

Image Processing and Computer Graphics

Image Processing

Class 1 Introduction

- Language: English (older German recordings available)
- Type of lecture:
 - 3 hours classroom lecture, 1 hours tutorial (6 ECTS = 180h)
 - Course gives you an overview on computer vision and graphics
 - Required for a specialization in our groups (e.g. Bachelor thesis, projects)
- Prerequisites
 - Solid undergraduate mathematics
 - Programming experience in C/C++
(for the programming assignments)

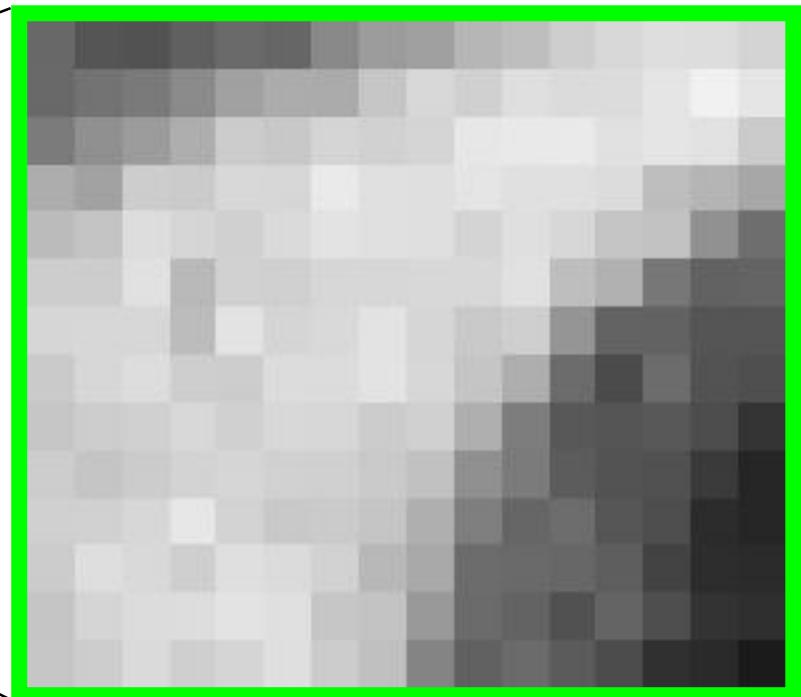
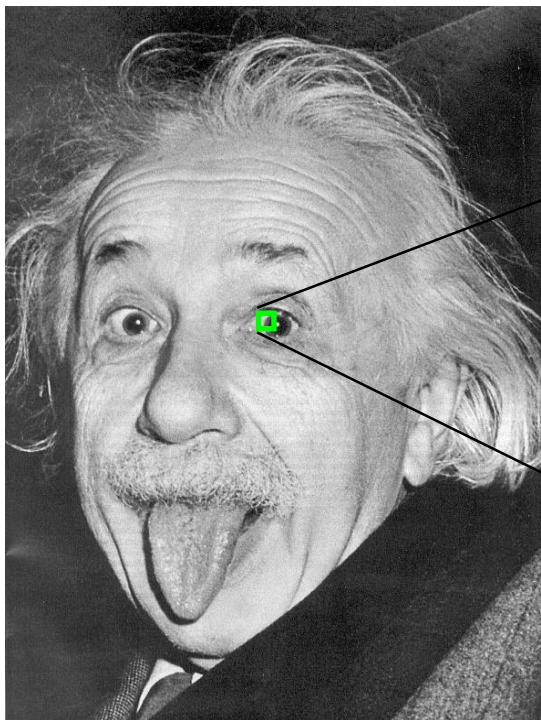
Organizational issues

- Tutorials
 - ~every second Tuesday in the computer pools (see webpage for dates)
 - Advisors: Mostafa Morsy, Benjamin Ummenhofer, Nima Sedaghat
 - Consists of programming assignments
 - Online forum with link on the course webpage (access with your RZ-account)
- Exam
 - Written exam
 - Make sure you are registered for the exam in time
- These slides are made available at:
<http://lmb.informatik.uni-freiburg.de/lectures> (user: open, passwd: thebox)

On the purpose of studying

- Aim primarily for **education**, not just expert knowledge or the next exam
- You're not in school anymore:
It is your responsibility to make good use of what we offer
- This course provides you with opportunities to:
 - Improve your understanding of math and abstract concepts
 - Train your programming and debugging skills
 - Challenge your ability to solve problems independently
 - Efficiently search for missing information (asking the right questions)
 - Exploit the advantages of team work
 - Improve your English skills
- Not all of this will be tested and graded in the exam, but in life
- Improve your abilities, not just your CV

What is an image?



Author: Daniel Cremers

- Digital image: regular grid I_{ij} of intensity values
- Continuous function $I : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}$ $(x, y) \mapsto I(x, y)$

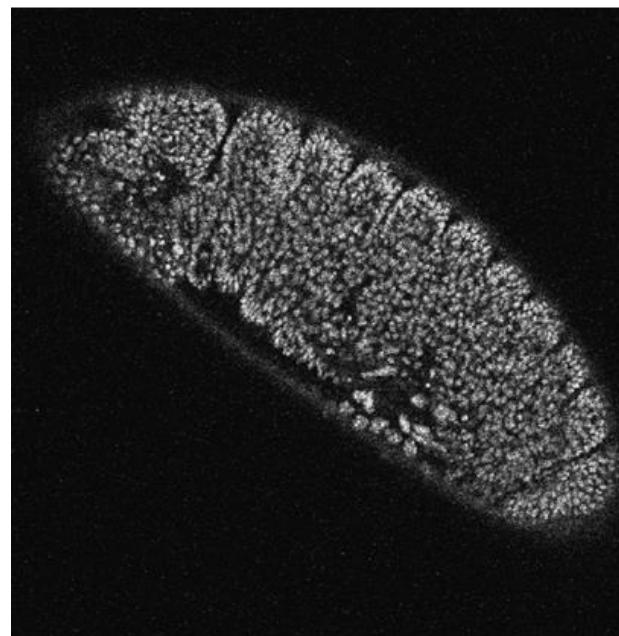
Where do images and image processing play a role?

- Imaging (e.g. photography, ultrasound, magnetic resonance, microscopy)
- Image enhancement and modification (e.g. Adobe Photoshop)
- Image and video compression
- Computer graphics (model → image)
- Computer vision (image → model)
- Focus of our group: computer vision

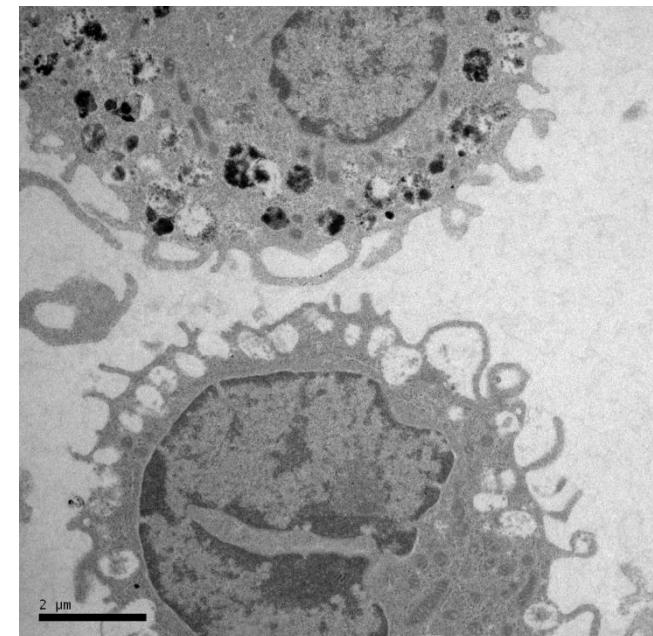
Imaging – MRI, confocal or electron microscopy



Magnetic resonance
image of a human head

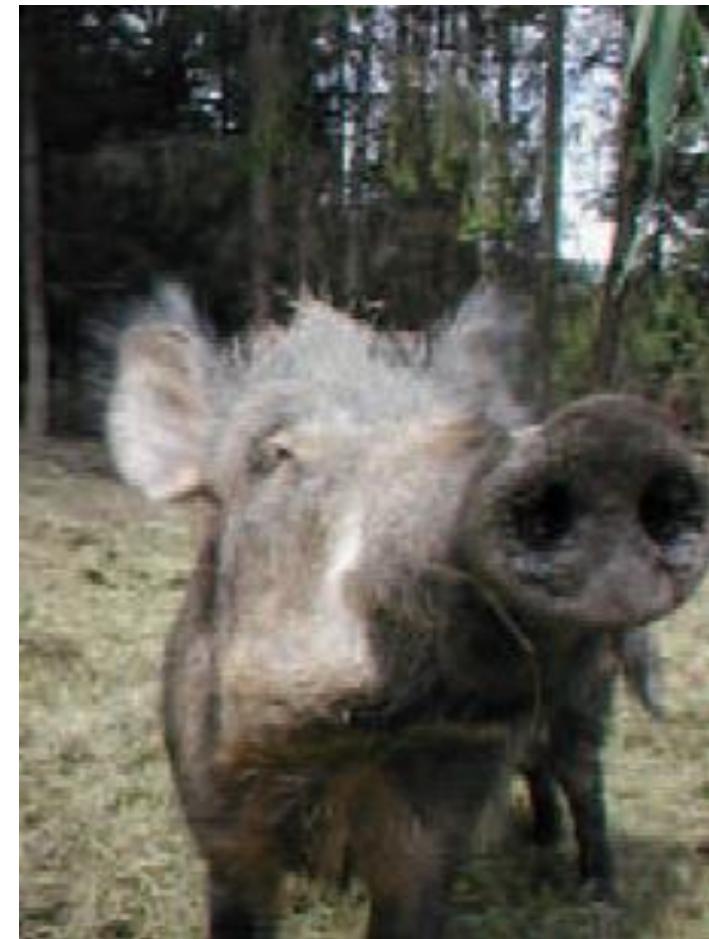


Confocal microscope
image of a fruit fly embryo



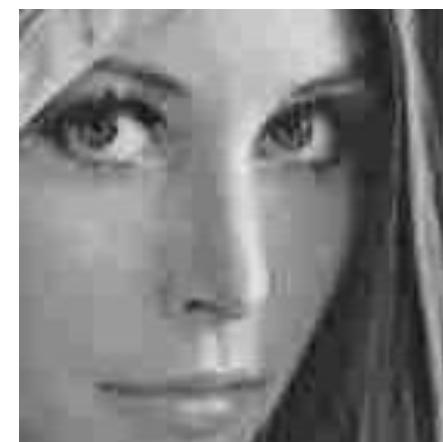
Mast cell image from an
electron microscope

Image enhancement - deblurring



Welk et al. 2005: variational deblurring

Image compression - JPEG

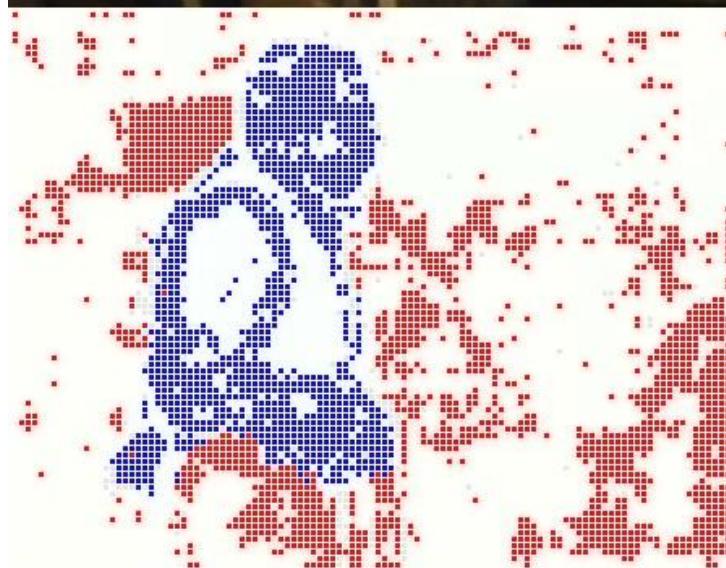


Computer vision - segmentation



Wedel et al. 2007: obstacle detection and segmentation

Computer vision – motion segmentation



Brox-Malik ECCV 2010: motion segmentation from point trajectories

Citywall Data Set

Input: 256 million points

P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov

P. v.d. Smagt, D. Cremers, T. Brox

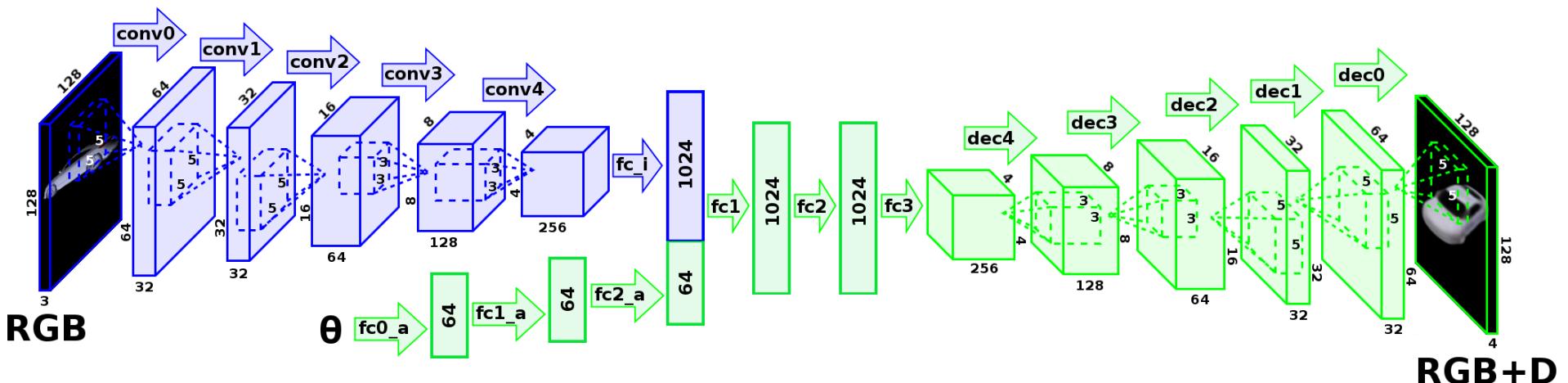
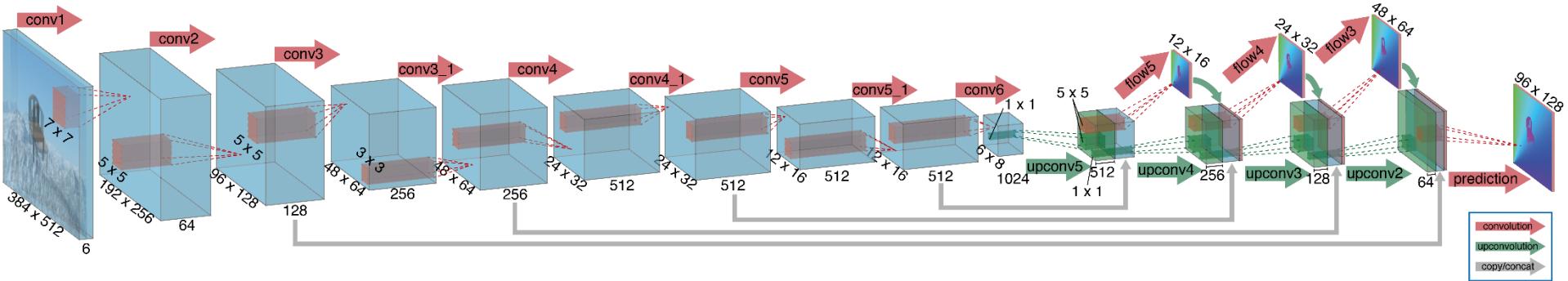
FlowNet: Learning Optical Flow with Convolutional Networks

Depth estimation with a deep network



DispNet result on KITTI 2015

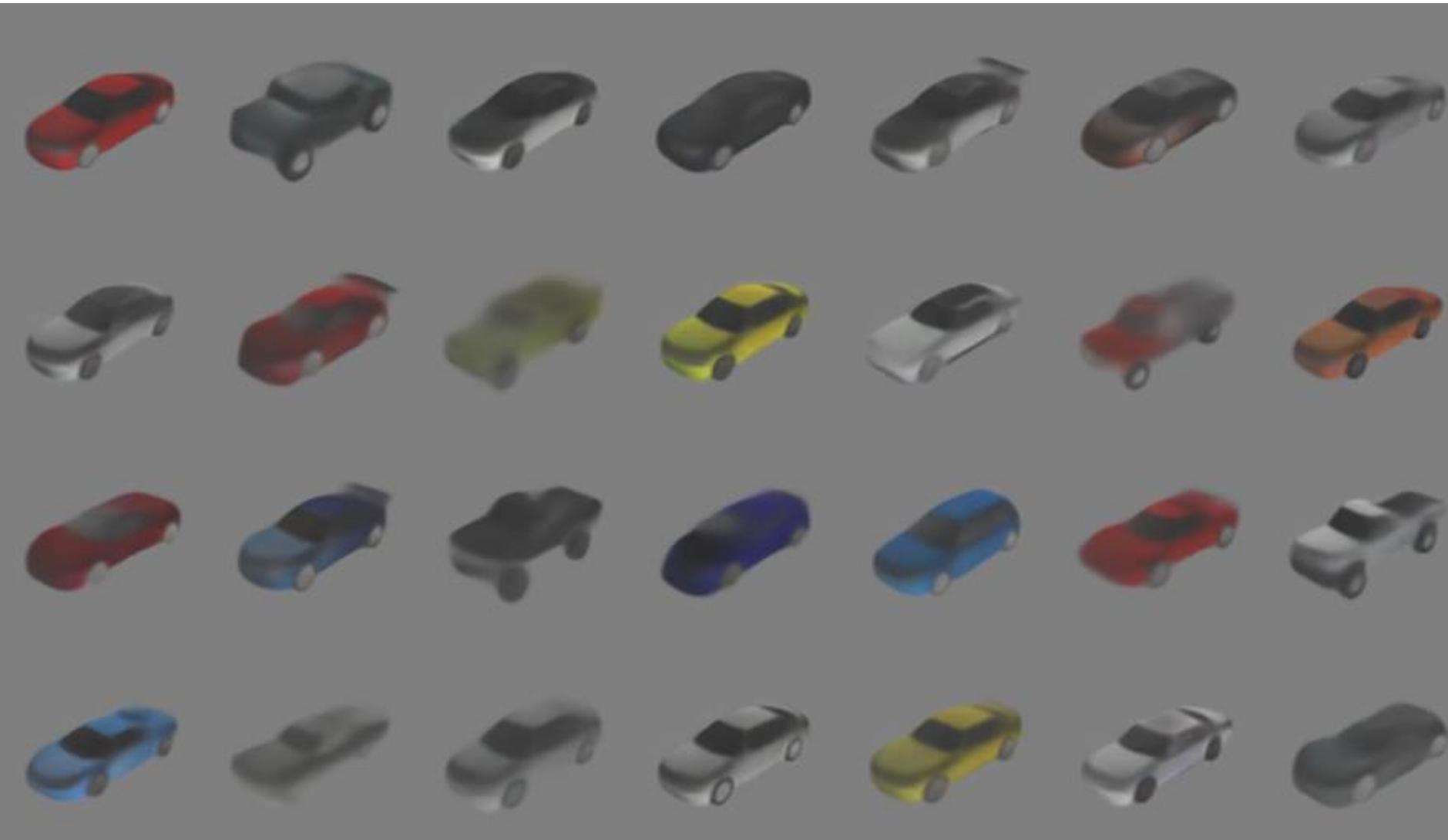
Deep learning can help with (almost) everything in computer vision



Generating images



3D morphing of cars



Artistic style transfer for videos

Manuel Ruder
Alexey Dosovitskiy
Thomas Brox

University of Freiburg
Chair of Pattern Recognition and Image Processing

Master project of Manuel Ruder (Ruder et al. 2016)
Won a GCPR Best Paper Honorable Mention
Featured in the ARD Nachtmagazin

Computer vision – object class recognition and segmentation



Carreira et al. ECCV 2012

Image caption generation



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"little girl is eating piece of cake."



"baseball player is throwing ball in game."



"woman is holding bunch of bananas."

Karpathy-Li 2014

Difference between image processing and computer vision

- The term “image processing” is used for the general subject of processing images with a computer
- Computer vision signifies the specific task to have the computer **interpret the content of images**
- Requires pattern recognition and machine learning techniques
- Computer vision has similar motivations as artificial intelligence and cognitive neuroscience (biological vision)

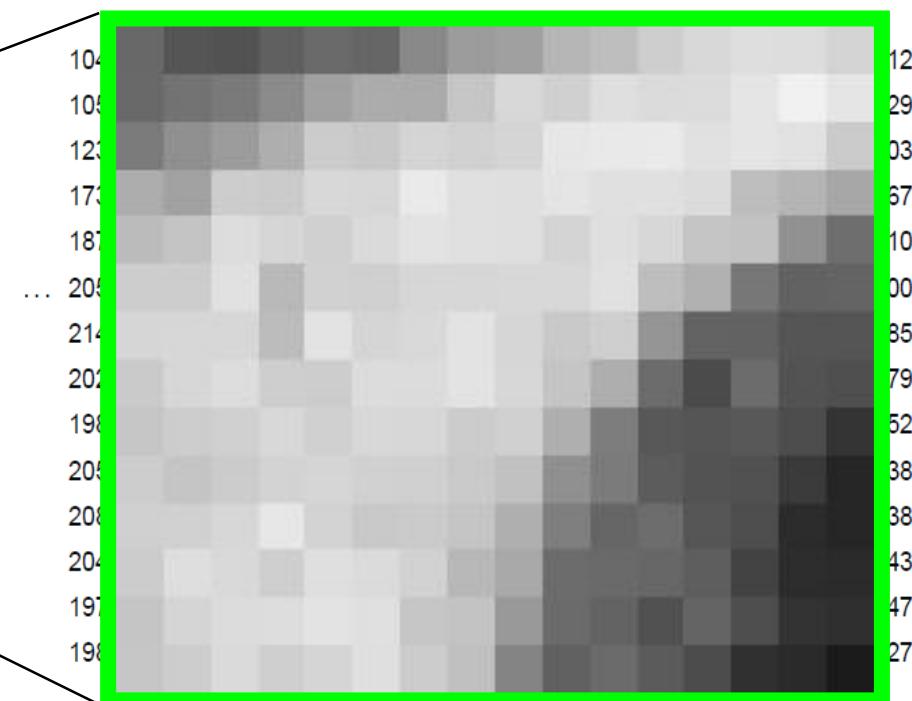
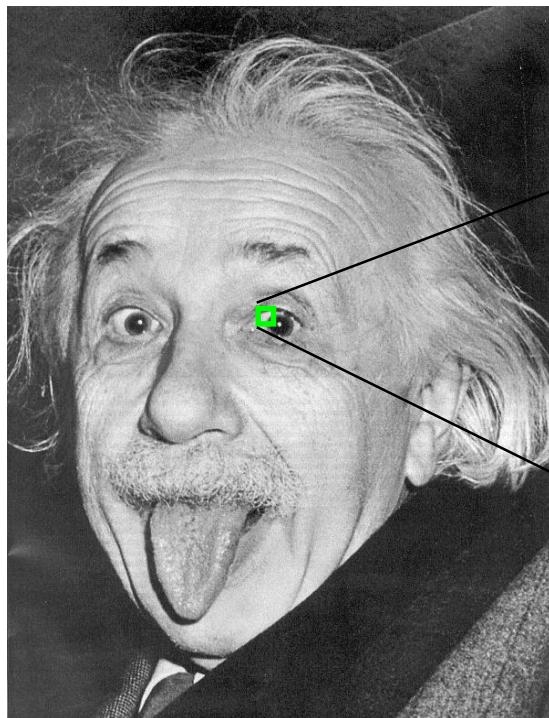
Why is computer vision difficult?

- Vision is a natural and easy task for humans (and many animals)
- This is not for free: ~50% of the primate's cortex deals with the processing of visual information (Felleman-van Essen 1991)
- Matching human visual capabilities means to solve a large part of the AI problem



Images are only a structured grid of numbers

- What's this?



Author: Daniel Cremers

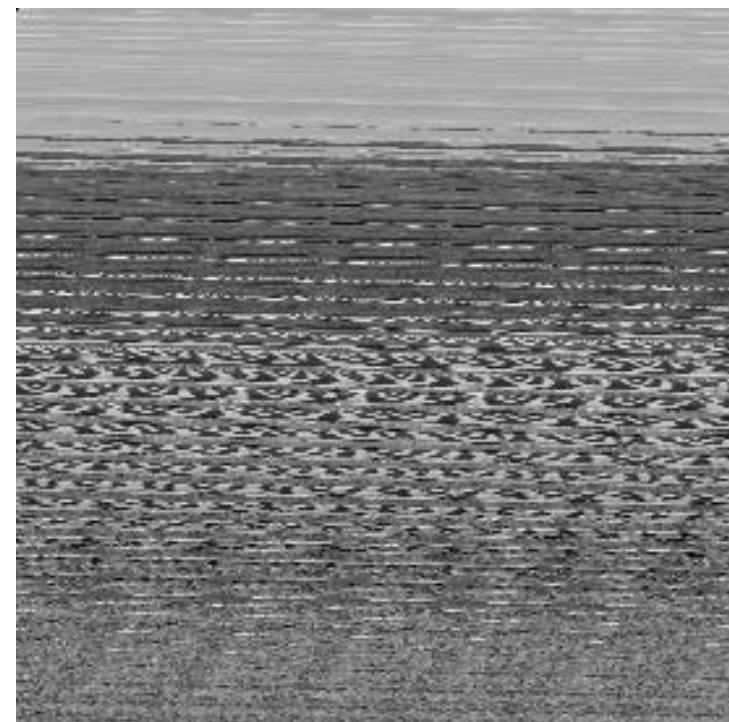
- This is how the computer sees Einstein
- Demonstrates what our brain achieves at the unconscious level

Importance of the spatial arrangement

- Image content is defined by the spatial arrangement of intensities
- It is not sufficient to treat images as vectors and to analyze these vectors



Zebra image



Same image with a different
row length

Ambiguities resolved by context



Importance of image processing and computer vision

- Computer vision is a quite “young” research field
 - Main computer vision conference (ICCV) founded in 1987
 - Main computer vision journal (IJCV) founded in 1988
- Today one of the most influential fields in computer science
- Playground for new machine learning techniques
- Computer vision applications are sky rocketing due to progress in deep learning

Rank	Abbreviated Journal Title (linked to journal information)	ISSN	Total Cites	Impact Factor
1	ACM COMPUT SURV	0360-0300	2888	8.000
2	IEEE T PATTERN ANAL	0162-8828	25060	5.308
3	IBM J RES DEV	0018-8646	3049	5.216
4	INT J COMPUT VISION	0920-5691	9898	5.151
5	MIS QUART	0276-7783	7419	5.041
6	SIAM J IMAGING SCI	1936-4954	322	4.500
7	IEEE T EVOLUT COMPUT	1089-778X	4681	4.403
8	MED IMAGE ANAL	1361-8415	2901	4.364
9	INT J NEURAL SYST	0129-0657	816	4.237
10	HUM-COMPUT INTERACT	0737-0024	1175	4.000
11	J CHEM INF MODEL	1549-9596	9556	3.822
12	IEEE COMMUN SURV TUT	1553-877X	576	3.692
13	IEEE T MED IMAGING	0278-0062	11114	3.639
14	ACM T GRAPHIC	0730-0301	5892	3.632
15	J ACM	0004-5411	6116	3.375
16	J COMPUT AID MOL DES	0920-654X	3273	3.374
17	MATCH-COMMUN MATH CO	0340-6253	1677	3.291
18	COMPUT-AIDED CIV INF	1093-9687	856	3.170
19	INT J INF TECH DECIS	0219-6220	534	3.139
20	J AM MED INFORM ASSN	1067-5027	3619	3.088

Impact factors of the top 20 computer science journals

Related sciences

- Computer science
 - Machine learning
 - Robotics
 - Computer graphics
 - Parallel programming
- Mathematics
 - Optimization
 - Numerics
 - Statistics
 - Linear algebra
 - Functional analysis
 - Graph theory
- Electrical engineering
 - Signal processing
 - Systems engineering
 - Embedded systems
- Cognitive neuroscience
 - Psychophysics
 - Neurophysiology
 - Computational neuroscience

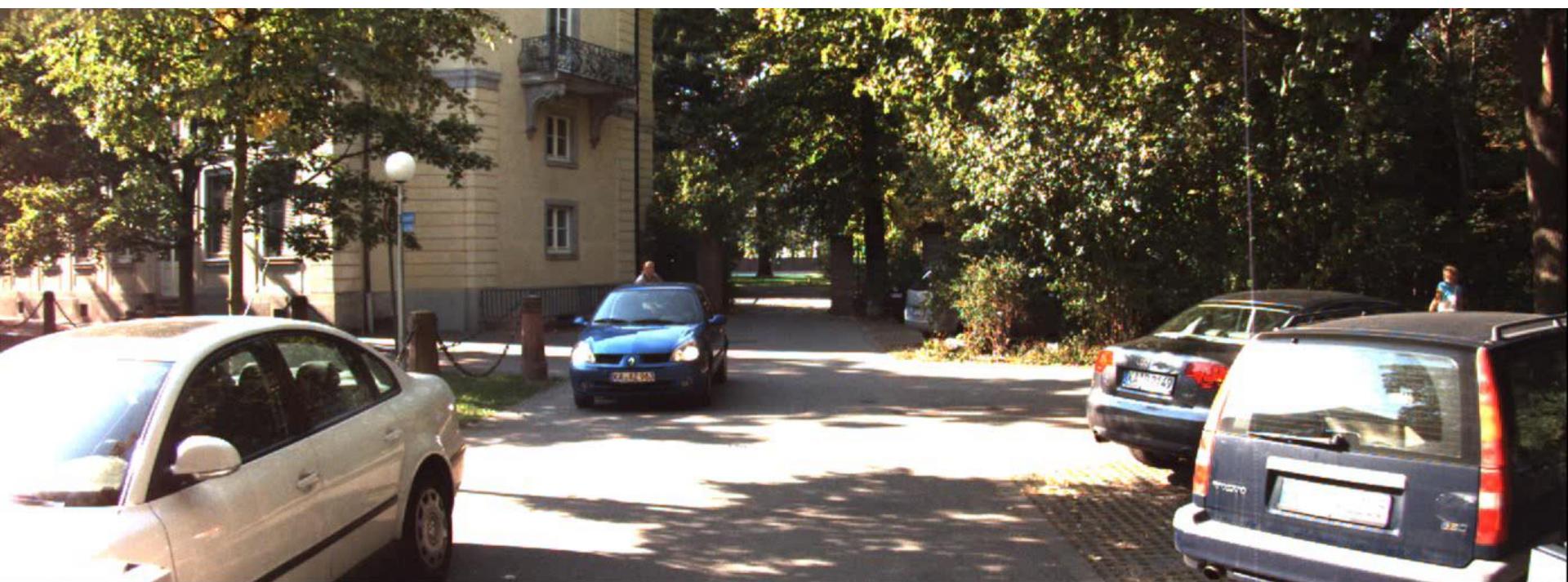
Conferences and Journals

- Conferences
 - **ICCV:** International Conference on Computer Vision
 - **ECCV:** European Conference on Computer Vision
 - **CVPR:** Int. Conference on Computer Vision and Pattern Recognition
 - **NIPS:** Neural Information Processing Systems
 - **ICML:** International Conference on Machine Learning
 - **ICLR:** International Conference on Learning Representations
 - **GCPR:** German Conference on Pattern Recognition
- Journals
 - **IEEE Transactions on Pattern Analysis and Machine Intelligence**
 - **International Journal of Computer Vision**
 - IEEE Transactions on Image Processing
 - SIAM Journal of Imaging Sciences
 - Journal of Mathematical Imaging and Vision

Some image processing applications

- Quality control, visual inspection
- Security systems
 - Fingerprint recognition
 - Iris recognition
 - Face recognition
 - Surveillance systems
- Medical and biological tools
 - Image enhancement
 - Routine diagnostics
 - Bioinformatics
- Entertainment industry and consumer products
 - Motion capture
 - Augmented reality
 - Human-machine interaction
 - Smartapps
- Web scale data analysis
 - Image Search
 - Video Search
- Photography and video editing
 - Smart cameras
 - Video editing tools
- Driver assistance systems
 - Collision avoidance
 - Autonomous driving
- Robotics
 - Object recognition
 - Visual SLAM
 - Visual learning
 - Vision based control

No self-driving cars without computer vision



Video from the KITTI dataset

- Class 1: Introduction
 - Class 2: Human vision and image basics
 - Class 3: Noise, basic operations and filters
 - Class 4: Energy minimization
 - Class 5: Variational methods
 - Class 6: Motion estimation
 - Class 7: Segmentation
 - Class 8: Local descriptors
 - Class 9: Shape from X
 - Class 10: Recognition and deep learning
 - Class 11: Biomedical image analysis (Dr. Falk)
-
- Computer Graphics (Prof. Teschner)

Other courses in image processing

3. Semester	4.	5. Semester	6. Semester	7. Semester	8. Semester	9. Semester	10.
	Optimierung	Kursvorlesung Bildverarbeitung u. Computergraphik	Spezialvorlesung Statistische Mustererkennung	Spezialvorlesung Computer Vision			
		Spezialvorlesung Engineering meets Biology					
			Seminar	Seminar	Seminar	Seminar	
	Deep Learning	Bachelor Thesis		Deep Learning	GPU Programming	Master project	Master project
						Interdisciplinary Project	Interdisciplinary Project
							Master Thesis

Computer Vision is part of the focus “Cognitive Technical Systems”

Also see courses in

Robotics (Burgard), Machine Learning (Boedecker, Hutter),
Optimization (Diehl)

- Image processing, especially computer vision, is an important research field.
- Thanks to deep learning it is currently growing enormously.
- Image processing makes use of techniques from various other sciences.
- Especially machine learning is closely related.

- R. Szeliski: Computer Vision: Algorithms and Applications, 2010. Available online for free:
<http://szeliski.org/Book/>
- D. Forsyth, J. Ponce: Computer Vision: A Modern Approach, Prentice Hall, 2nd edition, 2012.
- R. C. Gonzalez, R. E. Woods: Digital Image Processing, Addison-Wesley, Reading, 2nd Edition, 2002.
- CV Online: Online compendium on numerous image processing and computer vision topics,
<http://homepages.inf.ed.ac.uk/rbf/CVonline/>

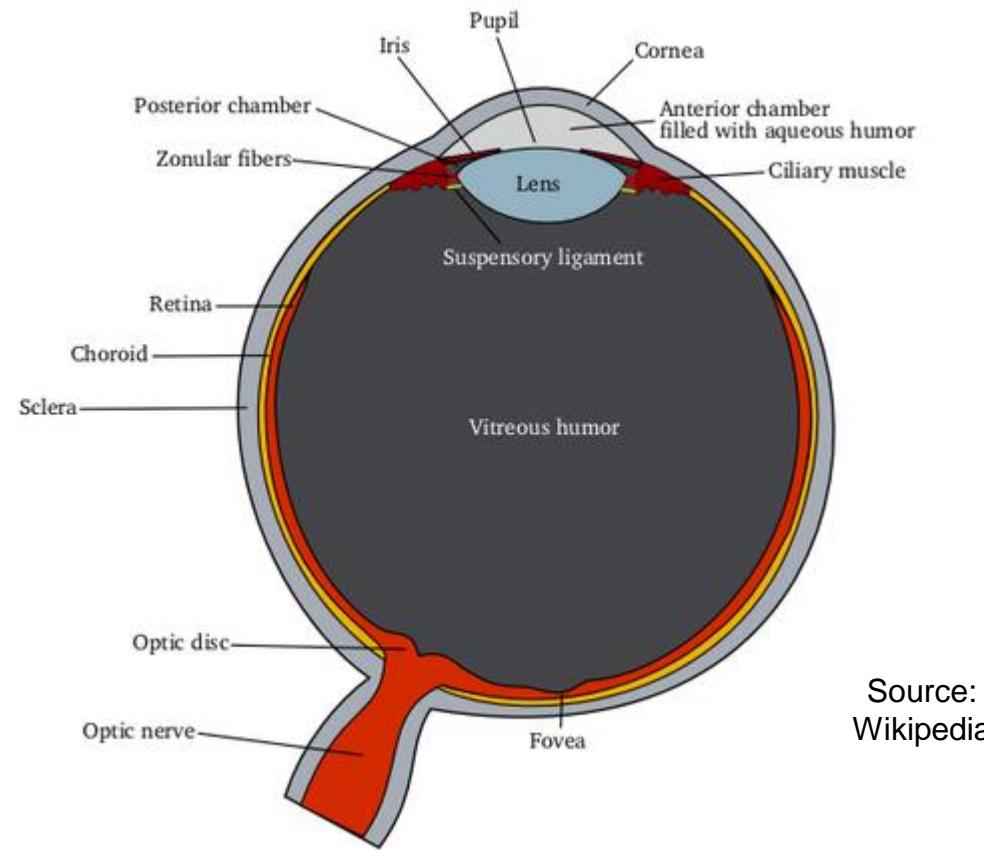
Image Processing and Computer Graphics

Image Processing

Class 2

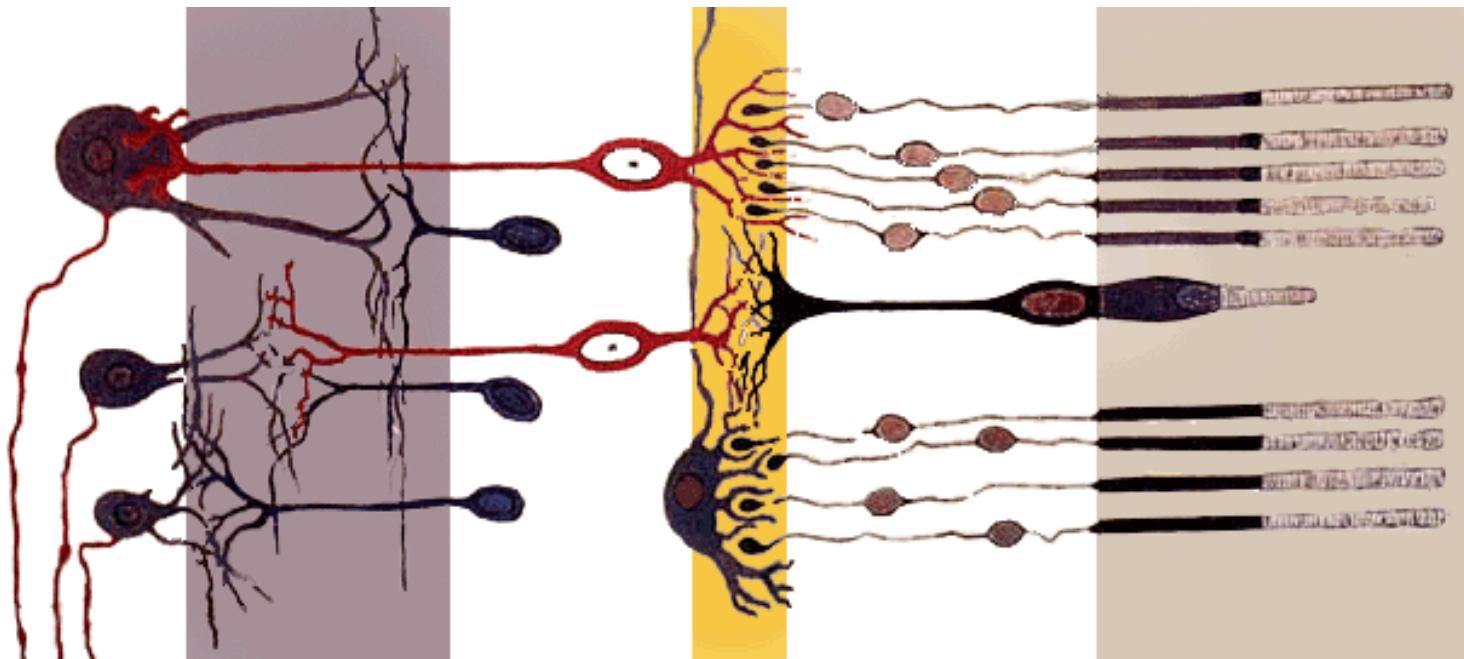
Human vision and image basics

The human eye



- Looks crude, but it's very well optimized
- Especially the ability to adapt to changing light intensities is noteworthy

Front



Back

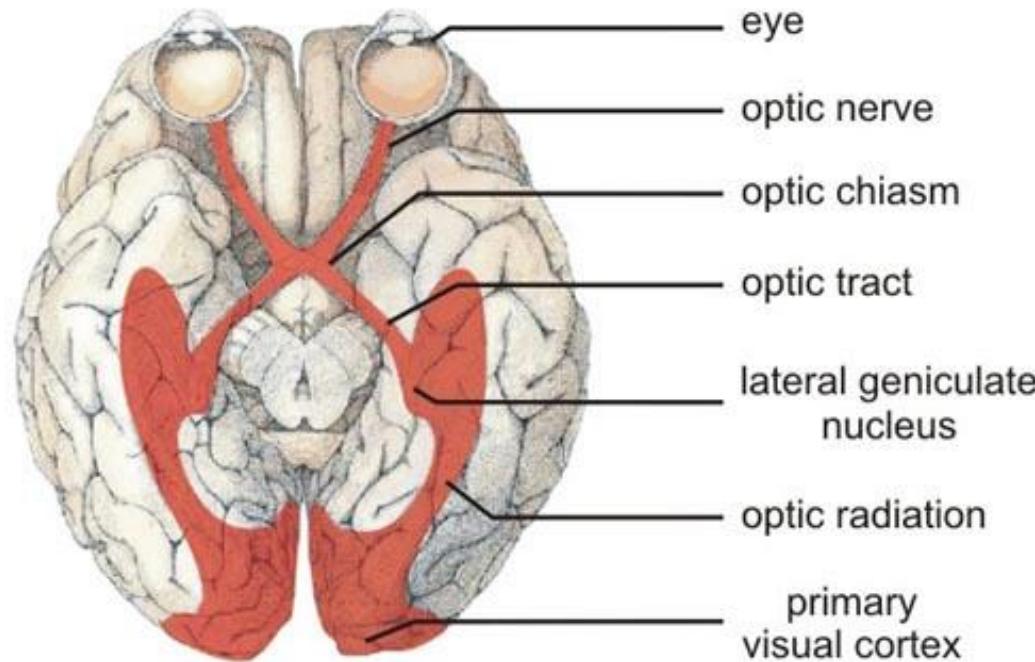
Source: Wikipedia

- Cones (color vision) and rods (gray, high sensitivity)
- Most cones near the optic center
- A lot of analog preprocessing is going on here (mainly smoothing and edge detection)
- Sharp image only in the center (very different from digital images)

What do you think?

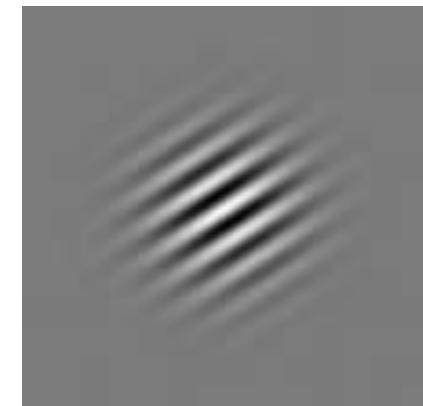
- The retina significantly reduces the amount of data before it is sent to the brain. In the visual cortex the data is then expanded again.

Can you imagine why?

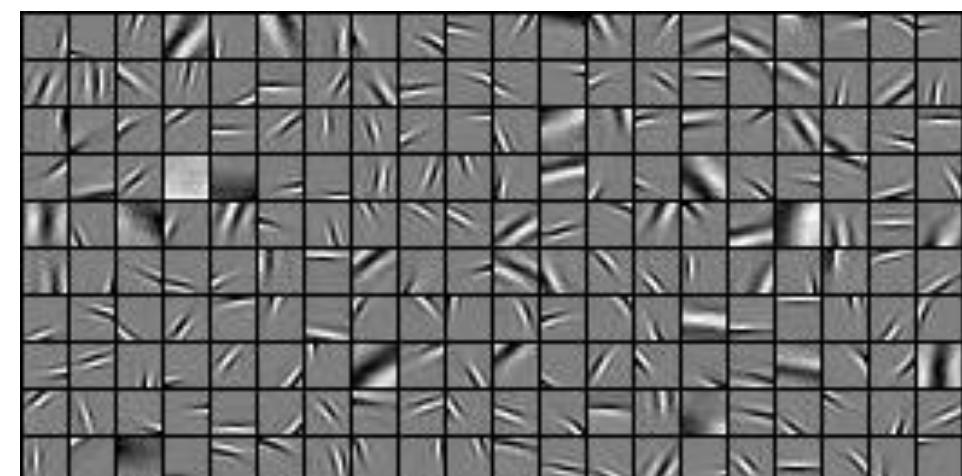


The primary visual cortex (V1)

- The brain's representation is “digital” – spiking or not spiking
- Detailed information coded in spike rate and timing
- Orientation selective cells
 - Selective response to stripes of certain orientation and scale
 - Correspond to Gabor filters
 - Most basic features in an image
 - Found by Hubel-Wiesel 1959 (in cat, later in primates)
- Sparse coding
 - Over-complete code (many different basis functions)
 - Only few are active when presenting a certain signal

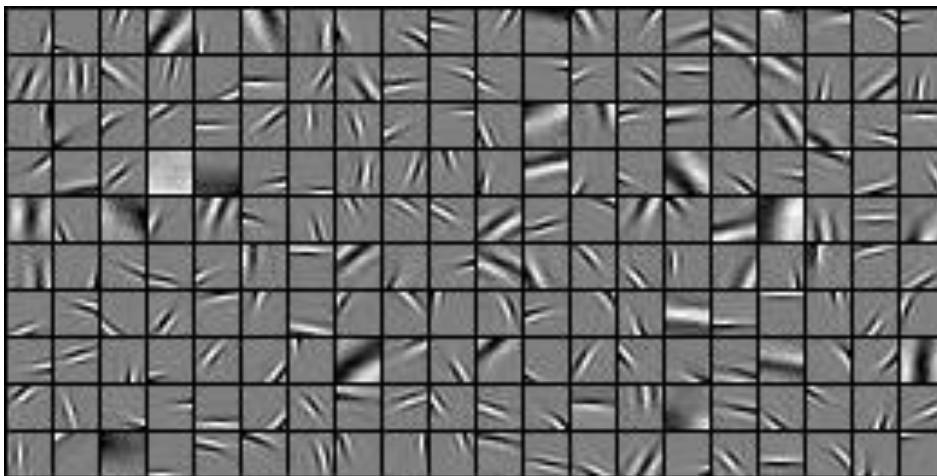


Gabor filter

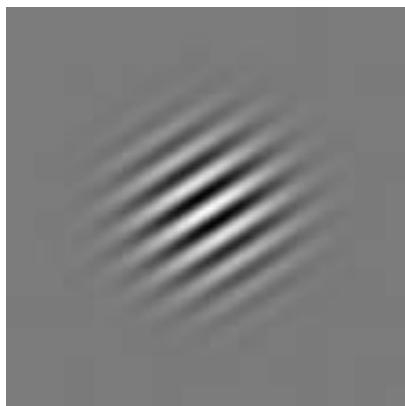


Basis functions of sparse coding model
(Author: Bruno Olshausen)

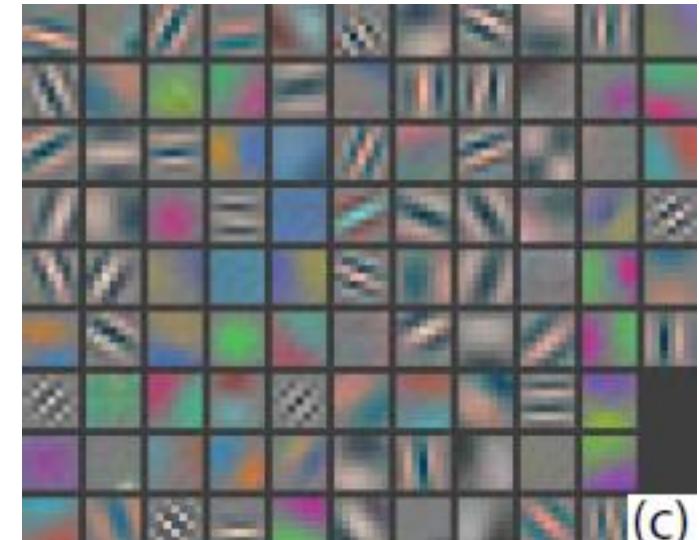
It's well understood how V1 works



Basis functions of sparse coding model
(Author: Bruno Olshausen)



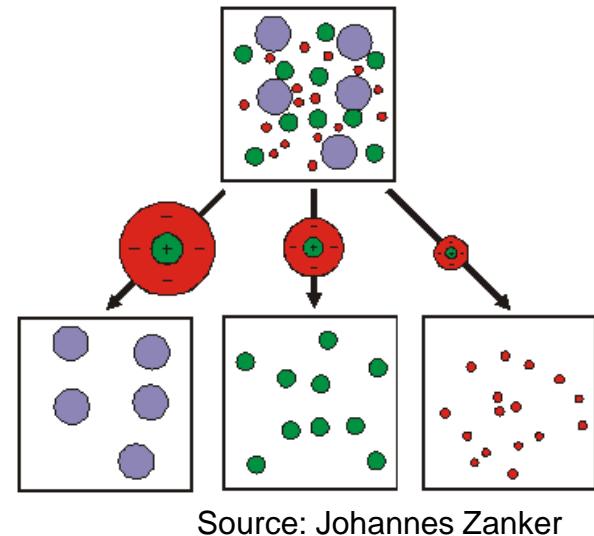
Gabor filter



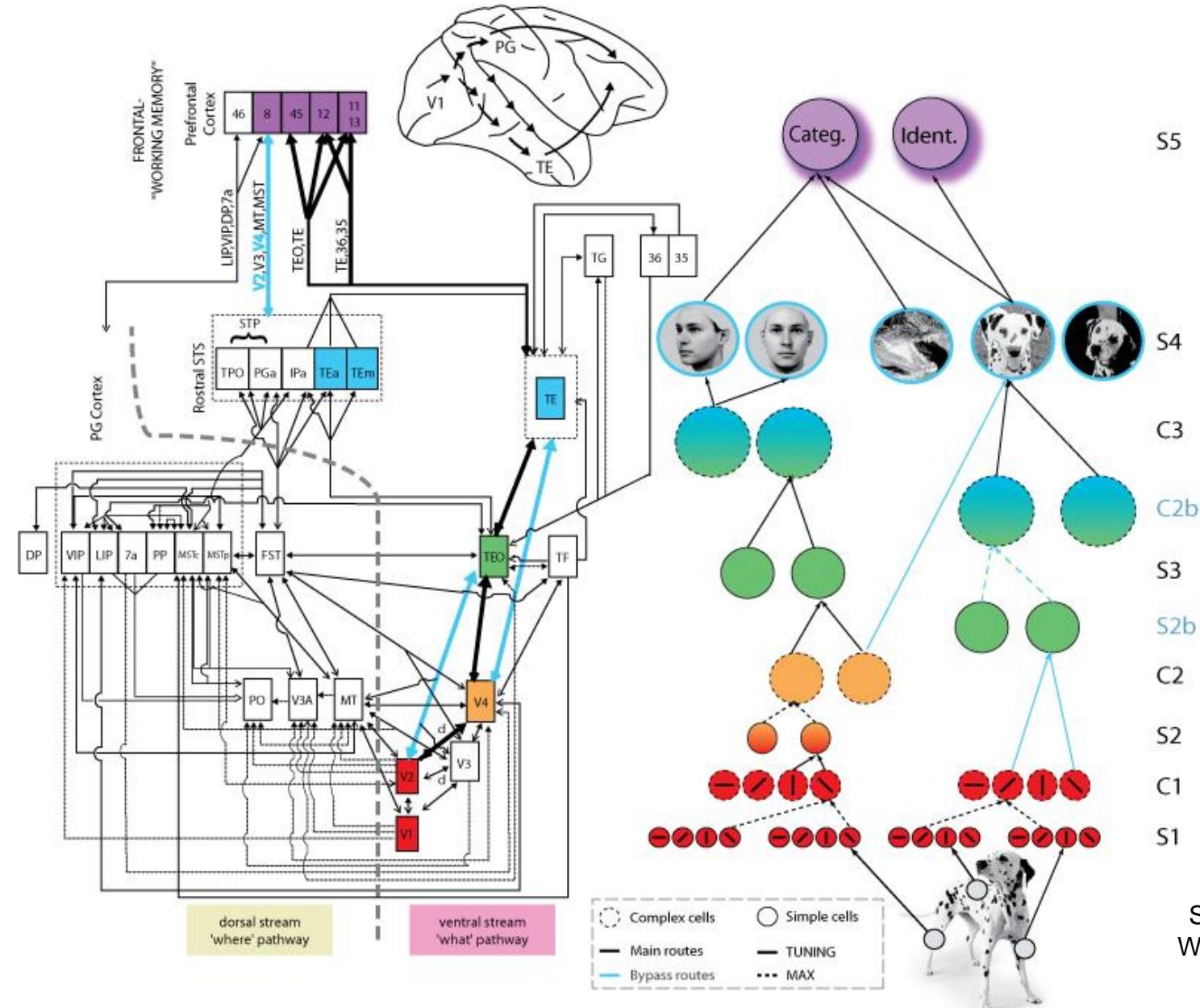
First layer filters of a deep
network trained on image
classification
(Author: Matthew Zeiler)

Receptive fields and integration of information

- In lower cortical areas (especially V1), neurons respond only to signals in a very small local area (**receptive field**)
- In higher cortical areas, the receptive fields get larger and larger due to integration of information from lower areas
- How this integration works, how it is learned, is interesting in both neuroscience and computer science
- Part of machine learning concerned with this problem (deep learning)
- From neuroscience we know which brain areas are responsible for which tasks, and how they coarsely communicate with each other



“System overview”

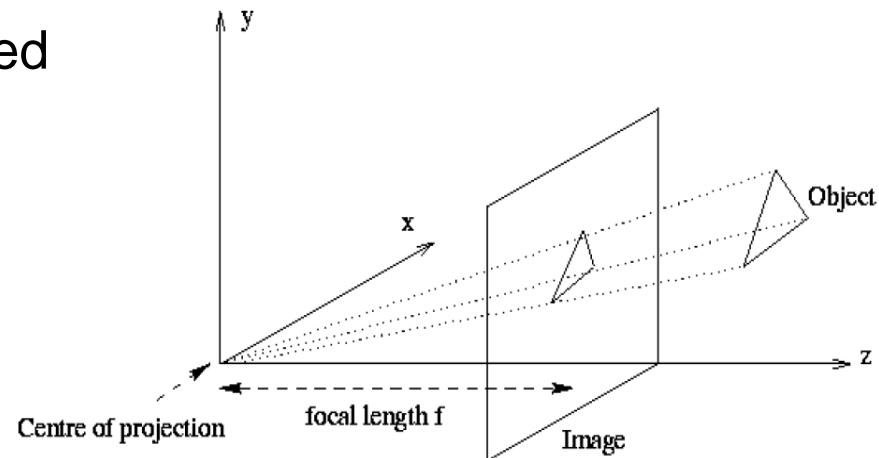


Imaging model: pinhole camera

- Objects points (X, Y, Z) are projected to image points (x, y) by

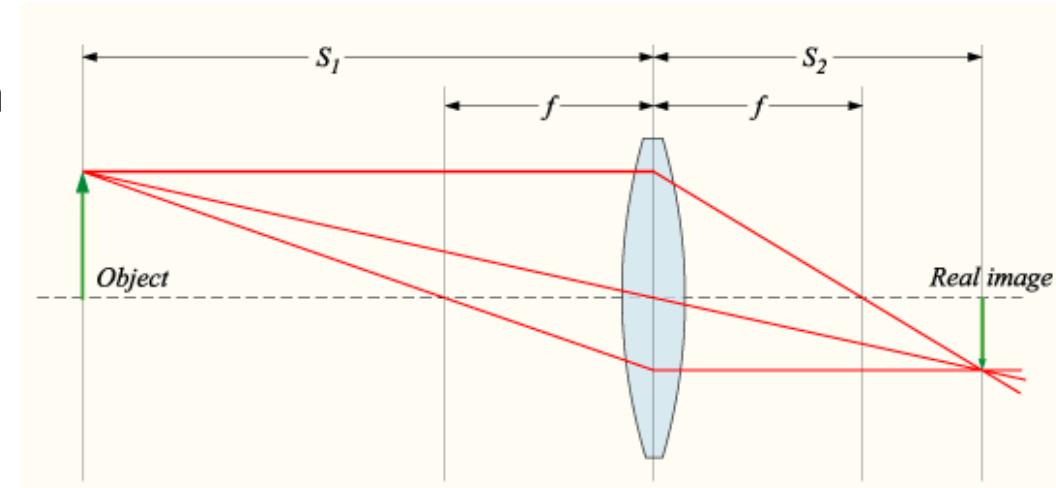
$$x = f \frac{X}{Z} \quad y = f \frac{Y}{Z}$$

- Simplified camera model
- Practical problem:
sharpness vs. light intensity
- Ratio steered by **aperture** (size of the hole)



Optical camera

- Light focusing (solves problem of pinhole camera)
- Large aperture and sharpness possible at a certain depth
- Thin lenses: $\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}$
- Wide lenses: focal length depends on orientation and color
- Computer vision practice: pinhole camera + correction of lens effects
- Some exceptions, where the effect of lenses is more important:
 - Shape from defocus
 - Image analysis in microscopy
 - Wide-angle cameras



Fisheye effect of wide-angle cameras



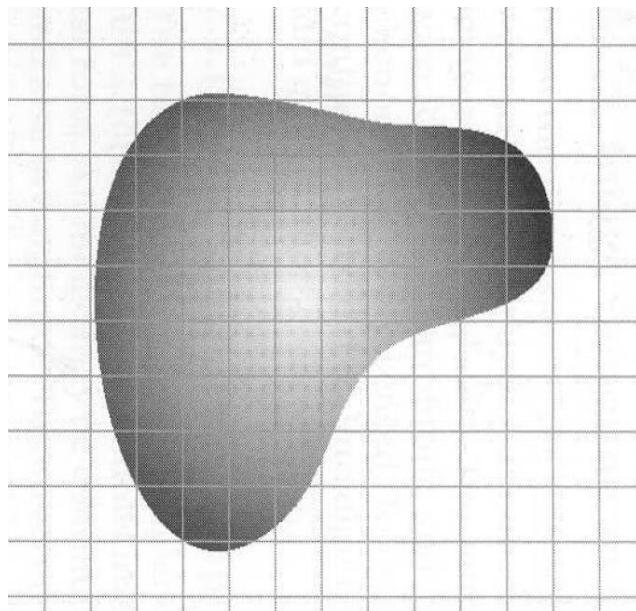
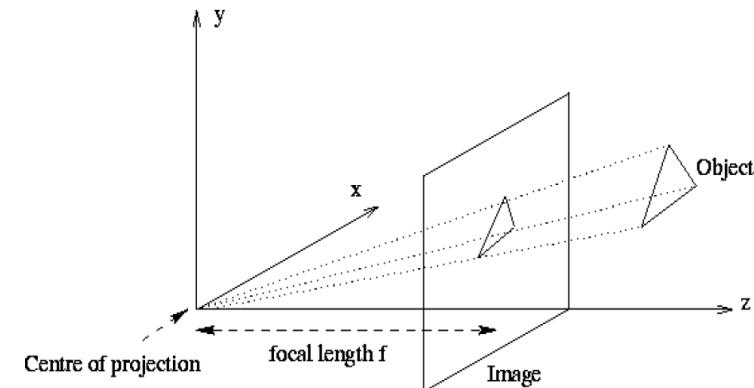
Image representation: gray value images

- Continuous 3D world projects light intensities to a 2D plane

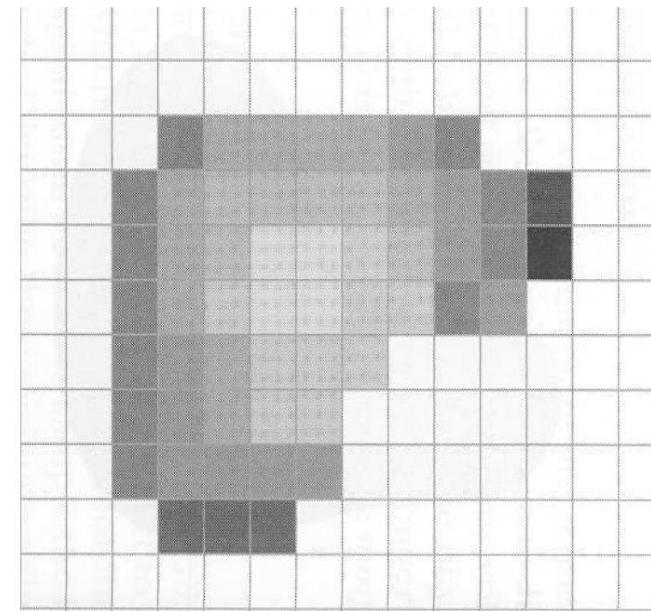
- Image is a continuous function

$$I : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}$$

- Ω is called **image domain**; it is usually rectangular



Continuous image



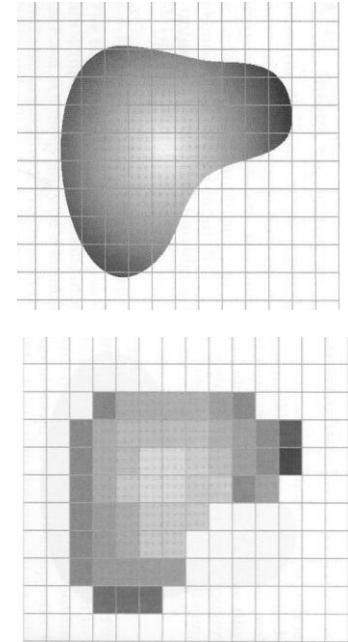
Discrete, sampled image

Sampling

- In digital images, intensities are only given on a pixel grid (e.g. grid of a CCD chip)

$$\{I_{ij} | i = 1, \dots, N; j = 1, \dots, M\}$$

- Discretization of the image domain
- Grid points are called **pixels** (picture elements)
- Grid is usually a rectangular point grid with equal spacing
- **Grid size h** defines the spacing of pixels
- Often same spacing in all directions (square pixels): $h = h_x = h_y$
- If true spacing not known → grid size of input image set to $h = 1$



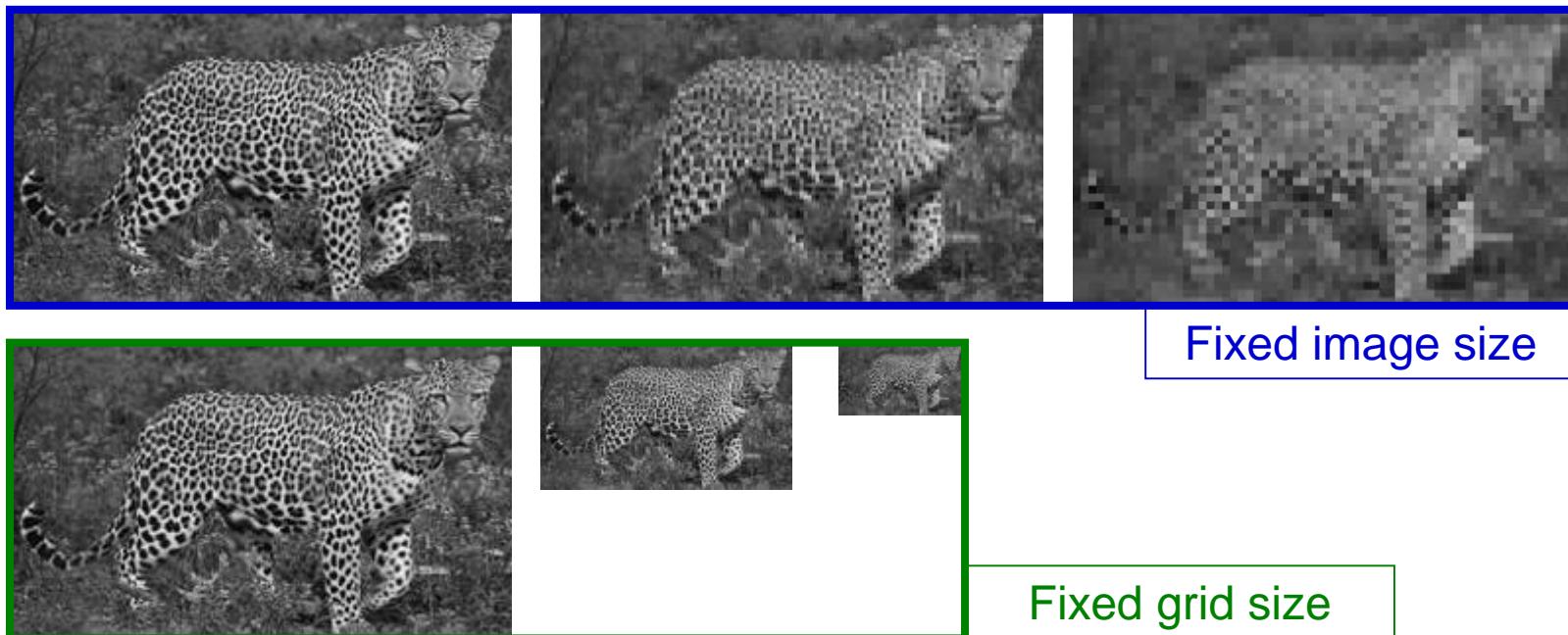
What do you think?

- What are the pros and cons for representing images as continuous functions rather than discrete grids?

- Advantages of continuous models
 - The scene parameters are continuous
 - The continuous model is the limiting case for successively finer grids
 - It ensures certain properties (e.g. rotational invariance, exact length measurement)
 - Certain details (subpixel accuracy) can only be recovered with continuous models
- Reasons for discrete models
 - The sensor data is discrete
 - Model is closer to its implementation
- No clear winner → computer vision works with both paradigms

Image resolution, downsampling, upsampling

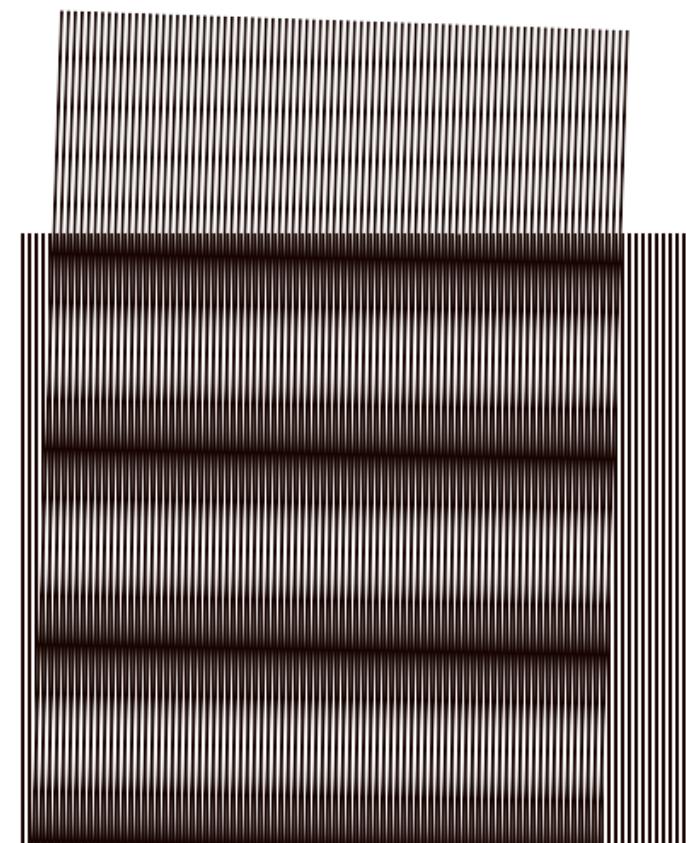
- Given a certain image of a scene, the number of grid points to represent the discrete image is called the **image resolution**
- Reducing the number of grid points is called **downsampling**



- Increasing the number of grid points is called **upsampling**

Aliasing and Moiré effect

- A function can be represented by its frequency components (Fourier transform)
- A discrete signal can only represent frequencies up to a certain limit
→ **Nyquist frequency**
- Ignorance of the Nyquist frequency leads to **aliasing** artifacts (e.g. straight lines become stepped)
- Sampling of periodic signals is a frequency modulation (multiplication of two periodic signals)
- It can lead to Moiré effects (a special aliasing artifact)
- Videos can exhibit temporal aliasing



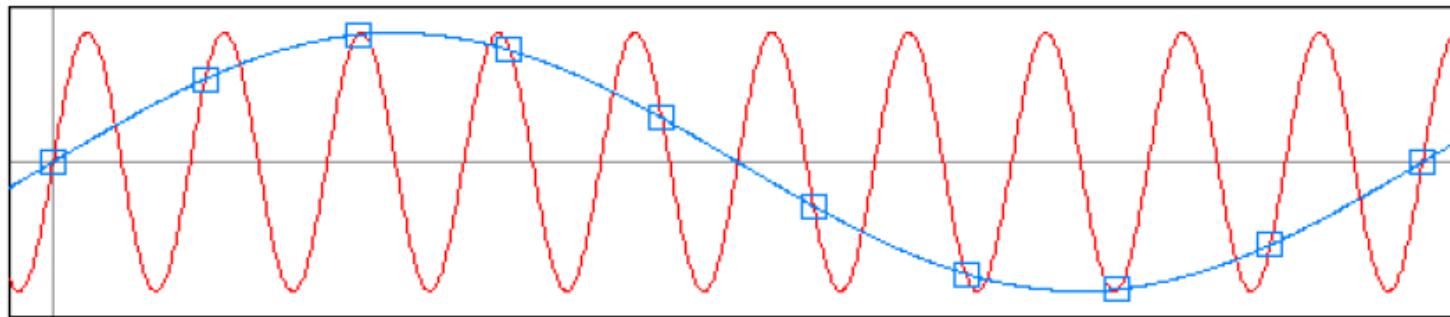
Moiré effect, Source: Wikipedia

Spinning wheel example for temporal aliasing



Nyquist-Shannon theorem

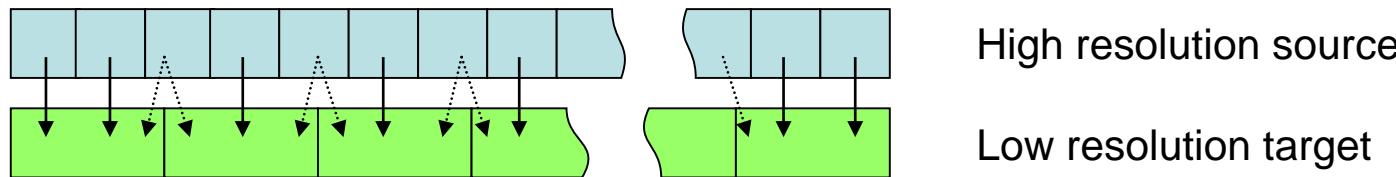
- Two different frequencies may lead to the same discrete signal



- An input signal can be reconstructed from samples in a unique way if the sampling rate is at least two times the bandwidth of the input signal
- Reduction of bandwidth (= maximum frequency) can be achieved by smoothing the signal
- Consequence: smooth your input image sufficiently before downsampling

Downsampling

- Decanting operator: ensures minimum smoothing necessary to avoid aliasing



- Pixels from high resolution image spill their intensity to the pixels of the low resolution image
- In case of overlap, intensity distribution to both cells according to the overlap ratio
- Normalization of intensity value in the low resolution image by the downsampling factor
- Operator is **separable**: can be applied sequentially along all axes

Upsampling, bilinear interpolation

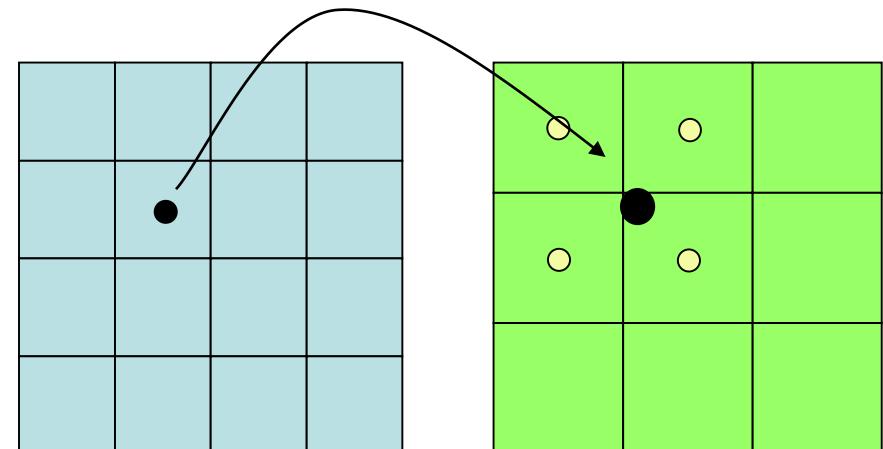
- Bilinear interpolation: weighted average of neighboring pixels

- Project fine grid point to available coarse grid

- Compute weighted average along x-axis

$$a_j = (1 - \alpha)I_{i,j} + \alpha I_{i+1,j}$$

$$a_{j+1} = (1 - \alpha)I_{i,j+1} + \alpha I_{i+1,j+1}$$



- Compute weighted average along y-axis

$$I_{k,l} = (1 - \beta)a_j + \beta a_{j+1}$$

- General concept to retrieve values at points between grid points

- Can be extended to arbitrary dimensions (trilinear interpolation)

- Discretization of the co-domain $\mathbb{R} \mapsto \{1, \dots, N\}$
- Needed for representation in the computer (float).
Integer representation is more common.
- Usual image formats have 256 gray scales \rightarrow 8 bit per pixel (bpp)
(often more in microscopy and industrial cameras)
- Humans can distinguish only 40 gray scales (but several thousand colors)
- Optimal quantization by clustering (e.g. k-means)
- We will usually assume $I(x, y) \in \mathbb{R}, 0 \leq I(x, y) \leq 255$

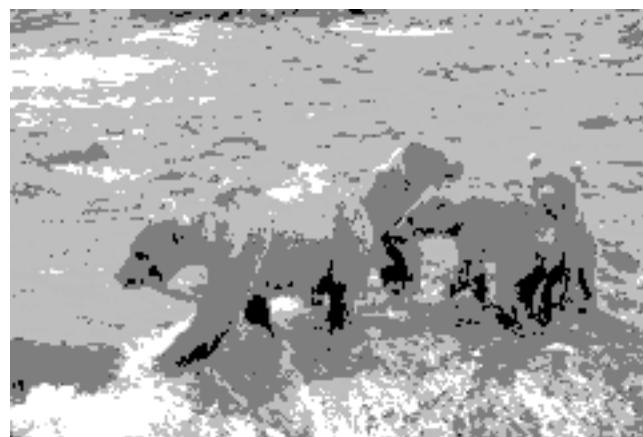
Quantization



256 scales (8 bpp)



16 scales (4 bpp)



4 scales (2 bpp)

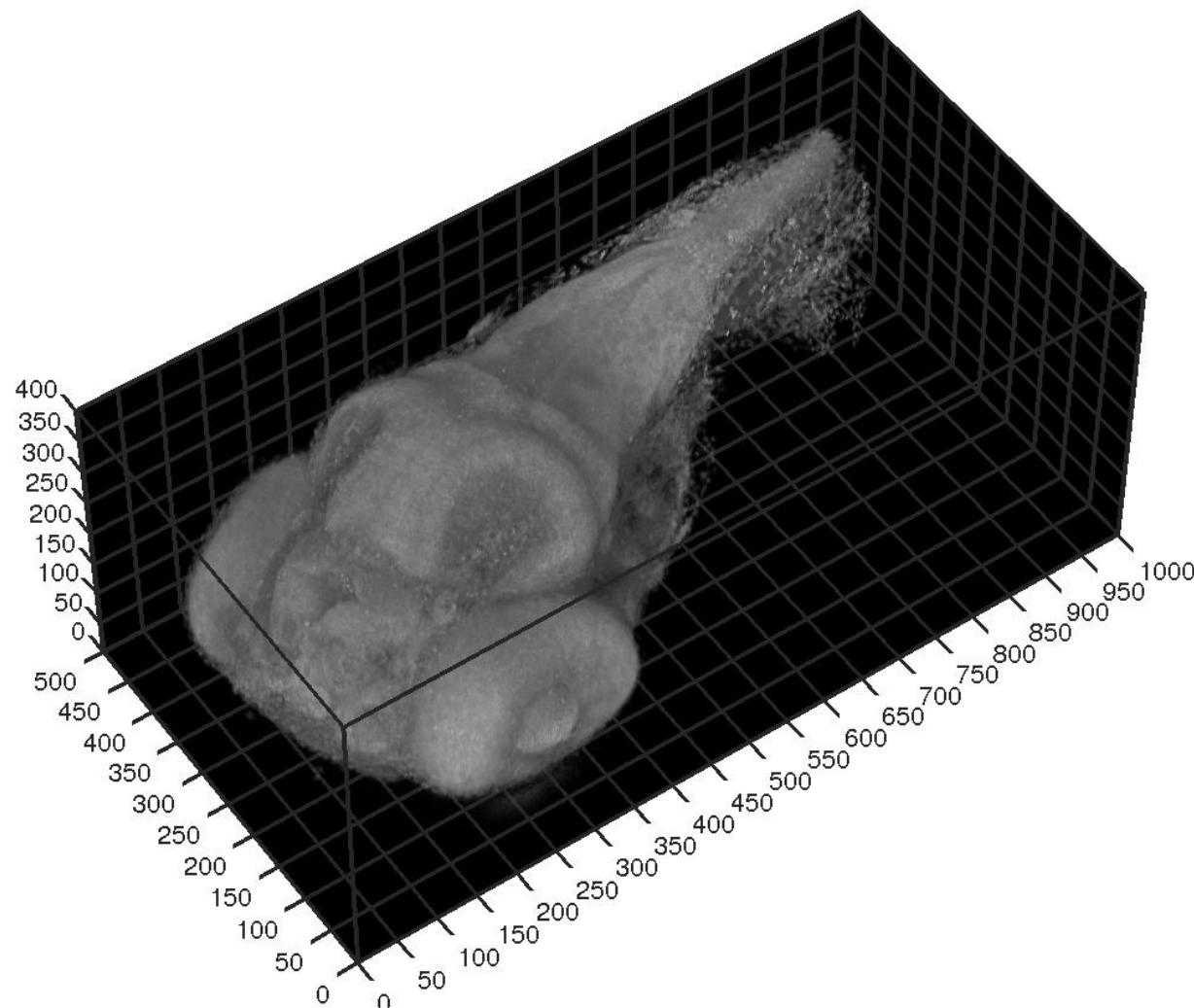


2 scales (binary image)

Types of images

- Generalization of images with respect of
 - dimensionality of the image domain
 - dimensionality of the co-domain
- “Standard” images: image domain is two-dimensional, co-domain one-dimensional (gray scale)
- Other dimensionalities of the image domain:
 - 1D signals
 - 3D images (volumetric images, image sequences)
 - 4D images (sequence of volumetric images)
- Other co-domains
 - Vectors (e.g. color images)
 - Matrices

Volumetric 3D data



Volumetric dataset obtained with a confocal microscope showing a zebrafish larvae

Image sequences



Color images



Color image



Red channel

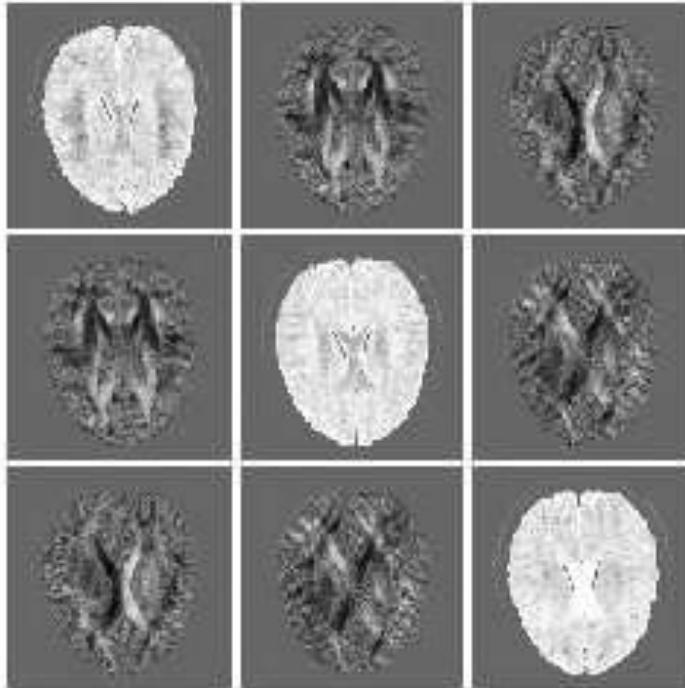


Green channel

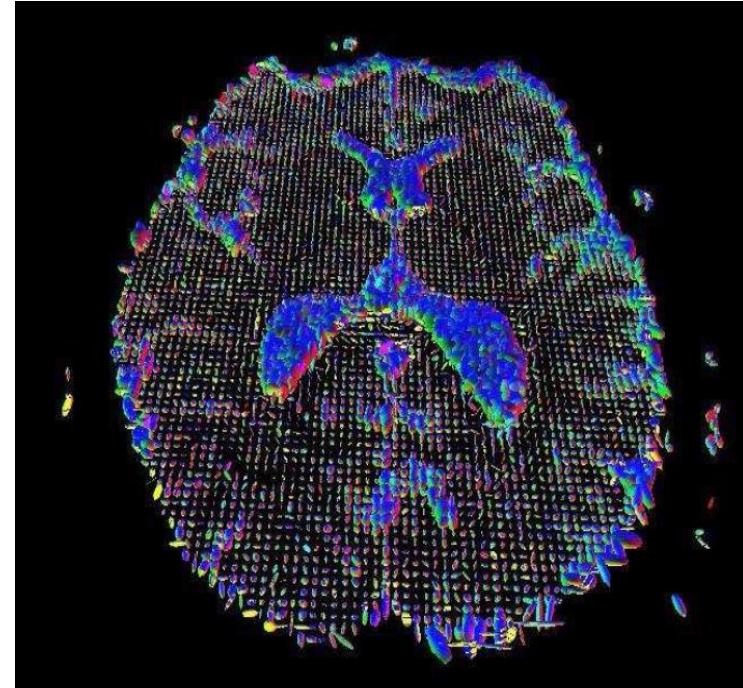


Blue channel

Matrix-valued images



Matrix channels of DT-MRI
Weinstein-Kindlmann-Lundberg 1999



3D visualization via ellipsoids
Data: Anna Villanova, BMT, Eindhoven
Visualization: MIA group, Saarland University

- Diffusion tensor MRI: flow preferences of water molecules
- Each voxel (volume element) comprises a 3×3 matrix

- The human visual system shows great performance but is only partially understood
- Images are projections of the real, continuous world and therefore continuous functions
- Digital images are discrete approximations of these functions
- (Down)sampling of images requires smoothing/averaging to avoid aliasing artifacts
- There are various types of images. Often (but not always) they can be processed by the same algorithms.

Image Processing and Computer Graphics

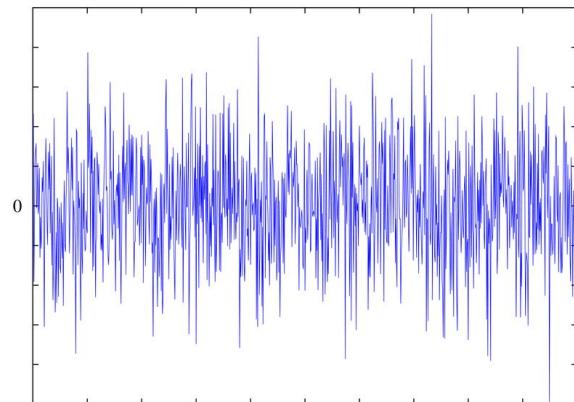
Image Processing

Class 3

Noise, basic operations, and filters

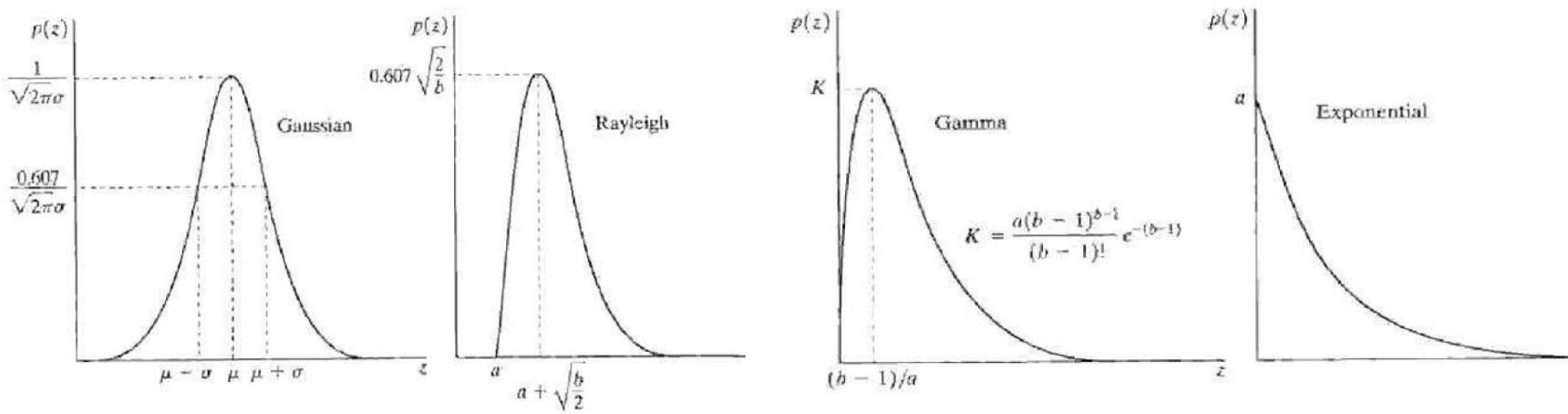
What is noise?

- Noise is a disturbance of the image data by image acquisition or the transmission of images
- Sometimes the term noise is also used more generally for the component of the data that does not fit the underlying model (model noise)
- One classical goal of image enhancement: removal of noise
- Computer vision: rather than removing noise, choose a model that can deal with noise



Additive noise

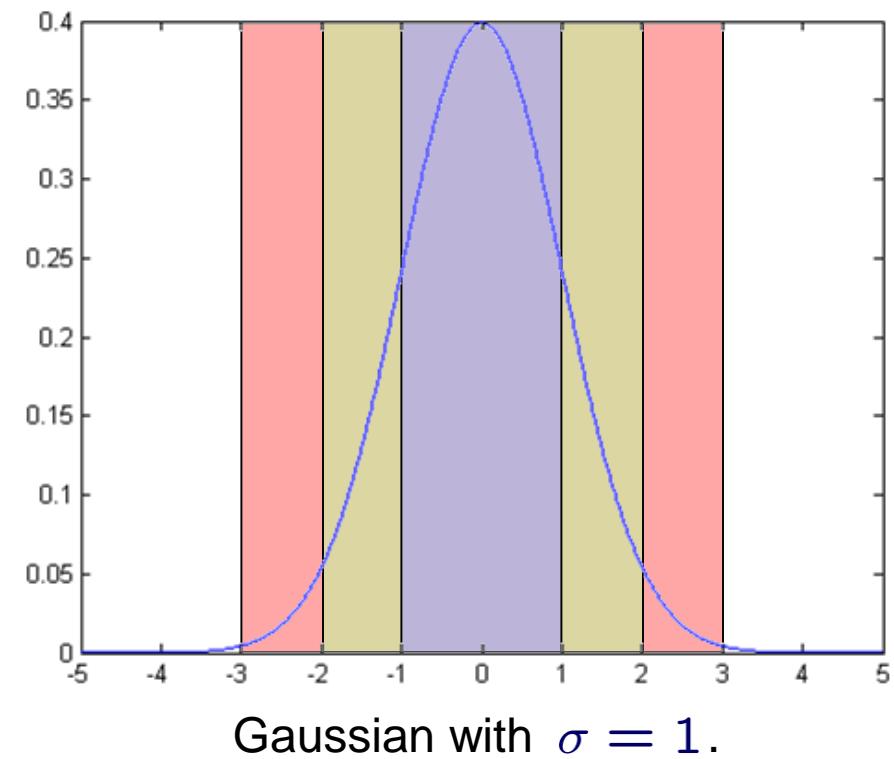
- Assumption: gray values and noise are independent: $I_{ij} = I_{ij}^* + n_{ij}$
- Noise distribution depends on the sensor
 - Poisson noise, Gaussian noise (CCD cameras)
 - Rayleigh distribution (radar)
 - Gamma distribution (laser imaging)
 - Exponential distribution (laser imaging)



From Gonzales-Woods 2002

Gaussian noise and Gaussian distribution

- Density function $G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- Usually zero-mean Gaussian noise
- Good approximation in many practical situations
- In particular: thermic sensor noise in CCD cameras
- 1σ interval: 68% of values
- 2σ interval: 95.5% of values
- 3σ interval: 99.7% of values



Gaussian with $\sigma = 1$.

Gaussian noise



Lena test image without noise



Gaussian noise with $\sigma = 20$ added

- Signal dependent: $I_{ij} = I_{ij}^*(1 + n_{ij})$
- More difficult to handle in image enhancement algorithms
- Useful trick:

- Transform the image by applying the logarithm

$$\log I_{ij} = \log I_{ij}^* + \log(1 + n_{ij})$$

- Noise becomes additive noise (can be removed by standard algorithms)
 - Apply backtransform to denoised image

$$I_{ij}^* = \exp \log I_{ij}^*$$

Impulse noise

- A certain percentage of pixels is replaced by one (unipolar impulse noise) or two (bipolar impulse noise) fixed values
- Caused, for instance, by pixel defects of CCD chips
- Special case salt-and-pepper noise: some pixels replaced by white or black values



Original



5% salt-and-pepper
noise



20% salt-and-pepper
noise

Uniform noise

- A certain percentage of pixels is replaced by uniformly distributed random variables
- Very unpleasant noise, no a-priori knowledge in the noise model



Original



5% uniform noise



20% uniform noise

Evaluation, Signal-to-noise ratio (SNR)

- Quantitative measure for the degradation of an image I versus a noise-free version I^* (**ground truth**)
- Based on the variance of the image versus the variance of the noise

- Variance of the image: $\sigma_I^2 = \frac{1}{N} \sum_i (I_i^* - \mu)^2$

- Additive, zero-mean noise model: $I_i = I_i^* + n_i$

- Variance of the noise: $\sigma_n^2 = \frac{1}{N} \sum_i (I_i^* - I_i)^2$

- Signal-to-noise ratio:** $\text{SNR} = 10 \log_{10} \left(\frac{\sigma_I^2}{\sigma_n^2} \right) = 10 \log_{10} \left(\frac{\sum_i (I_i^* - \mu)^2}{\sum_i (I_i^* - I_i)^2} \right)$

- Peak signal-to-noise ratio:** $\text{PSNR} = 10 \log_{10} \left(\frac{N(\max_i I_i^* - \min_i I_i^*)^2}{\sum_i (I_i^* - I_i)^2} \right)$

- Their unit is decibel (dB), the higher the better

Peak signal-to-noise-ratio



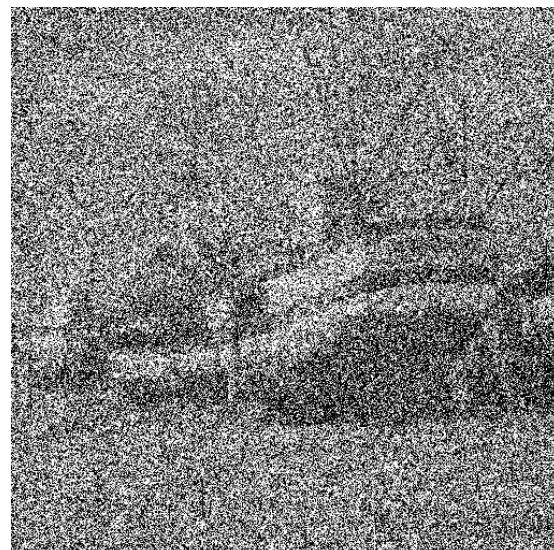
Noise free image



$\sigma_n = 10$, PSNR = 28.15



$\sigma_n = 2$, PSNR = 42.1



$\sigma_n = 200$, PSNR = 7.63

Point operations

- The most simple way to enhance an image is to treat each pixel independently:

$$u_{ij} = f(I_{ij})$$

- This kind of operation mainly transforms intensities in a way that relevant structures are in a range that can be well observed.
- Example: changing brightness

$$u(x, y) = I(x, y) + b$$

- Darkening for $b < 0$
- Values that exceed the allowed range must be clipped

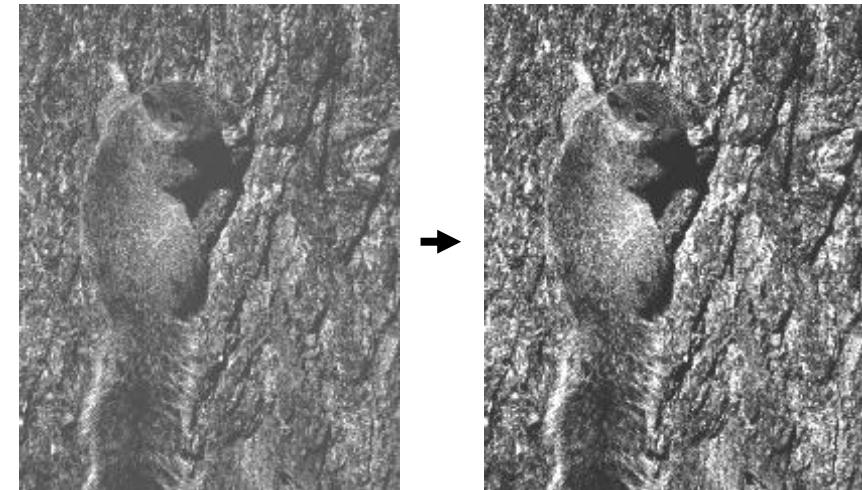


Contrast enhancement and Gamma correction

- Contrast enhancement

$$u(x, y) = aI(x, y)$$

- $a > 1$
- Contrast attenuation for $a < 1$
- Clipping of values that exceed the allowed range

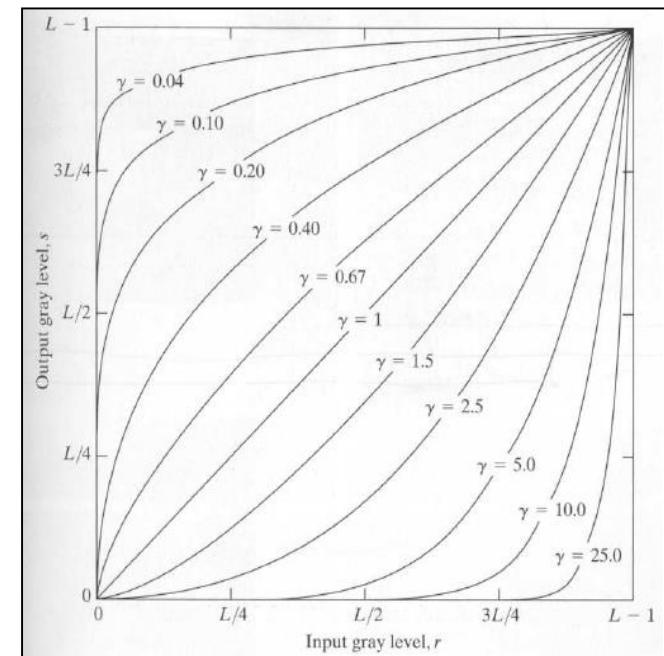


- Gamma correction

- Camera chips have different response curves than the human eye, usually $I \propto I^\gamma$
- Compensation of these effects by a gamma correction

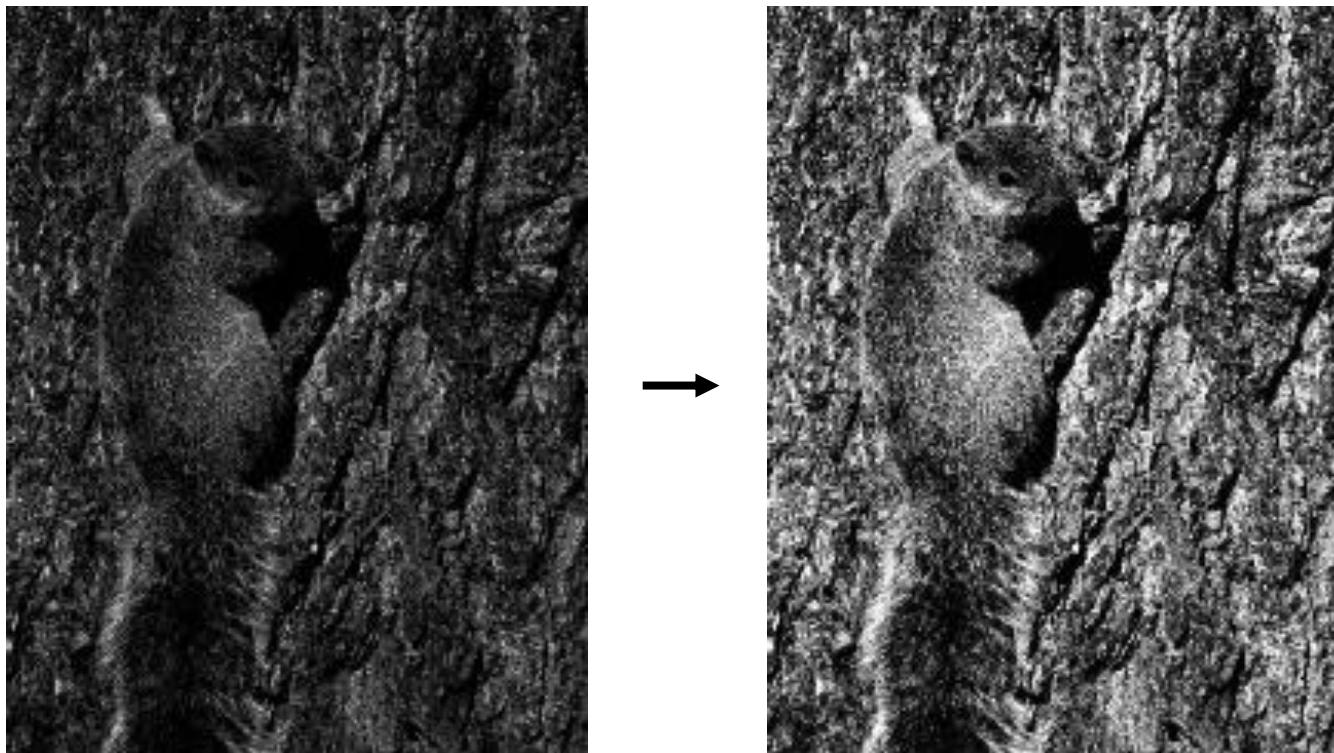
$$f(I(x, y)) = I_{\max} \left(\frac{I(x, y)}{I_{\max}} \right)^{\frac{1}{\gamma}}, \quad \gamma > 0$$

- The range $[0, I_{\max}]$ is not affected



From Gonzales-Woods 2002

Gamma correction



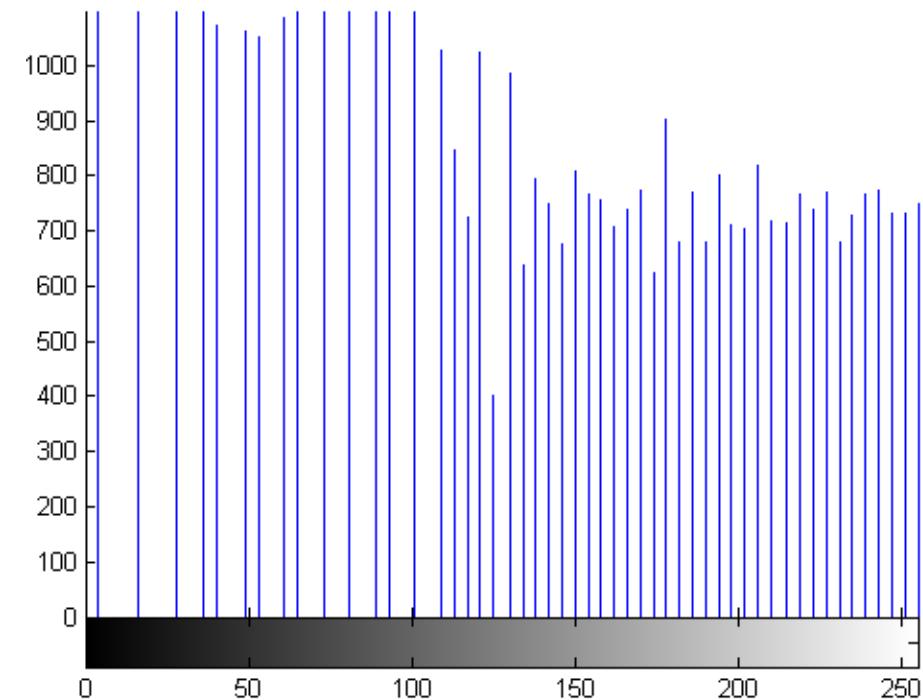
Dark areas become brighter without leading to oversaturation in the brighter areas

Gray value histogram

- The gray value **histogram** contains the number of pixels in the image that have a certain gray value



Input image



Histogram

- Histogram equalization: transformation such that all gray values are equally frequent

Difference image

- Subtracting the grayvalues of two images from each other yields a difference image:

$$I_{\Delta} = |I_1 - I_2|$$

- Can be used for detecting parts of moving objects in static scenes



Input images



Difference image



Difference image after
thresholding

Background subtraction

- Special difference image: **background subtraction**
 - Take one image of the static background without the object
 - Difference image to this background image indicates the object
 - Can be used for object tracking and segmentation of a person in front of a static background



- Difference image of color images: $I_{\Delta} = \frac{1}{3} \sum_{k=1}^3 |I_{k,1} - I_{k,2}|$



Input images



Difference image



Difference image after thresholding

Linear filters, convolution theorem

- Filters take neighboring pixels into account to improve the signal
- In signal processing, filters are often designed in the Fourier domain
→ global frequency analysis, takes all pixels into account
- Filtering in the Fourier domain can be translated to filtering in the spatial domain via the **convolution theorem**:

$$\mathcal{F}(f) \cdot \mathcal{F}(h) = \mathcal{F}(f * h)$$

↑
Fourier transform

- In image processing, we usually design the filter h in the spatial domain instead of $\mathcal{F}(h)$ in the Fourier domain because:
 - More interested in a local analysis
 - Better handling of image boundaries
 - Easier to generalize to nonlinear filters

Convolution

- Convolution with a static filter h is a **linear operation**
→ linear filtering

$$(f * h)(x) := \int h(-x') f(x + x') dx'$$

- Convolution in 2D

$$(I * h)(x, y) := \int h(-x', -y') I(x + x', y + y') dx' dy'$$

- Some general properties of convolution:

- Linearity $(\alpha f + \beta g) * h = \alpha(f * h) + \beta(g * h), \quad \alpha, \beta \in \mathbb{R}$
- Shift invariance $f(x) * g(x + \delta) = (f * g)(x + \delta), \quad \forall \delta \in \mathbb{R}$
- Commutativity $f * g = g * f$
- Associativity $(f * g) * h = f * (g * h)$

- Correlation: $f(x) \star h(x) := \int h(x') f(x + x') dx'$

Discrete convolution and separability

- Discrete convolution in 1D:

$$(f * h)_i = \sum_{k=1}^n f_k h_{i-k} \quad i = 1, \dots, N$$

- Discrete convolution in 2D:

$$(f * h)_{i,j} = \sum_{k=1,l=1}^{n,m} f_{k,l} h_{i-k,j-l} \quad i = 1, \dots, N; j = 1, \dots, M$$

- Separability: A filter is separable if

$$h(x, y) = \delta(x, y) * h_1(x, \cdot) * h_2(\cdot, y)$$

where δ is the Dirac distribution

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & \text{else} \end{cases}, \quad \int \delta(x) dx = 1$$

- Separable filters can be implemented via successive 1D convolutions (associativity of convolution). This reduces the computational complexity of the convolution from $O(NMnm)$ to $O(NM(n + m))$

- Most popular (linear) filter for image smoothing
- Convolution with a Gaussian kernel of width σ

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- The Gaussian filter is separable, the smoothed image can be derived as
$$\tilde{I}(x, y) = I(x, y) * G_\sigma(x, \cdot) * G_\sigma(\cdot, y)$$
- Discrete filter stencil usually derived by sampling the continuous Gaussian in the 3σ -interval
- Computational complexity: $O(NM\sigma)$
- Alternative implementation in the Fourier domain is $O(NM \log(NM))$

Gaussian convolution



Input image



$\sigma = 1$



$\sigma = 2$



$\sigma = 4$

Boundary conditions

- The image domain Ω is generally not an infinite domain but has boundaries $\partial\Omega$.

- Different types of boundary conditions

- **Dirichlet boundary conditions:**

$$I(x, y) = 0, \quad (x, y) \in \partial\Omega$$

- **Homogeneous Neumann boundary conditions:**

$$\frac{\partial}{\partial \mathbf{n}} I(x, y) = 0 \quad (x, y) \in \partial\Omega$$

where \mathbf{n} is the outer normal vector on $\partial\Omega$ and $\frac{\partial I}{\partial \mathbf{n}} := \mathbf{n}^\top \nabla I$ is the directional derivative

- Usually Neumann boundary conditions are preferred as they have nicer properties (e.g. preservation of average intensity)
- They are obtained by **mirroring the image** at the boundary

Gaussian convolution – noise removal



Noisy input image



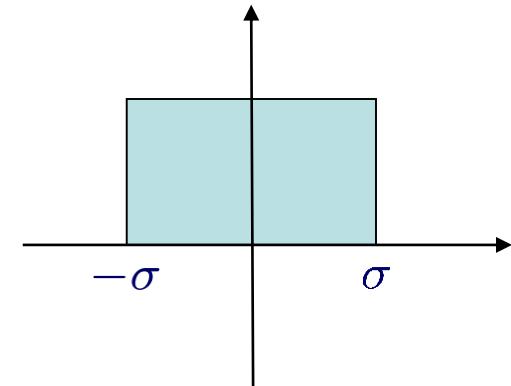
$\sigma = 1$

Satisfied?

Too bad, it removes some of the noise, but also part of the signal
→ Nonlinear diffusion

Why a Gaussian filter and not a box filter?

a) The box filter is not separable



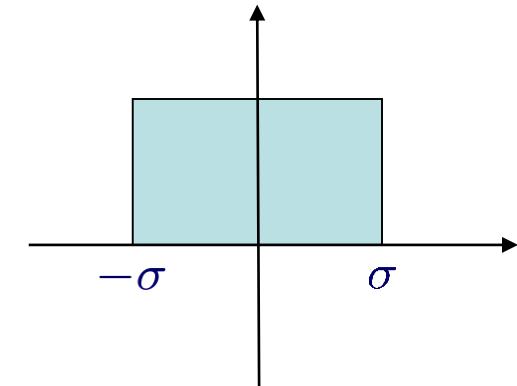
b) The Gaussian filter is rotationally invariant, the box filter is not

c) Of course we prefer a box filter!

Box filter

- Alternative to a Gaussian kernel: box kernel

$$B_\sigma(x) = \begin{cases} \frac{1}{2\sigma} & |x| < \sigma \\ 0 & \text{else} \end{cases}$$



- Simple (unweighted) averaging of neighboring values
- Disadvantages:
 - Result not as smooth (differentiability is increased only by one order)
 - Not rotationally invariant
- Advantage: convolution with a large kernel is much more efficient, since

$$\tilde{f}_{i+1} = \frac{1}{2\sigma} \sum_{j=i+1-\sigma}^{i+1+\sigma} f_j = \frac{1}{2\sigma} \left(\sum_{j=i-\sigma}^{i+\sigma} f_j + f_{i+1+\sigma} - f_{i-\sigma} \right) = \tilde{f}_i + \frac{1}{2\sigma} (f_{i+1+\sigma} - f_{i-\sigma})$$

- Filter is separable
- Complexity $O(NM)$ independent of σ

Box filter



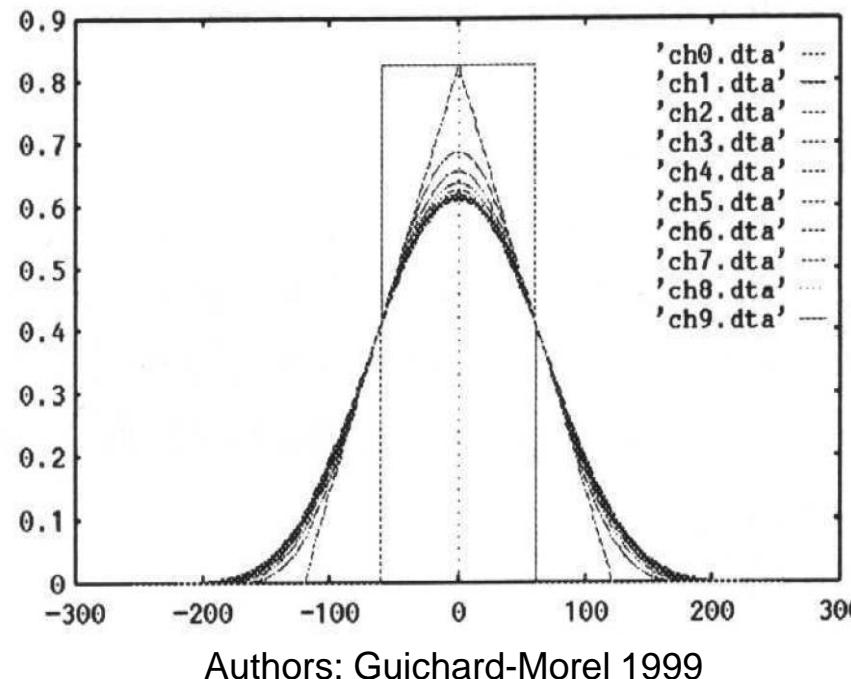
Gaussian filter $\sigma = 4$



Box filter $\sigma = 6$

Relations to Gaussian filter

- Successive convolution of the box kernel with itself yields filters that resemble more and more the Gaussian kernel
- Central limit theorem in statistics: iterated averaging kernels (= positive kernels) converge to Gaussians
- Smoothing result with filter of order k is k -times differentiable



Iterated box filter



Gaussian filter $\sigma = 4$



3 iterations of a box filter

Recursive filter

- Another fast alternative to the Gaussian filter (Deriche 1990)
- Idea: recursively propagate information in both directions of the signal



α : smoothness parameter

$$f_i = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} (I_i + e^{-\alpha}(\alpha - 1)I_{i-1}) + 2e^{-\alpha} f_{i-1} - e^{-2\alpha} f_{i-2}$$

$$g_i = \frac{(1 - e^{-\alpha})^2}{1 + 2\alpha e^{-\alpha} - e^{-2\alpha}} (e^{-\alpha}(\alpha + 1)I_{i+1} - e^{-2\alpha} I_{i+2}) + 2e^{-\alpha} g_{i+1} - e^{-2\alpha} g_{i+2}$$

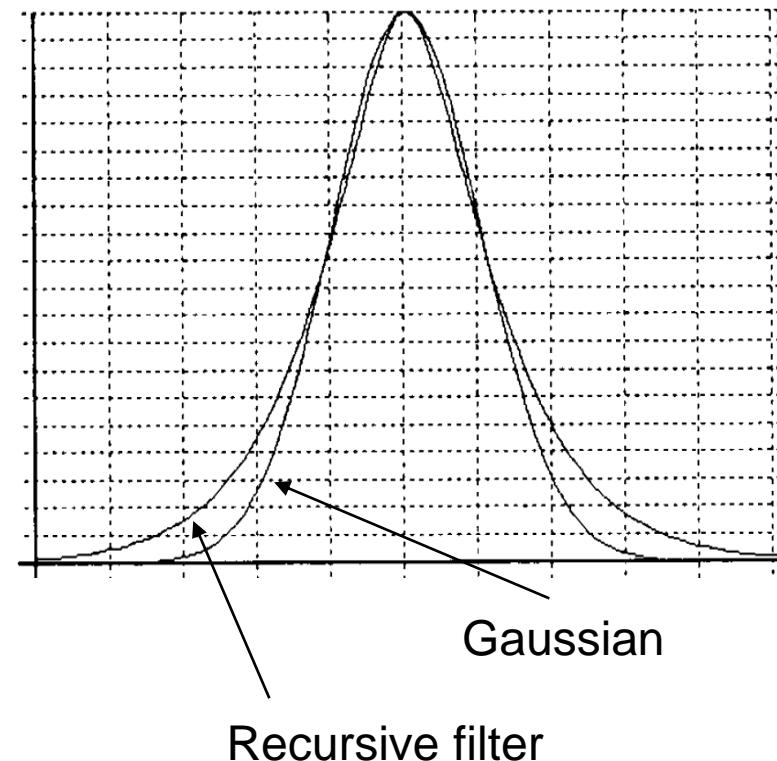
$$\tilde{I}_i = f_i + g_i$$

Recursive filter

- Recursive filter approximates Gaussian convolution
- Relation between α and σ

$$\alpha \cdot \sigma = \frac{5}{2\sqrt{\pi}}$$

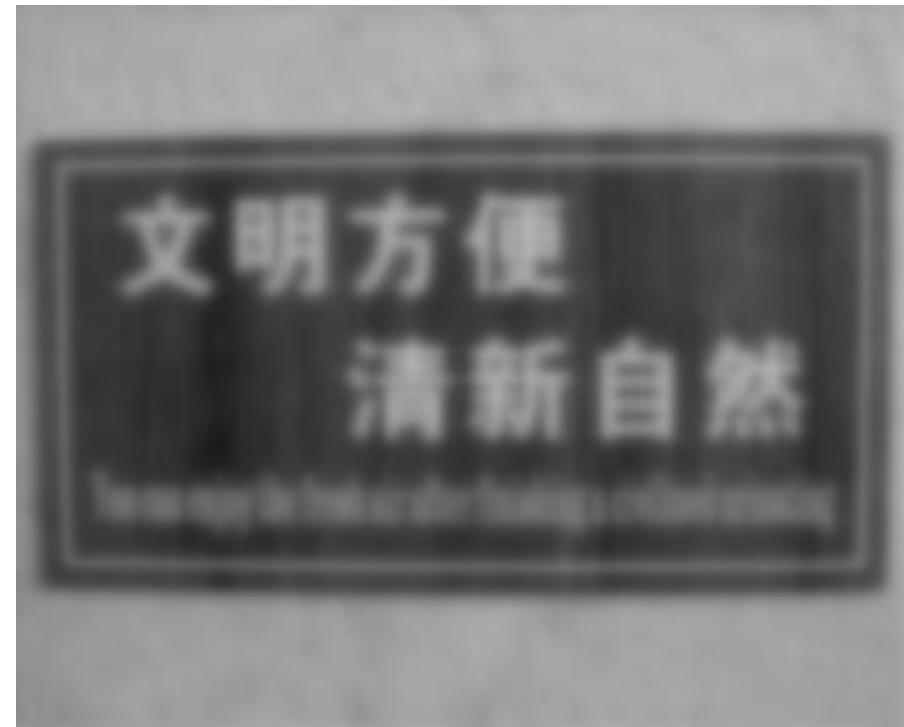
- Filter is separable and rotationally invariant
- Complexity $O(NM)$ is independent of α
- Hard (impossible?) to implement Neumann boundary conditions



Recursive filter



Gaussian filter $\sigma = 4$



Recursive filter with $\sigma = 4$

- Important aspect in image processing: measuring the **local change** of intensities (e.g. can indicate object boundaries)
- In continuous functions, the local change is given by the derivative $\frac{df}{dx}$ of the function
- Counterpart in multidimensional functions (such as images) is the gradient

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)^\top$$

- Abbreviated notation: $\nabla I = (I_x, I_y)^\top$
- Image must be differentiable

Gaussian derivative

- Differentiability is ensured by combining the derivative with a small amount of Gaussian smoothing → **Gaussian derivatives**
- Convolution and derivatives are both linear operations
→ Applying the derivative to the image or to the Gaussian does not matter

$$\frac{\partial}{\partial x}(I(x) * G_\sigma(x)) = I(x) * \frac{\partial}{\partial x}G_\sigma(x)$$

- The Gaussian function is in \mathcal{C}^∞ → arbitrarily high order derivatives exist
- Sampling the derivative of the Gaussian yields a discrete filter mask for implementation
- A common mask for the first derivative is $(-1/2, 0, 1/2)$. This is called **central difference**.

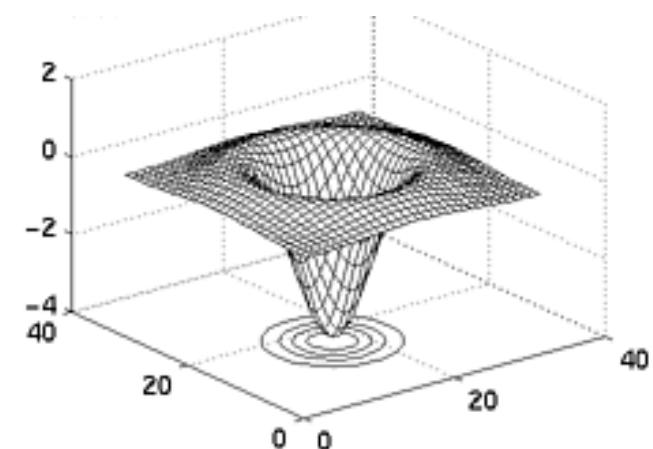
Higher order derivatives

- Sometimes we are interested in higher derivatives than the first derivative
- A popular example is the **Laplace filter**, which is based on second derivatives

$$\Delta I := \frac{\partial^2 I}{\partial^2 x} + \frac{\partial^2 I}{\partial^2 y} = I_{xx} + I_{yy}$$

- Zero-crossings of this filter correspond to image edges
- Another way to detect edges is by the gradient magnitude (based on first derivatives)

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$



Edge detection



Gradient magnitude



Laplacian
(normalized to a [0,255] range)

- There are different noise models depending on the source of noise
- The quality of an image enhancement method can be measured by the signal-to-noise ratio
- Simple point operations like gamma correction or histogram equalization can make dark structures better visible
- Difference images can (under certain conditions) detect moving objects
- An important smoothing filter is the Gaussian filter
- There are fast approximations for large amounts of smoothing, such as the iterated box filter and the recursive filter
- Image edges can be detected with derivative filters

- R. Deriche: Fast algorithms for low-level vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 12(1):78-87, 1990.

Programming assignment

- Get used to the very basic programming environment that we will use in this tutorial. Look at the file `Ex01.cpp` in `ImageProcessingEx01.zip` with any editor. Compile it using the included makefile (just type `make`) or with

```
g++ Ex01.cpp NMath.cpp -I. -o Ex01
```

and run the program via

```
./Ex01
```

This will just create a copy of the image `lena.pgm` called `lenaNoisy.pgm`. You can look at images with `display`. We also provide code for displaying images from within your running program.

- Now add Gaussian noise with standard deviation 10 and 20 to the Lena image by filling in the missing code. Use the Box-Muller method, which is described below, to simulate the noise.

The Box-Muller method creates a Gaussian distributed random variable with $\mu = 0, \sigma = 1$ given uniformly distributed random numbers. The function to generate uniformly distributed random numbers in C/C++ is `rand()`. It generates numbers between 0 and `RAND_MAX`.

First create two independent random variables U and V with uniform distribution in $[0,1]$.

Then compute $N = \sqrt{-2 \ln U} \cos(2\pi V)$ $M = \sqrt{-2 \ln U} \sin(2\pi V)$

N and M are independent Gaussian distributed random variables with $\mu = 0, \sigma = 1$

When saving the degraded image to disk (e.g. PGM format) ensure not to leave the interval $[0,255]$. Clip the values.

Programming assignment

- Measure the PSNR of the noisy images
- Create a sequence of 50 noisy images. Average these images: $\bar{I} = \frac{1}{N} \sum_{i=1}^N I_i$
Measure the PSNR of the averaged image. What do you find?
- Think of a general sequence of images. Why does the above trick not work with general sequences?
- Compute the difference image of `Sidenbladh.ppm` and `SidenbladhBG.ppm`.
Color images can be handled with the class `CTensor`.
- Implement the Gaussian filter, the iterated box filter, and the recursive filter. Note that you must mirror the image at the boundaries to have Neumann boundary conditions. Test these filters on `chinaToilet.pgm`. Compare their results and computation times, particularly for large amounts of smoothing.
- What about color images? Is color a problem? Run your favorite filter on `fallingMangoes.ppm`.

Image Processing and Computer Graphics

Image Processing

Class 4 Energy Minimization

1. Formalize your model assumptions and cast the task as an **optimization problem**

$$E(x) = A_1(x) +, \dots, + A_n(x)$$

2. Solve the optimization problem

$$x^* = \operatorname{argmin}_x E(x)$$

- Objective function $E(x)$ often called **energy** (motivated from physics)
- In machine learning it is often called **loss function**

Example: image denoising

- First step: formulate the model assumptions
 - The outcome should be similar to the input image
 - The result should be smooth

- Second step: formalize these assumptions
 - Similarity to the input data (data term):

$$E_D(u_{i,j}) := \sum_{i,j} (u_{i,j} - I_{i,j})^2 \rightarrow \min$$

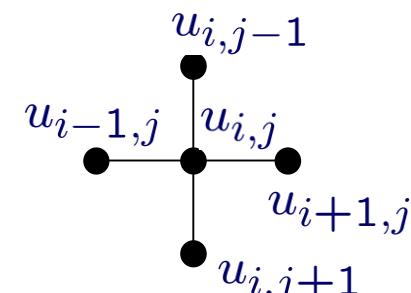
- Similarity to neighboring values (smoothness term):

$$E_S(u_{i,j}) := \sum_{i,j} (u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 \rightarrow \min$$

- Yields an energy minimization problem including a weighting parameter α

$$u_{i,j}^* = \operatorname{argmin}_{u_{i,j}} (E_D(u_{i,j}) + \alpha E_S(u_{i,j}))$$

- Third step: solve this optimization problem



Advantages

- All model assumptions are clearly stated
→ transparency
- Global approach: all variables are optimized in a joint manner;
interdependencies are not lost by intermediate decisions → optimality
- Theoretical aspects can be analyzed:
 - Existence and uniqueness of solutions
 - Stability of solutions with respect of the input data
 - Difficulty of the problem class
- Usually fewer parameters than in heuristic multi-step methods
- Combination of energy minimization approaches is straightforward

Problematic issues

- Formalizing the assumptions is rather ad-hoc: Why minimizing the sum of squared differences and not some other distance?

$$E_D(u_{i,j}) := \sum_{i,j} (u_{i,j} - I_{i,j})^2 \rightarrow \min$$

→ Statistical interpretations (Bayesian models) can usually answer this question.

- Choosing the weighting parameter(s) is not easy and often depends on the data.

→ (Fixed) parameters can be learned from a training dataset.
- Global optimization is often hard.

Heuristics obliterate the initial transparency of the model.

Energy minimization in our denoising example

- Here is our energy from the denoising example

$$E(u_{i,j}) := \sum_{i,j} ((u_{i,j} - I_{i,j})^2 + \alpha ((u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2))$$

- How do we find the minimum?

- Necessary condition** for a minimum: the first derivatives must be zero

$$\frac{dE}{du} = 0 \Leftrightarrow \frac{\partial E}{\partial u_{i,j}} = 0 \quad \forall i, j$$

- Here we go:

$$\begin{aligned} \frac{\partial E}{\partial u_{i,j}} &= 2(u_{i,j} - I_{i,j}) \\ &\quad + 2\alpha(u_{i,j} - u_{i-1,j}) - 2\alpha(u_{i+1,j} - u_{i,j}) \\ &\quad + 2\alpha(u_{i,j} - u_{i,j-1}) - 2\alpha(u_{i,j+1} - u_{i,j}) = 0 \end{aligned}$$

- At boundary pixels some terms are missing due to missing neighbors

Linear system of equations

- Necessary conditions...

$$\begin{aligned}\frac{\partial E}{\partial u_{i,j}} &= (u_{i,j} - I_{i,j}) \\ &\quad + \alpha(u_{i,j} - u_{i-1,j}) - \alpha(u_{i+1,j} - u_{i,j}) \\ &\quad + \alpha(u_{i,j} - u_{i,j-1}) - \alpha(u_{i,j+1} - u_{i,j}) = 0\end{aligned}$$

- ...can be written as a large linear system of equations (schematic view)

$$\left(\begin{array}{cccccc} 1+2\alpha & -\alpha & & -\alpha & & \\ -\alpha & 1+3\alpha & -\alpha & & -\alpha & \\ & -\alpha & 1+3\alpha & -\alpha & & -\alpha \\ -\alpha & & -\alpha & 1+4\alpha & -\alpha & -\alpha \\ -\alpha & & -\alpha & -\alpha & 1+3\alpha & -\alpha \\ & & -\alpha & & -\alpha & 1+3\alpha & -\alpha \\ & & & -\alpha & & -\alpha & 1+2\alpha \end{array} \right) \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_{N-1} \\ I_N \end{pmatrix}$$

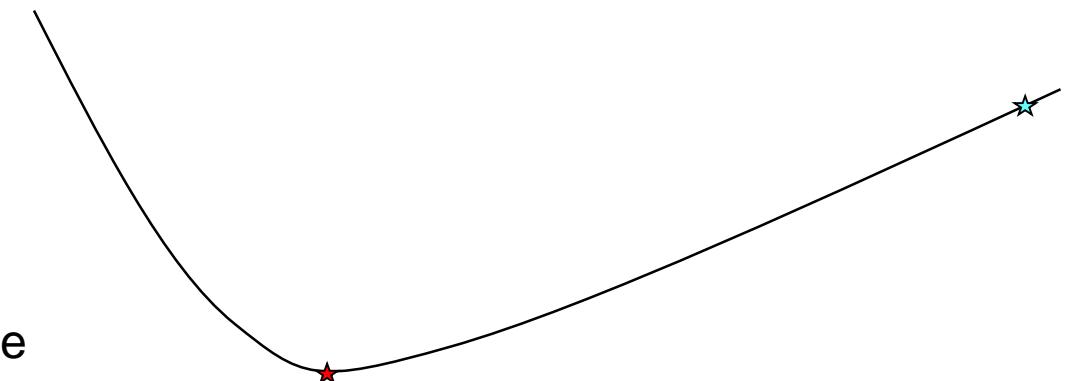
- $N \times N$ system matrix (for N pixels) is symmetric and positive definite
- It contains one main diagonal (central pixels) and four off-diagonals (for each of the four neighbors of a pixel)

Now what?

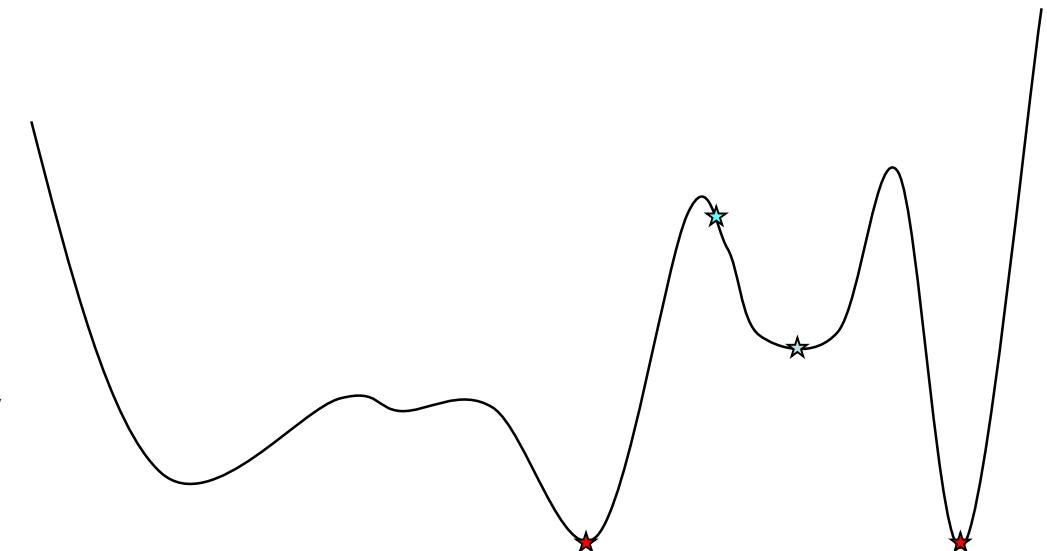
$$\begin{pmatrix} 1+2\alpha & -\alpha & & -\alpha & & \\ -\alpha & 1+3\alpha & -\alpha & & -\alpha & \\ & -\alpha & 1+3\alpha & -\alpha & & -\alpha \\ -\alpha & -\alpha & 1+4\alpha & -\alpha & & -\alpha \\ & -\alpha & -\alpha & 1+3\alpha & -\alpha & \\ & & -\alpha & -\alpha & 1+3\alpha & -\alpha \\ & & & -\alpha & -\alpha & 1+2\alpha \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_{N-1} \\ I_N \end{pmatrix}$$

- The system matrix A is **sparse** (almost all entries are 0)
- Positive definite \rightarrow the inverse A^{-1} exists and we can solve for \mathbf{u}
- Questions:
 - When do we get such a linear system with a unique solution?
 - How can this system be solved (efficiently)?

- Convex functions:
 - Positive curvature
 - No local minima
 - Global minimum is unique
 - Minimization by setting the derivative to 0 and solving the emerging linear or nonlinear system



- Non-convex functions:
 - Usually many local minima (and maxima)
 - Global minimum may not be unique
 - Global minimization is usually impossible, only heuristics exist



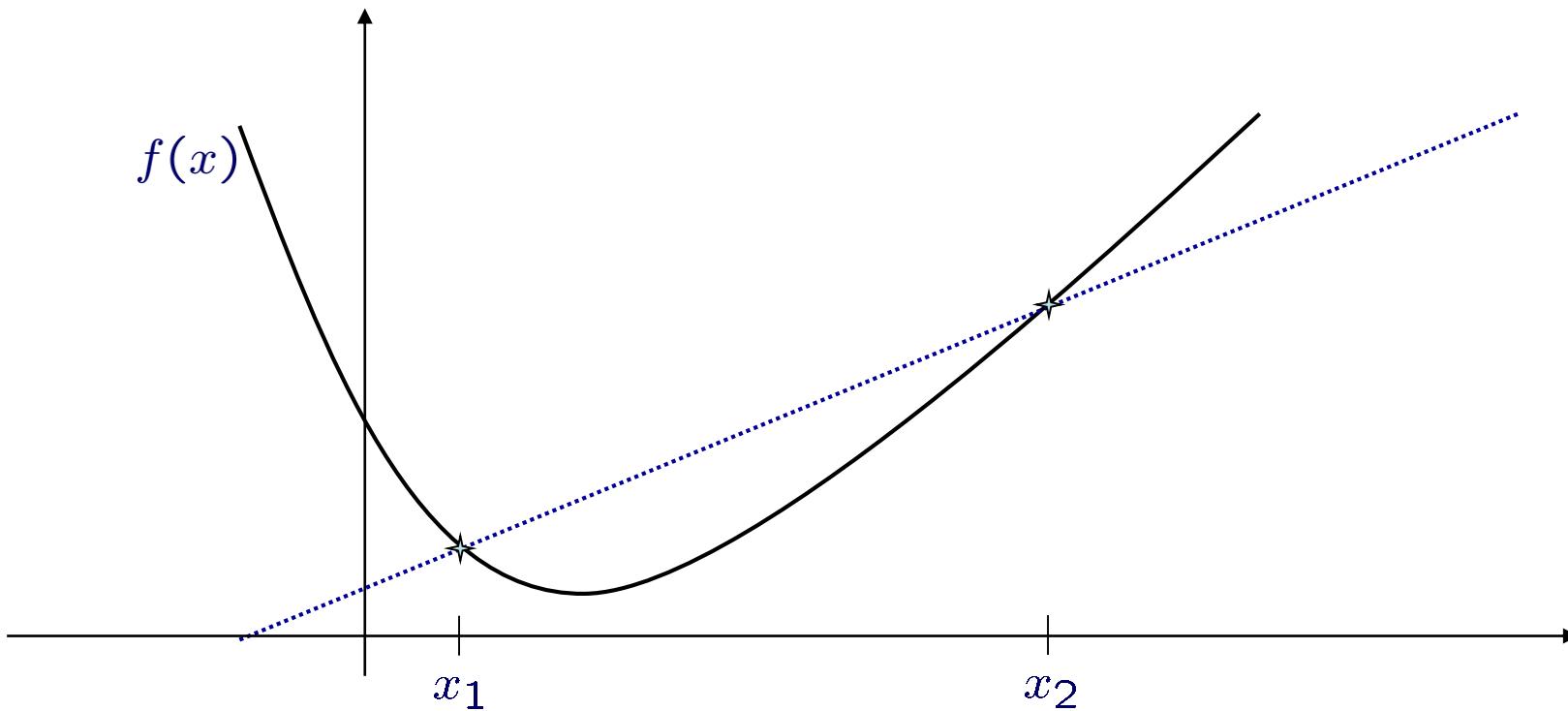
Convexity

- A function is **convex** if

$$f((1-\alpha)x_1 + \alpha x_2) \leq (1-\alpha)f(x_1) + \alpha f(x_2) \quad \forall x_1, x_2, \forall \alpha \in (0, 1)$$

- A function is **strictly convex** if

$$f((1-\alpha)x_1 + \alpha x_2) < (1-\alpha)f(x_1) + \alpha f(x_2) \quad \forall x_1, x_2, \forall \alpha \in (0, 1)$$



- Theorem: every convex combination of (strictly) convex functions is again (strictly) convex
- Proof:

$$h(x) := \gamma f(x) + \delta g(x) \quad \gamma, \delta \in \mathbb{R}, \quad \gamma, \delta \geq 0$$

$$\begin{aligned} h((1 - \alpha)x_1 + \alpha x_2) &= \gamma f((1 - \alpha)x_1 + \alpha x_2) + \delta g((1 - \alpha)x_1 + \alpha x_2) \\ &\leq \gamma(1 - \alpha)f(x_1) + \gamma\alpha f(x_2) + \delta(1 - \alpha)g(x_1) + \delta\alpha g(x_2) \\ &= (1 - \alpha)(\gamma f(x_1) + \delta g(x_1)) + \alpha(\gamma f(x_2) + \delta g(x_2)) \\ &= (1 - \alpha)h(x_1) + \alpha h(x_2) \end{aligned}$$

- Our energy function

$$E(u_{i,j}) := \sum_{i,j} \left((u_{i,j} - I_{i,j})^2 + \alpha \left((u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 \right) \right)$$

is a convex combination of strictly convex (quadratic) functions
→ It is strictly convex as well
→ It has a unique global minimum

Solving the linear system

$$\begin{pmatrix} 1+2\alpha & -\alpha & & & \\ -\alpha & 1+3\alpha & -\alpha & & \\ & -\alpha & 1+3\alpha & -\alpha & \\ & & -\alpha & 1+4\alpha & -\alpha \\ -\alpha & -\alpha & -\alpha & 1+3\alpha & -\alpha \\ & -\alpha & -\alpha & -\alpha & 1+3\alpha \\ & & -\alpha & -\alpha & -\alpha \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = \begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ \vdots \\ I_{N-2} \\ I_{N-1} \end{pmatrix}$$

The matrix is circled in red, showing its structure. The main diagonal has entries $1+2\alpha, 1+3\alpha, 1+3\alpha, 1+4\alpha, 1+3\alpha, 1+3\alpha, 1+2\alpha$. The super-diagonal has entries $-\alpha, -\alpha, -\alpha, -\alpha, -\alpha, -\alpha$. The sub-diagonal has entries $-\alpha, -\alpha, -\alpha, -\alpha, -\alpha, -\alpha$. There are two additional off-diagonals highlighted in blue, which are $-\alpha$ and $-\alpha$, located at positions (1,3), (2,4), (3,5), (4,6), (5,7), and (6,8).

- The two additional off-diagonals (together with the size of matrix) rule out Gauß-elimination (in 1D, however, Gauß-elimination is very efficient).
- An iterative solver is needed to preserve the sparsity of the matrix
- Simplest iterative solver: **Jacobi method**
- Converges if the matrix is **strictly diagonal dominant**

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}| \quad \forall i$$

Jacobi method

- Decompose the matrix into its diagonal part D and its off-diagonal part M

$$A = D + M$$

- For the linear system this means

$$Ax = b \Leftrightarrow (D+M)x = b \Leftrightarrow Dx = b - Mx$$

- D^{-1} can be computed very easily: just replace the diagonal elements by their inverse.

- Now we can compute the solution x iteratively. Starting with any initialization x^0 , iterate

$$x^{k+1} = D^{-1}(b - Mx^k)$$

- Iterate until the norm of the **residual** $r^k := Ax^k - b$ is smaller than a threshold or the change in the solution $(x^{k+1} - x^k)^2$ becomes small. When the change is 0, the iterate has **converged**.

- Advantages:
 - Simple
 - Can be implemented in parallel
- Disadvantages:
 - Slow
 - Convergence only for $k \rightarrow \infty$
 - Computation not in-place
- Alternatives:
 - Gauß-Seidel, Successive Over-relaxation (faster, in-place)
 - Conjugate gradient (convergence after finite number of iterations)
 - Multigrid methods (sometimes much faster)

Gauß-Seidel method

- Split the off-diagonal part M into the lower triangle L and the upper triangle U

$$Ax = b \Leftrightarrow Dx = b - Lx - Ux$$

- During iteration, traverse the vector x from top to bottom and use already the new values for multiplication with the lower triangle

$$x^{k+1} = D^{-1}(b - Lx^{k+1} - Ux^k)$$

- In our denoising example this reads

$$u_i^{k+1} = \frac{I_i + \alpha \sum_{j \in \mathcal{N}^-(i)} u_j^{k+1} + \alpha \sum_{j \in \mathcal{N}^+(i)} u_j^k}{1 + \sum_{j \in \mathcal{N}(i)} \alpha}$$

- Converges if A is positive or negative definite
- For already updated neighbors take the new value → in-place computation
- Recursive propagation of information → faster

Successive over-relaxation (SOR)

- Emphasize the Gauß-Seidel idea by over-relaxing the new solution

$$x^{k+1} = (1 - \omega)x^k + \omega D^{-1}(b - Lx^{k+1} - Ux^k)$$

- For $\omega = 1$ this is the Gauß-Seidel method
- Converges for positive- or negative-definite matrices (all eigenvalues positive or negative, respectively), if $\omega \in (0, 2)$
- Over-relaxation for $\omega > 1$: faster convergence
- Under-relaxation for $\omega < 1$: can help establish convergence in case of divergent iterative processes
- Optimal ω must be determined empirically

Conjugate gradient (CG)

- Two non-zero vectors u and v are **conjugate** with respect to A if the inner product $\langle u, v \rangle_A := u^\top A v = 0$. This means the two vectors are orthogonal with respect to this special scalar product.
- A set of n conjugate vectors $\{p_k\}$ forms a basis of \mathbb{R}^n , so the solution x^* of $Ax = b$ can be expanded as $x^* = \alpha_1 p_1 +, \dots, + \alpha_n p_n$
- The coefficients α_k are derived as follows:

$$Ax^* = \alpha_1 A p_1 +, \dots, + \alpha_n A p_n = b$$

$$p_k^\top A x^* = p_k^\top \alpha_1 A p_1 +, \dots, + p_k^\top \alpha_n A p_n = p_k^\top b \quad (\text{expansion with } p_k)$$

$$\alpha_k = \frac{p_k^\top b}{p_k^\top A p_k}$$

- After n computations we obtain the exact solution x^*
- Good choice of $\{p_k\} \rightarrow$ few coefficients approximate the solution well

Conjugate gradient: iteratively assembling the basis

- Start with some initial point x^0
- Let p_0 be the residual $r_0 = b - Ax^0$. This is the gradient of

$$E(x) = \frac{1}{2}x^\top Ax - b^\top x$$

the minimizer of which is x^* . Therefore the name conjugate gradient.

- Iteratively compute:

$$\alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k} \quad x^{k+1} = x^k + \alpha_k p_k \quad r_{k+1} = r_k - \alpha_k A p_k$$

$$\beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k} \quad p_{k+1} = r_{k+1} + \beta_k p_k$$

- Stop when residual is small. Guaranteed solution after n iterations.

Conjugate gradient: convergence and preconditioning

- Matrix A must be symmetric and positive definite
- Usually in image processing, computing the exact solution is not an option since n is the number of pixels
- Number of iterations needed to get a good approximate solution depends on the **condition number** of A (largest vs. smallest eigenvalue). The same holds for the other iterative methods (Gauß-Seidel, etc.)

- Sometimes so-called **preconditioners** P^{-1} are used to have a small condition number for $P^{-1}A$

$$Ax = b \quad \Leftrightarrow \quad P^{-1}Ax = P^{-1}b$$

- Simplest preconditioner: Jacobi preconditioner

$$P = D \quad \Leftrightarrow \quad P^{-1} = D^{-1}$$

$$Ax = b \quad \Leftrightarrow \quad D^{-1}Ax = D^{-1}b$$

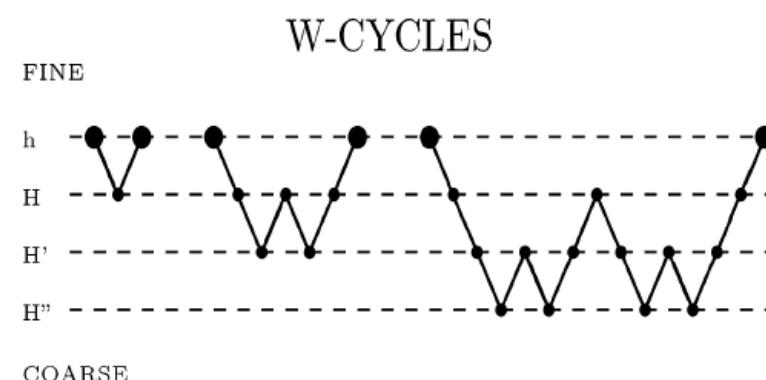
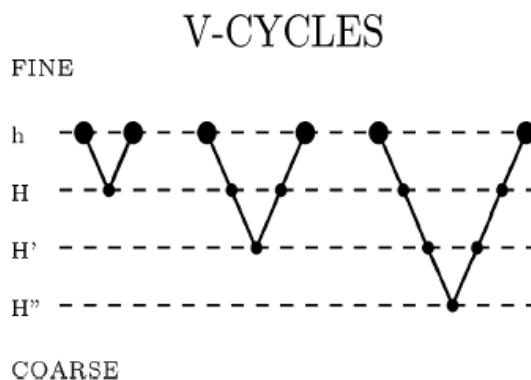
- All previous linear solvers have the drawback that they only act locally.
- This is due to the sparsity of the matrix: one iteration only distributes information at a pixel to its four neighbors (the recursive nature of Gauß-Seidel relaxes this statement a bit).
- Idea of multigrid solvers: shorten distances by regarding the system from a coarser point of view
- Additional effect: coarse versions of the system have fewer entries
→ iterations are faster at coarse levels
- Different types of multigrid solvers
 - Unidirectional (cascadic) multigrid
 - Bidirectional (correcting) multigrid
 - Full multigrid (a combination of both)

Unidirectional (cascadic) multigrid

- Create downsampled versions of the linear system (usually by downsampling the image and deriving the linear system from this)
- Compute first approximate solution at the coarse grid (e.g. with SOR)
- Take upsampled result as initial guess for the next finer grid
- Refine result there (again with SOR)
- Advantages:
 - Iterations at the coarse level are very fast
 - Simple implementation
- Disadvantage:
 - Coarse level systems often do not approximate the original system well
→ Iterations at coarse levels do not really help

Correcting (bidirectional) multi-grid

- Basic idea: do not downsample the image but the error
- Compute first solution at fine grid
- Correct error at coarse grid
- Refine result at finer grid



Author: Andrés Bruhn

Correcting multigrid in detail

1. Presmoothing relaxation step $A^h x^h = b^h$

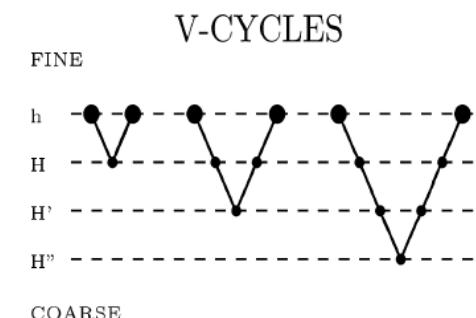
- Run some iterations at the fine grid
- Yields approximate solution \tilde{x}^h
- Remaining error: $e^h = x^h - \tilde{x}^h$

2. Correction step:

- Goal: compute error $A^h e^h = r^h$ $r^h = b^h - A^h \tilde{x}^h$
- Local part of error already removed
→ Solve this system at coarser grid $A^H e^H = r^H$
- Transfer error to fine grid and correct the solution $\tilde{\tilde{x}}^h = \tilde{x}^h + \tilde{e}^h$

3. Postsmothing relaxation step

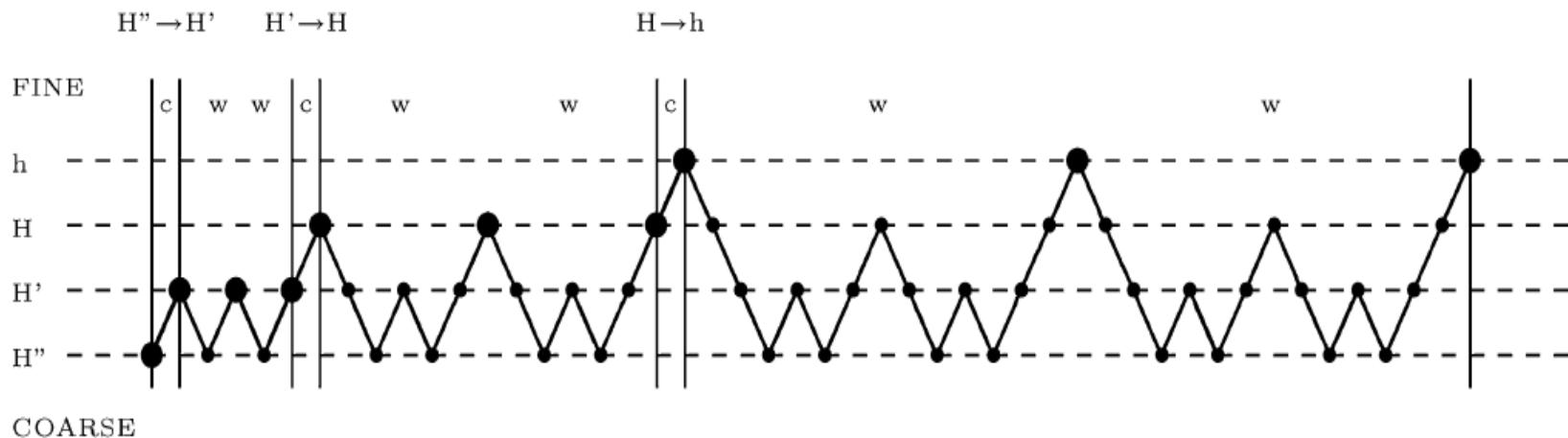
- Apply some further iterations at fine grid to remove local errors introduced by \tilde{e}^h



Full multigrid

- Combination of cascadic and correcting multigrid
- Start at coarse grid with downsampled image
- At each finer level apply a W-cycle

FULL MULTIGRID



Author: Andrés Bruhn

- Here we were considering problems with continuous variables (each vector component of the solution is a real number)
- Segmentation and matching typically leads to integer problems
- These are combinatorial problems, only few of them being solvable in polynomial time
- Some typical problems arising in computer vision are:
 - Linear programs (LP)
 - Integer quadratic programs (IQP) including special cases like min-cut
 - Second order cone programs (SOCP)
- More in the Computer Vision course

- The energy minimization framework is a sound way to model and solve image processing problems
- All model assumptions are clearly stated, no hidden assumptions
- Global optimization is “easy” if the energy function is convex
- The necessary condition for a minimum is that the gradient is zero
- Leads to a large, but sparse, linear or nonlinear system of equations
- There are several methods to solve sparse linear systems iteratively, some are easier to implement, others are faster

- D. Young: Iterative Solution of Large Linear Systems, Academic Press, 1971.
Reprint by Dover 2003.
- J. R. Shewchuk: An introduction to the Conjugate Gradient method without the agonizing pain. CMU Technical Report, 1994.

Image Processing and Computer Graphics

Image Processing

Class 5 Variational Methods

Discrete vs. continuous energies

- So far we have only considered discrete energy minimization problems

$$u^* = \operatorname{argmin}_u E(u)$$

with a vector $u \in \mathbb{R}^N$

- Alternative approach: continuous formulation

$$u^*(x) = \operatorname{argmin}_{u(x)} E(u(x))$$

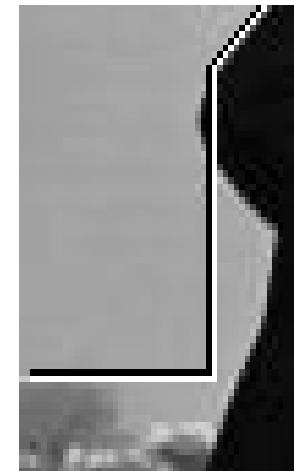
with a function $u(x) : \Omega \rightarrow \mathbb{R}$

- In this case, the energy function becomes an **energy functional**
- Optimization is based on the **calculus of variation** (Variationsrechnung)
- Yields necessary condition(s) for a minimum: **Euler-Lagrange equations**
- Discretization postponed to these equations

Discrete or continuous energies?

Advantage of continuous energies

- It is not ensured that a discrete energy is **consistent** with the continuous one.
Typical example: measuring line lengths.



Optimum contours for a
discrete energy

Disadvantage

- Gradients in direction of a function have to be computed.

However, this is not as difficult as it seems...



Optimum contours for a
continuous energy

Author: Maria Klodt

Consistency

- Discretization of continuous quantities can lead to artifacts if done wrong
- For instance, we want the solution of a problem to be independent of the rotation of the grid (rotation invariance)
- A discretization is called **consistent** if it converges to the continuous model with finer grid sizes
- Example: discrete approximation of the gradient magnitude
 - Inconsistent discretization

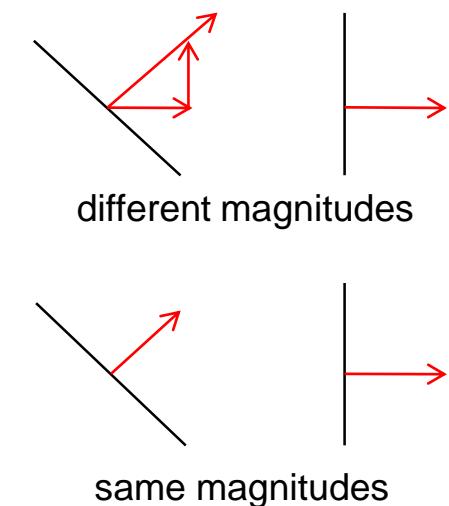
$$|\nabla u| \approx |u_{i+1,j} - u_{i-1,j}| + |u_{i,j+1} - u_{i,j-1}|$$

There is an error independent of the grid size

- Consistent discretization

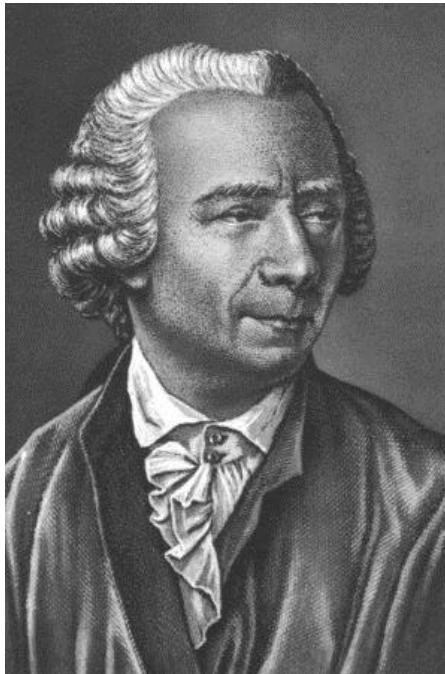
$$|\nabla u| \approx \sqrt{(u_{i+1,j} - u_{i-1,j})^2 + (u_{i,j+1} - u_{i,j-1})^2}$$

All errors depend on the grid size

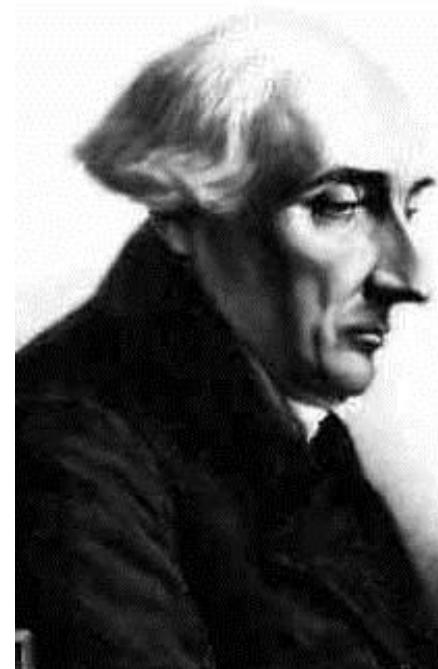


- In the discrete setting we obtained a necessary condition for a minimum by computing the gradient of the function $E(u)$ with respect to the unknown vector $u \in \mathbb{R}^N$.
- This gradient $\frac{\partial E(u)}{\partial u}$ was computed by the tool of partial derivatives $\frac{\partial E(u)}{\partial u_i}$
- How do we get the gradient of a functional $\frac{\partial E(u(x))}{\partial u(x)}$ with respect to a function $u(x)$?
- The answer is the calculus of variation and the **Gâteaux derivative**.
- It has been invented by the French mathematician René Gâteaux (who died in World War I) and has been published posthumously 1919.
- Similar ideas date back to the works of Leonhard Euler and Joseph-Louis Lagrange in the 18th century. The necessary conditions are hence called Euler-Lagrange equations.

Euler and Lagrange



Leonhard Euler (1707-1783)



Joseph-Louis Lagrange (1736-1813)

- Born in Basel, Switzerland
- One of the most important mathematicians in history (Euler angle, Euler number, Euler formula,...)
- 13 children
- Published almost 900 papers and books, most of them in the last 20 years of his life
- Became totally blind in 1771

- Born in Turin, Italy
- One of 11 children, but only two of them lived to adulthood
- Autodidact
- Became professor in Turin at the age of 19
- Later he worked in Berlin and Paris
- Worked together with Euler on the calculus of variation

Gâteaux derivative

- A functional maps each element u of a vector space (e.g. a function) to a scalar $E(u)$.
- The Gâteaux derivative generalizes the directional derivative to infinite-dimensional vector spaces

$$\frac{\partial E(u)}{\partial u} \Big|_h = \lim_{\epsilon \rightarrow 0} \frac{E(u + \epsilon h) - E(u)}{\epsilon} = \frac{dE(u + \epsilon h)}{d\epsilon} \Big|_{\epsilon=0}$$

- This can be interpreted as the projection of the gradient to the direction h

$$\frac{\partial E(u)}{\partial u} \Big|_h = \left\langle \frac{dE(u)}{du}, h \right\rangle = \int \frac{dE(u)}{du}(\mathbf{x}) h(\mathbf{x}) d\mathbf{x}$$

- This gradient $\frac{dE(u)}{du}(\mathbf{x})$ is needed to minimize $E(u)$

Denoising example

- A continuous denoising energy is defined by the functional

$$E(u(\mathbf{x})) = \int_{\Omega} (u(\mathbf{x}) - I(\mathbf{x}))^2 + \alpha |\nabla u(\mathbf{x})|^2 d\mathbf{x}$$

- Rather than on a vector, the optimization problem is on a function
- Short writing: usually one does not explicitly indicate the dependency of the functions on \mathbf{x}

$$E(u) = \int_{\Omega} (u - I)^2 + \alpha |\nabla u|^2 d\mathbf{x}$$

- Necessary condition for a minimum:

$$\frac{\partial E(u)}{\partial u} \Big|_h = 0 \quad \forall h$$

Deriving the Euler-Lagrange equation

- Compute the Gâteaux derivative and simplify

$$\begin{aligned}
 \frac{\partial E(u)}{\partial u} \Big|_h &= \lim_{\epsilon \rightarrow 0} \frac{E(u + \epsilon h) - E(u)}{\epsilon} = \frac{dE(u + \epsilon h)}{d\epsilon} \Big|_{\epsilon=0} \\
 &= \frac{d}{d\epsilon} \int_{\Omega} (u + \epsilon h - I)^2 + \alpha |\nabla(u + \epsilon h)|^2 d\mathbf{x} \Big|_{\epsilon=0} \\
 &= \frac{d}{d\epsilon} \int_{\Omega} (u + \epsilon h - I)^2 + \alpha \left(\frac{d}{dx}(u + \epsilon h) \right)^2 + \alpha \left(\frac{d}{dy}(u + \epsilon h) \right)^2 d\mathbf{x} \Big|_{\epsilon=0} \\
 &= \int_{\Omega} 2(u + \epsilon h - I)h + 2\alpha \left(\frac{d}{dx}(u + \epsilon h) \frac{dh}{dx} + \frac{d}{dy}(u + \epsilon h) \frac{dh}{dy} \right) d\mathbf{x} \Big|_{\epsilon=0} \\
 &= \int_{\Omega} 2(u - I)h + 2\alpha (u_x h_x + u_y h_y) d\mathbf{x}
 \end{aligned}$$

- Partial integration to turn h_x and h_y into h

$$\begin{aligned}
 \frac{\partial E(u)}{\partial u} \Big|_h &= \int_{\Omega} 2(u - I)h - 2\alpha(u_{xx}h + u_{yy}h) d\mathbf{x} + (u_x h)_{\partial\Omega_x} + (u_y h)_{\partial\Omega_y} \\
 &= \int_{\Omega} (2(u - I) - 2\alpha(u_{xx} + u_{yy}))h d\mathbf{x} + ((\mathbf{n}^\top \nabla u)h)_{\partial\Omega}
 \end{aligned}$$

where $\partial\Omega$ denotes the boundary and \mathbf{n} the boundary normal.

Gradient and boundary conditions

In a minimum, the directional derivative must be 0 for all directions

$$\frac{\partial E(u)}{\partial u} \Big|_h = \int_{\Omega} (2(u - I) - 2\alpha(u_{xx} + u_{yy}))h \, d\mathbf{x} + ((\mathbf{n}^\top \nabla u)h)_{\partial\Omega} = 0$$

Yields two conditions:

1. The gradient must be zero \rightarrow Euler-Lagrange equation:

$$\frac{dE(u)}{du} = 2(u - I) - 2\alpha(u_{xx} + u_{yy}) = 0$$

2. Boundary conditions:

$$(\mathbf{n}^\top \nabla u)_{\partial\Omega} = 0$$

These boundary conditions are called **natural boundary conditions** as they naturally emanate from the Gâteaux derivative. In the special case here, they coincide with **Neumann boundary conditions**.

- Discretization of

$$\frac{dE(u)}{du} = (u - I) - \alpha(u_{xx} + u_{yy}) = 0$$

leads to a (in this case) linear system of equations

$$\frac{\partial E}{\partial u_{i,j}} = (u_{i,j} - I_{i,j}) - \alpha(u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}) = 0$$

- This is the same linear system one has to solve for minimizing the discrete energy from last class.
- Same linear solvers can be applied
- The two different discretization stages do not always lead to the same algorithm

Smoothing results – do they remind you of some other method?



Input image



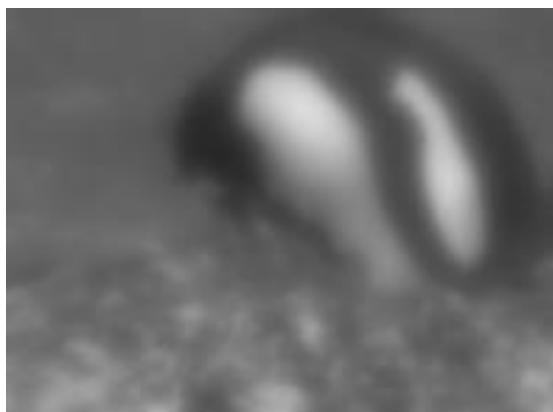
$\alpha = 1$



$\alpha = 5$



$\alpha = 10$

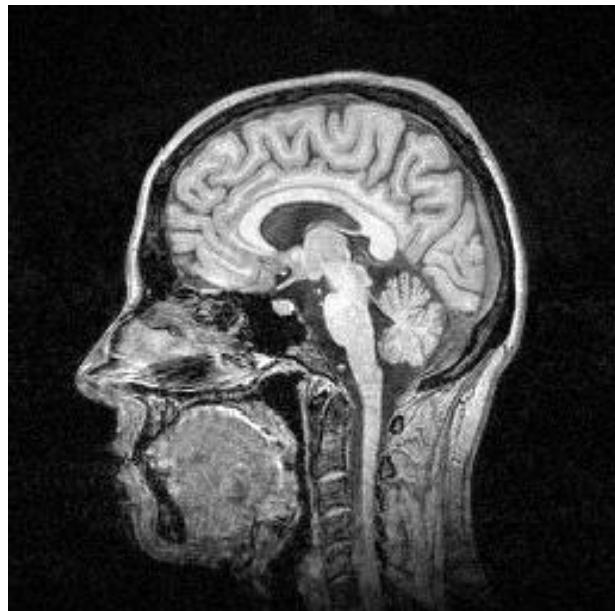


$\alpha = 20$

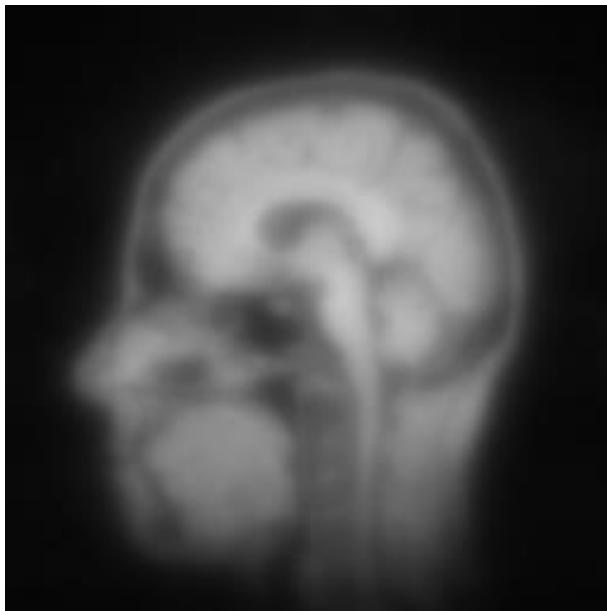


$\alpha = 100$

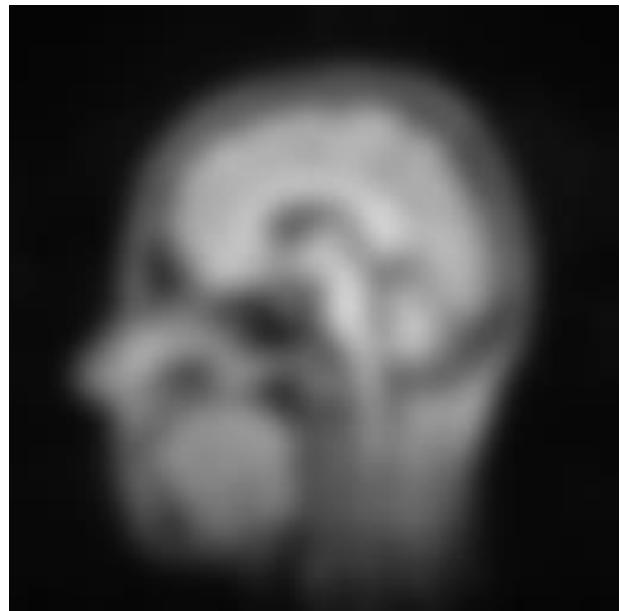
Tikhonov regularization vs. Gaussian smoothing



Input image



Tikhonov regularization
 $\alpha = 20$



Gaussian smoothing
 $\sigma^2 = 40$

Discontinuity preserving regularization

- A quadratic regularizer leads to blurred edges
- Why? Well, we said the result should be smooth everywhere, right?
- Not true for edges
→ we must allow for some exceptions (so-called **outliers**)
- This can be done by using non-quadratic regularizers
- Two ways to understand this (leading to the same solution)
 1. Nonlinear diffusion methods (Computer Vision course)
 2. Statistical interpretation of regularization

- Energy minimization can also emerge from a probabilistic approach taking into account likelihoods and prior probabilities
- It builds upon the **Bayes formula**:

$$p(u|I) = \frac{p(I|u)p(u)}{p(I)}$$

a posteriori probability

marginal probability

evidence/likelihood

a priori probability

- Find solution u that maximizes the posterior probability

$$p(u|I) \rightarrow \max$$

→ maximum a-posteriori (MAP) approach

- Marginal does not depend on u → can be ignored in the maximization

$$p(u|I) \propto p(I|u)p(u)$$

How is this related to energy minimization?

- MAP estimation

$$p(u|I) \propto p(I|u)p(u) \rightarrow \max$$

- Noise model for the data: i.i.d. Gaussian noise
(i.i.d. = independently and identically distributed)

$$p(I|u) \propto \prod_{\mathbf{x} \in \Omega} \exp(-(u(\mathbf{x}) - I(\mathbf{x}))^2)$$

- A-priori model: gradient is likely to be small (smoothness)
→ i.i.d. zero mean Gaussian noise with variance $1/2\alpha$ on the gradient

$$p(u) \propto \prod_{\mathbf{x} \in \Omega} \exp\left(-\frac{|\nabla u(\mathbf{x})|^2}{1/\alpha}\right)$$

- Turn maximization into minimization of the negative logarithm (which is monotonous) → energy minimization problem

$$-\log p(I|u) - \log p(u) = \int_{\Omega} (u - I)^2 + \alpha |\nabla u|^2 d\mathbf{x} \rightarrow \min$$

- Other noise models lead to different penalizers in the energy functional
- Expecting outliers in the smoothness assumption, the Gaussian noise model should be replaced, e.g., by a Laplace distribution

$$p(u) \propto \prod_{\mathbf{x} \in \Omega} \exp \left(-\frac{|\nabla u(\mathbf{x})|}{1/\alpha} \right)$$

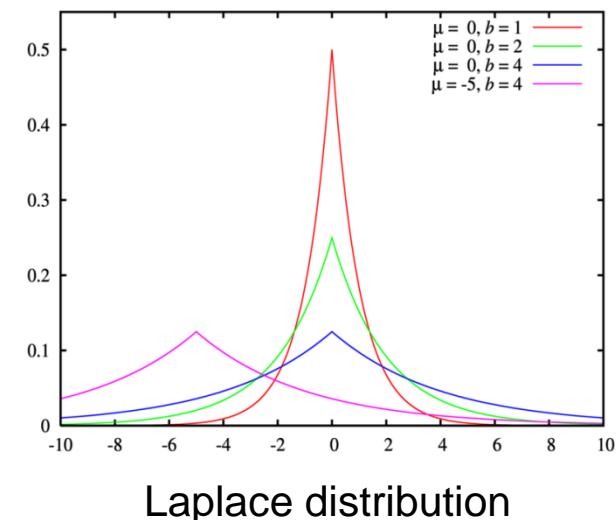
- Leads to an energy functional that preserves image edges

$$E(u) = \int_{\Omega} (u - I)^2 + \alpha |\nabla u| d\mathbf{x}$$

- More general:

$$E(u) = \int_{\Omega} (u - I)^2 + \alpha \Psi(|\nabla u|^2) d\mathbf{x} \quad \text{here: } \Psi(s^2) = \sqrt{s^2}$$

- (Quadratic) Tikhonov regularizer is another special case: $\Psi(s^2) = s^2$



How does this affect minimization?

- Corresponding Euler-Lagrange equation (verify at home):

$$\operatorname{div} \left(\Psi'(|\nabla u|^2) \nabla u \right) - \frac{u - I}{\alpha} = 0$$

where $\Psi'(s^2)$ is the derivative of $\Psi(s^2)$ with respect to its argument.

- So we get

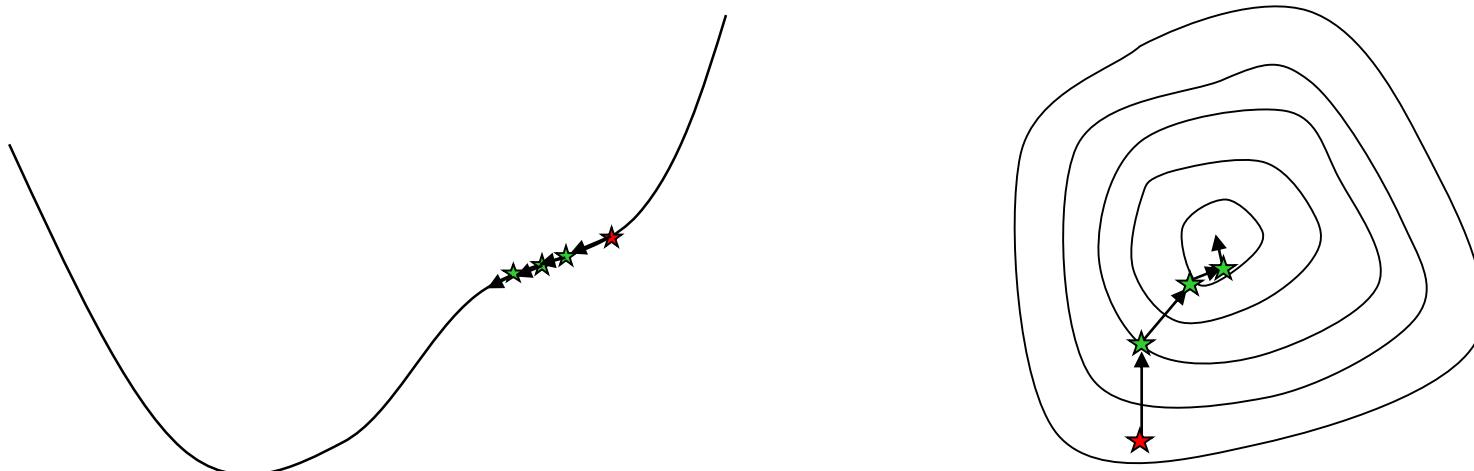
$$\Psi(s^2) = \sqrt{s^2} \quad \Rightarrow \quad \Psi'(s^2) = \frac{1}{2\sqrt{s^2}}$$

$$\operatorname{div} \left(\frac{\nabla u}{2|\nabla u|} \right) - \frac{u - I}{\alpha} = 0$$

- Do you see the trouble?

Gradient descent

- The Euler-Lagrange equation is nonlinear in the unknowns
→ discretization leads to a nonlinear system of equations
- A general way to minimize discrete or continuous energies (even if the gradient is nonlinear) is by **gradient descent**
- Iterative technique: start with an initial value, iteratively move in direction of largest decrease in energy (negative gradient direction)



- Converges to the next (with regard to the initialization) local minimum

Is the problem convex or non-convex?

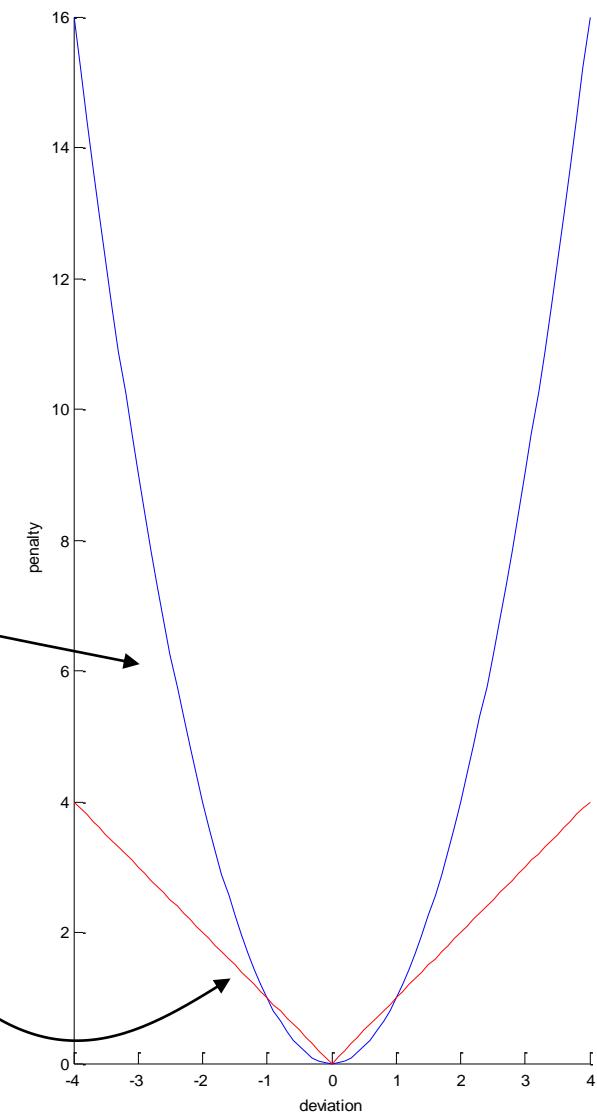
- Let's see....

$$E(u) = \int_{\Omega} (u - I)^2 + \alpha |\nabla u| dx$$

clearly
convex

This was close!
But yes, convex

- Linear combination is convex
→ gradient descent will find a global optimum



Gradient descent

- Initialize u^0

- Introduce artificial time

$$\frac{\partial u}{\partial t} = -\frac{dE(u)}{du} = \operatorname{div} \left(\frac{\nabla u}{2|\nabla u|} \right) - \frac{u - I}{\alpha}$$

- Compute solution for $t \rightarrow \infty$

- Discrete steps of size τ forward in time

$$u^{k+1} = u^k + \tau \left(\operatorname{div} \left(\frac{\nabla u^k}{2|\nabla u^k|} \right) - \frac{u^k - I}{\alpha} \right)$$

- Will converge, if τ is “small enough” (more in course Computer Vision)

- Set $\Psi' = \frac{1}{\sqrt{|\nabla u|^2 + \epsilon^2}}$, $\epsilon = 0.01$, then $\tau \leq \frac{\epsilon}{4}$

→ slow convergence

A faster numerical scheme: lagged diffusivity

$$\operatorname{div} \left(\Psi' \left(|\nabla u|^2 \right) \nabla u \right) - \frac{u - I}{\alpha} = 0$$

- Keep the nonlinear prefactor fixed (now we have again a linear system) and compute updates in an iterative manner

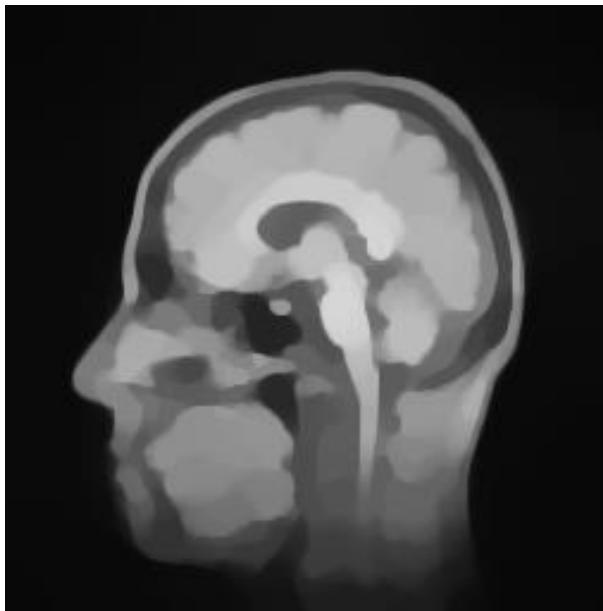
$$\operatorname{div} \left(\Psi'^k \nabla u^{k+1} \right) - \frac{u^{k+1} - I}{\alpha} = 0$$

- This scheme is called **lagged diffusivity** and has been proven to converge if the linear system in each step is solved exactly
- In practice, only few iterations of the iterative linear solver are computed before Ψ' is updated → increased efficiency
- Much faster than gradient descent

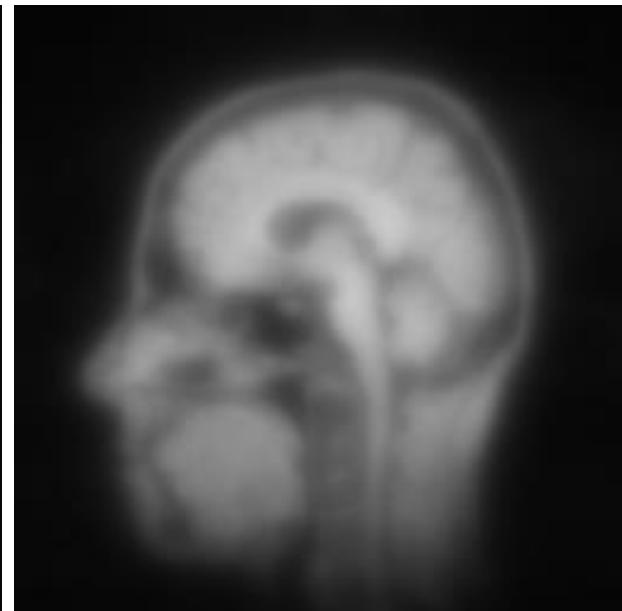
TV regularization vs. Tikhonov regularization



Input image



TV regularization
 $\alpha = 20$



Tikhonov regularization
 $\alpha = 20$

- The energy minimization framework also works with continuous models.
- Energy functions then turn into energy functionals where the unknowns are infinite-dimensional.
- Gradients of such functionals can be derived from the calculus of variation and the Gâteaux derivative. They lead to the Euler-Lagrange equation(s).
- The Bayes formula and the MAP approach lead to a statistical interpretation of many energy minimization problems
- We can design an edge-preserving smoothing method by choosing certain non-quadratic penalizers
- Optimization by gradient descent or the lagged diffusivity approach

- L. D. Elsgolc: Calculus of Variations, Pergamon Press, 1961. Reprint by Dover 2007.
- O. Scherzer, J. Weickert: Relations between regularization and diffusion filtering, Journal of Mathematical Imaging and Vision, 12:43-63, 2000.

Image Processing and Computer Graphics

Image Processing

Class 6 Motion estimation

Motion estimation

- Given a sequence of images $I(x, y, t)$, what is the motion of each pixel between subsequent frames?



Hamburg taxi sequence



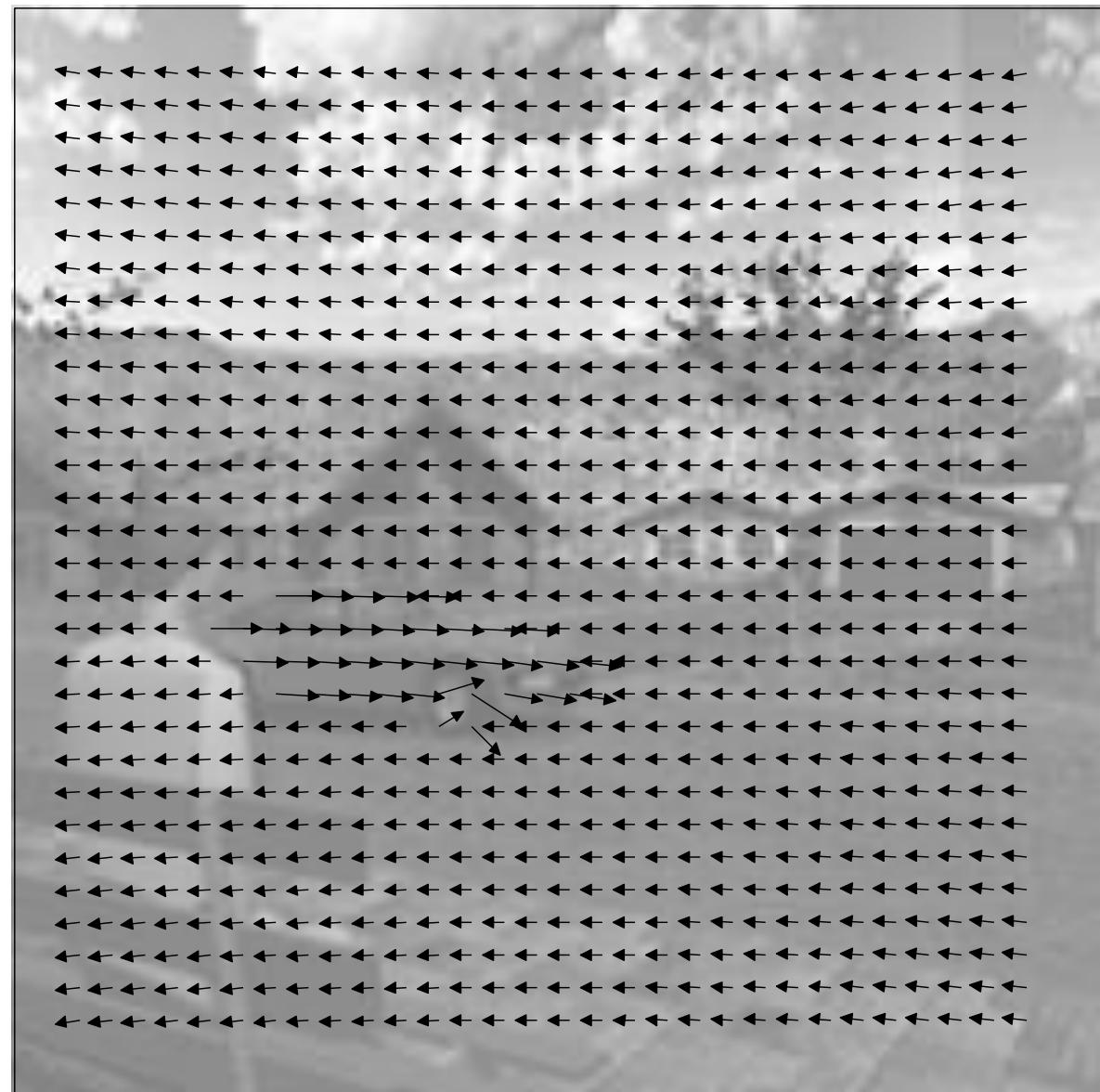
Sequence from VSB100 dataset

- Formally, we seek a vector field $(u, v)(x, y, t)$ that transforms the second image into the first one.
- This vector field is called **optical flow**. It individually moves each point (x, y) at time t such that it fits the point at time $t + 1$.

Motion estimation



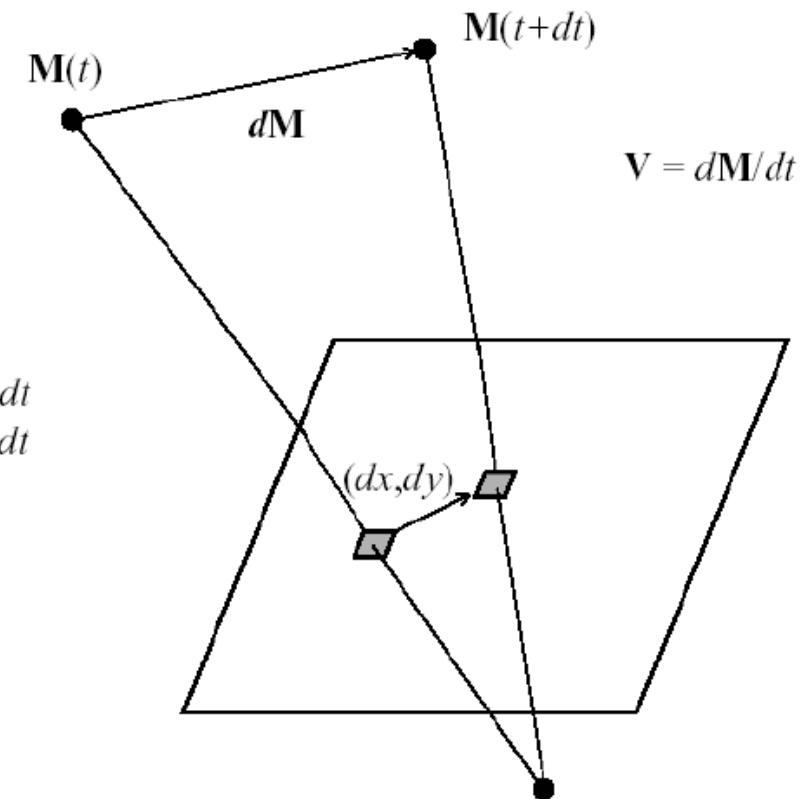
Street sequence



Importance of motion for vision

- Optical flow provides important information for many other tasks in vision.
- It can be used for detection or segmentation of objects by comparing the object motion and the background motion → **motion segmentation**
- It can be used for estimating the 3D motion of objects, the so-called **scene flow**. It is important, e.g., for catching a fly or for avoiding collisions.
- Optical flow can also help estimate the 3D structure of objects. This is called **structure from motion**.
 - Moving your head, the image of near objects moves faster than the image of distant objects (motion parallax).
 - For many animals, motion is the only way to see 3D.

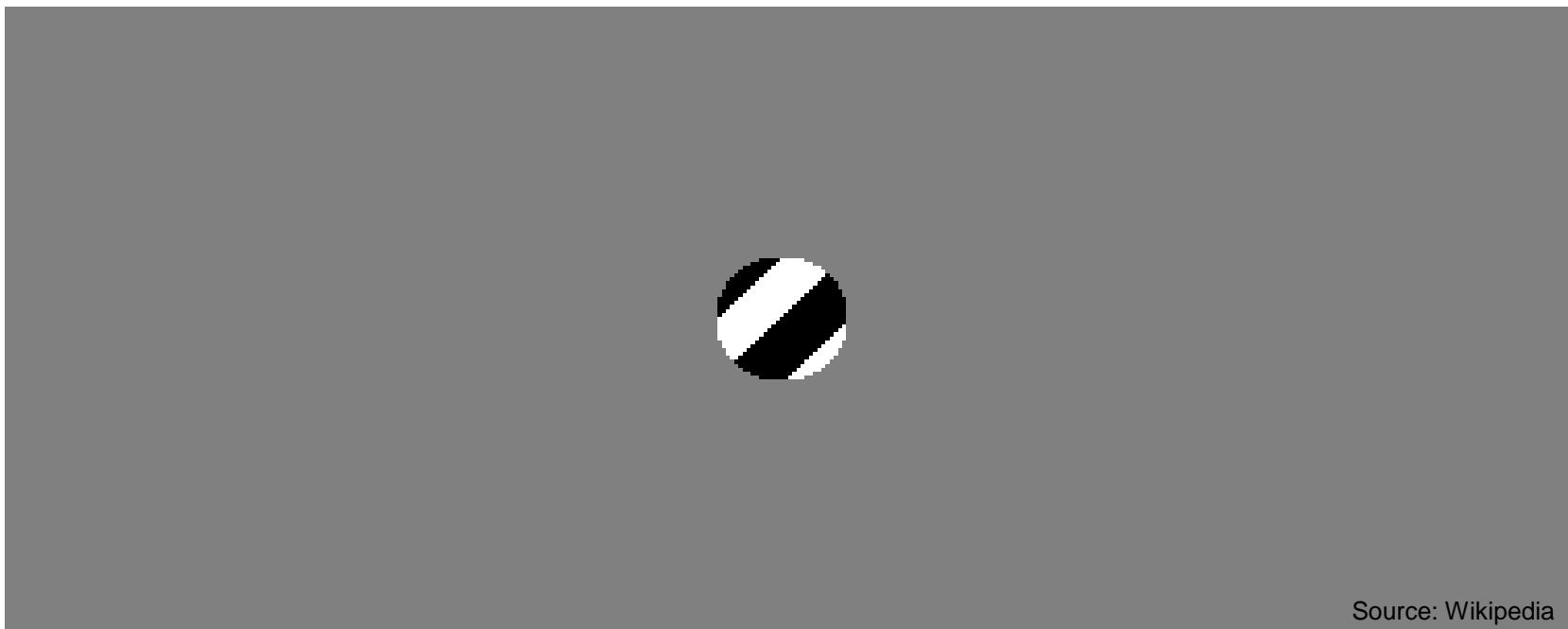
- There is a difference between 3D motion (in the scene) and 2D motion (in the image).
- Moreover, there can be a difference between the true motion in the image and the apparent motion.
- How comes that there is a difference between the true and the apparent motion?



Author: Martial Hebert

Ambiguities – the aperture problem

- The optical flow is usually not uniquely determined from the image data alone. A black pixel potentially could have moved to any black pixel in the other frame.
- Prior assumptions (e.g. that neighboring pixels move the same way) are needed for a unique solution. The neighborhood (aperture) determines the solution. Therefore, the ambiguity is also called the **aperture problem**.



Source: Wikipedia

Optic flow constraint

- The most common assumption for optical flow estimation is that the gray value of a moving pixel stays constant over time.

$$I(x + u, y + v, t + 1) - I(x, y, t) = 0$$

- This constraint is nonlinear in the unknown flow, which makes estimation difficult.
- To ease optical flow estimation, we can linearize the first expression with a Taylor expansion:

x-derivative of the image

$$I(x+u, y+v, t+1) = I(x, y, t) + I_x u + I_y v + I_t + O(u^2, v^2)$$

- This leads to the (linearized) **optic flow constraint**:

$$I_x u + I_y v + I_t = 0$$

- This linearization is only sufficiently precise, if the images are smooth (locally linear) and the flow is small.

- Optic flow constraint: $I_x u + I_y v + I_t = 0$
- We see the aperture problem: regarding a single pixel, we cannot uniquely determine its flow vector. There is only one equation, but two unknowns.
- We need a second assumption to obtain a unique solution.
- The **Lucas-Kanade method** assumes that the motion is constant within a local neighborhood.
- We then obtain multiple equations for the two unknowns, i.e., an over-determined system of equations:

$$I_x(x', y', t)u + I_y(x', y', t)v + I_t(x', y', t) = 0 \quad \forall x', y' \in \mathcal{R}(x, y)$$

Lucas-Kanade method

- In general, such an over-determined system has not a solution anymore.
- Instead we seek a solution that minimizes the total squared error:

$$\operatorname{argmin}_{u,v} \sum_{x,y \in \mathcal{R}} (I_x(x,y)u + I_y(x,y)v + I_t(x,y))^2$$

- The necessary condition for a minimum is:

$$\frac{\partial E}{\partial u} = 2 \sum_{x,y \in \mathcal{R}} (I_x(x,y)u + I_y(x,y)v + I_t(x,y))I_x(x,y) = 0$$

$$\frac{\partial E}{\partial v} = 2 \sum_{x,y \in \mathcal{R}} (I_x(x,y)u + I_y(x,y)v + I_t(x,y))I_y(x,y) = 0$$

- This leads to a linear system at each pixel

$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{pmatrix}$$

Lucas-Kanade method

- Instead of a uniformly weighted, box-shaped window, we can also use a Gaussian window.
- The sums weighted by the window function come down to a convolution:

$$\begin{pmatrix} K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & K_\rho * I_y^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -K_\rho * I_x I_t \\ -K_\rho * I_y I_t \end{pmatrix}$$

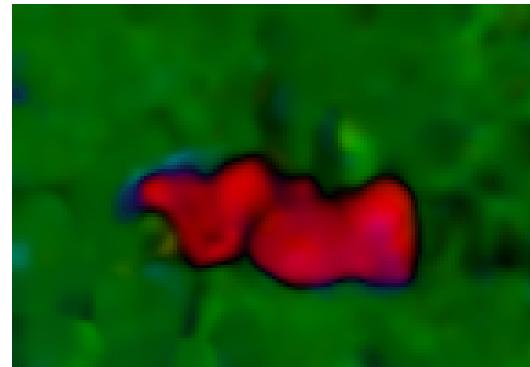
- A unique solution will be obtained only if the system is not singular.
- What does this mean?
 - The local neighborhood must contain non-parallel gradients.
 - Otherwise the aperture problem is still present.



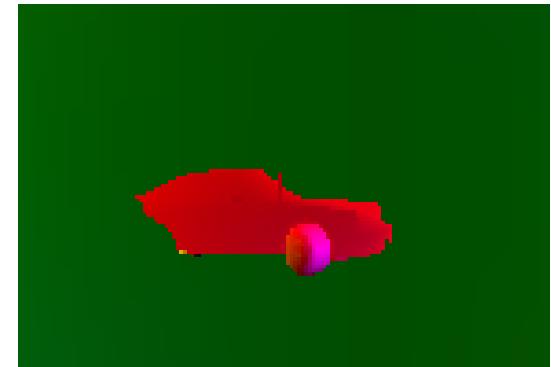
Lucas-Kanade method: example



Input sequence

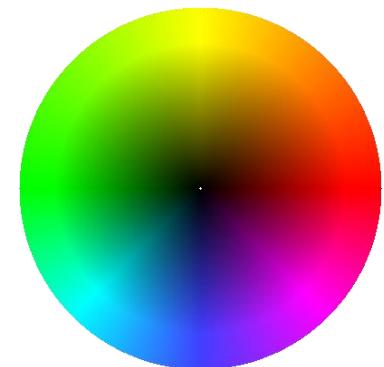


Optical flow estimated
with Lucas-Kanade
method



Correct flow field

- Drawbacks of the Lucas-Kanade method:
 - The assumption of locally constant motion often is not realistic (e.g. at motion boundaries).
 - There are no direct constraints on the smoothness of the resulting flow field.
- Advantages: fast and simple



From local to global optic flow estimation

- The Lucas-Kanade method is a “parametric” or “local” method
 - It assumes a parametric motion model.
 - The parameters are estimated from data in a local neighborhood of each point.
 - ➔ Point estimates are computed independently.
- In contrast, global methods have a global dependency between points due to a global smoothness assumption.
- A global solution is usually derived with variational methods.
- Most basic variational model: Horn-Schunck method
- This model can be further extended to yield highly accurate estimates in many practical situations.

Horn-Schunck model

- The first assumption is the same as in the Lucas-Kanade method: gray value constancy leading to the optic flow constraint.

$$I_x u + I_y v + I_z = 0$$

- The second assumption is different. The Horn-Schunck model assumes global smoothness of the flow field:

$$|\nabla u|^2 + |\nabla v|^2 \rightarrow \min$$

- Both assumptions can be combined in an energy functional:

$$E(u, v) = \int_{\Omega} (I_x u + I_y v + I_z)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2) dx dy$$

- The sought optical flow is the minimizer of this energy.
- Note the similarities to the denoising model from the previous class.

- We can find the minimizer of this functional with the calculus of variation.
- The energy is convex \rightarrow we get a unique global optimum.
- With the Gâteaux derivatives

$$\frac{d}{d\epsilon} E(u(x) + \epsilon h(x), v(x)) \Big|_{\epsilon=0} = 0 \quad \forall h(x)$$

$$\frac{d}{d\epsilon} E(u(x), v(x) + \epsilon h(x)) \Big|_{\epsilon=0} = 0 \quad \forall h(x)$$

we obtain the Euler-Lagrange equations

$$(I_x u + I_y v + I_z) I_x - \alpha \Delta u = 0$$

$$(I_x u + I_y v + I_z) I_y - \alpha \Delta v = 0$$

Laplacian: $\Delta v = v_{xx} + v_{yy}$

Linear system of equations

- The discretized versions of these equations read

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} - \frac{1}{\alpha}(I_x u_{i,j} + I_y v_{i,j} + I_z) I_x = 0$$

$$\frac{v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}}{h^2} - \frac{1}{\alpha}(I_x u_{i,j} + I_y v_{i,j} + I_z) I_y = 0$$

- The equations are linear in (u, v) . Thus we obtain a large linear system to solve (as in the denoising case).
- This system can be written in matrix-vector notation: $A^h w = b$, where $w := (u, v)$.
- As in the denoising case, the system matrix is sparse with only few diagonals different from zero.

Linear system of equations

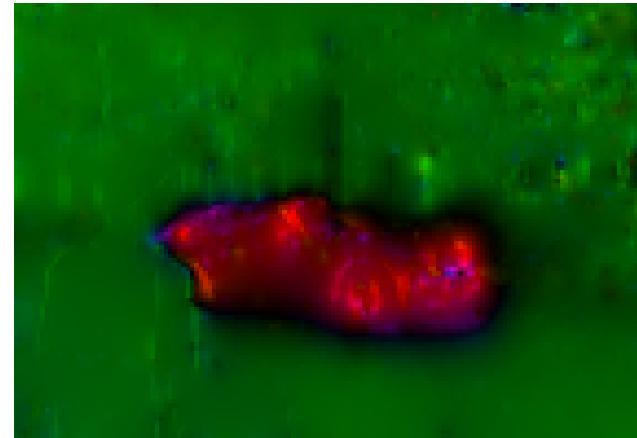
$$A^h = \left(\begin{array}{cc|cc} d & & d & \\ & d & & d \\ & & d & d \\ & & & d \\ \hline & d & d & d \\ & & d & d \\ & & & d \\ & & & d \end{array} \right) + \left(\begin{array}{cc|cc} s & s & s & \\ s & s & s & s \\ s & s & s & \\ s & s & s & s \\ \hline s & s & s & \\ s & s & s & s \\ s & s & s & s \\ s & s & s & s \end{array} \right)$$

- The blocks structure is due to the coupling of u and v .
- The **data term** contributes only to block main diagonals.
- The **smoothness term** has four additional block off-diagonals (coupling of neighboring pixels).
- We can solve this linear system with the linear solvers from class 4 (Gauss-Seidel, SOR, Multigrid).

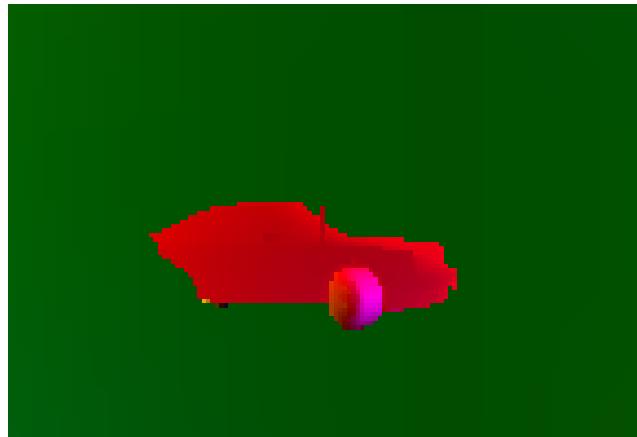
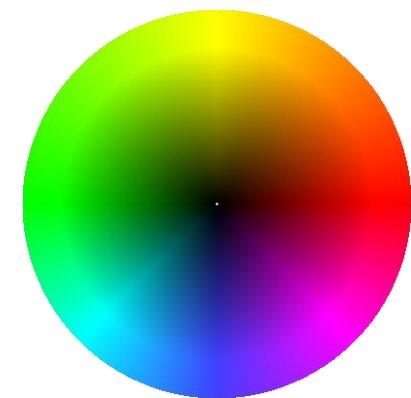
Results with Horn-Schunck method



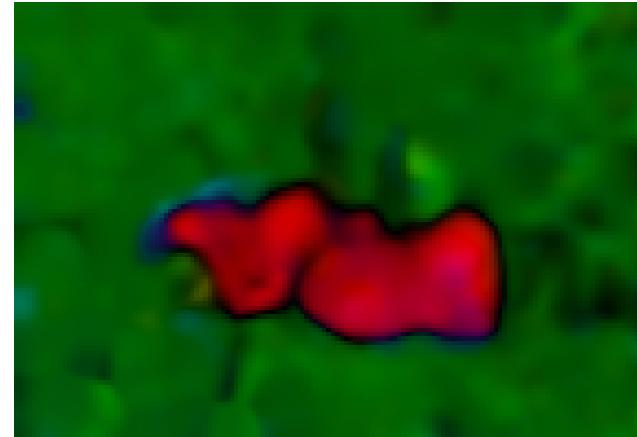
Input sequence



Horn-Schunck (10.4°)



Ground truth



Lucas-Kanade (10.54°)

Average angular error

- An established measure of how accurately the optical flow has been estimated is the **average angular error**.
- This requires, of course, knowledge about the true motion field, the so-called **ground truth**. For some synthetic sequences such ground truth data is available.
- The average angular error measures the 3D angle between the correct and the estimated flow vectors:

$$\text{AAE} := \frac{1}{n} \sum_{i=1}^n \arccos \left(\frac{(u_c)_i u_i + (v_c)_i v_i + 1}{\sqrt{((u_c)_i^2 + (v_c)_i^2 + 1)(u_i^2 + v_i^2 + 1)}} \right)$$

- Note that this also captures errors in the length of the 2D flow vector.
- An alternative measure is the **end point error**:

$$\text{EPE} := \frac{1}{n} \sum_{i=1}^n \sqrt{(u_i - (u_c)_i)^2 + (v_i - (v_c)_i)^2}$$

Middlebury flow benchmark

- Available at <http://vision.middlebury.edu/flow/>



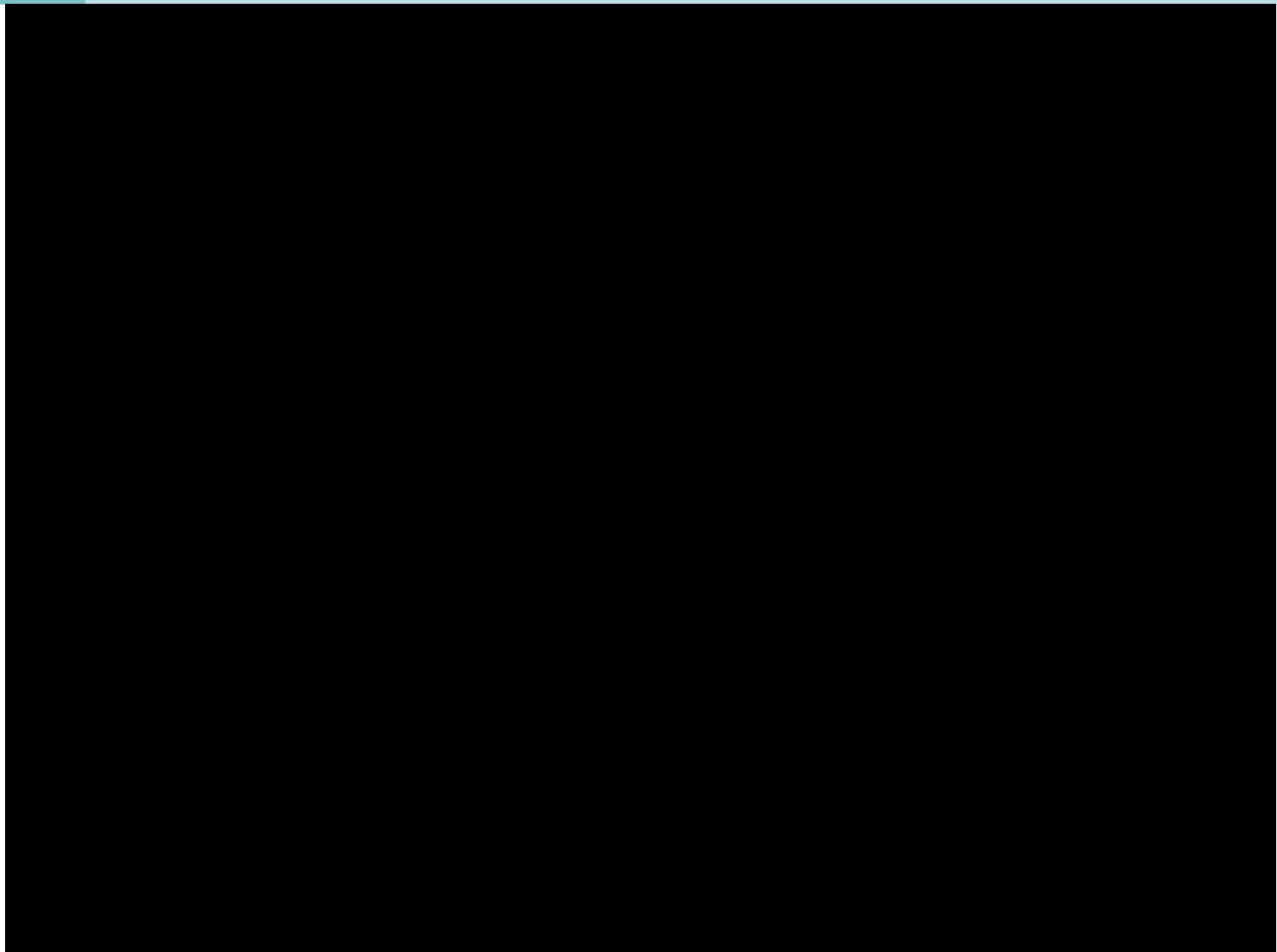
8 training sequences with ground truth (for parameter tuning)



8 test sequences with ground truth (for evaluation)

Too few test data → overfitting

Sintel – an open source movie



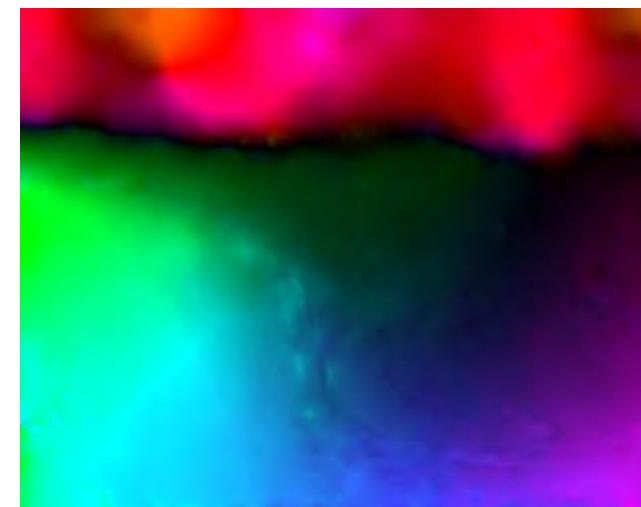
Sample frames and optical flow ground truth (Butler et al. ECCV 2012)



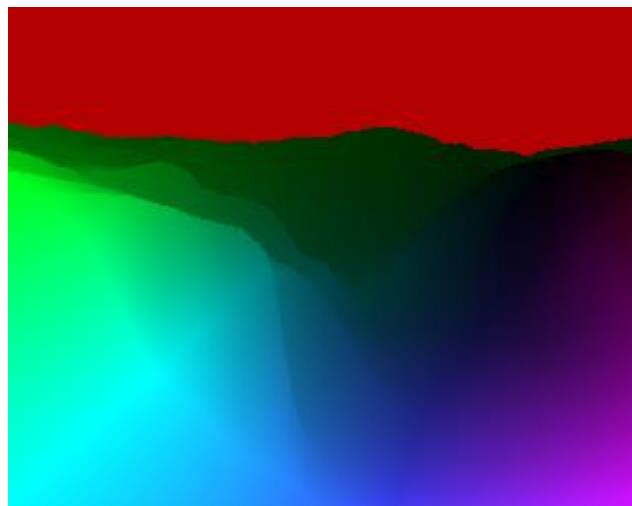
Results with Horn-Schunck method



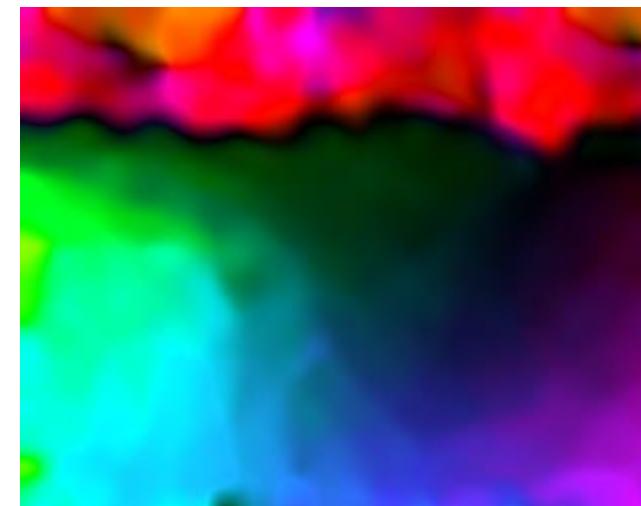
Input sequence



Horn-Schunck (7.17°)



Ground truth

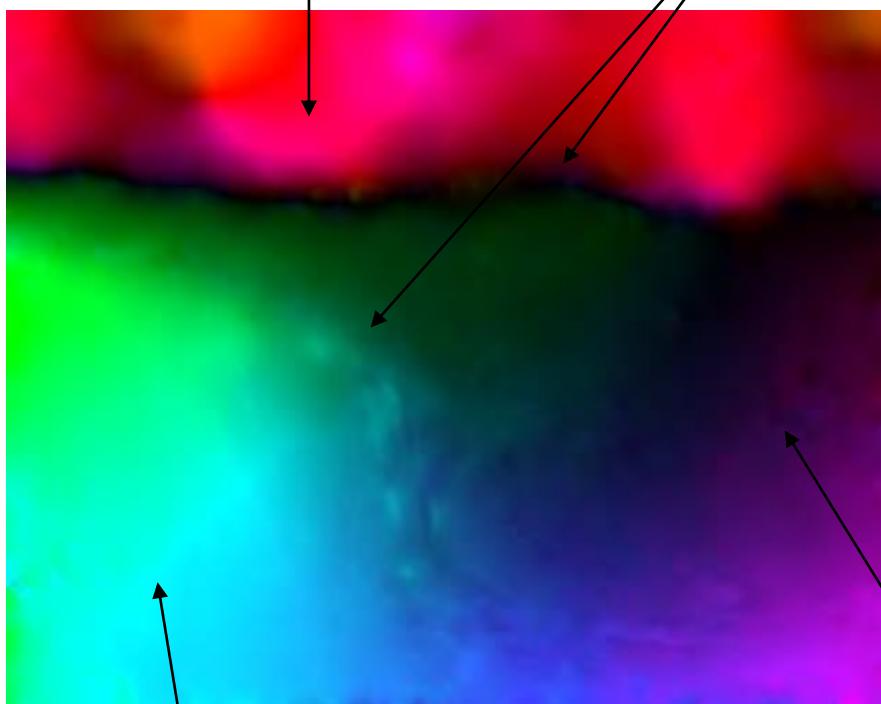


Lucas-Kanade (8.78°)

Limitations of the Horn-Schunck method

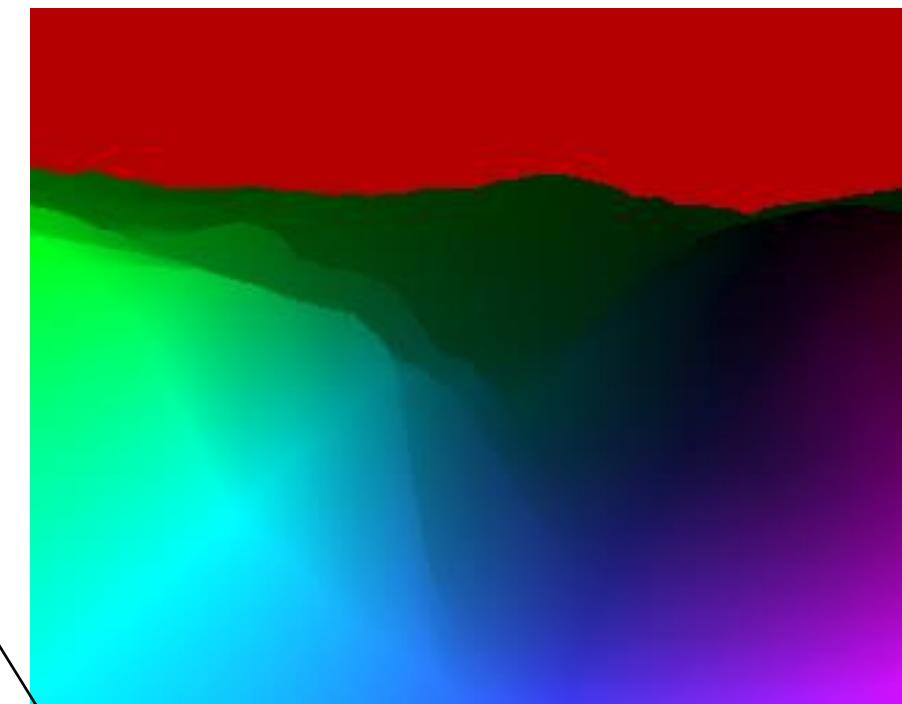
Cloud region not well estimated due to illumination changes

Motion discontinuities blurred



Underestimation of large displacements

Outliers due to non-Gaussian noise or occlusion



Tackling motion discontinuities and occlusions

- As in the denoising case we can consider a **robust function** applied to the smoothness term to allow for motion discontinuities, for instance:

$$\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$$

- Occlusions can be approached by such a robust function applied to the data term. The energy then reads (Mémin-Pérez 1998):

$$E(u, v) = \int_{\Omega} \Psi((I_x u + I_y v + I_z)^2) + \alpha \Psi(|\nabla u|^2 + |\nabla v|^2) \, dx dy$$

- We obtain the Euler-Lagrange equations:

$$\Psi'((I_x u + I_y v + I_z)^2) (I_x u + I_y v + I_z) I_x - \alpha \operatorname{div} (\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla u) = 0$$

$$\Psi'((I_x u + I_y v + I_z)^2) (I_x u + I_y v + I_z) I_y - \alpha \operatorname{div} (\Psi'(|\nabla u|^2 + |\nabla v|^2) \nabla v) = 0$$

- This nonlinear system can be solved with **lagged nonlinearity**.

Illumination changes



"Schefflera" from Middlebury benchmark



Two frames from a Miss Marple movie

Typically caused by:

- Shadows
- light source flickering
- self-adaptive cameras
- different viewing angles and non-Lambertian surfaces

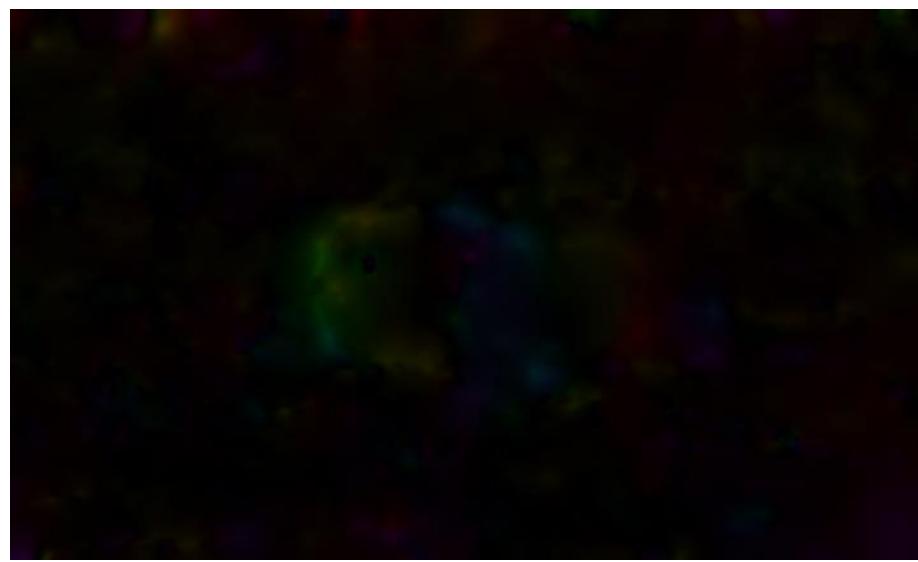
Large motion

- So far, we linearized the constancy assumptions.

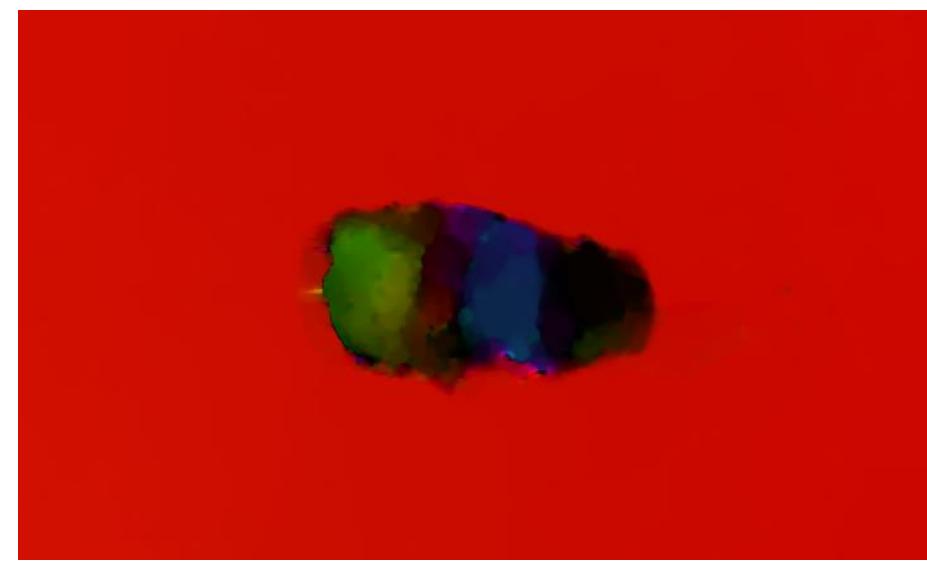
$$I(x+u, y+v, z+1) - I(x, y, z) = 0 \quad \rightarrow \quad I_x u + I_y v + I_z = 0$$

- This linearization is only valid for very small displacements (actually subpixel displacements).
- To tackle displacements larger than ~ 4 pixels we must refrain from such a linearization.
- This raises several difficulties:
 - The energy becomes non-convex, i.e., we can only find local minima.
 - The Euler-Lagrange equations get highly nonlinear.
- These problems can be handled with the Gauss-Newton method and a coarse-to-fine approach working on an image pyramid (Brox et al. 2004).
- Details in the Computer Vision course.

Large motion

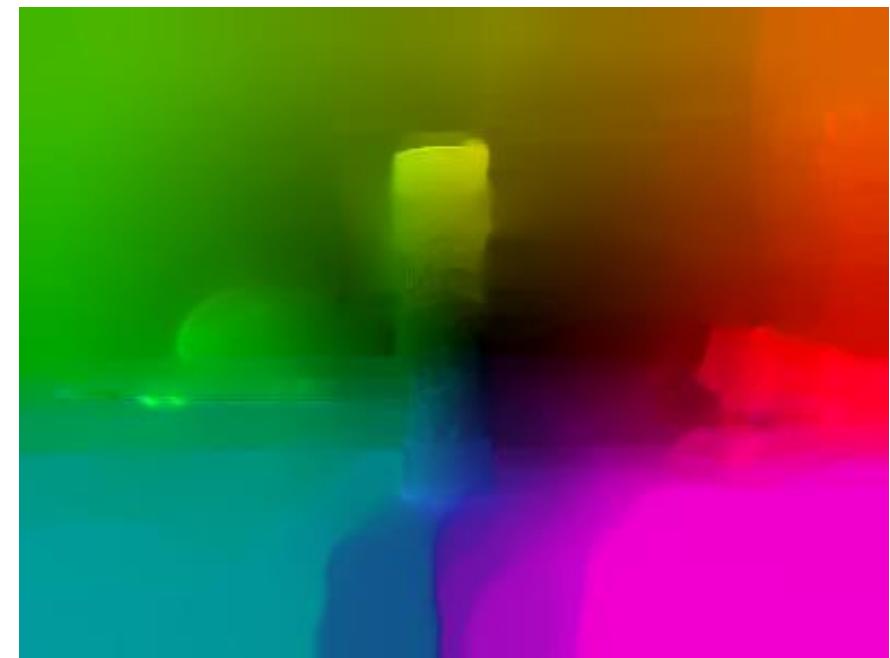


Horn&Schunck

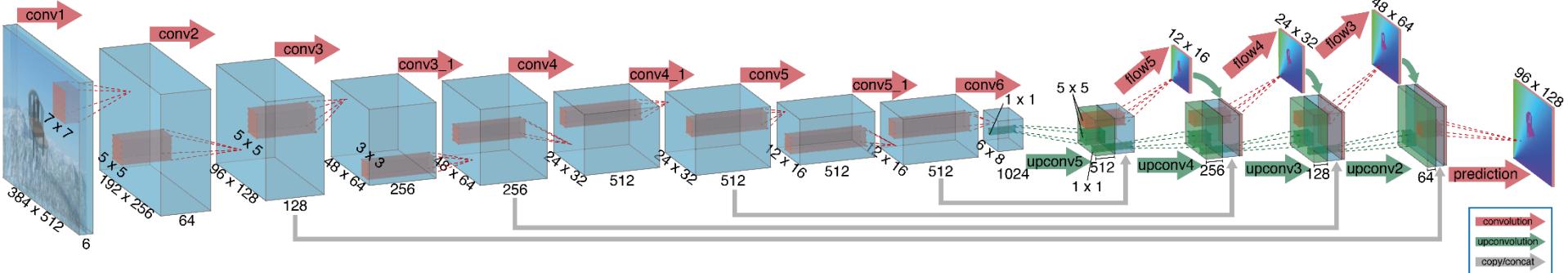


Brox et al. 2004

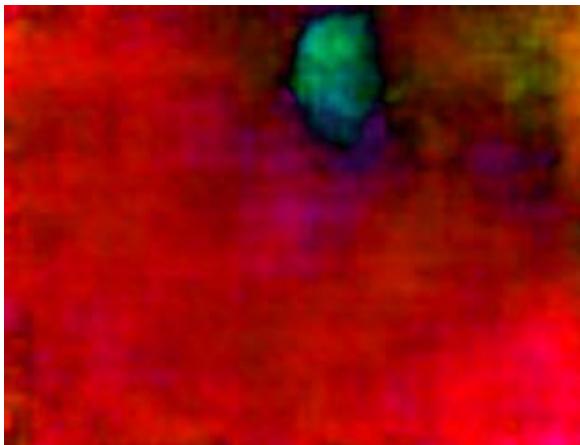
Results: zoom into a scene



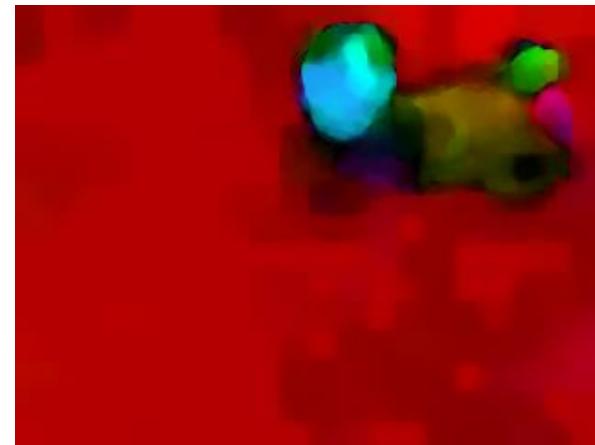
FlowNet: Learning to estimate optical flow (Dosovitskiy et al. 2015)



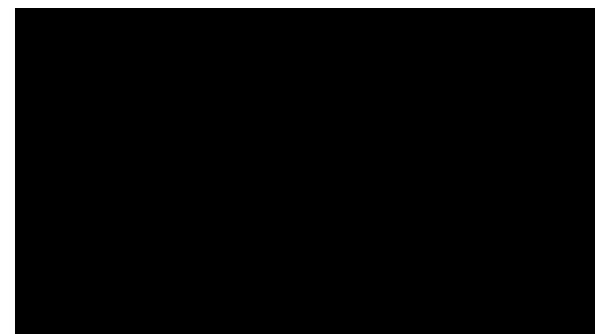
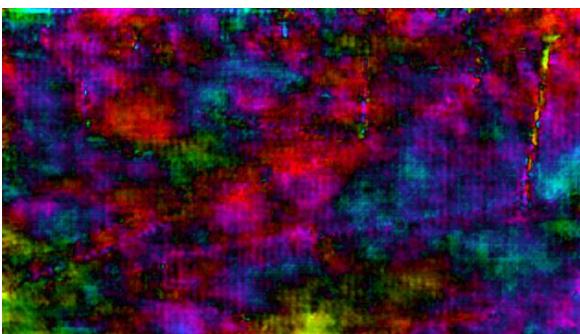
Input



FlowNet



Variational method



P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov
P. v.d. Smagt, D. Cremers, T. Brox

FlowNet: Learning Optical Flow with Convolutional Networks

- The optical flow is the apparent motion we can measure in the image plane.
- Optical flow estimation is usually based on the gray value constancy assumption.
- Local methods like the Lucas-Kanade method estimate motion parameters independently for each pixel by considering a local neighborhood.
- Global methods rather assume smoothness of the flow field and estimate the whole displacement field in one piece. The solution is found by variational techniques.
- Variational methods are still the state of the art in motion estimation
- This could change next week.

FlowNet 2.0: Preliminary numbers on Sintel.train.clean

	EPE	runtime
Variational method (Revaud et al. 2015)	2.4	15-20s
FlowNet 1.0 (Dosovitskiy et al. 2015)	4.5	80ms (12 fps)
FlowNet 2.0	2.1	90ms (11 fps)
FlowNet 2.0	2.6	28ms (36 fps)
FlowNet 2.0	4.5	9ms (110 fps)

References

- B. D. Lucas, T. Kanade: An iterative image registration technique with an application to stereo vision, *Proc. International Joint Conference on Artificial Intelligence*, pp. 674-679, 1981.
- B. Horn, B. Schunck: Determining optical flow, *Artificial Intelligence*, 17:185-203, 1981.
- E. Mémin, P. Pérez: Dense estimation and object-based segmentation of the optical flow with robust techniques, *IEEE Transactions on Image Processing*, 7(5):703-719, 1998.
- T. Brox, A. Bruhn, N. Papenberg, J. Weickert: Highly accurate optical flow estimation based on a theory for warping, *European Conference on Computer Vision*, Springer LNCS 3024, pages 25-36, 2004.
- D. Butler, J Wulff, G. Stanley, M. Black: A naturalistic open source movie for optical flow evaluation. *European Conference on Computer Vision*, 2012.
- A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, T. Brox: FlowNet: learning optical flow with convolutional networks, *International Conference on Computer Vision*, 2015.

Programming and theoretical assignment

- Implement the Lucas-Kanade method for optical flow estimation. For your convenience, make use of the convolution function `NFilter::filter` and the predefined filter masks `CSmooth` and `CDerivative` in `CFilter.h`.
Try your implementation with the Street sequence. For visualization you can make use of the function `flowToImage`.
Adjust the size of the Gaussian neighborhood and play with this parameter.
Presmooth the input images before computing the gradients. Play with the amount of smoothing. Can you explain why the results get better with some presmoothing?
- As an exercise for the exam, derive the Euler-Lagrange equations of the Horn and Schunck functional using Gateaux derivatives.
- Implement the basic Horn-Schunck method. You can implement it via any linear solver discussed in class 4. The SOR solver is recommended. Find a regularization parameter that gives good results. Compare the Horn-Schunck results to the Lucas-Kanade results.
- Extra assignment (optional): Extend your Horn-Schunck method by an edge preserving penalizer.

Image Processing and Computer Graphics

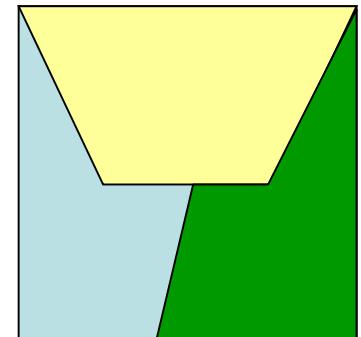
Image Processing

Class 7 Segmentation and Grouping

What is image segmentation?

- Partitioning of the image domain Ω into several (usually disjoint) regions Ω_i

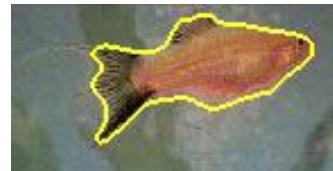
$$\Omega = \bigcup_i \Omega_i \quad \Omega_i \cap \Omega_j = \emptyset \quad \forall i \neq j$$



- Frequent special case: two-region segmentation (foreground-background)
- Important difference to **edge detection**: requires closed contours
- Edge detection is only one part of the solution as it provides pieces of potential contours.



What makes a segmentation a good segmentation?



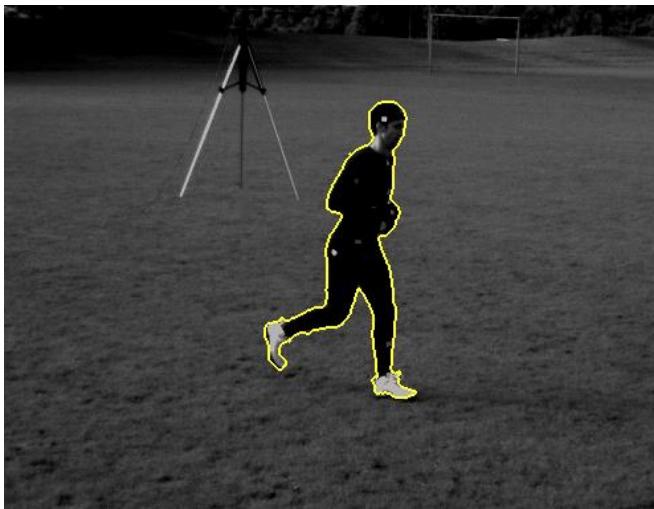
- There are exponentially many possibilities to partition an image.
- Ideally, we wish a hierarchical decomposition of a scene in its objects and their parts → **object segmentation**
- This is impossible from static images without prior knowledge on the appearance of objects
- But image segmentation can also be seen just as the grouping process that combines pixels with similar appearance to regions → **superpixels**



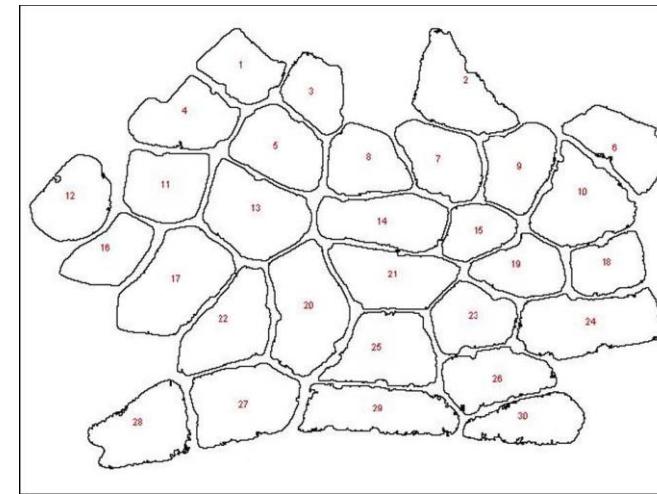
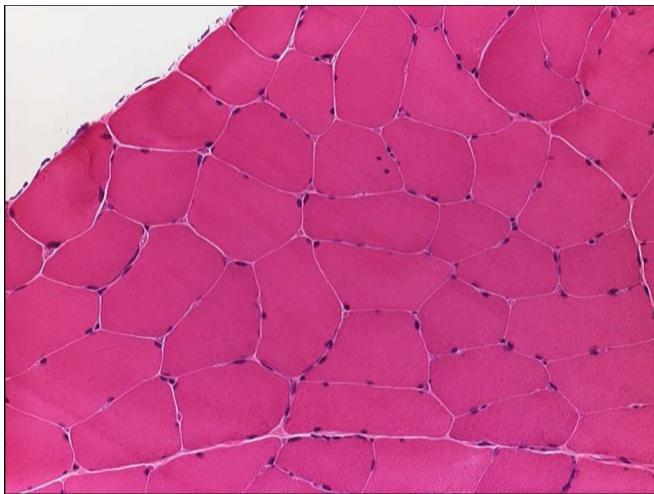
Arbelaez et al. CVPR 09

Segmentation with an underlying application

- Track the shape of a human body (Brox et al. 2007)



- Find the rims of muscle fibers (Kim et al. 2007)



Segmentation with an underlying application

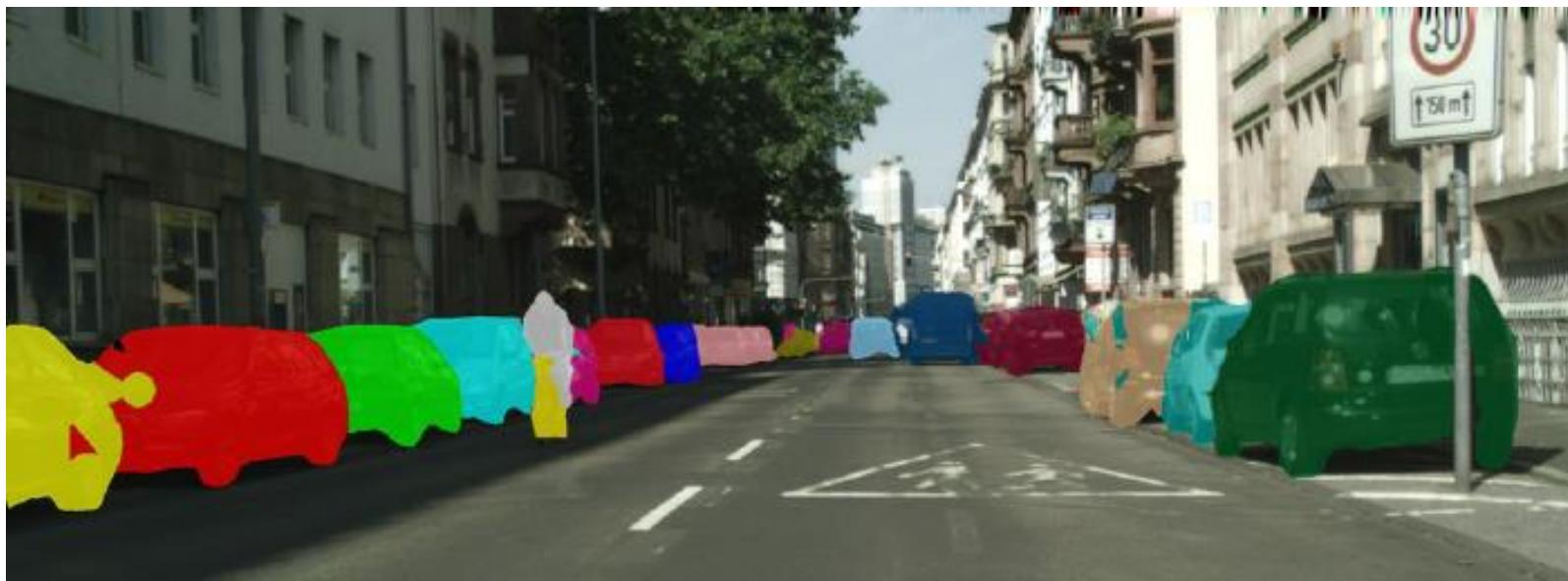
- Object class segmentation (Carreira et al. 2014)



Segmentation as a learning task



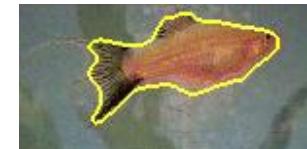
Road segmentation (Oliveira et al. 2016)



Instance segmentation (Uhrig et al. 2016)

Feature space

- Various features can be used to distinguish one region from another
 - Intensity
 - Color
 - Texture
 - Motion in videos
 - Disparity in stereo images
 - Depth in depth cameras
- We can distinguish **first-order features** and **second-order features**.
- First-order features are provided directly by the sensor, while second-order features must be derived from first-order data (texture, motion, disparity).
- Second-order features are usually not precisely localized (texture) and/or are not densely available with full confidence (motion, disparity)

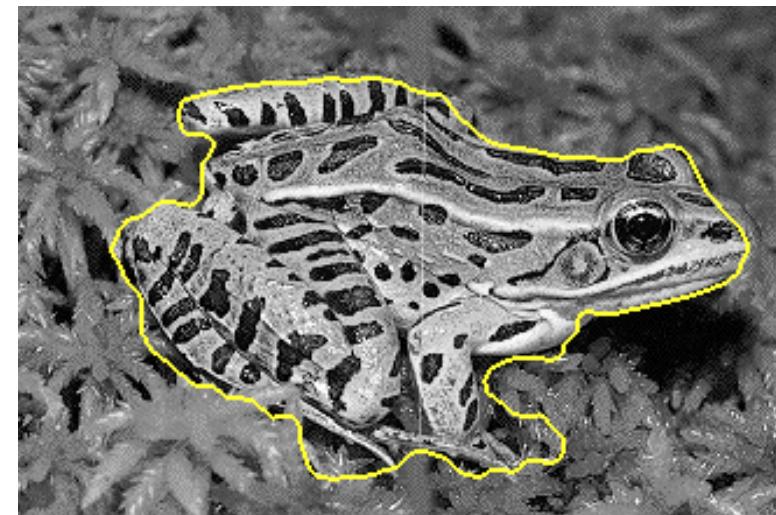
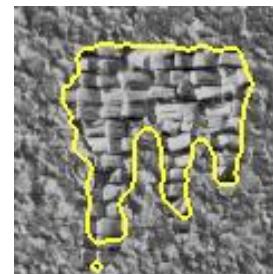
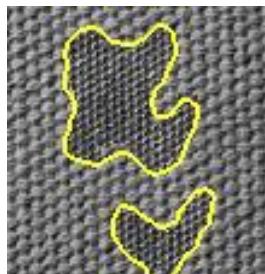


Features for segmentation: intensity/color



Author: Mikaël Rousson

Features for segmentation: texture



Features for segmentation: motion

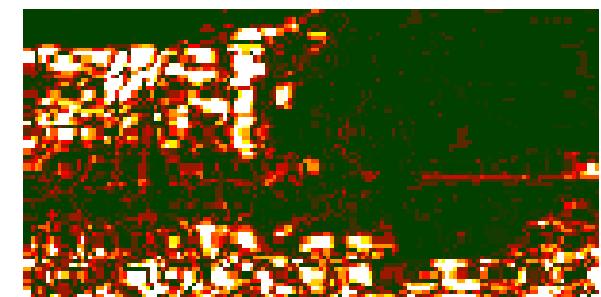
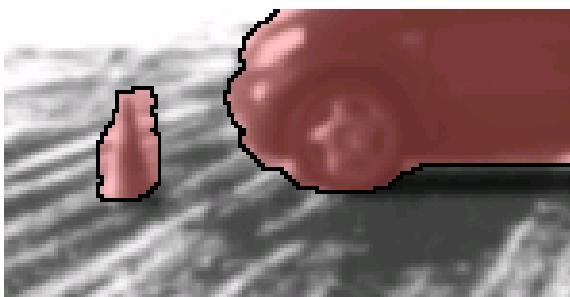
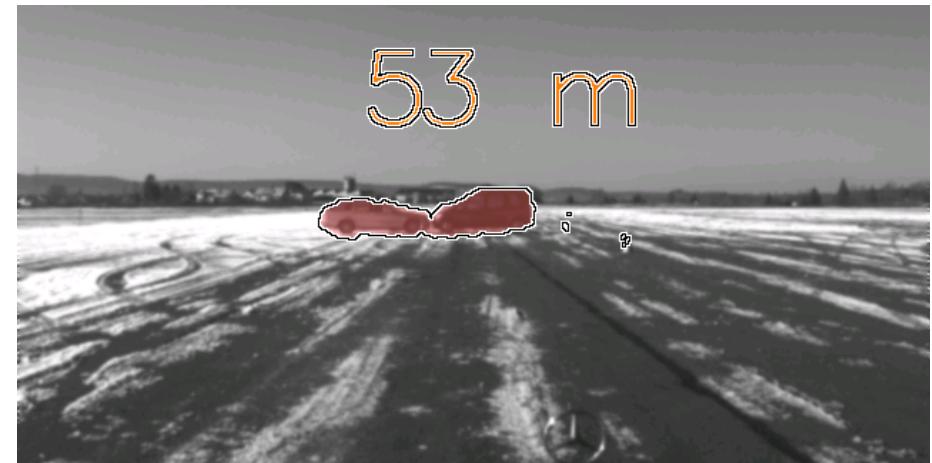


Author: Daniel Cremers



Author: Thomas Brox

Features for segmentation: motion



Author: Andreas Wedel

Depth images



Depth and color image from the Microsoft Kinect camera



Depth estimate and color image from a stereo camera

Edge-based vs. region-based methods

- Given the feature space, segmentation methods can be classified into **edge-based techniques** and **region-based techniques**.
- Edge-based techniques employ an **edge indicator** or **pairwise similarities** between pixels. Then they fit closed contours, such that the contour coincides with strong edges (low similarities).
- Region-based techniques employ a **statistical model** of each region. Regions should be as homogeneous as possible according to the model. This automatically includes that pixels in different regions are maximally different.
- There are region-based techniques based on local statistics that approach edge-based techniques in the limit.

- The most simple way to come to a segmentation is thresholding.

- Converts an intensity image into a binary image:

$$u(x, y) = \begin{cases} 255 & I(x, y) > \theta \\ 0 & I(x, y) \leq \theta \end{cases}$$

- The two states of the binary image assign pixels to the two regions.
- Can be generalized to N regions by introducing $N - 1$ thresholds.
- Problems:
 - Often there is no threshold that separates the objects.
 - Point-based operation: the spatial context is ignored.
- Thresholding is a region-based technique with a very simple (and inflexible) region model.

Relationship to clustering

- Image segmentation is similar to clustering: assigning data points (pixels) to clusters (regions)
- There are various clustering methods.
- Most popular: **k-means clustering**
 - Initialize the pixels to belong to a random region
 - Compute the mean feature vector in each region
 - Move a pixel to another region if this decreases the total distance

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- Iterate until pixels do not move any longer
- With $K = 2$, k-means is like thresholding with an automatically determined threshold.



Author: Christopher Bishop

Clustering does not enforce spatial consistency

- Clustering methods ignore spatial context. Only the feature vector determines the assignment of a pixel, not its position in the image.
- We can add the pixel coordinates to the feature vector to enforce compact regions, but this is not equivalent to enforcing smooth region contours.



Intensity+color



Intensity+color+position

- **Region growing**
 - **Seed points** represent initial regions
 - For each point on the region boundary: if a neighbor is similar enough, it is assigned to this region (the region grows)
 - Grow until there are no more similar pixels along the region boundary
- **Region merging**
 - Initially all pixels represent their own region.
 - The two most similar regions are successively merged to one larger region.
 - Repeat until a similarity threshold or a given number of regions is reached.
- Some dissimilarity criteria:
 - Euclidean distance of the features' means: $d^2(\mathcal{R}_1, \mathcal{R}_2) = (\mu_1 - \mu_2)^2$
 - Mean Euclidean distance along common boundary
 - (...)

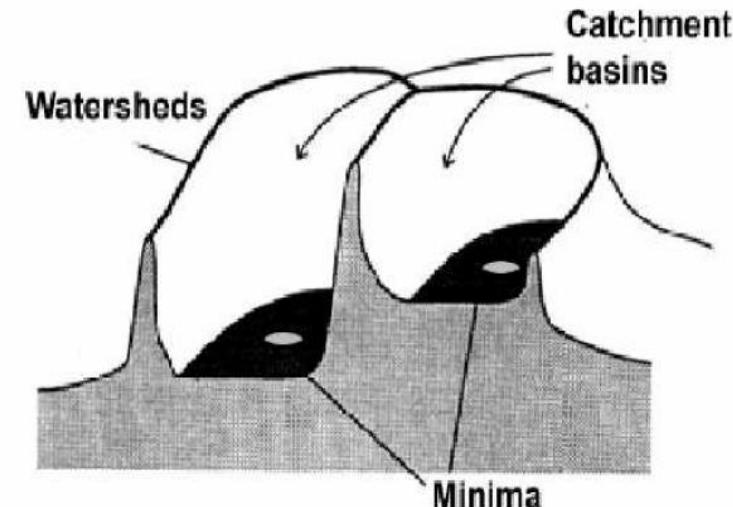
Region merging



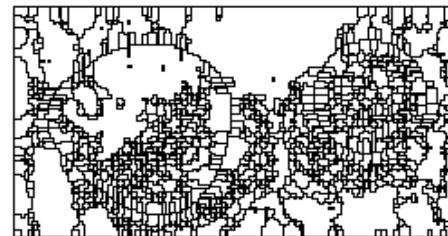
Author: unknown

Watershed segmentation

- Illustrative description: regard the **image gradient magnitude** as mountains and let it rain.
- Water flows downhill and gathers in catchment basins: the regions.
- Regions meet at the watersheds (edges),
→ region boundaries.
- Tiny gradient fluctuations result in over-segmentations
→ can be reduced by presmoothing the image



Author: P. Soille



Energy minimization: the snakes model

- Kass, Witkin, and Terzopoulos proposed the following energy functional:

$$E(C) = - \int_0^1 |\nabla I(C(s))|^2 ds + \alpha \int_0^1 |C_s(s)|^2 ds$$

- $C : [0, 1] \rightarrow \Omega$ is a parametric contour. C_s denotes the first derivative of this contour.
- The first term is also called **external energy**, since it depends on the (external) input image. The second term is called **internal energy**, since it is inherent to the model and independent of the data.
- Minimizing the external energy drives the contour to follow maxima of the gradient.
- Rather than seeking such maxima and to group them to a contour, we consider all possible contours and choose the one that best captures the maxima.

Energy minimization: the snakes model

- The external energy alone is not sufficient. It would lead to a fractal contour with infinite length.

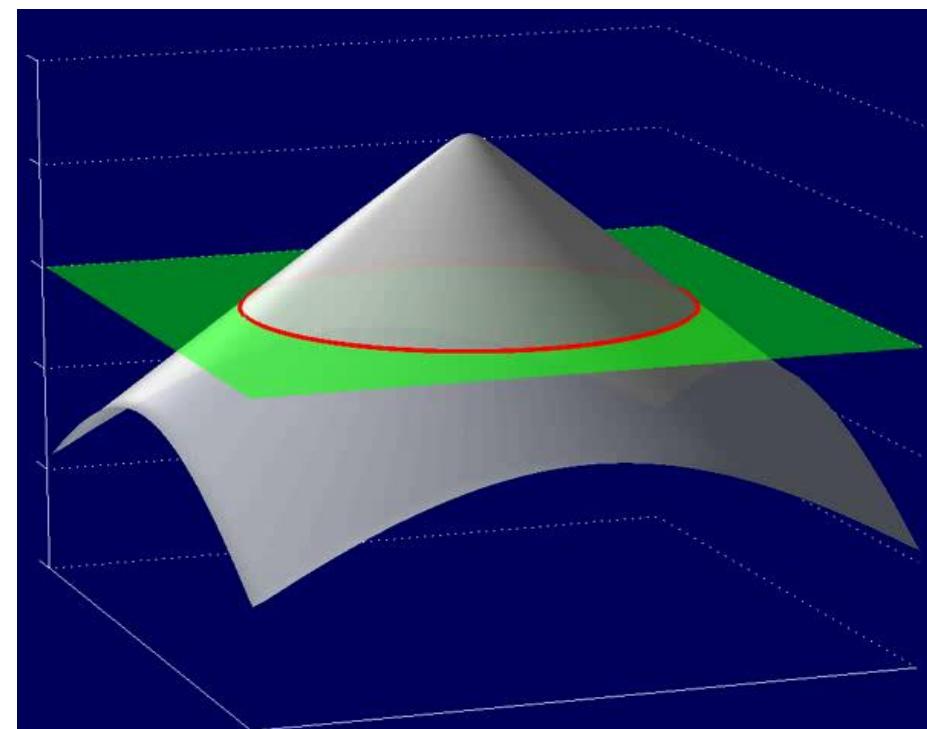
$$E_{ext}(C) = - \int_0^1 |\nabla I(C(s))|^2 ds$$

- This is avoided by the internal energy, which penalizes the length of the contour

$$E_{int}(C) = \alpha \int_0^1 |C_s(s)|^2 ds$$

- The external and internal energy together prefer a compromise of a short contour which captures as much image gradient as possible.
- Energy minimization with the calculus of variation and gradient descent
→ local minima

- Introduce an **indicator function** $\phi : \Omega \rightarrow [-1, 1]$
- The zero-level line represents the contour
$$C = \{\mathbf{x} \in \Omega \mid \phi(\mathbf{x}) = 0\}$$
- For evolving C evolve ϕ
- Allows for topological changes
- Can be applied in any dimension
- Represents the contour and the enclosed region



Author: Daniel Cremers

Region-based active contours

- Energy minimization based on regions statistics

- The energy states the optimal separation of pixel intensities:

$$E(C) = \int_{\Omega_1} (I - u_1)^2 dx + \int_{\Omega_2} (I - u_2)^2 dx + \nu |C|$$

- This is similar to k-means clustering (two-means), but with an additional constraint on the length of the separating contour.

- We can express this using an implicit representation of the contour:

$$E(\phi) = \int \phi \left((I - u_1)^2 - (I - u_2)^2 \right) + \nu |\nabla \phi| dx$$

- Given ϕ , u_1 and u_2 can be found analytically as the mean intensities inside the two regions

$$u_1 = \frac{\int H(\phi) I dx}{\int H(\phi) dx} \quad u_2 = \frac{\int (1 - H(\phi)) I dx}{\int (1 - H(\phi)) dx}$$

Region-based active contours



Author: Daniel Cremers

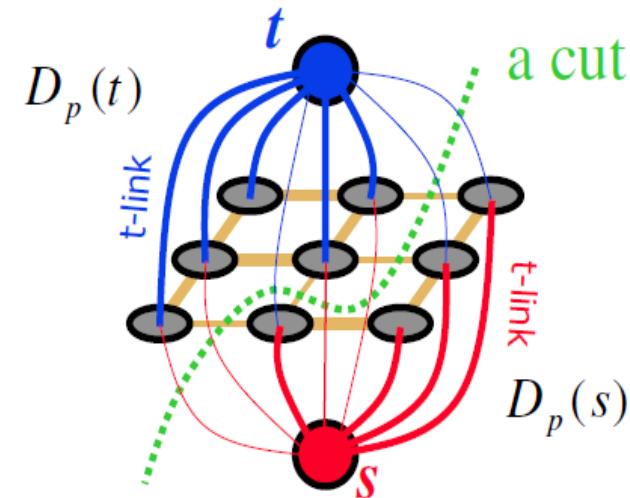
Graph cuts for segmentation

- Example of combinatorial optimization

- Graph structure for **min-cut**:

- Each pixel represented by a **node**. It is interconnected to its neighbors via **edges**.
- Neighborhoods can have different complexity.
Most simple: 4-neighborhood.
- Two extra nodes (source and target) connected to all pixel nodes

- Goal: find the minimum cut through the graph that separates source and target.
- Source and target nodes correspond to regional models. A pixel is assigned to either of the two regions.
- The connection between neighbors enforces a “regular” labeling.



Energy corresponding to a graph

- Generally the energy reads:

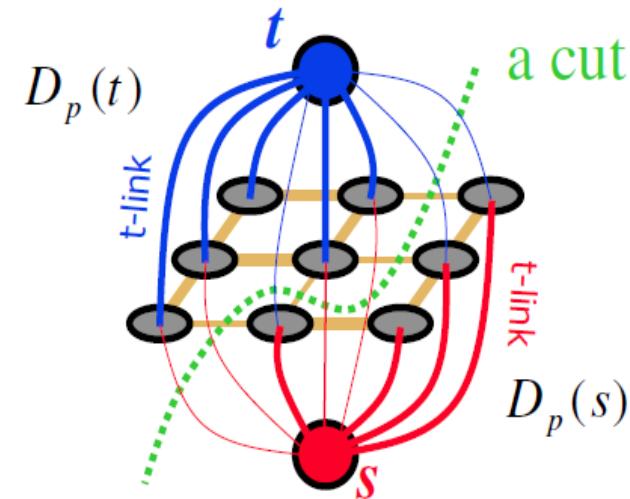
$$E(u) = \sum_i D(u_i) + \sum_{i,j \in \mathcal{N}(i)} w_{ij} \delta(u_i \neq u_j)$$

- First term comprises the links to the source and target nodes, the **t-links**.

- A simple region model is specified by the mean. This leads to the t-link weights:

$$D_i(0) = (I_i - \mu_1)^2, \quad D_i(1) = (I_i - \mu_2)^2$$

- Second term comprises the **n-links** between neighboring pixels. Usually the weights are fixed, but they can, e.g., depend on the image gradient.
- For fixed means μ_1, μ_2 , this combinatorial minimization problem can be solved in polynomial (average case: linear) time.



Semantic segmentation



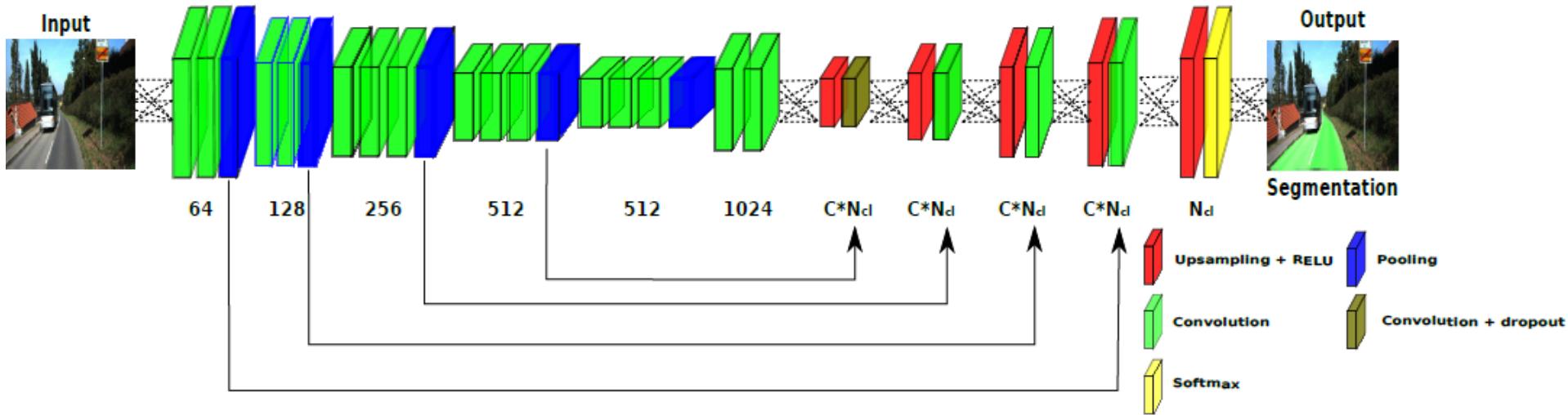
Author: Ladicky et al. 2008

- For each pixel run a classifier, which yields a score $S_i(x)$ for each class i. Here: two classes. In general, multiple classes.

$$E(C) = - \int_{\Omega_1} S_1(x) dx - \int_{\Omega_2} S_2(x) dx + \nu |C|$$

- Can be minimized with level sets or graph cuts

Semantic segmentation with deep networks



Oliveira et al. 2016

- Certain network architectures can combine classification and pixel-wise segmentation
→ usage and combination of features is learned
- Sometimes there is a CRF (graph cut) on top
- More in class 10 and in the Computer Vision course

- The goal of segmentation/grouping depends much on the application.
- We can distinguish image segmentation and object segmentation.
- Object segmentation requires special features (e.g. motion) or top-down knowledge (e.g. shape priors).
- There are many methods
 - Algorithmic approaches
 - Clustering
 - Energy models with contour constraints (contour length, shape prior)
 - Deep learning

References

- M. Kass, A. Witkin, D. Terzopoulos: Snakes: active contour models. *International Journal of Computer Vision* 1:321-331, 1988.
- D. Mumford, J. Shah: Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics* 42: 577-685, 1989.
- T. Chan, L. Vese: Active contours without edges. *IEEE Transactions on Image Processing* 10(2):266-277, 2001.
- Y. Boykov, V. Kolmogorov: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124-1137, 2004. Code available: <http://pub.ist.ac.at/~vnk/software.html>
- L. Ladicky, C. Russell, P. Kohli, P.H. Torr: Graph cut based inference with co-occurrence statistics. *European Conference on Computer Vision*, 2008.
- J. Carreira, R. Caseiro, J. Batista, C. Sminchisescu: Free-Form Region Description with Second-Order Pooling, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.
- J. Uhrig, M. Cordts, U. Franke, T. Brox: Pixel-level encoding and depth layering for instance-level semantic segmentation, *German Conference on Pattern Recognition*, 2016.
- G. Oliveira, W. Burgard, T. Brox: Efficient deep methods for monocular road segmentation, *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

Image Processing and Computer Graphics

Image Processing

Class 8

Interest points and local descriptors

Matching of local structures

- Key problem in computer vision appearing in:
 - Motion estimation
 - Camera calibration
 - Stereo
 - Image retrieval
 - Object recognition



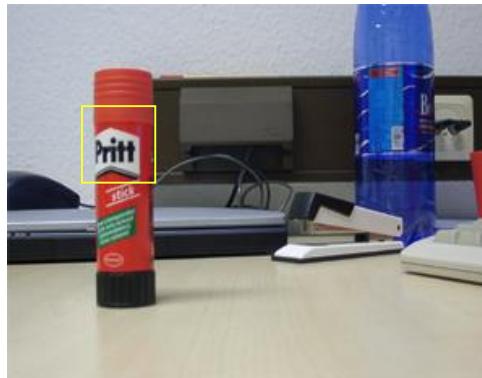
Object recognition: training image on the left, test image on the right. Matching here is quite hard.



Stereo pair: point matching needed to compute depth

Block matching

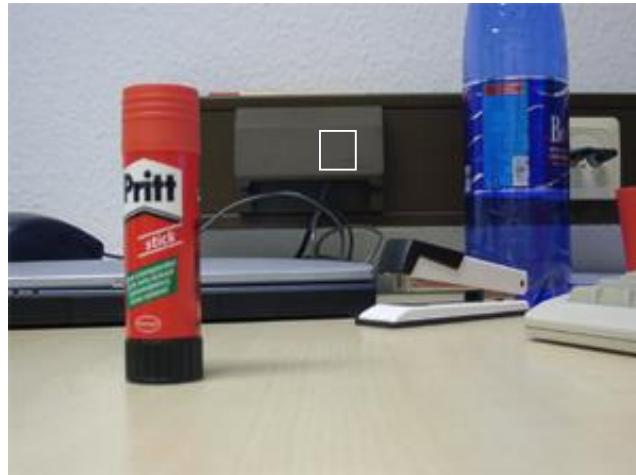
- Straightforward way to match points in images:
 - Regard the image patch around each point in image 1
 - Compare it to the image patches around all points in image 2



- Computationally expensive
 $O(kN^2)$, k : size of patch, N : size of image (in pixels)
- Not **invariant** to typical appearance changes

Interest points

- Often we need only a limited number of matches
- Idea: Do not match all points in the images, but only promising subsets
→ significantly reduced complexity
- Requirements for good interest points:
 1. Points must come with enough information for unique matching



2. Subset in image 2 must contain matches from subset in image 1

- Choose points with high information content and clear localization
→ typically corner points
- Corner detection with the structure tensor:
(Förstner-Gülch 1987, Harris-Stevens 1988)

$$J_\rho = K_\rho * (\nabla I \nabla I^\top) = \begin{pmatrix} K_\rho * I_x^2 & K_\rho * I_x I_y \\ K_\rho * I_x I_y & K_\rho * I_y^2 \end{pmatrix}$$

- Measure of cornerness (fast to compute):

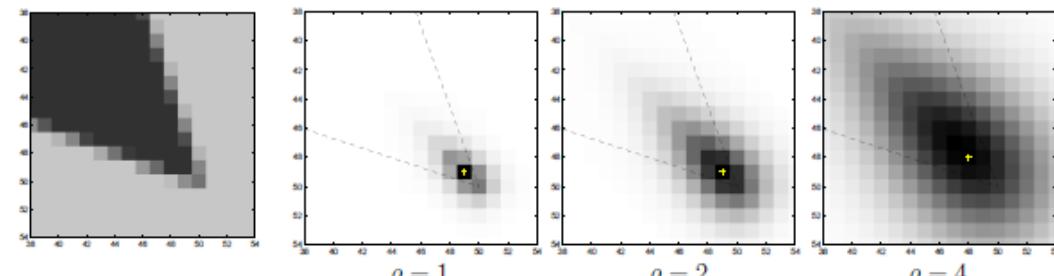
$$c = \det J_\rho - \alpha \text{tr} J_\rho$$

= gradient magnitude

- Eigenvalue decomposition of the structure tensor:

$$J_\rho = T \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} T^\top$$

$$c = \lambda_2$$



Input and second eigenvalue for different ρ

- Interpretation:

- Smoothing of J integrates gradients from the neighborhood
- Eigenvectors in T yield the dominant orientation in this neighborhood and the perpendicular orientation
- Eigenvalues yield the structure magnitude in these directions
- A large second eigenvalue indicates strong structures in multiple directions → corners

Corner detection

- Corners: local maxima of the second eigenvalue



Scale invariance

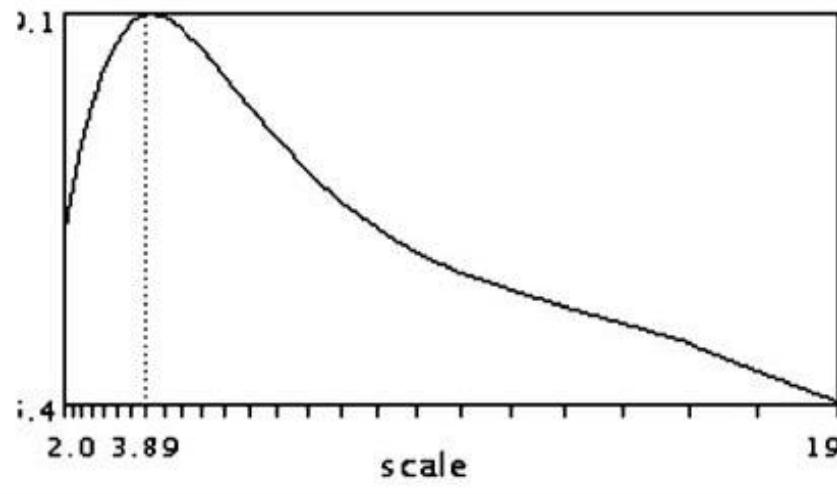
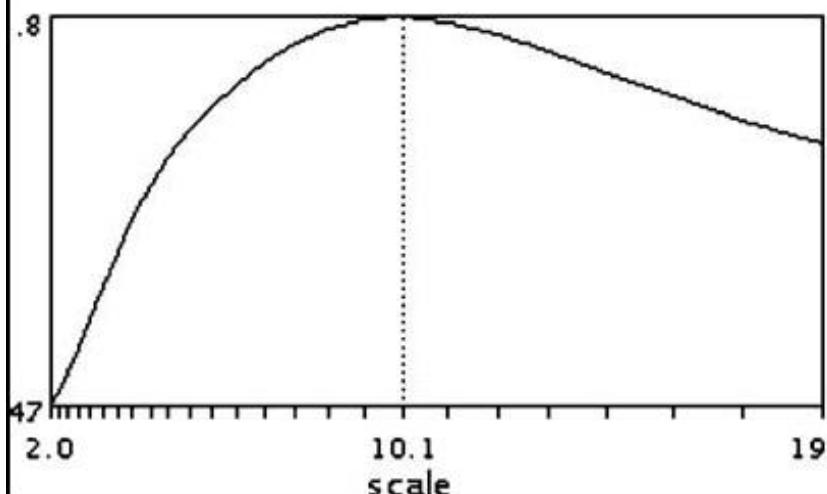
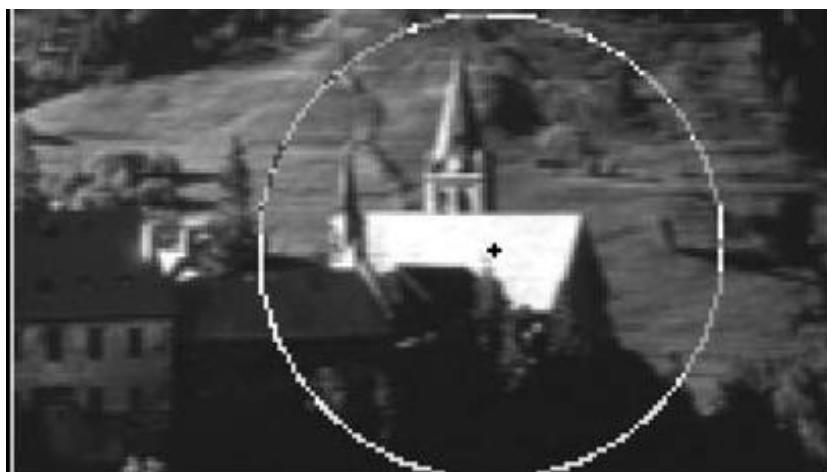
- Problem: Detected corners depend on the image scale



Characteristic scale

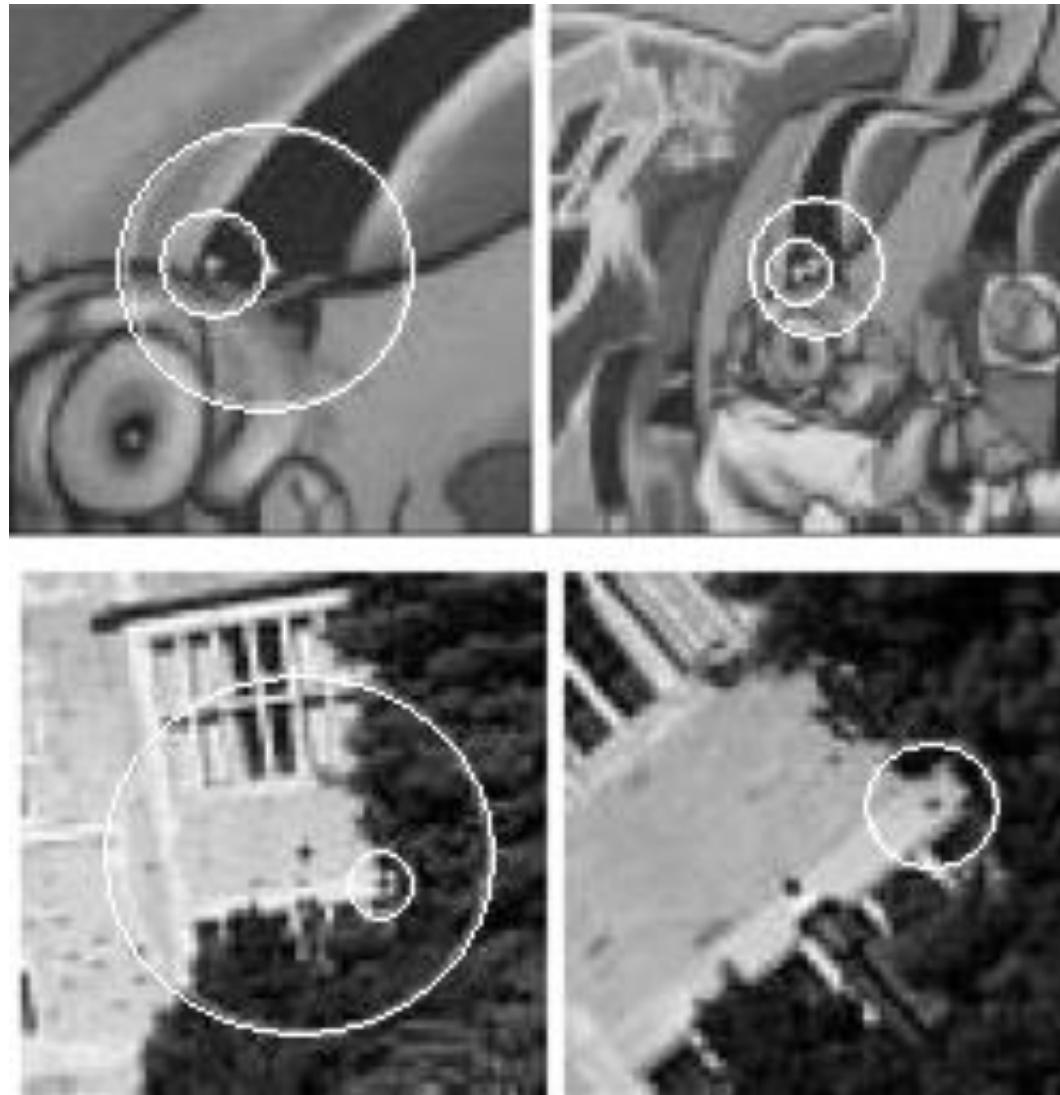
- Consider a **Gaussian pyramid** of smoothed images
- The characteristic scale can be computed based on the **Laplacian**:
(Lindeberg 1998)
$$\sigma_c = \operatorname{argmax}_\sigma (\sigma^2 \cdot |\partial_{xx}(K_\sigma * I) + \partial_{yy}(K_\sigma * I)|)$$
- Yields an estimate of the scale shift between two images
- Uniqueness is not ensured
 - There may be multiple local maxima
 - Even the global maximum need not be unique
- Maximum operator is not robust
→ a little noise can lead to a very different outcome

Characteristic scale



Authors: Krystian Mikolajczyk and Cordelia Schmid

Harris-Laplace detector



Authors: Krystian Mikolajczyk and Cordelia Schmid

- Alternative to Harris-Laplace detector
- Considers local maxima of the Laplacian in scale space:

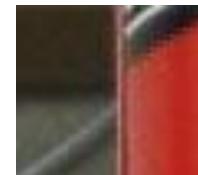
$$x^*, y^*, \sigma^* = \operatorname{argmax}_{x,y,\sigma} \left(\sigma^2 \cdot |\partial_{xx}(K_\sigma * I(x,y)) + \partial_{yy}(K_\sigma * I(x,y))| \right)$$

- Advantage: Does not mix apples and oranges (corner detector and Laplacian)
- Laplacian focuses on blobs rather than corners
 - complementary information
 - one might be interested in using both

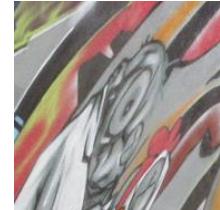
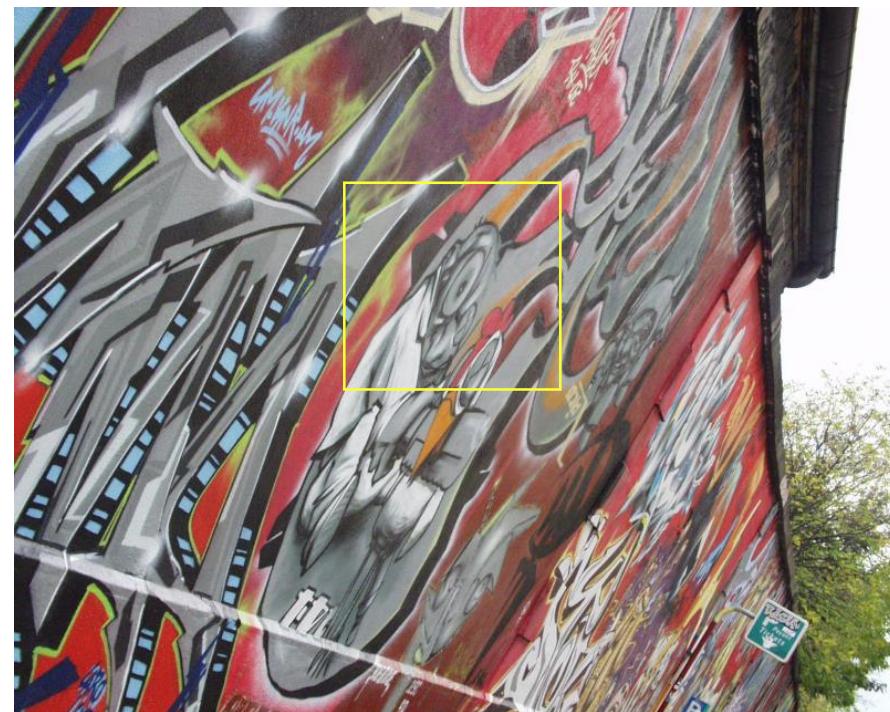
Block matching at interest points

- Positive issue of interest points:
 - Significantly reduced complexity
 - With 100 detected points in both images, one has to compare only 10000 patches instead of 96 billion(!) in 640x480 images
- Negative issues:
 - Non-dense displacement fields (important matches might be missed)
 - Corresponding patches can be slightly shifted
- Further problems (independent of interest points)
 - Patches in both images may look very different due to:
 - Rotations
 - Projective transformations (different viewing angles)
 - Lighting changes (shadows, flickering)
 - Blurring
 - Subpixel accuracy not available

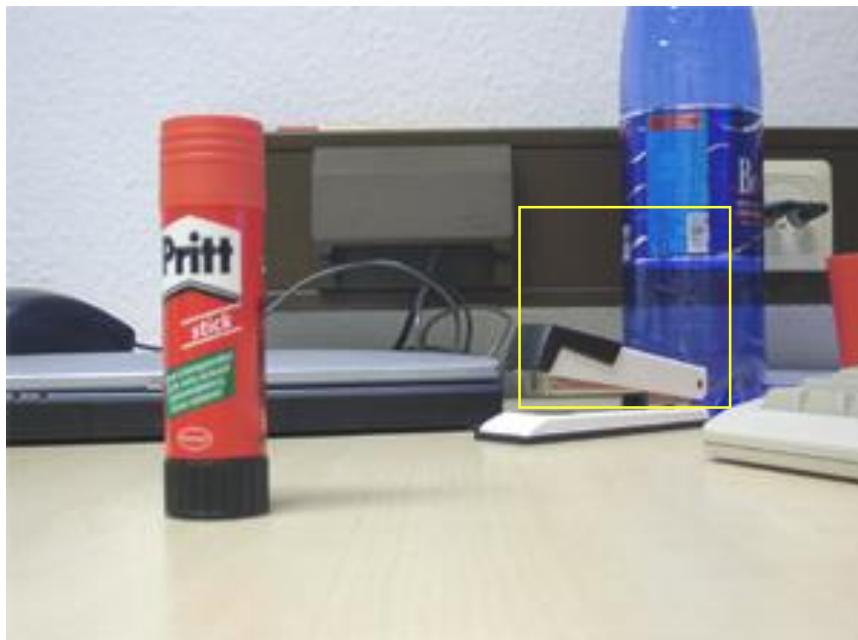
Example: rotation and scaling



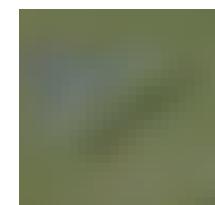
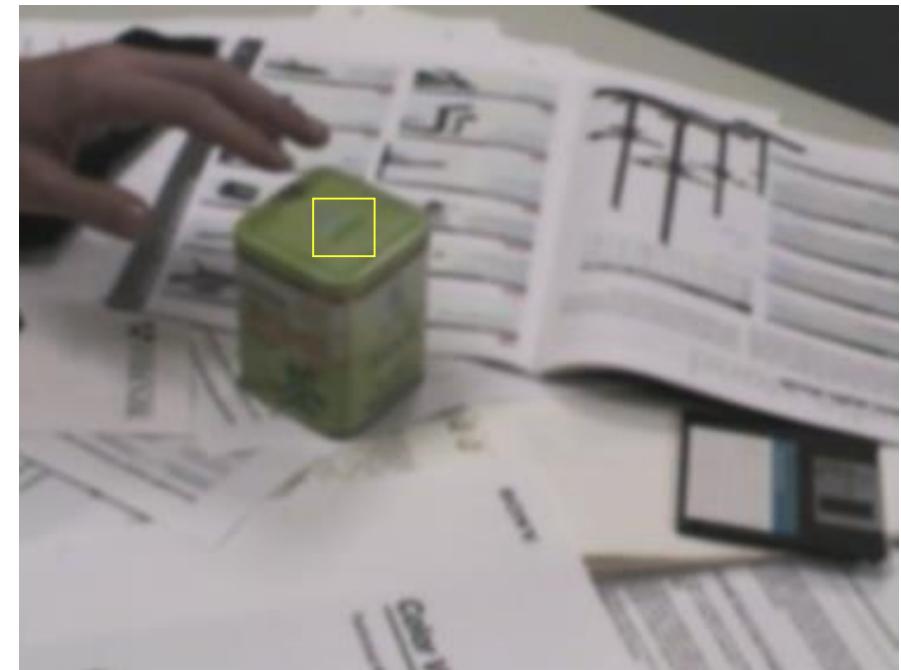
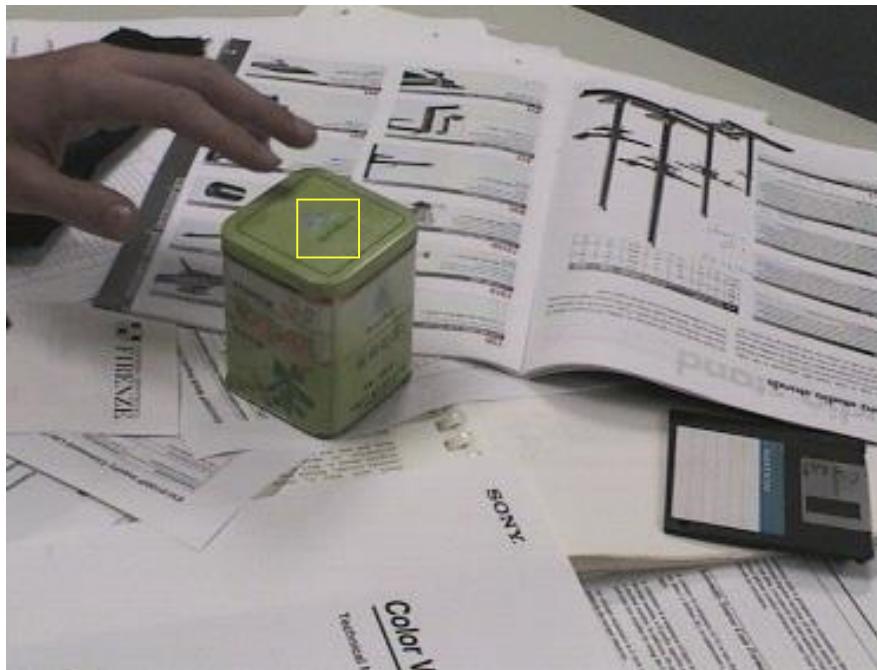
Example: projective transformation



Example: lighting changes



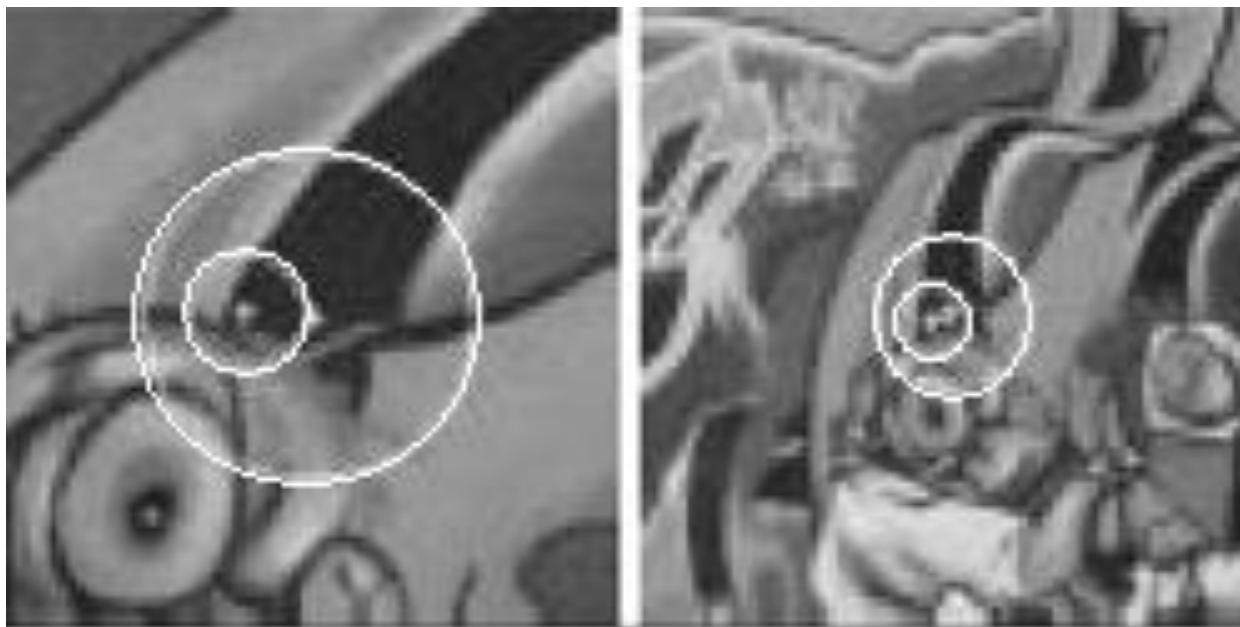
Example: blurring



- **Local descriptors:** vectors that contain information about the local neighborhood of an interest point
- Simplest local descriptor: block of a certain size centered at the interest point
- Goal: design local descriptors that are **invariant** under the mentioned transformations

Providing scale invariance

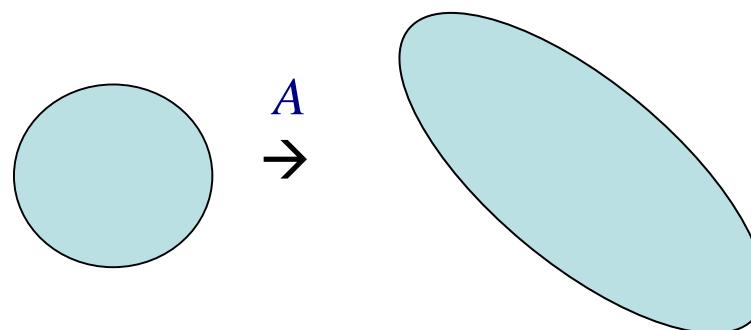
- Use characteristic scale from interest point detection
- Choose and normalize the size of the blocks, such that structures have the same scale in both images



- Affine transformation:

$$f(x) = Ax + t$$

- Maps a circle to an ellipse (or vice-versa):



- Approximation of a projective transformation
- Parameters can be estimated, e.g., from a region detector (maximally stable extremal regions)

Affine region detector

- Maximally stable extremal regions
(Matas et al. 2002)
 - Regions encircled by large gradients
 - Obtained by watershed-like algorithm

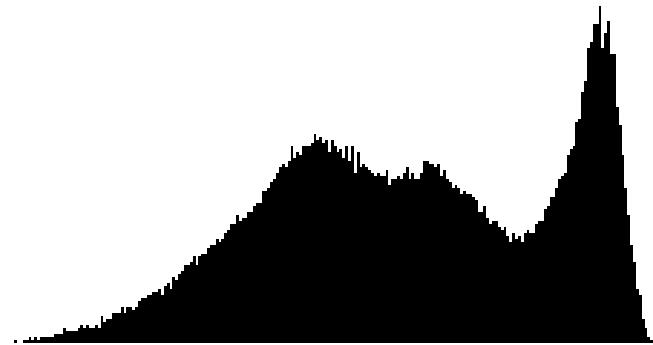


Maximally stable extremal regions and fitted ellipses. Author: Andrea Vedaldi

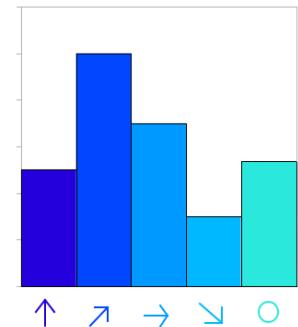
- Apart from scale also yields elongation of fitted ellipses
→ allows for affine invariance

Histograms

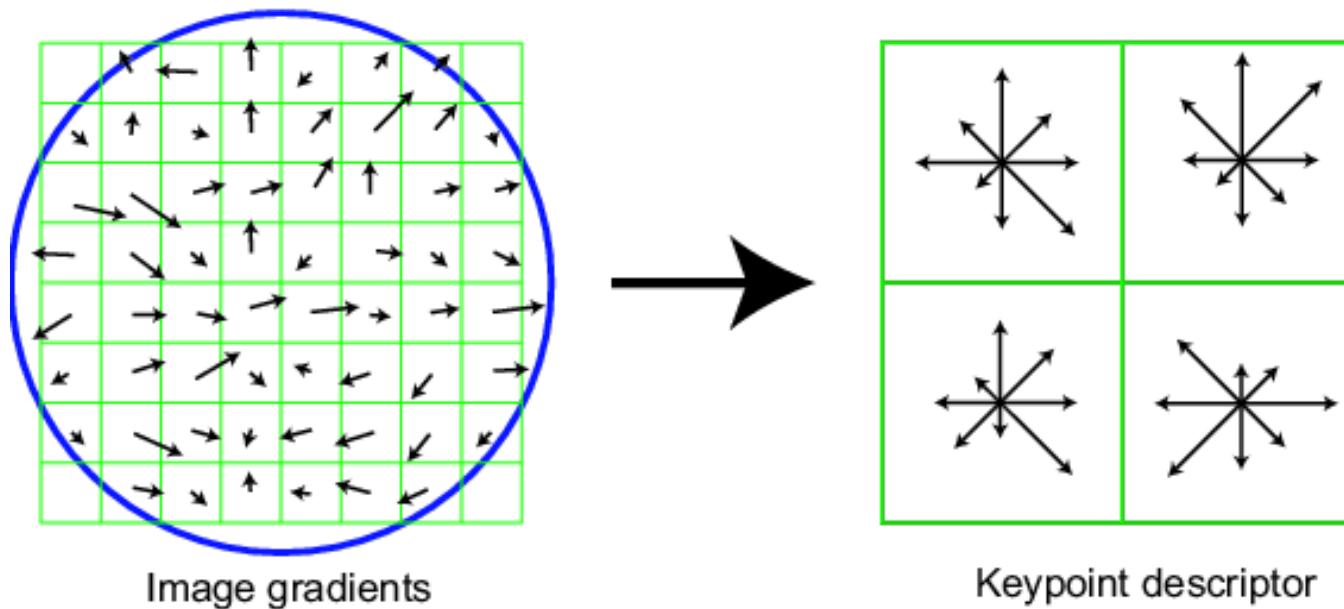
- Alternative to a normalized neighborhood:
derive invariant features within the fixed block
- Gray value histogram:
 - Rotational invariance
 - Invariant to blurring
 - Sensitive to lighting changes (bad)
 - Significant loss of information (very bad)



- Histogram of the gradient direction (**orientation histograms**)
 - Invariant to (additive) lighting changes
 - Building block of many successful descriptors

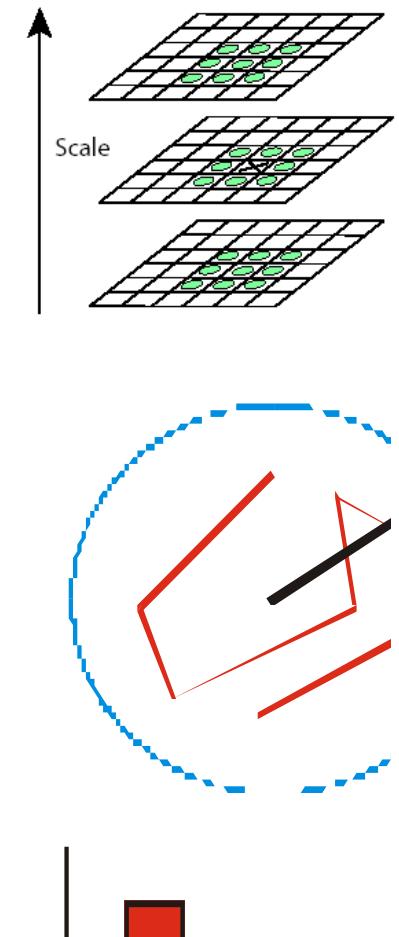


- Very popular local descriptor (several variants exist)
- Based on local assembly of orientation histograms and adaptive local neighborhoods



Author: David Lowe

- Extract SIFT feature points
 - Strongest responses of Laplacian in scale space
→ position and scale
 - Fit quadratic function to obtain subpixel accuracy
 - Create orientation histogram at selected scale
 - Peak of smoothed histogram estimates orientation
 - In case of two peaks, create two feature points
- Estimation of position, scale, and orientation
- Affine invariance can be provided with MSER
 - In object recognition: dense sampling of such points at all positions and all scales, no rotation invariance



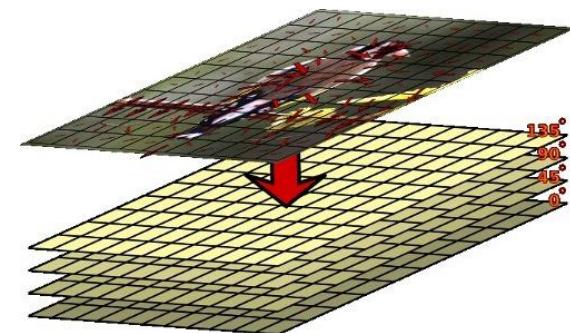
Author: David Lowe

Dense computation of SIFT/HOG descriptors

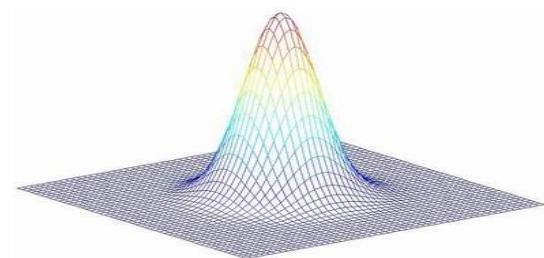
1. Compute gradient orientation and magnitude at each pixel



2. Compute orientation indicator at each pixel
 - Create $N \times M \times 8$ array and initialize with zero
 - Quantize the orientation at each pixel (here 8 bins) and add the respective magnitude to the respective entry in the array



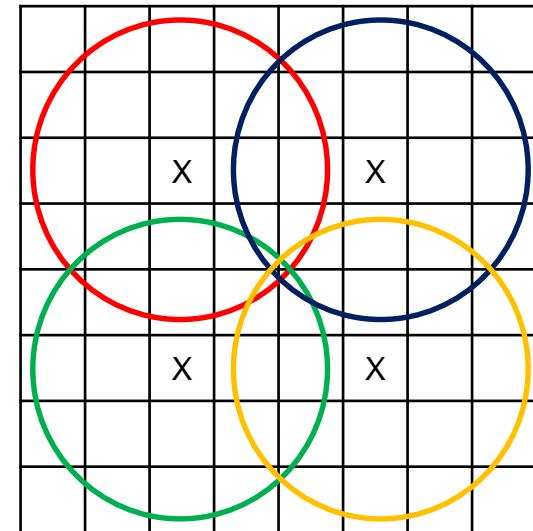
3. Local integration → orientation histogram
- Smooth array with a Gaussian kernel
4. Smooth in orientation direction (among neighboring channels)



Dense computation of SIFT/HOG descriptors

5. Sample feature vectors from the histogram image

- Original SIFT:
4 pixel spacing, 4x4 histogram array
→ 128-D vector
- HOG for person detection (Dalal-Triggs 2005):
6 pixel spacing, 16x8 histogram array,
9 orientations
→ 1152-D vector



Block consisting of
4 cells

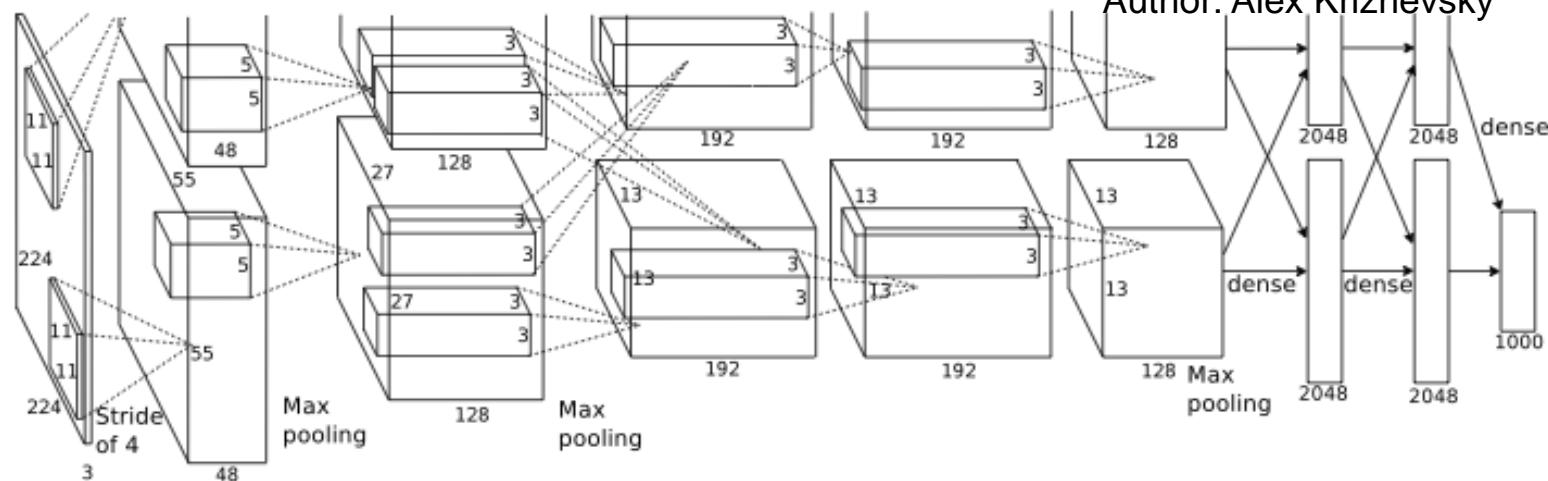
6. Normalize the feature vector

- SIFT: normalize to unit length
- HOG: normalize all **cells** relative to the neighbors of a **block**

Descriptors learned with convolutional networks

Input
image

Author: Alex Krizhevsky



Intermediate representations

Class
output

- CNNs are trained on large datasets with class labels (e.g. ImageNet with 1M images)
→ network learns a representation that is good for object classification
- Intermediate layer outputs turn out to be good generic descriptors

Unsupervised training to trigger invariant features

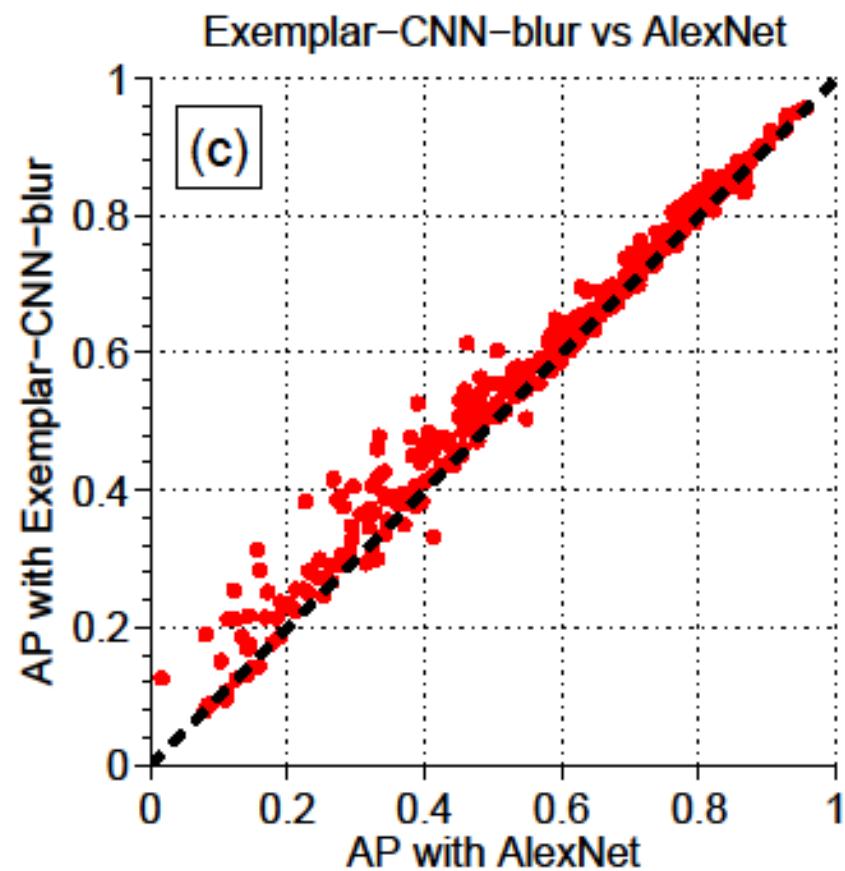
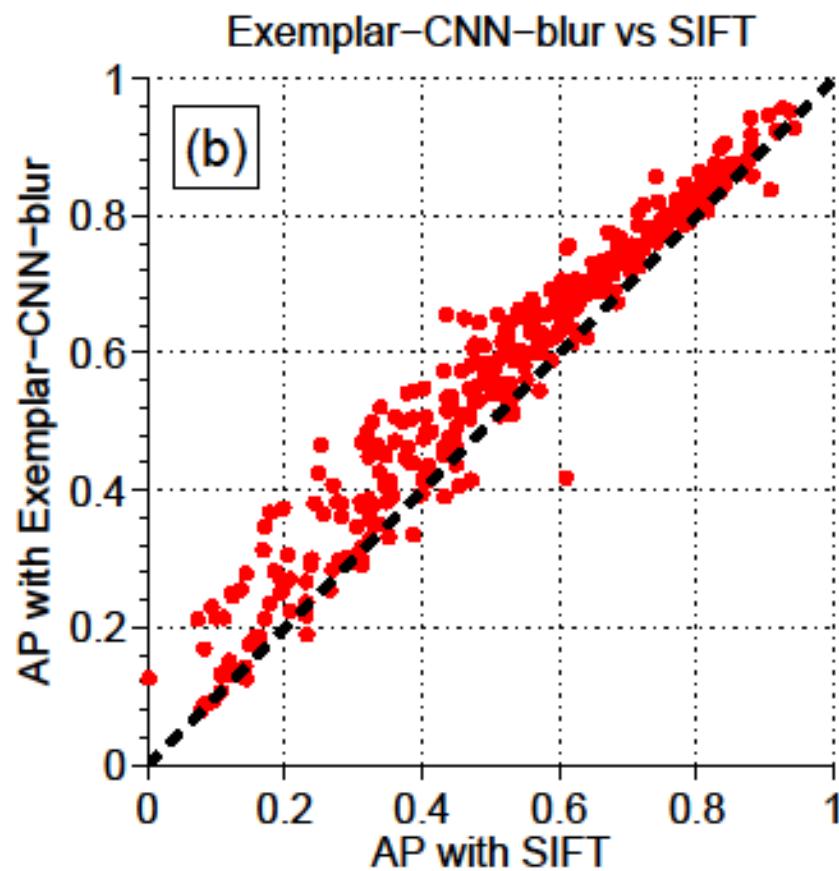
- Train CNN to discriminate **surrogate classes** (Dosovitskiy et al. 2015)



Seed patch and transformed versions of it make up a surrogate class

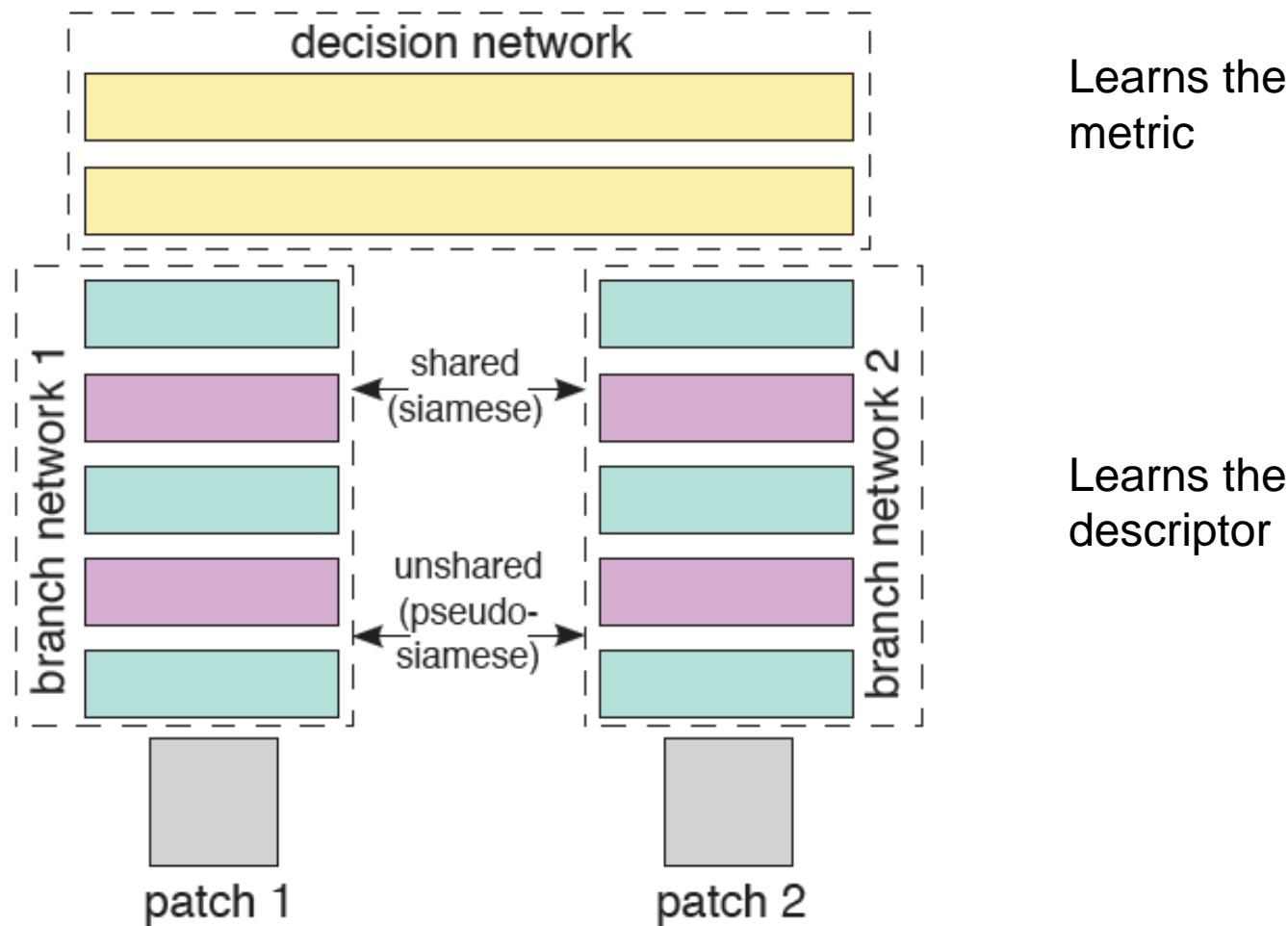
- Applied transformations:
translation, rotation, scaling, color, contrast, brightness, blur
- Transformations define invariance properties of the features to be learned by the network

Descriptor matching performance



Siamese networks

- Trained directly on matching and non-matching patches



Example from Zagoruyko&Komodakis 2015

- Interest points are distinctive points in an image with a significant information content in their neighborhood
- Interest point detection can help establish invariance to certain image transformations.
- Local descriptors describe a local area in the image for the purpose of matching.
- The SIFT descriptor is based on a grid of orientation
- Intermediate layers of ConvNets yield good descriptors

References

- T. Lindeberg: Feature detection with automatic scale selection, *International Journal of Computer Vision*, 30(2): 79-116, 1998.
- D. G. Lowe: Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* 60(2):91-110, 2004.
- J. Matas, O. Chum, M. Urban, T. Pajdla: Robust wide baseline stereo from maximally stable extremal regions, *Proc. British Machine Vision Conference*, 2002.
- N. Dalal, B. Triggs: Histograms of oriented gradients for human detection, *CVPR*, 2005.
- S. Belongie, J. Malik, J. Puzicha: Shape matching and recognition using shape contexts, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509-522, 2002.
- A. Dosovitskiy, P. Fischer, T. Springenberg, M. Riedmiller, T. Brox: Discriminative unsupervised feature learning with convolutional neural networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.
- S. Zagoruyko, N. Komodakis: Learning to Compare Image Patches via Convolutional Neural Networks, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Programming assignment

- Implement the corner detector based on the second eigenvalue of the structure tensor
 - For computing derivatives and for smoothing images you can make use of the predefined filter masks as well as the convolution operations in `CFilter.h`.
 - The structure tensor of a color image is the sum of tensors over all channels
 - See an online math lecture if you do not remember how to compute the eigenvalues of a matrix <http://www.khanacademy.org/>
- Apply the corner detector to the images in `ImageProcessing08Ex03.zip` and play with the parameters
- Implement the dense SIFT/HOG descriptor (without the detector). Use a 4 pixel spacing and a 3x3 grid of histograms. You can ignore scale and rotation invariance and even skip normalization for this exercise.
- Run your corner detector on `tennis500.ppm` and manually select among the interest points the 10 visually most interesting ones. Compute SIFT descriptors for these points.
- Compute SIFT descriptors for all points in `tennis505.ppm`. For each descriptor in `tennis500.ppm` find the best match in `tennis505.ppm` and visualize them in your result image.
- Play with the amount of smoothing, the spacing, and the number of histograms when computing the descriptors.

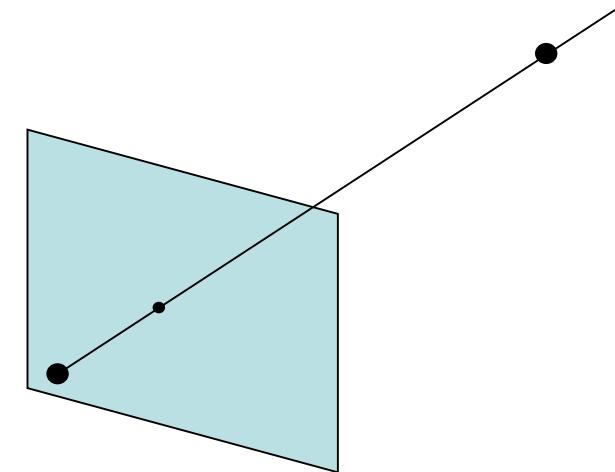
Image Processing and Computer Graphics

Image Processing

Class 9
Shape from X

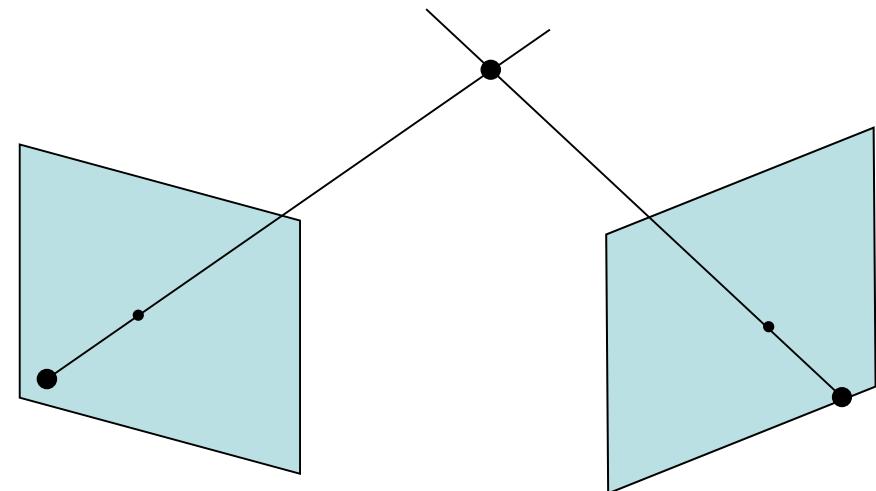
Depth reconstruction

- Another important goal in computer vision is to reconstruct the 3D structure of a scene.
- The missing cue is the depth of an observed point. This depth information has been lost by the projection of the scene to the image plane.
- Any point along the projection ray that passes the camera center and the image point could have caused the observation.
- Thus from the position of a single image point we cannot determine the coordinates of the corresponding 3D point.
- We need additional information either from further images or from a-priori knowledge about the scene.

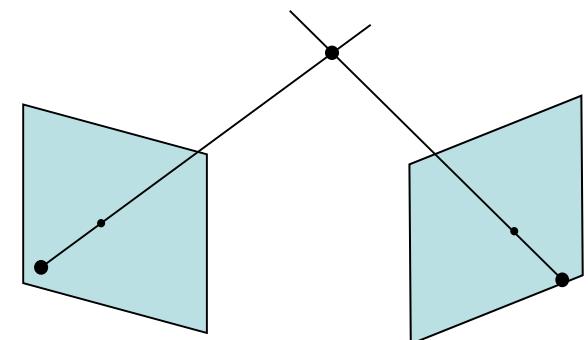


Shape from X

- There are many ways to reconstruct the depth of a surface point. Often these techniques are called **shape from X**, where X stands for the cue that helps reconstruct the depth.
- Some of these cues are rich of information others not so much.
- Most prominent is the reconstruction of depth from binocular images.
- Knowing the corresponding image point in the second image and the camera center of the second camera yields a unique 3D point.
- The camera center of the second camera must not coincide with the one of the first camera.



- For reconstructing the 3D point, we need the two corresponding image points. Finding this pair is the key challenge in stereo reconstruction
→ **disparity estimation**
- With the corresponding points known, we must reconstruct the two projection rays. The cross section tells us the sought 3D point.
- For reconstructing the projection rays, we need to know the projection properties of the cameras.
- These properties are comprised in the projection matrix **P**
- Finding **P** is called **camera calibration**.



- We can add further cameras to reconstruct surface points in the scene.
- Geometrically, only two cameras are necessary to reconstruct the 3D point.
- In practice, we have several challenges:
 - We must **find corresponding points**. Correspondences must be expected to be noisy and some can be even completely wrong.
 - For **occluded points** we cannot establish a correspondence.
 - The **accuracy** of the correspondences is limited by the pixel grid.
- With more cameras, part of the noise in the correspondences is averaged out and the total accuracy of the depth estimates is increased.
- Drawback: multiple cameras are more difficult to calibrate.

Multi-view reconstruction example



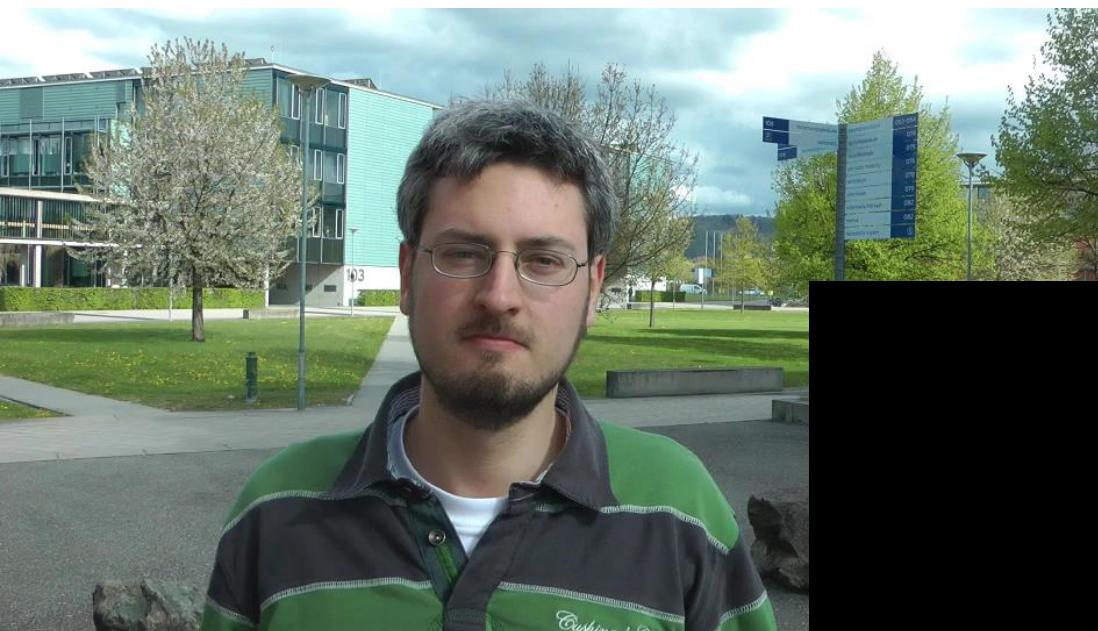
Multiple views



Reconstructed surface

- Instead of multiple static cameras that simultaneously observe a 3D point, we can also consider a single moving camera that observes a static 3D point.
- Advantages:
 - We obtain plenty of images from a single camera.
 - With a reasonable frame rate the displacements in successive views are small
→ easier correspondence search and smaller occlusion areas
- Problems:
 - We cannot use a fixed, calibrated camera setup. We must estimate the camera motion (**egomotion**), together with the structure.
 - The scene must be static. Otherwise we get ambiguities between the depth, the camera motion, and the motion in the scene.
- Sometimes the camera motion is approximately known (due to some other sensors, e.g, in robotics). This can simplify the task.

Structure from motion example



Decomposition of the projection matrix

- For structure from motion, we must factorize the projection matrix:

$$\mathbf{P} = \mathbf{K}\mathbf{M}$$

- The first part is the **camera calibration matrix**

$$\mathbf{K} = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

which contains the camera's **internal parameters**.

- The second part is the pose of the camera relative to a world coordinate system

$$\mathbf{M} = (\mathbf{R}|\mathbf{t})$$

consisting of a rotation matrix and a translation vector, each having 3 degrees of freedom. These are the **external parameters**.

- First the camera's **internal parameters** are calibrated (details in Computer Vision)
- We then only need to calibrate the **external parameters** online. This is the translation and rotation of the camera. This egomotion can be estimated from point correspondences (details in Computer Vision)
- At this point it is assumed that the internal parameters stay fixed.
- In case of autofocus, this is no longer the case. Also some of the internal parameters can be estimated online (**autocalibration**), but this decreases the robustness of the estimation.
- With the camera being fully calibrated (we know $\mathbf{P} = \mathbf{K}\mathbf{M}$), we can estimate the 3D structure of the scene from point correspondences.

Structure from motion: loop closing

- In robotics, a related problem is known as **self-localization and mapping (SLAM)** → leads to the same type of optimization problem
- Localization errors accumulate over time.
- When the camera returns to an earlier position and if localization errors are small enough, the image can be matched to the earlier image. This is called **loop closing**.
- This matching is a special case of object recognition (instance recognition)
- Loop closing is the only way to correct the accumulated errors (**drift**).

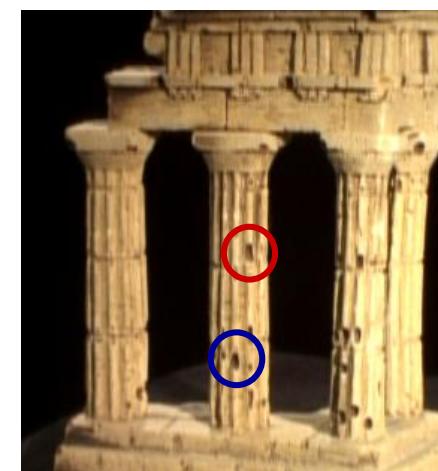
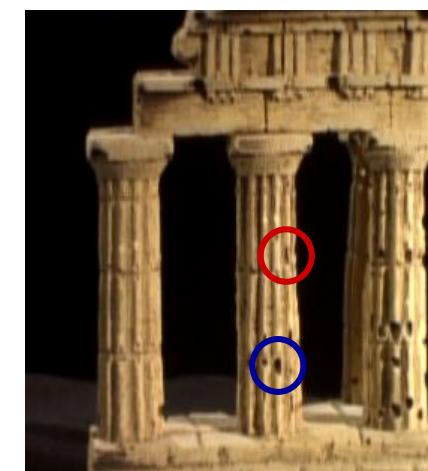
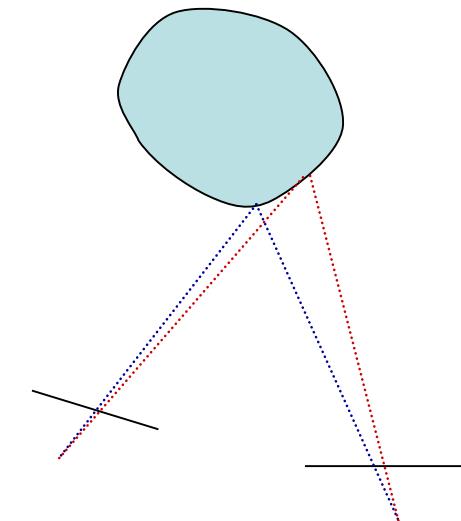


Iterative correction after loop closing

Author: Benjamin Ummenhofer

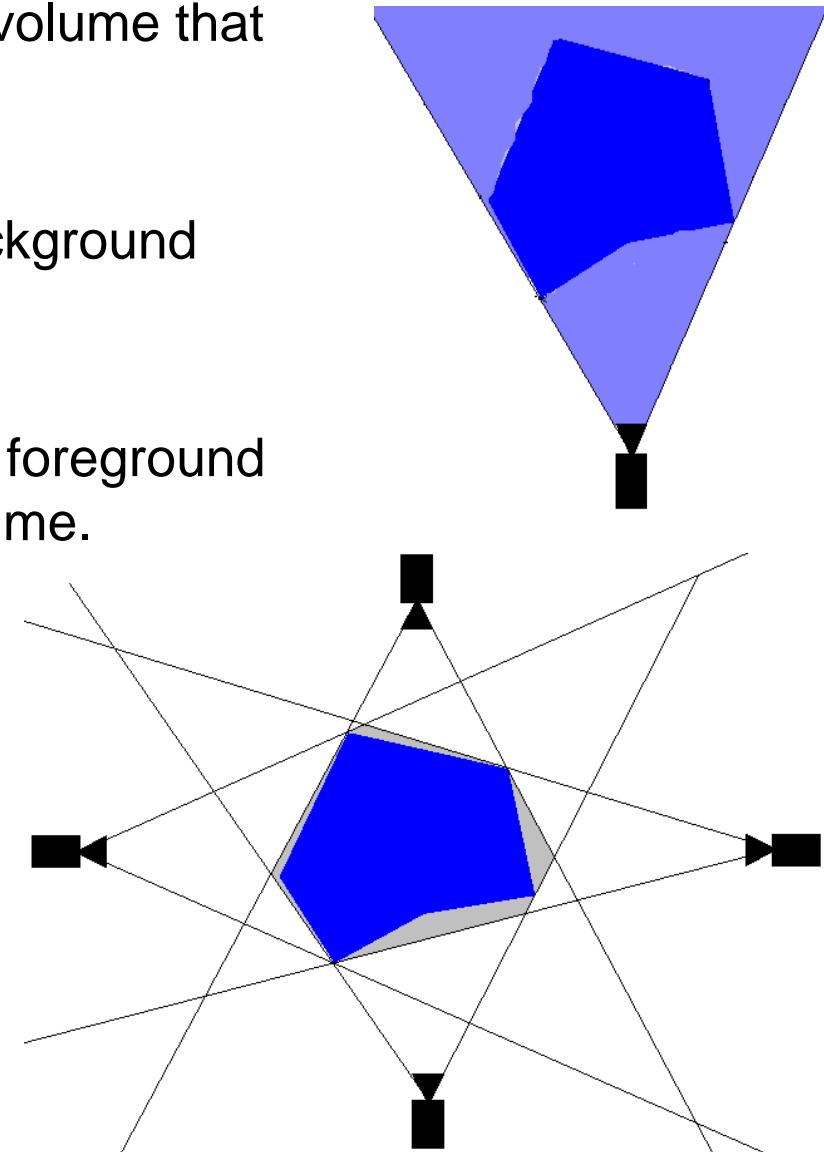
Shape from silhouette

- Most shape reconstruction methods are based on point correspondences between images from different viewing directions.
- Correspondences are preferably established for points whose projection rays hit the surface in nearly orthogonal direction.
- If the projection ray is nearly tangential to the surface in one camera, structures on the surface are severely deformed in the image and do not allow for robust matching anymore.
- **Shape from silhouette** is kind of the contrary approach:
shape is inferred from points where the surface is tangential to the viewing direction.



Shape from silhouette

- The silhouette in the image restricts the volume that can be occupied by the observed object.
- Points along projection rays that see background cannot belong to the object volume.
- Points along the projection rays that see foreground may or may not belong to the object volume.
- With additional cameras, the volume that can be occupied by the object is successively reduced.
- The remaining volume is the maximal volume that is consistent with all silhouettes. It is called the **visual hull**.



Author: Keith Yerex

No intrusions



color-based reconstruction

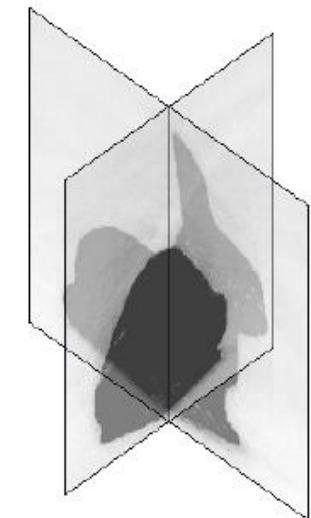
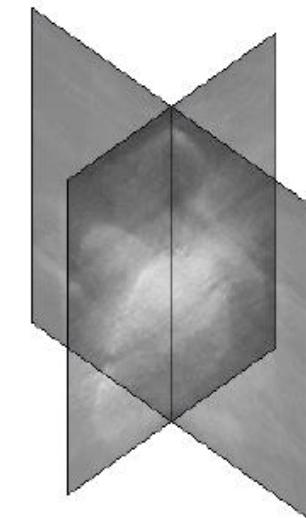
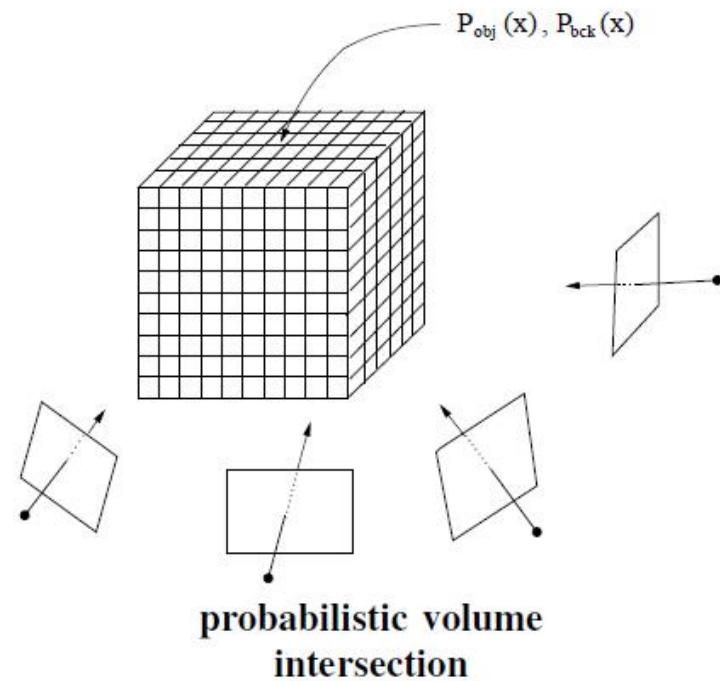


stereo-refined reconstruction

Author: Kalin Kolev

- Intrusions do not show in any of the silhouettes → considered as object
- Shape from silhouette mainly helps find good initializations for other reconstruction methods, e.g., stereo.

Shape from silhouette as 3D segmentation



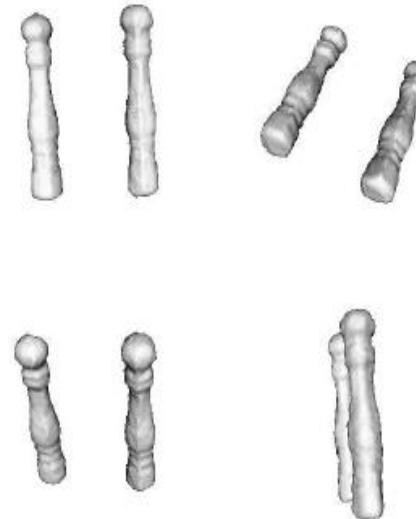
- Finding the silhouette in the image (segmentation) and reconstructing the surface can also be considered as a single 3D segmentation problem.
- The task is to find a surface that consistently separates all images into a foreground and a background region.

Examples

Authors: A. Yezzi, S. Soatto



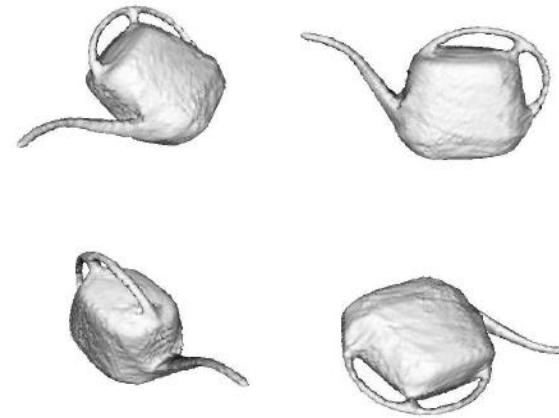
4 of 22 input images



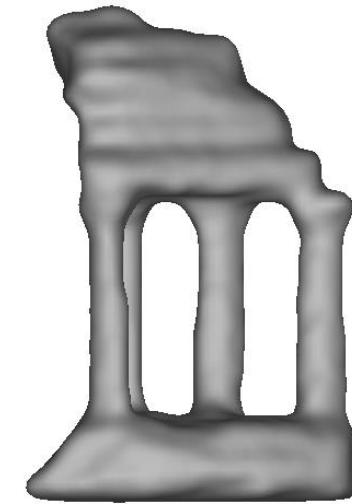
Result



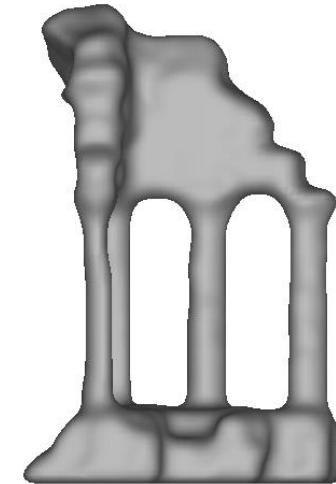
Input image



Result



Shape from silhouette



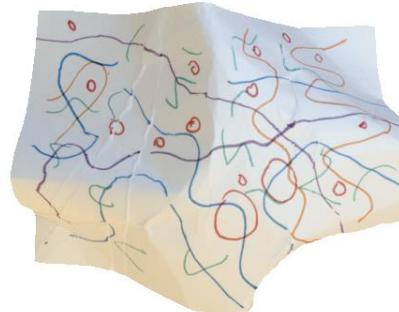
Multi-view stereo

Author: Kalin Kolev

Shape from shading: general case

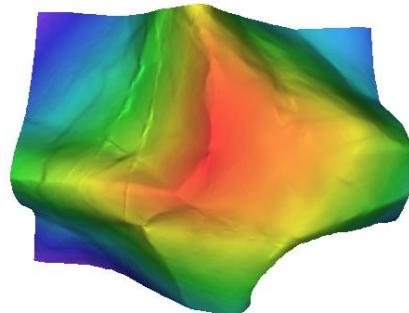
Author: Jon Barron

Given:

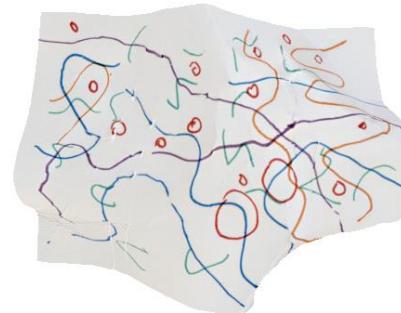


a single image

Sought:



Shape



Albedo



Scattering



Reflectance

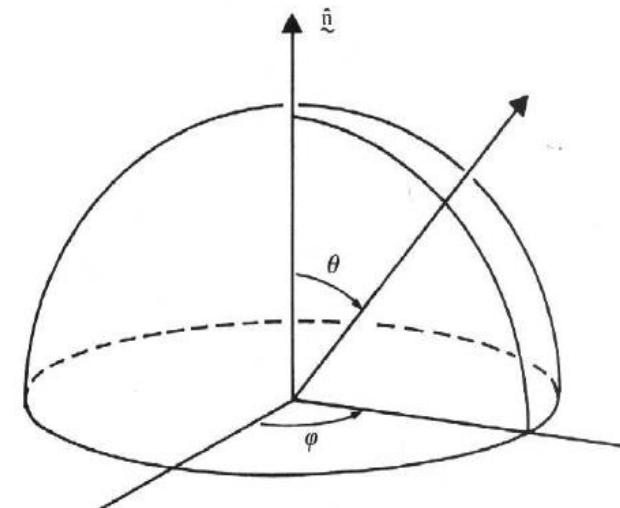
Illumination

Inverse rendering: from the observed image, reconstruct shape, reflectance, and illumination

→ severely under-constrained problem, requires assumptions

Bidirectional reflectance distribution function (BRDF)

- Consider a single light source and a surface element.
- Assume the direction of the light source is given in polar coordinates (θ, ϕ) and the incoming energy is $E(\theta, \phi)$.
- The light emitted by the surface in direction (θ_e, ϕ_e) is given by $L(\theta_e, \phi_e)$.
- The function that describes the reflectance properties of the material is called the **bidirectional reflectance distribution function (BRDF)**:
$$L(\theta_e, \phi_e) = f(\theta, \phi, \theta_e, \phi_e)E(\theta, \phi)$$
- In some more detailed models, the BRDF can have more than four parameters.



Author: B. Horn

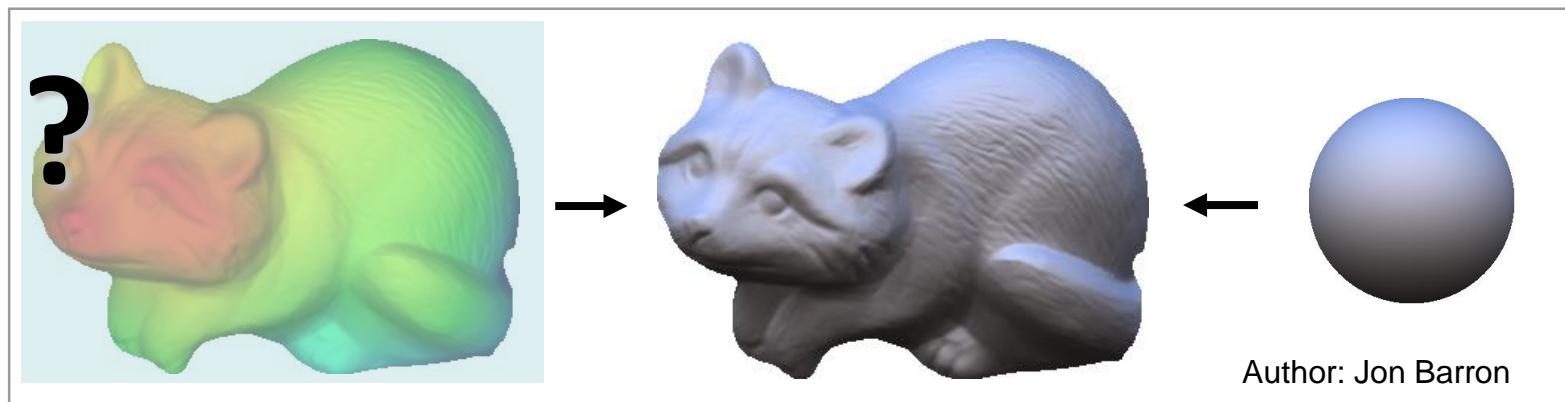
Lambertian reflectance

- A **Lambertian surface** emits the incoming light uniformly in all directions, i.e., the **radiance** only depends on the angle between the surface normal \mathbf{n} and the light source direction \mathbf{s} :

$$I(x, y) = R(X, Y, Z) = \rho \mathbf{s}^\top \mathbf{n}$$

- Corresponds to rough materials (e.g. stone, textiles).
- Most of these materials also absorb some of the incoming light. The non-absorbed fraction of the incoming light is called **albedo** ρ .
- Lambertian reflectance model often used in computer vision, e.g., in stereo matching
- Assuming constant albedo ρ and given light source direction \mathbf{s} leads to a basic shape from shading approach

Basic shape from shading approach (Horn 1970)



Lambertian reflectance; illumination and albedo are known

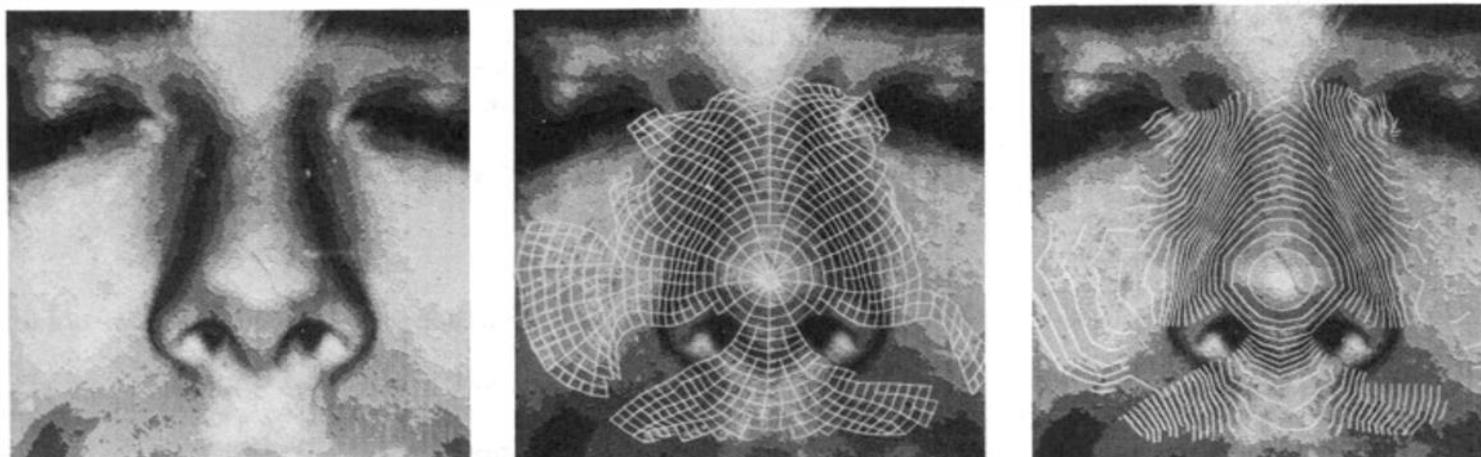


Figure 11-7. The shape-from-shading method is applied here to the recovery of the shape of a nose. The first picture shows the (crudely quantized) gray-level image available to the program. The second picture shows the base characteristics superimposed, while the third shows a contour map computed from the elevations found along the characteristic curves.

B. K. P. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical report, MIT, 1970.

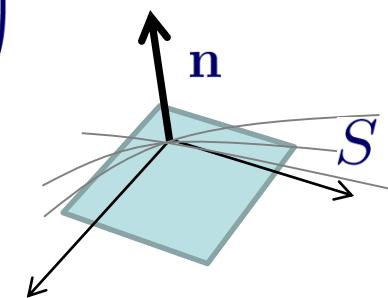
Basic shape from shading approach (Horn 1970)

- The sought depth map $Z(x, y)$ can be related to the surface normal direction $\mathbf{n} \rightarrow$ estimate \mathbf{n}
- The derivatives of the surface $S(x, y) = (x, y, Z(x, y))$ with respect to x and y yield two vector fields that span the tangential plane:
$$\partial_x S = (1, 0, p)^\top, \quad \partial_y S = (0, 1, q)^\top$$
- The cross product of the two vector fields yields the normal field:

$$\begin{pmatrix} 1 \\ 0 \\ p \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ q \end{pmatrix} = \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix} \quad \mathbf{n} = \frac{1}{\sqrt{p^2 + q^2 + 1}} \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix}$$

- Plug into Lambertian radiance function:

$$I(x, y) = \frac{\rho}{\sqrt{p^2 + q^2 + 1}} \mathbf{s}^\top \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix}$$



→ like in optical flow: one equation, two unknowns

- Assuming a smooth surface, leads to a variational approach (Ikeuchi-Horn 1981):

$$E(p, q) = \int (I - R(p, q))^2 + \alpha(|\nabla p|^2 + |\nabla q|^2) dx dy$$

↑
Tuning parameter

with

$$R(p, q) = \frac{\rho}{\sqrt{p^2 + q^2 + 1}} \mathbf{s}^\top \begin{pmatrix} -p \\ -q \\ 1 \end{pmatrix}$$

Lambertian reflectance equation

- After deriving the Euler-Lagrange equations, a minimizer for p and q can be found by gradient descent.
- This energy is non-convex and can have multiple local minima. Coarse-to-fine methods can be used as a heuristic to find the global minimum.

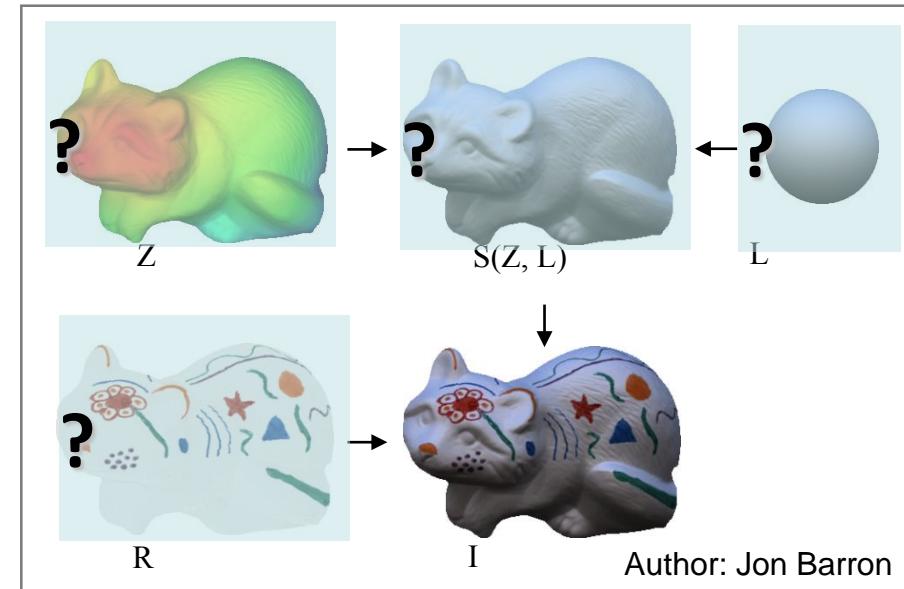
SIRFS: Shape, Illumination and Reflectance From Shading

- Recent framework by Barron and Malik considers more components of inverse rendering

$$\begin{array}{ll} \text{minimize}_{Z,R,L} & g(R) + f(Z) + h(L) \\ \text{subject to} & I = R + S(Z, L) \end{array}$$

with appropriate cost functions

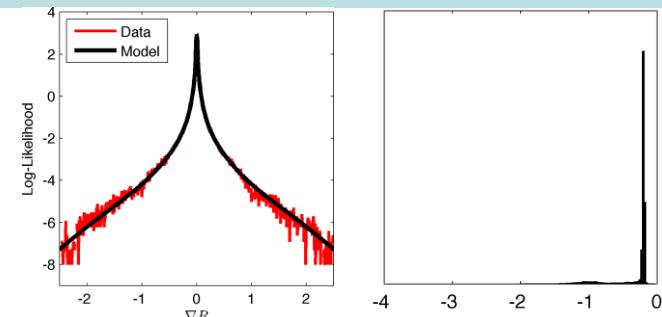
g, f, h



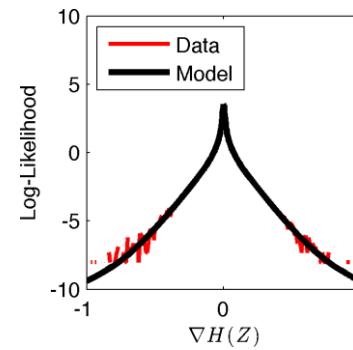
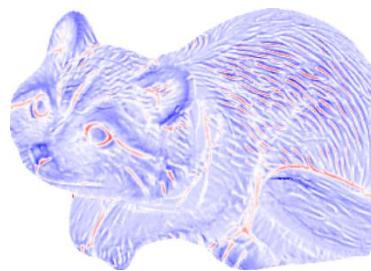
- Includes multiple sources of prior knowledge

Priors learned from training data (four among many)

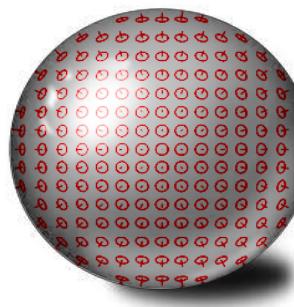
1. Reflectance is mostly smooth and the histogram is sparse



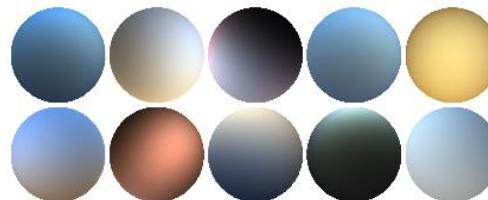
2. Shape is mostly smooth



3. Isotropy of shapes and the fact that an observed surface is more likely to point towards the observer

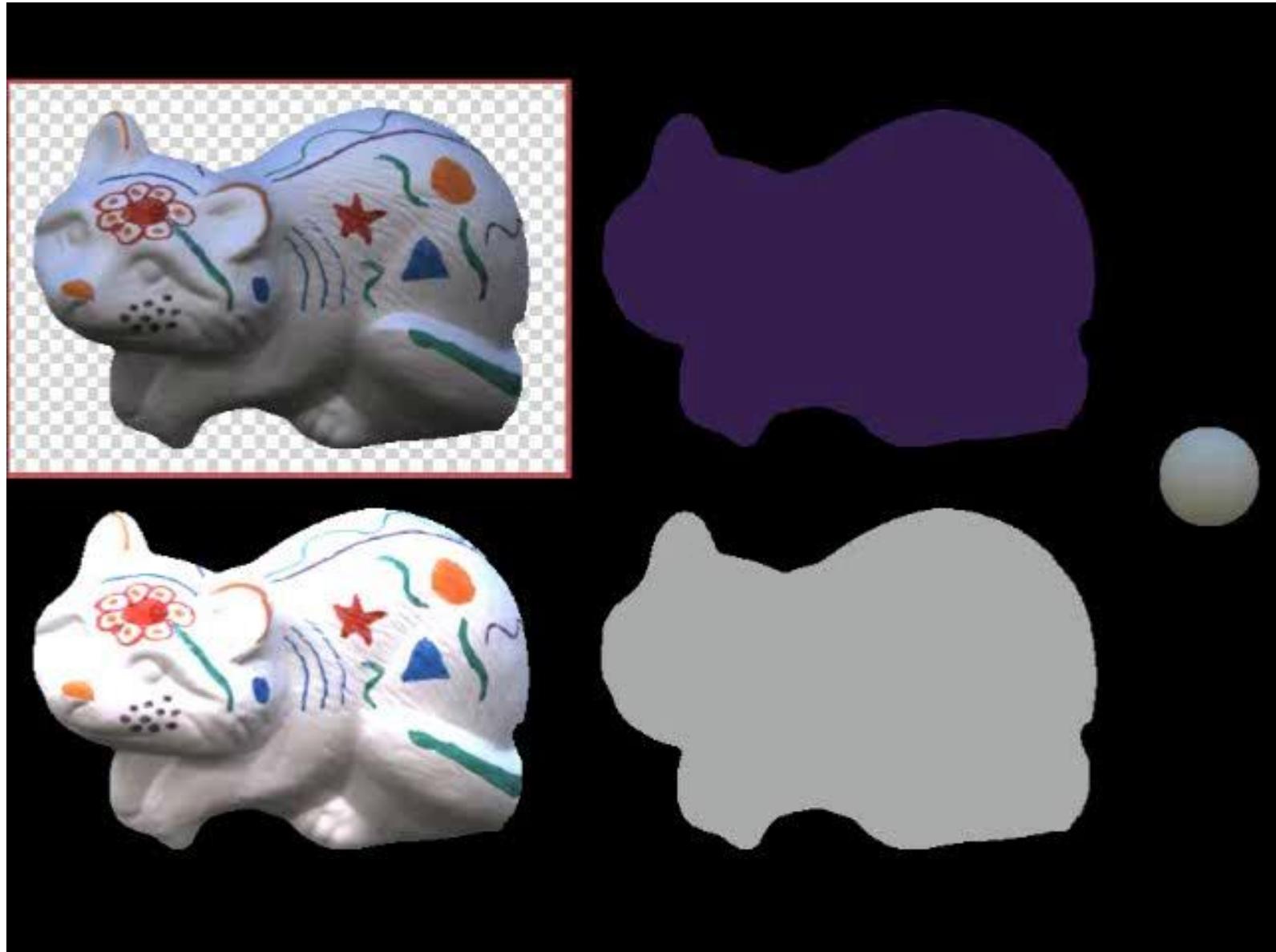


4. Natural lighting prior



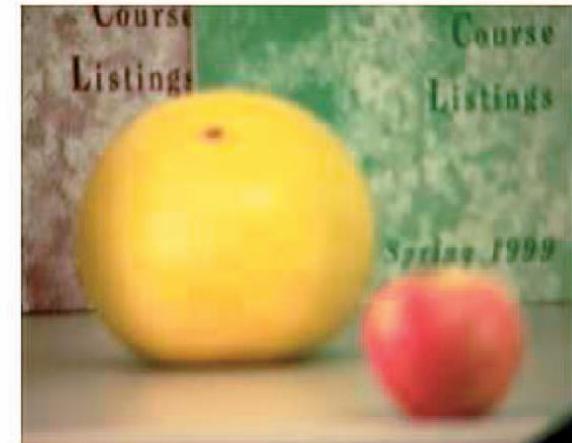
Author: Jon Barron

Optimization with coarse-to-fine Quasi-Newton method



Author: Jon Barron

Shape from defocus



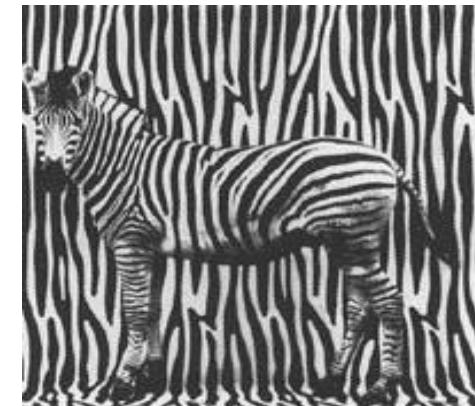
- Given a series of images, where the focus was set to different depths, one can reconstruct the local depth in the image (Favaro-Soatto 2005).
- Estimate the relative blur $d\sigma$ needed to match the other image (Subbarao-Surya 1992)
- The amount of blur needed determines the depth

Shape from texture

- If the appearance of a texture is known, e.g. a regular repetitive pattern, the shape of the surface can be reconstructed from the texture's deformation.
- Usually the texture is not known. Then practical assumptions are that the texture is homogenous, isotropic, and stationary.
- From the repetitive nature of texture elements under these conditions we can estimate the non-deformed shape of such an element and the surface that causes its deformation.



Author: Angelina Loh



Source: CV Lab, UC Berkeley



Source: MPI Saarbrücken

- Reconstructing the depth of points that has been lost by the projection to the image plane is one of the central computer vision tasks.
- There are multiple cues for reconstructing 3D shapes:
 - stereo images
 - camera motion
 - multiple silhouette images
 - shading
 - defocus
 - texture
- Most prominent are stereo reconstruction and structure from motion.

References

- O. Faugeras, Q.-T. Luong: *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*, MIT Press, 2004.
- R. Hartley, A. Zisserman: *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2nd edition, 2004.
- A. Yezzi, S. Soatto: Stereoscopic segmentation, *International Journal of Computer Vision*, 53(1):31-43, 2003.
- K. Ikeuchi, B. Horn: Numerical shape from shading and occluding boundaries, *Artificial Intelligence*, 17:141-184, 1981.
- J. Barron, J. Malik: Color constancy, intrinsic images, and shape estimation, *European Conference on Computer Vision*, 2012.
- P. Favaro, S. Soatto: A geometric approach to shape from defocus, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):406-417, 2005.
- A. M. Loh: *The recovery of 3-D structure using visual texture patterns*, PhD thesis, 2006.

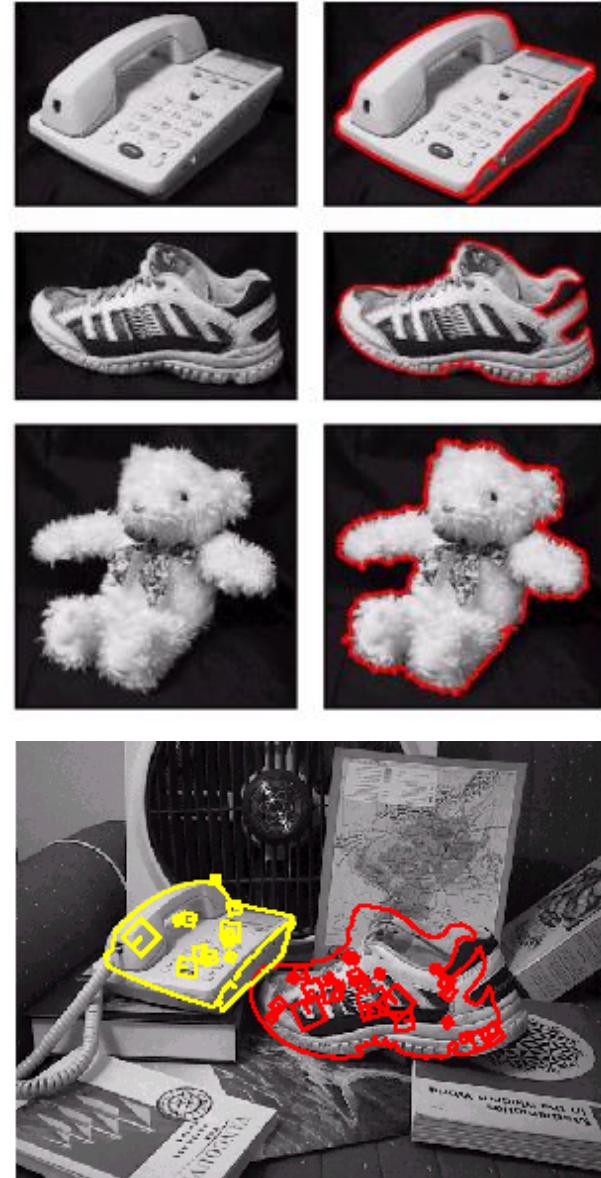
Image Processing and Computer Graphics

Image Processing

Class 10
Object recognition and deep learning

What is object recognition?

- **Instance recognition** seeks to detect the presence of a known object in a new image.
- Often the object should also be localized in the image (detection).
- Major problem: the object can look quite different in the new image due to different viewpoint, lighting, occlusions.
- Important difference to **object class recognition** or object class localization: the same instance of an object class is seen in the two images.
- There can be large differences between two instances of an object class (e.g. two dogs).



Author: David Lowe

A classification problem

- Object recognition consists of two main parts:
 1. A set of **features** that describe the object
 2. A **classifier** that separates objects/object classes
- Classical approach:
Use a handcrafted feature representation and learn the classifier
- Deep learning:
Learn the feature representation and the classifier
- Typical classifiers (see also Statistical Pattern Recognition):
 - **Support vector machine** (two-class)
 - **Logistic regression** (multi-class, used in deep networks)
 - Nearest neighbor classifier (multi-class)

Support vector machine

- Basic idea: learn a decision function with maximum margin (distance to most critical training points).

- Decision function modeled as a linear combination of features:

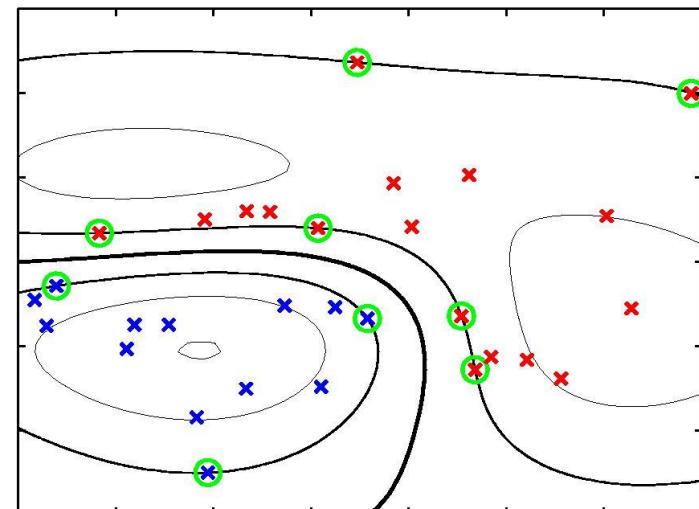
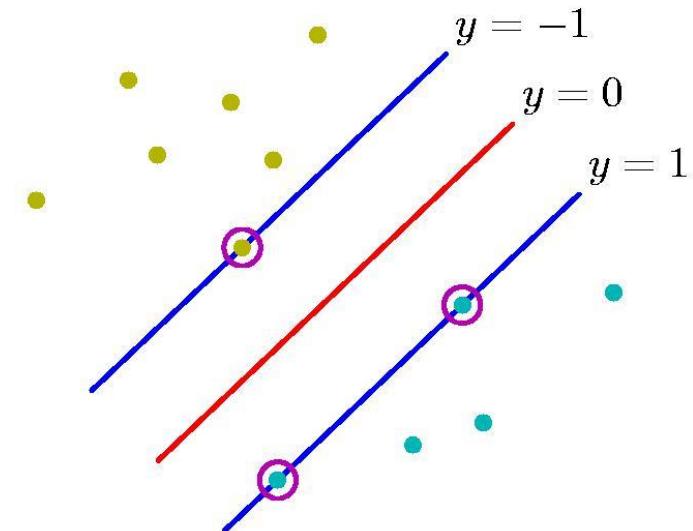
$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

- Large margin concept leads to a convex optimization problem:

$$\operatorname{argmin}_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } t_n(\mathbf{w}^\top \phi(\mathbf{x}_n) + b) \geq 1$$

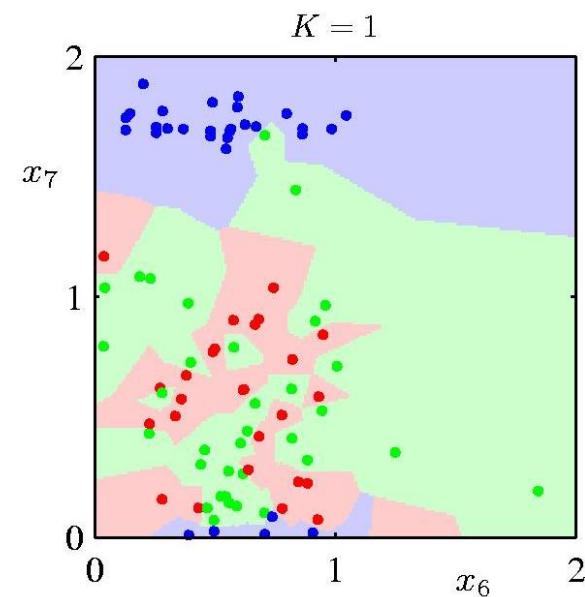
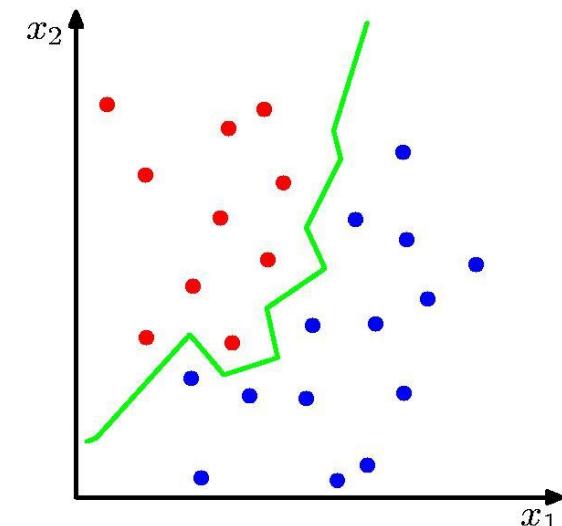
- Efficient code is publicly available (e.g. libSVM, liblinear)



Author: Christopher Bishop

Nearest neighbor classifier

- Simple idea: assign the class label of the most similar training point.
- Advantages:
 - Simple concept
 - Works for multiple classes
- Drawbacks:
 - All training samples must be stored.
 - Search for most similar training sample may consume too much time
 - Does not generalize well



Author: Christopher Bishop

Feature representation

- Color
 - Discriminative capabilities very restricted (tomato = red car)
- Local descriptors (e.g. SIFT)
 - Discriminative properties good (if sufficiently textured)
 - Easy to extract from input images
 - Describe the object **locally**
→ robust to occlusions, local variation
 - Fixed level of abstraction
→ problems with complex variations
- Deep representations
 - Multiple levels of abstraction
 - Learned from training data

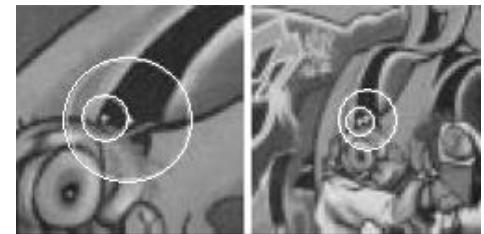


Invariance requirements in object recognition

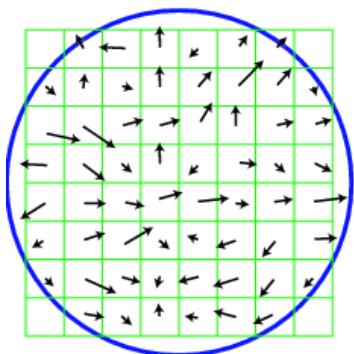
- Aim: the object should be recognized even if its appearance has changed due to some typical transformations.
- Then the descriptors and classifiers are called **invariant** with respect to this transformation.
- There is a tradeoff between invariance and discriminative power of a descriptor
- Instance recognition: invariance needed with respect to
 - Viewpoint (includes translation, scaling, rotation, perspective distortion)
 - Background
 - Lighting
 - Partial occlusion
- Class recognition: needs complex invariant features learned from training examples

Some popular local descriptors

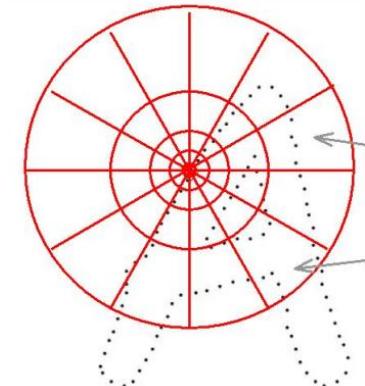
- Distortion corrected patches
 - distortion correction provides affine invariance



- SIFT/HOG (Lowe 2004, Dalal-Triggs 2005)
 - based on gradient orientation histograms
 - can be made invariant to scaling, rotation, and brightness/contrast change

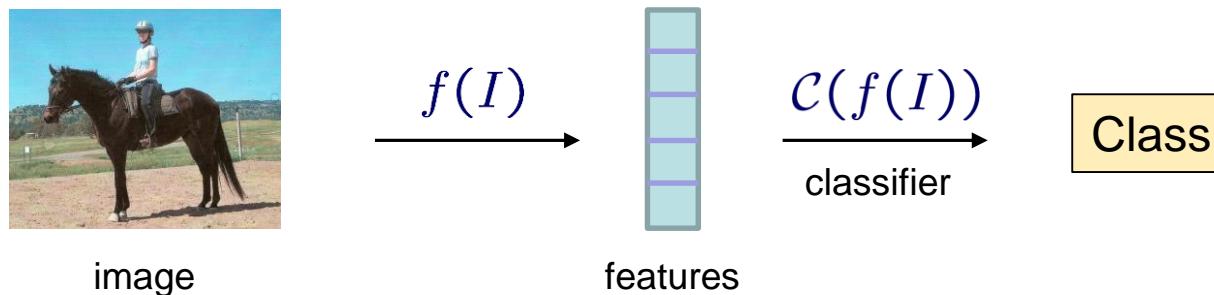


- Shape context (Belongie-Malik 2002)
 - based on object contours or edges
 - histogram of relative position of other contour points in the local neighborhood



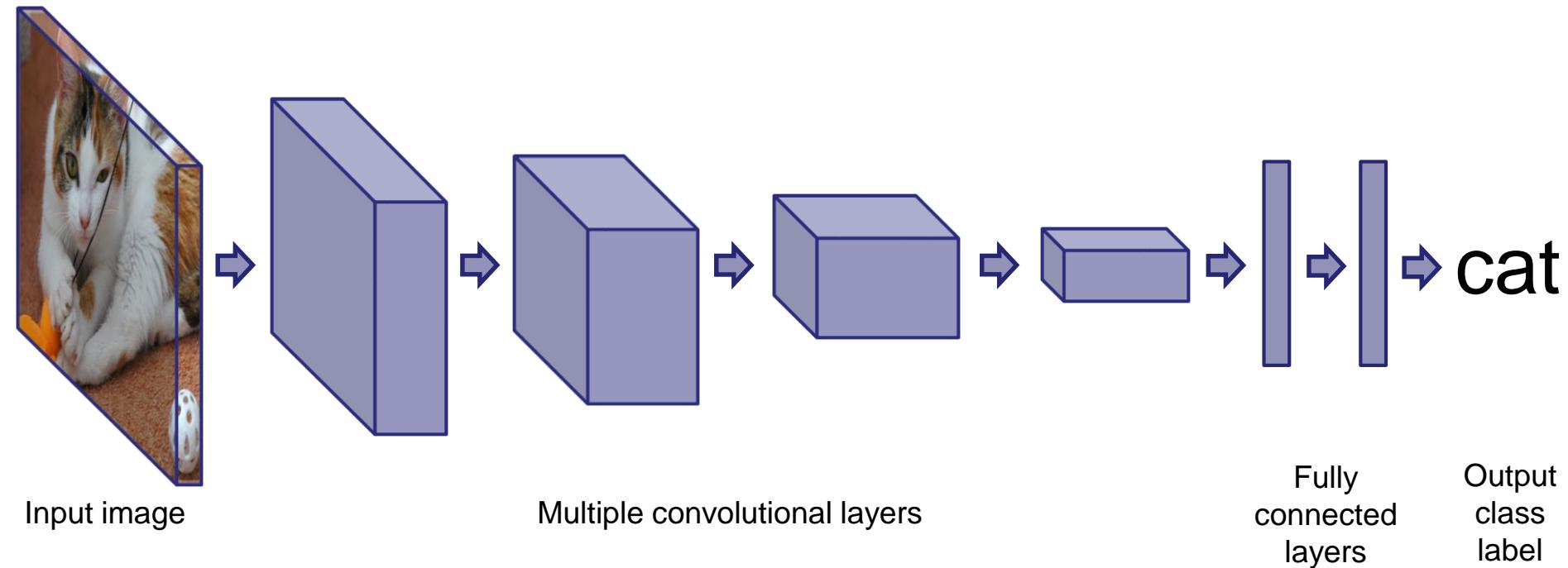
Feature learning

- Instead of manual descriptor design, let the computer find the optimal descriptor for a task defined by a training set
- Task: object classification
→ training set consists of images and their class labels



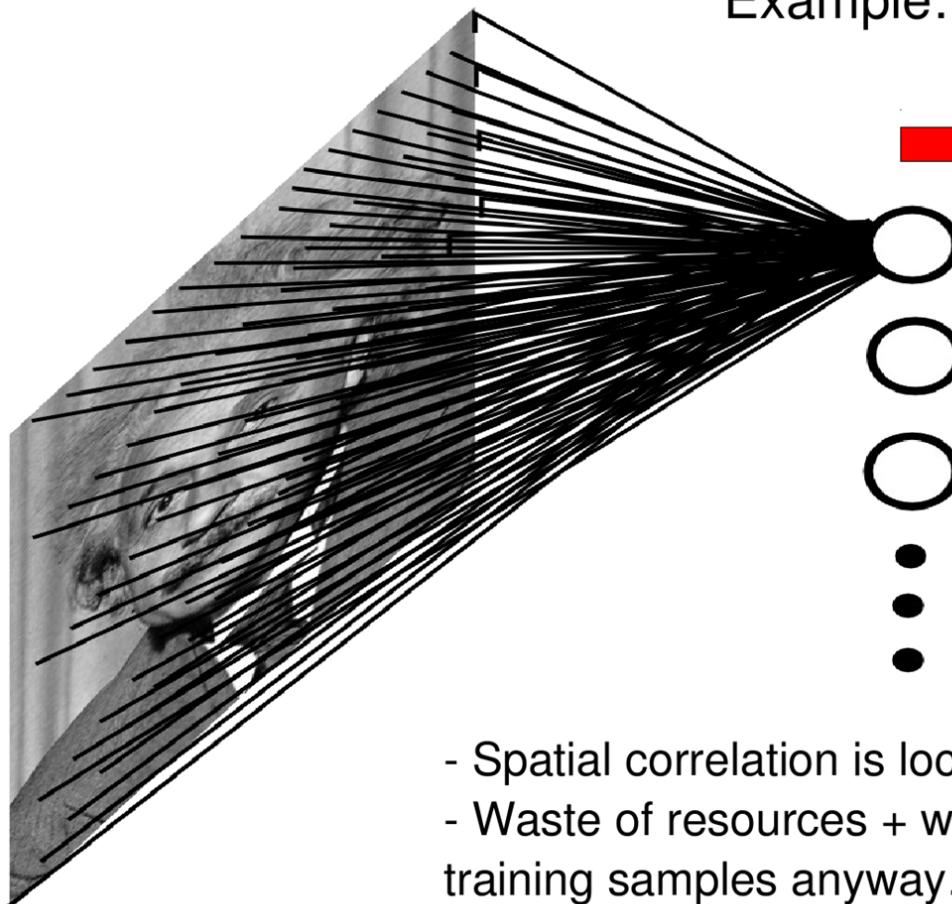
- Shallow modeling of the function $f(I)$ is not efficient to cover all the variation that appears in an object class
→ hierarchy of functions, “deep” representation

Typical deep network for image classification



- Each layer produces a more abstract representation of the input
- Layers are similar in structure
- Weights of connections between layers (filters) learned from training data

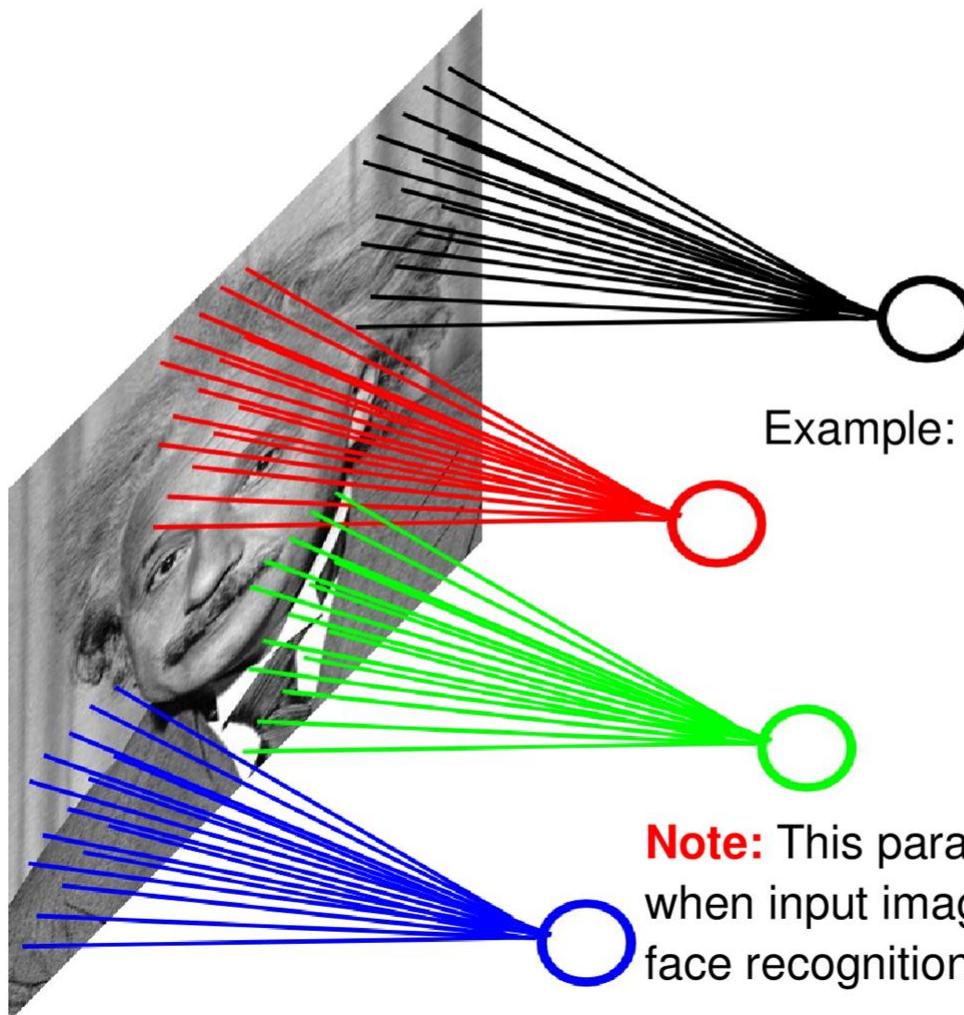
Fully Connected Layer



40

Ranzato

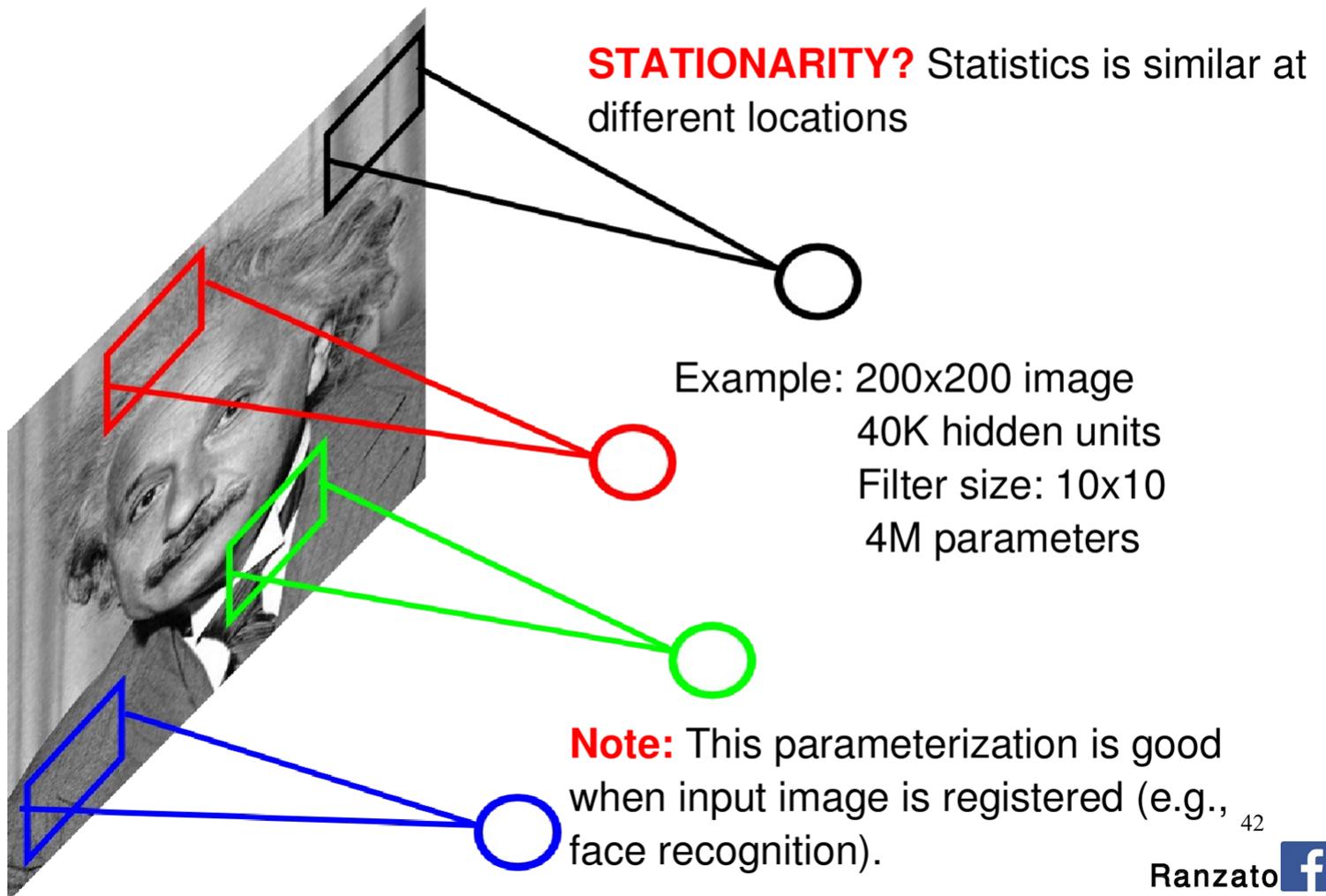
Locally Connected Layer



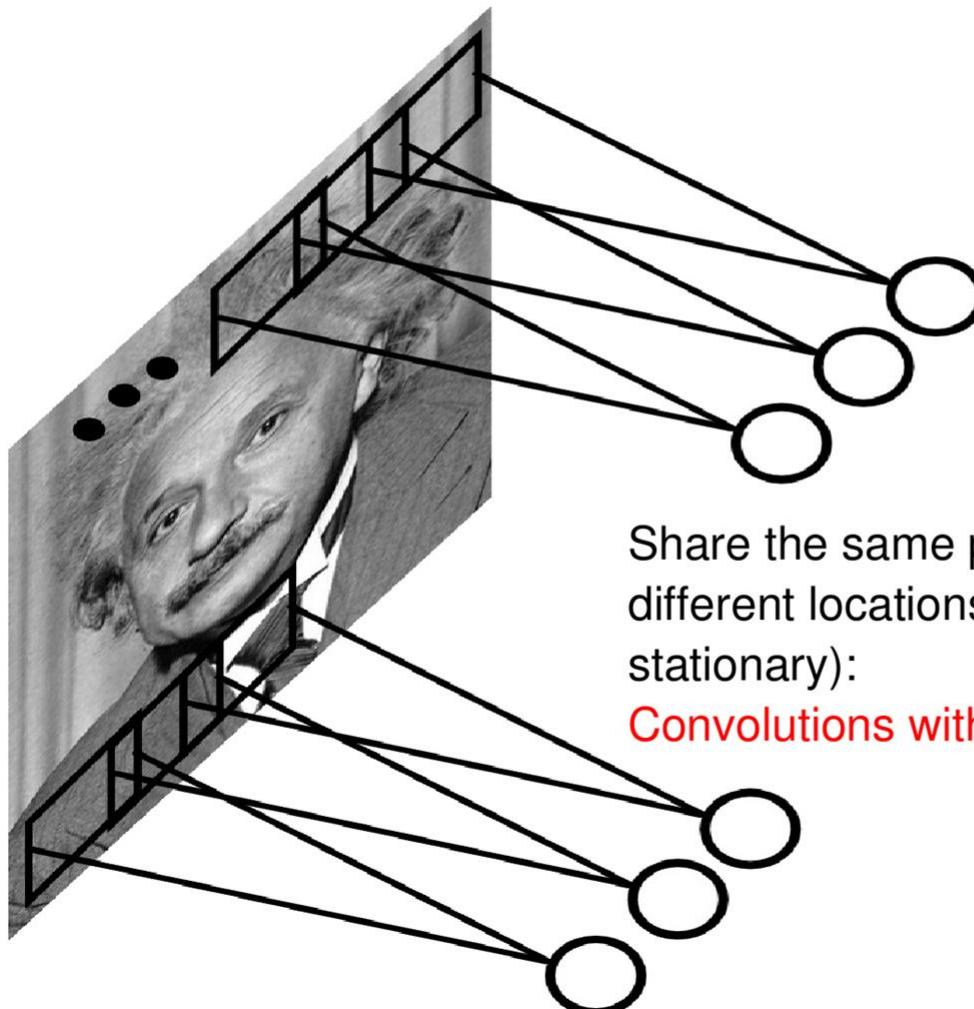
Example:
200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good
when input image is registered (e.g.,
face recognition).

Locally Connected Layer



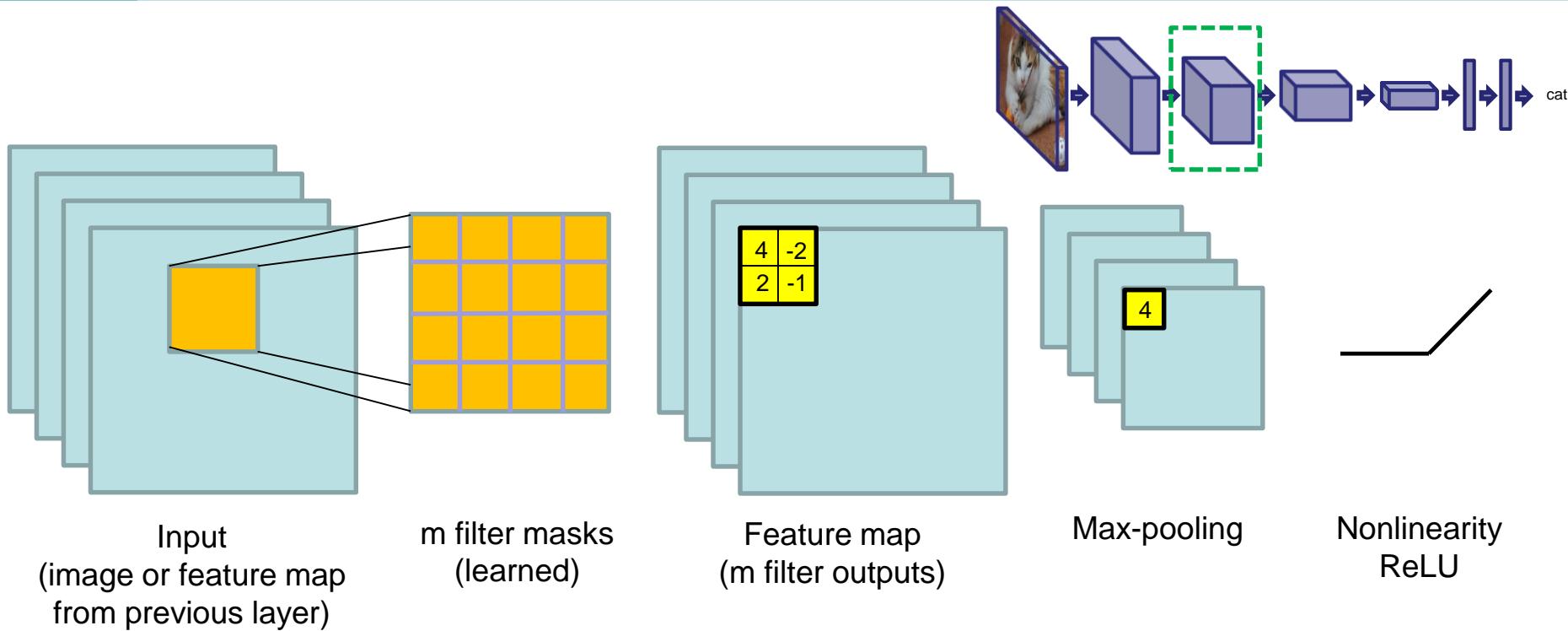
Convolutional Layer



43

Ranzato

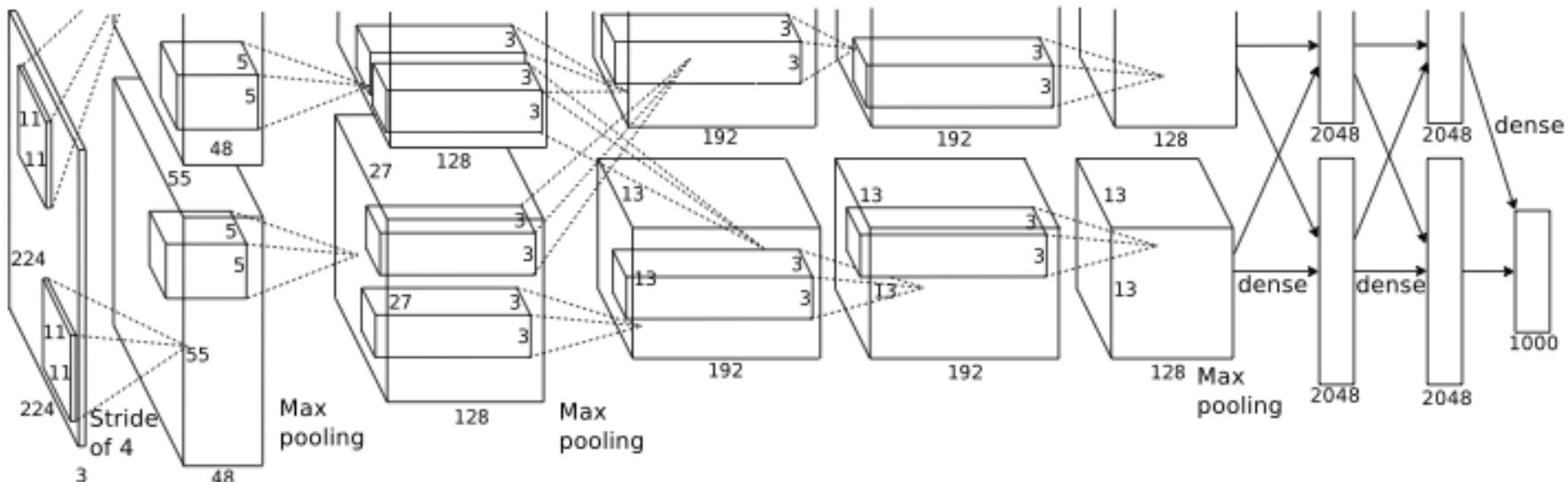
Single layer module of a convolutional network



- **Max-pooling** replaces a small local area of the feature map by the maximum value in that area (= downsampling)
→ data reduction, loss of localization accuracy
→ allows for larger receptive fields in the next layer
- **ReLU nonlinearity** sets negative values to 0 (no activation)

Large ConvNet

Author: Alex Krizhevsky



- Modern networks have 30 layers or more
- Cost function for training the weights: cross entropy on training set (in principle the classification error, but better)

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} - \sum_i \sum_c y_i^c \log(f_{\mathbf{w}}^c(x_i))$$

- Optimization by gradient descent (**back-propagation**)

Back-propagation

Optimization problem:

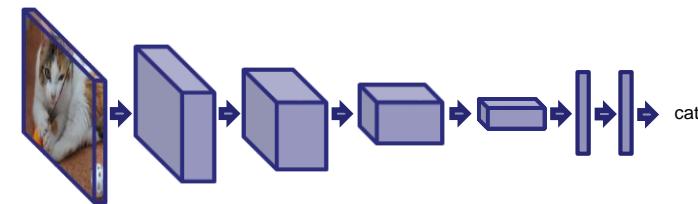
$$L(\mathbf{w}) = - \sum_i y_i^\top \log(f_{\mathbf{w}}(x_i)) \quad \mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} L(\mathbf{w})$$

1. Initialize the weights (starting point for gradient descent)
2. Compute $f_{\mathbf{w}}(x_i)$ by forward-propagating a sample through the network
3. Gradient with regard to a weight in a certain layer: use **chain rule**

$$\frac{dL}{df} = \sum_i (f_{\mathbf{w}}(x_i) - y_i) \quad \text{Error}$$

$$\frac{dL}{dw_{l_{\max}}} = \boxed{\frac{dL}{df}} \frac{df}{dw_{l_{\max}}} \quad \text{Error propagated to the last layer}$$

$$\frac{dL}{dw_{l_{\max}-1}} = \boxed{\frac{dL}{df} \frac{df}{dh_{l_{\max}-1}}} \frac{dh_{l_{\max}-1}}{dw_{l_{\max}-1}} \quad \text{Error propagated to the second last layer}$$



and so on

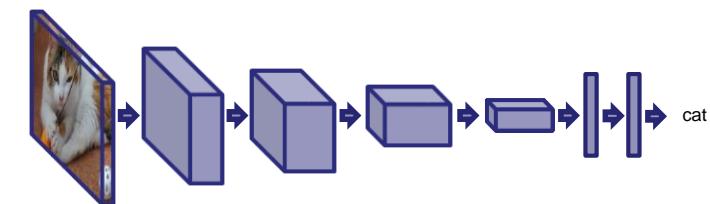
4. Update the weights

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \tau \nabla L$$

Nice properties of CNNs

- Filters of different layers capture different scales due to max-pooling
 - Filters at low layers are well localized and consider only a small image area (**small receptive field**).
 - Filters at high layers are badly localized and capture the context of a large image area (**large receptive field**).
- Feature sharing: the same features obtained at low layers can be useful for assembling various complex features at higher layers

- Successive abstraction

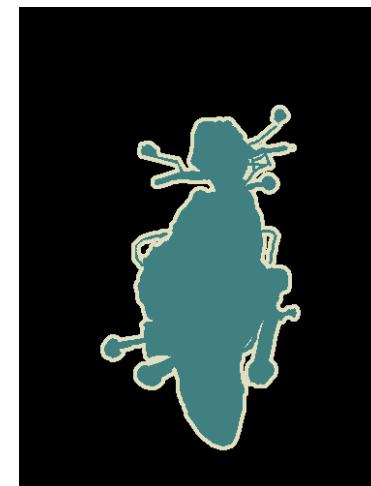


Features close to the image input resemble simple edge filters

Features close to the class output are invariant to the typical appearance variations

Localization tasks

- Image classification only provides a class label per image
- **Object localization:** provide a bounding box of the object
- **Object detection:** bounding boxes for potentially many object instances in the image
- **Semantic segmentation:** say for each pixel to which object class it belongs
- **Instance segmentation:** additionally separates different class instances



Sliding window approach (Viola-Jones 2001)

- Define features in a local window
- Consider many (or all) positions and scales and make a binary decision:
Is this window a car or not?

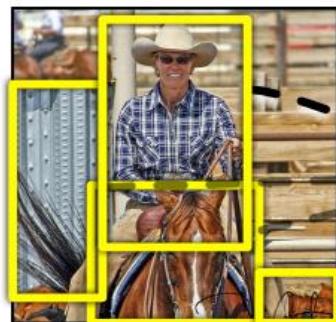


- Non-maximum suppression: keep only the local maxima
- Convolutional networks and the sliding window concept are redundant
→ exploit for larger efficiency (Sermanet et al. 2014)

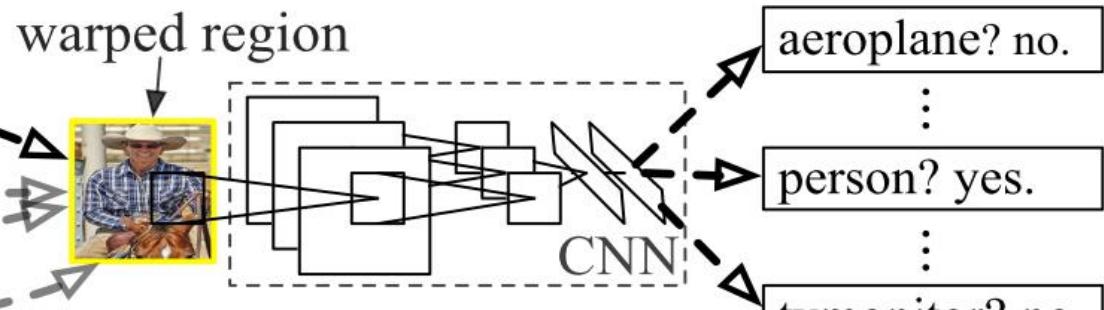
Deep network classification on object window proposals (R-CNN)



1. Input image



2. Extract region proposals (~2k)



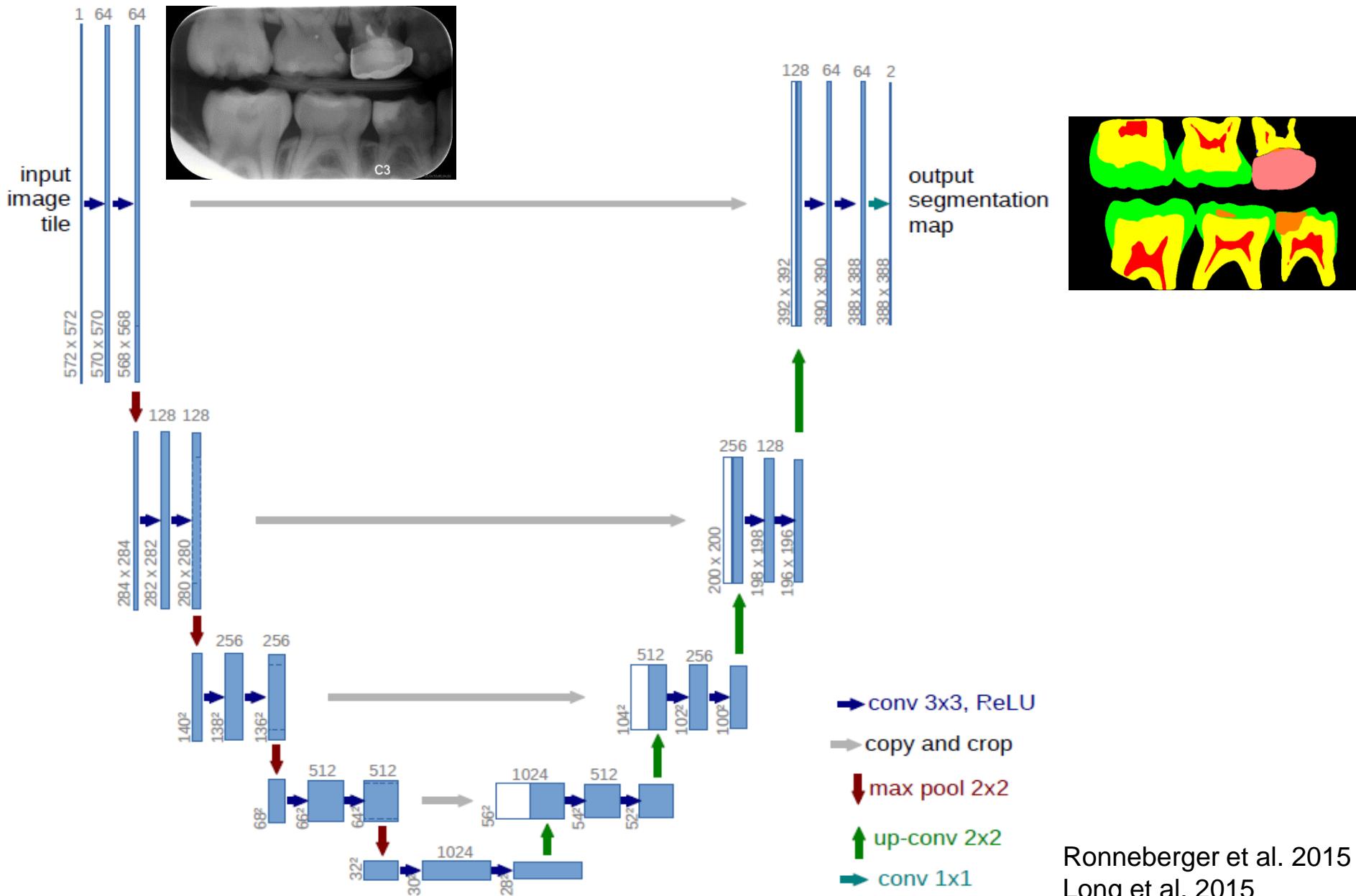
3. Compute CNN features

4. Classify regions

Girshick et al. 2014

- Instead of all windows, only ~1000 region proposals are considered
- Faster implementations first compute the ConvNet activations of the whole image and use them to classify the proposal windows

Semantic segmentation with a deep network



Ronneberger et al. 2015
Long et al. 2015

Object recognition benchmarks

- For comparing the performance of different recognition techniques, a common benchmark is needed.
- There are several public benchmarks:
 - ImageNet (localization with 1000 classes, detection):
<http://www.image-net.org/>
 - PASCAL Visual Object Classes (classification, detection, segmentation, ...)
<http://pascallin.ecs.soton.ac.uk/challenges/VOC/>
 - Microsoft CoCo (detection, image caption generation)
<http://mscoco.org/>
- Benchmarks come with a **training set** and a **test set** (both annotated)
 - Training set: train classifiers and optimize parameters
 - Test set: run the final method and measure performance
- Detection performance is assessed by precision-recall curves

Precision-recall curves

- **Precision:** which part of the detected objects is correct?

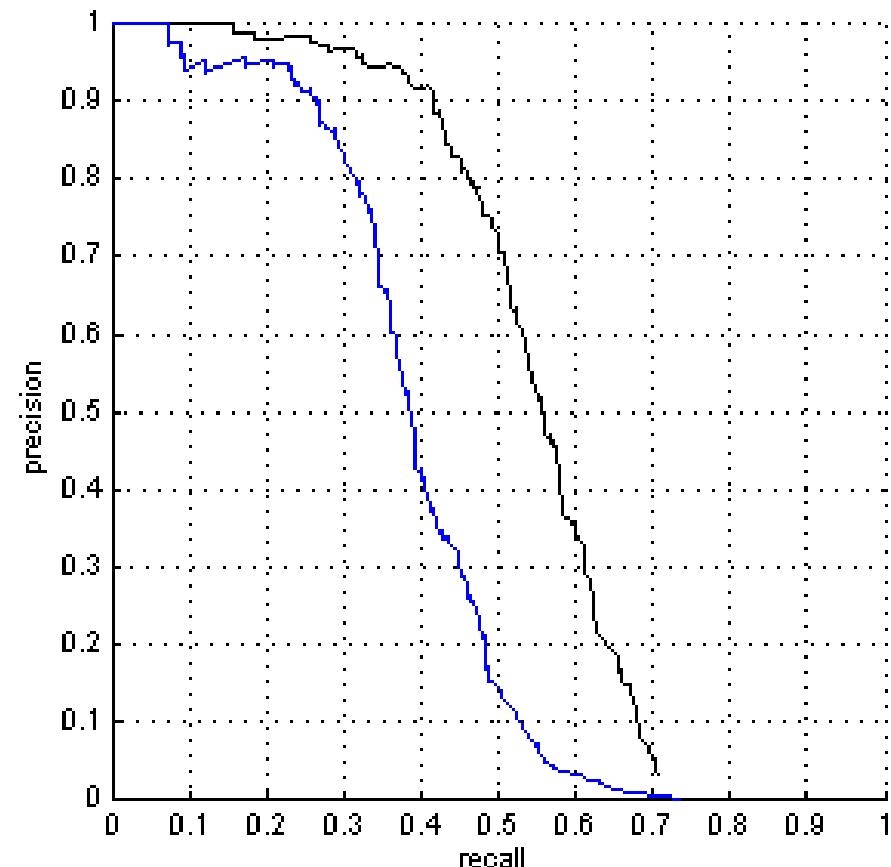
$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **Recall:** which percentage of objects is detected?

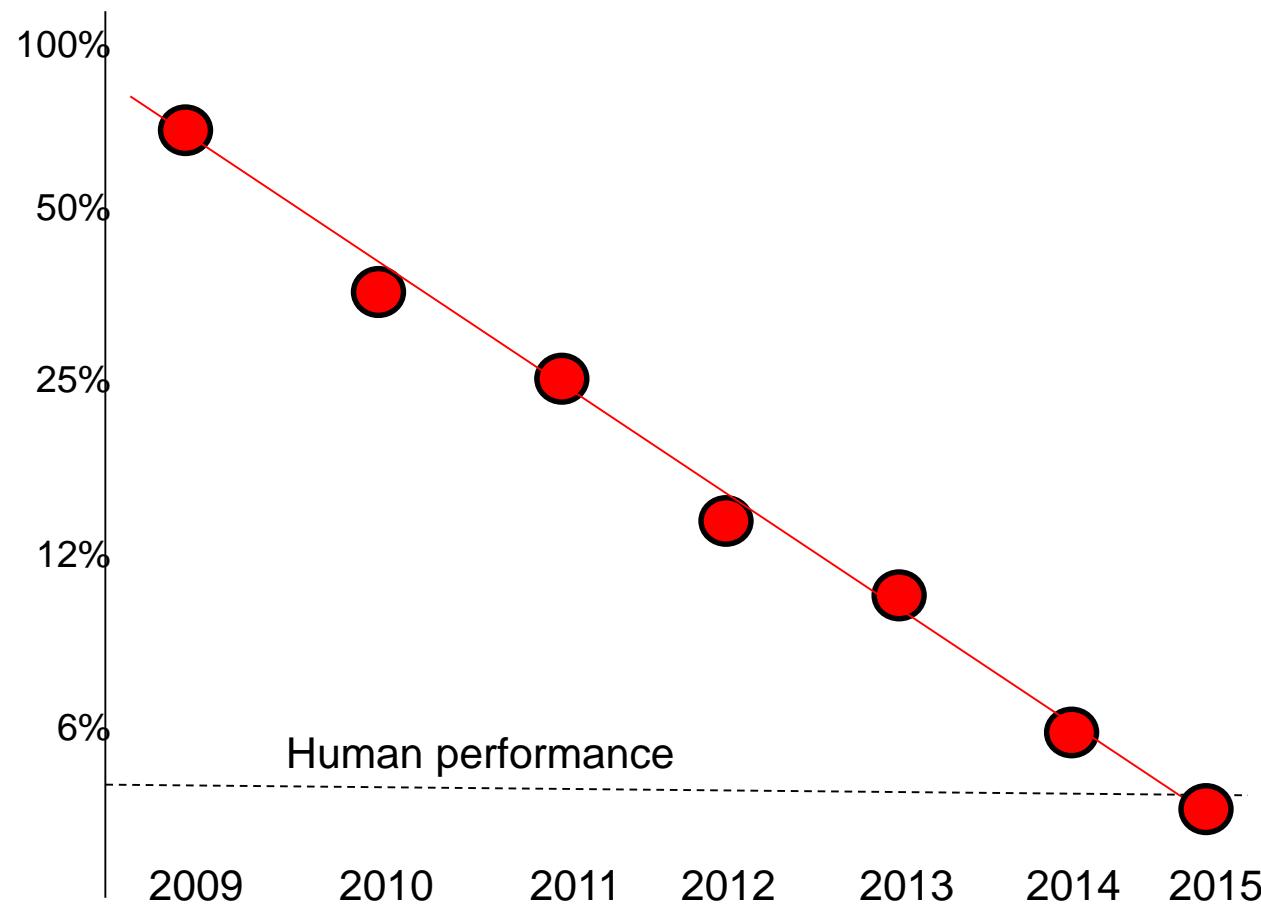
$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

- Precision-recall curves obtained by different thresholds on the classification score

- Single number: **average precision** (area under the curve)



Progress on image classification



Classification error on ImageNet

- Image enhancement
- Superresolution
- Body part and pose estimation
- Action recognition
- Depth from single image
- Optical flow estimation
- Disparity estimation
- Structure from motion
- Image based control
- Image based planning

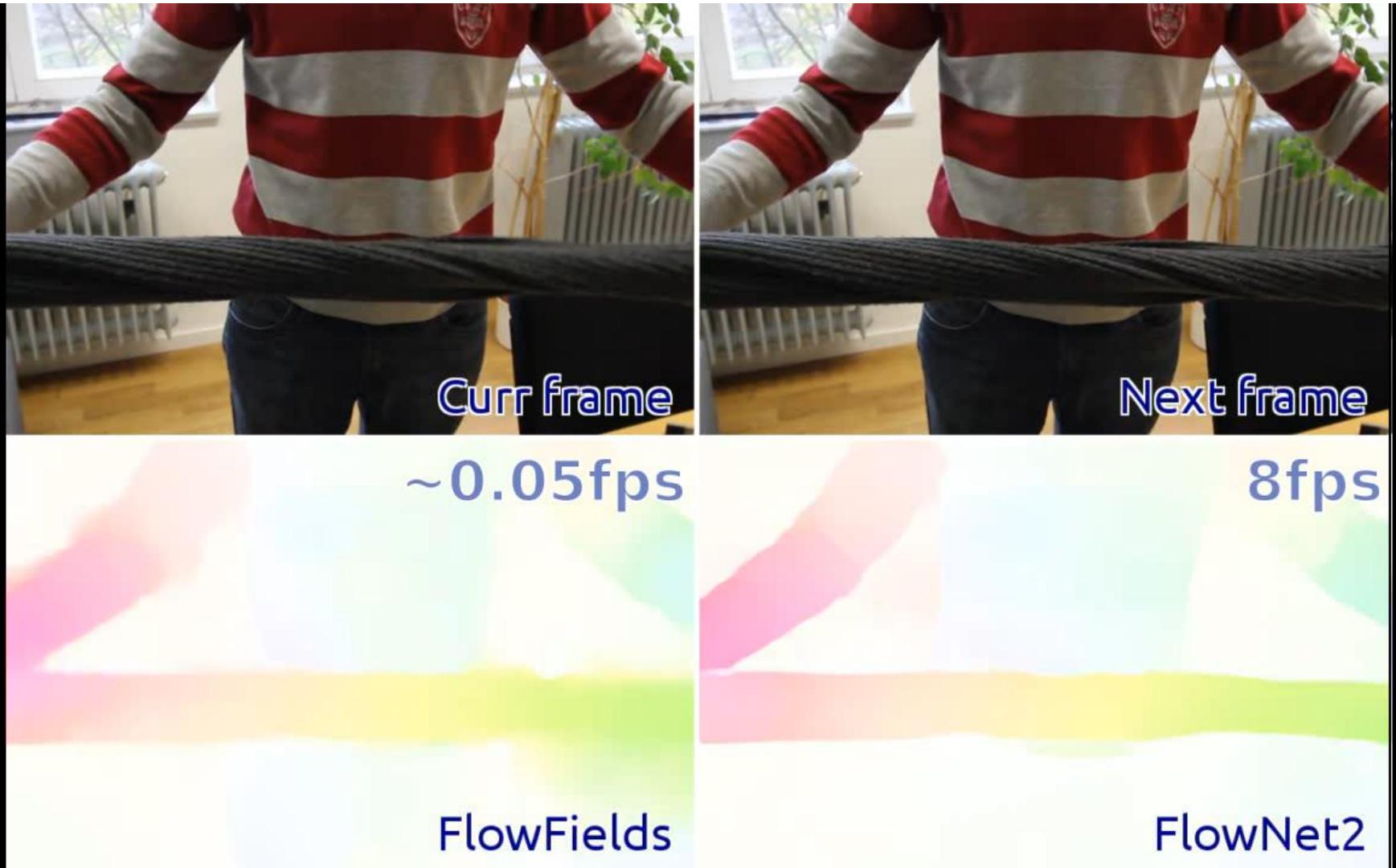
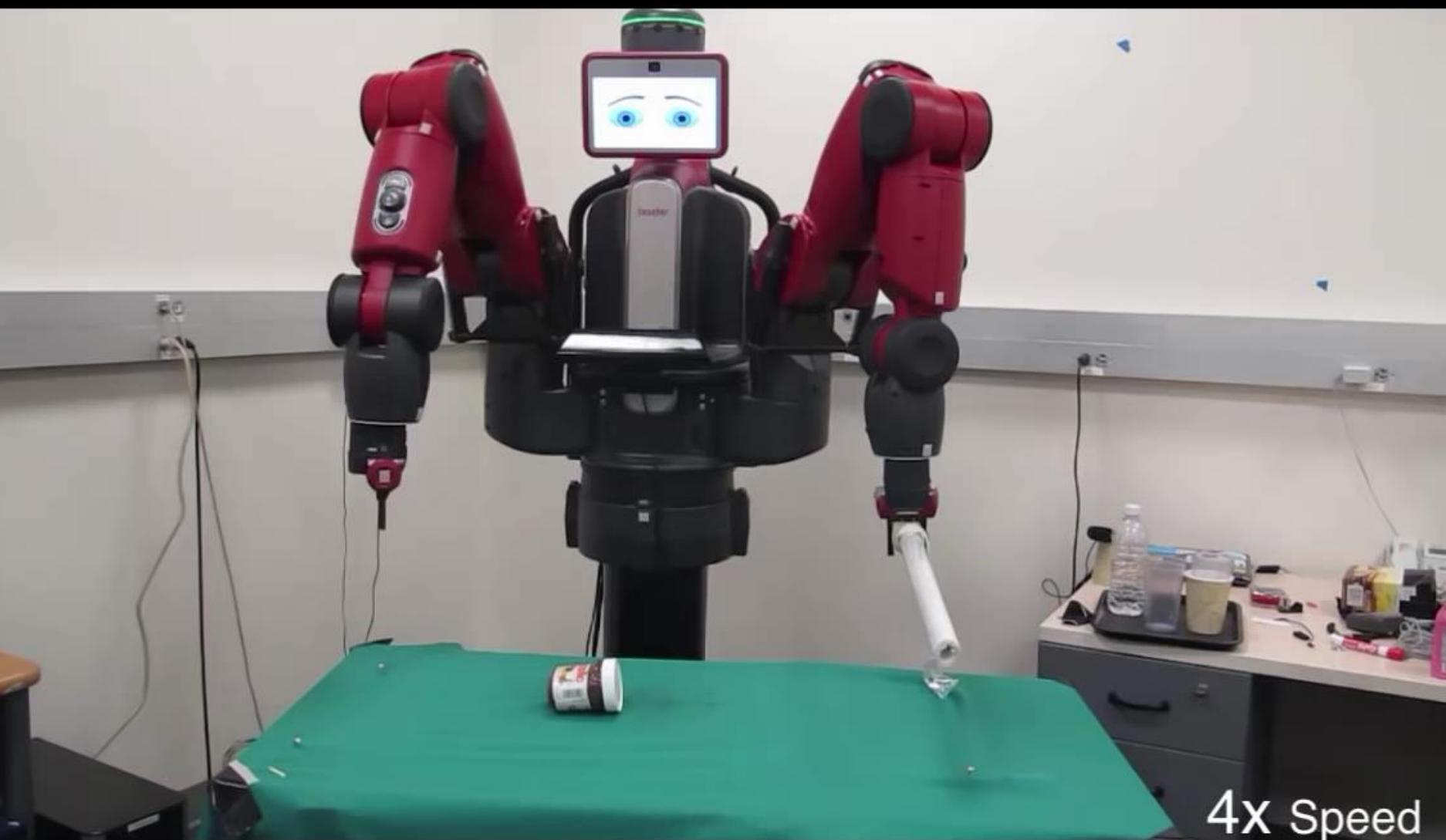


Image based control



Agrawal et al.: Learning to poke by poking

- Recognition tasks consist of a feature representation and a classifier
- Deep learning learns a hierarchical feature representation that is usually more powerful than hand-crafted features.
- Deep learning comes down to optimizing a simple (but highly nonlinear and non-convex) loss function with gradient descent
- Convolutional networks use localized filters that are shared across the whole image → vast reduction in the number of parameters
- Object detection and object segmentation require localization of the object (also possible with special convolutional network architectures)
- More or less all tasks can be formulated as learning problems

References

- A. Krizhevsky, I. Sutskever, G. Hinton: Imagenet classification with deep convolutional neural networks, *Neural Information Processing Systems (NIPS)*, 2012.
- P. Viola, M. J. Jones: Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137-154, 2004.
- P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun: Overfeat: Integrated recognition, localization and detection using convolutional networks, *International Conference on Learning Representations (ICLR)*, 2014.
- R. Girshick, J. Donahue, T. Darrell, J. Malik: Rich feature hierarchies for accurate object detection and semantic segmentation, *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- J. Long, E. Shelhamer, T. Darrell: Fully convolutional networks for semantic segmentation, *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- O. Ronneberger, P. Fischer, T. Brox: U-Net: convolutional networks for biomedical image segmentation, *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

Upcoming classes and courses

- Next week: Computer Graphics
- If you liked the field of image processing and computer vision:
 - Statistical Pattern Recognition (summer, 2+2)
 - Computer Vision (winter, 2+2)
 - Seminar (winter and summer)
 - Deep Learning lab course (winter)
 - GPU Programming lab course (summer)
 - Projects, theses

Image Processing

Class 11

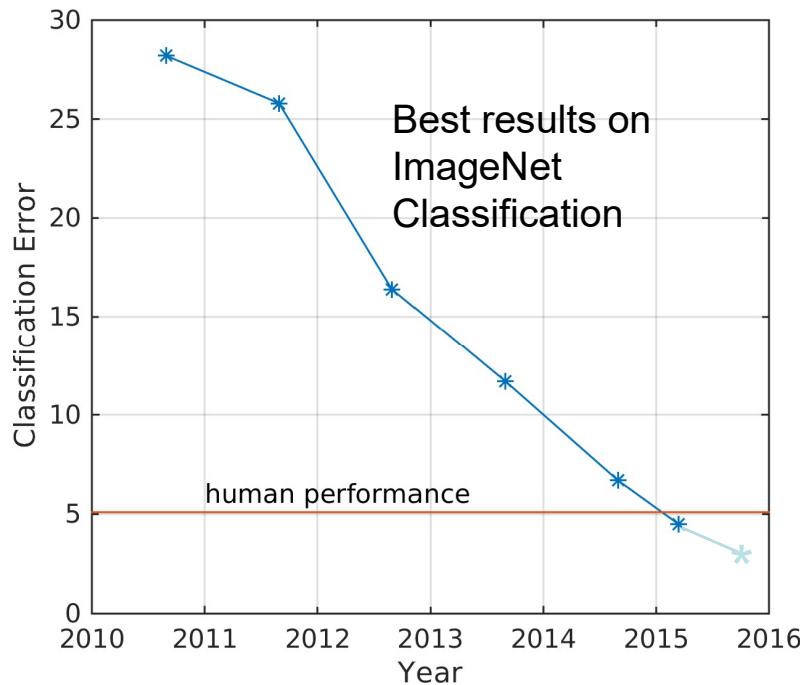
Deep Learning for Biomedical Image Analysis



Recap: Deep Learning with Artificial Neural Networks



Deep convolutional neural networks surpassed human-level performance in Feb. 2015



GT: mountain tent
 1: sleeping bag
 2: **mountain tent**
 3: parachute
 4: ski
 5: flagpole



GT: geyser
1: geyser
 2: volcano
 3: sandbar
 4: breakwater
 5: leatherback turtle

K. He, X. Zhang, S. Ren, J. Sun: "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." (2015) arXiv:1502.01852 [cs.CV]

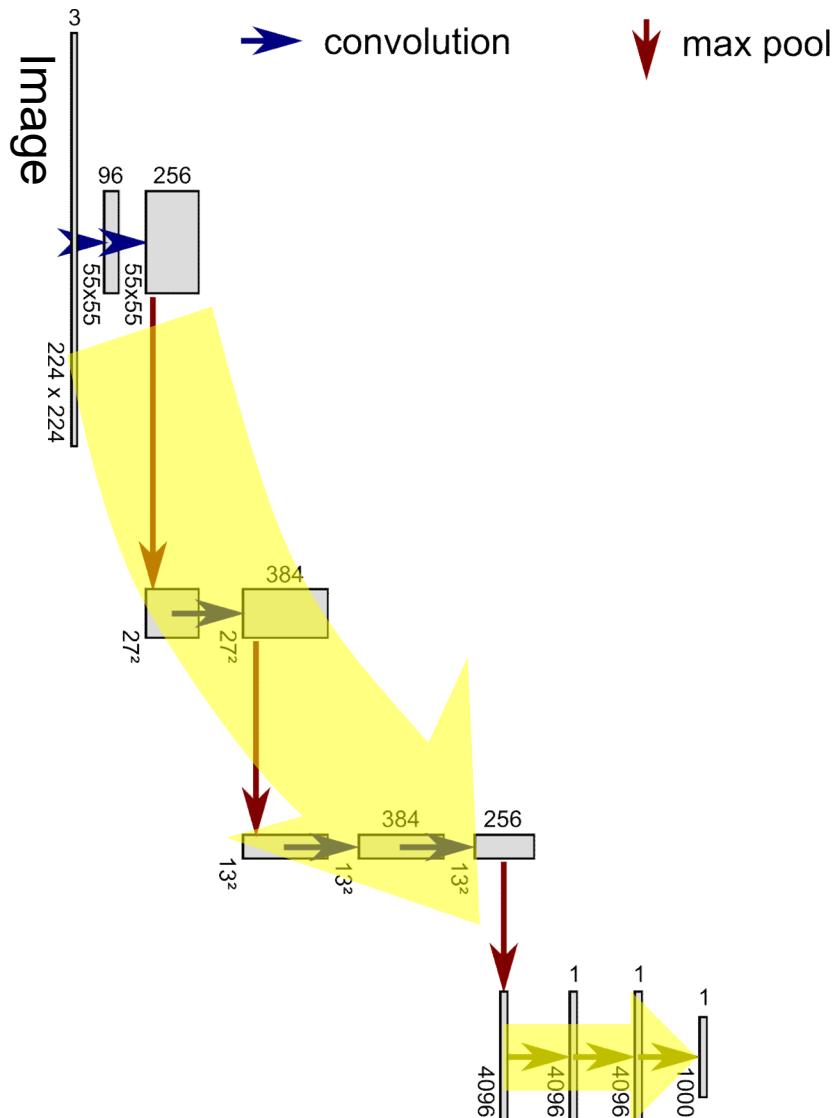
K. He, X. Zhang, S. Ren, J. Sun: "Deep Residual Learning for Image Recognition." (2015) arXiv:1512.03385 [cs.CV]

Training a neural network for image classification requires many labeled images

- E.g. “ImageNet”: 1.2 Mio. images from 1000 object classes



[O. Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge.” (2014) arXiv:1409.0575 [cs.CV]



Feature extraction

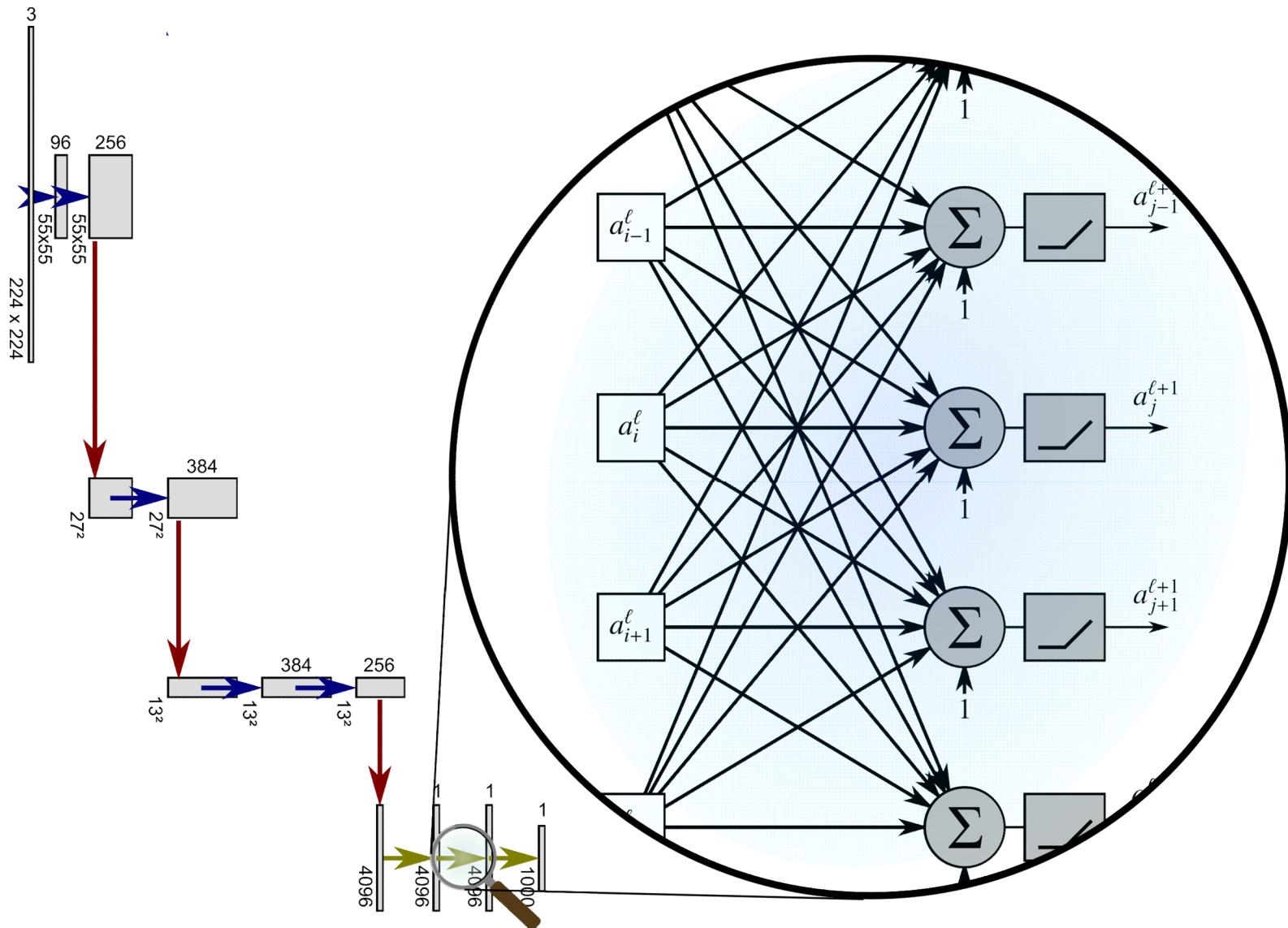
- Step-wise non-linear transformation of image content
 - Learned Filters (Convolution)
 - Subsampling (Max-Pooling)

Classification

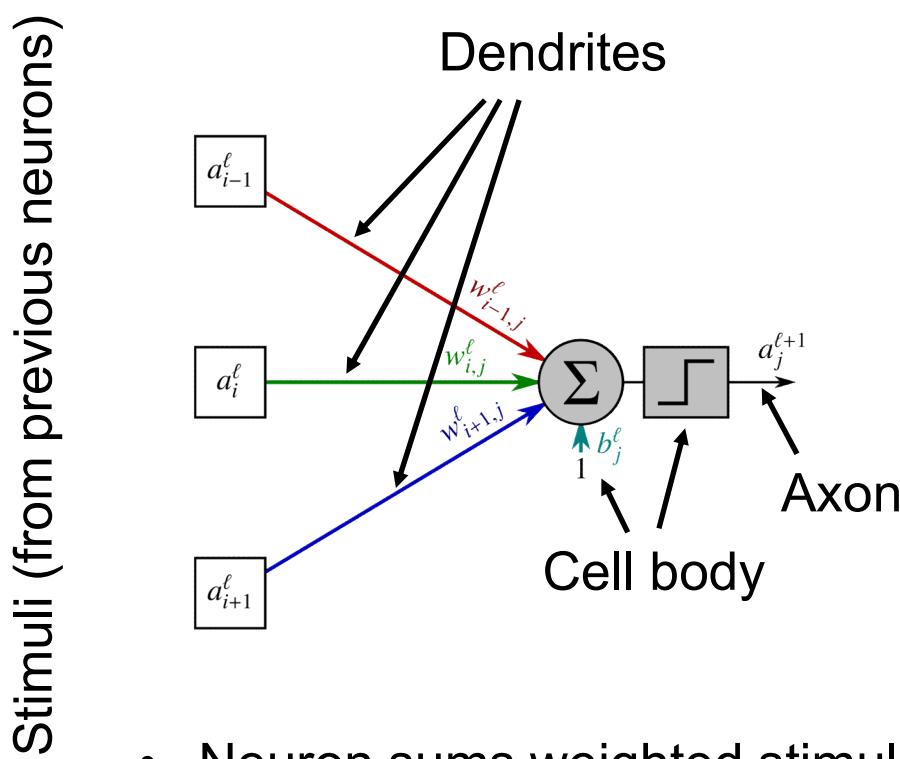
- Final representation is mapped to class label
 - Fully connected network

Both **automatically learned** from image/label pairs only!

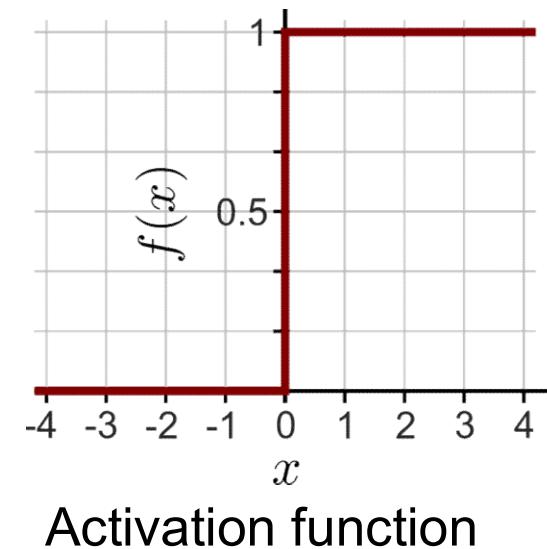
Fully Connected Networks for Classification



$$a_j^{\ell+1} = f \left(w_{i-1,j}^\ell a_{i-1}^\ell + w_{i,j}^\ell a_i^\ell + w_{i+1,j}^\ell a_{i+1}^\ell + b_j^\ell \right)$$



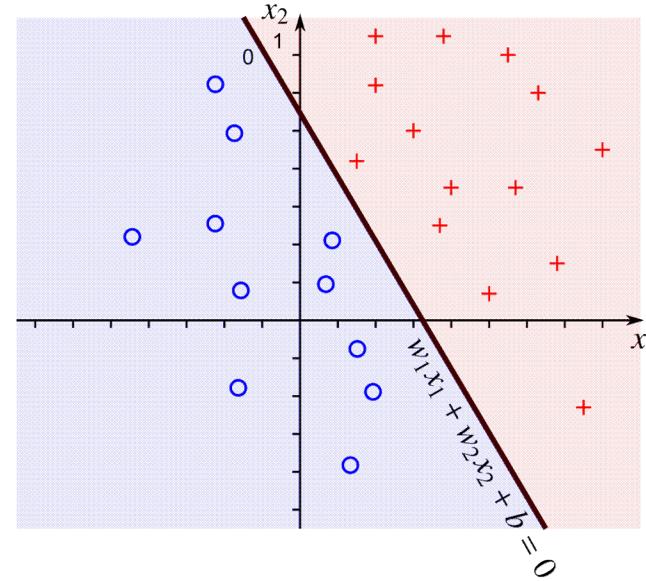
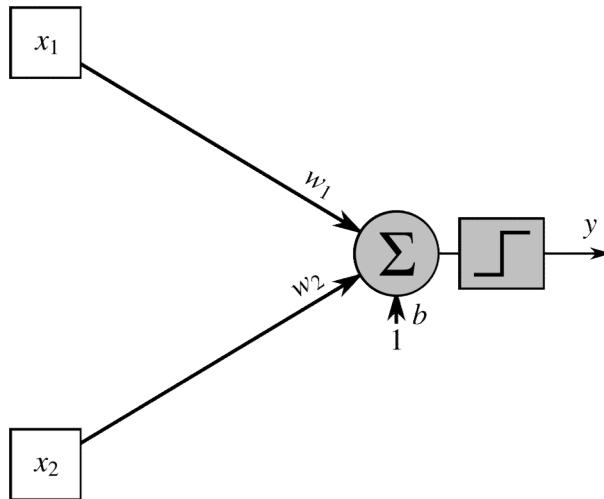
4 learnable parameters



- Neuron sums weighted stimuli from previous neurons
- Then transforms the result into binary decision (neuron fires)

Idea: Divide Input Space in Two Halves

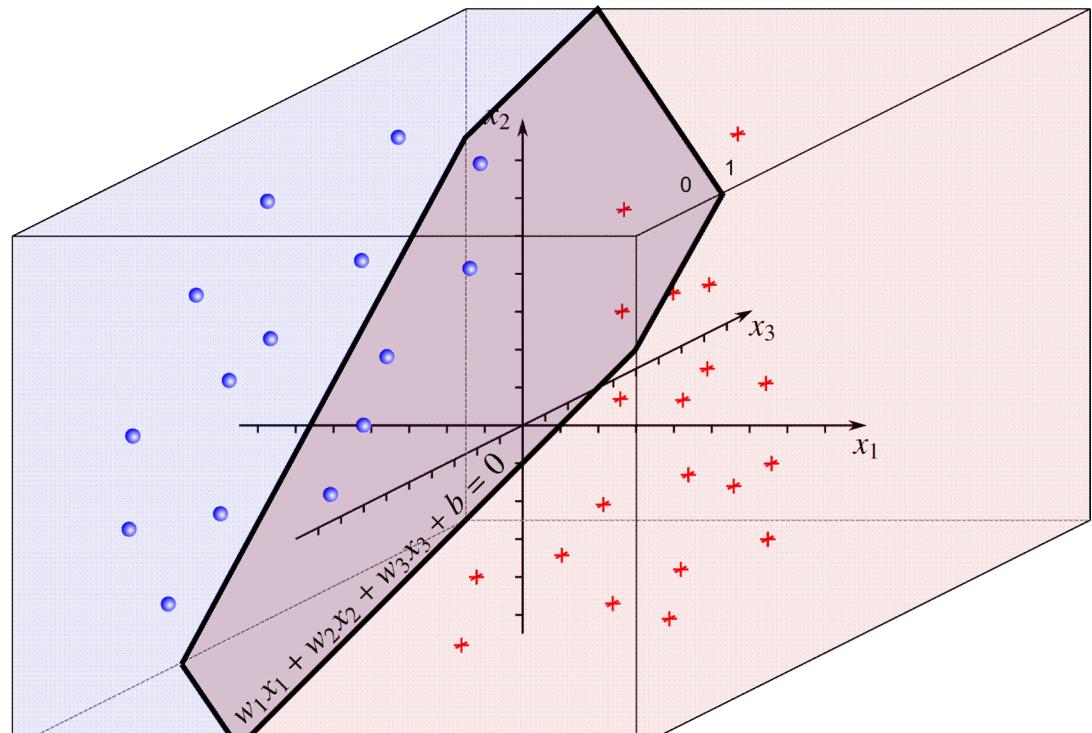
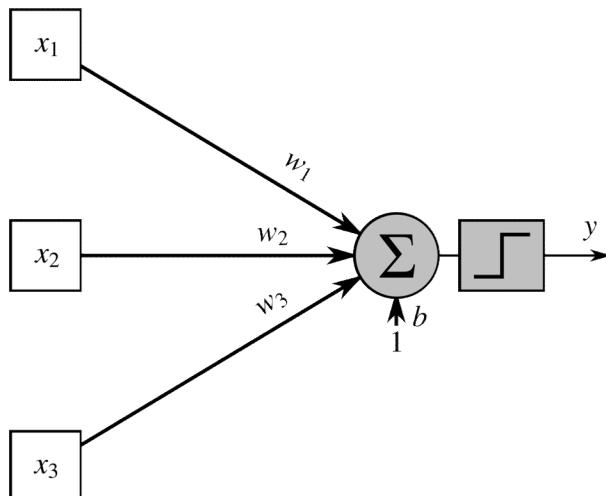
- Weights of a **single neuron model a line** in 2D space
 - Inputs left to the line → class 0 (circle)
 - Inputs right of the line → class 1 (cross)



Input / Feature space



- Weights of a **single neuron model a plane** in 3D space
 - Inputs left to the plane → class 0 (sphere)
 - Inputs right of the plane → class 1 (cross)

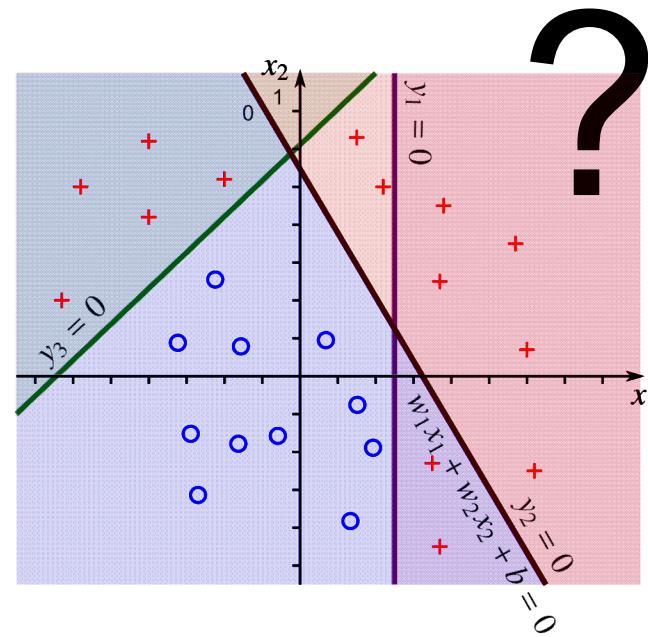
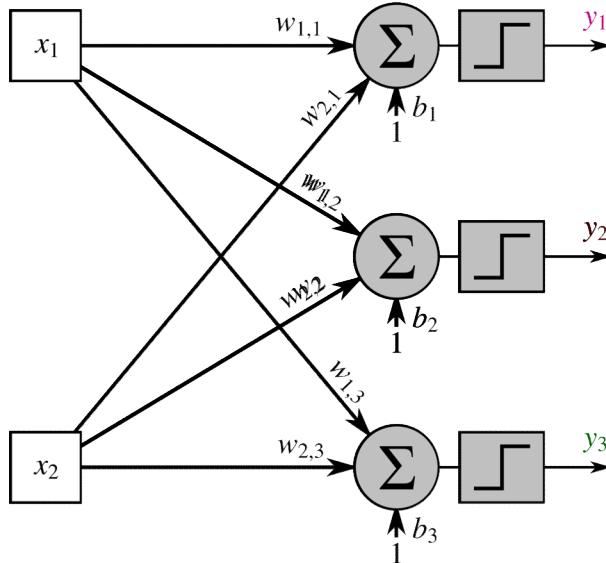


Input / Feature space



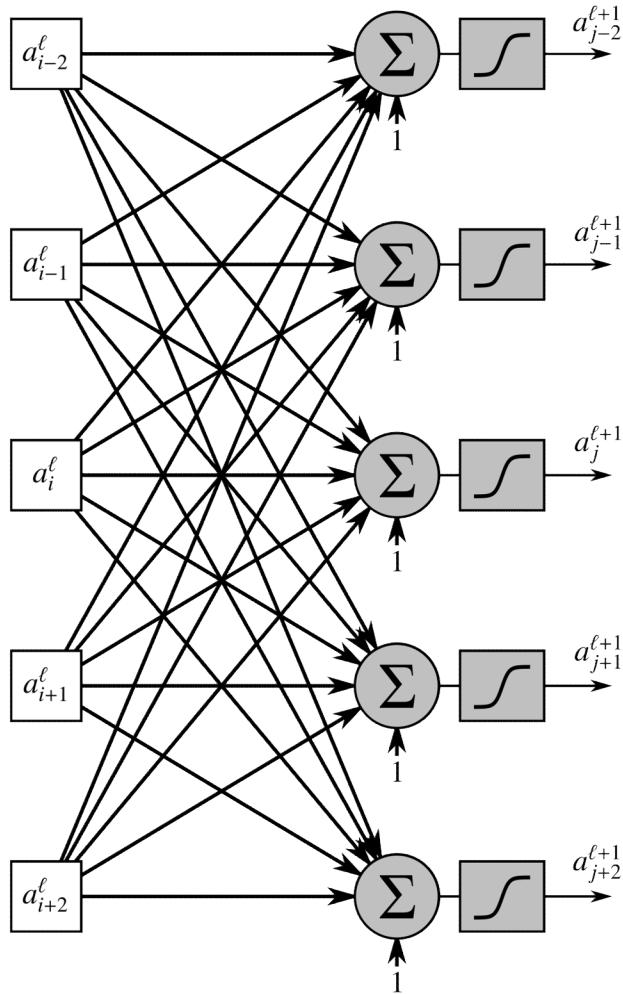
Idea: Divide Input Space in Two Halves

- What if inputs are **not linearly separable**?
- Add more lines = Add **more output neurons!**
- **Postpone classification** to following layers
- **3 layers** are sufficient to learn **classification of arbitrary inputs!**



Input / Feature space

Stimuli (from previous neurons)



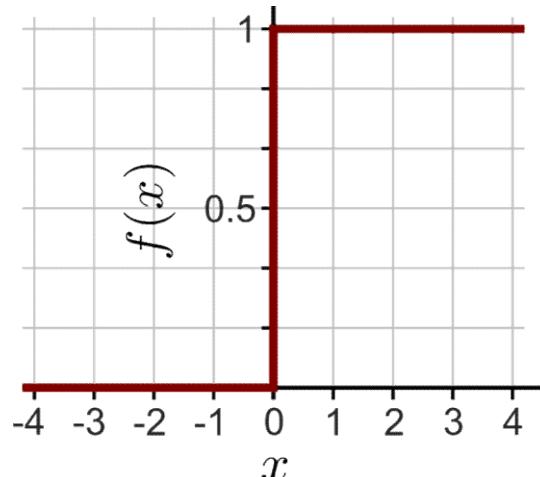
$$a_j^{\ell+1} = f \left(\sum_i w_{i,j}^{\ell} a_i^{\ell} + b_j^{\ell} \right)$$

$5 \times 5 + 5 = 30$ learnable parameters!

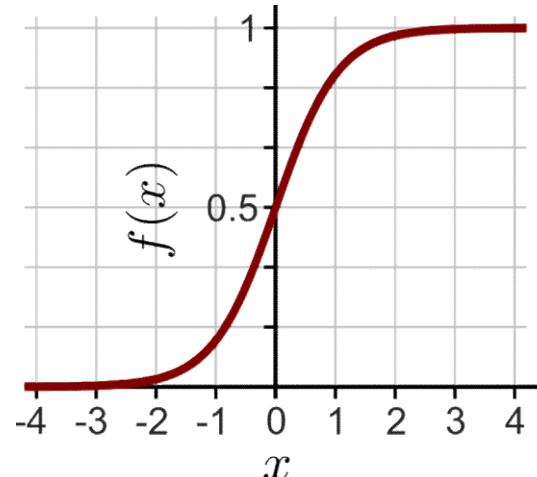
Many parameters → High capacity

- Can learn arbitrary dependencies between inputs
- Tends to **learn by heart** (over-fits training data)
- **Bad generalization** to unseen data

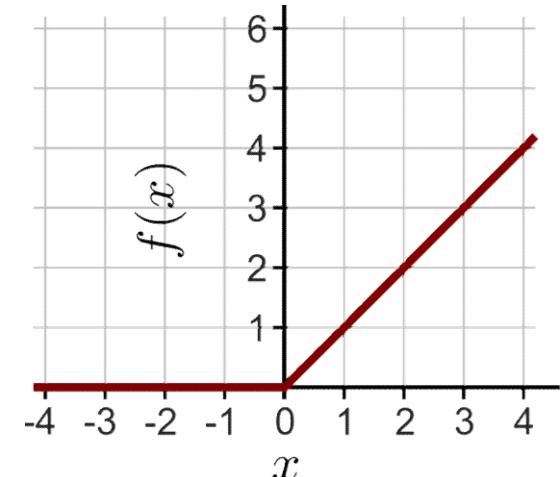




Step Activation



Sigmoid Activation



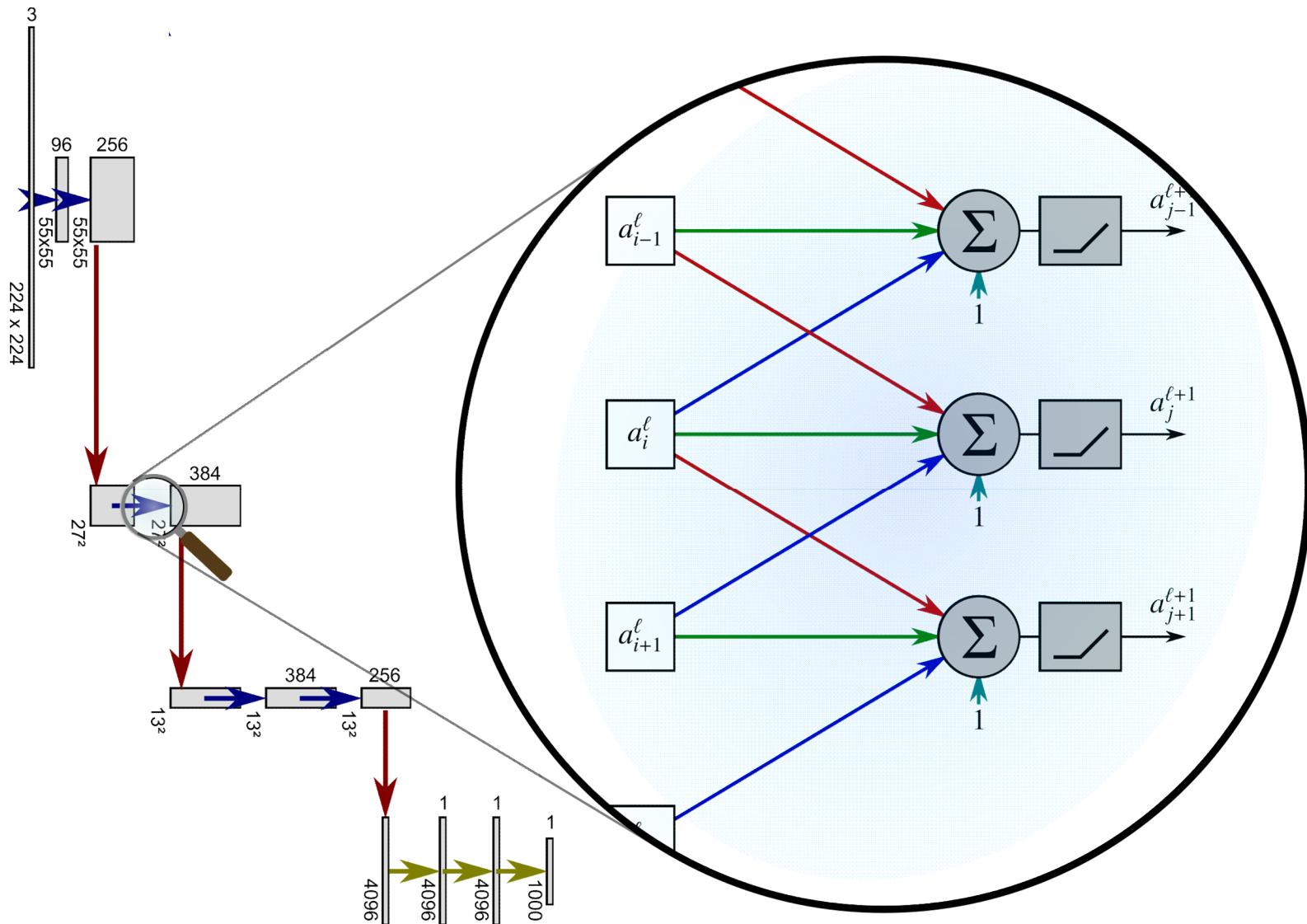
Rectified Linear Unit

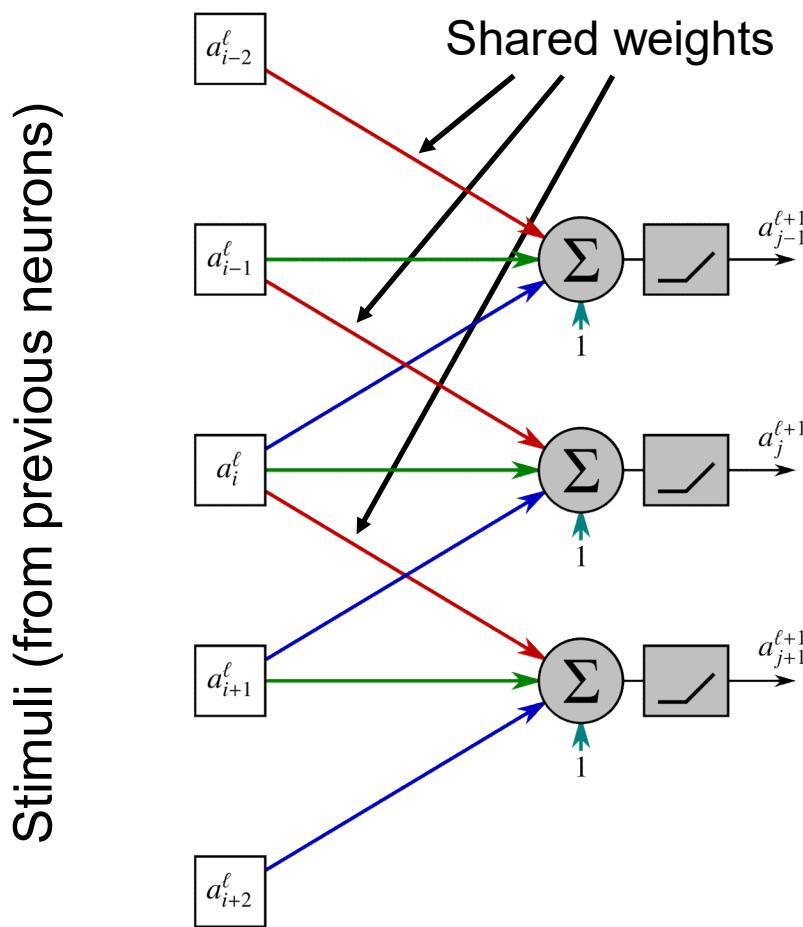
Many optimizers rely on derivatives

- **Derivative of step is zero**, except for $x=0$ where it is **undefined**
- Idea: Use **smooth step approximation**: Sigmoid
 - Downside: Quite expensive to evaluate and differentiate
- New Idea: Use a **mainly linear function** (ReLU)
 - Easy derivatives (derivative at $x=0$ must be defined to be 0 or 1)



Convolutional Networks for Feature Extraction





$$a_i^{\ell+1} = f \left(w_{-1}^\ell a_{i-1}^\ell + w_0^\ell a_i^\ell + w_1^\ell a_{i+1}^\ell + b^\ell \right)$$

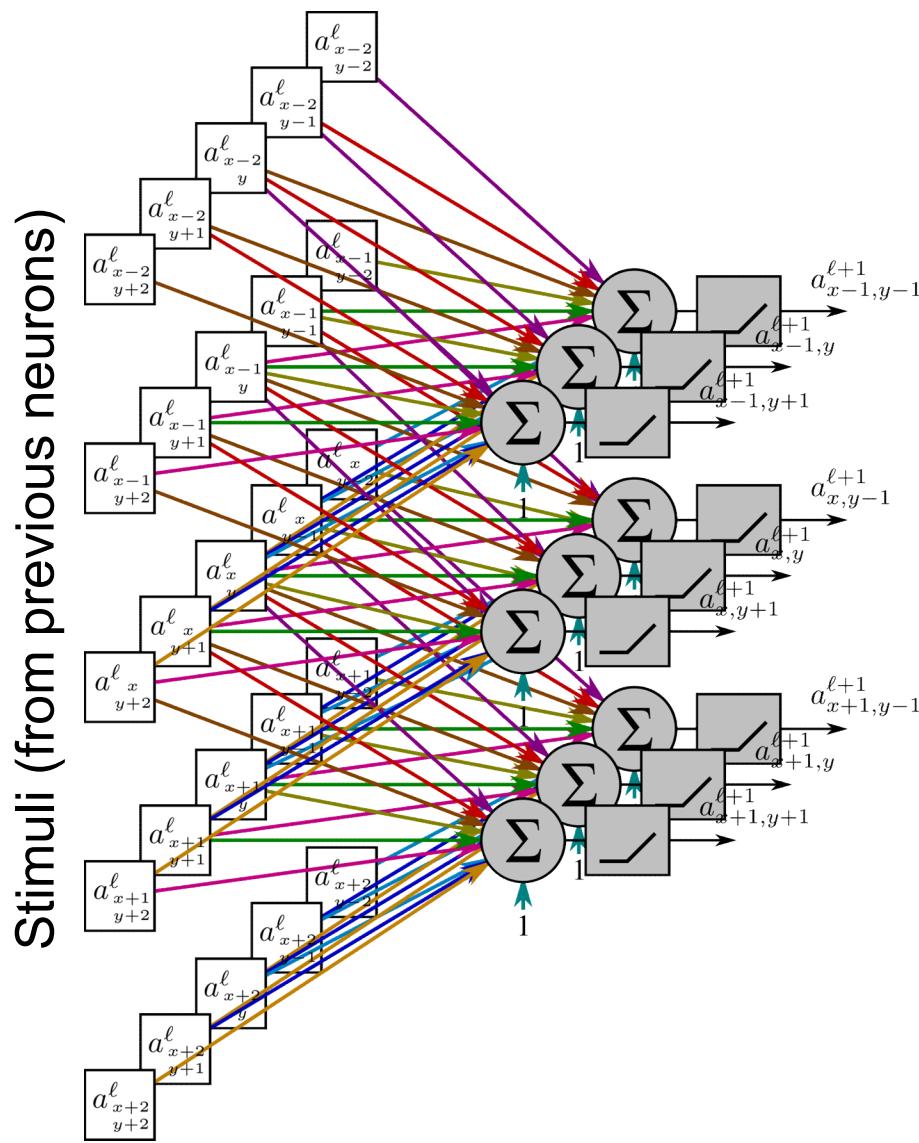
Only 4 learnable parameters!

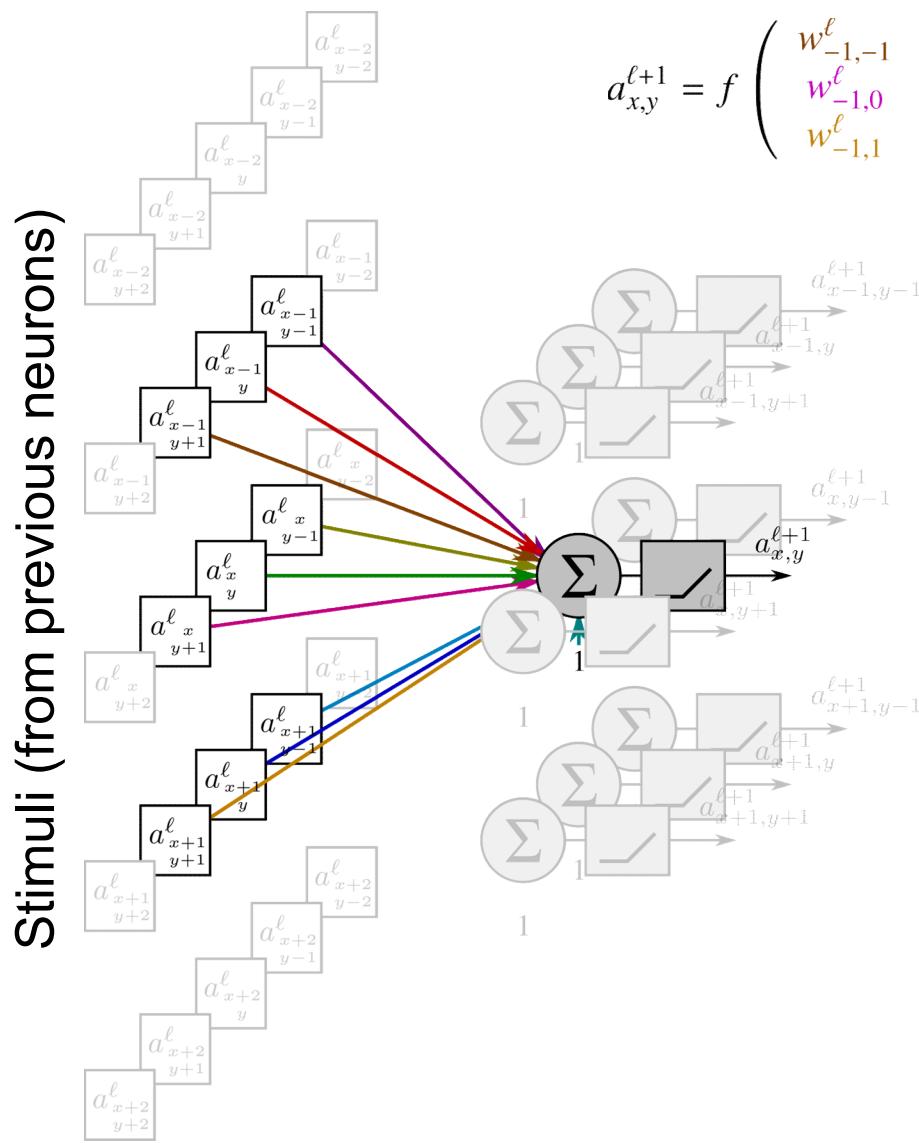
Exploits image structure

Lower capacity

- Must learn **underlying concept**
- **Better generalization**







10 learnable parameters!

Learns a non-linear filter

- Coefficients = Weights

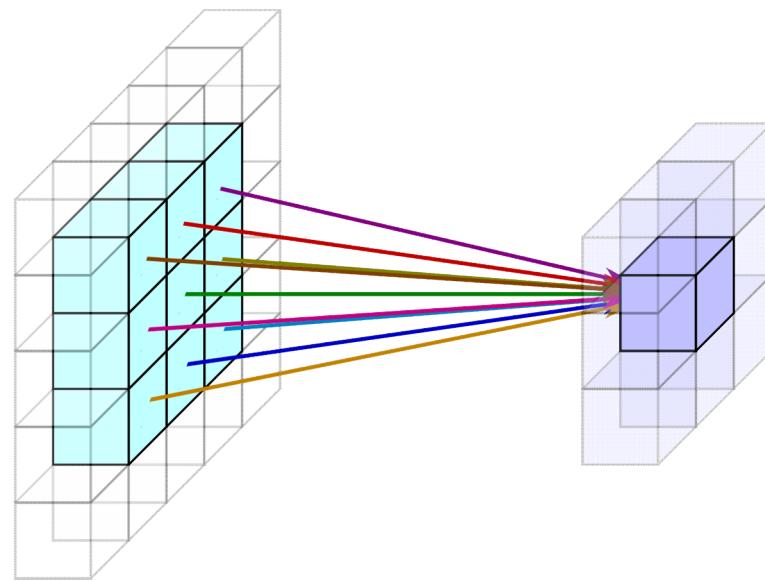
Filter size adjustable

- Here: 3x3
- Common: 3x3, ..., 11x11

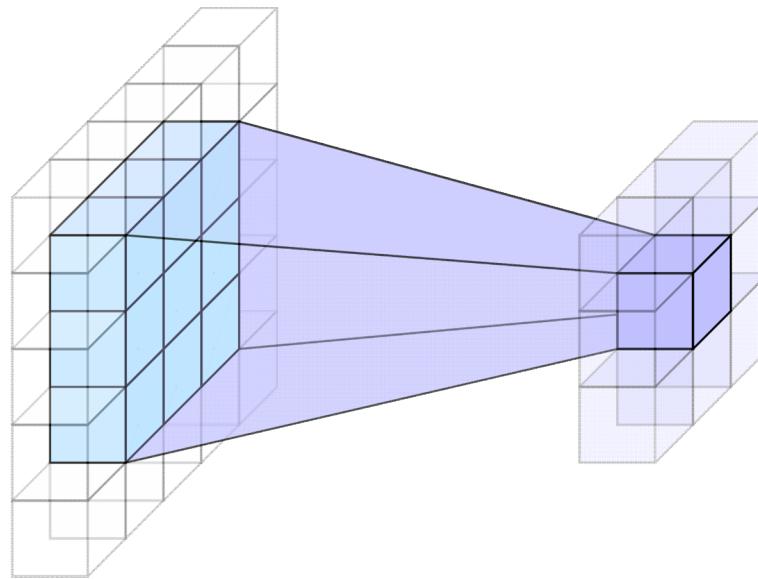
2D Convolutional Neuronal Networks

$$a_{x,y}^{\ell+1} = f \left(\begin{array}{ccccccccc} w_{-1,-1}^\ell & a_{x-1,y-1}^\ell & + & w_{0,-1}^\ell & a_{x,y-1}^\ell & + & w_{1,-1}^\ell & a_{x+1,y-1}^\ell & + \\ w_{-1,0}^\ell & a_{x-1,y}^\ell & + & w_{0,0}^\ell & a_{x,y}^\ell & + & w_{1,0}^\ell & a_{x+1,y}^\ell & + \\ w_{-1,1}^\ell & a_{x-1,y+1}^\ell & + & w_{0,1}^\ell & a_{x,y+1}^\ell & + & w_{1,1}^\ell & a_{x+1,y+1}^\ell & + \end{array} b^\ell \right)$$

Stimuli (from previous neurons)



Stimuli (from previous neurons)



$$a_{x,y}^{\ell+1} = f \left(\sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^\ell a_{x+x',y+y'}^\ell + b^\ell \right)$$

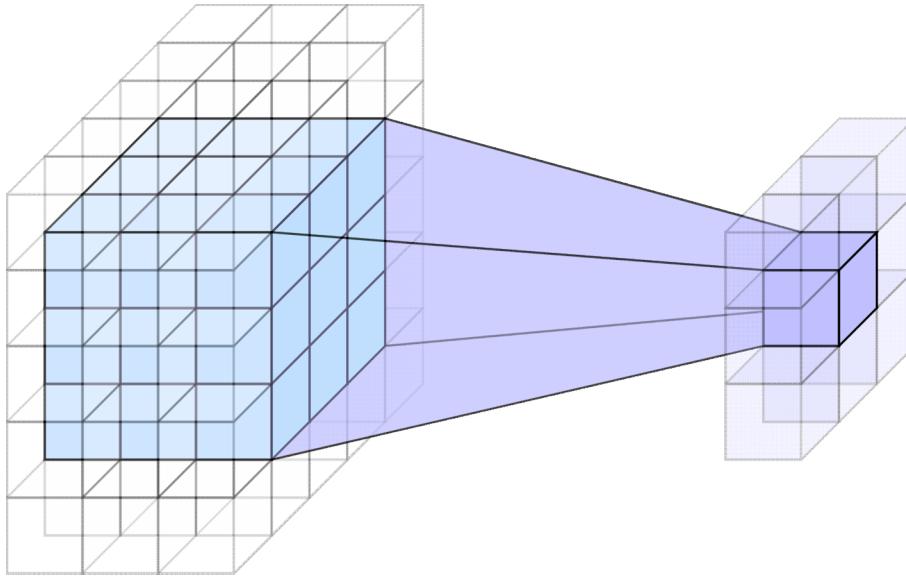
Single input channel:

- Nonlinear 2D-filter



2D Convolutional Neuronal Networks – Multiple Channels

Stimuli (from previous neurons)



$$a_{x,y}^{\ell+1} = f \left(\sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^{\ell,c_{\text{in}}} a_{x+x',y+y'}^{\ell,c_{\text{in}}} + b^{\ell} \right)$$

Single input channel:

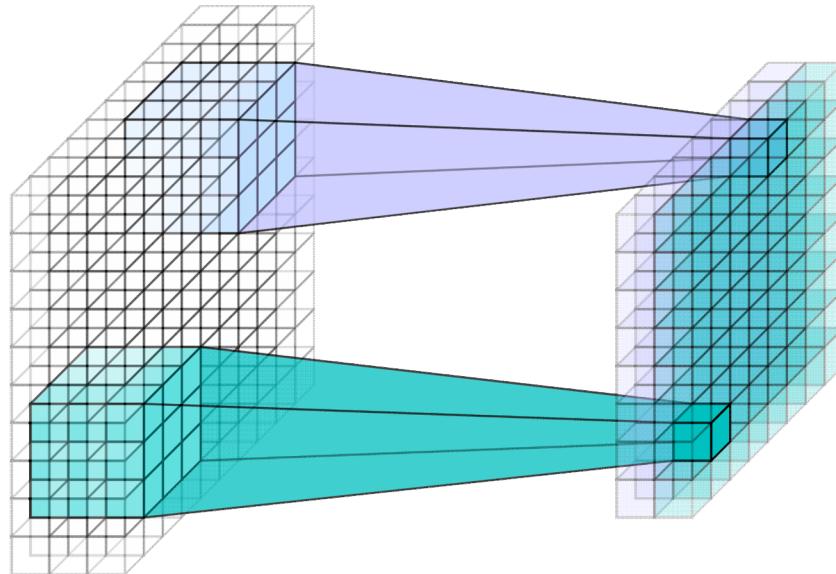
- Nonlinear 2D-filter

Multiple input channels:

- Simply extend filter along channel dimension

2D Convolutional Neuronal Networks – Multiple Filters

Stimuli (from previous neurons)



$$a_{x,y}^{\ell+1,c_{\text{out}}} = f \left(\sum_{c_{\text{in}}=1}^{C_{\text{in}}} \sum_{x'=-\lfloor \frac{k_x}{2} \rfloor}^{\lfloor \frac{k_x}{2} \rfloor} \sum_{y'=-\lfloor \frac{k_y}{2} \rfloor}^{\lfloor \frac{k_y}{2} \rfloor} w_{x',y'}^{\ell,c_{\text{in}},c_{\text{out}}} a_{x+x',y+y'}^{\ell,c_{\text{in}}} + b^{\ell,c_{\text{out}}} \right)$$

Single input channel:

- Nonlinear 2D-filter

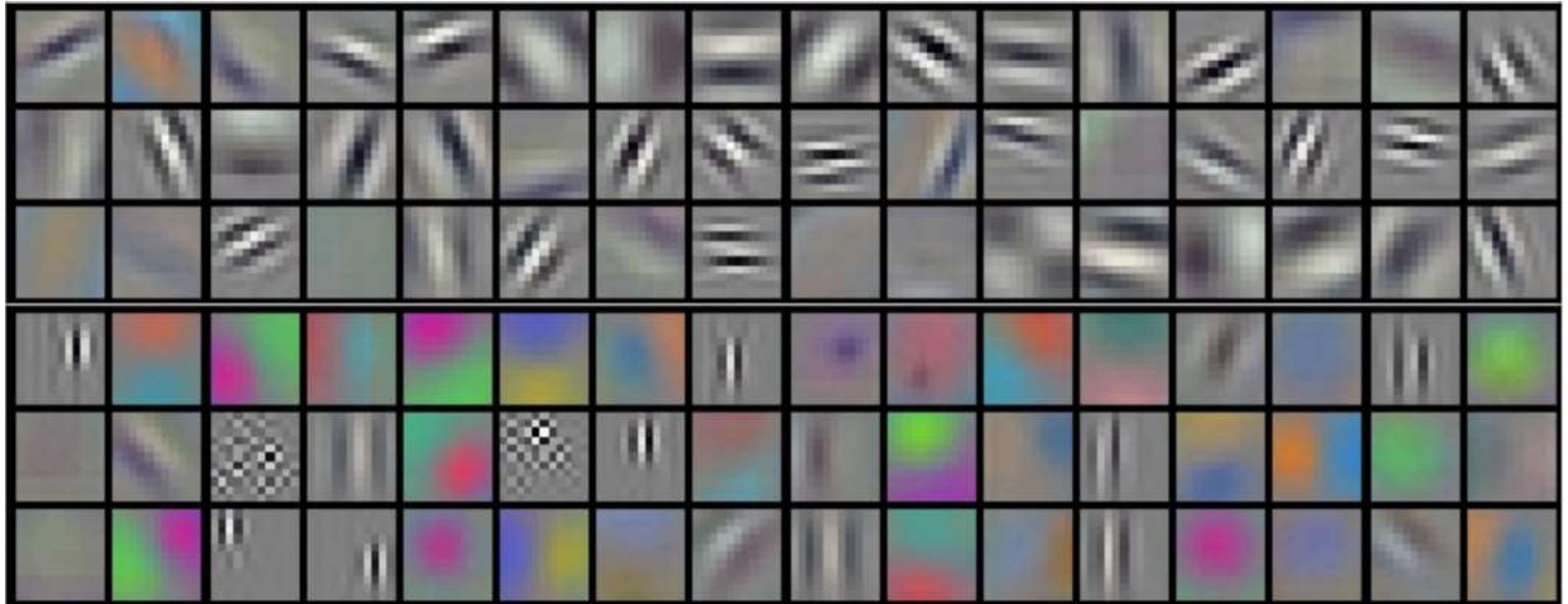
Multiple input channels:

- Simply extend filter along channel dimension

Multiple filters:

- Generate one output channel per filter





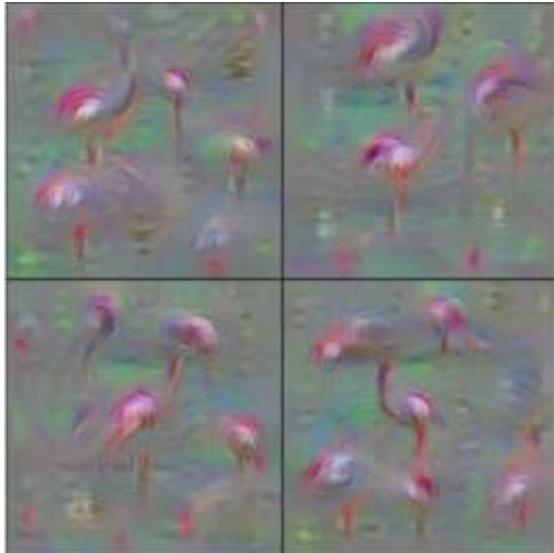
[Krizhevski et al., NIPS 2012]

Filters in the first layer encode local structure

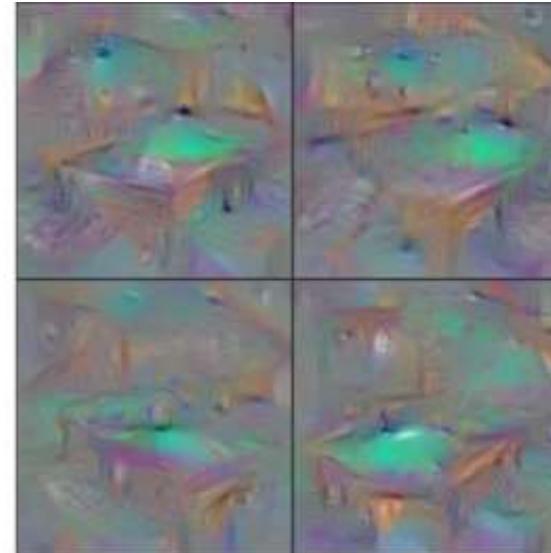
- Edges (variable frequency and orientation), texture, color blobs, ...
- Similar to the first ganglions in the eye's retina
- Deeper layers harder to interpret



Learned Filters (Deeper Layers)



Flamingo



Billard table



School bus

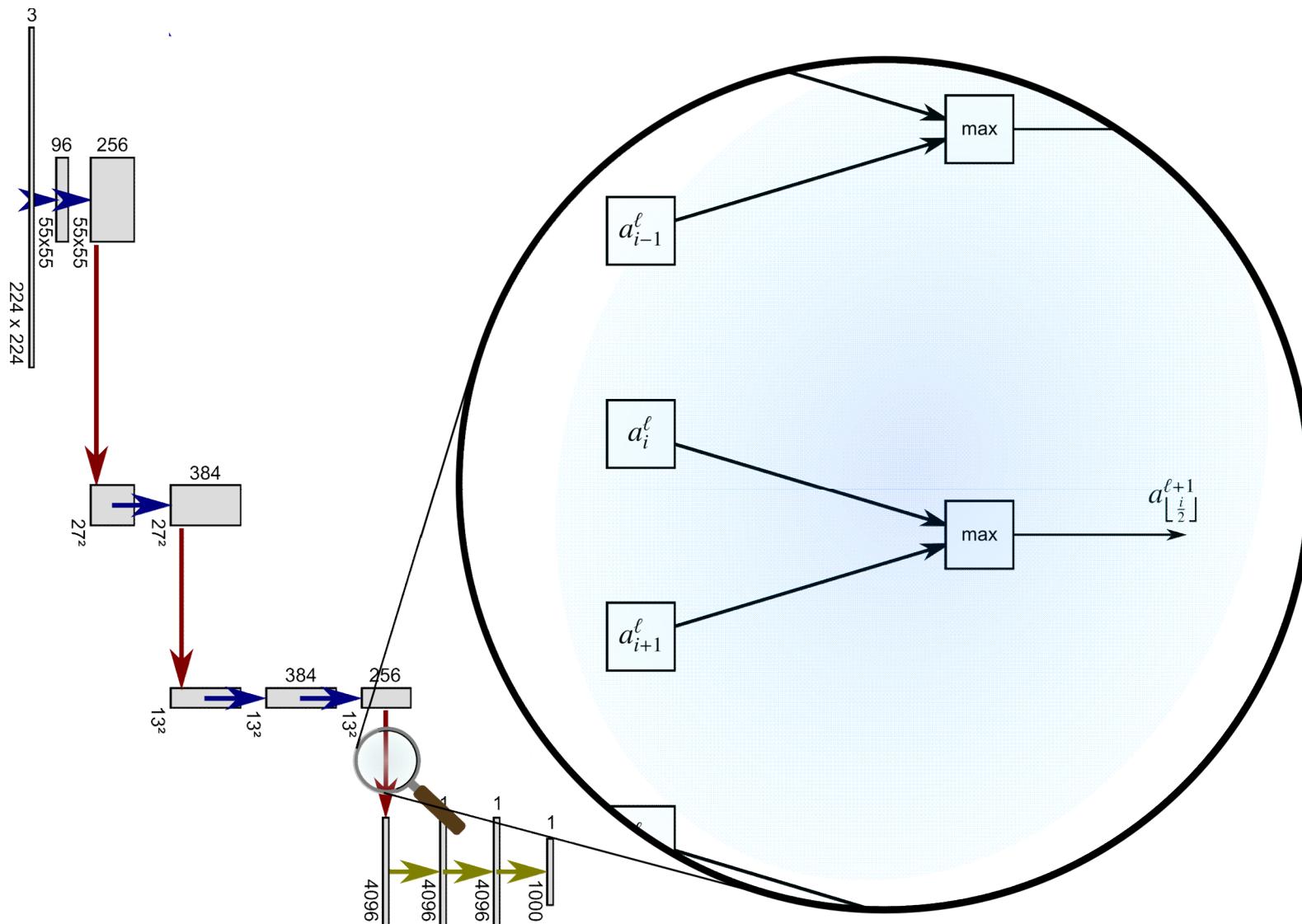
[Yosinski et al., ICML 2015]

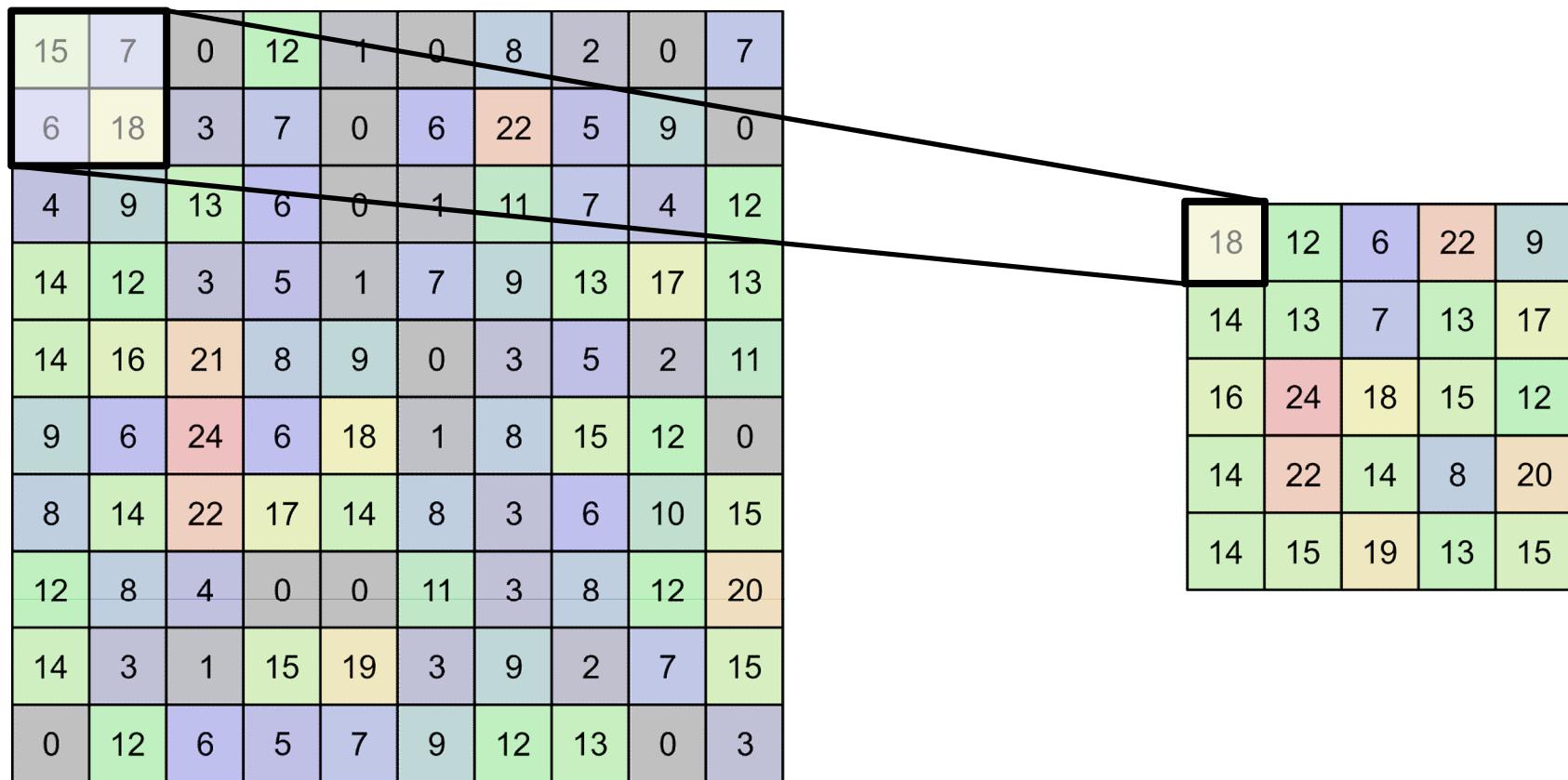
Filters combine abstract outputs of previous layers

- Don't show filter coefficients
- Generate images maximizing the response of selected neurons
 - See what the neuron "wants" to see



Max-Pooling to Increase Field of View





Simple sub-sampling

- No learned parameters
- Reduces output dimensions by factor 2
- Doubles the field-of-view of output neurons for each dimension

15	7	0	12	1	0	8	2	0	7
6	18	3	7	0	6	22	5	9	0
4	9	13	6	0	1	11	7	4	12
14	12	3	5	1	7	9	13	17	13
14	16	21	8	9	0	3	5	2	11
9	6	24	6	18	1	8	15	12	0
8	14	22	17	14	8	3	6	10	15
12	8	4	0	0	11	3	8	12	20
14	3	1	15	19	3	9	2	7	15
0	12	6	5	7	9	12	13	0	3

18	12	6	22	9
14	13	7	13	17
16	24	18	15	12
14	22	14	8	20
14	15	19	13	15

Simple sub-sampling

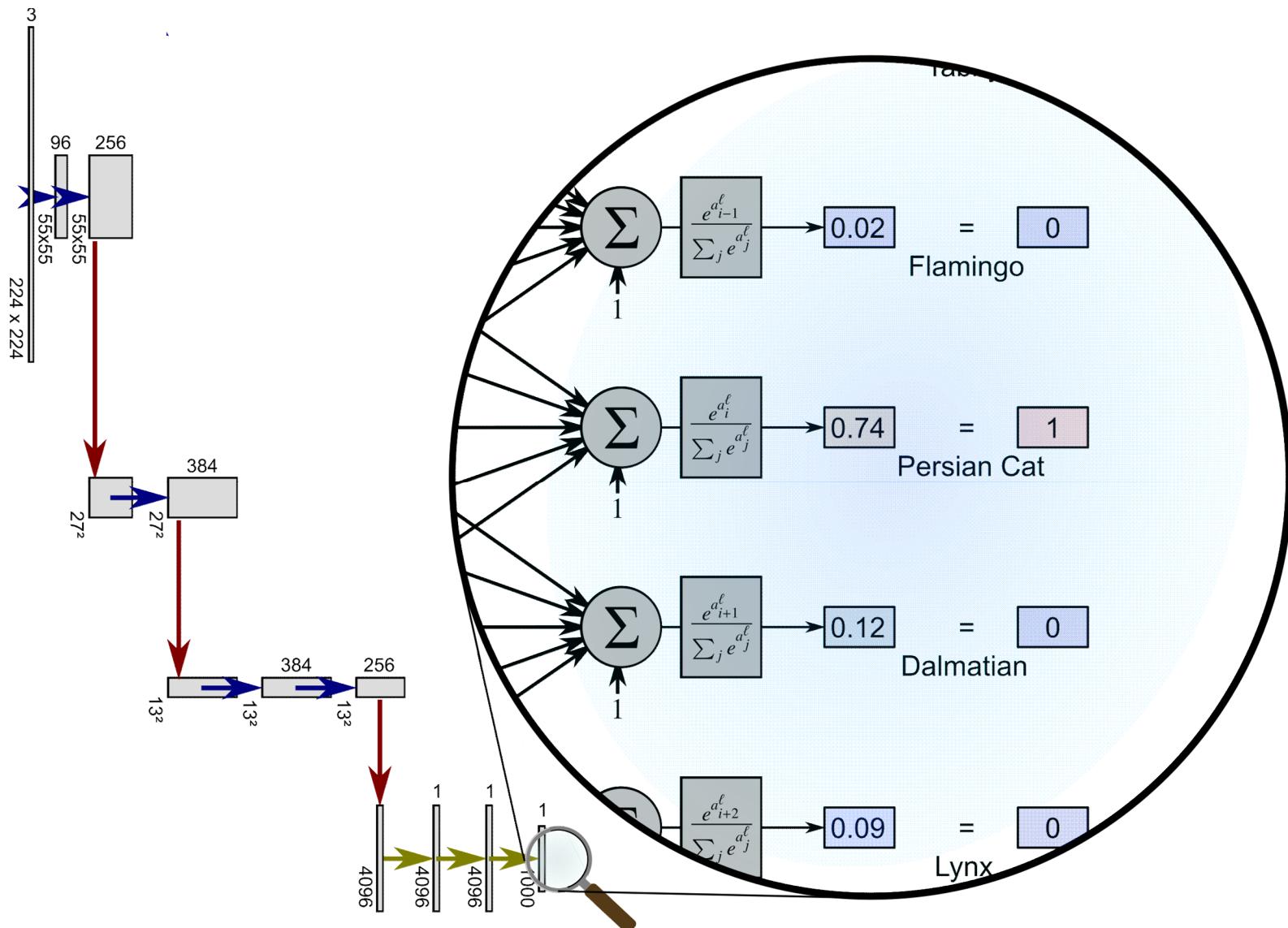
- No learned parameters
- Reduces output dimensions by factor 2
- Doubles the field-of-view of output neurons for each dimension

15	7	0	12		1	0	8	2	0	7
6	18	3	7		0	6	22	5	9	0
4	9	13	6		0	1	11	7	4	12
14	12	3	5	1	7	9	13	17	13	
14	16	21	8	9	0	3	5	2	11	
9	6	24	6	18	1	8	15	12	0	
8	14	22	17	14	8	3	6	10	15	
12	8	4	0	0	11	3	8	12	20	
14	3	1	15	19	3	9	2	7	15	
0	12	6	5	7	9	12	13	0	3	

18	12	6	22	9
14	13	7	13	17
16	24	18	15	12
14	22	14	8	20
14	15	19	13	15

Simple sub-sampling

- No learned parameters
- Reduces output dimensions by factor 2
- Doubles the field-of-view of output neurons for each dimension



Store class label as {0,1} vector

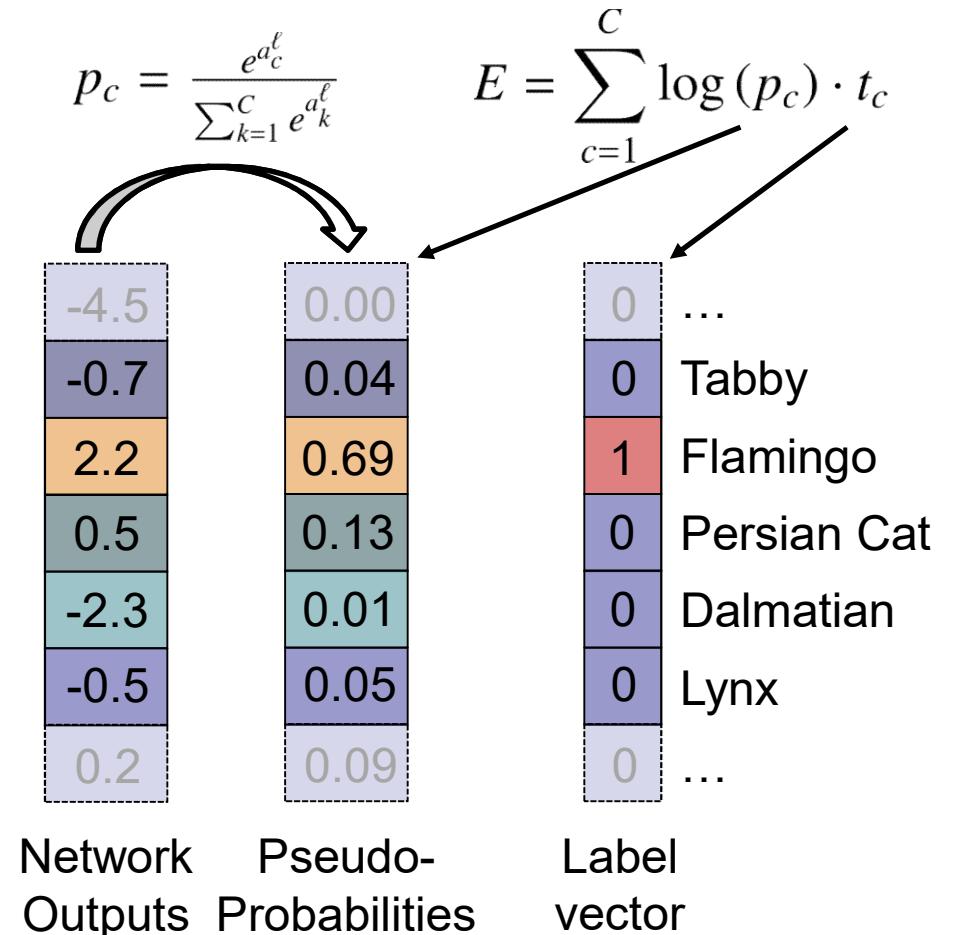
1 = this class

0 = other class

1. Convert **network outputs** to **pseudo probabilities** in [0,1]
 2. **Compare** pseudo probabilities with label vector
 3. **Sum** over all comparisons
- “**Energy**” E

Minimize E using an optimizer

Parameters: all network weights



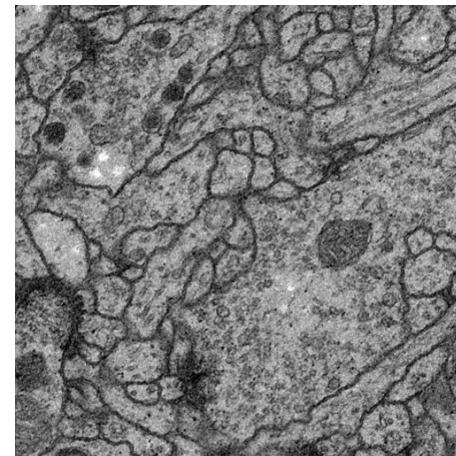
Application to (Real) Biomedical Problems



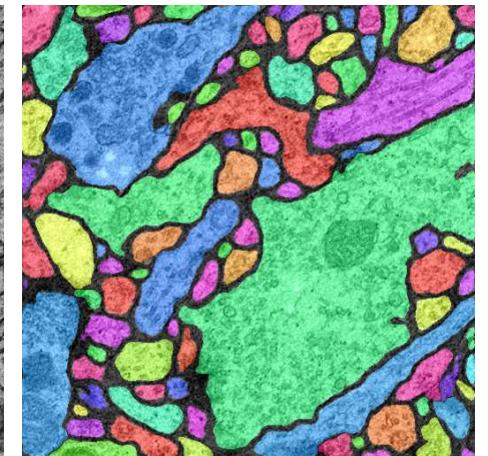
Image classification is of limited use in biology!

More interesting:

- Image segmentation
- Detection of specific structures



EM image



Segmentation

What is the difference?

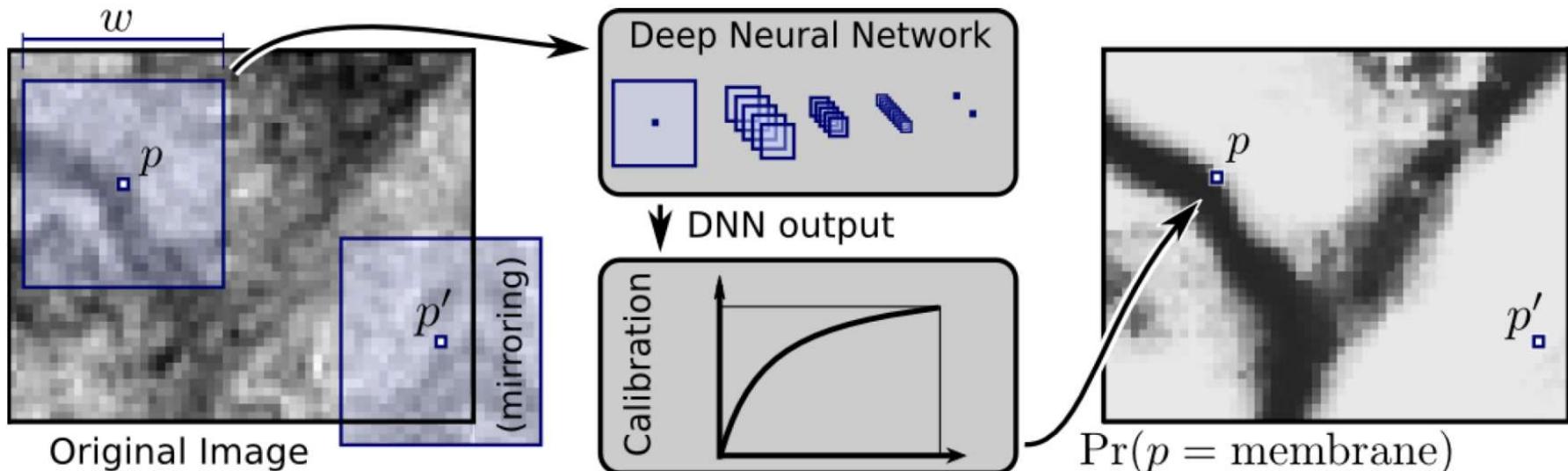
- Classification: **One class label per image**
- Segmentation/Detection: **One class label per pixel**

Naïve Idea: Use classification network with sliding window

- One network evaluation per pixel → very expensive, but works!



Pixel-wise classification using Sliding Windows



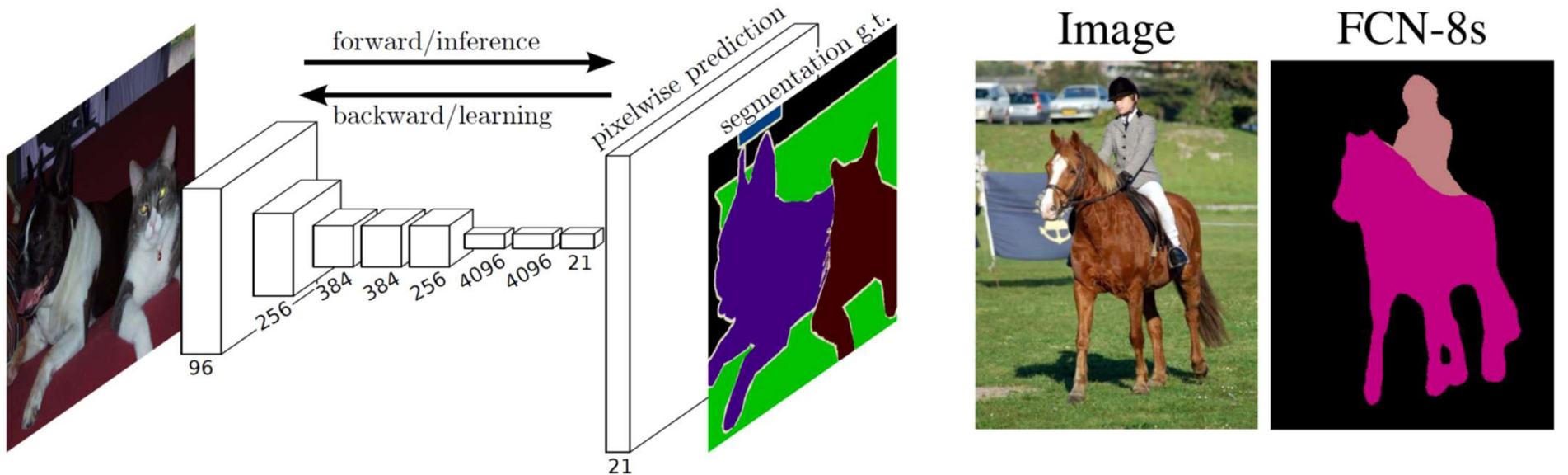
D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber - Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images (NIPS 2012)

Winner of the EM segmentation challenge (ISBI 2012)

- ✓ Advantage: intuitive
- ✗ One network evaluation per pixel → extremely slow!
- ✗ Evaluations highly redundant

Idea: Exploit redundancy → parallelize classification

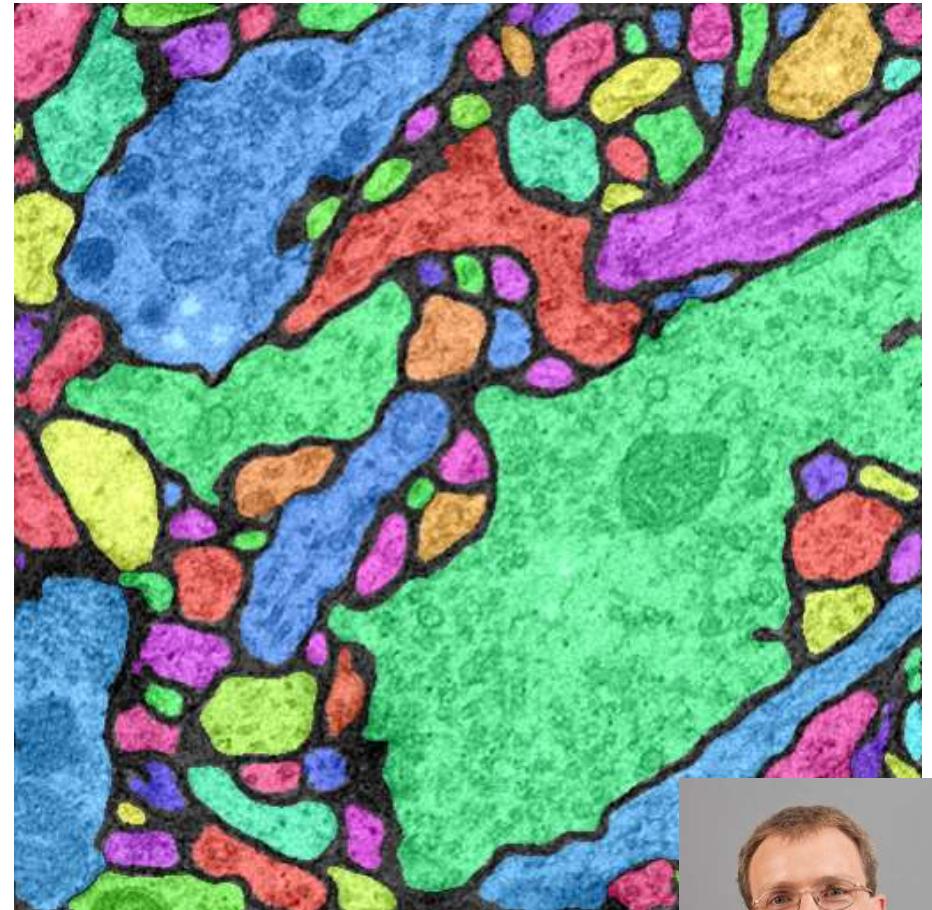
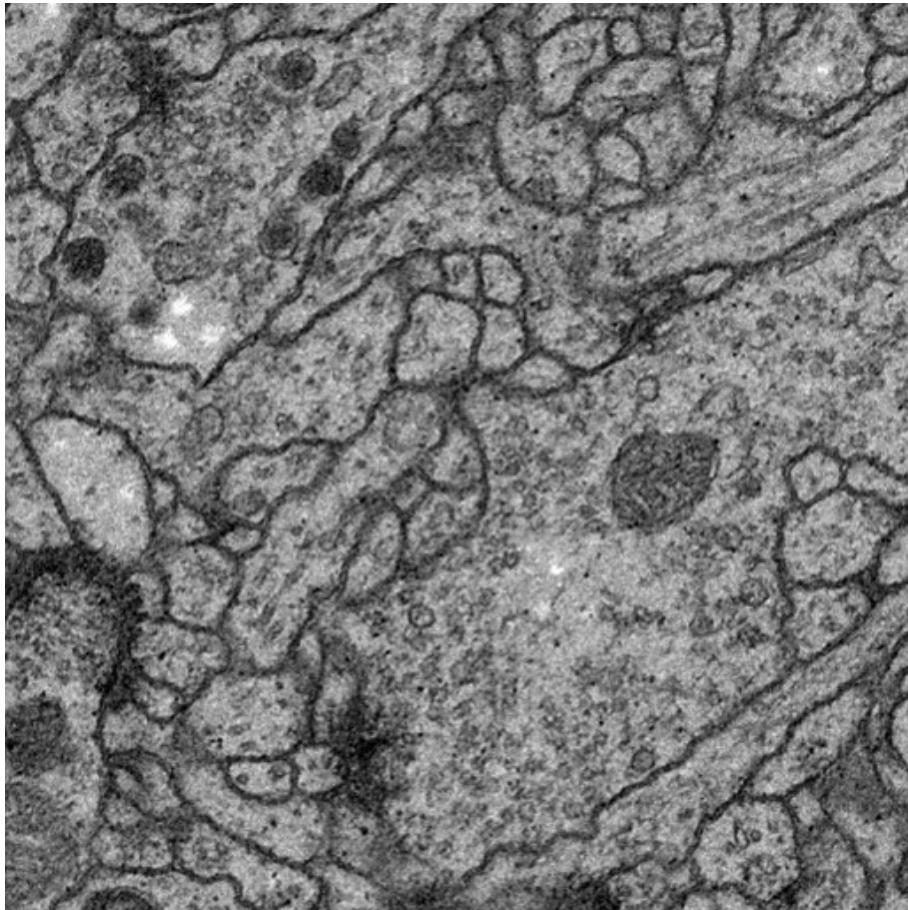




J. Long, E. Shelhamer, T. Darrell: "Fully Convolutional Networks for Semantic Segmentation".
CVPR 2015, arXiv:1411.4038 [cs.CV]

Replace fully connected layers by up-sampling steps

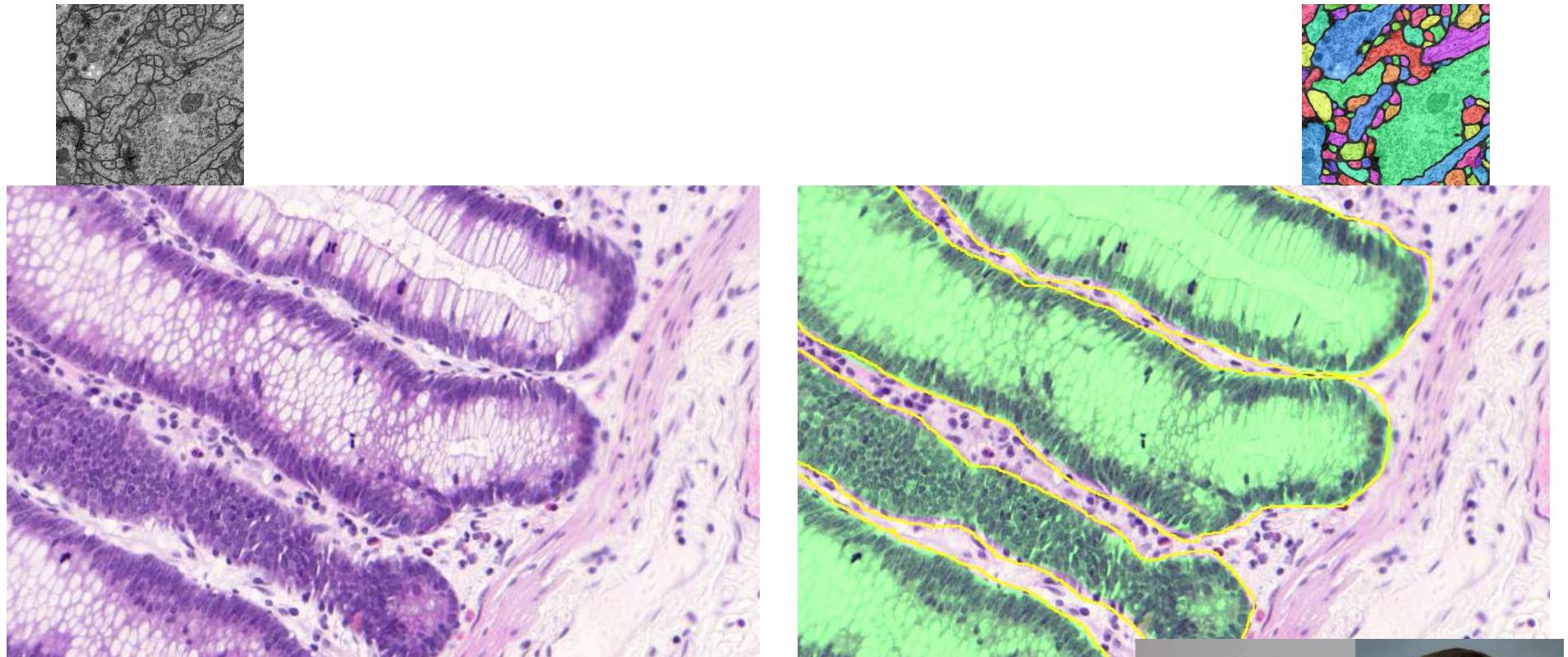
- **Outputs a (lower-resolution) image instead of a class label**
- **Uses a pre-trained classification network**
- **Only one feature channel per class at lowest resolution**
- **Uses zero-padded convolutions → no tiling possible**



Membrane Segmentation in electron microscopic (EM) images

- 2D/3D, two classes: Membrane / Rest
- Challenges: similar structures, weak contours, topology preservation

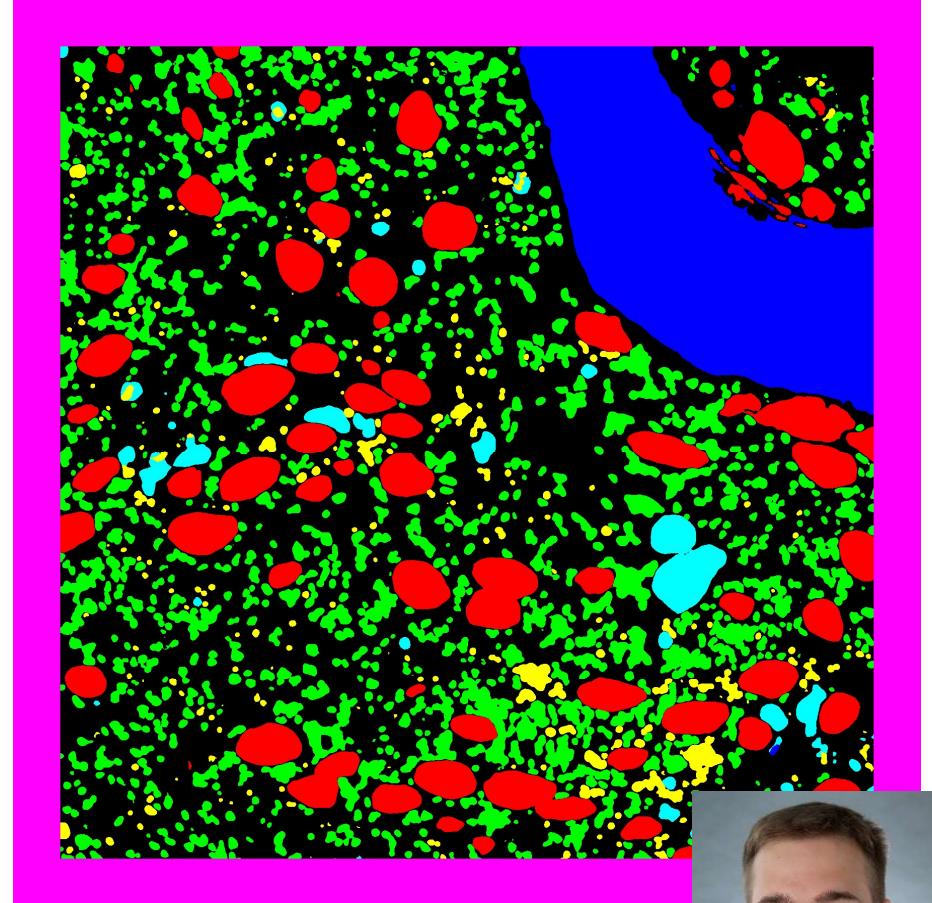
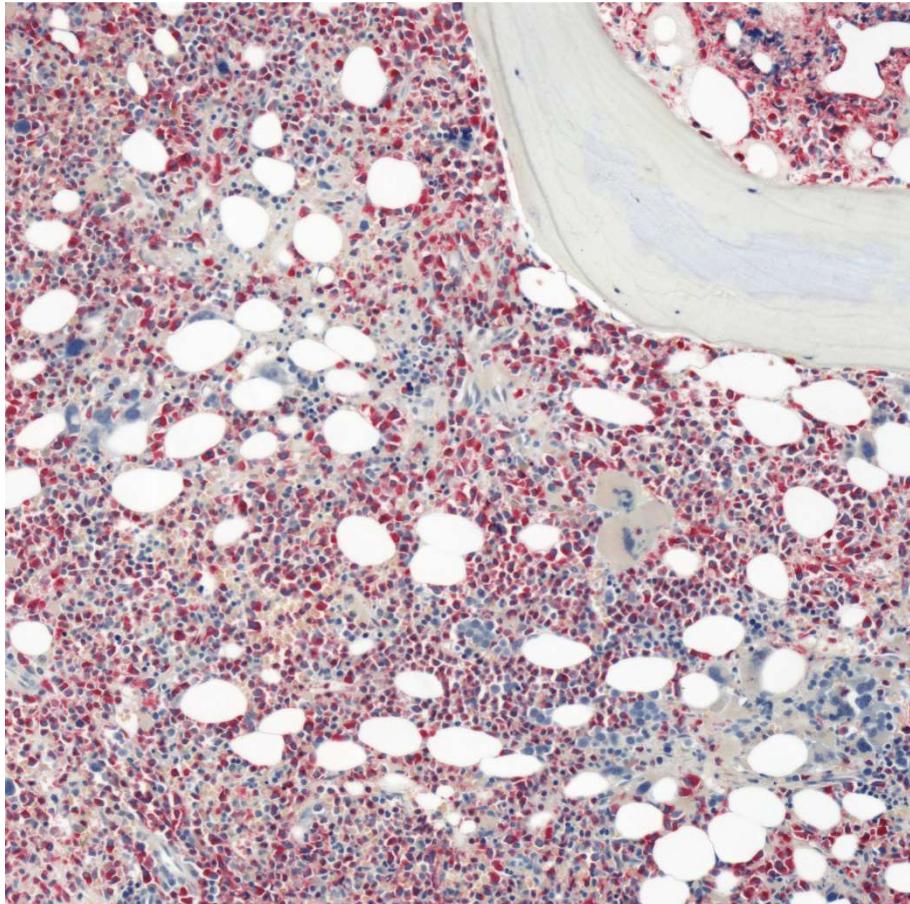
Olaf Ronneberger



Gland Segmentation in histologically stained tissue sections

- 2D, two classes: Gland / Rest
- Challenges: Similar textures, large structures

Olaf Ronneberger & Anton Böhm

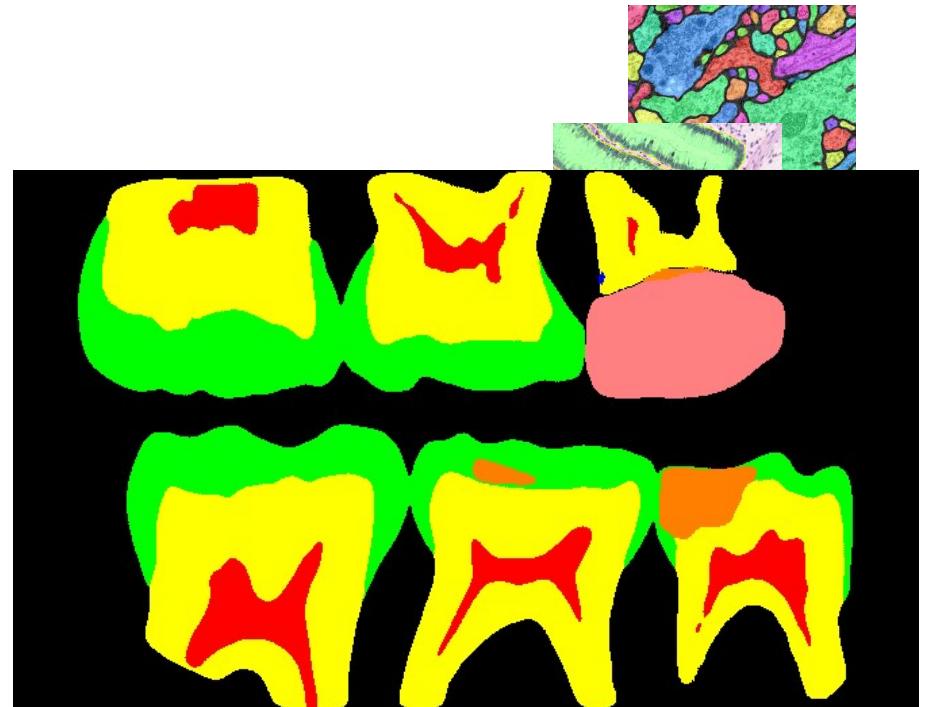
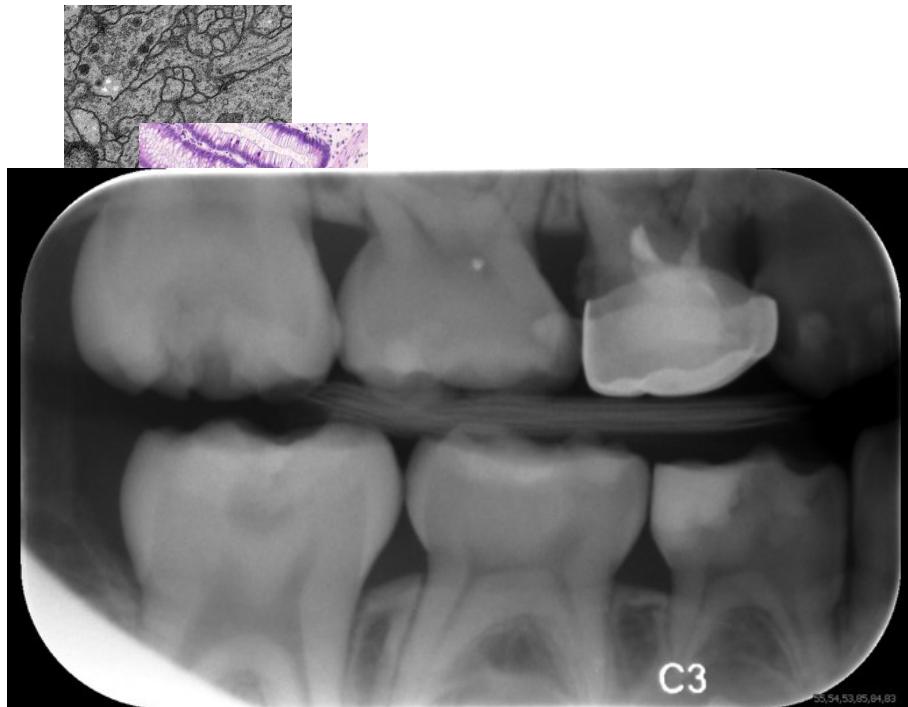


Segmentation and classification in histologically stained tissue sections

- 2D, 6 classes: Granulo-, Erythropoiesis, Megacaryocyte, Fat, Bone, BG
- Challenges: Similar textures, large structures, variable staining quality

Anton Böhm



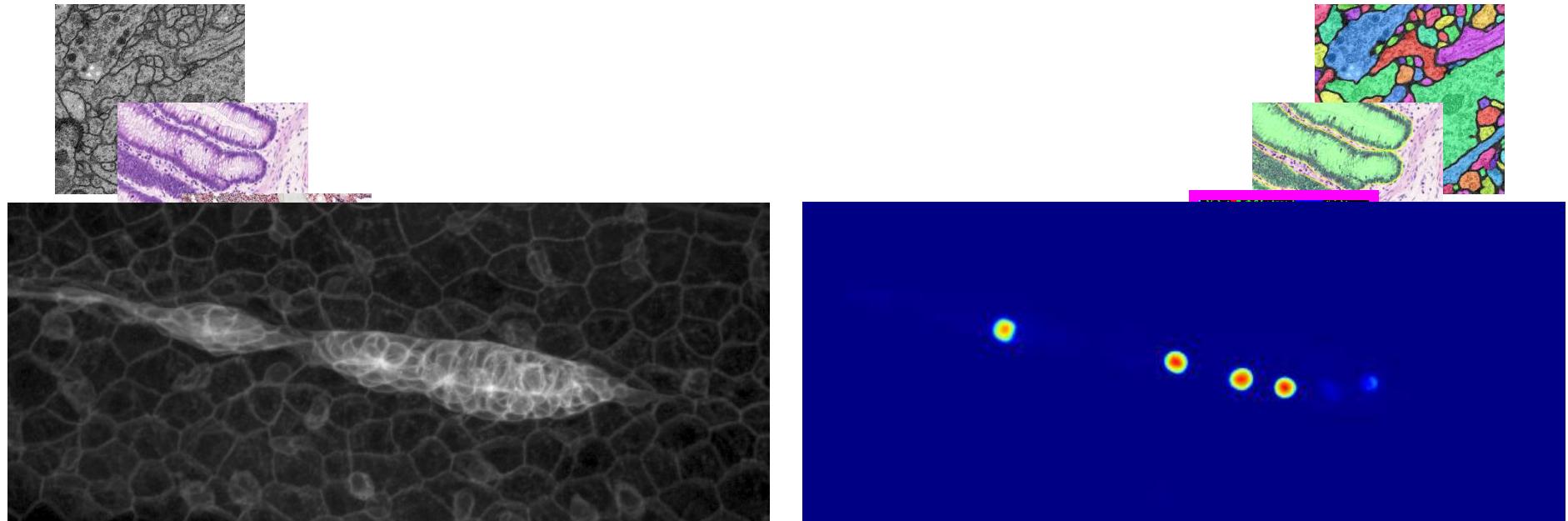


Segmentation and classification in dental x-ray images

- 2D, 8 classes: Dentin, Enamel, Pulp, Caries, ..., BG
- Challenges: Weak contrast, strong image quality variations

Olaf Ronneberger



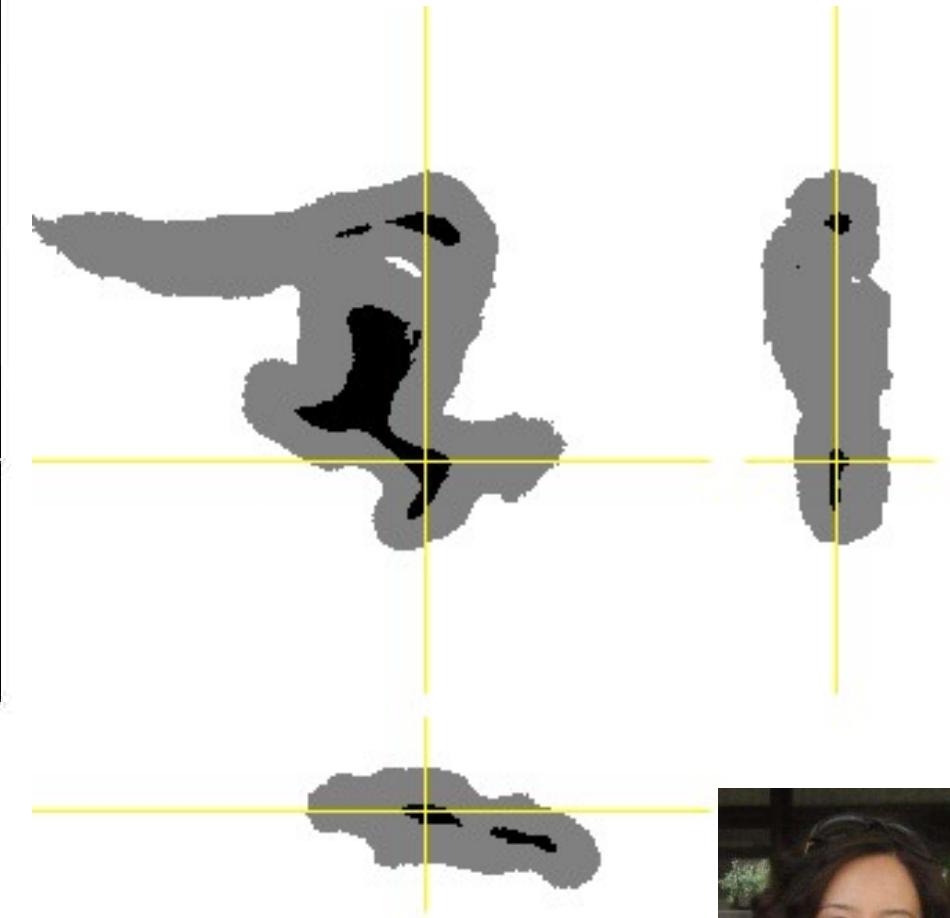
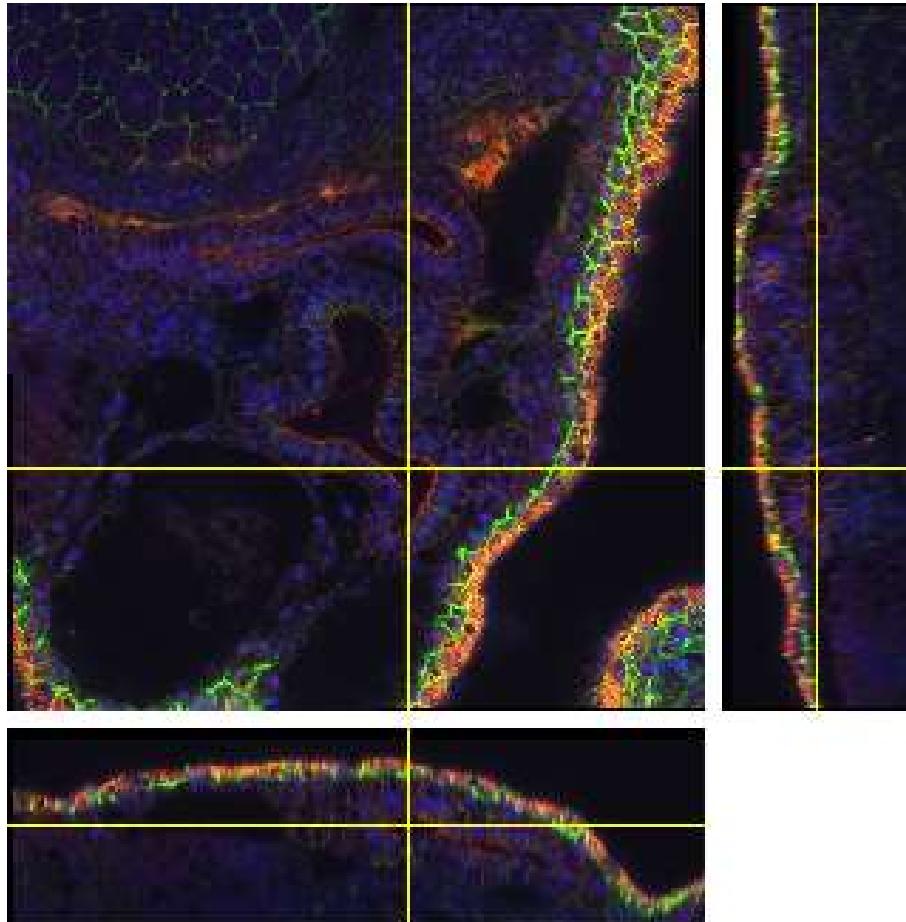


Rosette-Detection and tracking in Zebrafish primordia

- 2D+t, 2 classes: Primordium / Rest
- Challenges: Subtle cues in dense tissue

Robert Bensch



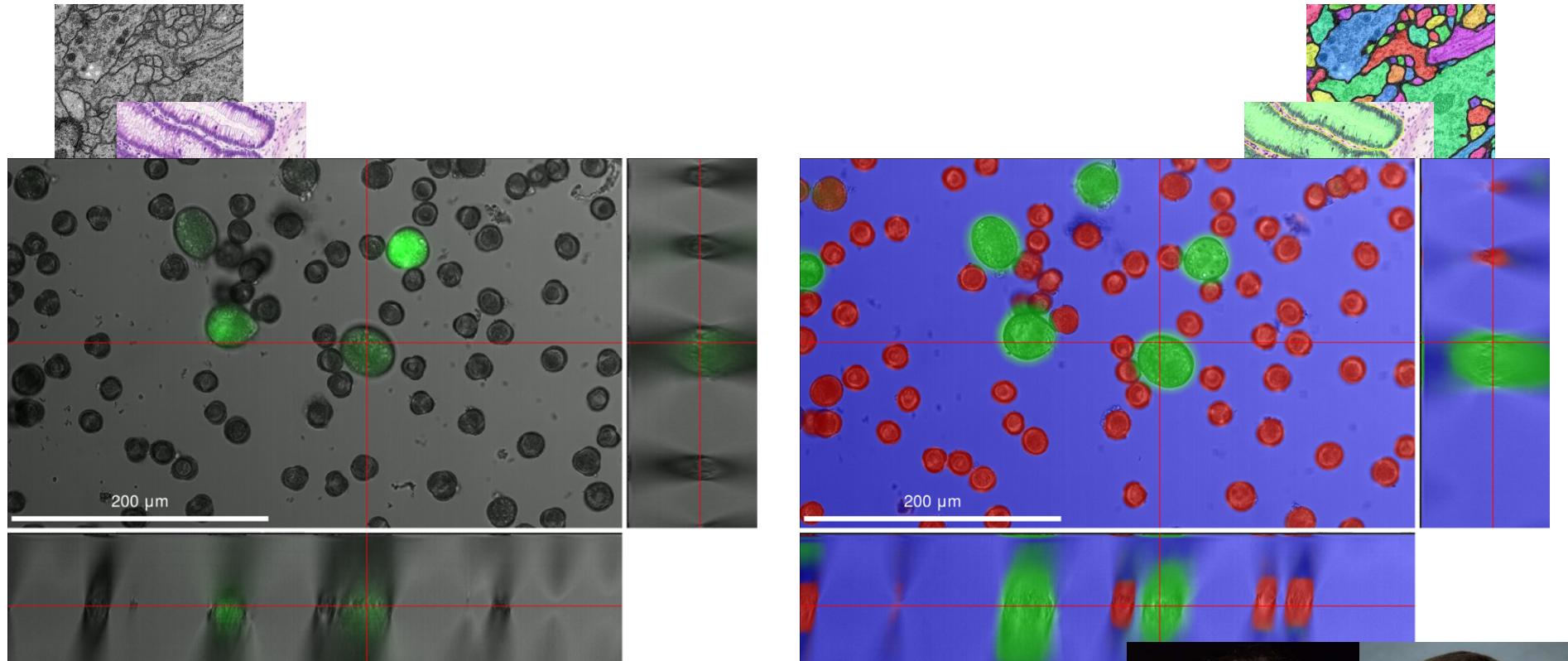


Renal tube segmentation in Xenopus

- 3D, 3 classes: Inner tube, Tube wall, Rest
- Challenges: Sparse Annotations, Structure similarities

Özgün Çiçek



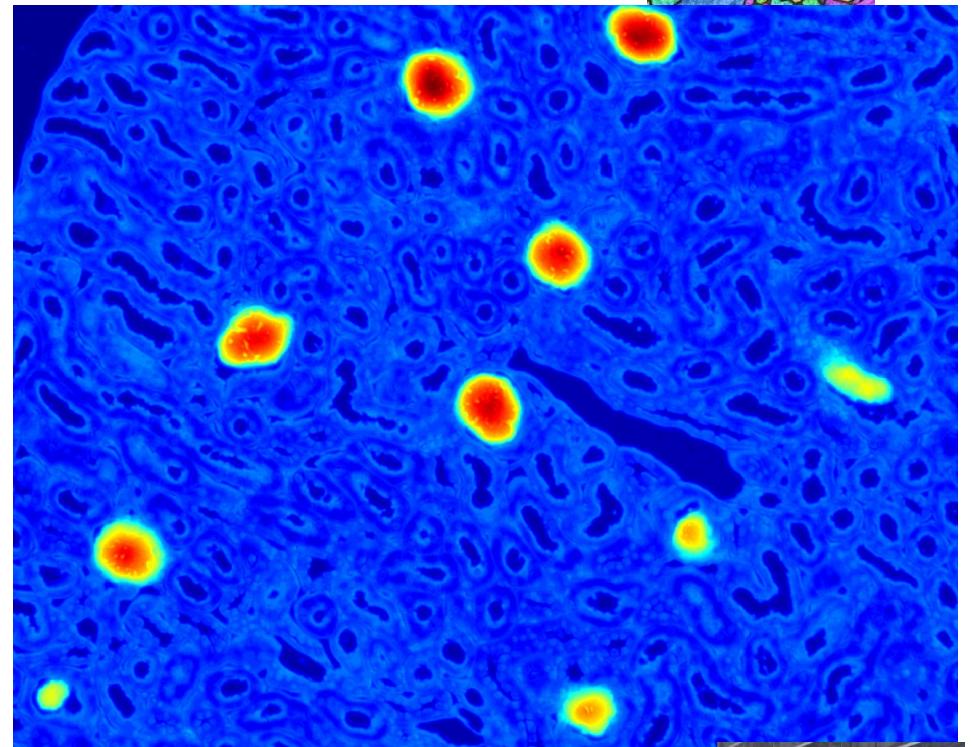
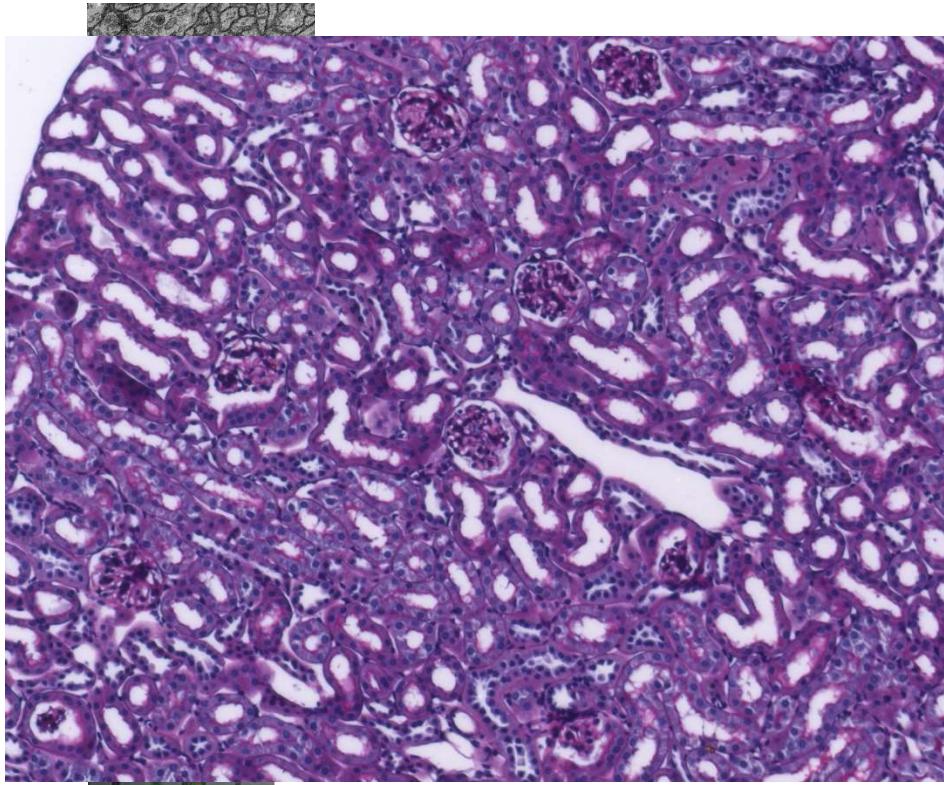


Microspore segmentation and classification (living/dead)

- 2.5D, 3 classes: Living, Dead, Background
- Challenges: Inaccurate automatic ground truth

Thorsten Falk & Anton Böhm

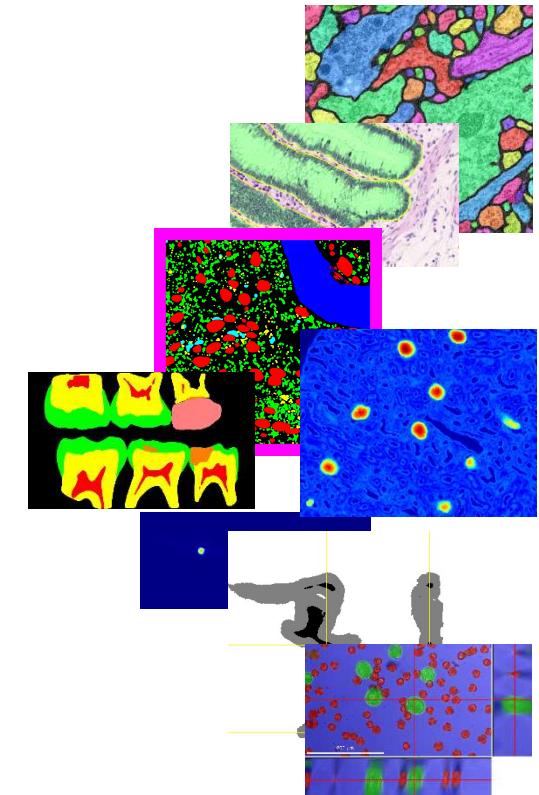
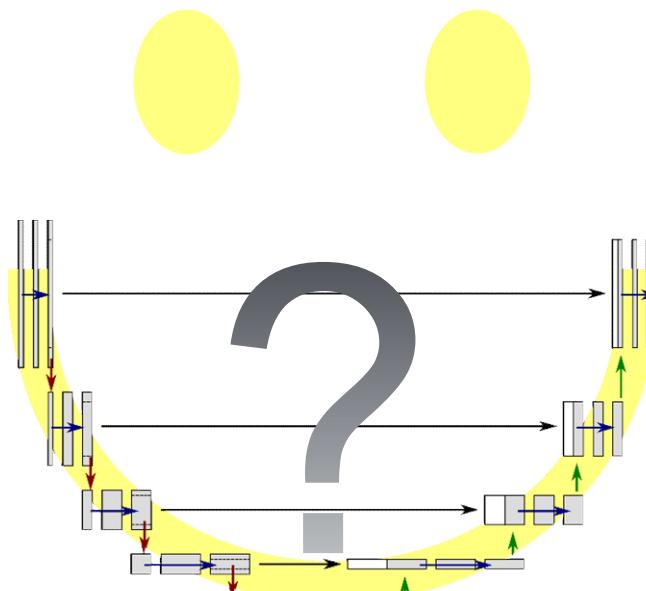
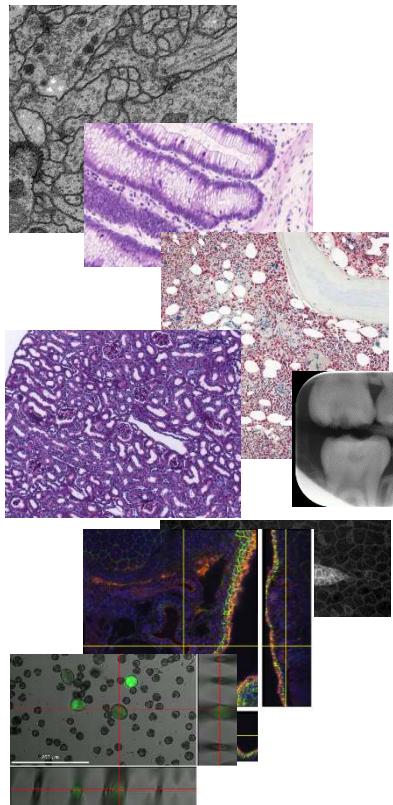




Glomeruli segmentation in histochemically stained renal tissue sections

- 2D, 2 classes: Glomerulus / Rest
- Challenges: Subtle texture differences, staining variations

Dominic Mai



Idea: Refine fully convolutional network of Long et al.

Pre-training on natural images is of little use

- **End-to-end training from little training data required!**

Low-dimensional feature representation sacrifices resolution

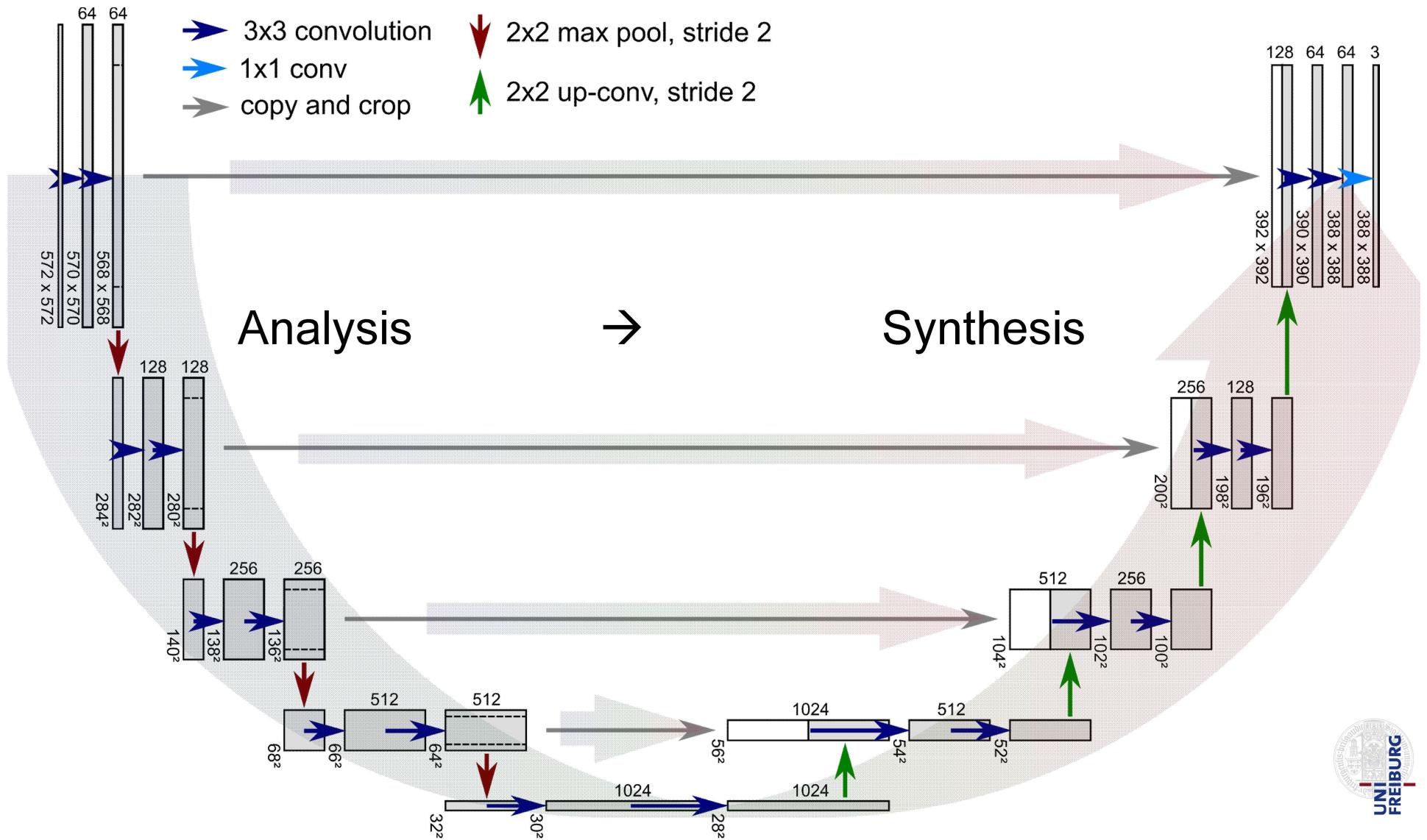
- Retain **high-dimensional features** for „smart image synthesis“
- Also use high-resolution features for synthesis (**shortcut connections**)

Padded convolutions restrict network to fixed input image size

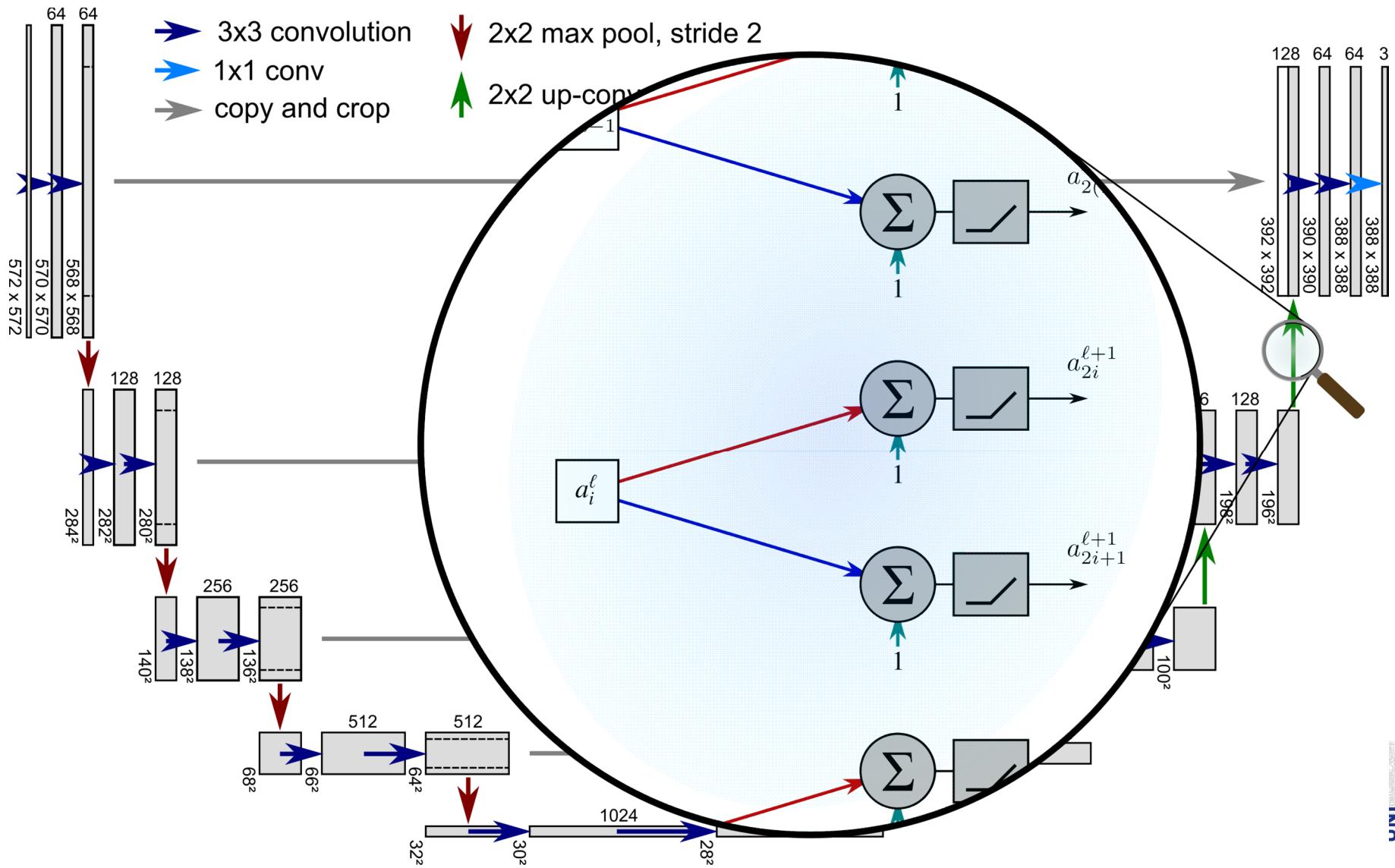
- Use only **valid part of convolutions** to allow **segmentation of arbitrarily sized images**

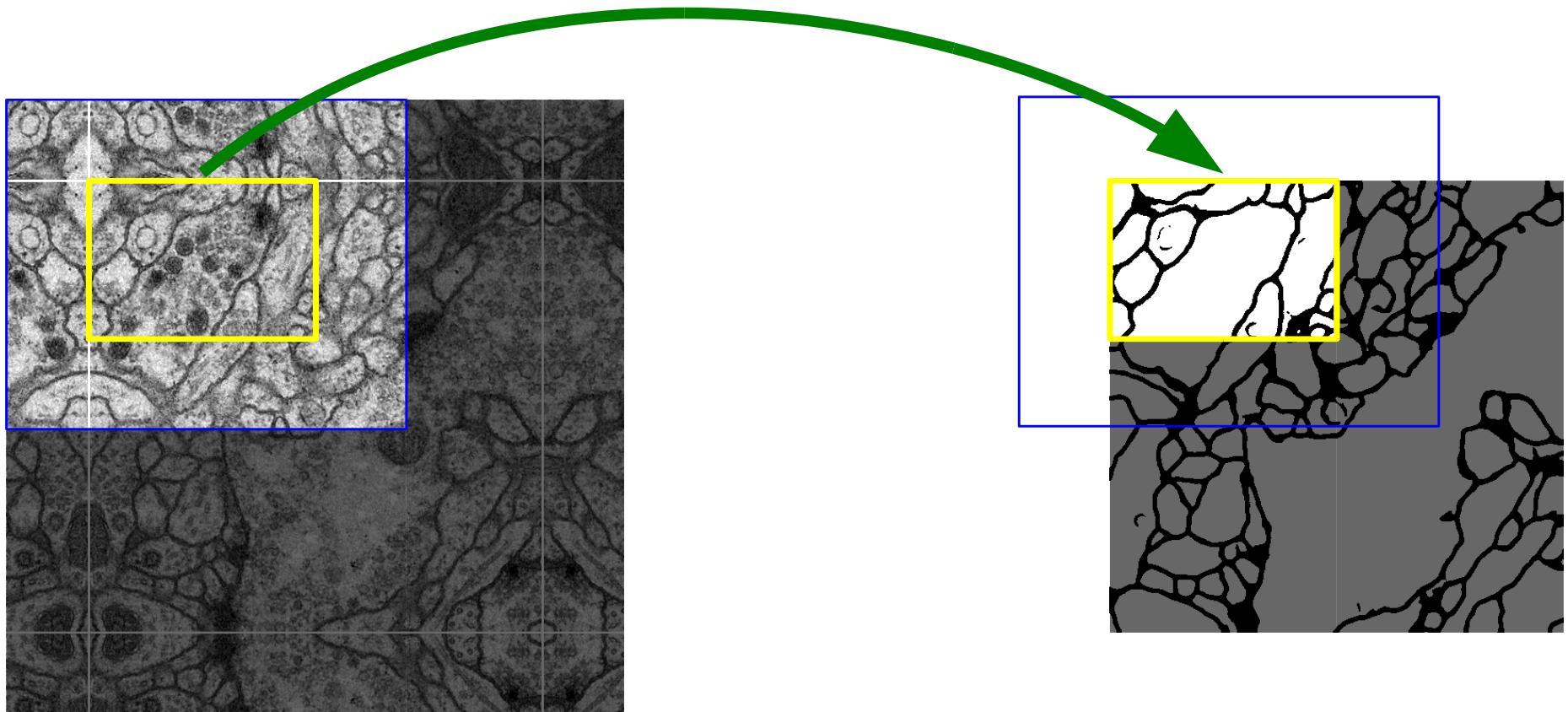


CNN for Image Analysis and Synthesis: The U-Net



Up-Convolutions to Recover Original Resolution





- Segmentation of the yellow area needs input data of the blue area
- Raw data extrapolation by mirroring

Generation of Training Data



Many images with manual annotation required!?

- Depends on complexity of the problem and the network
- For simple biomedical applications few images may be sufficient

Biomedical data

- ✓ Scales are known
- ✓ No projective distortions
- ✓ No (self-)occlusion
- ✓ Position and orientation of structures often irrelevant
- ✗ High variability
 - Biological variations
 - Sample preparation
 - Imaging
- ✗ Subtle differences relevant
- ✗ Out-of-focus regions
- ✗ Ambiguous annotation



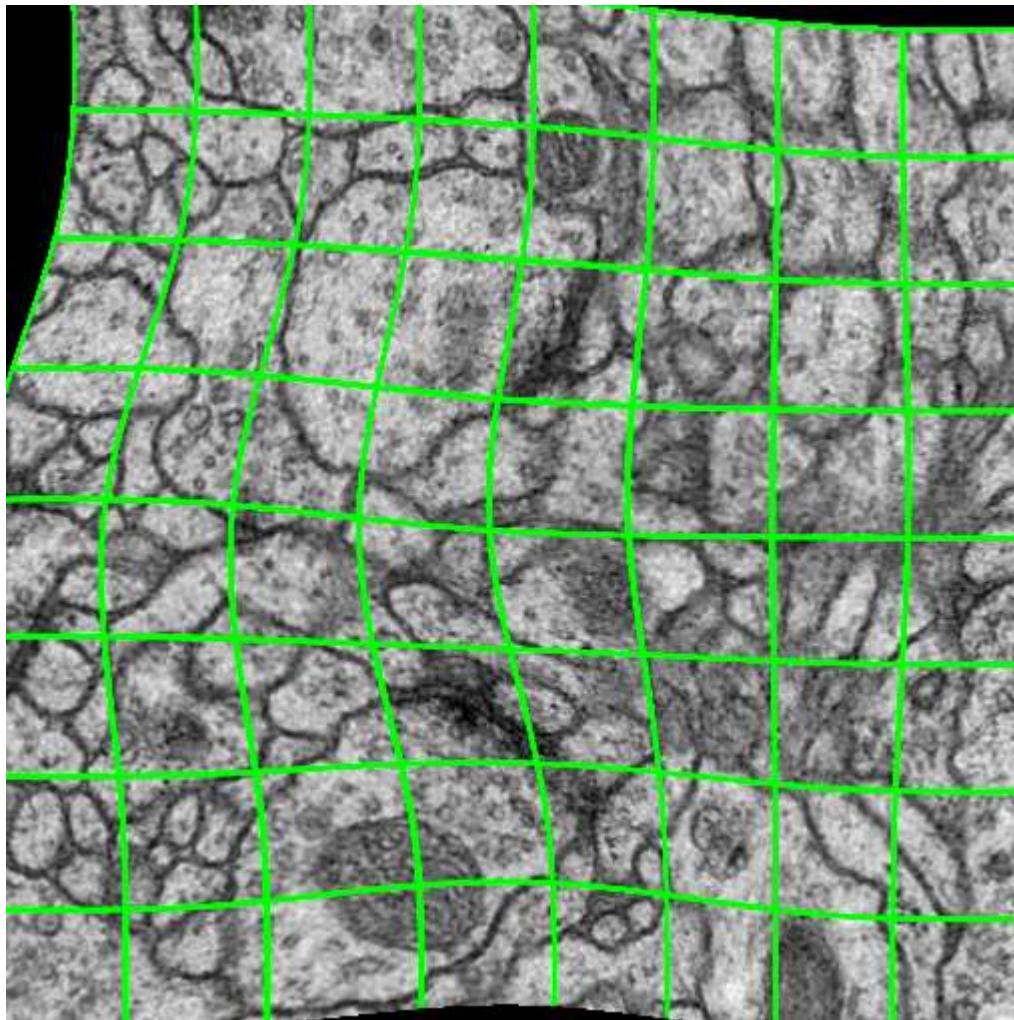
Many transformations do not alter biological meaning:

- **Shift and Rotation** irrelevant
 - Shift and rotate original images and annotations
- **Smooth shape variation** often irrelevant
 - Apply smooth deformations to original images and annotations
- **Absolute brightness and contrast** irrelevant
 - Apply different intensity transformations to original images
- **Noise** irrelevant
 - Add different noise levels to original images
- **Additional network regularization** possible
 - Dropout, Weight-Decay, ...

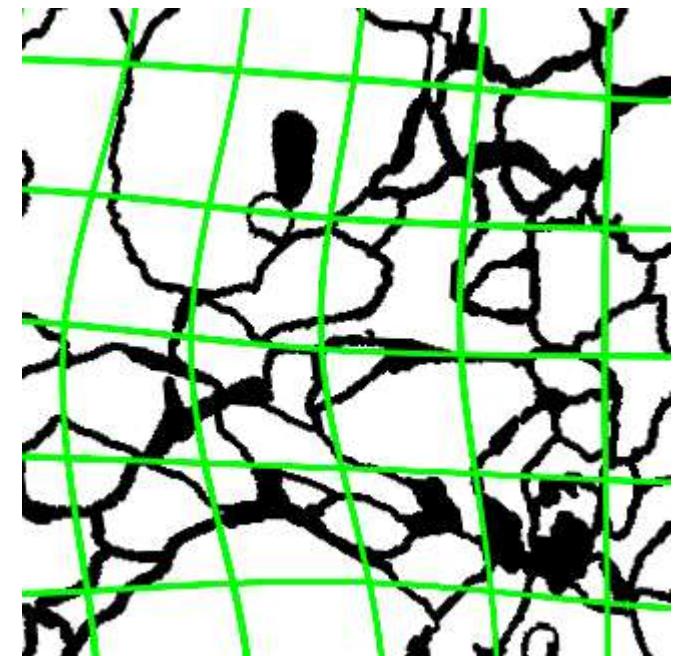
Few annotations can yield large training set!



Virtual Training Samples via Data Augmentation

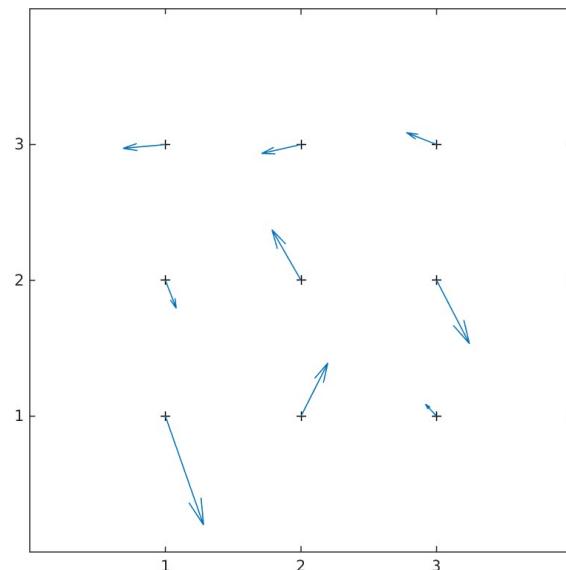


Randomly deformed image
(for visualization: no rotation, no shift, no extrapolation)

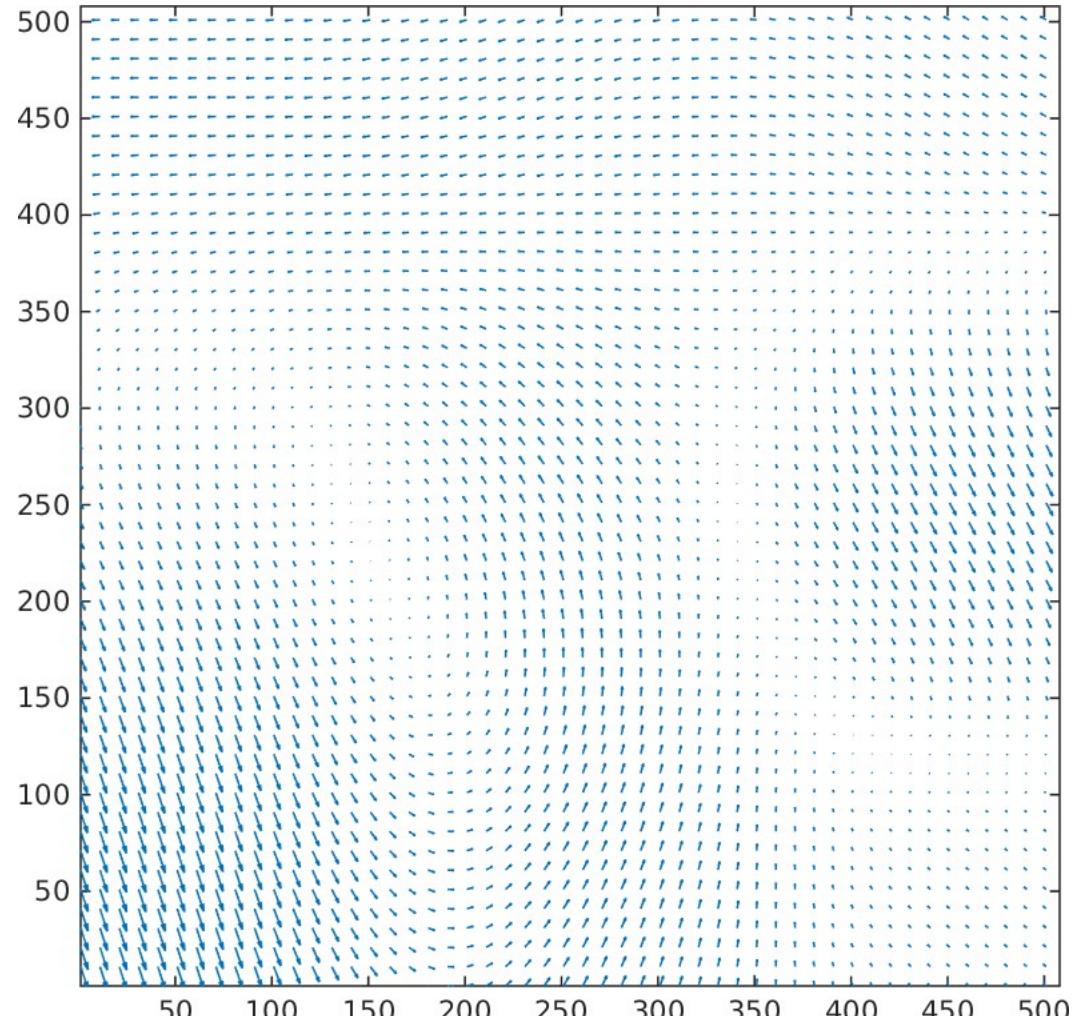


correspondingly deformed
manual labels

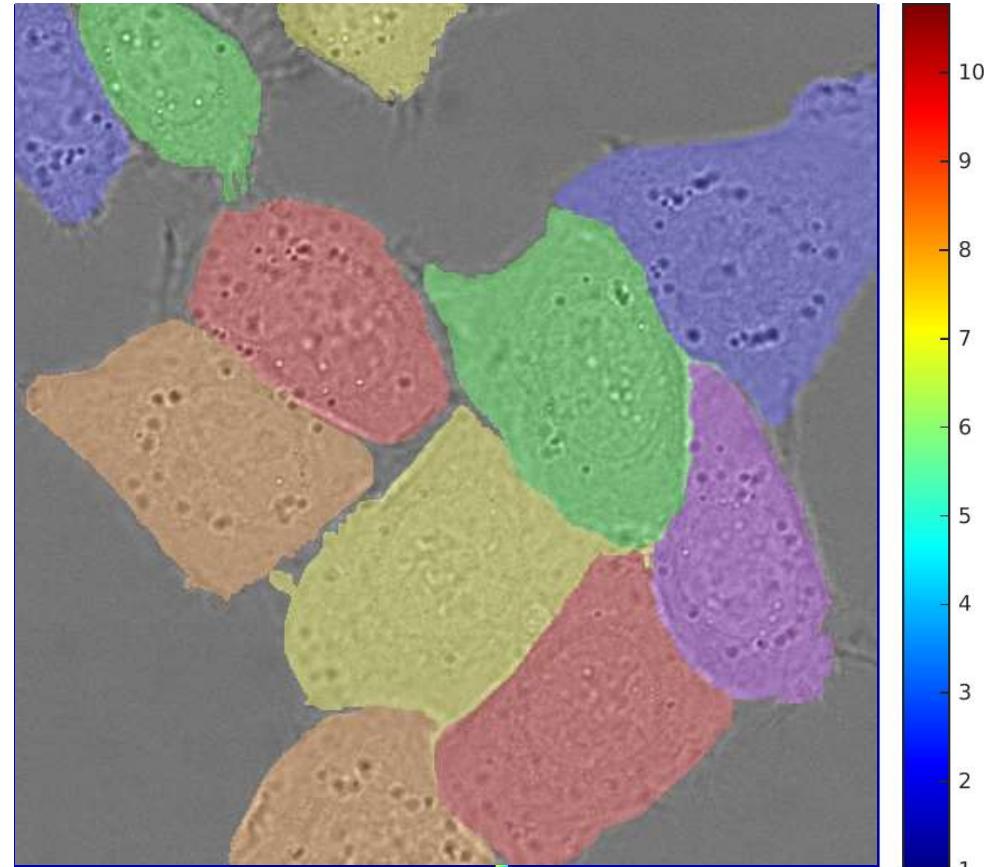
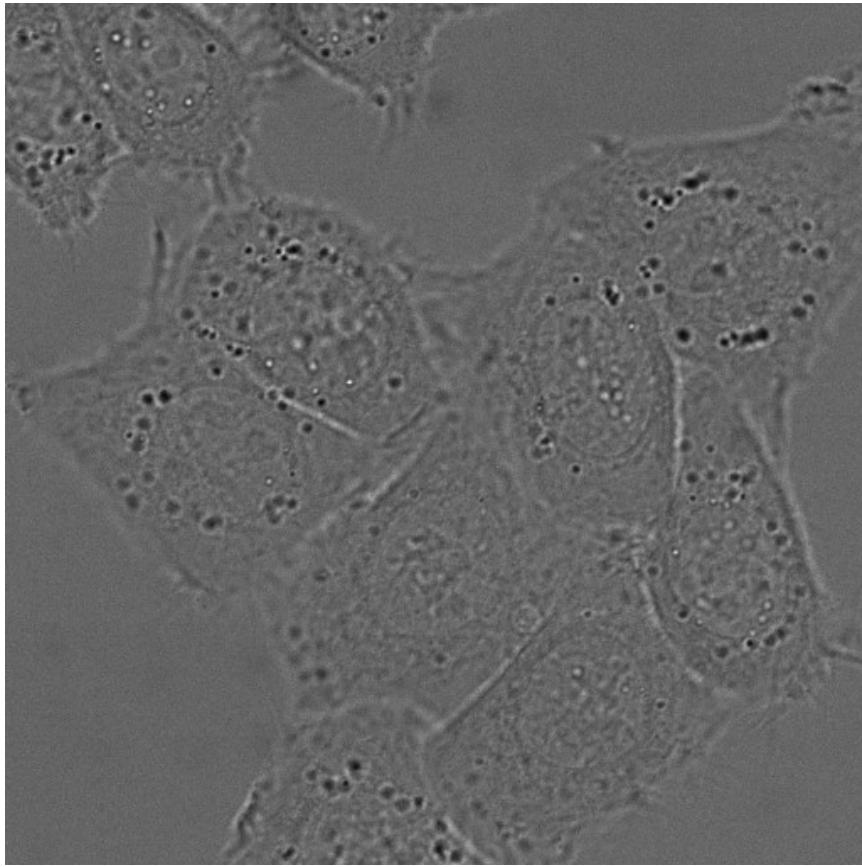
Deformation Field Generation



3x3 random deformation vectors



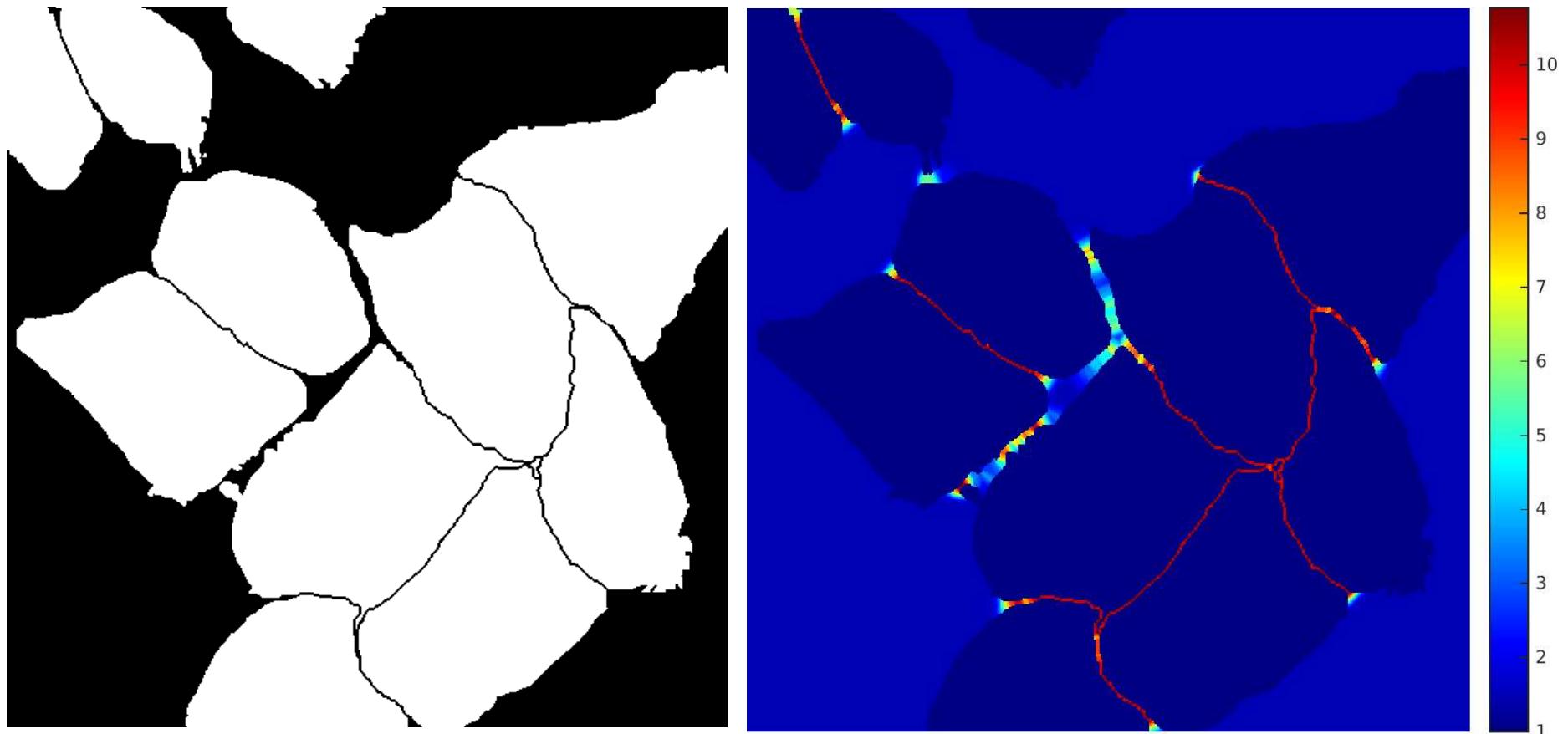
Full deformation field by **bicubic interpolation**



Semantic segmentation vs. Instance segmentation

- Not only cell/no cell, but one instance label per cell
- If instances touch, semantic segmentation merges them

Separation of Touching Instances



Ad-hoc idea:

- Add one pixel-wide background region between instances
- Give these interfaces high weight during training

Deep networks are extremely flexible

- Learn full image analysis pipeline from examples
- Allow classification, detection, segmentation, tracking, ...
- Many more applications:
 - speech recognition
 - Natural language processing and automatic translation

Data augmentation allows training with few annotated images

Network training between hours and weeks

- Infeasible without parallel GPU hardware

Network evaluation is extremely fast

- Full image segmentation within seconds on a GPU

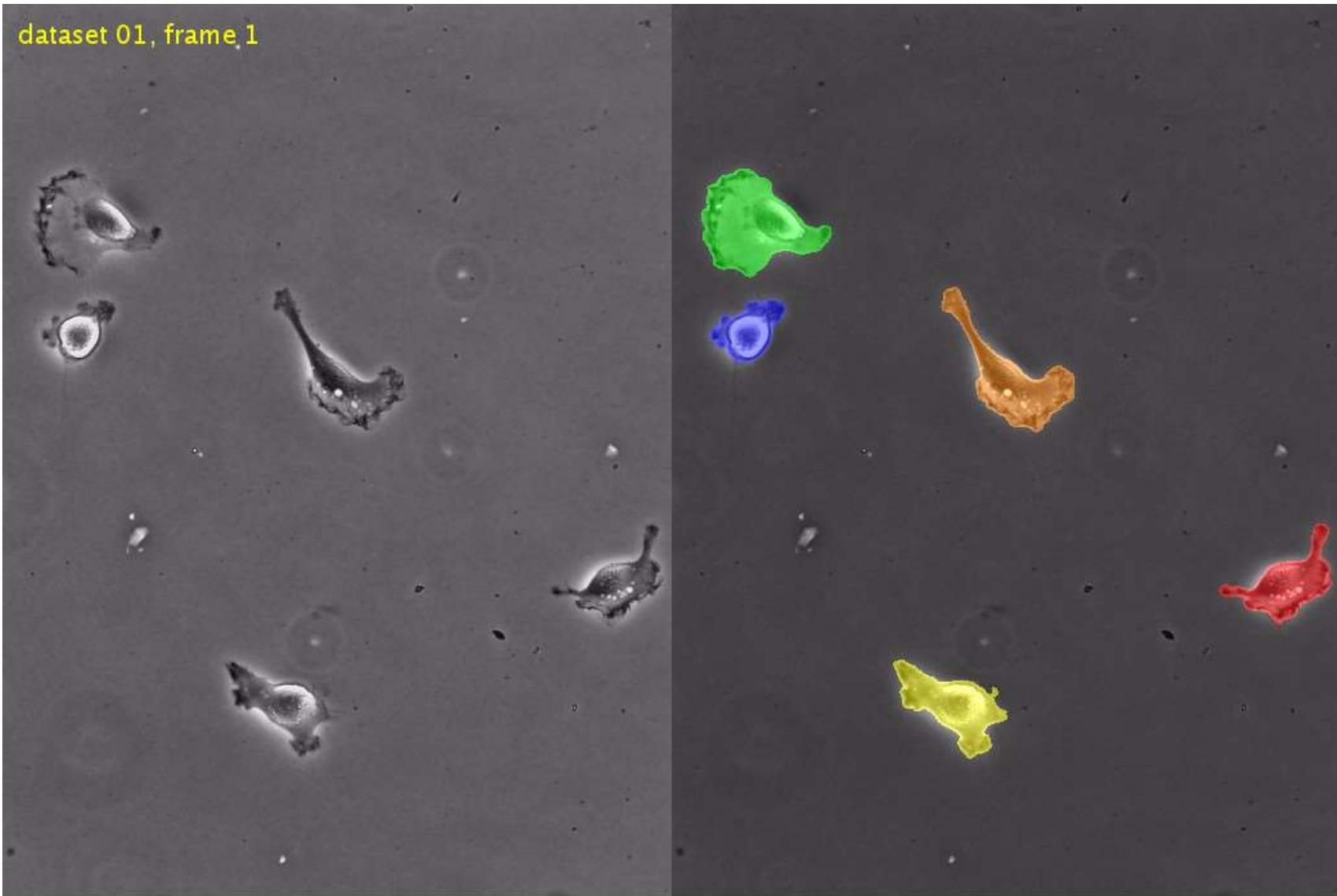


- Get familiar with the caffe neural network framework
<http://caffe.berkeleyvision.org>
- Train a neural network for handwritten digit recognition (MNIST):
<http://caffe.berkeleyvision.org/gathered/examples/mnist.html>
This can be done on CPU or GPU
- If you don't want to compile caffe yourself (quite a lot of third party libraries), we provide a ready compiled version (Linux, 64bit) on the course webpage
- Optional: Do some cool stuff with the ready trained networks from the „model zoo“ on the caffe homepage. Here you might really need a GPU. The pool computers are equipped with an nVidia GTX1060.

If you want to learn more, subscribe for our lab course on deep learning and/or our Seminar/Blockseminar!



Extensions: Cell Tracking over Time



U-net: Convolutional Networks for Biomedical Image Segmentation

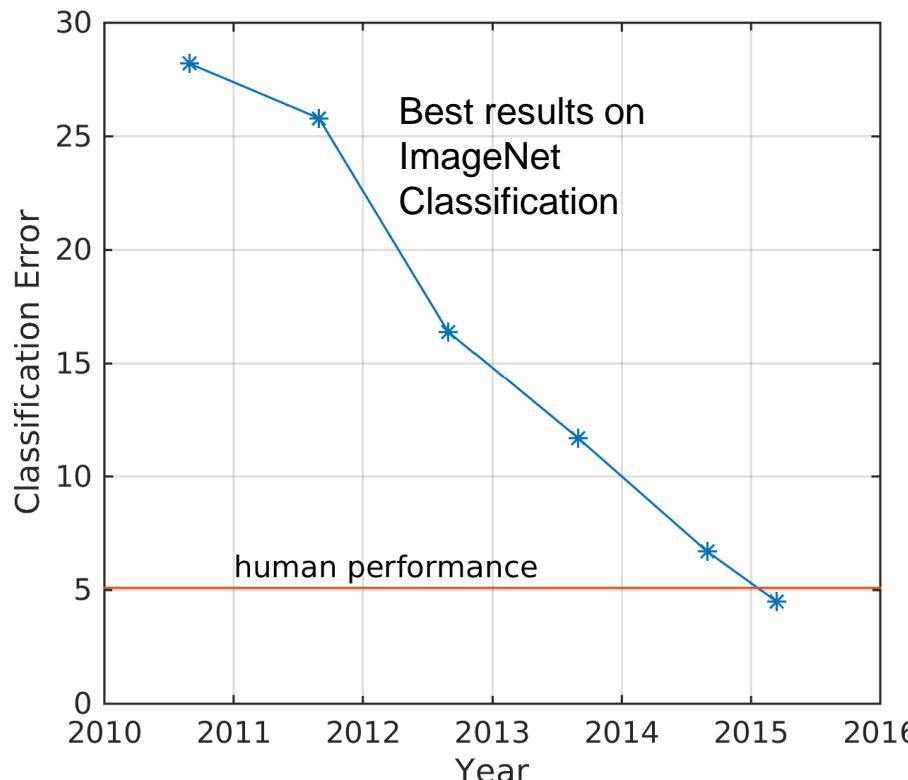
Olaf Ronneberger

Computer Science Department and
BIOSS Centre for Biological Signalling Studies
University of Freiburg, Germany

O. Ronneberger, P. Fischer, T. Brox: U-net: Deep Convolutional Networks for Biomedical Image Segmentation.
MICCAI 2015, Springer, LNCS, Vol.9351: 234--241, 2015

Image Classification State of the Art

- Deep convolutional neural networks surpassed human-level performance in Feb 2015



GT: mountain tent
1: sleeping bag
2: mountain tent
3: parachute
4: ski
5: flagpole

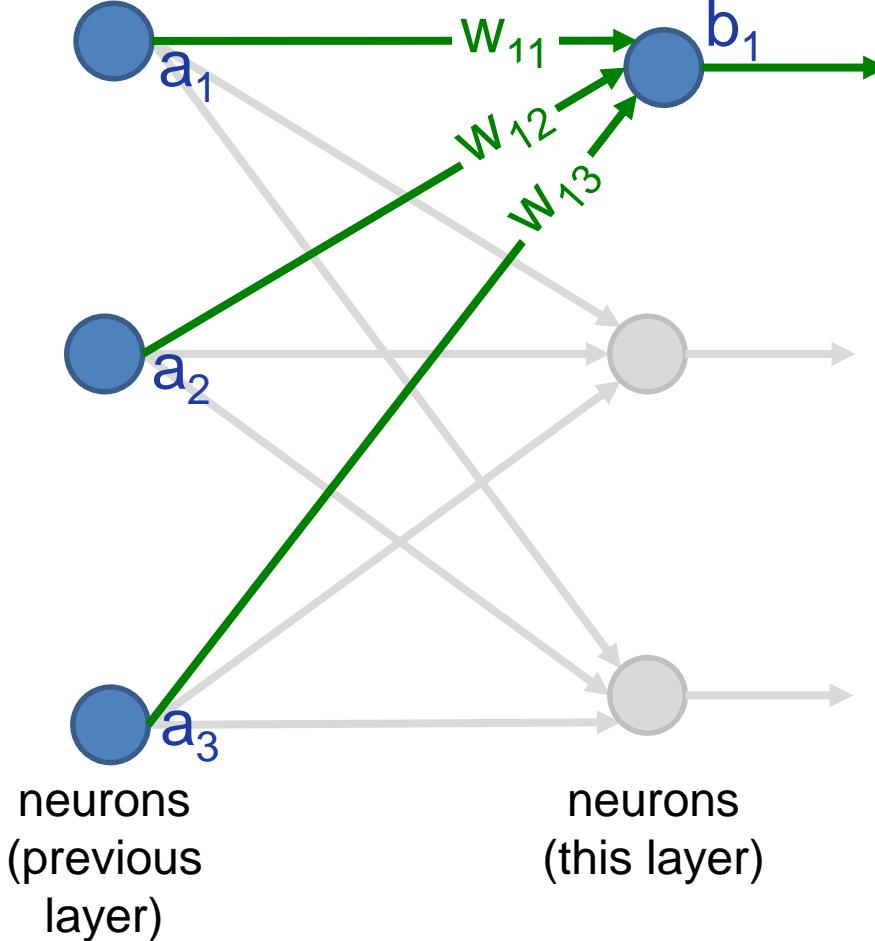


GT: geyser
1: geyser
2: volcano
3: sandbar
4: breakwater
5: leatherback turtle

K. He, X. Zhang, S. Ren, J. Sun: "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." (2015) arXiv:1502.01852 [cs.CV]

Neural Networks

- Neural networks are a very old concept. Idea: simulate a brain in the computer



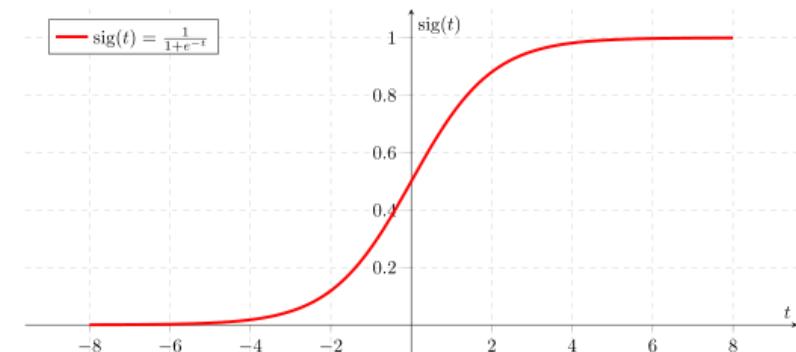
$$b_1 = f(w_{11}a_1 + w_{12}a_2 + w_{13}a_3)$$

$a_1 - a_3$: output of neurons in previous layer

$w_{11} - w_{13}$: weights

b_1 : output of neuron in this layer

$f(x)$: activation function



Neural Networks

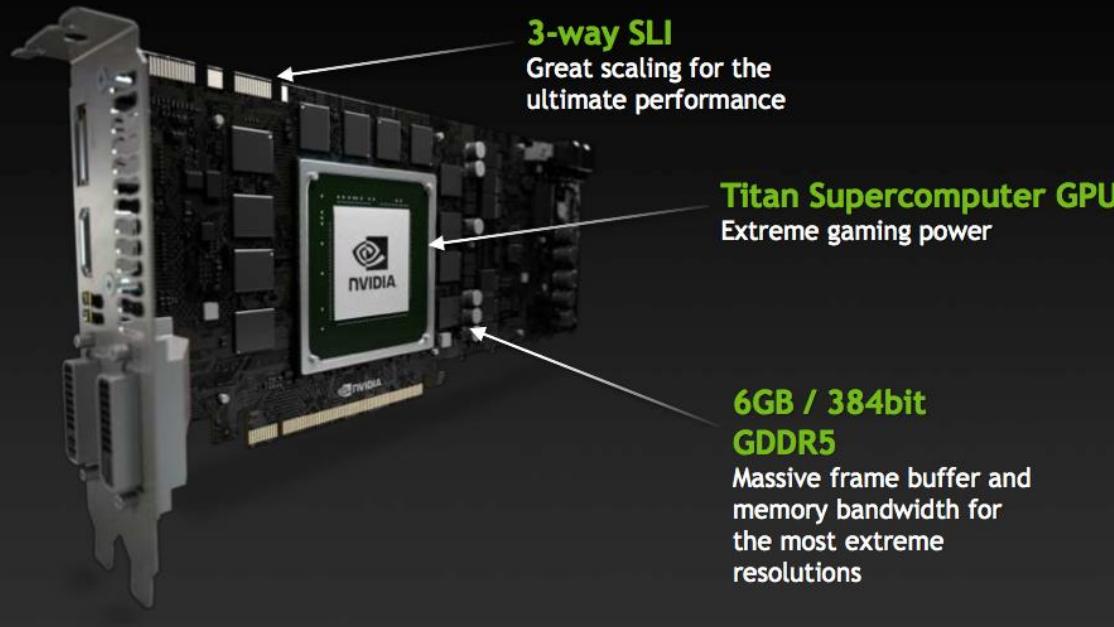
- Several hypotheses
- Only successful on very specific problems: e.g. “handwritten digits recognition”
- **Why do they work now?**

A grid of handwritten digits from 0 to 9, arranged in 10 rows and 10 columns. The digits are written in a cursive style and are slightly overlapping. The first row contains zeros, the second row contains ones, the third row contains twos, the fourth row contains threes, the fifth row contains fours, the sixth row contains fives, the seventh row contains sixes, the eighth row contains sevens, the ninth row contains eights, and the tenth row contains nines.

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

Availability of Enormous Computing Power

Customized for High Performance Gaming



- Thanks to all people who support the development of these supercomputers!

Availability of Millions of Annotated Images

ILSVRC



flamingo



cock



ruffed grouse



quail



partridge

...



Egyptian cat



Persian cat



Siamese cat

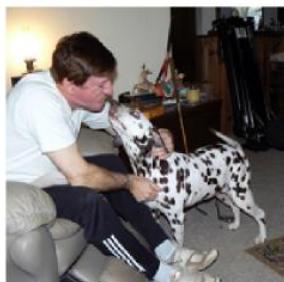


tabby



lynx

...



dalmatian



keeshond



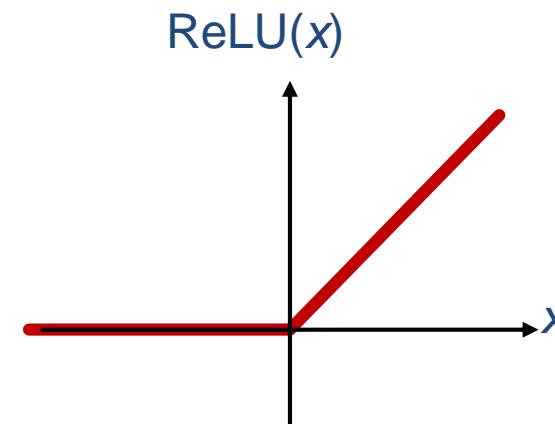
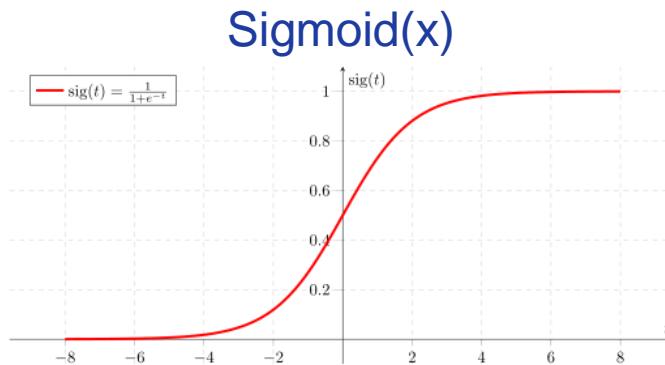
miniature schnauzer standard schnauzer giant schnauzer



...

[O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." (2014) arXiv:1409.0575 [cs.CV]

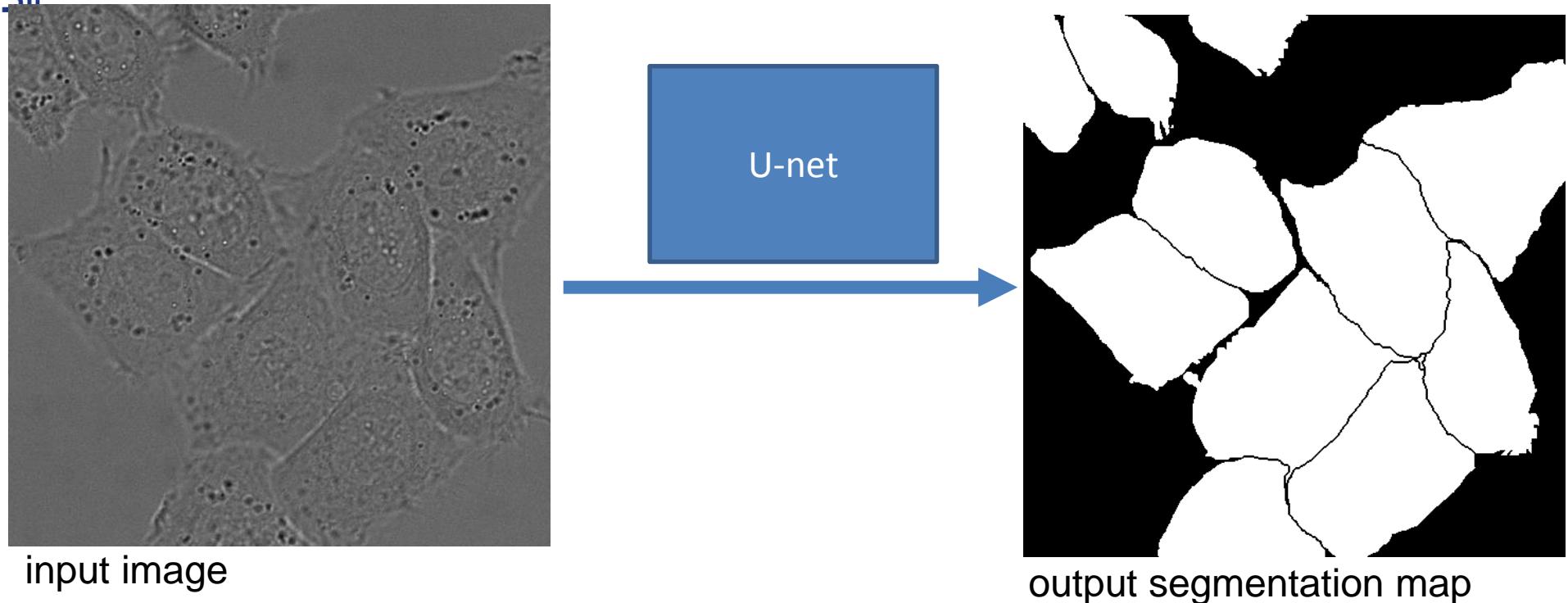
Slight Improvements in the Architecture



- Non-saturating neurons: no “vanishing gradients”
- Drop-Out Layers: less overfitting
- Efficient GPU implementations
- Very deep networks (millions of neurons)

- It is definitely time to apply these networks to something really useful

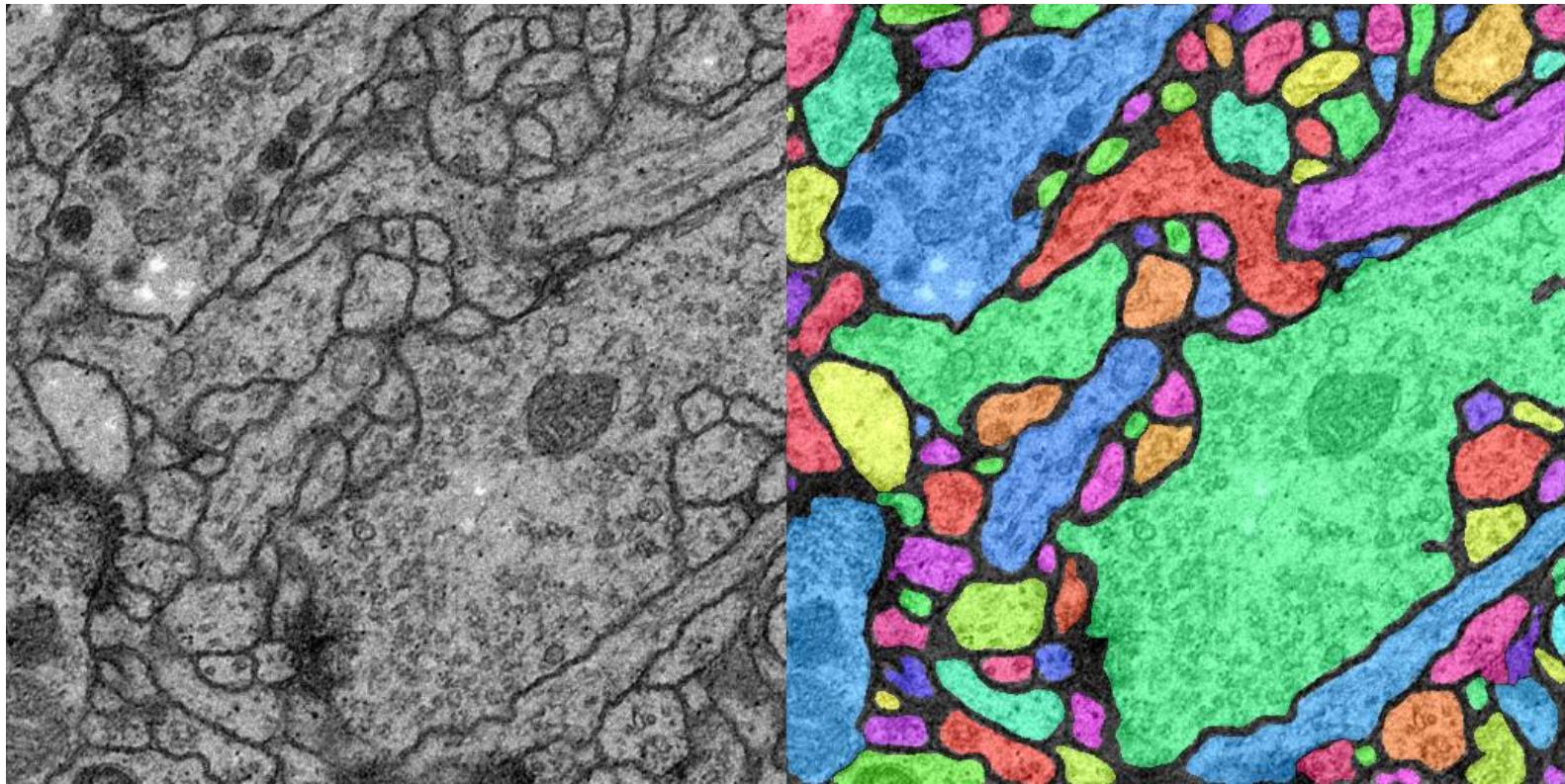
Biomedical Image Segmentation with U-net



- U-net **learns segmentation** in an end-to-end setting
- **Very few annotated images** (approx. 30 per application)
- **Touching objects** of the same class

Data provided by Dr. Gert van Cappellen, Erasmus Medical Center, Rotterdam, The Netherlands

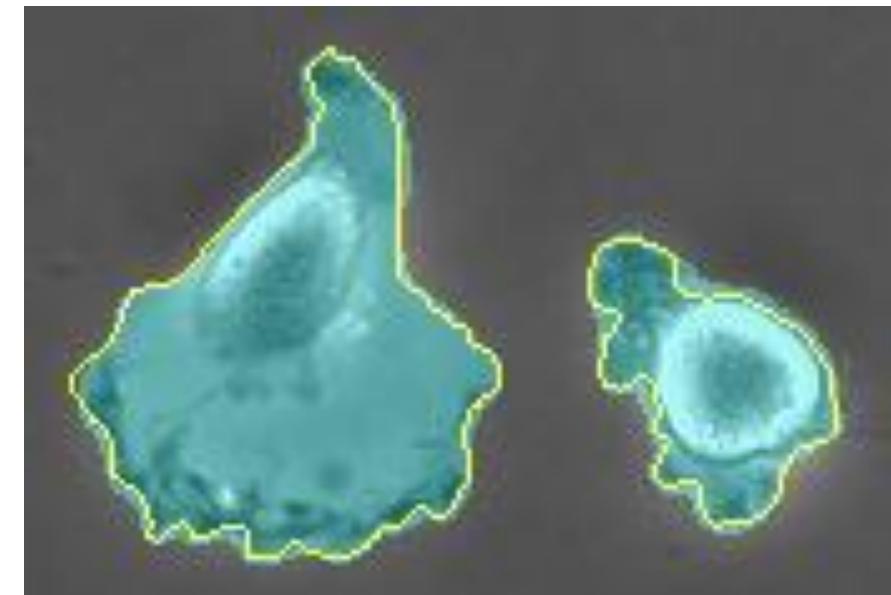
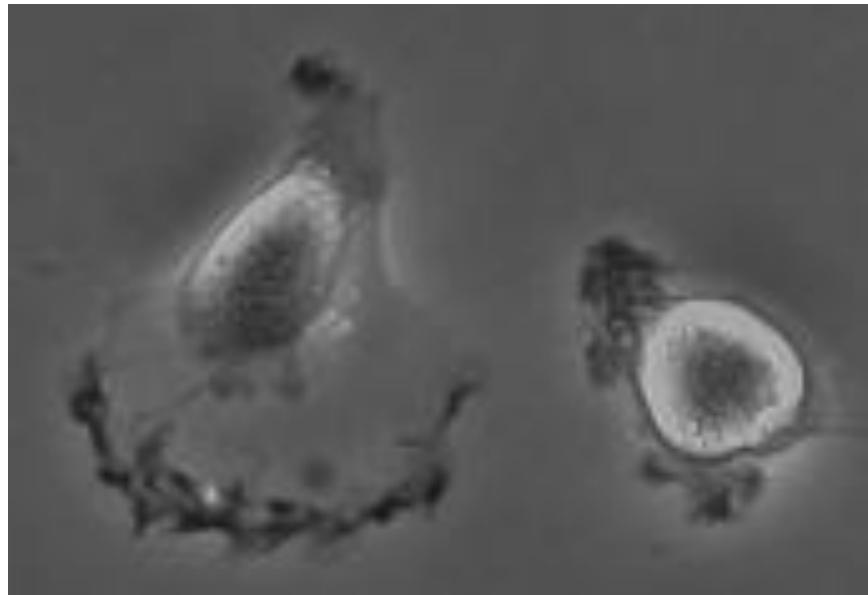
Neuronal Structures in EM



- EM segmentation challenge (ongoing since ISBI 2012):
 - Our result: 0.000353 warping error
(New best score at submission march 6th, 2015)
 - Sliding-window CNN: 0.000420

Data provided by the ISBI 2012 EM segmentation challenge.
Cardona, A. et al.: PLoS Biol 8(10), e1000502 (2010)

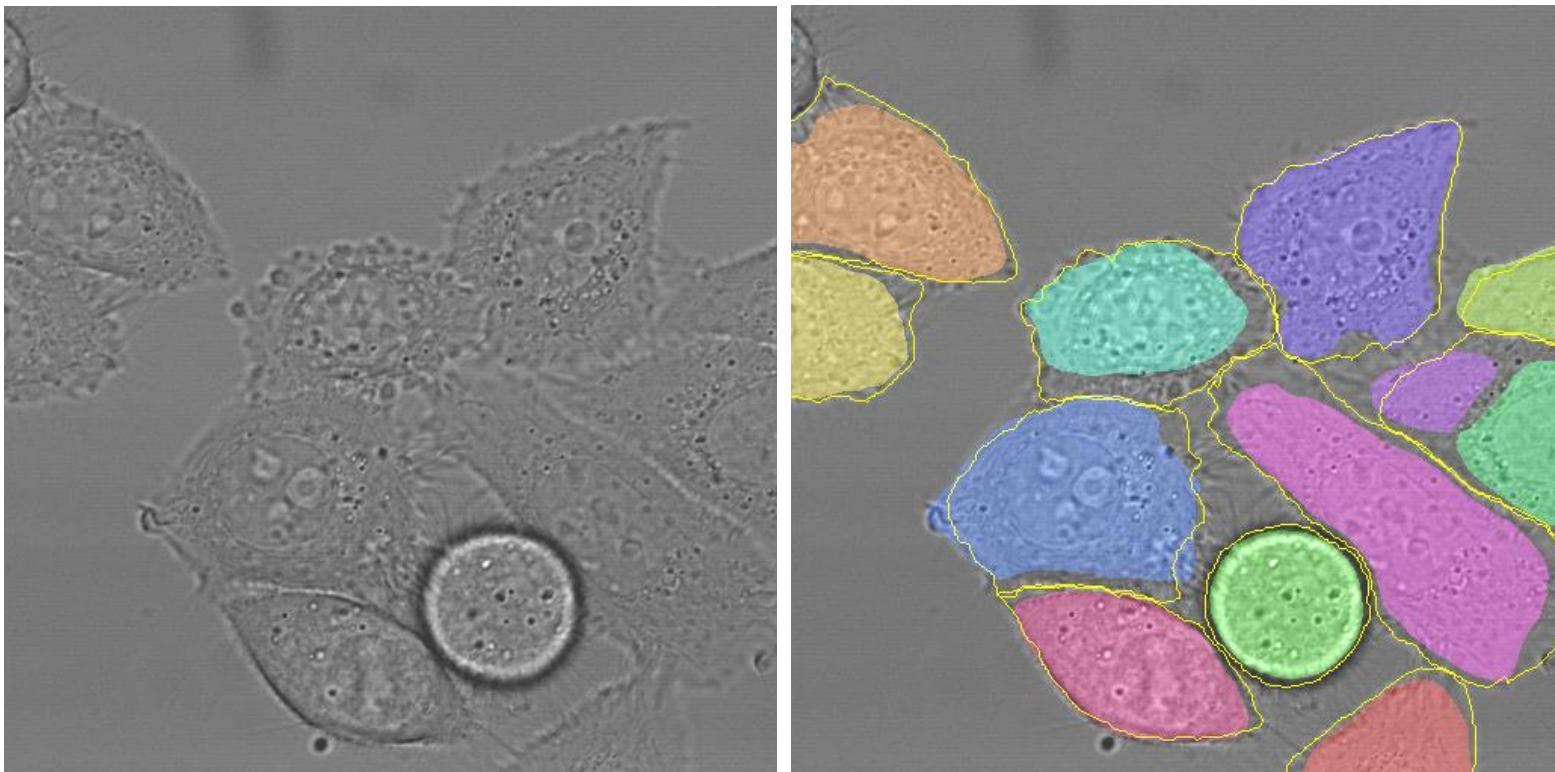
Cells in Phase-Contrast Microscopy



- ISBI Cell Tracking Challenge 2015
Dataset „PhC-U373“
 - Our Result: 92% Intersection over Union, **Winner**
 - Second best result: 83%

Data provided by Dr. Sanjay Kumar. Department of Bioengineering
University of California at Berkeley. Berkeley CA (USA)

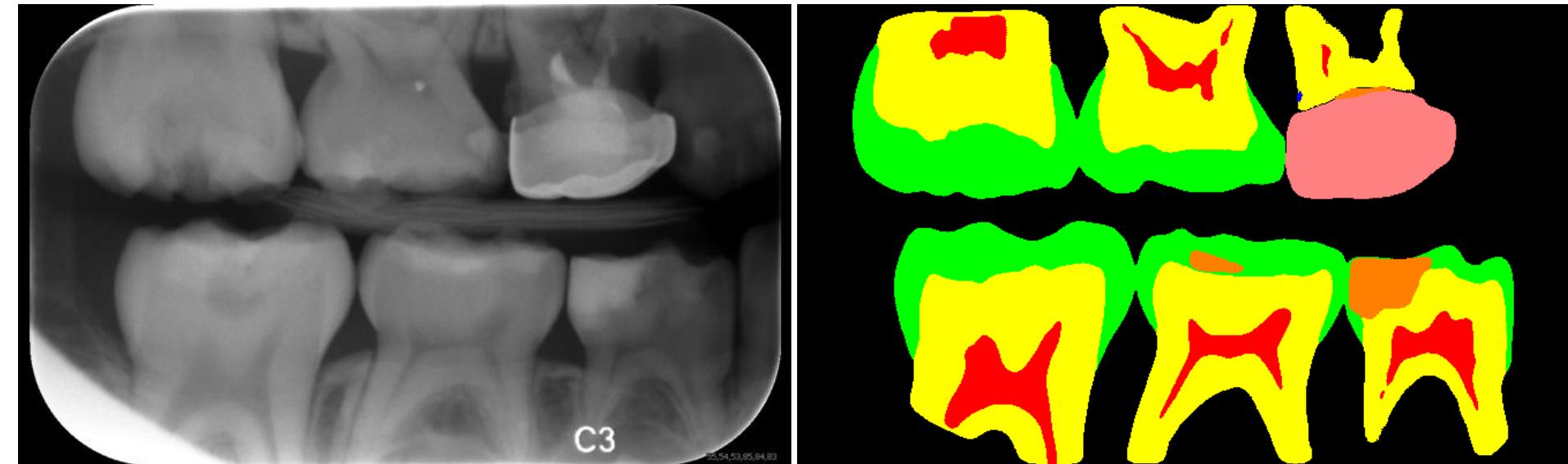
Cells in DIC Microscopy



- ISBI Cell Tracking Challenge 2015
Dataset „DIC-Hela“
 - Our Result: 77.6% Intersection over Union, **Winner**
 - Second best result: 46,0%

Data provided by Dr. Gert van Cappellen,
Erasmus Medical Center, Rotterdam, The Netherlands

Dental X-Ray Images

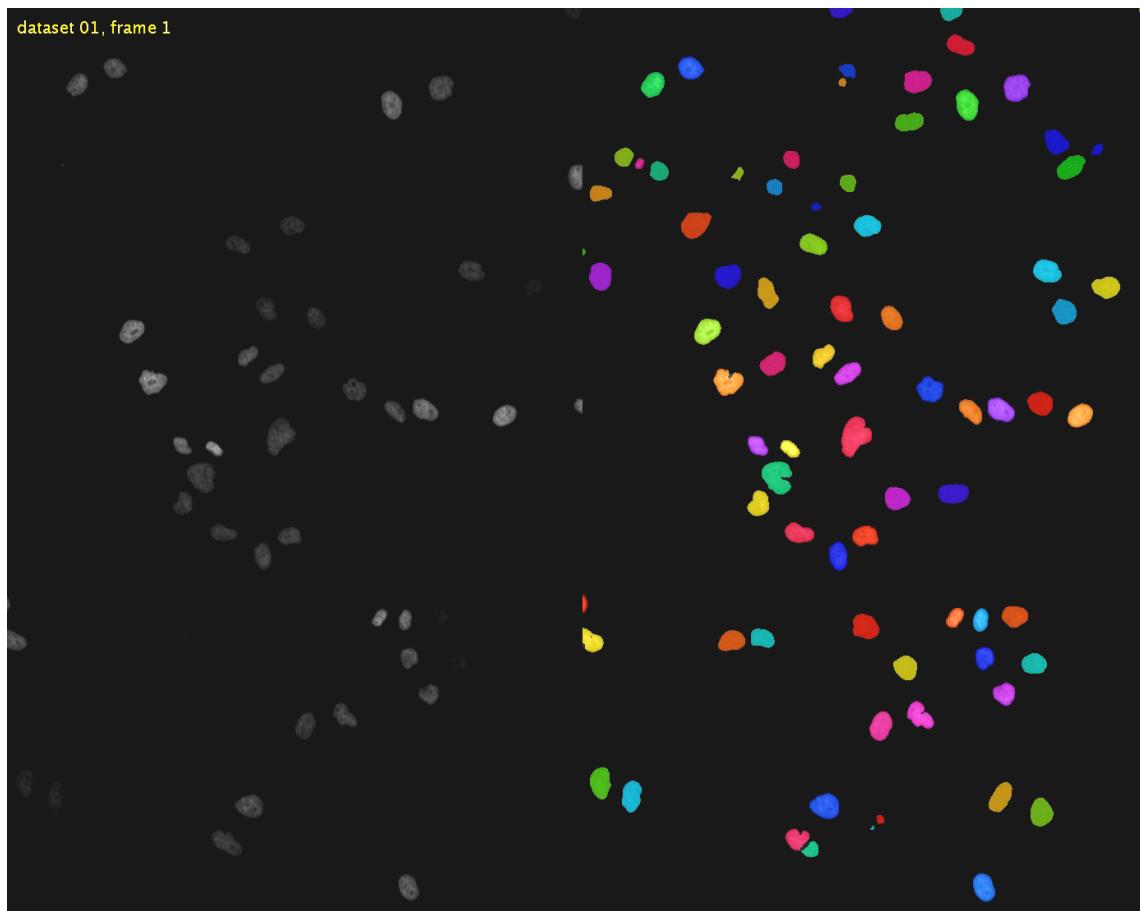


- Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI 2015
 - Our Result: 0.525 (average F-score), **Winner**
 - Second best result: 0.287

Data provided by Prof Ching-Wei Wang, PhD, Graduate Institute of Biomedical Engineering, National Taiwan University of Science and Technology, Taiwan,

Fluorescence Microscopy

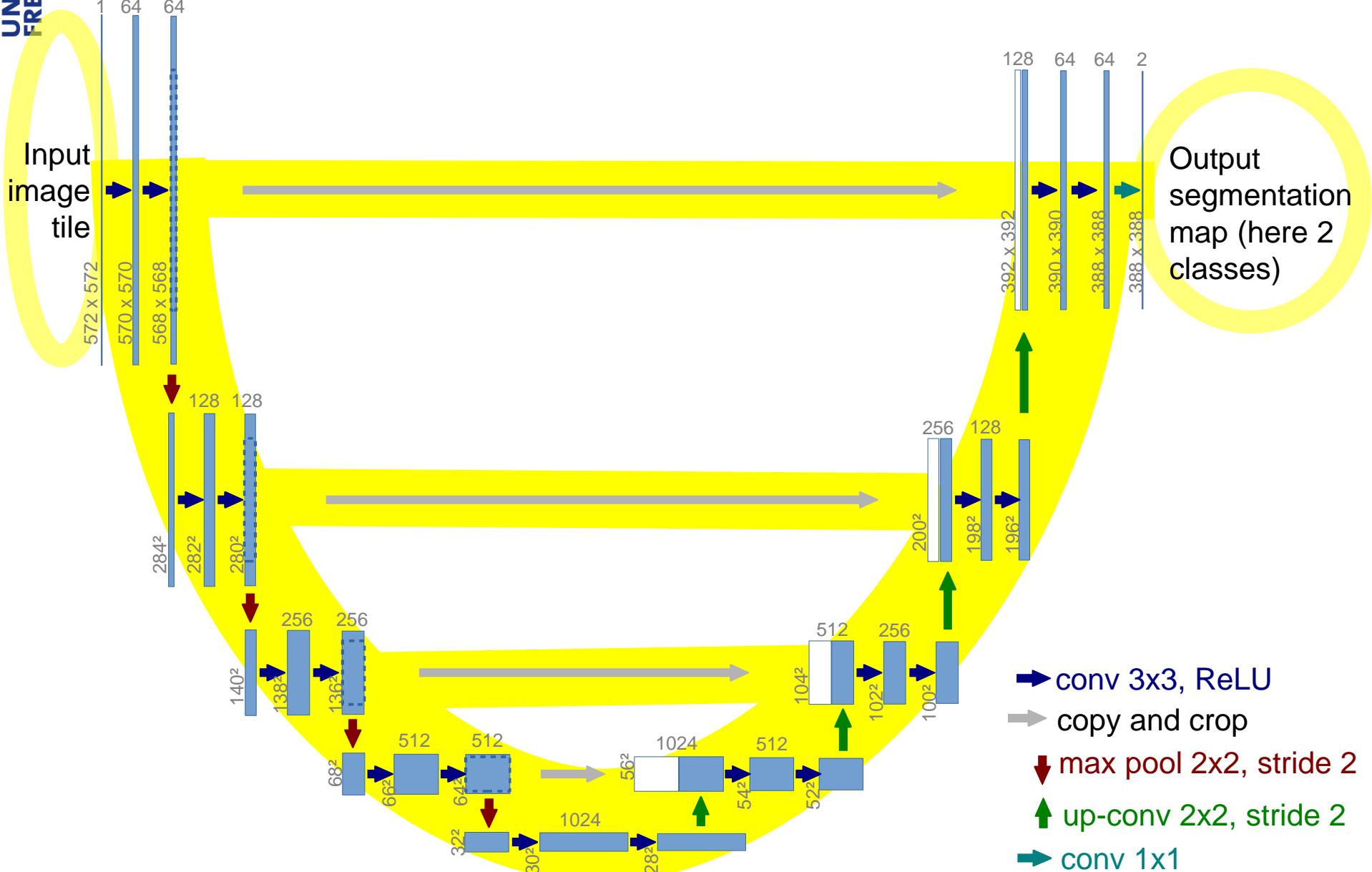
dataset 01, frame 1



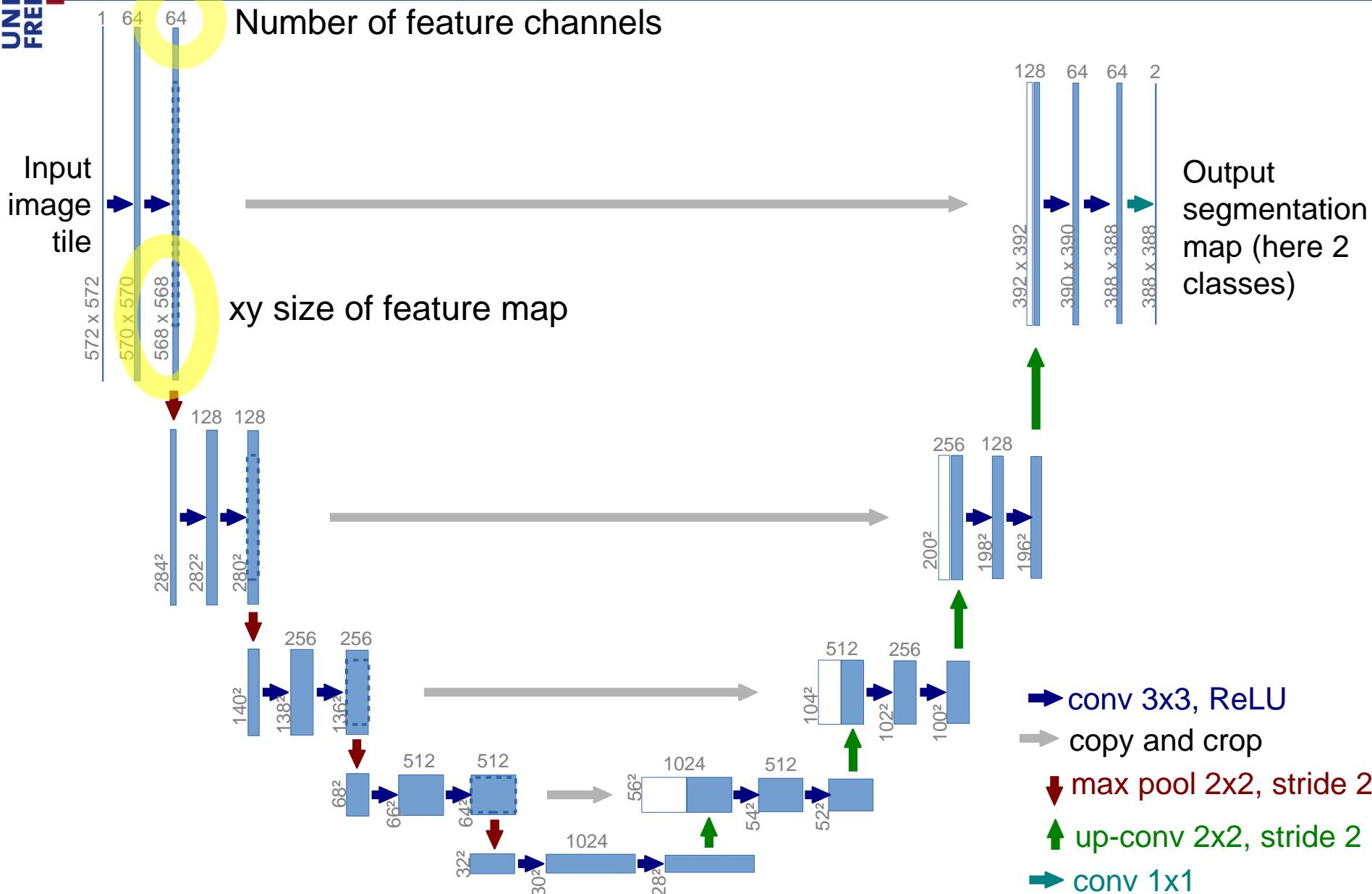
- ISBI Cell Tracking Challenge 2015 Dataset „Fluo-HeLa“
 - Our Result: 90% IoU, **(Winner)**
 - Second best result: 89%

Data provided by Mitocheck Consortium

U-net Architecture

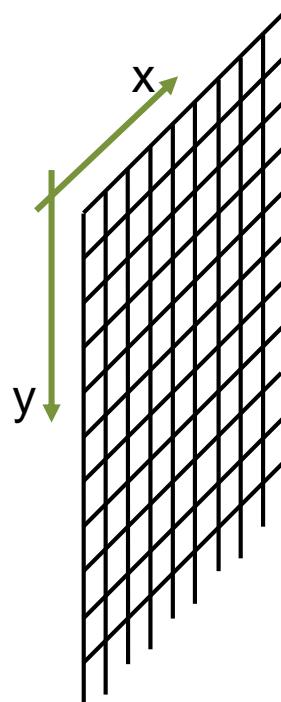


U-net Architecture

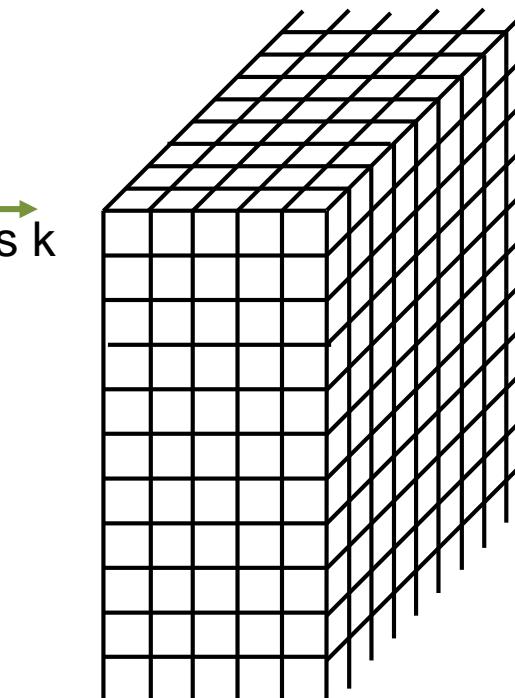
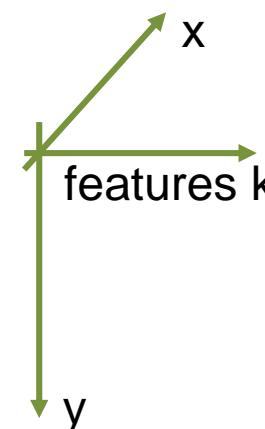


Feature maps

Feature maps are many neurons arranged in a grid

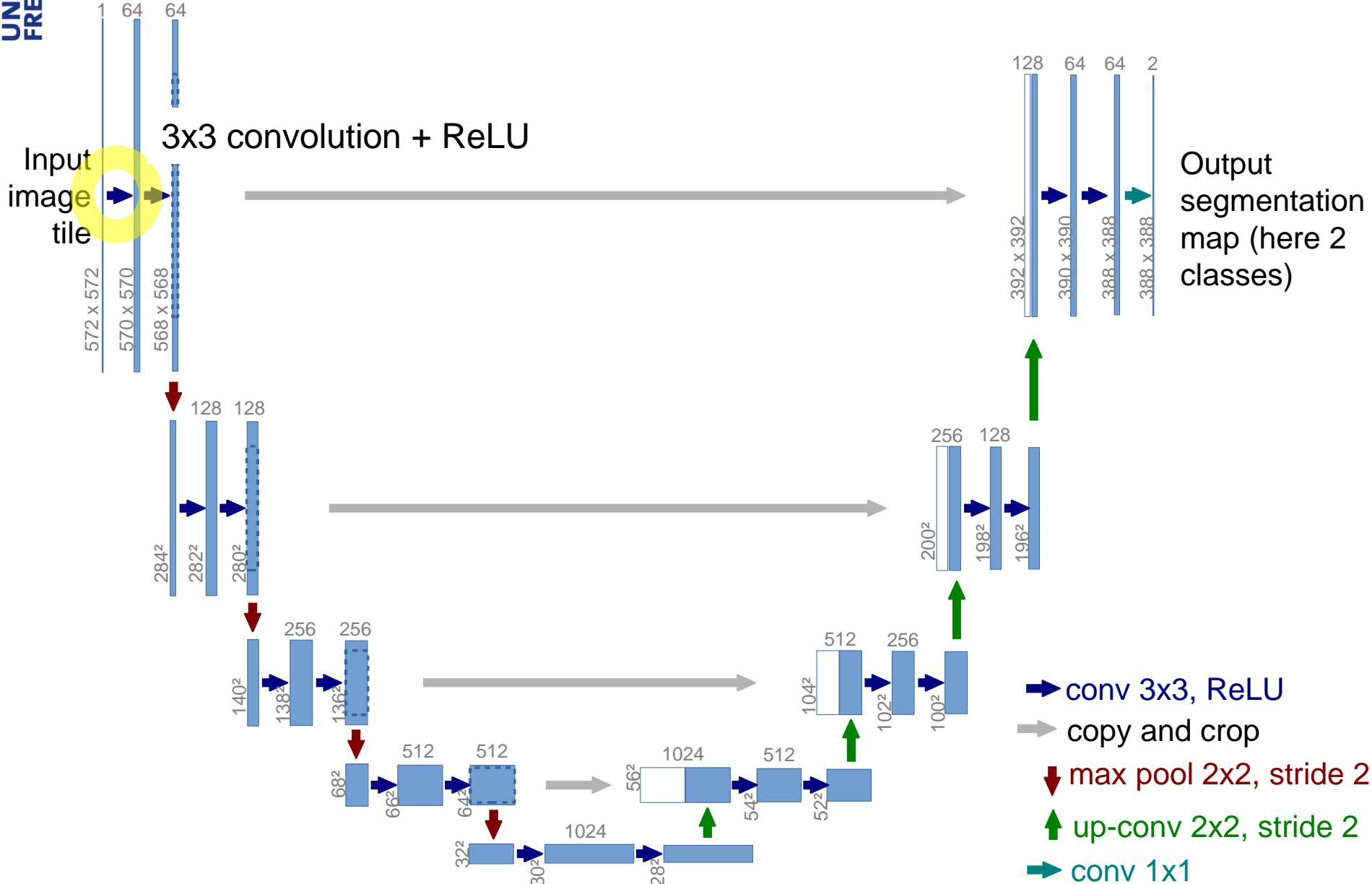


Example feature map
(1 channel)
e.g. gray image

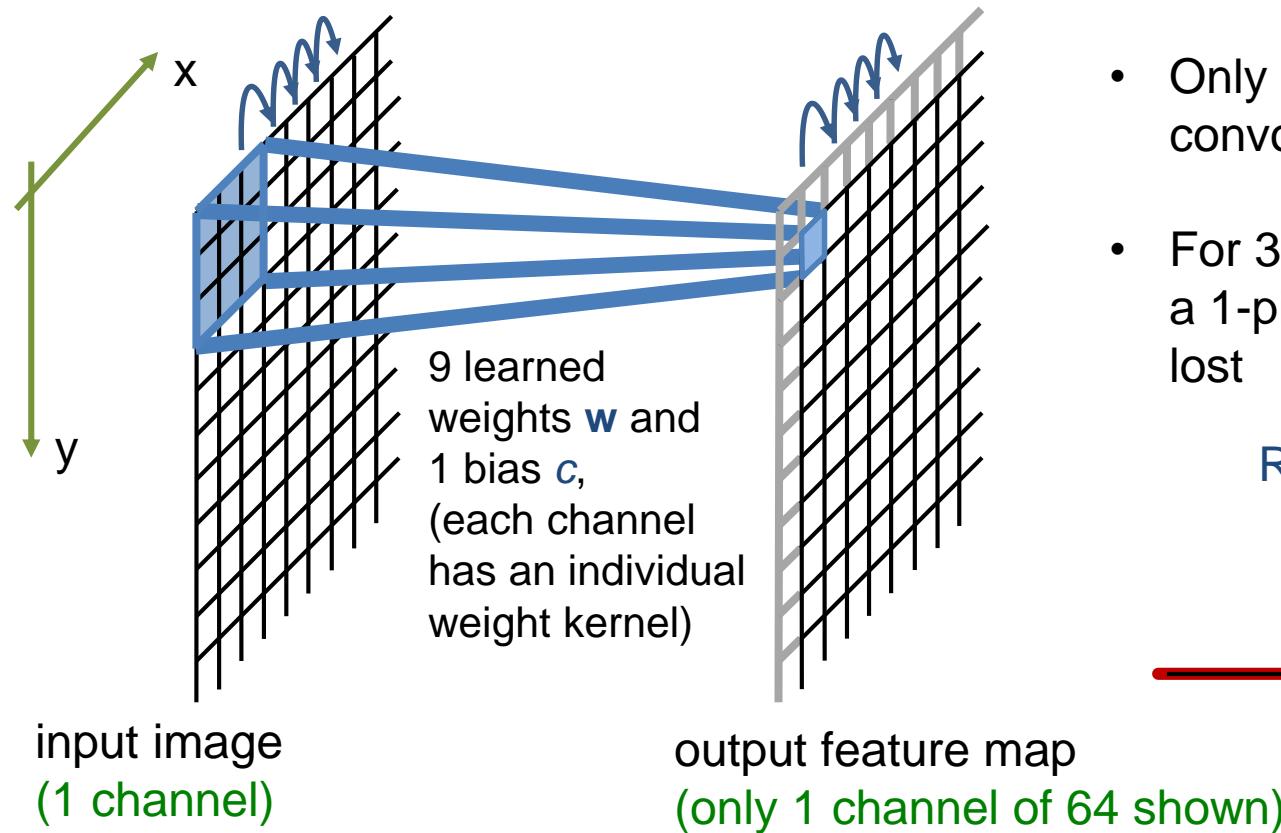


Example feature map
(5 channels)

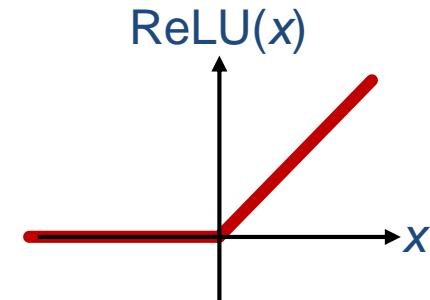
U-net Architecture



3x3 convolution + ReLU (first Layer)



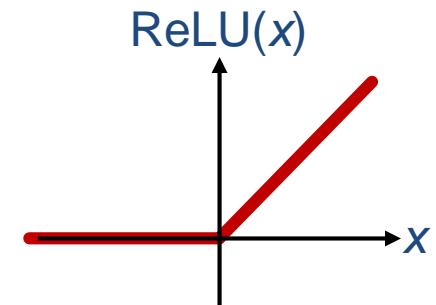
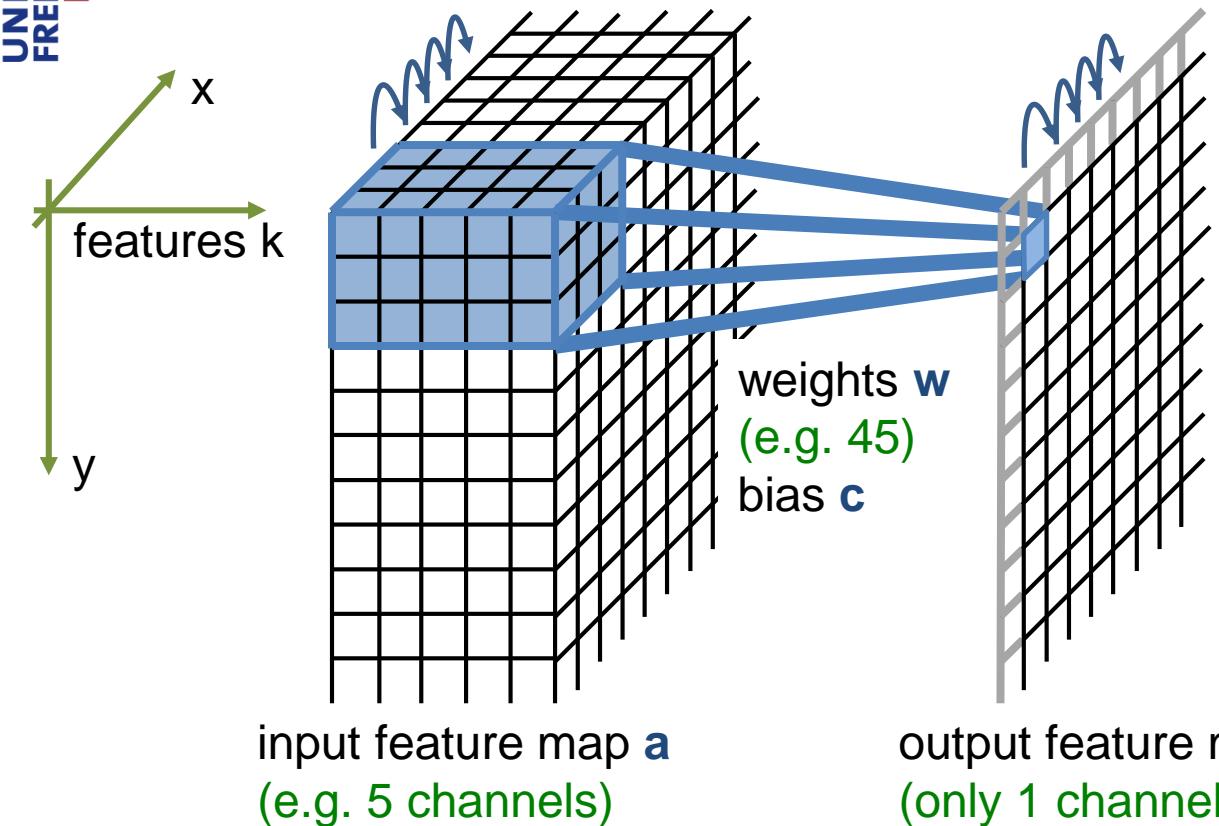
- Only valid part of convolution is used.
- For 3x3 convolutions a 1-pixel border is lost



$$b_{x,y,l} = \text{ReLU}\left(\sum_{\substack{i \in \{-1,0,1\} \\ j \in \{-1,0,1\}}} w_{i,j,l} \cdot a_{x+i,y+j} + c_l\right)$$

Intuition: Output neuron fires, when a certain input structure is seen
 (in the first layer tiny parts, like edges, corners, spots, texture elements, ...)

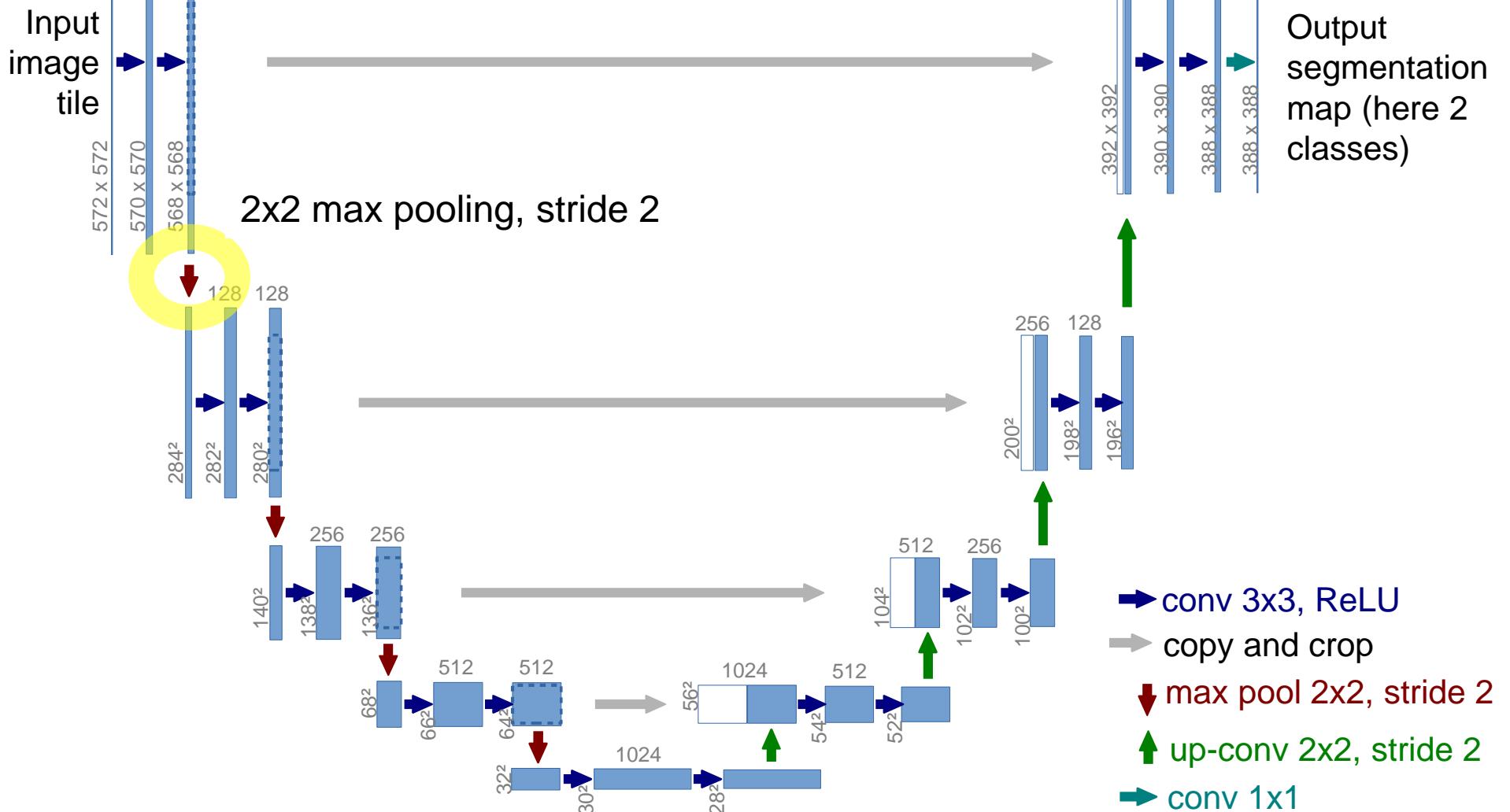
3x3 convolution + ReLU (all Subsequent Layers)



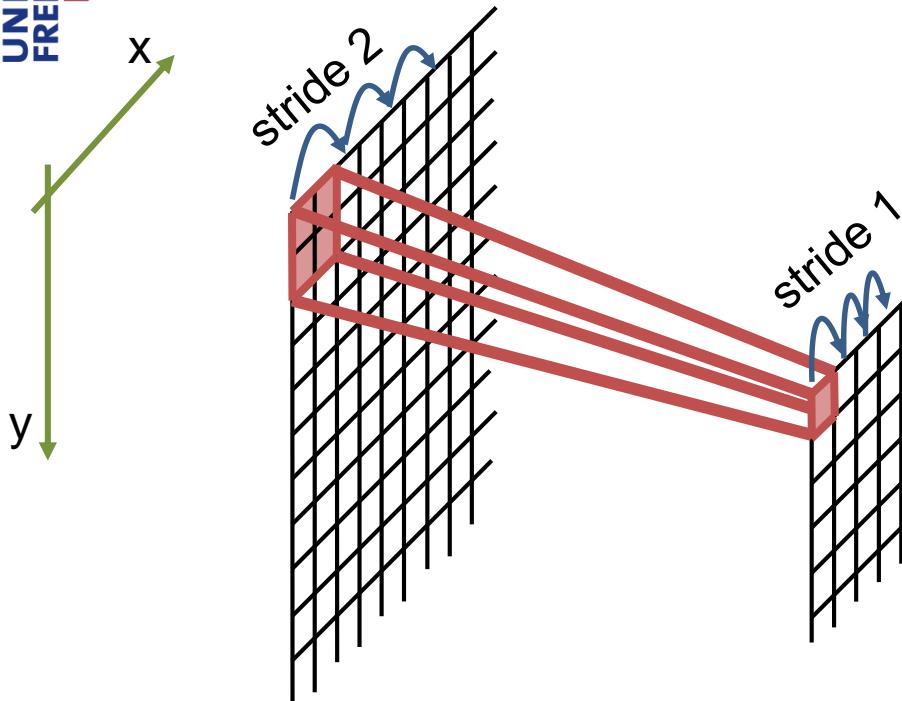
$$b_{x,y,l} = \text{ReLU}\left(\sum_{\substack{i \in \{-1,0,1\} \\ j \in \{-1,0,1\} \\ k \in \{1, \dots, K\}}} w_{i,j,k,l} \cdot a_{x+i,y+j,k} + c_l\right)$$

Intuition: Neurons fire, when **constellation of small parts build a certain larger part**

U-net Architecture



2x2 max-pooling



One channel of the input feature map **a**

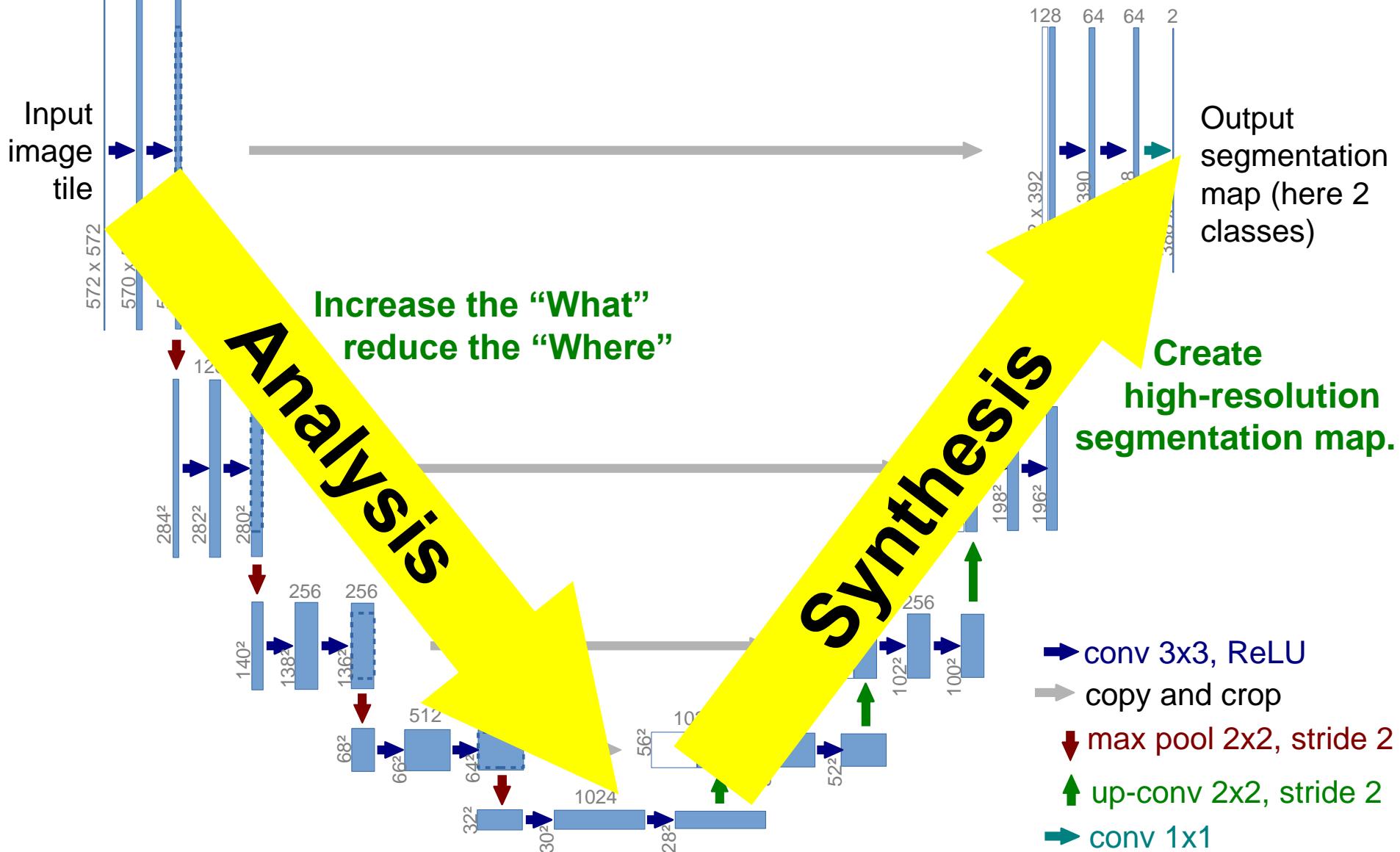
One channel of the output feature map **b**

resulting feature map has factor 2 lower spatial resolution

$$b_{x,y,k} = \max_{\substack{i \in \{0,1\} \\ j \in \{0,1\}}} (a_{2x+i, 2y+j, k})$$

Intuition: strongest activation (in local surrounding) is propagated
→ Robustness to spatial variations
→ reduce spatial resolution to increase context

U-Net Architecture



Synthesis Path

Input
image
tile

1 64 64
572 x 572
570 x 570
568 x 568

sequence of up-convolutions and concatenation with high-resolution features from analysis path

128 128
284²
282²
280²

256 256
140²
138²
136²

512 512
68²
66²
64²
32²
30²

512 256 1024
104² 102² 100²

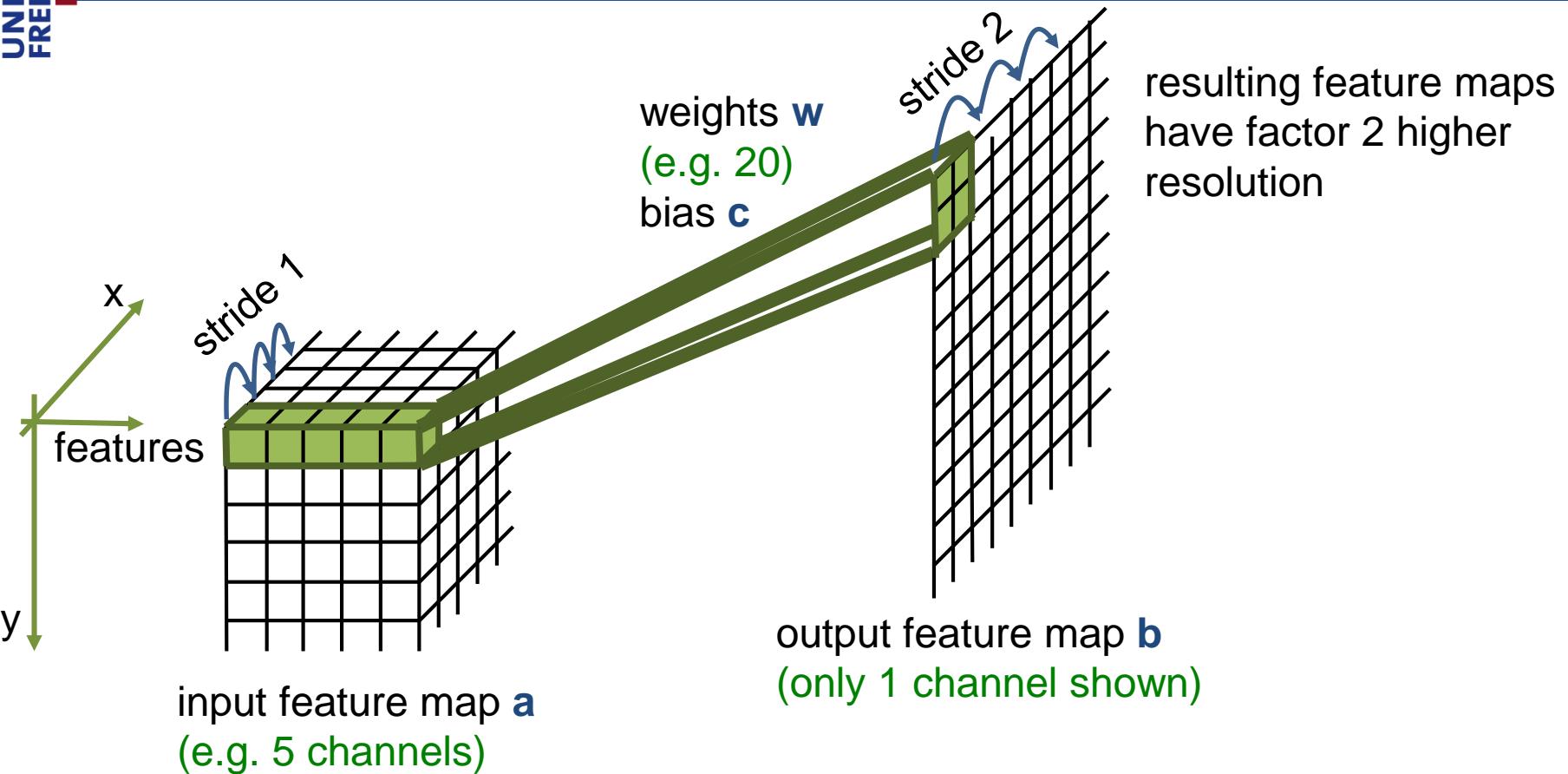
256 128
200²
198²
196²

128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388

Output
segmentation
map (here 2
classes)

- conv 3x3, ReLU
- copy and crop
- ↓ max pool 2x2, stride 2
- ↑ up-conv 2x2, stride 2
- conv 1x1

2x2 up-convolution



$$b_{2x+i, 2y+j, l} = \text{ReLU}\left(\sum_{\substack{i \in \{0,1\} \\ j \in \{0,1\} \\ k \in \{1, \dots, K\}}} w_{i,j,k,l} \cdot a_{x,y,k} + c_l\right)$$

Intuition: Learned “Upsampling”

→ Synthesize high-res images from low-res feature vectors

Synthesis Path

Input
image
tile

1 64 64
 572×572
 570×570
 568×568

128 128
 284^2
 282^2
 280^2

256 256
 140^2
 138^2
 136^2
 68^2
 66^2
 64^2
 32^2
 30^2

512 512
 1024
 56^2
 54^2
 52^2
 1024
 28^2

256 128
 200^2
 198^2
 196^2

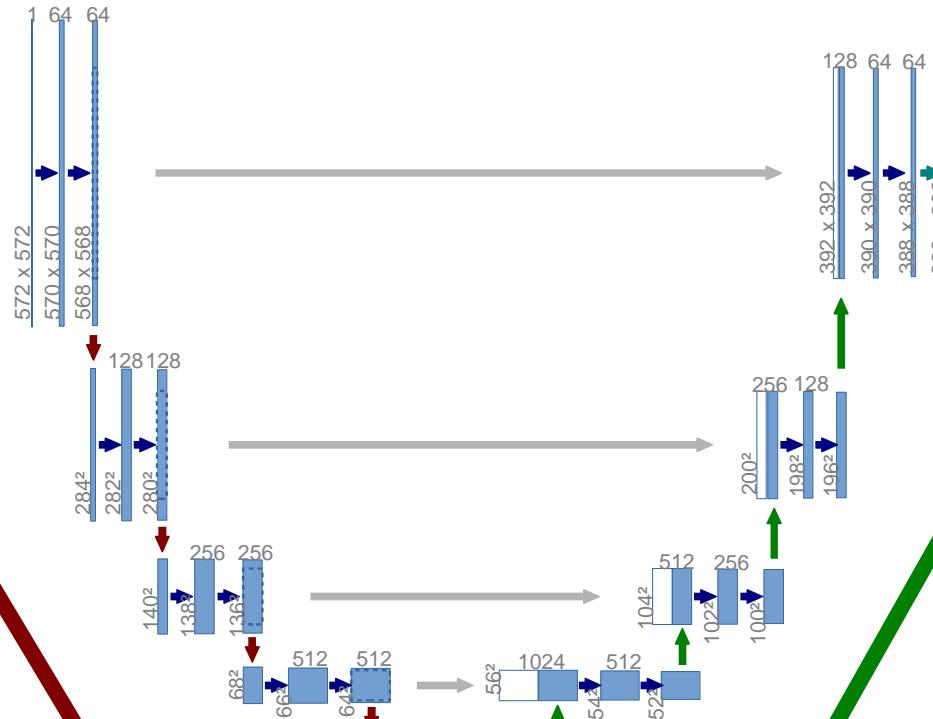
128 64 64 2
 392×392
 390×390
 388×388
 388×388

Output
segmentation
map (here 2
classes)

- conv 3x3, ReLU
- copy and crop
- ↓ max pool 2x2, stride 2
- ↑ up-conv 2x2, stride 2
- conv 1x1

What is in the Feature Maps?

Activation maps for
tiny parts (edges,
corners, spots, ...)



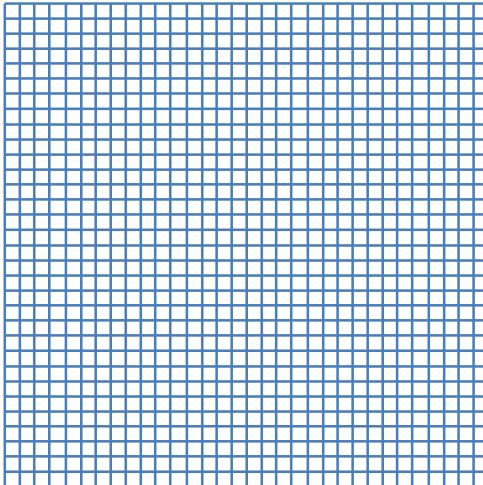
Activation maps for **large parts**
(full cells, mitochondria, cell
constellations, ...)

Activation maps that
encode the final
fine segmentation
structures

Activation maps that encode
coarse segmentation structures

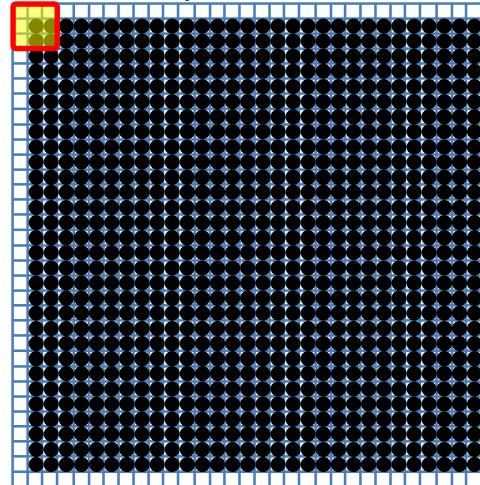
Analysis Path: Resulting Feature Vectors

Input image

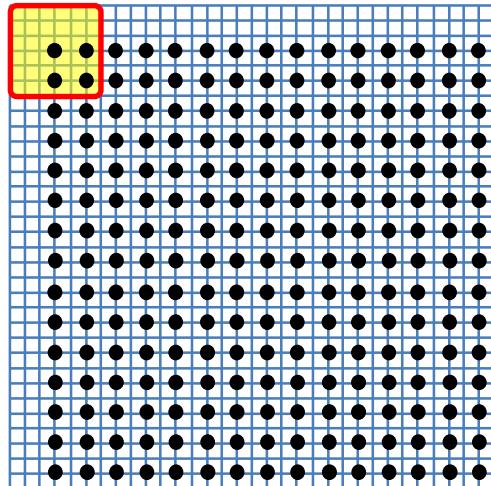


Conv.
3x3,
ReLU

64 Features at every pixel
From 3x3 pixel context

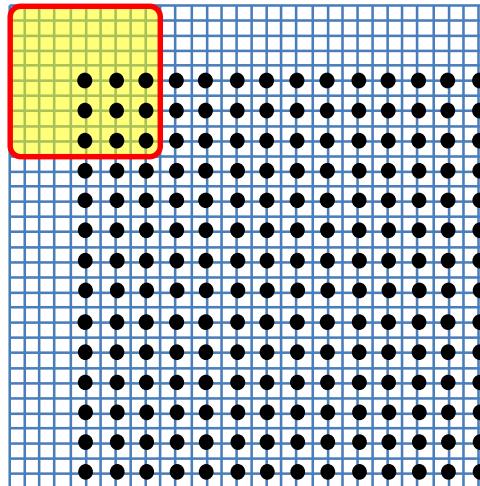


64 Features at every 2nd pixel
From 6x6 pixel context



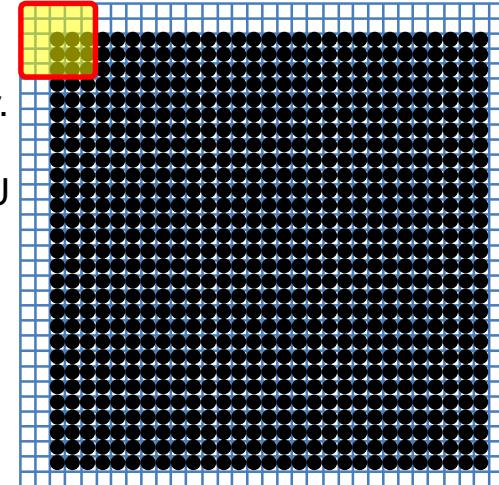
Conv.
3x3,
ReLU

128 Features at every 2nd pixel
From 10x10 pixel context



Conv.
3x3,
ReLU

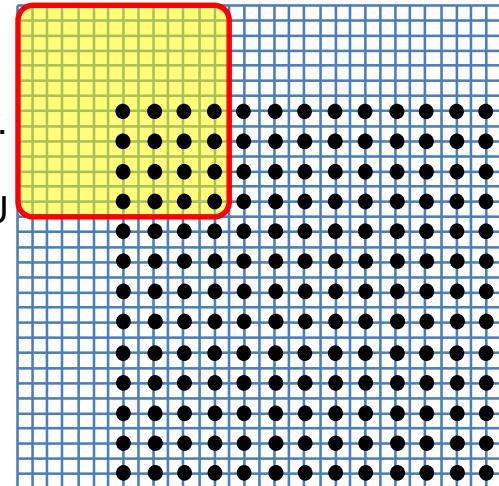
64 Features at every pixel
From 5x5 pixel context



Max pool
2x2

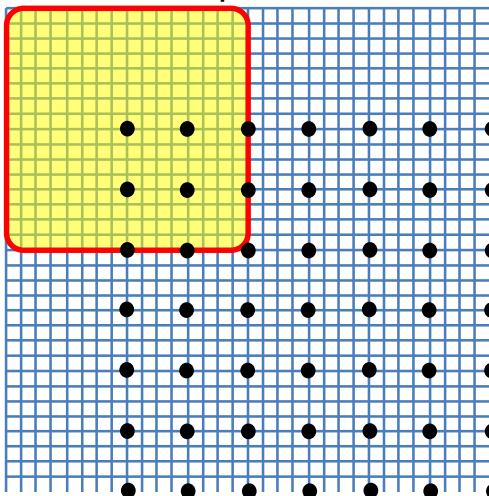
Max pool
2x2

128 Features at every 2nd pixel
From 14x14 pixel context

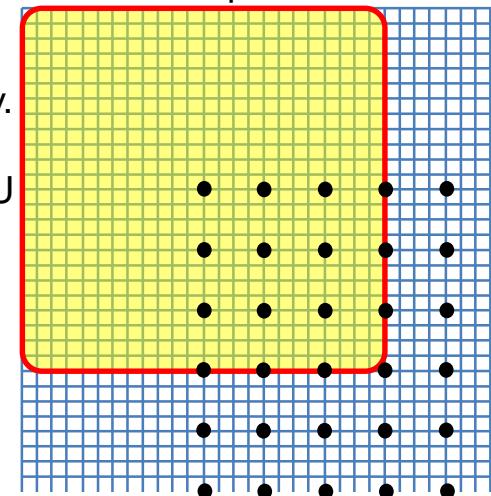


Anaylsis Path: Resulting Feature Vectors

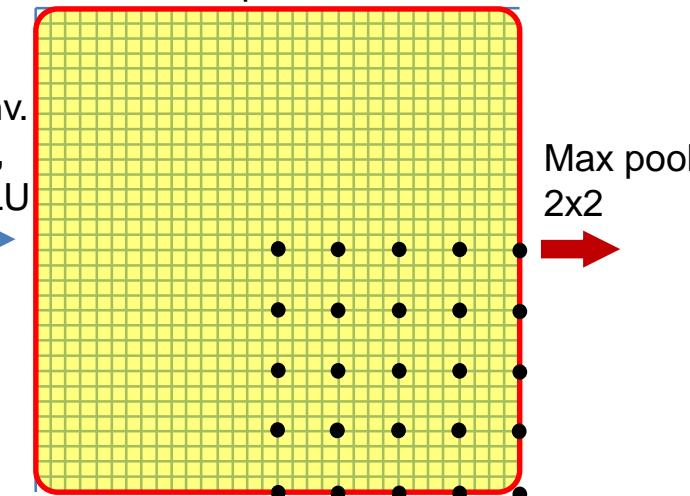
128 Features at every 4th pixel
From 16x16 pixel context



256 Features at every 4th pixel
From 24x24 pixel context



256 Features at every 4th pixel
From 32x32 pixel context

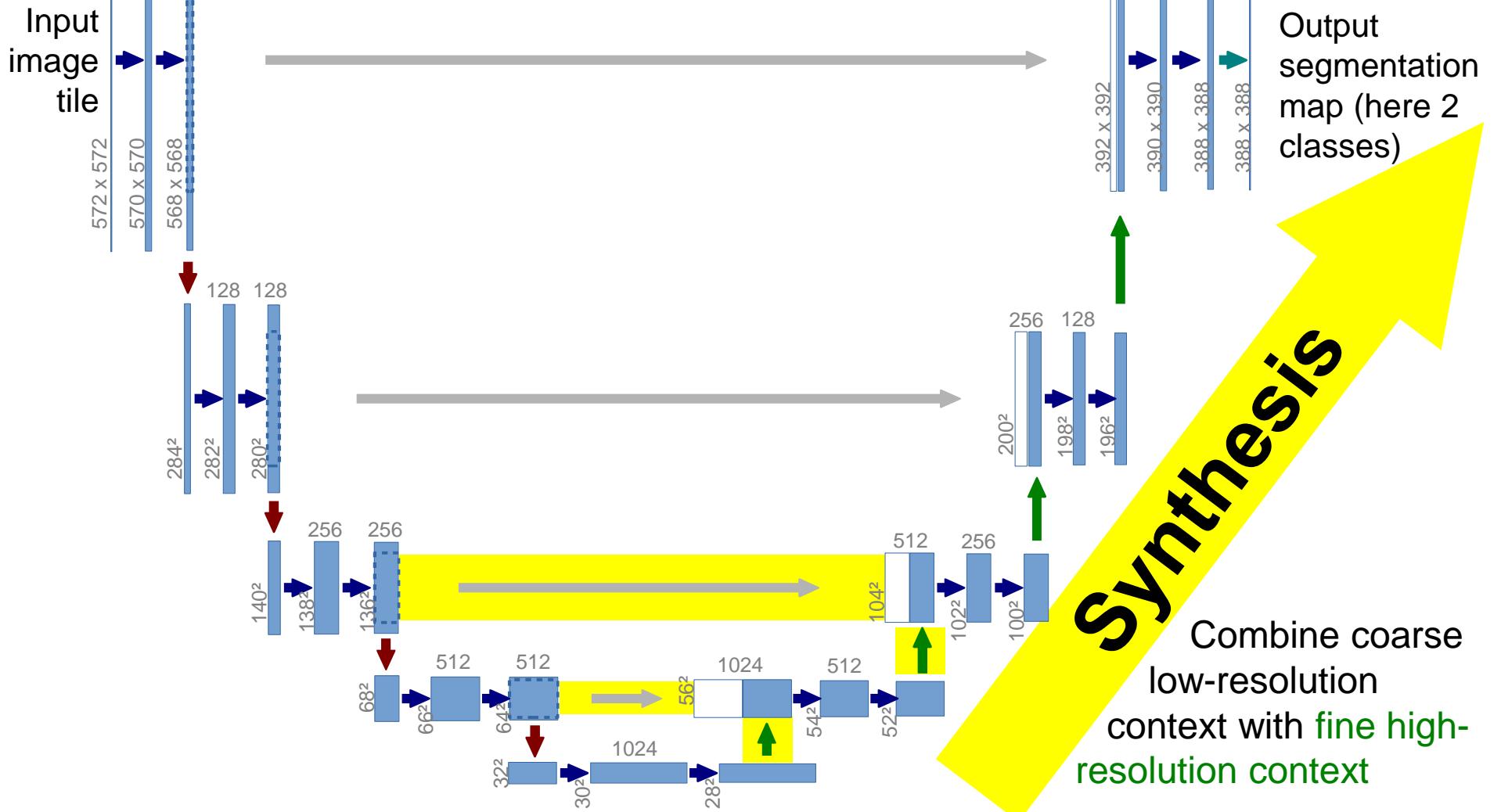


and so on: Increase the “**What**”, reduce the “**Where**”

Features	256	512	512	512	1024	1024
Sampling	8	8	8	16	16	16
Context	36	52	68	76	108	140

... until 1024 features at every 16th pixel from 140x140 pixel context

U-net Architecture



Synthesis Path

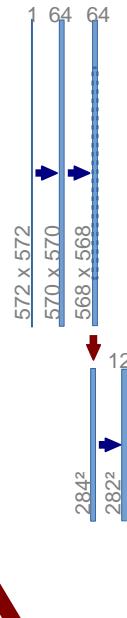
- Combine coarse low-resolution context with **fine** high-resolution context

Features	512 +512	512	512	256 +256	256	256	128 +128	128	128	64 +64	64	64	7
Sampling	8	8	8	4	4	4	2	2	2	1	1	1	1
Context	132 (68)	148	164	160 (32)	168	176	174 (14)	178	182	181 (5)	183	185	185

- ... until 7 features (scores for the 7 classes) at every pixel from 185x185 pixel context.

Visualization of Standard Classification Networks

Activation maps for
tiny parts (edges,
corners, spots, ...)

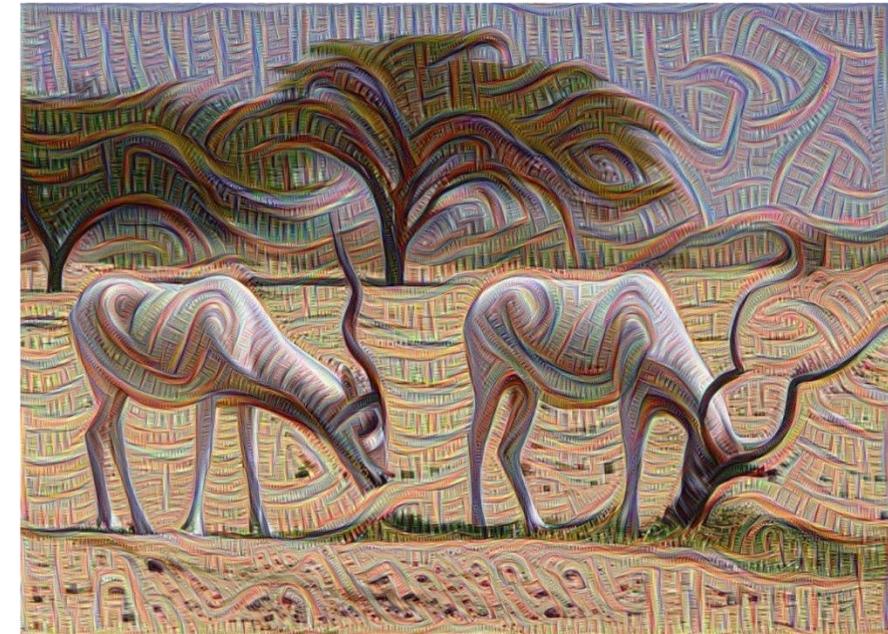


Activation maps for **large parts**
(full cells, mitochondria, cell
constellations, ...)

Activation maps that
encode the final
fine segmentation
structures

Activation maps that encode
coarse segmentation structures

Visualization of Neuron Activations



- propagate the image through the network
- check with neurons get the **strongest activation**
- change the image, such that these activations are further **amplified**
- **Here:** Neurons in **early layers** (encoding small-scale structures)

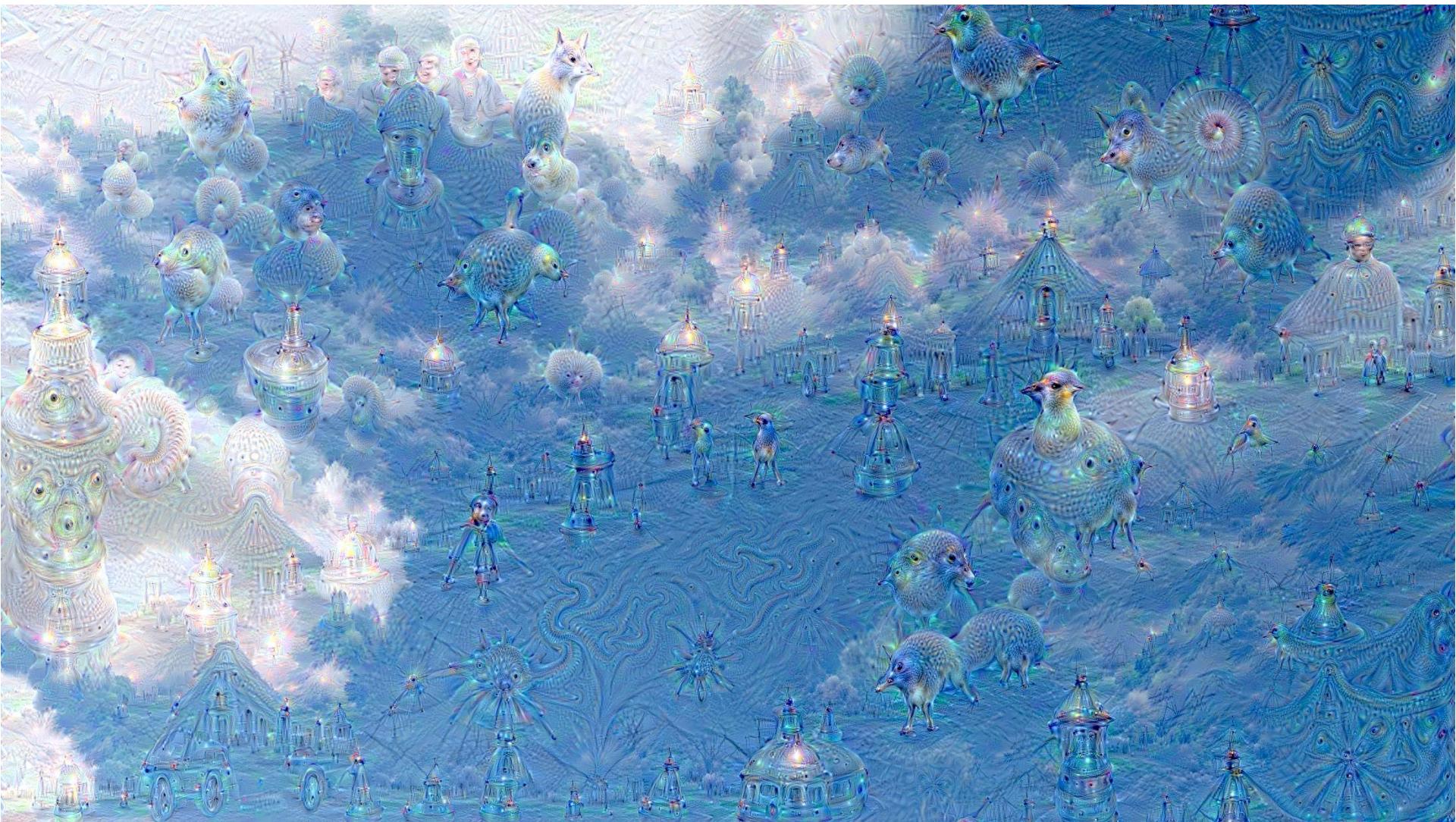
<http://googleresearch.blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html>

Neuron Activations in Late Layers

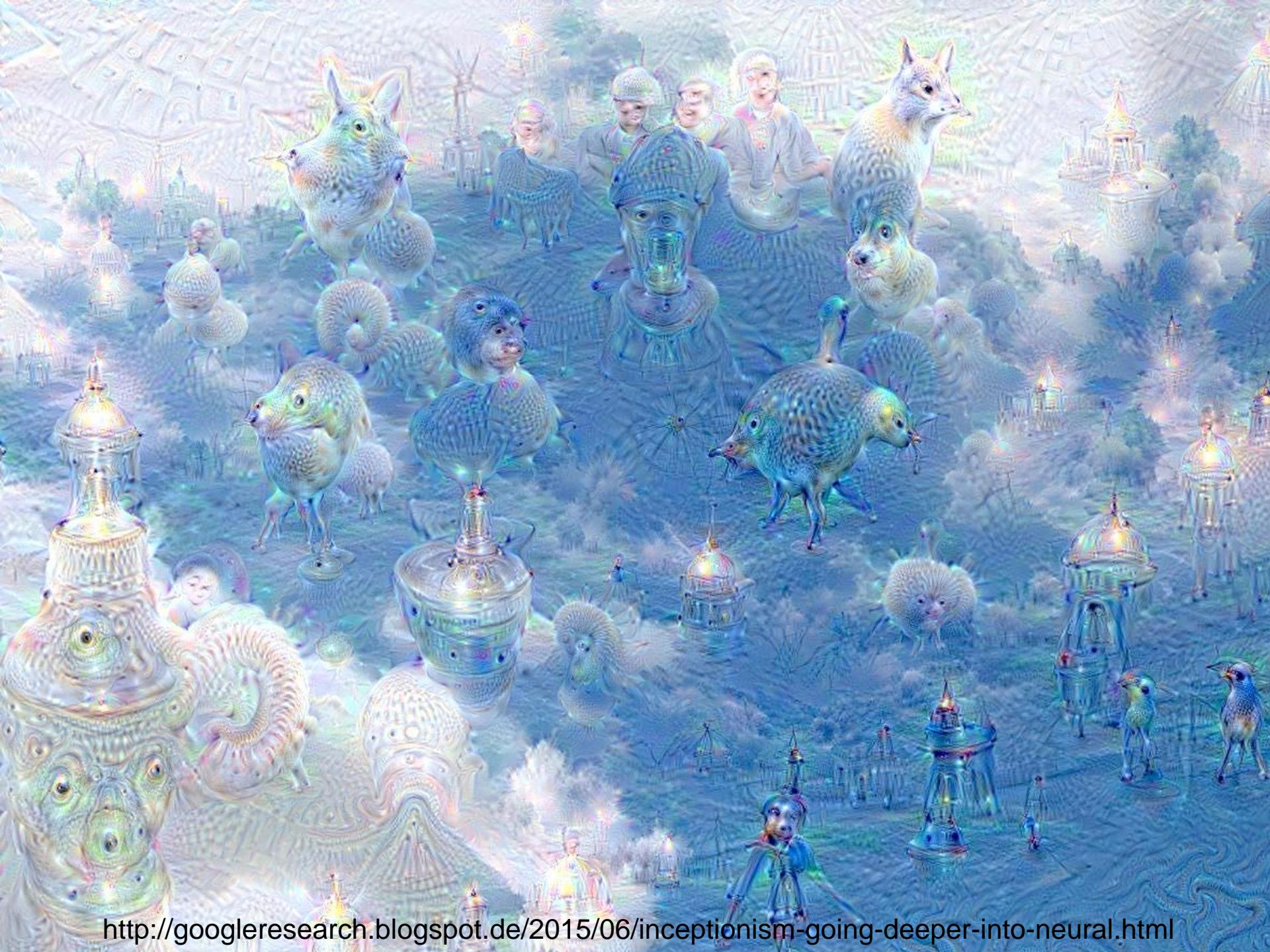


<http://googleresearch.blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html>

Amplification of large-scale structures



<http://googleresearch.blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html>

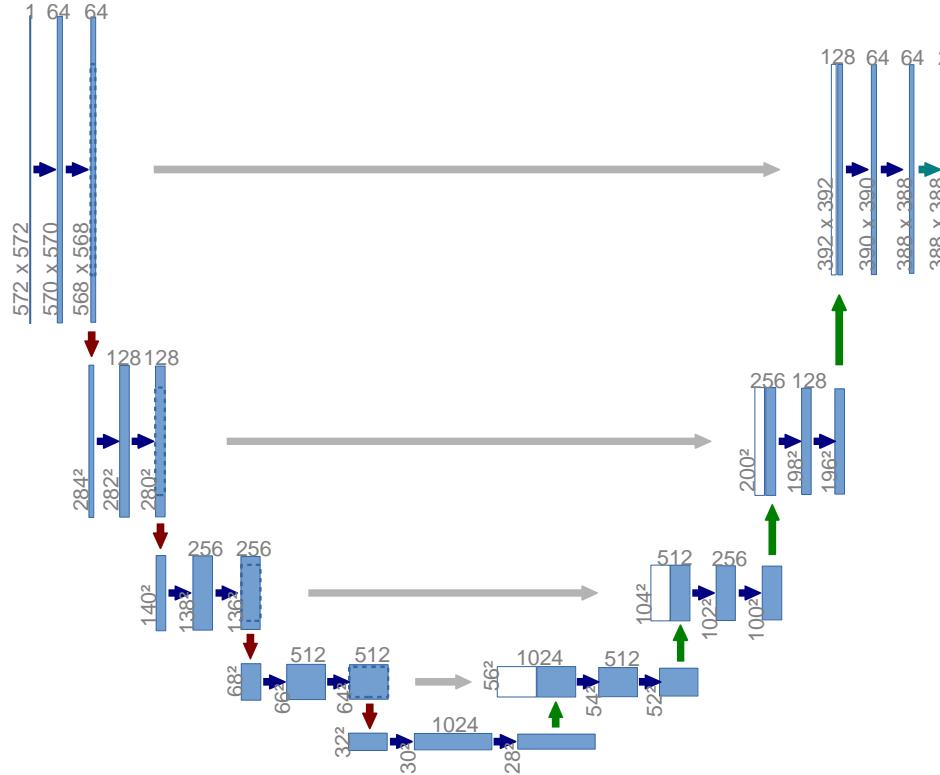


<http://googleresearch.blogspot.de/2015/06/inceptionism-going-deeper-into-neural.html>



Overlap-tile strategy for arbitrary large images

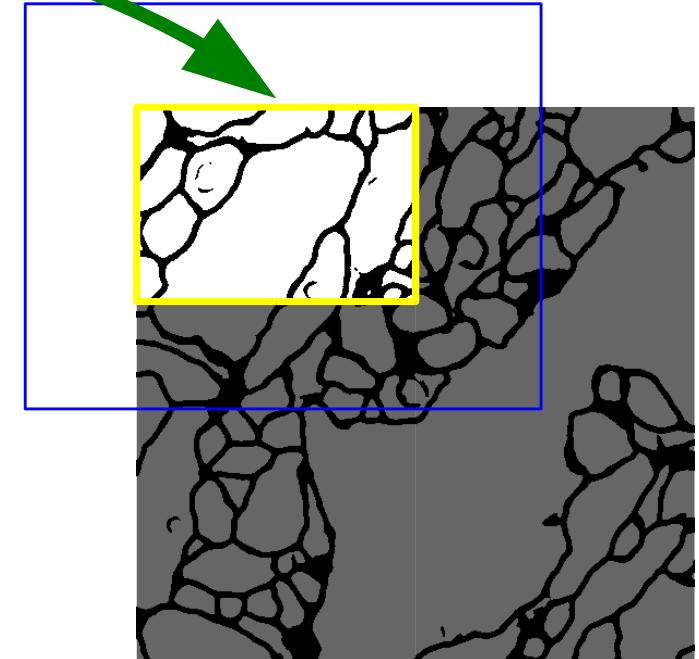
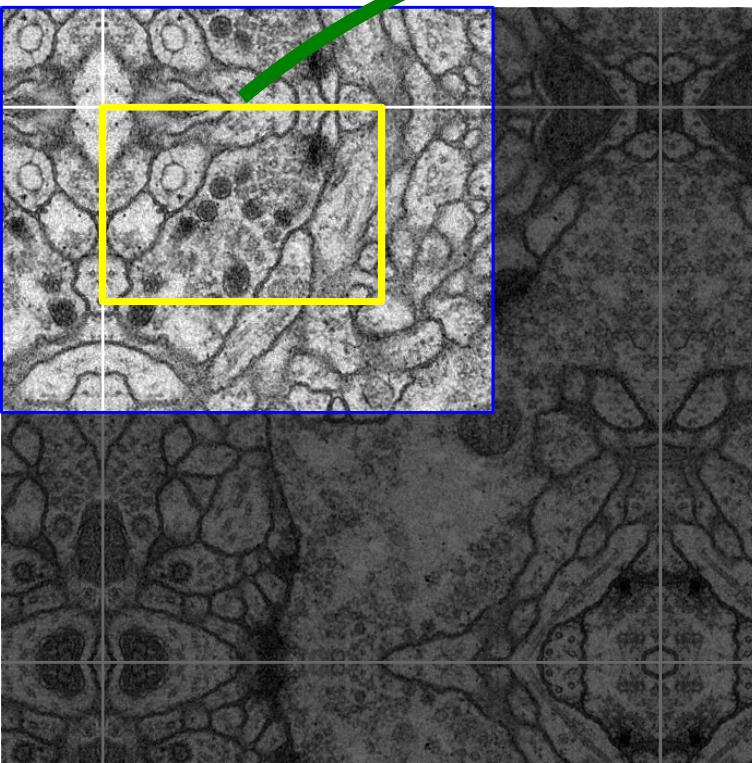
Input image tile,
572 x 572 pixel



Output
segmentation map:
388 x 388 pixel

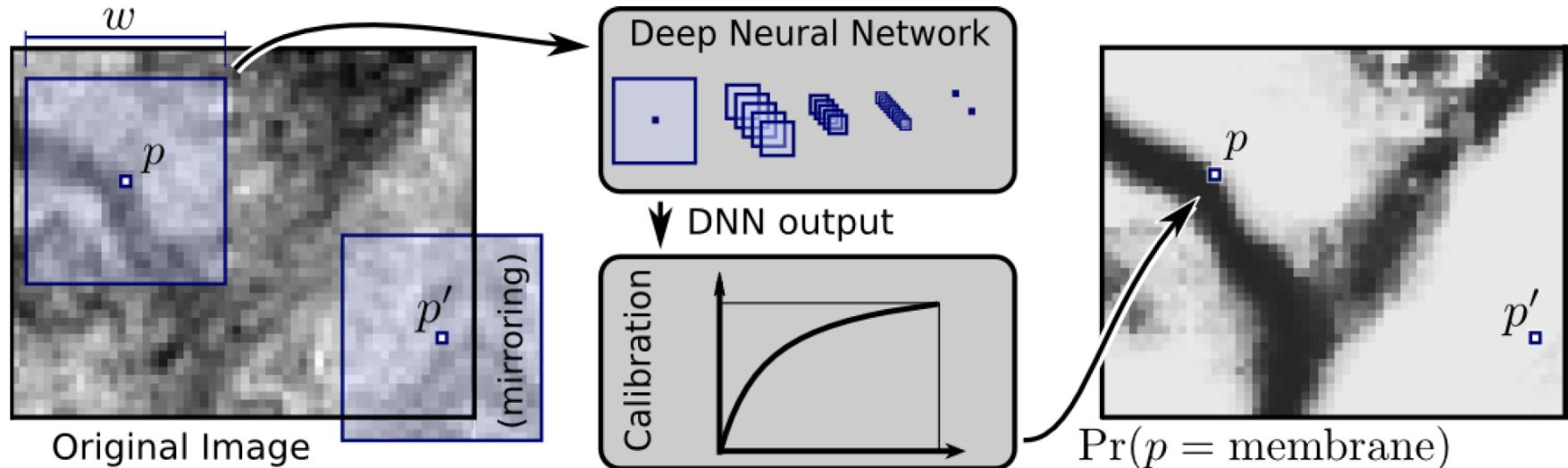
Use only the valid part of each convolution

Overlap-tile strategy for arbitrary large images



- Segmentation of the yellow area needs input data of the blue area
- Raw data extrapolation my mirroring

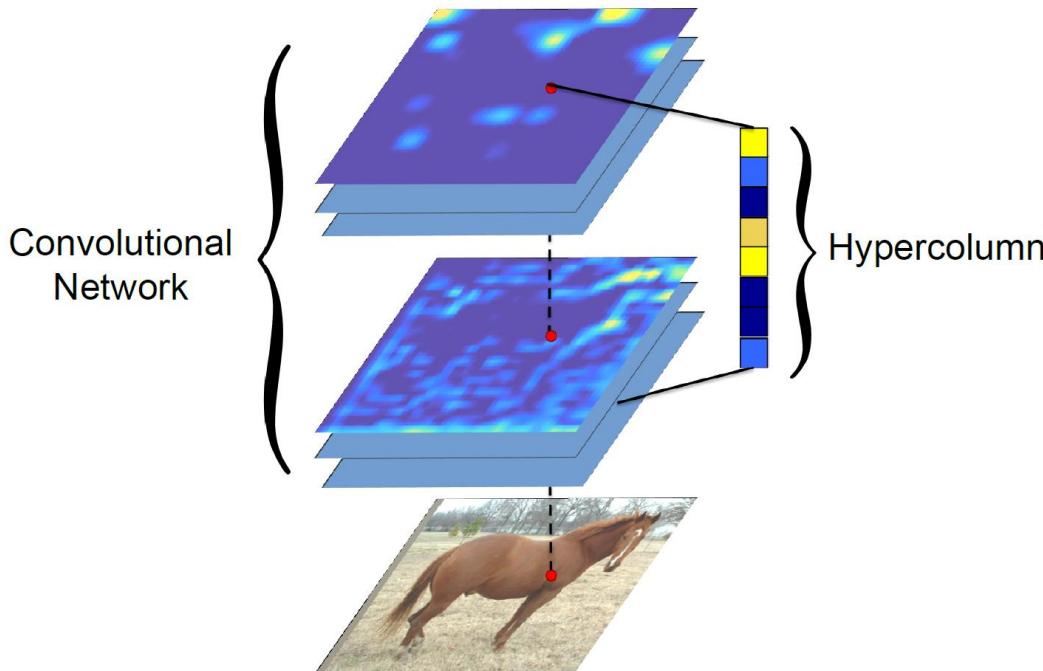
Related Work: Sliding-Window Approach



D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber - Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images (NIPS 2012)

- **Sliding-window approach** with a standard classification network
- Winner of **EM segmentation challenge** at ISBI 2012
- Needs **full network evaluation** for each pixel
- **Trade-off** between precise localization and large context (larger windows need **more max pooling layers**)
- (u-net outperformed them now)

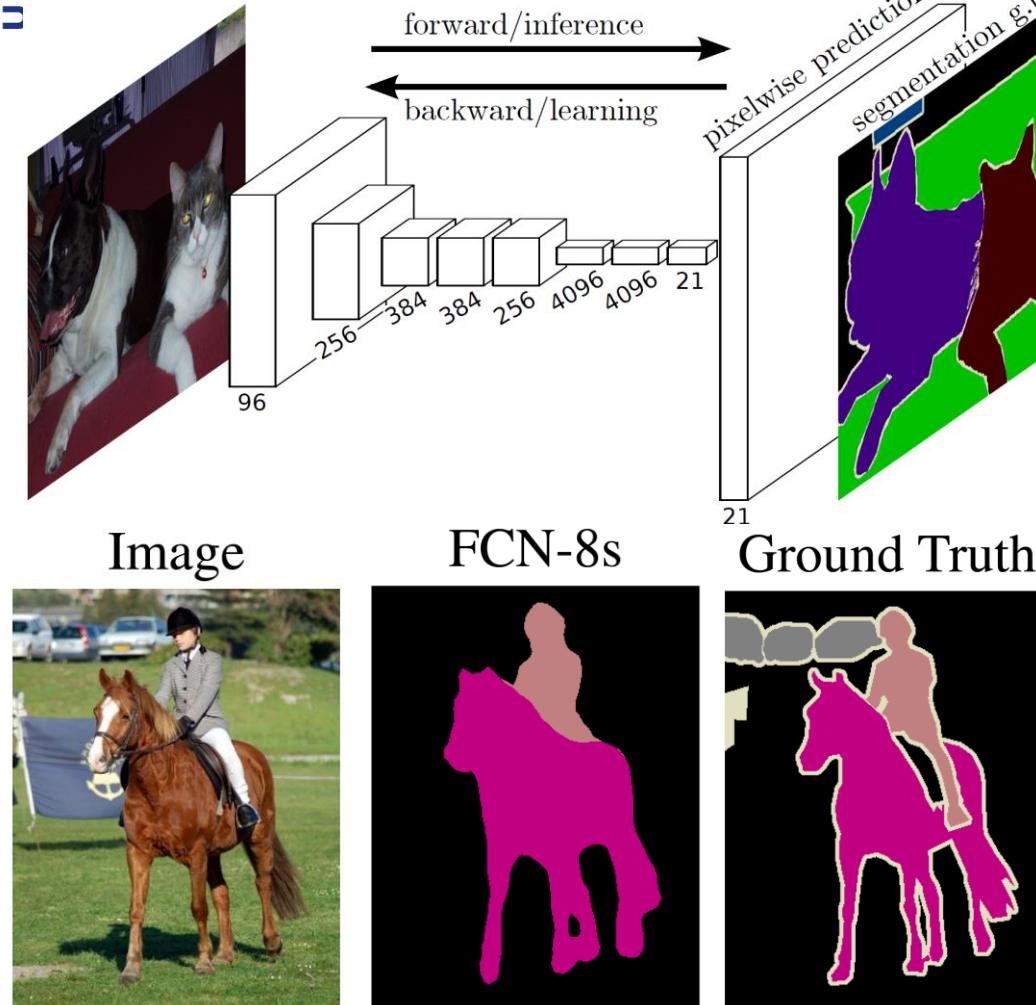
Related Work: Hypercolumns



- **Rescale all feature maps** to original image size.
- Use **support vector machine** to predict label at each position.
- Needs a **pre-trained classification network**.
- **No end-to-end** training possible.

B. Hariharan, P. Arbeláez, R. Girshick, J. Malik: "Hypercolumns for Object Segmentation and Fine-grained Localization" CVPR, 2015. arXiv:1411.5752 [cs.CV]

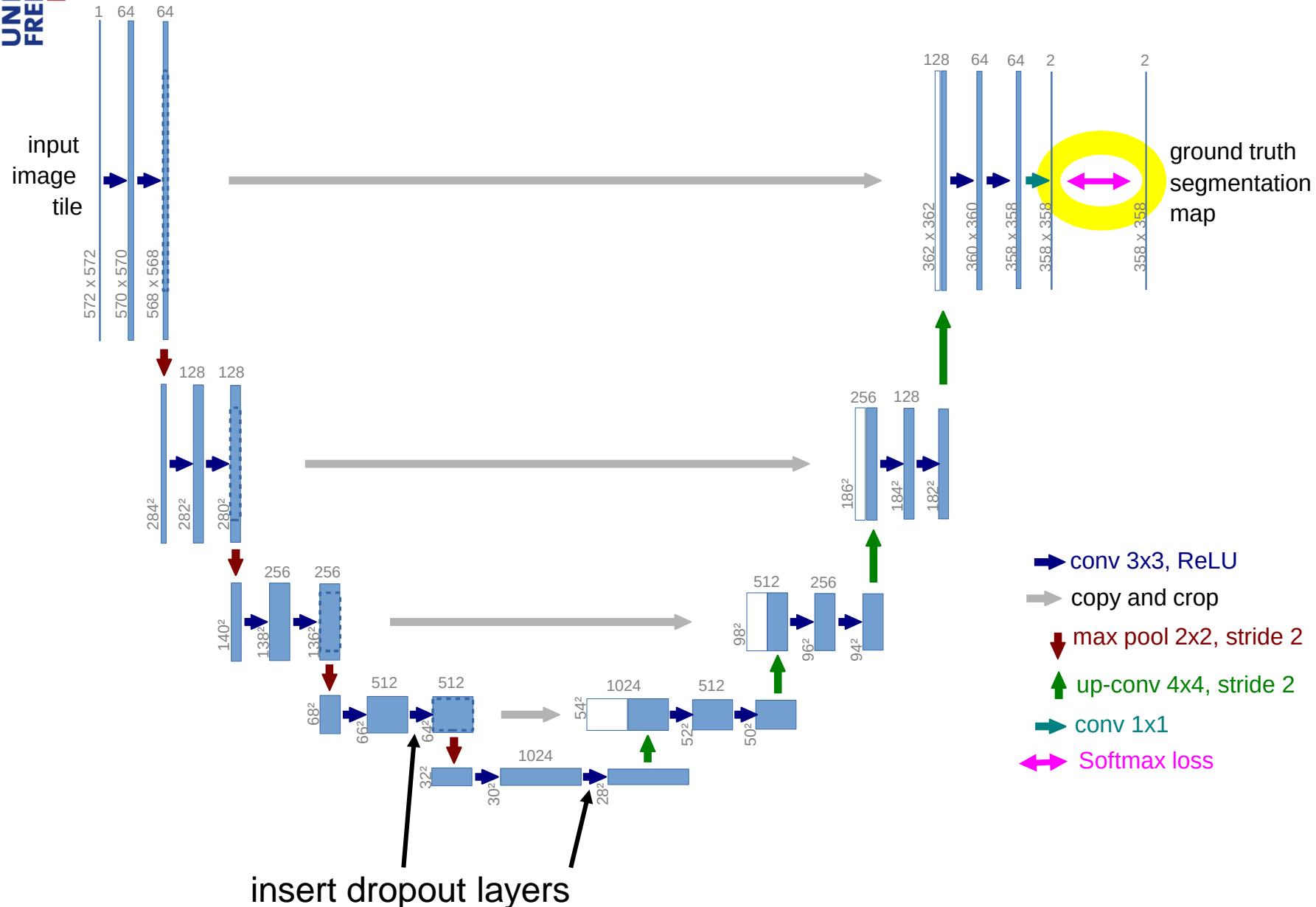
Related Work: Fully Convolutional Network



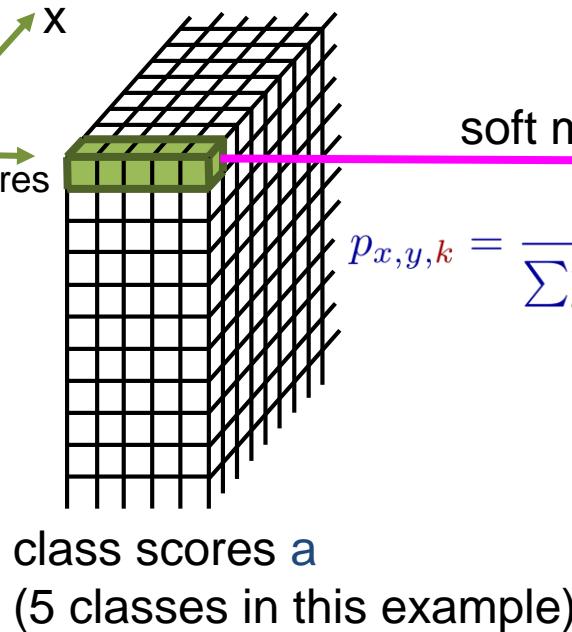
- **Nearly identical architecture** like u-net
- Uses a **pre-trained classification network**
- Uses only **1 feature channel per class** in lowest resolution
- Does not go back to full resolution in **synthesis path**
- Uses **padded convolutions**, no tiling possible

J. Long, E. Shelhamer, T. Darrell: "Fully Convolutional Networks for Semantic Segmentation".
CVPR 2015, arXiv:1411.4038 [cs.CV]

Training the U-net

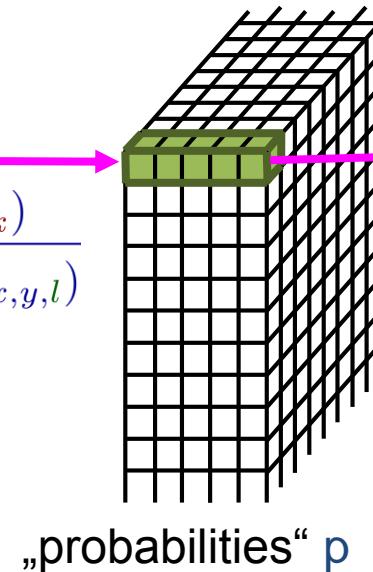


Softmax-loss layer



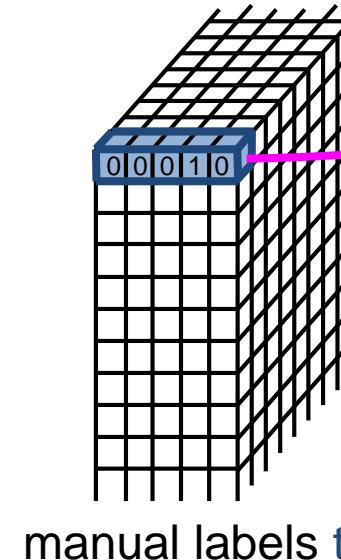
soft max

$$p_{x,y,k} = \frac{\exp(a_{x,y,k})}{\sum_{l=1}^K \exp(a_{x,y,l})}$$

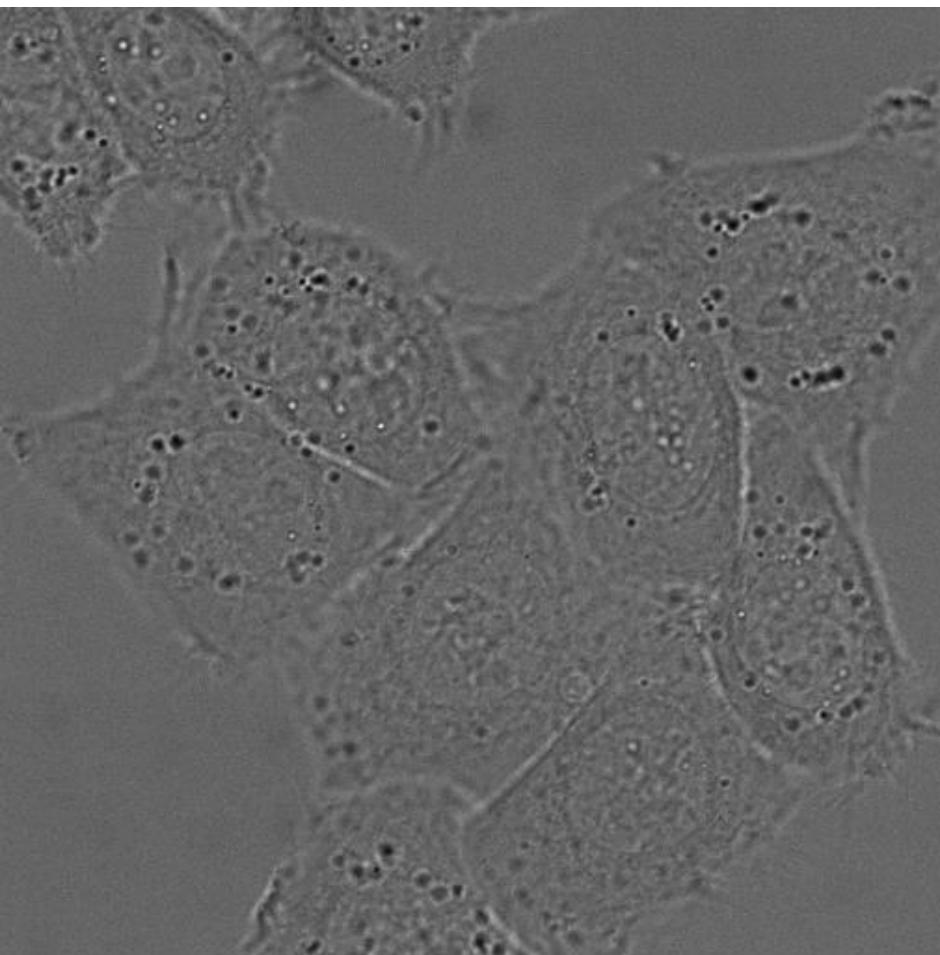


$$E = - \sum_{\substack{(x,y) \in \Omega \\ k \in \{1, \dots, K\}}} t_{x,y,k} \cdot \log(p_{x,y,k})$$

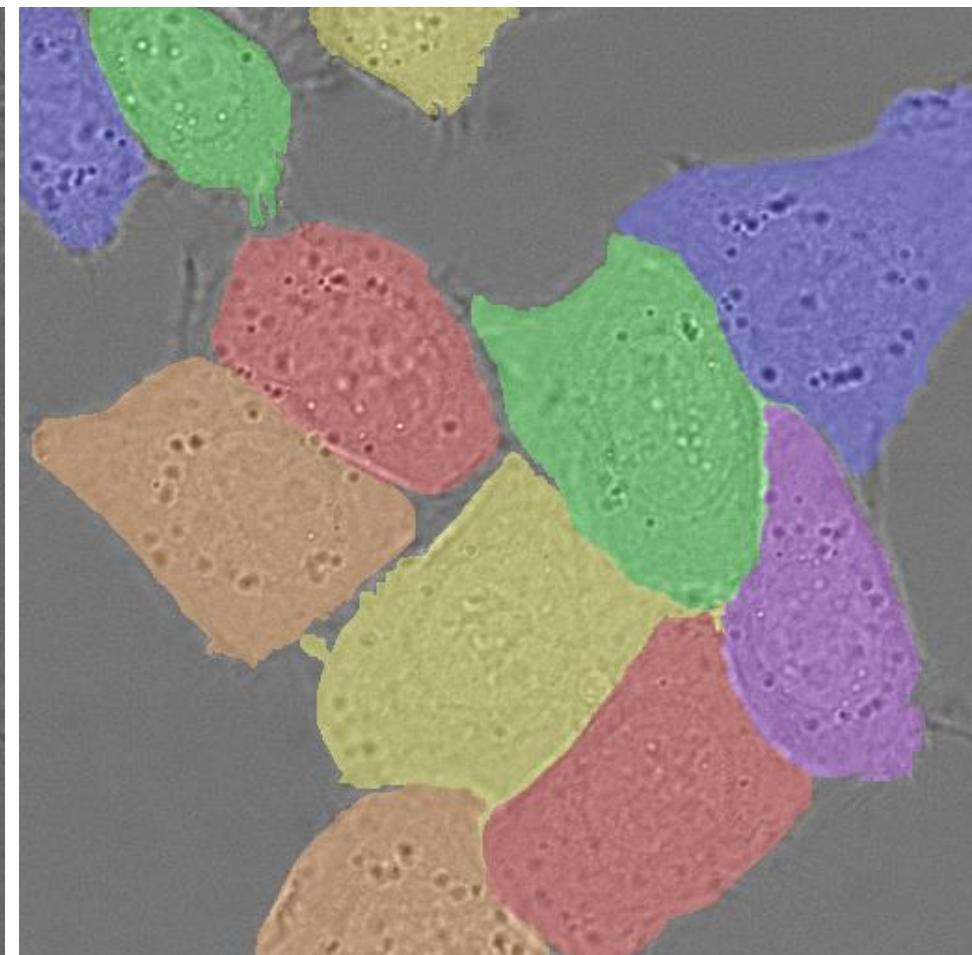
cross
entropy



Segmentation of Touching Objects of the Same Class



HeLa cells recorded with DIC microscopy

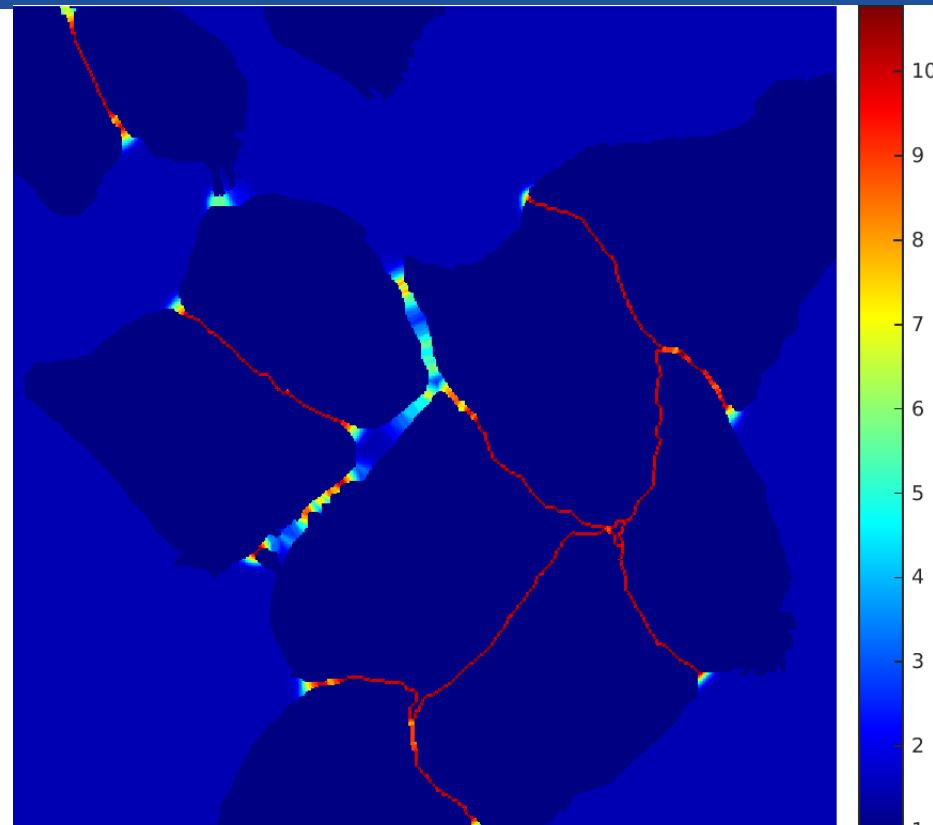


manual segmentation
(colors: different instances)

Ensure Separation of Touching Objects



Segmentation mask for training
(inserted background between
touching objects)



Loss-weight for each pixel

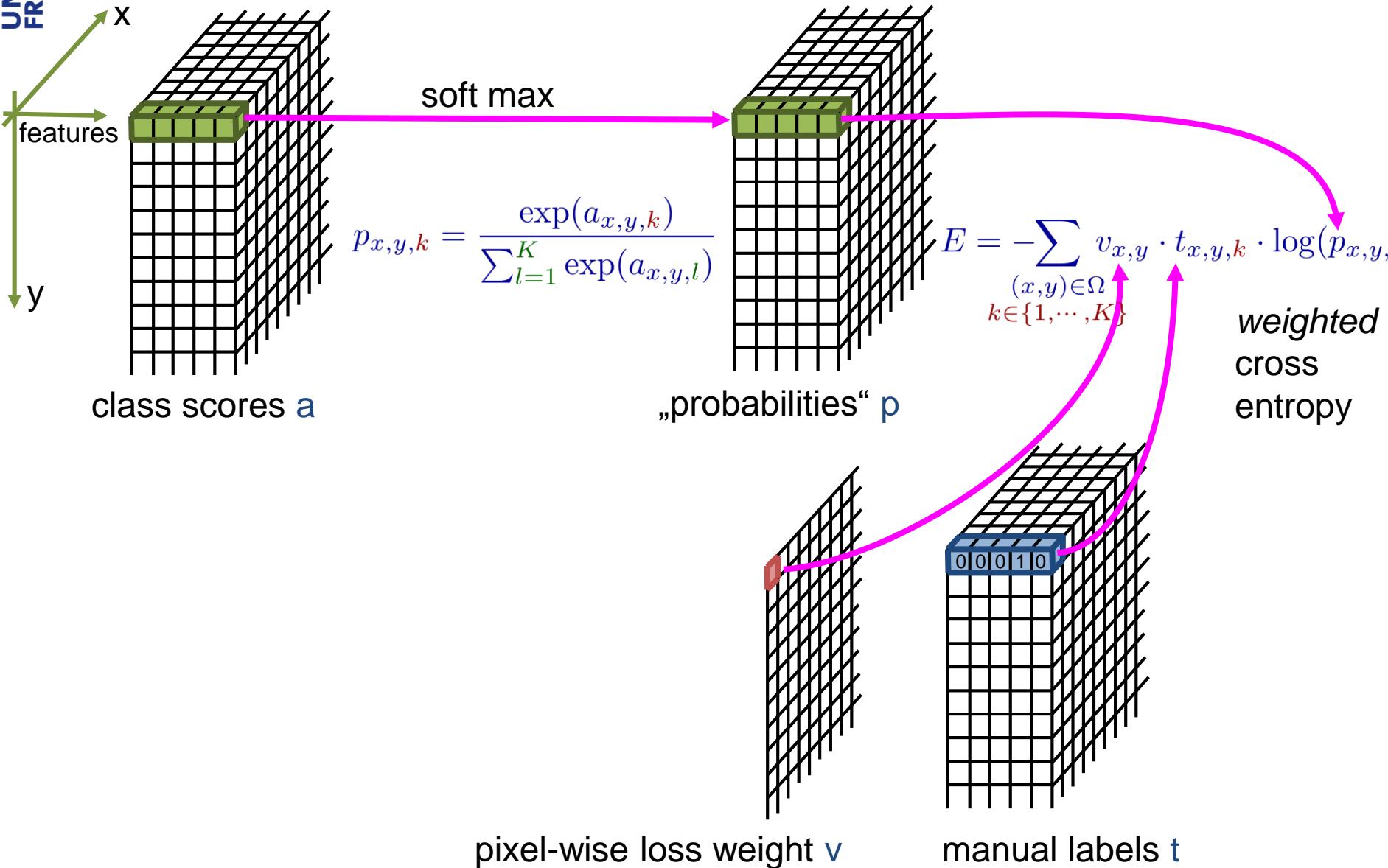
$$v(\mathbf{x}) = v_{\text{bal}}(\mathbf{x}) + v_0 \cdot \exp \left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2} \right)$$

v_{bal} : weight map to balance class frequency

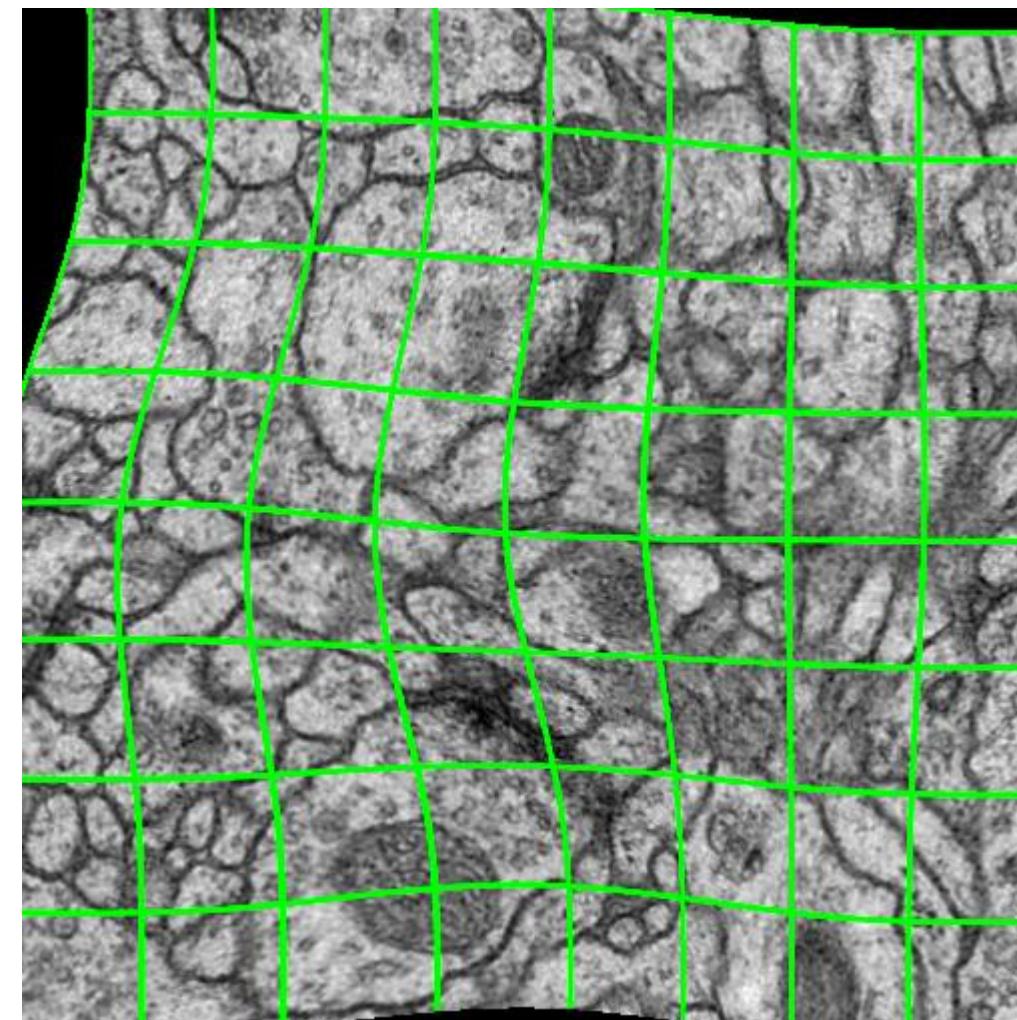
d_1 : distance to border of closest object

d_2 : distance to border of second closest object

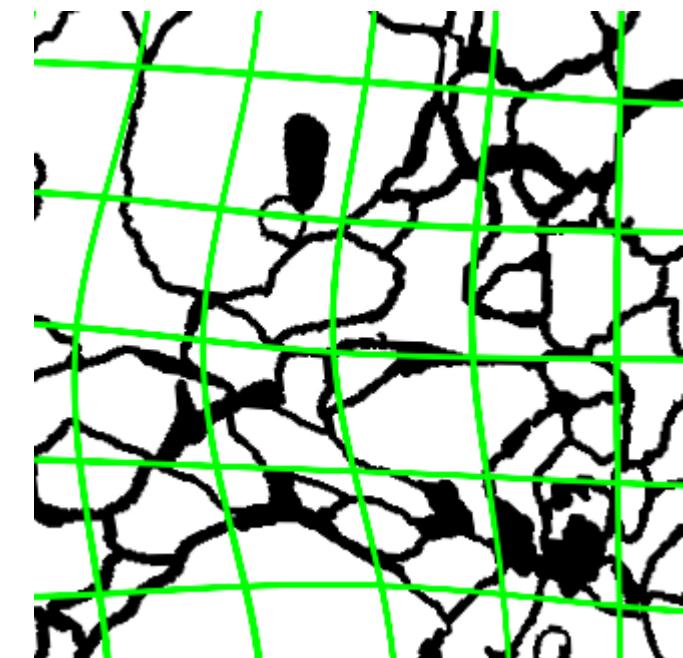
Weighted Softmax-loss layer



Augment Training Data using Deformations

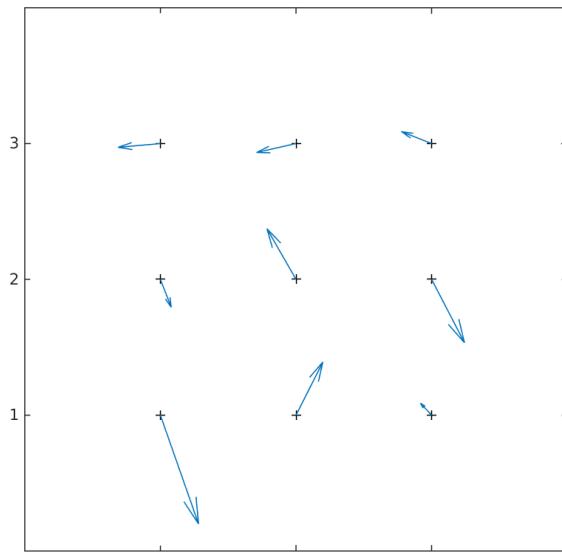


Randomly deformed image
(for visualization: no rotation, no shift, no extrapolation)

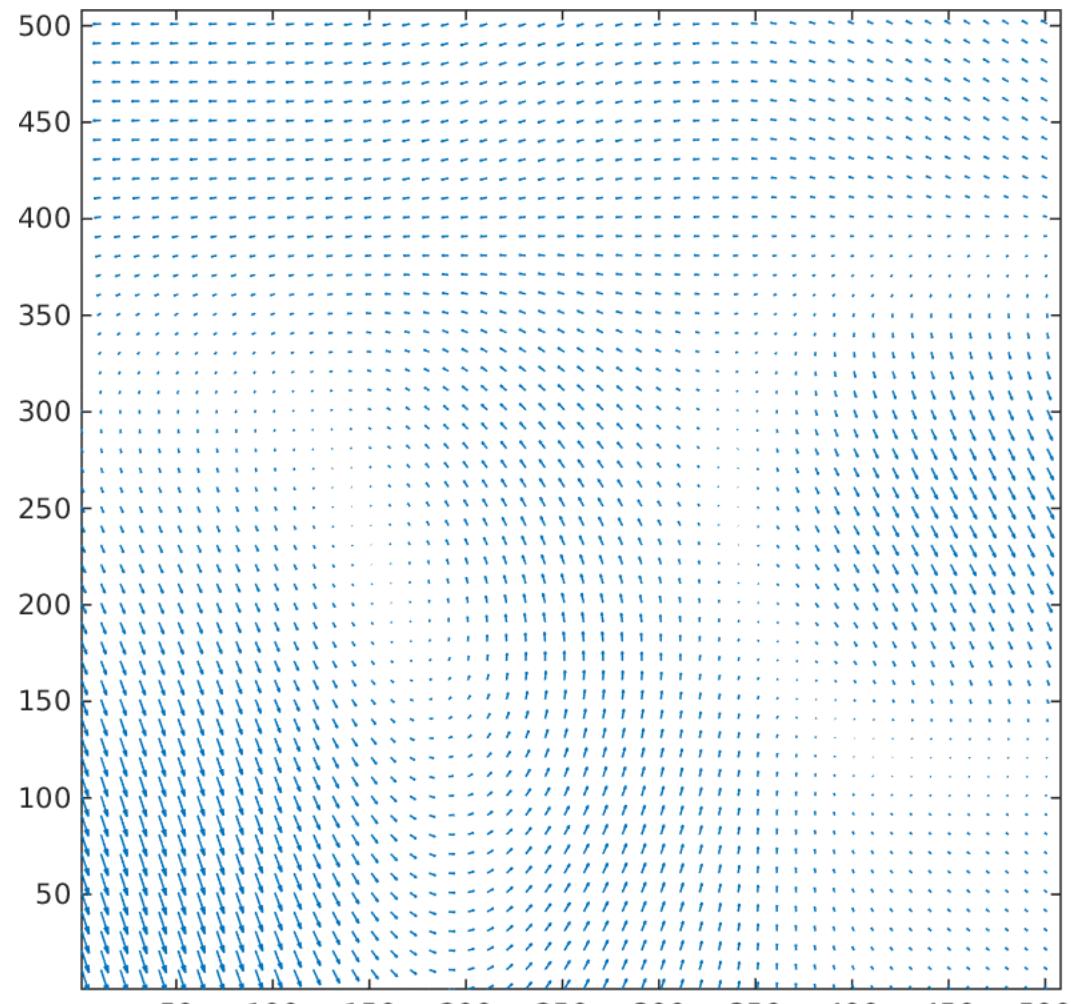


correspondingly deformed
manual labels

Creating Random Deformations



3x3 random deformation vectors



Full deformation field by **bicubic interpolation**

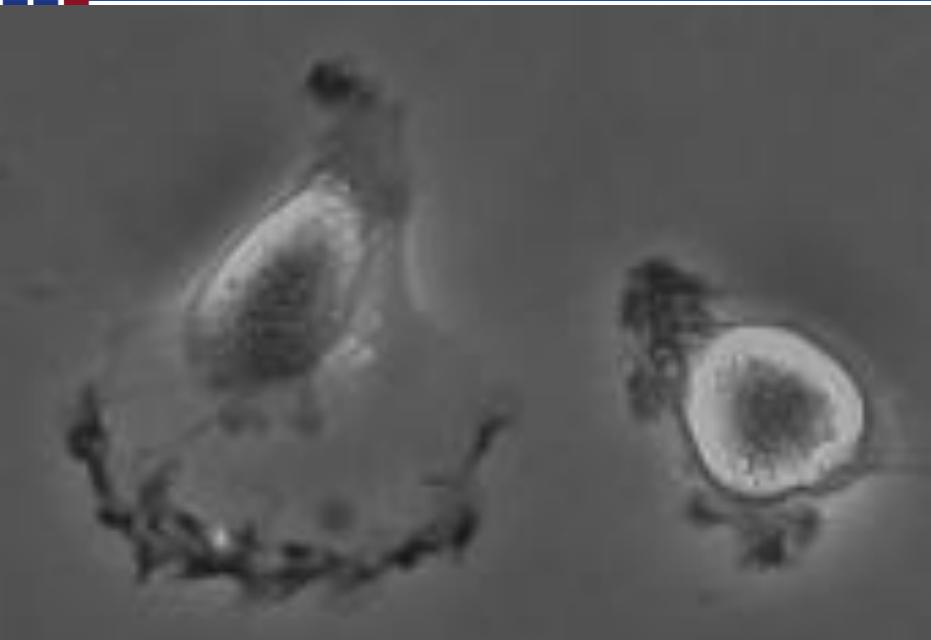
Implementation Details

- **Caffe neural network library** with the „Deconvolution“ and Cropping Layer from the FCN
- Added **$\text{sqrt}(2/N)$** initialization, and **weighted softmaxloss** layer
- **Data augmentation** and weight computation in matlab
 - Create 20,000 augmented image tiles (with corresponding segmentation and weight maps)
 - Rotation augmentation: 360° (25° for dental x-ray)
 - Random cropping position (shift augmentation)
 - Random mirroring
 - Random deformation: 3×3 normally distributed displacement vectors with 10 pixels standard deviation
 - Intensity augmentation 10%: resulting factor: $(1 + N(0,0.1))$

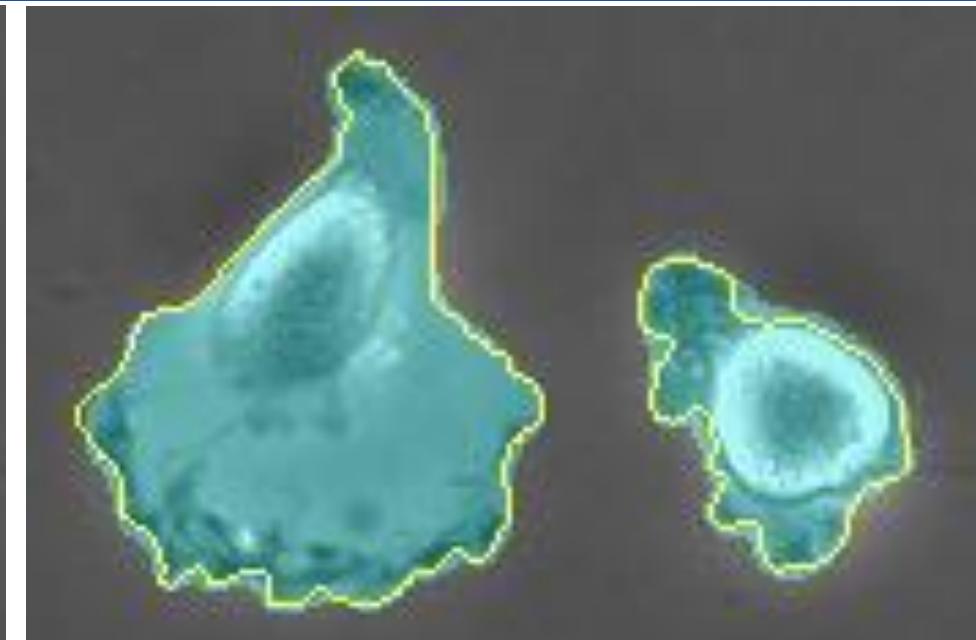
Implementation Details

- **Overlap-tile classification** in matlab (using caffe's matlab interface). **Averaging** over 7 rotated versions (EM data) or 4 mirrored versions (all other applications)
- **Training:** Stochastic gradient descent with momentum
 - „batch“-size: 1 image
 - Initial learning rate: 0.001
 - Momentum: 0.99 (**very high to virtually increase batch size**),
 - All 20,000 iterations, reduce learning rate by 1/10
 - 60,000 iterations total
 - Training time: approx 10 hours on a Nvidia Titan GPU. **First meaningful results already after a few 1000 iterations (half an hour or less)**

ISBI cell tracking challenge: dataset PhC-U373



Glioblastoma-astrocytoma U373 cells on a polyacrylimide substrate
Phase contrast microscopy

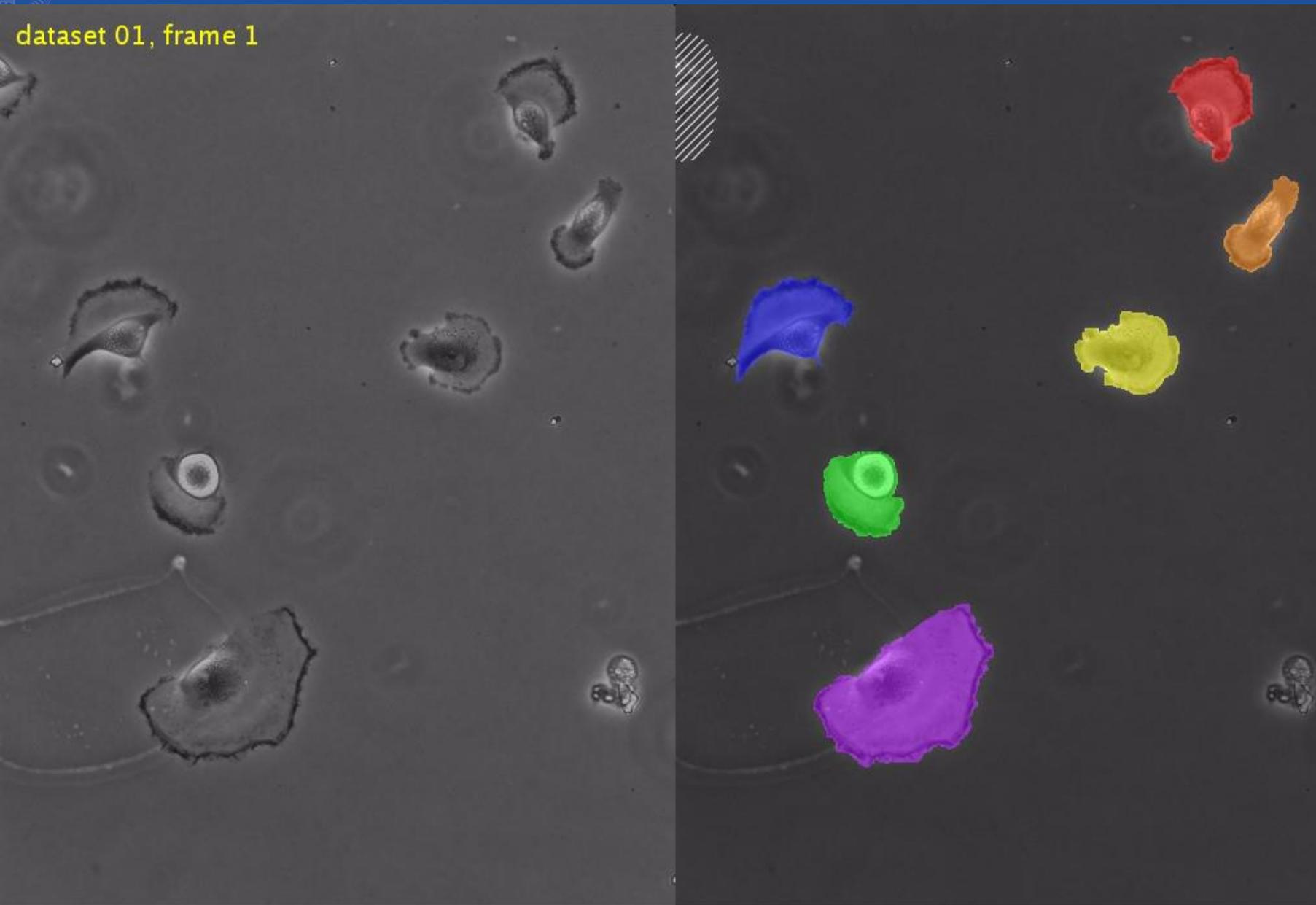


Cyan: segmentation by u-net
Yellow borders: our manual ground truth

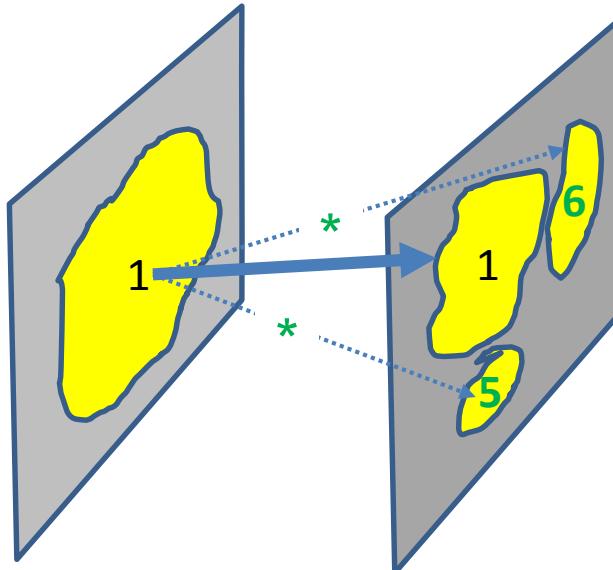
- Strong shape variations
- Weak outer borders, strong irrelevant inner borders
- Cytoplasm has same structure like background

Data provided by Dr. Sanjay Kumar. Department of Bioengineering
University of California at Berkeley. Berkeley CA (USA)

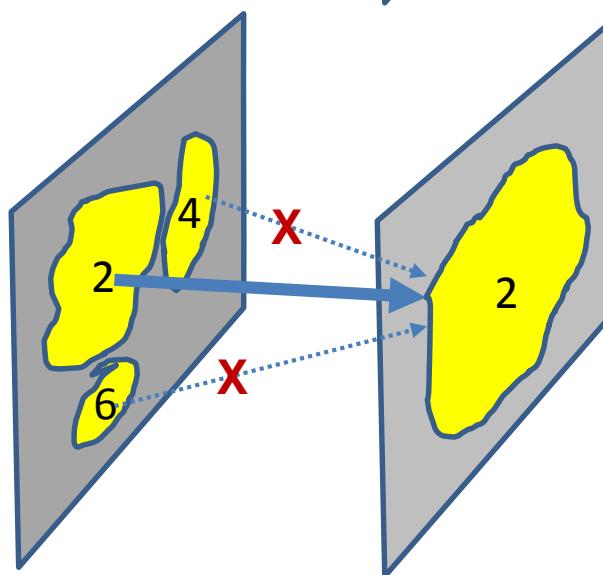
PhC-U373: Complete Training Set



Tracking

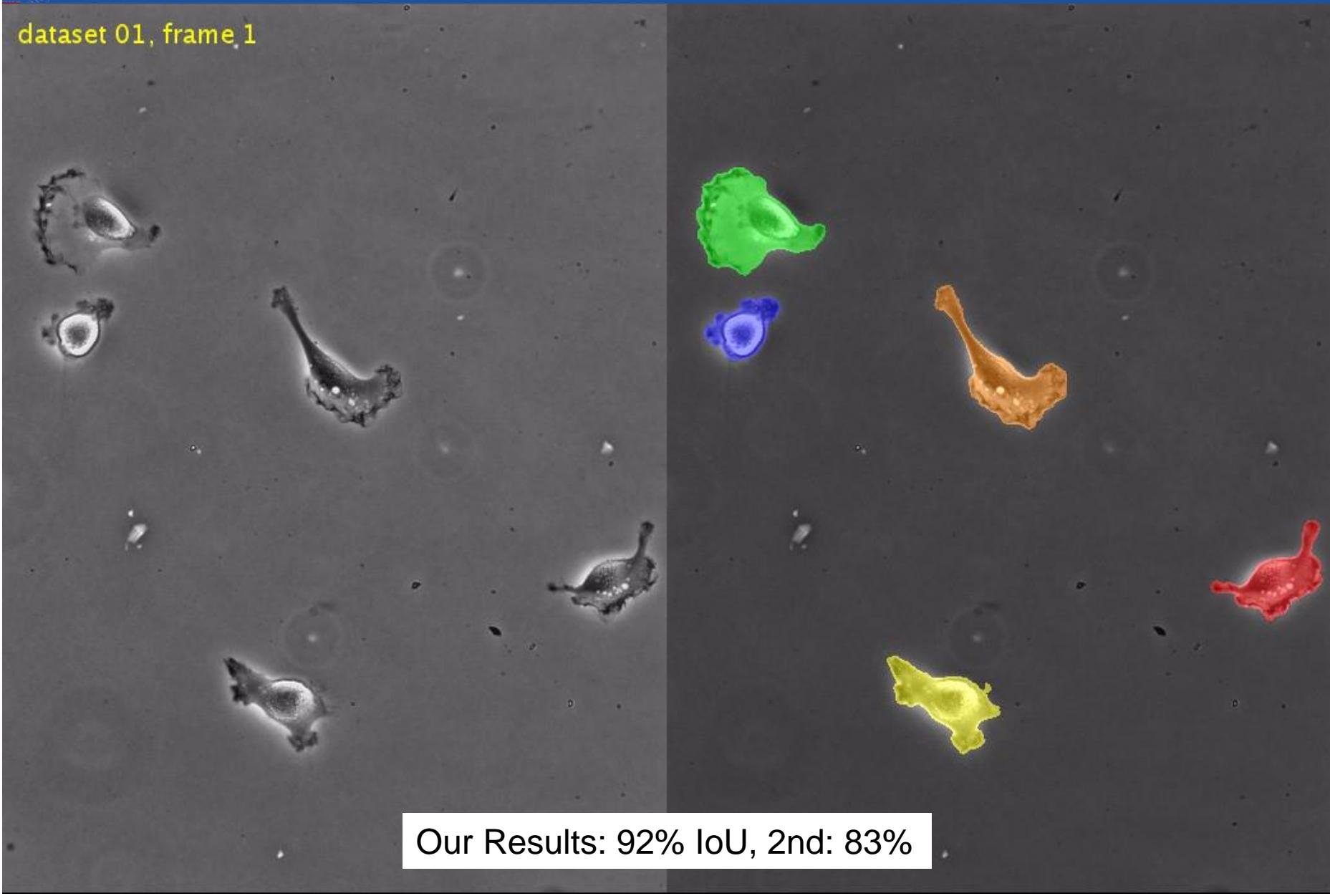


- Propagate Labels to overlapping Segments
- Resolve **one-to-many** correspondences:
 - Propagate label to max. IOU
 - Invent new labels
- Resolve **many-to-one** correspondences
 - Take label from max. IOU
 - Kill other labels



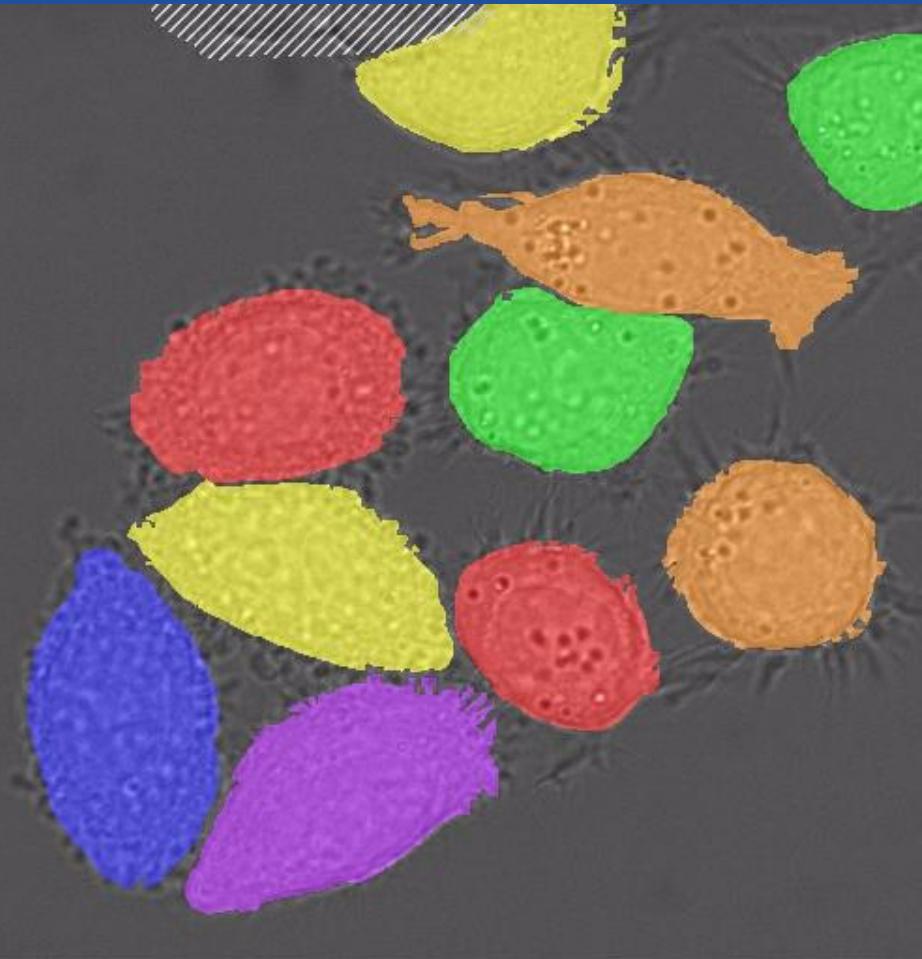
PhC-U373:

dataset 01, frame 1



DIC-HeLa: Complete Training Set

dataset 01, frame 1



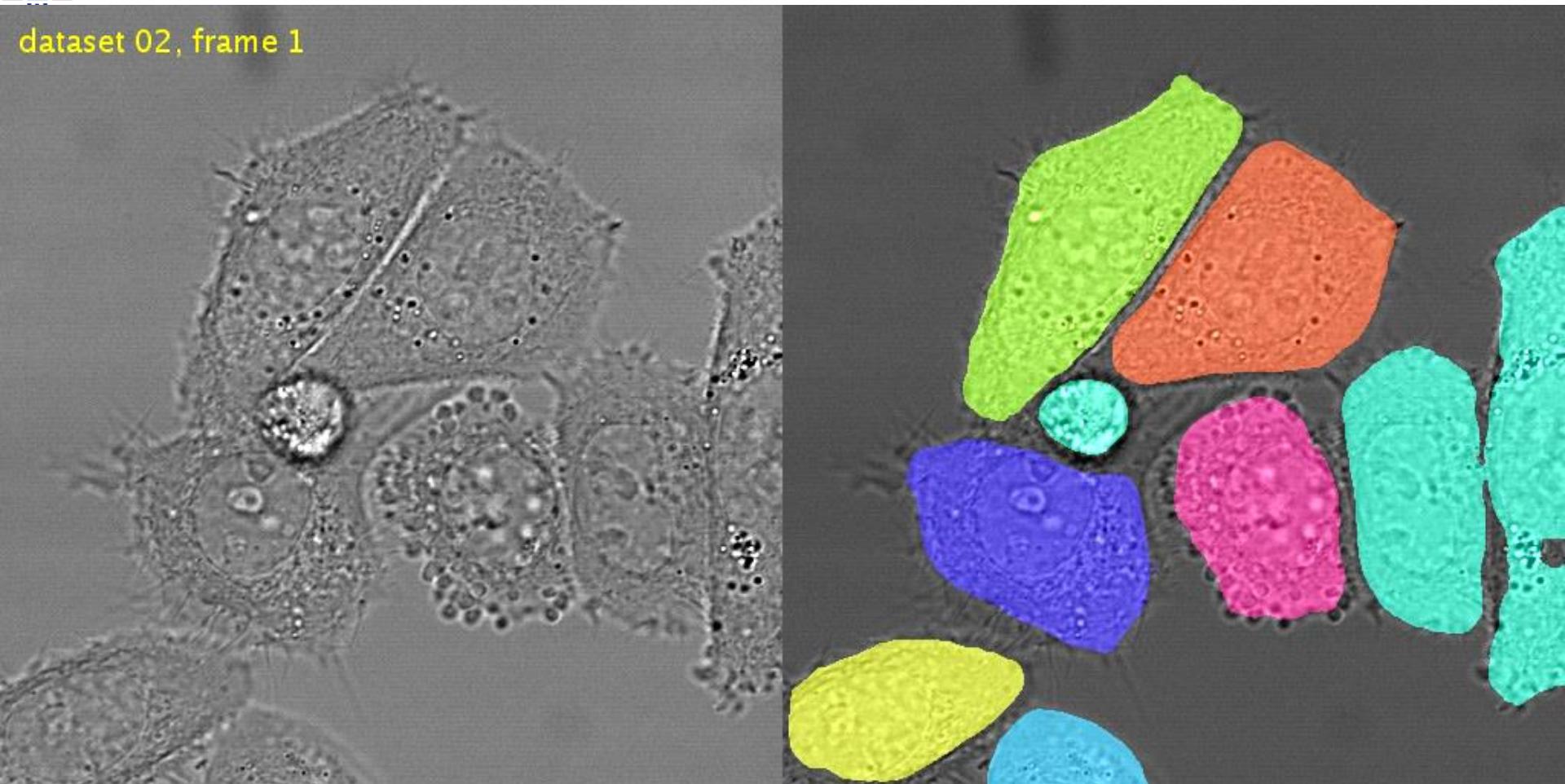
HeLa cells on a flat glass, Differential Interference Contrast (DIC) microscopy

- Touching and overlapping cells (more than one layer)
- partially invisible borders
- cells leave focal plane

Data provided by Dr. Gert van Cappellen,
Erasmus Medical Center, Rotterdam, The Netherlands

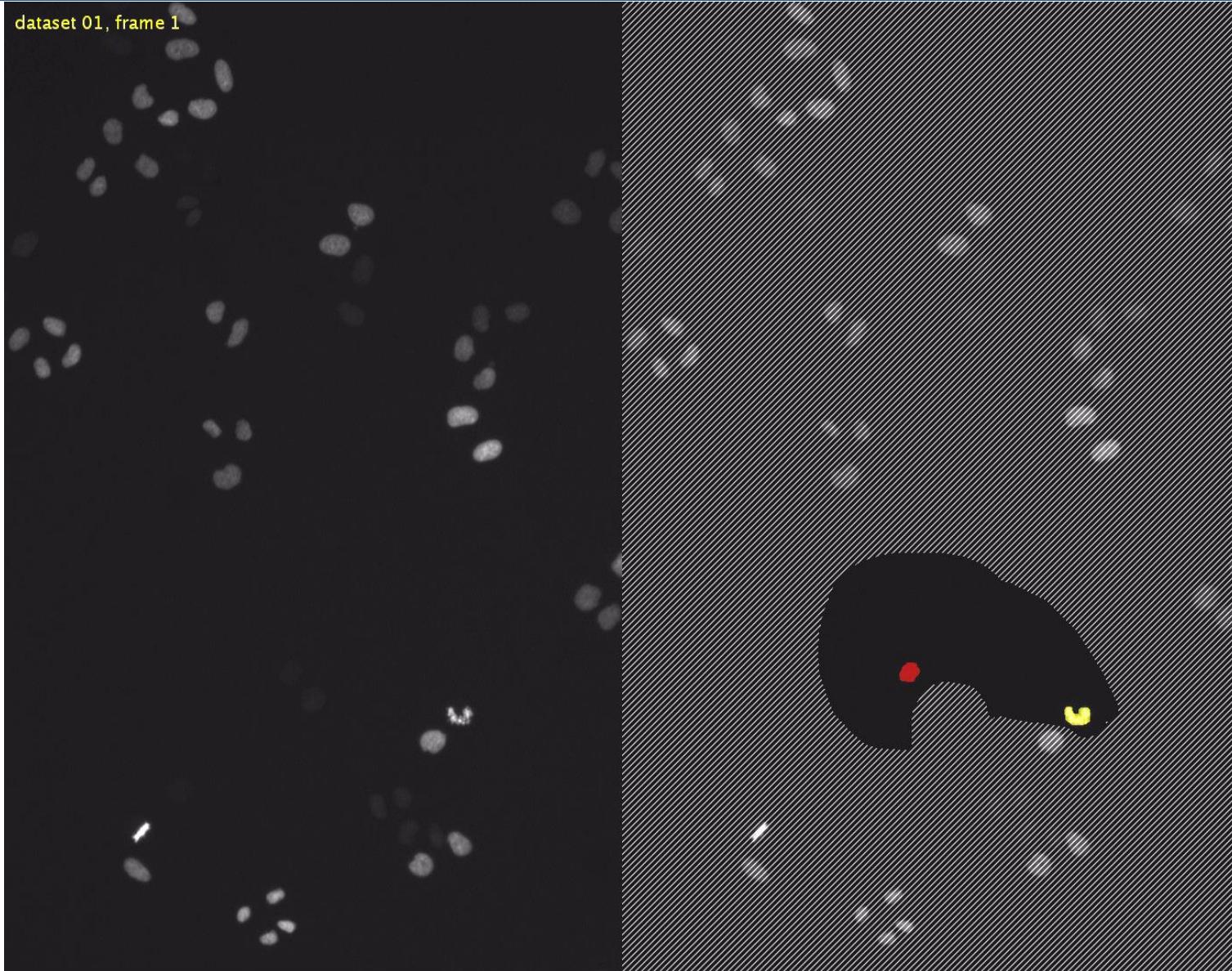
DIC-HeLa:

dataset 02, frame 1



Our Results: 77.6%, 2nd: 46.0%

Fluo-HeLa: Training



HeLa cells stably expressing H2b-GFP

Data provided by Mitocheck Consortium

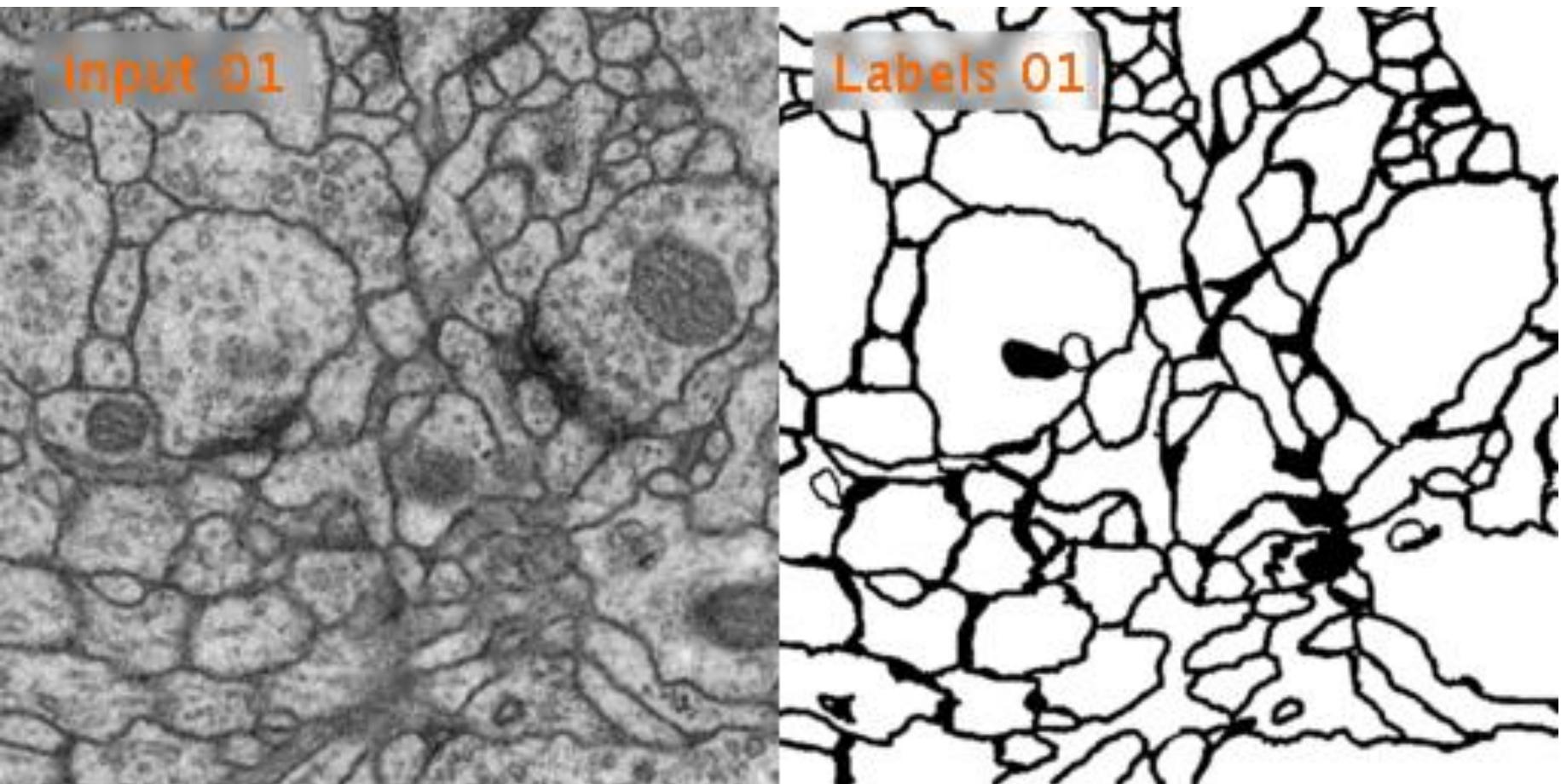
Fluo-HeLa:

dataset 01, frame 1



Our Results: 90%, 2nd: 89%

Experiments: Neuronal structures in EM

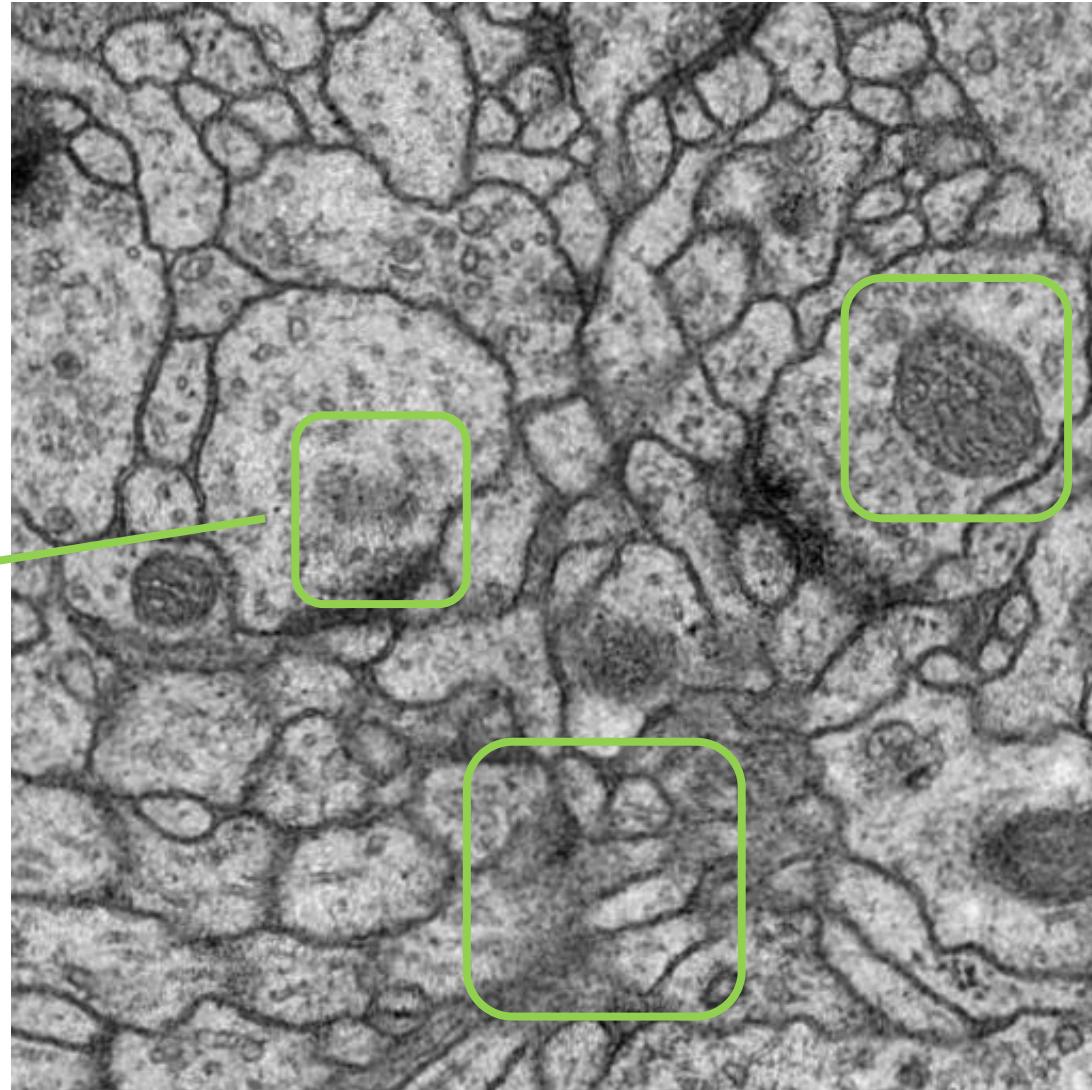


Input images

Manual labels
membranes: black,
rest (mainly cytoplasm): white

Challenges in this Data Set

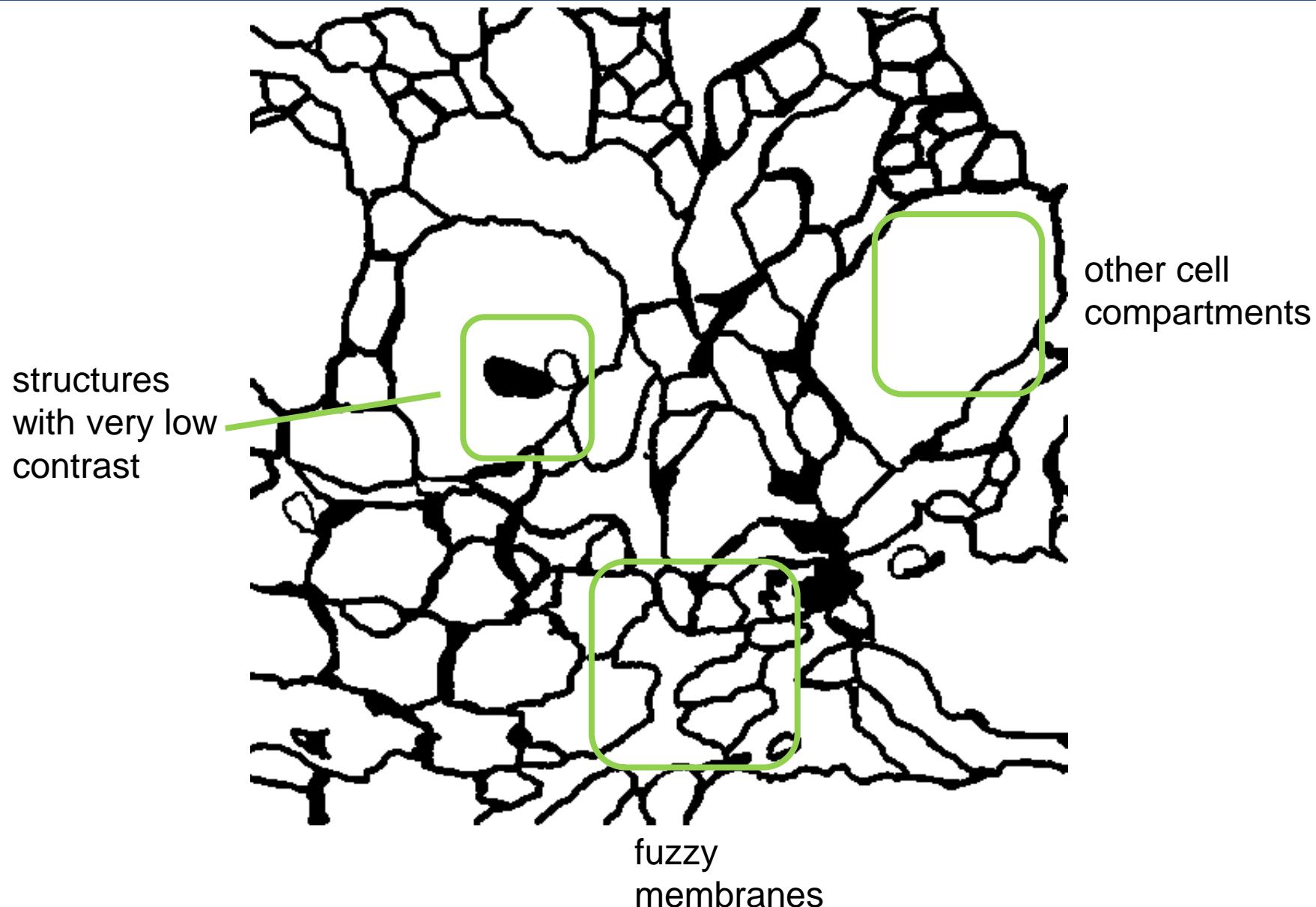
structures
with very low
contrast



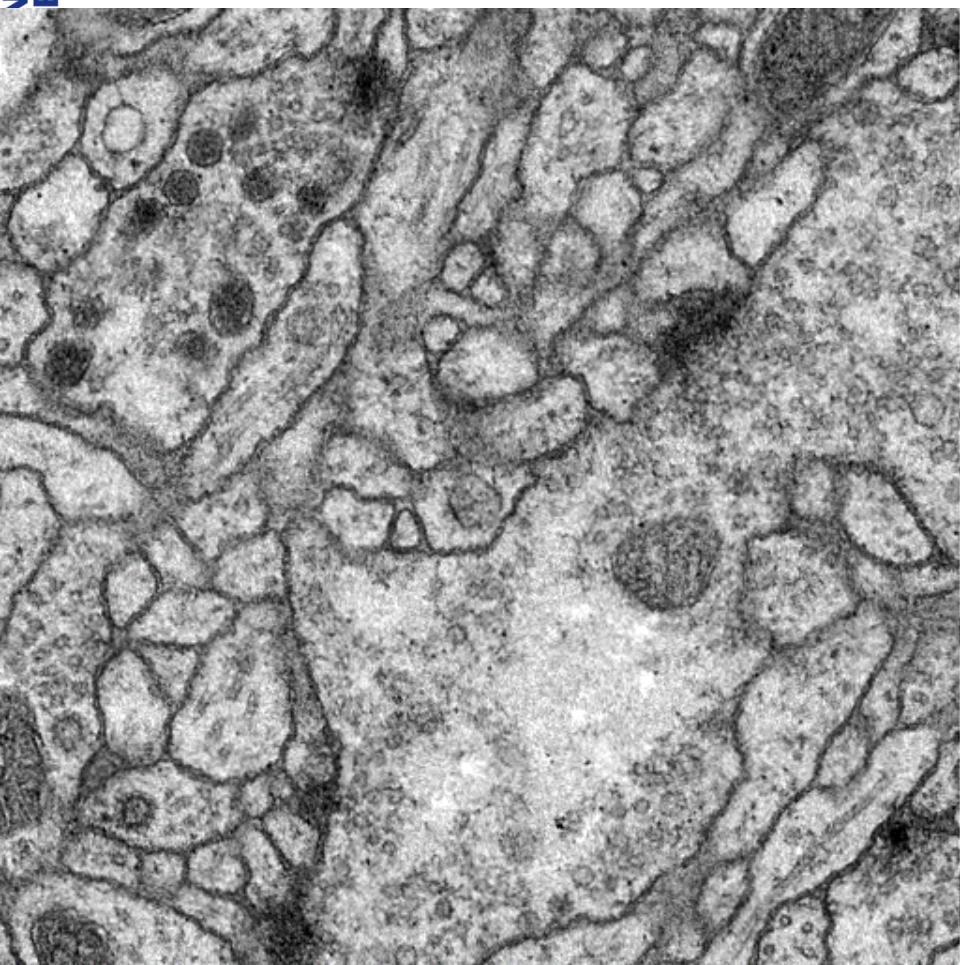
fuzzy
membranes

other cell
compartments

Challenges in this Data Set



Our Results



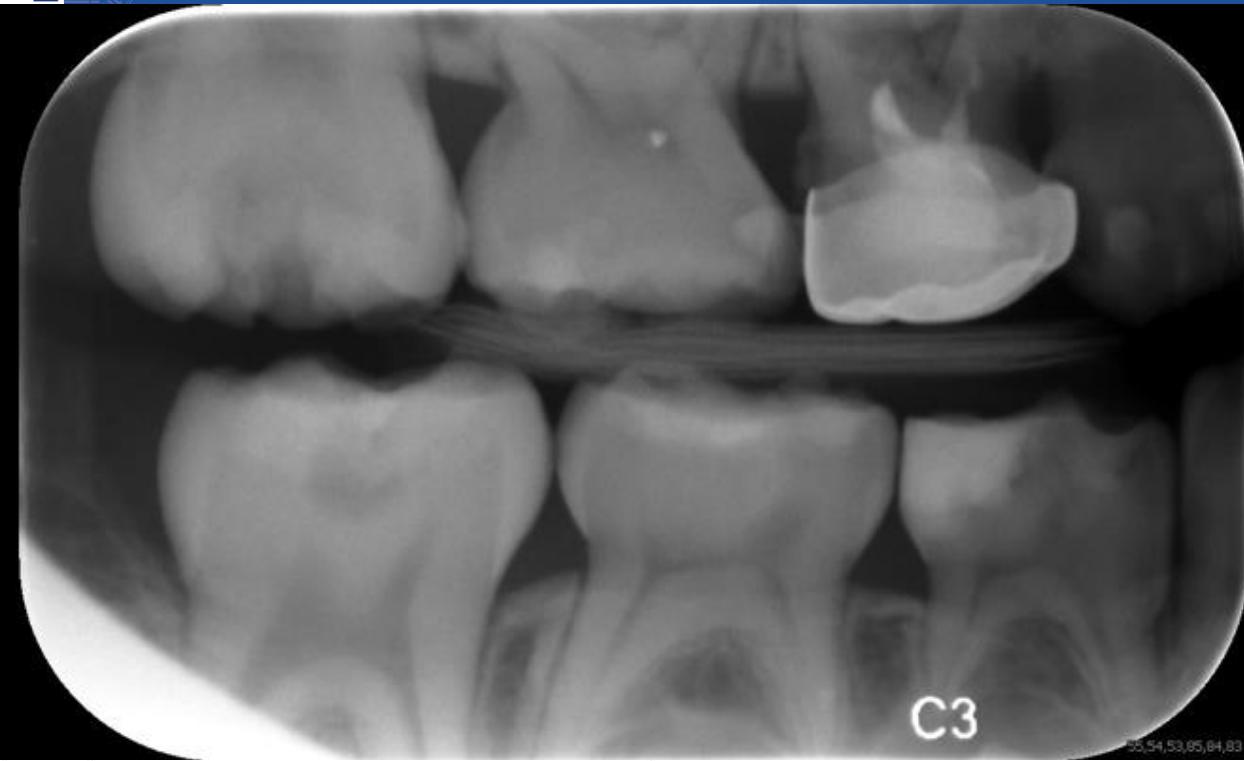
Input image



Membrane score

Our result: 0.000353 warping error (**New best score** at submission march 6th, 2015)
Sliding-window CNN: 0.000420

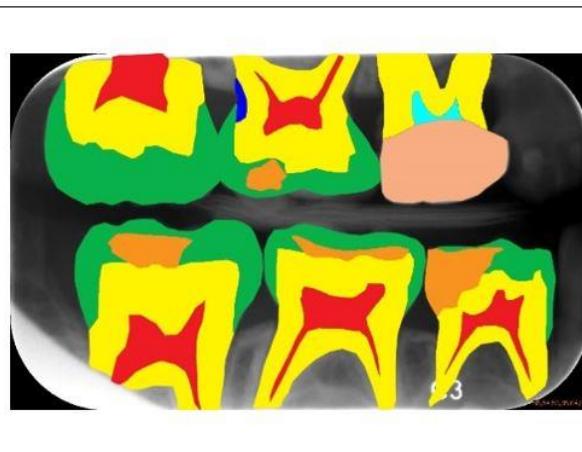
Segmentation of Dental X-Ray images



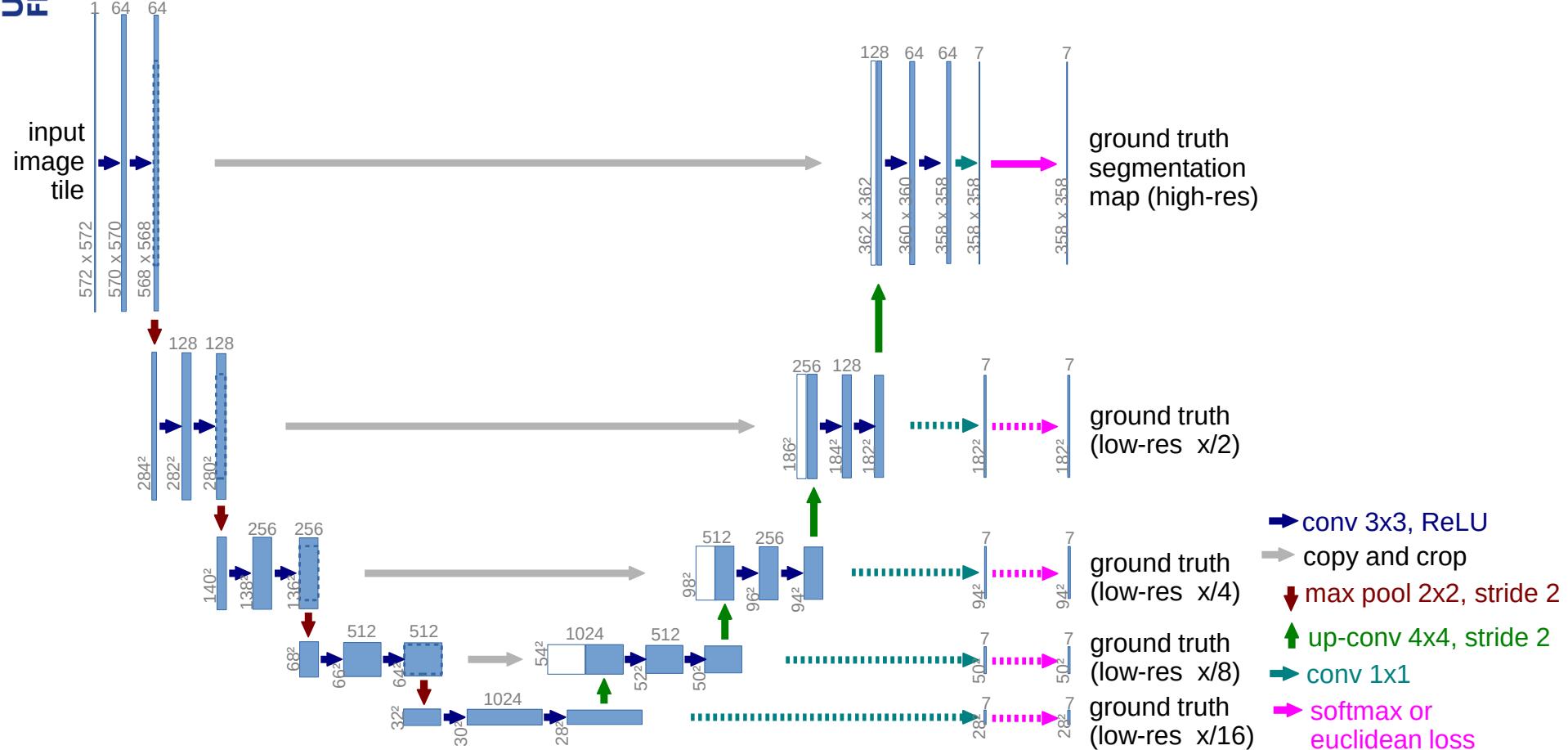
Challenges:

- subtle gray value variations at specific positions define the classes
- partial teeth not labelled

No.	Important Properties Parts
1	caries (blue color)
2	enamel (green color)
3	dentin (yellow color)
4	pulp (red color)
5	crown (skin color)
6	restoration (orange color)
7	root canal treatment (cyan color)



Additional Low-Resolution Loss Layers

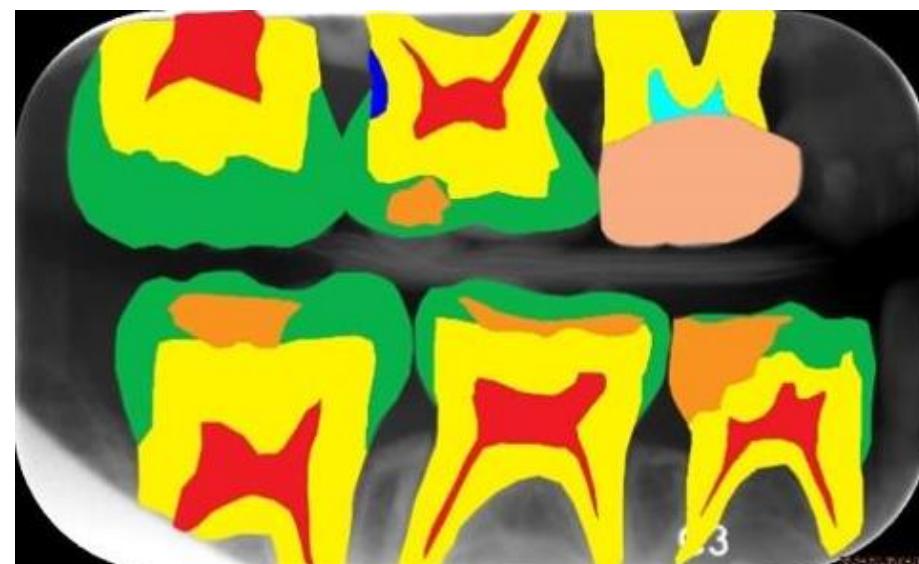


- Try to guide the training in the lower layers

Results with different Training Schemes



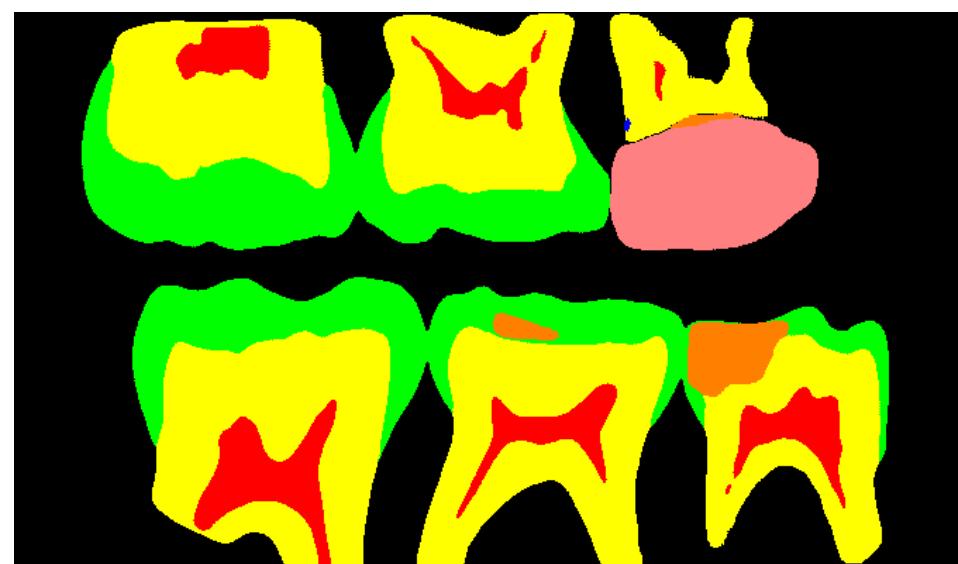
raw image



Ground truth segmentation

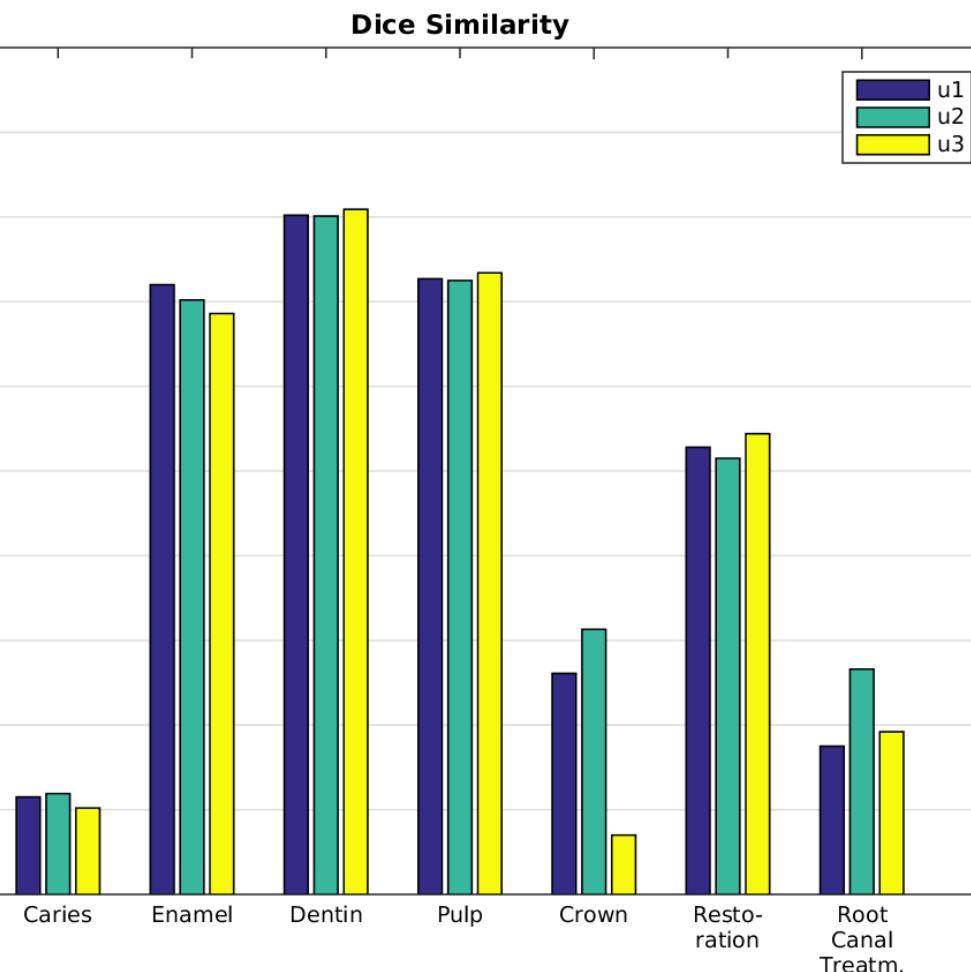


u1: only high-res loss layer



u2: add. low-res softmax loss layers

Quantitative Evaluation from Challenge Organizers

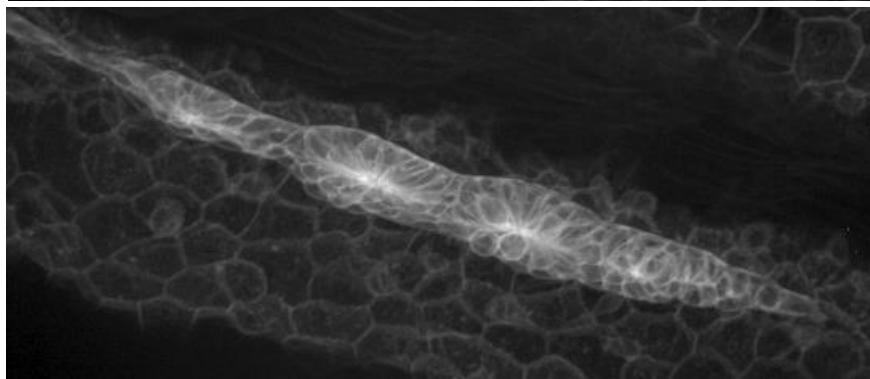
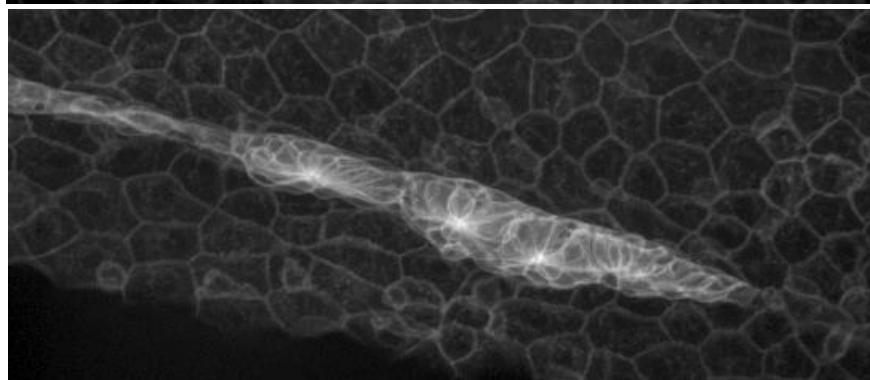
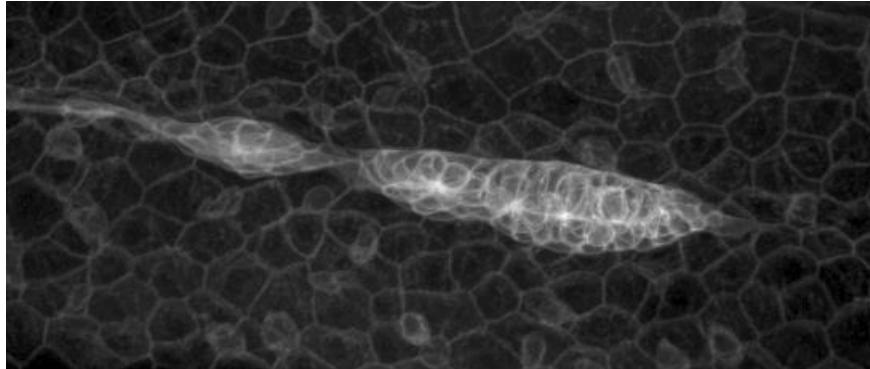


u1: high-res loss only
u2: add. low-res softmax loss
u3: add. low-res Euclidean loss

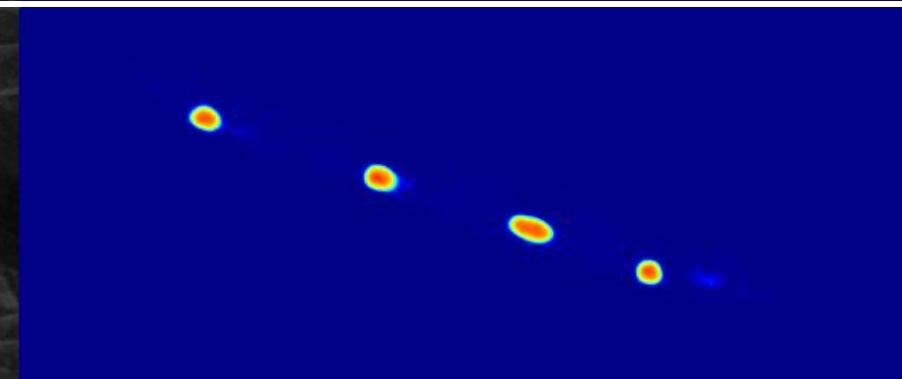
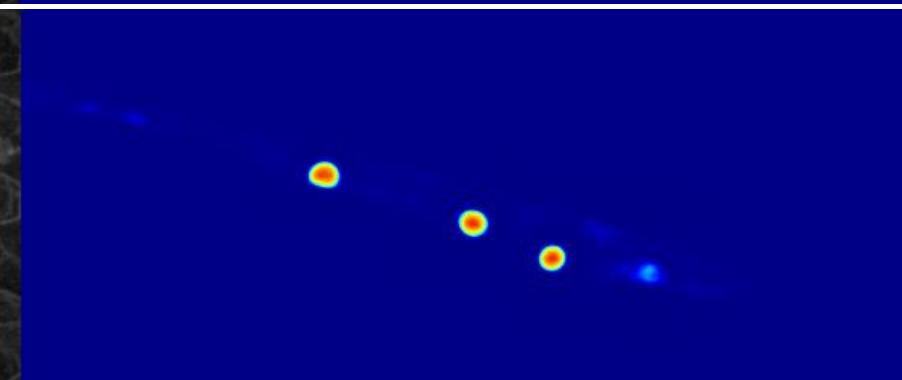
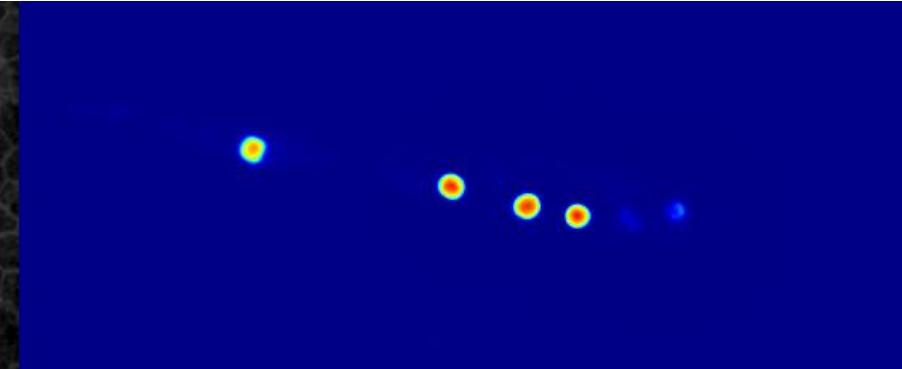
Phase 1 (pre-Conference competition):
Our result: **56.7%**
Second best: 32.2%

Phase 2 (On-site competition)
Our result: **52.5%**
Second best: 28.7%

Detection of Cell Rosettes in the Zebrafish Primordium



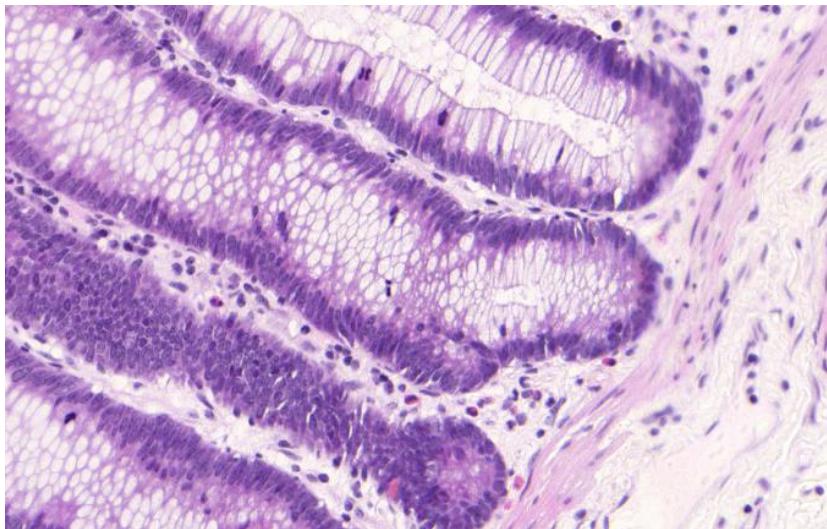
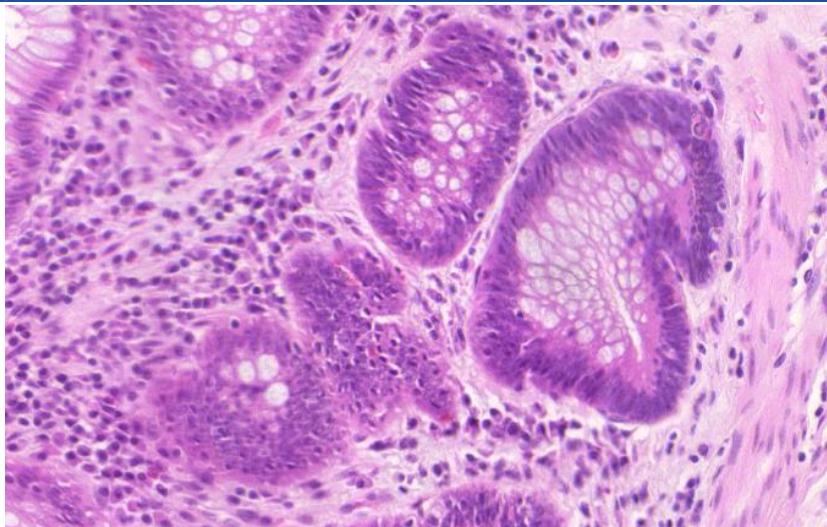
Raw images



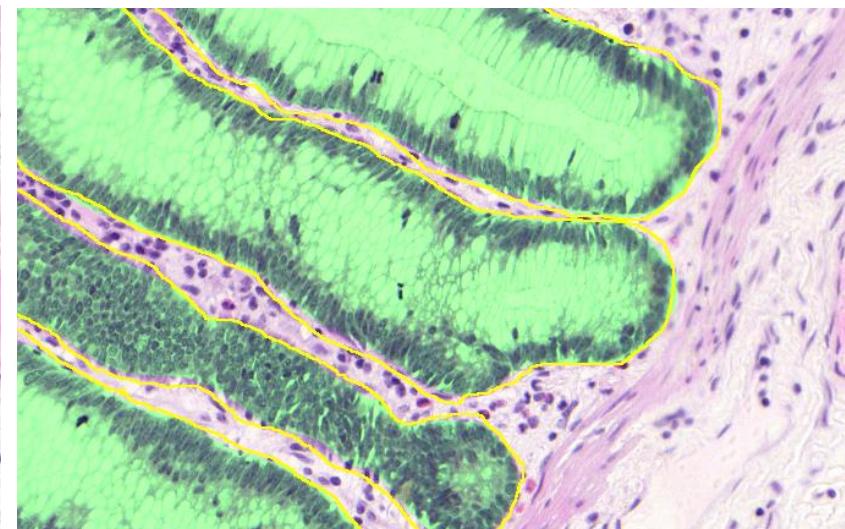
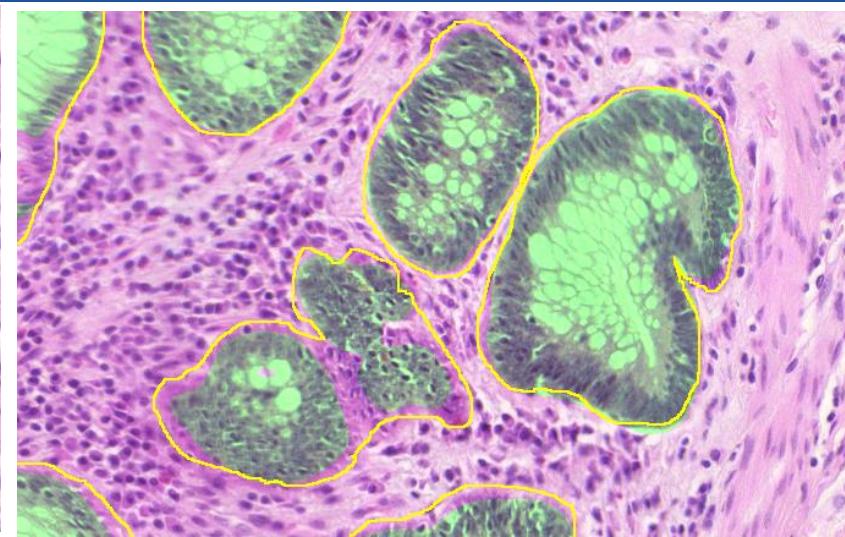
Detection score maps

[Robert Bensch, collaboration with Virginie Lecaudey, Biology, Uni Freiburg]

Segmentation of Histopathological Images



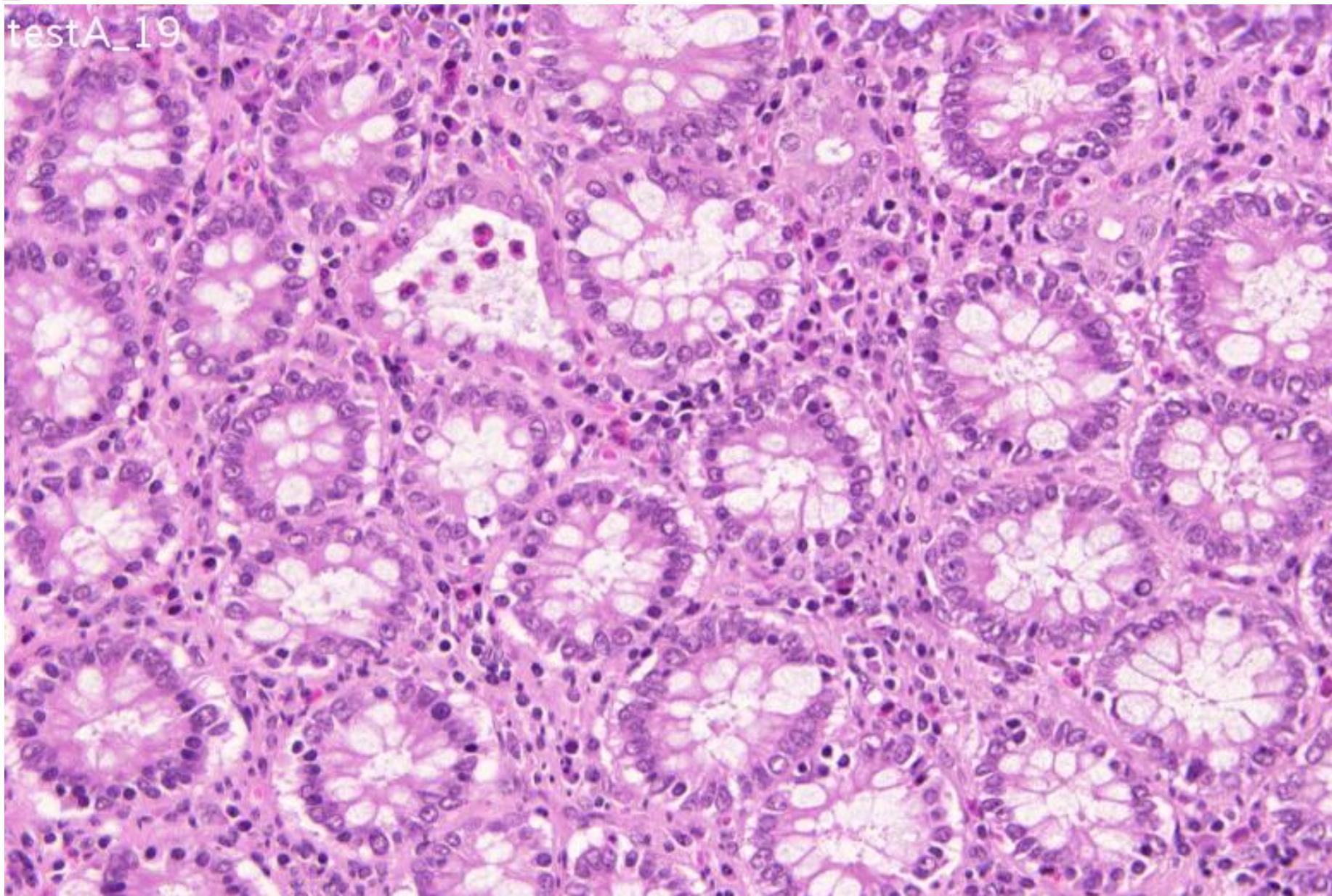
Glands in histology images



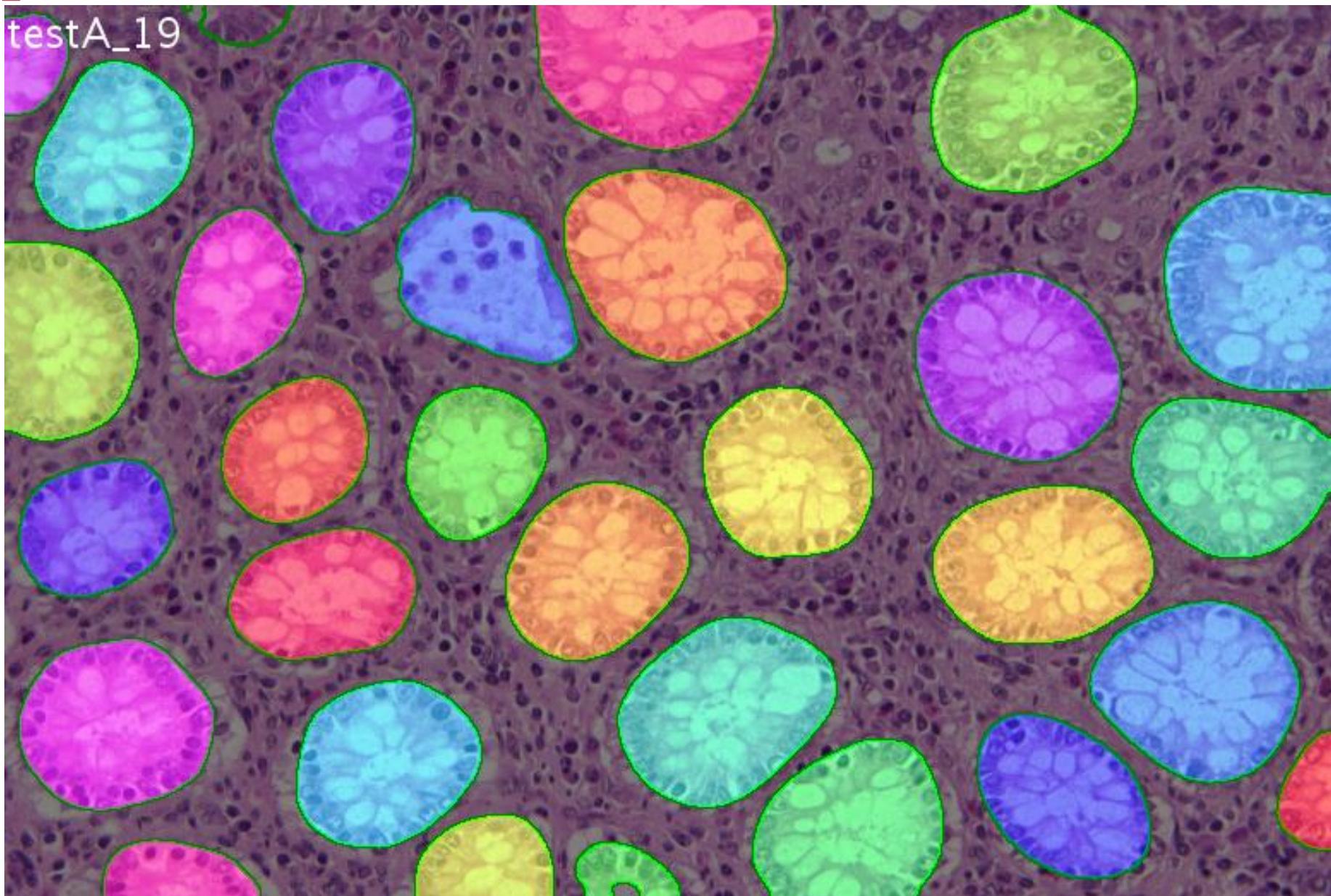
green mask: u-net result;
yellow border: ground truth annotation

[Anton Böhm, data from GlaS@MICCAI'2015: Gland Segmentation Challenge Contest]

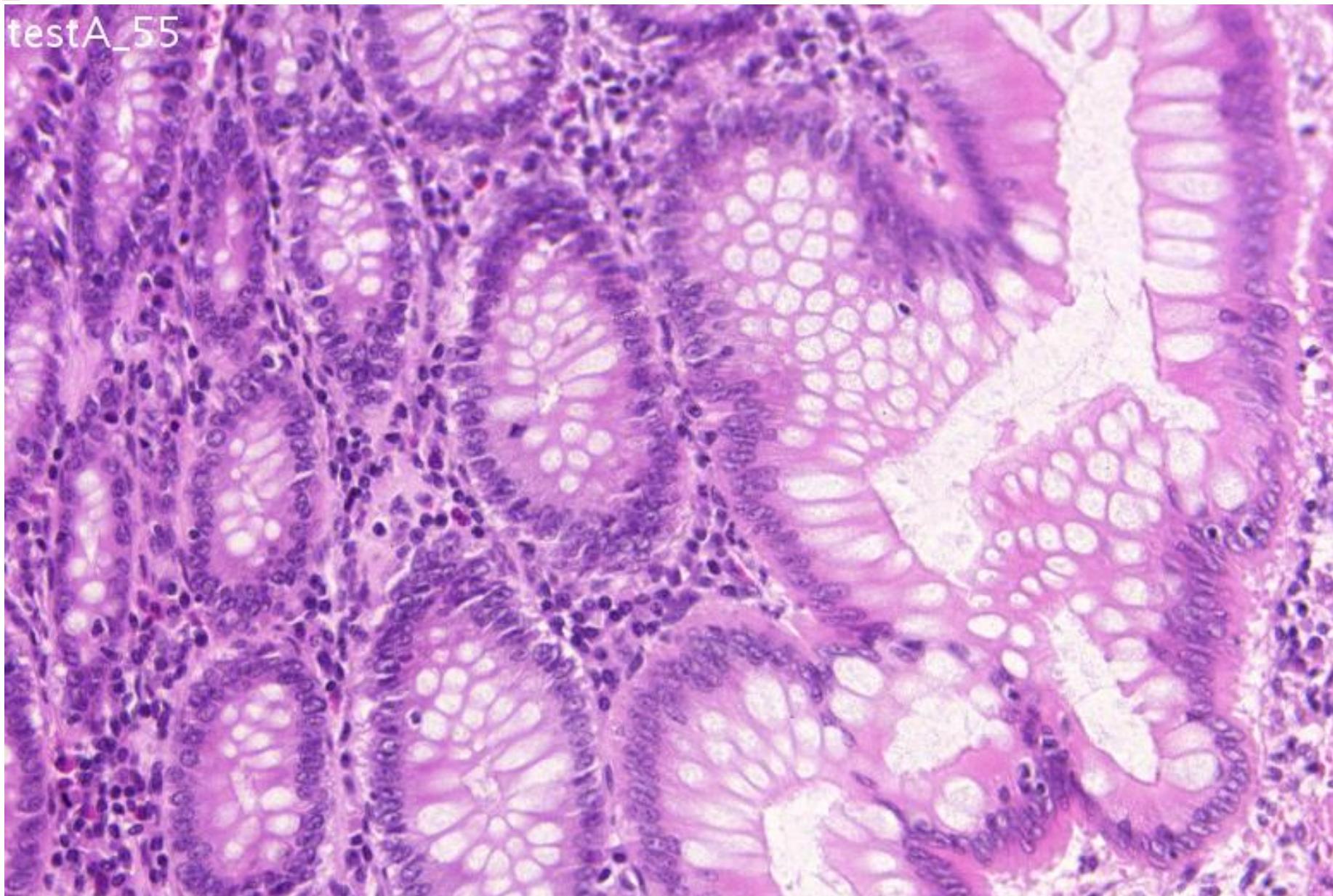
Qualitative Results: Glas Challenge Testset A



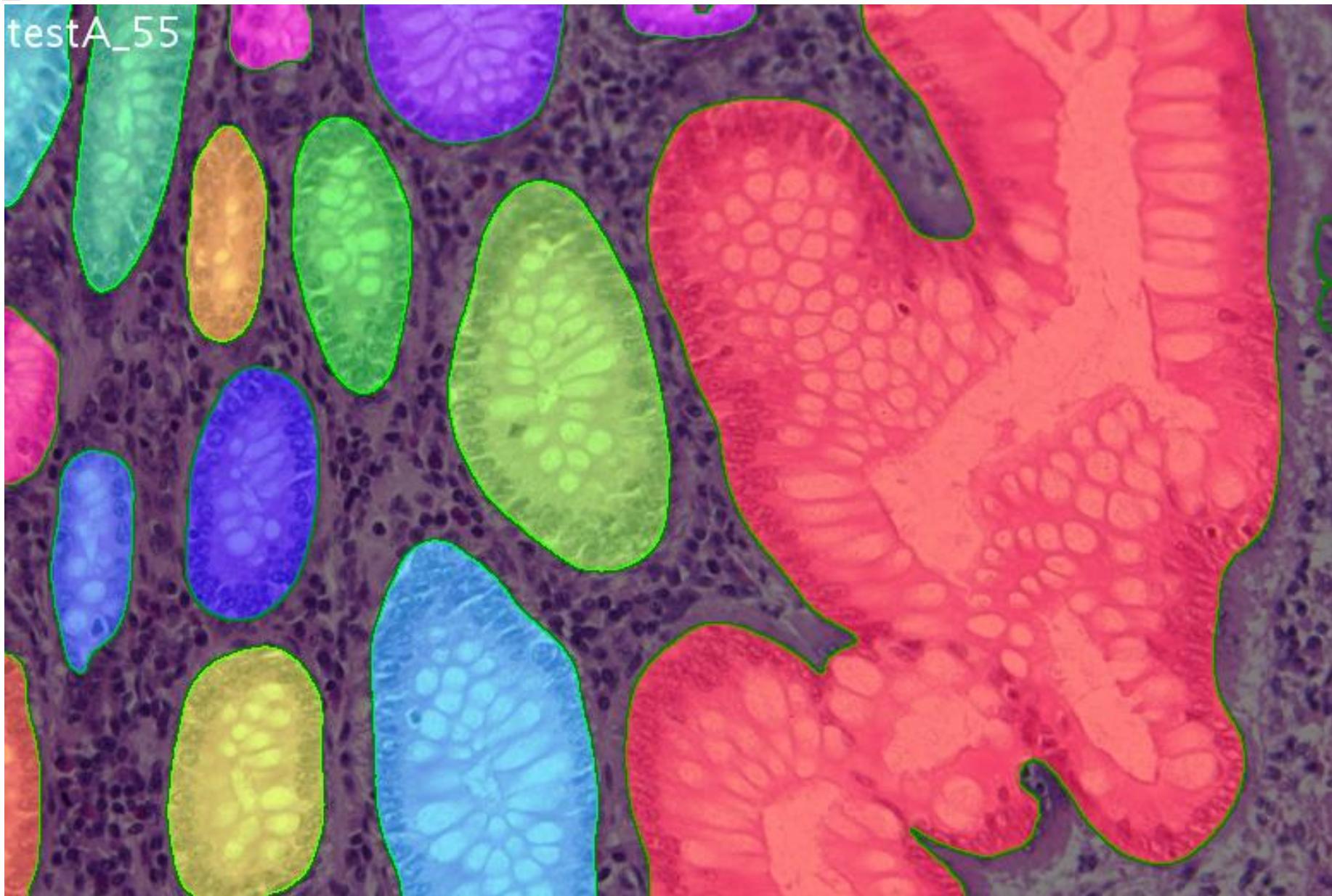
Qualitative Results: Glas Challenge Testset A



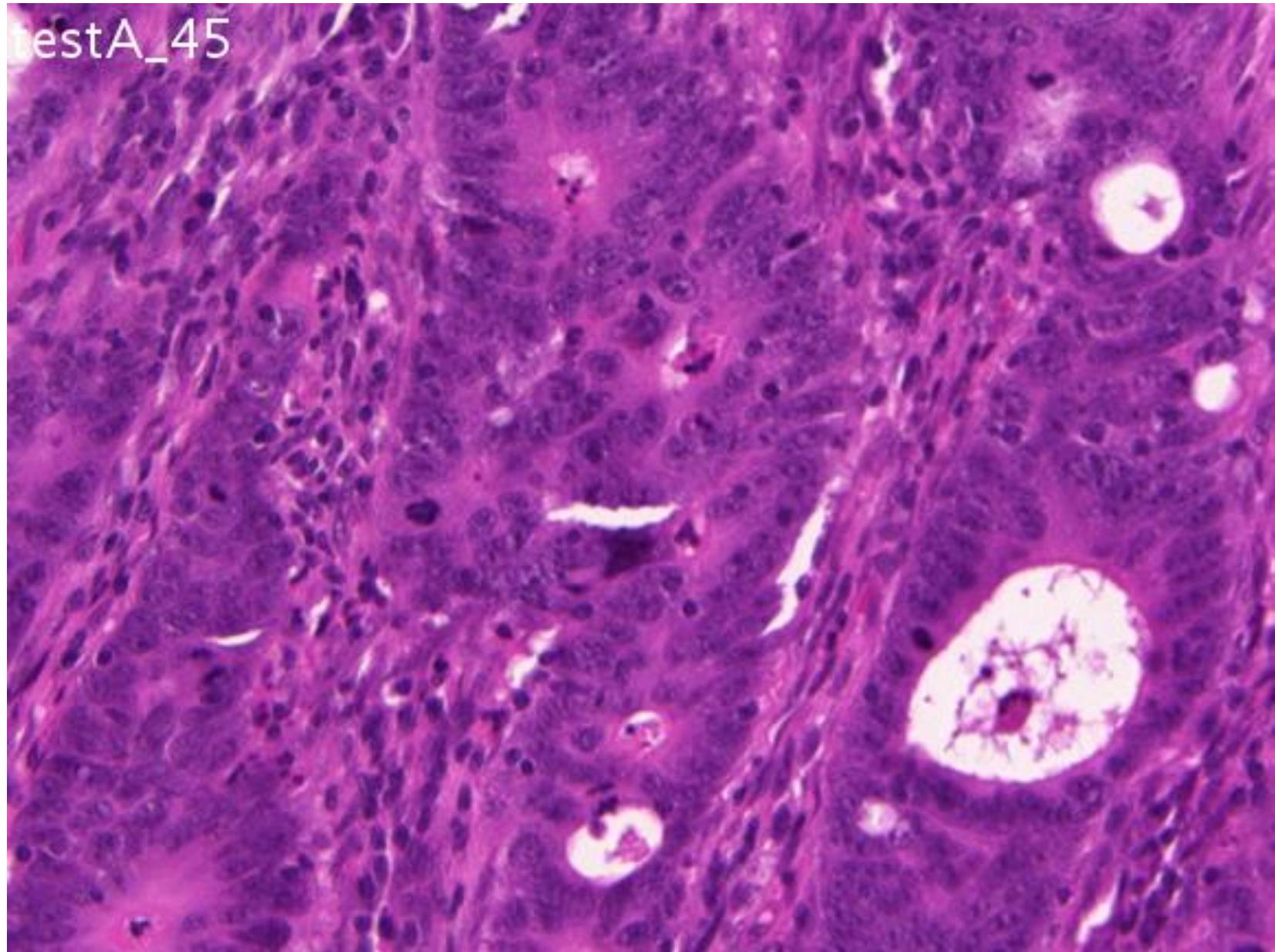
Qualitative Results: Glas Challenge Testset A



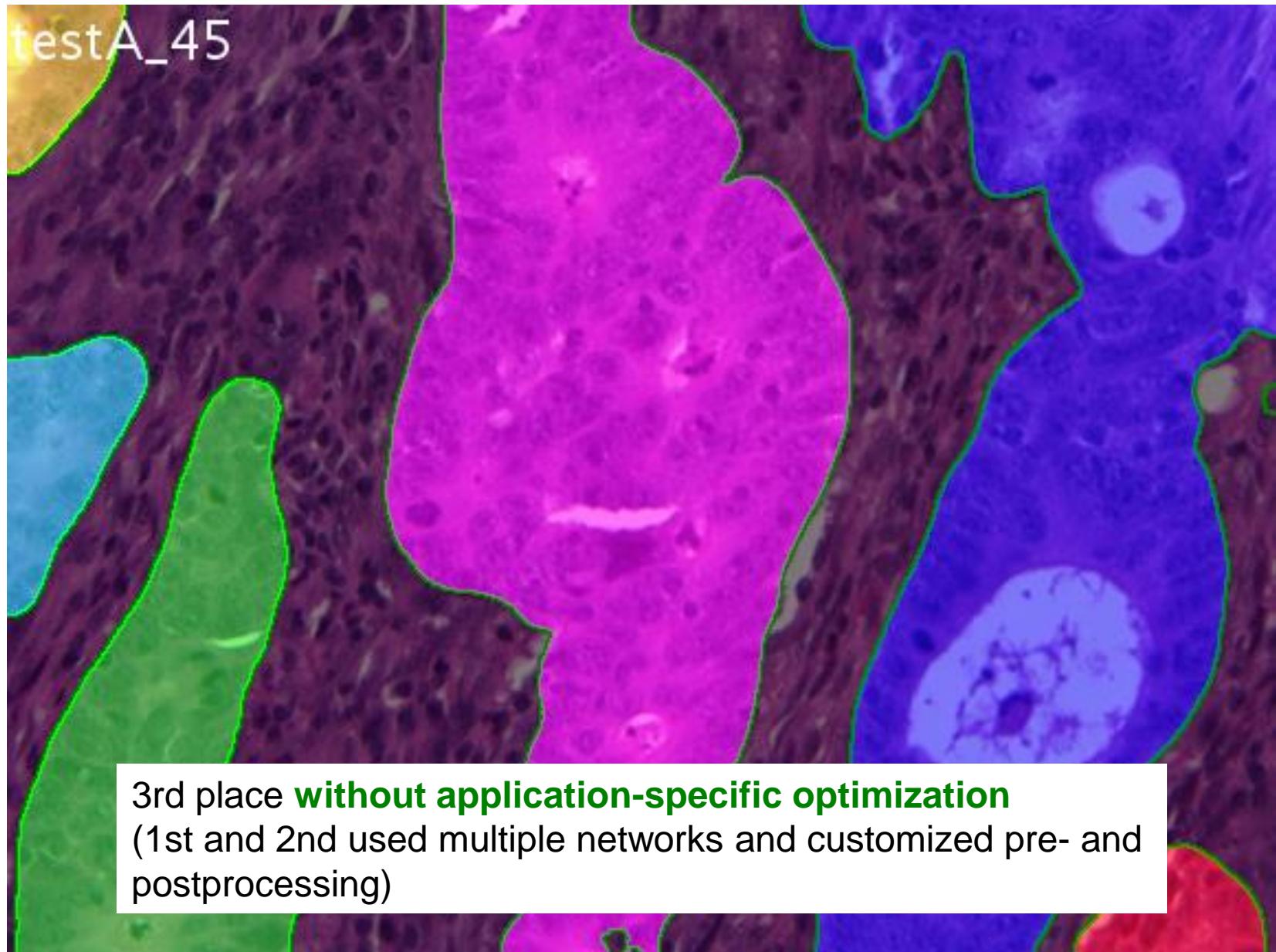
Qualitative Results: Glas Challenge Testset A



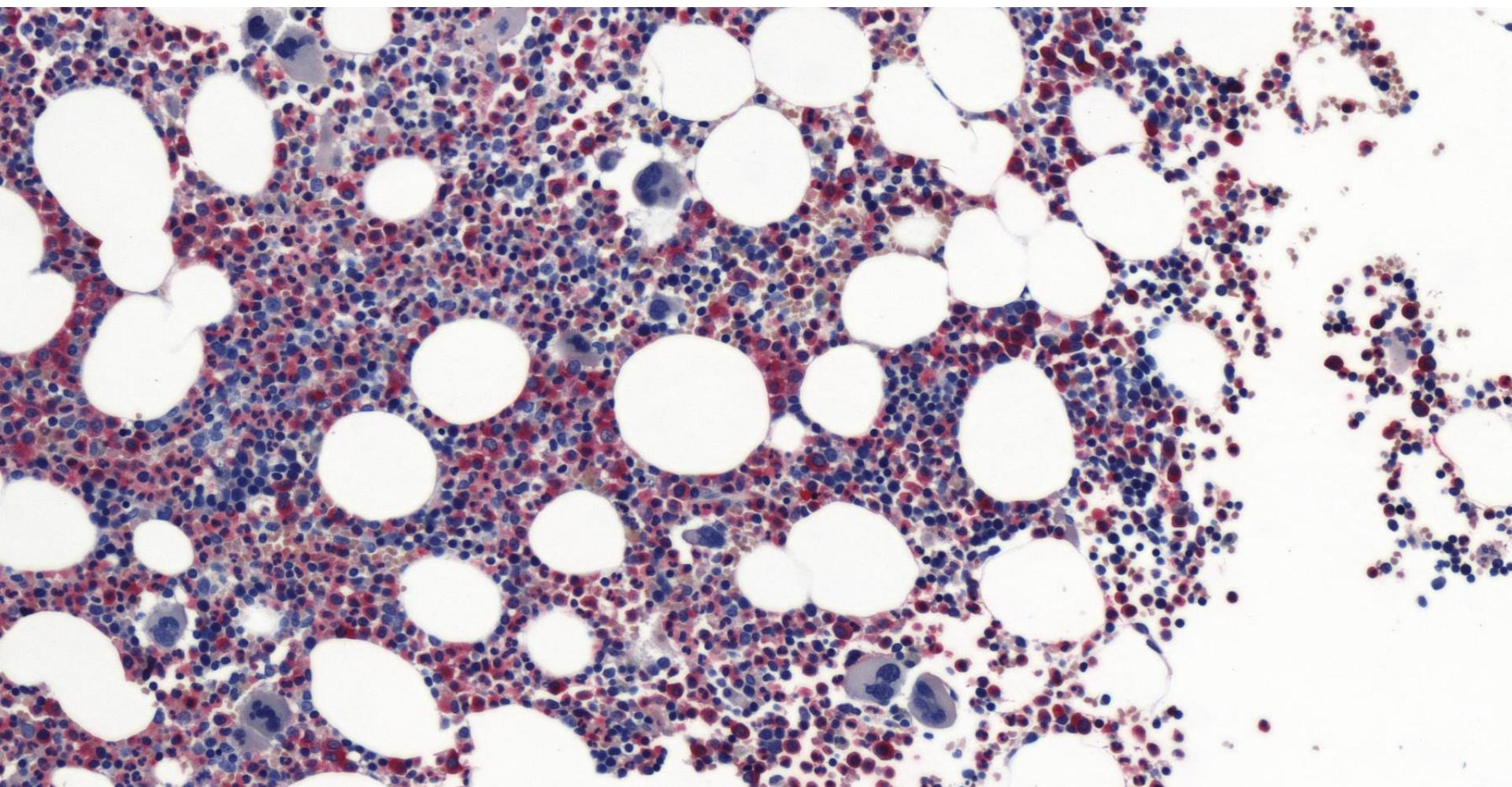
Qualitative Results: Glas Challenge Testset A



Qualitative Results: Glas Challenge Testset A

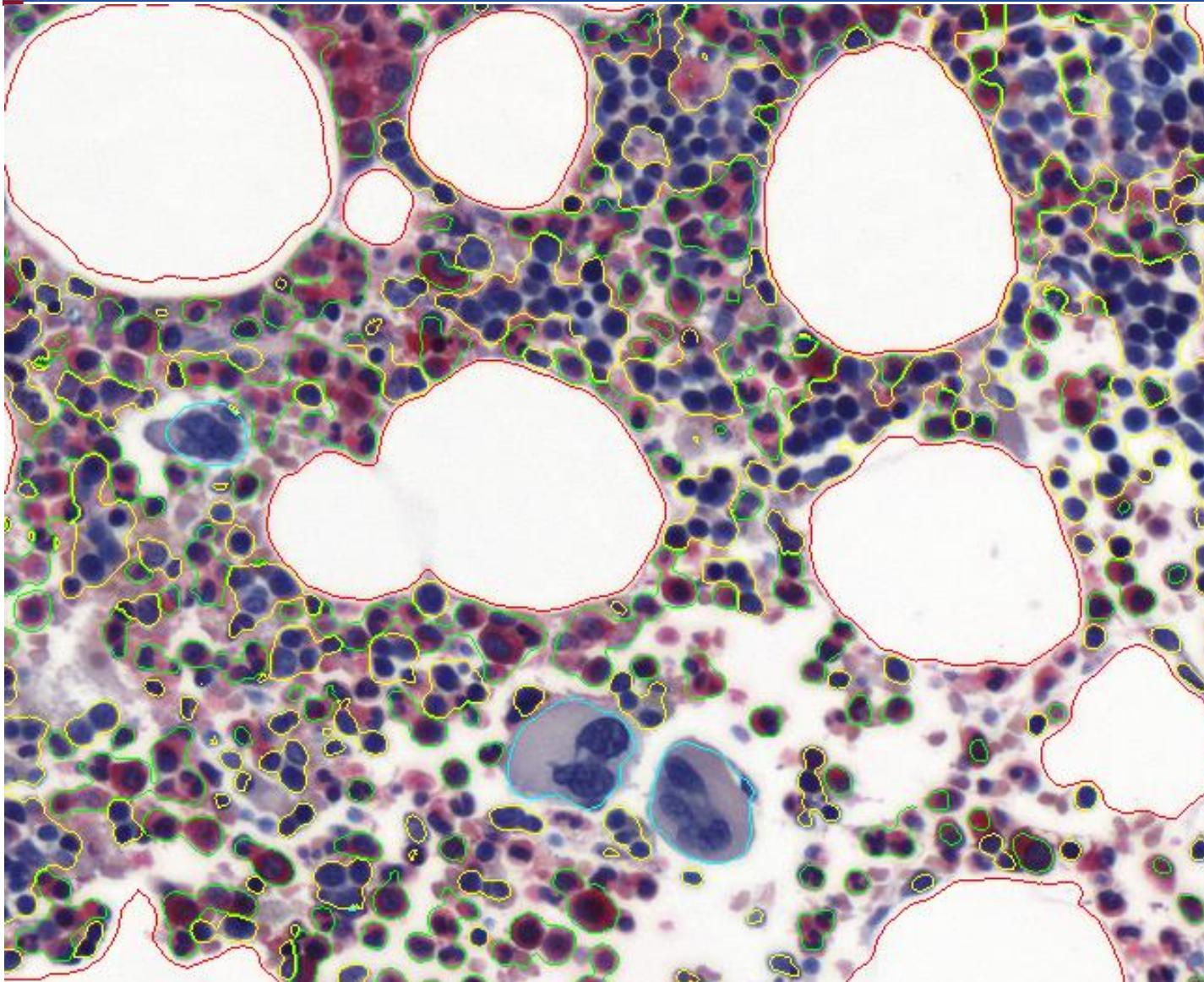


Segmentation of Bone Marrow



Anton Böhm, Collaboration with K. Aumann and M. Werner (Pathology)

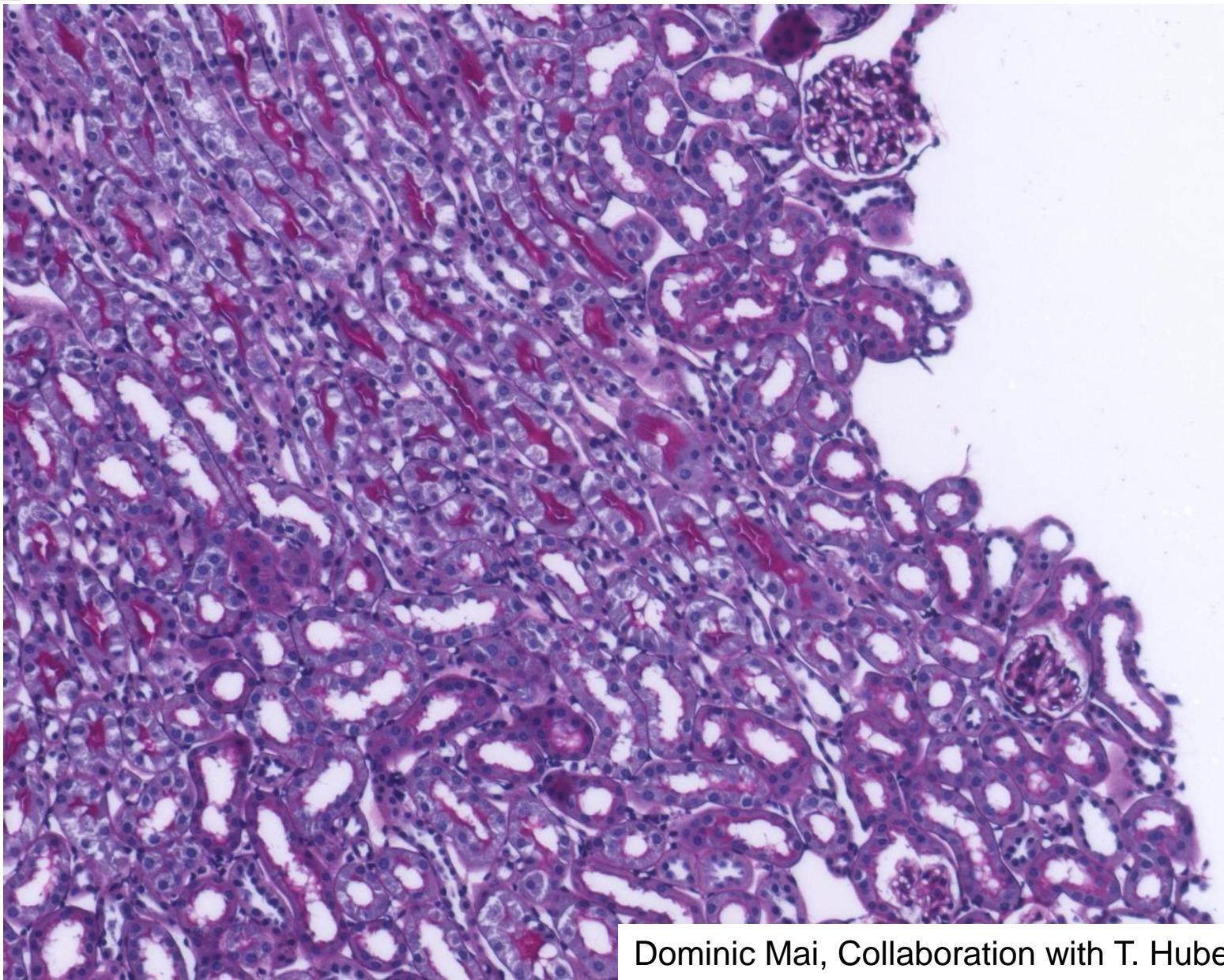
Segmentation of Bone Marrow



Erythropoiesis,
Granulopoiesis,
Megakaryocytes,
fat cells
trabecular bone

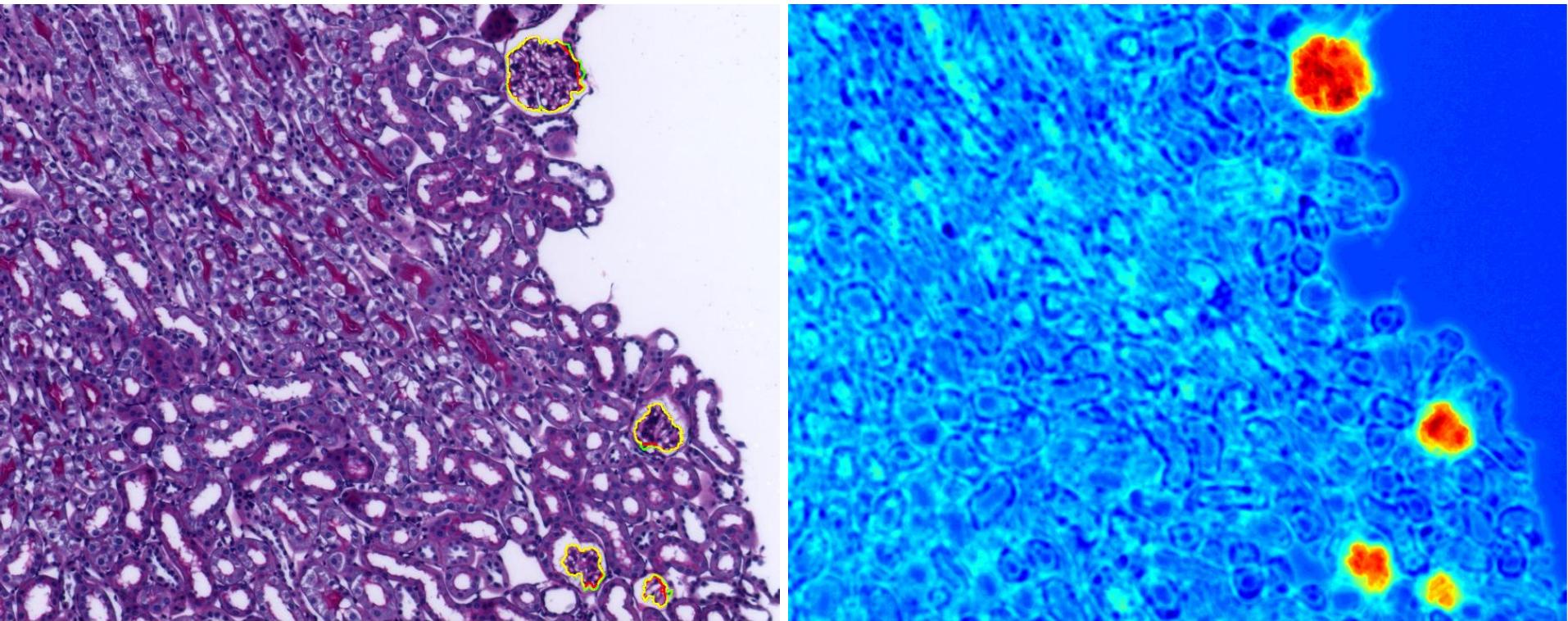
Anton Böhm, Collaboration with K. Aumann and M. Werner (Pathology)

Segmentation of Glomeruli (Kidney)



Dominic Mai, Collaboration with T. Huber (Nephrology)

Segmentation of Glomeruli (Kidney)



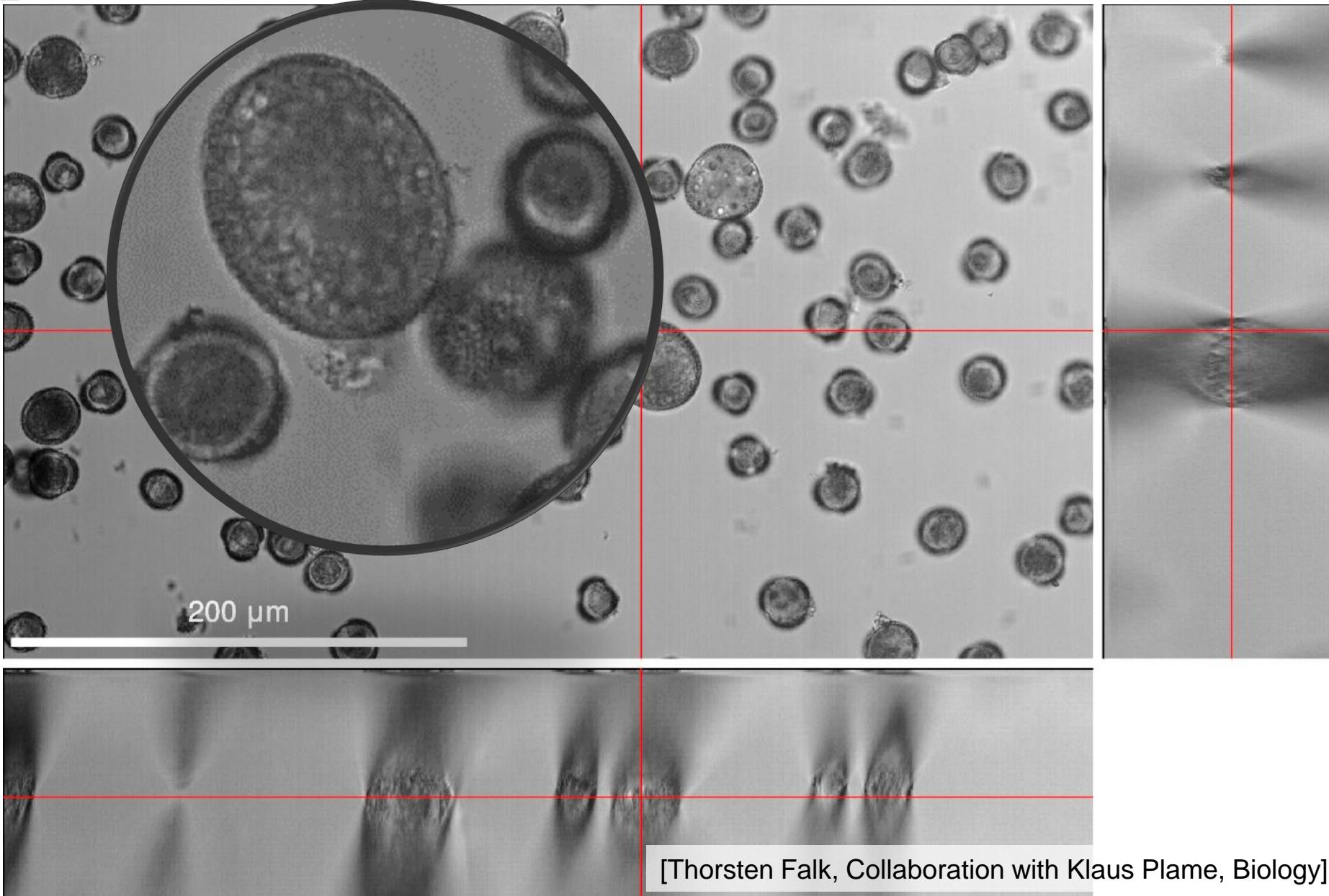
green: ground truth,
red: U-net

score map

Mean Intersection over Union: 82%

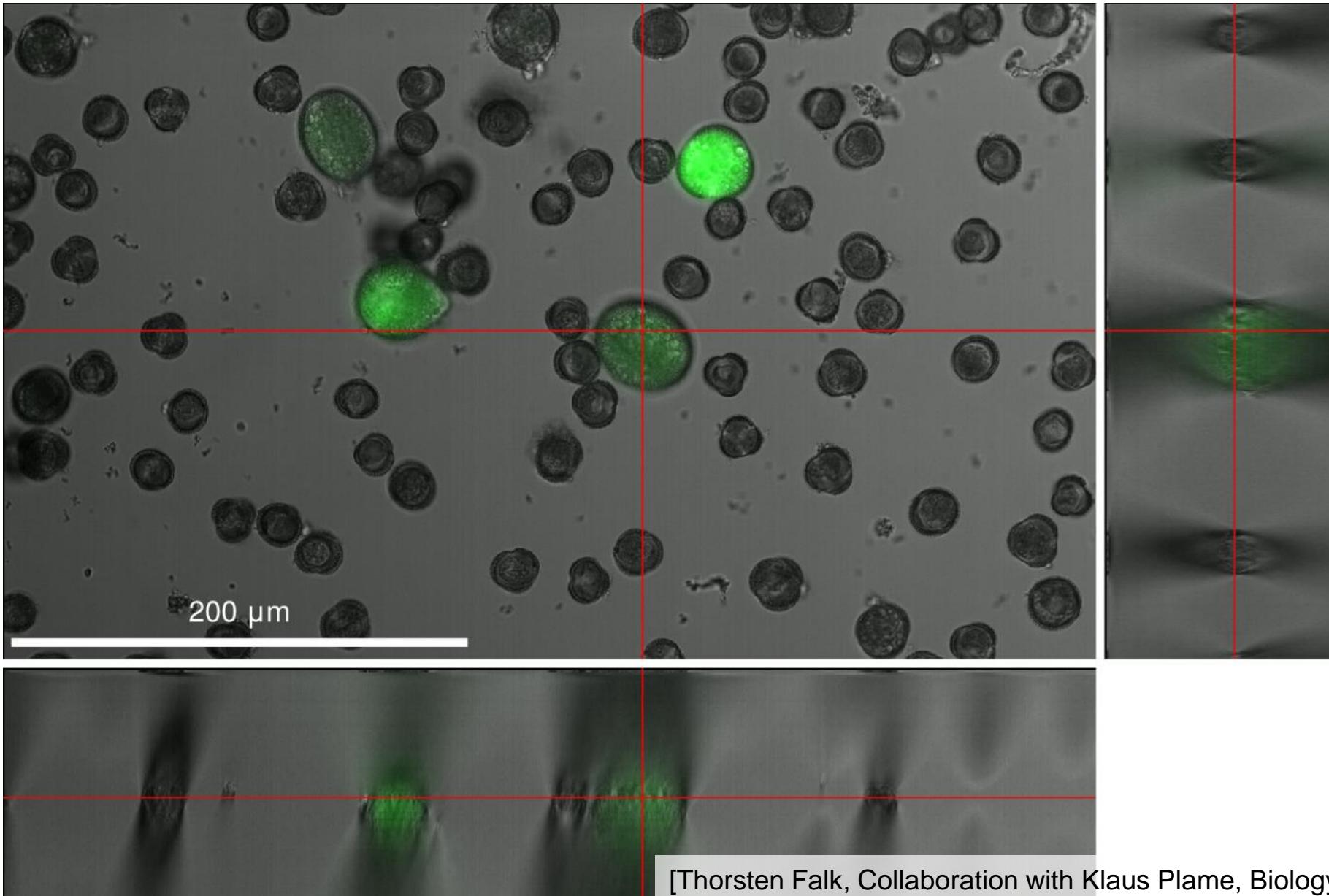
Dominic Mai, Collaboration with T. Huber (Nephrology)

Distinguish Living and Dead Cells (Short Video from Transmitted Light Microscopy)



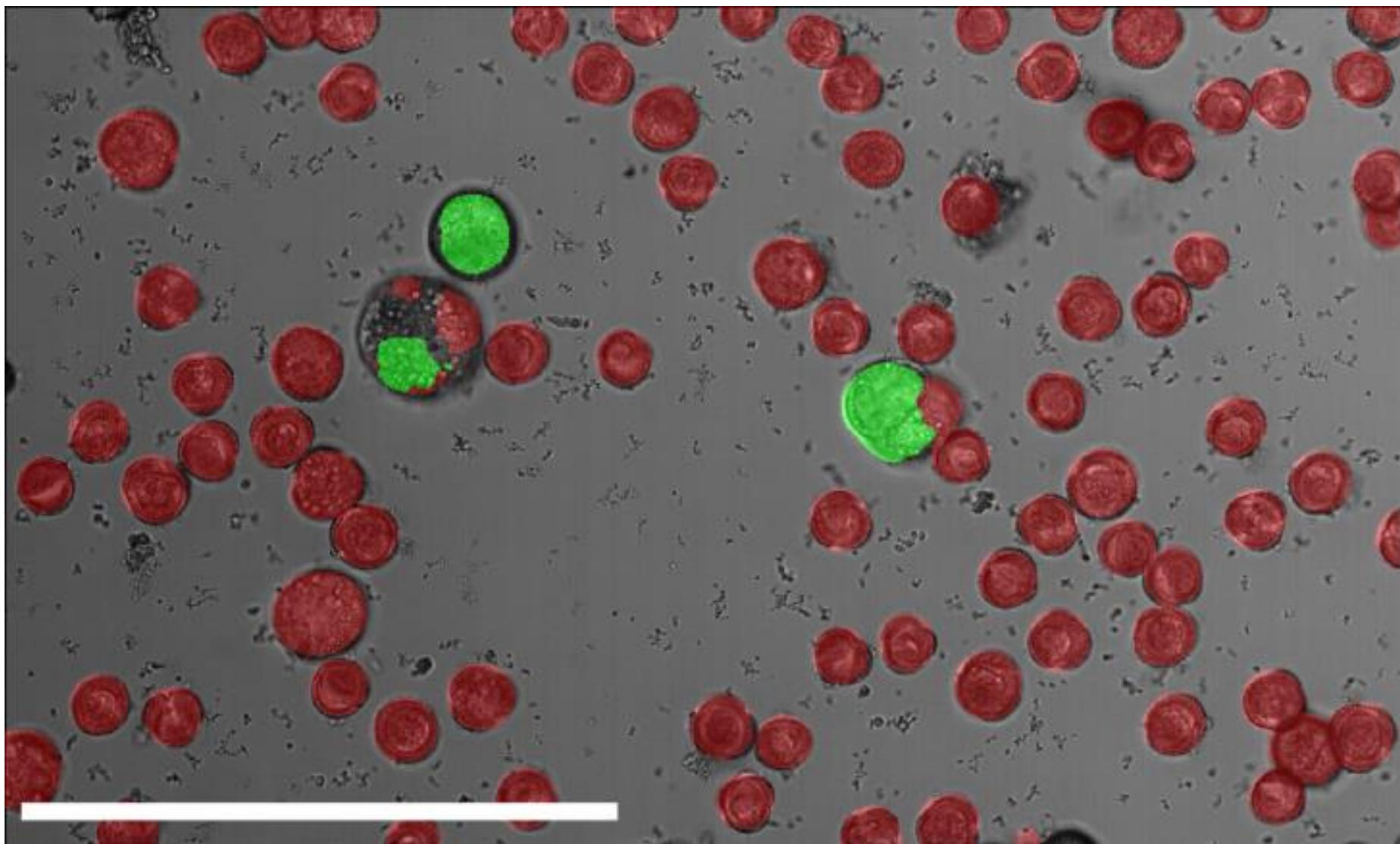
[Thorsten Falk, Collaboration with Klaus Plame, Biology]

Training Labels from CFDA Staining



[Thorsten Falk, Collaboration with Klaus Plame, Biology]

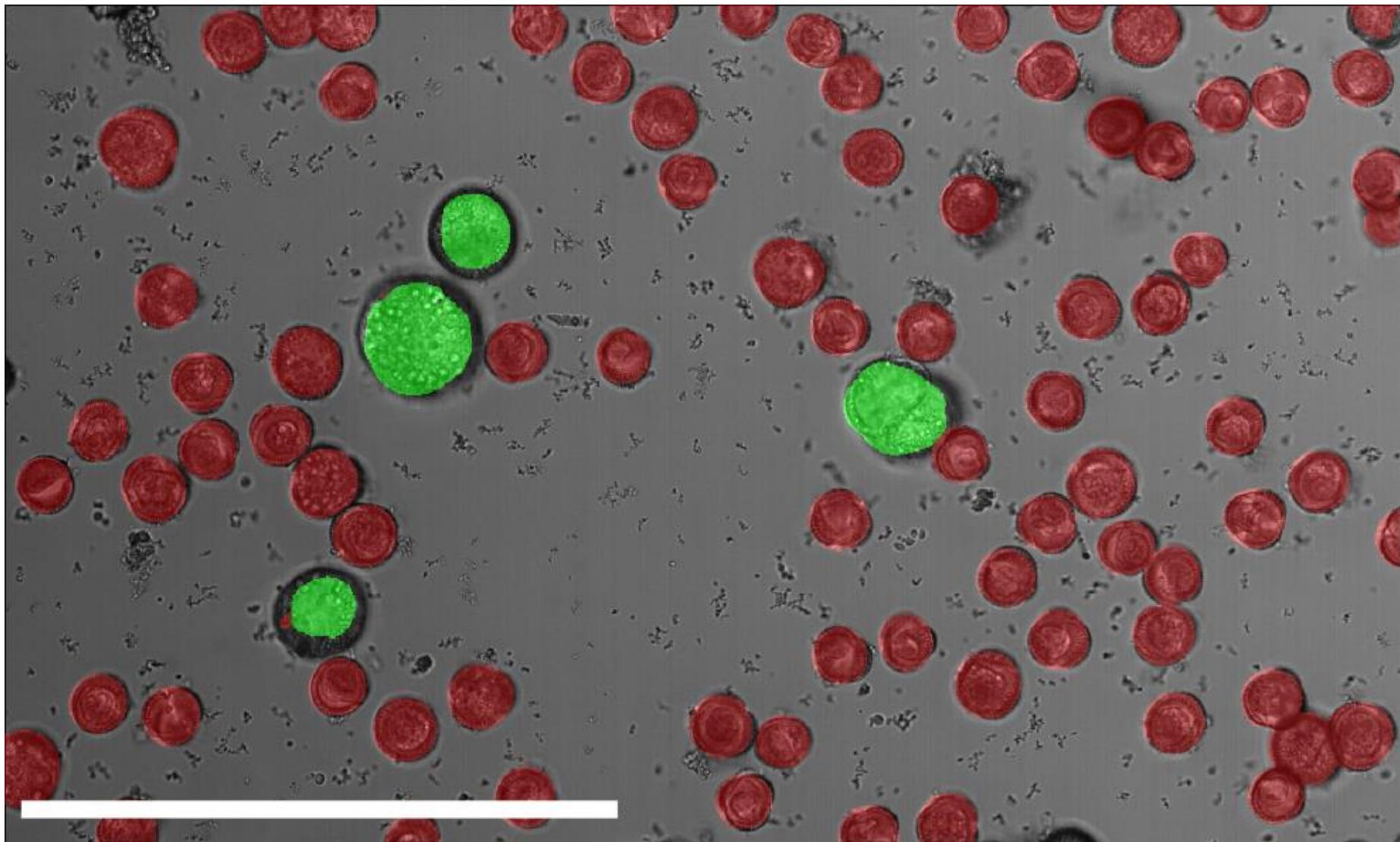
U-net Recognizes and Segments Living and Dead Cells



Result **without** time information

[Thorsten Falk, Collaboration with Klaus Plame, Biology]

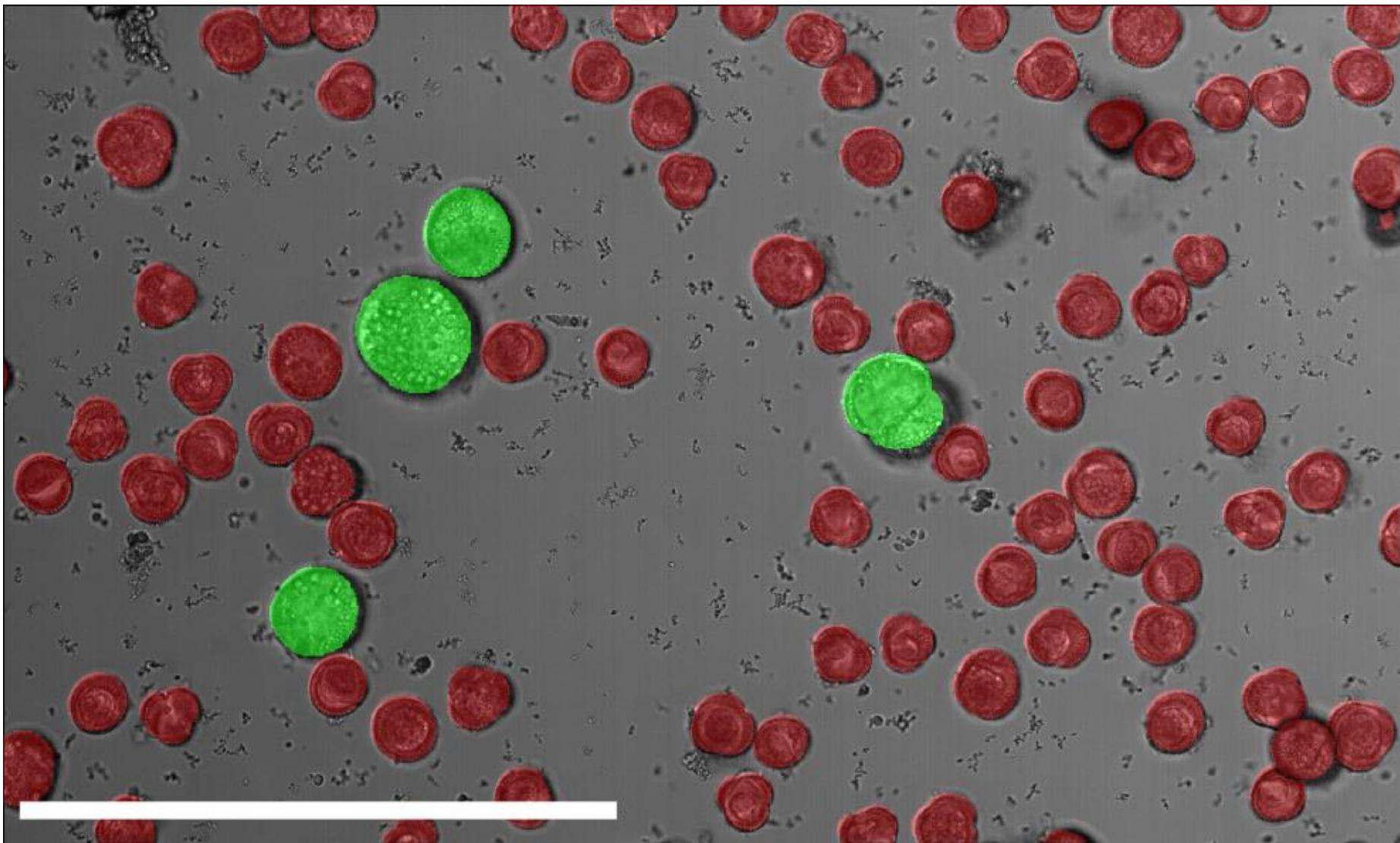
U-net Recognizes and Segments Living and Dead Cells



Result **with** time information

[Thorsten Falk, Collaboration with Klaus Plame, Biology]

U-net Recognizes and Segments Living and Dead Cells

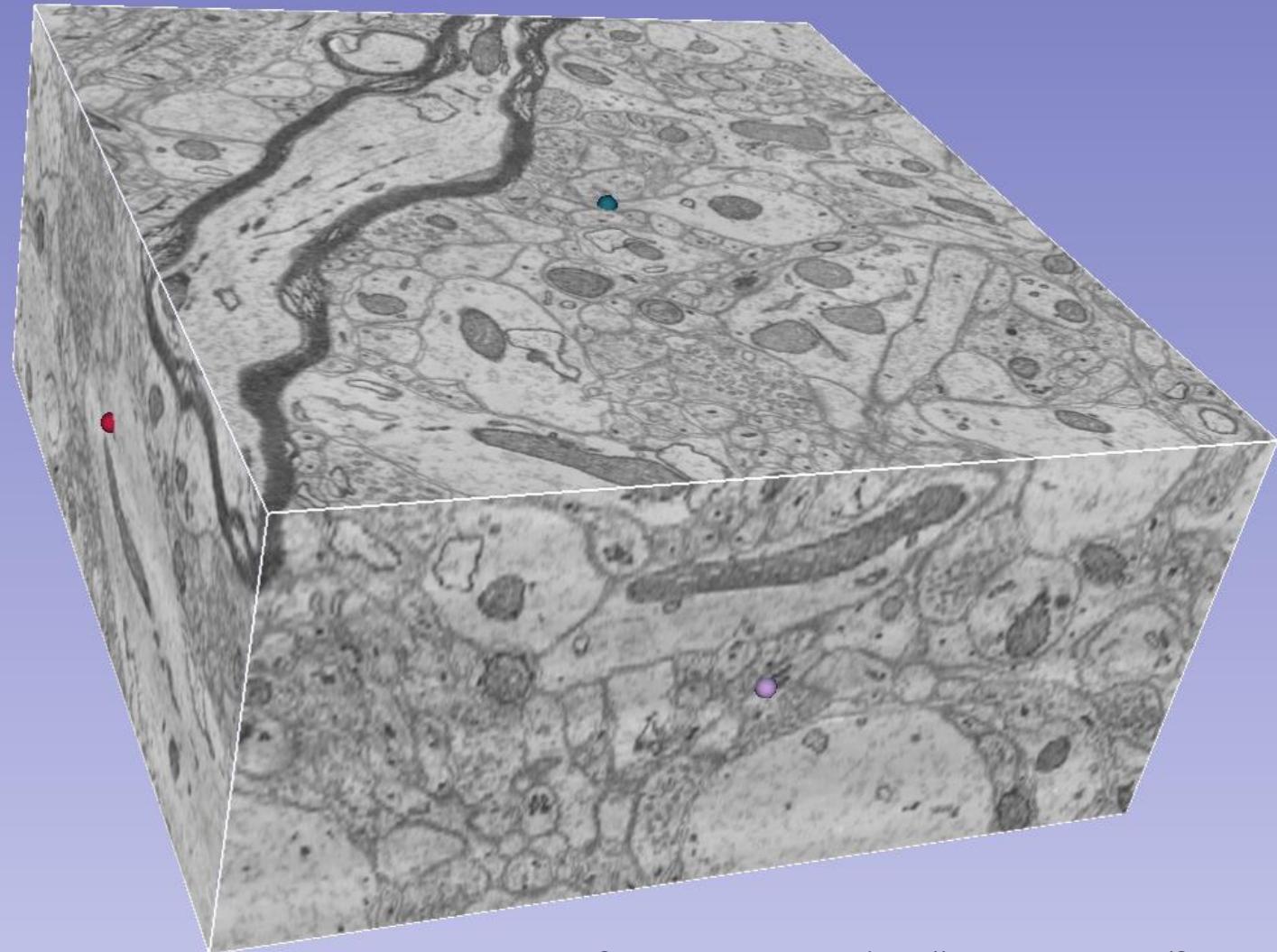


Manual ground truth annotation

[Thorsten Falk, Collaboration with Klaus Plame, Biology]

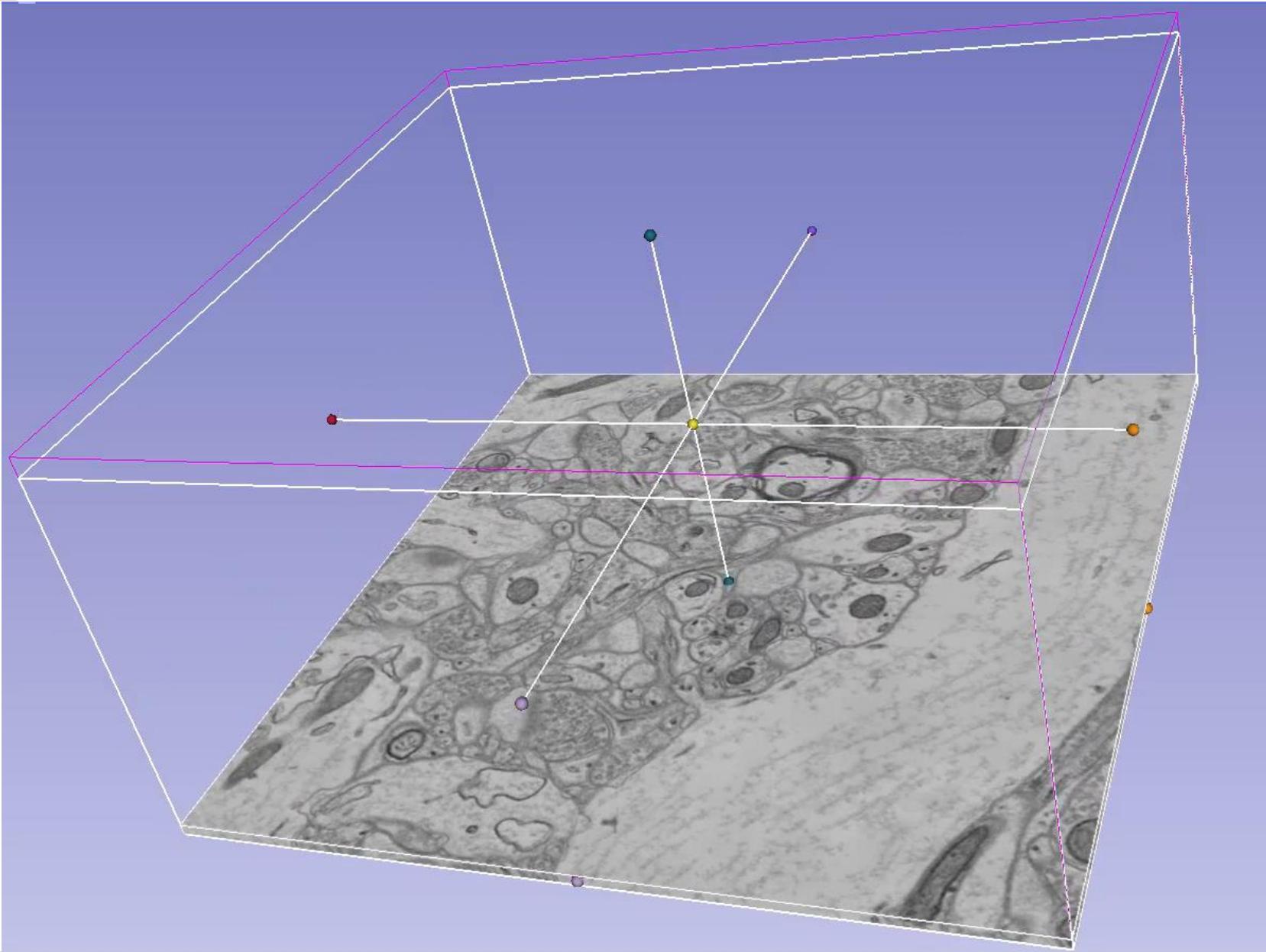
Extension to Volumetric Data

Serial section electron microscopy of mouse cortex.
1024 x 1024 x 100 voxels (element size: 6 x 6 x 30 nm³)

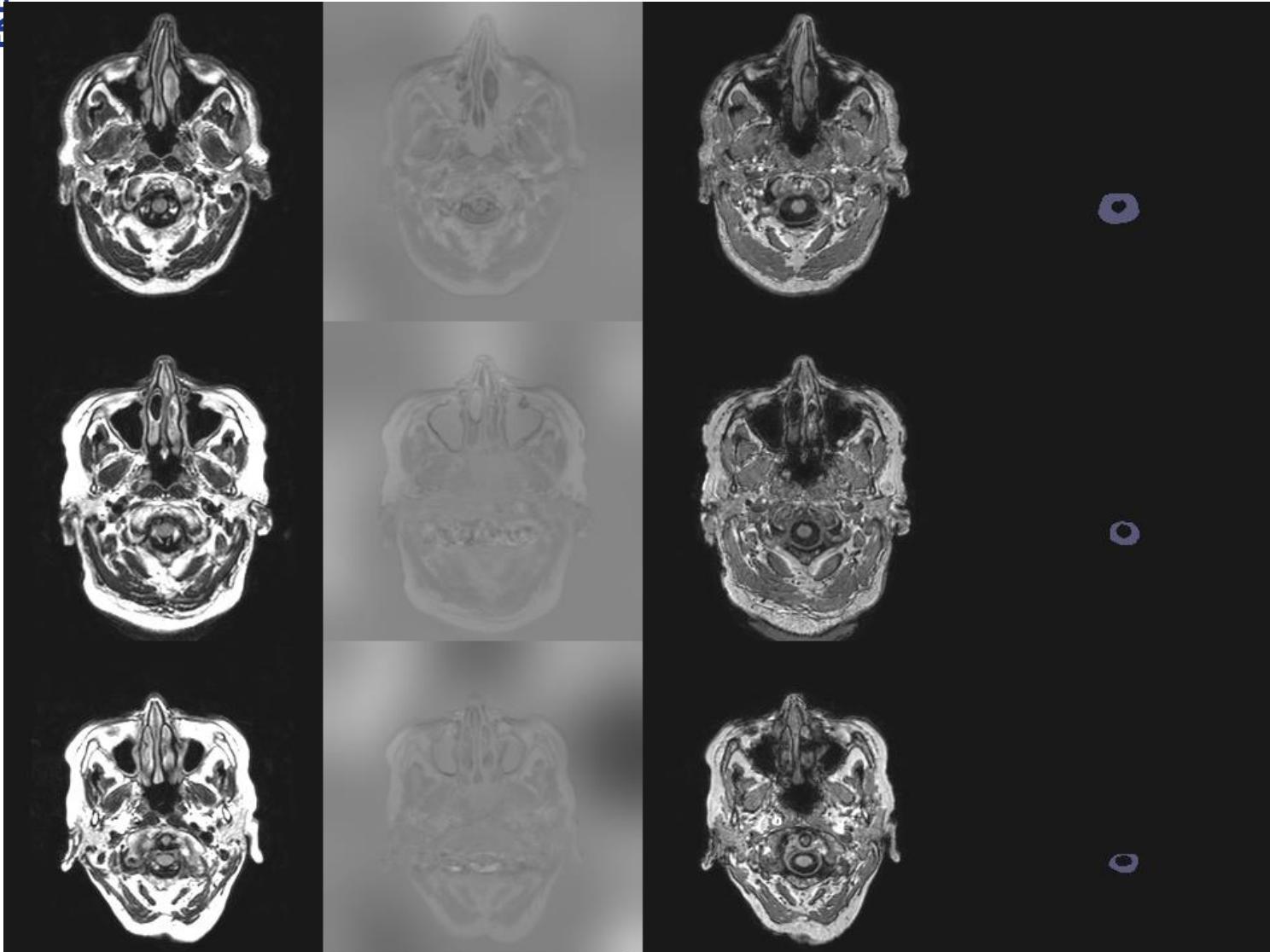


Data provided by the SNEMI3D challenge (<http://brainiac2.mit.edu/SNEMI3D/home>)

Results



Brain Segmentation Training Images (3 of 5)



T2-FLAIR

T1-IR

T1

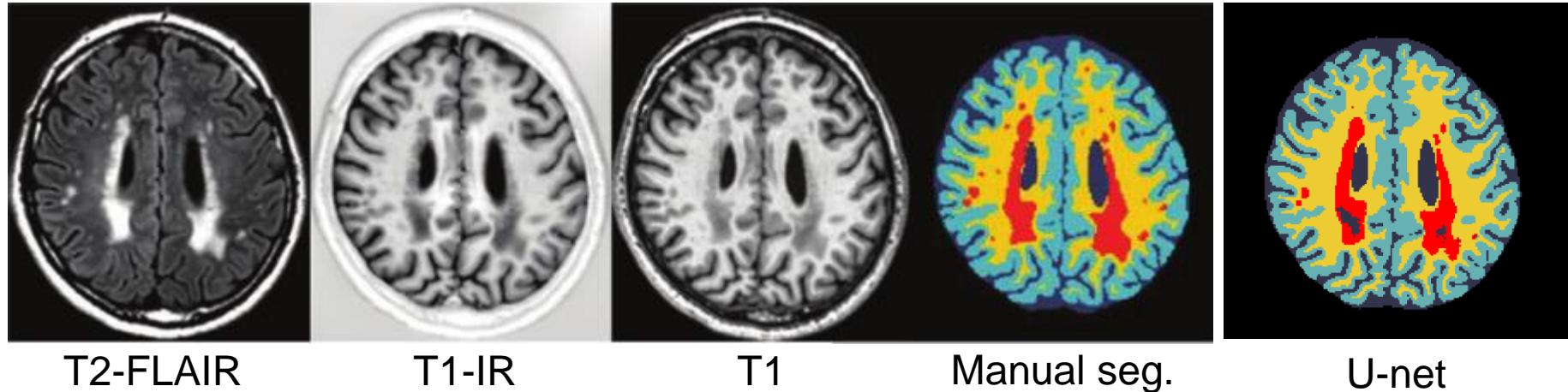
manual seg.

Ahmed Abdulkadir,
Collaboration with
Stefan Klöppel,
Neurology

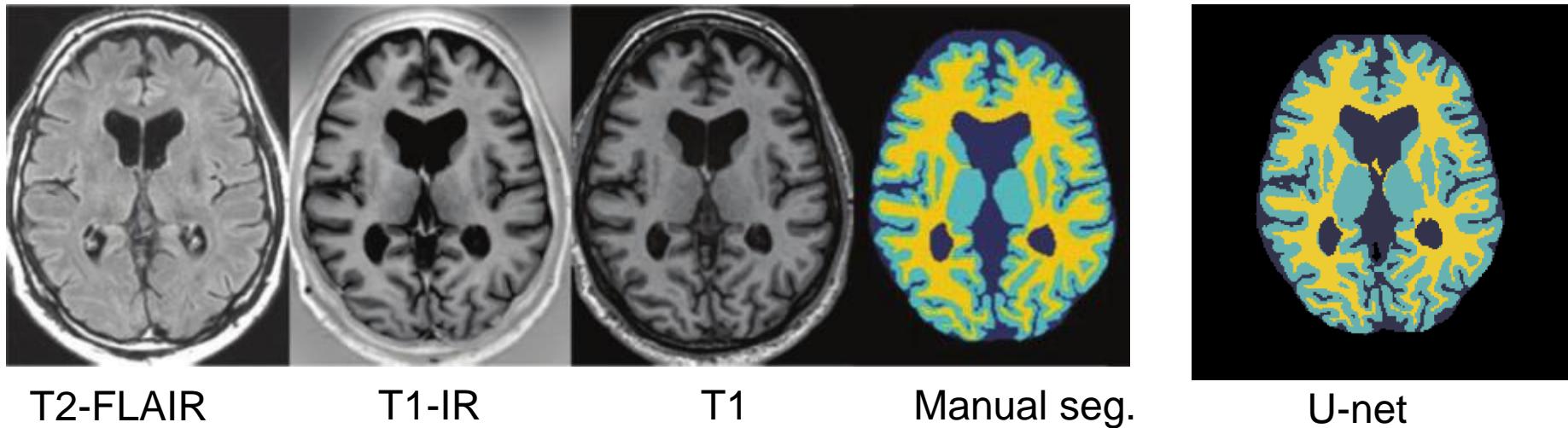
Data provided by MRBrainS Challenge 2013 <http://mrbrains13.isi.uu.nl/>

Brain Segmentation Preliminary Results

Test subject 03, slice 31



Test subject 09, slice 22



Conclusion

- Deep convolutional networks take over large parts of biomedical image analysis
- Learning segmentation (of biomedical structures) requires only few annotated training samples
- U-net architecture and elastically deformed training data do the job
- 2D implementation (based on Caffe) and ready trained networks available on our homepage
<http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net>

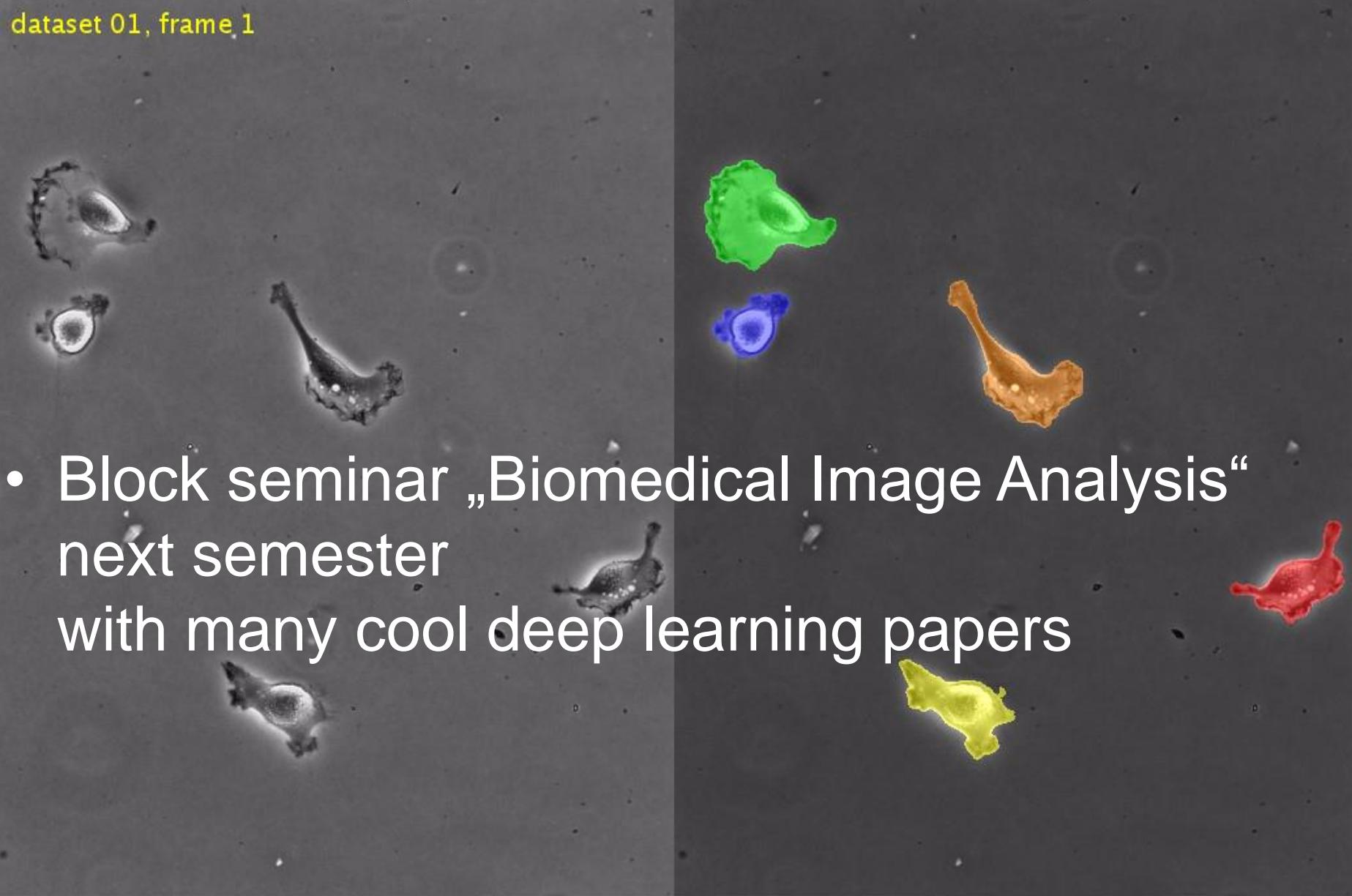
[O. Ronneberger, P. Fischer, T. Brox: U-net: Deep Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015, available of ArXiv]

Programming Assignment

- Get familiar with the caffe neural network framework
<http://caffe.berkeleyvision.org/>
- Train a neural network for handwritten digits recognition (MNIST dataset):
<http://caffe.berkeleyvision.org/gathered/examples/mnist.html>
This can be done on CPU or GPU
- If you don't have a recent NVidia GPU this exercise might be a good argument why your parents should put a Titan X under the Christmas tree -- of course a much smaller GPU or the CPU alone is sufficient for MNIST. You know, I know, but maybe your parents don't ☺
- If you don't want to compile caffe yourself (quite a lot of third party libraries), we will provide a ready compiled version (Linux, 64bit) on the course homepage.
- Optional: Do some cool stuff with the ready trained networks from the "model zoo" on the caffe homepage. Here you might really need a GPU.

Advertisement

dataset 01, frame 1



- Block seminar „Biomedical Image Analysis“
next semester
with many cool deep learning papers