Image Segmentation

Some slides: courtesy of O. Capms, Penn State,
J.Ponce and D. Fortsyth, Computer Vision Book

Regions and Edges

- Ideally, regions are bounded by closed contours
  - We could "fill" closed contours to obtain regions
  - We could "trace" regions to obtain edges
- Unfortunately, these procedures rarely produce satisfactory results.

Regions and Edges

- **Edges** are found based on **DIFFERENCES** between values of adjacent pixels.
- **Regions** are found based on **SIMILARITIES** between values of adjacent pixels.
- Goal associate some higher level – more meaningful units with the regions of the image

Segmentation

- Useful mid-level representation of an image - can facilitate better further tasks
- Partitioning image into regions should be homogeneous with respect to some characteristic
- (gray level, texture, color, motion)
- The type of desired segmentation depends on the task

- Variety of approaches/algorithms
- segmentation mid-level representation for higher level tasks
- Applications - finding people, summarizing video, annotation figures, background subtraction, finding buildings/rivers in satellite images

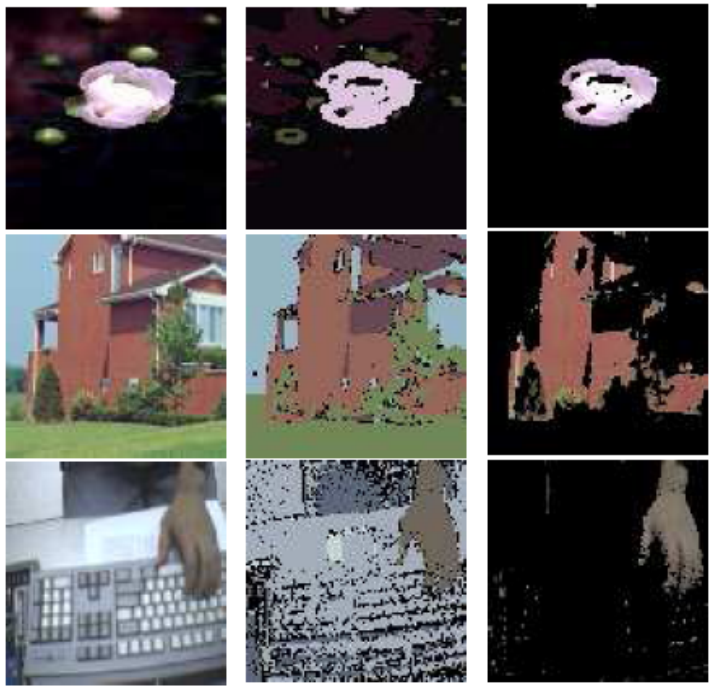
Image Segmentation

- Today: few simple segmentation approaches
- Binary image segmentation
- Segmentation with k-means clustering
- Motion segmentation

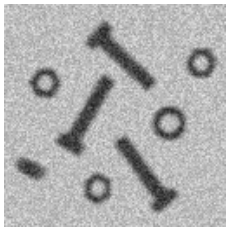
- Later:
- Graph Based Segmentation
- Graph Cut Segmentation

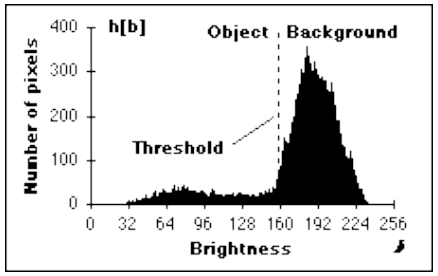
Segmentation grouping

- Segmentation and grouping
- Group pixels based on some similarity
- Group video to shots
- Object –level grouping (find cars, bikes)
- Determine image regions belonging to objects
- Group foreground/background pixels
- Interactive foreground/background segm.

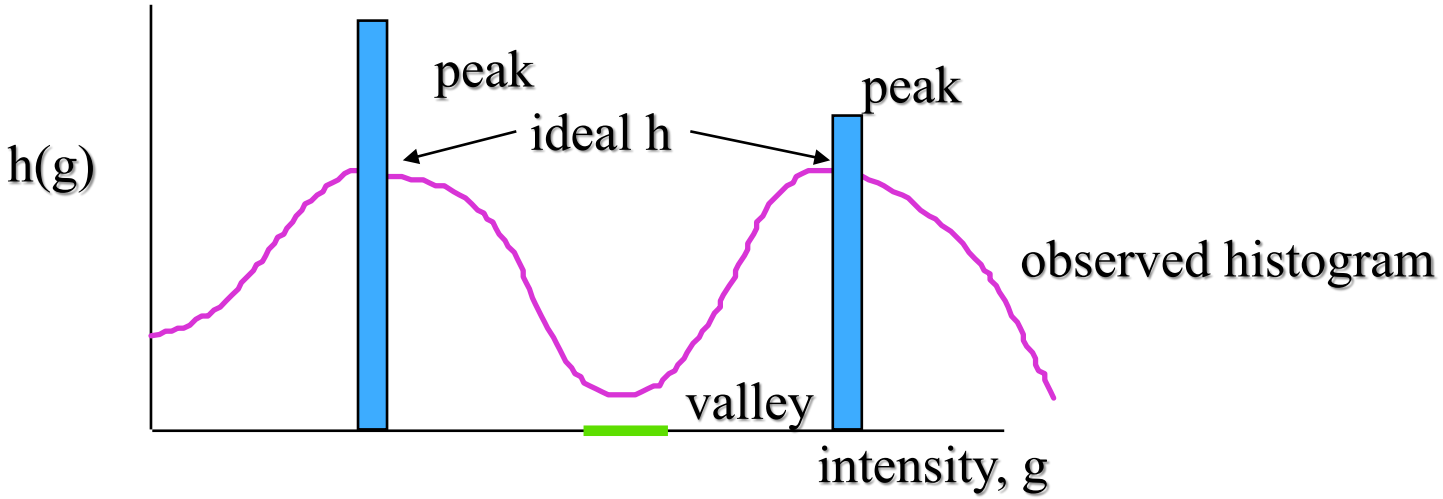


Binary segmentation



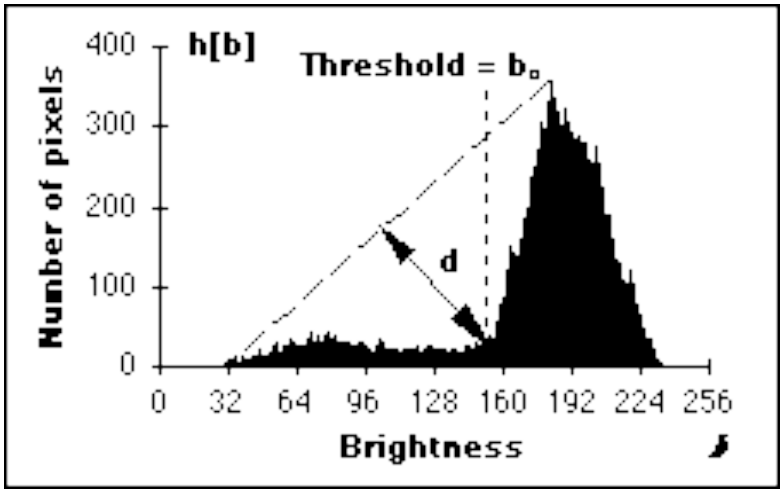


- Segmentation for simple binary images
- How do we choose the threshold t for segmentation?
- Histogram (h) - gray level frequency distribution of the gray level image F.
  - $h_F(g)$ = number of pixels in F whose gray level is g
  - $H_F(g)$ = number of pixels in F whose gray level is <=g



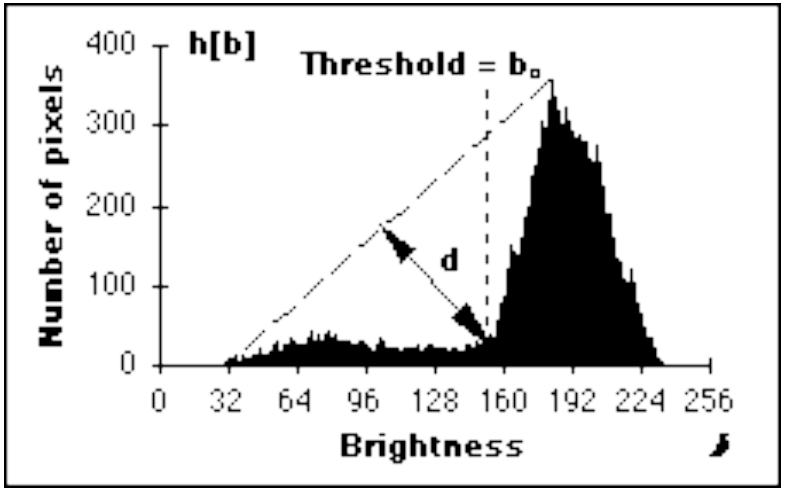
Thresholding

- Peak and valley method
  - Find the two most prominent peaks of h
    - g is a peak if $h_F(g) > h_F(g \pm \Delta g)$, $\Delta g = 1, \ldots, k$
  - Let g_1 and g_2 be the two highest peaks, with $g_1 < g_2$
  - Find the deepest valley, g, between g_1 and g_2
    - g is the valley if $h_F(g) <= h_F(g')$, g,g' in $[g_1, g_2]$
  - Use g as the threshold



Triangle algorithm

- A line is constructed between the maximum of the histogram at brightness b_{max} and the lowest value b_{min} = (p=0)% in the image.
- The distance d between the line and the histogram h[b] is computed for all values of b from b = b_{min} to b = b_{max}.
- The brightness value b_o where the distance between h[b_o] and the line is maximal is the threshold value.
- This technique is particularly effective when the object pixels produce a weak peak in the histogram.



Thresholding

- Hand selection
  - select a threshold by hand at the beginning of the day
  - use that threshold all day long!
- Many threshold selection methods in the literature
  - Probabilistic methods
    - make parametric assumptions about object and background intensity distributions and then derive "optimal" thresholds
  - Structural methods
    - Evaluate a range of thresholds wrt properties of resulting binary images
      - one with straightest edges, most easily recognized objects, etc.
  - Local thresholding
    - apply thresholding methods to image windows

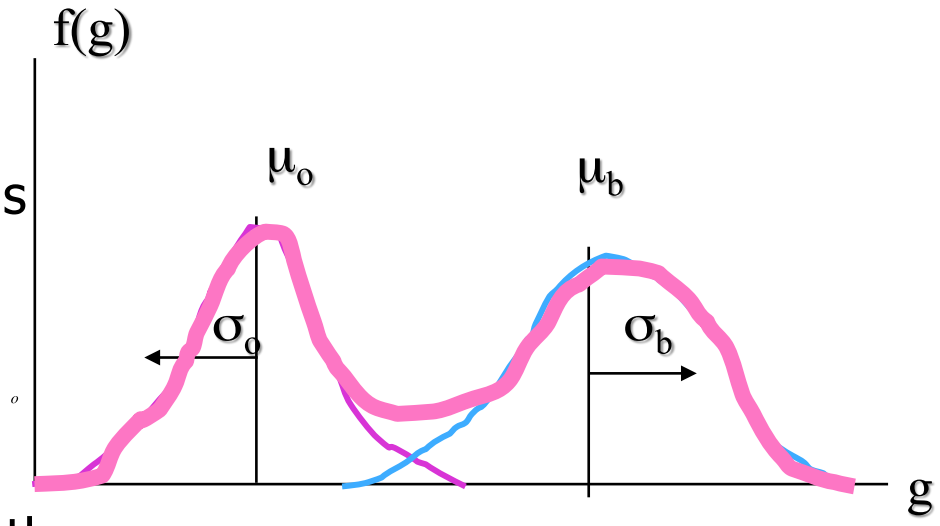
An advanced probabilistic threshold selection method - minimizing Kullback information distance

- Suppose the observed histogram, f, is a mixture of the gray levels of the pixels from the object(s) and the pixels from the background
  - in an ideal world the histogram would contain just two peaks (this depends of the class of images/objects)
  - but
    - measurement noise
    - model noise (e.g., variations in ink density within a character)
    - edge blur (misalignment of object boundaries with pixel boundaries and optical imperfections of camera)

  spread these spikes out into hills

Kullback information distance

- Make a parametric model of the shapes of the component histograms of the objects(s) and background
- Parametric model - the component histograms are assumed to be Gaussian
  - p_o and p_b are the proportions of the image that comprise the objects and background
  - μ_o and μ_b are the mean gray levels of the objects and background
  - σ_o and σ_b- are their standard deviations



$$f_o(g) \quad \frac{p_o}{\sqrt{2}\,\sigma_o} e^{1/2\left(\frac{g - \mu_o}{\sigma_o}\right)^2}$$

$$f_b(g) \quad \frac{p_b}{\sqrt{2}\,\sigma_b} e^{1/2\left(\frac{g - \mu_b}{\sigma_b}\right)^2}$$

Kullback information distance

- Now, if we hypothesize a threshold, t, then all of these unknown parameters can be approximated from the image histogram.
- Let f(g) be the observed and normalized histogram
  - f(g) = percentage of pixels from image having gray level g

$$p_o(t) \quad \sum_{g=0}^{t} f(g) \qquad p_b(t) \quad 1 \quad p_0(t)$$

$$_o(t) \quad \sum_{g=0}^{t} f(g)g \qquad _b(t) \quad \sum_{g=t+1}^{\max} f(g)g$$

Kullback information distance

- So, for any hypothesized t, we can "predict" what the total normalized image histogram **should** be if our model (mixture of two Gaussians) is correct.
  - $P_t(g) = p_o f_o(g) + p_b f_b(g)$
- The total normalized image histogram is **observed to be** f(g)
- So, the question reduces to:
  - determine a suitable way to measure the <u>similarity</u> of P and f
  - then search for the t that gives the highest similarity

Kullback information distance

- A suitable similarity measure is the Kullback directed divergence, defined as

  If P_t matches f exactly, then each term of the sum is 0 and K(t) takes on its minimal value of 0
- Gray levels where P_t and f disagree are penalized by the log term, weighted by the importance of that gray level (f(g))

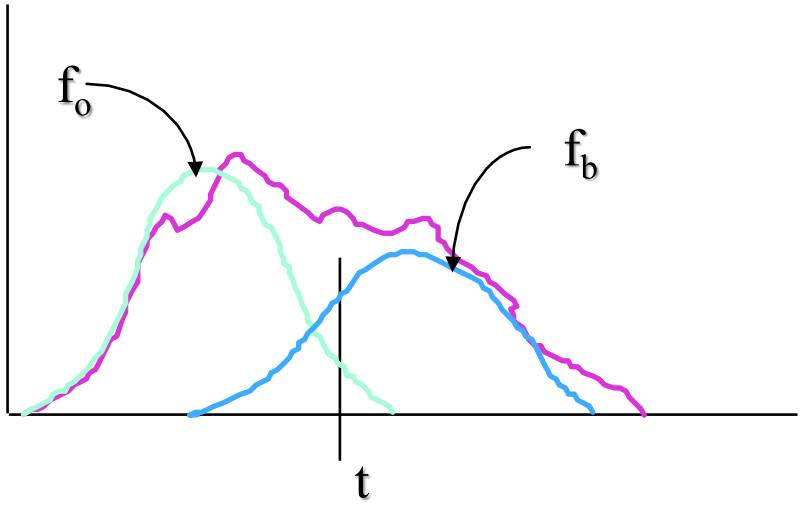
$$K(t) \sum_{g=0}^{\max} f(g) \log\left[\frac{f(g)}{P_t(g)}\right]$$

An alternative - minimize probability of error

- Using the same mixture model, we can search for the t that minimizes the predicted probability of error during thresholding
- Two types of errors
  - background points that are marked as object points. These are points from the background that are darker than the threshold
  - object points that are marked as background points. These are points from the object that are brighter than the threshold

An alternative - mimimize probability of error

- For each "reasonable" threshold
  - compute the parameters of the two Gaussians and the proportions
  - compute the two probability of errors
- Find the threshold that gives
  - minimal overall error
  - most equal errors



t

$$e_b(t) \quad p_b \quad f_b(g)$$

$$g = 0$$

max

$$e_o(t) \quad p_o \quad f_o(g)$$

$$g = t + 1$$

Segmentation by Clustering

Find set of clusters such that the least squares

Error is minimized

$$E = \sum_{k=1}^{K} \sum_{x_i \in C_i} \|x_i - m_k\|^2$$

Iterative K-means clustering algorithm

Set iter = 1;

1. Choose randomly K-means m1, … mk
2. For each data point xi, compute distance
   to each of the means and assign the point
   the cluster with the nearest mean

4. iter = iter + 1
5. Recompute the means based on the new assignments
   of points to clusters
6. Repeat 3-5 until the cluster centers do not change much

Image

Clusters on intensity

Clusters on color

K-means clustering using intensity alone and color alone

Image

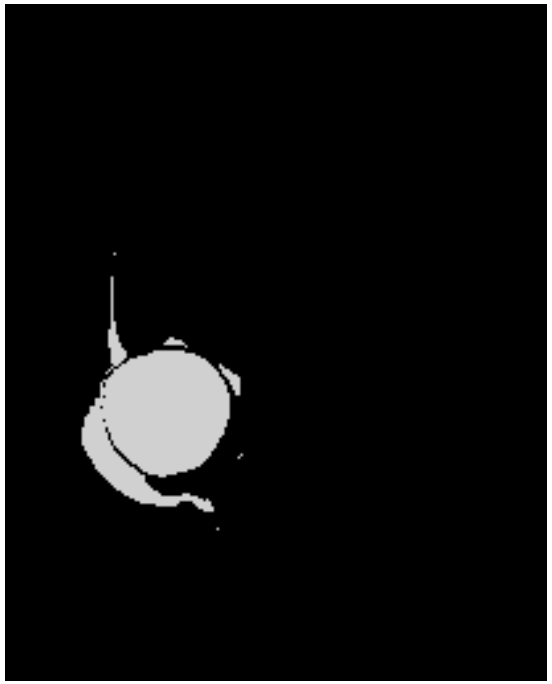
Clusters on color

K-means using color alone, 11 segments

K-means using
color alone,
11 segments.



K-means using colour and position, 20 segments

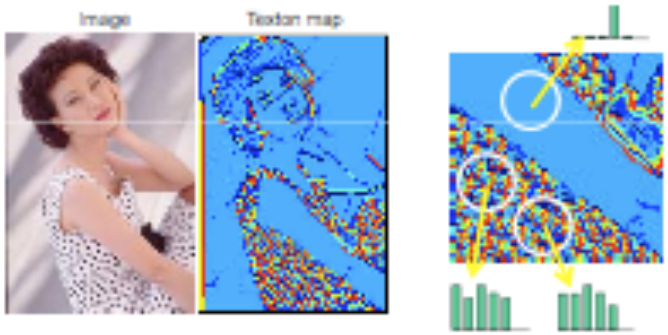
Clustering

- Pros
- simple, fast to compute
- If k is large (approximate nearest neighbour methods for computing distances to clusters)
- Converges to local minimum of within cluster squared error

- Cons
- How large is K ?
- Sensitive to initial outliers
- Detection of spherical clusters
- Assumes that means can be computed

- Issues: Depending what we choose as feature space we get different clusters (color, textures, motion etc)
- Clusters often not spatially coherent

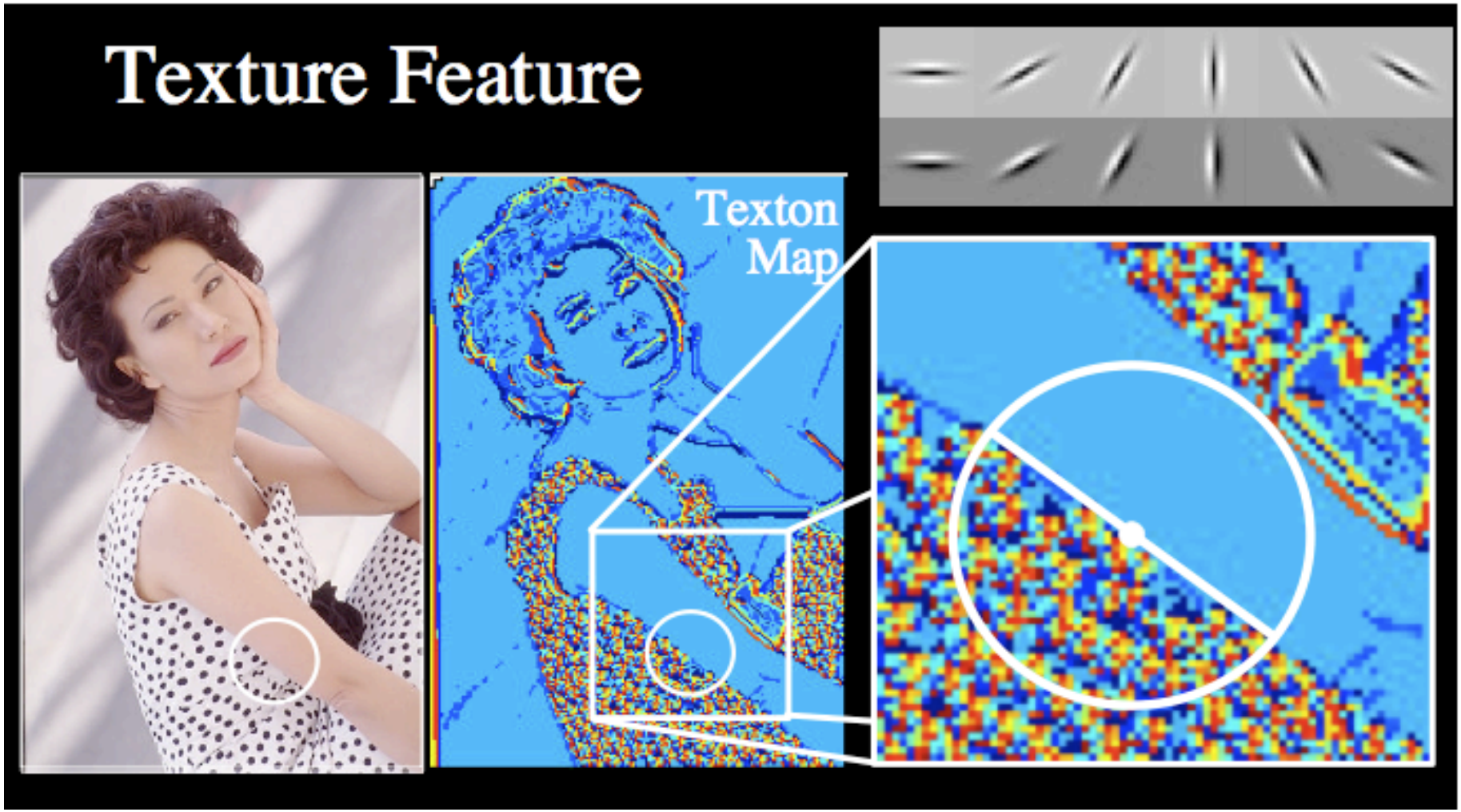
Segmentation and clustering

- Texture based clustering
- Group pixels based on texture similarity
- Texture representation
- Cluster output of the filter banks – clusters are so called textons
- 

- In the lecture on texture – we can classify the entire image based on histogram of textons
- Recognition was done by finding a images with the closest histogram
- Histogram distance was measure using chi-squared distance between histograms

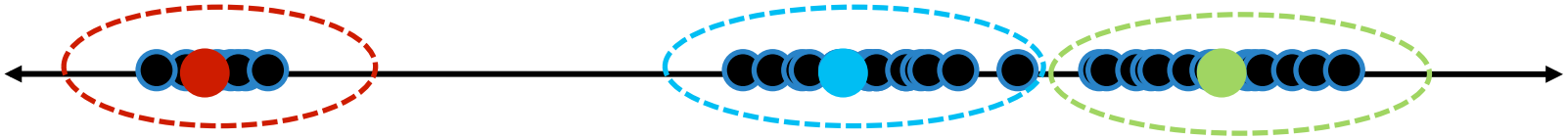
$$\chi^2(h_i, h_j) = \sum_{1}^{K} \frac{(h_i(k) - h_j(k))^2}{h_i(k) + h_j(k)}$$

Texture Segmentation

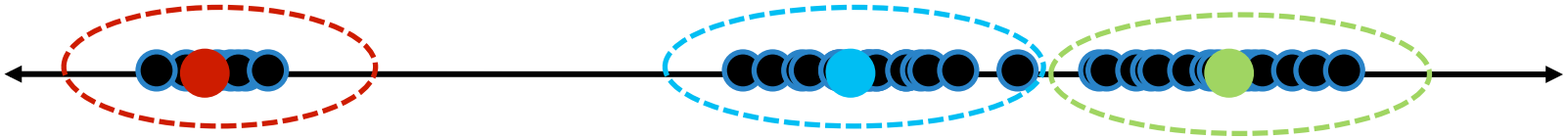


Clustering

- With this objective, it is a "chicken and egg" problem:
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



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

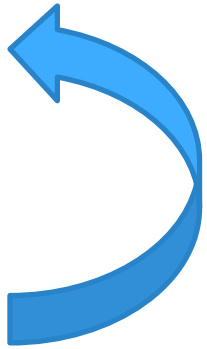


Slides: Kristen Grauman

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, ..., 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



Properties

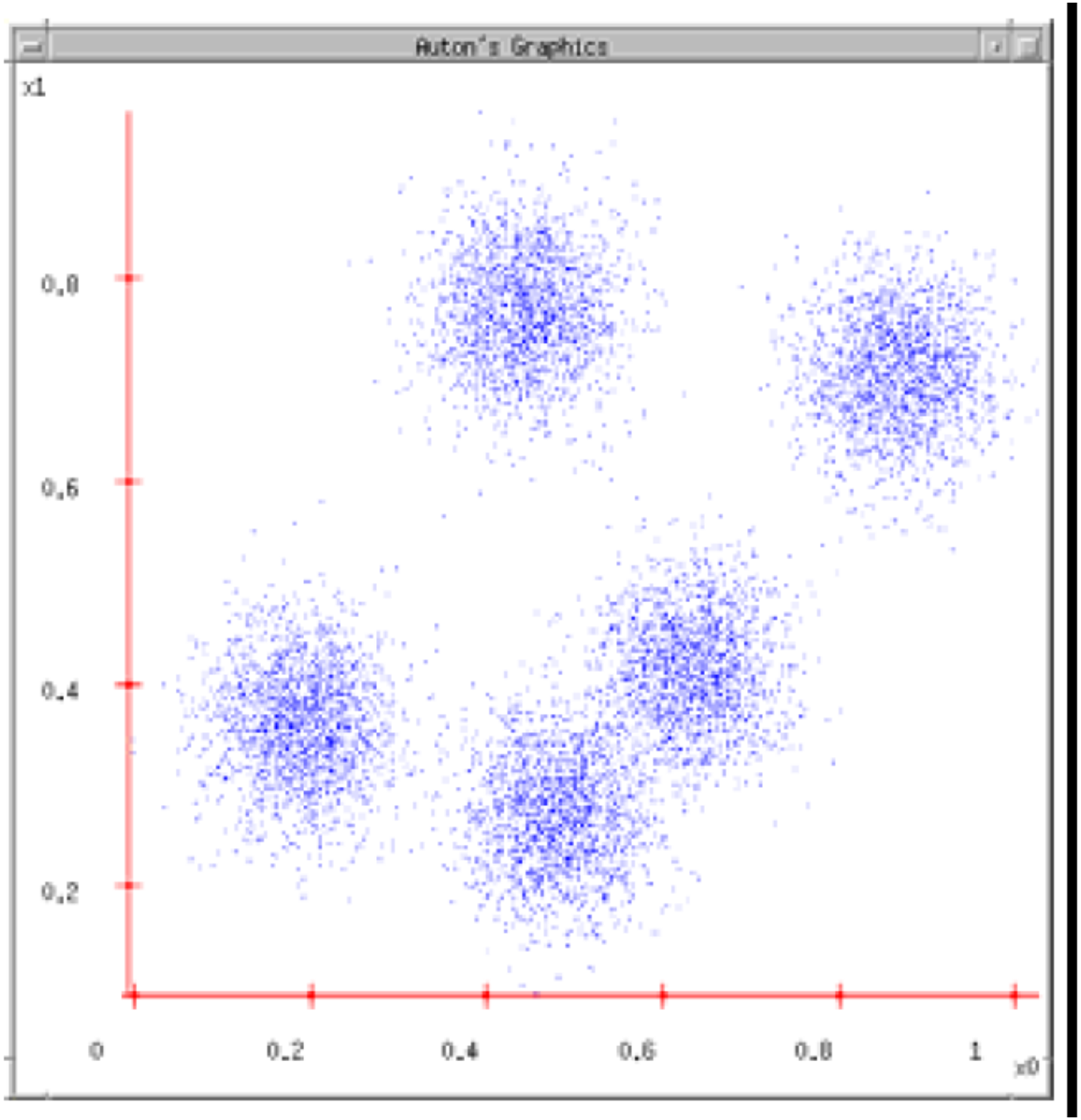
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

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

Source: Steve Seitz

K-means

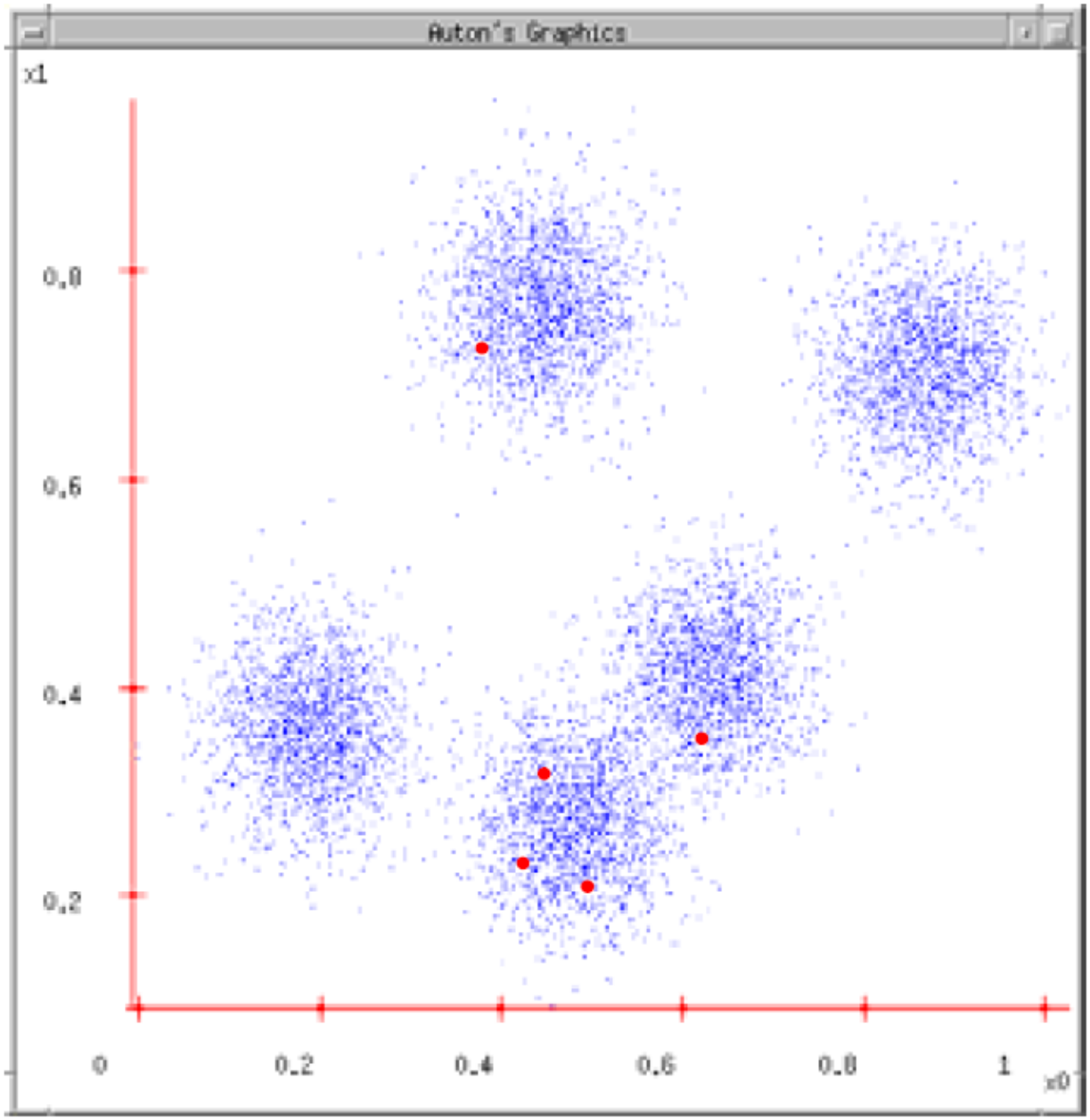
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Andrew Moore

K-means

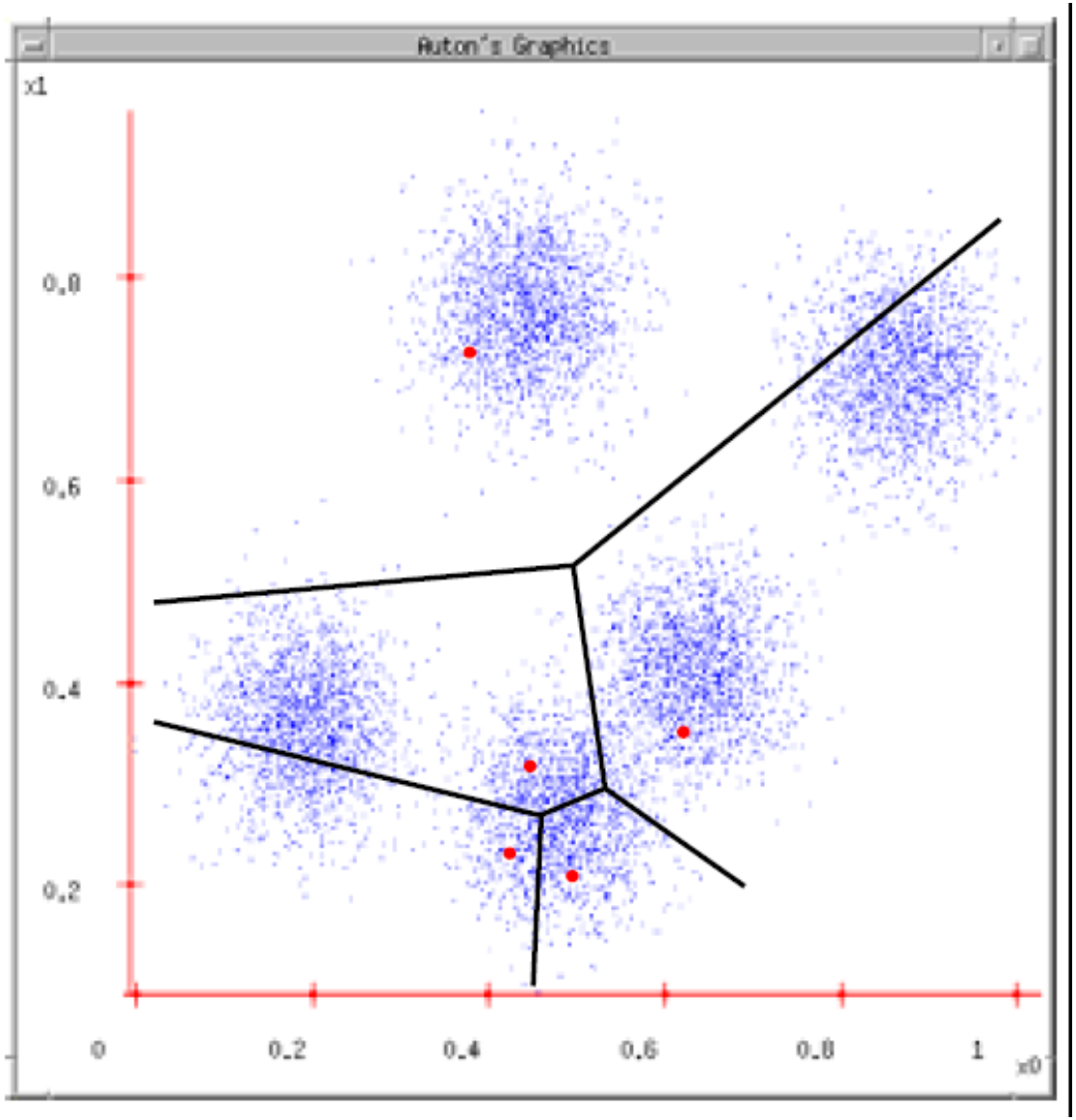
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



Andrew Moore

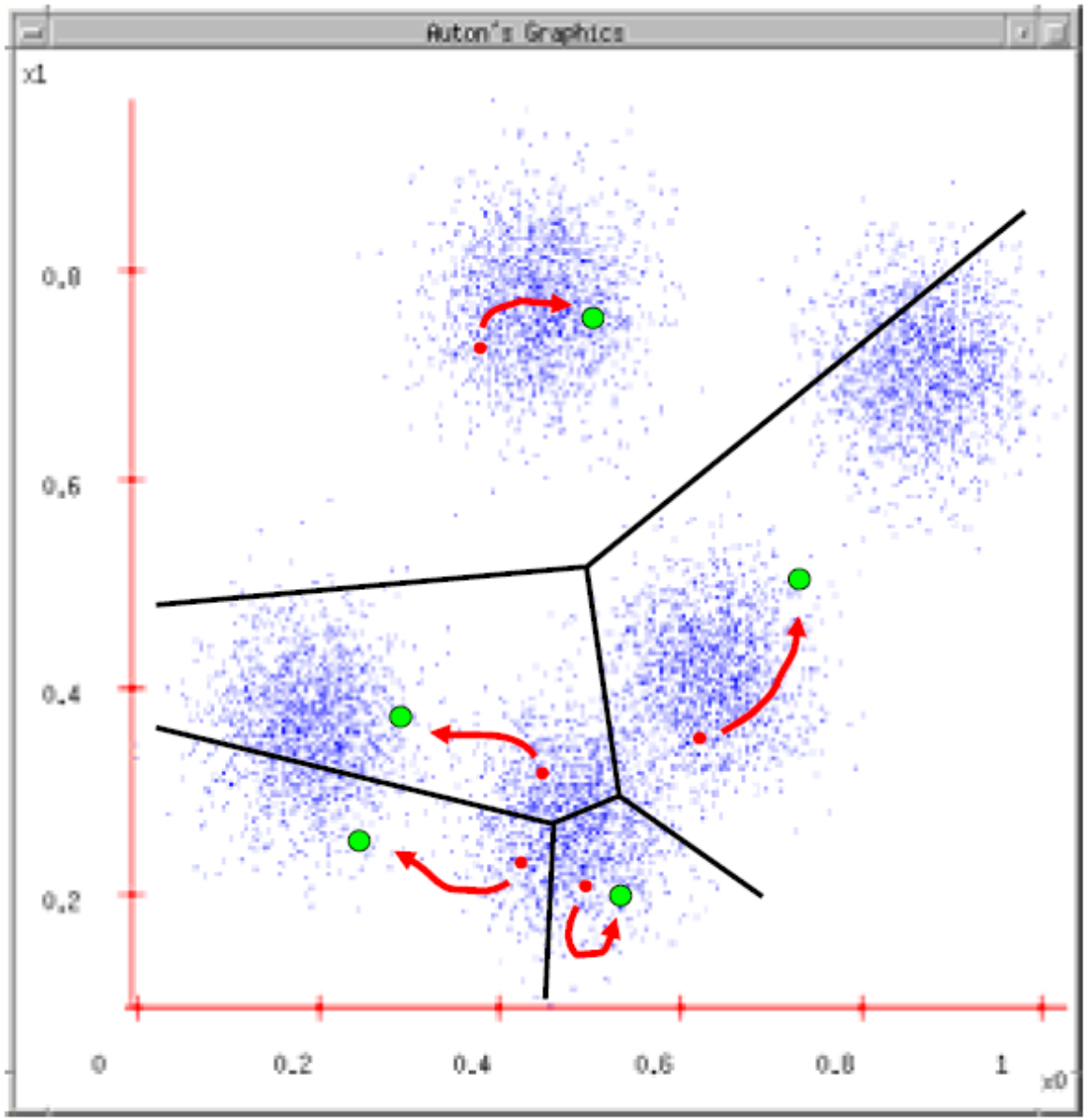
K-means

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)



K-means

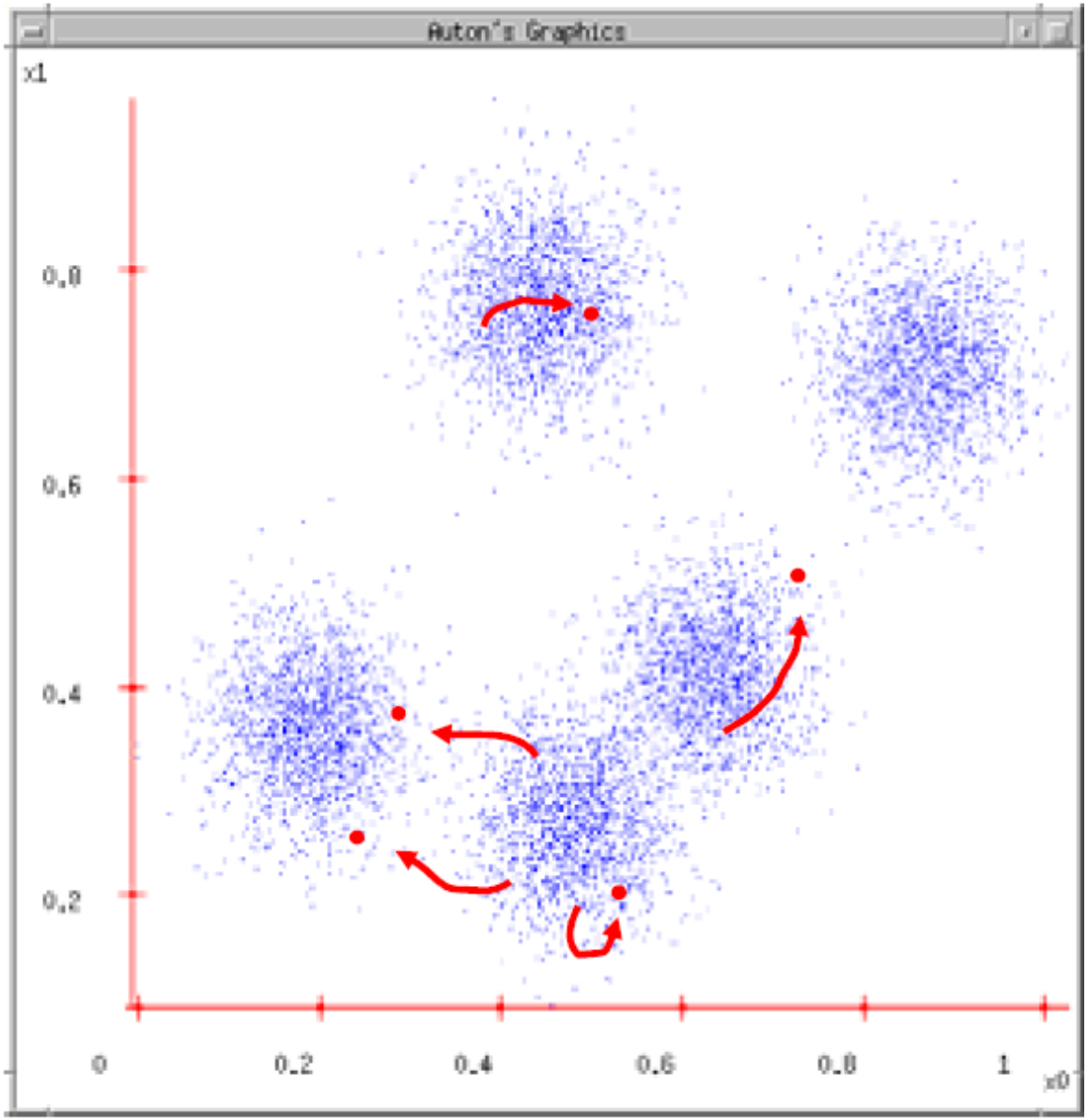
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



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

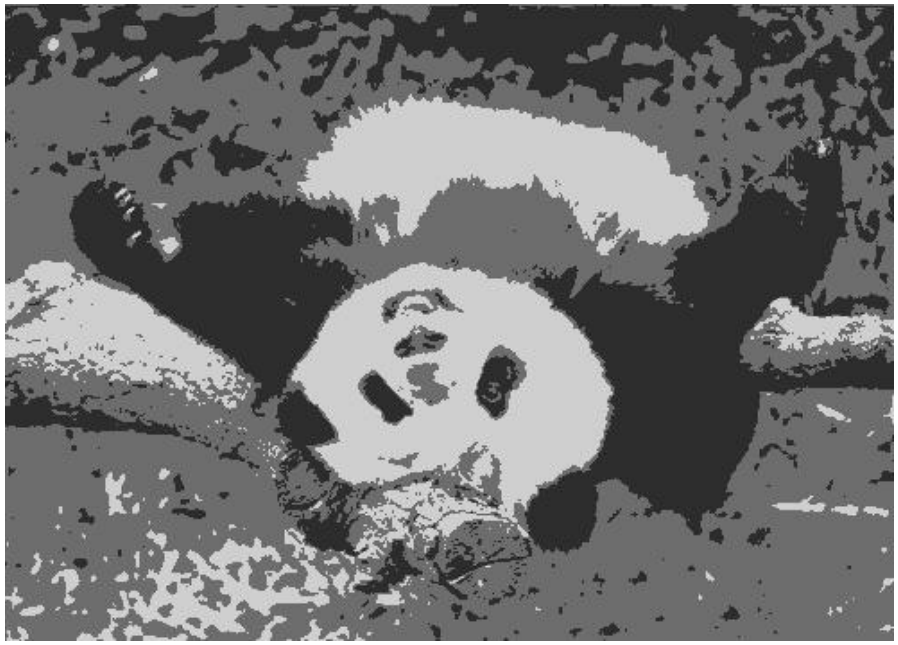
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

K=2

K=3

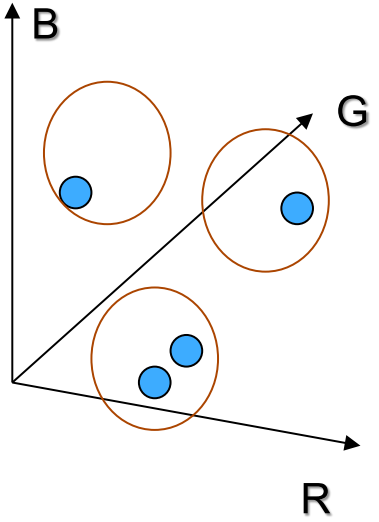


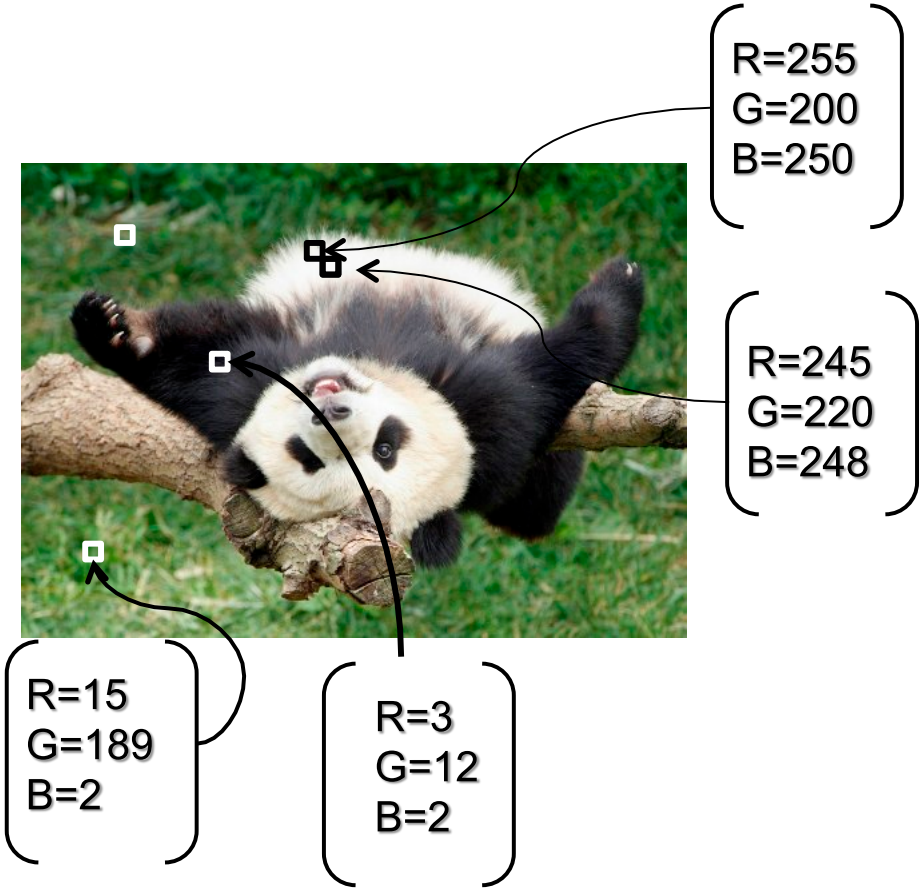
quantization of the feature space;
segmentation label map

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

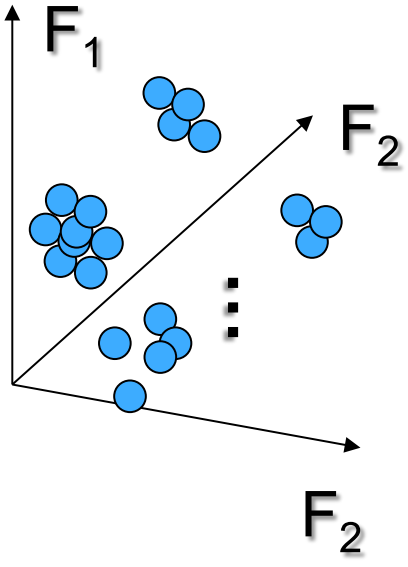
- Color, brightness, position alone are not enough to distinguish all regions…

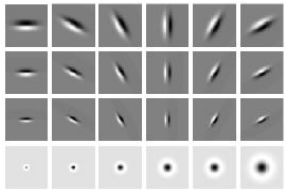


Segmentation as clustering

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

Grouping pixels based on **texture** similarity

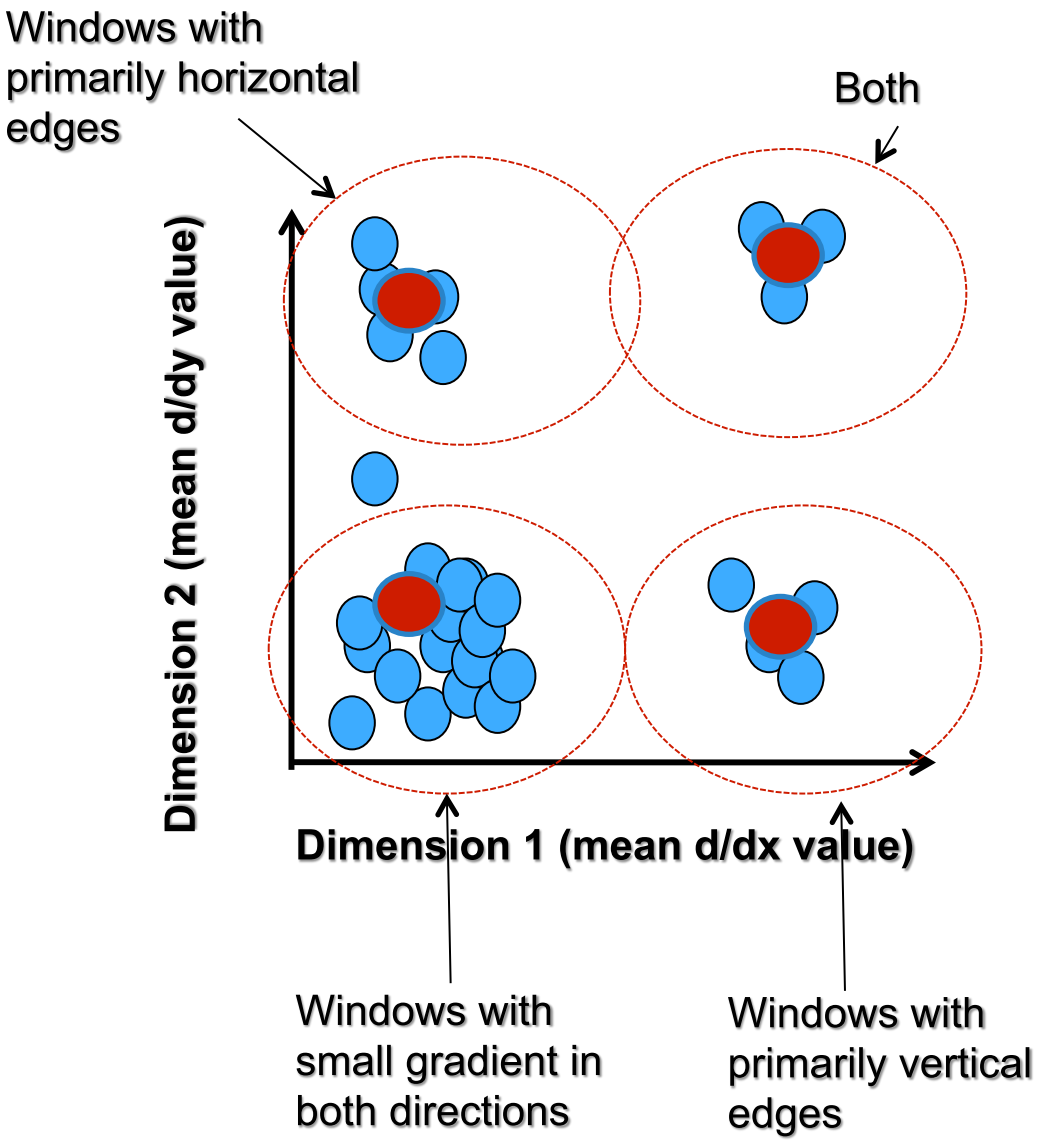


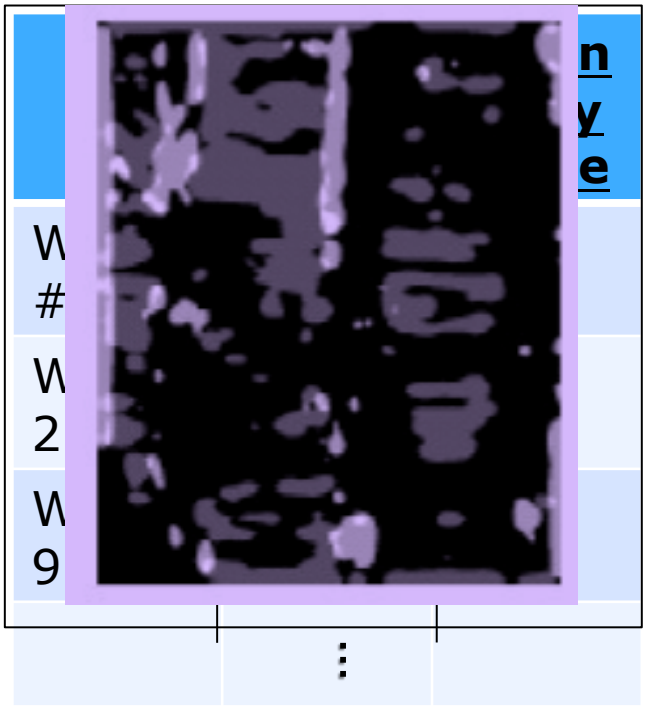


Filter bank of 24 filters

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

Recall: texture representation example



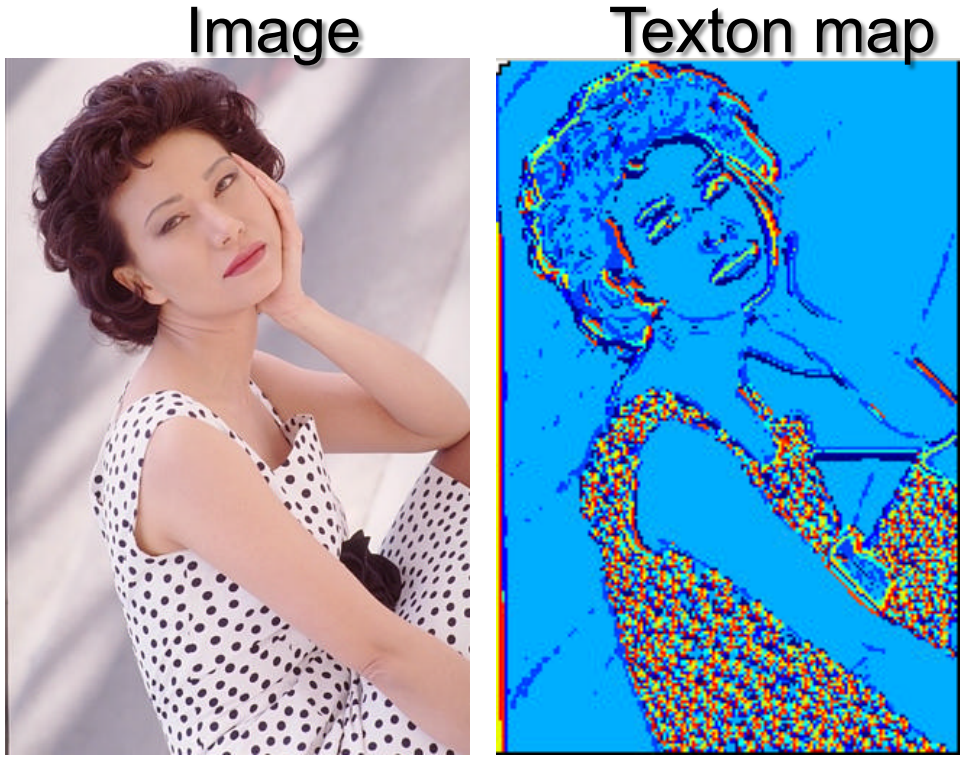


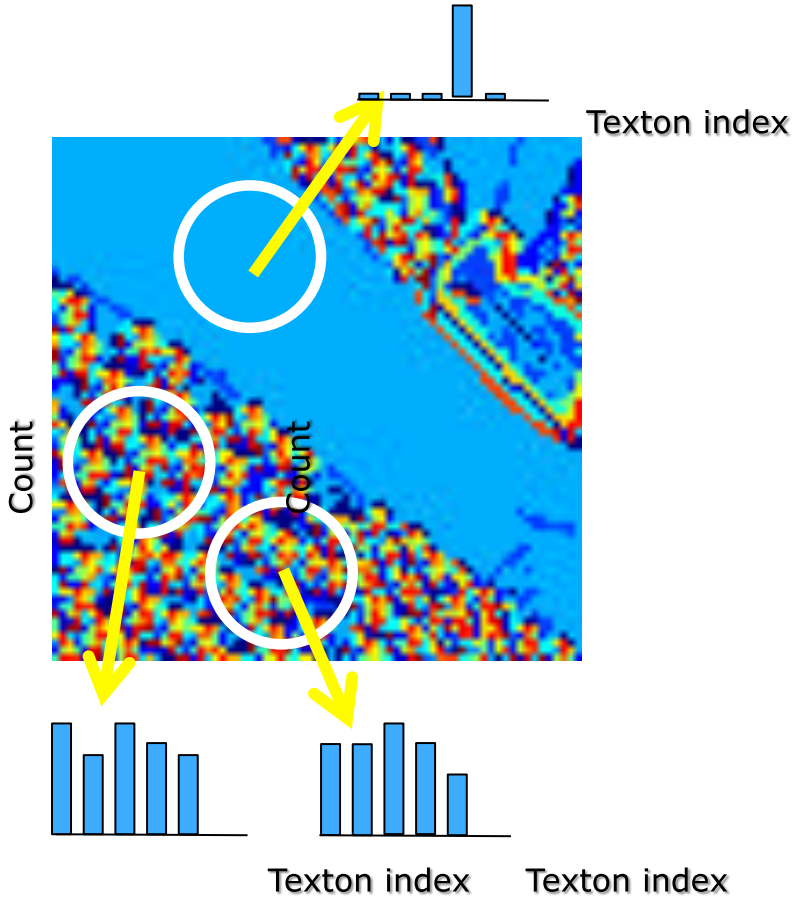
statistics to summarize patterns in small windows

Kristen Grauman

Segmentation with texture features

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





Malik, Belongie, Leung and Shi. IJCV 2001.

Adapted from Lana Lazebnik

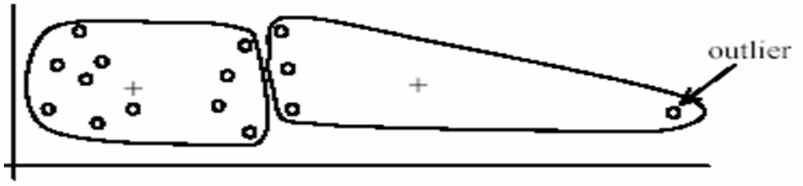
K-means: pros and cons

Pros

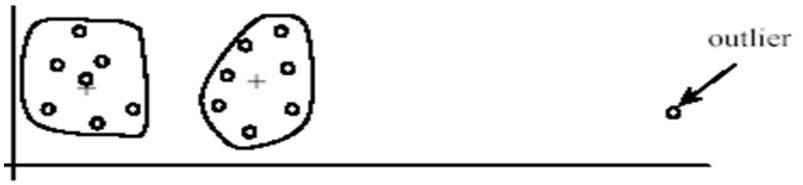
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

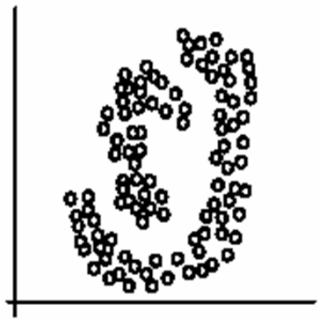
- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed



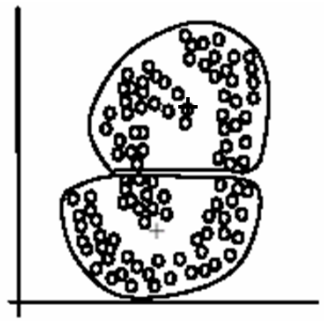
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters



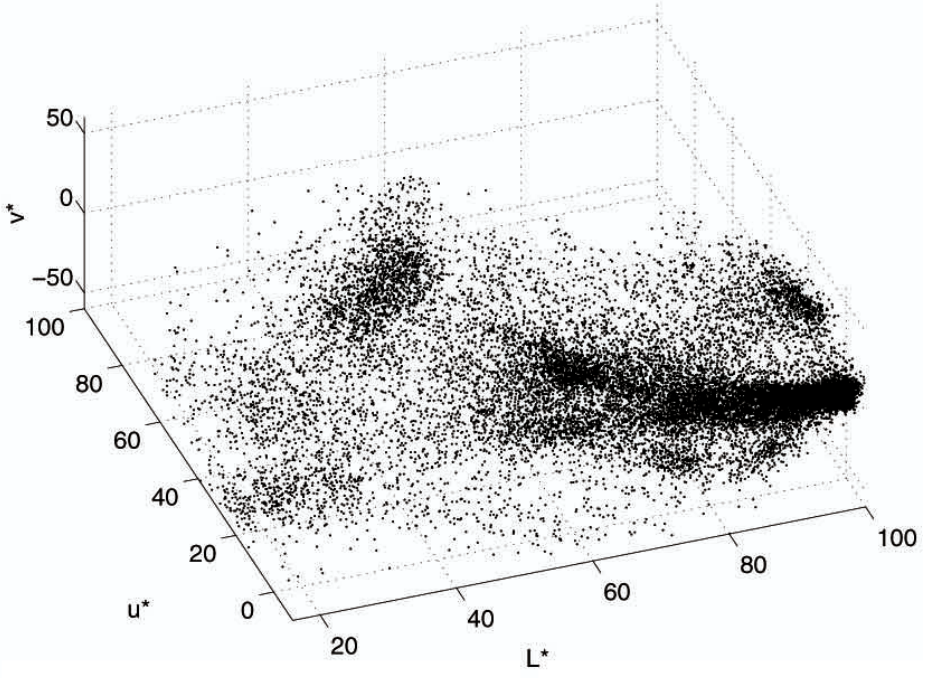
(B): *k*-means clusters

Mean shift algorithm

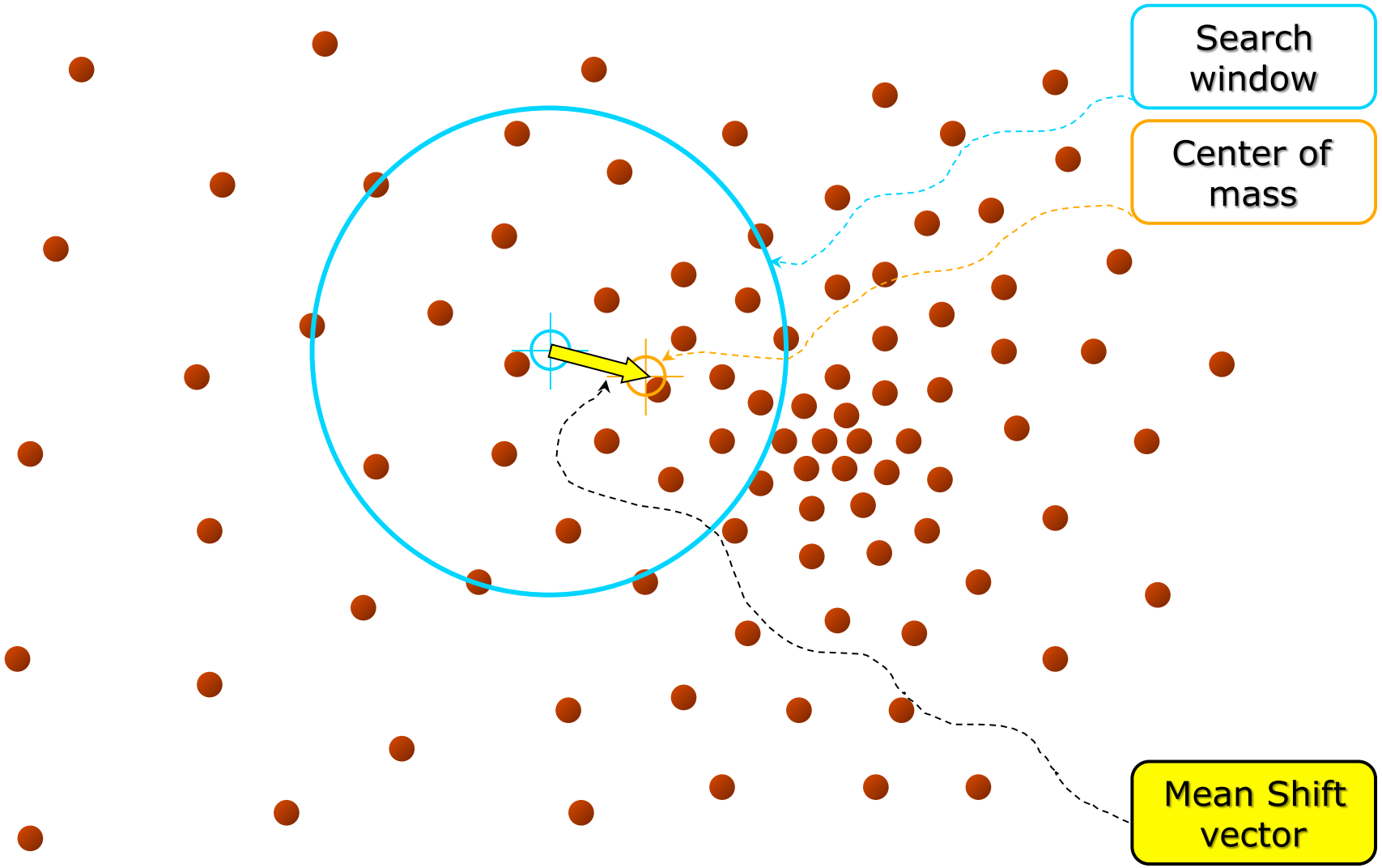
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

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

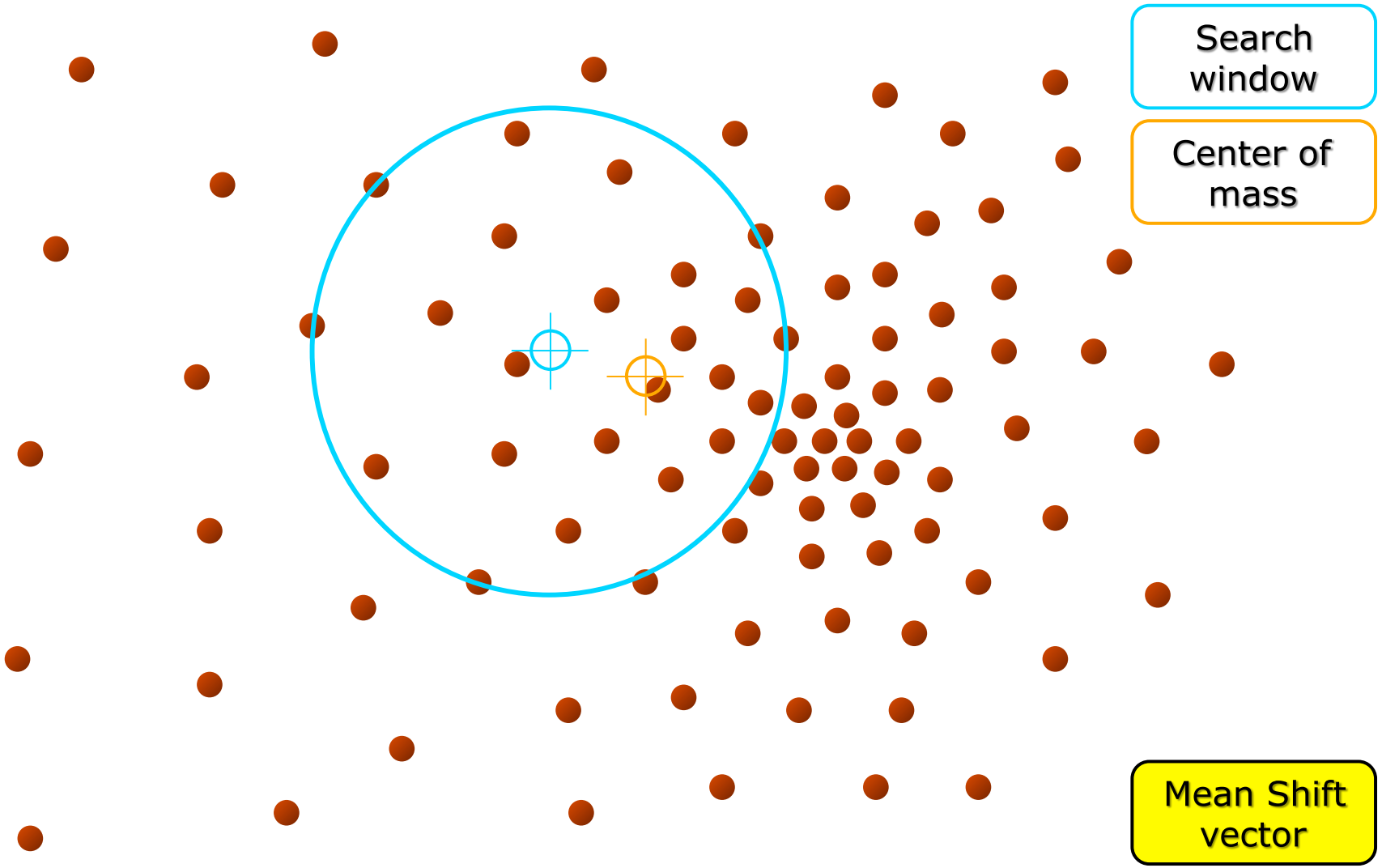


Mean shift



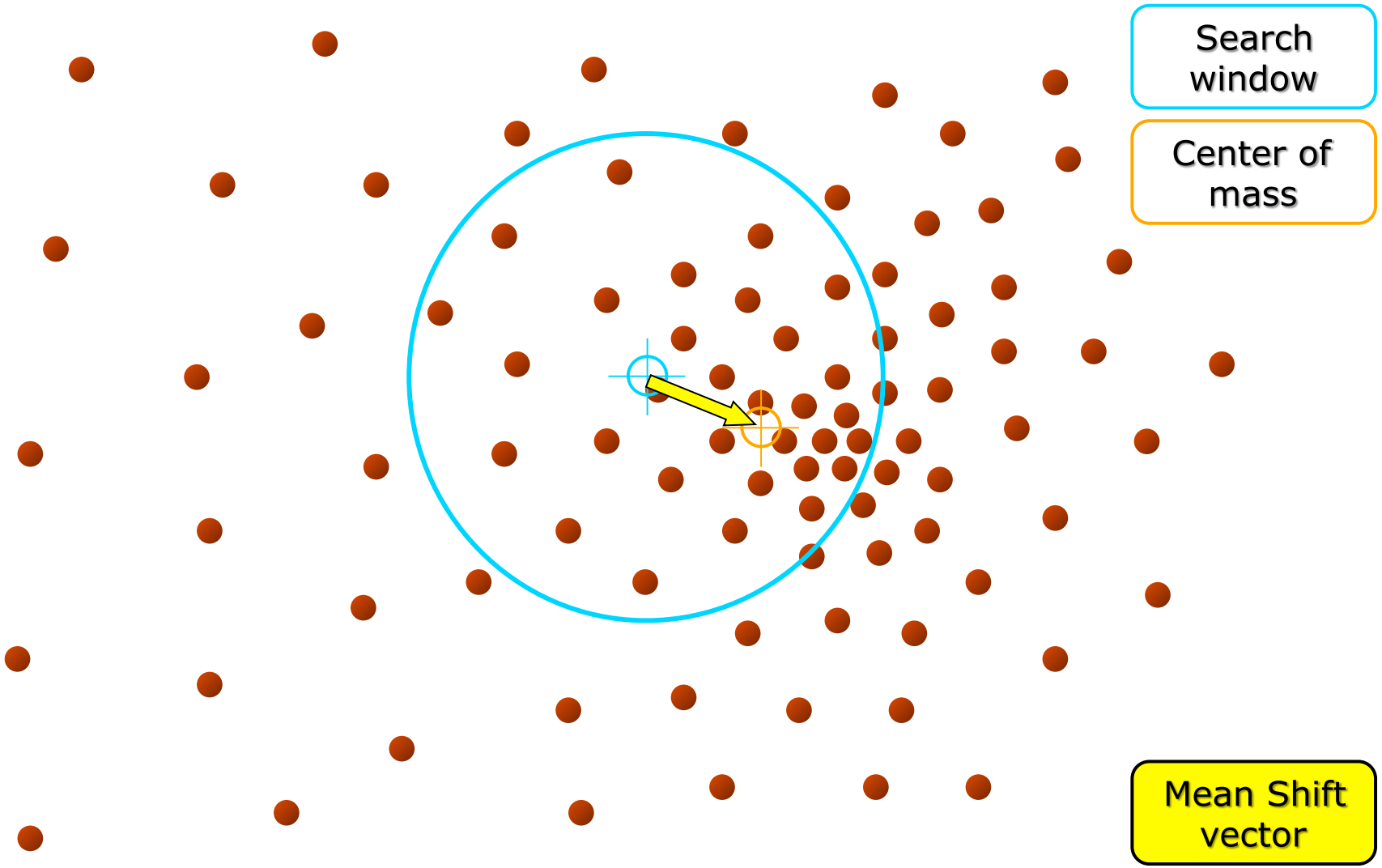
Slide by Y. Ukrainitz & B. Sarel

Mean shift



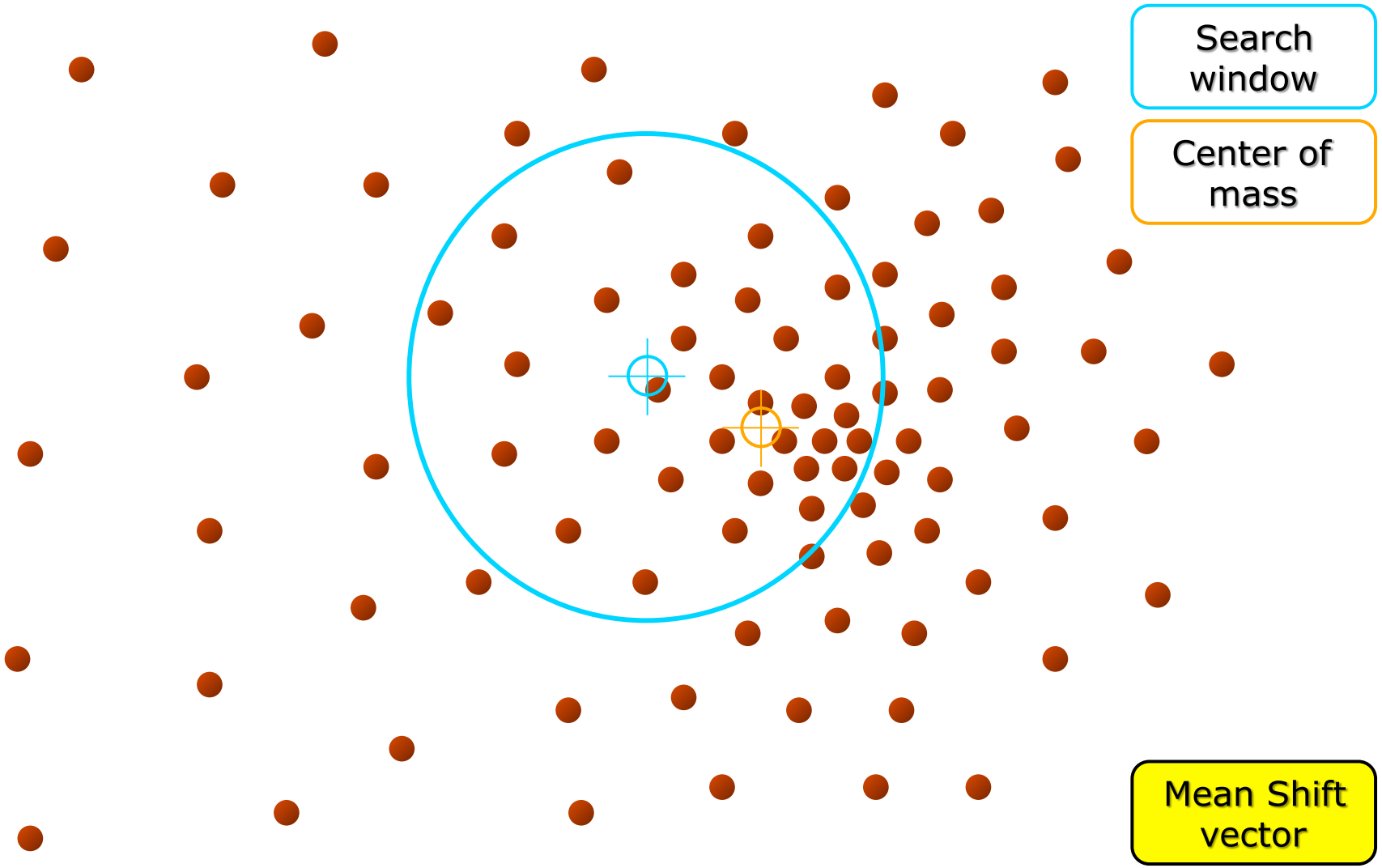
Slide by Y. Ukrainitz & B. Sarel

Mean shift



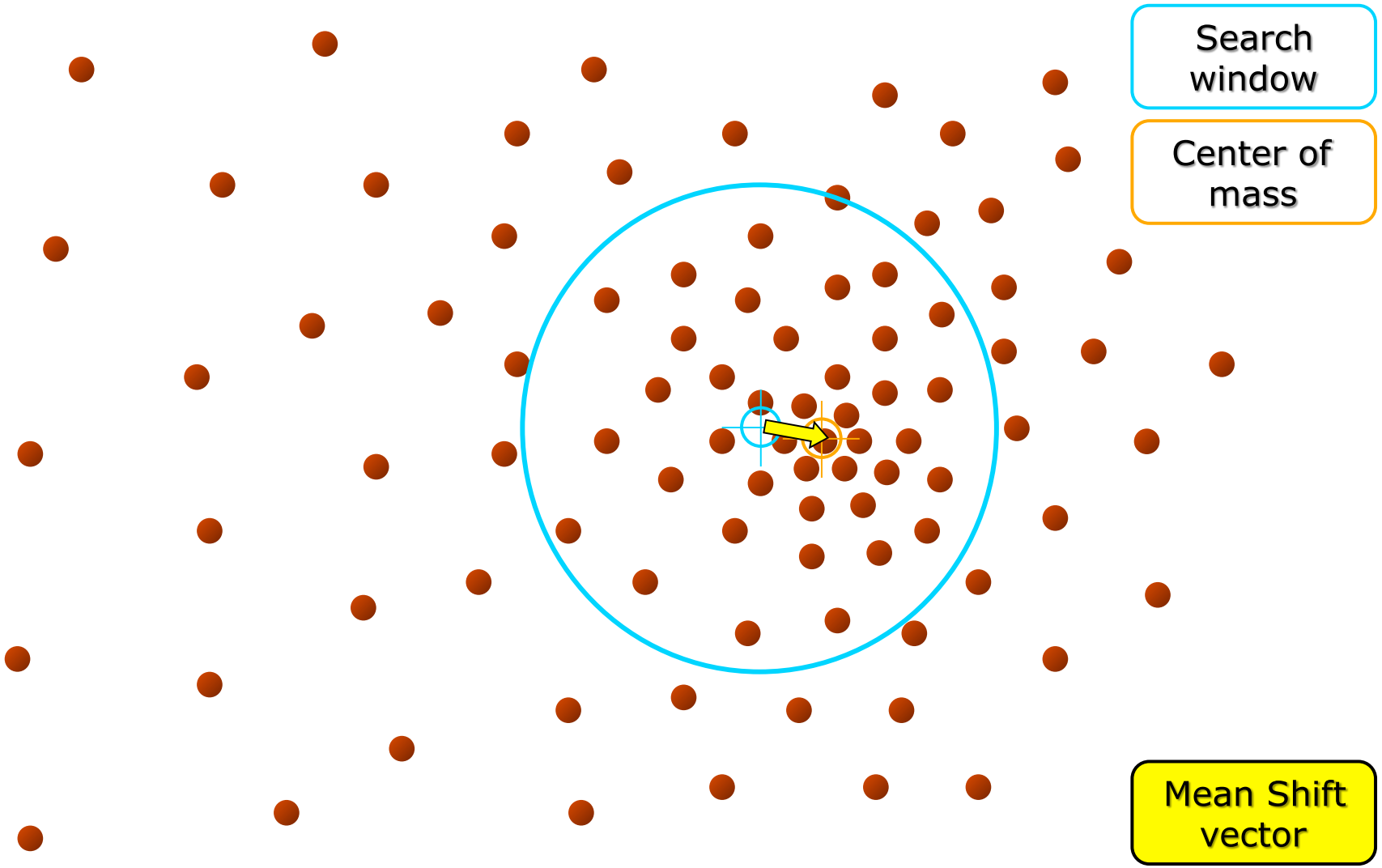
Slide by Y. Ukrainitz & B. Sarel

Mean shift



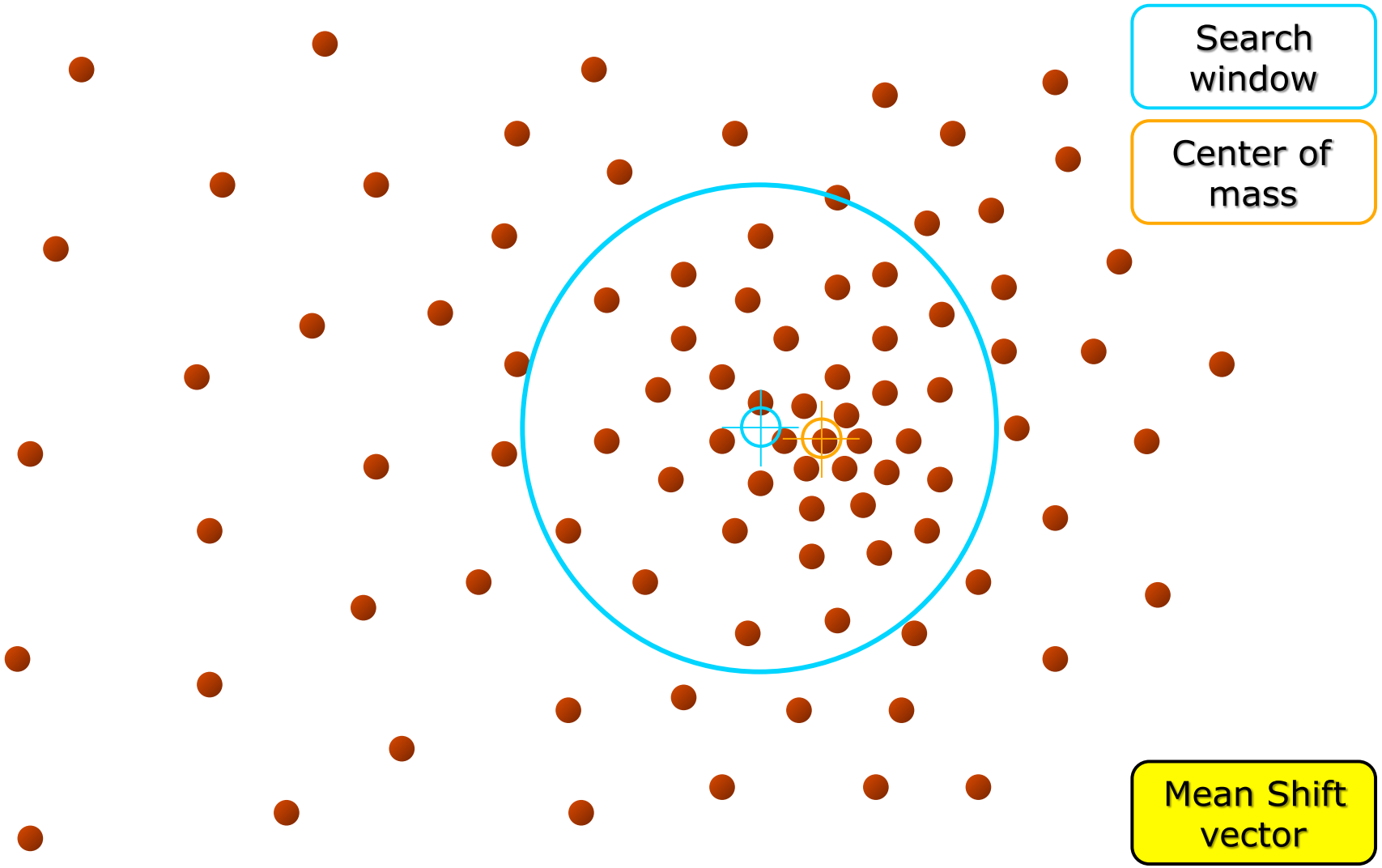
Slide by Y. Ukrainitz & B. Sarel

Mean shift



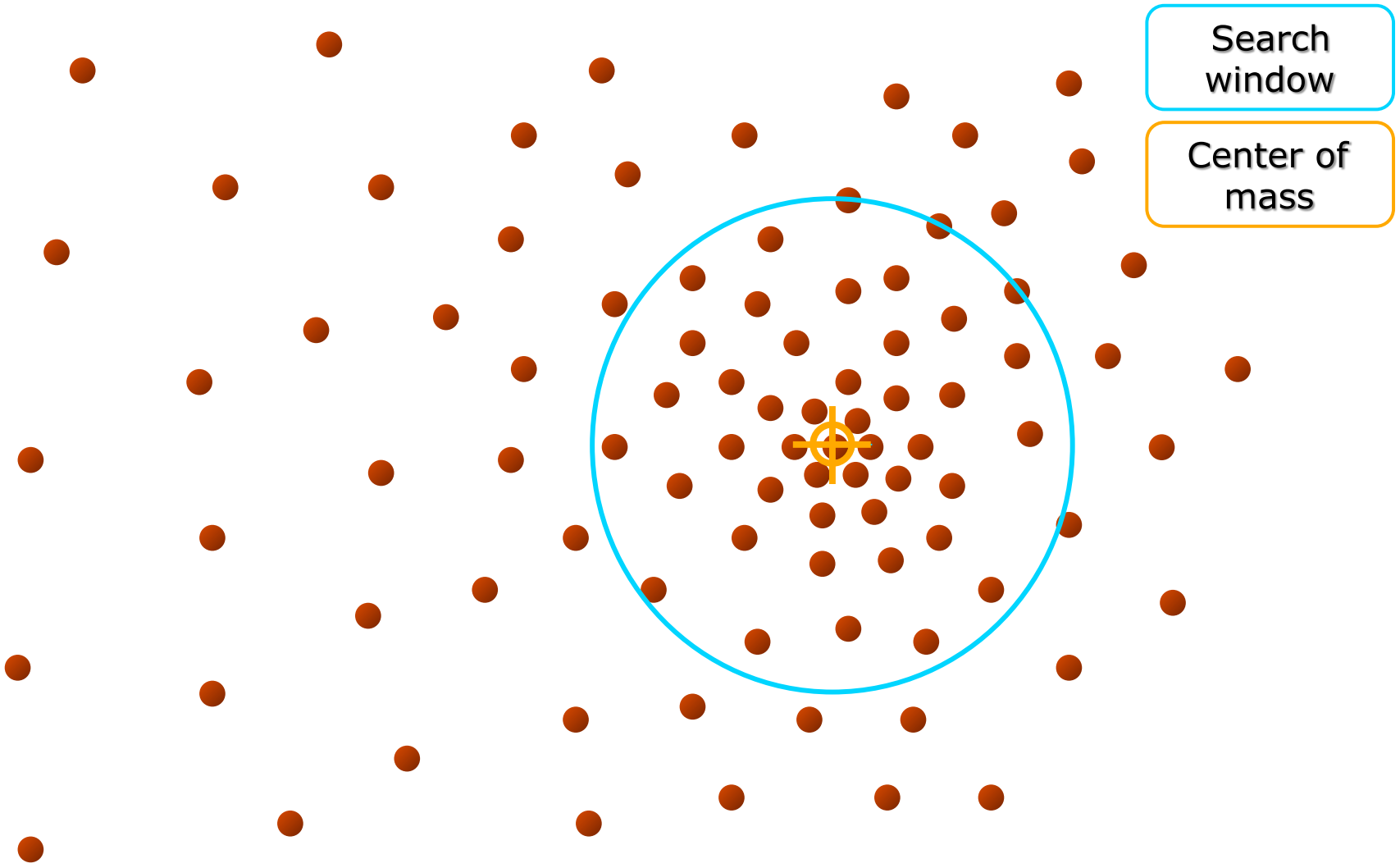
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

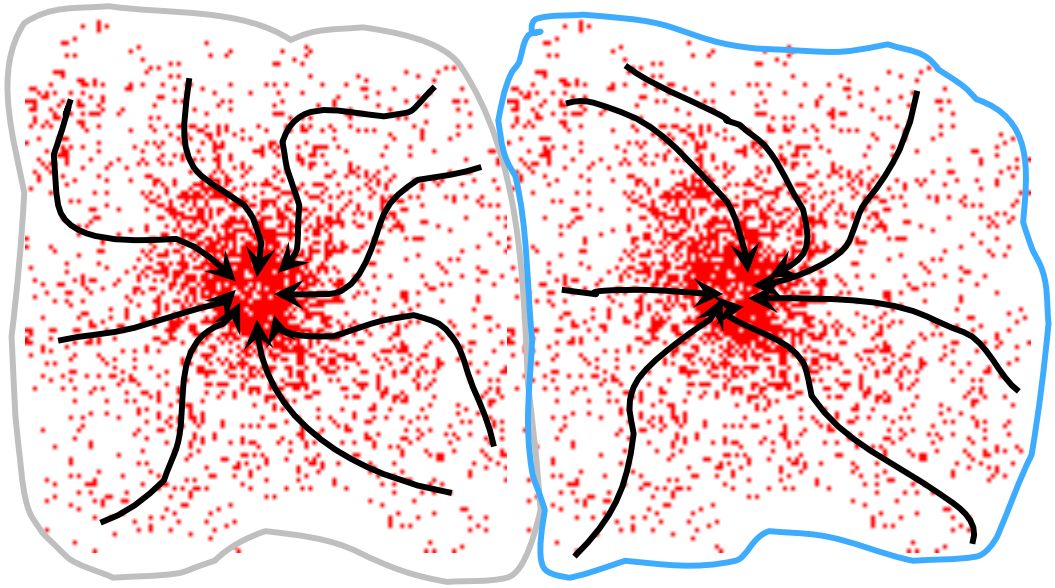
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

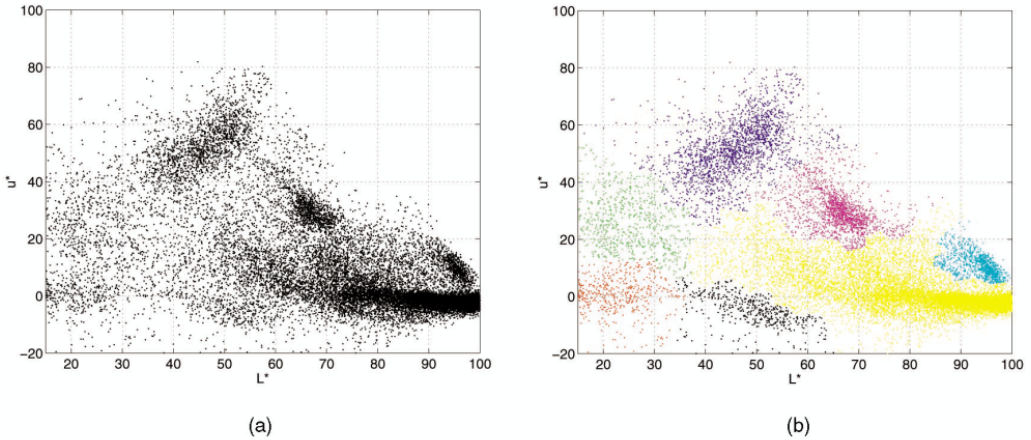
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

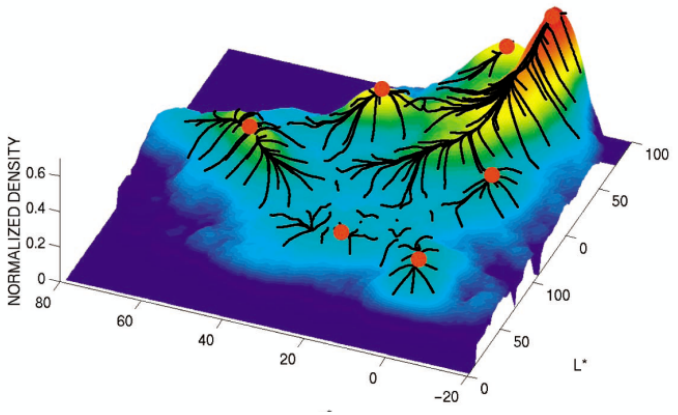


Slide by Y. Ukrainitz & B. Sarel

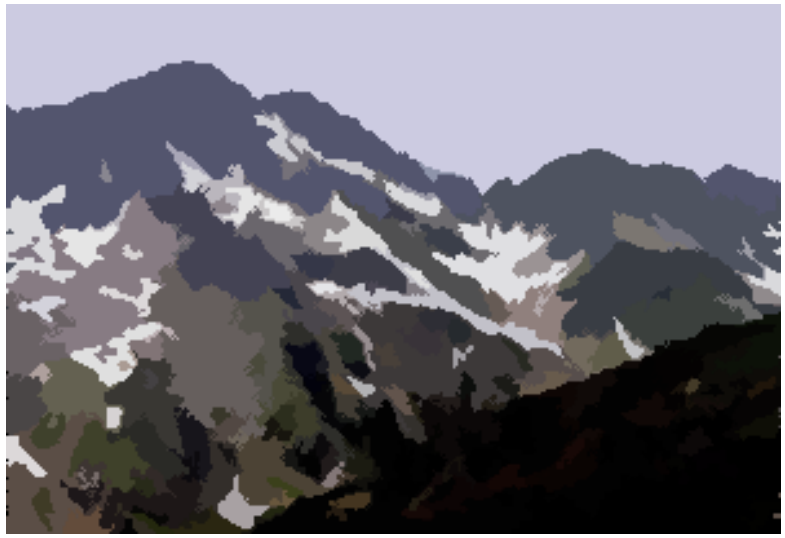
Mean shift clustering/segmentation

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





Mean shift segmentation results



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

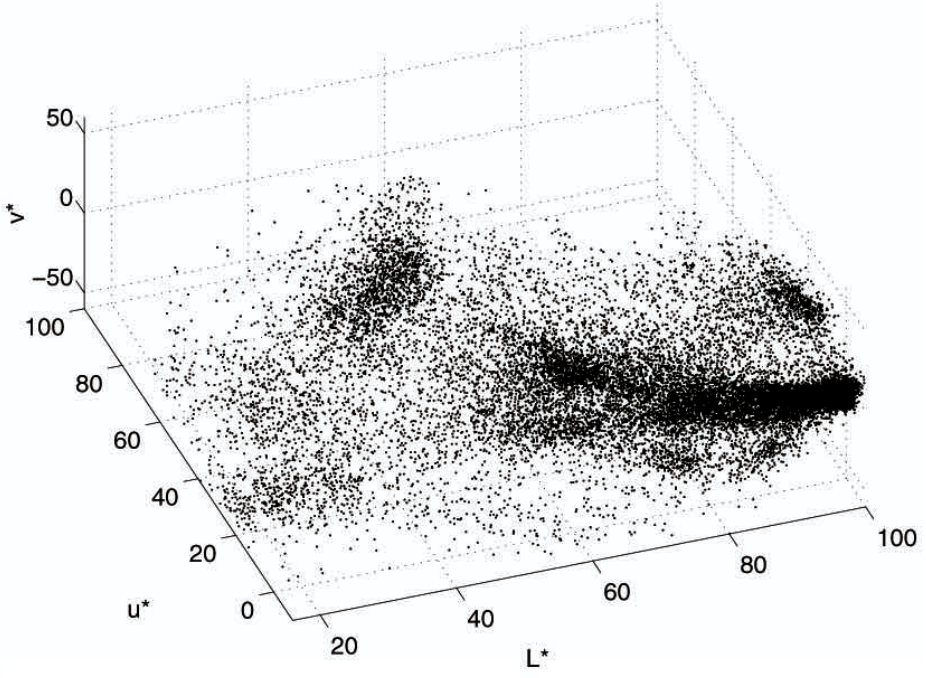
Mean shift segmentation results

Mean shift algorithm

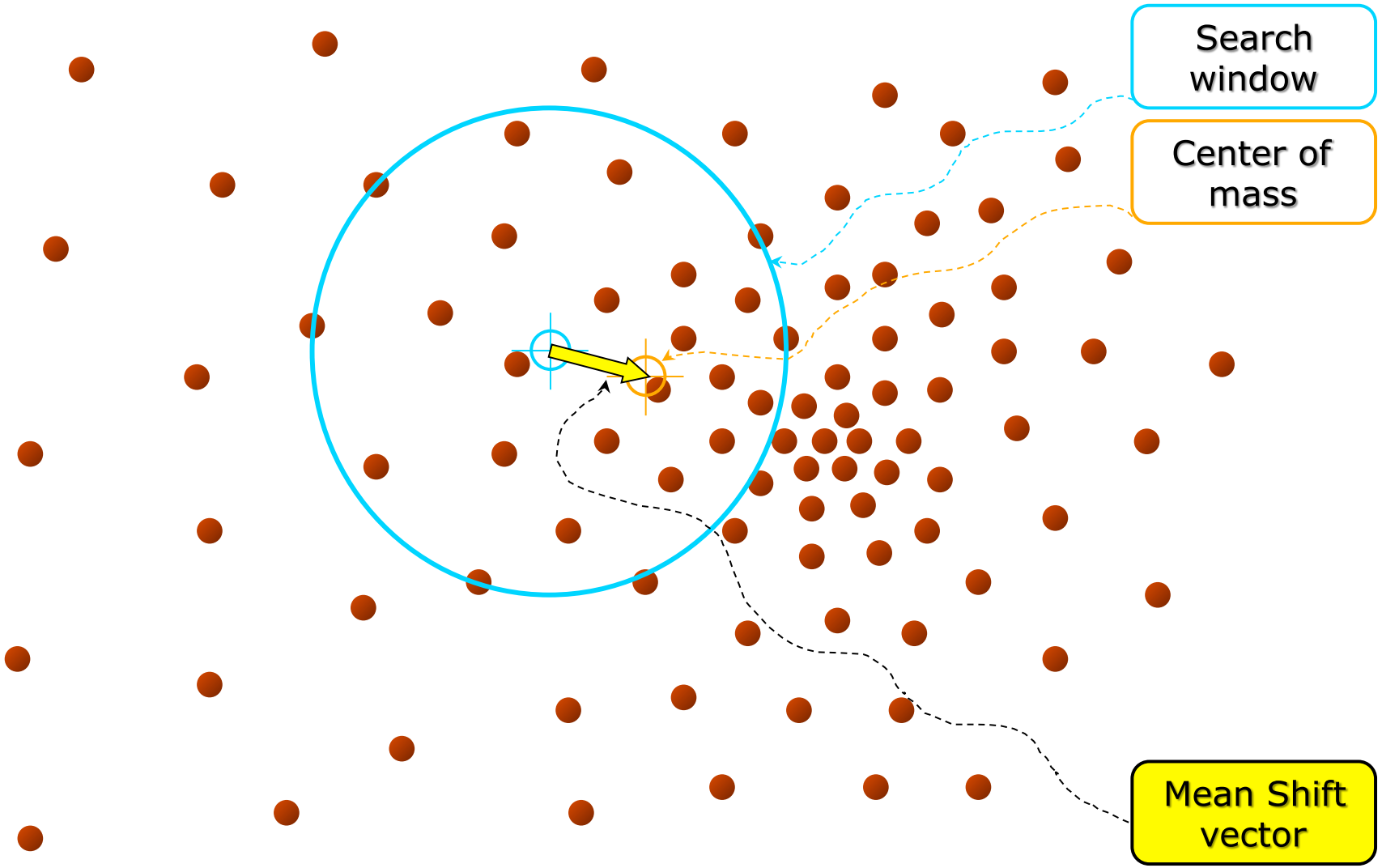
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

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

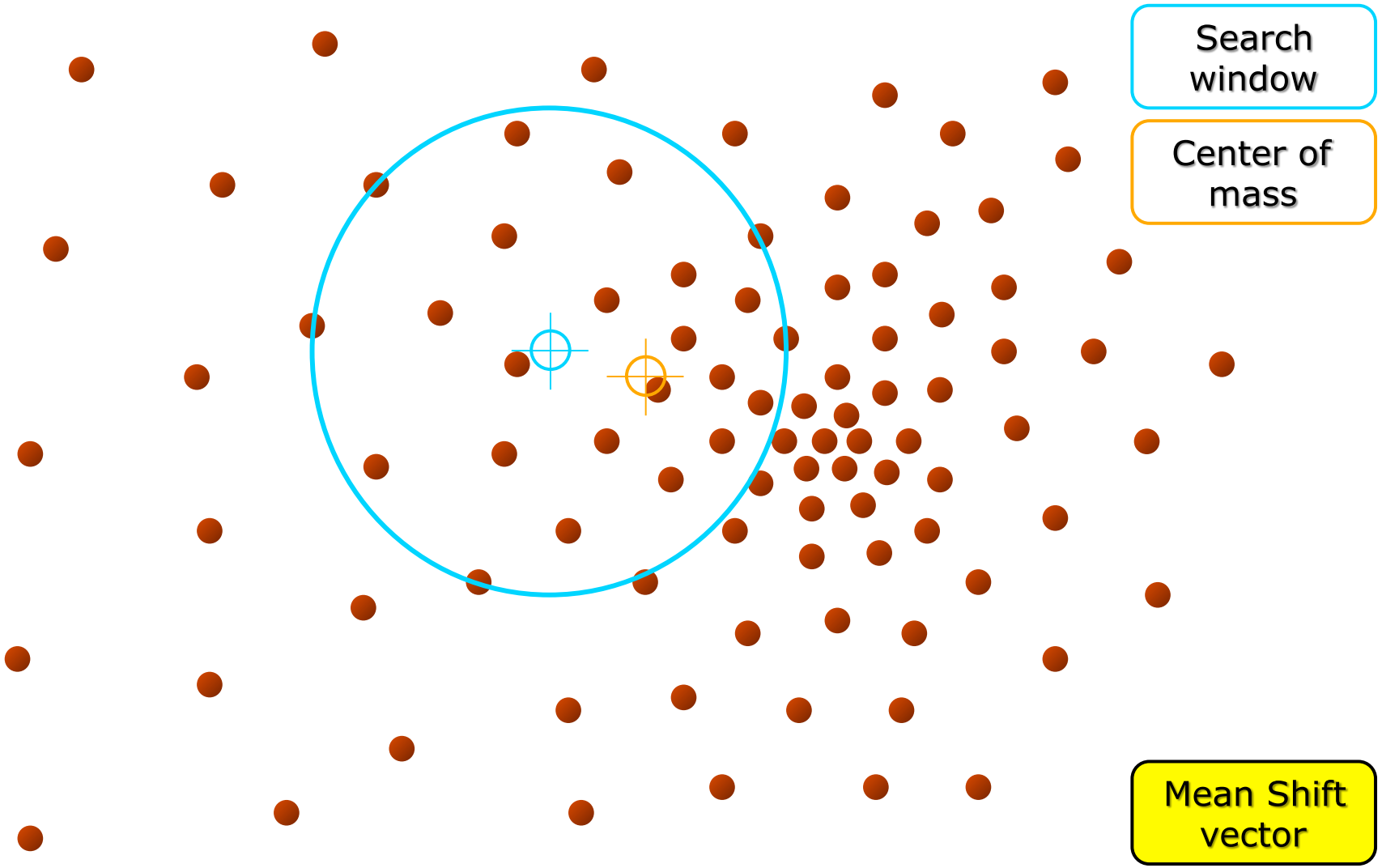


Mean shift



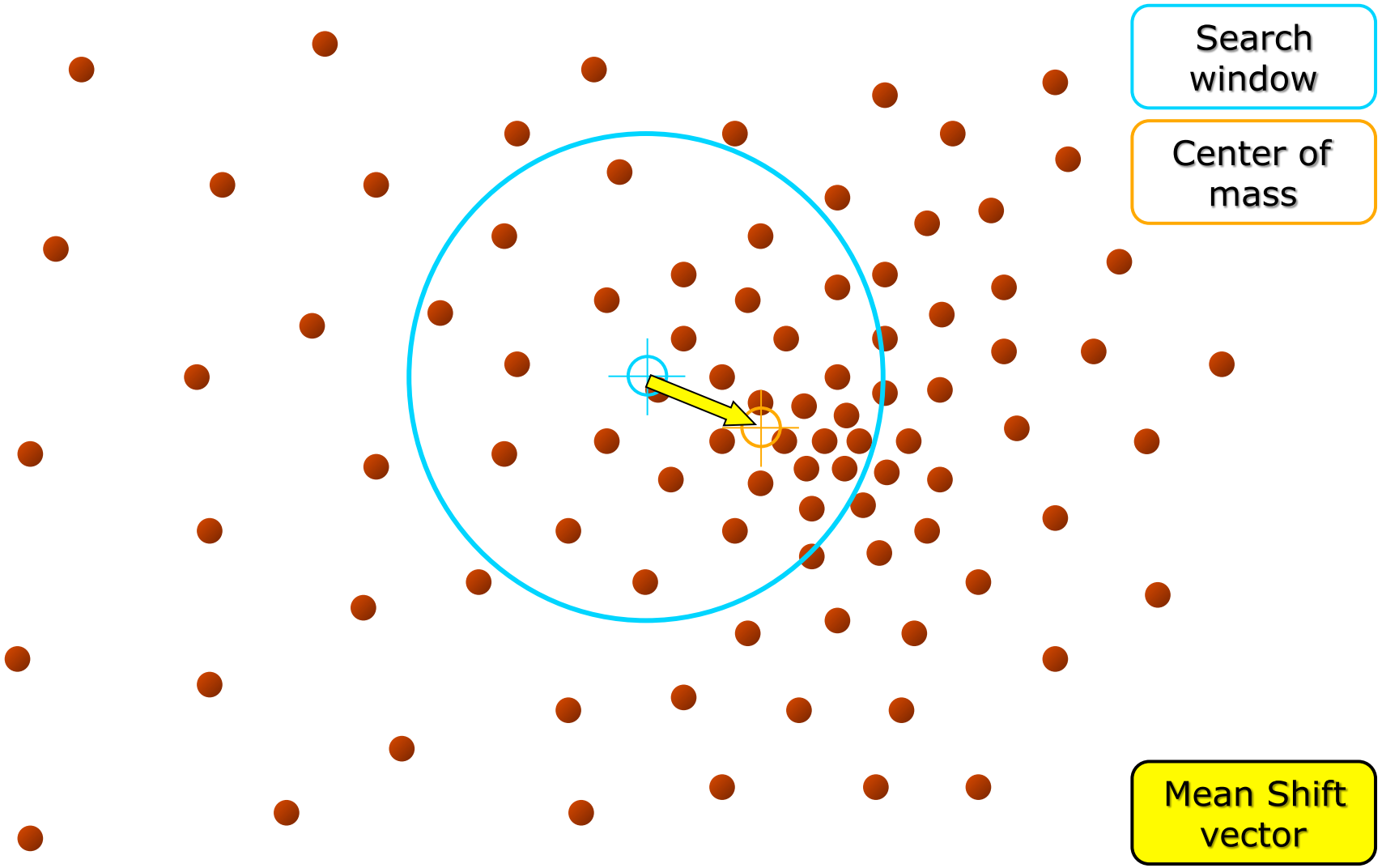
Slide by Y. Ukrainitz & B. Sarel

Mean shift



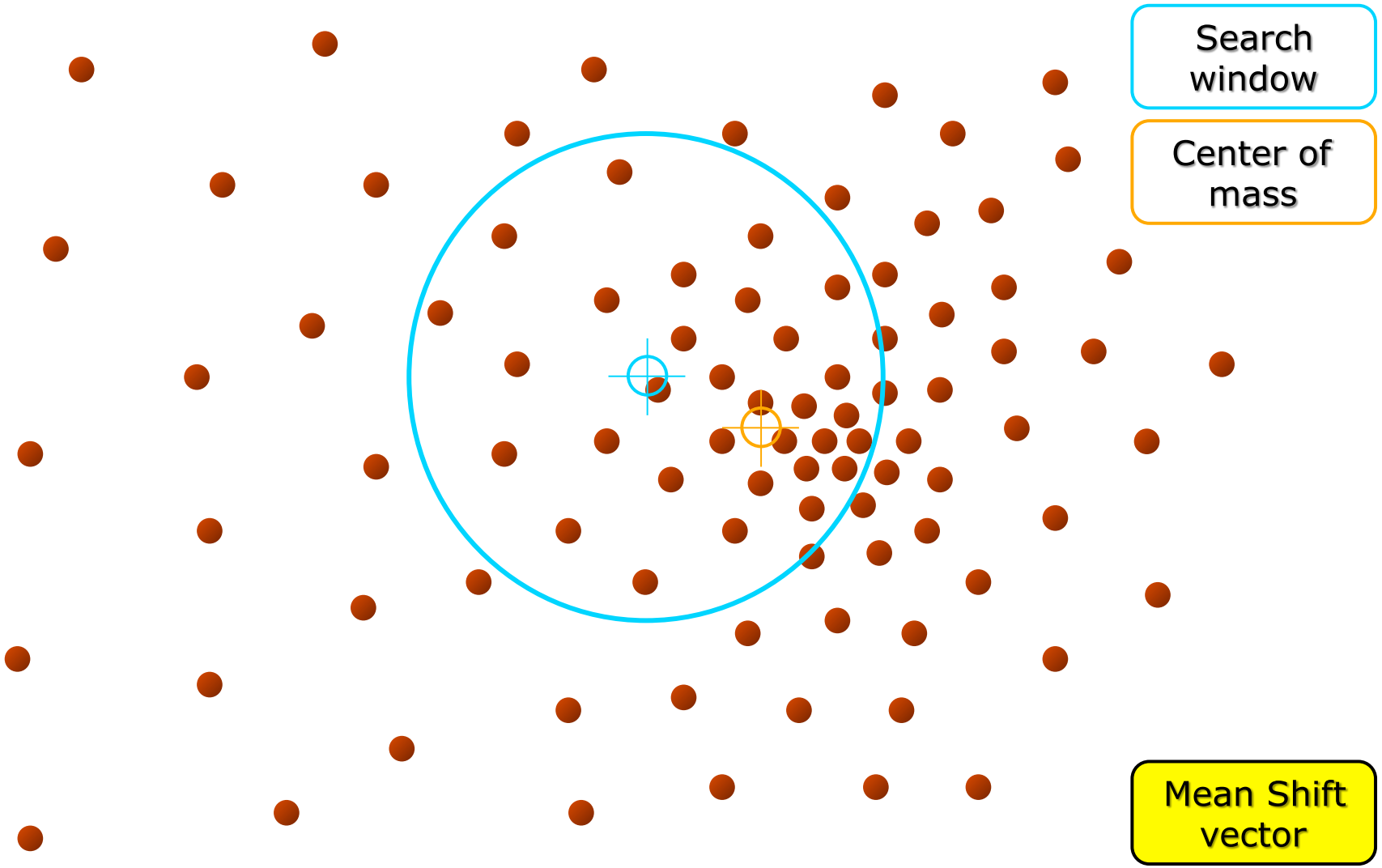
Slide by Y. Ukrainitz & B. Sarel

Mean shift



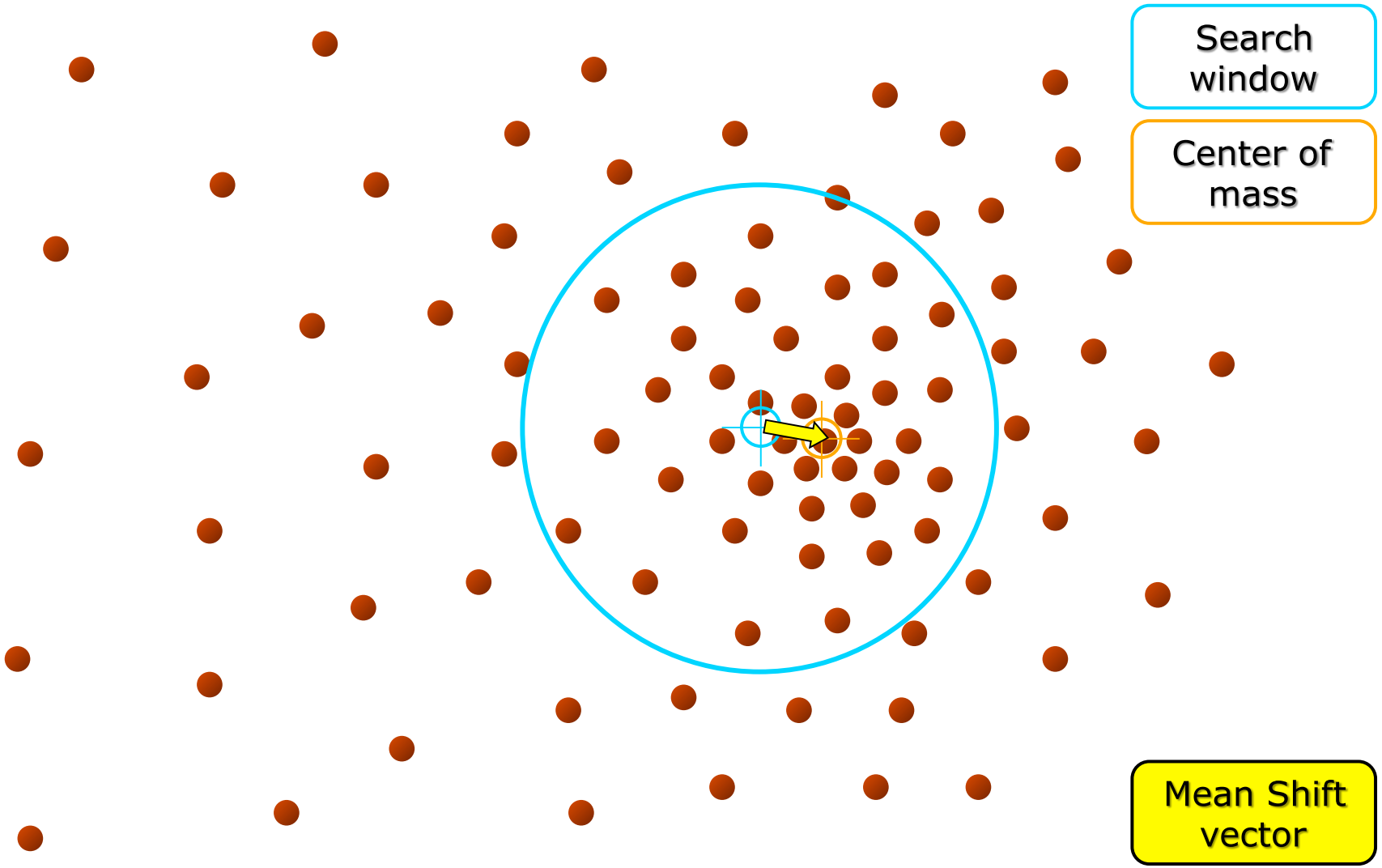
Slide by Y. Ukrainitz & B. Sarel

Mean shift



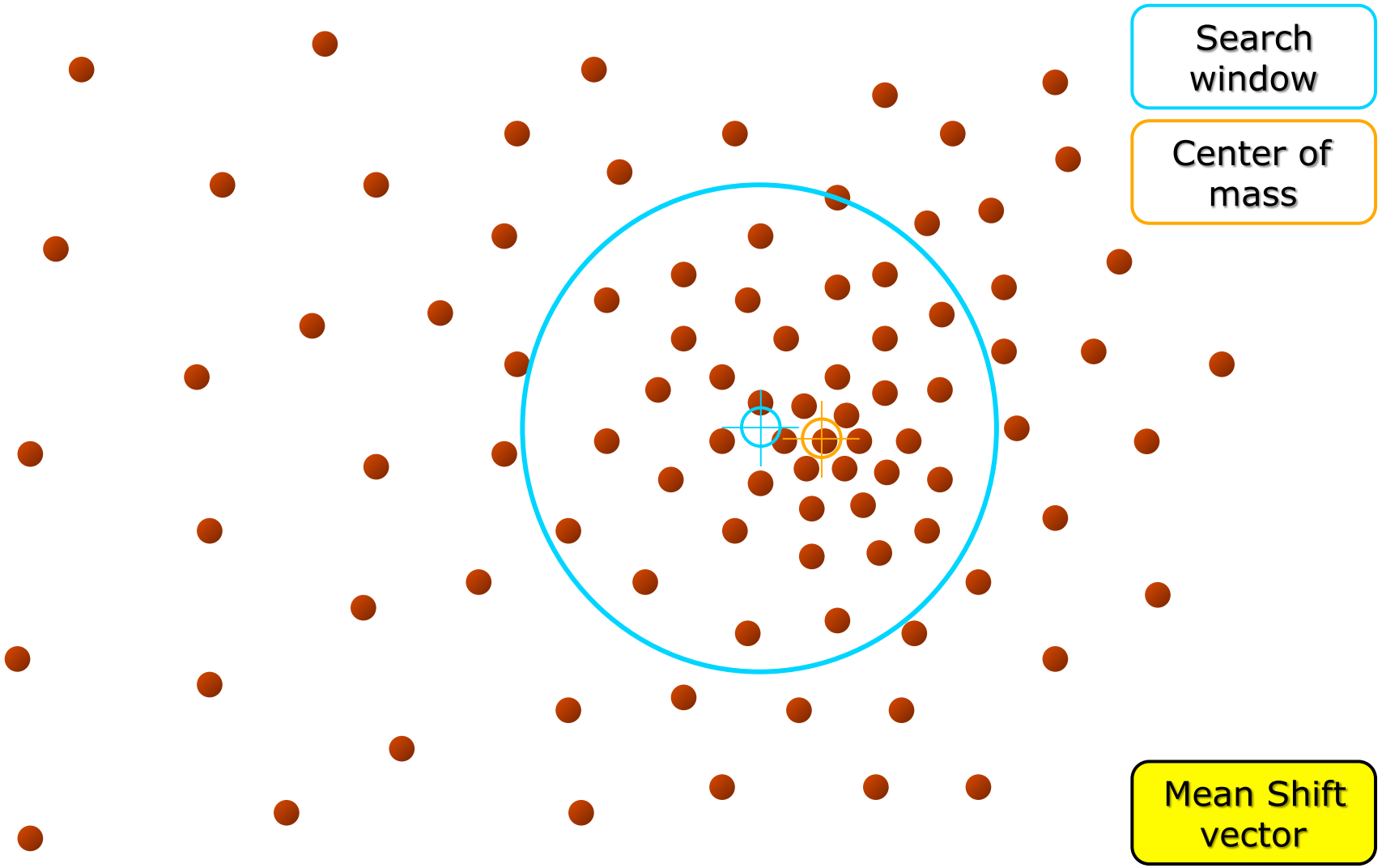
Slide by Y. Ukrainitz & B. Sarel

Mean shift



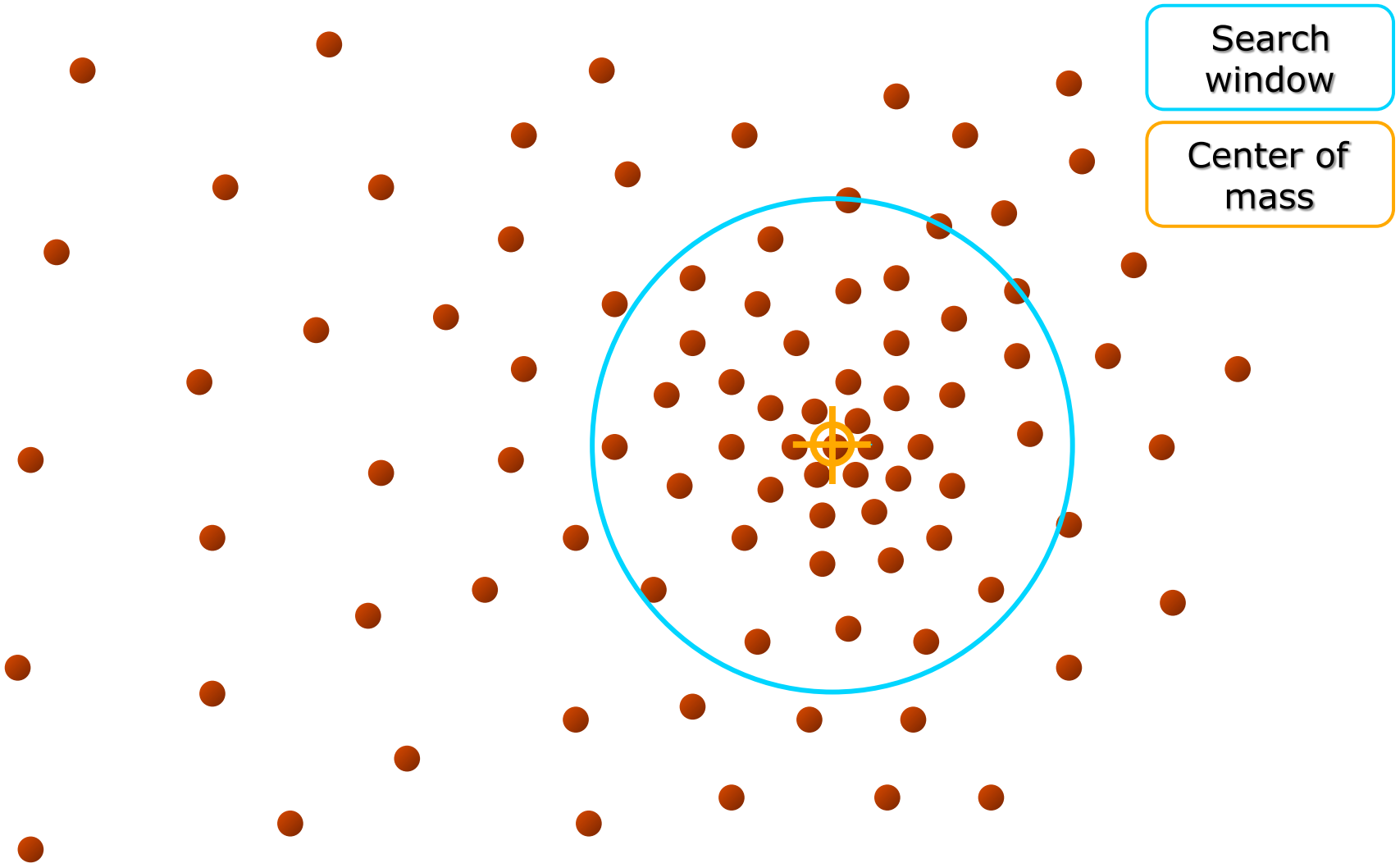
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

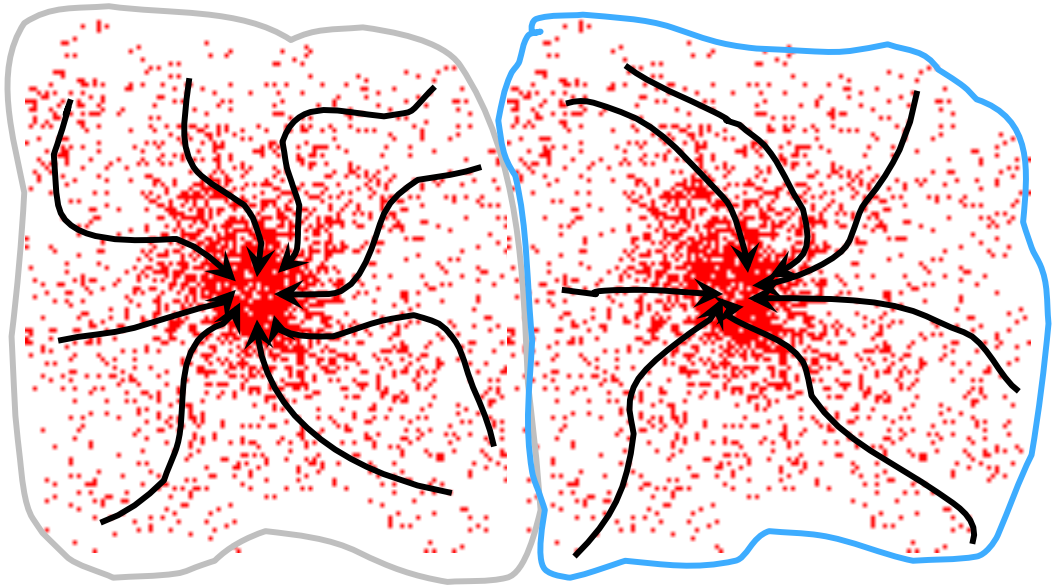
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

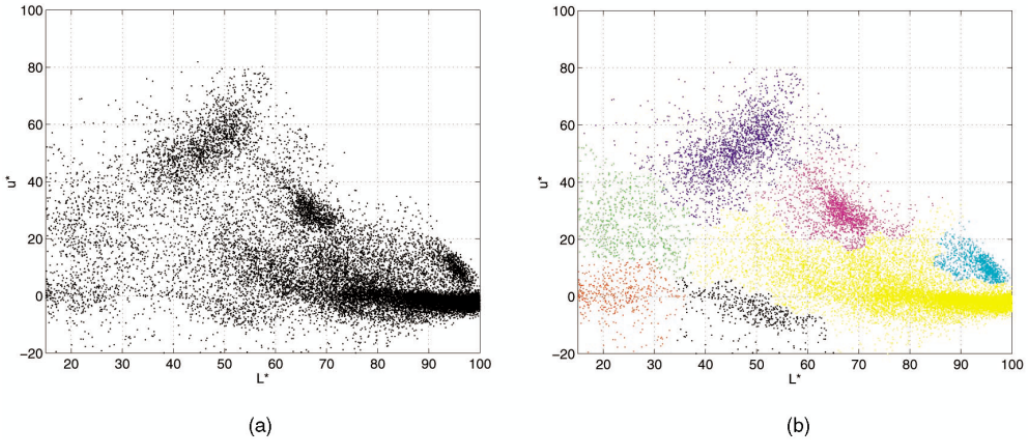
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

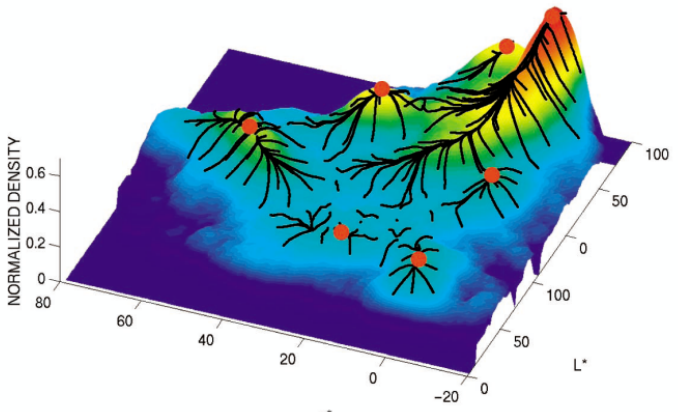


Slide by Y. Ukrainitz & B. Sarel

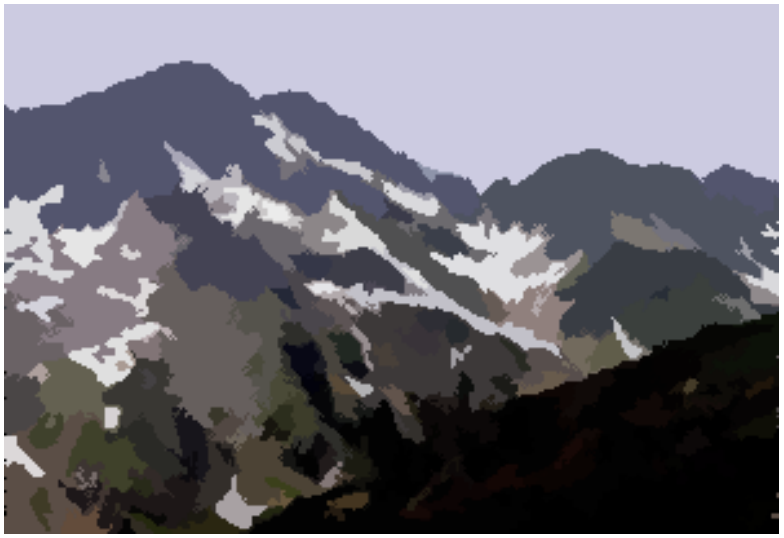
Mean shift clustering/segmentation

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





Mean shift segmentation results



ip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean shift segmentation results

Background Subtraction

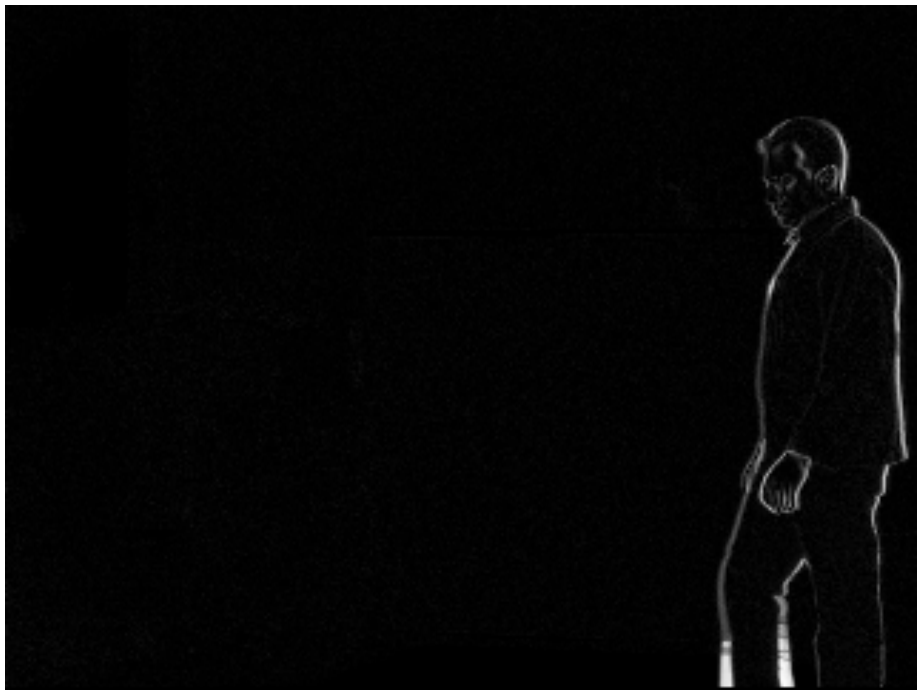
- If we know what the background looks like, it is easy to identify "interesting bits"
- Applications
  - Person in an office
  - Tracking cars on a road
  - surveillance
- Approach:
  - use a moving average to estimate background image
  - subtract from current frame
  - large absolute values are interesting pixels
    - trick: use morphological operations to clean up pixels

Image Differencing




Image Differencing: Results

1 frame difference



5 frame difference

Motion detection

- Background subtraction
  - create an image of the stationary background by averaging a long sequence
    - for any pixel, most measurements will be from the background
    - computing the median measurements, for example, at each pixel, will with high probability assign that pixel the true background intensity - fixed threshold on differencing used to find “foreground” pixels
    - can also compute a distribution of background pixels by fitting a mixture of Gaussians to set of intensities and assuming large population is the background - adaptive thresholding to find foreground pixels

A 300-Frame Sequence with a “Busy” Background

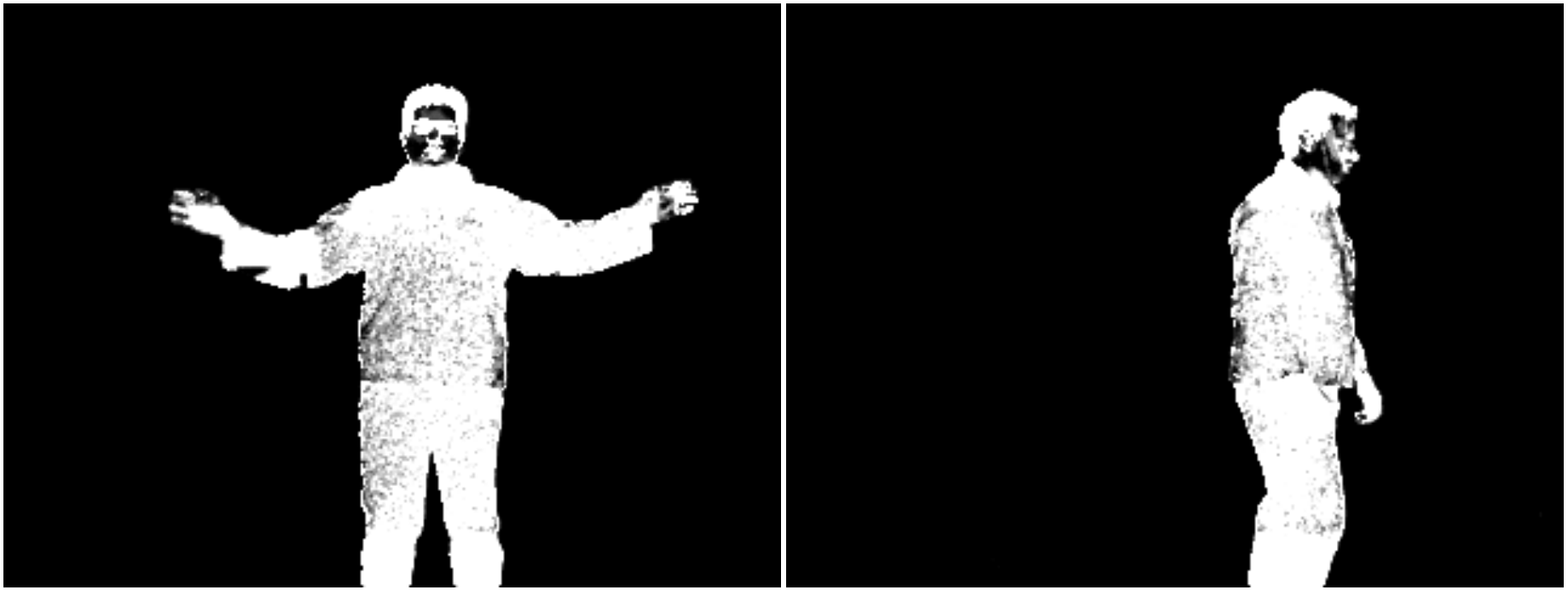
click to start movie

Motion Detection

  - difference a frame from the known background frame
    - even for interior points of homogeneous objects, likely to detect a difference
    - this will also detect objects that are stationary but different from the background
    - typical algorithm used in surveillance systems
- Motion detection algorithms such as these only work if the camera is stationary and objects are moving against a fixed background

Background Subtraction: Results

Confidence corresponds to gray-level value.
High confidence – bright pixels, low confidence – dark pixels.



Background modeling: color-based

- At each pixel model colors *(r,g,b)* or gray-level values *g.* The following equations are used to recursively estimate the mean and the variance at each pixel:

$$\mu_{t+1} = \alpha\mu_t + (1-\alpha)z_{t+1}$$
$$\sigma^2_{t+1} = \alpha(\sigma^2_t + (\mu_{t+1} - \mu_t)^2) + (1-\alpha)(z_{t+1} - \mu_{t+1})^2$$

where z_{t+1} is the current measurement. The mean μ and the variance σ can both be time varying. The constant α is set empirically to control the rate of adaptation ($0<\alpha<1$).

- A pixel is marked as foreground if given red value *r* (or for any other measurement, say *g* or *b*) we have

$$|r - \mu_t| > 3\max(\sigma_r, \sigma_{rcam})$$

Background model

- σ_{rcam} is the variance of the camera noise, can be estimated from image differences of any two frames.
- If we compute differences for all channels, we can set a pixel as foreground if any of the differences is above the preset threshold.
- Noise can be cleaned using connected component analysis and ignoring small components.
- Similarly we can model the chromaticity values rc, gc and use them for background subtraction:

  $r_c=r/(r+g+b)$, $g_c=g/(r+g+b)$

Background model: edge-based

- Model edges in the image. This can be done two different ways:
  - Compute models for edges in a the average background image
  - Subtract the background (model) image and the new frame; compute edges in the subtraction image; mark all edges that are above a threshold.
    - The threshold can be learned from examples
    - The edges can be combined (color edges) or computed separately for all three color channels

Foreground model

- Use either color histograms (4-bit per color), texture features, edge histograms to model the foreground
- Matching the foreground objects between frames: **tracking**
- Can compare foreground regions directly: shift and subtract. SSD or correlation: *M, N* are two foreground regions.

$$SSD = \sum_{i=1}^{n}\sum_{j=1}^{n}[M(i,j) - N(i,j)]^2$$

$$C = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)N(i,j)}{\left[\sum_{i=1}^{n}\sum_{j=1}^{n} M(i,j)^2 \sum_{i=1}^{n}\sum_{j=1}^{n} N(i,j)^2\right]^{1/2}}$$

Histogram Matching

- Histogram Intersection

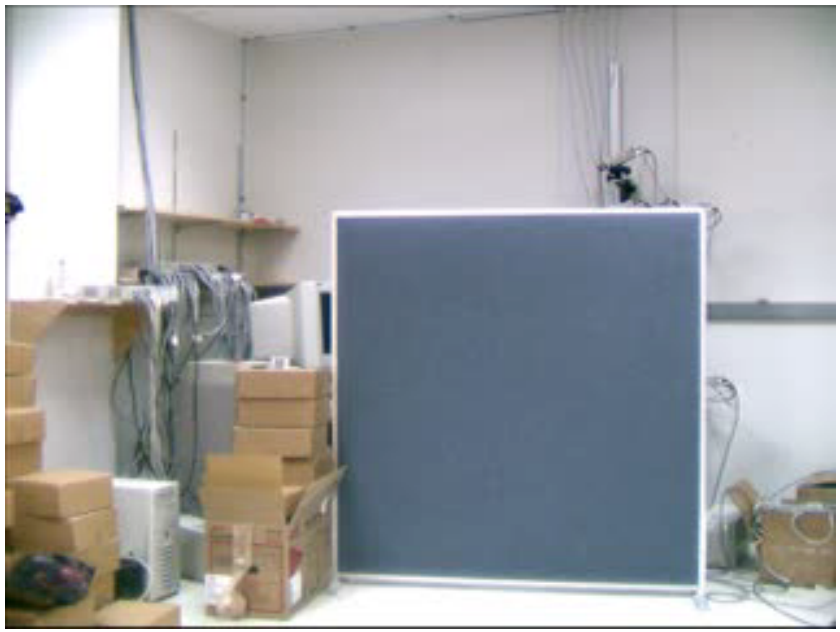
$$I(h_c, h_b) = \frac{\sum_i \min\{h_c(i), h_b(i)\}}{\sum_i \max\{h_c(i), h_b(i)\}}$$

- Chi Squared Formula

$$\chi^2(h_c, h_b) = \sum_i 2\frac{(h_c(i) - h_b(i))^2}{h_c(i) + h_b(i)}$$

Surveillance: Interacting people

Background Subtraction



Motion Segmentation

Given a set of image points obtain:

- Number of independently moving objects
- Segmentation: object to which each point belongs
- Motion: rotation and translation of each object
- Structure: depth of each point

Motion Segmentation Problem

Three frames from the MPEG "flower garden" sequence

- Given optical flow at each point
- partition/segment the flow field into regions belonging to individual planes "layers"

Figure from "Representing Images with layers,", by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

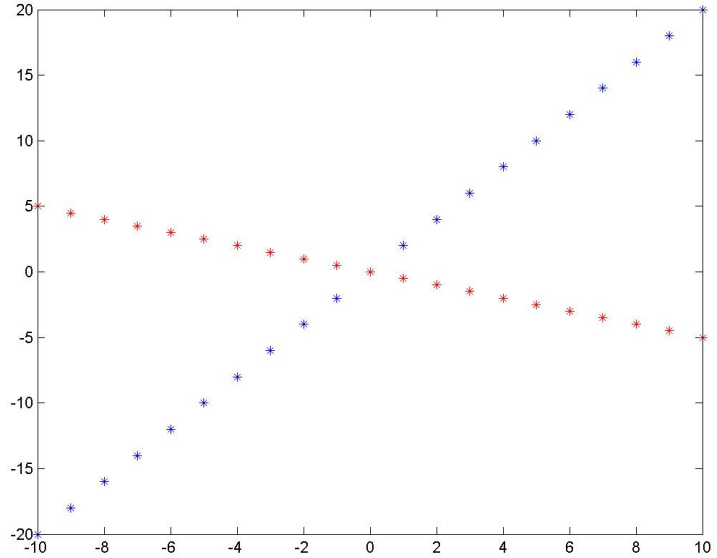
Some example slides from Forsythe and Ponce. Computer Vision, A modern approach.

Model Estimation and Grouping

- Given a set of data points and a particular model
- The model parameters can be estimated by LSE fitting data to the model
- Previous model examples – essential/fundamental matrix, homographies, lines/planes
- In order to estimate the model parameters we need to know which data point belongs to which model
- Difficulty – we have multiple models – we do not know initially which data point belongs to which model and we do not the model parameters
- chicken and egg problem

Model Estimation and Grouping

- Line Example
- Set of points belonging to two lines
- We need to estimate
  1. parameters of the lines
     slope and intercept
  2. which point belongs to
     which line



Solution: EM algorithm
Idea: Each of the above steps
Assumes the other one is
solved and iterate

EM algorithm

- Basic structure of the EM algorithm

- 1. Start with random parameter values for each model
- 2. Iterate until parameter values converge

  E step: assign points to the model that fits best

  M step : update the parameters of the models using only points assigned to it

Simplistic explanation here –

Theoretical foundation probabilistic (model parameters are random variables) - EM (Expectation Maximization)

E- Step

- Case of two lines given by slopes and intercepts
  (a_1, b_1) and (a_2, b_2)
- For each data point i, estimate the residual
  (difference between the prediction and the model)

$$r_1(i) = a_1 x_i + b_1 - y_i$$

$$r_2(i) = a_2 x_i + b_2 - y_i$$

- Calculate the weights, which correspond to the probabilities of particular data point belonging to particular model

$$w_1(i) = \frac{e^{-r_1^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}} \qquad w_2(i) = \frac{e^{-r_2^2(i)/\sigma^2}}{e^{-r_1^2(i)/\sigma^2} + e^{-r_2^2(i)/\sigma^2}}$$

M-step

- Given the weights recalculate the parameters of the model

$$(a_1, b_1) \text{ and } (a_2, b_2)$$

- Least squares estimation of line parameters

$$\begin{bmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & \sum 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i x_i y_i \\ \sum_i y_i \end{bmatrix}$$

- In our case we will have weighted least squares estimation of line parameters

$$\begin{bmatrix} \sum_i w_i x_i^2 & \sum_i w_i x_i \\ \sum_i w_i x_i & \sum w_i 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_i w_i x_i y_i \\ \sum_i w_i y_i \end{bmatrix}$$

- Solve such estimation problem twice – once for each line

M-step

- Iterate until the parameters of the
- Lines converge

- Issues with EM
- Local maxima
  - can be a serious nuisance in some problems
  - no guarantee that we have reached the "right" maximum
- Starting if we do not know how many models we have
  - k means to cluster the points is often a good idea

Example: motion segmentation

- Consider motion model, when the flow field can be approximated by some parametric model with small number of parameters

- We can write x and y parameters of the flow (u, v) field – (u_1, v_1) (u_2, v_2)

  assume that models are locally translational, i.e. we can locally approximate the model by pure translation
- Suppose entire flow field can be explained by two translational models
- EM algorithm can be applied in analogous way

Example: motion segmentation

- Compute residuals

$$r_1(i,j)^2 = (u_1 - v_x(i,j))^2 + (v_1 - v_y(i,j)^2)$$

$$r_2(i,j)^2 = (u_2 - v_x(i,j))^2 + (v_2 - v_y(i,j)^2)$$

- Compute associated weights

$$w_1(i,j) = \frac{e^{-r_1^2(i,j)/\sigma^2}}{e^{-r_1^2(i,j)/\sigma^2} + e^{-r_2^2(i,j)/\sigma^2}} \quad w_2(i,j) = \frac{e^{-r_2^2(i,j)/\sigma^2}}{e^{-r_1^2(i,j)/\sigma^2} + e^{-r_2^2(i,j)/\sigma^2}}$$

- $$\begin{bmatrix} \sum_{i,j} w_1(i,j) & 0 \\ 0 & \sum_{i,j} w_1(i,j) \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \sum_{i,j} w_1(i,j) v_x(i,j) \\ \sum_{i,j} w_1(i,j) v_y(i,j) \end{bmatrix}$$

Iterate until convergence

Example: motion segmentation

- Model image pair (or video sequence) as consisting of regions of parametric motion
  - affine motion – commonly used –
  - Approximates locally motion of the planar surface

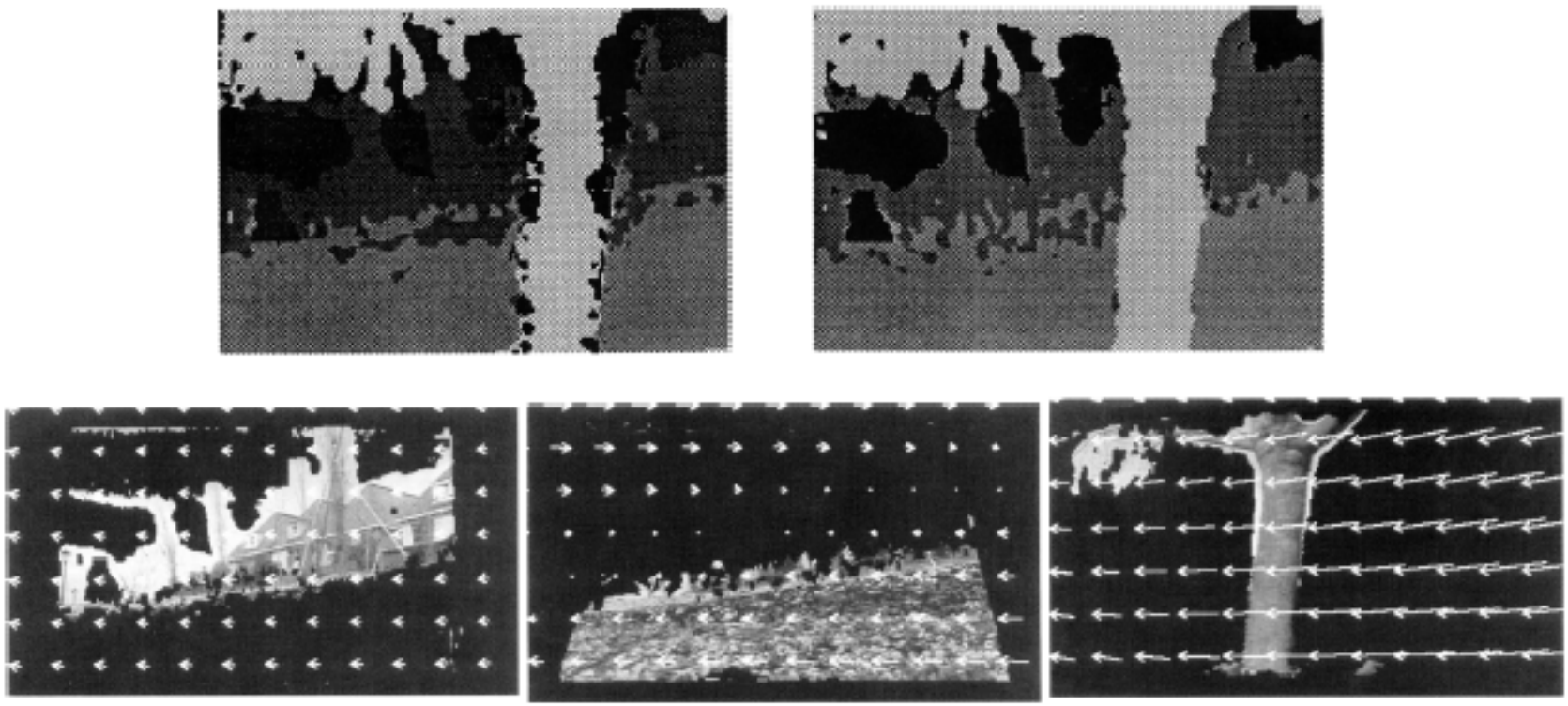
$$\begin{bmatrix} v_x(x,y) \\ v_y(x,y) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$$

- Now we need to
  - determine which pixels belong to which region
  - estimate parameters

Three frames from the MPEG “flower garden” sequence

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

Grey level shows region no. with highest probability



Segments and motion fields associated with them

Figure from “Representing Images with layers,”, by J. Wang and E.H. Adelson, IEEE Transactions on Image Processing, 1994, c 1994, IEEE

Next, graph based segmentation …