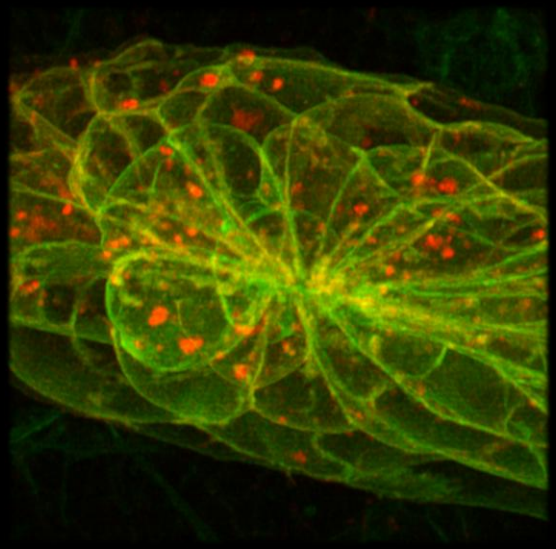
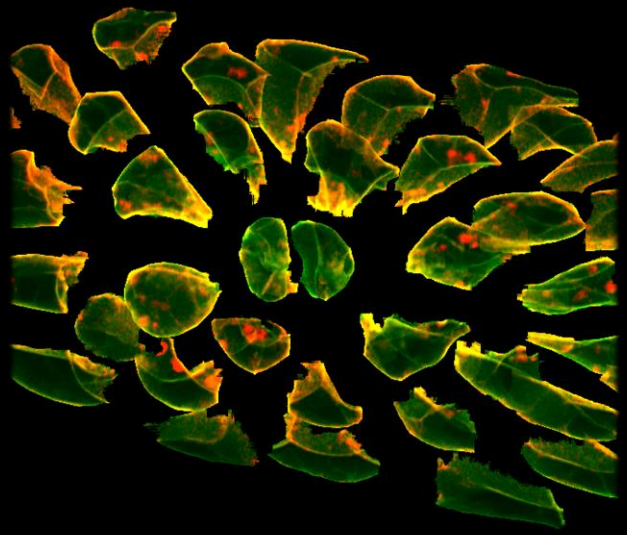
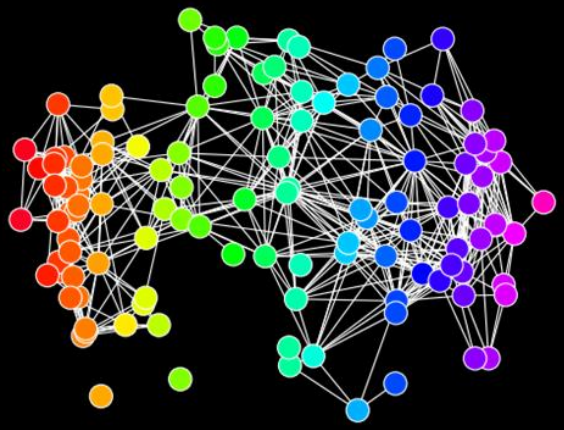
Computational Image Analysis

Jonas Hartmann







EMBL Bio-IT Course: *Image Processing with Python*

8. – 10. May 2017

EMBL Heidelberg

Agenda

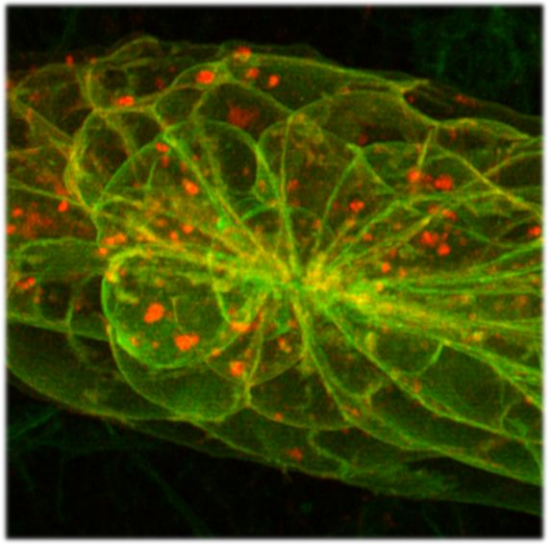
- Intro
- Understanding digital images
- Analysis pipelines & methods
- (Coffee break)
- Intro to course tutorial

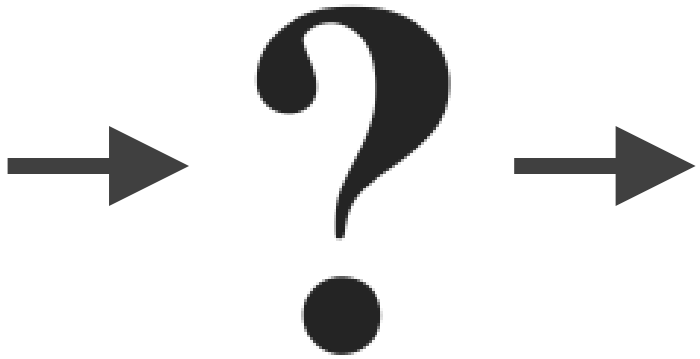
Why Quantitative Image Analysis?

- **Microscopy is an incredibly powerful method for biology**
  - **Abundance and distribution of components**
  - **Spatial organization of sample**
  - **Live samples and 'proper' time courses**
  - **Countless additional methods**
    - **Measurements: FRAP, FRET, photconversion, FCS, …**
    - **Perturbations: laser ablation, drug uncaging, optogenetics, …**
- **However… interpreting image data is hard!**
  - **Images contain a lot of information we *don't* want**
  - **Human eye prone to mistakes and bias***
  - **Classical tools (statistics, plots, …) not directly applicable**

***Important distinction from other 'computer vision' tasks!**

Intro: Why Quantitative Image Analysis?

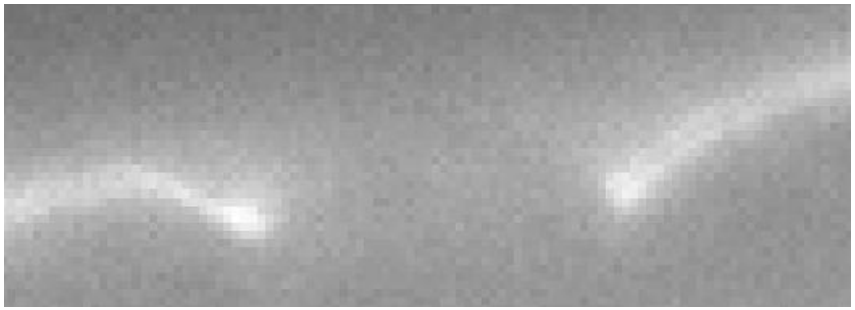




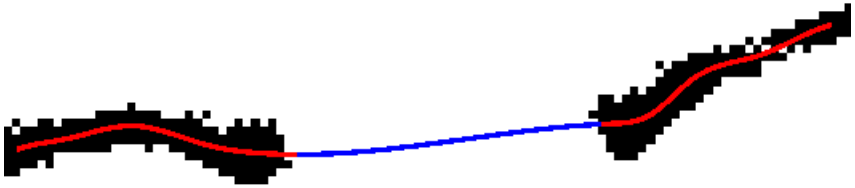
Example: Basal Lamina Invasion

- *C. elegans* basal lamina breached by invasive cell
- Goal: extract intensity profile along breached lamina
- Approach: thresholding and spline fitting

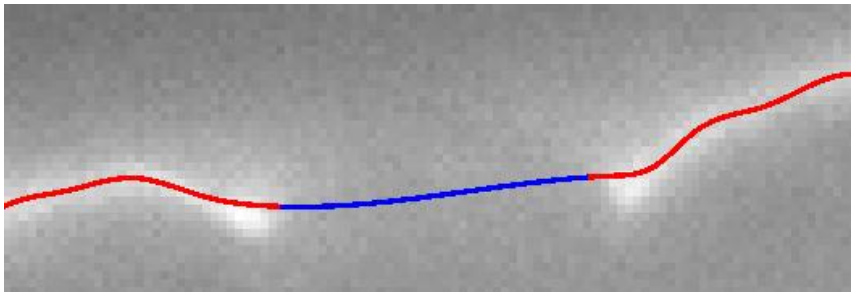
RAW IMAGE

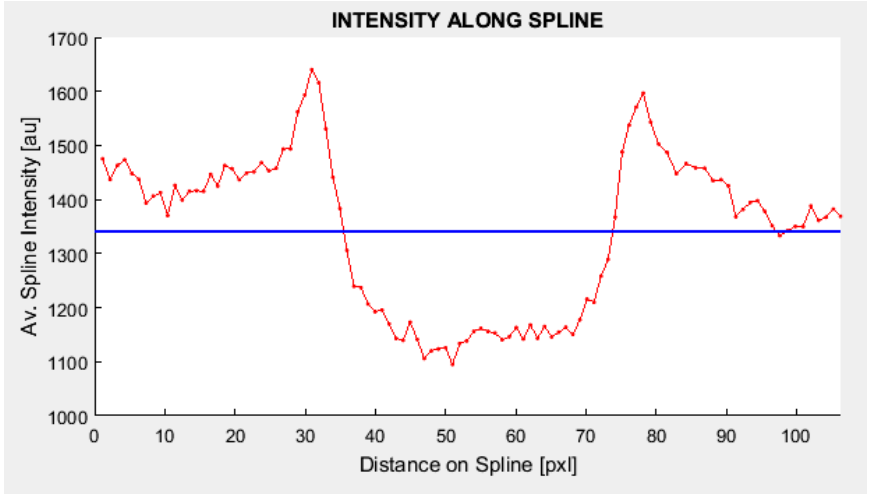


SPLINE FIT OVER THRESHOLDED IMAGE



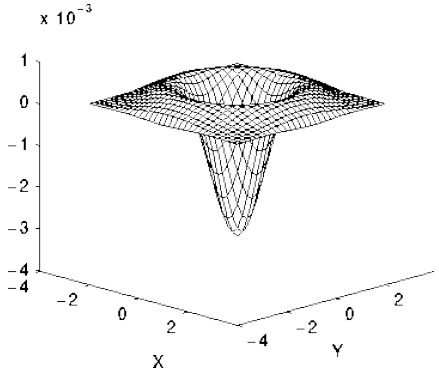
SPLINE FIT OVER RAW IMAGE



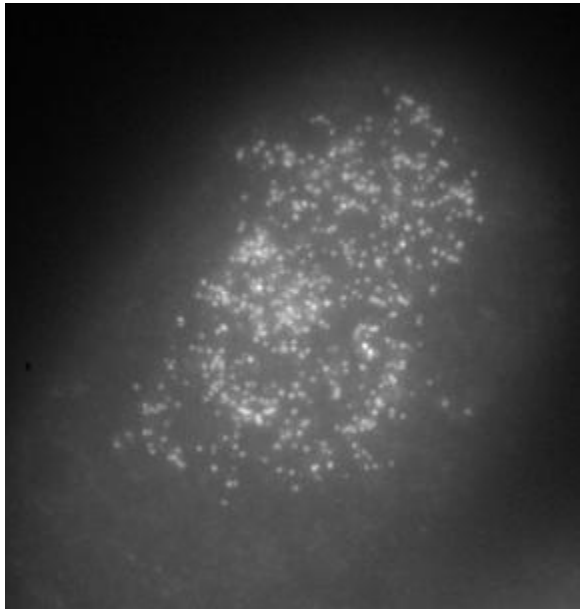


Example: smFISH Spot Detection

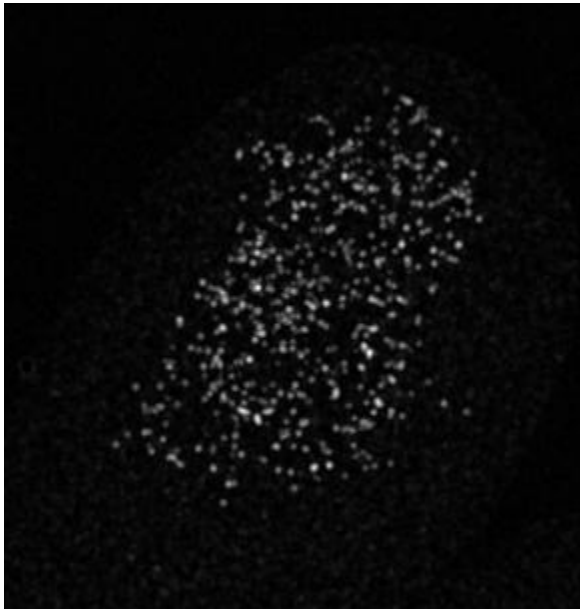
- Single-molecule Fluorescence In-Situ Hybridization
- Goal: detect and count spots
- Approach: LoG filter and thresholding



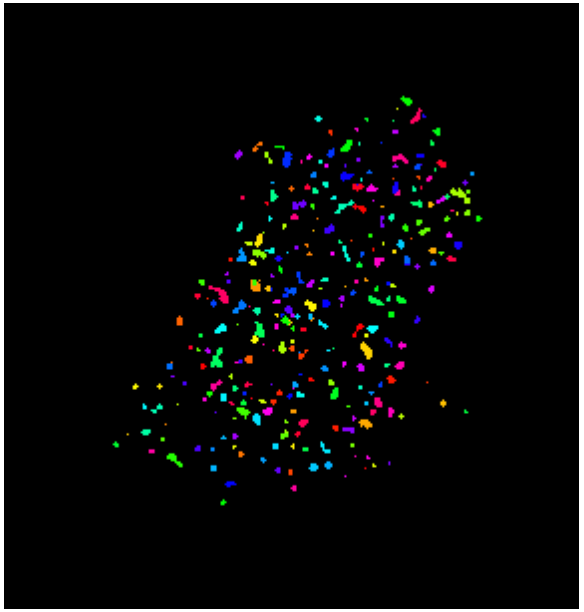
RAW IMAGE



LoG FILTERED



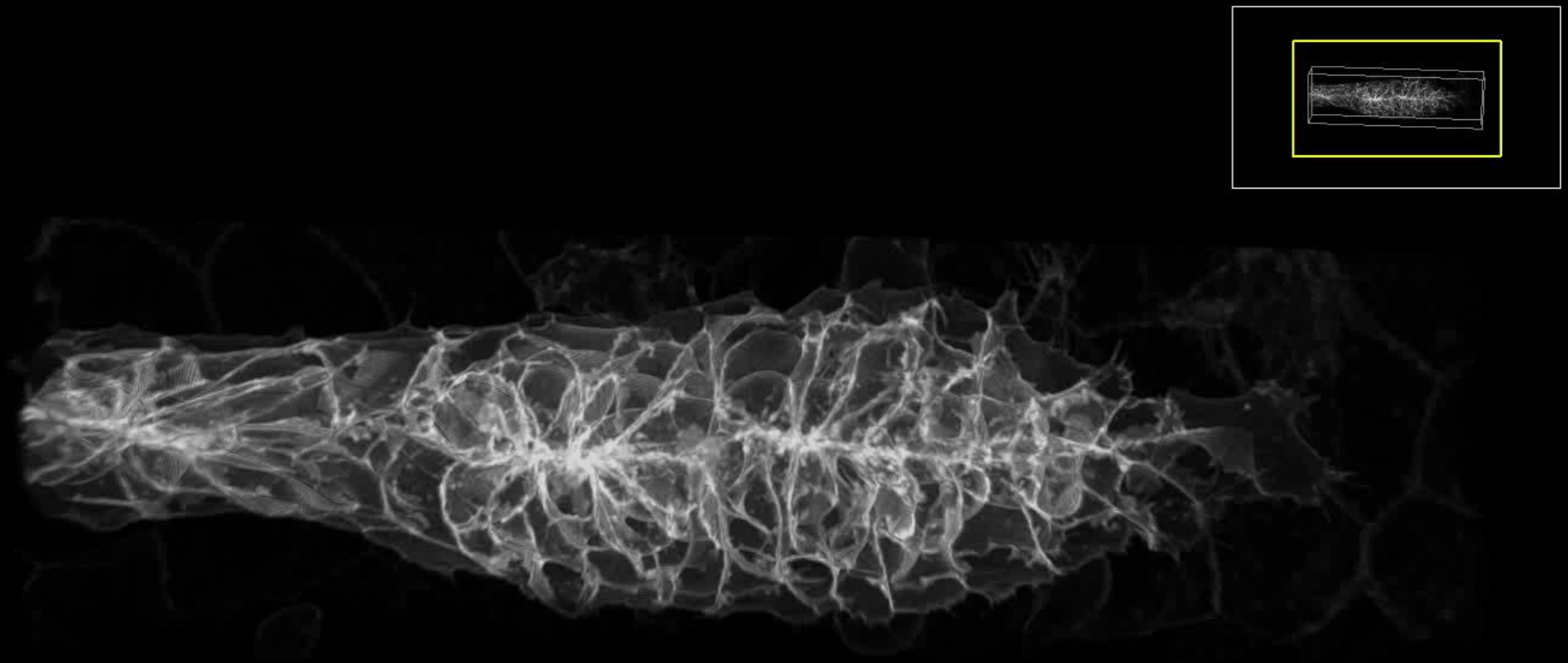
LABELED SPOTS



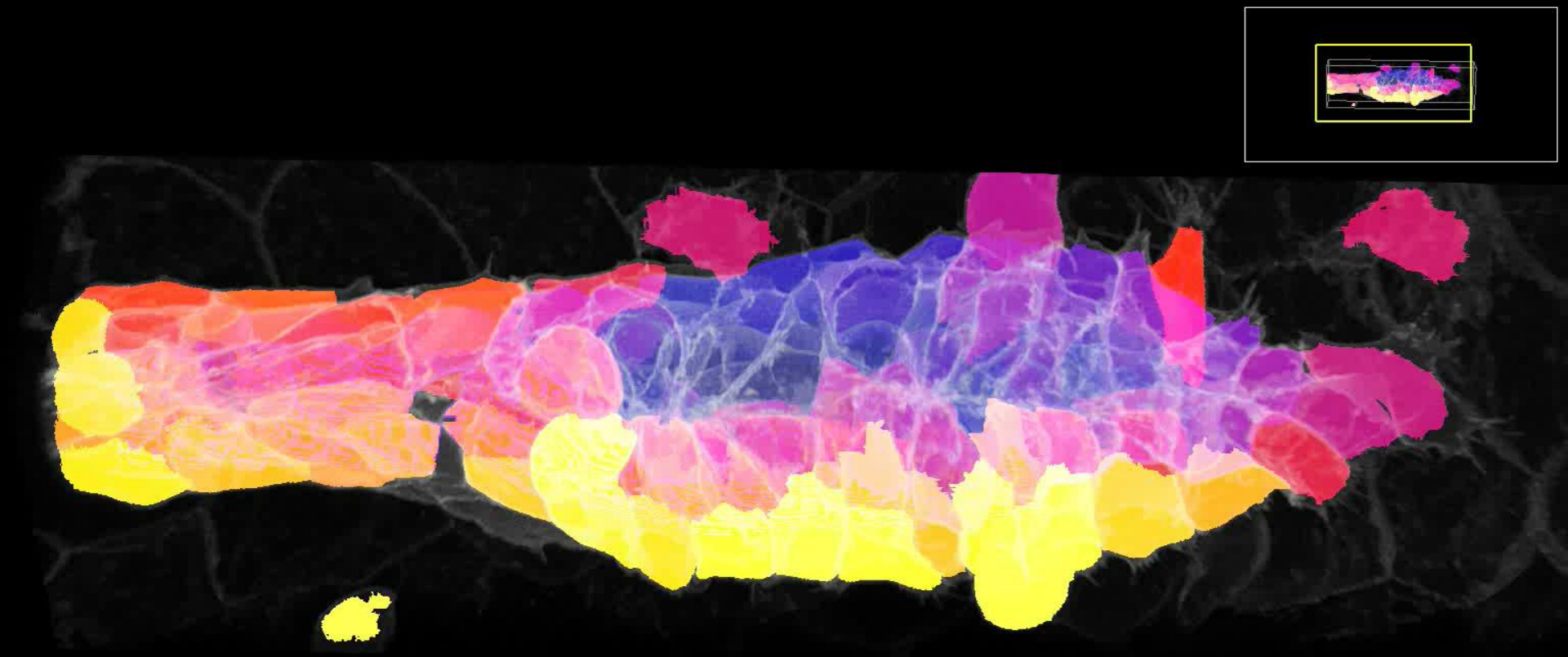
Example: 3D Single-Cell Segmentation

- 3D membrane label stack of zebrafish lateral line primordium
- Goal: segment each individual cell
- Approach: multi-step pipeline

Example: 3D Single-Cell Segmentation

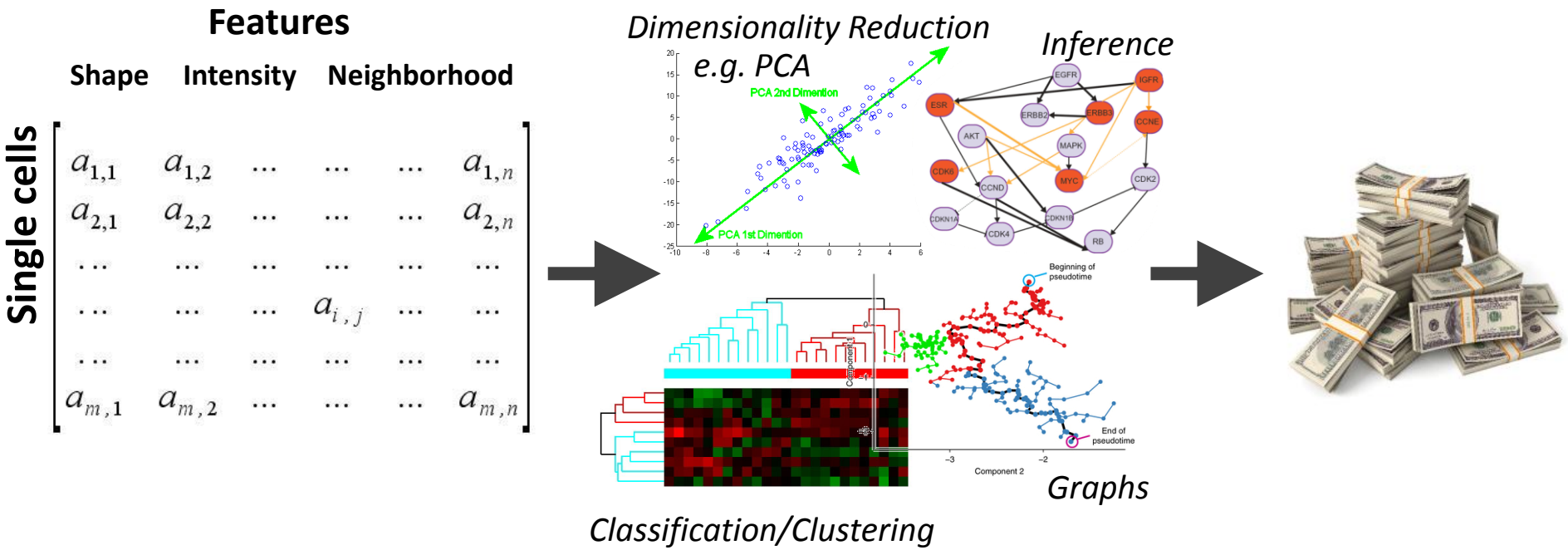


Example: 3D Single-Cell Segmentation



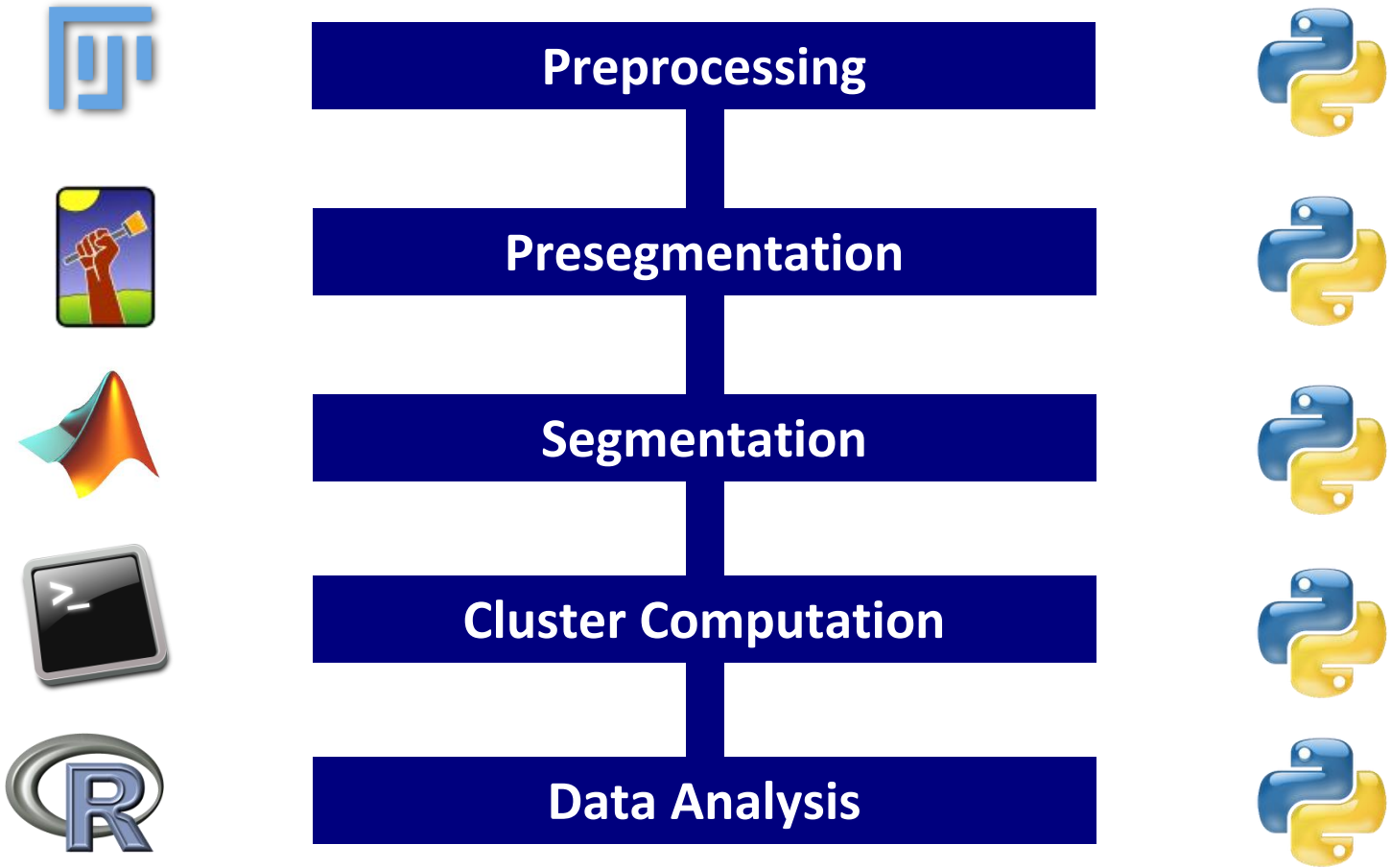
Example: 3D Single-Cell Segmentation

- 3D membrane label stack of zebrafish lateral line primordium
- Goal: segment each individual cell
- Approach: multi-step pipeline
- Next: feature extraction and single-cell analysis



Why Python?

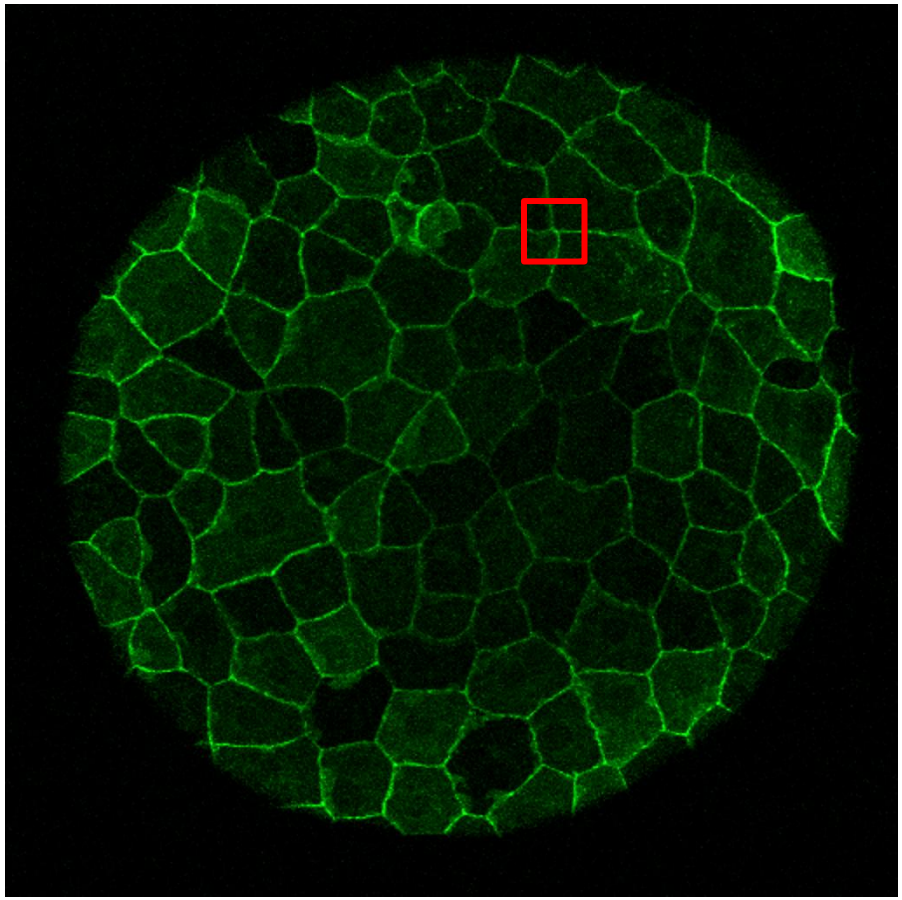
- The usual reasons: easy, versatile, packages, community, …
- But for image analysis in particular:

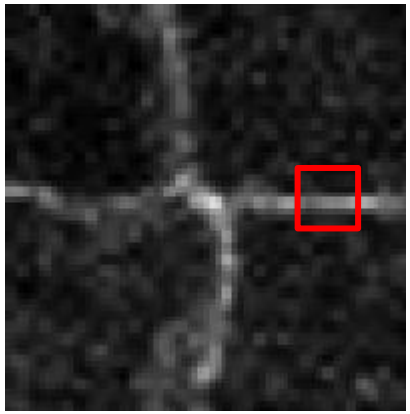


Let‘s get into it...

Digital Images

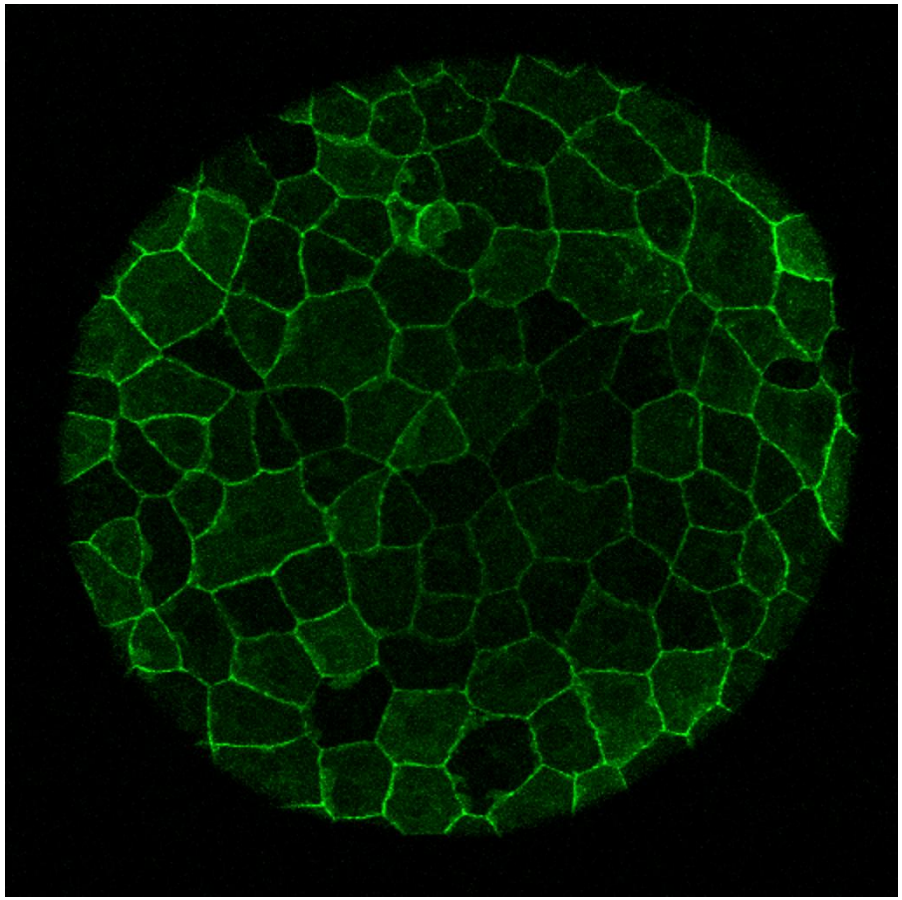
- **Digital images are arrays of numbers**





Digital Images

- **Digital images are arrays of numbers**

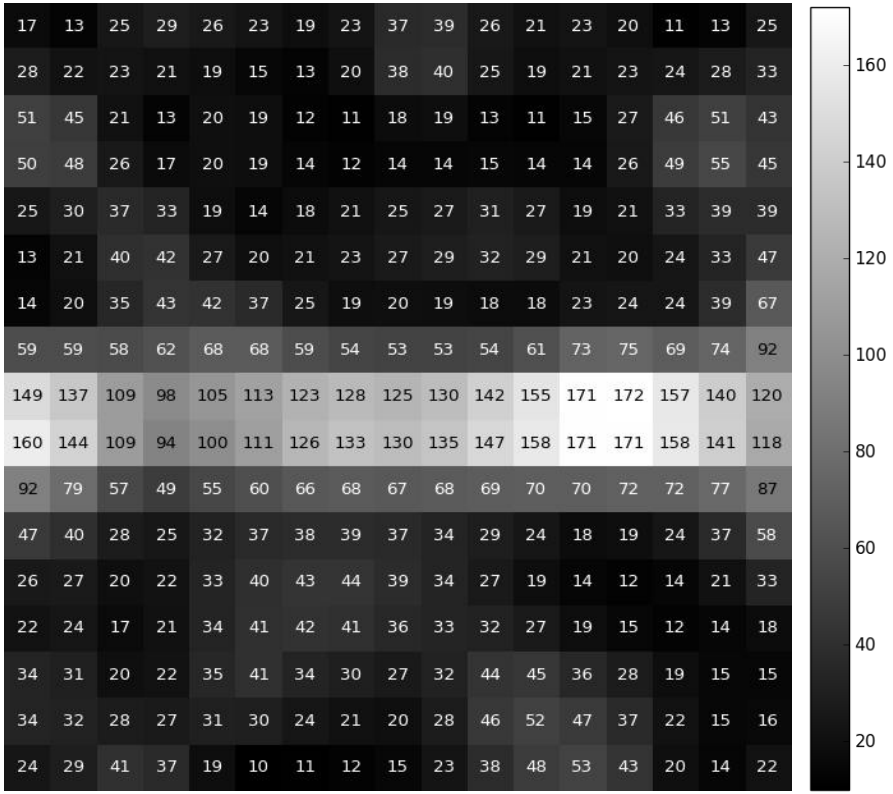


17	13	25	29	26	23	19	23	37	39	26	21	23	20	11	13	25
28	22	23	21	19	15	13	20	38	40	25	19	21	23	24	28	33
51	45	21	13	20	19	12	11	18	19	13	11	15	27	46	51	43
50	48	26	17	20	19	14	12	14	14	15	14	14	26	49	55	45
25	30	37	33	19	14	18	21	25	27	31	27	19	21	33	39	39
13	21	40	42	27	20	21	23	27	29	32	29	21	20	24	33	47
14	20	35	43	42	37	25	19	20	19	18	18	23	24	24	39	67
59	59	58	62	68	68	59	54	53	53	54	61	73	75	69	74	92
149	137	109	98	105	113	123	128	125	130	142	155	171	172	157	140	120
160	144	109	94	100	111	126	133	130	135	147	158	171	171	158	141	118
92	79	57	49	55	60	66	68	67	68	69	70	70	72	72	77	87
47	40	28	25	32	37	38	39	37	34	29	24	18	19	24	37	58
26	27	20	22	33	40	43	44	39	34	27	19	14	12	14	21	33
22	24	17	21	34	41	42	41	36	33	32	27	19	15	12	14	18
34	31	20	22	35	41	34	30	27	32	44	45	36	28	19	15	15
34	32	28	27	31	30	24	21	20	28	46	52	47	37	22	15	16
24	29	41	37	19	10	11	12	15	23	38	48	53	43	20	14	22

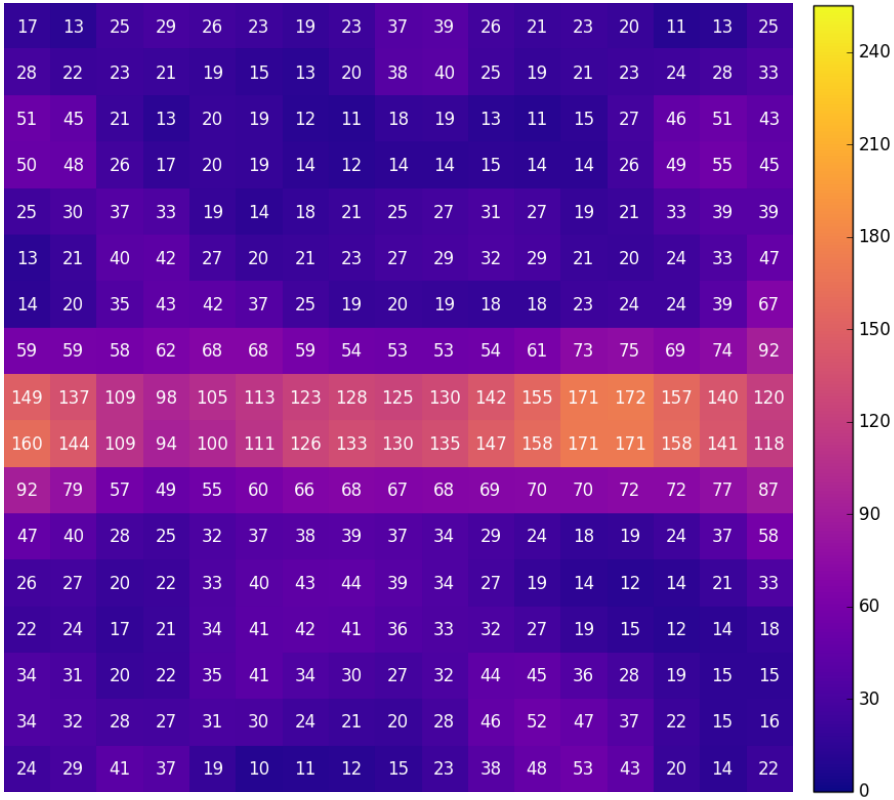
LUTs

- The link between an ‘images’ and ‘numbers’ are Look-Up Tables (LUTs)

‘GRAY’ LUT

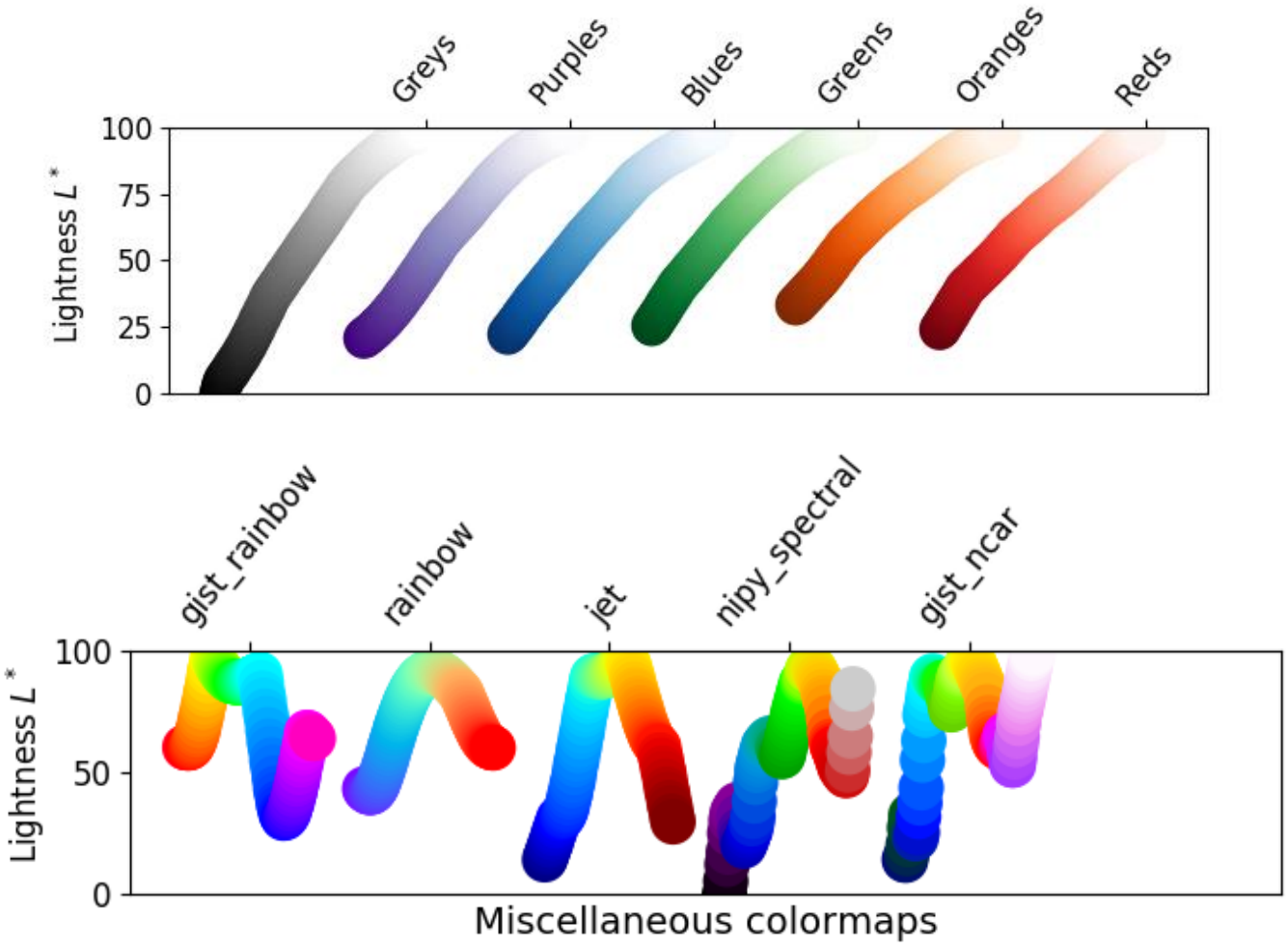


‘PLASMA’ LUT



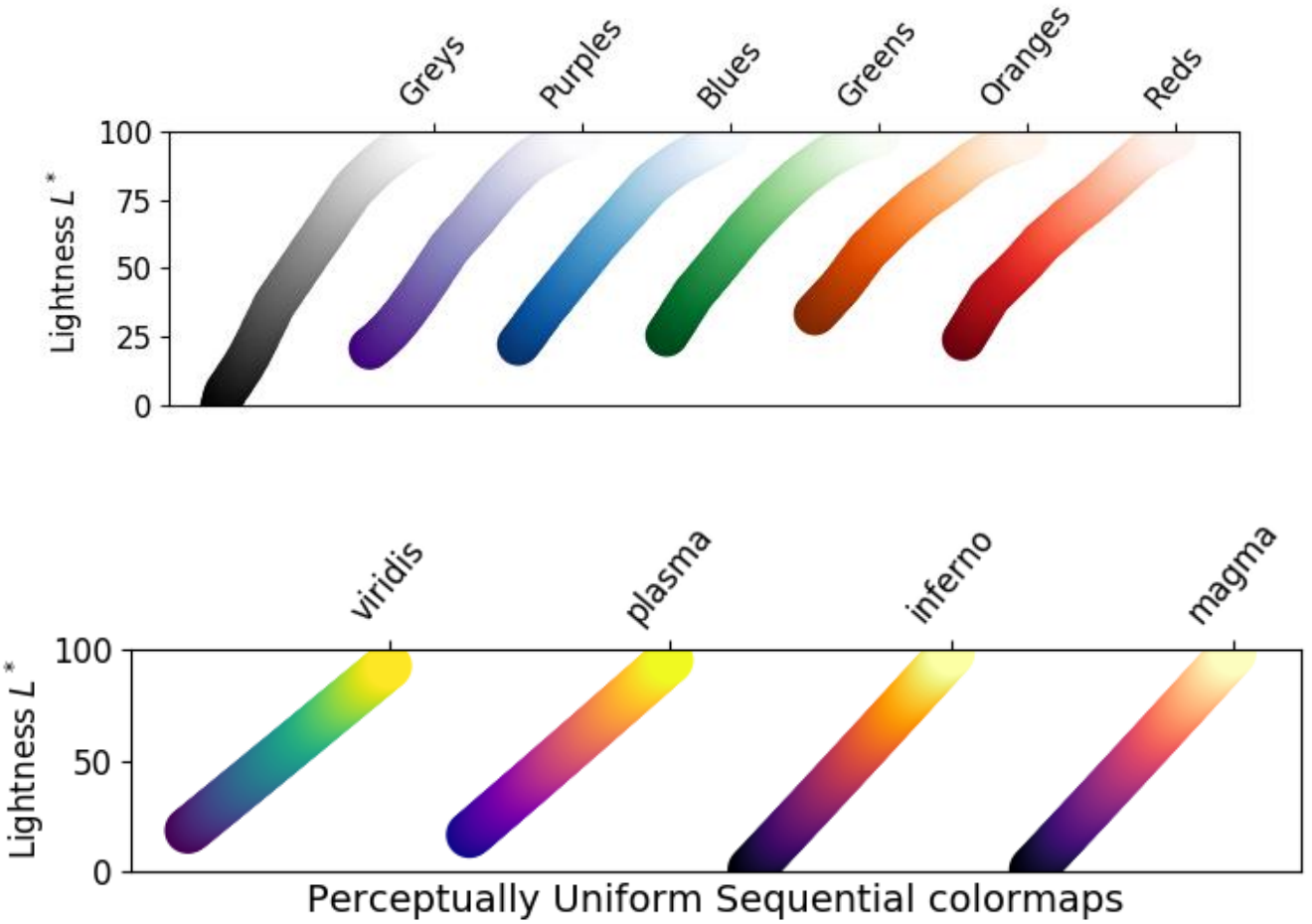
LUTs

- **Side note: LUTs can trick the eyes…**



LUTs

- **Side note: LUTs can trick the eyes…**



Dimensions

- **Multi-channel images, stacks and time courses are still just arrays…**

  …but with more dimensions

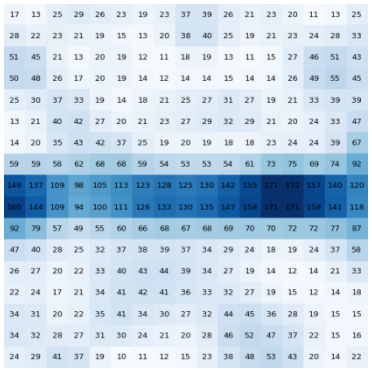
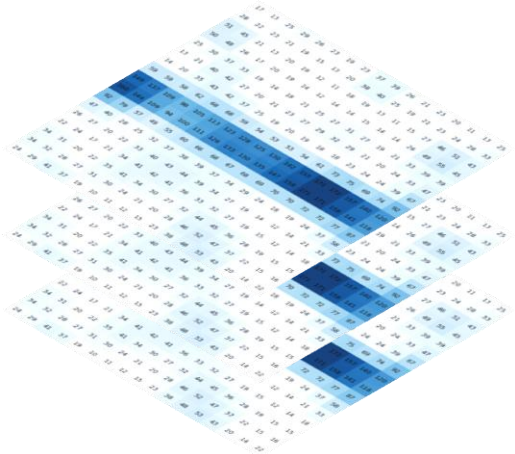
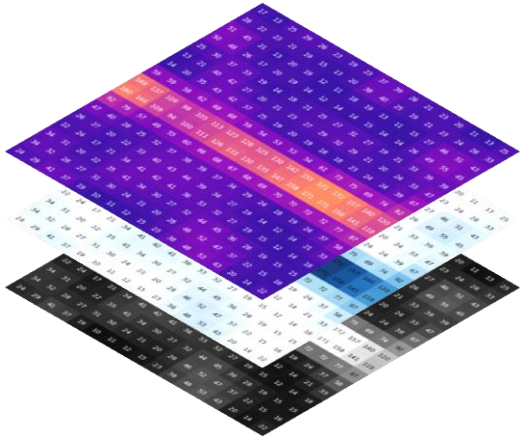


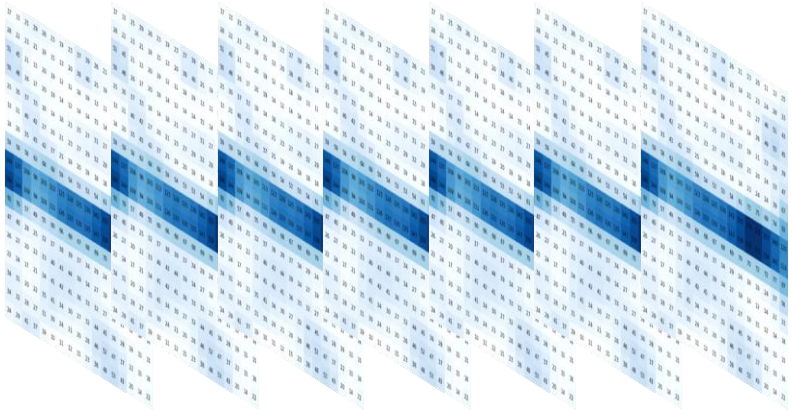
Image (y,x)



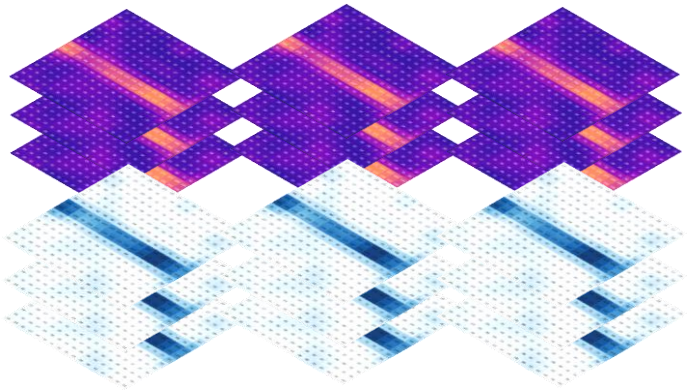
Stack (z,y,x)



Channels (c,y,x)



Time Series (t,y,x)



(t,c,z,y,x)

Data Types

- **Digital representations of numbers have some peculiarities…**
- **Most common number types**

int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa

Used in image processing

Produced by microscope

Data Types

- **What could possibly go wrong?**

A (np.uint8) **B** (np.uint8) **A – B = C** (np.uint8)

-32	77	-32	-89	55
-50	-62	-134	-16	-153
21	-46	113	34	-49
-34	161	-209	14	20
-20	-227	3	34	-165

A.astype(np.int16) – B.astype(np.int16) = C (np.int16)

Data Types

- **What could possibly go wrong?**

204	207	141	69	117
20	139	24	148	36
112	176	164	94	174
122	175	16	77	164
26	5	42	103	45

A (np.uint8)

/

236	130	173	158	62
70	201	158	164	189
91	222	51	60	223
156	14	225	63	144
46	232	39	69	210

B (np.uint8)

=

0	1	0	0	1
0	0	0	0	0
1	0	3	1	0
0	12	0	1	1
0	0	1	1	0

A / B = C (np.uint8)

This happens automatically in py3!

In py2, do `from __future__ import division`!

0.86	1.59	0.82	0.44	1.89
0.29	0.69	0.15	0.90	0.19
1.23	0.79	3.22	1.57	0.78
0.78	12.50	0.07	1.22	1.14
0.57	0.02	1.08	1.49	0.21

A.astype(np.float32) – B.astype(np.float32) = C (np.float32)

Data Types

- **What could possibly go wrong?**

673	704	862	960	795
142	614	806	638	704
988	467	2	505	219
802	86	283	847	624
806	155	592	57	339

A (np.uint16)

=

161	192	94	192	27
142	102	38	126	192
220	211	2	249	219
34	86	27	79	112
38	155	80	57	83

A.astype(np.uint8) = C (np.uint8)

Rescales an array between 0 and x:

$$\frac{A - min(A)}{max(A) - min(A)} \cdot x = C$$

173	181	222	247	205
36	158	207	164	181
255	120	0	130	56
206	21	72	218	160
207	39	152	14	87

(((A - A.min()) / (A.max() - A.min()))*255).astype(np.uint8) = C (np.uint8)

Data Types

- **What could possibly go wrong?**

673	704	862	960	795
142	614	806	638	704
988	467	2	505	219
802	86	283	847	624
806	155	592	57	339

A (np.uint16)

=

161	192	94	192	27
142	102	38	126	192
220	211	2	249	219
34	86	27	79	112
38	155	80	57	83

A.astype(np.uint8) = C (np.uint8)

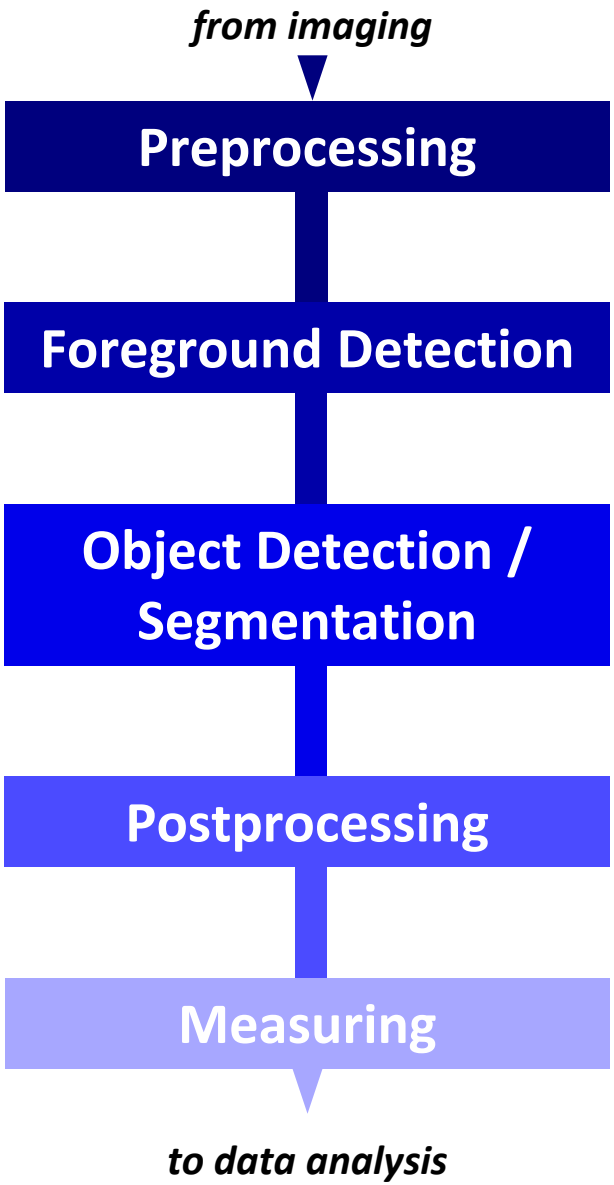
Warning 1: Rescaling is lossy!

Warning 2: Rescaling prevents absolute comparisons!
Careful when processing multiple images!

173	181	222	247	205
36	158	207	164	181
255	120	0	130	56
206	21	72	218	160
207	39	152	14	87

(((A - A.min()) / (A.max() - A.min()))*255).astype(np.uint8) = C (np.uint8)

Standard Pipeline Overview

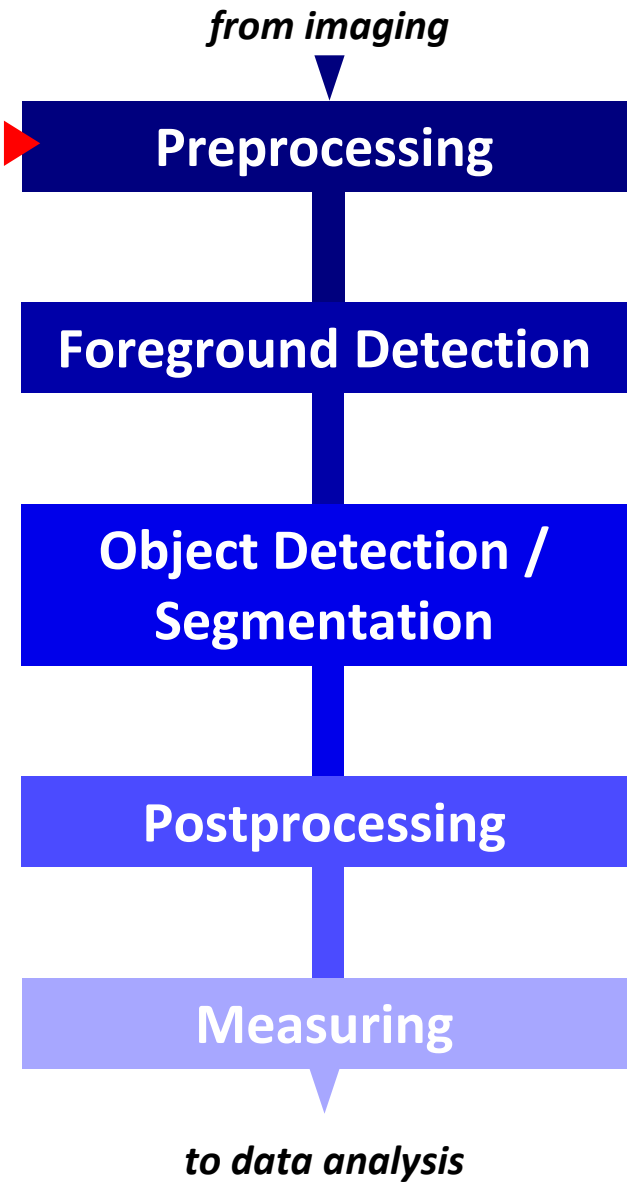


Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

Standard Pipeline Overview



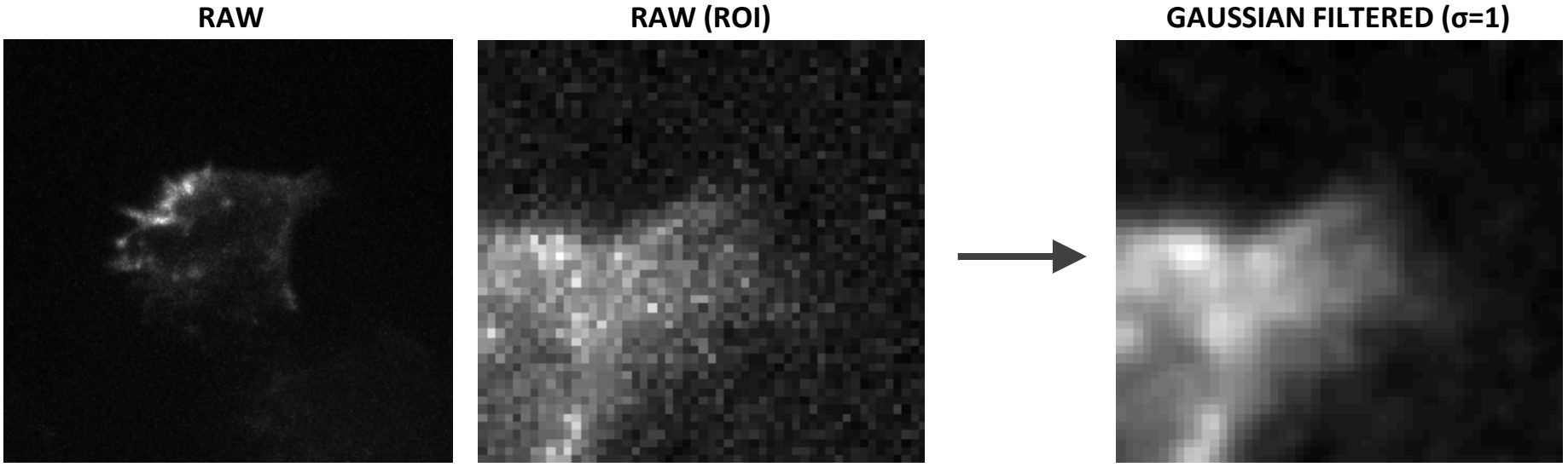
Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

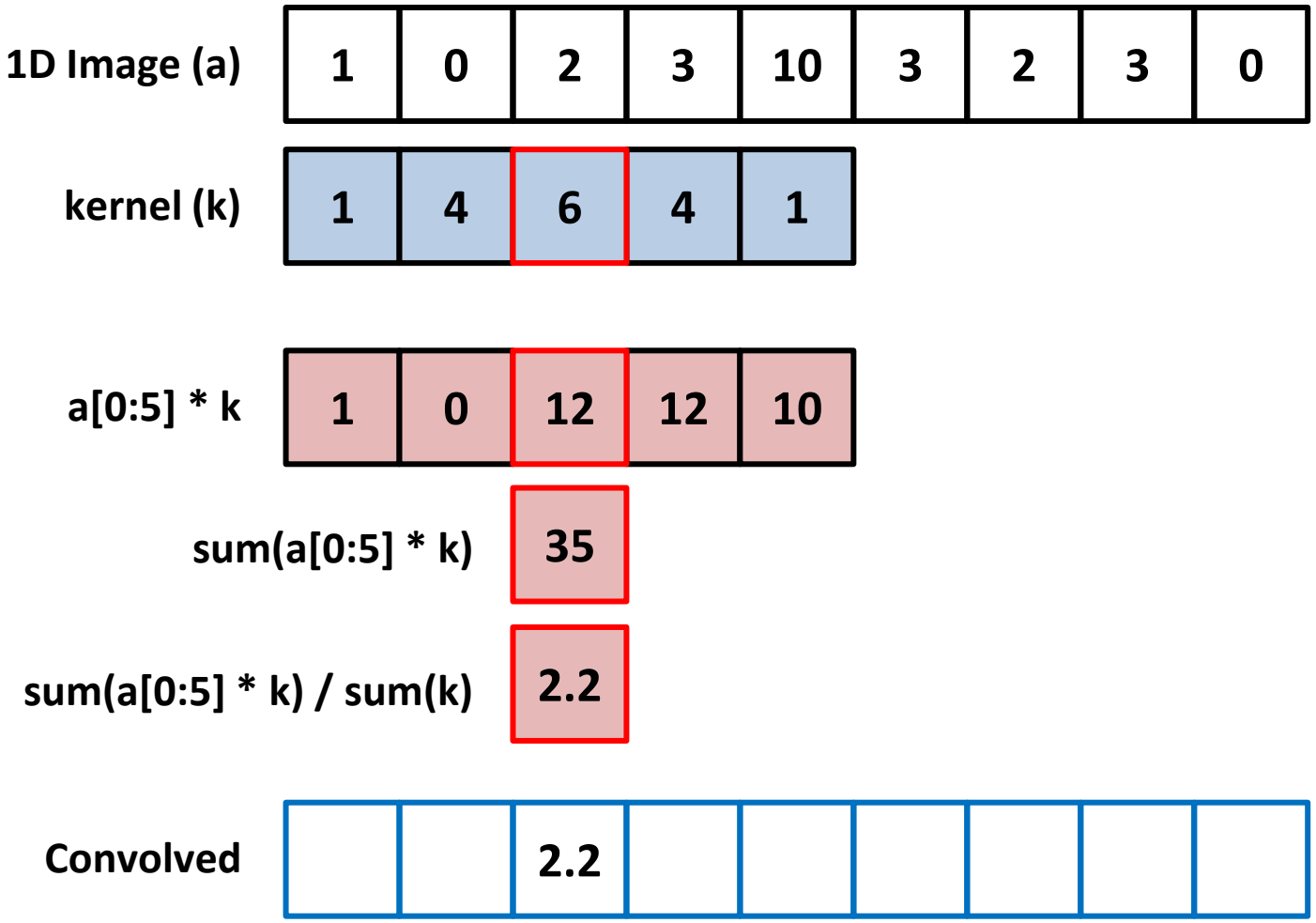
Preprocessing: Filtering

- Goal: removing noise but preserving structure
- Common filters
  - Gaussian filter (smoothing, general noise reduction)
  - Median filter (removing noise)
  - LoG filter (dots), Sobel filter (edges)
- Example: Gaussian filter



Preprocessing: Filtering

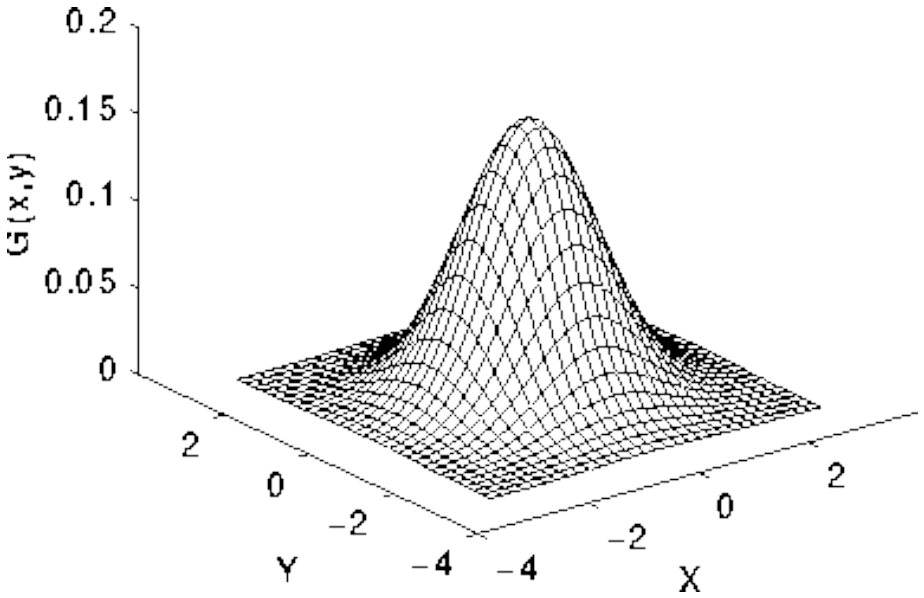
- How it works: kernels & convolution



Preprocessing: Filtering

- **How it works: kernels & convolution**

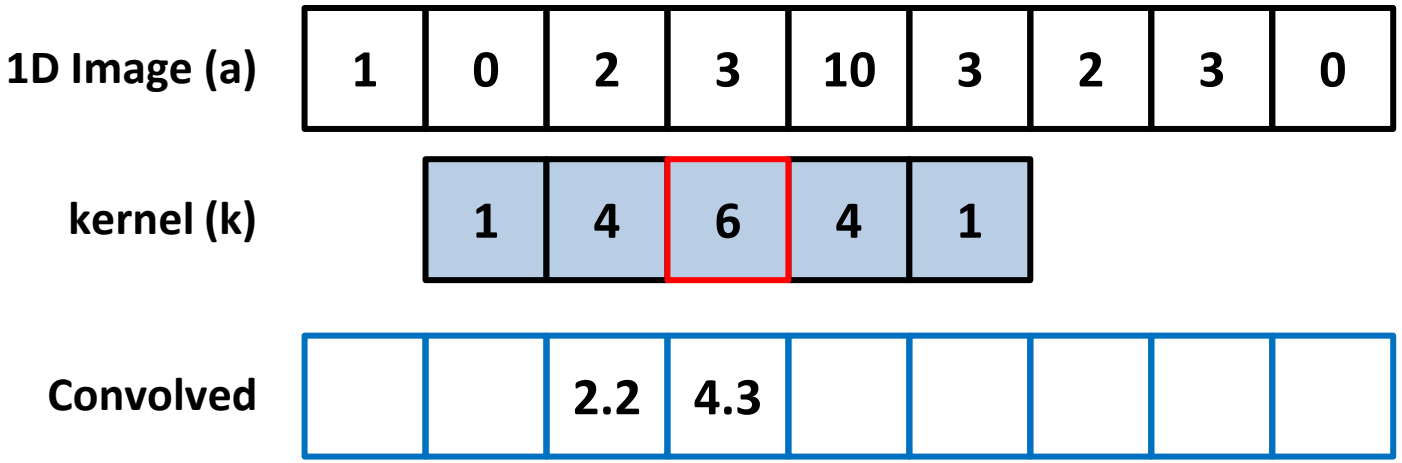
2D Gaussian kernel



0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	10	10	10	0	0	0	0
0	0	10	10	20	30	20	10	10	0	0
0	0	10	30	50	50	50	30	10	0	0
0	10	20	50	70	80	70	50	20	10	0
0	10	30	60	80	90	80	60	30	10	0
0	10	20	50	70	80	70	50	20	10	0
0	0	10	30	50	50	50	30	10	0	0
0	0	10	10	20	30	20	10	10	0	0
0	0	0	0	10	10	10	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

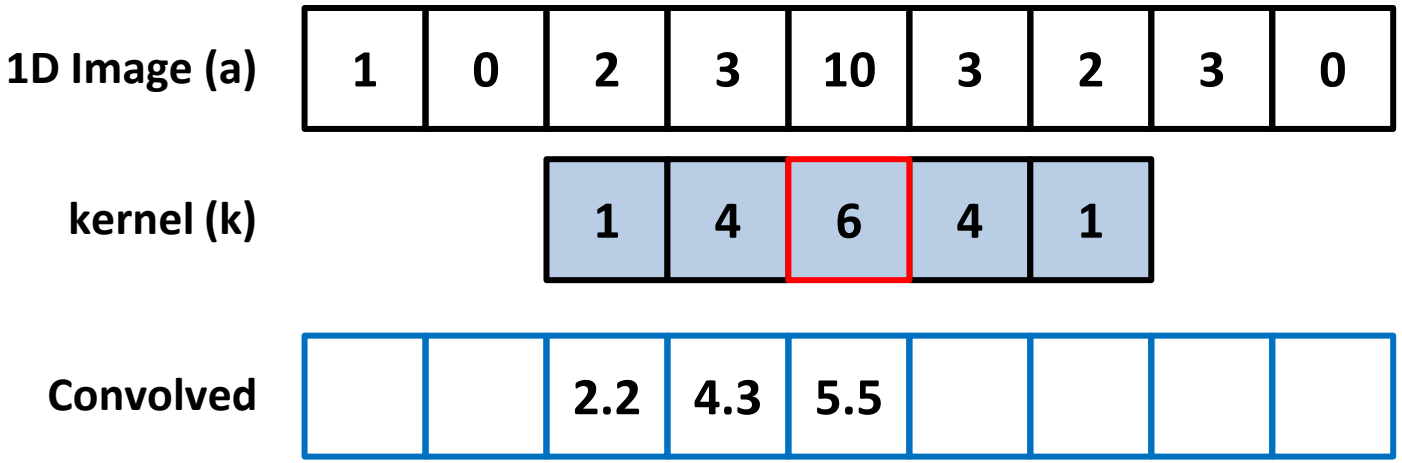
Preprocessing: Filtering

- How it works: kernels & convolution



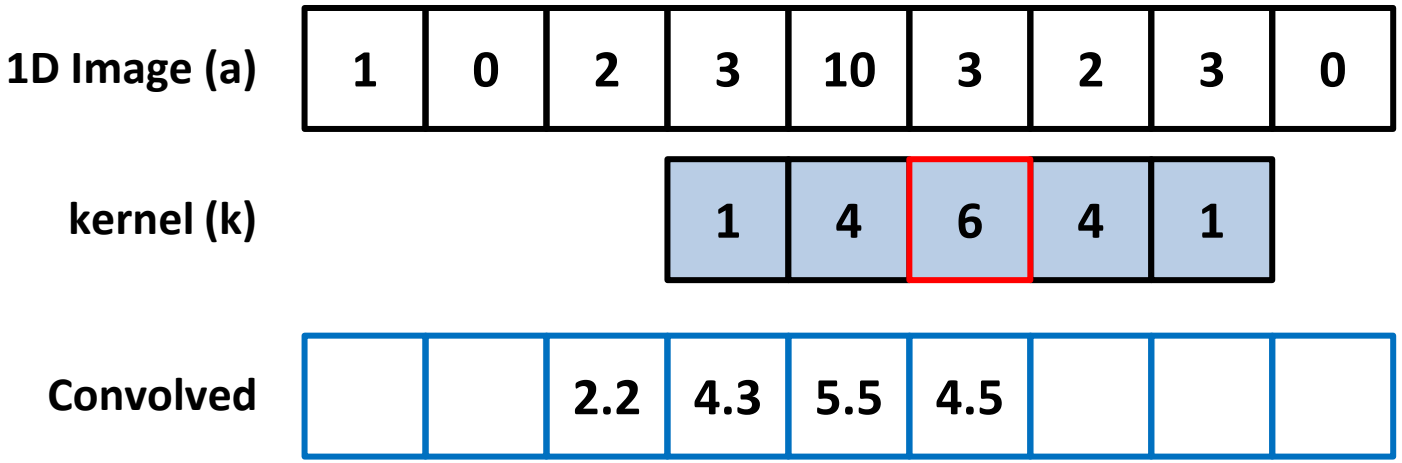
Preprocessing: Filtering

- How it works: kernels & convolution



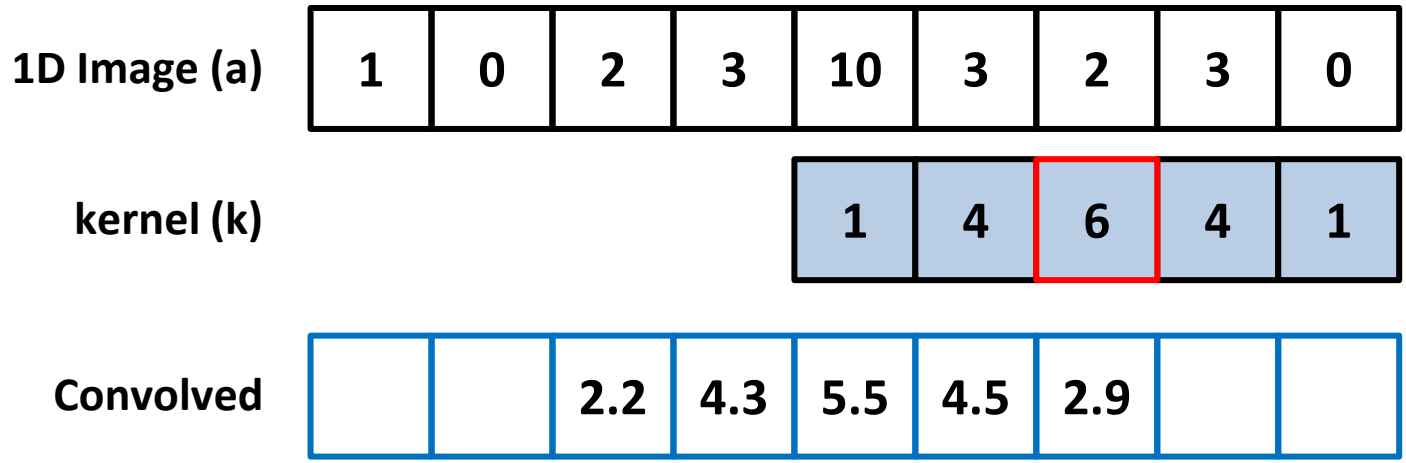
Preprocessing: Filtering

- How it works: kernels & convolution



Preprocessing: Filtering

- How it works: kernels & convolution

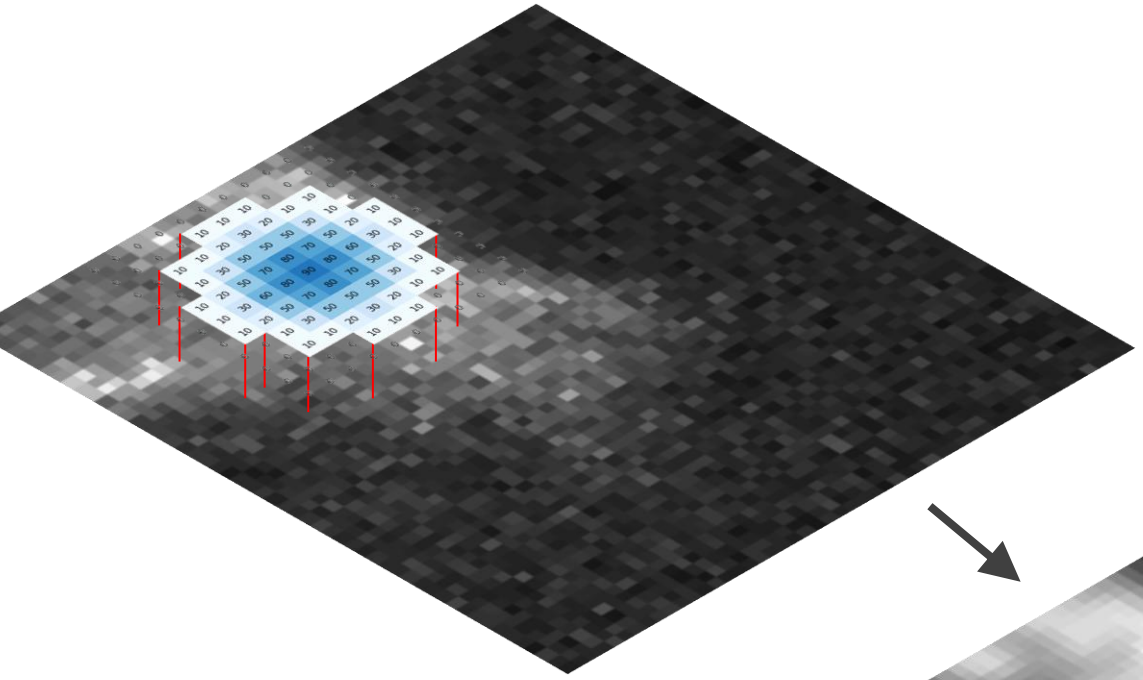


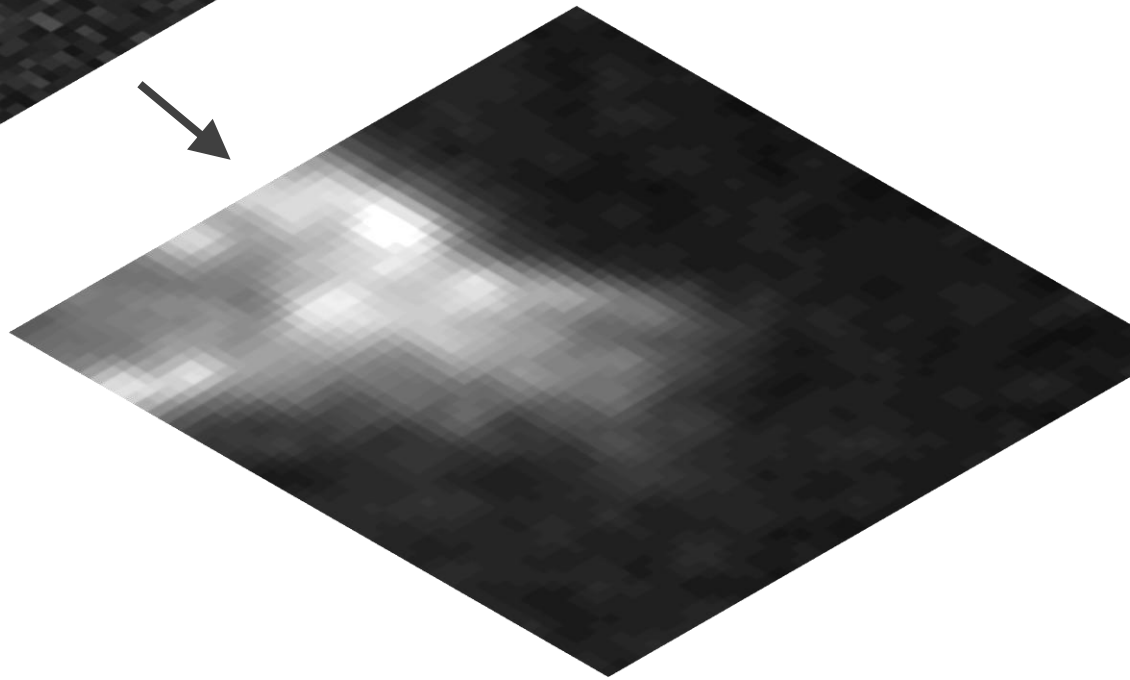
Note: Behavior at edges is undefined. Default in *scipy* is `reflect`.

Preprocessing: Filtering

- How it works: kernels & convolution

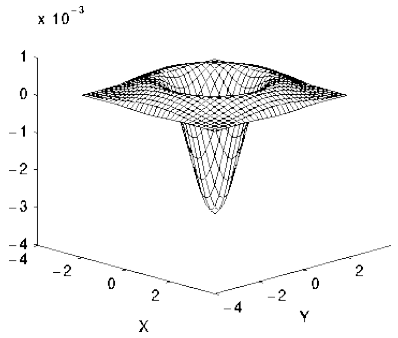
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	10	10	10	0	0	0	0
0	0	10	10	20	30	20	10	10	0	0
0	0	10	30	50	50	50	30	10	0	0
0	10	20	50	70	80	70	50	20	10	0
0	10	30	60	80	90	80	60	30	10	0
0	10	20	50	70	80	70	50	20	10	0
0	0	10	30	50	50	50	30	10	0	0
0	0	10	10	20	30	20	10	10	0	0
0	0	0	0	10	10	10	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0





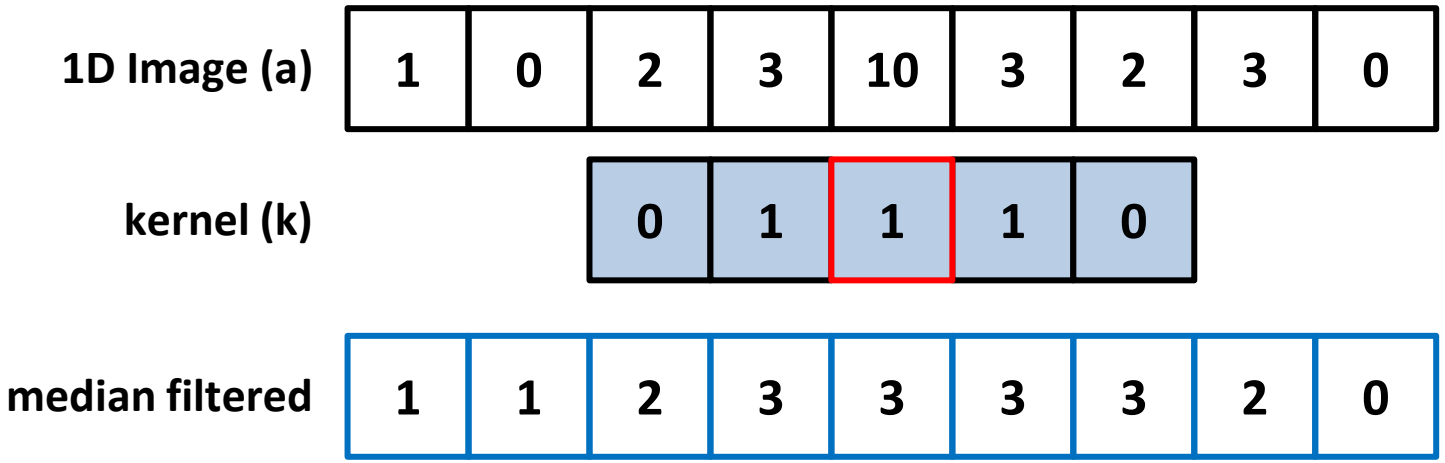
Preprocessing: Filtering

- Other filter have different kernels
  - e.g. LoG filter



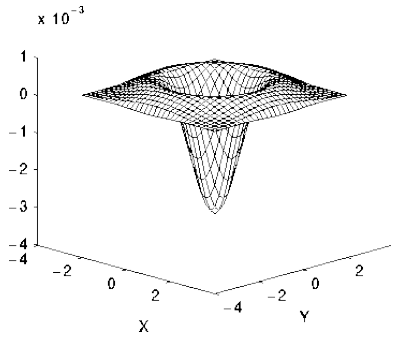
0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

- Or they perform different operations
  - e.g. median filter



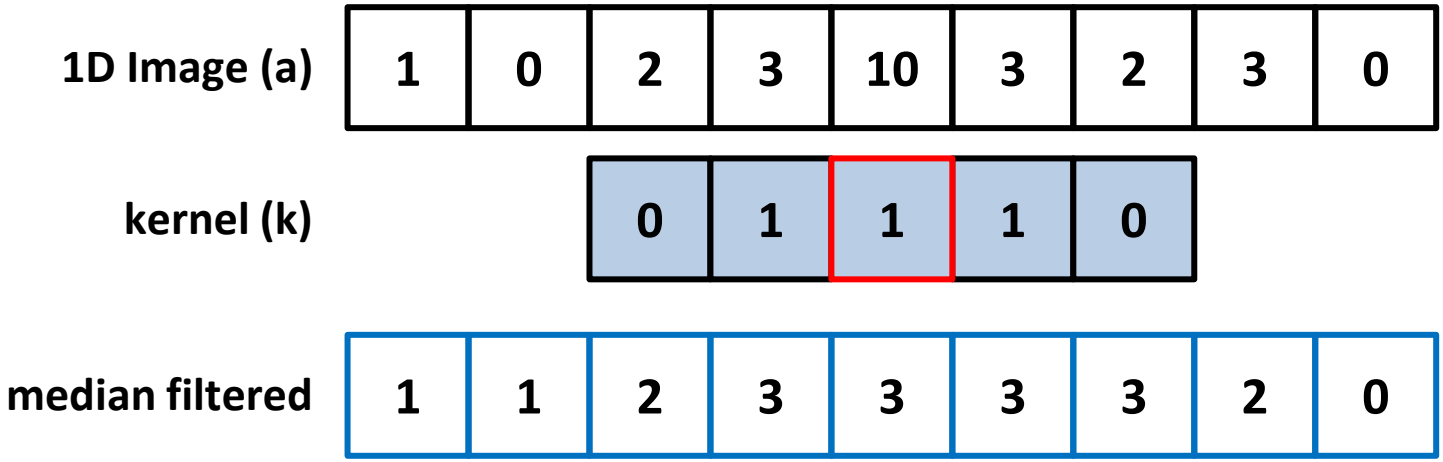
Preprocessing: Filtering

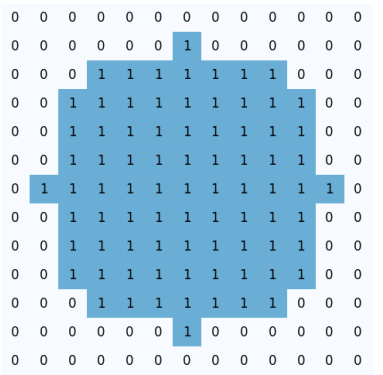
- Other filter have different kernels
  - e.g. LoG filter



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

- Or they perform different operations
  - e.g. median filter





Preprocessing: Background Subtraction

- **Goal: removing background signal**
  - **Enhancing vs. non-enhancing**
  - **Manual vs. automated**
  - **Uniform vs. adaptive**

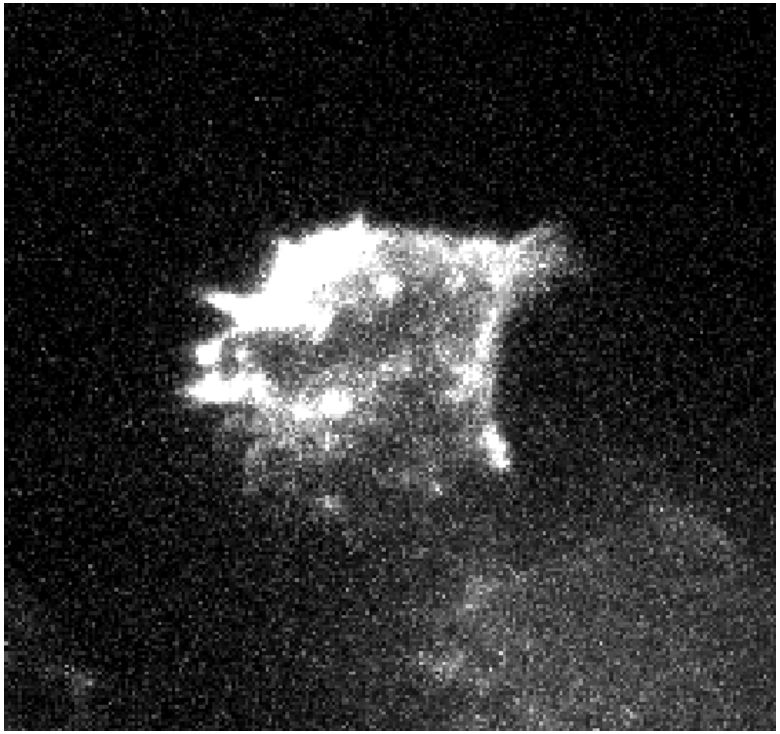
Preprocessing: Background Subtraction

- Goal: removing background signal
  - Enhancing vs. non-enhancing
  - Manual vs. automated
  - Uniform vs. adaptive

RAW

Subtract mean

BGSUB

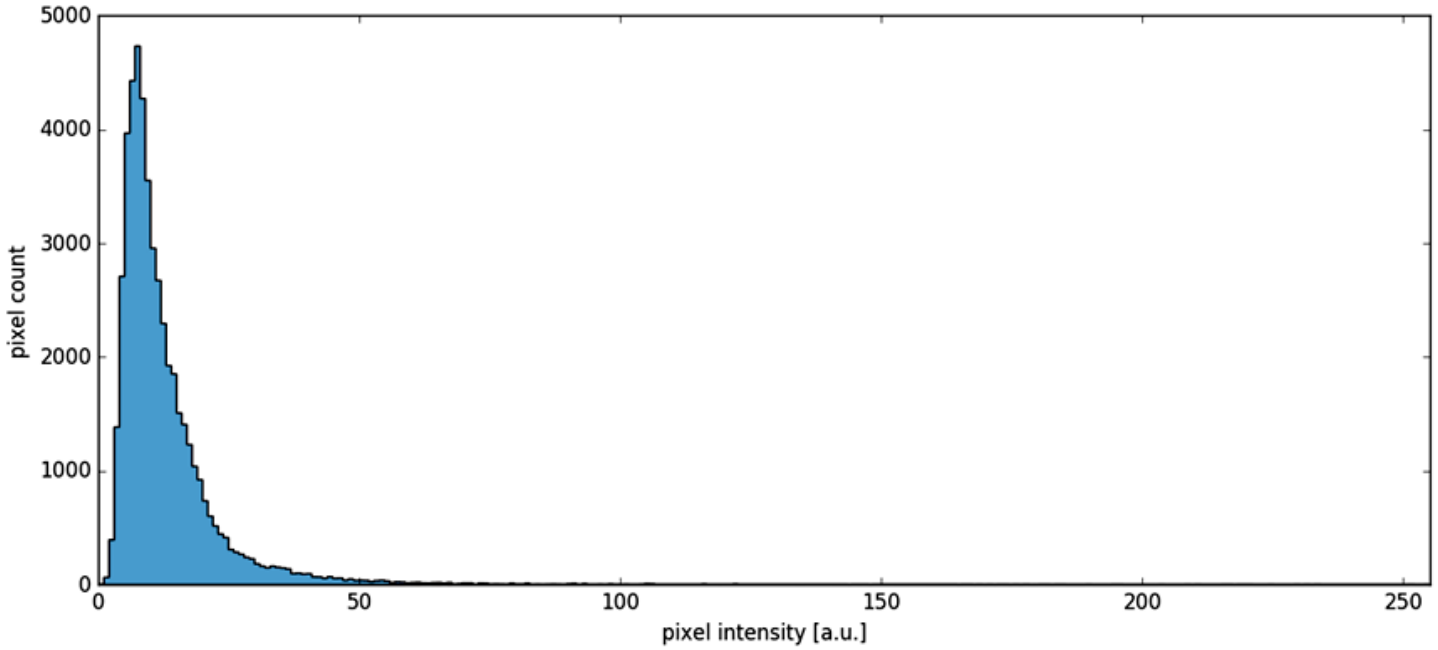


Preprocessing: Background Subtraction

- Goal: removing background signal
  - Enhancing vs. non-enhancing
  - Manual vs. automated
  - Uniform vs. adaptive

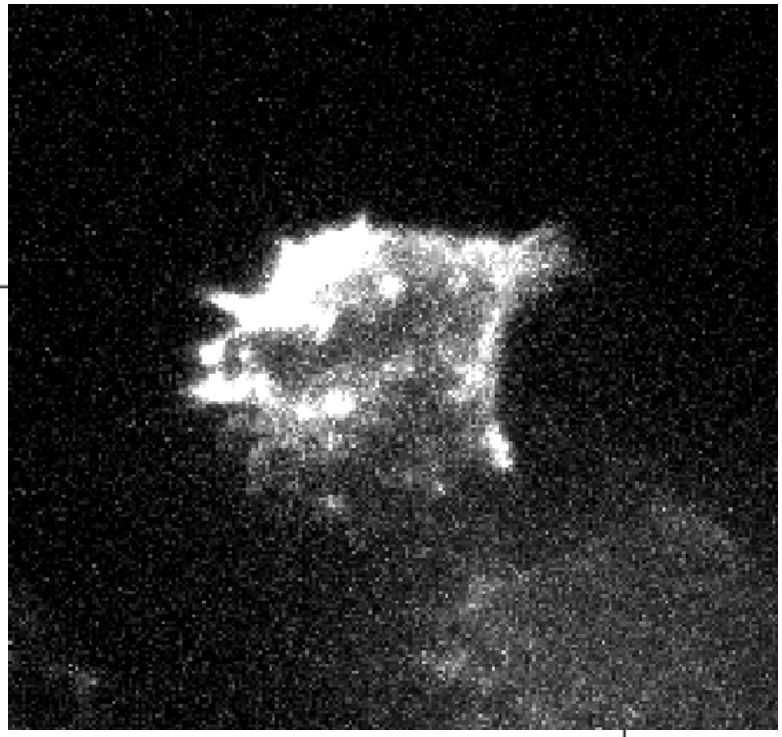
Preprocessing: Background Subtraction

- **Goal: removing background signal**
  - **Enhancing vs. non-enhancing**
  - **Manual vs. automated**
  - **Uniform vs. adaptive**



Preprocessing: Background Subtraction

- **Goal: removing background signal**
  - **Enhancing vs. non-enhancing**
  - **Manual vs. automated**
  - **Uniform vs. adaptive**

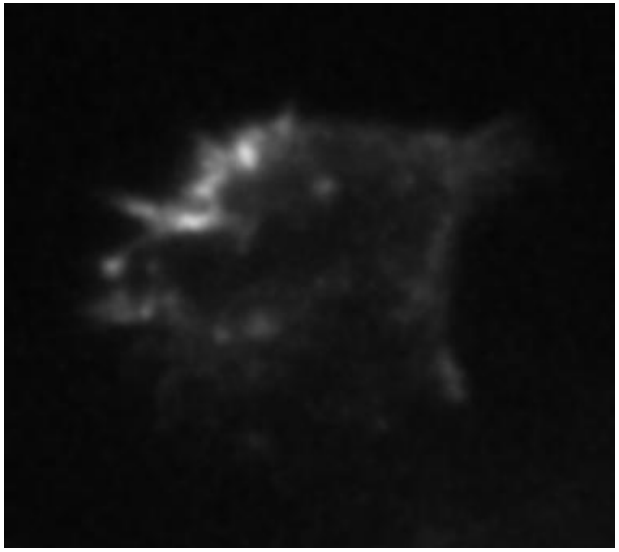


argmax(hist) = 7

Preprocessing: Background Subtraction

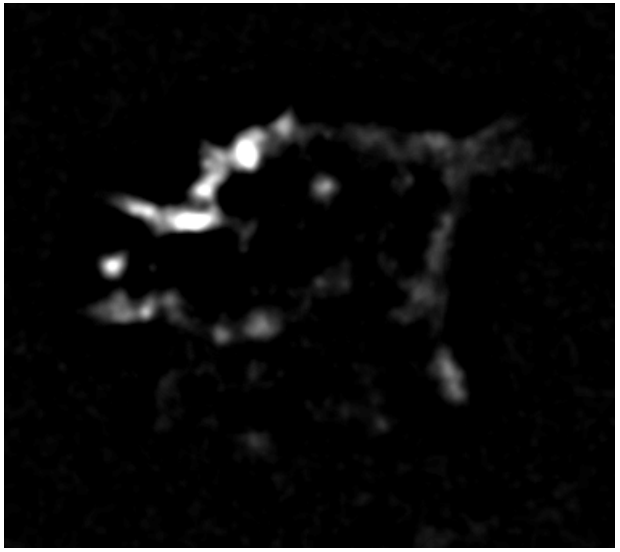
- Goal: removing background signal
  - Enhancing vs. non-enhancing
  - Manual vs. automated
  - Uniform vs. adaptive

a = gaussian_filter(raw,2)

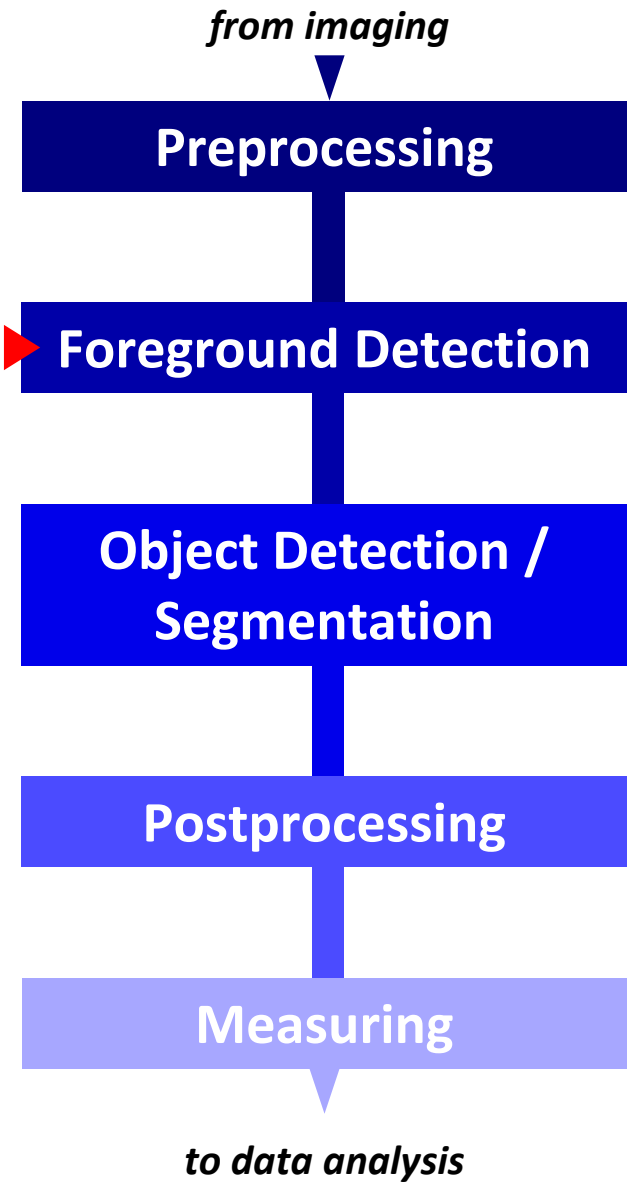


bg = mean_filter(a,30)

bgsub = a - bg



Standard Pipeline Overview



Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

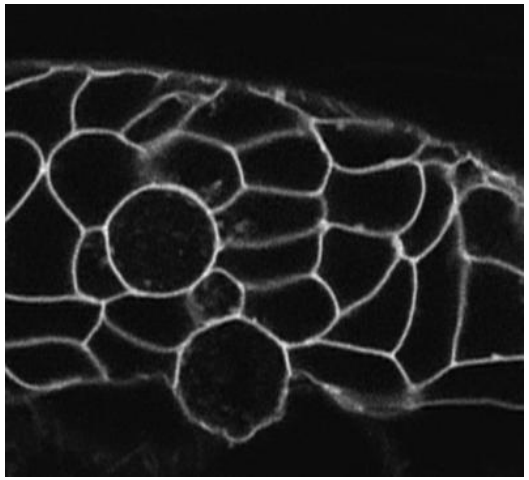
+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

Foreground Detection: Thresholding

- **Goal: binarizing the image into foreground and background**
  - **Manual vs. automated**
  - **Uniform vs. adaptive**
- **How to find the threshold?**
  - **Manually test different options…**
  - **Histogram-based approaches (e.g. Otsu's method)**
  - **Optimization-based approaches (e.g. object-count thresholding)**

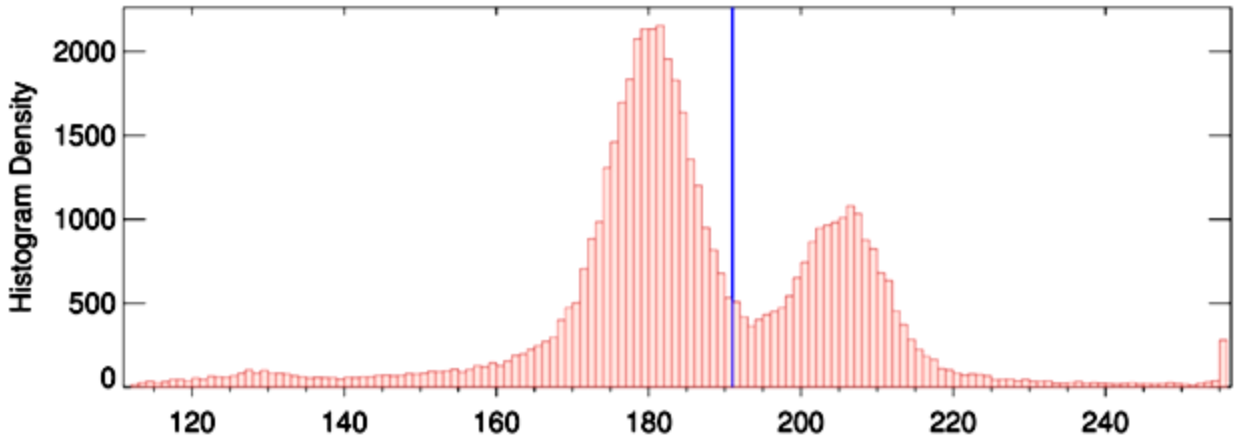
RAW



THRESHOLDED

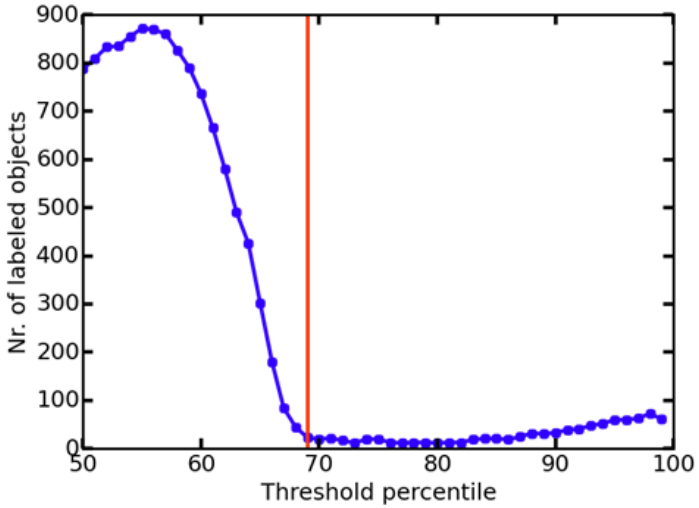
Foreground Detection: Thresholding

- Goal: binarizing the image into foreground and background
  - Manual vs. automated
  - Uniform vs. adaptive
- How to find the threshold?
  - Manually test different options…
  - Histogram-based approaches (e.g. Otsu's method)
  - Optimization-based approaches (e.g. object-count thresholding)



Foreground Detection: Thresholding

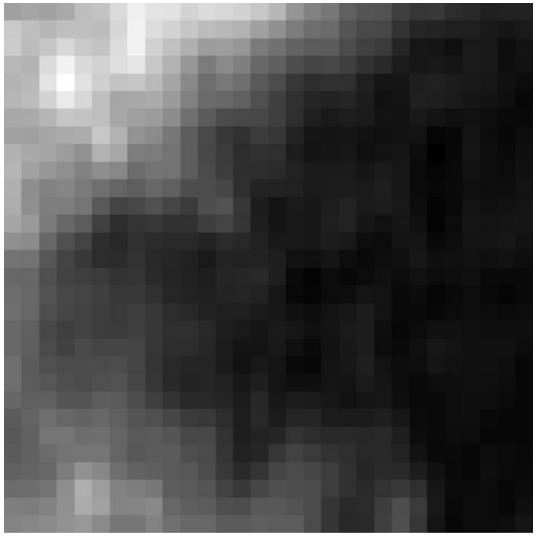
- Goal: binarizing the image into foreground and background
  - Manual vs. automated
  - Uniform vs. adaptive
- How to find the threshold?
  - Manually test different options…
  - Histogram-based approaches (e.g. Otsu's method)
  - Optimization-based approaches (e.g. object-count thresholding)



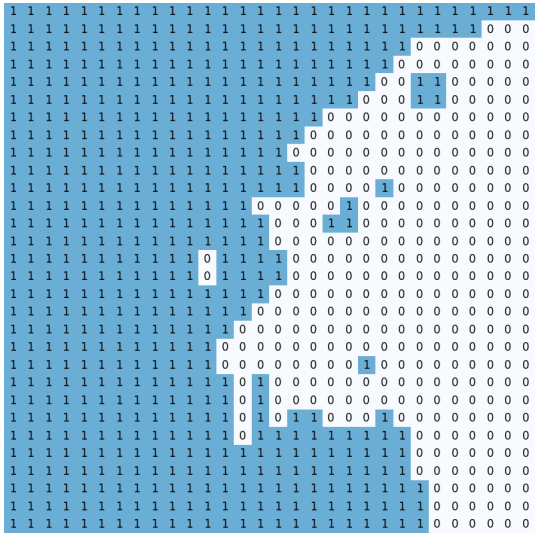
Foreground Detection: Thresholding

- Thresholded images are Boolean arrays (aka '**masks**')
- They can be used to operate specifically on the masked images in other arrays

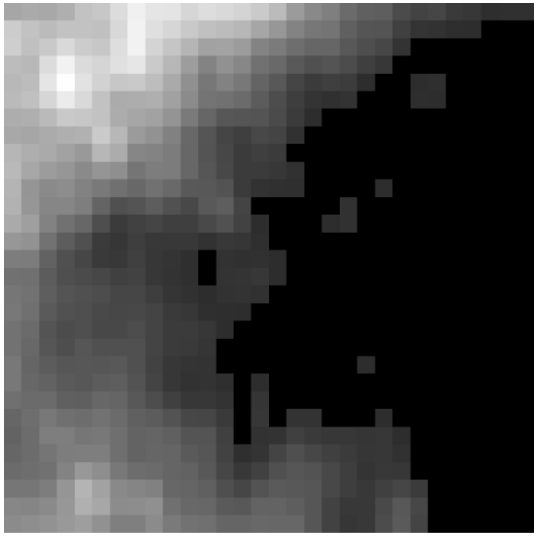
RAW



THRESHOLDED



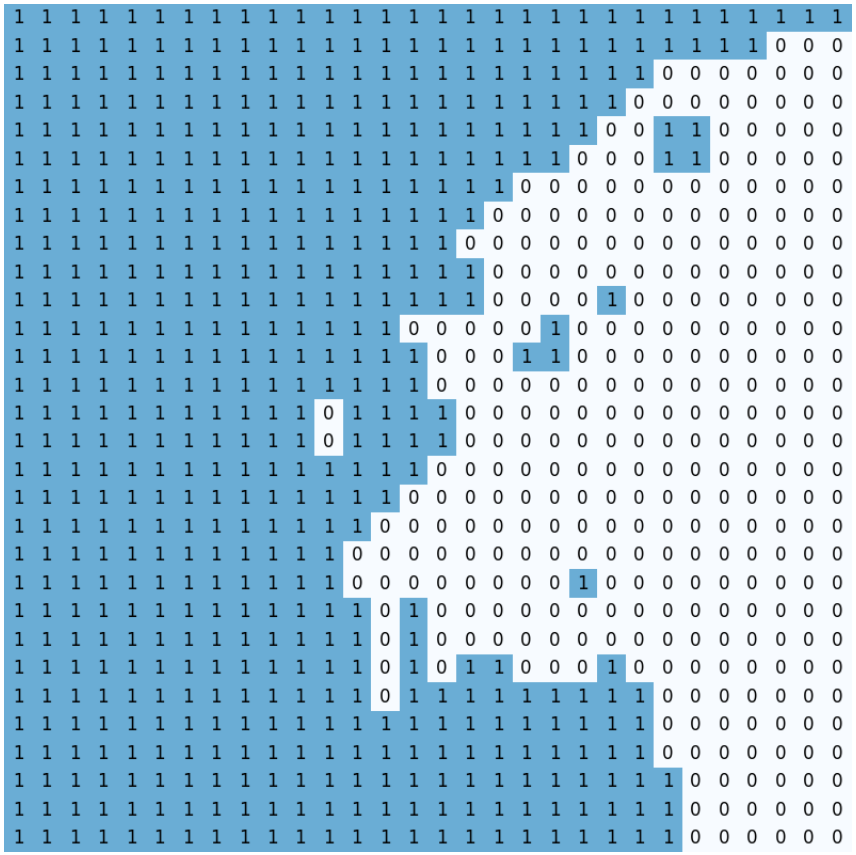
MASKED



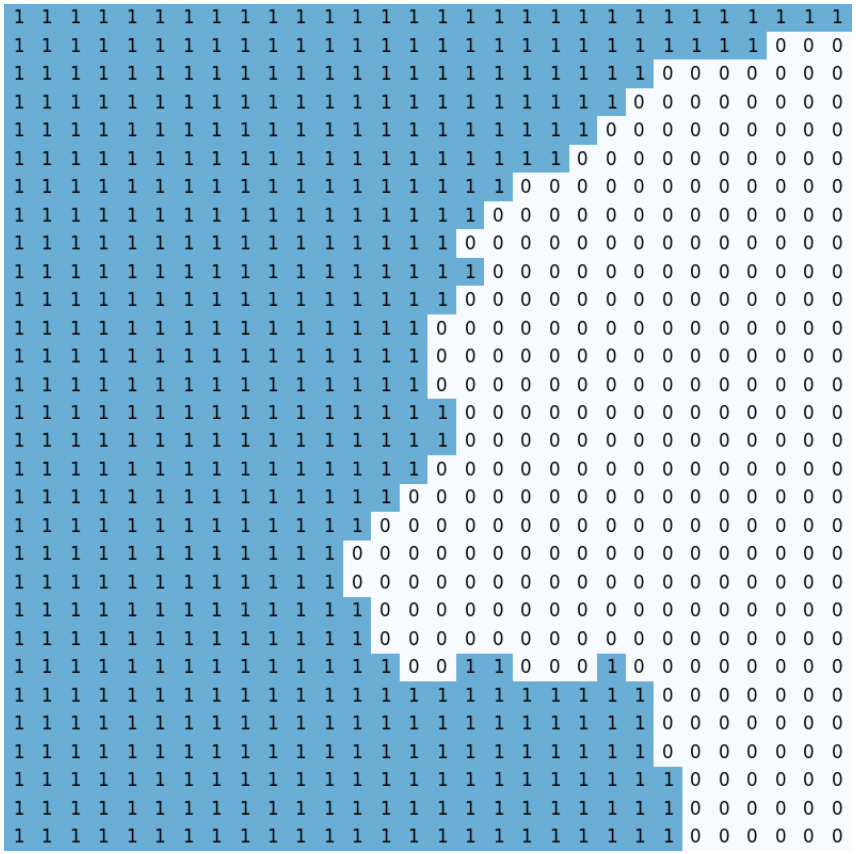
Foreground Detection: Morphological Operations

- **Goal: operating on binary masks (to improve/correct them)**

THRESHOLDED

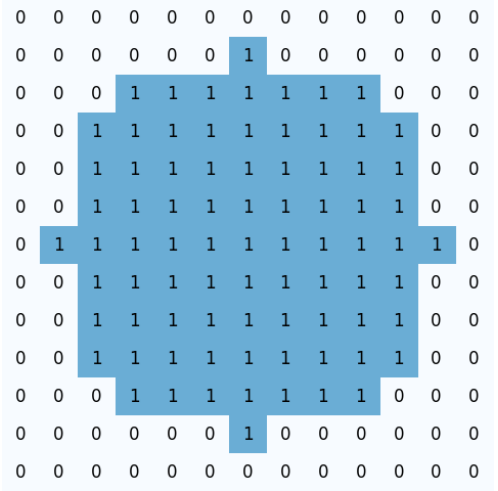


MORPHOLOGICALLY PROCESSED



Foreground Detection: Morphological Operations

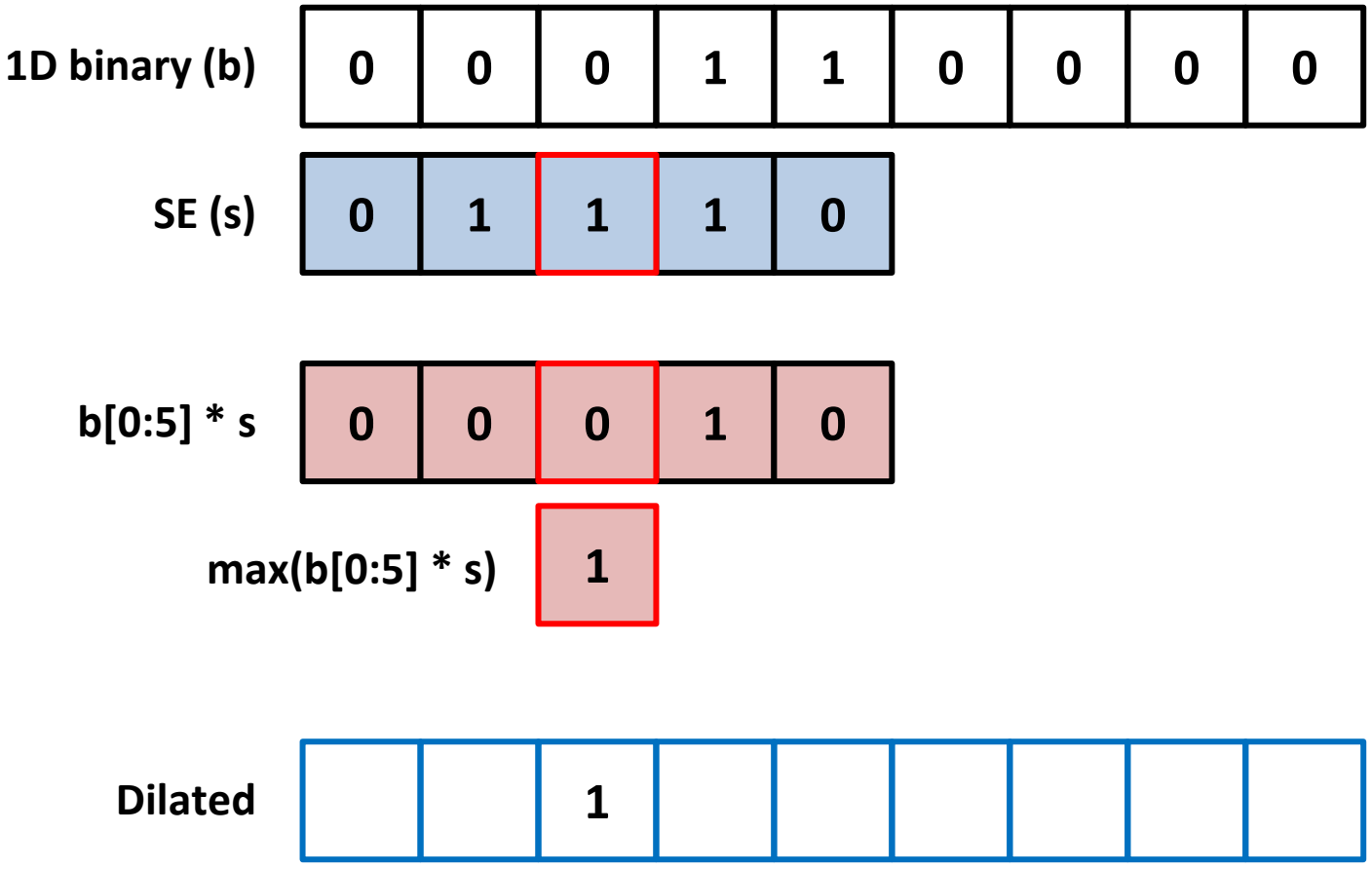
- Goal: operating on binary masks (to improve/correct them)
- Common morphological operations
  - Erosion & Dilation
  - Opening & Closing
  - Hole filling
- Principle very much the same as in filtering
  - Use of a `structural element` (SE); basically the same as a kernel



DISC-SHAPED SE

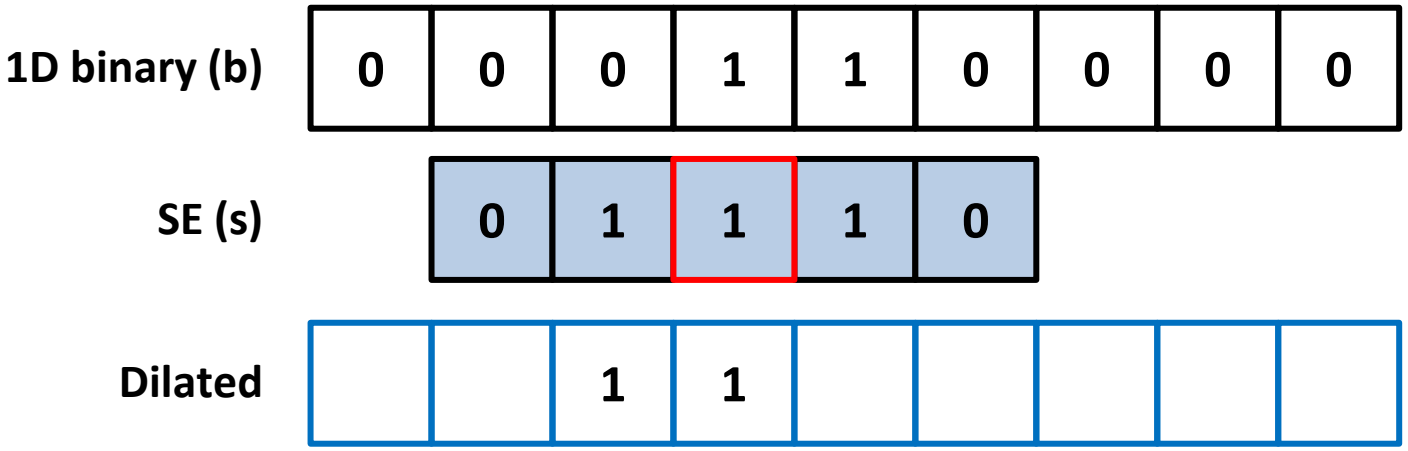
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**



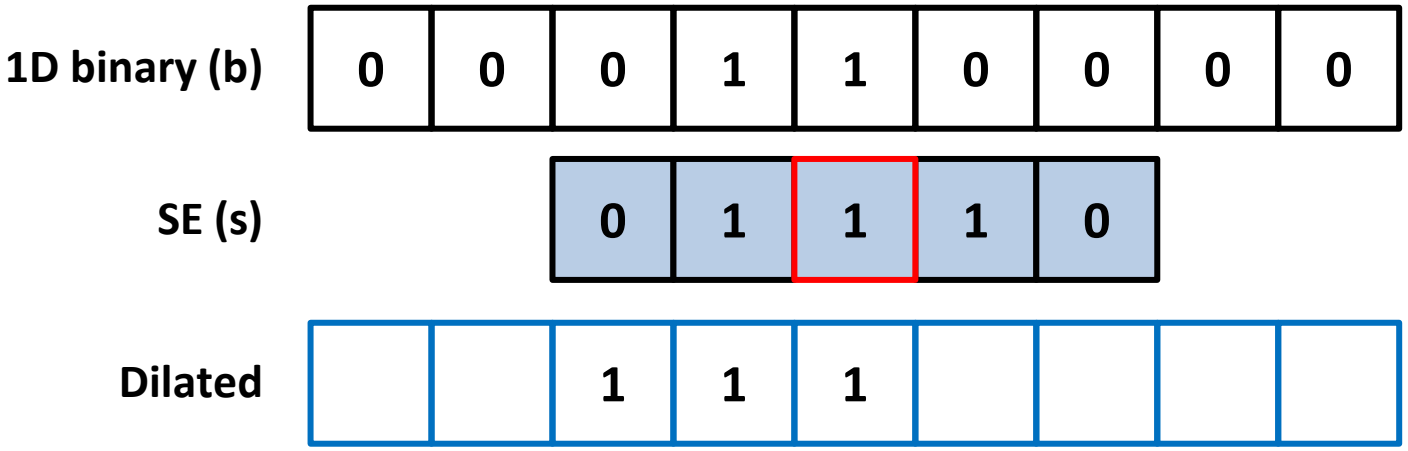
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**



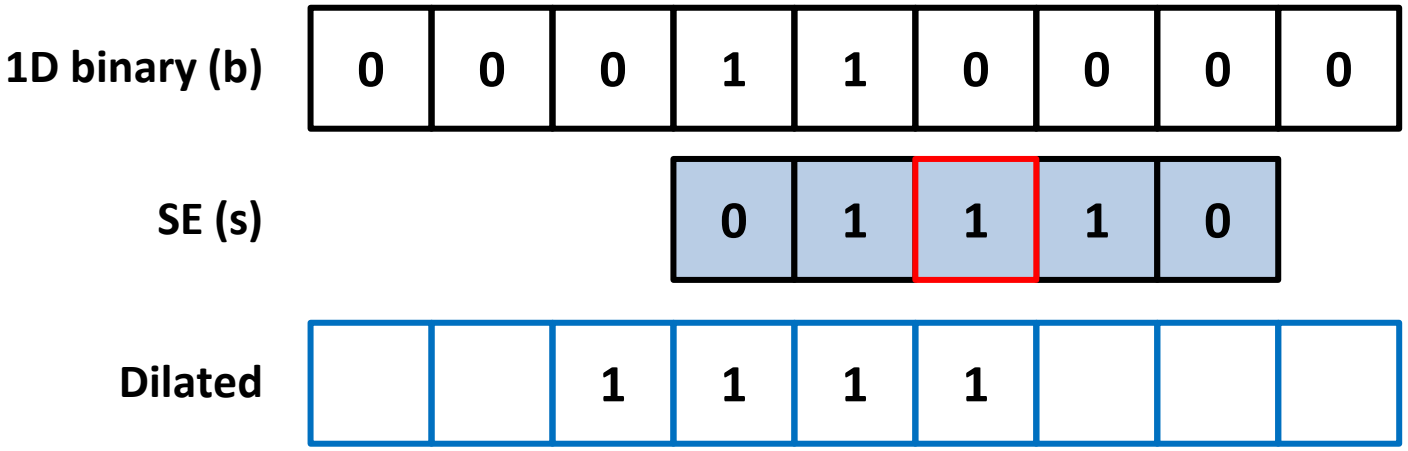
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**



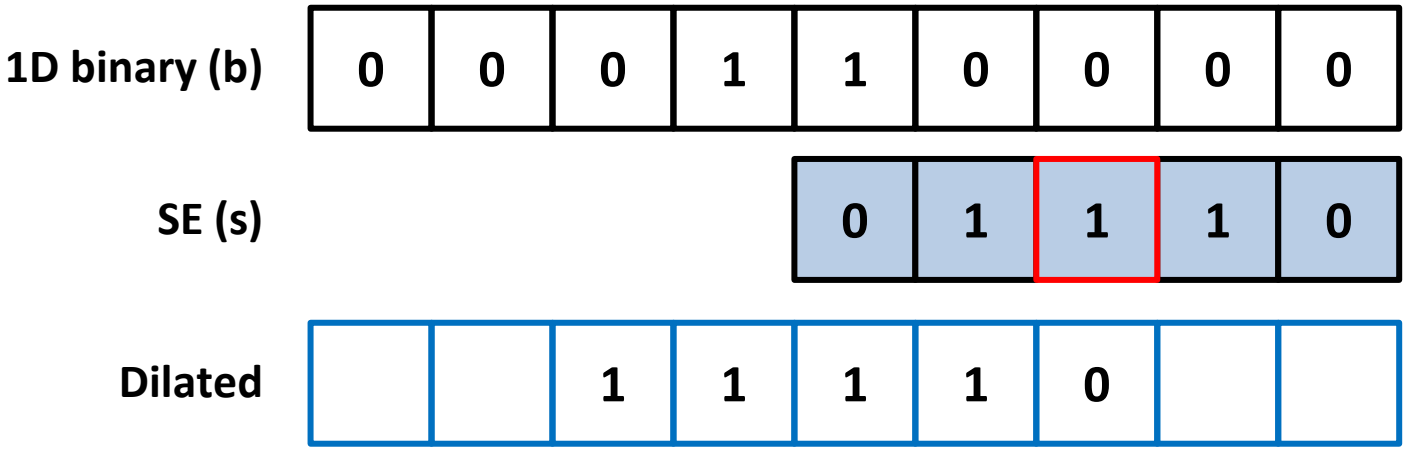
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**



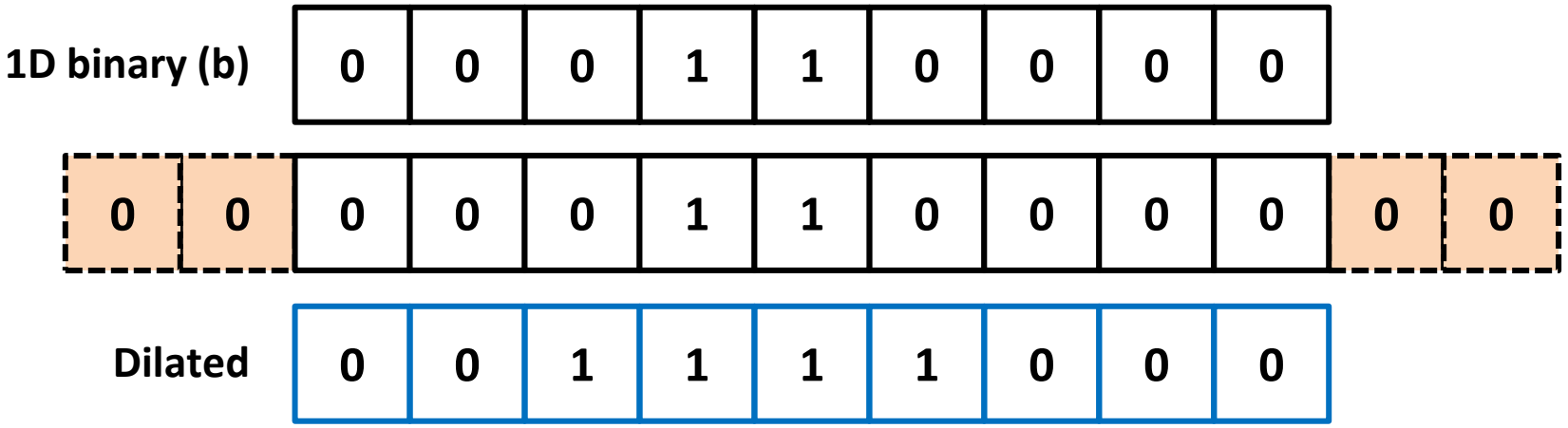
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**



Foreground Detection: Morphological Operations

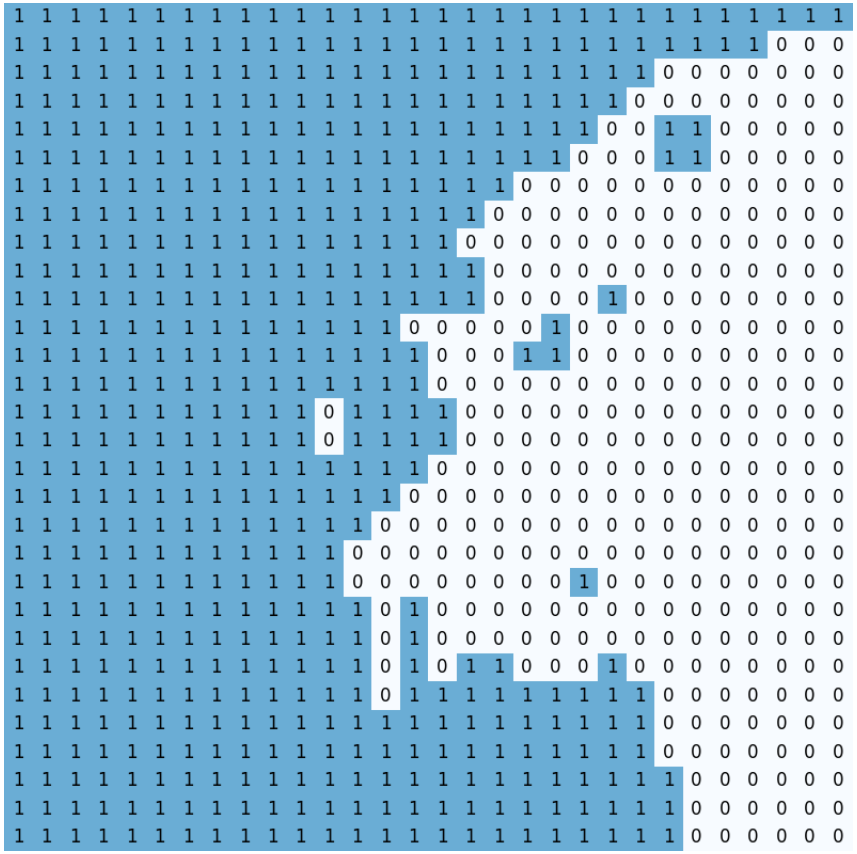
- **Dilation: expanding masks**



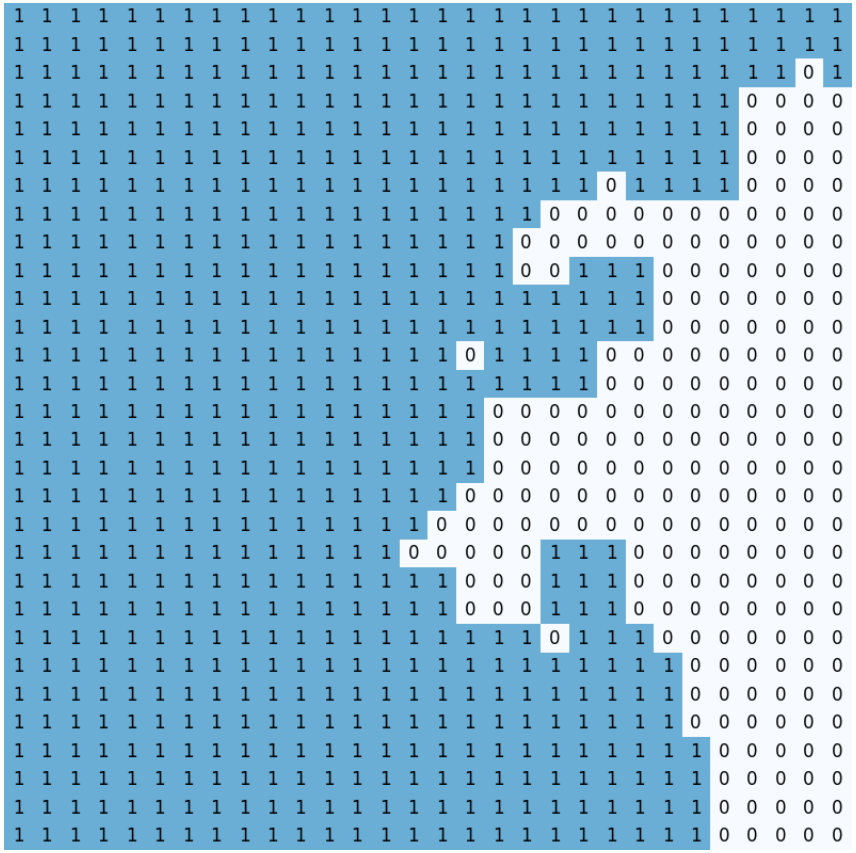
Foreground Detection: Morphological Operations

- **Dilation: expanding masks**

THRESHOLDED



DILATED (SE np.ones((3,3)))



Foreground Detection: Morphological Operations

- **Common morphological operations**

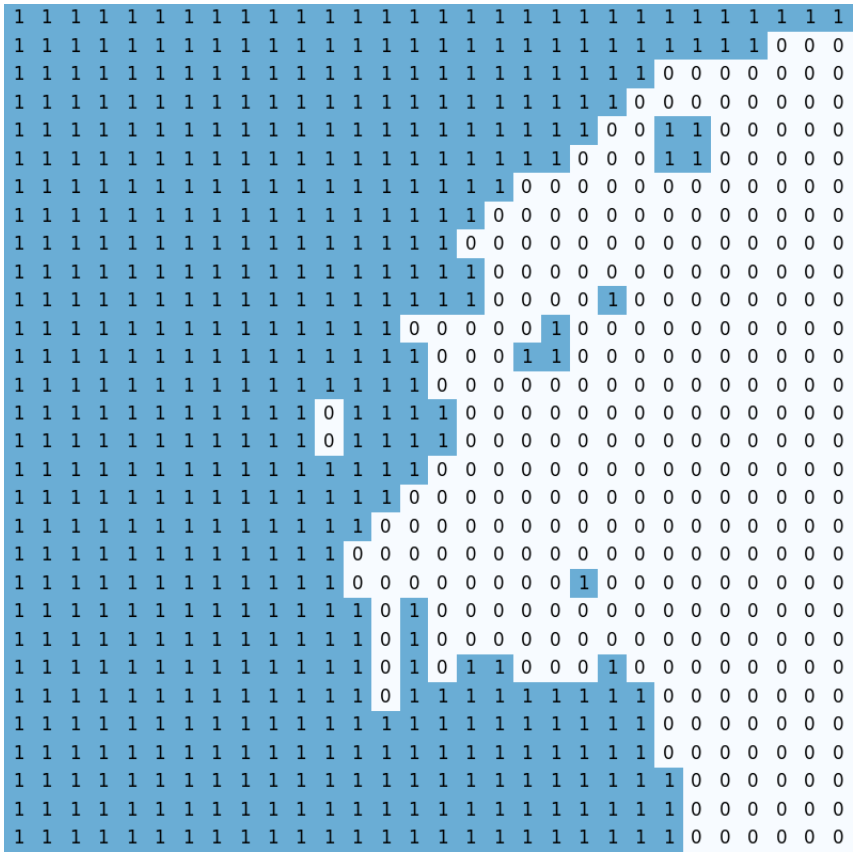
```
Dilation:   max(b[0:5] * s)
Erosion:    min(b[0:5] * s)
Closing:    erosion(dilation(b,s))
Opening:    dilation(erosion(b,s))
Hole filling [more complicated]
```

- **Some notes**
  - Closing and opening more or less preserve mask area
  - The shape of the SE matters a lot (disc-shapes are usually preferred)
  - Combine morphological operations to get the desired effect

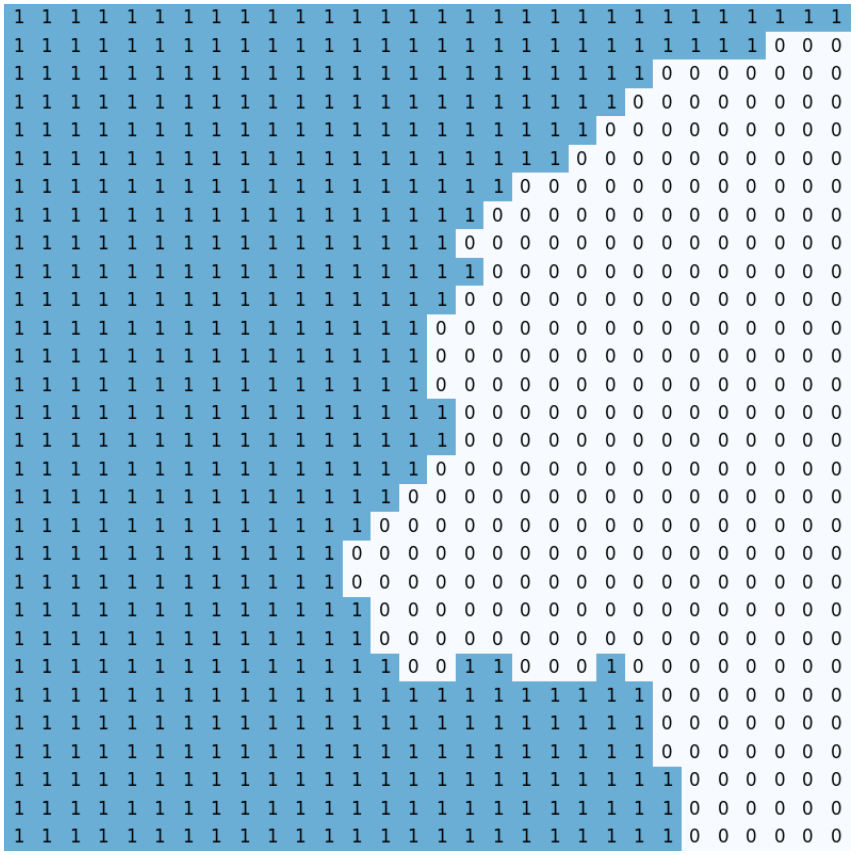
Foreground Detection: Morphological Operations

- **Combine morphological operations to get the desired effect**

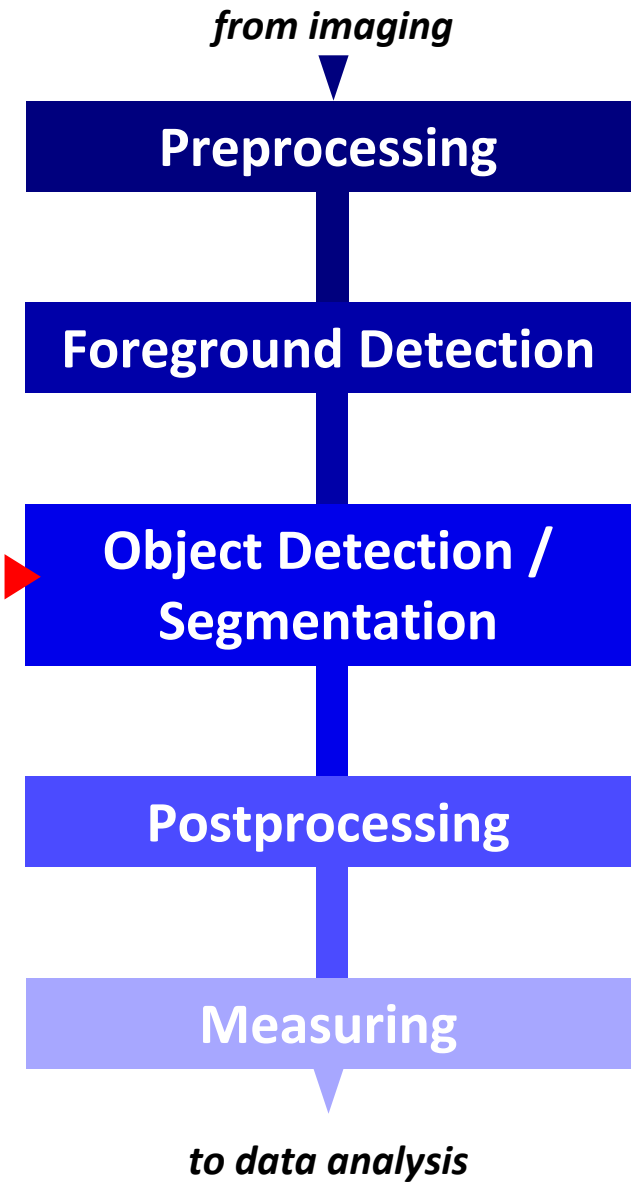
THRESHOLDED



CLOSING(OPENING(THRESHOLDED))



Standard Pipeline Overview



	Frequent Tasks	Important Concepts
Preprocessing	Filtering Background subtraction	Convolution Filter kernels
Foreground Detection	Thresholding Morphology	Manual vs. automated Uniform vs. adaptive Morphological operations
Object Detection / Segmentation	Labeling Seeding & Expansion	
Postprocessing	Object filtering	

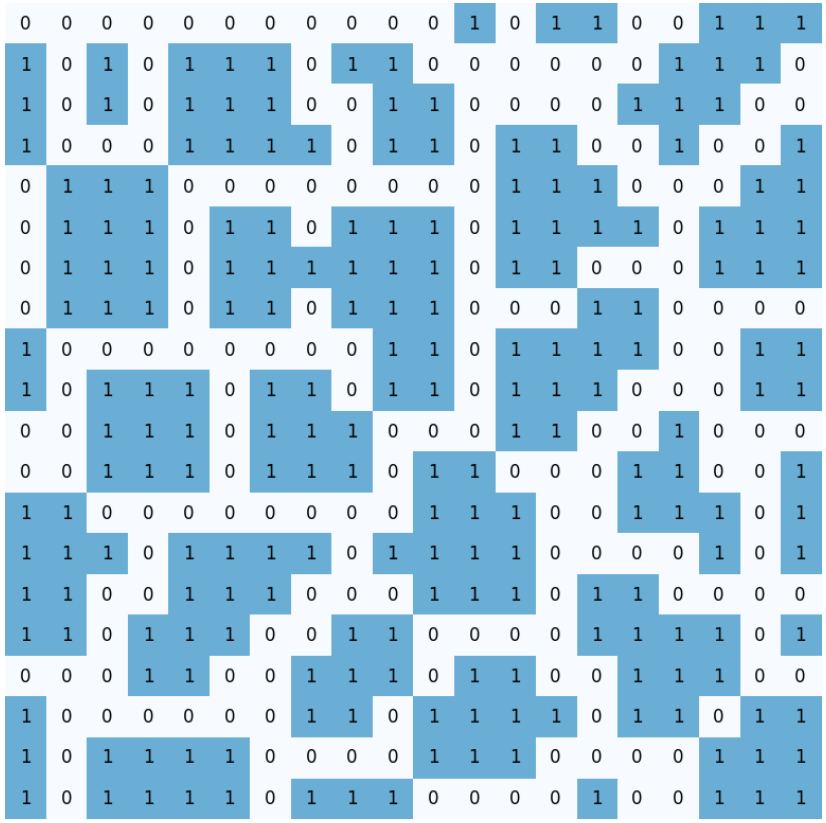
+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

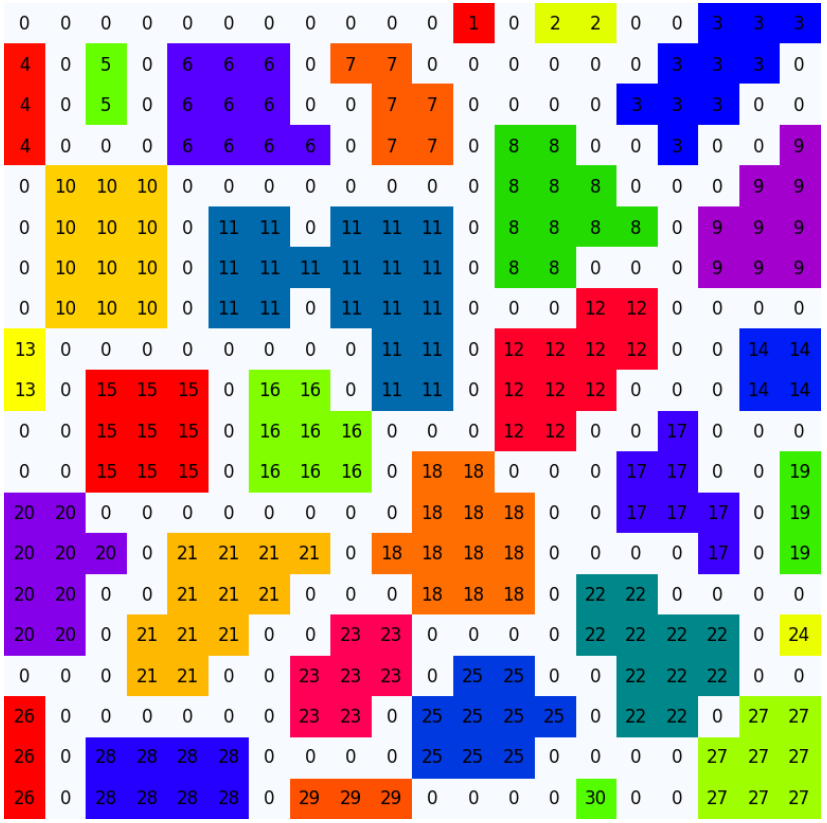
Segmentation: Connected Components Labeling

- **Goal: ‘multi-object’ masks**
  - **Achieved by labeling each isolated foreground object with a unique integer**

MASK



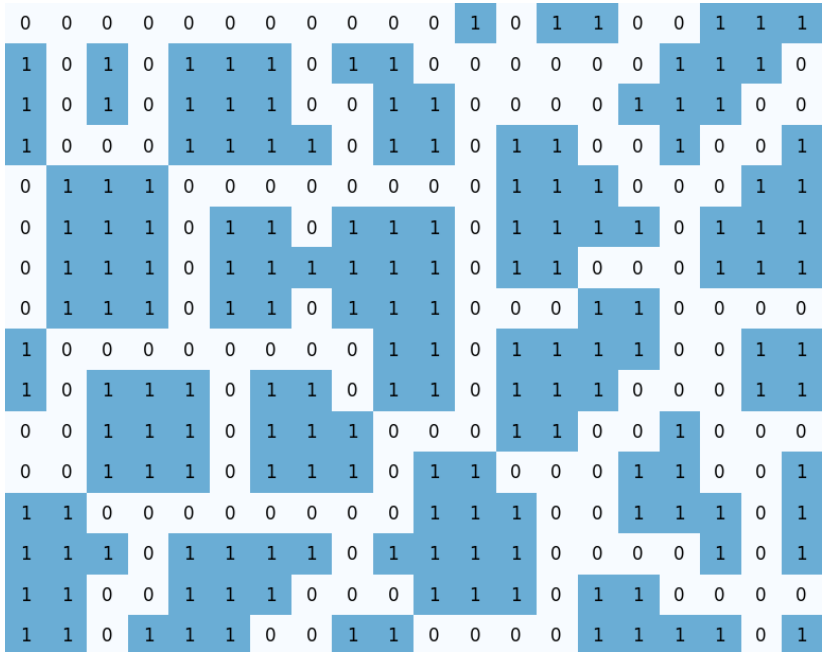
LABELS



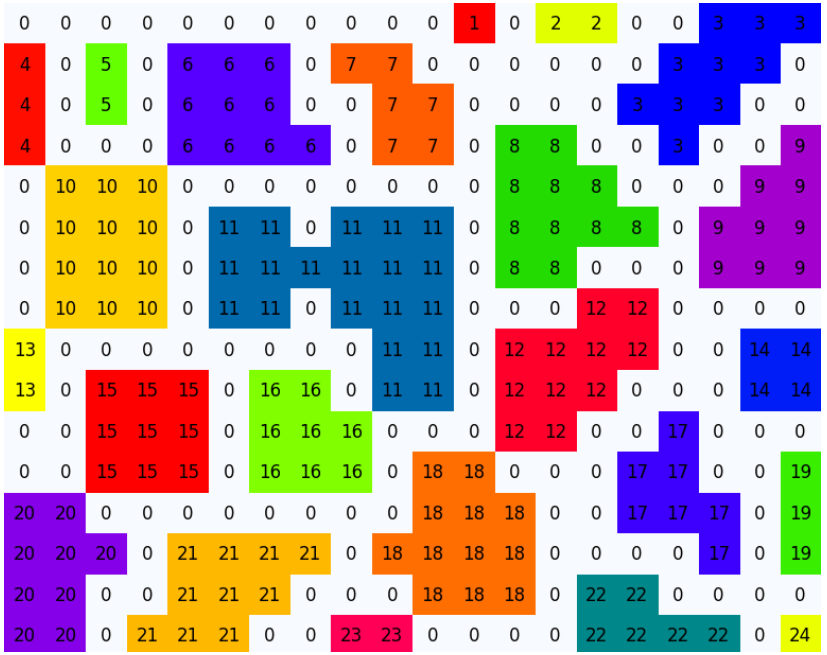
Segmentation: Connected Components Labeling

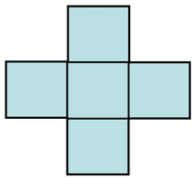
- **Goal: ‘multi-object’ masks**
  - **Achieved by labeling each isolated foreground object with a unique integer**

MASK

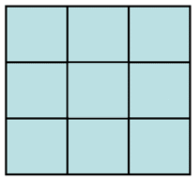


LABELS





4-connected

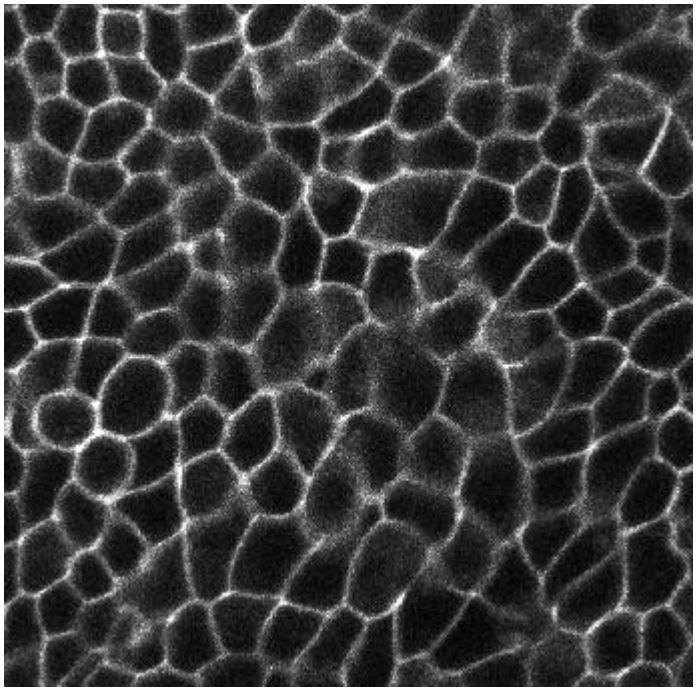


8-connected

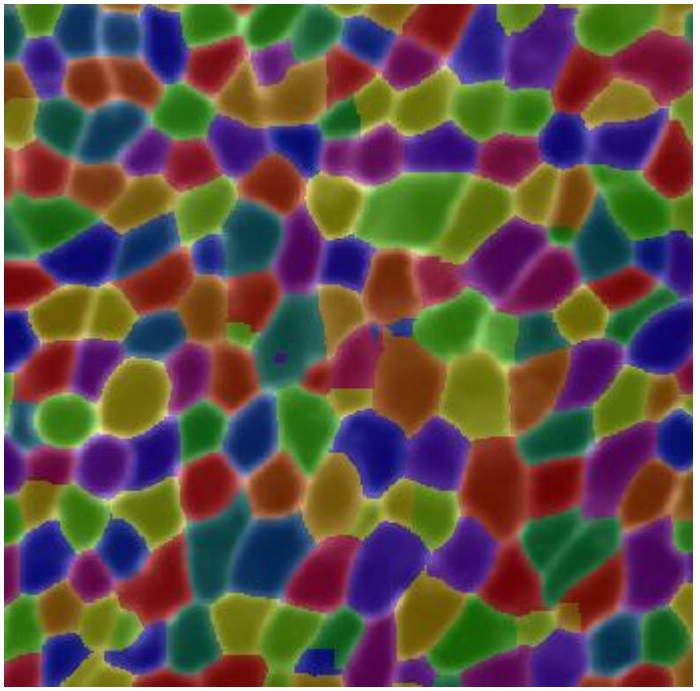
Segmentation: Seeding & Expansion

- **Goal: full segmentation of ‘hollow’ objects**

RAW



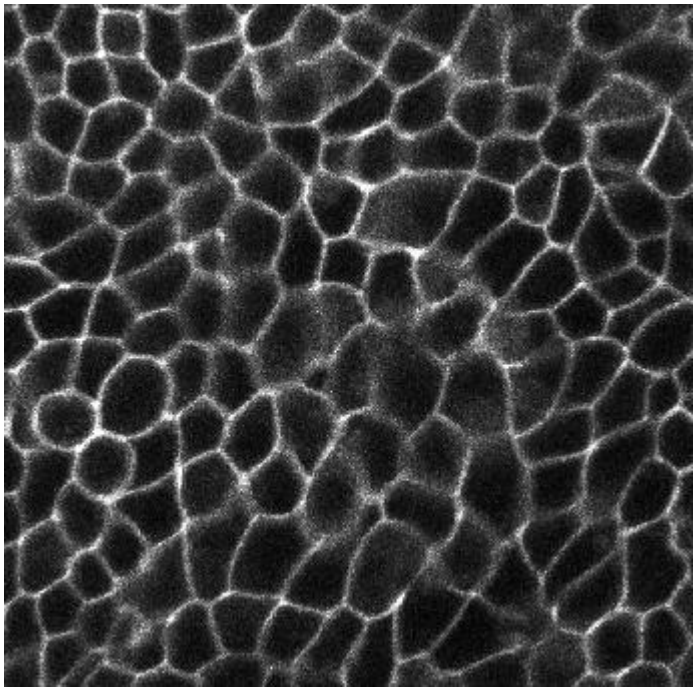
FULL SEGMENTATION



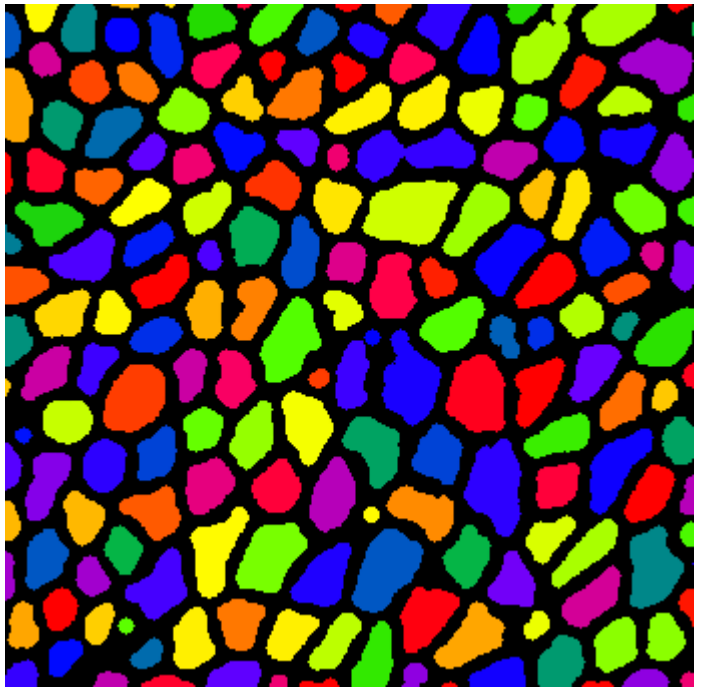
Segmentation: Seeding & Expansion

- **Goal: segmentation of 'hollow' objects**
  - **Problems: assigning boundaries, splitting at holes**

RAW



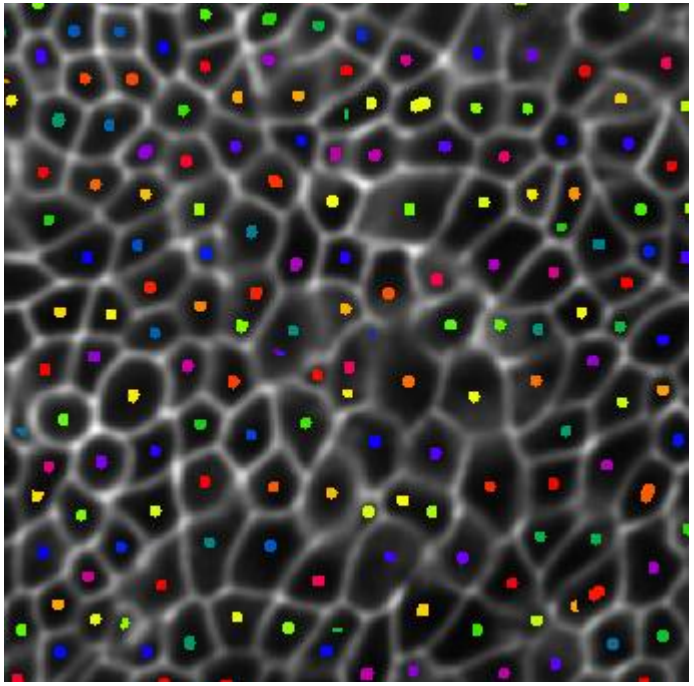
MASKED + MORPH. PROCESSED + LABELED

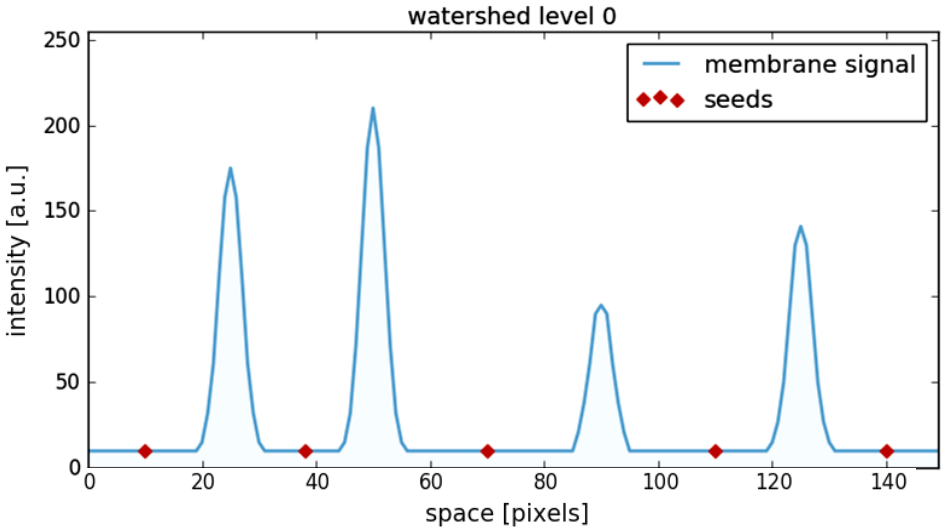


Segmentation: Seeding & Expansion

- **Solution: more advanced algorithms!** ☺
  - **Most common approach: seeding & watershed**
  - **Alternatives: level set, random walker, …**

seeds

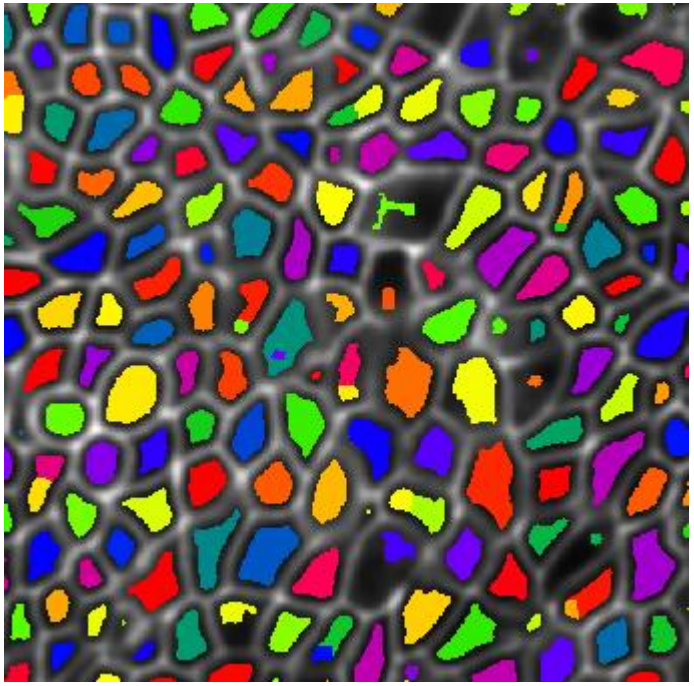


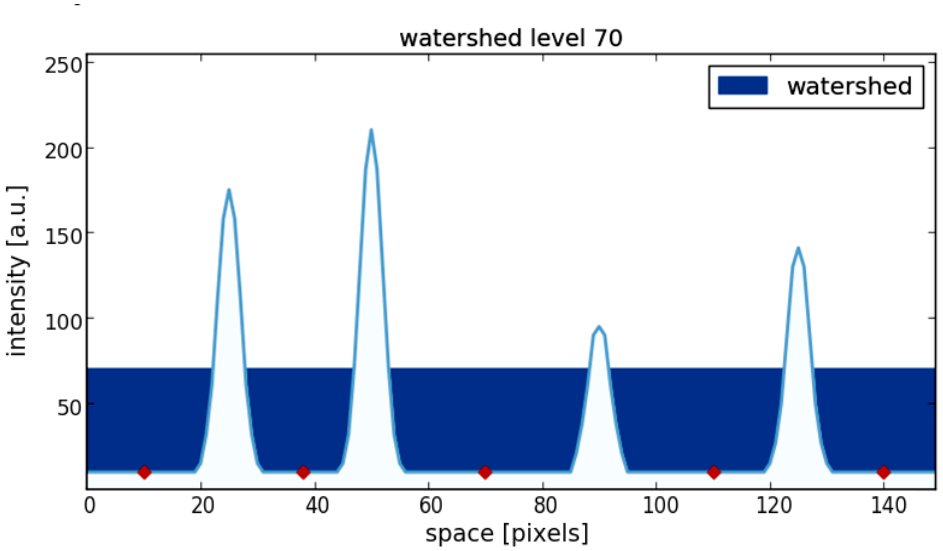


Segmentation: Seeding & Expansion

- **Solution: more advanced algorithms!** ☺
  - **Most common approach: seeding & watershed**
  - **Alternatives: level set, random walker, …**

watershed level 70

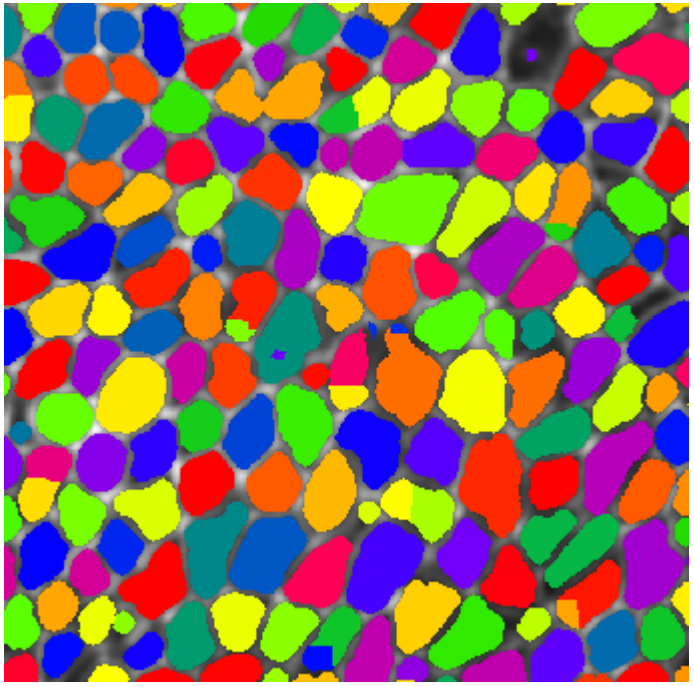


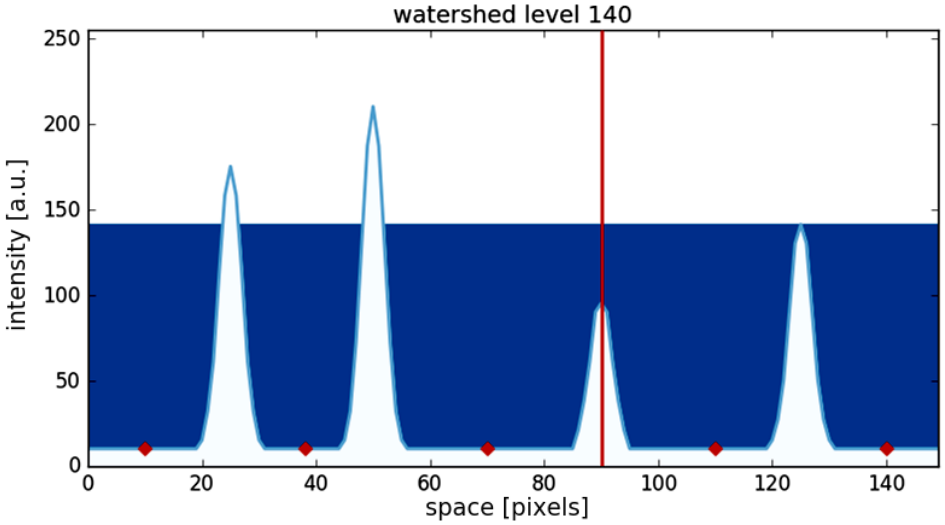


Segmentation: Seeding & Expansion

- **Solution: more advanced algorithms!** ☺
  - **Most common approach: seeding & watershed**
  - **Alternatives: level set, random walker, …**

watershed level 140

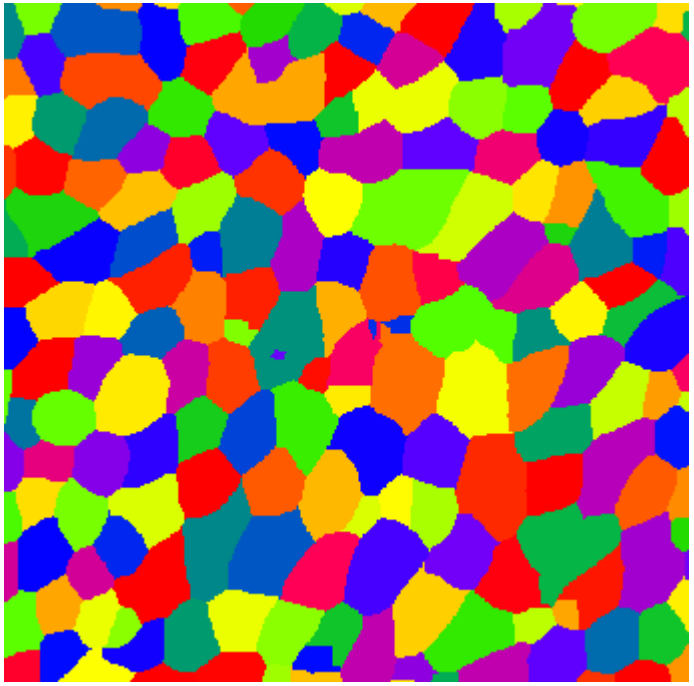


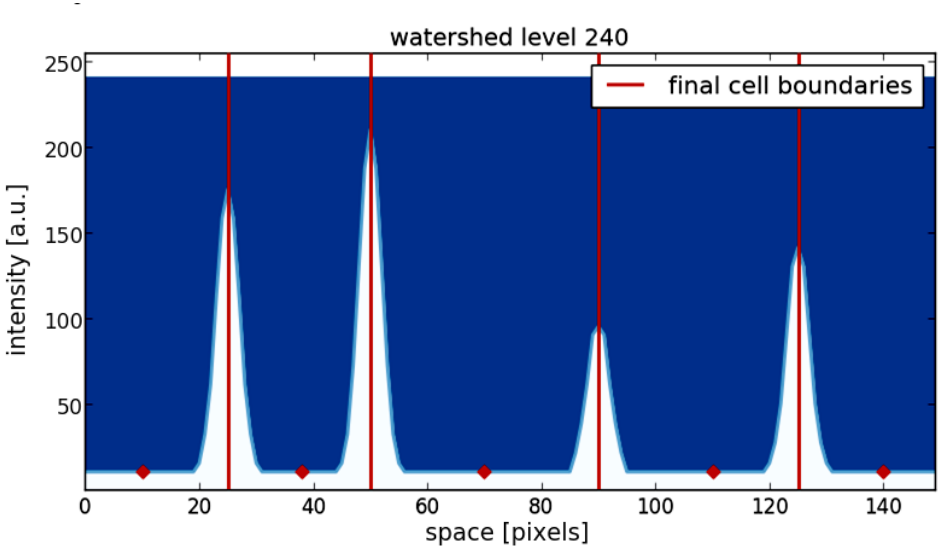


Segmentation: Seeding & Expansion

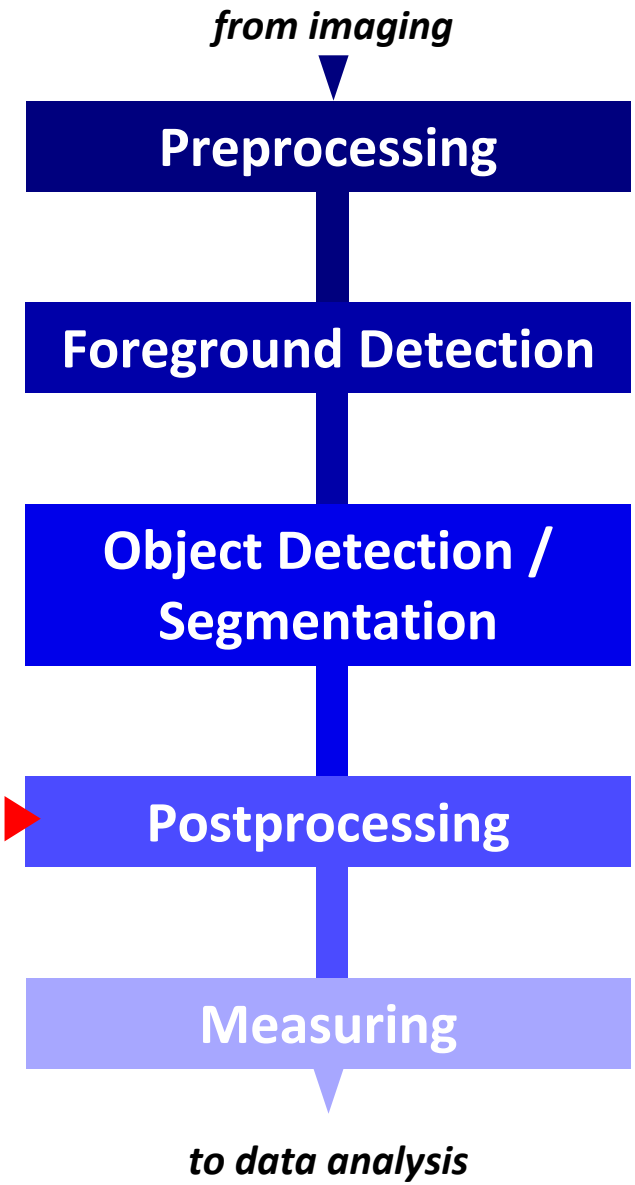
- **Solution: more advanced algorithms!** ☺
  - **Most common approach: seeding & watershed**
  - **Alternatives: level set, random walker, …**

final cell segmentation





Standard Pipeline Overview



Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

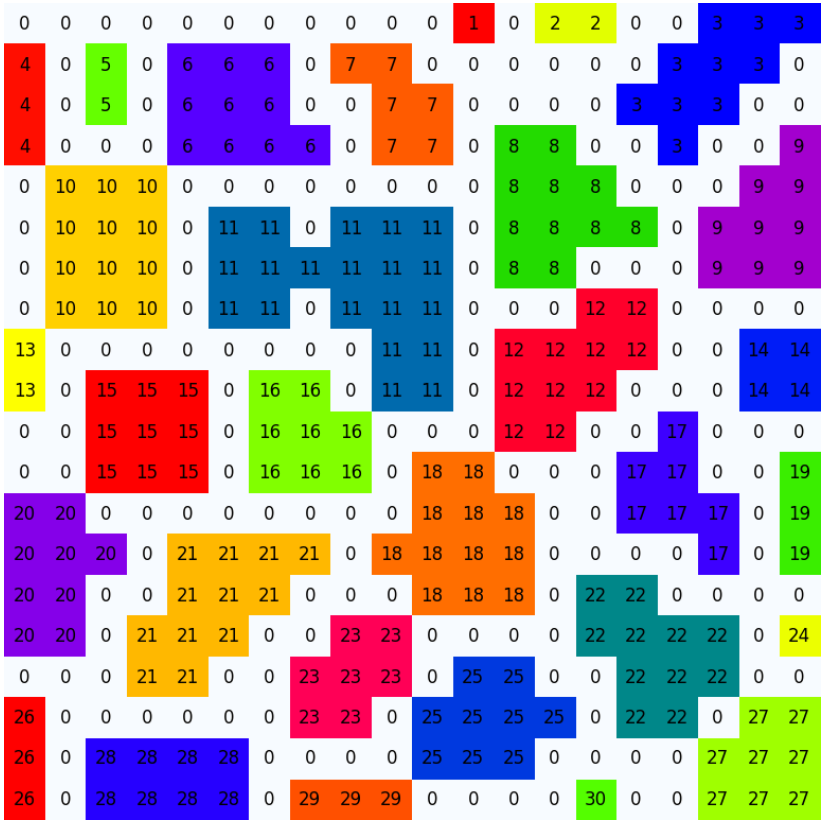
+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

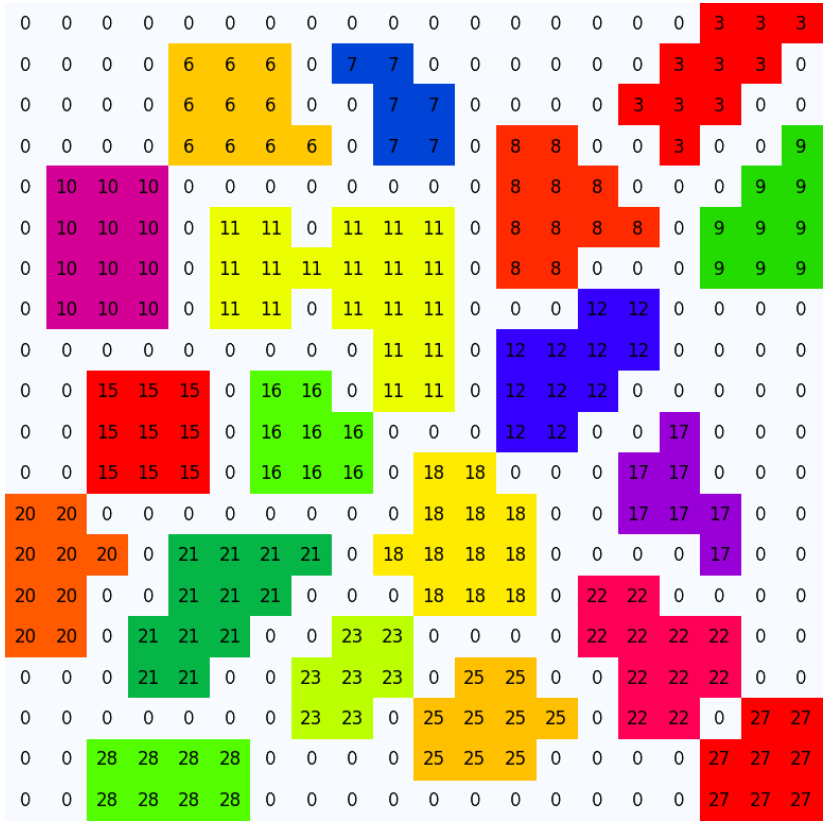
Postprocessing: Object Filtering

- **Identify unwanted objects and discard them**
  - **e.g. very small objects**

LABELED

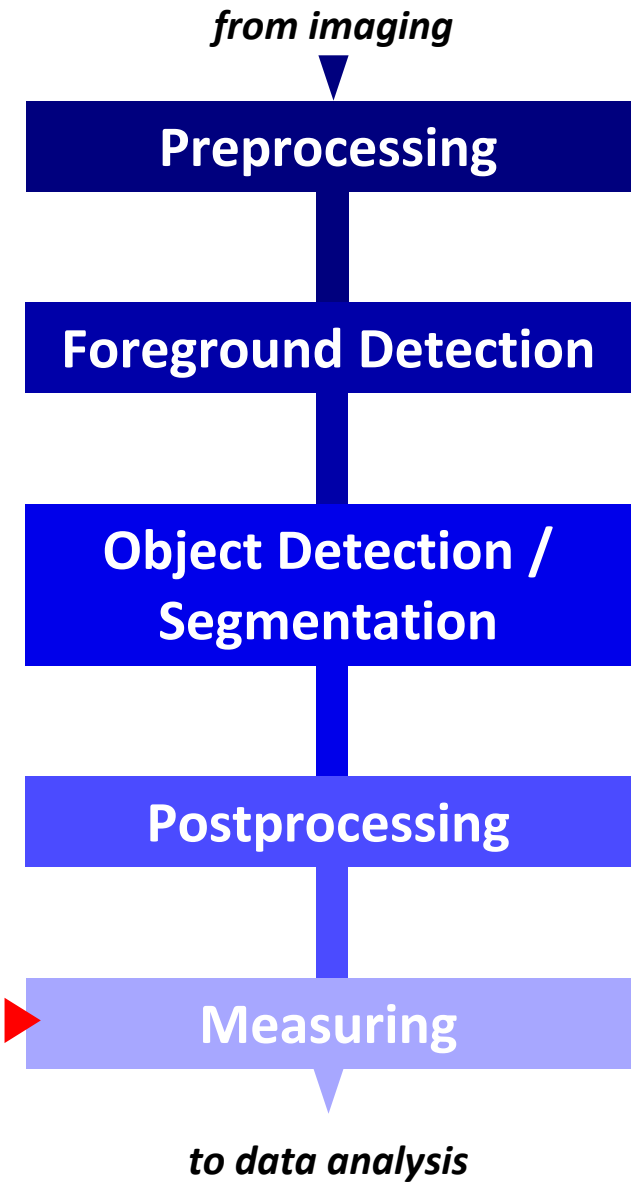


FILTERED BY OBJECT SIZE



Warning: **Objects are data. Deleting outliers at will is not permissible!**

Standard Pipeline Overview



Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

+ Some bonus stuff tomorrow

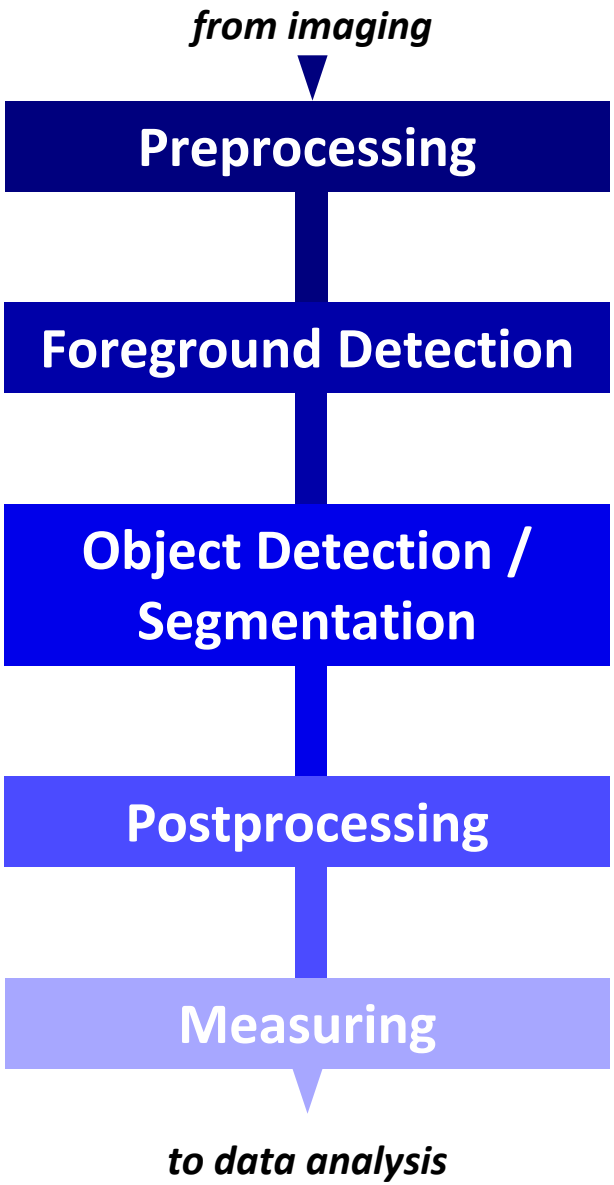
Machine Learning
Tracking
Smart Microscopy
Data Analysis

Measuring Stuff

- **Goal: extracting measurements from (segmented) images**
  - **How this is done will become clear in the tutorial**
  - **I may mention some additional concepts tomorrow**

- **Important note: Images are *scientific data!***
  - **Obviously not permissible: intentional falsifications (very rare!)**
  - **Also not permissible: *unintentional inappropriate handling!***
  - **Guidelines: www.ncbi.nlm.nih.gov/pmc/articles/PMC4114110/**

Standard Pipeline Overview



Frequent Tasks	Important Concepts
Filtering	Convolution
Background subtraction	Filter kernels
Thresholding	Manual vs. automated
Morphology	Uniform vs. adaptive
	Morphological operations
Labeling	
Seeding & Expansion	
Object filtering	

+ Some bonus stuff tomorrow

Machine Learning
Tracking
Smart Microscopy
Data Analysis

Before we get started, let‘s have some coffee!

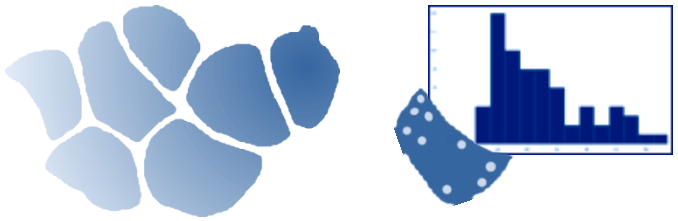
It‘s time to get to work!

Tutorial Pipeline: Goal & Motivation

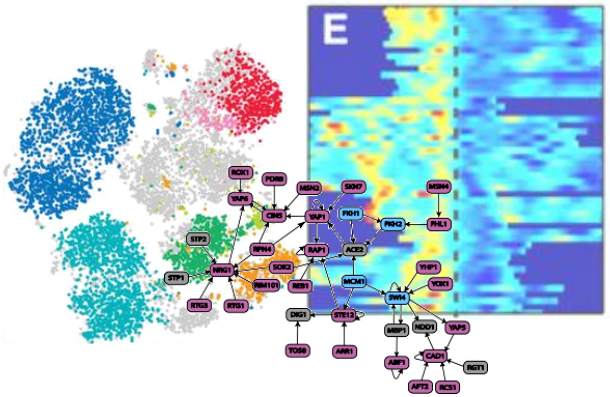
Automated Single-Cell Segmentation

- **Permits quantification of:**
  - Cell shapes
  - Cellular heterogeneity
  - Subcellular reporter distribution

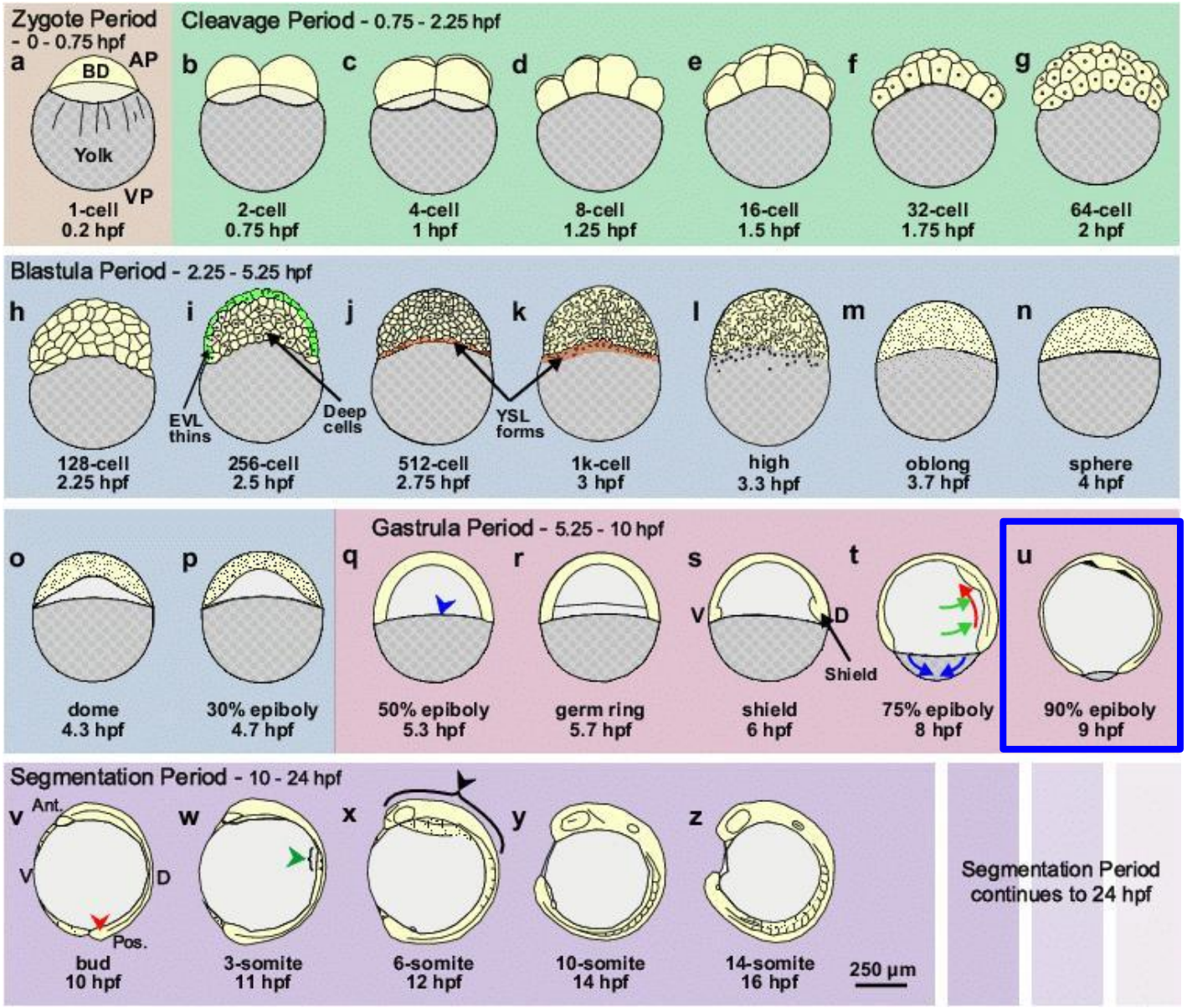
- **Improves quantification of:**
  - Reporter gene expression
  - Stainings (antibodies, smFISH, ...)



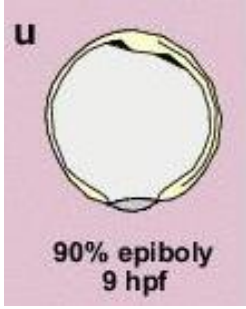
- **Allows advanced analysis**
  - Bioinformatics approaches (clustering, etc…)
  - Overlaying experiments (‘atlases’)
  - Inference of interactions and mechanisms



Tutorial Pipeline: Sample Images

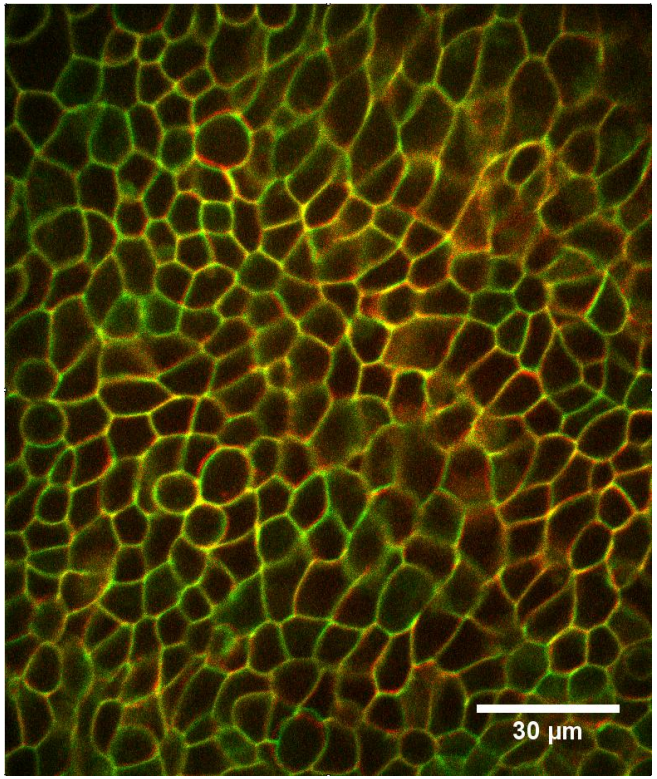


Tutorial Pipeline: Sample Images

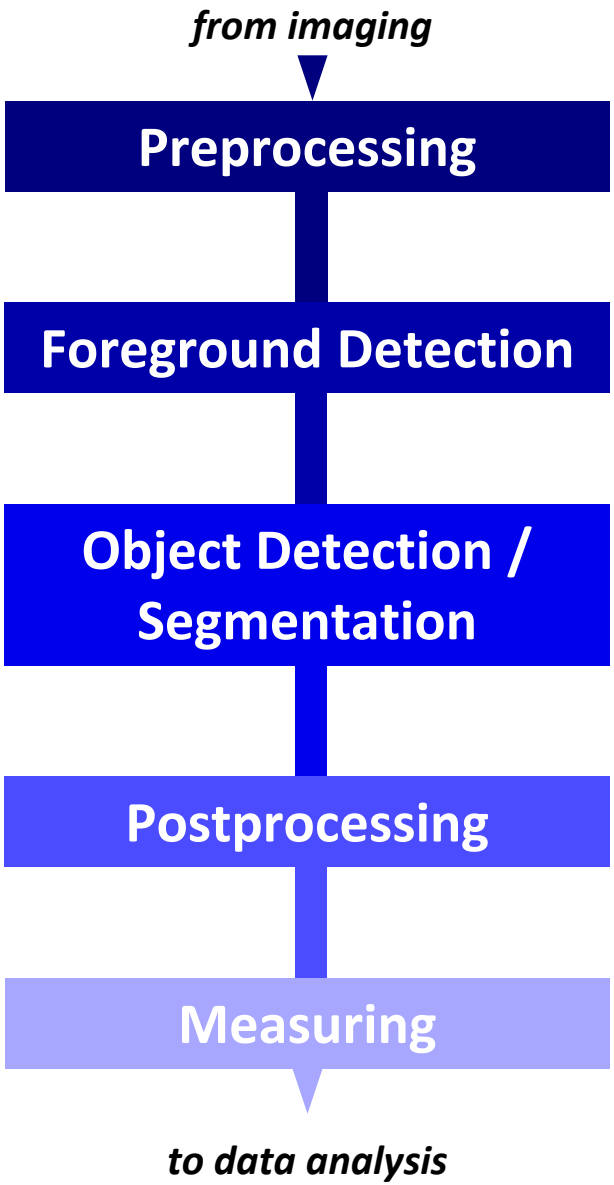


- Early zebrafish embryo
- An “in vivo cell culture”
- Observable: division, migration, differentiation, morphogenesis

- 40X spinning disc confocal slice
- Green: mNG:G*γ9* *(G-protein)*
  - *The important channel for us!*
- Red: *Cxcr4b:tagRFP-T (GPCR)*
- “Real-World Data”



Tutorial Pipeline: Outline



Tutorial Pipeline

Import to numpy array

Gaussian smoothing

Adaptive thresholding

Morphological clean-up

Seeding with distance transform

Watershed

Filter Objects at Image Border

Intensity and Shape Parameters

Simple Analysis and Visualization

Saving of data and figures

Good Luck! ;)