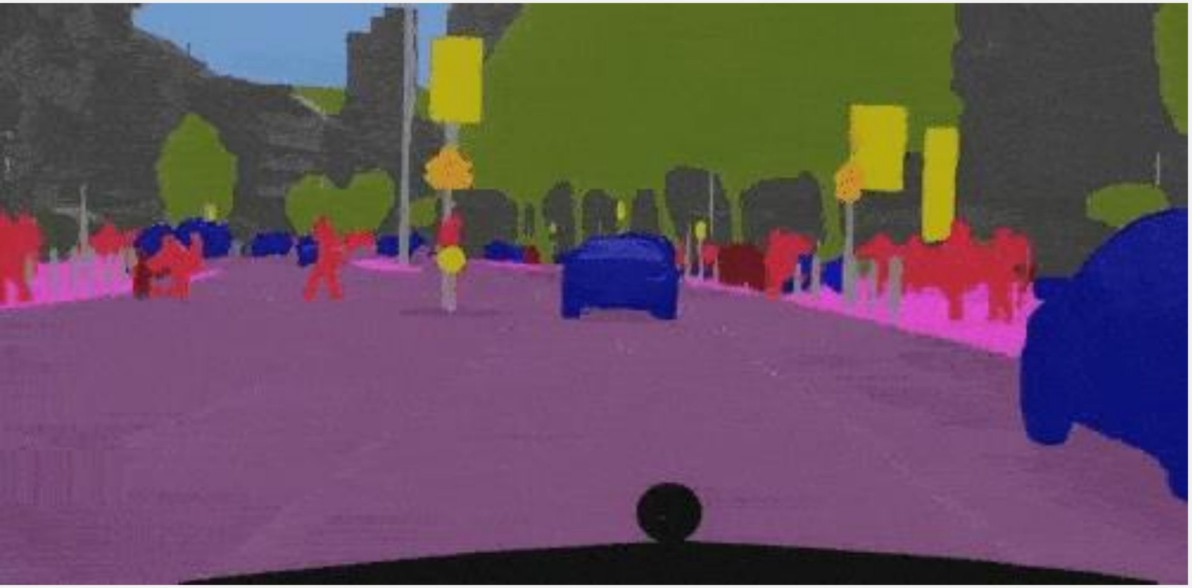
Department of Computer Science
University of Bristol

COMS30030 - Image Processing and Computer Vision



Lecture 05

Segmentation - The Basics

Majid Mirmehdi
majid@cs.bris.ac.uk

Examples of Image Segmentation

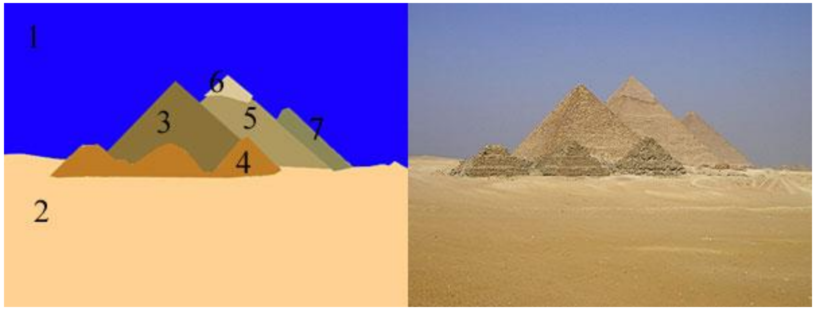
- **Image Segmentation ...**

  ... is the process of spatial subsectioning of a (digital) image into multiple <u>partitions of pixels</u> (i.e. segments or regions) according to given criteria.

Example: segmentation of an image into locally coherent regions

Motivation: Why Segment Images?

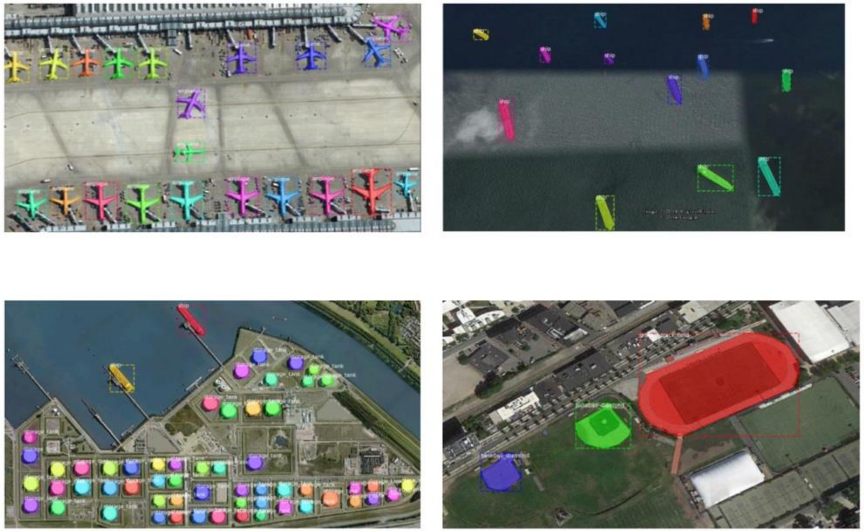
- **Image Simplification**
  - an image may contain millions of pixels but only a few regions

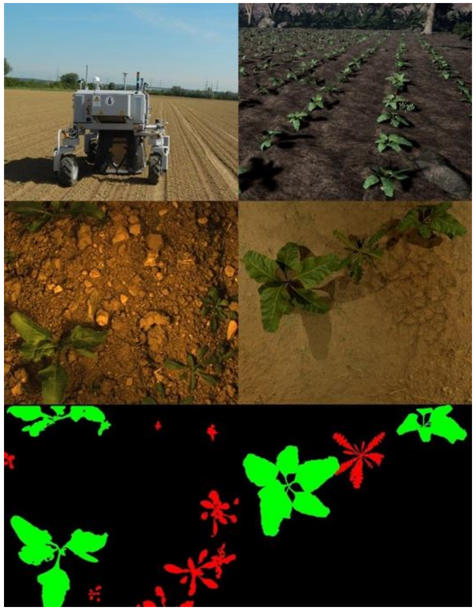


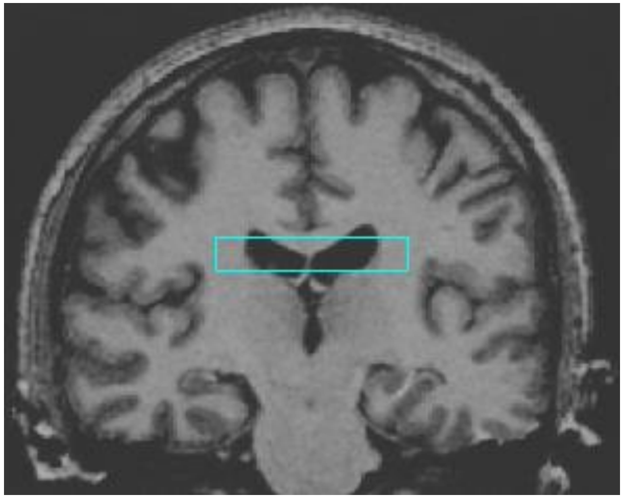
- **Higher-level Object Description**
  - regions tend to belong to the same class of object
  - regions may provide object properties (e.g. shape, colour, ...)

- **Input for Content Classifiers**
  - region descriptions can be input data for higher level classifiers, e.g. Bayesian Classifiers or Neural Networks.

Why Segment Images?







Examples from https://medium.com/cogitotech and Alberto Pretto

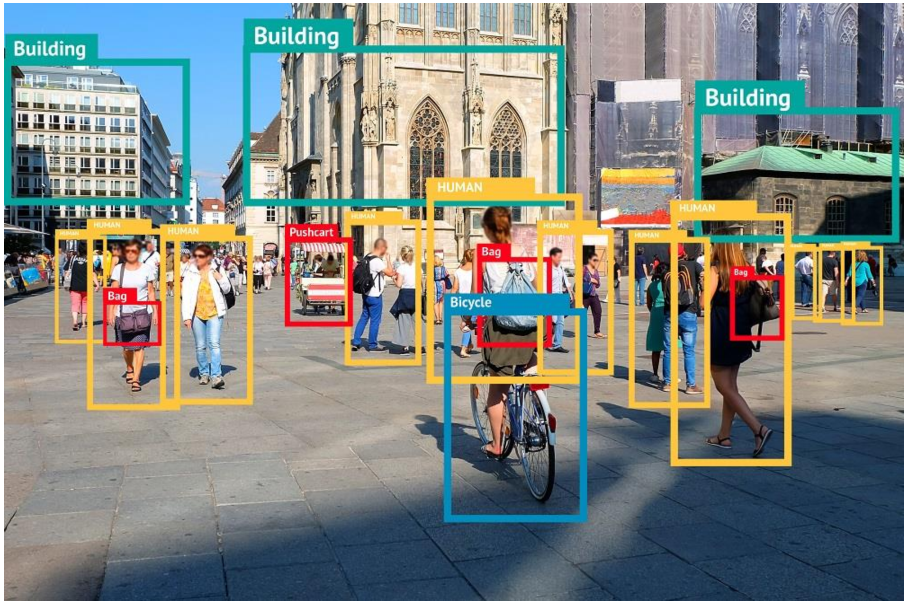
Grouping Pixels

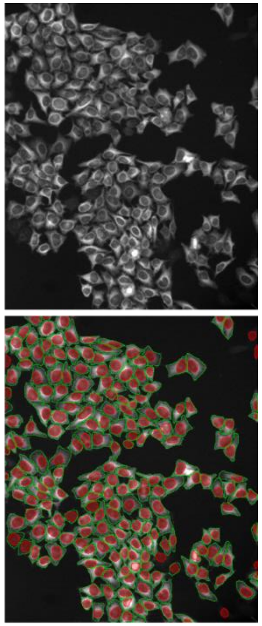
Goals:

- Gather pixels/features that belong together
- Obtain an intermediate representation that compactly describes key image (video) parts

Top-down vs. bottom-up segmentation

- Top-down: pixels belong together because they are from the same object
- Bottom-up: pixels belong together because they look similar



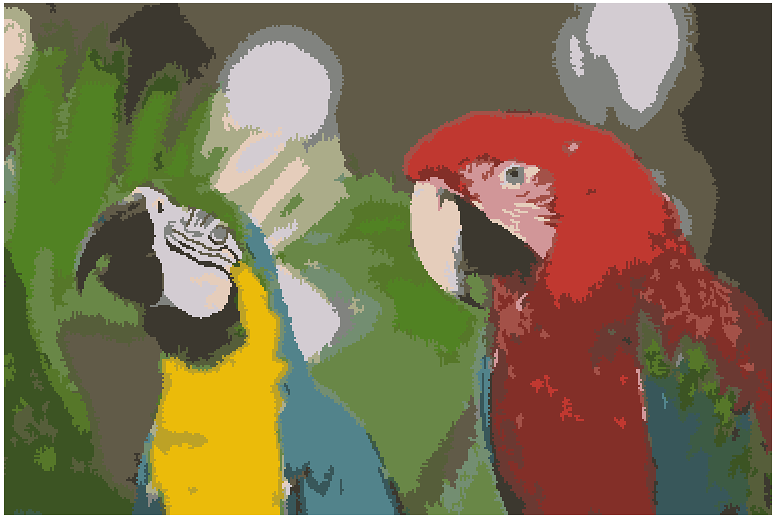


Hard to measure success: what is interesting depends on the application.

Example of Over-Segmentation

Original image

Over-segmentation

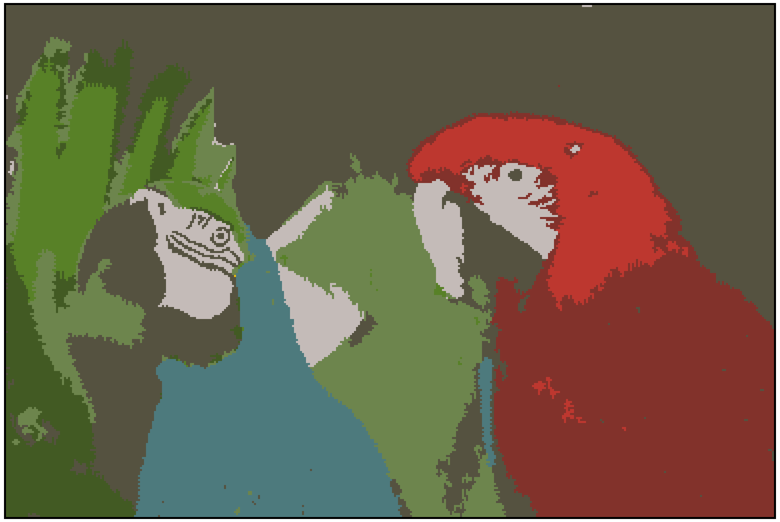


Over-segmentation: pixels belonging to the same region [object] are classified as belonging to different regions [objects]

Example of Under-Segmentation

Original image

Under-segmentation

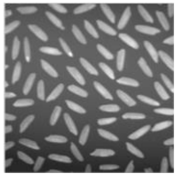


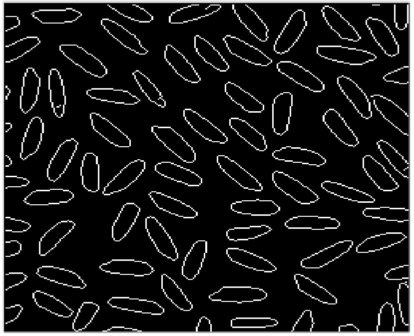
Under-segmentation: pixels belonging to different regions [objects] are classified as belonging to the same region [object]

So many segmentation methods…

Thresholding Methods

- pixels are categorized based on intensity
- only useful when sufficient contrast exists





Edge-based Methods

- region boundaries are constructed from edgemaps

Region-based Methods

- region growing from seed pixels
- region splitting and merging for efficient spatial encoding

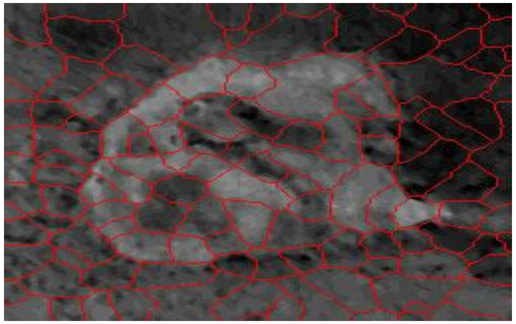
So many segmentation methods…

Clustering and Statistical Methods

- global, often histogram based image partitioning, e.g. *K*-means, Gaussian Mixture Model

Topographic Methods **(out of scope in this unit)**

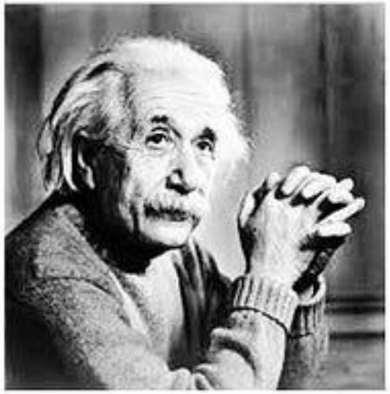
- stepwise simplifications that take spatially wider (topographical) image configurations into account e.g. watershed transform, variational based methods



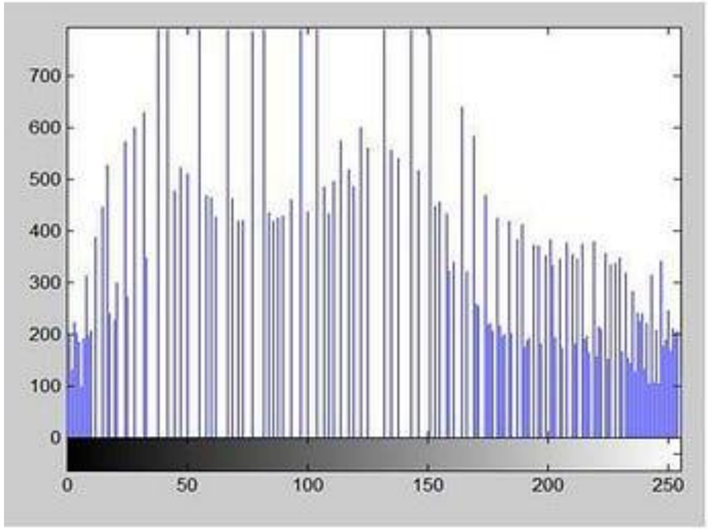
…and many more…

Image Histogram

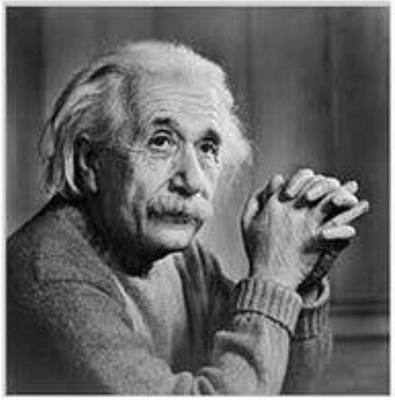
Brighter



Histogram



Darker



Histogram

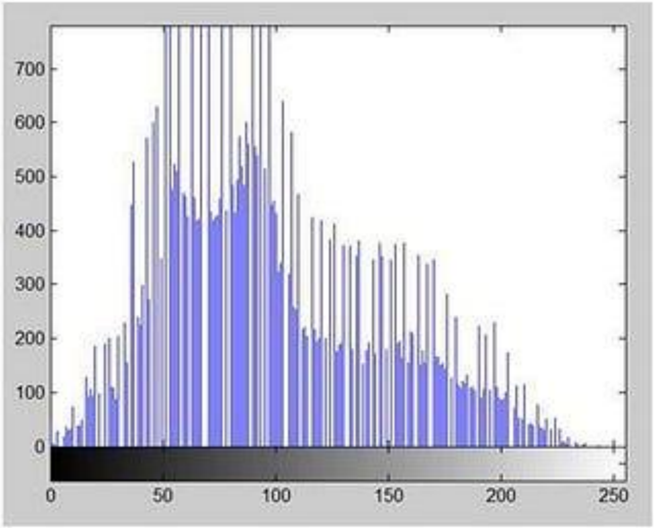
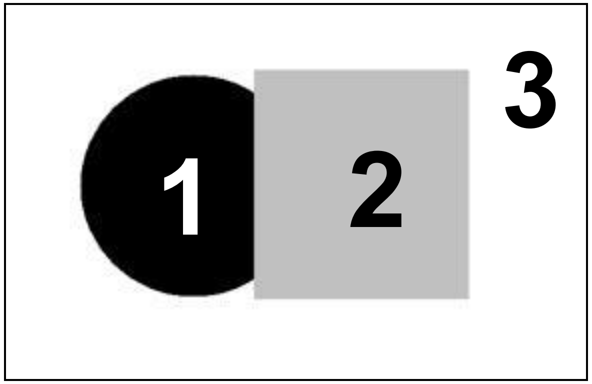
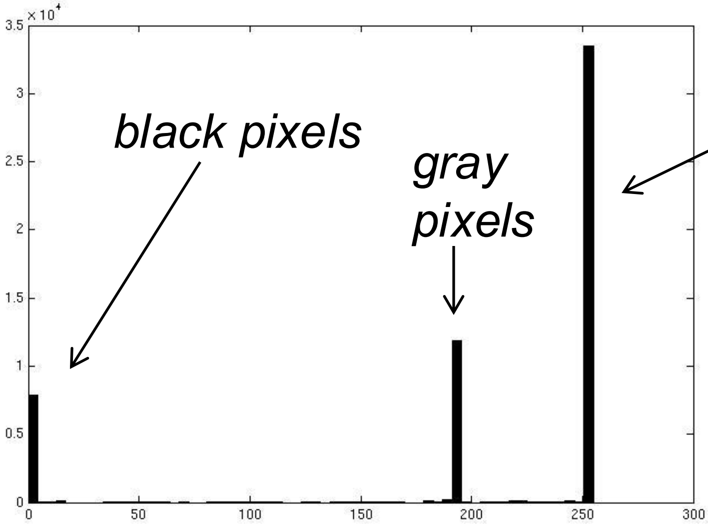


Image segmentation: toy example



input image

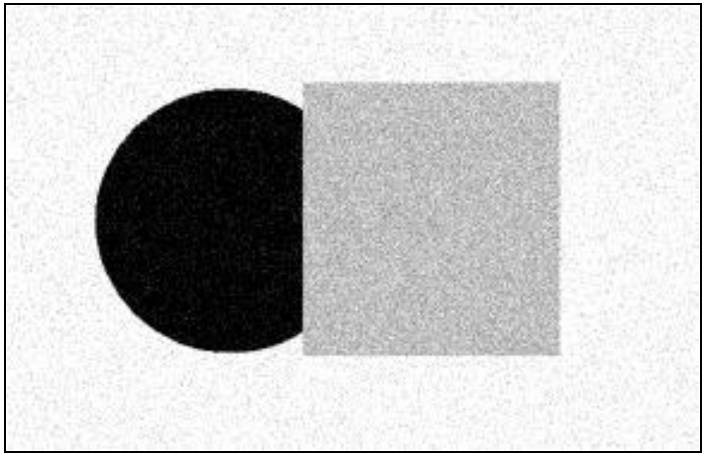


- The intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.

Slide credit: Kristen Grauman

Simple thresholding not enough!

What if the image isn't quite so simple?



input image

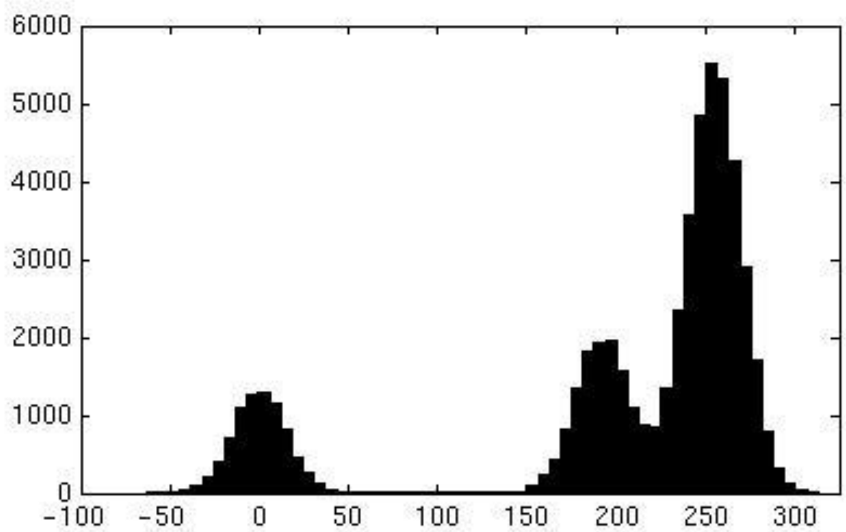
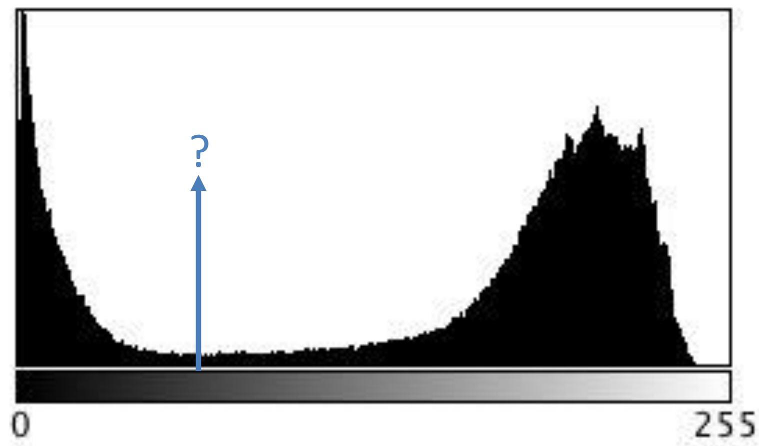
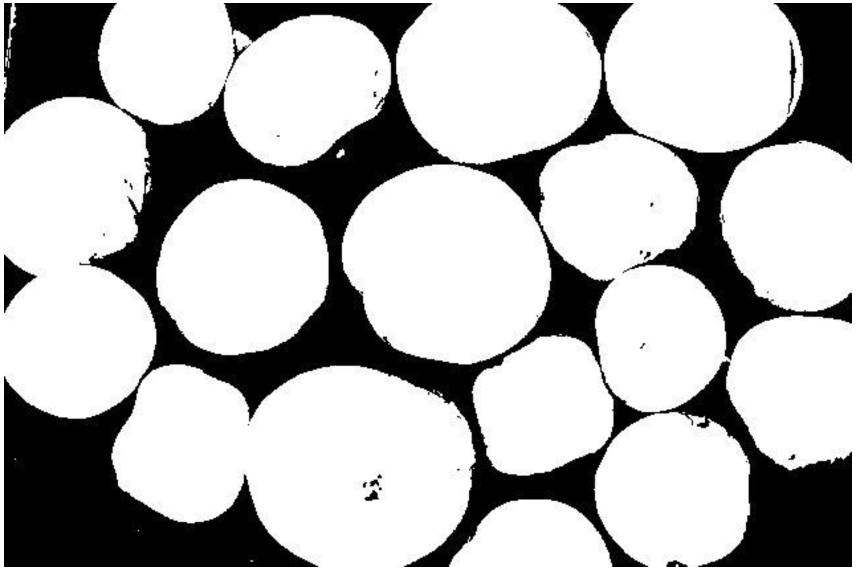


Image Segmentation

Perfect segmentation is difficult to achieve:

- a pixel may straddle the "real" boundary of objects such that it partially belongs to two or more objects
- effects of noise, non-uniform illumination, occlusions etc. give rise to the problem of *over-segmentation* and *under-segmentation*





Images from craftofcoding.wordpress.com

Thresholding Example

- If the image contains a dark object on a light background
  - choose a threshold value, *T*
  - for each pixel
    - if the brightness at that pixel is less than *T*, it is a pixel of interest
    - otherwise it is part of the background

T = 128

- The value of the threshold is very important
  - if too high → background pixels classified as foreground
  - If too low → foreground pixels classified as background

T = 96

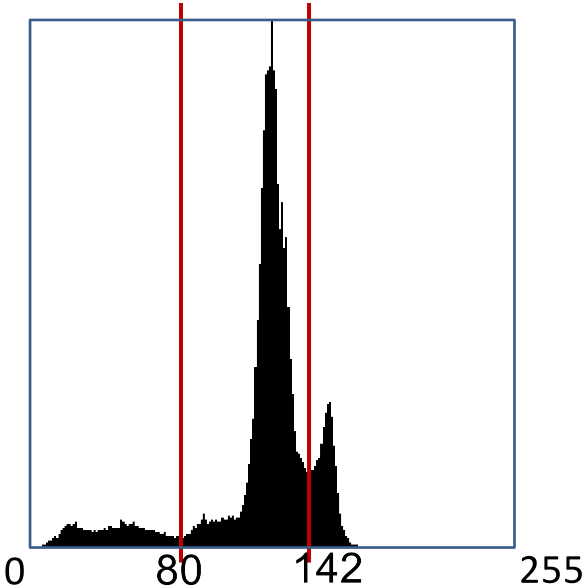
T = 64

Using Histograms to Stipulate Regions

Maybe apply multiple thresholds?

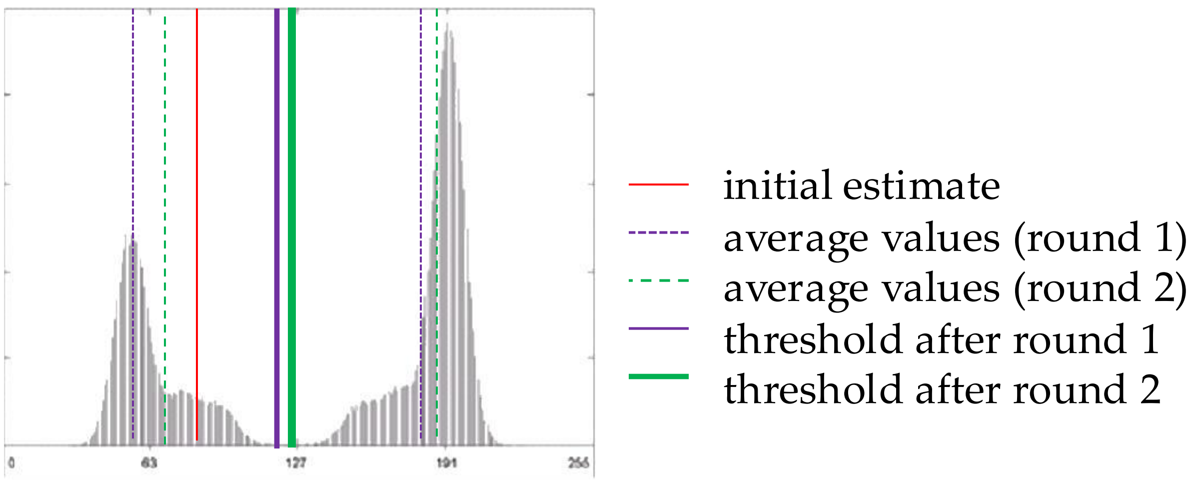
The seal image shows three regions

- one below $T_1 = 80$
- one above $T_2 = 142$
- one between the two thresholds



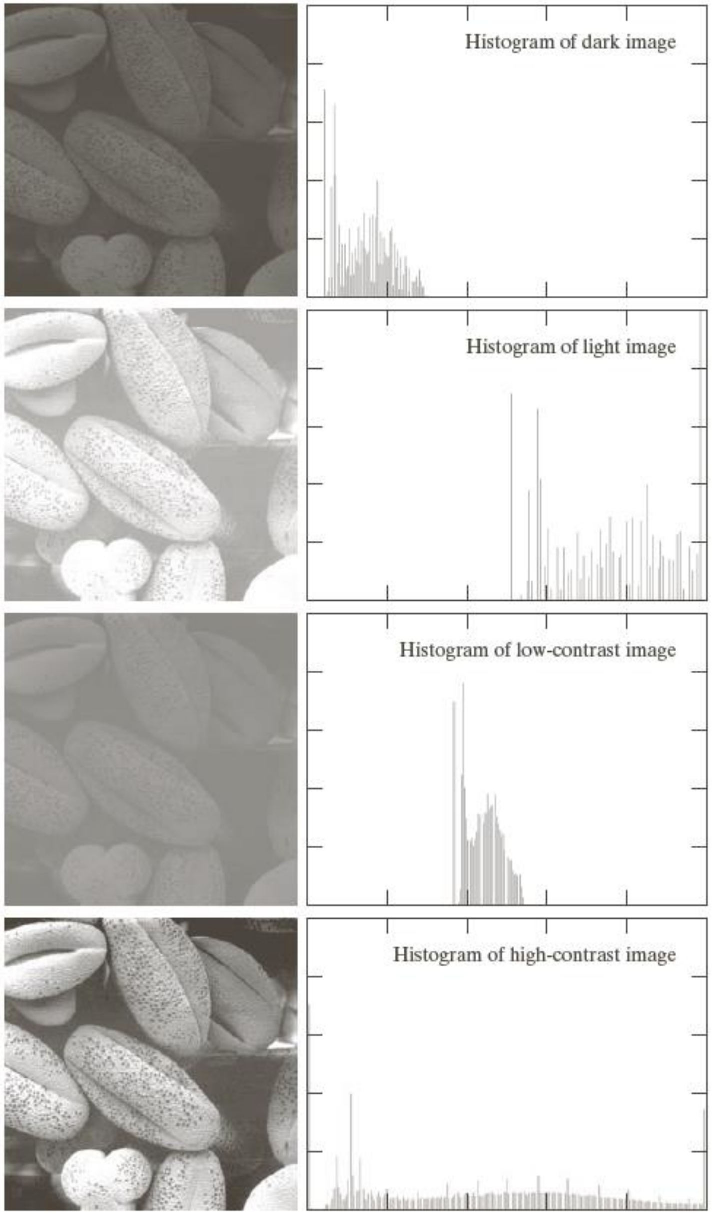
Iterative Threshold Selection Algorithm

1. Select an initial estimate for the threshold T
2. Segment the image using T.
   This will produce two groups of pixels: G_1 consisting of all pixels with grey levels $>T$ and G_2 consisting of pixels with grey values $\leq T$.
3. Compute the average grey level values m_1 and m_2 for the pixels in regions G_1 and G_2.
4. Compute a new threshold value: $T = (m_1 + m_2)/2$
5. Repeat steps *(2.)* through *(4.)* until convergence



Thresholding for Segmentation

Not always a good solution!



Four problematic image types: dark, light, low contrast, and high contrast.

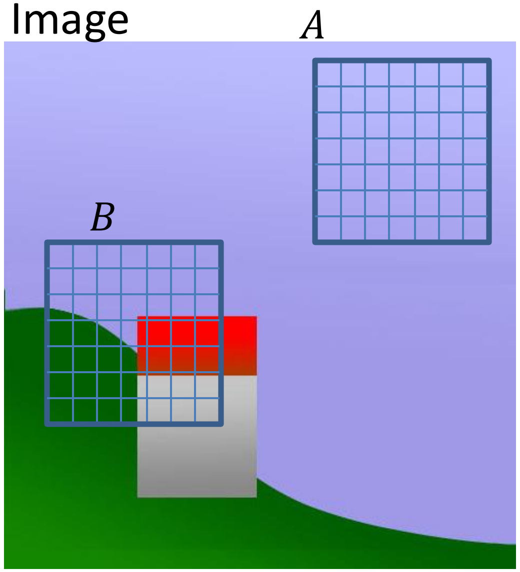
Split & Merge Segmentation – Divide & Conquer

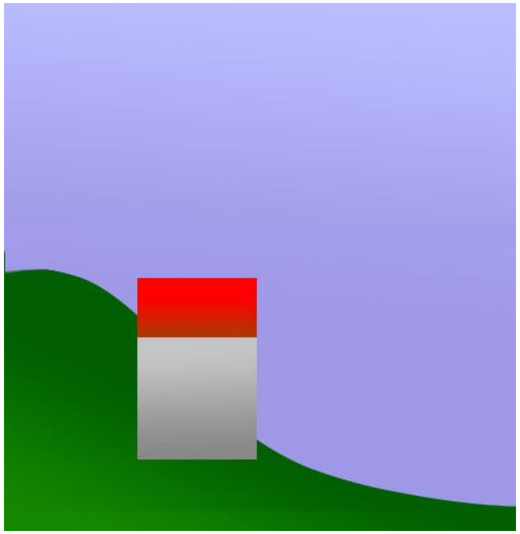
<u>**Homogenity function *H***</u>

$$H(Region\ A) = 1$$ (homogeneous)

$$H(Region\ B) = 0$$ (inhomogeneous)

Image

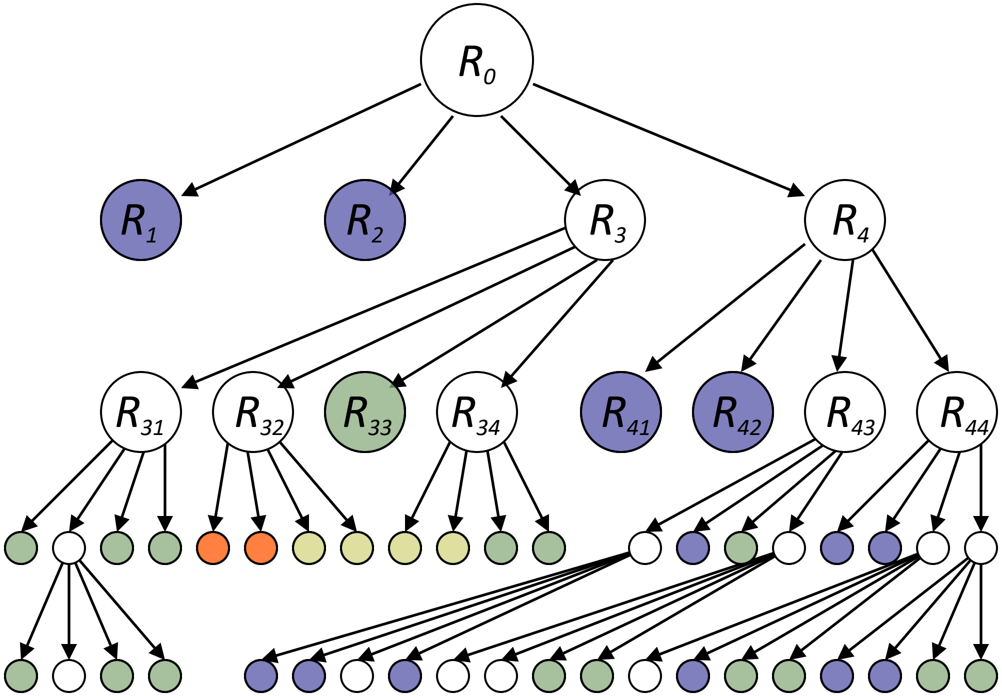




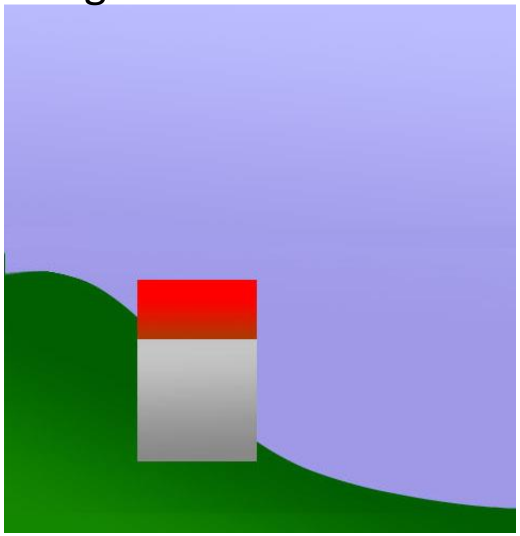
Segmentation Result

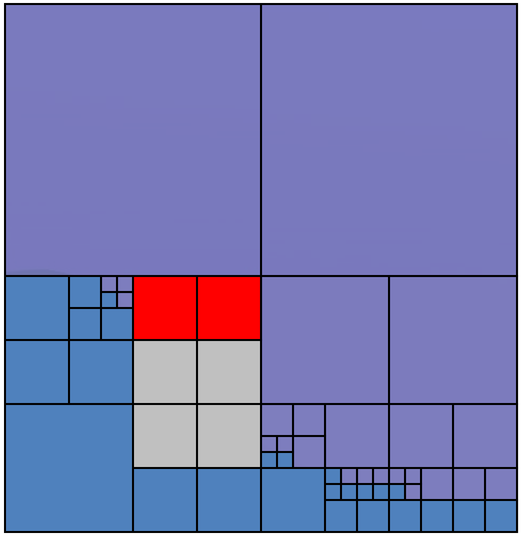
Split & Merge Segmentation – Divide & Conquer

1. Start with R_0 that represents the entire image
2. If $H(R_i) = 0$ (inhomogeneous) then {split area into 4 blocks (quadtree splitting) and process each area with step (2.)}
3. Merge all subregions that pairwise satisfy $H(R_i \cup R_j) = 1$ (homogenous)



Image





Segmentation Result

Split & Merge – Summary

Conceptual Summary:

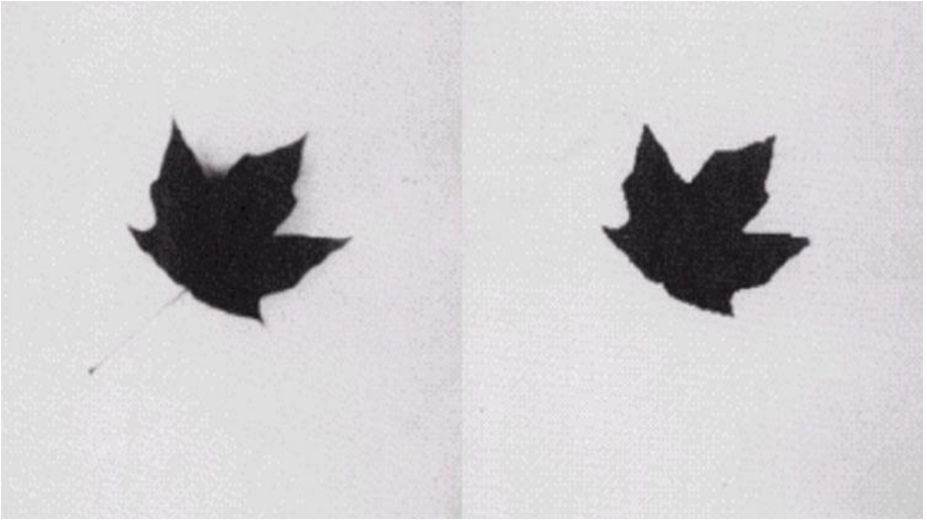
- Iteratively decompose an image into regions of a maximally sized selected shape (e.g. rectangle) that do not satisfy a homogeneity condition. (split step)
- Then merge regions that together satisfy a homogeneity condition. (merge step)

Some Comments:

- Using quadtrees, the results of split and merge tend to be *blocky*.
- Can have an adaptive homogeneity condition that, for instance, changes depending on the region size.

Example *H*

- $H(R_i)=1$ if at least 80% of the pixels in R_i have the property $|z_j - m_i| < 2\sigma_i$ where z_j is the grey level of the j^{th} pixel in R_i , m_i is the mean grey level of the region and σ_i is the standard deviation of the grey levels in R_i
- If $H(R_i)=1$ then set all the pixels in R_i to value m_i

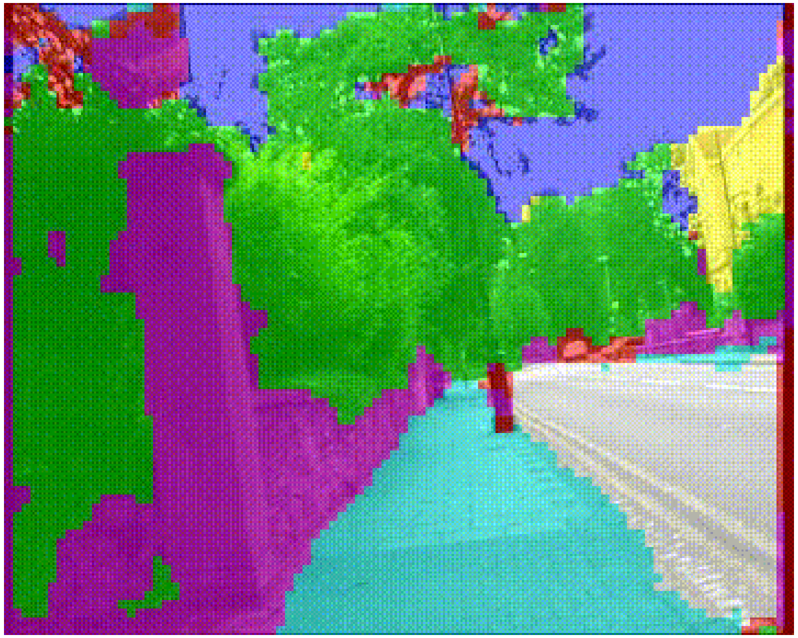


Original Result

Split & Merge – Bristol Video Scene Segmentation

Original Video

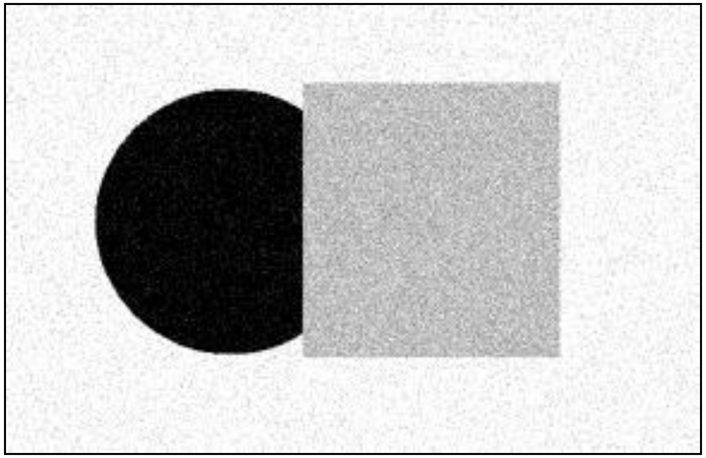
Segmentation Result



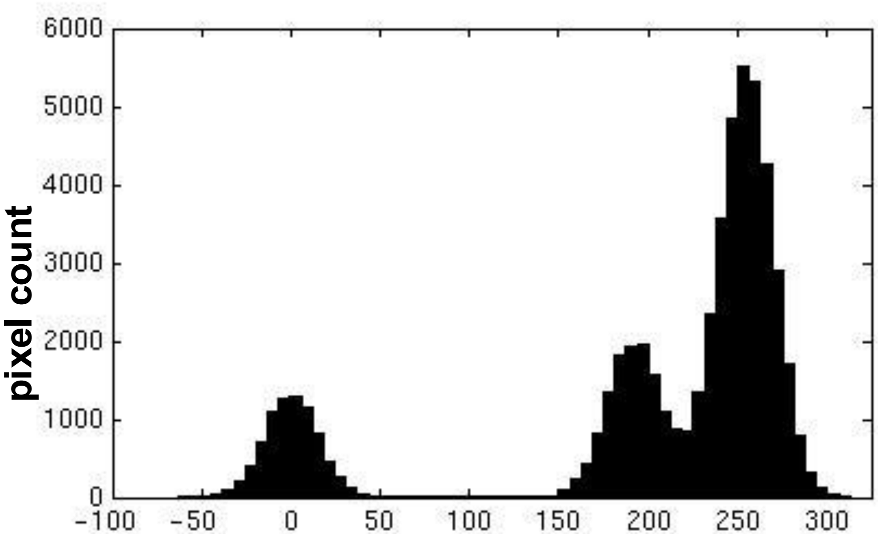
- Images are segmented using a Split-And-Merge technique. (Note the blocky nature of the regions!)
- Regions are then labelled by a Neural Network to associate the segments with semantics (colouration).
- This project dates back to around 27 years ago!

Image segmentation: toy example

What if the image isn't quite so simple?



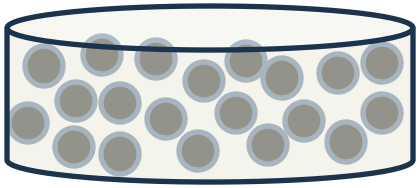
input image



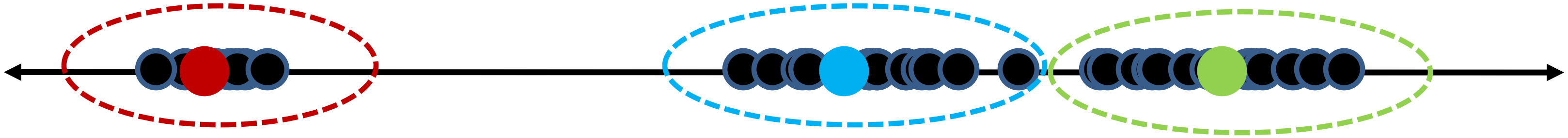
intensity

One answer is: use **clustering**…

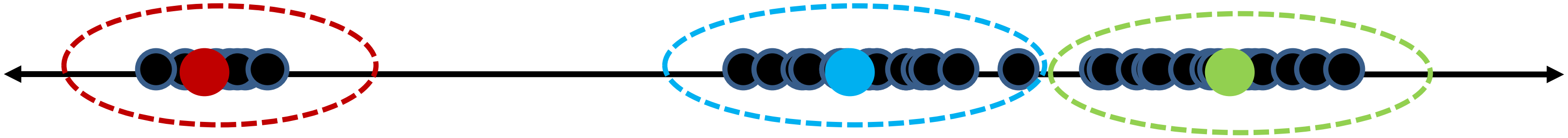
Clustering dilemma



If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



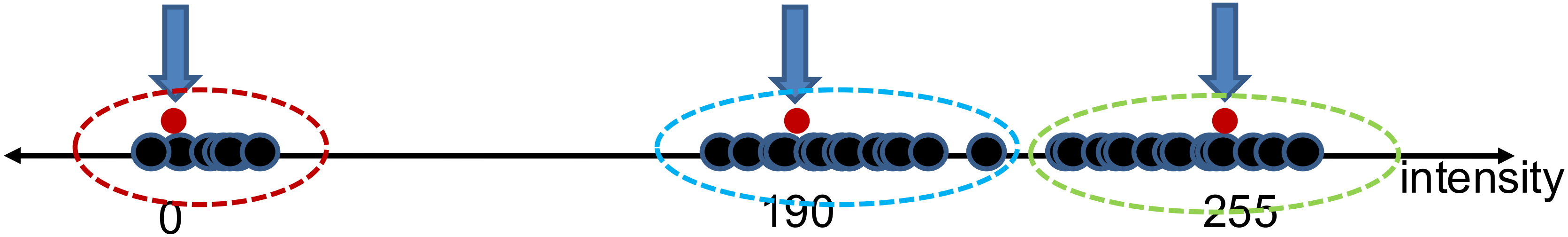
If we knew the **group memberships**, we could get the centers by computing the mean per group.

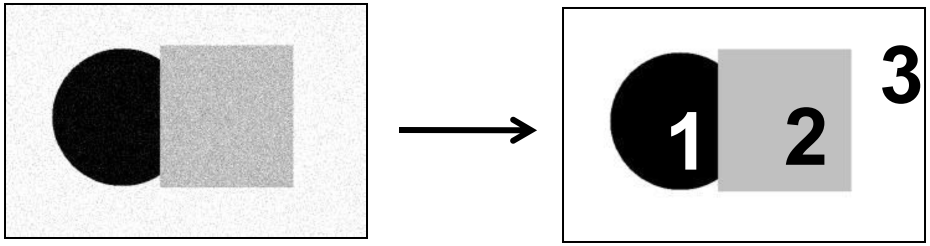


A "chicken and egg" problem!

Slide by Kristen Grauman

Image segmentation: toy example





- Goal: choose three "centres" as the representative intensities, and label every pixel according to which of these centres it is nearest to.
- Best cluster centres are those that minimize SSD between all points and their nearest cluster centre $\boldsymbol{\mu}_j$

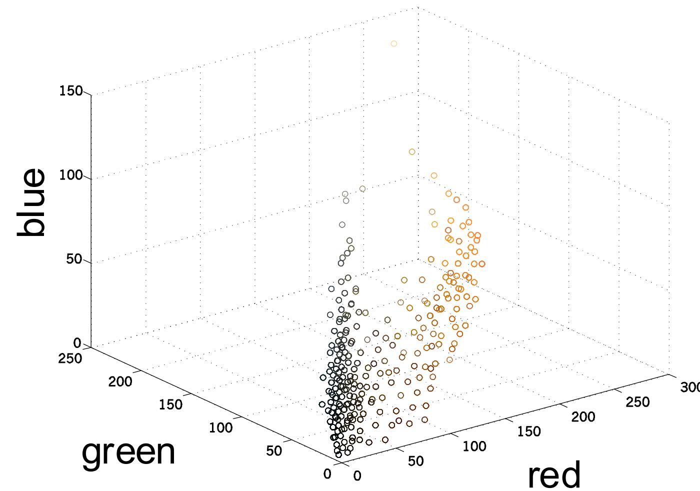
$$\Theta(clusters, data) = \sum_{j \in clusters} \left[\sum_{i \in j^{th} cluster} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 \right]$$

Slide by Kristen Grauman

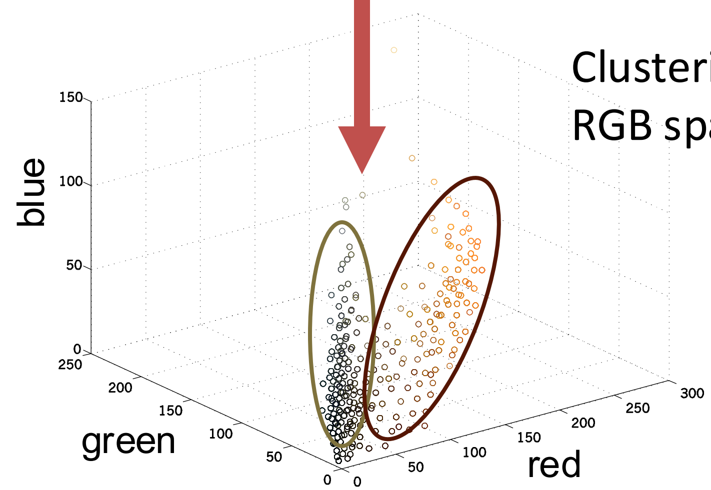
Clustering for image segmentation

map to 3D

RGB space

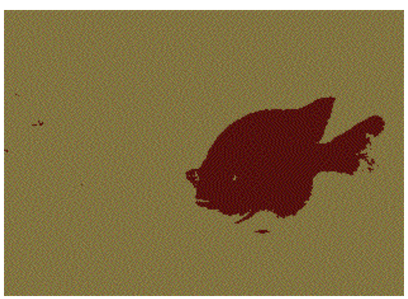


Clustering in RGB space



map back to

pixel space



K-means clustering – theoretical view

- It minimises the following objective function:

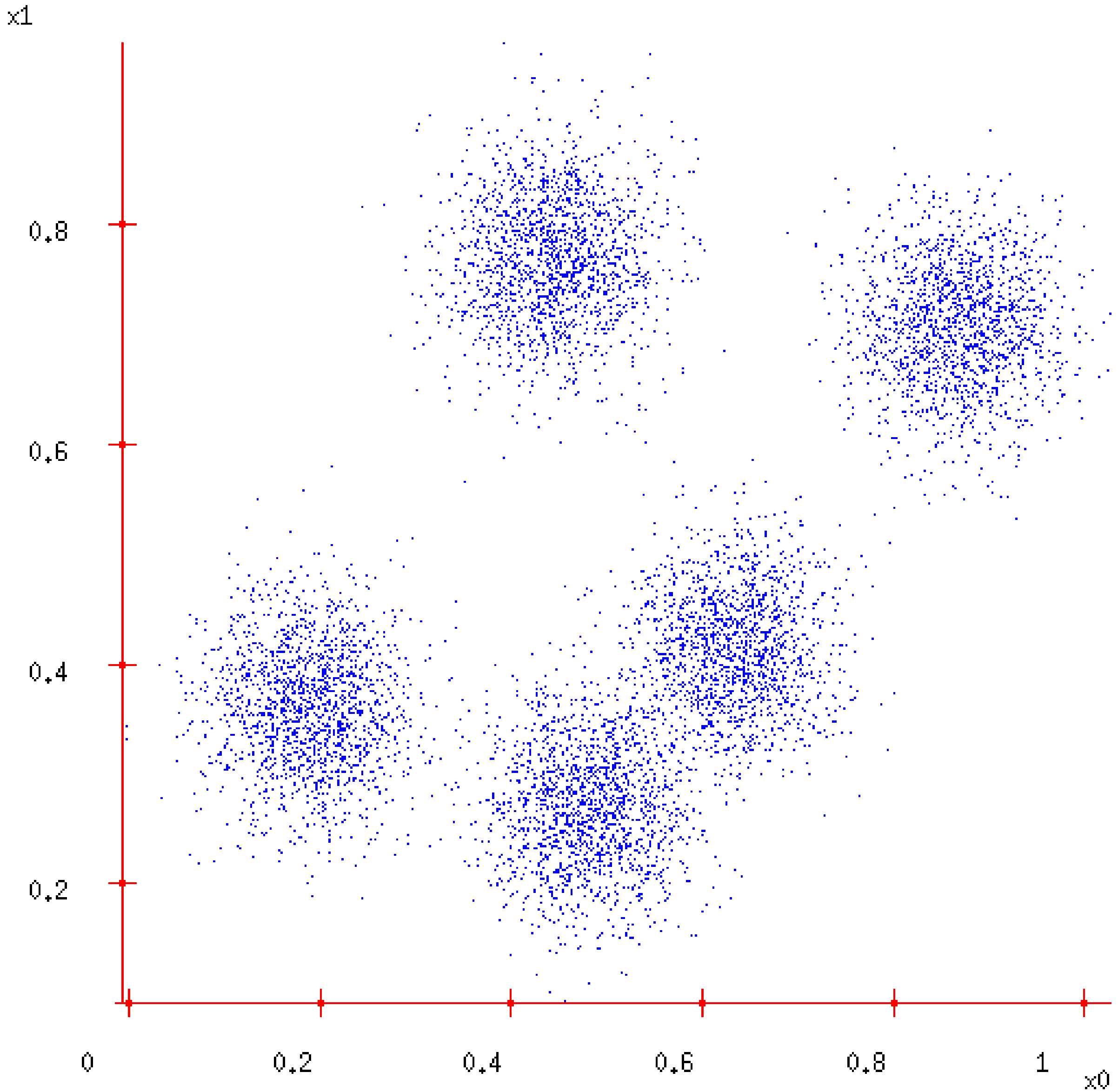
$$\Theta(clusters, data) = \sum_{j \in clusters} \left[\sum_{i \in j^{th} cluster} \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2 \right]$$

- An iterative clustering algorithm
  - Pick *K* random points as cluster centres (means)
  - Iterate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
    - Stop when no point's assignment changes

K=4

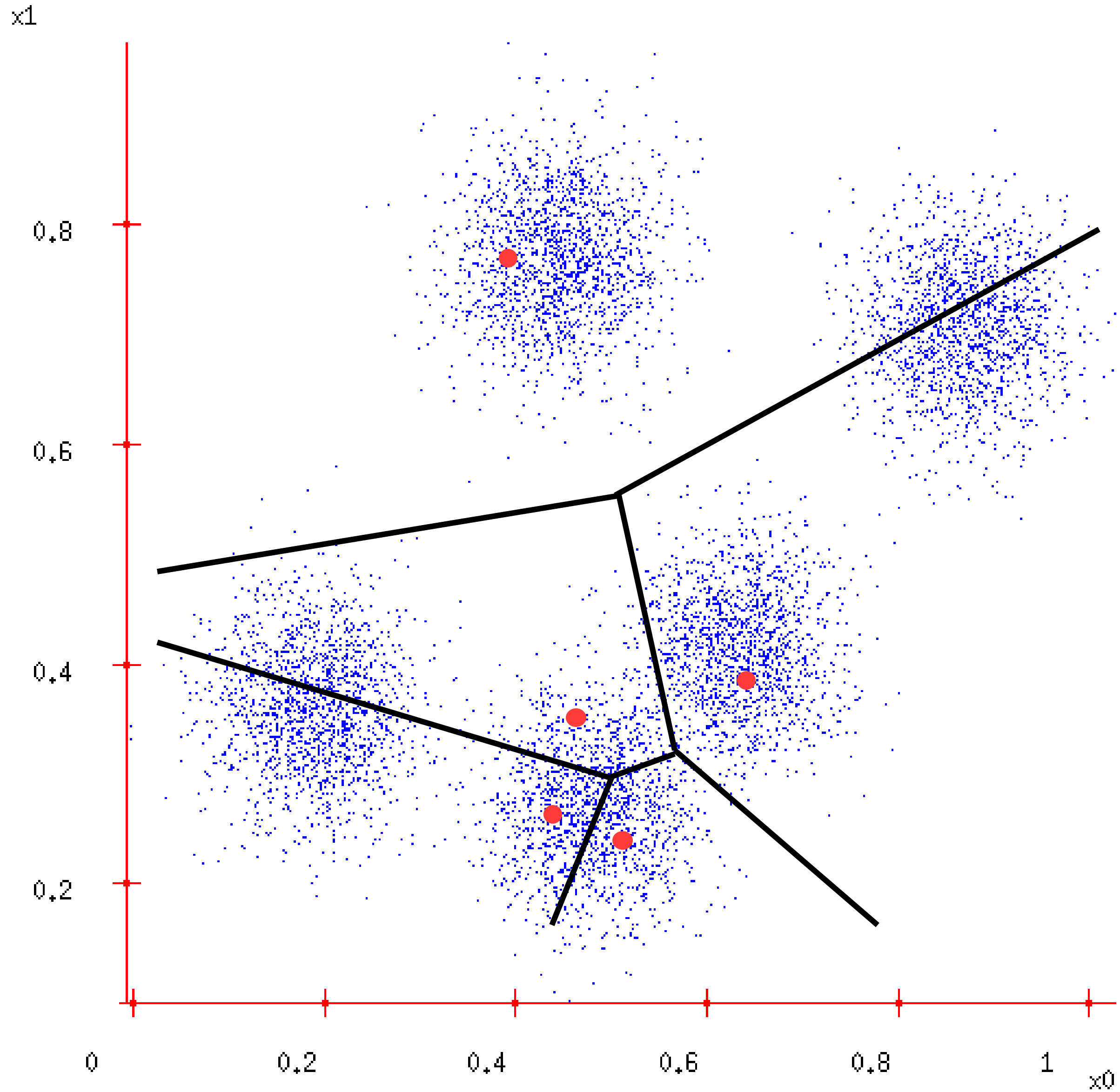
K-means clustering

1. Ask user how many clusters they'd like (e.g., K=5)



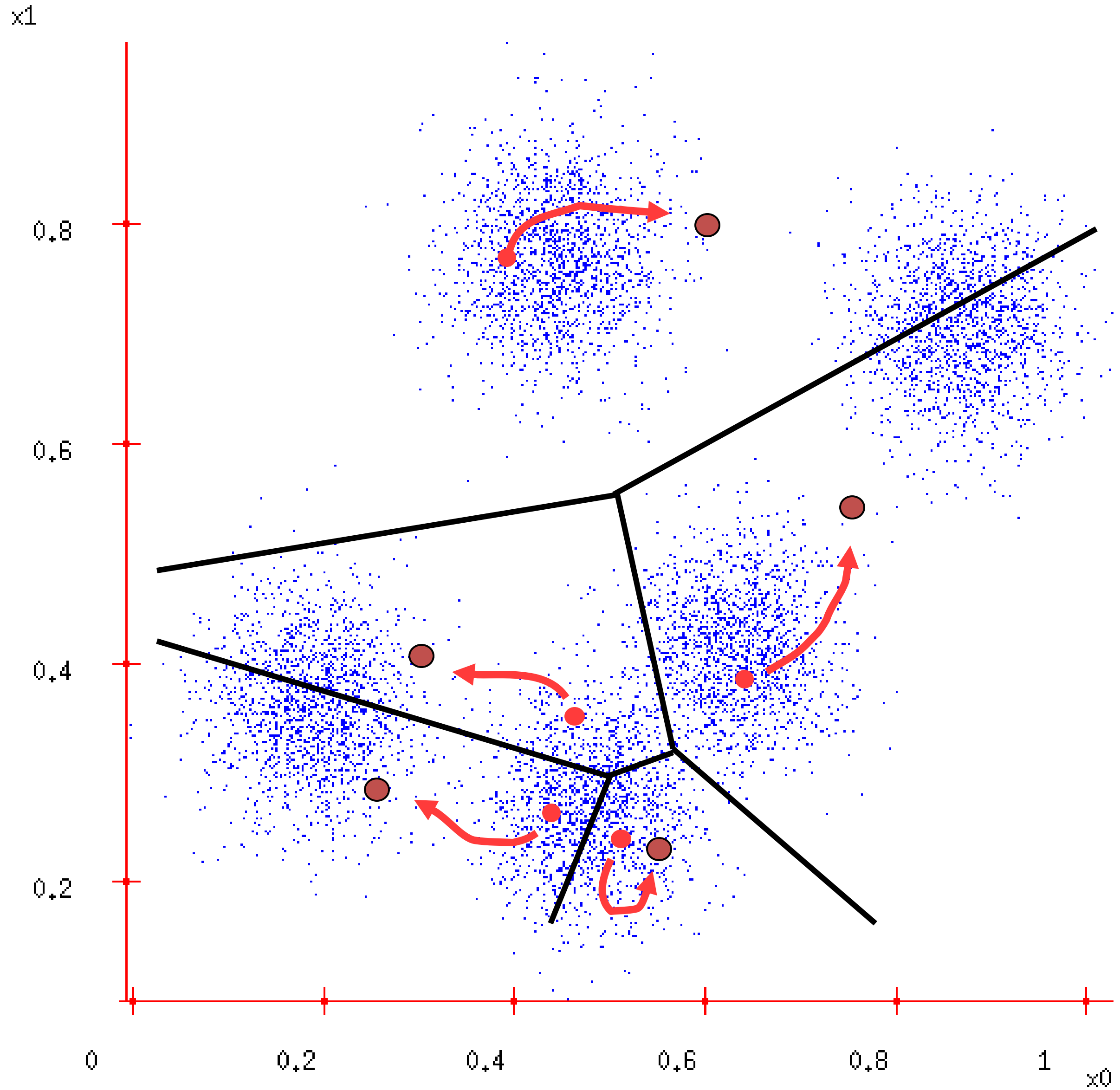
K-means clustering

1. Ask user how many clusters they'd like (e.g., K=5)
2. Randomly guess K cluster centre locations ($\boldsymbol{\mu}_1 \ \ldots \ \boldsymbol{\mu}_K$)
3. Each datapoint finds out which centre it's closest to (thus each centre "owns" a set of datapoints)



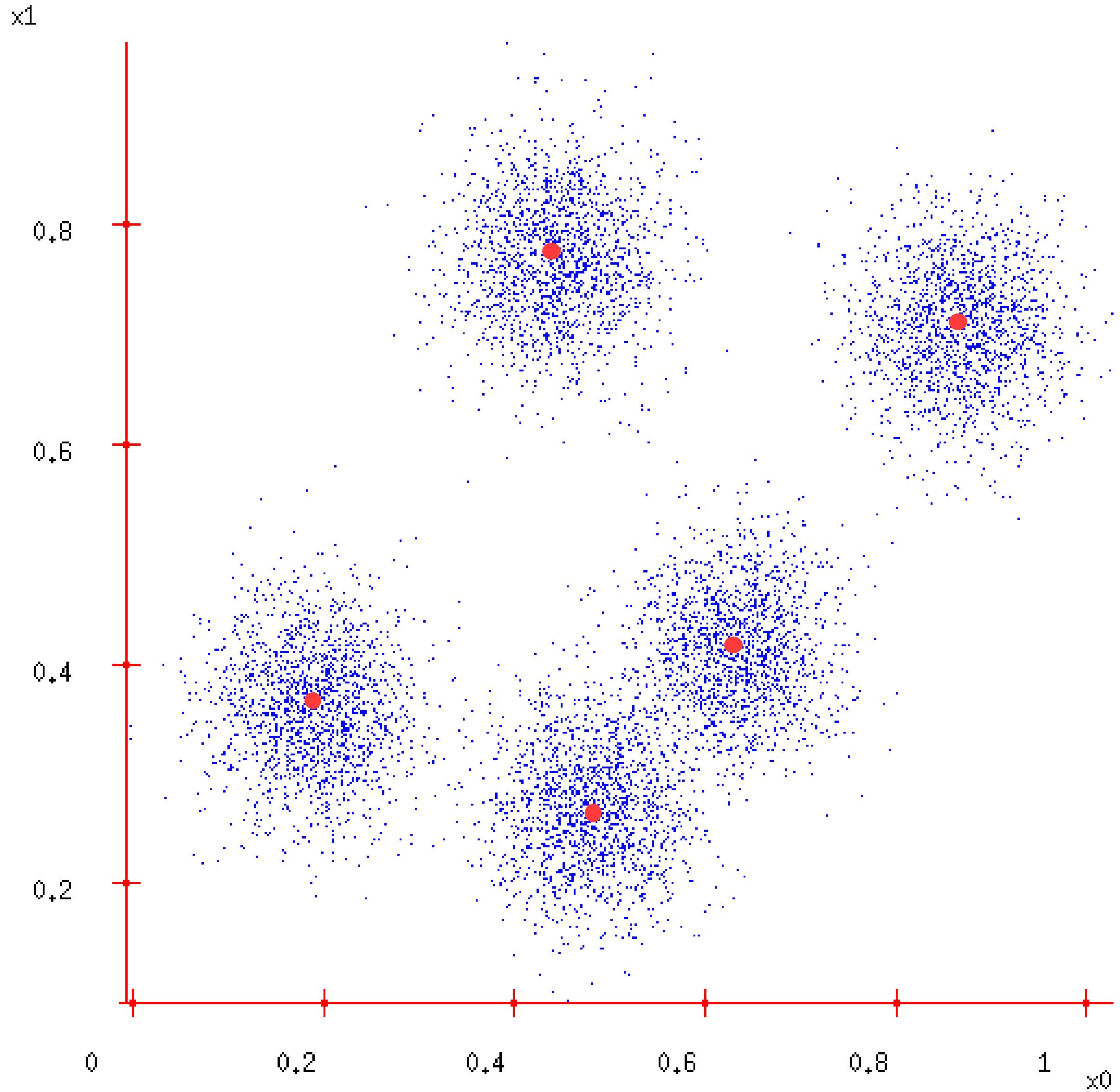
K-means clustering

1. Ask user how many clusters they'd like (e.g., K=5)
2. Randomly guess K cluster centre locations ($\boldsymbol{\mu}_1 \ \ldots \ \boldsymbol{\mu}_K$)
3. Each datapoint finds out which centre it's closest to
4. Each centre finds the centroid of the points it owns…
5. …and jumps there

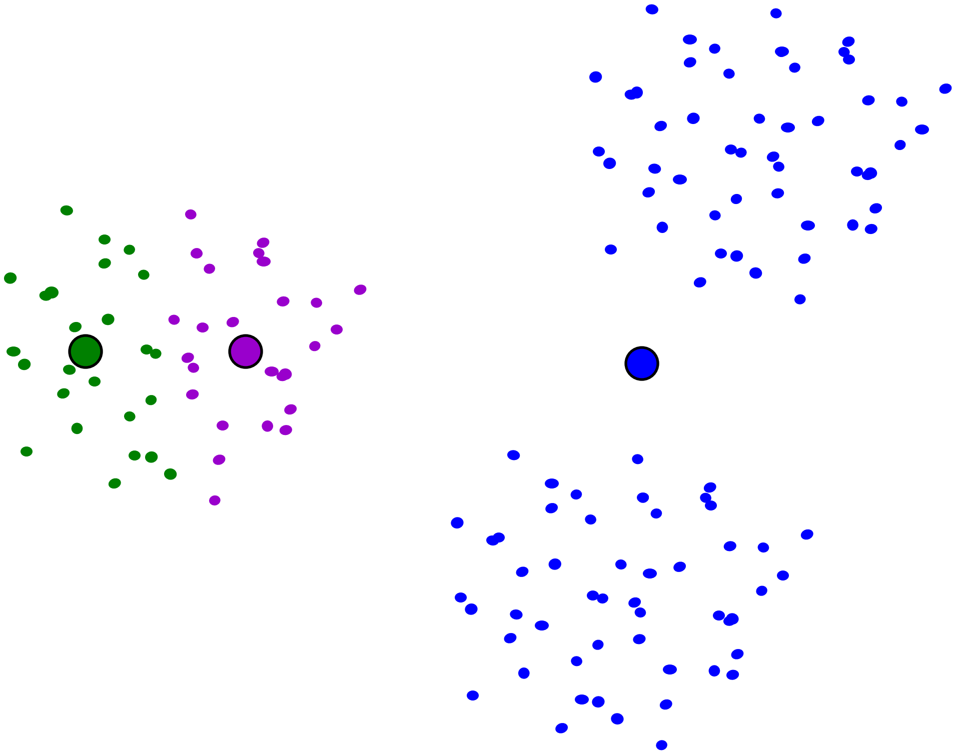


K-means clustering

1. Ask user how many clusters they'd like (e.g., K=5)
2. Randomly guess K cluster centre locations ($\boldsymbol{\mu}_1 \ \dots \ \boldsymbol{\mu}_K$)
3. Each datapoint finds out which centre it's closest to
4. Each centre finds the centroid of the points it owns…
5. …and jumps there
6. Repeat from 3 until terminated!



K-means gone wrong!



Slide by Dan Klein

Reflection on the *K*-means Algorithm

- **What does it do?**
  - *K*-means attempts to find a configuration $\mu_1 \ldots \mu_K$ that minimises within-cluster scatter: total squared distance between point x_i and centroid μ_j in j^{th} cluster:

$$\sum_i \left\| \mathbf{x}_i - \boldsymbol{\mu}_j \right\|^2$$

  - This is equivalent to maximising the between-cluster scatter (total squared distance between each cluster centroid and the global centroid of all points)

- **Does it work?**

1. The algorithm terminates.
2. It finds a local optimum from which no further improvement is possible by making local changes.
3. It does not necessarily find a global optimum.

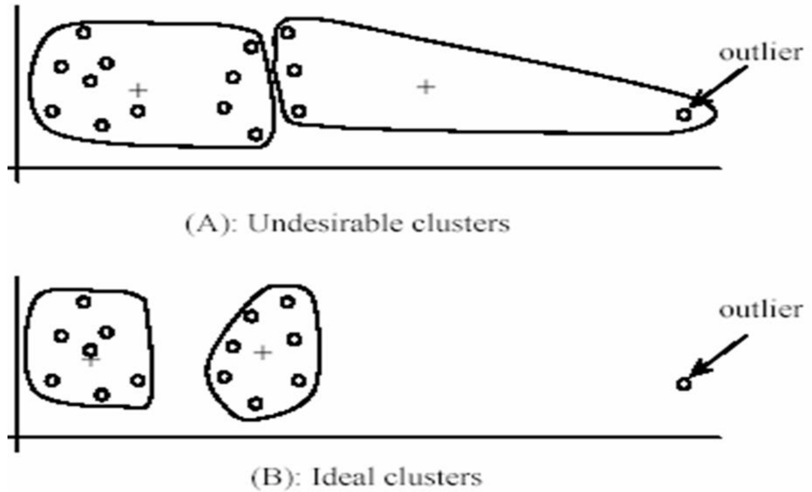
K-means Pros and Cons

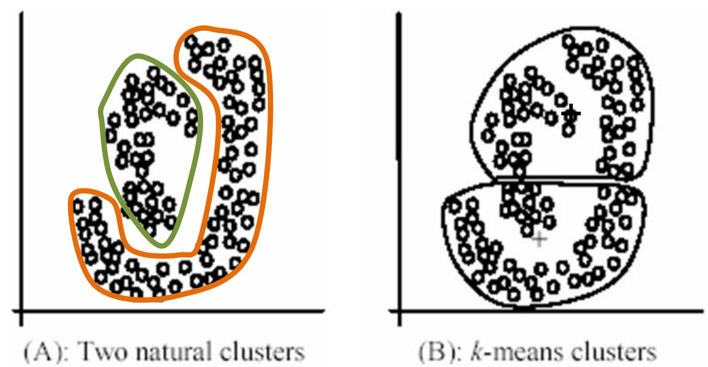
Pros

- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

- Setting *K*?
- Sensitive to initial centres
- Sensitive to outliers
- Detects spherical clusters





Slide by Dan Klein

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

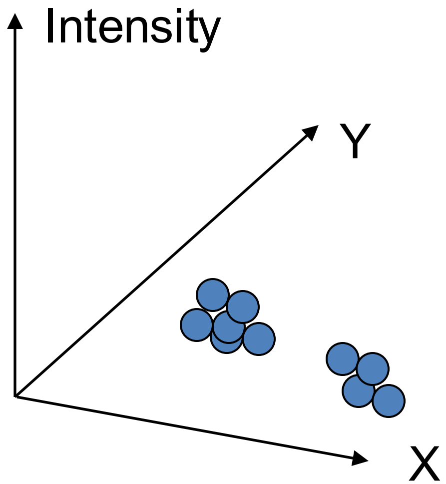
Feature space: intensity value (1D)

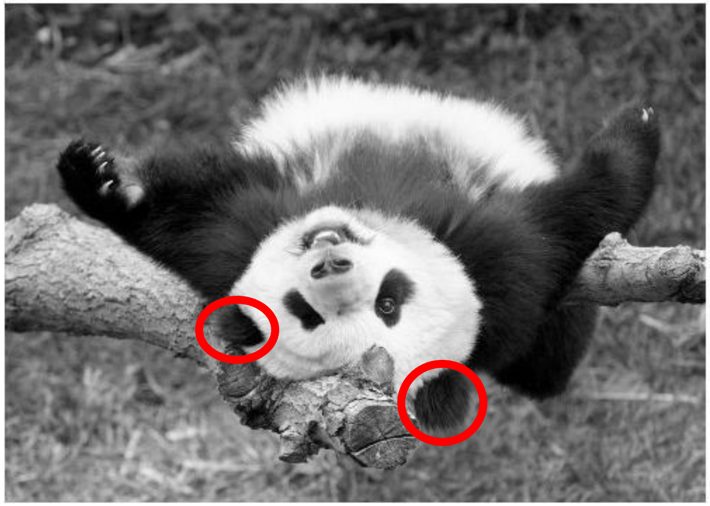
Slide by Kristen Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity AND position** similarity





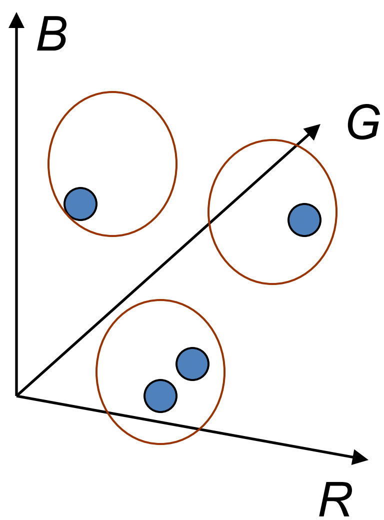
Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; so encode both similarity & proximity.

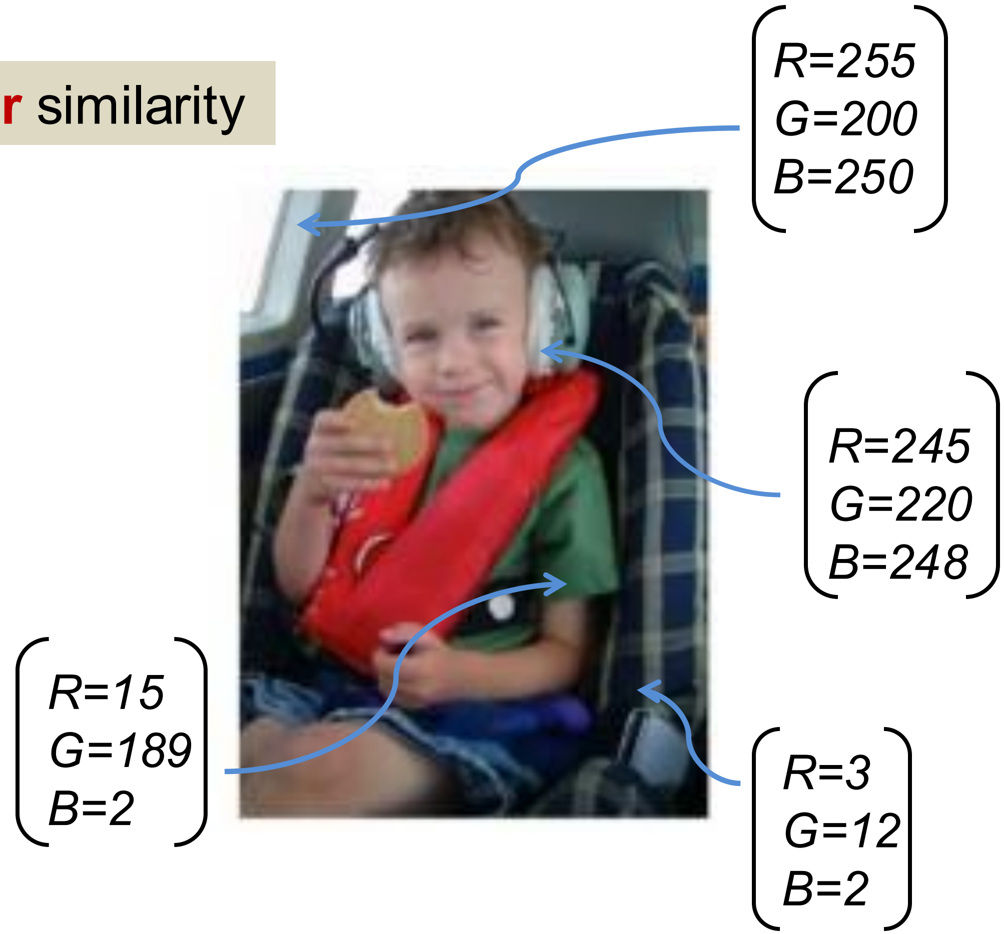
Slide by Kristen Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **colour** similarity





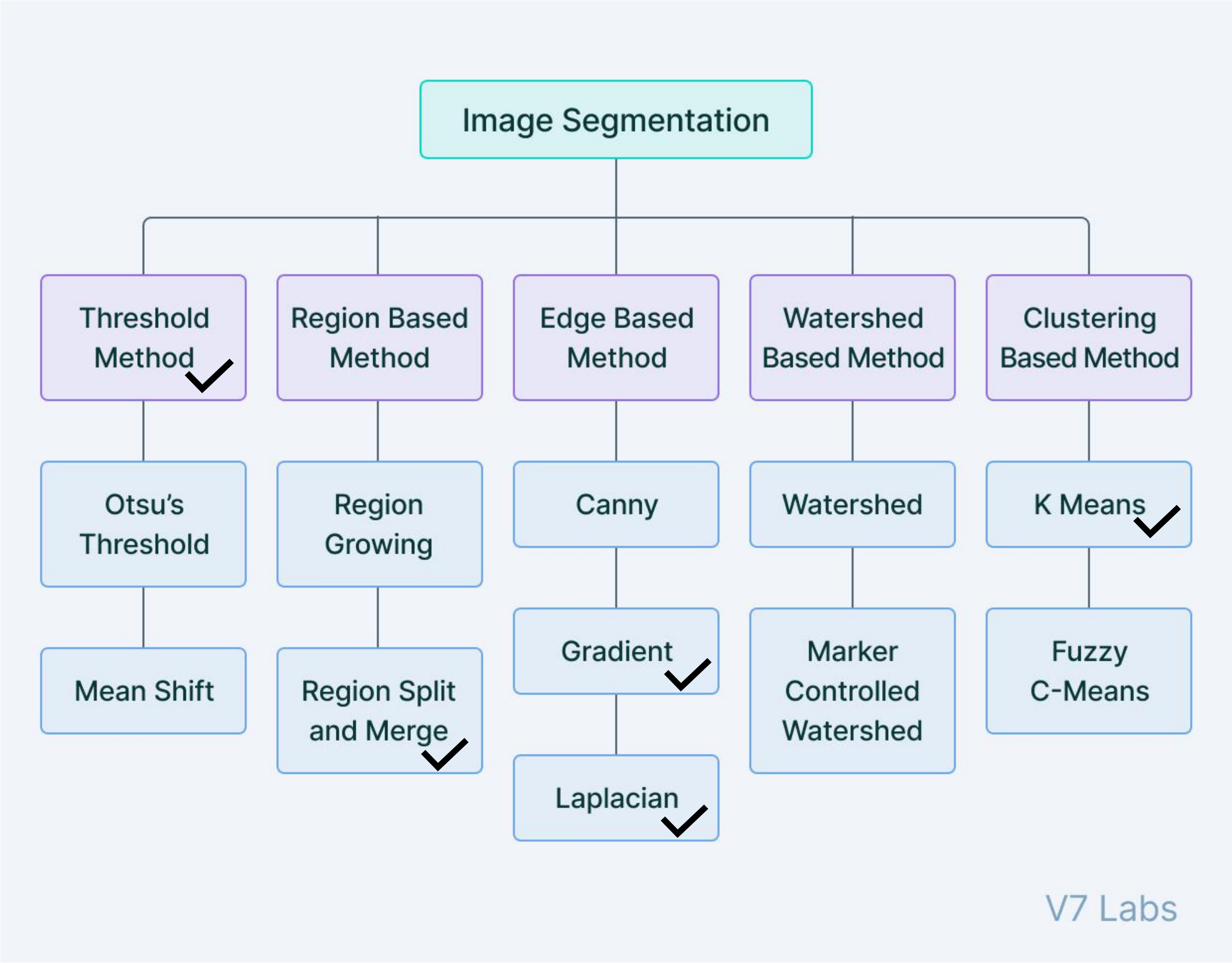
Feature space: intensity values (3D)

Slide inspired from Kristen Grauman

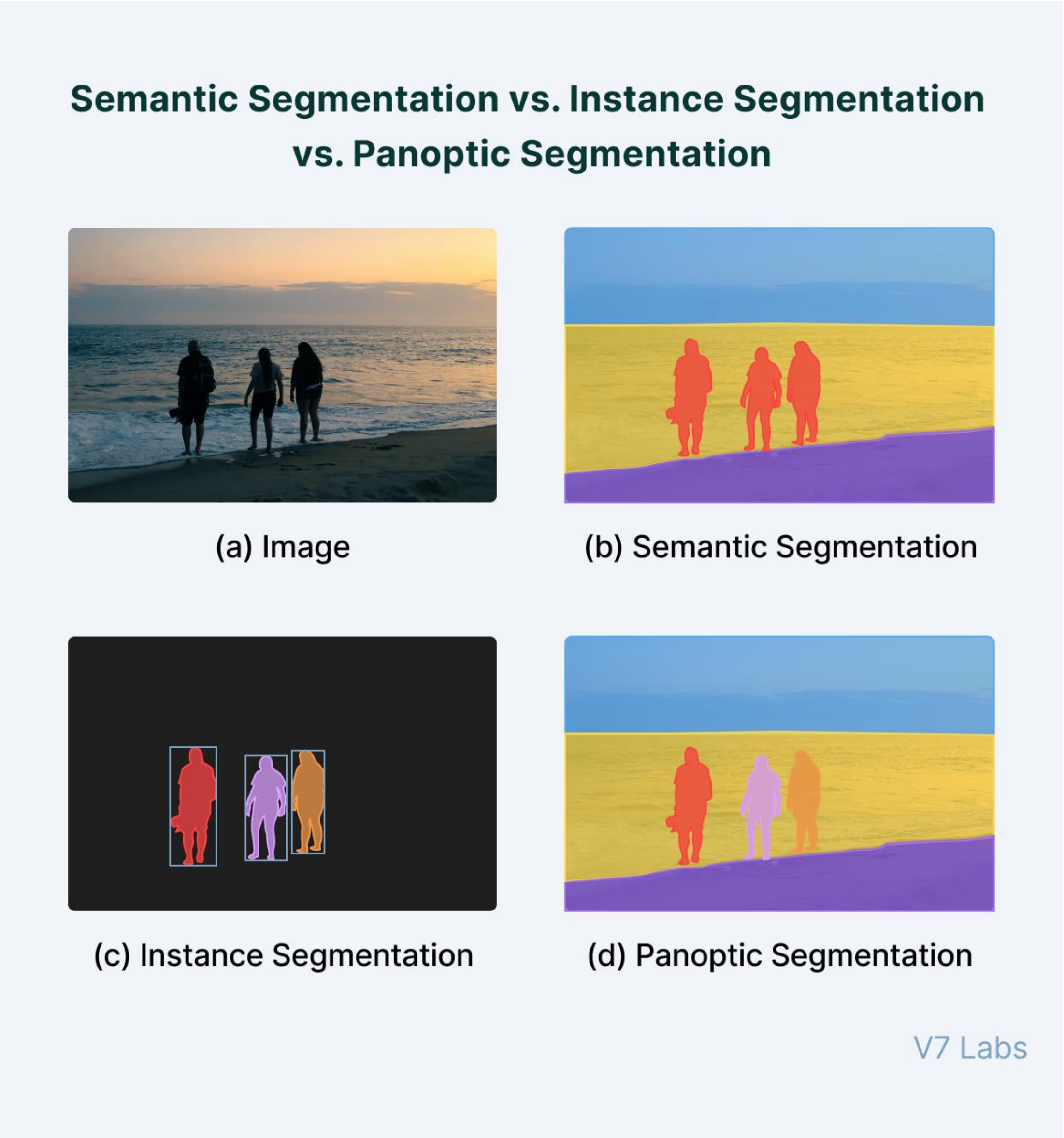
K-means Colour Segmentation

Original image | $K = 2$ | $K = 3$ | $K = 10$

Summary: Image Segmentation



Latest Segmentation Research



Next two lectures

Object Detection