Image Segmentation I

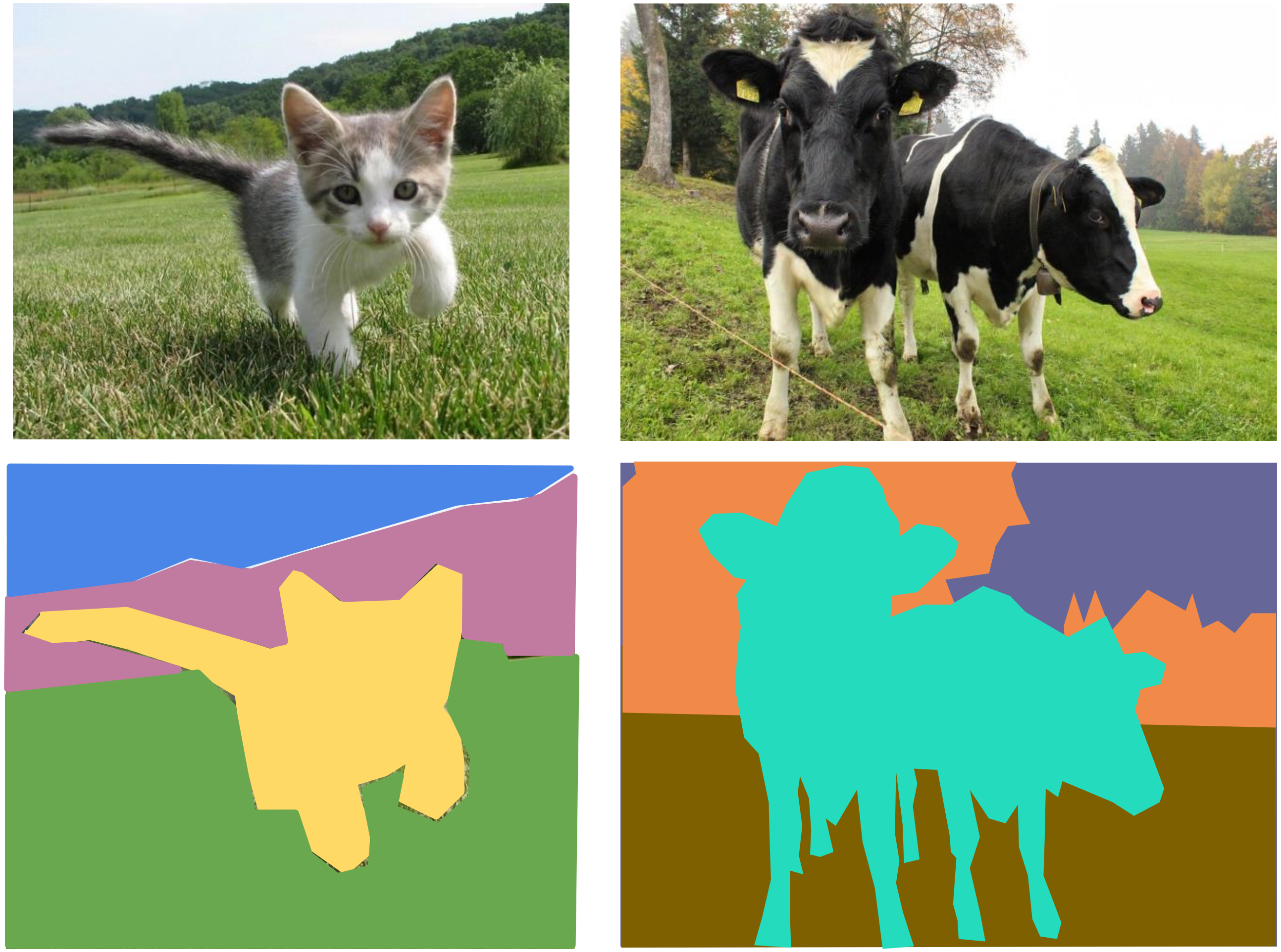
Semester 2, 2021

Kris Ehinger

Demo

- https://chocopoule.github.io/grabcutweb/

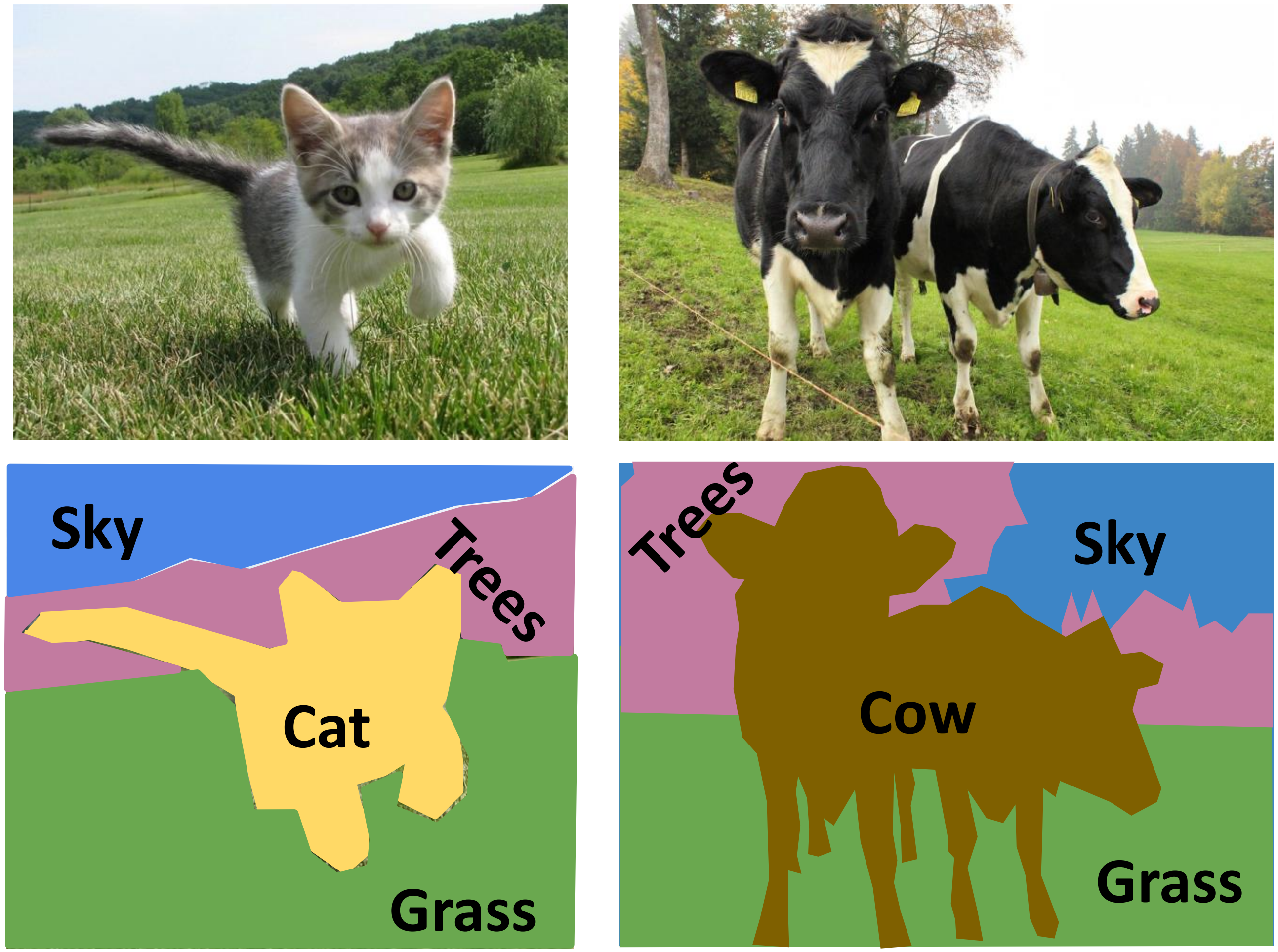
Segmentation



Separate image into different regions (objects, textures)

Image: J. Johnson

Semantic segmentation



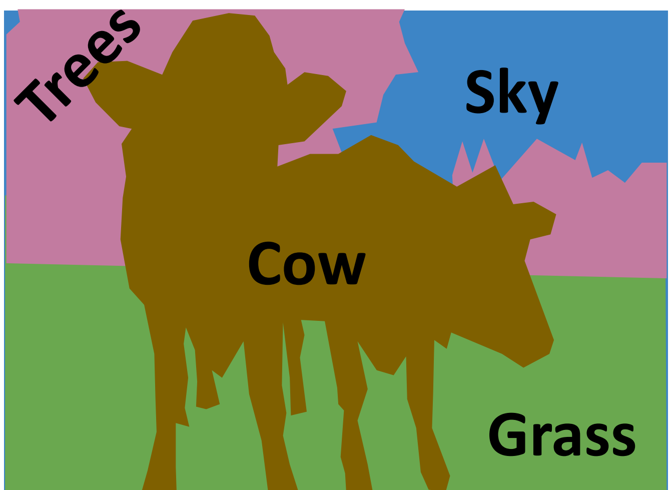
Separate image into different *labelled* regions

Image: J. Johnson

Instance segmentation

Semantic segmentation

Instance segmentation



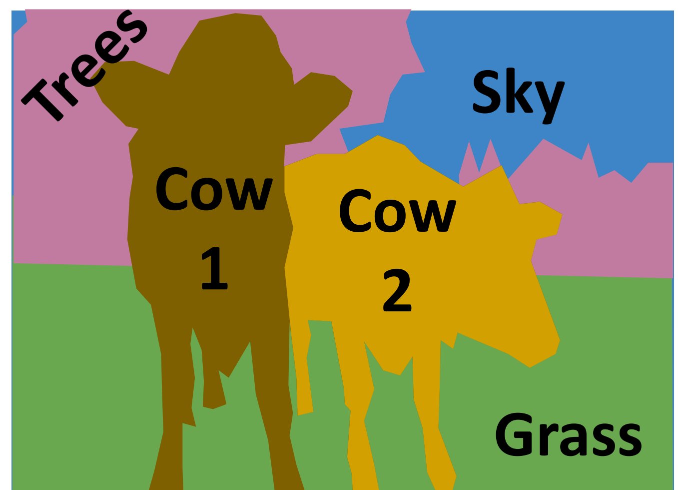
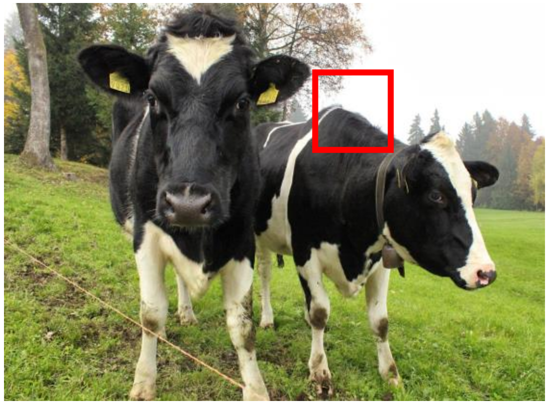
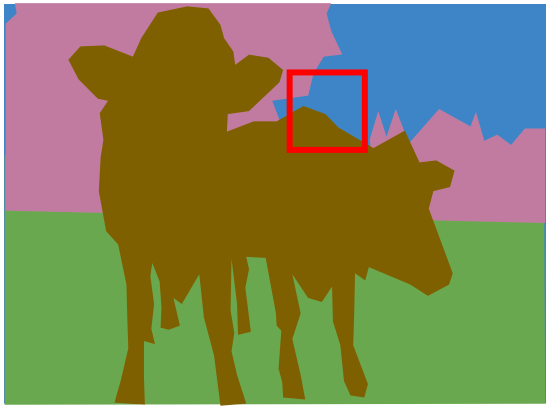


Image segmentation



Input: Image



Output: Pixel classification
(and, optionally, labels)

Clustering?
Graph cuts?
Classification?

Outline

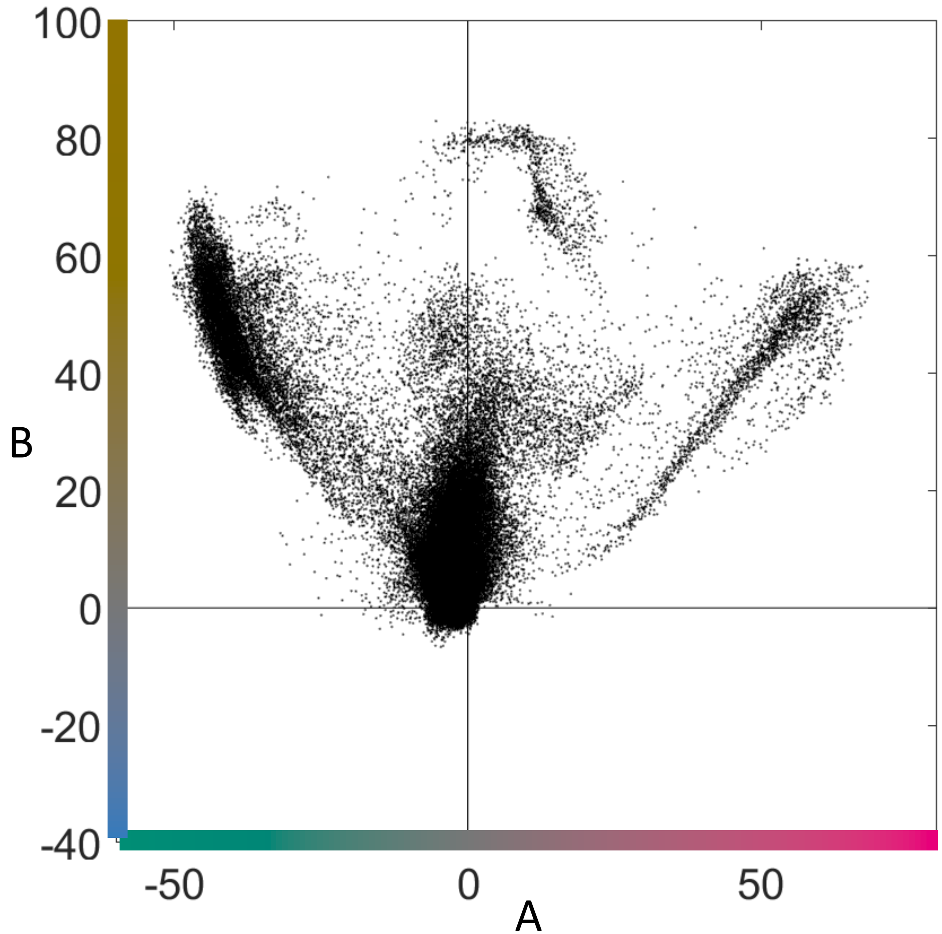
- Pixel clustering
- Superpixel segmentation
- Graph-based segmentation

Learning objectives

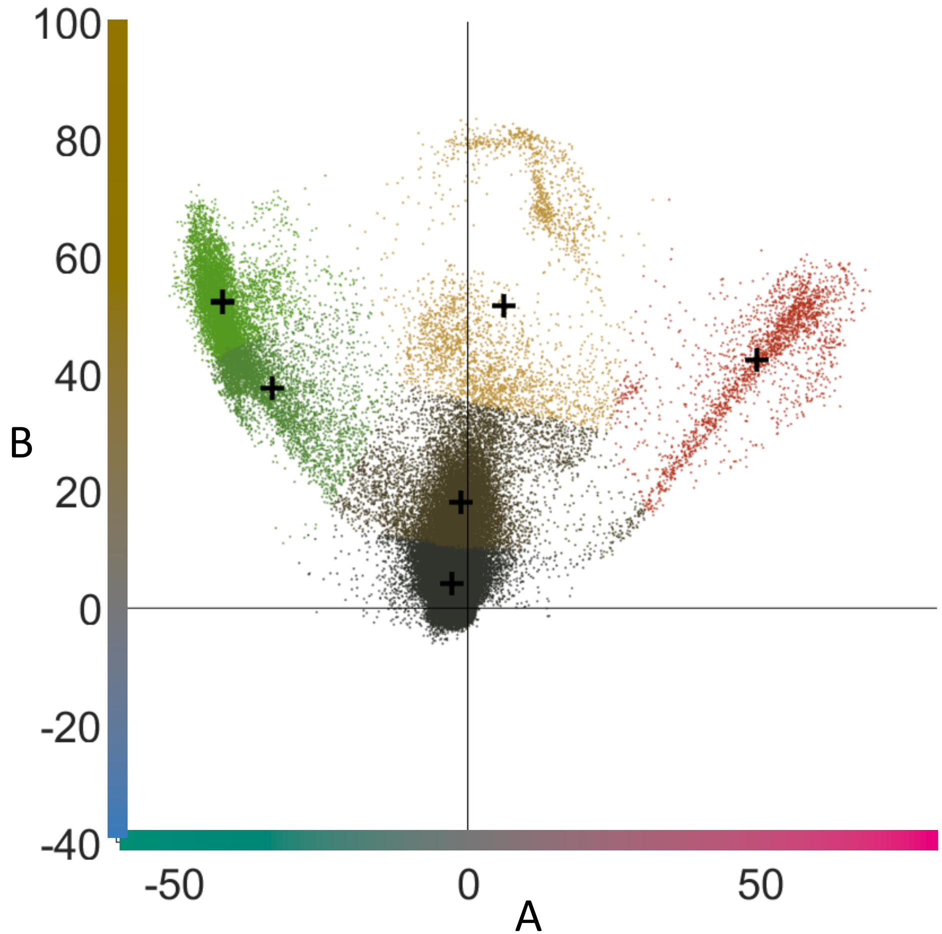
- Implement clustering algorithms for segmentation and compare/contrast clustering methods
- Implement an algorithm for computing superpixels and explain their common applications
- Explain graph-based methods for image segmentation

Pixel clustering

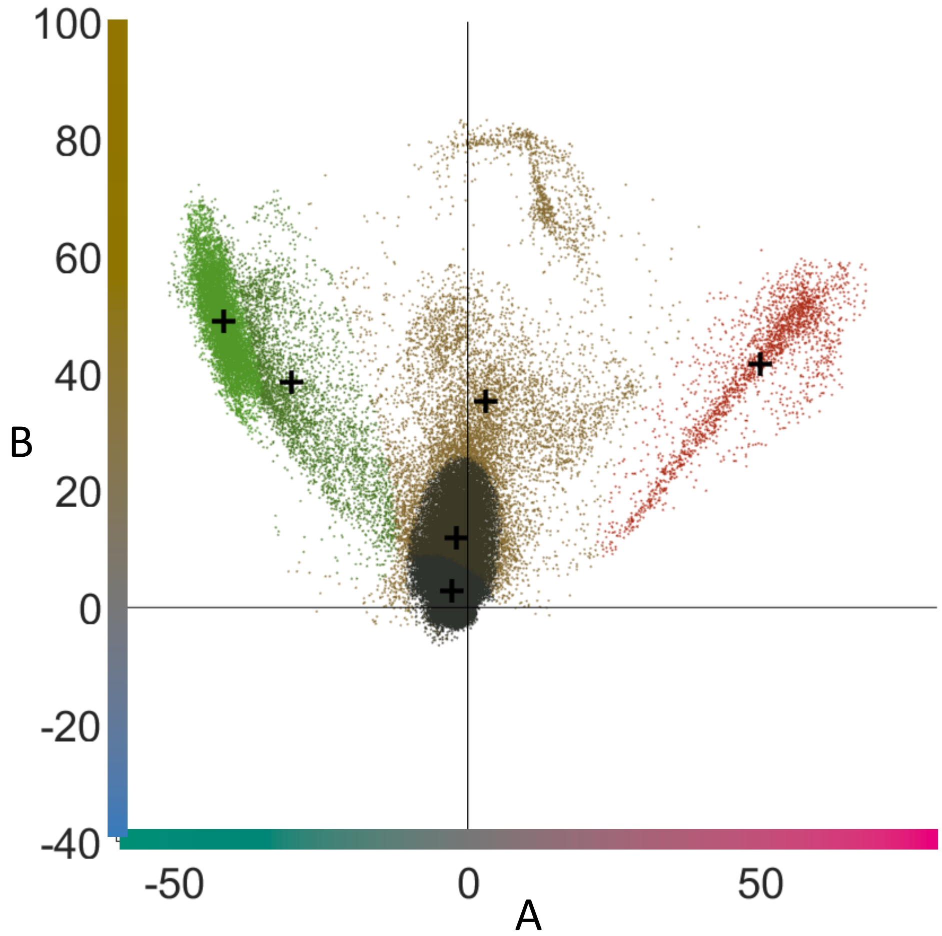
Colour clustering



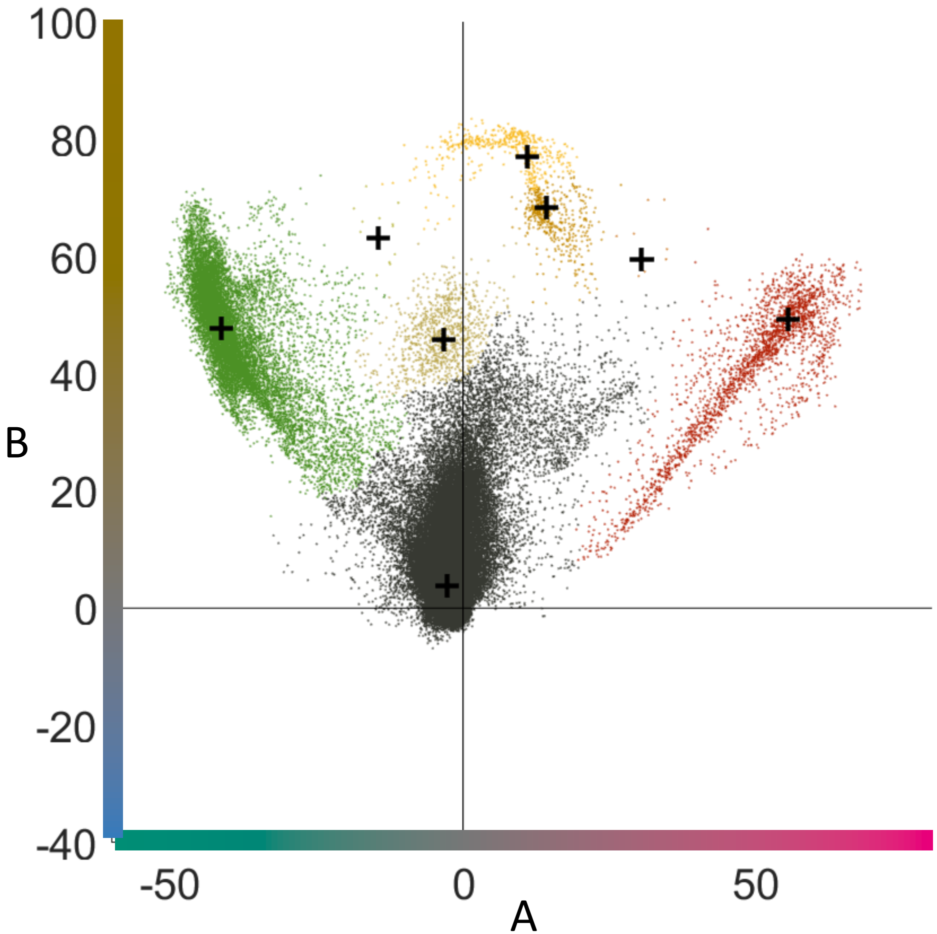
K-means (k = 6)



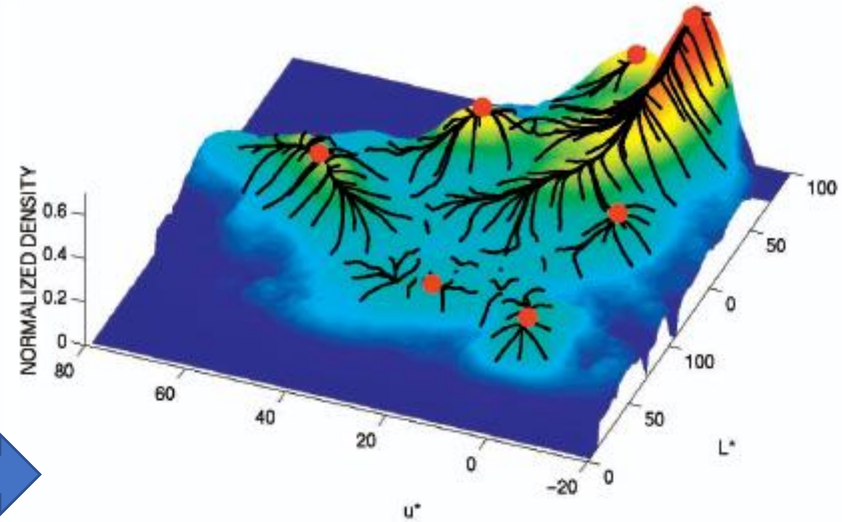
Gaussian mixture model (k = 6)

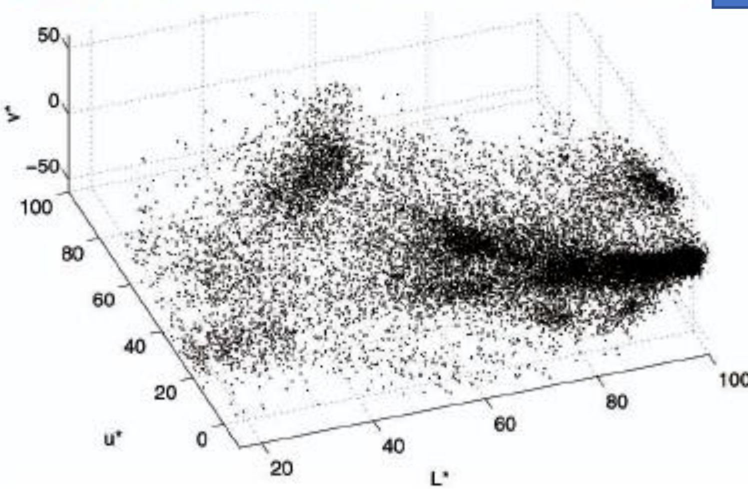


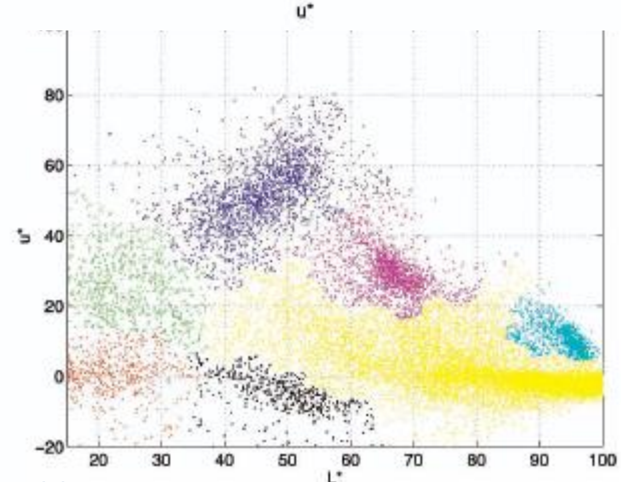
Mean shift (bandwidth = 7)



Mean shift clustering

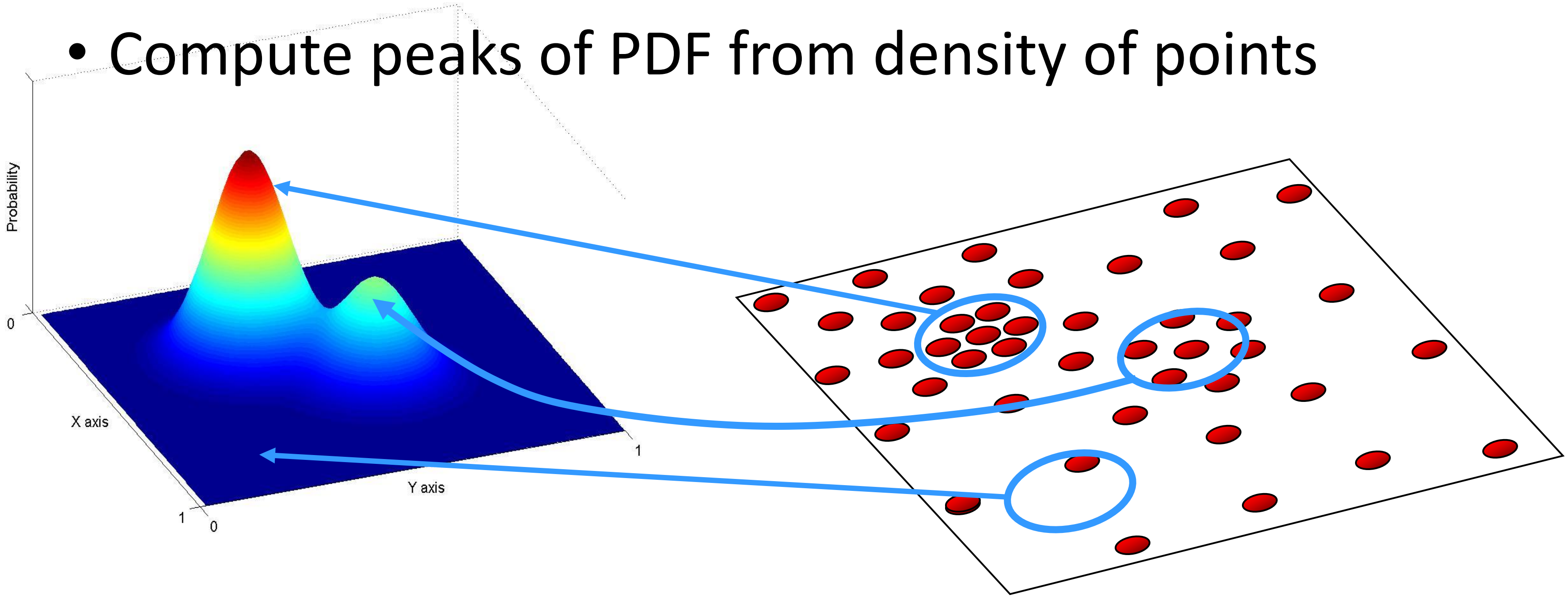






Mean shift clustering

- Assume points are samples from an underlying probability density function (PDF)
- Compute peaks of PDF from density of points



Assumed Underlying PDF

Real Data Samples

Image: Y. Ukrainitz & B. Sarel

Mean shift

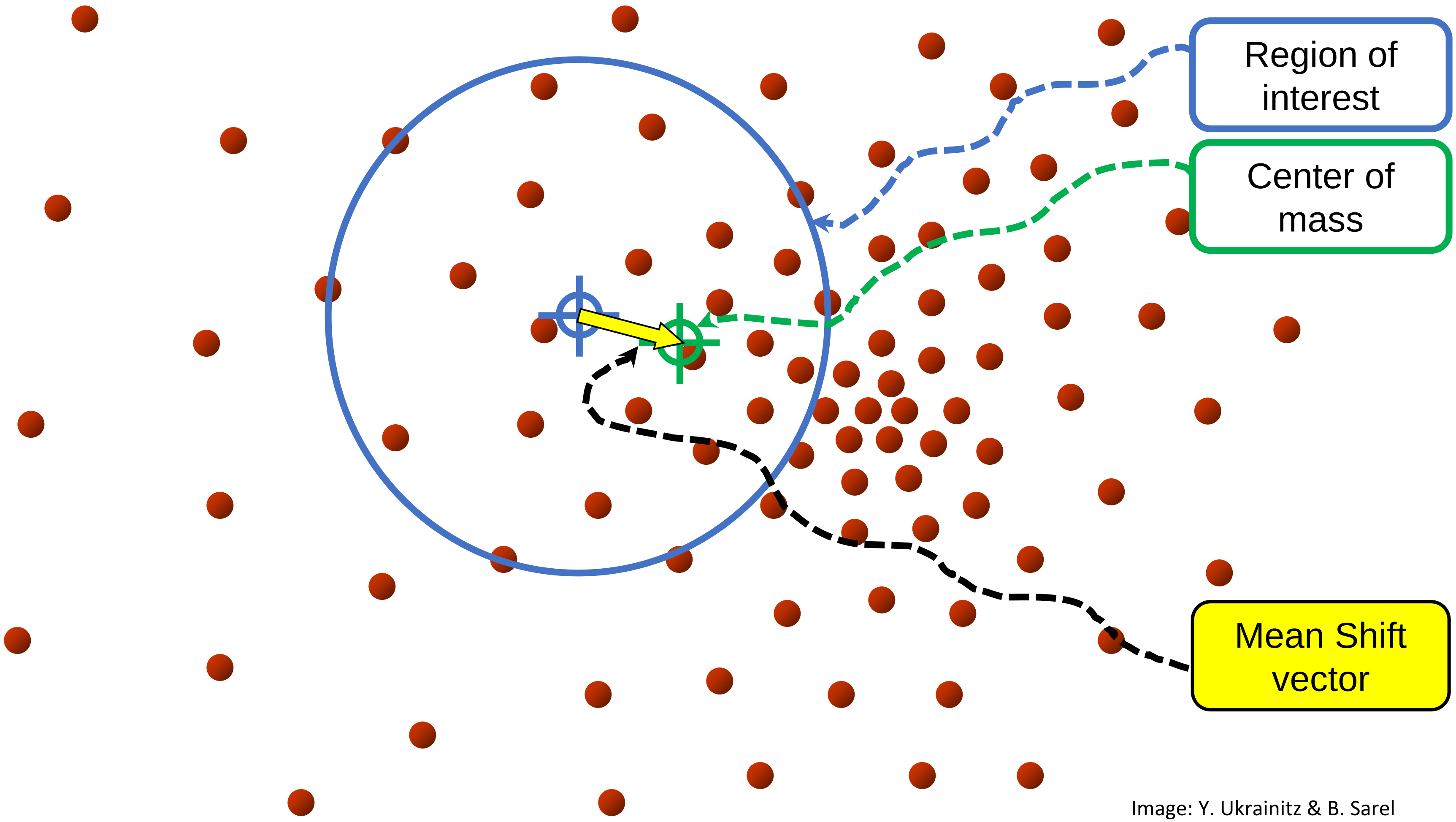
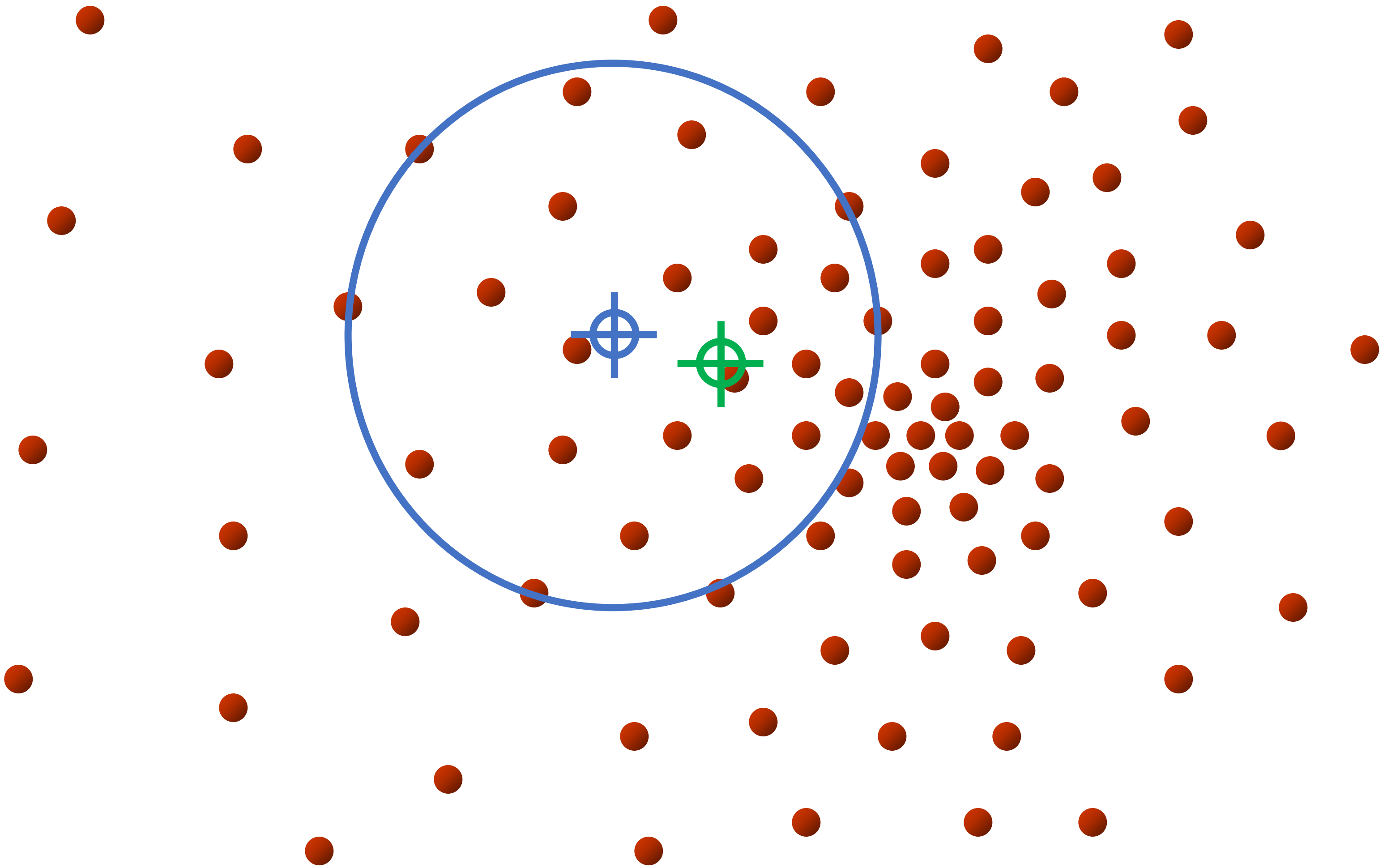
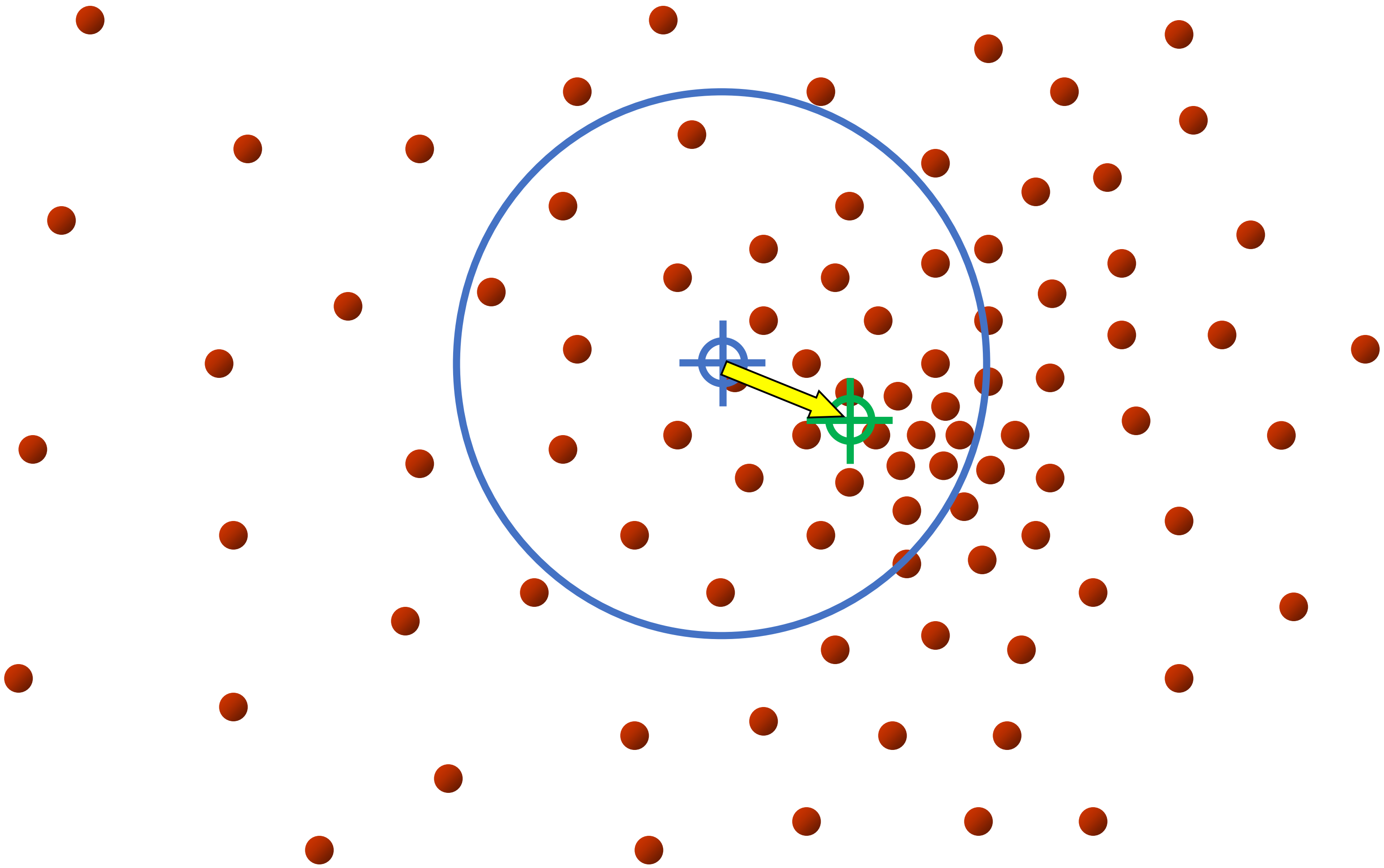


Image: Y. Ukrainitz & B. Sarel

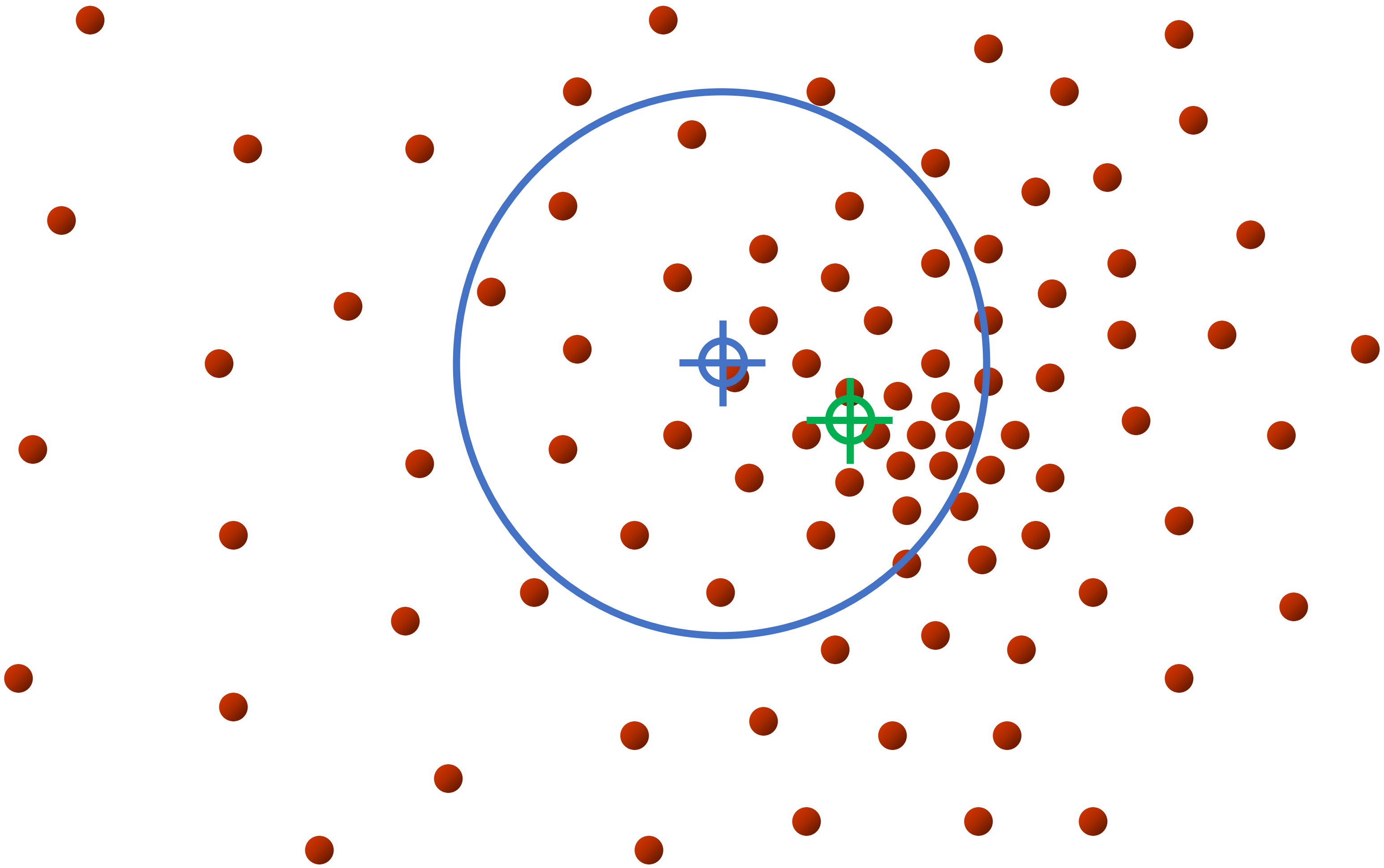
Mean shift



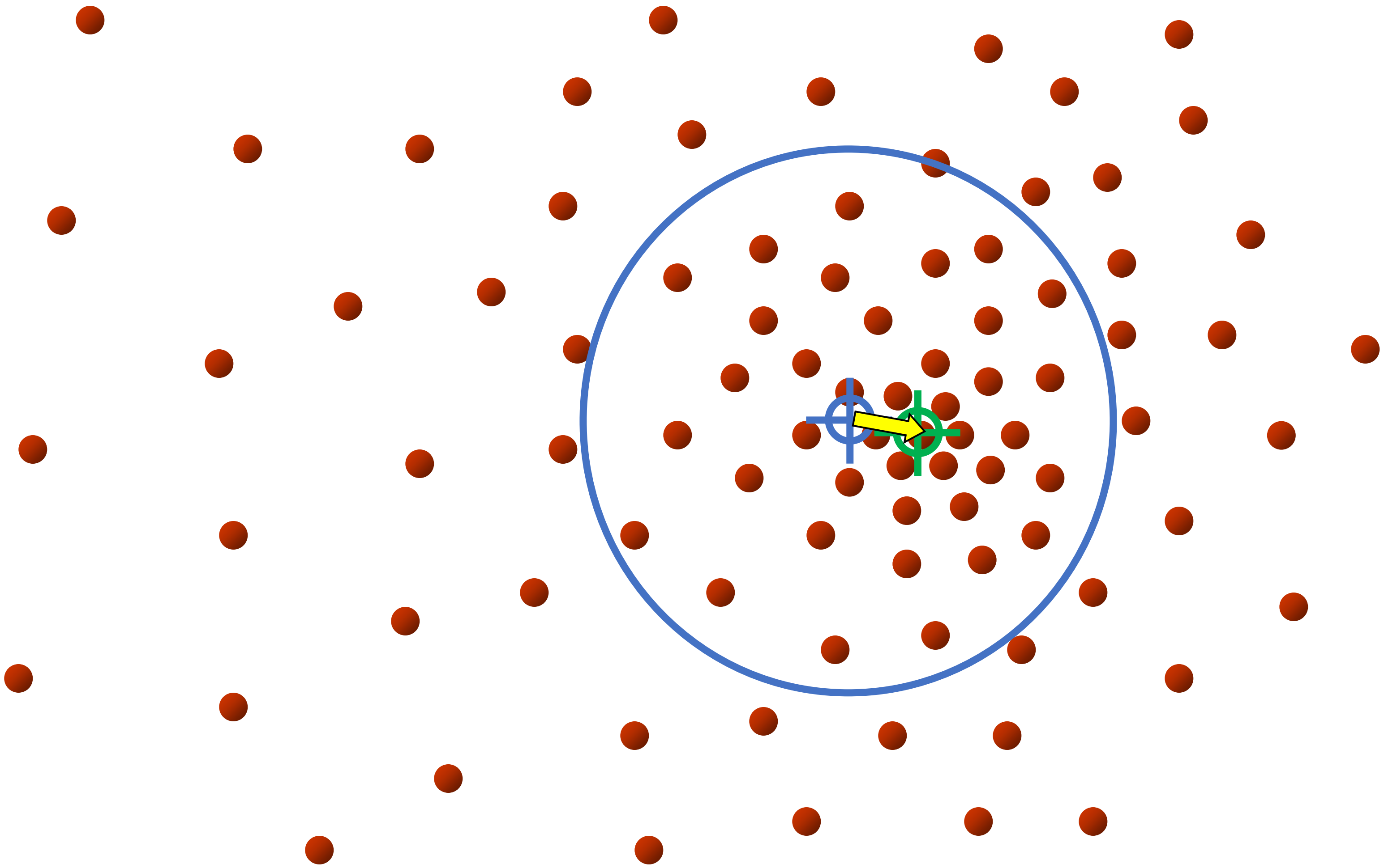
Mean shift



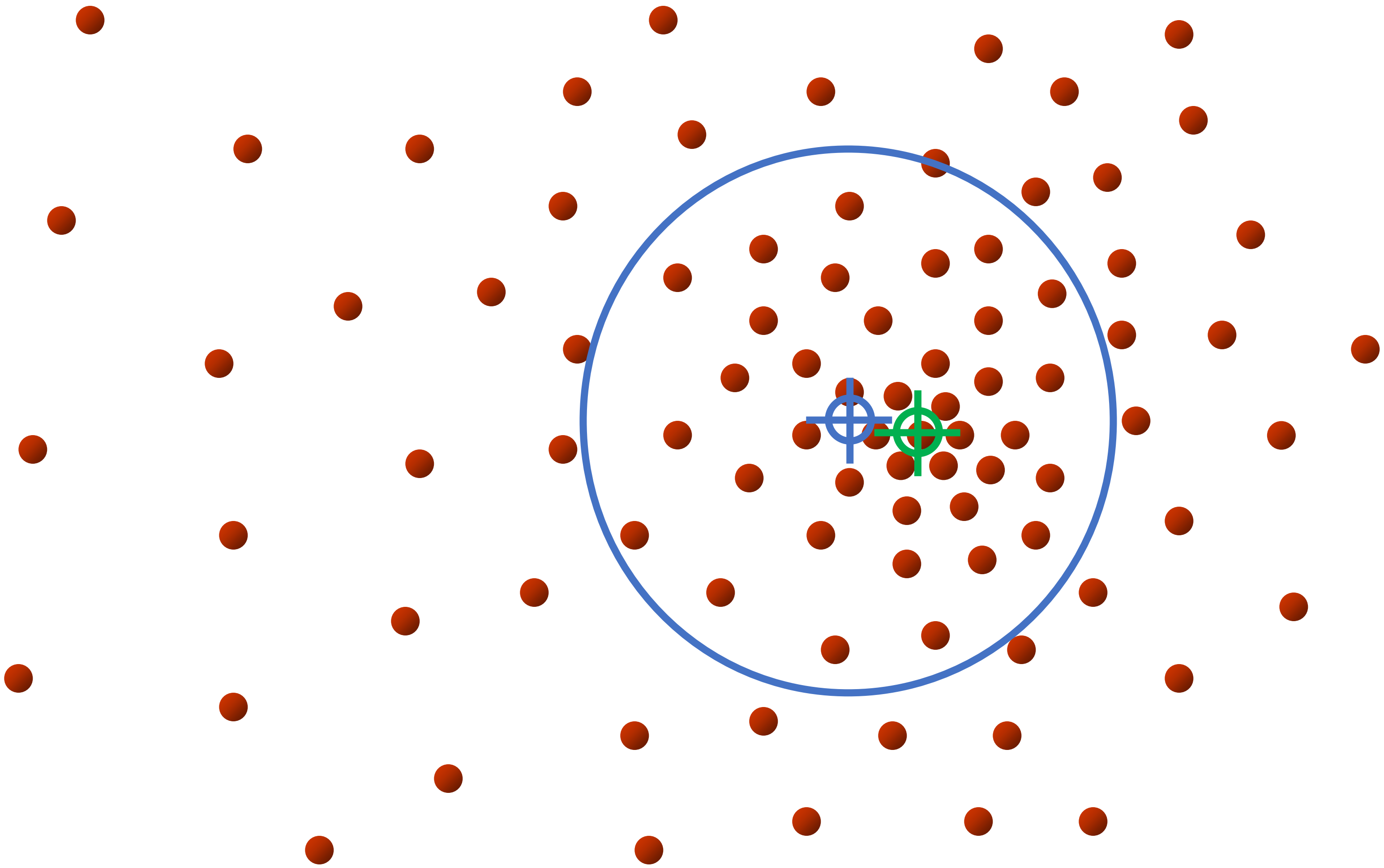
Mean shift



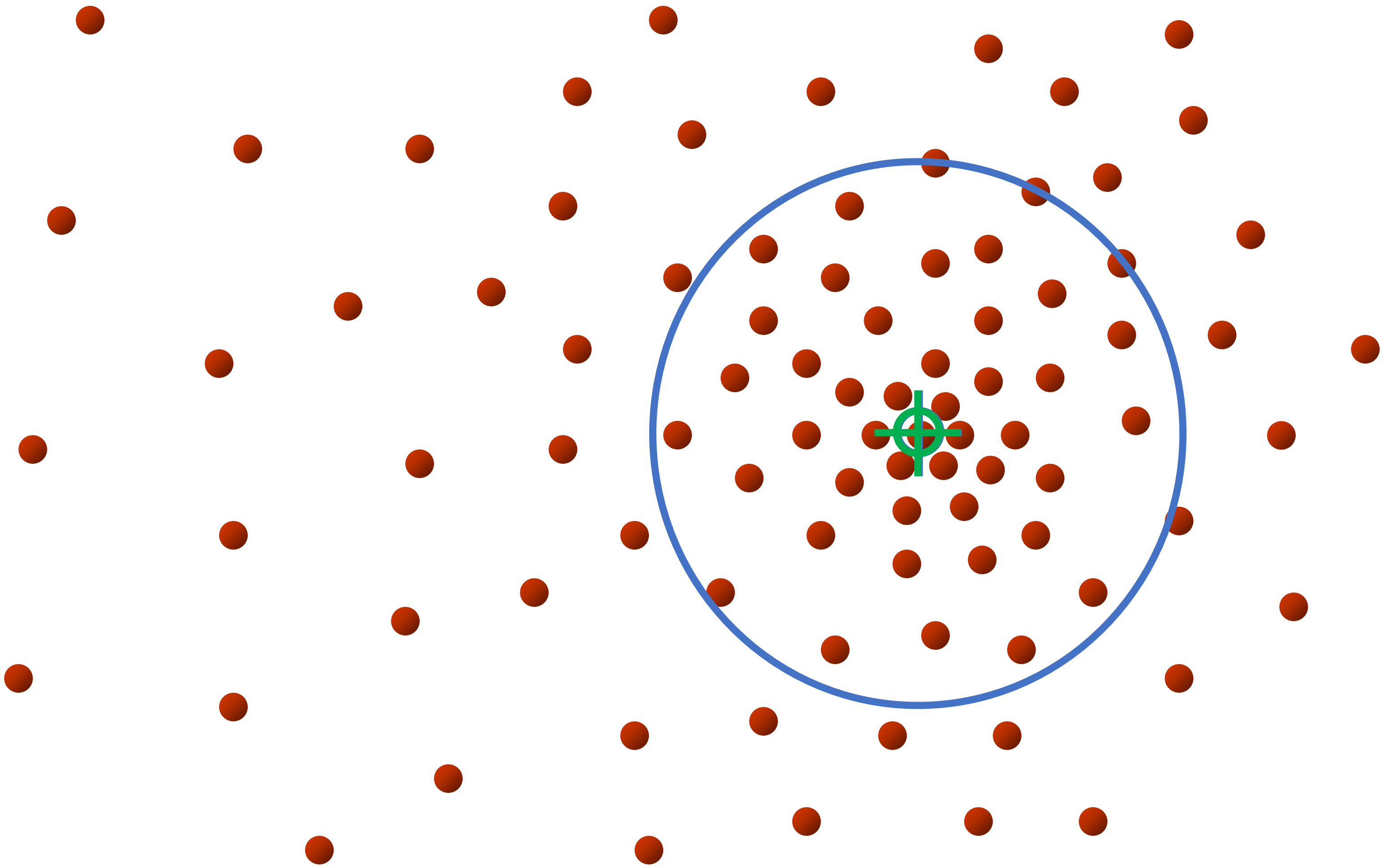
Mean shift



Mean shift



Mean shift



Mean shift algorithm

- Compute mean shift vector **m(x)**
- Translate kernel window by **m(x)**

Gaussian kernel:

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

$$\mathbf{m}(\mathbf{x}) = \left[\boxed{\frac{\sum_{i=1}^{n} \mathbf{x}_i \, g\left(\frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)}} - \boxed{\mathbf{x}} \right]$$

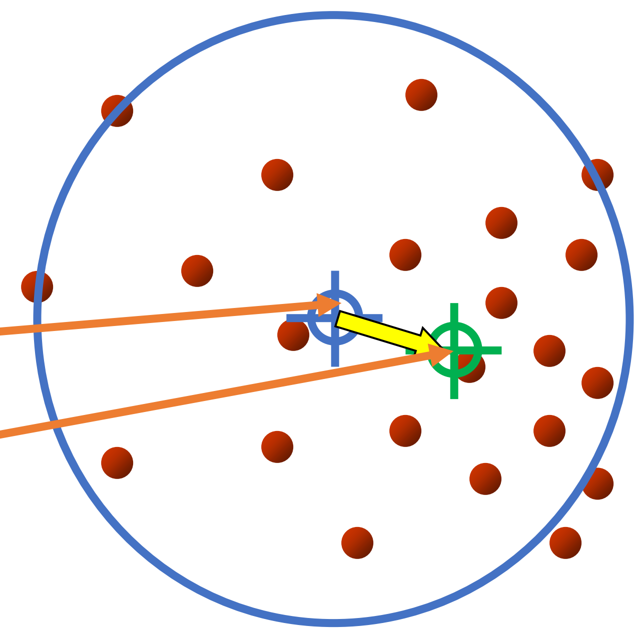


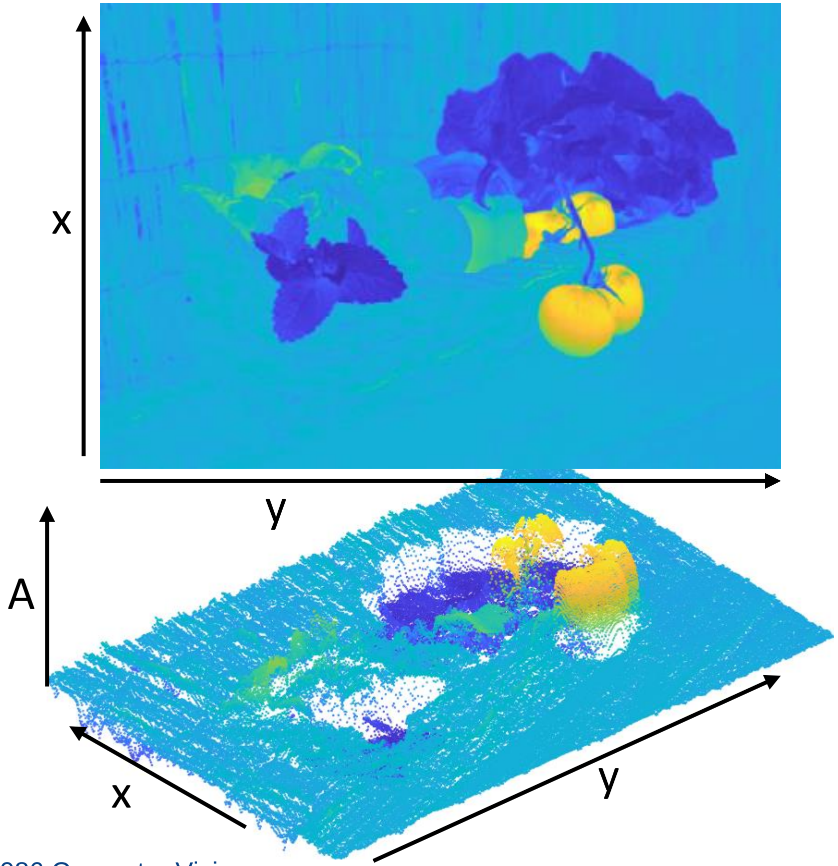
Image: Y. Ukrainitz & B. Sarel

Mean shift algorithm

- For each point:
  - Centre a window on that point
  - Compute the mean of the data in the search window
  - Centre the search window at the new mean location
  - Repeat (b,c) until convergence
- Assign points that lead to nearby modes to the same cluster
- Free parameters: kernel (commonly Gaussian), bandwidth

Mean shift segmentation

- Cluster in spatial+colour space; e.g.: (x,y,R,G,B) or (x,y,L,A,B) coordinates



Mean shift parameters

Increasing bandwidth ↓

Increasing spatial bandwidth relative to colour →

Summary

- Pixel clustering is a fast, simple approach to image segmentation
- Example: mean shift clustering in the colour+spatial domain
  - Automatically discover number of clusters; no need to choose k
  - But do need to choose bandwidth
- Pixel clustering separates colour regions – regions may not correspond to objects

Superpixels

Superpixels

- **Oversegmentation** methods segment image into regions that are smaller than objects
  - Objects are separated from background
  - But objects are also separated into many parts
- Superpixels = groups of adjacent pixels with similar characteristics (e.g., colour)

Superpixel segmentation

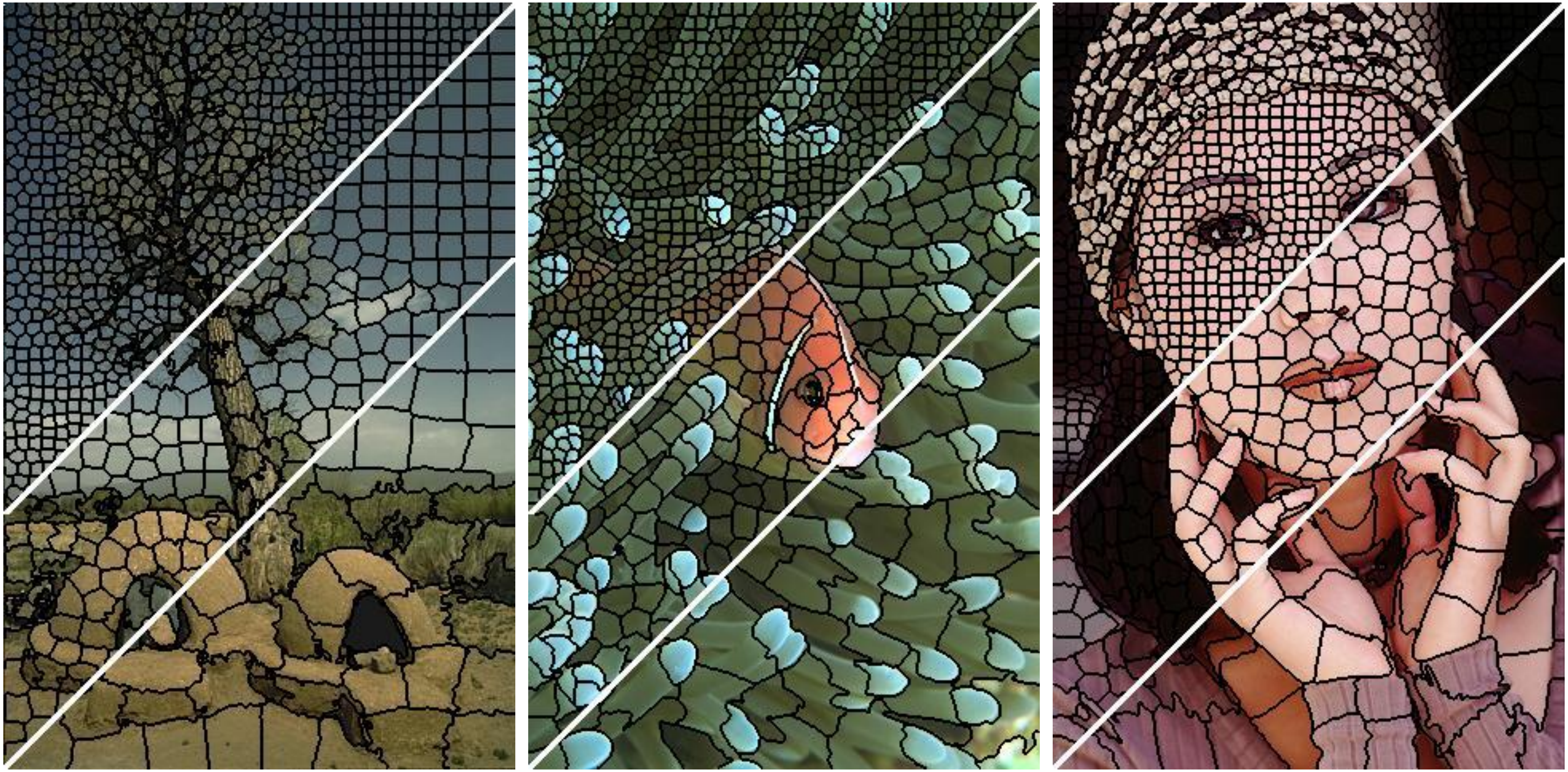


Image: https://www.epfl.ch/labs/ivrl/research/slic-superpixels/

SLIC superpixel algorithm

- Initialise cluster centres on non-edge pixels:
  - Initialise k cluster centres $c_k = [x_k, y_k, l_k, a_k, b_k]$ by sampling the image in a regular grid
  - For each centre c_k, check an N x N neighbourhood around c_k to find the pixel with lowest gradient. Set c_k to this pixel's [x, y, l, a, b].

SLIC superpixel algorithm

- For each cluster centre c_k:
  - In a 2M x 2M square neighbourhood around c_k, measure pixel similarity to c_k
  - Assign pixels with similarity < threshold to cluster k
  - Compute new cluster centre c_k
- Repeat until average change in cluster centres (L1 distance) falls below a threshold
- Similarity measure: $D = D_{lab} + \frac{\alpha}{M} D_{xy}$

$$D_{lab} = \sqrt{(l - l)^2 + (a - a_k)^2 + (b - b_k)^2}$$

$$D_{xy} = \sqrt{(x - x_k)^2 + (y - y_k)^2}$$

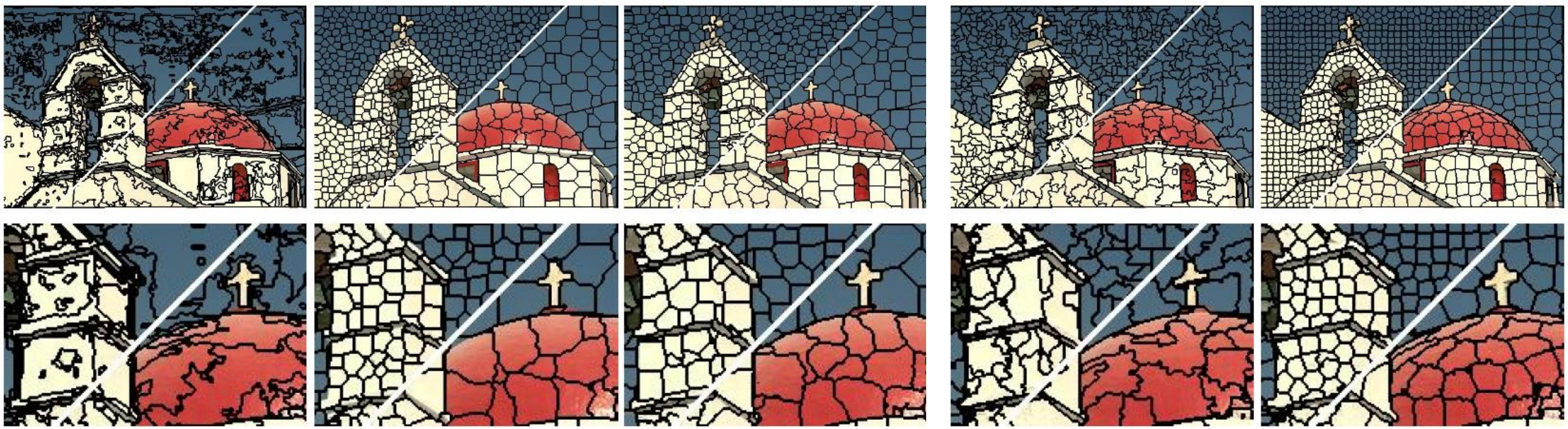
$\alpha =$ weighting parameter

SLIC superpixel algorithm

- Similarity metric does not guarantee that clusters will be connected pixels
- To enforce connectivity, pixels not connected to main cluster are re-assigned to closest adjacent cluster

Superpixel methods

Graph-based methods | Gradient-descent-based



Felzenszwalb & Huttenlocher (2004)	Veksler, Boykov, & Mehrani (2010) – spatially compact	Veksler, Boykov, & Mehrani (2010) – constant colour	QuickShift (Vedaldi & Soatto, 2008)	SLIC

Image: Archana, et al. (2012)

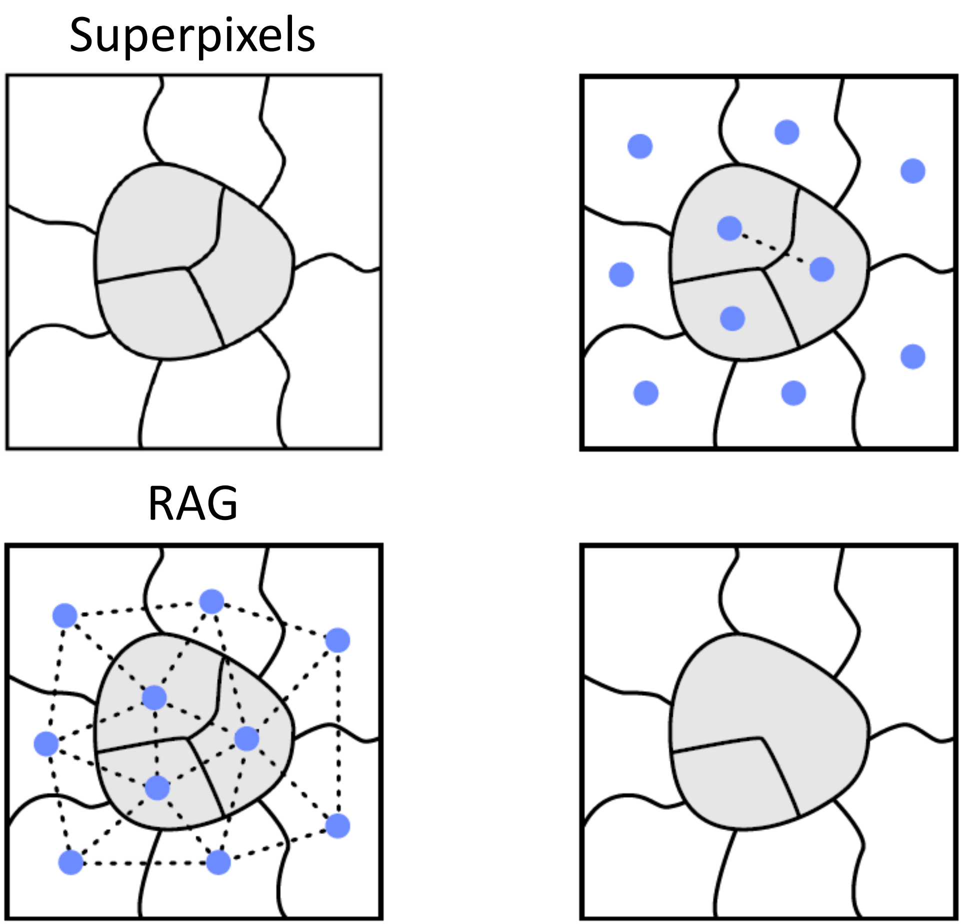
Superpixel applications

- Superpixels are a multipurpose intermediate image representation
- More compact representation for algorithms with high time complexity (600x800 pixels -> 200 superpixels)
- Common application: object segmentation
  - Oversegment image
  - Combine superpixels to find objects

Superpixel merging

- Region Adjacency Graph (RAG)
  - Vertices = image regions (pixels or superpixels)
  - Edge weights = difference between regions
- To merge superpixels:
  - Identify edges below a threshold and re-label superpixels connected by these edges as one region
  - Or iteratively:
    - Find lowest-weight edge, relabel connected superpixels as one region
    - Recompute RAG, repeat until a criterion is met (e.g., all edges above a threshold)

Superpixel merging



Find lowest-weight edge, merge superpixels

Recompute RAG, repeat

Image: Galvão, Guimarães, & Falcão (2020)

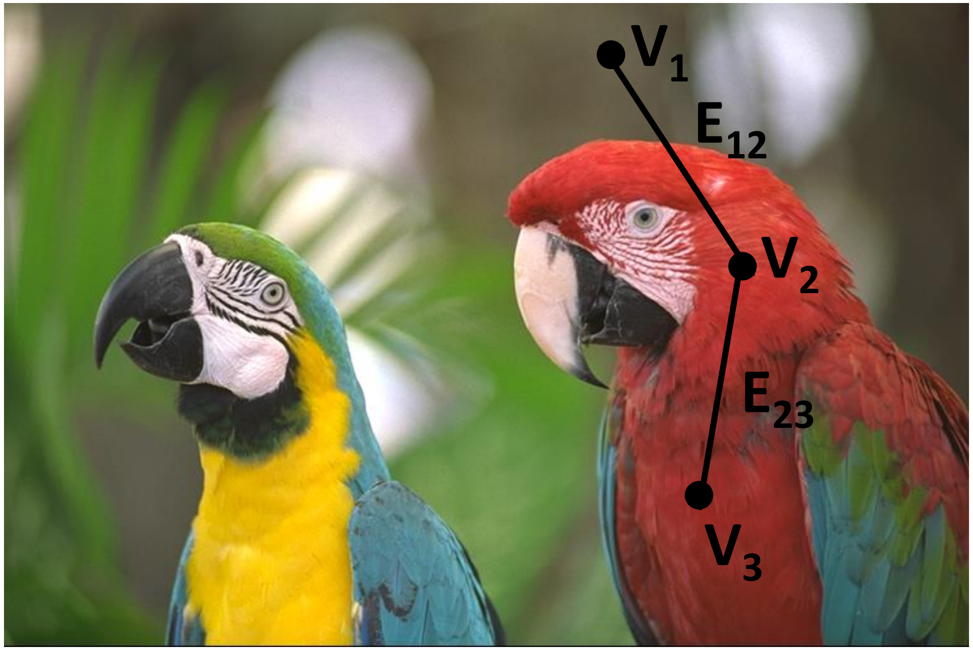
Summary

- Superpixels = regions of similar pixels, produced through oversegmentation
- Various algorithms for computing superpixels, SLIC is one common option
- Superpixels are a compact, intermediate representation used as a first step for:
  - Segmentation (especially graph-based methods)
  - Object detection/localisation
  - Video tracking

Graph-based segmentation

Images as graphs

- Represent image as a graph **G = (V,E)**
  - Vertices = image regions (pixels or superpixels)
  - Edge weights = similarity between regions



$E_{12} < E_{23}$

Graph cuts

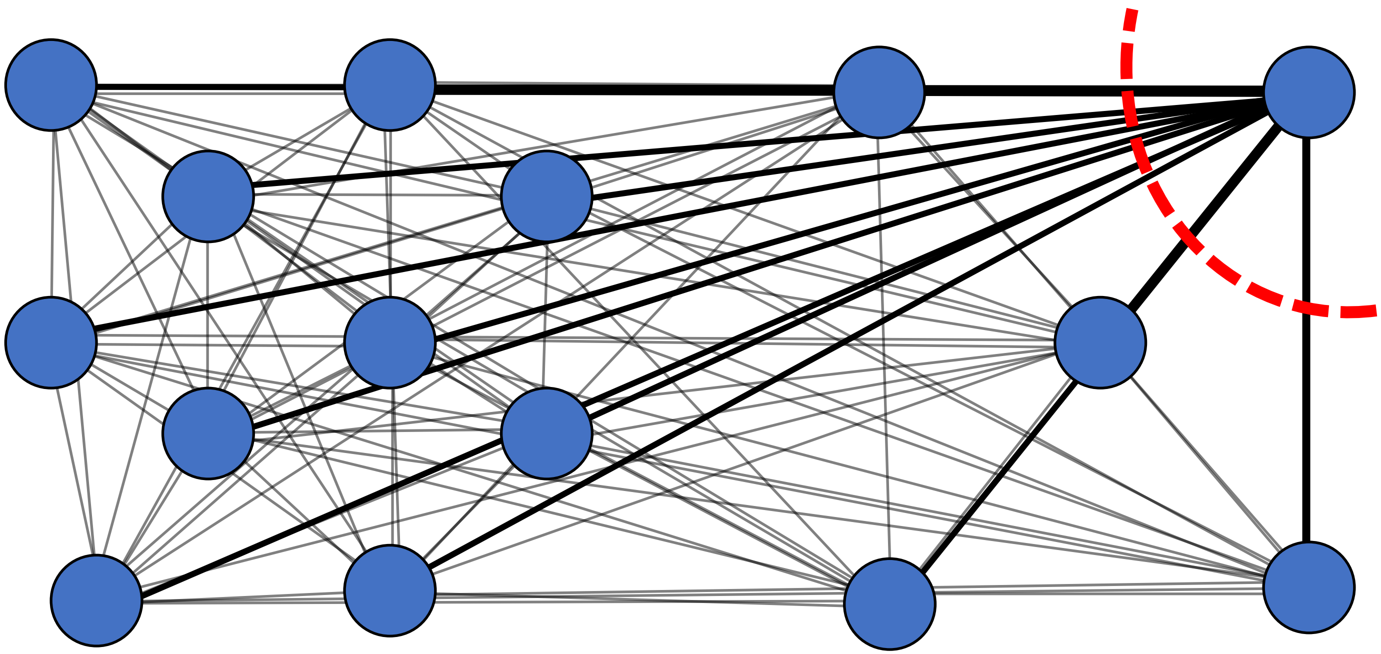
- Consider image as a fully-connected graph
- Partition graph into disjoint sets A,B to maximize total edge weight = remove low-weight edges between dissimilar regions
- Minimize value of cut:

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

Weight of edge connecting u and v

Graph cuts

- Not ideal for image segmentation – tends to create small, isolated sets



Edge weight = 1/distance

Normalised cuts

- Instead of minimizing cut value, minimize cut value as a fraction of total edge connections in entire graph (normalised cut)
- Normalised cut (Shi & Malik, 2000):

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= \frac{\sum_{u \in A, v \in B} w(u,v)}{\sum_{u \in A, t \in V} w(u,t)} + \frac{\sum_{u \in A, v \in B} w(u,v)}{\sum_{v \in B, t \in V} w(v,t)}$$

Normalized cuts results

Image: D. Hoiem

GrabCut

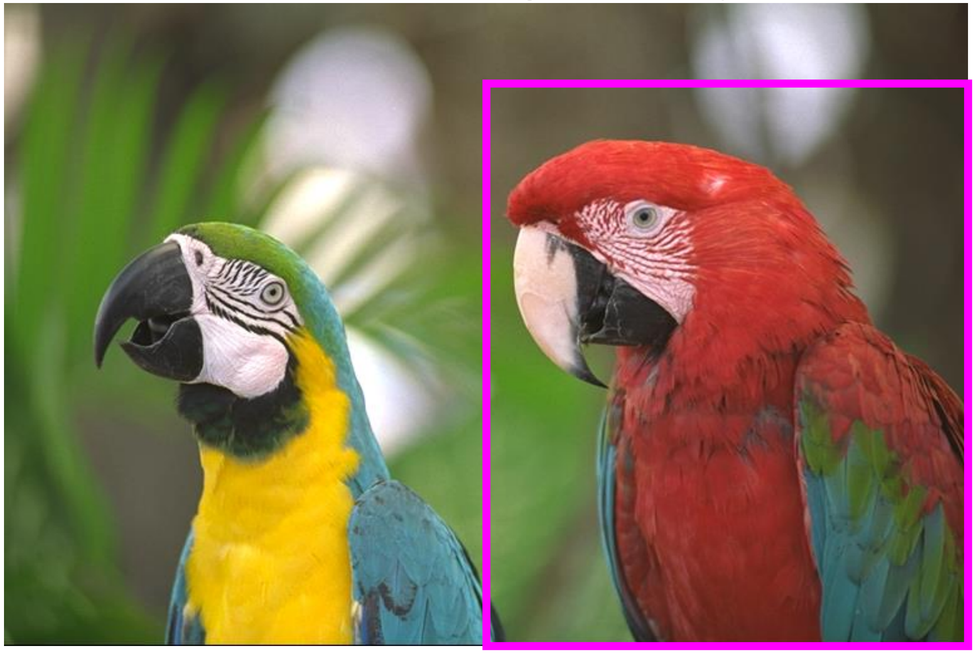
- Segments image pixels into just two classes: foreground (object) and background
- Uses colour clustering + graph cuts to find optimal classification of pixels into each class

Rother, Kolmogorov, & Blake (2004)

GrabCut algorithm

- Requires user to initialise algorithm with a bounding box

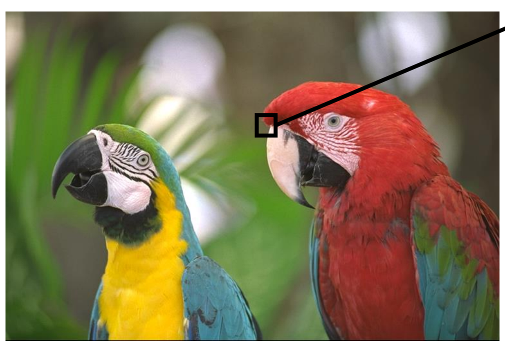
Outside box = background pixels

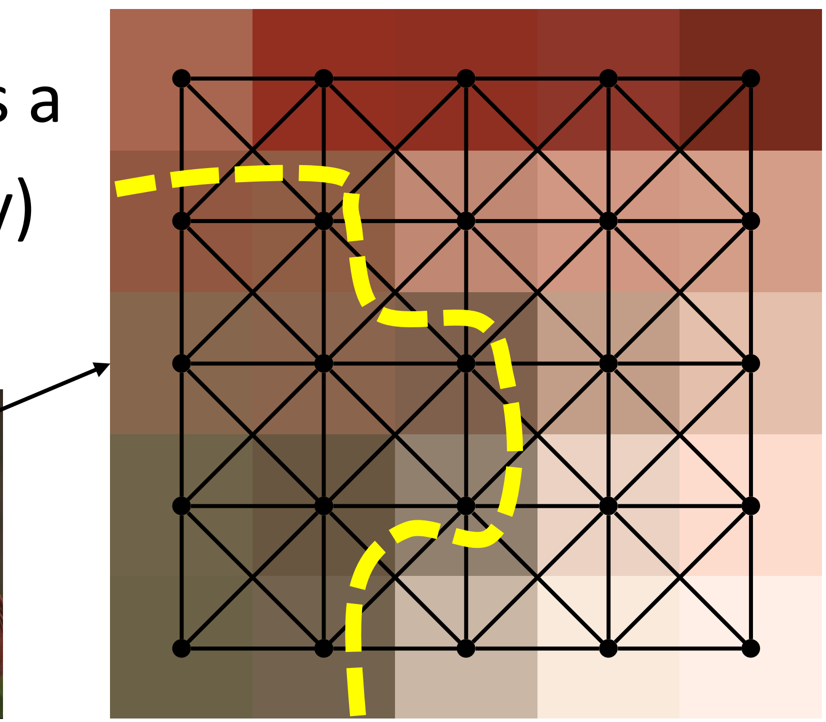


Inside box = treat as foreground pixels initially

GrabCut algorithm

- For each class (foreground, background), represent distribution of pixel colour as a Gaussian mixture model (GMM)
- Represent image pixels as a graph (8-way connectivity)





GrabCut algorithm

- Denote the pixel graph as **G** and the GMM as θ
- α indicates label of each pixel (foreground or background)
- Iterate until convergence:
  - Find graph cut (label assignment) to minimize
    $$E(\alpha, \theta, \mathbf{G}) = U(\alpha, \theta, \mathbf{G}) + \gamma V(\alpha, \mathbf{G})$$
    -log likelihood of cluster assignments in GMM

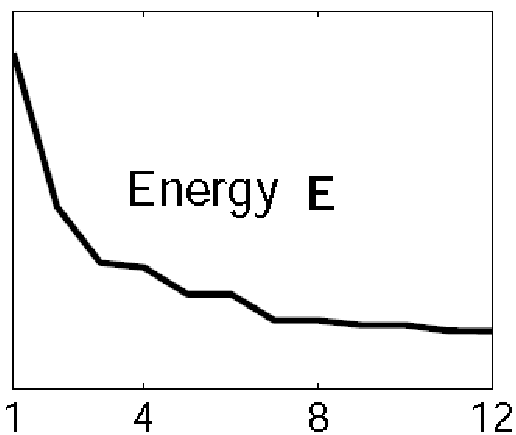
    Weighting parameter

    Smoothness penalty based on colour similarity, applied to neighbouring pixels with different labels in α
  - Recompute GMM for new label assignment

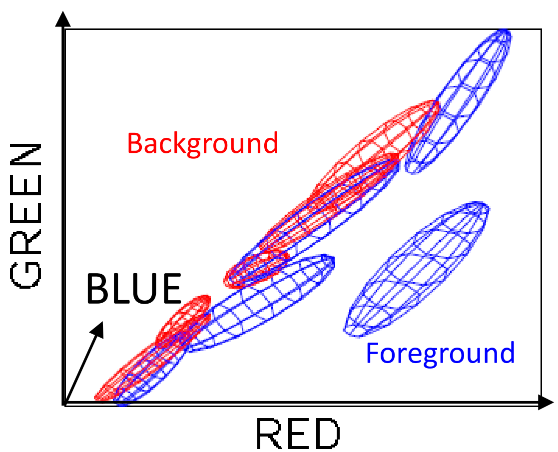
GrabCut example

Initialisation

E over iterations



Initial GMM



Final GMM

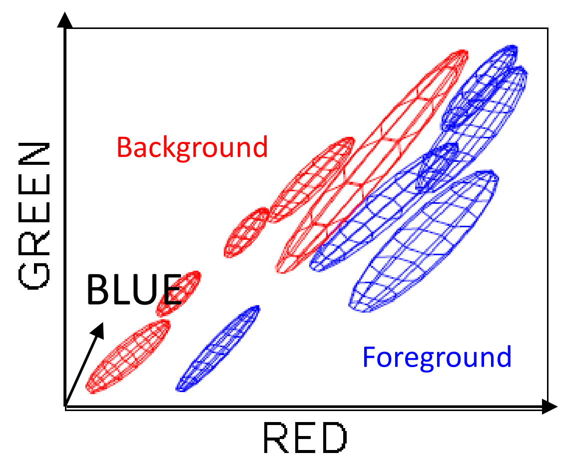
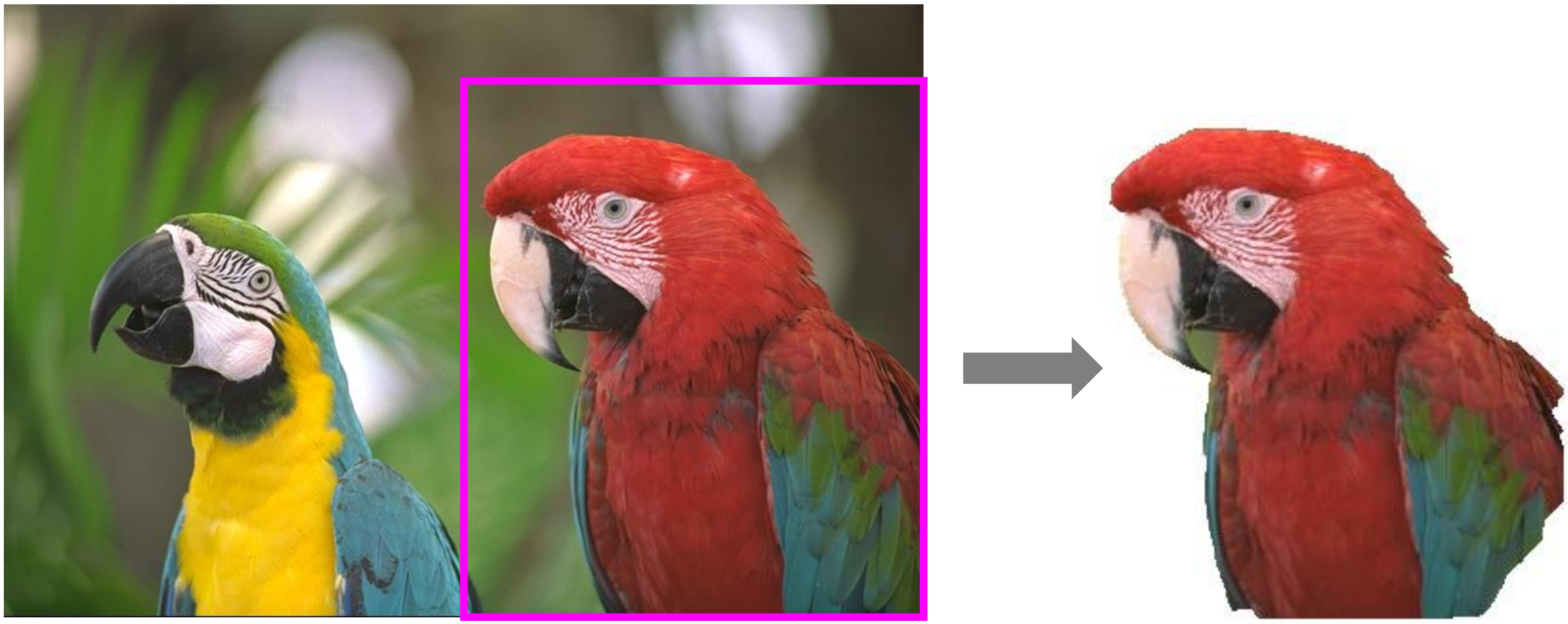


Image: Rother, Kolmogorov, & Blake (2004)

GrabCut result



https://chocopoule.github.io/grabcutweb/

Summary

- Graph-based methods represent an image as a graph (of pixels or superpixels)
- Segmentation removes edges to break graph into subgraphs, generally trying to optimize:
  - Similarity within connected region
  - Dissimilarity across disconnected regions
  - Smoothness/connectivity of connected regions
- Normalized cuts – segment into multiple regions
- GrabCut – segment into foreground/background

Summary

- Various ways to approach image segmentation, but many methods use some combination of pixel clustering and graph analysis
- The methods discussed so far do segmentation but not semantic segmentation (regions with no labels)
- How to get labels?
  - Unlabelled regions can be input to an object classification method
  - Or, segmentation and classification can be done simultaneously (next lecture)