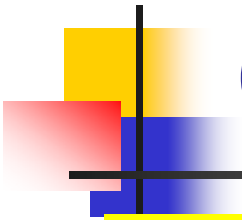
Image and Vision Computing
Segmentation

Instructor: Chen Yisong

HCI & Multimedia Lab, Peking University

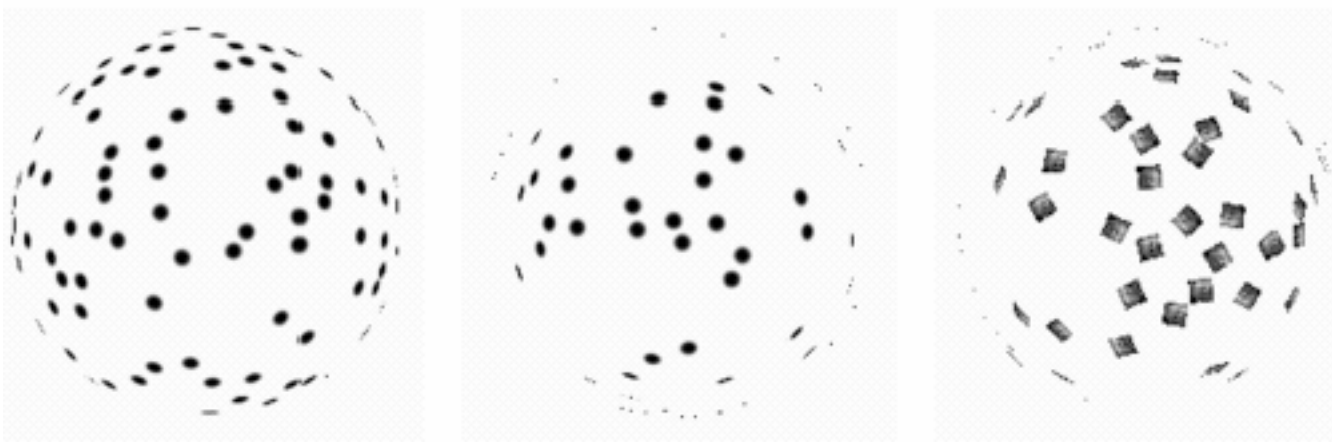
* Pictures from Mean Shift: A Robust Approach toward Feature Space Analysis, by D. Comaniciu and P. Meer http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.htm



Outline

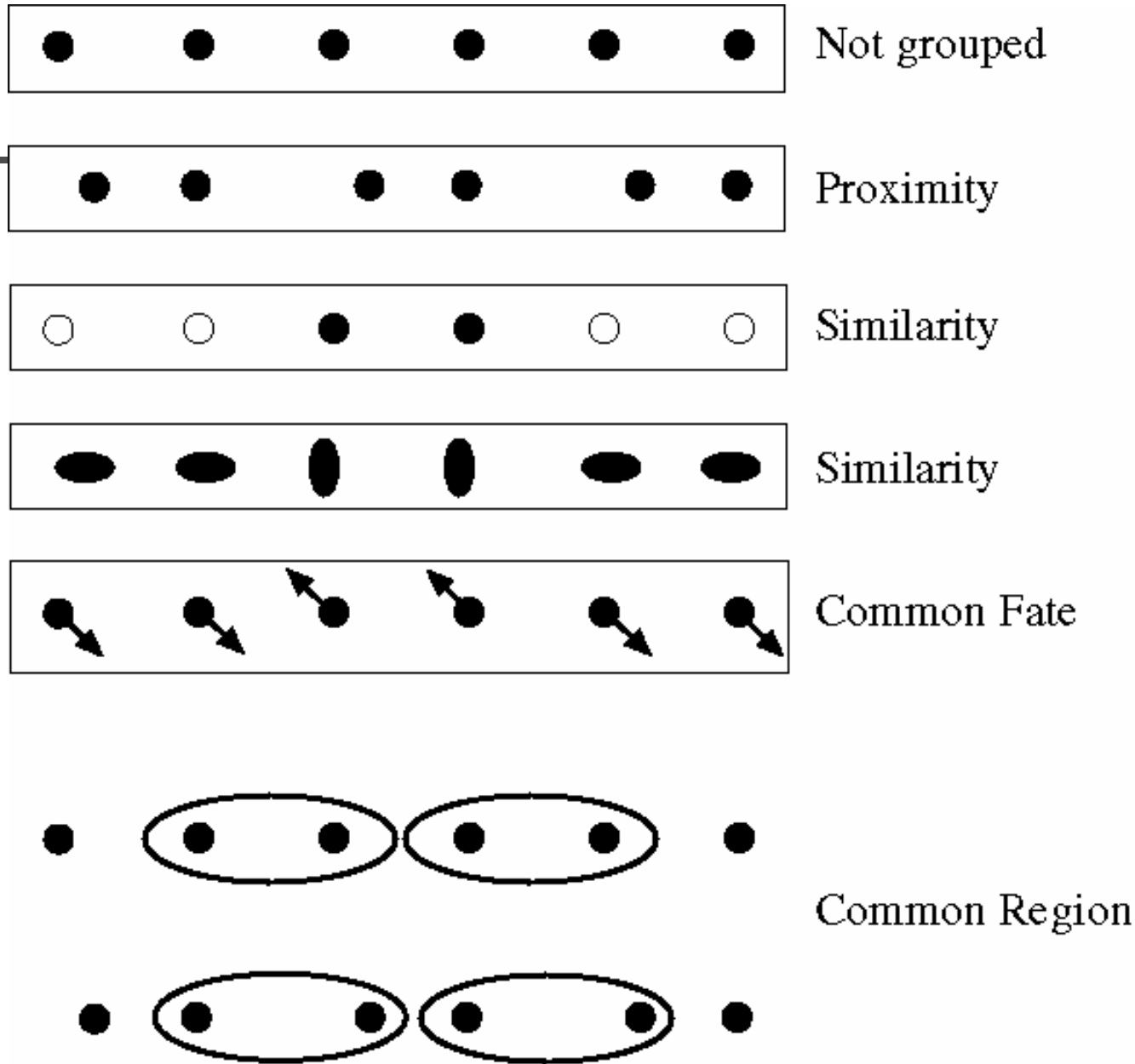
- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph Cuts

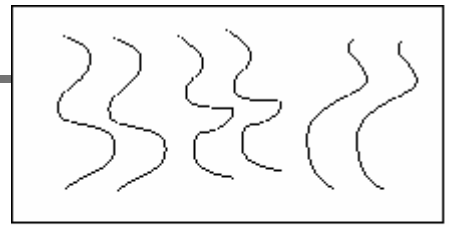
When the 3D nature of grouping is apparent:



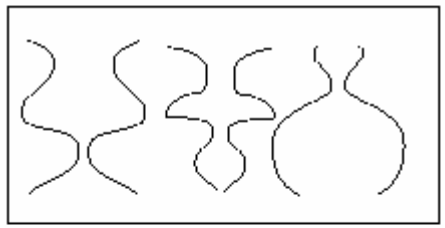
Why do these tokens belong together?

For humans at least, Gestalt psychology identifies several properties that result In grouping/segmentation:

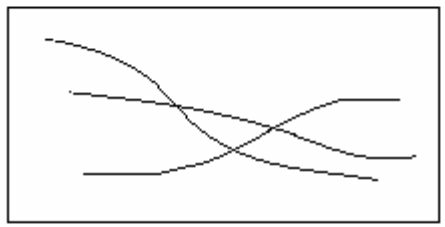




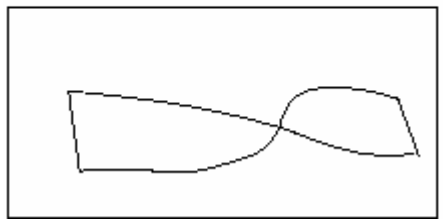
Parallelism



Symmetry

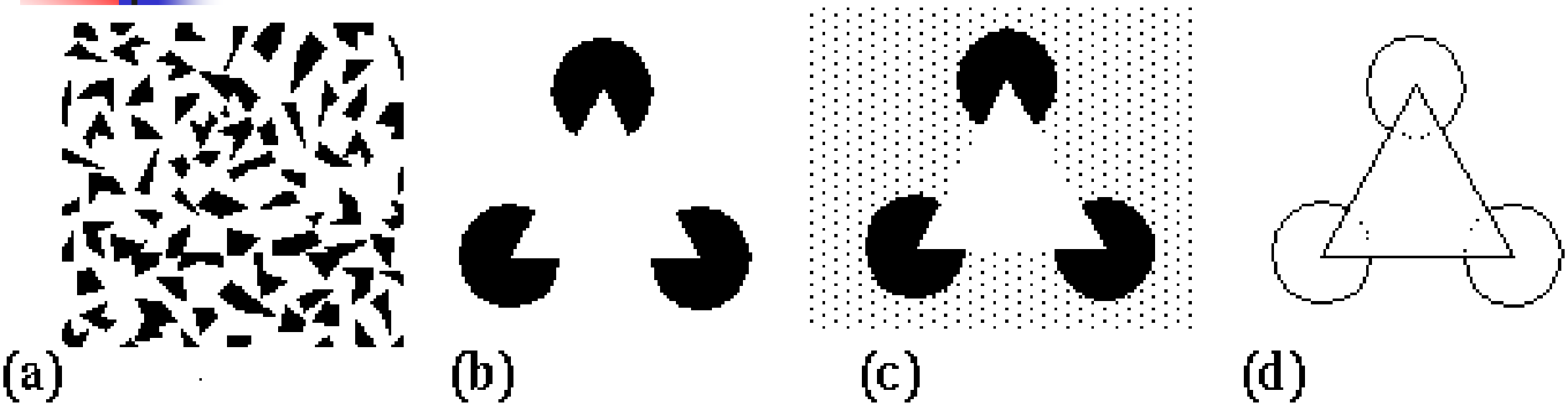


Continuity

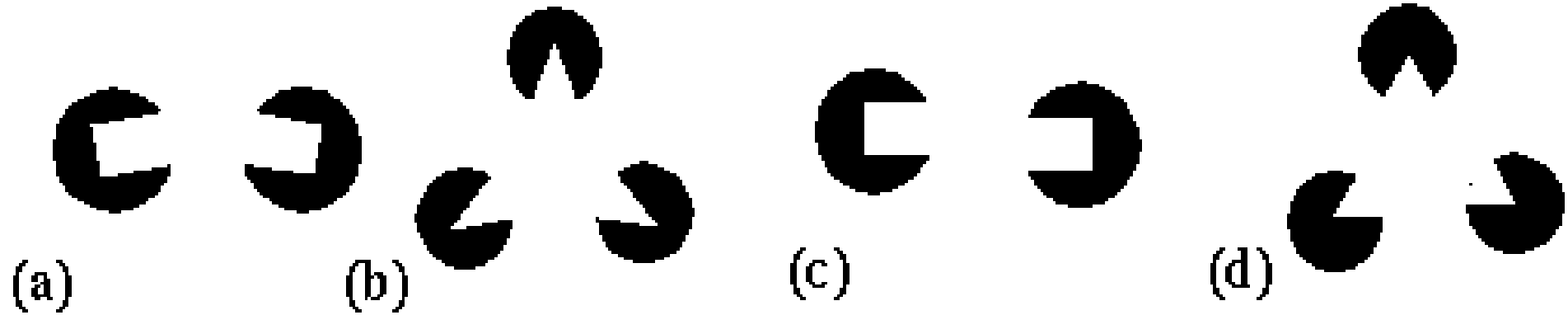


Closure

Groupings by Invisible Completions

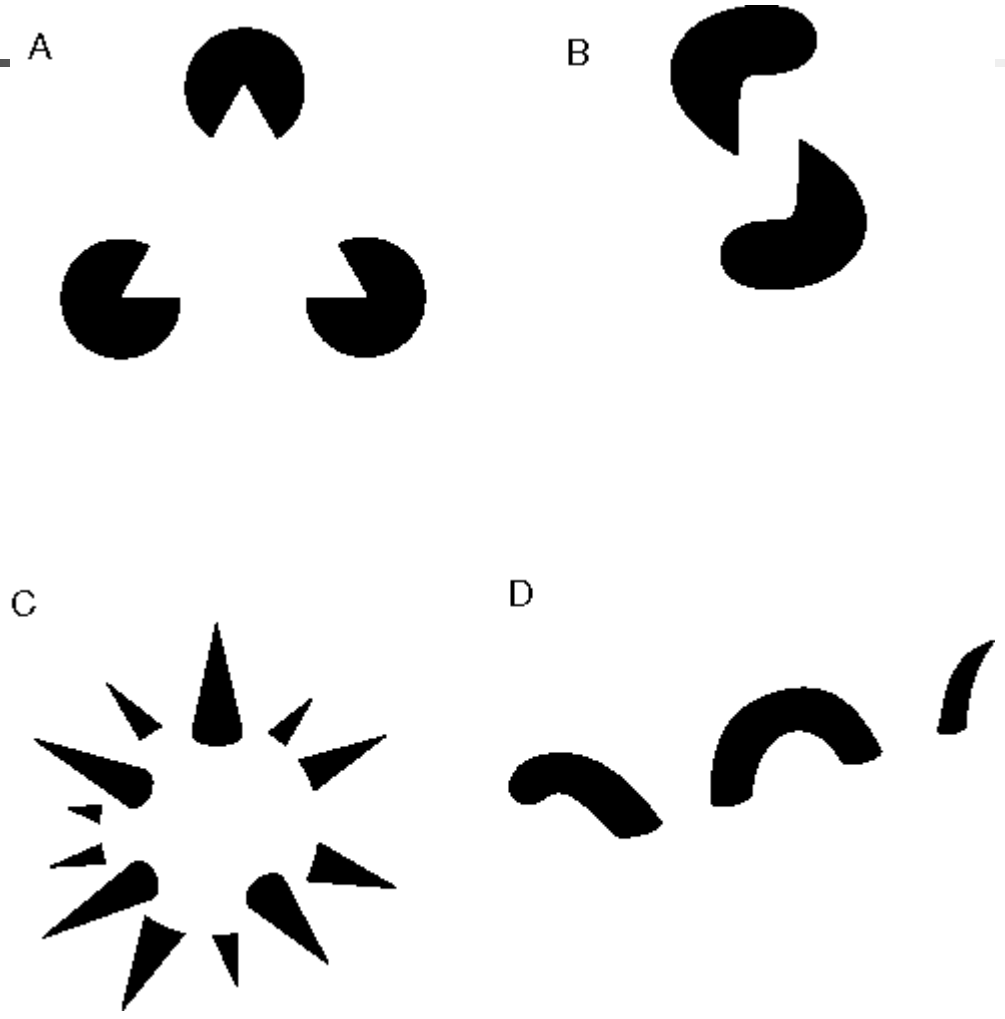


Stressing the invisible groupings:

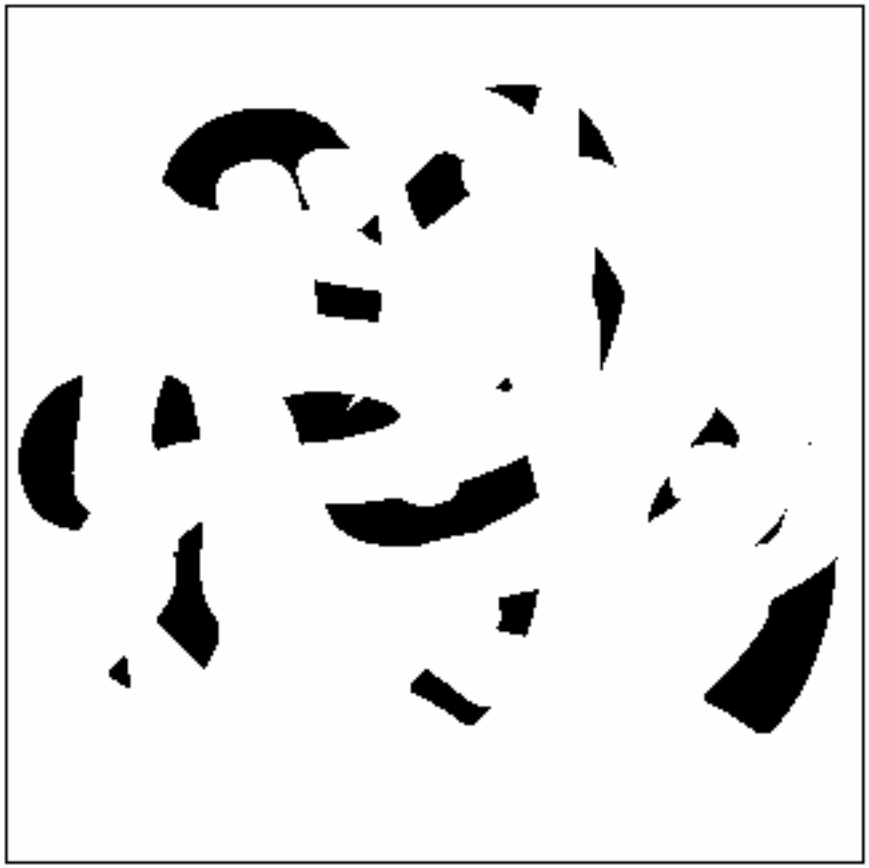


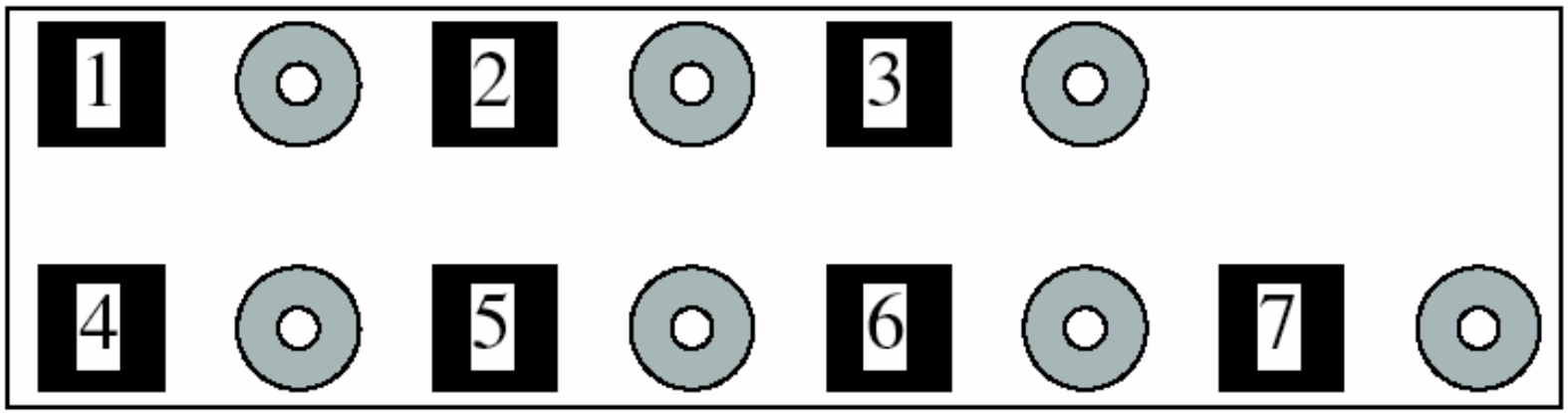
* Images from Steve Lehar's Gestalt papers: http://cns-alumni.bu.edu/pub/slehar/Lehar.html

Groupings by Invisible Completions



* Images from Steve Lehar's Gestalt papers: http://cns-alumni.bu.edu/pub/slehar/Lehar.html




1
2
3
4
5
6
7

Segmentation and Grouping

- Motivation:
  - for recognition
  - for compression
- Obtain a compact representation from an static or dynamic sequence/set of tokens
- Always for a goal or application
- Broad theory is absent at present

Segmentation breaks an image into groups over space and/or time

Segmentation and Grouping

Tokens are

- The things that are grouped (pixels, points, surface elements, etc., etc.)

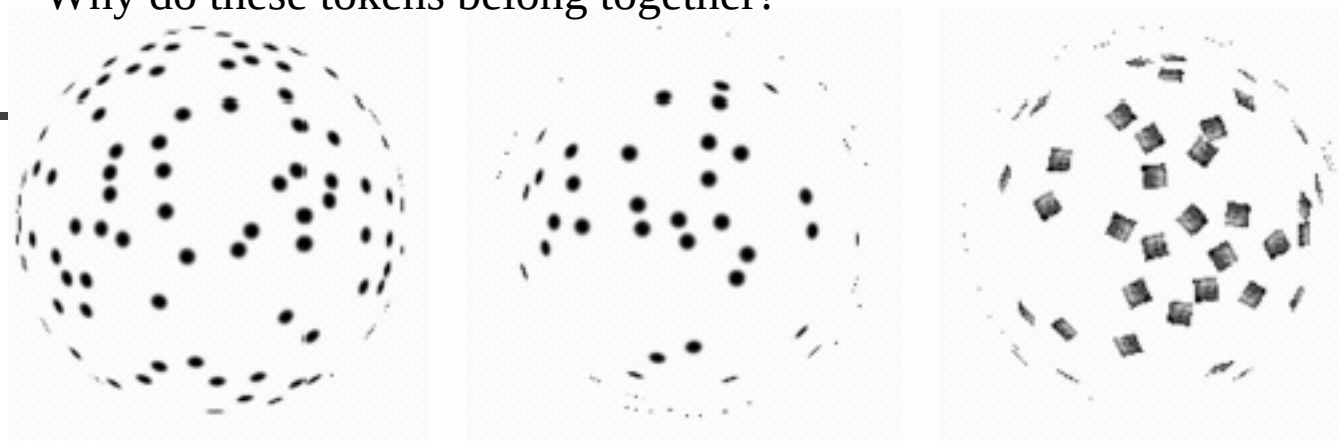
- top down segmentation
  - tokens grouped because they lie on the same object
- bottom up segmentation
- tokens belong together because of some local affinity measure
- Bottom up/Top Down need not be mutually exclusive

Segmentation and Grouping

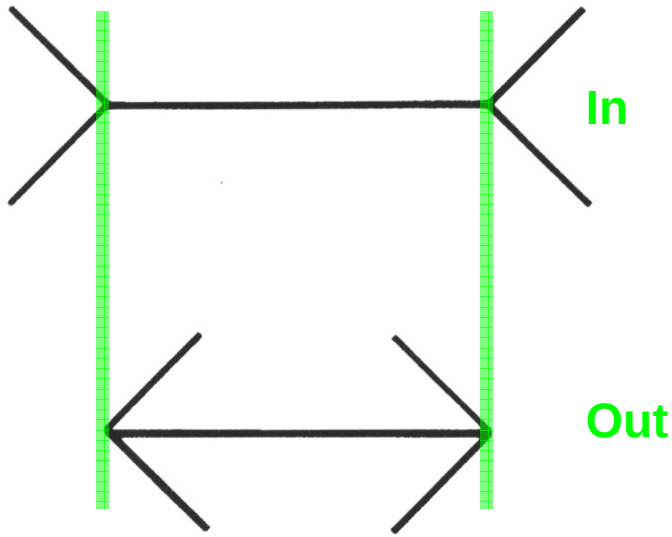
- Gestalt Psychology(完形心理学)
  - elements in a collection of elements can have properties that result from relationships (Muller-Lyer effect)
  - A series of factors affect whether elements should be grouped together
- Edge extraction: grouping and completion
- Image segmentation

Here, the 3D nature of grouping is apparent:

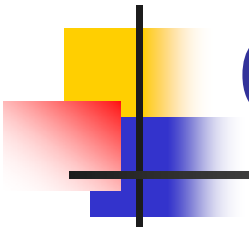
Why do these tokens belong together?



Corners and creases in 3D, length is interpreted differently:



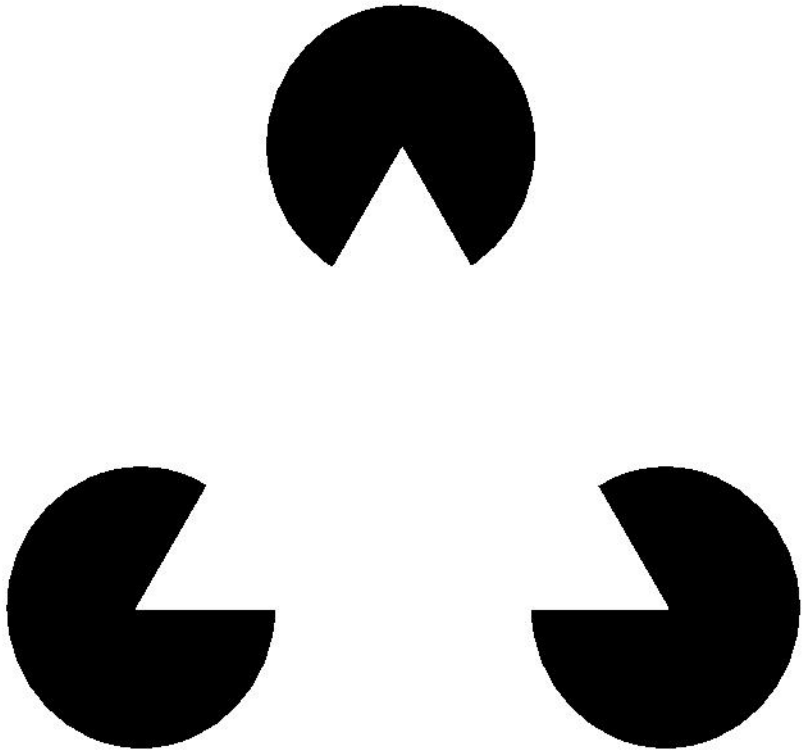
The (in) line at the far end of corridor must be longer than the (out) near line if they measure to be the same size

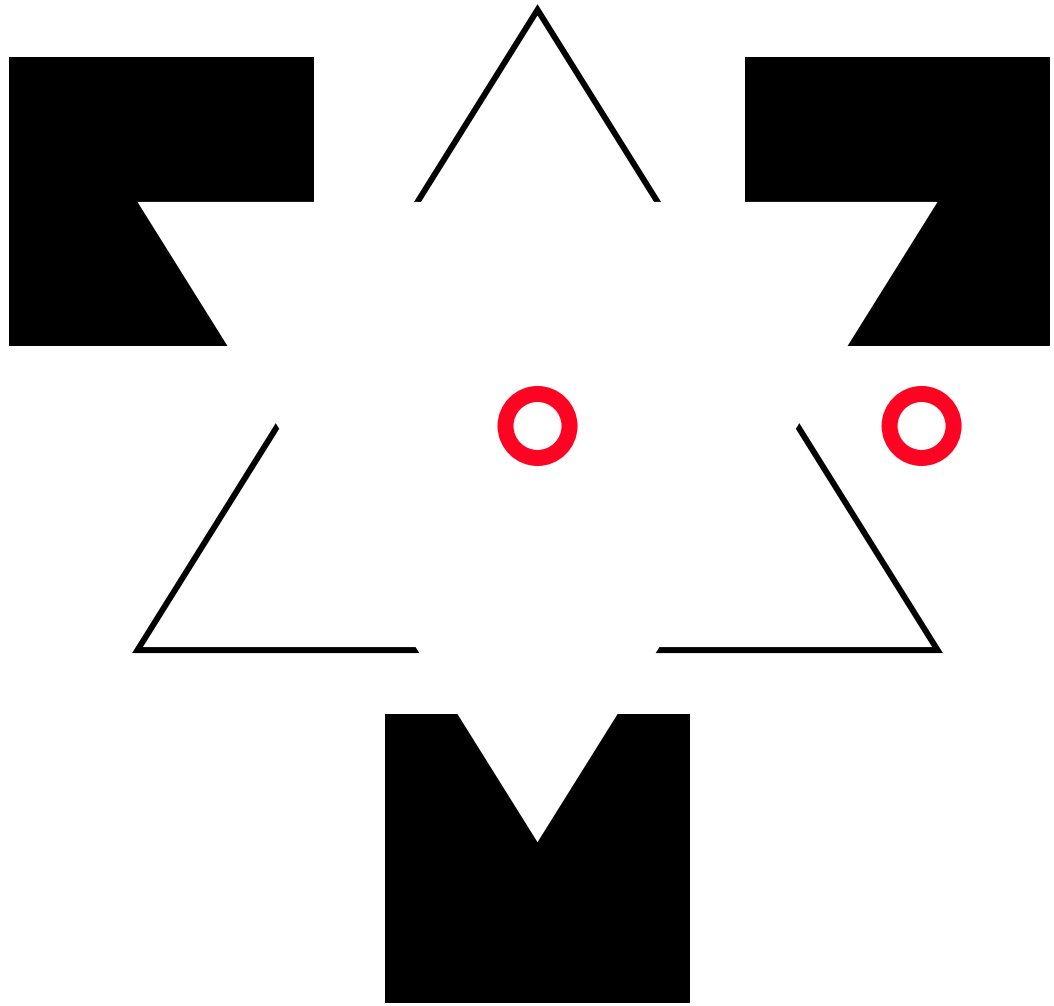


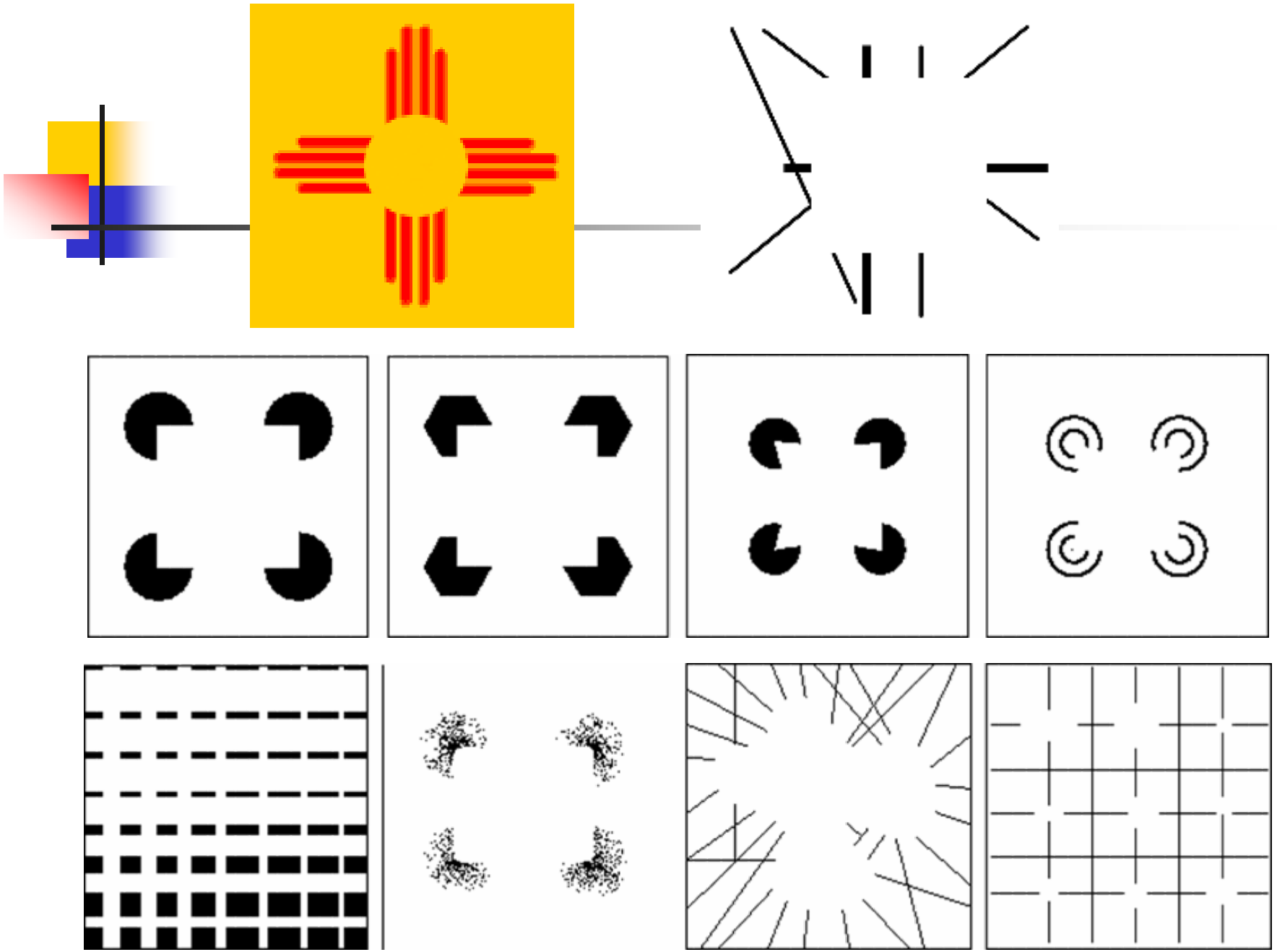
General Ideas

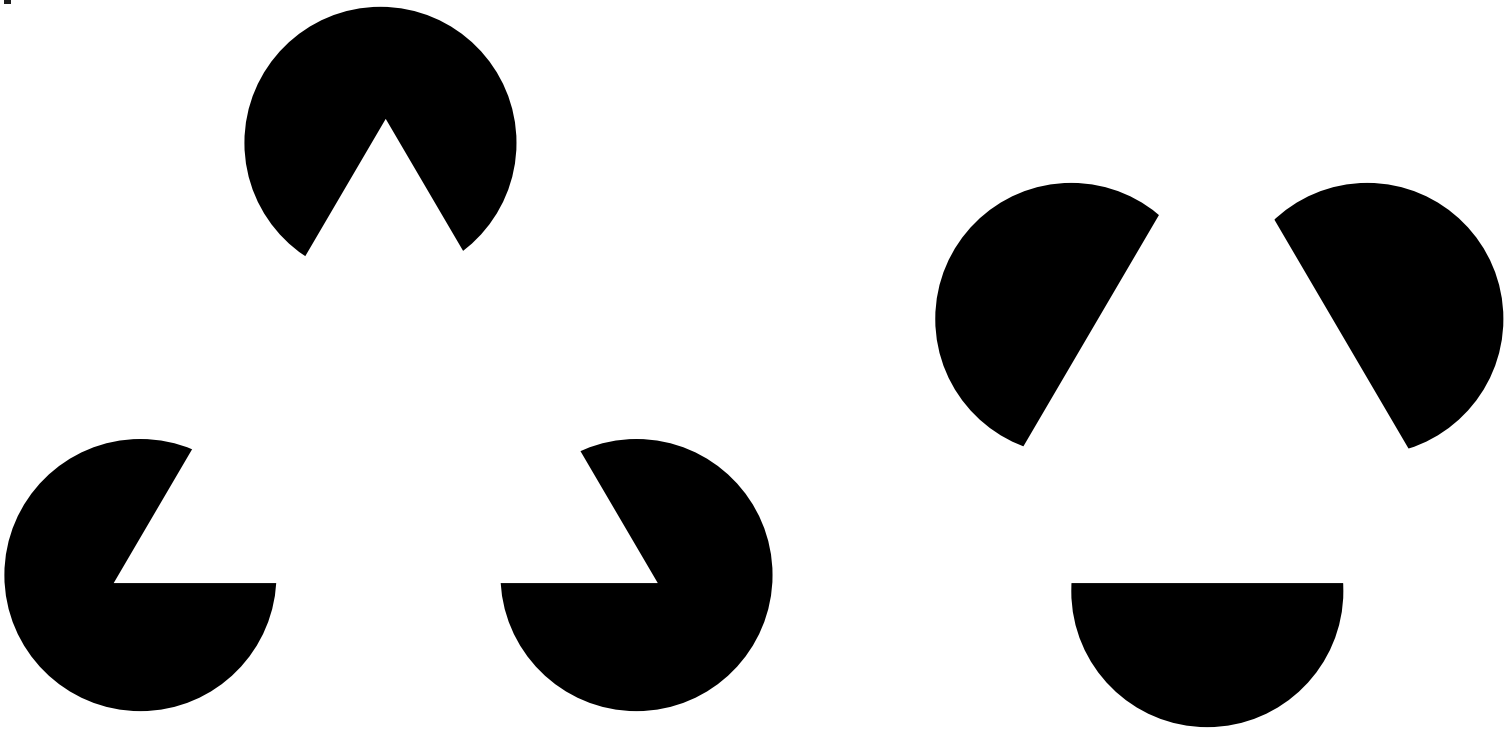
- Grouping (or clustering)
  - collect together tokens that “belong together”
- Fitting
  - associate a model with tokens
  - issues
    - which model?
    - which token goes to which element?
    - how many elements in the model?

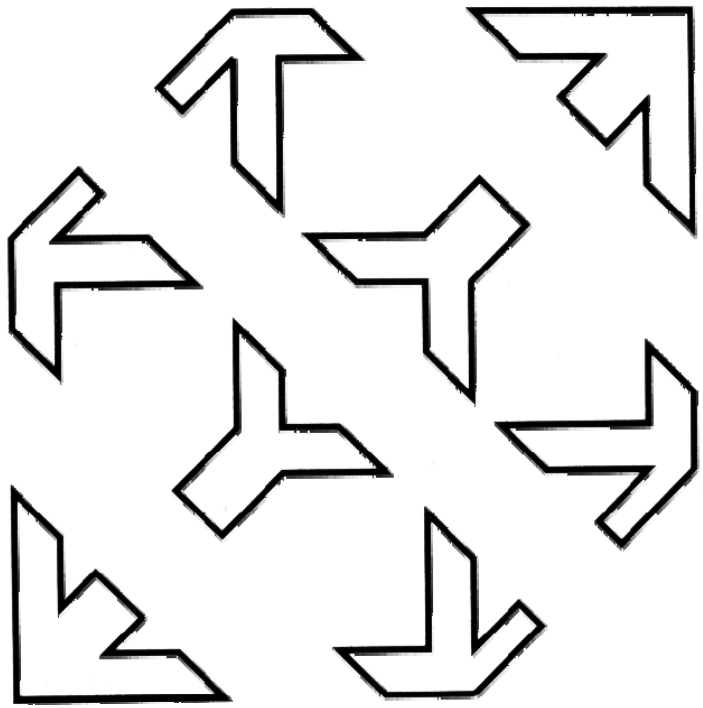
Kanizsa Triangle

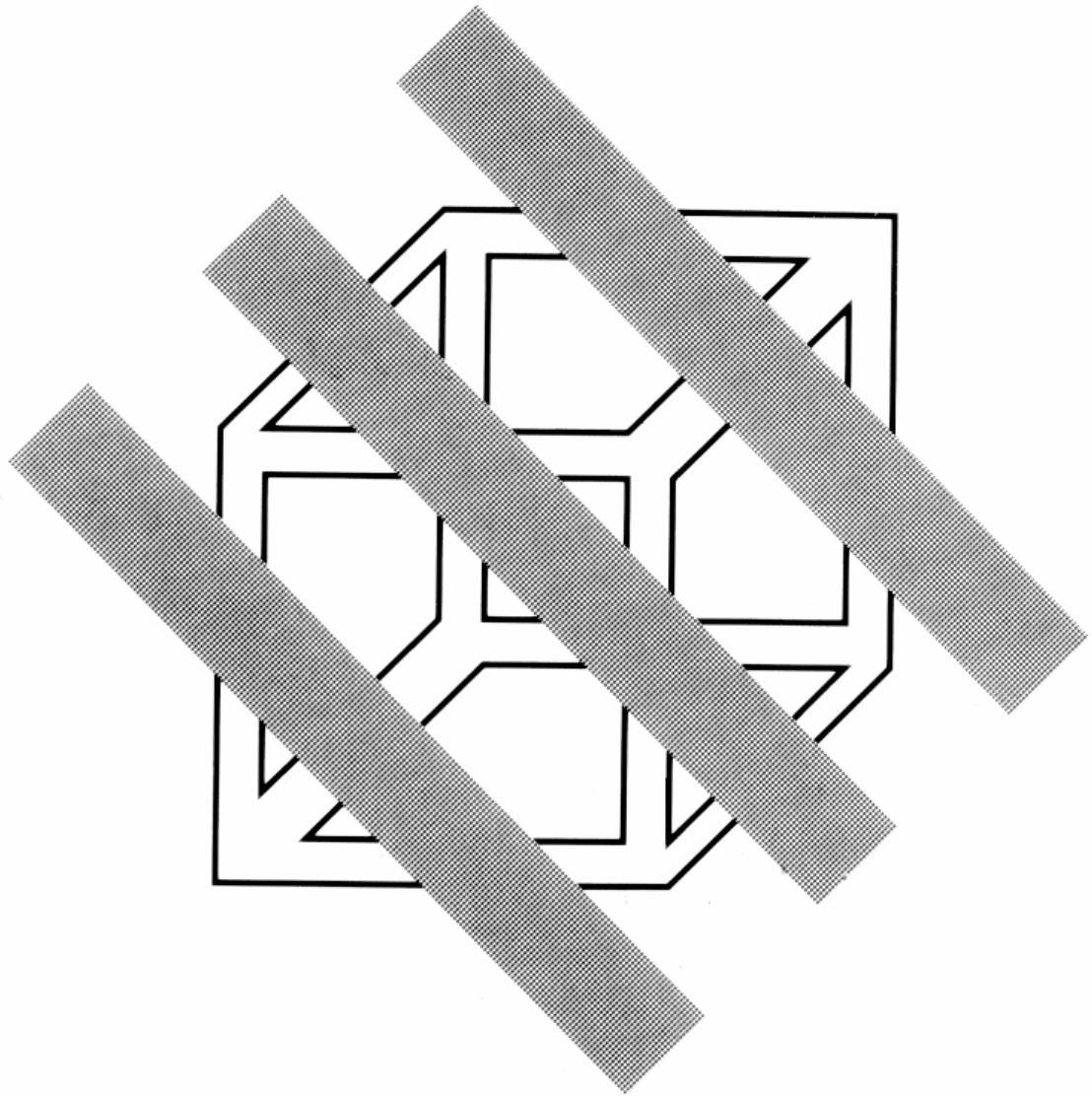


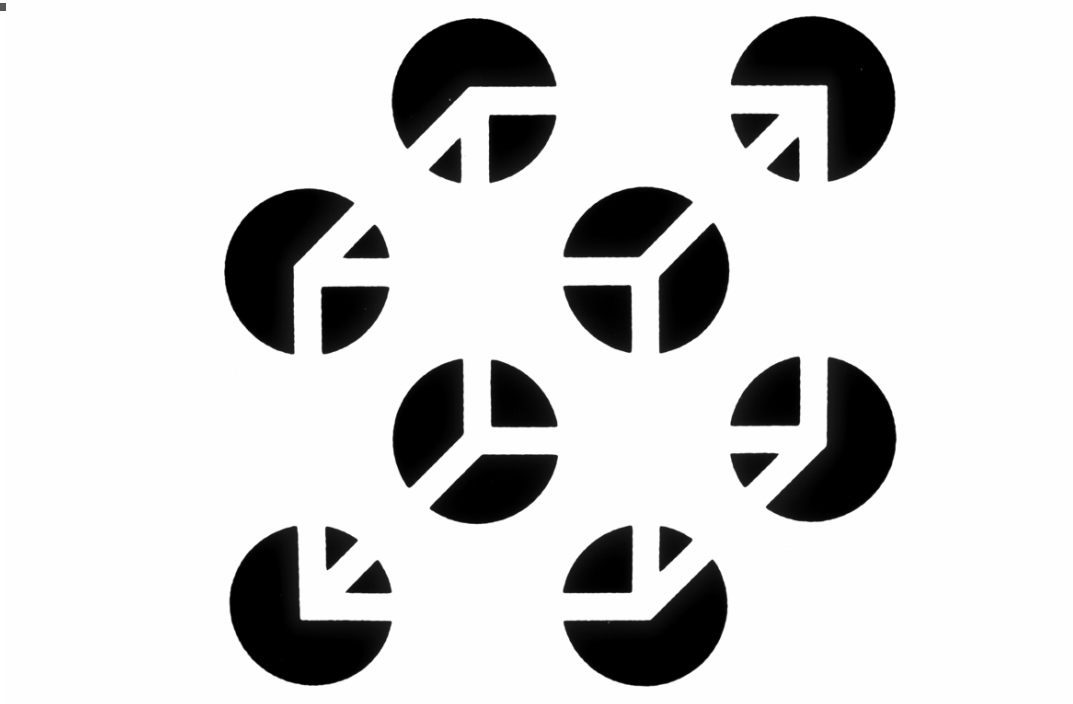












Perceptual Organization

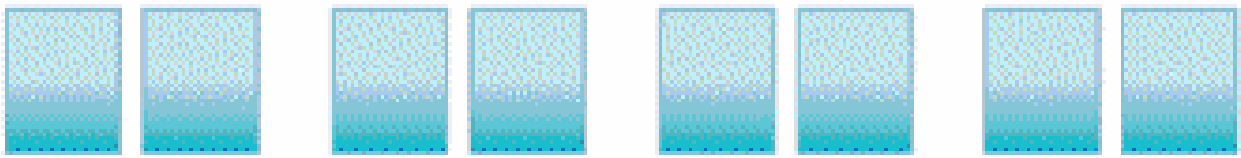
- Atomism, reductionism:
  - Perception is a process of decomposing an image into its parts.
  - The whole is equal to the sum of its parts.
- Gestalt (Wertheimer, Köhler, Koffka 1912)
  - The whole is larger than the sum of its parts.

Mona Lisa

Mona Lisa

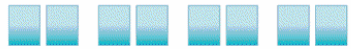
Gestalt Principles (格斯塔原理)

- Proximity

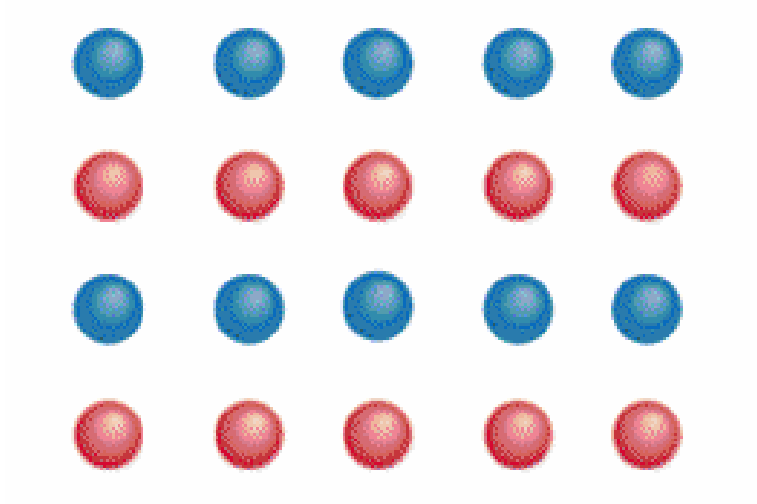


Gestalt Principles

- Proximity



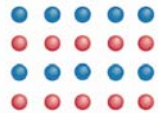
- Similarity



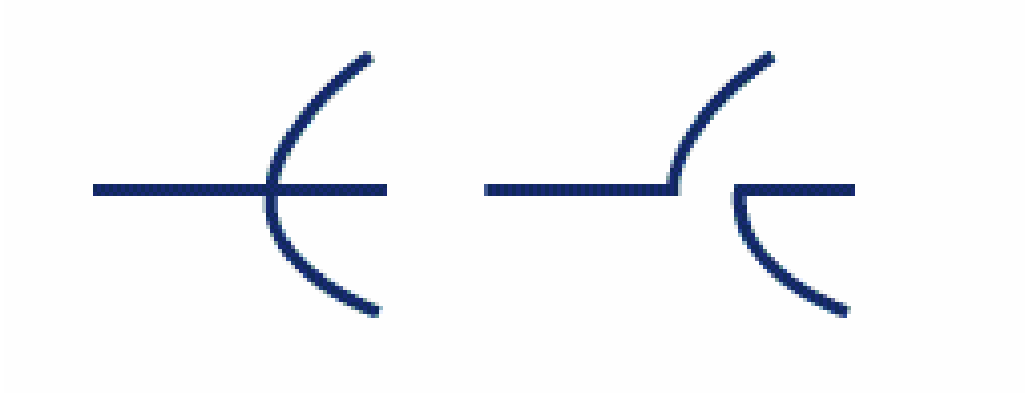
Gestalt Principles

- Proximity

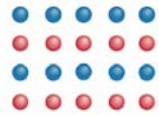
- Similarity

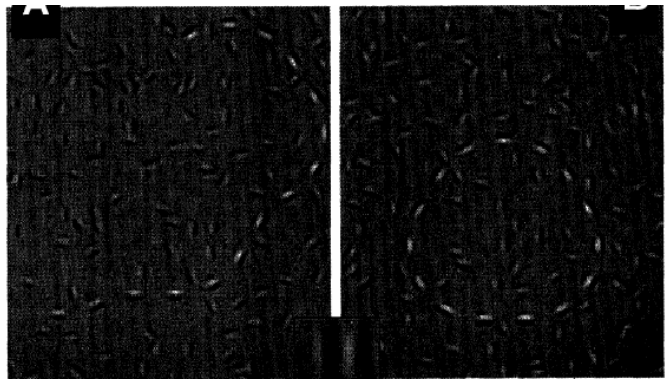


- Continuity



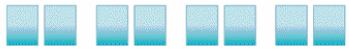
Gestalt Principles

- Proximity 
- Similarity 
- Continuity 
- Closure

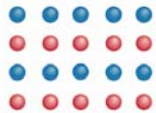


Gestalt Principles

- Proximity

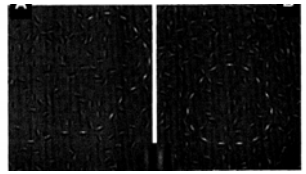


- Similarity



- Continuity

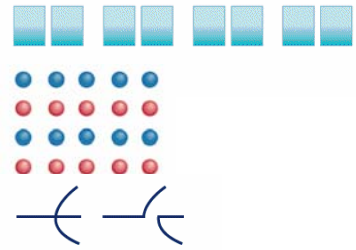
- Closure



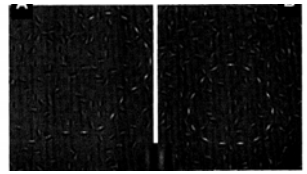
- Common Fate

Gestalt Principles

- Proximity
- Similarity
- Continuity

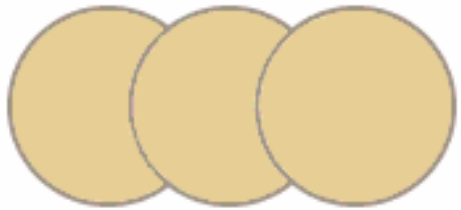


- Closure



- Common Fate

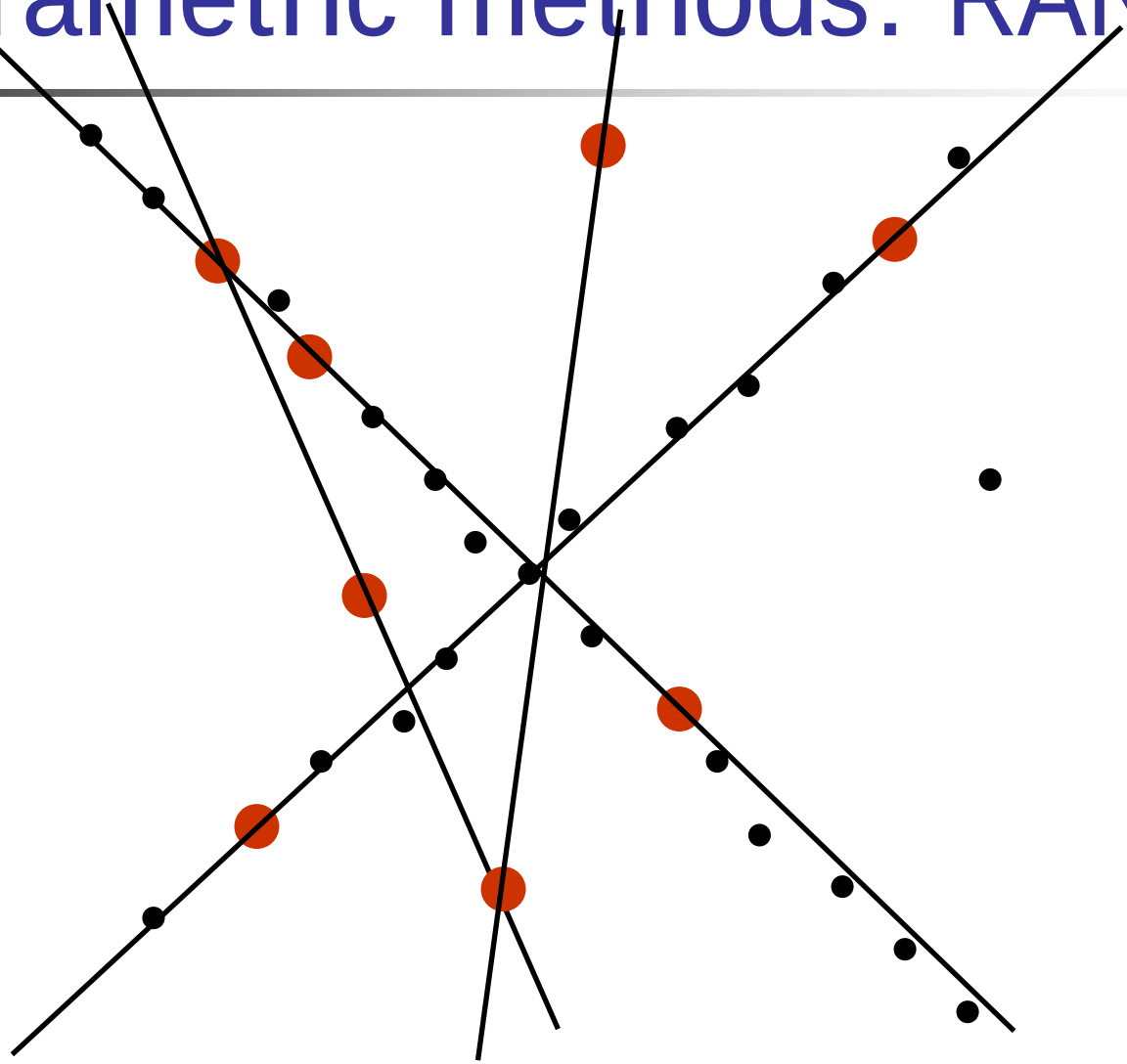
- Simplicity



Smooth Completion

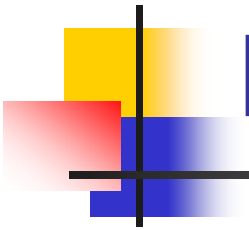
- Isotropic
- Smoothness
- Minimal curvature
- Extensibility

Parametric methods: RANSAC



RANSAC

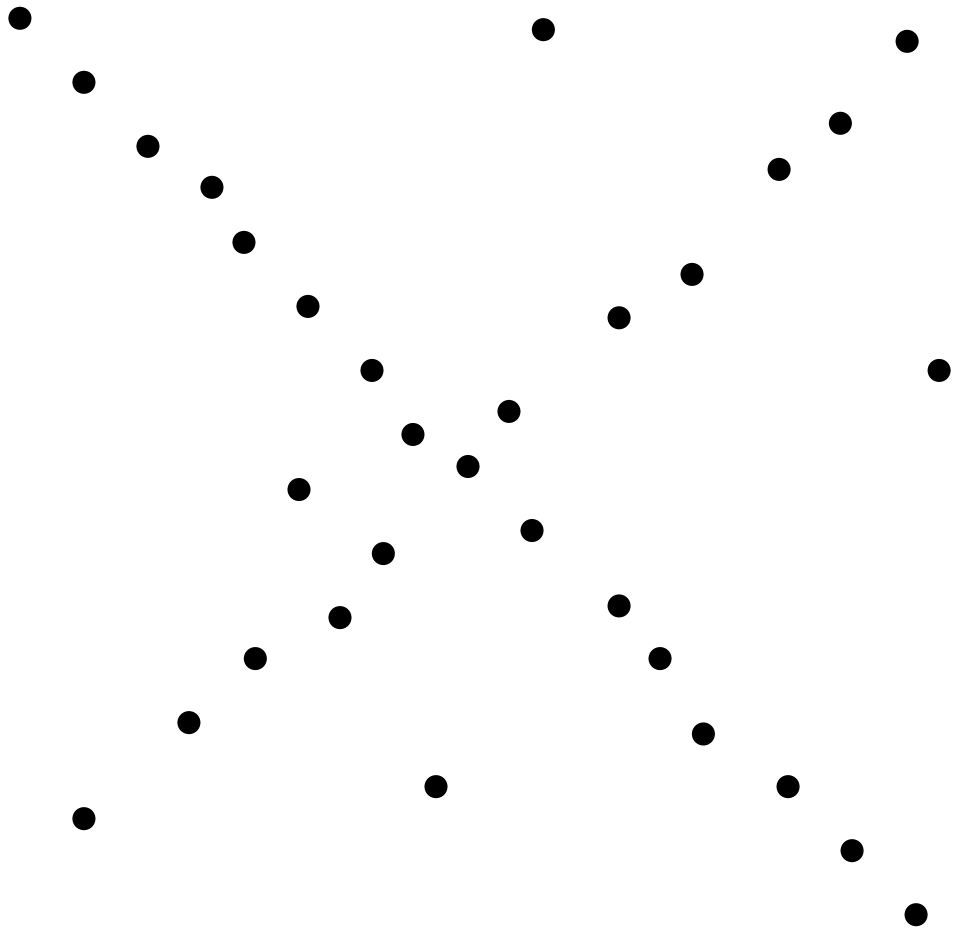
- RANdom SAmple Concensus
- Complexity:
  - Need to go over all pairs: $O(n^2)$
  - For each pair check how many more points are consistent: $O(n)$
  - Total complexity: $O(n^3)$



RANSAC

- Another application of RANSAC:
  Find transformation between images
- Example: compute homography
  - Compute homography for every 4 pairs of corresponding points
  - Choose the homography that best explains the image
  - $m^4 n^4$ sets should be tested
- Another example: compute epipolar lines
  - How many correspondences are needed?

Hough Transform

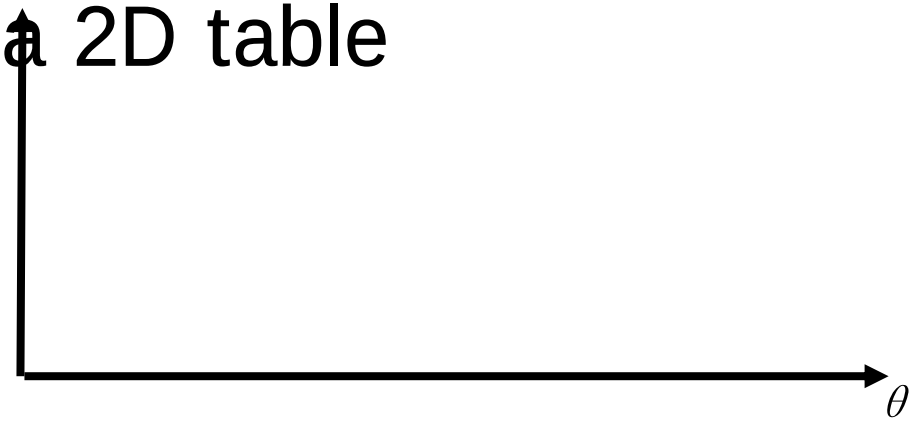


Hough Transform

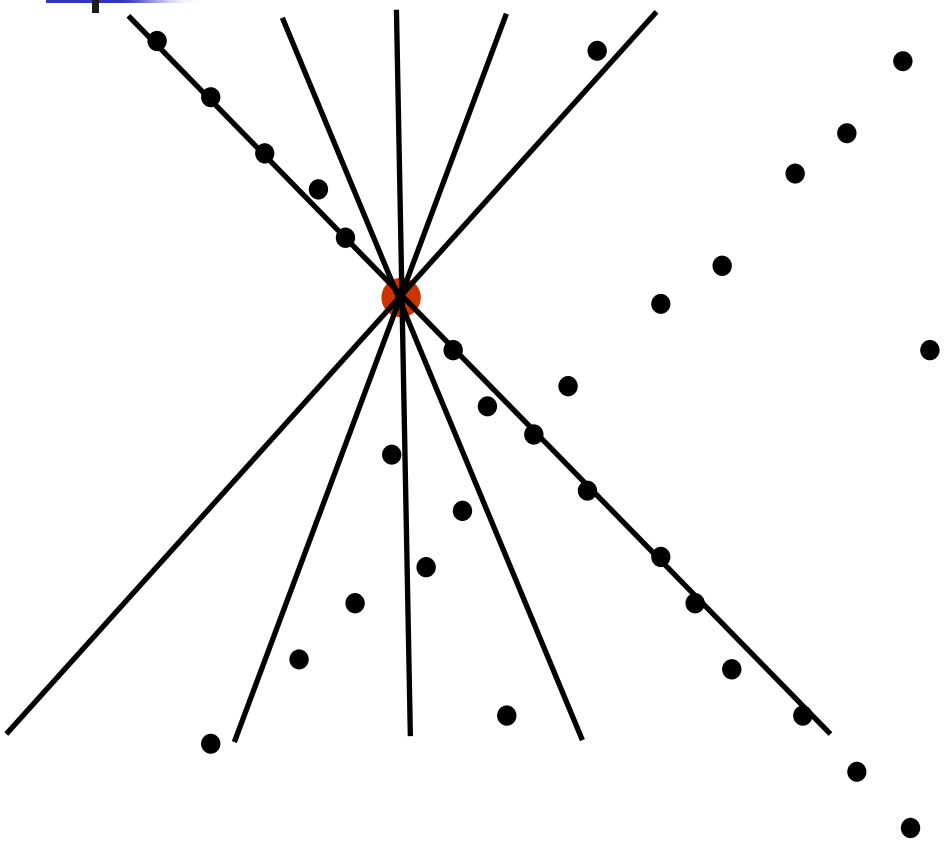
- Linear in the number of points
- Describe lines as

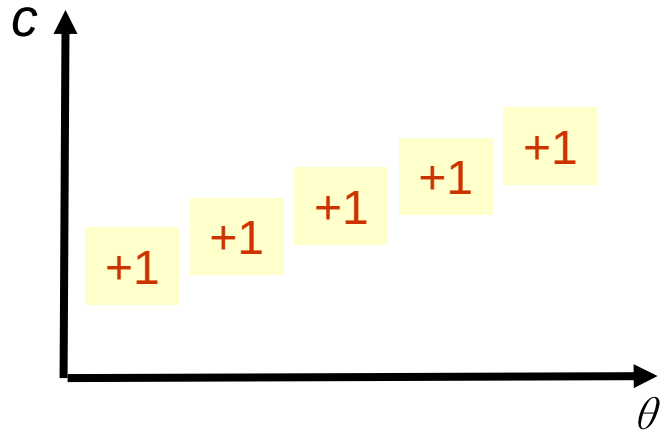
$$y = mx + n$$

- Or better $x\cos\theta + y\sin\theta = c$
- Prepare a 2D table

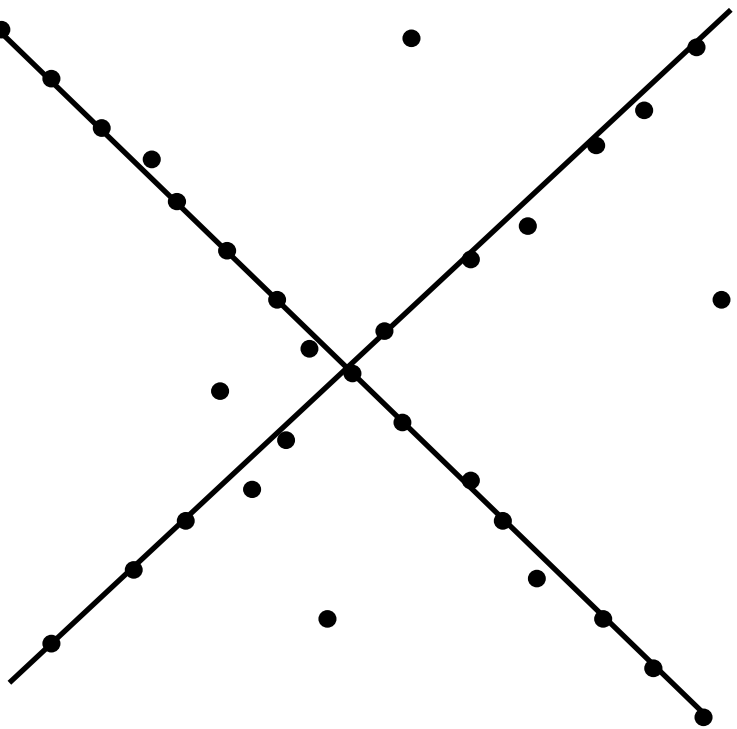


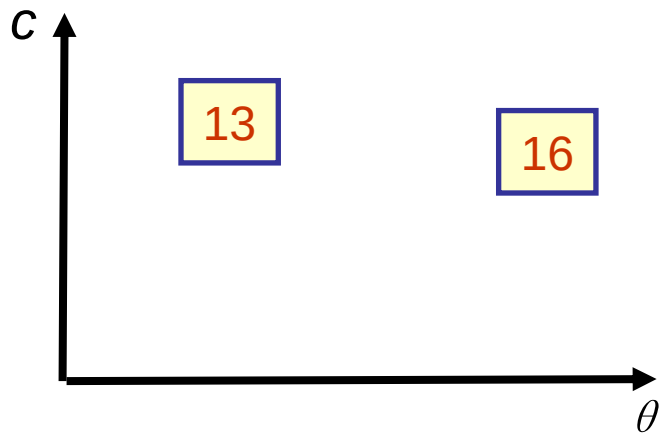
Hough Transform



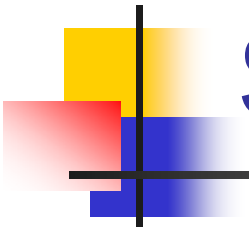


Hough Transform





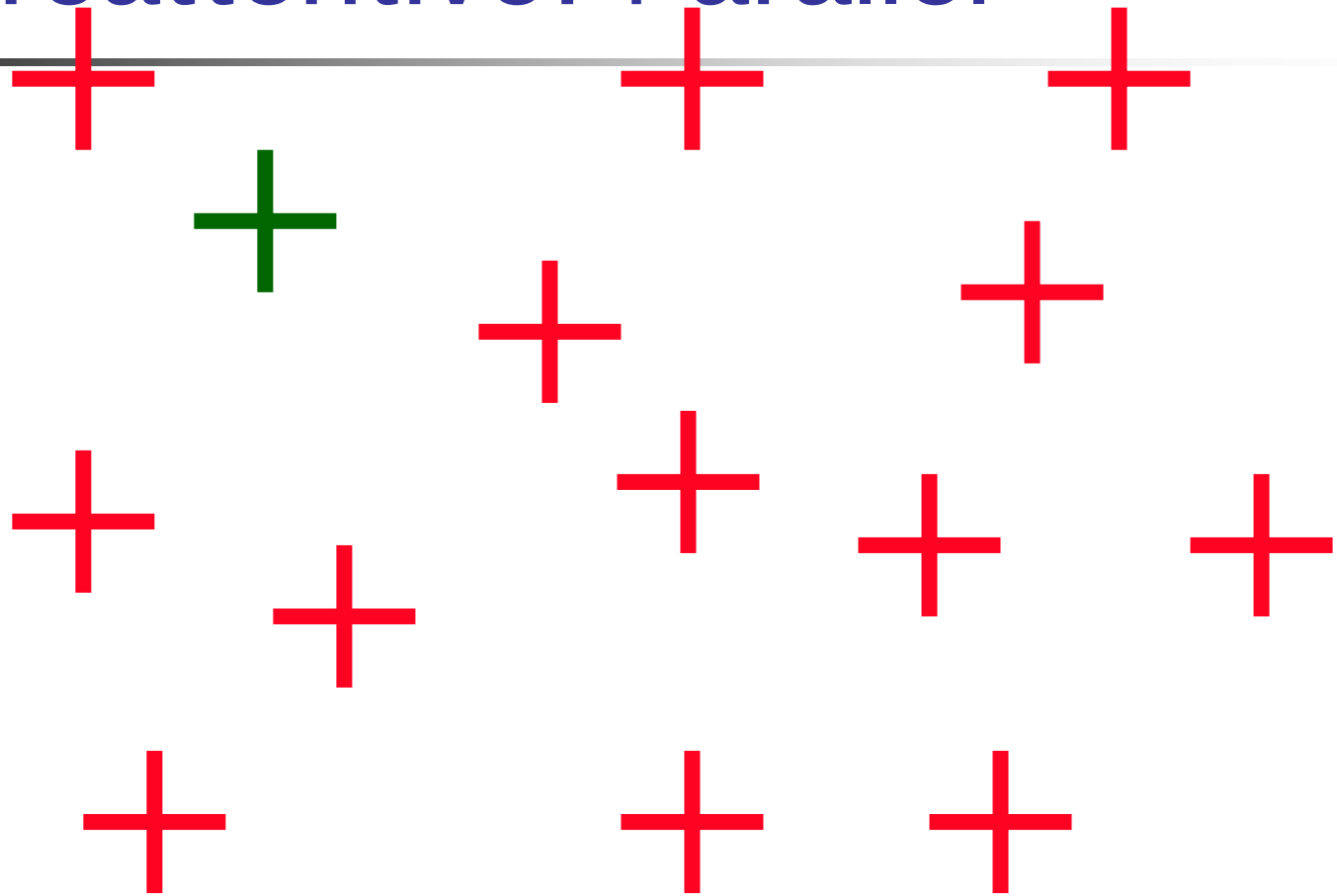
What if we want to find circles?



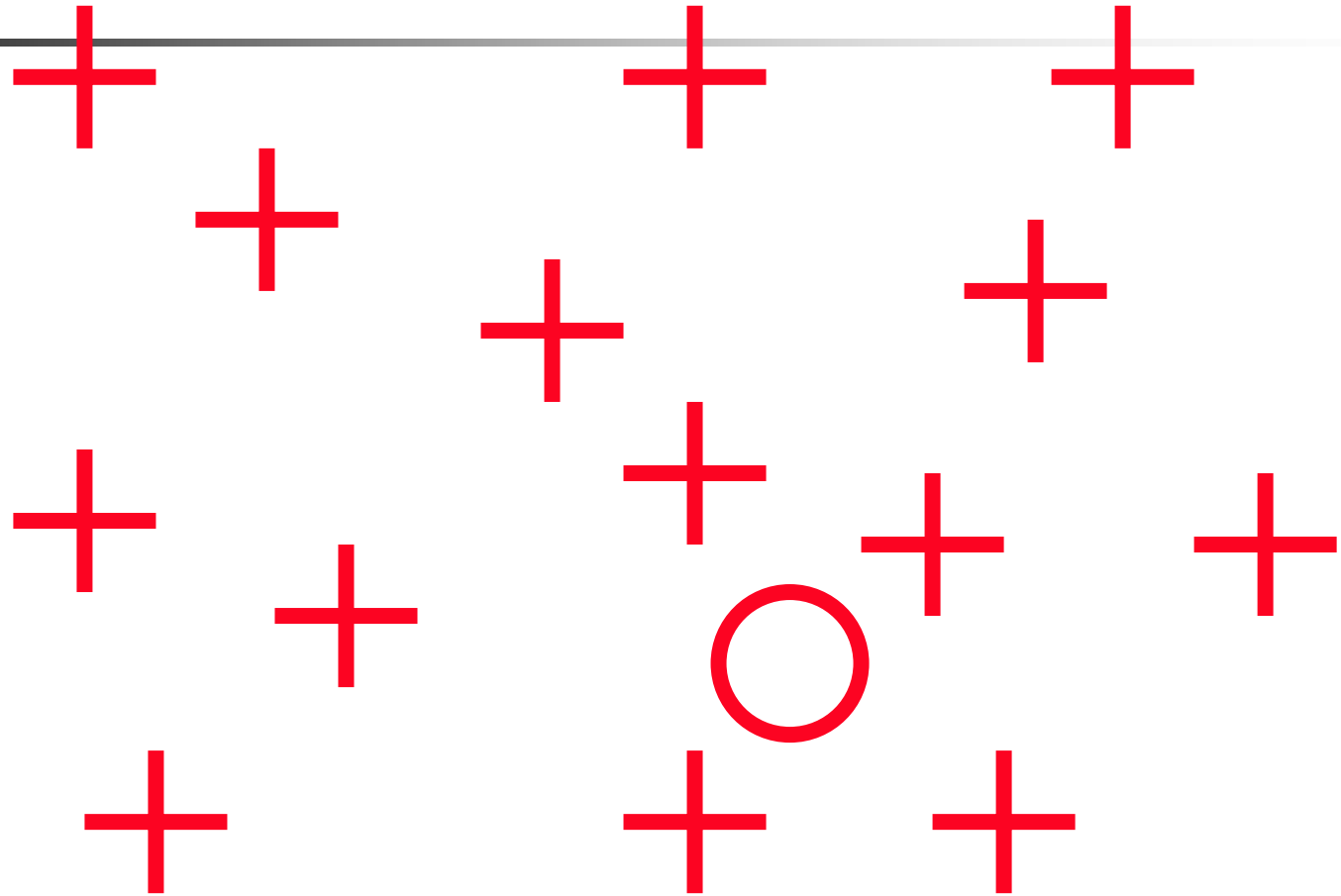
Summary

- Local processing is often insufficient to separate objects
- We reviewed several approaches for
  - curve extraction, completion
  - region segmentation

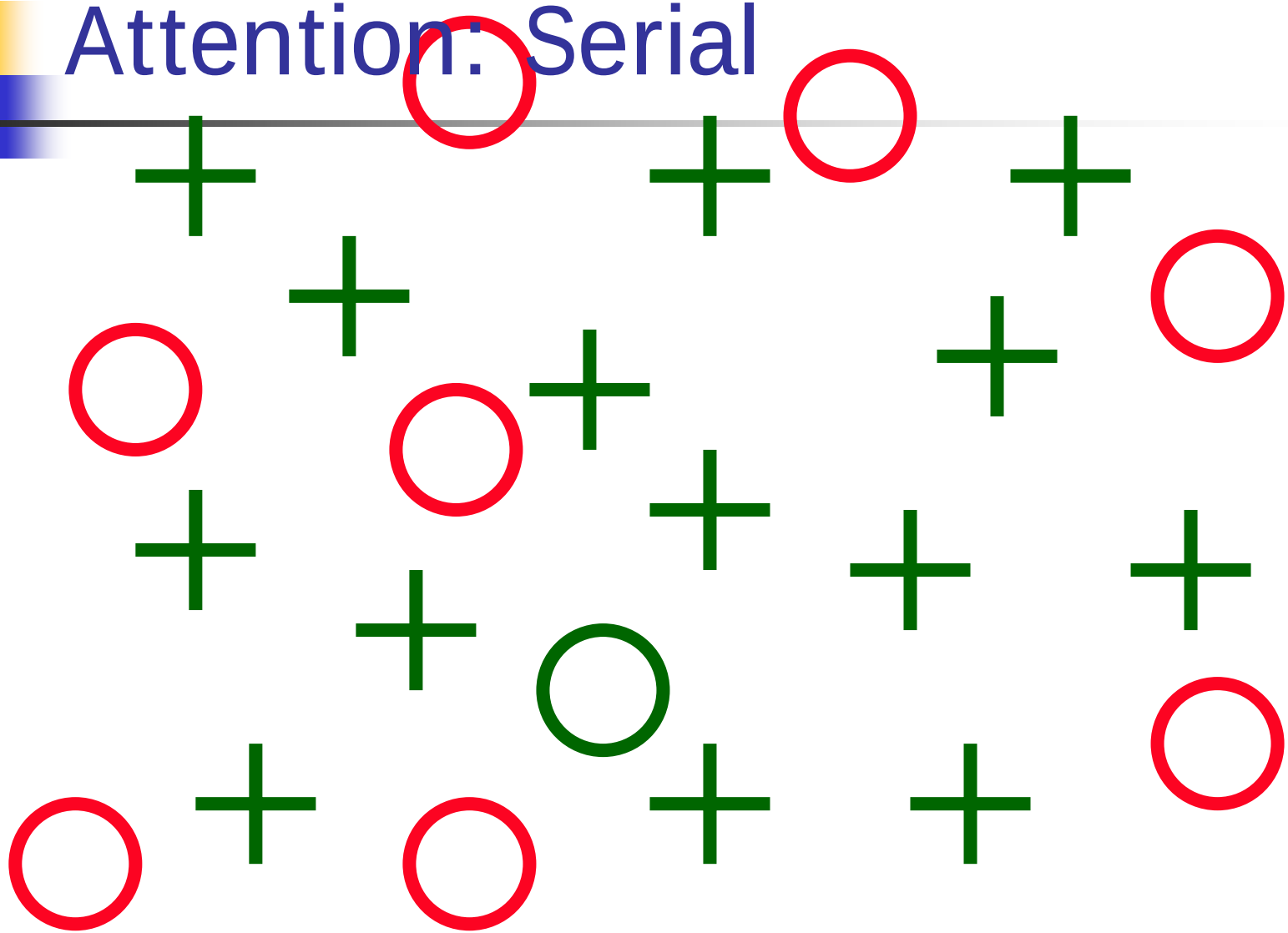
Preattentive: Parallel



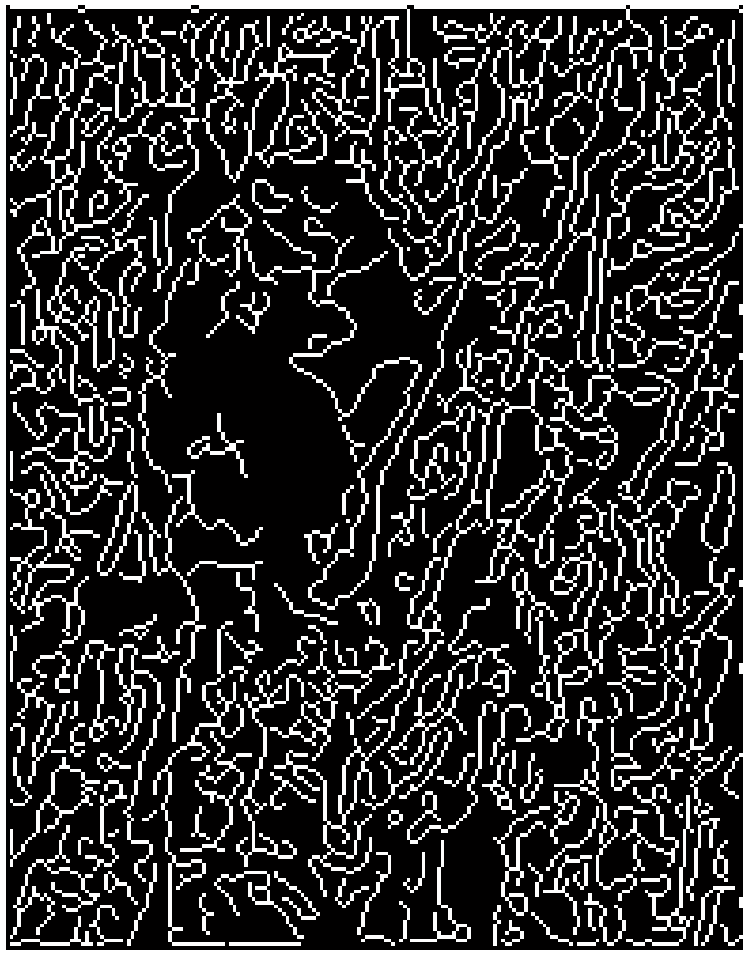
Preattentive: Parallel

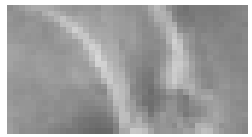


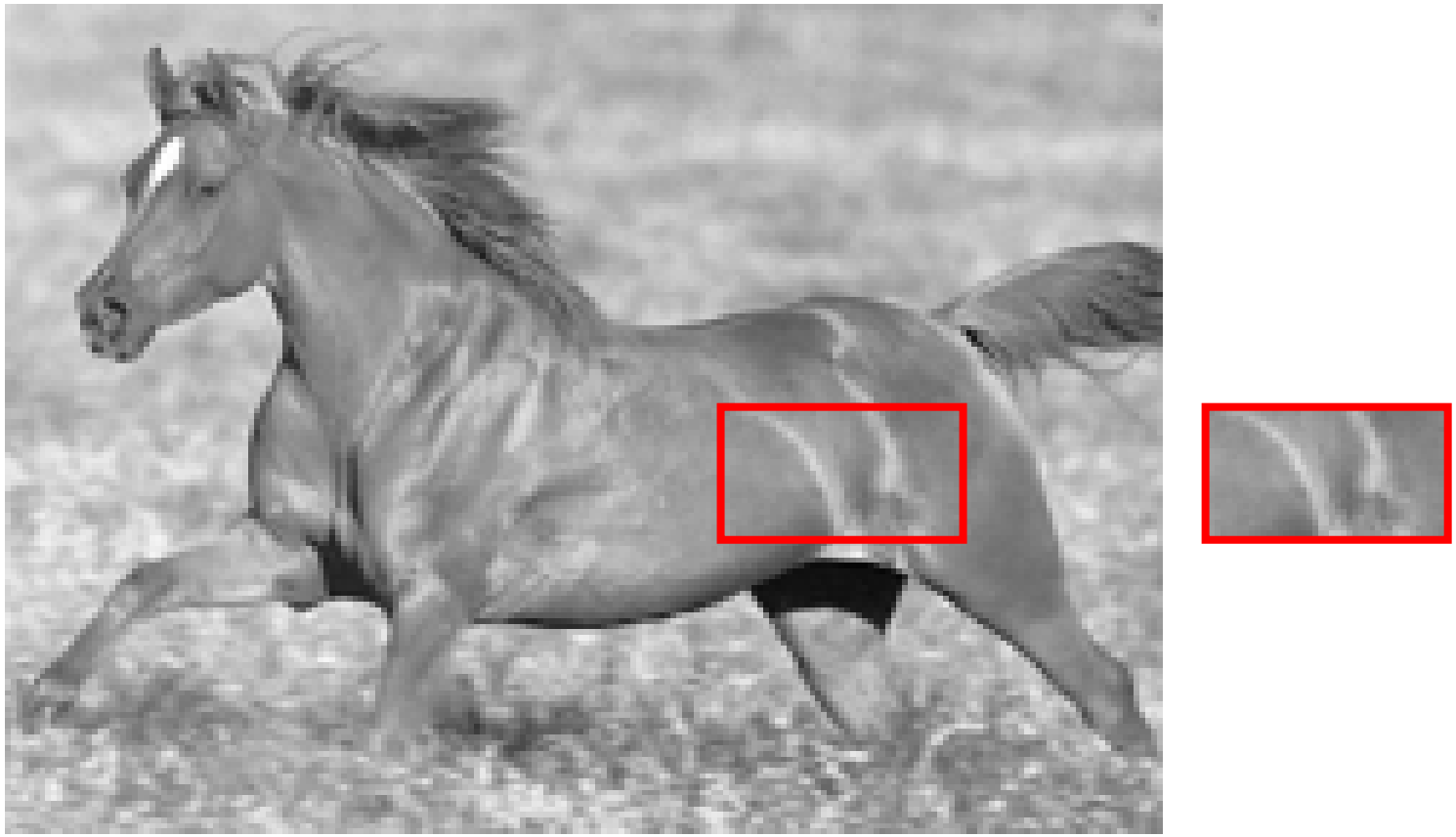
Attention: Serial

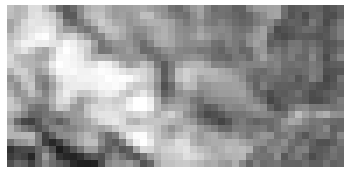


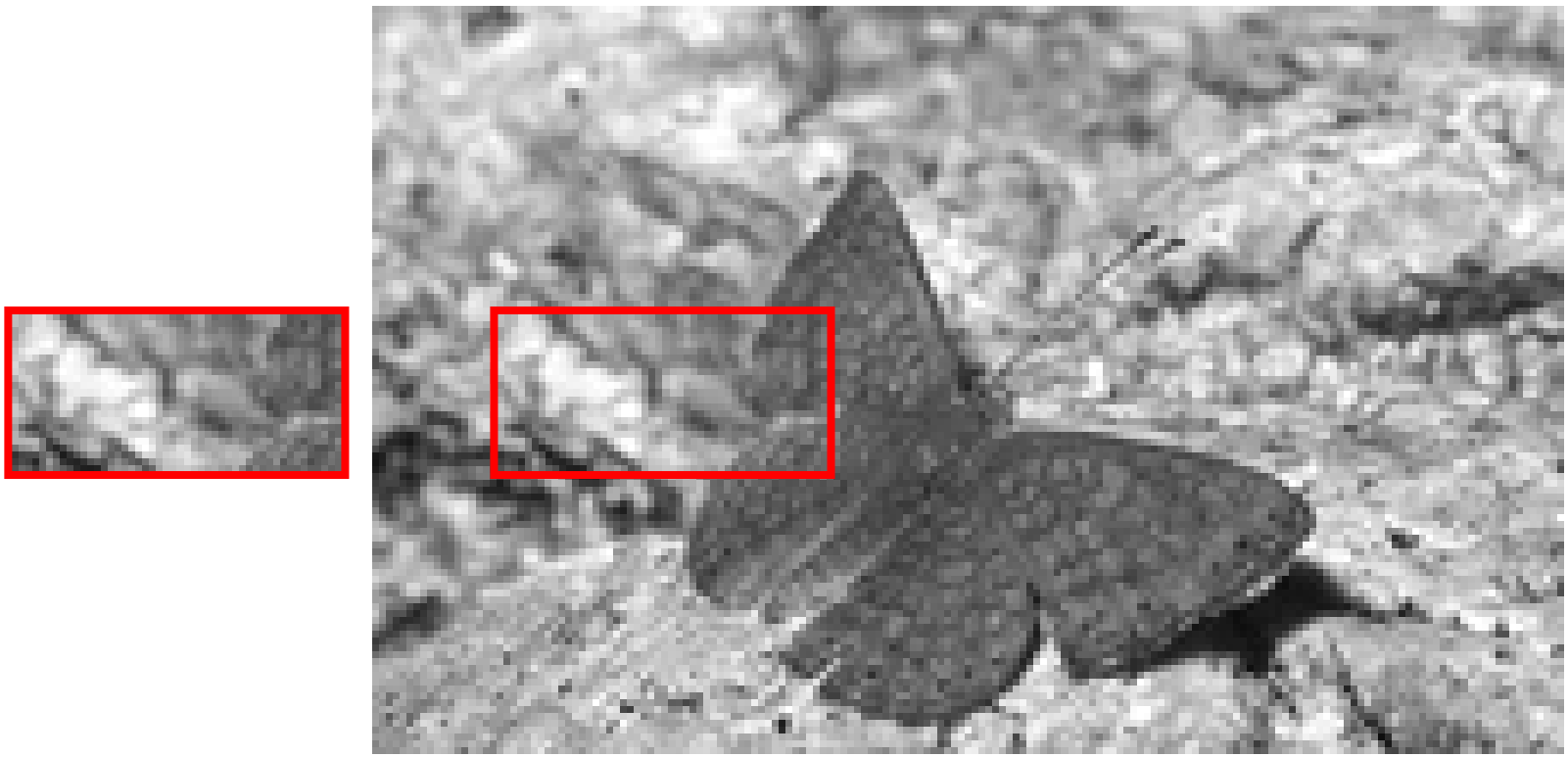
Camouflage(伪装)

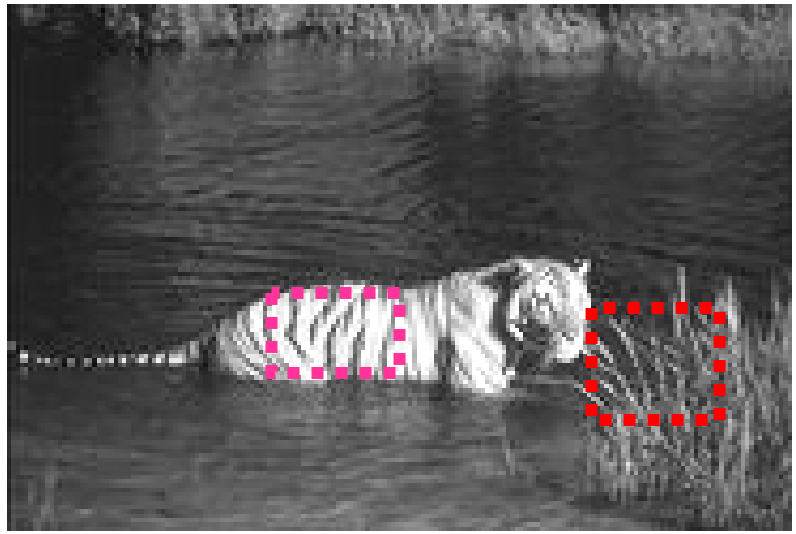


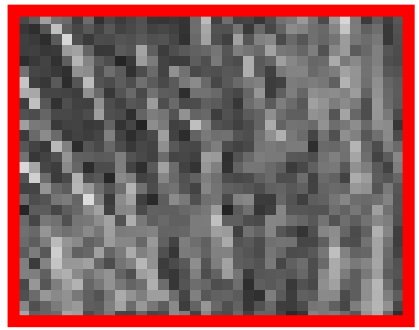


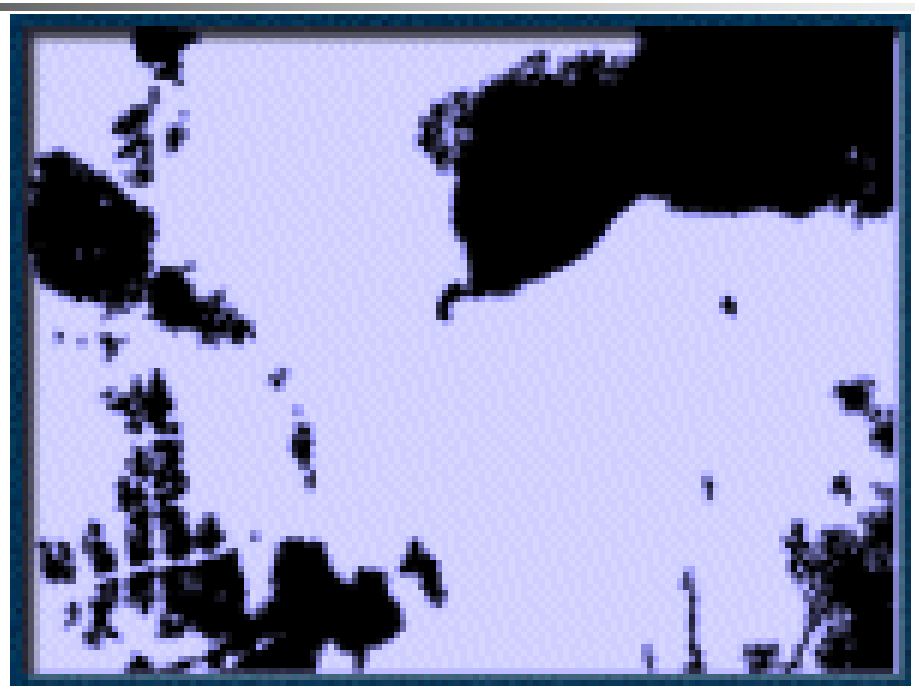


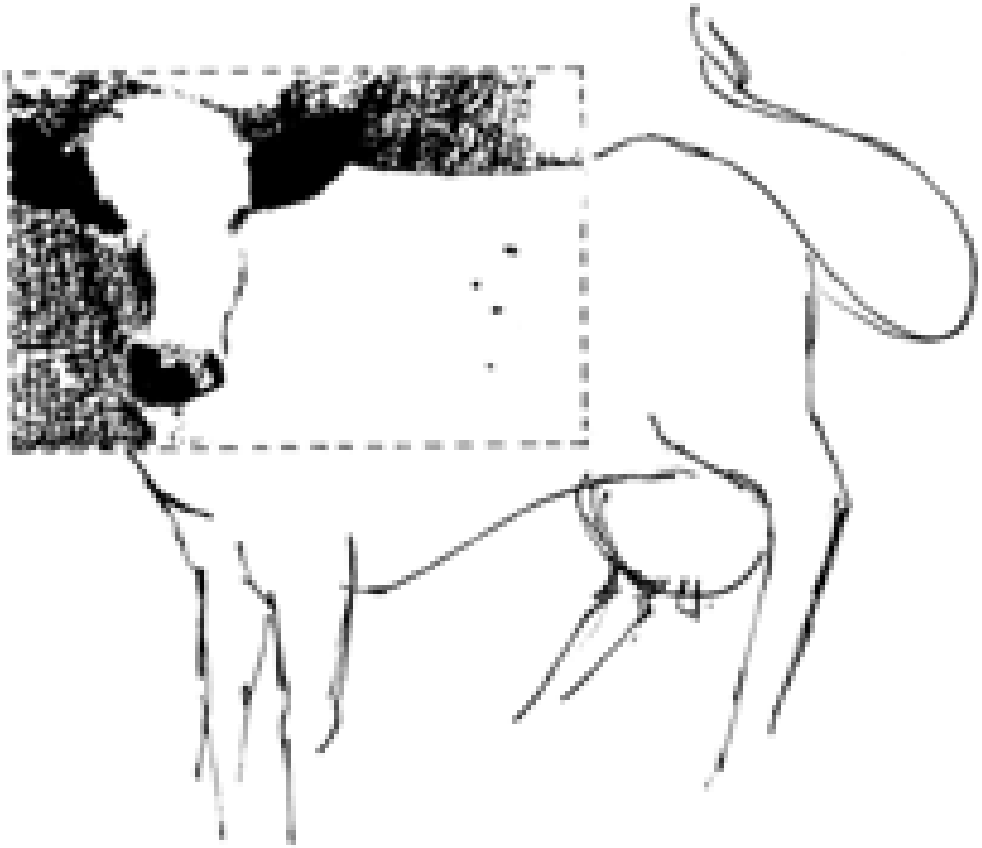






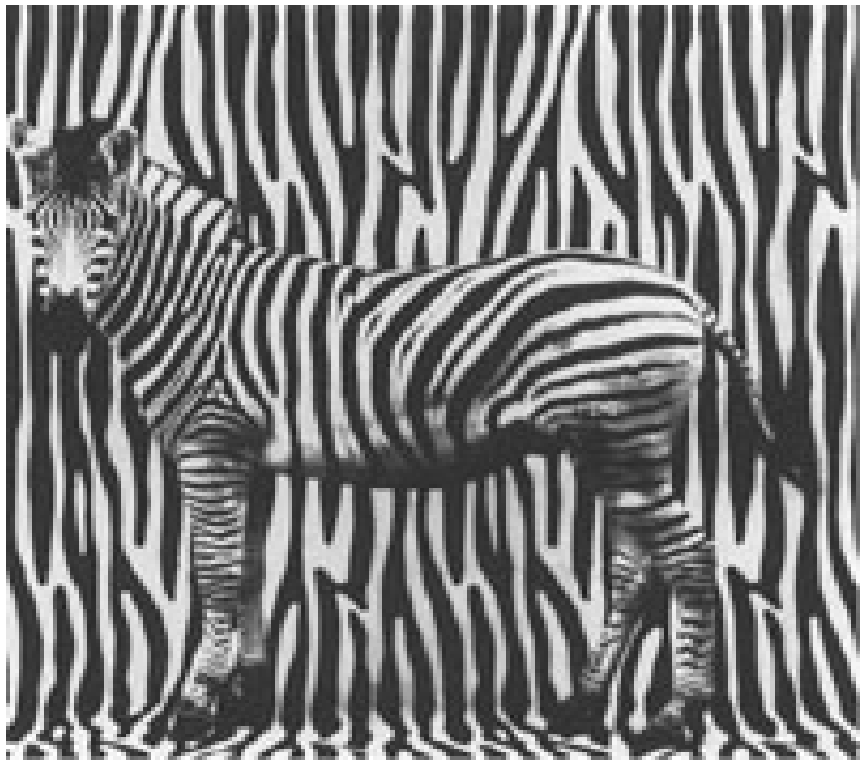


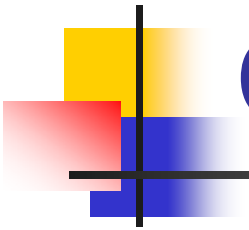




The famous invisible dog eating under a tree:

A Final Example

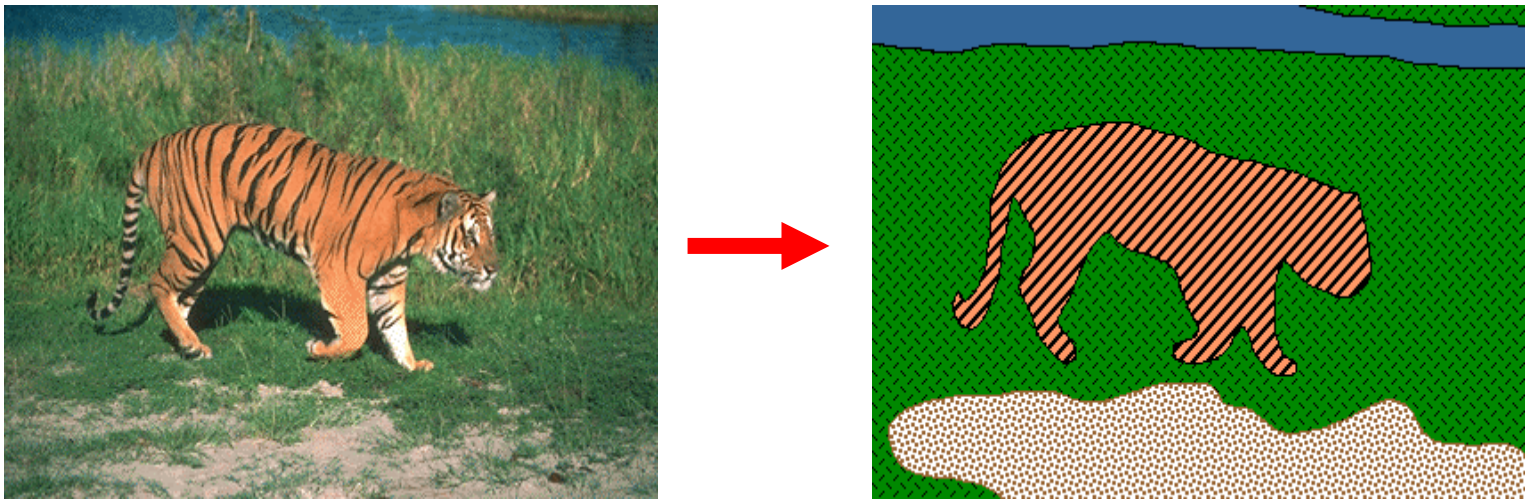




Outline

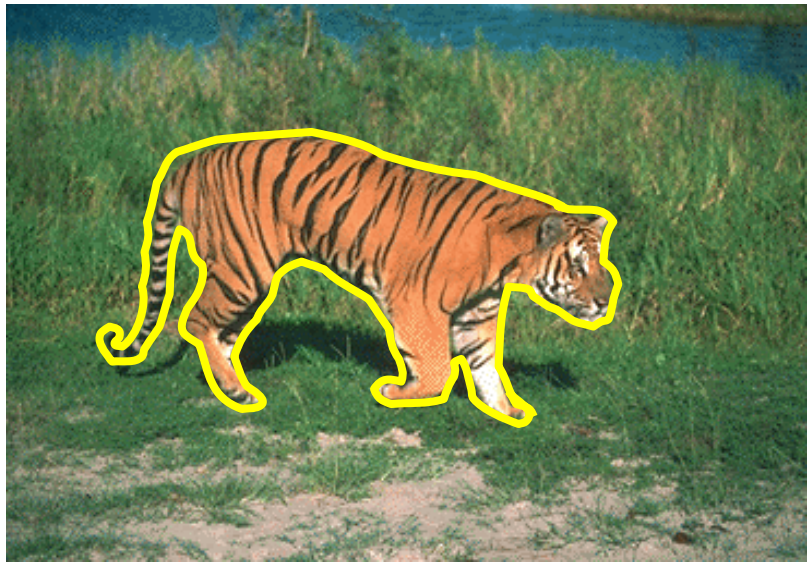
- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph Cuts

From images to objects



- What Defines an Object?
  - Subjective problem, but has been well-studied
  - Gestalt Laws seek to formalize this
    - proximity, similarity, continuation, closure, common fate

Extracting objects



- How could this be done?

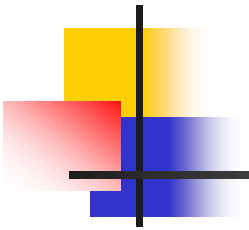
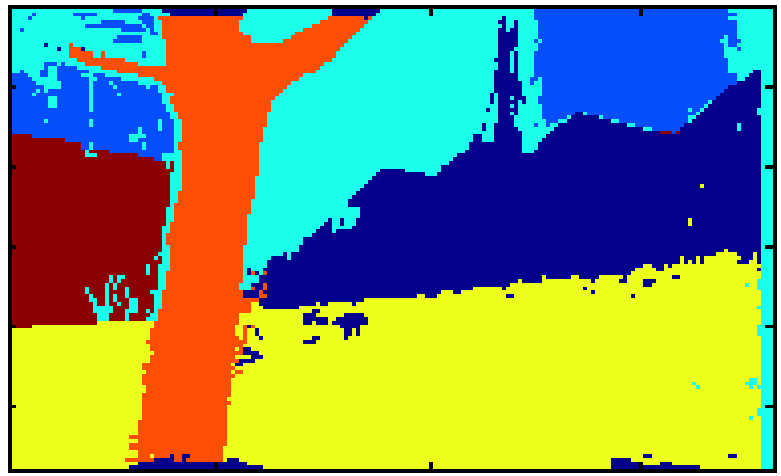


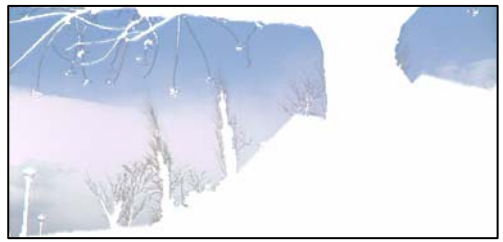
Image Segmentation

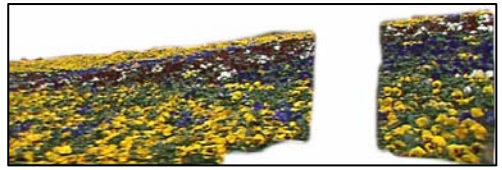
- Many approaches proposed
  - cues: color, texture, regions, contours…
  - automatic vs. user-guided
  - no clear winner
- we'll consider several approaches today

Region Segmentation



Layer Representation



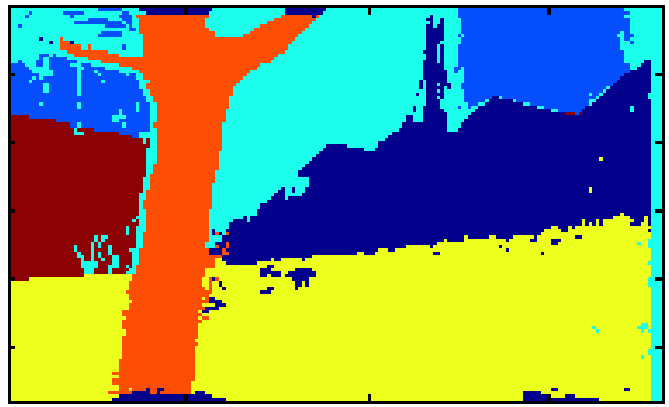


Segmentation

- Find set of regions $R_1, R_2, \ldots, R_n$ such that

$$\bigcup_{i=1}^{n} R_i = I \qquad \forall i \neq j, R_i \cap R_j = \varnothing$$

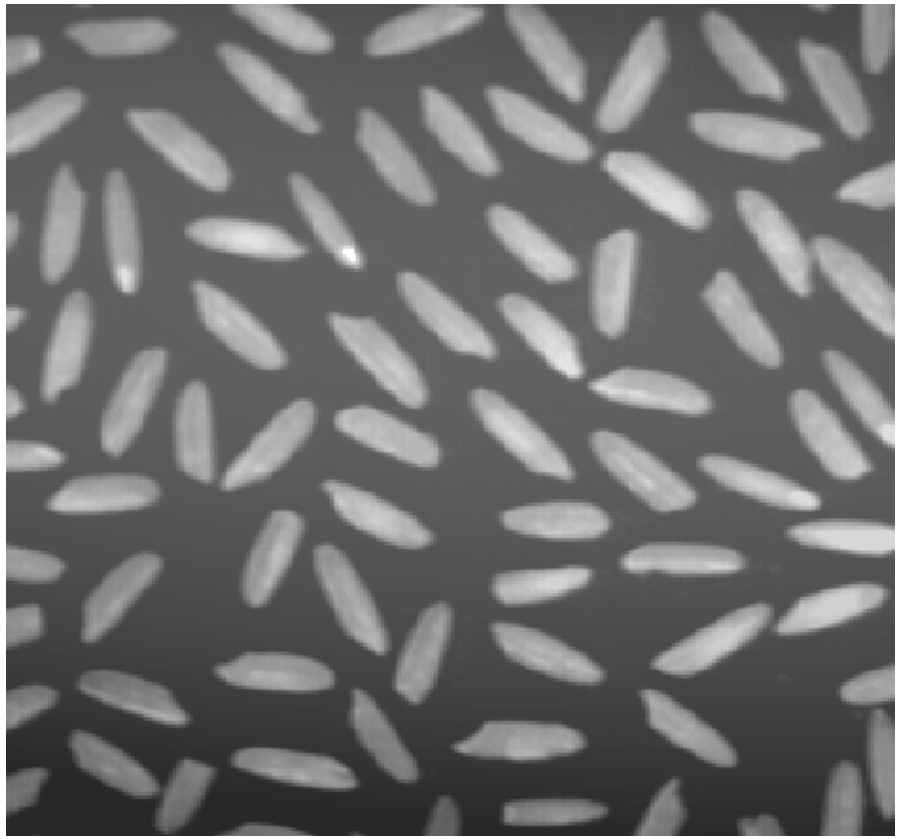
- All pixels in region i satisfy some similarity constraint



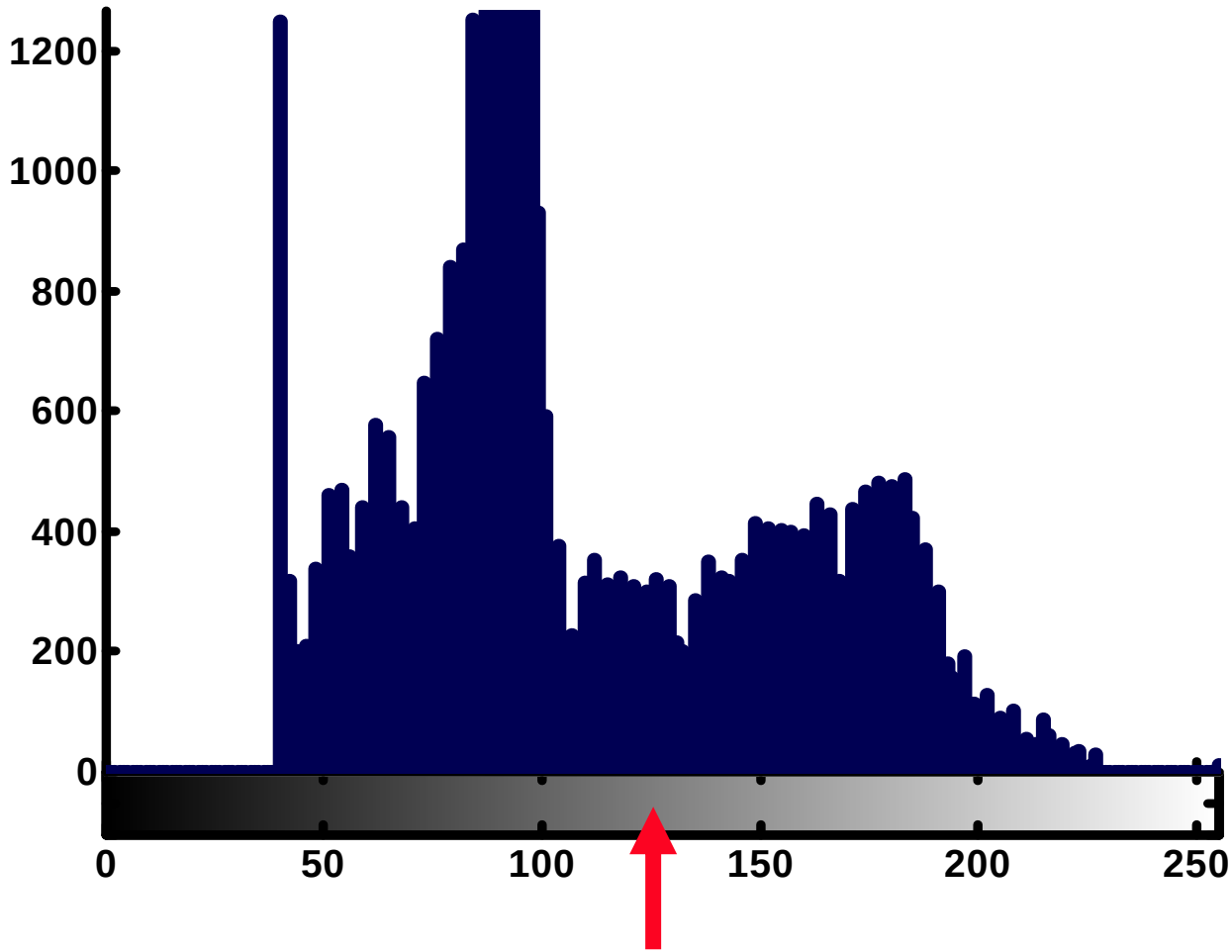
Similarity Constraints

- All pixels in any sub-image must have the **same** gray levels.
- All pixels in any sub-image **must not differ** more than some threshold
- All pixels in any sub-image **may not differ** more than some threshold from the mean of the gray of the region
- The **standard deviation** of gray levels in any sub-image must be small.

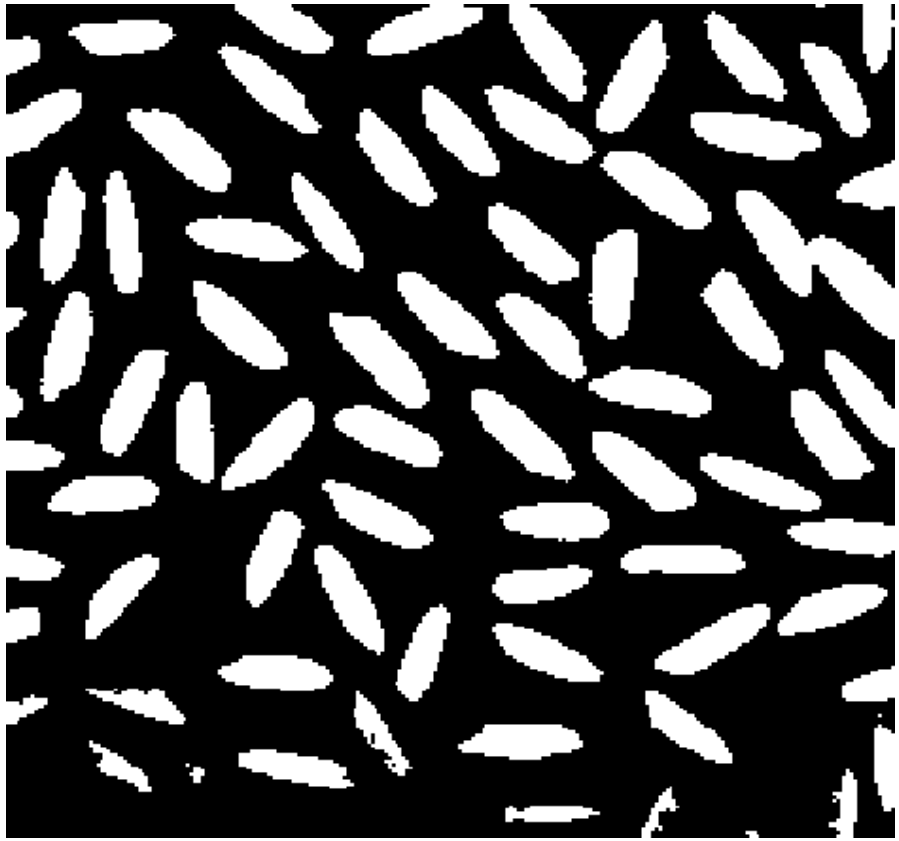
Image Segmentation: Thresholding



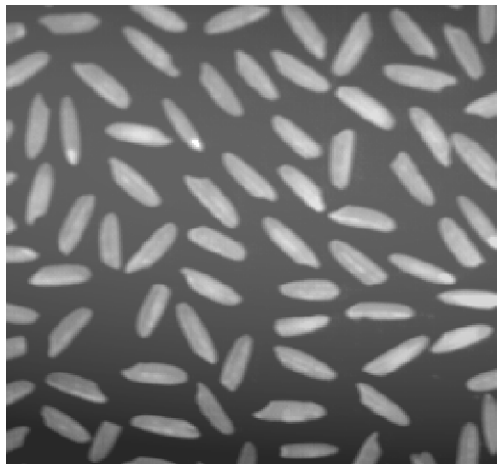
Histogram

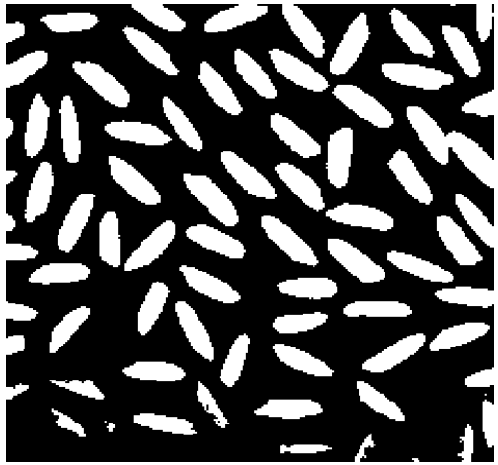


Thresholding



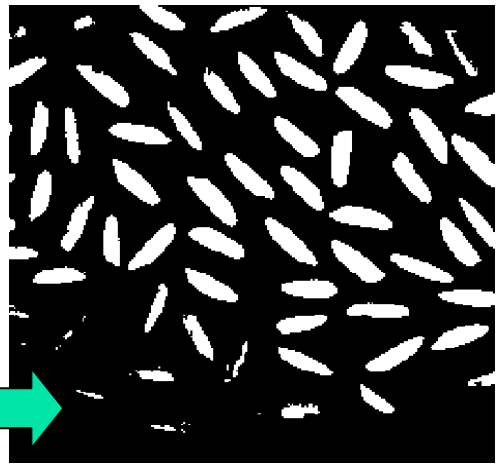
Thresholding

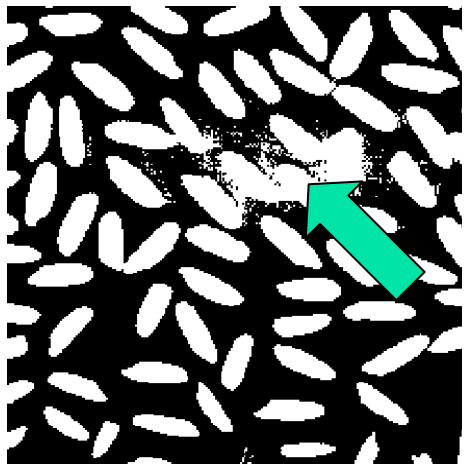




125

99

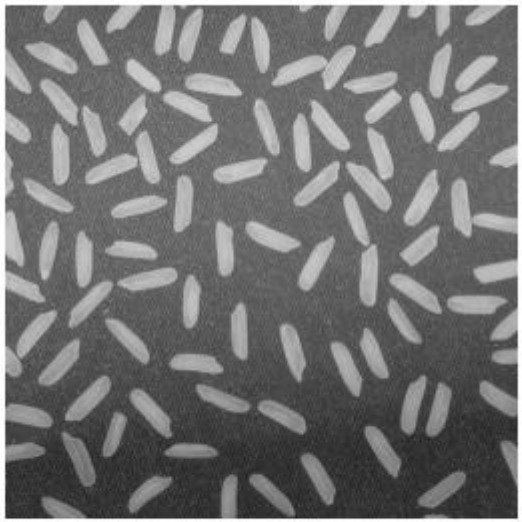


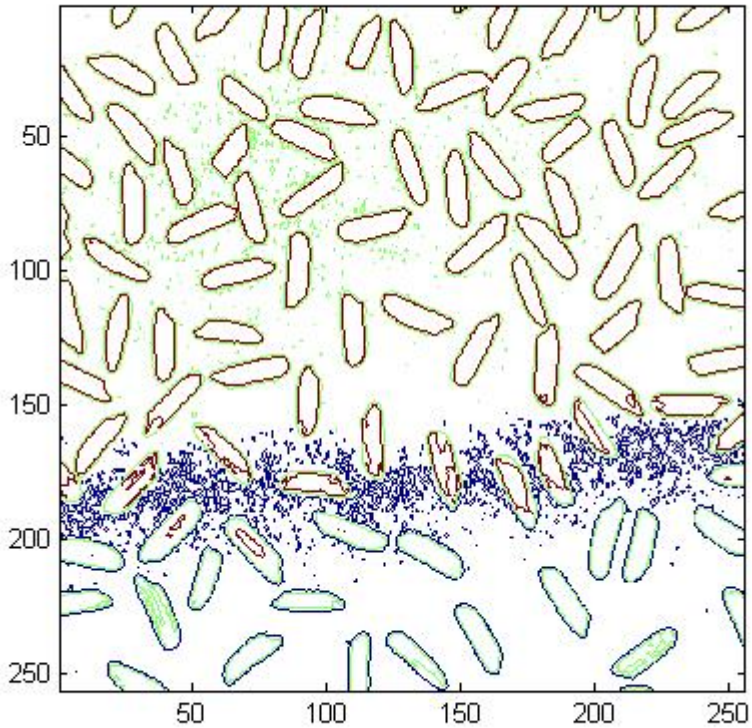


156

Matlab Demo 0: Contours

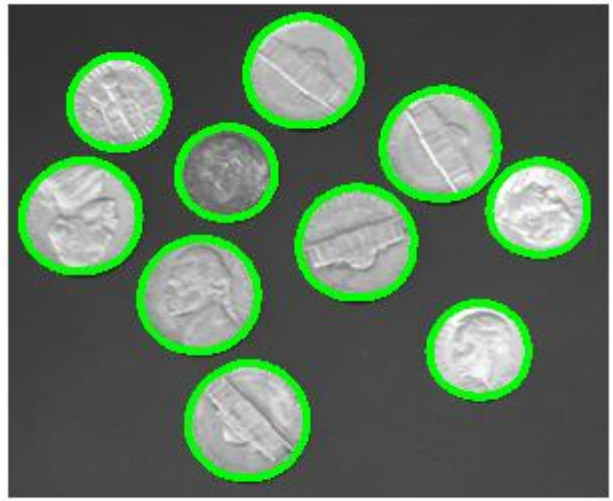
- I = imread('rice.png');
- imshow(I);
- figure, imcontour(I,3);





Matlab Demo 1- boundary

```
I = imread('coins.png');
imshow(I);
BW = im2bw(I);
figure;imshow(BW);
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)));
boundary = bwtraceboundary(BW,[row, col],'N');
figure;imshow(I);
hold on;
plot(boundary(:,2),boundary(:,1),'g','LineWidth',3);
BW_filled = imfill(BW,'holes');% fills holes in the binary image BW
boundaries = bwboundaries(BW_filled); %compute all boundaries
figure;imshow(I);
hold on;
for k=1:10
   b = boundaries{k};
   plot(b(:,2),b(:,1),'g','LineWidth',3);
end
```



Simple Segmentation

$$B(x,y) = \begin{cases} 1 & \text{if } I(x,y) < T \\ 0 & \text{Otherwise} \end{cases}$$

$$B(x,y) = \begin{cases} 1 & \text{if } T_1 < I(x,y) < T_2 \\ 0 & \text{Otherwise} \end{cases}$$

$$B(x,y) = \begin{cases} 1 & \text{if } I(x,y) \in Z \\ 0 & \text{Otherwise} \end{cases}$$

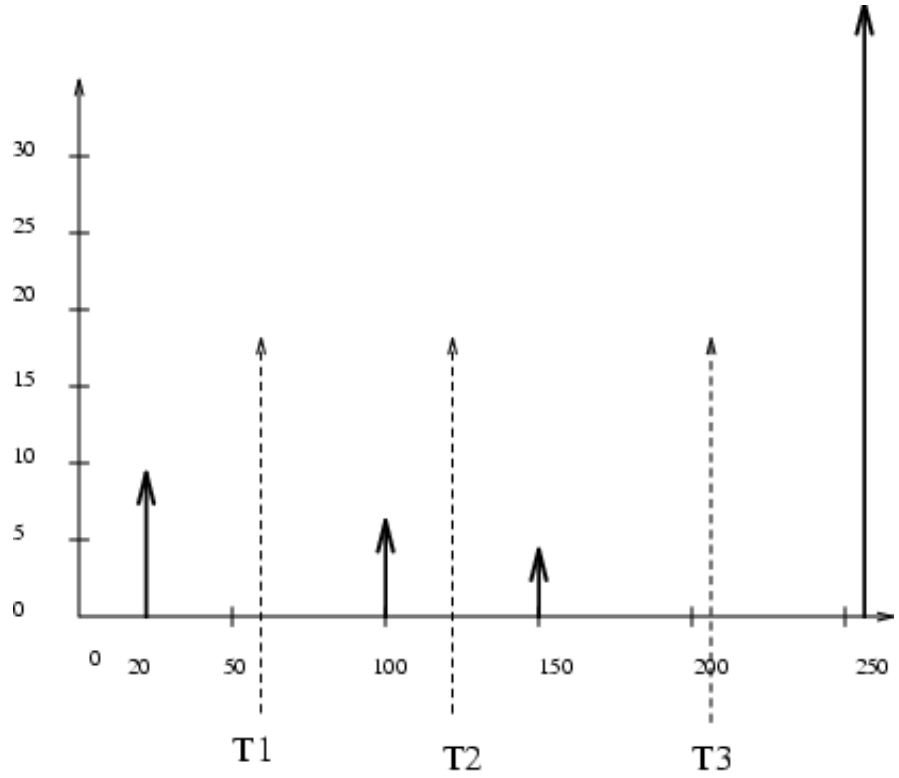
Image Histogram

- Histogram graphs the number of pixels with a particular gray level as a function of the image of gray levels.

255	255	255	255	255	255
255	255	255	255	255	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	100	100	100	255
255	255	255	255	255	255

Segmentation Using Histogram Simple Case

255	255	255	255	255	255	255	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	100	100	255	20	20
255	255	255	255	255	255	20	20
255	255	255	255	255	255	255	255
150	150	255	255	255	255	255	255
150	150	255	255	255	255	255	255



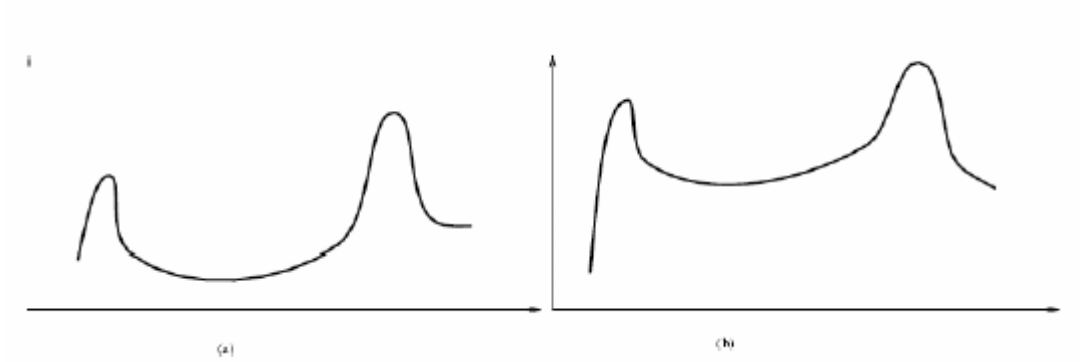
Segmentation Using Histogram Simple Case

$$B_1(x,y) = \begin{cases} 1 & \text{if } 0 < f(x,y) < T_1 \\ 0 & \text{Otherwise} \end{cases}$$

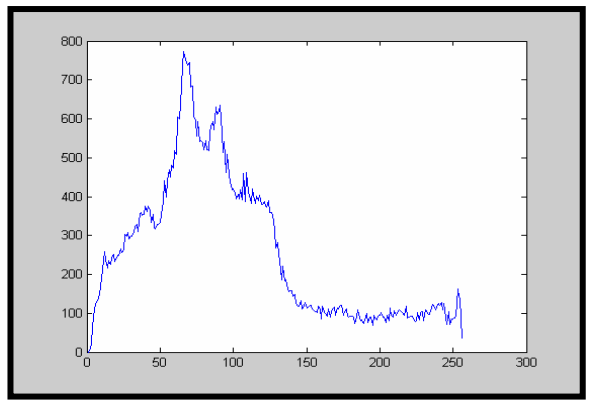
$$B_2(x,y) = \begin{cases} 1 & \text{if } T_1 < f(x,y) < T_2 \\ 0 & \text{Otherwise} \end{cases}$$

$$B_3(x,y) = \begin{cases} 1 & \text{if } T_2 < f(x,y) < T_3 \\ 0 & \text{Otherwise} \end{cases}$$

Realistic Histograms



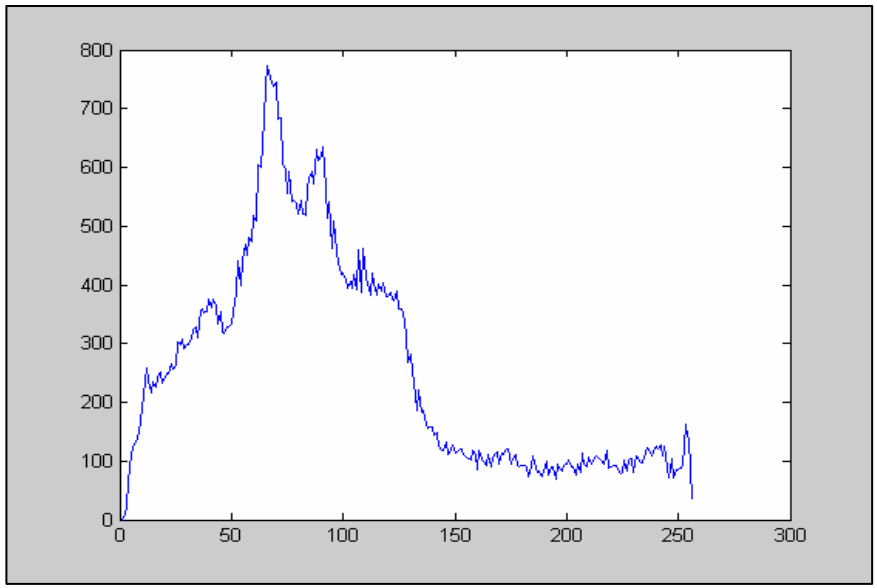
Not realistic

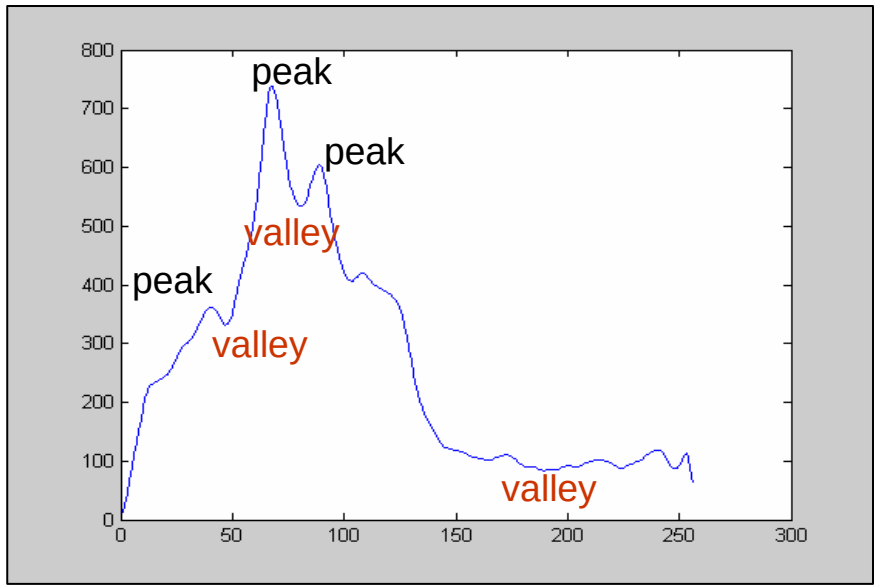


Real (noise)

Realistic Histograms

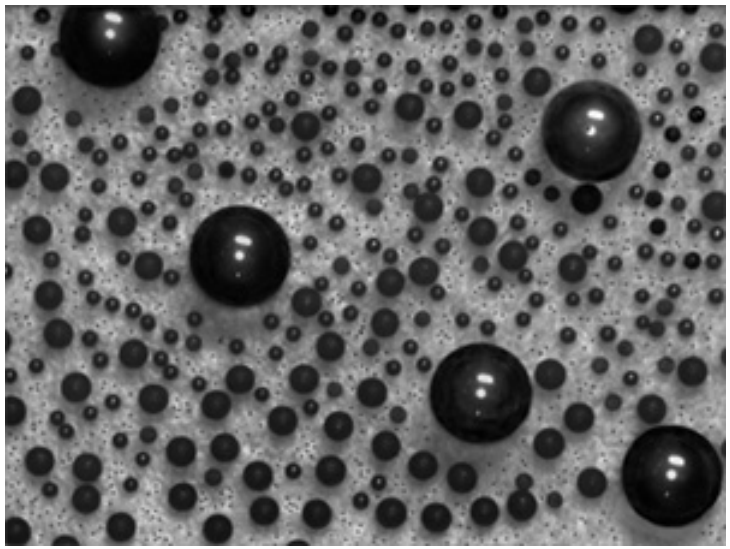
- Smooth out noise in the histogram
  - Convolve by averaging or 1D Gaussian filter

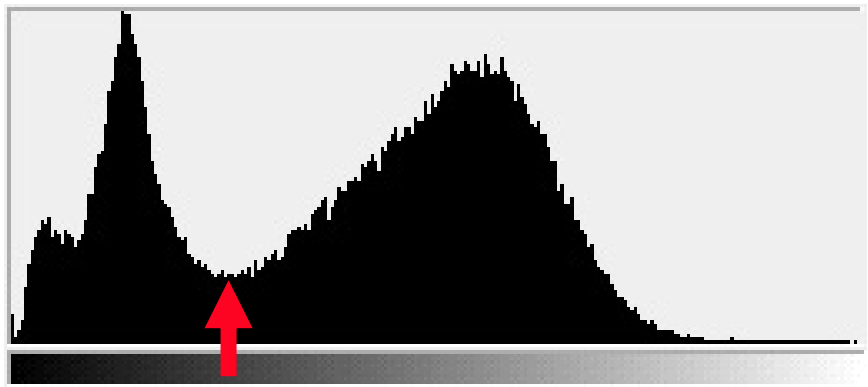




Histogram-based segmentation

- Goal: Break the image into K regions (segments)
- Solution: Reduce the number of colors to K and mapping each pixel to the closest color
  - Histogram-based threshold is a convenient scheme

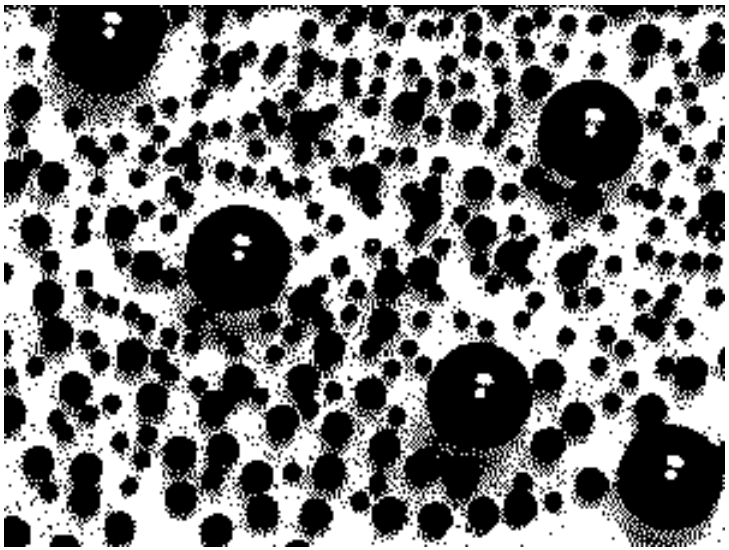


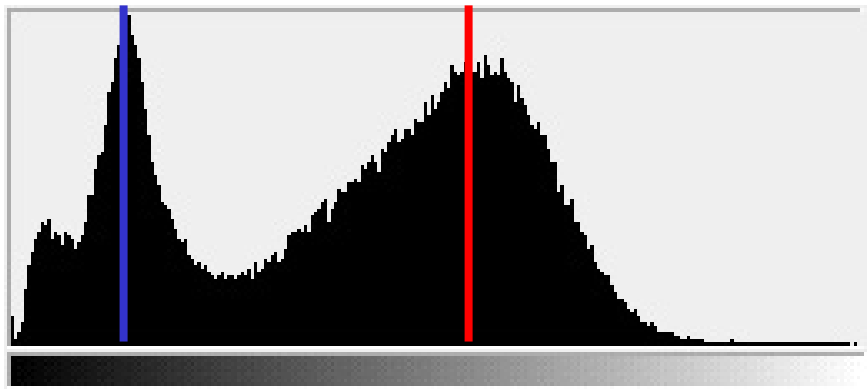


Cut here

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
    - photoshop demo



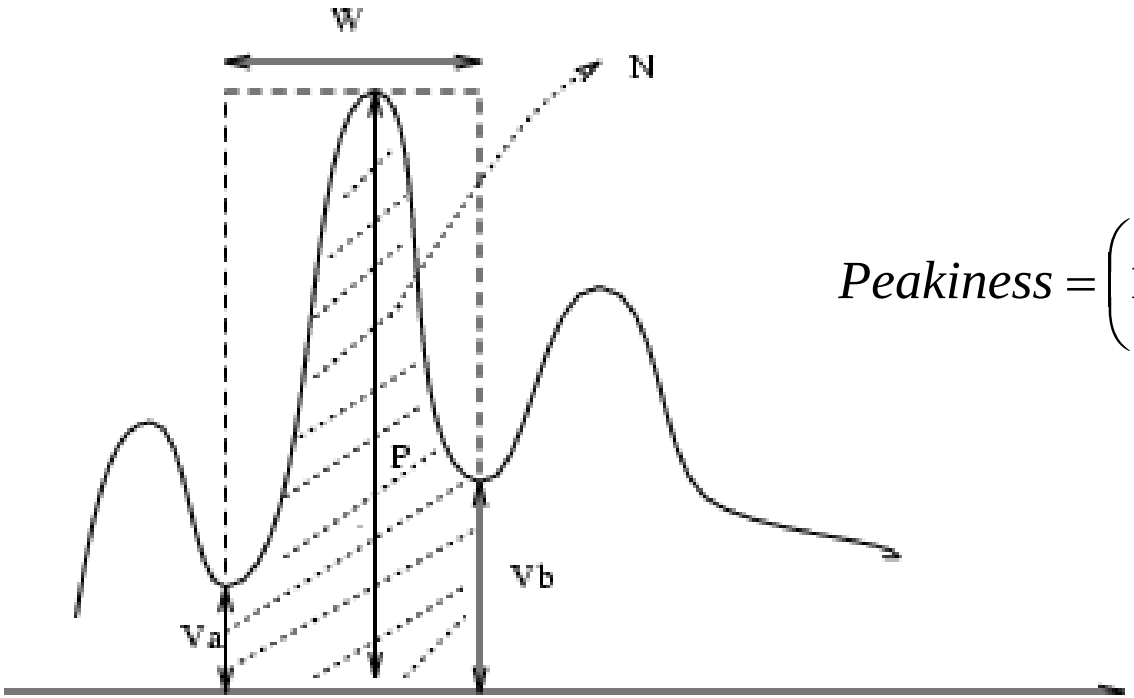


- Here's what it looks like if we use two colors

Segmentation Using Histogram Real image histograms

1. Compute the histogram of a given image.
2. Smooth the histogram by averaging peaks and valleys in the histogram.
3. **Detect good peaks by applying thresholds at the valleys.**
4. Segment the image into several binary images using thresholds at the valleys.
5. **Apply connected component algorithm to each binary image find connected regions.**

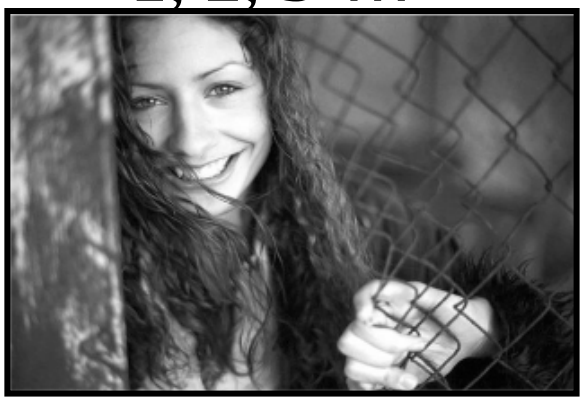
Good Peaks
Peakiness Test

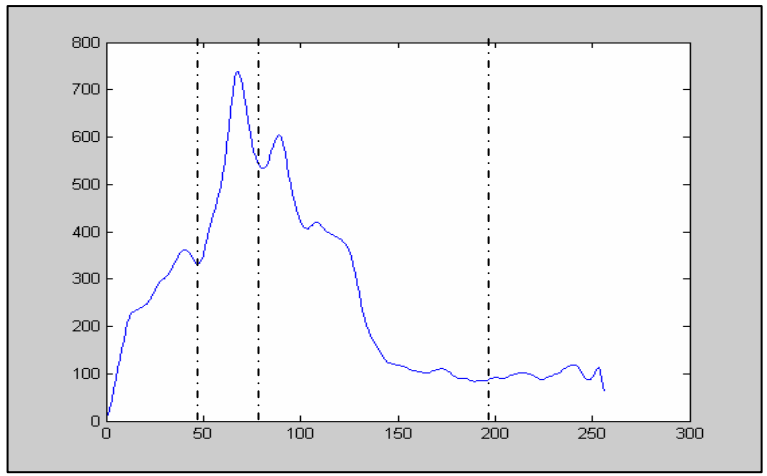


$$Peakiness = \left(1 - \frac{(V_a + V_b)}{2P}\right).\left(1 - \frac{N}{(W.P)}\right)$$

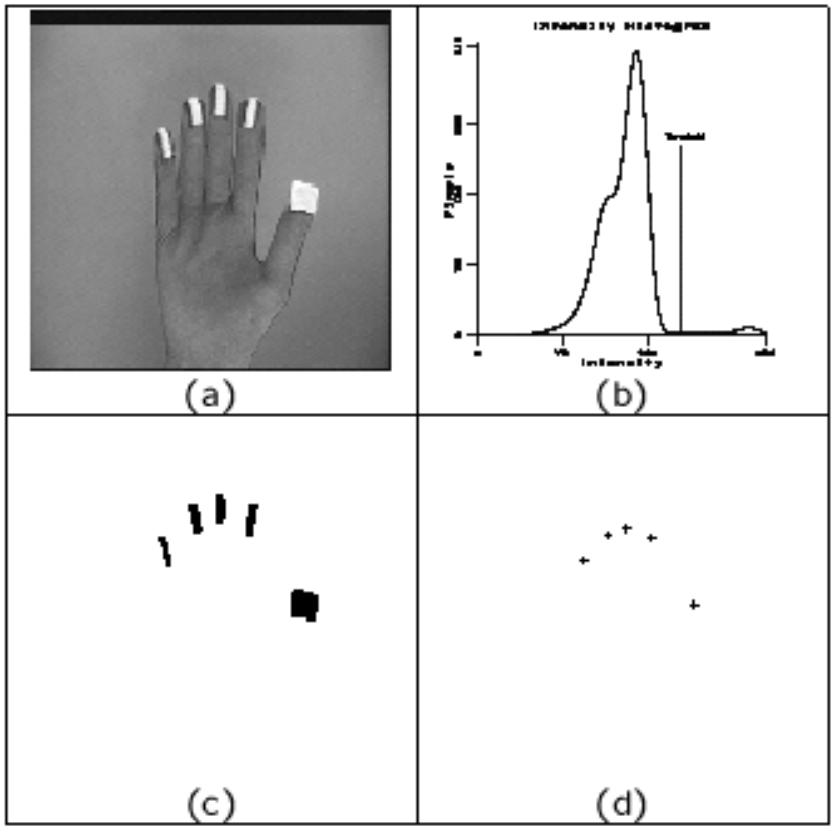
Segmentation Using Histograms

- Select the valleys as thresholds
  - Apply threshold to histogram
  - Label the pixels within the range of a threshold with same label, i.e., a, b, c … or $1, 2, 3$ …

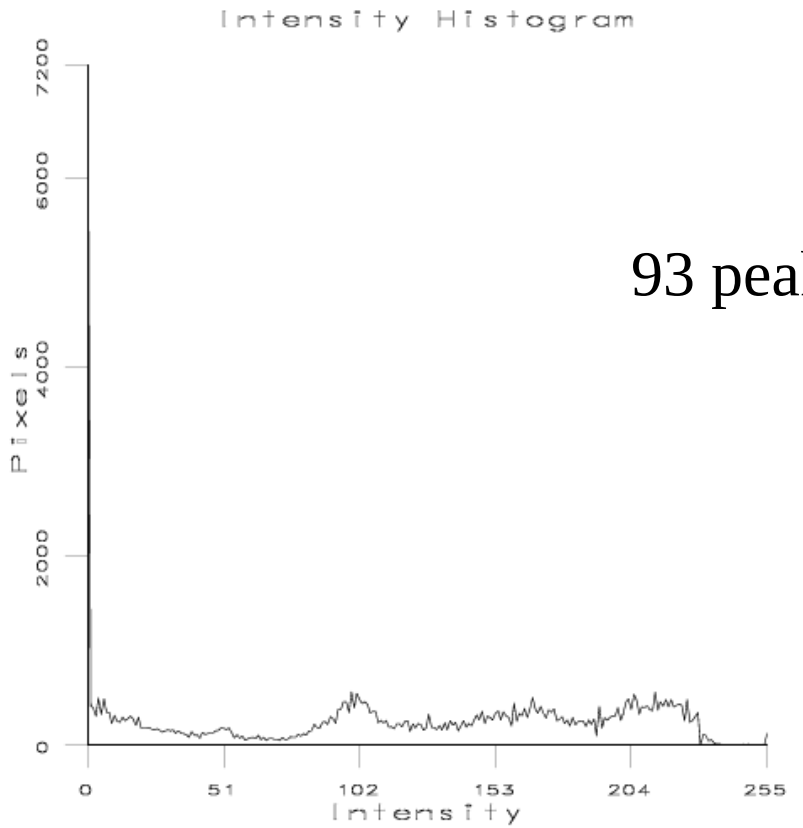




Example:
Detecting Finger Tips (marked white)

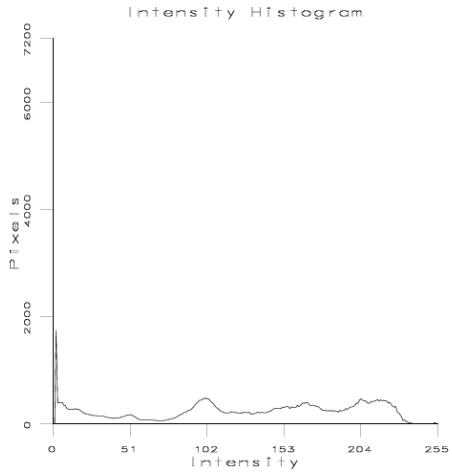


Example
Segmenting a bottle image

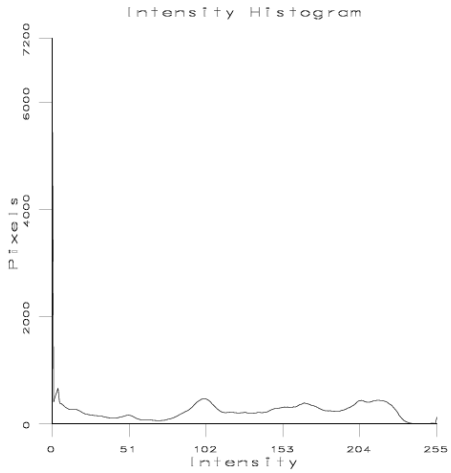


93 peaks

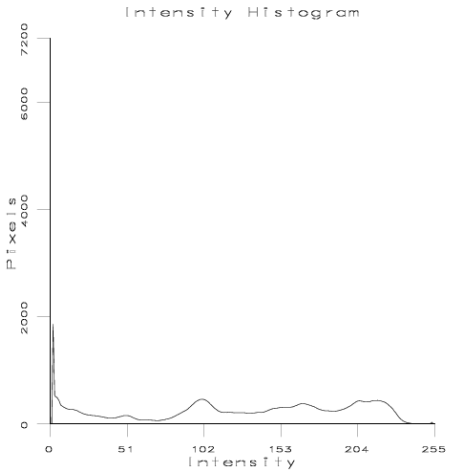
Example
Segmenting a bottle image



Smoothed histogram
(averaging using mask
Of size 5)
54 peaks (once)
After peakiness 18

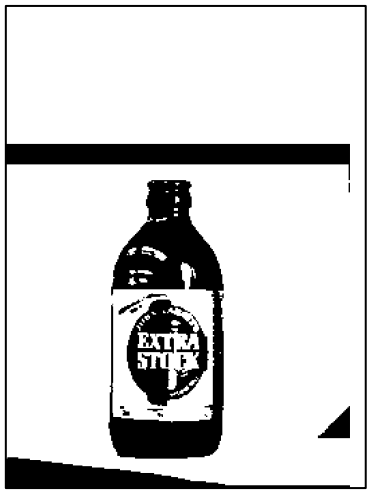


Smoothed histogram
21 peaks (twice)
After peakiness 7

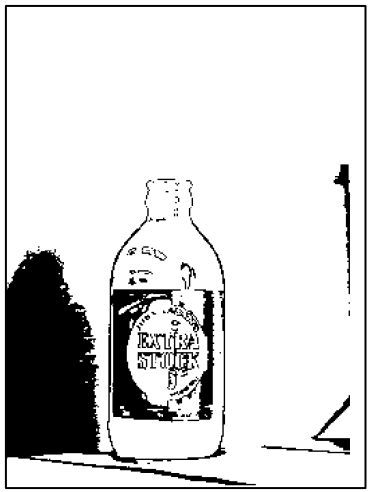


Smoothed histogram
11 peaks (three times)
After peakiness 4

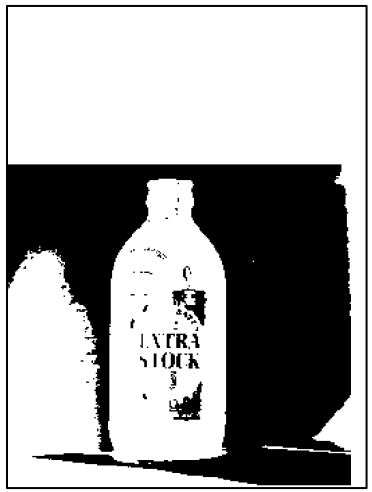
Example
Segmenting a bottle image



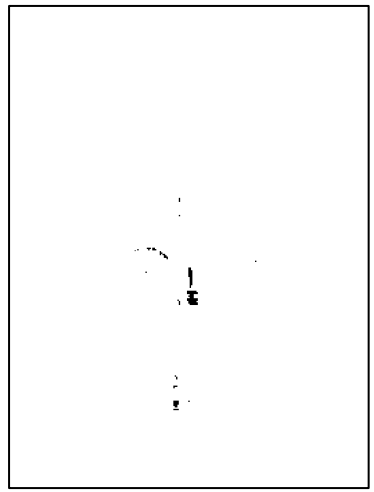
(0,40)



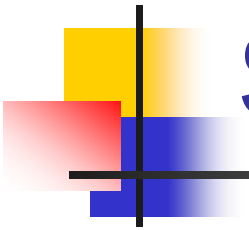
(40, 116)



(116,243)



(243,255)



Steps in Seed Segmentation

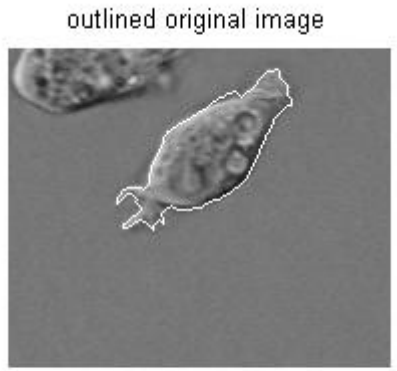
1. Compute the histogram.
2. Smooth the histogram
3. Detect good peaks
4. Segment image into binary images using thresholds at the valleys.
5. Apply connected component algorithm.

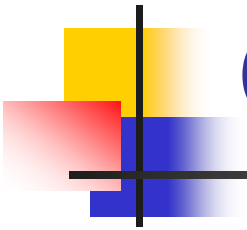
Difference Between Segmentation and Edge Detection

- Closed boundary
  - Edges are usually open
  - Segmentation provides closed boundaries
- Local or global
  - Edges are computed in the locality
  - Segmentation is global
- Increasing feature vector dimensionality
  - Does not drastically improve edge detection
  - Improves segmentation (motion, texture information etc.)

Matlab Demo 2- contrast

```
I = imread('cell.tif');
figure, imshow(I), title('original image');
[junk threshold] = edge(I, 'sobel'); %use Sobel operator to calculate the threshold value
fudgeFactor = .5;
BWs = edge(I,'sobel', threshold * fudgeFactor); %use edge again to obtain the binary mask
figure, imshow(BWs), title('binary gradient mask');
se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);
BWsdil = imdilate(BWs, [se90 se0]);       %dilate to remove gaps
figure, imshow(BWsdil), title('dilated gradient mask');
BWdfill = imfill(BWsdil, 'holes');        %hole filling
figure, imshow(BWdfill);title('binary image with filled holes');
BWnobord = imclearborder(BWdfill, 4);   %remove object on border
figure, imshow(BWnobord), title('cleared border image');
seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);           %smoothen the object by repeated eroding
figure, imshow(BWfinal), title('segmented image');
BWoutline = bwperim(BWfinal);             %place an outline
Segout = I;
Segout(BWoutline) = 255;
figure, imshow(Segout), title('outlined original image');
```

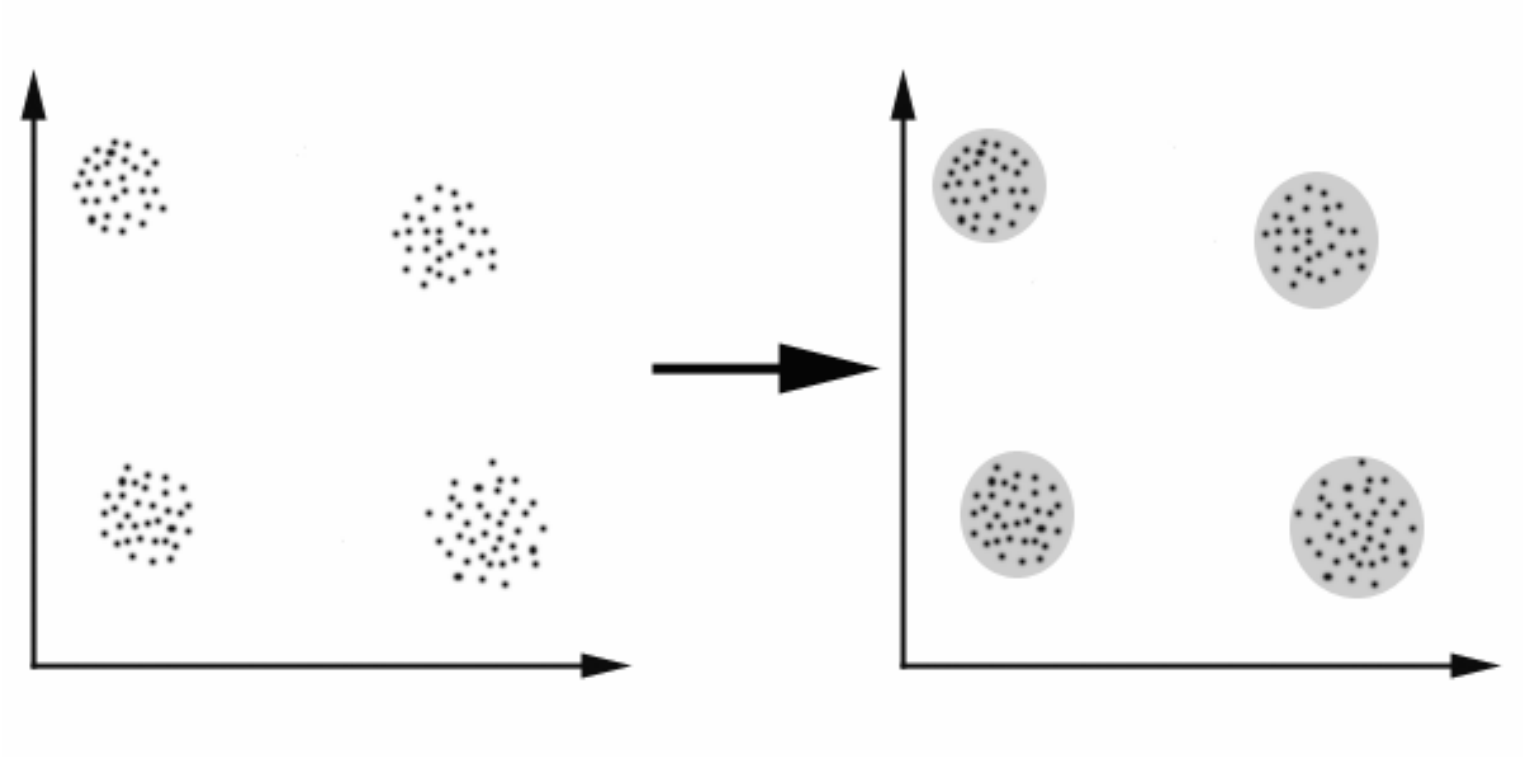




Outline

- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph Cuts

Clustering Principle

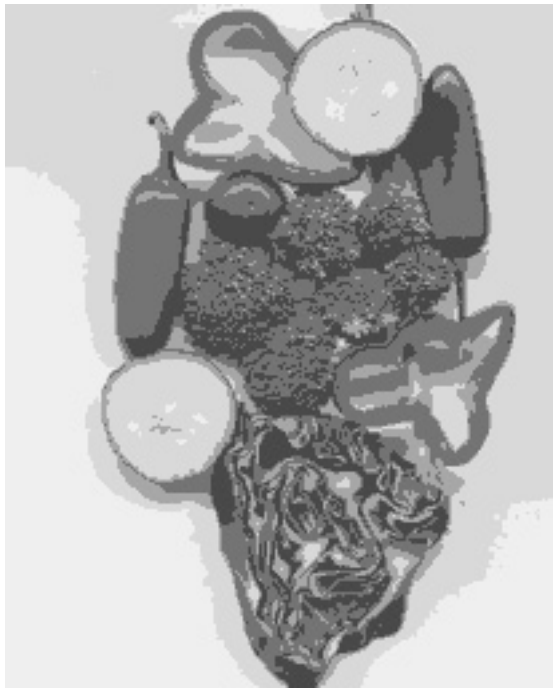


Segmentation as clustering

- Cluster together (pixels, tokens, etc.) that belong together
- Agglomerative (凝聚) clustering (bottom-up)
  - attach closest to cluster it is closest to
  - repeat
- Divisive (分裂) clustering (top-down)
  - split cluster along best boundary
  - Repeat
- Point-Cluster distance (merge/split rules)
  - single-link clustering
  - complete-link clustering
  - group-average clustering
- Dendrograms (树形图)
  - yield a picture of output as clustering process continues

Example

Image



Clusters on intensity

Clusters on color

Simple clustering algorithms

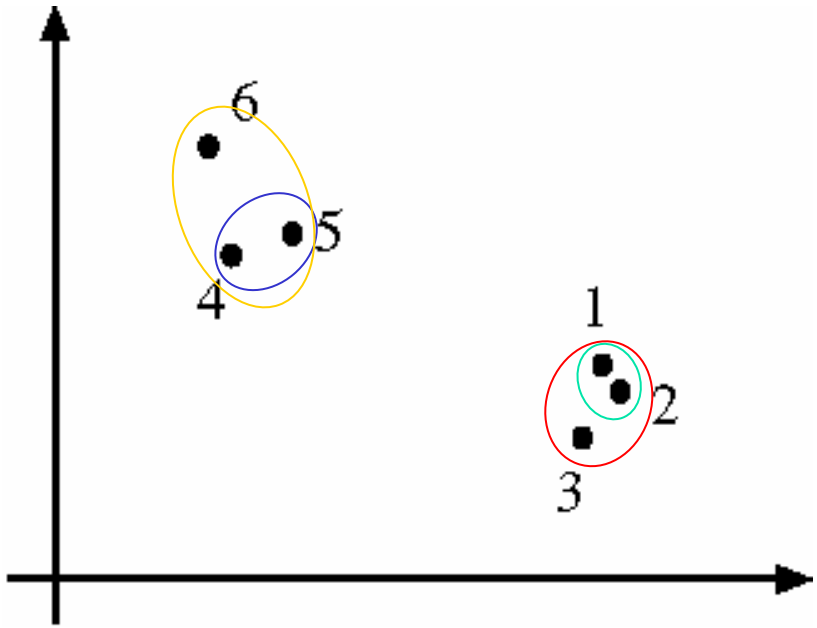
Algorithm 15.3: Agglomerative clustering, or clustering by merging

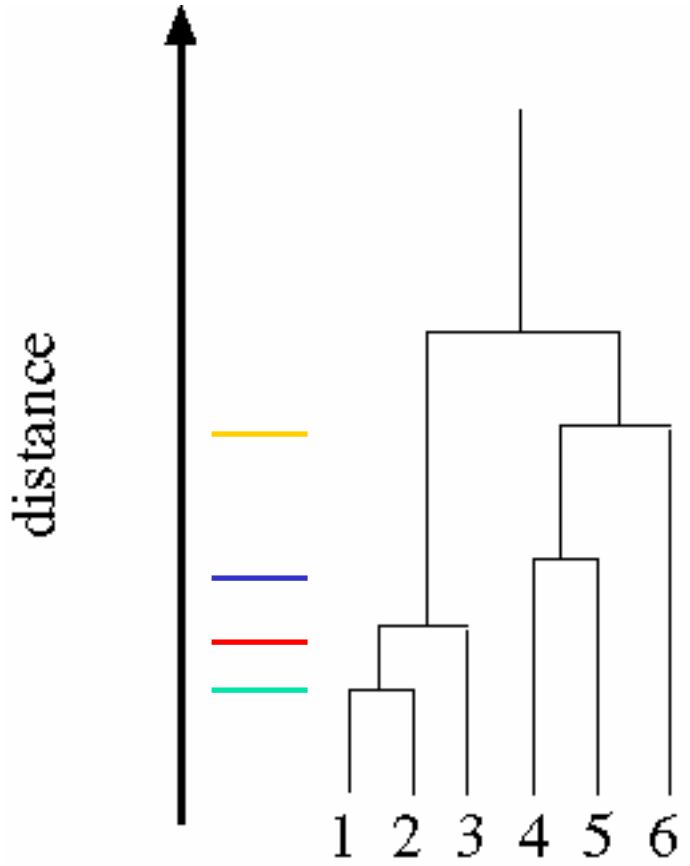
```
Make each point a separate cluster
Until the clustering is satisfactory
     Merge the two clusters with the
       smallest inter-cluster distance
end
```

Algorithm 15.4: Divisive clustering, or clustering by splitting

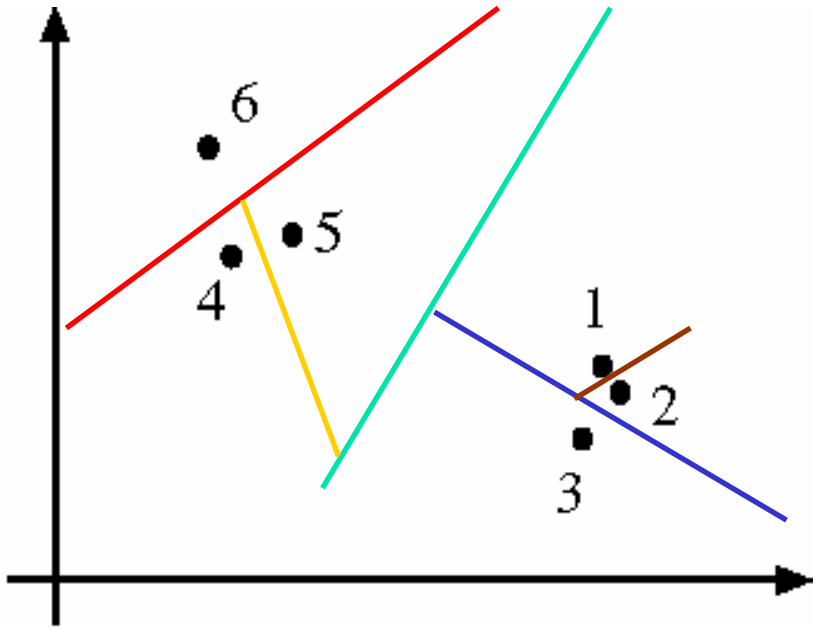
```
Construct a single cluster containing all points
Until the clustering is satisfactory
     Split the cluster that yields the two
       components with the largest inter-cluster distance
end
```

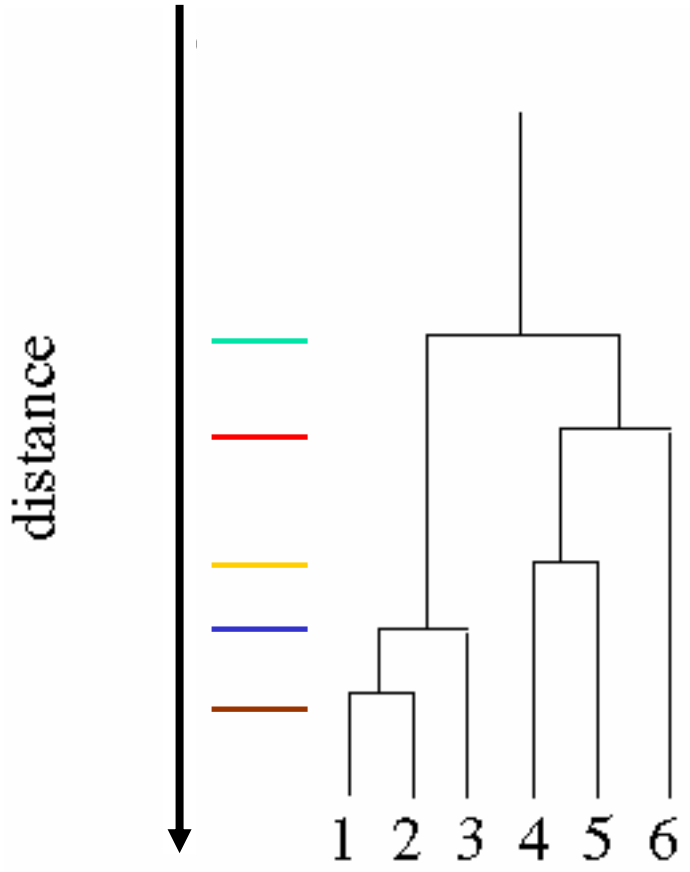
Agglomerative clustering—Clustering by merging





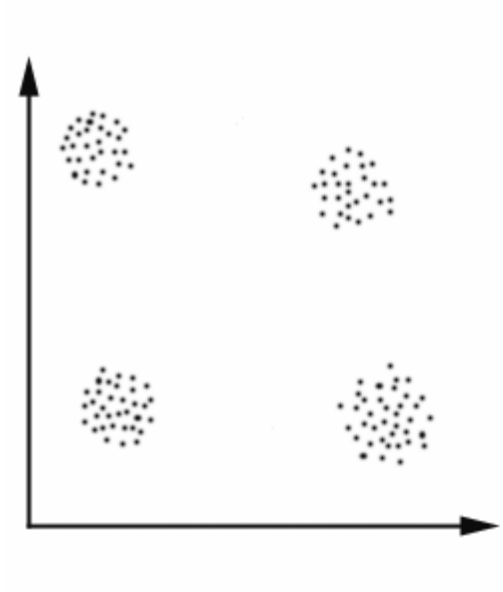
Divisive clustering—Clustering by splitting





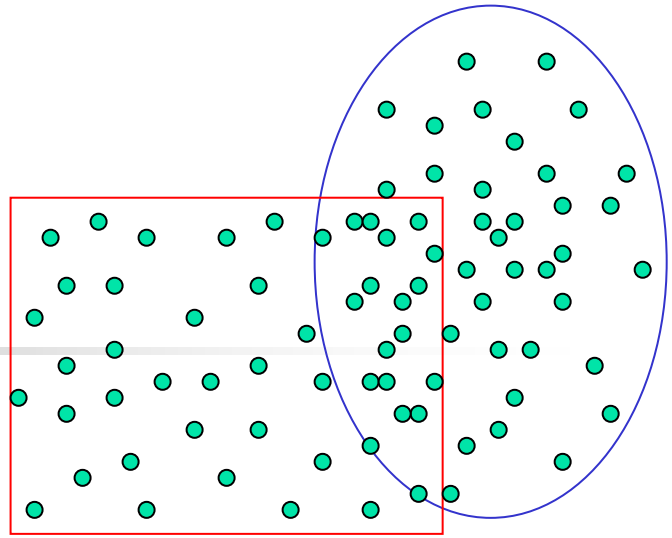
Clustering

- We want to group together some primitives



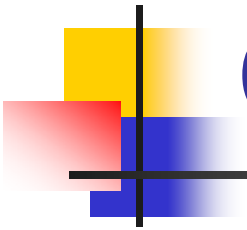
Seems easy, but…

Clustering



$$f_1 = Unit(a,b),\ f_2 = Normal(\mu,\Sigma)$$

- We want to group together some primitives
- If we knew which items belongs to a group…
  - A good description of the groups can be drawn
  - Position, intensity, texture…
- If we knew a good description of the group…
  - We may figure out which primitives belong to which groups
  - Or at least the probability…
- This is a chicken and egg problem…

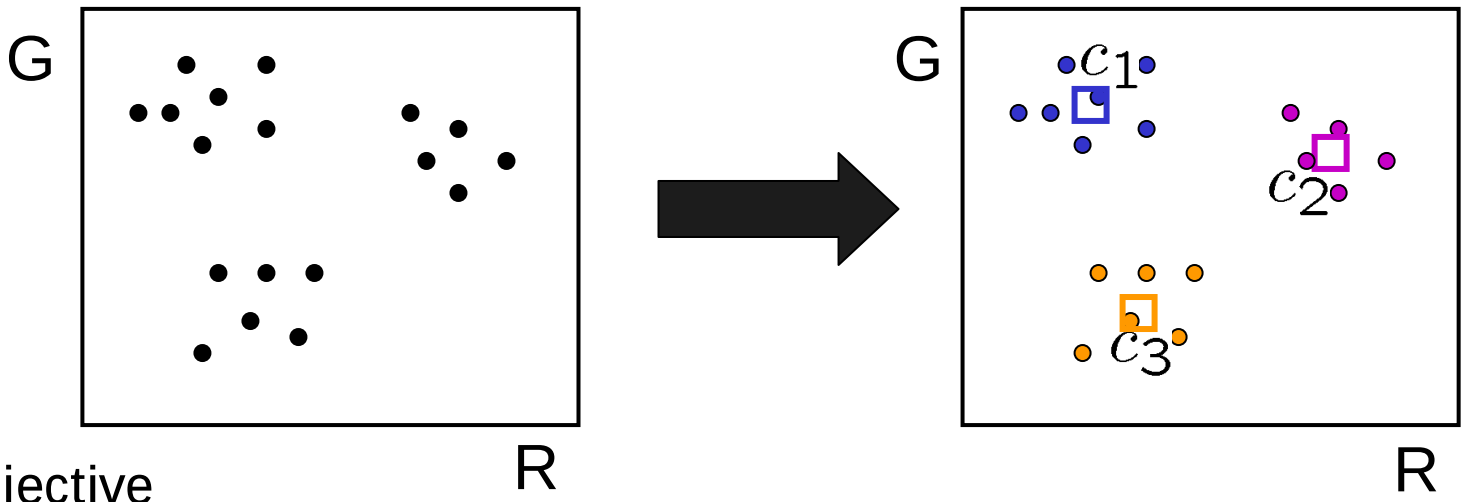


Clustering

- Iterative solution:
  - Guess one side of the answer
  - Figure out the other side
  - Refigure out the first side
  - Keep going till we converge

Clustering

- How to choose the representative colors?
  - This is a clustering problem!

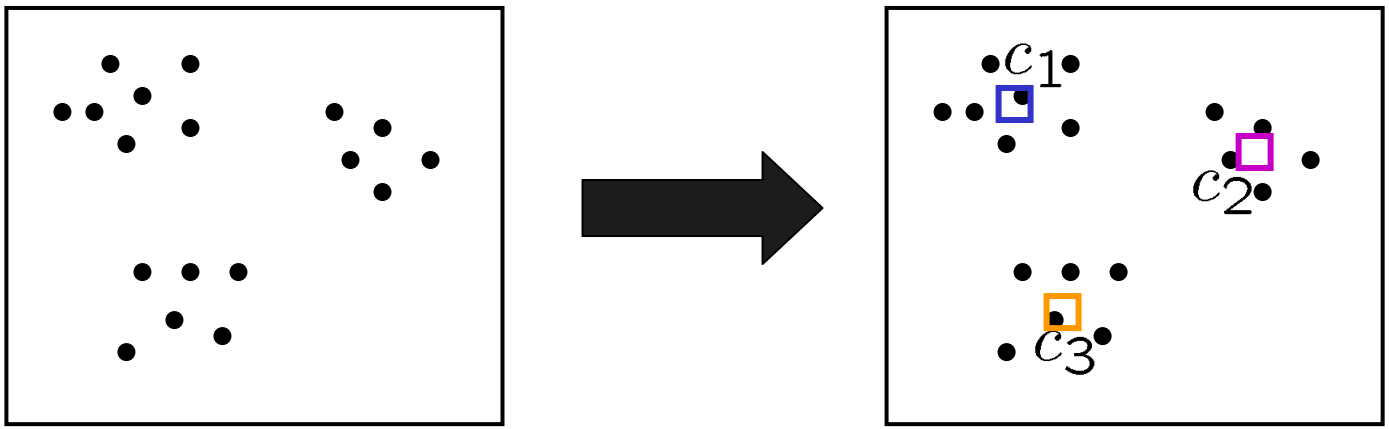


- Objective
  - Each point should be as close as possible to a cluster center
    - Minimize sum squared distance of each point to closest center

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

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

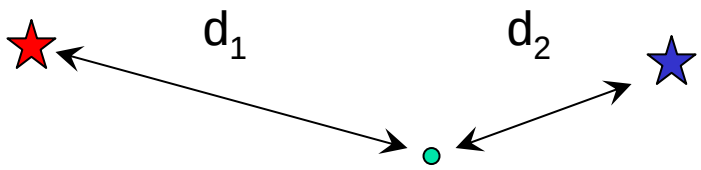
- K-means clustering algorithm
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean 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:

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

Convergence of the algorithm

- The iteration always reduces the error measure
  - *Reassigning a point to the nearest center reduces error*
  - *The center that minimizes MSE is the average*



$d_2 < d_1$

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

Recall – Fitting a constant function

- For constant function *y=a*
  - Minimizing squares gives *a=mean*

$$Min(E = \sum_i (y_i - a)^2)$$

$$\frac{\partial E}{\partial a} = \sum_i -2(y_i - a) = -2(\sum_i y_i - n \cdot a) = 0$$

$$a = \sum_i y_i / n = mean(Y)$$

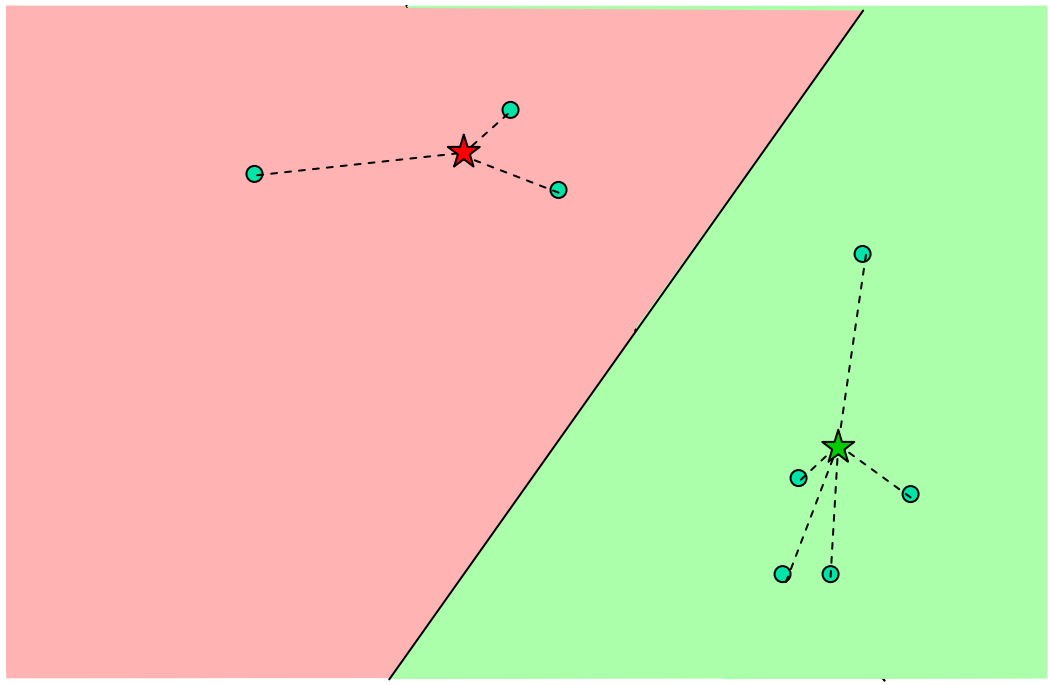
K-Means

- Choose a fixed number of clusters
- Choose cluster centers and point-cluster allocations to minimize error

$$\sum_{i \in \text{clusters}} \left\{ \sum_{j \in \text{elements of i'th cluster}} \left\| x_j - \mu_i \right\|^2 \right\}$$

- can't do this by search, because there are too many possible allocations.
- Algorithm
  - fix cluster centers; allocate points to closest cluster
  - fix allocation; compute best cluster centers
- x could be any set of features for which we can compute a distance (careful about scaling)

K-Means

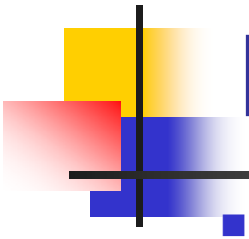


K-Means

```
Choose k data points to act as cluster centers

Until the cluster centers are unchanged

    Allocate each data point to cluster whose center is nearest

    Now ensure that every cluster has at least
    one data point; possible techniques for doing this include .
    supplying empty clusters with a point chosen at random from
    points far from their cluster center.

    Replace the cluster centers with the mean of the elements
    in their clusters.

end
```

Algorithm 16.5: *Clustering by K-Means*

* From Marc Pollefeys COMP 256 2003



K-means: Discussion

- What does the term “close” mean
  - Color, position, texture, shape, motion…
- The convergence of the algorithm
  - Each iteration reduces the error measure.
  - Must converge in a finite number of steps.
- Local minima: Initial guess is crucial
  - Try cluster 2, 6, 12 into two clusters
    - Start from (3,10) -> (4,12)
    - Start from (0,6) -> (2,9)
- How to initialize and how many clusters
  - Try many initializations and pick the best answer
  - Determining the number of clusters is always a problem

Image Segmentation by K-Means

- Select a value of K
- Select a feature vector for every pixel (color, texture, position, or combination of these etc.)
- Define a similarity measure between feature vectors (Usually Euclidean Distance).
- Apply K-Means Algorithm.
- Apply Connected Components Algorithm.
- Merge any components of size less than some threshold to an adjacent component that is most similar to it.

* From Marc Pollefeys COMP 256 2003

Results of K-Means Clustering:

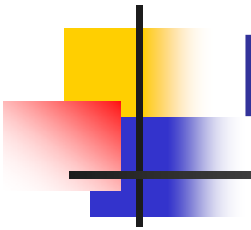
Image

Clusters on intensity

Clusters on color

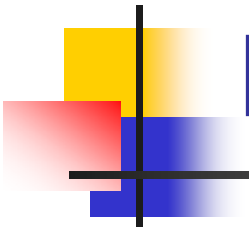
K-means clustering using intensity alone and color alone

* From Marc Pollefeys COMP 256 2003



Expectation-Maximization

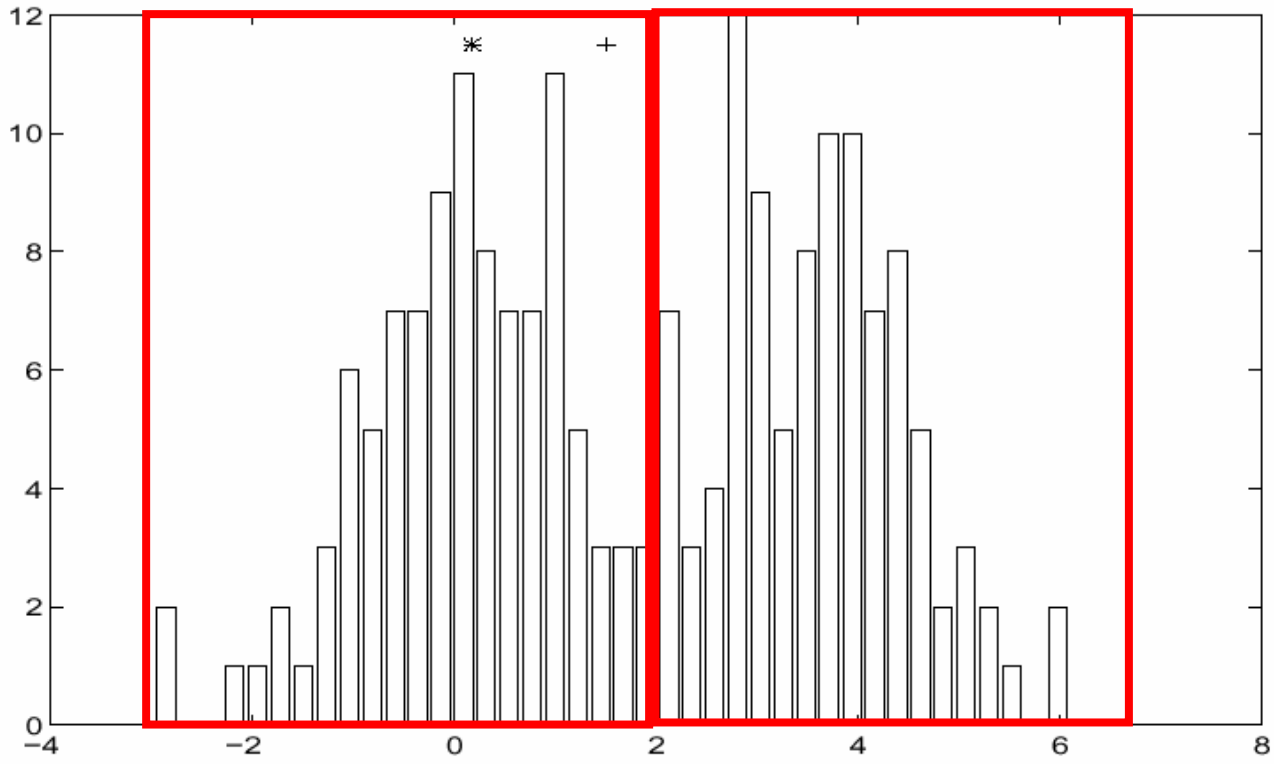
- Can we go farther than K-means?
  - Idea 1: make soft assignments: Expectation-Maximization
    - A point is partially assigned to all clusters
    - Use probabilistic formulation (e.g. weather forcasting)
    - Each cluster is a probability distribution over possible primitives
    - Assign a probability that each primitive belongs to each cluster
  - Idea 2: Take more feasible similarity definition: Mean-shift
    - Mean-shift is intrinsically a E-M algorithm
    - The computation is executed in a kernel space



K-Means and E-M

- K-Means an approximation to EM
  - Model (hypothesis space): Mixture of N Gaussians
  - Latent variables: Correspondence of data and Gaussians
- We notice:
  - Given the mixture model, it's easy to calculate the correspondence
  - Given the correspondence it's easy to estimate the mixture models

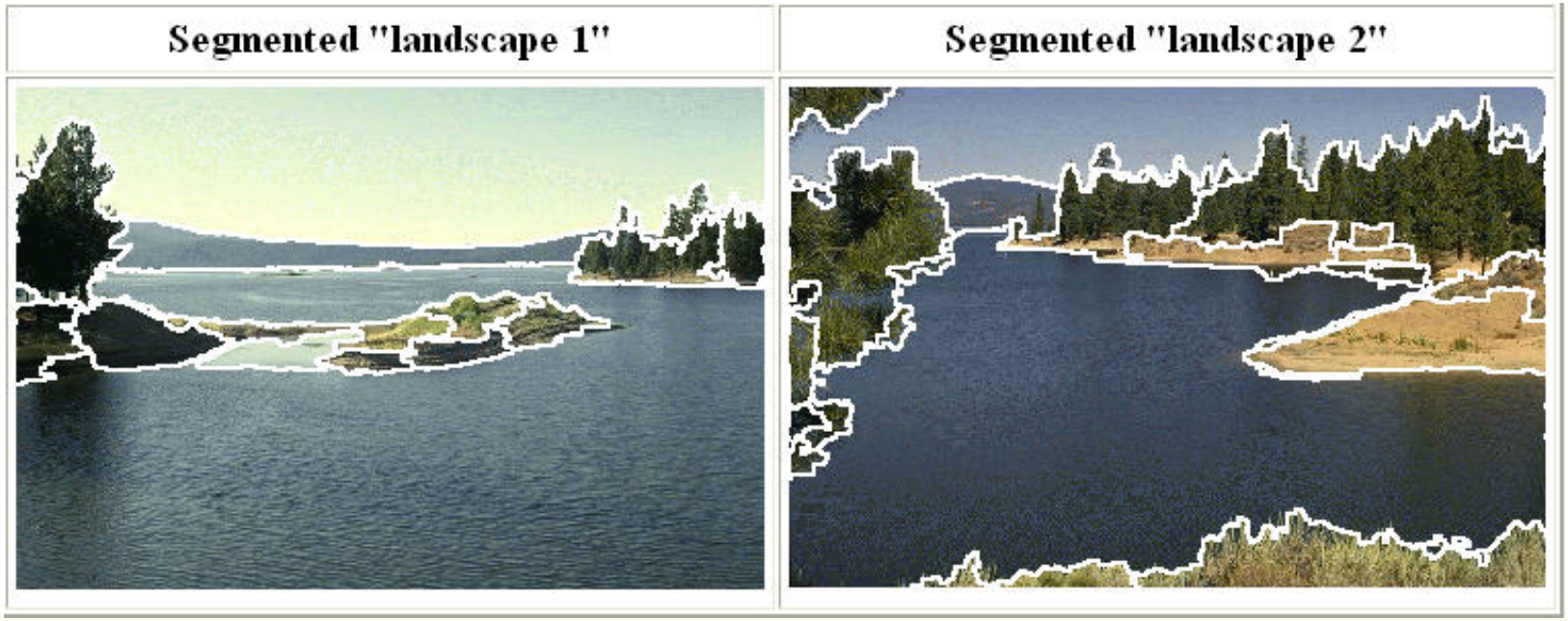
Finding Modes in a Histogram



- How Many Modes Are There?
  - Easy to see, hard to compute

Mean Shift Segmentation

- One of the most popular techniques



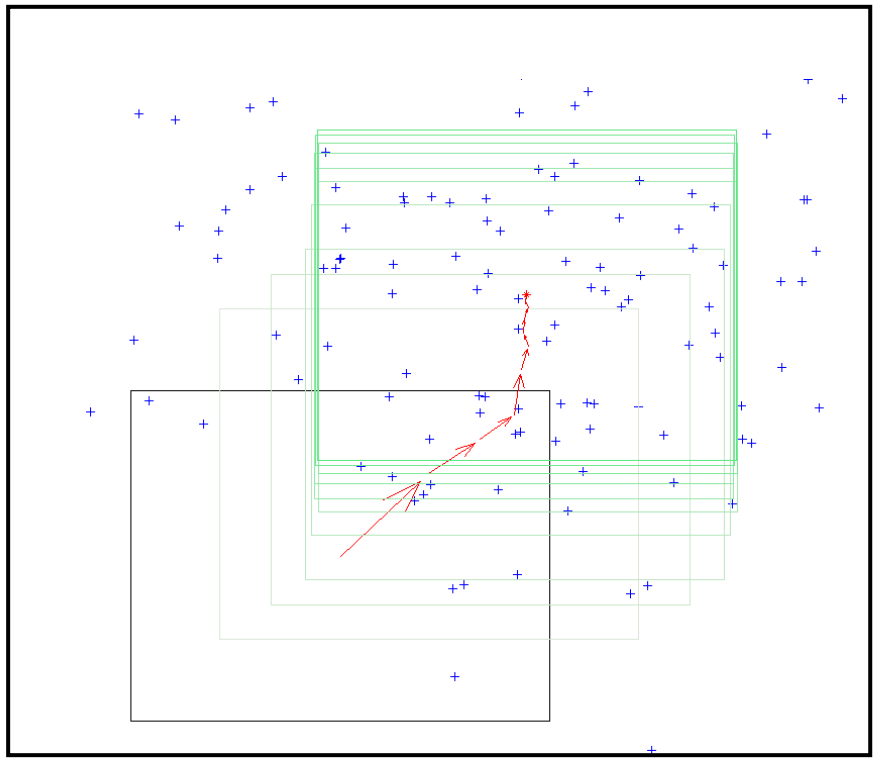
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean Shift Algorithm

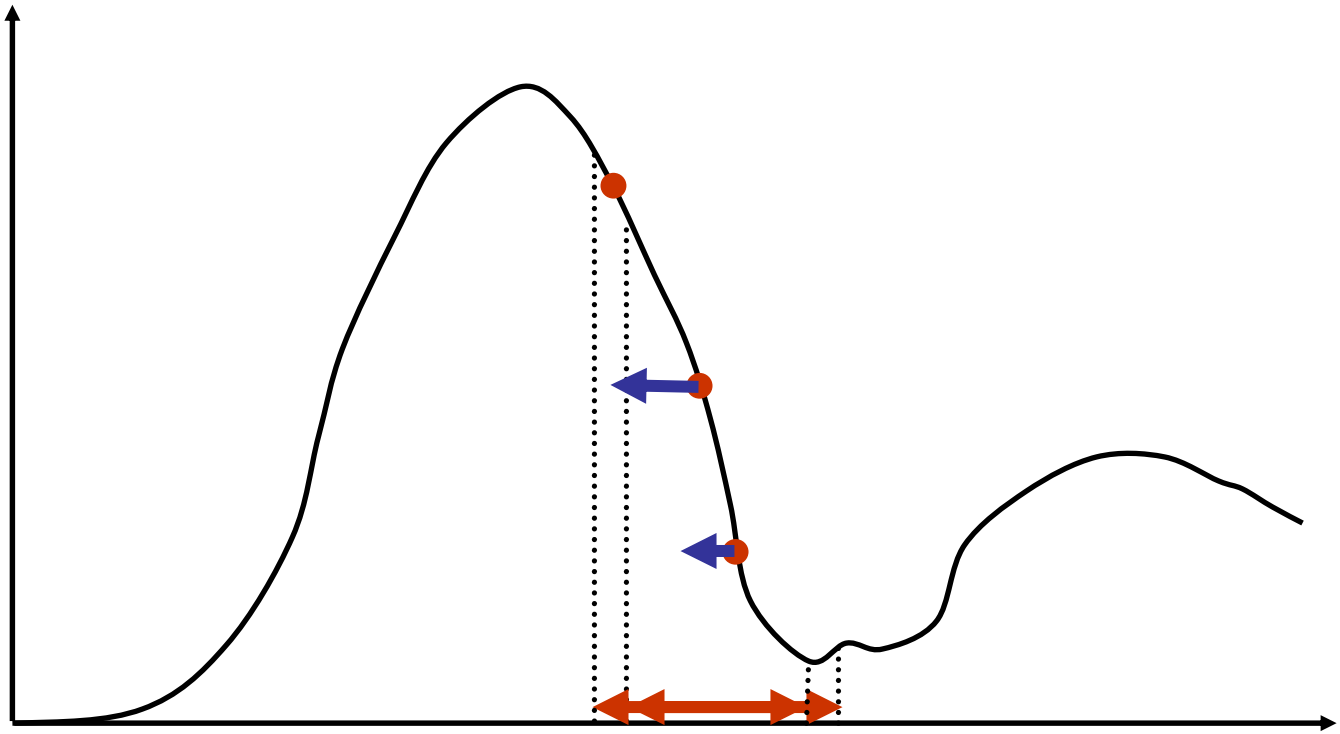
Mean Shift Algorithm

1. Choose a search window size.
2. Choose the initial location of the search window.
3. Compute the mean location (centroid of the data) in the search window.
4. Center the search window at the mean location computed in Step 3.
5. Repeat Steps 3 and 4 until convergence.

The mean shift algorithm seeks the “mode” or point of highest density of a data distribution:

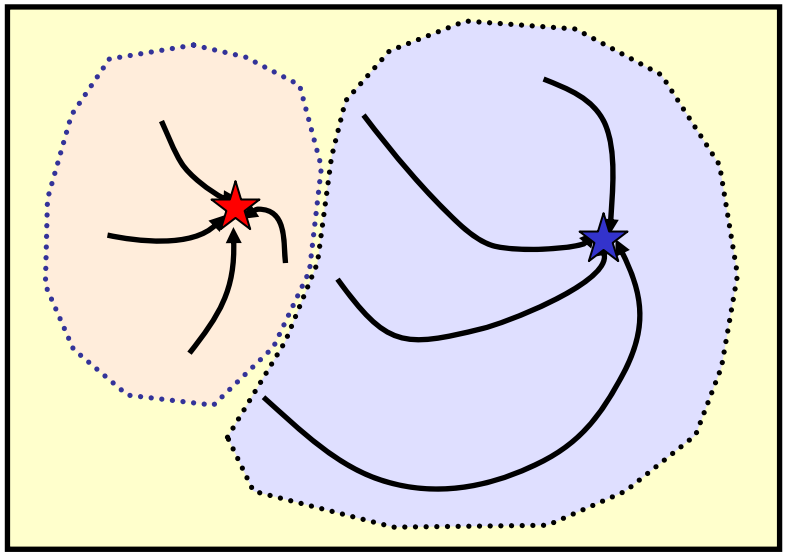


Mean Shift Segmentation

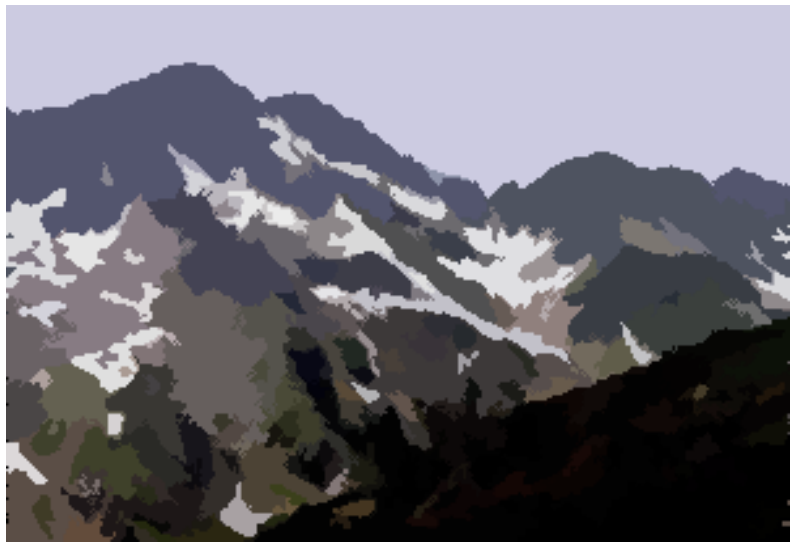


Mean Shift Segmentation

- Given an image, convert it to a function that is inversely related to edgeness
- Perform mean shift from every pixel
- Cluster pixels that lead to the same peak

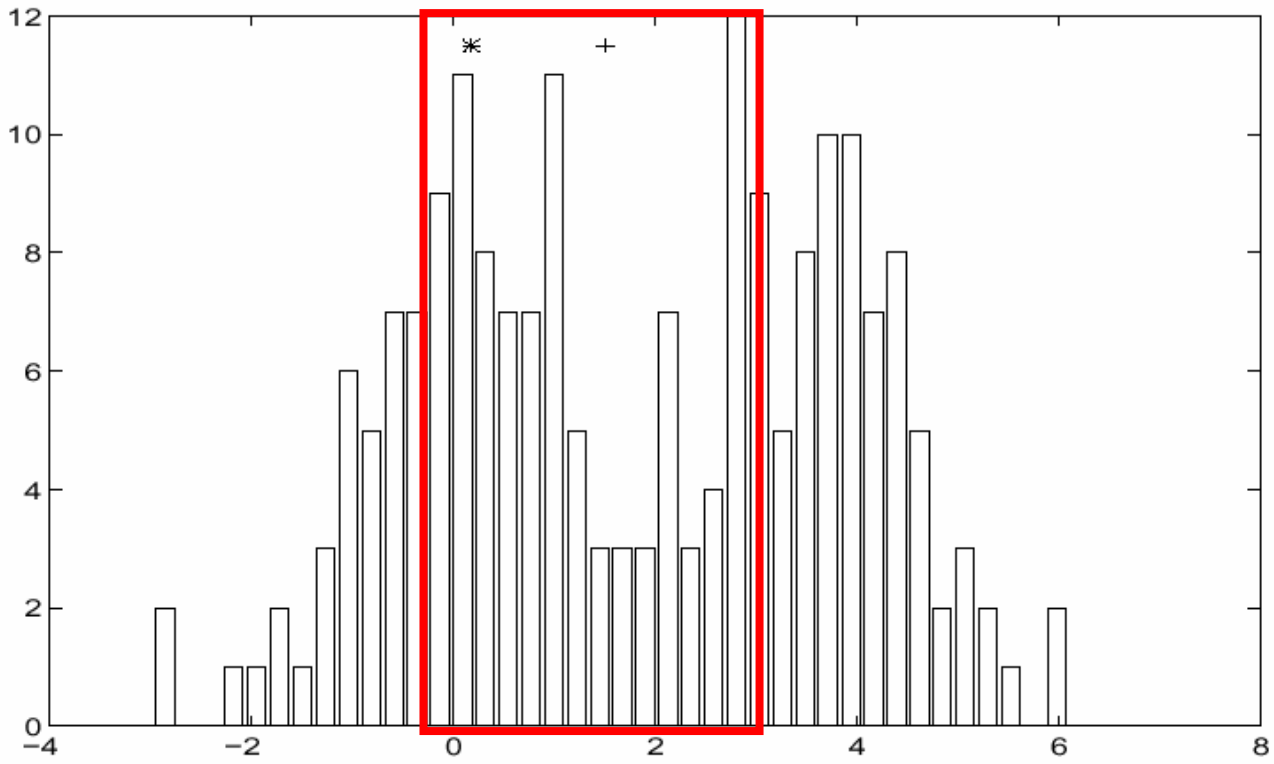


Mean Shift Segmentation Results



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

Mean Shift [Comaniciu & Meer]

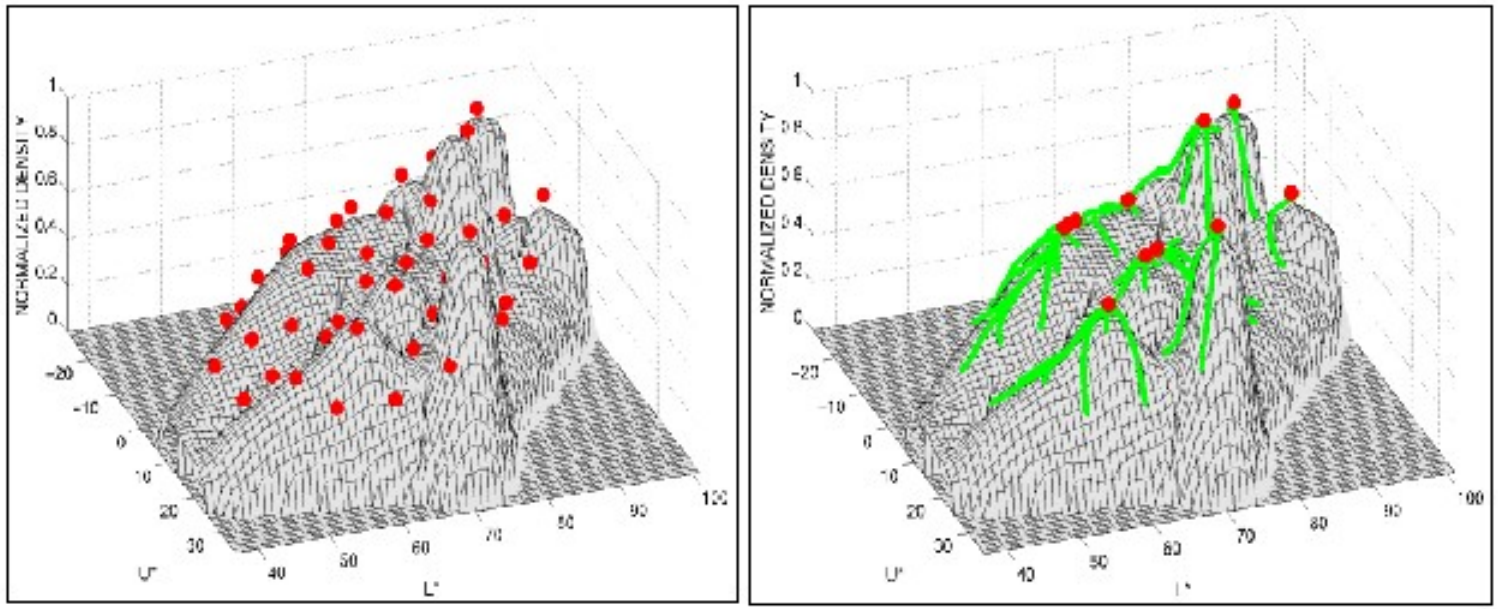


- Iterative Mode Search
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the “mean”) of W: $\sum_{x \in W} xH(x)$
  3. Translate the search window to the mean
  4. Repeat Step 2 until convergence

Mean-Shift

- Approach
  - Initialize a window around each point
  - See where it shifts—this determines which segment it's in
  - Multiple points will shift to the same segment

- Classification based on the basin of attraction.



Mean shift trajectories

Mean-shift for image segmentation

- Useful to take into account spatial information
  - instead of (R, G, B), run in (R, G, B, x, y) space
  - D. Comaniciu, P. Meer, Mean shift analysis and applications, *7th International Conference on Computer Vision*, Kerkyra, Greece, September 1999, 1197-1203.
    - http://www.caip.rutgers.edu/riul/research/papers/pdf/spatmsft.pdf

Figure 2: The *house* image, 255 × 192 pixels, 9603 colors.

More Examples: http://www.caip.rutgers.edu/~comanici/segm_images.html

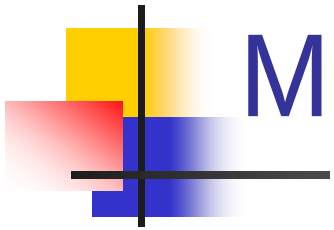
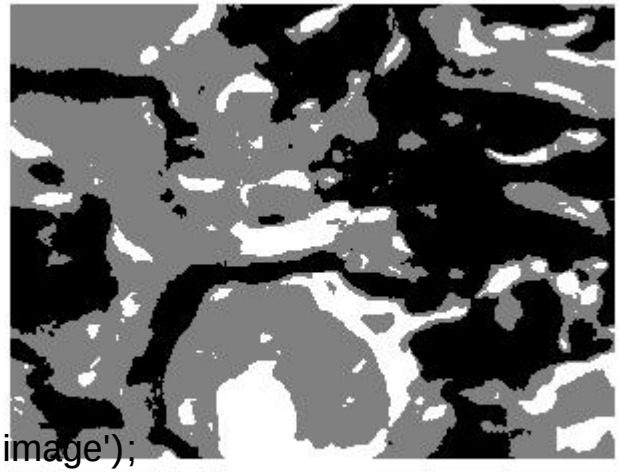
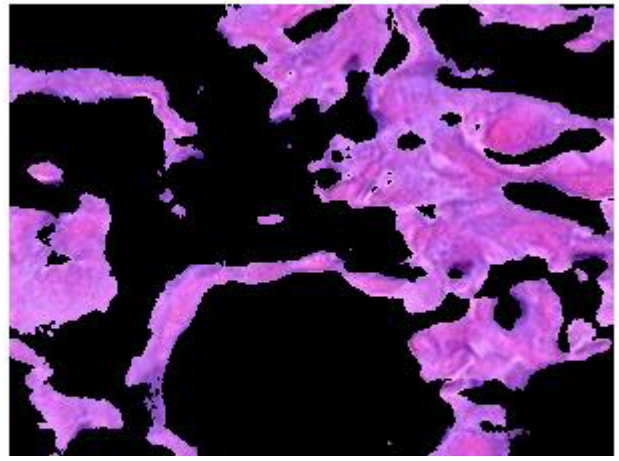


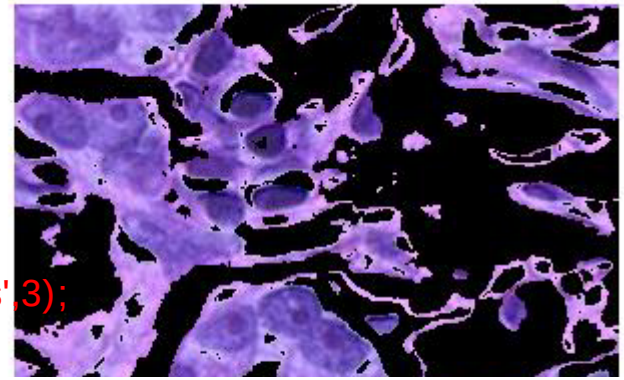
image labeled by cluster index



objects in cluster 1

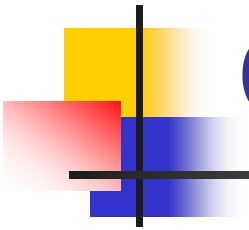


objects in cluster 2



objects in cluster 3

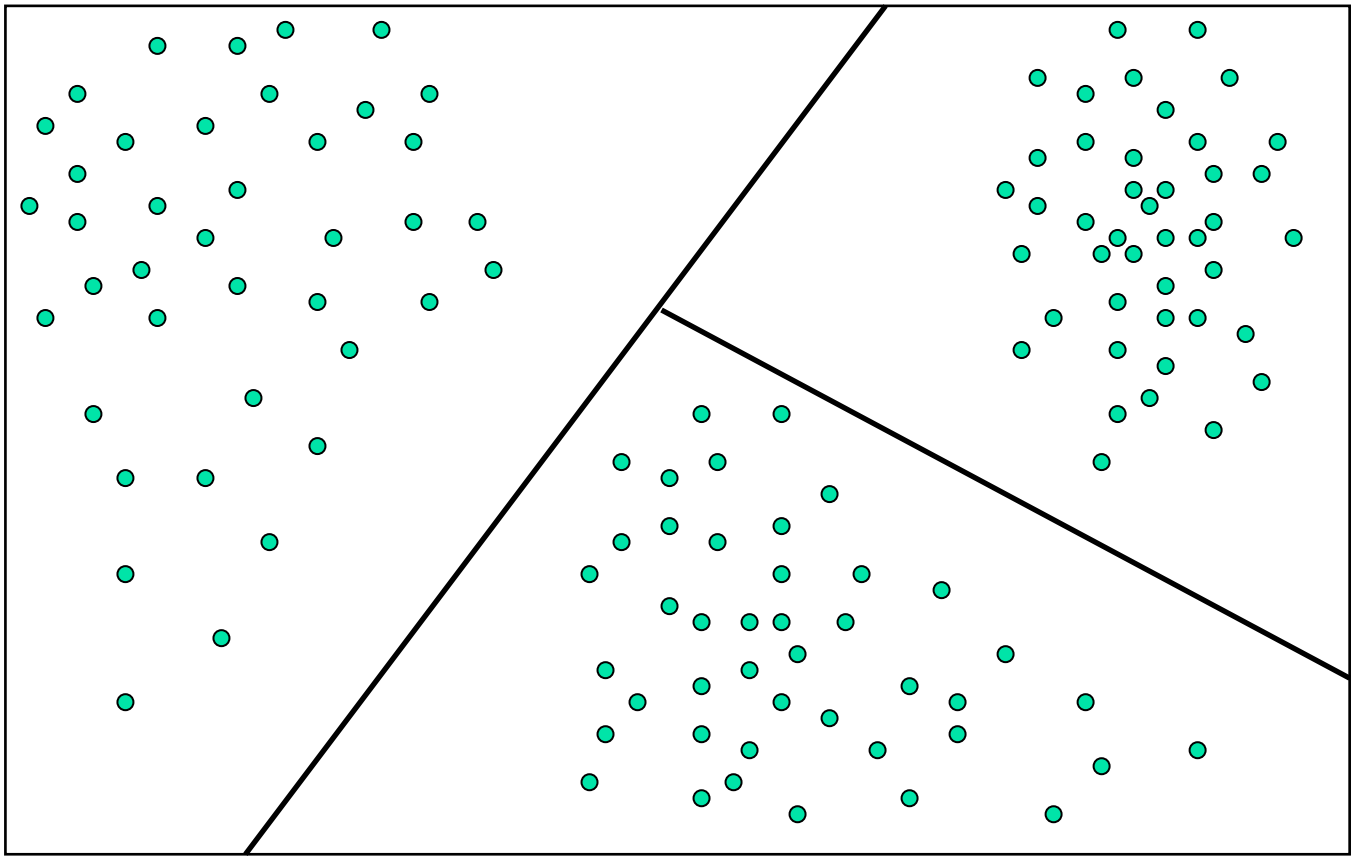
```
he = imread('hestain.png');
figure,imshow(he), title('H&E image');
cform = makecform('srgb2lab');
lab_he = applycform(he,cform);
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);
nColors = 3; % repeat the clustering 3 times to avoid local minima
[cluster_idx cluster_center] = kmeans(ab,nColors,'distance','sqEu','Replicates',3);
pixel_labels = reshape(cluster_idx,nrows,ncols);
figure,imshow(pixel_labels,[]), title('image labeled by cluster index');
segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);
for k = 1:nColors
    color = he;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
end
figure,imshow(segmented_images{1}), title('objects in cluster 1');
figure,imshow(segmented_images{2}), title('objects in cluster 2');
figure,imshow(segmented_images{3}), title('objects in cluster 3');
```



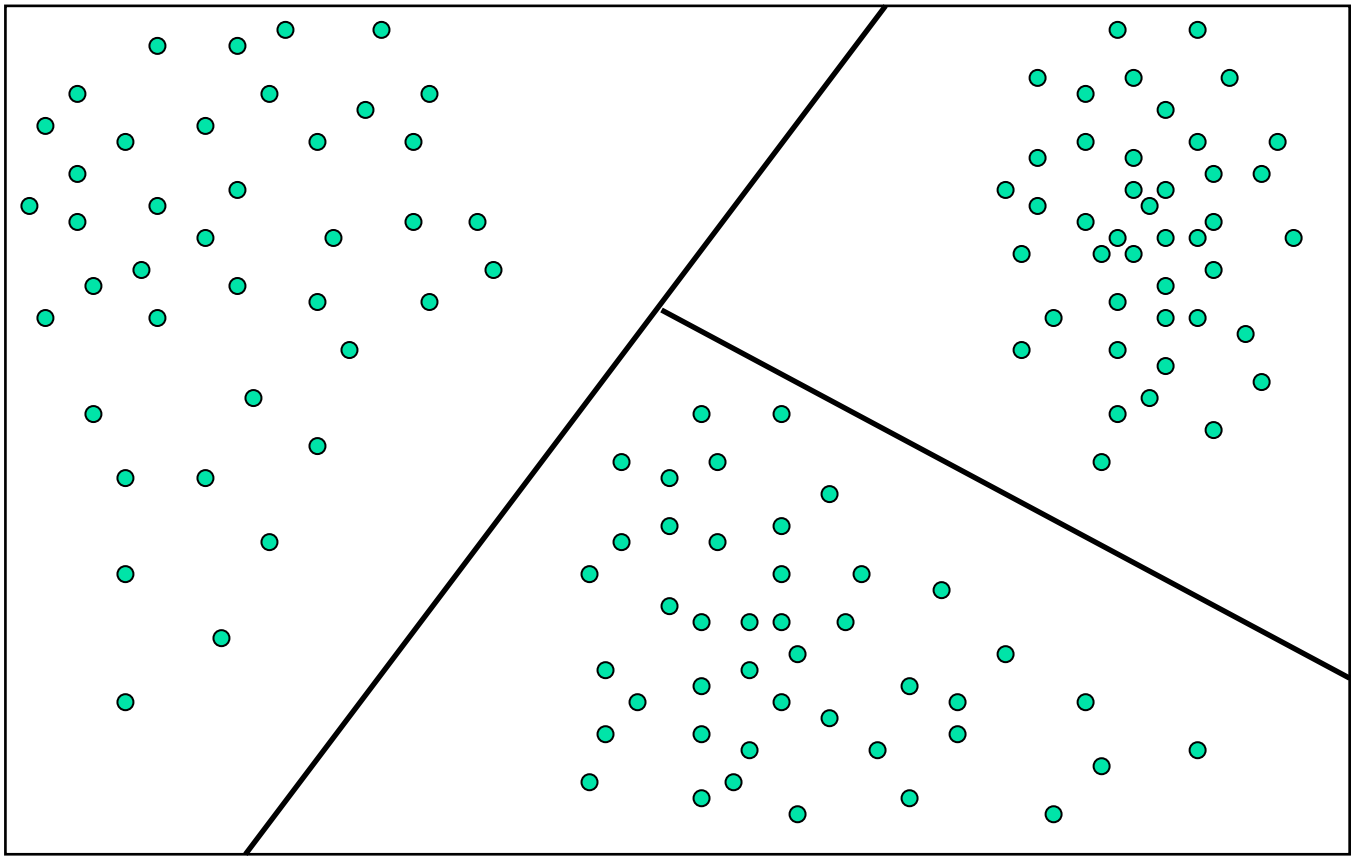
Outline

- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph Cuts

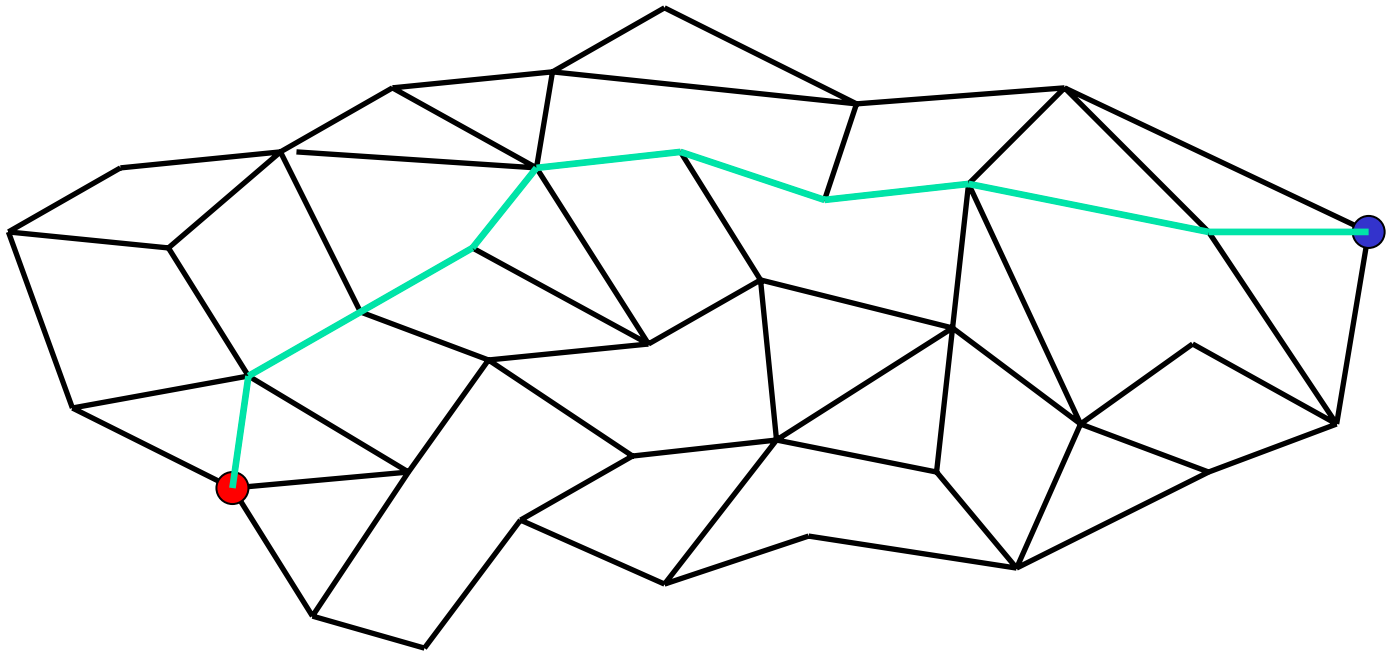
Clustering – Determine Regions



Graph Cut – Determine Boundaries



Preface—shortest path



How to find the shortest path connecting the start point and the end point ?

Intelligent Scissors (demo)

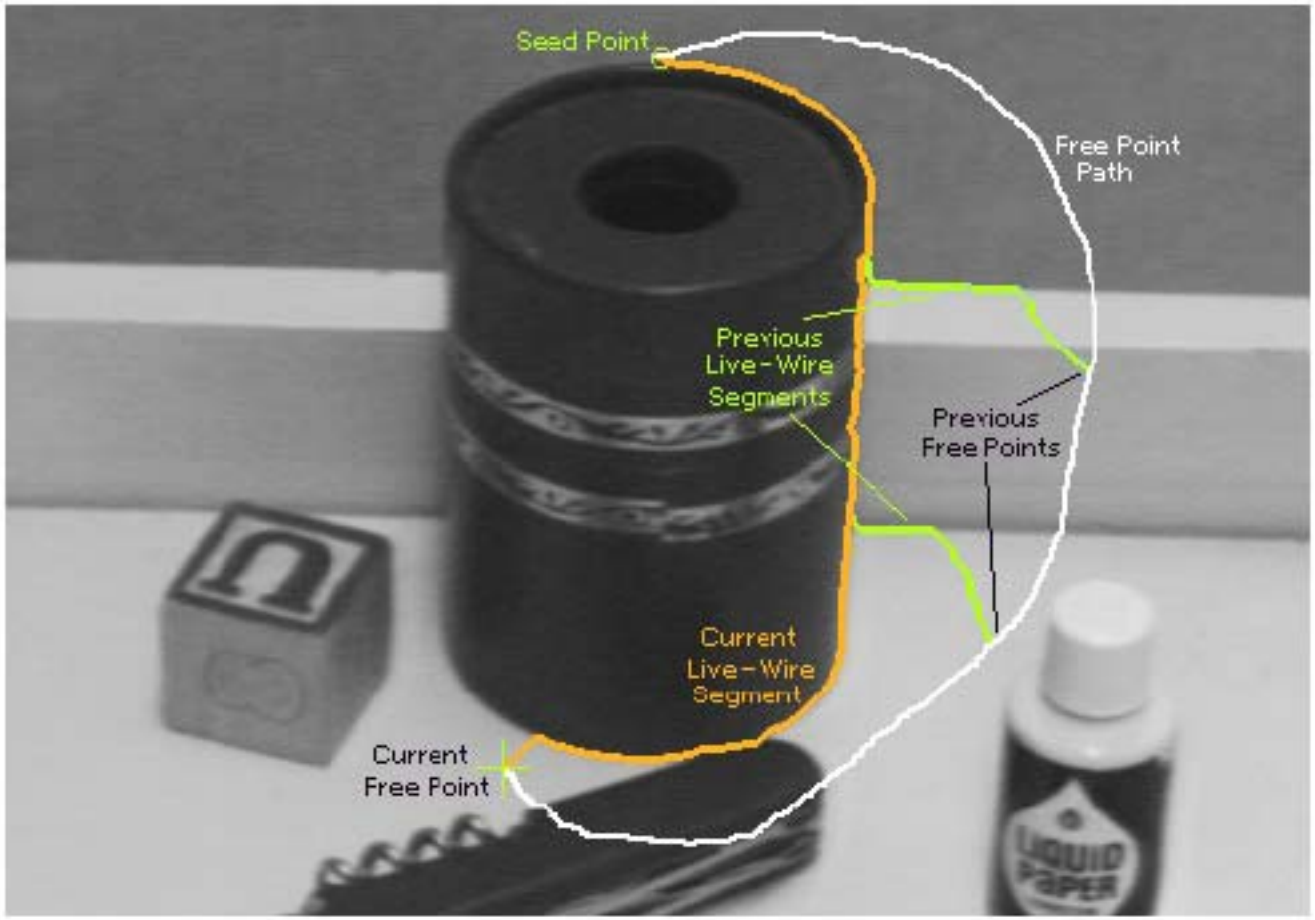


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors [Mortensen 95]

- Approach answers a basic question
  - Q: how to find a path from seed to mouse that follows object boundary as closely as possible?

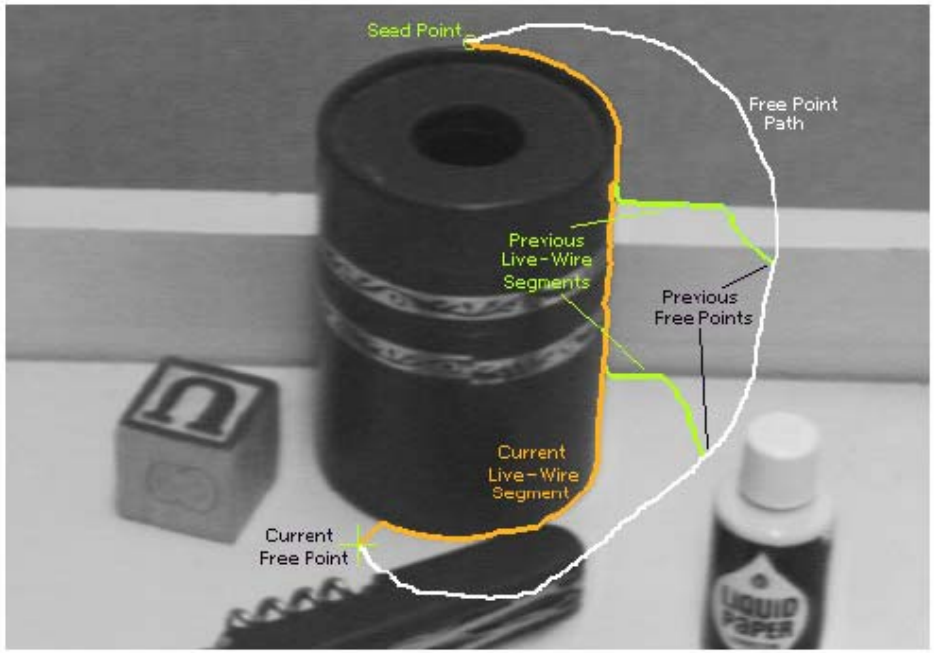


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors

- Basic Idea
  - Define edge score for each pixel
    - edge pixels have low cost
  - Find lowest cost path from seed to mouse

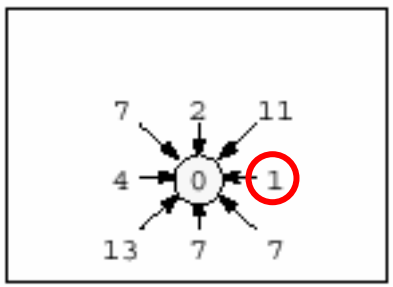
Questions

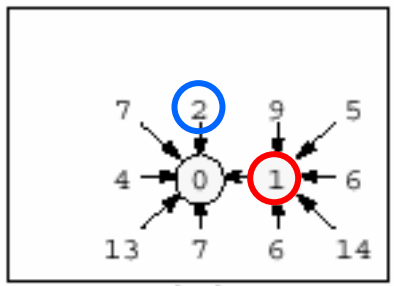
- How to define costs?
- How to find the path?

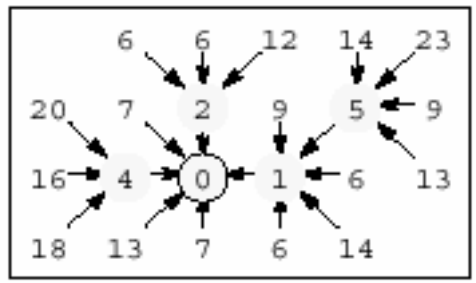
Path Search (basic idea)

- Graph Search Algorithm
  - Computes minimum cost path from seed to *all other pixels*

11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	2	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

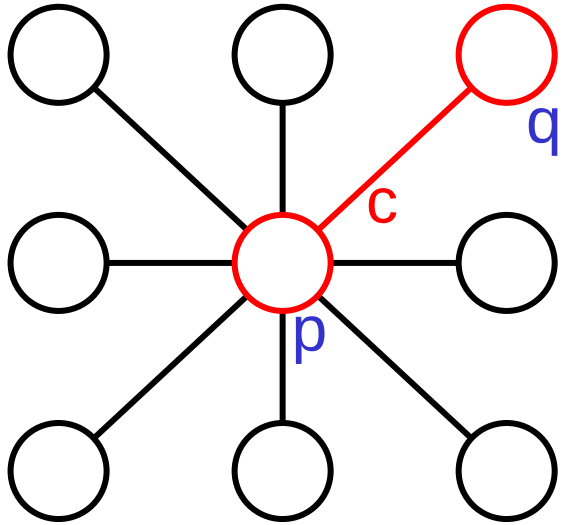






How does this really work?

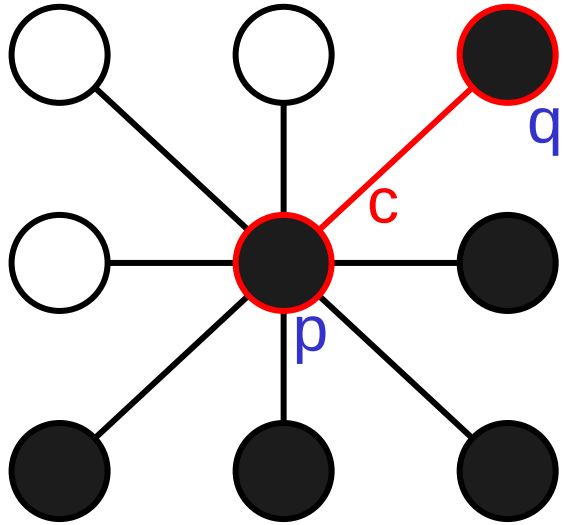
- Treat the image as a graph



- Graph
  - node for every pixel **p**
  - link between every adjacent pair of pixels, **p**,**q**
  - cost **c** for each link
- Note: each *link* has a cost
  - this is a little different than the figure before where each pixel had a cost

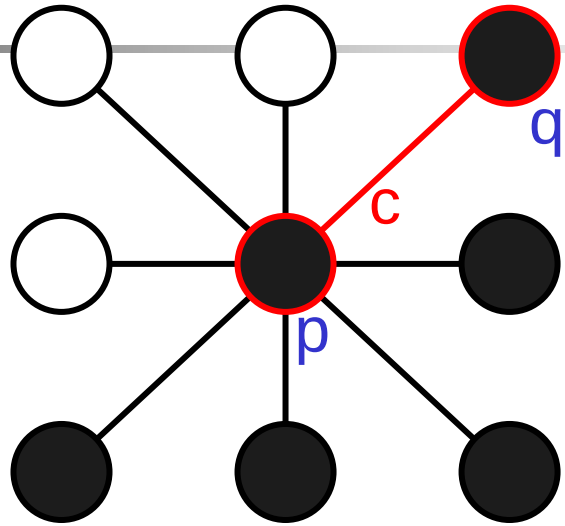
Defining the costs

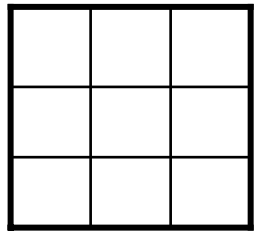
- Treat the image as a graph



- Want to hug image edges: how to define cost of a link?
  - the link should follow the intensity edge
    - want intensity to change rapidly with respect to the link
  - $c \approx$ - |difference of intensity with respect to link|

Defining the costs

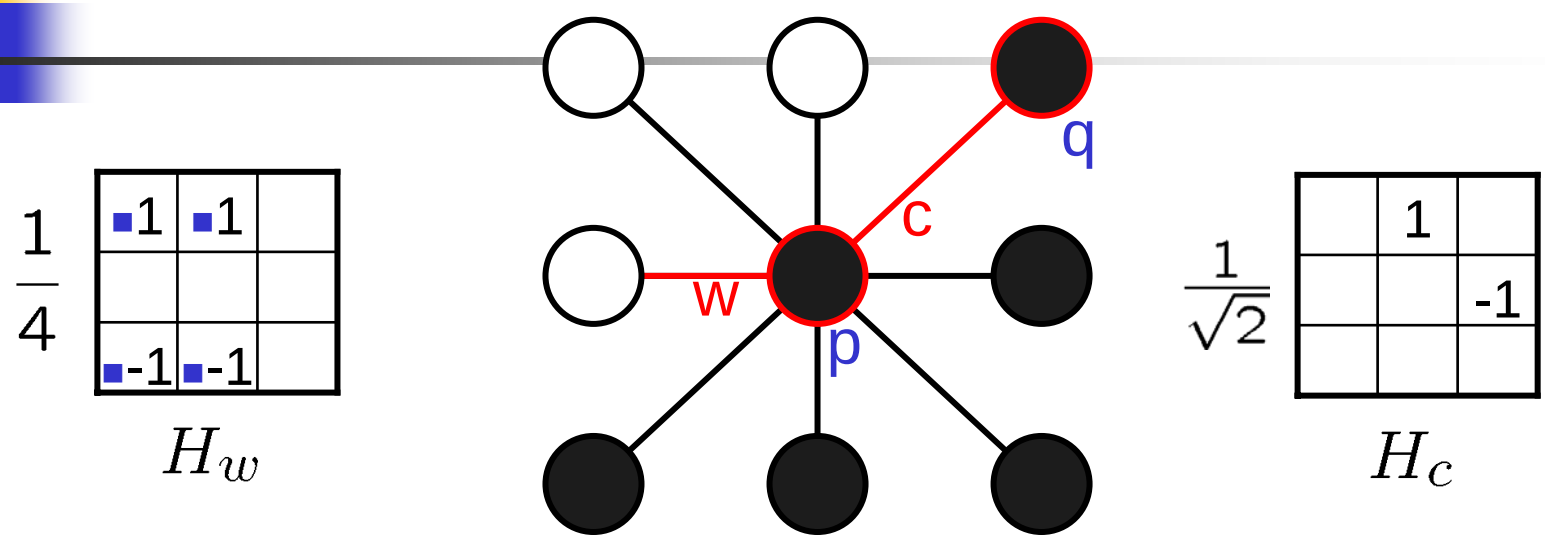




H_c

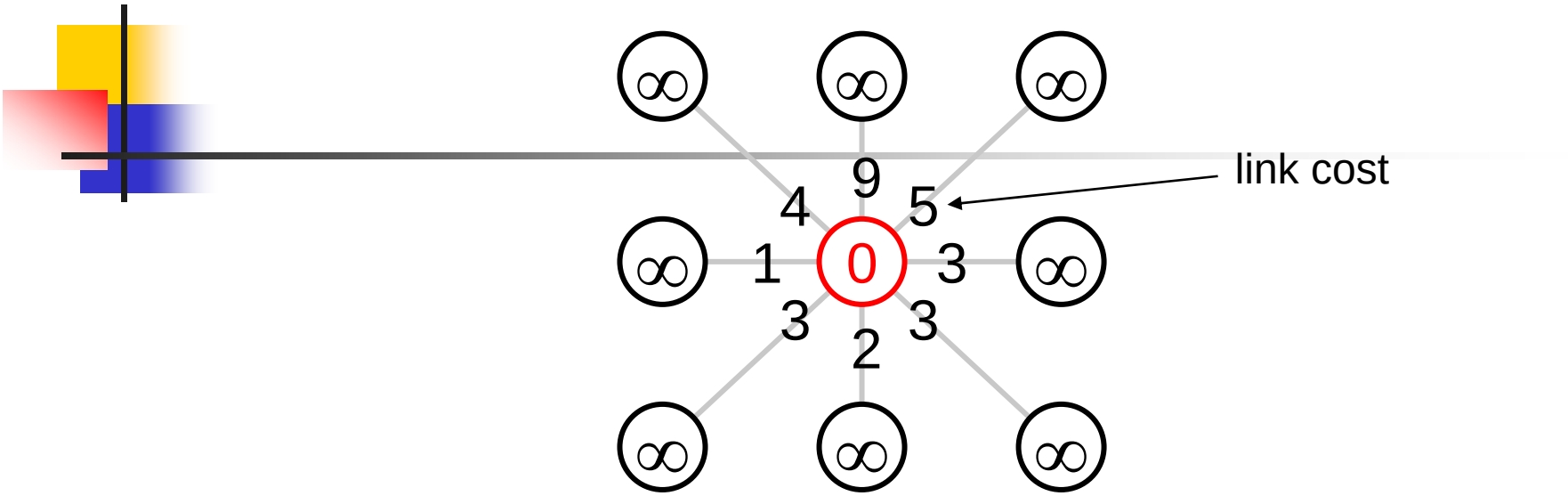
- c can be computed using a cross-correlation filter
  - assume it is centered at p
- Also typically scale c by its length
  - set c = (max-|filter response|)
    - where max = maximum |filter response| over all pixels in the image

Defining the costs



- c can be computed using a cross-correlation filter
  - assume it is centered at p
- Also typically scale c by its length
  - set c = (max-|filter response|)
    - where max = maximum |filter response| over all pixels in the image

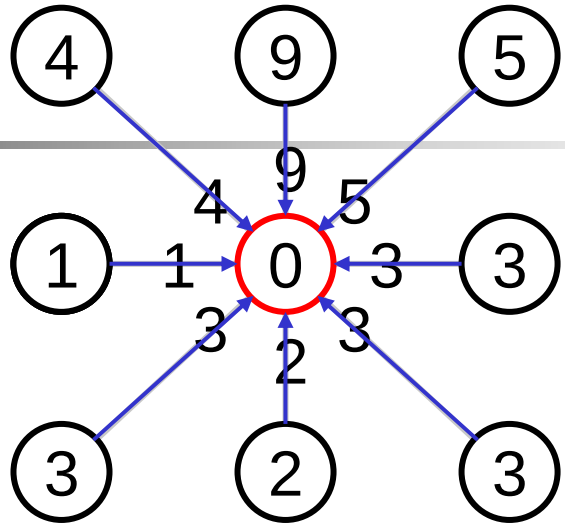
Dijkstra's shortest path algorithm



- Algorithm
  1. init node costs to ∞,
  2. Init an active set p = seed point, cost(p) = 0
  3. expand p as follows:

     for each of p's neighbors q that are not expanded
     - set cost(q) = min(cost(p) + c_{pq}, cost(q))

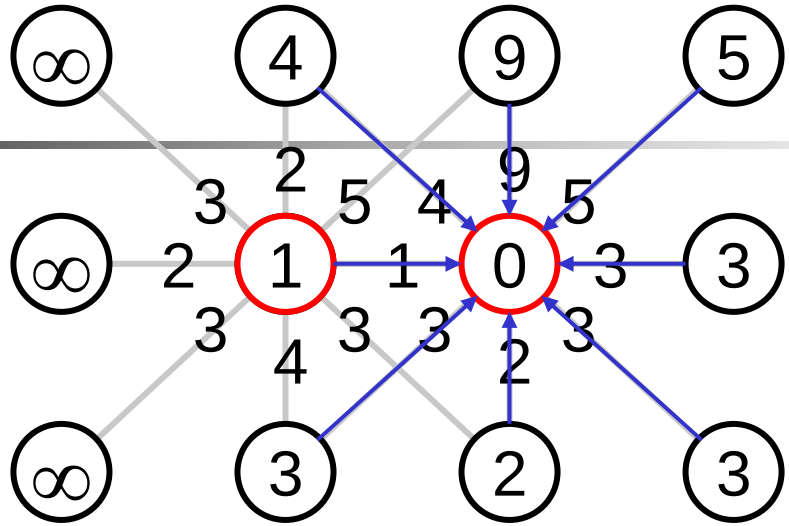
Dijkstra's shortest path algorithm



- Algorithm
  1. init active set p = seed point, cost(p) = 0
  2. expand p as follows:

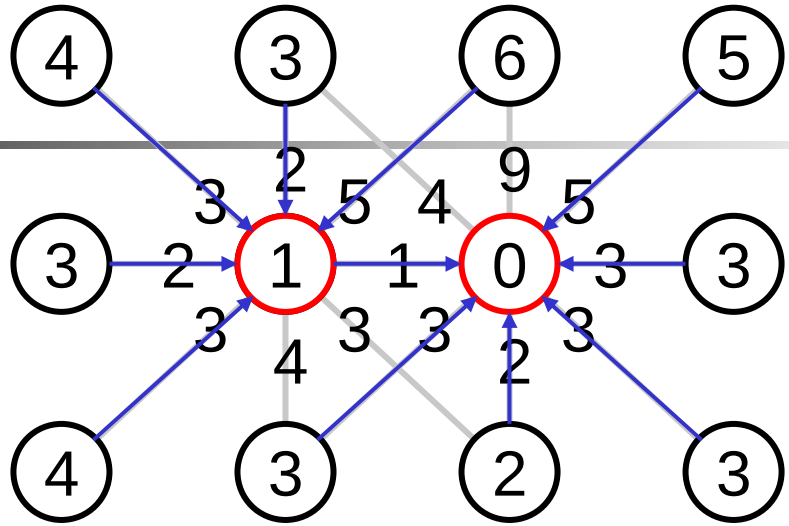
     for each of p's neighbors q that are not expanded
     - set cost(q) = min(cost(p) + c_{pq}, cost(q))
     - if q's cost changed, make q point back to p
  3. set *r = node with minimum cost* on the ACTIVE list

Dijkstra's shortest path algorithm



- Algorithm
  1. init node costs to ∞, set p = seed point, cost(p) = 0
  2. expand p as follows:
     for each of p's neighbors q that are not expanded
     - set cost(q) = min(cost(p) + c_{pq}, cost(q))
     - if q's cost changed, make q point back to p
  3. set *r = node with minimum* cost on the ACTIVE list
  4. repeat Step 2 for p = r

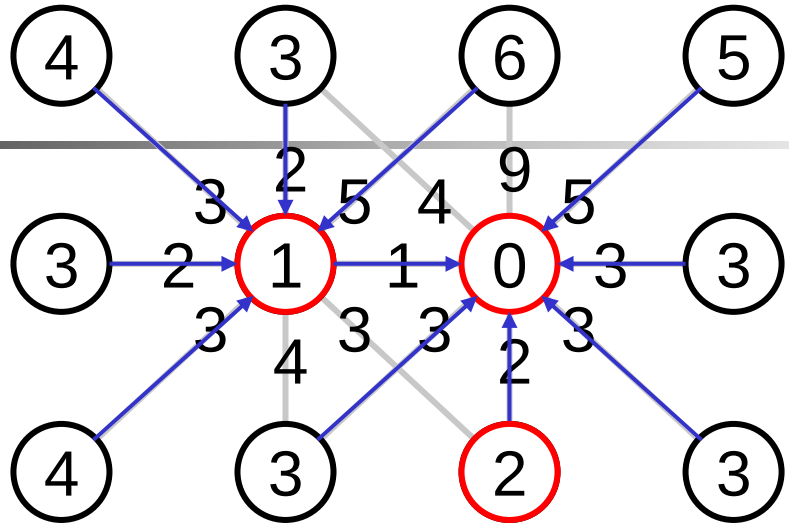
Dijkstra's shortest path algorithm



- Algorithm
  1. init node costs to ∞, set p = seed point, cost(p) = 0
  2. expand p as follows:

     for each of p's neighbors q that are not expanded
     - set cost(q) = min(cost(p) + c_{pq}, cost(q))
       - if q's cost changed, make q point back to p
     - put q on the ACTIVE list (if not already there)
  3. set r = node with minimum cost on the ACTIVE list
  4. repeat Step 2 for p = r

Dijkstra's shortest path algorithm



- Algorithm
  1. init node costs to ∞, set p = seed point, cost(p) = 0
  2. expand p as follows:

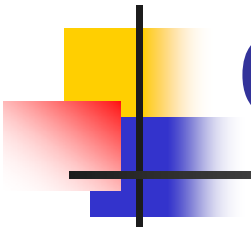
     for each of p's neighbors q that are not expanded
     - set cost(q) = min(cost(p) + c_{pq}, cost(q))
       - if q's cost changed, make q point back to p
     - put q on the ACTIVE list (if not already there)
  3. set r = node with minimum cost on the ACTIVE list
  4. repeat Step 2 for p = r

Dijkstra's shortest path algorithm

- Properties
  - It computes the minimum cost path from the seed to every node in the graph. This set of minimum paths is represented as a *tree*
  - Running time, with N pixels:
    - $O(N^2)$ time if you use an active list
    - O(N log N) if you use an active priority queue (heap)
    - takes fraction of a second for a typical (640x480) image
  - Once this tree is computed once, the optimal path can be extracted from any point to the seed in O(N) time. it runs in real time as the mouse moves
  - What happens when the user specifies a new seed?

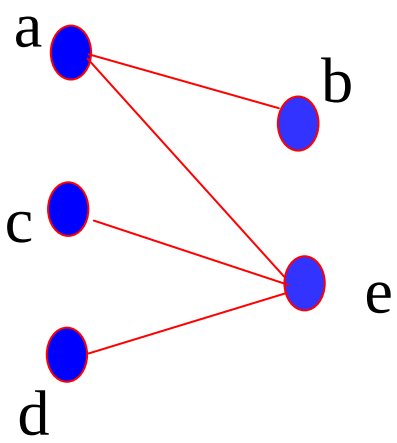
Intelligent Scissors Results

Graph theoretic clustering



- Represent tokens (which are associated with each pixel) using a weighted graph.
  - affinity matrix (亲合矩阵)
- Cut up this graph to get subgraphs with strong interior links and weaker exterior links

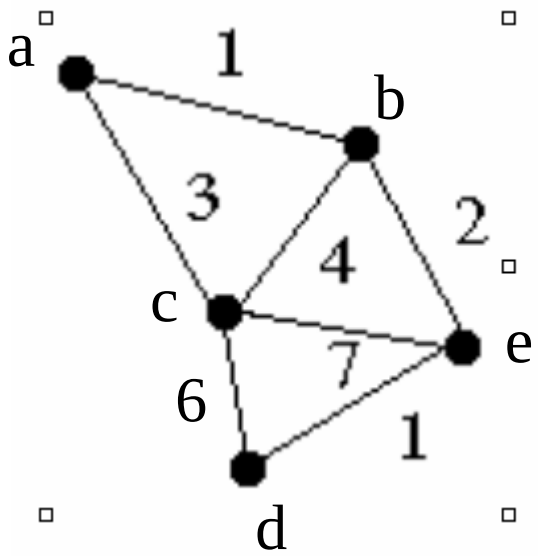
Graphs Representations



$$
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array}
\begin{array}{c}
\begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}
\end{array}
$$

Adjacency Matrix: W

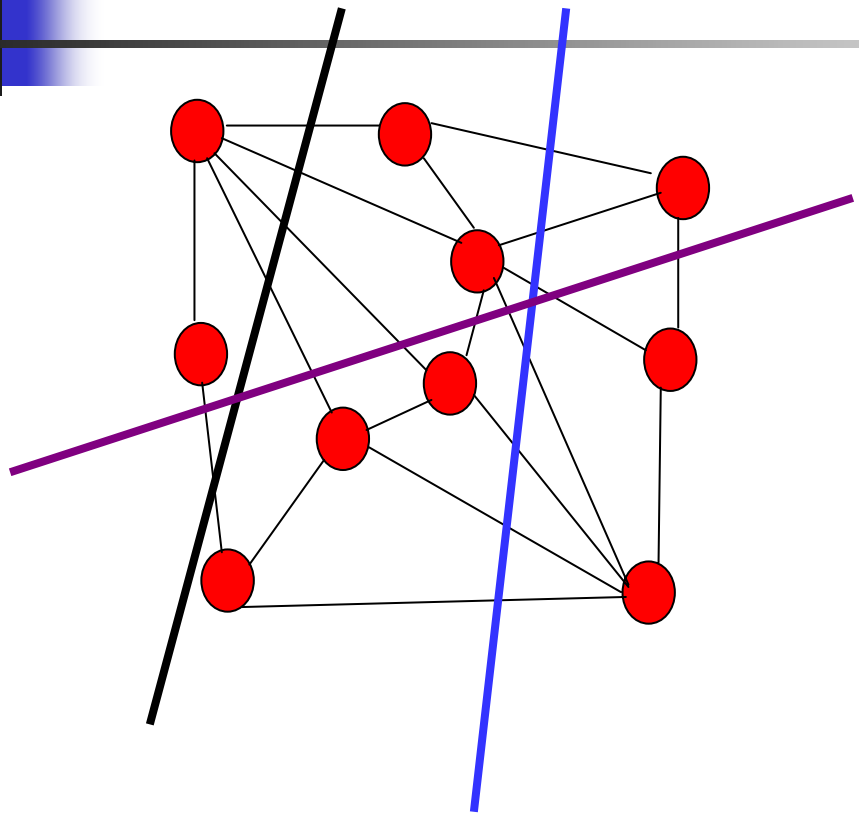
Weighted Graphs



	a	b	c	d	e
a	0	1	3	0	0
b	1	0	4	0	2
c	3	4	0	6	7
d	0	0	6	0	1
e	0	2	7	1	0

Weight Matrix: W

Minimum Cut



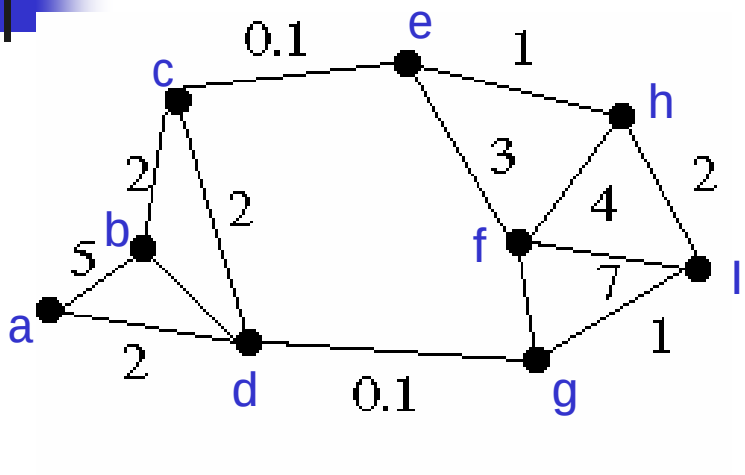
A cut of a graph G is the set of edges S such that removal of S from G disconnects G.

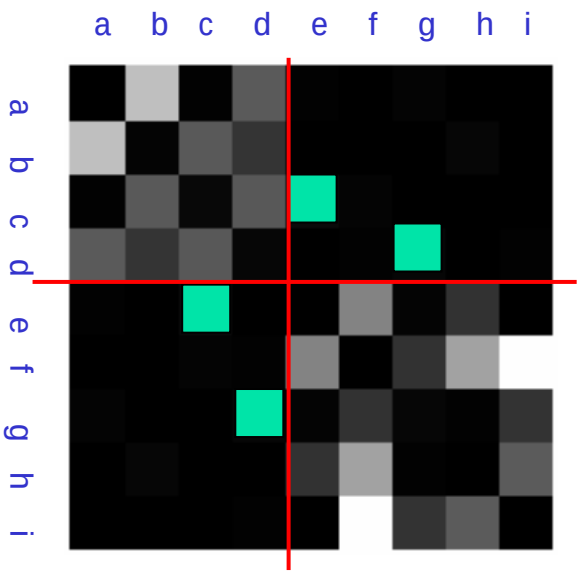
Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as

$$w(\langle A,B\rangle) = \sum_{x\in A, y\in B} w(x,y)$$

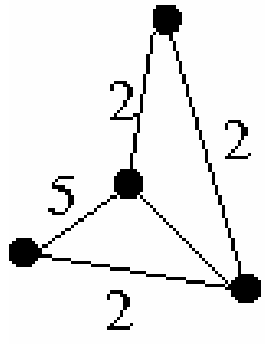
* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Minimum Cut and Clustering





	a	b	c	d	e	f	g	h	i
a	0	5	0	2	0	0	0	0	0
b	5	0	2	1	0	0	0	0	0
c	0	2	0	2	0.1	0	0	0	0
d	2	1	2	0	0	0	0.1	0	0
e	0	0	0.1	0	0	3	0	1	0
f	0	0	0	0	3	0	1	4	7
g	0	0	0	0.1	0	1	0	0	1
h	0	0	0	0	1	4	0	0	2
i	0	0	0	0	0	7	1	2	0



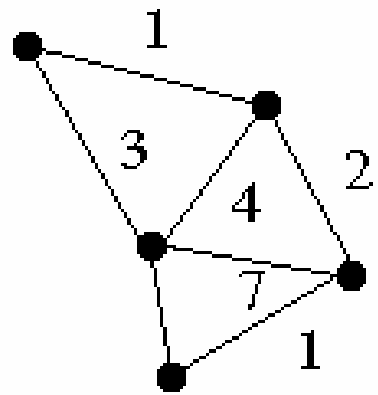
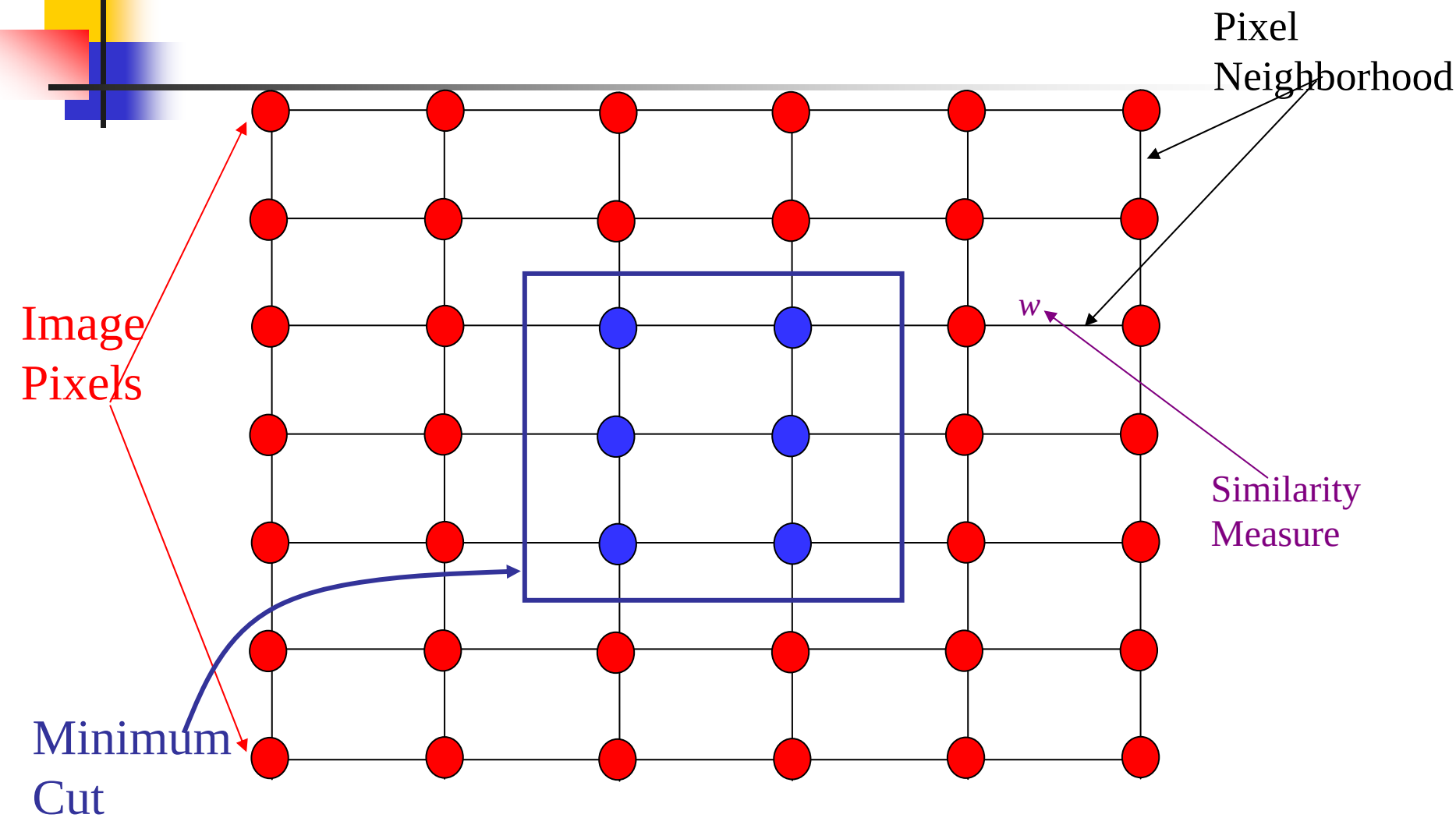
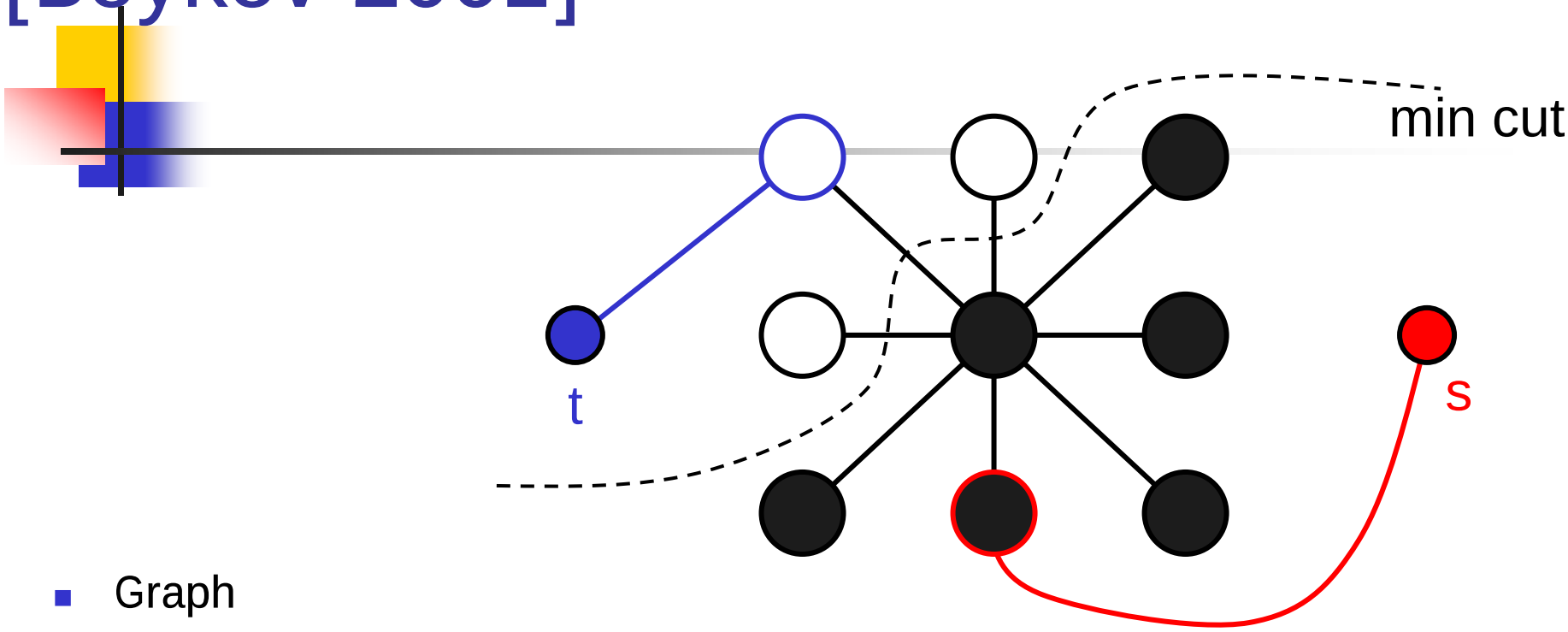


Image Segmentation & Minimum Cut

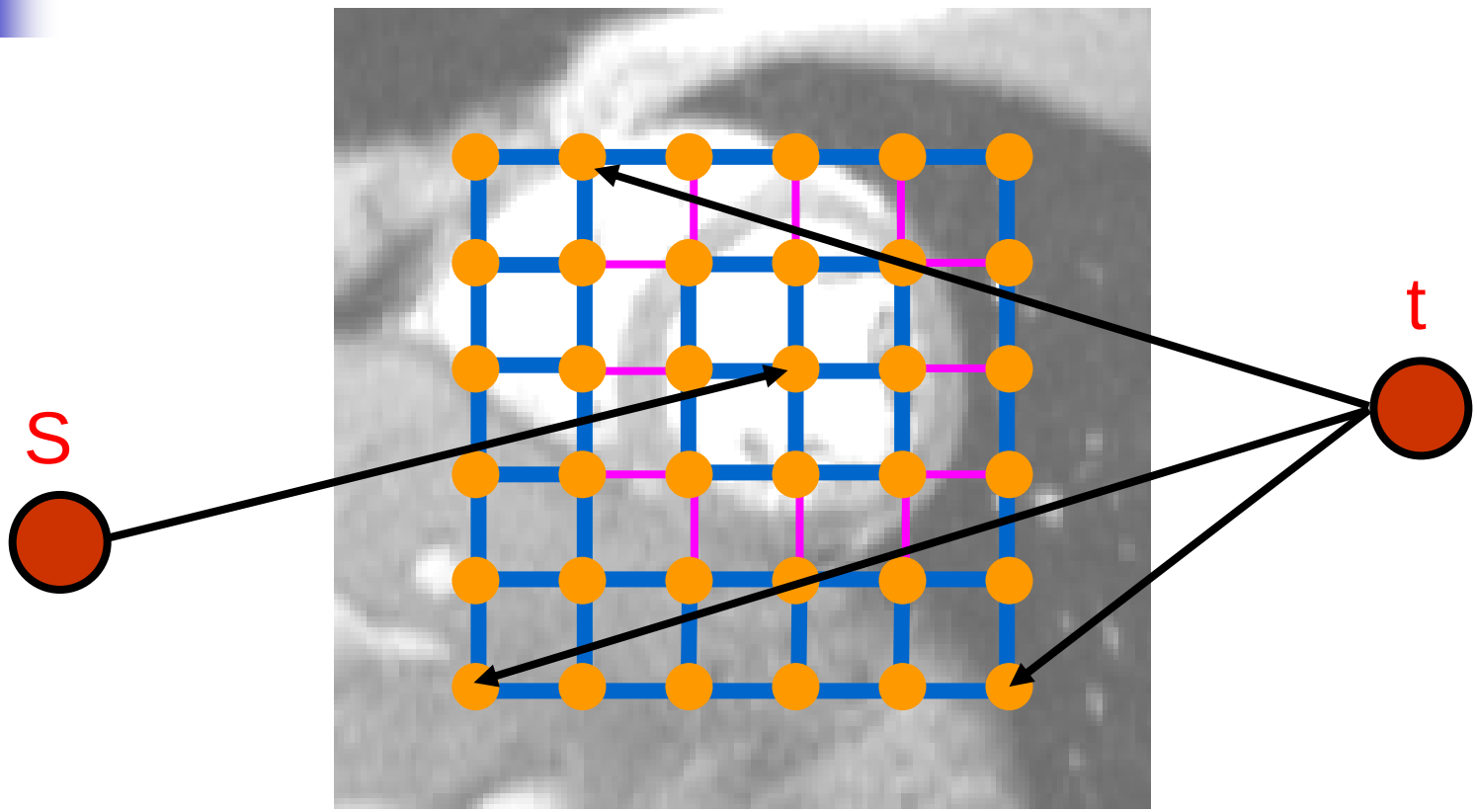


Segmentation by min (s-t) cut [Boykov 2001]

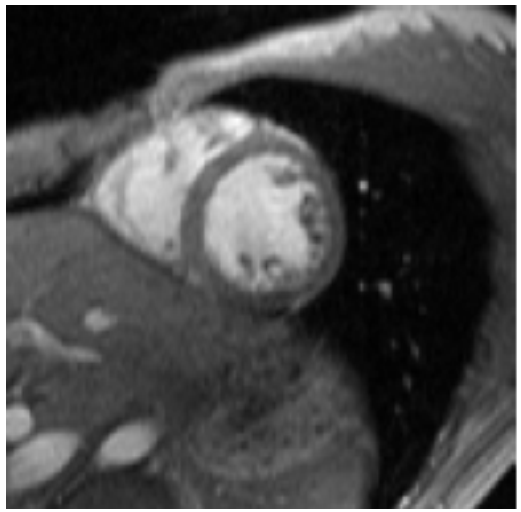


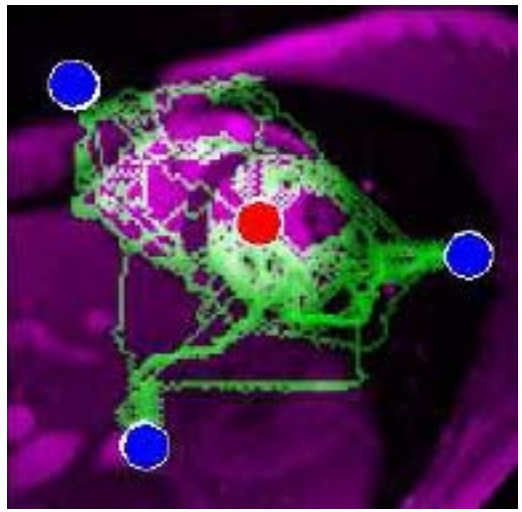
- Graph
  - node for each pixel, link between pixels
  - specify a few pixels as foreground and background
    - create an infinite cost link from each bg pixel to the “t” node
    - create an infinite cost link from each fg pixel to the “s” node
  - compute min cut that separates s from t
  - how to define link cost between neighboring pixels?

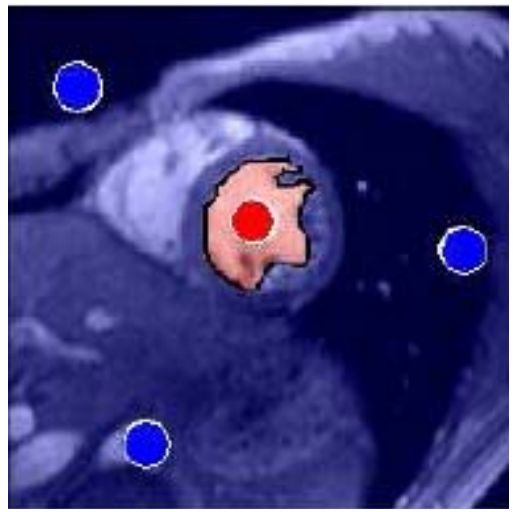
S-T Min-Cut/Max Flow



S-T Min-Cut/Max Flow

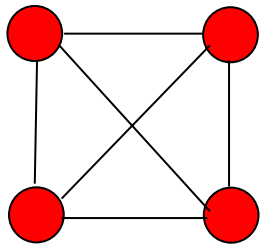






Minimum Cut

- There can be more than one minimum cut in a given graph

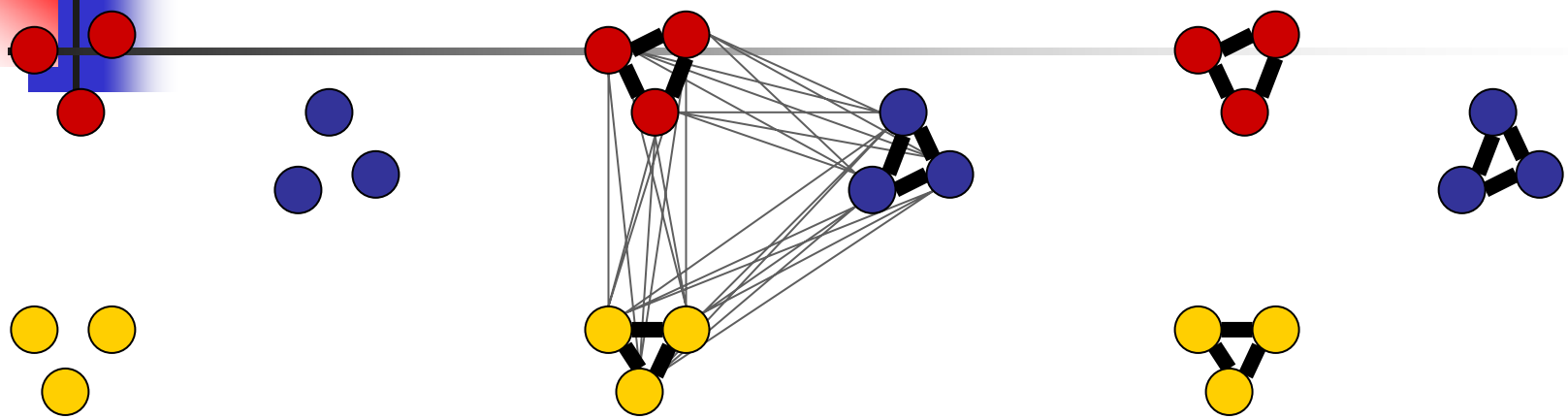


$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- All minimum cuts of a graph can be found in polynomial time[1].

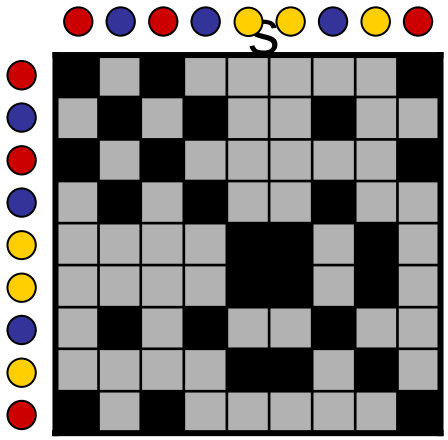
[1]H. Nagamochi, K. Nishimura and T. Ibaraki, “Computing all small cuts in an undirected network. SIAM J. Discrete Math. 10 (1997) 469-481.

Finding the Minimal Cuts: Spectral Clustering Overview

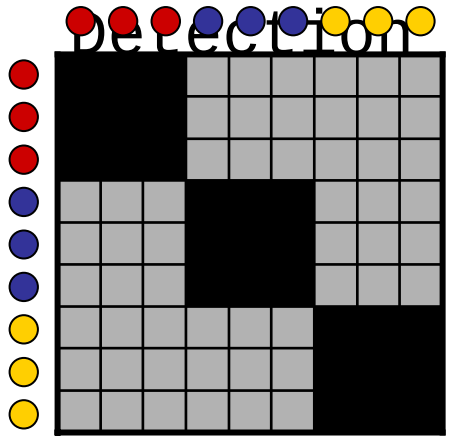


Data

Similaritie
s



Block-
Detection

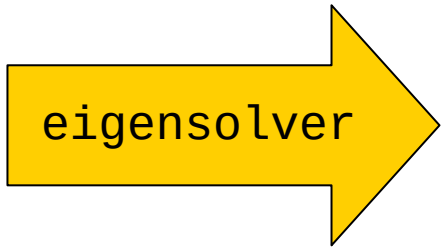


* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Eigenvectors and Blocks

- Block matrices have block eigenvectors:

1	1	0	0
1	1	0	0
0	0	1	1
0	0	1	1

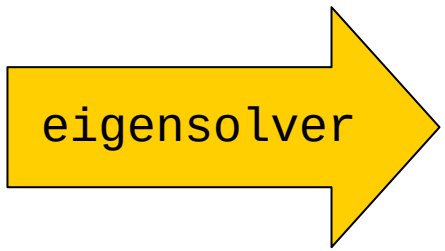


$_1 = 2$ $_2 = 2$ $_3 = 0$ $_4 = 0$

.71	0
.71	0
0	.71
0	.71

- Near-block matrices have near-block eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



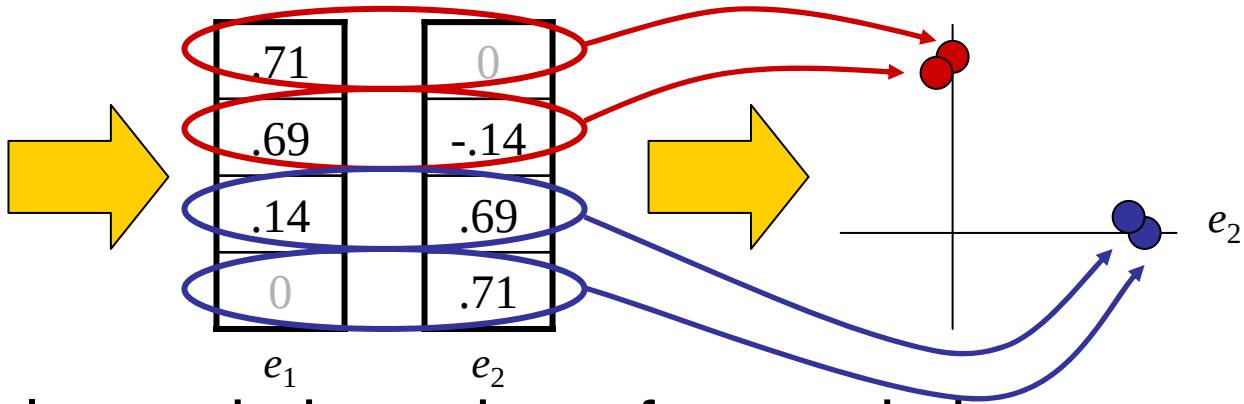
$_1 = 2.02$ $_2 = 2.02$ $_3 = -0.02$ $_4 = -0.02$

.71	0
.69	-.14
.14	.69
0	.71

Spectral Space

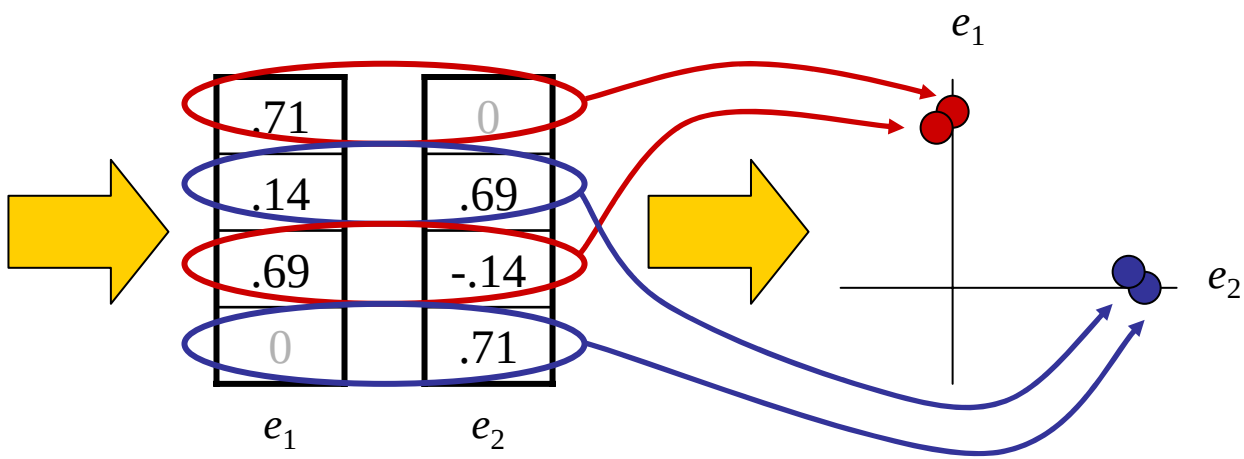
- Can put items into blocks by eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



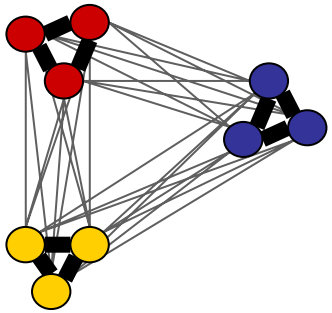
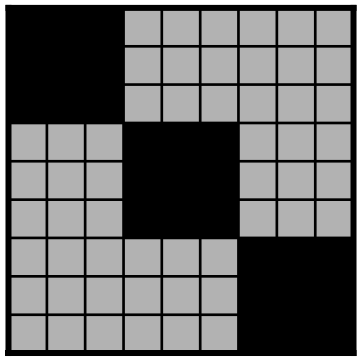
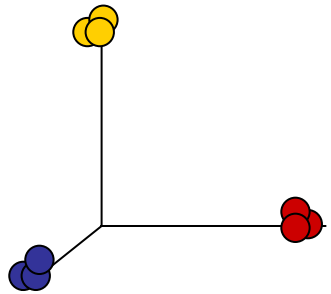
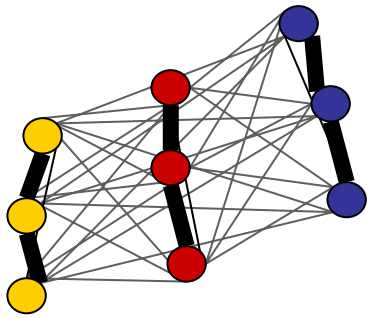
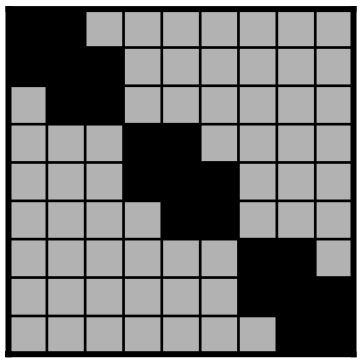
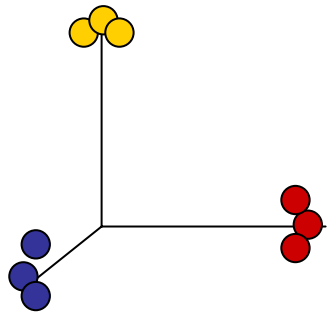
- Resulting clusters independent of row ordering:

1	.2	1	0
.2	1	0	1
1	0	1	-.2
0	1	-.2	1



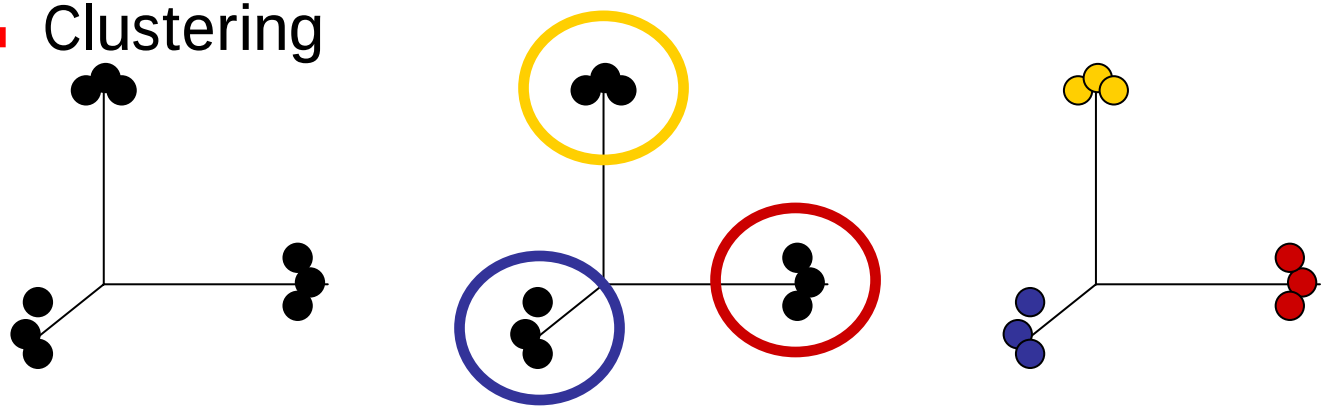
The Spectral Advantage

- The key advantage of spectral clustering is the spectral space representation:

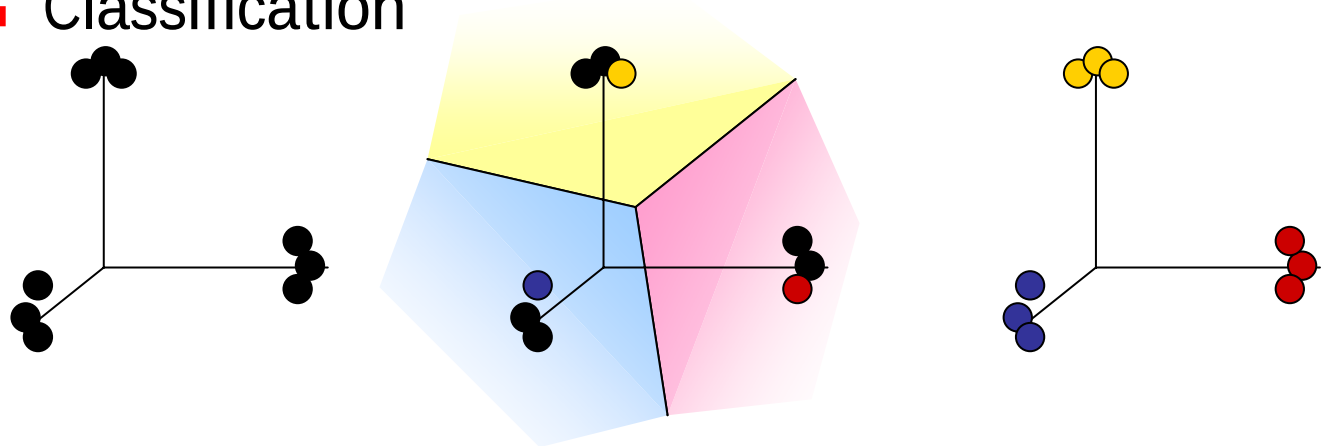







Clustering and Classification

- Once our data is in spectral space:
  - Clustering



  - Classification



Measuring Affinity(亲合力)

Intensity

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_i^2\right)\left(\|I(x)-I(y)\|^2\right)\right\}$$

Distance

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$

Texture

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_t^2\right)\left(\|c(x)-c(y)\|^2\right)\right\}$$

Geometry, Topology, Motion…

Eigenvectors and cuts

- Simplest idea: we want a vector *a* giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize

$$a^T A a$$

- But need the constraint

$$a^T a = 1$$

- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

Eigenvectors and cuts

- Maximize $a^T A a$ subject to $a^T a = 1$

- Introduce the Lagrange multiplier λ, The Lagrangian is

$$a^T A a + \lambda(a^T a - 1)$$

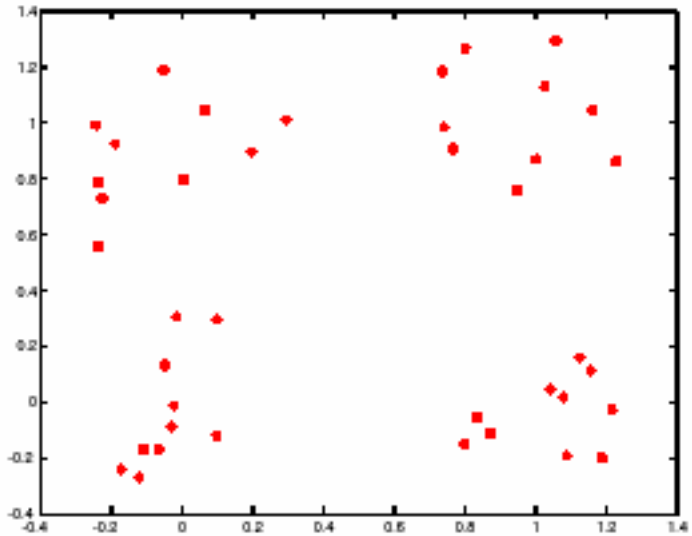
- Differentiation and dropping a factor of two yields

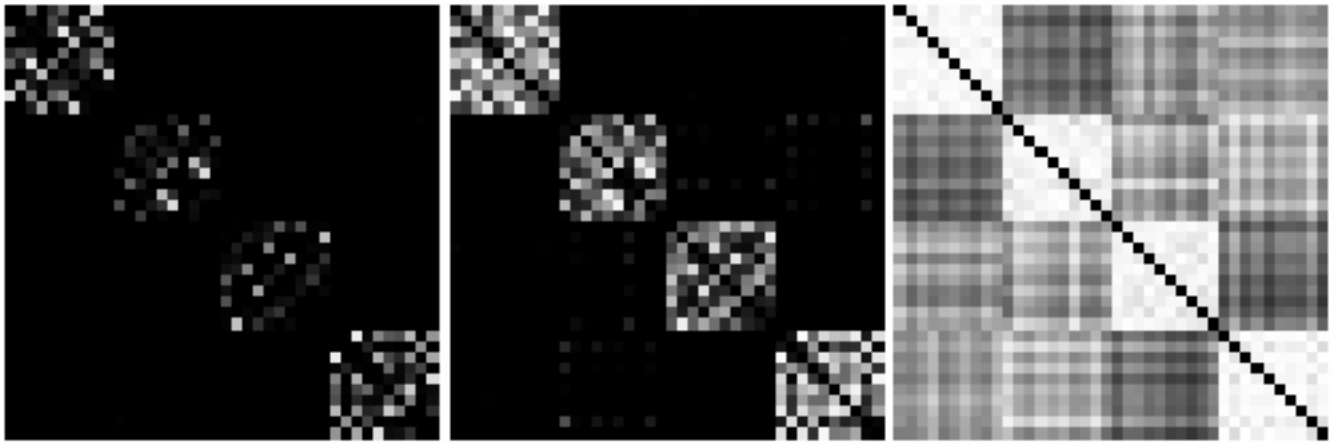
$$Aa = \lambda a$$

- *a* is an eigenvector of *A*

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_t^2\right)\left(\|c(x)-c(y)\|^2\right)\right\}$$

Scale affects affinity





σ=.1 σ=.2 σ=1

Scale affects affinity

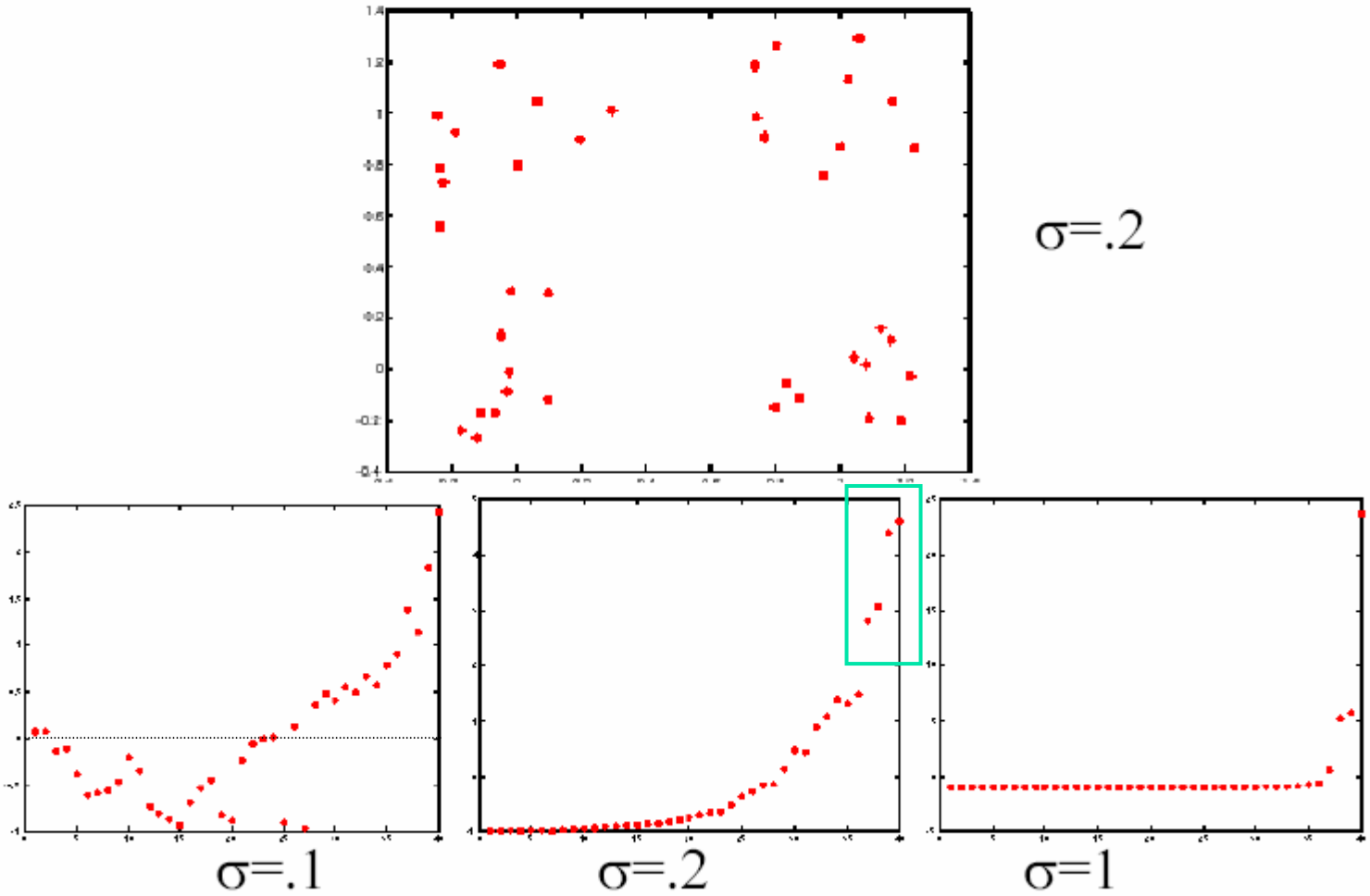
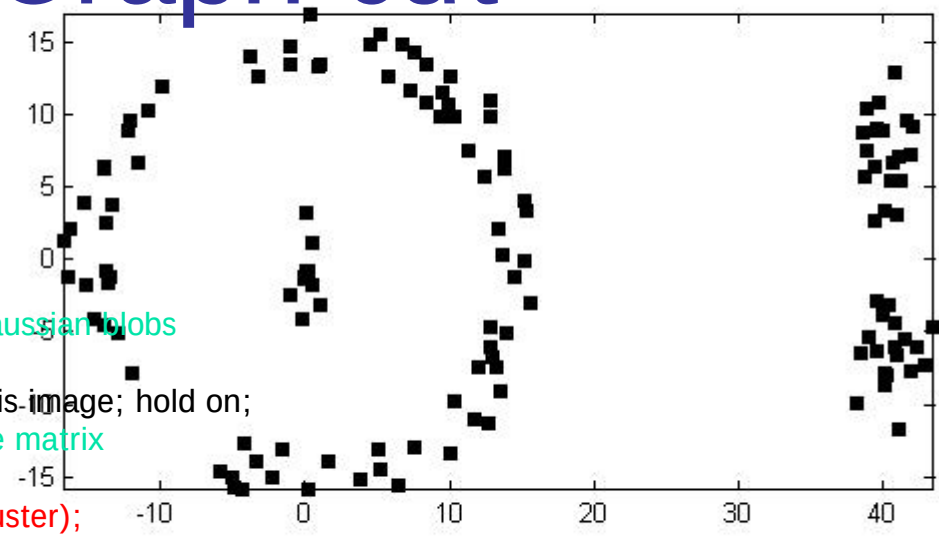
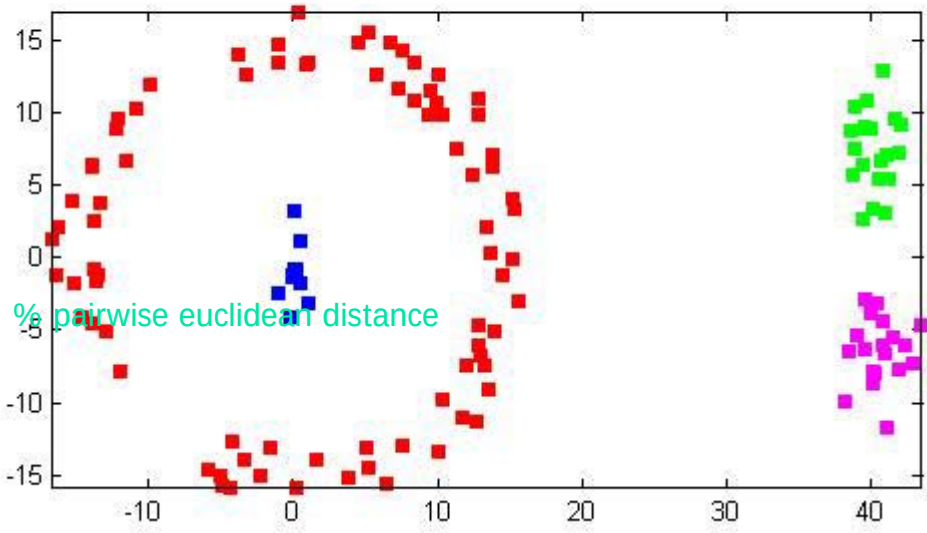


FIGURE 15.21: The number of clusters is reflected in the eigenvalues of the affinity matrix.

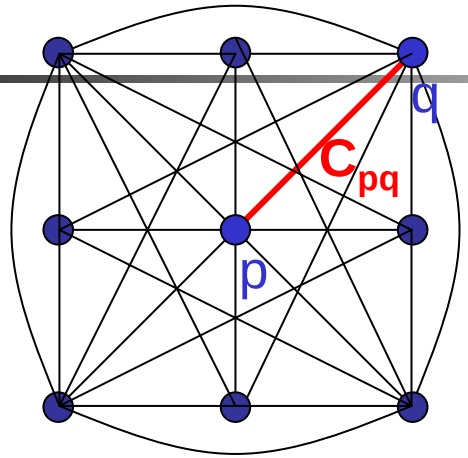
Matlab demo 4—Graph cut

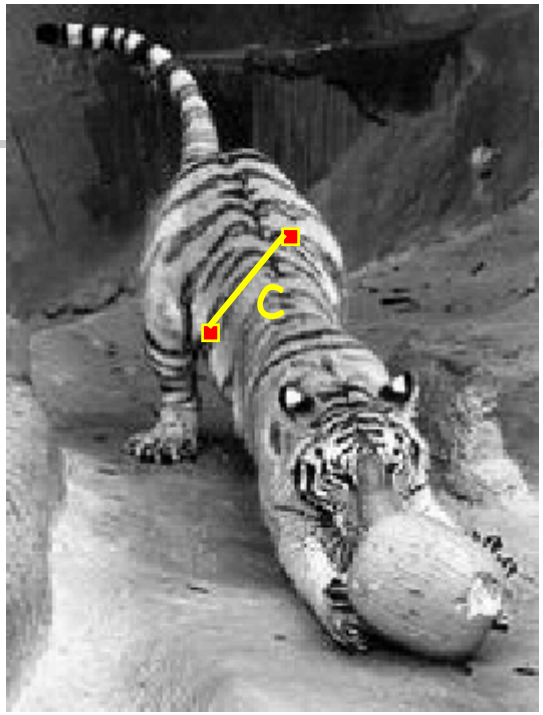



```
caseid = 3;
[data,size_cluster] = build_scene(caseid);%one circular plus two gaussian blobs
figure(1);clf;
plot(data(1,:),data(2,:),'ks', 'MarkerFaceColor','k','MarkerSize',5); axis image; hold on;
[W,Dist] = compute_relation(data);  %data is a 2D point coordinate matrix
nbCluster = 4;%the number of clusters has to be preassigned
[NcutDiscrete,NcutEigenvectors,NcutEigenvalues] = ncutW(W,nbCluster);
cluster_color = ['rgbmyc']; % display clustering result
figure(2);clf;
for j=1:nbCluster,
    id = find(NcutDiscrete(:,j));
    plot(data(1,id),data(2,id),[cluster_color(j),'s'], 'MarkerFaceColor',cluster_color(j),'MarkerSize',5);
    hold on;
end
hold off; axis image;

function [A,B] = compute_relation(data,scale_sig,order)
order = 2;
B = zeros(length(data),length(data));   %build the affinity matrix
for j = 1:length(data),
  B(j,:) = ( sqrt( (data(1,:)-data(1,j)).^2 +(data(2,:)-data(2,j)).^2)); % pairwise euclidean distance
end
scale_sig = 0.05*max(B(:));
tmp = (B/scale_sig).^order;
A = exp(-tmp);  %Mahalanobis distance based probability matrix
```



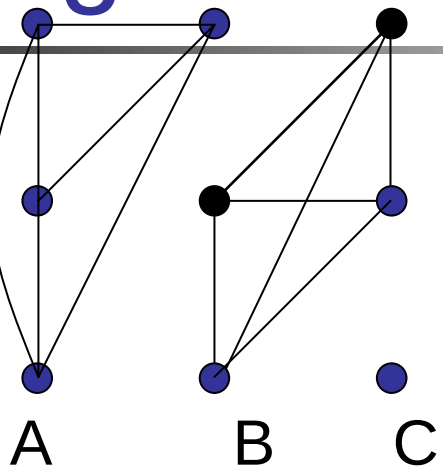
Summary: Automatic graph cut





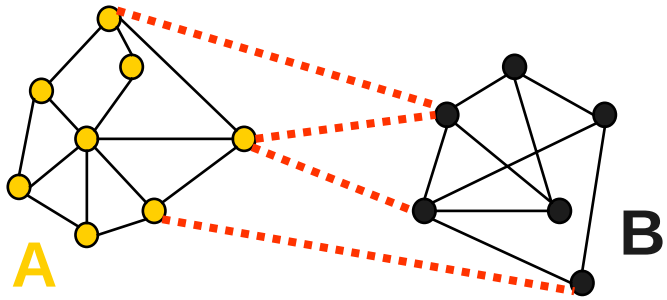
- *Fully-connected* graph
  - node for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - cost $\mathbf{c_{pq}}$ for each link
    - $\mathbf{c_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference in color and position

Segmentation by Graph Cuts



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

Cuts in a graph



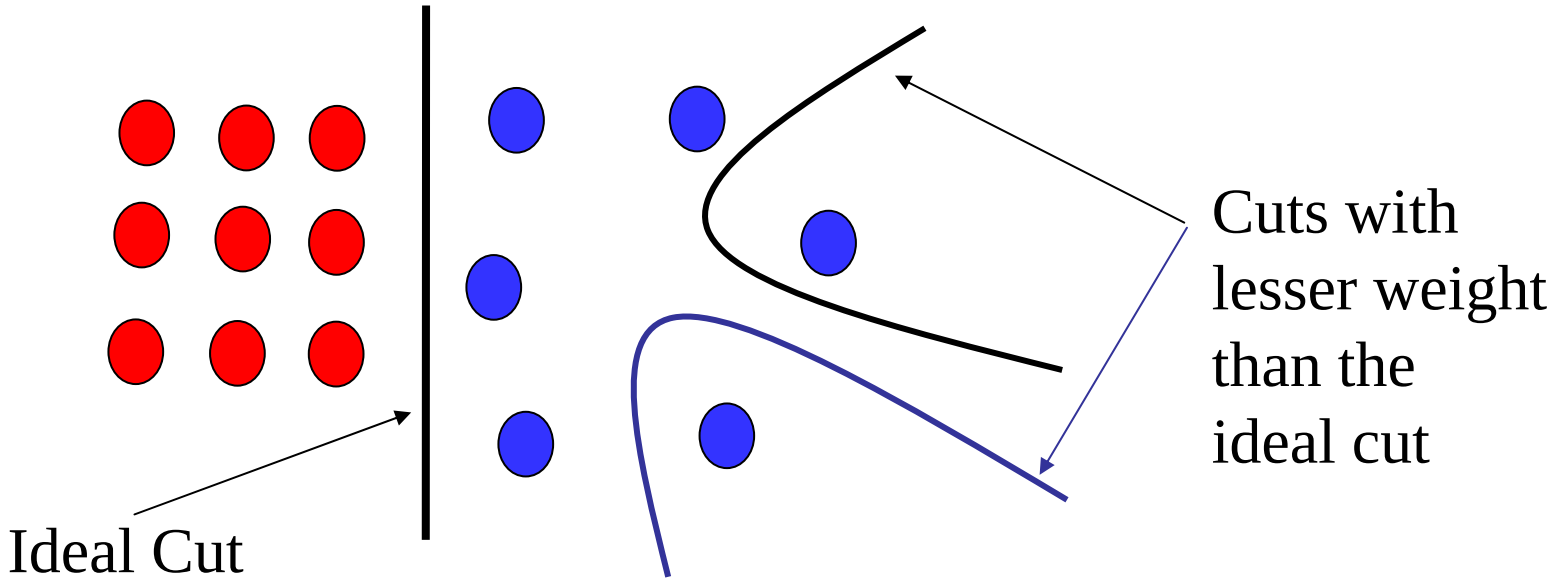
- Link Cut
  - set of links whose removal makes a graph disconnected
  - cost of a cut: $cut(A, B) = \sum_{p \in A, q \in B} c_{p,q}$

Find minimum cut

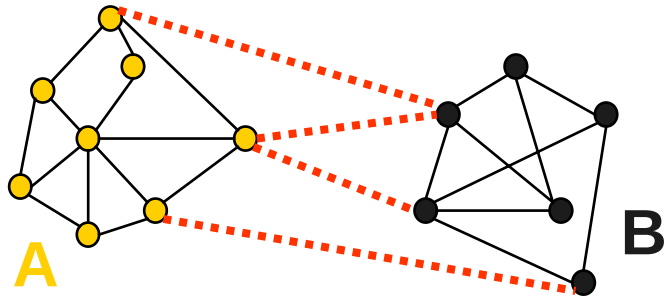
- gives you a segmentation

Drawbacks of Minimum Cut

- Weight of cut is directly proportional to the number of edges in the cut.



Cuts in a graph



Normalized Cut

- a cut penalizes large segments
- fix by normalizing for size of segments

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

- volume(A) = sum of costs of all edges that touch A

Normalized Cuts[1]

- Normalized cut is defined as

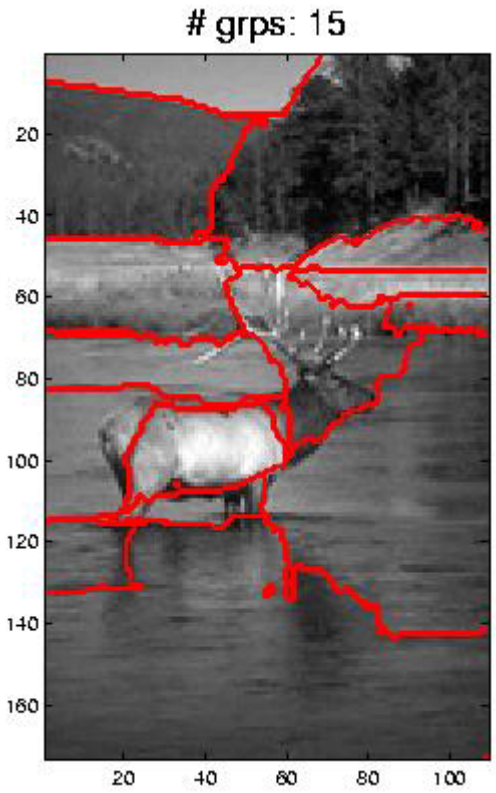
$$N_{cut}(A,B) = \frac{w(\langle A,B\rangle)}{\sum_{x\in A, y\in A} w(x,y)} + \frac{w(\langle A,B\rangle)}{\sum_{z\in B, y\in B} w(z,y)}$$

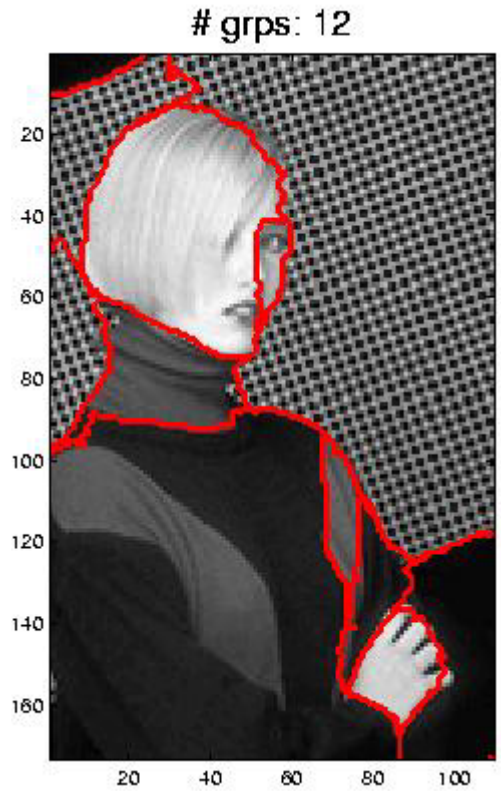
- $N_{cut}(A,B)$ is the measure of dissimilarity of sets A and B.
- Small if
  - Weights between clusters small
  - Weights within a cluster large
- Minimizing $N_{cut}(A,B)$ maximizes a measure of similarity within the sets A and B

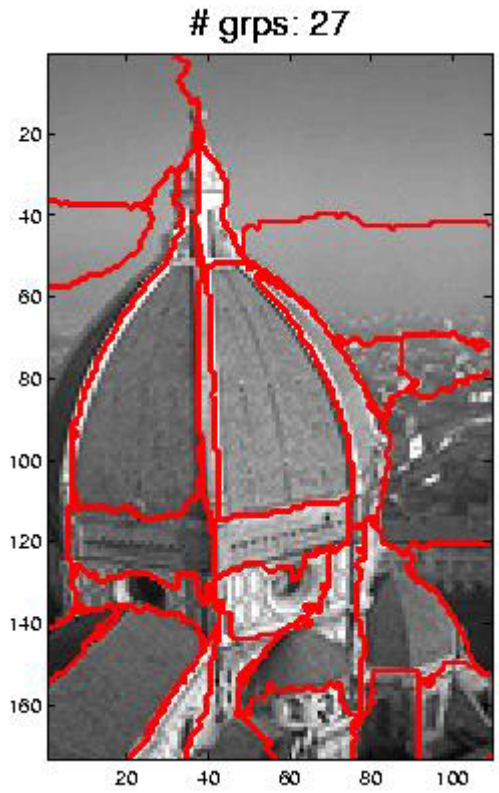
Finding Minimum Normalized-Cut

- Finding the Minimum Normalized-Cut is NP-Hard.
- Polynomial Approximations are generally used for segmentation

Normalized Cuts



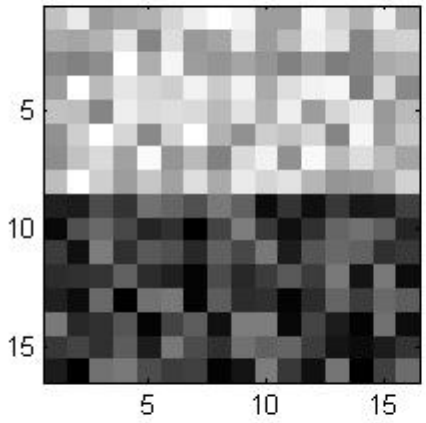


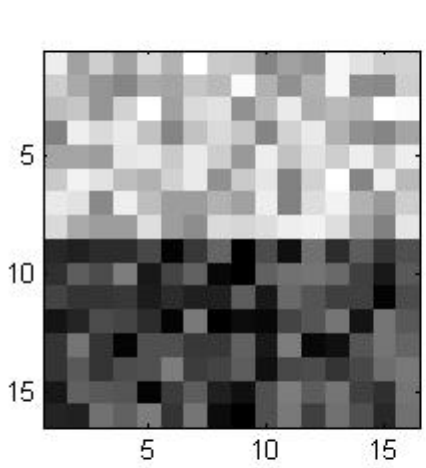


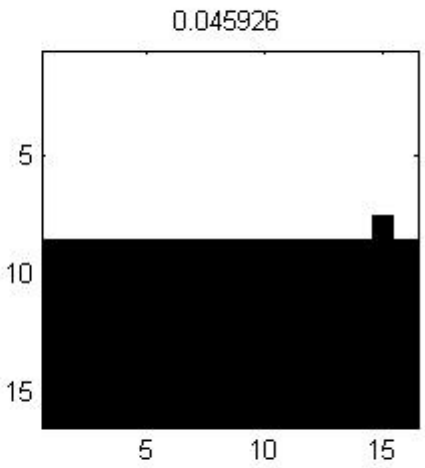
Normalized Cuts

Matlab demo 4—Graph cut

```
dim = 16;
I = rand( dim, dim );                % generate a random image
I( 1:dim/2,:) = I( 1:dim/2,:) + 1;   % brighten the top half
[X,Y] = meshgrid( [1:dim]/dim-0.5, [1:dim]/dim-0.5 );
I = I(:); X = X(:);Y = Y(:);
N = length(I);   % number of pixels
W = zeros( N );
sigD = 0.1; % variance for distance
sigI = 0.1; % variance for intensity
for k = 1 : N
      dist = sqrt( (X(k)-X).^2 + (Y(k)-Y).^2 );
      W(:,k) =  exp( -((I(k)-I).^2)/sigI  ) .* exp( -(dist.^
end
subplot(121); imagesc( reshape(I,dim,dim) ); axis imag

%%% CLUSTER
d = sum(W);D1 = diag(d);D2 = diag(1./sqrt(d));
[vv, dd] = eig(D2 * (D1-W) * D2);
v2 = vv(:,2) > 0;
cc = sum(vect(W(v2==0,v2==1)))/sum(d(v2==0)) +s
v2 = reshape( v2, dim, dim ); % 2 clusters by a 6 bina
subplot(122); imagesc( v2 ); axis image; title( cc );    %
colormap gray;

 %%% VECTORIZE A MATRIX
 function [c] = vect(m)
    c = m(:);
```







Example Results

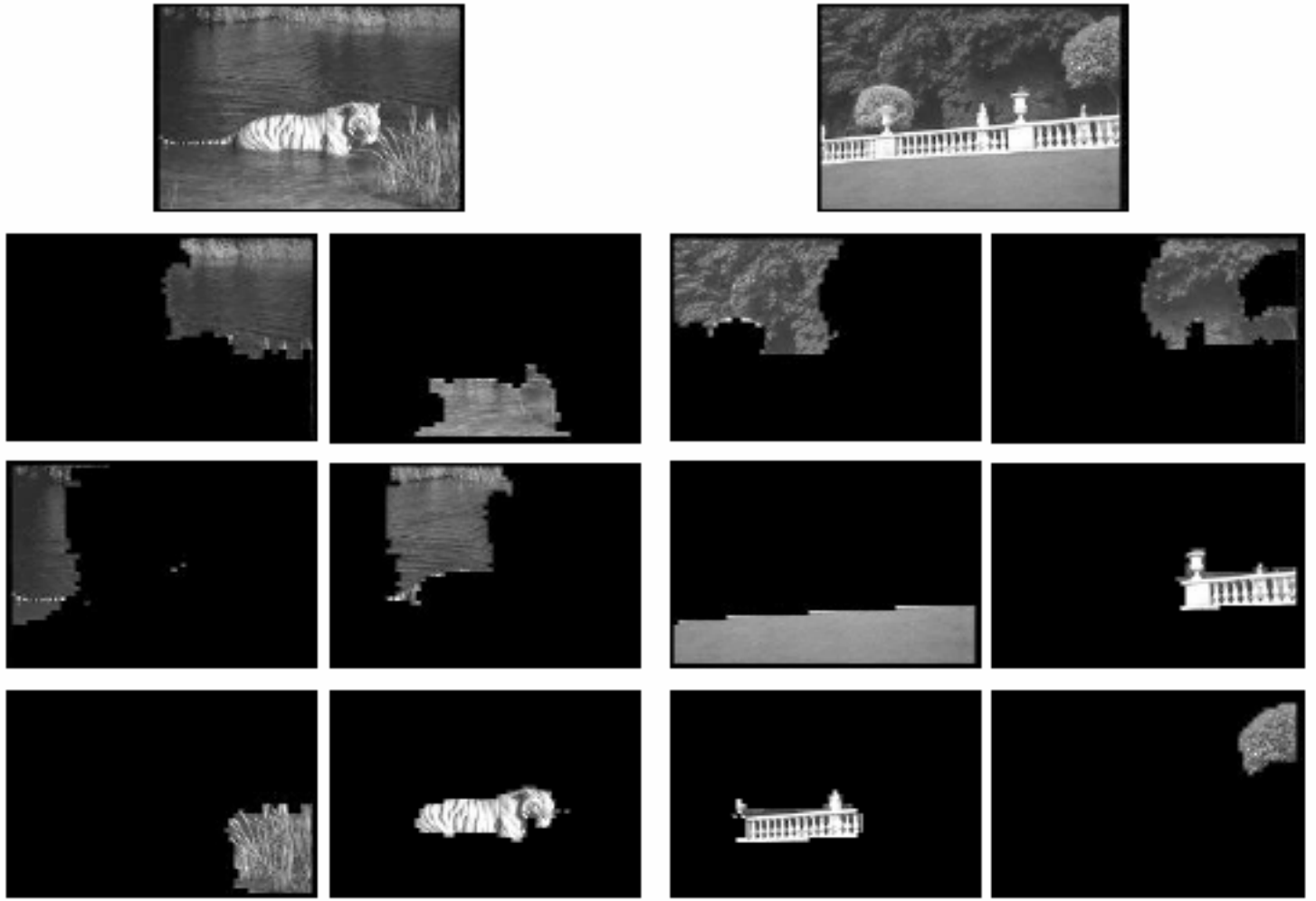


Figure from “Image and video segmentation: the normalised cut framework”, by Shi and Malik, 1998

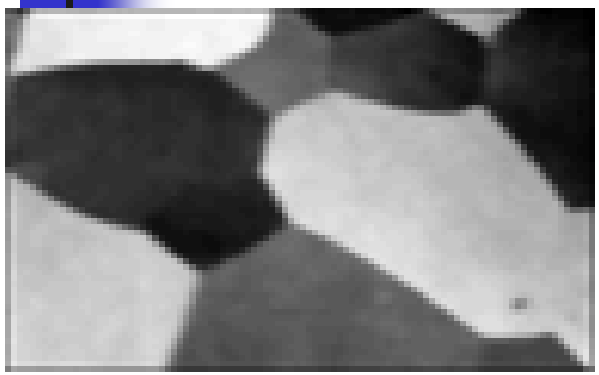
More Results

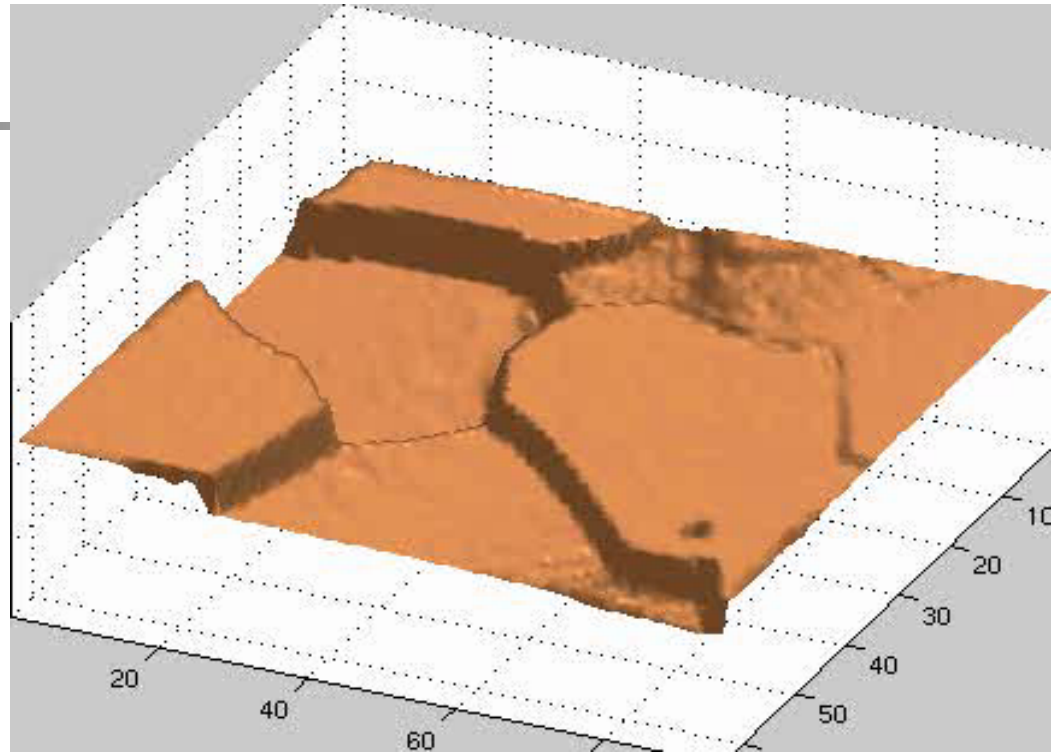
Figure from “Normalized cuts and image segmentation,” Shi and Malik, 2000

Drawbacks of Minimum Normalized Cut

- Huge Storage Requirement and time complexity
- Bias towards partitioning into equal segments
- Have problems with textured backgrounds

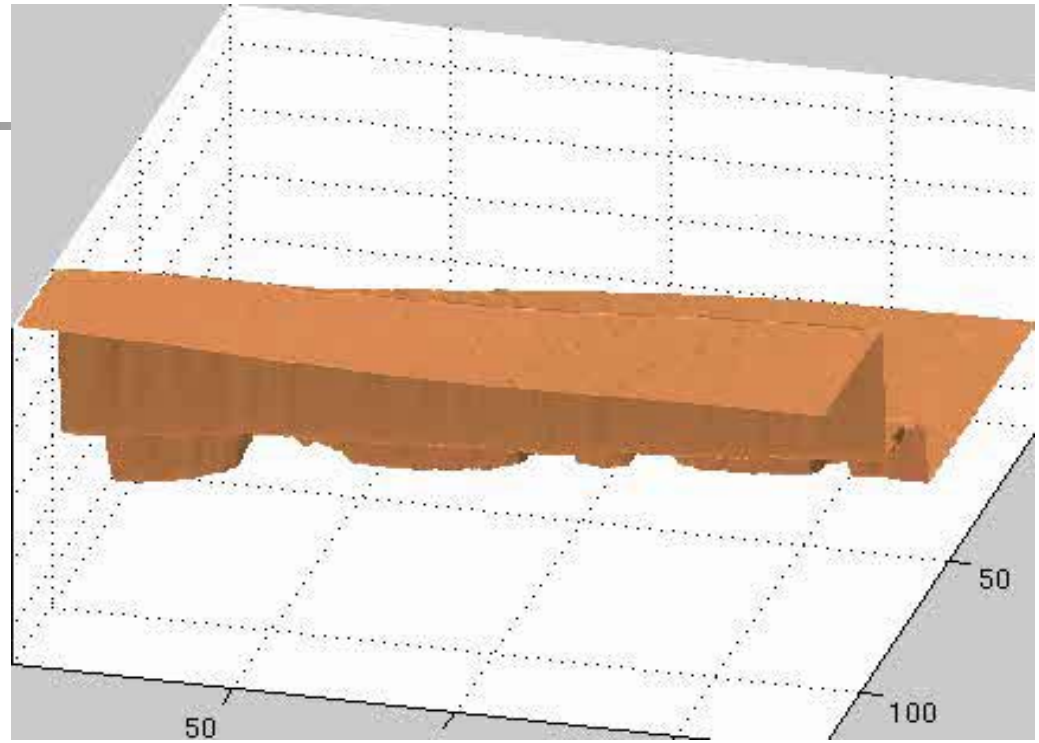
Interpretation as a Dynamical System





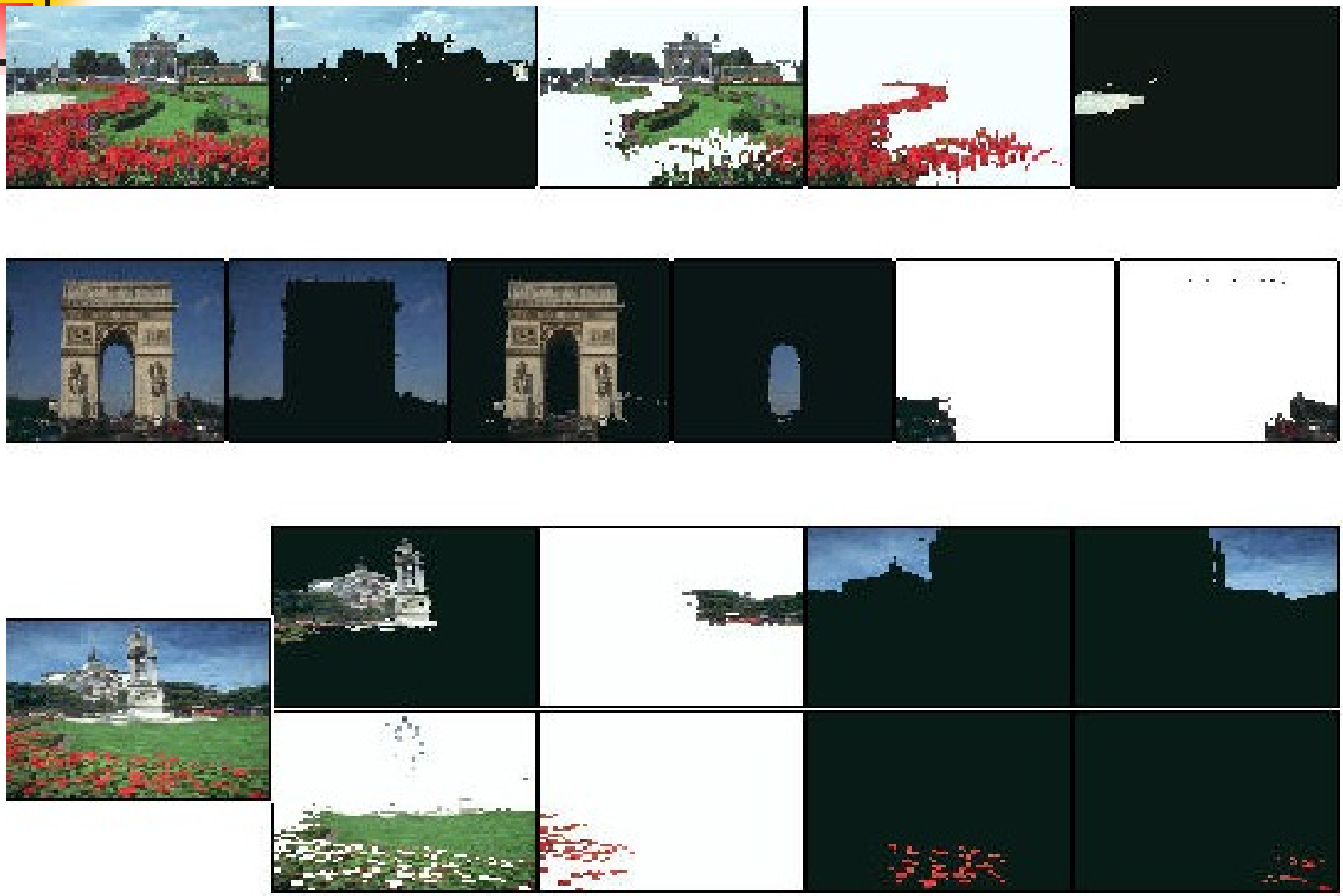
- Treat the links as springs and shake the system
  - elasticity proportional to cost
  - vibration “modes” correspond to segments
    - can compute these by solving an eigenvector problem
    - http://www.cis.upenn.edu/~jshi/papers/pami_ncut.pdf

Interpretation as a Dynamical System

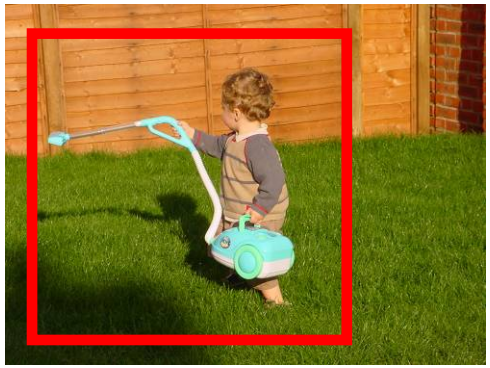


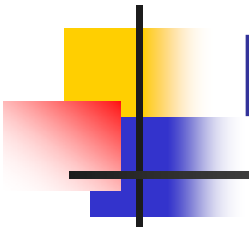
- Treat the links as springs and shake the system
  - elasticity proportional to cost
  - vibration “modes” correspond to segments
    - can compute these by solving an eigenvector problem
    - http://www.cis.upenn.edu/~jshi/papers/pami_ncut.pdf

Color Image Segmentation



Grabcut [Rother et al., SIGGRAPH 2004]

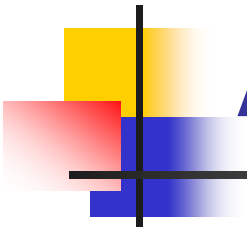




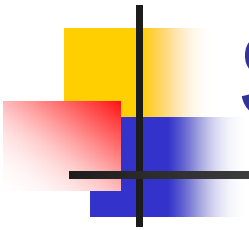
References

- Shi and Malik, "Normalized Cuts and Image Segmentation," Proc. CVPR 1997.
- Comaniciu and Meer, "Mean shift analysis and applications," Proc. *ICCV* 1999.

- Mortensen and Barrett, "Intelligent Scissors for Image Composition," Proc. *SIGGRAPH* 1995.
- Boykov and Jolly, "Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images," Proc. ICCV, 2001.



Applications



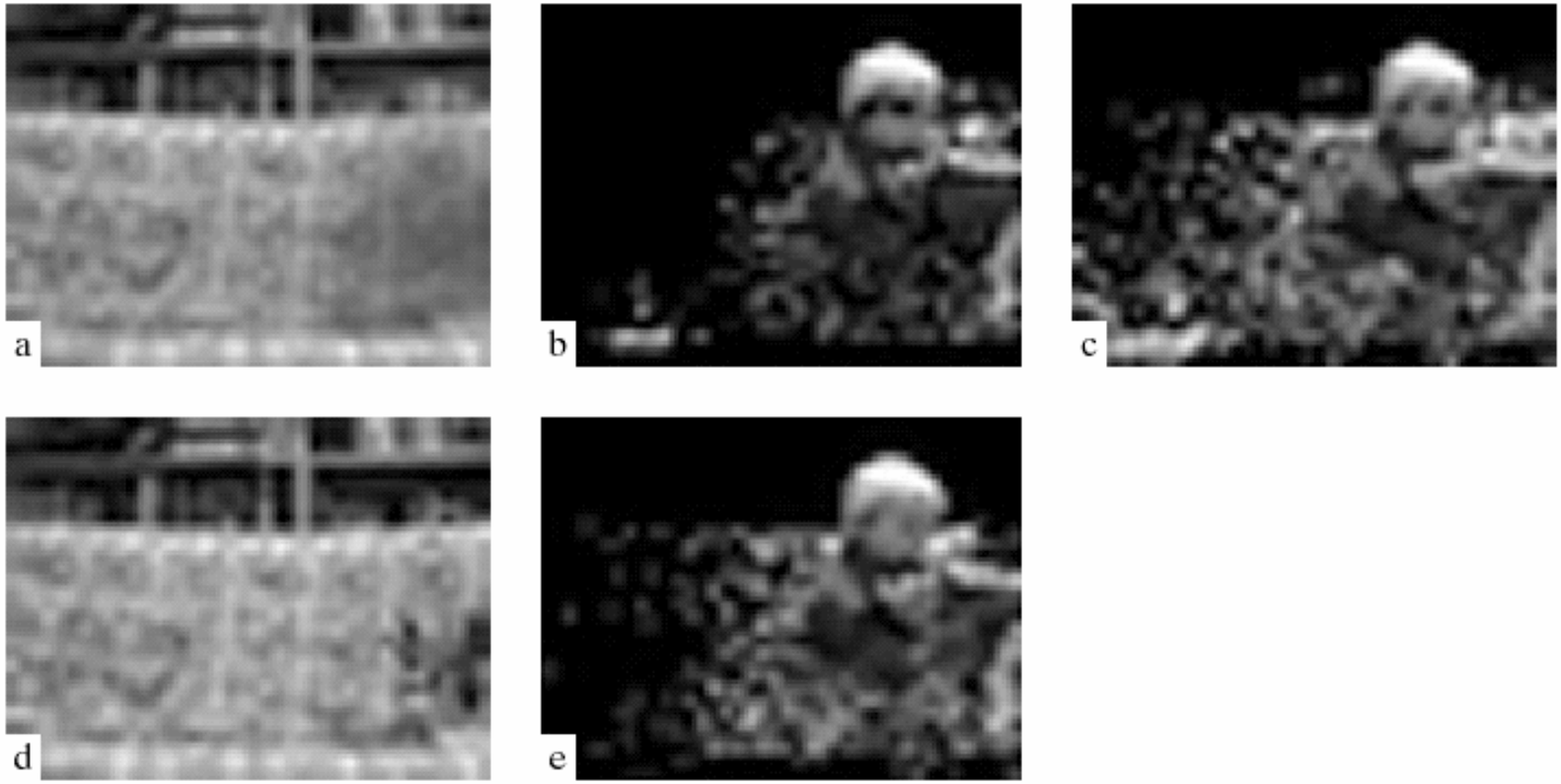
Shot Boundary Detection

- Find the shots in a sequence of video
  - shot boundaries usually result in big differences between succeeding frames
- Strategy:
  - compute inter-frame distances
  - declare a boundary where these are big
- Possible distances
  - frame differences
  - histogram differences
  - block comparisons
  - edge differences
- Applications:
  - representation for movies, or video sequences
  - Support search

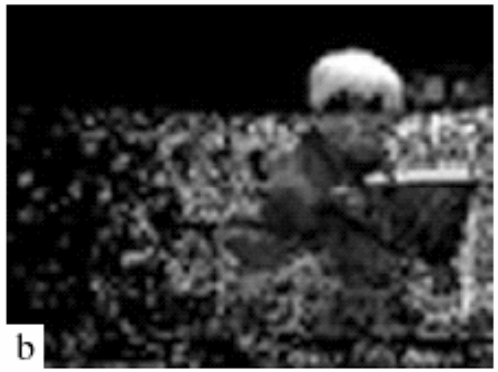
Background Subtraction

- 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


a
b
c
d
e


a


b


c


d


e

Classic Background Subtraction model

- Background is assumed to be mostly static
- Each pixel is modeled as by a gaussian distribution in YUV space
- Model mean is usually updated using a recursive low-pass filter

Given new image, generate silhouette
by marking those pixels that are significantly
different from the “background” value.

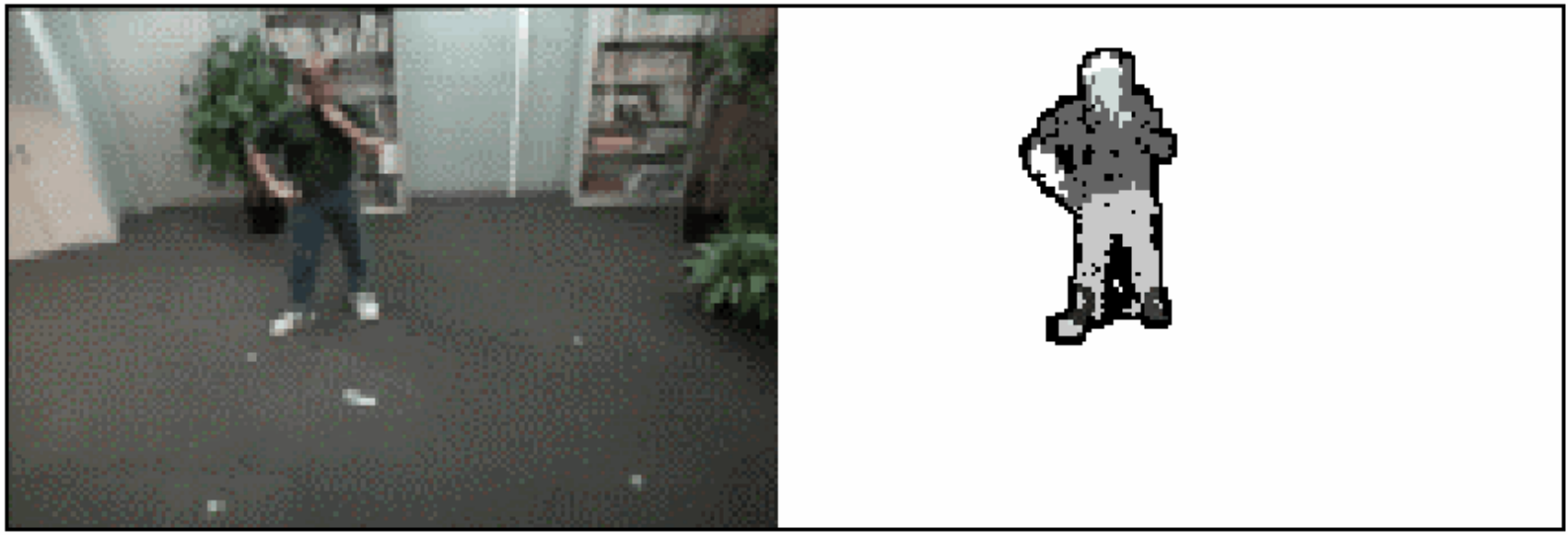
Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

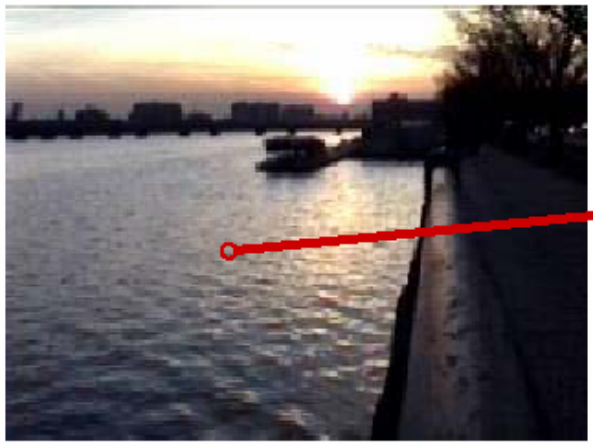
Static Background Modeling Examples

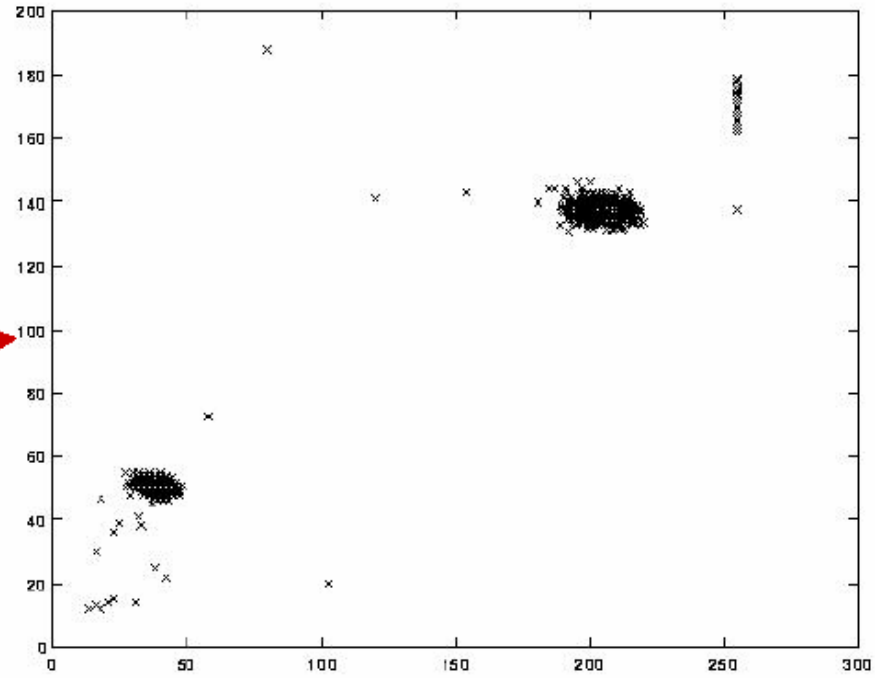


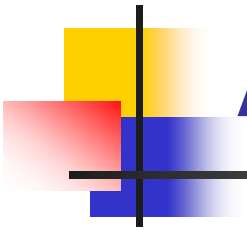
[MIT Media Lab Pfinder / ALIVE System]

Dynamic Background

BG Pixel distribution is non-stationary:



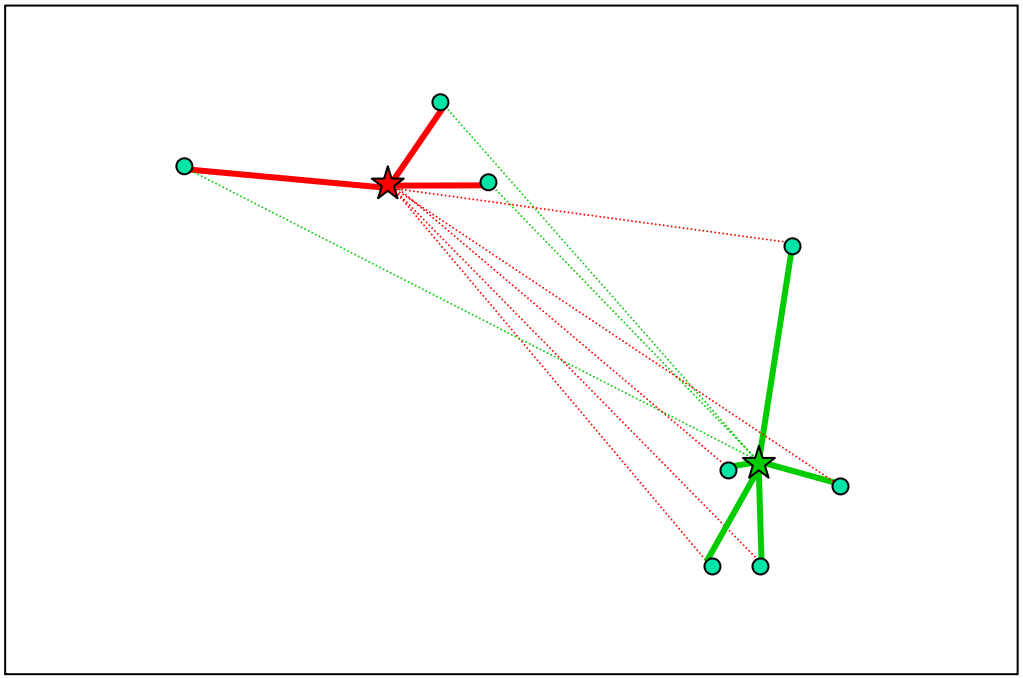




Appendix

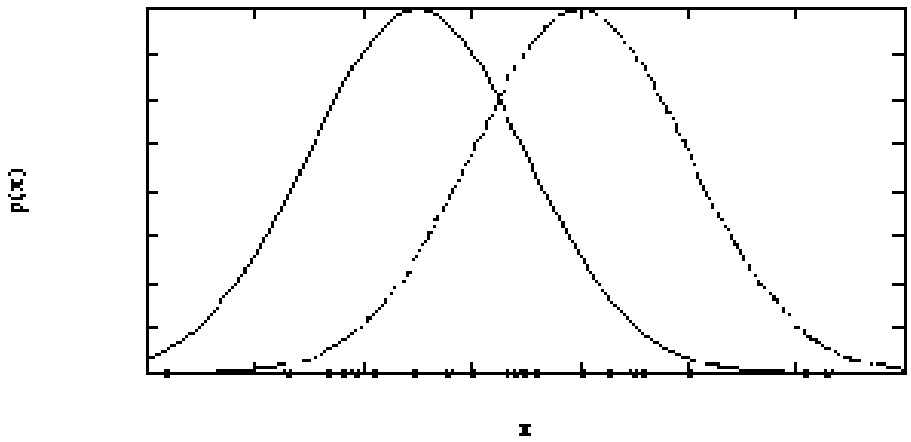
- Expectation-Maximization

Generalized K-Means (EM)



Idea

- Data generated from mixture of Gaussians



- Latent variables: Correspondence between Data Items and Gaussians

Learning a Gaussian Mixture
(with known covariance)

E-Step

$$E[z_{ij}] = \frac{p(x = x_i \mid \mu = \mu_j)}{\sum_{n=1}^{k} p(x = x_i \mid \mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

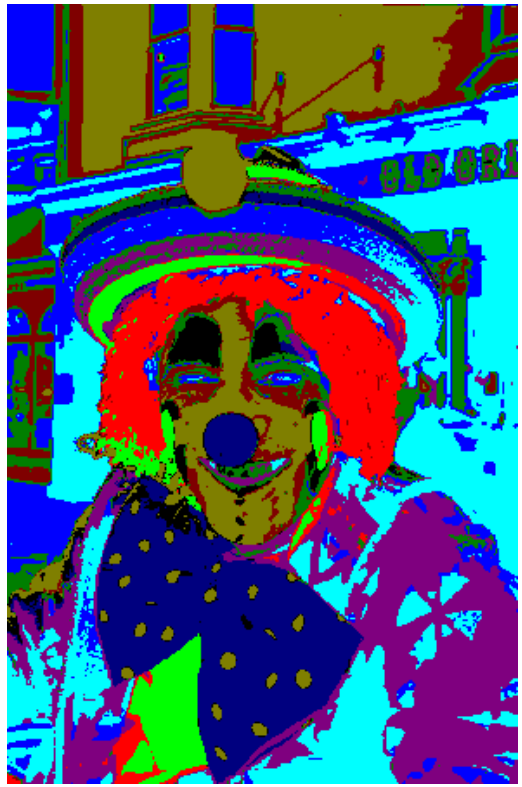
M-Step

$$\mu_j \leftarrow \frac{1}{m} \sum_{i=1}^{m} E[z_{ij}]\, x_i$$

Generalized K-Means

- Converges!
- Proof [Neal/Hinton, McLachlan/Krishnan]:
  - E/M step does not decrease data likelihood
  - Converges at saddle point

EM Clustering: Results

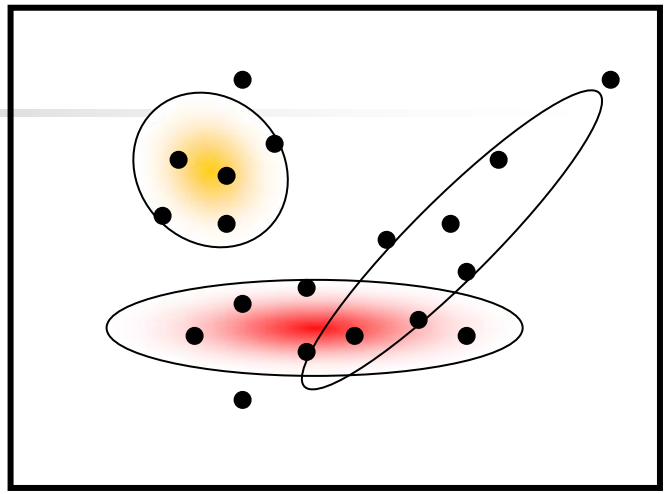


Probabilistic clustering

- Basic questions
  - what's the probability that a point **x** is in cluster m?
  - what's 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 θ

Mixture of Gaussians

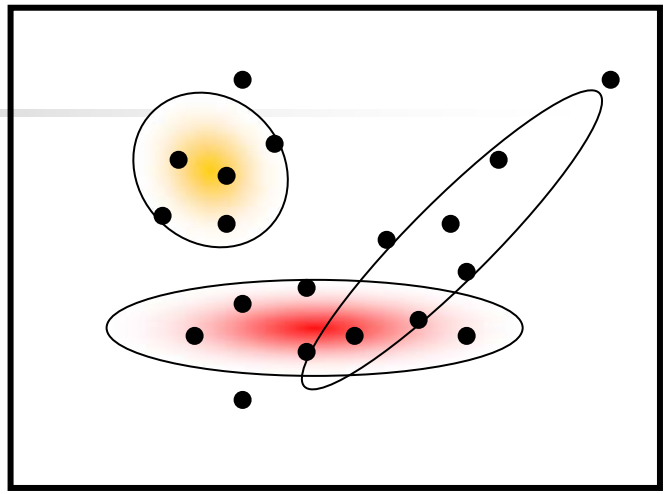


- One generative model is a mixture of Gaussians (MOG)
  - K Gaussian blobs with means μ_b covariance matrices $\mathbf{V_b}$, dimension d
    - blob *b* defined by: $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - blob *b* is selected with probability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

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b)$$

  - where $\theta = [\mu_1, \ldots, \mu_n, V_1, \ldots, V_n]$

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 i, 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
  - compute probability that blob b is selected

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

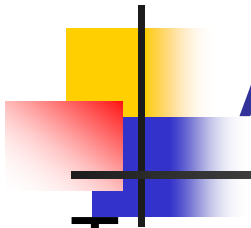
N data points

  - mean of blob b

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

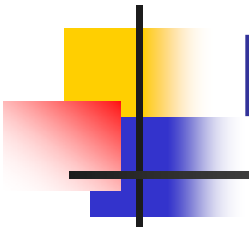
  - covariance of blob b

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



Applications of EM

- Turns out this is useful for all sorts of problems
  - any clustering problem
  - any model estimation problem
  - missing data problems
  - finding outliers
  - segmentation problems
    - segmentation based on color
    - segmentation based on motion
    - foreground/background separation
  - …



Problems with EM

- Local minima
- Need to know number of segments
- Need to choose generative model

Normalized Cuts[1]

- Normalized cut is defined as

$$N_{cut}(A,B) = \frac{w(\langle A,B\rangle)}{\sum_{x\in A, y\in A} w(x,y)} + \frac{w(\langle A,B\rangle)}{\sum_{z\in B, y\in B} w(z,y)}$$

- $N_{cut}(A,B)$ is the measure of dissimilarity of sets A and B.
- Small if
  - Weights between clusters small
  - Weights within a cluster large
- Minimizing $N_{cut}(A,B)$ maximizes a measure of similarity within the sets A and B

[1]J. Shi and J. Malik, "Normalized Cuts & Image Segmentation," IEEE Trans. of PAMI, Aug 2000.

Finding Minimum Normalized-Cut

- Finding the Minimum Normalized-Cut is NP-Hard.
- Polynomial Approximations are generally used for segmentation

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Finding Minimum Normalized-Cut

$W = N \times N$ symmetric matrix, where

$$W(i,j) = \begin{cases} e^{-\|F_i - F_j\|/\sigma_F^2} \times e^{-\|X_i - X_j\|/\sigma_X^2} & \text{if } j \in N(i) \\ 0 & \text{otherwise} \end{cases}$$

$\|F_i - F_j\|$ = Image feature similarity

$\|X_i - X_j\|$ = Spatial Proximity

$D = N \times N$ diagonal matrix, where $D(i,i) = \sum_j W(i,j)$

Finding Minimum Normalized-Cut

- It can be shown that

$$\min N_{cut} = \min_{\mathbf{y}} \frac{\mathbf{y}^{\mathbf{T}}(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^{\mathbf{T}}\mathbf{D}\mathbf{y}}$$

such that $y(i) \in \{1, -b\}, 0 < b \leq 1, \text{and } \mathbf{y}^T \mathbf{D1} = 0$

- If *y* is allowed to take real values then the minimization can be done by solving the generalized eigenvalue system

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

See: Forsyth Chapters in segmentation (pages 323-326)

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Algorithm

- Compute matrices W & D
- Solve $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$ for eigen vectors with the smallest eigen values
- Use the eigen vector with second smallest eigen value to bipartition the graph
- Recursively partition the segmented parts if necessary.

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003