IMAGE SEGMENTATION

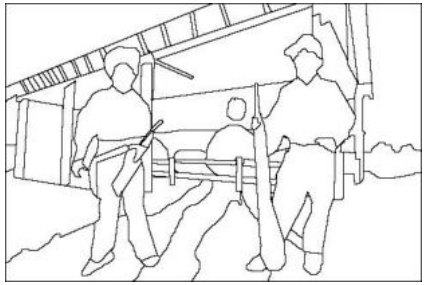
Class 4: Artificial Vision

Forsyth & Ponce: Computer Vision: A modern approach, Prentice Hall, 2002, Chapter 15

Last lecture

- Various kernels for edge extraction
  - Sobel
  - Prewitt
  - Derivative of Gaussian
  - Laplacian of Gaussian
  - Canny edges – candidates for the best edge extractors
- Edges extraction vs. image segmentation

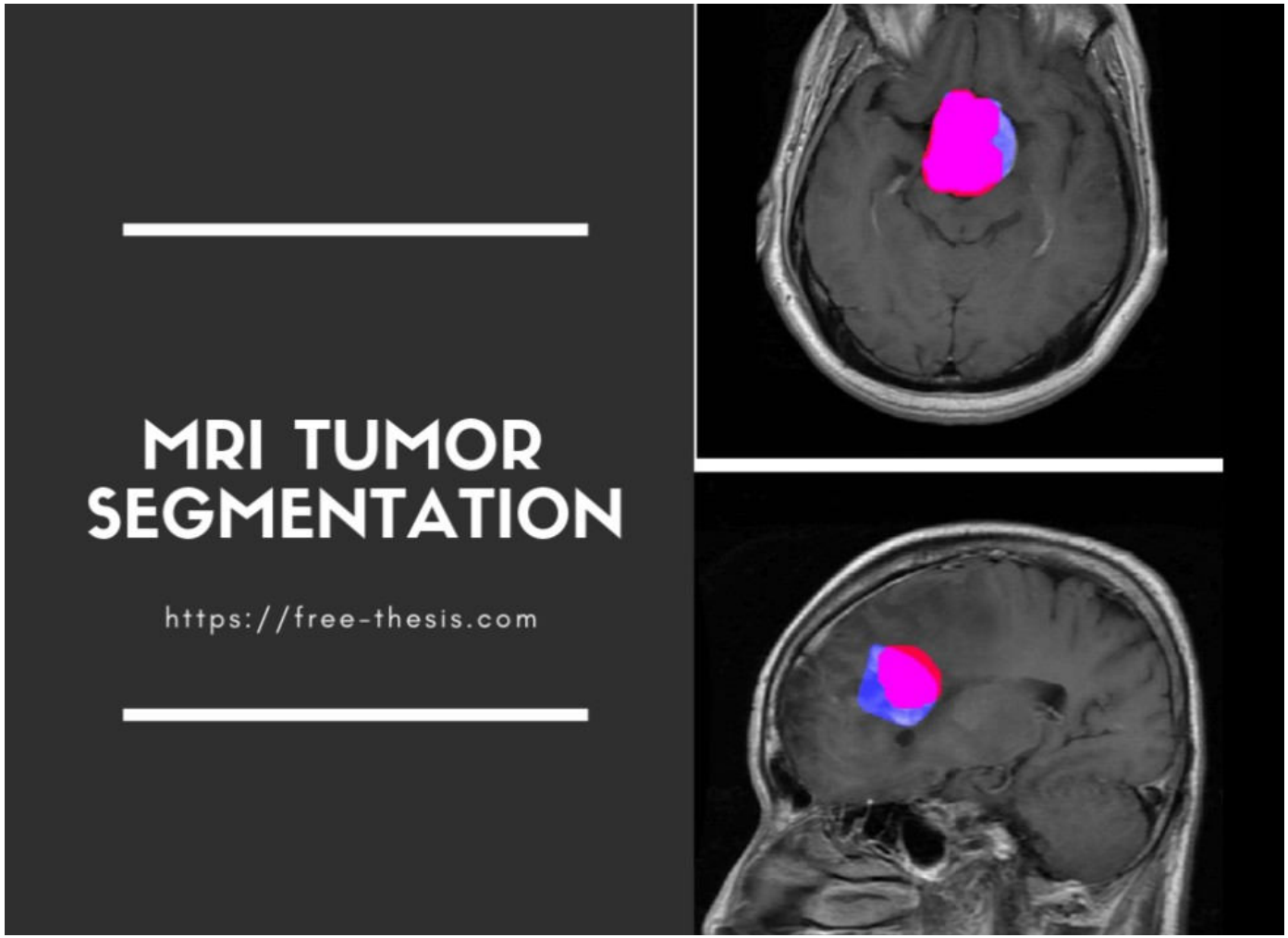
image



human segmentation

gradient magnitude

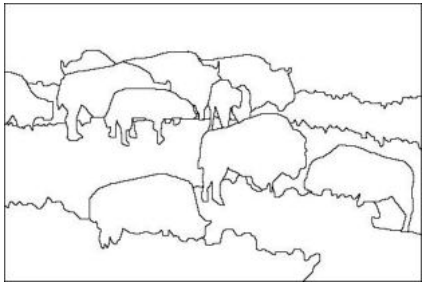
Imagine



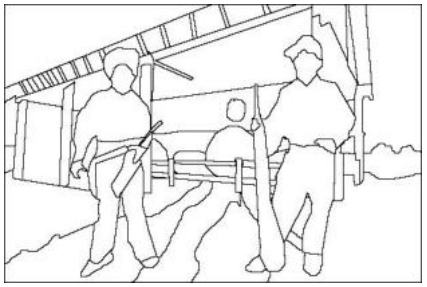
The goal of segmentation

- Separate the image into coherent “objects”
- How?
  - “Bottom-up” (from pixels to objects), or
  - “Top-down” process (from objects of interest to image pixels).

image



human segmentation



What is the difference between segmentation and edge detection?

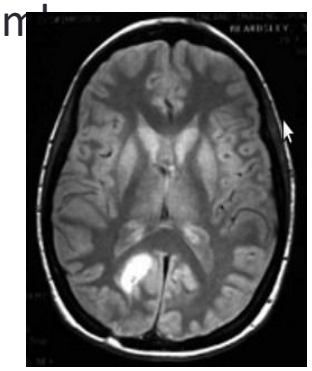
Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

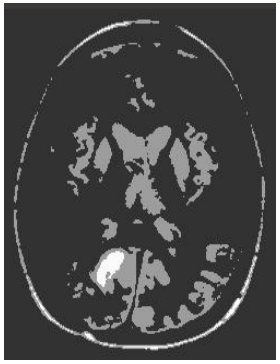
Clustering: what and why?

Clustering: group together similar points and represent them with a single token

- **Segmentation**
  - Separate the image into different regions

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster num





- **Use**:
  - Object recognition
  - Compression

Slide: Derek Hoiem

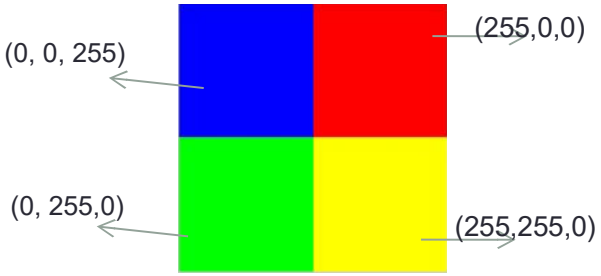
From edges to contours (index)

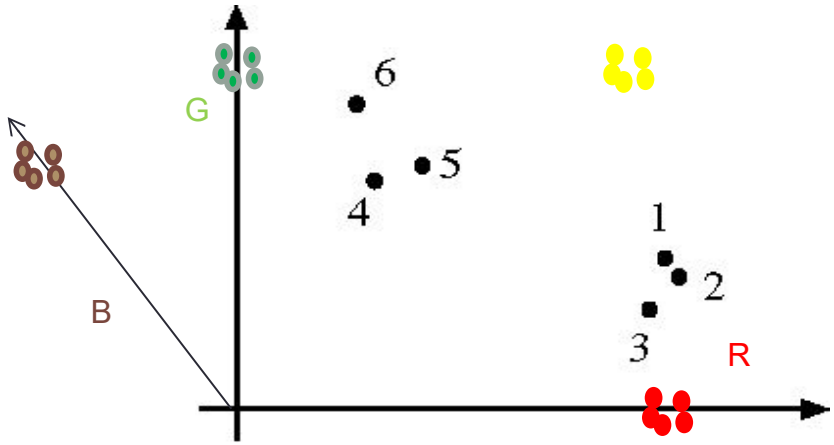
- Segmentation & Gestalt principles
- Segmentation
  - Background subtraction
  - Image pixels clustering
  - K-means and slick
  - Falzenswalb
  - Quickshift

Segmentation and Grouping

- **Goal**: Obtain a compact representation of the object from an image
- **W**e need to group pixels according to their similar appearance

Feature space: the space, where the pixels are represented.



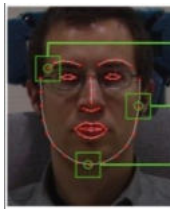


Can this space assure that separate objects are segmented separately?

Grouping vs. fitting

- **Fitting**: Top-down segmentation
  - Pixels belong together because they lie on the same object
  - Issues
    - which model?
    - which pixel goes to which element?
    - how many elements in the model?
- **Grouping/Clustering**: Bottom up segmentation
  - Pixels belong together because they are locally coherent
- These two are not mutually exclusive!



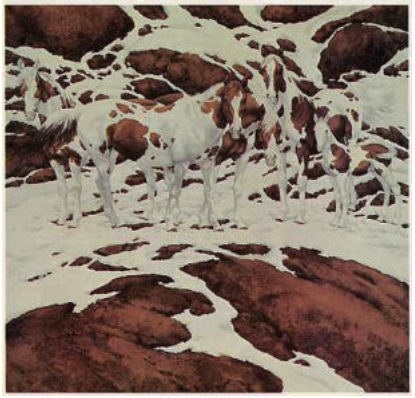
- Exercise: Given the images, in which image which approach is more suitable?

Human vision: Grouping and Gestalt

The Gestalt school of psychologists considers grouping as an important part of understanding human vision.

- A common experience of segmentation is the way that an image can resolve itself into a figure — typically, the significant, important object — and a ground — the background on which the figure lies.

Their work was characterized by attempts to write down a series of rules by which image elements would be associated together and interpreted as a group.



Perceptual organization

"…the processes by which the bits and pieces of visual information that are available in the retinal image are structured into the larger units of perceived objects and their interrelations"

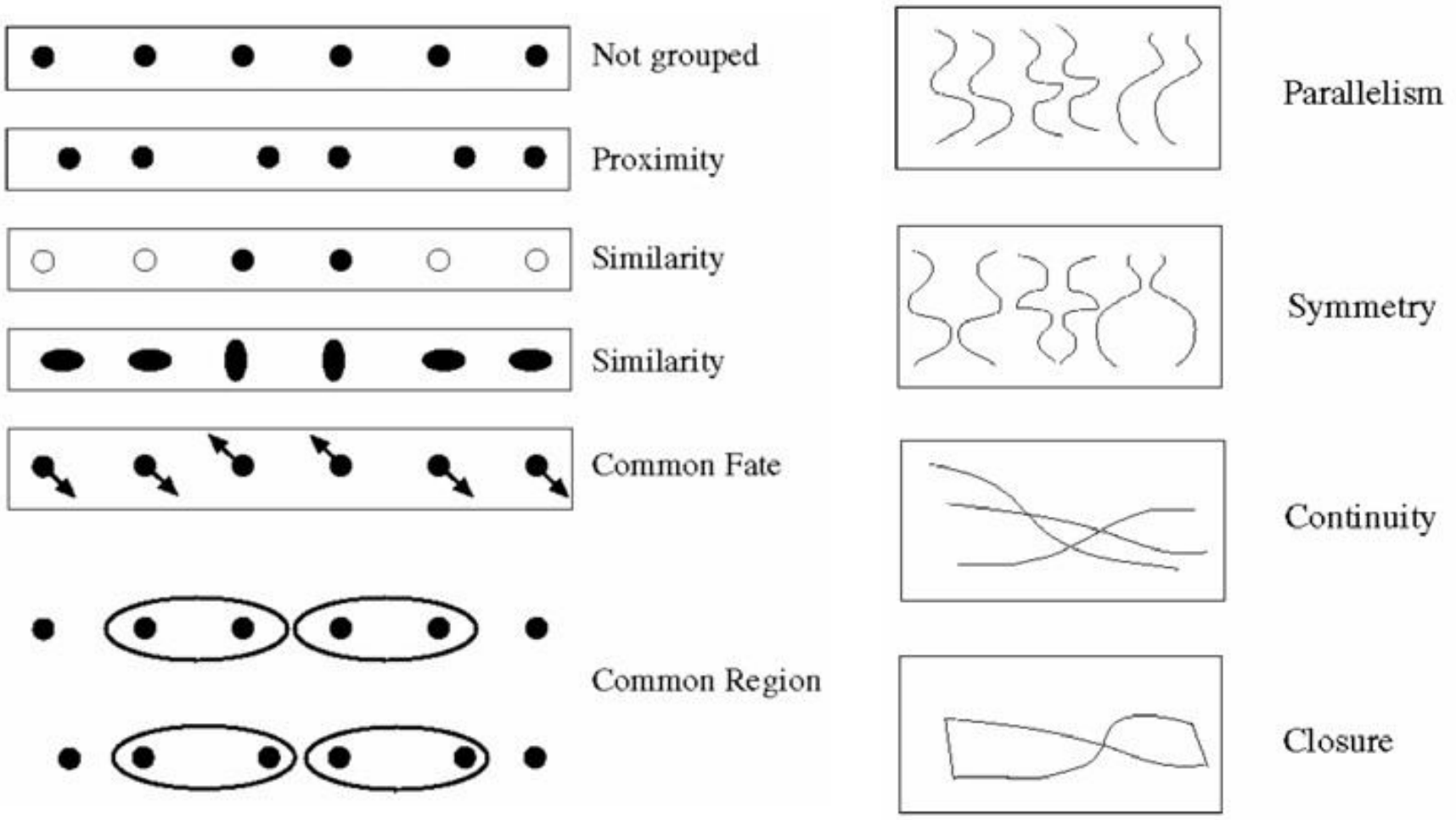
Stephen E. Palmer, *Vision Science, 1999*

http://en.wikipedia.org/wiki/Gestalt_psychology

Basic ideas of grouping in humans

- **Figure-ground discrimination**:
  - A series of factors affect whether elements should be grouped together
  - Gestalt factors
- Grouping can be seen in terms of allocating some elements to an object, others to the background.
- Elements in a collection of elements can have properties that result from their relationships (aka Muller-Lyer effect).

Gestalt factors

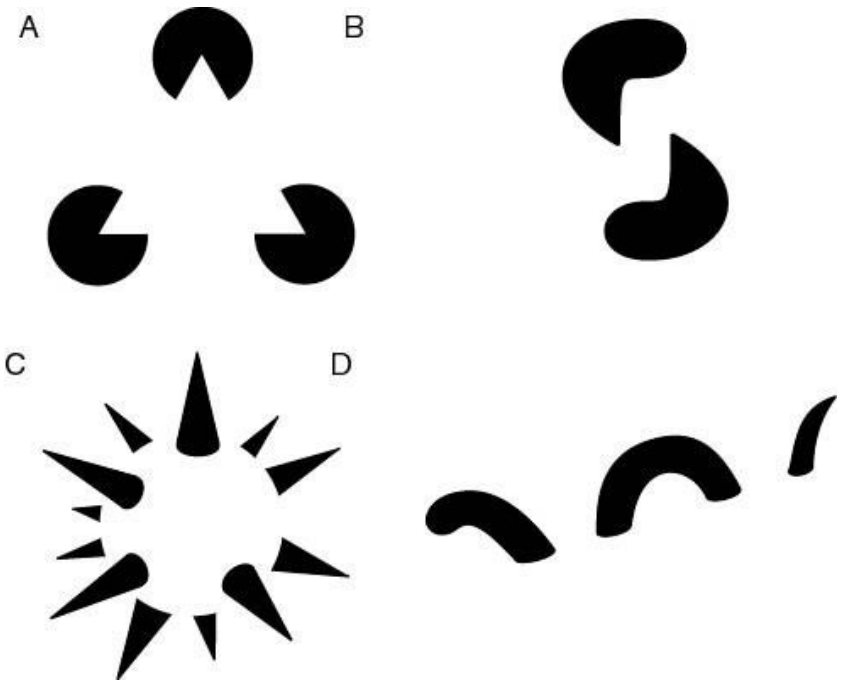


- These factors make intuitive sense, but are very difficult to translate into algorithms

The Gestalt school

- "The whole is greater than the sum of its parts"

subjective contours



occlusion

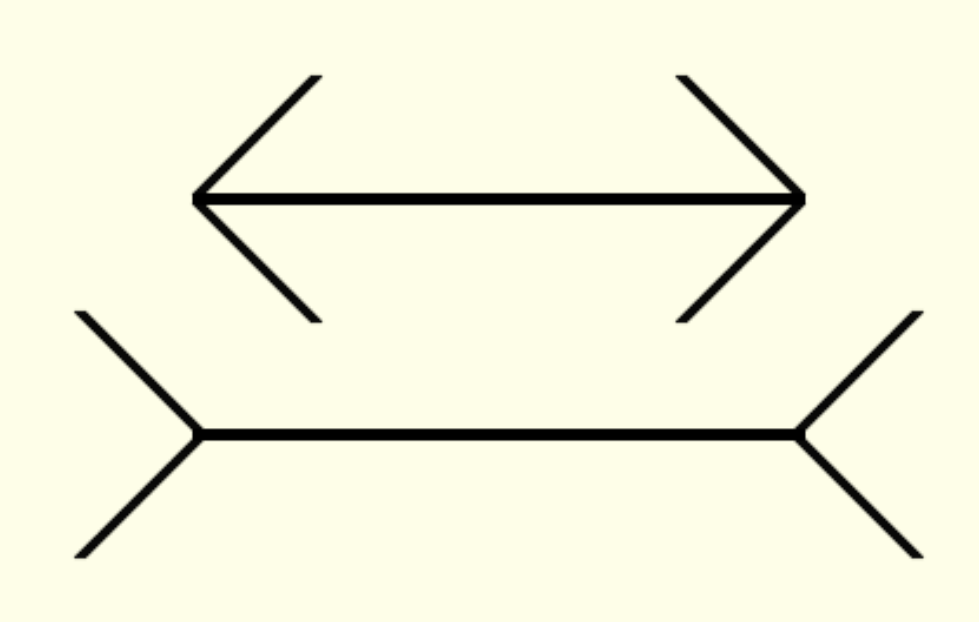
familiar configuration

http://en.wikipedia.org/wiki/Gestalt_psychology

Inspiration from psychology

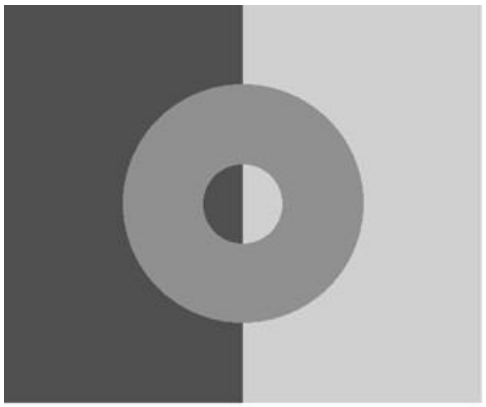
- The Gestalt school: Grouping is key to visual perception

The Muller-Lyer illusion

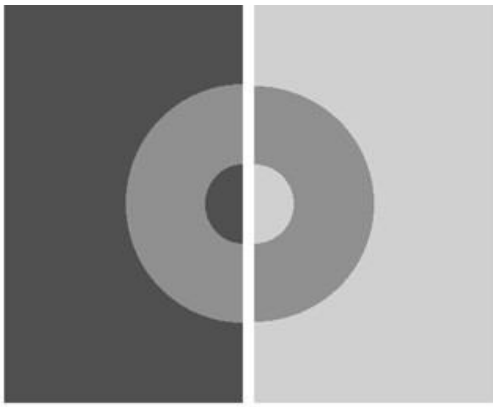


http://en.wikipedia.org/wiki/Gestalt_psychology

Influences of grouping



a



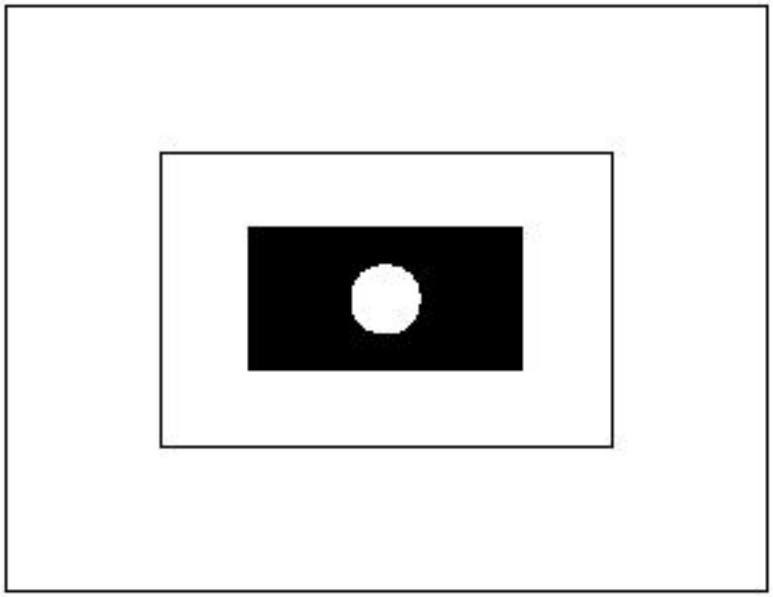
b

c

Grouping influences other perceptual mechanisms such as lightness perception

http://web.mit.edu/persci/people/adelson/publications/gazzan.dir/koffka.html

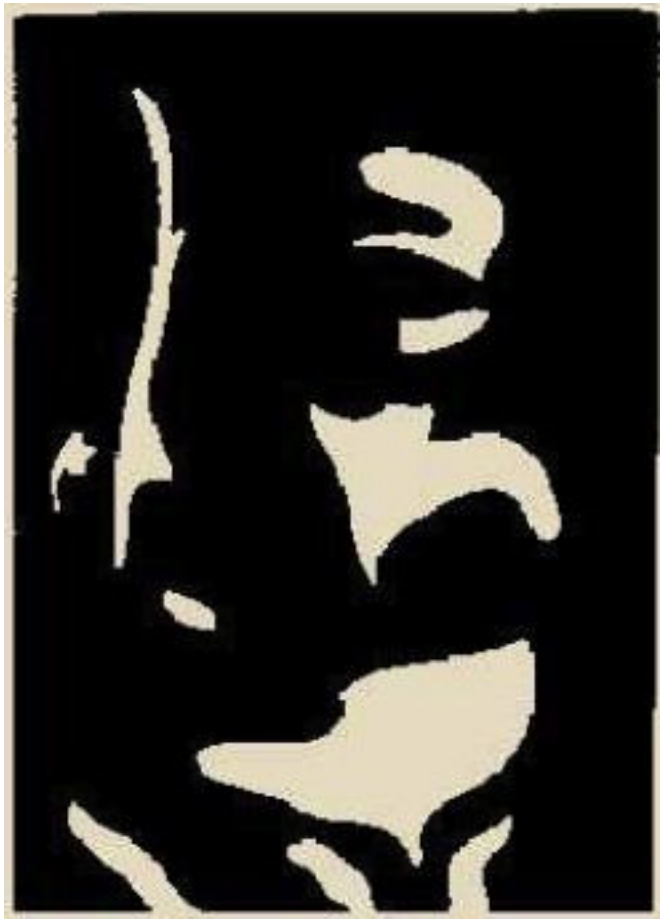
Human vision: Grouping and Gestalt

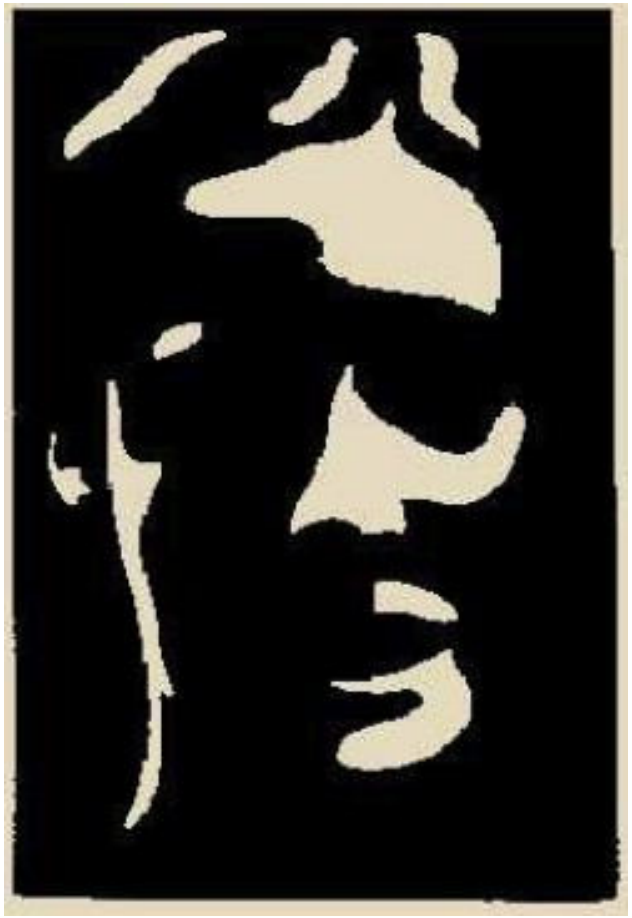


Segmentation is not just to think about pictures in terms of separating figure and ground (e.g. foreground and background).

- This is (partially) because there are too many different possibilities in pictures like this one.
  - Is a square with a hole in it the figure? Or white circle on a black box?

Familiarity





Human vision: Grouping and Gestalt

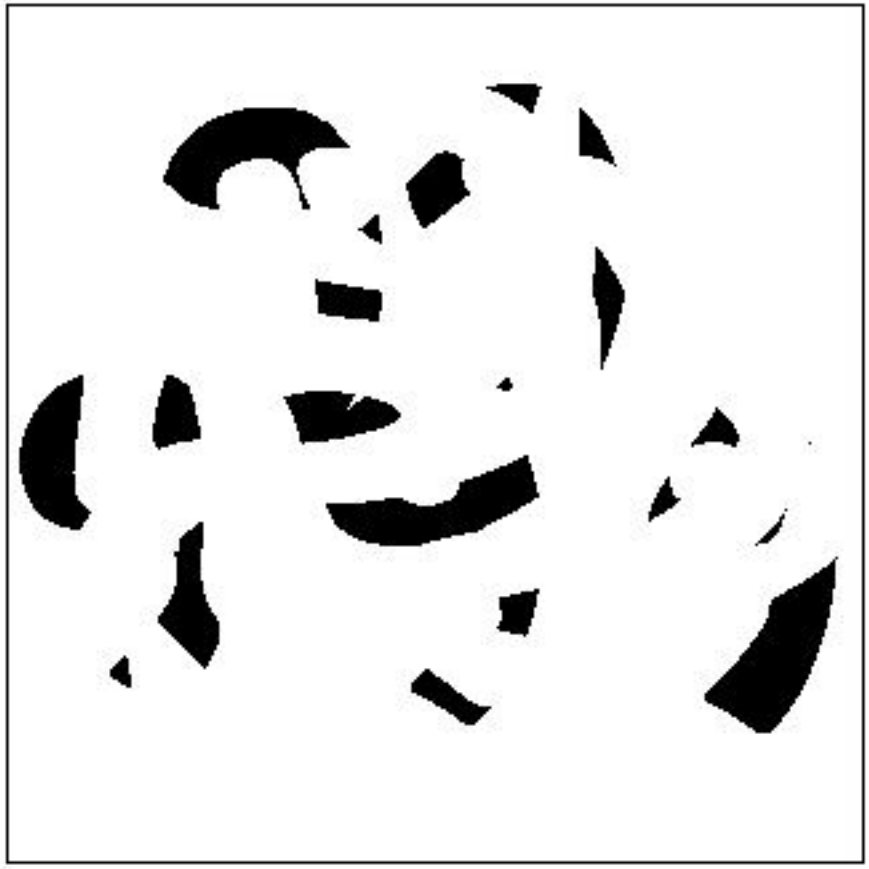
Why do these tokens belong together?

A possibility is that they all lie on a sphere -- but then, if we didn't know that the tokens belonged together, where did the sphere come from?

Occlusions

Magritte, 1957

Occlusions and segmentation:
Segmentation is a global process

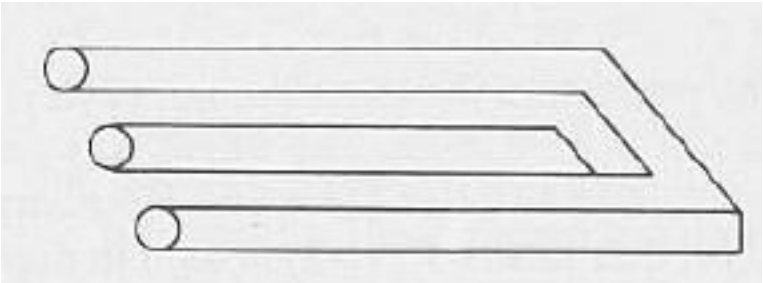


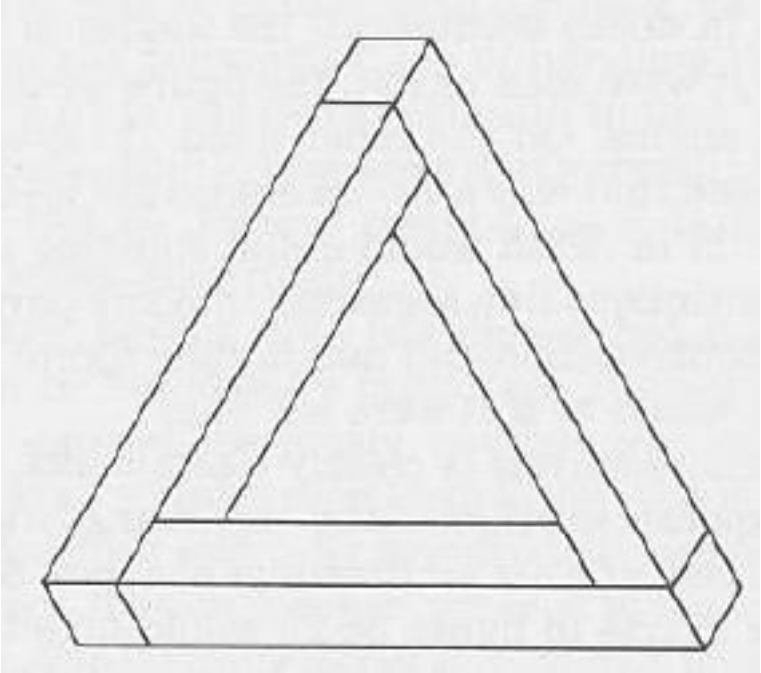
What are the occluded numbers?

Occlusion cues seem to be very important in grouping.

This tendency to prefer interpretations that are explained by occlusion leads to interesting effects.

… but not too global





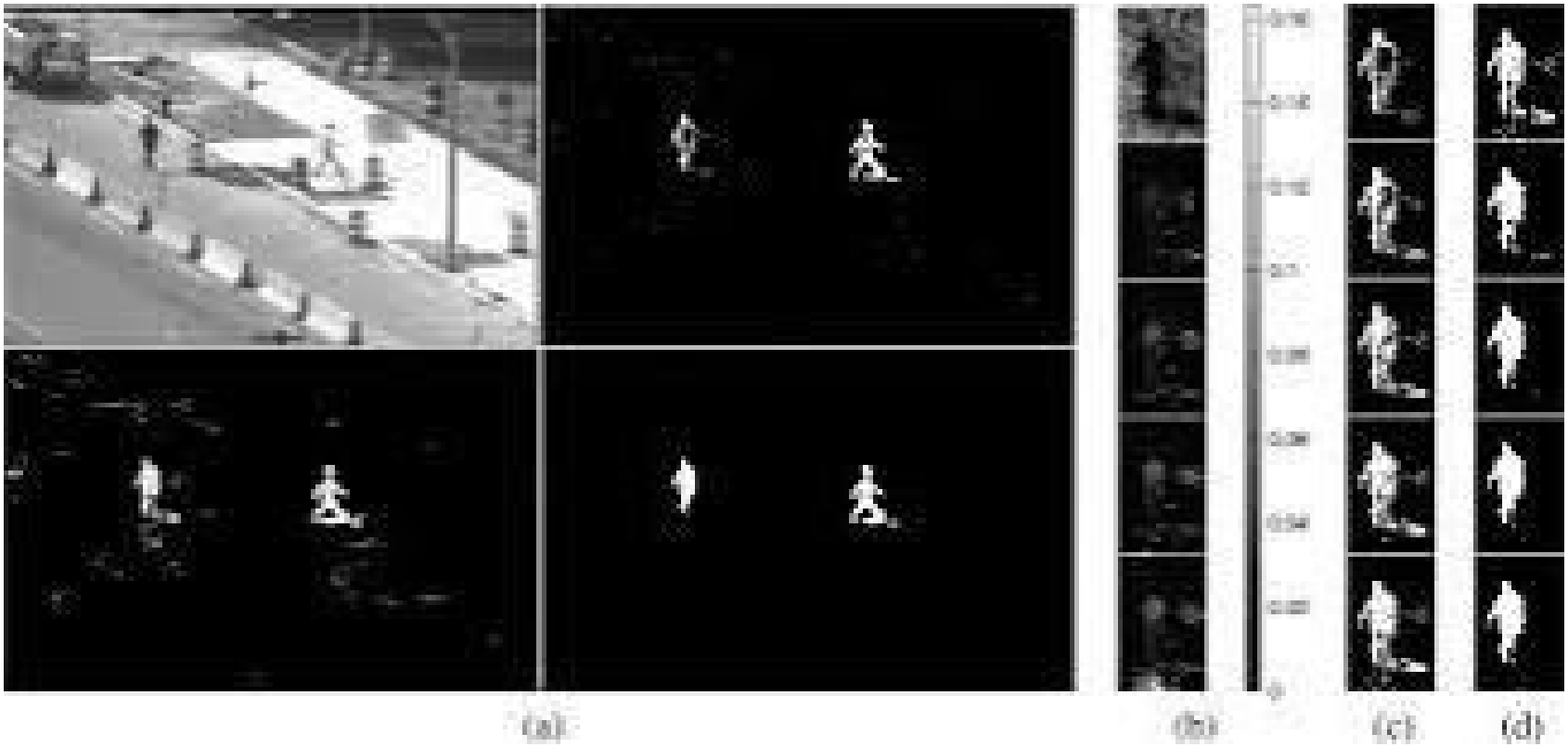
From Gestalt, there are some interesting top-down segmentation techniques.

We will work mainly with bottom-up segmentation techniques.

From edges to contours (index)

- Segmentation & Gestalt principles
- Automatic segmentation
  - Background subtraction
  - Image pixels clustering
  - K-means and slick
  - Falzenswalb
  - Quickshift

Background subtraction



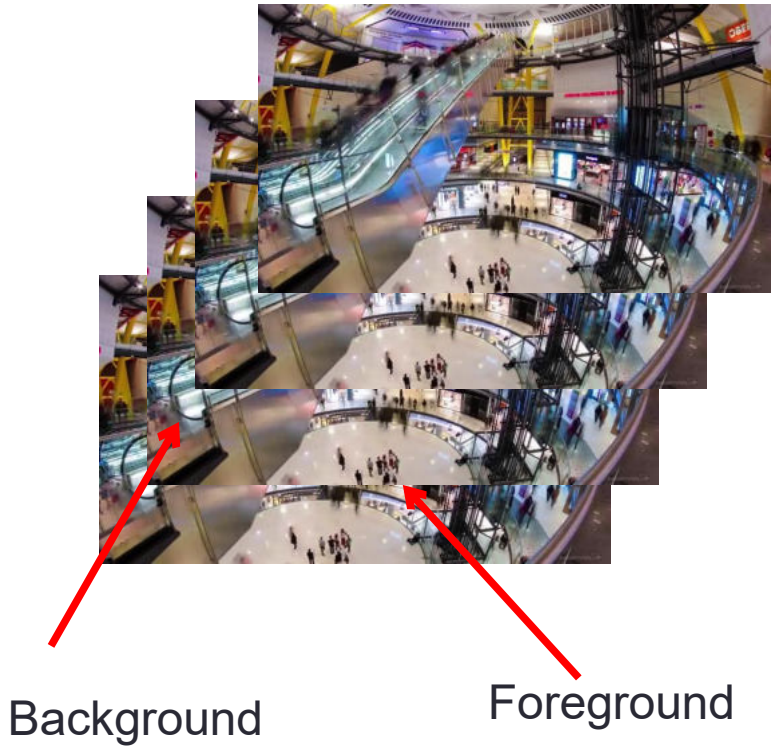
Technique: Background Subtraction

If we know what the background looks like, it is easy to identify “interesting bits”

- Applications
  - Person in an office
  - Tracking cars on a road
  - Surveillance, etc.

Technique: Background Subtraction

How to extract the background? What is the difference between background and foreground?



Technique: Background Subtraction

How to extract the background? What is the difference between background and foreground?

How do the foreground pixels change in time?

- dynamic

How do the background pixels change in time?

- static

Background

Foreground

Technique: Background Subtraction

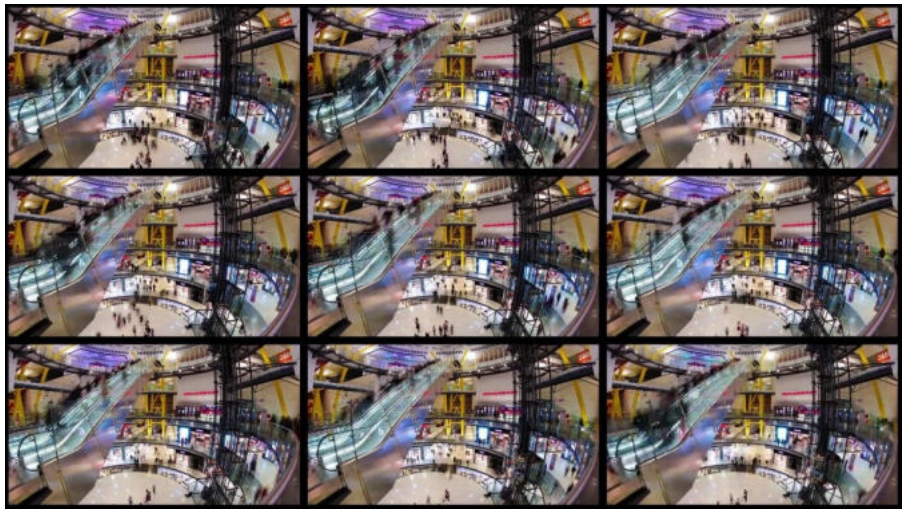
How to extract the background? What is the difference between background and foreground?

If we think of a concrete pixel, what would be its state (foreground, background) most of the time?

What filter would give us the most frequent value?

Technique: Background Subtraction

- **Approach**:
  - Apply a median filter to all pixels along the temporal axis to estimate the background image (BI)
  - Subtract BI from the current frame
  - Large absolute values are interesting pixels



Background image

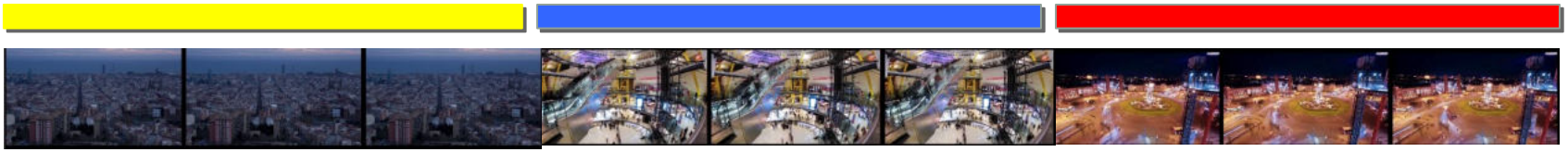
Foreground object

Technique: Background Subtraction

- Pros:
  - Simple and fast
- Cons:
  - Needs many images
  - Needs static background
  - Noisy segmentation

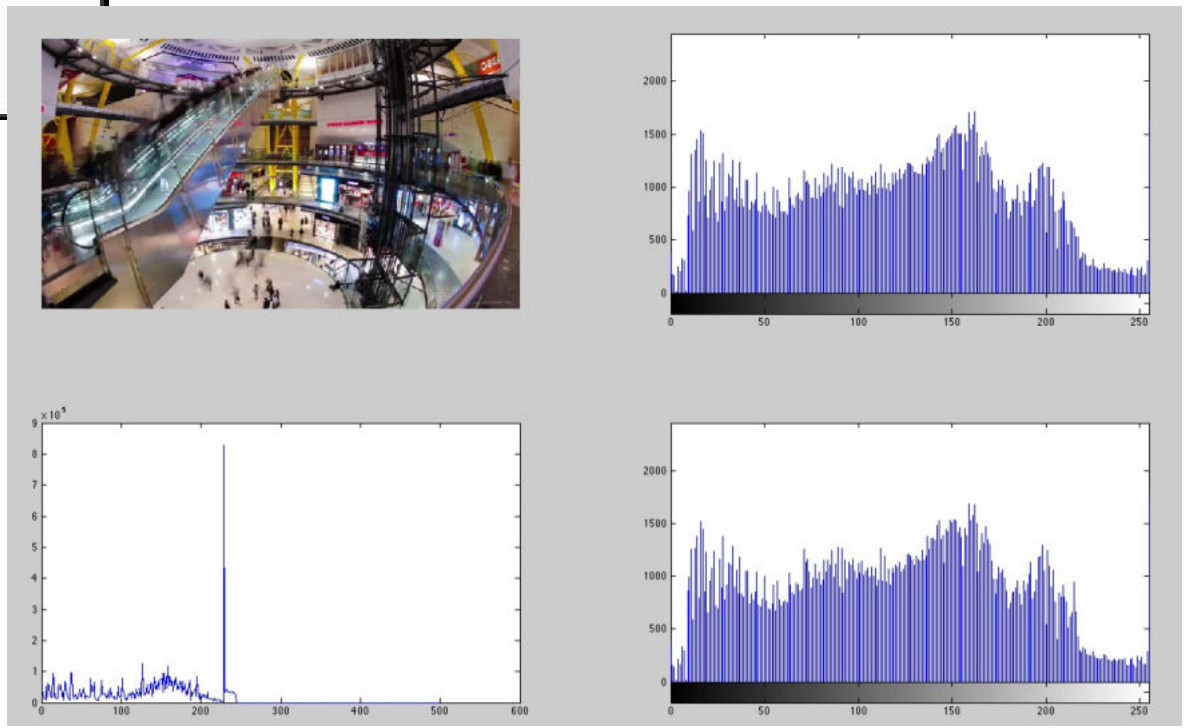
Can we apply it to any video?
Video segmentation

Video segmentaion consists of splitting
the vídeo in temporal segments



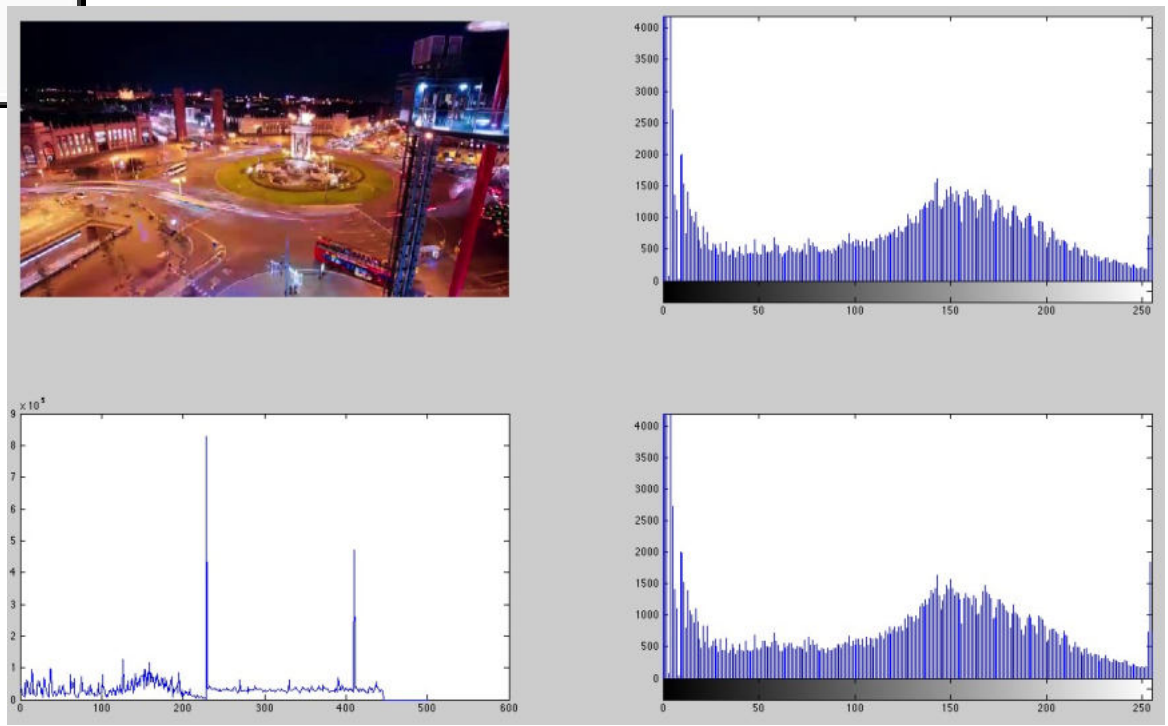
How to extract different events of the video?

Video segmentation



What happens with the histograms of consecutive frames?

Video segmentation

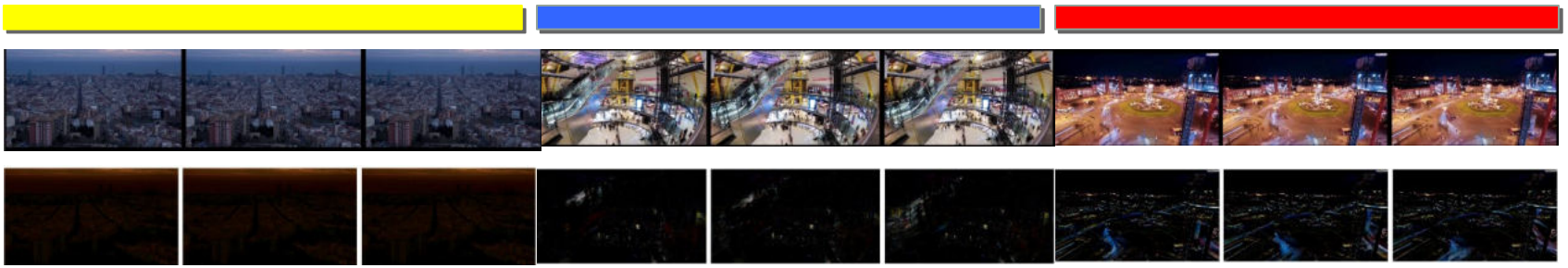


How to extract different events of the video?

Video segmentation

Algorithm:

1. Obtain the difference between consecutive frames (e.g. by histograms differences)
2. Decide where there is a jump in the frames appearance.
3. Cut video segments.
4. Obtain the foreground image (e.g. using the median filter).
5. Extract the foreground objects of the detected video segments.



Technique: Video segmentation

- Pros:
  - Simple and fast
- Cons:
  - Too simple, it does not capture continuous transition
  - **It does not separate and track objects**

Original video

Background image

Foreground object

From edges to contours (index)

- Human grouping & Gestalt principles
- Segmentation
  - Background subtraction
  - Image pixels clustering (bottom-up segmentation)
  - K-means and slick
  - Falzenswalb
  - Quickshift

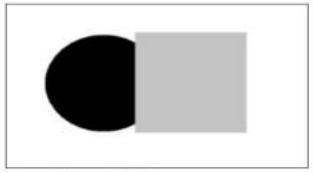
Clustering

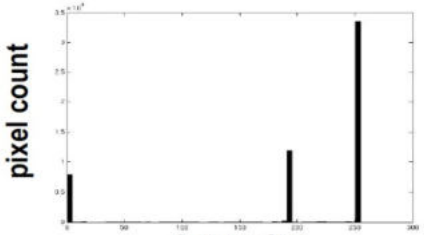
Key Challenges:

1) What does make two points/images/patches similar?

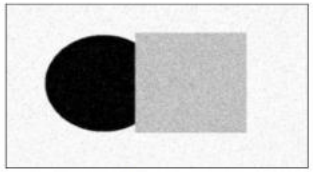
2) How do we compute an overall grouping from pairwise similarities?

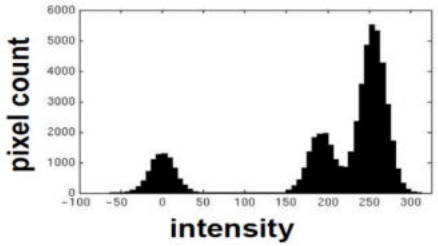
In the ideal case:





In practice:

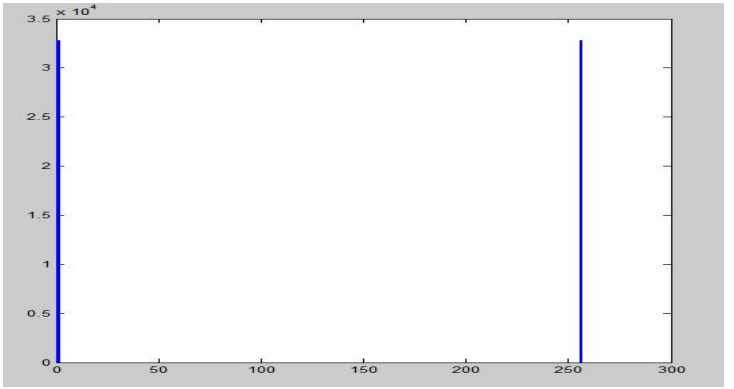




Slide: Derek Hoiem

Issues

- How do we decide that two pixels are likely to belong to the same object?



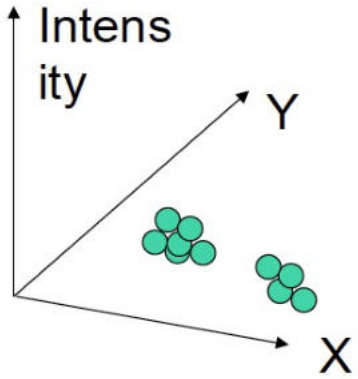
- How many regions are there?

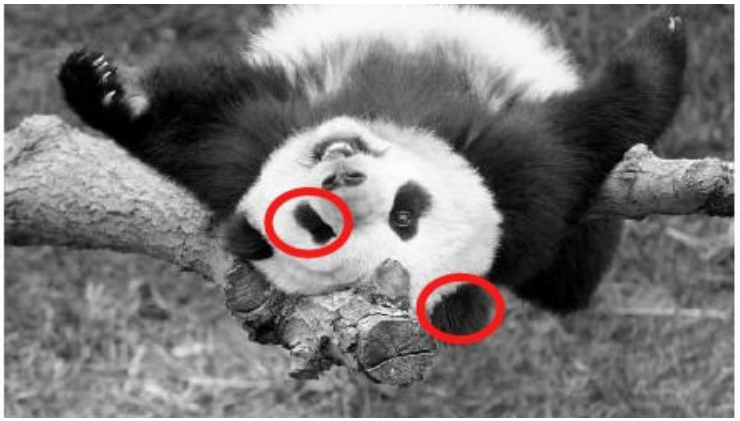
Is the histogram enough to separate image objects?

Feature space for clustering

- **Feature space:** the pixel information used to describe it and treat it (segment it, classify it, etc.).

- What information to use in order to group pixels in an object?



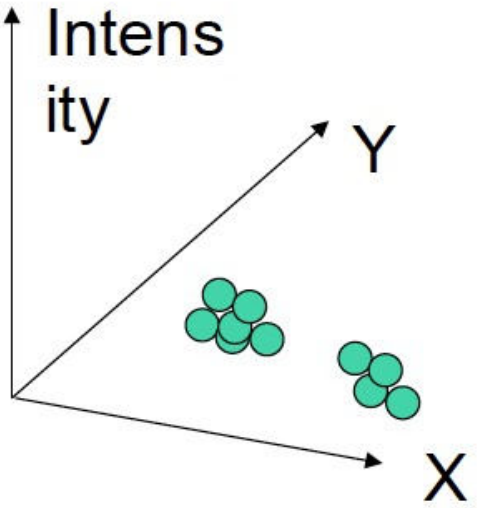


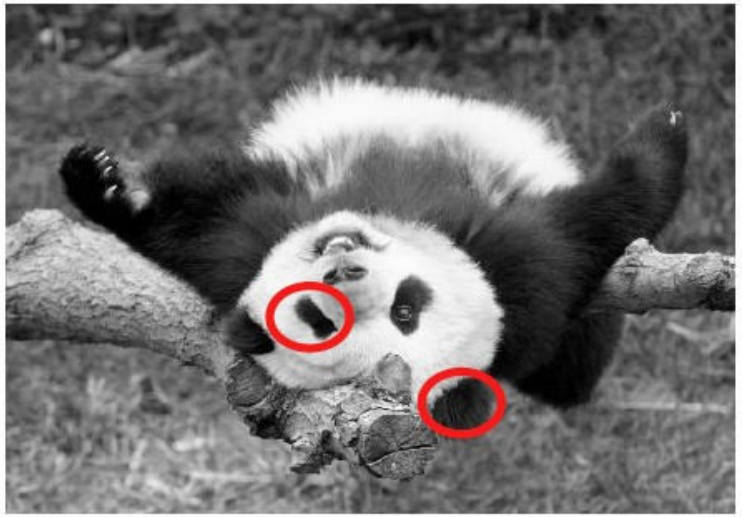
Kristen Grauman

Feature space for clustering

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

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





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

Kristen Grauman

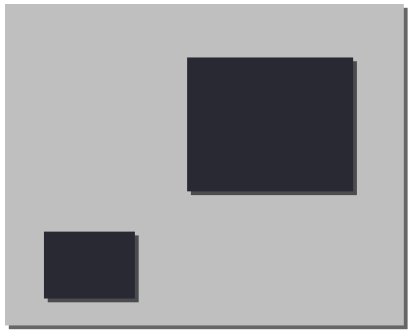
Feature space for clustering: including spatial relationships

Augment data to be clustered with spatial coordinates.

Exercice: Draw the feature space for clustering if:

a) Feature space is the grey level space

b) Feature space is the grey level space and spatial coordinates



$$z = \begin{pmatrix} I \\ x \\ y \end{pmatrix}$$

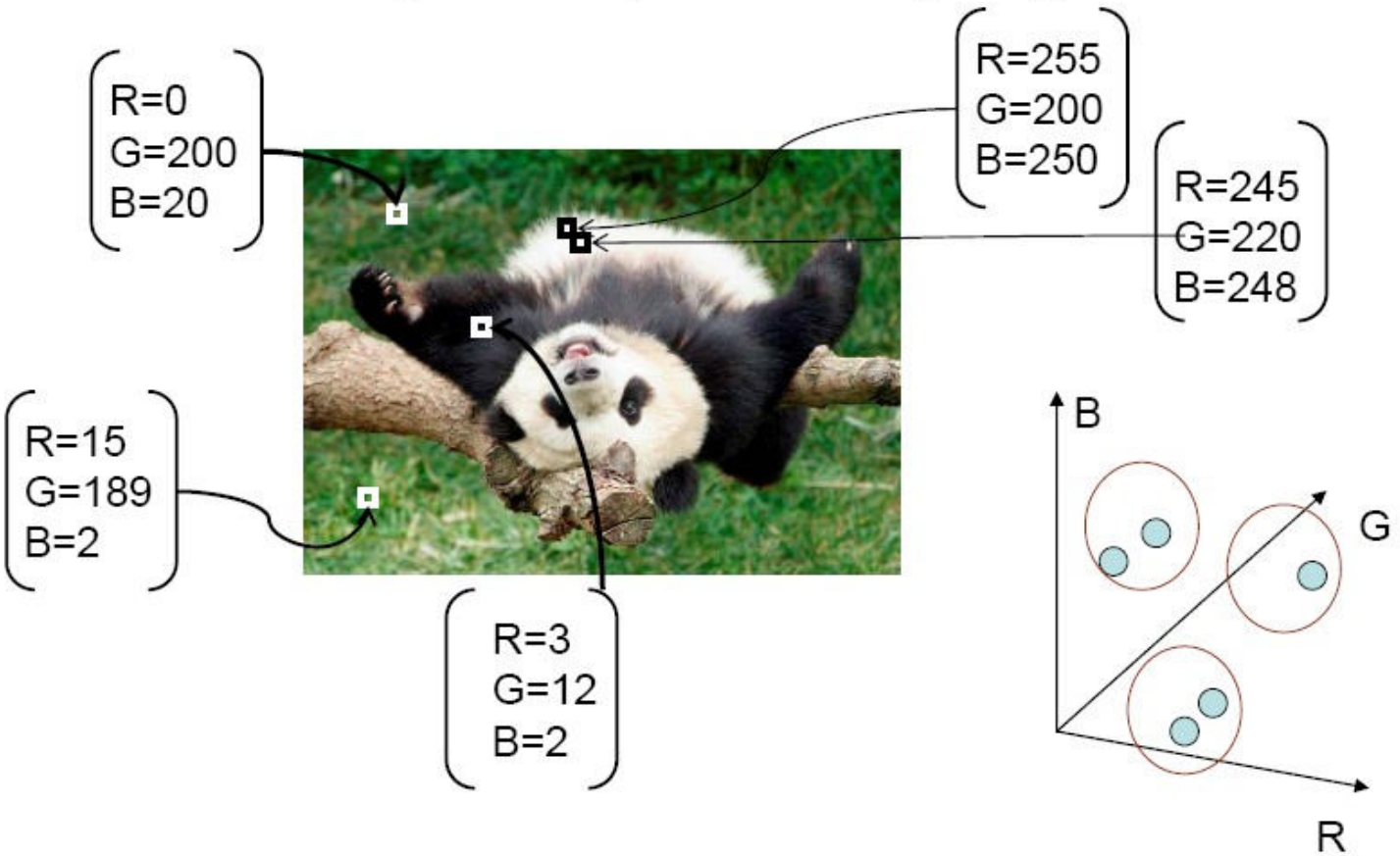
Intensity coordinates

Spatial coordinates

What about color images?

Feature space for clustering

- Cluster similar pixels (features) together

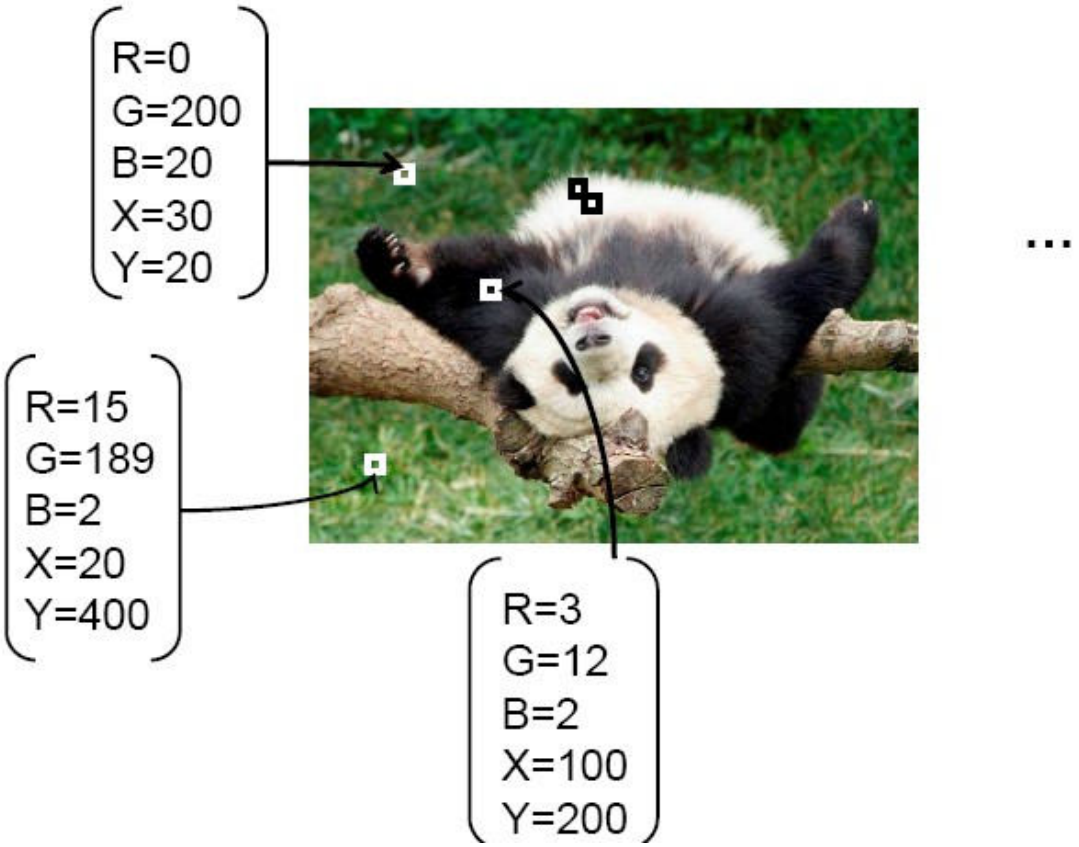


What do we miss here?

Source: K. Grauman

Feature space for clustering

- Cluster similar pixels (features) together



Source: K. Grauman

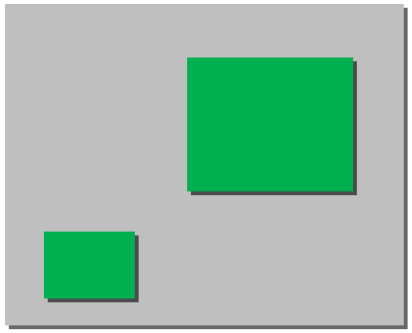
Feature space for clustering: including spatial relationships

Augment data to be clustered with spatial coordinates.

Exercice: Draw the feature space for clustering in

a) Feature space is the color space

b) Feature space is the color space and spatial coordinates



$$z = \begin{pmatrix} R \\ G \\ B \\ x \\ y \end{pmatrix}$$

Color coordinates

Spatial coordinates

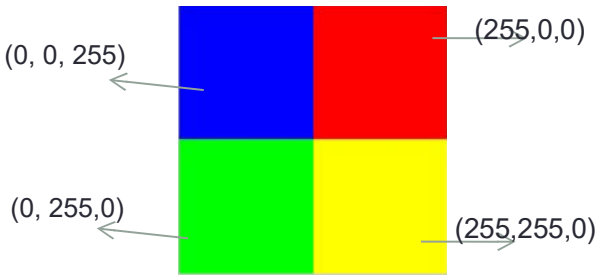
A simple segmentation algorithm

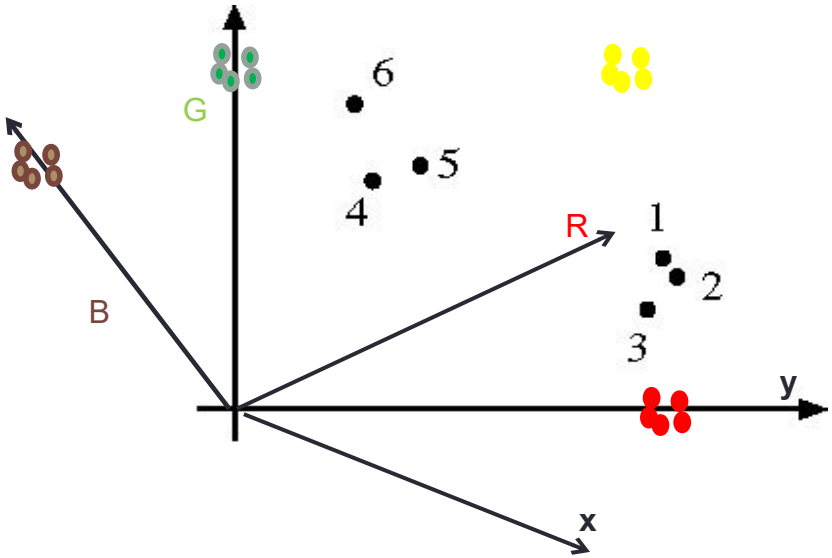
- Each pixel of the image is described by a vector in the feature space:

  $z = (R,G,B,x,y), \ldots$

- Run a clustering algorithm using some distance between pixels (e.g. Euclidean distance):

$$D(\text{pixel}_i, \text{pixel}_j) = || z_i - z_j ||^2$$





How do we cluster?

We will see 2 main approaches:

- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest two clusters until some criteria of stopping merging.

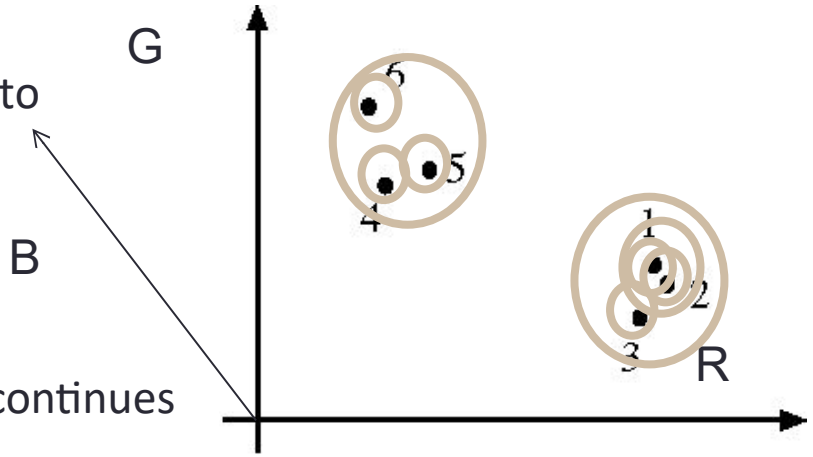
- K-means
  - Iteratively re-assign points to the nearest cluster center

Agglomerative clustering

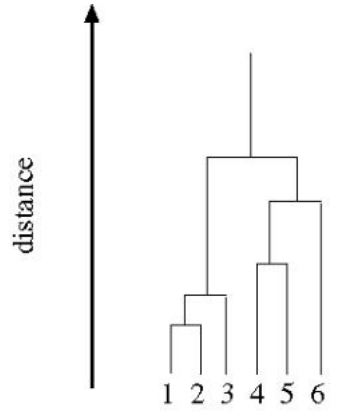
- Cluster together (pixels, tokens, etc.) that belong together.

- Agglomerative clustering
  - Attach the pixel closest to a cluster it is closest to
  - Repeat until…

- Dendrograms
  - Yield a picture of output as clustering process continues



- Divisive clustering
  - Split cluster along best boundary
  - Repeat until…



The final clusters represent the segmented objects.

Agglomerative clustering

- Pros:
  - Simple
- Cons:
  - When to stop?
  - How to calculate the distance between points?
  - No information about pixels distribution

Algorithm 15.3: Agglomerative clustering, or clustering by merging

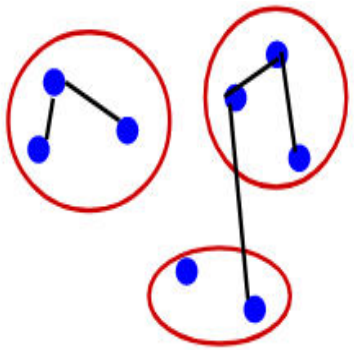
```
Make each point a separate cluster
Until the clustering is satisfactory
    Merge the two clusters with the
      smallest inter-cluster distance
end
```

Algorithm 15.4: Divisive clustering, or clustering by splitting

```
Construct a single cluster containing all points
Until the clustering is satisfactory
    Split the cluster that yields the two
      components with the largest inter-cluster distance
end
```

Felzenszwalb

Fast, minimum spanning tree based clustering on the image grid.



Graph G = (V, E)

V is set of nodes (i.e. pixels)

E is a set of undirected edges between pairs of pixels

$w(v_i, v_j)$ is the weight of the edge between nodes v_i and v_j.

S is a segmentation of a graph G such that G' = (V, E')

where $E' \subset E$.

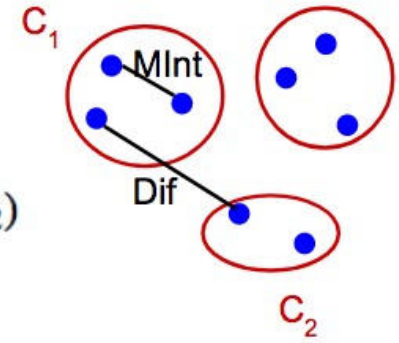
S divides G into G' such that it contains distinct components (or regions) C.

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

Predicate D determines whether there is a boundary for segmentation.

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$



Where

$Dif(C_1, C_2)$ is the difference between two components.

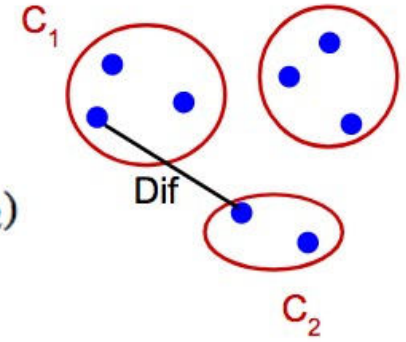
Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

Predicate D determines whether there is a boundary for segmentation.

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

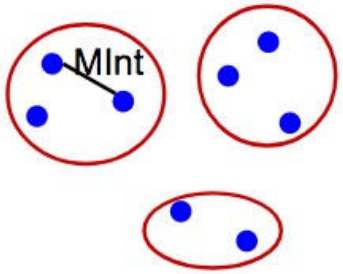


The different between two components is the minimum weight edge that connects a node v_i in component C_1 to node v_j in C_2

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

Predicate D determines whether there is a boundary for segmentation.



$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

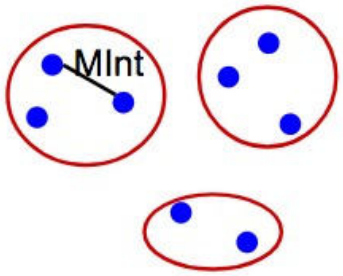
$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

Int(C) is to the maximum weight edge that connects two nodes in the same component.

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb



Predicate D determines whether there is a boundary for segmentation.

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

where $\tau(C) = k/|C|$

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

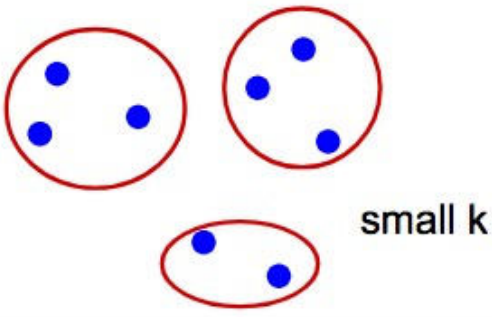
$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

where $\tau(C) = k/|C|$

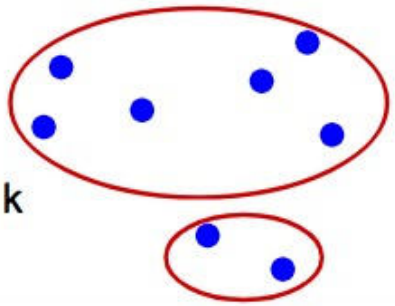
T(C) sets the threshold by which the components need to be different from the internal nodes in a component.

Properties of constant k:

- If k is large, it causes a preference of larger objects.
- k does not set a minimum size for components.



small k



large k

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

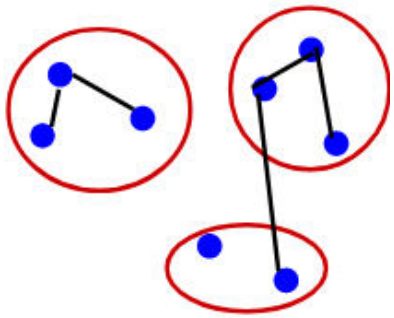
Felzenszwalb

Nearest Neighbor Graph Weights

Project every pixel into feature space defined by (x, y, r, g, b).

Weights between pixels are determined using L_2 (Euclidian) distance in feature space.

Edges are chosen for only top ten nearest neighbors in feature space to ensure run time of O(n log n) where n is number of pixels.



Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

Nearest Neighbor Graph Results

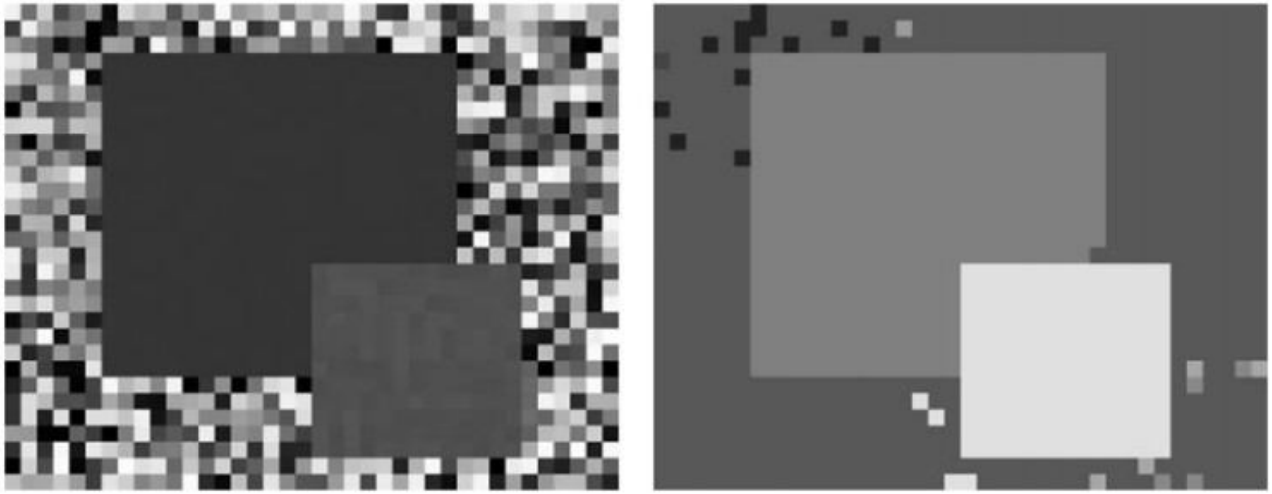


Image Segmentation

- Highly variable region is placed in one large segment.
- Captures global image features.

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

Nearest Neighbor Graph Results

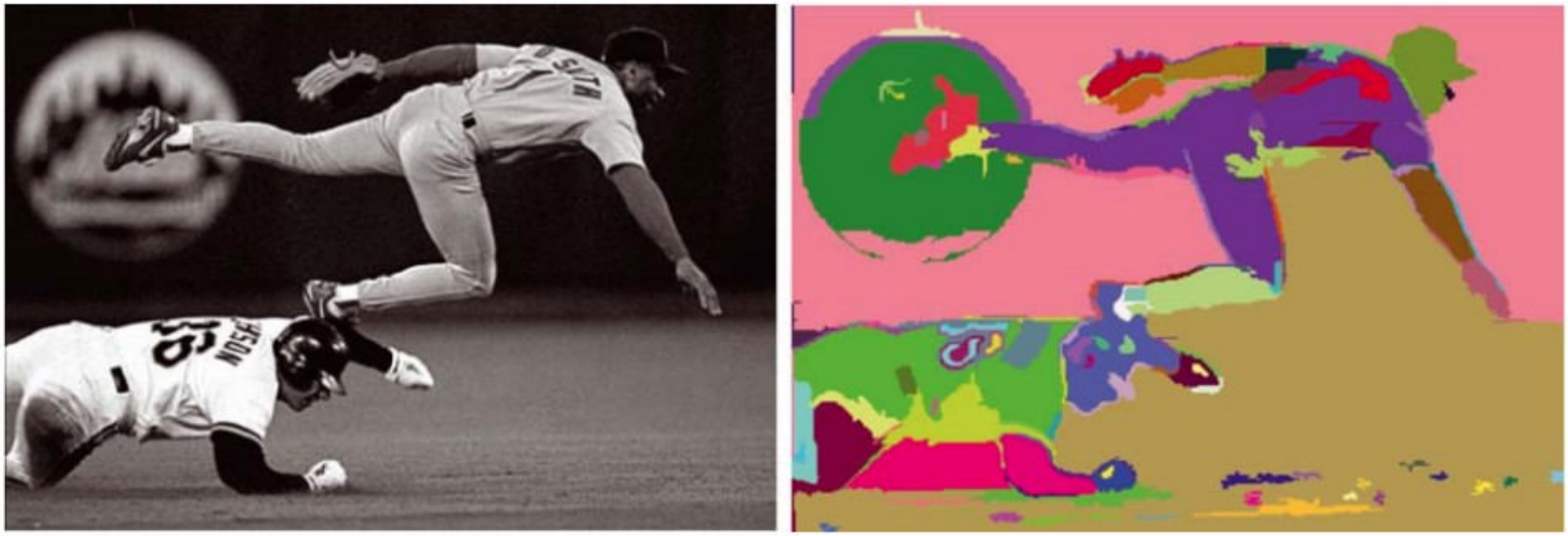
Non Spatially connection regions of the image are placed in the same component. For example:

- Flowers on the picture to the right
- Tower and lights on picture to the right.

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenszwalb

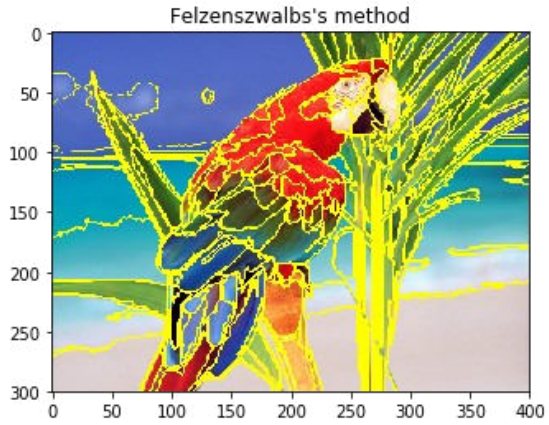
Grid Graph Results



- grass and clothes with variations each have their own component.
- Due to long slow change in intensity from grass to black area, it gets mis-segmented into one component.
- Preserves small components like name tags and numbers.

Efficient graph-based image segmentation, Felzenszwalb, P.F. and Huttenlocher, D.P. International Journal of Computer Vision, 2004

Felzenswalb segmentation



Skimage:

Complexity: O(NlogN)

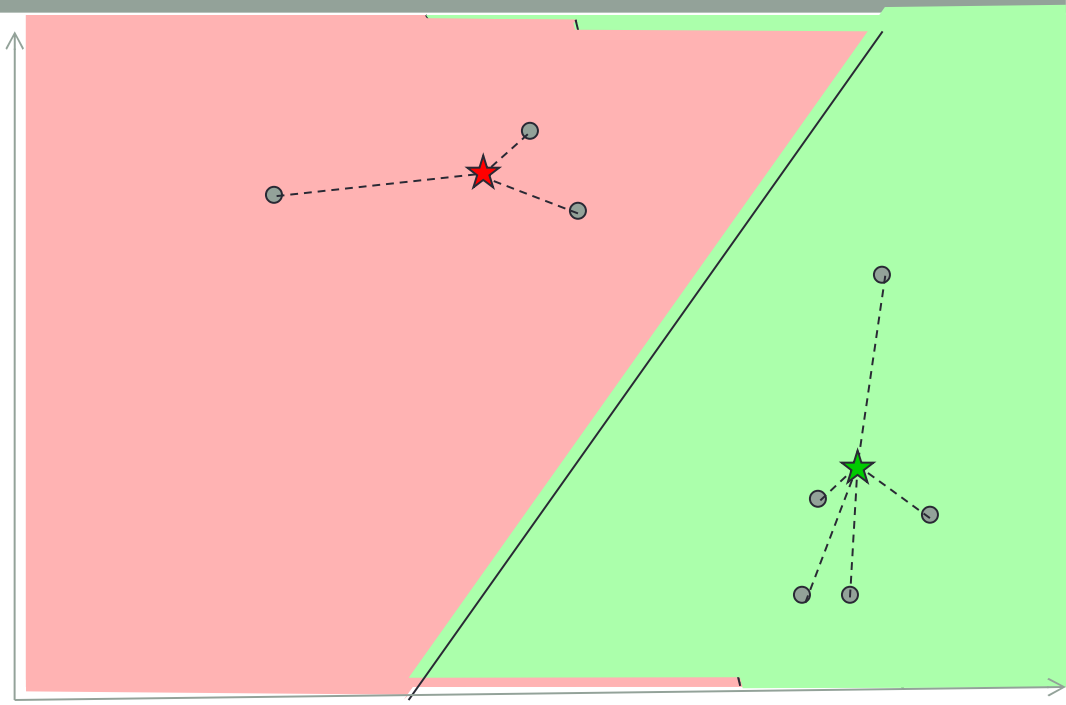
```
segments_fz = segmentation.felzenszwalb(img, scale=100, sigma=0.5,
min_size=50)

plt.imshow(segmentation.mark_boundaries(img, segments_fz))
```

- The parameter *scale* sets an observation level. Higher *scale* means less and larger segments.
- *sigma* is the diameter of a Gaussian kernel, used for smoothing the image prior to segmentation.

- The number of produced segments as well as their size can only be controlled indirectly through scale.
- Segment size within an image can vary greatly depending on local contrast.

K-means clustering

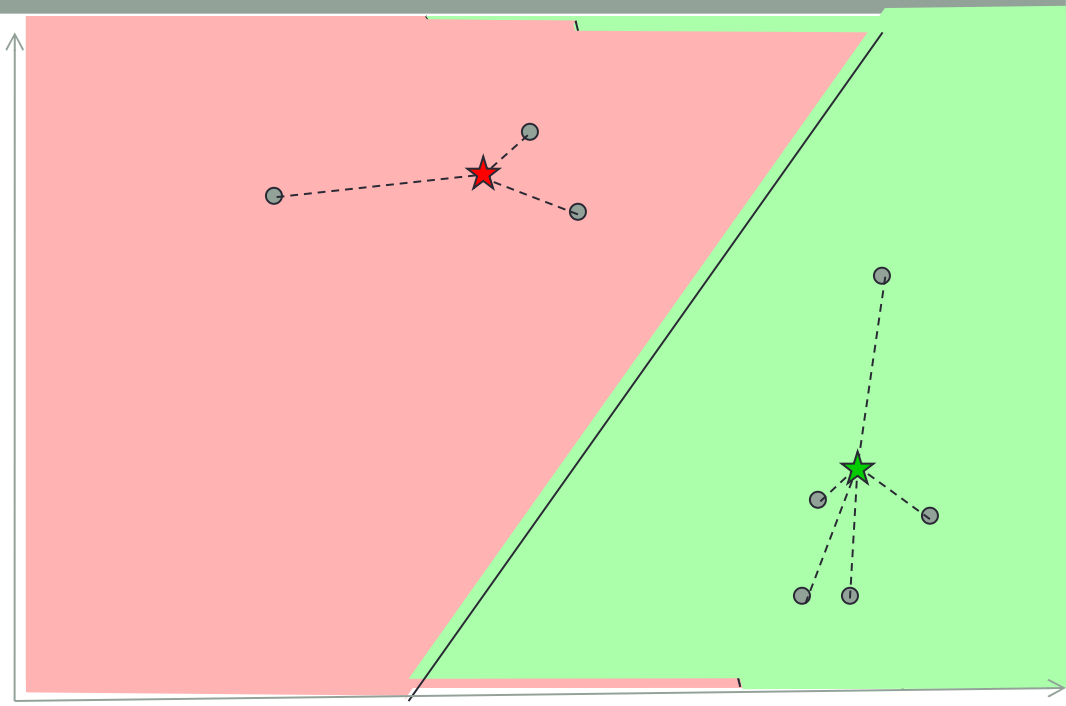


K-means clustering

Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

K-means clustering



Properties

- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

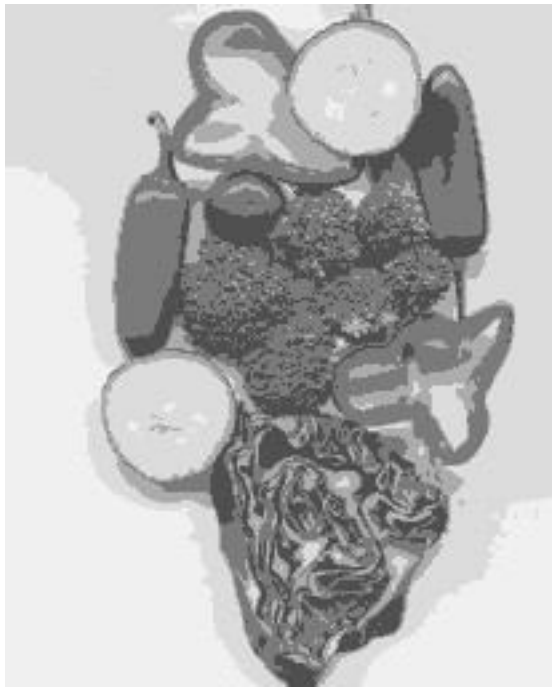
Source: Steve Seitz

K-means clustering

- K-means clustering based on intensity or color
  - Clusters do not have to be spatially coherent

Image

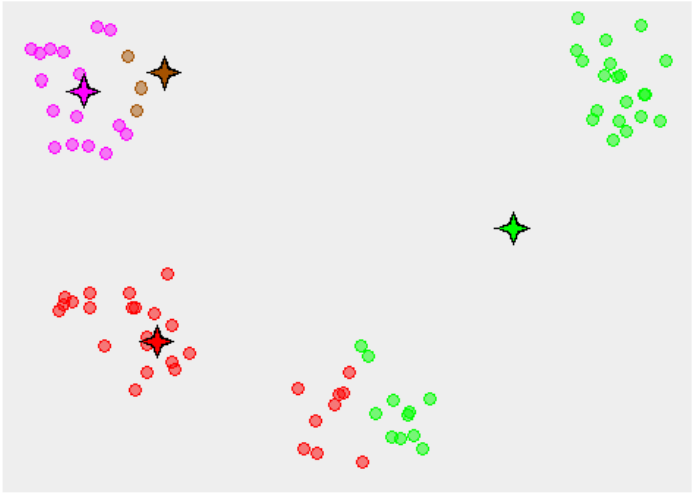
Intensity-based clusters



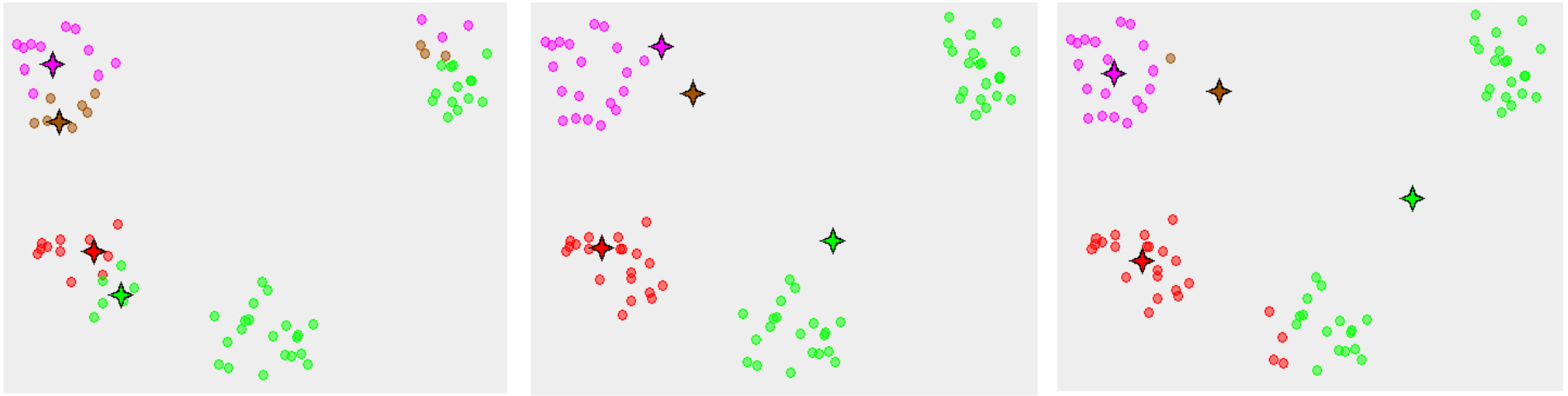
Color-based clusters

K-means: design choices

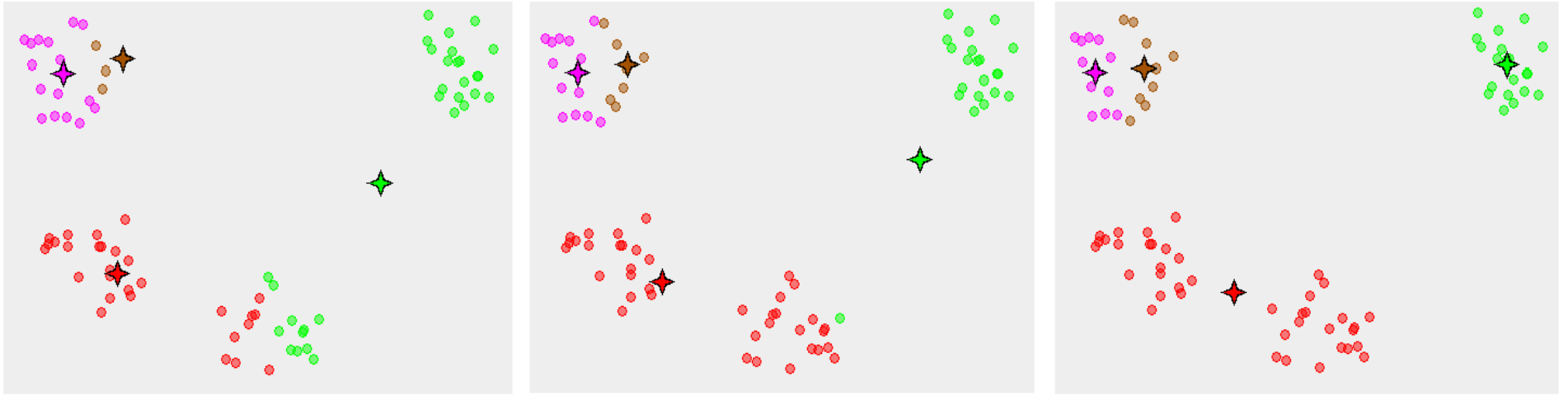
- Initialization
  - Randomly select K points as initial cluster center
- Distance measures
  - Traditionally Euclidean used, could be others
    - D((x1,y1),(x2,y2))=sqrt((x1-x2)^2+(y1-y2)^2)
- Optimization
  - Will converge to a *local minimum*
  - May want to perform multiple restarts



K-means converges to a local minimum



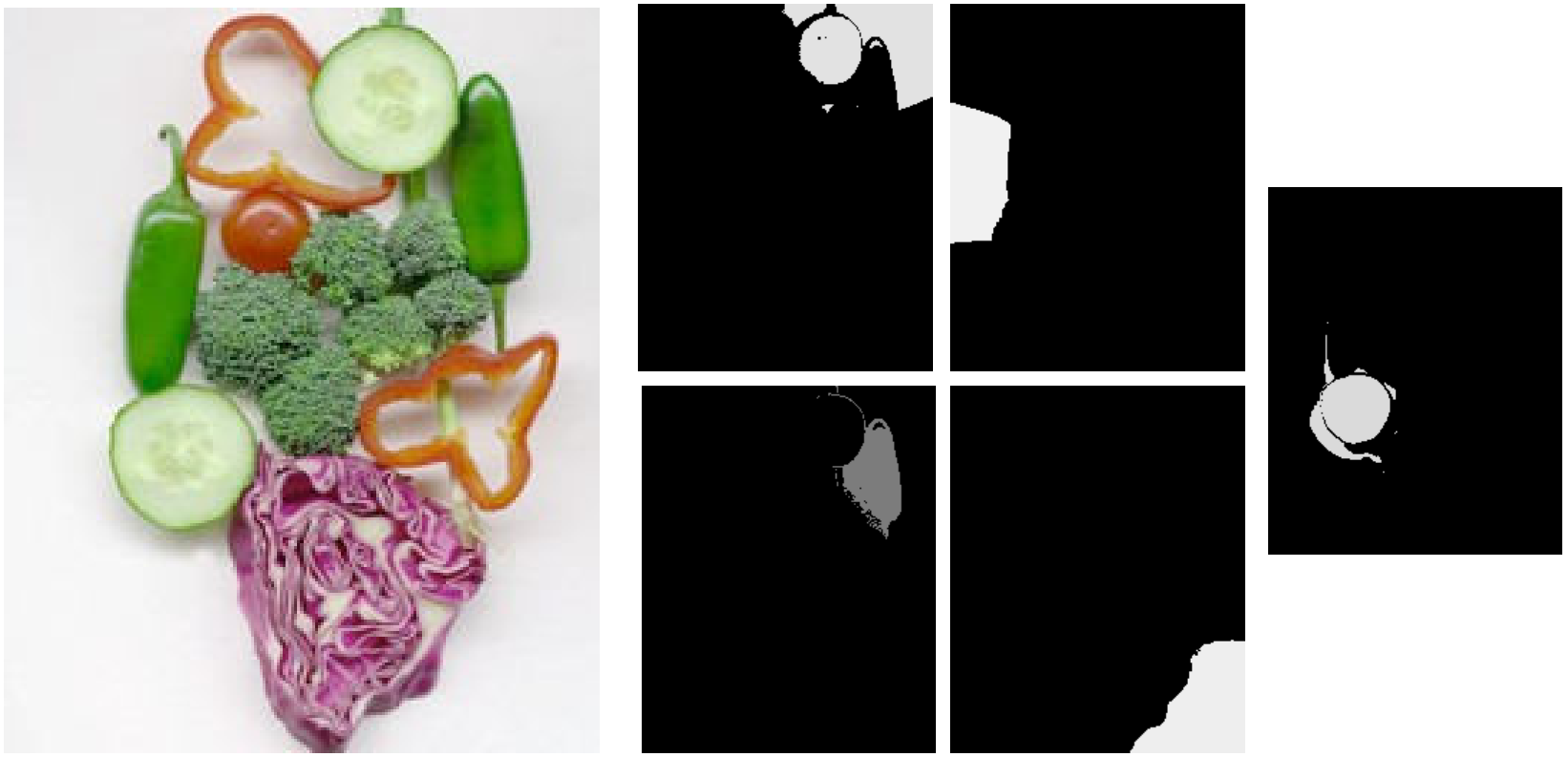
Initial seeds position



Final seeds position and segmentation

Segmentation as clustering

- Clustering based on (R,G,B,x,y) values enforces more spatial coherence



Simple linear iterative clustering (SLIC)

1. Smooth the image with a Gaussian (sigma)

2. Construct a 5-dimensional feature space based on color and spatial information

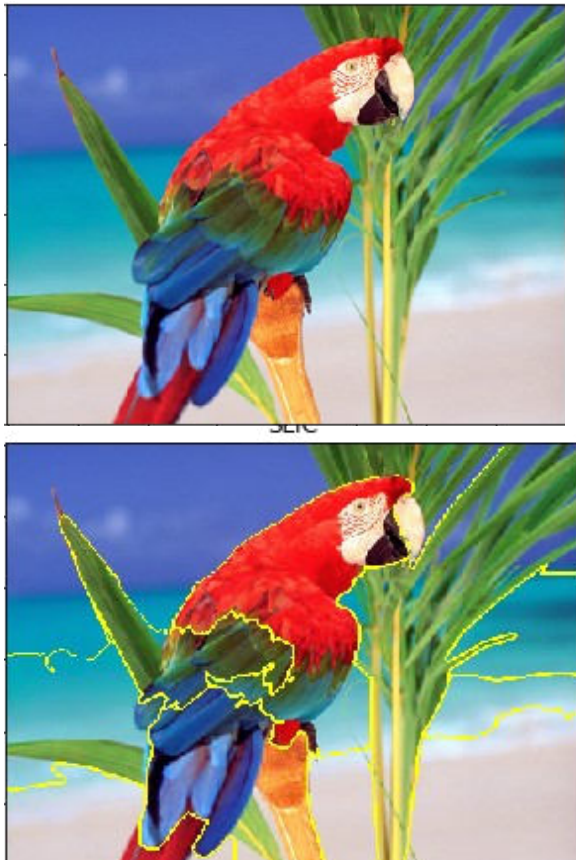
3. A novel distance measure enforces compactness and regularity in the superpixel shapes:

   - Higher values give more weight to space proximity, making superpixel shapes more square/cubic.

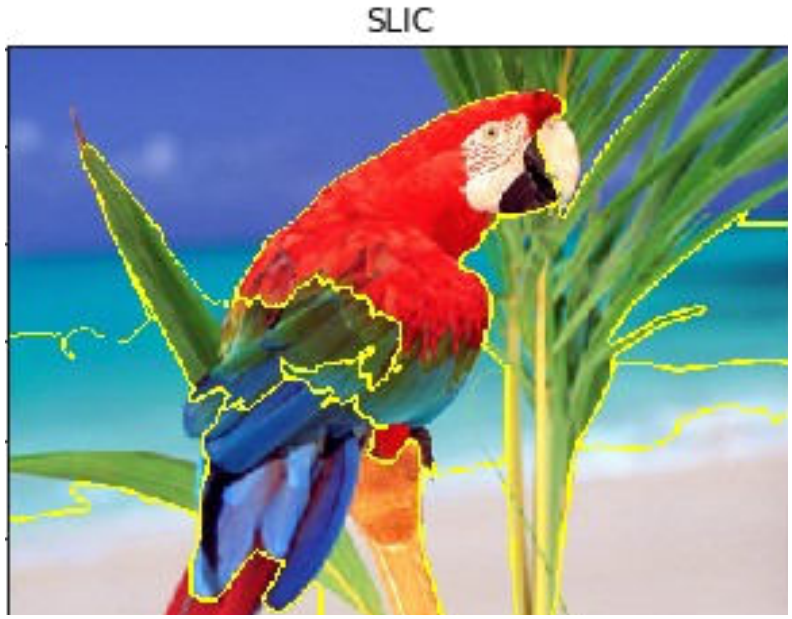
$$d(p1,p2)\_color = ((I1[i,j,0]-I2[i,j,0])^2 + (I1[i,j,1]-I2[i,j,1])^2 + (I1[i,j,2]-I2[i,j,2])^2),$$
$$p1=(i1,j1), p2=(i2,j2)$$

$$d(p1,p2)\_spatial = ((i1-i2)^2 + (j1-j2)^2)$$

$$d(p1,p2) = d\_color + m * d\_spatial$$



Simple linear iterative clustering (SLIC)



Skimage:

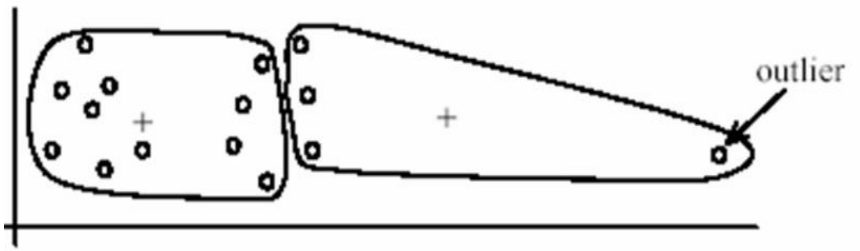
segments_slic = segmentation.slic(img, n_segments=20, compactness=0.1, sigma=1)
plt.imshow(segmentation.mark_boundaries(img, segments_slic))

http://www.kev-smith.com/papers/SLIC_Superpixels.pdf

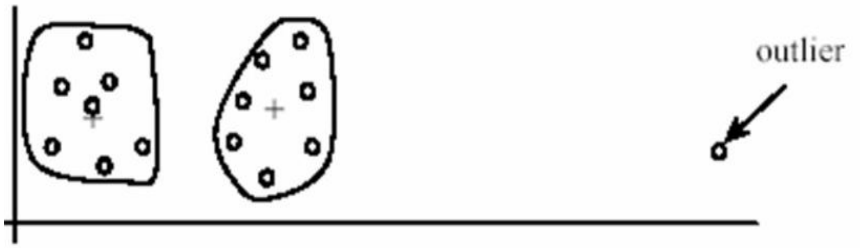
K-Means for segmentation

- Pros
  - Very simple method
  - Converges to a local minimum of the error function

- Cons
  - Memory-intensive
  - Need to pick K (number of seeds)
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds “spherical” clusters



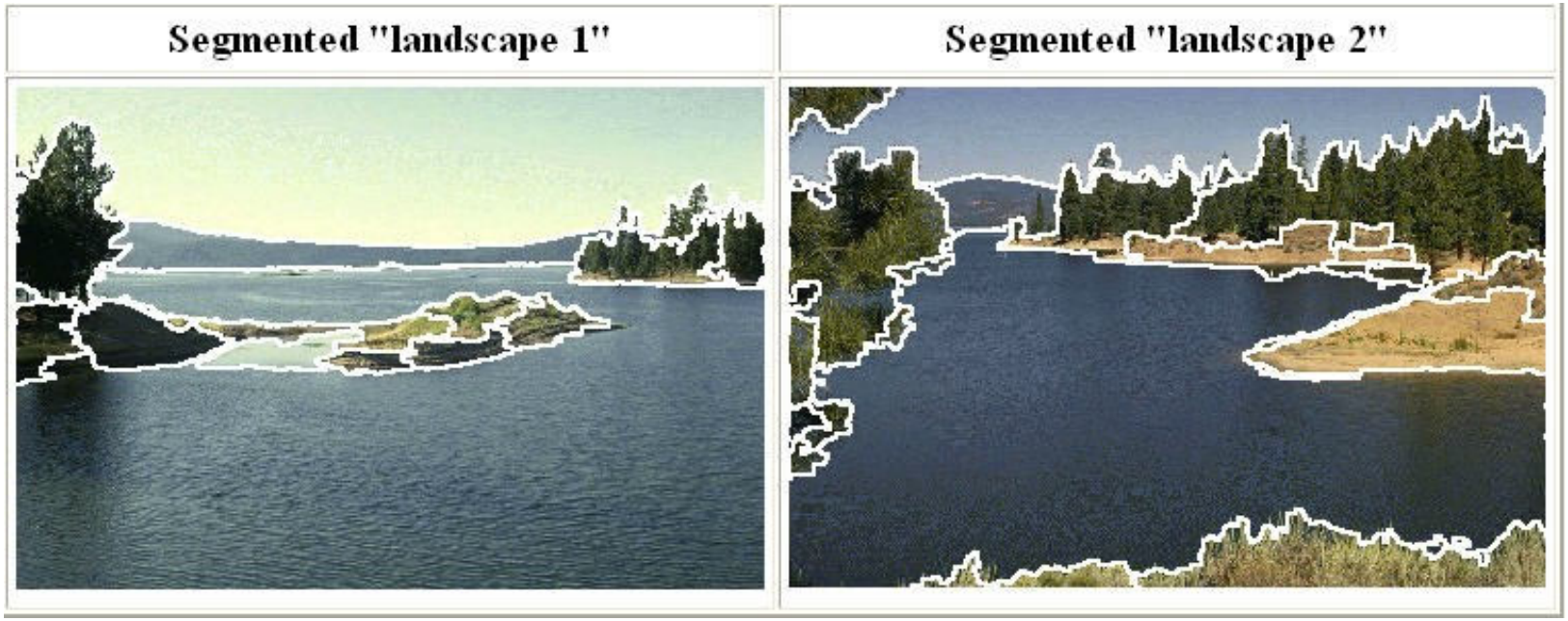
(A): Undesirable clusters



(B): Ideal clusters

Quickshift - Segmentation

- Perhaps one of the best techniques to date…



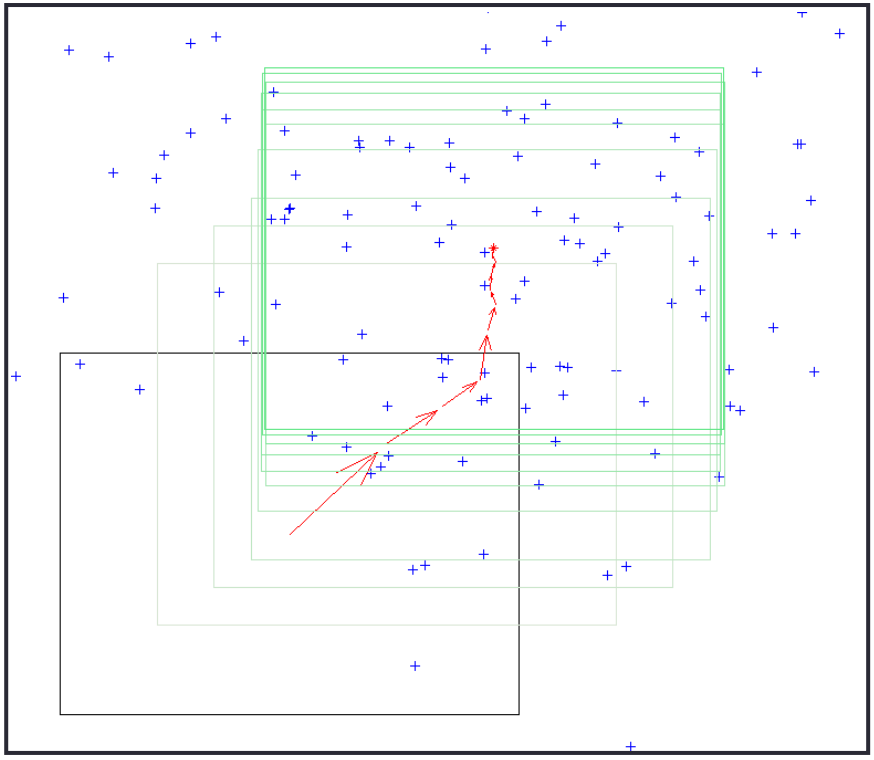
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

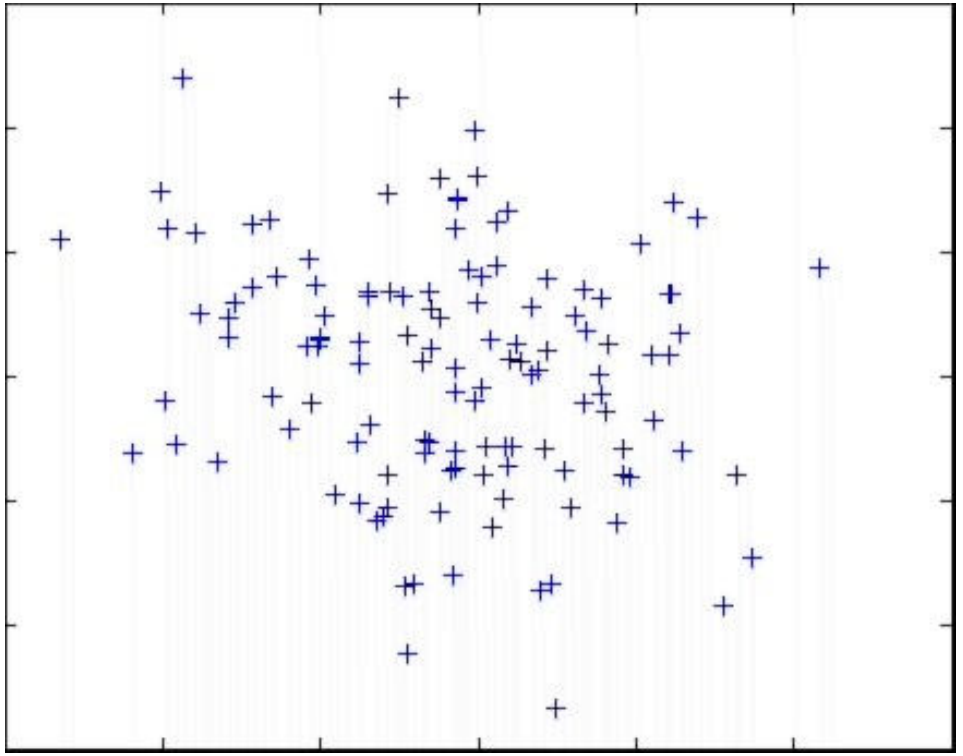
Quick-Shift Algorithm

Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the “mode” or point of highest density of a data distribution:





Quick-Shift Segmentation

Algorithm

1. Convert the image into tokens (via color, gradients, texture measures, etc.).
2. Choose initial search window locations uniformly in the data.
3. Compute the mean shift window location for each initial position.
4. Merge windows that end up on the same “peak” or mode.
5. The data these merged windows traversed are clustered together.

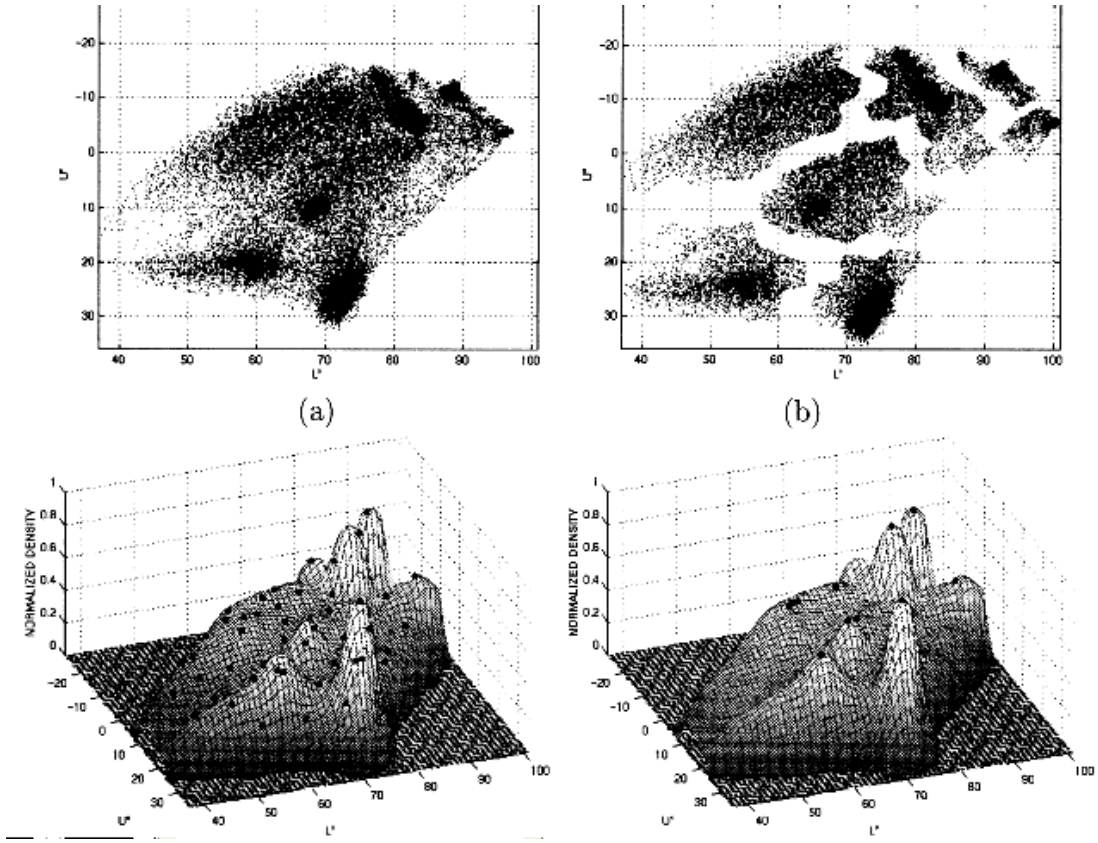
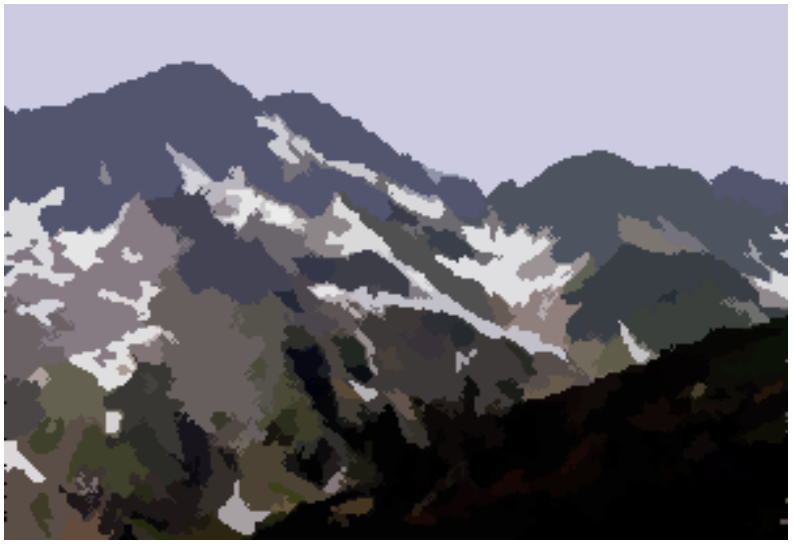


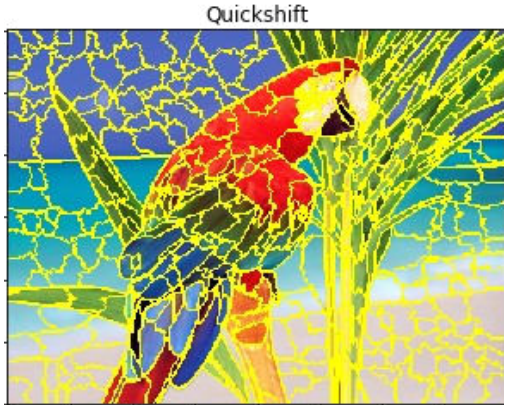
*Image From: Dorin Comaniciu and Peter Meer, Distribution Free Decomposition of Multivariate Data, Pattern Analysis & Applications (1999)2:22–30

Quick-Shift Segmentation results:



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Quick-shift



Skimage:

segments_quick = segmentation.quickshift(img, kernel_size=3, max_dist=6, ratio=0.5)

plt.imshow(segmentation.mark_boundaries(img, segments_quick))

- max_dist : float, optional, Cut-off point for data distances. Higher means fewer clusters.
- ratio : float, optional, between 0 and 1, Balances color-space proximity and image-space proximity. Higher values give more weight to color-space.

Summary

Segmentation – an open problem, very active currently
Different methods exist: k-means, etc., but still too simple…

We need to segment **highly dynamic and complex environments**:

- Segmentation is important, but without high-level knowledge it is very difficult!
- Last works combine: segmentation and object recognition.

Challenge: understanding complex environments