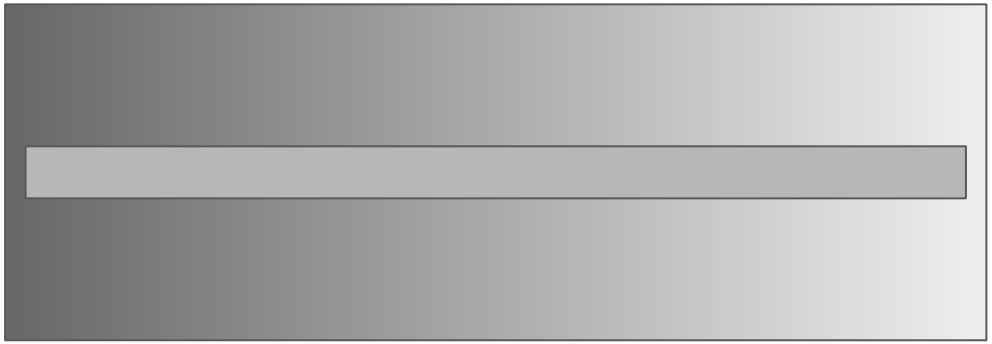
Fundamentals in Digital Image processing

Lectures on Advanced Microscopy

Guillaume Witz, PhD
Science IT Support - MIC

Why do we need image processing?

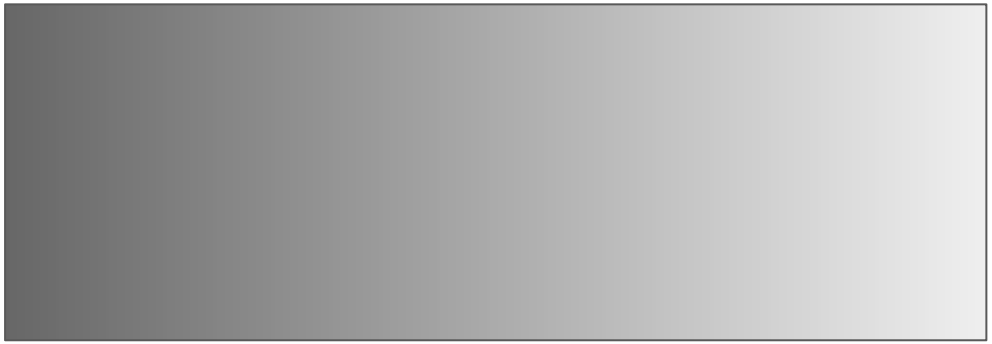


We are easily fooled and tend to see patterns everywhere.

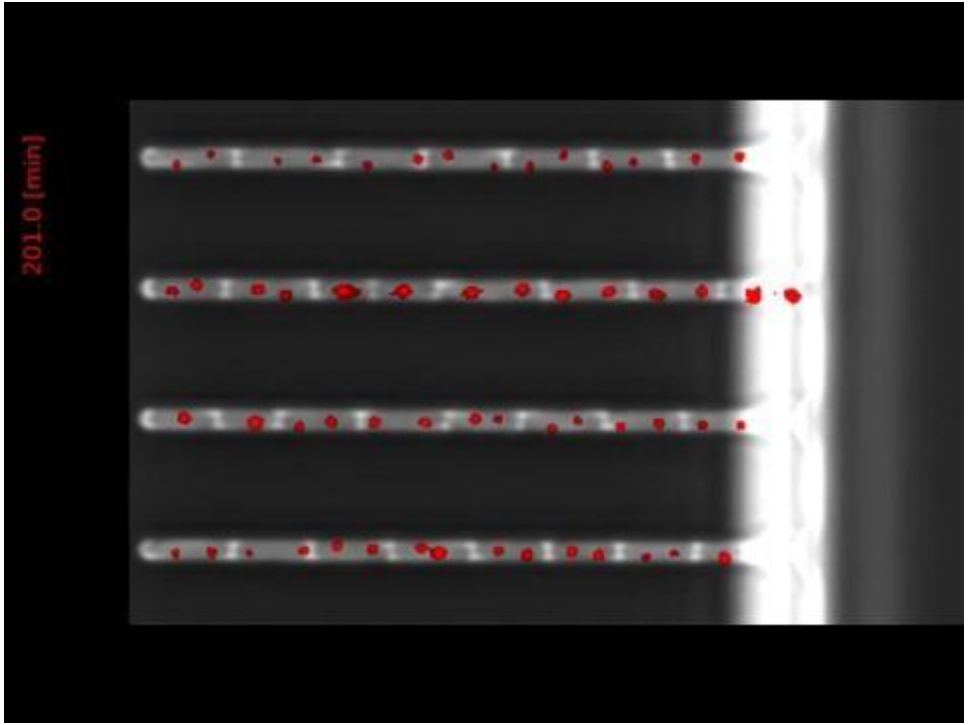
Reproducibility requires a standardized data analysis

https://www.flickr.com/photos/eworm/3401596917

Why do we need image processing?



Why do we need image processing?



E. coli growing in microfluidics channels.
Tagged origin of replication.

10 positions
24 hours
3 min intervals
3 wavelengths
==
14'400 images

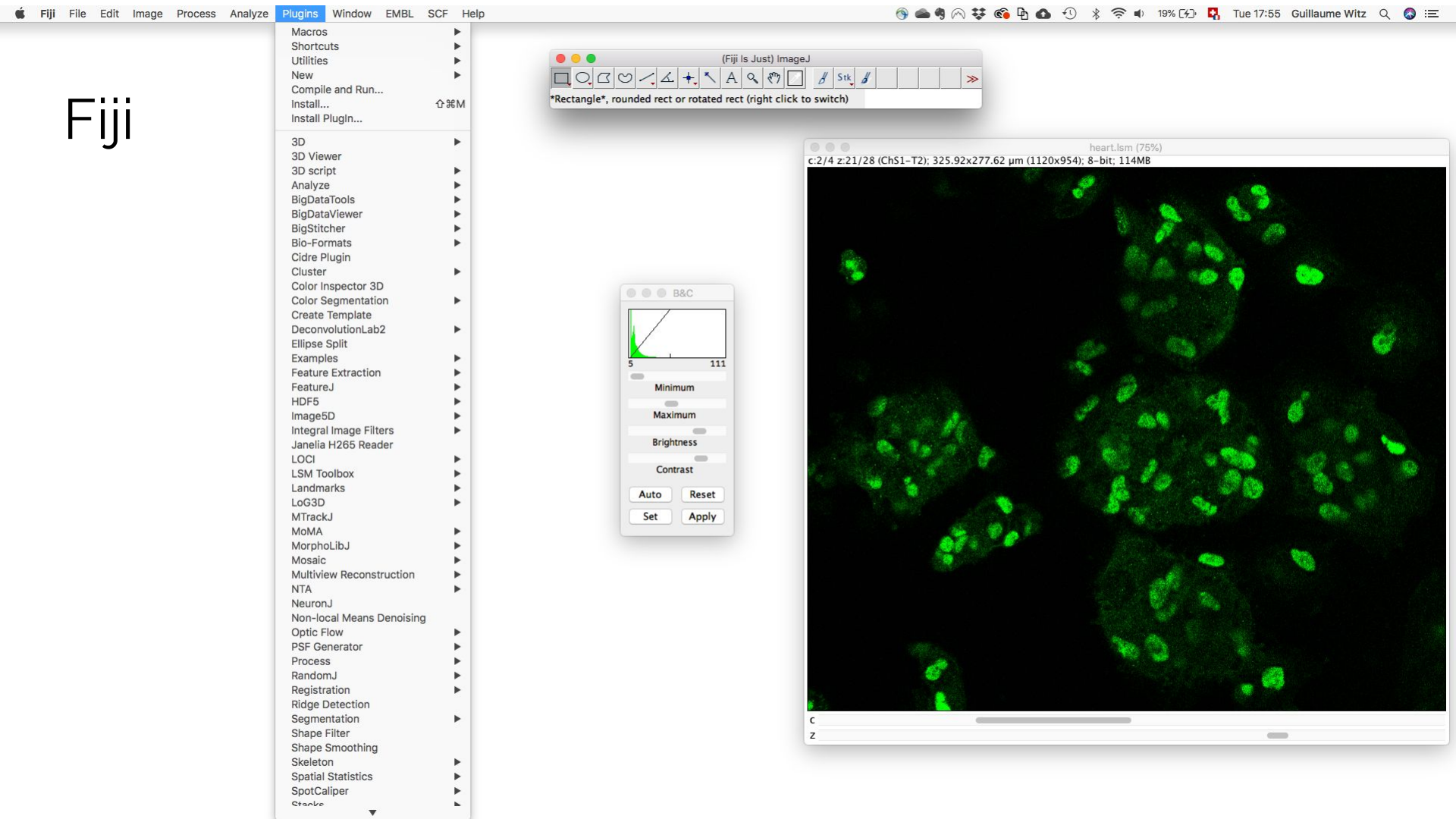
Modern microscopy methods and the trend towards quantitative biology make computational approaches necessary.

Tools

One can do 99% of image processing tasks using open source tools. Two ecosystems are currently dominating:

Fiji (Java)

Fiji


Fiji File Edit Image Process Analyze Plugins Window EMBL SCF Help
Tue 17:55 Guillaume Witz
Macros
Shortcuts
Utilities
New
Compile and Run...
Install... ⇧⌘M
Install PlugIn...
3D
3D Viewer
3D script
Analyze
BigDataTools
BigDataViewer
BigStitcher
Bio-Formats
Cidre Plugin
Cluster
Color Inspector 3D
Color Segmentation
Create Template
DeconvolutionLab2
Ellipse Split
Examples
Feature Extraction
FeatureJ
HDF5
Image5D
Integral Image Filters
Janelia H265 Reader
LOCI
LSM Toolbox
Landmarks
LoG3D
MTrackJ
MoMA
MorphoLibJ
Mosaic
Multiview Reconstruction
NTA
NeuronJ
Non-local Means Denoising
Optic Flow
PSF Generator
Process
RandomJ
Registration
Ridge Detection
Segmentation
Shape Filter
Shape Smoothing
Skeleton
Spatial Statistics
SpotCaliper
(Fiji Is Just) ImageJ
Rectangle, rounded rect or rotated rect (right click to switch)
B&C
5
111
Minimum
Maximum
Brightness
Contrast
Auto
Reset
Set
Apply
heart.lsm (75%)
c:2/4 z:21/28 (ChS1-T2); 325.92x277.62 µm (1120x954); 8-bit; 114MB
c
z

Python/Jupyter

jupyter_example - Jupyter Not

localhost:8888/notebooks/GoogleDrive/BernMIC/Trainings/Fundamentals_image_processing/Notebooks/jupyter_example.ipynb

jupyter jupyter_example Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code Voila

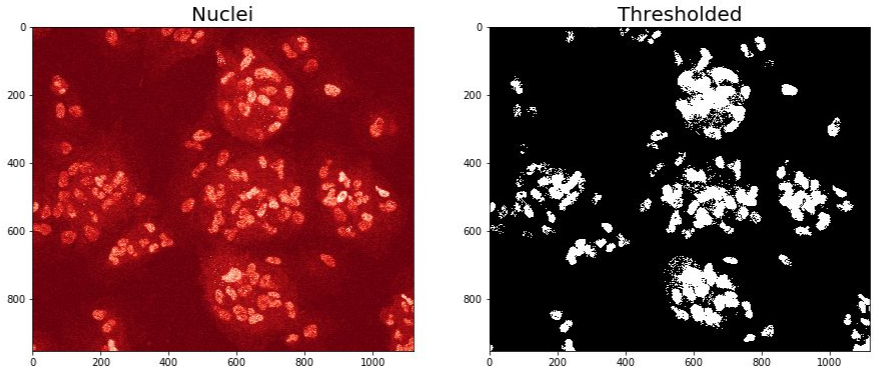
Thresholding

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import skimage
```

In [9]:
```
image = skimage.io.imread('Samples/heart.lsm', plugin='tifffile')
```

In [18]:
```
projection = np.mean(image[0,:,:,:,1],axis = 0)
thresholded = projection > skimage.filters.threshold_otsu(projection)
```

In [35]:
```
fig, ax = plt.subplots(1,2, figsize = (15,7))
ax[0].imshow(-np.max(image[0,:,:,:,1],axis = 0),cmap = 'Reds');
ax[0].set_title('Nuclei',fontdict = {'fontsize':20})
ax[1].imshow(thresholded,cmap = 'gray');
ax[1].set_title('Thresholded',fontdict = {'fontsize':20});
```



In [ ]:

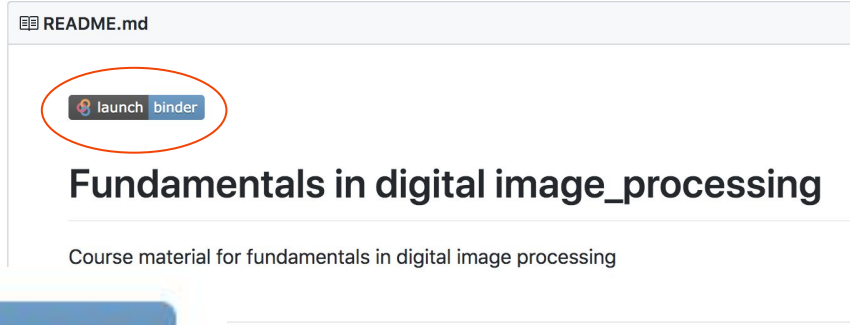
Interactive examples

You will find interactive examples illustrating some of the concepts presented in this lecture here :

https://fl-7-45.zhdk.cloud.switch.ch

If the link on the previous page does not work 1/2

You can play with little interactive web-apps by clicking following this link: https://github.com/guiwitz/Fundamentals_image_processing



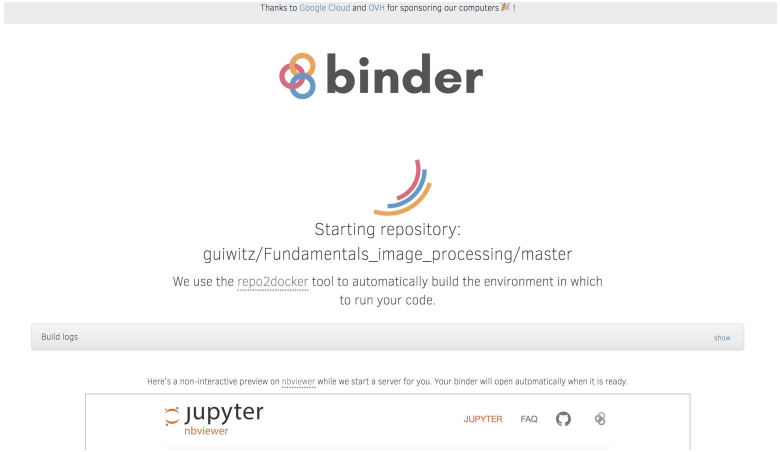
And pressing this button: launch binder

Or by following this link: https://mybinder.org/v2/gh/guiwitz/Fundamentals_image_processing/master?urlpath=voila

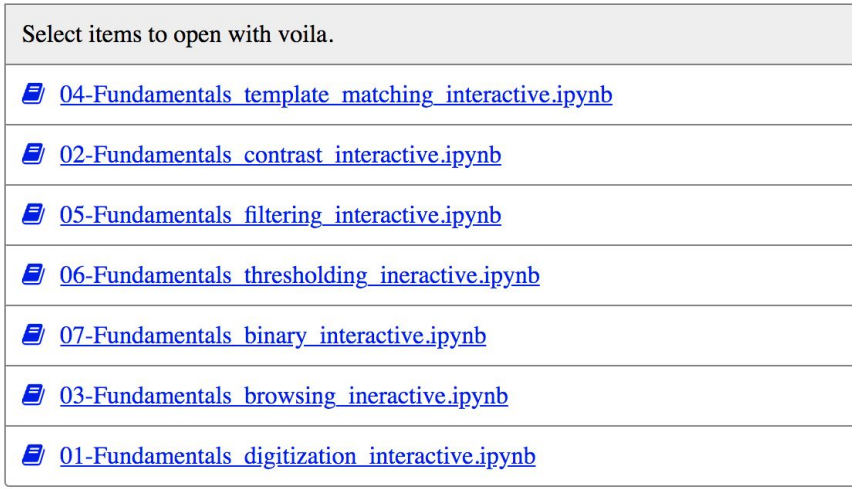
For those who are curious, the code for demonstration is available on GitHub (looking at the code IS **NOT A REQUIREMENT OF THE COURSE**):

If the link on the previous page does not work 2/2

This should appear:



Just wait until this appears:



To run the web-app, just click on the appropriate link. You can come back to this view using the ← button of the browser.

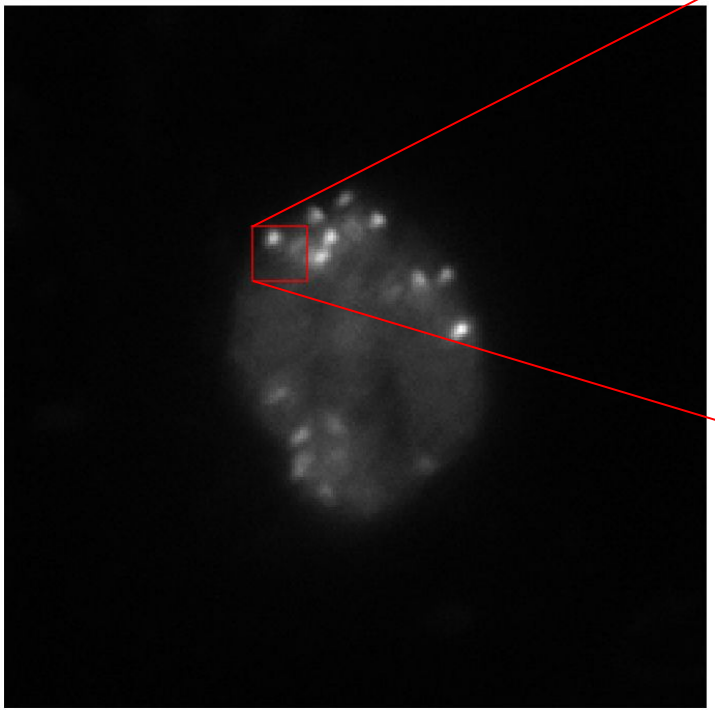
If things stop working, just click again on the binder button:

launch binder

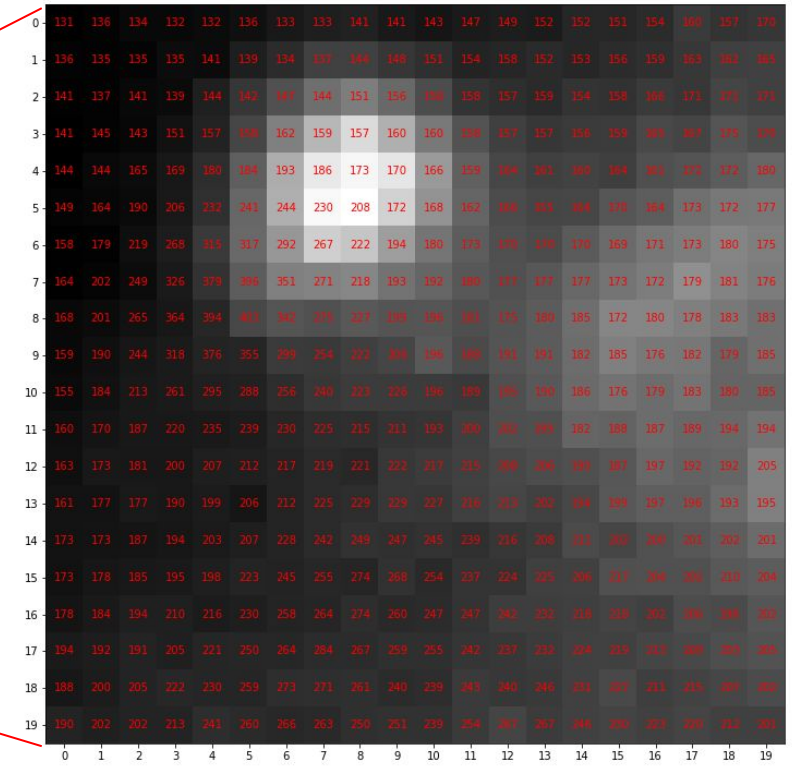
Table of contents

I.
What is an image ?

What is an image?



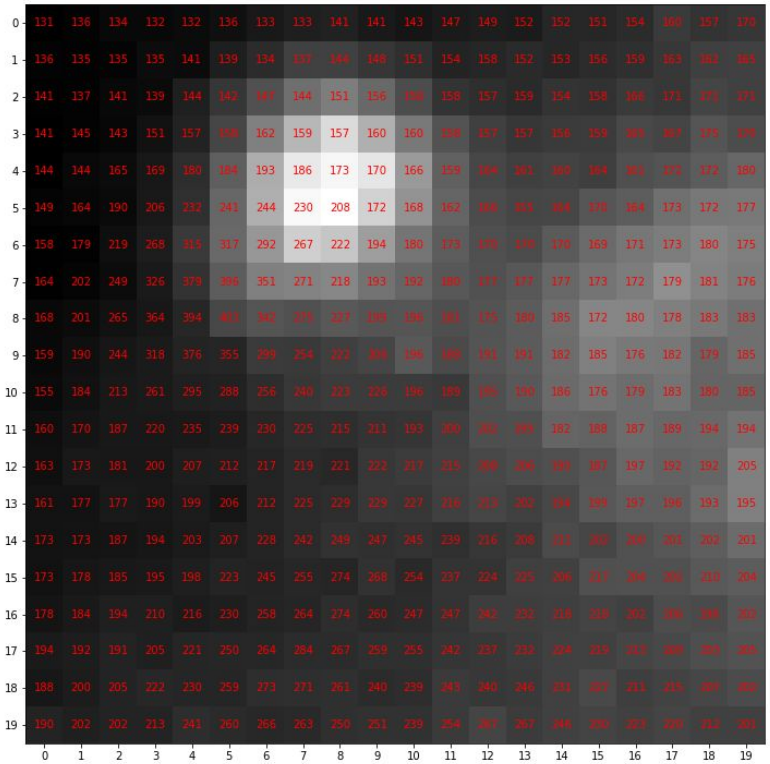
Human fibroblast infected by toxoplasma gondii (here GFP-tagged)
http://cellimagelibrary.org/images/10494 doi:10.7295/W9CIL10494



An image is simply a collection of pixels arranged as a **2 matrix**

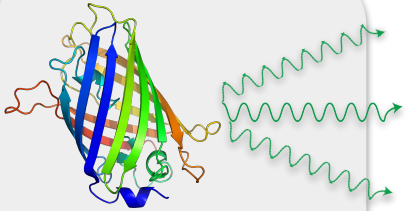
In raw image data, the pixel value has a **physical meaning** (e.g. number of photons emitted by a fluorescent protein).

Keeping raw data raw

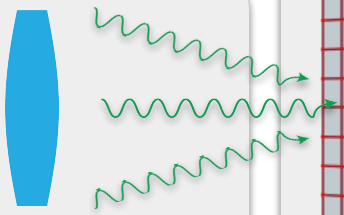


Any time you change the pixel values e.g. by drawing or by saving in another format e.g. jpg the physical information is ***LOST***

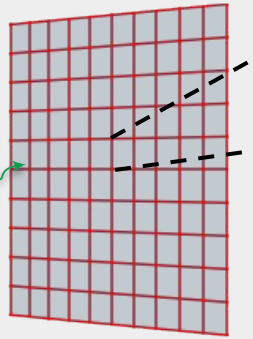
From physics to numbers



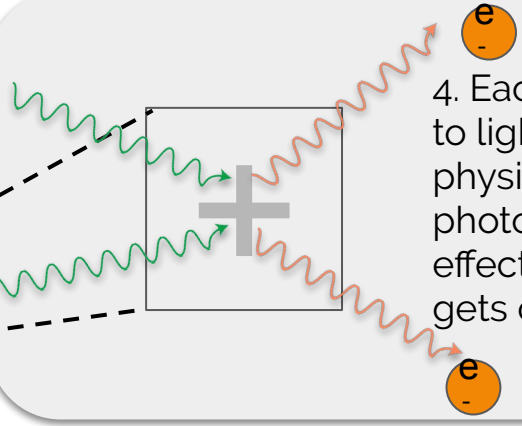
1. Fluorescent protein emits light



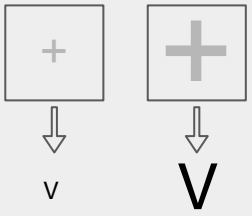
2. Emitted light goes through microscope optics



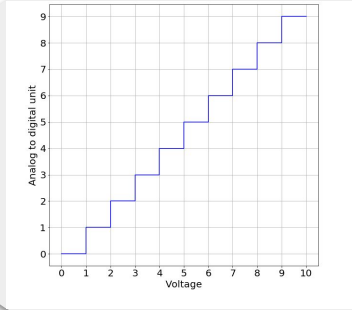
3. Light lands on camera detector composed of pixels



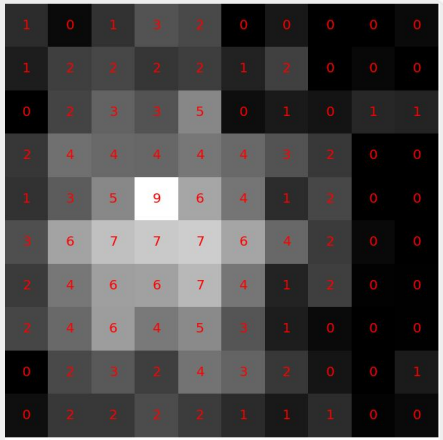
4. Each pixel reacts to light through a physical effect (e.g. photoelectric effect where pixel gets charged)



5. Pixel charge translated into voltage



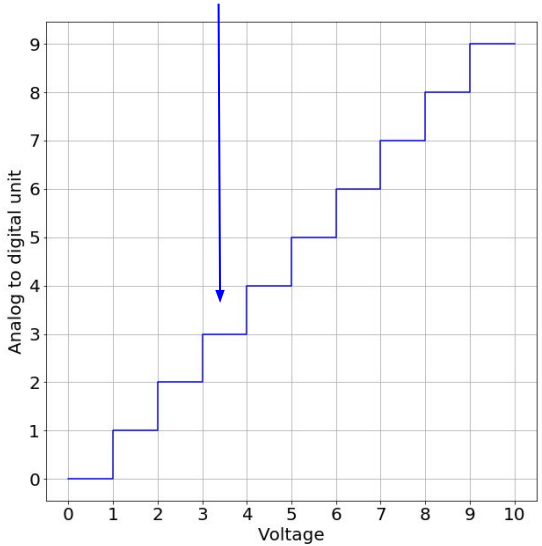
6. Voltage of each pixel is digitized, i.e. assigned a number depending on the voltage range to which it belongs.



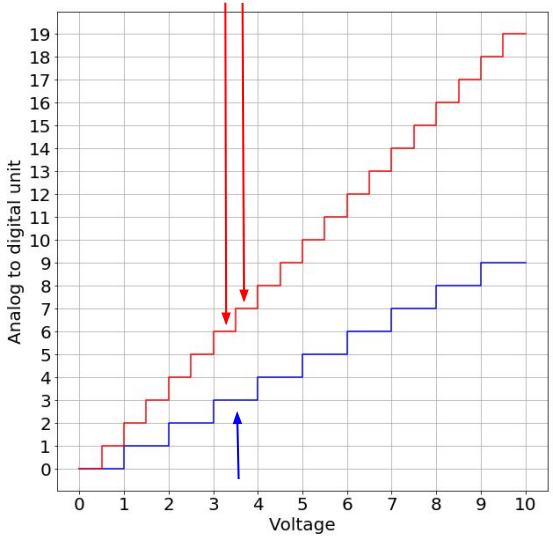
7. **For plotting purposes**, a color shade is assigned to each digitization level

Digitization

Voltage from 3V to 4V is assigned value 3



Voltage from 3V to 4V is split into two values: more “detail” visible



By convention, images are most of the time encoded into 8, 16 or 32 bits, i.e

8 bits : 256 values
16 bits : 65536 values
32 bits: ~4 * 10^9 values

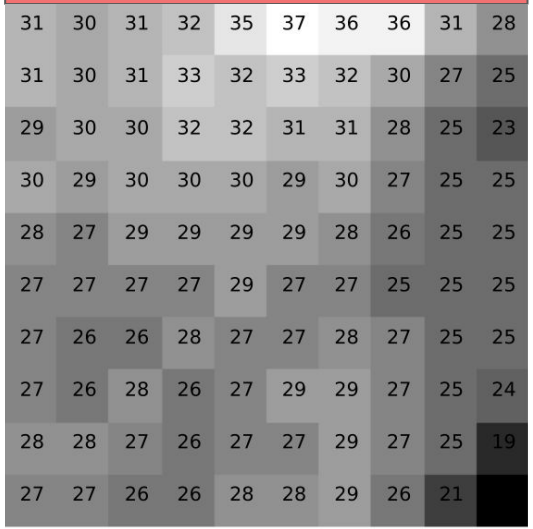
[You can play with the ***digitization*** web-app]

II.
How is an image stored ?

Metadata: reconstructing the physical meaning

Metadata:

- What microscope, what camera etc
- Exposure time
- Position of acquisition (e.g. 96 well plate)



Information about the image is stored in a "**header**"

Pixel information is stored in binary form in the file body

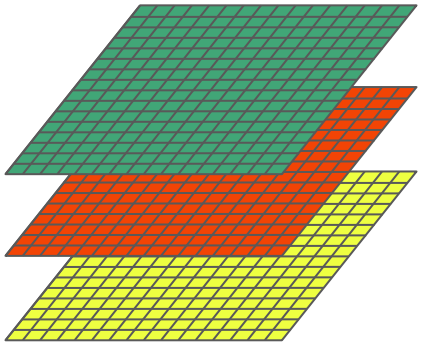
Metadata: reconstructing the physical meaning

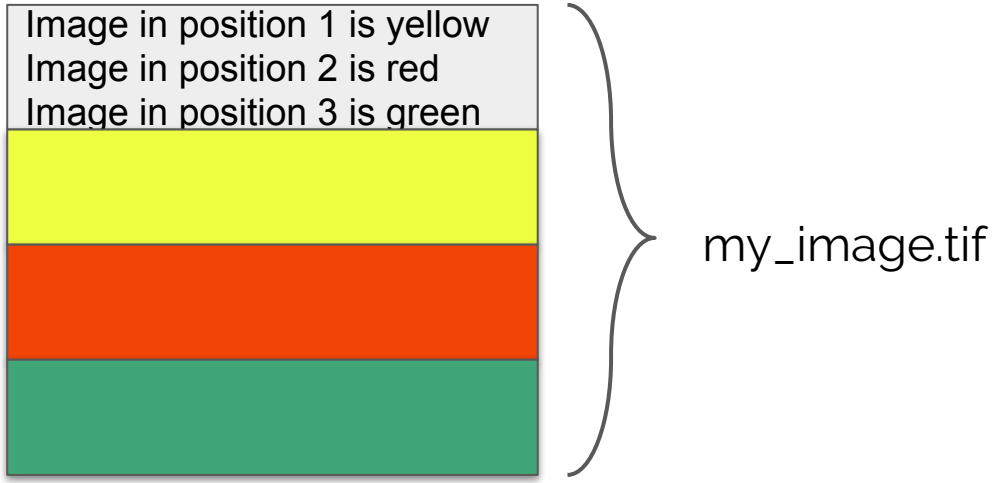
OME Metadata - 190322_MacrophagesSlaug_DAPI__Phall633_Iba1555_GFPLm_2h_60x_0005.oir

```
<OME xmlns="http://www.openmicroscopy.org/Schemas/OME/2016-06" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.openmicroscopy.org/Schemas/OME/2016-06 http://www
  <Instrument ID="Instrument:0">
    <Laser ID="LightSource:0:0" Model="LD405_65792" Wavelength="405.0" WavelengthUnit="nm"/>
    <Laser ID="LightSource:0:1" Model="LD445_65793"/>
    <Laser ID="LightSource:0:2" Model="LD488_65794" Wavelength="488.0" WavelengthUnit="nm"/>
    <Laser ID="LightSource:0:3" Model="LD514_65795"/>
    <Laser ID="LightSource:0:4" Model="LD561_65796" Wavelength="561.0" WavelengthUnit="nm"/>
    <Laser ID="LightSource:0:5" Model="LD594_65797"/>
    <Laser ID="LightSource:0:6" Model="LD640_65798" Wavelength="640.0" WavelengthUnit="nm"/>
    <Detector Gain="1.0" ID="Detector:0:0" Offset="0.0" Voltage="419.0" VoltageUnit="V"/>
    <Detector Gain="1.0" ID="Detector:0:1" Offset="0.0" Voltage="350.0" VoltageUnit="V"/>
    <Detector Gain="1.0" ID="Detector:0:2" Offset="2.0" Voltage="350.0" VoltageUnit="V"/>
    <Detector Gain="1.0" ID="Detector:0:3" Offset="1.0" Voltage="397.0" VoltageUnit="V"/>
    <Detector Gain="1.0" ID="Detector:0:4" Offset="0.0" Voltage="408.0" VoltageUnit="V"/>
    <Objective ID="Objective:0:0" Immersion="Oil" LensNA="1.3" Model="UPLSAPO 60" NominalMagnification="60.0" WorkingDistance="0.3" WorkingDistanceUnit="mm"/>
  <Image ID="Image:0" Name="190322_MacrophagesSlaug_DAPI__Phall633_Iba1555_GFPLm_2h_60x_0005.oir">
    <AcquisitionDate>2019-05-29T12:41:34.450</AcquisitionDate>
    <InstrumentRef ID="Instrument:0"/>
    <ObjectiveSettings ID="Objective:0:0" RefractiveIndex="1.406"/>
    <Pixels BigEndian="false" DimensionOrder="XYCZT" ID="Pixels:0" Interleaved="false" PhysicalSizeX="0.096775563118596" PhysicalSizeXUnit="µm" PhysicalSizeY="0.096775563118596" PhysicalSizeYUnit="µm"
      <Channel Color="16777215" EmissionWavelength="481.0" EmissionWavelengthUnit="nm" ExcitationWavelength="441.0" ExcitationWavelengthUnit="nm" ID="Channel:0:0" Name="CH1" PinholeSize="238.0"
        <LightSourceSettings ID="LightSource:0:0"/>
        <DetectorSettings ID="Detector:0:1"/>
        <LightPath/>
      <Channel Color="16711935" EmissionWavelength="540.0" EmissionWavelengthUnit="nm" ExcitationWavelength="500.0" ExcitationWavelengthUnit="nm" ID="Channel:0:1" Name="CH2" PinholeSize="238.0"
        <LightSourceSettings ID="LightSource:0:2"/>
        <DetectorSettings ID="Detector:0:2"/>
        <LightPath/>
      <Channel Color="-8453889" EmissionWavelength="614.0" EmissionWavelengthUnit="nm" ExcitationWavelength="564.0" ExcitationWavelengthUnit="nm" ID="Channel:0:2" Name="CH3" PinholeSize="238.0"
        <LightSourceSettings ID="LightSource:0:4"/>
        <DetectorSettings ID="Detector:0:3"/>
        <LightPath/>
      <Channel Color="-16776961" EmissionWavelength="744.0" EmissionWavelengthUnit="nm" ExcitationWavelength="644.0" ExcitationWavelengthUnit="nm" ID="Channel:0:3" Name="CH4" PinholeSize="238.0"
        <LightSourceSettings ID="LightSource:0:6"/>
        <DetectorSettings ID="Detector:0:4"/>
        <LightPath/>
      <MetadataOnly/>
```

More than one image in a file

Often images of the same “object” are acquired in **multiple dimensions:** different **modes** (wave length), **positions** (x, y and depth z) or over **time**. Such ensembles of pictures can be saved within a single file. The header can be used to specify “what is where”.

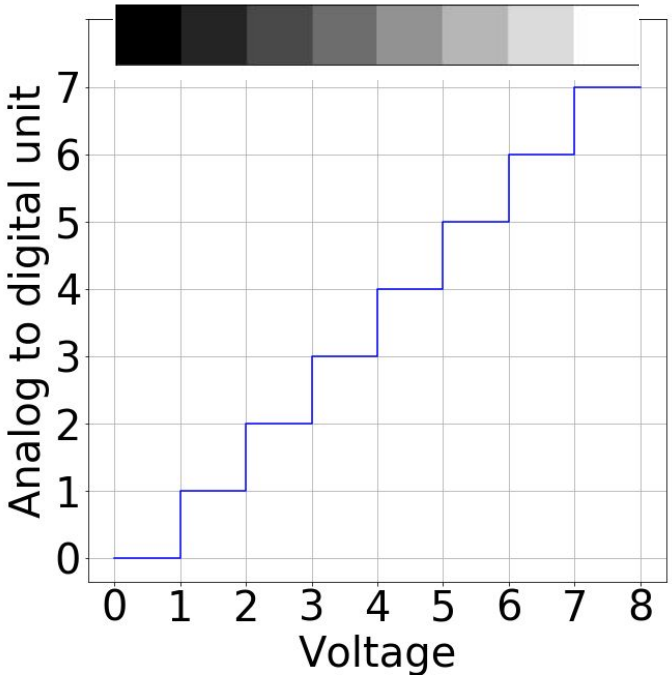




[You can play with the ***browsing*** web-app]

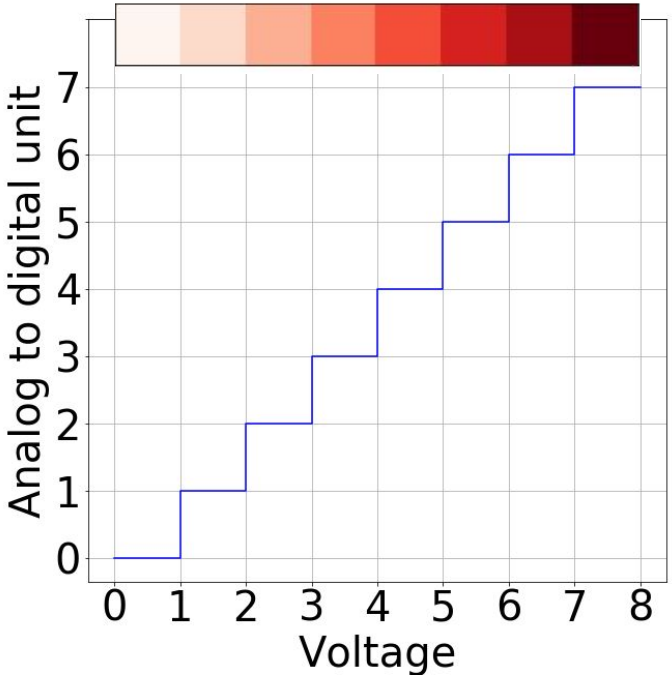
III
Representing an image: Non-destructive operations

Color maps

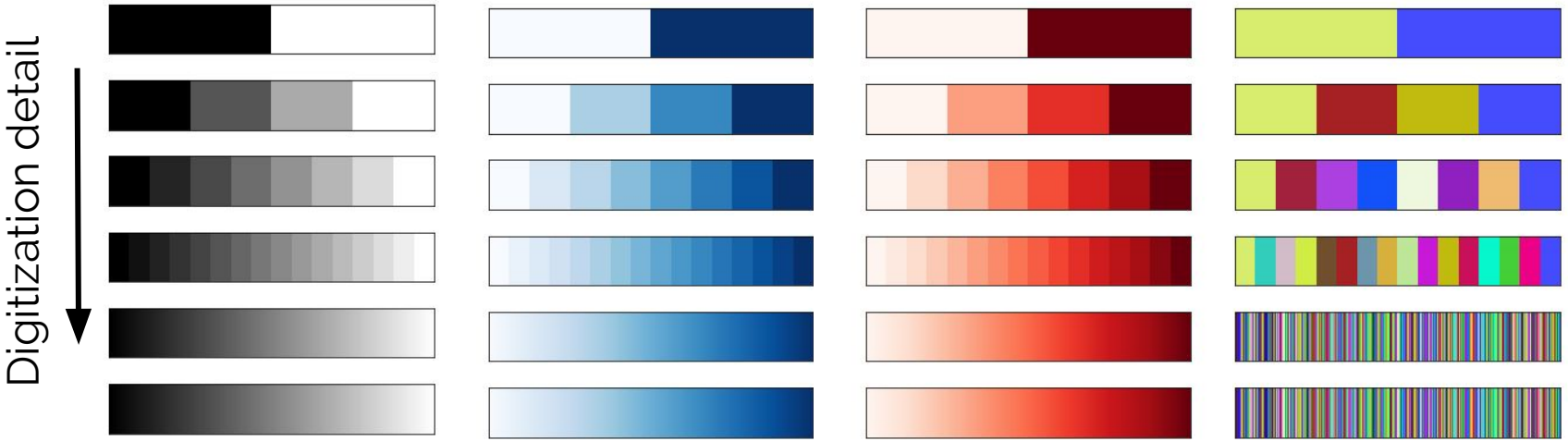


The color we see on a screen is only a mapping of ADU to a given color map.

The choice of color map does not have a physical meaning.

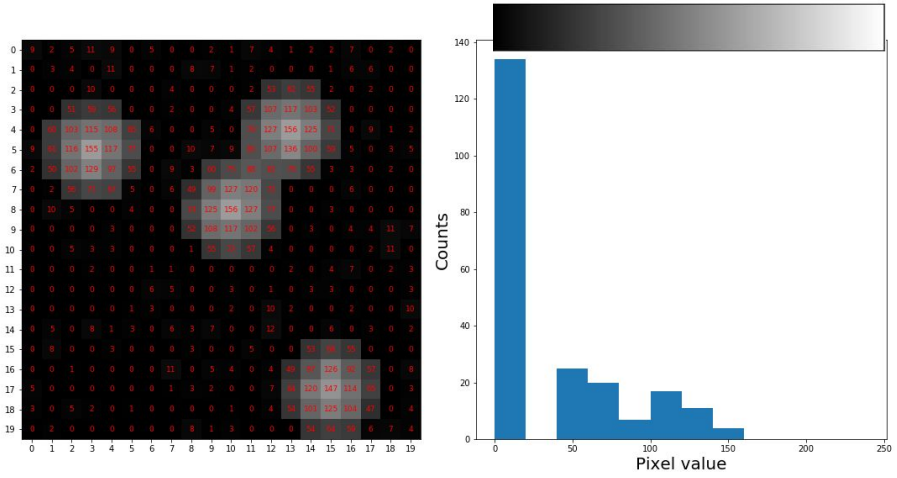


Color maps

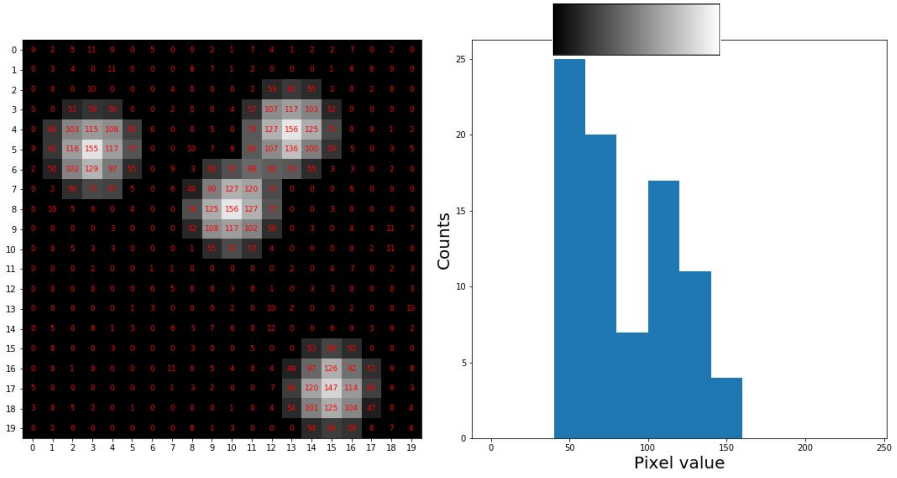


Contrast

If a range of values is not actually used in the image, the associated color shades are “lost”. One can then use a single color shade for all those values, and use the other shades to see better see intensity changes in the image, i.e. increase the contrast.



Low contrast

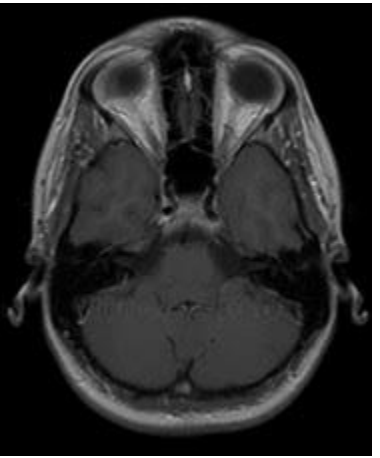


High contrast

[You can play with the ***contrast*** web-app]

Volumetric representation

3D data can also be represented in 3D. Different strategies exist to show the depth e.g. projection of the brightest pixel along the viewing axis, surface representation etc.



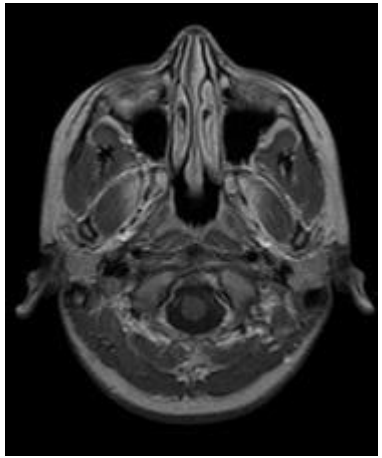
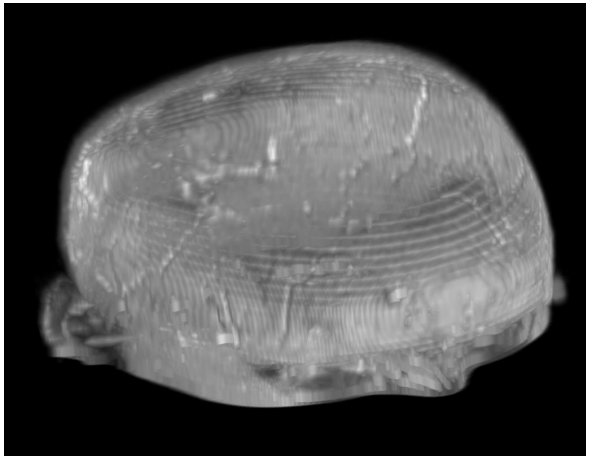
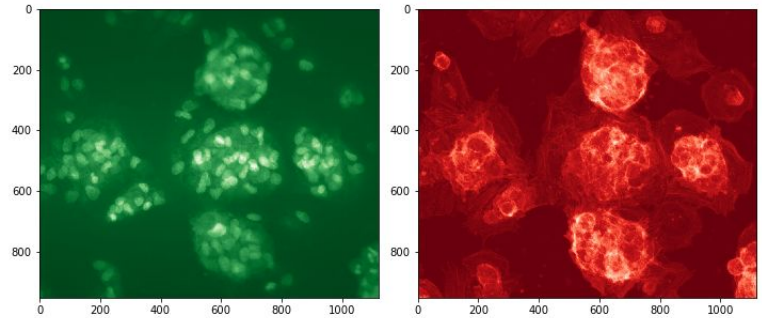


Image stack turned into transparent volume

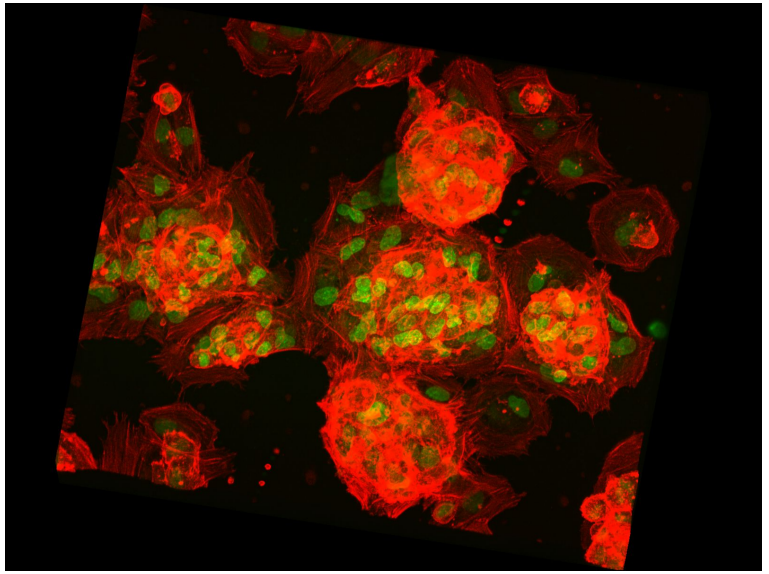


Volumetric representation

3D data can also be represented in 3D. Different strategies exist to show the depth e.g. projection of the brightest pixel along the viewing axis, surface representation etc.



Blending two channels in 3D



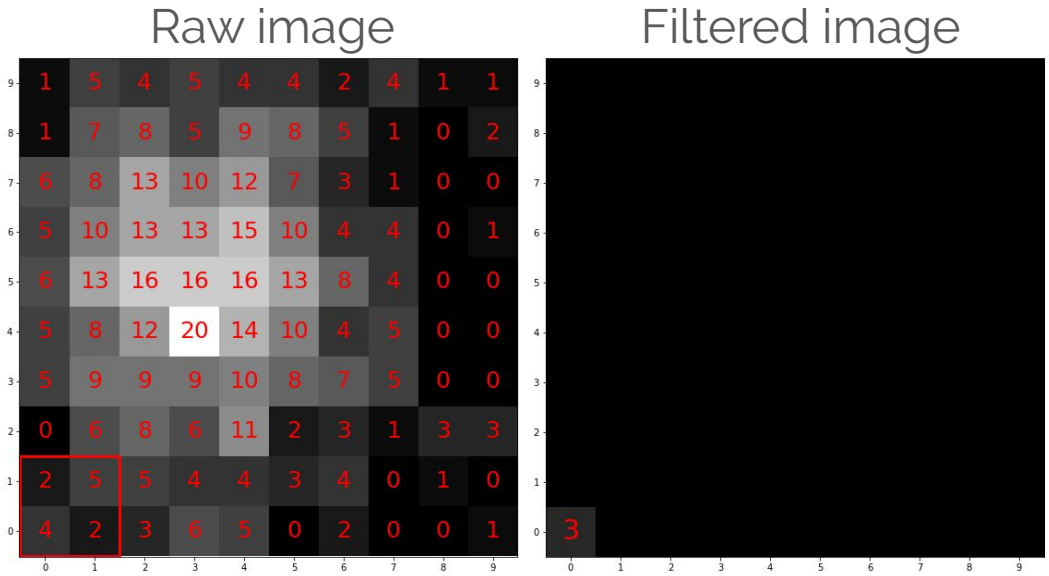
Automated Segmentation of Fluorescence Microscopy Images for 3D Cell Detection in human-derived Cardiospheres, Salvi et al. Scientific Rep. 2019

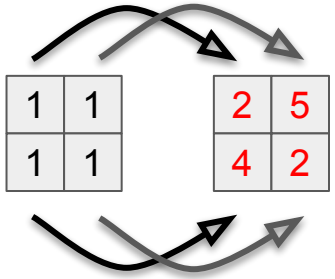
IV.
Resampling, filtering etc.
Destructive operations

Linear Filtering

A small filter window travels across the image. At each position, each pixel of the travel window is multiplied with the corresponding pixel of the image window. The average (or sum) of these multiplications is the pixel value of the filtered image.

This operation is also called a **convolution** (can be done in Fourier space for large images).





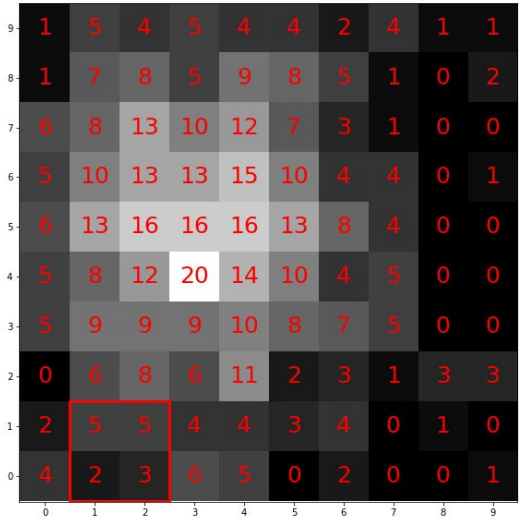
(2*1+5*1+4*1+5*1)/4 = 13/4 = 3.25

round(3.25) = 3

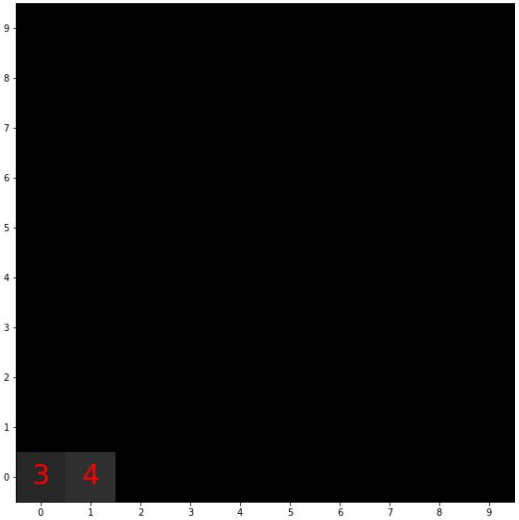
Linear Filtering

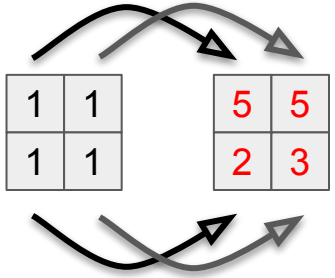
A small filter window travels across the image. At each position, each pixel of the travel window is multiplied with the corresponding pixel of the image window. The average (or sum) of these multiplications is the pixel value of the filtered image.

Raw image



Filtered image





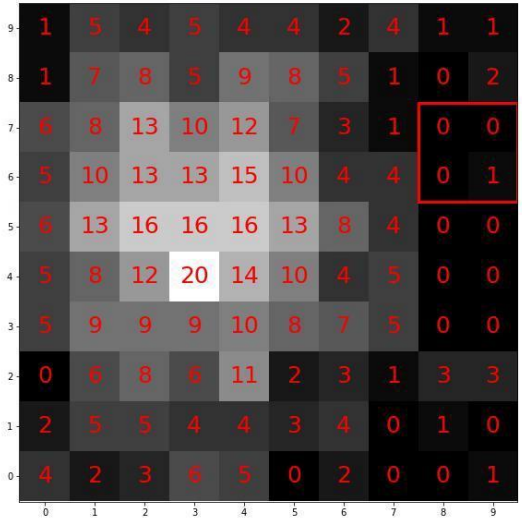
(5*1+5*1+2*1+3*1)/4 = 15/4 = 3.75

round(3.75) = 4

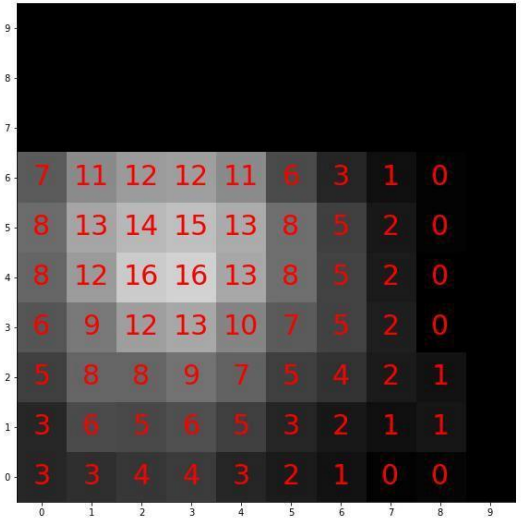
Linear Filtering

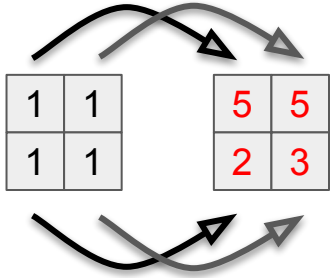
A small filter window travels across the image. At each position, each pixel of the travel window is multiplied with the corresponding pixel of the image window. The average (or sum) of these multiplications is the pixel value of the filtered image.

Raw image



Filtered image



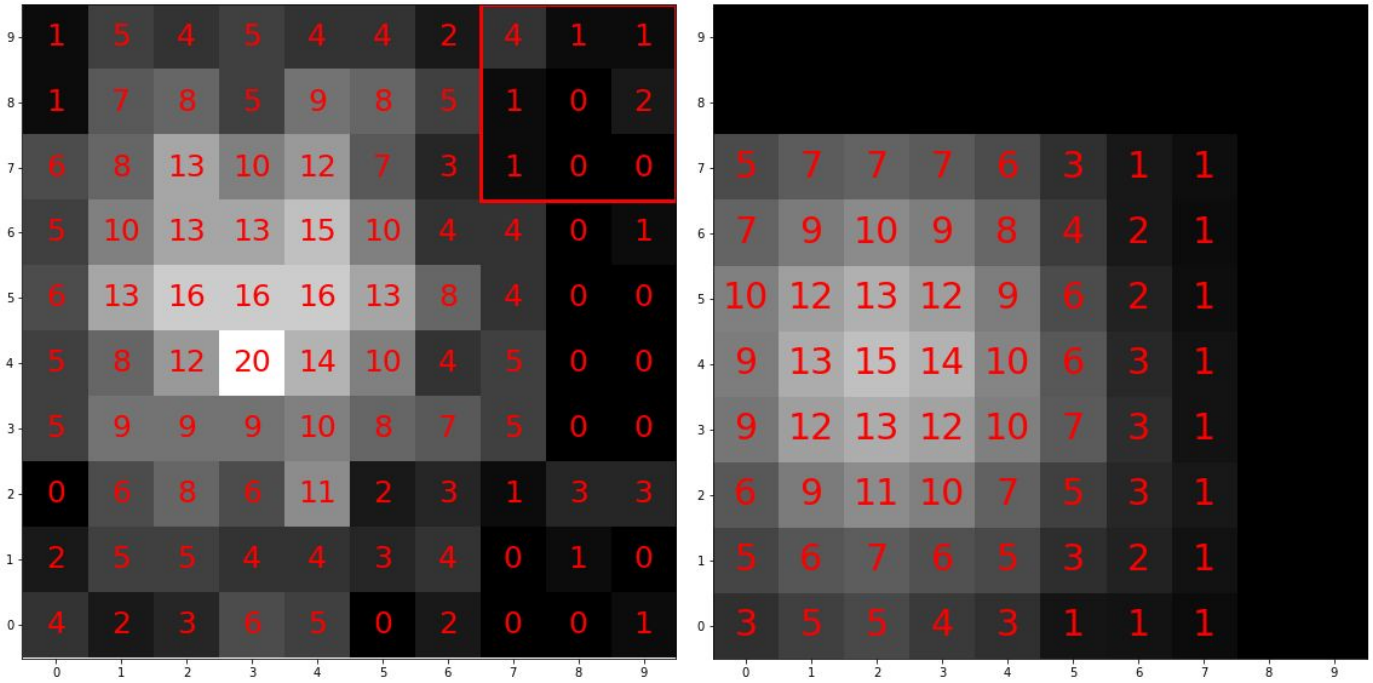


(5*1+5*1+2*1+3*1)/4 = 15/4 = 3.75

round(3.75) = 4

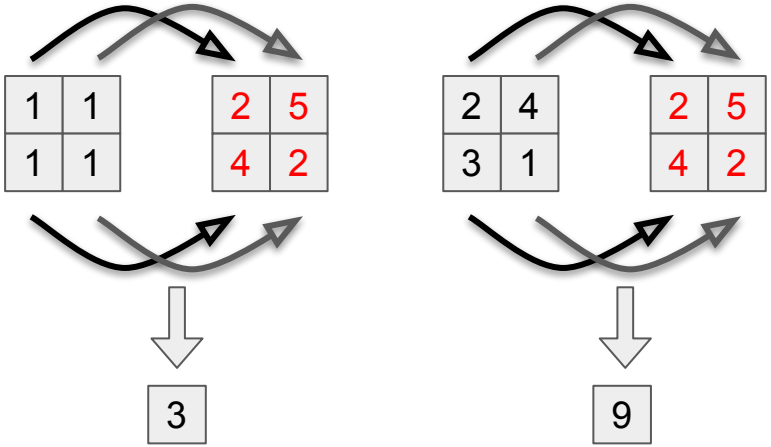
Linear Filtering

The filter can have **any size.** Here it covers a 3x3 region

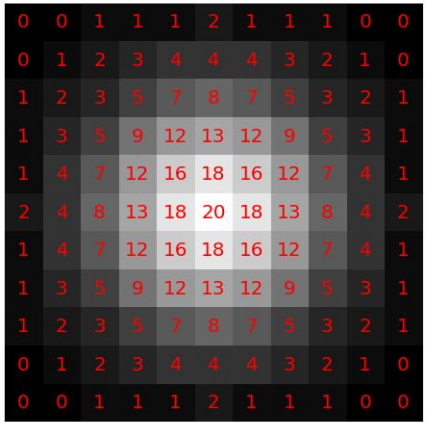


Linear filtering: the filter can be any image

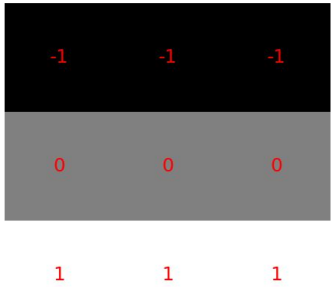
The filter is generally not filled with 1's



Filtering gives maximal response when filter and image region are identical. Therefore convolution can be used to detect specific features.



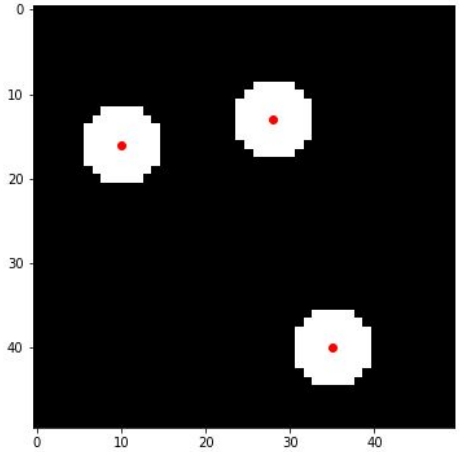
Gaussian filter (spot detection)



Prewitt filter (ridge detection)

Linear filtering for detection

If the shape of the object to be detected is known, one can use a special type of filtering (template matching) to detect them: high response (white) in the filtered image corresponds to objects.



Clean image

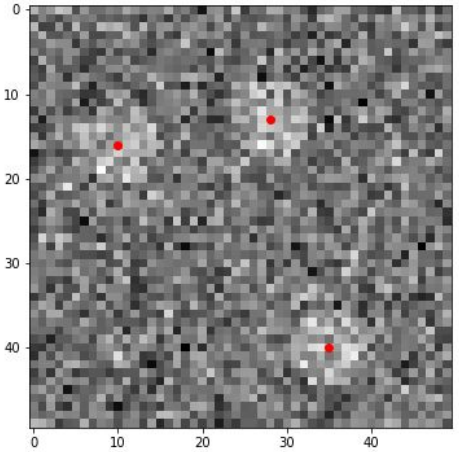
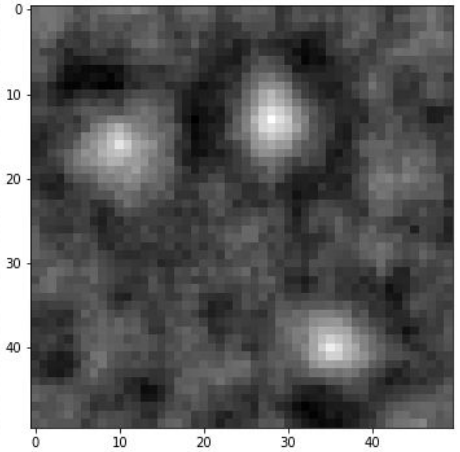
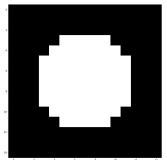


Image corrupted by noise



Filtered image

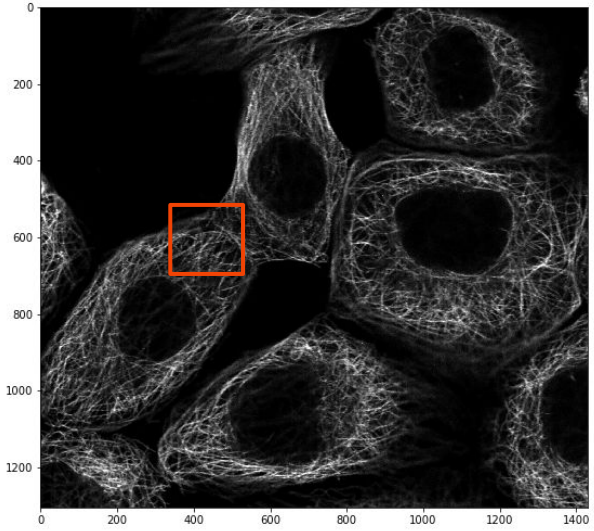


Template

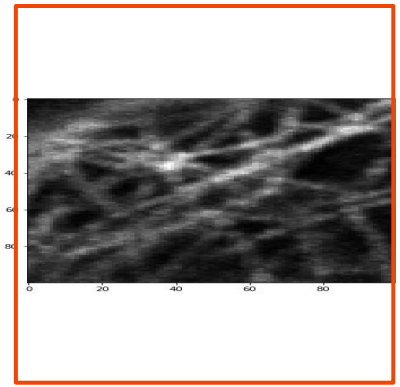
[You can play with the ***template-matching*** web-app]

Linear filtering for detection

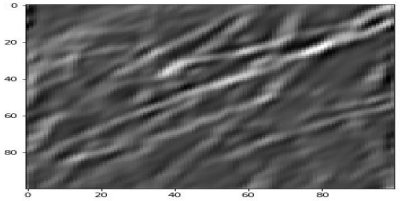
Other filters can for example detect the orientation of elongated structures (e.g. Gabor filters.



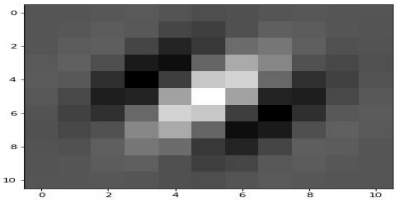
Actin labeling



Zoom



Filtered image

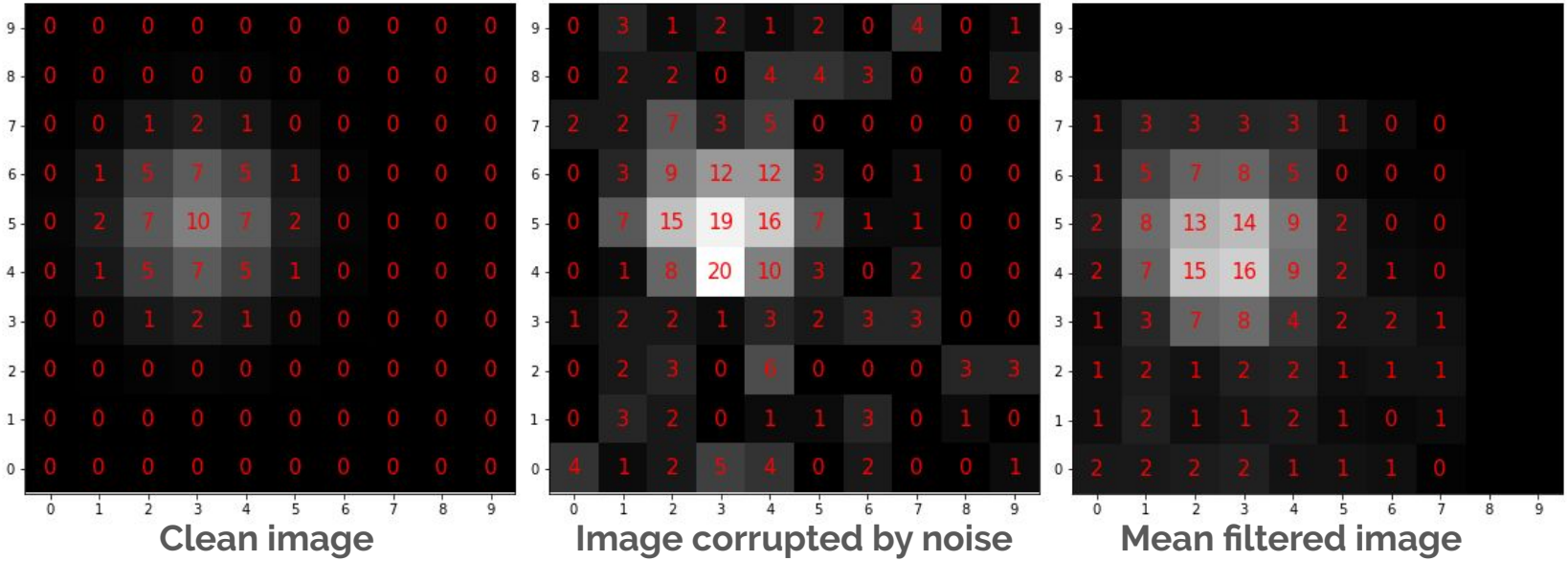


Filter

You can play with the ***orientation*** web-app

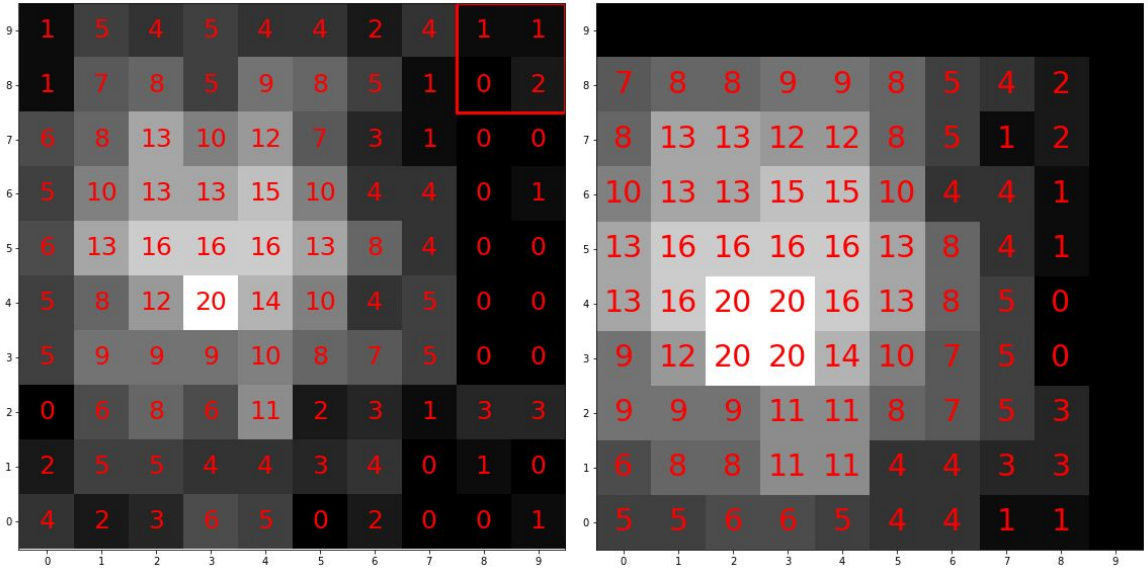
Linear filtering for noise removal

As seen before, if the filter is composed of 1's, it simply calculates a local mean for each pixel (mean filter). This can be used to remove noise in an image: pixel with deviating values are averaged out in this process



Non-linear filters

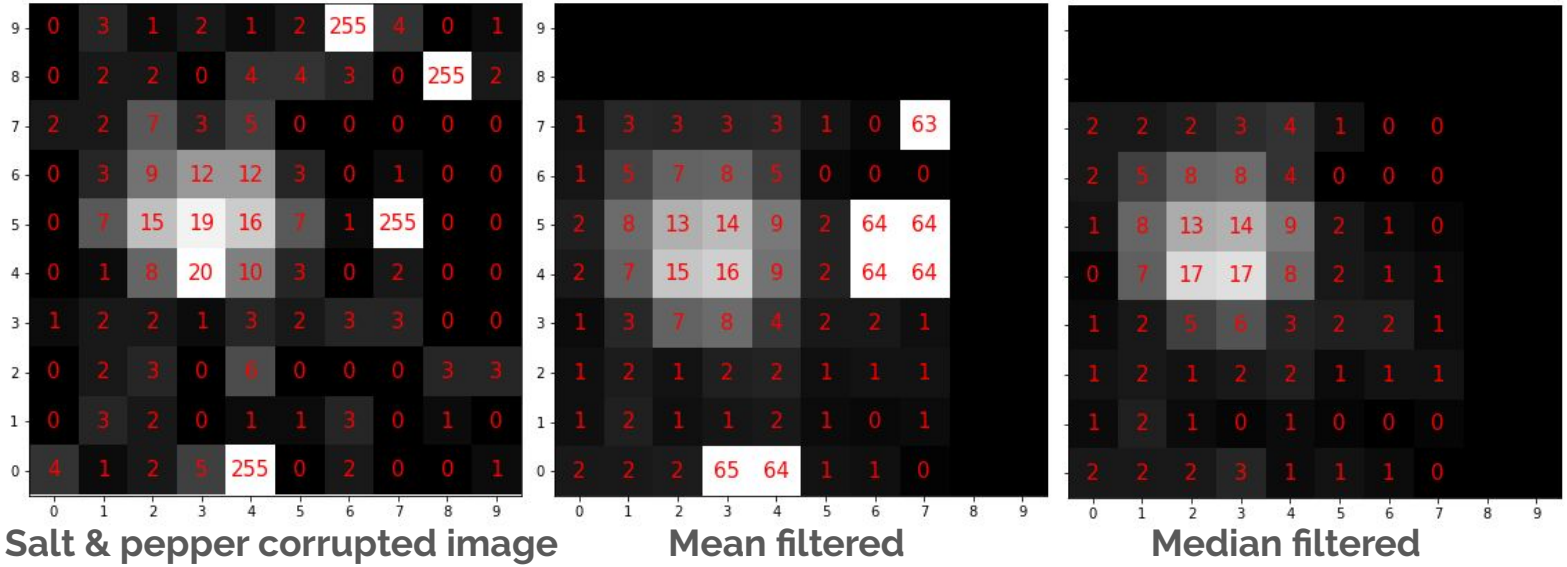
Instead of simply multiplying and summing pixels, non-linear filters can apply an operation on the sub-regions they travel through. For example, one can calculate the local maximum at each pixel position.



$$\max\left(\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 2 \\ \hline\end{array}\right) = 2$$

Different filters for different usages

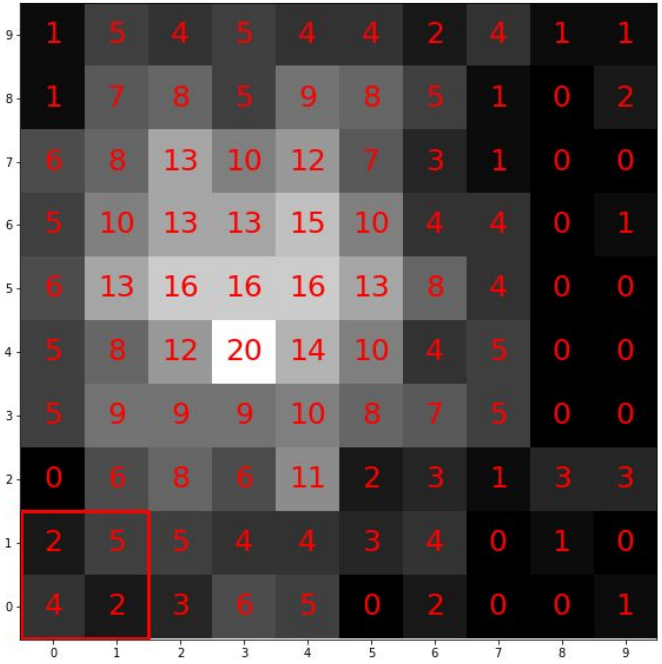
In presence of a few pixels with extreme values, the mean filter is a bad choice. The median filter in contrast is very efficient at this task.

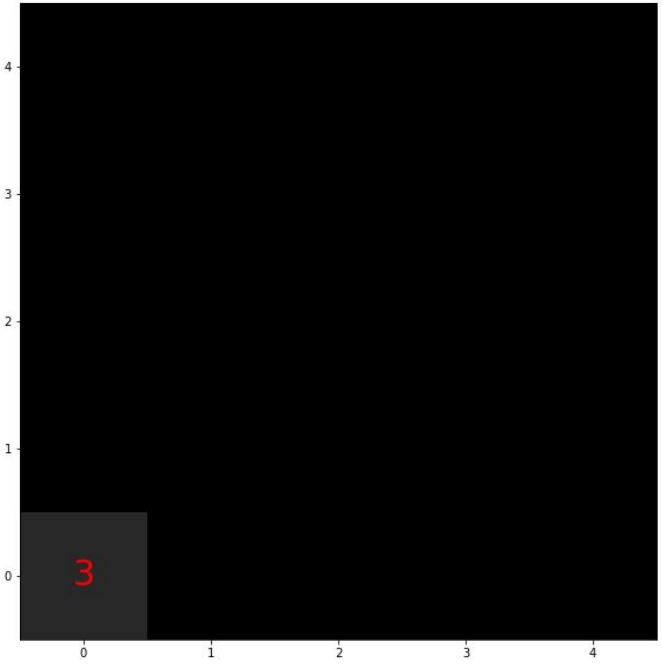


[You can play with the ***filtering*** web-app]

Resampling (downsampling)

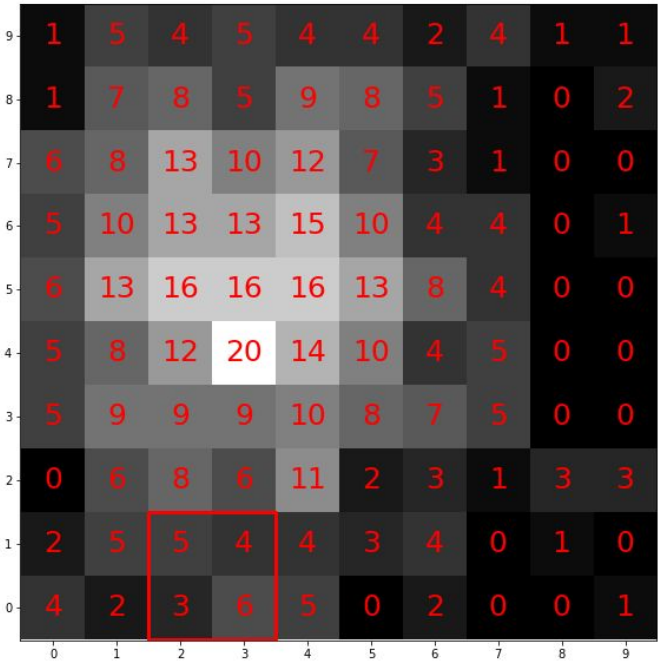
When applying a mean filter on non-overlapping sub-regions, one can down-sample an image. This is often used to decrease computational time.

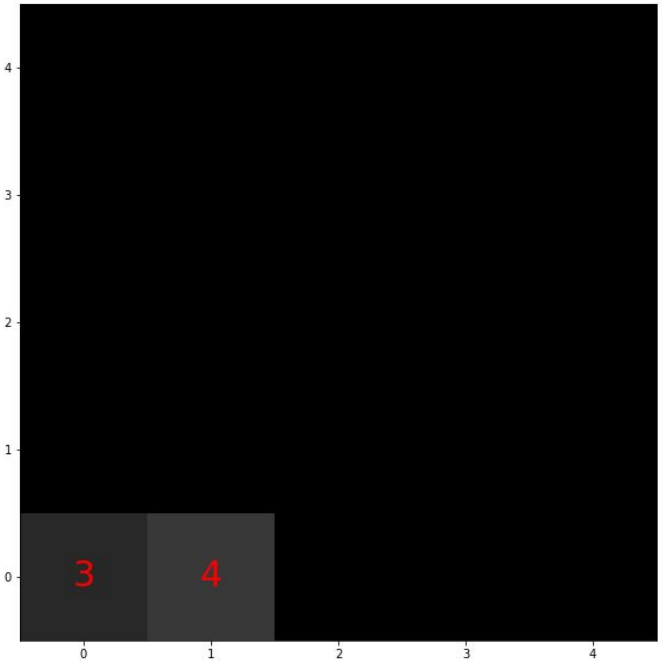




Resampling (downsampling)

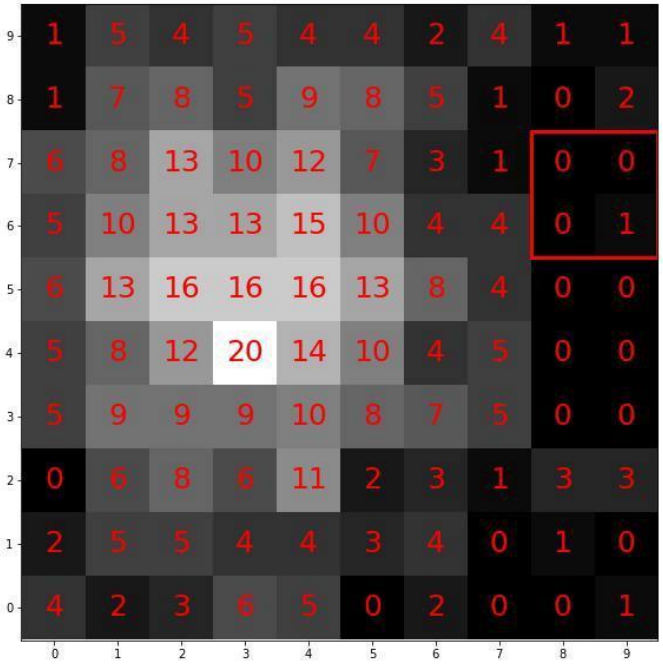
When applying a mean filter on non-overlapping sub-regions, one can down-sample an image. This is often used to decrease computational time.

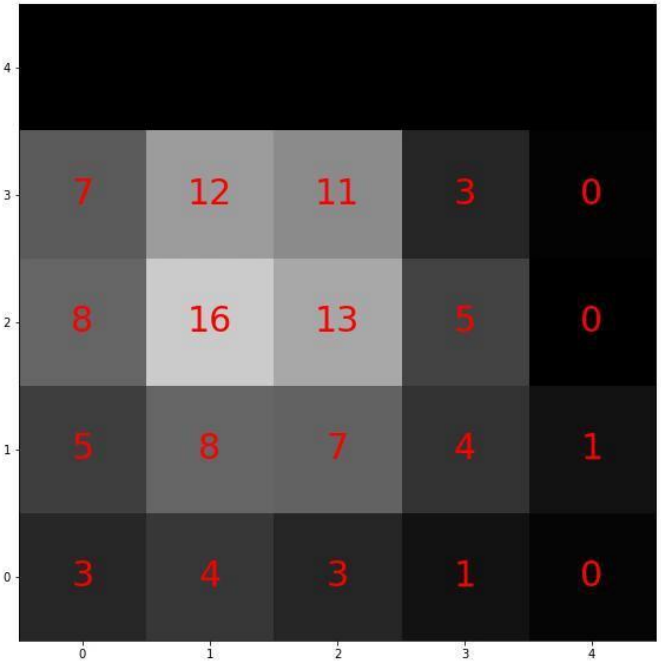




Resampling (downsampling)

When applying a mean filter on non-overlapping sub-regions, one can down-sample an image. This is often used to decrease computational time.

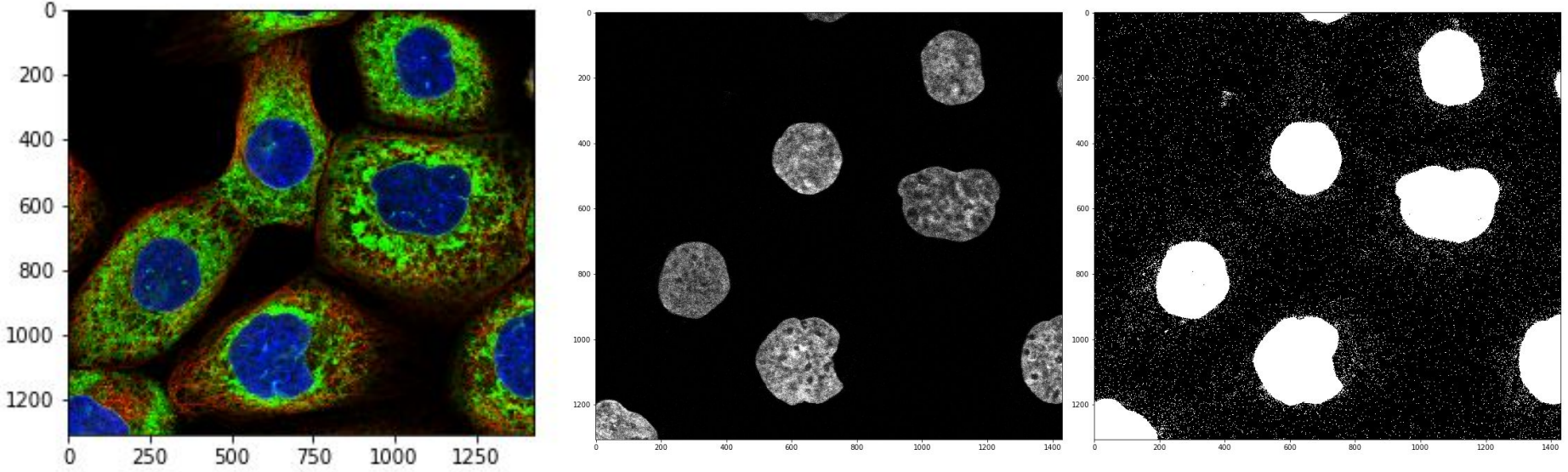




V.
Segmentation

Thresholding (binarization)

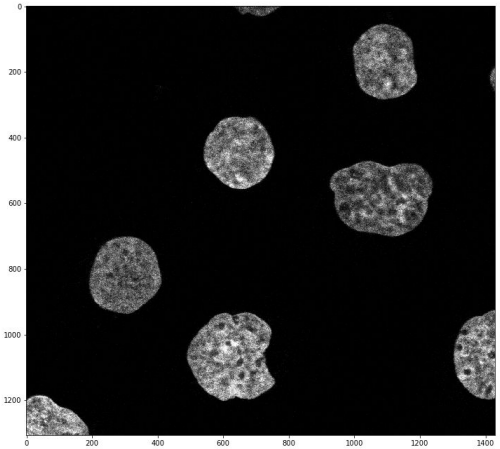
Creation of a binary image where any pixel with a value within a certain range becomes 1 and the others 0. A typical example is the segmentation of nuclei in fluorescence microscopy.

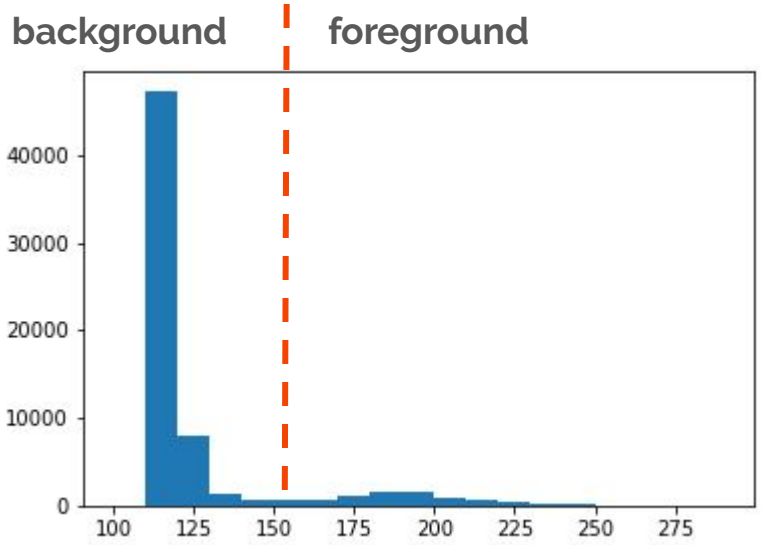


Epiderm human cells, image from the protein atlas (blue nuclei, red tubulin, green atlas antibody)
http://cellimagelibrary.org/images/41678 https://www.proteinatlas.org/ doi:10.7295/W9CIL41678

Thresholding: automatization

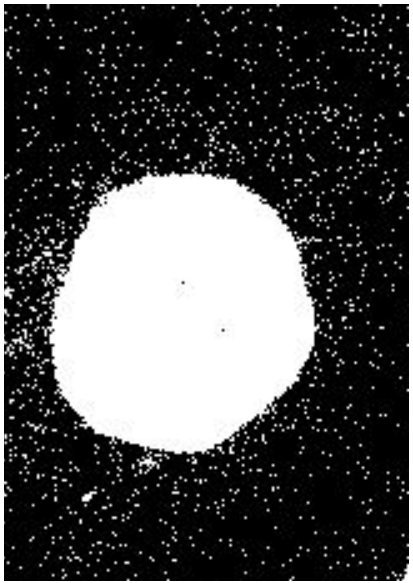
Typically, an image contains foreground and background pixels. This can be seen in the histogram of pixel values composed of two distributions. Automated thresholding methods attempt to find the “best” separation of the two distributions (e.g. minimize the variance of both distributions)

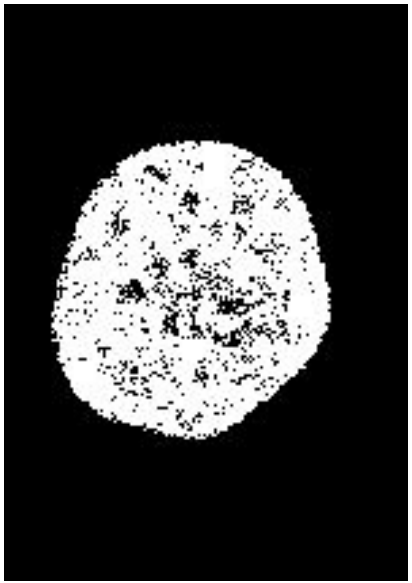




Binary operations

The binary image obtained from thresholding often needs to be cleaned-up. In our example, we might have noisy pixels ON outside the nucleus, or OFF inside





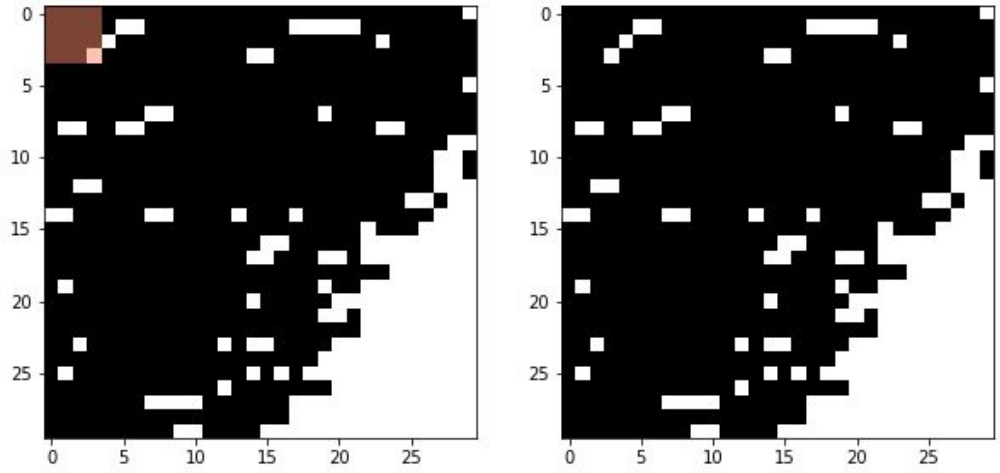
You can play with the ***thresholding*** web-app

Binary operations

To handle these cases, we use binary operators. These are filters that turn pixels ON/OFF depending on the local neighborhood. The simplest ones are the eroding and dilating filters.

Erosion/dilation: Whenever the subregion contains at least an OFF/ON pixel, turn that subregion location (here, upper left corner) OFF/ON

Erosion

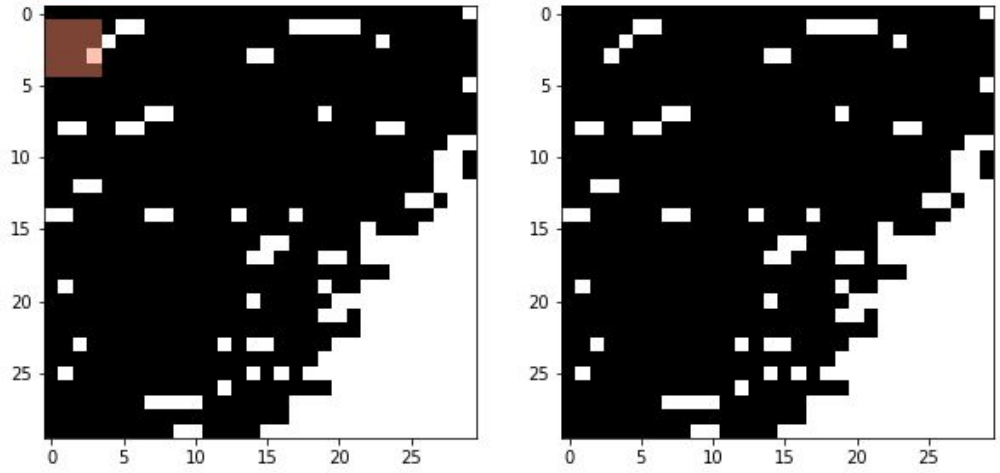


Binary operations

To handle these cases, we use binary operators. These are filters that turn pixels ON/OFF depending on the local neighborhood. The simplest ones are the eroding and dilating filters.

Erosion/dilation: Whenever the subregion contains at least an OFF/ON pixel, turn that subregion location (here, upper left corner) OFF/ON

Erosion

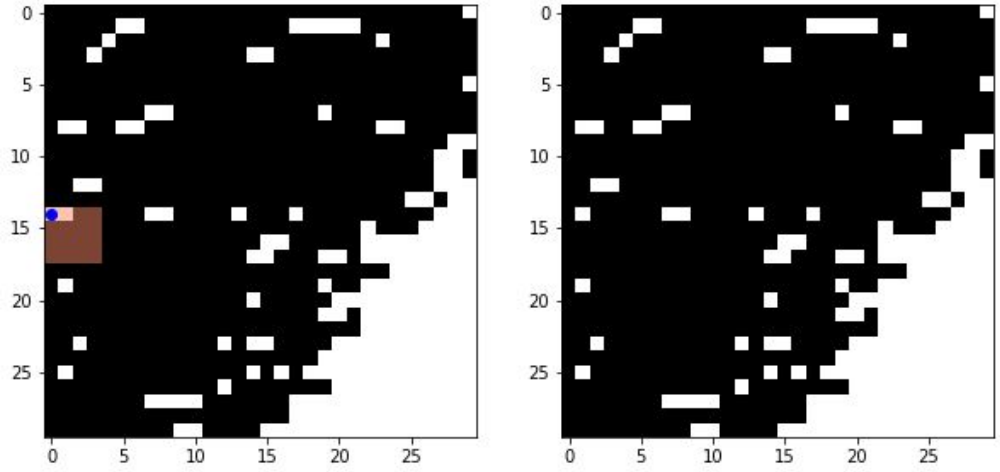


Binary operations

To handle these cases, we use binary operators. These are filters that turn pixels ON/OFF depending on the local neighborhood. The simplest ones are the eroding and dilating filters.

Erosion/dilation: Whenever the subregion contains at least an OFF/ON pixel, turn that subregion location (here, upper left corner) OFF/ON

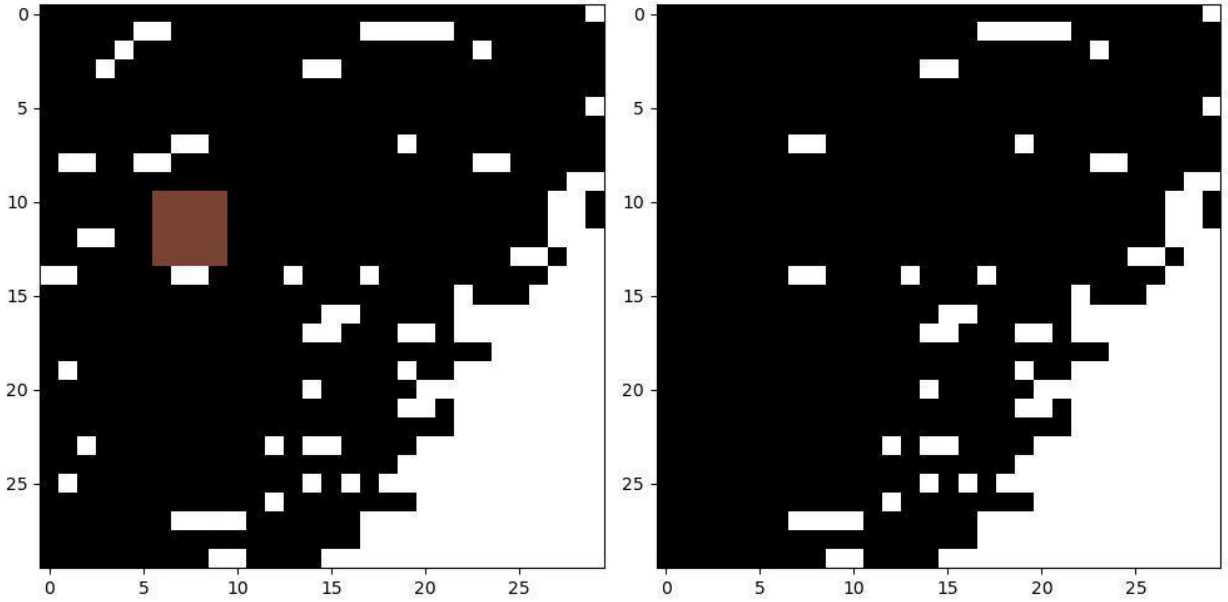
Erosion



You can play with the ***binary*** web-app

Binary operations

To handle these cases, we use binary operators. These are filters that turn pixels ON/OFF depending on the local neighborhood. The simplest ones are the eroding and dilating filters.



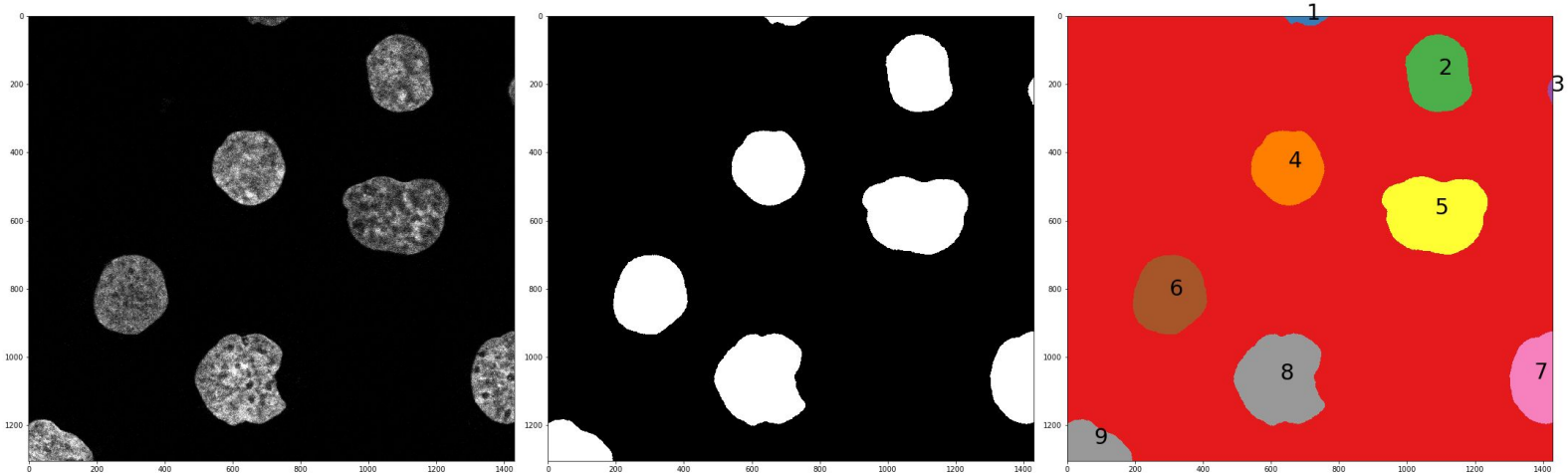
Operators can be combined (e.g. erosion + dilation)

Other more sophisticated operators exist (e.g. thinning)

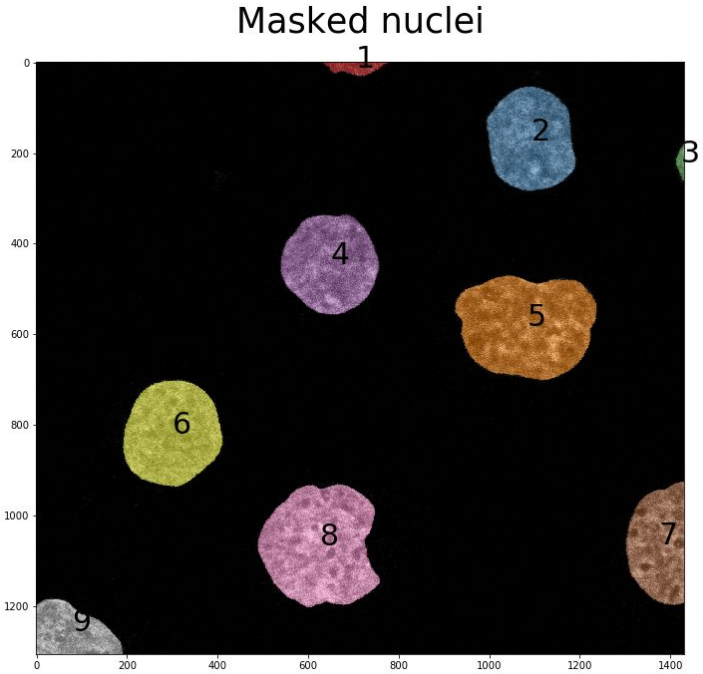
Binary operations: regions

The cleaned binary image (middle), it can be used to recover information:

- About the structures themselves (size, shape etc.)
- About the “content” of these structures imaged in other channels by using the binary image as a **mask**



Binary operations: region properties



Index	Area	Mean intensity	Eccentricity (shape)
1.0	2881.0	69.1	0.97
2.0	35179.0	105.3	0.56
3.0	877.0	88.4	0.97
4.0	36579.0	121.7	0.28
5.0	56095.0	86.1	0.68
6.0	39680.0	93.5	0.44
7.0	27606.0	121.2	0.87
8.0	55195.0	132.4	0.36
9.0	17178.0	146.0	0.83

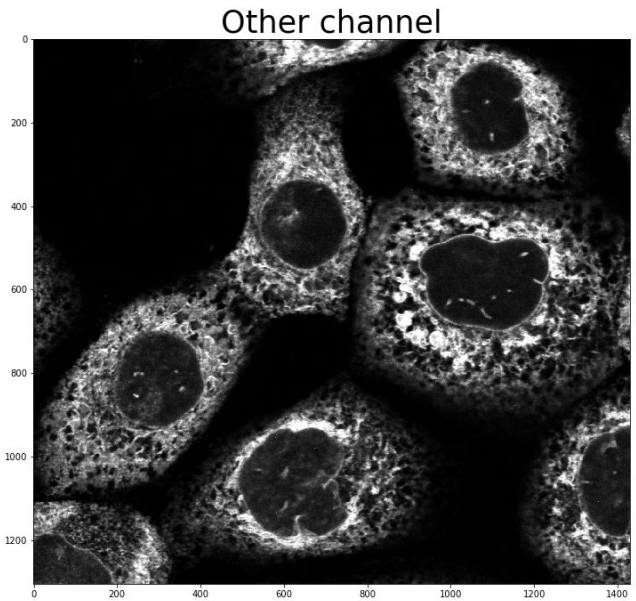
The mask can be used to recover information about the underlying image which can be any channel.

Watershed segmentation

What can we do if we want to segment an image where boundaries are not as clean as e.g. for the nuclei?

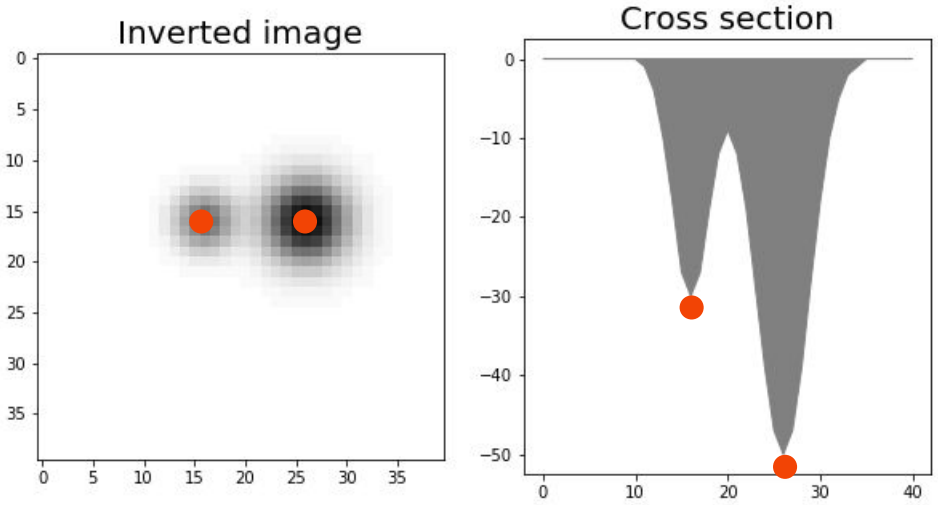
We can exploit our knowledge about nuclei (we know how many cells there are and where they are centered) and “propagate” it to a larger are.

For that we consider the nuclei as “seeds” from where we explore the area around.



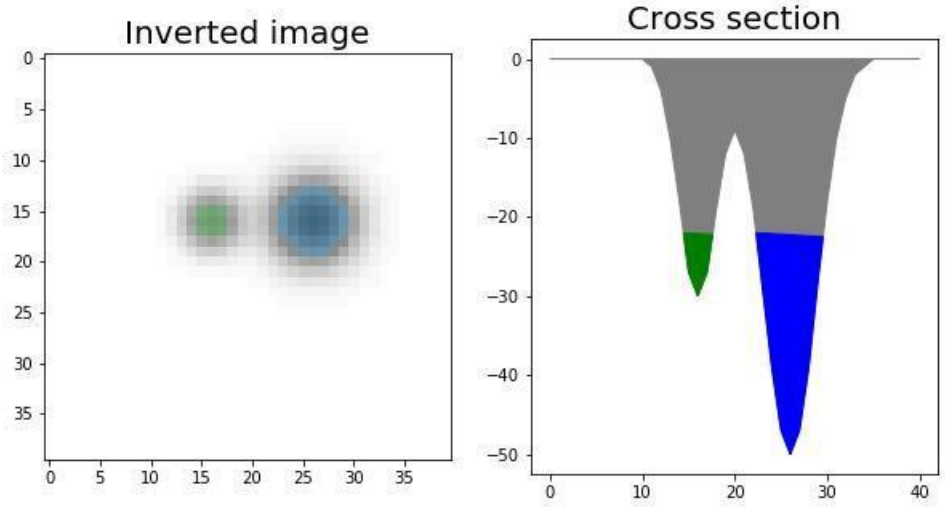
Watershed segmentation

- We start filling from the bottom of the wells, in our case the nuclei **seeds**
- We fill up each well
- When two wells “meet” we create a boundary and extend it until the image is covered (various solutions)



Watershed segmentation

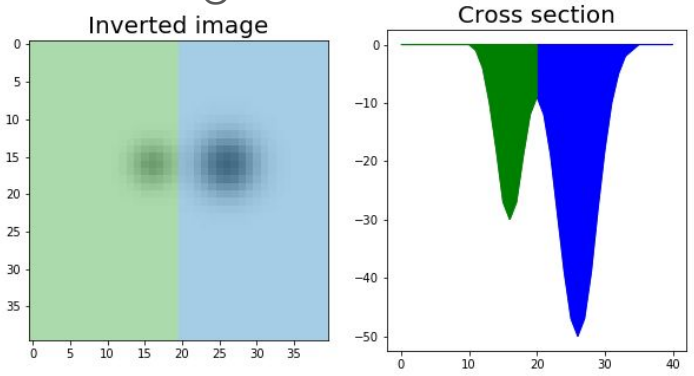
- We start filling from the bottom of the wells, in our case the nuclei **seeds**
- We fill up each well
- When two wells “meet” we create a boundary and extend it until the image is covered (various solutions)



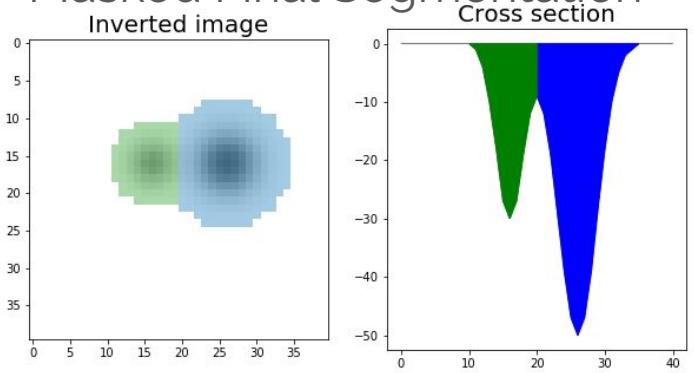
Watershed segmentation

- We start filling from the bottom of the wells, in our case the nuclei **seeds**
- We fill up each well
- When two wells “meet” we create a boundary and extend it until the image is covered (various solutions)
- One can apply at the end a global mask to only retain interesting regions.

Final Segmentation



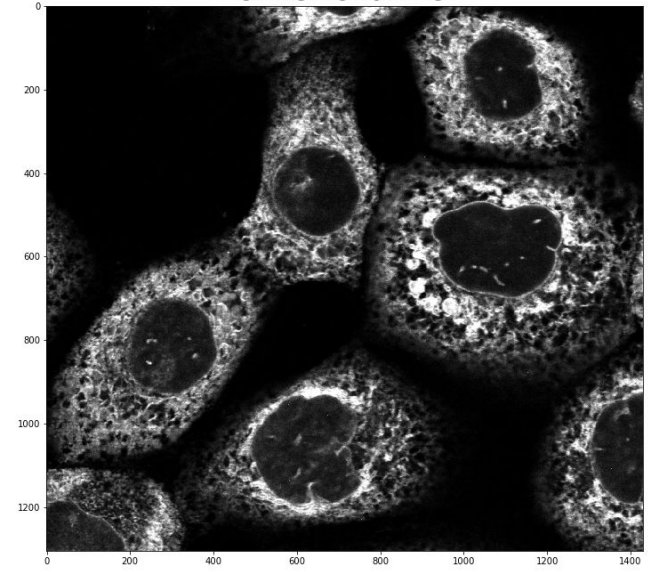
Masked Final Segmentation



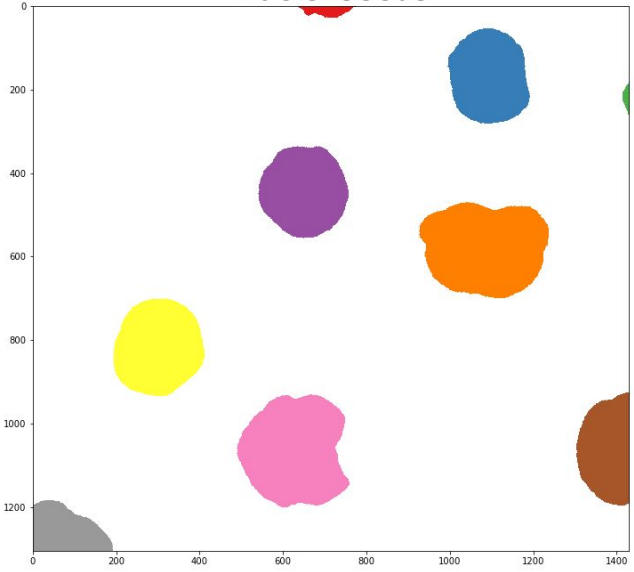
Watershed segmentation

Applying this method on our difficult case, allows us to at least approximately segment the whole cells.

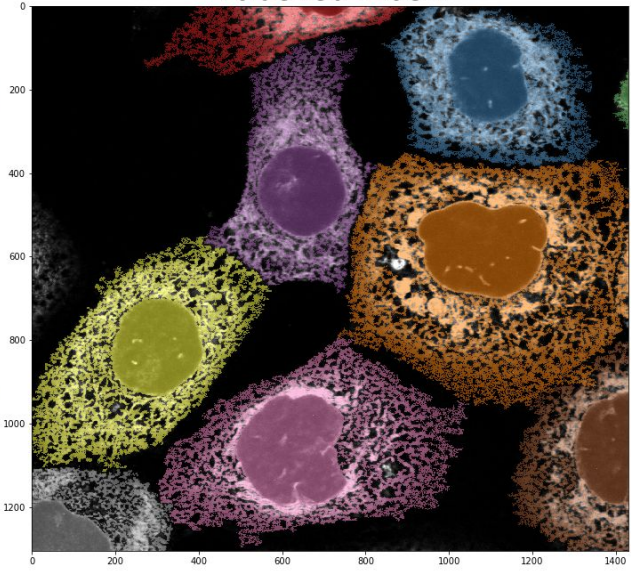
Other channel



Nuclei seeds

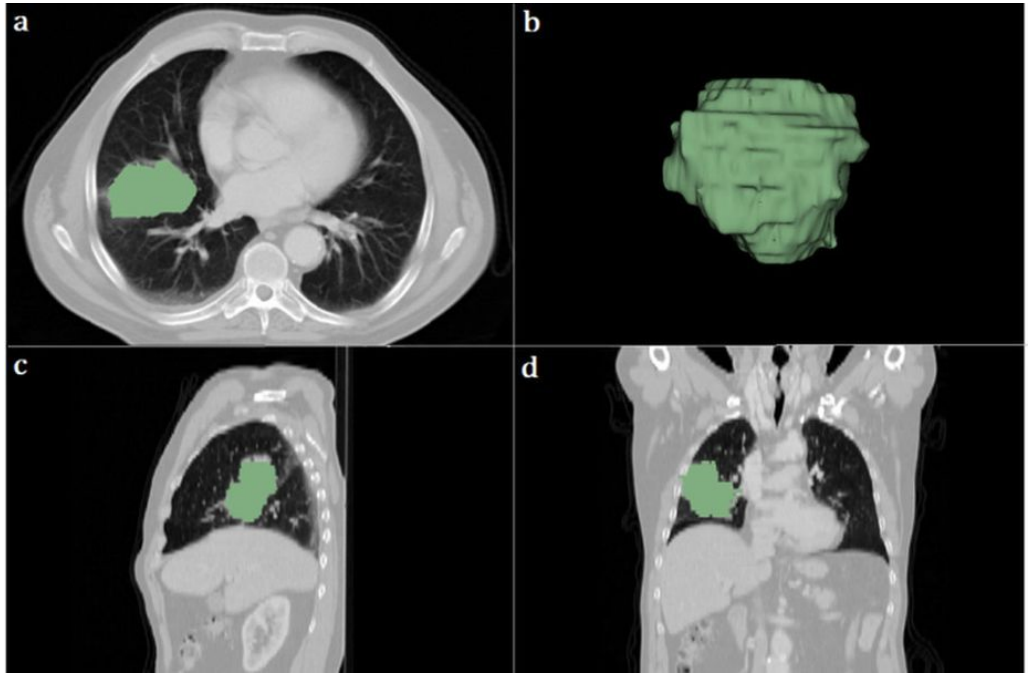


Labelled mask



3D segmentation

Most of the algorithms used in 2D can be extended to 3D at the price of computational time. Here for example, a tumor is segmented in 3D in a lung CT-scan.



Volumetric CT-based segmentation of NSCLC using 3D-Slicer, Velasquez et al. Scientific Rep. 2013

VI.
Advanced image processing

Pixel classification

How do we classify things that we see ? Cat or not cat ?

Four legs ?	0.9
Horns ?	0.1
Pointy ears ?	0.7
Carapace ?	0.01
Fur ?	0.95
etc...	

From having seen many cats we know how these properties match with cats, i.e. our brains have been trained to associate a set of properties with cats.

→ Cat 80 %

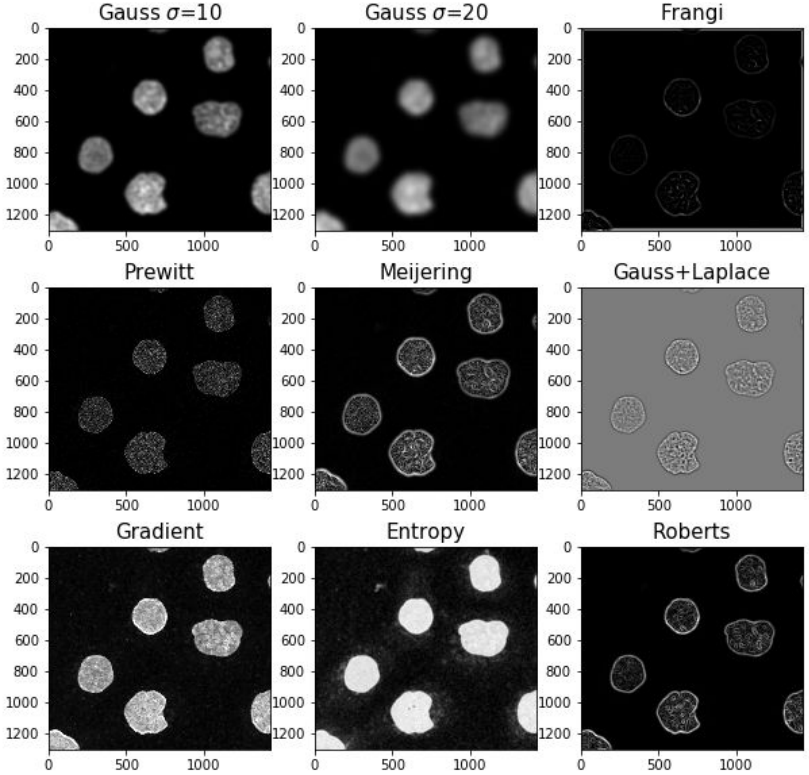
We can do the same for pixels !

Pixel classification: calculating multiple properties

We can apply a series of filters to our image. For each pixel we have now a list of 9 values, one for each filter, that we can use as info (just like “four legs?”, “fur?”) to classify them.

Still missing: what information says “this is a nucleus”? We have to provide learning examples! In this artificial example, we already have the segmentation.

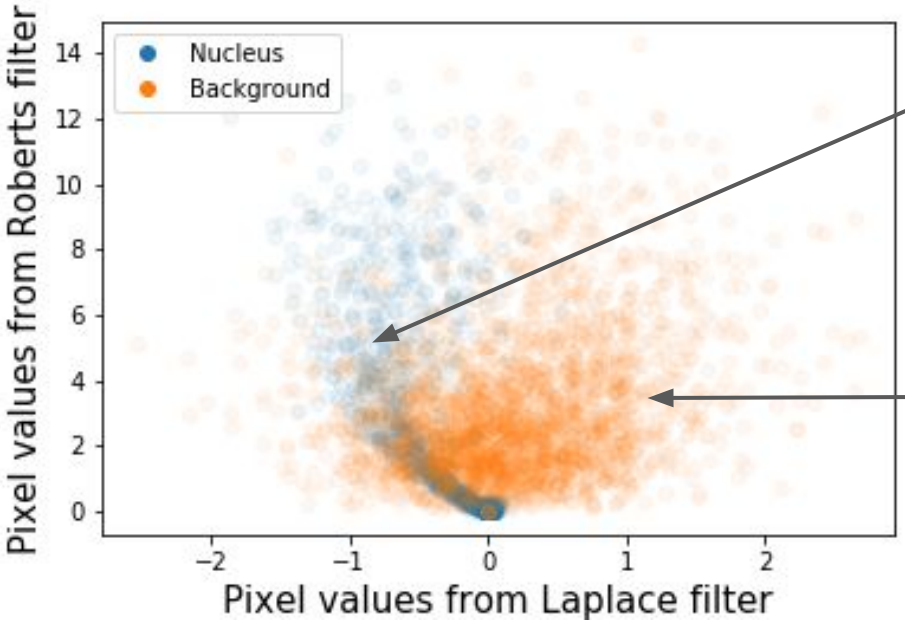
In a normal case we would have e.g. to manually draw regions to provide examples of “what is a nucleus”.



Various filters applied to nuclei image

Pixel classification: with just two filters

Each colored dot represents a single pixel.

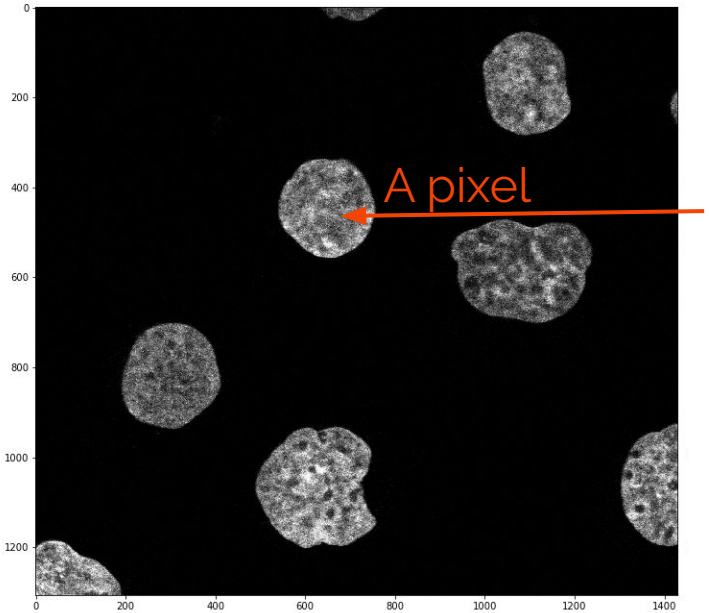


It's unclear what a pixel in this region is: nucleus or background ?

We need another variable!

A pixel with value 1 in Laplace filter and 4 in Roberts filter has a good chance of being Background

Pixel classification



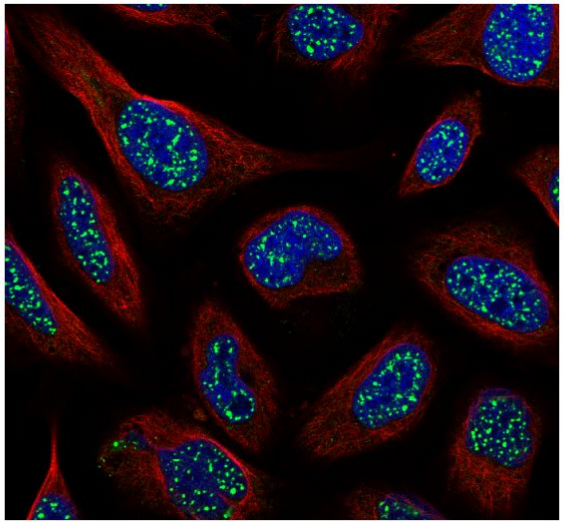
Gauss	12
Frangi	1
Gradient	4
Laplace	23
Entropy	0.1
etc...	

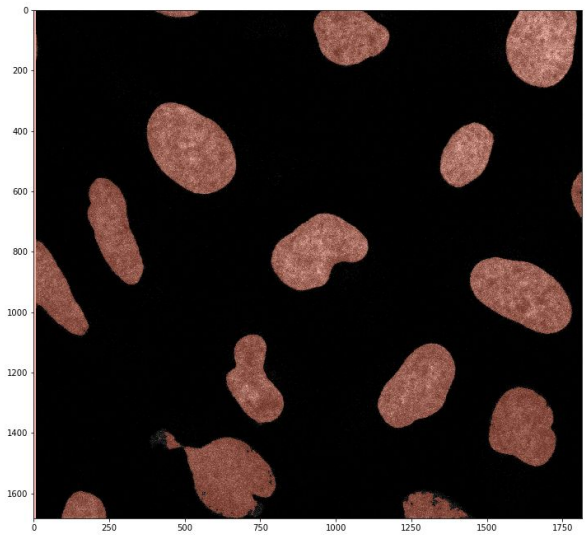
Trained ML algorithm → Nucleus 85 %

Pixel classification: unseen image

We can now use the trained classifier on an image it has never seen.

We don't have to worry about cleaning the image (binary operations). The classifier has learned that dark pixels surrounded by bright ones belong to nuclei.



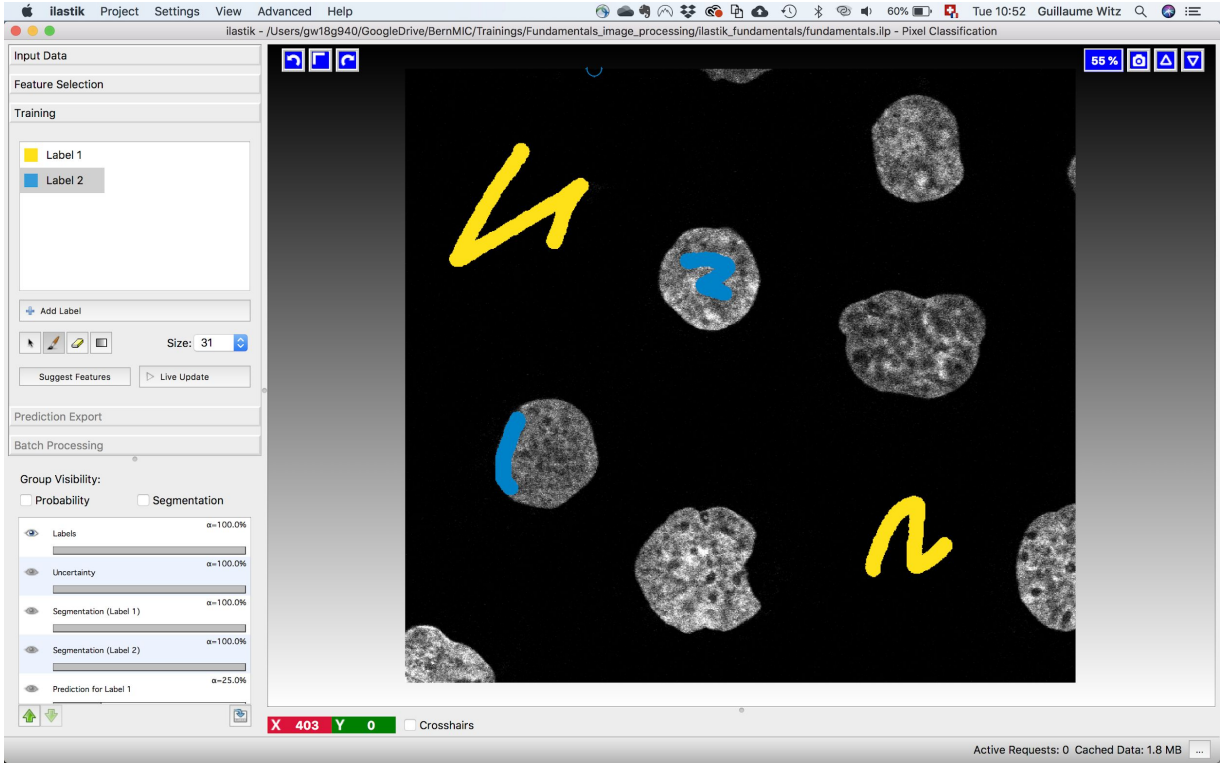


http://cellimagelibrary.org/images/41688
https://www.proteinatlas.org/
doi:10.7295/W9CIL41688

Read-to-use solutions: Ilastik

Usually one does not have a segmentation available for training.

Ilastik is a software where one can manually indicate to what category regions belong.



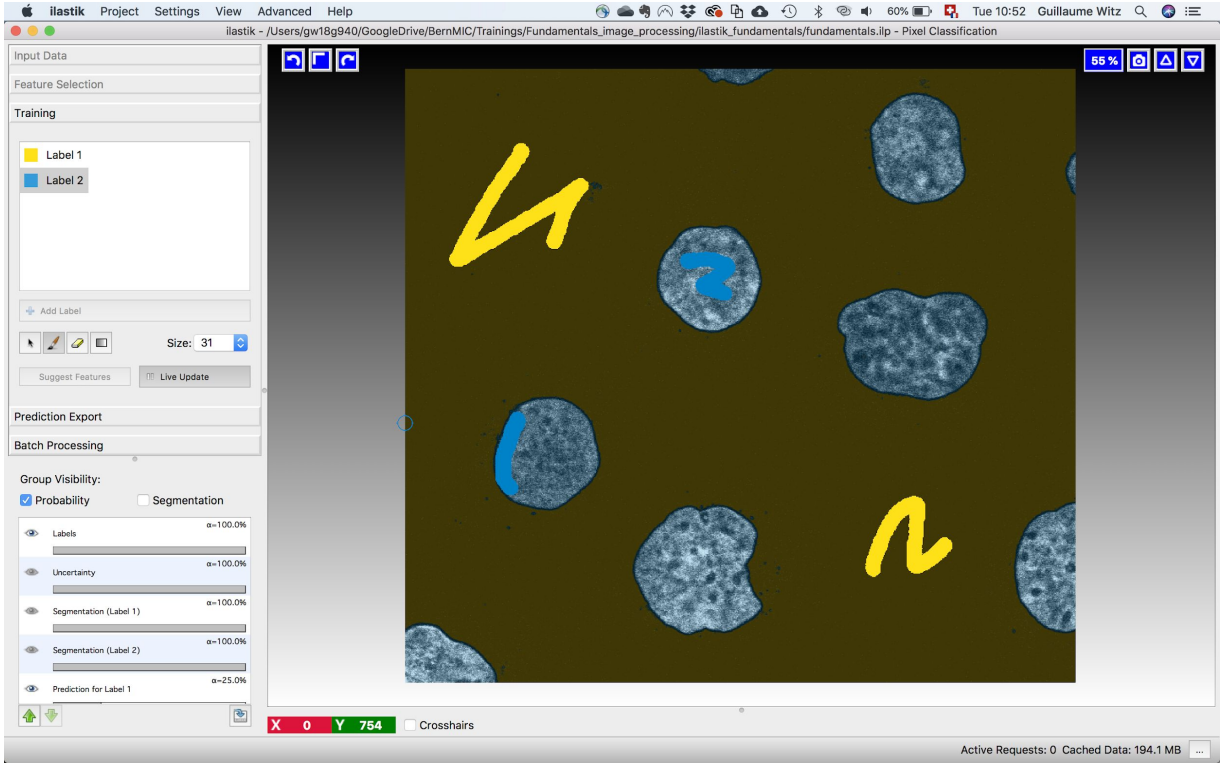
https://www.ilastik.org

Read-to-use solutions: Ilastik

Usually one does not have a segmentation available for training.

Ilastik is a software where one can manually indicate to what category regions belong.

Using that information the software then runs a classifier and provides a segmentation.

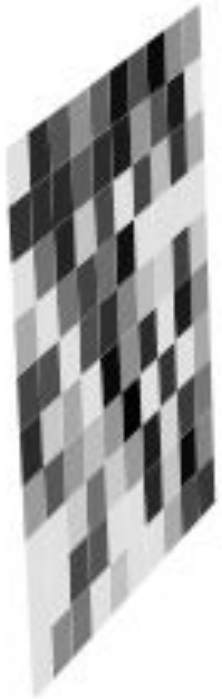


https://www.ilastik.org

Deep learning

Deep learning uses the basic concepts covered before (filtering, downsampling etc.) and combines them in a powerful way.

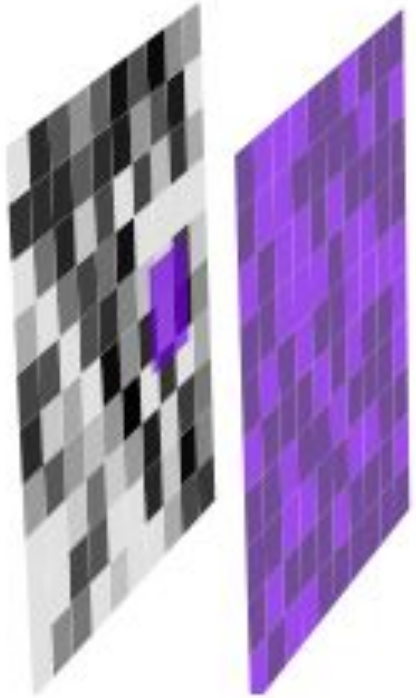
We start with an image we want to segment.



Deep learning

Convolution:

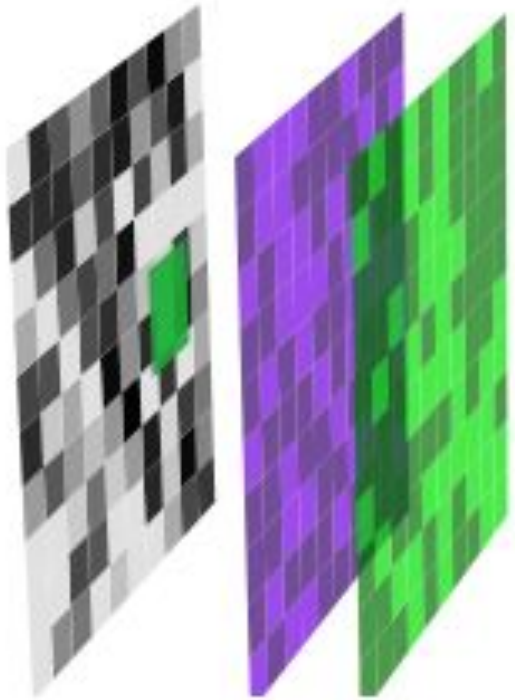
We filter the image with a series of filters to identify features



Deep learning

Convolution:

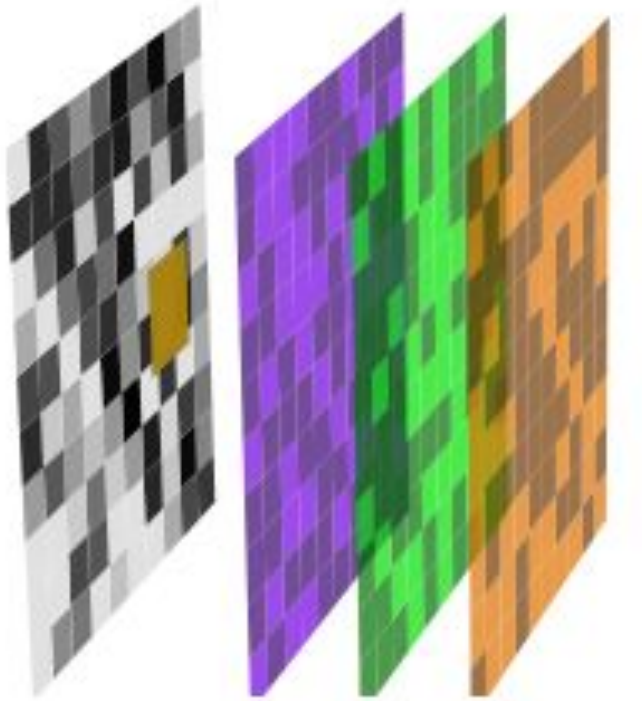
We filter the image with a series of filters to identify features



Deep learning

Convolution:

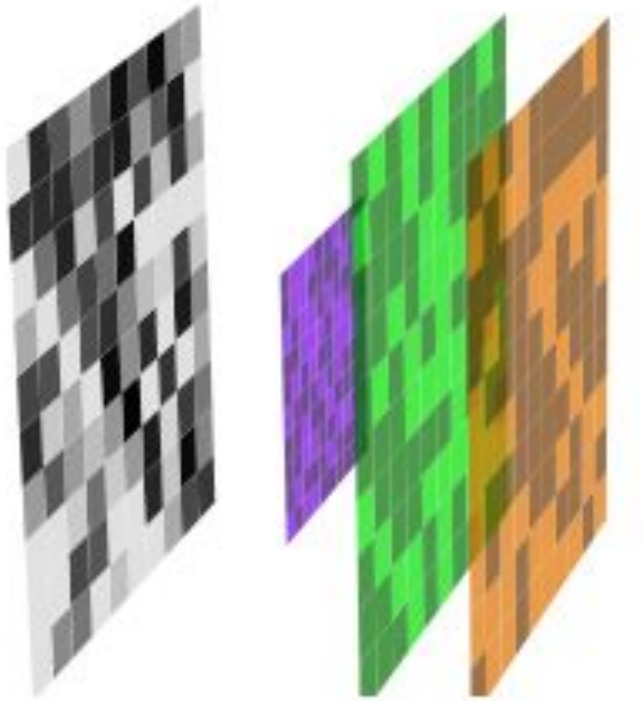
We filter the image with a series of filters to identify features



Deep learning

Pooling:

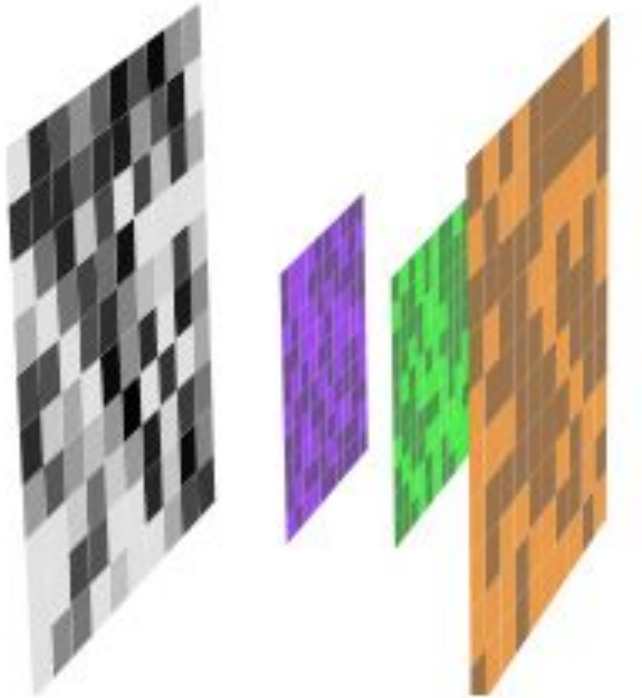
We bin the filtered images in order to look at another **scale**



Deep learning

Pooling:

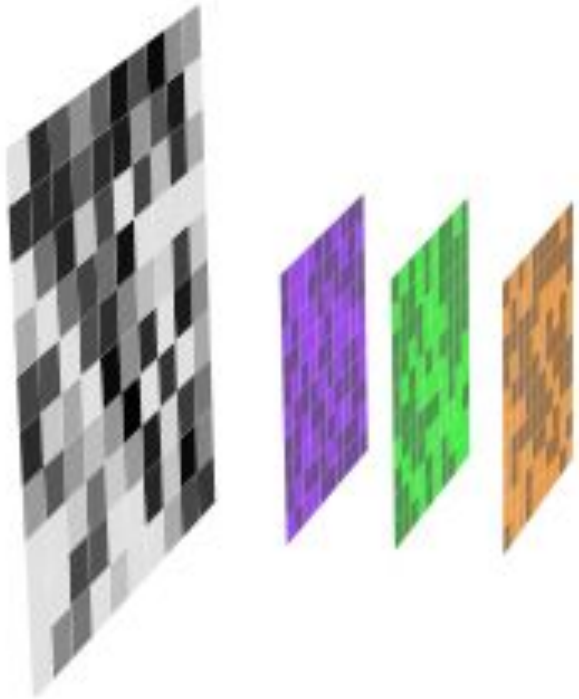
We bin the filtered images in order to look at another **scale**



Deep learning

Pooling:

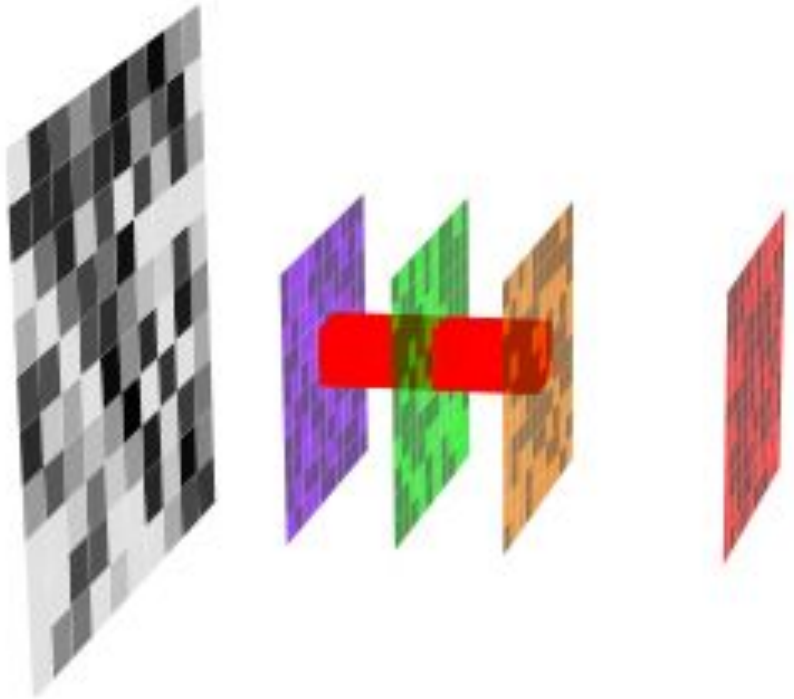
We bin the filtered images in order to look at another **scale**



Deep learning

Convolution:

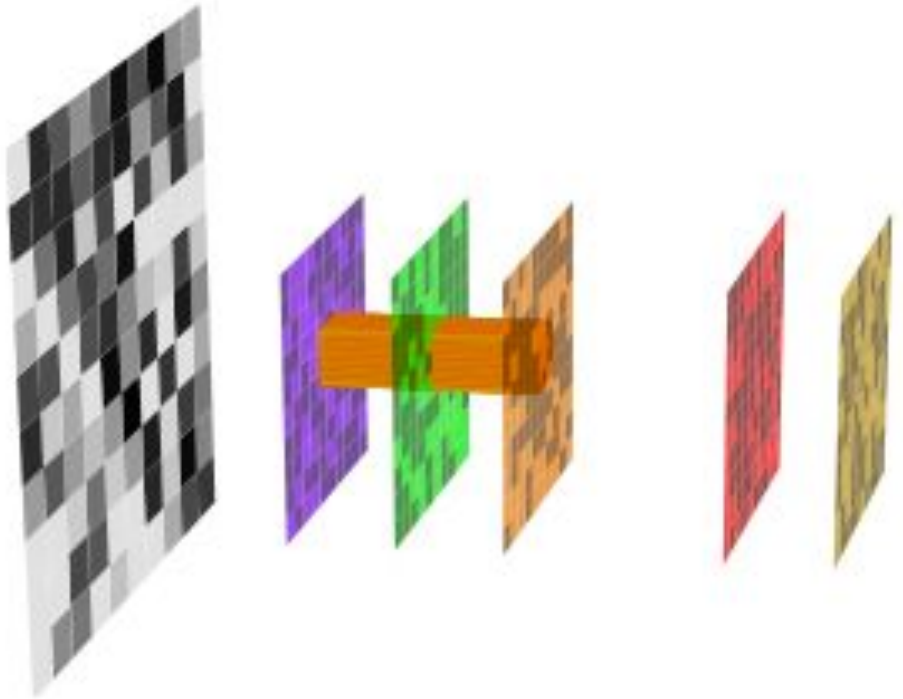
We make a new round of filtering, but now we combine the information of the different layers.



Deep learning

Convolution:

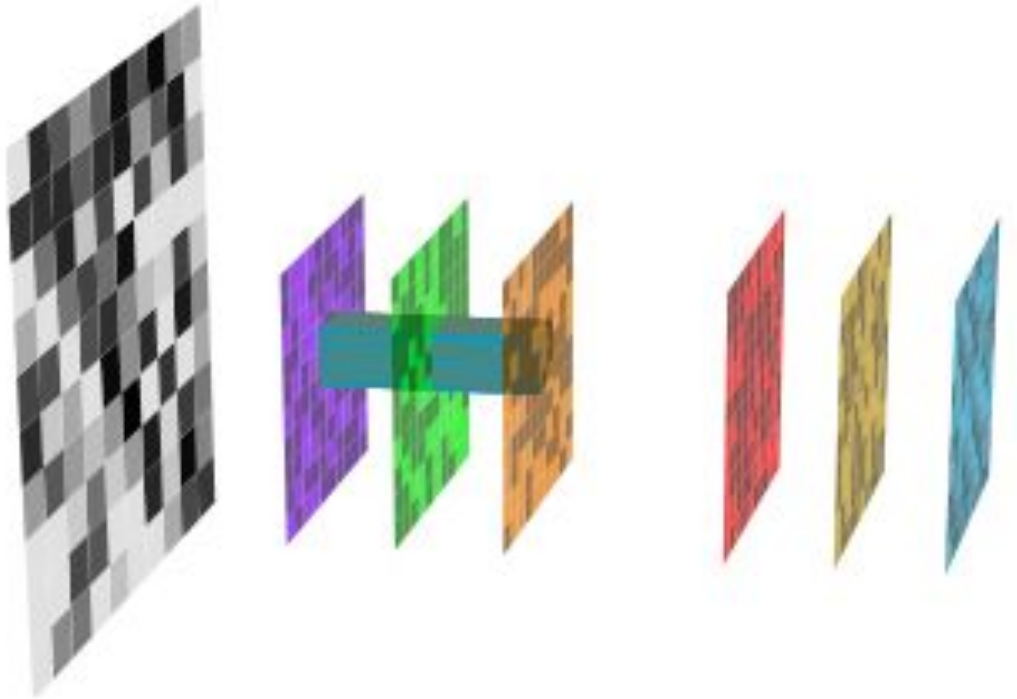
We make a new round of filtering, but now we combine the information of the different layers.



Deep learning

Convolution:

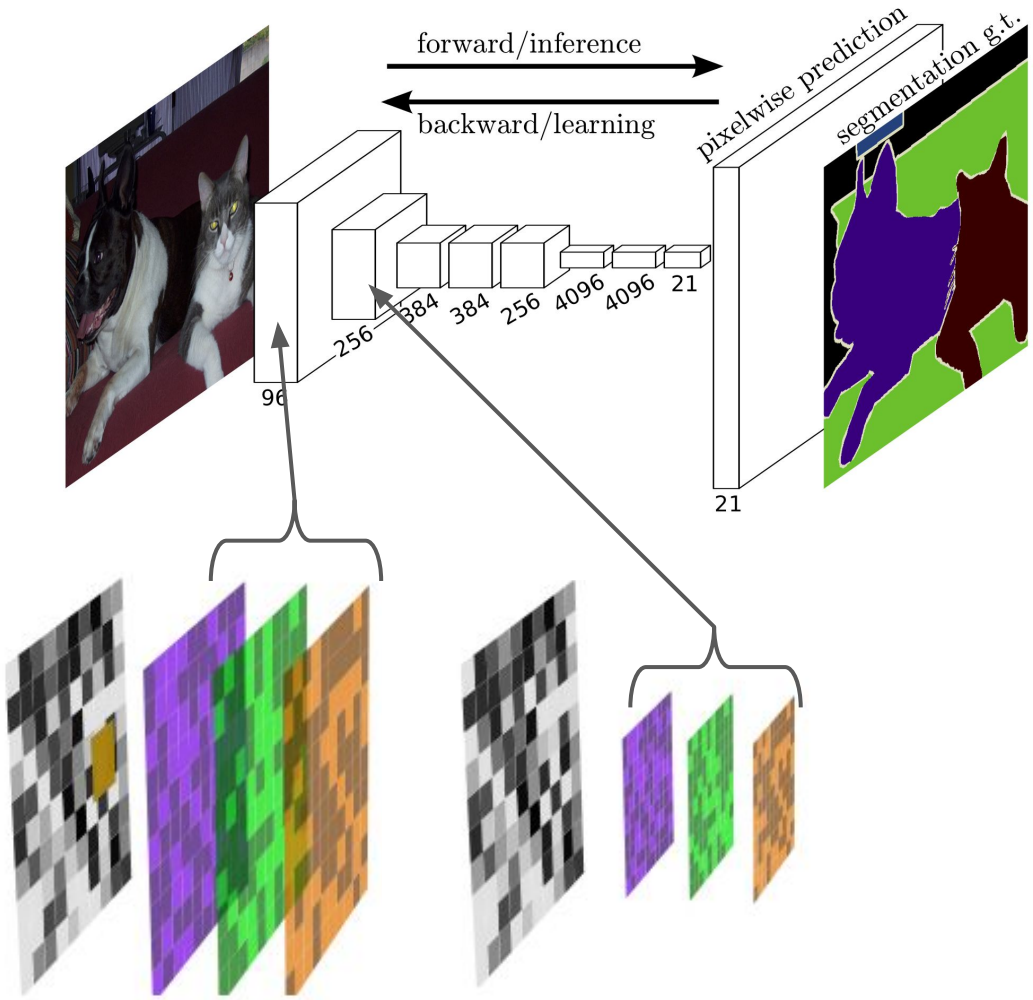
We make a new round of filtering, but now we combine the information of the different layers.



Deep learning

Segmentation:

We can combine many of these layers (**deep** learning). The output of our network should be a map indicating the category of each pixel.

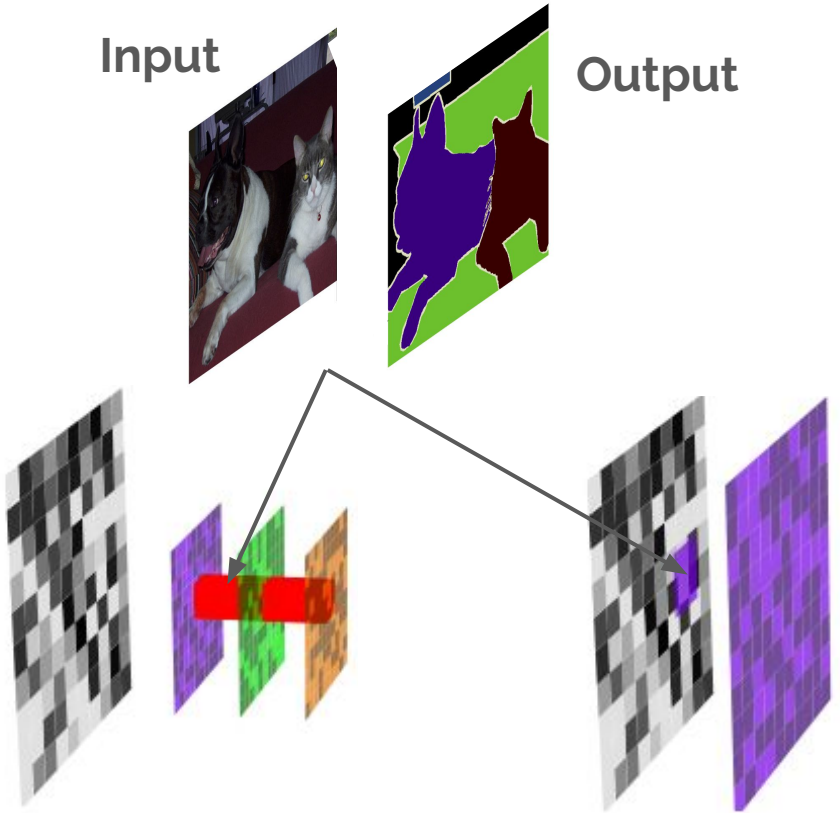


https://arxiv.org/pdf/1411.4038.pdf

Deep learning

Training:

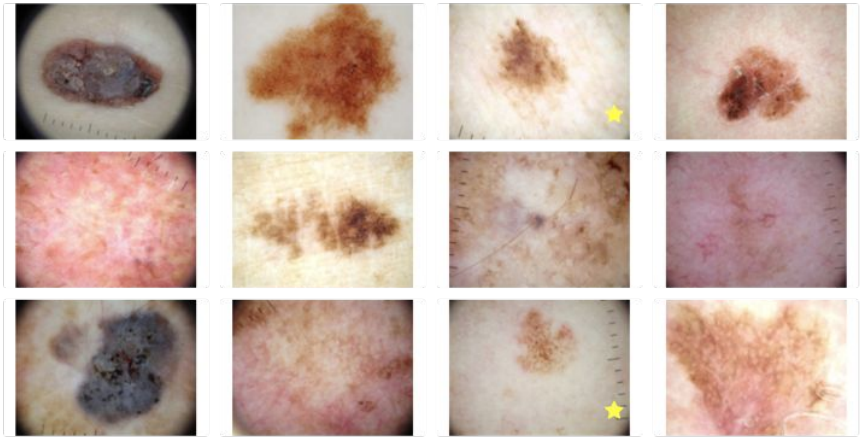
What gets trained here are all the filters. The filters are adjusted in many small steps (as done when fitting are function to data) so that the input of the network (cat picture) matches the output (segmentation) **for the training set.** Then **new images** can be passed through the network.



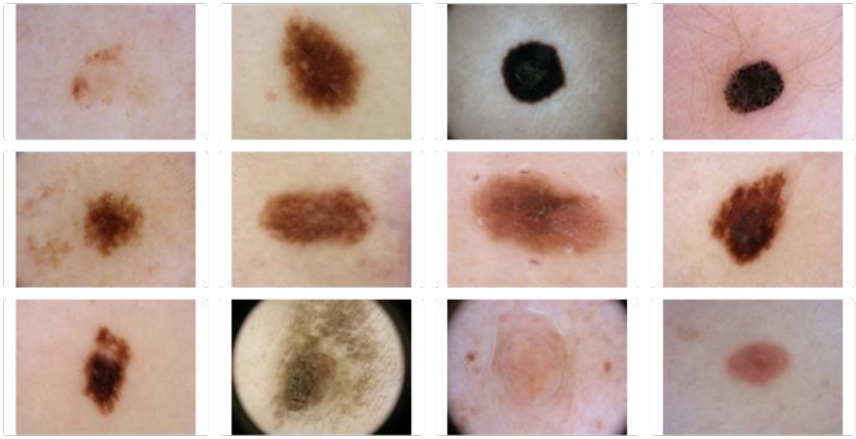
Word of caution

In the pixel classifier seen previously, we designed the filters and could figure out which were important.

In the deep learning case, the inner workings of the network are (for the moment) very difficult to understand. One has therefore to be very careful during training, and thoroughly verify the quality of the results.



Malignant cancer (often with ruler)



Benign cancer (no ruler)

Questions ?

If you want to learn more about image processing, the next date for my 1 day course is 21 January 2020. The material is available here:
https://github.com/guiwitz/Python_image_processing

You can find more infos about that course and the other courses of Science IT Support here:
http://www.scits.unibe.ch/training/training_and_workshops/