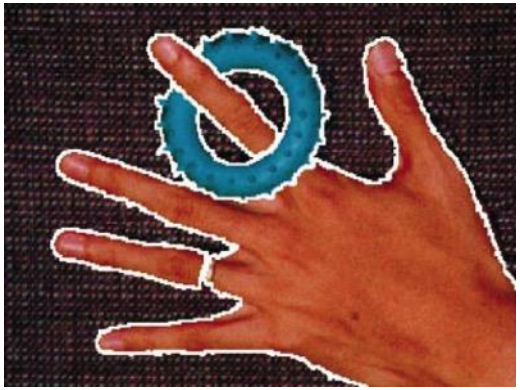
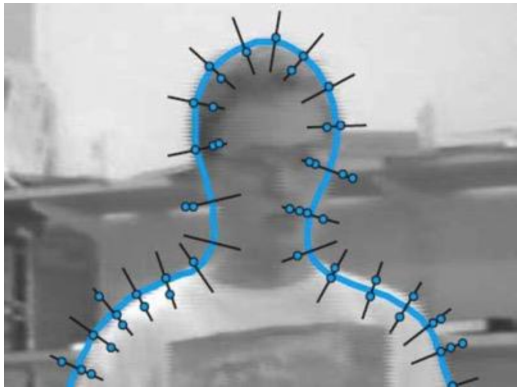




COMP9517: Computer Vision

Image Segmentation

Part 1

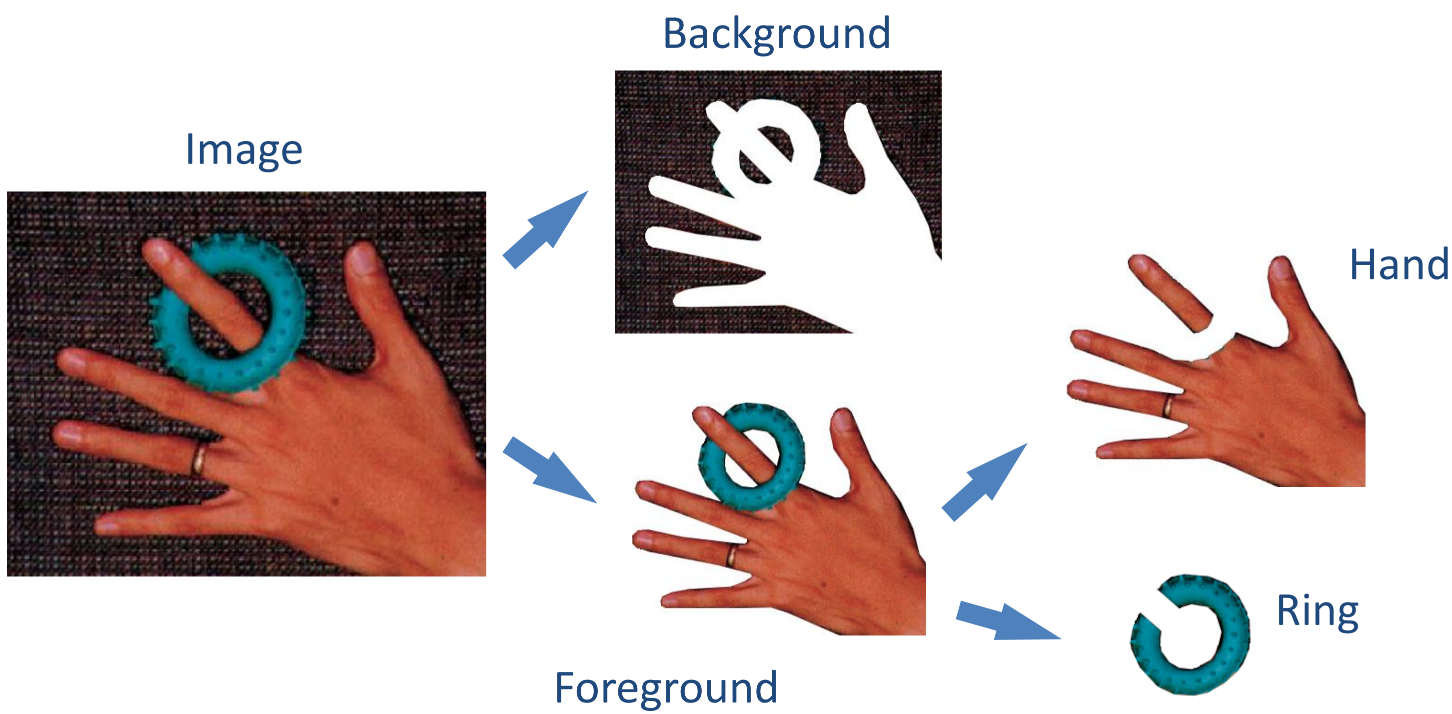
Introduction

- What do you see in this image?

Introduction

- Segmentation is the process of partitioning an image into a set of meaningful regions for further analysis

One of the oldest and most widely studied problems in computer vision



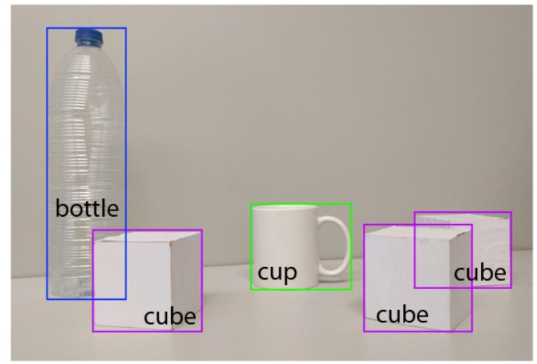
Introduction

- Region properties to facilitate image segmentation
  - Regions should be **uniform / homogeneous** in some characteristics
  - Region interiors should be **simple** and without holes or missing parts
  - Adjacent regions should have **significantly different** values in terms of the characteristics in which individually they are uniform
  - Boundaries of each region should be **smooth and spatially accurate**

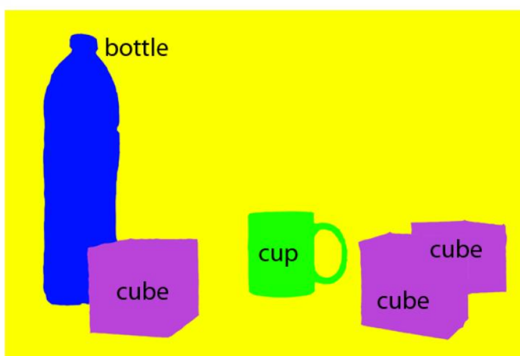
Introduction

- Different levels of region identification and segmentation

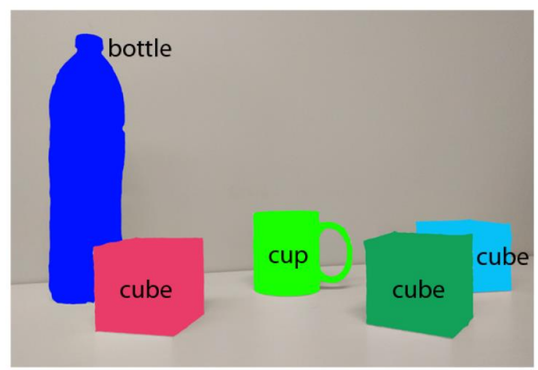
Image classification



Object localization



Semantic segmentation

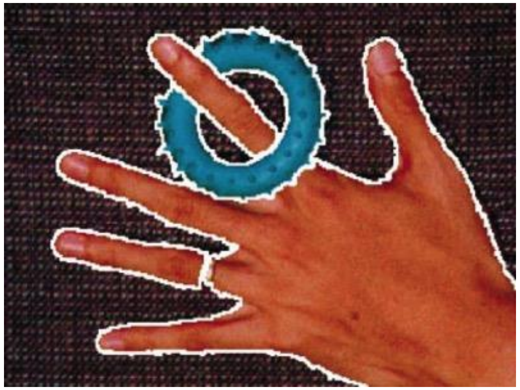
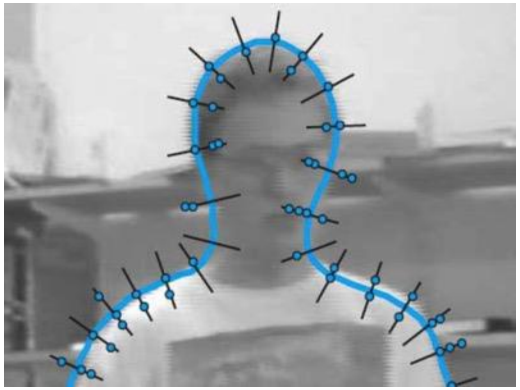


Instance segmentation

 https://arxiv.org/abs/1704.06857

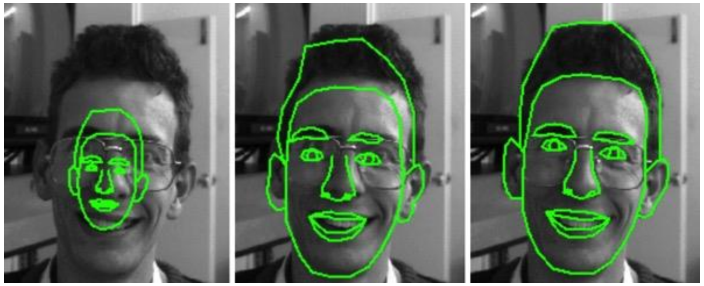
Introduction

- Segmentation approaches
  - Region based
  - Contour based
  - Template matching based
  - Splitting and merging based
  - Global optimisation based

Introduction

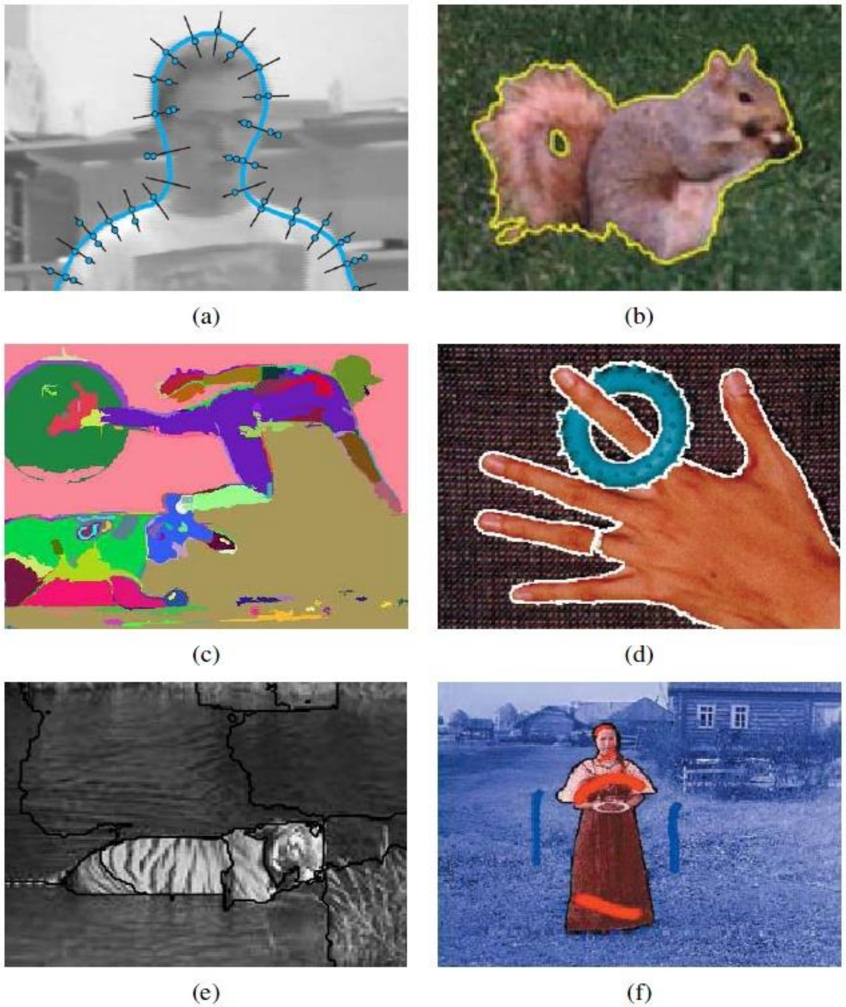
- Issues and challenges
  - So far there is **no single** segmentation method working well for all problems
  - Special **domain knowledge** of the application is typically essential for the development of successful computer vision methods for segmentation



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

Introduction

- Results from several popular segmentation techniques
  a) Active contours
  b) Level sets
  c) Graph-based merging
  d) Mean shift
  e) Normalised cuts
  f) Binary MRF



Outline

- Basic segmentation methods
  - Thresholding
  - K-means clustering
  - Feature extraction and classification

- More sophisticated segmentation methods
  - **Region splitting and merging**
  - **Watershed segmentation**
  - **Maximally stable extremal regions**
  - **Mean-shift algorithm**
  - **Superpixel segmentation**
  - **Conditional random field**
  - Active contour segmentation
  - Level-set segmentation

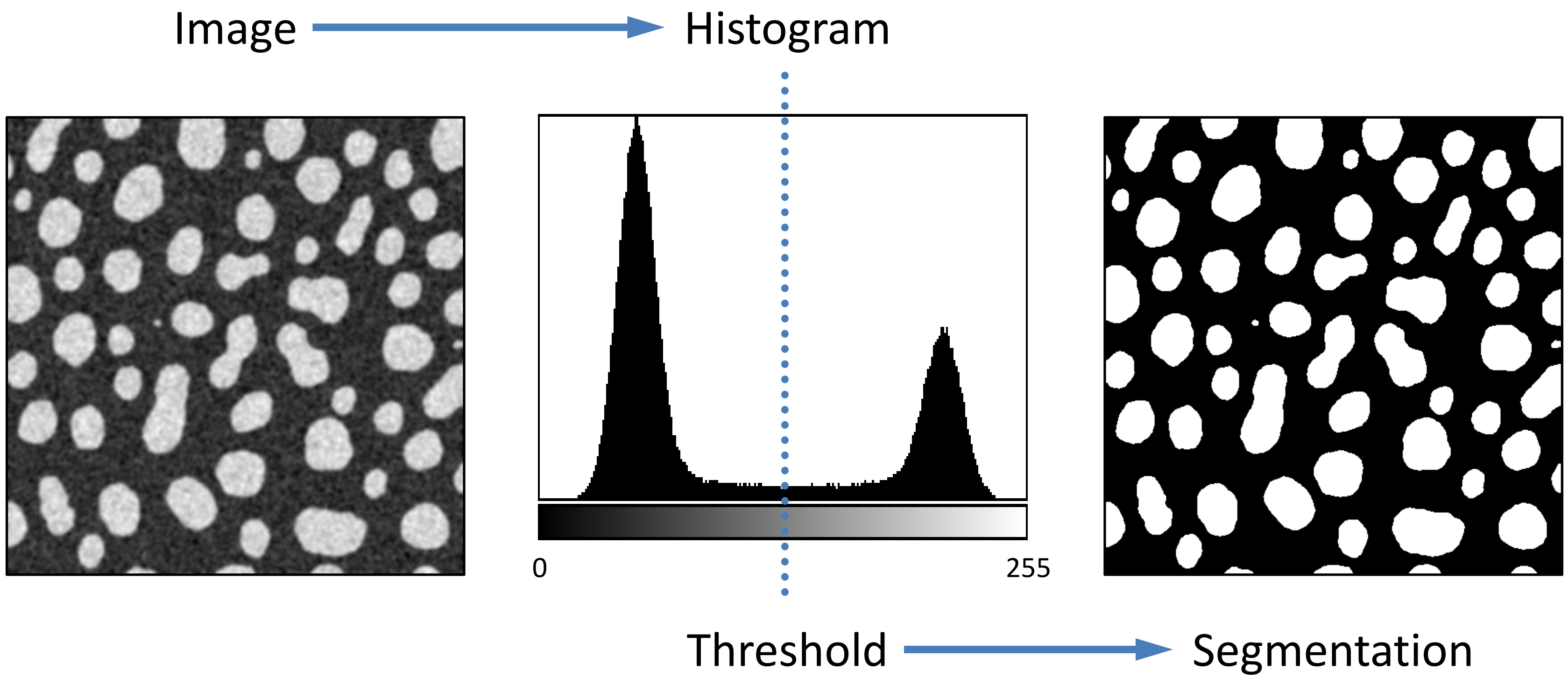
Outline

- Region Representation
  - Labelled images (the most commonly used)
  - Overlays
  - Boundary coding
  - Quad trees
  - Property tables

- Evaluating segmentation methods
  - Quantitative evaluation metrics
  - Receiver operating characteristic

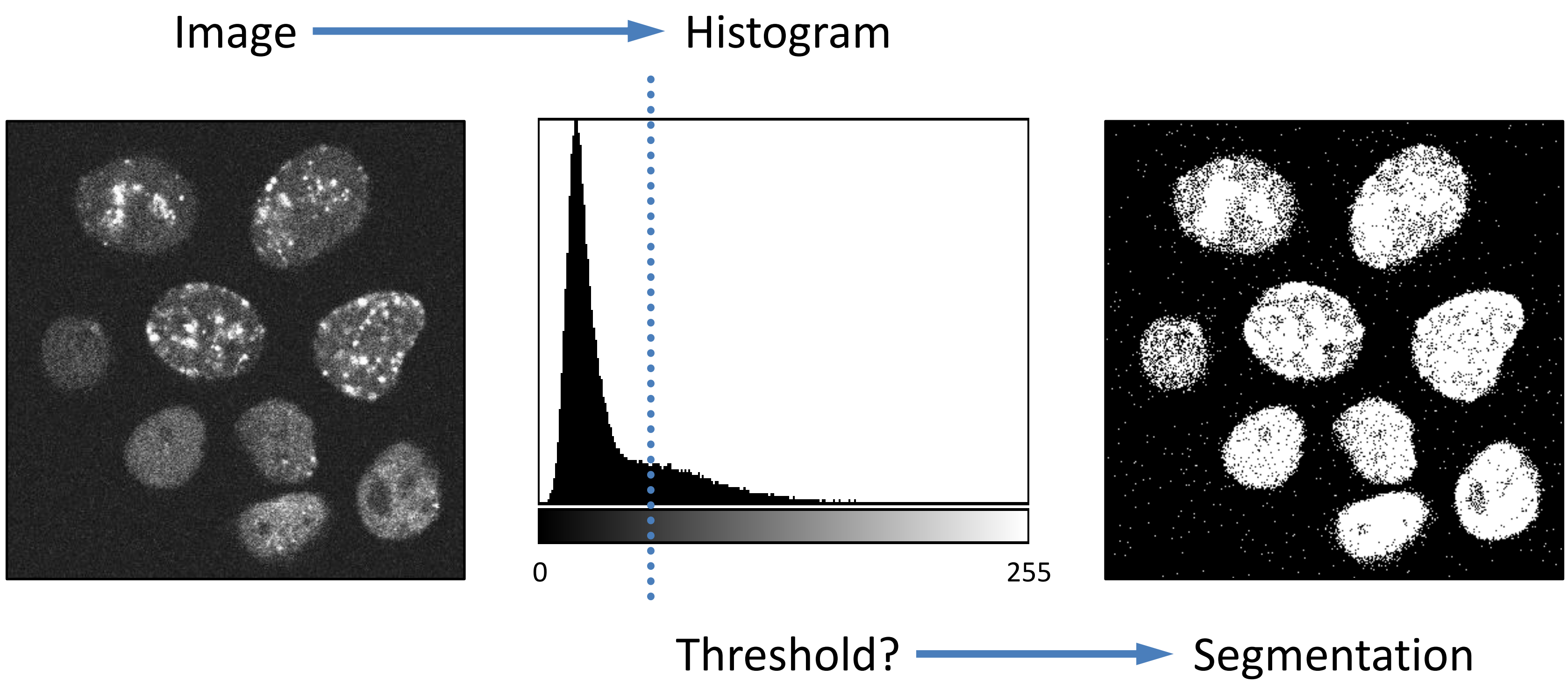
Thresholding

- Fine if regions have sufficiently different intensity distributions



Thresholding

- Problematic if regions have overlapping intensity distributions



K-Means Clustering

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

Original Image

Labeled Image

Feature Based Classification

- Segmentation by sliding-window patch-wise feature extraction and then classification… requires many examples for training

Original Image Ground Truth Labeled Image

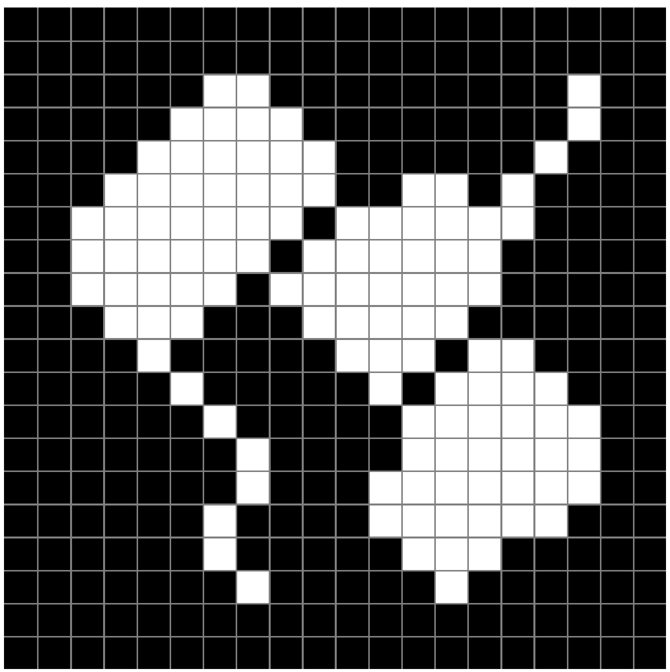
Schroff et al. [Single-histogram class models for image segmentation](). In: Computer Vision, Graphics and Image Processing, Springer, 2006.

Region Splitting and Merging

- Overview
  - Recursively split the whole image into regions based on region statistics
  - Recursively merge regions together in an hierarchical fashion
  - Combine splitting and merging sequentially

Region Splitting and Merging

- The simplest possible technique
  - Apply thresholding and then compute connected components
  - Rarely sufficient due to lighting and intra-object statistical variations

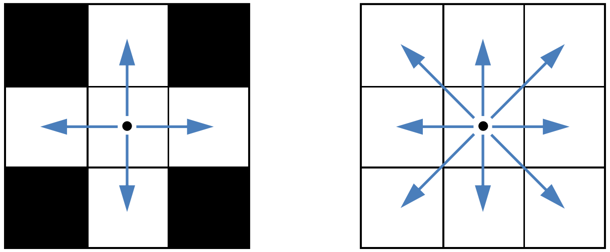


$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t \\ 0 & \text{else} \end{cases}$$

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

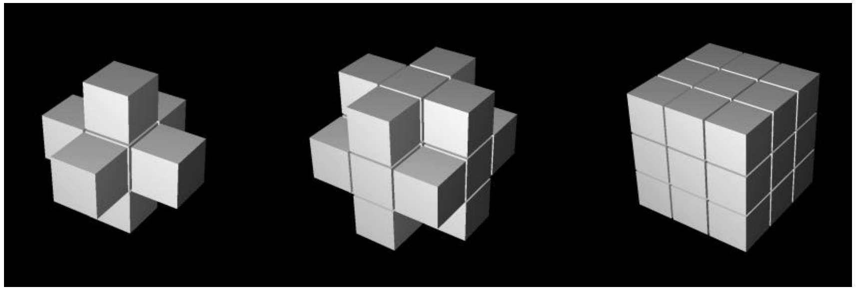
Connected Components

- Connectivity in two dimensions (2D)



4-connected 8-connected

- Connectivity in three dimensions (3D)

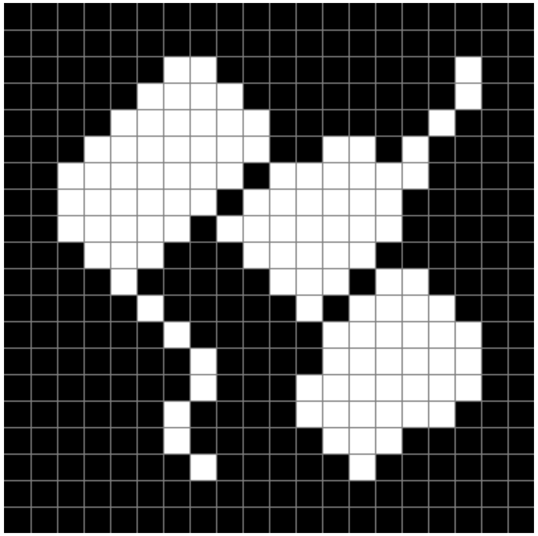


6-connected 18-connected 26-connected

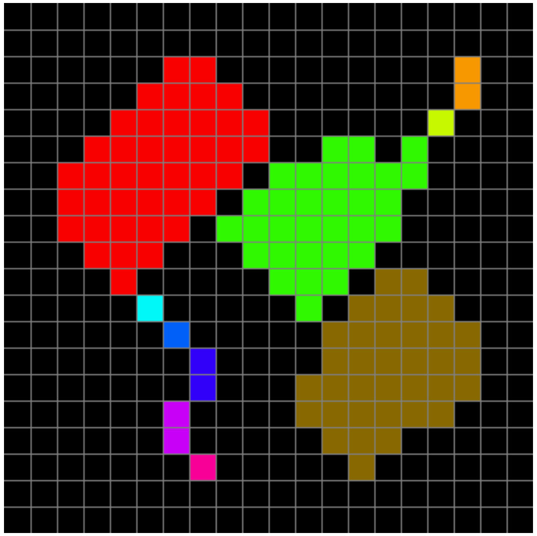
Connected Components

- Number of components depends on the chosen connectivity

Thresholded image

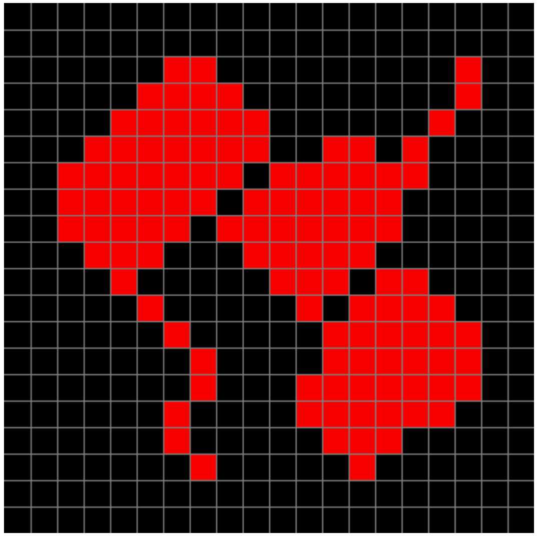


4-connected objects



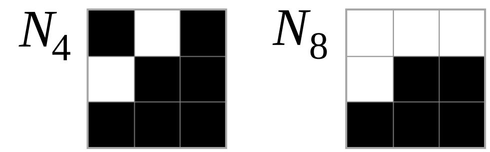
Number of objects = 10

8-connected objects



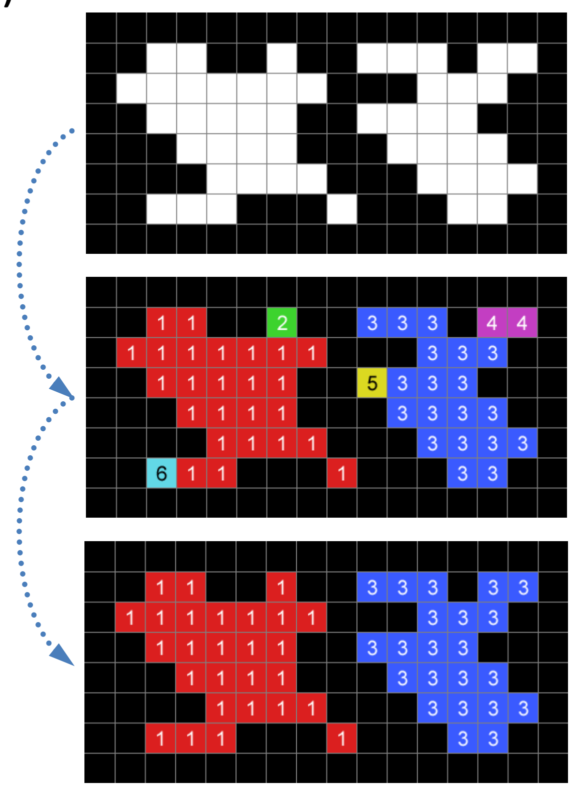
Number of objects = 1

Connected Components 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

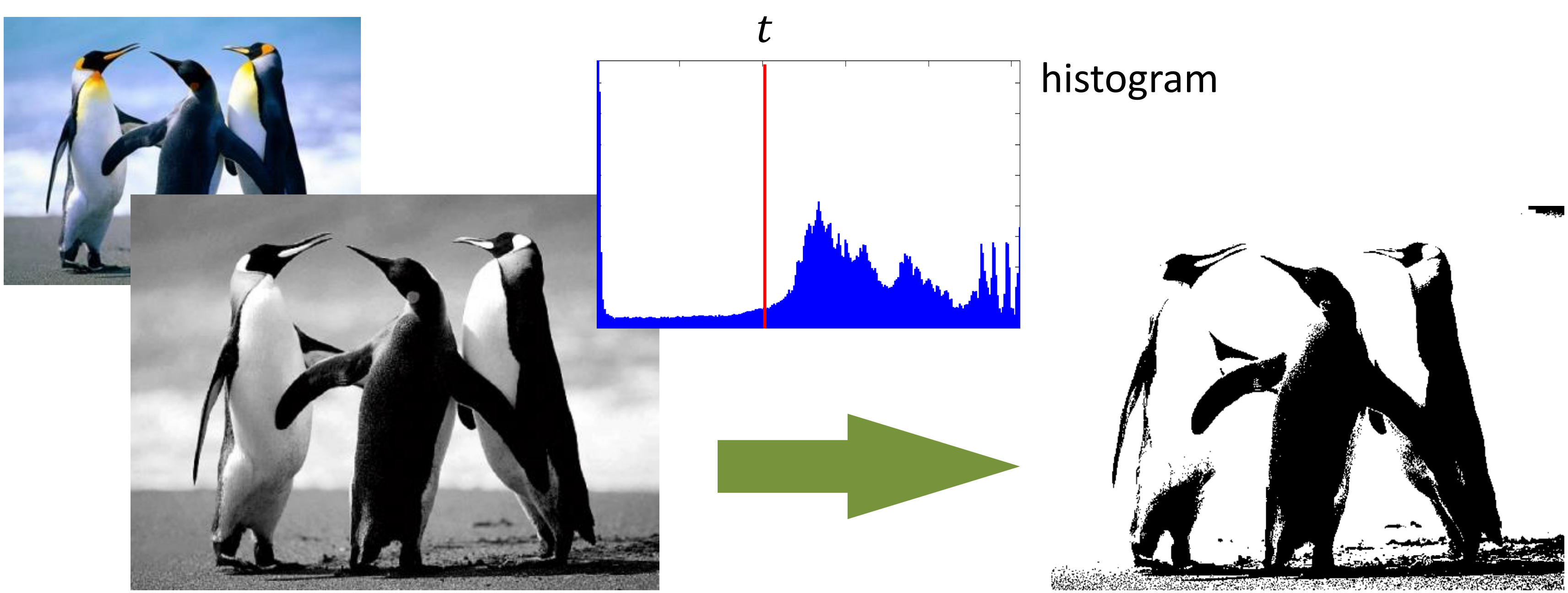
    Equivalence sets {1,2,6} {3,4,5}



- 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

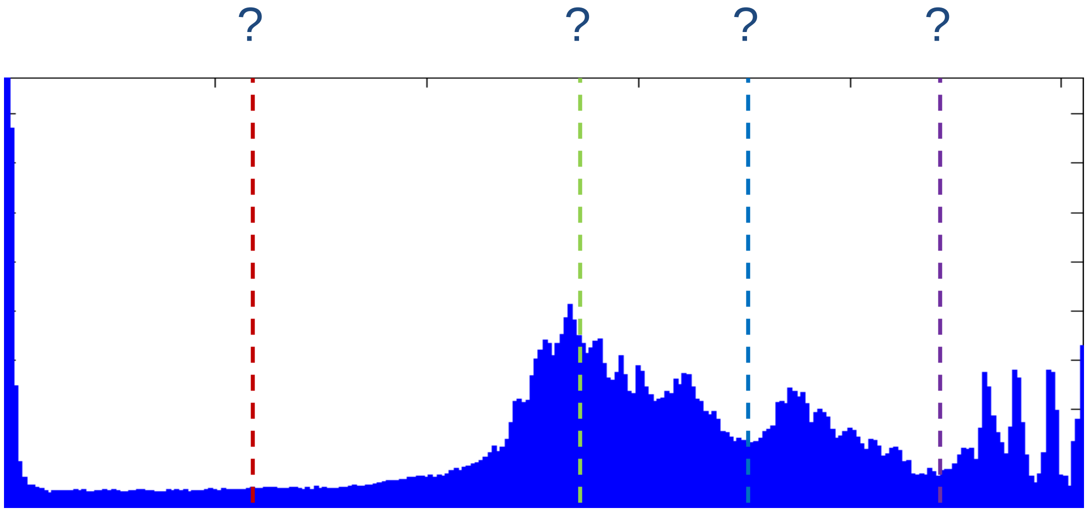
Region Splitting

- Basic computational approach
  - One of the oldest techniques in computer vision
  - First compute the histogram for the whole image
  - Then find a threshold t that best separates the peaks in the histogram



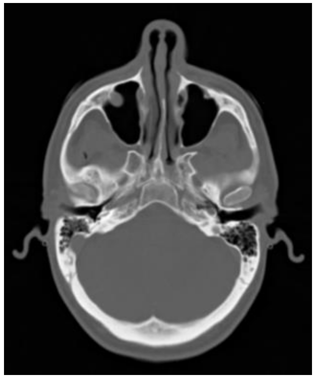
Region Splitting

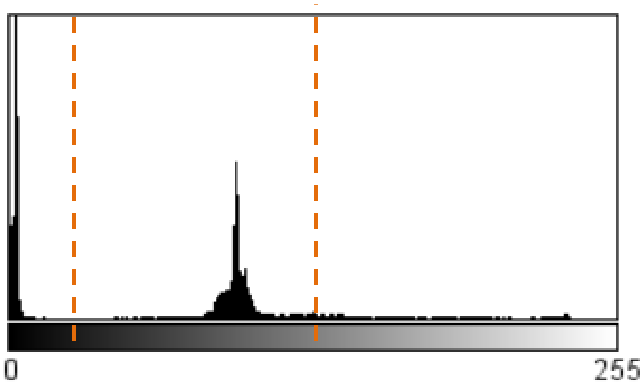
- Basic computational approach
  - Repeat until regions are either fairly uniform or below a certain size
  - Optimise metrics of **intra-region similarity** and **inter-region dissimilarity**



Histogram based Region Splitting

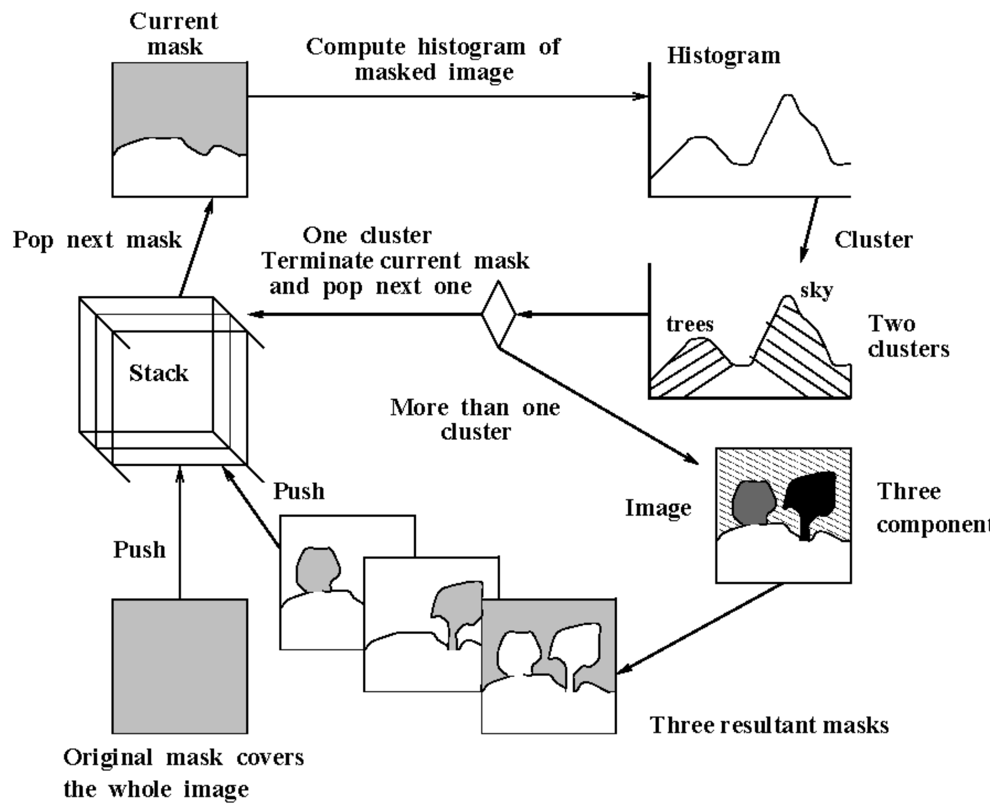
– Follow a measurement space clustering process
– Assume homogeneous objects in the image appear as clusters in measurement space (the histogram in our example)
– Histogram clustering can be accomplished by finding the valleys and declaring the intervals between them as the clusters
– Segmentation = mapping the clusters back to the image domain
– Connected components of the cluster labels constitute the segments





Histogram based Region Splitting

- Recursive version
  - Perform histogram mode seeking first on the whole image
  - Then on each of the regions obtained from the resultant clusters
  - Until the regions obtained cannot be decomposed further



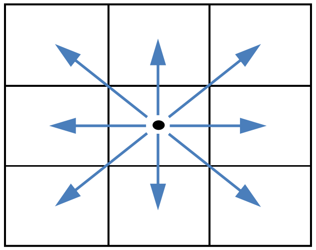
Heuristics based Region Merging

- merge regions based on their relative **boundary lengths** and the strength of the visible edges at the boundaries
- link clusters together based either on the distance between their **closest points**, their **farthest points** or some thing in between
- merge adjacent regions whose average colour difference is below a threshold or whose regions are too small
- a **useful pre-processing stage** to make higher-level algorithms faster and more robust

Graph Based Region Merging

- Use relative dissimilarities between regions R_i to merge
- Represent regions as a graph using minimum spanning tree (MST)
- Define a dissimilarity measure ω such as intensity difference
- Compute internal region dissimilarity using the graph edges e

$$I(R) = \max_{e \in \mathrm{MST}(R)} \omega(e)$$



- Compute dissimilarity between adjacent regions

$$D(R_i, R_j) = \min_{e \in (v_i \in R_i, v_j \in R_j)} \omega(e)$$

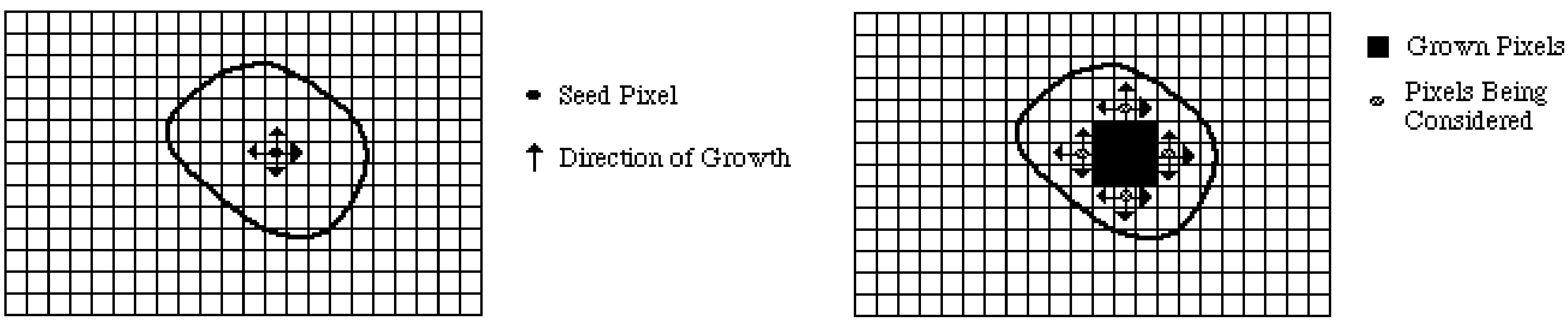
- Merge any two adjacent regions whose dissimilarity is smaller than the minimum of the internal differences of these two regions

Region Merging

Felzenszwalb & Huttenlocher. [Efficient graph-based image segmentation](). International Journal of Computer Vision 59(2): 167–181, 2004.

Region 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

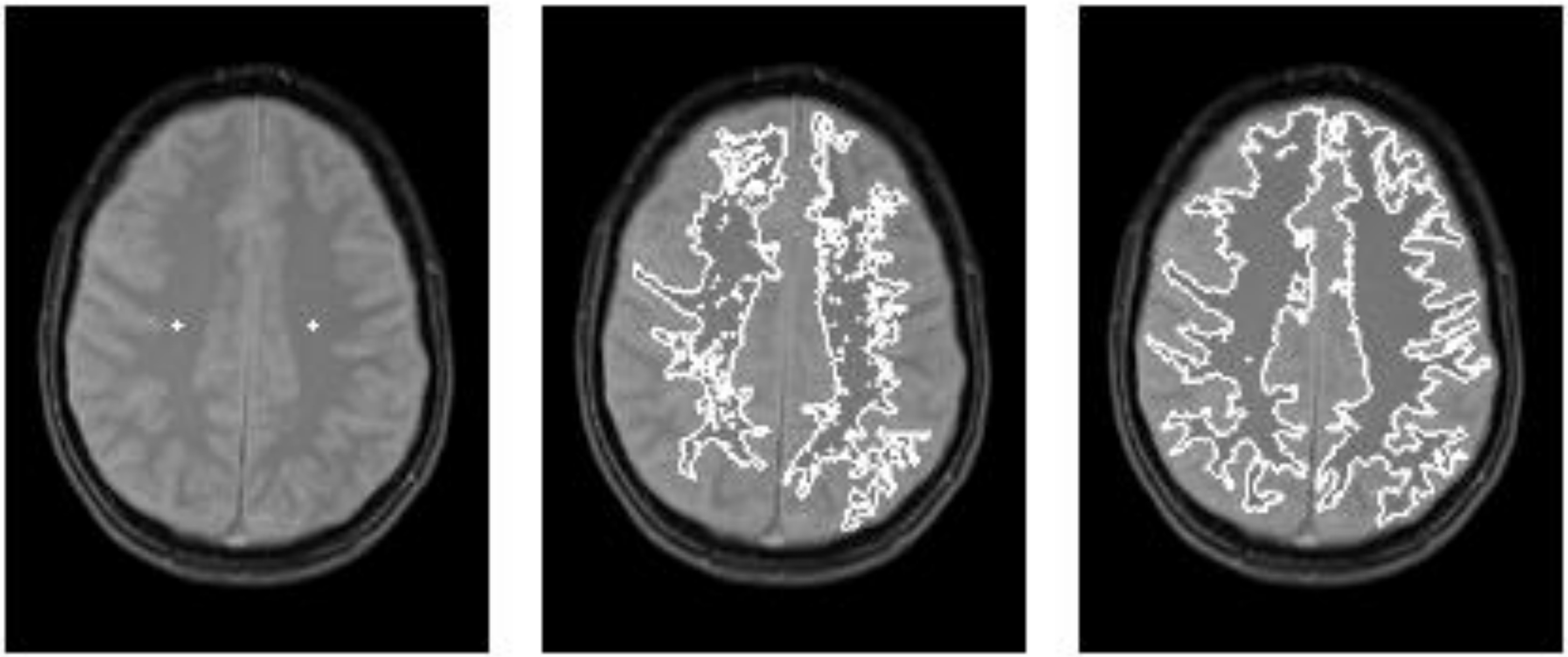


(a) Start of Growing a Region

(b) Growing Process After a Few Iterations

https://users.cs.cf.ac.uk/Dave.Marshall/Vision_lecture/node35.html

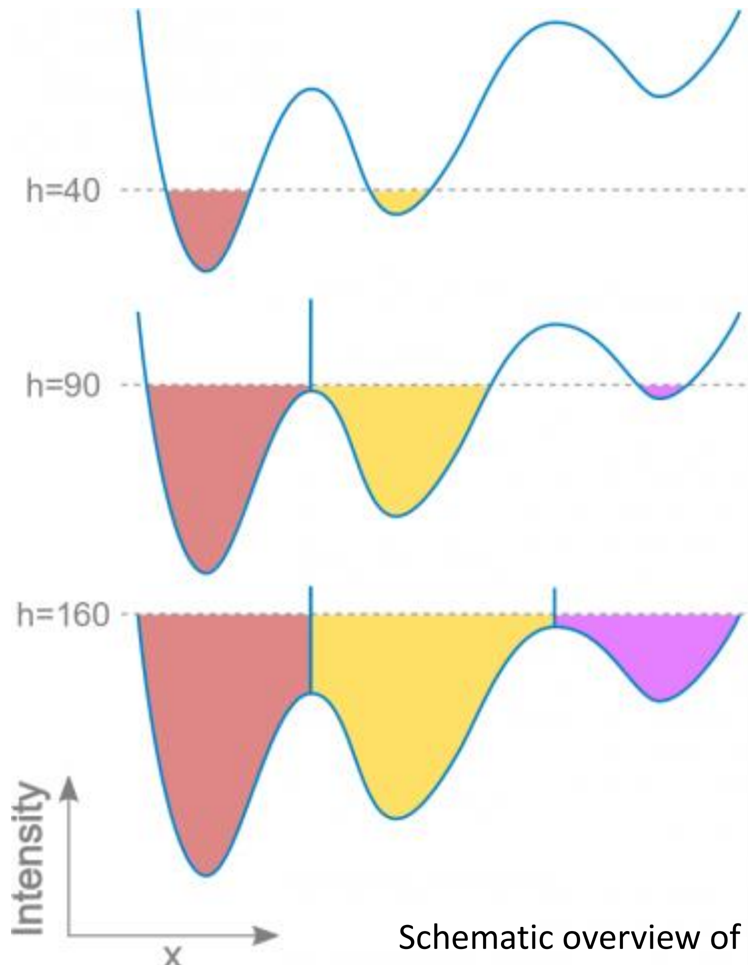
Region Growing



https://sbme-tutorials.github.io/2019/cv/notes/6_week6.html

Watershed

- Basics:
  - segment an image into several **catchment basins** - regions of an image where rain would flow into the same lake
  - start flooding the landscape at all of the **local minima** and to label ridges whenever different evolving components meet



Schematic overview of watershed flooding in 1D

Watershed

- Basics:
  - a technique related to **thresholding**
  - operate on a grayscale image
  - often used as part of an interactive system where the user first marks seed locations that correspond to the centres of different desired components

Watershed Segmentation

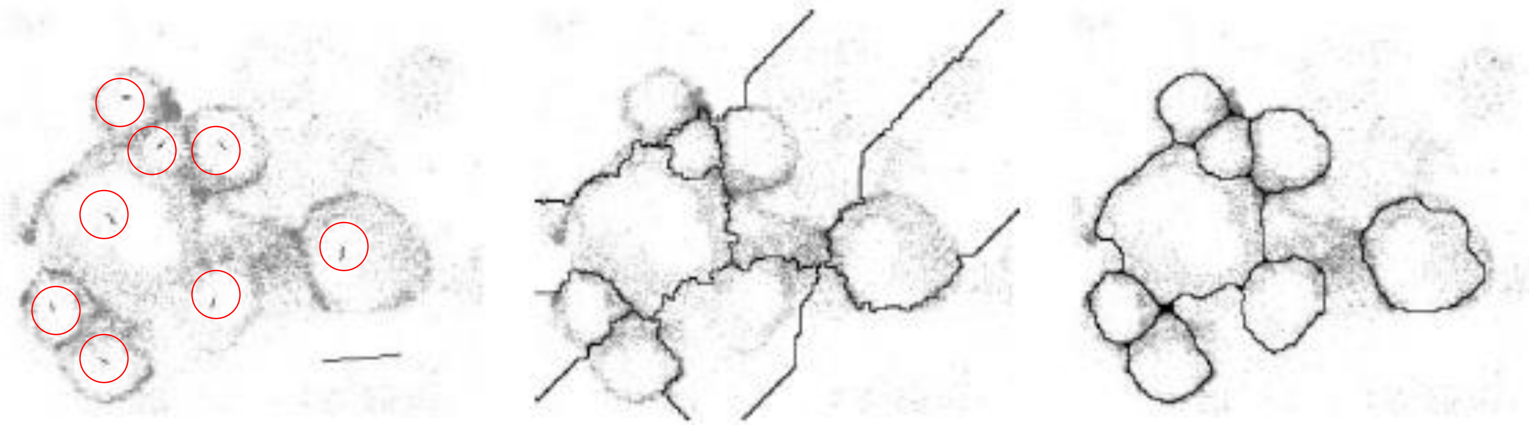
- Meyer's flooding algorithm

1) Choose a set of markers to start the flooding. These could be, for example, all local minima. Give each a different label.
2) Put neighbouring pixels of each marker into a priority queue with a priority level corresponding to the gray value of the pixel.
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

- Example

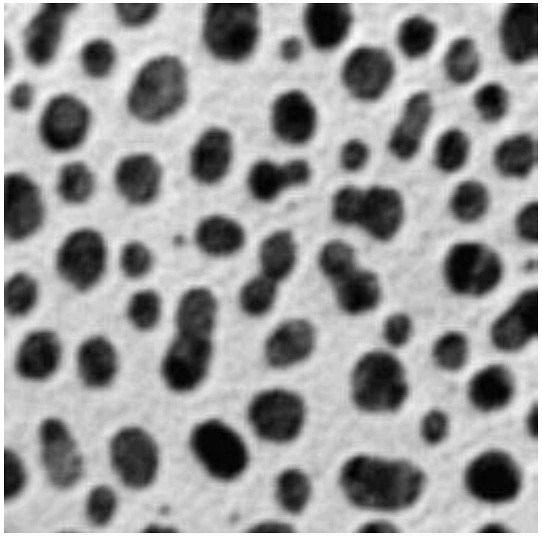


(a) (b) (c)

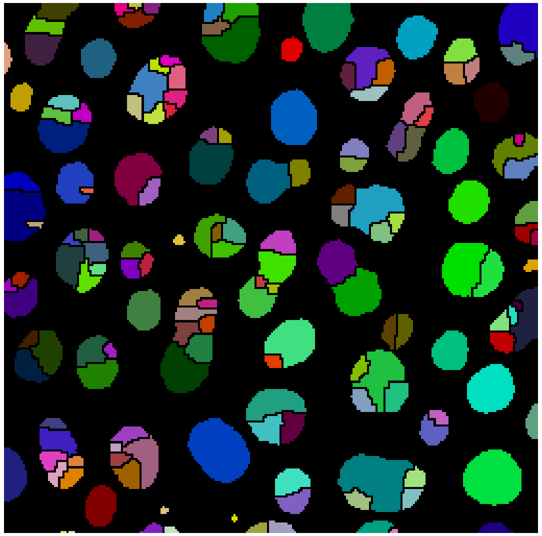
Watershed Segmentation

- Example segmentation results

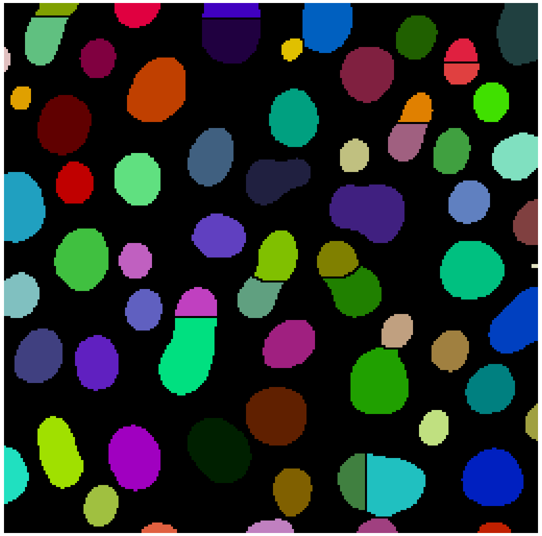
Input image



Segmentation results



Watershed of input

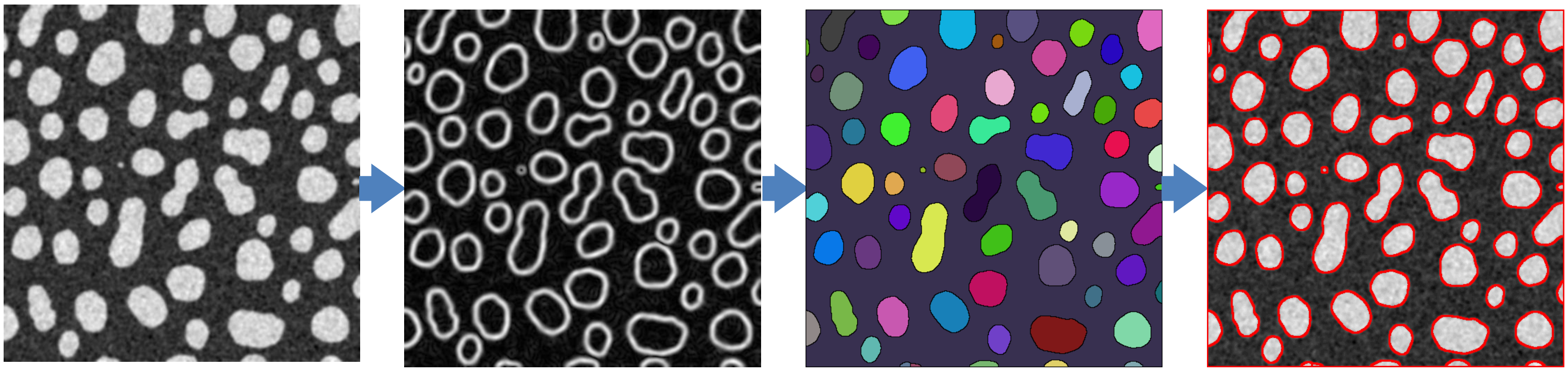


Watershed of smoothed input

https://imagej.net/Classic_Watershed

Watershed Segmentation

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



- Important notes regarding watershed based segmentation
  - 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

Maximally Stable Extremal Regions

- Basics
  - MSER regions are connected areas characterised by almost uniform intensity, surrounded by contrasting background
  - They are constructed through a process of trying multiple thresholds
  - The selected regions are those that maintain unchanged shapes over a large set of thresholds.

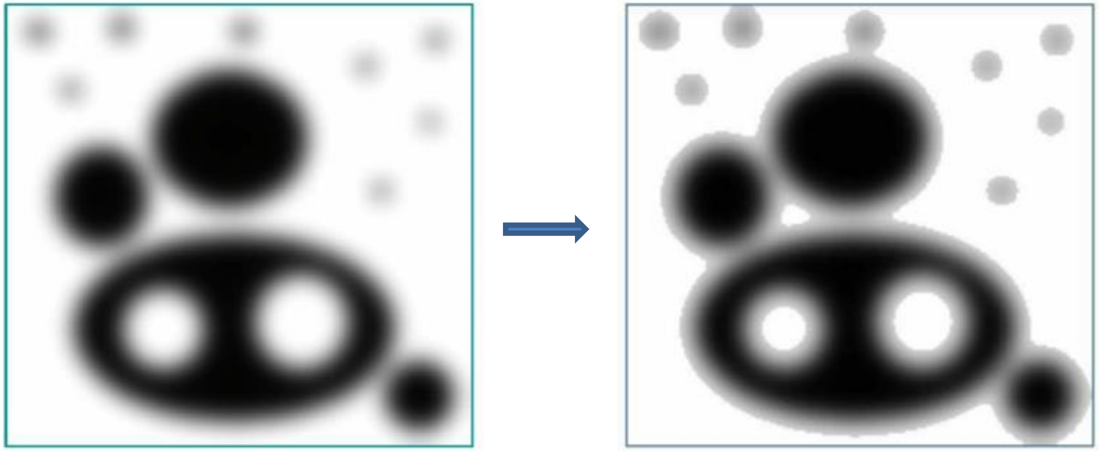


Image — Extremal regions

https://courses.cs.washington.edu/courses/cse455/10au/notes/MSER.pdf

Maximally Stable Extremal Regions

- Algorithm
  - For each threshold, compute the connected binary regions
  - Compute a function, area $A(i)$, at each threshold value i
  - Analyze this function for each potential region to determine those that persist with similar function value over multiple thresholds.

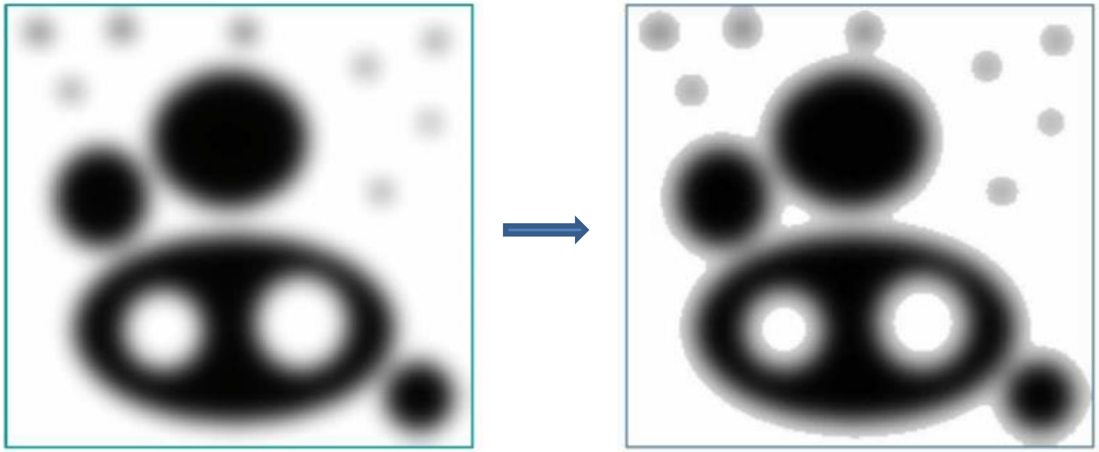


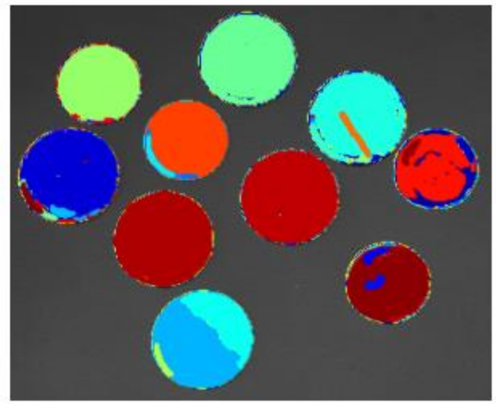
Image　　　　Extremal regions

https://courses.cs.washington.edu/courses/cse455/10au/notes/MSER.pdf

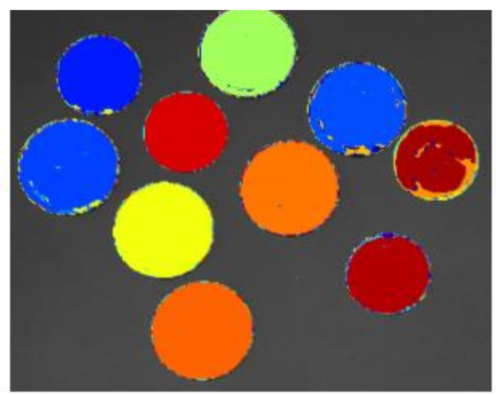
Maximally Stable Extremal Regions

- Example

Image



Extremal regions

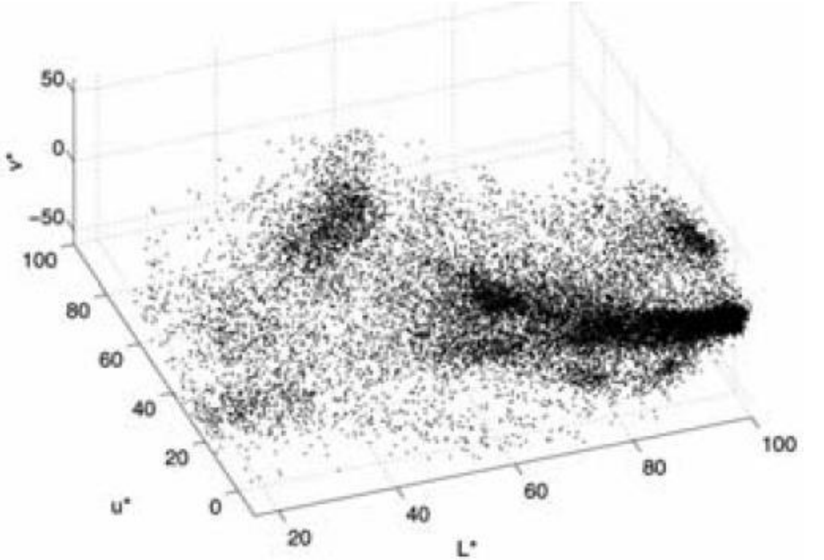


After filtering non-circular regions

https://au.mathworks.com/help/vision/ref/detectmserfeatures.html

Mean Shift and Model Finding

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



K-means Clustering

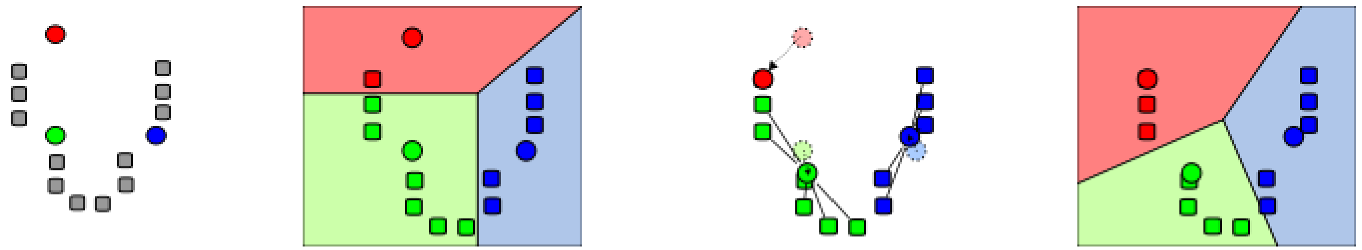
- Mean Shift is an alternative to K-means
- K-means
  - models the probability density as a superposition of spherically symmetric distributions
  - given the number of clusters k
  - randomly initialises sampling centres
  - iteratively updates the cluster centre location based on the samples that are closest to each centre
  - techniques exist for splitting or merging cluster centres to accelerating the process

K-means Clustering

- *Iterative K-means clustering*
  - set iteration count *ic* to 1
  - randomly choose a set of K means $m_1(1)$, $m_2(1),\ldots,m_K(1)$
  - dor each vector x_i compute distance $D(x_i, m_k(ic))$ for each $k = 1,\ldots,K$ and assign x_i to the cluster C_j with the nearest mean
  - increment ic by 1 and update the means to get a new set $m_1(ic)$, $m_2(ic),\ldots,m_K(ic)$
  - repeat step 3 and 4 until $C_k(ic) = C_k(ic +1)$ for all k

K-means Clustering

- *Iterative K-means clustering*



Mean Shift

- K-means clustering has limitations
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima

- Mean shift is a good alternative in many applications
  - Seeks stationary points (peaks/modes) in a density function
  - Attempts to find all possible cluster centres in feature space
  - Does not require knowing the number of clusters a priori
  - Is a variant of iterative steepest-ascent method

Mean Shift

- Iterative mode searching

1. Initialize a random seed point x and window N
2. Calculate the mean (center of gravity) $m(x)$ of all samples within N
3. Shift the search window to the mean
4. Repeat Step 2 until convergence

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

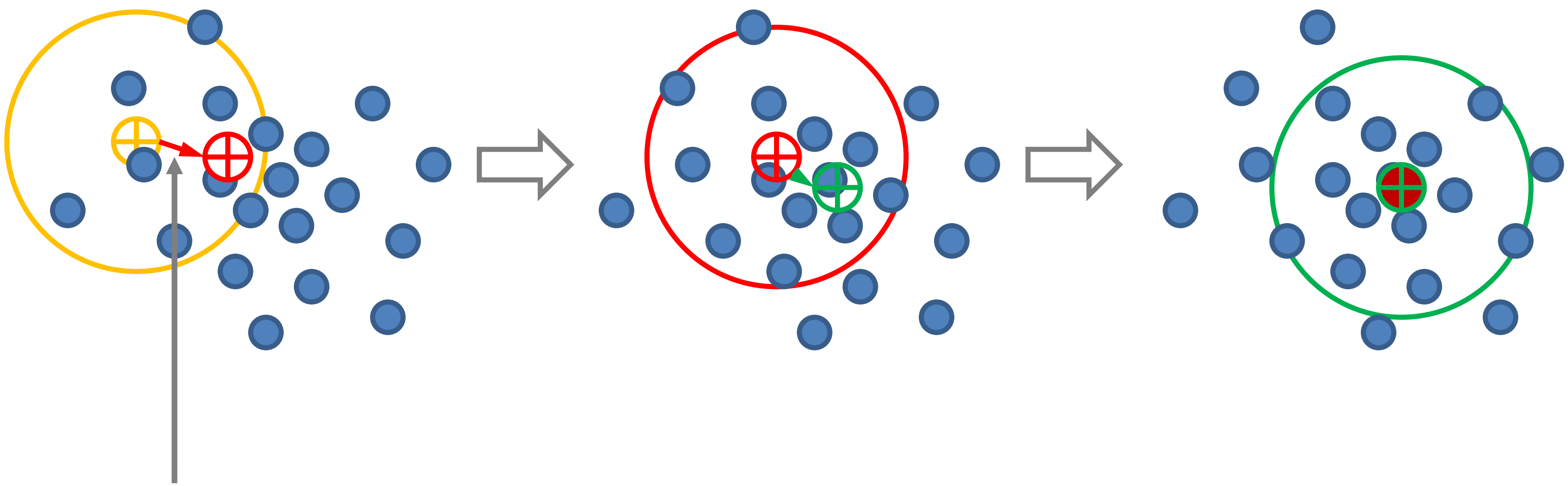
Mean (centre of gravity)

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

Kernel (weight function)

Mean Shift

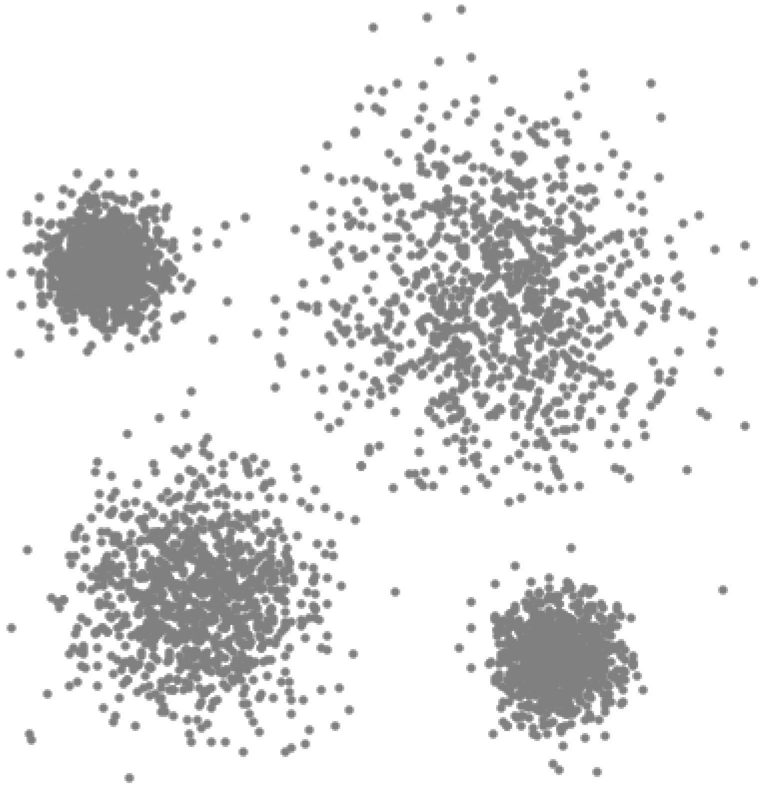
- Iterative re-estimation of the weighted mean



Mean shift vector $m(x) - x$

Mean Shift

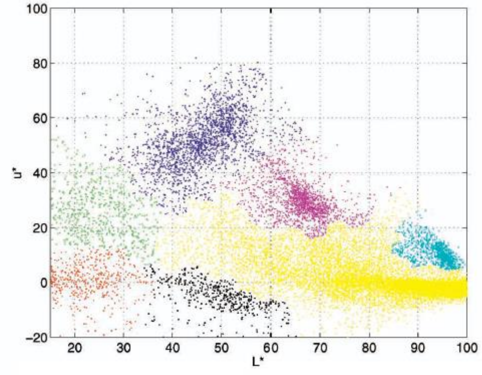
- Use a set of seed points to find all possible cluster centers

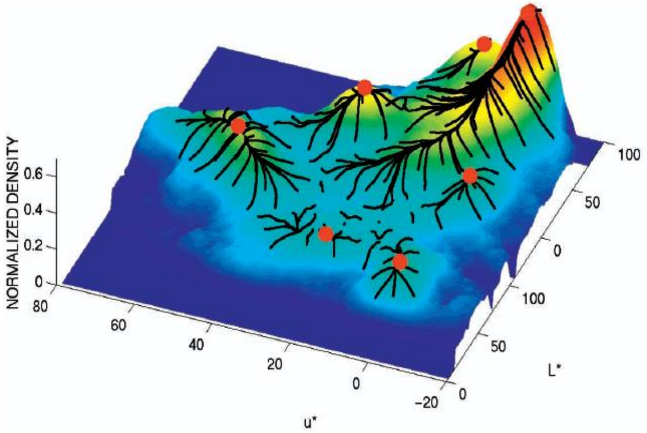


https://sbme-tutorials.github.io/2019/cv/notes/6_week6.html

Mean Shift for Image Segmentation

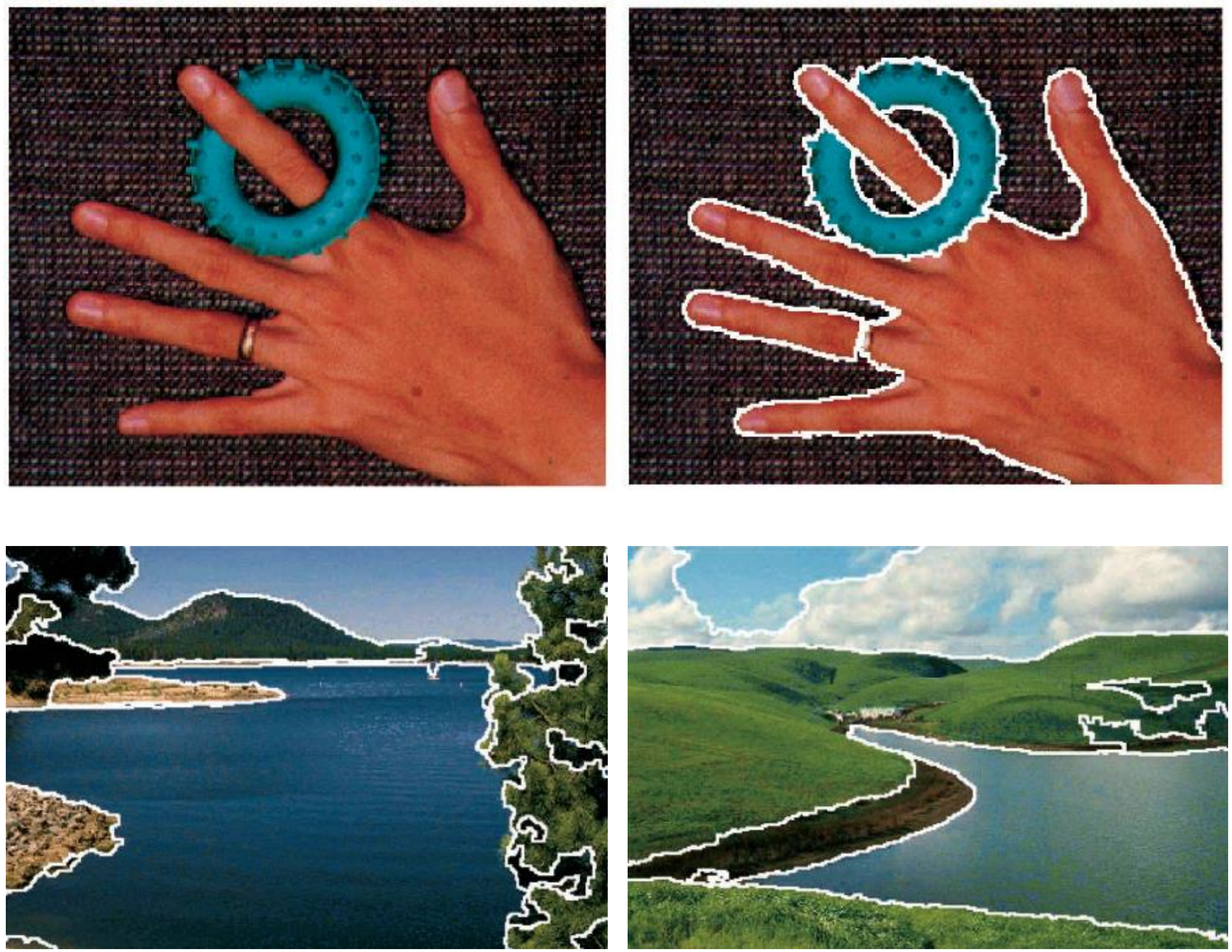
- Define features (colour, gradients, texture, et cetera)
- Transform image pixels to points in the feature space
- Initialise windows at multiple seed locations in feature space
- Perform mean shift for each window until convergence
- Merge windows that end up near the same location
- Cluster all points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





https://doi.org/10.1109/34.1000236

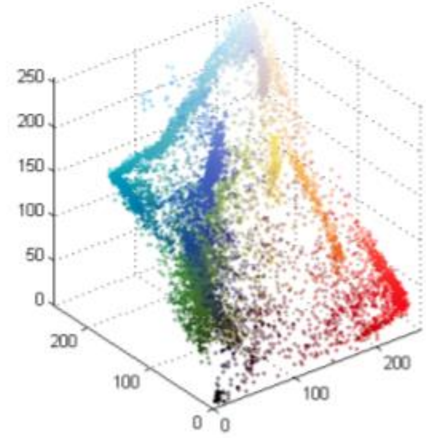
Mean Shift



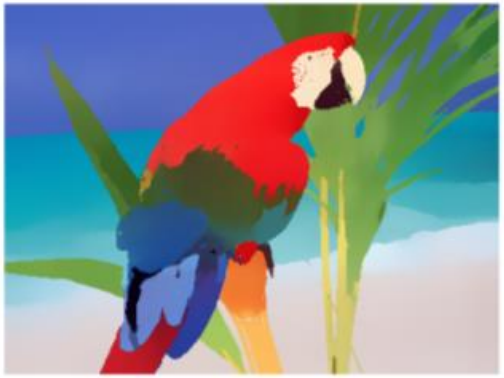
Mean Shift

input image

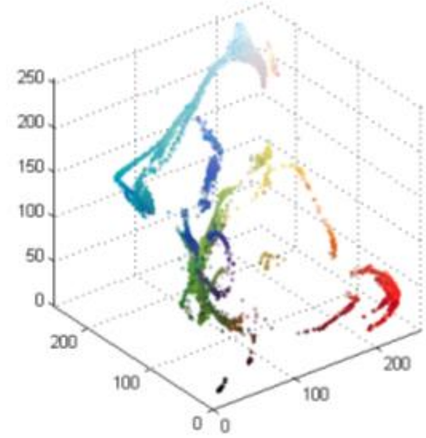
Pixel Distribution Before Meanshift



output image

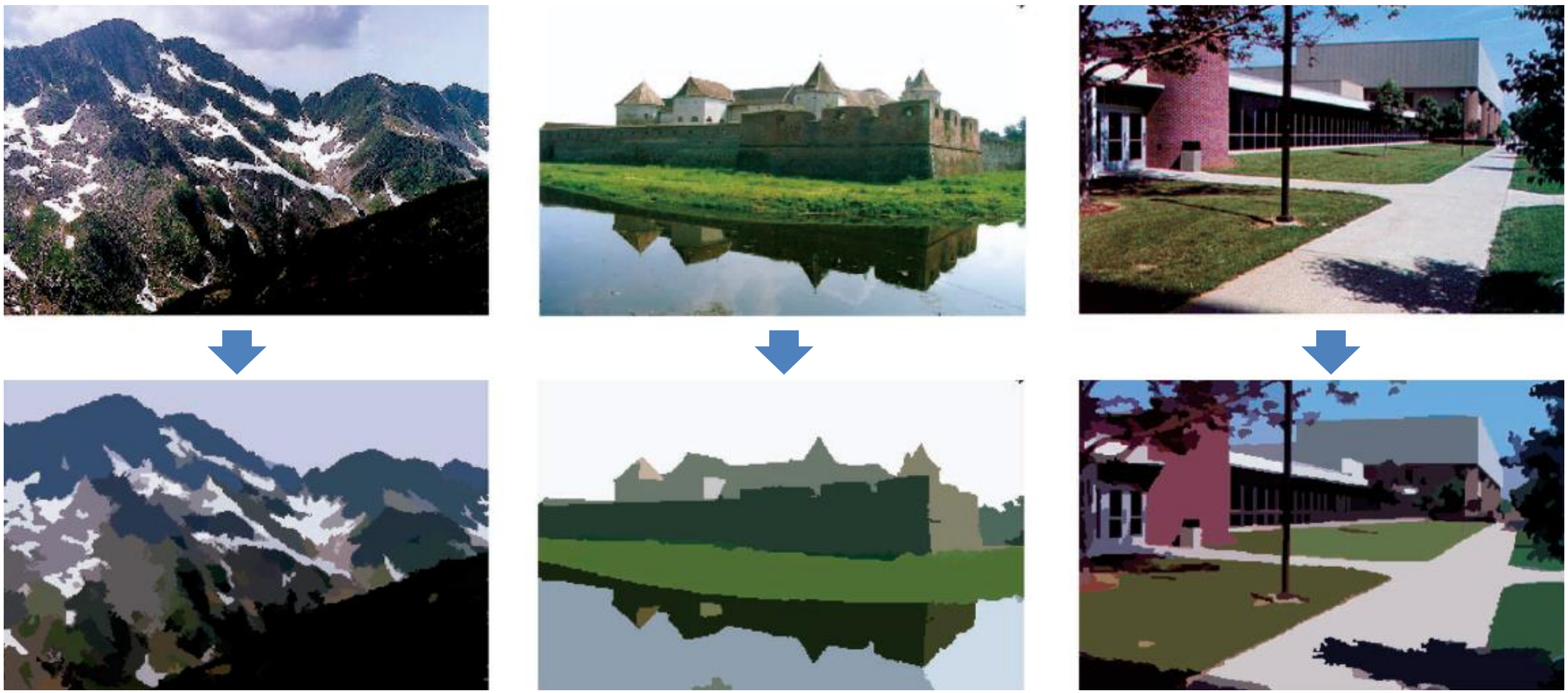


Pixel Distribution After Meanshift



Mean Shift

- Replace segmented region colours by their cluster means



https://doi.org/10.1109/34.1000236

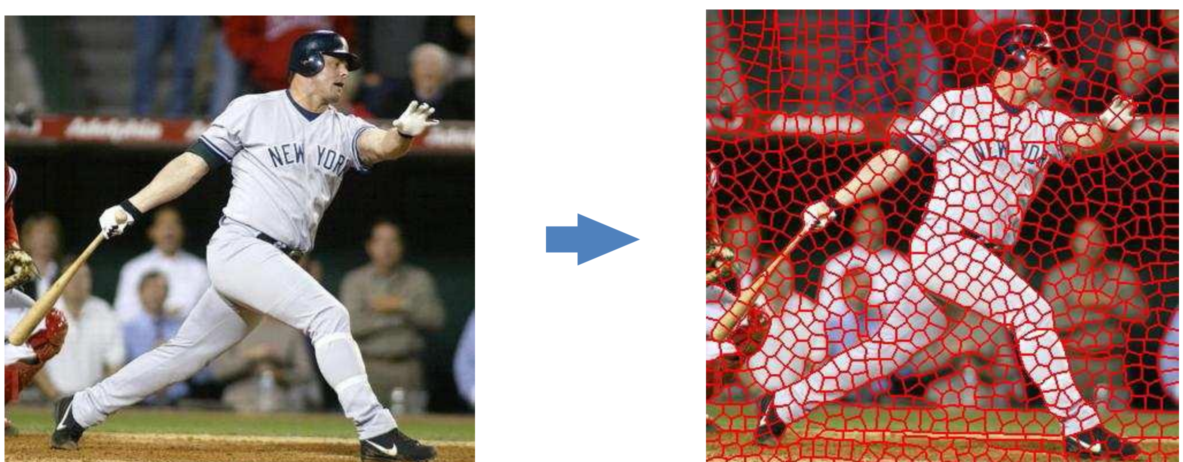
Mean Shift

- 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

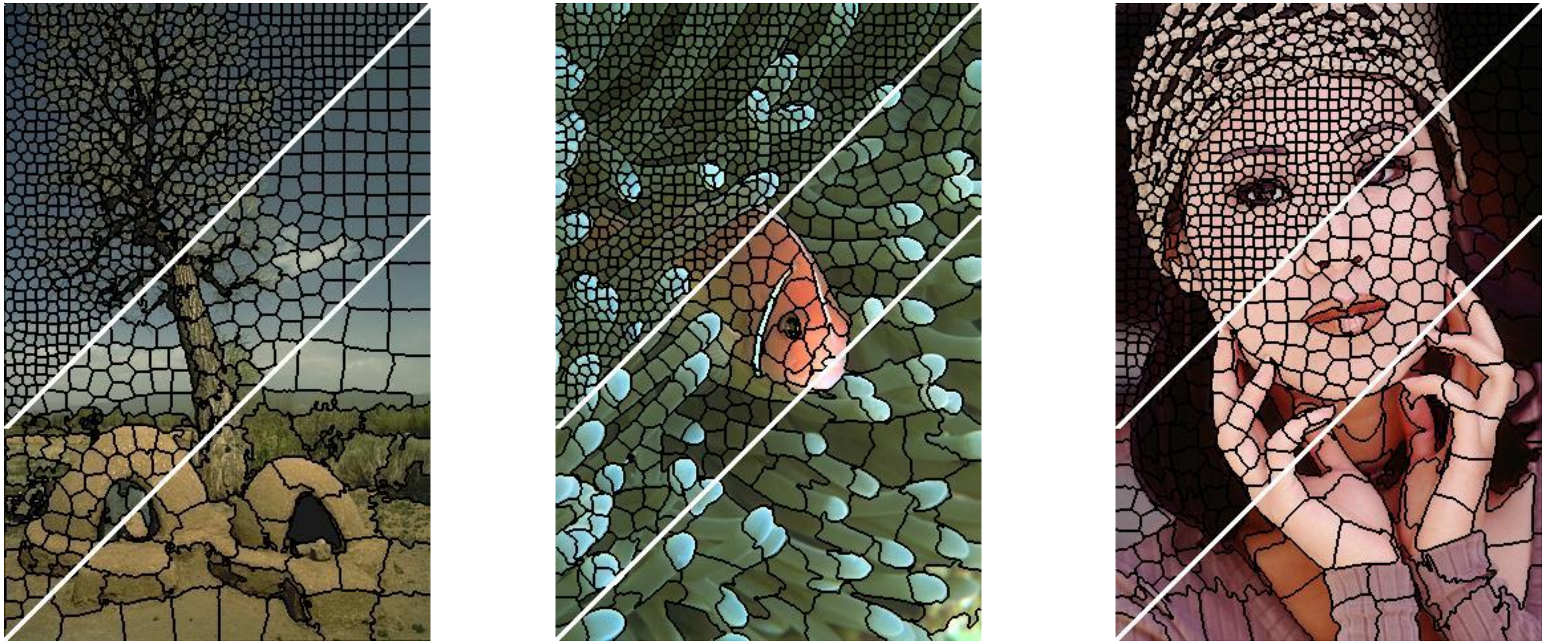
Superpixel Segmentation

- Pixel-wise sliding window segmentation
  - Too many windows (a.k.a. image patches)
- Superpixel-based segmentation improves efficiency
  - Group similar pixels into a superpixel
  - Superpixels together are an over-segmentation of the image
  - Ultimate segmentation (classification) performed on superpixels



Superpixel Segmentation

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



Superpixels of size 64, 256, and 1024 pixels (approximately)
https://ivrl.epfl.ch/research-2/research-current/research-superpixels/

Superpixel Segmentation

- 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

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

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

$$D = \sqrt{\frac{d_{lab}^2}{m^2} + \frac{d_{xy}^2}{S^2}}$$ (weight m controls influence of color over spatial distance)

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

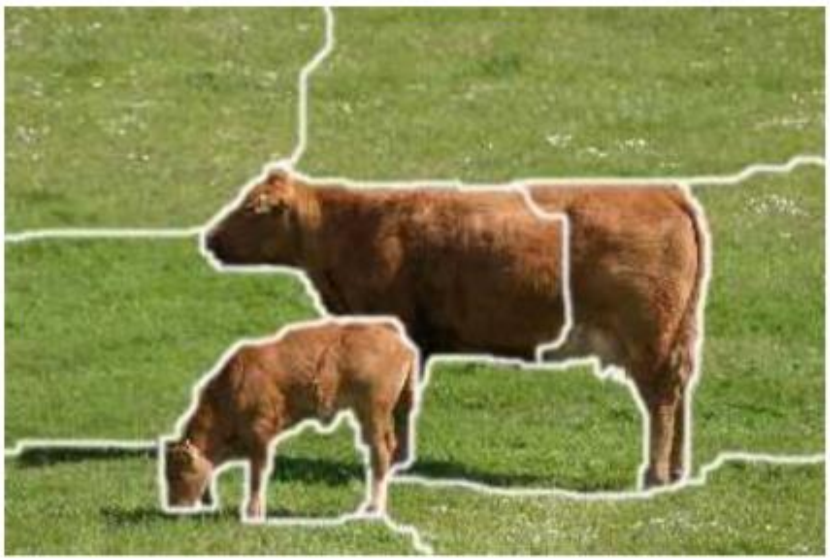
Conditional Random Field

- 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

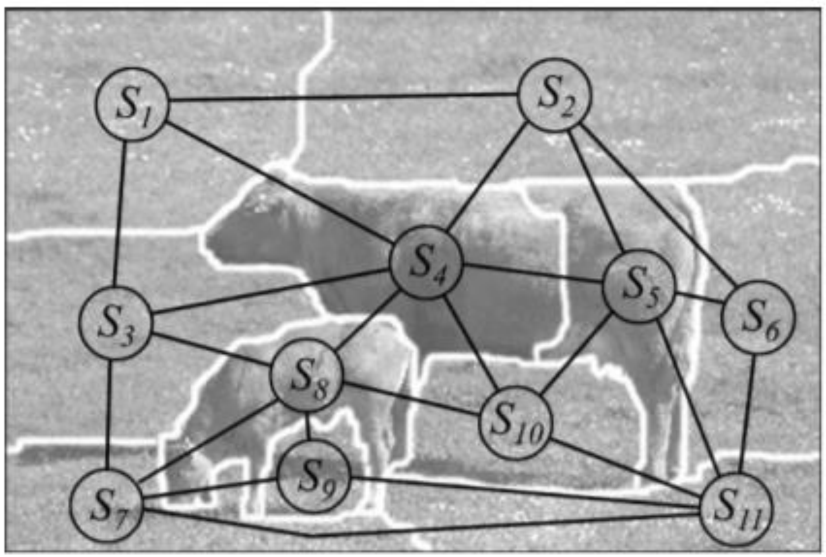
  A probabilistic graphical model that encodes the relationships between observations (i.e. superpixels) and constructs a consistent interpretation (i.e. segmentation) for a set of data (i.e. an image)

Conditional Random Field

- An undirected graphical structure
  - 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

Conditional Random Field

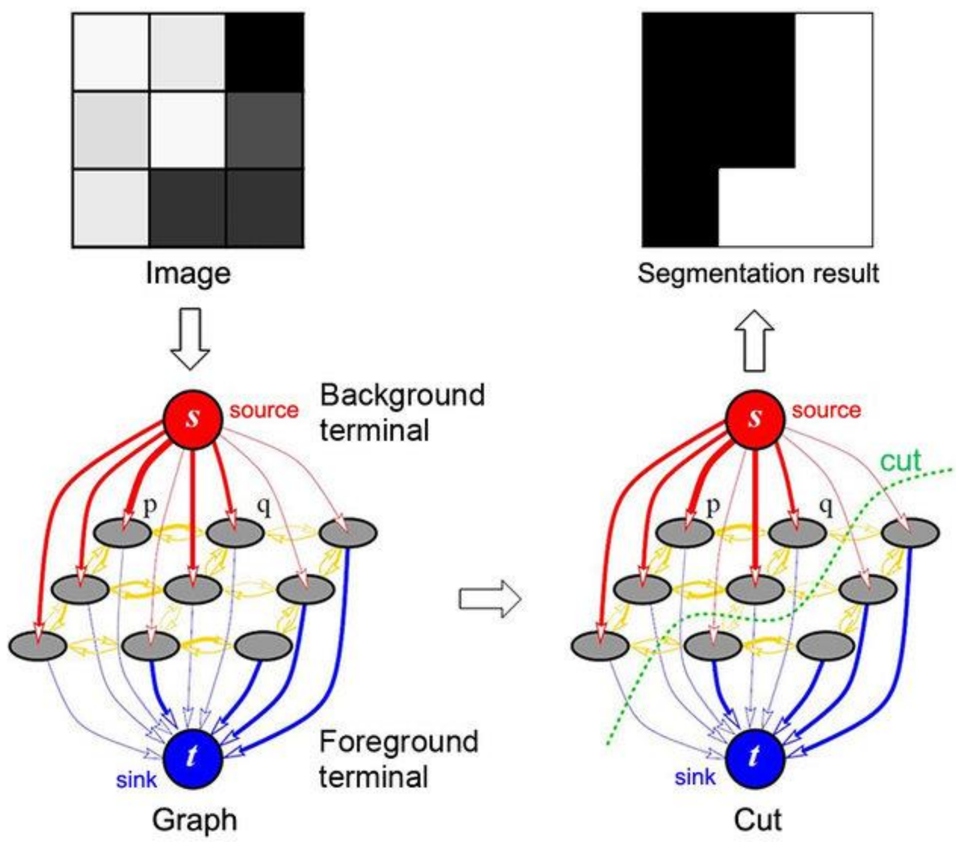
- Segmentation as an energy minimisation problem

$$E(s,c) = \sum_i \varphi(s_i, c_i) + \sum_{ij} \psi(s_i, s_j)$$

  - 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)

Conditional Random Field

- Energy minimization is solved via *graph cut* (max-flow min-cut)



https://doi.org/10.1109/TPAMI.2004.60
https://doi.org/10.1109/TGRS.2016.2627638

Conditional Random Field

- Segmentation example with multiple source-sink terminals

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

References and Acknowledgements

- Chapter 3 & 5 of Szeliski 2010
- Shapiro and Stockman 2001

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