ECE5554 – Computer Vision
Lecture 7c – Segmentation by Clustering

Creed Jones, PhD

Today’s Objectives

Segmentation by Clustering

- Intensity clustering
- Brief discussion of RGB space
- Color space clustering using KMeans
- Texture clustering
- The Mean-shift approach

Segmentation as clustering

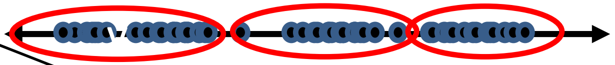
Depending on what we choose as the *feature space*, we can group pixels in different ways.

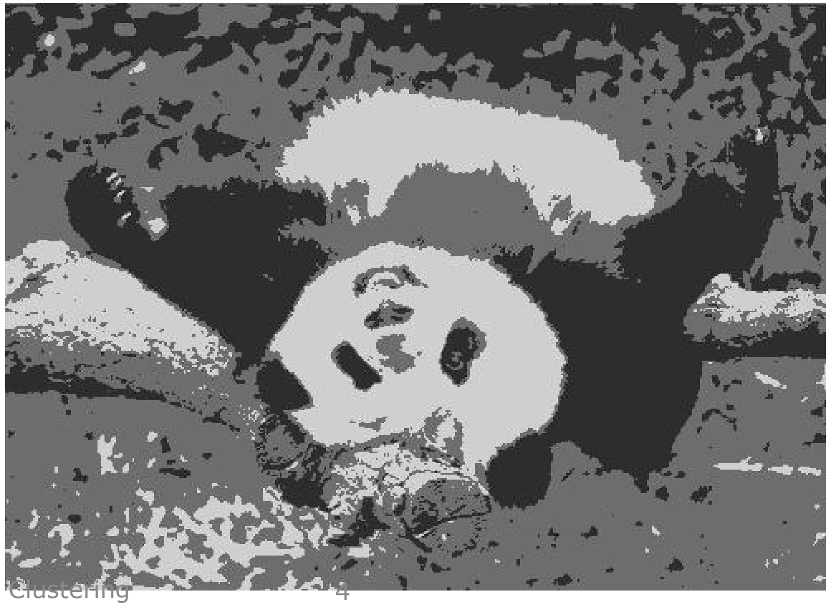
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-dimensional)

K=2

K=3



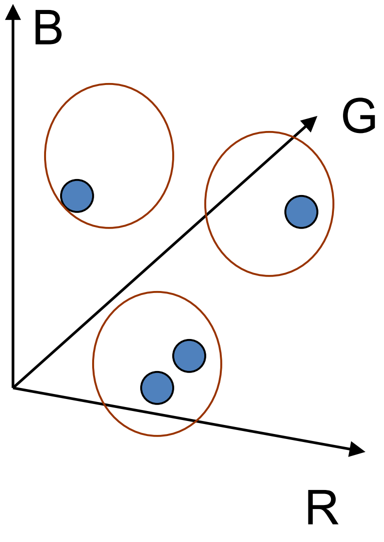


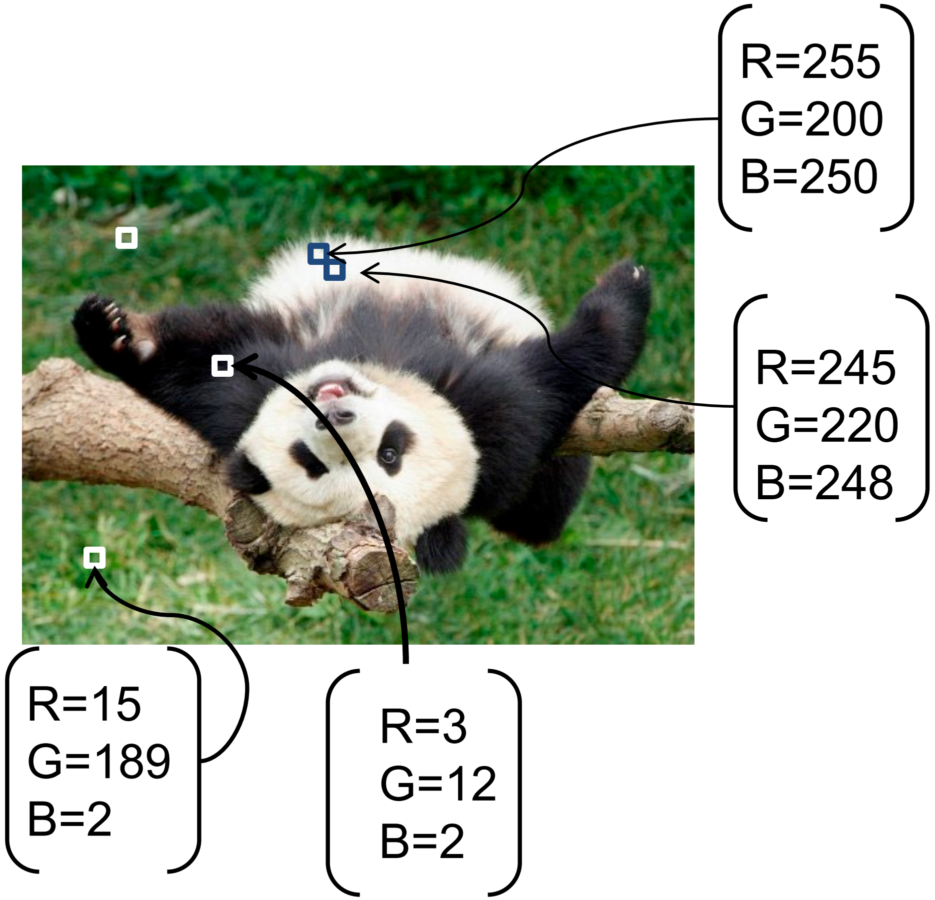
Clustering in the intensity axis accomplishes *Quantization* of the feature space; segmentation label map

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

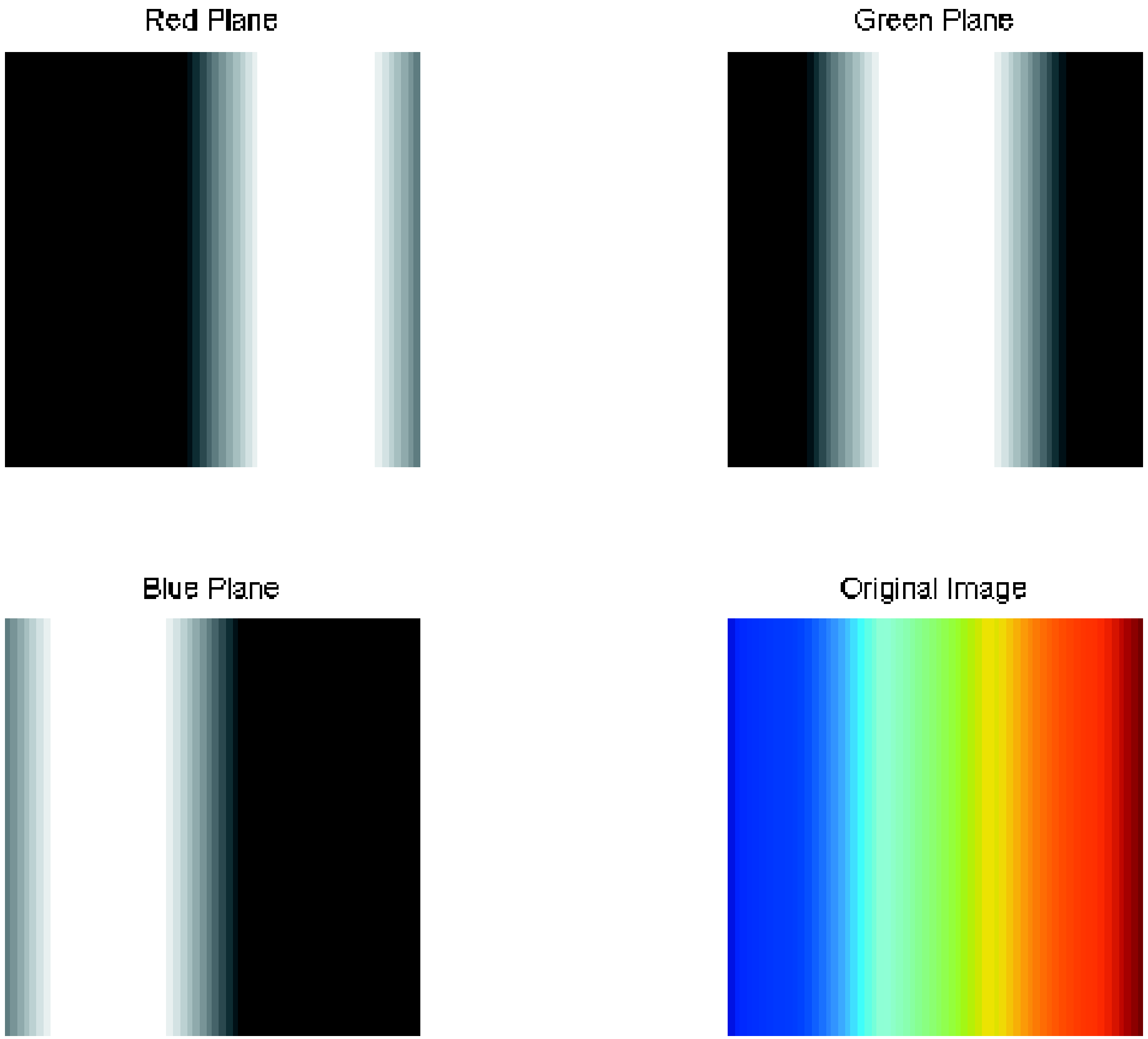




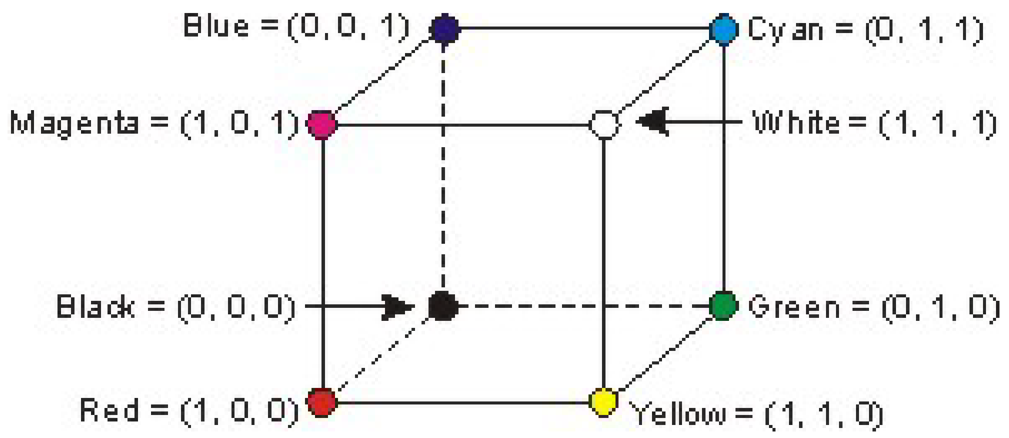
Feature space: color value (3-D)

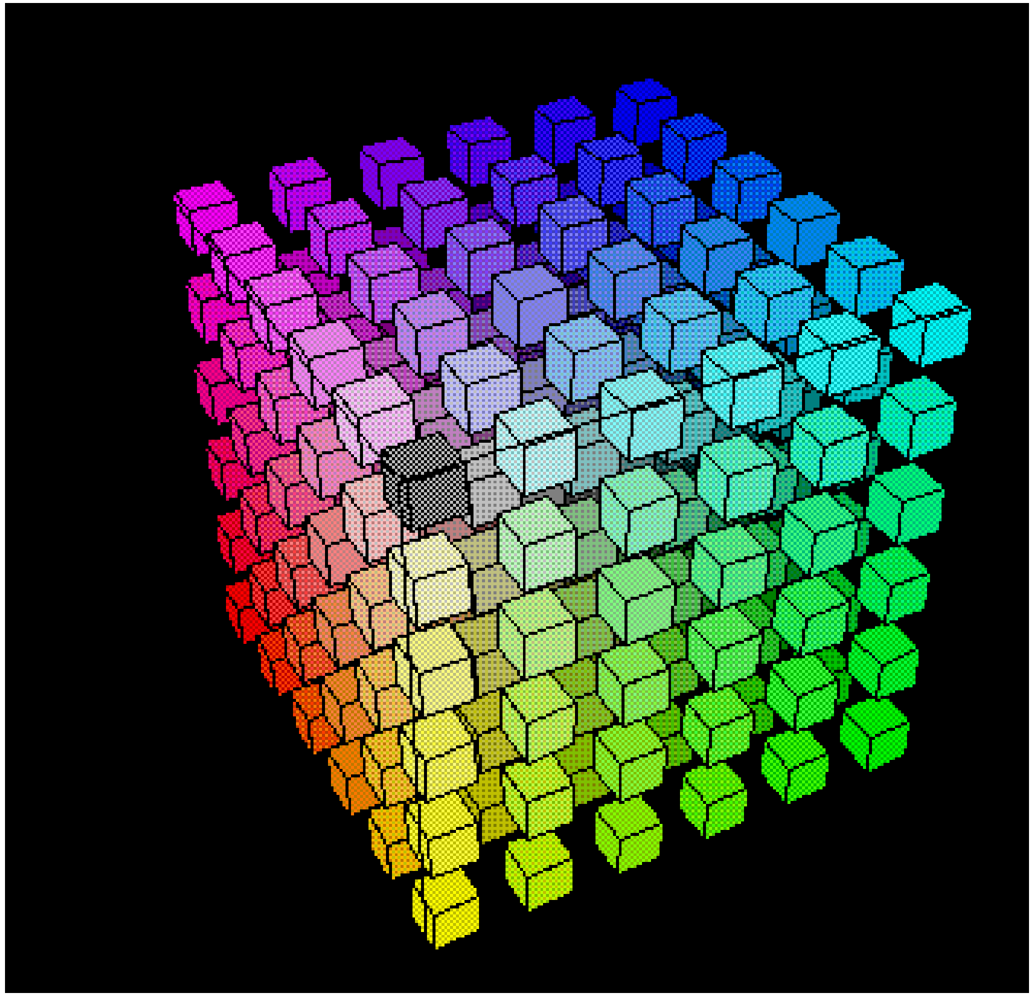
A color image in three components

- The overall image can be thought of as a vector function of two dimensions
- $\boldsymbol{I}(x,y) = \begin{bmatrix} I_R(x,y) \\ I_G(x,y) \\ I_B(x,y) \end{bmatrix}$
- Each component image is bright where the overall image is high in that color



RGB color space can be thought of as 3D Cartesian space

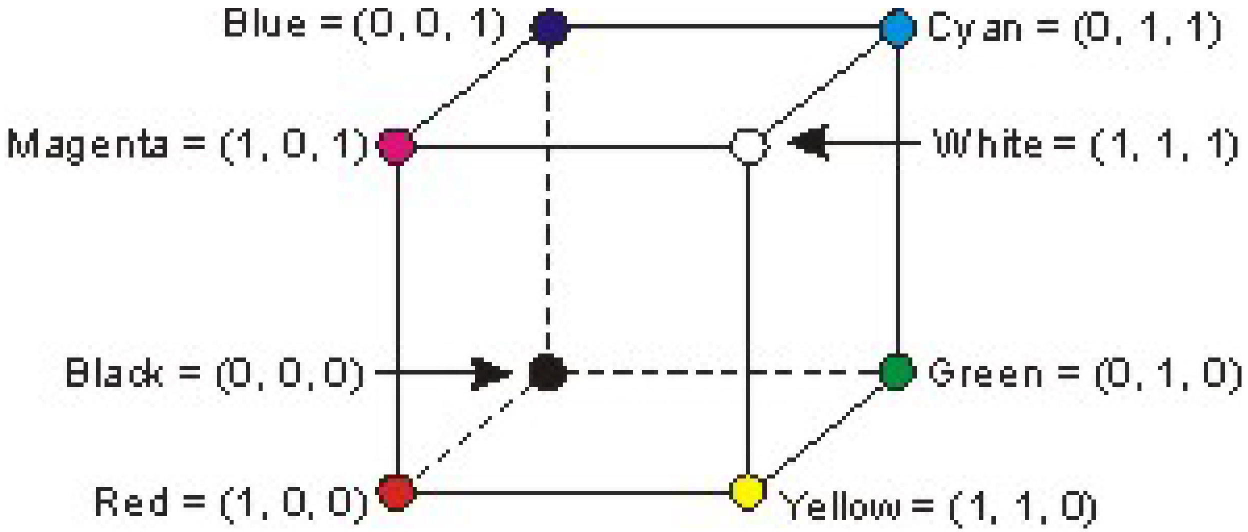




- Red, Green and Blue are the 3 axes
- Black is at the origin
- Shades of gray are along the diagonal of the cube
- Widely used in color cameras and displays

Color variations in RGB space

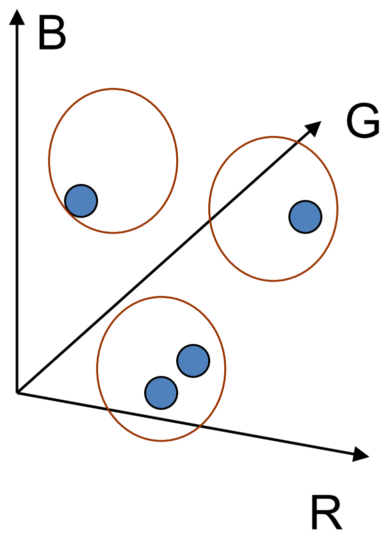
- Grays are found along the diagonal line where R=G=B (no discernible tone)
- To make a color darker without changing the tone, move along the line towards the origin
  - To make lighter, move out along the same line
- Moving towards any of the corners changes the tone

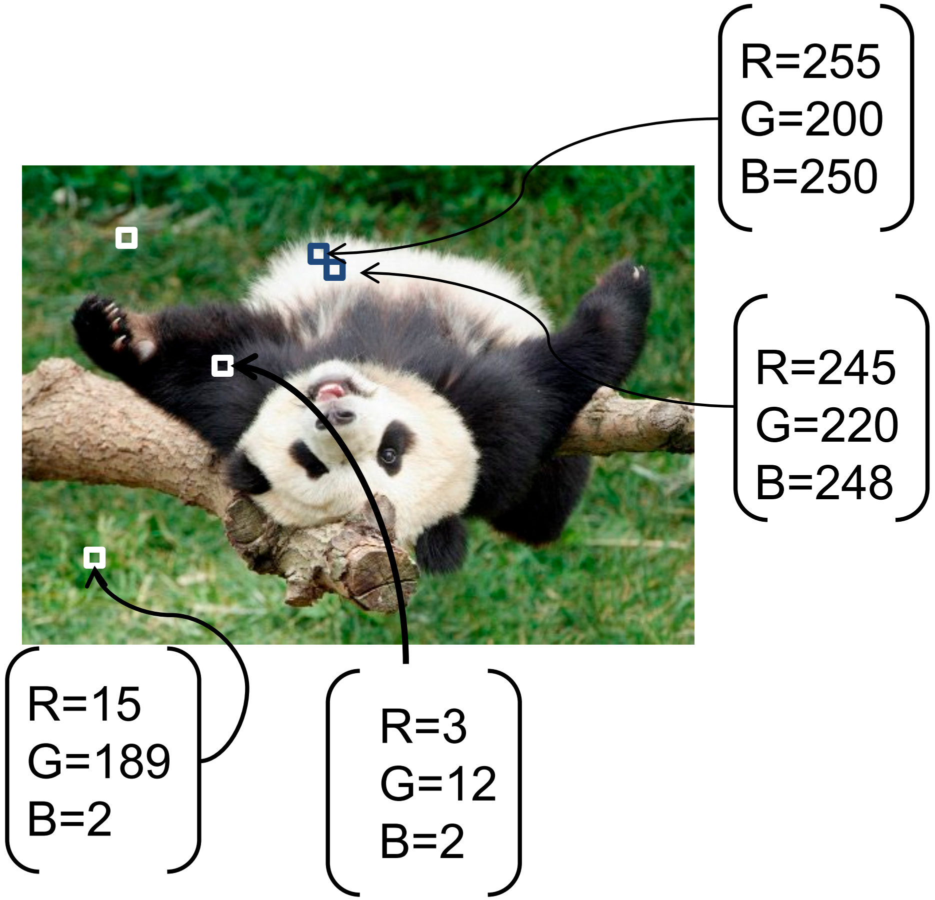


Segmentation as clustering

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)

Look at the output of KMeans clustering in RGB space on a color image

- find clusters
- set each pixel to its cluster number
- assign false colors to each cluster

Look at the output of KMeans clustering in RGB space on a color image

- find clusters
- set each pixel to its cluster number
- assign false colors to each cluster

K=4

K=5

K=6

K=12

Color space KMeans clustering in opencv/Python

```
import numpy as np
import cv2
import sklearn.cluster as cl

# Load an image and segment on color alone
nclusters = 8
img = cv2.imread('C:\\Data\\spock.jpg')
(ROWS, COLS, PLANES) = img.shape
print("Image shape is" + str(img.shape))
colors = img.reshape((ROWS*COLS, 3))
clus = cl.KMeans(nclusters)
clus.fit(colors)
newcolors = np.uint8(clus.predict(colors))
newimg = newcolors.reshape((ROWS, COLS))
pcolor = cv2.applyColorMap(cv2.equalizeHist(newimg), cv2.COLORMAP_RAINBOW)
cv2.imwrite('C:\\Data\\ColorSeg\\INPUT.png',img)
cv2.imwrite('C:\\Data\\ColorSeg\\NEW.png', pcolor)
```

Segmentation as clustering

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

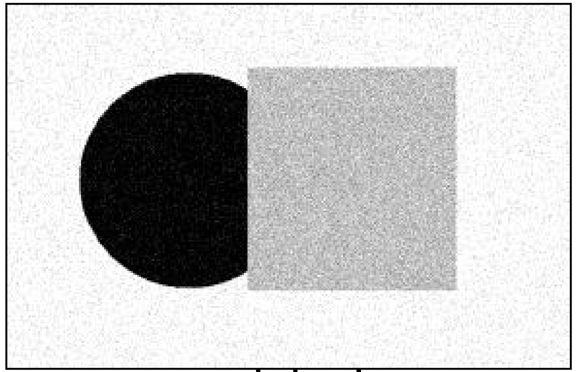
Grouping pixels based on **intensity** similarity

Clusters based on intensity similarity don’t have to be spatially coherent.

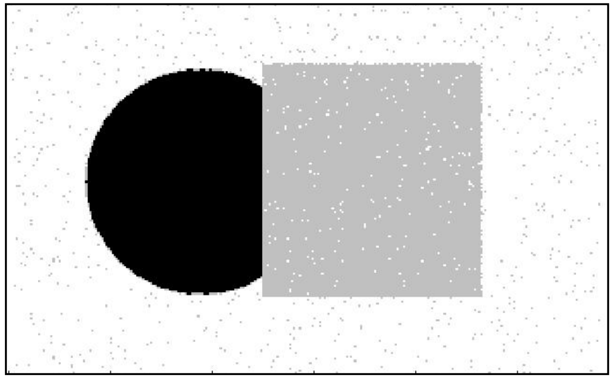
Kristen Grauman

An aside: Smoothing out cluster assignments

- Assigning a cluster label per pixel may yield outliers:



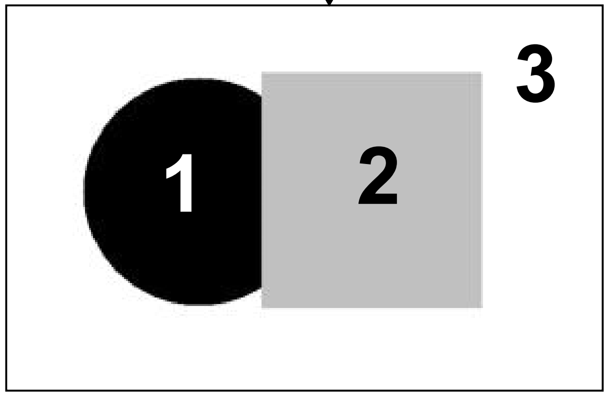
original



labeled by cluster center's intensity

?

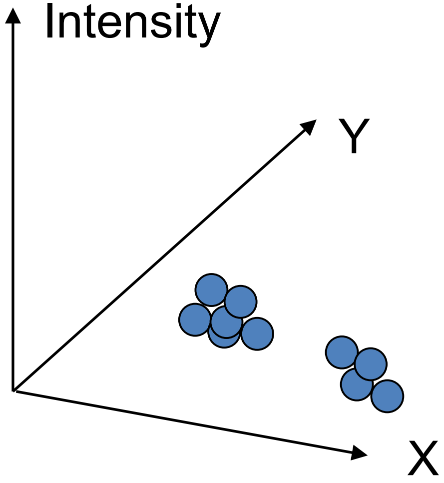
- How to ensure they are spatially smooth?

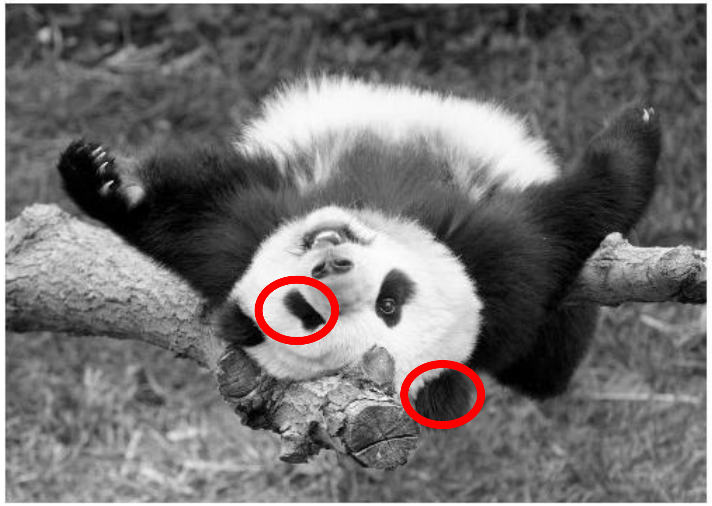


Segmentation as clustering

- Clusters based on intensity similarity don't have to be spatially coherent
- The feature space could be designed to include <u>proximity</u>

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

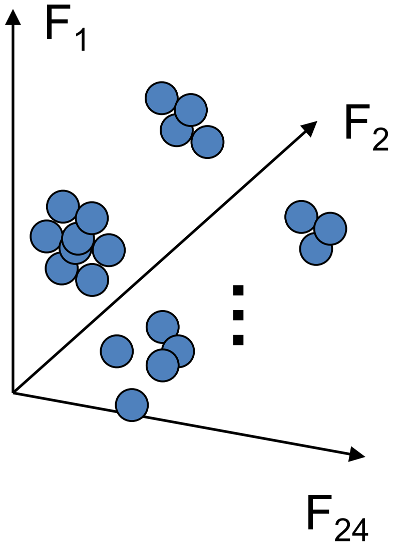
Clustering on color and position gives different results than clustering on color alone; regions are connected

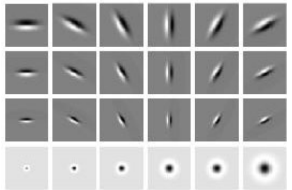
```python
import numpy as np
import cv2
import sklearn.cluster as cl

# Load an image and segment on color and position
img = cv2.imread('C:\\Data\\spock.jpg')
(ROWS, COLS, PLANES) = img.shape
posScale = 0.25
features = np.zeros((ROWS*COLS, 5), np.uint16)
part1 = np.arange(0, COLS)
part2 = np.arange(0, ROWS)
gridd = np.meshgrid(part1, part2)
features[:, 0:3] = img.reshape((ROWS*COLS, 3))
features[:, 3] = posScale * gridd[0].reshape((ROWS*COLS))
features[:, 4] = posScale * gridd[1].reshape((ROWS*COLS))
clus = cl.KMeans(nclusters)
clus.fit(features)
newfeatures = np.uint8(clus.predict(features))
newAugmentedImage = newfeatures.reshape((img.shape[0], img.shape[1]))
pcolor2 = cv2.applyColorMap(cv2.equalizeHist(newAugmentedImage), cv2.COLORMAP_RAINBOW)

cv2.imwrite('C:\\Data\\ColorSeg\\NEW2.png', pcolor2)
combined = np.concatenate((img, pcolor, pcolor2), 1)
fname = "COMBINED_ncl-" + str(nclusters) + "_scl-" + str(posScale)
cv2.imwrite("C:\\Data\\ColorSeg\\" + fname + ".png", combined)
```

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

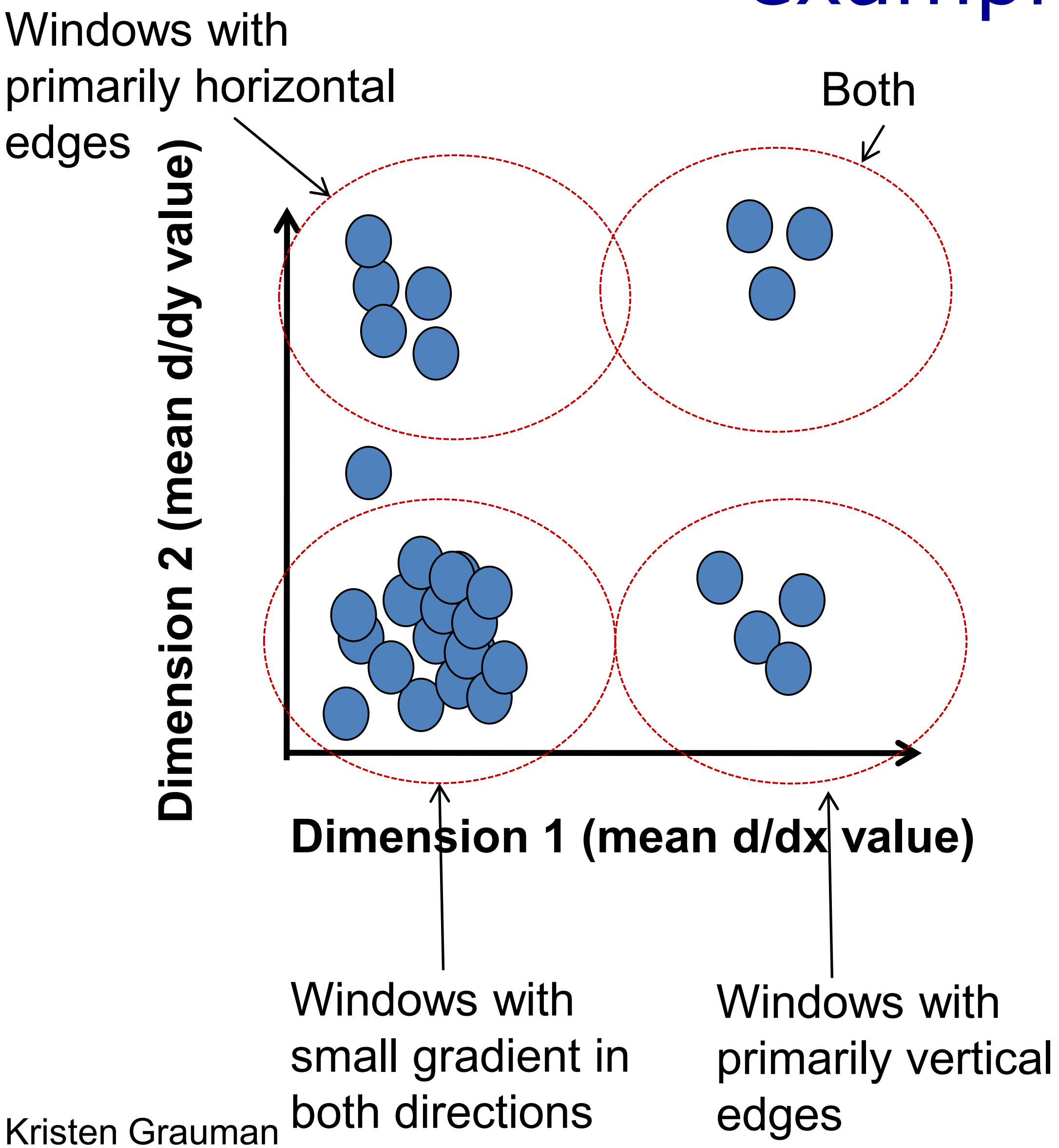




Filter bank of 24 filters

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

As a preview: 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

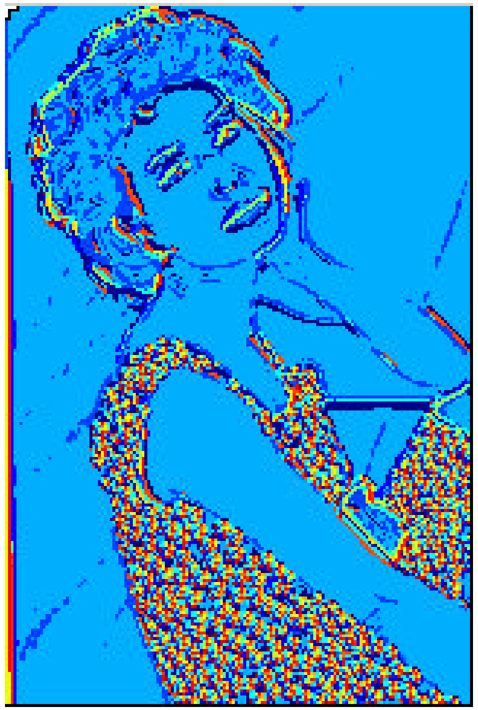
Kristen Grauman

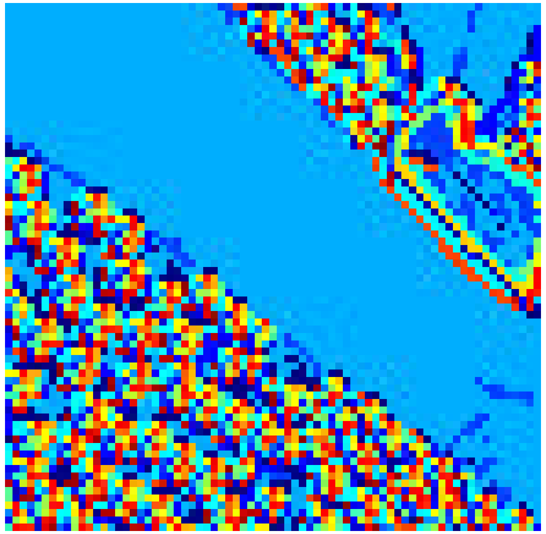
Segmentation with texture features

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

Image

Texton map





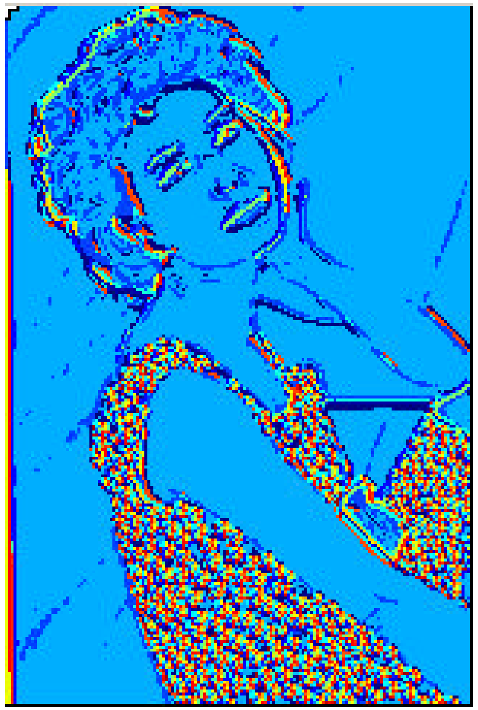
Malik, Belongie, Leung and Shi. IJCV 2001.

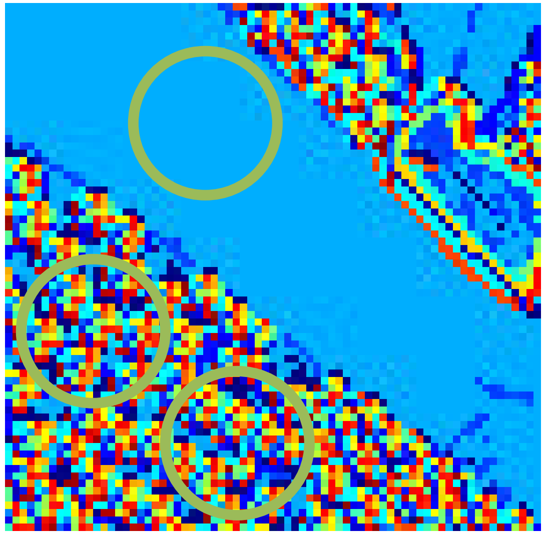
Segmentation with texture features

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

Image

Texton map





Malik, Belongie, Leung and Shi. IJCV 2001.

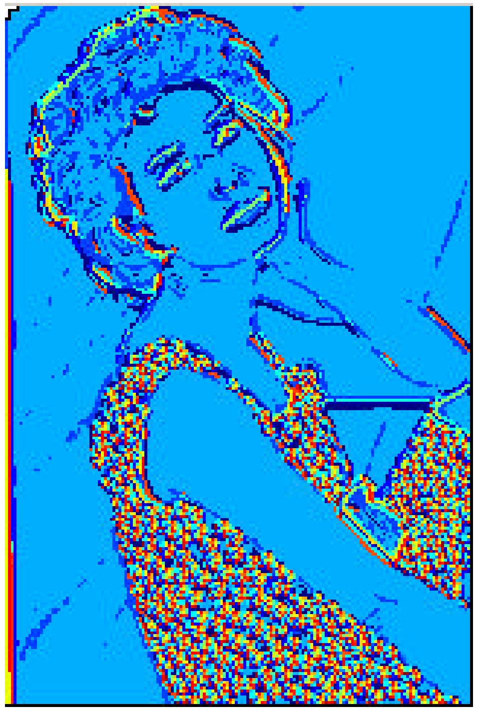
Adapted from Lana Lazebnik

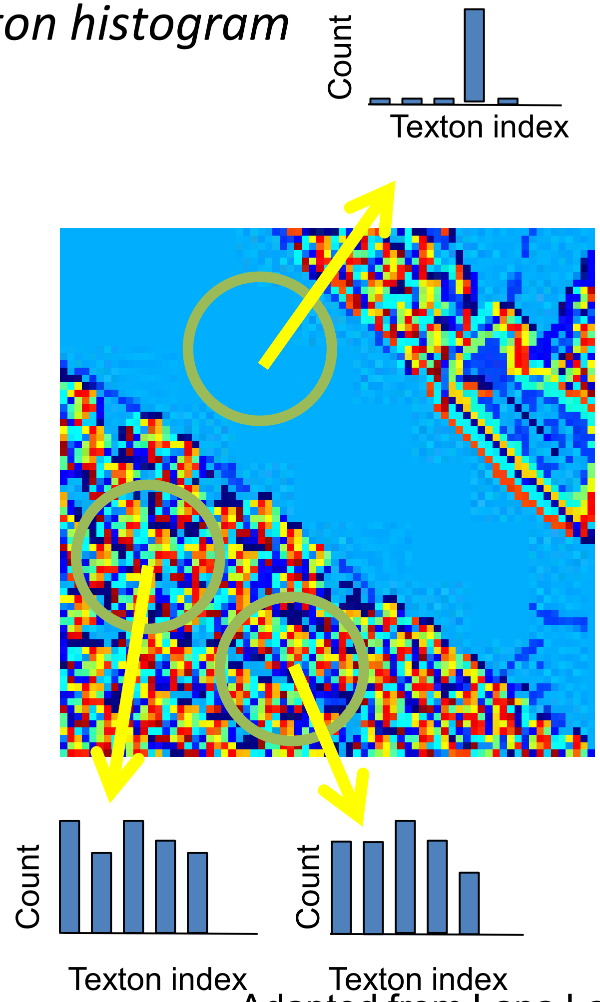
Segmentation with texture features

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

Image

Texton map

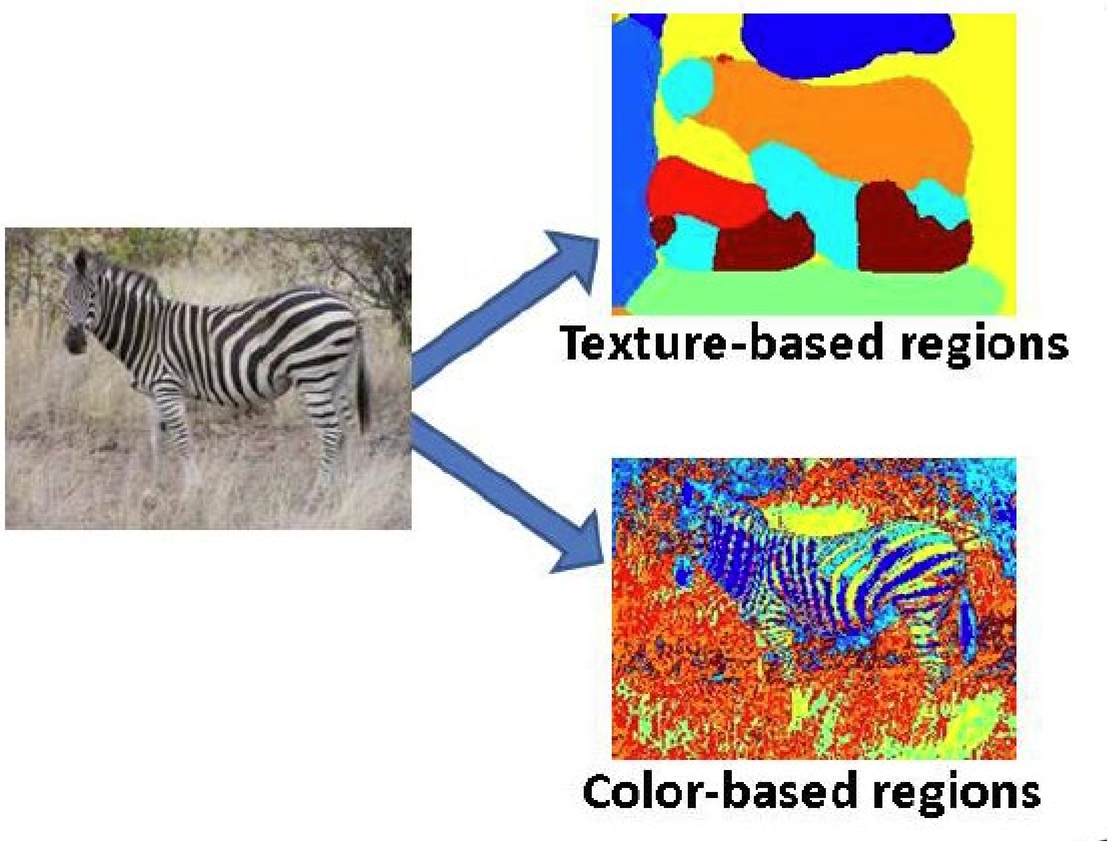




Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

For the same image, different segmentation strategies will define different regions; some may be irrelevant to what *we* think the object boundaries are



Kristen Grauman

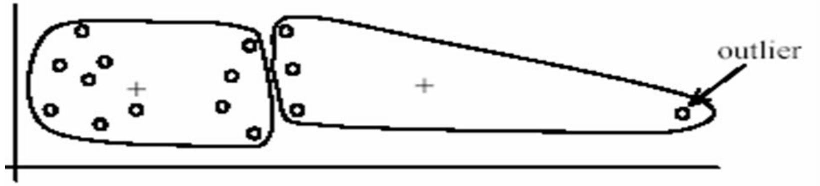
K-means: pros and cons

Pros

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

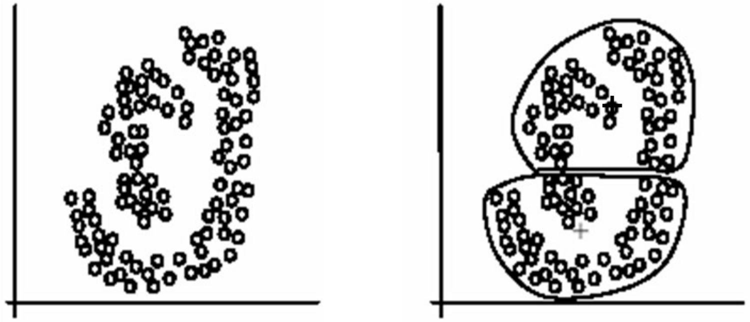
Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- **Detects spherical clusters**
- Assuming means can be computed



(A): Undesirable clusters

(B): Ideal clusters



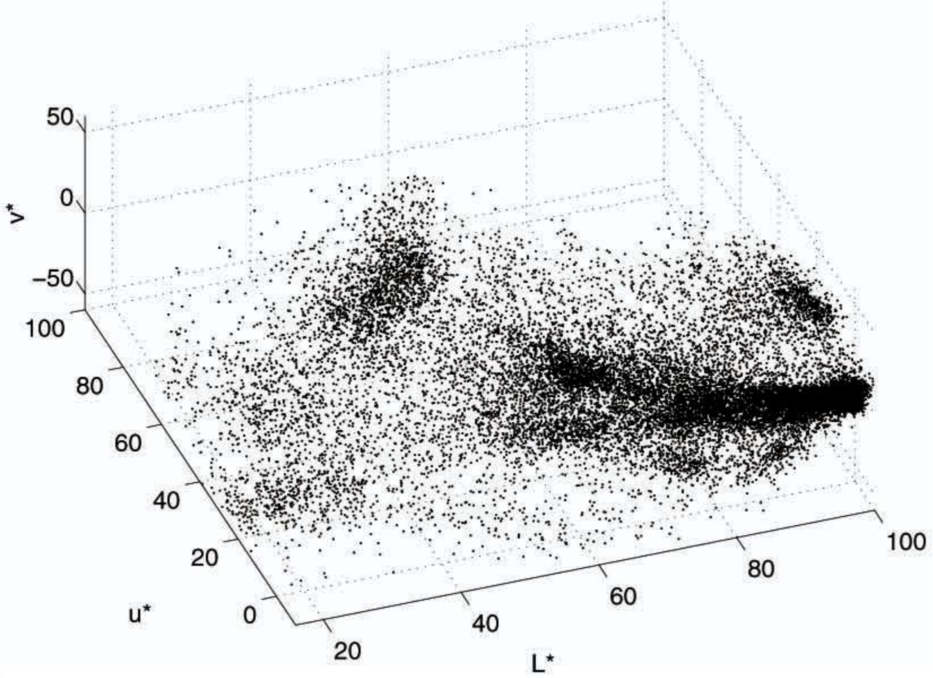
(A): Two natural clusters (B): *k*-means clusters

Mean shift algorithm

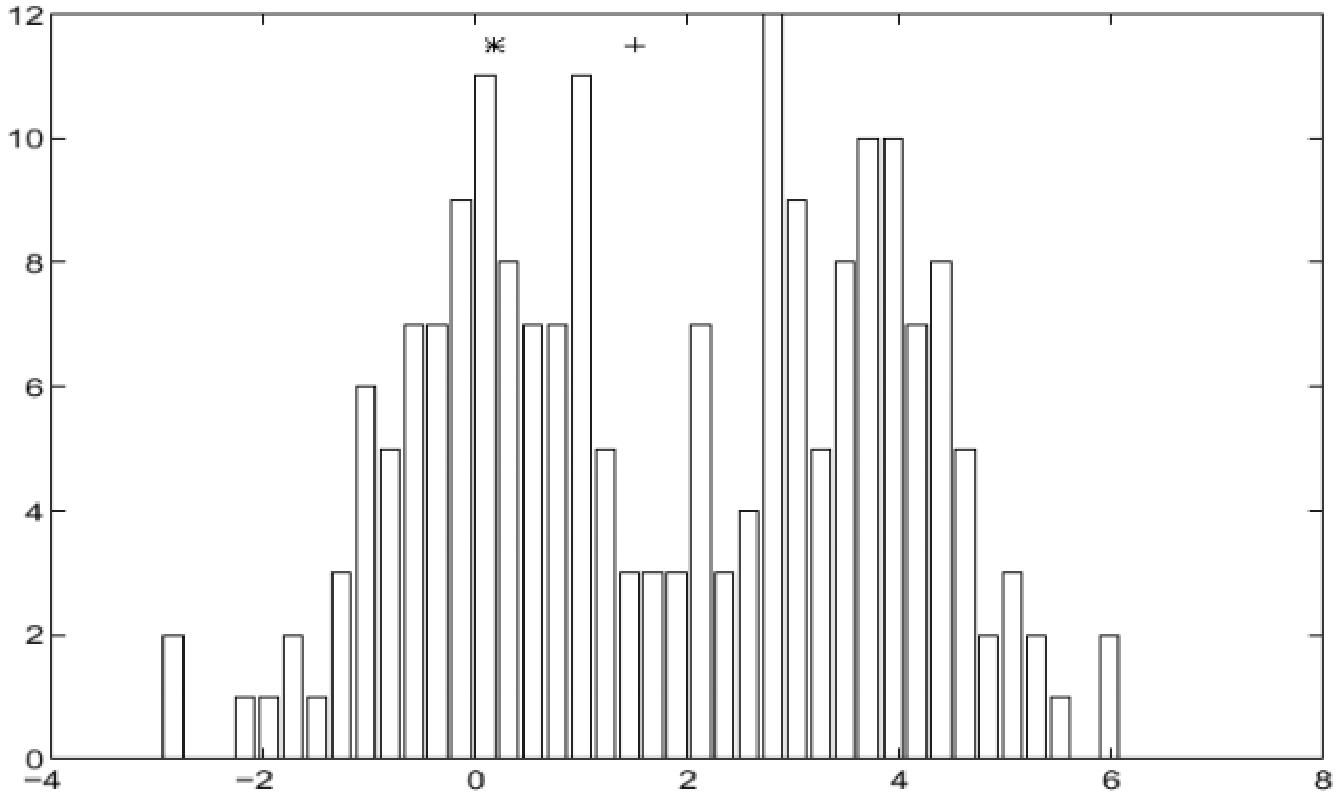
- The mean shift algorithm seeks local maxima ("modes") of density in the feature space

Image

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



Mean-Shift Algorithm

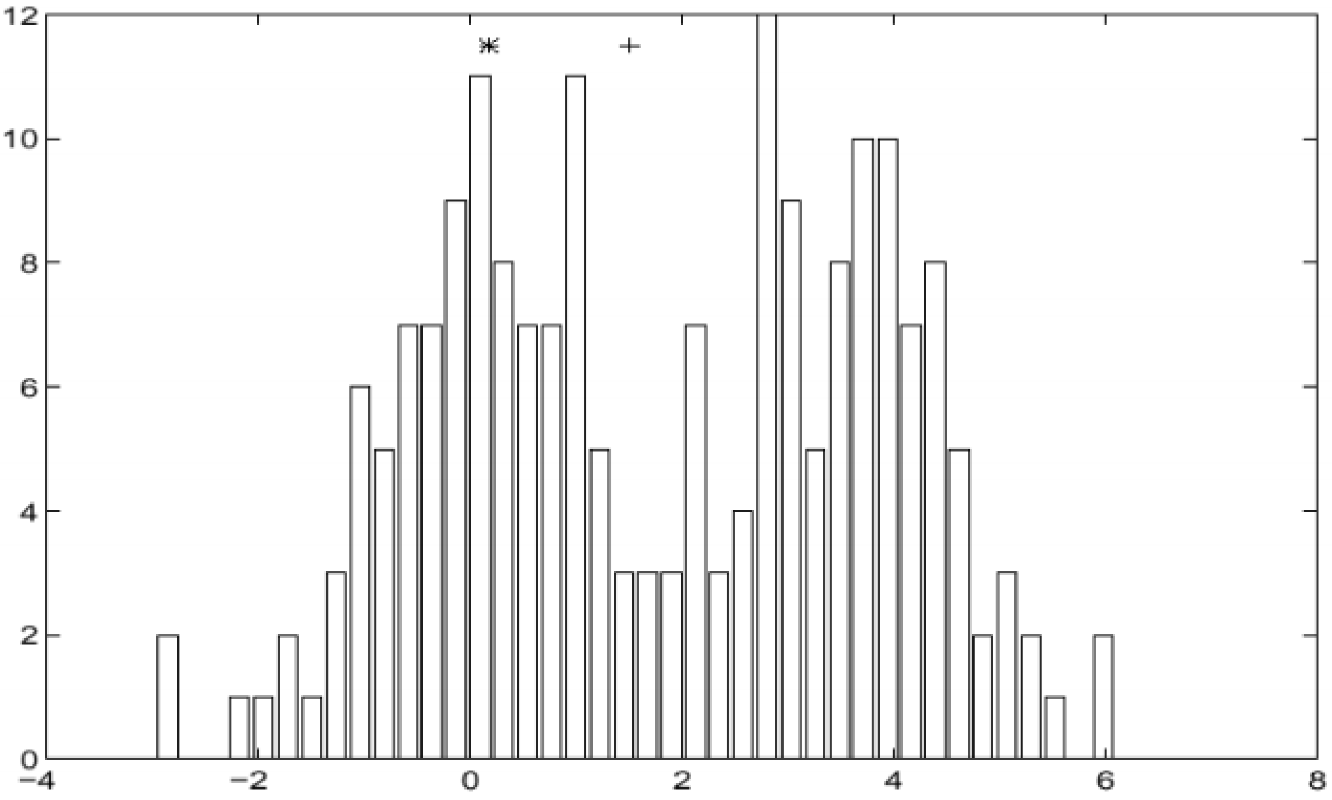


- Iterative Mode Search
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} xH(x)$
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

Slide credit: Steve Seitz

http://vision.stanford.edu/teaching/cs131_fall1314_nope/

Mean-Shift Algorithm



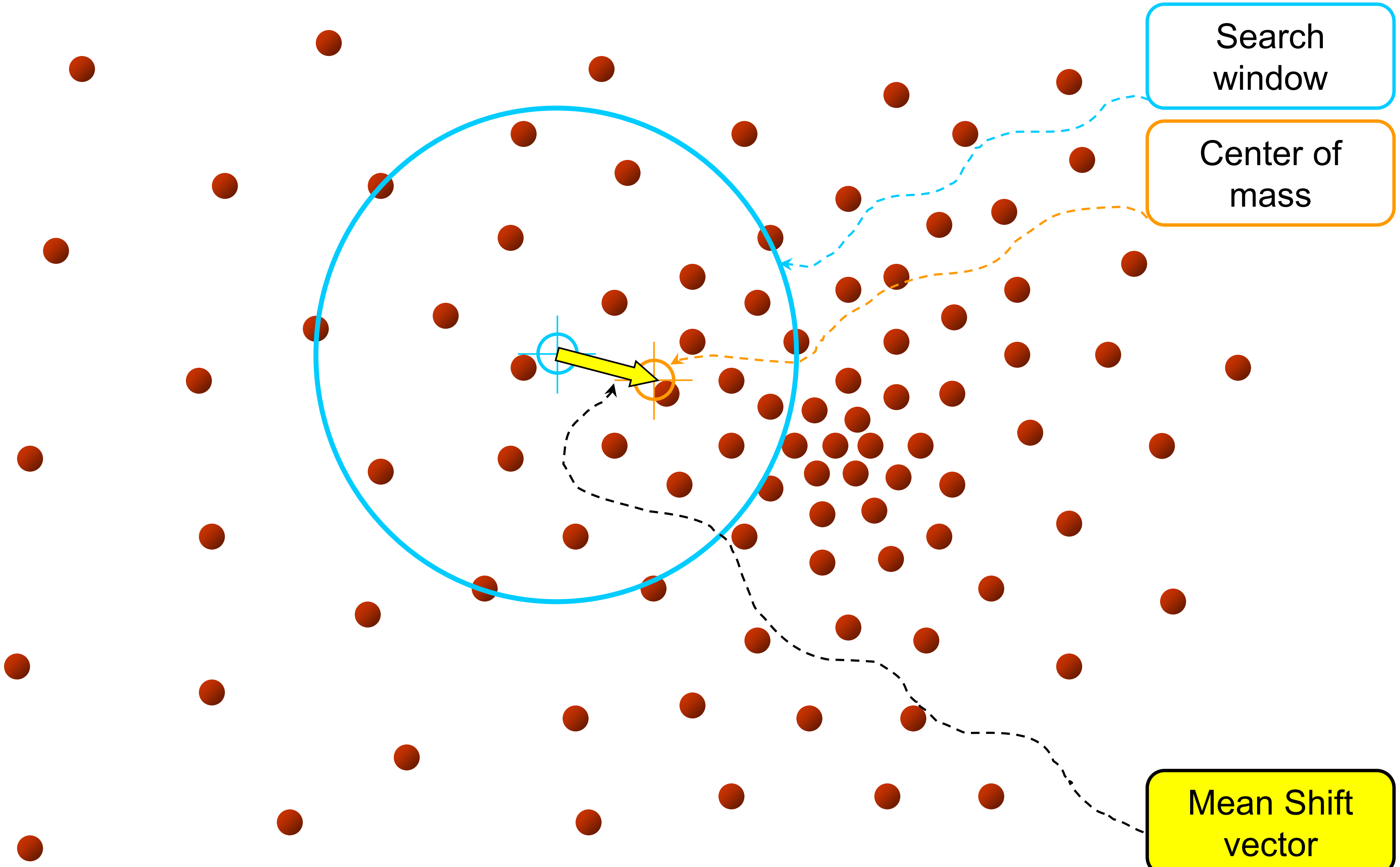
- Iterative Mode Search
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} xH(x)$
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

Note the importance of the window W
- **Size**
- **Need not be square**
- **Need not be circular**

Slide credit: Steve Seitz

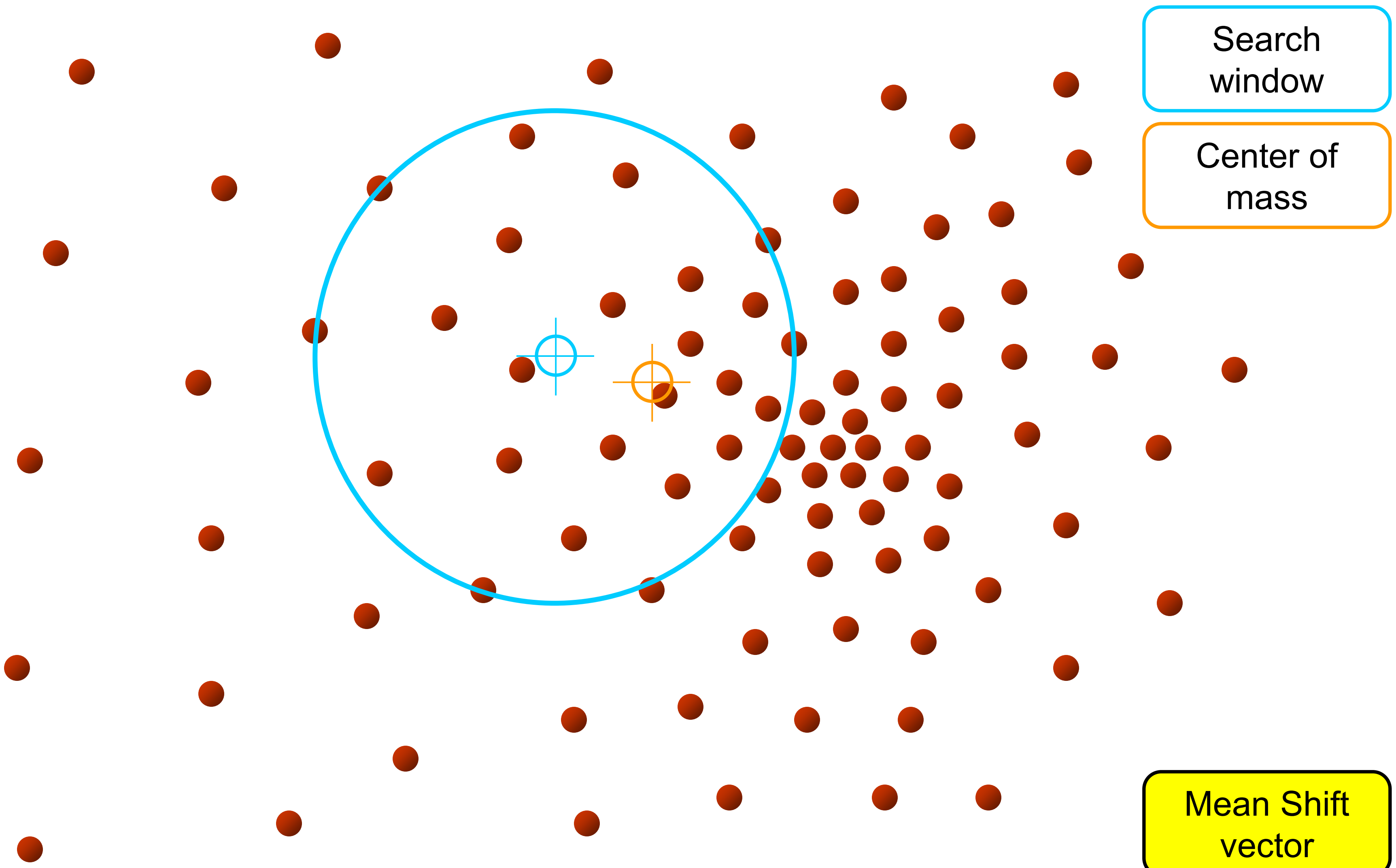
http://vision.stanford.edu/teaching/cs131_fall1314_nope/

Mean shift



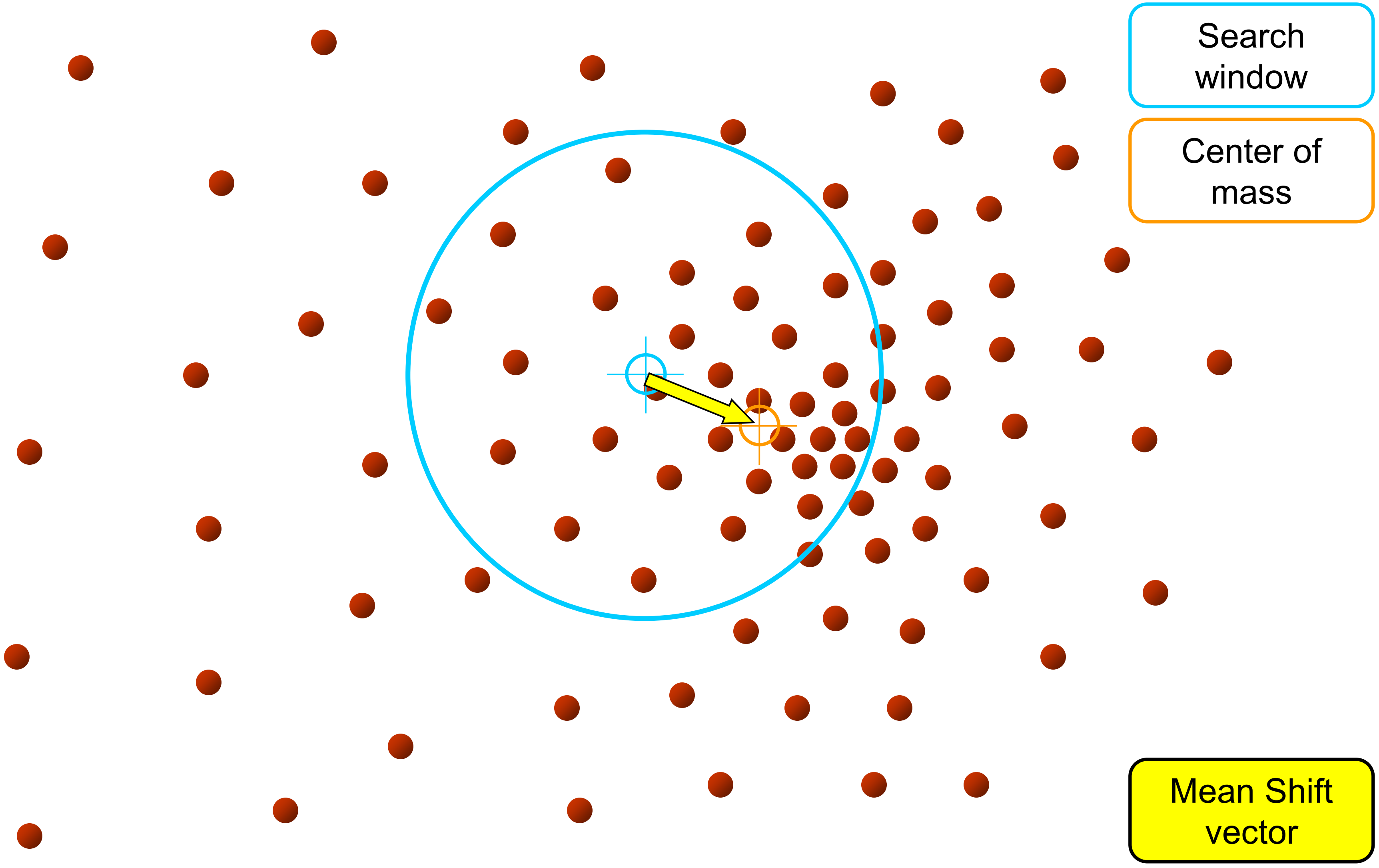
Slide by Y. Ukrainitz & B. Sarel

Mean shift



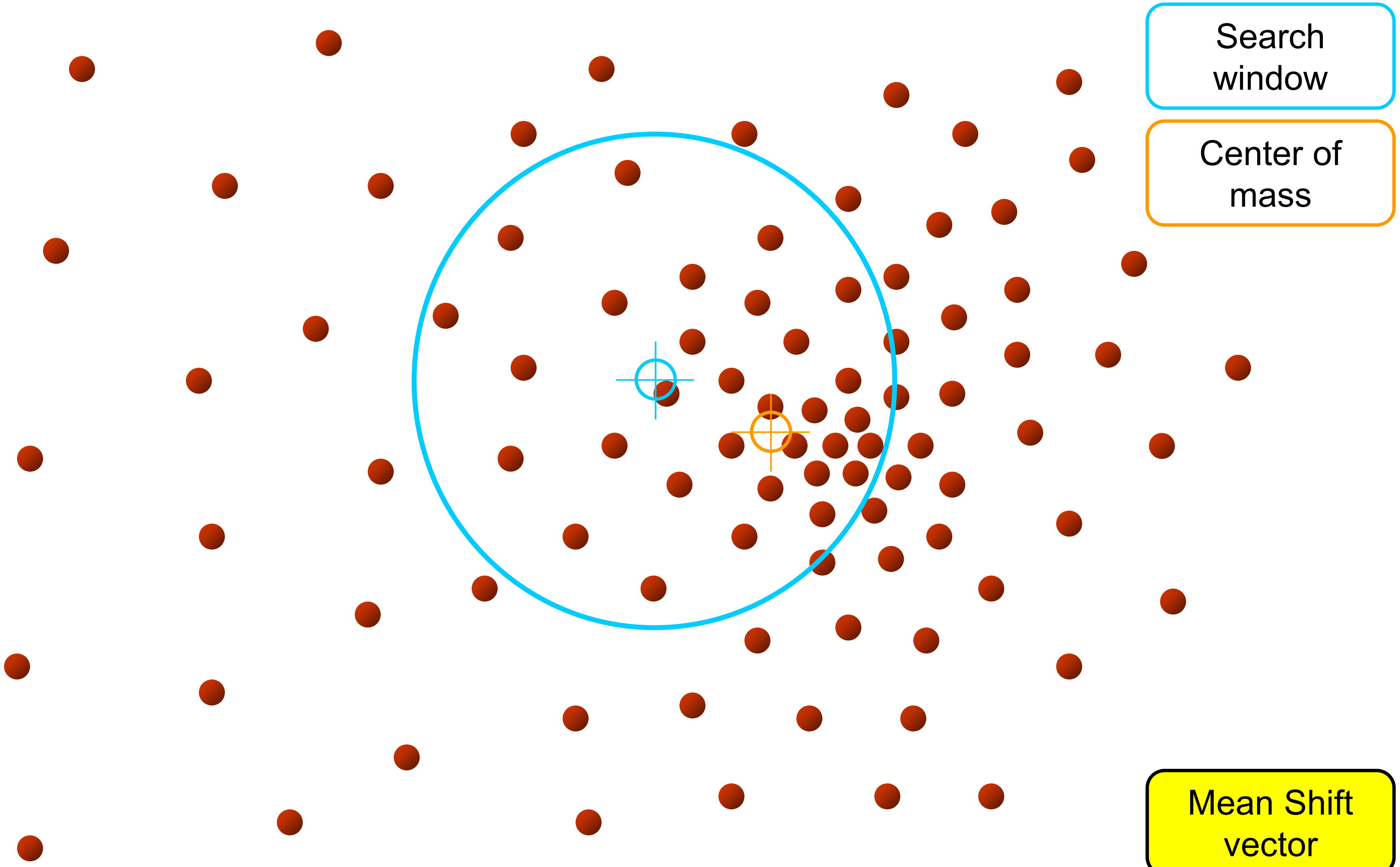
Slide by Y. Ukrainitz & B. Sarel

Mean shift



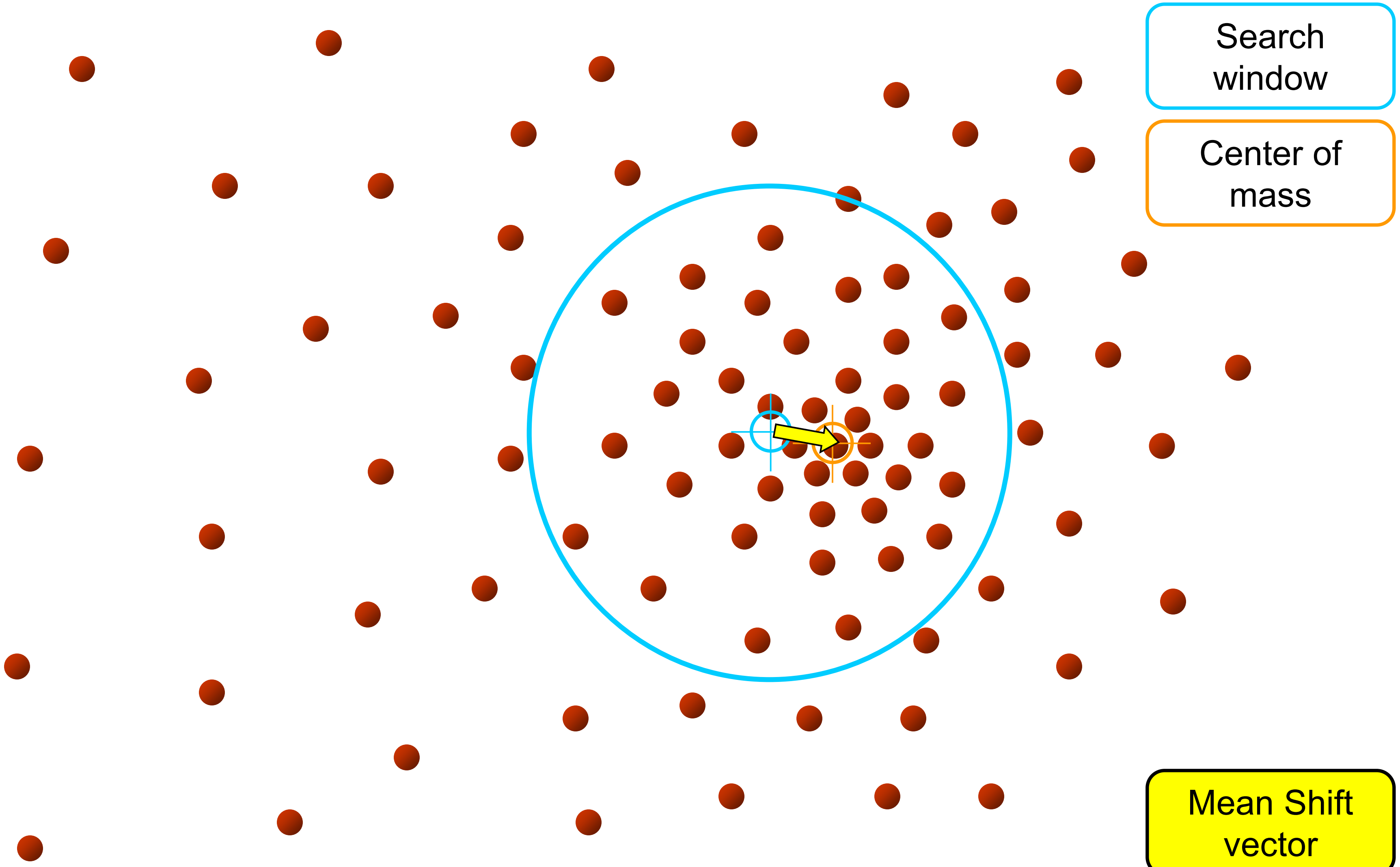
Slide by Y. Ukrainitz & B. Sarel

Mean shift



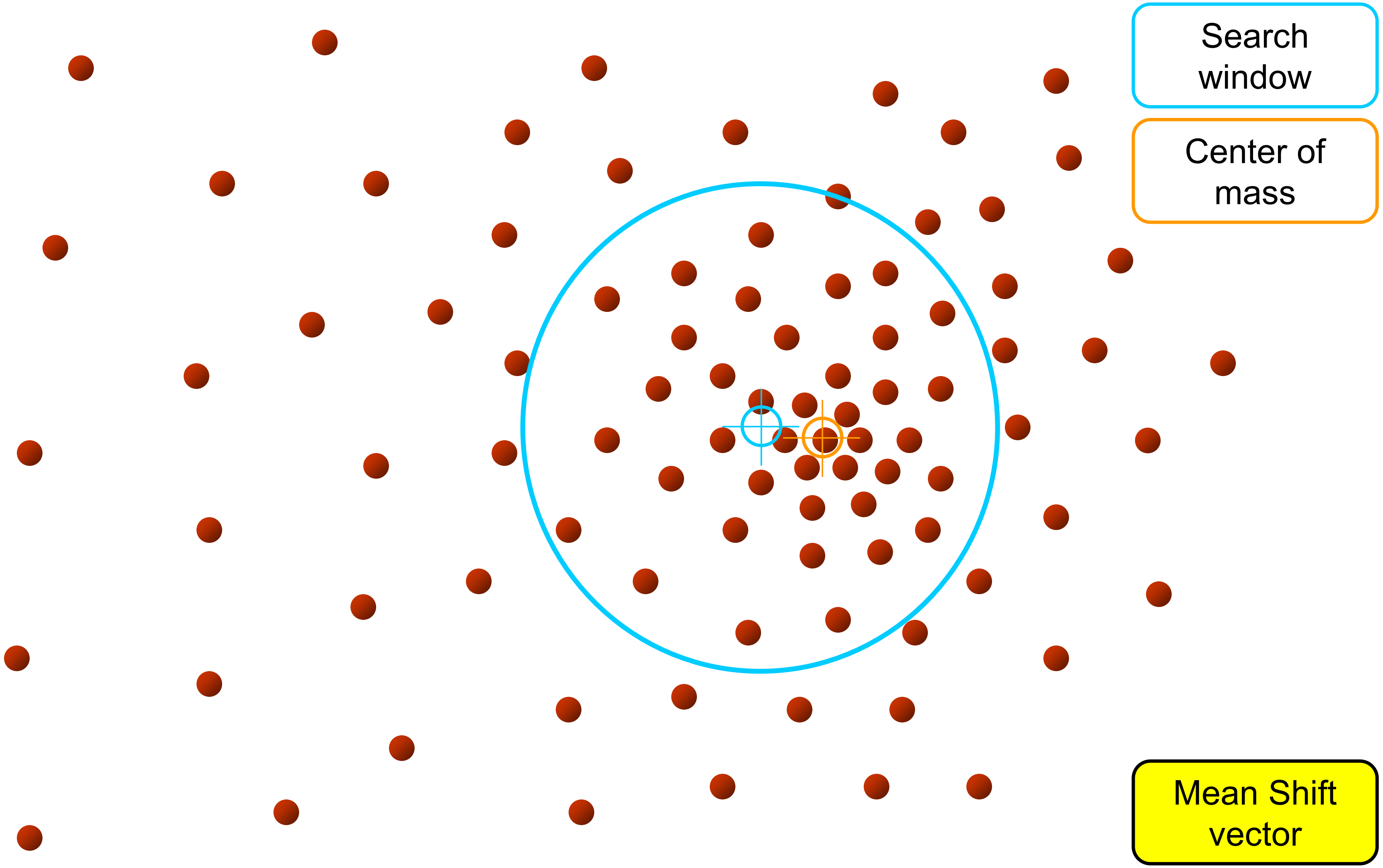
Slide by Y. Ukrainitz & B. Sarel

Mean shift



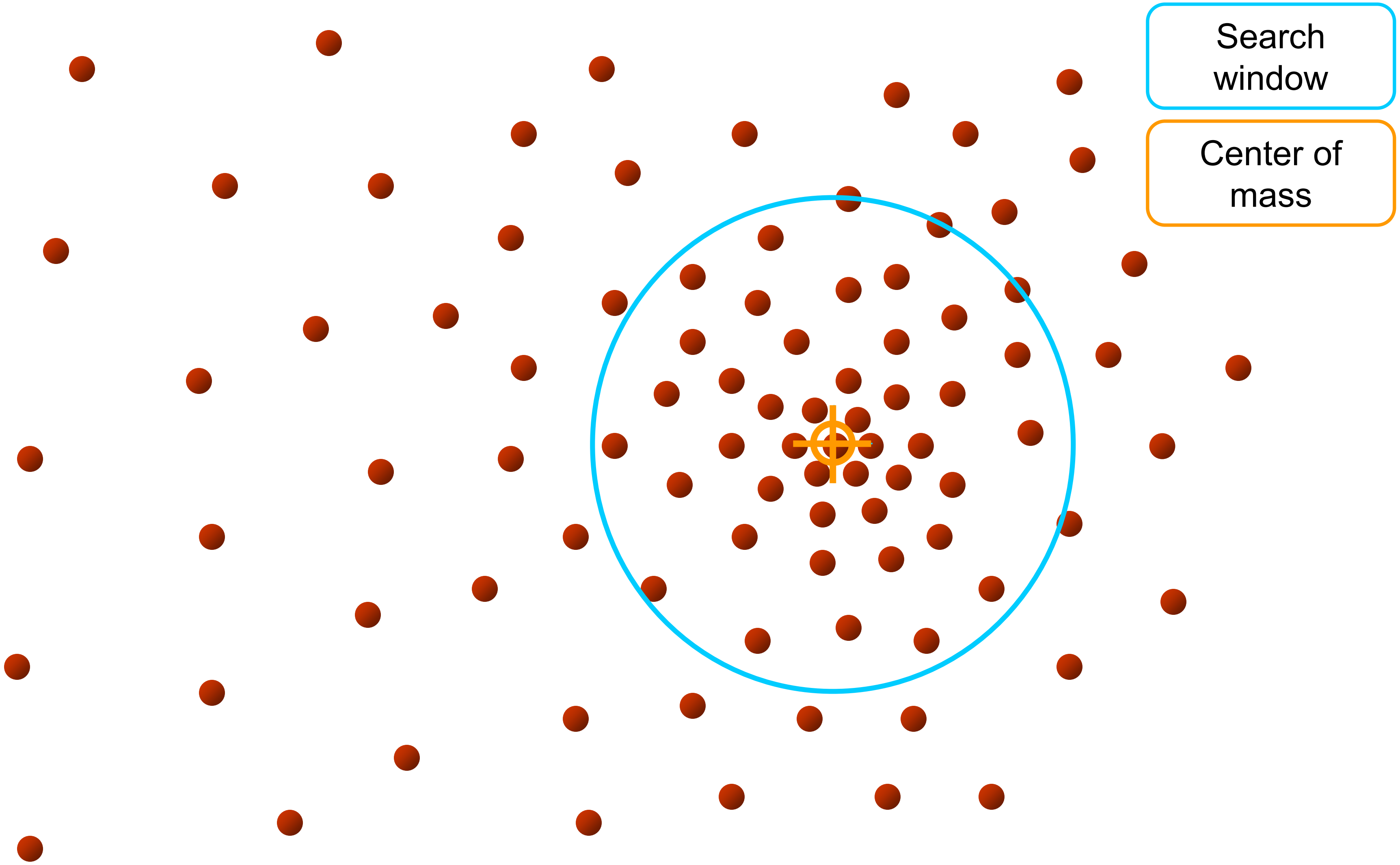
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

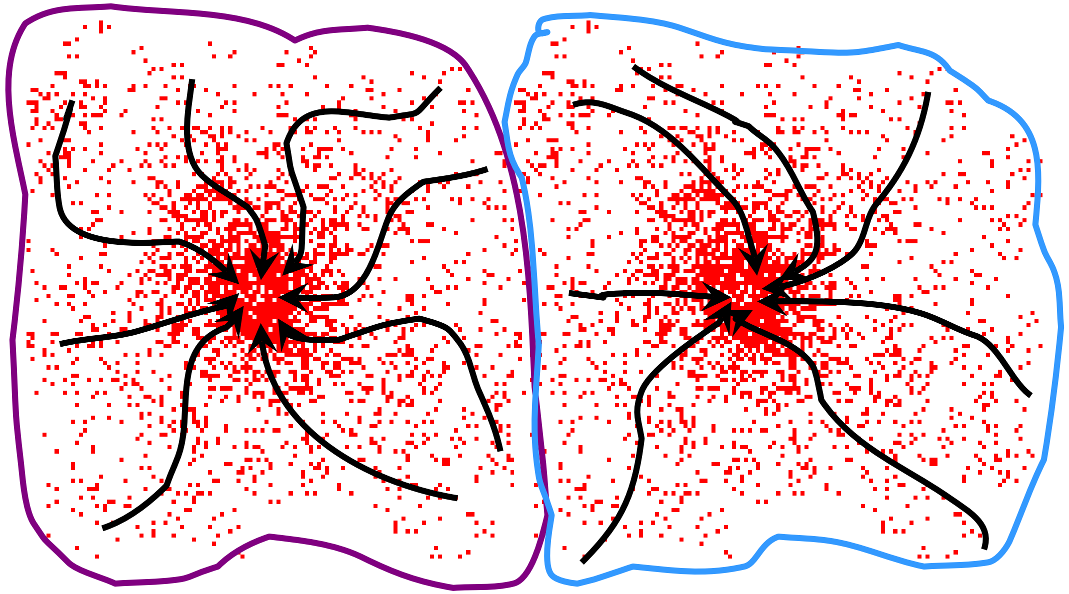
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

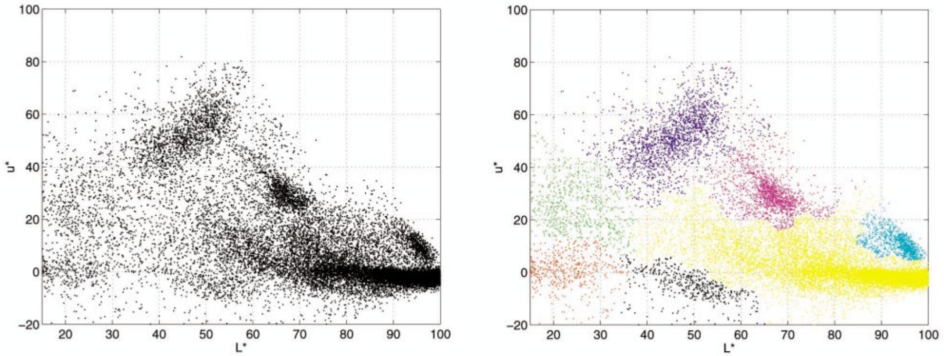
- Cluster: all data points in the *attraction basin* of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



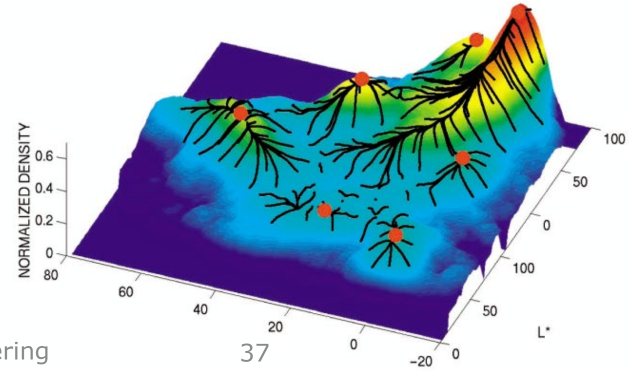
Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering/segmentation

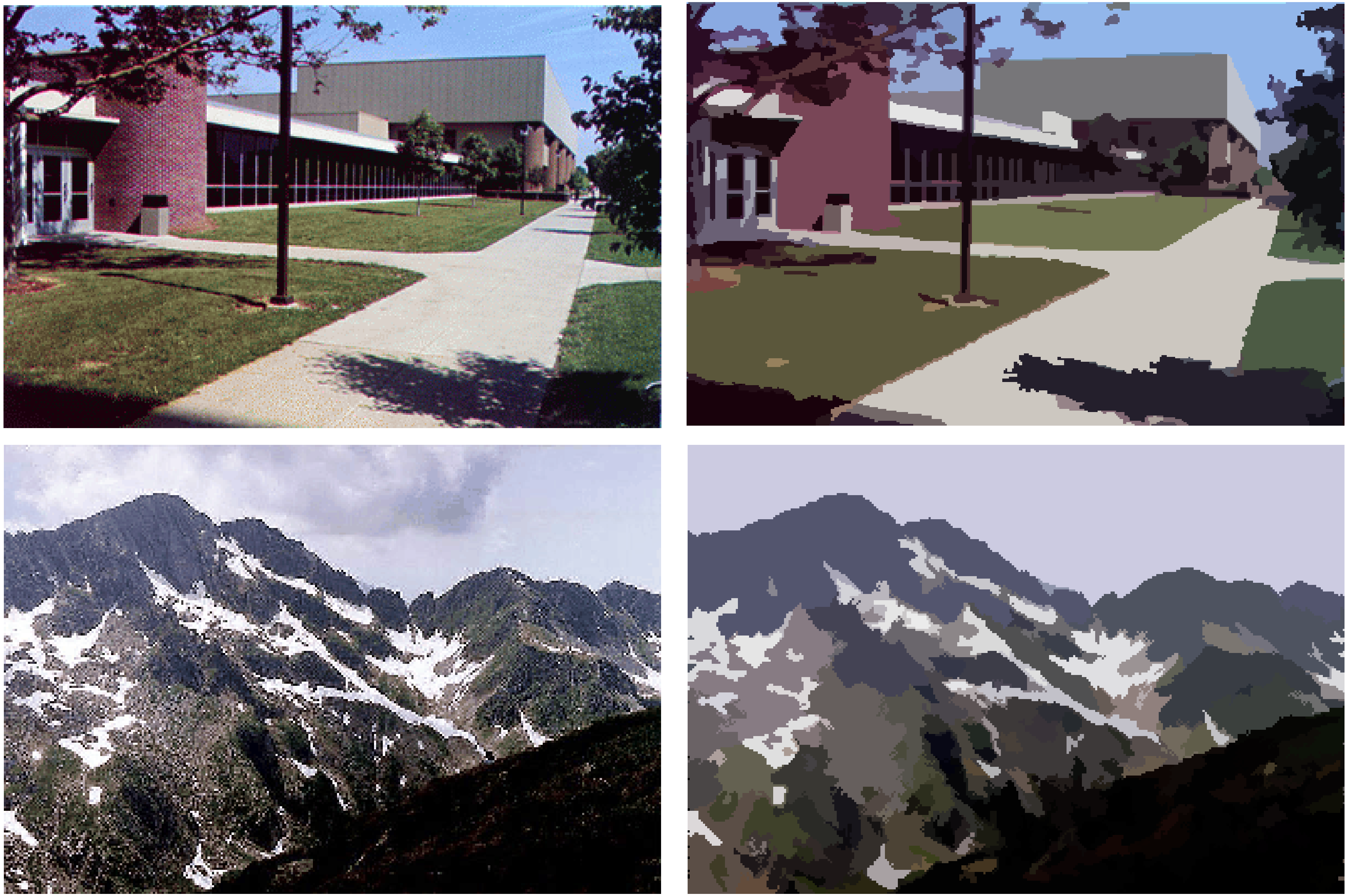
- Find features (color, gradients, texture, etc.)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same "peak" or mode



(a) (b)



Mean shift segmentation results (note color mapping)



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

Mean shift segmentation results (outlines of regions)

Mean shift – pros and cons

- 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

Kristen Grauman

Today's Objectives

Segmentation by Clustering

- Intensity clustering
- Brief discussion of RGB space
- Color space clustering using Kmeans
- Texture clustering
- The Mean-shift approach