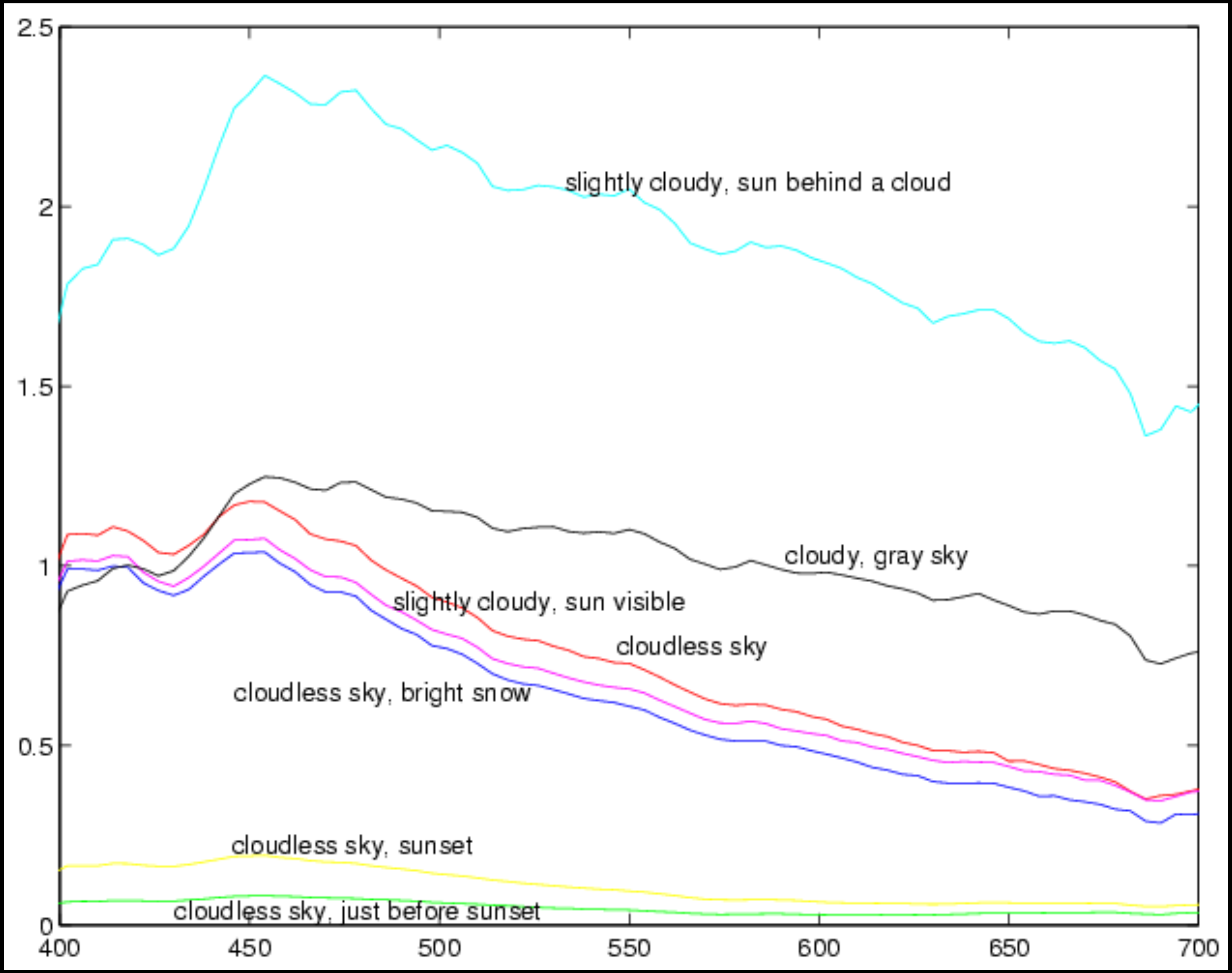
Introduction to Computer Vision

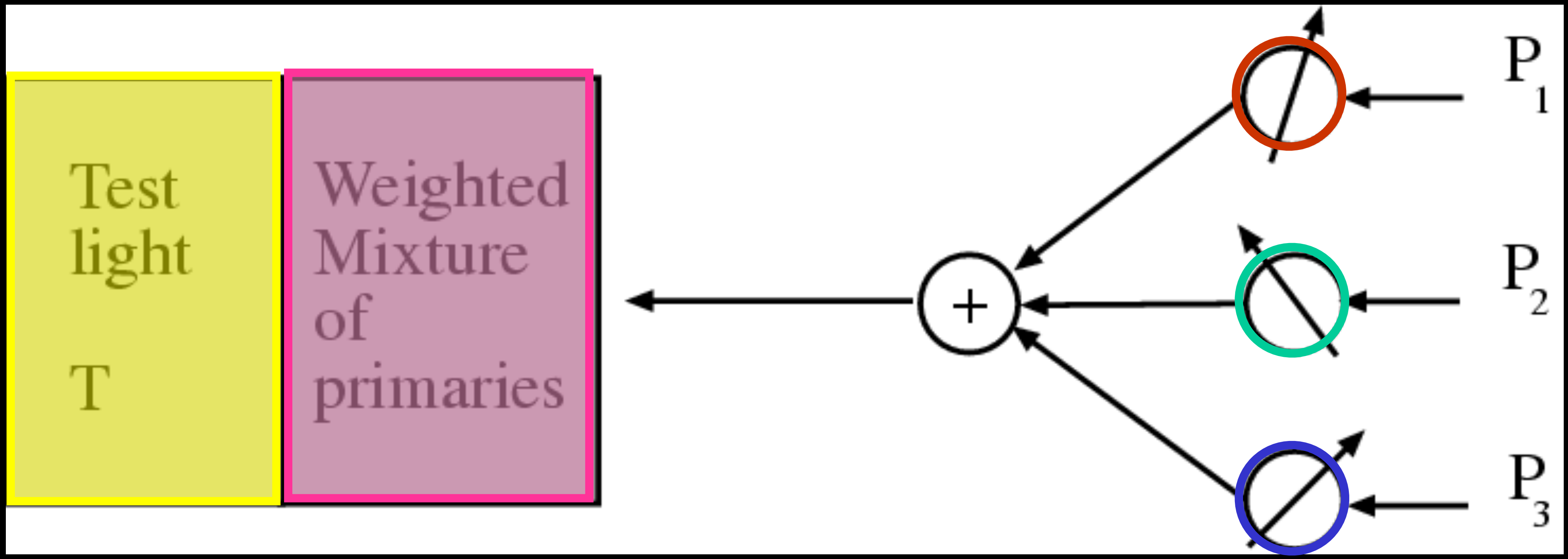
Instructors: Jean Ponce and Matthew Trager
jean.ponce@inria.fr, matthew.trager@cims.nyu.edu

TAs: Jiachen (Jason) Zhu and Sahar Siddiqui
jiachen.zhu@nyu.edu, ss12414@nyu.edu

Spectral energy density



Color Matching Experiments



Adjust the knobs on the primaries until the split field looks uniform.

$$T = w_1P_1+w_2P_2+ \dots +w_kP_k$$

Notation only!

$$T+w_1P_1+ \dots +w_nP_n = w_{n+1}P_{n+1}+ \dots +w_kP_k$$

Subtractive matching

Linearity of Colour Matching (Grassman's Laws)

$$T_a = w_{a1}P_1 + w_{a2}P_2 + w_{a3}P_3 \quad \text{and} \quad T_b = w_{b1}P_1 + w_{b2}P_2 + w_{b3}P_3$$

$$\Downarrow$$

$$\lambda\, T_a + \mu\, T_b = (\lambda\, w_{a1} + \mu\, w_{b1})P_1 + (\lambda\, w_{a2} + \mu\, w_{b2})P_2 + (\lambda\, w_{a3} + \mu\, w_{b3})P_3$$

$$T_a = w_1P_1 + w_2P_2 + w_3P_3$$

and

$$T_b = w_1P_1 + w_2P_2 + w_3P_3$$

$$\Rightarrow \quad T_a = T_b$$

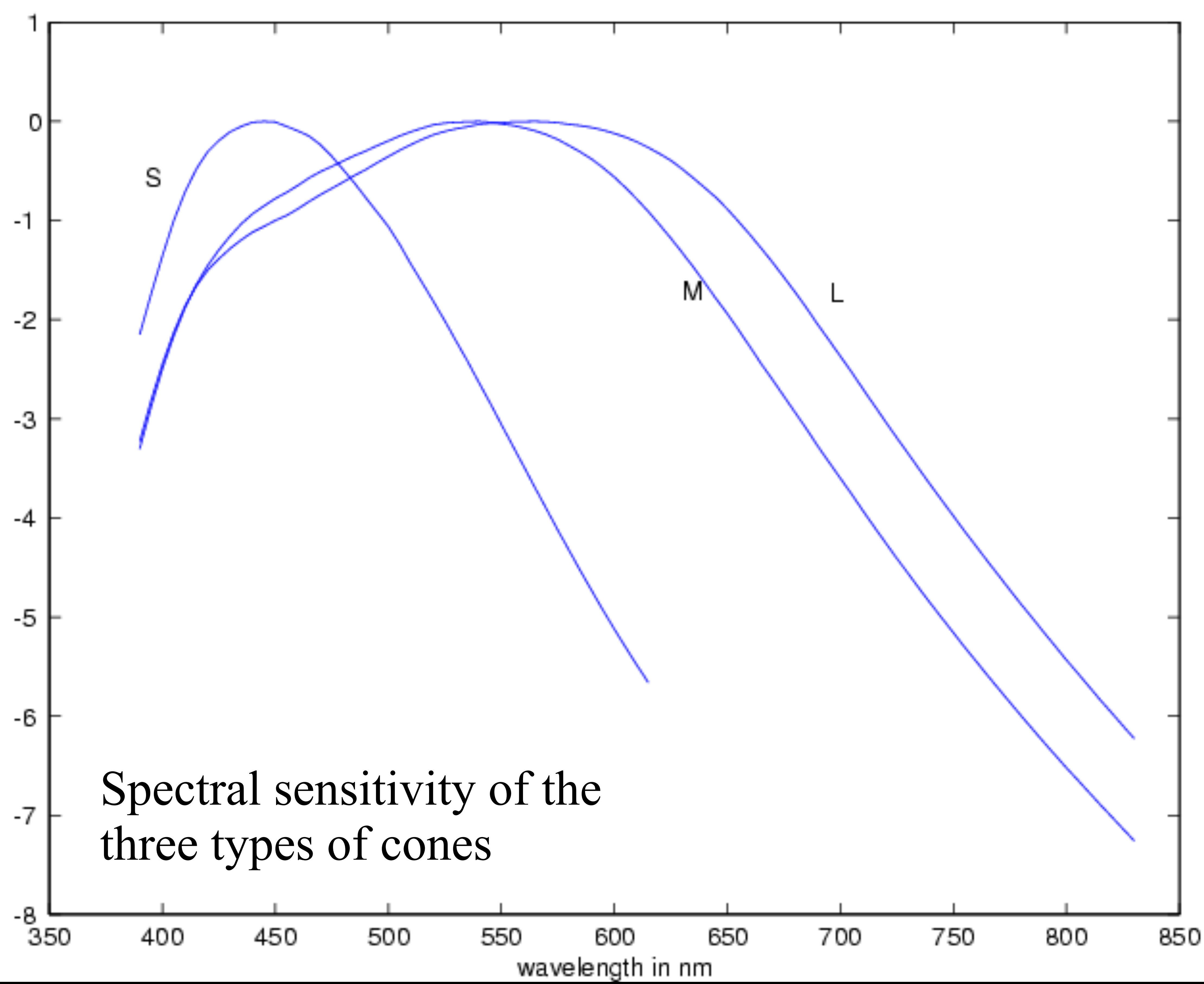
Note: this does not mean that T_a and T_b have the same spectrum.

Principle of Univariance

- There are (usually) three types of photoreceptors in the human eye. They do not directly measure spectral radiance.
- The response of these receptors is 1D (strong or weak). No information on wavelength.

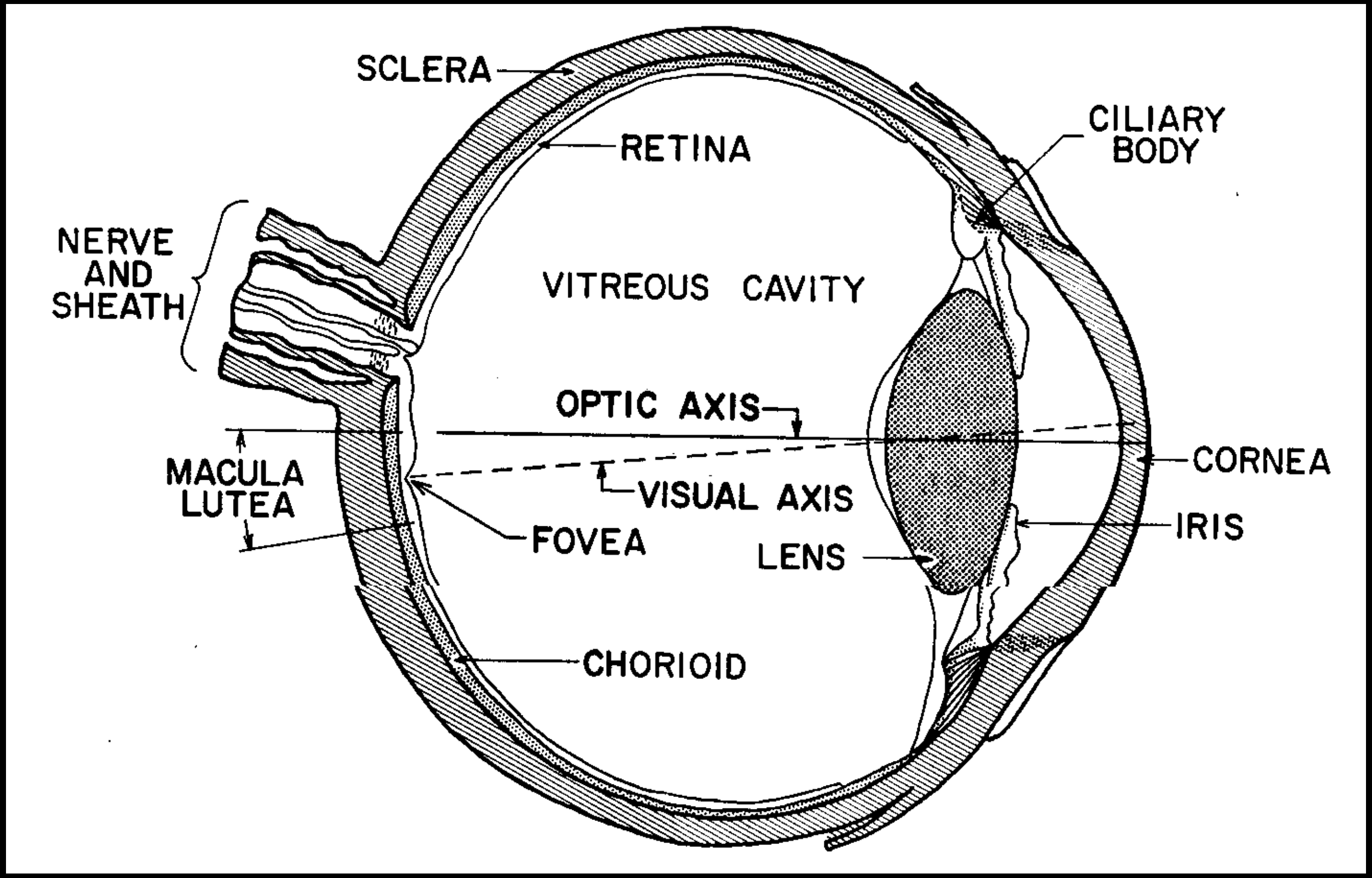
- Because of linearity, the response of each photoreceptor can be modeled as $p_k = \int_\Lambda \sigma_k(\lambda) E(\lambda) d\lambda$

- The sensitivity σ_k can be measured experimentally.



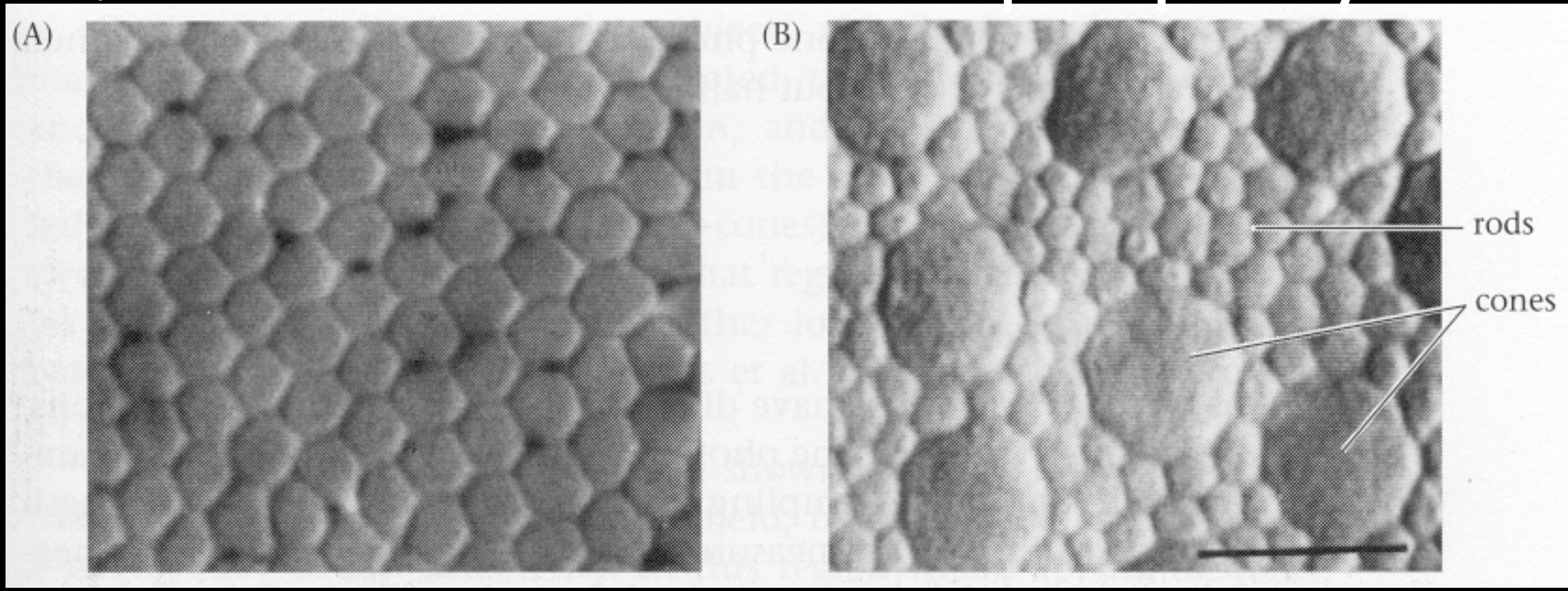
Spectral sensitivity of the three types of cones

The human eye..



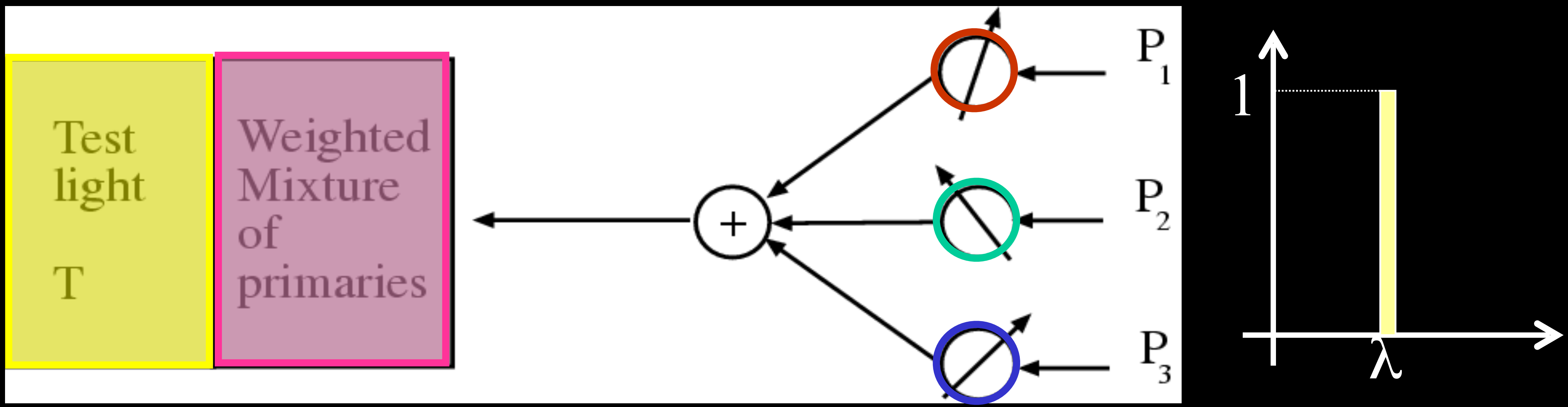
Cones in the fovea

Rods and cones in the periphery



Color Matching Functions

- Problem: given a set of primaries, what are the weights matching a given spectral radiance?
- Experiments:



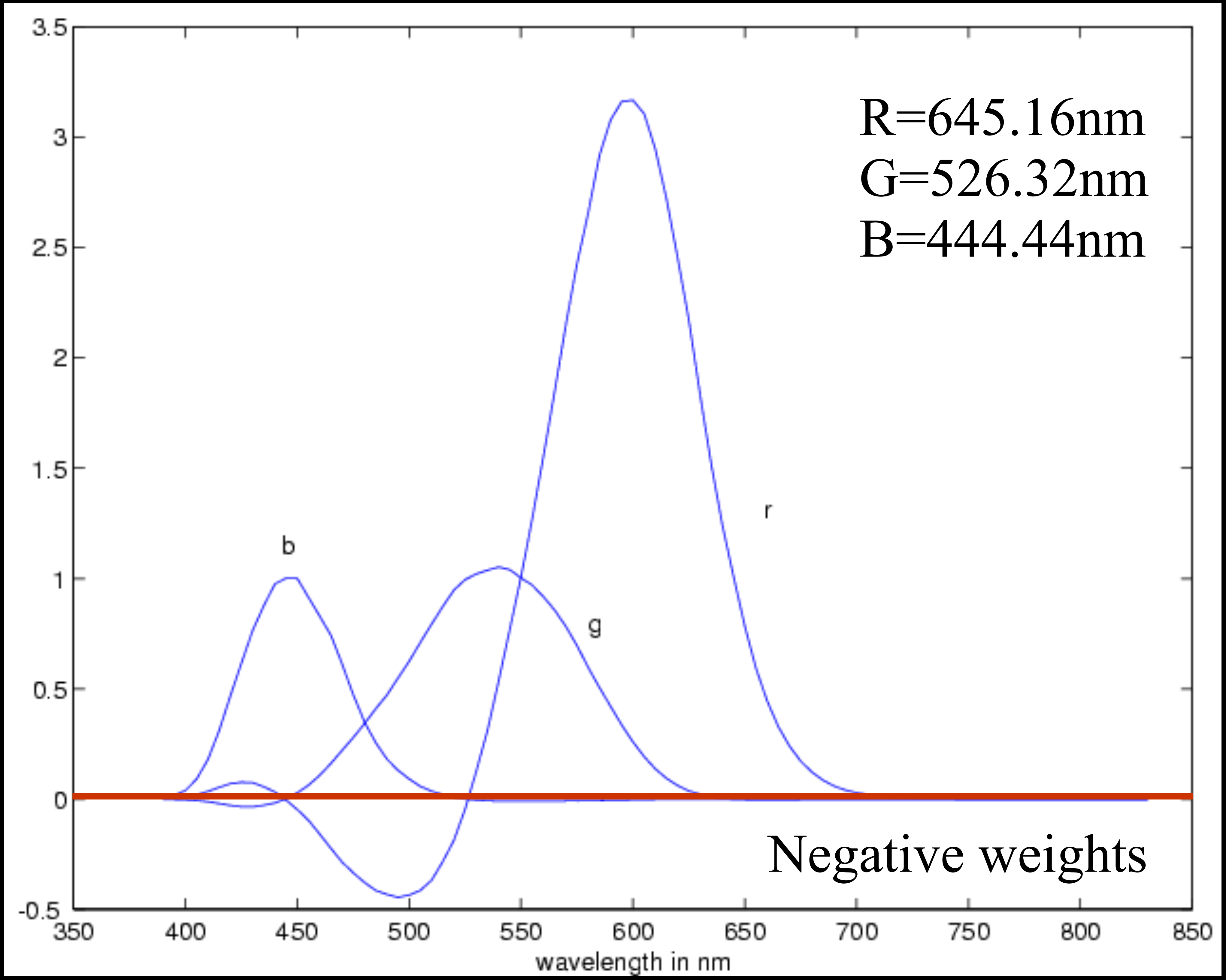
$$L(\lambda) = f_1(\lambda)\,P_1 + f_2(\lambda)\,P_2 + f_3(\lambda)\,P_3$$

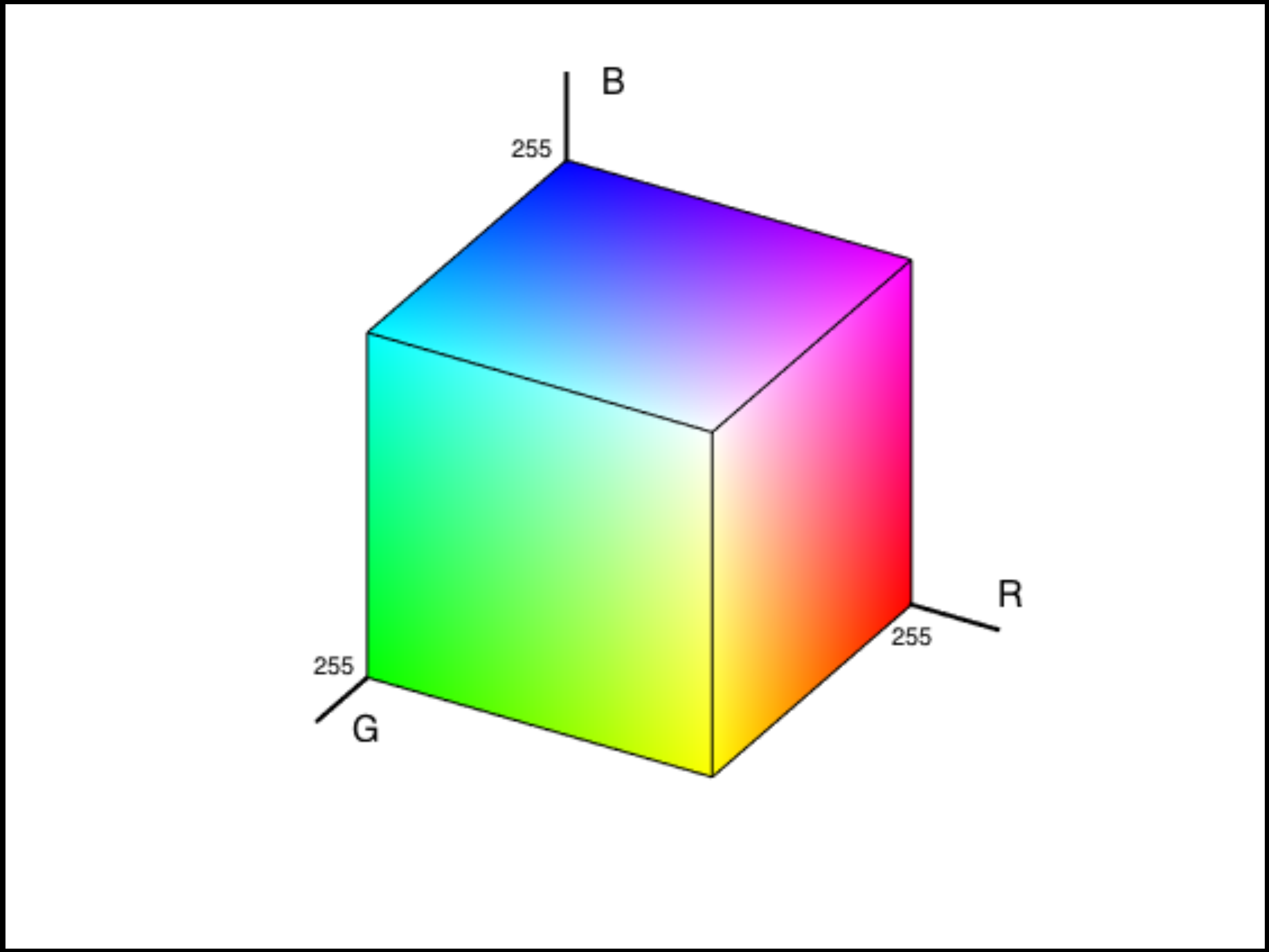
Color matching functions

- To match T use linearity:

$$\boxed{\left\{\int_\Lambda f_1(\lambda)T(\lambda)d\lambda\right\}P_1 + \left\{\int_\Lambda f_2(\lambda)T(\lambda)d\lambda\right\}P_2 + \left\{\int_\Lambda f_3(\lambda)T(\lambda)d\lambda\right\}P_3}$$

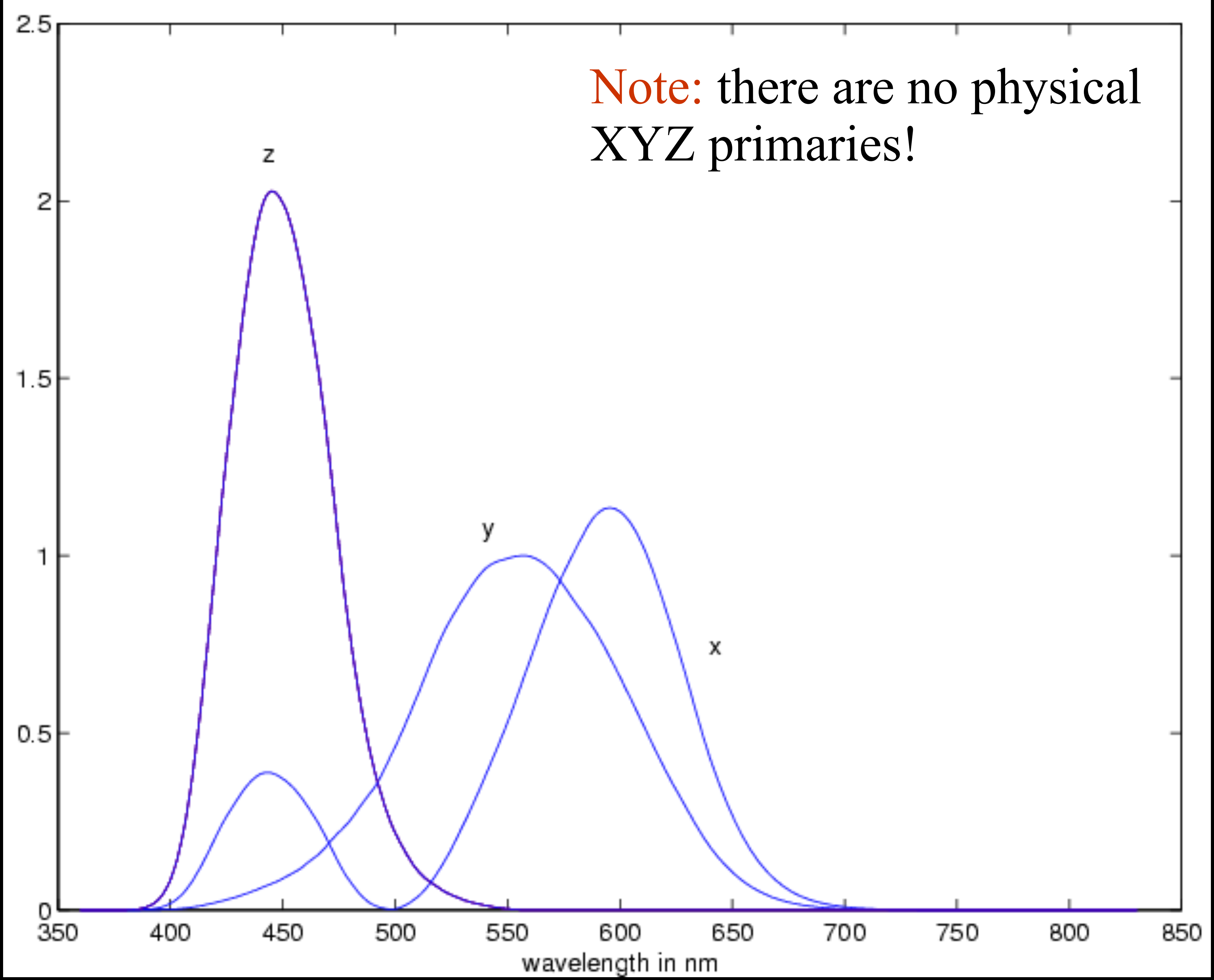
RGB Color Matching Functions

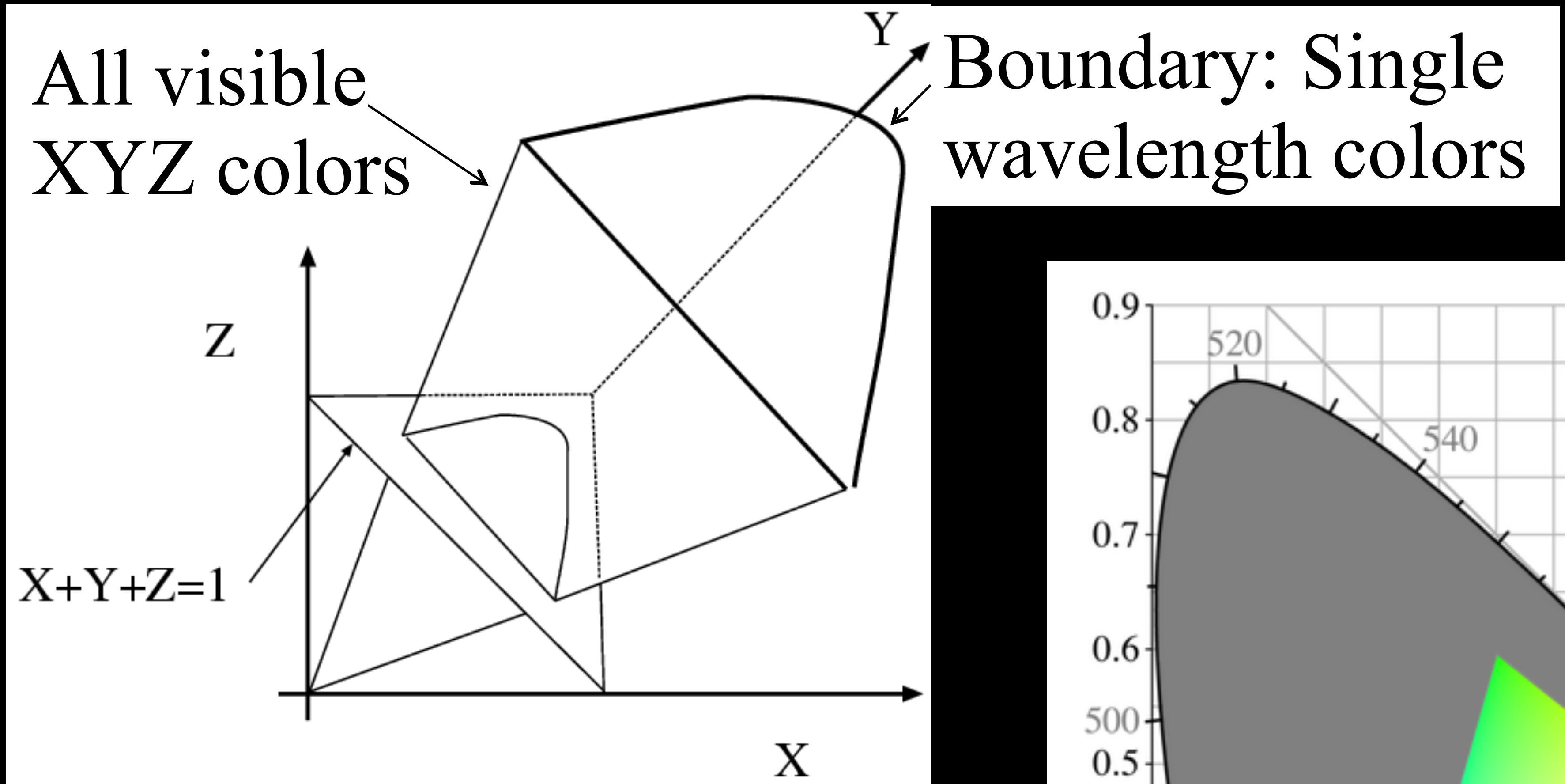


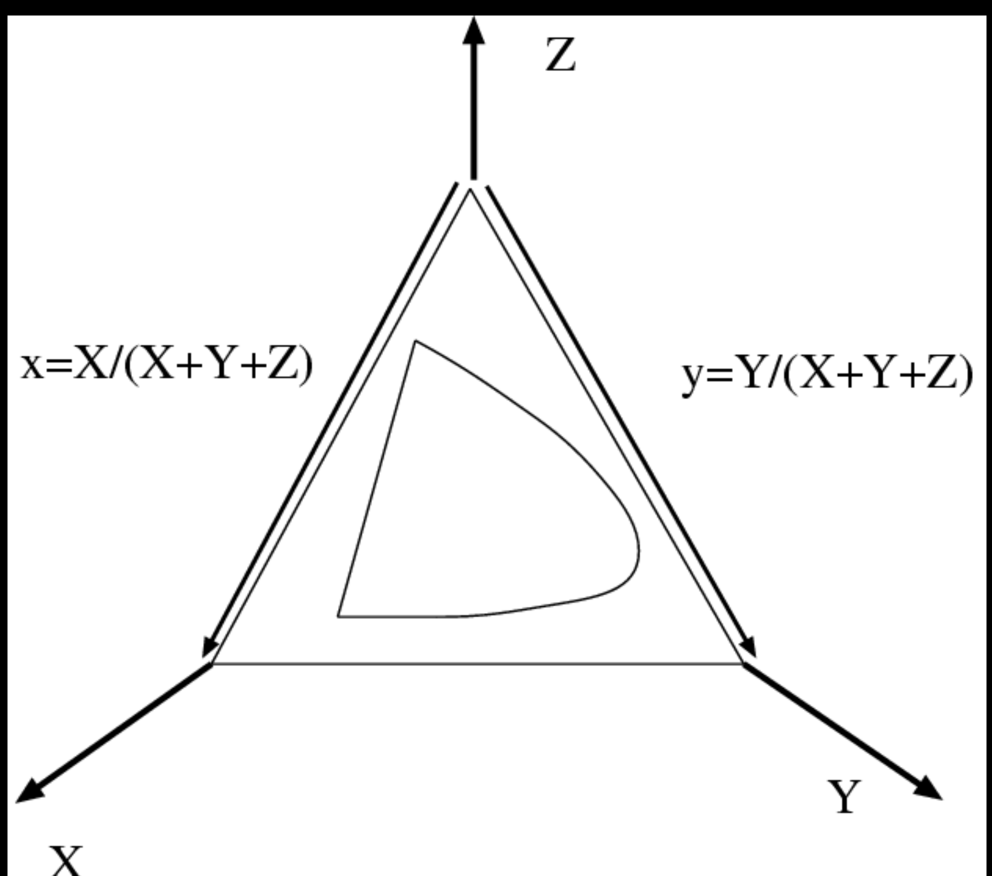


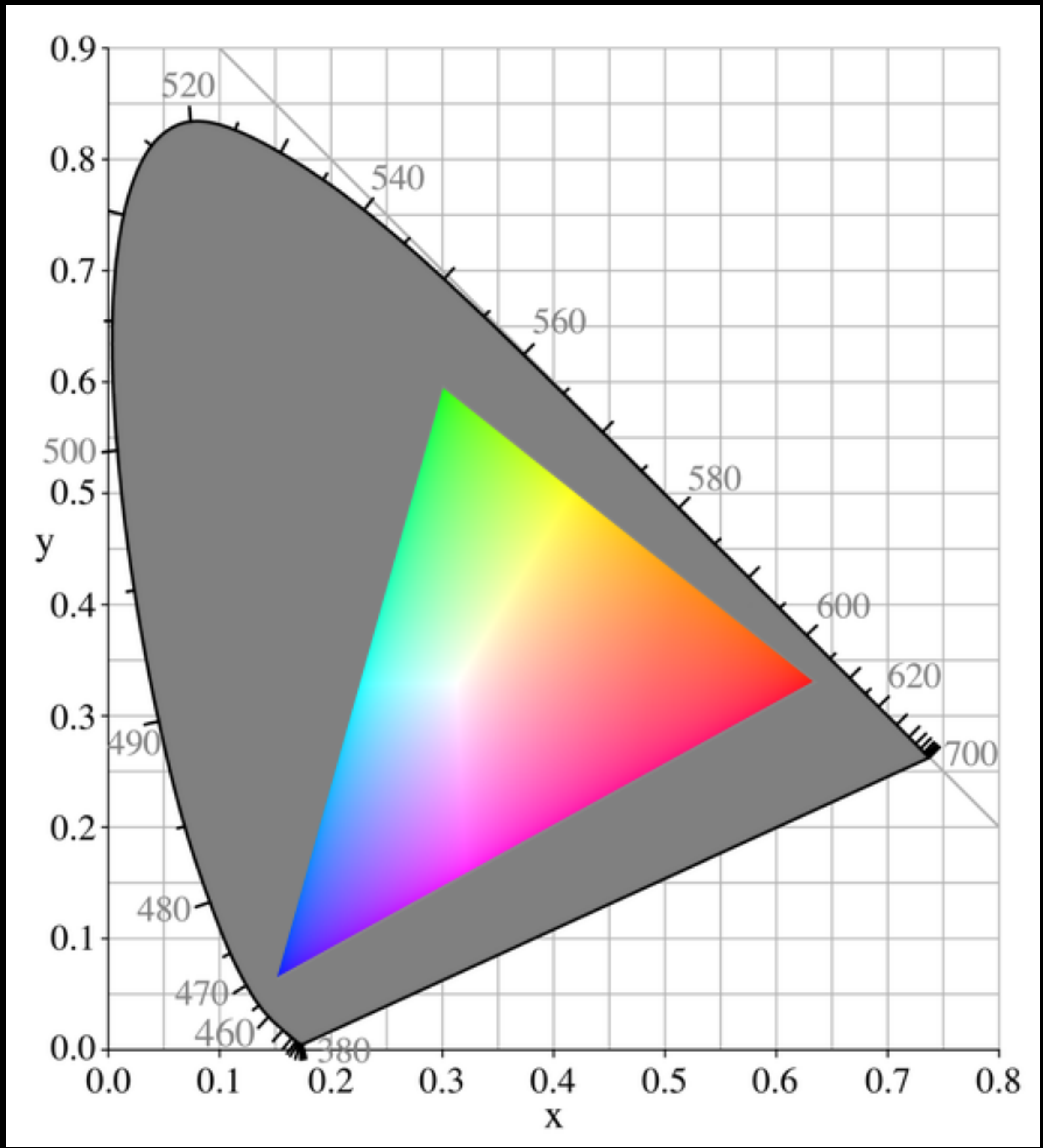
The RGB color cube

CIE XYZ Color Matching Functions









CIE XYZ and xy spaces

Outline

- Texture
  - Textons
  - Bags of words
- Segmentation
  - K means and EM algorithm
  - Mean-shift algorithm
  - Graph cuts

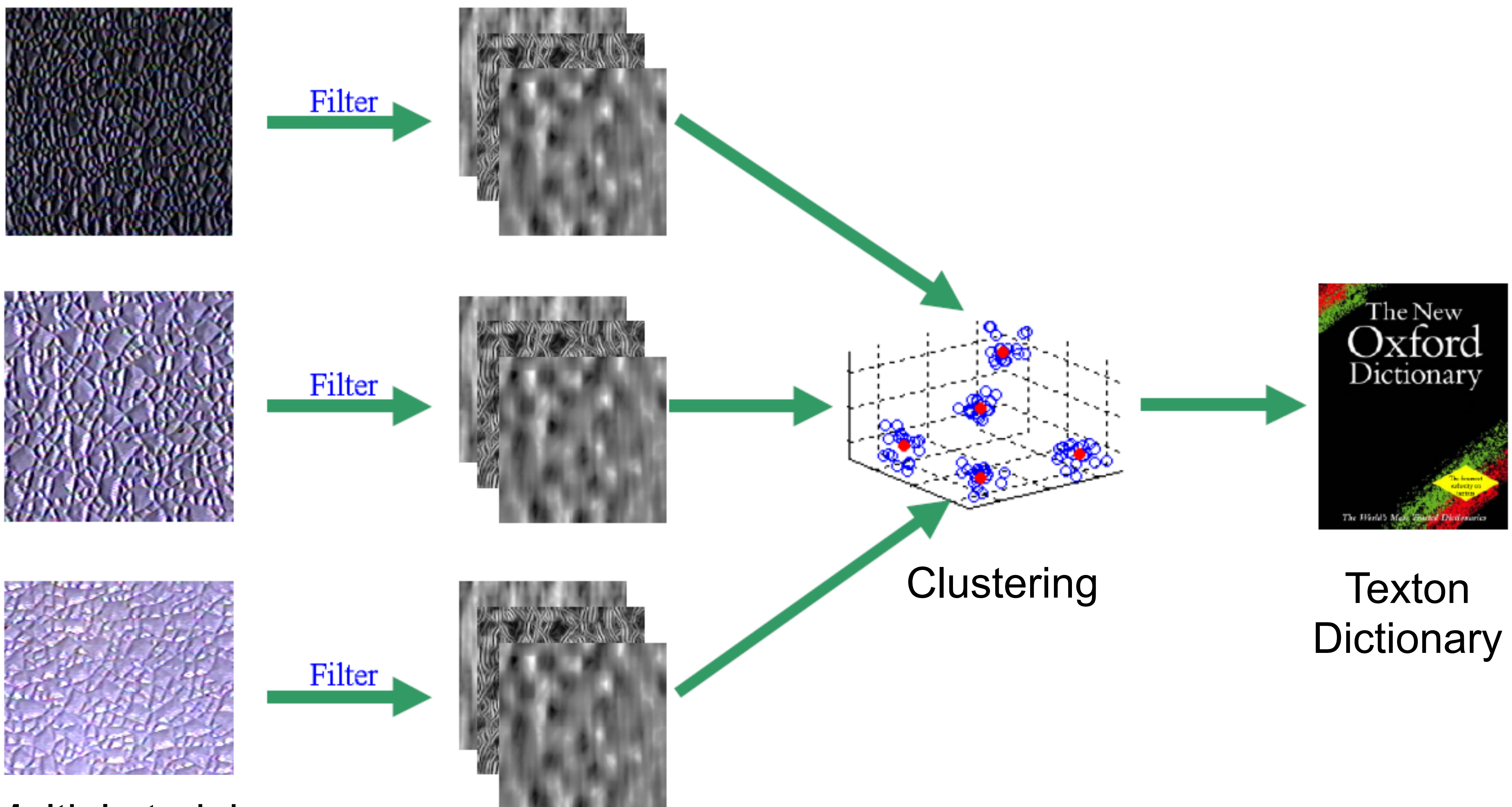
Texture Classification

- Profound observation: Grass and sea pictures don’t look the same!
- Basic idea: Model the distribution of “texture” over the image (or over a region) and classify in different classes based on the texture models learned from training examples.

Grass

Sea

The concept of “texton”

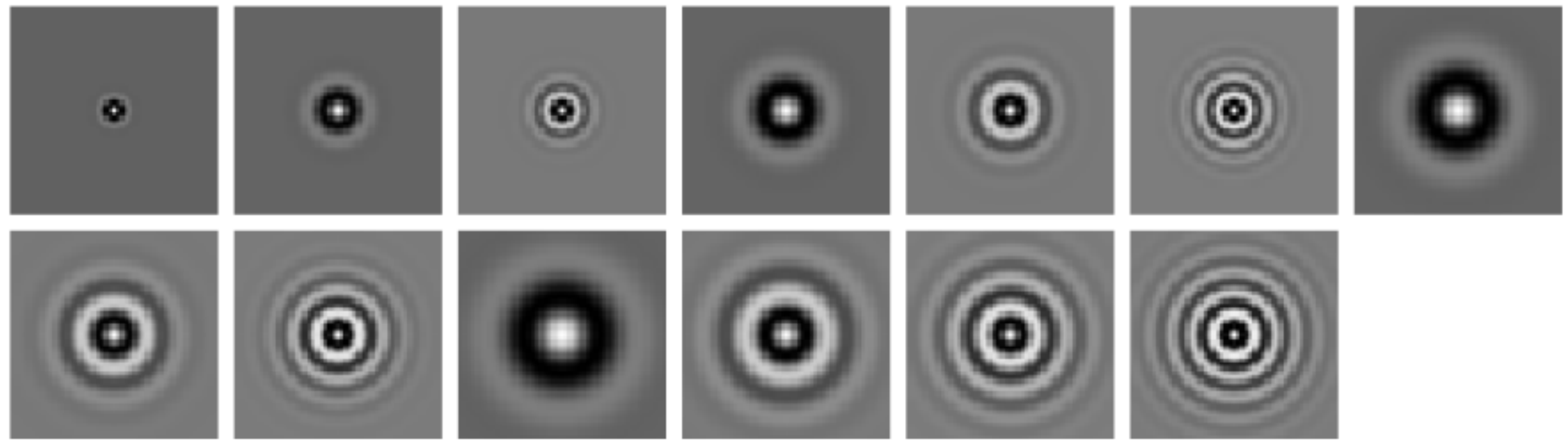


Multiple training images of the same texture

Filter responses over a bank of filters

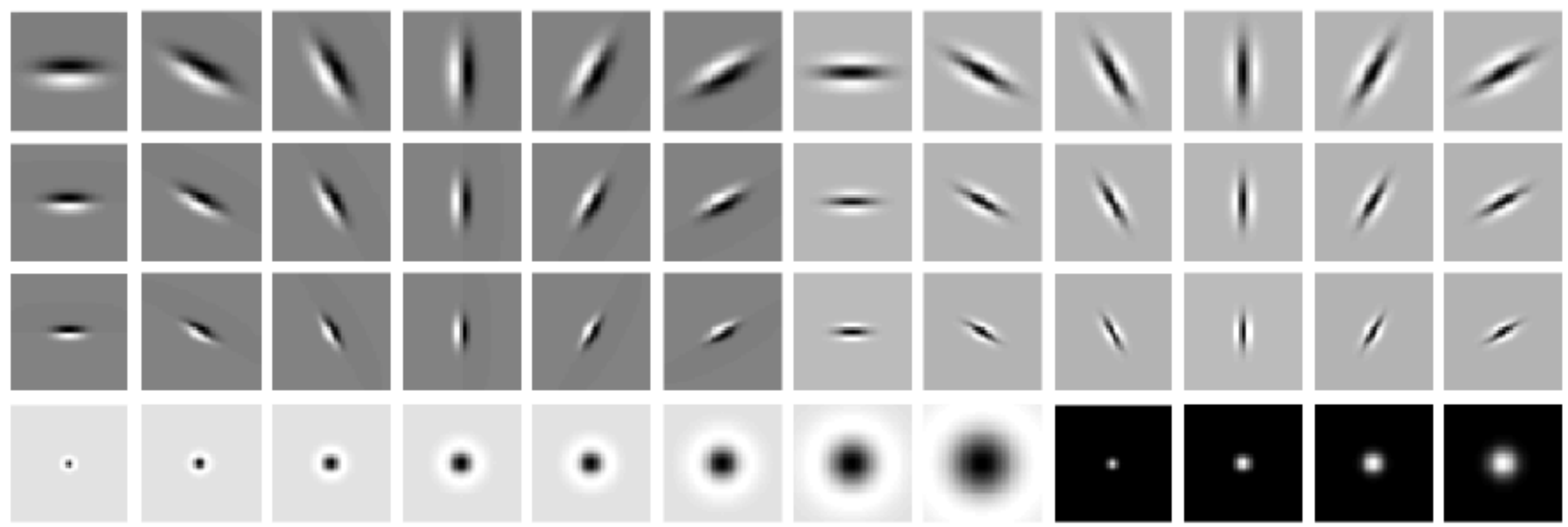
Example of filter banks

Isotropic

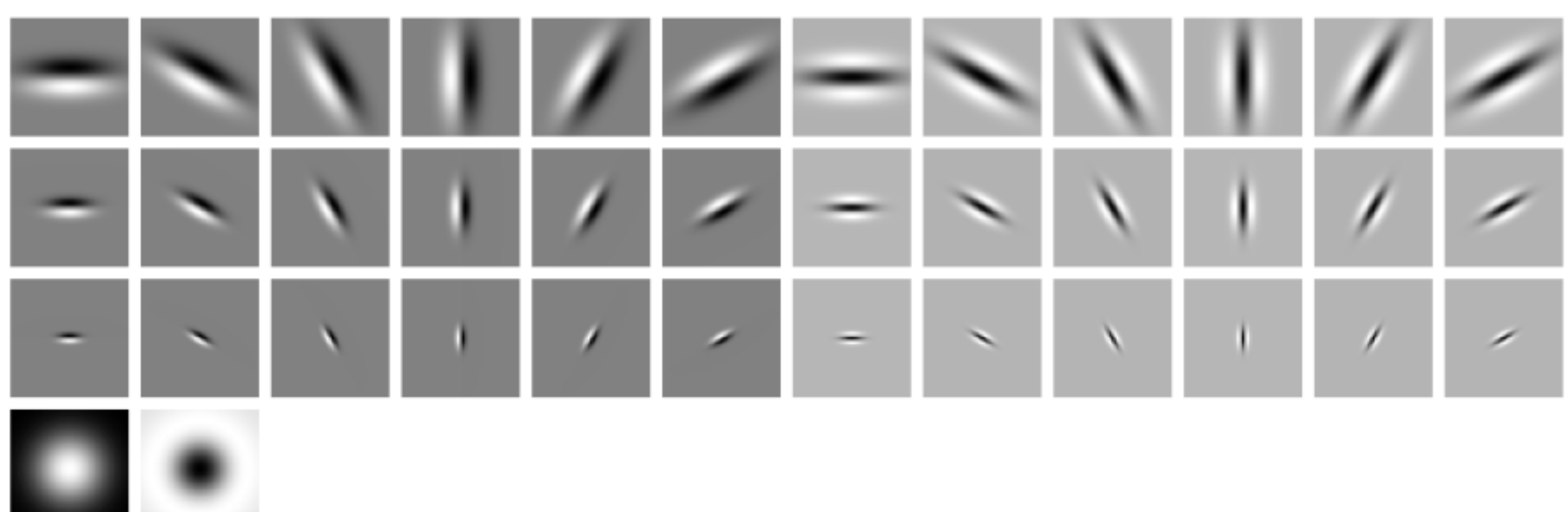


'S'

Gaussian derivatives at different scales and orientations

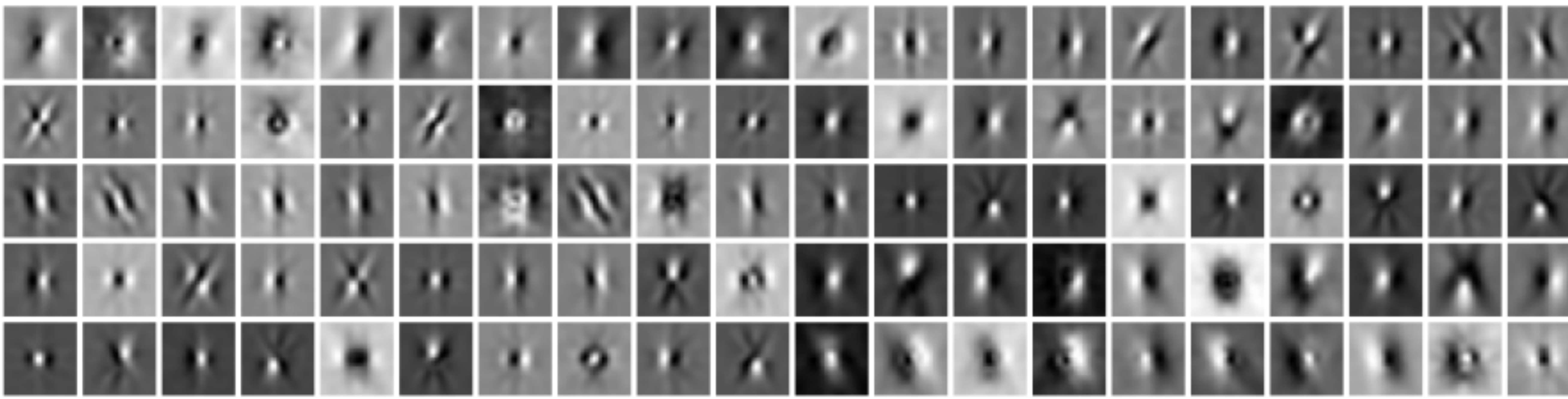


'LM'



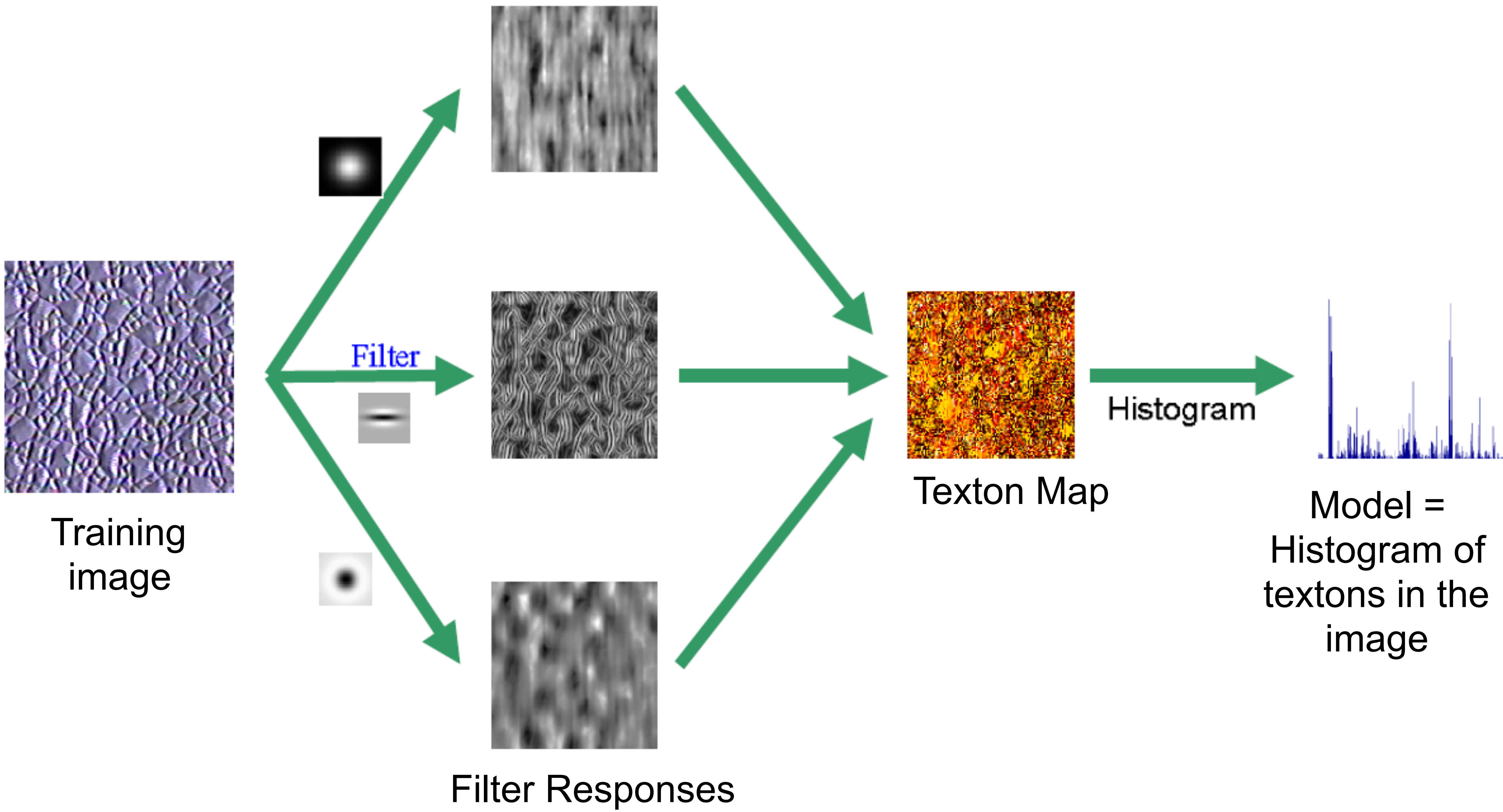
'MR8'

Example of textons (LM)

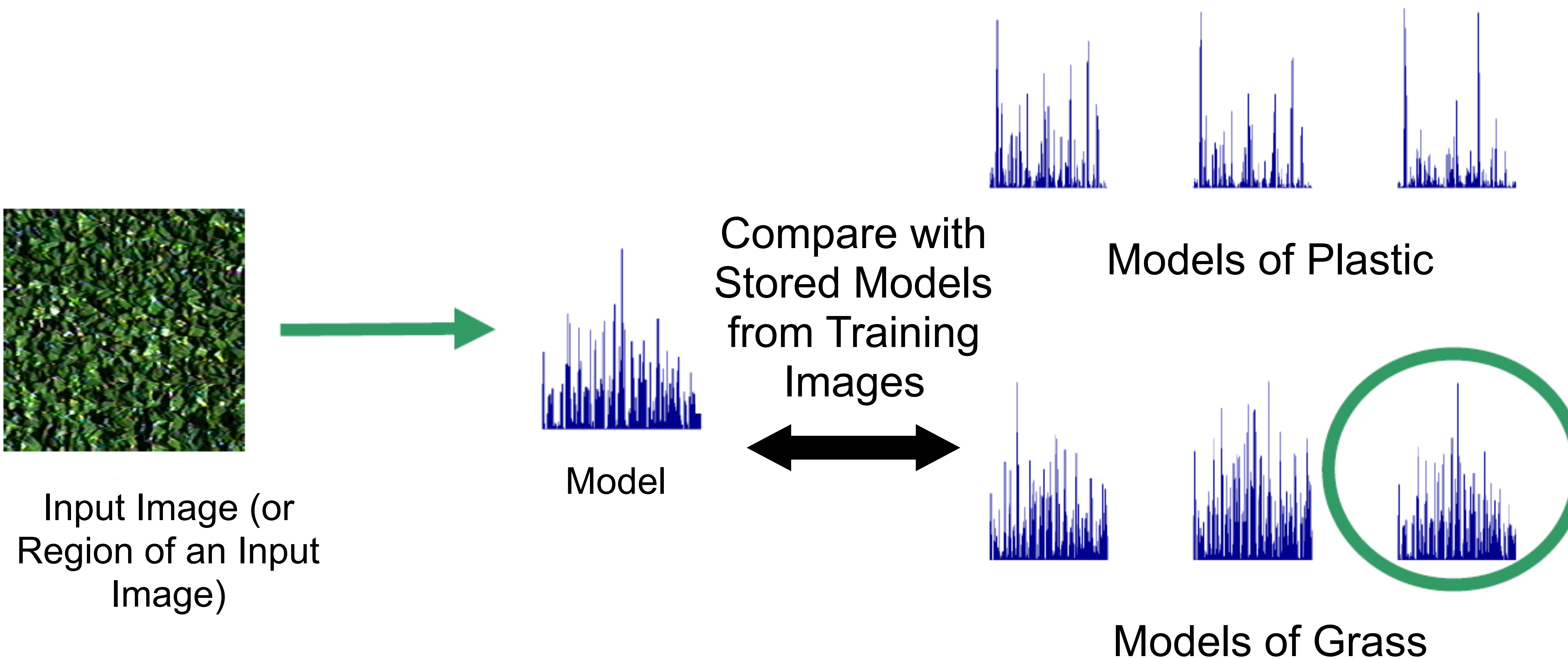


(Linear combinations of filters corresponding to cluster centers)

Modeling texton distributions



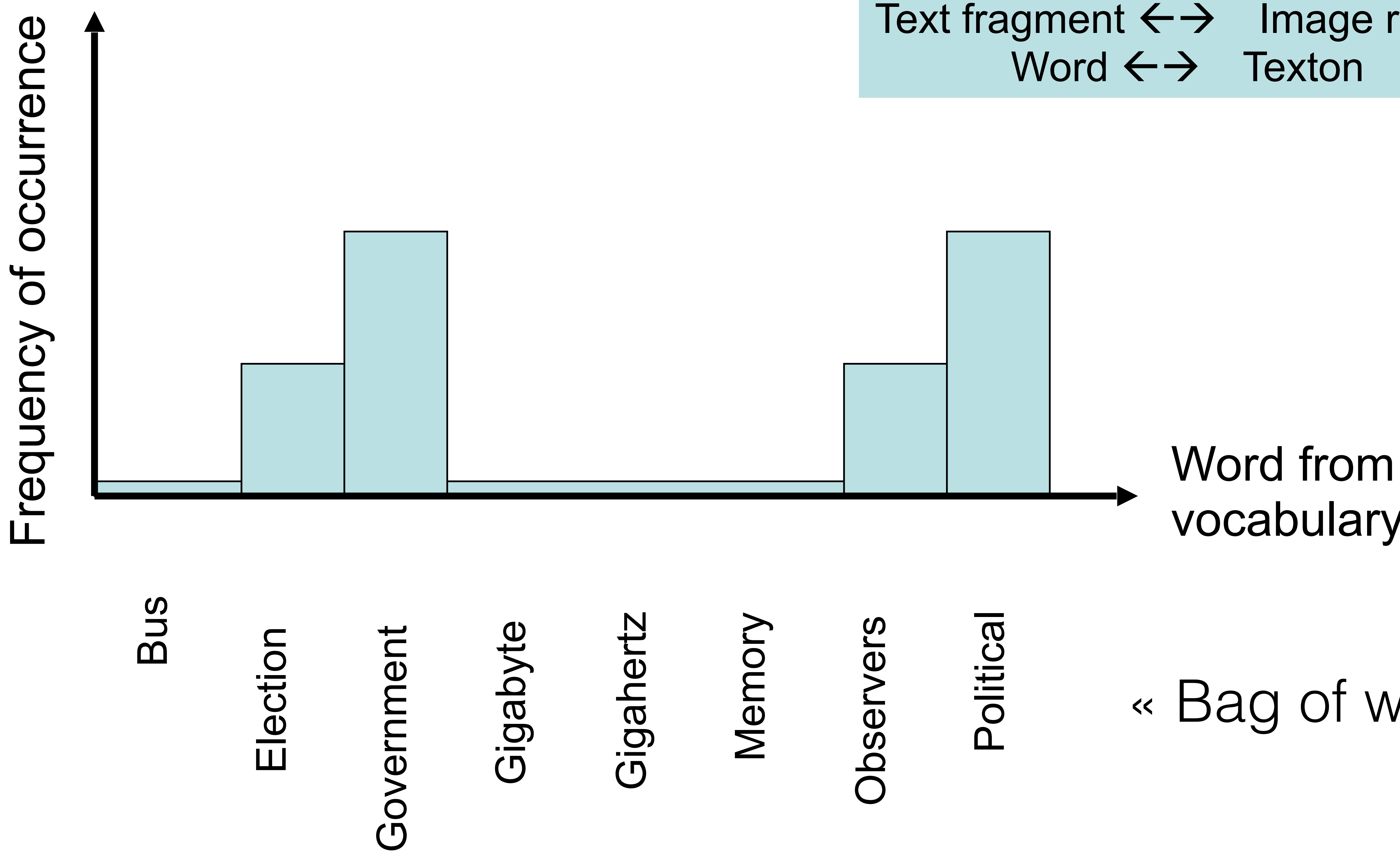
Classification



Analogy with Text Analysis

Political observers say that the government of Zorgia does not control the political situation. The government will not hold elections …

Analogy:
Text fragment ←→ Image region
Word ←→ Texton

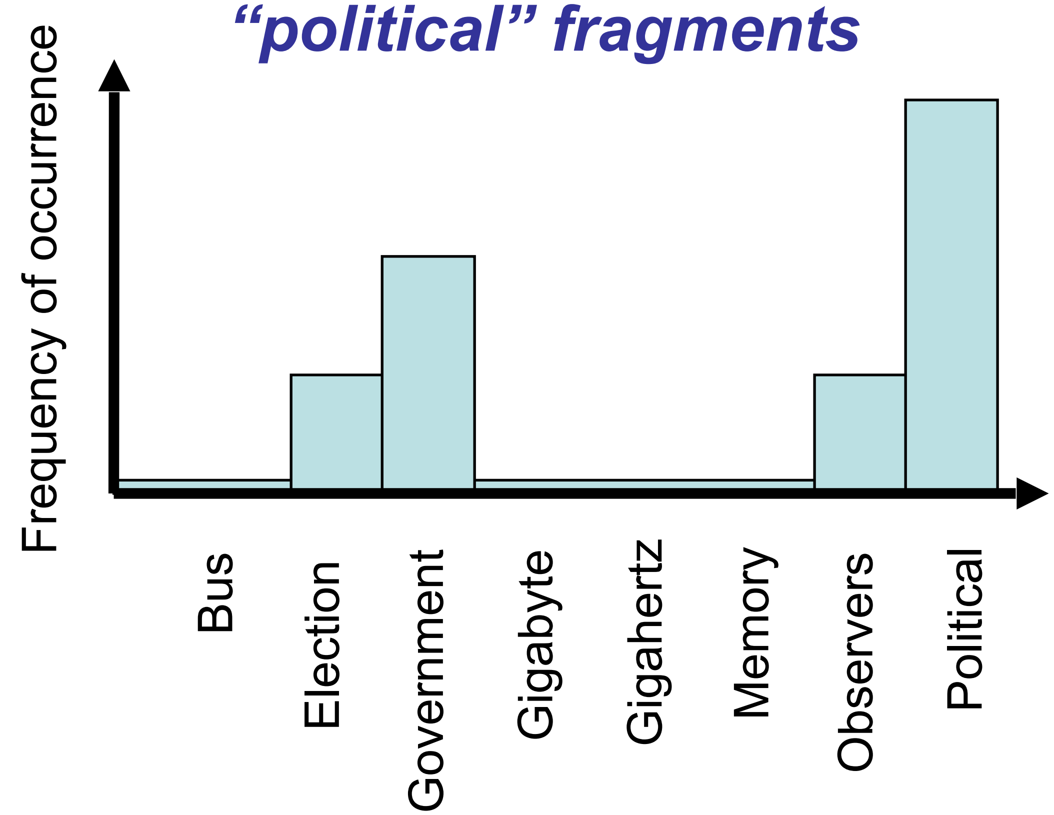


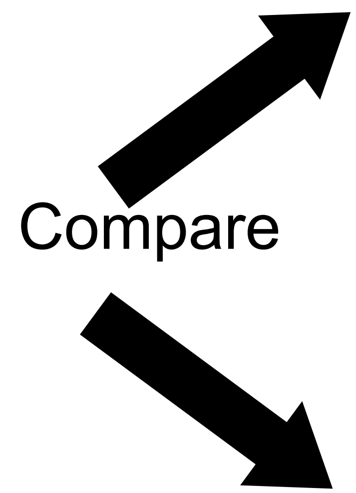
« Bag of words »

Analogy with Text Analysis

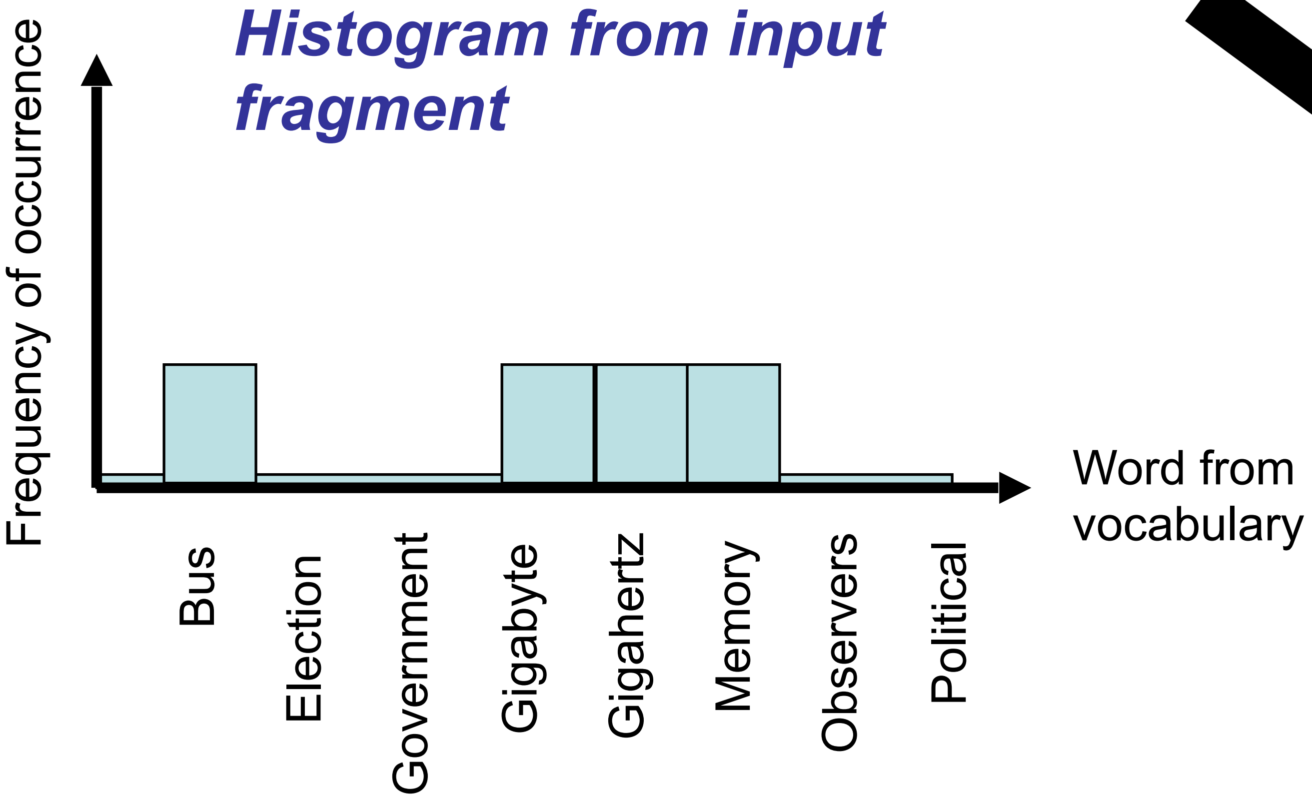
The ZH-20 unit is a 200Gigahertz processor with 2Gigabyte memory. Its strength is its bus and high-speed memory……

Histogram from training "political" fragments

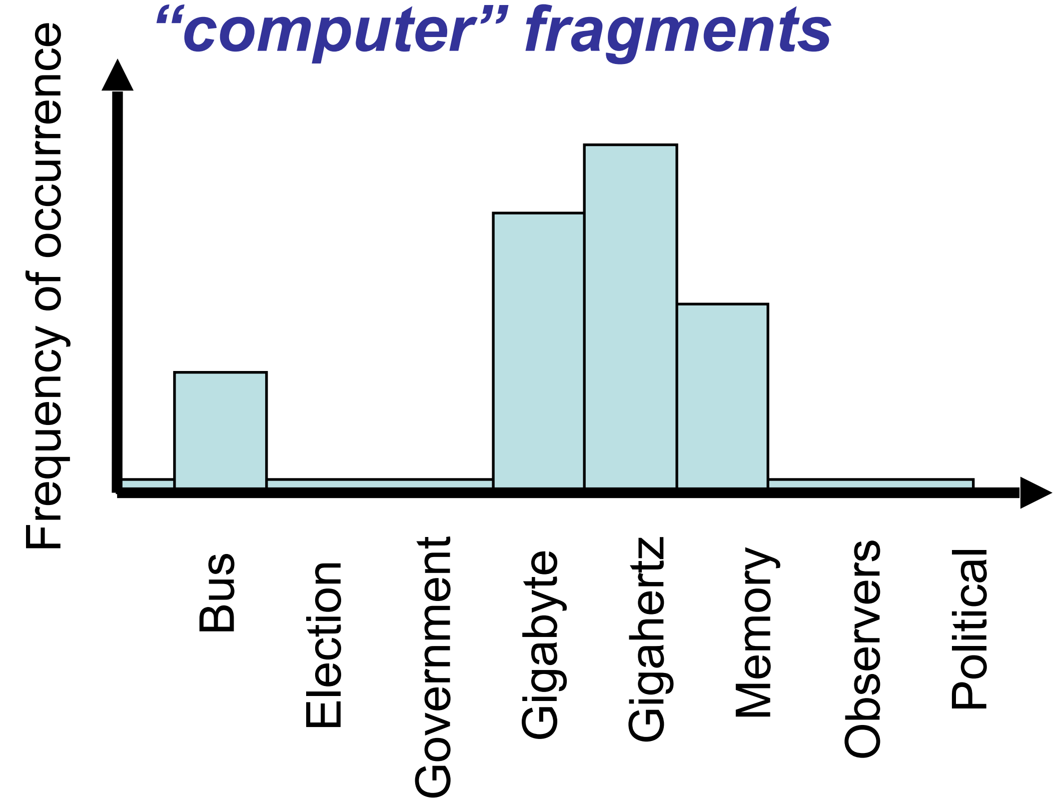




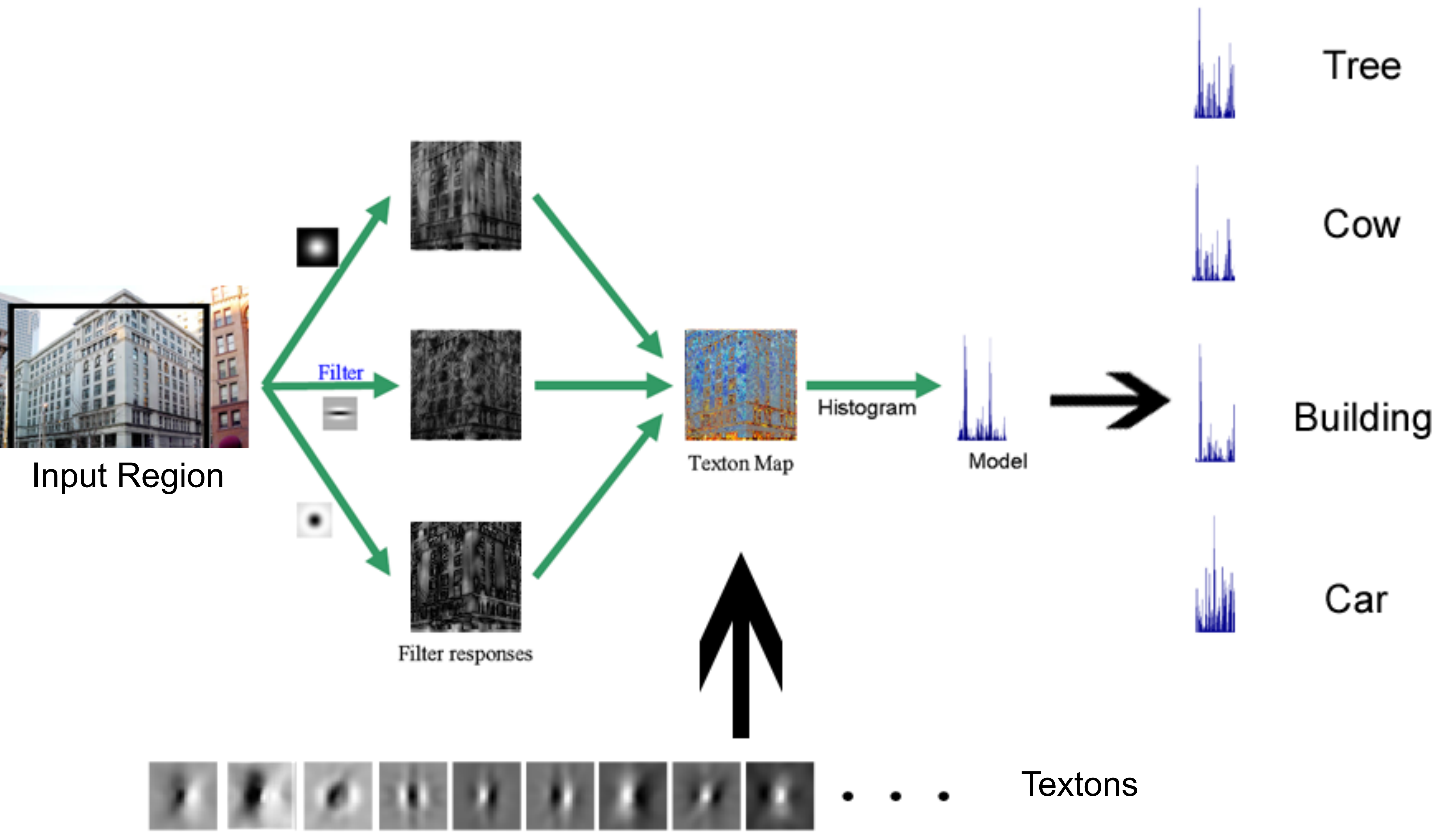
Histogram from input fragment



Histogram from training "computer" fragments



Example Classification



Examples

- 1


cow
cow
grass
sky
aeropla
grass
tree
car
cow
sky
aeroplane
tree
car
car
bike
sky
building
grass
face
bike
bike
building
tree
face
grass

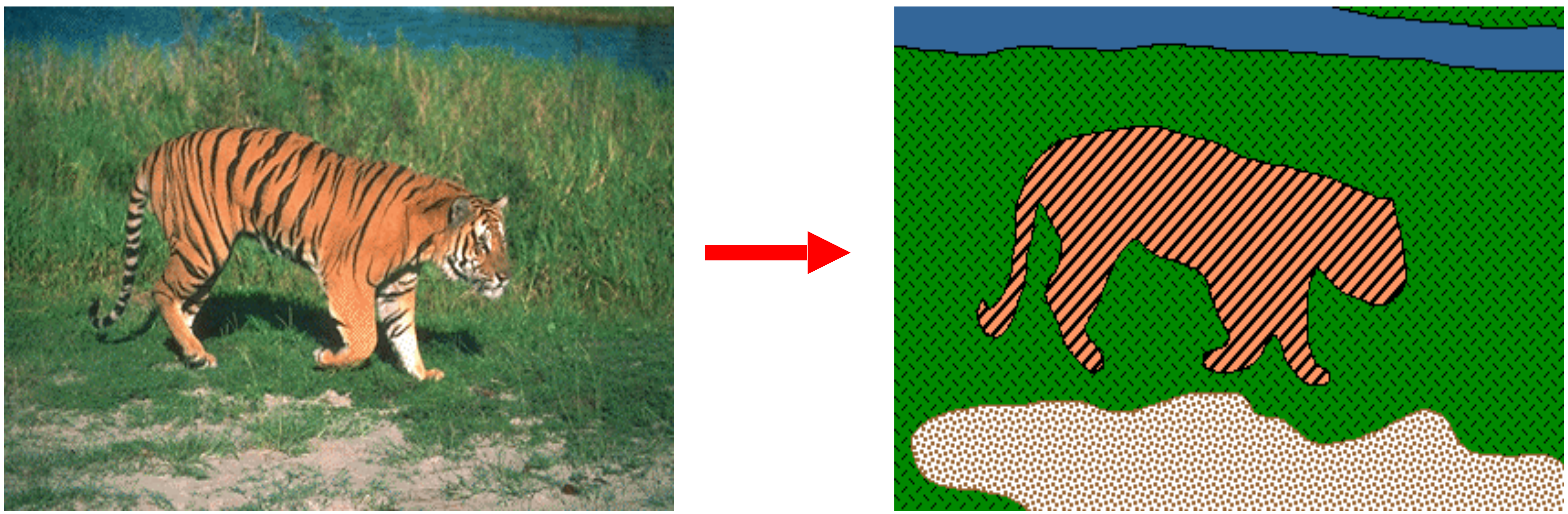
Summary

- Sources:
  - J. Winn, A. Criminisi and T. Minka. Object Categorization by Learned Universal Visual Dictionary. Proc. IEEE Intern. Conf. Comp. Vision. 2005. (Also Csurka et al., 2004)
  - M. Varma and A. Zisserman. A statistical approach to texture classification from single images. IJCV, 62(1–2):61–81, April 2005. (Also Lazebnik et al., 2003)

- Questions:
  - How many textons/words?
  - What filters?
  - How to construct clusters?
  - How to compare histogram distributions?
  - How to exploit the spatial distribution of textons (these examples completely ignore the relative positions of textons in the image)?

- Will be revisited for object recognition

Segmentation and clustering



Segmentation challenges

Kittens are distinguishable by color (sort of), but not texture.

Chameleon is distinguishable by texture, but not color.

Wheels are part of the car, but not similar in color or texture.

How do we recognize that the head and body/sweater are the same “person”?

Segmentation challenges

http://optical-illusions.wikia.com/wiki/Emergence

Segmentation issues

- “Bottom-up” or “top-down” process?
- Supervised or unsupervised?
- What is the application?

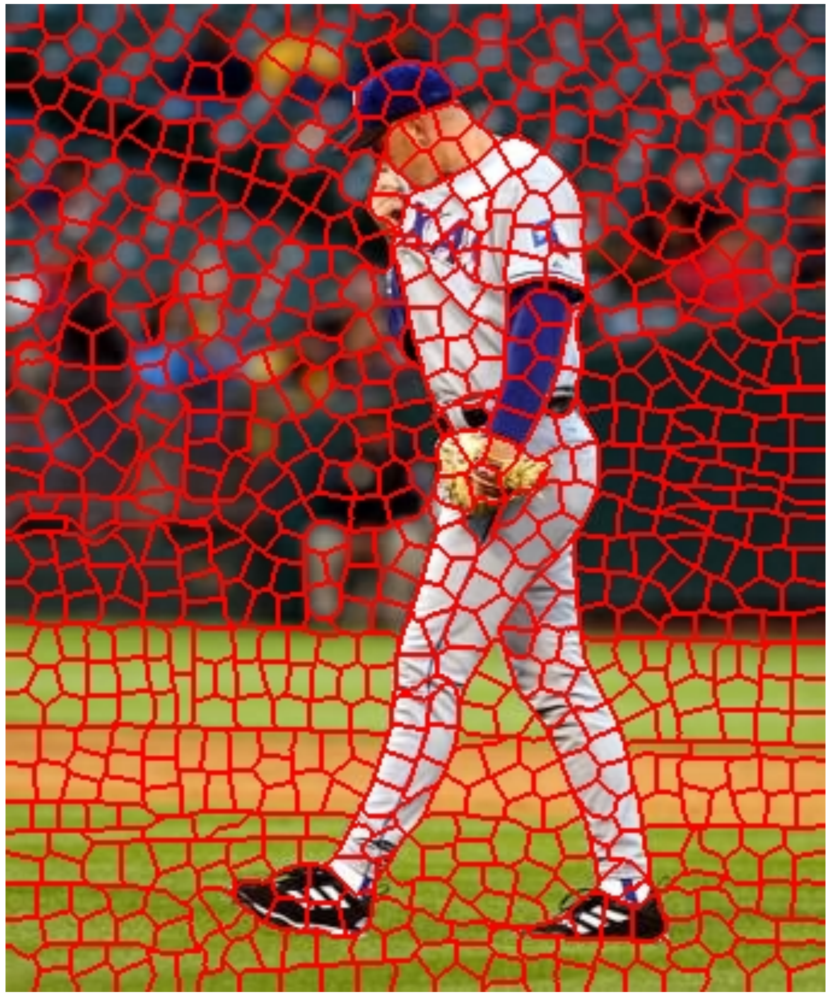
Outline

- Segmentation as clustering
  - K-means
  - EM algorithm
  - Mean shift
- Segmentation as graph cutting
  - Normalized cuts
  - CRF energy functions, graph cut optimization

The goals of segmentation

- Group together similar-looking pixels for efficiency of further processing
  - "Bottom-up" process
  - Unsupervised

"superpixels"



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

The goals of segmentation

- Separate image into coherent "objects"
  - This is an ill-defined task!

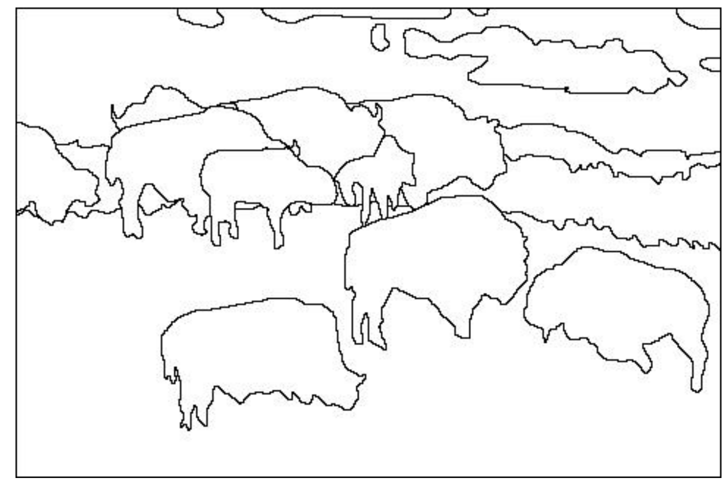
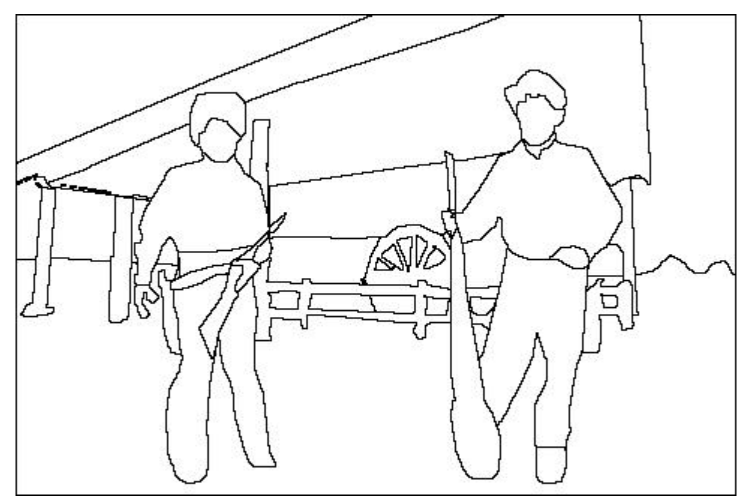



image　　human segmentation



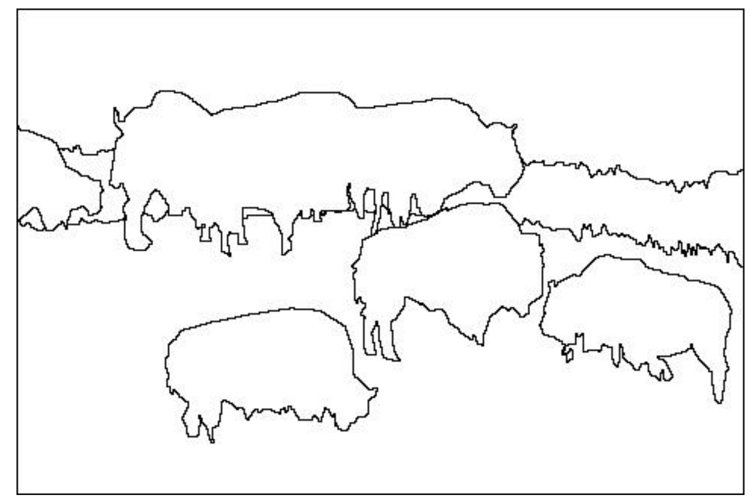
Berkeley segmentation database:

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

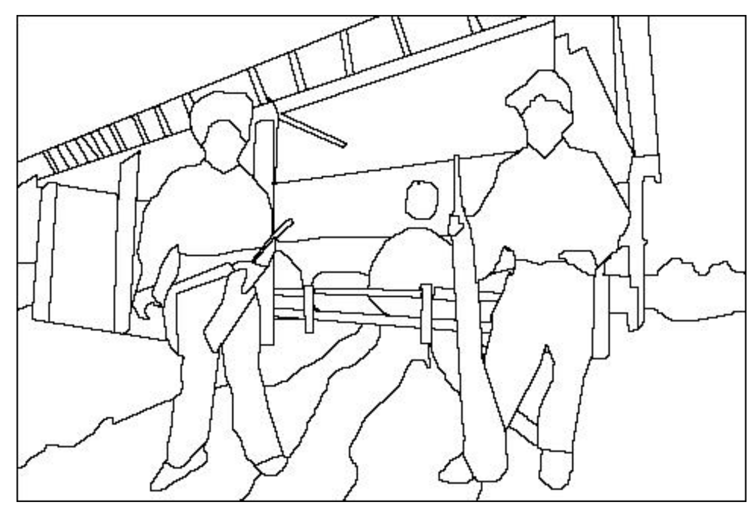
The goals of segmentation

- Separate image into coherent "objects"
  - This is an ill-defined task!

image



human segmentation



Berkeley segmentation database:

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

More About Background Removal in Office 2010

Office Team 30 Nov 2009 5:37 AM | 22

Hi, I'm Tucker Hatfield, a PM on the Office Graphics team, and a while back I posted an introduction to Background Removal (The Magic of Background Removal). In this post I'll build on what I showed you last time by giving an example of how to make more detailed background removal and how to do some interesting things with the results.

As I showed before, in many cases you'll find you can effectively remove a background with nothing more than a very little adjustment of the marquee, but other times it takes a little work. Here's a good example of a photo that takes a bit more attention.

If we just clicked Remove Background and accepted the default marquee for this picture, you'd get the result below. You can see that even with the default marquee it did a pretty good job of guessing what the subject was, but there are a few problems.

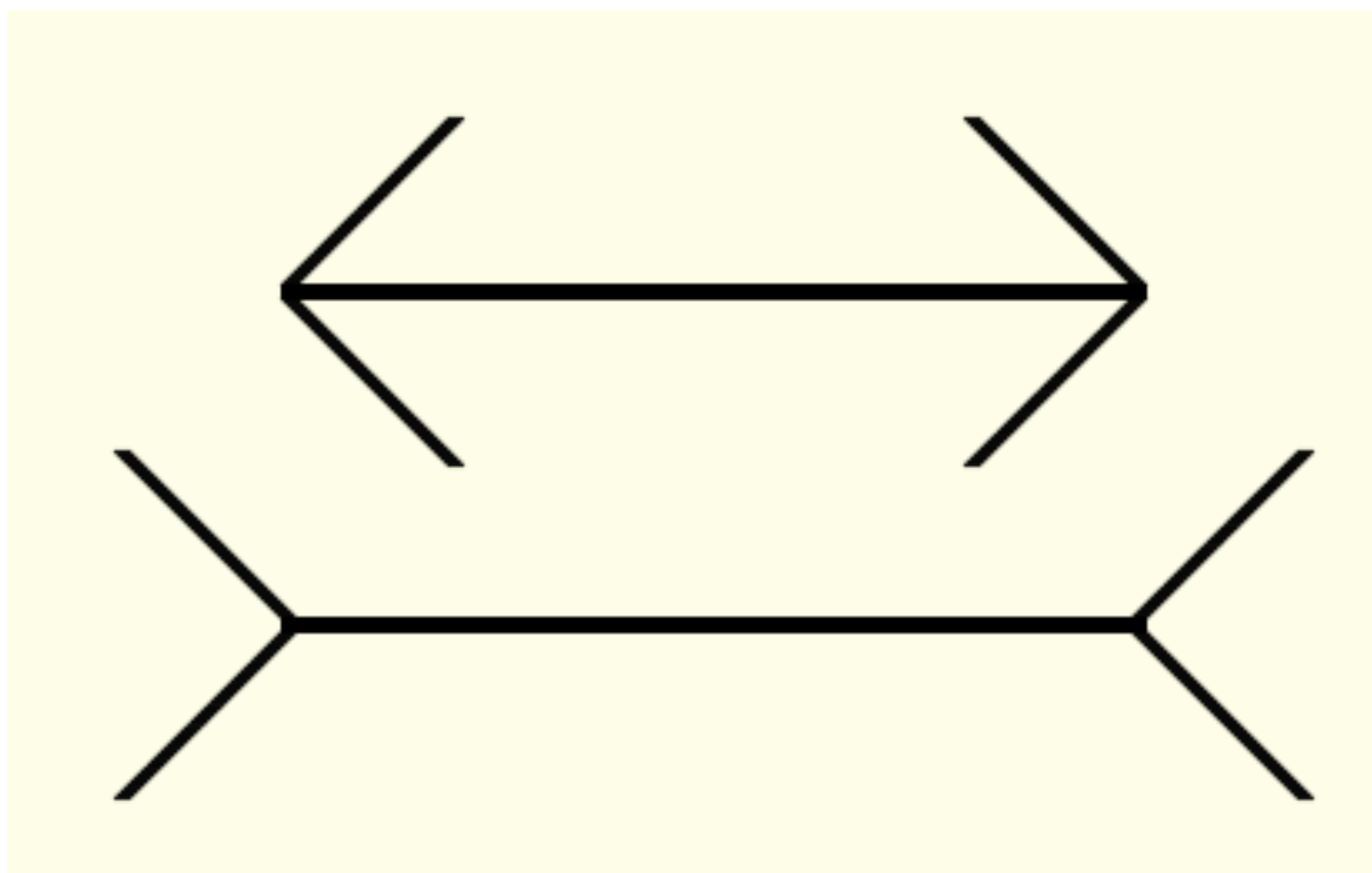
Interactive segmentation

Inspiration from psychology

- The Gestalt school: Grouping is key to visual perception

  (German: *Gestalt* - "essence or shape of an entity's complete form")

The Muller-Lyer illusion



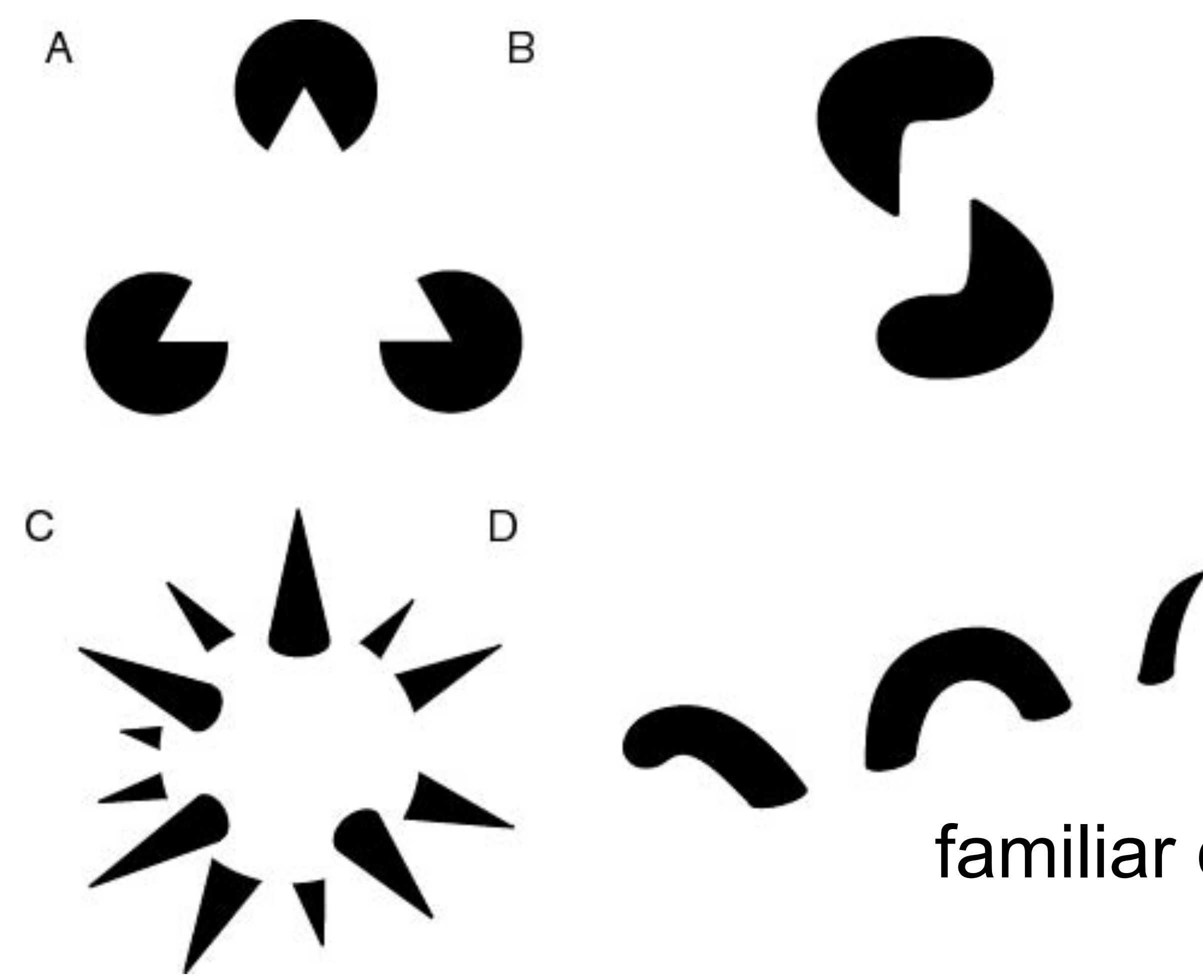
http://en.wikipedia.org/wiki/Gestalt_psychology

The Gestalt school

- Elements in a collection can have properties that result from relationships
  - “The whole is greater than the sum of its parts”

subjective contours

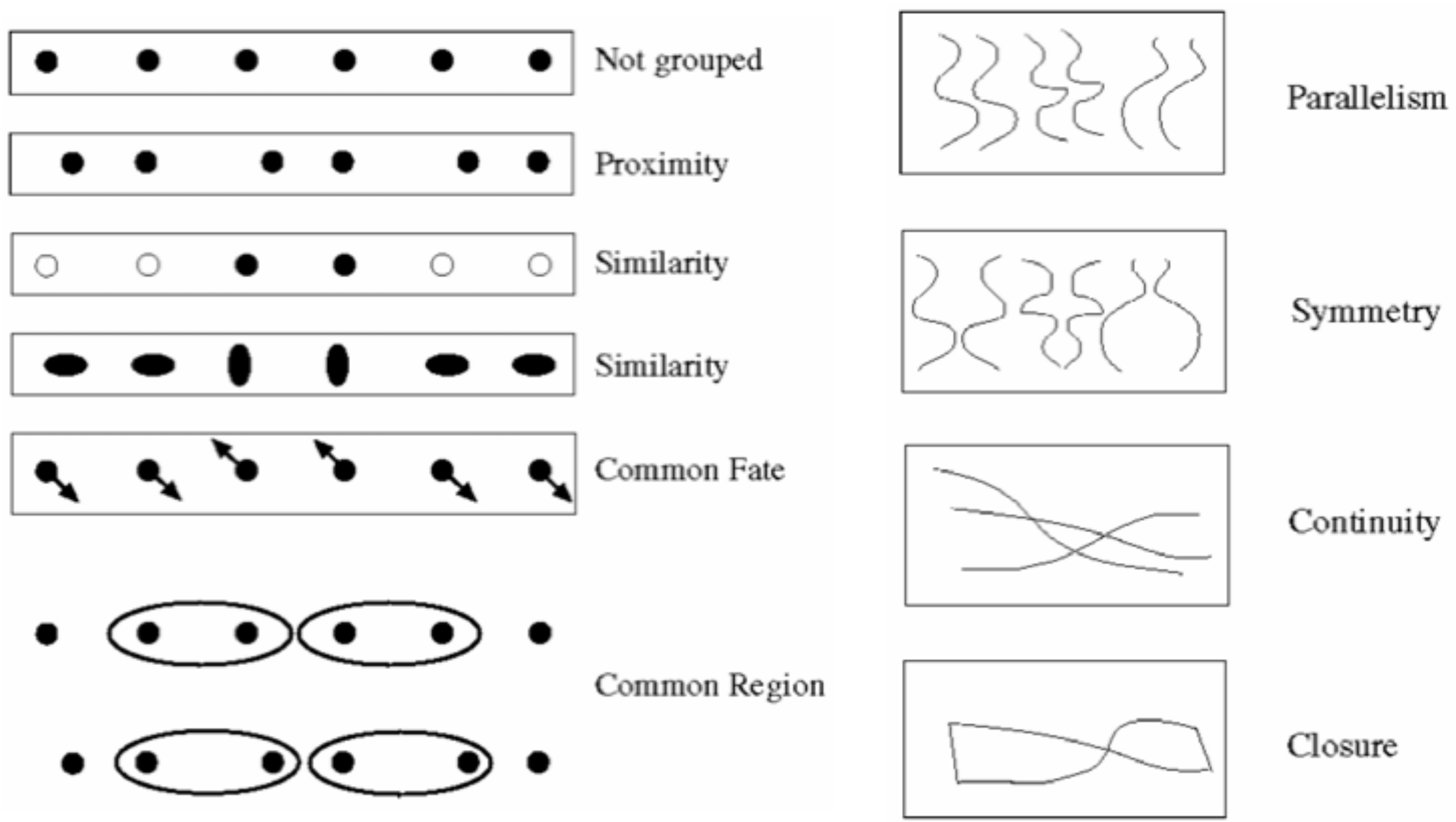
occlusion



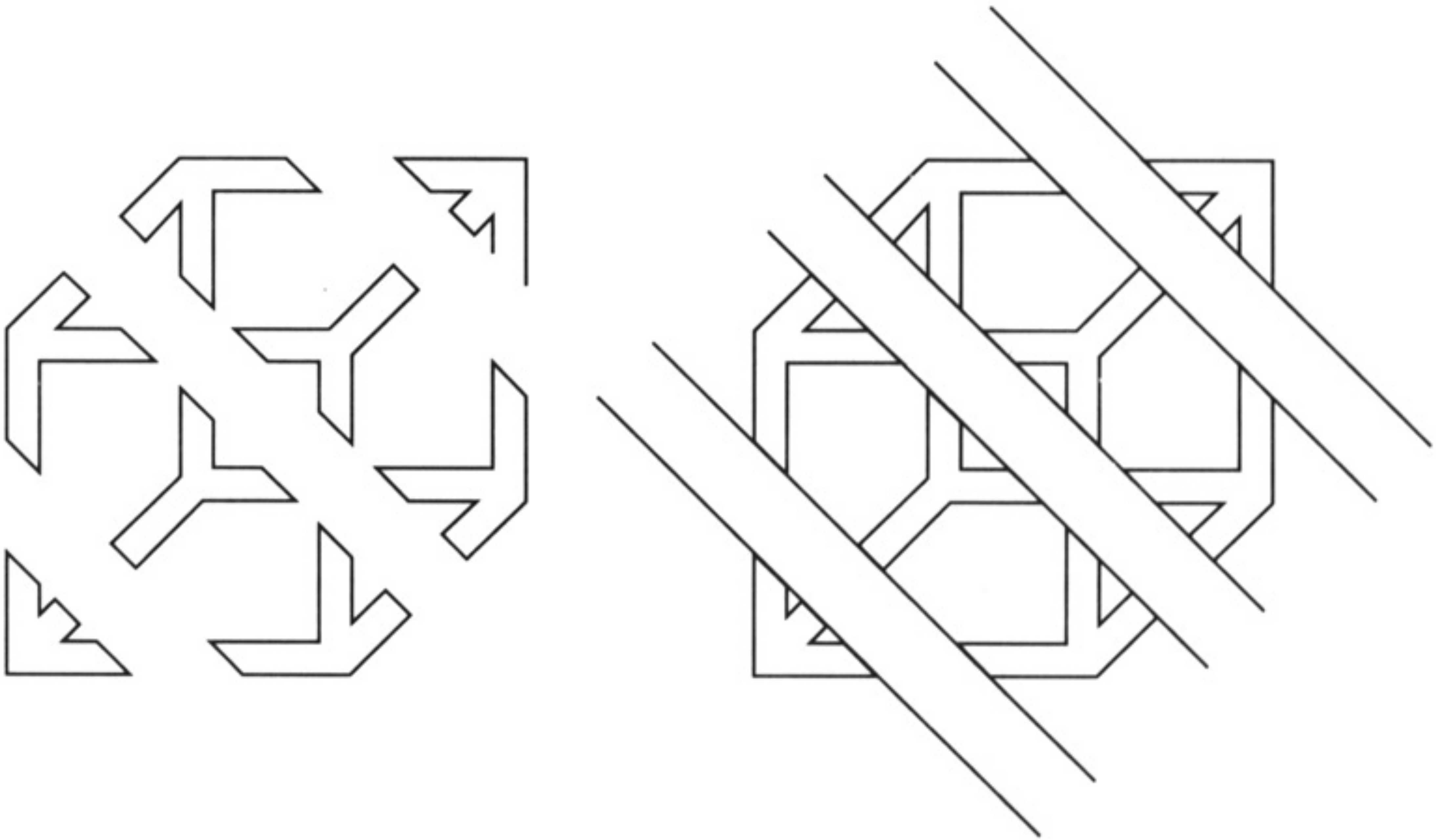
familiar configuration

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

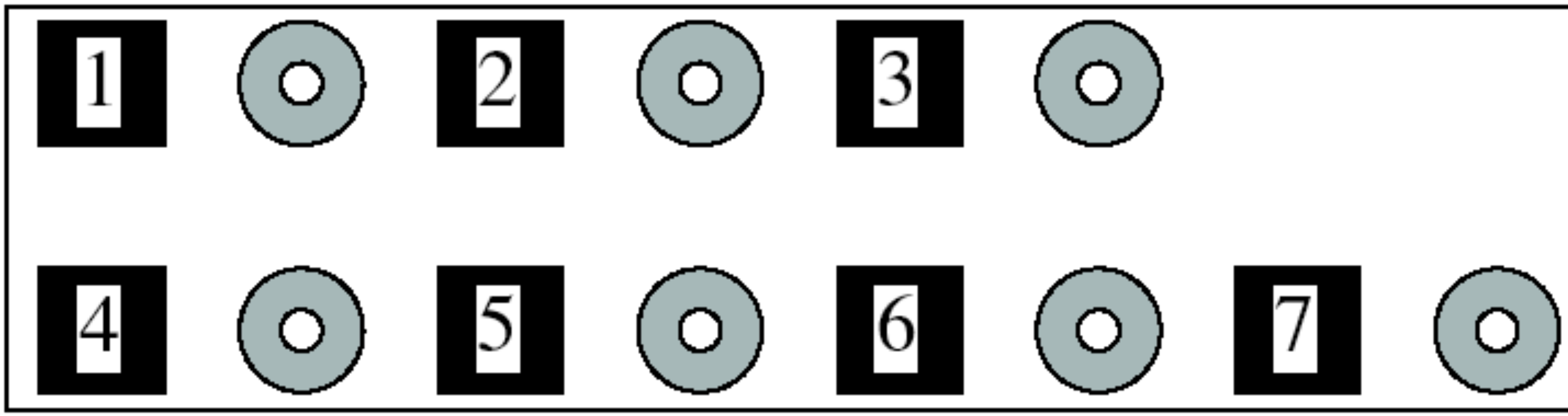
Gestalt factors



Grouping by occlusion

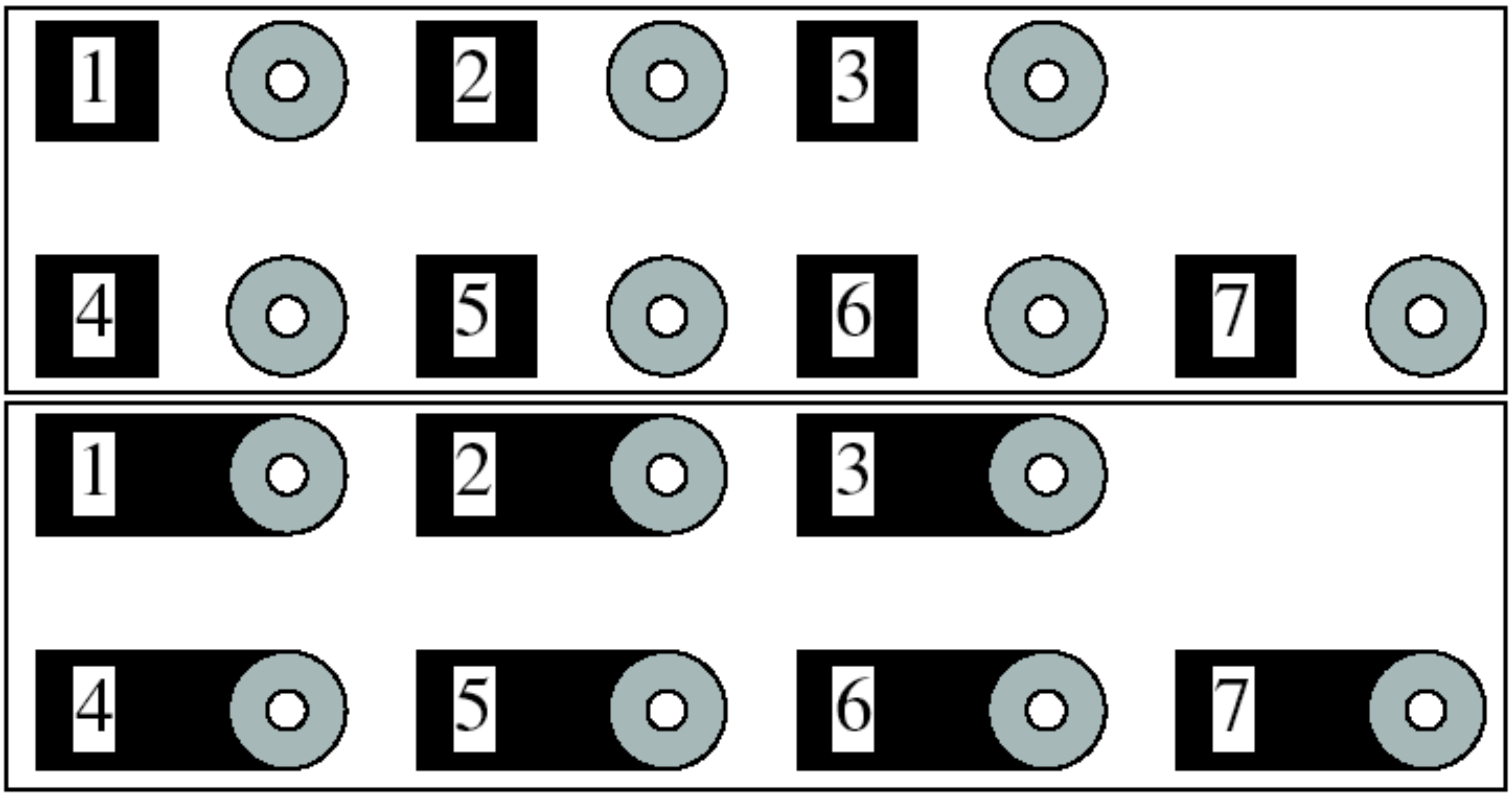


Grouping phenomena in real life



A Berkeley elevator example from D.A. Forsyth

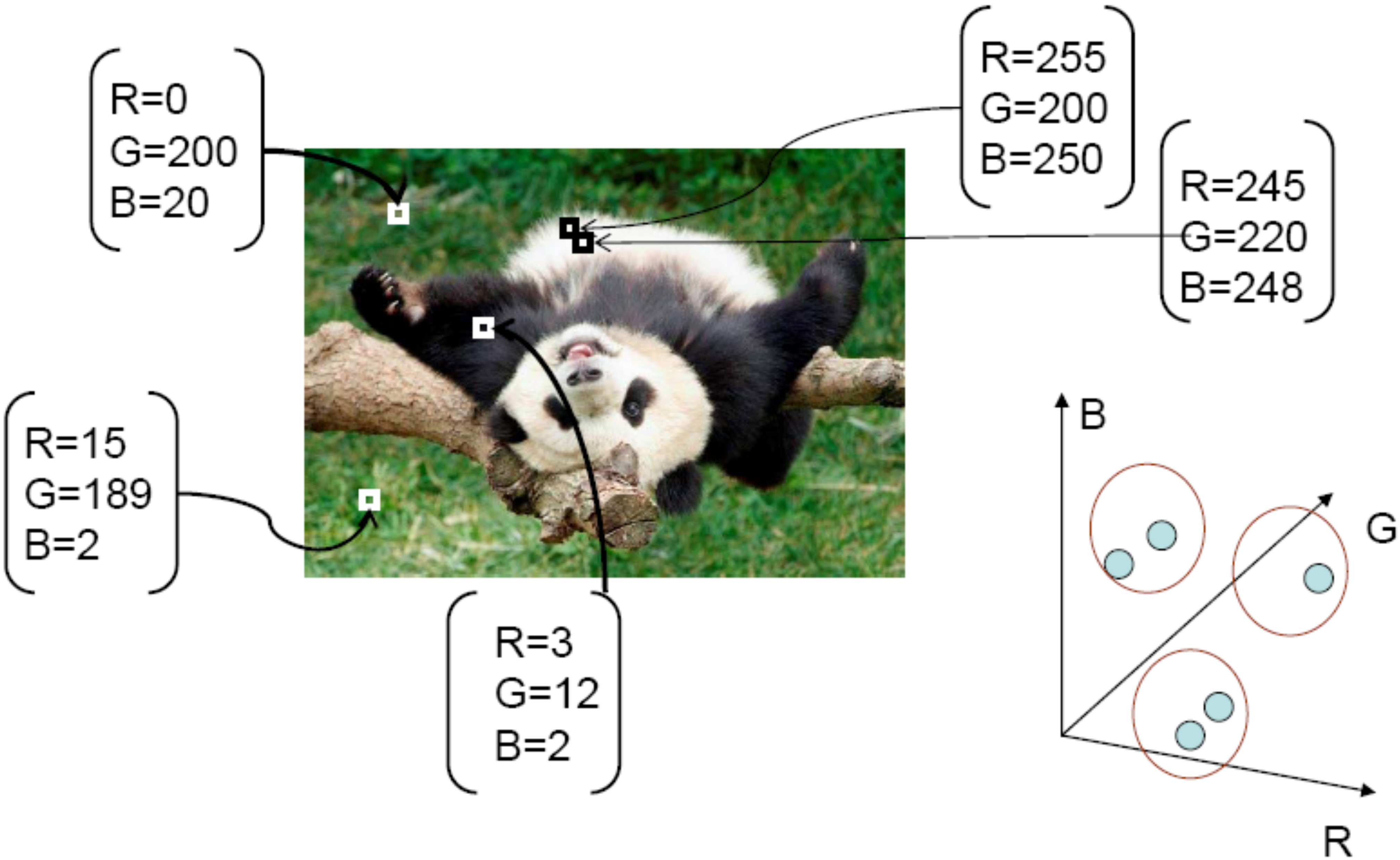
Grouping phenomena in real life



A Berkeley elevator example from D.A. Forsyth

Segmentation as clustering

- Cluster similar pixels (features) together



Source: K. Grauman

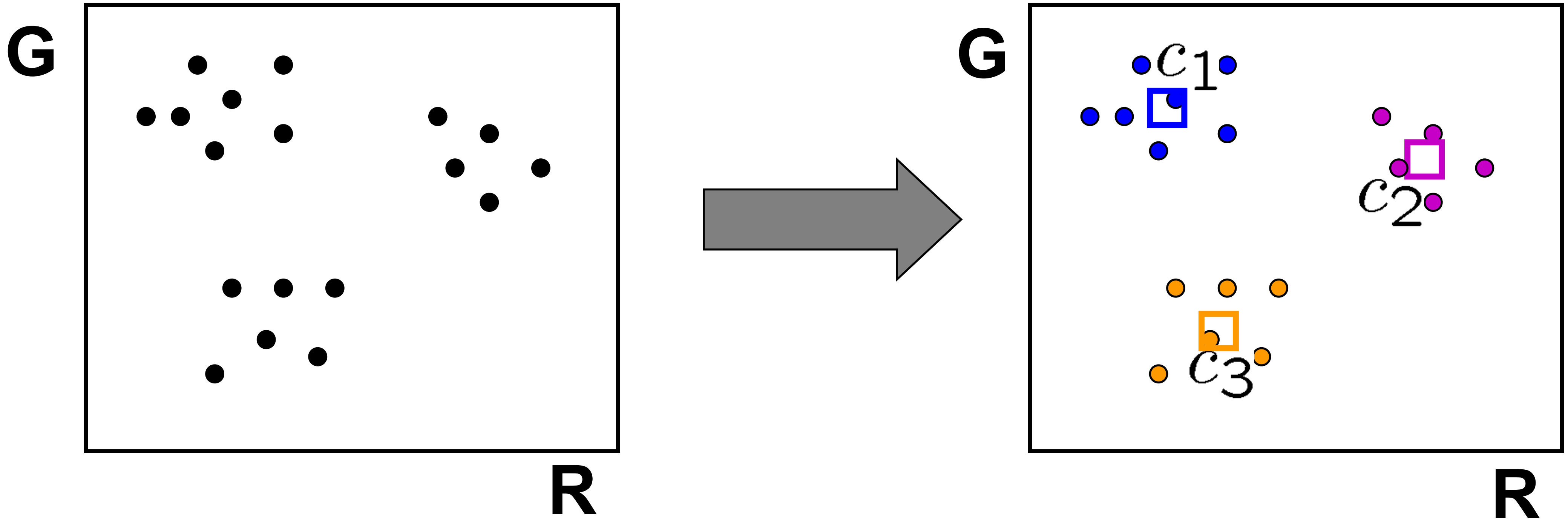
Segmentation as clustering

- Simplest methods:
- Agglomerative clustering (Merge):
  - grouping stuff that belongs together, or
  - iteratively merging the closest clusters
- Divisive clustering (Split):
  - split clusters recursively, or
  - iteratively split the cluster that yields the most diverse cluster
- Split and merge

Clustering

How to choose the representative colors?

– This is a clustering problem!



Objective

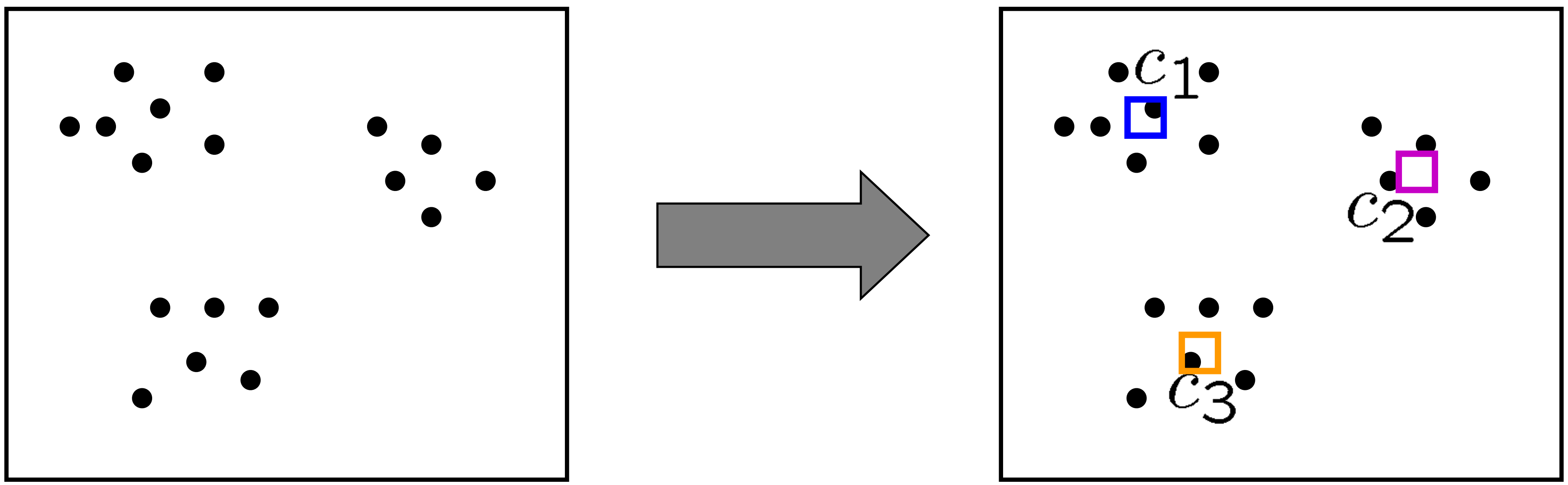
- **Each point should be as close as possible to a cluster center**
  - **Minimize sum squared distance of each point to closest center**

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

Solution: break it down into subproblems

Suppose I tell you the cluster centers c_i

- **Q: how to determine which points to associate with each c_i?**
- **A: for each point p, choose closest c_i**



Suppose I tell you the points in each cluster

- **Q: how to determine the cluster centers?**
- **A: choose c_i to be the mean of all points in the cluster**

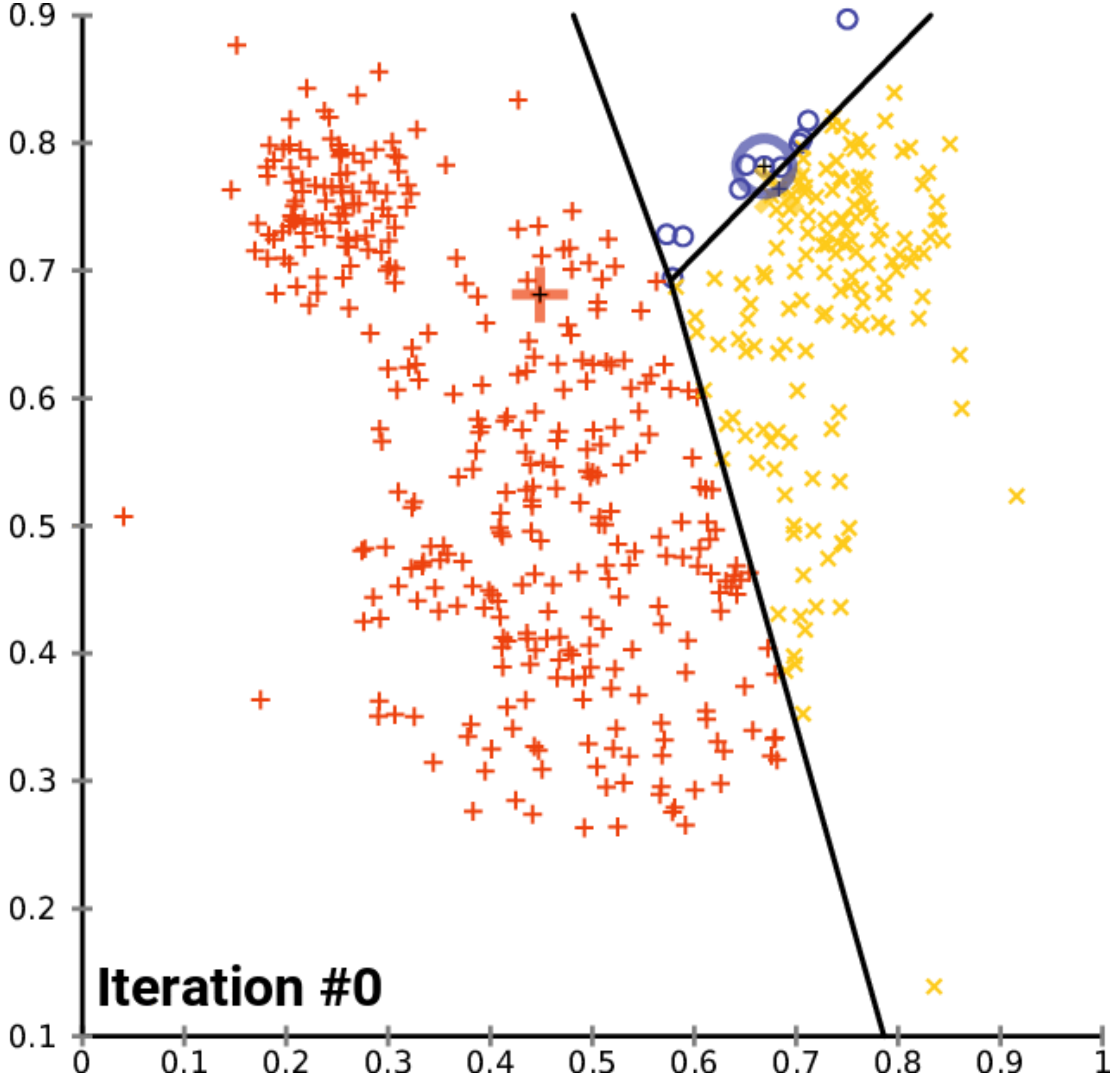
K-means clustering

- 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 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:

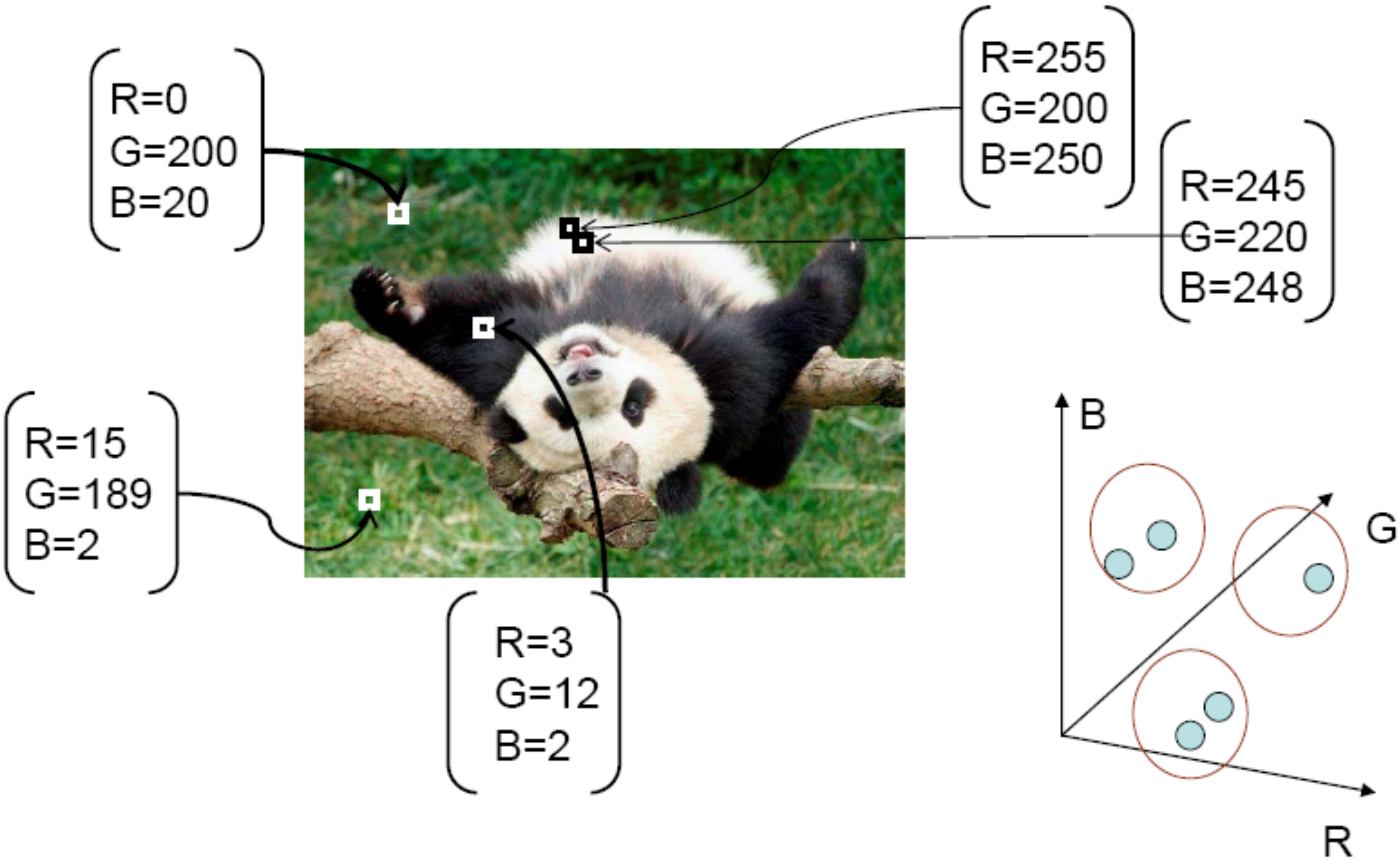
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

K-means



Segmentation as clustering

- Cluster similar pixels (features) together

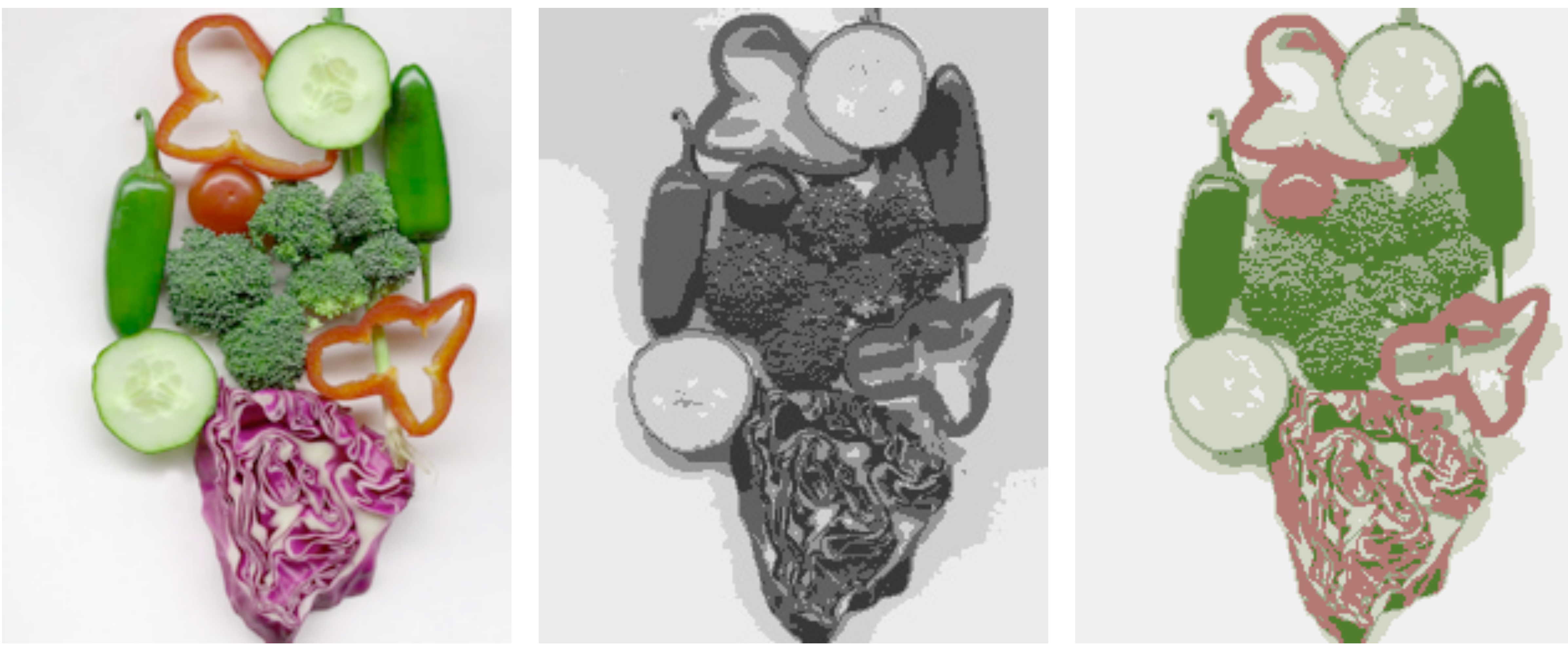


Source: K. Grauman

Segmentation as clustering

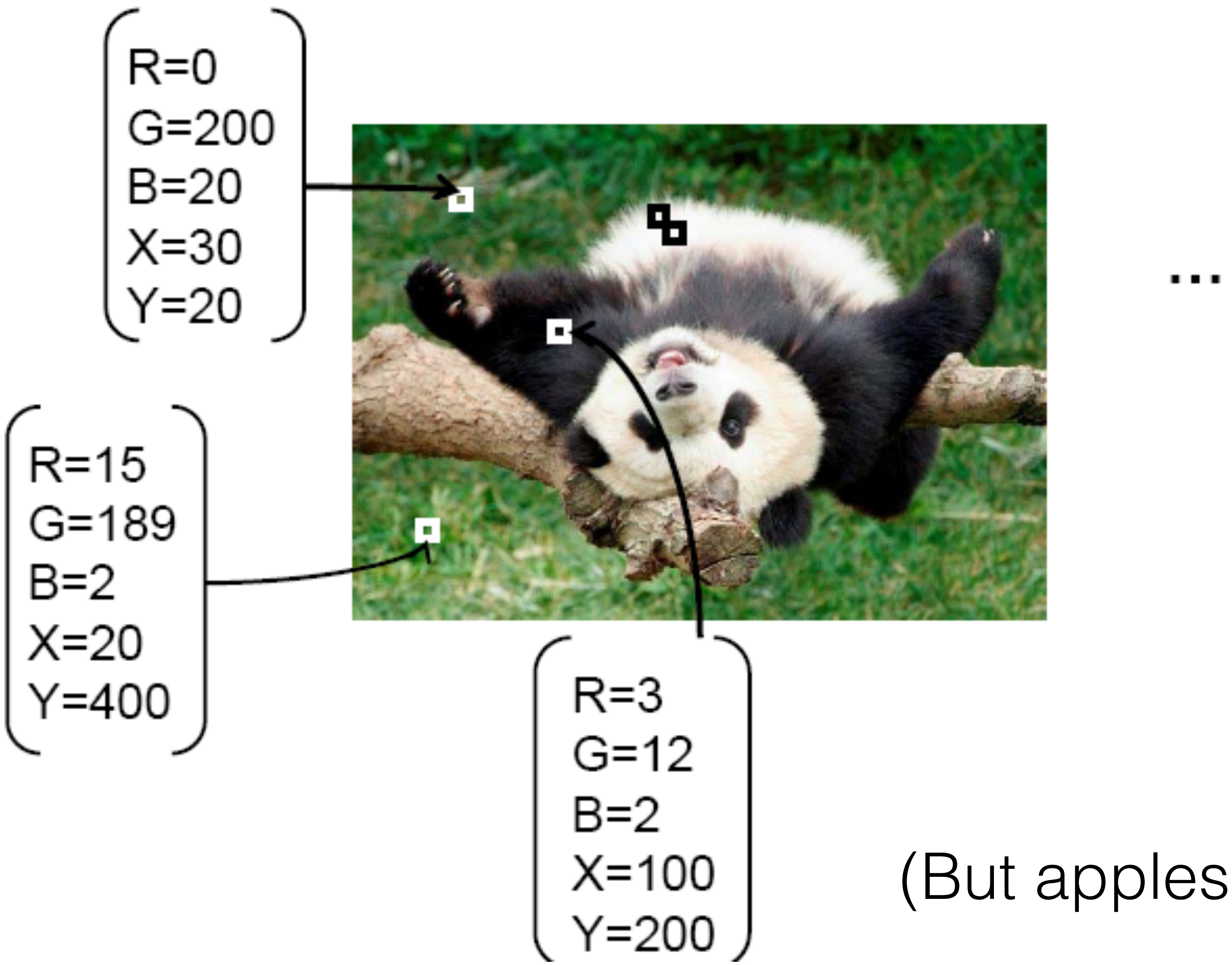
- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent

Image | Intensity-based clusters | Color-based clusters



Segmentation as clustering

- Cluster similar pixels (features) together

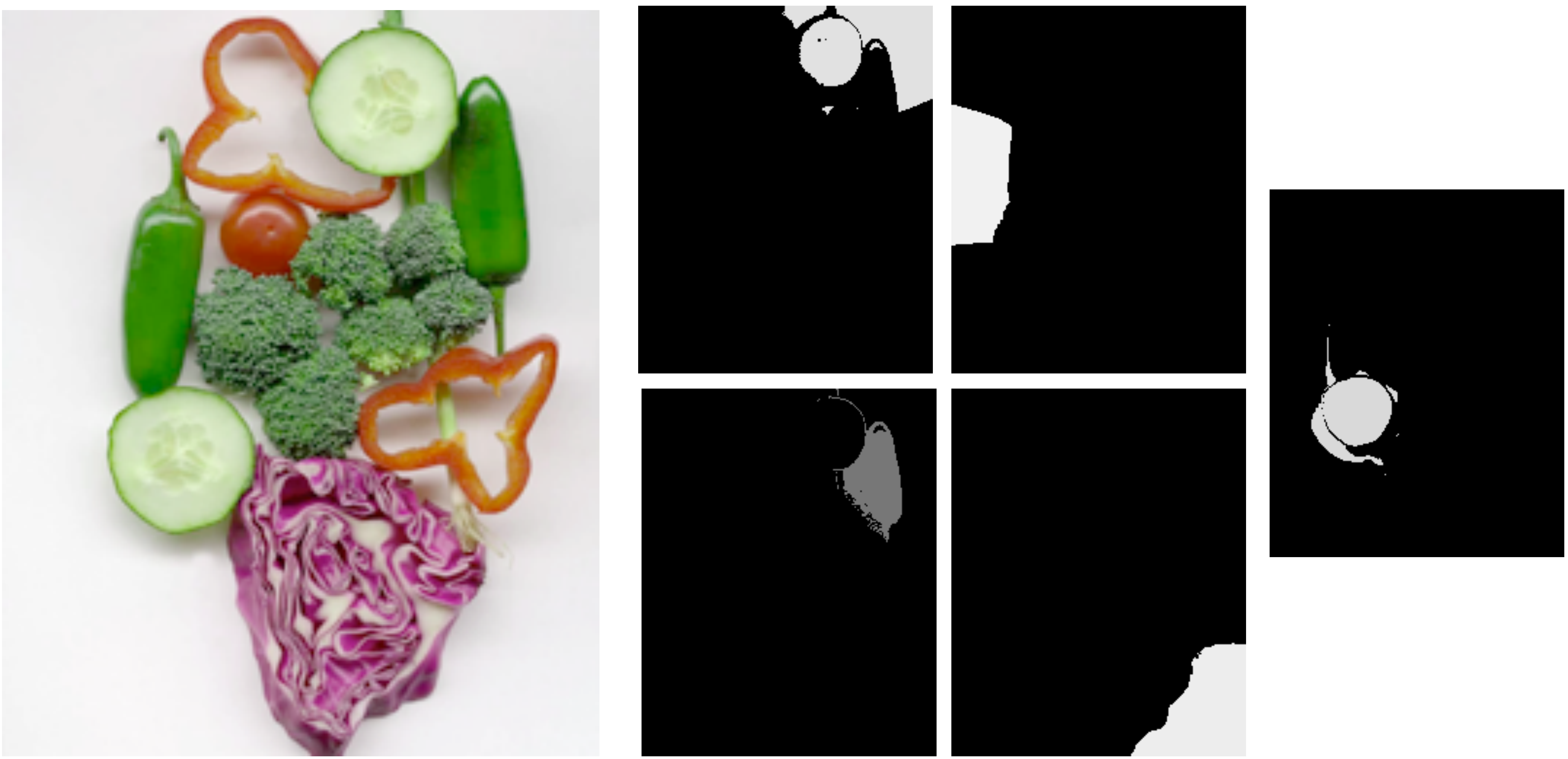


(But apples and oranges)

Source: K. Grauman

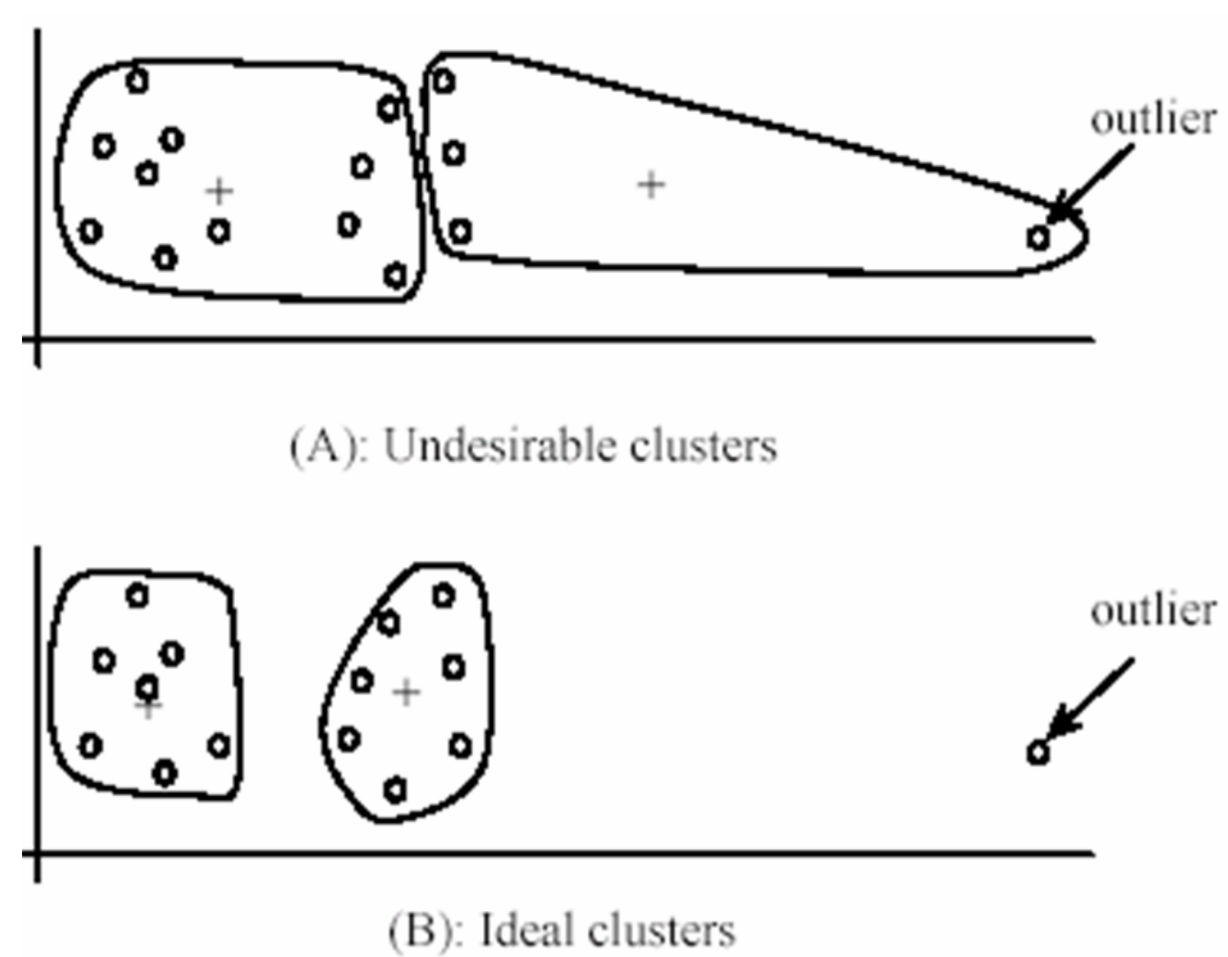
Segmentation as clustering

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



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



Probabilistic clustering

Basic questions

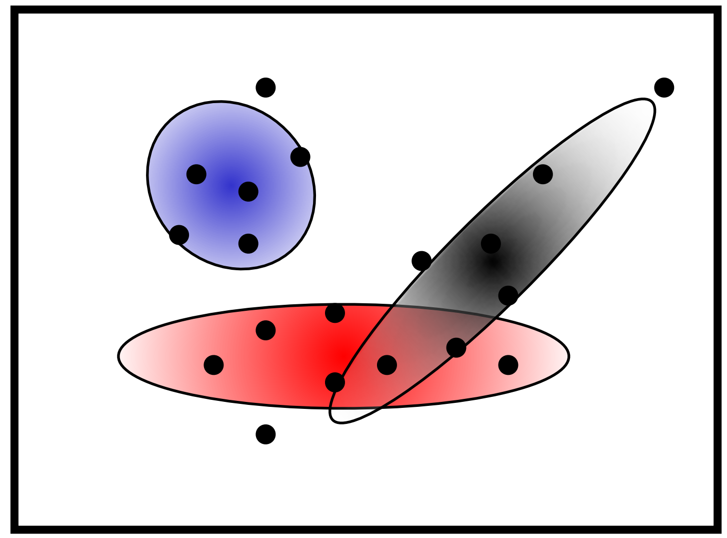
- what is the probability that a point **x** is in cluster m?
- what is the shape of each cluster?

K-means doesn't answer these questions

Basic idea

- instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function
- This function is called a **generative model**
  - defined by a vector of parameters $\boldsymbol{\theta}$

Mixture of Gaussians



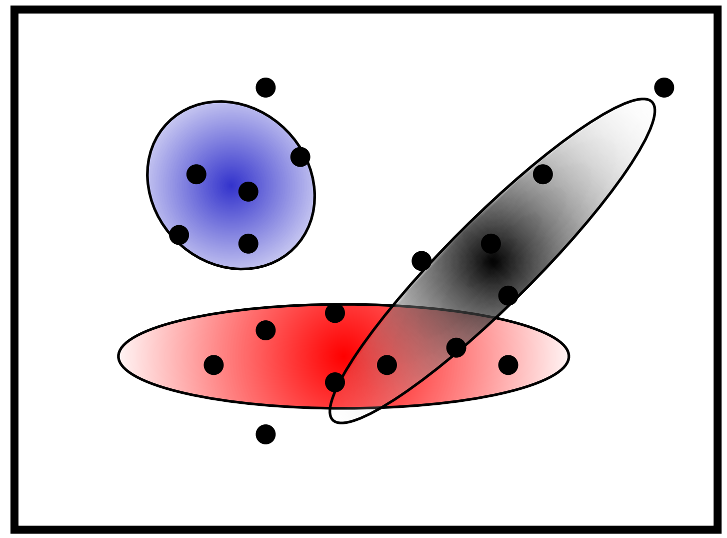
One generative model is a mixture of Gaussians (MOG)

- K Gaussian blobs with means $\boldsymbol{\mu}_{\mathbf{b}}$ covariance matrices $\mathbf{V}_{\mathbf{b}}$, dimension d
  - blob *b* defined by: $$P\left(x \mid \mu_b, V_b\right) = \frac{1}{\sqrt{(2\pi)^d \left| V_b \right|}} e^{-\frac{1}{2}\left(x - \mu_b\right)^T V_b^{-1}\left(x - \mu_b\right)}$$
- blob *b* is selected with probability α_b
- the likelihood of observing **x** is a weighted mixture of Gaussians

$$P(x \mid \theta) = \sum_{b=1}^{K} \alpha_b P(x \mid \mu_b, V_b)$$

where $\theta = [\alpha_1, \ldots, \alpha_K, \mu_1, \ldots, \mu_K, V_1, \ldots, V_k]$

Expectation maximization (EM)



Goal

- find blob parameters θ that maximize the likelihood function:

$$P(data|\theta) = \prod_x P(x|\theta)$$

Approach:

1. E step: given current guess of blobs, compute ownership of each point
2. M step: given ownership probabilities, update blobs to maximize likelihood function
3. repeat until convergence

EM details

E-step

– compute probability that point **x** is in blob b, given current guess of **θ**

$P(b)$

$$\underbrace{P(b|x,\mu_b,V_b)}_{P(b|x)} = \frac{\overbrace{\alpha_b}^{P(b)}\overbrace{P(x|\mu_b,V_b)}^{P(x|b)}}{\underbrace{\sum_{i=1}^{K}\alpha_i P(x|\mu_i,V_i)}_{P(x)}}$$

EM details

E-step

- compute probability that point **x** is in blob b, given current guess of **θ**

$$P(b|x, \mu_b, V_b) = \frac{\alpha_b P(x|\mu_b, V_b)}{\sum_{i=1}^{K} \alpha_i P(x|\mu_i, V_i)}$$

M-step (maximize the log likelihood)

- compute probability that blob b is selected

$$\alpha_b^{new} = \frac{1}{N} \sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)$$

N data points

- mean of blob b

$$\mu_b^{new} = \frac{\sum_{i=1}^{N} x_i P(b|x_i, \mu_b, V_b)}{\sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)}$$

- covariance of blob b

$$V_b^{new} = \frac{\sum_{i=1}^{N} (x_i - \mu_b^{new})(x_i - \mu_b^{new})^T P(b|x_i, \mu_b, V_b)}{\sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)}$$

Applications of EM

Turns out this is useful for all sorts of problems

- any clustering problem
- any model estimation problem   (probabilistic problem formulation)
- missing data problems
- finding outliers
- segmentation problems
  - segmentation based on color
  - segmentation based on motion
  - foreground/background separation
- ...

Problems with EM

1. Local minima

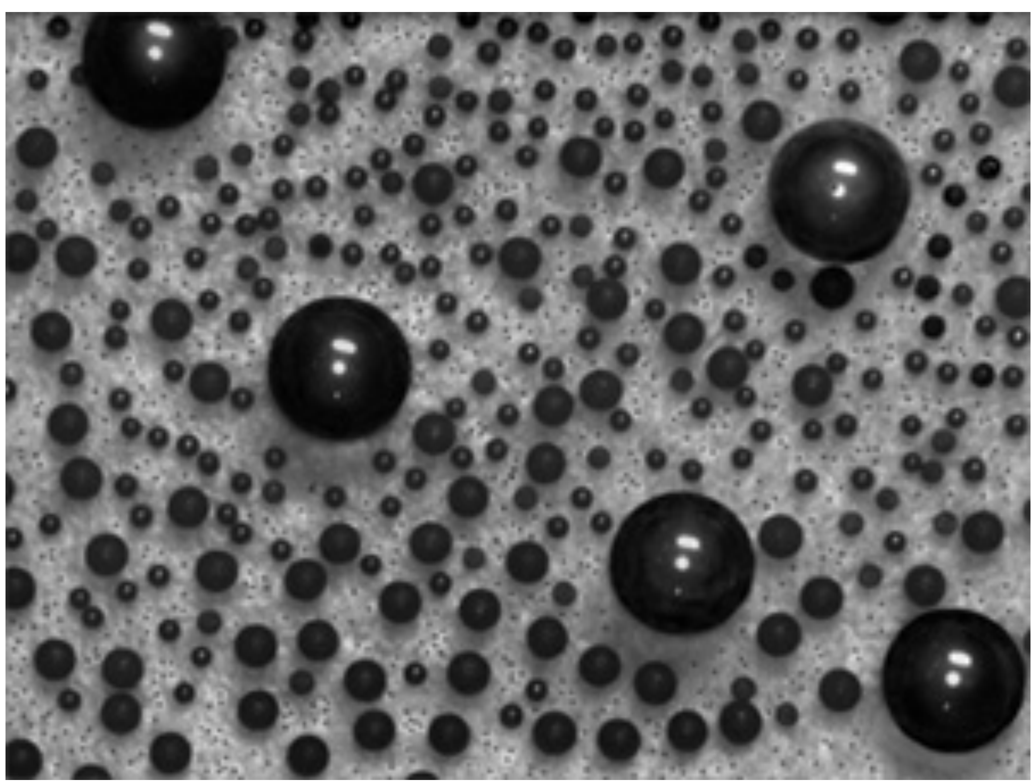
2. Need to know number of segments

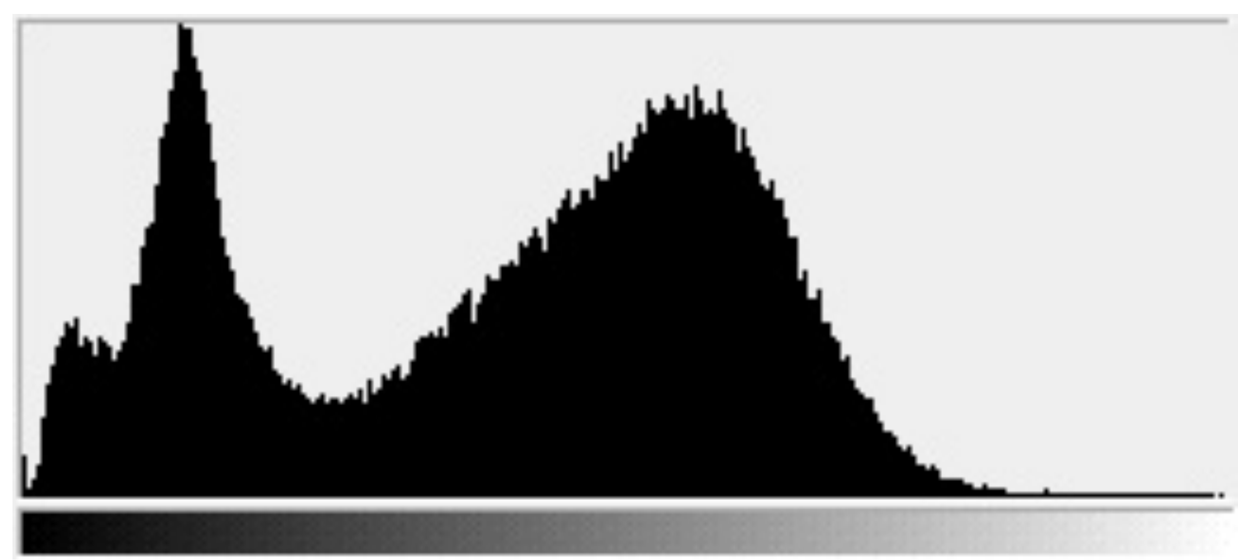
3. Need to assume generative model

Histogram-based segmentation

Goal

- Break the image into K regions (segments)
- Solve this by reducing the number of colors to K and mapping each pixel to the closest color

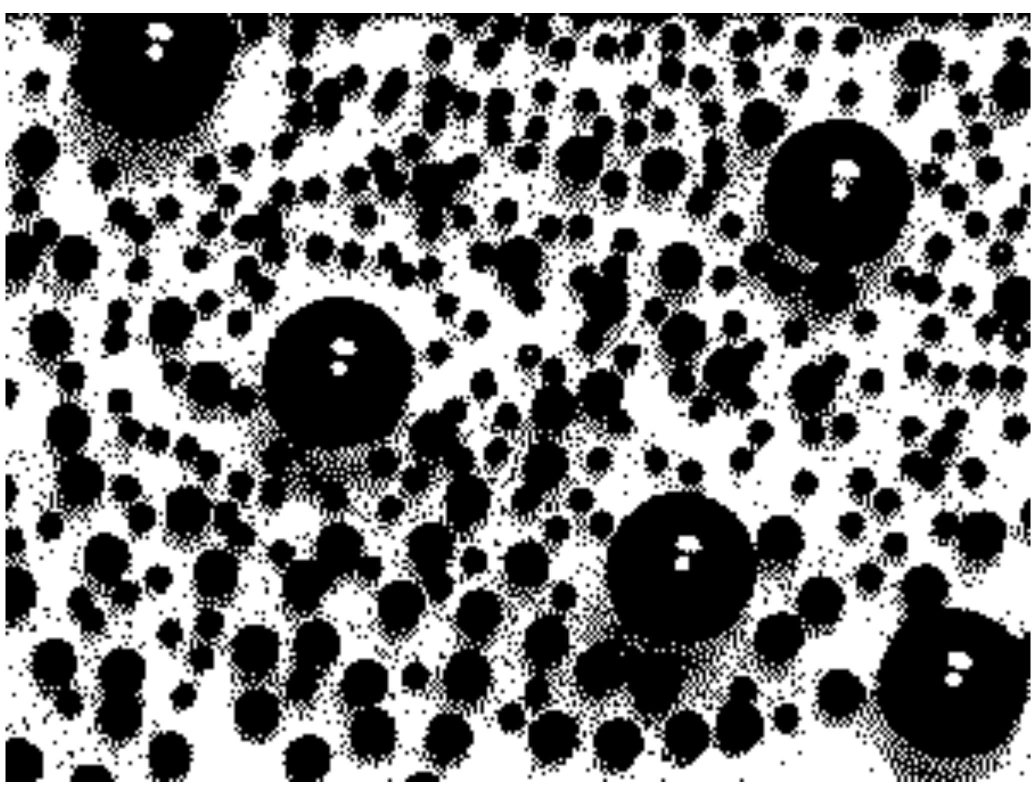


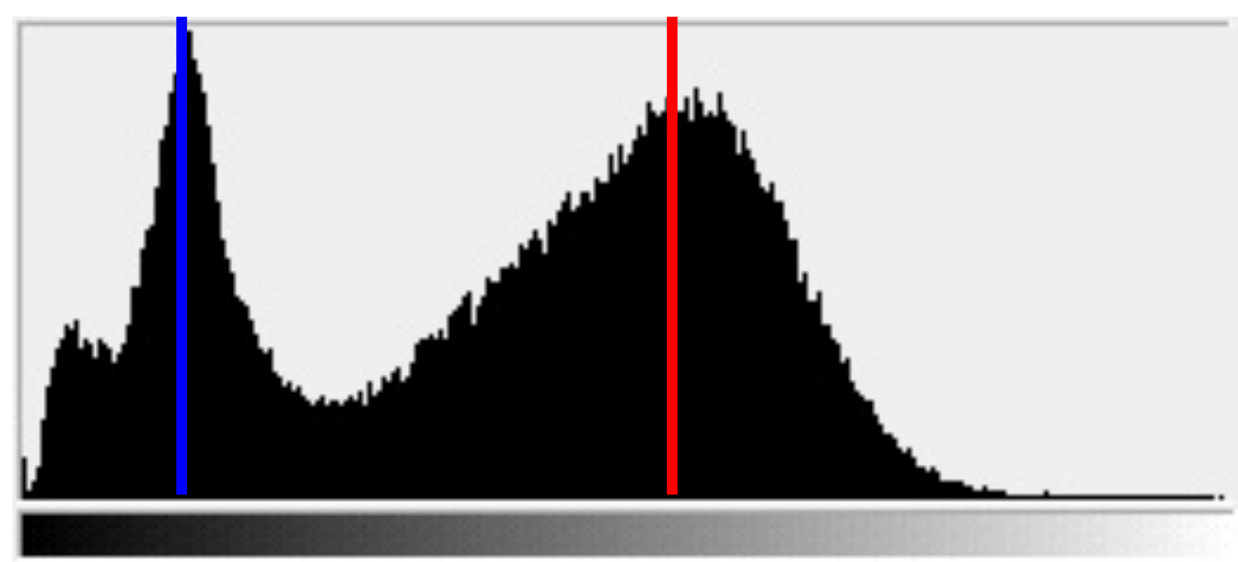


Histogram-based segmentation

Goal

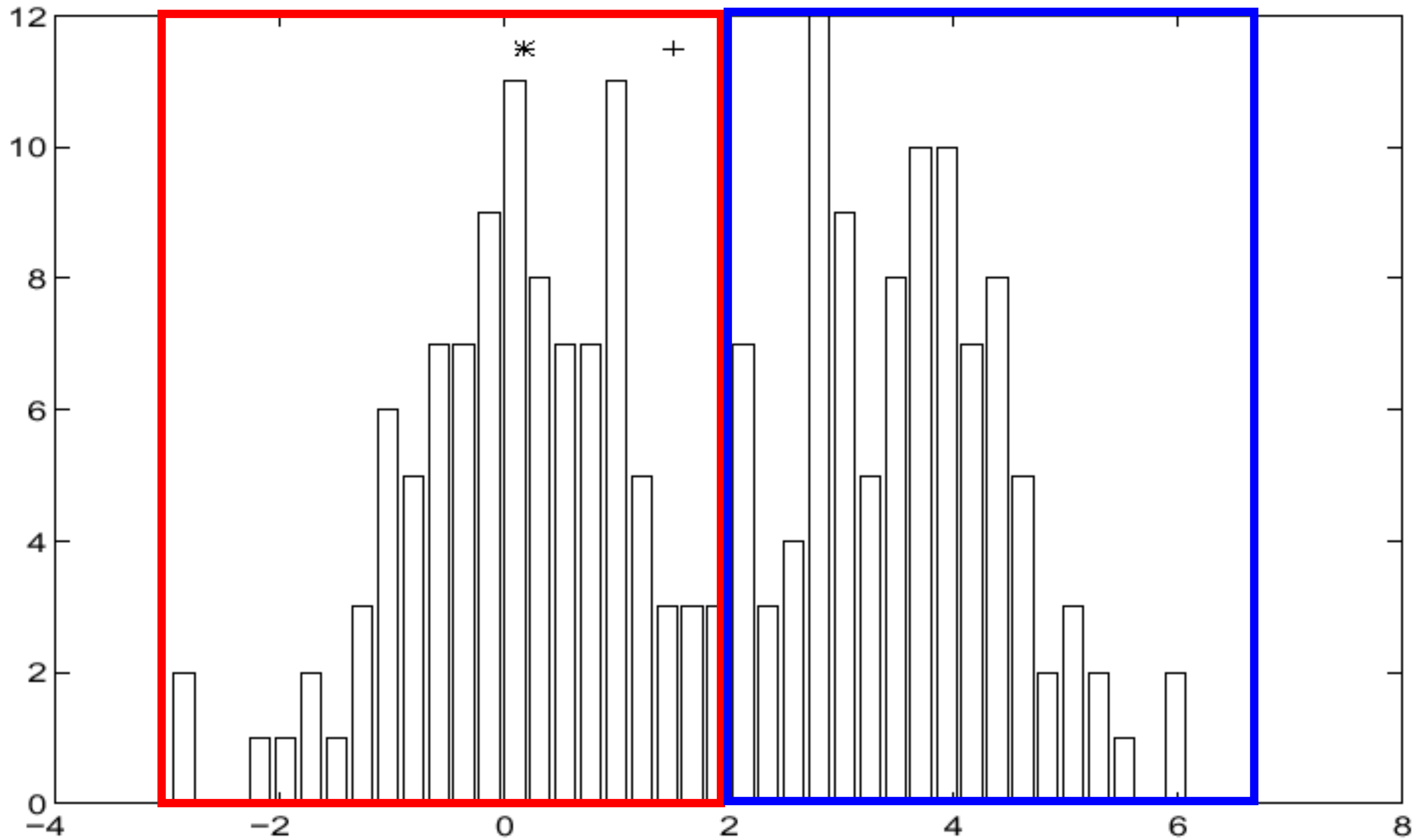
- Break the image into K regions (segments)
- Solve this by reducing the number of colors to K and mapping each pixel to the closest color





Here's what it looks like if we use two colors

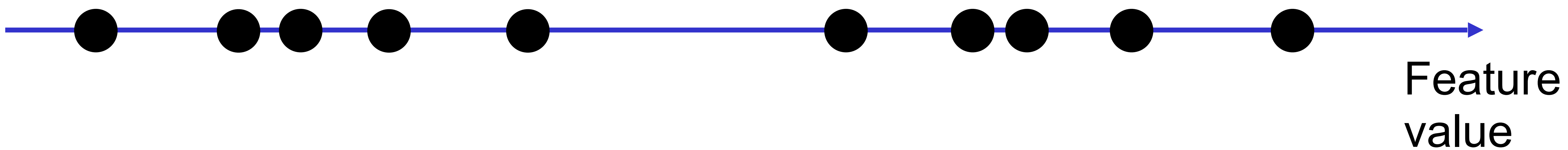
Finding Modes in a Histogram



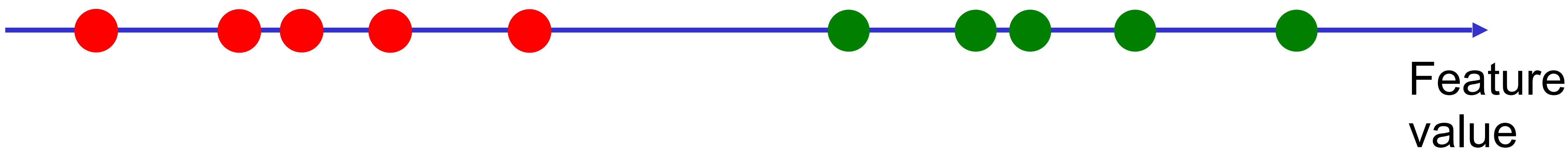
How Many Modes Are There? What are they? Which points belong with which modes?

– Easy to see, hard to compute

A 1-D Example

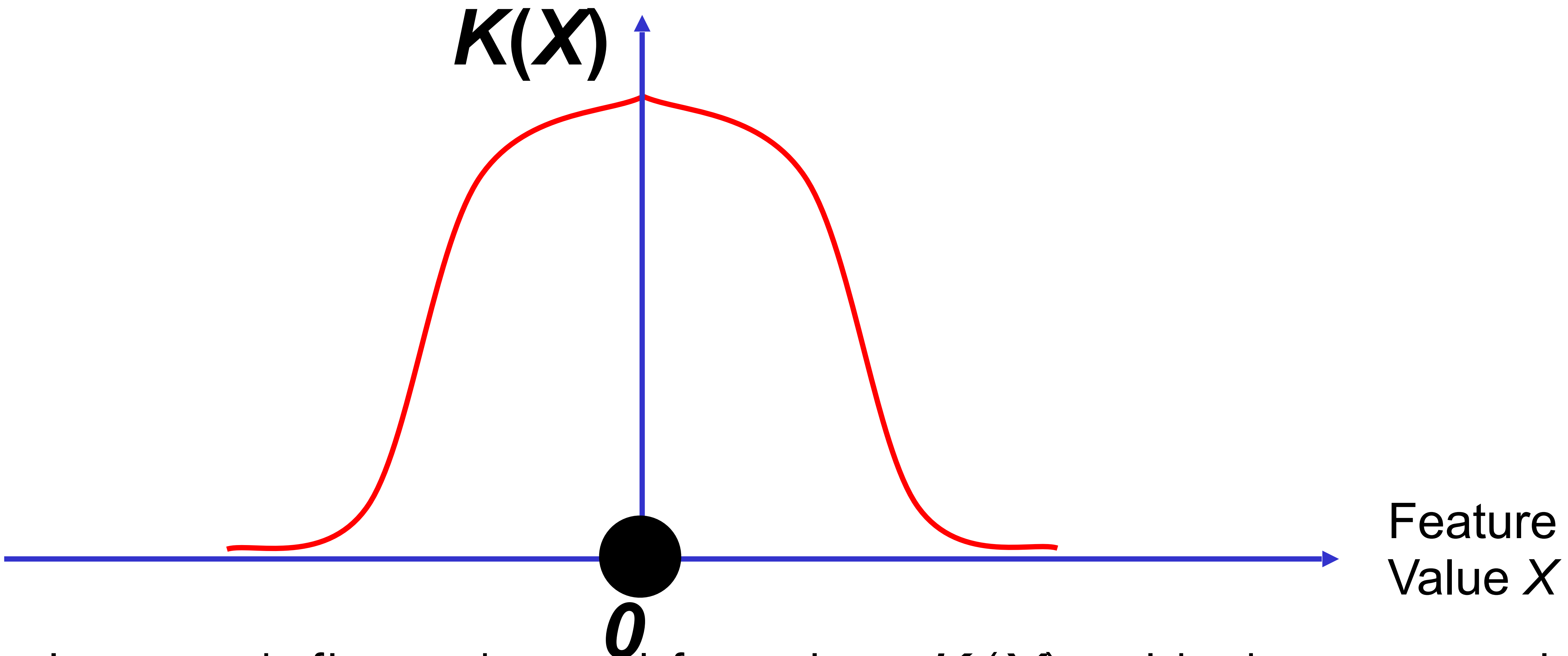


A 1-D Example



- “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*

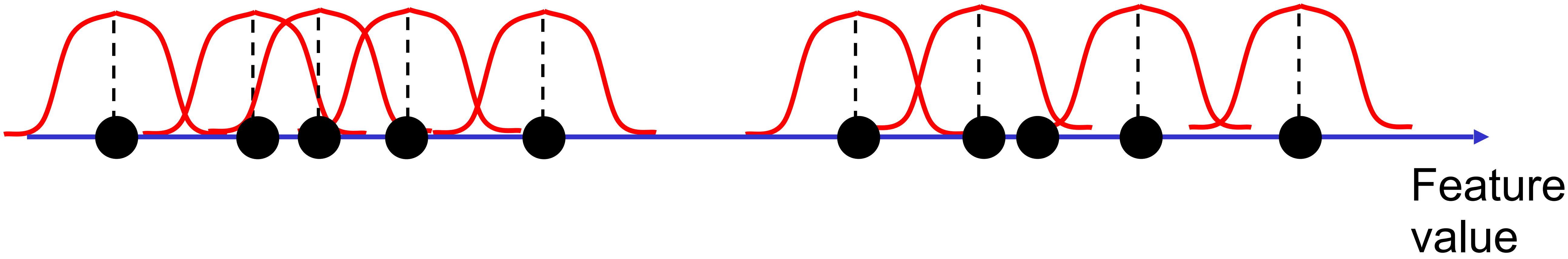
A 1-D Example



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

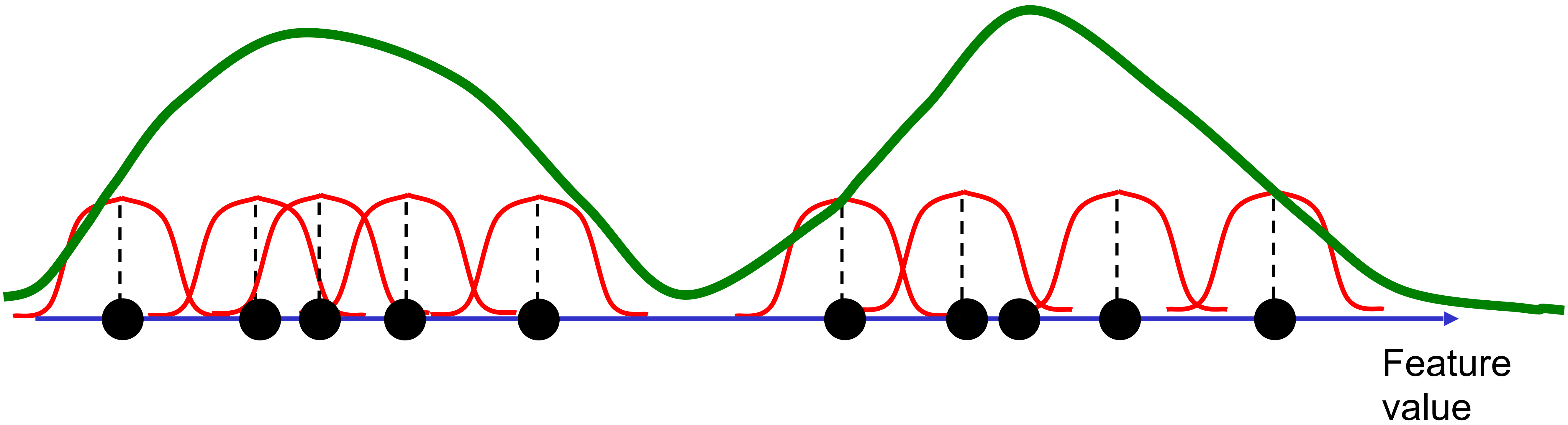


- We can define the kernel at each data point and average the result into a single function:

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

A 1-D Example

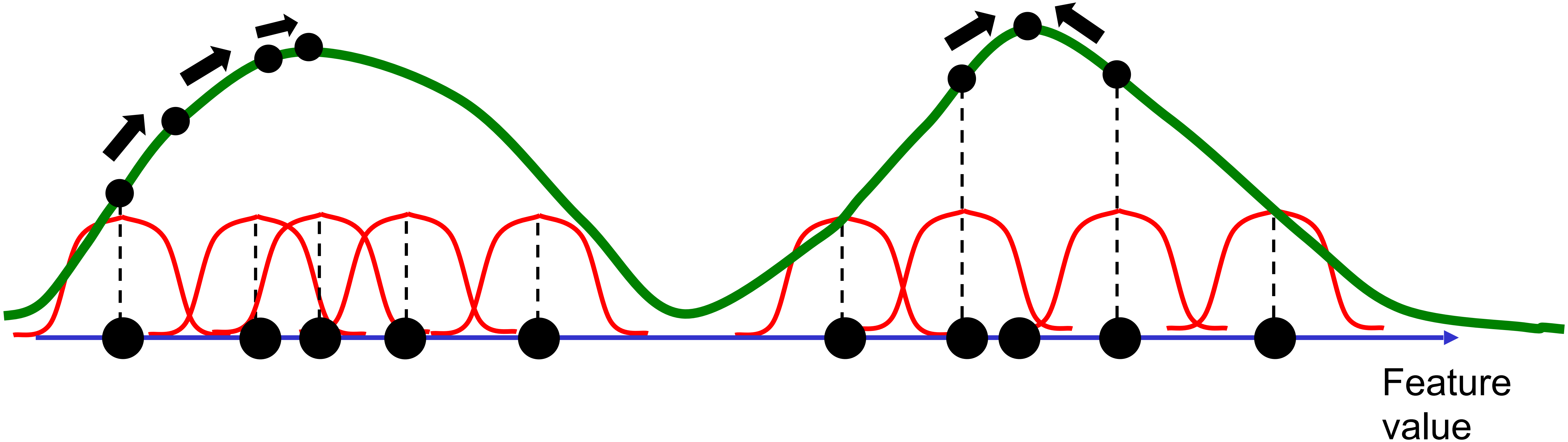
$$f(X) = \frac{1}{N}\sum_i K(X - X_i)$$



- *(Should be normalized to unit integral)*
- The maxima of ***f*** (the modes of the pdf) correspond to the clusters in the data

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?

Basic algorithm: gradient ascent

- For $i = 1,\ldots,N$

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

- Repeat

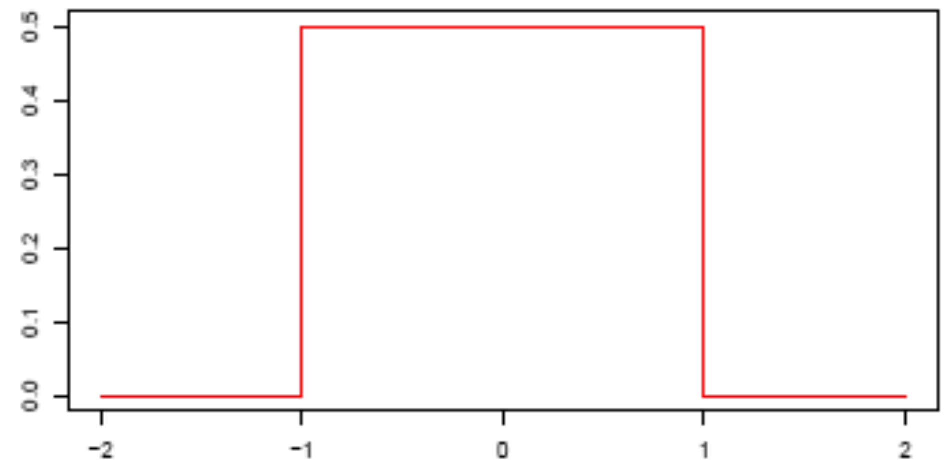
$$\boldsymbol{X} \leftarrow \boldsymbol{X}_i + \lambda \nabla \boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{X}_i + \frac{\lambda}{N} \sum_i \nabla \boldsymbol{K}\left(\boldsymbol{X} - \boldsymbol{X}_i\right)$$

until $\boldsymbol{X}$ does not change

Example kernels

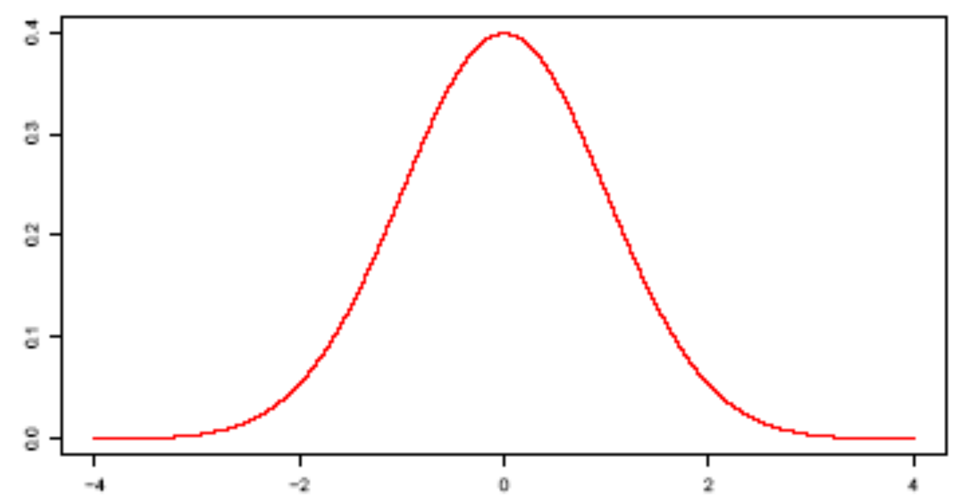
Uniform:

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



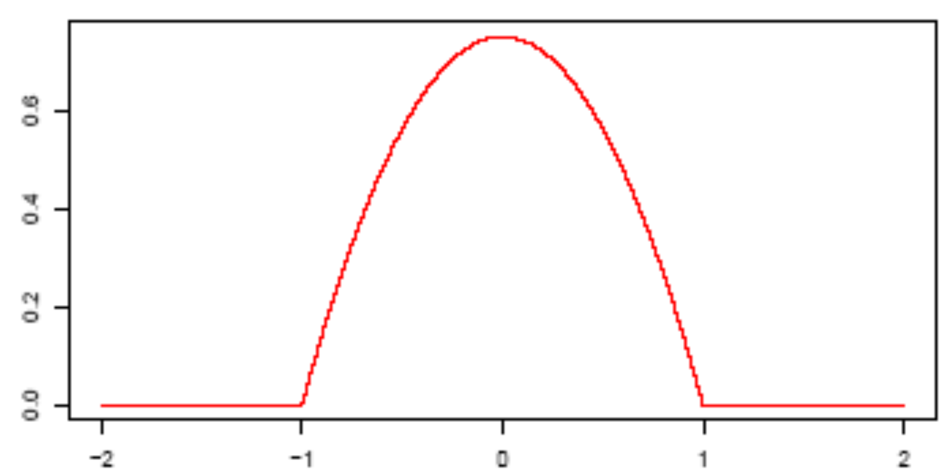
Gaussian:

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



Epanechnikov:

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



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

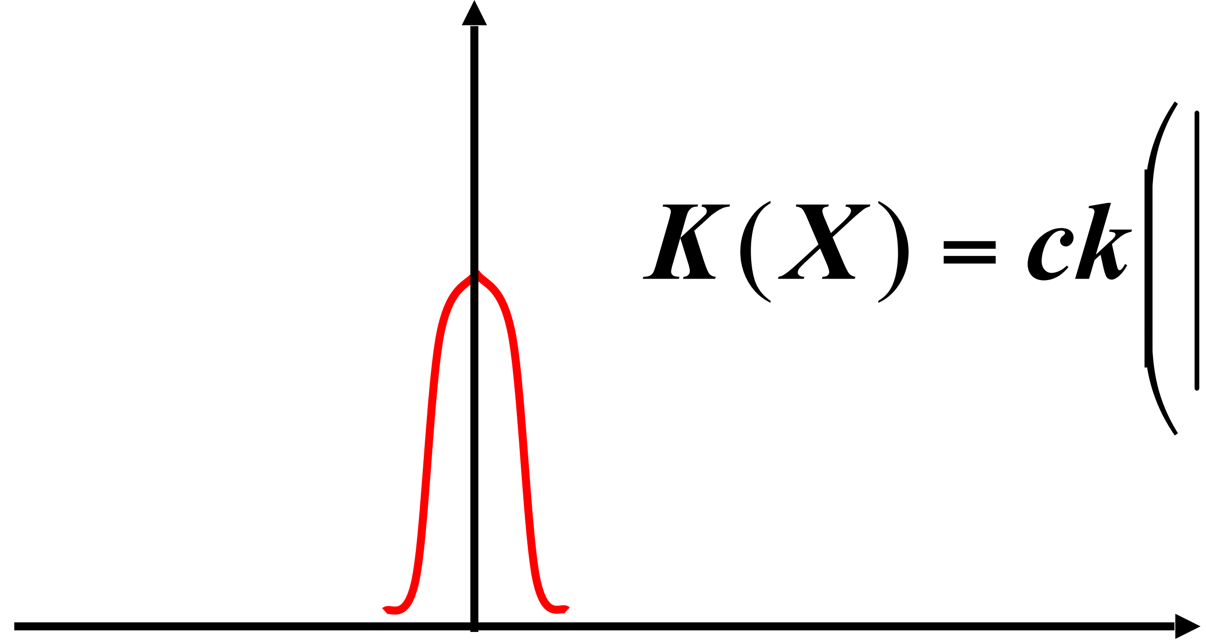
- Kernel is defined as:

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

- h is the bandwith of the kernel
- k is:

Bandwidth *h* controls the radius of influence of each data point.

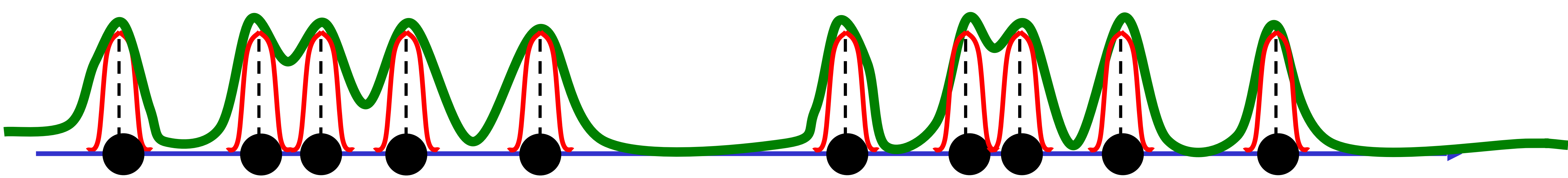
- Too small: Overfits the data points
- Too large: Smoothes out the details of the data



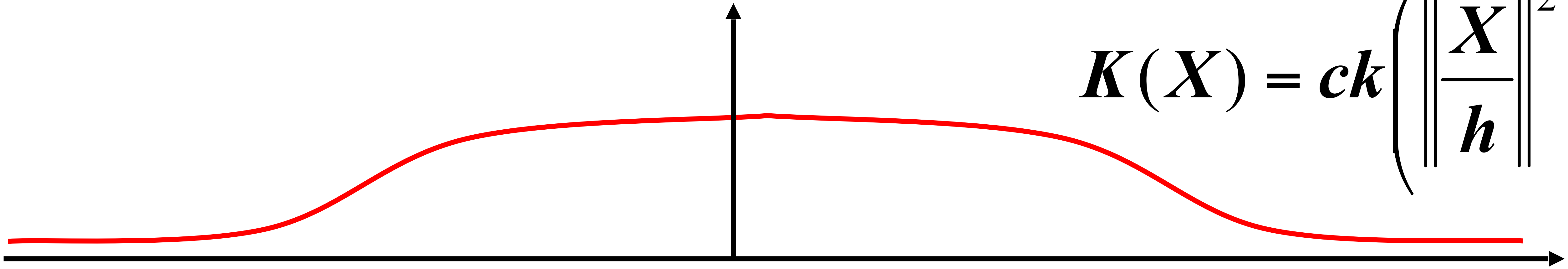
$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$

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

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

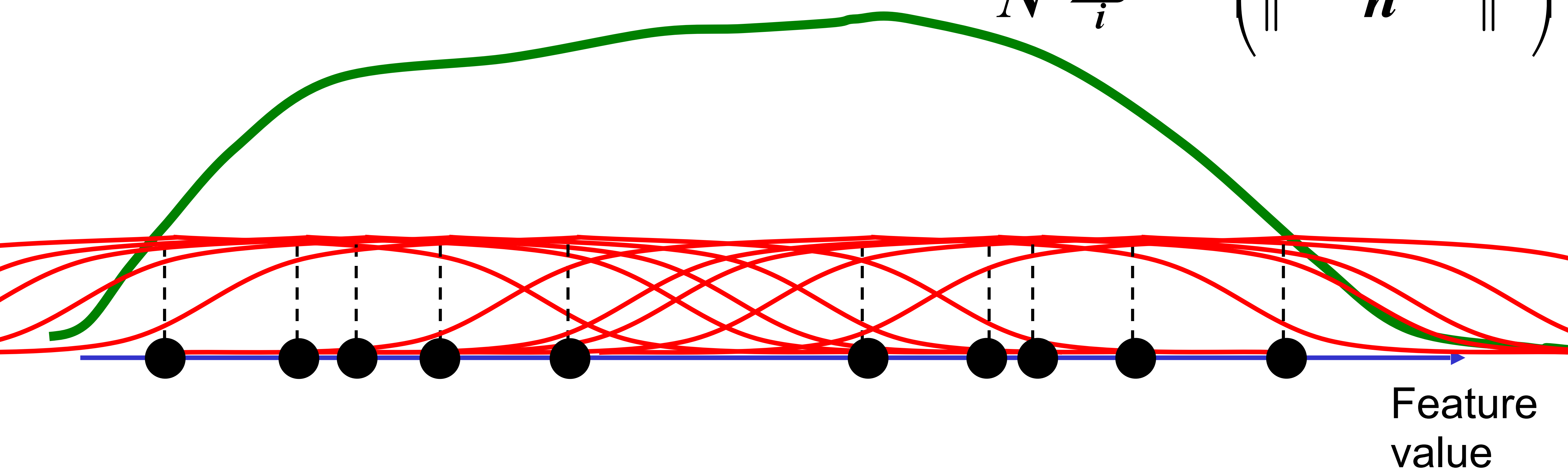


$$K(X) = ck\left(\left\|\frac{X}{h}\right\|^2\right)$$



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

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



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:

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

$$\Downarrow$$

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

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

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 (fixed point iterations for zero gradient):
  - Move *X* by the corresponding mean shift vector:

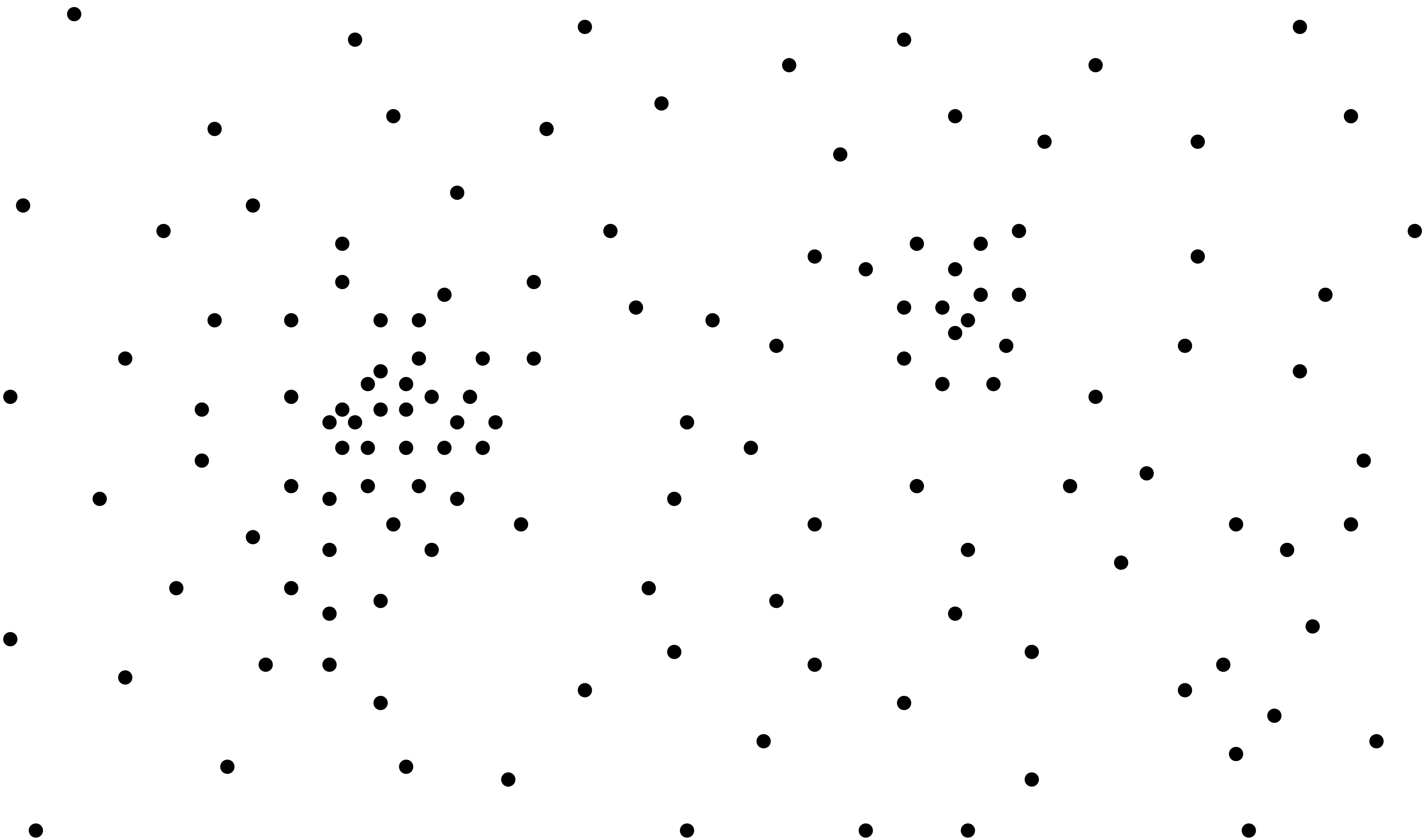
$$\boxed{\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{M}(\boldsymbol{X}) = \frac{\sum_i \boldsymbol{X}_i \boldsymbol{g}\left(\frac{\left\|\boldsymbol{X} - \boldsymbol{X}_i\right\|^2}{\boldsymbol{h}^2}\right)}{\sum_i \boldsymbol{g}\left(\frac{\left\|\boldsymbol{X} - \boldsymbol{X}_i\right\|^2}{\boldsymbol{h}^2}\right)}}$$

- Until *X* converges

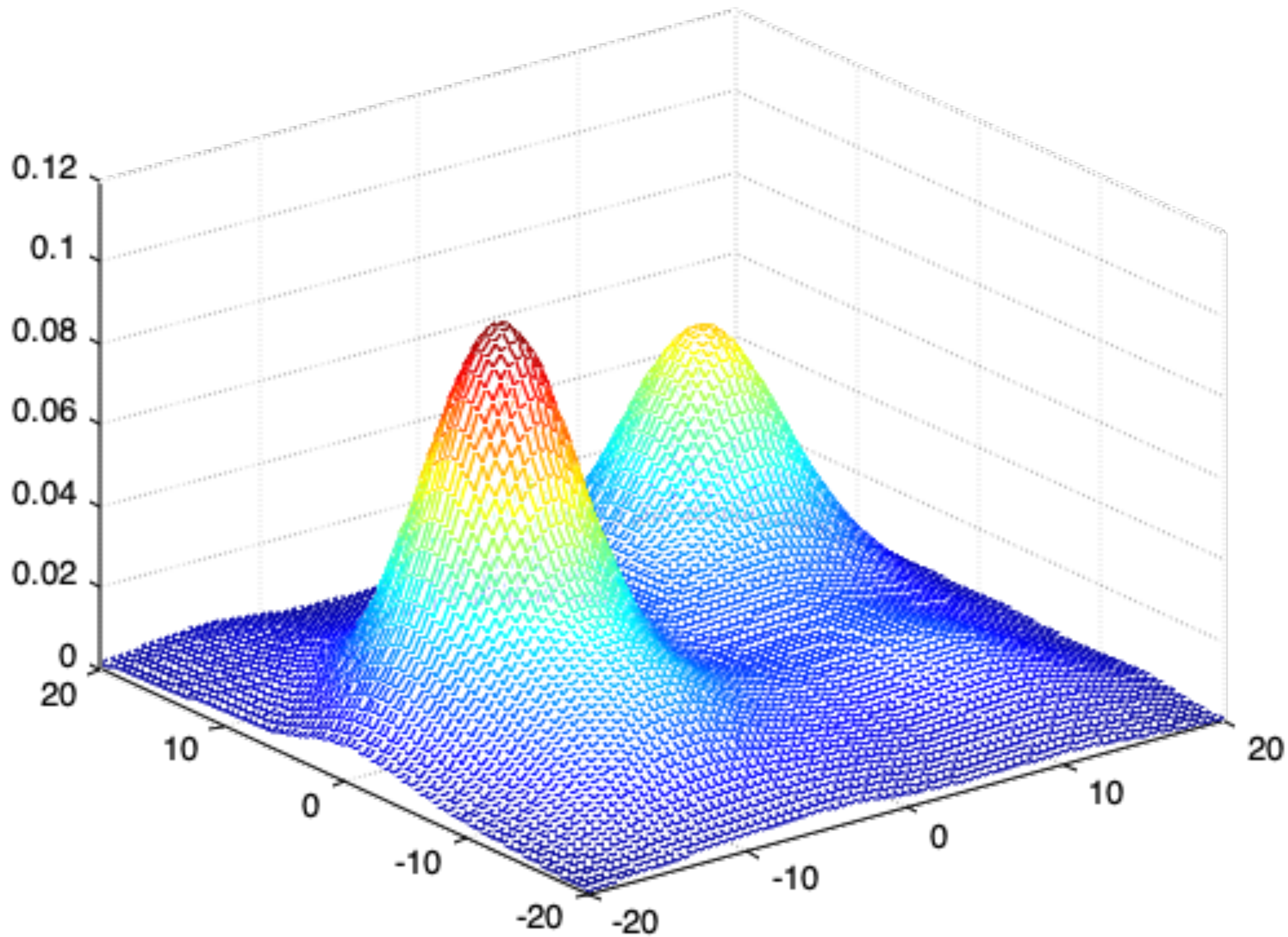
Note: Under mild conditions, convergence is guaranteed.

2-D Example

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

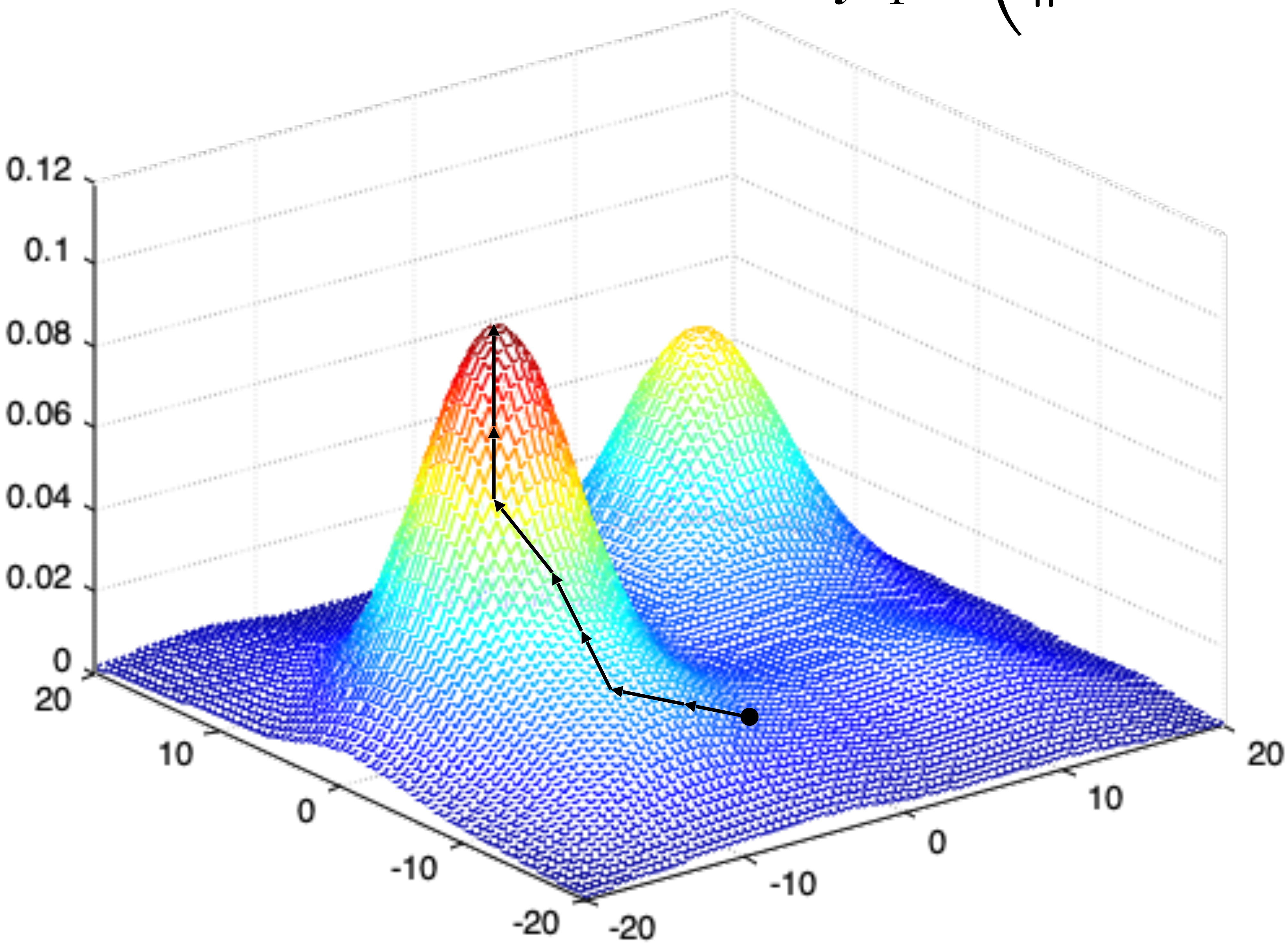


Estimated PDF: $$f(X) = \frac{c}{N}\sum_{i=1}^{N} k\left(\left\|\frac{X - X_i}{h}\right\|^2\right)$$

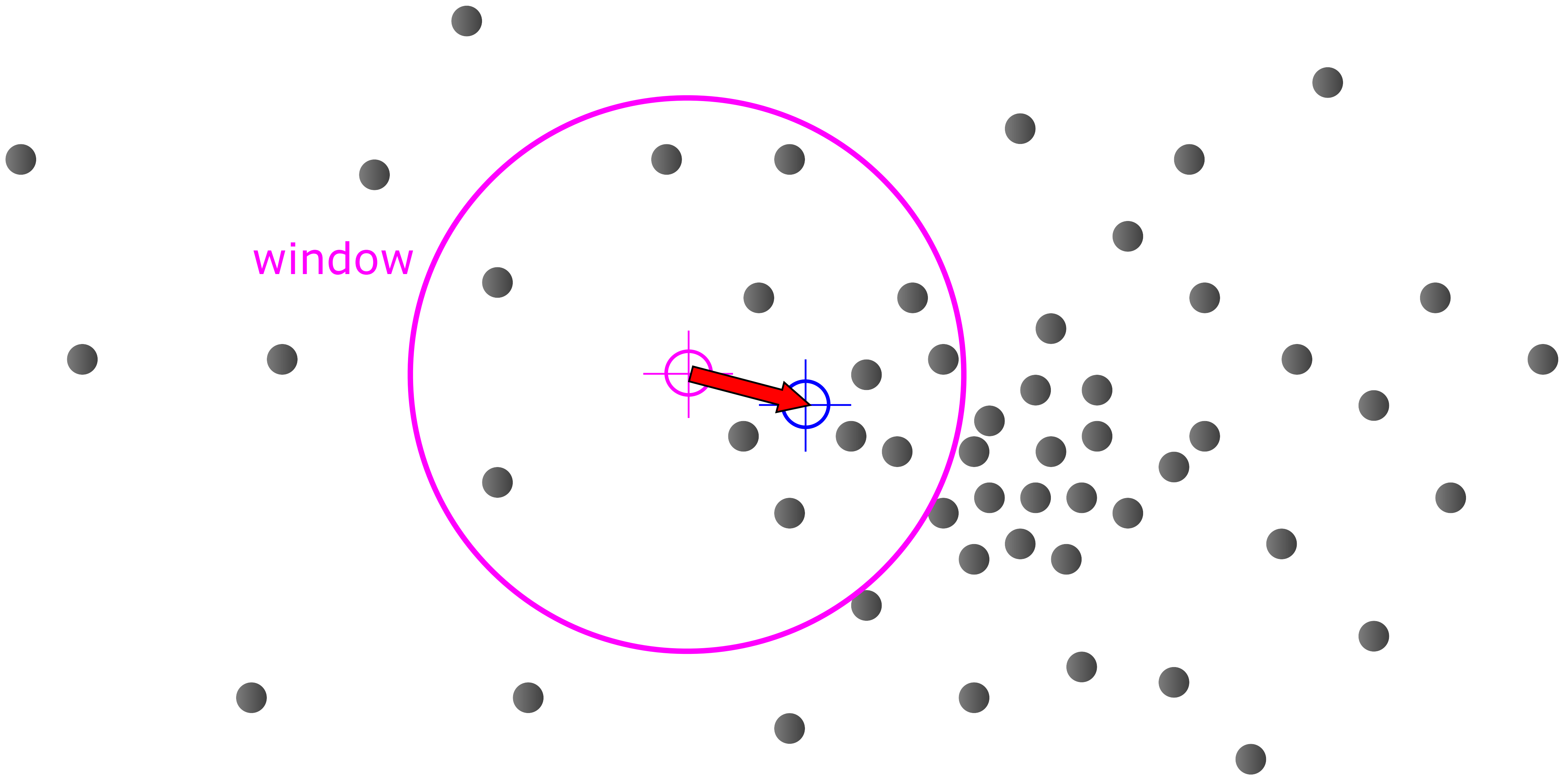


The trajectory of locations for finding modes

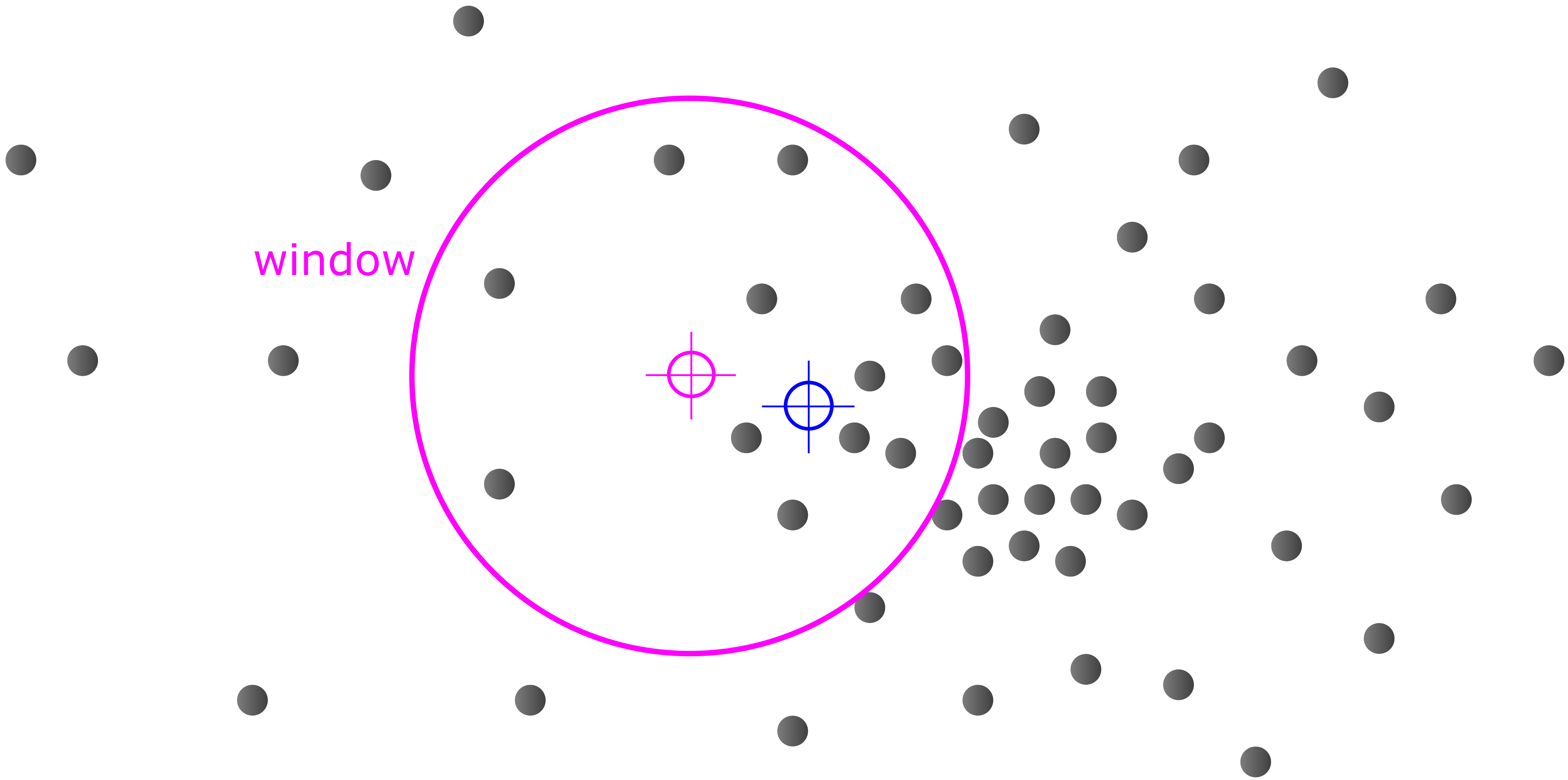
$$f(X) = \frac{c}{N} \sum_{i=1}^{N} k\left(\left\| \frac{X - X_i}{h} \right\|^2 \right)$$



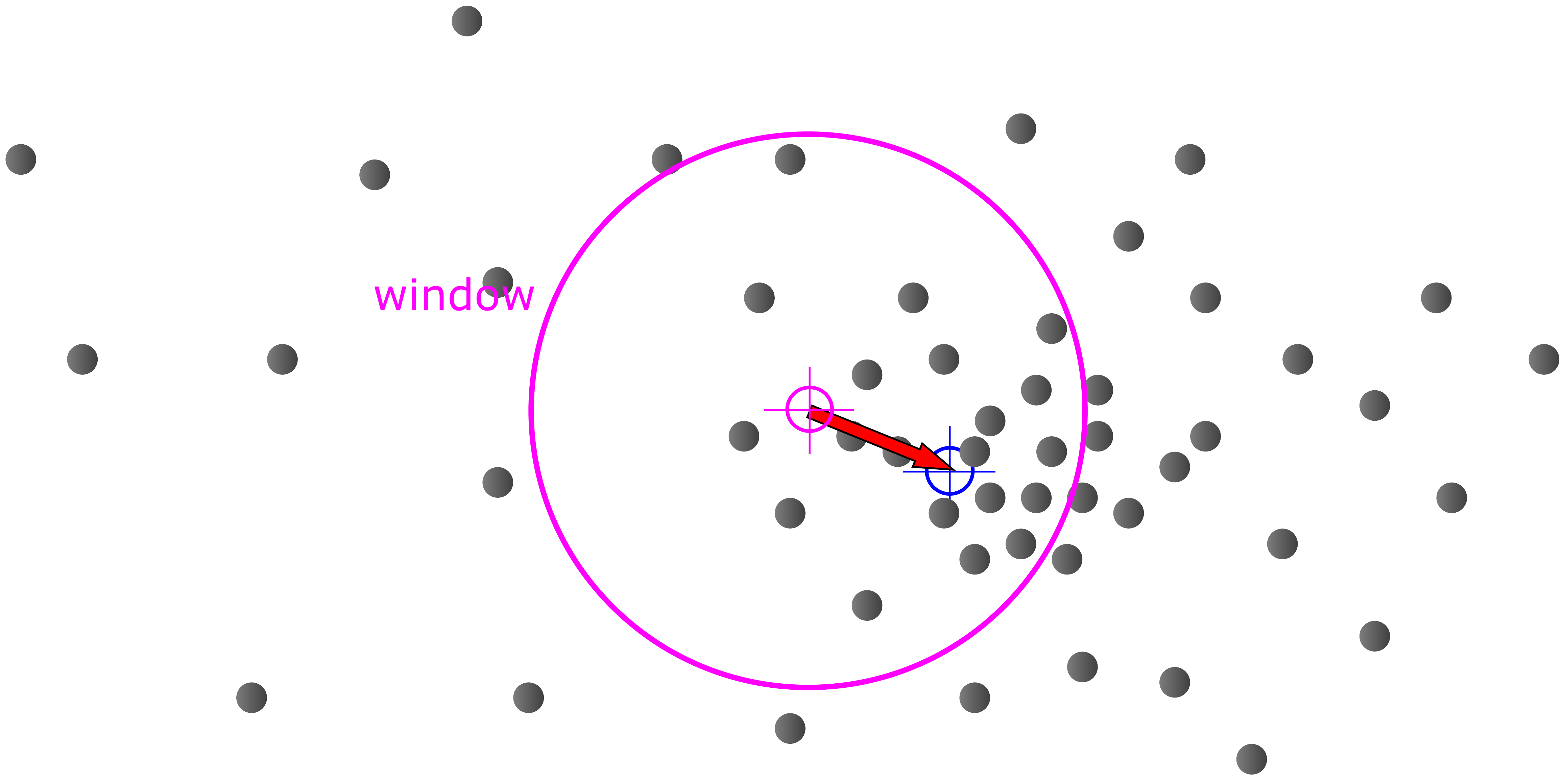
The Mean Shift Process



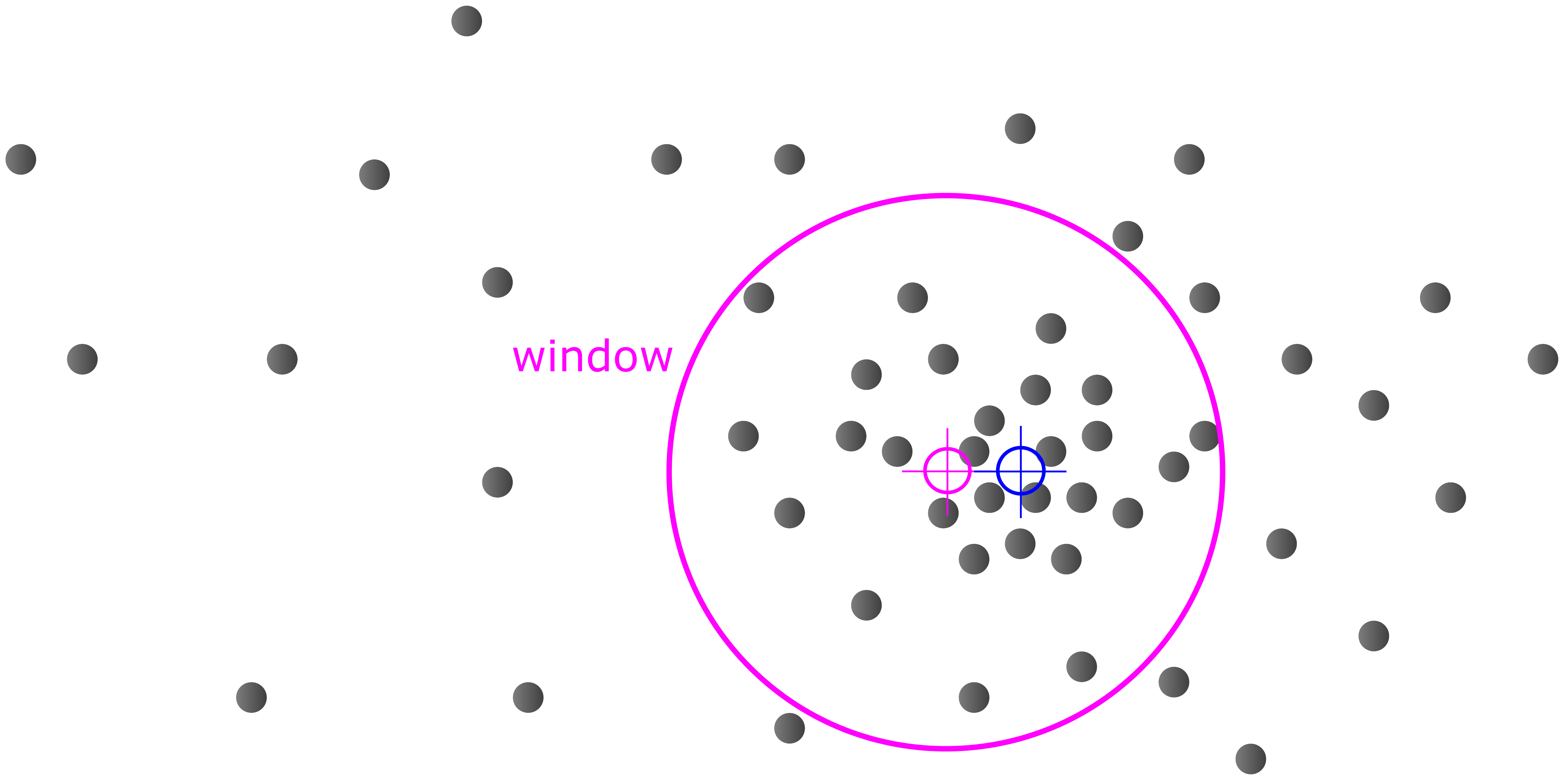
The Mean Shift Process



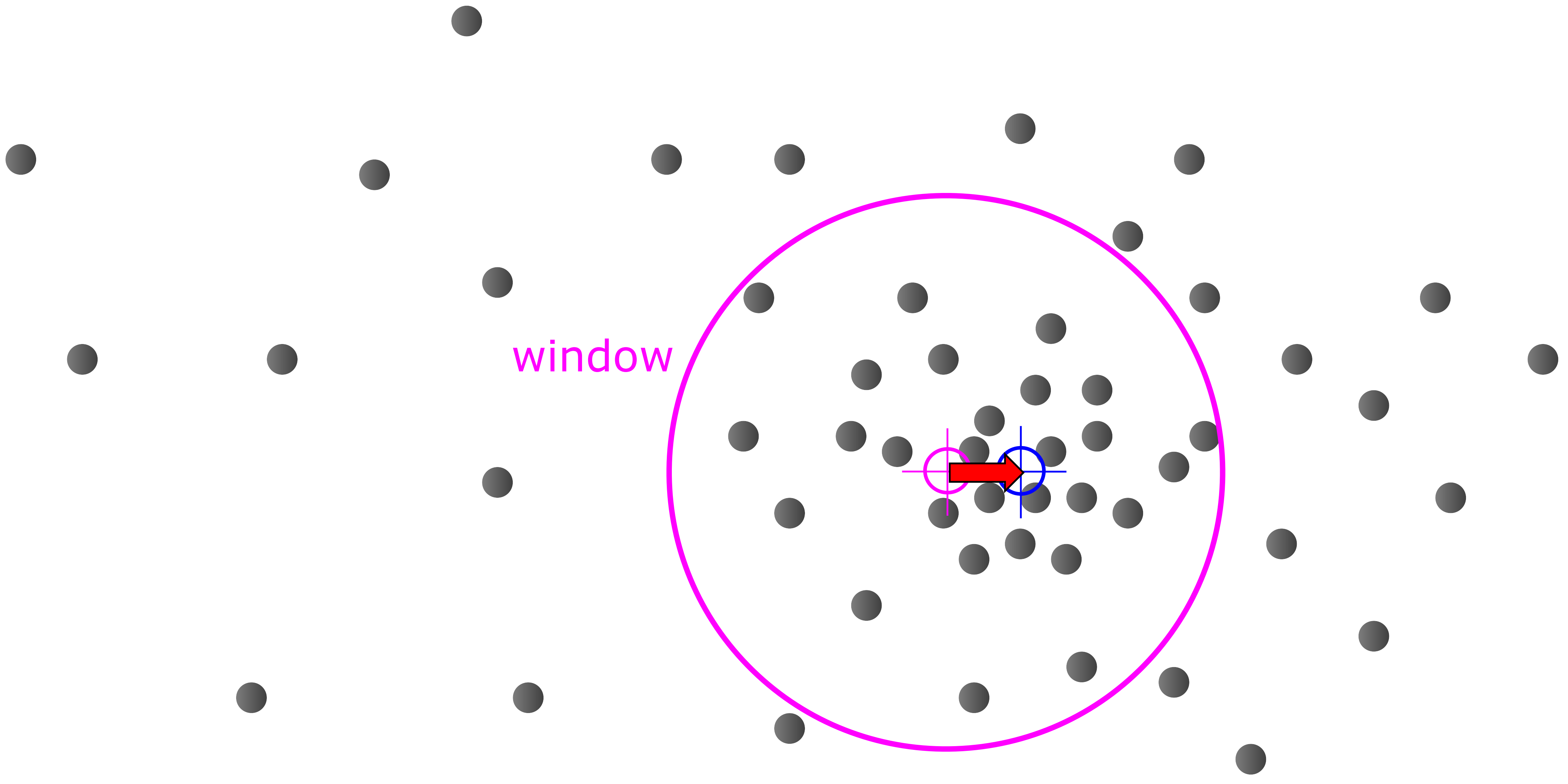
The Mean Shift Process



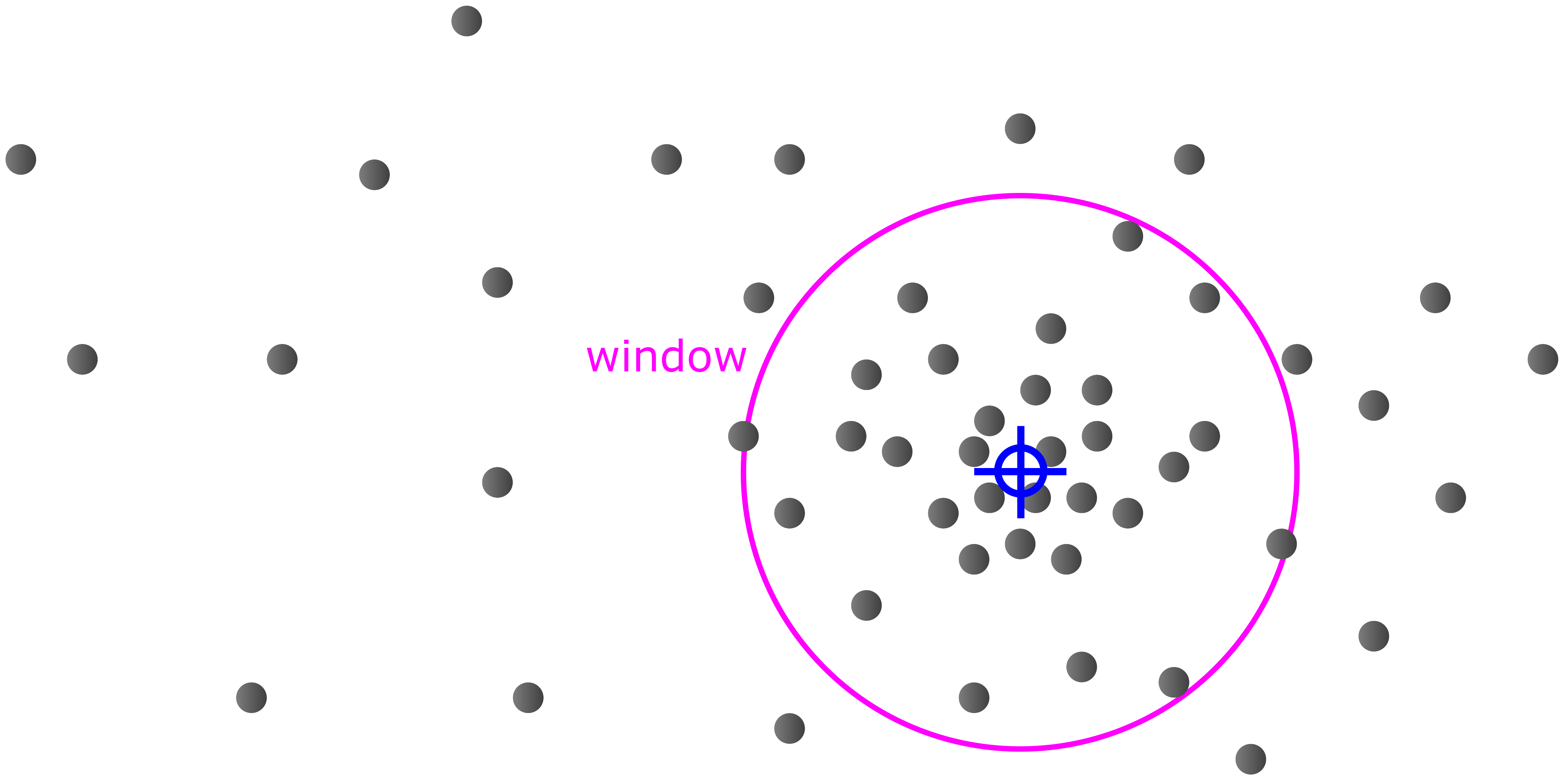
The Mean Shift Process



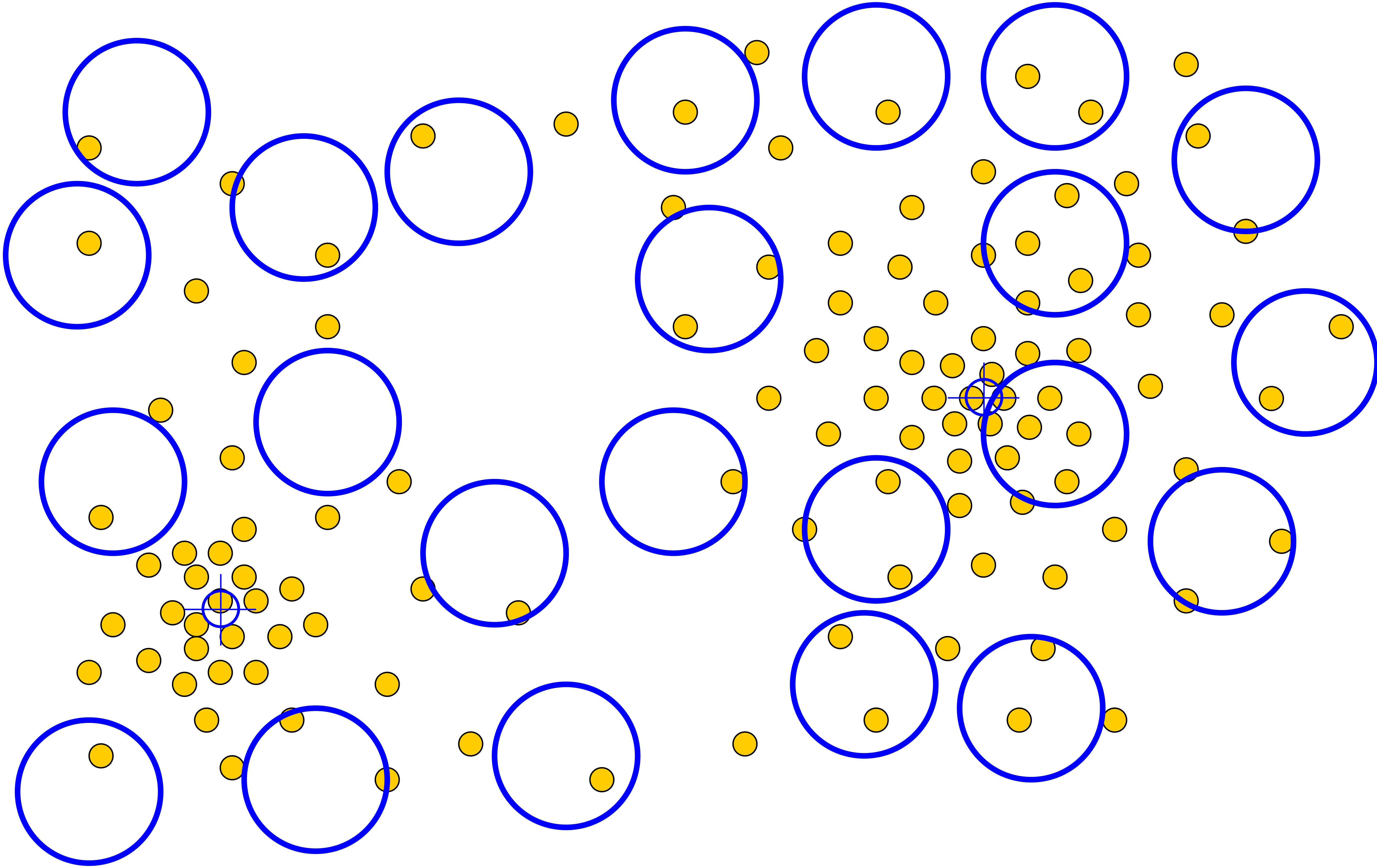
The Mean Shift Process



The Mean Shift Process

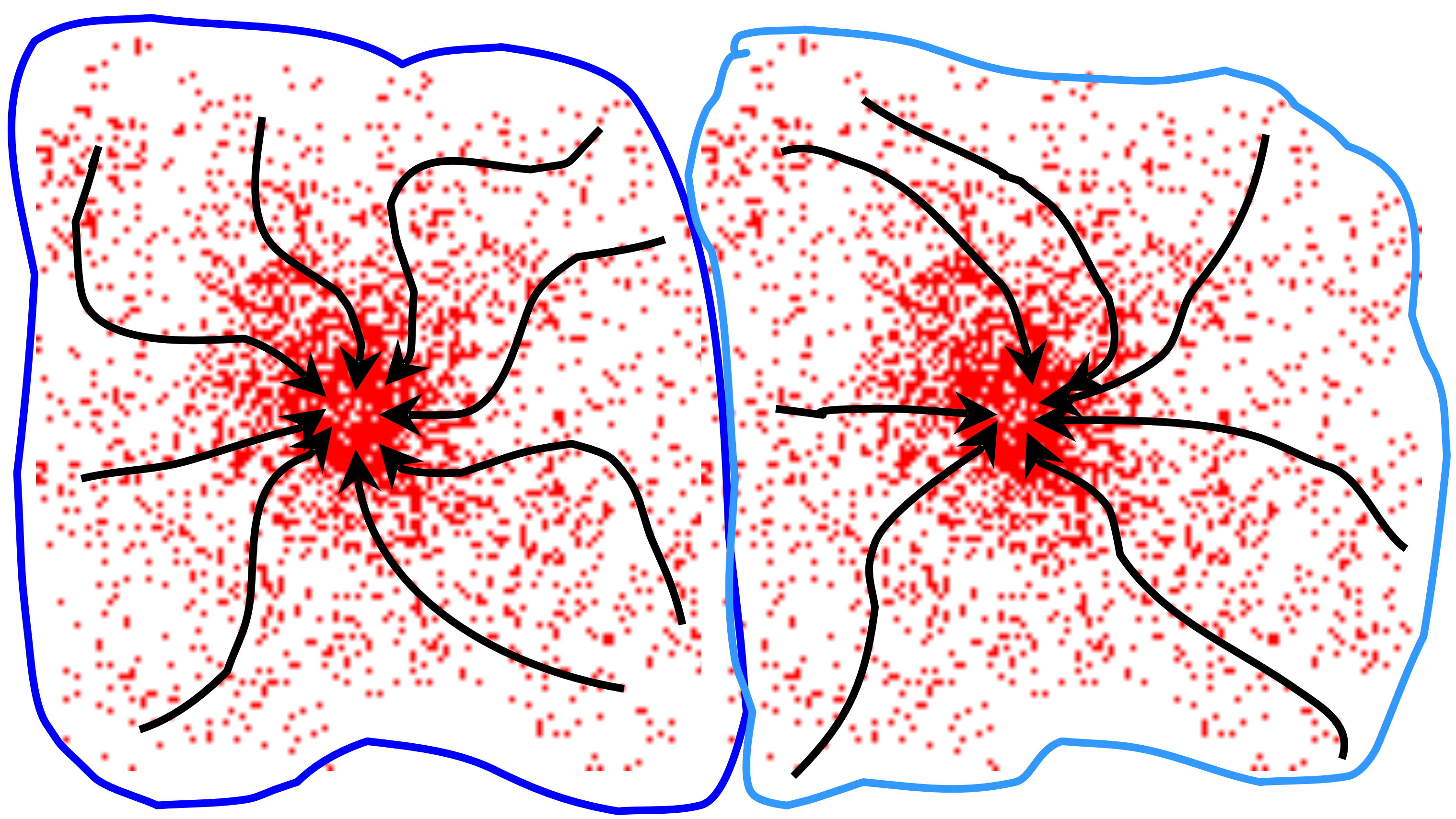


The Mean Shift Process



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

Example: Color Segmentation

Feature space: $(L,u,v,x,y) \rightarrow$ Intensity + (u,v) color channels + Position in image (x,y)

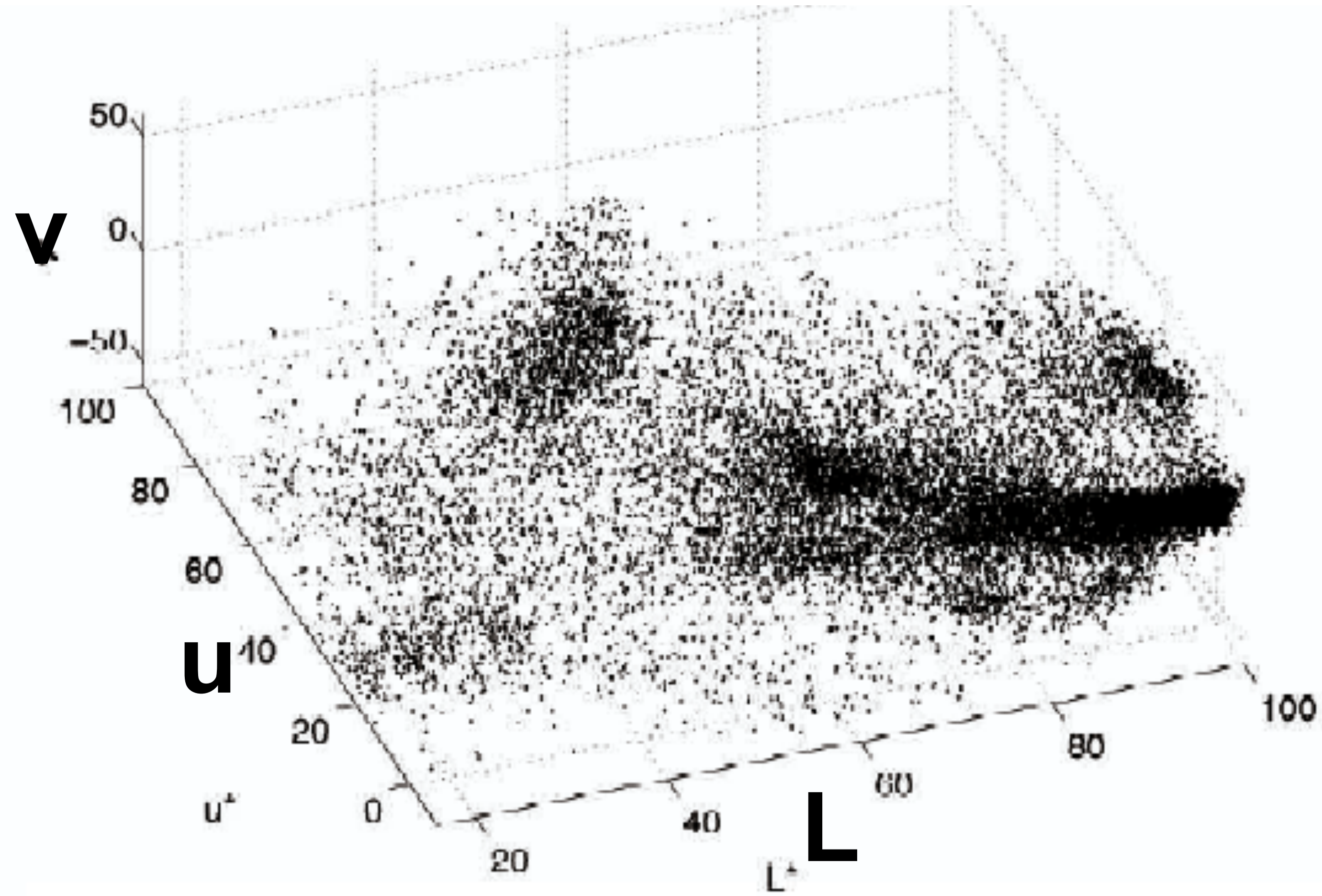
Apply meanshift in the 5-dimensional space

For each pixel (x_i,y_i) of intensity L_i and color (u_i,v_i), find the corresponding mode c_k

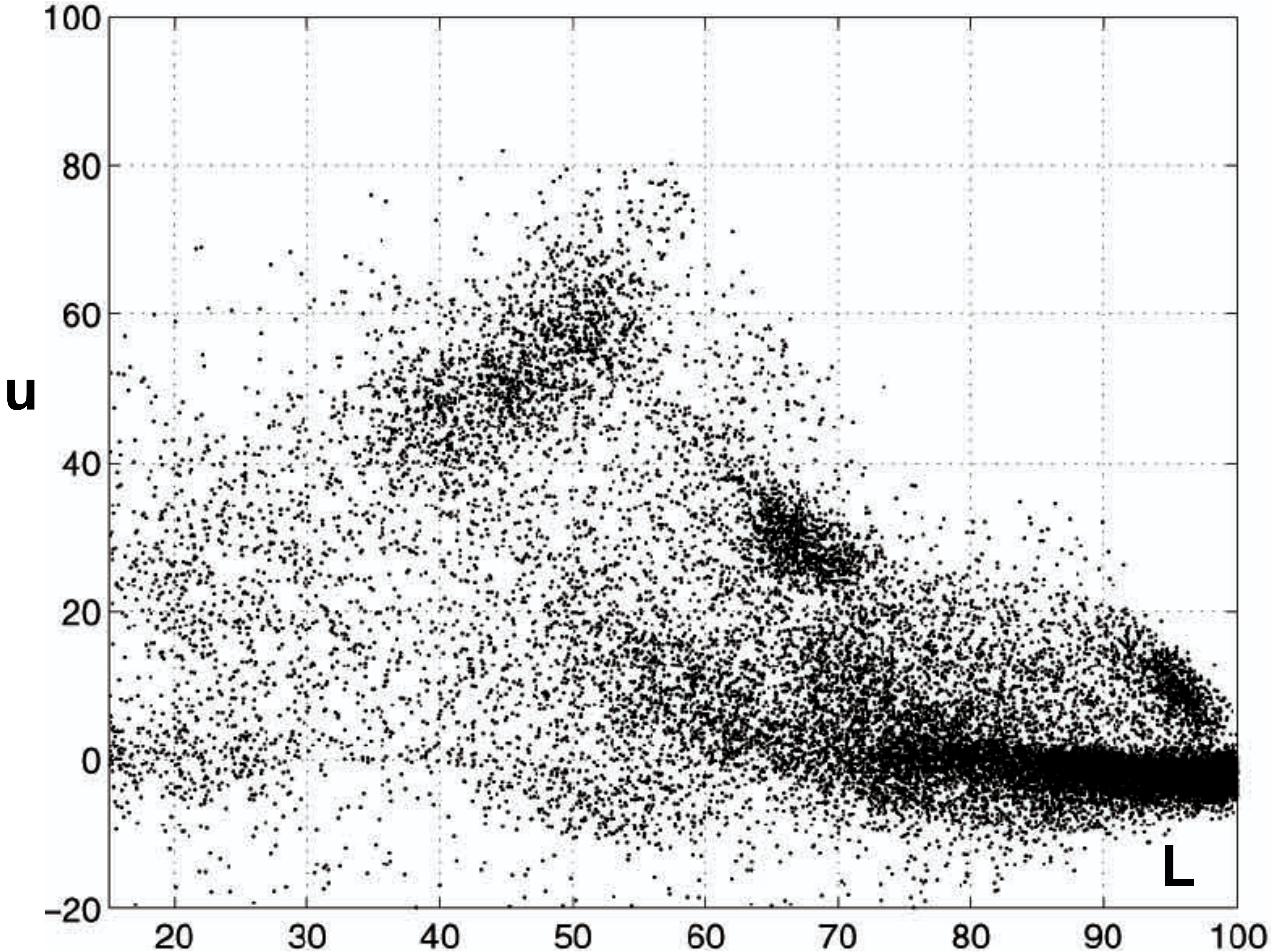
All of the pixel (x_i,y_i) corresponding to the same mode c_k are grouped into a single region

Example: Color Segmentation

Input Image



Luv Space ()



110,400 data points.

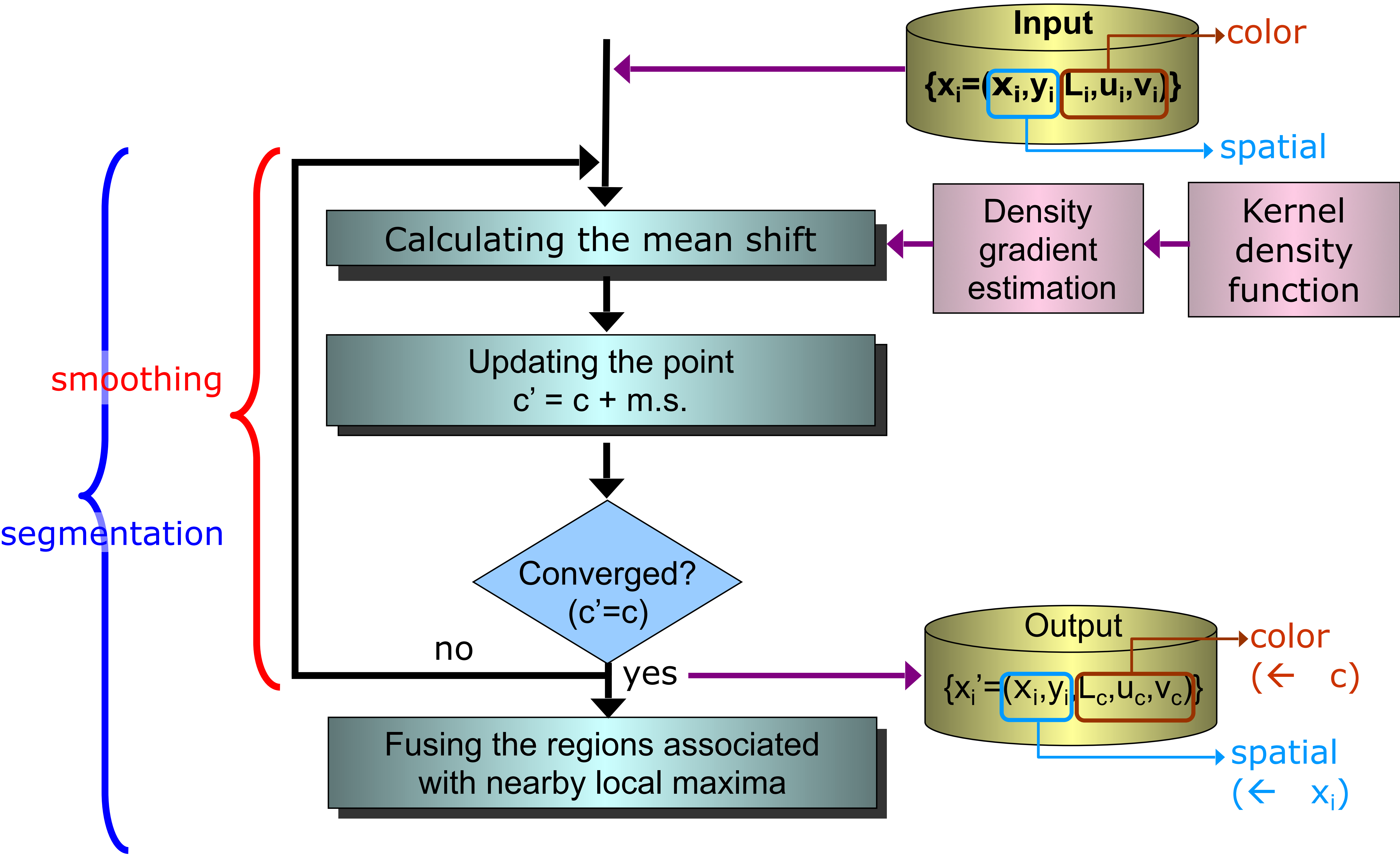
$$K_{h_{pos}h_{col}}(X) = c\,k\left(\frac{\left\|X_{pos}\right\|^2}{h_{pos}^{\ 2}}\right)k\left(\frac{\left\|X_{col}\right\|^2}{h_{col}^{\ 2}}\right)$$

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}

Mean Shift for Segmentation



The Mean Shift Process

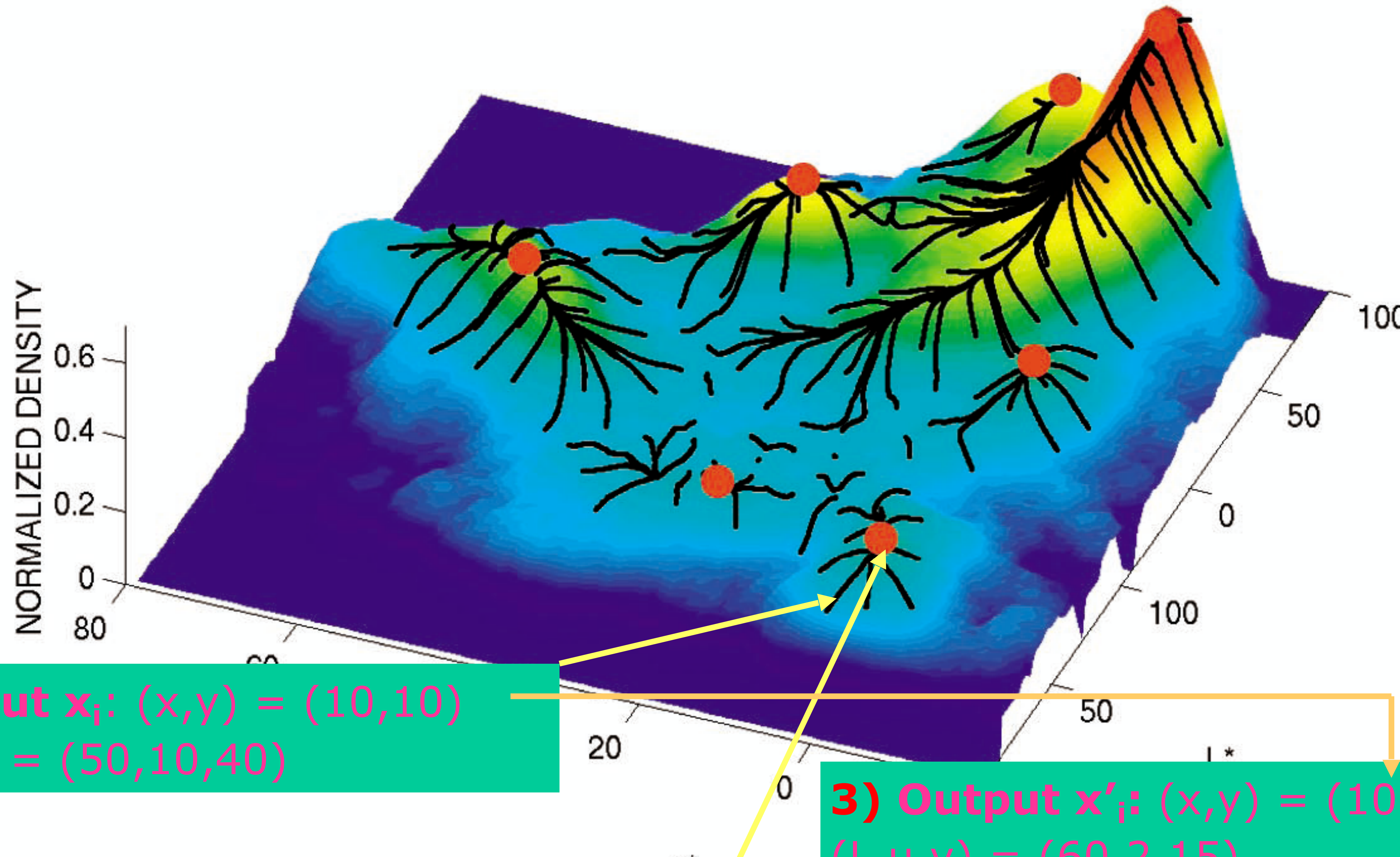
Notes:

• 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



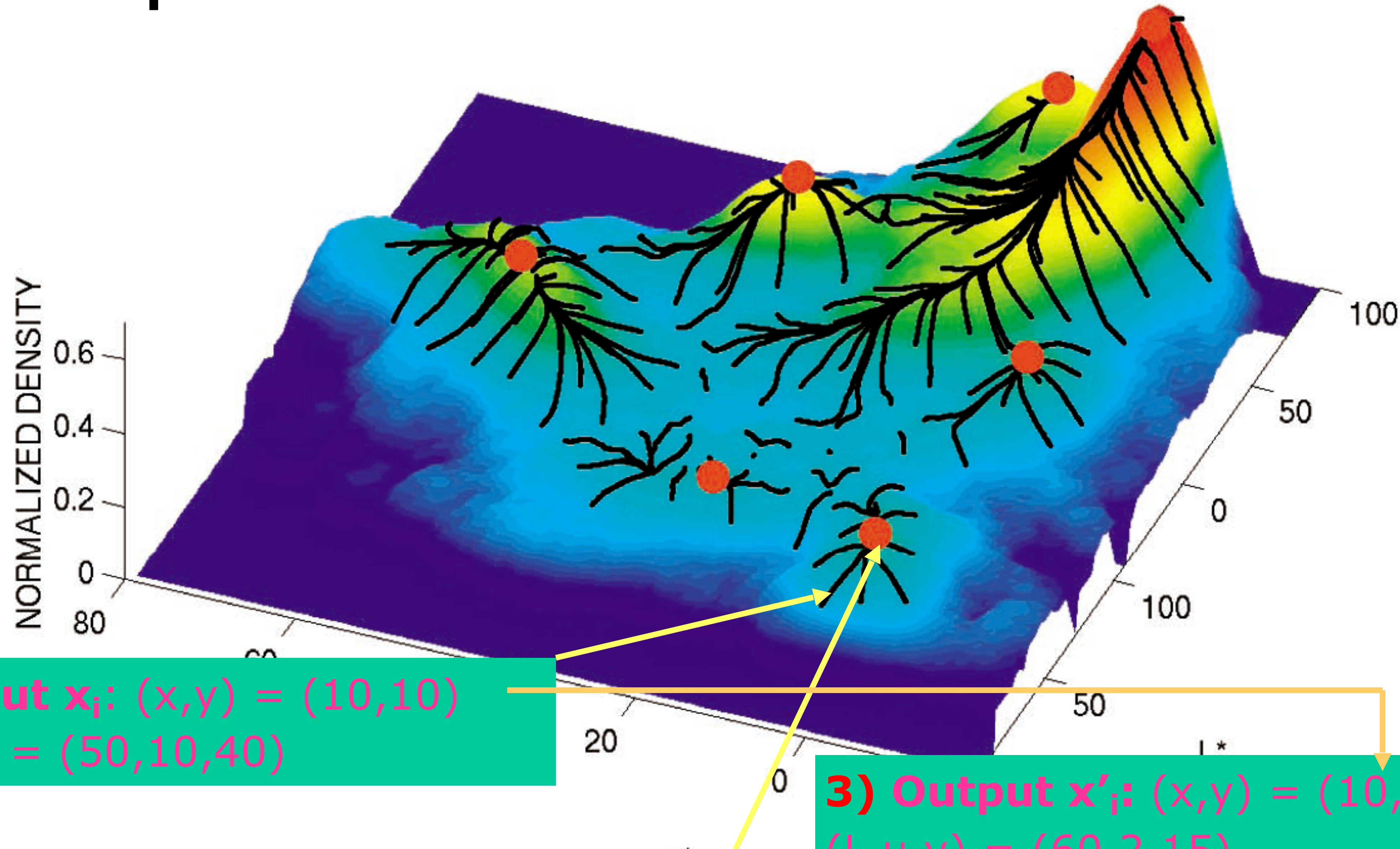
1) Input x_i: (x,y) = (10,10)
(L,u,v) = (50,10,40)

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)

Example: Color



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

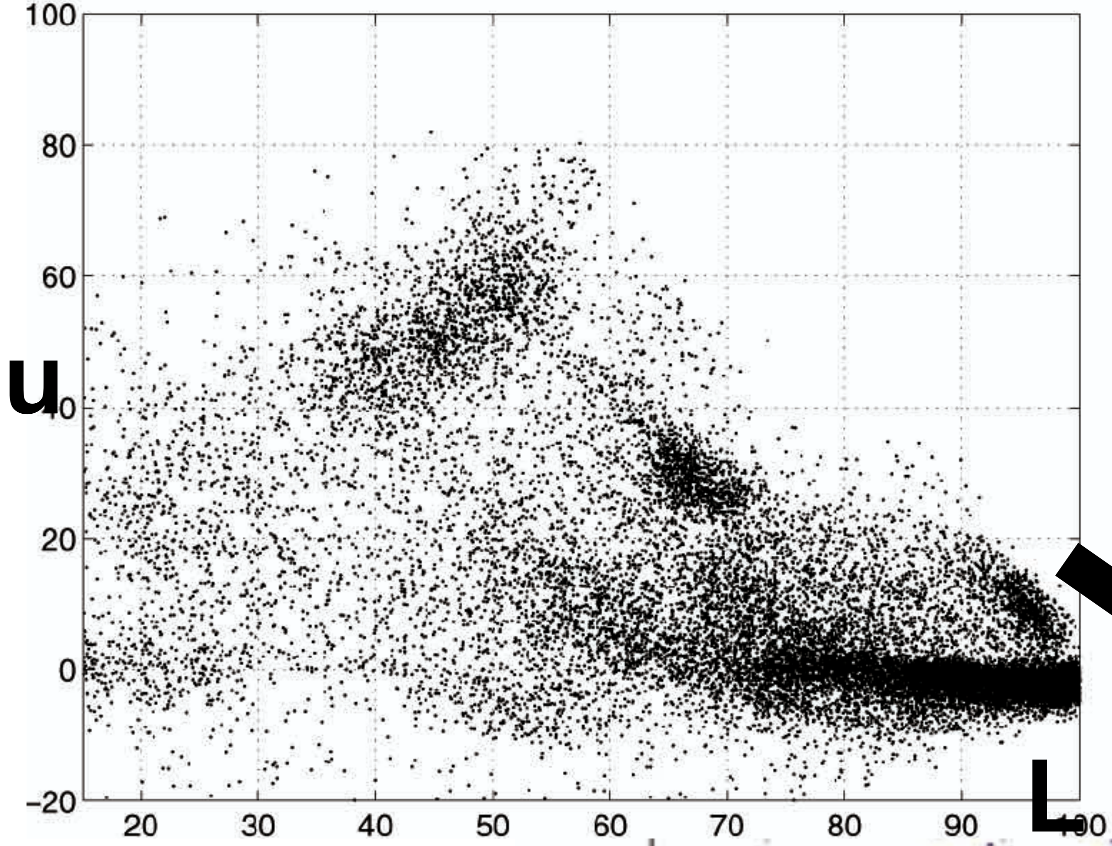
2) Apply mean shift till converged

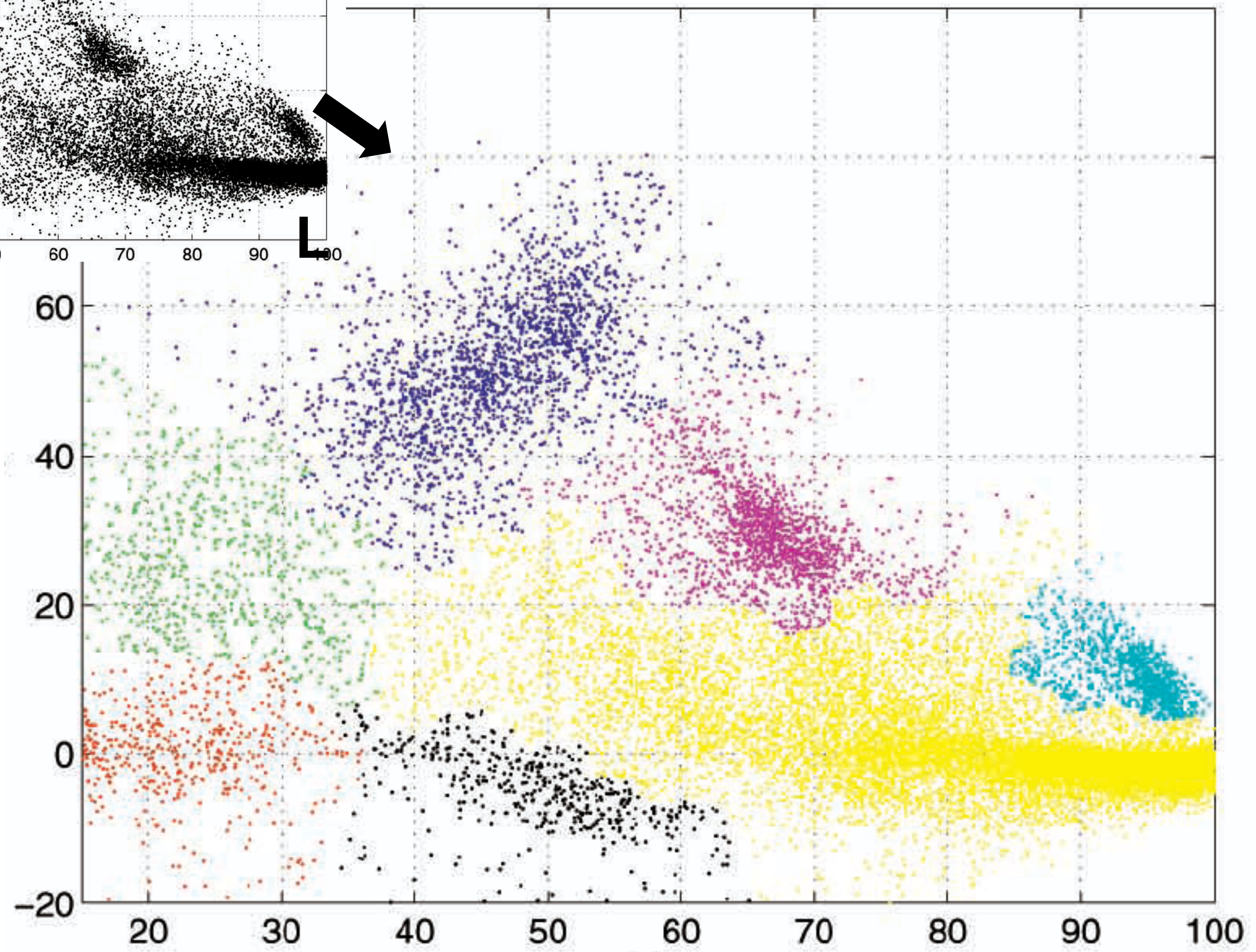
c_i: (x,y) = (15,20) (L,u,v) = (60,2,15)

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

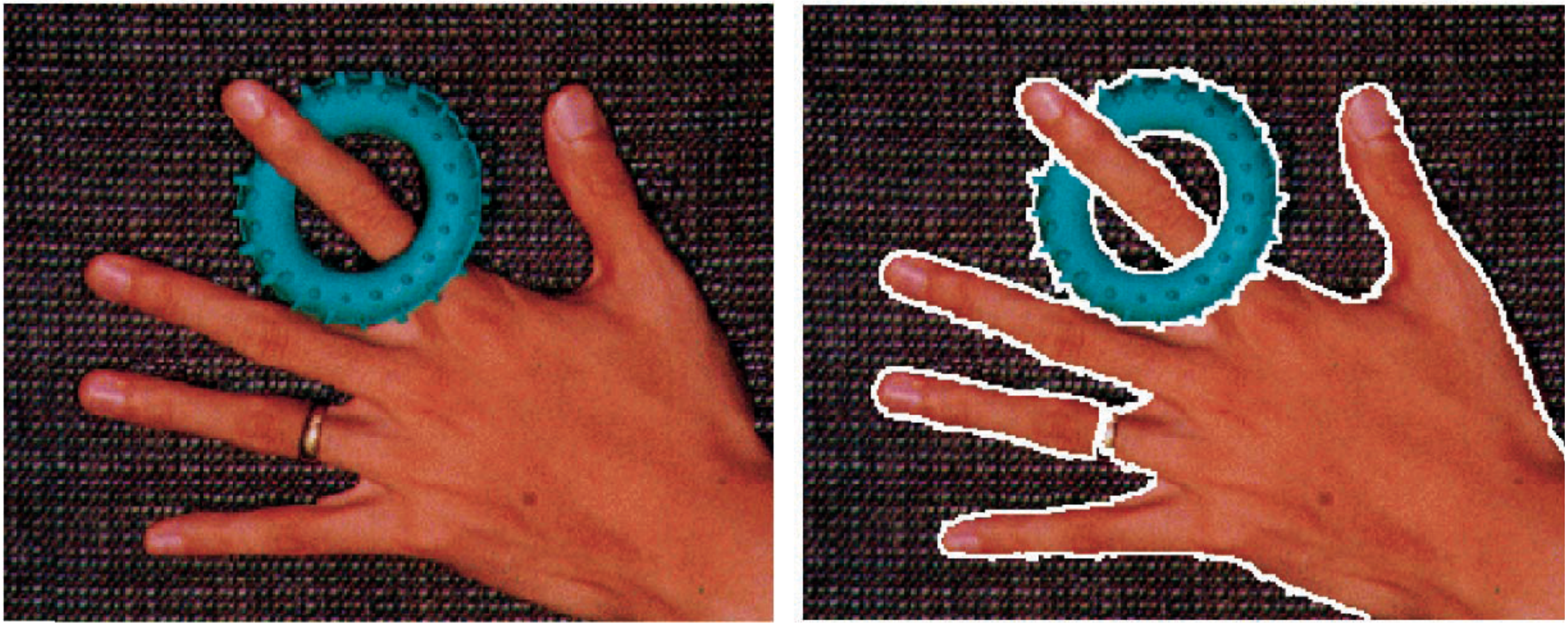
Note: In practice, all points may not converge to the same mode
→ Need an additional (easy) clustering step to group the converged locations to the location

Clustering Result

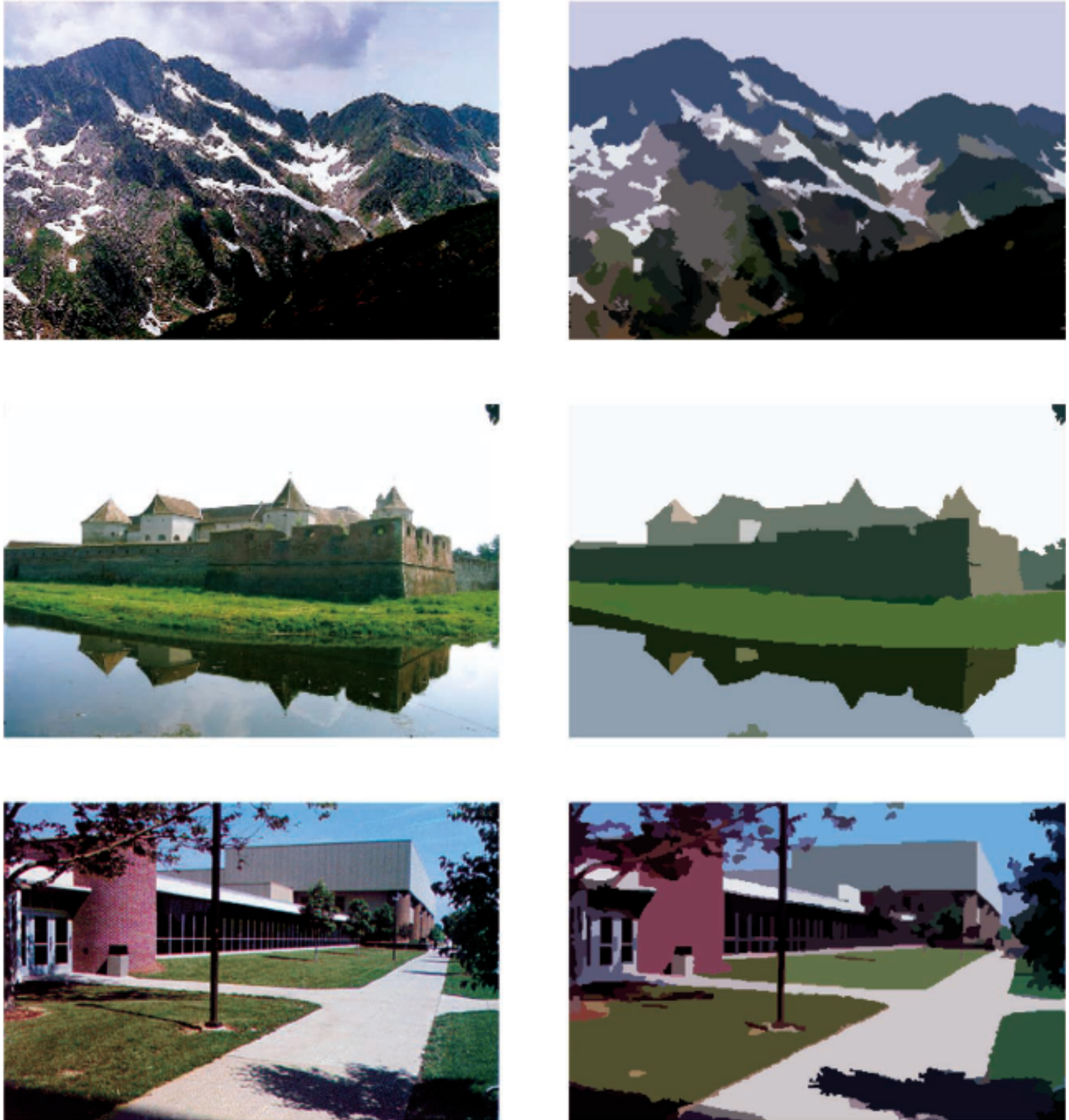




Experimental Results

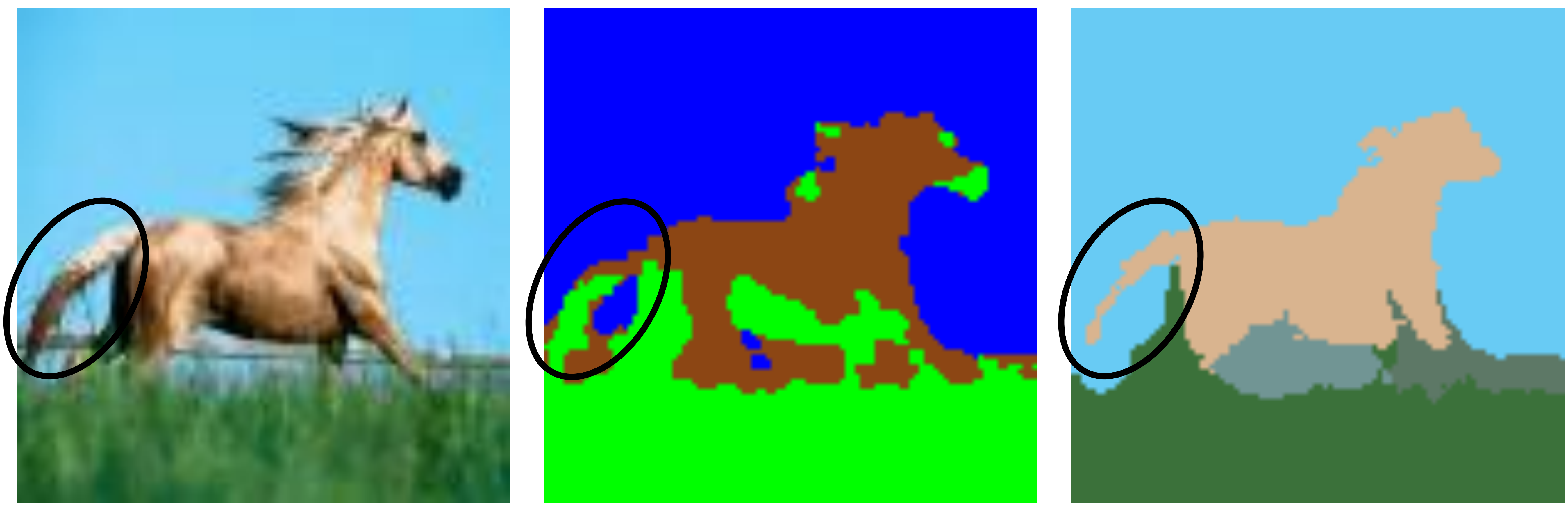


Experimental results

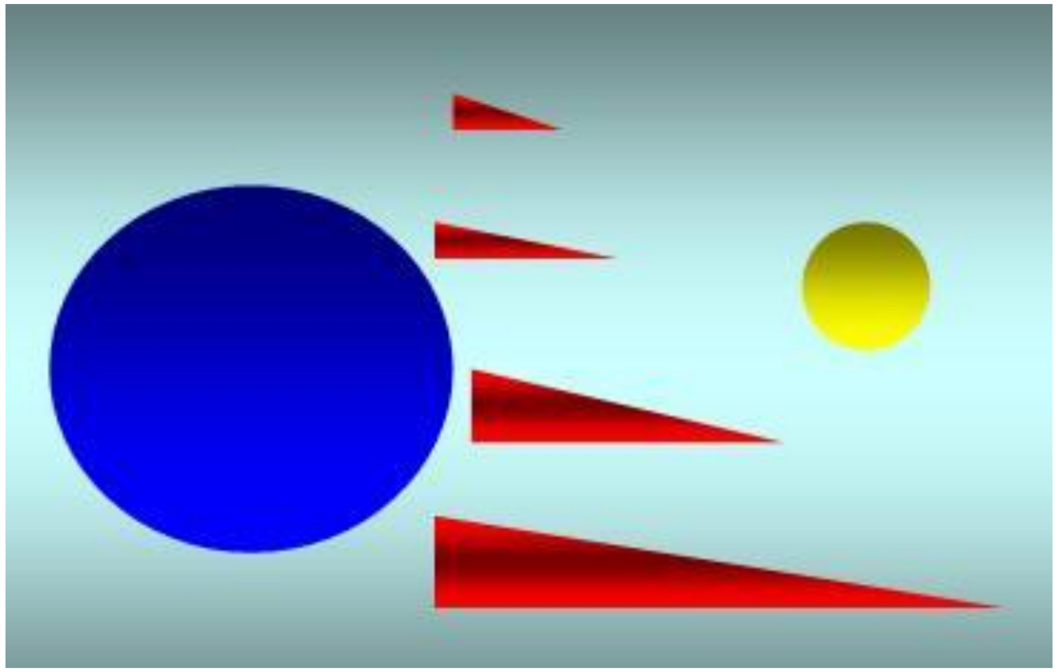


Results - Comparing to EM

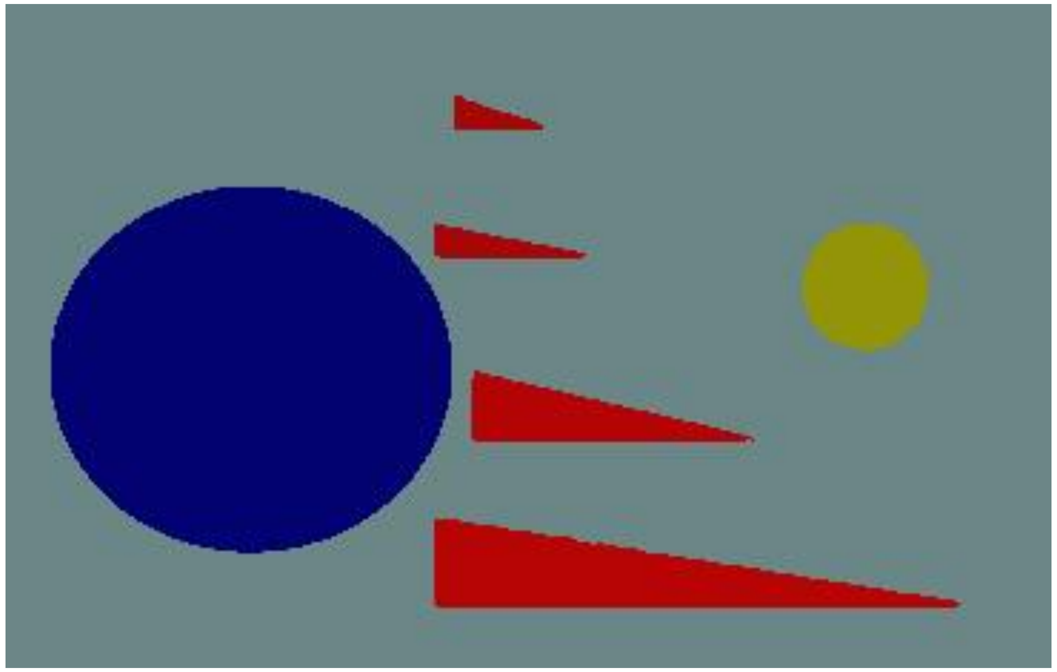
- Original
- EM with 3 clusters and 5 equally weighted features RGB and XY
- Mean shift $(h_{pos}, h_{col}) = (12, 16)$



Results - Comparing to EM



Original image



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

EM with 4 clusters

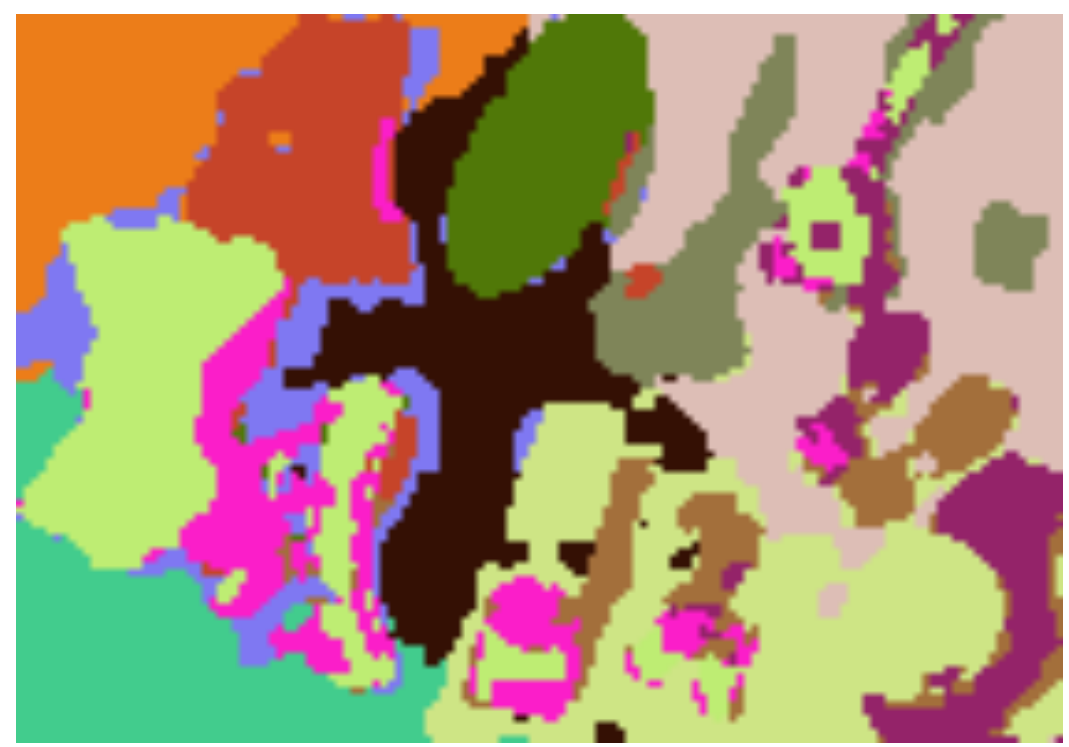
EM with 7 clusters

Results - Comparing to EM

Original image

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

EM with 5 clusters



EM with 13 clusters

Mean shift pros and cons

- Pros
  - Does not assume spherical clusters
  - Very few parameters (window size)
  - Finds variable number of modes
  - Robust to outliers
- Cons
  - Output depends on window size
  - Computationally expensive
  - Does not scale well with dimension of feature space

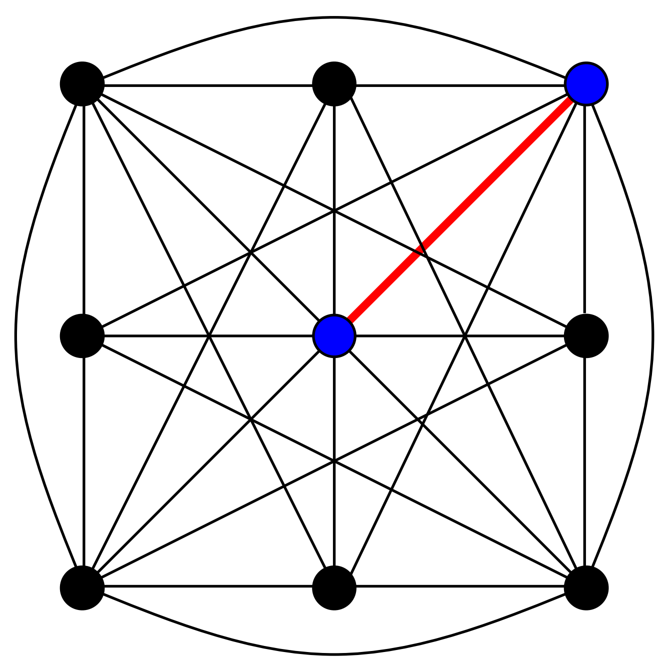
References

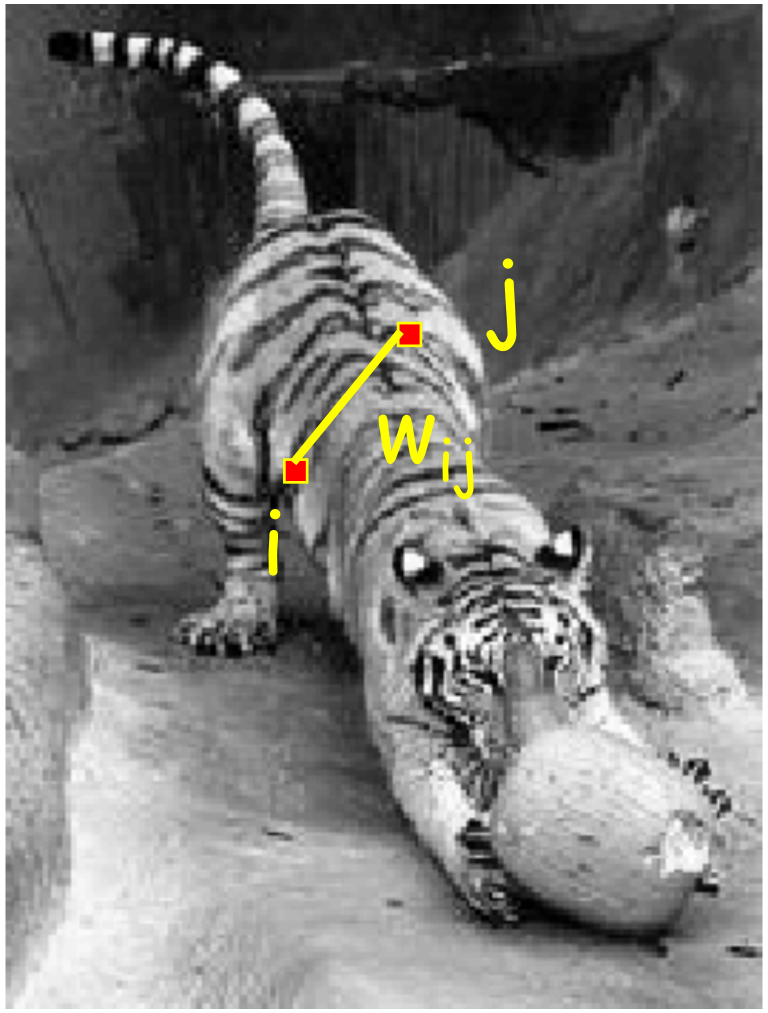
D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis". IEEE Trans. PAMI, Vol. 24, No. 5, 2002.

R. Szeliski Computer Vision: Algorithms and Applications, Chapter 5

http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

Graph-based segmentation





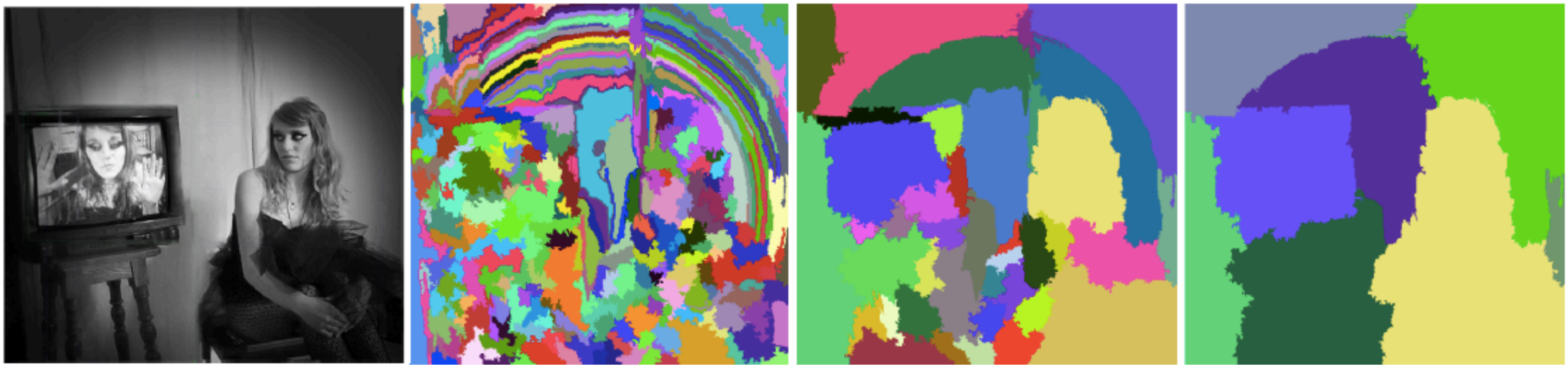
- Node = pixel
- Edge = pair of neighboring pixels
- Edge weight = similarity or dissimilarity of the respective nodes

Source: S. Seitz

Felzenszwalb & Huttenlocher algorithm

- Graph definition:
  - Vertices are pixels, edges connect neighboring pixels, weights correspond to ***dissimilarity*** in (x,y,r,g,b) space
- The algorithm:
  - Start with each vertex in its own component
  - For each edge in increasing order of weight:
    - If the edge is between vertices in two different components A and B, merge if the edge weight is lower than the internal dissimilarity within either of the components
    - Threshold is the minimum of the following values, computed on A and B:
      - (Highest-weight edge in minimum spanning tree of the component) + (k / size of component)

Efficient graph-based segmentation



- Runs in time nearly linear in the number of edges
- Easy to control coarseness of segmentations
- Results can be unstable

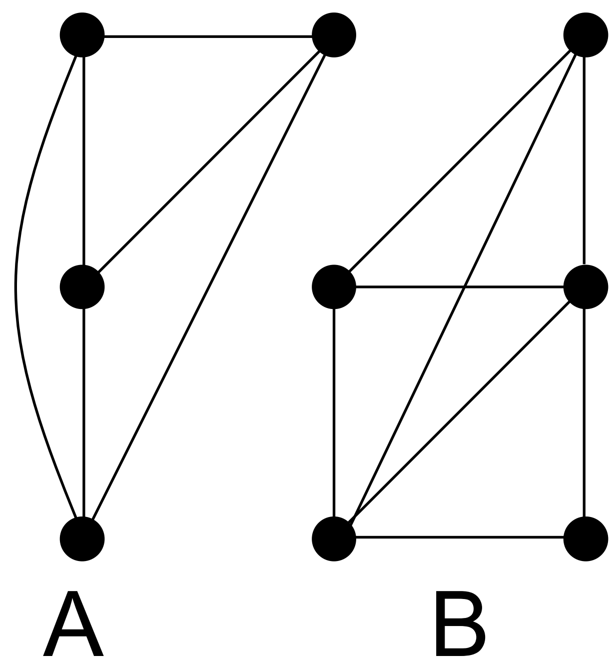
P. Felzenszwalb and D. Huttenlocher, [Efficient Graph-Based Image Segmentation](), IJCV 2004

Example results




http://www.cs.brown.edu/~pff/segment/

Segmentation by graph cuts

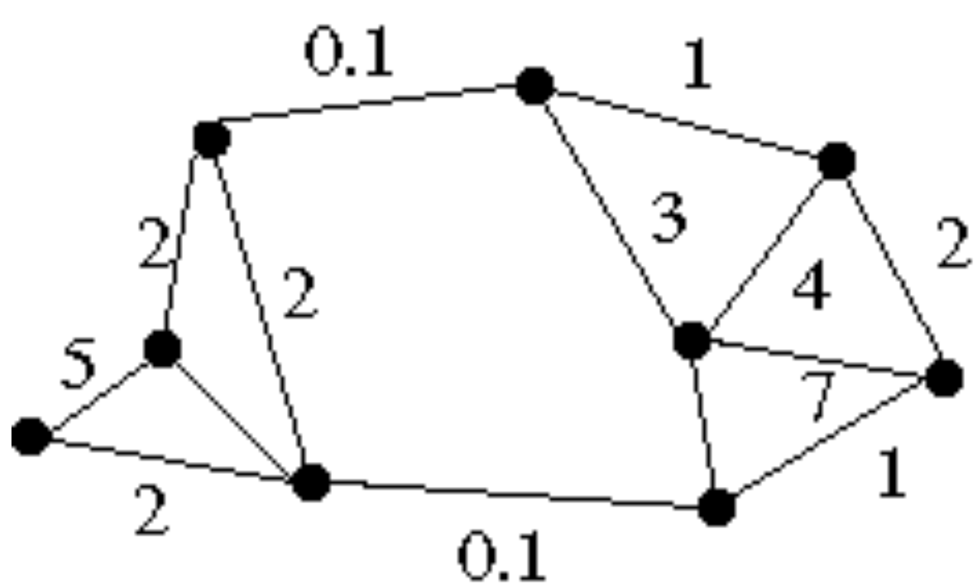


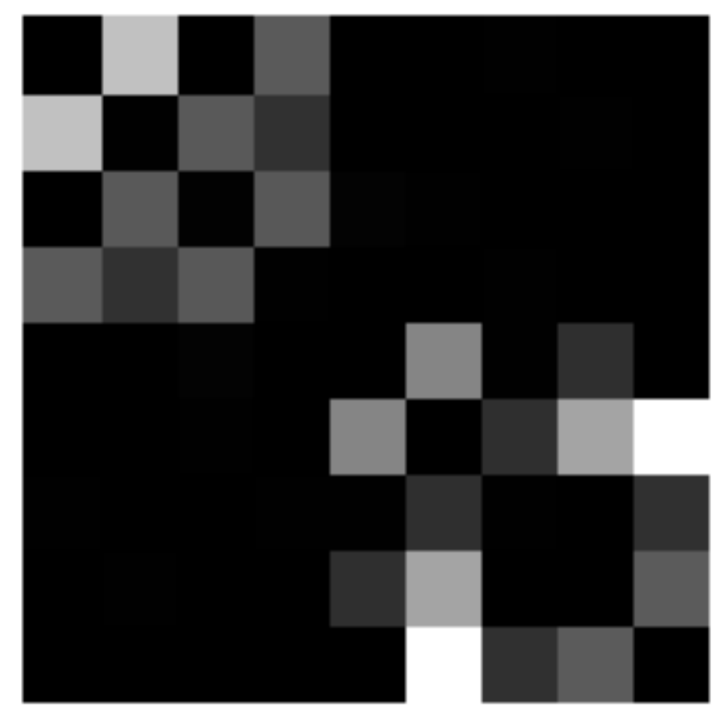
- Break graph into segments
  - Delete links that cross between segments
  - Easiest to break links that have low *affinity*
    - similar pixels should be in the same segments
    - dissimilar pixels should be in different segments

Source: S. Seitz

Segmentation by graph cuts

- A graph cut is a set of edges whose removal disconnects the graph
- Cost of a cut: sum of weights of cut edges
- Two-way minimum cuts can be found efficiently

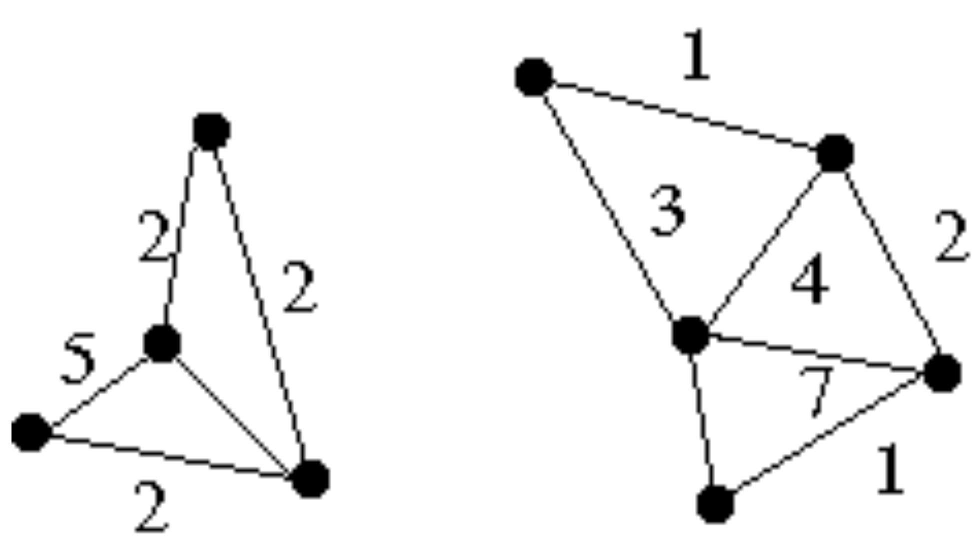




Affinity matrix

Segmentation by graph cuts

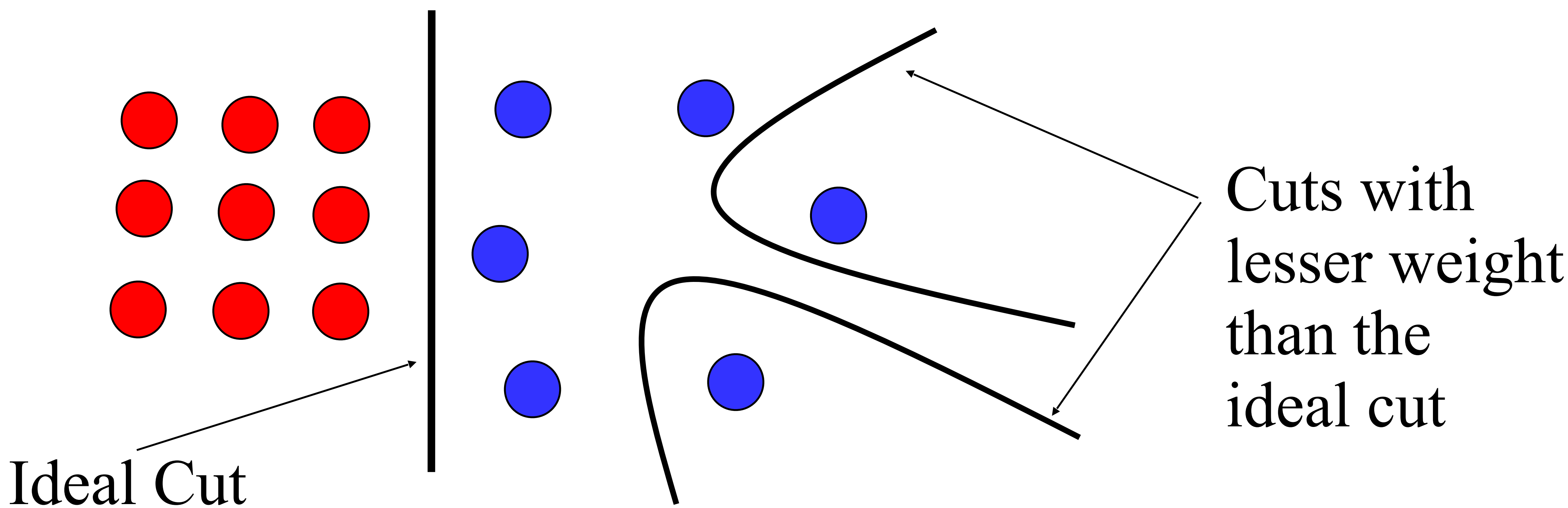
- A graph cut is a set of edges whose removal disconnects the graph
- Cost of a cut: sum of weights of cut edges
- Two-way minimum cuts can be found efficiently



Affinity matrix

Normalized cut

- Minimum cut tends to cut off very small, isolated components



Source: S. Lazebnik

Normalized cut

- To encourage larger segments, normalize the cut by the total weight of edges incident to the segment
- The *normalized cut* cost is:

$$ncut(A,B) = \frac{w(A,B)}{w(A,V)} + \frac{w(A,B)}{w(B,V)}$$

  - Intuition: big segments will have a large $w(A,V)$, thus decreasing $ncut(A, B)$

$w(A, B)$ = sum of weights of all edges between A and B

- Finding the globally optimal cut is NP-complete,
  but a relaxed version can be solved using a generalized eigenvalue problem

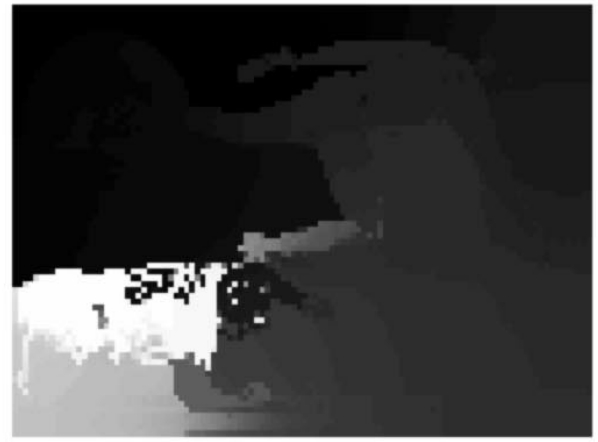
J. Shi and J. Malik. [Normalized cuts and image segmentation.] PAMI 2000

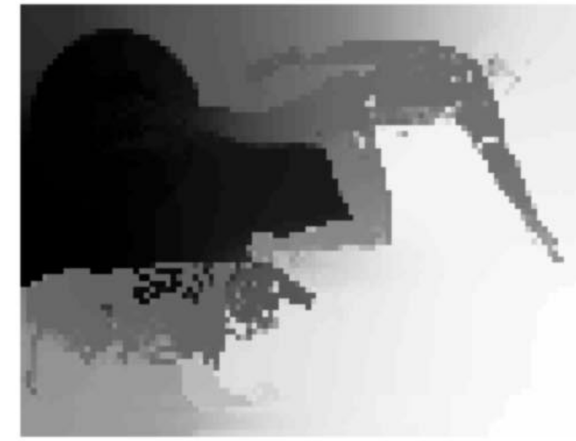
Normalized cut: Algorithm

- Let $\mathbf{W}$ be the affinity matrix of the graph (n x n for n pixels)
- Let $\mathbf{D}$ be the diagonal matrix with entries $\mathbf{D}(i, i) = \Sigma_j \mathbf{W}(i, j)$
- Solve *generalized eigenvalue problem* $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$ for the eigenvector with the second smallest eigenvalue
  - The ith entry of $\mathbf{y}$ can be viewed as a "soft" indicator of the component membership of the ith pixel
    - Use 0 or median value of the entries of $\mathbf{y}$ to split the graph into two components
  - To find more than two components:
    - Recursively bipartition the graph
    - Run k-means clustering on values of several eigenvectors

Example result

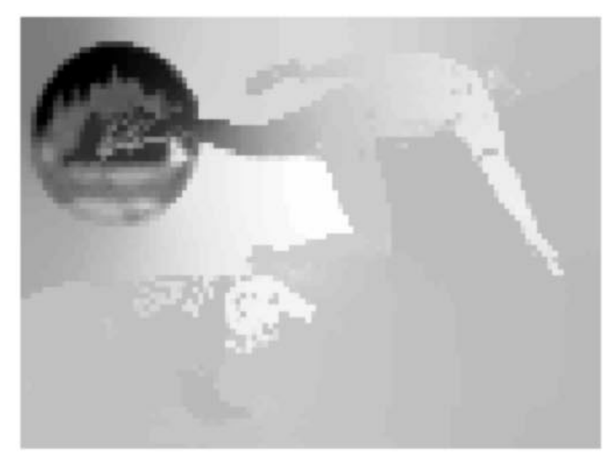
Original image





Eigenvectors for 2nd and 3rd smallest eigenvalues

More eigenvectors



Normalized cuts: Pro and con

- Pro
  - Generic framework, can be used with many different features and affinity formulations
- Con
  - High storage requirement and time complexity: involves solving a generalized eigenvalue problem of size n x n, where n is the number of pixels

Segmentation as labeling

- Suppose we want to segment an image into foreground and background
  - Binary pixel labeling problem

Segmentation as labeling

- Suppose we want to segment an image into foreground and background
    - Binary pixel labeling problem
    - Naturally arises in interactive settings

User scribbles

Labeling by energy minimization

- Define a labeling **c** as an assignment of each pixel to a class (foreground or background)


- Find the labeling that minimizes a global energy function:

$$E(\mathbf{c}|\mathbf{x}) = \sum_{i} \underbrace{f_i(c_i,\mathbf{x})} + \sum_{i,j\in\varepsilon} \underbrace{g_{ij}(c_i, c_j, \mathbf{x})}$$

Pixels

Unary potential *(local data term): score for pixel i and label* c_i

Neighboring pixels

Pairwise potential *(context or smoothing term)*

- These are known as Markov Random Field (MRF) or Conditional Random Field (CRF) functions

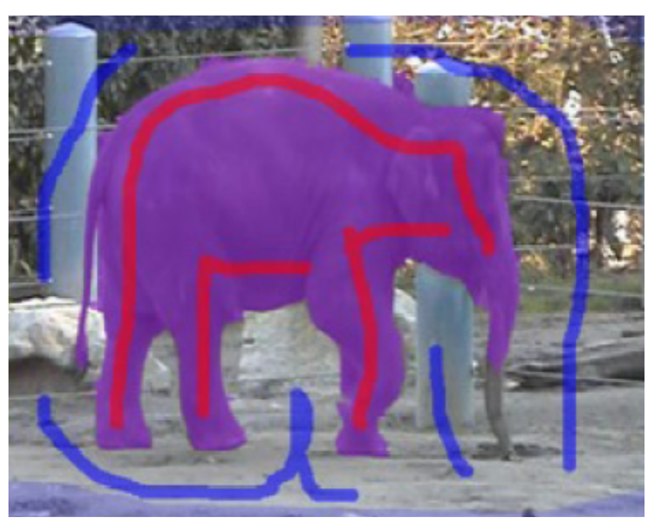
Source: S. Lazebnik

Segmentation by energy minimization

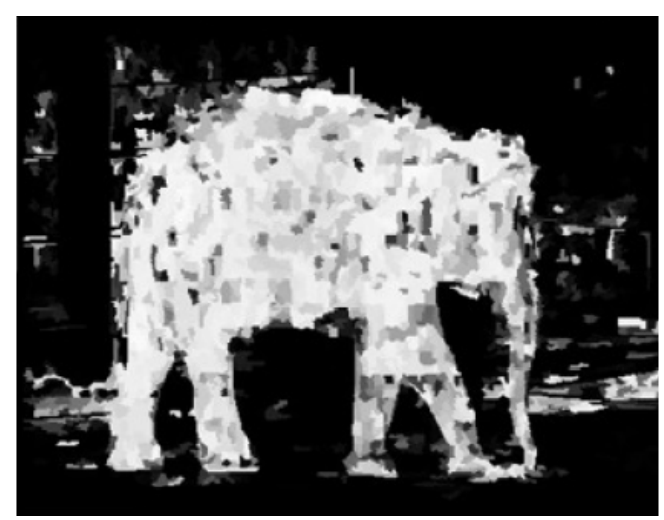
$$E(\mathbf{c} \mid \mathbf{x}) = \sum_{i} f_i(c_i, \mathbf{x}) + \sum_{i,j \in \varepsilon} g_{ij}(c_i, c_j, \mathbf{x})$$

- Unary potentials: $f_i(c, \mathbf{x}) = -\log P(c \mid \mathbf{x}_i)$
  - Cost is infinity if label does not match the user scribble
  - Otherwise, it is computed based on a color model of user-labeled pixels

User scribbles



P(foreground | $\mathbf{x}_i$)



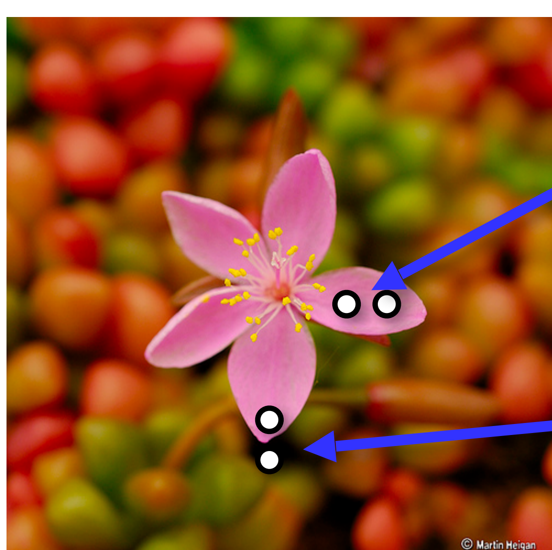
Source: S. Lazebnik

Segmentation by energy minimization

$$E(\mathbf{c} \mid \mathbf{x}) = \sum_{i} f_i(c_i, \mathbf{x}) + \sum_{i,j \in \varepsilon} g_{ij}(c_i, c_j, \mathbf{x})$$

- Unary potentials: $f_i(c, \mathbf{x}) = -\log P(c \mid \mathbf{x}_i)$
- Pairwise potentials: $g_{ij}(c, c', \mathbf{x}) = w_{ij} \left| c - c' \right|$
  - Neighboring pixels should have the same label unless they look very different

Affinity between pixels i and j



high affinity

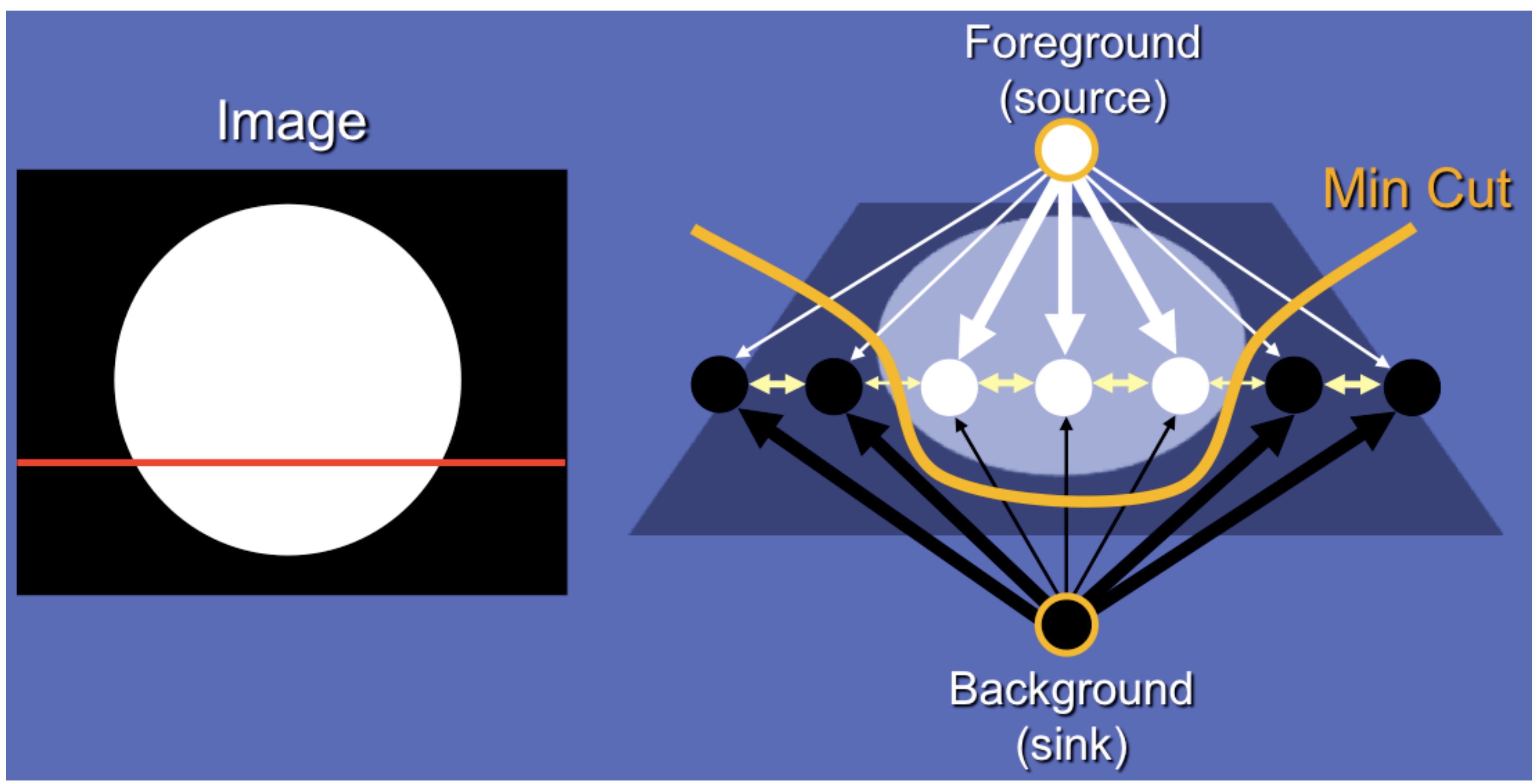
low affinity

Source: S. Lazebnik

Segmentation by energy minimization

$$E(\mathbf{c} \mid \mathbf{x}) = \sum_i f_i(c_i, \mathbf{x}) + \sum_{i,j \in \varepsilon} g_{ij}(c_i, c_j, \mathbf{x})$$

- Can be optimized by finding the minimum st-cut in the following graph:



Y. Boykov and M. Jolly, [Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images](), ICCV 2001

Source: S. Lazebnik

Summary: Segmentation

- Segmentation as Clustering:
  – K-means, EM algorithm, mean-shift
- Segmentation as graph cuts

- What about learning-based approaches (neural nets)?