltanen, Neil Dizon
- Assistants: Elli Karvonen, Victoria Kovalenko, Sara Parikka

Lecture rooms and times

Lectures

- Tuesdays: 10:15 – 12:00 (Exactum C222)

Exercise sessions

- Thursdays 10:15 – 12:00 (Exactum C222)

Course materials

Book: The course is based on the book: Mueller J L and Siltanen S, *Linear and Nonlinear Inverse Problems with Practical Applications*. SIAM 2012.

PDF: The most relevant parts of the book will be shared on Moodle as a pdf.

Moodle: We will post lecture notes, codes, exercises, etc in our Moodle page.

https://moodle.helsinki.fi/course/view.php?id=58918

Self-enrolment key: deconvolution23

Videos: Parts of the course will be based on video lectures, previously recorded at the university. The links will be published on Moodle.

Programming language: The examples and exercises in the course are programmed in MATLAB.

Schedule (Partial)

The complete schedule is in our Moodle page!

What?	When?	Notes
Lecture: Exactum C222	5.9. at 10:15 -12:00	Lecturer: Neil
Matlab exercise (*)	Published: 5.9. Deadline: 15.9. at 23.00	Optional but recommended
Lecture: Exactum C222	12.9. at 10:15 -12:00	Lecturer: Neil
Exercise session 1: Exactum C222 / Chat	14.9. at 10:15-12:00	Course assistant: Sara / Elli
Exercise 1	Published: 8.9. Deadline: 15.9. at 23.00	worth 10 points
Lecture: Exactum C222	19.9. at 10:15 -12:00	Lecturer: Samuli
Exercise session 2: Exactum C222 / Chat	21.9. at 10:15-12:00	Course assistant: Sara / Elli
Exercise 2	Published: 15.9. Deadline: 22.9. at 23.00	worth 10 points
Lecture: Exactum C222	26.9. at 10:15 -12:00	Lecturer: Samuli
Exercise session 3: Exactum C222 / Chat	28.9. at 10:15-12:00	Course assistant: Sara / Elli
Exercise 3	Published: 22.9. Deadline: 29.9. at 23.00	worth 10 points
Lecture: Exactum C222	3.10. at 10:15 -12:00	Lecturer: Samuli

Grading

Grading is based on weekly exercises and a final exam.

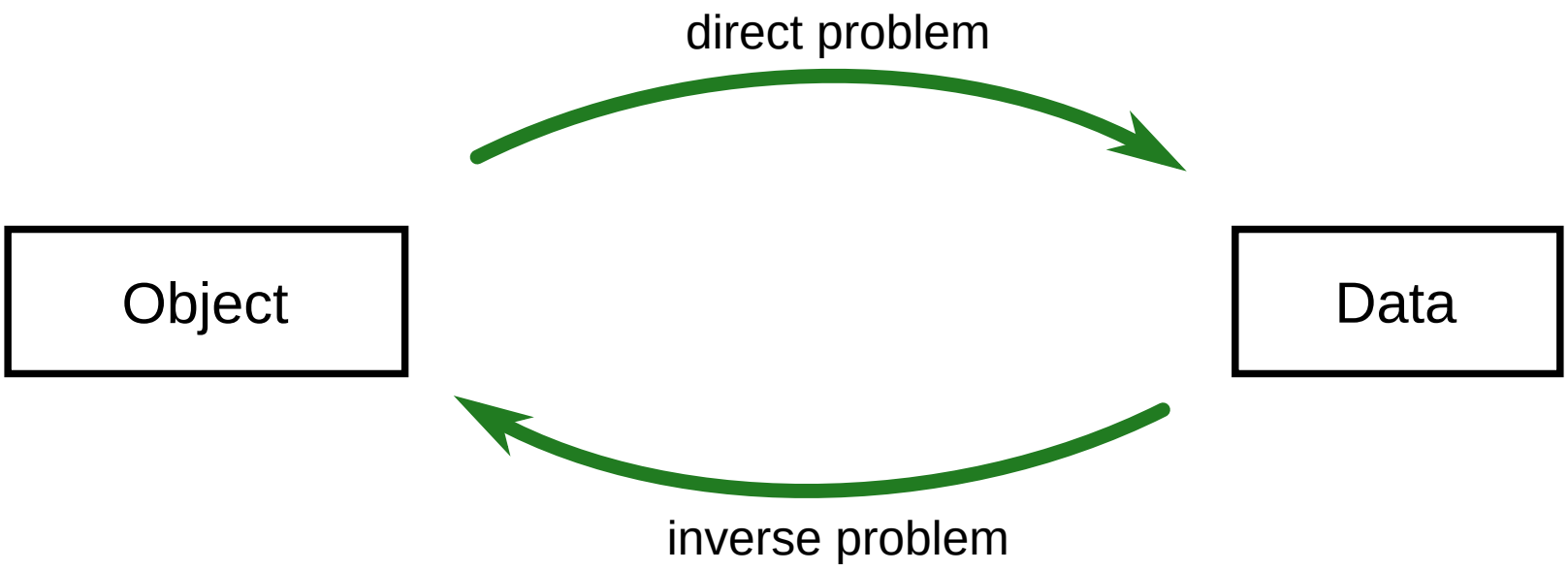
- The final exam is worth 40 points. (When? A poll is on Moodle.)
- There will be six exercise sets, published every week on our Moodle page, covering the material of that week. Each set is worth 10 points (total: $6 \times 10 = 60$ points).
- Students will have seven days to answer the exercises.

Grade	Total points
0	< 30(exercises) OR < 20(exam)
1	≥ 50
2	≥ 60
3	≥ 70
4	≥ 80
5	≥ 90

Direct vs. Inverse problems

Direct problem: *given object, determine data*

Inverse problem: *given noisy data, recover object*



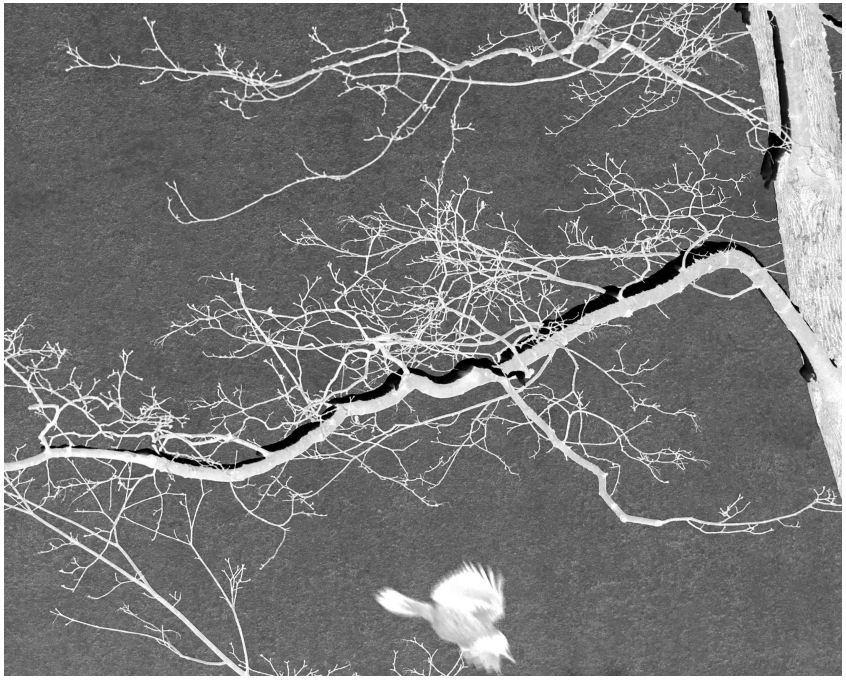
Direct vs. Inverse problems

Direct problem: *given object, determine data*

Inverse problem: *given noisy data, recover object*

Object (positive photograph)

Data (negative photograph)



Forward map: subtraction from a constant

Direct vs. Inverse problems

Direct problem: *given object, determine data*

Inverse problem: *given noisy data, recover object*

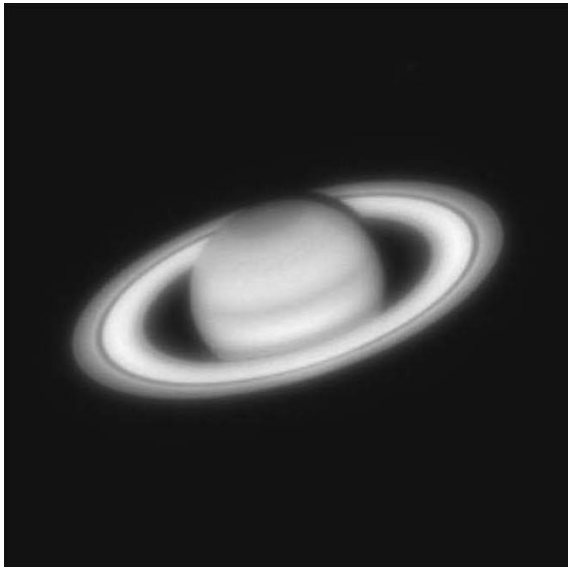
Object (sharp photograph)

Data (blurred and noisy photo)

Forward map: convolution operator

The Hubble space telescope, launched in 1990, first gave blurred images due to a flawed mirror

Hubble telescope

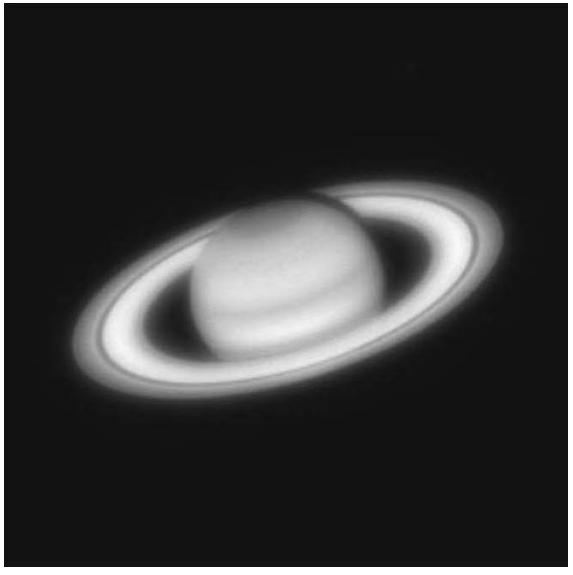


Saturnus (blurred)

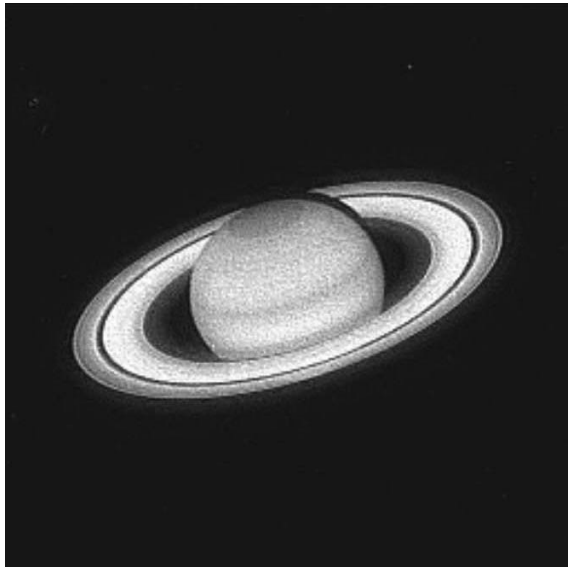
Images: NASA, ESA, Quarktet

The mirror was replaced in 1993. The new sharp images are further enhanced with deconvolution!

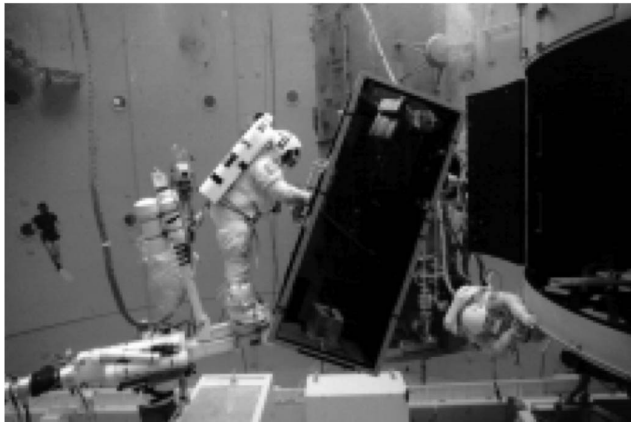
Hubble telescope



Saturnus (blurred)



Saturnus (corrected)



Images: NASA, ESA, Quarktet

At a glance

Convolution is a mathematical model that can describe physical phenomena like image blurring and averaging of signals.

Figure: Motion blur

At a glance

Convolution is a mathematical model that can describe physical phenomena like image blurring and averaging of signals.

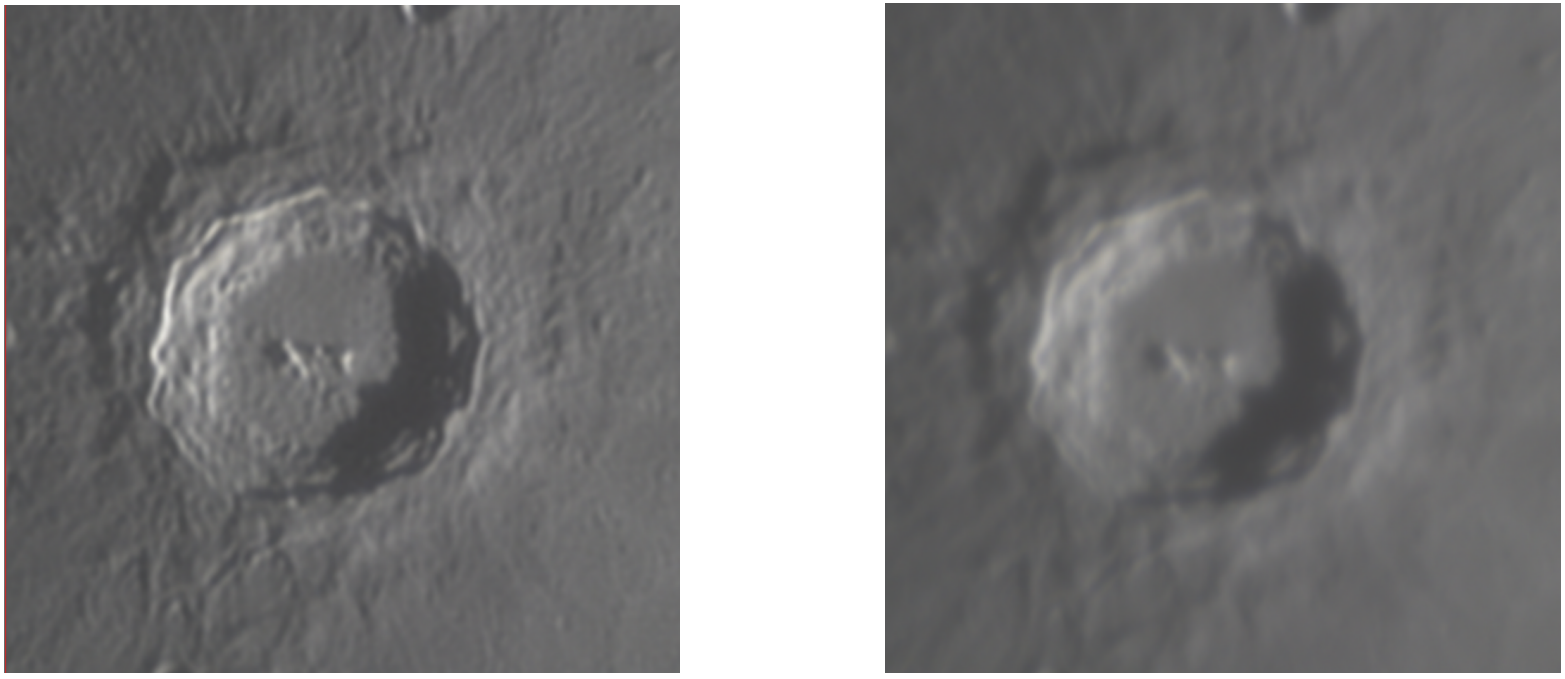
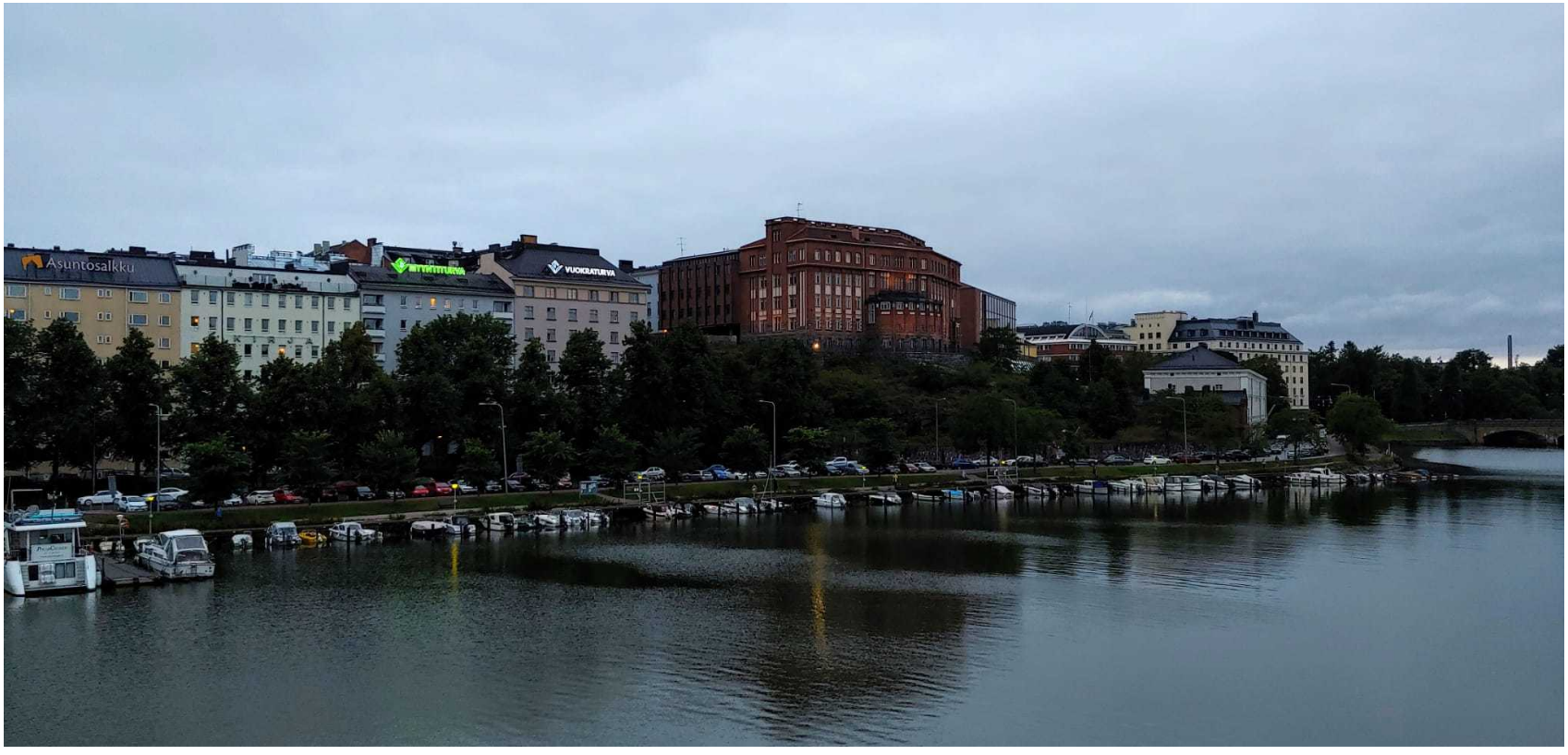


Figure: Defocus aberration

Credits: `https://en.wikipedia.org/wiki/Deconvolution`

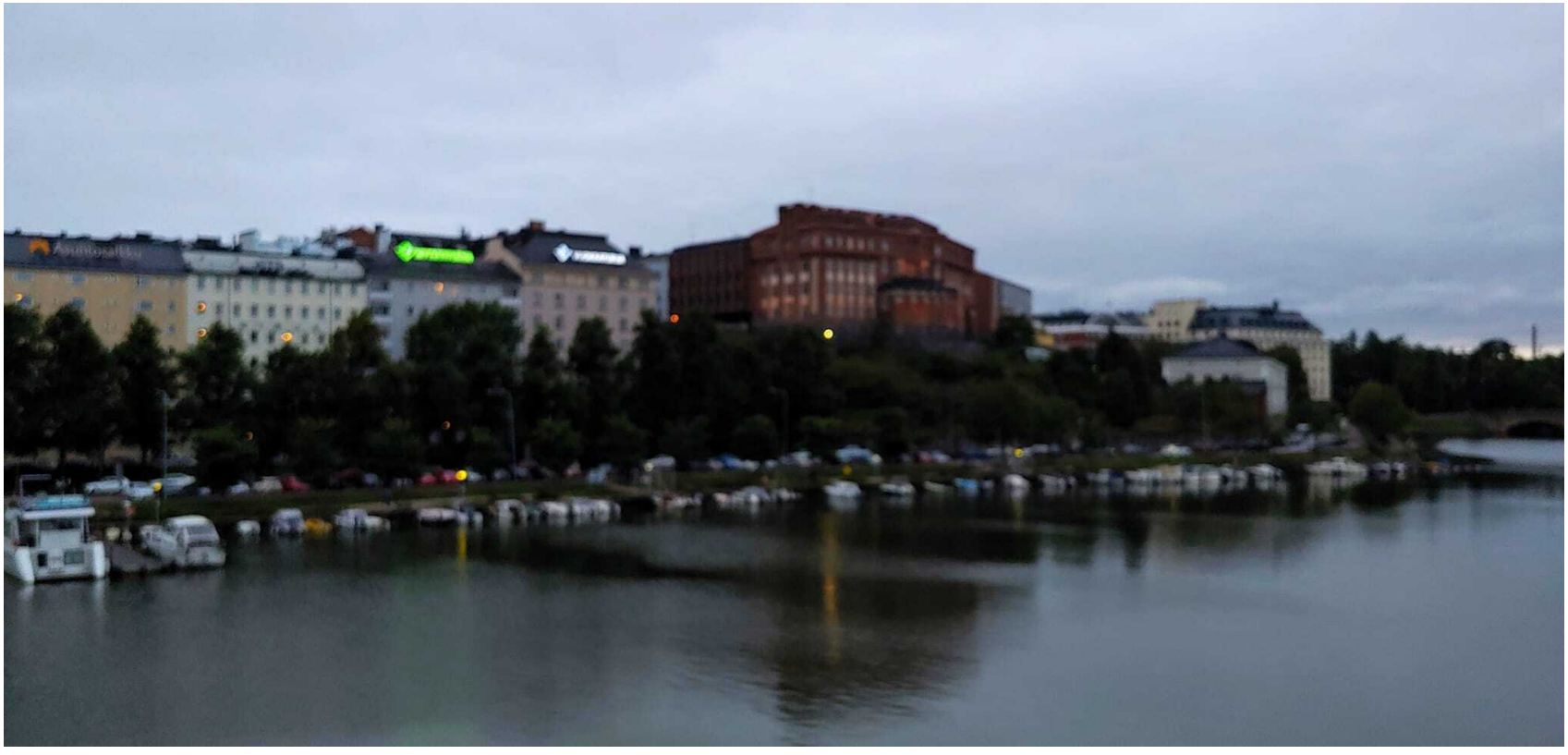
Problem with focus?

When taking a picture, we should always **adjust the focus** of the camera...



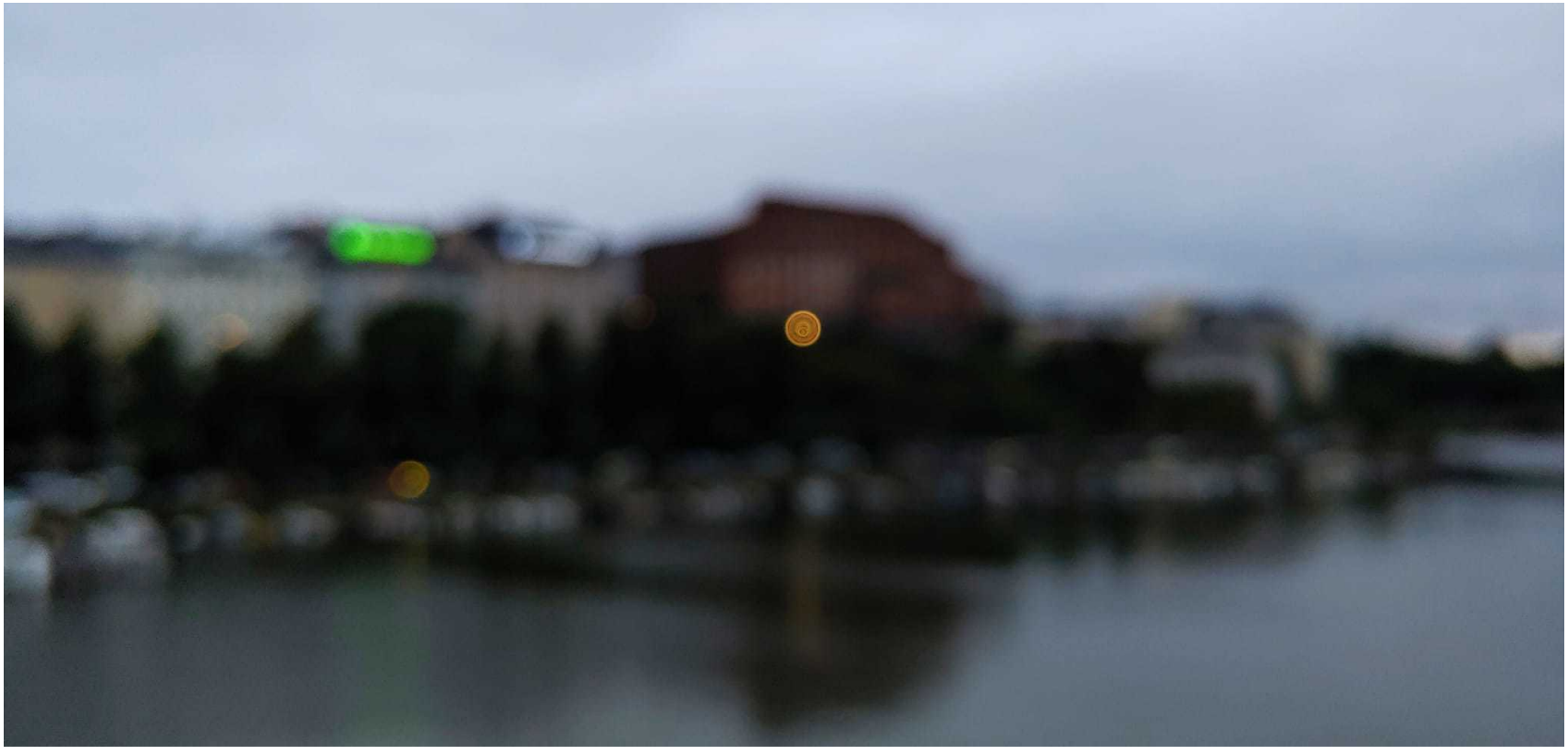
Problem with focus?

When taking a picture, we should always **adjust the focus** of the camera...



Problem with focus?

When taking a picture, we should always **adjust the focus** of the camera...

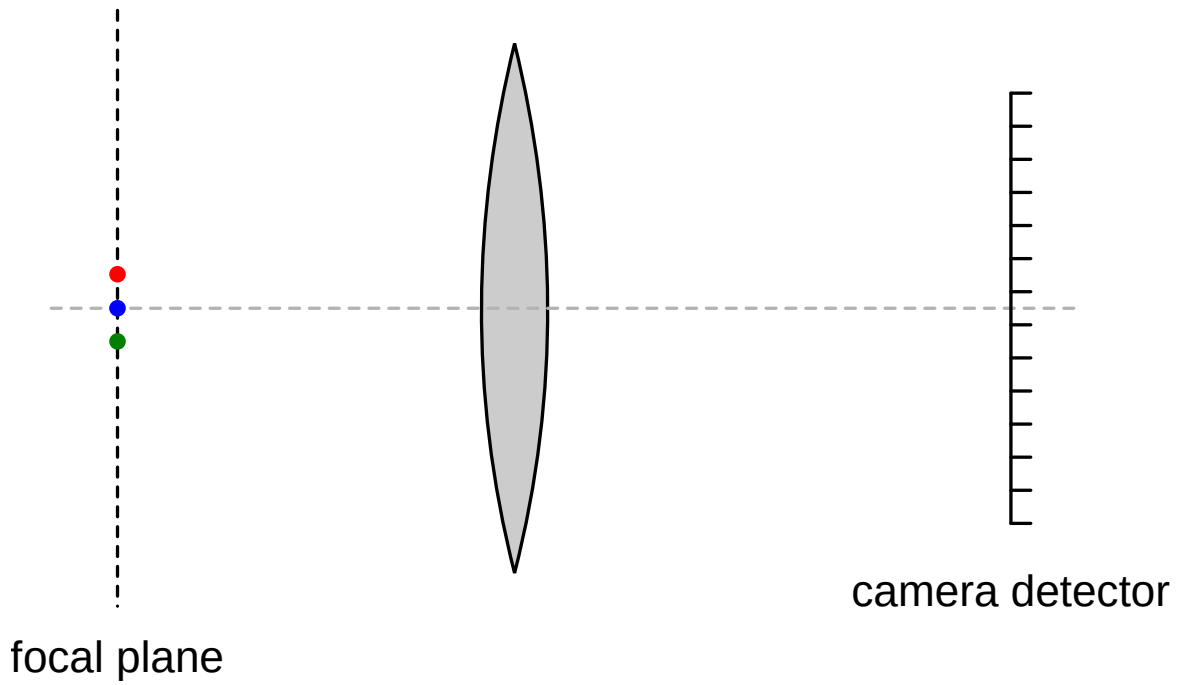


Main idea

Convolution occurs when the value of a signal in one point is influenced by the values of the neighboring points.

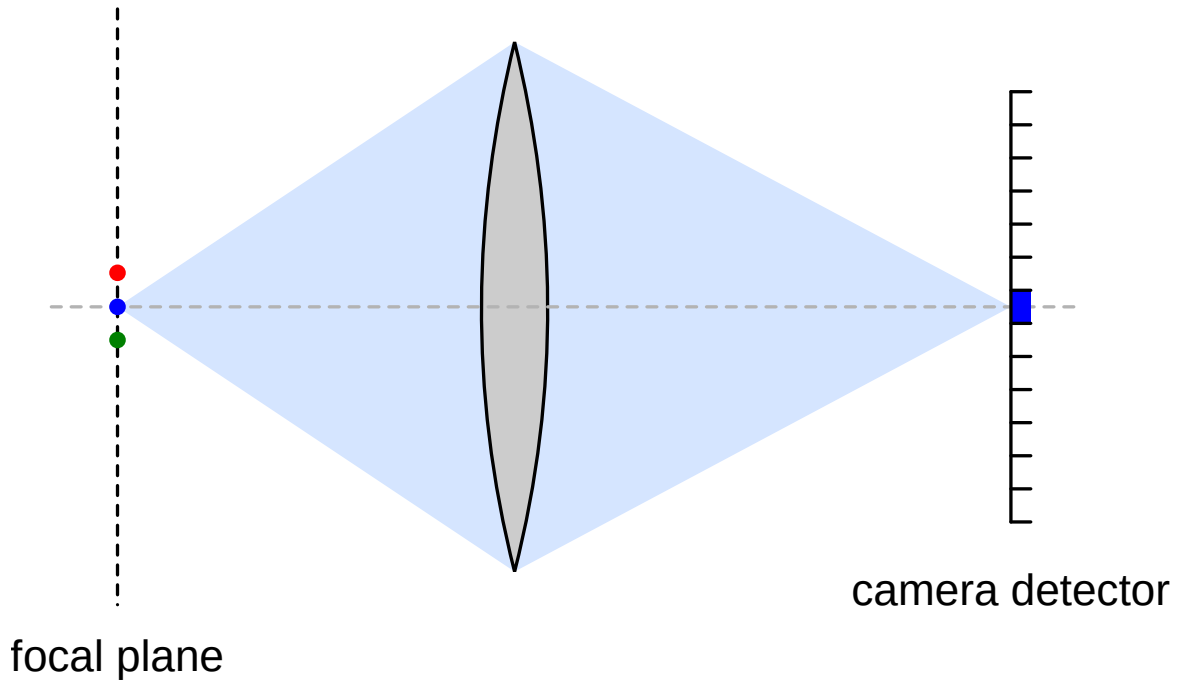
Some physics can help

Some basic concepts of optics can describe the behaviour of lenses



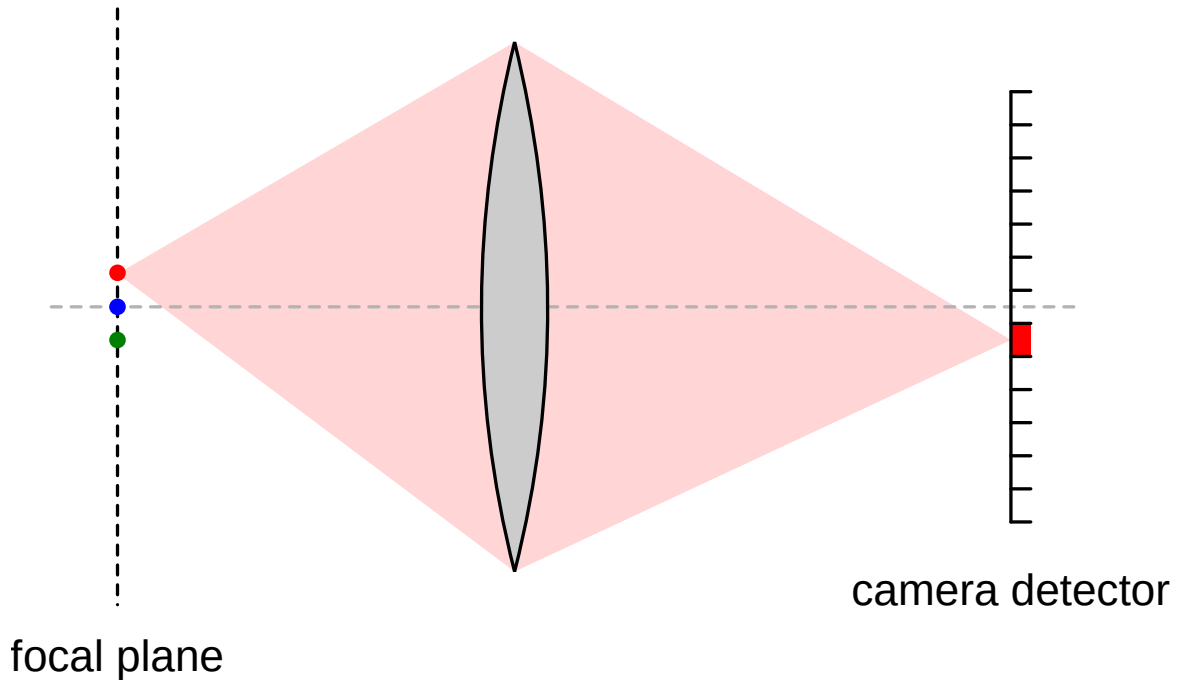
Some physics can help

An object on focus: all the light emitted by a point hit a single cell in the detector



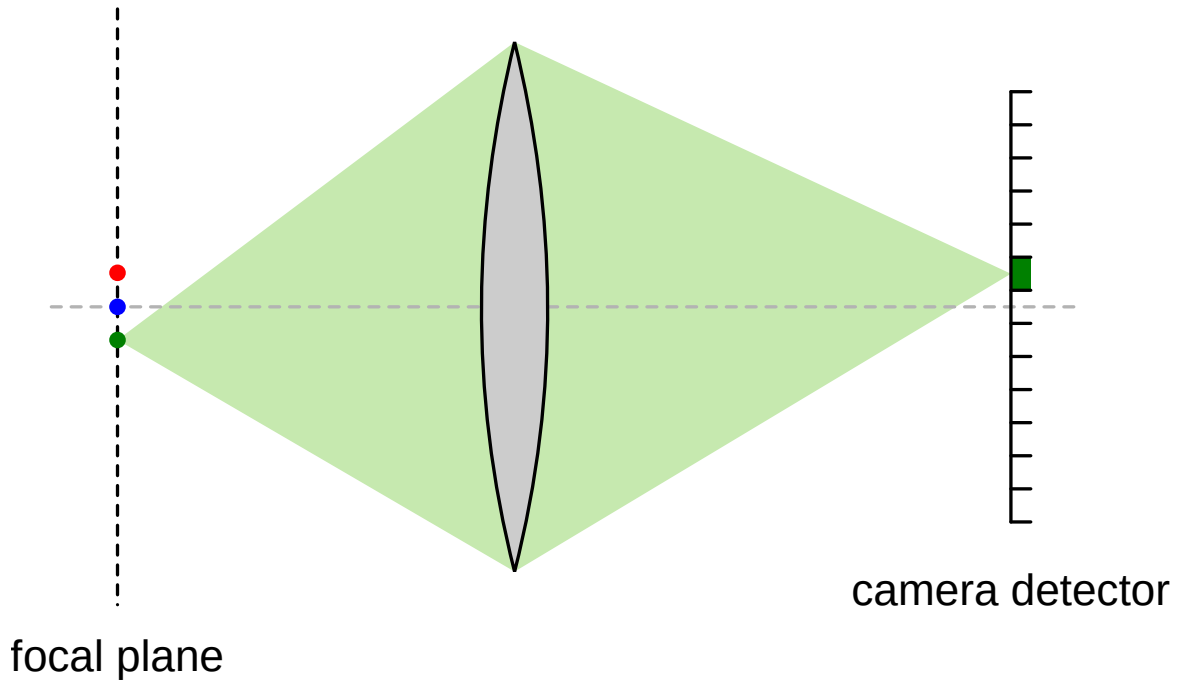
Some physics can help

An object on focus: all the light emitted by a point hit a single cell in the detector



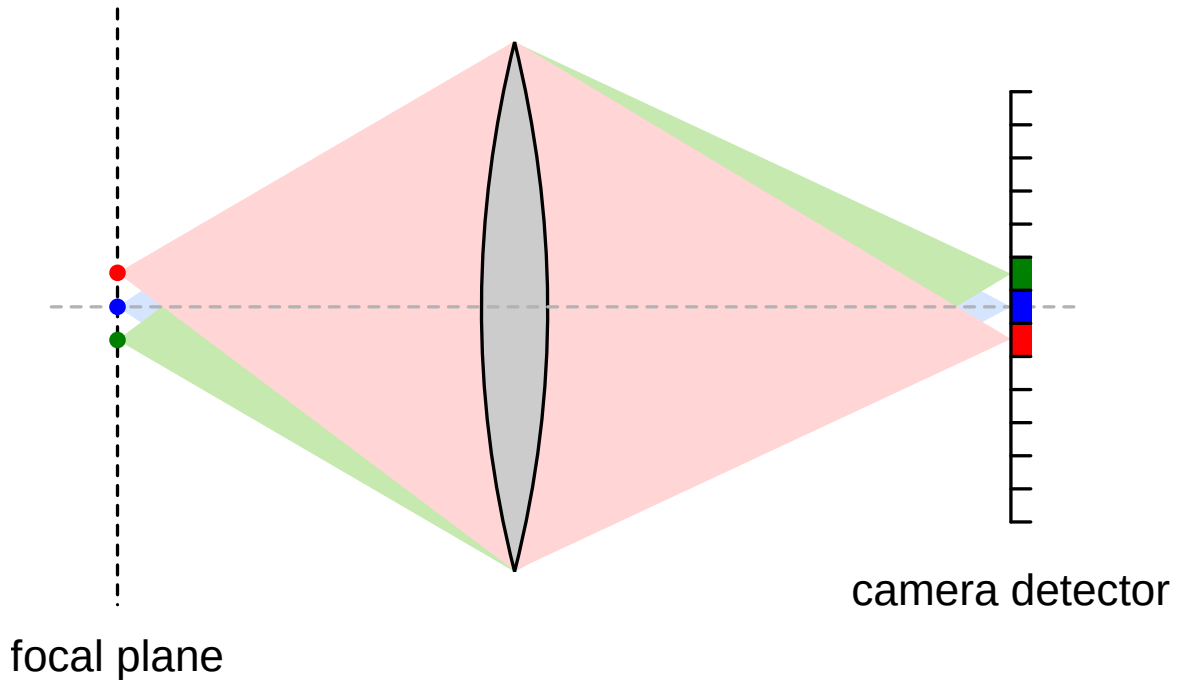
Some physics can help

An object on focus: all the light emitted by a point hit a single cell in the detector



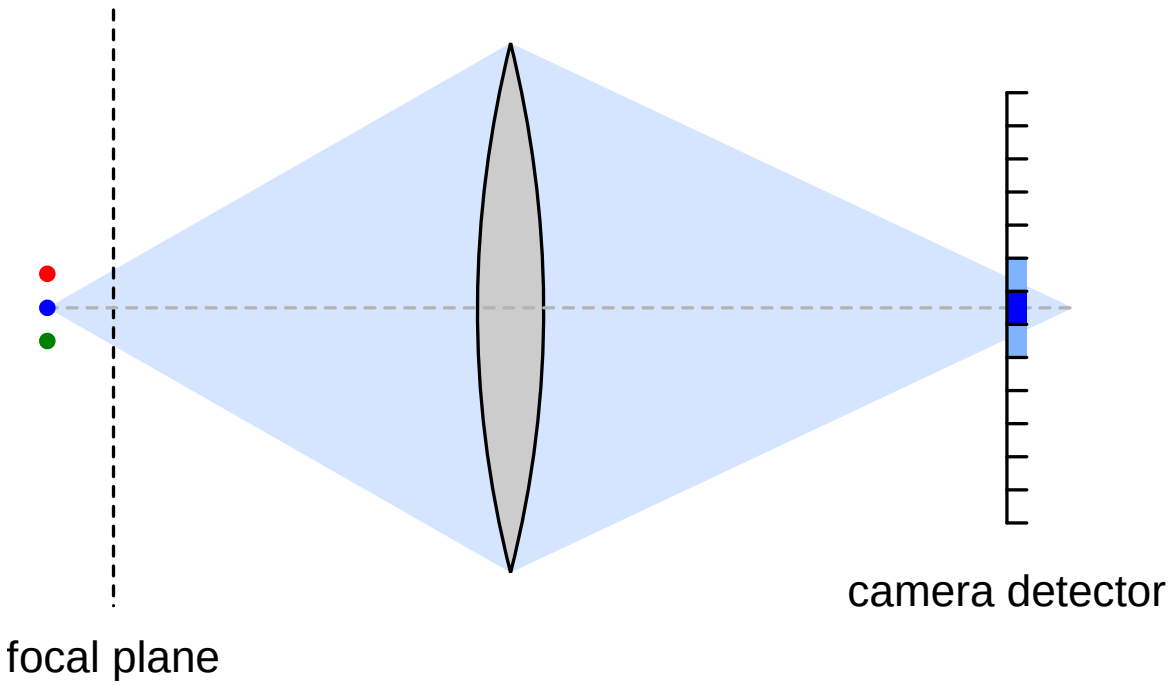
Some physics can help

An object on focus: all the light emitted by a point hit a single cell in the detector



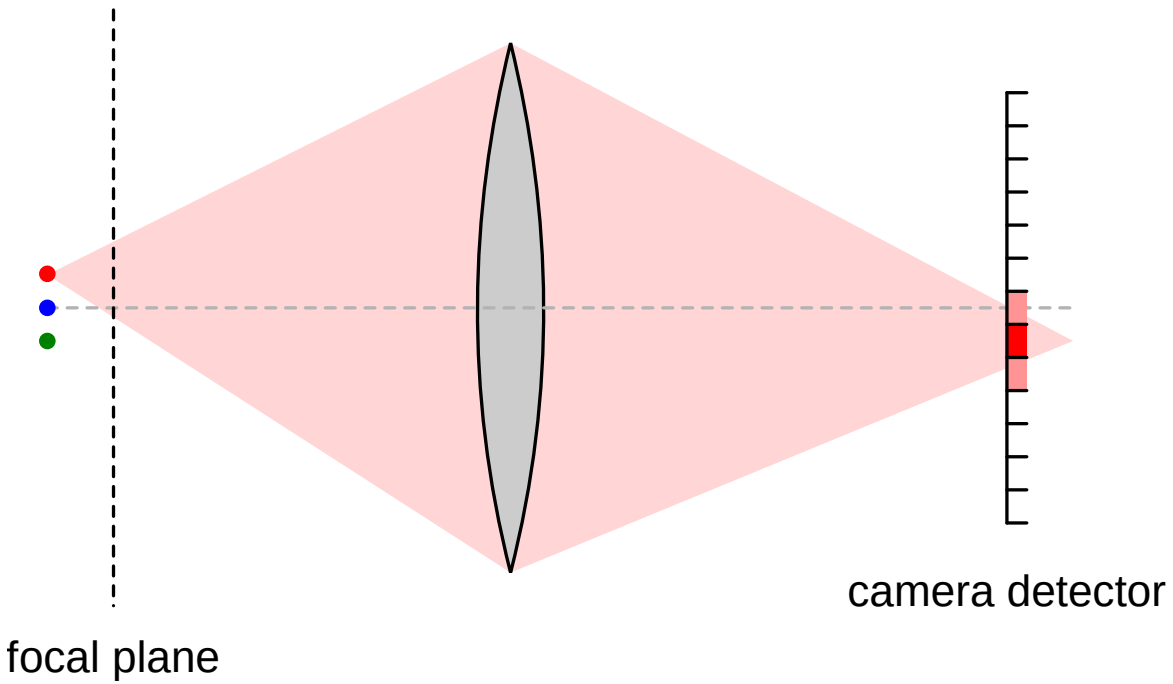
Some physics can help

An object out of focus: light emitted by a point hit more than one cell in the detector



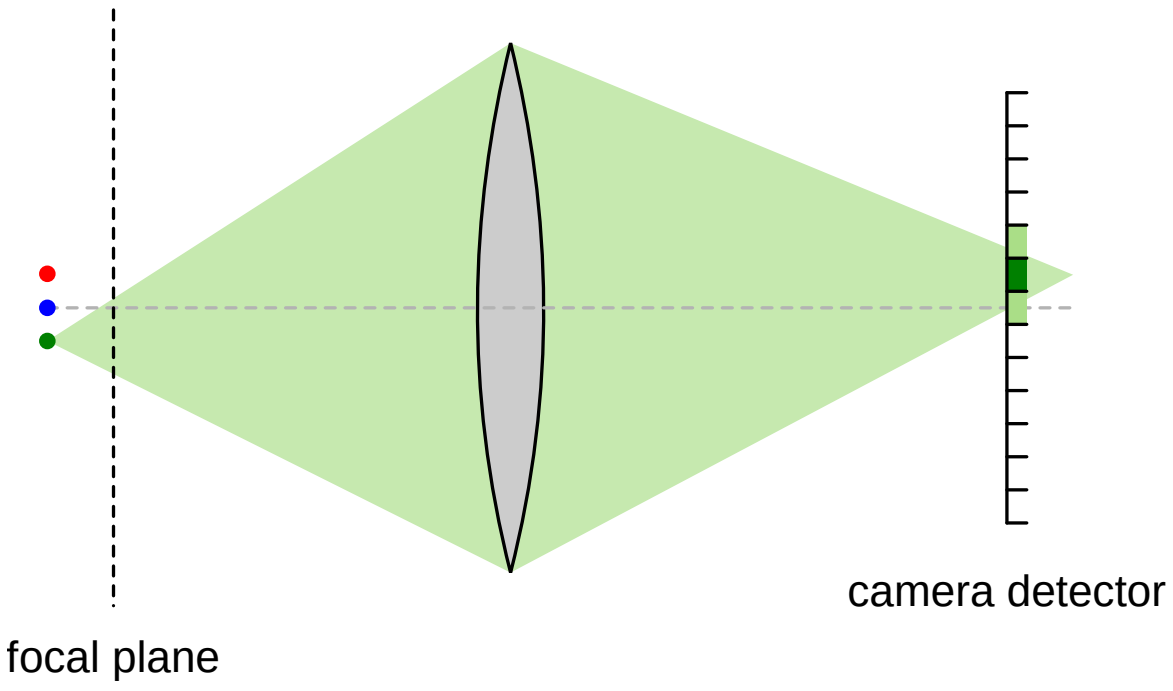
Some physics can help

An object out of focus: light emitted by a point hit more than one cell in the detector



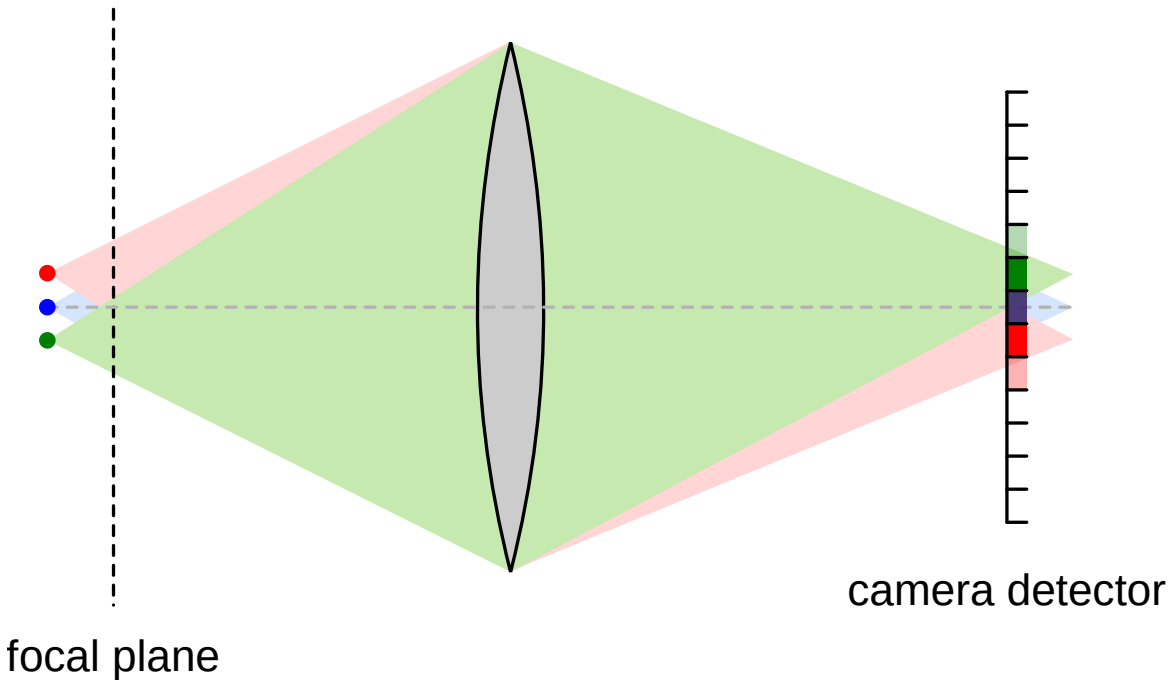
Some physics can help

An object out of focus: light emitted by a point hit more than one cell in the detector



Some physics can help

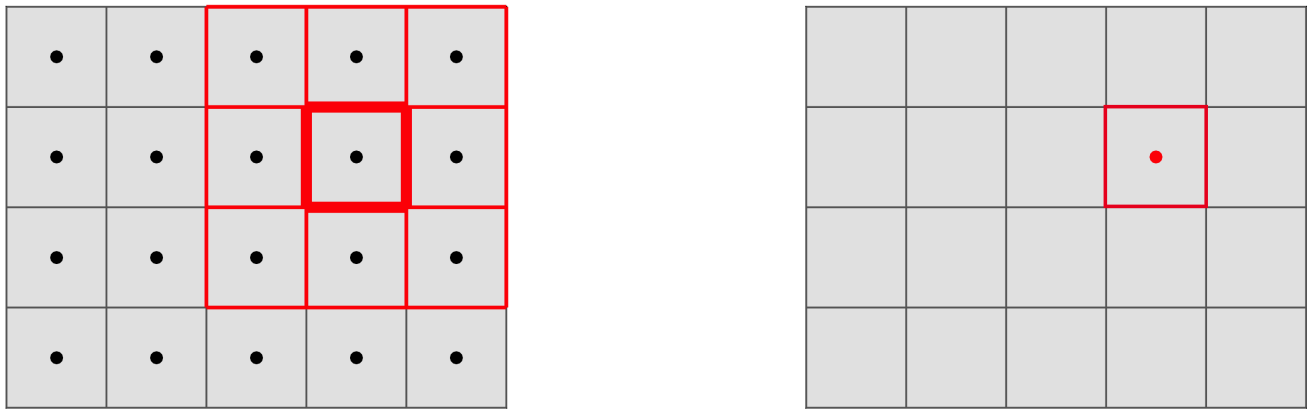
An object out of focus: light rays are spread



Visualization of convolution

Digital pictures are stored as pixel maps (small, uniformly colored, squares). Grayscale pictures: each pixel is described by a single **number** ranging from 0 (black) to 1 (white).

The effect of **convolution**: each pixel of the blurred picture is also affected by the light that should converge in the neighboring pixels.



How do the neighboring pixels affect the current pixel? This information is encoded in the **point spread function**.

One-dimensional convolution

Convolution can occur on images but also on **one-dimensional** signals.

Example: in audio signals, convolution can describe to same extent the phenomenon of **reverb**.

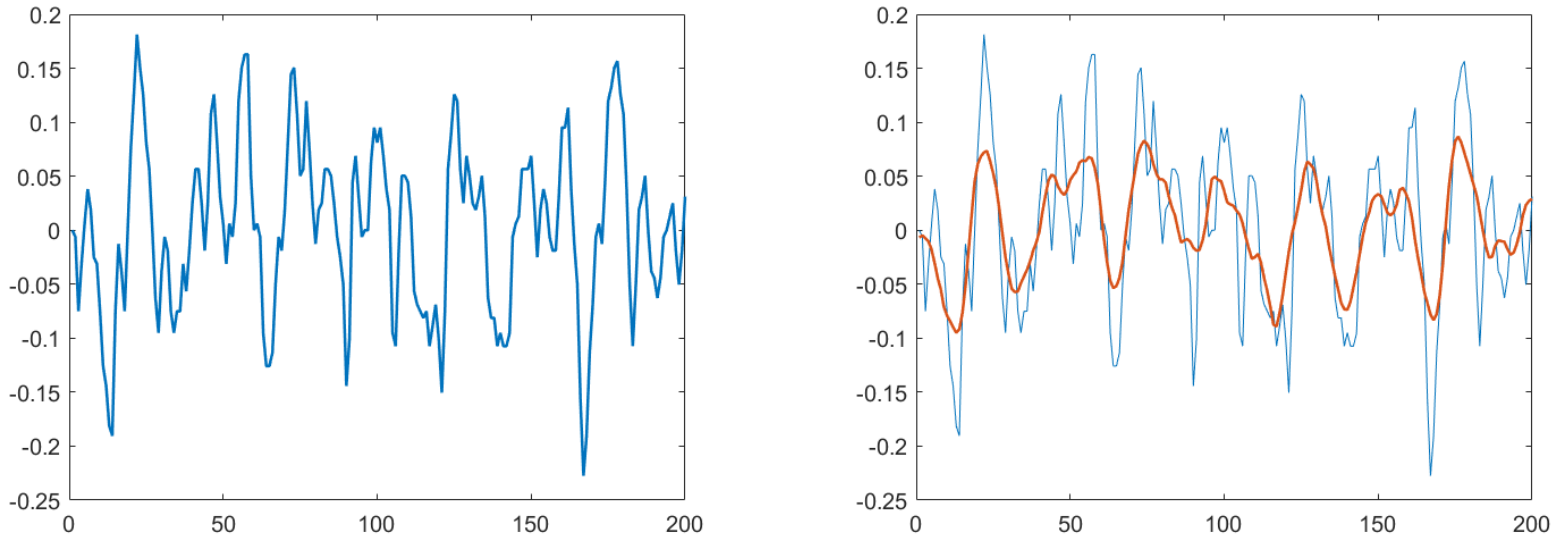
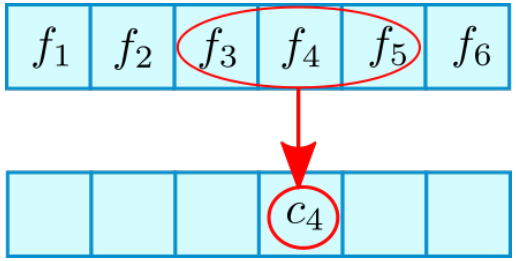


Figure: Left: original audio; Right: original and low-pass filter

Mathematical intuition

Schematic visualization: neighboring values affect the signal in each component.

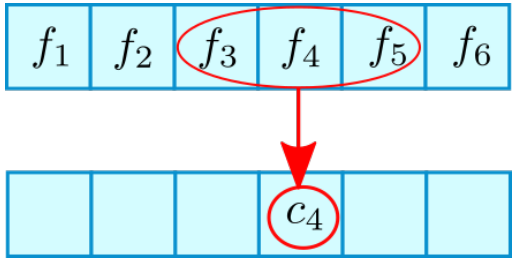


Mathematical formulation - idea

Convolution is an operation taking an original signal (f) and returning a new one (c). Each component of the new signal is a suitable **weighted sum** of the neighboring ones.

Mathematical intuition

Schematic visualization: neighboring values affect the signal in each component.



Mathematical formulation - idea

Convolution is an operation taking an original signal (f) and returning a new one (c). Each component of the new signal is a suitable **weighted sum** of the neighboring ones.These weights are coded in the **point spread function** (a.k.a. **convolution kernel**).

This generalizes also to 2D and even more. Other math can help more: e.g., *linear operator can be represented by matrices...*

Let's study a simple example of the convolution process

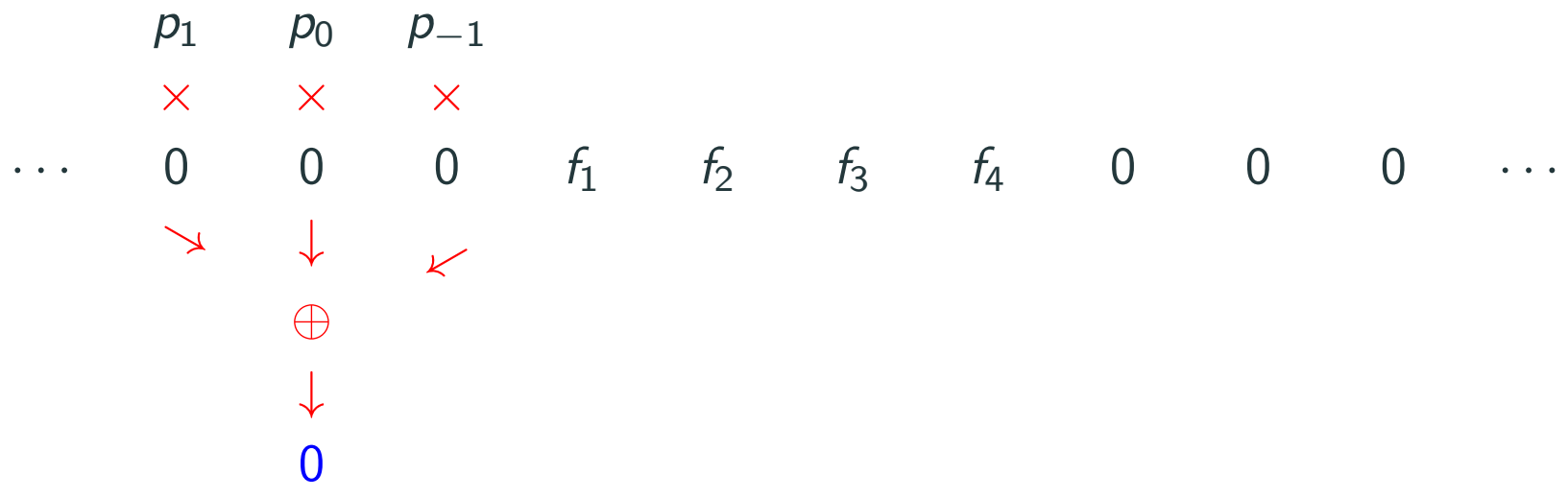
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

	p_1	p_0	p_{-1}							
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

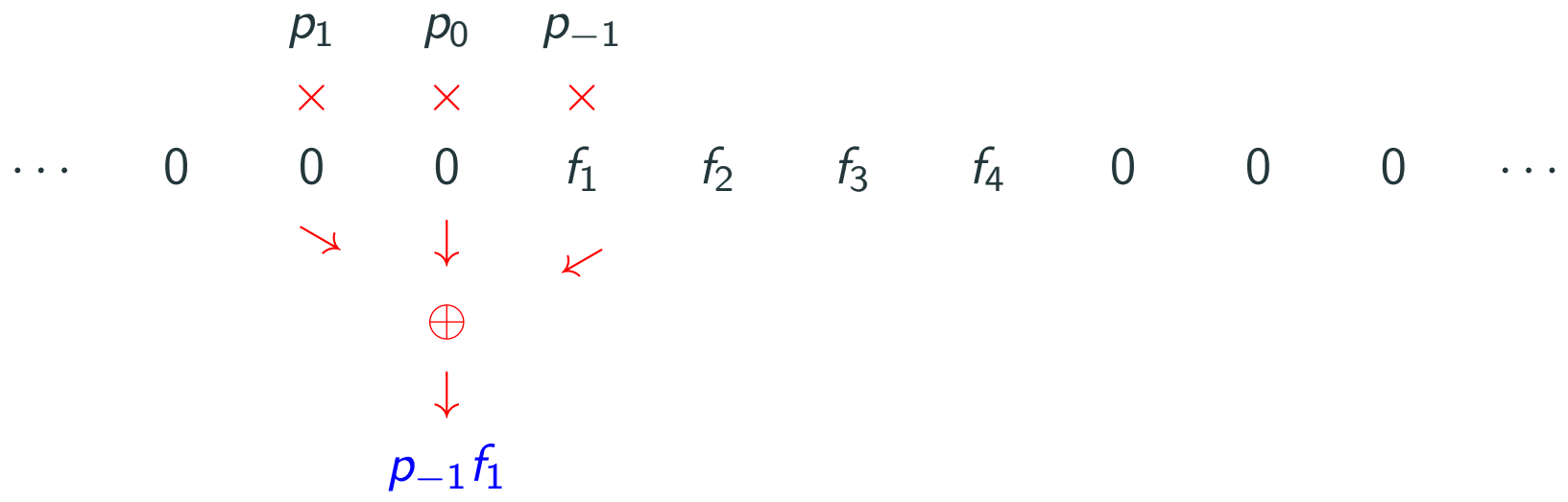
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

		p_1	p_0	p_{-1}						
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

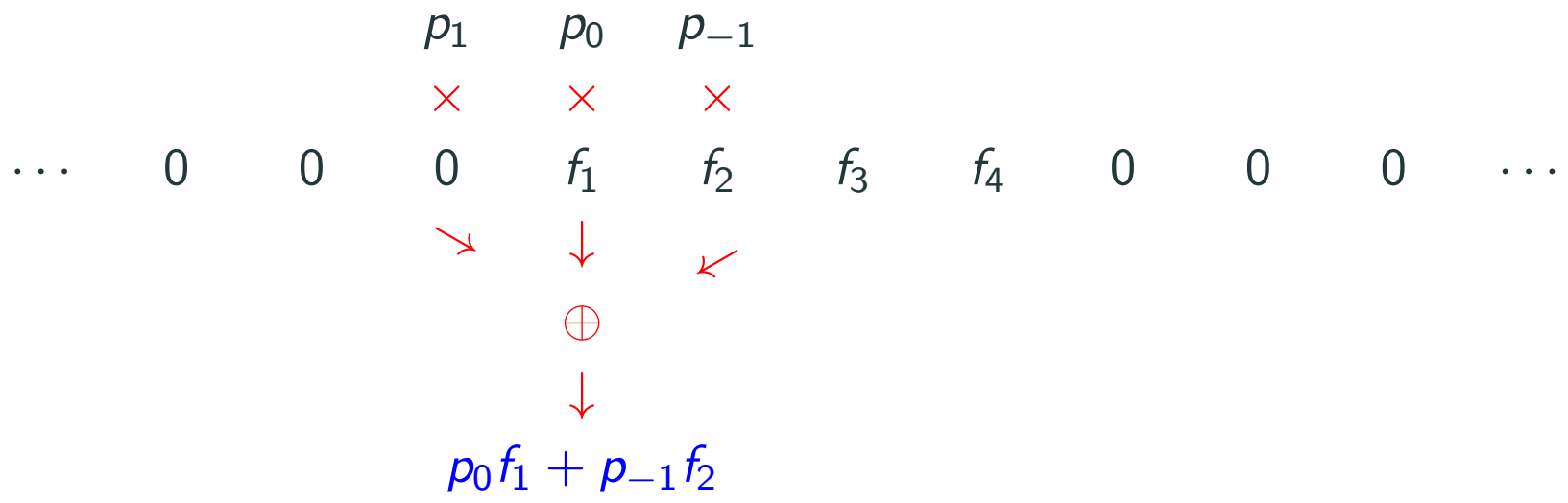
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

			p_1	p_0	p_{-1}						
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

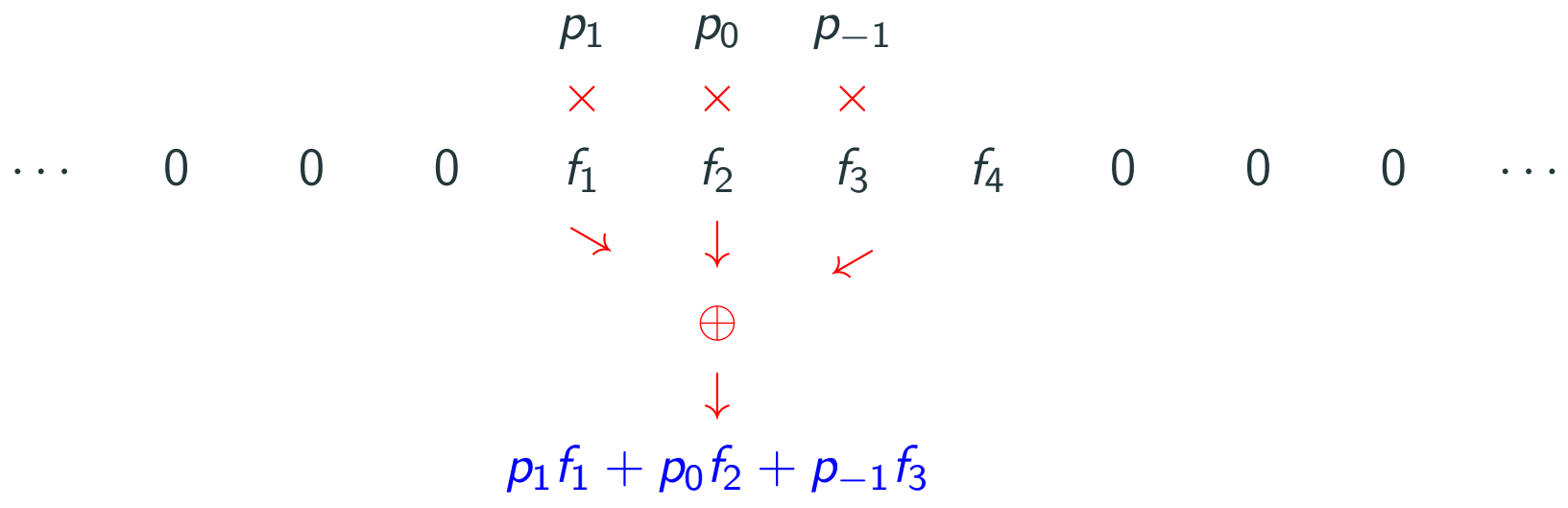
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

$$\begin{array}{cccccccccccc} & & & & p_1 & p_0 & p_{-1} & & & & & \\ \cdots & 0 & 0 & 0 & f_1 & f_2 & f_3 & f_4 & 0 & 0 & 0 & \cdots \end{array}$$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

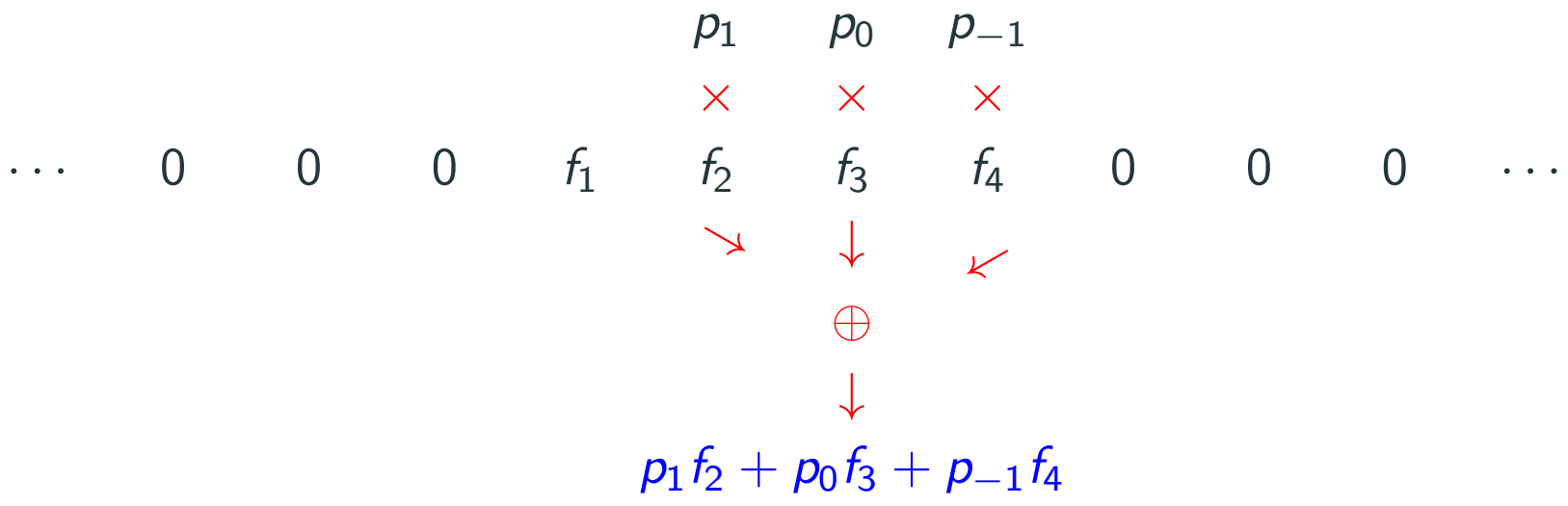
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

$$\begin{array}{cccccccccccc} & & & & p_1 & p_0 & p_{-1} & & & & & \\ \cdots & 0 & 0 & 0 & f_1 & f_2 & f_3 & f_4 & 0 & 0 & 0 & \cdots \end{array}$$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

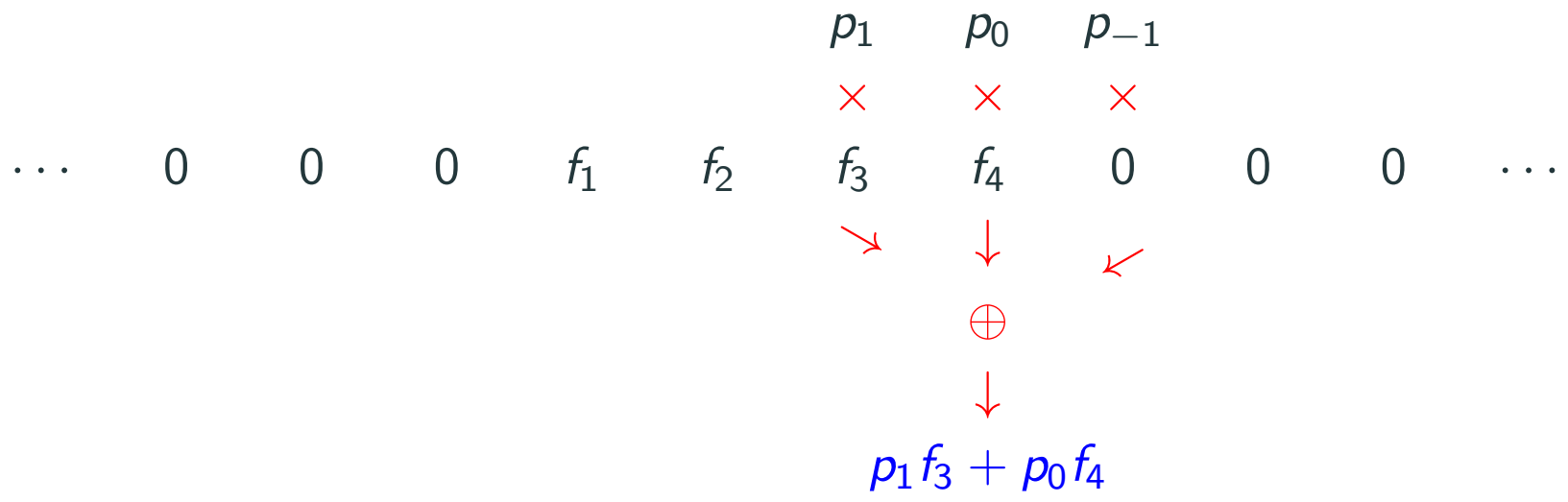
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

						p_1	p_0	p_{-1}			
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

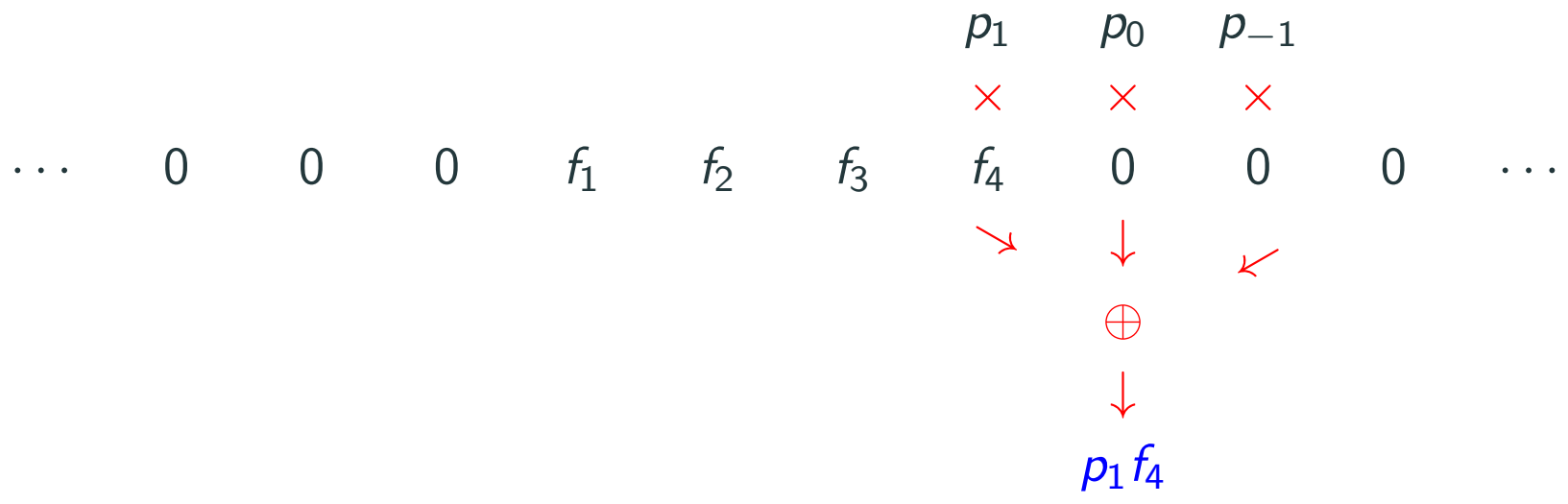
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

							p_1	p_0	p_{-1}		
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

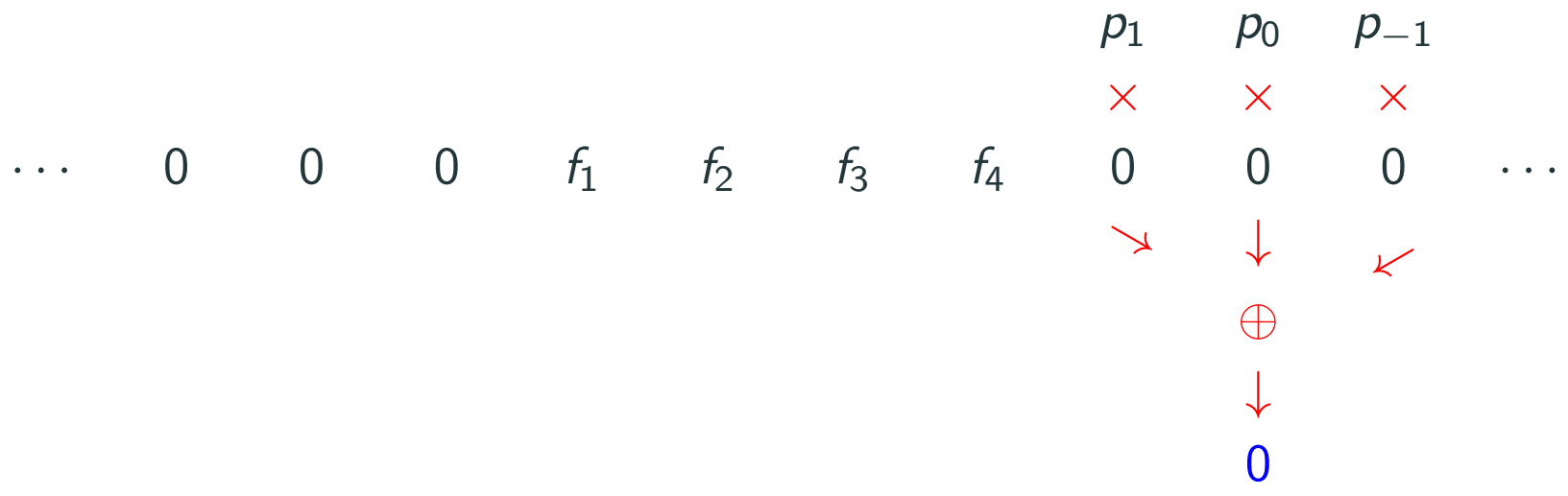
A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:

								p_1	p_0	p_{-1}	
$\cdots$	0	0	0	f_1	f_2	f_3	f_4	0	0	0	$\cdots$

Result: another infinite sequence arising as linear combinations of f_i.

Let's study a simple example of the convolution process

A **convolution kernel** $p = [p_{-1}\ p_0\ p_1]^T \in \mathbb{R}^3$ is moving step-by-step on top of an infinite number sequence, where only four elements (f_1, f_2, f_3, f_4) are allowed to be nonzero:



Result: another infinite sequence arising as linear combinations of f_i.

Just a note about the convolution kernel

There are many kinds of important convolution kernels.

We will often work with a kernel having the following properties.

- Positive elements: $p_i > 0$.
- Normalized: $p_{-1} + p_0 + p_1 = 1$, so that constant signals stay unchanged under convolution.
- The central element p_0 is close to one, so $0 < p_{-1} \ll 1$ and $0 < p_1 \ll 1$. This means that when we try to measure f_j, actually a little portion of f_{j-1} and f_{j+1} is added to the mix: $(p * f)_j = p_{-1} f_{j+1} + p_0 f_j + p_1 f_{j-1}$.
- Symmetry: $p_{-1} = p_1$.

Write the above process in terms of vectors

Convolution kernel $p \in \mathbb{R}^3$ and signal f as vectors:

$$p = \begin{bmatrix} p_{-1} \\ p_0 \\ p_1 \end{bmatrix}, \qquad f = \begin{bmatrix} \vdots \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \end{bmatrix}$$

Elements of the infinite convolution sequence $p * f$ are defined by

$$(p * f)_j = \sum_{k=-1}^{1} p_k f_{j-k} \qquad j \in \mathbb{Z}$$

But if f_1, f_2, f_3, f_4 are the only nonzero inputs, could we limit the index j to a finite range? We surely can, and there are several options.

Limiting the index j

Option 1: Consider only the (possibly) nonzero stuff

In this case we have $(p * f) \in \mathbb{R}^6$, with $(p * f)_j = \sum_{k=-1}^{1} p_k f_{j-k}$. Explicitly written as a 6-dimensional vector:

$$p * f = \begin{bmatrix} (p*f)_0 \\ (p*f)_1 \\ \vdots \\ \\ \\ \\ (p*f)_5 \end{bmatrix} = \begin{bmatrix} p_{-1} f_1 \\ p_0 f_1 + p_{-1} f_2 \\ p_1 f_1 + p_0 f_2 + p_{-1} f_3 \\ p_1 f_2 + p_0 f_3 + p_{-1} f_4 \\ p_1 f_3 + p_0 f_4 \\ p_1 f_4 \end{bmatrix}.$$

(Note that $j = 0, 1, 2, 3, 4, 5$ in formula $(p * f)_j = \sum_{k=-1}^{1} p_k f_{j-k}$.)

Limiting the index j

Option 1: Consider only the (possibly) nonzero stuff

Written as matrix-vector product:

$$p * f = Af = \begin{bmatrix} p_{-1} & 0 & 0 & 0 \\ p_0 & p_{-1} & 0 & 0 \\ p_1 & p_0 & p_{-1} & 0 \\ 0 & p_1 & p_0 & p_{-1} \\ 0 & 0 & p_1 & p_0 \\ 0 & 0 & 0 & p_1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

Matrix A is called **convolution matrix**.

Note that in this case $A : \mathbb{R}^4 \to \mathbb{R}^6$

Building square-matrix models for convolution

In some cases we might want to have a square matrix that describes the convolution.

For that, we ignore elements outside the limits of f. For our example, if $f = [f_1, f_2, f_3, f_4] \in \mathbb{R}^4$

$$
\begin{aligned}
\cancel{(p * f)_0} &= p_1 f_{-1} + p_0 f_0 + p_{-1} f_1 \\
(p * f)_1 &= p_1 f_0 + p_0 f_1 + p_{-1} f_2 \\
(p * f)_2 &= p_1 f_1 + p_0 f_2 + p_{-1} f_3 \\
(p * f)_3 &= p_1 f_2 + p_0 f_3 + p_{-1} f_4 \\
(p * f)_4 &= p_1 f_3 + p_0 f_4 + p_{-1} f_5 \\
\cancel{(p * f)_5} &= p_1 f_4 + p_0 f_5 + p_{-1} f_6
\end{aligned}
$$

In this case we have $(p * f) \in \mathbb{R}^4$, with $(p * f)_j = \sum_{k=-1}^{1} p_k f_{j-k}$. We still need to deal with the 'ghost' elements outside the limits of f. We have a few options...

Limiting the index j

Option 2: zero boundary conditions: assume $f_j = 0$ for any index j outside the range of f. Element-by-element it looks like this:

$$\begin{aligned}
(p * f)_1 &= \cancel{p_1 f_0}^{\,0} + p_0 f_1 + p_{-1} f_2 \\
(p * f)_2 &= p_1 f_1 + p_0 f_2 + p_{-1} f_3 \\
(p * f)_3 &= p_1 f_2 + p_0 f_3 + p_{-1} f_4 \\
(p * f)_4 &= p_1 f_3 + p_0 f_4 + \cancel{p_{-1} f_5}^{\,0}
\end{aligned}$$

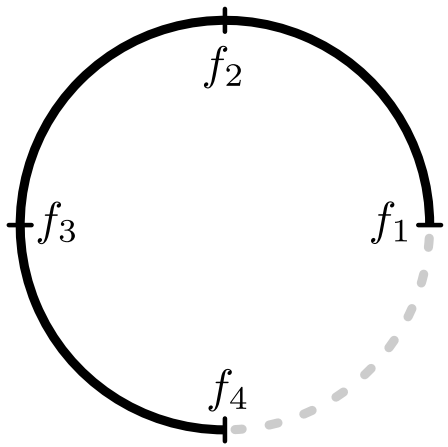
Written as matrix-vector product:

$$p * f = Af = \begin{bmatrix} p_0 & p_{-1} & 0 & 0 \\ p_1 & p_0 & p_{-1} & 0 \\ 0 & p_1 & p_0 & p_{-1} \\ 0 & 0 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

Note that in this case $A : \mathbb{R}^4 \to \mathbb{R}^4$

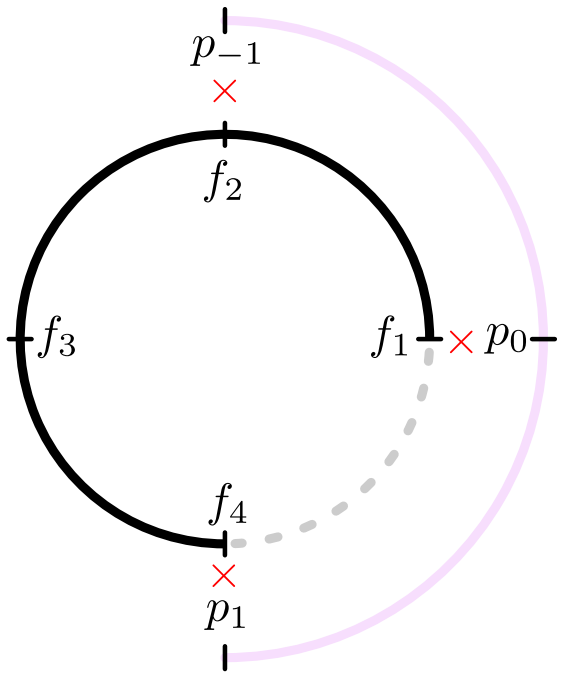
Limiting the index j

Option 3: we can arrange the signal along a circle (periodic boundary conditions).



Limiting the index j

Option 3: we can arrange the signal along a circle (periodic boundary conditions). The convolution kernel rotates around the signal

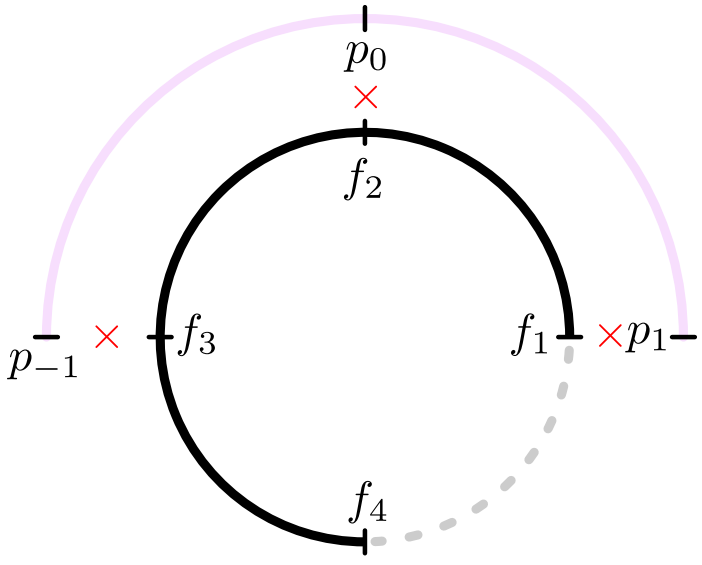


$$(p * f)_1 = p_1 f_0 + p_0 f_1 + p_{-1} f_2$$

f_4

Limiting the index j

Option 3: we can arrange the signal along a circle (periodic boundary conditions). The convolution kernel rotates around the signal

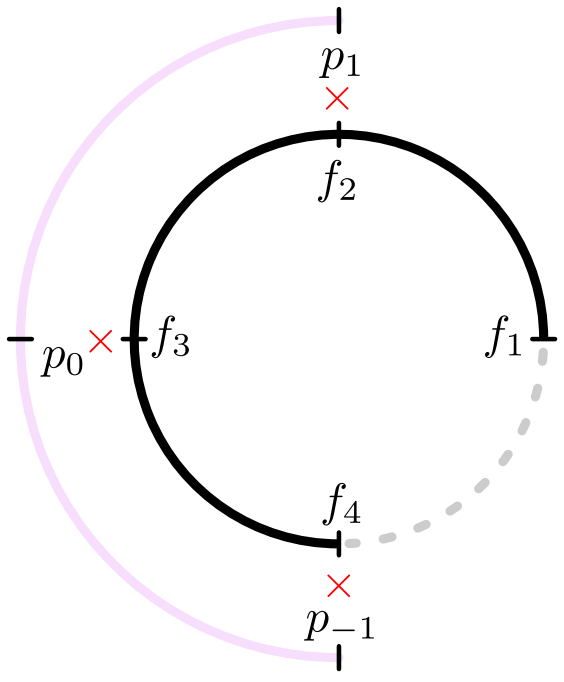


$$(p * f)_1 = p_1 f_0 + p_0 f_1 + p_{-1} f_2$$

$$(p * f)_2 = p_1 f_1 + p_0 f_2 + p_{-1} f_3$$

Limiting the index j

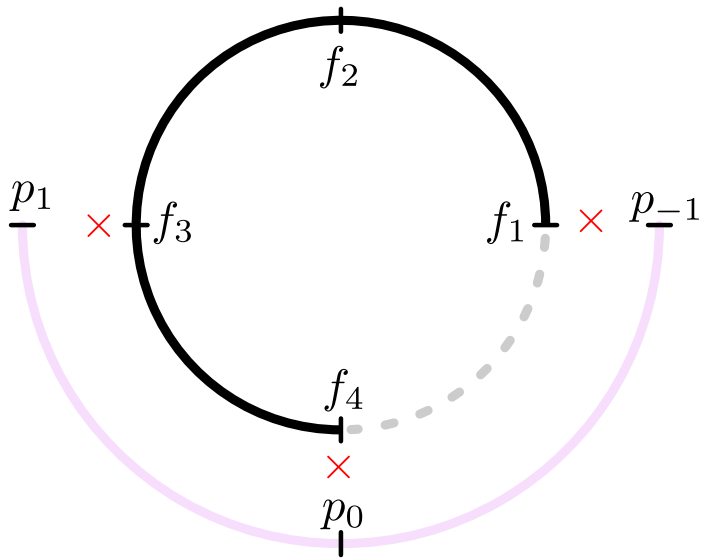
Option 3: we can arrange the signal along a circle (periodic boundary conditions). The convolution kernel rotates around the signal



$$(p * f)_1 = p_1 f_0 + p_0 f_1 + p_{-1} f_2$$
$$(p * f)_2 = p_1 f_1 + p_0 f_2 + p_{-1} f_3$$
$$(p * f)_3 = p_1 f_2 + p_0 f_3 + p_{-1} f_4$$

Limiting the index j

Option 3: we can arrange the signal along a circle (periodic boundary conditions). The convolution kernel rotates around the signal



$$(p * f)_1 = p_1 f_0 + p_0 f_1 + p_{-1} f_2 \quad (f_4)$$

$$(p * f)_2 = p_1 f_1 + p_0 f_2 + p_{-1} f_3$$

$$(p * f)_3 = p_1 f_2 + p_0 f_3 + p_{-1} f_4$$

$$(p * f)_4 = p_1 f_3 + p_0 f_4 + p_{-1} f_5 \quad (f_1)$$

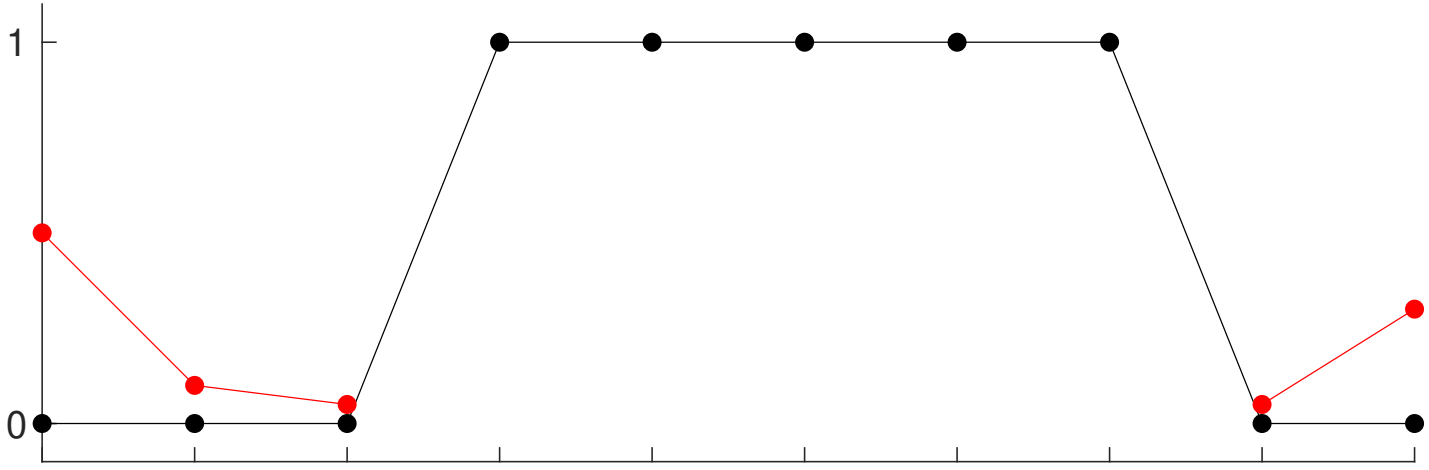
Limiting the index j

Written as matrix-vector product:

$$p * f = Af = \begin{bmatrix} p_0 & p_{-1} & 0 & p_1 \\ p_1 & p_0 & p_{-1} & 0 \\ 0 & p_1 & p_0 & p_{-1} \\ p_{-1} & 0 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix},$$

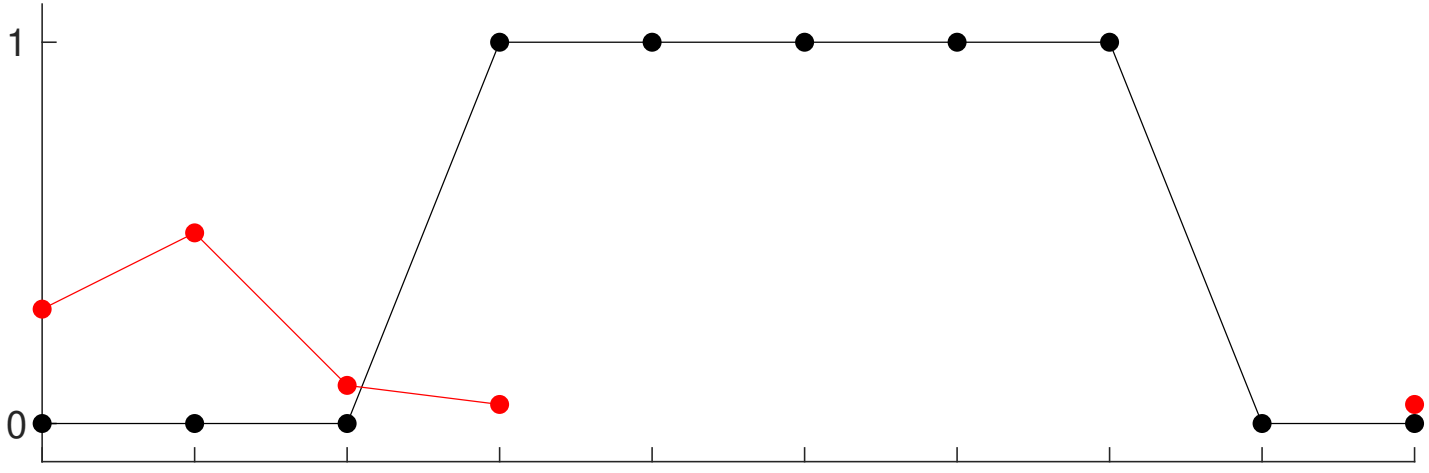
so this time we define $f_0 := f_4$ and $f_5 := f_1$ instead of putting them to zero (periodic boundary conditions).

Periodic boundary condition



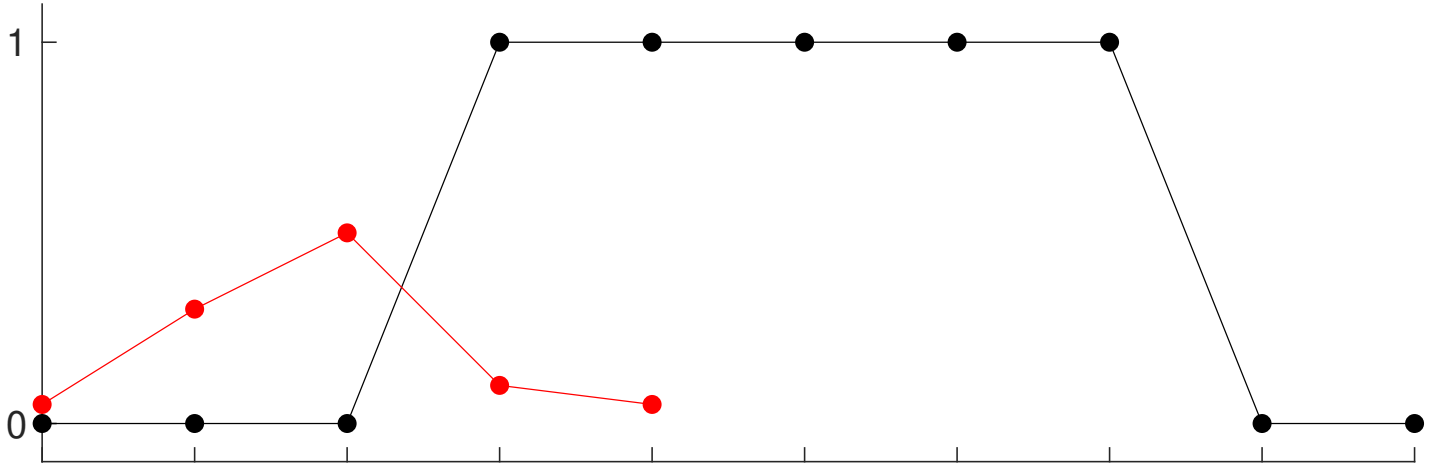
$$\begin{bmatrix} 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \end{bmatrix}$$

Periodic boundary condition



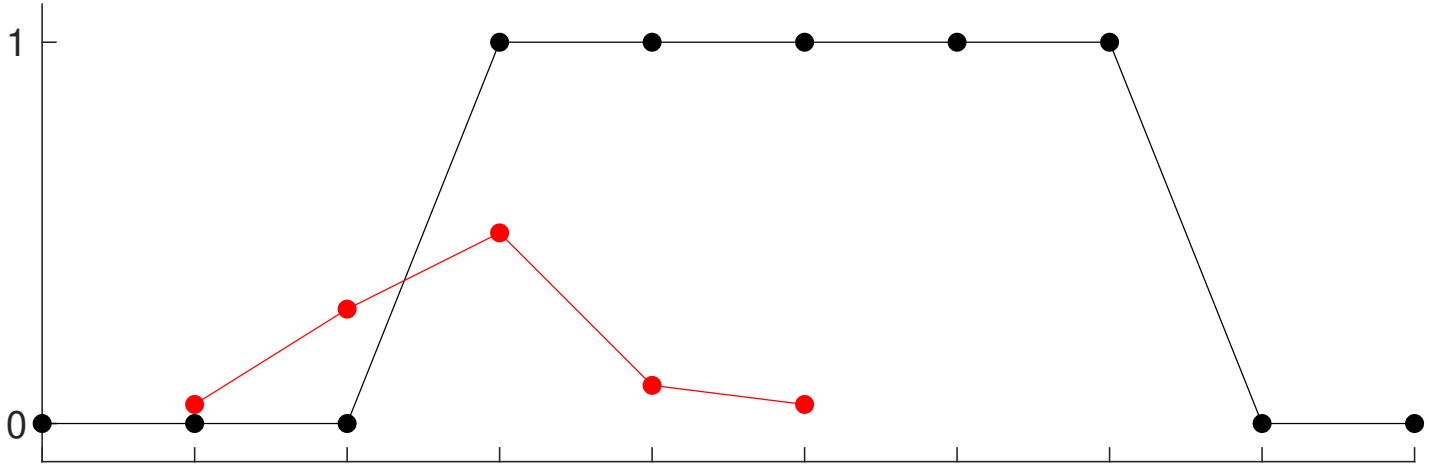
$$\begin{bmatrix} 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\ 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \end{bmatrix}$$

Periodic boundary condition



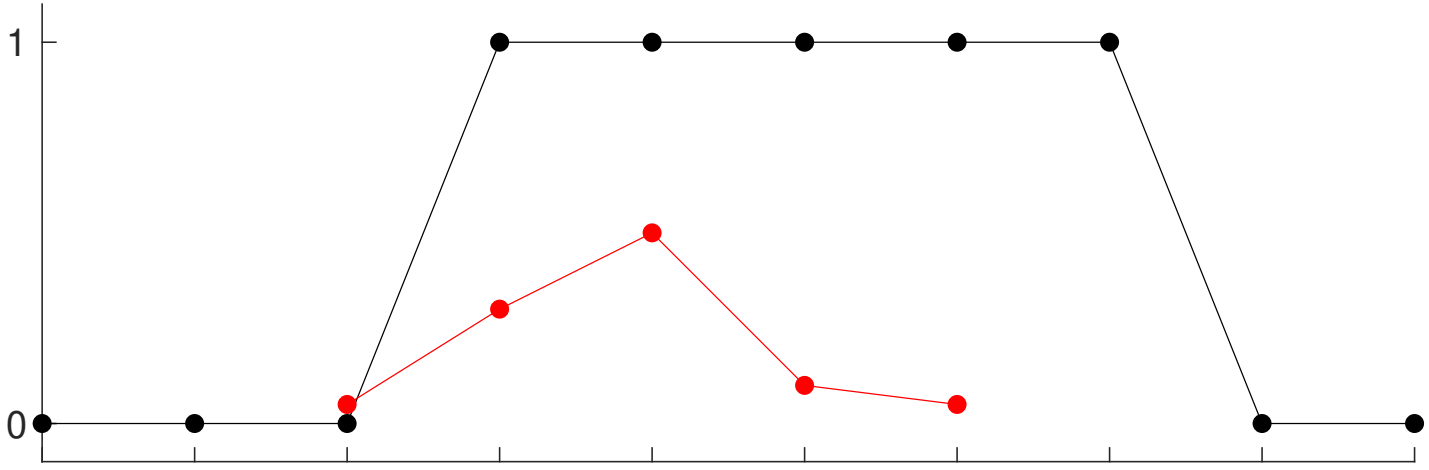
$$\begin{bmatrix} 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\ 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\ 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Periodic boundary condition



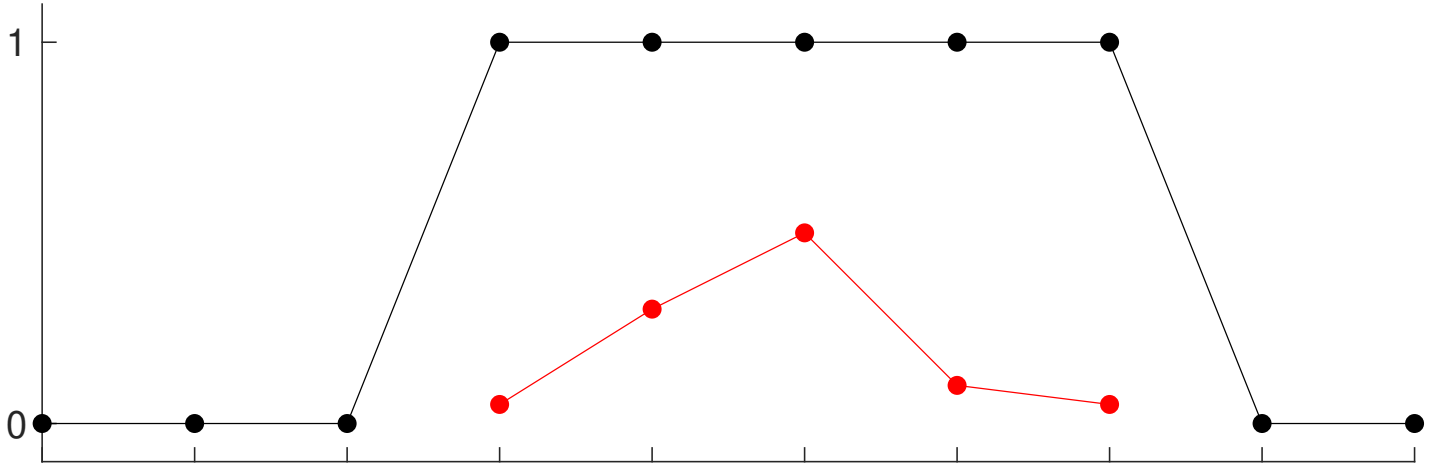
$$\begin{bmatrix} 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\ 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\ 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Periodic boundary condition



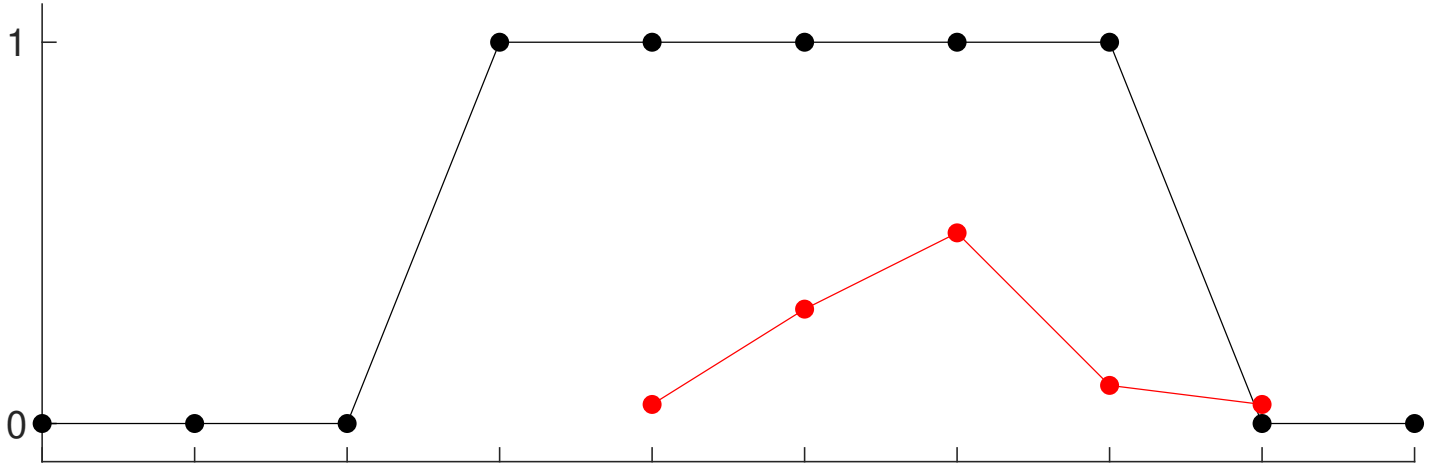
$$
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0
\end{bmatrix}
$$

Periodic boundary condition



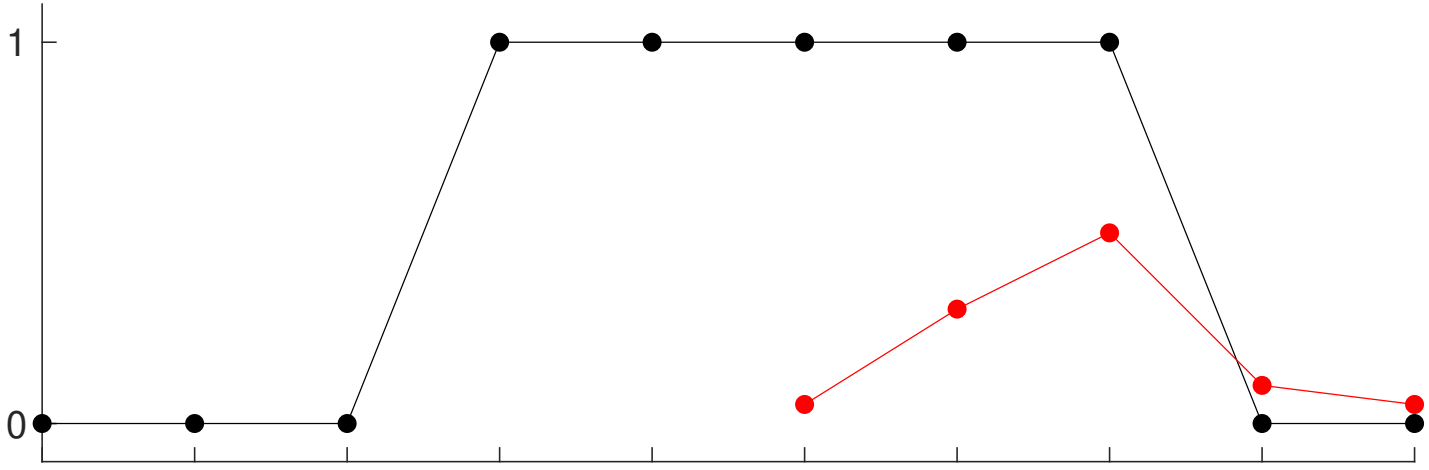
$$
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0
\end{bmatrix}
$$

Periodic boundary condition



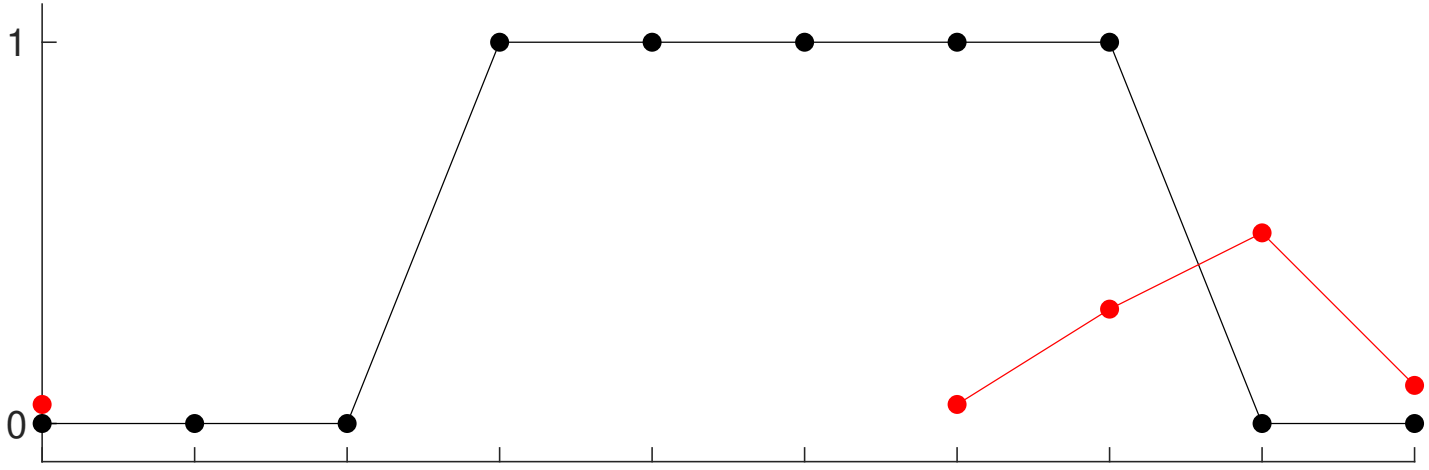
$$\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0
\end{bmatrix}$$

Periodic boundary condition



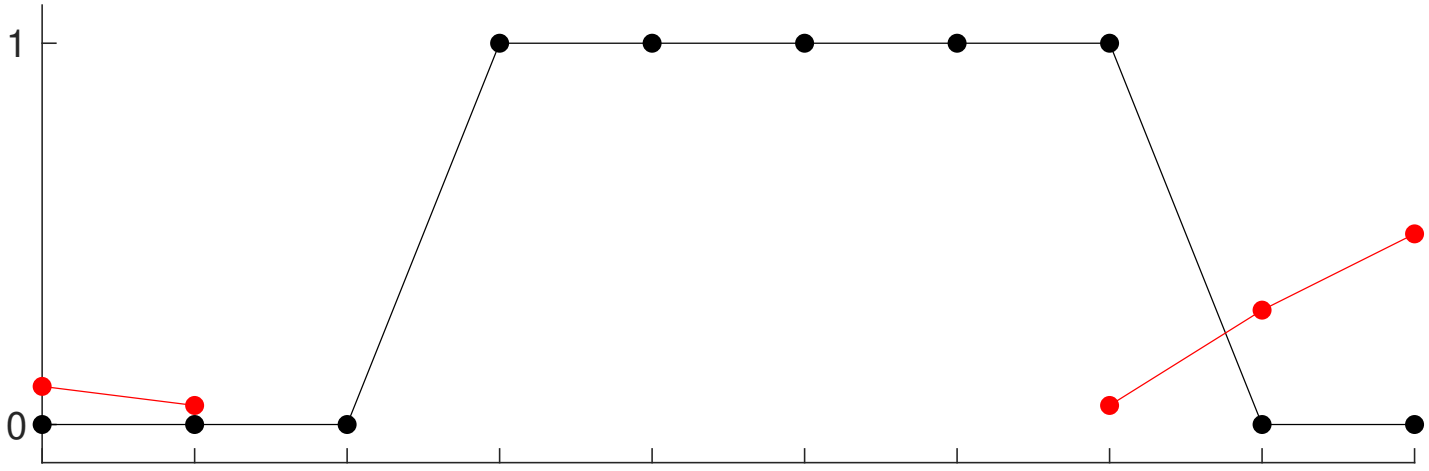
$$
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05
\end{bmatrix}
$$

Periodic boundary condition



$$\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10
\end{bmatrix}$$

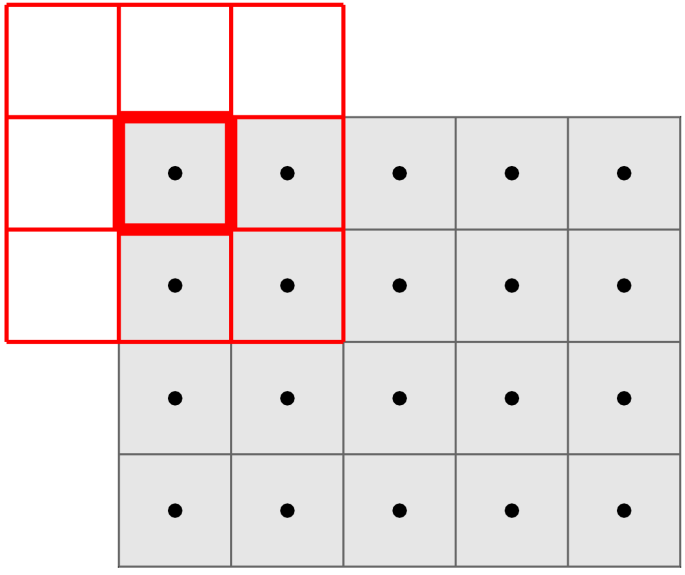
Periodic boundary condition



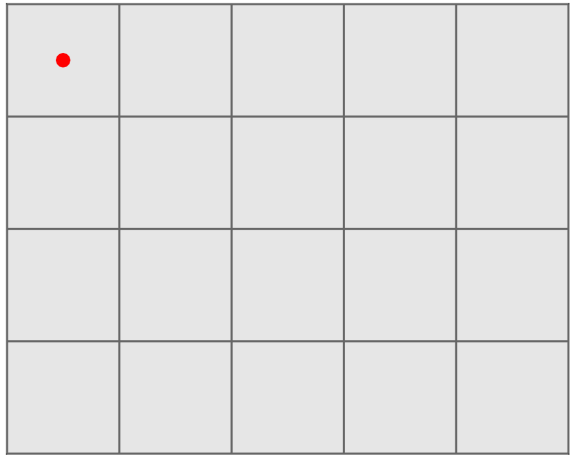
$$\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 \\
0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50
\end{bmatrix}$$

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



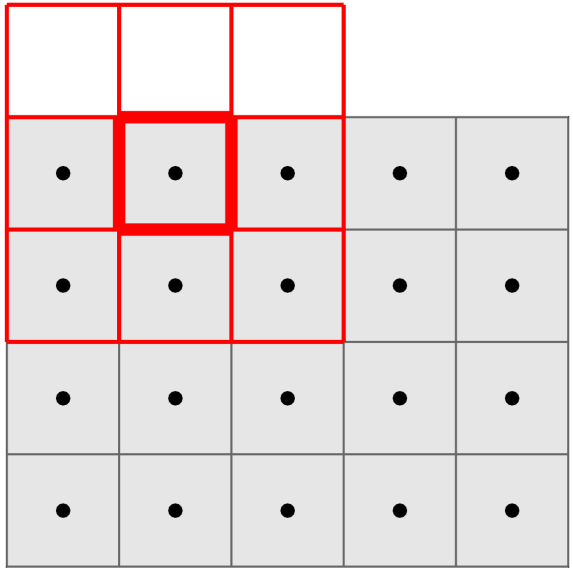
Original image and convolution kernel



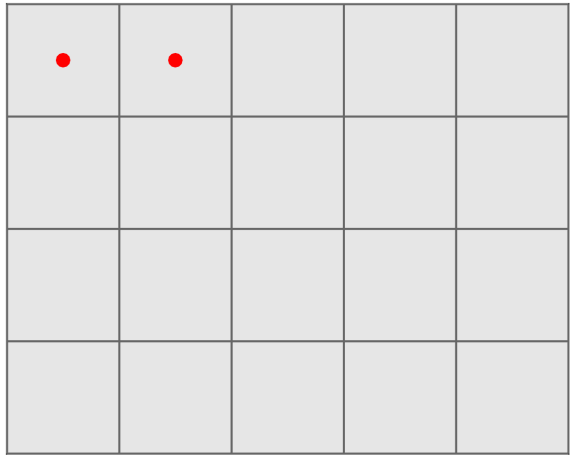
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



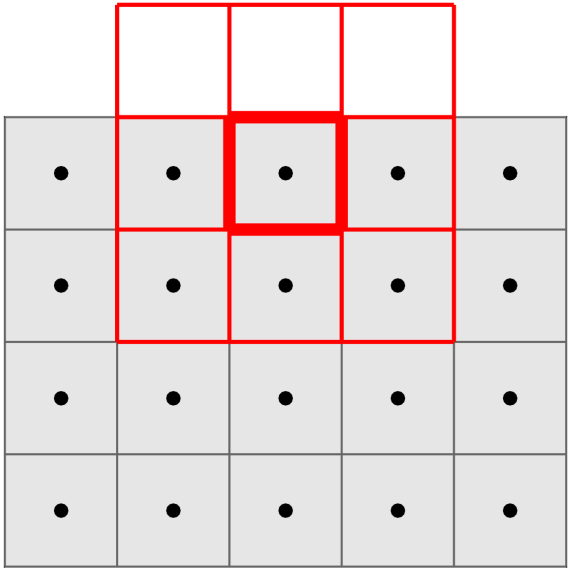
Original image and convolution kernel



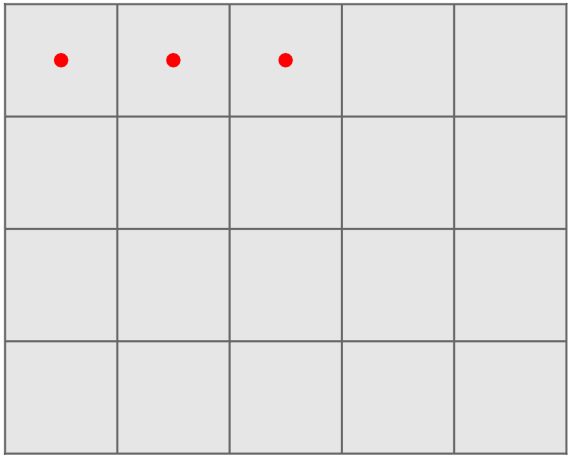
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



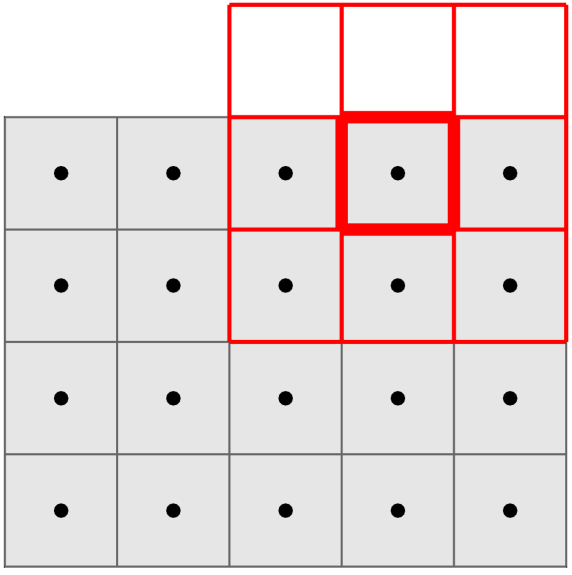
Original image and convolution kernel



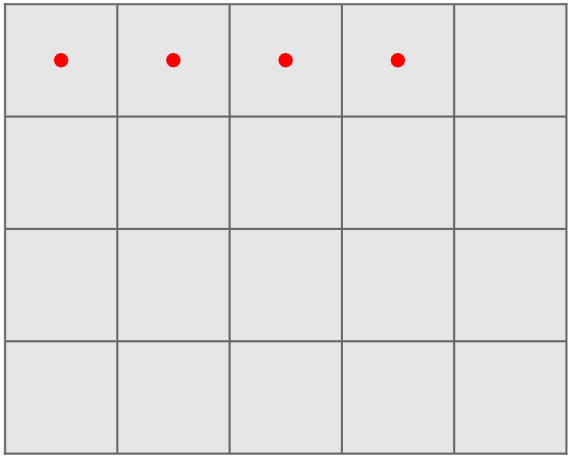
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



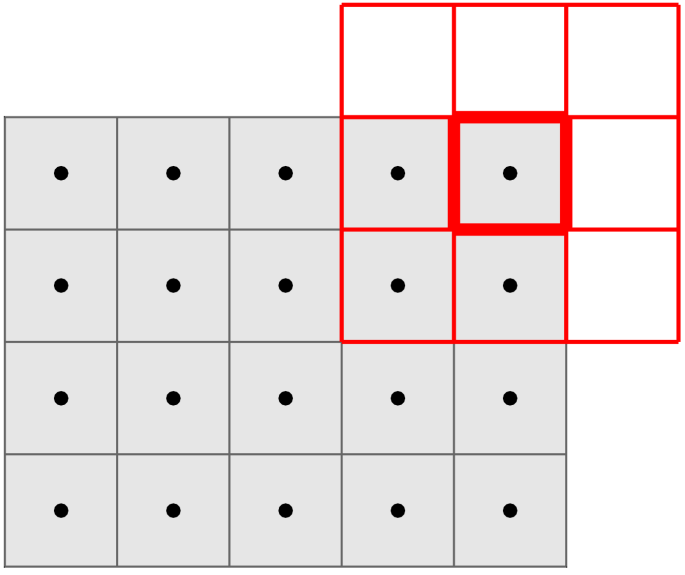
Original image and convolution kernel



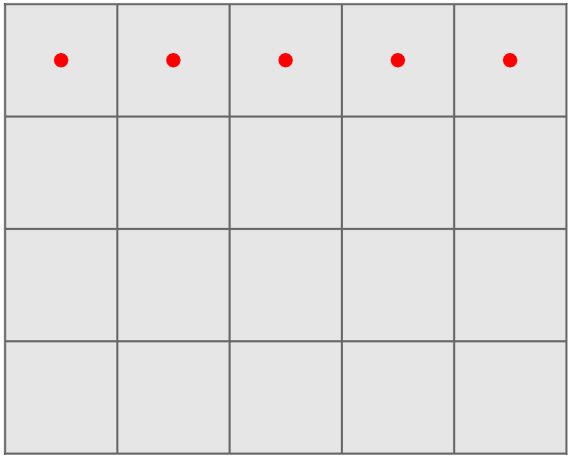
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



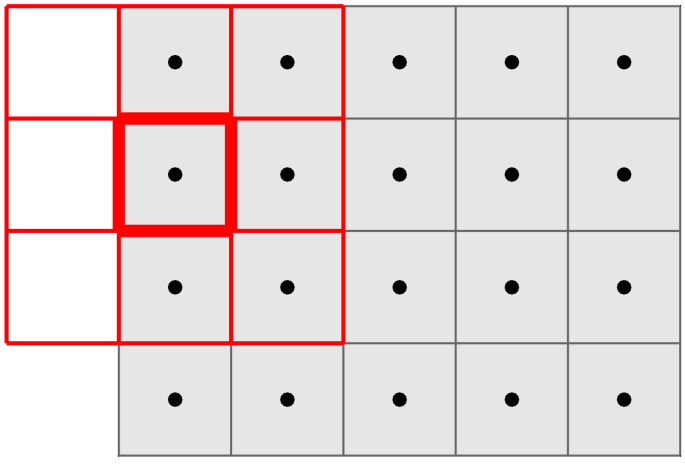
Original image and convolution kernel



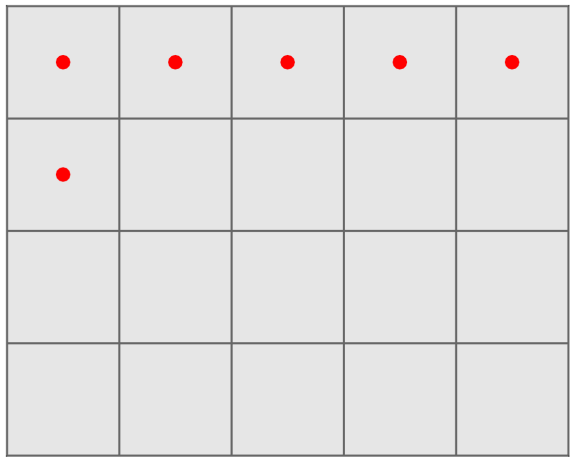
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



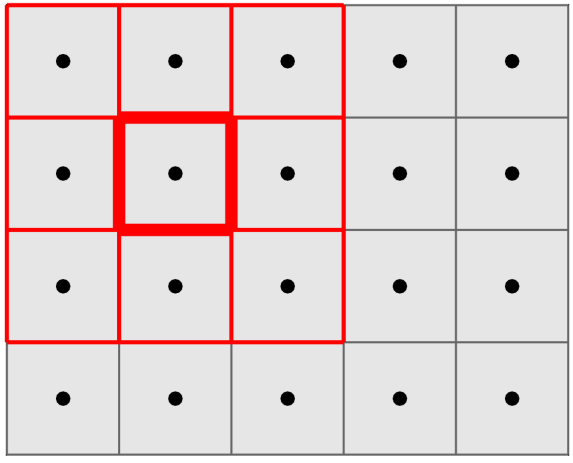
Original image and convolution kernel



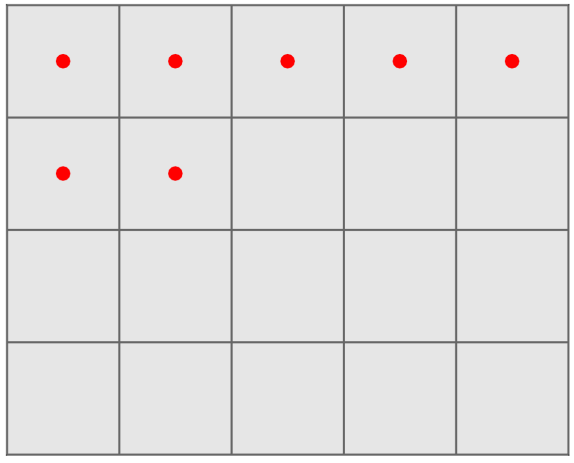
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



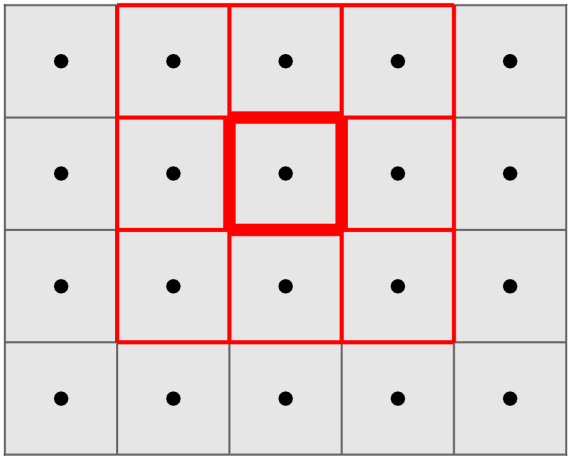
Original image and convolution kernel



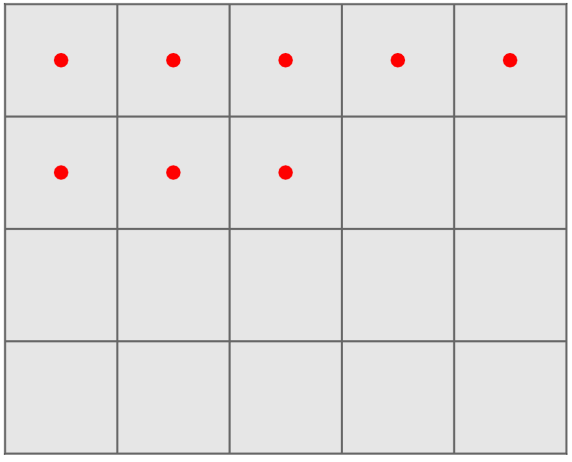
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



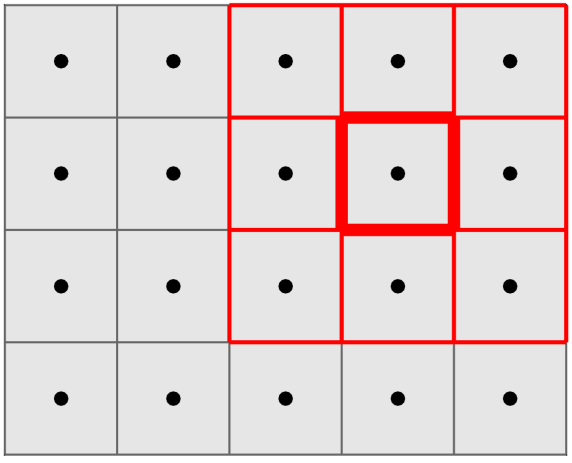
Original image and convolution kernel



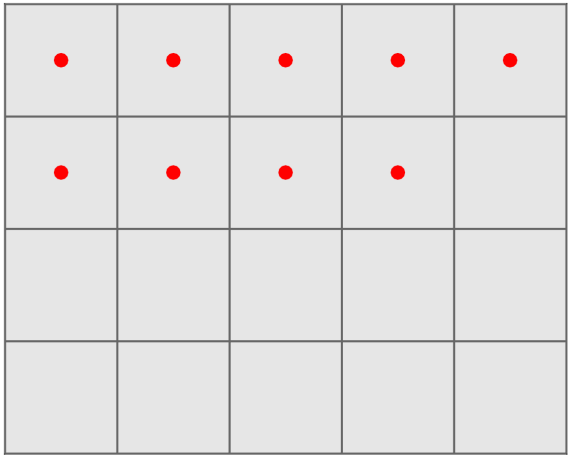
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



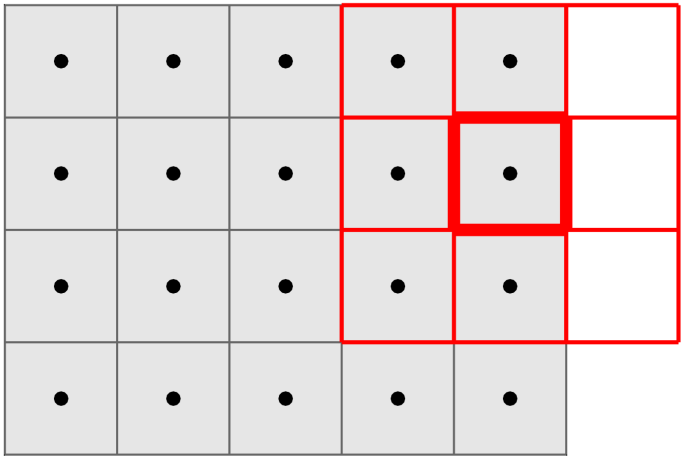
Original image and convolution kernel



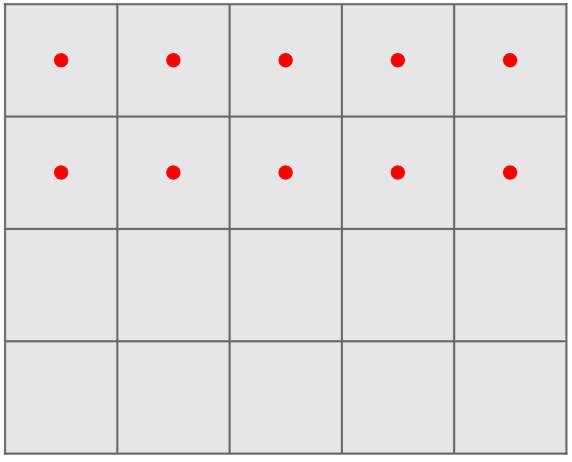
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



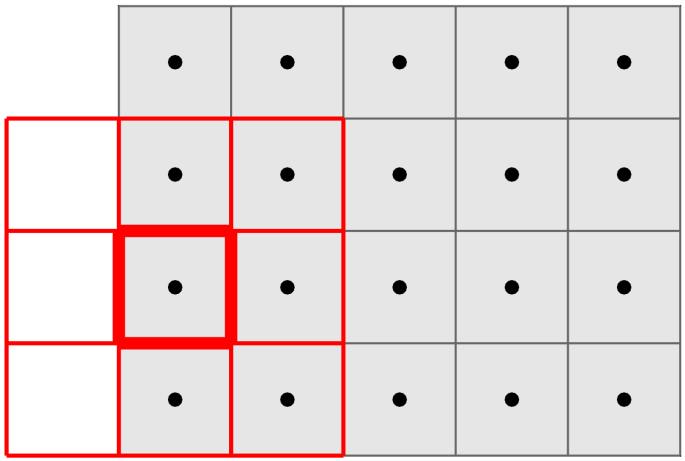
Original image and convolution kernel



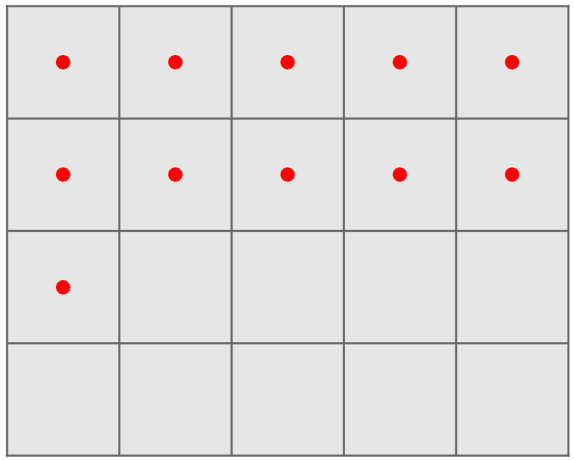
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



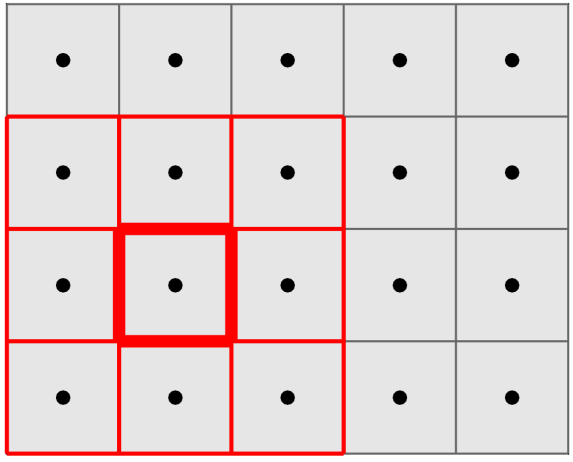
Original image and convolution kernel



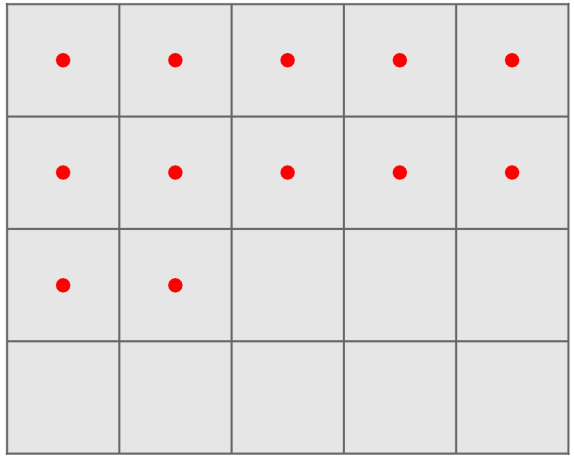
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



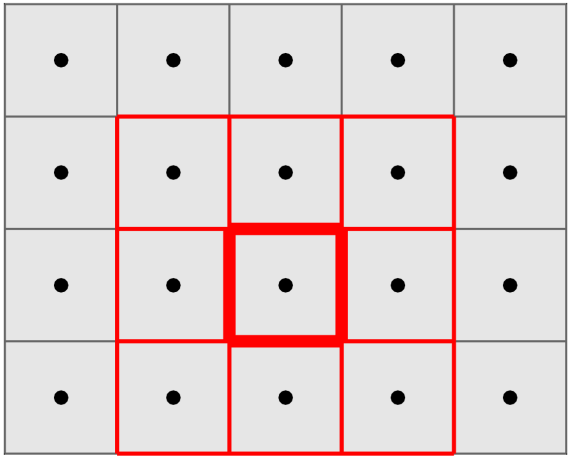
Original image and convolution kernel



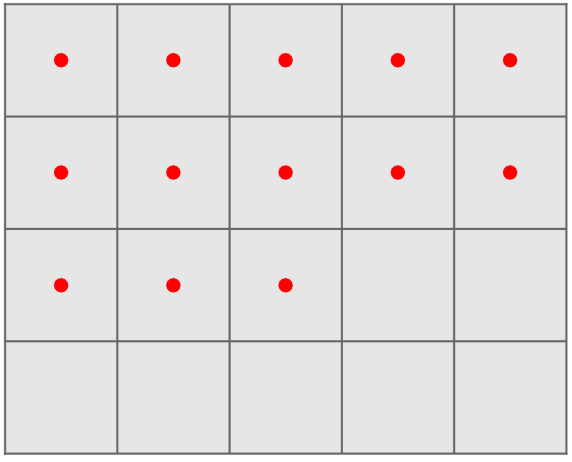
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



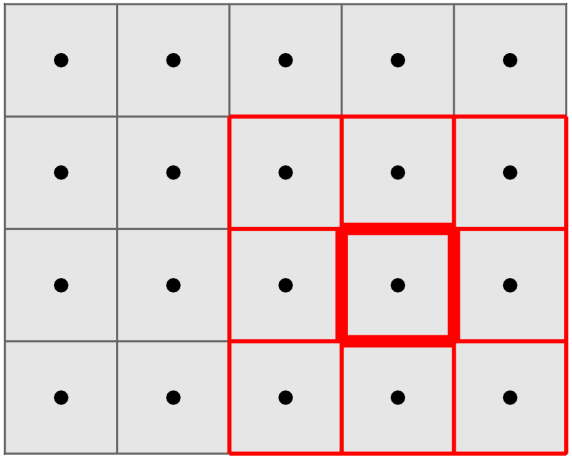
Original image and convolution kernel



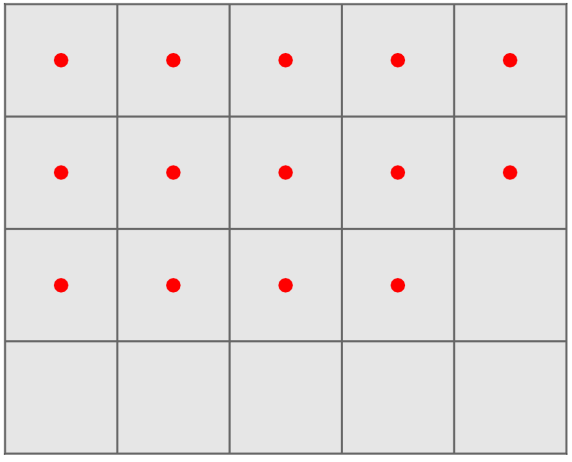
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



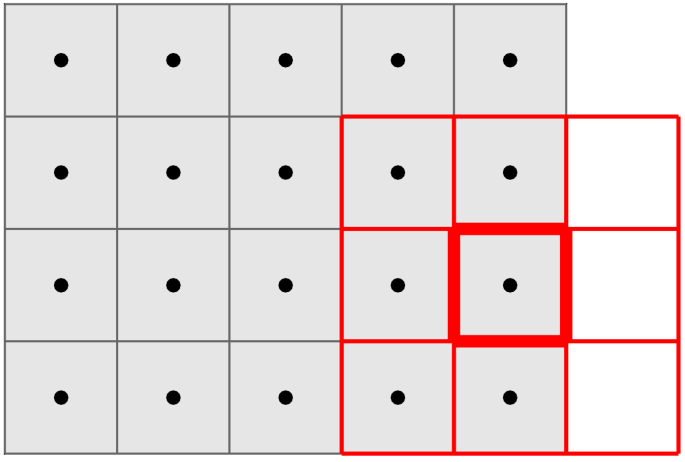
Original image and convolution kernel



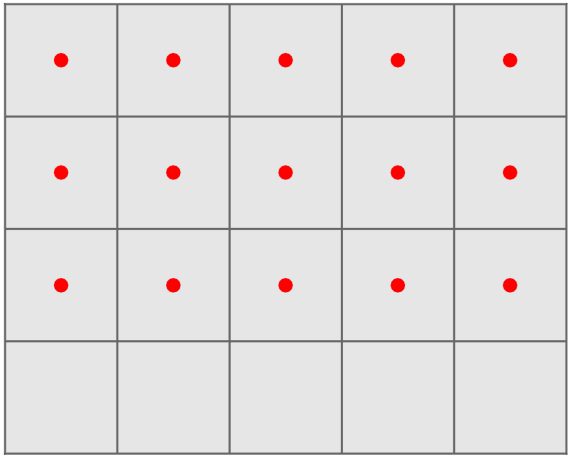
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



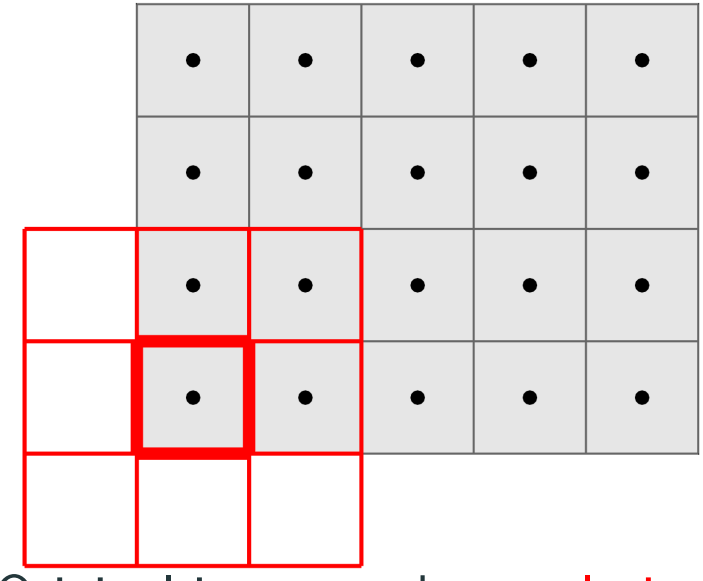
Original image and convolution kernel



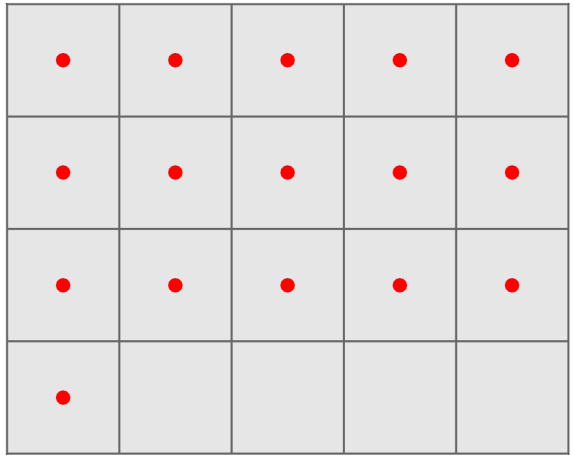
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



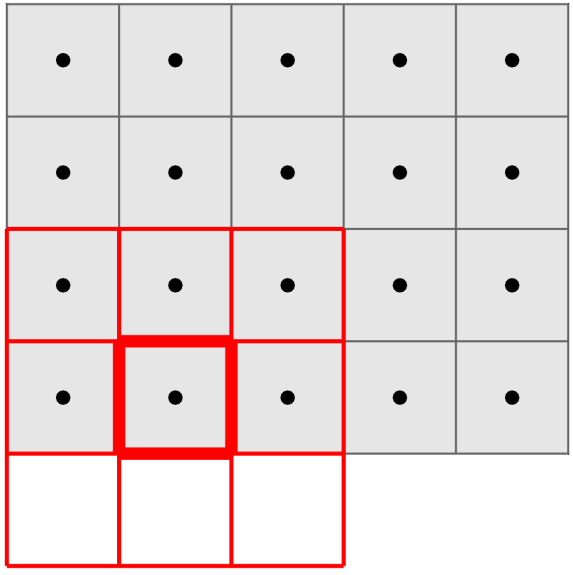
Original image and convolution kernel



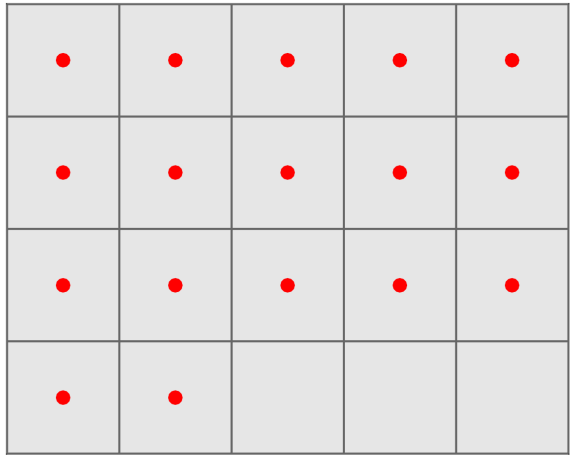
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



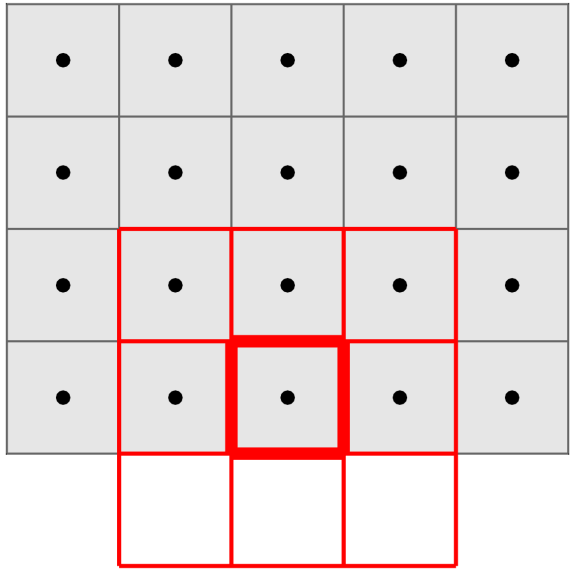
Original image and convolution kernel



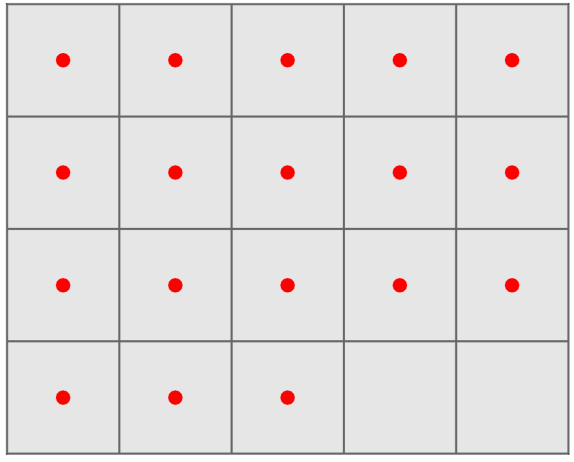
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



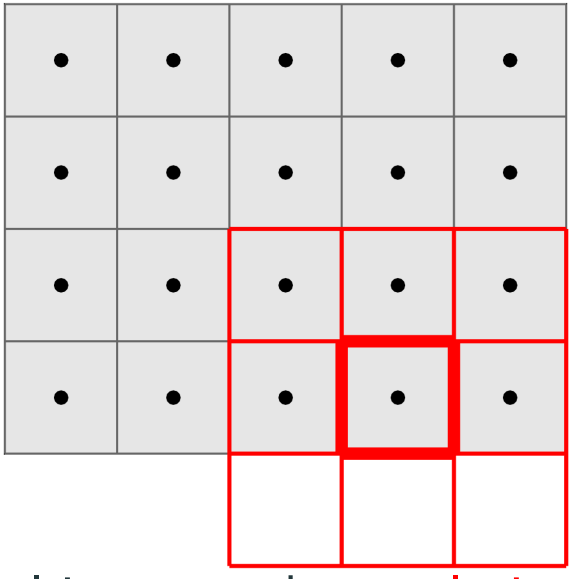
Original image and convolution kernel



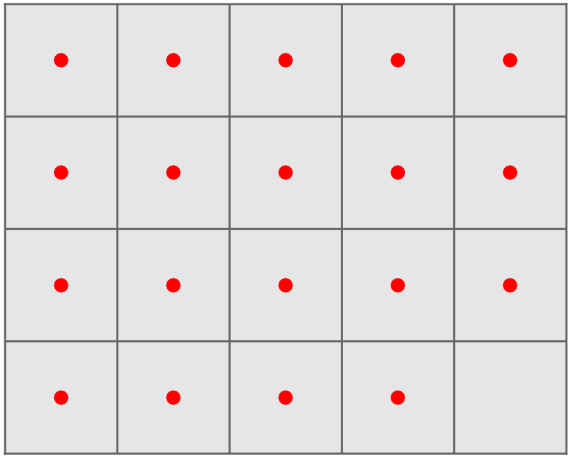
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.



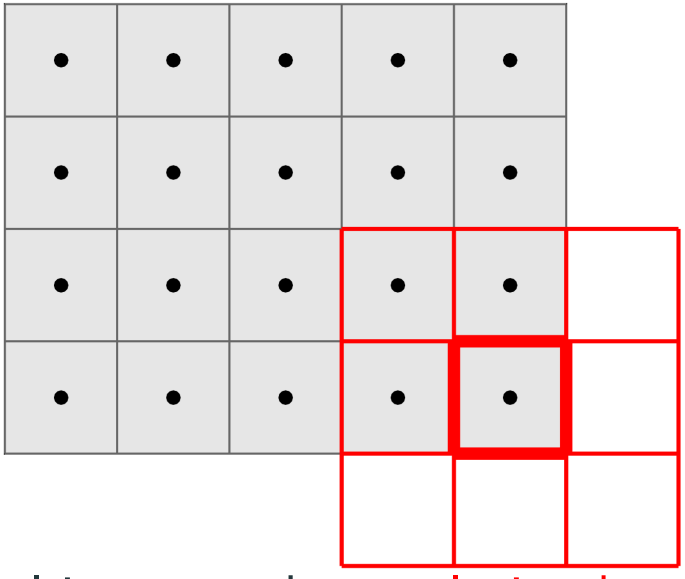
Original image and convolution kernel



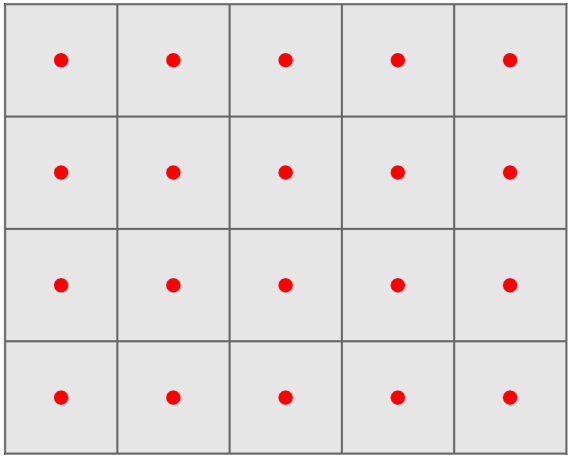
Result of convolution

Two-dimensional convolution step-by-step

The convolution in 2D signals is performed the same way but now we have a 2D convolution kernel.

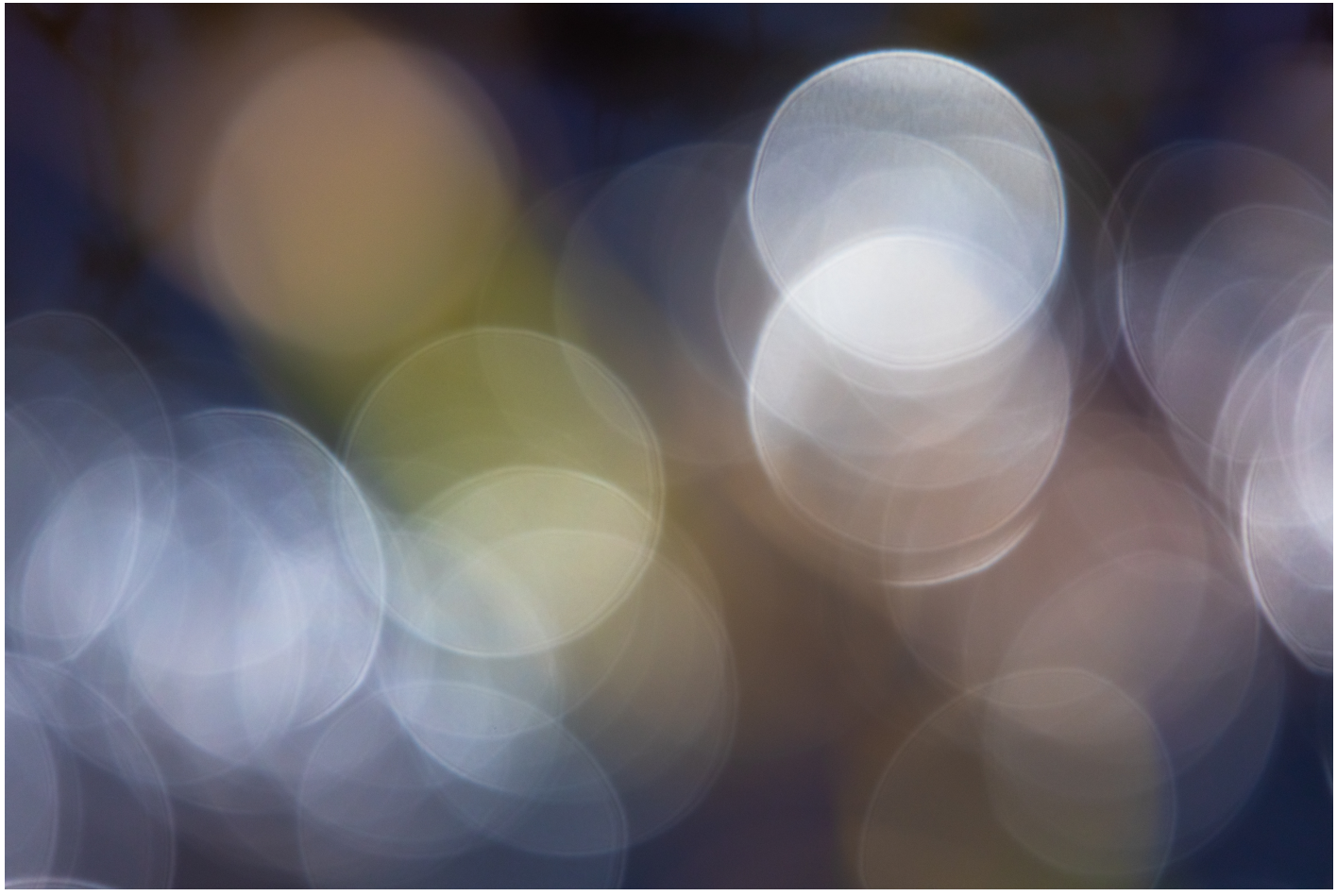


Original image and convolution kernel



Result of convolution

Real world PSF examples (Images by Markus Juvonen)



Real world PSF examples (Images by Markus Juvonen)

Real world PSF examples (Images by Markus Juvonen)

Another note about the convolution kernel

Usual names: convolution kernel, point spread function, convolution window, impulse response, convolution filter

Caution: we will reserve the term *convolution matrix* for the matrix form of the convolution operation A. Unfortunately this term is very often used for the convolution kernel. You should not use it. =)

You can play with 2D convolution kernels for images in GIMP and Photoshop (up to 5x5):

Photoshop: `Filter` $\rightarrow$ `Other` $\rightarrow$ `Custom`

GIMP: `Filters` $\rightarrow$ `Generic` $\rightarrow$ `Convolution Matrix`

(bad naming GIMP! No cookies for you.)

Fun fact: GIMP allows you to play with a few boundary conditions.

What is deconvolution?

Deconvolution

Given the convolution kernel p and a measurement vector m, our task is to find the original signal f assuming that

$$p * f = m.$$

OK, lets try to deconvolve our example ($p \in \mathbb{R}^3$ and $m \in \mathbb{R}^6$).

Naive solution of the deconvolution problem 1/3

For that, we can write the system of equations and start with the first and last equations in our system (remember: we have p and m and we want to find f)

$$\begin{aligned}
m_1 &= p_{-1} f_1 \\
m_2 &= p_0 f_1 + p_{-1} f_2 \\
m_3 &= p_1 f_1 + p_0 f_2 + p_{-1} f_3 \\
m_4 &= p_1 f_2 + p_0 f_3 + p_{-1} f_4 \\
m_5 &= p_1 f_3 + p_0 f_4 \\
m_6 &= p_1 f_4
\end{aligned}$$

We can readily solve for the unknowns f_1 and f_4 (if $p_{-1} \neq 0 \neq p_1$)

$$f_1 = \frac{m_1}{p_{-1}}, \qquad f_4 = \frac{m_6}{p_1}.$$

Nice!

Naive solution of the deconvolution problem 2/3

We can now solve for f_2 using the second equation in the system

$$\begin{aligned} m_1 &= p_{-1} f_1 \\ m_2 &= p_0 f_1 + p_{-1} f_2 \\ m_3 &= p_1 f_1 + p_0 f_2 + p_{-1} f_3 \\ m_4 &= p_1 f_2 + p_0 f_3 + p_{-1} f_4 \\ m_5 &= p_1 f_3 + p_0 f_4 \\ m_6 &= p_1 f_4 \end{aligned}$$

Using the previous result $f_1 = \frac{m_1}{p_{-1}}$, we get

$$f_2 = \frac{m_2 - p_0 f_1}{p_{-1}} = \frac{m_2 - \frac{p_0 m_1}{p_{-1}}}{p_{-1}} = \frac{m_2}{p_{-1}} - \frac{p_0 m_1}{p_{-1}^2}.$$

Naive solution of the deconvolution problem 3/3

The third equation $m_3 = p_1 f_1 + p_0 f_2 + p_{-1} f_3$ in our system leads to

$$\begin{aligned} f_3 &= \frac{m_3 - p_1 f_1 - p_0 f_2}{p_{-1}} = \frac{m_3 - \frac{p_1 m_1}{p_{-1}} - \frac{p_0 m_2}{p_{-1}} + \frac{p_0^2 m_1}{p_{-1}^2}}{p_{-1}} \\ &= \frac{m_3}{p_{-1}} - \frac{p_1 m_1}{p_{-1}^2} - \frac{p_0 m_2}{p_{-1}^2} + \frac{p_0^2 m_1}{p_{-1}^3} \end{aligned}$$

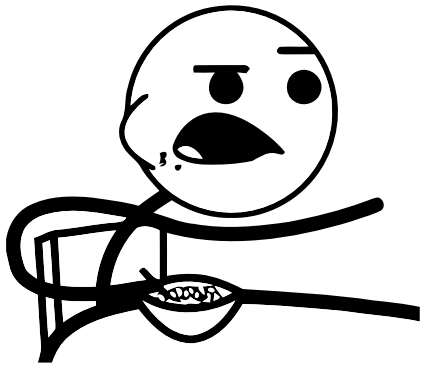
Remember that we already had the solutions

$$f_1 = \frac{m_1}{p_{-1}}, \qquad f_2 = \frac{m_2}{p_{-1}} - \frac{p_0 m_1}{p_{-1}^2}, \qquad f_4 = \frac{m_6}{p_1},$$

so we are all done!

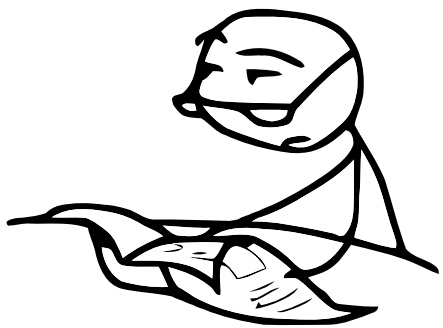
Naive solution of the deconvolution problem

Is this the whole story of deconvolution? So simple?



Naive solution of the deconvolution problem

Well, it is called a **naive solution** for a reason, don't you think?



Random measurement noise destroys our solution

In practice, the measurement we get from our device contains *noise*:

$$\widetilde{m} = \overbrace{p * f}^{m_{true}} + \epsilon = m_{true} + \epsilon,$$

where $\epsilon \in \mathbb{R}^6$ adds a random error to each of the six components.

Let's try to recover f_1 using the equation we derived above:

$$\widetilde{f}_1 = \frac{\widetilde{m}_1}{p_{-1}} = \frac{m_{1,true} + \epsilon_1}{p_{-1}} = \overbrace{\frac{m_{1,true}}{p_{-1}}}^{f_{1,true}} + \frac{\epsilon_1}{p_{-1}}.$$

Now the number $1/p_{-1}$ is large if $0 < |p_{-1}| \ll 1$. This means that

$$\widetilde{f}_1 = f_{1,true} + \underbrace{\frac{\epsilon_1}{p_{-1}}}_{\text{big error!}}$$

Random measurement noise destroys our solution

Let's try to recover f_2 using the equation we derived above:

$$\widetilde{f}_2 = \frac{\widetilde{m}_2}{p_{-1}} - \frac{p_0 \widetilde{m}_1}{p_{-1}^2} = \frac{m_{2,true} + \epsilon_2}{p_{-1}} - \frac{p_0 (m_{1,true} + \epsilon_1)}{p_{-1}^2}$$

$$= \overbrace{\frac{m_{2,true}}{p_{-1}} - \frac{p_0 m_{1,true}}{p_{-1}^2}}^{f_{2,true}} + \underbrace{\frac{\epsilon_2}{p_{-1}}}_{\text{big error}} - \underbrace{\frac{p_0 \epsilon_1}{p_{-1}^2}}_{\text{huge error}}$$

For f_3 it gets even worse (exercise). We can conclude that random noise is amplified in this simple deconvolution attempt.

What if we have inaccurate information?

The main problem with deconvolution is instability:

Instability

Small perturbations on the datum (blurred signal) can cause large perturbations on the solution (deconvolved signal)

Instability depicted

Noise happens. In applications, a deconvolution problem (e.g., getting rid of blur from images) is always a deconvolution and denoising problem.

Blurred image (without noise)

Deconvolved image

Instability depicted

Noise happens. In applications, a deconvolution problem (e.g., getting rid of blur from images) is always a deconvolution and denoising problem.

Blurred image (noise level 10^{-10})

Deconvolved image

Instability depicted

Noise happens. In applications, a deconvolution problem (e.g., getting rid of blur from images) is always a deconvolution and denoising problem.

Blurred image (noise level 10^{-7})

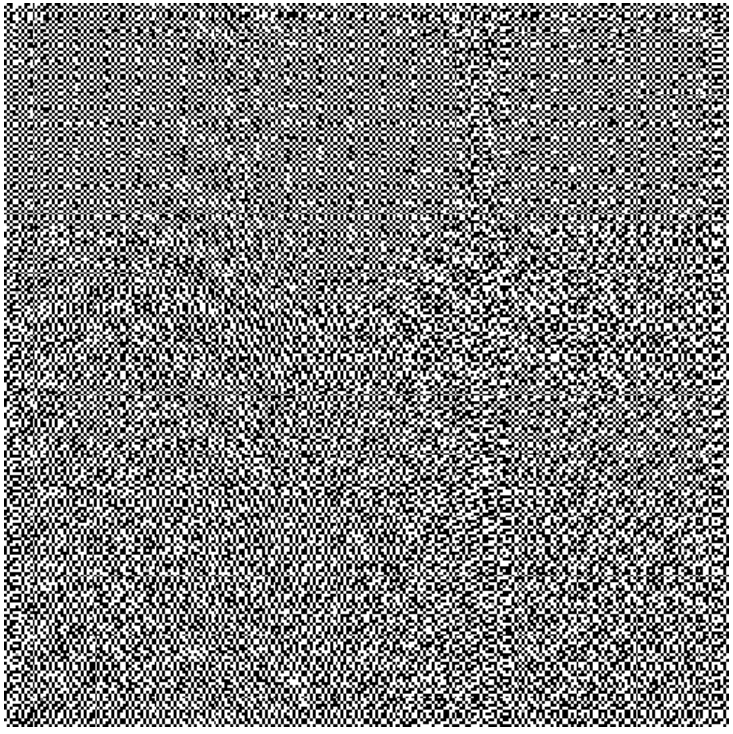
Deconvolved image

Instability depicted

Noise happens. In applications, a deconvolution problem (e.g., getting rid of blur from images) is always a deconvolution and denoising problem.

Blurred image (noise level 10^{-3})

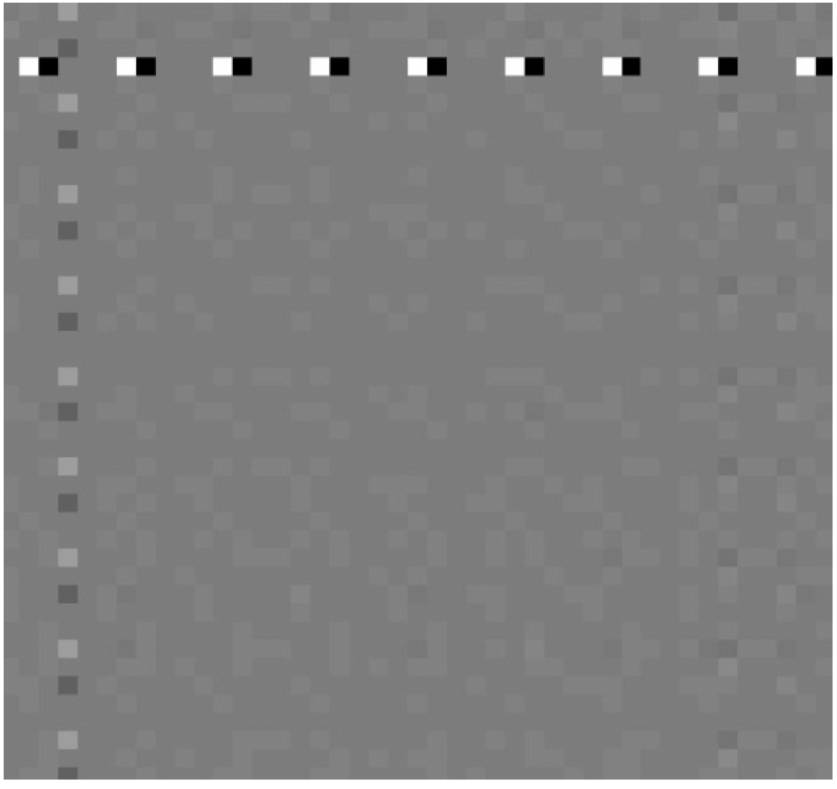
Deconvolved image



Naive Inversion

If we just invert the convolution matrix, the result is just numerical garbage

Object Naive solution



Data (blurred and noisy photo)

Main aims of the course

This course will provide you the following mathematical tools:

A general formulation for the deconvolution problem: this will allow us to treat many cases together, causing almost no effort in moving from 1D to 2D

A first approach to (linear) inverse problems: this will provide a framework to study instability and in general ill-posedness of a mathematical problem

A solid grasp of the regularization theory of inverse problems: which consists in a robust strategy to overcome the instability of the problem by incorporating (or promoting) additional features the solution is expected to possess.

Regularized inversion

We will study several regularization techniques, learning how to implement them. With appropriate regularization, the blurred image can be sharpened to some extent.

Reconstruction (Tikhonov)

Data (blurred and noisy photo)

Regularized inversion

We will study several regularization techniques, learning how to implement them. With appropriate regularization, the blurred image can be sharpened to some extent.

Reconstruction (TV)

Data (blurred and noisy photo)

Regularized inversion

We will study several regularization techniques, learning how to implement them. With appropriate regularization, the blurred image can be sharpened to some extent.

Reconstruction (TGV)

Data (blurred and noisy photo)

Deconvolution can be hard with a matrix as well

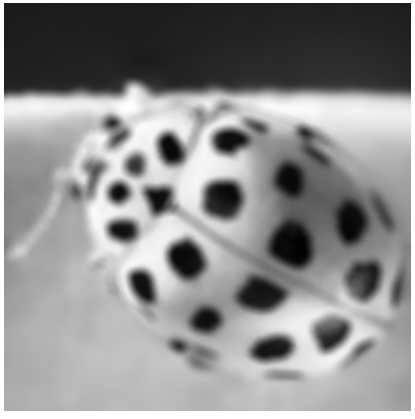
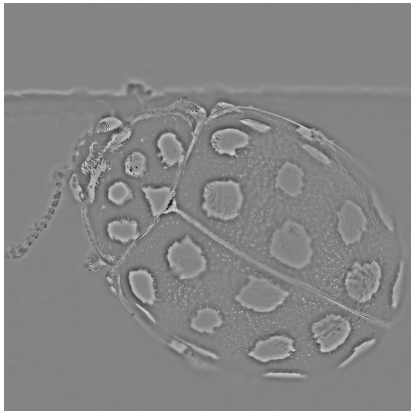
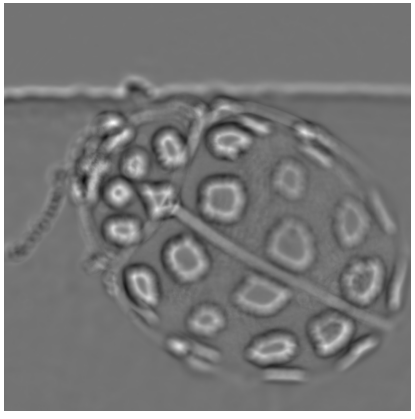
Watch this video

Singular Value Decomposition: the simplest case 

Simplest 2D deconvolution trick: "unsharp masking"

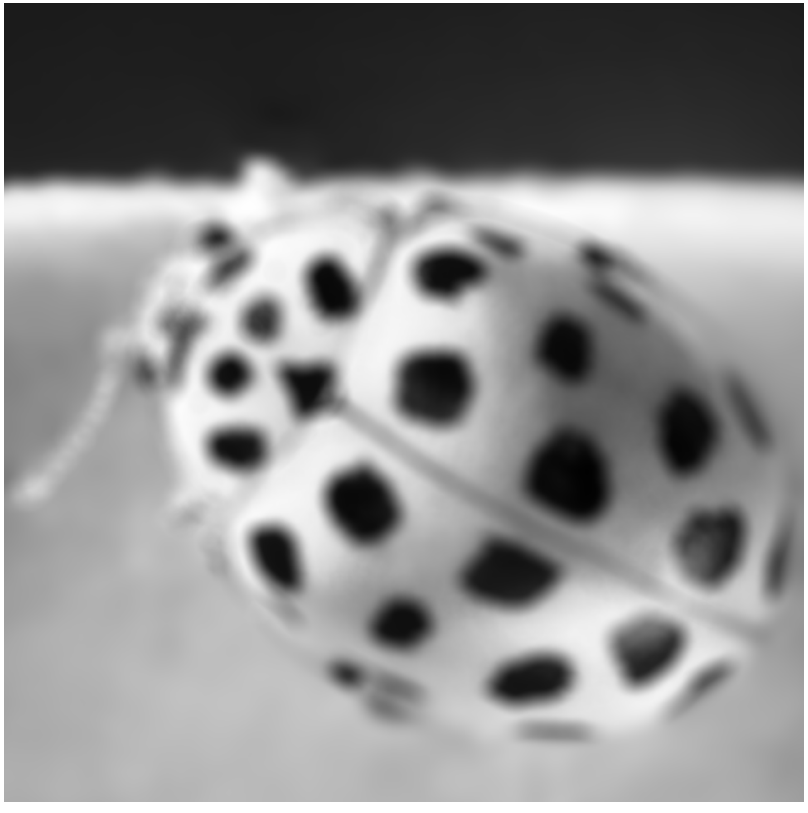
$$
\begin{aligned}
f &= m - (m - f) \\
&\approx m - p * (m - f) \\
&= m - (p * m - p * f) \\
&= m + (m - p * m)
\end{aligned}
$$

Original image f | Data $m = p * f$ | Difference $m - f$ | Blur-diff $m - p * m$

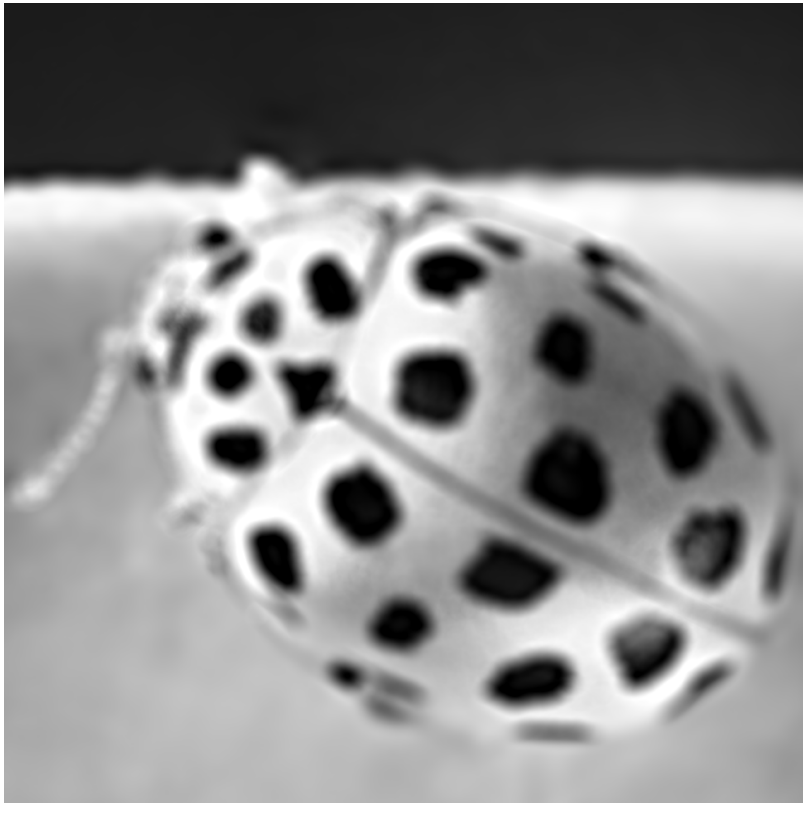





Basic unsharp masking looks like this

Data: blurred image m



Unsharp-masked image $m + (m - p * m)$



Unsharp masking often has extra tweaks (take a look at Photoshop or GIMP implementations)

$$
\begin{aligned}
& m + (m - p * m) \approx f \\
& m + (m - p' * m) && \text{new PSF} \\
& m + \alpha(m - p' * m) && \text{strength } \alpha > 0 \\
& m + \alpha Q_\tau(m - p' * m) && \text{threshold } \tau \geq 0
\end{aligned}
$$

The nonlinear thresholding operator applied to image g is

$$
(Q_\tau g)_{ij} = \begin{cases} 0 & \text{if } |g_{ij}| < \tau, \\ g_{ij} & \text{otherwise.} \end{cases}
$$

Check this link 

for more details (although the thresholding strategy is slightly different)

You can play with unsharp masking in Photoshop or GIMP:

Photoshop: `Filter` $\rightarrow$ `Sharpen` $\rightarrow$ `Unsharp Mask`

GIMP: `Filters` $\rightarrow$ `Enhance` $\rightarrow$ `Sharpen (Unsharp mask)`