CS 4495 Computer Vision

Segmentation

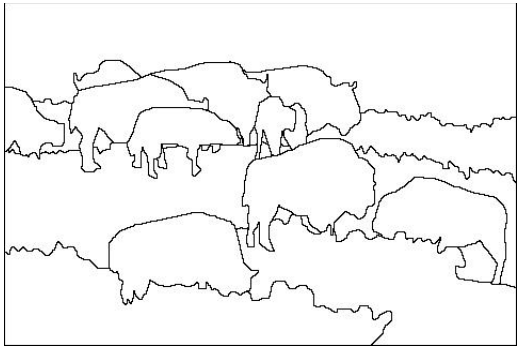
Aaron Bobick (slides by Tucker Hermans)
School of Interactive Computing

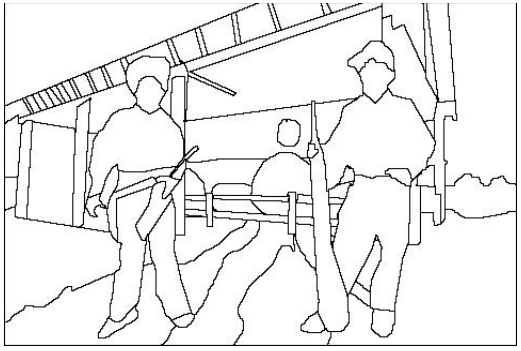
Administrivia

- PS 4: Out but I was a bit late so due date pushed back to Oct 29.
- OpenCV now has real SIFT again (the “notfree” packages). If using Python and OpenCV you should be able to use those calls.
- We’re still investigating SIFT for Python for those *not* using OpenCV.

Why segmentation?

Segmentation of Coherent Regions

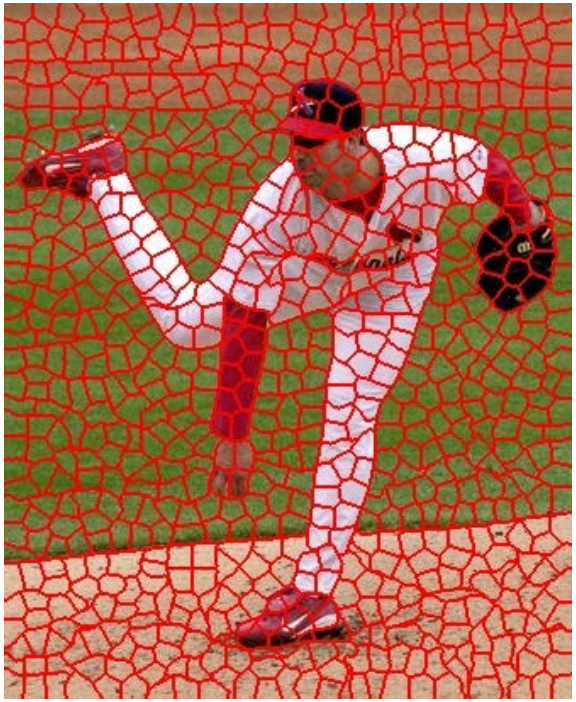



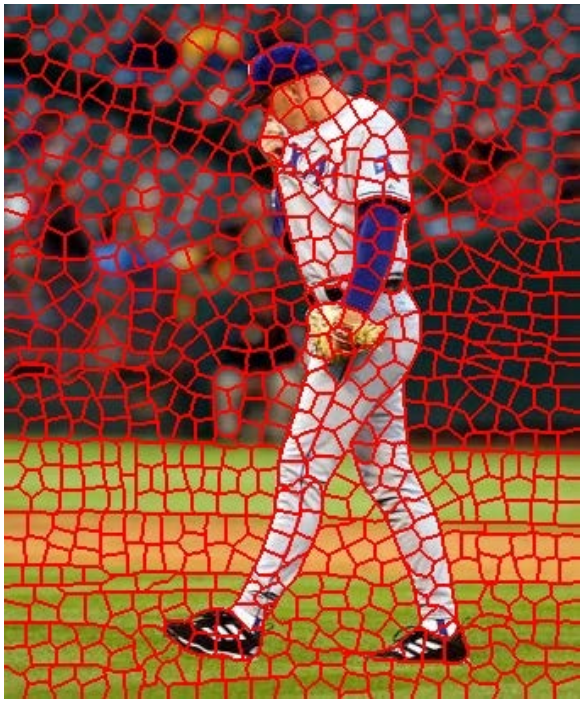
Berkeley segmentation database:

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

Slide by Svetlana Lazebnik

Grouping of Similar Neighbors





X. Ren and J. Malik. Learning a classification model for segmentation. ICCV 2003.

Slide by Svetlana Lazebnik

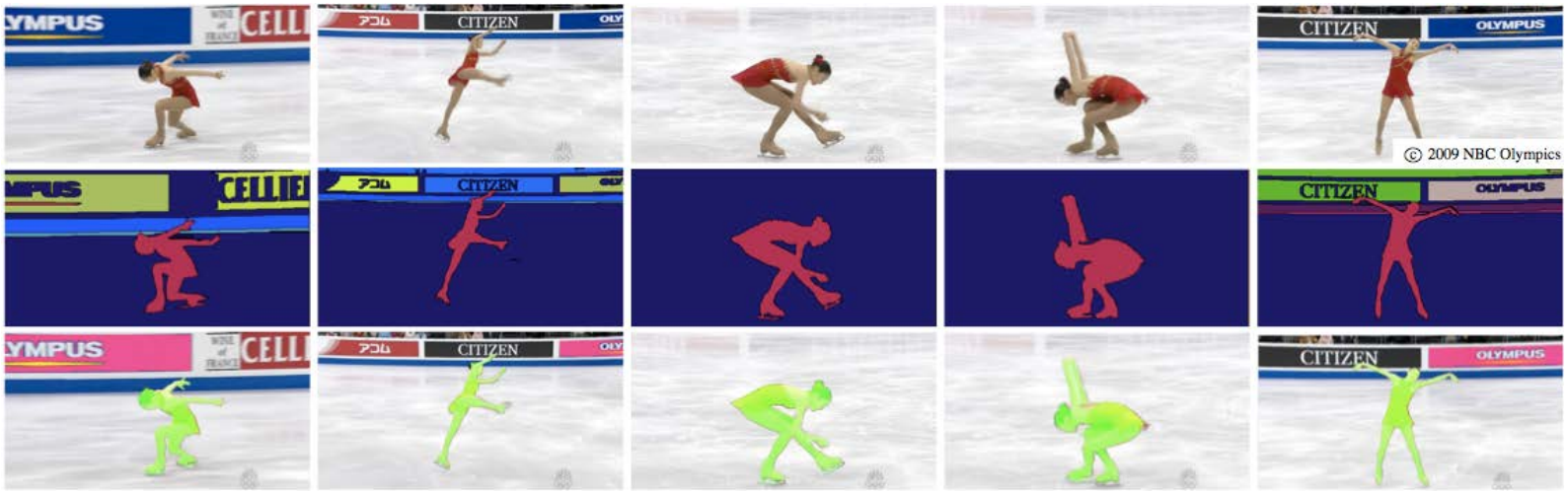
Figure Ground Segmentation

- Separate the foreground object (figure) from the background (ground)

D. Tsai, M. Flagg, and J. M. Rehg. “Motion Coherent Tracking with Multi-label MRF optimization.” BMVC 2010.

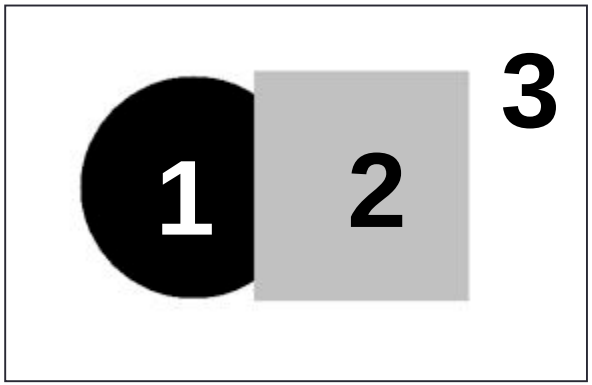
Extensions Beyond Single Images

J. Strom, A. Richardson, E. Olson. "Graph-based Segmentation for Colored 3D Laser Point Clouds." IROS 2010.

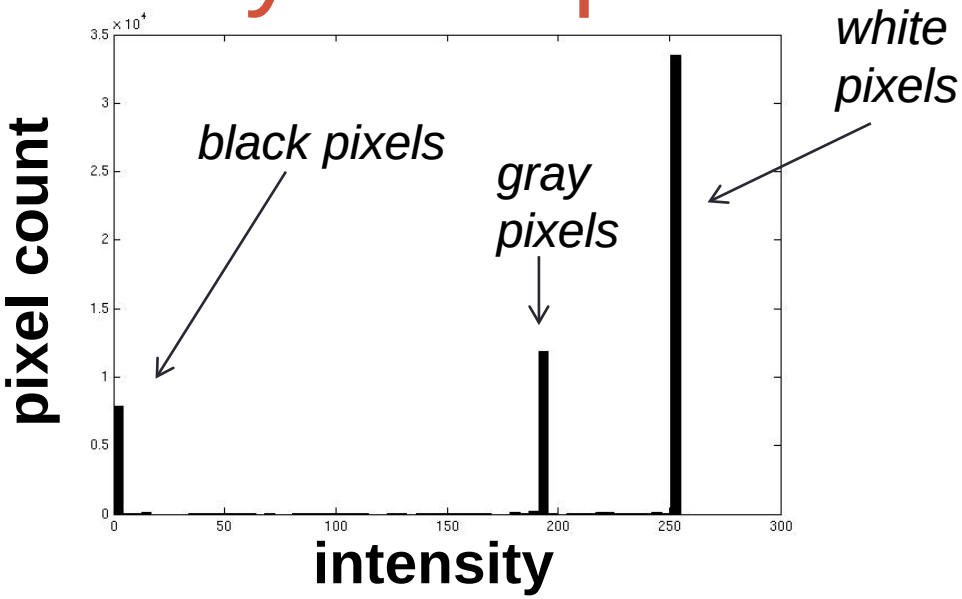


M. Grundmann, V. Kwatra, M. Han, I. Essa. "Efficient Hierarchical Graph-Based Video Segmentation." CVPR 2010.

Image segmentation: toy example

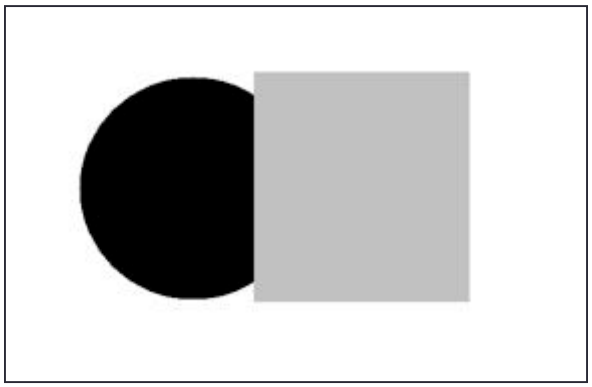


input image

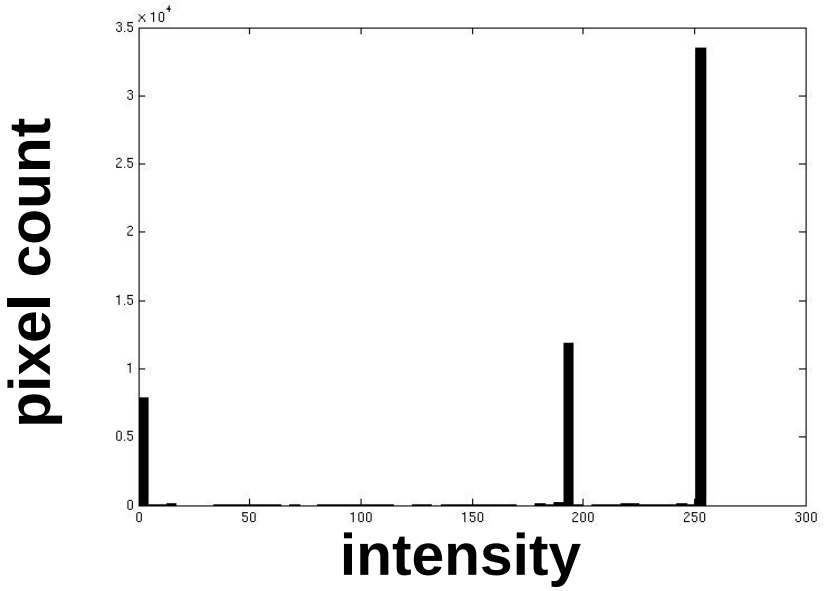


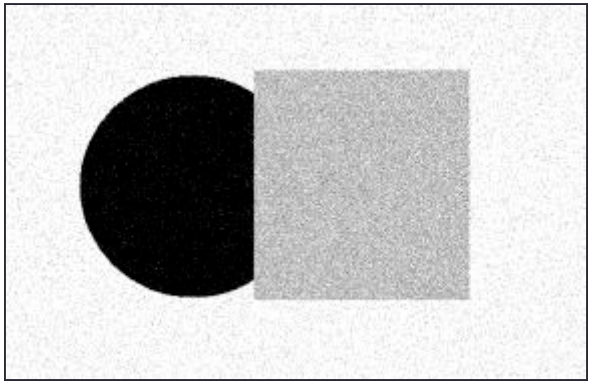
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Noisy Images

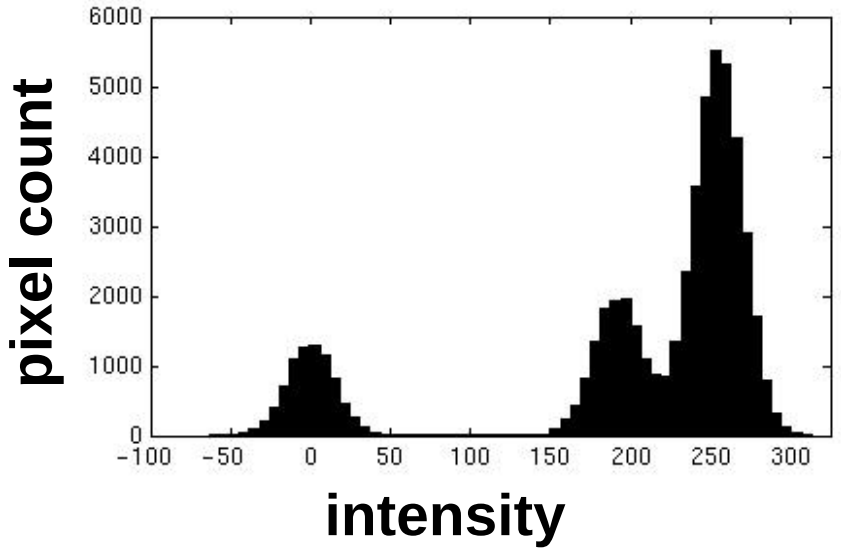


input image

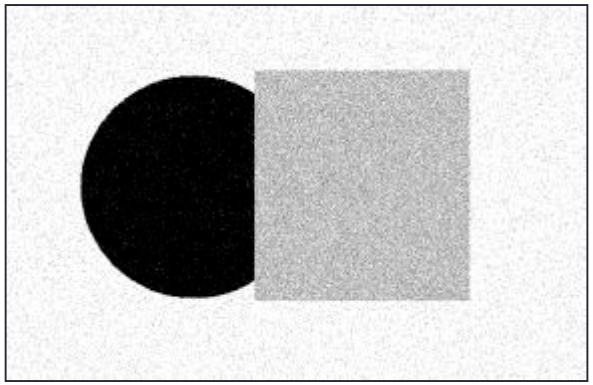




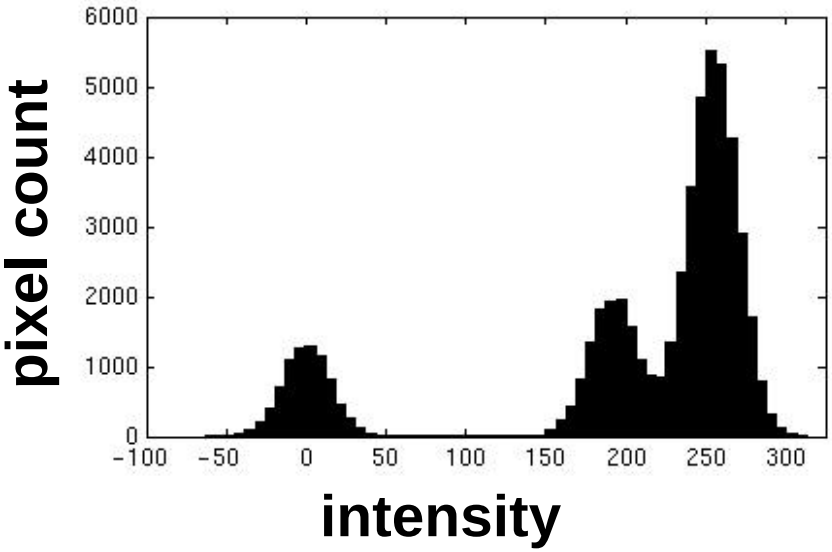
input image



Noisy Images

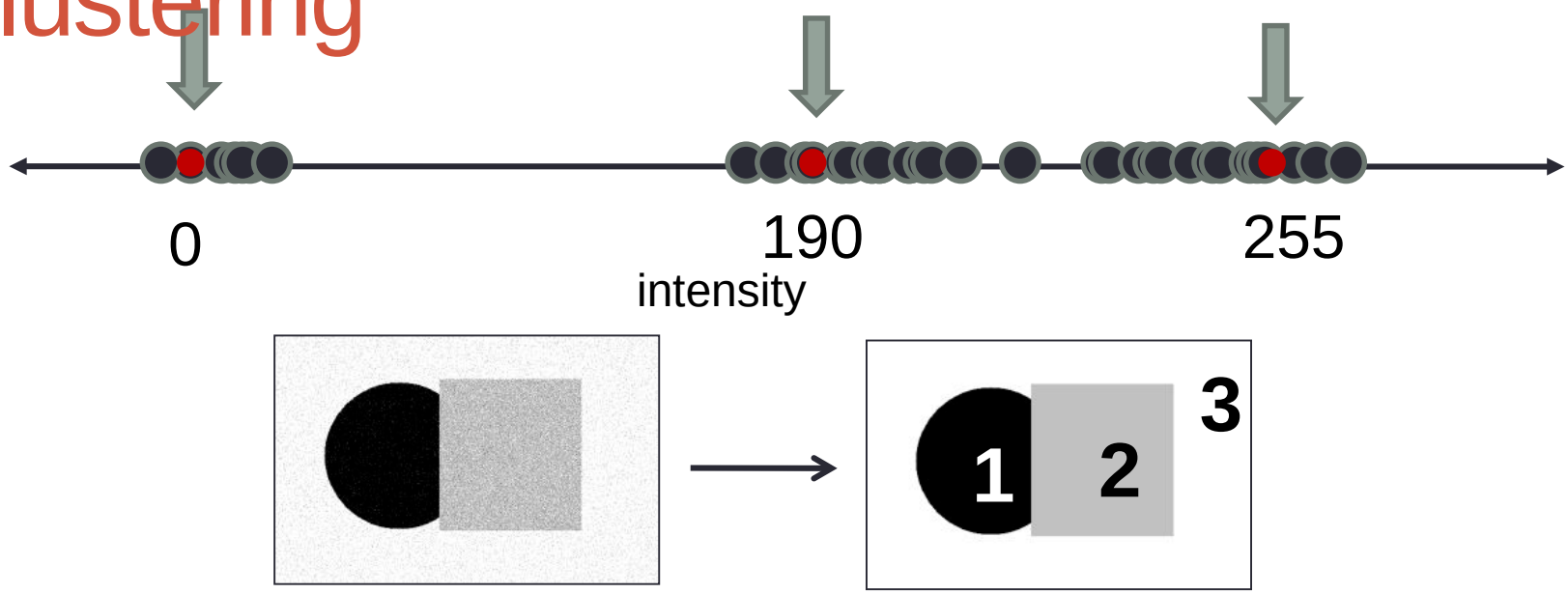


input image



- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***

Clustering

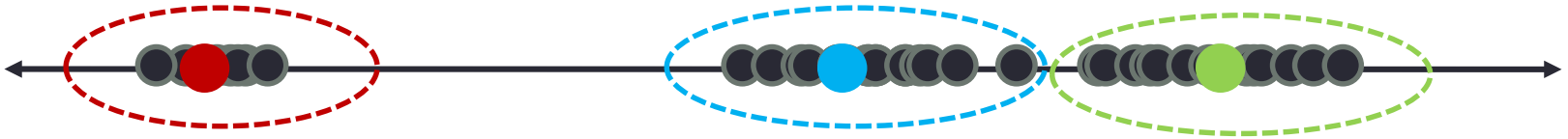


- Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center ci:

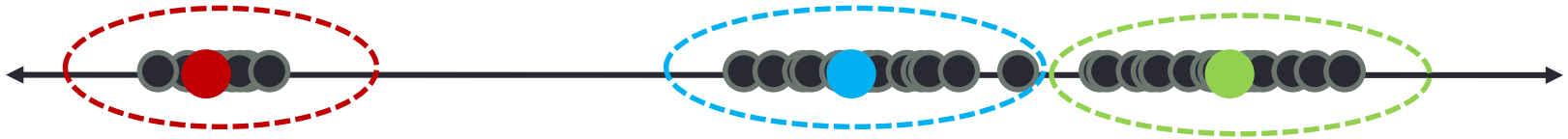
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Clustering

- With this objective, it is a "chicken and egg" problem:
  - Q: how to determine which points to associate with each **cluster center**, c_i?
  - A: for each point p, choose closest c_i



  - Q: If we knew the **group memberships**, how do we get the centers?
  - A: choose c_i to be the mean of all points in the cluster

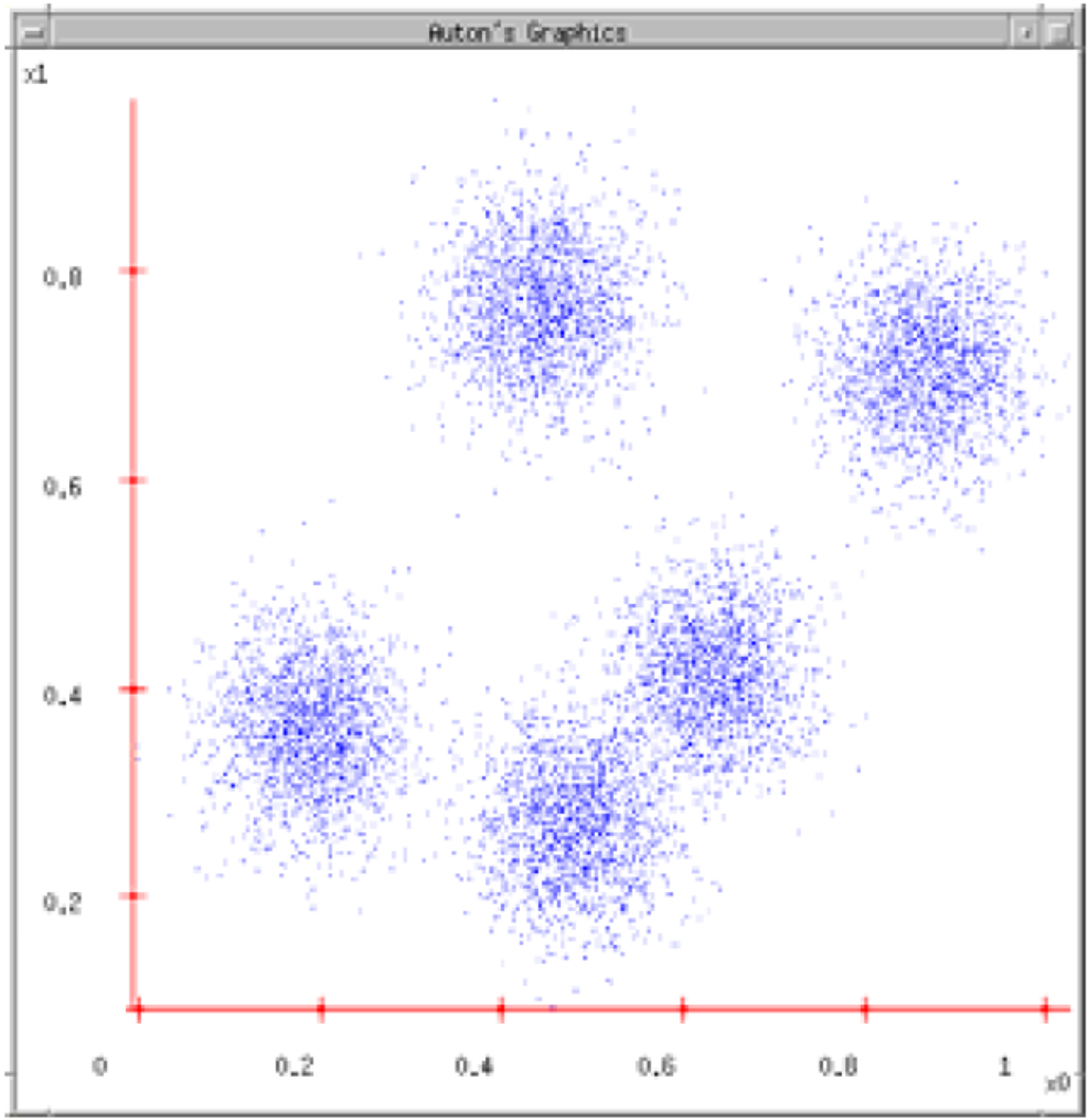


K-means clustering: Algorithm

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 ci
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

K-means

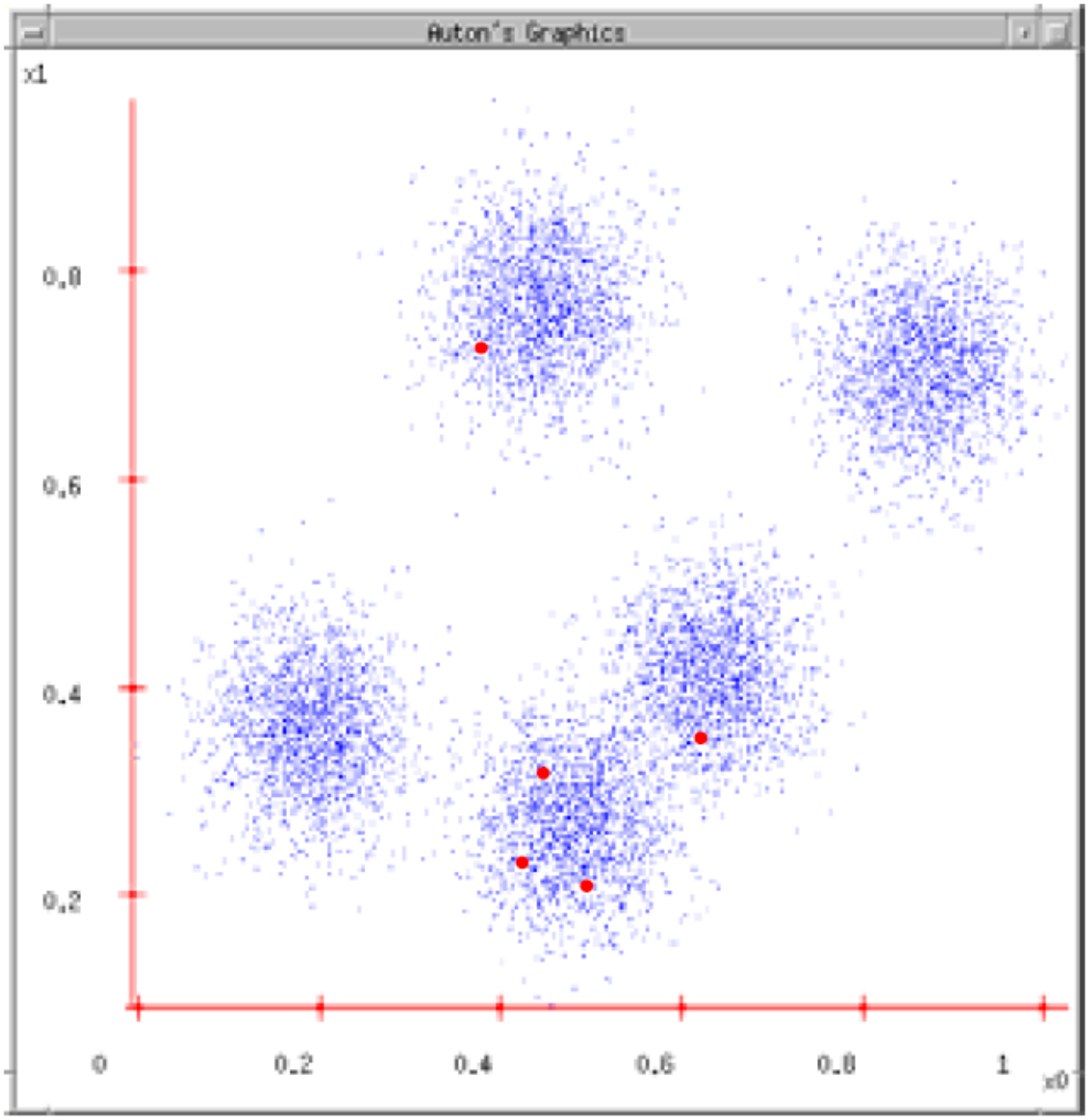
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Andrew Moore

K-means

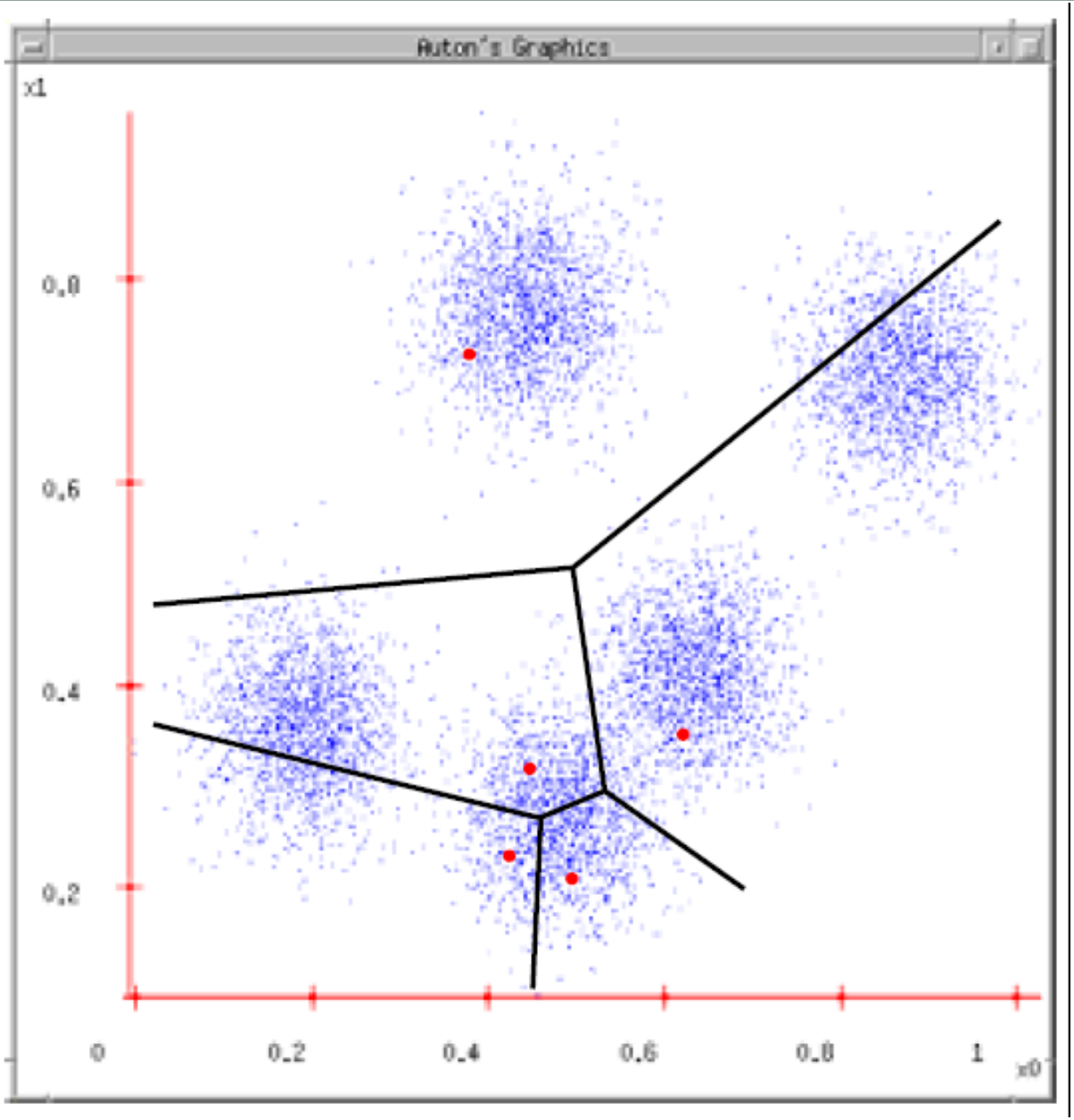
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Andrew Moore

K-means

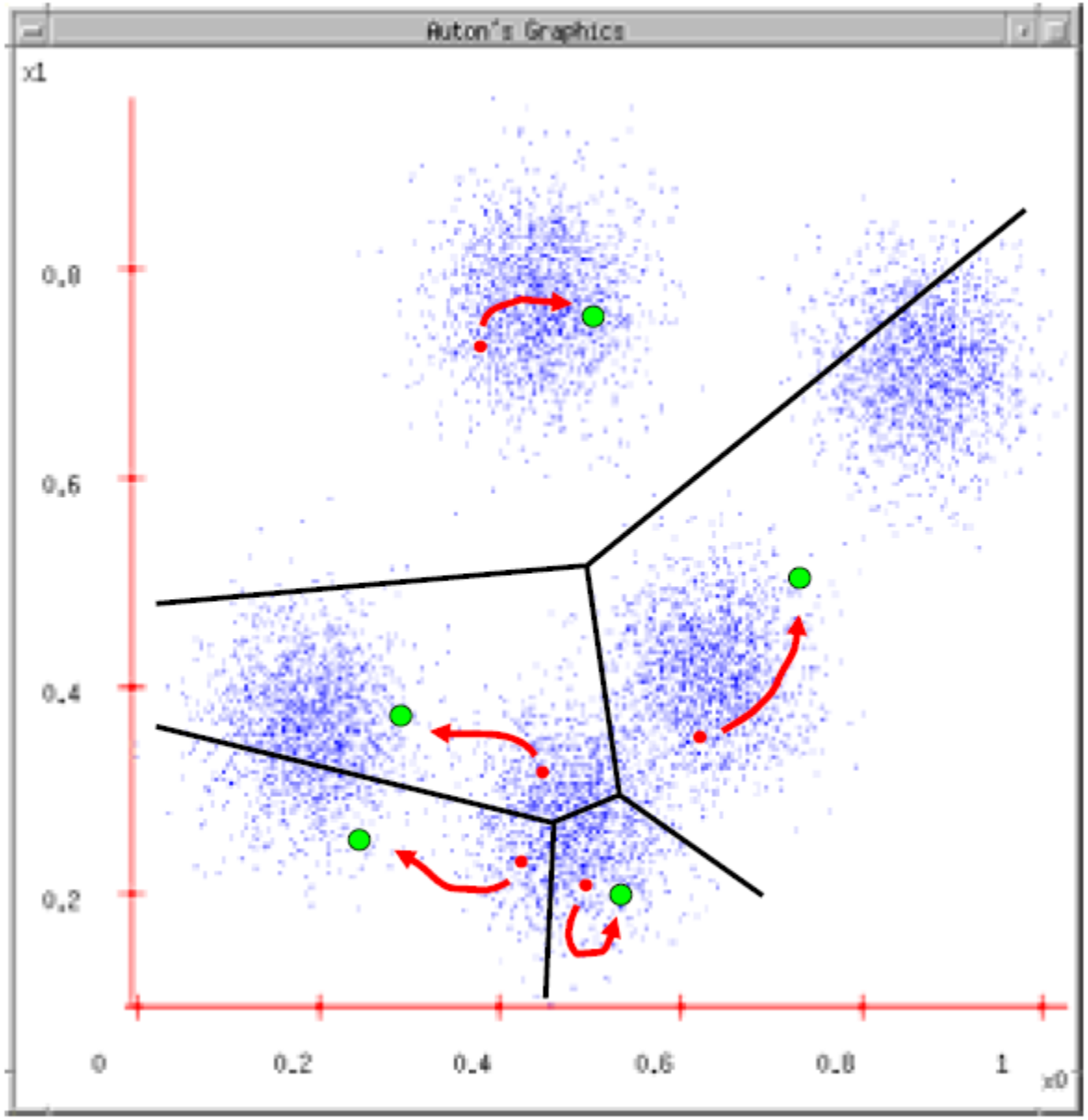
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Andrew Moore

K-means

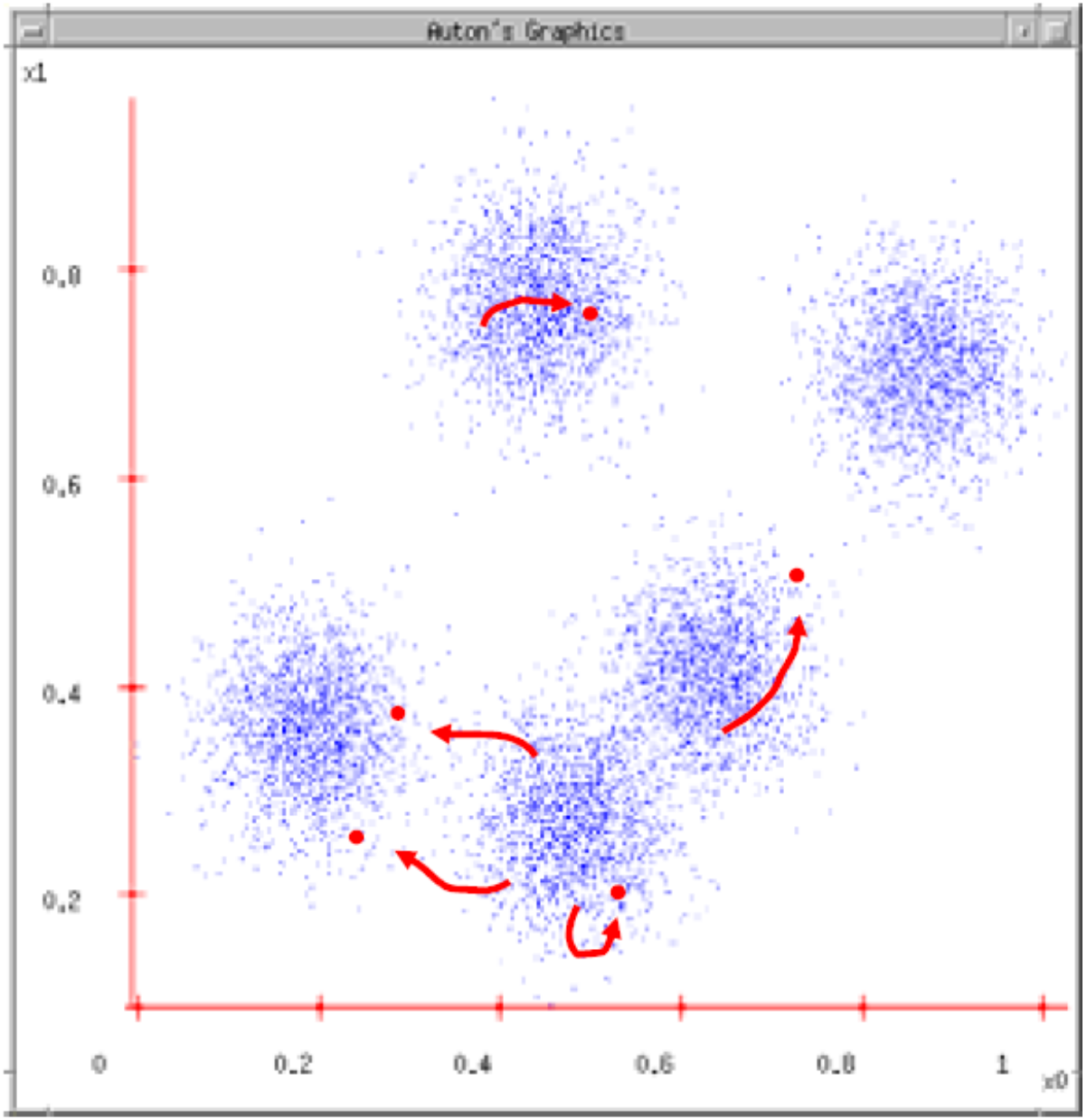
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Andrew Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Andrew Moore

Segmentation as clustering

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

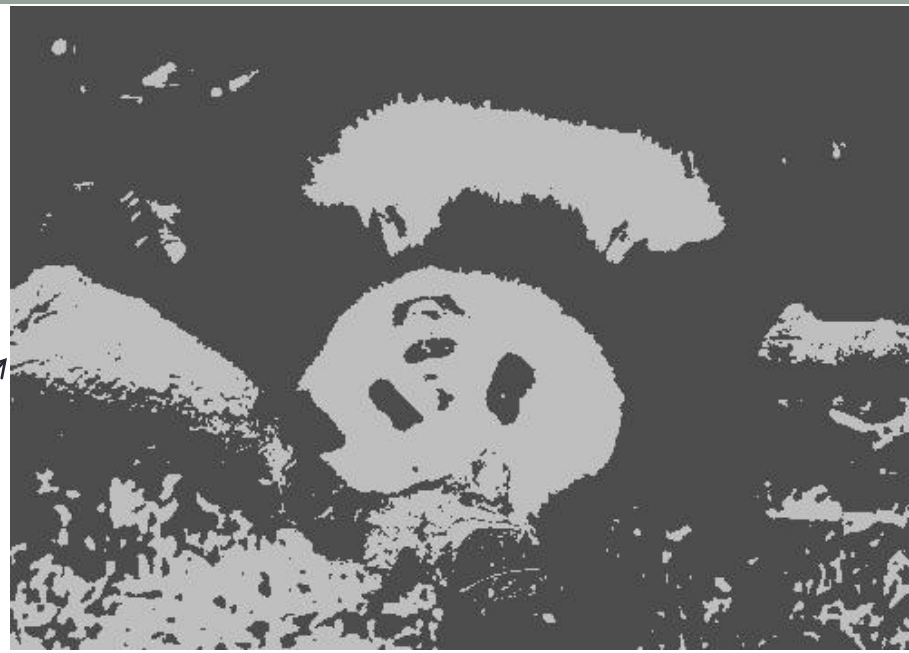
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

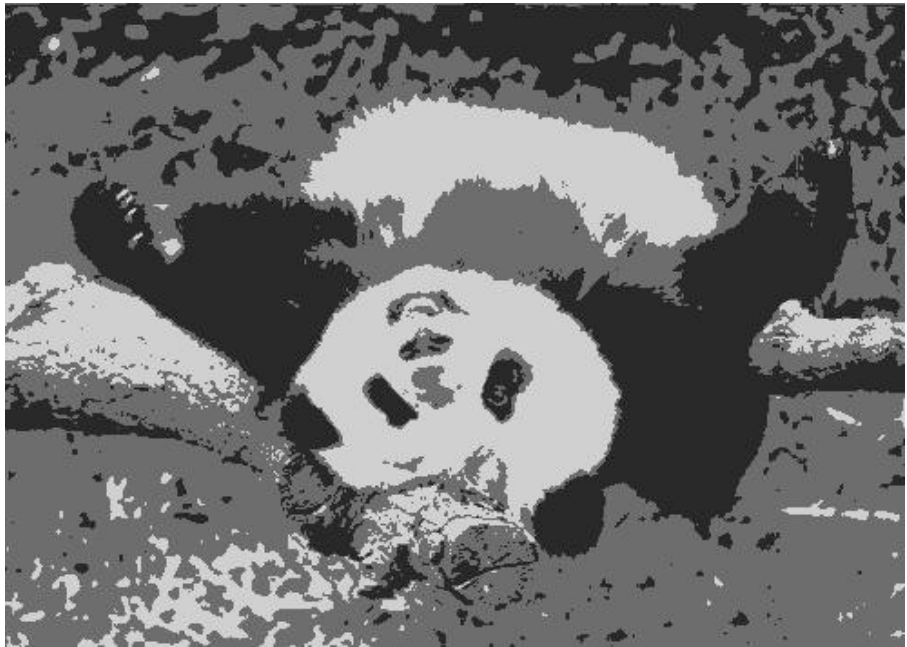
Source: K. Grauman

Number of Clusters

K=2



K=3

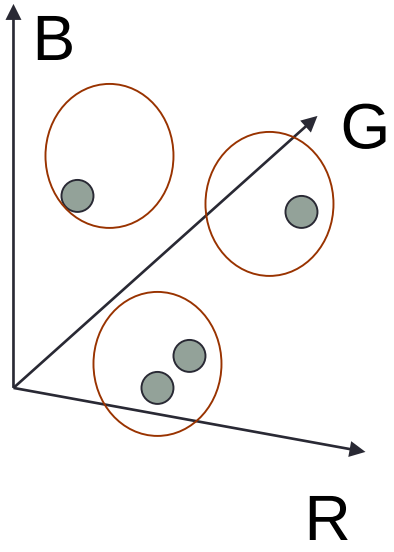


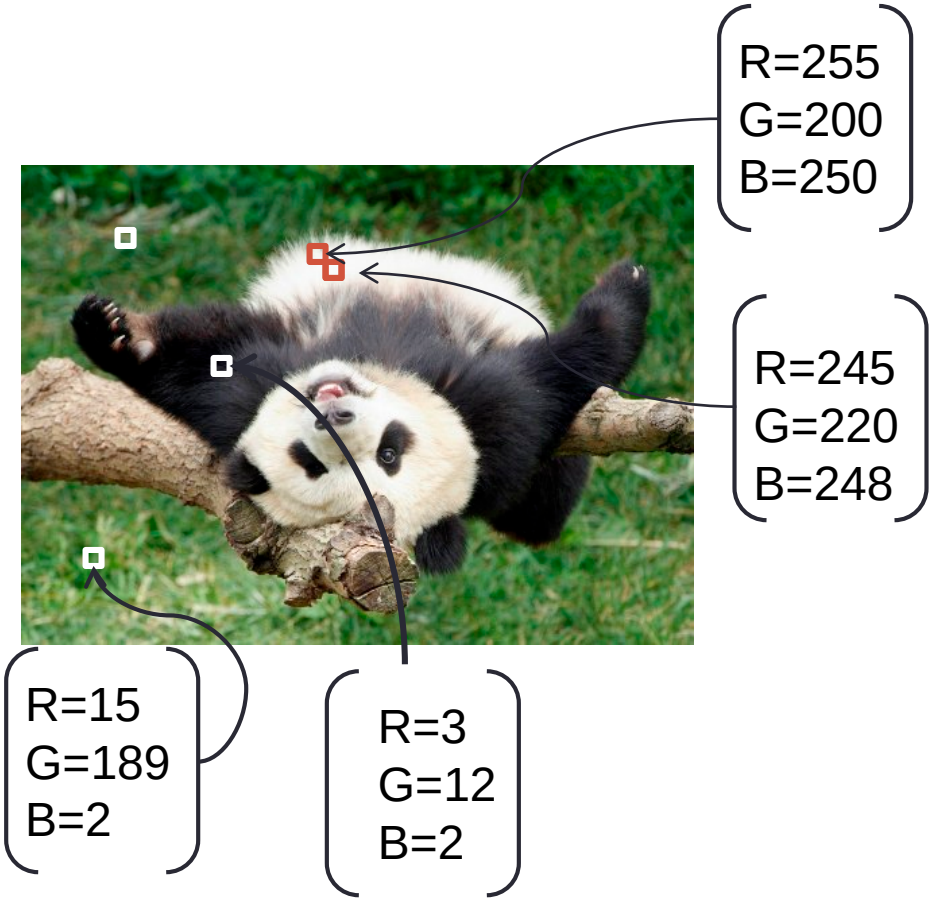
quantization of the feature space; segmentation label map

Segmentation as clustering

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

Grouping pixels based on **color** similarity





Feature space: color value (3-d)

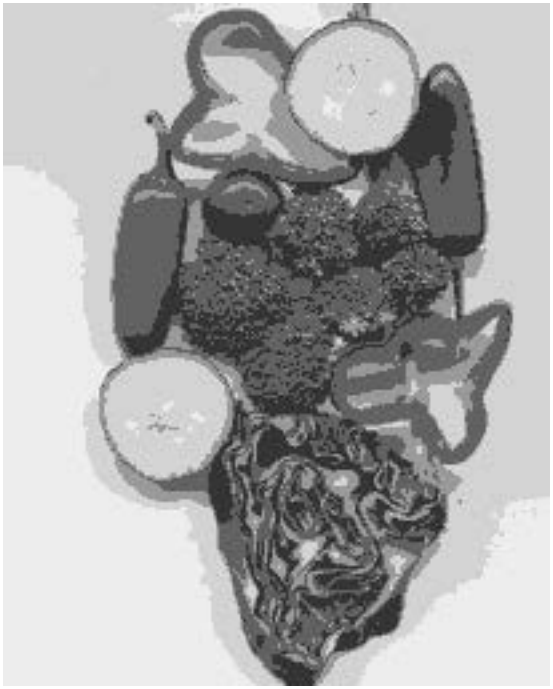
Kristen Grauman

Segmentation as clustering

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes

Image

Intensity-based clusters



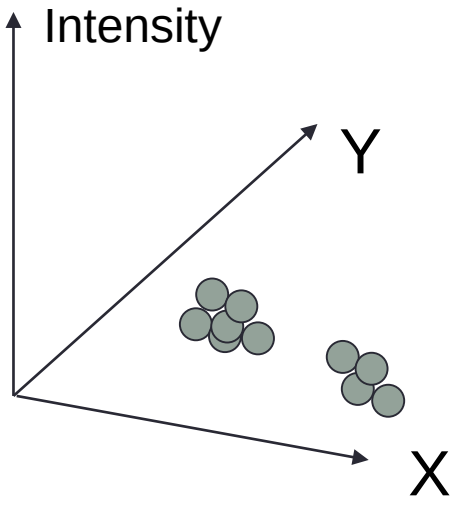
Color-based clusters

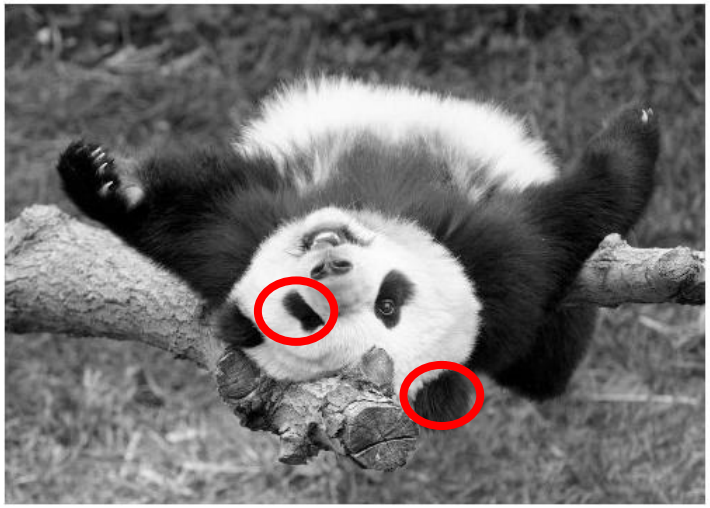
Slide by Svetlana Lazebnik

Segmentation as clustering

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

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

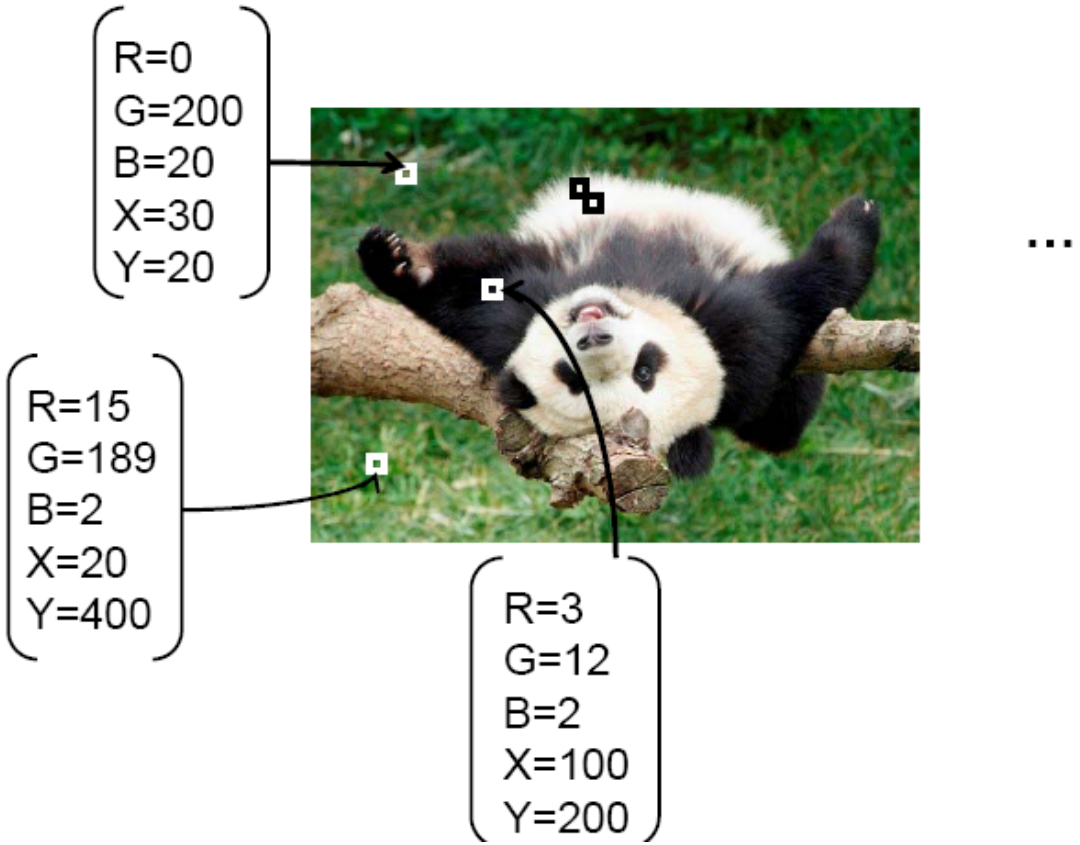




Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Segmentation as clustering

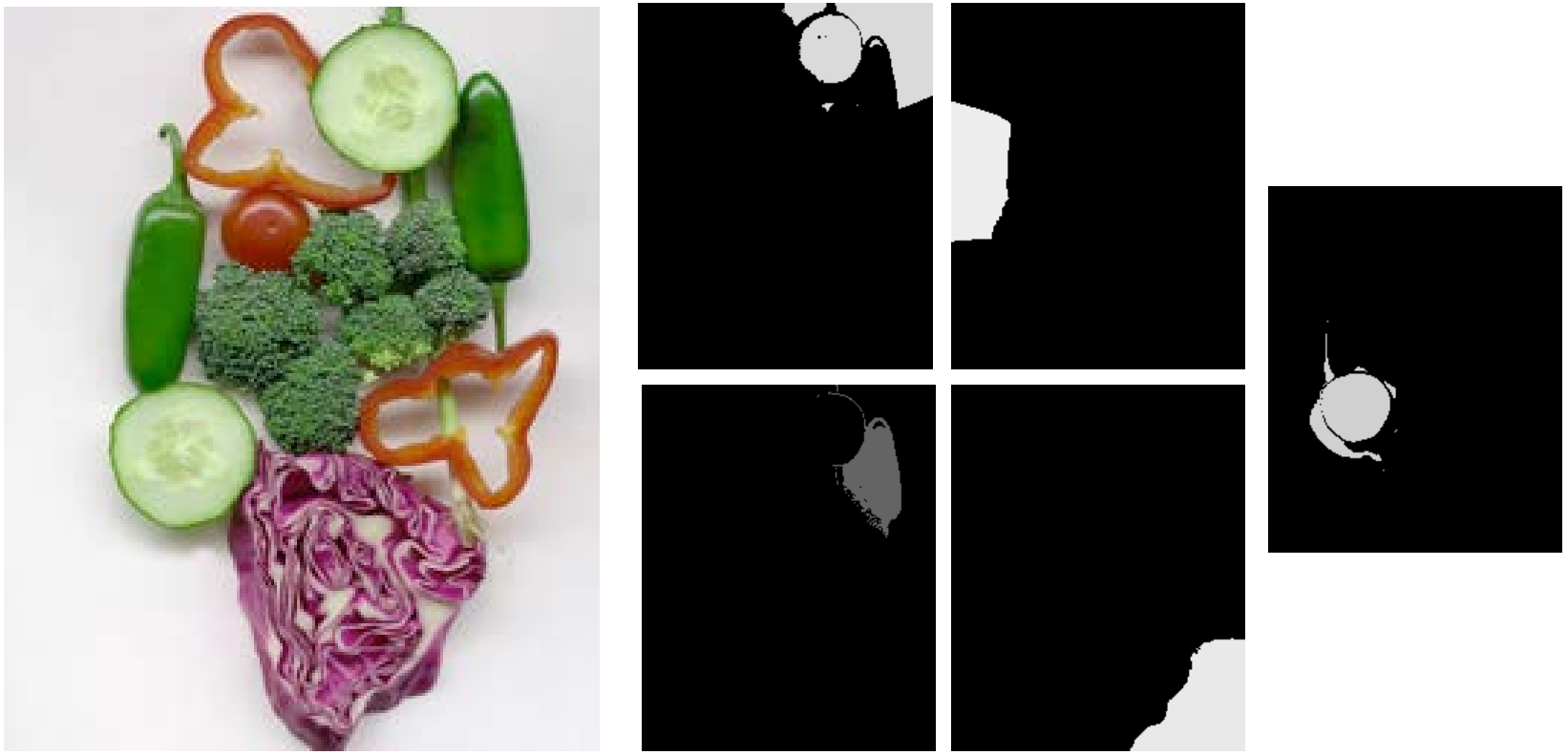
- Cluster similar pixels (features) together



Source: K. Grauman

Segmentation as clustering

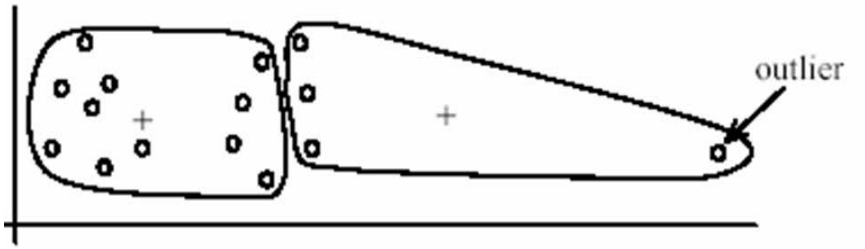
- Clustering based on (r,g,b,x,y) values enforces more spatial coherence



Slide by Svetlana Lazebnik

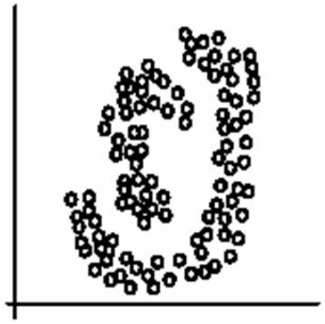
K-Means for segmentation

- Pros
  - Very simple method
  - Converges to a local minimum of the error function
- Cons
  - Memory-intensive
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds “spherical” clusters

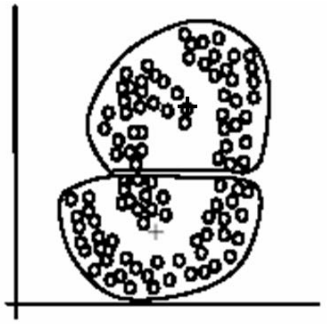


(A): Undesirable clusters

(B): Ideal clusters



(A): Two natural clusters

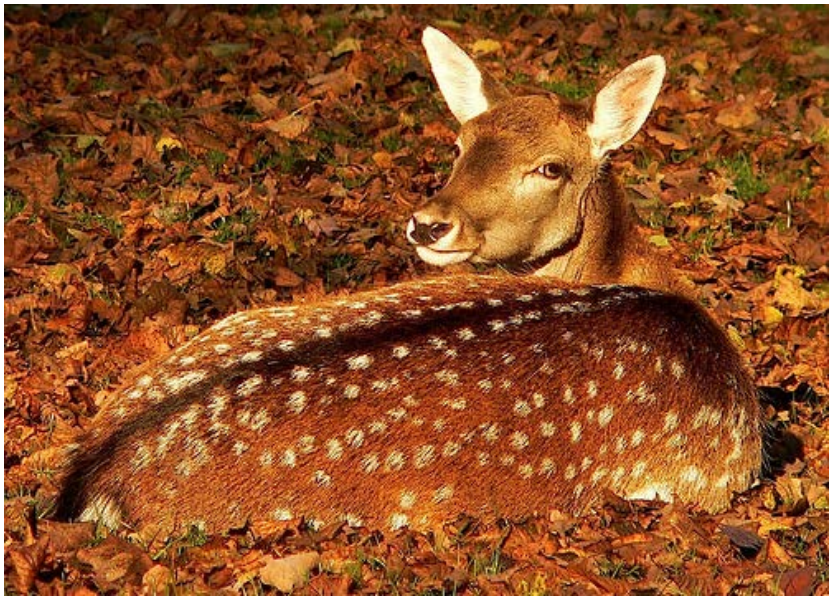


(B): *k*-means clusters

Slide by Svetlana Lazebnik

Segmentation as clustering

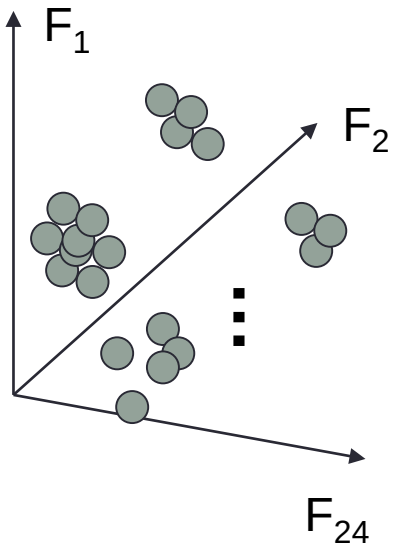
- Color, brightness, position alone are not enough to distinguish all regions…

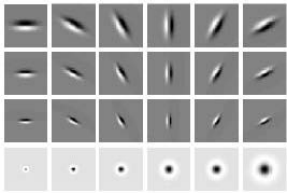


Segmentation as clustering

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

Grouping pixels based on **texture** similarity

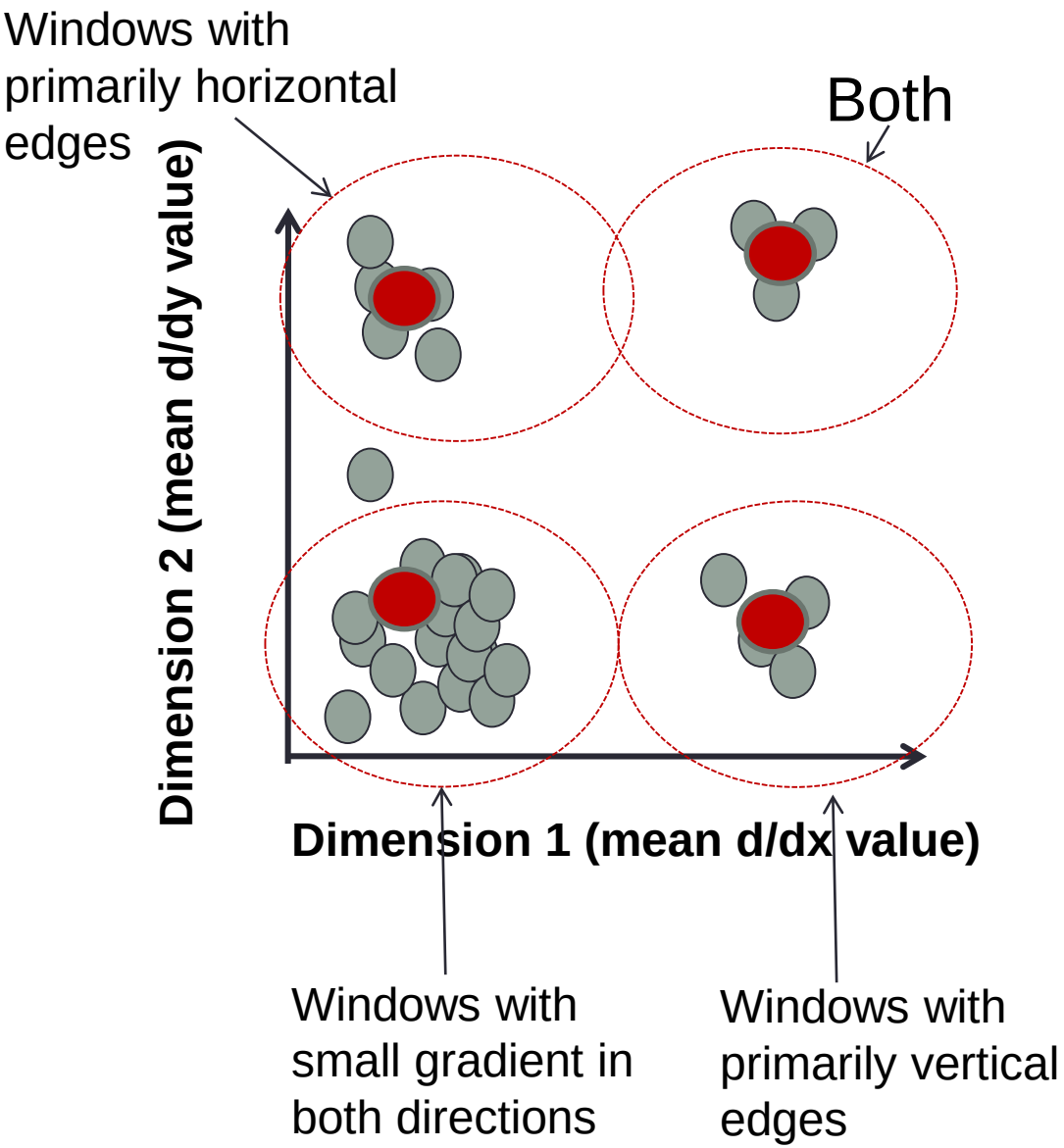




Filter bank of 24 filters

Feature space: filter bank responses (e.g., 24-d)

Aside: Texture representation example



	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

⋮

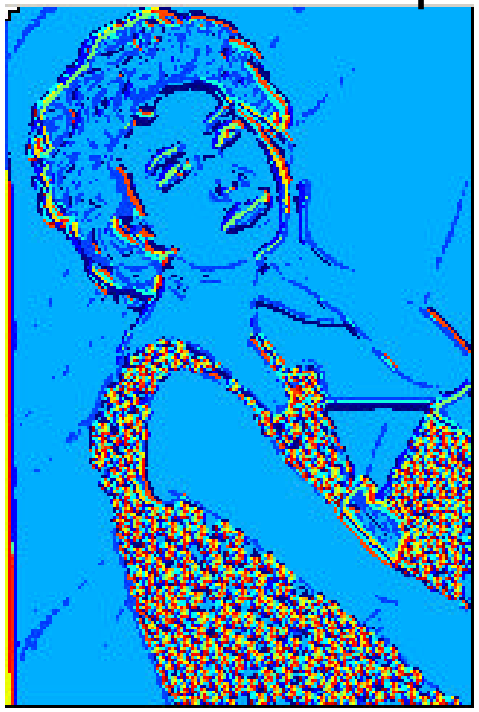
statistics to summarize patterns in small windows

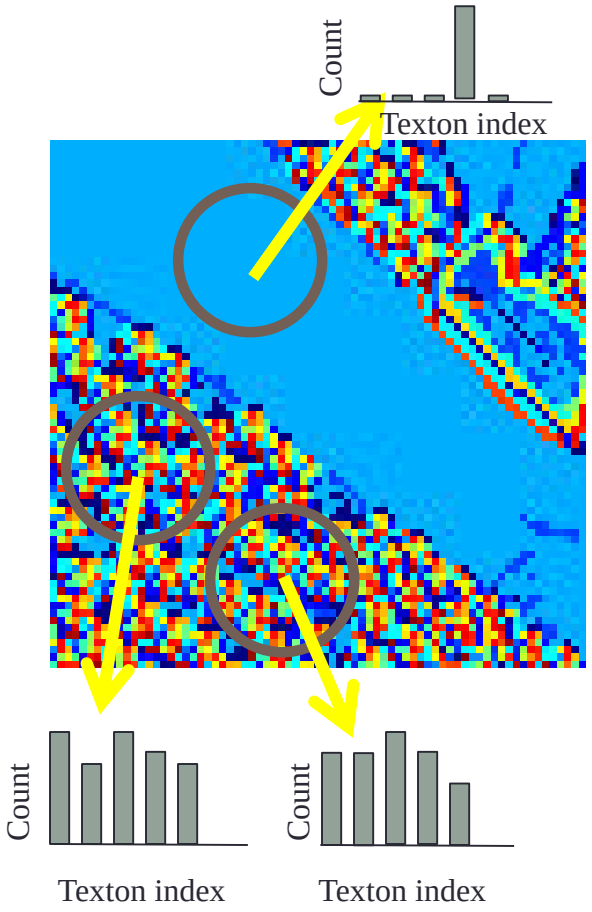
Aside: Texture features

- Find “textons” by **clustering** vectors of filter bank outputs
- Describe texture in a window based on its *texton histogram*

Image

Texton map

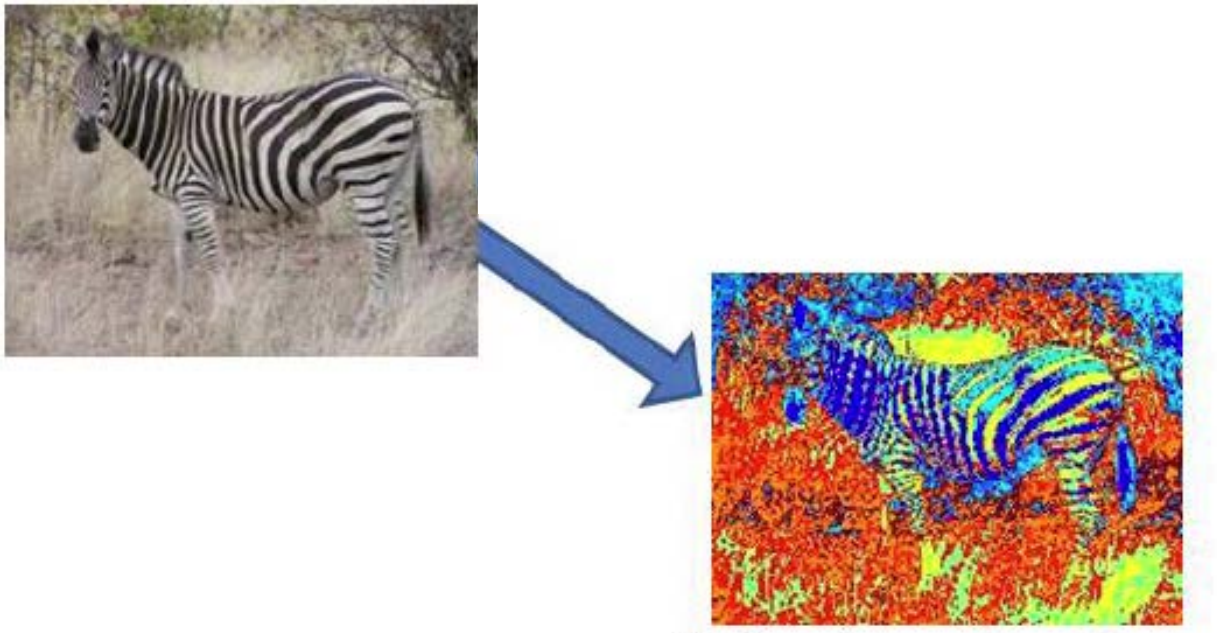




Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

Image segmentation example



Kristen Grauman

Make it better

- *K-means* heavily sensitive to initial conditions and (typically) need to know *K* in advance.
- Suppose we assume that there are a few *modes* in the image and that all the pixels come from these modes.
- If you could find the modes you might be able to segment the image.

Mean shift algorithm

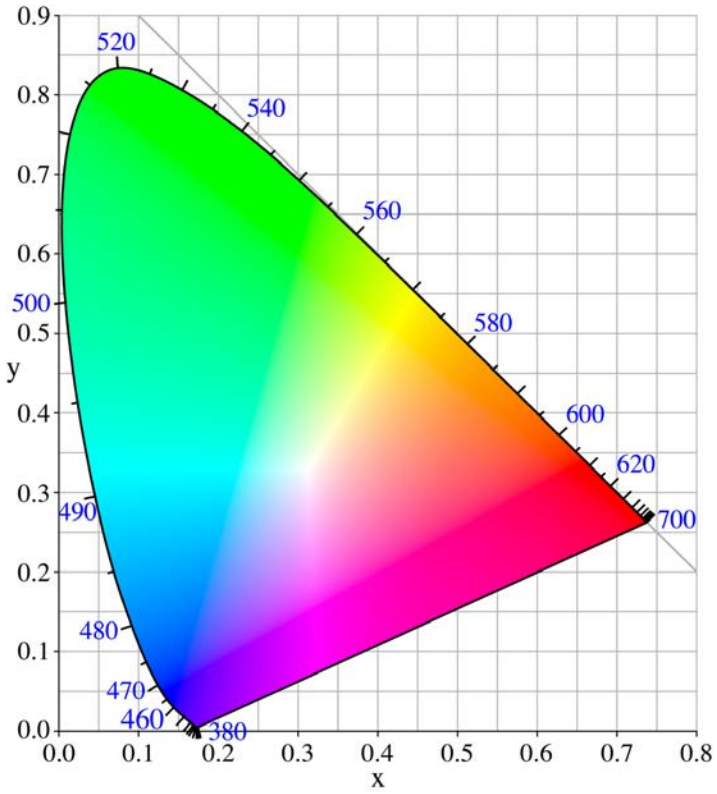
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

A digression about color…

- “Color” is an inherently perceptual phenomena.
- Only related but not the same as wavelength of light energy.
- In fact only some colors are found in the spectrum…

Colors perceivable by the human eye



CIE xy chromaticity diagram, 1931

CIE XYZ color space (1931)

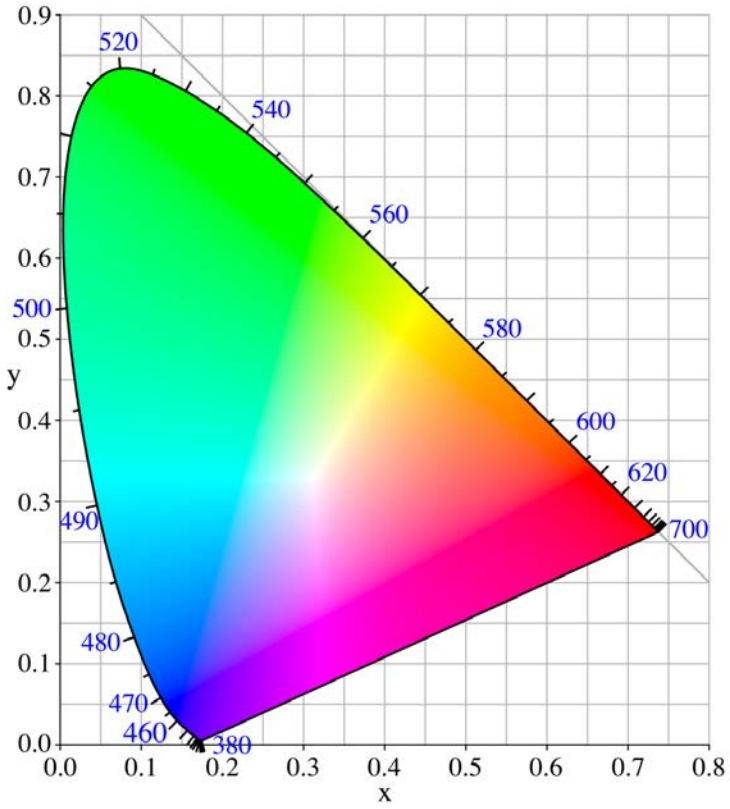
A space with desired properties

- Easy to compute – linear transform of CIE RGB
- Y: Perceived luminance
- X, Z: Perceived color
- Represents a wide range of colors

Colors perceivable by the human eye

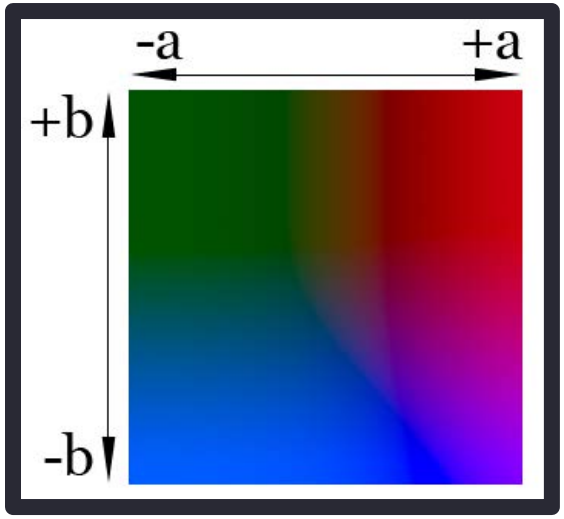
$$y = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

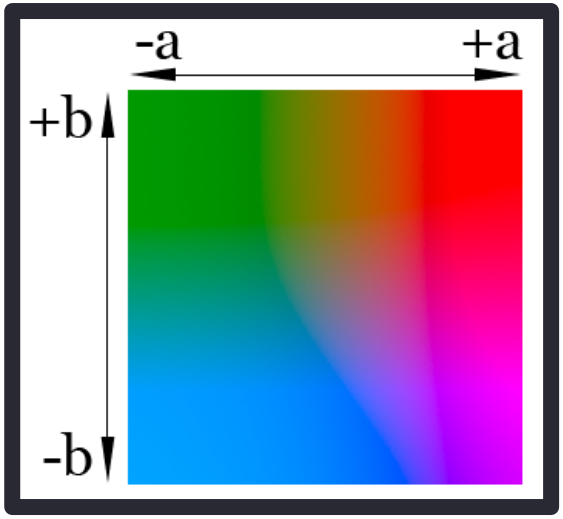


CIE xy chromaticity diagram, 1931

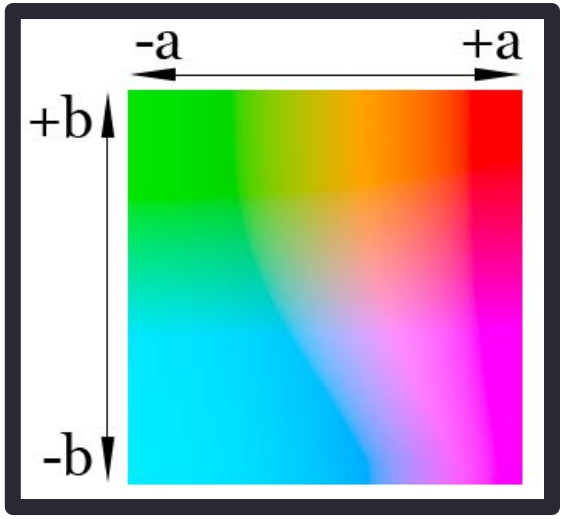
CIE L*a*b* color space



L = 25%



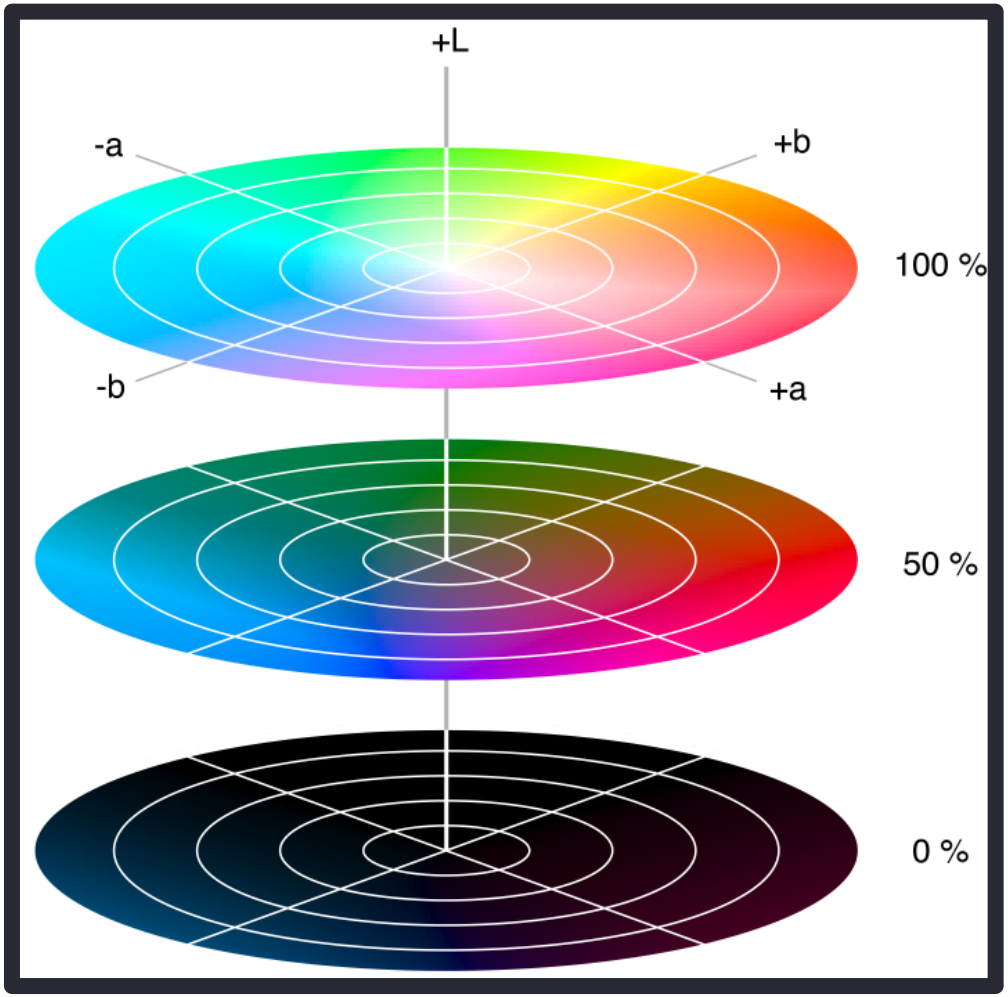
L = 50%



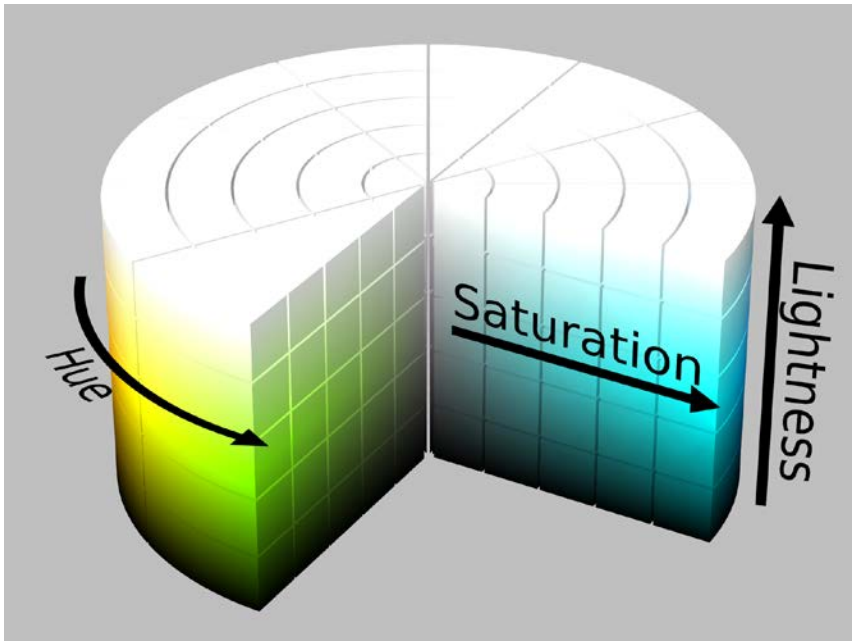
L = 75%

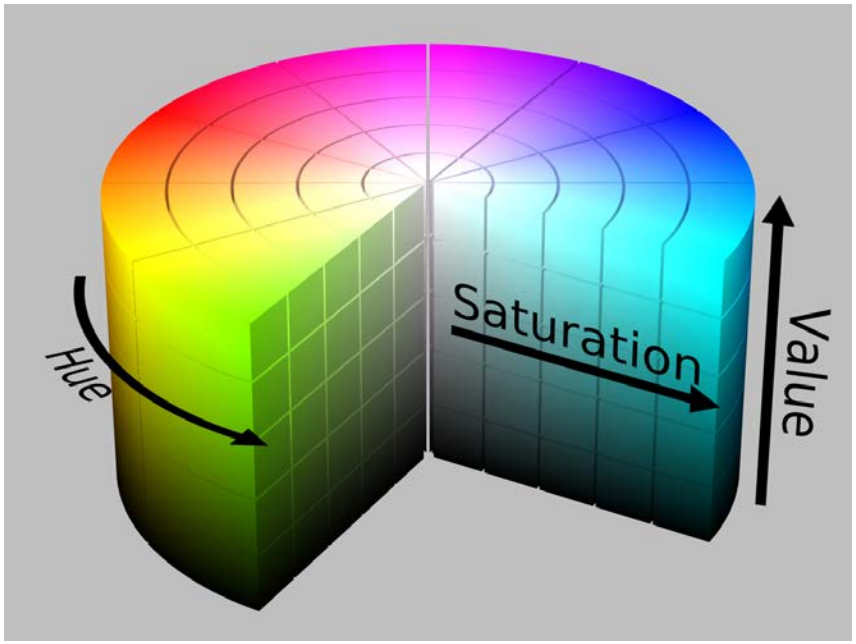
Cylindrical view

Think of chroma (here a*, b*) defining a planar disc at each luminance level (L)

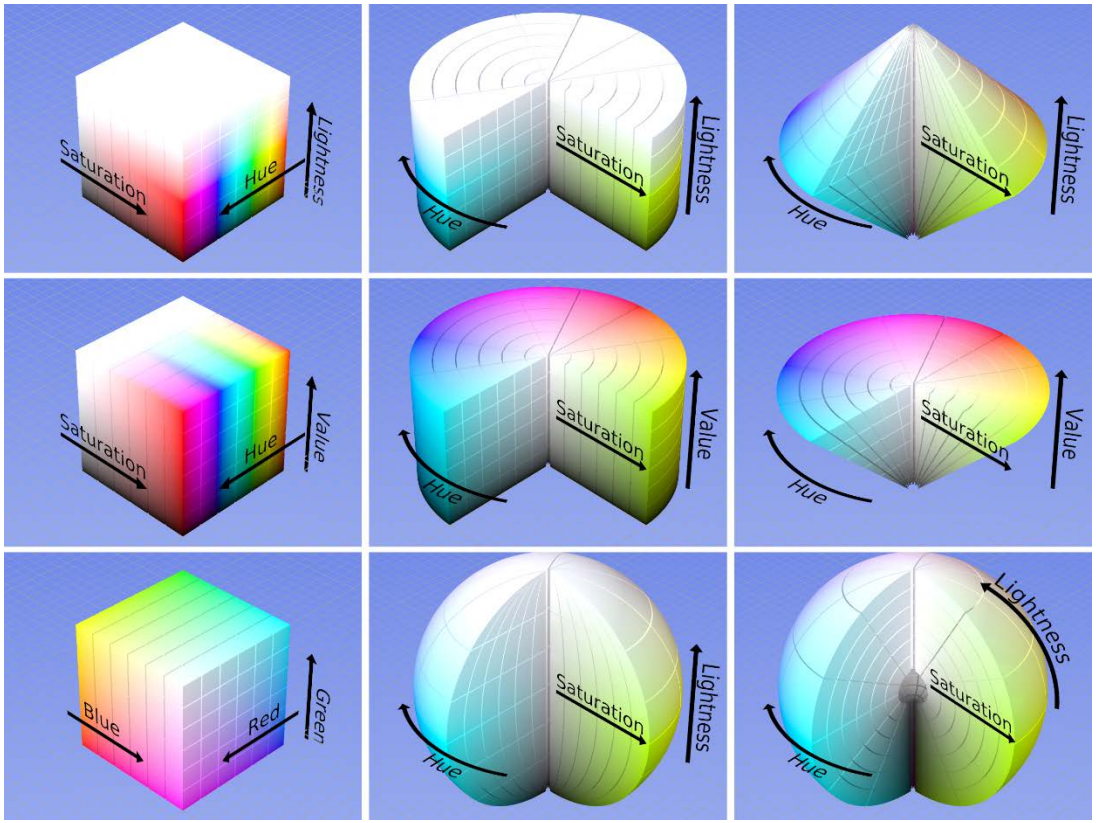


HSL and HSV color spaces



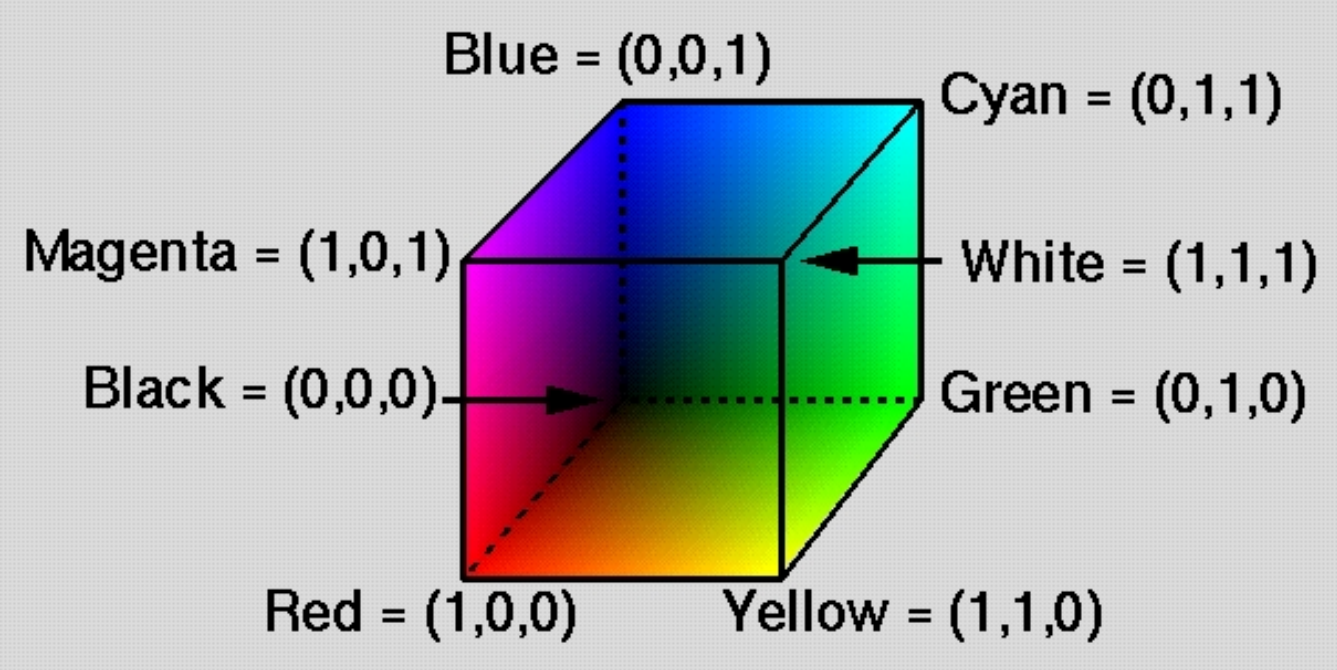


But there are lots of color spaces

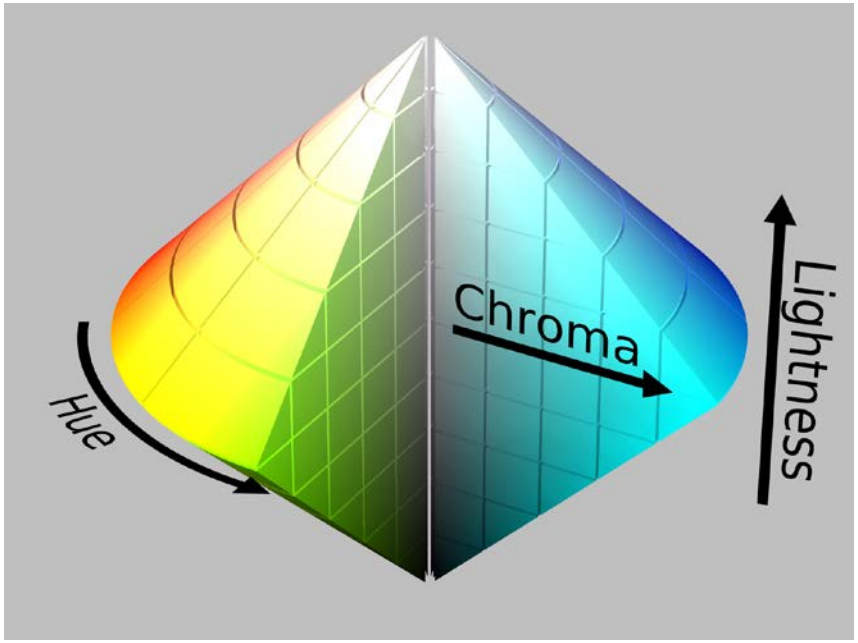


The one we know best…

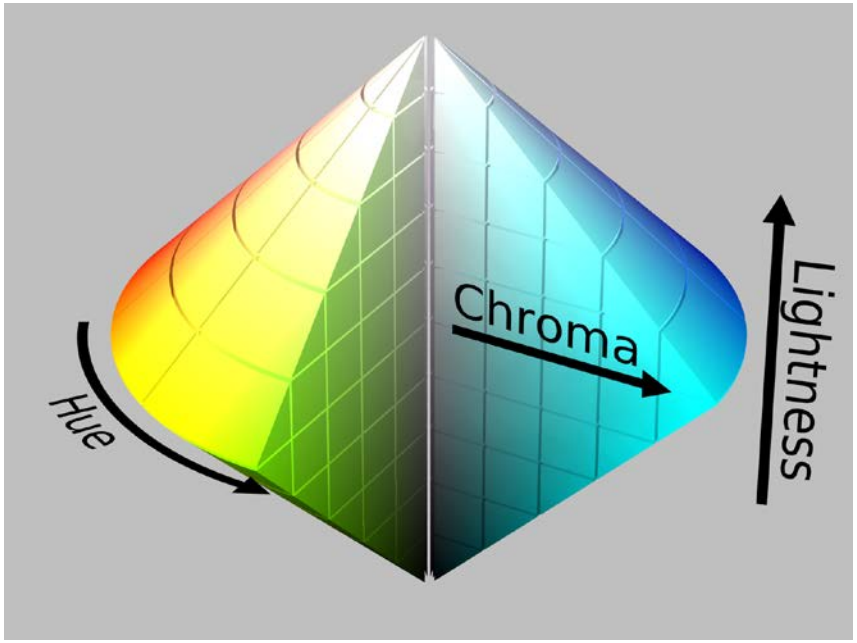
- RGB color space

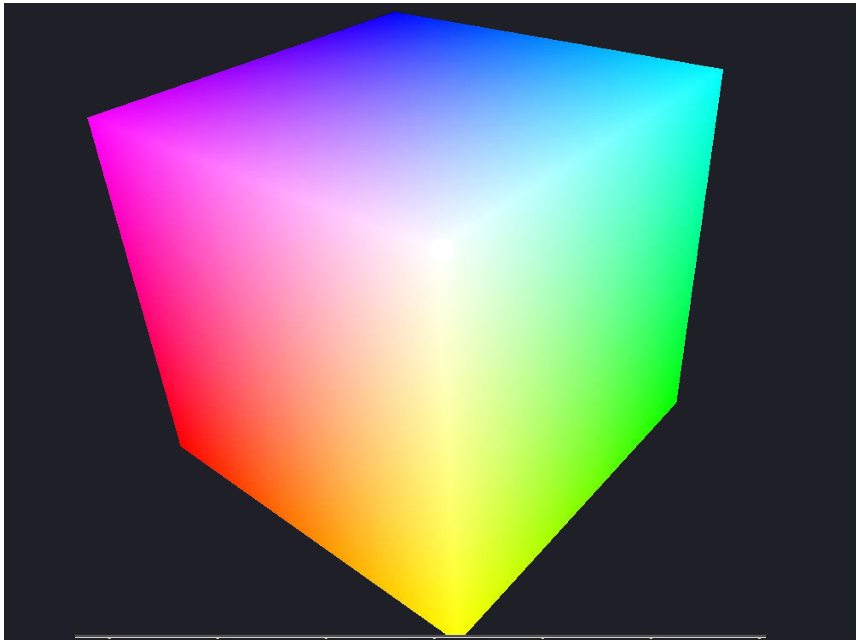


My favorite



Like a squared double cone?



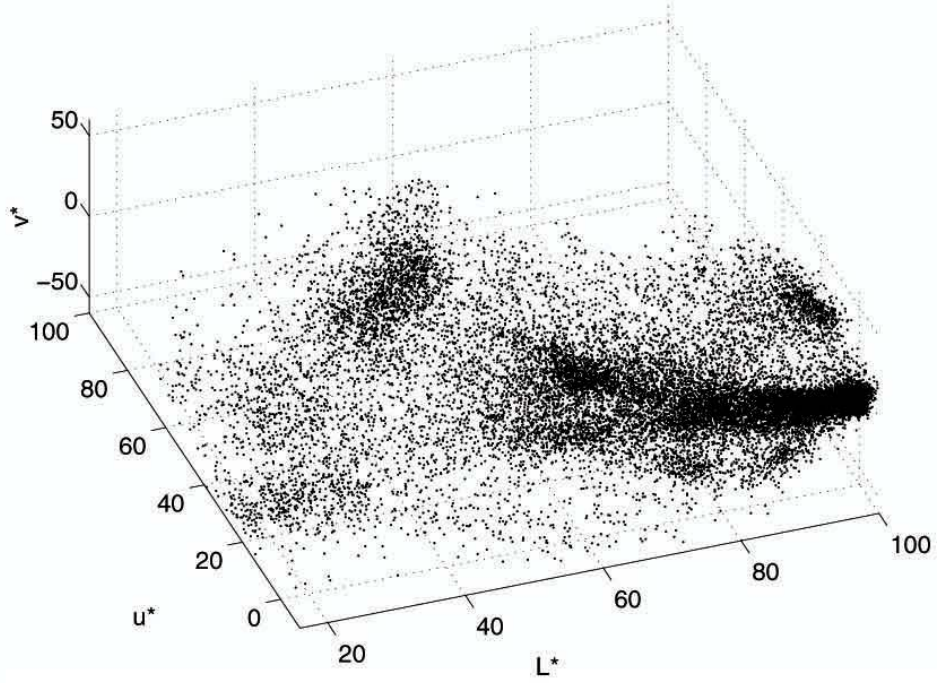


Mean shift algorithm

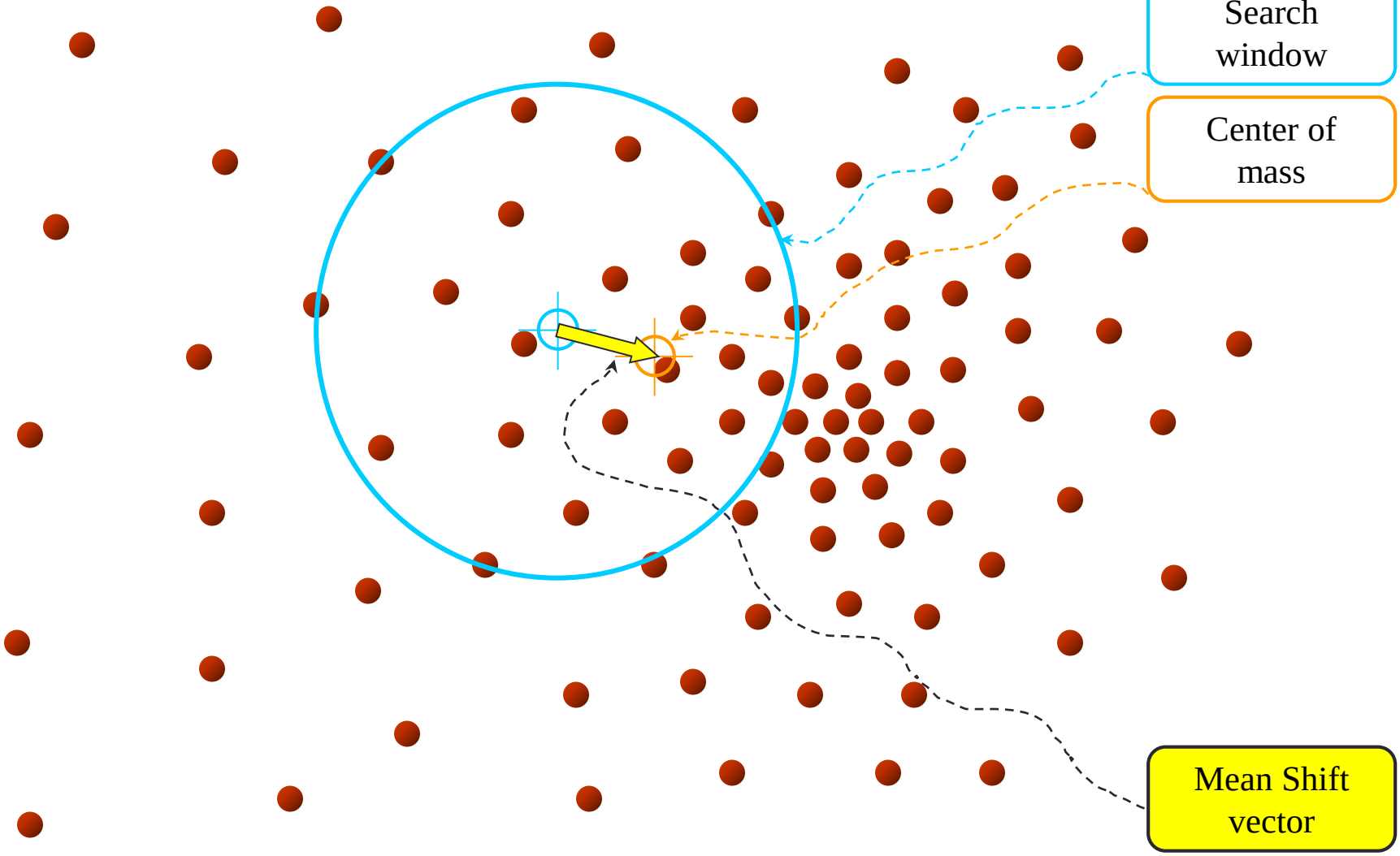
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

Feature space (L*u*v* color values)

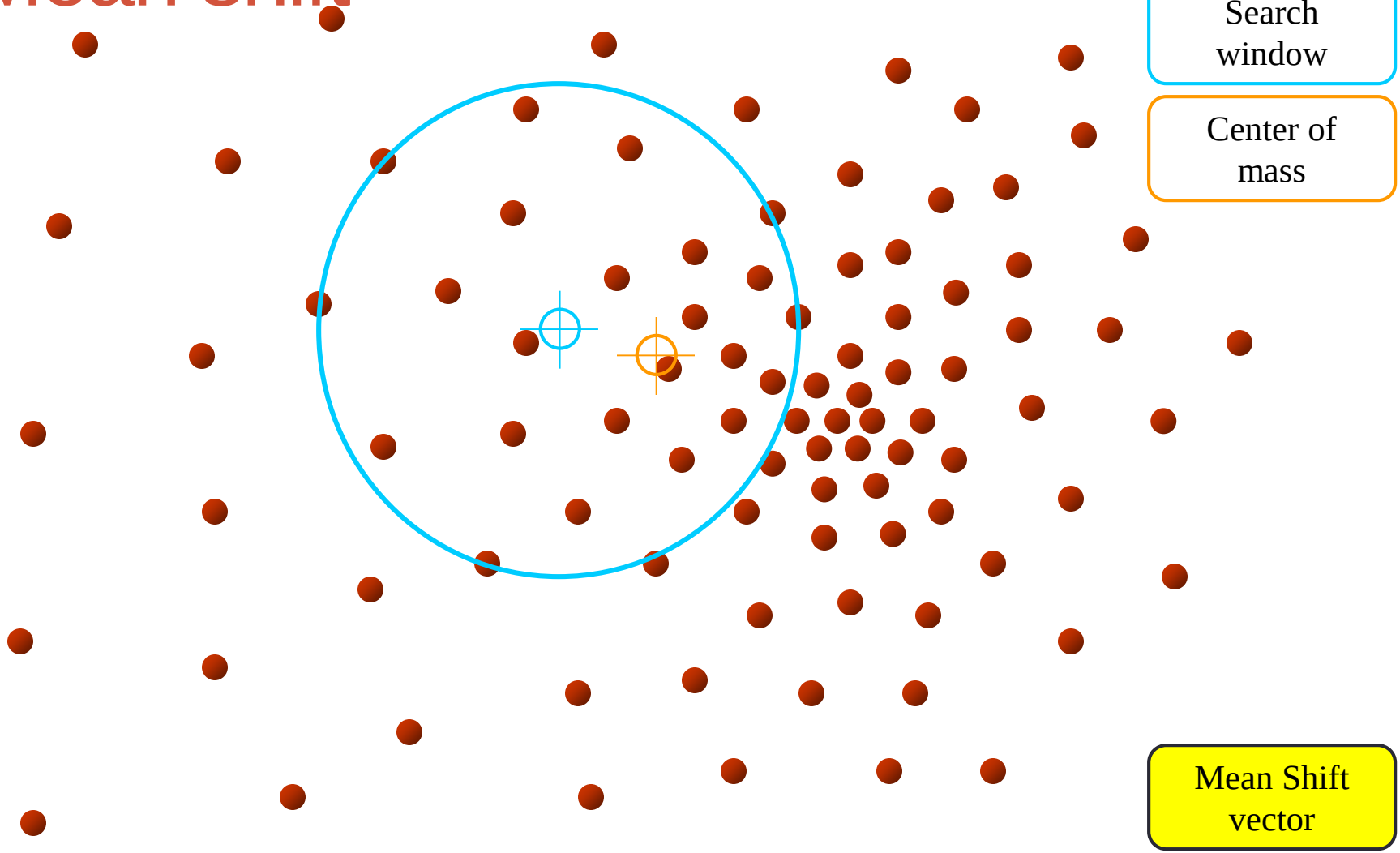


Mean shift



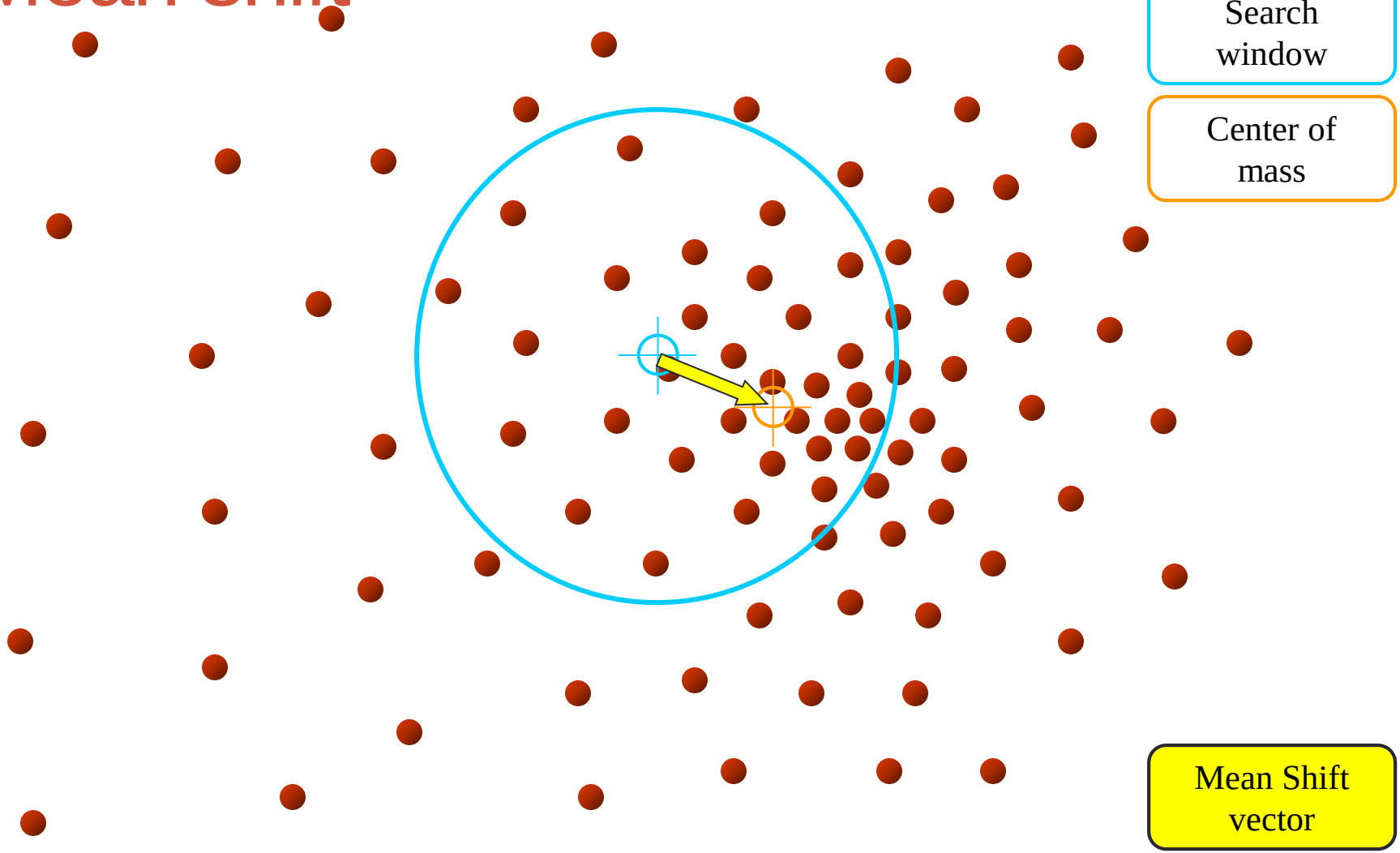
Slide by Y. Ukrainitz & B. Sarel

Mean shift



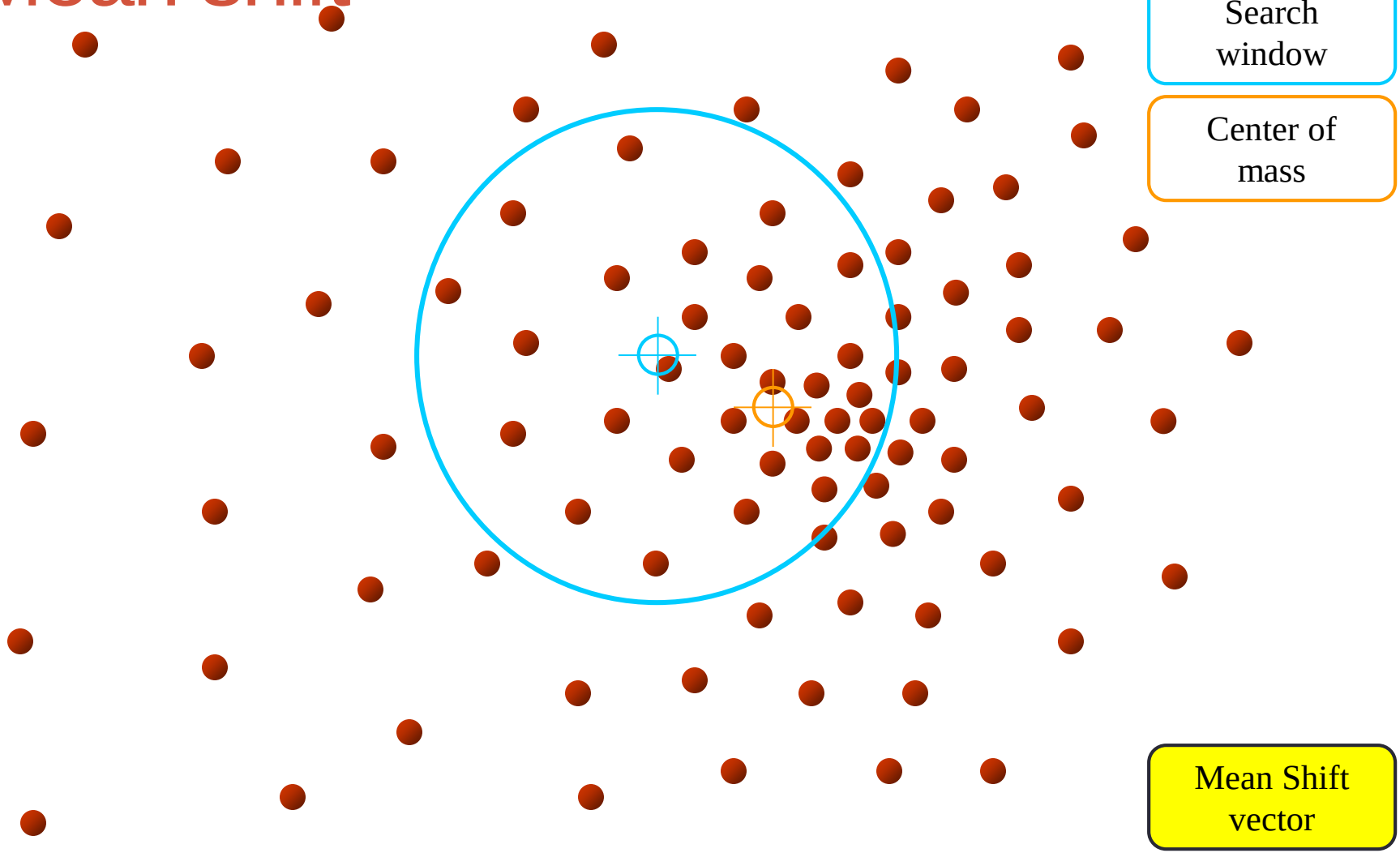
Slide by Y. Ukrainitz & B. Sarel

Mean shift



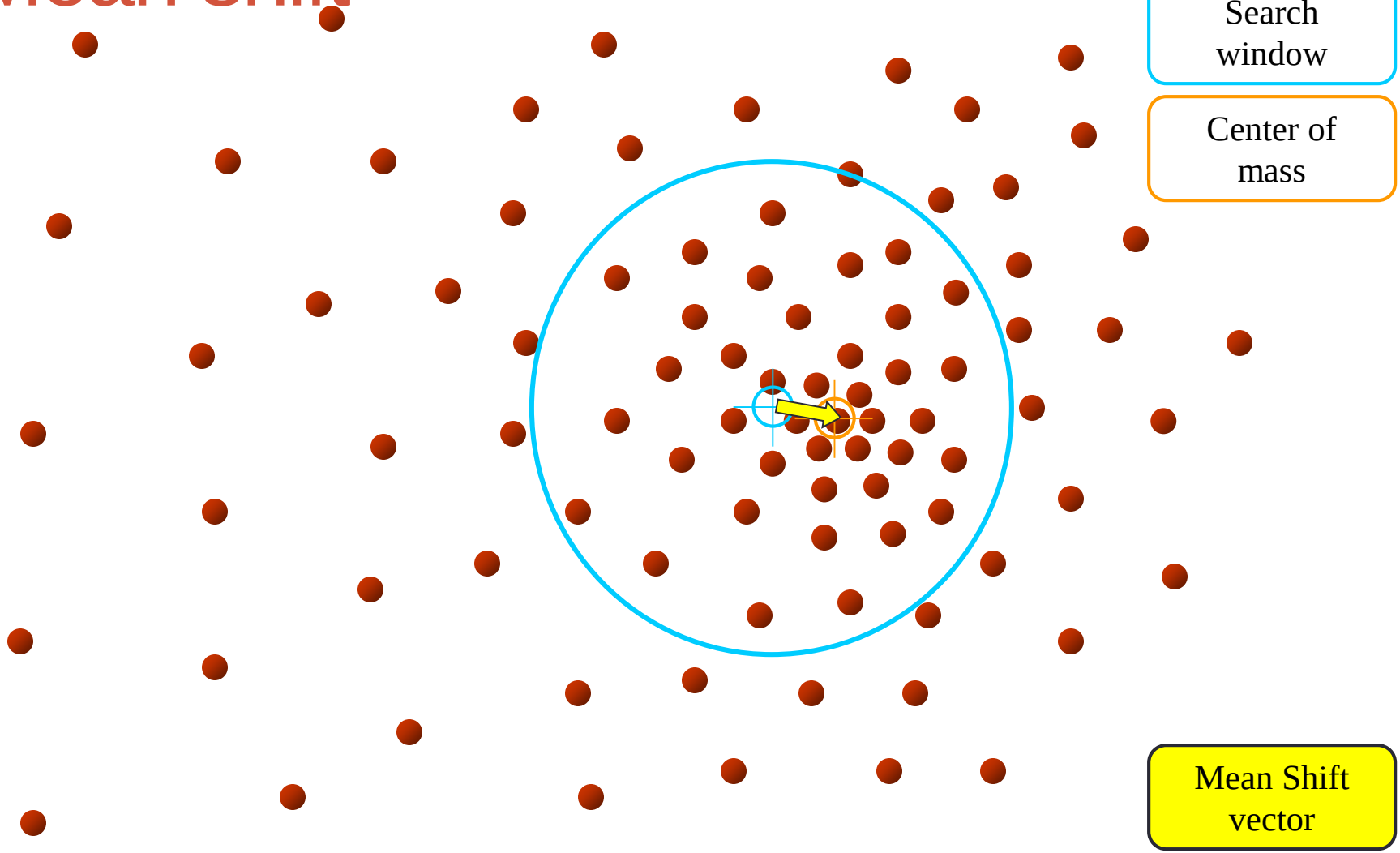
Slide by Y. Ukrainitz & B. Sarel

Mean shift



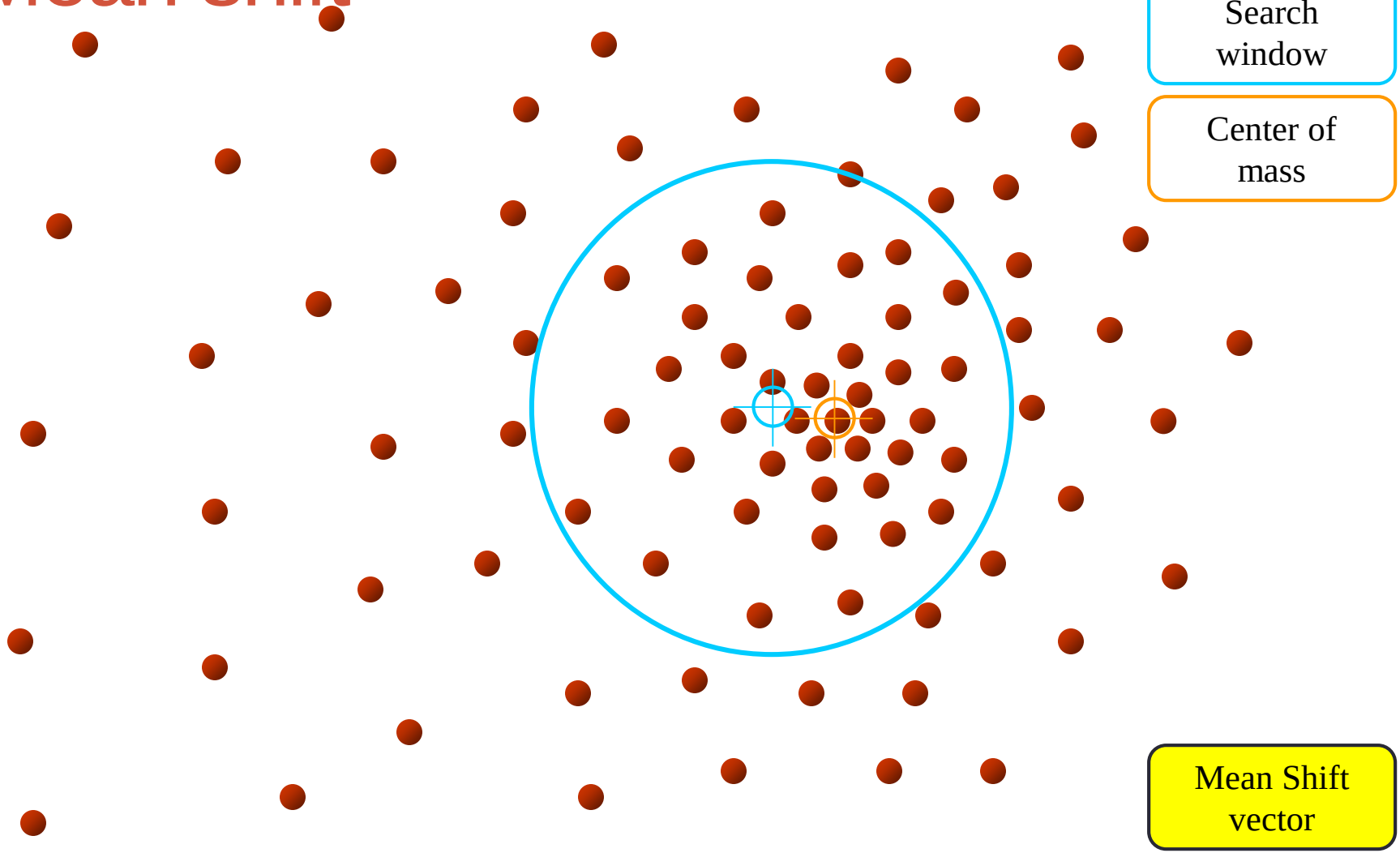
Slide by Y. Ukrainitz & B. Sarel

Mean shift



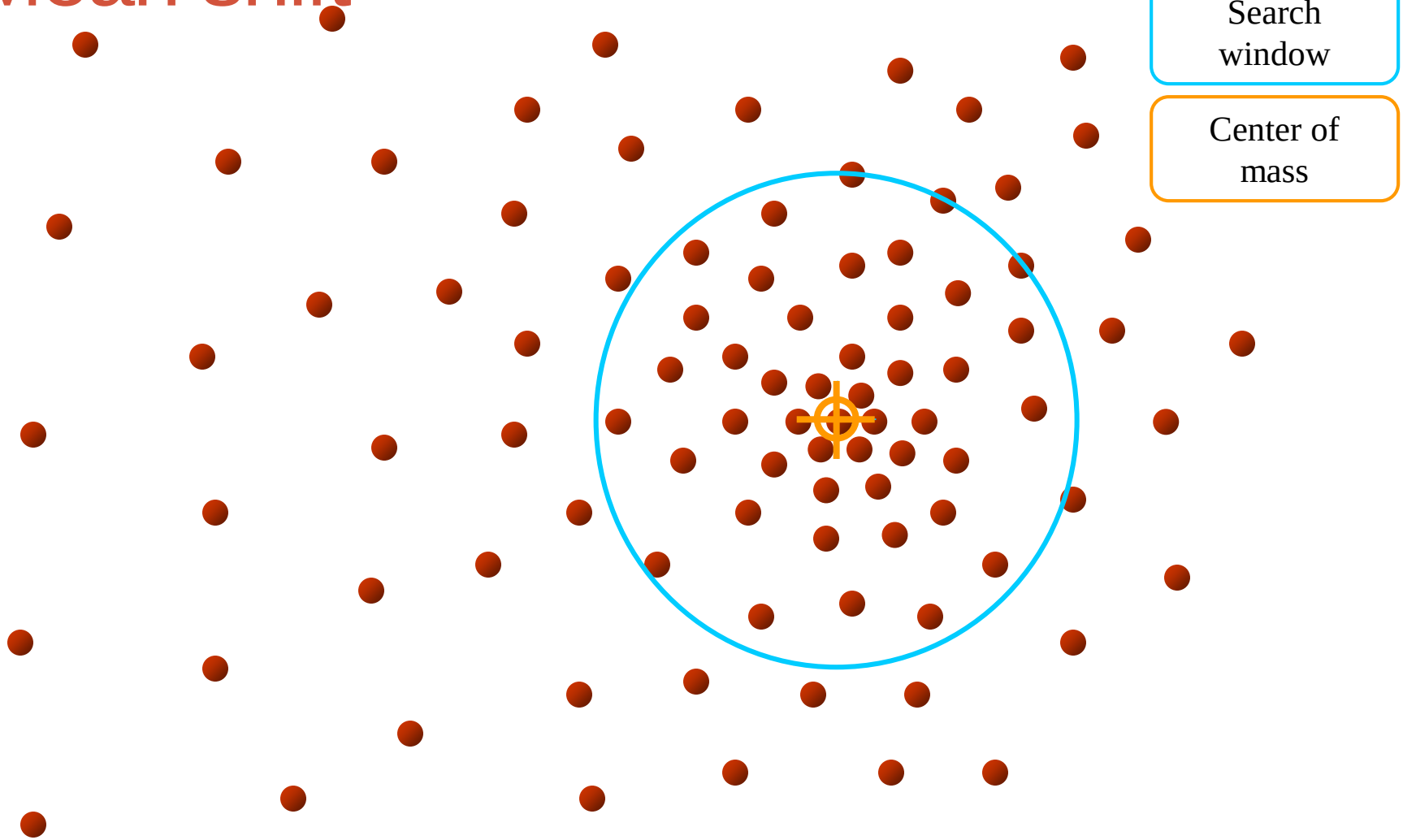
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

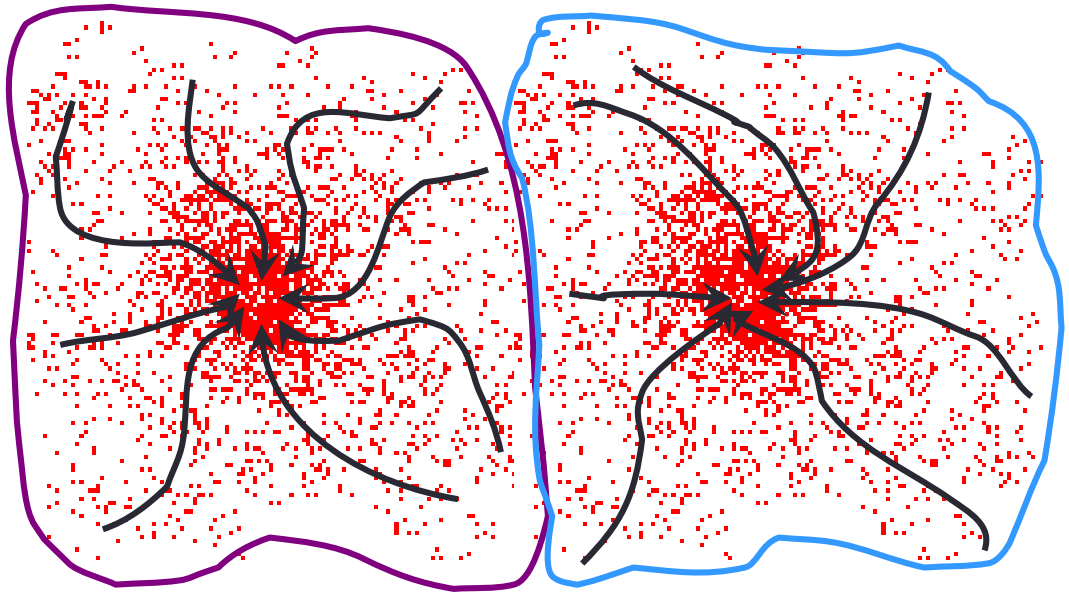
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

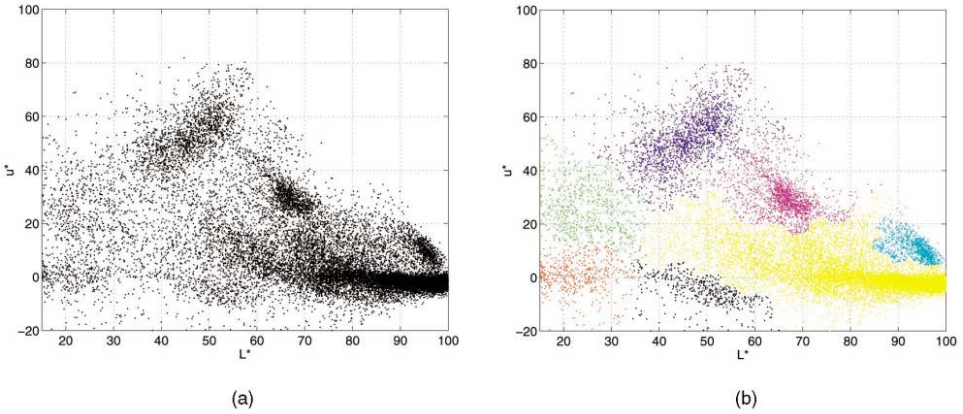
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

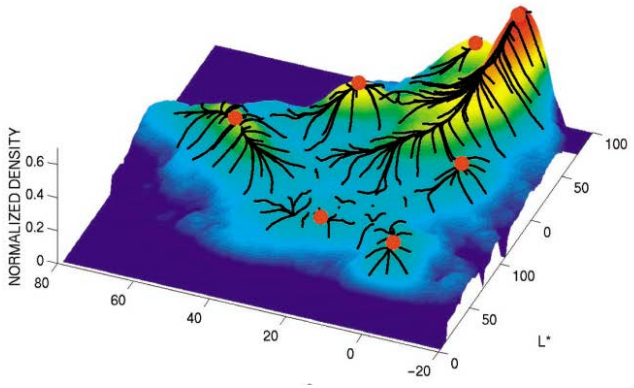


Slide by Y. Ukrainitz & B. Sarel

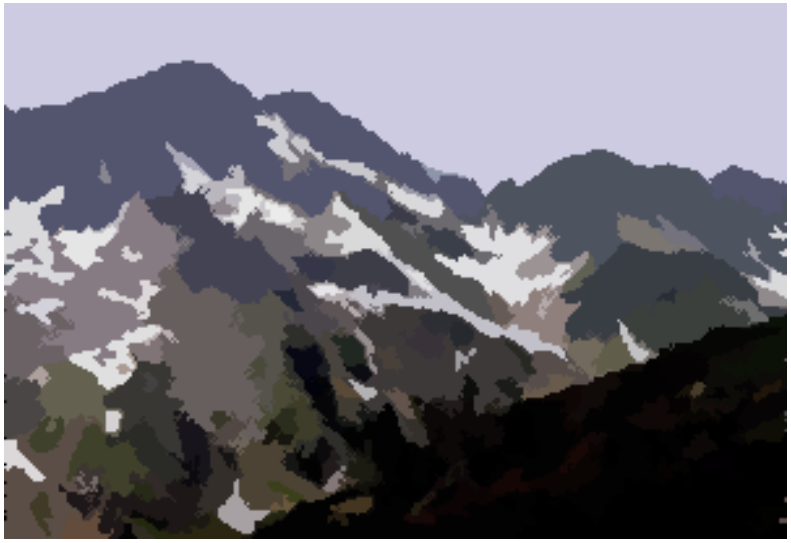
Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points (pixels)
- Perform mean shift for each window (pixel) until convergence
- Merge windows (pixels) that end up near the same “peak” or mode





Mean shift segmentation results



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

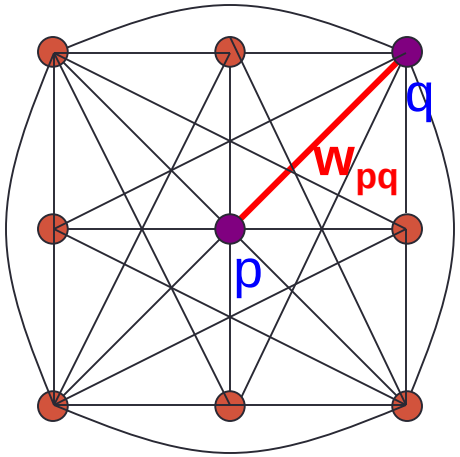
Mean shift segmentation results

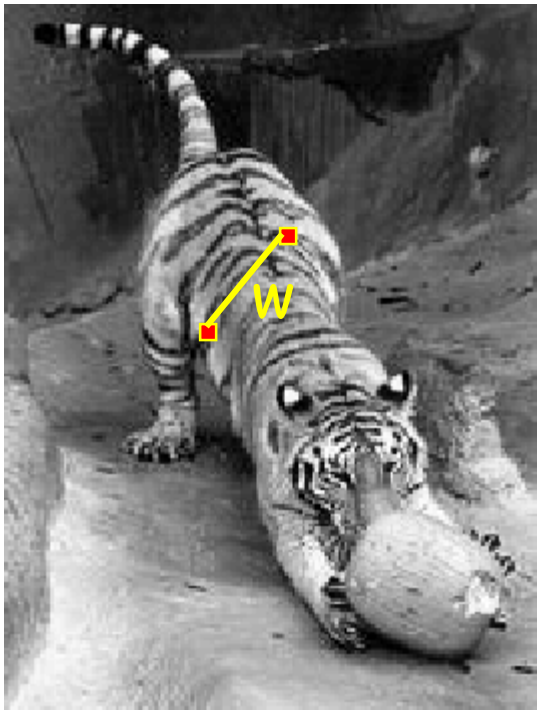



Mean shift

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size)
  - Generic technique
  - Find multiple modes
- Cons:
  - Selection of window size
  - Does not scale well with dimension of feature space

Images as graphs





- *Fully-connected* graph
  - node (vertex) for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - affinity weight $\mathbf{w_{pq}}$ for each link (edge)
    - $\mathbf{w_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference (in color and position…)

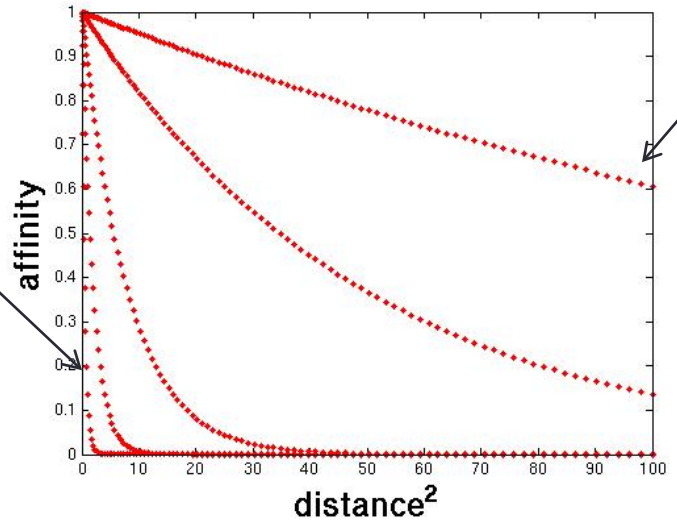
Source: Steve Seitz

Measuring affinity

- One possibility:

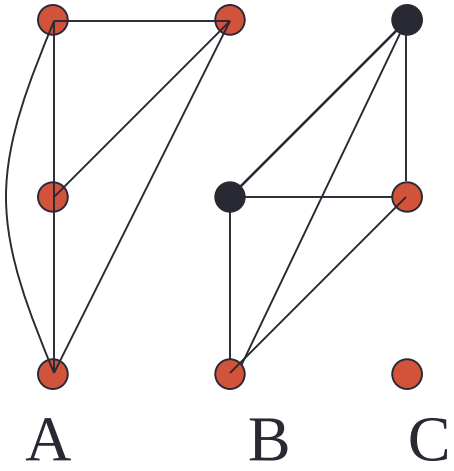
$$aff(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$

Small sigma: group only nearby points



Large sigma: group distant points

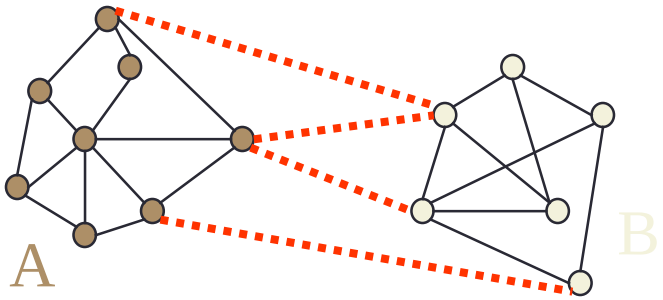
Segmentation by graph partitioning



- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low affinity
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz

Graph cut



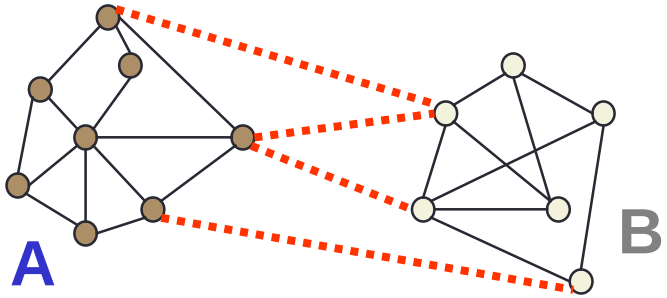
- Set of edges whose removal makes a graph disconnected
- Cost of a cut: sum of weights of cut edges

$$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

- A graph cut gives us a segmentation
  - What is a “good” graph cut and how do we find one?

Source: S. Seitz

Cuts in a graph: Min cut



$$cut(A, B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this (we may see this…)

Source: Steve Seitz

Minimum cut

- Problem with minimum cut:

  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

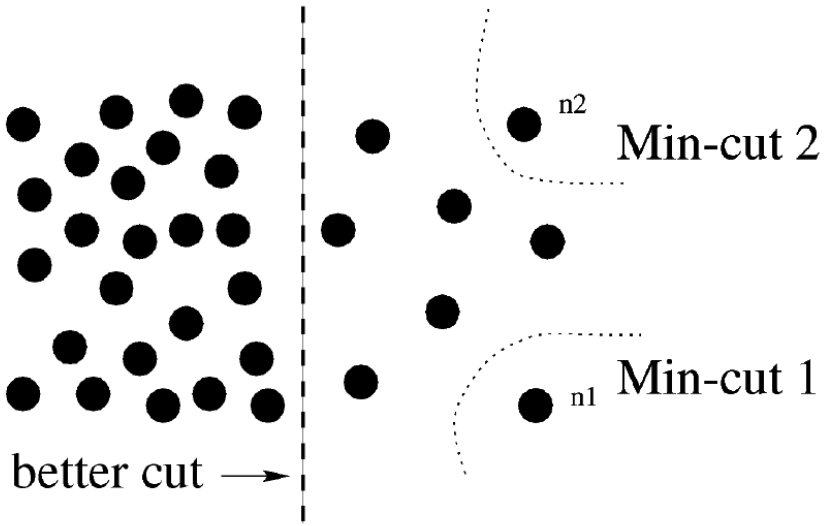
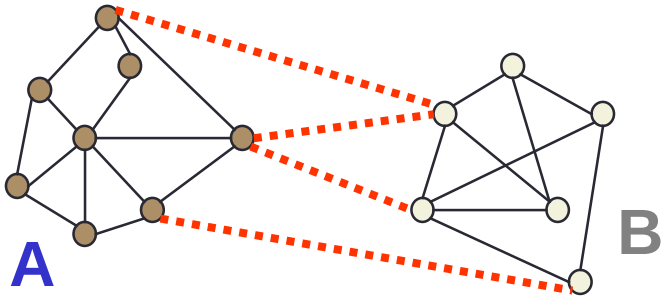


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut



Normalized Cut

- fix bias of Min Cut by **normalizing** for size of segments:

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

assoc(A,V) = sum of weights of all edges that touch A

- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

J. Shi and J. Malik, Normalized Cuts and Image Segmentation, CVPR, 1997

Source: Steve Seitz

Example results

Results: Berkeley Segmentation Engine

http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: pros and cons

Pros:

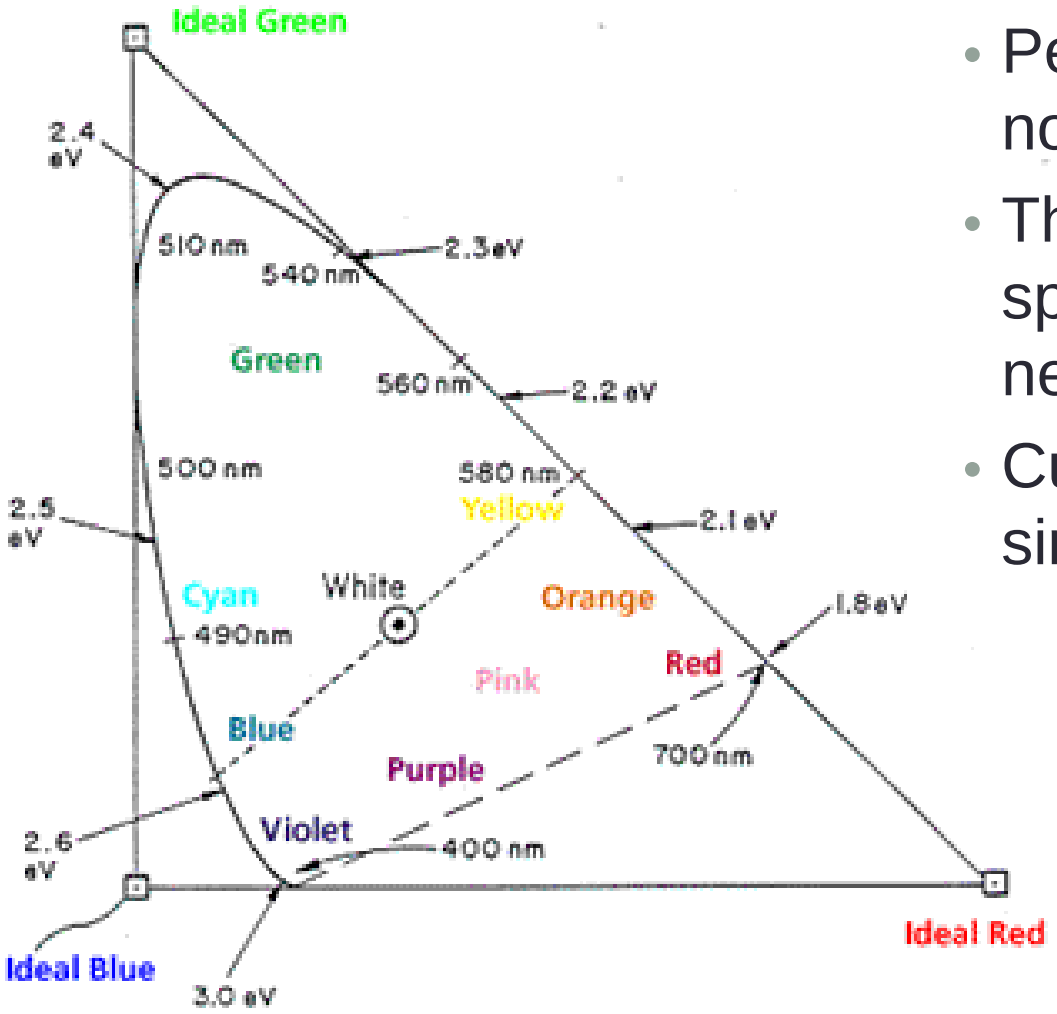
- Generic framework, flexible to choice of function that computes weights (“affinities”) between nodes
- Does not require model of the data distribution

Cons:

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

The end…

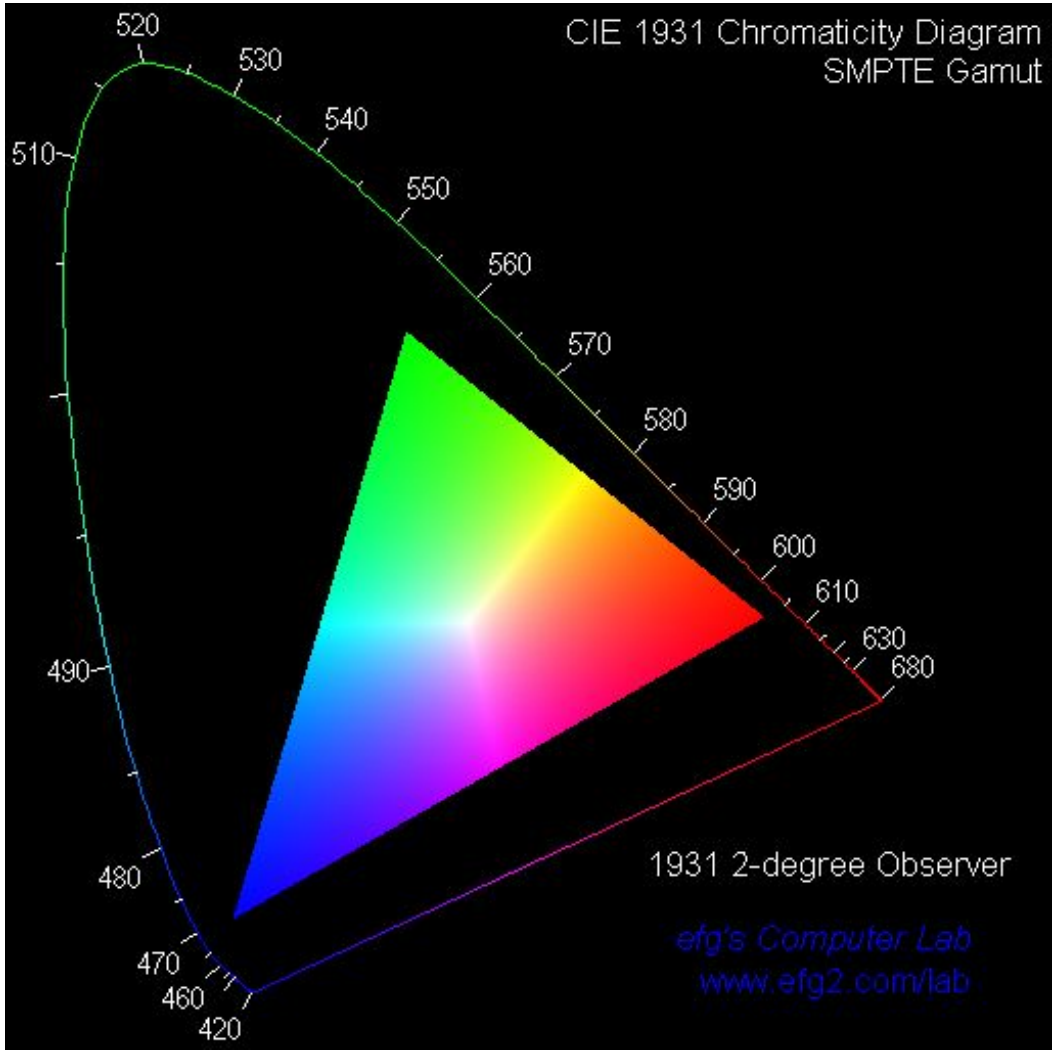
Geometry of Color (CIE)



- Perceptual color spaces are non-convex
- Three primaries can span the space, but weights may be negative.
- Curved outer edge consists of single wavelength primaries

Source: Jim Rehg

RGB Color Space



Many colors cannot be represented (phosphor limitations)

Source: Jim Rehg

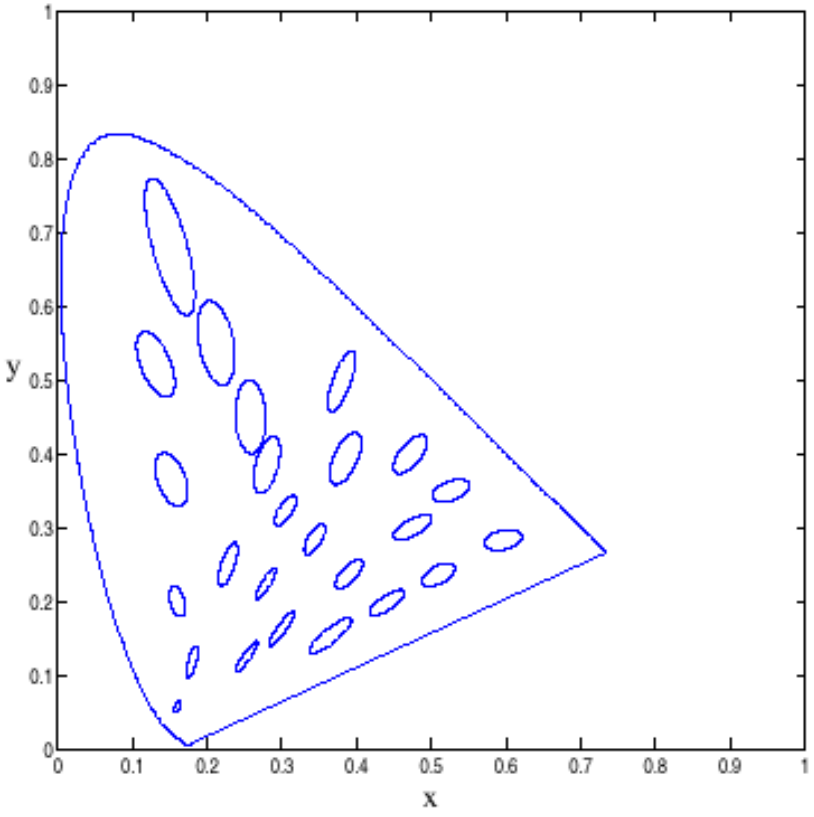
Uniform color spaces

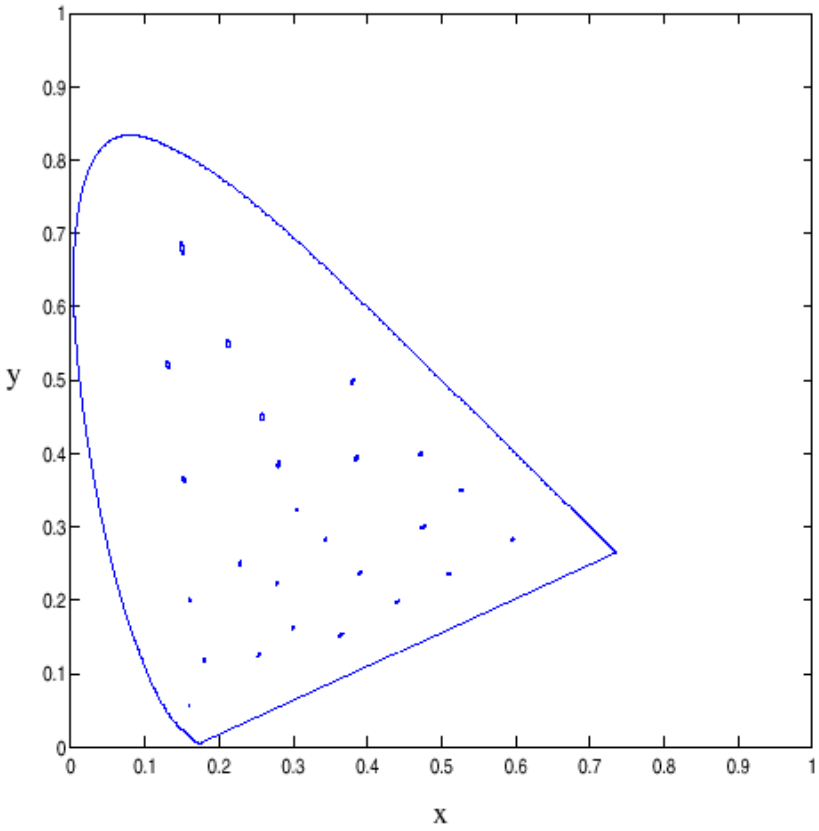
- McAdam ellipses (next slide) demonstrate that differences in x,y are a poor guide to differences in color
- Construct color spaces so that differences in coordinates are a good guide to differences in color.

Source: Jim Rehg

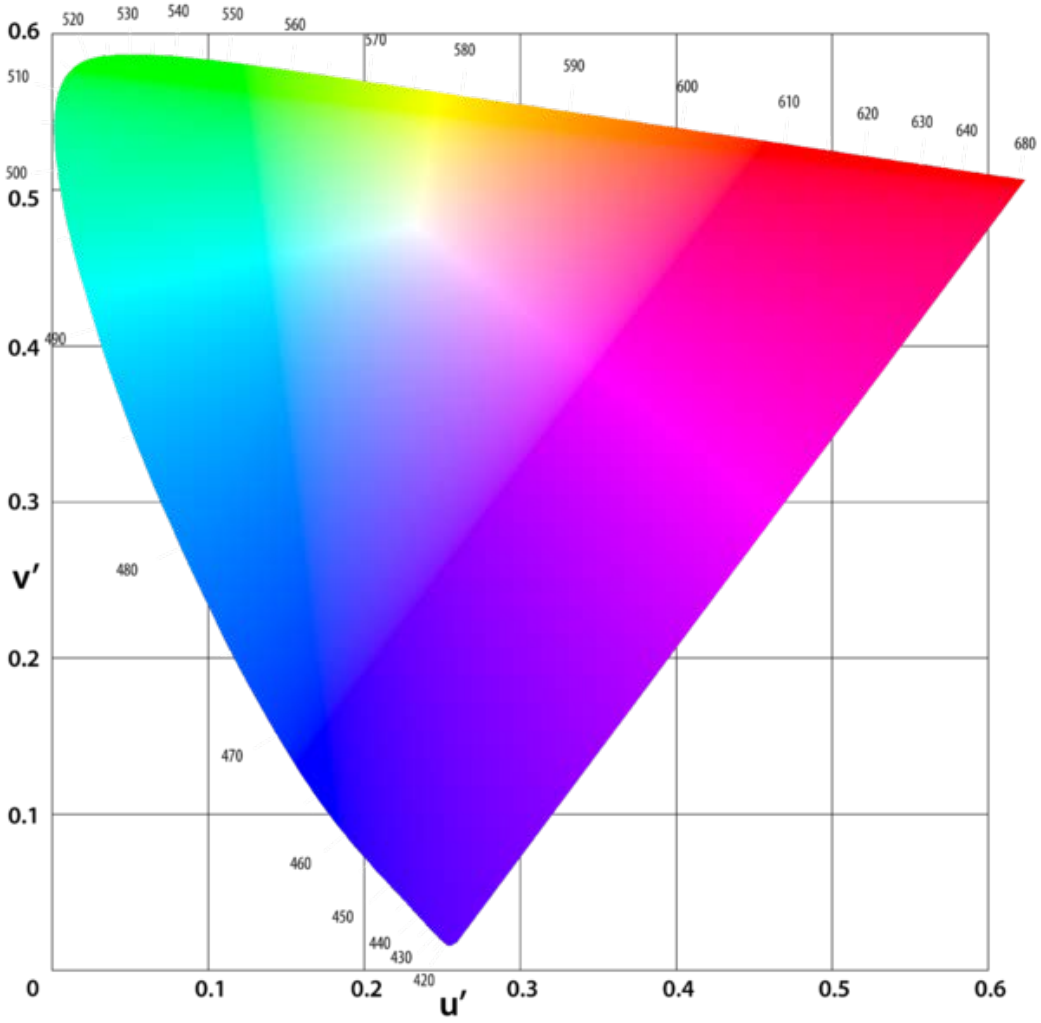
McAdam ellipses

Figures courtesy of
D. Forsyth



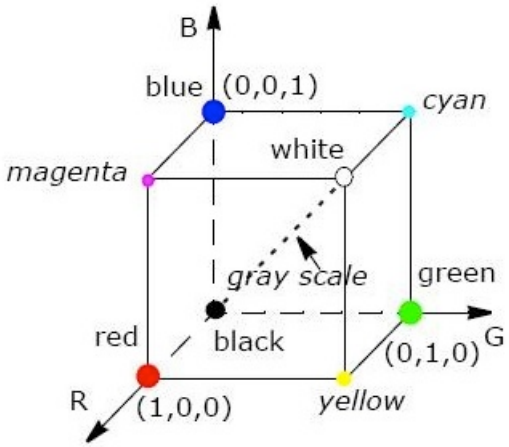


LUV Color Space

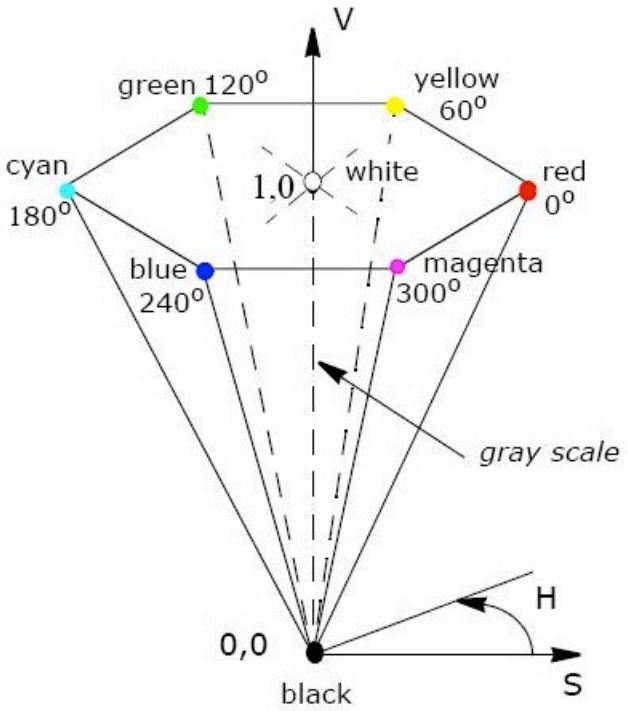


HSV Color Space

- RGB



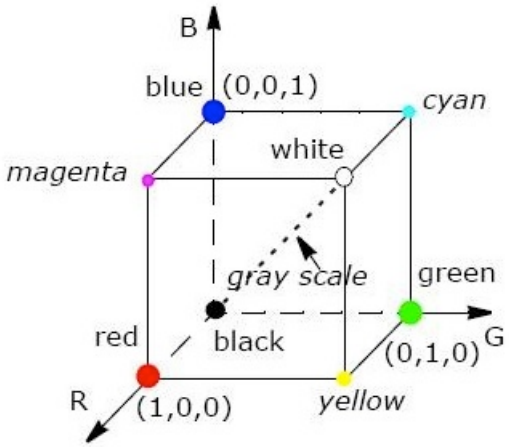
- HSV



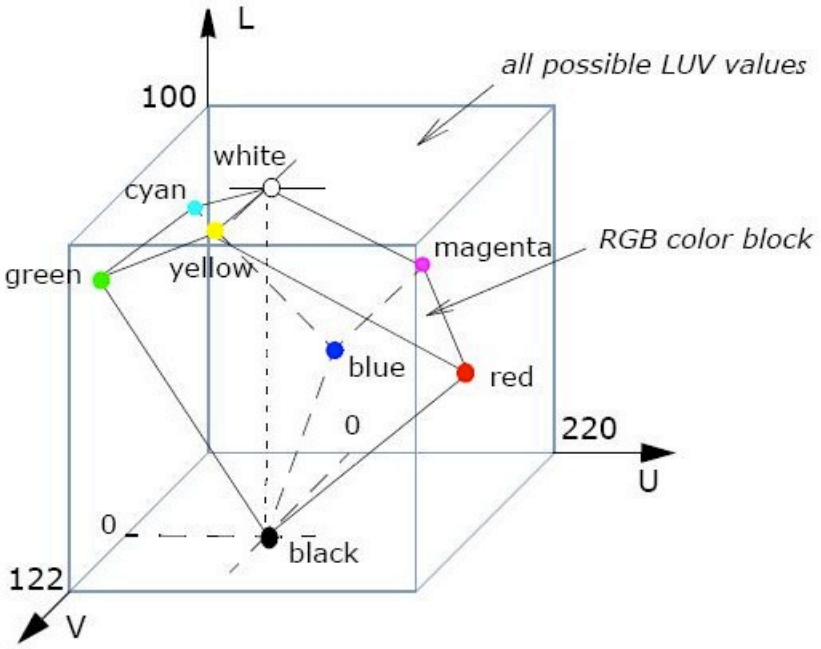
Source: Intel IPP

LUV Color Space

- RGB



- LUV



More info see: http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.html

Source: Intel IPP