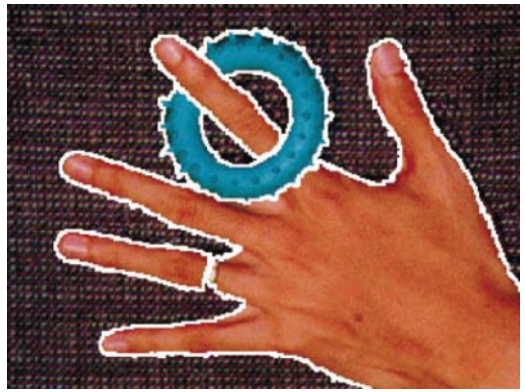
COMP9517
Computer Vision

2024 Term 3 Week 5

Professor Erik Meijering

UNSW SYDNEY



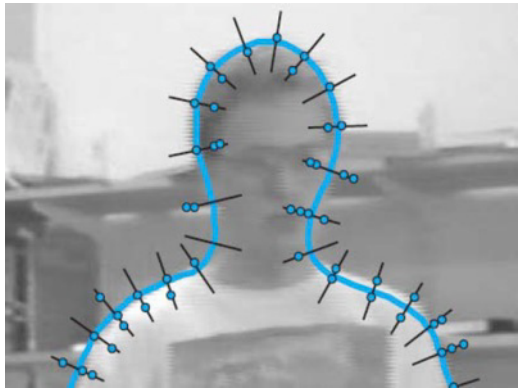


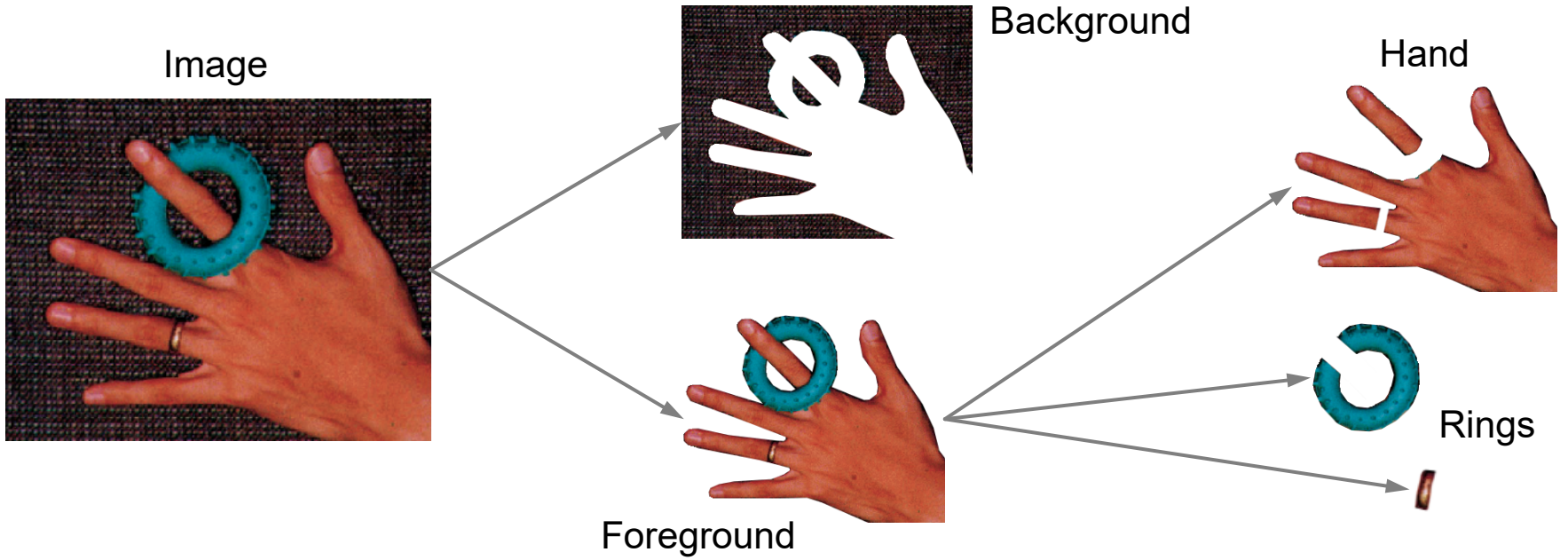
Image Segmentation

Part 1

What do you see in this image?

What is image segmentation?

- Partitioning an image into a set of meaningful segments for further analysis
- One of the oldest and most widely studied problems in computer vision



Segments
(a.k.a. regions)

Facilitating image segmentation

- Regions should be **uniform / homogeneous** in some characteristics
- Adjacent regions should be **significantly different** in these characteristics
- Region interiors should be **simple** and without holes or missing parts
- Boundaries of each region should be **smooth and spatially accurate**

This is almost never the case

Some important terminology

https://arxiv.org/abs/1704.06857

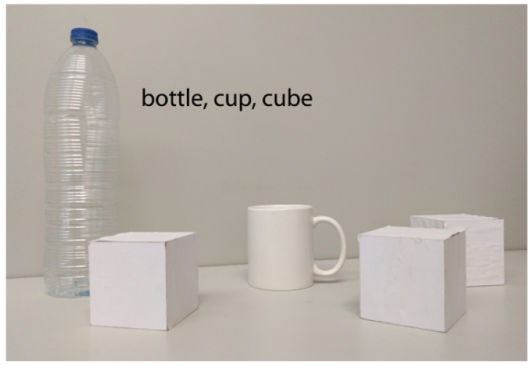
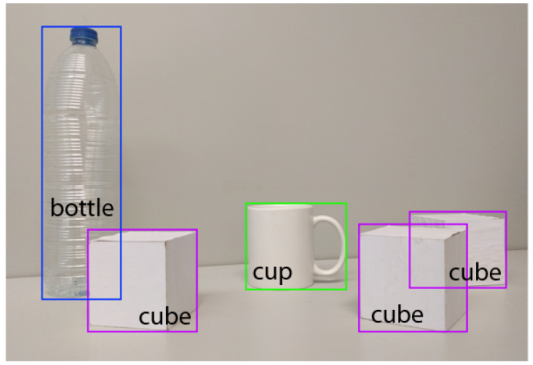


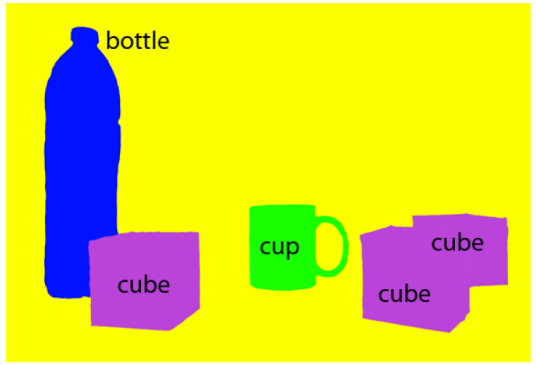
Image classification

Assigning a single label to the whole image



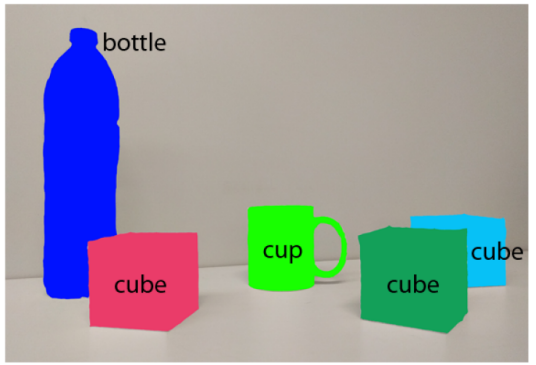
Object localisation

Finding the bounding boxes of all relevant objects in the image



Semantic segmentation

Classifying each pixel in the whole image

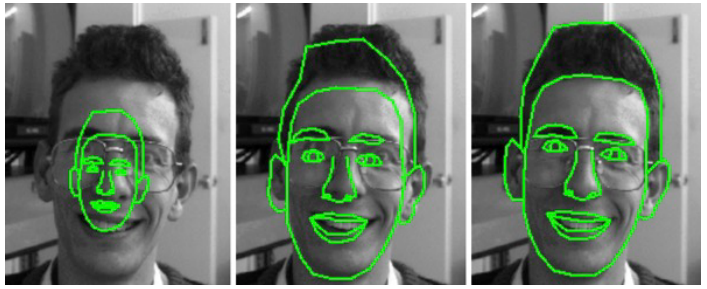


Instance segmentation

Uniquely labelling the pixels of each object instance in the image

Segmentation problems and challenges

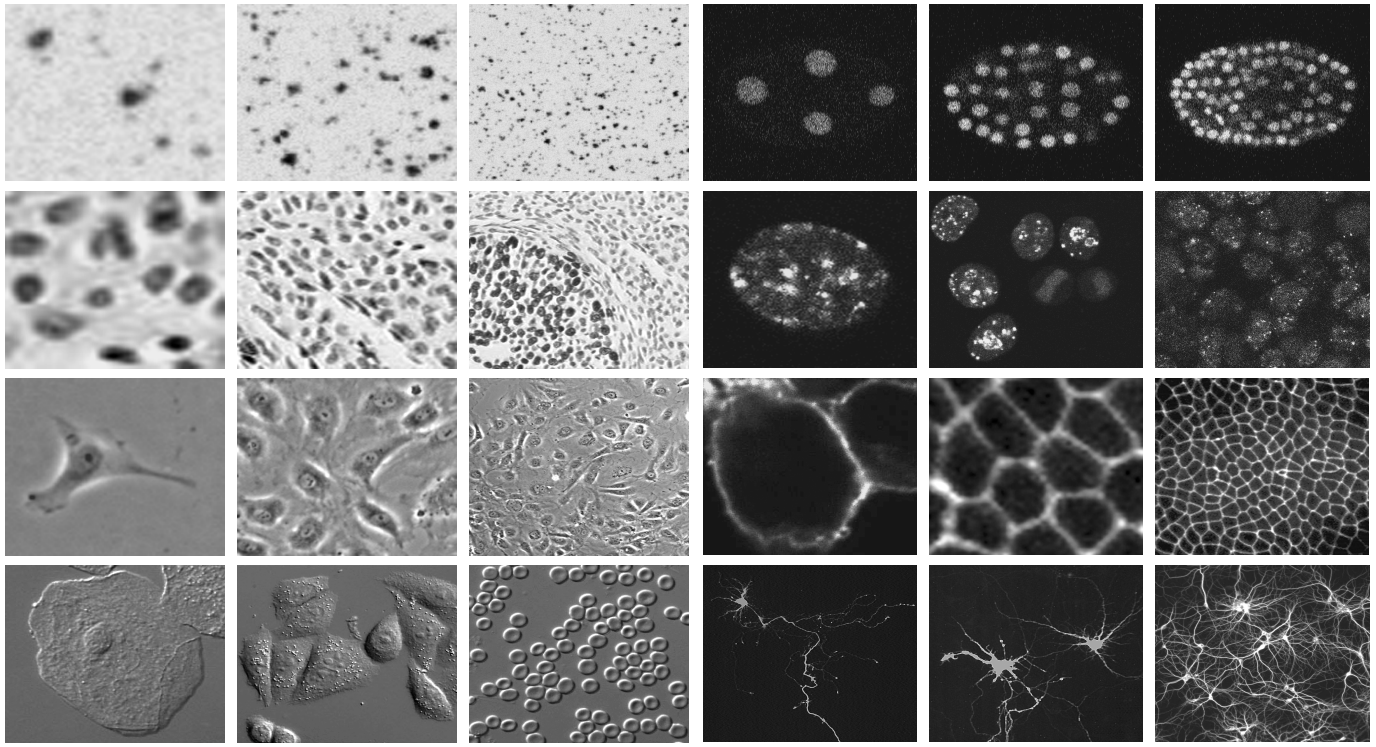
- There is no single segmentation method that works well for all applications
- Special domain knowledge of the application is typically essential for success



http://www.isbe.man.ac.uk/~bim/Models/asms.html

Segmentation problems and challenges

- Even within a given application domain images may still vary widely



All these examples are microscope images of biological cells

Topics and learning goals

- Recap **basic segmentation** methods

  Thresholding, K-means clustering, feature based pixel classification

- Describe more **advanced segmentation** methods

  1. Region splitting and merging
  2. Watershed segmentation
  3. Maximally stable extremal regions
  4. Mean shifting
  5. Superpixel segmentation
  6. Conditional random field
  7. Active contour segmentation
  8. Level-set segmentation

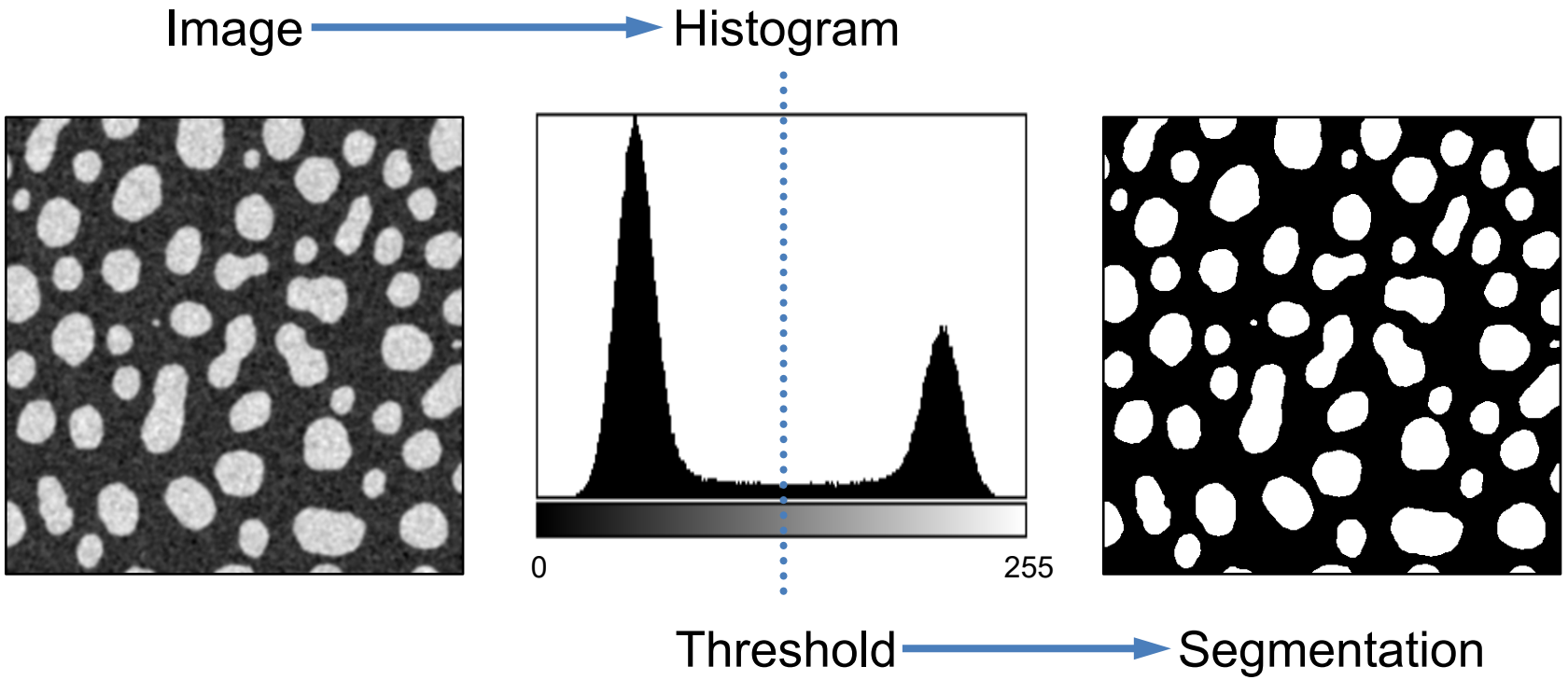
- Understand how to **evaluate segmentation** methods

  Pixel classification, quantitative evaluation metrics, receiver operating characteristic

Basic segmentation methods

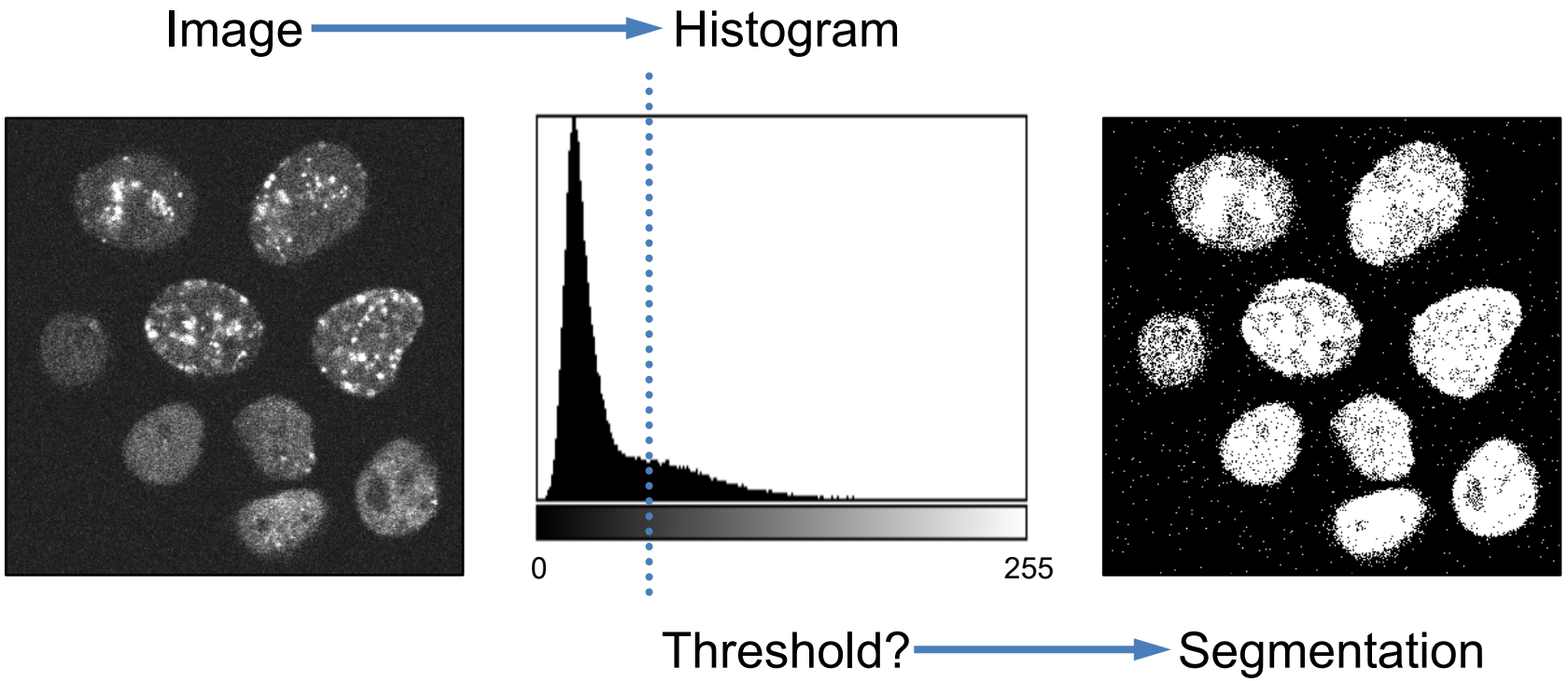
Thresholding

- Works fine if regions have sufficiently different intensity distributions



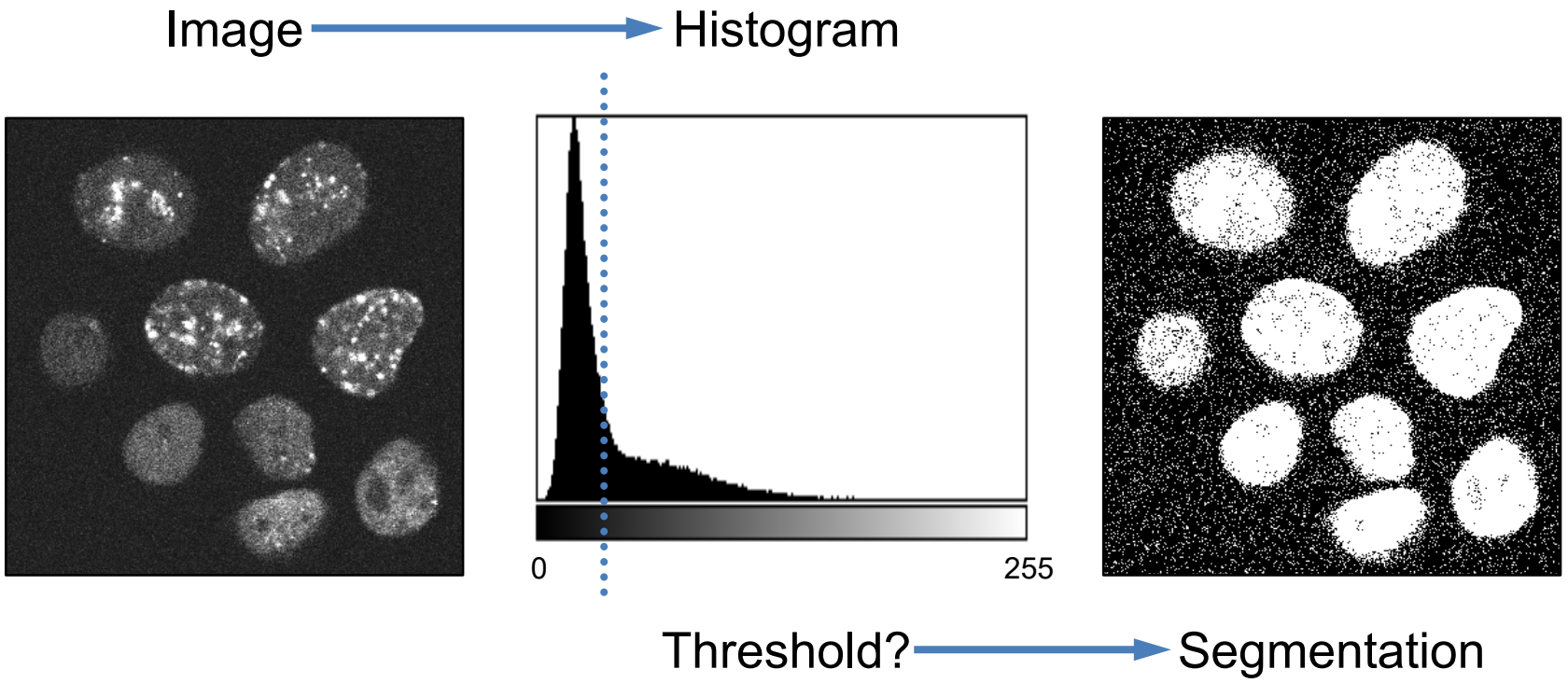
Thresholding

- Problematic if regions have overlapping intensity distributions



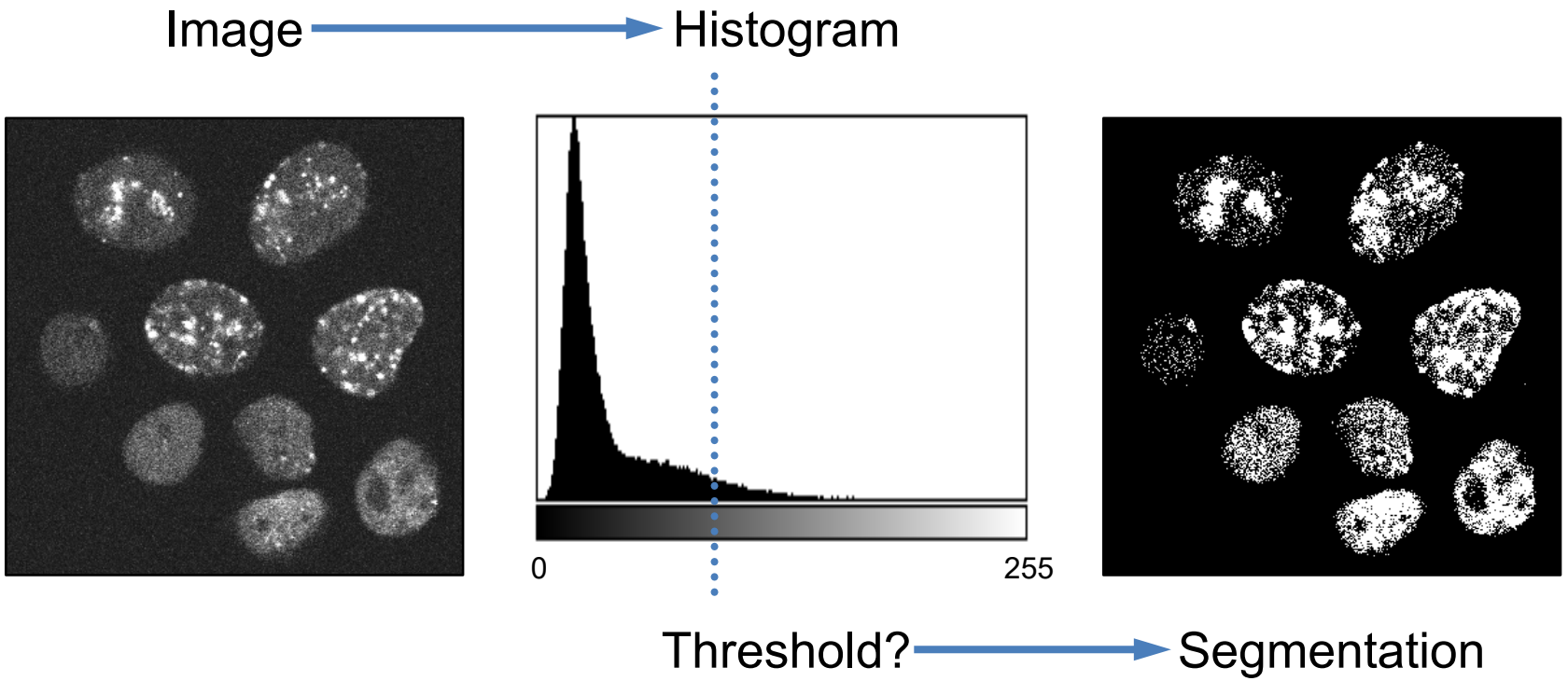
Thresholding

- Problematic if regions have overlapping intensity distributions



Thresholding

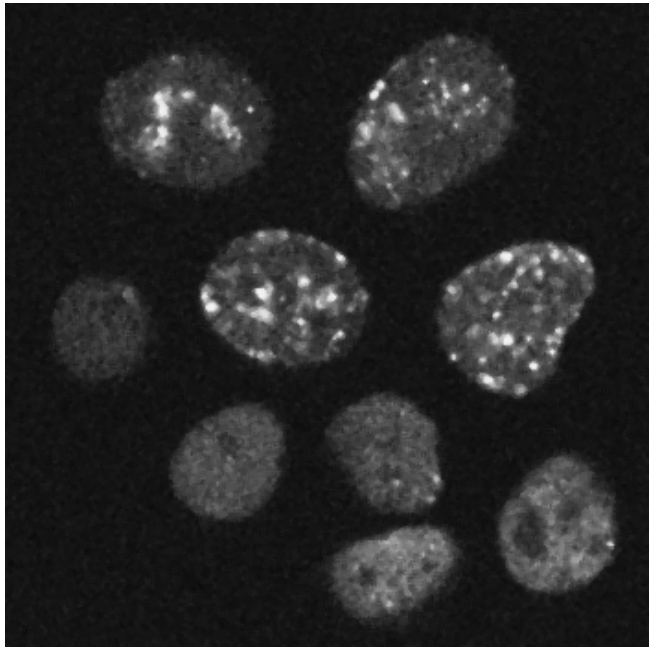
- Problematic if regions have overlapping intensity distributions



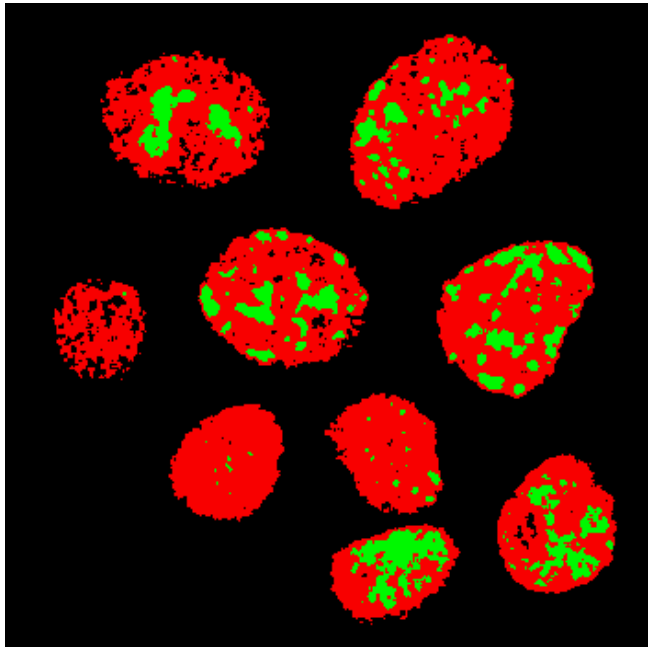
K-means clustering

- May work if the number of clusters is known a priori

Original Image



Labeled Image



K-means clustering

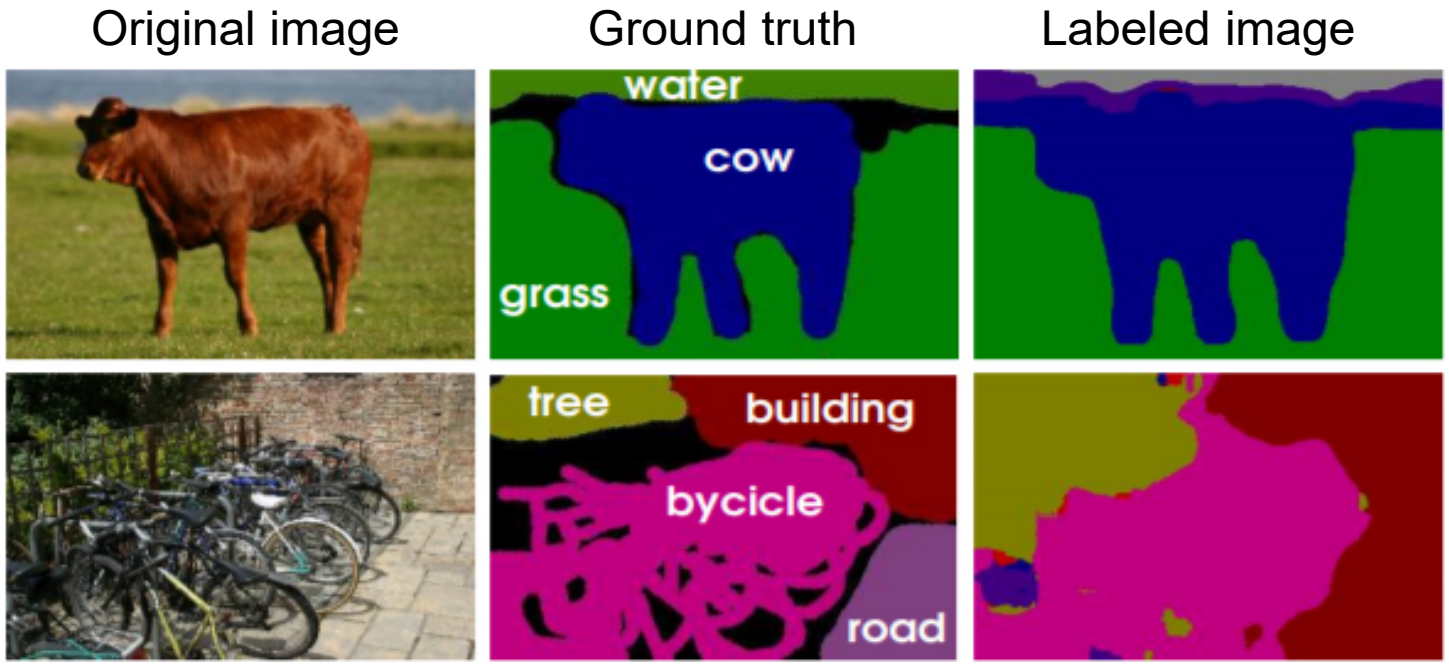
- Problematic if the number of clusters is not known a priori

Original Image

Labeled Image

Feature based pixel classification

- Extract a patch around each pixel and compute its features
- Classify each pixel based on its features using a trained classifier



Requires many example patches for training the classifier

https://doi.org/10.1007/11949619_8

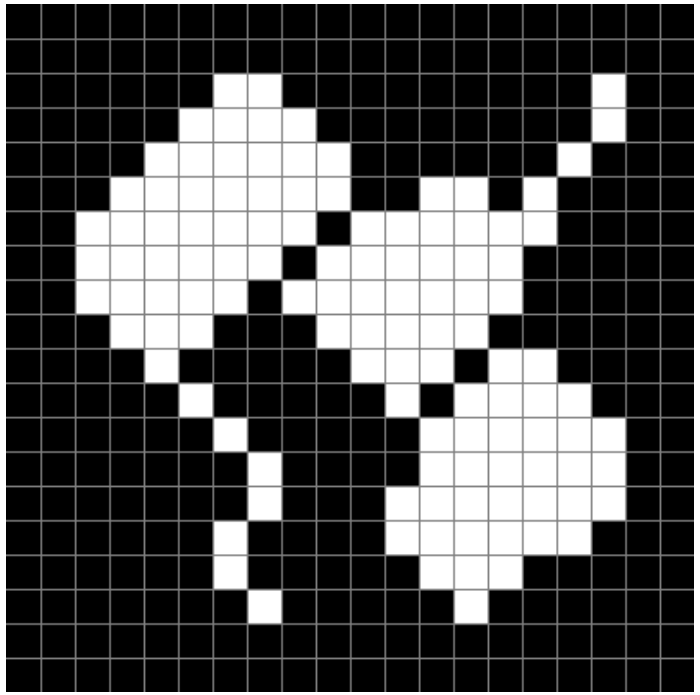
Advanced segmentation methods

1. Region splitting and merging

- Recursively split the whole image into pieces based on local statistics
- Recursively merge pieces together (in a hierarchical fashion)
- Combine splitting and merging sequentially

Simple(st) example

- Apply thresholding (splitting) and find regions of connected pixels (merging)



Segment pixels individually:

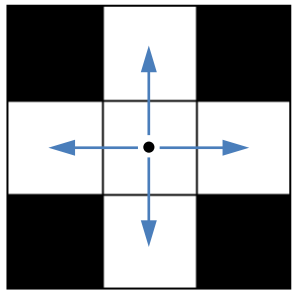
$$S(x, y; \tau) = \begin{cases} 1 & \text{if } I(x, y) \geq \tau \\ 0 & \text{else} \end{cases}$$

How many connected components (separate objects) are there in this thresholded image?

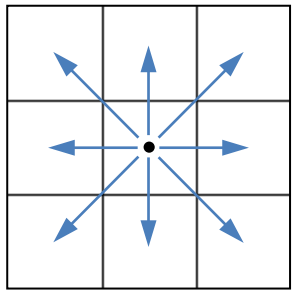
Connectivity

- Connectivity in two dimensions (2D)

  Consider pixels in surrounding rows and columns



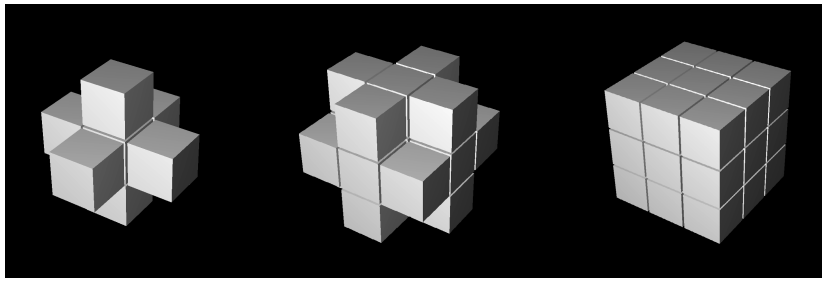
4-connected



8-connected

- Connectivity in three dimensions (3D)

  Consider voxels in surrounding rows, columns, planes

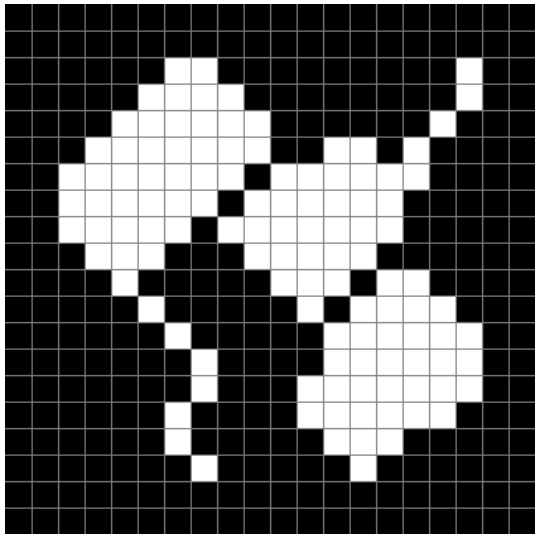


6-connected 18-connected 26-connected

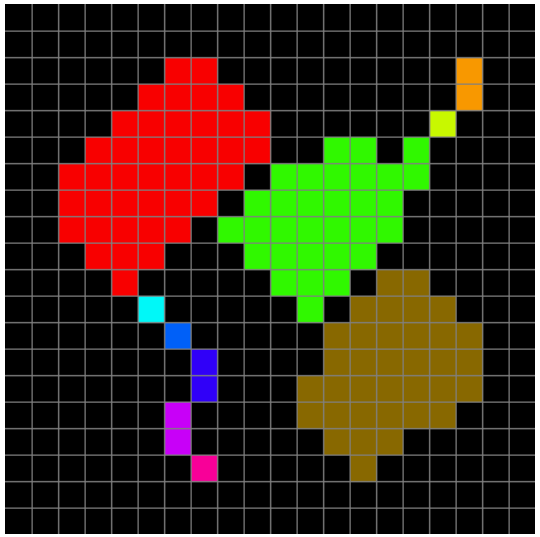
Connected components

- Number of components depends on the chosen connectivity

Thresholded image

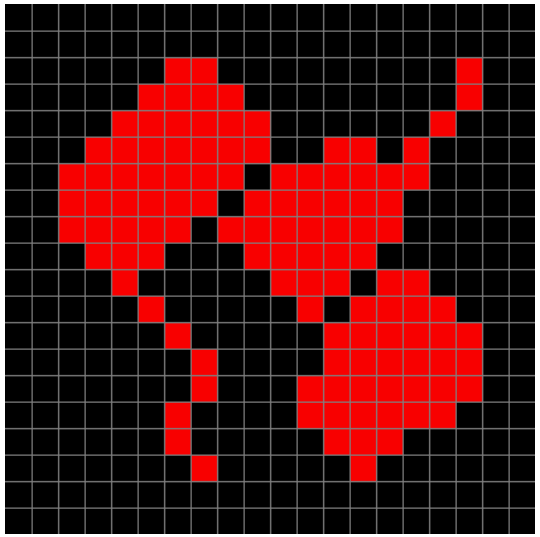


4-connected objects



Number of objects = 10

8-connected objects



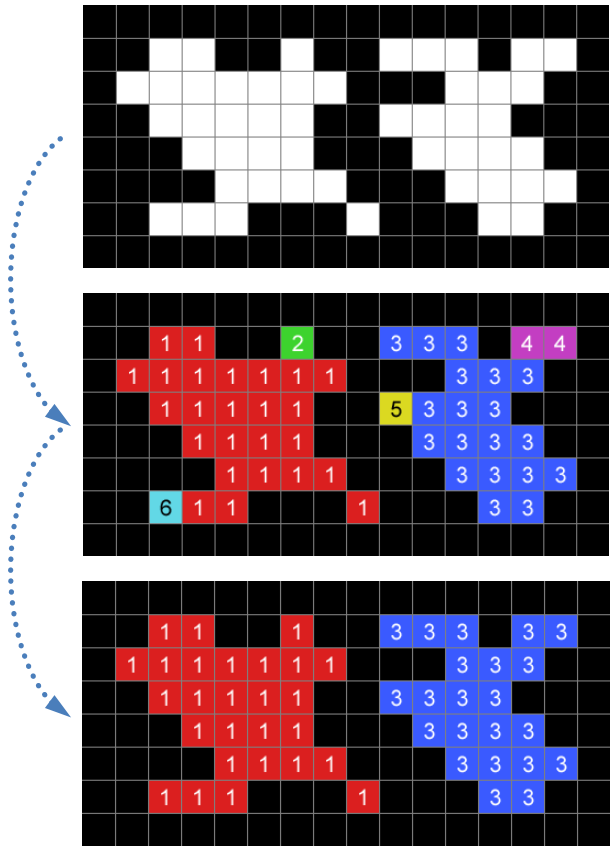
Number of objects = 1

Connected components labelling algorithm

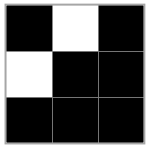
First pass

- Check each pixel (top-left to bottom-right)
- If an object pixel, check its neighbours (N_4 or N_8)
- If no neighbours have labels, assign a new label
- If neighbours do have labels, assign the smallest
- Record label equivalences while assigning

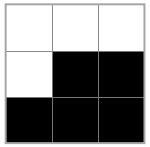
  Equivalence sets: {1,2,6} {3,4,5}



N_4



N_8

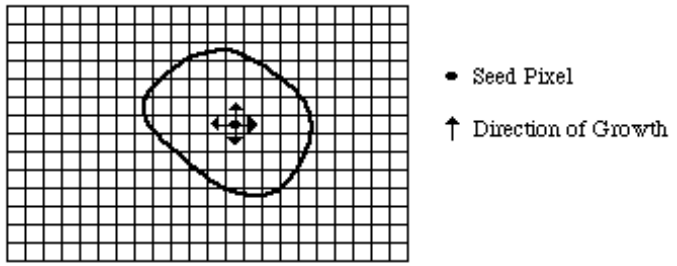


Second pass

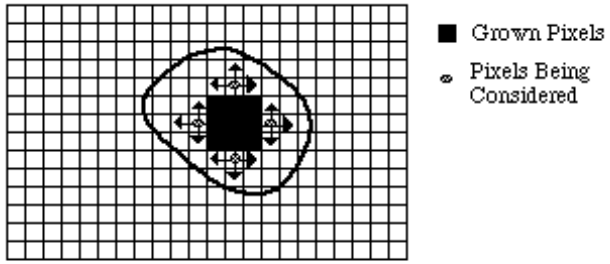
- Check each pixel (top-left to bottom-right)
- Replace each label with its smallest equivalent
- All background pixels default to the zero-label

Merging by region growing

- Define a similarity measure
- Start from one seed pixel for the region
- Add neighbouring pixels to the region if they are similar
- Repeat previous step until no more pixels are similar



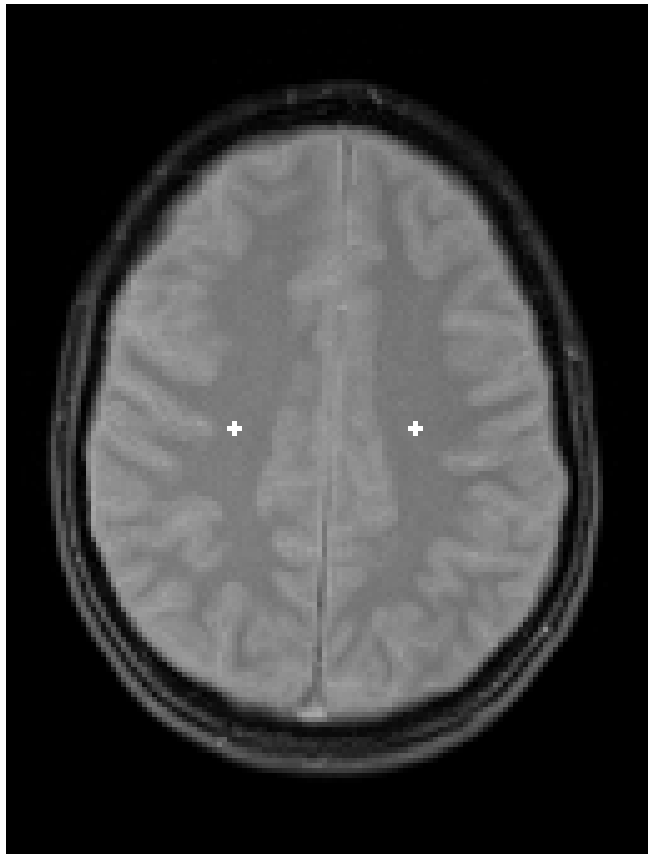
(a) Start of Growing a Region

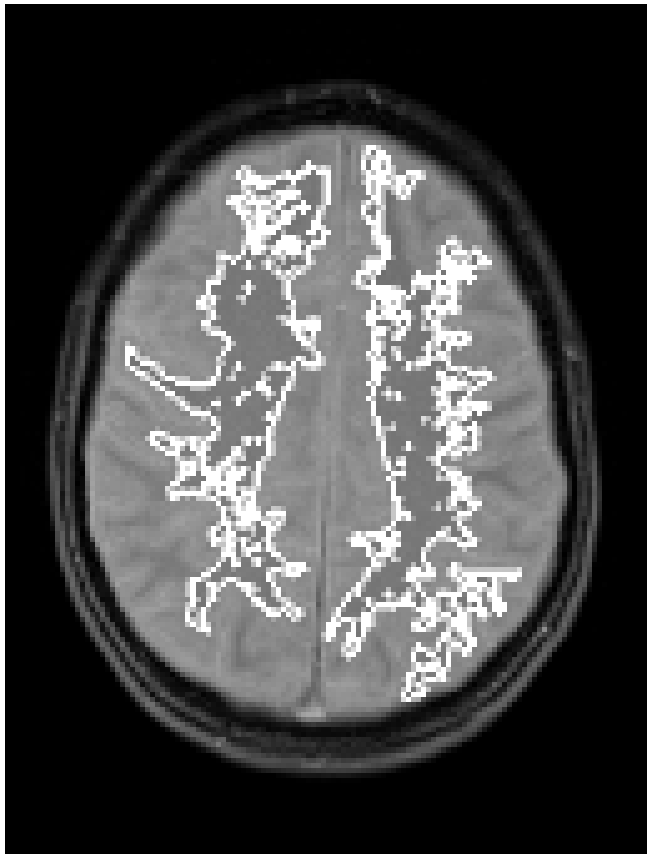


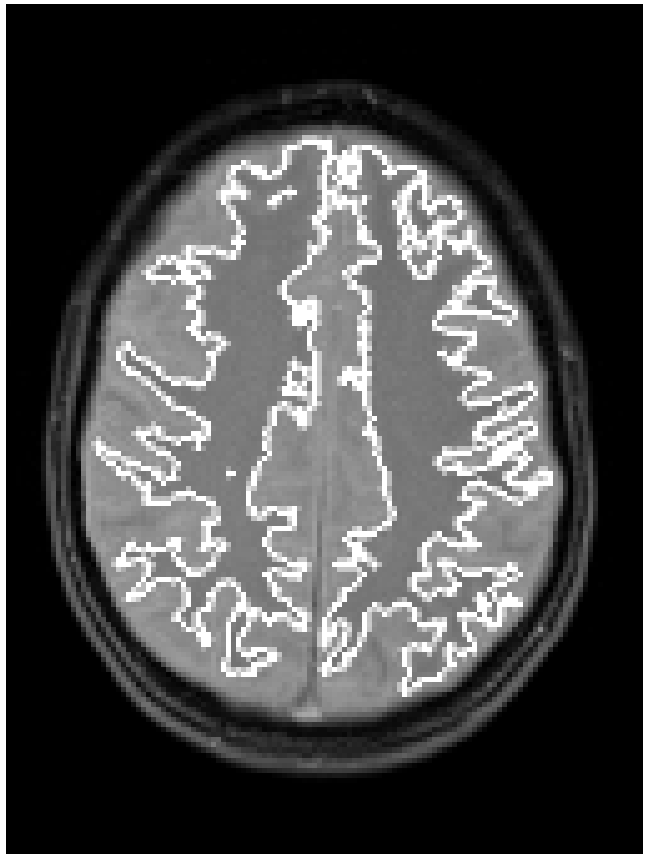
(b) Growing Process After a Few Iterations

Source

Region growing example



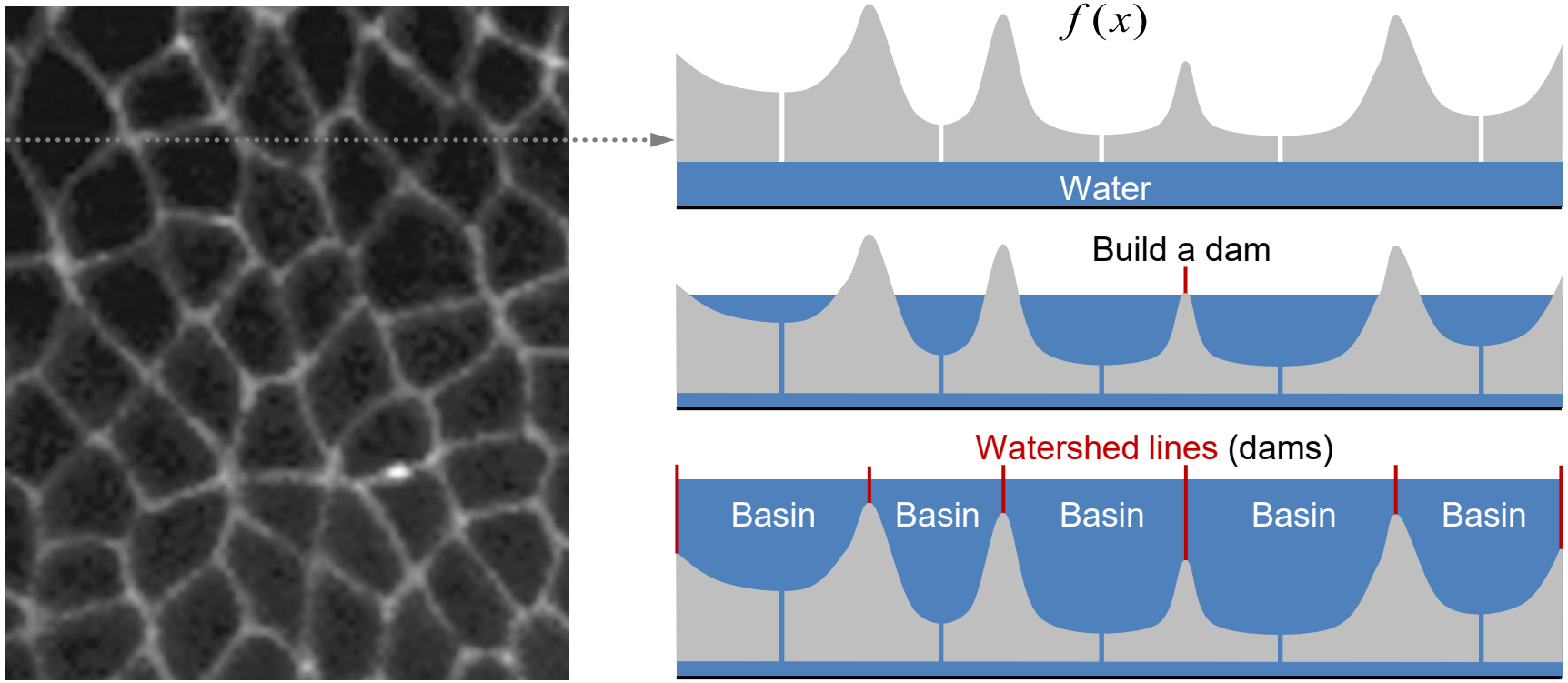




Source

2. Watershed segmentation

- Based on the analogy of immersion of a topographic surface



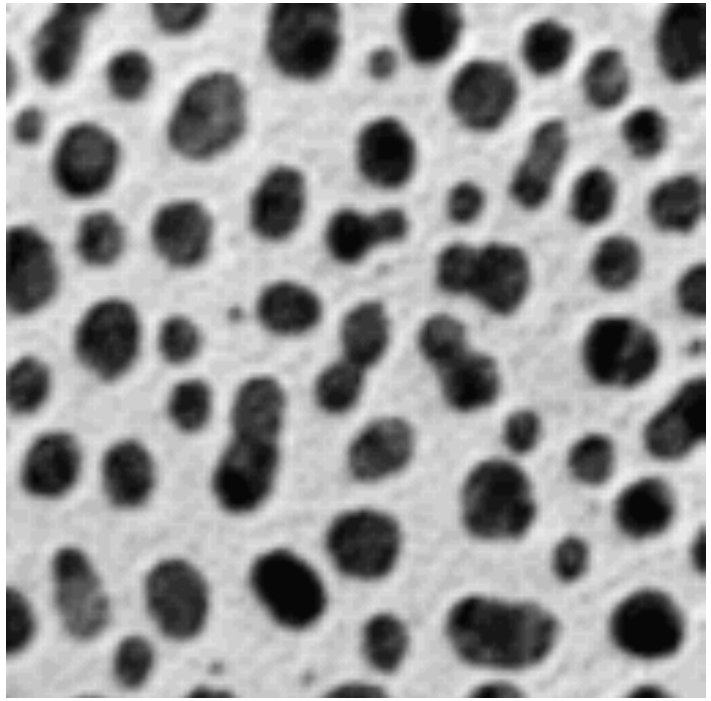
Watershed segmentation algorithm

Meyer's flooding algorithm

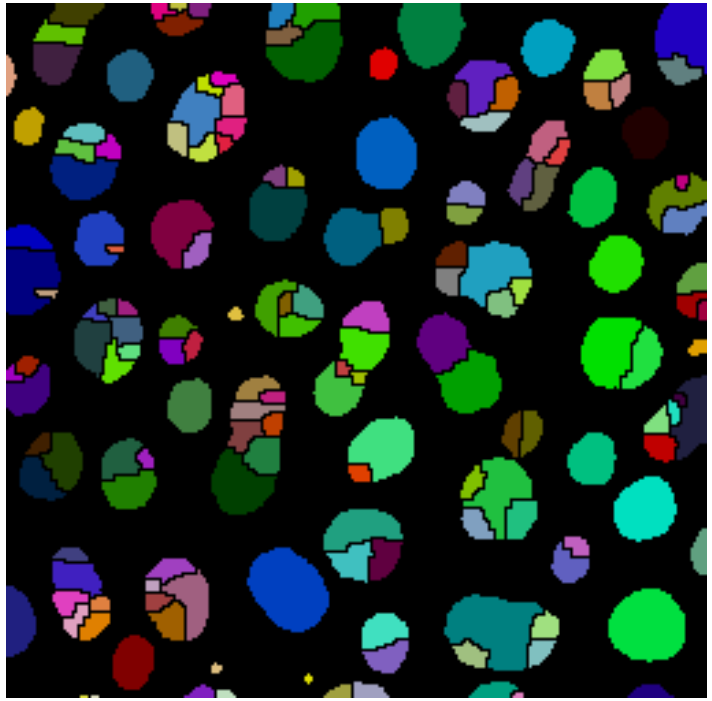
1. Choose a set of markers to start the flooding.
   Example: Local minima. Give each a different label.
2. Put neighbouring pixels of each marker into a priority queue.
   A pixel with more similar gray value has higher priority.
3. Pop the pixel with the highest priority level from the queue. If the neighbours of the popped pixel that have already been labelled all have the same label, then give the pixel that same label. Put all non-labelled neighbours that have never been in the queue into the queue.
4. Repeat step 3 until the queue is empty.

The resulting non-labelled pixels are the watershed lines

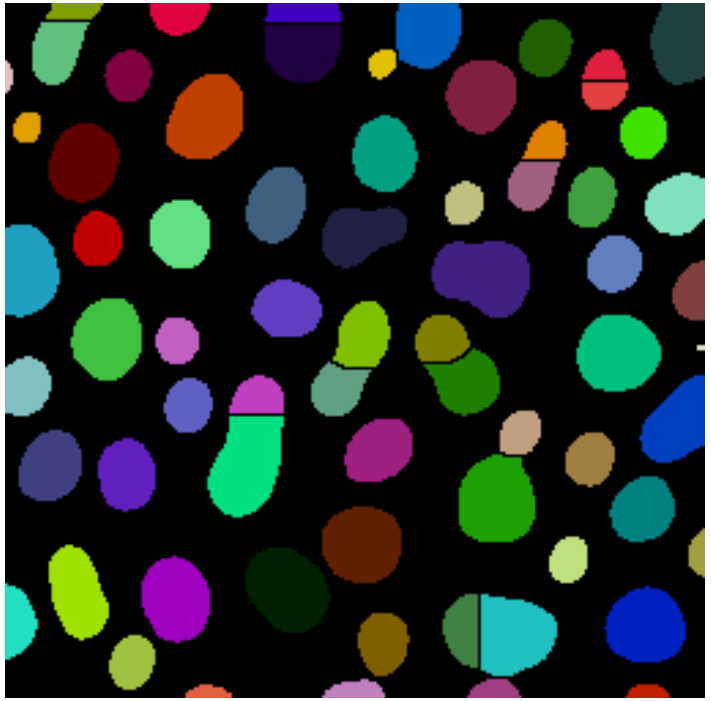
Watershed segmentation example



Input image



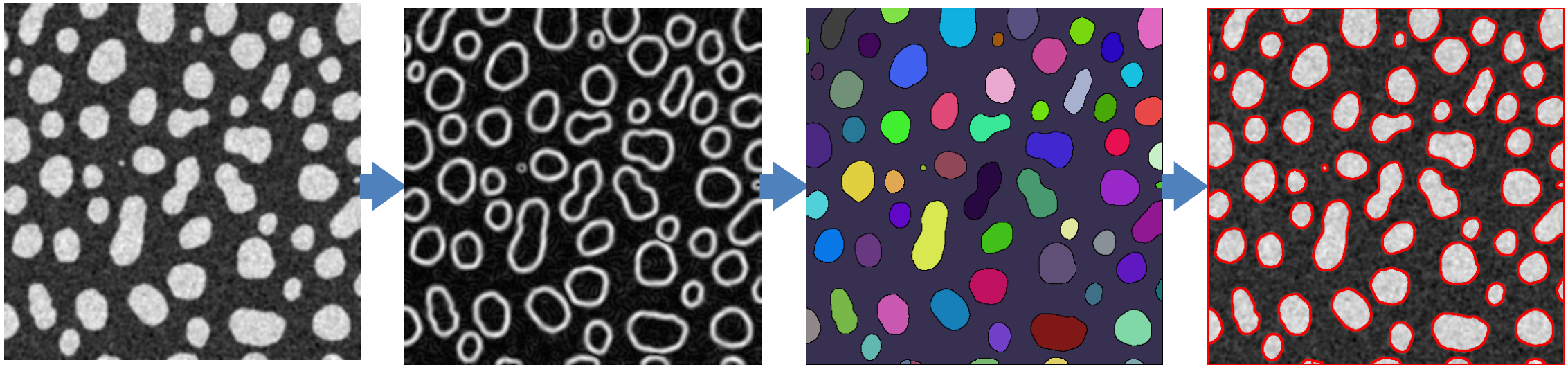
Watershed of input



Watershed of smoothed input

Preprocessing for watershed segmentation

- Invert the image or compute edges if needed to get local minima markers

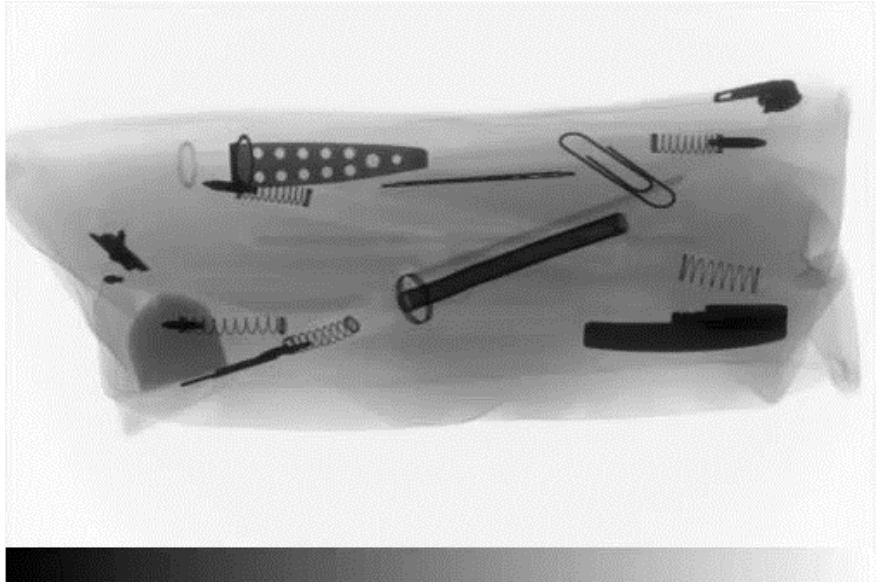


Watershed based segmentation notes

- Images often have many local minima, leading to heavy oversegmentation
- Preprocessing (image smoothing) may be needed to reduce false minima
- Postprocessing (basin merging) may be needed to reduce fragmentation
- Many different implementations and pre/postprocessing criteria exist
- It is possible to start from user-defined markers instead of local minima

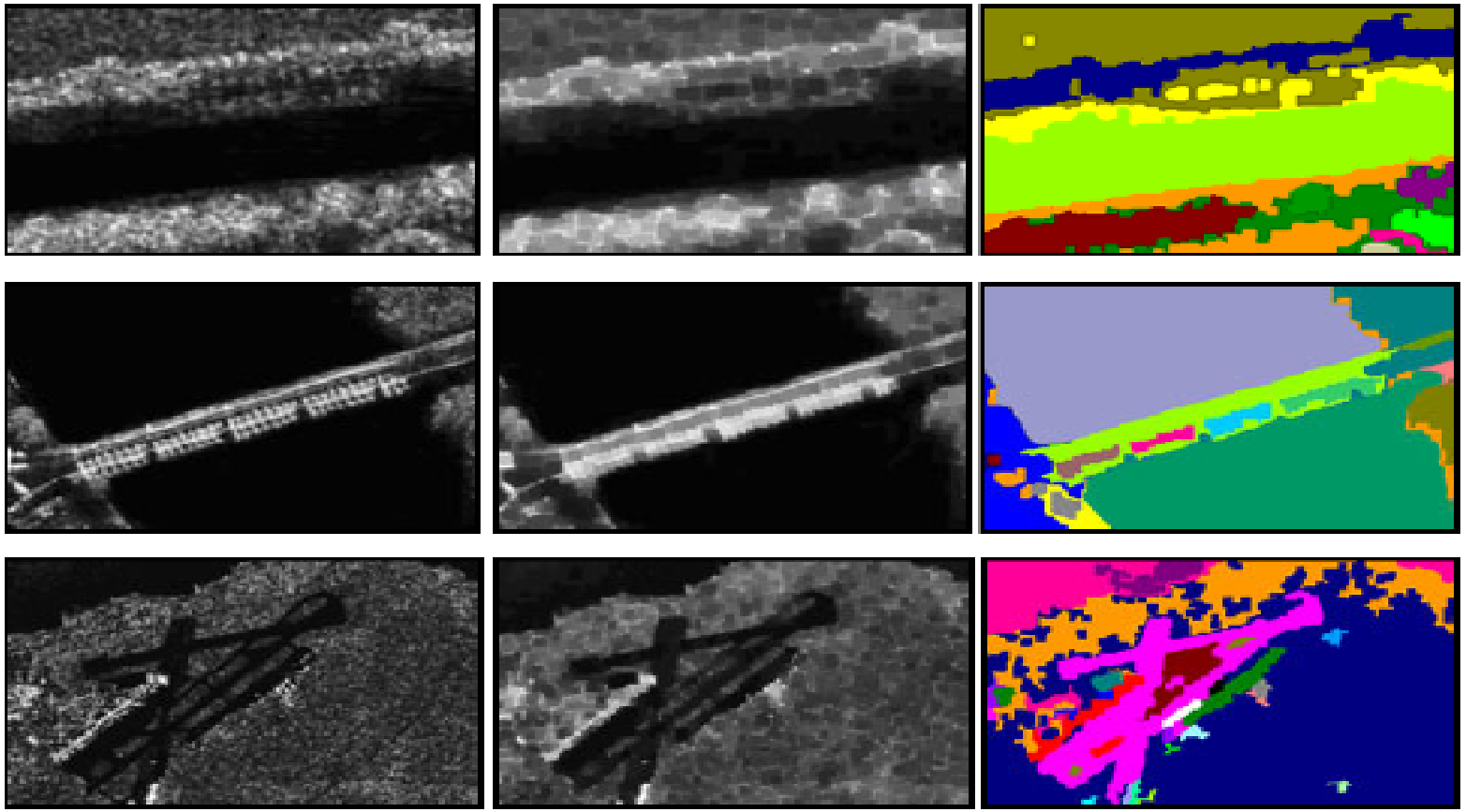
3. Maximally stable extremal regions

- Try multiple thresholds and analyse the shape of the connected components
- Select regions with virtually constant shape over many thresholds



https://www.youtube.com/watch?v=tWdJ-NFE6vY

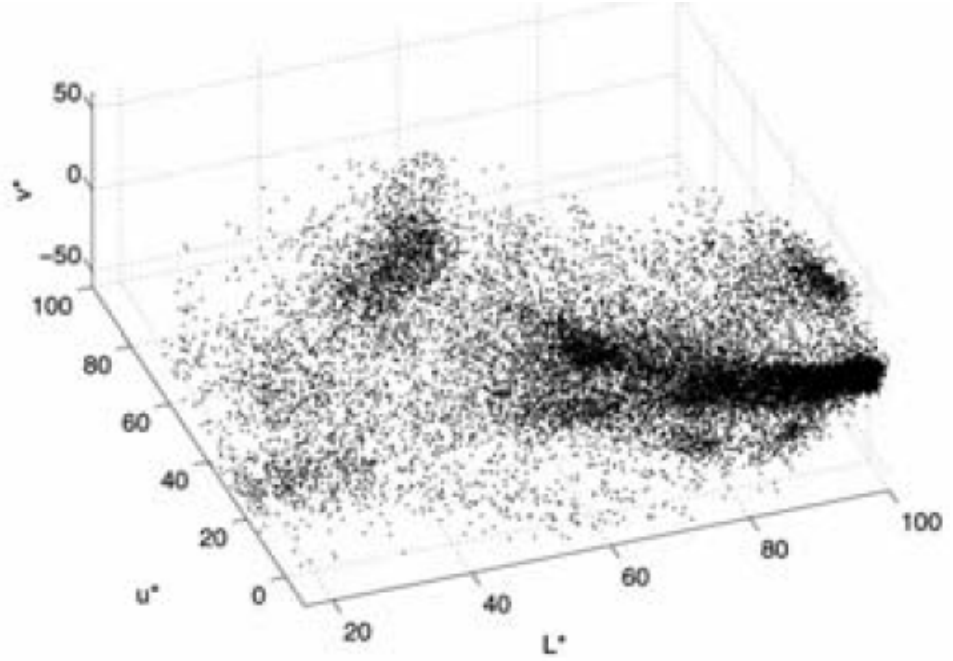
MSER segmentation examples



https://doi.org/10.1186/1687-6180-2012-83

4. Mean shifting

- How would you segment this image based on colour alone?



K-means clustering versus mean shifting

- K-means clustering has limitations
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
- Mean shifting is a good alternative in many applications
  - Seeks stationary points (peaks/modes) in a density function
  - Attempts to find all possible cluster centers in feature space
  - Does not require knowing the number of clusters a priori
  - Is a variant of iterative steepest-ascent method

Mean shifting algorithm

1. Initialize a random seed point x and window N
2. Calculate the mean (center of gravity) $m(x)$ within N

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

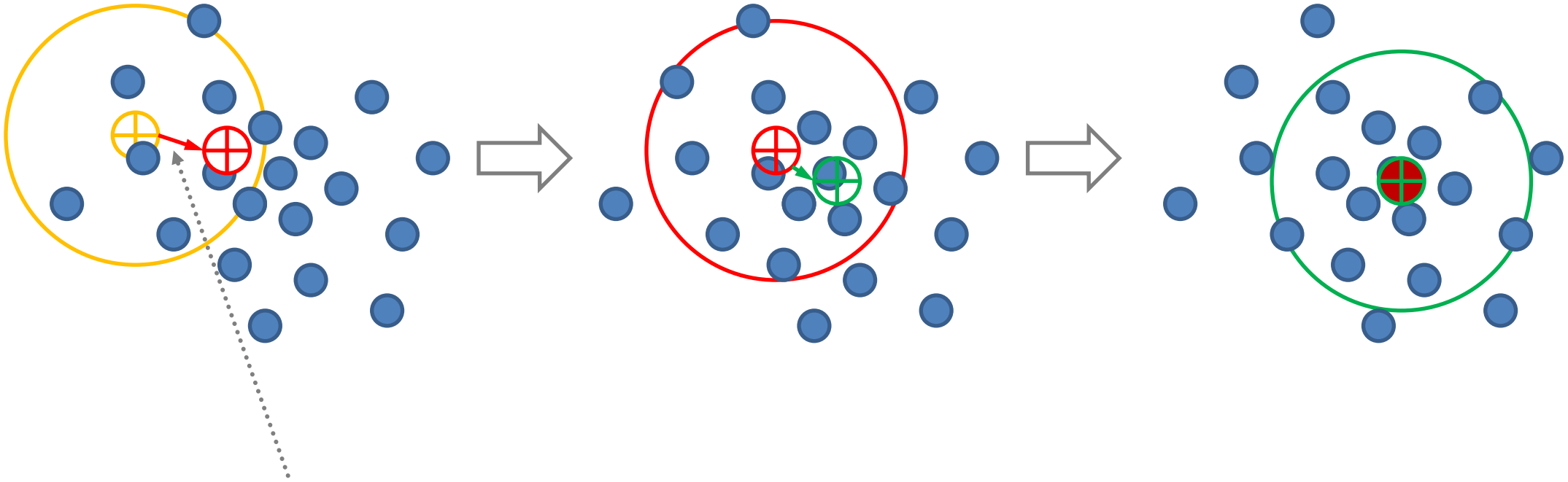
Mean (center of gravity)

$$K(x) = \exp(-|x|^2)$$

Kernel (weight function)

3. Shift the search window to the mean
4. Repeat Step 2 until convergence

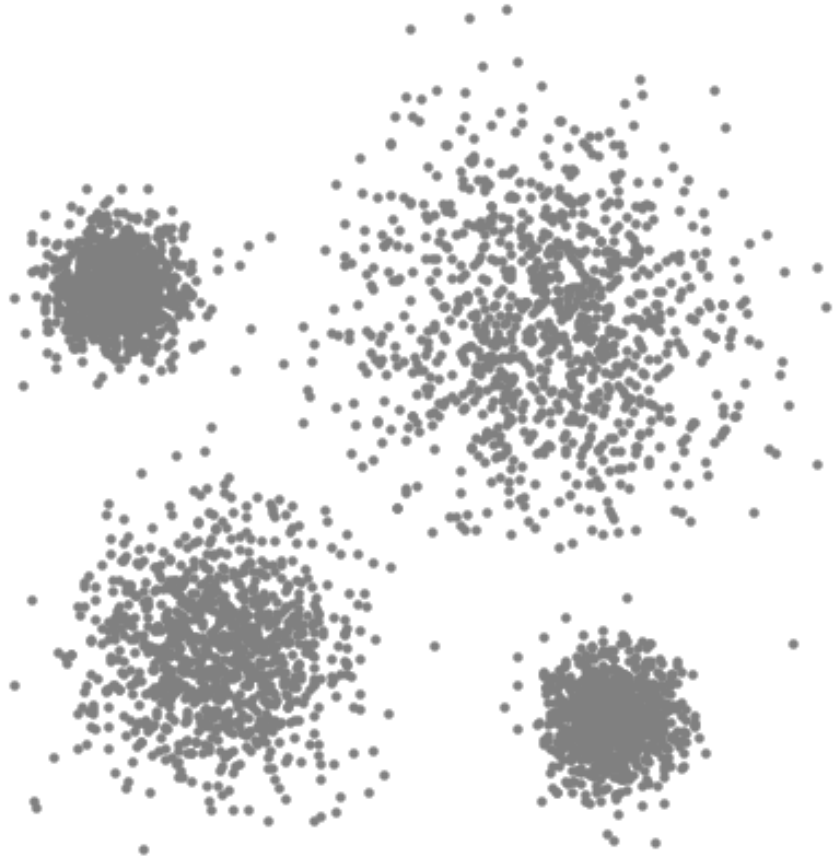
Mean shifting example



Mean shift vector $m(x) - x$

Mean shifting in action

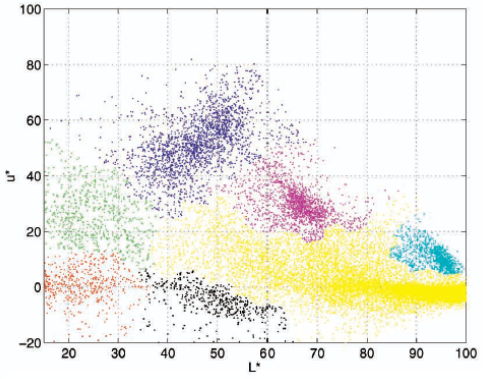
- Use a set of seed points to find all possible cluster centers
  - Initialise seeds on a regular grid
  - Iteratively apply mean shifting to all
  - Converged if seeds no longer move
  - Merge converged seeds if very close
  - Final (merged) seeds are cluster centres
  - Cluster cloud points accordingly

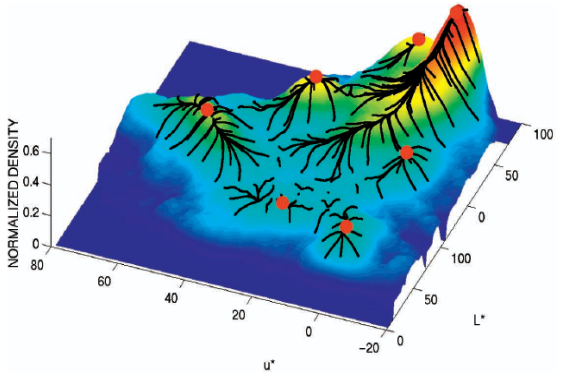


Source

Segmentation by mean shifting

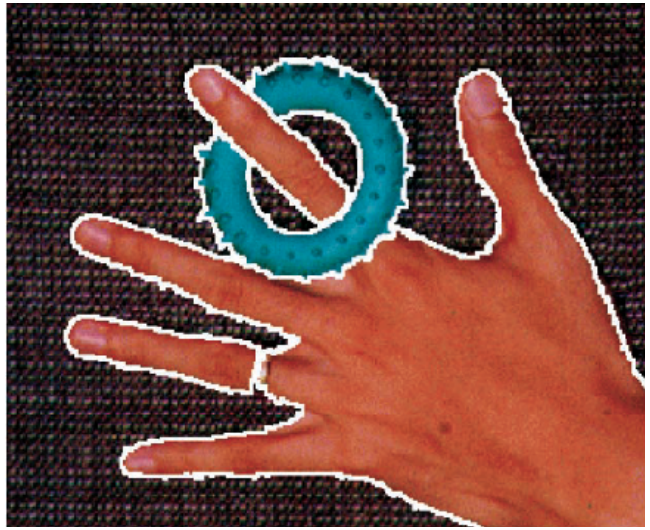
- Define features (colour, gradients, texture, et cetera)
- Transform image pixels to points in the feature space
- Initialise windows at multiple seed locations in feature space
- Perform mean shifting for each window until convergence
- Merge windows that end up near the same location
- Cluster all points according to window traversal





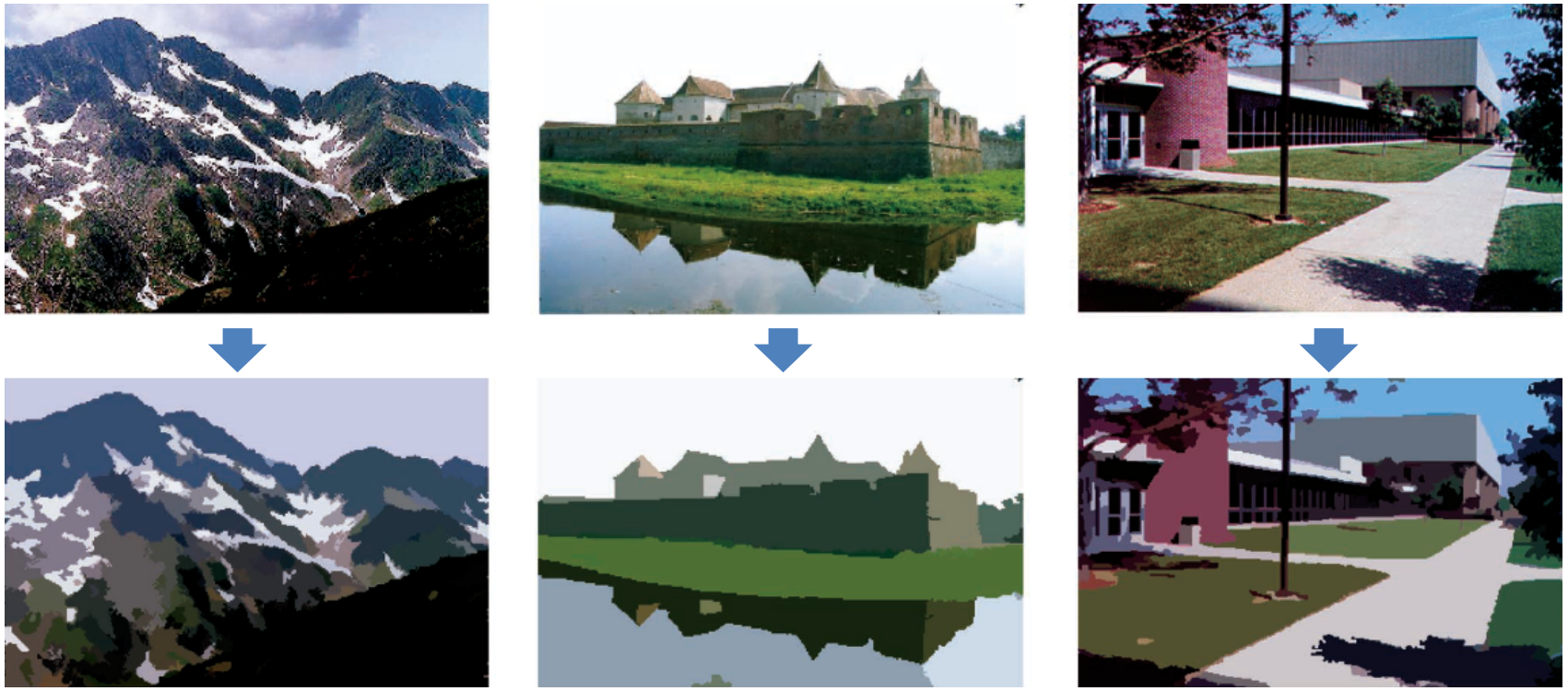
https://doi.org/10.1109/34.1000236

Mean-shift segmentation examples



Mean-shift segmentation art

- Replace segmented region colours by their cluster means



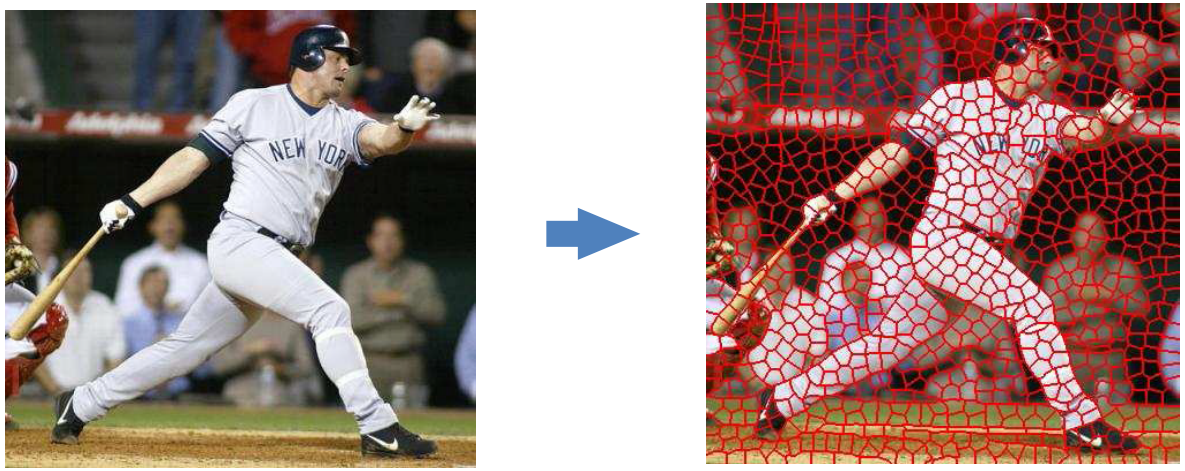
https://doi.org/10.1109/34.1000236

Mean-shift segmentation notes

- Advantages
  - Model-free (does not assume any prior shape on data clusters)
  - Has just a single parameter (window size)
  - Finds variable number of modes (clusters)
  - Robust to outliers
- Limitations
  - Computationally expensive (need to shift many windows)
  - Output depends on window size parameter value
  - Window size (bandwidth) selection is not trivial
  - Does not scale well with dimensionality of feature space

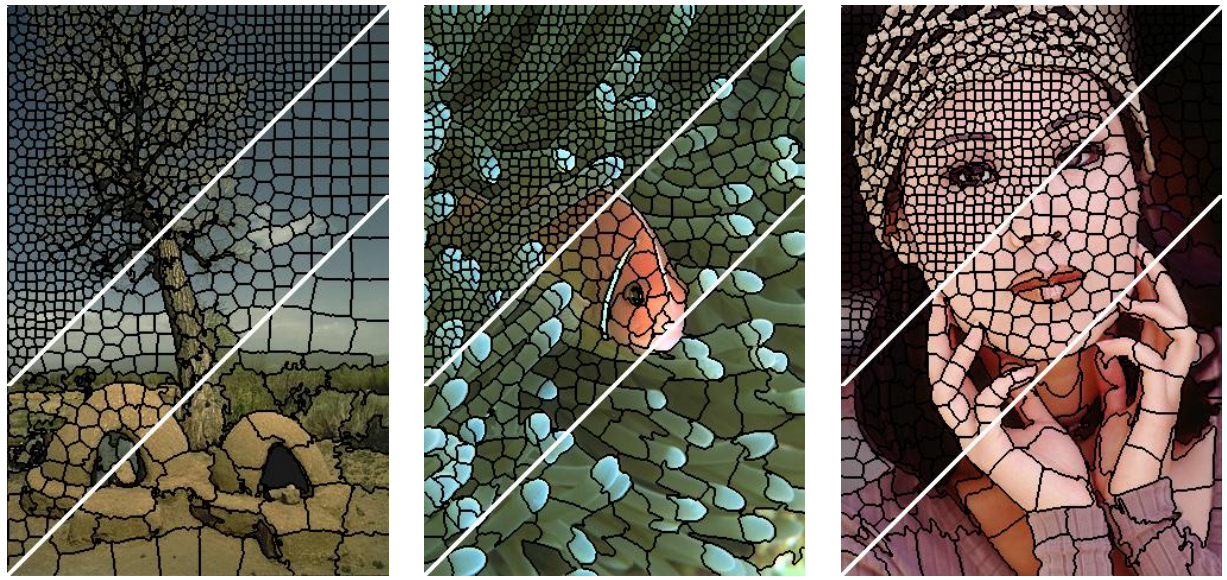
5. Superpixel segmentation

- Superpixel-based segmentation improves efficiency
  - Group similar pixels into a superpixel
  - Superpixels together are an oversegmentation of the image
  - Ultimate segmentation (classification, merging) performed on superpixels



Superpixel algorithm

- Simple linear iterative clustering (SLIC)
  - A popular superpixel generation algorithm
  - Preserves object boundaries, is fast, and memory efficient



Superpixels of size

64, 256, 1024 pixels

(approximately)

https://ivrl.epfl.ch/research-2/research-current/research-superpixels/

Simple linear iterative clustering (SLIC)

1. Initialise cluster centers C_j on pixel grid with step size S
2. Move C_j to position in 3×3 window with smallest gradient
3. Compute distance D_{ij} for each pixel i in $2S \times 2S$ window around C_j
4. Assign each pixel i to the cluster C_j with smallest distance D_{ij}
5. Recompute cluster centres as mean colour and position of pixels in C_j
6. Iterate (go to Step 3) until the residual error is small

https://doi.org/10.1109/TPAMI.2012.120

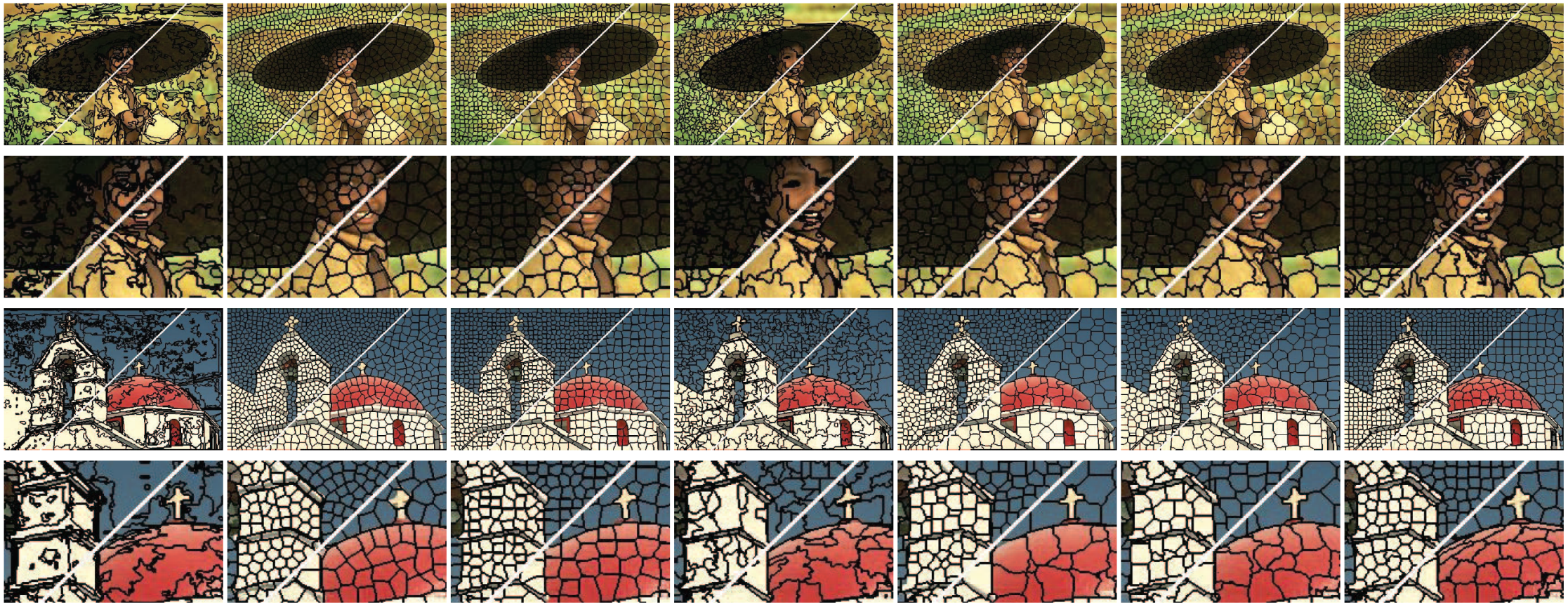
$$D = \sqrt{\frac{d_{lab}^2}{m^2} + \frac{d_{xy}^2}{S^2}}$$

(weight m controls influence of colour over spatial distance)

$$d_{lab} = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$ (distance in CIELAB space)

$$d_{xy} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$ (distance in pixel space)

Comparison of superpixel methods



6. Conditional random field

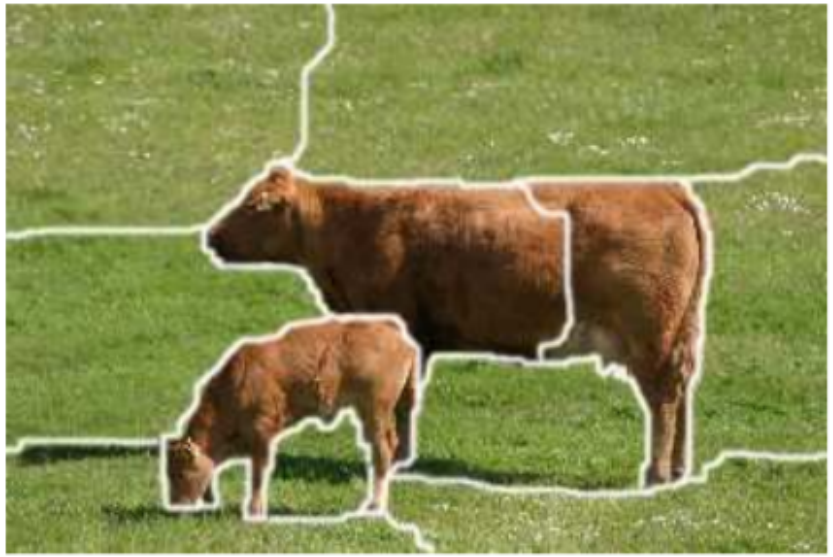
- Superpixels provide a basis for further segmentation
  - Determine spatial relationship between the superpixels
  - Compute similarities between superpixels
  - Group superpixels to form larger segments

- Conditional random field (CRF) approach

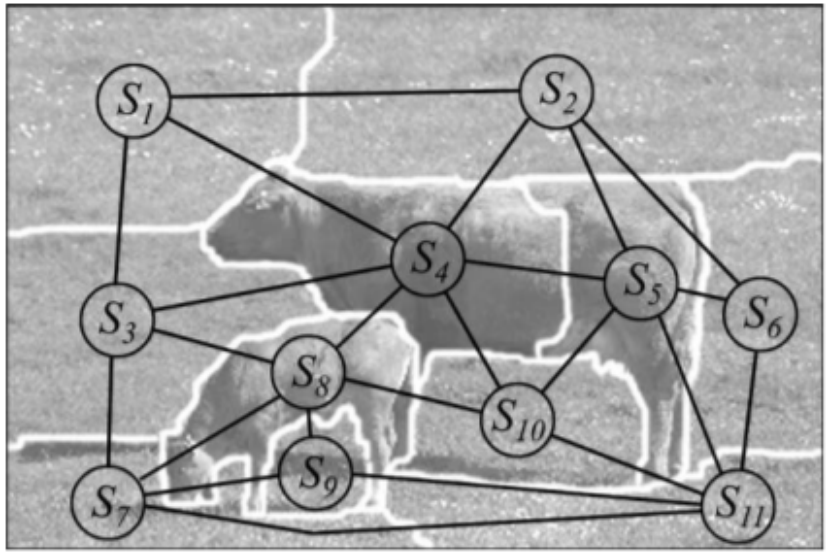
  A probabilistic graphical model that encodes the relationships between observations (i.e. superpixels) and constructs a consistent interpretation (i.e. segmentation) for a set of data (i.e. an image)

Graph representation of superpixels

- Nodes: superpixels (value based on features of superpixels)
- Edges: adjacency (value based on similarity between superpixels)



Superpixels



Graph

https://doi.org/10.1007/s11263-008-0140-x

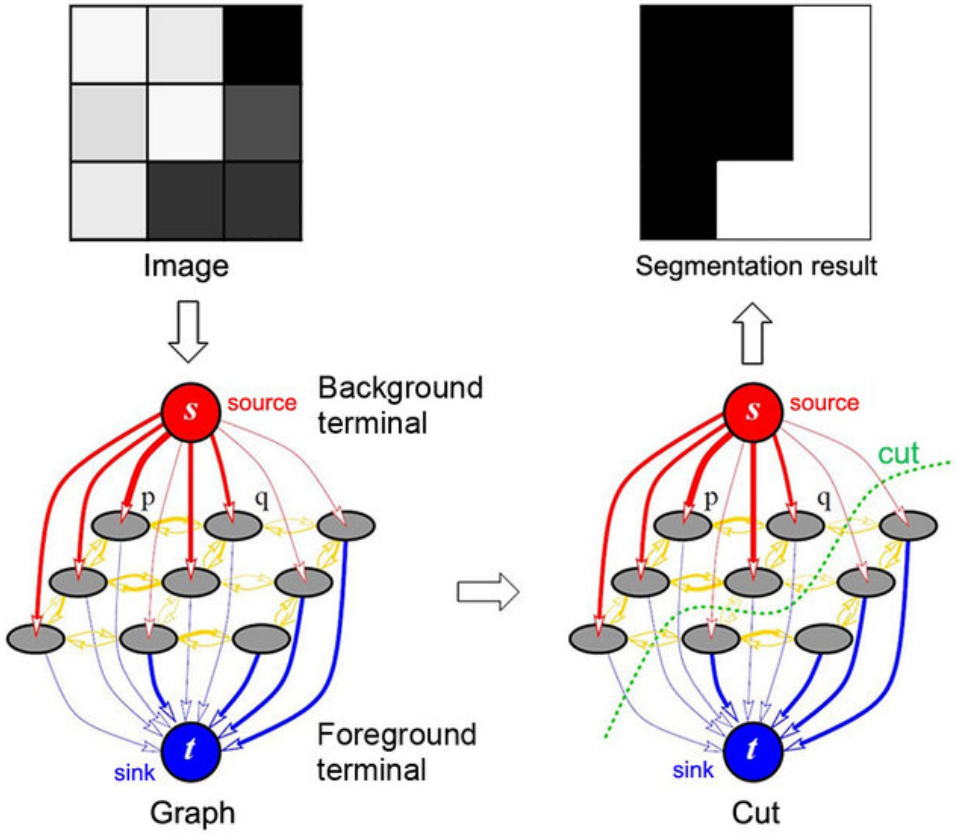
Segmentation by graph cutting

Formulated as an energy minimisation problem: $E(s,c) = \sum_i \varphi(s_i, c_i) + \sum_{ij} \psi(s_i, s_j)$

- Unary potentials φ
  - Data term based on graph node values
  - Computes the cost of superpixel s_i belonging to class c_i
  - A lower cost means a higher likelihood of s_i belonging to c_i
  - Can be obtained via superpixel classification
- Pairwise potentials ψ
  - Smoothness term based on graph edge values
  - Computes a cost of neighbourhood consistency
  - A cost is assigned if adjacent superpixels are assigned to different classes
  - Higher similarity results in a lower cost (0 if assigned to the same class)

Every attempted graph cut gives different values for φ and ψ and thus E

Graph cutting by min-cut/max-flow

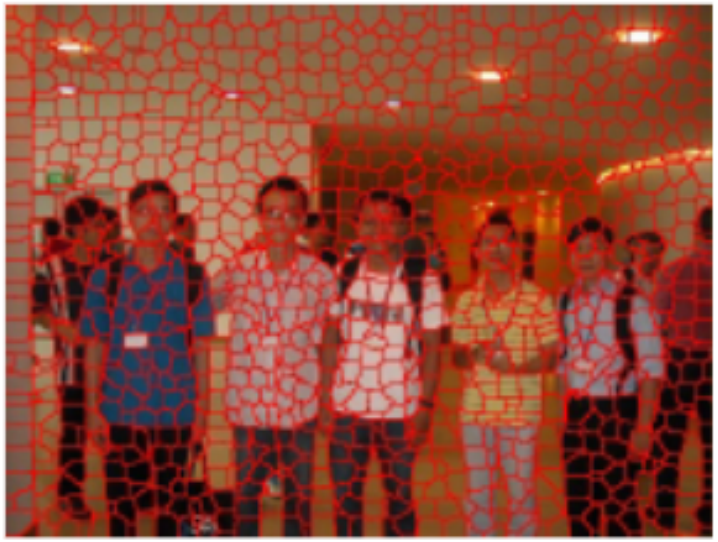


Maximising the flow from source to sink is equivalent to minimising the energy

https://doi.org/10.1109/TPAMI.2004.60

https://doi.org/10.1109/TGRS.2016.2627638

Graph cutting using multiple sources/sinks



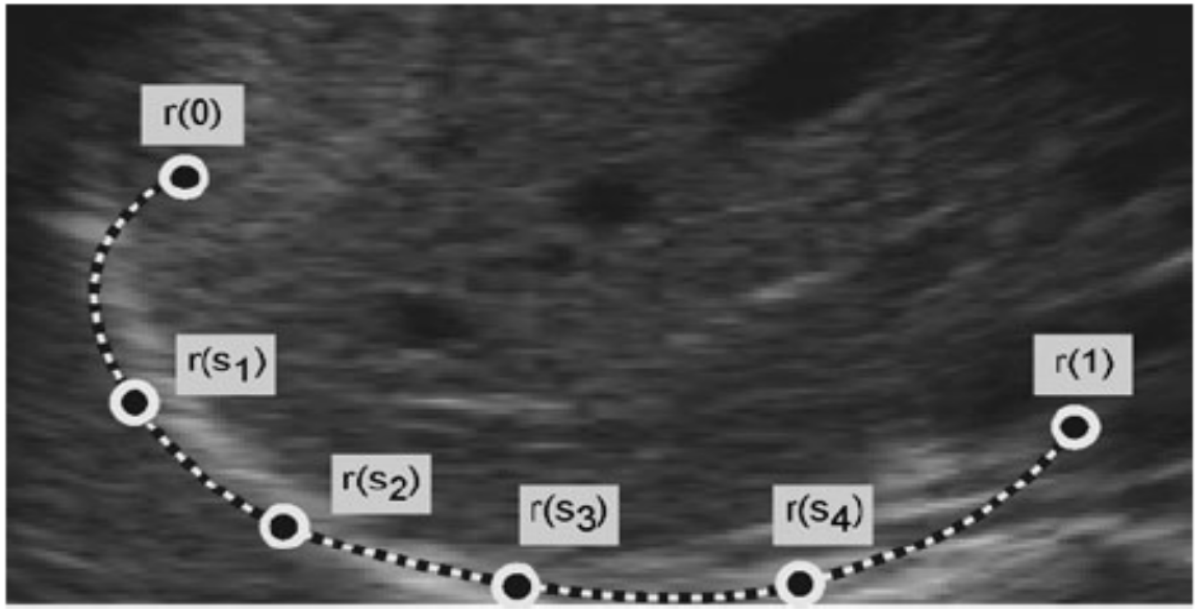
https://doi.org/10.1109/rivf.2012.6169870

7. Active contour segmentation

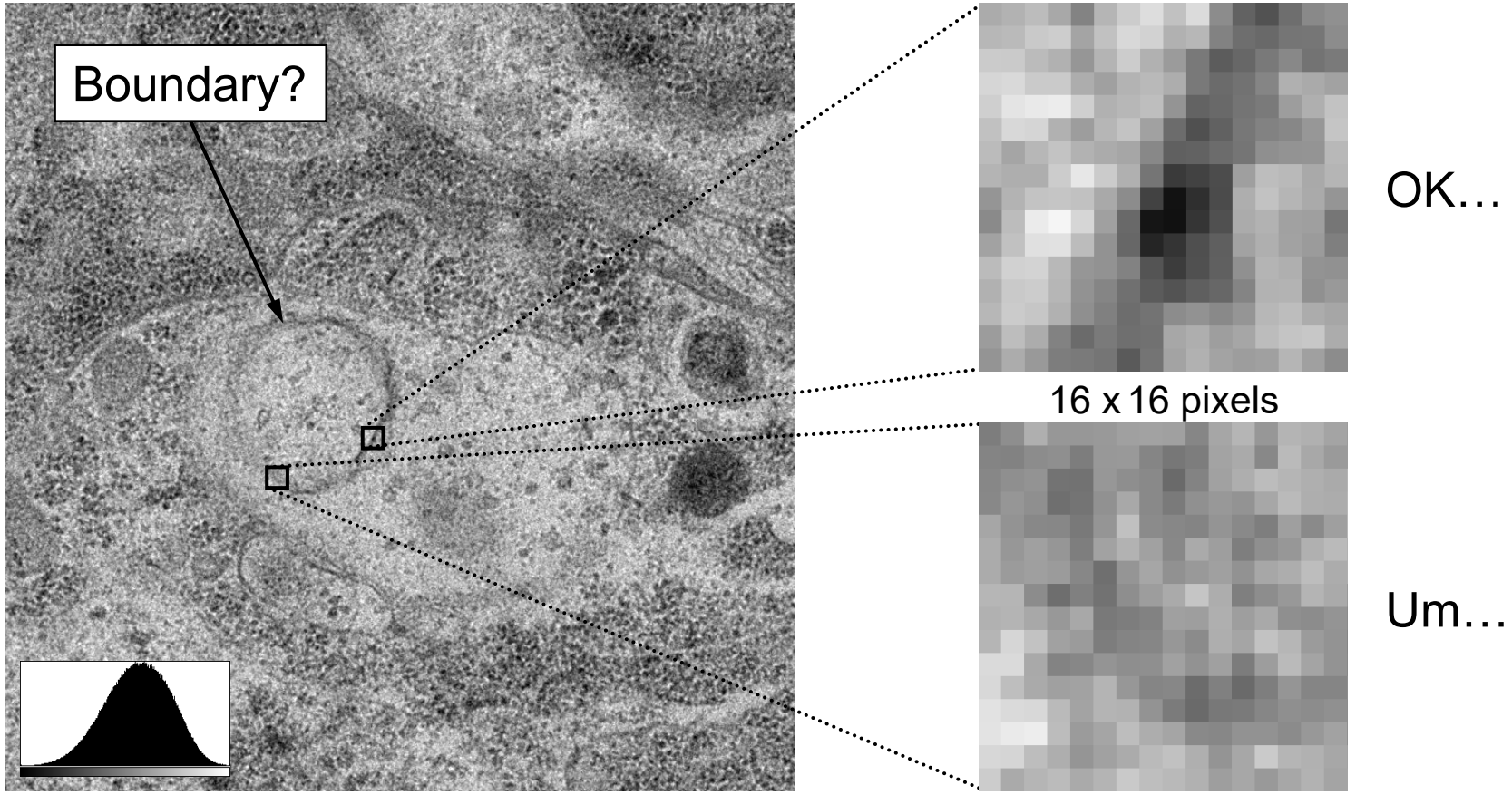
- A contour-based approach to object segmentation
- Aims to locate object boundaries in images by curve fitting
- Represents the curve by a set of control points and interpolation
- Iteratively moves the control points to fit the curve to the object
- Uses image, smoothness and user-guidance forces along curve

Also known as the snakes method

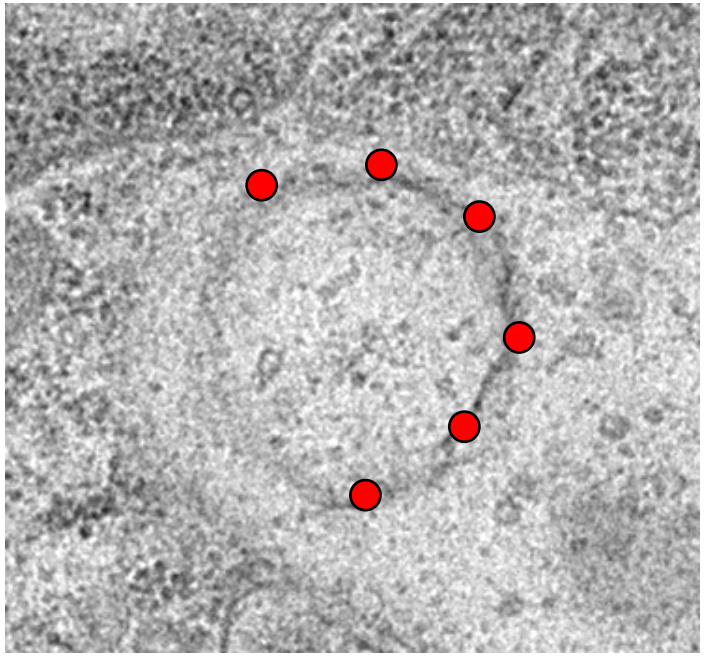
- Smoothly follows high intensity gradients at the object boundary
- Bridges areas of noise or missing gradients using smooth interpolation



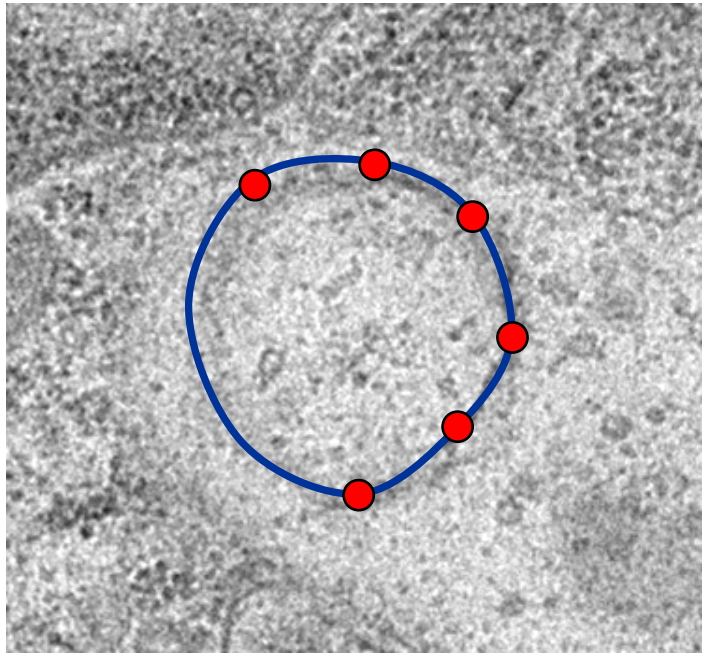
Missing gradient problem



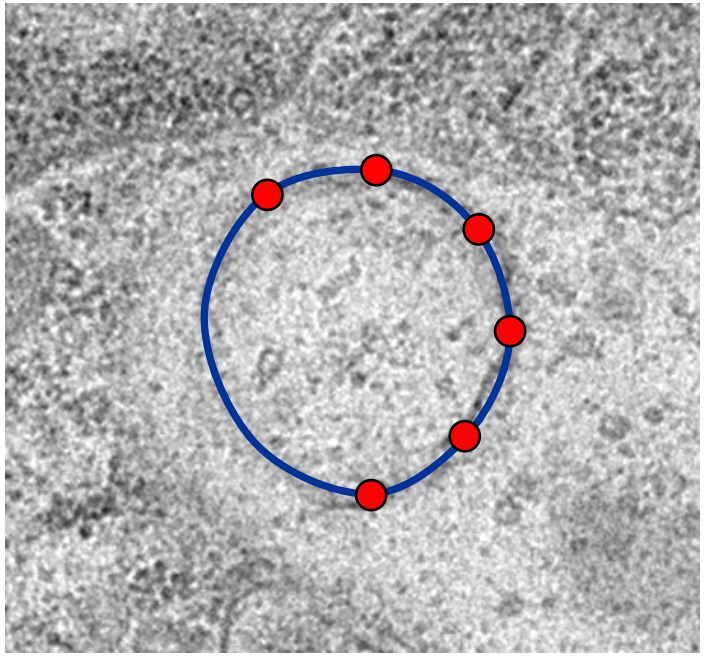
Enforcing a solution



Click control points



Interpolate minimum-energy contour



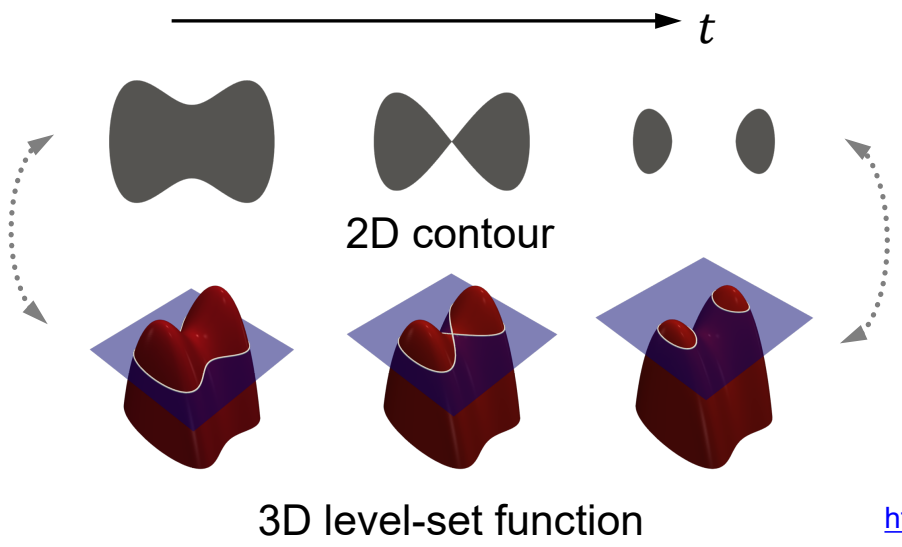
Further minimise energy by control point optimisation

8. Level-set segmentation

- Active contours / snakes are parametric models
  - Explicit representations of the object boundaries
  - Typically requires manual interaction to initialize the curve
  - It is challenging to change the topology of the curve as it evolves
  - Curve reparameterization may be required for big shape changes
- Level-set methods have become more popular alternatives
  - Implicit representations of the object boundaries
  - Boundaries defined by the zero-set of a higher dimensional function
  - Level-set function evolves to make the zero-set fit and track objects
  - Easily accommodates topological changes in object shape
  - Computationally more demanding than active contours

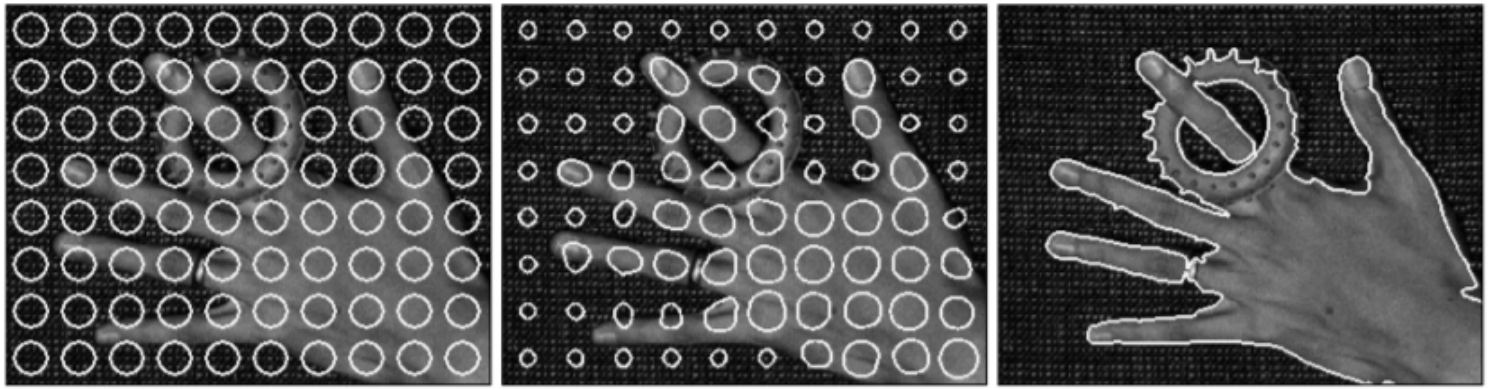
Level-set shape representation

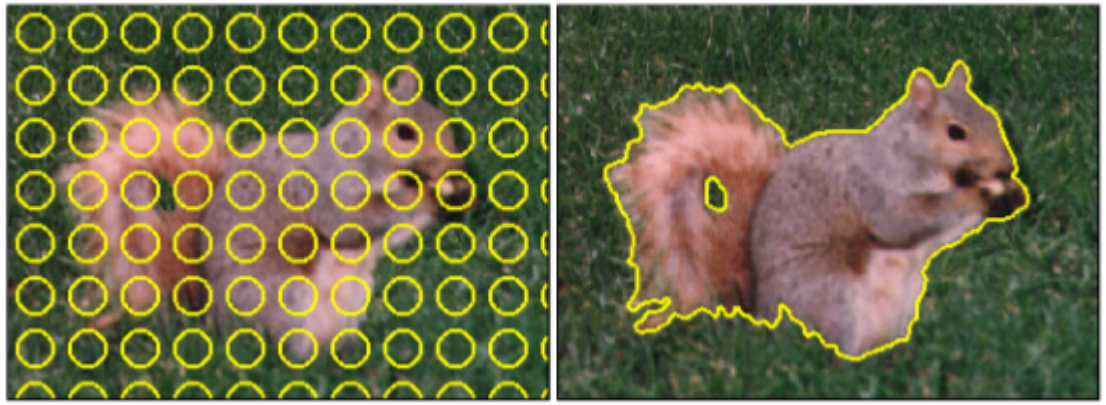
- Define an initial 3D level-set function
- Zero-level plane represents the 2D shape
- Iteratively deform the function to fit the shape



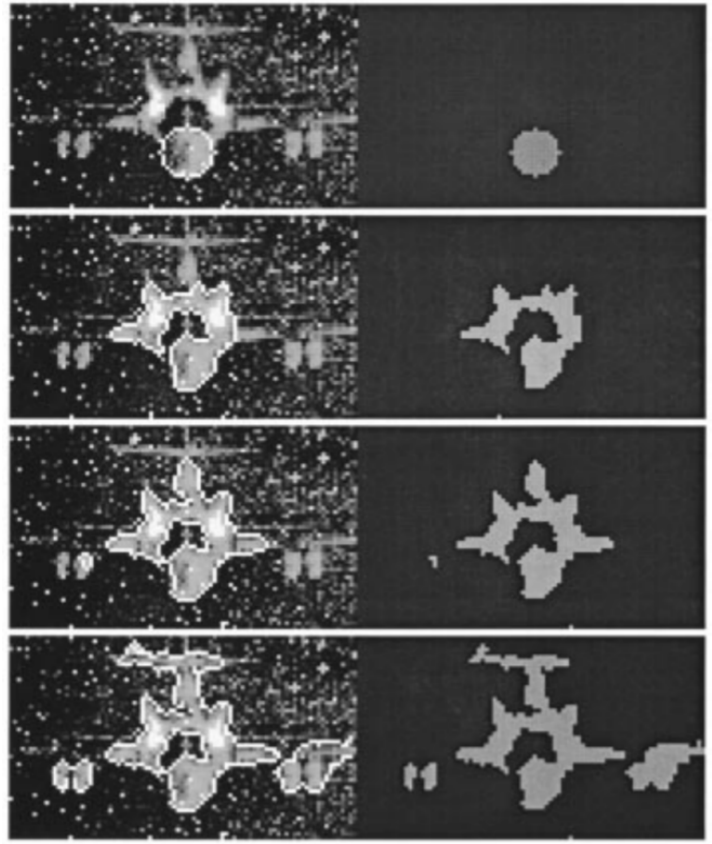
https://en.wikipedia.org/wiki/Level-set_method

Level-set segmentation examples





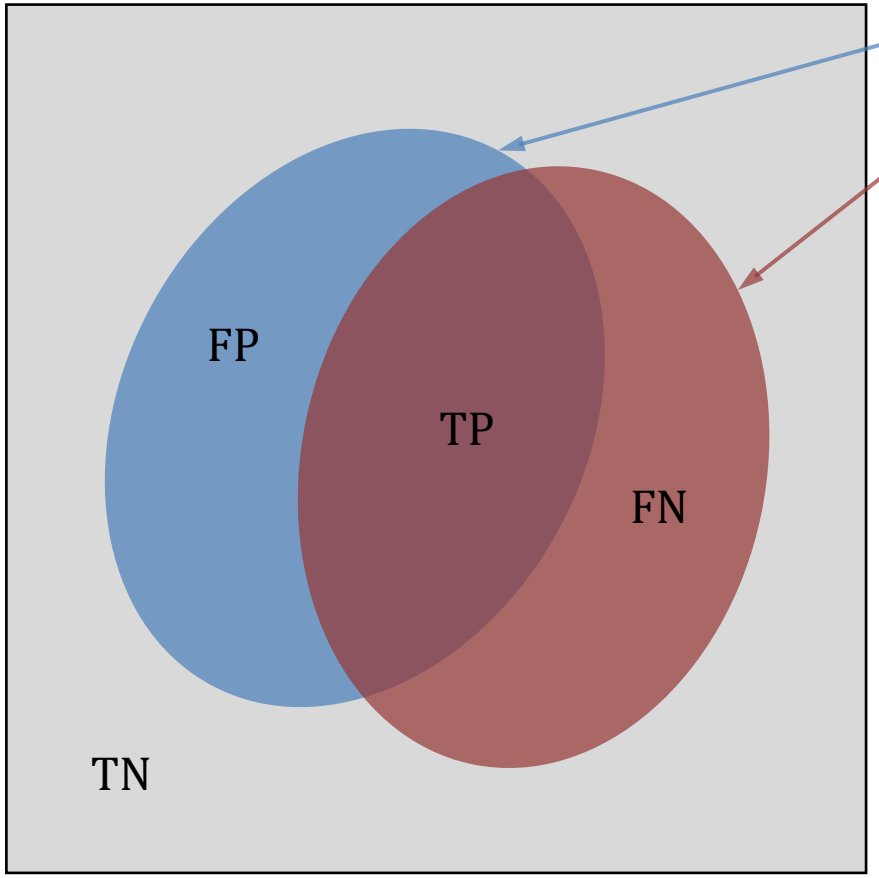
https://doi.org/10.1007/s11263-006-8711-1



https://doi.org/10.1109/83.902291

Evaluating segmentation methods

Classifying pixels using ground truth

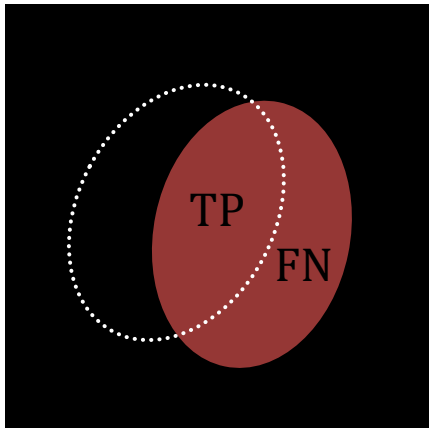


Segmented object pixels: S

True object pixels: T

- True positives: $\mathrm{TP} = S \cap T$
  Pixels correctly segmented as object
- True negatives: $\mathrm{TN} = S^c \cap T^c$
  Pixels correctly segmented as background
- False positives: $\mathrm{FP} = S \cap T^c$
  Pixels incorrectly segmented as object
- False negatives: $\mathrm{FN} = S^c \cap T$
  Pixels incorrectly segmented as background

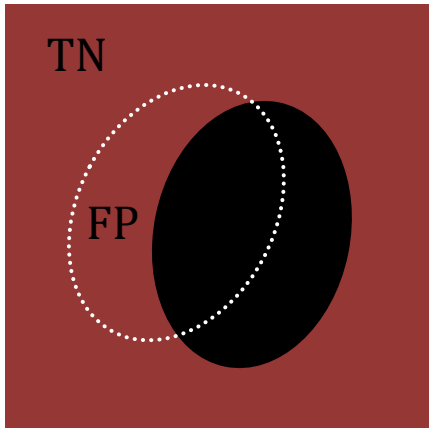
Sensitivity and specificity



- Sensitivity (= true-positive rate)

  Fraction of the true object that is correctly segmented

$$\mathrm{TPR} = \frac{|\mathrm{TP}|}{|\mathrm{TP}| + |\mathrm{FN}|}$$



- Specificity (= true-negative rate)

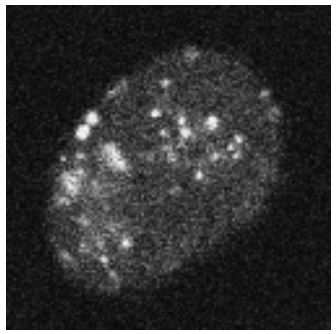
  Fraction of the true background that is correctly segmented

$$\mathrm{TNR} = \frac{|\mathrm{TN}|}{|\mathrm{TN}| + |\mathrm{FP}|}$$

Receiver operating characteristic (ROC)

Plot the true-positive rate (sensitivity) versus the false-positive rate (one minus the specificity) of a method as a function of its free parameter(s)

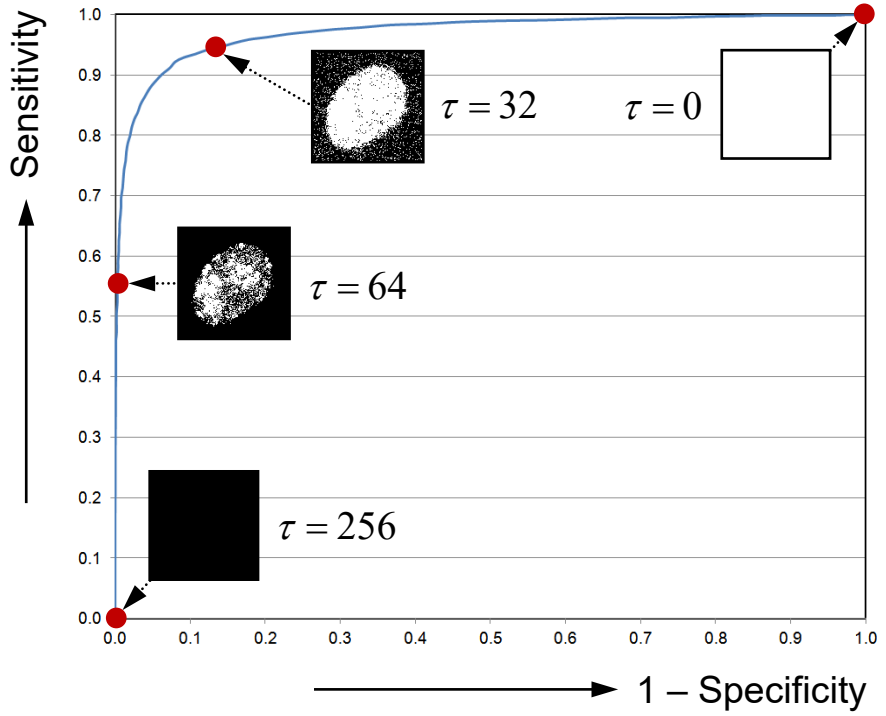
Example: Thresholding



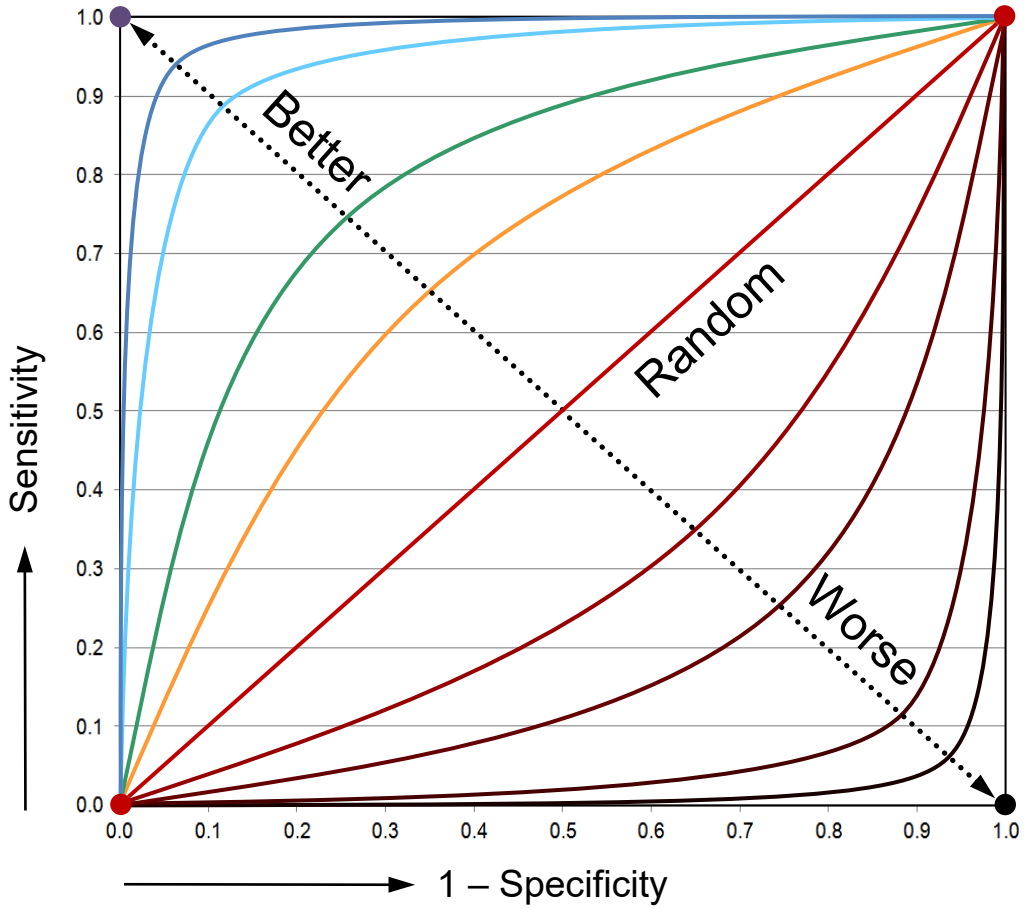
Image

Truth

Compute the sensitivity and specificity for all possible intensity thresholds τ and plot the results



Comparing methods by ROC analysis

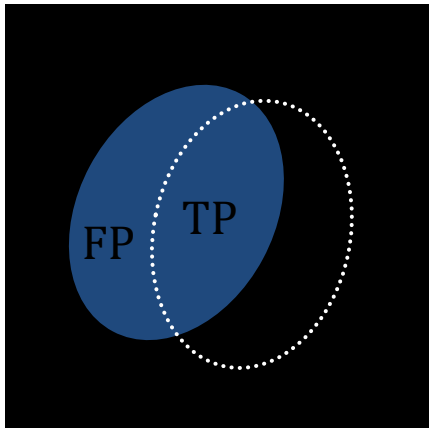


Area under the curve (AUC or AUROC)

Worst = 0.0
0.1
0.2
0.3
0.4
Random = 0.5
0.6
0.7
0.8
0.9
Best = 1.0

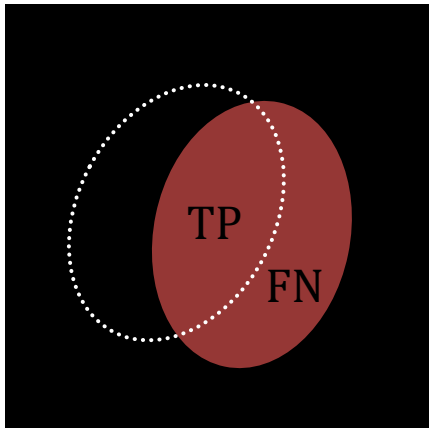
Higher AUROC = better method

Precision, recall, F-measure



- Precision (= positive predictive value)
  Fraction of the segmented object
  that is correctly segmented

$$P = \frac{|TP|}{|TP| + |FP|}$$



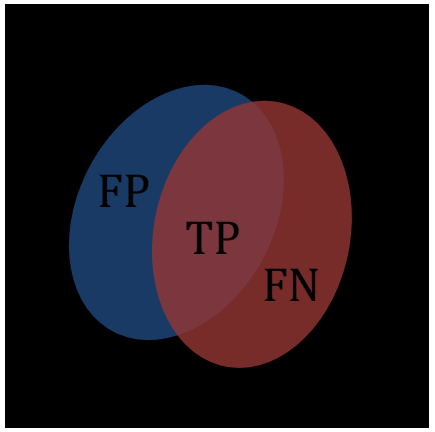
- Recall (= sensitivity)
  Fraction of the true object that is
  correctly segmented

$$R = \frac{|TP|}{|TP| + |FN|}$$

- F-measure
  Harmonic mean of
  precision and recall

$$F1 = \frac{2\,R\,P}{R + P}$$

Jaccard and Dice similarity coefficients



- Jaccard similarity coefficient (JSC)

  Intersection over union (IoU) = Correctly segmented fraction of the union of the segmented object and the true object

$$\mathrm{JSC} = \frac{|S \cap T|}{|S \cup T|} = \frac{|\mathrm{TP}|}{|\mathrm{FP}| + |\mathrm{TP}| + |\mathrm{FN}|}$$

- Dice similarity coefficient (DSC)

  Correctly segmented fraction of the segmented object set joined with the true object set

$$\mathrm{DSC} = \frac{2|S \cap T|}{|S| + |T|} = \frac{2|\mathrm{TP}|}{|\mathrm{FP}| + 2|\mathrm{TP}| + |\mathrm{FN}|}$$

Further reading on discussed topics

- Chapters 4, 6, 7 of Szeliski 2022
- Shapiro and Stockman 2001

Acknowledgements

- Some images drawn from Szeliski 2010
- Some images drawn from papers as indicated

Example exam question

Given the following binary image after segmentation.

To automatically identify the objects (in white) in this image we can use the connected components labelling algorithm.

How many separate objects will this algorithm find here if it uses 4-connectivity?

A. 4

B. 5

C. 6

D. 7