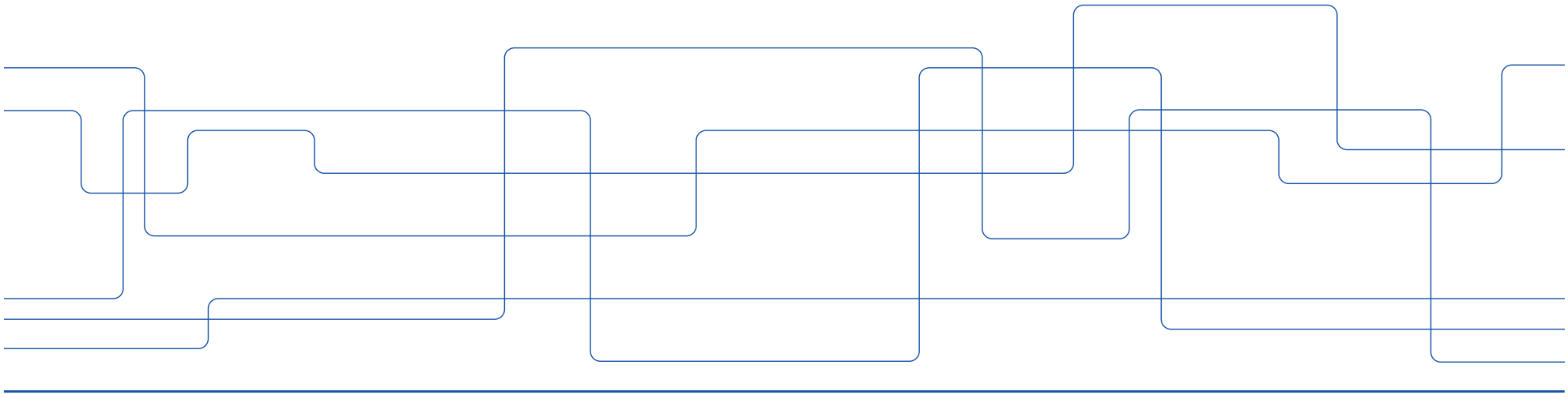
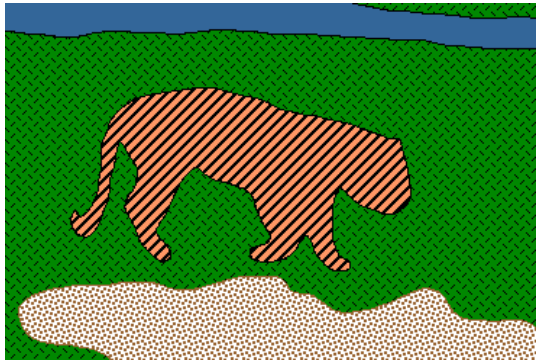
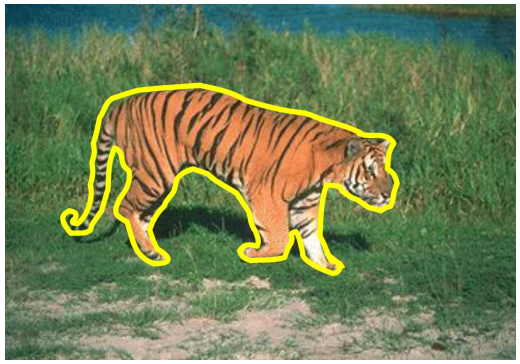

# Image segmentation

Mårten Björkman

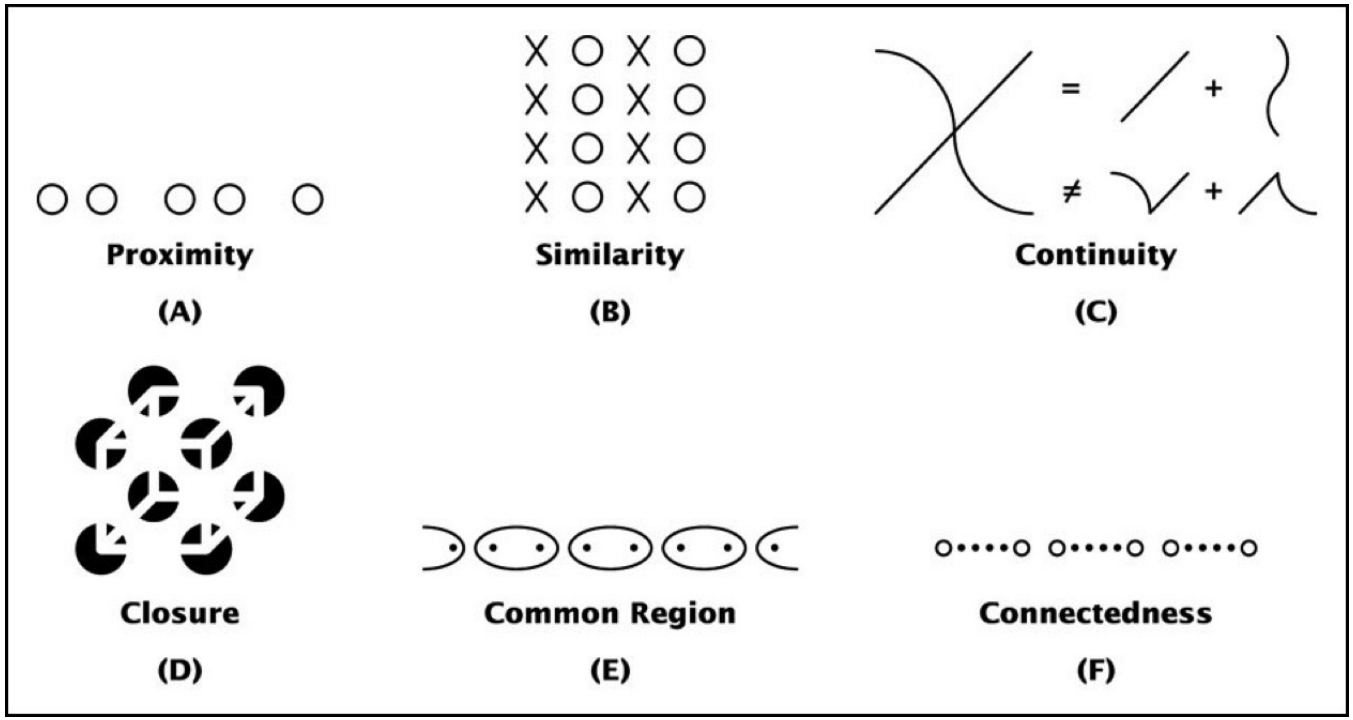


# From images to objects

- Image segmentation: Dividing images into semantically meaningful regions.
- Object segmentation: Cut out a particular object, but ignore the background.
- Reliable segmentation is possible with prior information, but such information is often not available.

# Perceptual grouping (Gestalt Theory)



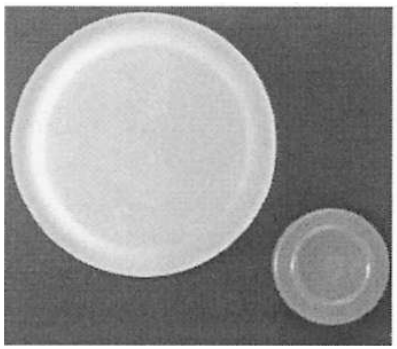

Researchers have been interested in segmentation (perceptual grouping) since at least 1890 and there are many theories.

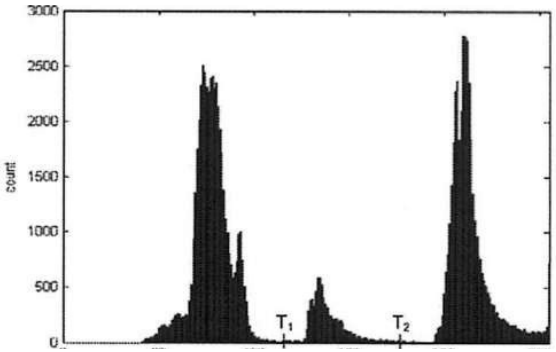
# Perceptual grouping (Gestalt Theory)

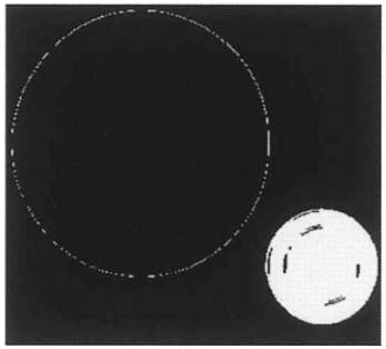
We perceive the whole scene, not just the individual parts. Sometimes it is impossible to say anything about the scene, by only looking at the parts.

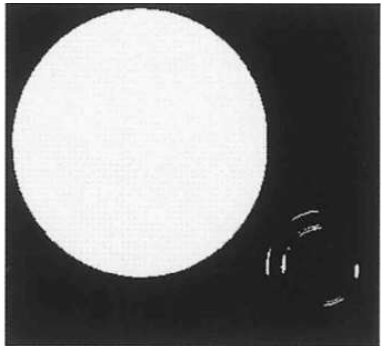
# Histogram based segmentation

- Threshold the grey-level values to create a binary image.
- How to automatically find a good threshold?
- Common problems: measurement noise, non-uniform illumination, non-constant intensity of objects, unknown size of objects, ...
- But… if it works, why complicate it.



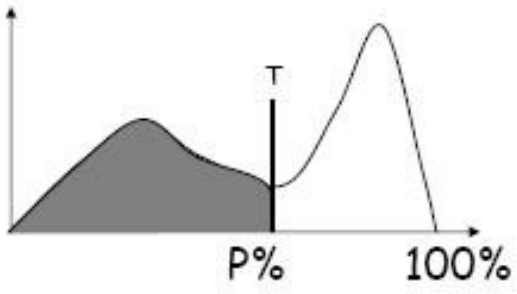


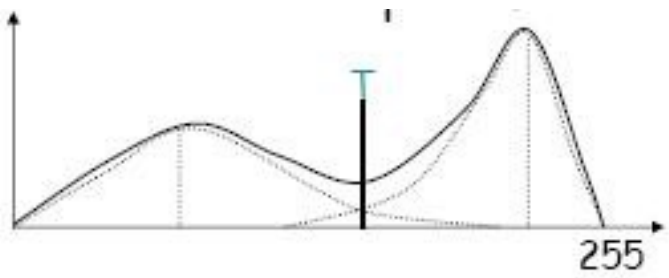





# Thresholding – two very simple methods

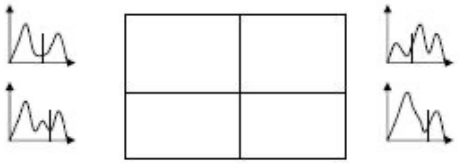
- P-tile method
  - Assume the object is either lighter or darker than the background and has a size that covers P% of the whole image.
  - Choose a threshold such that P% of the pixels are above or below the threshold.
- Mode method
  - Find the peaks and valleys of the histogram.
  - Set the threshold at the point of the deepest value with respect to the nearby peaks.

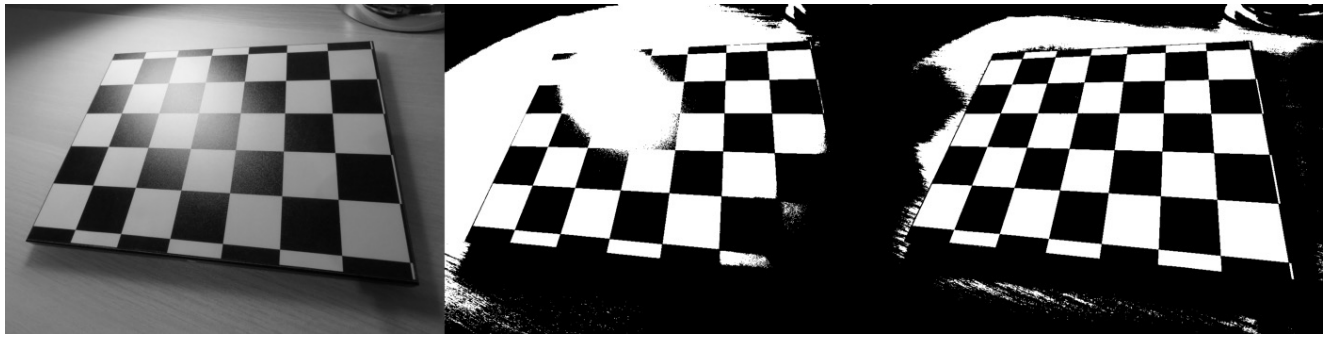


# Adaptive thresholding

- In case of uneven illumination, a global threshold does not work well.
- One approach is to divide an image into $m \times m$ subimages and determine a threshold for each subimage.
- Extension: fit local thresholds to a smooth function over the whole image.





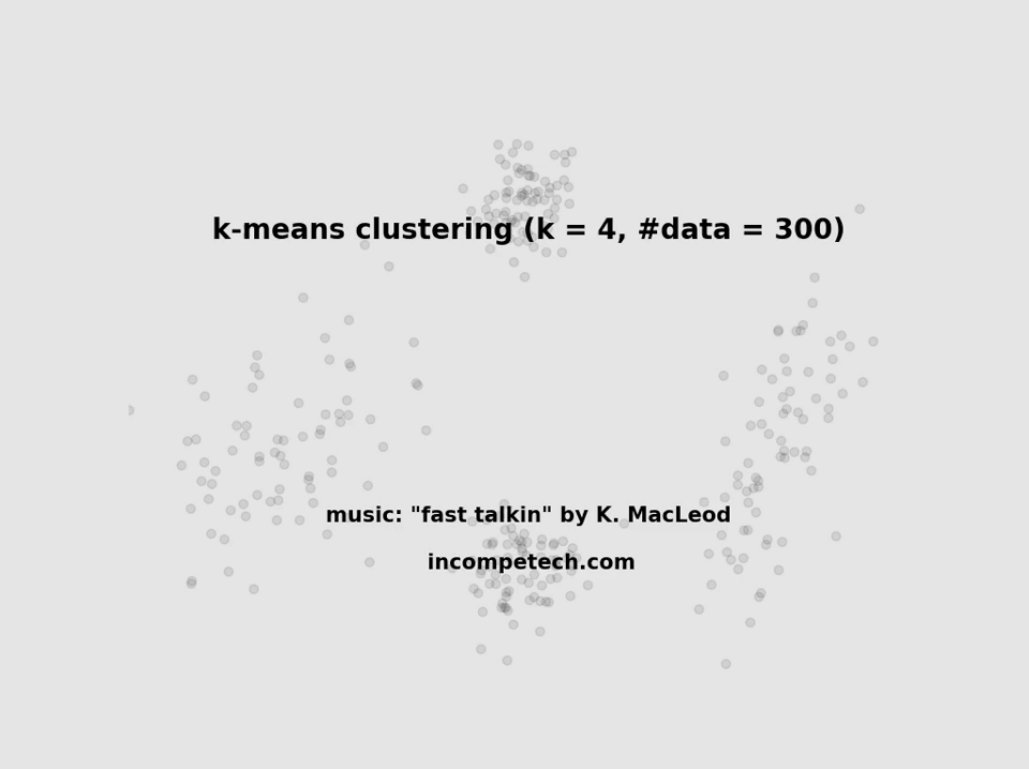
# Over-segmentation

- One way to simplify the problem of image segmentation is to first group similar pixels together into so called superpixels.
- These superpixels can then be merged into larger regions.

# K-means clustering

- Group pixels based on similarity in colours (or any other measure).
- K-means algorithm:
  1. Choose K initial mean values (possibly randomly).
  2. Assign each pixel to the mean that is closest.
  3. Update means as the average of pixels assigned to each mean.
  4. Iterate until there is no change in mean values.
- Problem: segments can be split up into pieces.

# K-means clustering



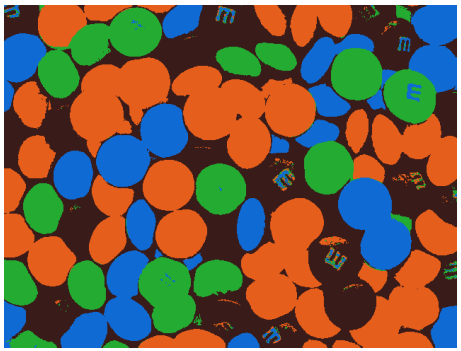

- Two iterative steps: 1) associate points to closest cluster centres, 2) recompute cluster centres as mean of points associated to them.

# K-means clustering

- How to initialize the clusters?
  1. Randomly pick K colours.
     > *Problem: these might not be representative to the pixel colours in the image*
  2. Randomly pick K pixels and use their colours.
     > *Problem: the same colour might be picked twice*
  3. Randomly pick a pixel and use its colour, then a pixel that is most different from the first and next a pixel that is most different from the first two pixels… and so on.

Original

4 clusters



8 clusters

# Gaussian mixture models (GMM)

- Gaussian mixture models can be viewed as an extension to K-means.
- Assume that a pixel has a combination of colours from multiple clusters, instead of just coming from one particular cluster.
- Then the colour distribution of pixel $i$ might be written as

$$P(c_i) = \sum_k P(z_i = k) P(c_i | z_i = k) = \sum_k \pi_k \, \mathcal{N}(c_i; \mu_k, \Sigma_k),$$

  where $\pi_k = P(z_i = k)$ is the probability that pixel $i$ belongs to cluster $k$ (assumed to be the same for all pixels) and $\mathcal{N}(c_i; \mu_k, \Sigma_k)$ is a Gaussian distribution with mean $\mu_k$ and variance $\Sigma_k$.
- Goal: Find the maximum likelihood estimate of model parameters.

# Gaussian mixture models (GMM)

Model parameters can be found with Expectation-Maximization.

1. Expectation step (update membership probabilities):

$$T_{i,k} \leftarrow P(z_i = k | c_i, \{\pi_k, \mu_k, \Sigma_k\}) = \frac{\pi_k \, \mathcal{N}(c_i; \mu_k, \Sigma_k)}{\sum_j \pi_j \, \mathcal{N}(c_i; \mu_j, \Sigma_j)}$$

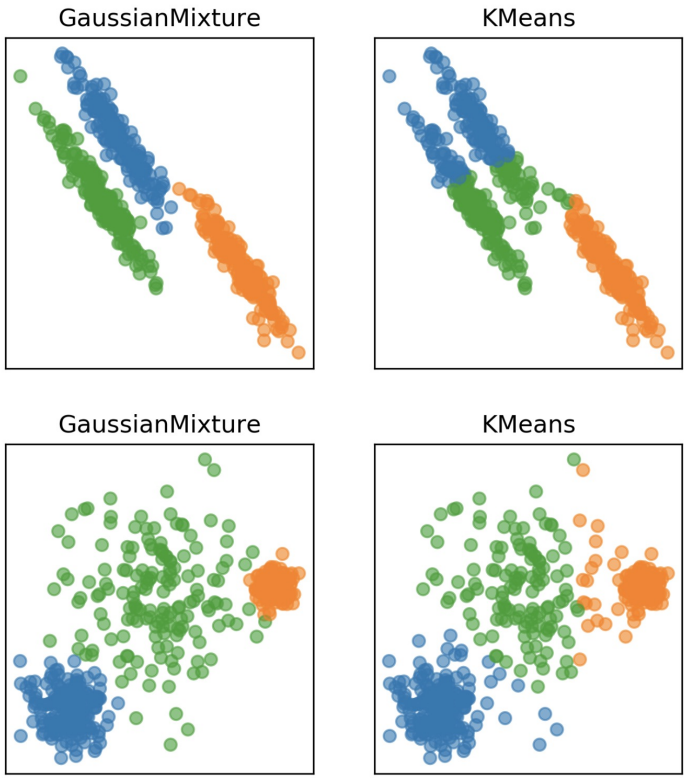
2. Maximization step (update model parameters):

$$\pi_k \leftarrow \frac{1}{N} \sum_i^N T_{i,k}, \quad \mu_k \leftarrow \frac{\sum_i^N T_{i,k} c_i}{\sum_i^N T_{i,k}}$$

$$\Sigma_k \leftarrow \frac{\sum_i^N T_{i,k} (c_i - \mu_k)(c_i - \mu_k)^T}{\sum_i^N T_{i,k}}$$

3. Iterate until convergence.

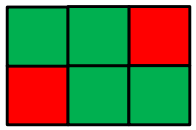
# Gaussian mixture models (GMM)



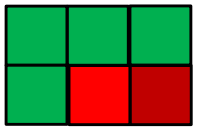

K-means assumes ’round’ clusters, GMM allows them to be elliptic.

# Spatial coherence

- Methods mentioned so far neglect dependency between neighbouring pixels.
- This may cause segments to be split up into different pieces.
- If spatial coherence between pixels is taken into account, this can be avoided.
- The dependency between neighbouring pixels or regions can be modelled in many different ways.



Group the two red pixels? Maybe not.



Group the two red pixels? Probably.

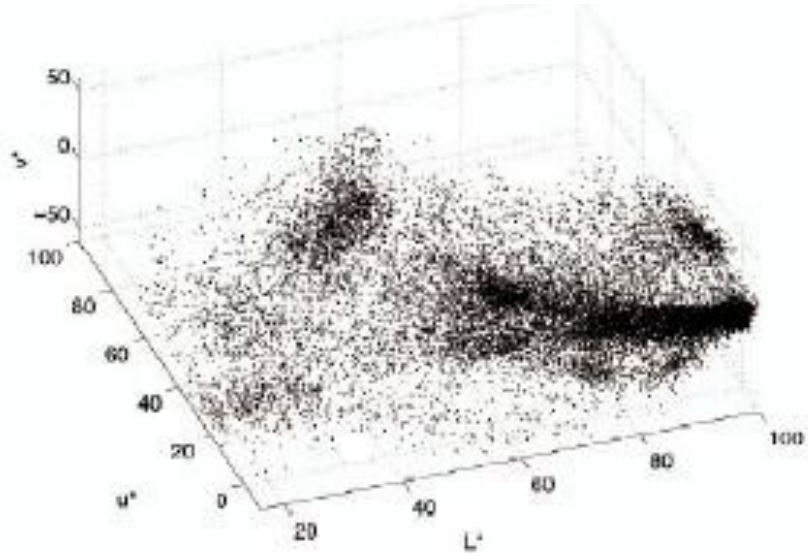
# Mean-shift segmentation

- Mean-shift is a common method for kernel density estimation.
- When used for segmentation, it often uses not just in colours, but also positions.

$$\Longrightarrow_{each\ pixel} \begin{bmatrix} x \\ y \\ R \\ G \\ B \end{bmatrix}$$

- Note: You could use $(x, y)$ also for K-means and GMM. However, for Mean-shift it is more common to do so.

# Mean-shift segmentation

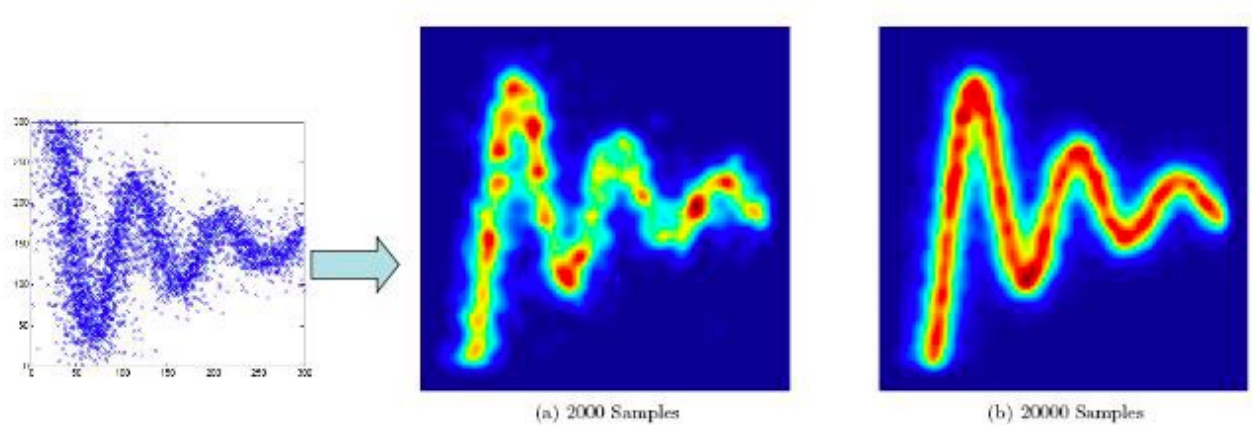



Example distribution of colour values in $Luv$ space

($L$ =luminance, $uv$ =colour)

# Mean-shift segmentation

- Mean-shift tries to find points (in 5D space) with maximum density.
- Problem: We just have a bunch of discrete samples!
- However, each sample is noisy and assumed to come from some distribution.
- Solution: We place a small “ball” $K(x - x_i)$ around each sample, with maximum density in the centre $x_i$.



(a) 2000 Samples (b) 20000 Samples

# Formally: an iterative scheme

We want to find maxima of the total density function

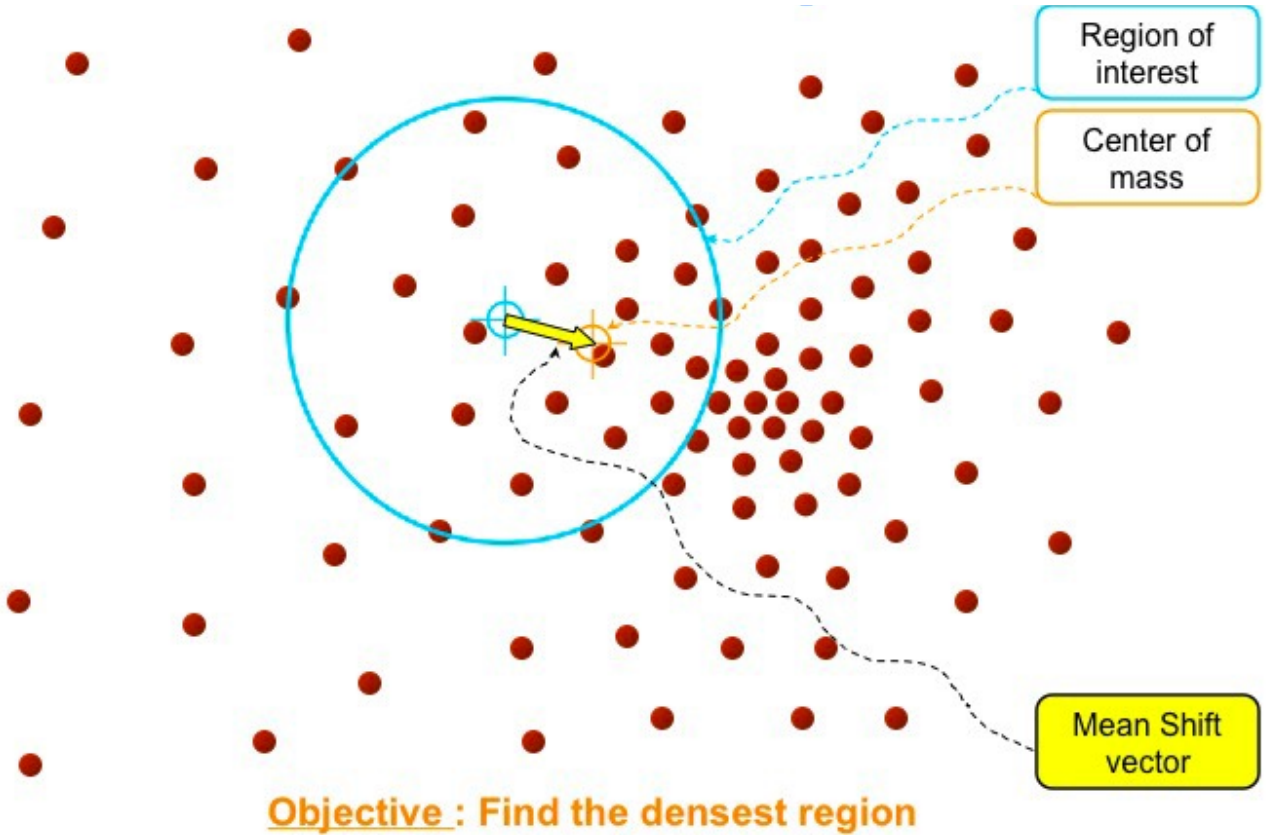
$$f(x) = \frac{1}{n}\sum_{i=1}^{n} K(x - x_i) \qquad \text{where} \qquad K(x) = C\ k(\|x\|^2)$$

Then the gradient should be

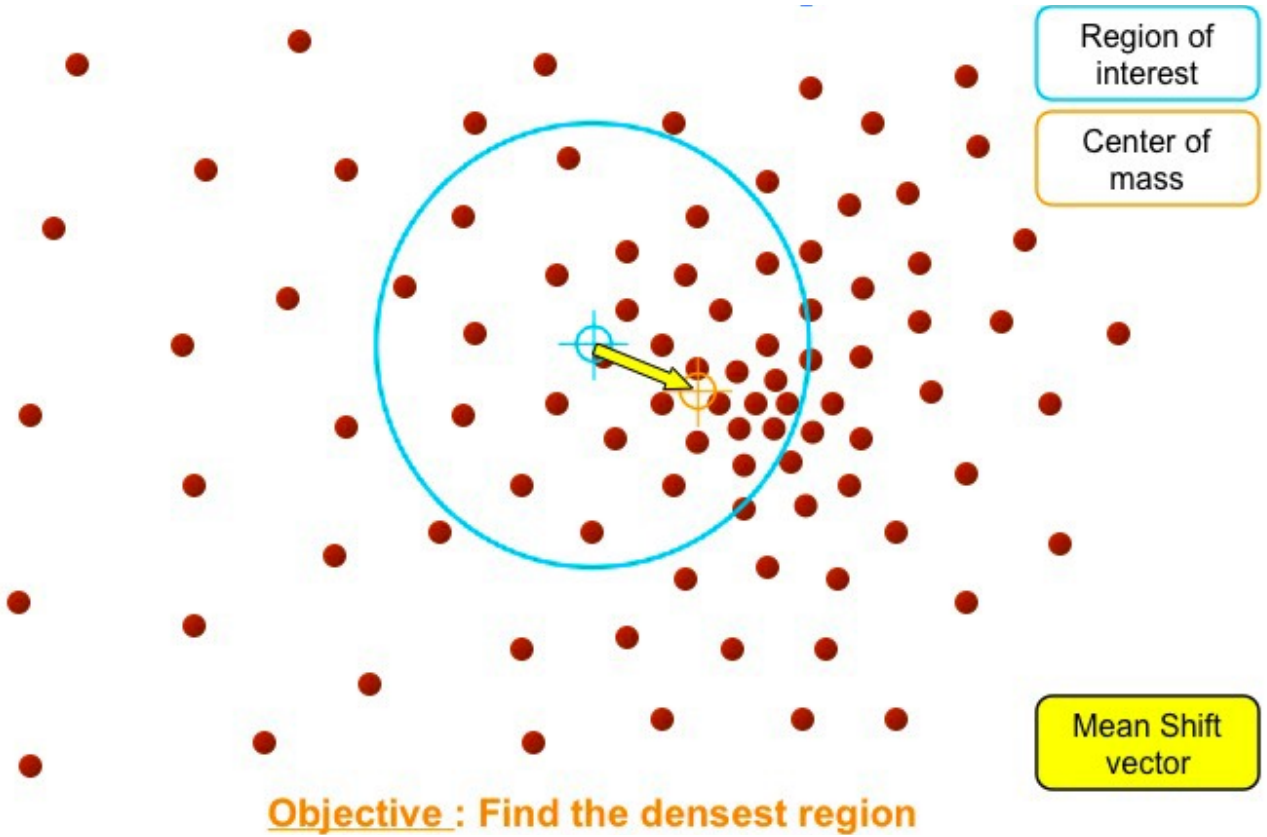
$$\nabla f(x) = \frac{2C}{n}\sum_{i=1}^{n}(x - x_i)k'(\|x - x_i\|^2) = 0 \quad \Rightarrow \quad x^{new} = \frac{\sum_{i=1}^{n} x_i\ k'(\|x - x_i\|^2)}{\sum_{i=1}^{n} k'(\|x - x_i\|^2)}$$

Most common kernel (’ball’): Gaussian kernel, for which the derivative also is a Gaussian.

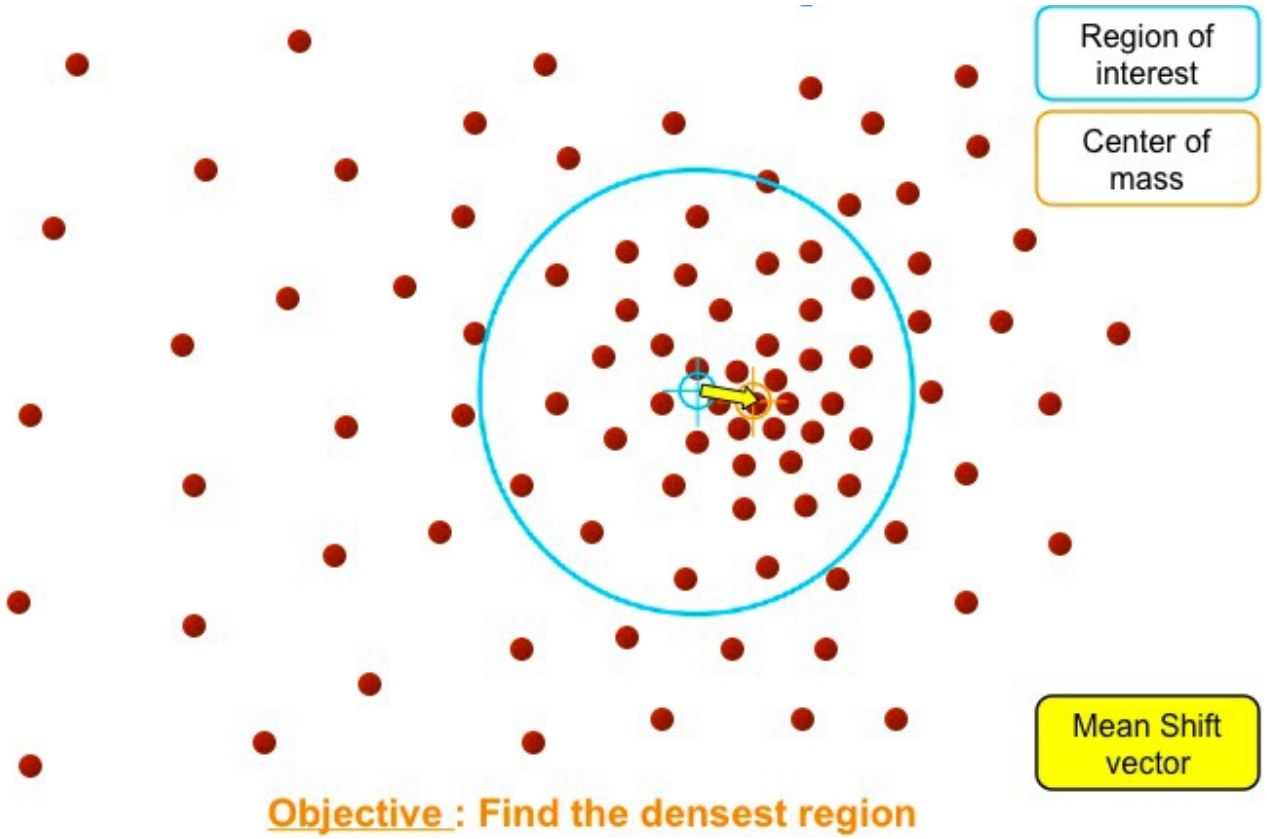
# Mean-shift segmentation



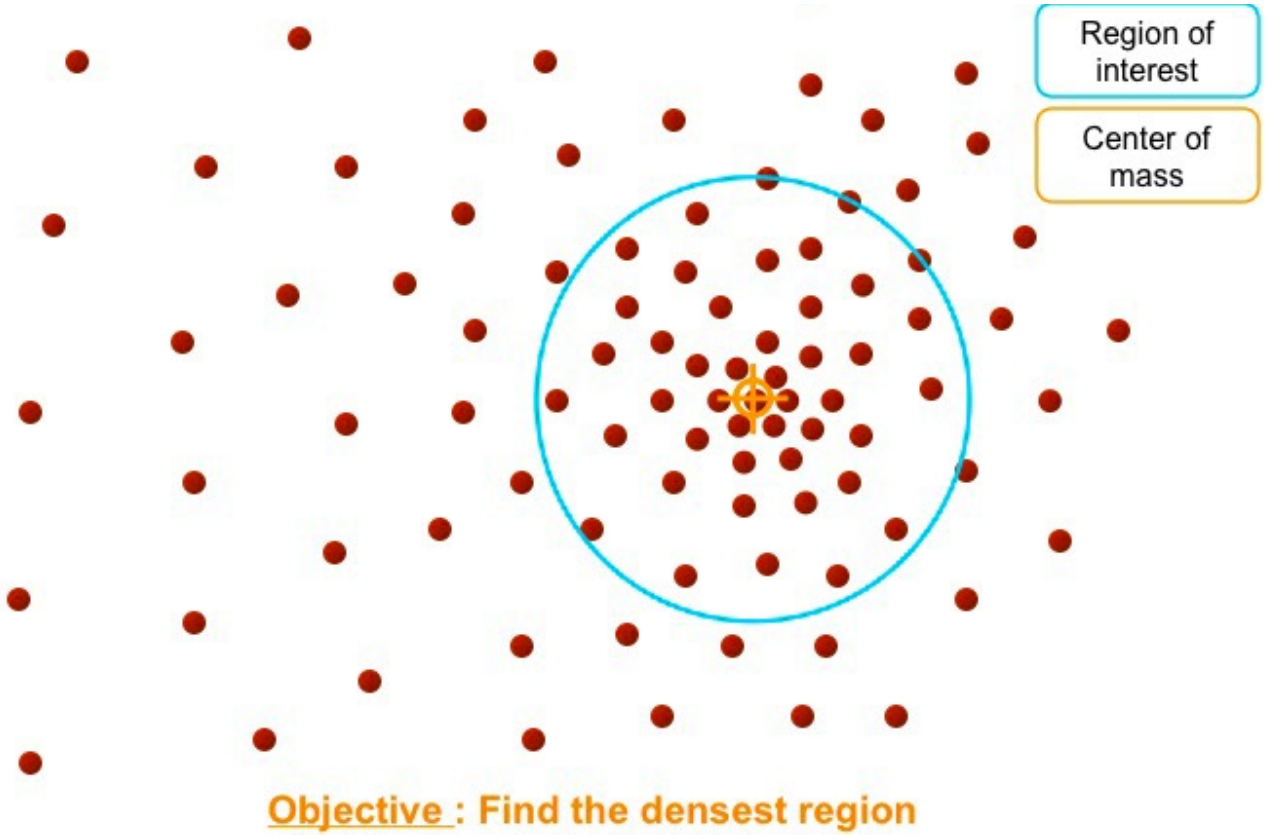
# Mean-shift segmentation



# Mean-shift segmentation

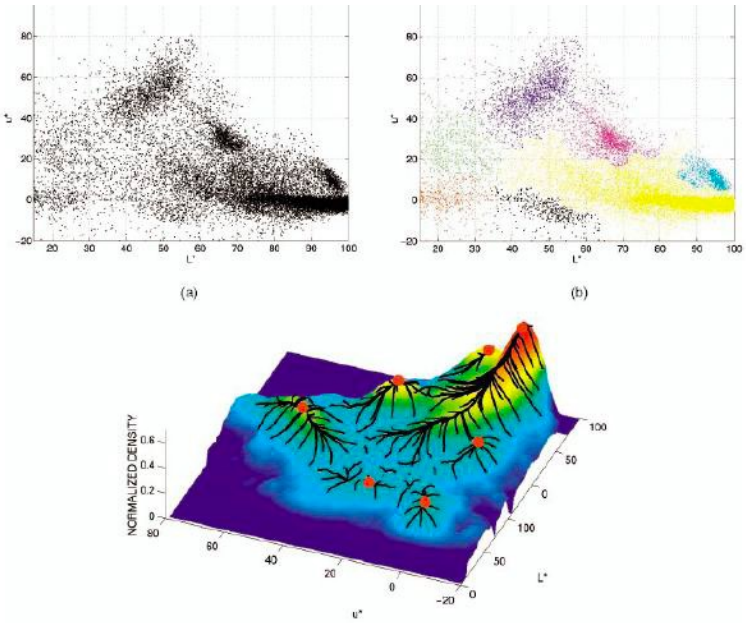


# Mean-shift segmentation



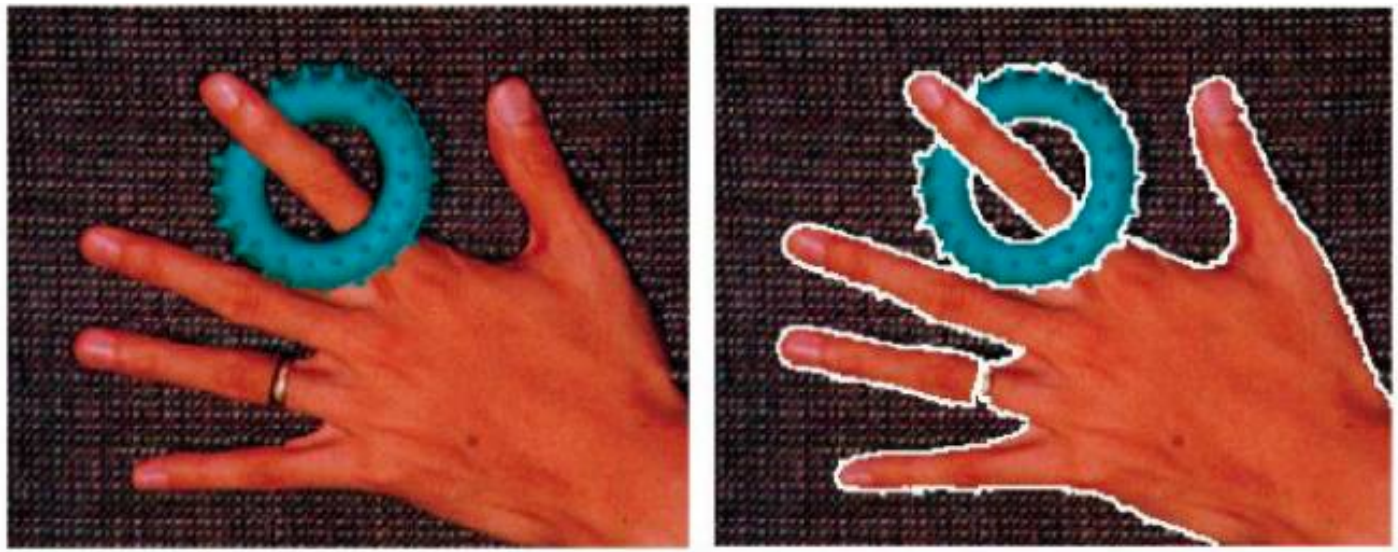
# Mean-shift segmentation

- The density function will have peaks (also called modes).
- If we start at each pixel and do gradient-ascent, we will converge to one of these modes. Then cluster image based on the modes pixels converged to.



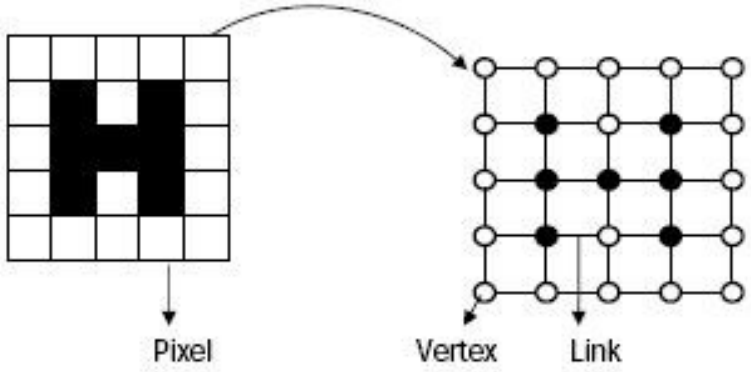
# Mean-shift segmentation example

- Get a segmentation by starting with the 5D value of each pixel, iterate and see which peak (mode) you end up with.
- This is very different from K-means, since iterations are done for each pixel in sequence, instead of for all pixels simultaneously.



# Graph theory in image segmentation

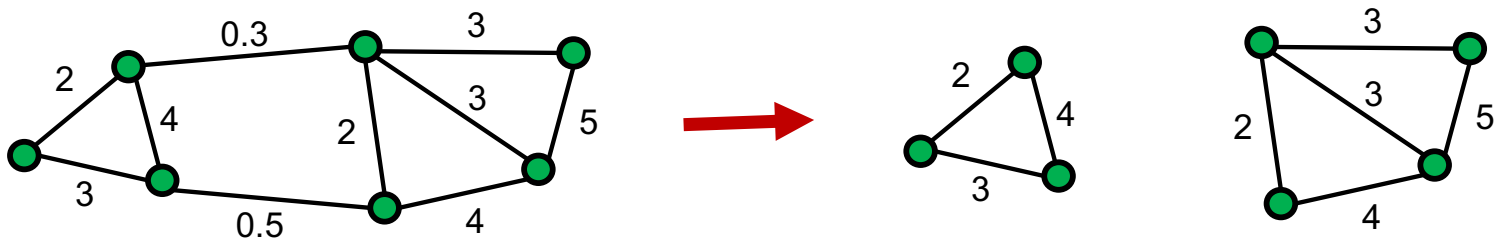
- Graph theory can be used to analyse and segment images.
- Each pixel (or superpixel) in the image corresponds to a node.
- Neighbouring pixels are connected by links.
- Nodes and links have weights.
  - Node weights are often based on the pixel colours, and
  - Link weights on similarities between neighbouring pixels.



Four-way Connected Graph

# Graph theoretic clustering

- In graph-theoretic clustering, the links in the graph represents similarities between the nodes.
- These can be summarized as a large affinity (similarity) matrix $W$.
- Problem: Split the graph in two (or more) pieces so that the cut links have as low weights as possible.



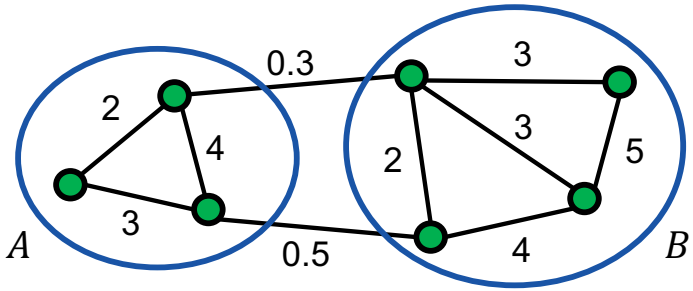

# Measuring affinity (similarity)

- The affinity matrix can for example have the following elements:
  - Intensity: $W(x,y) = e^{-|I(x)-I(y)|^2/(2s^2)}$
  - Position: $W(x,y) = e^{-|x-y|^2/(2s^2)}$
  - Colour: $W(x,y) = e^{-|c(x)-c(y)|^2/(2s^2)}$
- Here $s$ represents a scale factor that can be thought of how different two pixels can be and we still regard them as similar.
- The best methods combine many possible similarity measures.
- Nowadays, common to use features learned with deep networks.

# Normalized cuts

- Goal: Maximize sum of within cluster similarities, while minimizing sum of across cluster similarities.
- Minimize Normalized Cut: $Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$

  where
  - $A$ and $B$ are two disjoint sets of vertices and $V = A \cup B$.
  - $cut(A,B)$ – sum of links between vertices in $A$ and $B$.
  - $assoc(A,V)$ – sum of links connected to any vertex in $A$.
- Segmentation found by solving a generalized eigenvalue problem.




$cut(A,B) = 0.8$
$assoc(A,V) = 9.8$
$assoc(B,V) = 17.8$
$Ncut(A,B) = 0.1265$

# Normalized cuts

- Let $W$ be affinity matrix and $D$ diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
- Minimizing $Ncut(A, B)$ is equivalent to solving

$$\min_y \frac{y^T(D - W)y}{y^T D y}$$

  where elements in $y$ indicate whether nodes belong to $A$ or $B$.
- Equivalent to solving the generalized eigenvalue problem

$$(D - W)y = \lambda D y$$

  or after normalization

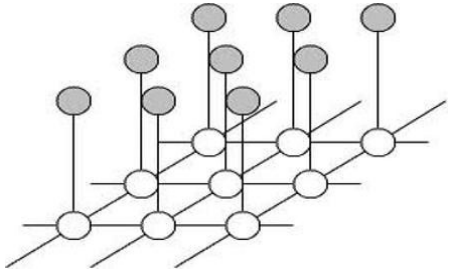
$$\left(I - D^{-1/2} W D^{-1/2}\right) z = \lambda z, \quad \text{where } z = D^{1/2} y$$

- Not necessary to remember the details of this! Look it up when you need it!

# Normalized cuts example

# Semantic segmentation with CRF

- Create a Conditional Random Field (CRF) with one node per pixel and links between all neighbouring nodes.
- Assume you have a set of models. Examples:
  - $L = \{foreground, background\}$
  - L = $\{sky, ground, vegetation, people, cars, houses\}$
- Each pixel $x$ has a label $l_x \in L$ denoting which model it belongs to.
- Models are usually trained beforehand using a large database.
  - A model can for example be represented by Gaussian Mixture Models.



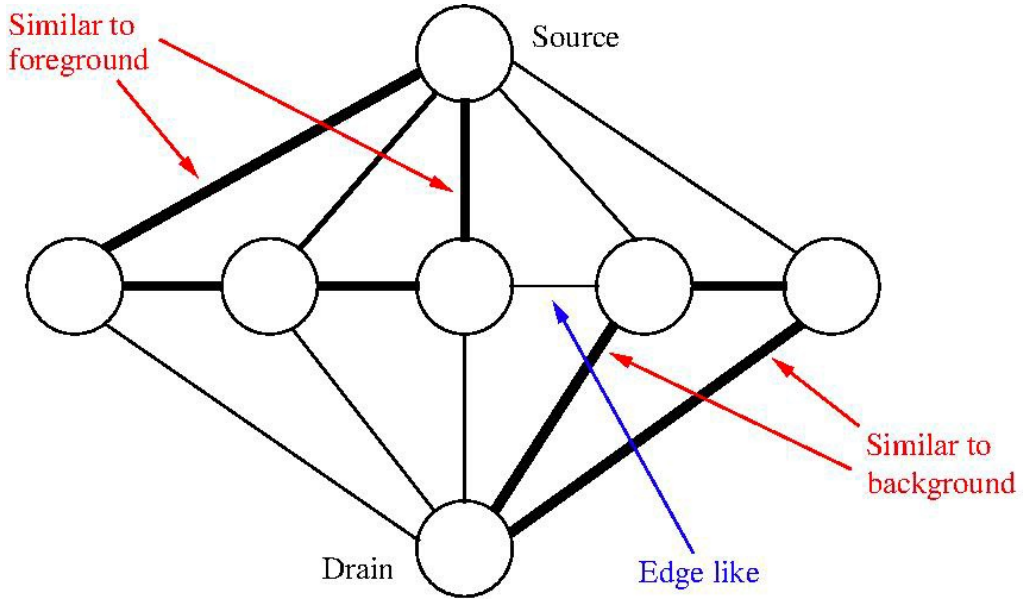

# Semantic segmentation with CRF

- Idea: Set up the problem as a energy (cost) minimization problem.
- Find the combination of labels that minimizes the cost

$$E = \sum_{x} \psi_x(l_x) + \sum_{x,y \in N_x} \psi_{x,y}(l_x, l_y)$$

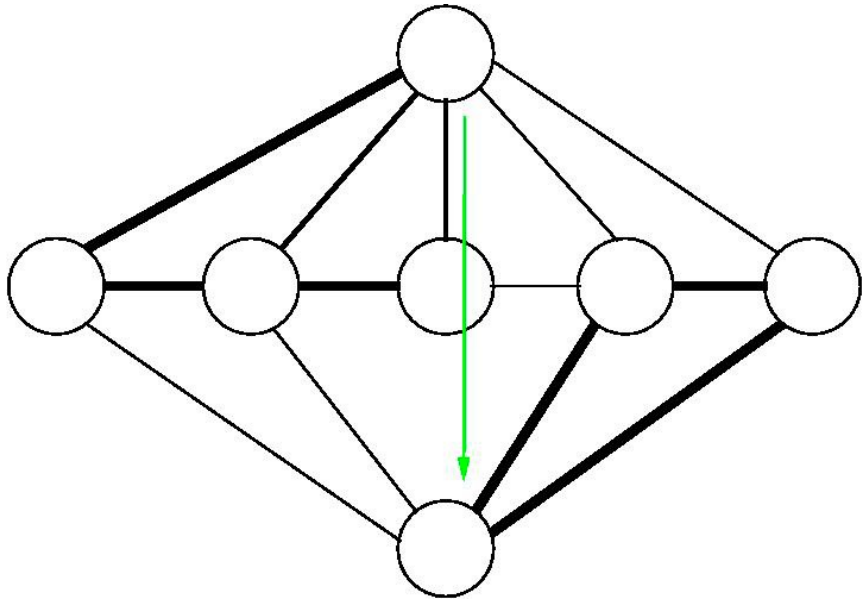
- Cost per node $\psi_x(l_x)$
  - Low cost if pixel has a colour similar to model $l_x$ , high otherwise.
- Cost per link $\psi_{x,y}(l_x, l_y)$
  - Normally $\psi_{x,y}(l_x, l_y) = 0$, if $x = y$.
  - Low cost if pixels have different colours (edge)
  - High cost if pixels have similar colours (smooth surface)
- Result of energy minimization: assign pixels to most similar models, while aligning borders of segments to edges.

# Graph cuts with 5 pixels



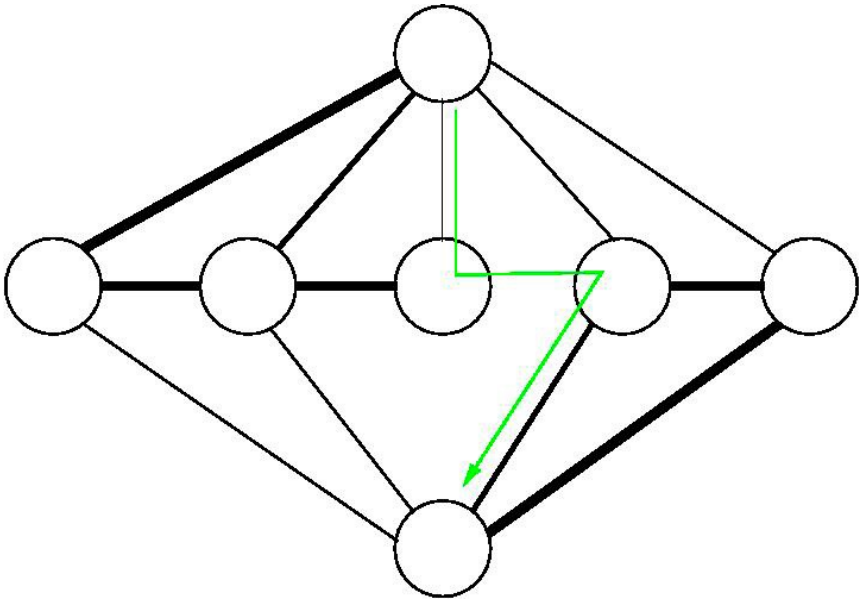

- Graph cuts can be used to minimize the energy $E$ for $|L| = 2$.
- Add links to Source (foreground) and Drain (background) with costs given by $\psi_x(l_x)$, and links with costs $\psi_{x,y}(l_x, l_y)$ between neighbours.
- Goal: Find the lowest cost split of the graph into two pieces.


# Graph cuts



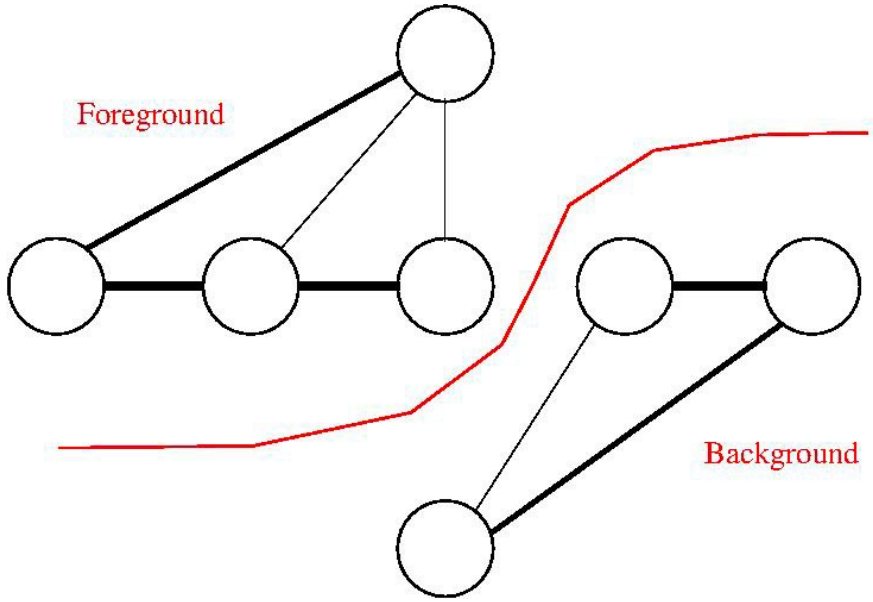
- Push ”flow” from Source to Drain until a link saturates.
- Imagine links are pipes and you push as mush flow (water) as possible.
- Width of links illustrate how much more flow (capacity) can be pushed.

# Graph cuts



- Try all possible paths from Source to Drain and gradually empty the capacity.

# Graph cuts



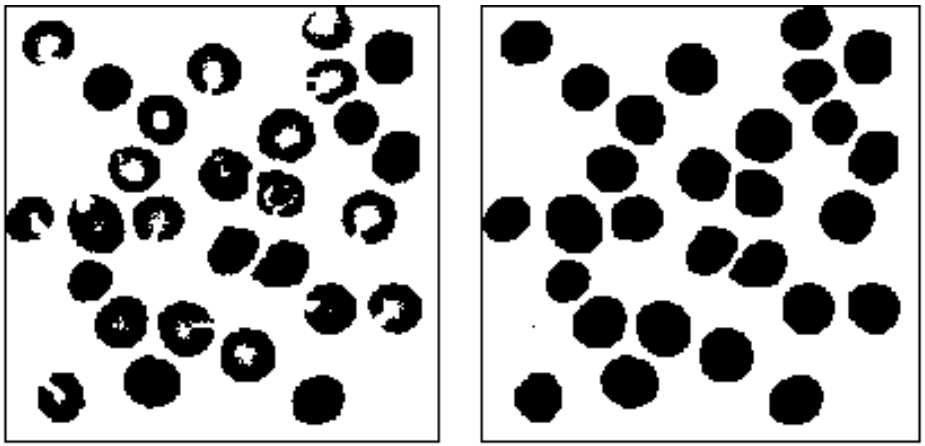

- In the end no more flow can be pushed from Source to Drain and it stops.
- Nodes have been divided into two parts, connected to either Source or Drain.
- The saturated links correspond to the non-zero terms in $E$.

# GrabCut: example

- Left: A couple of strokes are applied to create colour models represented as Gaussian Mixture Models of background and foreground.
- Right: Afterwards background can be changed to something else.

# Mathematical Morphology

- Often there is a need to ’clean up’ the segmentation.
  - Some segments are simply too small and only exist due to noise.
  - Others have holes, because interior colours are similar to the background.
- Solution: Apply (non-linear) morphological operations.



A B


# Mathematical morphology

- A morphological operator is defined by a structuring element (or kernel) of size $n \times n$ and a set operator.
- Just like a normal filter, the kernel is shifted over the image and its elements are compared to the underlying pixels.
- If the two sets of elements match the condition defined by the set operator, the pixel underneath the centre of the structuring element is set to a pre-defined value (0 or 1 for binary images).

| | |
|---|---|
| $A$ | the (usually binary) image |
| $A^C$ | the complement of the image (inverse) |
| $A \cup B$ | the union of images $A$ and $B$ |
| $A \cap \mathrm{B}$ | the intersection of images $A$ and $B$ |
| $A - B = A \cap \mathrm{B^C}$ | the difference between $A$ and $B$ (pixels in $A$ not in $B$) |
| $\#A$ | the cardinality of $A$ (area of the object) |

# Dilation

- The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels.
- Areas of foreground pixels grow in size, while holes in regions become smaller.
- Let $A$ and $B$ denote sets in $\mathbb{R}^2$ with elements $a$ and $b$. Then

$$A \oplus B = \{c \in \mathbb{R}^2 : c = a + b\}$$

- Typically: $A$ = binary image, $B$ = mask (structuring element)

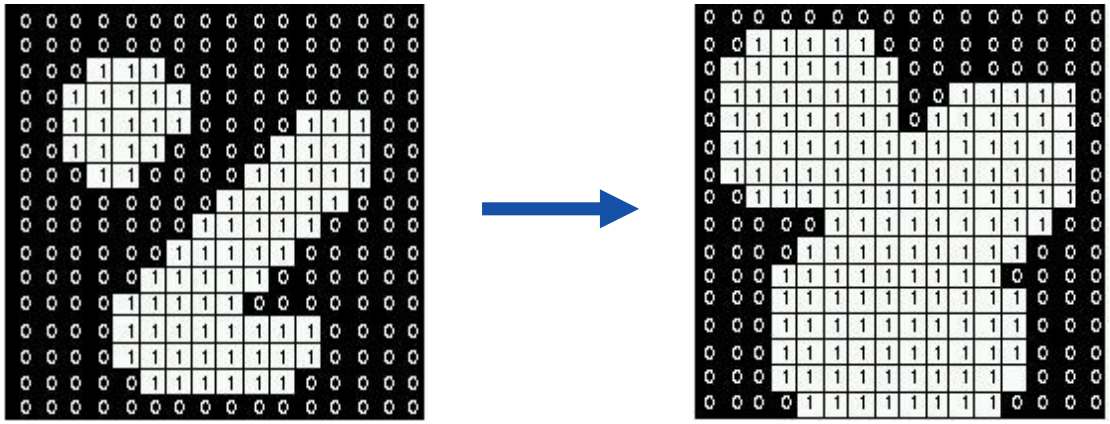
  Example: $3\times3$ structuring element

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$B = \{(-1,-1), (0,-1), (1,-1), (-1,0), (0,0), (1,0), (-1,1), (0,1), (1,1)\}$$

# Dilation

- If at least one pixel in the structuring element coincides with a foreground pixel in the image underneath, then the input pixel is set to the foreground value.
- If all the corresponding pixels in the image are background, however, the input pixel is left at the background value.




# Erosion

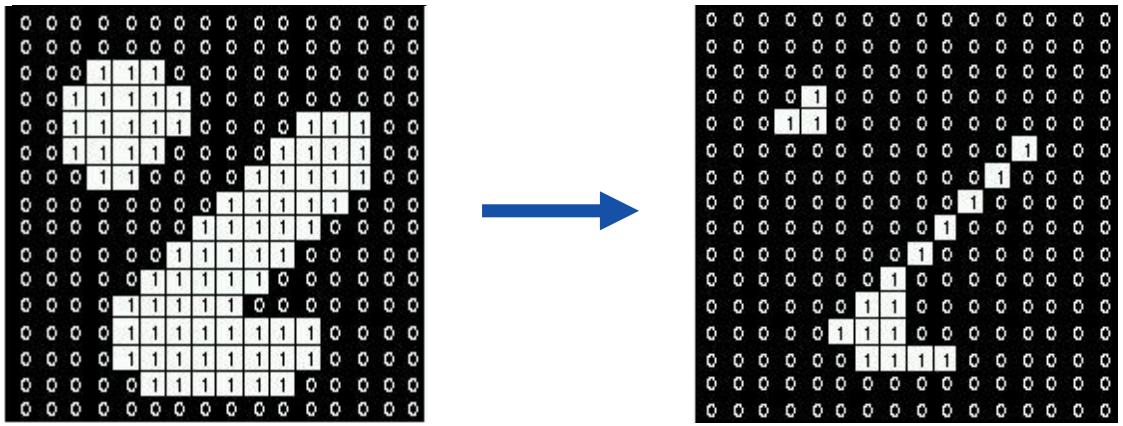
- The basic effect of the operator on a binary image is to gradually shrink the boundaries of regions of foreground pixels.
- Thus areas of foreground pixels decrease in size, and holes within those areas become larger.
- Let $A$ and $B$ denote sets in $\mathbb{R}^2$ with elements $a$ and $b$. Then

$$A \ominus B = \{c \in \mathbb{R}^2 : c + b \in A, \forall b \in B\}$$

- This is a bit harder to understand. Easier to see visualized.

# Erosion

- If for all the pixels in the structuring element, the corresponding pixel in the image underneath is a foreground pixel, then the input pixel is left as it is.
- If any of the corresponding pixels in the image are background, however, the input pixel is also set to background value.



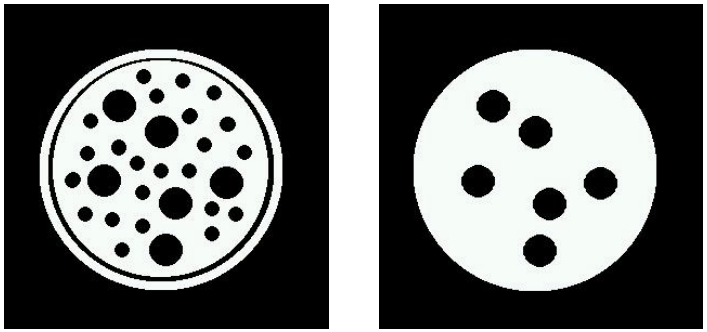
# Opening and closing

- Opening: an opening is defined as an erosion followed by a dilation using the same structuring element for both operations.

$$A \circ B = (A \ominus B) \oplus B$$

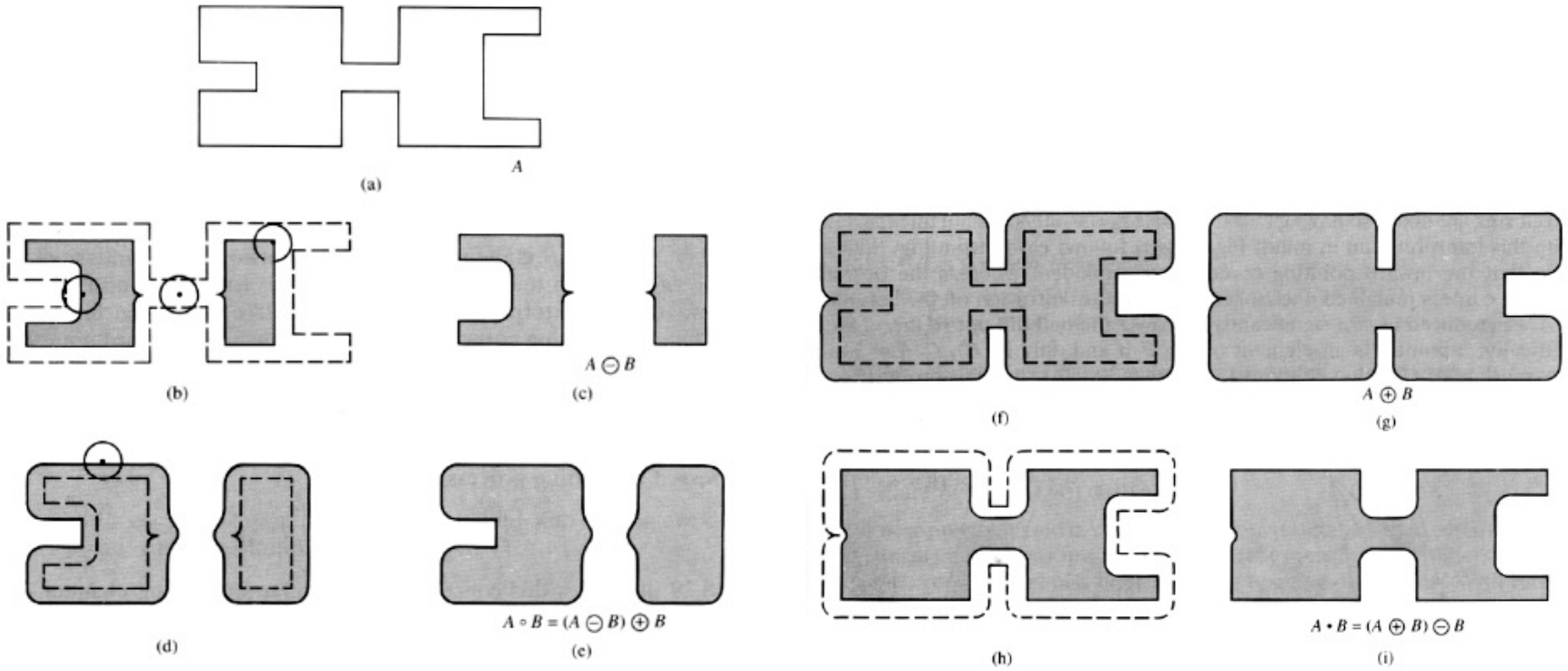
- Closing: an closing is defined as an dilation followed by a erosion using the same structuring element for both operations.

$$A \bullet B = (A \oplus B) \ominus B$$



- Opening is the dual of closing: opening the foreground pixels with a structuring element is equivalent to closing the background pixels with the same element.

# Opening and closing example




# Summary of good questions

- What is the purpose of image segmentation?
- What is Gestalt Theory?
- How to select a threshold for histogram based segmentation?
- How does K-means work?
- Why is spatial coherence important for segmentation?
- What does a mean-shift algorithm try to do and how?
- What is an affinity matrix?
- What does Normalized Cuts try to optimize?
- What cost functions does an energy formulation for segmentation often include?
- What is the purpose of graph cuts for segmentation?
- How does a graph cut work?
- What does a morphological opening and closing operation do?


# Recommended reading

- Gonzalez & Woods: Chapters 9.1-9.3, 10.3, 10.5-10.6
- Szeliski: Chapters 4.3, 5.2.1-5.2.2, 6.4 and 7.5