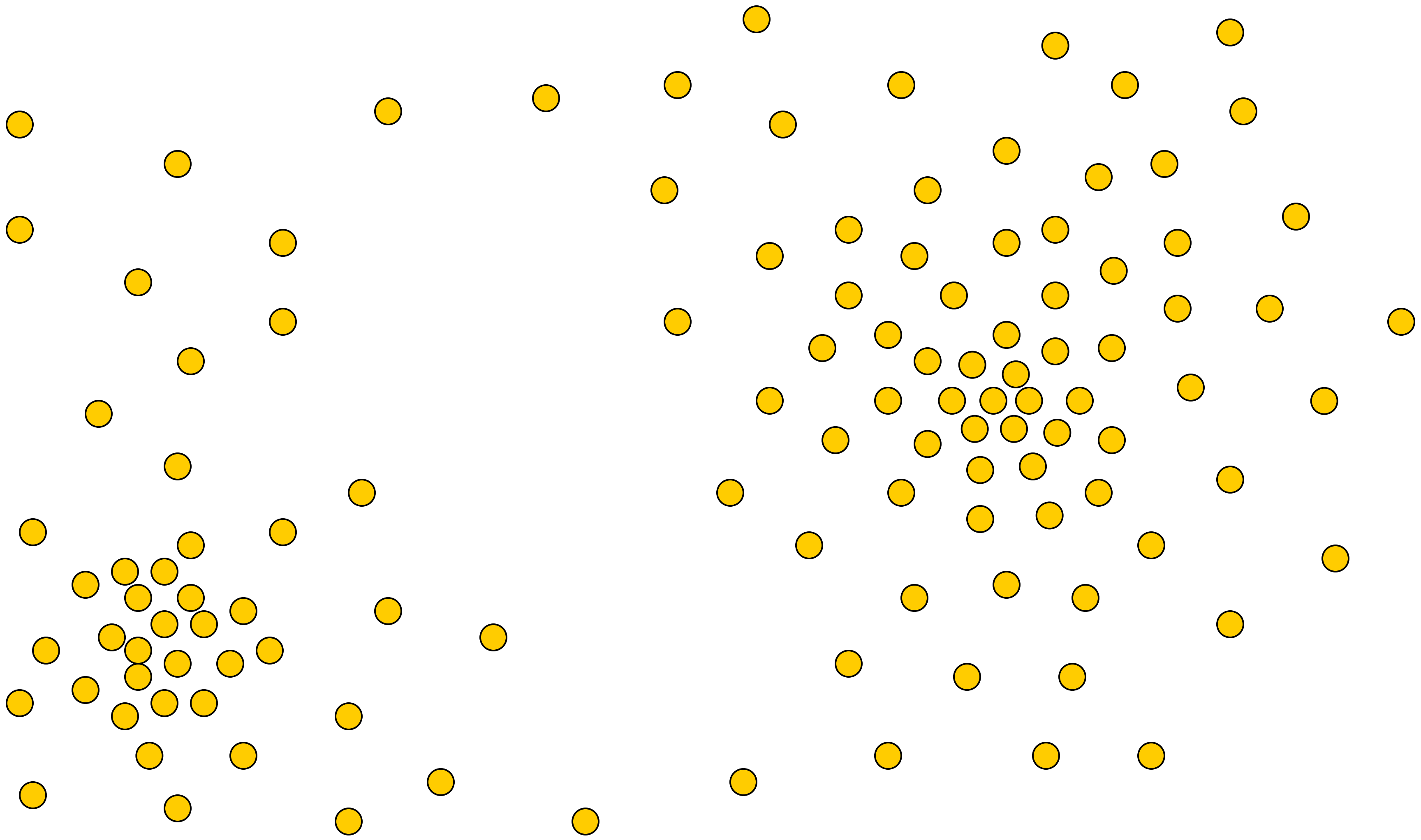
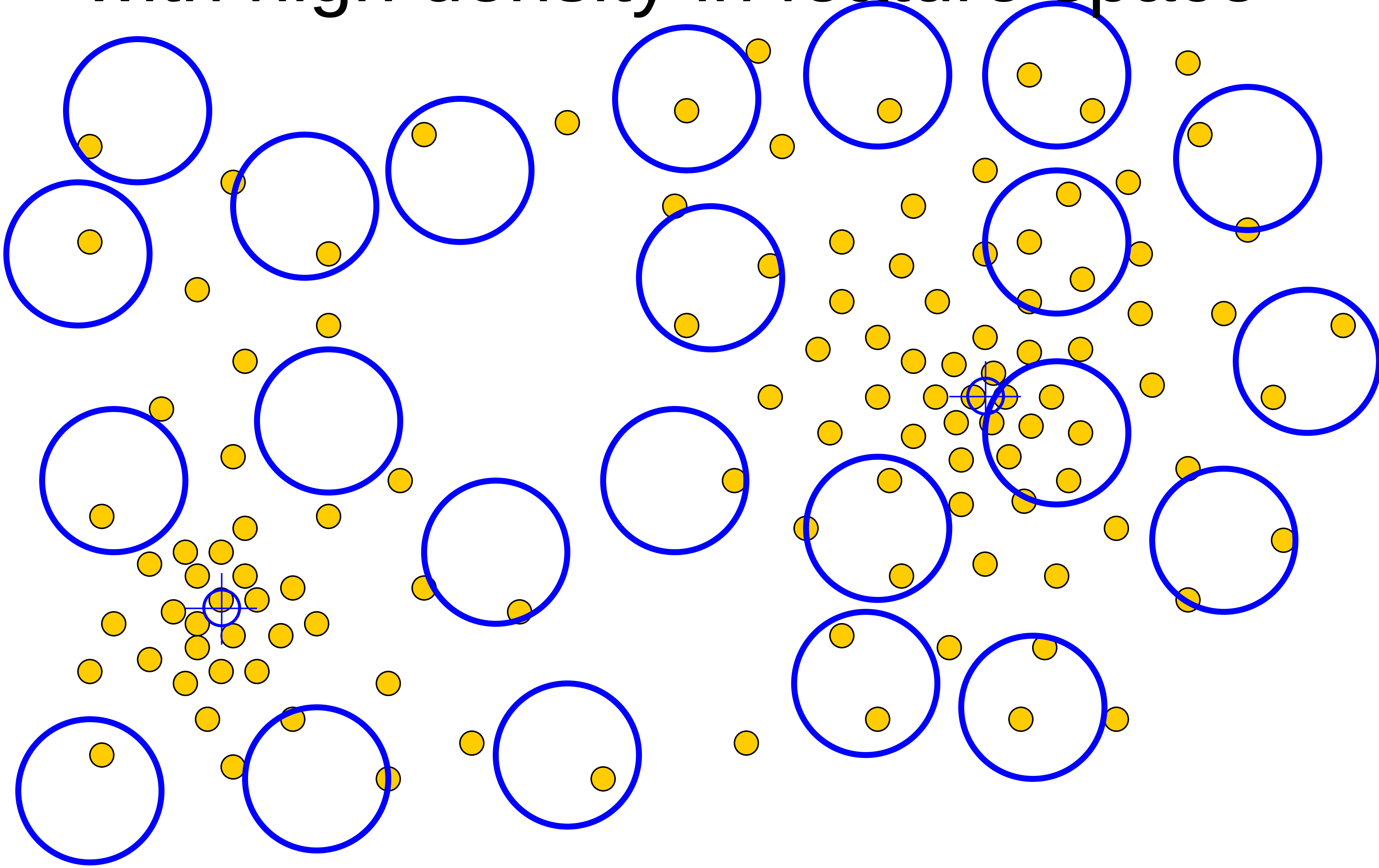
Segmentation

Mean-Shift Algorithm

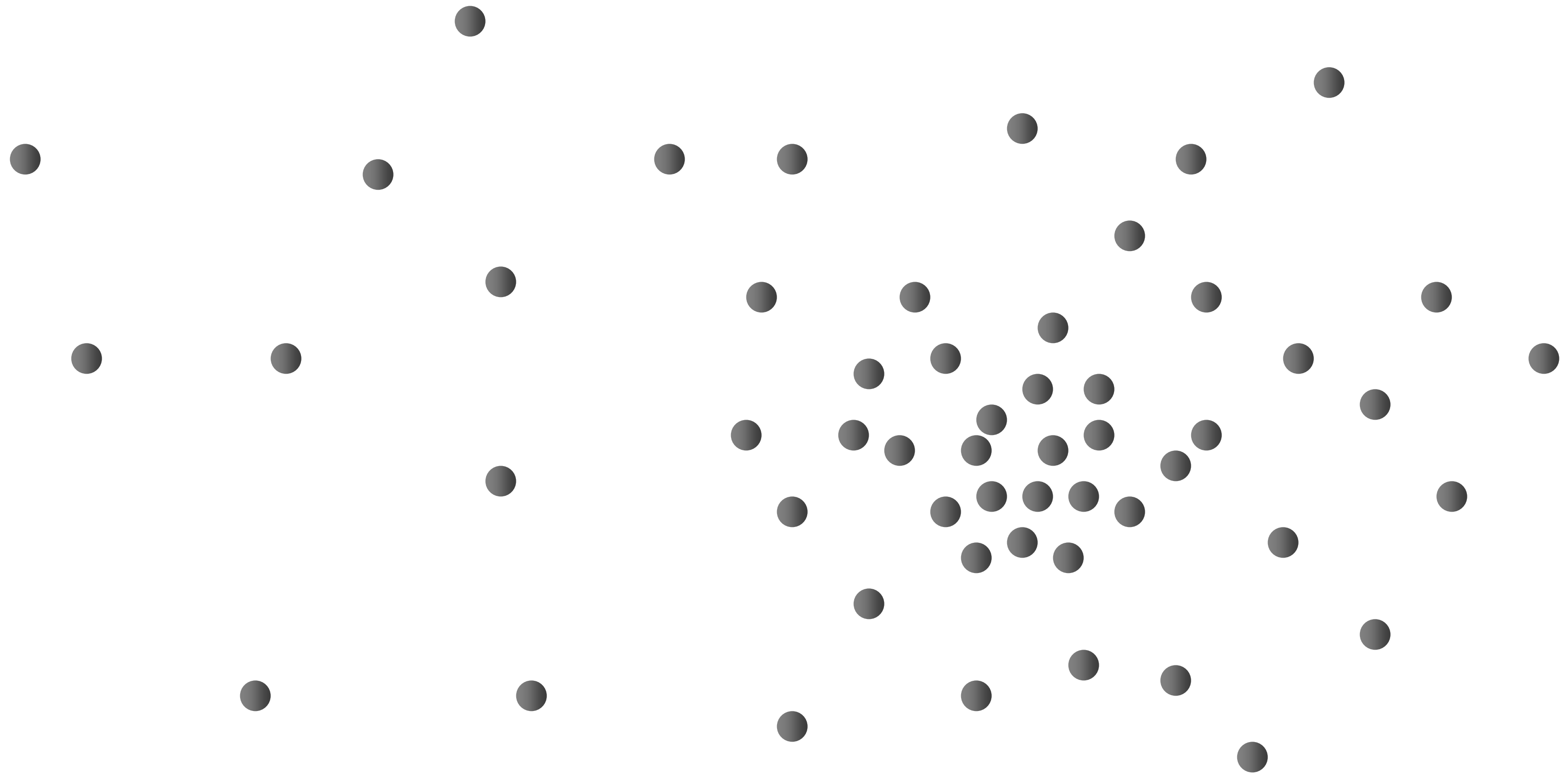
Segmentation as finding places with high density in feature space



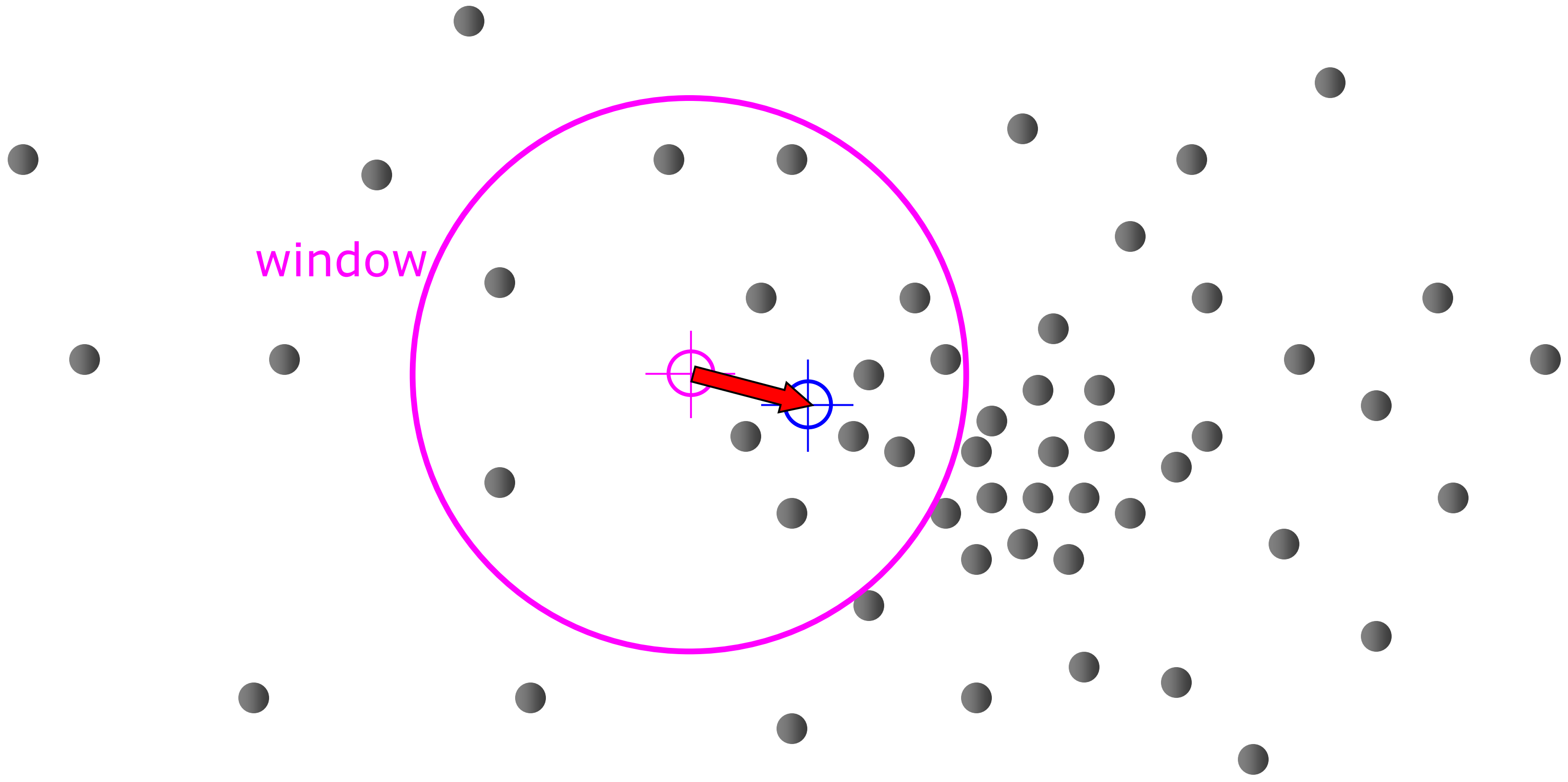
Segmentation as finding places with high density in feature space



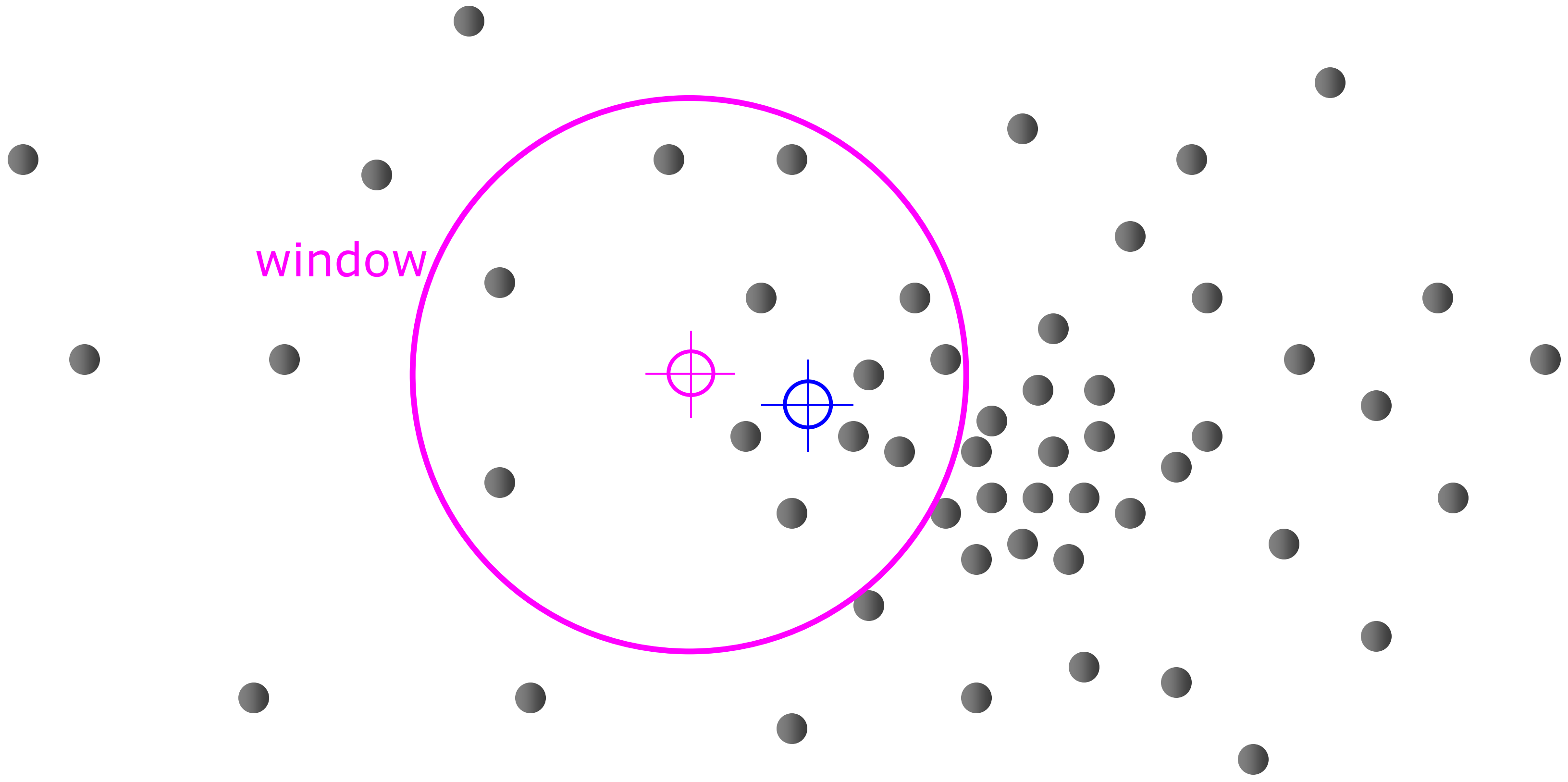
Segmentation as finding places with high density in feature space



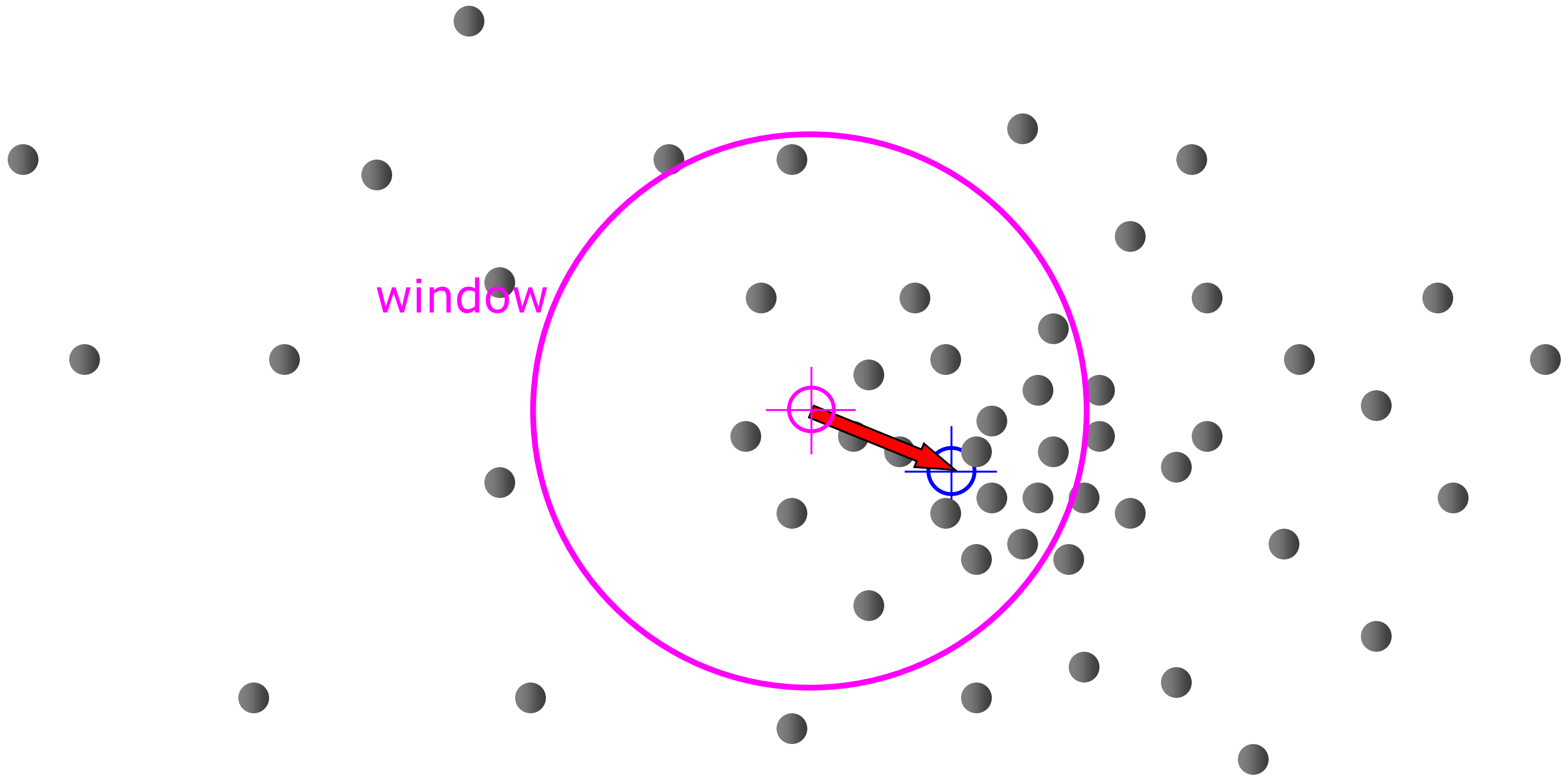
Example from: Ukrainitz&Sarel, Weizmann



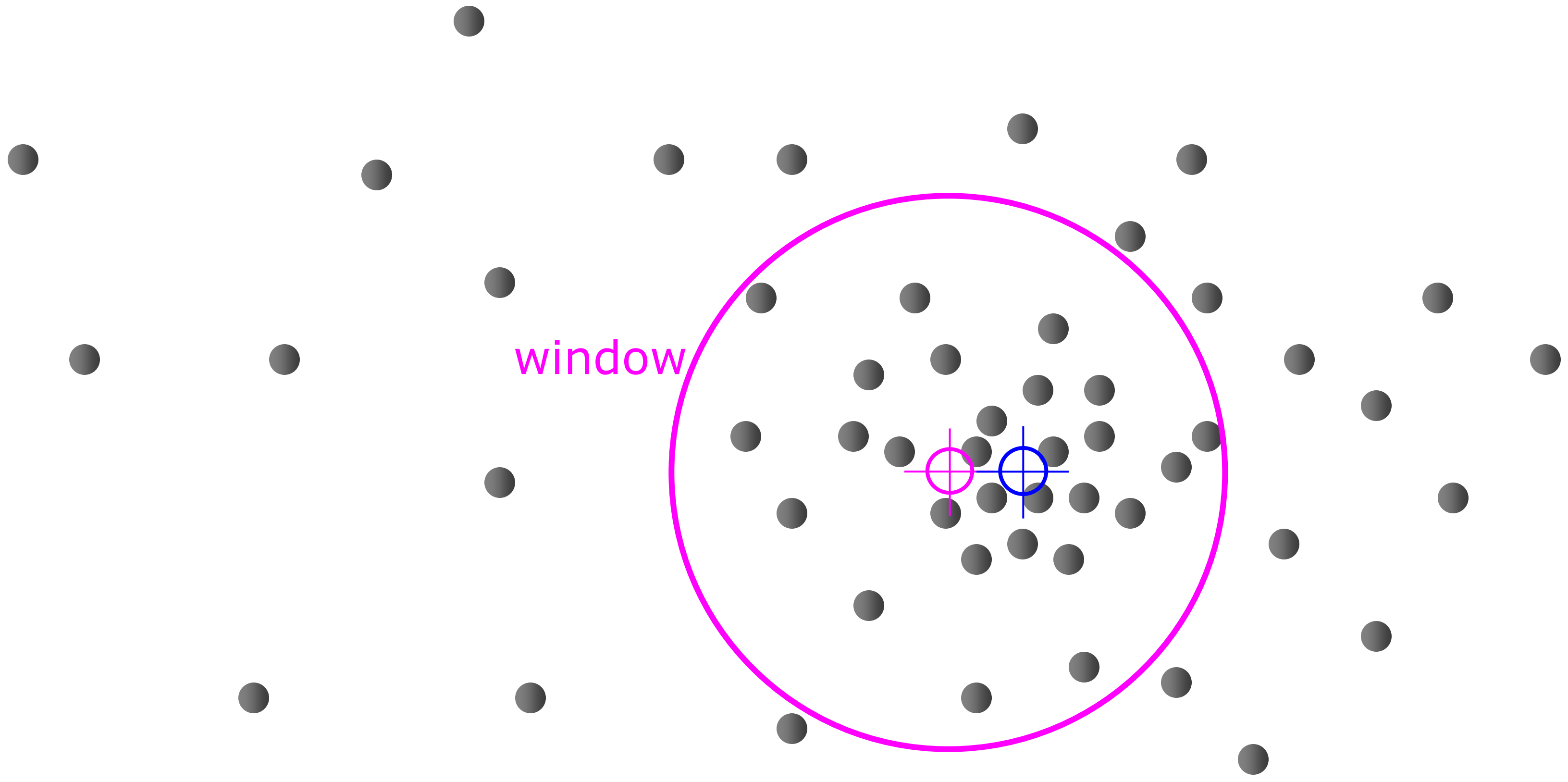
Example from: Ukrainitz&Sarel, Weizmann



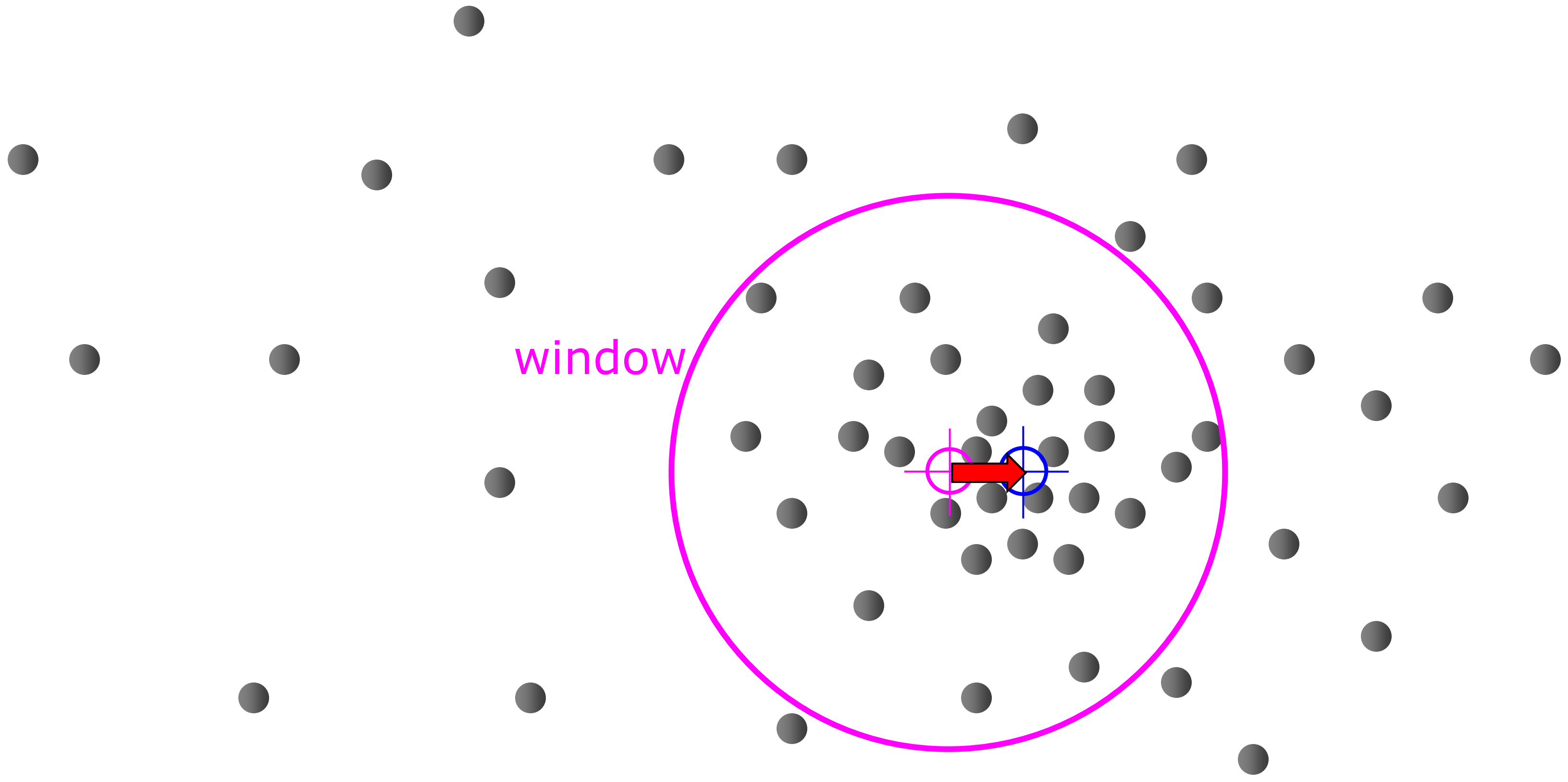
Example from: Ukrainitz&Sarel, Weizmann



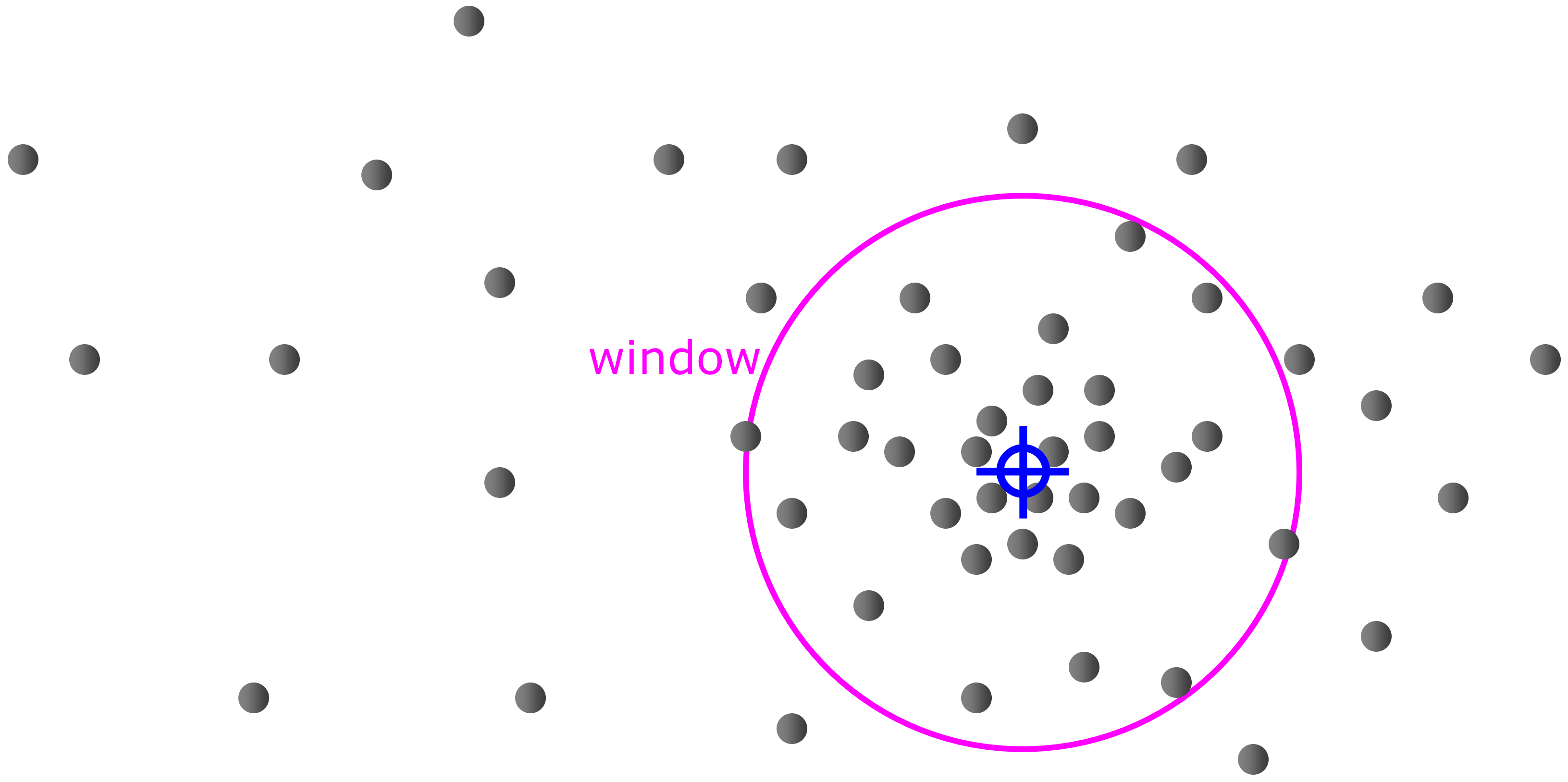
Example from: Ukrainitz&Sarel, Weizmann



Example from: Ukrainitz&Sarel, Weizmann

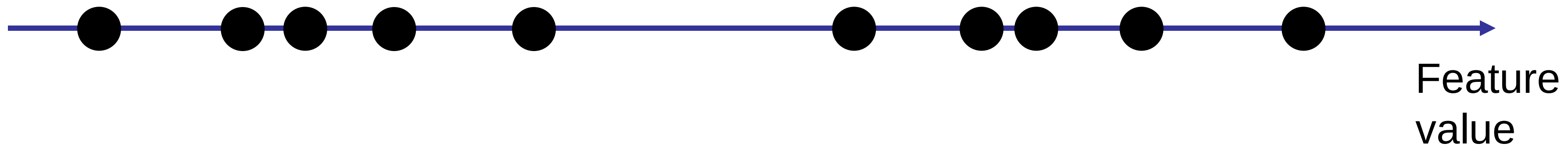


Example from: Ukrainitz&Sarel, Weizmann



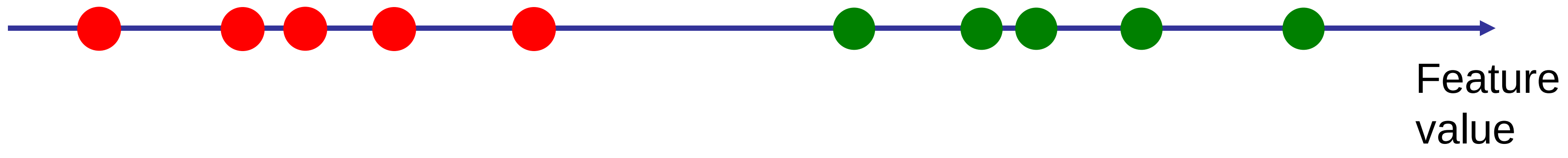
Example from: Ukrainitz&Sarel, Weizmann

A 1-D Example



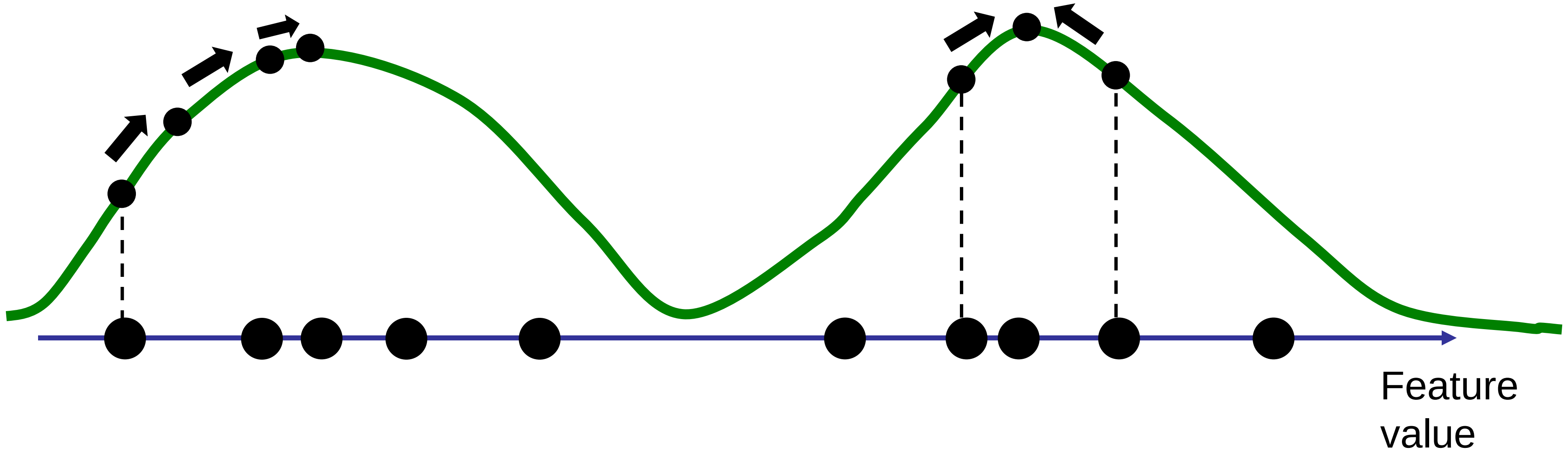
- Consider a set of points in a boring, one-dimensional feature space

A 1-D Example



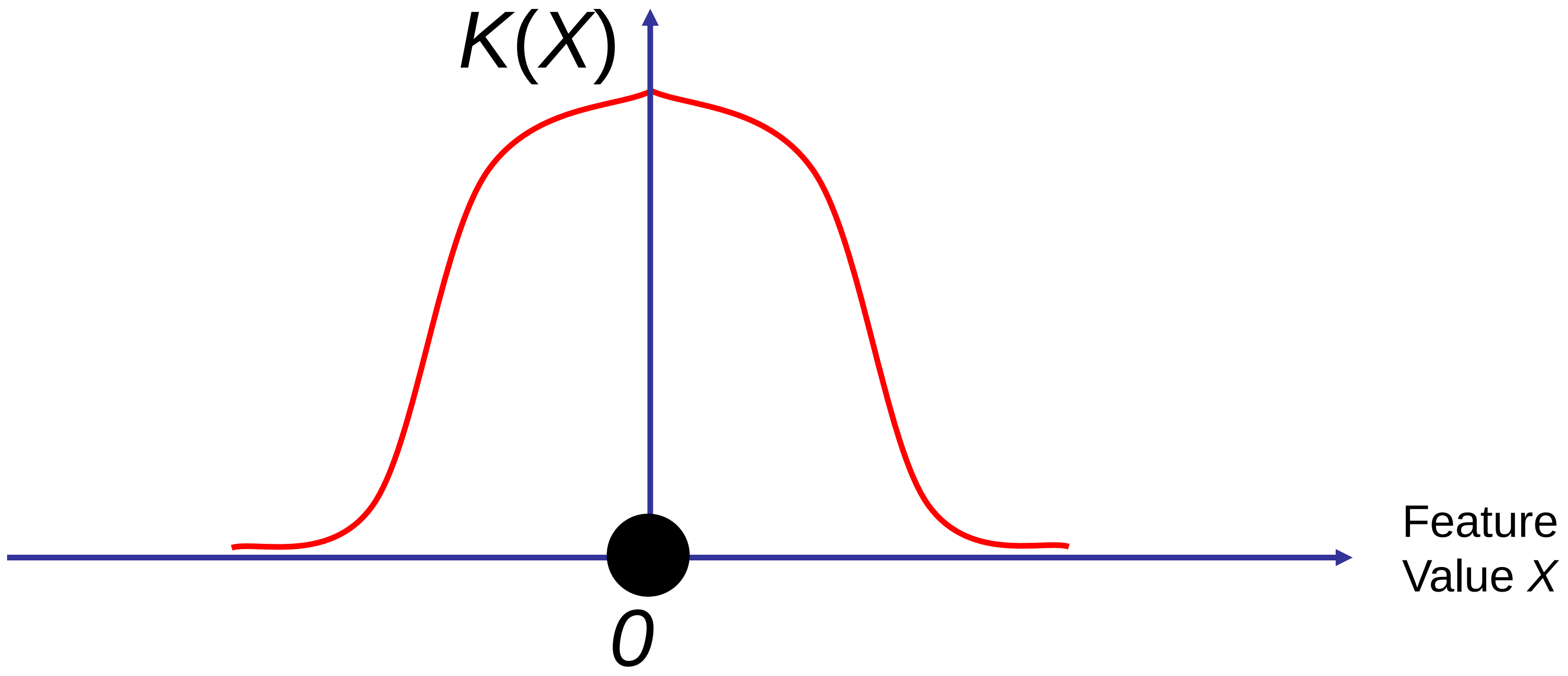
- Obviously, we would like to generate two groups, corresponding to the two parts of the feature space in which we have a *high density* of points
- How can we capture this notion of “high density” → *kernel density estimation*

A 1-D Example



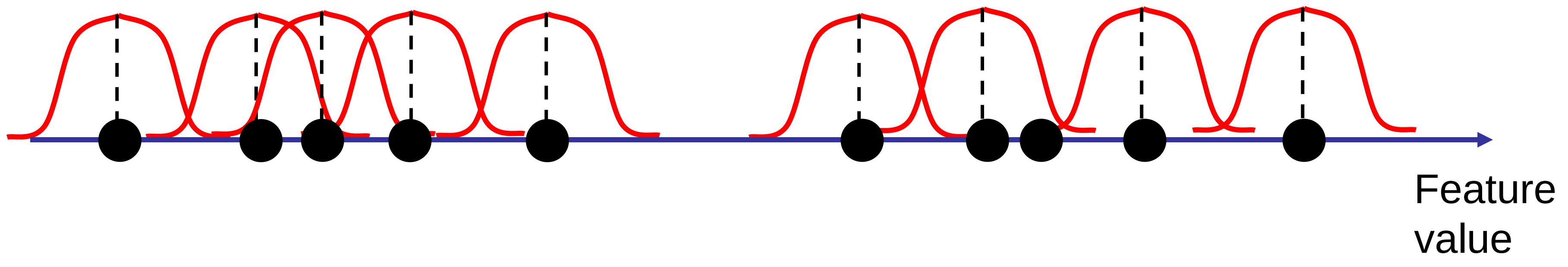
- If we had a continuous function instead of a bunch of data points, we could find the maxima by gradient ascent.
- How can we convert our set of points to a continuous function?

A 1-D Example



- Let us define a kernel function: $K(X)$, with the properties:
- K decays to zero far from 0
- K is maximum at 0
- K is symmetric

A 1-D Example

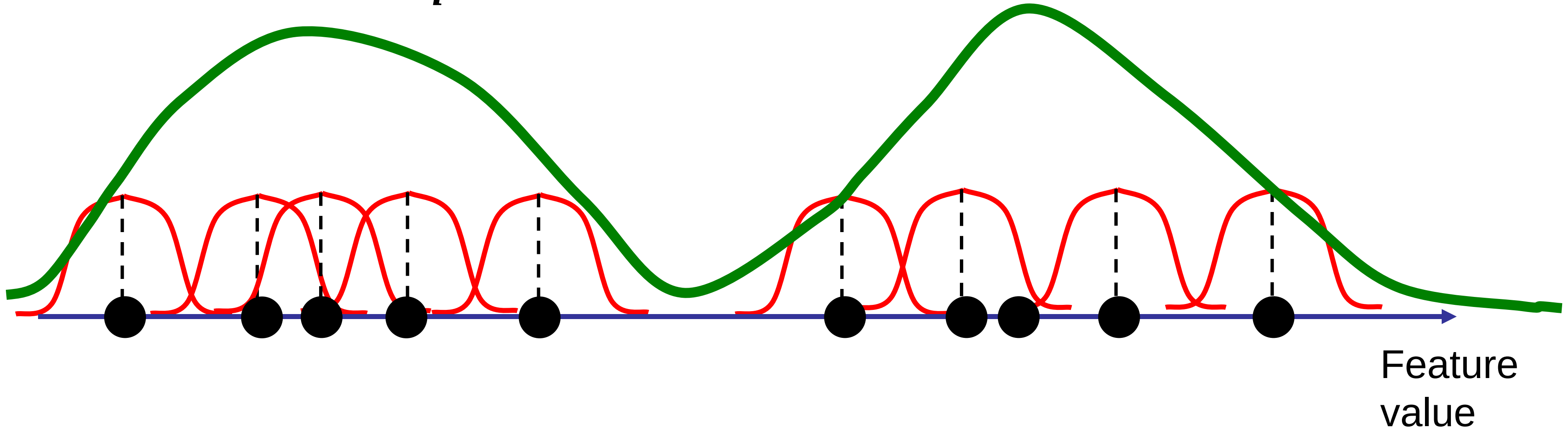


- We can define the kernel at each data point and sum up the result into a single function:

$$f(X) = \frac{1}{N}\sum_i K(X - X_i)$$

A 1-D Example

$$f(X) = \frac{1}{N}\sum_i K(X - X_i)$$



- *V* is a normalization term
- *f(X) approximates the probability that feature X is observed given the data points*
- The maxima of *f* (the modes of the pdf) correspond to the clusters in the data

What do these kernels really mean?

- Recall affinity values from previous lecture:

$$\boldsymbol{m}_{ij} = \exp - \left\| \boldsymbol{X}_i - \boldsymbol{X}_j \right\|^2 / \sigma^2$$

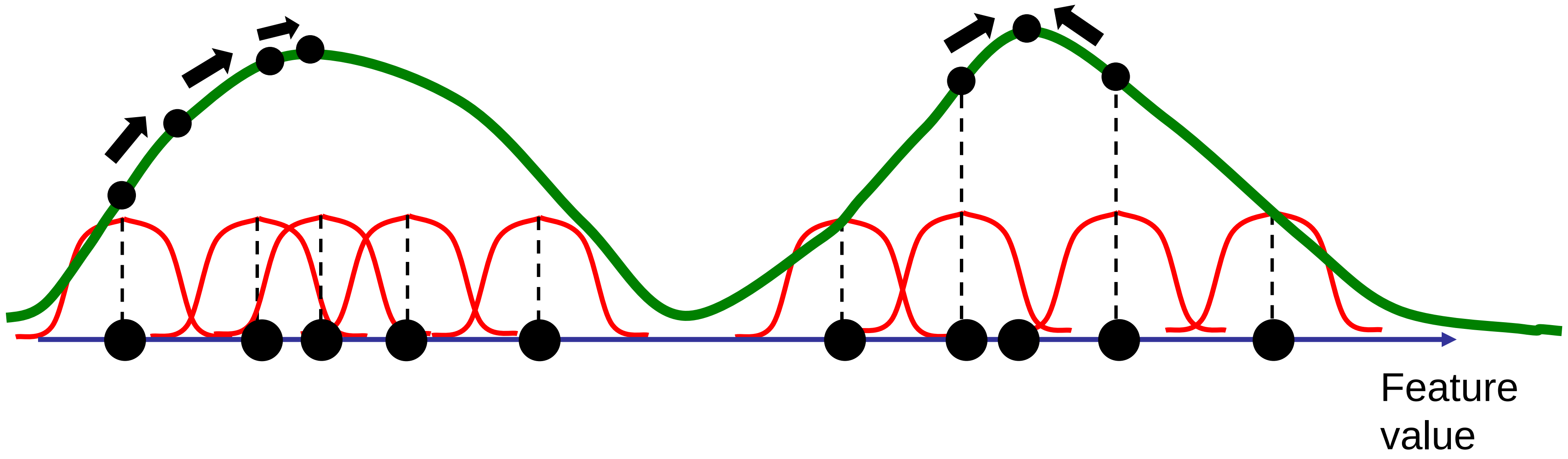
- Think of a kernel as measuring how much two data points look alike

$$\boldsymbol{m}_{ij} = \exp - \left\| \boldsymbol{X}_i - \boldsymbol{X}_j \right\|^2 / \sigma^2 = \boldsymbol{K}(\boldsymbol{X}_i - \boldsymbol{X}_j)$$

$$\boldsymbol{K}(\boldsymbol{X}) = \exp - \left\| \boldsymbol{X} \right\|^2 / \sigma^2$$

A 1-D Example

$$f(X) = \frac{1}{N} \sum_i K(X - X_i)$$



- If we move each point in the direction of the gradient, we will converge to the closest mode
- How can we do this efficiently?

General Algorithm

- For *i* =1,..,*N*

  $$\boldsymbol{X} \leftarrow \boldsymbol{X}_i$$

  – Repeat

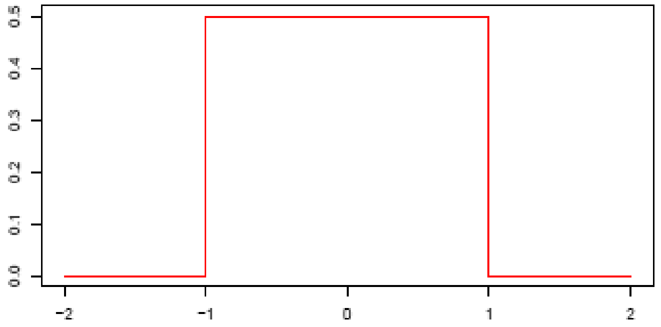
  $$\boldsymbol{X} \leftarrow \boldsymbol{X}_i + \nabla \boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{X}_i + \frac{1}{\boldsymbol{N}} \sum_{\boldsymbol{i}} \nabla \boldsymbol{K}(\boldsymbol{X} - \boldsymbol{X}_i)$$

  – Until *X* does not change

Example kernels

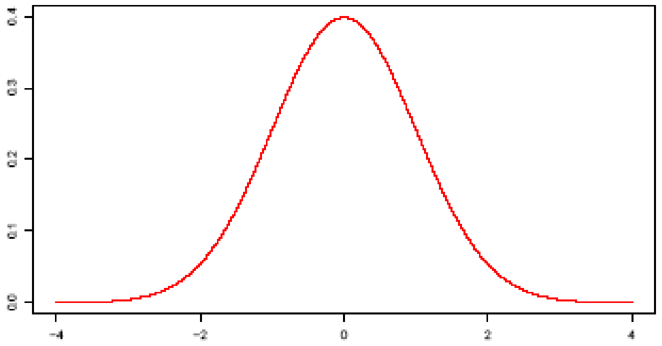
Uniform:

$$K_U(\mathbf{x}) = \begin{cases} c & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$



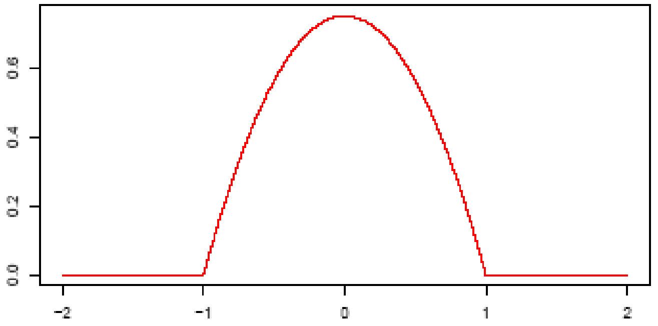
Gaussian:

$$K_N(\mathbf{x}) = c \cdot \exp\left(-\frac{1}{2} \|\mathbf{x}\|^2 \right)$$



Epanechnikov:

$$K_E(\mathbf{x}) = \begin{cases} c \;\; 1 - \|\mathbf{x}\|^2 & \|\mathbf{x}\| \le 1 \\ 0 & \text{otherwise} \end{cases}$$



Bandwith

- Kernel is defined as:

$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

- *h* is the bandwith of the kernel
- *k*(.) is:
  - For Gaussian: $k(t) = e^{-t/2}$
  - For Epanechnikov:

$$k(t) = \begin{cases} (1-t) & \text{if } |t| < 1 \\ 0 & \end{cases}$$

Bandwith

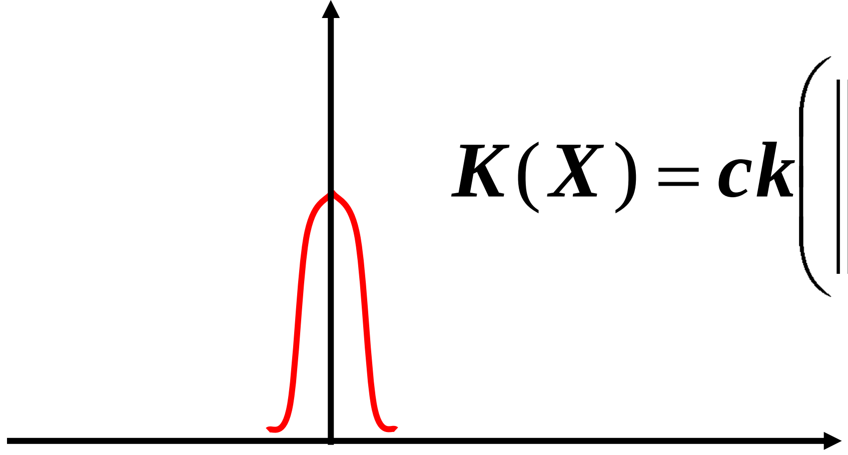
- Kernel is defined as:

$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

- *h* is the bandwith of the ke
- *k*( ) is:

Bandwidth *h* controls the radius of influence of each data point.
• Too small: Overfits the data points
• Too large: Smoothes out the details of the data

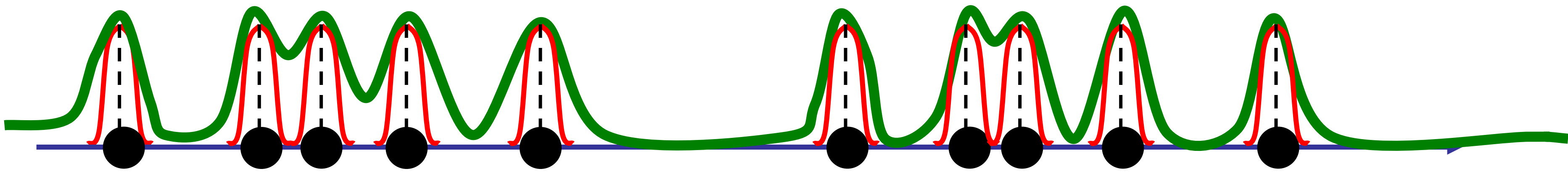
$$k(\;) = \begin{cases} \\ 0 \end{cases}$$



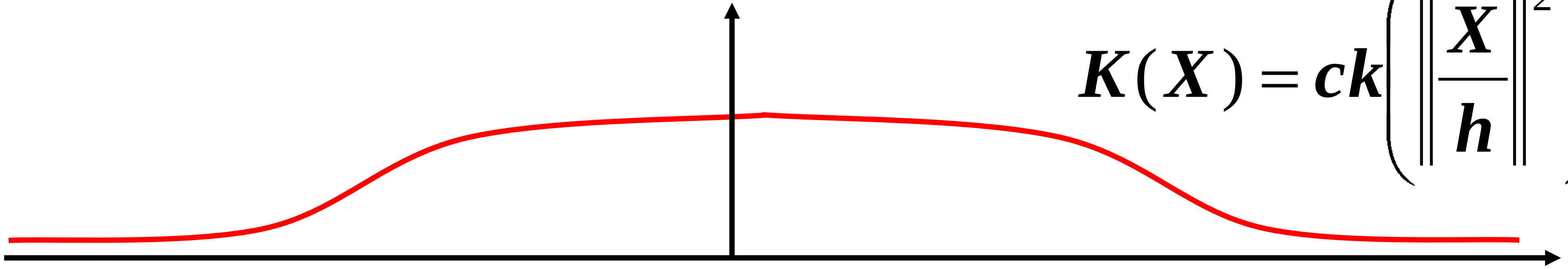
$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

h too small: The pdf overfits the noise in the data → Too many modes

$$f(X) = \sum_i ck\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$



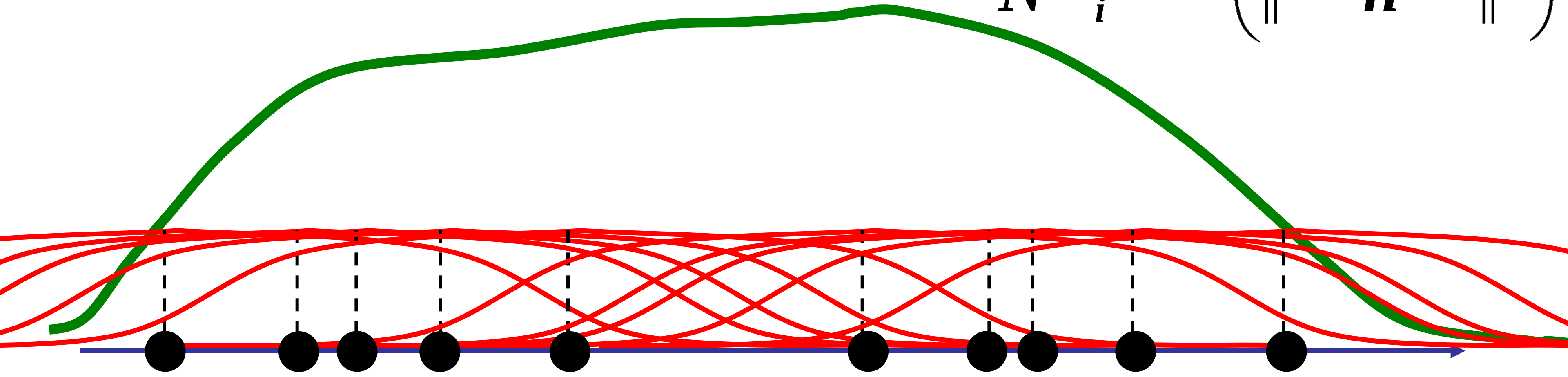
Feature value



$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

h too large: The details of the initial data are smoothed out
→ Too few modes

$$f(X) = \frac{1}{N}\sum_i ck\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$



Choice of kernel

- The kernel must satisfy a few technical conditions (*aka Parzen windows*).
- Integrates to 1 so that *f*(.) is a pdf: $$\boxed{\int_{R^d} K(\mathbf{x})d\mathbf{x} = 1}$$
- Symmetric
- Decays quickly (exponentially) as |*x*| increases: $$\boxed{\lim_{\|\mathbf{x}\|\to\infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0}$$
- The extent of the kernel is the same along all the dimensions: $$\boxed{\int_{R^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})d\mathbf{x} = c\mathbf{I}}$$

Computing the Gradient

- Now we have a representation of the pdf from which, in principle, we can find the modes by following the gradient.
- How can we do this efficiently?
- Notations:

$$g(t) = -k'(t)$$

- Gradient of each individual entry in the sum defining *f*(.):

$$\boxed{\nabla K(X - X_i) = \nabla\left(ck\left(\frac{\|X - X_i\|^2}{h^2} \right)\right) = \frac{2c}{h^2}(X_i - X)\, g\left(\frac{\|X - X_i\|^2}{h^2} \right)}$$

Computing the Gradient

- Gradient of the entire pdf:

$$\nabla f(X) = \frac{1}{N}\sum_i \nabla K(X - X_i) = \frac{2c}{Nh^2}\sum_i (X_i - X)\, g\left(\frac{\|X - X_i\|^2}{h^2}\right)$$

$$\Downarrow$$

$$\nabla f(X) = \left(\frac{2c}{Nh^2}\sum_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)\right)\left(\frac{\sum_i X_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)}{\sum_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)} - X\right)$$

$$\nabla f(\boldsymbol{X}) = \left(\frac{2c}{Nh^2} \sum_i g\left(\frac{\|\boldsymbol{X} - \boldsymbol{X}_i\|^2}{h^2} \right) \right) \underbrace{\left(\frac{\sum_i \boldsymbol{X}_i g\left(\frac{\|\boldsymbol{X} - \boldsymbol{X}_i\|^2}{h^2} \right)}{\sum_i g\left(\frac{\|\boldsymbol{X} - \boldsymbol{X}_i\|^2}{h^2} \right)} - \boldsymbol{X} \right)}$$

Mean shift vector, *M*(*X*) = Difference between *X* and the mean of the data points weighted by *g*(.) (points further from *X* count less)

- Key result: The mean shift vector points in the same direction as the gradient
- Solution: Iteratively move in the direction of the mean shift vector

The Mean-Shift Algorithm

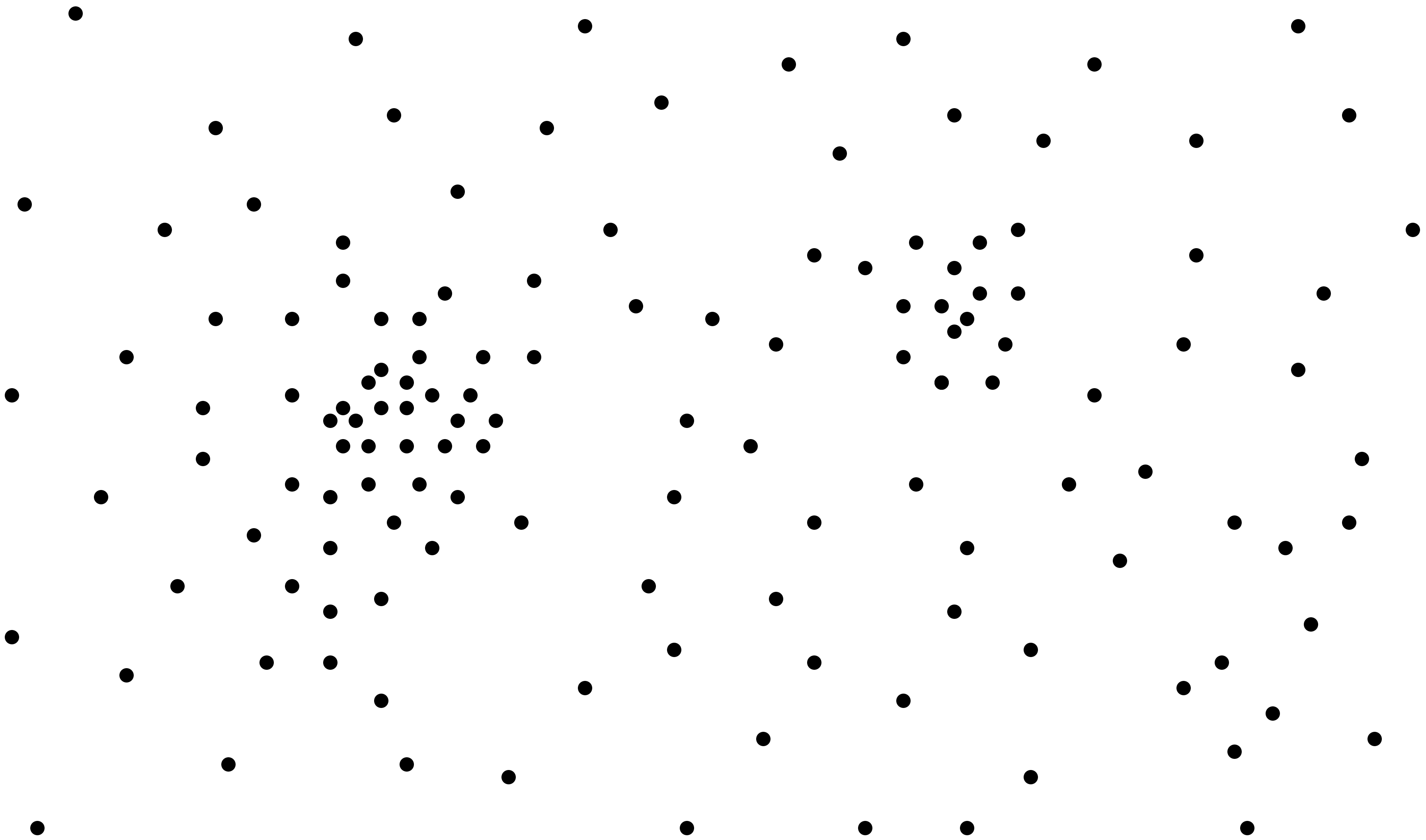
- Initialize: Set *X* to the value of the point to classify
- Repeat:
  - – Move *X* by the corresponding mean shift vector:

$$\boxed{X \leftarrow X + M(X) = \frac{\sum_i X_i\, g\left(\frac{\|X - X_i\|^2}{h^2}\right)}{\sum_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)}}$$

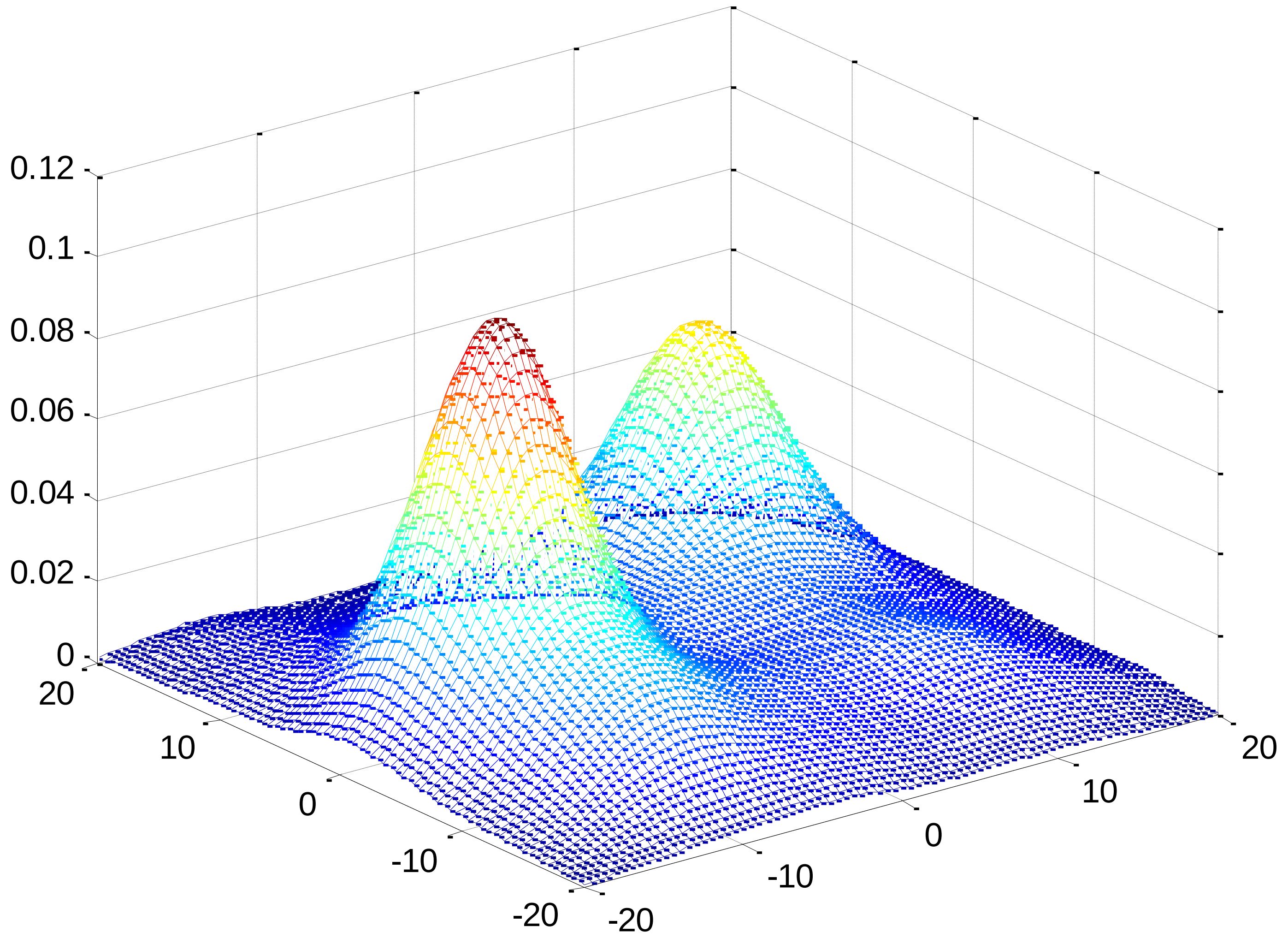
- Until *X* converges
- Note: Convergence is guaranteed.

2-D Example

$$\boldsymbol{f}(\boldsymbol{X}) = \frac{\boldsymbol{c}}{\boldsymbol{N}} \sum_{\boldsymbol{i}=1}^{N} \boldsymbol{k}\left(\left\| \frac{\boldsymbol{X} - \boldsymbol{X}_{\boldsymbol{i}}}{\boldsymbol{h}} \right\|^2 \right)$$

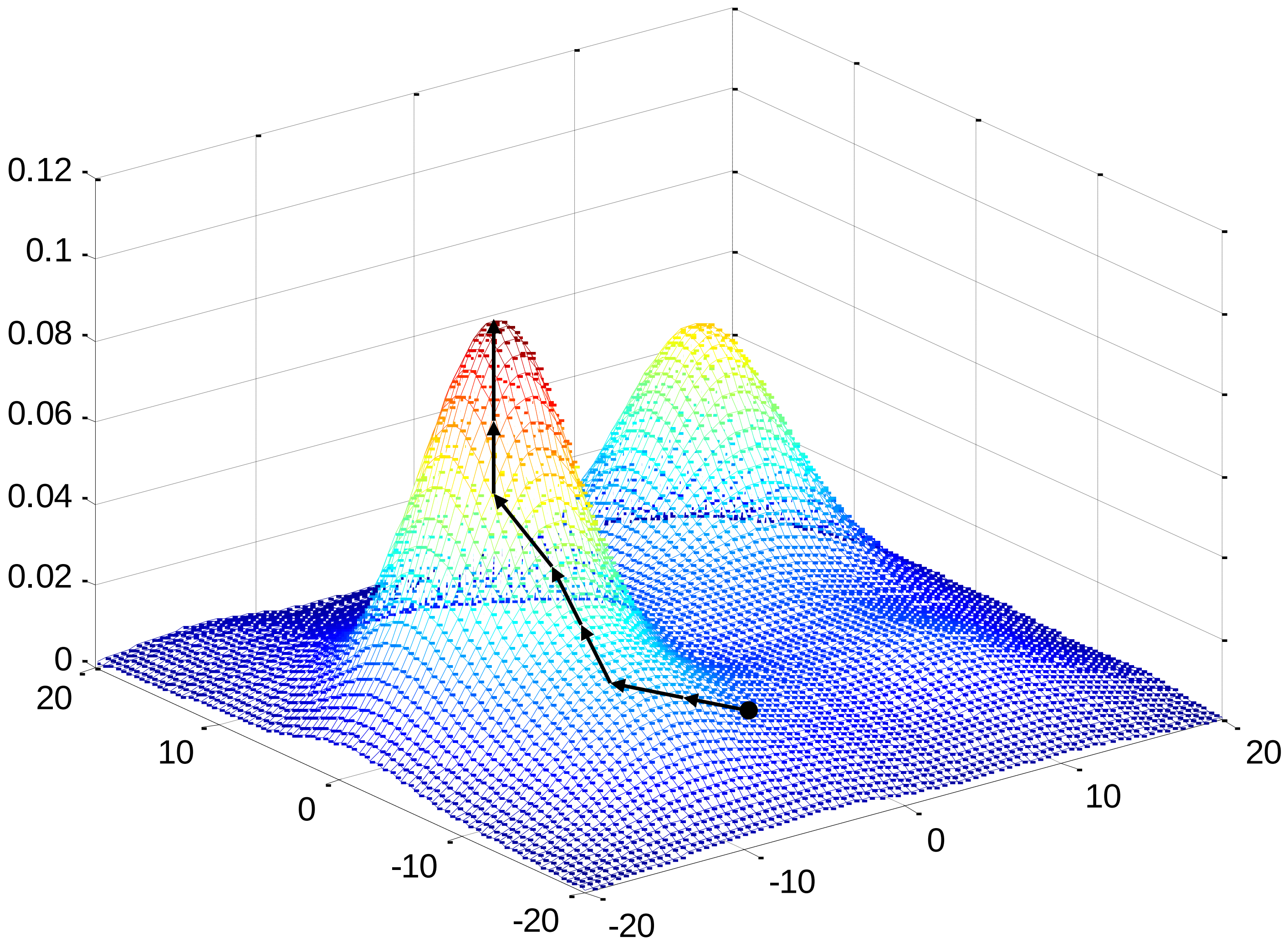


Estimated PDF: $f(X) = \frac{c}{N}\sum_{i=1}^{N} k\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$



The trajectory of locations for finding modes

$$f(X) = \frac{c}{N}\sum_{i=1}^{N} k\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$



The Reality

- This is all much simpler than it looks!!
- For Epanechnikov:
  - *k*(*t*) = (1-*t*) if |t| < 1, 0 otherwise
  - *g*(*t*) = 1if |t| < 1, 0 otherwise
- So, the "mean" part of *M*(*X*) is:

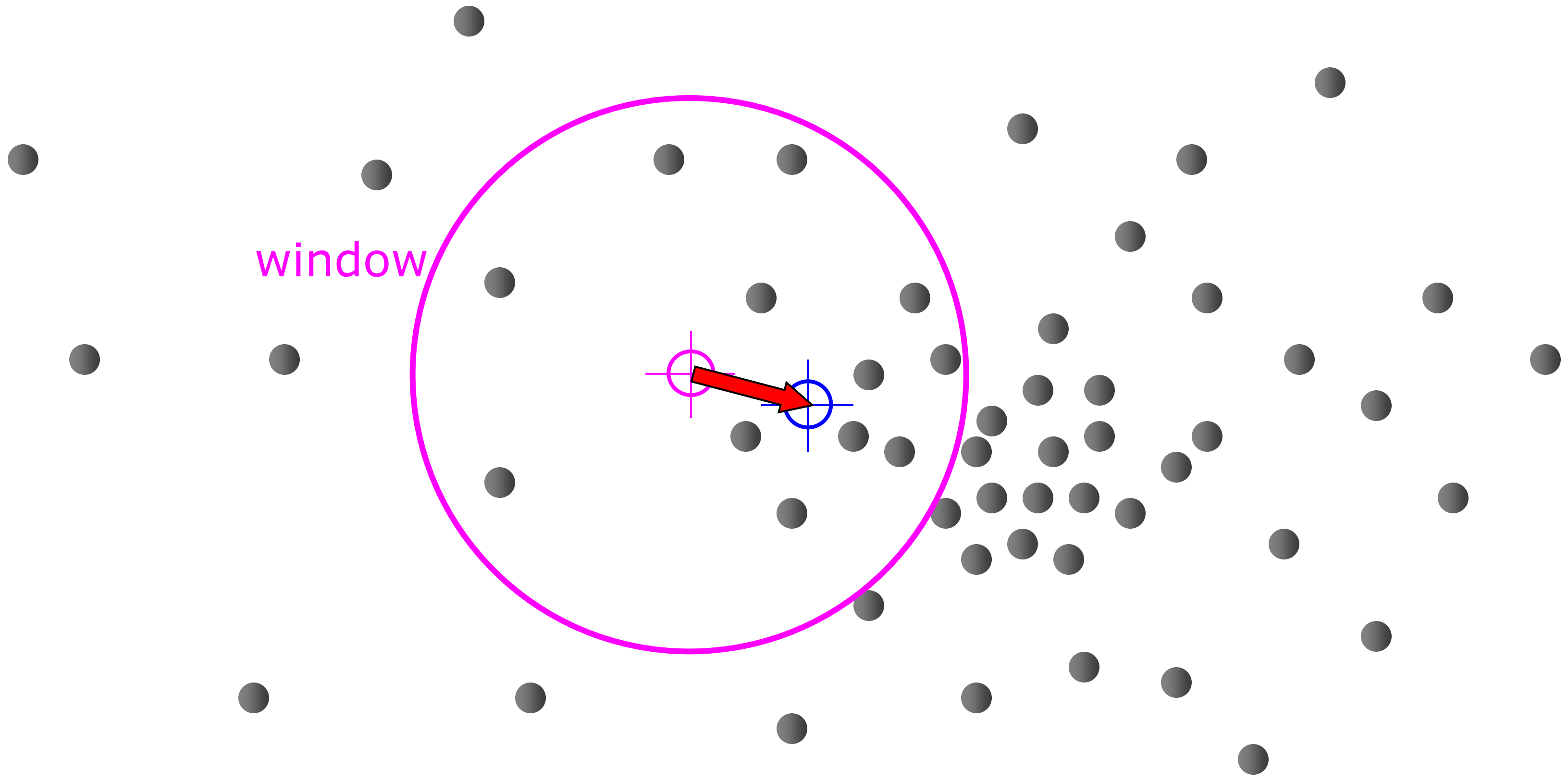
$$\frac{\sum_i X_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)}{\sum_i g\left(\frac{\|X - X_i\|^2}{h^2}\right)} = \frac{\sum_{\|X-X_i\|<h} X_i}{\sum_{\|X-X_i\|<h} 1} = \frac{\sum_{\|X-X_i\|<h} X_i}{N_h(X)}$$

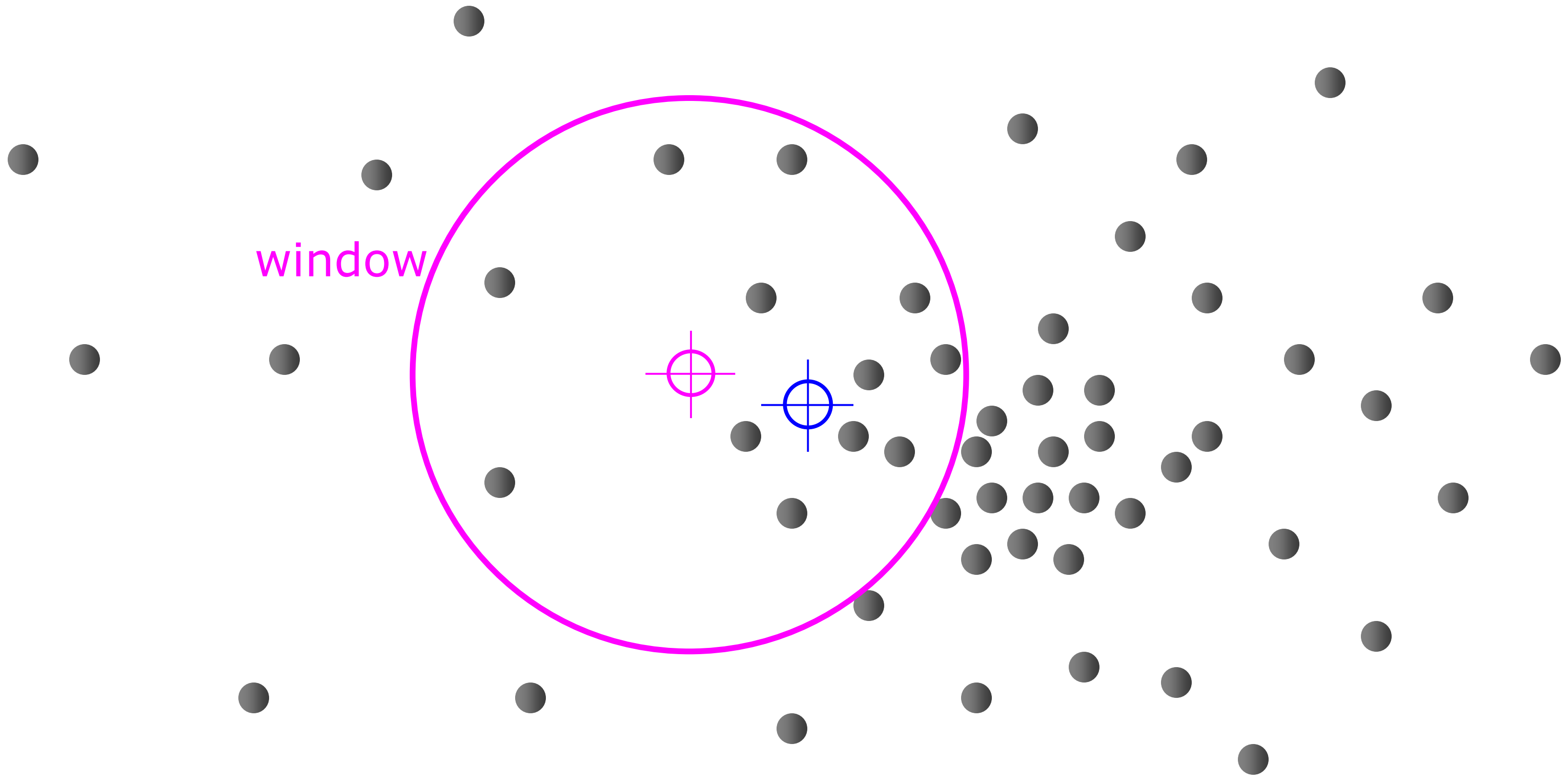
The Reality

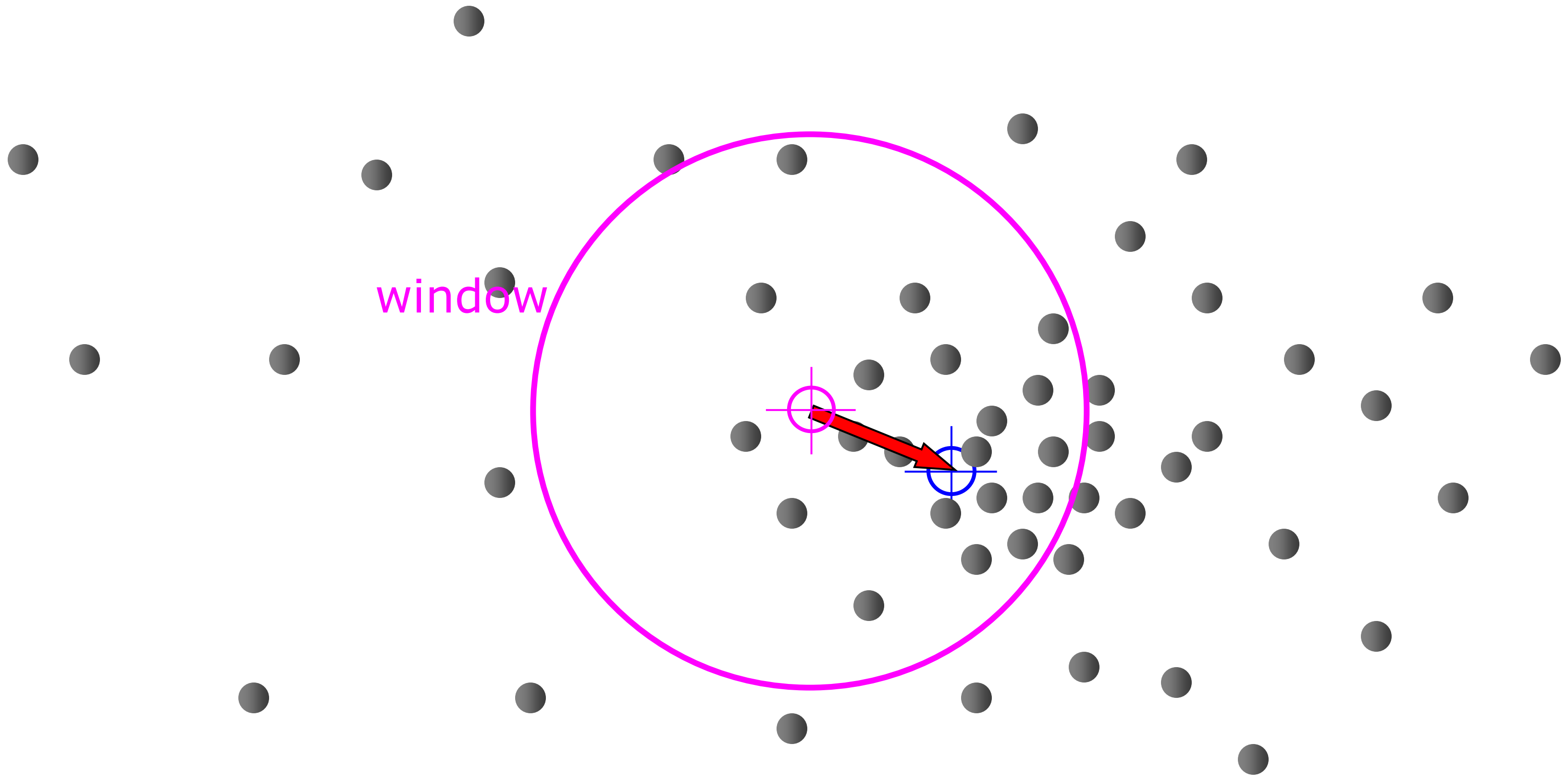
This is *simply* the *average* of the data points within a radius *h* of *X*!!!

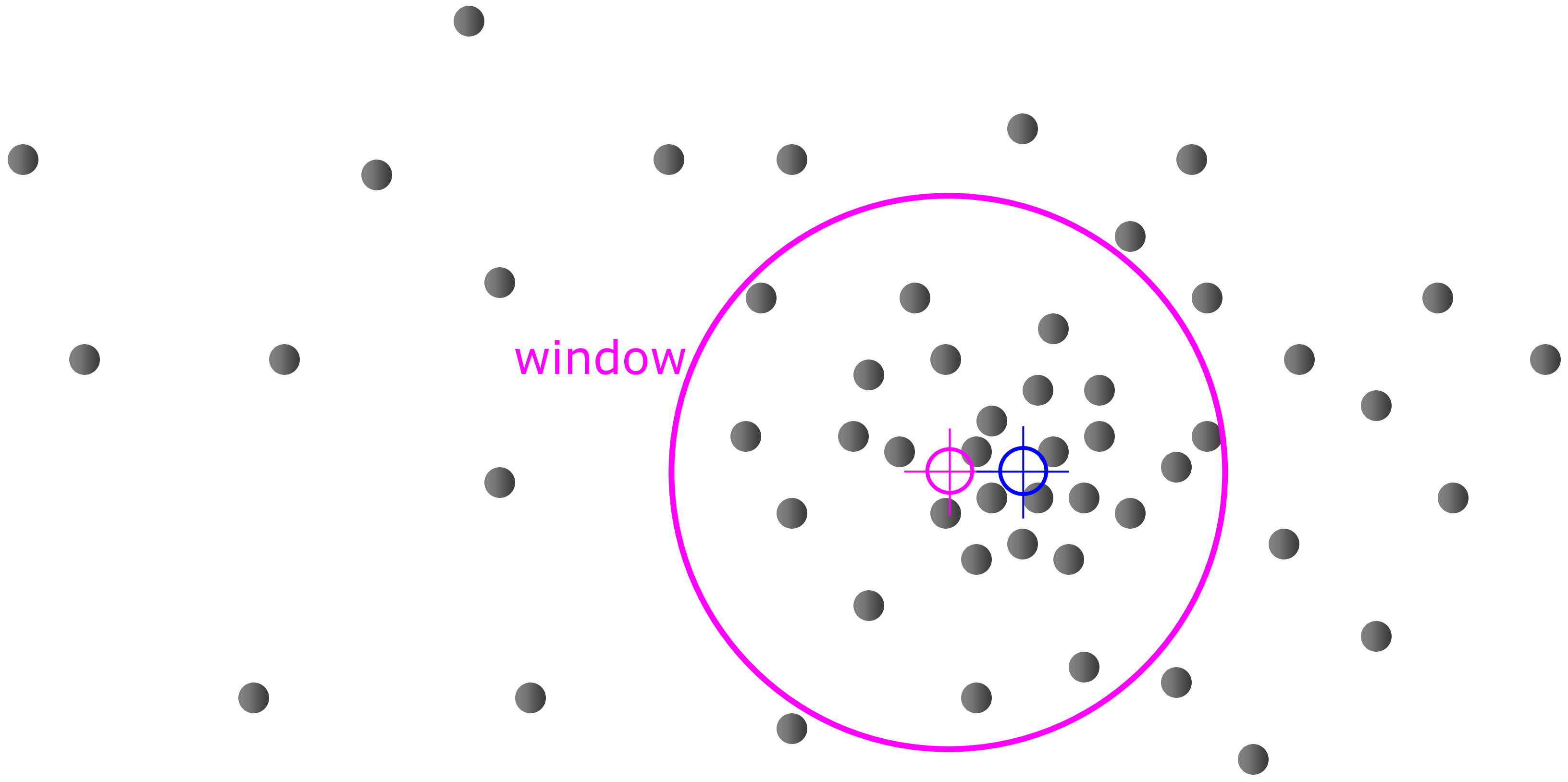
$$\frac{\sum_i X_i g\left(\frac{\left\| X - X_i \right\|^2}{h^2}\right)}{\sum_i g\left(\frac{\left\| X - X_i \right\|^2}{h^2}\right)} = \boxed{\frac{\sum_{\left\| X - X_i \right\| < h} X_i}{N_h(X)}}$$

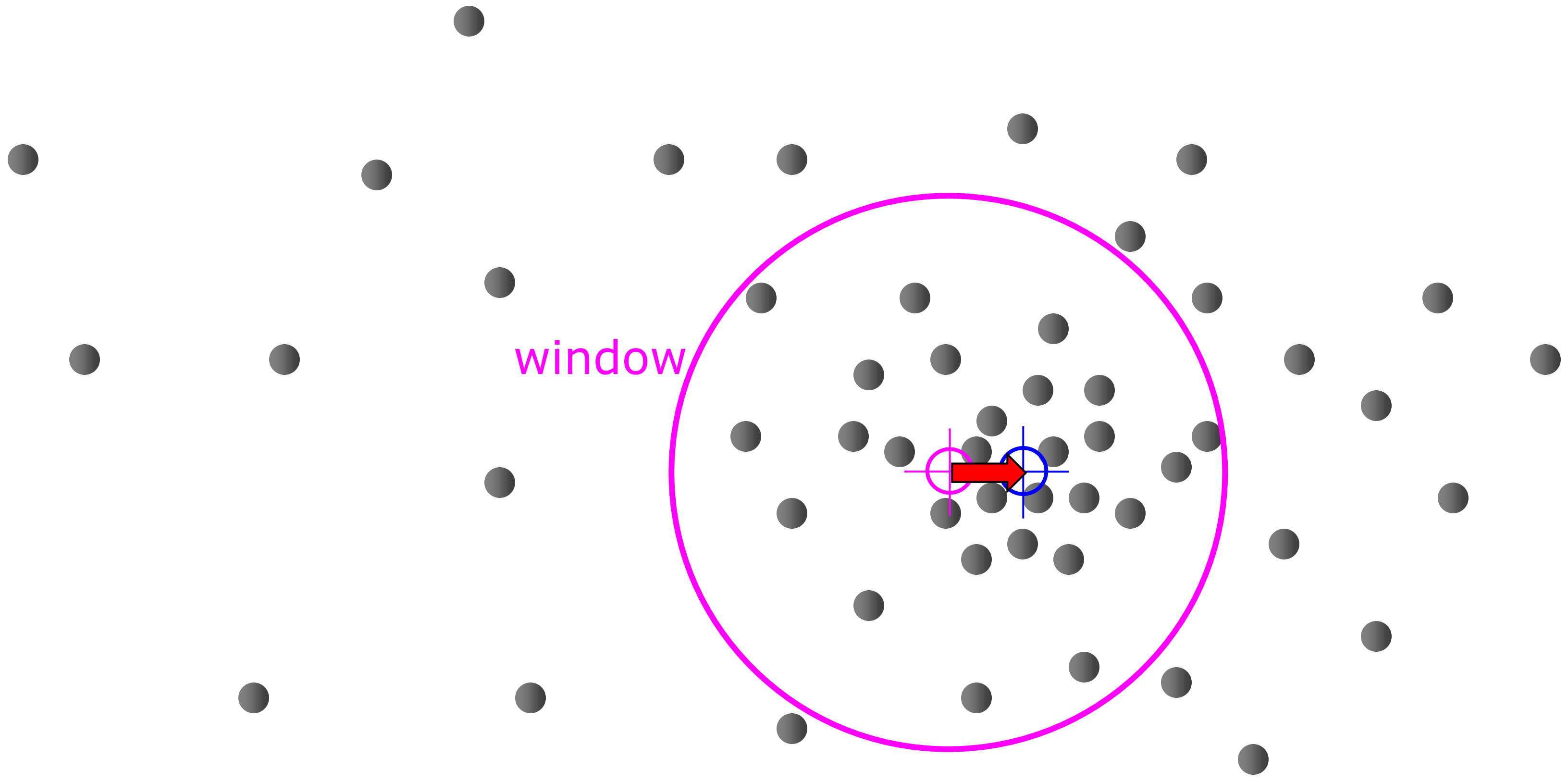
Number of data points within a radius *h* of *X*

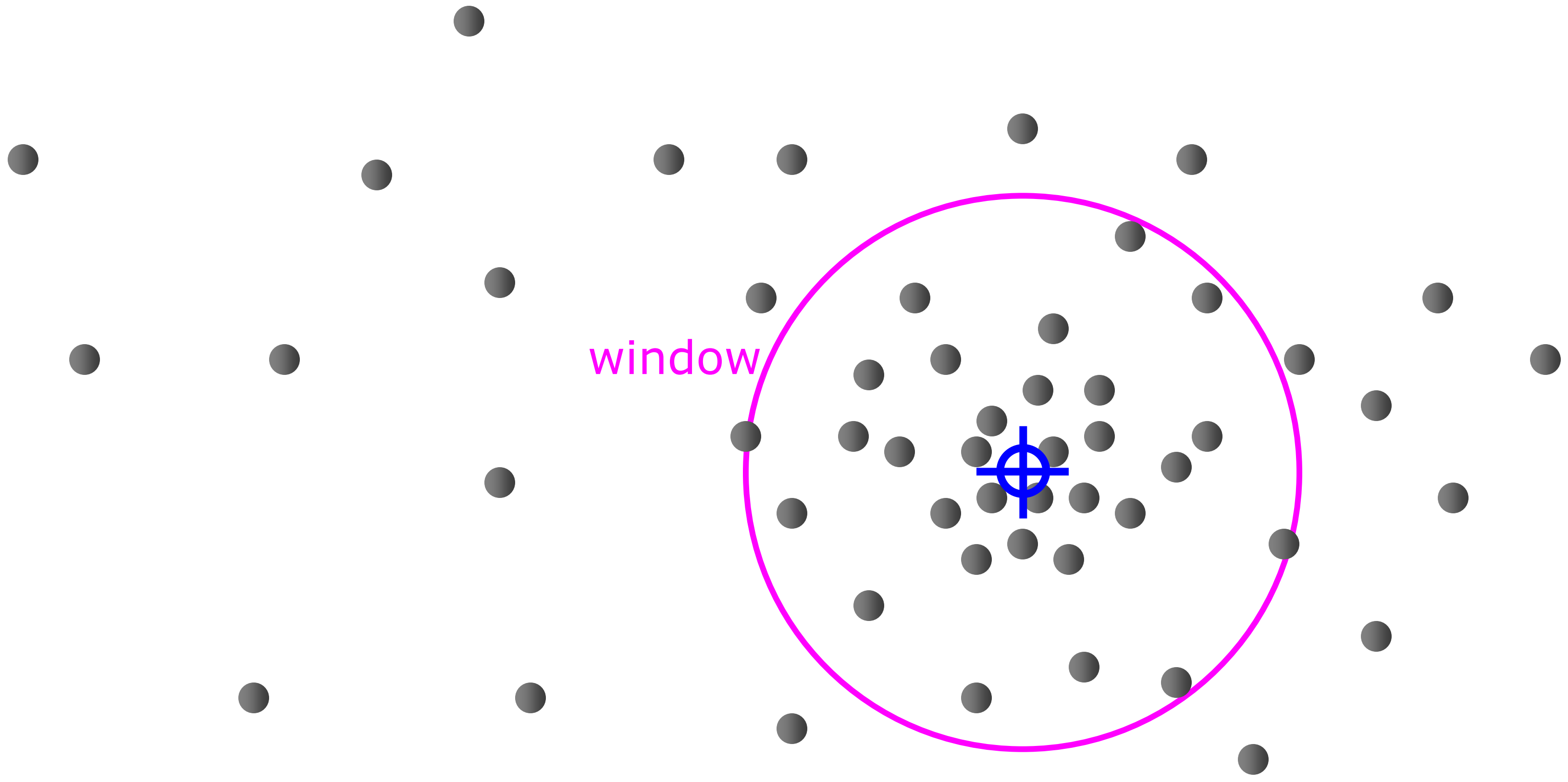

window


window

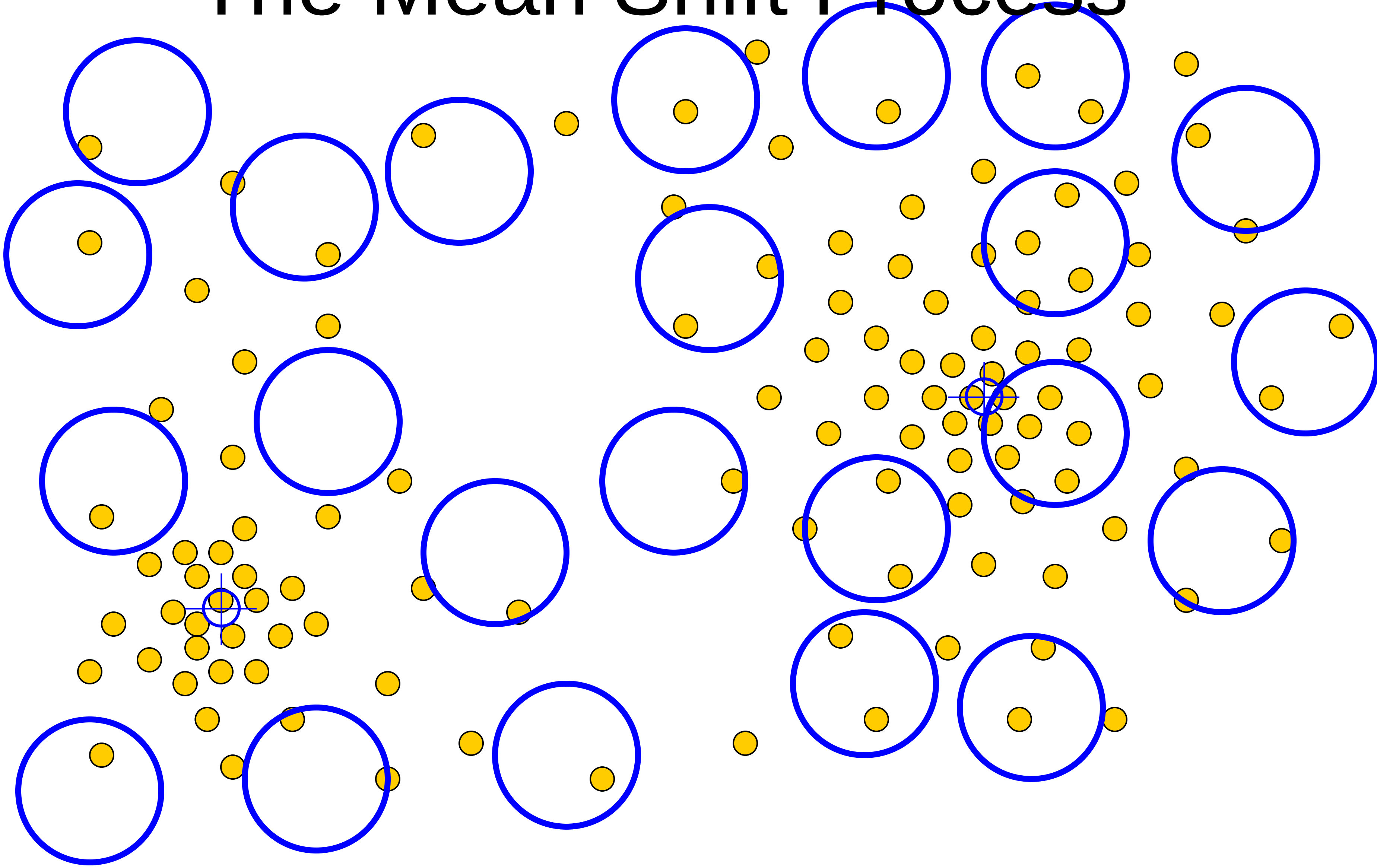

window


window


window


window

The Mean Shift Process

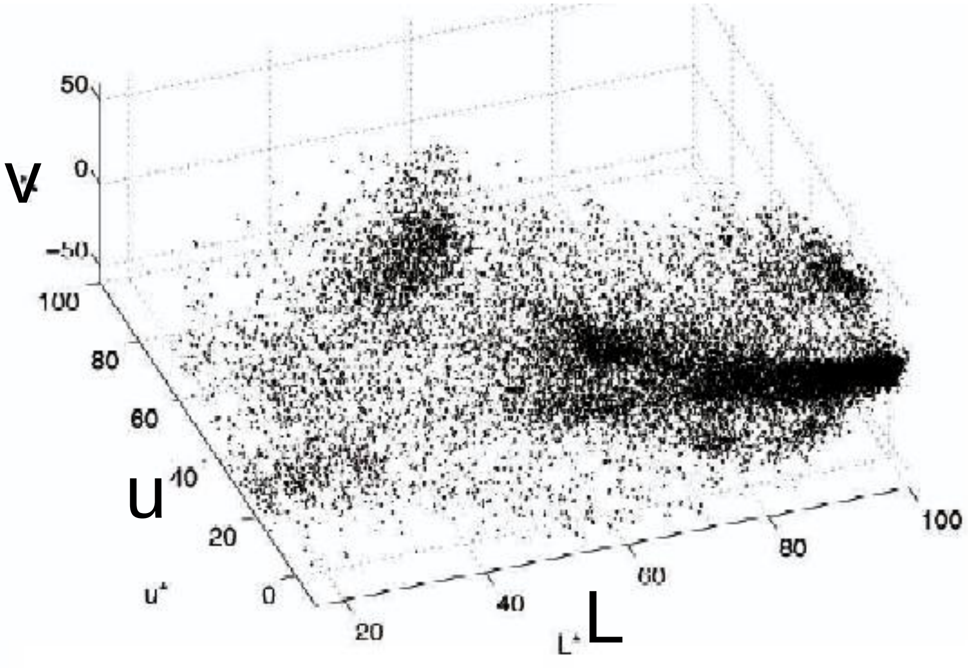


Example: Color Segmentation

- Feature space: (L,u,v,x,y) → Intensity + (u,v) color channels + Position in image (x,y)
- Apply meanshift in the 5-dimensional space
- For each pixel (x_i,y_i) of intensity L_i and color (u_i,v_i), find the corresponding mode c_k
- All of the pixel (x_i,y_i) corresponding to the same mode c_k are grouped into a single region

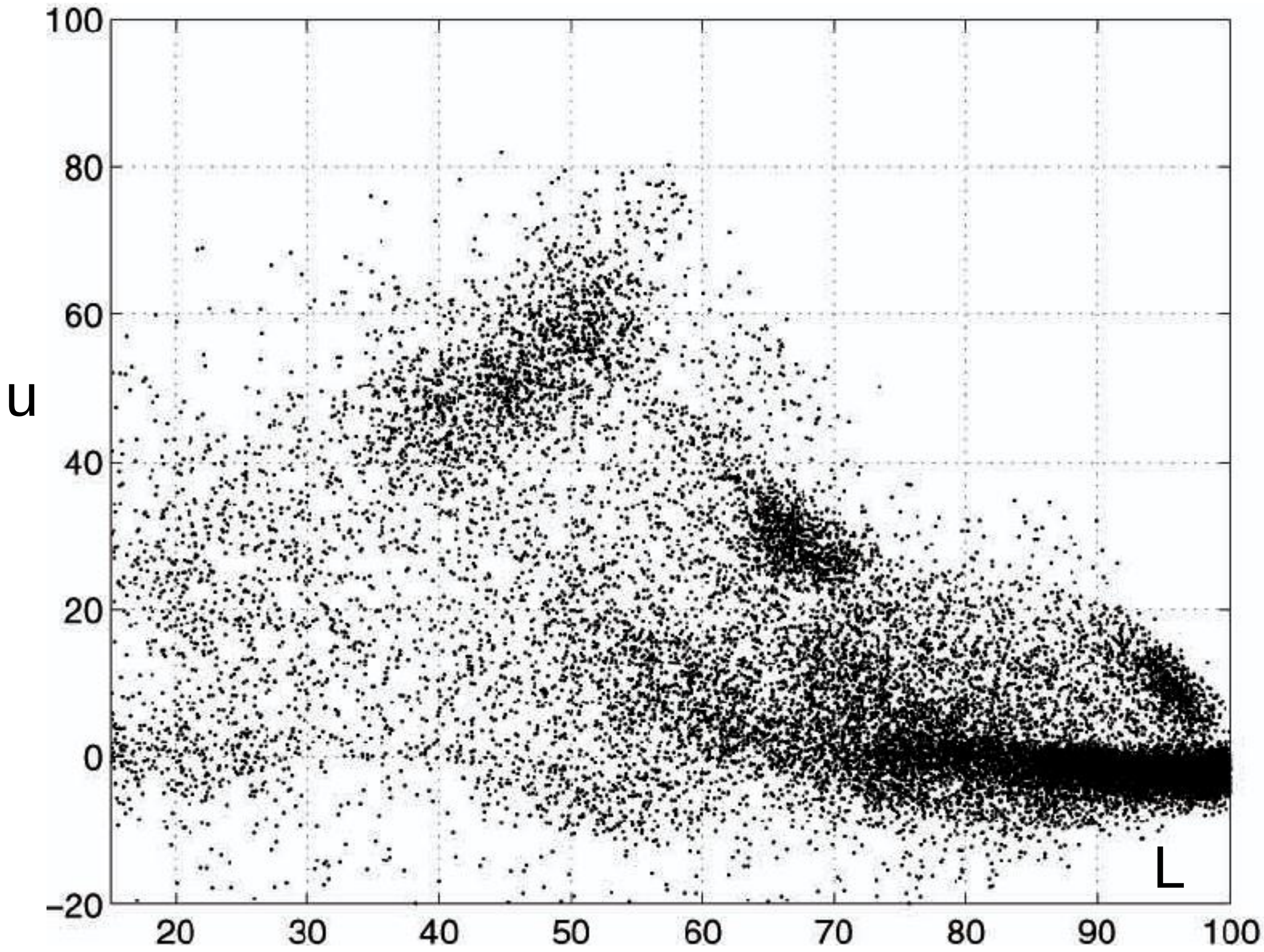
Example: Color Segmentation

Input Image



Luv Space ()

Example from D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis”.



110,400 data points.

$$K_{h_{pos}h_{col}}(X) = c\,k\left(\frac{\left\|X_{pos}\right\|^2}{h_{pos}^{\;2}}\right)k\left(\frac{\left\|X_{col}\right\|^2}{h_{col}^{\;2}}\right)$$

Kernel on position (*x*,*y*)

Kernel on color (*L*,*u*,*v*)

- *Good news:* We don’t need to know the number of regions (modes, clusters).
- *Bad news:* We need to choose the bandwidths h_{pos} and h_{col}

The Mean Shift Process

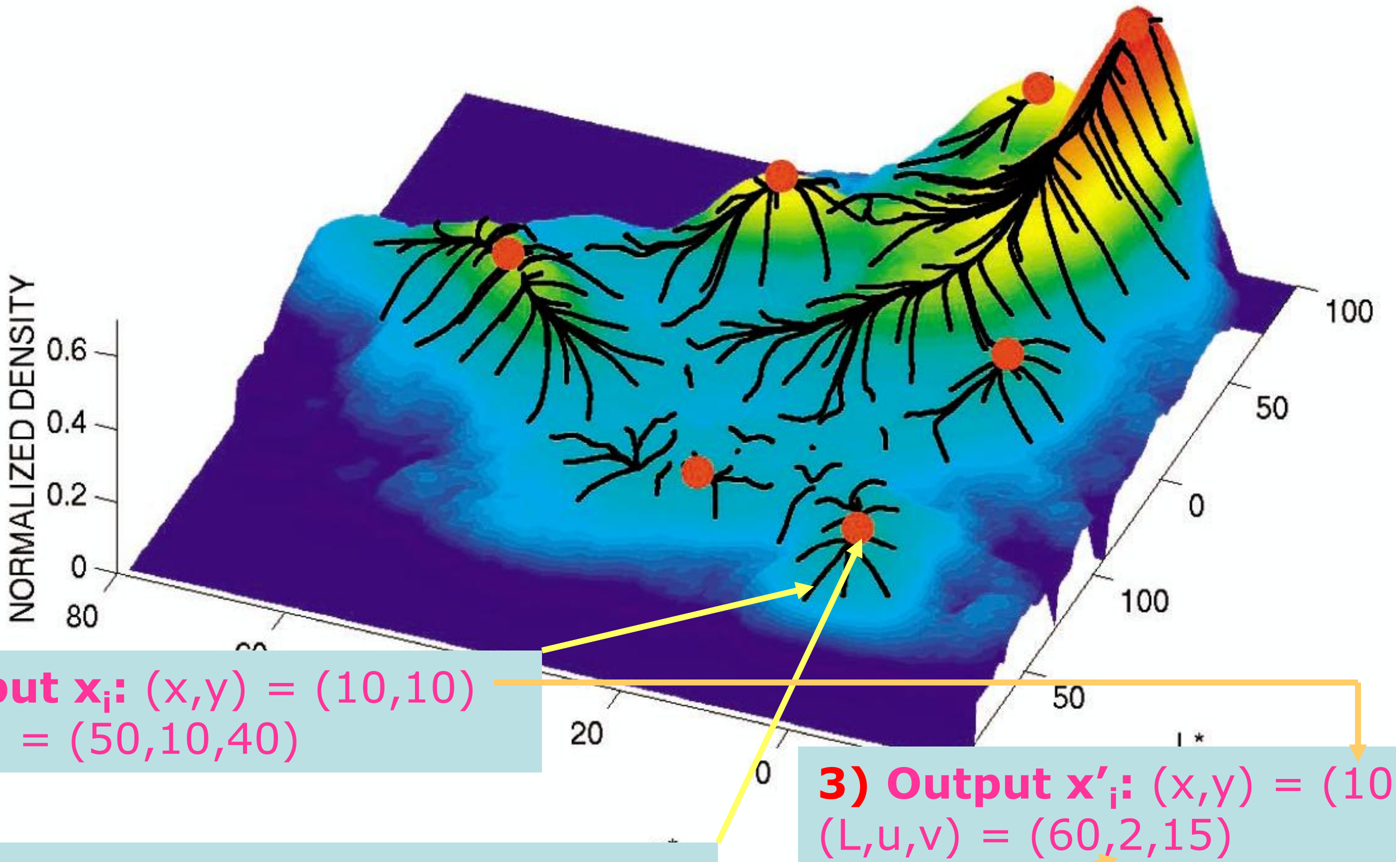
Notes:

• If we do not apply the last step, we get “smoothing”
→ Replacing each color by the closest mode

• The “color” part of the feature can be replaced by other things like texture (bank of filter outputs) or other values (multispectral). The only change is to increase the dimension *p* of the feature space

• The fundamental operation to compute the kernels is to find the neighbors within some radius (defined by *h*). This can be very expensive in high dimension with lots of points → Need smart “nearest-neighbor” data structures.

Example: Color



1) Input x_i: (x,y) = (10,10)
(L,u,v) = (50,10,40)

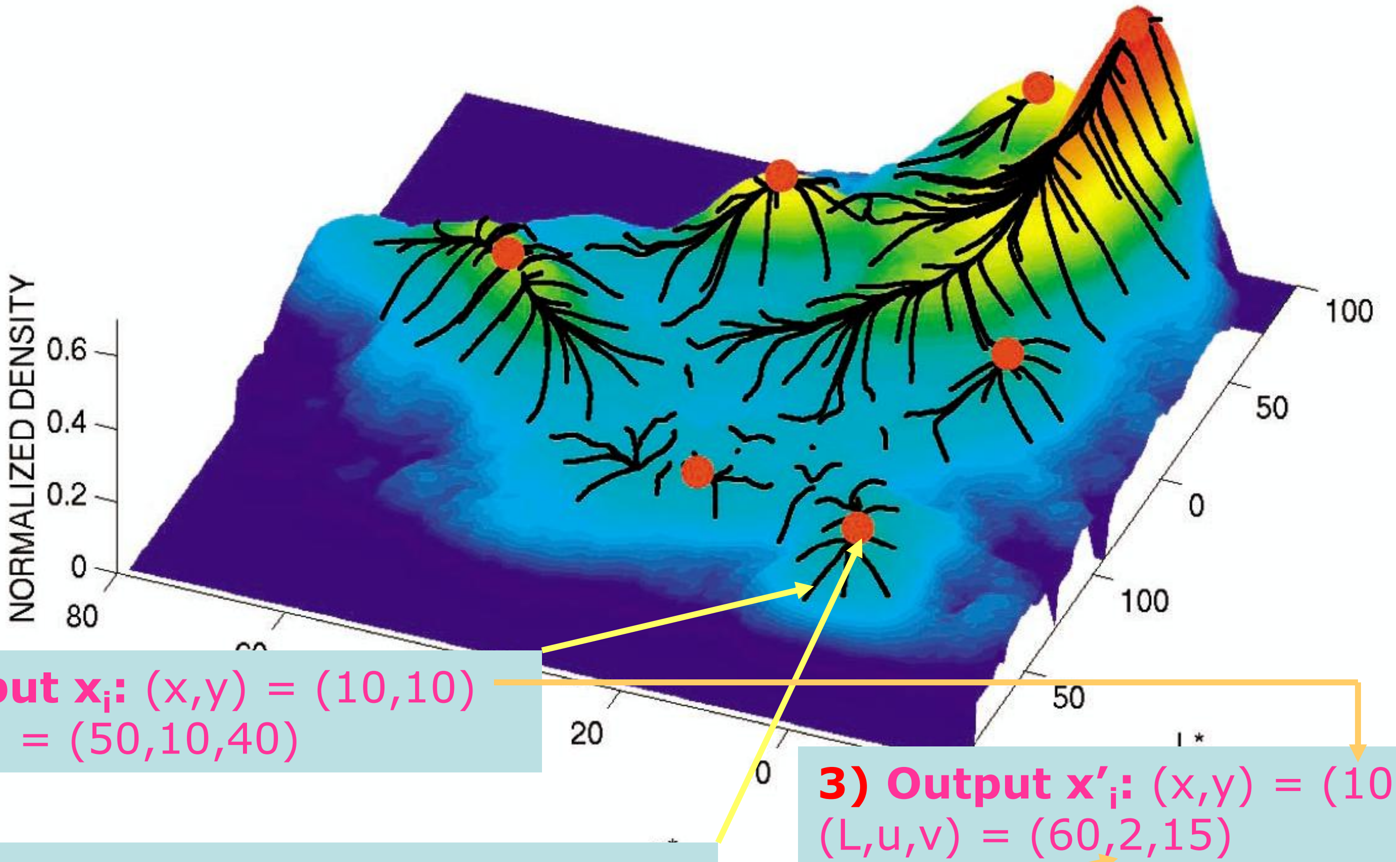
2) Apply mean shift till converged

c_i: (x,y) = (15,20) (L,u,v) = (60,2,15)

3) Output x'_i: (x,y) = (10,10)
(L,u,v) = (60,2,15)

D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis".

Example: Color



1) Input x_i: (x,y) = (10,10)
(L,u,v) = (50,10,40)

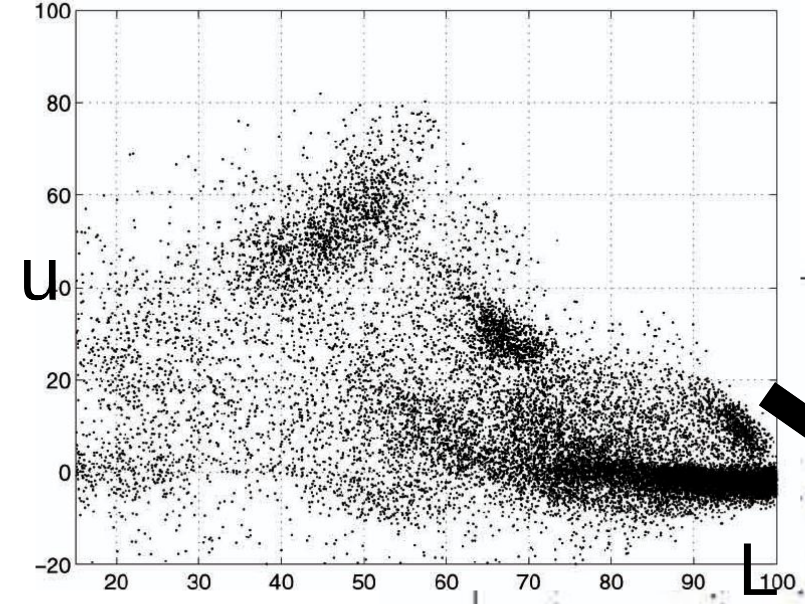
3) Output x'_i: (x,y) = (10,10)
(L,u,v) = (60,2,15)

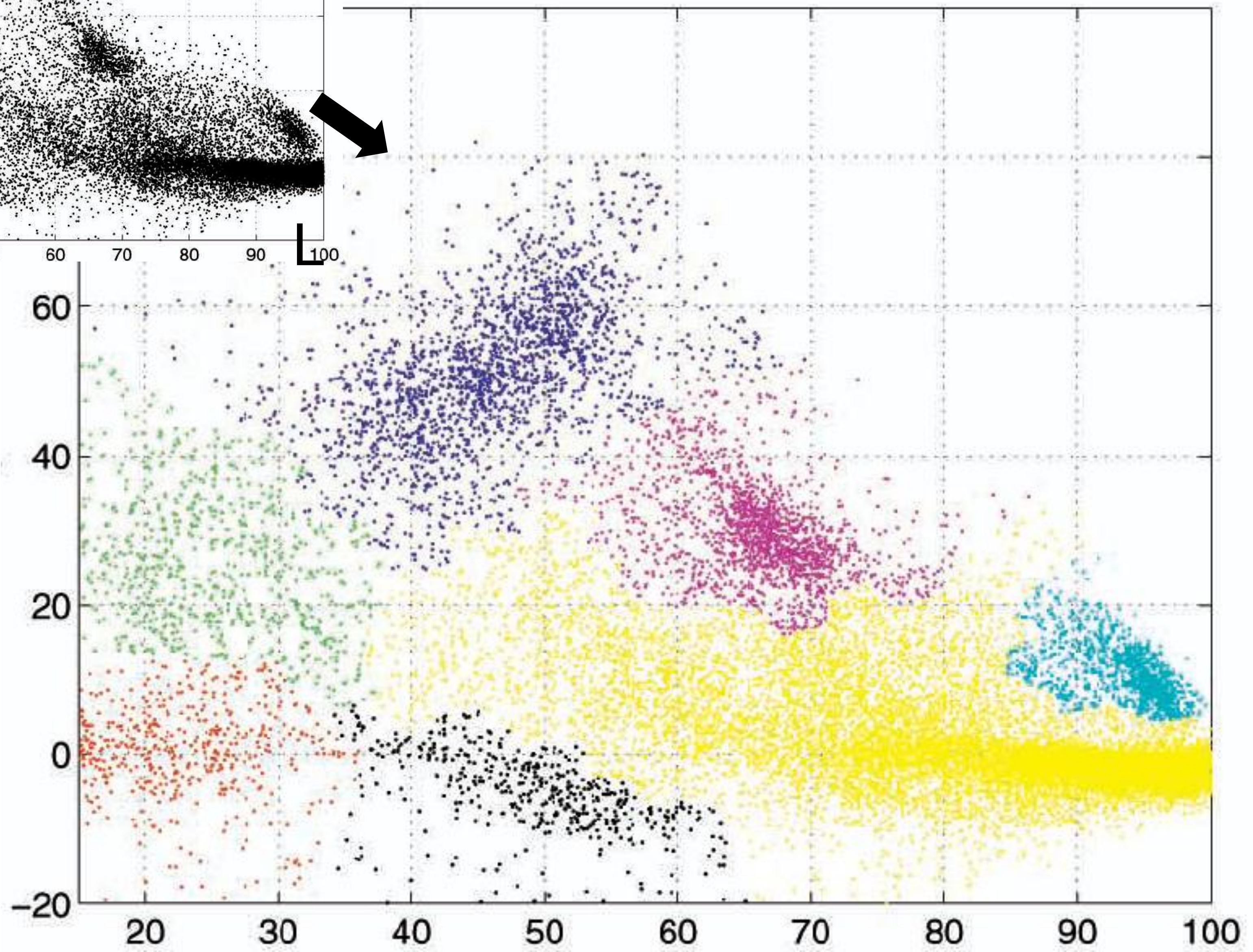
2) Apply mean shift till converged

c_i: (x,y) = (15,20) (L,u,v) = (60,2,15)

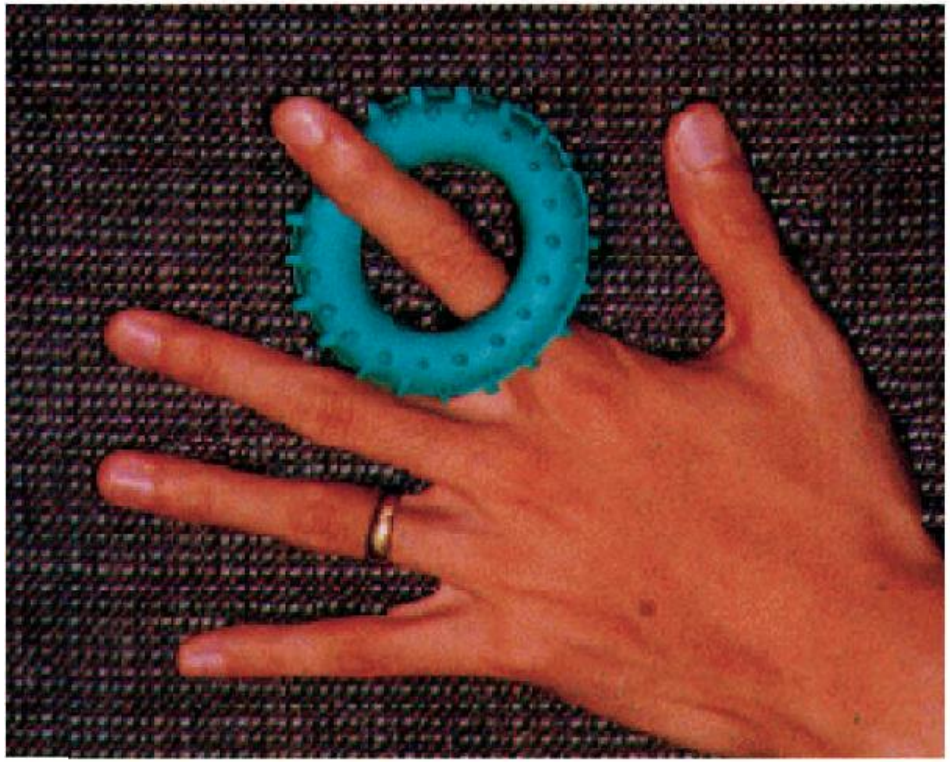
Note: In practice, all points may not converge to the same mode -→ Need an additional (easy) clustering step to group the converged locations to the location

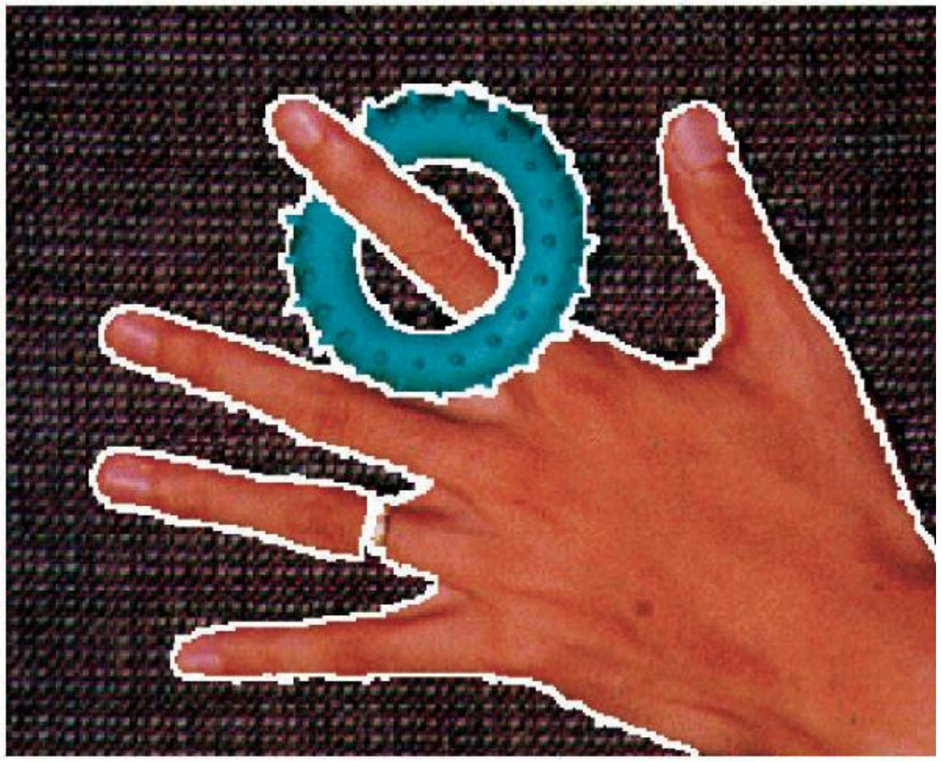
Clustering Result



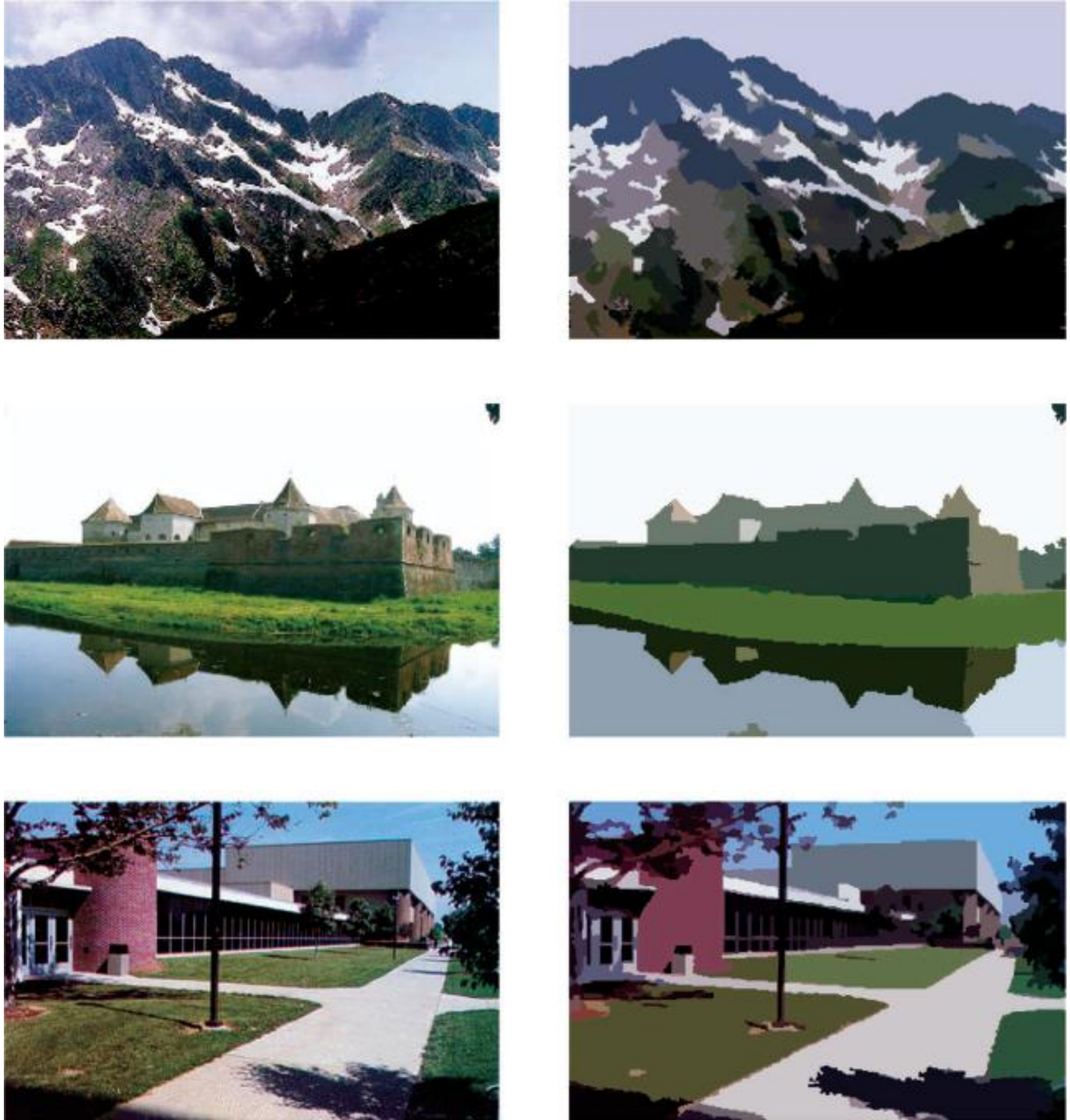


Experimental Results



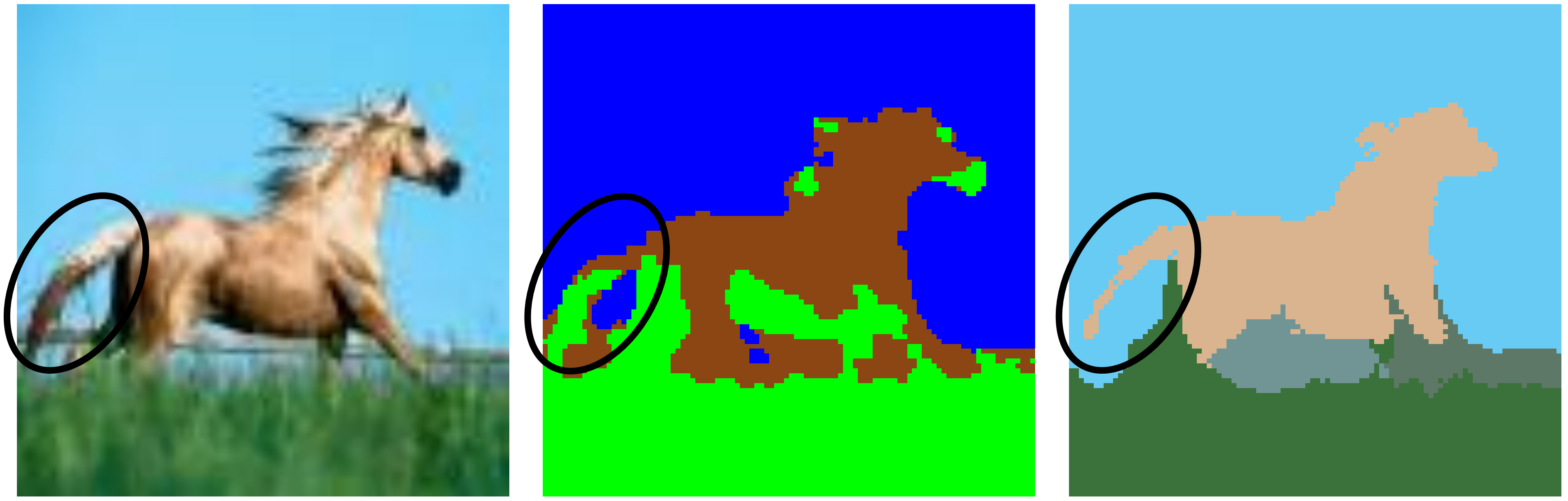


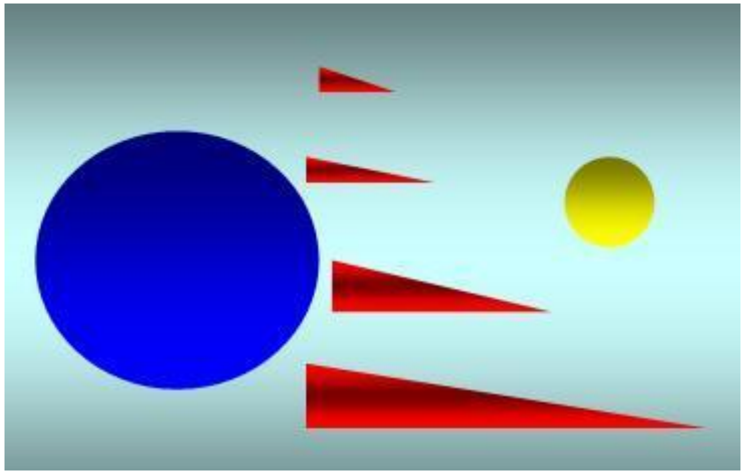
Experimental results

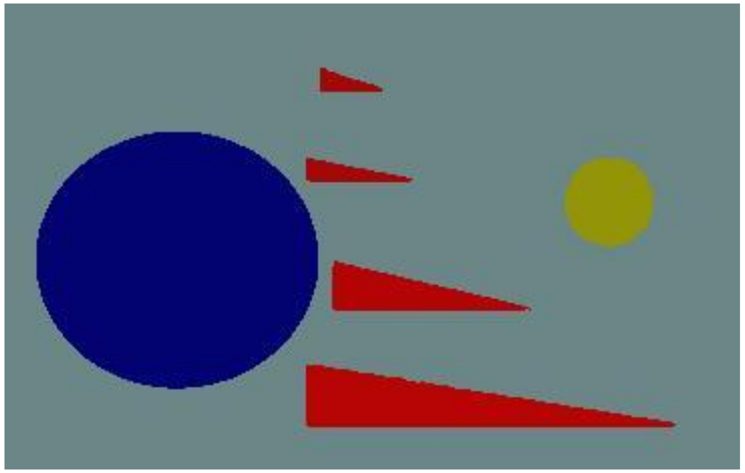


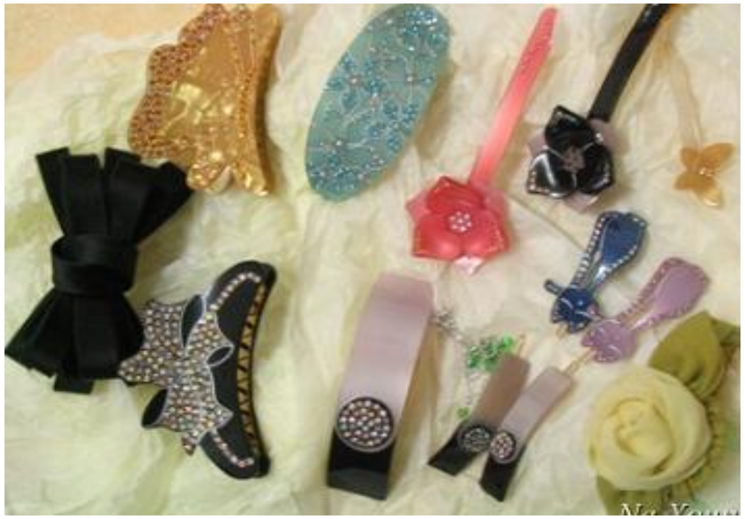
Results - Comparing to EM

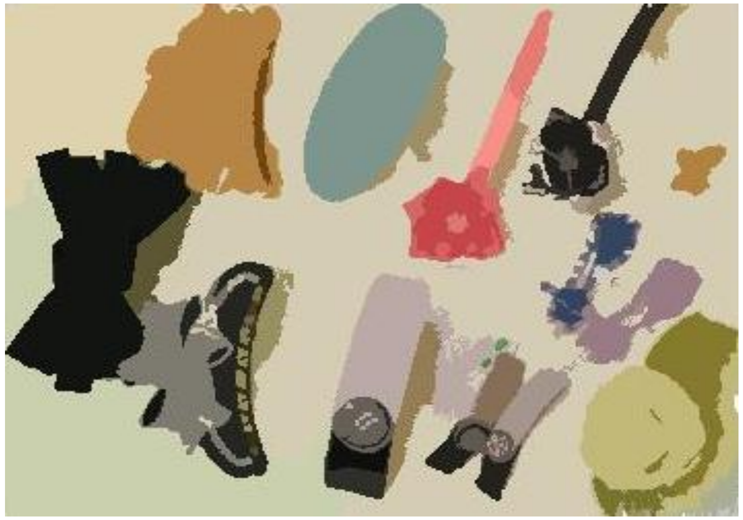
- Easy example – horse from HW6
  - Original
  - EM with 3 clusters and 5 equally weighted features RGB and XY
  - Mean shift $(h_{pos},h_{col}) = (12,16)$




Original image


Mean shift $(h_s,h_r,M) = (4,50,100)$


Original image


Mean shift $(h_s, h_r, M) = (10,10,10)$

Beyond segmentation: Mean shift tracking

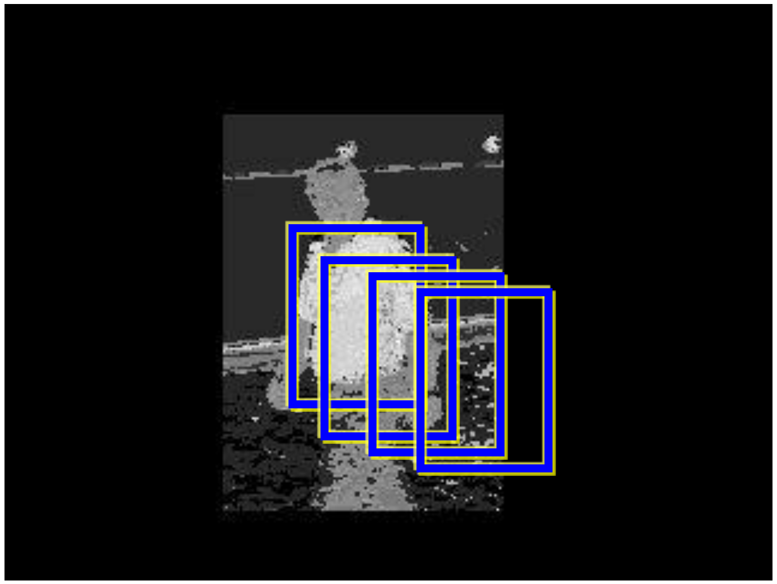
Weight images: Create a response map with pixels weighted by "likelihood" that they belong to the object being tracked.

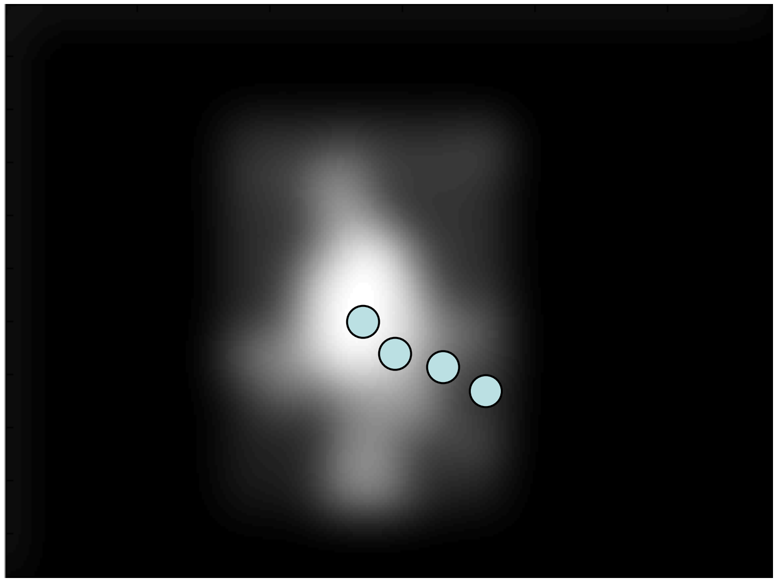
Histogram comparison: Weight image is implicitly defined by a similarity measure (e.g. Bhattacharyya coefficient) comparing the model distribution with a histogram computed inside the current estimated bounding box.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis Machine Intelligence*, 25(5):564–577, May 2003.

Mean-Shift on Weight Images

The pixels form a uniform grid of data points, each with a weight (pixel value). Perform standard mean-shift algorithm using this weighted set of points.





Example from Bob Collins, PSU

Beyond segmentation: Mean shift tracking

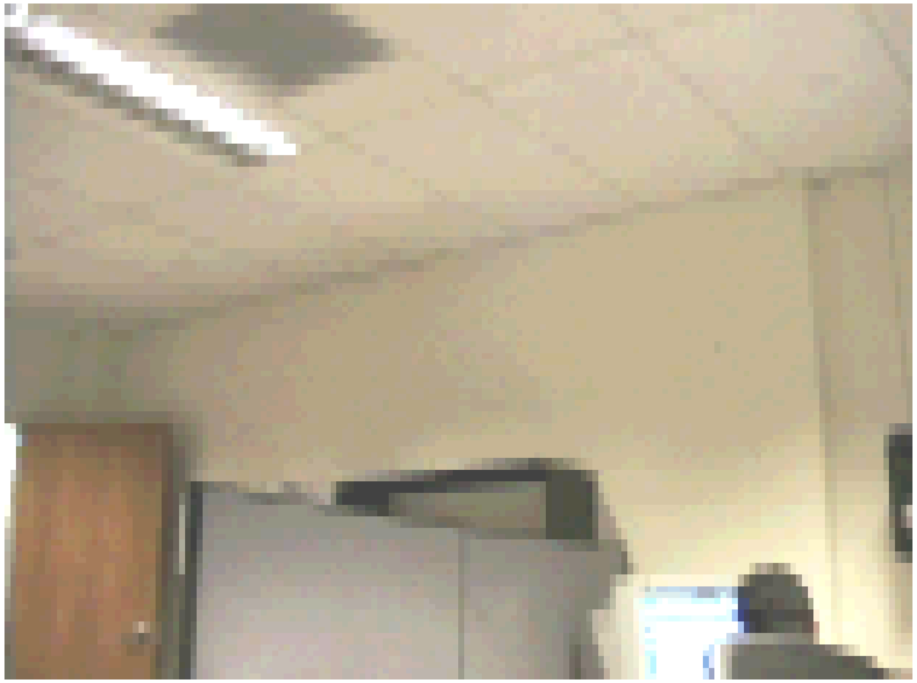
Weight images: Create a response map with pixels weighted by “likelihood” that they belong to the object being tracked.

Histogram comparison: Weight image is implicitly defined by a similarity measure (e.g. Bhattacharyya coefficient) comparing the model distribution with a histogram computed inside the current estimated bounding box.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis Machine Intelligence*, 25(5):564–577, May 2003.

Mean-Shift Tracking

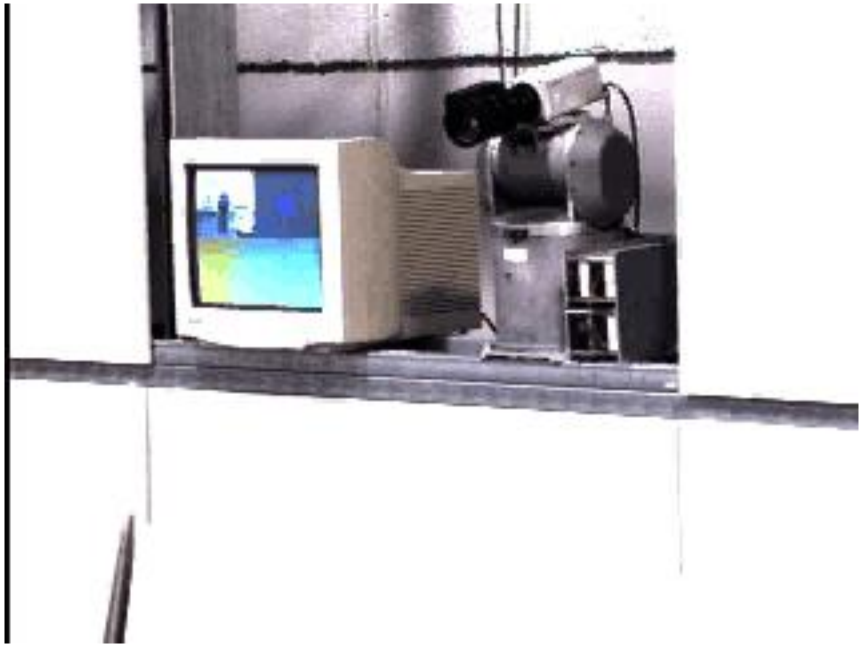
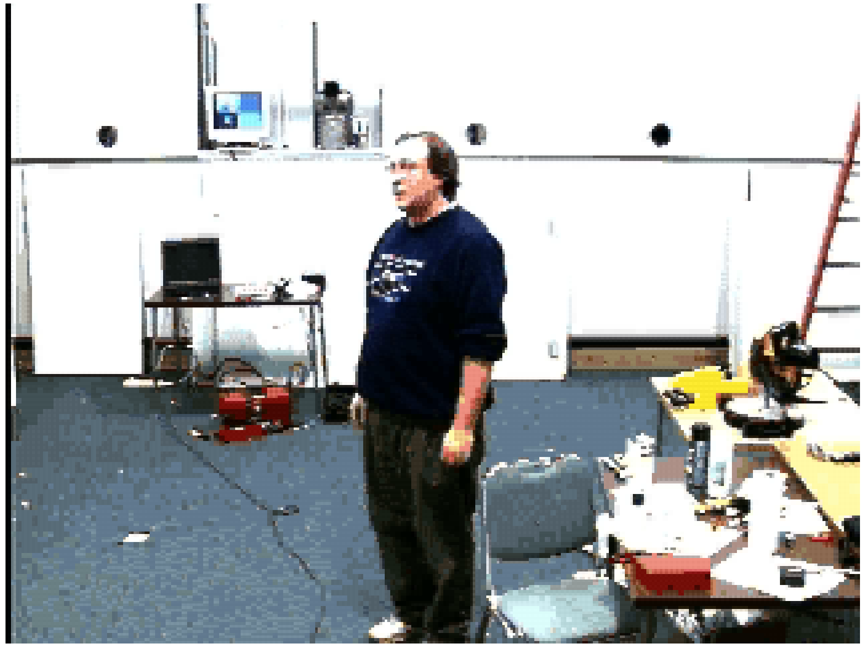
Gary Bradski, CAMSHIFT



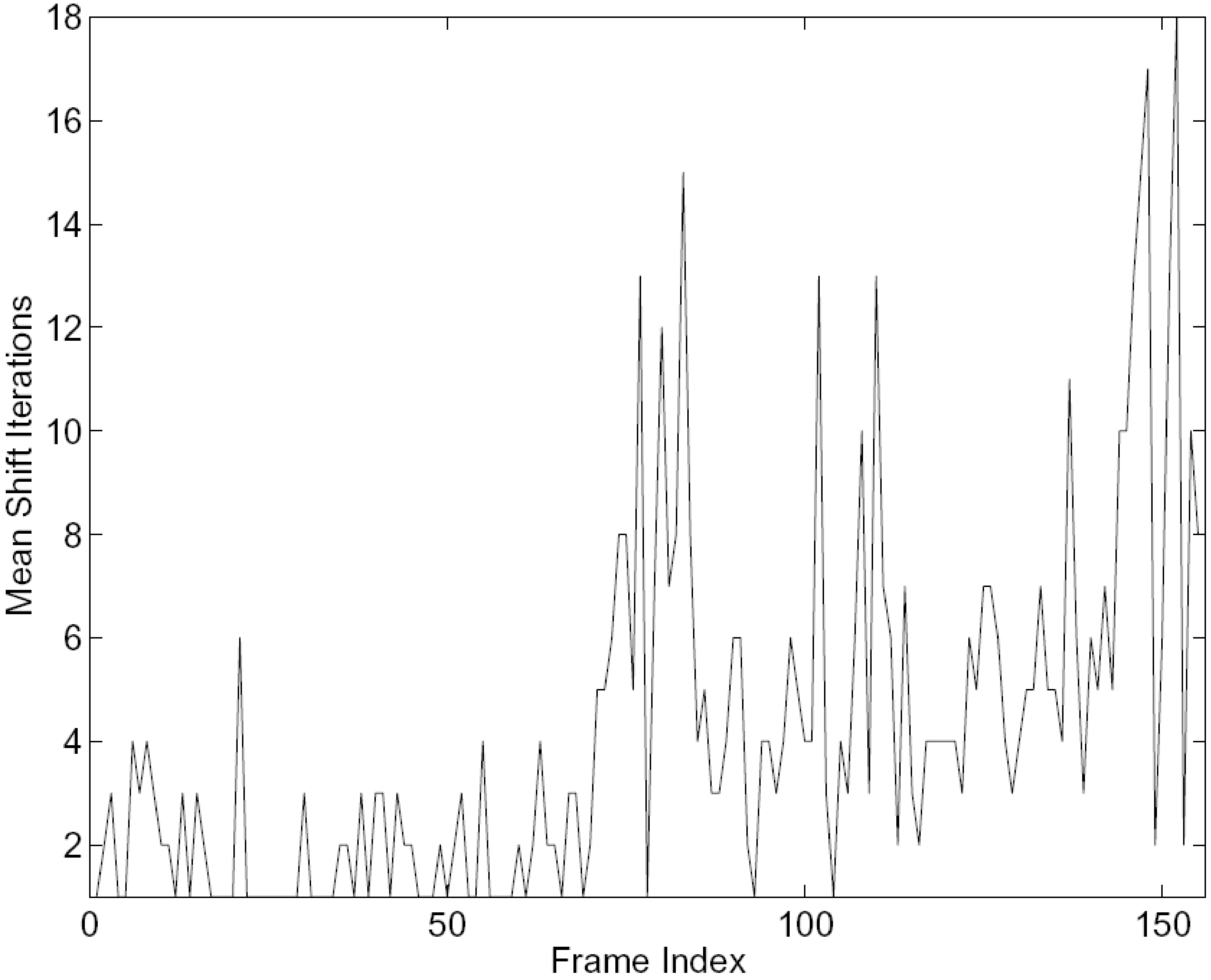
Comaniciu, Ramesh and Meer, CVPR 2000
(Best paper award)

Mean-Shift Tracking

Using mean-shift in real-time to control a pan/tilt camera.

Collins, Amidi and Kanade, An Active Camera System for Acquiring Multi-View Video, ICIP 2002.


Mean Shift Iterations
Frame Index
0
50
100
150
2
4
6
8
10
12
14
16
18

Notes

- You should read:

D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis". IEEE Trans. PAMI, Vol. 24, No. 5, 2002.

D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. Pattern Analysis Machine Intelligence*, 25(5):564–577, May 2003.

- Warning: The notations vary in different papers, in particular the constant *c* may be made explicit.
- The approach is attractive because 1) simple implementation 2) non-parametric, assumes no model of the clusters, including number of clusters.
- The mean shift approach can be used for tracking (using histograms of color distributions) $\rightarrow$ one of the most effective approach to tracking because it is non-parametric.
- Can be used with much larger feature spaces $\rightarrow$ For example, adding texture features from filter outputs or other features.
- An additional parameter is normally used to remove small, "noise", regions.
- *Problem:* Choice of bandwidth may be difficult. Extensions include adaptive bandwidth based on local data density
- *Problem:* Retrieval of data points for kernel computation may be expensive. Extensions include use of KD-tree, ANN (Approximate Nearest neighbor) techniques, etc.