Image analysis fundamentals

Fall 2024

Aaron Ponti

v1.8

Goals of the course

- Discuss some fundamental theory of image processing and (data) analysis.
- Put that theory in action using **Python**.

scikit-image
iaf
numpy
scipy
pandas
Matplotlib
…

https://python.org
https://docs.conda.io/en/latest/miniconda.html
[open source]

- Analyze the data from some fictitious drug discovery study.
- Solve a **graded** homework.

https://ia-res.ethz.ch/courses/iaf

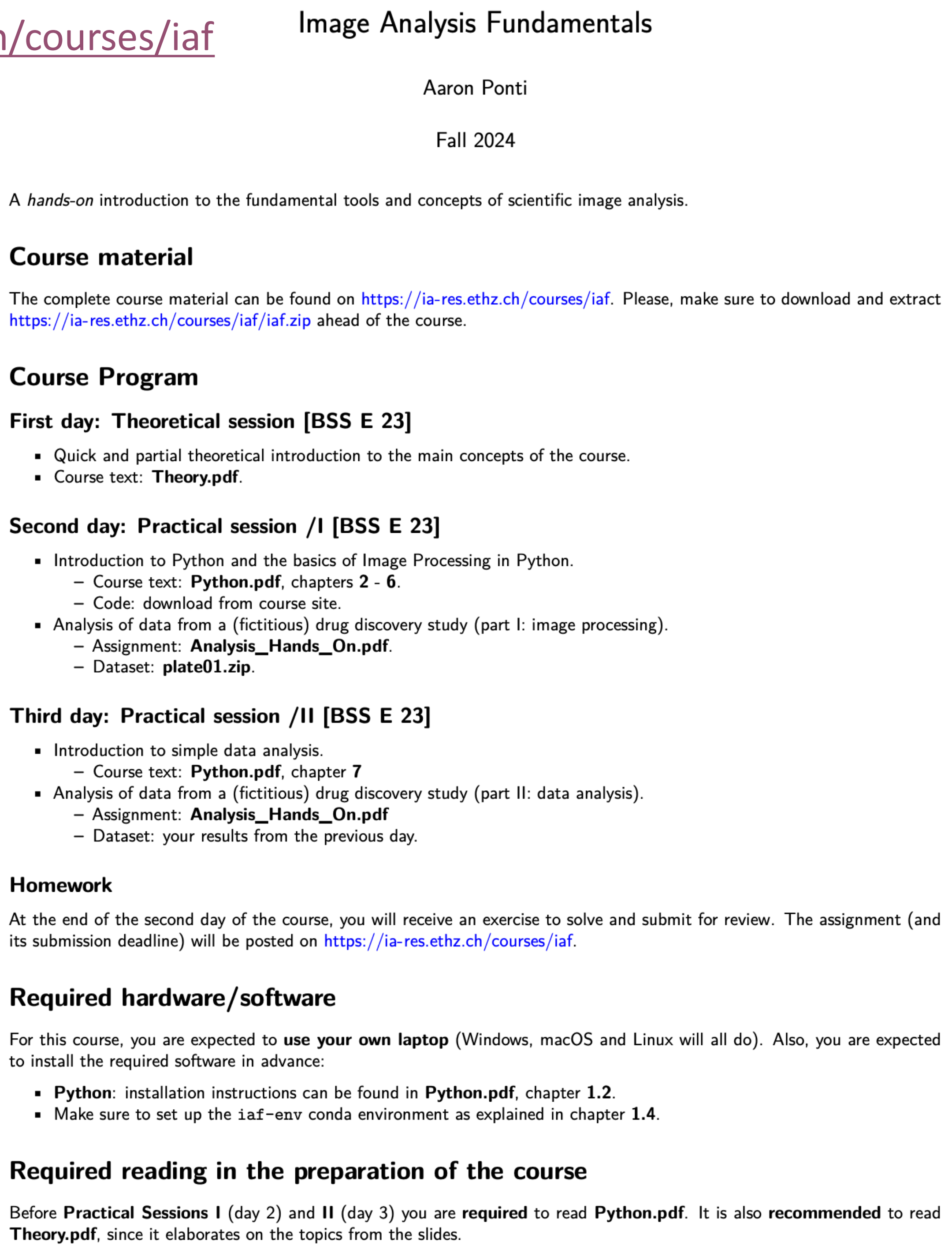


Image Analysis Fundamentals

Aaron Ponti

Fall 2024

A *hands-on* introduction to the fundamental tools and concepts of scientific image analysis.

Course material

The complete course material can be found on https://ia-res.ethz.ch/courses/iaf. Please, make sure to download and extract https://ia-res.ethz.ch/courses/iaf/iaf.zip ahead of the course.

Course Program

First day: Theoretical session [BSS E 23]

- Quick and partial theoretical introduction to the main concepts of the course.
- Course text: **Theory.pdf**.

Second day: Practical session /I [BSS E 23]

- Introduction to Python and the basics of Image Processing in Python.
  - Course text: **Python.pdf**, chapters **2** - **6**.
  - Code: download from course site.
- Analysis of data from a (fictitious) drug discovery study (part I: image processing).
  - Assignment: **Analysis_Hands_On.pdf**.
  - Dataset: **plate01.zip**.

Third day: Practical session /II [BSS E 23]

- Introduction to simple data analysis.
  - Course text: **Python.pdf**, chapter **7**
- Analysis of data from a (fictitious) drug discovery study (part II: data analysis).
  - Assignment: **Analysis_Hands_On.pdf**
  - Dataset: your results from the previous day.

Homework

At the end of the second day of the course, you will receive an exercise to solve and submit for review. The assignment (and its submission deadline) will be posted on https://ia-res.ethz.ch/courses/iaf.

Required hardware/software

For this course, you are expected to **use your own laptop** (Windows, macOS and Linux will all do). Also, you are expected to install the required software in advance:

- **Python**: installation instructions can be found in **Python.pdf**, chapter **1.2**.
- Make sure to set up the `iaf-env` conda environment as explained in chapter **1.4**.

Required reading in the preparation of the course

Before **Practical Sessions I** (day 2) and **II** (day 3) you are **required** to read **Python.pdf**. It is also **recommended** to read **Theory.pdf**, since it elaborates on the topics from the slides.

Introduction

Digital image processing

"**Digital image processing** is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image.

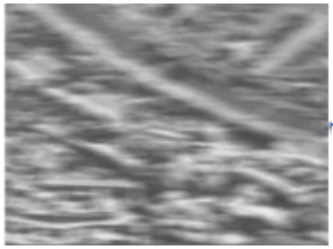
Source: https://en.wikipedia.org/wiki/Digital_image_processing

Input

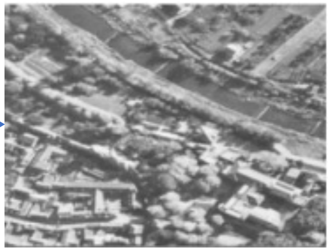
Output

Corrected or enhanced

(I)



$$\begin{tikzcd} \text{Input} \arrow[r] & \boxed{\text{IP}} \arrow[r] & \text{Output} \end{tikzcd}$$



(II)

$$\begin{tikzcd} \text{Input} \arrow[r] & \boxed{\text{IP}} \arrow[r] & \begin{array}{l}\text{Characteristics}\\ \text{Parameters}\\ \text{Measurements}\end{array} \arrow[r] & \begin{array}{l}\text{Analysis}\\ \text{Decision}\\ \text{Action}\end{array} \end{tikzcd}$$

Quantitative image analysis

Images

↓

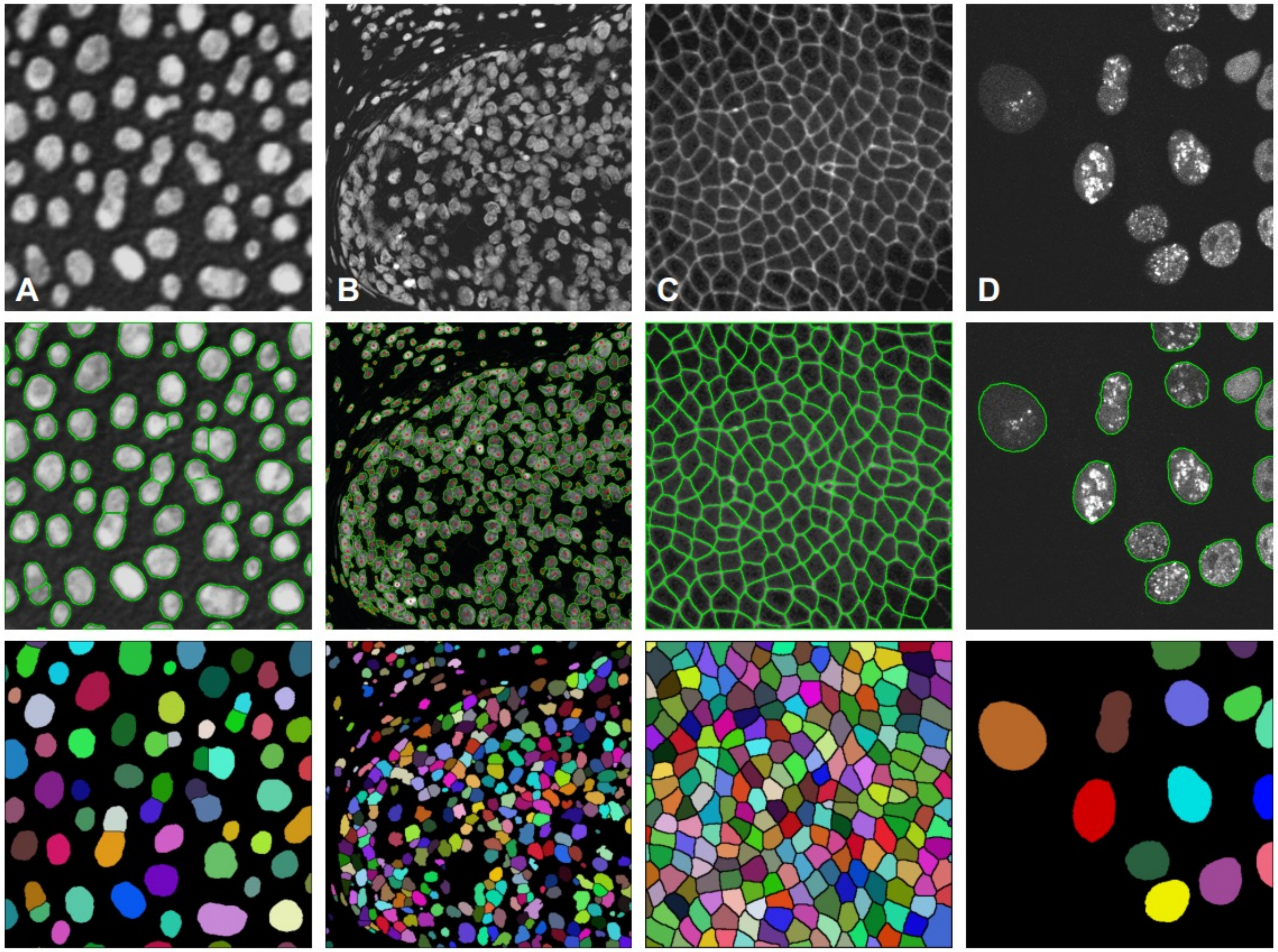
Objects

↓

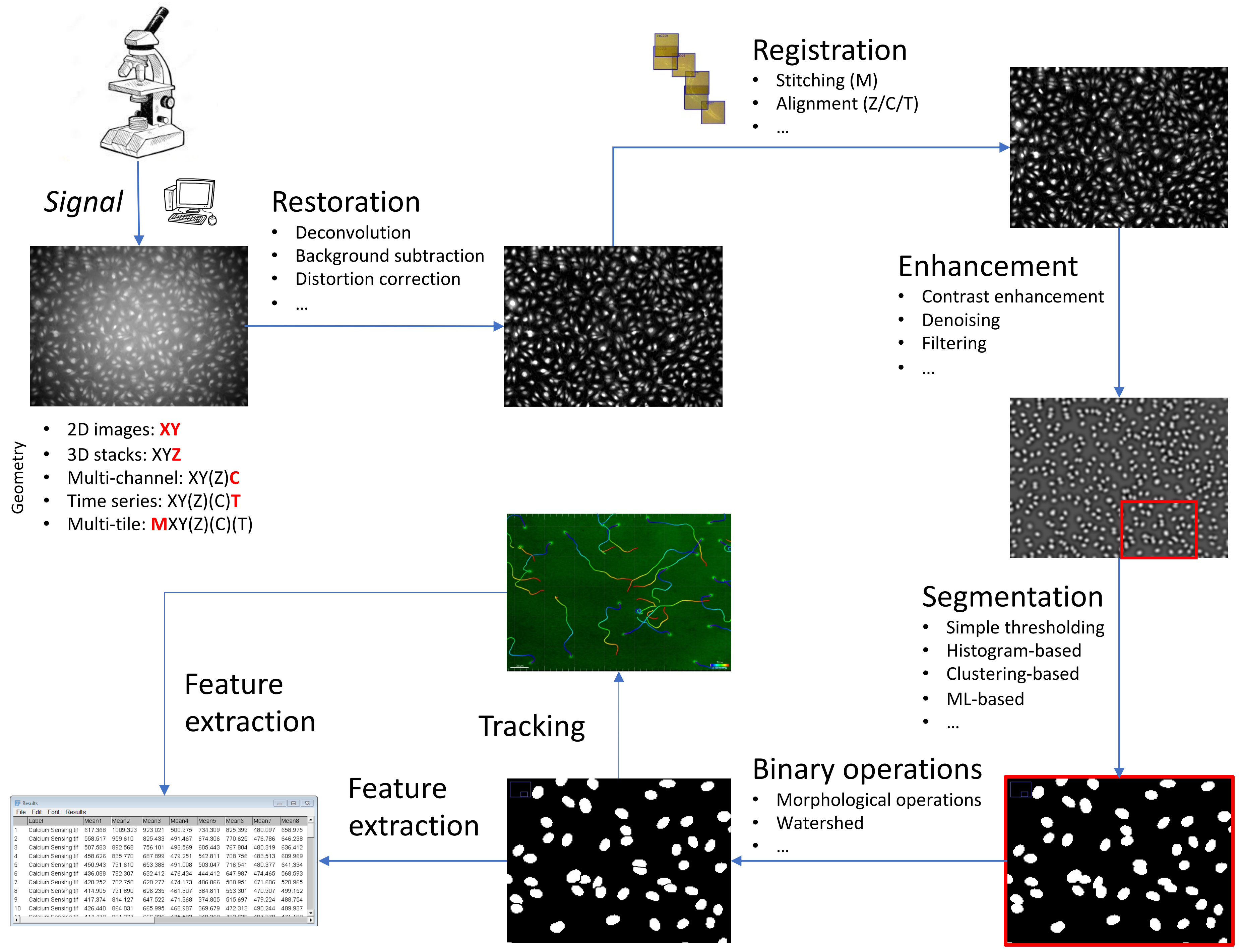
Features

↓

...

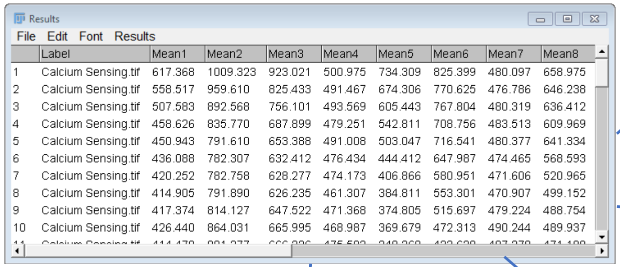


http://imagescience.org/meijering/publications/download/spm2012.pdf



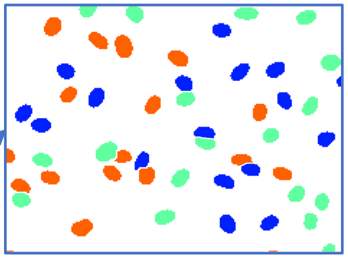
	Label	Mean1	Mean2	Mean3	Mean4	Mean5	Mean6	Mean7	Mean8
1	Calcium Sensing.tif	617.368	1009.323	923.021	500.975	734.309	825.399	480.097	658.975
2	Calcium Sensing.tif	558.517	959.610	825.433	491.467	674.306	770.625	476.786	646.238
3	Calcium Sensing.tif	507.583	892.568	756.101	493.569	605.443	767.804	480.319	636.412
4	Calcium Sensing.tif	458.626	835.770	687.899	479.251	542.811	708.756	483.513	609.969
5	Calcium Sensing.tif	450.943	791.610	653.388	491.008	503.047	716.541	480.377	641.334
6	Calcium Sensing.tif	436.088	782.307	632.412	476.434	444.412	647.987	474.465	568.593
7	Calcium Sensing.tif	420.252	782.758	628.277	474.173	406.866	580.951	471.606	520.965
8	Calcium Sensing.tif	414.905	791.890	626.235	461.307	384.811	553.301	470.907	499.152
9	Calcium Sensing.tif	417.374	814.127	647.522	471.368	374.805	515.697	479.224	488.754
10	Calcium Sensing.tif	426.440	864.031	665.995	468.987	369.679	472.313	490.244	489.937

Object **features**

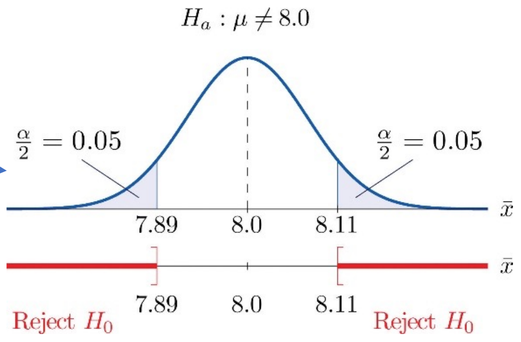


	Label	Mean1	Mean2	Mean3	Mean4	Mean5	Mean6	Mean7	Mean8
1	Calcium Sensing.tif	617.368	1009.323	923.021	500.975	734.309	825.399	480.097	658.975
2	Calcium Sensing.tif	558.517	959.610	825.433	491.467	674.306	770.625	476.786	646.238
3	Calcium Sensing.tif	507.583	892.568	756.101	493.569	605.443	767.804	480.319	636.412
4	Calcium Sensing.tif	458.626	835.770	687.899	479.251	542.811	708.756	483.513	609.969
5	Calcium Sensing.tif	450.943	791.610	653.388	491.008	503.047	716.541	480.377	641.334
6	Calcium Sensing.tif	436.088	782.307	632.412	476.434	444.412	647.987	474.465	568.593
7	Calcium Sensing.tif	420.252	782.758	628.277	474.173	406.866	580.951	471.606	520.965
8	Calcium Sensing.tif	414.905	791.890	626.235	461.307	384.811	553.301	470.907	499.152
9	Calcium Sensing.tif	417.374	814.127	647.522	471.368	374.805	515.697	479.224	488.754
10	Calcium Sensing.tif	426.440	864.031	665.995	468.987	369.679	472.313	490.244	489.937

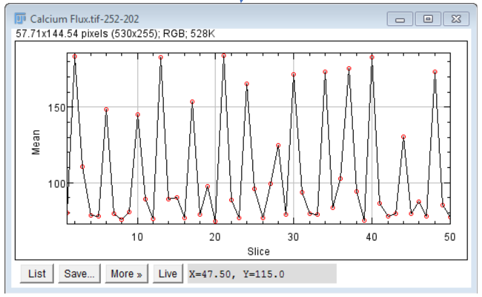
Assign to **classes**



Test statistical hypotheses



Follow temporal events



Estimate relationships

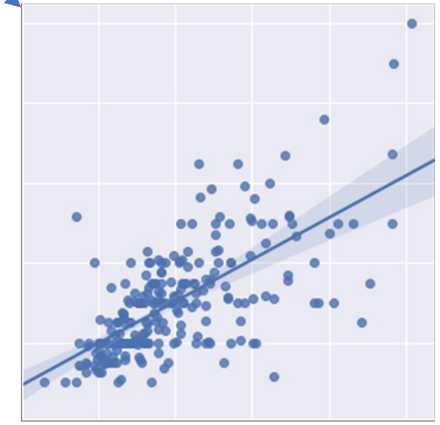


Image processing classes

We can roughly organize the plethora of image processing algorithms in a few broad classes:

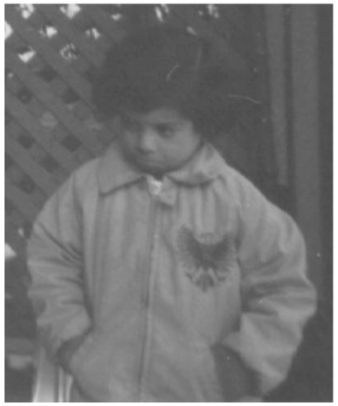
- Image enhancement
- Image restoration
- Image registration
- Image segmentation and classification
- Motion estimation and tracking
- Image compression
- Image visualisation

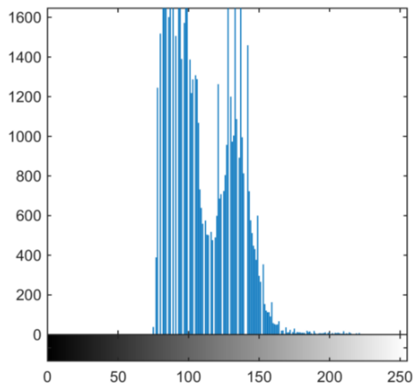
Image enhancement

The often-subjective manipulation or transformation of an image with the aim of increasing its usefulness or visual appearance.

Point-wise image operations, histogram operations, spatial and frequency domain filtering, pseudo-coloring.

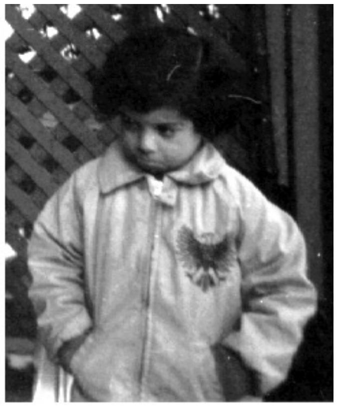
An image with low contrast

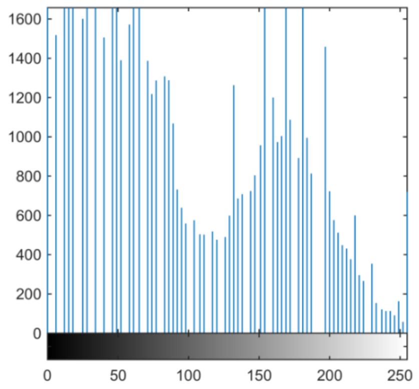




Original image histogram

The same image after contrast stretching





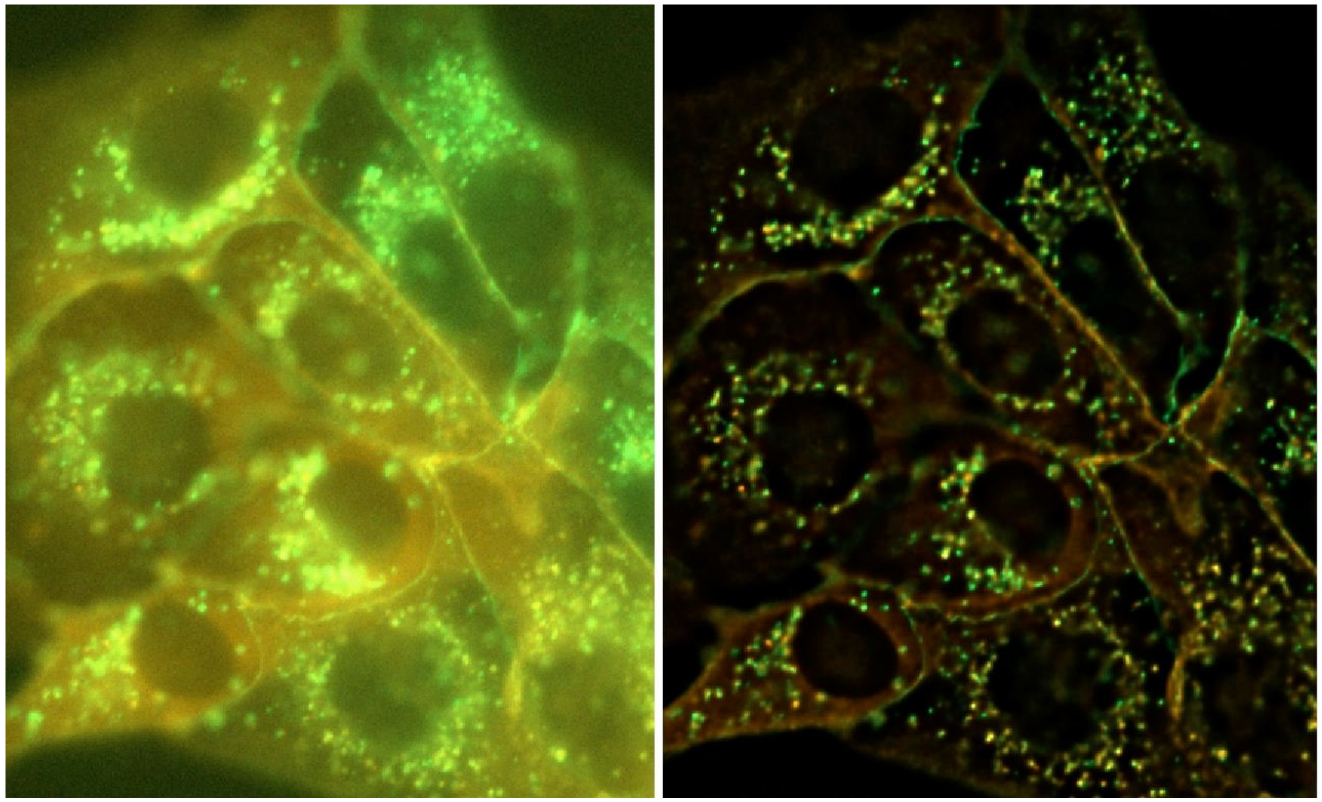
Histogram after contrast stretching

http://www.mathworks.ch/help/images/ref/stretchlim.html

Image restoration

The modeling of the degradation an image is subjected to, and the removal of this degradation based on an optimality criterion.

Noise smoothing, deconvolution, error concealment, inpainting, super-resolution, reconstruction from projections.



Deconvolution of cell-cell junctions of MDCK cells. MDCK (Madin-Darby Canine Kidney) cells cultured for 3 days, were stained for p120-catenin (mCherry - red) and Claudin3 (EGFP-green) and imaged with a Nikon Ti widefield microscope (Objective 40x; 1.3 NA oil lens). Shown are a single slice of the original z-stack either background-corrected (left) or deconvolved with Huygens (right).

Image acquired by Dr. Johan de Rooij, Hubrecht Institute, Utrecht, The Netherlands (source: http://www.svi.nl)

Image registration

The process of transforming different sets of data into one coordinate system. Data may be multiple photographs, data from different sensors, times, depths, or viewpoints.

Intensity-based, feature based, spatial domain methods, frequency domain methods (phase correlation), single-modality, multiple-modality.

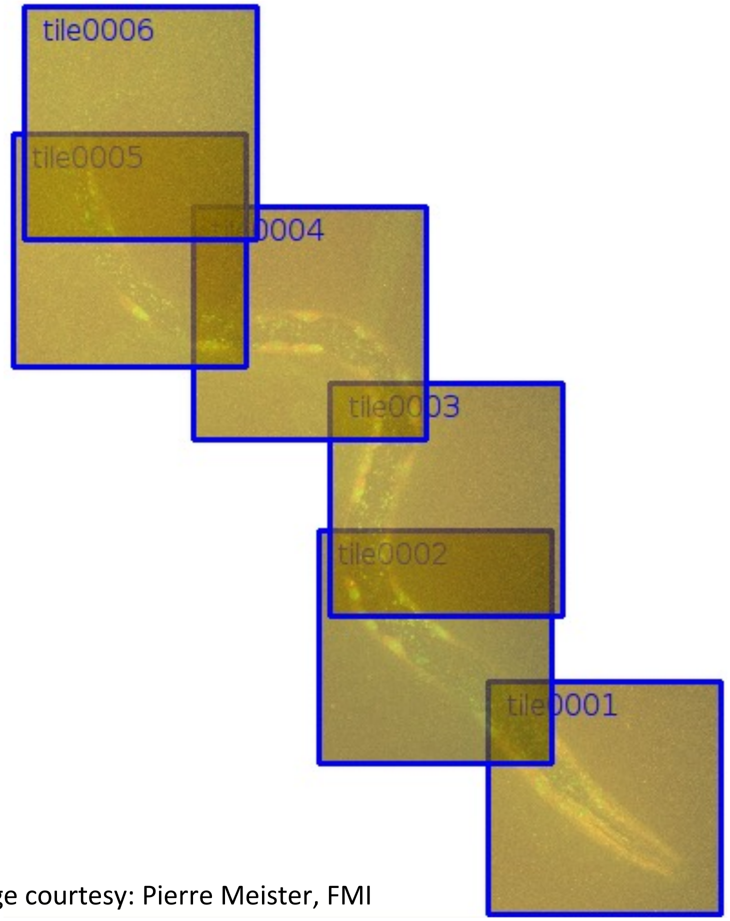


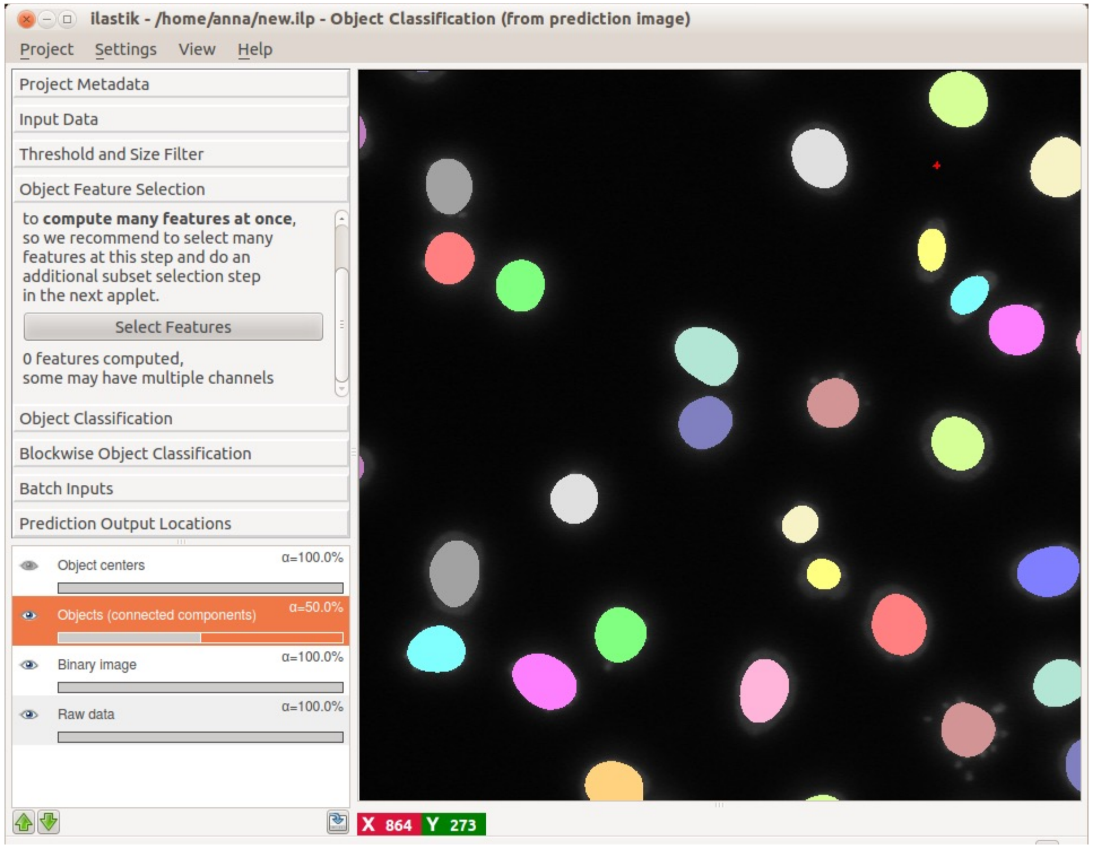
Image courtesy: Pierre Meister, FMI

http://www.xuvtools.org

Image segmentation and classification

The division of an image into constituent, “meaningful” regions and the successive characterization of those regions.

Edge segmentation, thresholding, histogram-based segmentation, region growing, splitting and merging, watershed, K-means algorithms, convolutional neural networks *(e.g., U-Net),* statistical methods, supervised classification.



ilastik - /home/anna/new.ilp - Object Classification (from prediction image)

to **compute many features at once**, so we recommend to select many features at this step and do an additional subset selection step in the next applet.

0 features computed, some may have multiple channels

Object centers α=100.0%

Objects (connected components) α=50.0%

Binary image α=100.0%

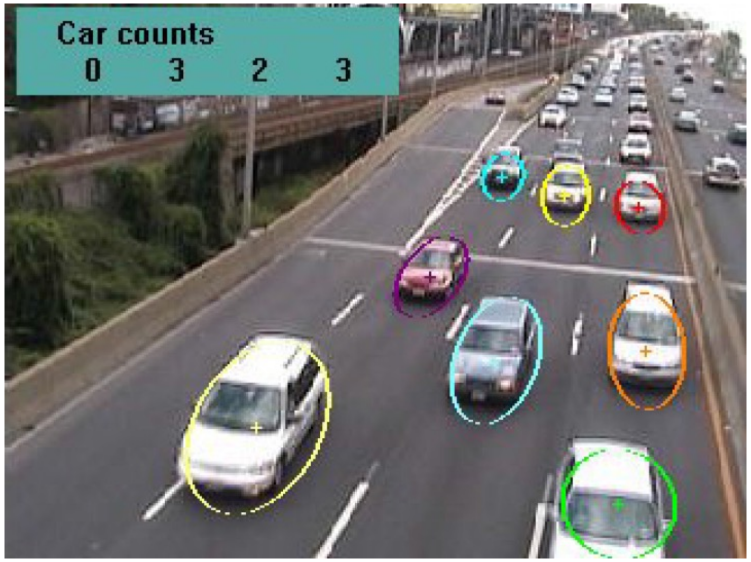
Raw data α=100.0%

http://www.ilastik.org

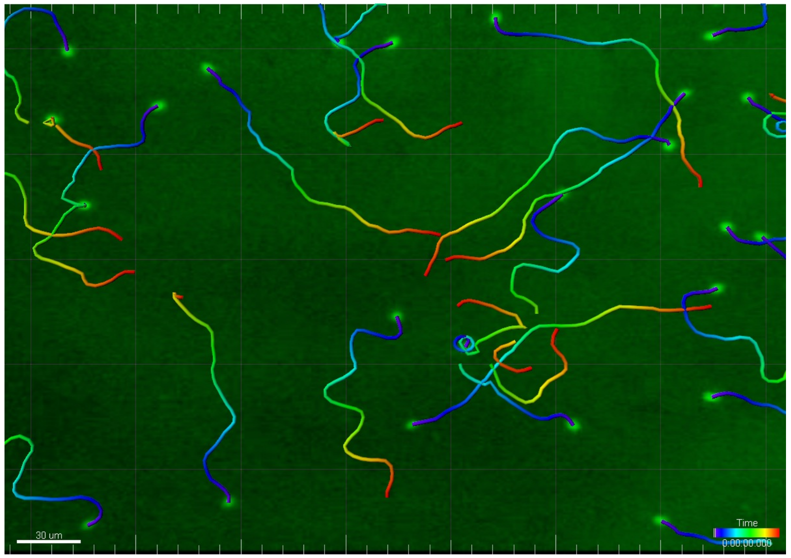
Motion estimation and tracking

The process of determining motion vectors that describe the transformation from one image to another; can also apply to extracted objects (tracking).

Phase correlation, block matching, optical flow, object tracking, single-particle tracking.



Richard Szeliski.
Computer Vision: Algorithms and Applications.
http://szeliski.org/Book/



Tracking swimming algae in Bitplane Imaris.

Image compression

The minimization of the number of bits in representing an image, either in a ***lossless*** or ***lossy*** fashion.

Huffman, arithmetic coding, dictionary methods, PCM, DPCM, JPG, MPEG.

4K 2160p @ 24 fps uncompressed	~600 MB/s (~4.8Gbit/s)
4k 2160p @ 24 fps compressed (YouTube, Netflix)	~3.12 MB/s (25Mb/s) (Netflix 4k requirements)

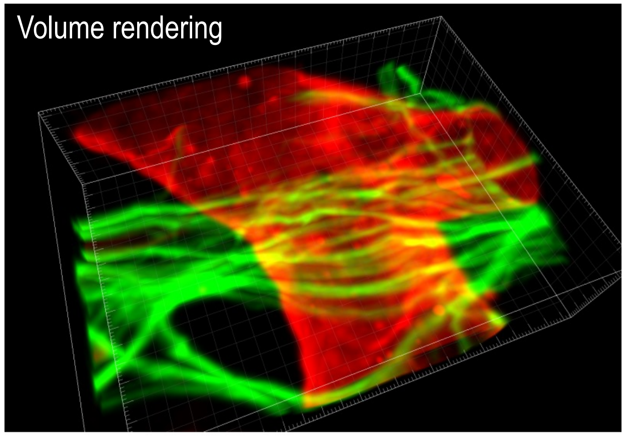
Beware the difference:

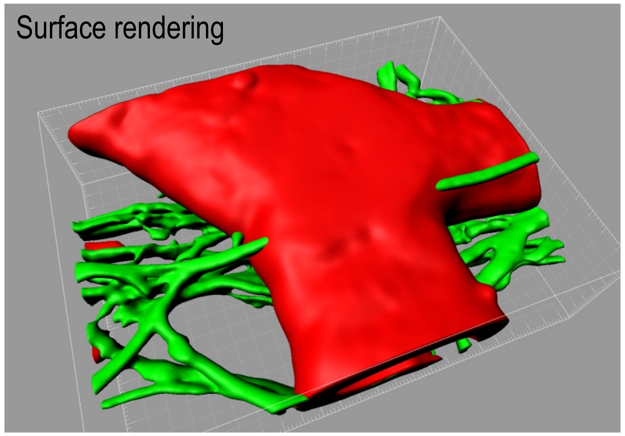
- **Lossless compression** is a class of data compression algorithms that allows the original data to be **perfectly reconstructed** from the compressed data.
  - Some lossless file formats: **TIFF**, **PNG**, and all microscopy vendor proprietary formats (**ND2**, **CZI**, **LIF**, **LSM**, **STK**, …)
- **Lossy compression** permits reconstruction only of an **approximation** of the original data, though this usually improves compression rates (and therefore reduces file sizes).
  - Some lossy file formats: **JPEG**, **JPEG2000**; for video: **MPEG-1/2/4**, **H.264**, **H.265**.
- For scientific (quantitative) purposes, **only** use **lossless compression!**

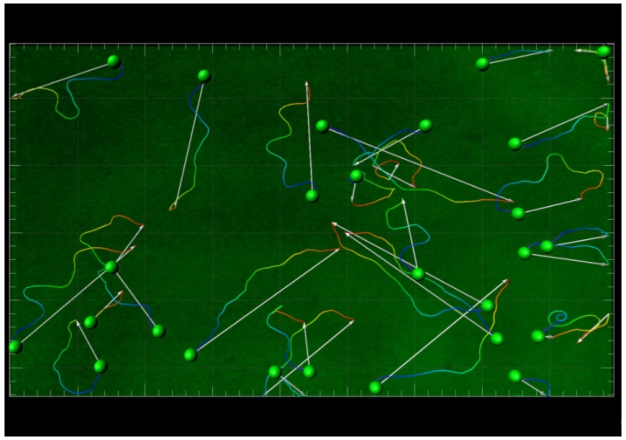
Image visualization

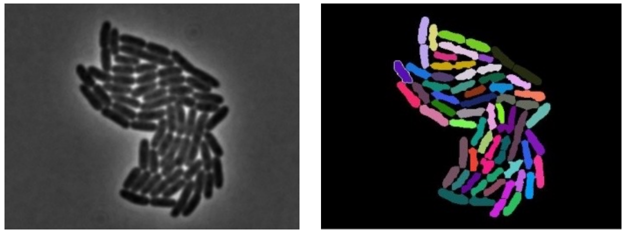
The transformation, selection, or representation of data from simulations or experiments, with an implicit or explicit geometric structure, to allow the exploration, analysis, and understanding of the data.

Volume rendering, MIP rendering, surface rendering.









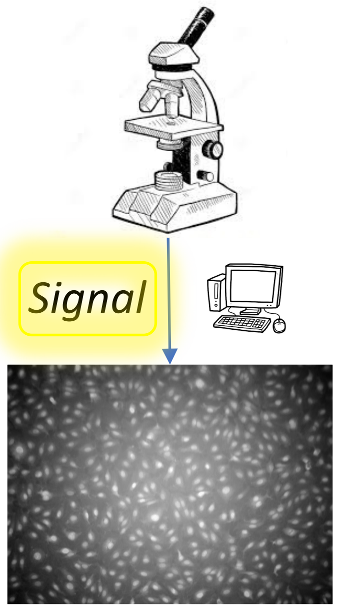
E. coli Segmentation

https://stat.duke.edu/research/software/west/celltracer/example1.html

Rendering of derived measurements along with the data

http://www.bitplane.com

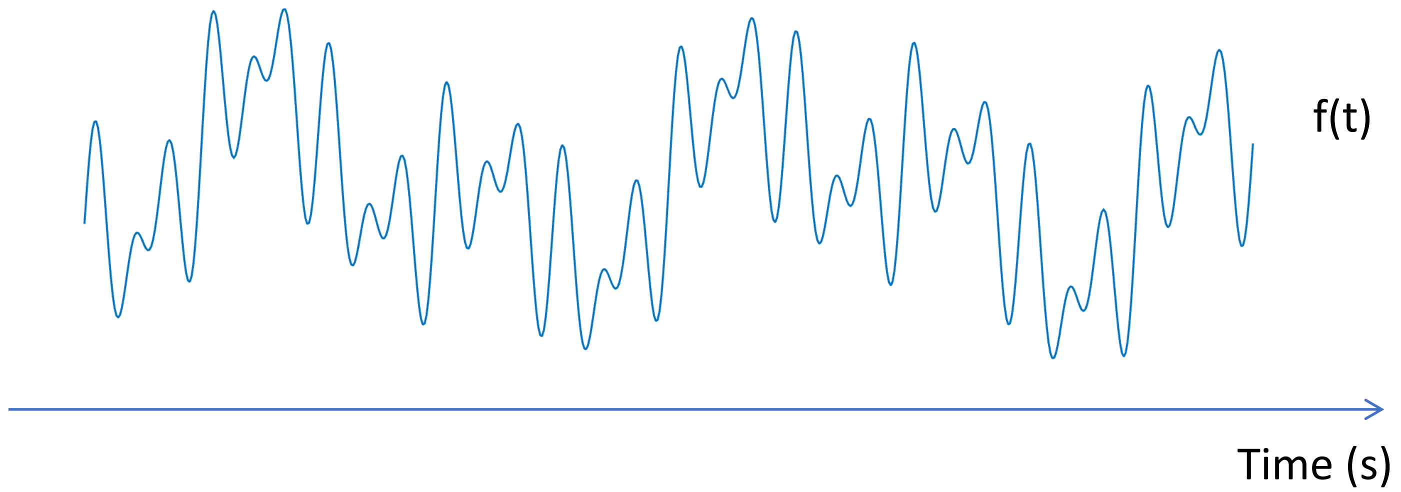
At the beginning of our analysis workflow…



Signals

Signals

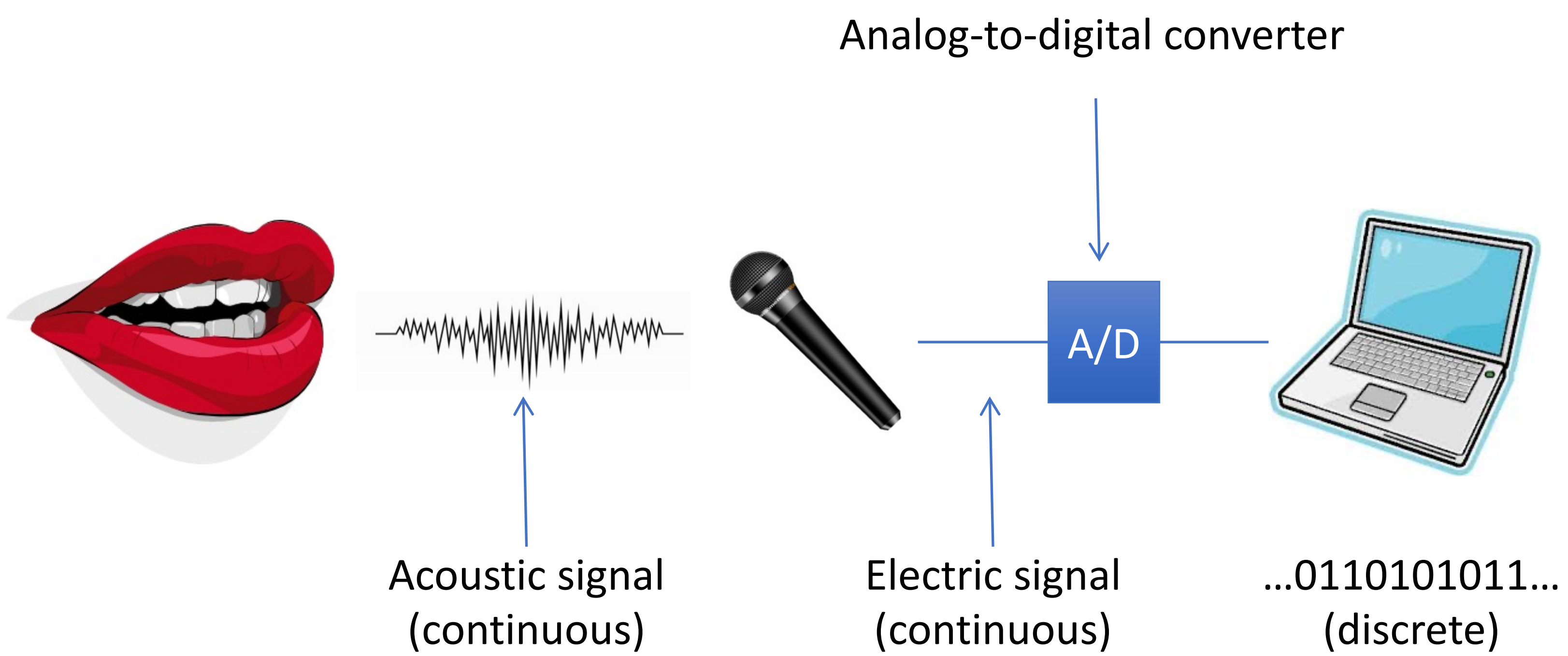
- A **function** containing information about some phenomenon of interest.



- A **quantity** exhibiting variation in time and/or space.

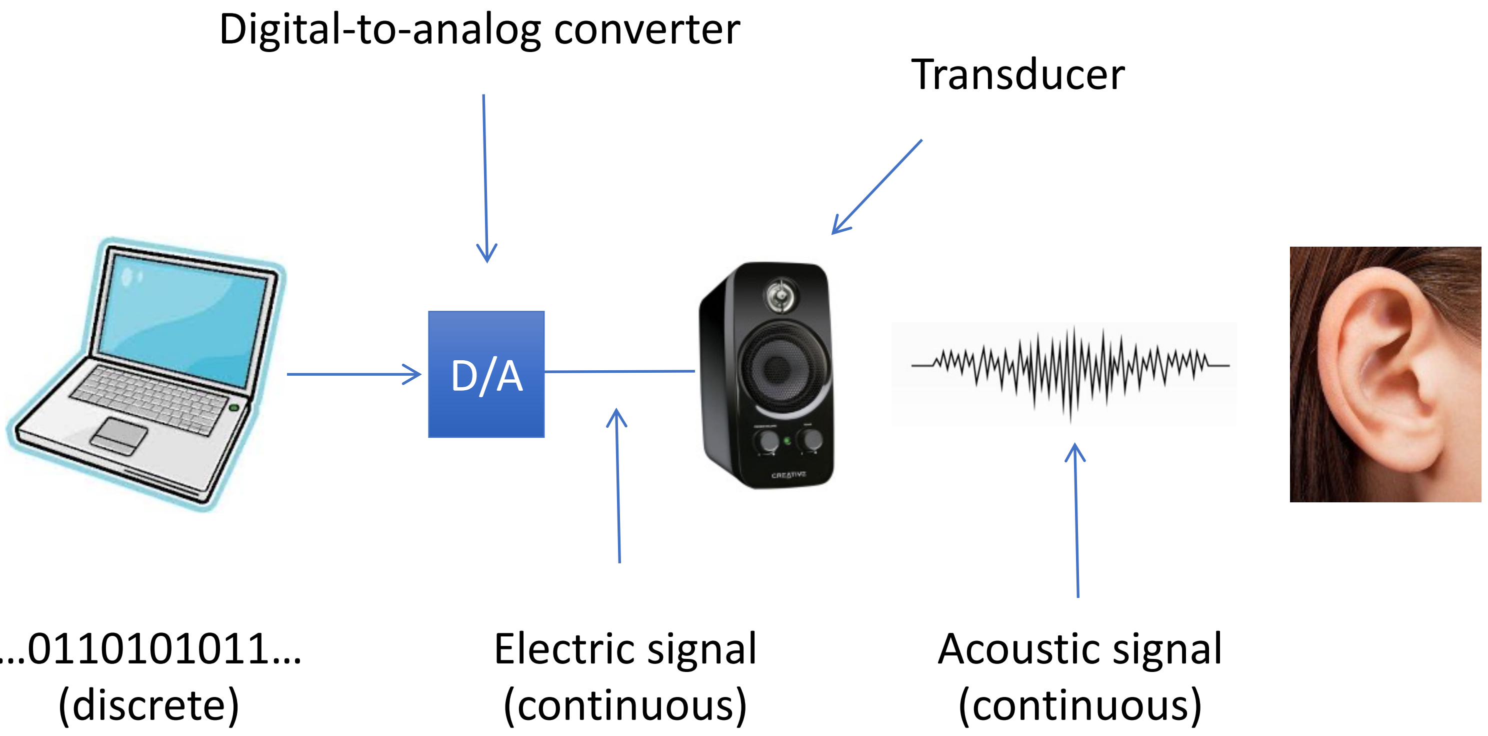
No variation $\longrightarrow$ No information

Analog and digital signals (1D)



1D

Analog and digital signals (1D)

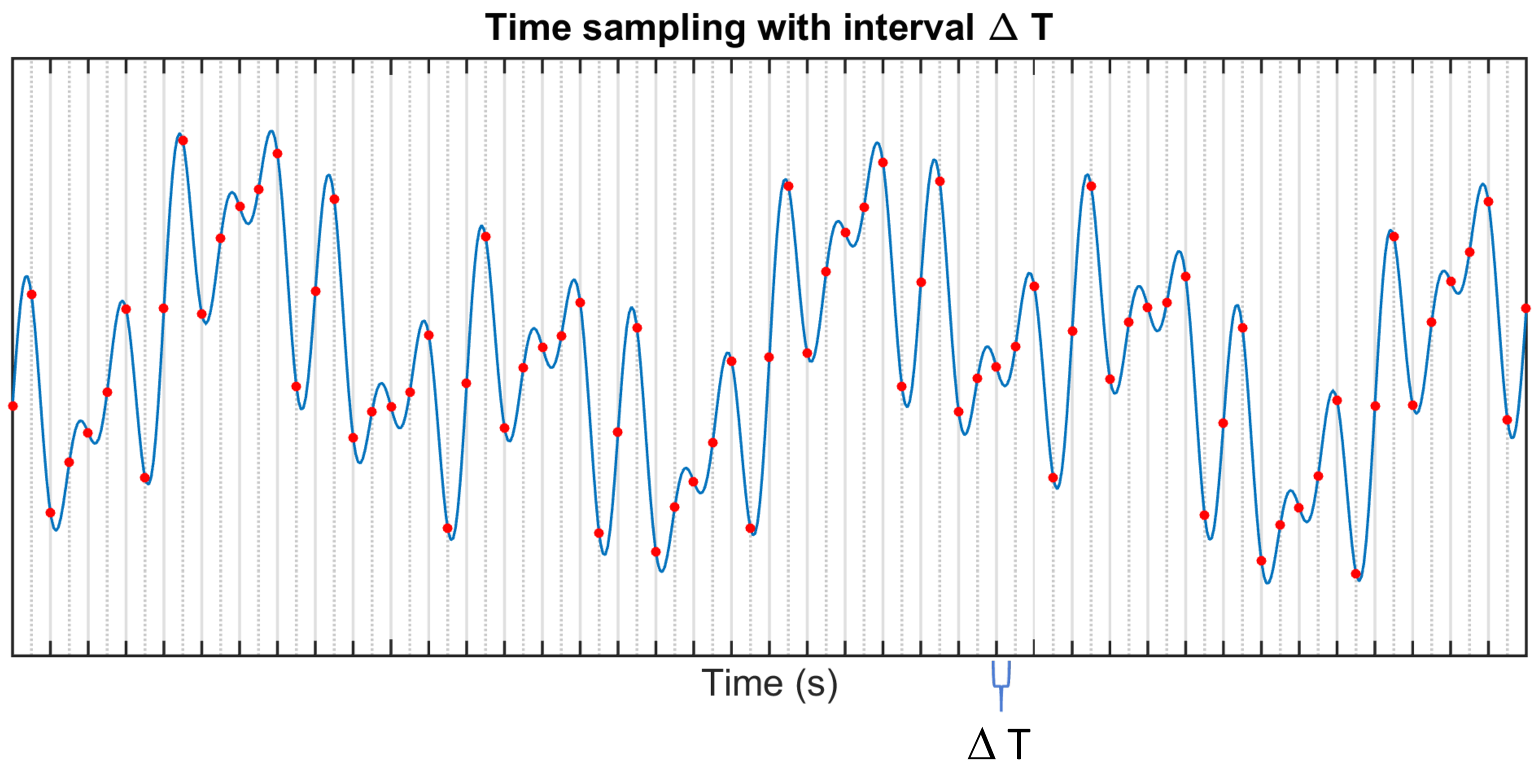


1D

A/D conversion

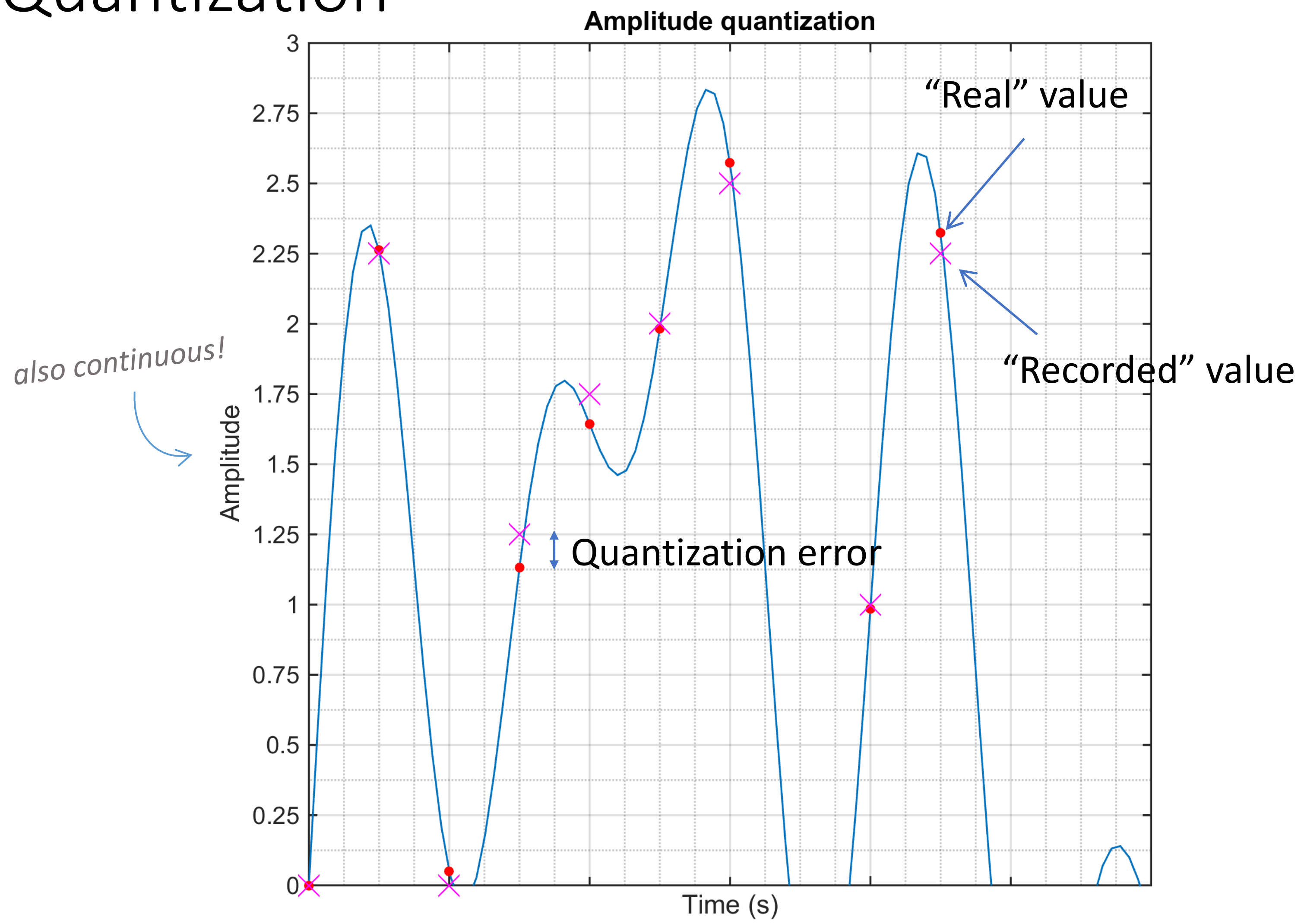
- Analog-to-digital conversion is a 2-step process:
  - **Sampling**: converts a **continuous** signal into a **discrete** one
  - **Quantization**: discretizes the **amplitude** of the signal.

Sampling



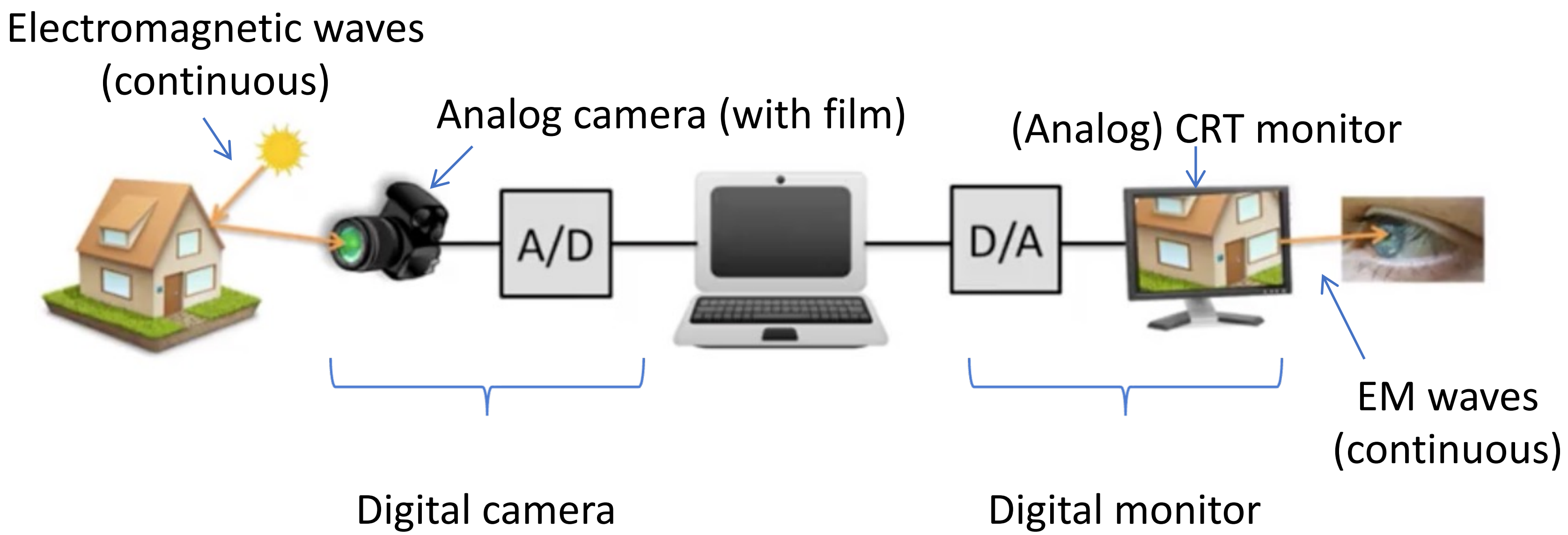
The continuous signal (blue) is measured at discrete time intervals (red dots).

Quantization

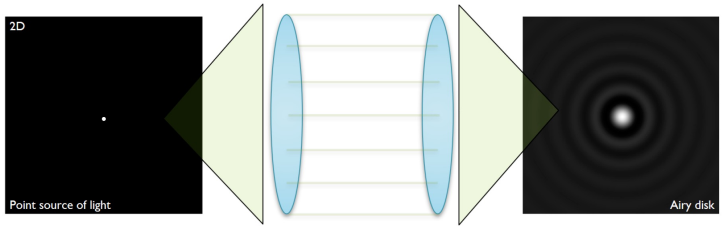


The continuous signal (blue) at discrete time intervals is approximated to a fixed number of discrete values (magenta crosses).

Analog and digital signals (2D)



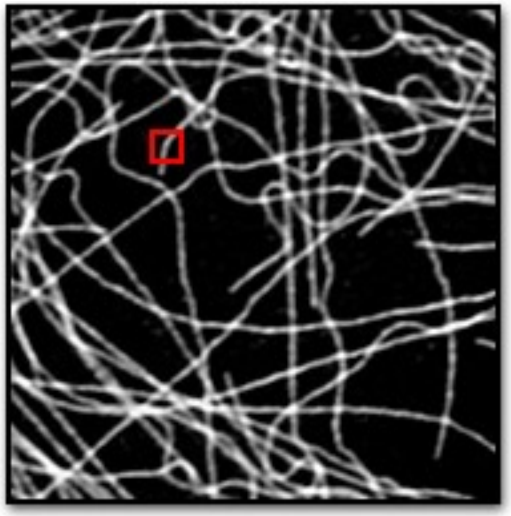
2D



Fluorescence

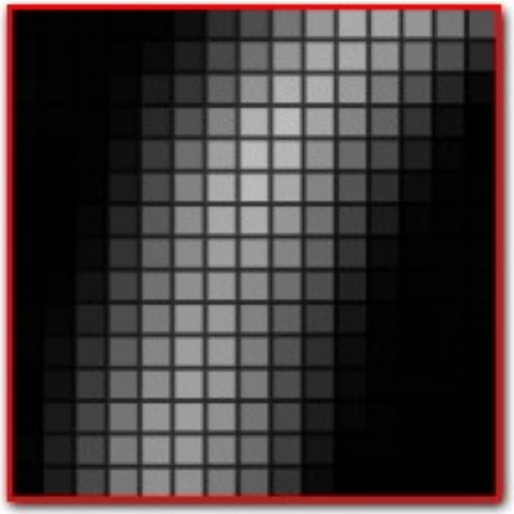
Sampling and quantization (2D)

Analog signal



***M*xN**

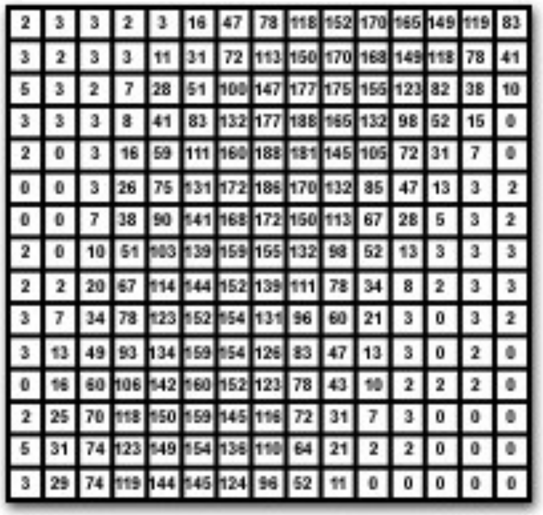
Sampling



Δx

Digital image

Quantization



2	3	3	2	3	16	47	78	118	152	170	165	149	119	83
3	2	3	3	11	31	72	113	150	170	168	149	118	78	41
5	3	2	7	28	51	100	147	177	175	155	123	82	38	10
3	3	3	8	41	83	132	177	188	165	132	98	52	15	0
2	0	3	16	59	111	160	188	181	145	105	72	31	7	0
0	0	3	26	75	131	172	186	170	132	85	47	13	3	2
0	0	7	38	90	141	168	172	150	113	67	28	5	3	2
2	0	10	51	103	139	159	155	132	98	52	13	3	3	3
2	2	20	67	114	144	152	139	111	78	34	8	2	3	3
3	7	34	78	123	152	154	131	96	60	21	3	0	3	2
3	13	49	93	134	159	154	126	83	47	13	3	0	2	0
0	16	60	106	142	160	152	123	78	43	10	2	2	2	0
2	25	70	118	150	159	145	116	72	31	7	3	0	0	0
5	31	74	123	149	154	136	110	64	21	2	2	0	0	0
3	29	74	119	144	145	124	96	52	11	0	0	0	0	0

f(x,y) = {0, ..., 255}

In this example, signal intensity is approximated by **256 discrete values (8 bits)**.

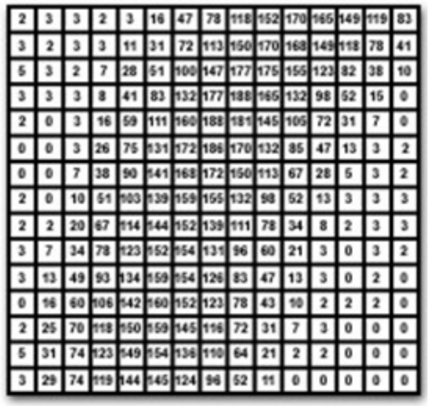
$$f(x,y) = \overbrace{\begin{bmatrix} f(0,0) & f(0,1) & \dots f(0,N-1) \\ f(1,0) & f(1,1) & \dots f(1,N-1) \\ \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots f(M-1,N-1) \end{bmatrix}}^{\mathbf{N}} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}}\right\} \mathbf{M}$$

<u>**Decisions**</u>:

- Sampling: array (image) size: ***M*x*N***
- Quantization: number of discrete values ***G*** for *f(x,y)*

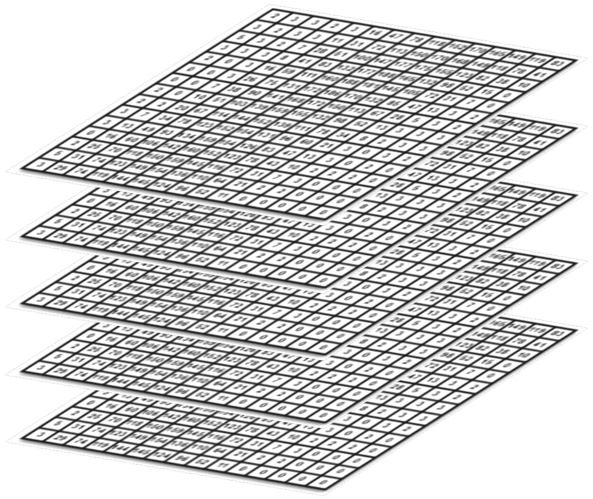
2D and 3D images

I(x,y)



2	3	3	2	3	16	47	78	118	152	170	165	149	119	83
3	2	3	3	11	31	72	113	150	170	168	149	118	78	41
5	3	2	7	28	51	100	147	177	175	155	123	82	38	10
3	3	3	8	41	83	132	177	188	165	132	98	52	15	0
2	0	3	16	59	111	160	188	181	145	105	72	31	7	0
0	0	3	26	75	131	172	186	170	132	85	47	13	3	2
0	0	7	38	90	141	168	172	150	113	67	28	5	3	2
2	0	10	51	103	139	159	155	132	98	52	13	3	3	3
2	2	20	67	114	144	152	139	111	78	34	8	2	3	3
3	7	34	78	123	152	154	131	96	60	21	3	0	3	2
3	13	49	93	134	159	154	126	83	47	13	3	0	2	0
0	16	60	106	142	160	152	123	78	43	10	2	2	2	0
2	25	70	118	150	159	145	116	72	31	7	3	0	0	0
5	31	74	123	149	154	136	110	64	21	2	2	0	0	0
3	29	74	119	144	145	124	96	52	11	0	0	0	0	0

- In 2D images, each grid element, or ***pixel*** (picture element), is defined as a *location* and a *value* representing the characteristic of the signal at that location.



I(x,y,z)

- In 3D images, the pixel is called ***voxel*** (volume element).
- Common in the field of **biomedical imaging**.

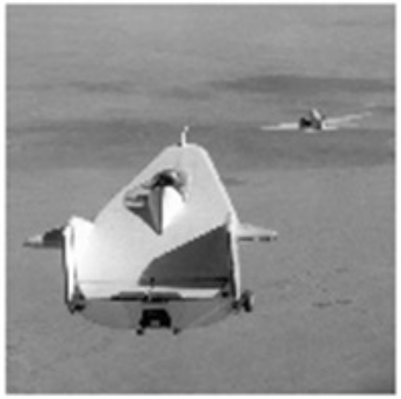
Digital image concepts

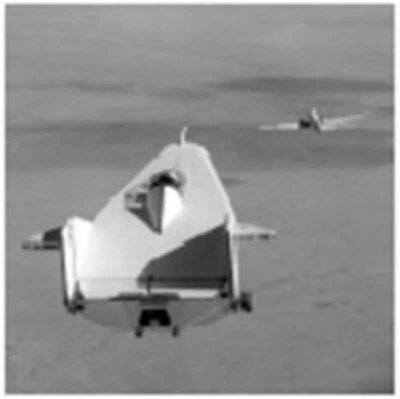
Sampling → **<u>Resolution</u>**
Number of pixels used to represent the original signal (*e.g.,* 512 x 512).

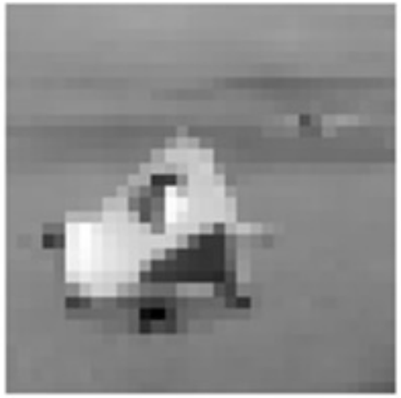
Quantization → **<u>Bit depth</u>**
Number of bits used for quantization → number of gray levels in an image (*e.g.,* 8 bits → 2^8 = 256 levels)

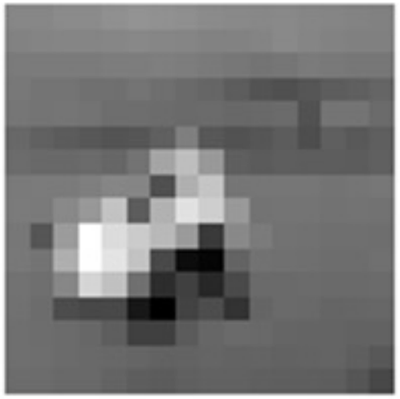
<u>Number of image channels</u>
An image can have one or more channels (*e.g.*, intensity (gray-value), composite, or RGB image).

Sampling → spatial resolution

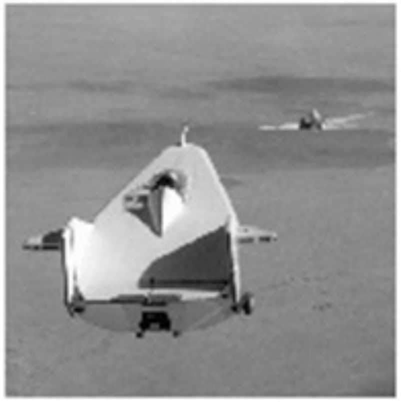

256 x 256

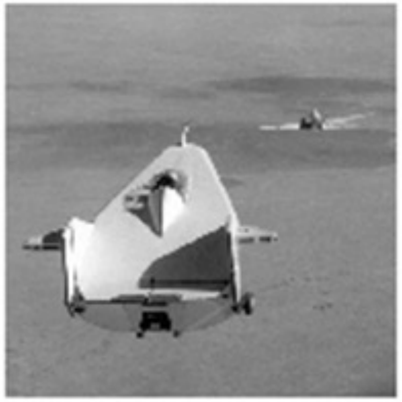

128 x 128

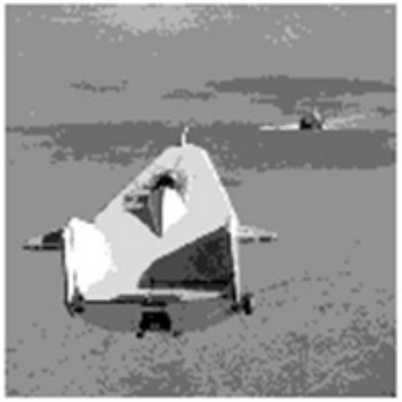

32 x 32

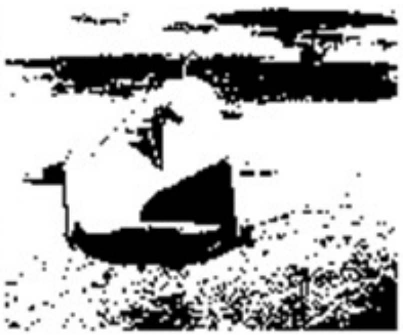

16 x 16

Quantization → grayscale resolution

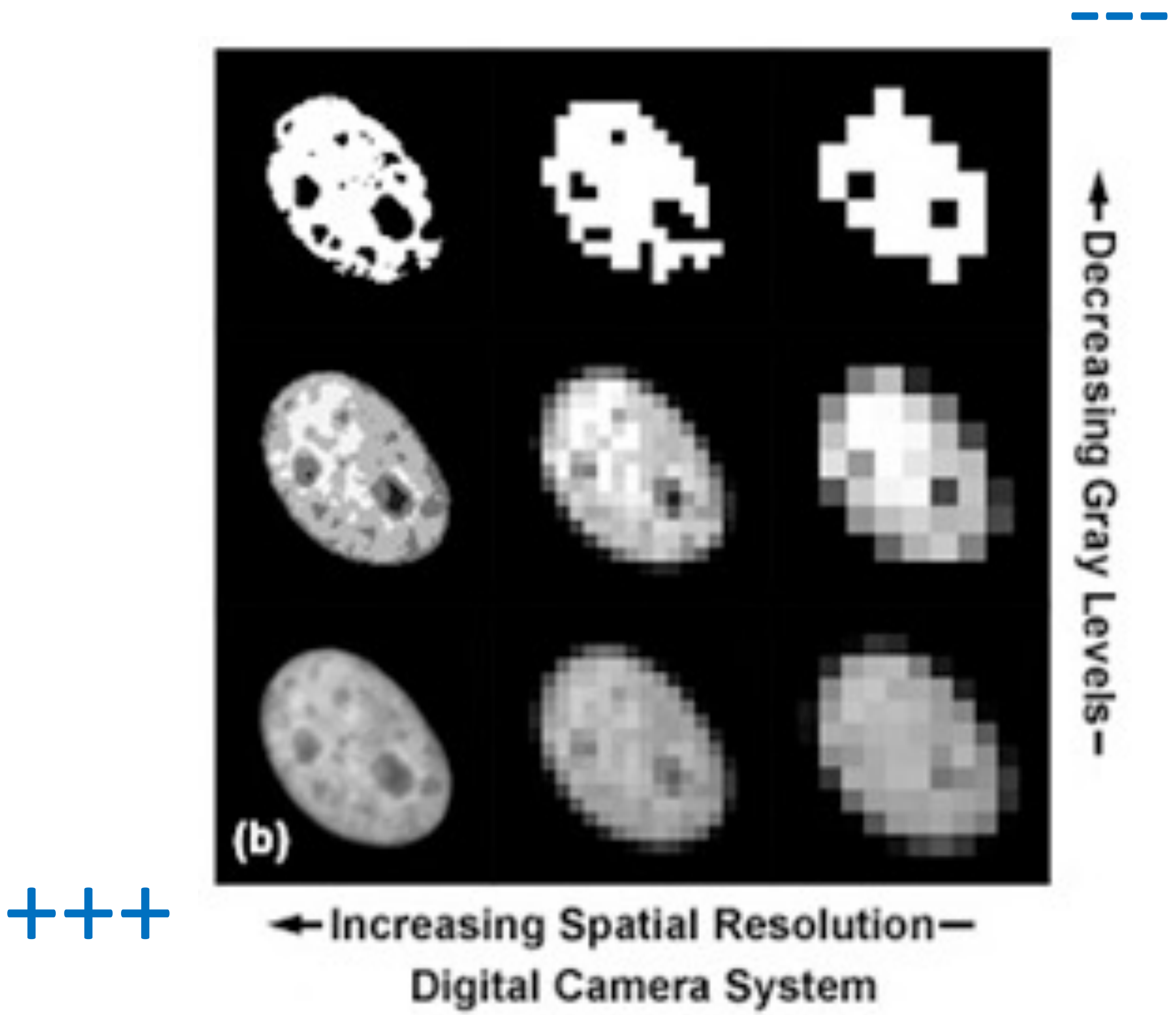

256 levels (8 bit)

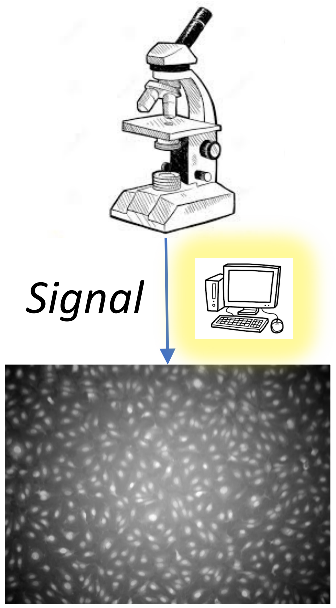

64 levels (6 bit)


8 levels (3 bit)


2 levels (1 bit)

Resolution summary





Systems

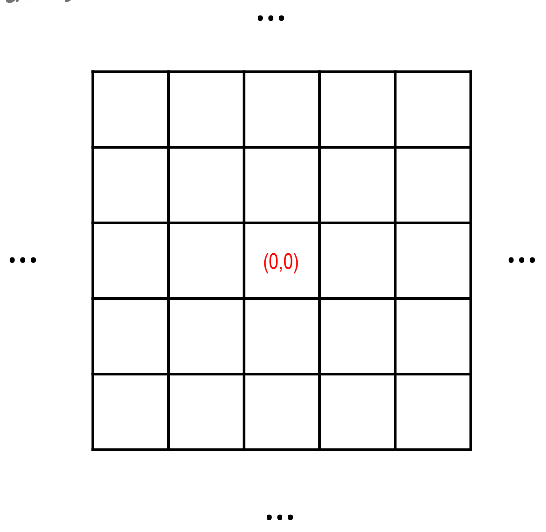
Will focus on discrete systems, but the following also applies to analog ones.

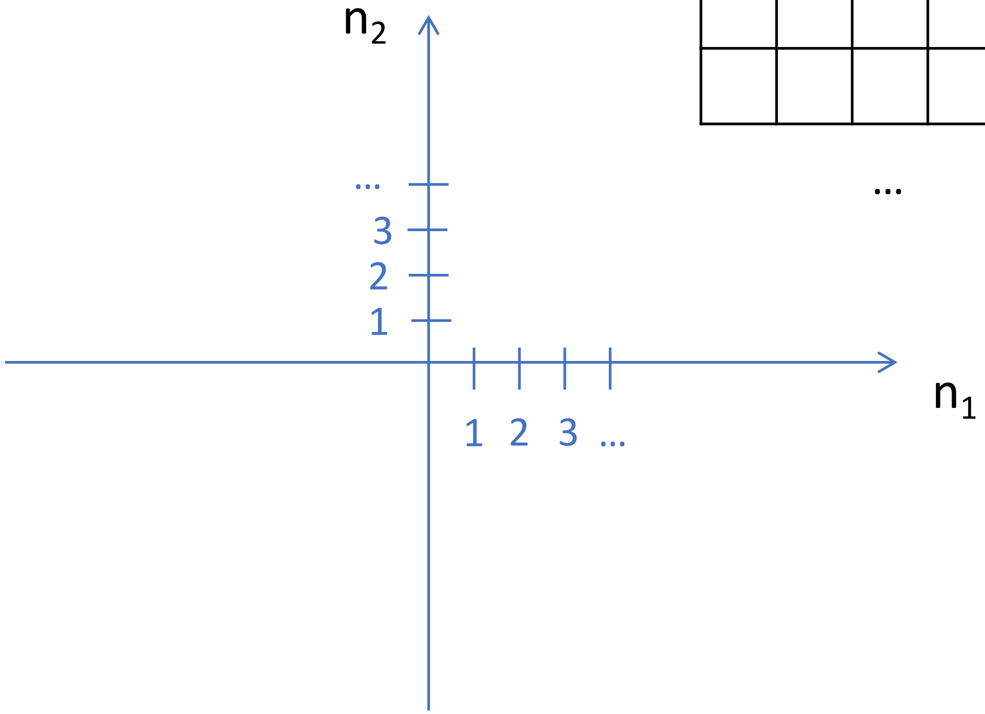
Discrete signal (notation)

Remember:
a discrete signal is the sampled
and quantized version of
a continuous signal!

$x(n_1, n_2)$

value of x at position (n_1, n_2)





Systems

$$\begin{tikzcd}
 & \textit{e.g., a digital image} \arrow[ld, bend left] & & \\
\text{Input} \arrow[d] & & & \text{Output} \arrow[d] \\
x(n_1, n_2) \arrow[r] & \boxed{T[\cdot]} \arrow[rr] & & y(n_1, n_2) = \mathbf{T}[x(n_1, n_2)] \\
 & & & \text{Transformation} \arrow[u]
\end{tikzcd}$$

<u>Examples</u>:

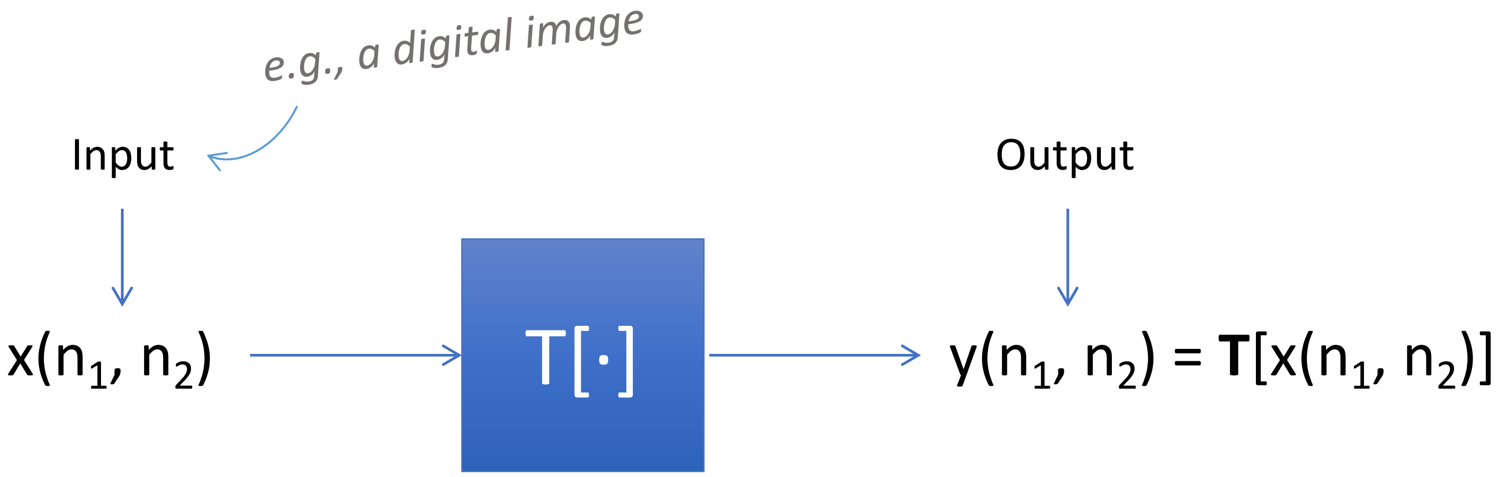
Inversion (8 bit)

$$y(n_1, n_2) = 255 - x(n_1, n_2)$$

$$y(n_1, n_2) = \text{median}(\underset{\uparrow}{\mathbf{N}}(x(n_1, n_2)))$$

A *neighborhood* of a given position (pixel) $x(n_1, n_2)$

Systems



T[] can be any sort of transformation (system) of the input signal $x(n_1, n_2)$.

We will now consider a family of systems with following properties:

- Linearity
- Spatial (shift) invariance

Why focusing on LSI systems?

- LSI systems:
  - describe processes involved in **image formation**, particularly **in the microscope** (*continuous* LSI systems);
  - are the foundation of **image processing operations** that we can perform on our images (*discrete* LSI systems).

Keep this in mind when you go to the microscope!

Try to link what's coming in the next slides to what happens to the fluorescence of your samples in the microscope.

Linear systems

a_1, a_2: weights

x_1, x_2: signals

T: system (transformation)

If

$$\mathbf{T}[a_1x_1(n_1, n_2) + a_2x_2(n_1, n_2)] = a_1\mathbf{T}[x_1(n_1, n_2)] + a_2\mathbf{T}[x_2(n_1, n_2)]$$

then **T**[] is linear.

The transformed version of a weighted sum of signals is the same as the weighted sum of the signals transformed individually.

(Alternatively, a linear system can be decomposed into constituents that are processed independently, and the result combined in the end.)

Shift-invariant systems

Given:

$$\mathbf{T}[x(n_1, n_2)] = y(n_1, n_2)$$

If

k_1, k_2: shifts

$$\mathbf{T}[x(n_1 - k_1, n_2 - k_2)] = y(n_1 - k_1, n_2 - k_2)$$

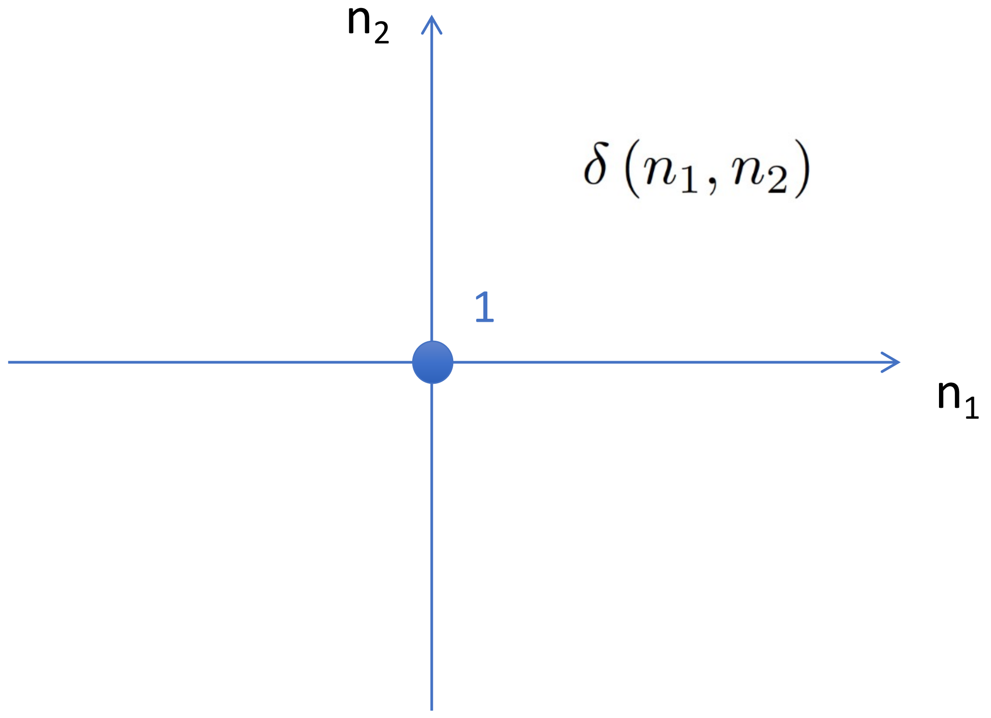
then **T**[] is shift-invariant.

If the input is shifted by a given amount, the output will be shifted by the same amount.

(Or, the location of the origin of the coordinate system is irrelevant.)

Discrete Unit Impulse

$$\delta\left(n_1, n_2\right) = \begin{cases} 1, & for\ n_1 = n_2 = 0 \\ 0, & otherwise \end{cases}$$



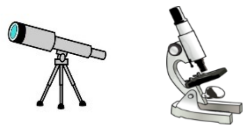
$$\begin{array}{ccccccc} & & & \cdots & & & \\ & 0 & 0 & 0 & 0 & 0 & \\ & 0 & 0 & 0 & 0 & 0 & \\ \cdots & 0 & 0 & 1 & 0 & 0 & \cdots \\ & 0 & 0 & 0 & 0 & 0 & \\ & 0 & 0 & 0 & 0 & 0 & \\ & & & \cdots & & & \end{array}$$

Linear Shift-Invariant systems

$$\begin{tikzcd}
\textbf{Unit impulse} \arrow[d] & & \textbf{Impulse response} \text{ of the LSI system} \arrow[d] \\
\delta(n_1, n_2) \arrow[r] & \boxed{\text{LSI}} \arrow[r] & h(n_1, n_2)
\end{tikzcd}$$

Linear Shift-Invariant systems

these are continuous systems, but the concept is the same



$$\begin{tikzcd}
\textbf{Unit impulse} \arrow[d] & & \textbf{Impulse response} \text{ of the LSI system} \arrow[d] \\
\delta(n_1, n_2) \arrow[r] & \boxed{\text{LSI}} \arrow[r] & h(n_1, n_2) \arrow[dl, bend left] \\
x(n_1, n_2) \arrow[r] & \boxed{h(n_1, n_2)} \arrow[r] & y(n_1, n_2)
\end{tikzcd}$$

we can measure it!

The system response to the unit impulse is all we need to fully describe the LSI system.

Convolution

$$\begin{tikzcd}
 & \text{Impulse response} \arrow[d] & \\
x(n_1, n_2) \arrow[r] & \boxed{h(n_1, n_2)} \arrow[r] & y(n_1, n_2)
\end{tikzcd}$$

LSI systems can be described and efficiently implemented by the mathematical operation of **convolution**.

$$\boxed{y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2)}$$

$$y\left(n_1, n_2\right) = x\left(n_1, n_2\right) \circledast h\left(n_1, n_2\right) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x\left(k_1, k_2\right) h\left(n_1 - k_1, n_2 - k_2\right)$$

Convolution (1D example)

$$y(n) = x(n) \circledast h(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

Signal

```
x = [1 2 3 4 5]

h = [2 4 6]
```

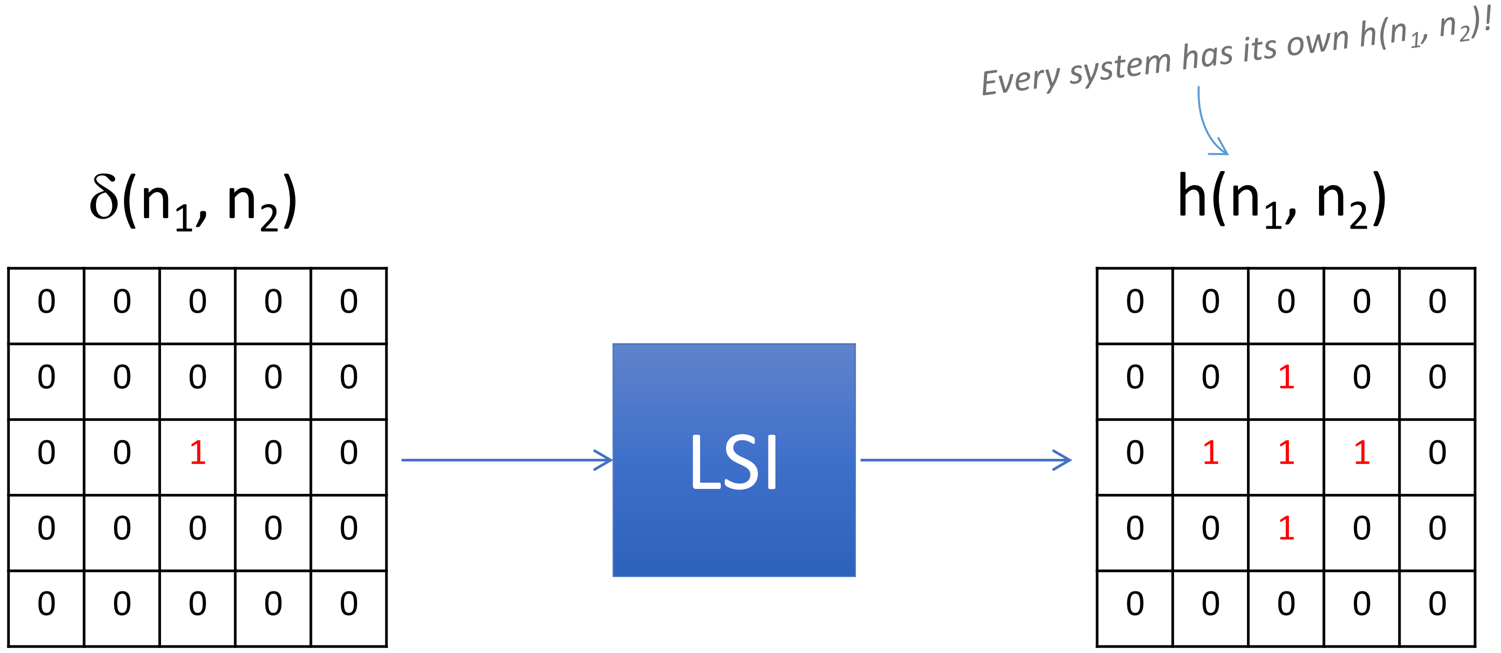
Impulse response

```
      [1 2 3 4 5]
  ←        [6 4 2]        5 * 6 = [30]

     [1 2 3 4 5]
  ←       [6 4 2]         6 * 4 + 5 * 4 = [44 30]

     [1 2 3 4 5]
  ←      [6 4 2]          3 * 6 + 4 * 4 + 5 * 2 = [44 44 30]

     [1 2 3 4 5]
  ←     [6 4 2]           2 * 6 + 3 * 4 + 4 * 2 = [32 44 44 30]

     [1 2 3 4 5]
  ←  [6 4 2]              1 * 6 + 2 * 4 + 3 * 2 = [20 32 44 44 30]

      [1 2 3 4 5]
  ← [6 4 2]               1 * 4 + 2 * 2 = [8 20 32 44 44 30]

       [1 2 3 4 5]
  ← [6 4 2]               1 * 2 = [2 8 20 32 44 44 30]
```

Full overlap

Full convolution: y(n) = [2 8 20 32 44 44 30]

Linear Shift-Invariant systems



0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

Impulse signal

0	0	0	0	0
0	0	1	0	0
0	1	1	1	0
0	0	1	0	0
0	0	0	0	0

Impulse response function

Linear Shift-Invariant systems

$x(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Shift invariance

$\xrightarrow{\text{LSI}}$

$y(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0

$$y(n_1, n_2) = x(n_1, n_2) \circledast h(n_1, n_2)$$

Each δ in the input signal gets its own response, and the shape of each response is independent of the location of the δ function in the input.

Linear Shift-Invariant systems

$x(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Linear (combination)

$$\xrightarrow{\text{LSI}}$$

$y(n_1, n_2)$

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	**1**	**1**	0	0	0
0	0	**1**	**2**	**2**	**1**	0	0
0	0	0	**1**	**1**	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

The output signal is formed as a linear combination (*i.e.*, weighted sum) of spatially-shifted impulse response functions.

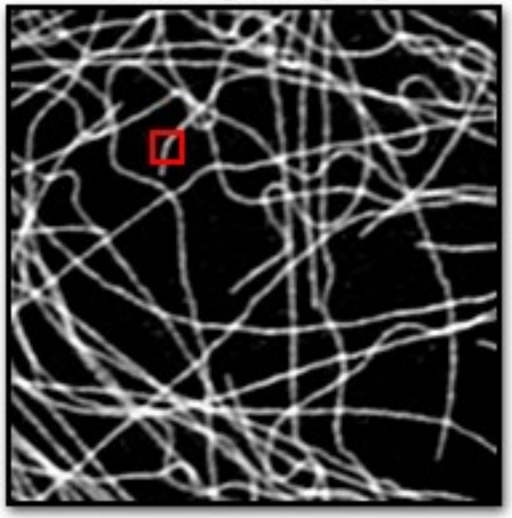
Digital images

The artists formerly known as discrete signals.

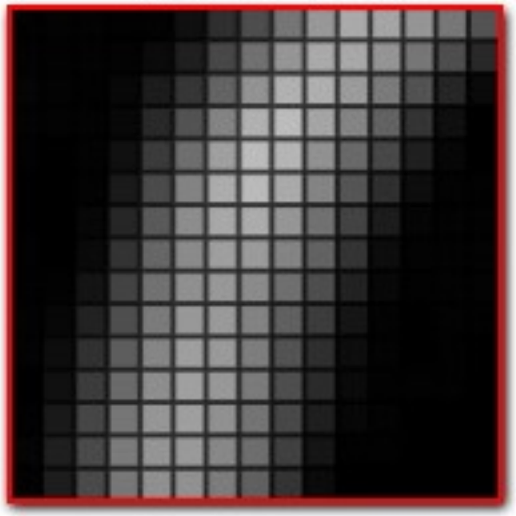
Digital image

A digital image is a numeric representation of a continuous 2D or 3D signal.

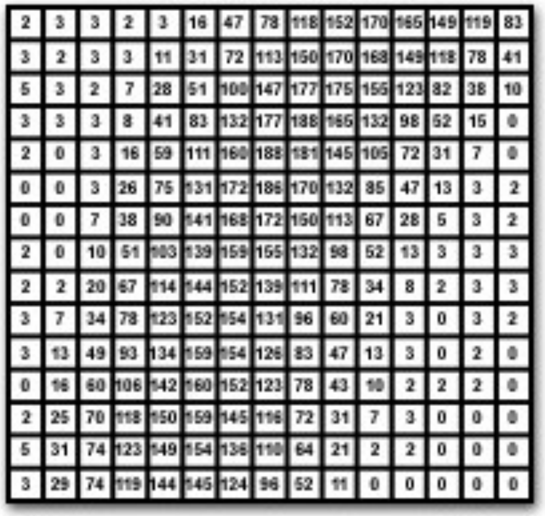
Analog Imaging



Digital Sampling



Pixel Quantization



2	3	3	2	3	16	47	78	118	152	170	165	149	119	83
3	2	3	3	11	31	72	113	150	170	168	149	118	78	41
5	3	2	7	28	51	100	147	177	175	155	123	82	38	10
3	3	3	8	41	83	132	177	188	165	132	98	52	15	0
2	0	3	16	59	111	160	188	181	145	105	72	31	7	0
0	0	3	26	75	131	172	186	170	132	85	47	13	3	2
0	0	7	38	90	141	168	172	150	113	67	28	5	3	2
2	0	10	51	103	139	159	155	132	98	52	13	3	3	3
2	2	20	67	114	144	152	139	111	78	34	8	2	3	3
3	7	34	78	123	152	154	131	96	60	21	3	0	3	2
3	13	49	93	134	159	154	126	83	47	13	3	0	2	0
0	16	60	106	142	160	152	123	78	43	10	2	2	2	0
2	25	70	118	150	159	145	116	72	31	7	3	0	0	0
5	31	74	123	149	154	136	110	64	21	2	2	0	0	0
3	29	74	119	144	145	124	96	52	11	0	0	0	0	0

How do we represent a digital image in the computer?

Data types

- **Integers**: 0, 13, -14378, …

$$\boxed{\text{1 bit: } \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right.}$$

black

white

00	$\rightarrow$ 0
01	$\rightarrow$ 1
10	$\rightarrow$ 2
11	$\rightarrow$ 3

111 = 7

3 bits: 8 values (2^3) *11111111 = 255*

8 bits (1 byte): 256 values (2^8)

16 bits (2 bytes): 65536 values (2^{16})

$\downarrow$

n bits: 2^n values; n: **bit depth**

range: $0 - (2^n - 1)$

- **Floating-point numbers**: 3.21, -0.001, 3.4e7, …

0-10000000-01000000000000000000000 = 2.5

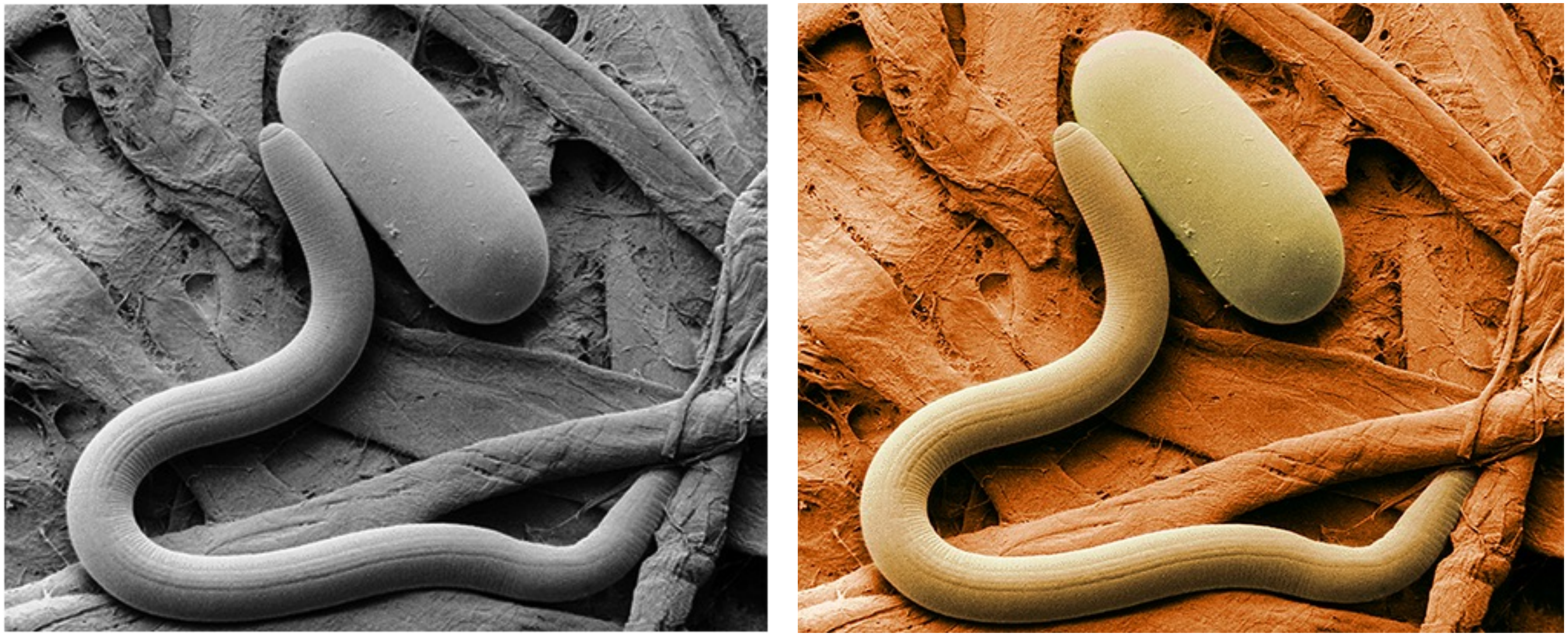
Precision	Sign	Exponent	Mantissa	Total	Range	Precision
Single	1	8	23	32	$1.17^{-38} - 3.40^{38}$	1.19^{-7}
Double	1	11	52	64	$2.22^{-308} - 1.80^{308}$	2.22^{-16}

Precision at 1.0

Floating point numbers **approximate** real numbers.

Color lookup tables (LUT)

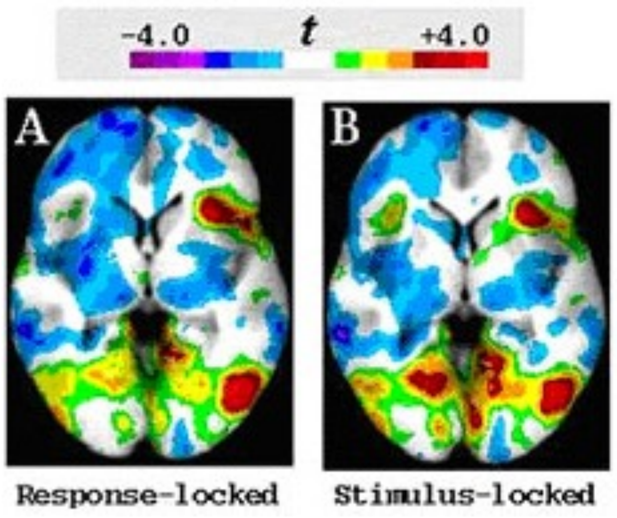
- In biomedical imaging it is common to apply **false colors** while retaining the original intensity content (*i.e.*, signal strength).



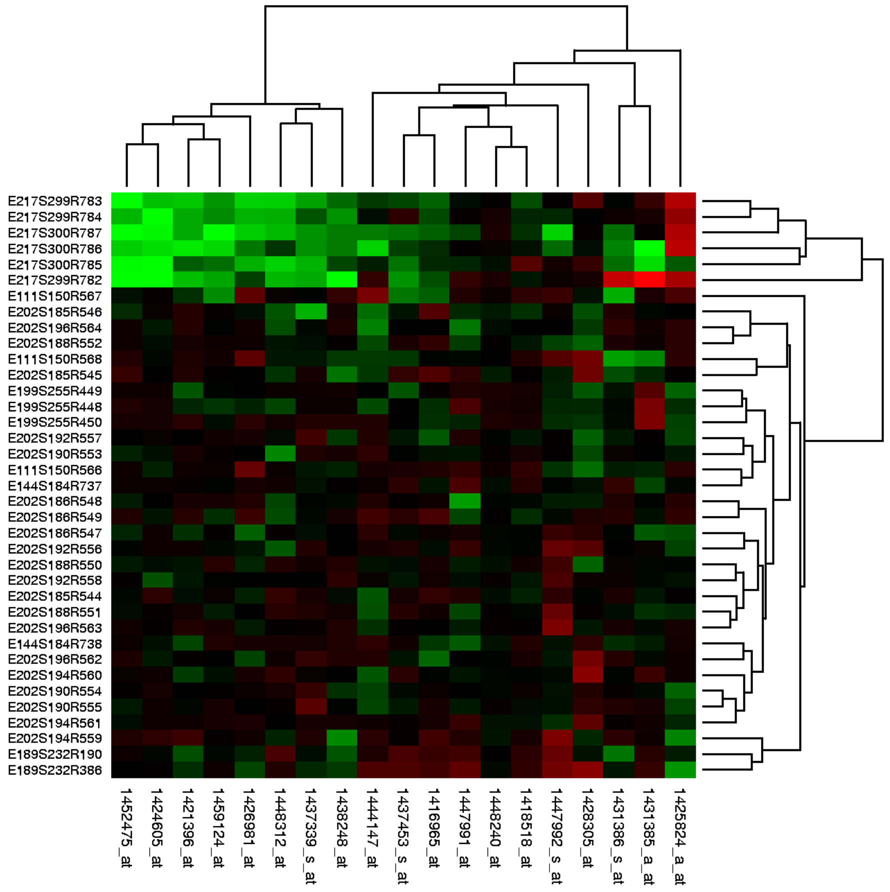
Colored SEM image of **soybean cyst nematode** and **egg**. The color makes the image easier for non-specialists to view and understand the structures and surfaces revealed in micrographs.

Color lookup tables (LUT)

- Color can also be used to convey a visual meaning to the values or measurements in the image.



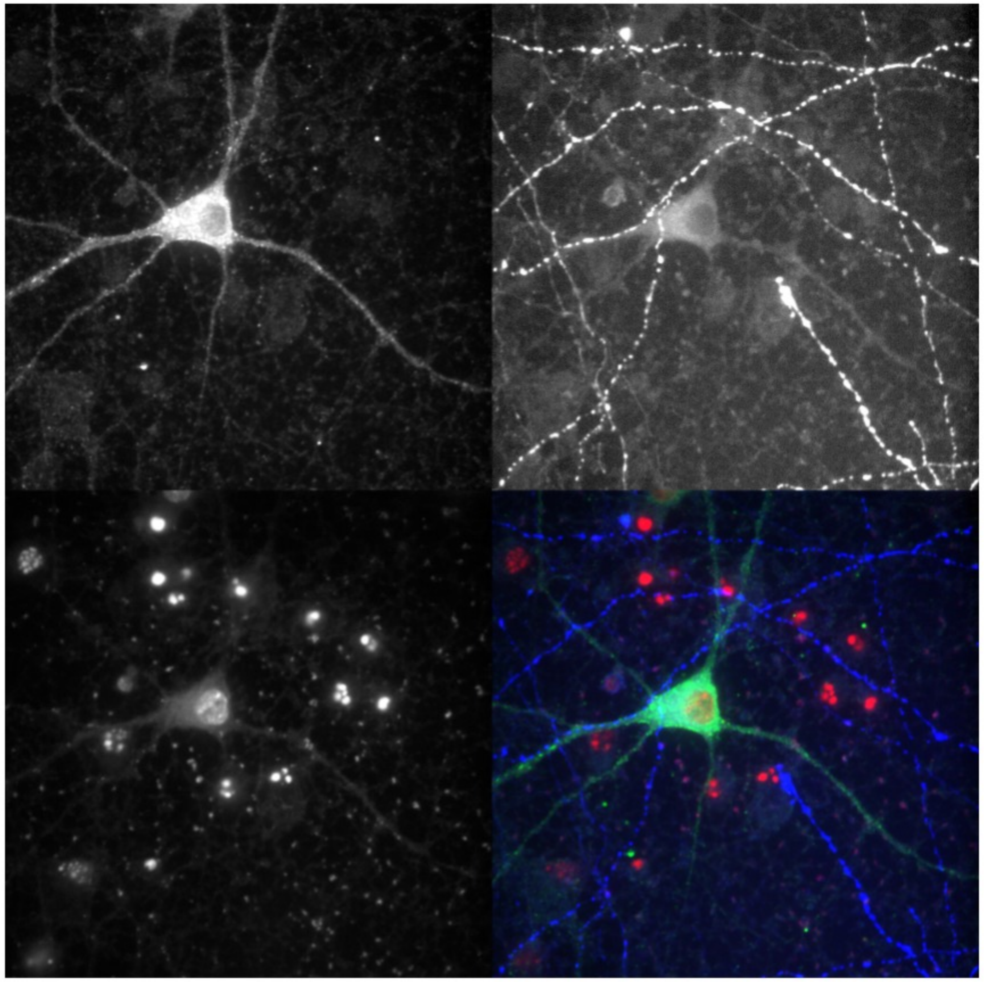
Differences in blood and oxygen levels in the brain shown by fMRI maps.



Heat map generated from DNA microarray data reflecting gene expression values in several conditions

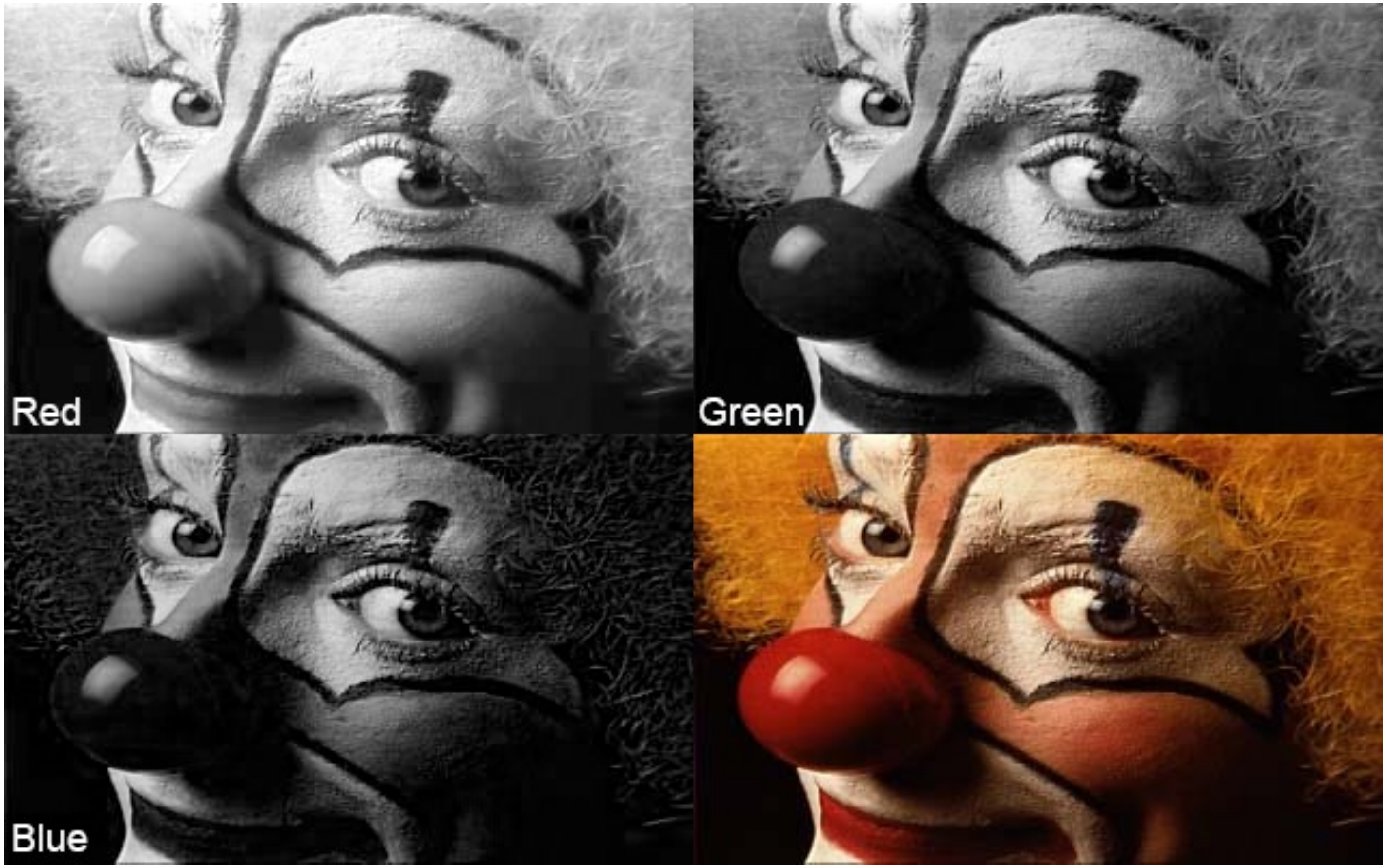
Multichannel images

Multi-channel images are common in scientific applications, for instance in light microscopy acquisitions. Each intensity channel is assigned a different color and then merged into a **composite**.



RGB

The **RGB** color model is an additive color model in which **R**ed, **G**reen, and **B**lue components are added together in various proportions to reproduce a broad array of colors.



Building blocks of image processing

Image histogram

The *histogram* counts the number of pixels in the image that have a certain value.

512 x 512, 256 gray levels

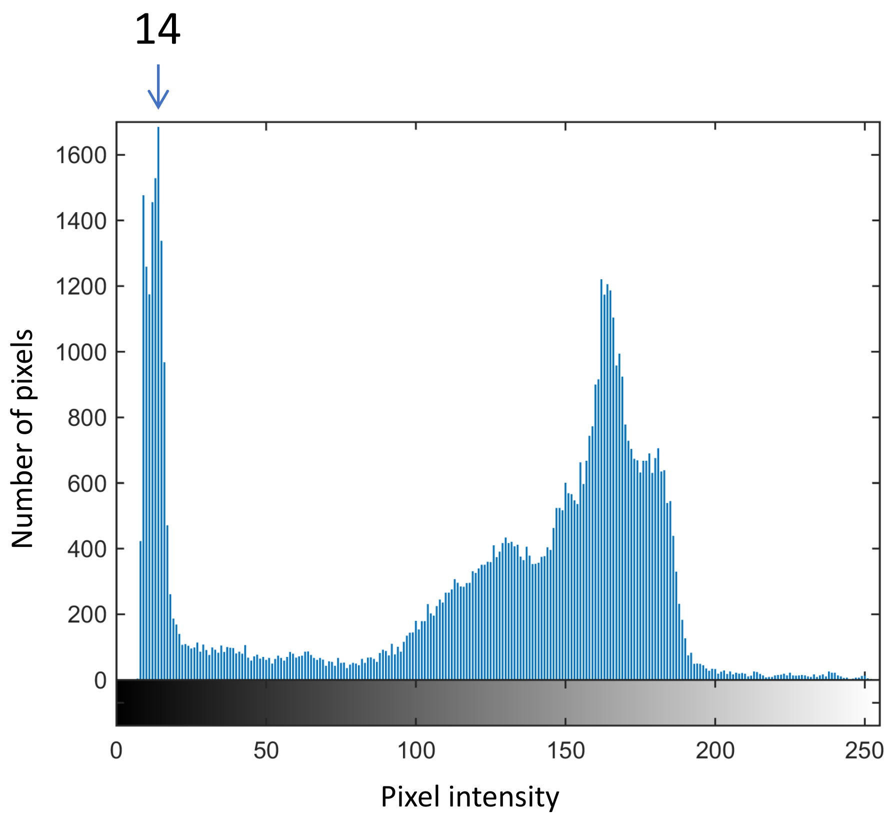
Pixel intensity	Number of pixels
0	0
1	0
…	
14	1685
…	
127	374
…	
254	0
255	0

There are 1685 pixels with intensity 14

Image histogram

Many image processing algorithms rely on the histogram!

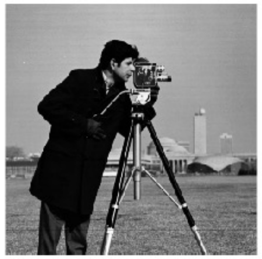
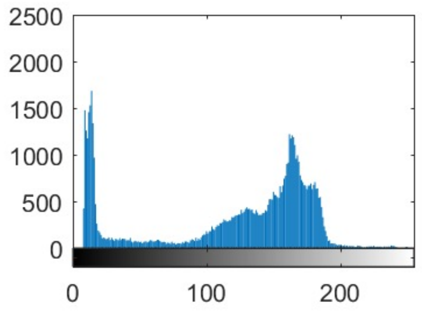
Pixel intensity	Number of pixels
0	0
1	0
…	
→ 14	1685 ←
…	
127	374
…	
254	0
255	0

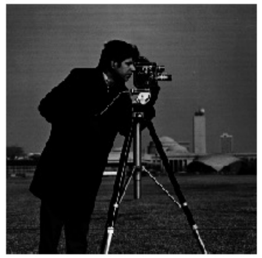
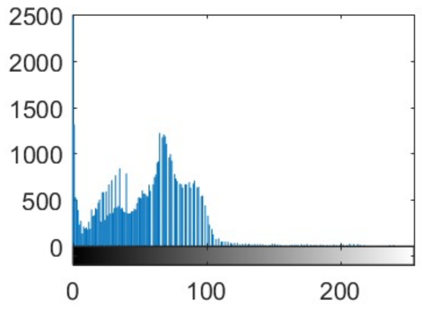


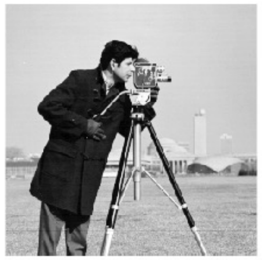
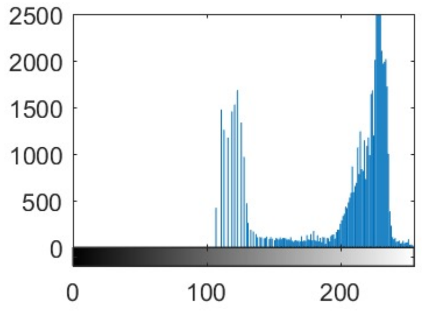
Histogram comparisons

The *shape* of the histogram can quickly reveal how the image looks like.

- If the histogram concentrates on the left, the image will be rather dark.
- If the histogram is skewed toward the right, the image will be rather light.
- A nicely spread histogram gives a nice contrast.

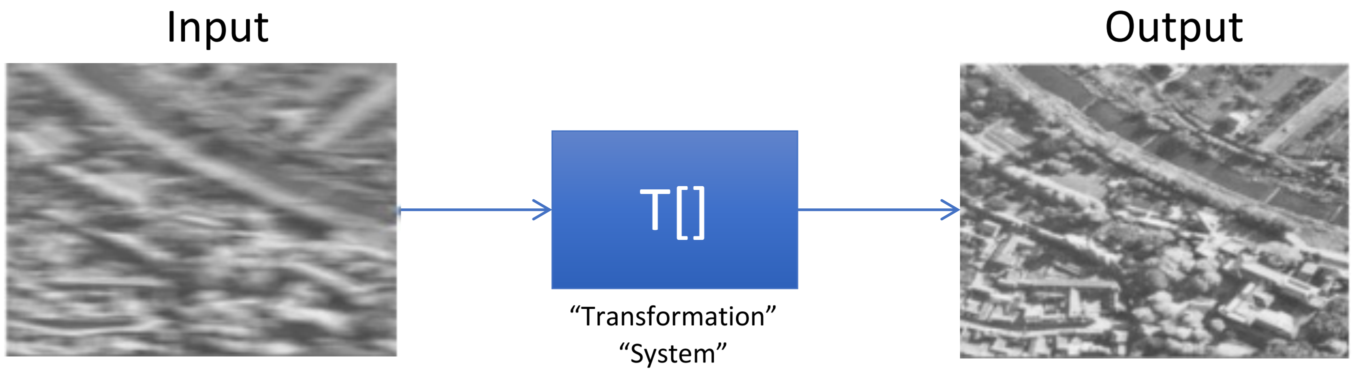



Fundamentals of image processing

We design our own systems!

In image processing, an input image is subjected to a *transformation* or passed through a *system* T to deliver an output image.



T[] can act:

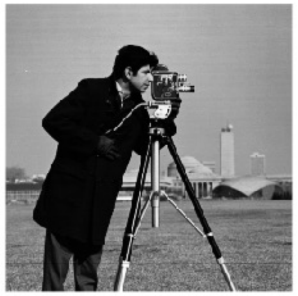
- on each pixel independently (point transformations)
- on a pixel "neighborhood" (*e.g.*, filters, morphology)
- on the whole image (*e.g.*, Fourier transform)

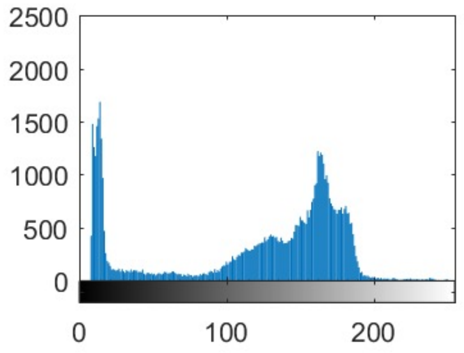
Point operations: Image negative

A **point operation** on a digital image is a function applied independently to every pixel in the image to create a new, modified image.

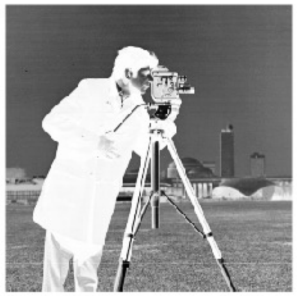
Input Transformation

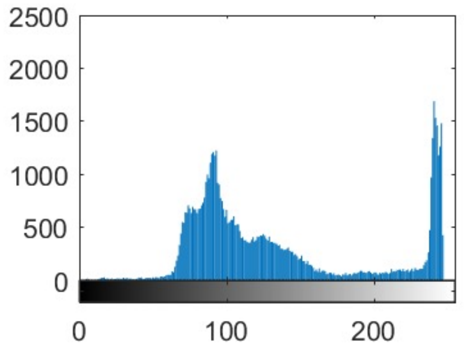
For an 8-bit image, the **image negative** replaces each pixel value $x(n_1, n_2)$ by $255 - x(n_1, n_2)$.





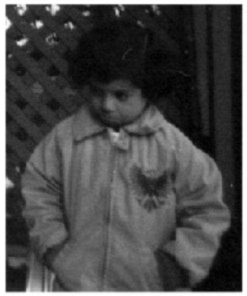
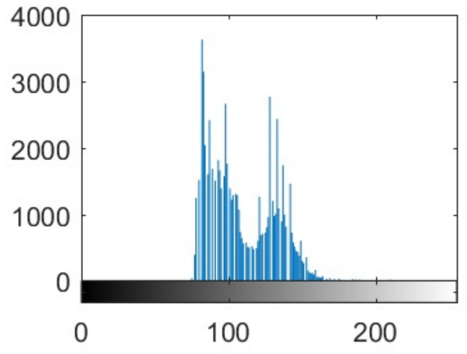
Output

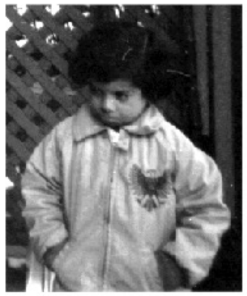
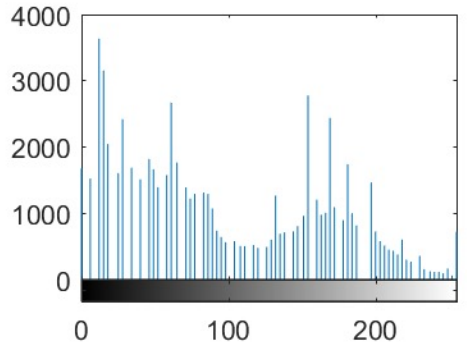




Point operations: full-scale histogram stretch

The **full-scale histogram stretch** (or **contrast stretch**) is a linear operation that expands the image histogram to cover the entire data type range.

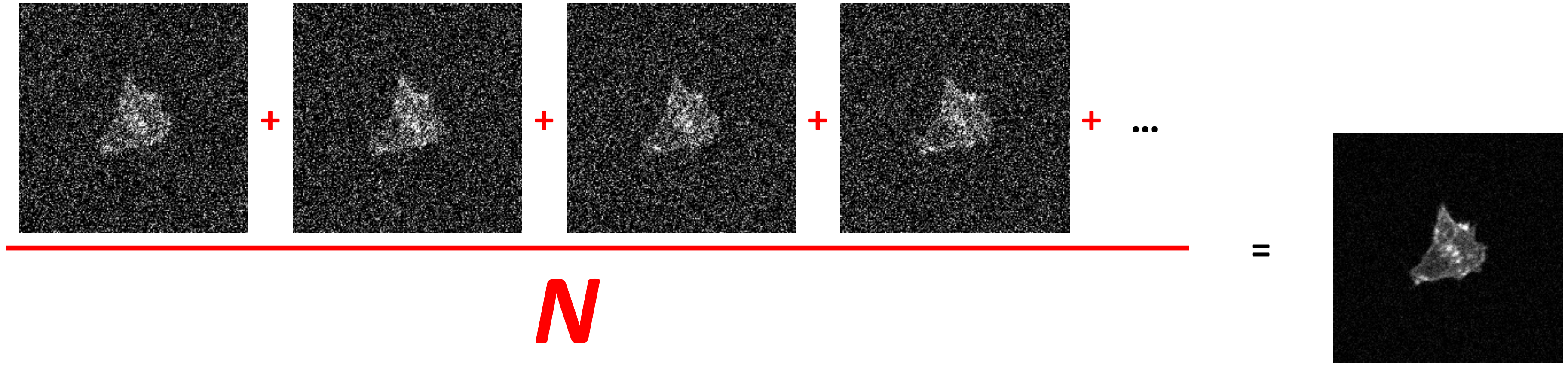



Point operations: arithmetic operations

Arithmetic operations are point operations (*i.e.*, pixel-wise) defined on multiple images. Images can be added, subtracted, multiplied, divided, …

image = signal + noise

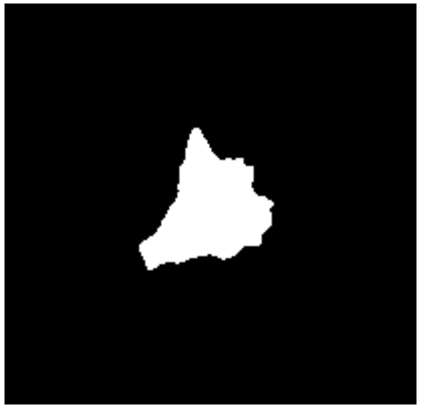
Example: Image averaging



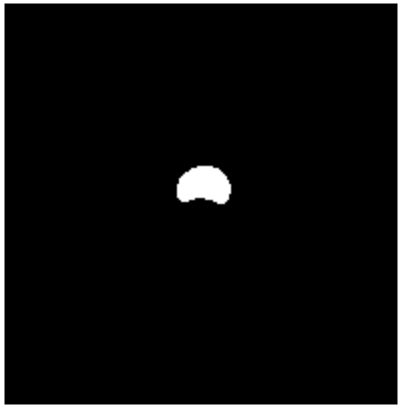
Images that suffer from low signal-to-noise ratio (SNR) can profit from averaging.

Point operations: arithmetic operations

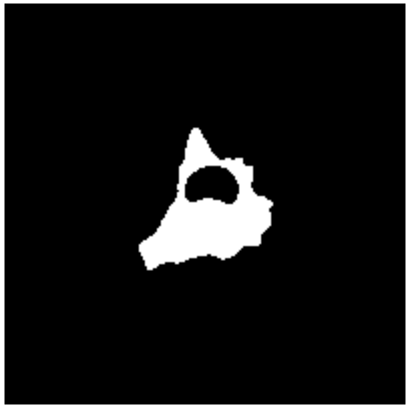
Example: Image masking


Whole-cell mask

−

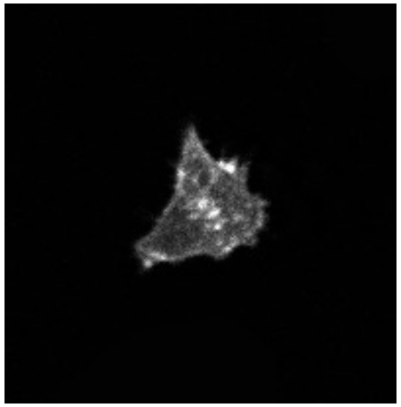

Nucleus mask

=

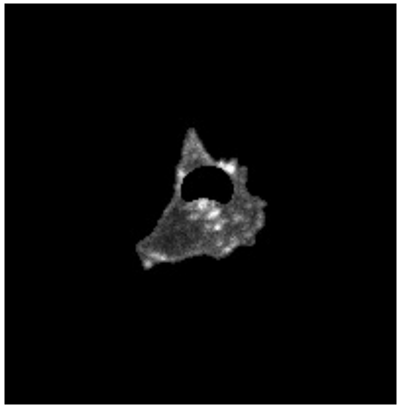

Cytoplasm mask

Cytoplasm mask

*

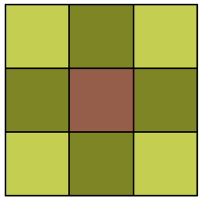

Original image

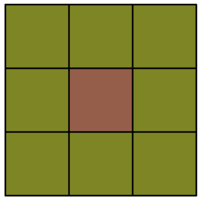
=


Masked image

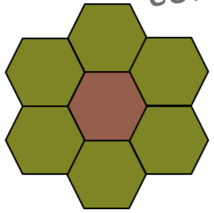
Pixel neighborhoods

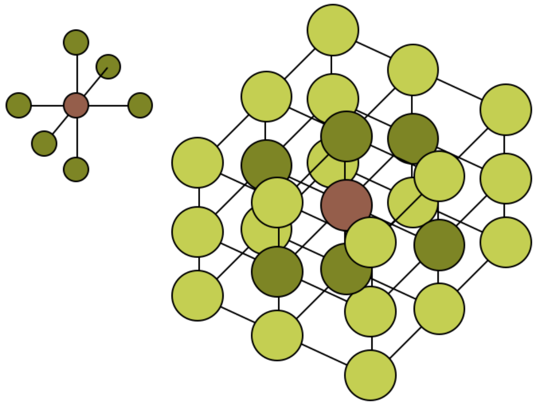
Transformations or systems can work on **pixel neighborhoods**. That is, the effect of the transformation on a pixel is not just a function of the pixel itself, but of several neighboring ones. Commonly and less commonly used neighborhoods are 4-, 8- and 6-connected (quite rare) in two dimensions and 6-, 18- and 26-connected neighborhoods in three dimensions.

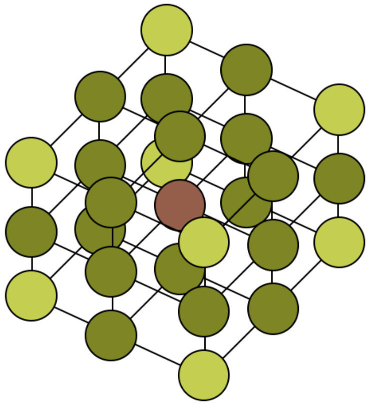

4-connected (2D)

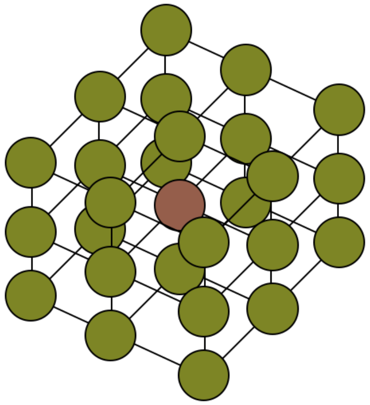

8-connected (2D)

computer games


6-connected (2D)


6-connected (3D)


18-connected (3D)


26-connected (3D)

(Convolution) kernels

An extension of the neighborhood is the **kernel** (also called **convolution matrix**). The kernel is a (small) matrix of numbers (**weights**) that is used in **convolution** (filtering).

3x3 kernel

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

5x5 kernel

w_1	w_2	w_3	w_4	w_5
w_6	w_7	w_8	w_9	w_{10}
w_{11}	w_{12}	w_{13}	w_{14}	w_{15}
w_{16}	w_{17}	w_{18}	w_{19}	w_{20}
w_{21}	w_{22}	w_{23}	w_{24}	w_{25}

Convolution

kernel

$$y(n_1, n_2) = x(n_1, n_2) \circledast \boxed{h(n_1, n_2)}$$

A kernel does not only define **which** neighbor pixels are important for the transformation, but also gives them **individual weights**.

Filtering

Filtering

- Filtering is used for *smoothing* an image, *i.e.*, **suppressing image irregularities** and **noise** while **preserving signal** as much as possible.

- In image processing, filtering is performed either in the **spatial domain** or in a **transform domain**.
  - The term spatial domain refers to the **image plane** itself, and spatial filtering **directly manipulates** the pixels in the image.
  - A transform domain first **converts the image into a different representation**, that is then manipulated before an **inverse transform** brings the result back into the spatial domain.
  - One very important transform domain is the **frequency domain**.

Spatial filtering

- A spatial filter consists of a **neighborhood** and a **predefined operation** applied to each pixel position in the input image to deliver a result to be stored at corresponding pixel position in the output image.

- If the operation applied to the neighborhood is **linear** (*i.e.*, is composed of multiplications and sums only), then the filter is called a **linear spatial filter**.

and the neighborhood is a kernel

- Otherwise, the filter is **non-linear.**

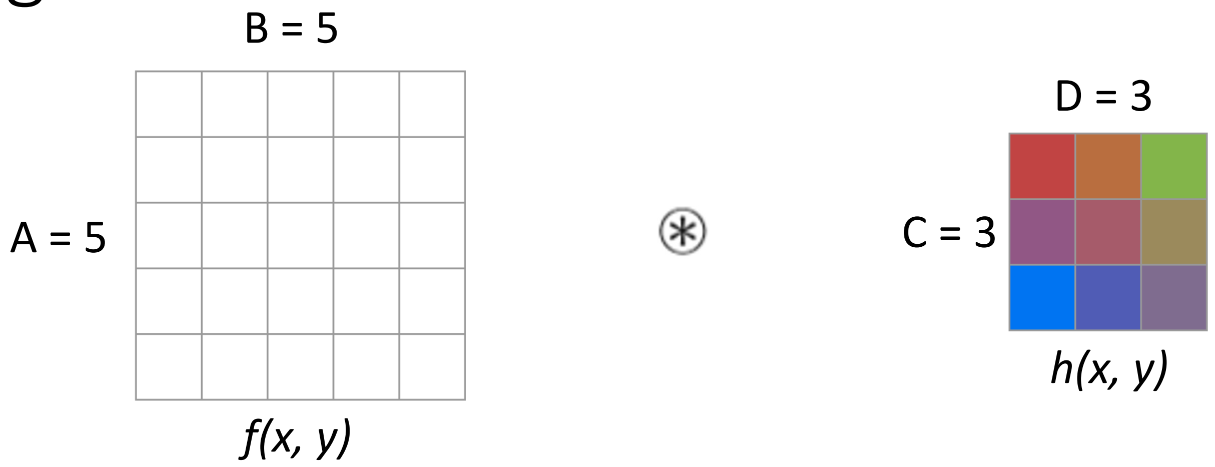
Linear filters through convolution

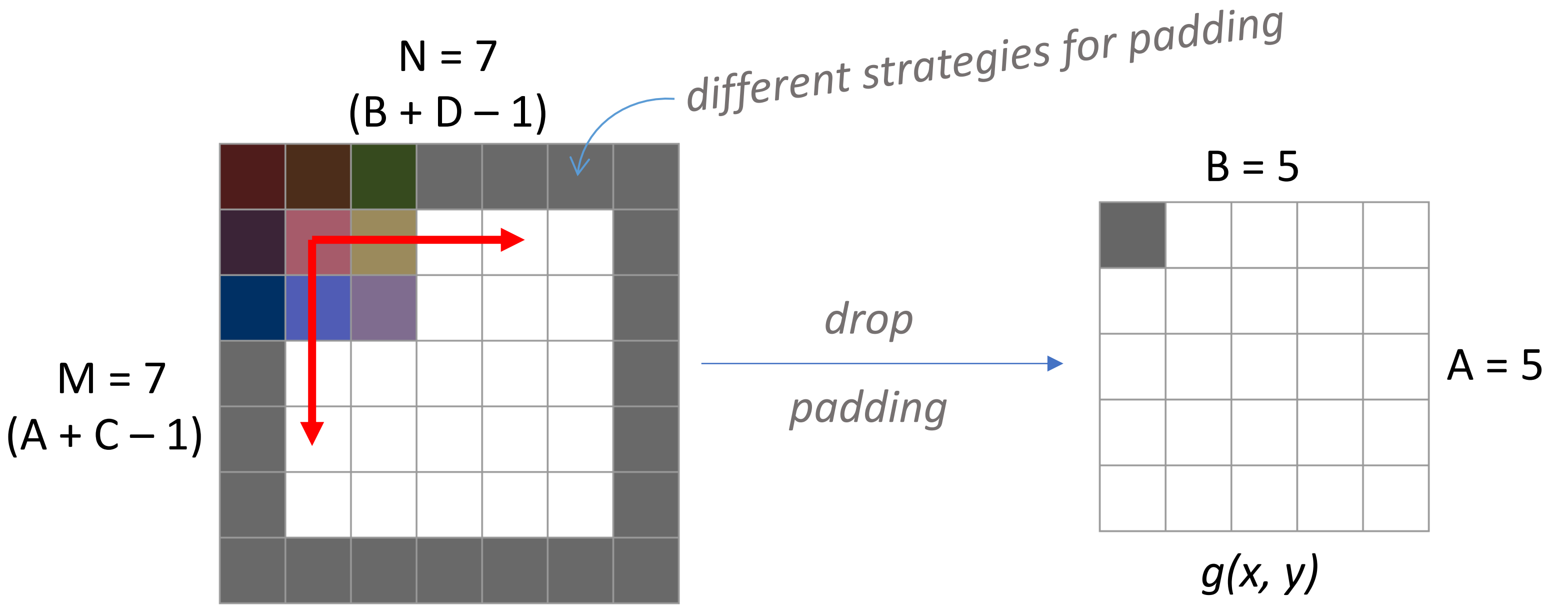
- Linear filters in the spatial domain are implemented via the operation of **convolution**.
- Given an image f(x, y) of size A x B and a **kernel** (or **mask)** h(x, y) of size C x D, the convolution is defined as:

$$f(x,y) \circledast h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)h(x-m,y-n)$$

- M = A + C – 1 and N = B + D – 1 give the full convolution result.
- In practice, the borders with partial kernel support are discarded, and the central A x B pixels of the convolution results are preserved.
- The **size** of the kernel defines the **support** of the filter.

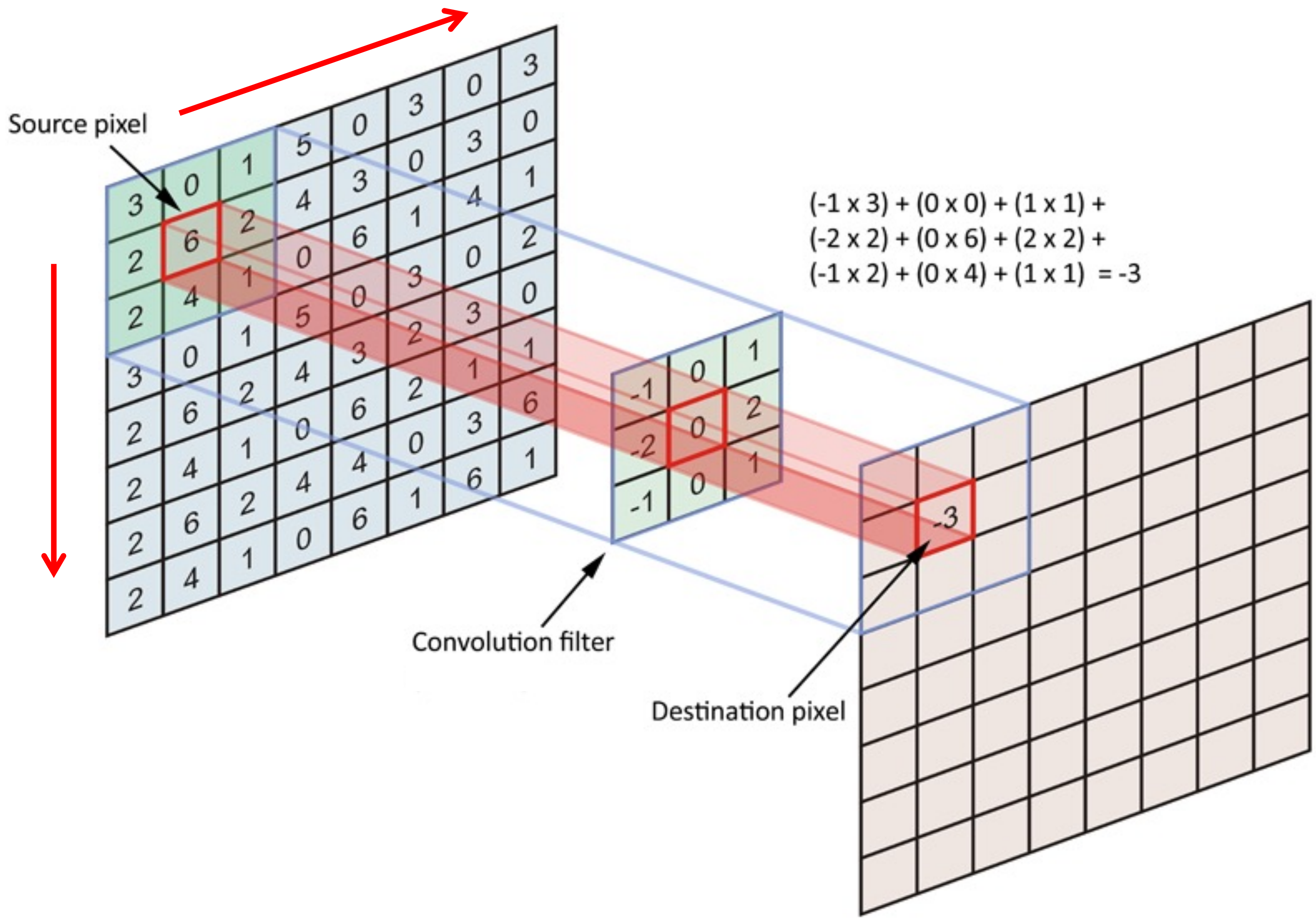
Padding





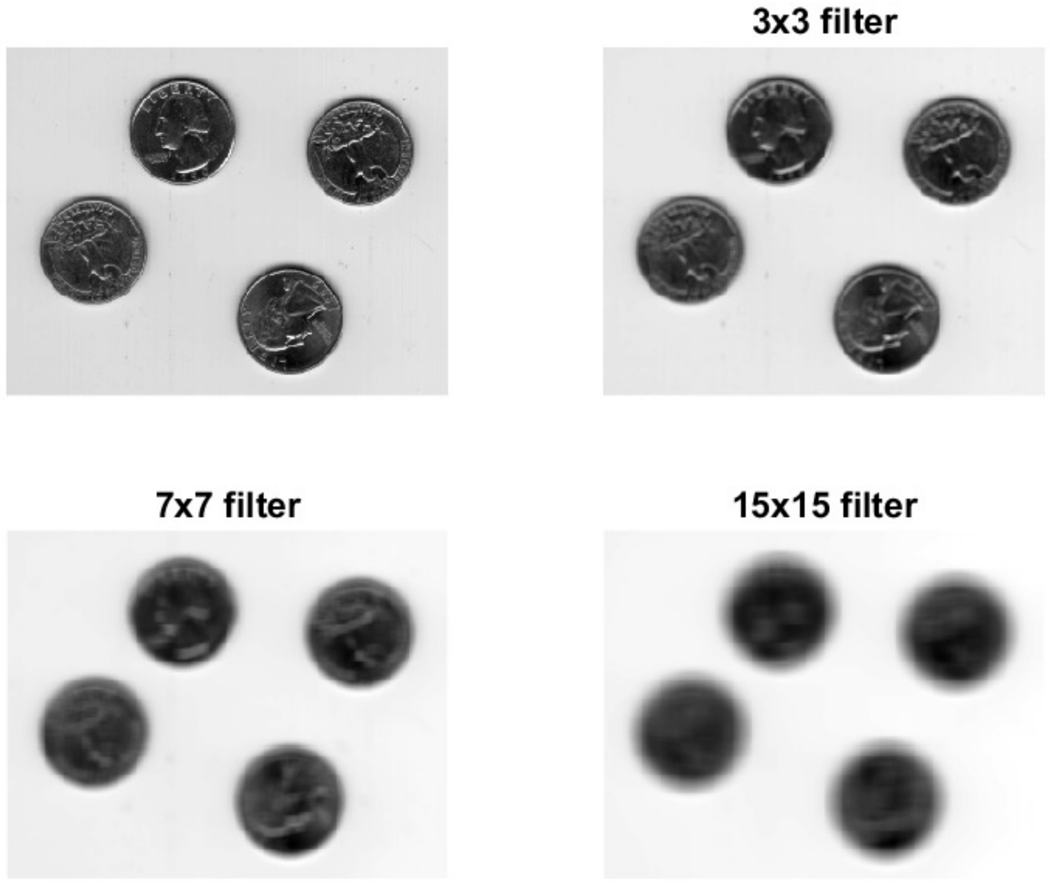
https://ezyang.github.io/convolution-visualizer/

Convolution scheme


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
Destination pixel

Linear filters: kernel size

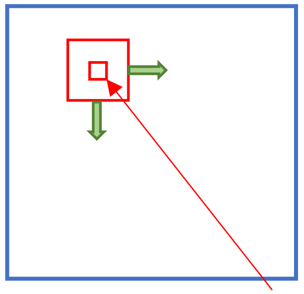
The **size** of the kernel defines the **support** of the filter. Large kernels will have a stronger filtering effect on the image than corresponding kernels of smaller size.



Linear filters: average filter

The **average filter** calculates for each pixel (*x, y*) in the input image the average of all pixel intensities in the neighborhood and stores this value at the same position (*x, y*) in the target image.

$$\frac{1}{9}\left(f(x-1,y-1)+f(x-1,y)+f(x-1,y+1)+f(x,y-1)+\cdots+f(x+1,y)+f(x+1,y+1)\right)$$

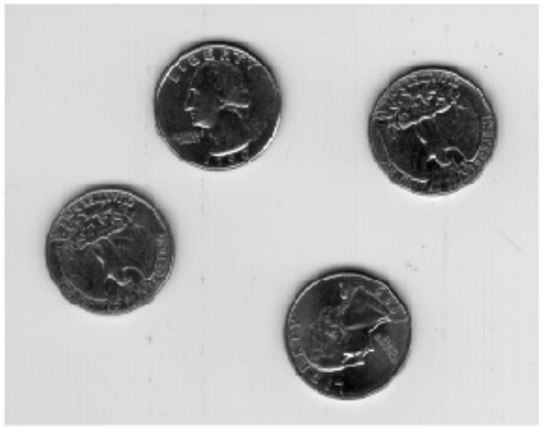


$$h=\frac{1}{9}\cdot\begin{bmatrix}1&1&1\\1&\textcircled{1}&1\\1&1&1\end{bmatrix}=\begin{bmatrix}\frac{1}{9}&\frac{1}{9}&\frac{1}{9}\\\frac{1}{9}&\frac{1}{9}&\frac{1}{9}\\\frac{1}{9}&\frac{1}{9}&\frac{1}{9}\end{bmatrix}$$

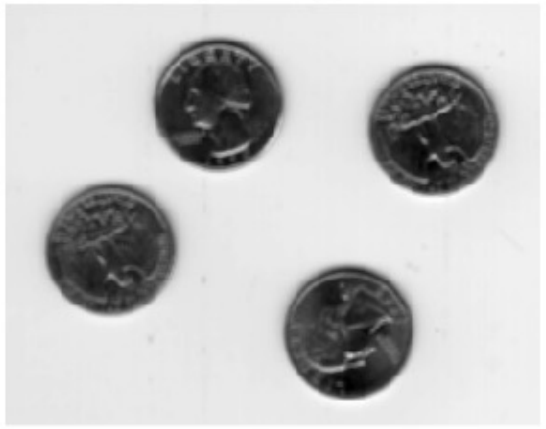
weights

f(x,y)

If the weights are different, we obtain a **weighted average** filter.



3x3 filter



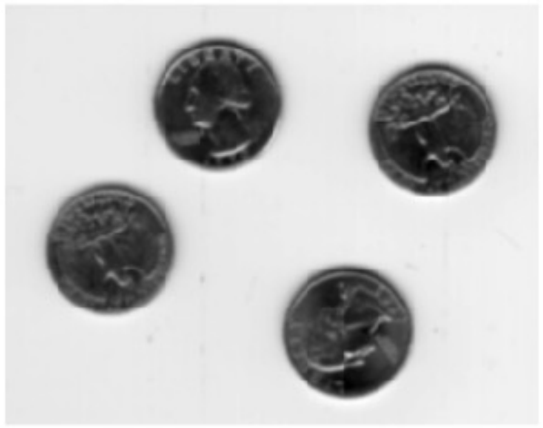
Linear filters: Gaussian filter

In contrast to the average filter, the Gaussian filter can better be adapted to preserve features of a given scale (or size). The weights of the Gaussian kernel are calculated as follows:

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

$$h = \begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

5x5 Gaussian filter, σ=1



Segmentation

Segmentation

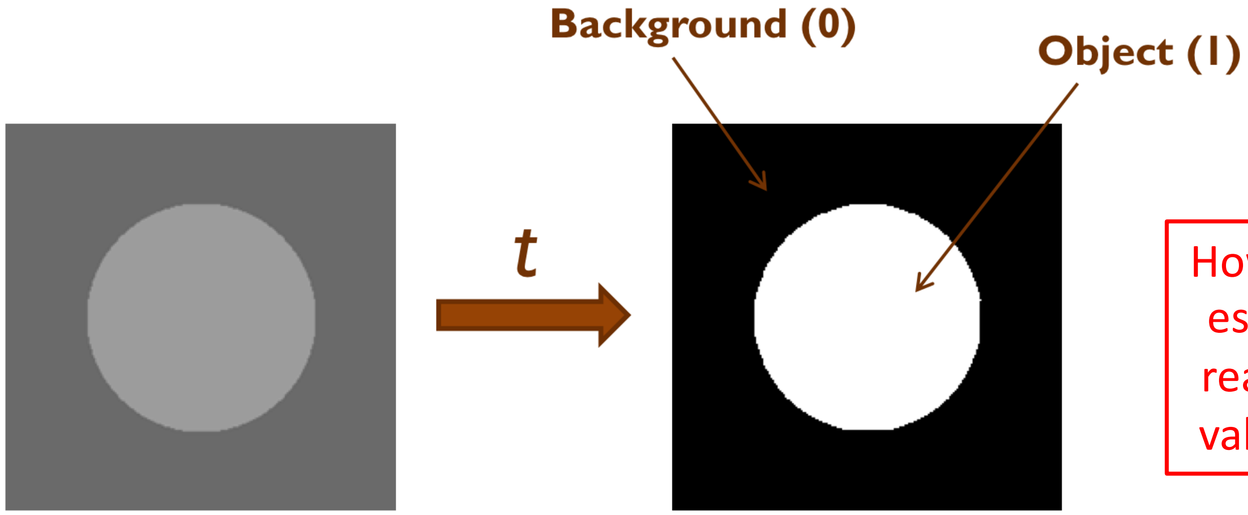
- **Segmentation** is one of the most fundamental image processing techniques. Its practical goal is to extract *interesting parts* from an image for subsequent analysis and operations.
- More precisely, segmentation partitions a digital image into **segments** (also called **super-pixels**) that are homogeneous in some characteristics (such as brightness, color, texture, motion, …).
- Segmentation is usually the first step of a series of operations performed in the **analysis** of an image.
- Segmentation is used to locate **objects** or their **boundaries**.
- There is no general solution to the image segmentation problem.
- There are dozens of approaches to segmentation. One goes from the simplest forms of **thresholding** all the way to **(deep) neural network algorithms** through **histogram-based** methods, **clustering** methods, **edge detection**, **watershed transforms**, **region growing**, **level sets**, **statistical methods**, …

simplest segmentation algorithm

Thresholding

Thresholding is most commonly and effectively applied to images that can be characterized as having *bimodal histograms* (*e.g.,* light objects against a dark background). Individual pixels in an image are marked as object pixels if their value is greater than some **threshold** value *t.*

$$B(x,y) = \begin{cases} 1, & if\ I(x,y) > t \\ 0, & if\ I(x,y) \le t \end{cases}$$



Original image I(x,y)

(ideal, noise-free)

Binary image B(x,y)

How can we estimate a reasonable value for *t*?

one of many algorithms

Iterative Thresholding

Iterative thresholding is a very simple algorithm that works pretty reliably without much external tuning and is rather robust against noise.

Algorithm:

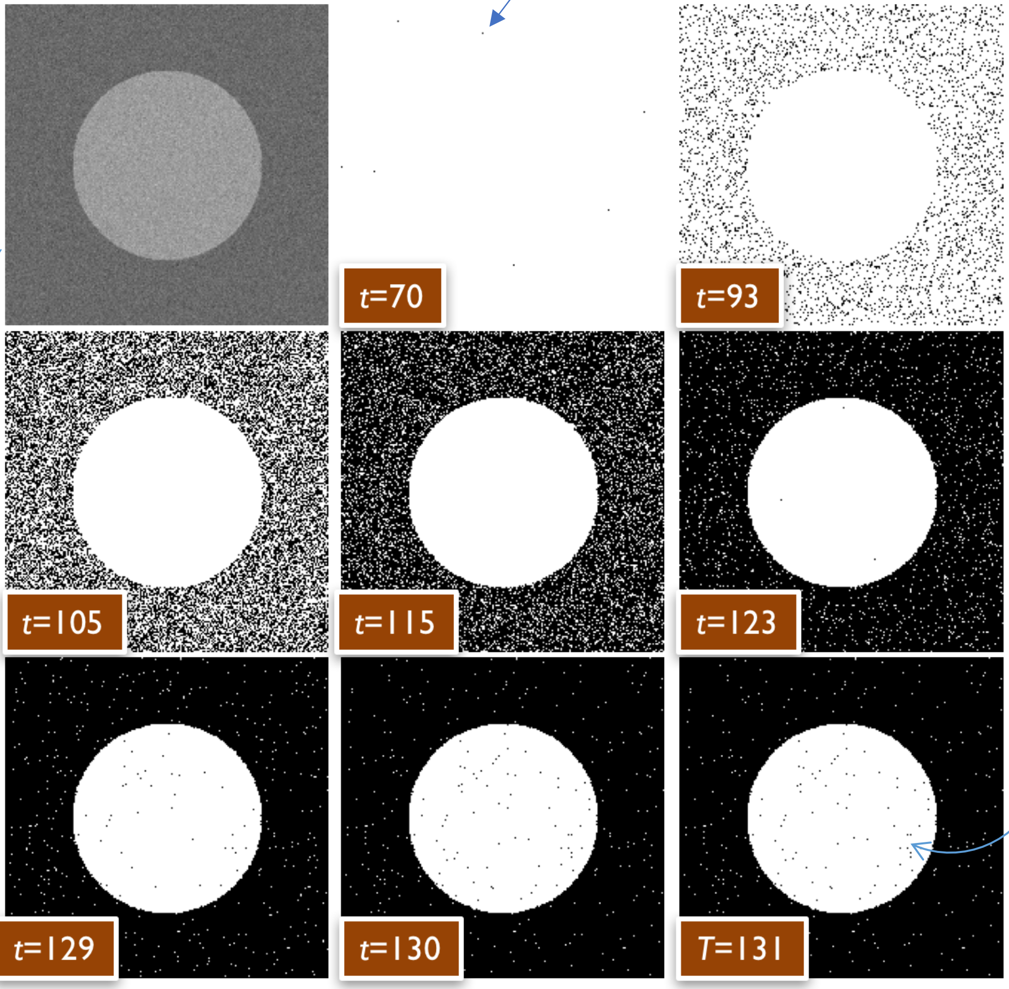
- Choose an initial threshold t, either randomly or according to any method desired
- Segment the image I into two sets:
  - $G_1 = \{I > t\}$
  - $G_2 = \{I \leq t\}$
- Calculate the average of each set:
  - $m_1 = \texttt{mean}\left(G_1\right)$
  - $m_2 = \texttt{mean}\left(G_2\right)$
- Update the threshold t as the average of m_1 and m_2
  - $t = \left(m_1 + m_2\right)/2$
- Repeat steps 2 - 4 until the new threshold t matches the one in the previous iteration (*i.e.*, the algorithm has *converged*)

Iterative Thresholding

background pixel

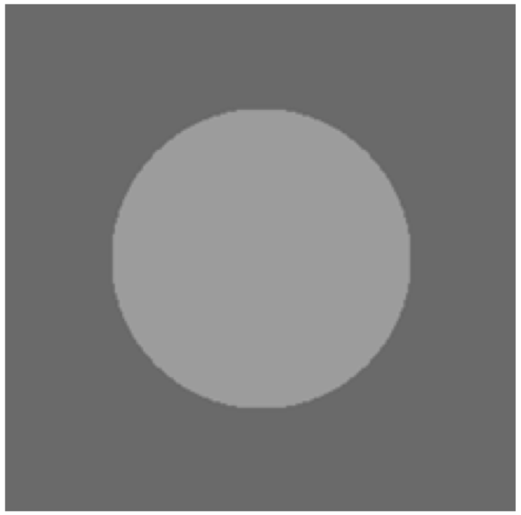
slightly noisy image

some pixels in the object are classified as background



Histogram-based segmentation

Histogram-based segmentation uses the distribution of the pixel intensities to segment the image. To be more precise, it uses the *shape* of the histogram to locate the clusters in the image. In the case of a noise free image with two peaks (*modes*) in the histogram, the selection of the threshold is trivial.


Ideal, noise-free image

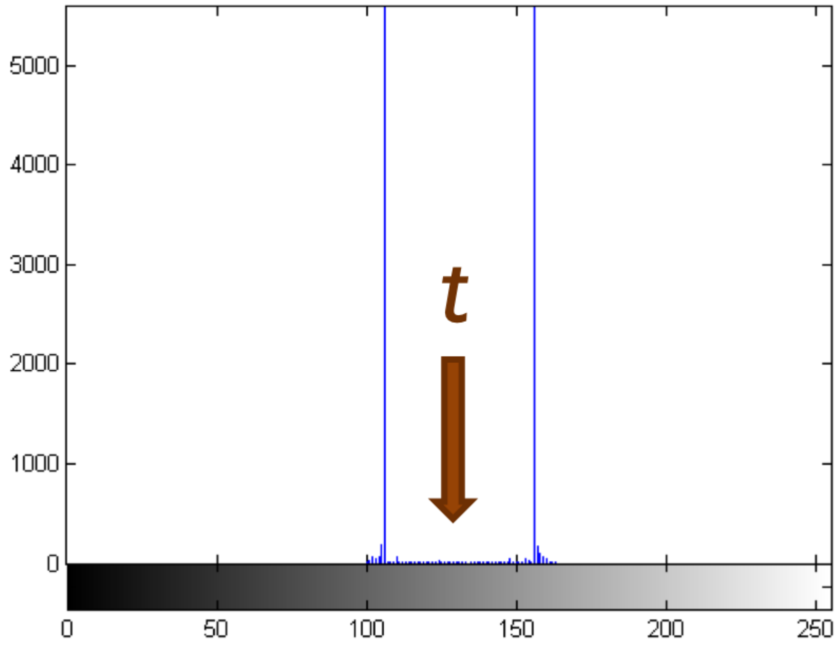
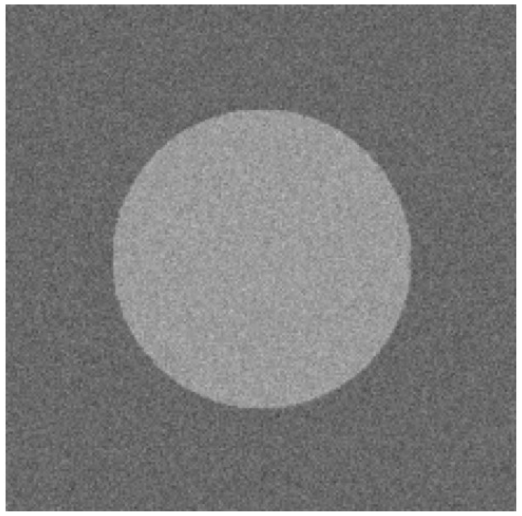


Image histogram

Histogram-based segmentation

In the case of moderate noise but still with two clear peaks (*modes*) in the histogram, the selection of the threshold is easy.


Low-noise image

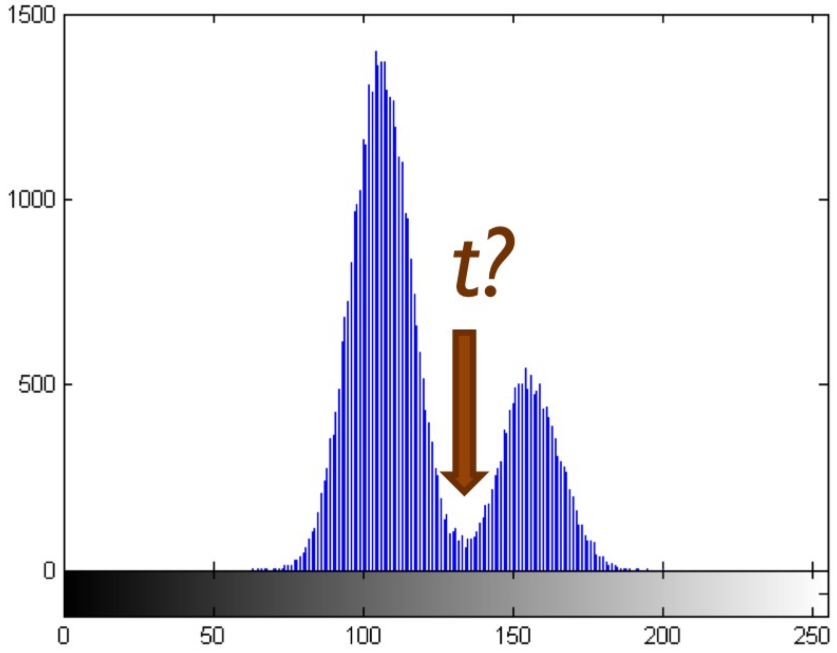
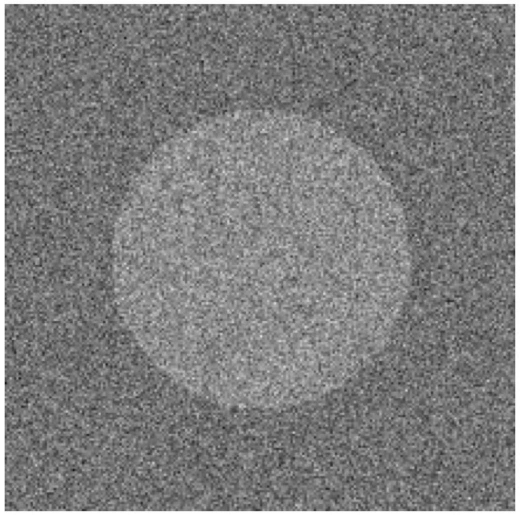


Image histogram

Histogram-based segmentation

In the case of very noisy images, it is not longer clear where the threshold should be.


High-noise image

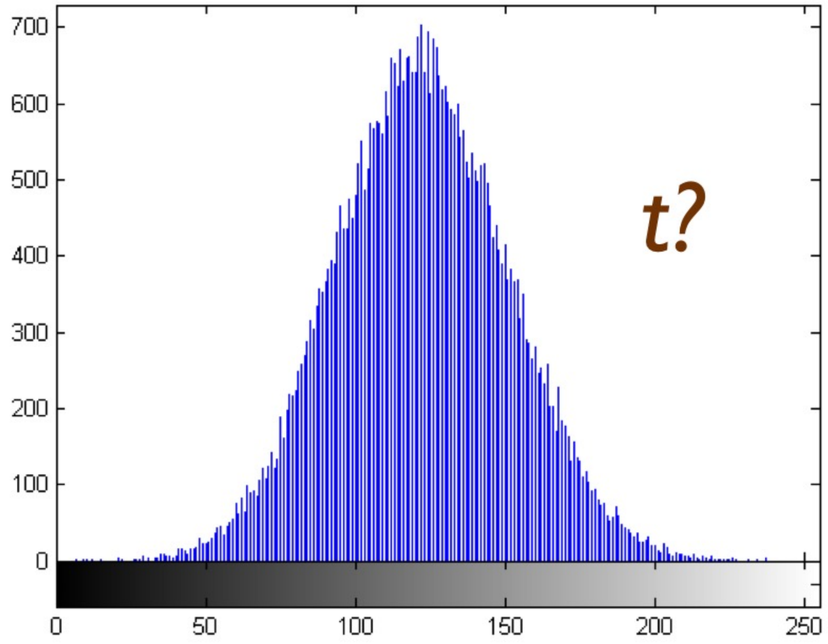
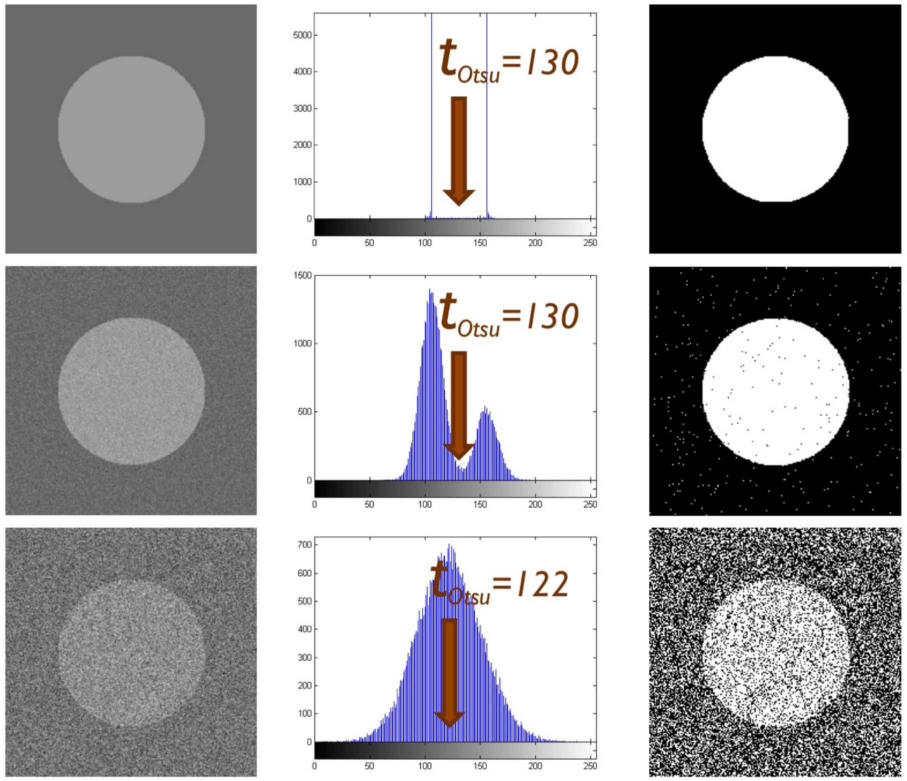


Image histogram

Otsu’s method

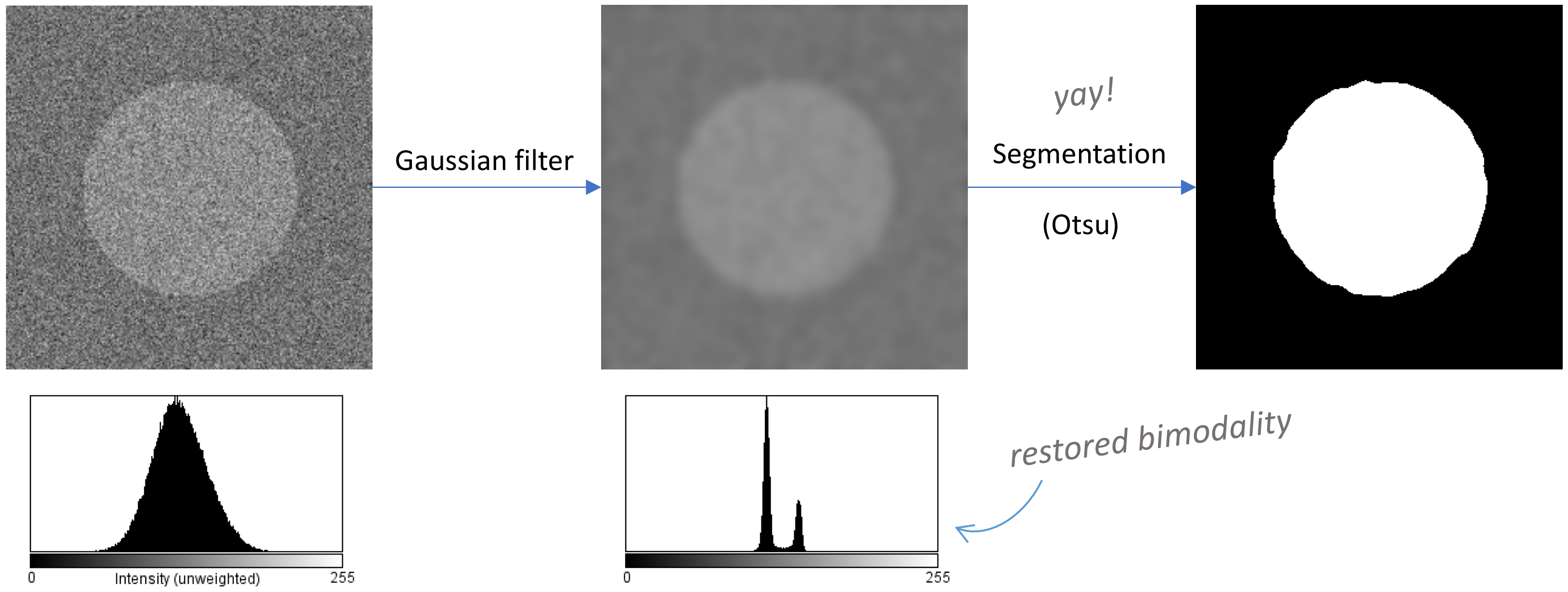
One common algorithm for histogram-based segmentation is Otsu’s method. The algorithm assumes that the image to be thresholded contains two classes of pixels (*e.g.*, foreground vs. background) then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal.

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \qquad \omega_0 = \sum_{i=0}^{t-1} p(i), \omega_1 = \sum_{i=t}^{L} p(i)$$



roughly half of the pixels in the central area are assigned to the background...

Filtering to the rescue



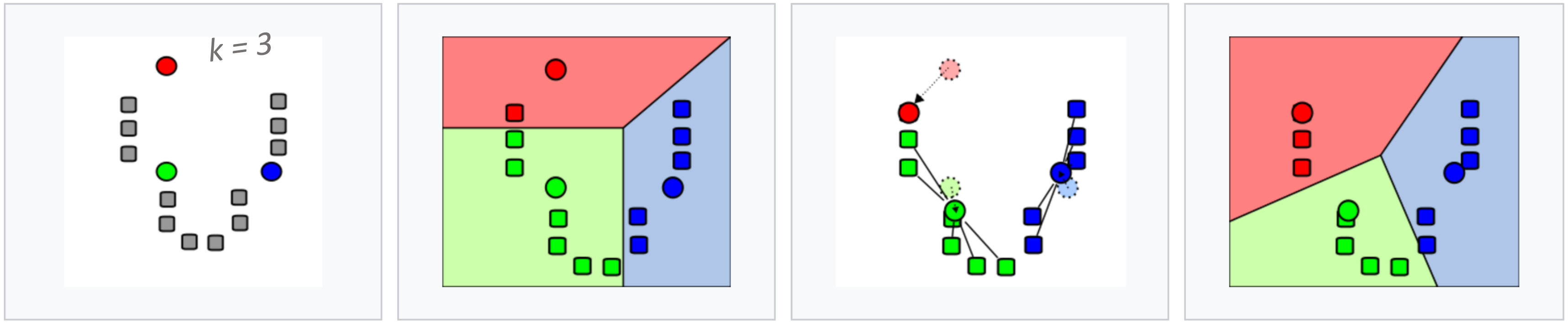
How does the Gaussian filter know what is the underlying signal distribution?

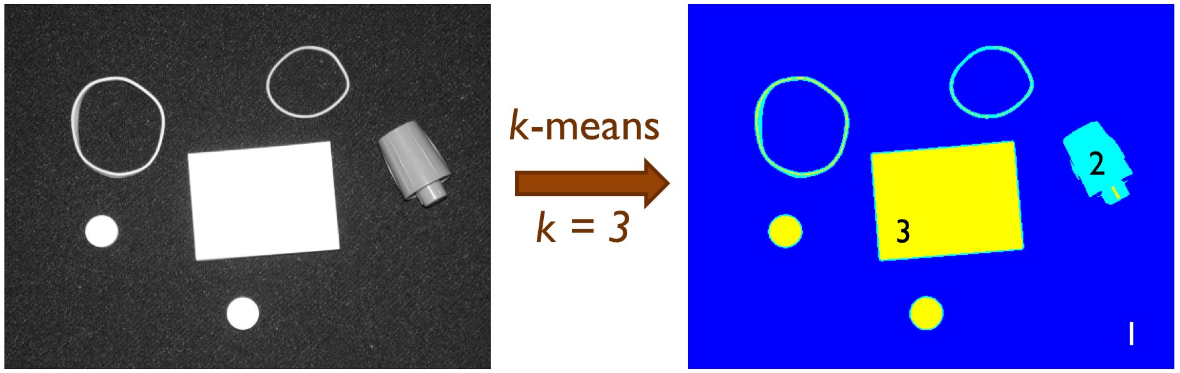
k-means clustering

more classes!

k-means clustering aims to partition *n* observations into *k* clusters (in *N* dimensions) in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

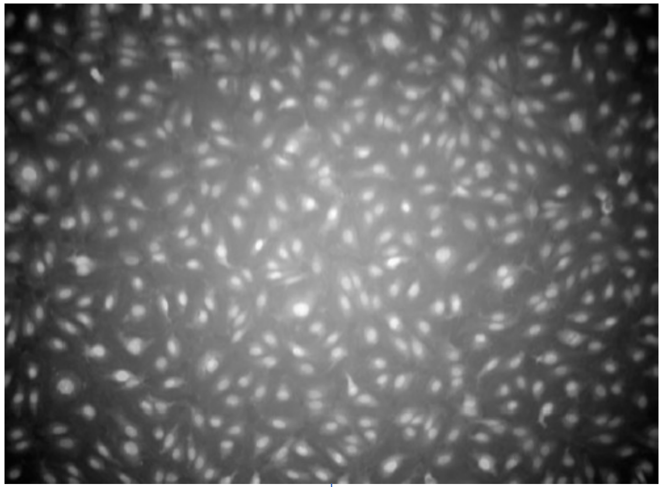
2D example



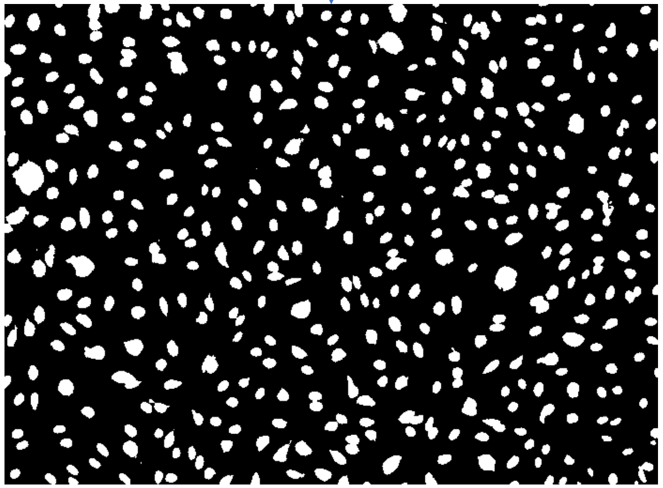


k-means in 1D (intensity)

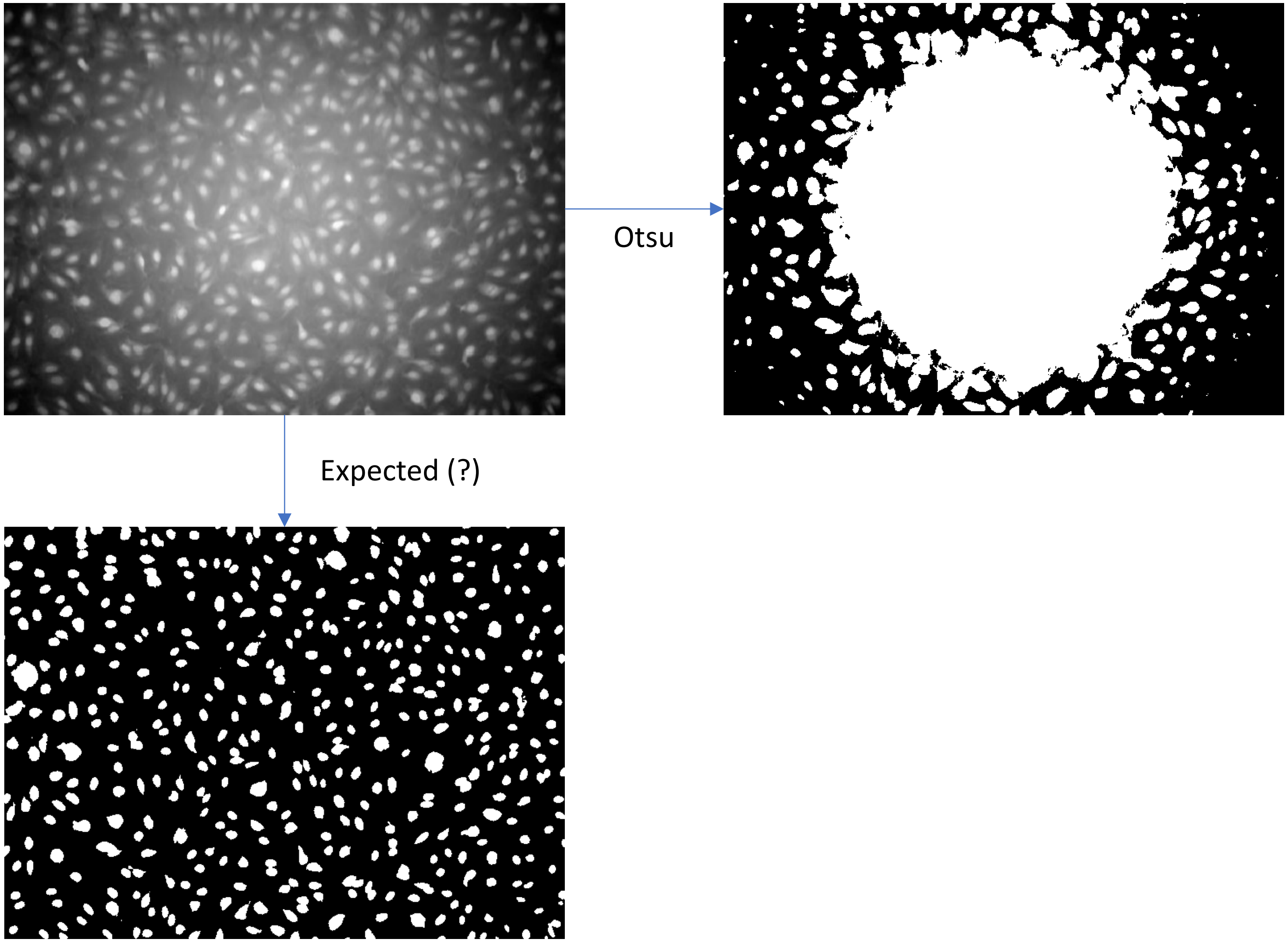
Background subtraction



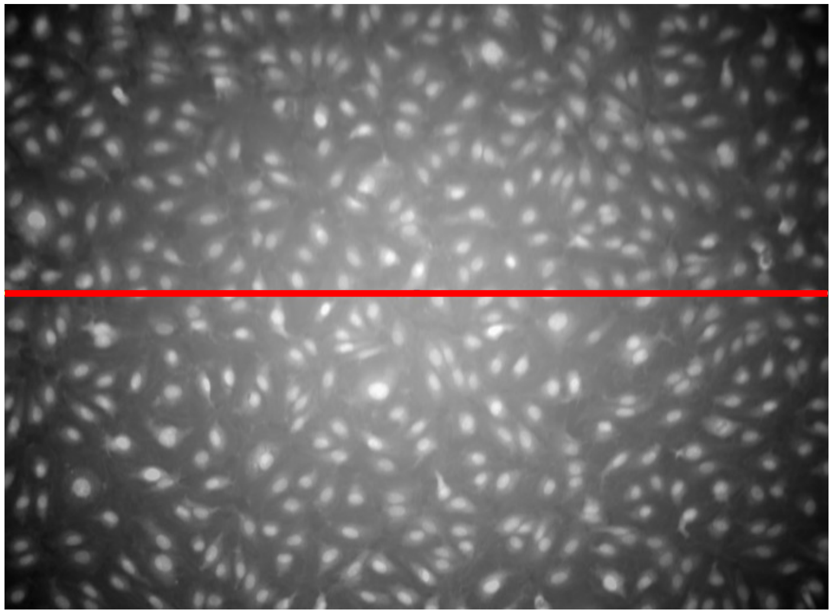
Expected (?)

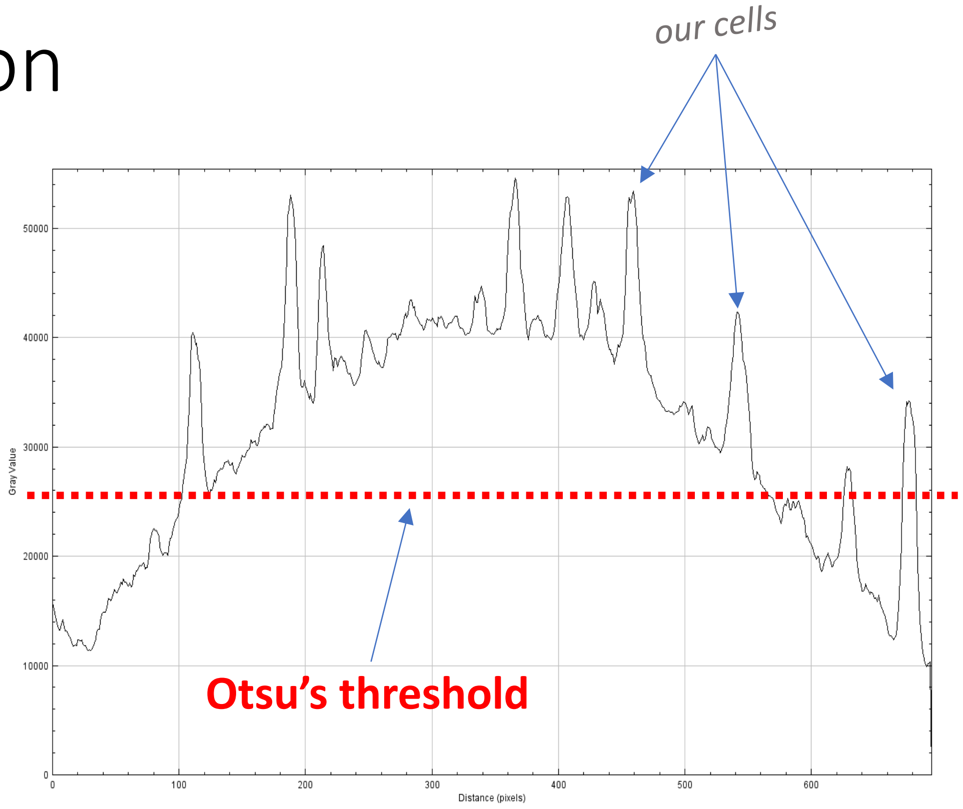


Background subtraction



Background subtraction



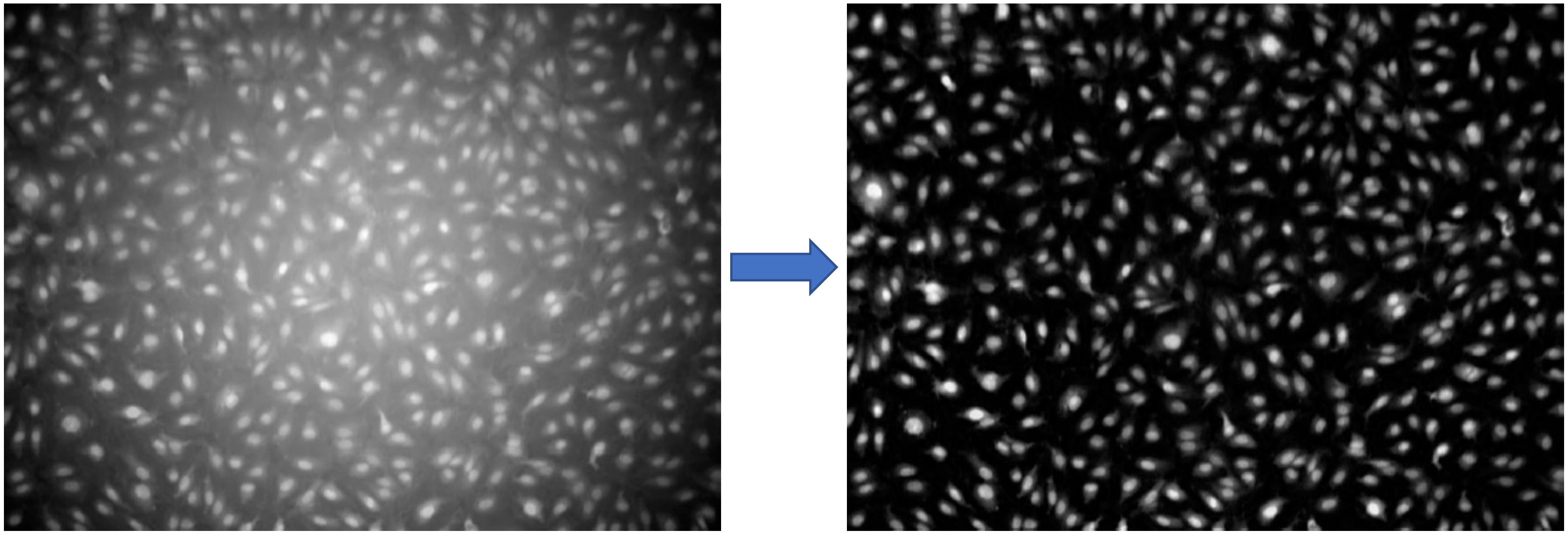


this kind of background shading is common in microscopy (maybe not as bad as this!)

The image background is not flat!

Background subtraction

now Otsu will work fine!

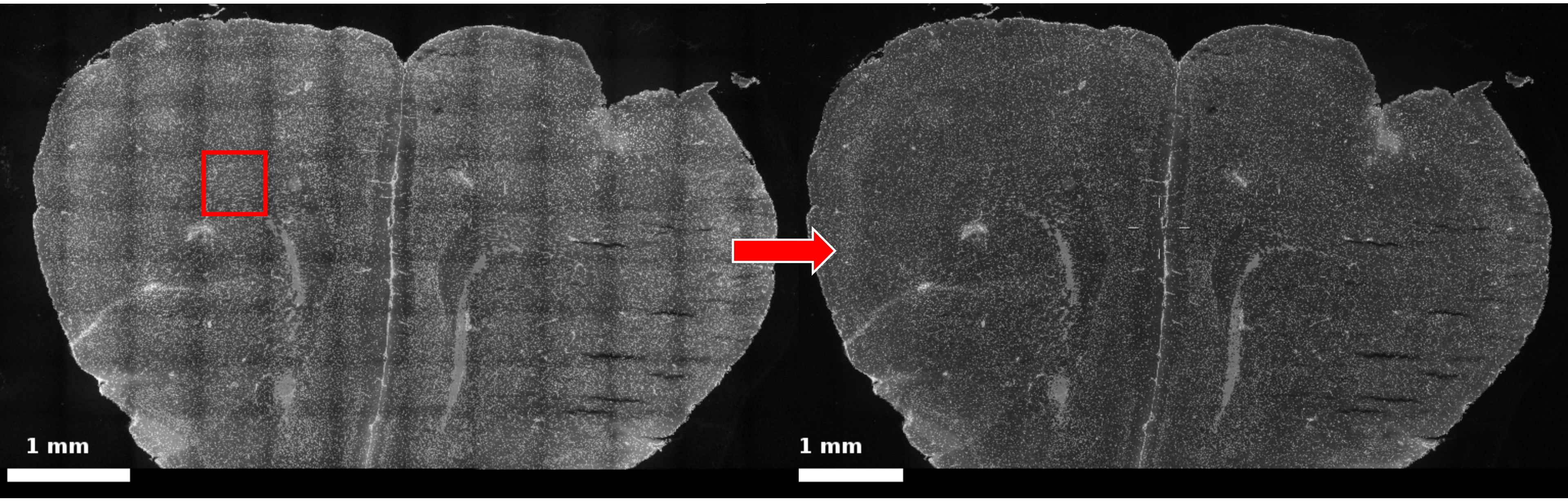


- **Acquire a background image** and subtract it from the image to correct
- **Estimate the background** from the image itself and subtract it:
  - **Rolling ball algorithm**
  - **Morphological opening**
  - **Low-pass filter with large Gaussian kernel**

In all estimation cases, the trick is to use an area (or kernel) that is larger than the details in the image (*i.e.*, the cells), thus leaving an estimate of the background that can be subtracted.

Vignetting

Many microscopy images suffer from uneven illumination of the field, with poorer illumination at the image periphery. This so-called **vignetting** problem is due to factors in the microscope light path, such as limitation in the lenses or in the camera. The vignetting effect becomes particularly evident when 2D or 3D tile images are **stitched** together.

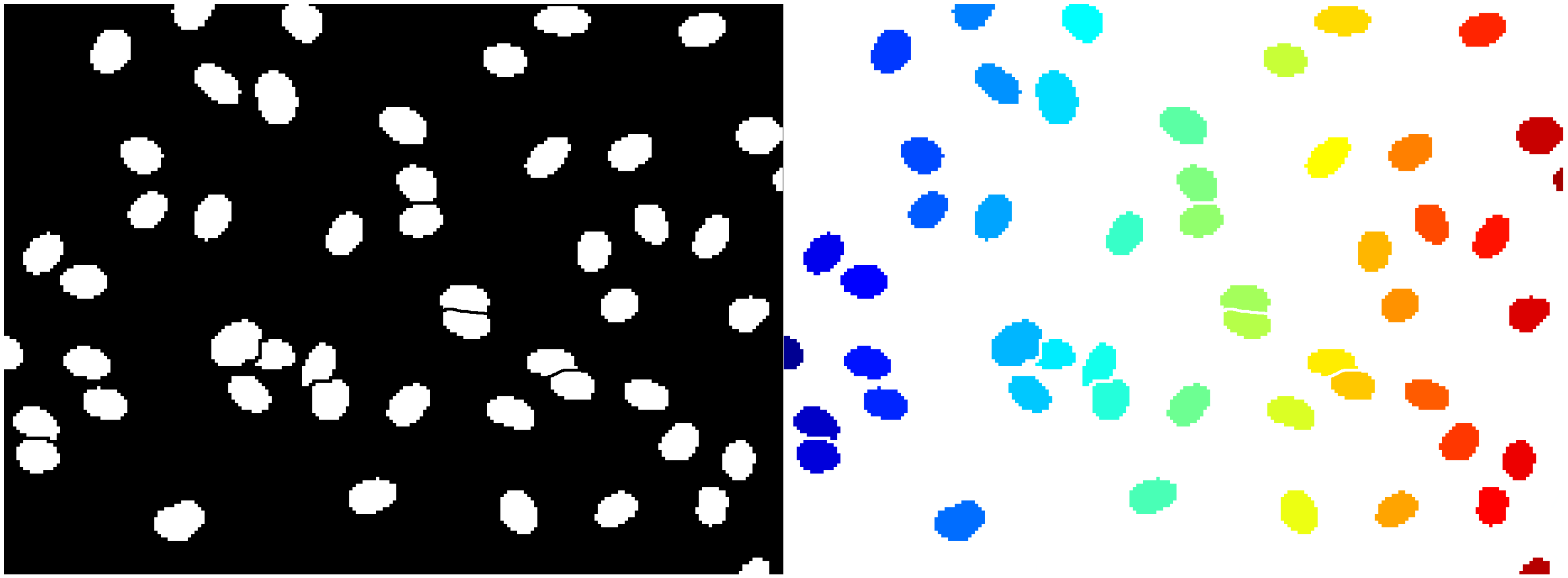


background subtraction helps in reducing the effects of vignetting

Taken with modifications from https://svi.nl/Vignetting.

Connected components

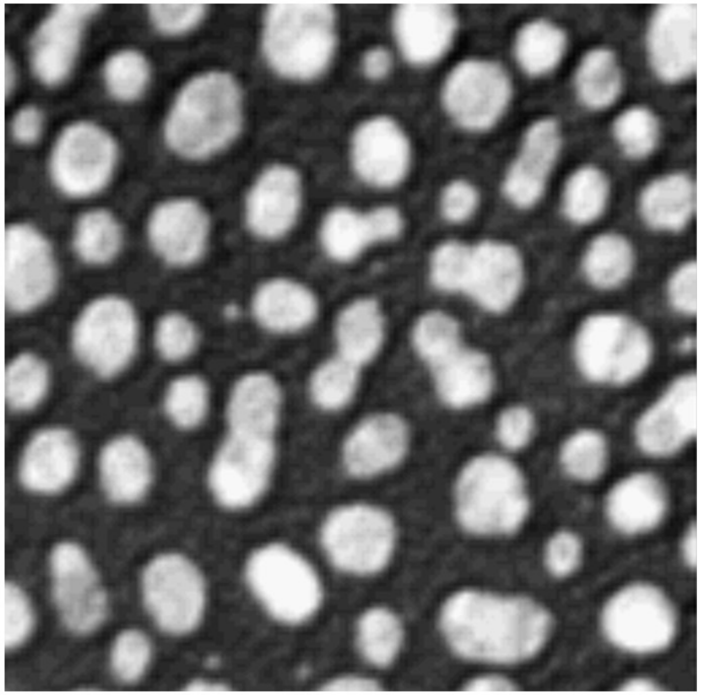
- The result of a segmentation is usually a **binary** image with all pixels belonging to the **background set** having value 0 and those belonging to the **foreground set** having value 1.
- For **analysis** purposes, we want to break the foreground set into **individual objects** (spatially separate), using a **connected component** algorithm.
- We can now extract **features from these objects** for further analysis.

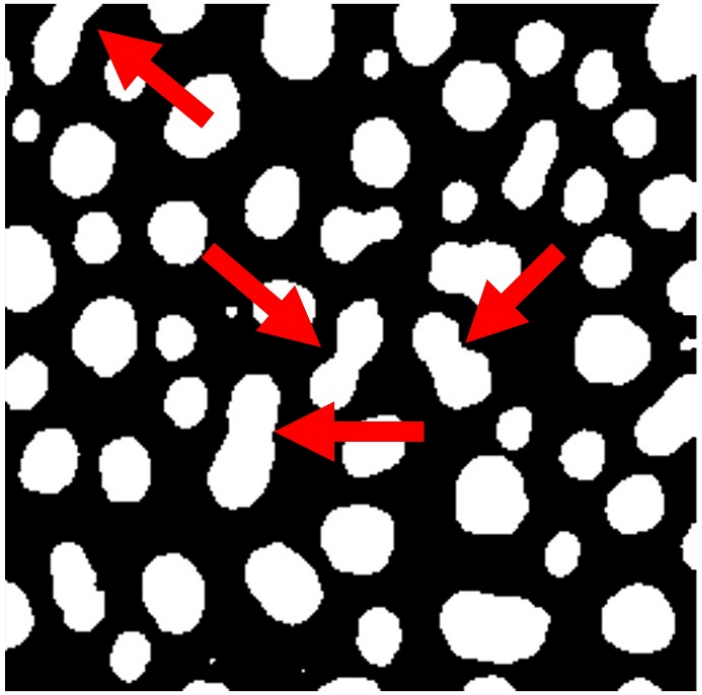


Morphological operations

Watershed

Often, objects in an image that are close to each other are difficult to segment and are fused into individual blobs in the binary image.

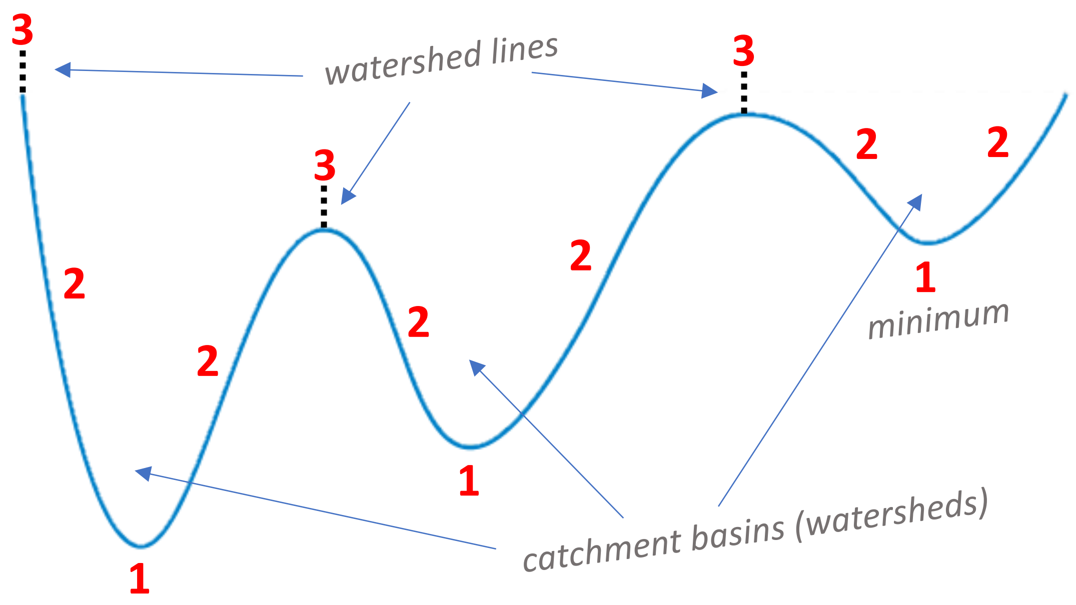




The watershed segmentation algorithm makes use of the *shape* of the objects to separate them.

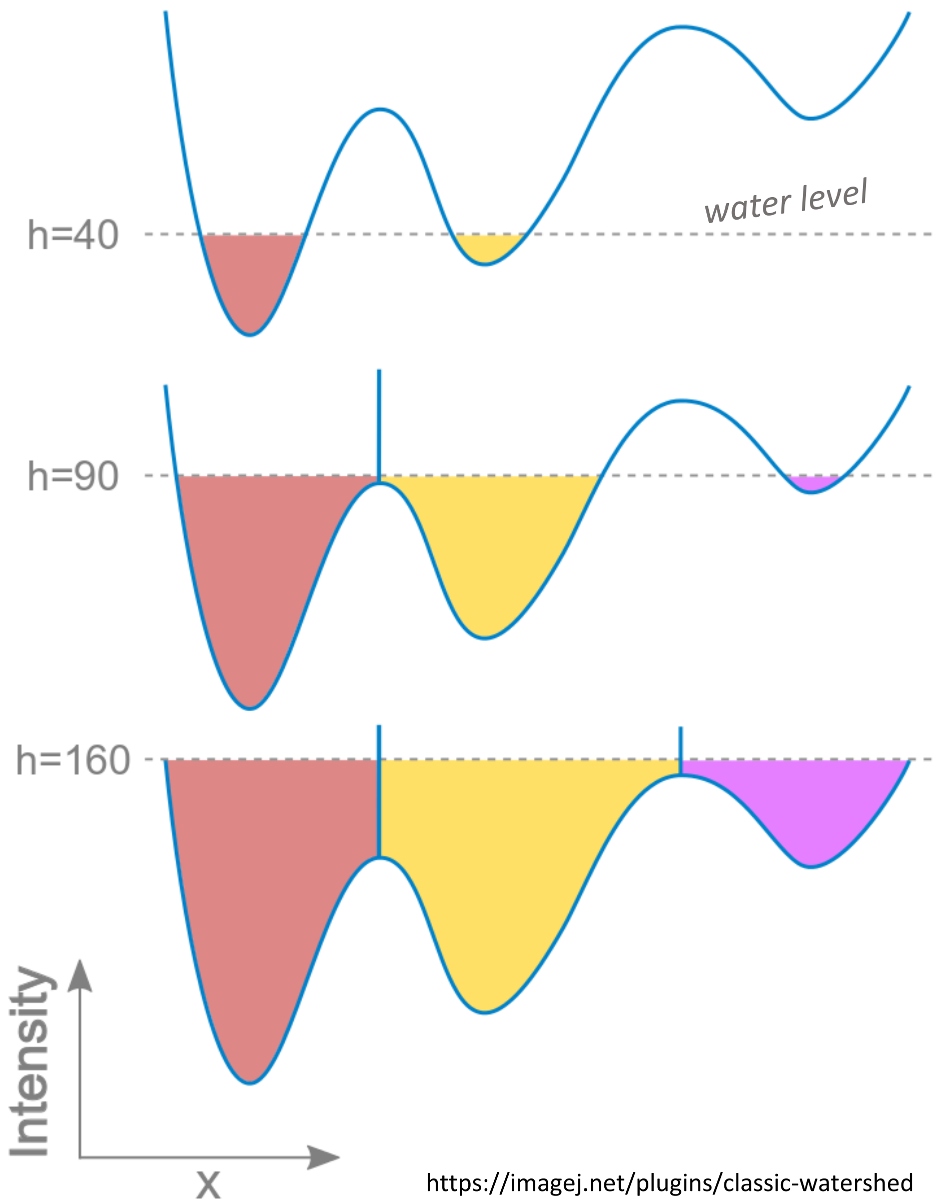
Watershed

- The idea behind watersheds is to **transform** the **binary** image into a **3D landscape**, where the new pixel value corresponds to the landscape elevation.
- In a rainy day, we can define three sets of points in this landscape:
  1. points at a regional minimum
  2. points where a raindrop would slide down towards one and only minimum (**catchment basin** or **watershed** for each minimum)
  3. points where a raindrop would be equally likely to fall to more than one such minimum (**divide lines** of **watershed lines**).



Watershed

- The principal idea of the watershed transform is to **find the watershed lines**.
- Algorithm:
  - a hole is punched at each of the local minima in the topographic surface
  - the entire topography is flooded from below by letting water raise through the holes at constant rate.
  - when the rising water from distinct catchment basins is about to merge, a dam is built to prevent the merging.
  - the flooding will eventually reach a stage where only the tip of the dams will be visible above the water level: the dams represent our watershed lines (and **the catchment basins** our **final segmentation**).



https://imagej.net/plugins/classic-watershed

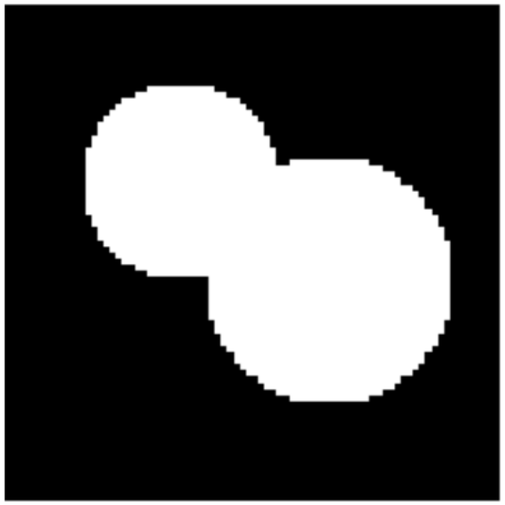
Watershed

- How do we generate a useful topographical surface from a black-and-white mask?
- The intensity at each position (x, y) of the **distance transform** of a binary image is the distance of that pixel from the closest background pixel, if the pixel is a foreground pixel; or 0 if it is a background pixel.
- The distance between any two pixels p_1 and p_2 can be calculated for instance as:

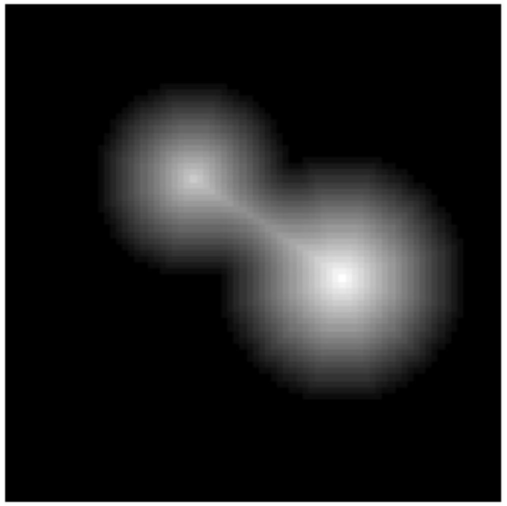
$$D_{p_1,p_2} = (\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2})$$

Euclidean distance

Overlapping objects



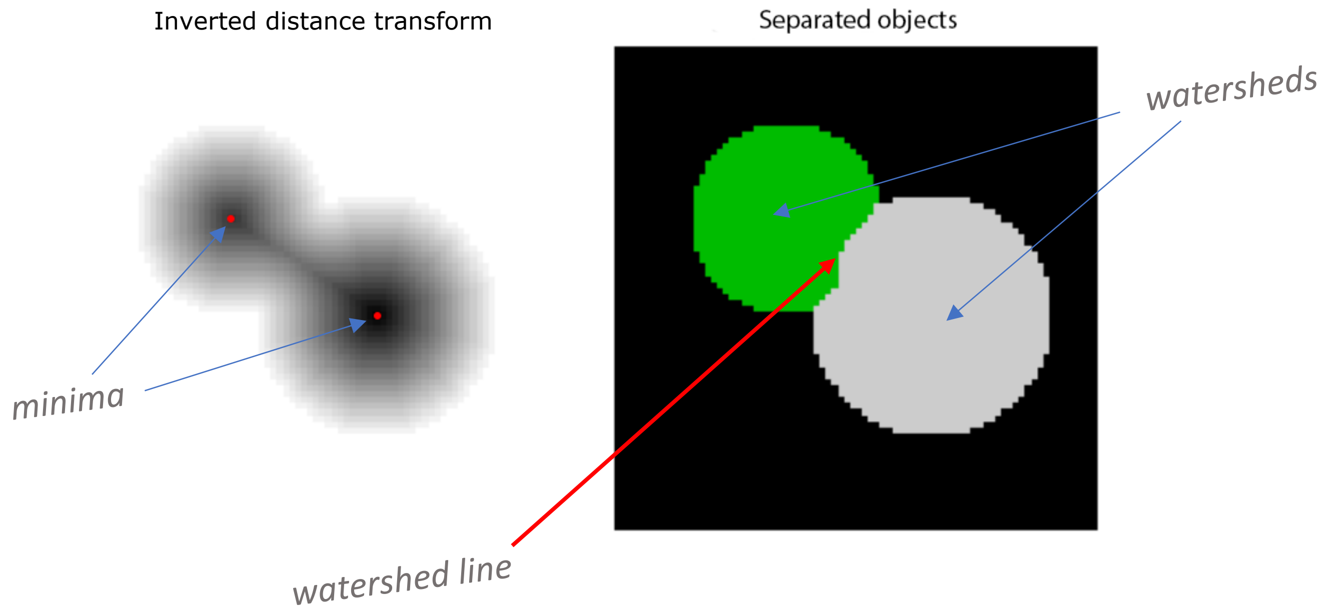
Distance transform



Watershed

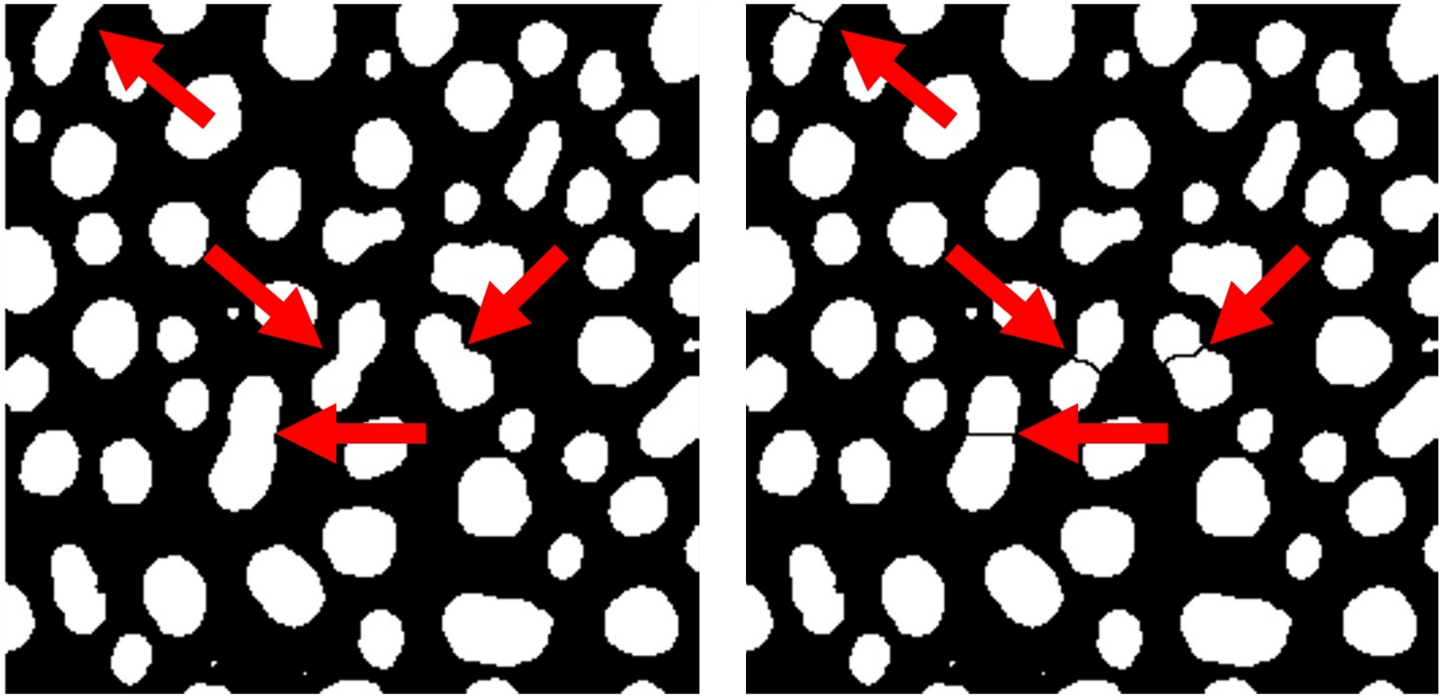
Since the watershed algorithm starts flooding the topology from local minima, we **invert** the distance transform so that the local maxima become local minima.

If we now apply the watershed algorithm, we can easily separate the touching objects.



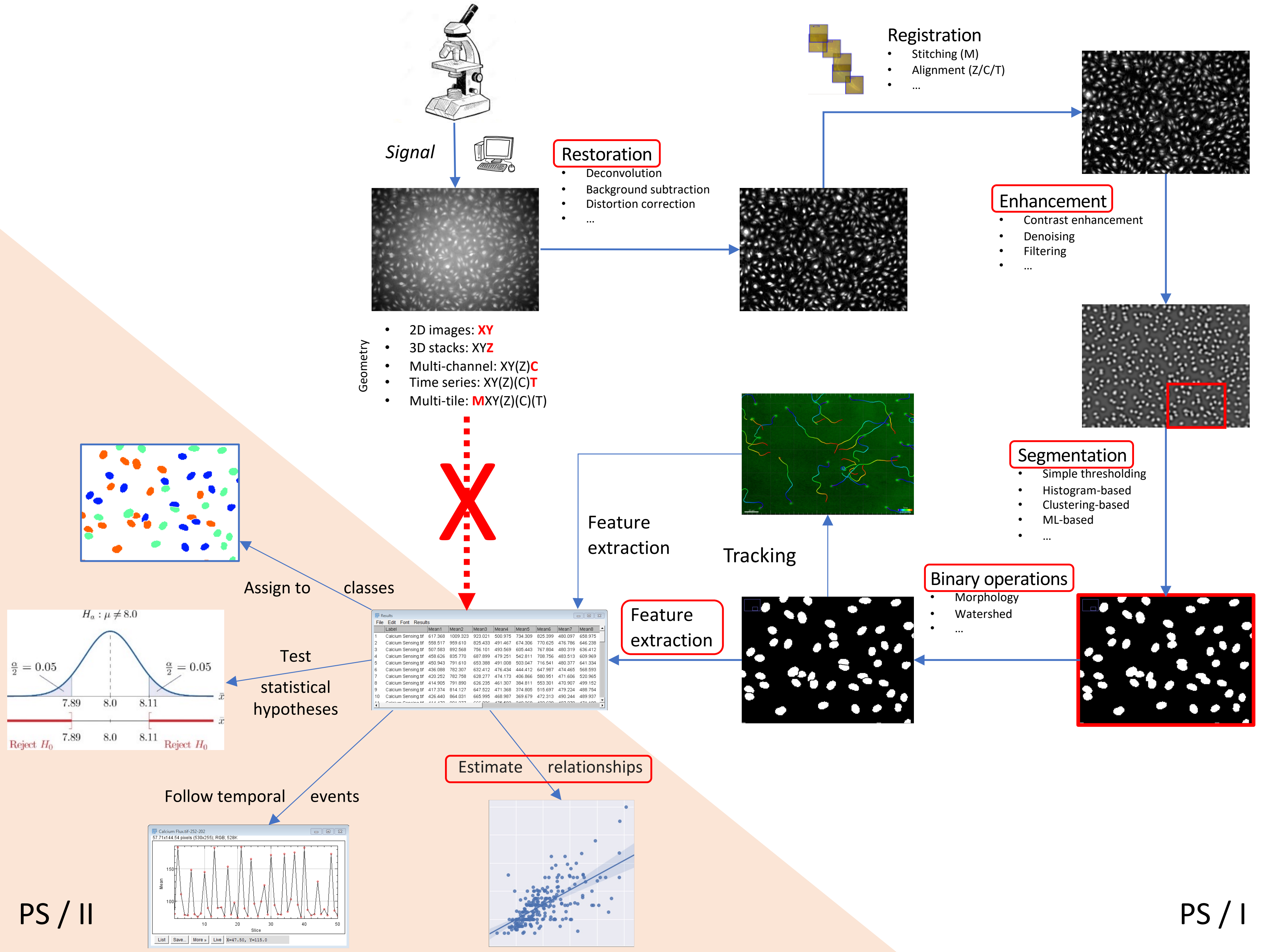
Watershed

Often, objects in an image that are close to each other are difficult to segment and are fused into individual blobs in the binary image.



After watershed, the objects are separated.

now we can measure some features!



Results

File Edit Font Results

	Label	Mean1	Mean2	Mean3	Mean4	Mean5	Mean6	Mean7	Mean8
1	Calcium Sensing.tif	617.368	1009.323	923.021	500.975	734.309	825.399	480.097	658.975
2	Calcium Sensing.tif	558.517	959.610	825.433	491.467	674.306	770.625	476.786	646.238
3	Calcium Sensing.tif	507.583	892.568	756.101	493.569	605.443	767.804	480.319	636.412
4	Calcium Sensing.tif	458.626	835.770	687.899	479.251	542.811	708.756	483.513	609.969
5	Calcium Sensing.tif	450.943	791.610	653.388	491.008	503.047	716.541	480.377	641.334
6	Calcium Sensing.tif	436.088	782.307	632.412	476.434	444.412	647.987	474.465	568.593
7	Calcium Sensing.tif	420.252	782.758	628.277	474.173	406.866	580.951	471.606	520.965
8	Calcium Sensing.tif	414.905	791.890	626.235	461.307	384.811	553.301	470.907	499.152
9	Calcium Sensing.tif	417.374	814.127	647.522	471.368	374.805	515.697	479.224	488.754
10	Calcium Sensing.tif	426.440	864.031	665.995	468.987	369.679	472.313	490.244	489.937

Reading material

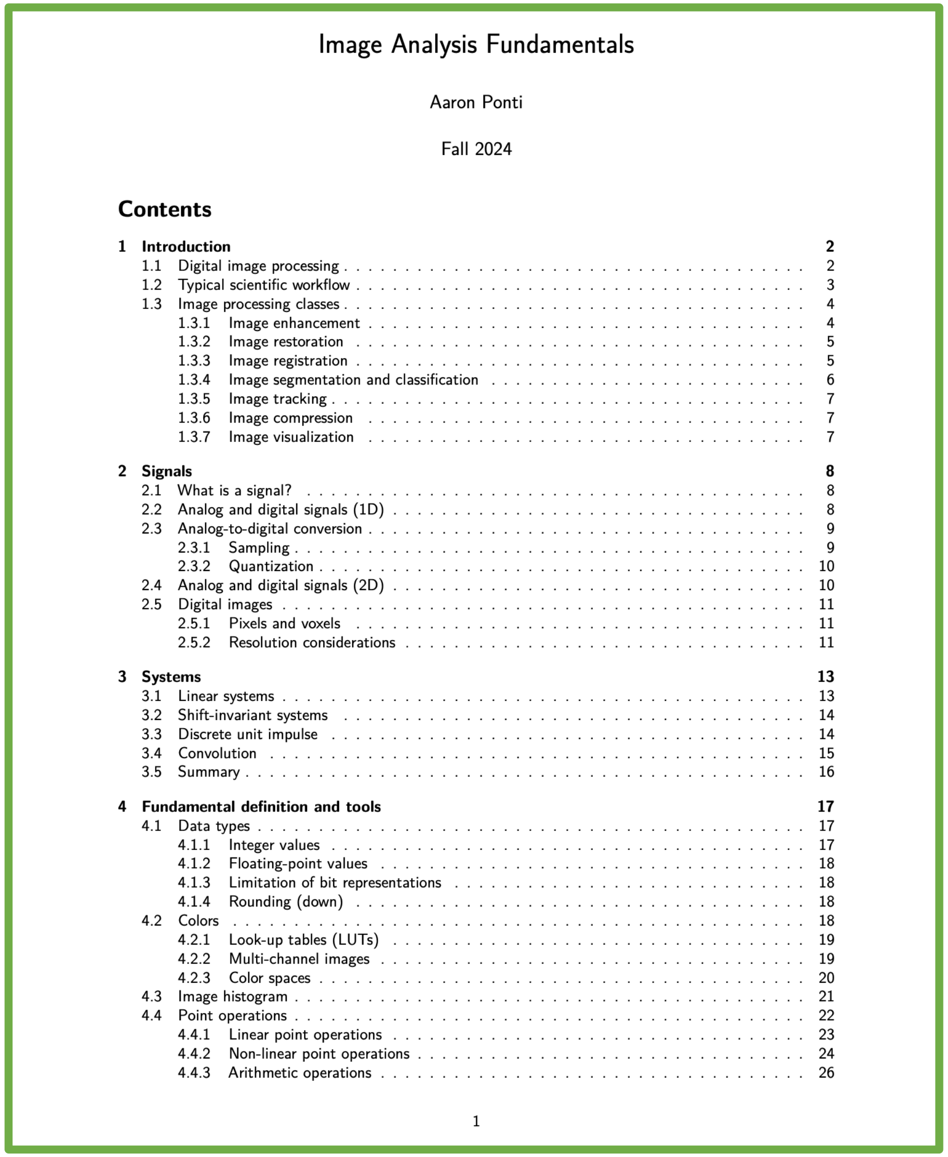


Image Analysis Fundamentals

Aaron Ponti

Fall 2024

Contents

1

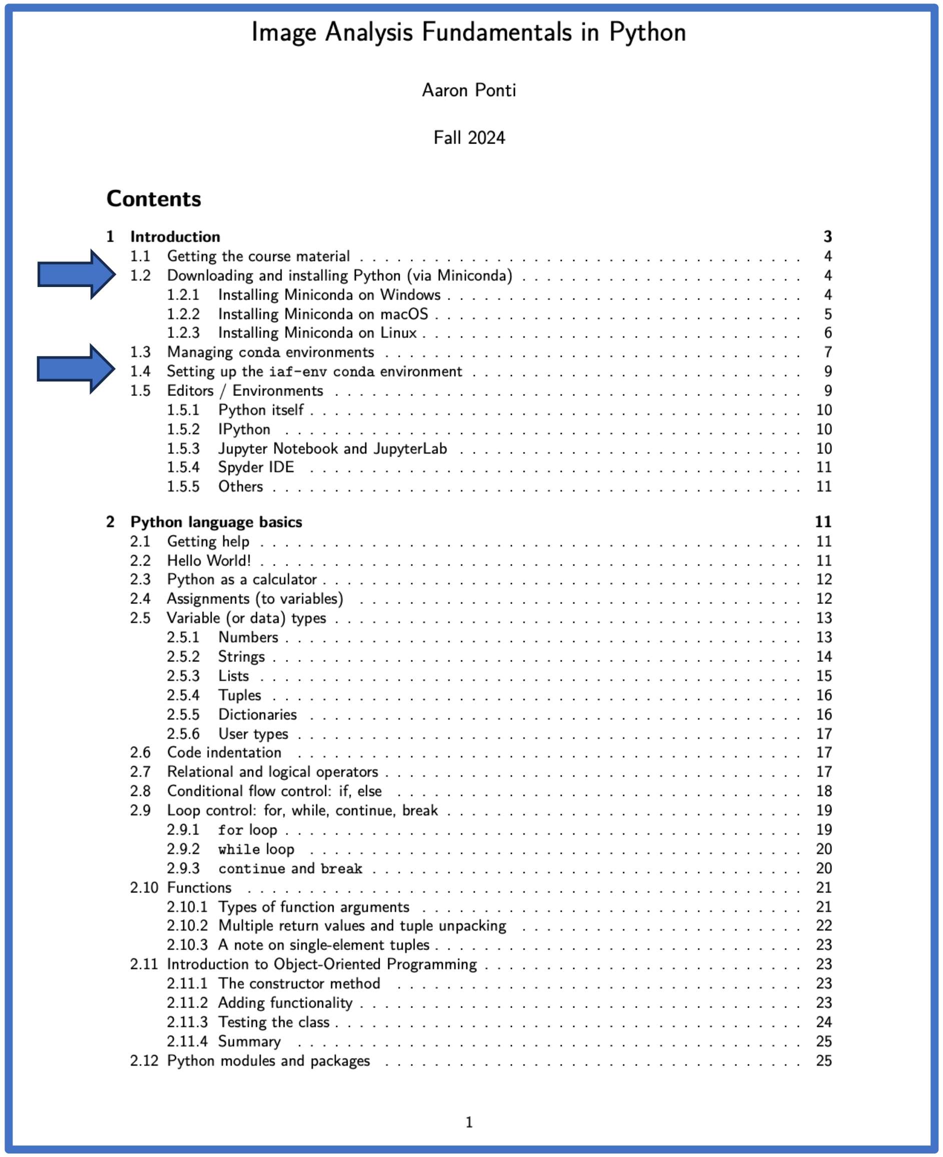


Image Analysis Fundamentals in Python

Aaron Ponti

Fall 2024

Contents

1