Digital Image Processing
Segmentation

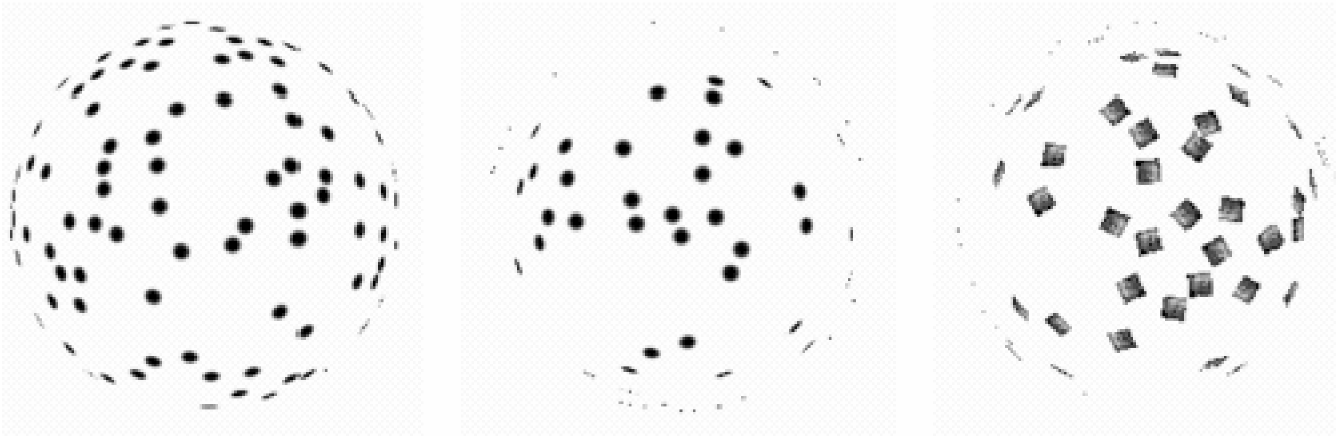
Instructor: Chen Yisong
HCI & Multimedia Lab, Peking University

Outline

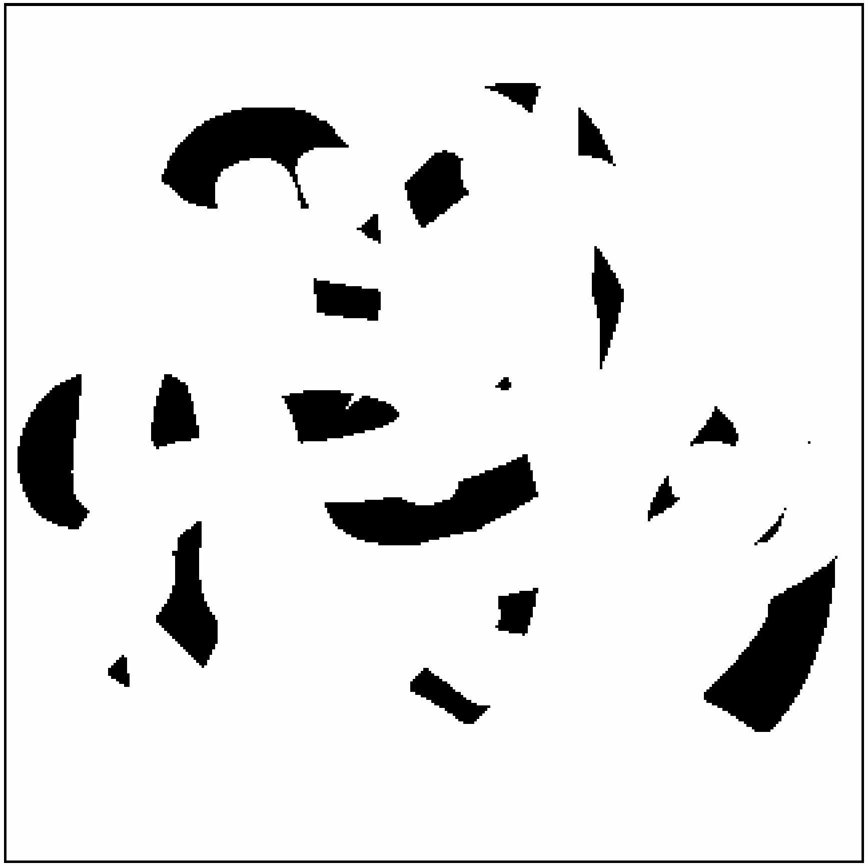
- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph

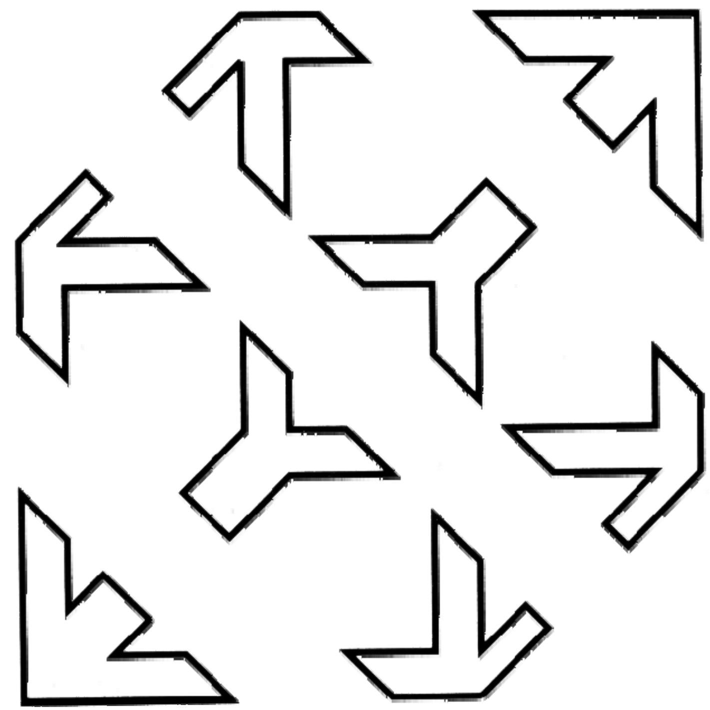
* Pictures from Mean Shift: A Robust Approach toward Feature Space Analysis, by D. Comaniciu and P. Meer http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.htm

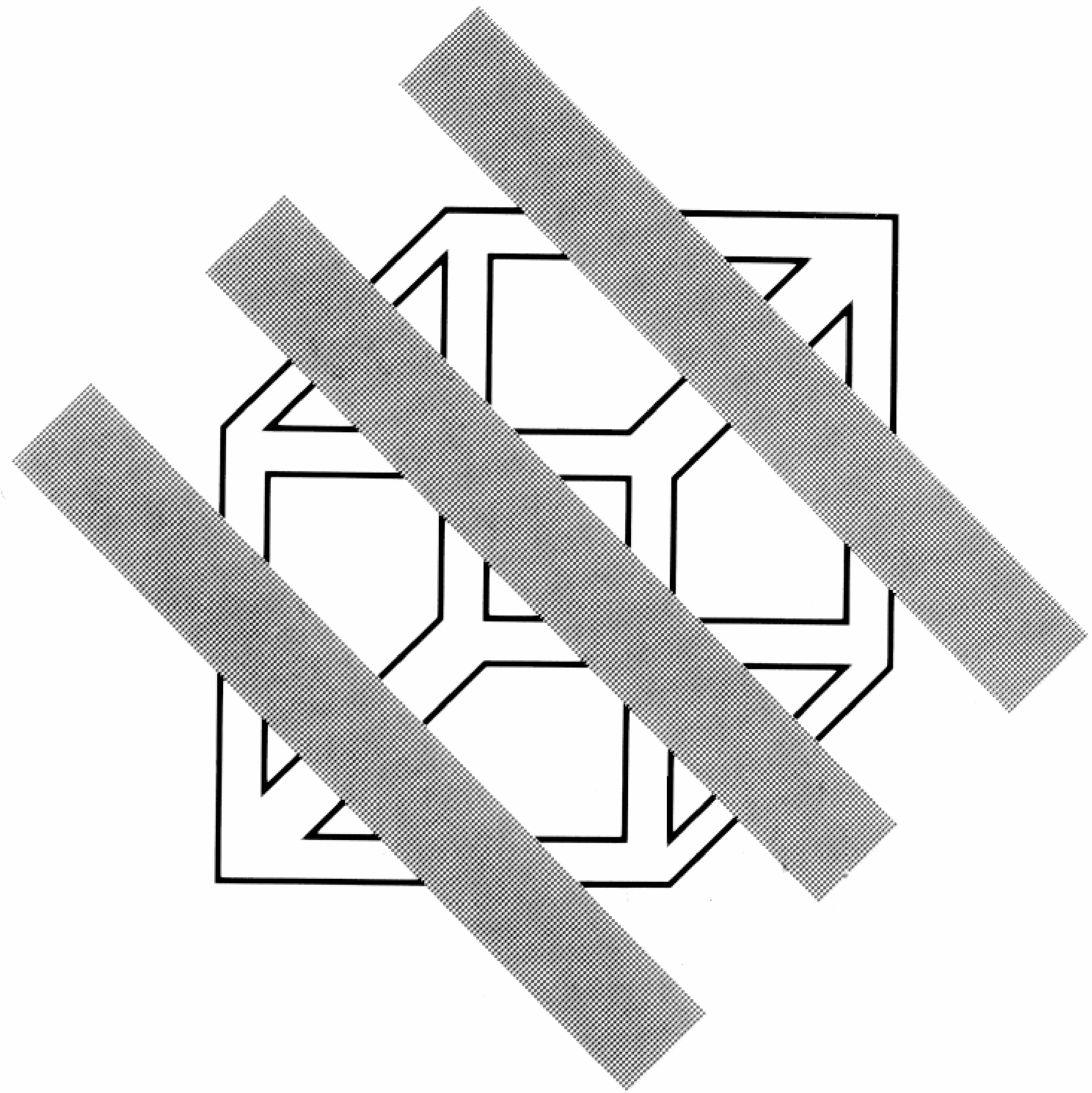
When the 3D nature of grouping is apparent:

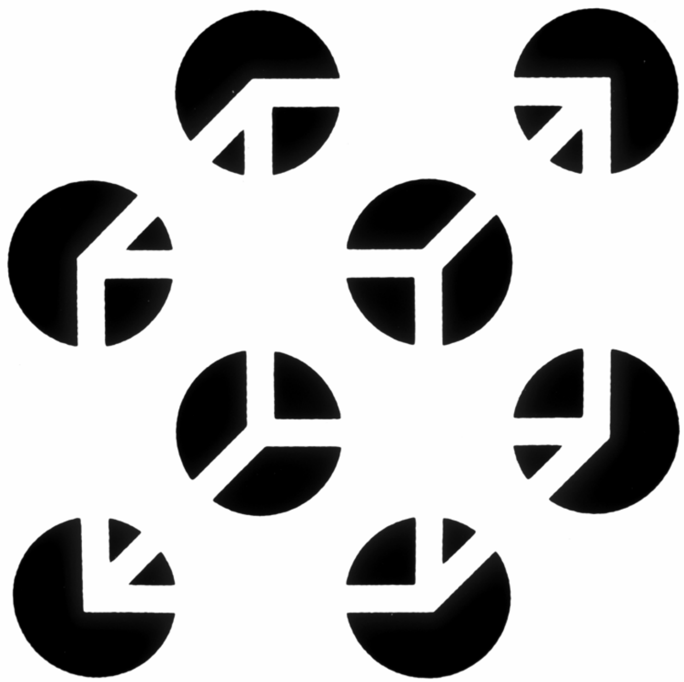


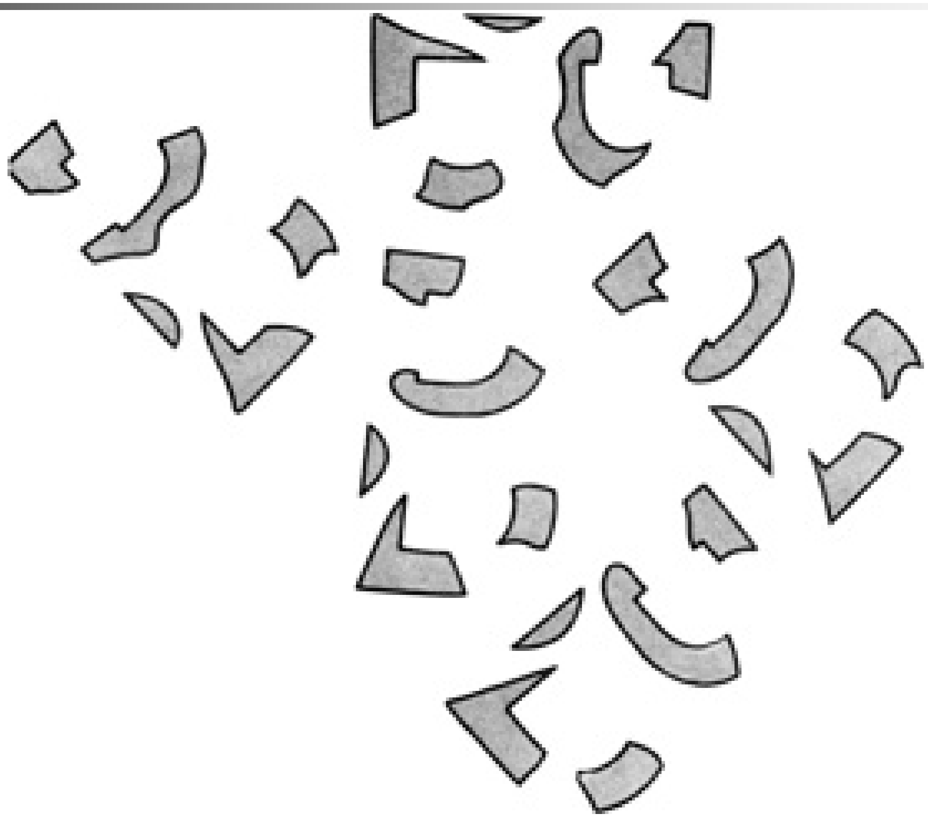
Why do these tokens belong together?











Gestalt psychology (完形心理学)

- 格式塔学派（德语：Gestalt theorie）是心理学重要流派之一，兴起于20世纪初的德国，又称为完形心理学[1]。由马科斯·韦特墨（1880~1943）、沃尔夫冈·苛勒（1887~1967）和科特·考夫卡（1886~1941）三位德国心理学家在研究似动现象的基础上创立。格式塔是德文Gestalt的译音，意即“模式、形状、形式”等，意思是指“动态的整体（dynamic wholes）。
- 格式塔学派主张人脑的运作原理是整体的，“整体不同于其部件的总和”。例如，我们对一朵花的感知，并非纯粹单单从对花的形状、颜色、大小等感官资讯而来，还包括我们对花过去的经验和印象，加起来才是我们对一朵花的感知。
- 格式塔体系的关键特征是整体性、具体化、组织性和恒常性。

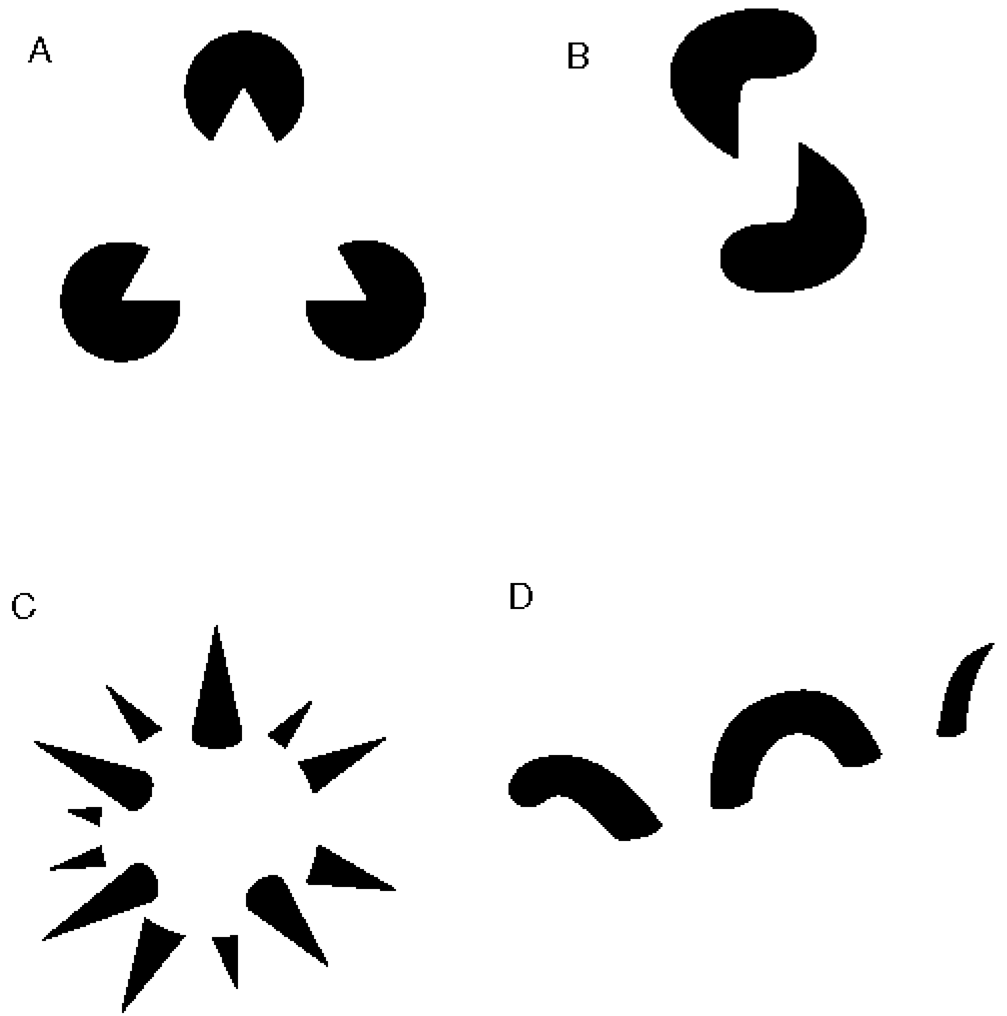
German: *Gestalt* - "essence or shape of an entity's complete form"

Gestalt psychology (完形心理学)

- Emergence(整体性)
  - 整体性(Emergence)的论据可见于“狗图片”的知觉，图片表现一条达尔马提亚狗在树荫下的地面上嗅。对狗的认知并不是首先确定它的各部分（脚、耳朵、鼻子、尾巴等等），并从这些组成部分来推断这是一条狗，而是立刻就将狗作为一个整体来认知。

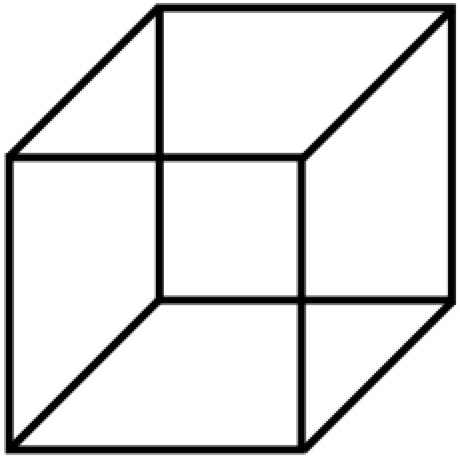
Gestalt psychology (完形心理学)

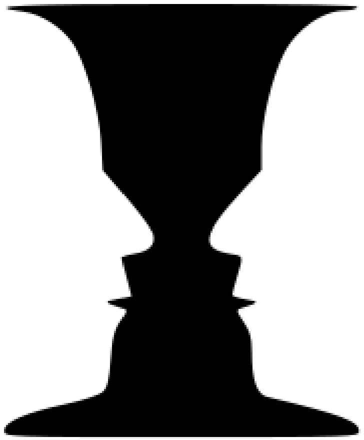
- Reification(具体化)
  - 具体化是知觉的“建设性”的或“生成性的”方面，这种知觉经验，比起其所基于的感觉刺激，包括了更多外在的空间信息。例如，图形A可以被可以知觉为三角形，尽管在事实上并未画三角形。图形C可以被视为三维形体，事实上也没有画三维形体。



Gestalt psychology (完形心理学)

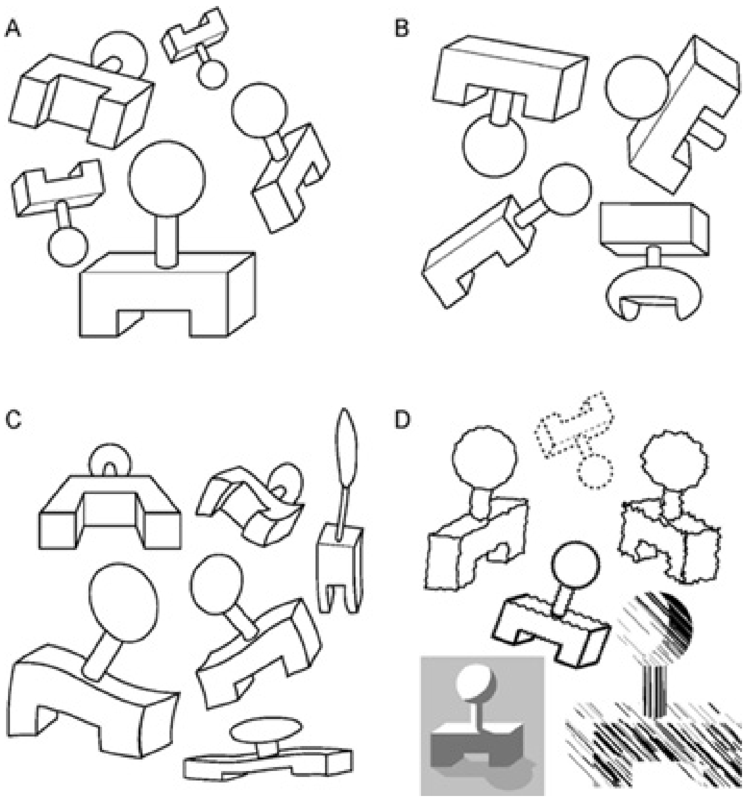
- Multistability (组织性)
  - “组织性”(Multistability)或“组织性知觉”(multistable perception)是趋势模糊知觉经验，不稳定地在两个或两个以上不同解释之间往返。例如下图所示“克尔立方体”和“鲁宾图/花瓶幻觉”。





Gestalt psychology (完形心理学)

- Invariance(恒常性)
  - 恒常性(Invariance) 知觉认可的简单几何组件，形成独立的旋转，平移、大小以及其他一些变化（如弹性变形，不同的灯光和不同的组件功能）。例如图例‘A’在图中都立即确认为相同的基本形式，立即有别于‘B’的形式。在弹性变形的‘C’，描绘时使用不同的图形元素，如‘D’类。 产生“具体化”、“多重稳定性”、“不变性”和“不可分模块单独进行建模”，它们是不同方面的统一机制。

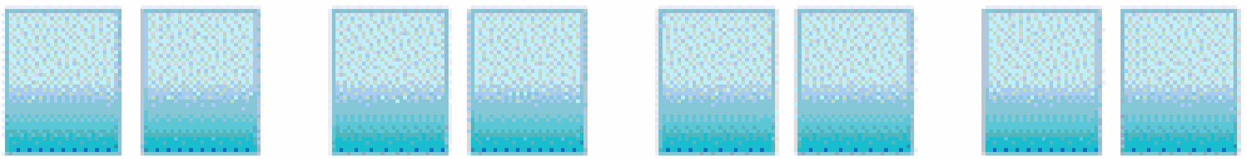


Gestalt Principles (格式塔原理)

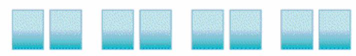
- The fundamental principle of gestalt perception is the law of prägnanz (German for pithiness) which says that we tend to order our experience in a manner that is regular, orderly, symmetric, and simple. Gestalt psychologists attempt to discover refinements of the law of prägnanz, and this involves writing down laws which hypothetically allow us to predict the interpretation of sensation, what are often called "gestalt laws".[1] These include:
  - Law of Closure — The mind may experience elements it does not perceive through sensation, in order to complete a regular figure (that is, to increase regularity).
  - Law of Similarity — The mind groups similar elements into collective entities or totalities. This similarity might depend on relationships of form, color, size, or brightness.
  - Law of Proximity — Spatial or temporal proximity of elements may induce the mind to perceive a collective or totality.
  - Law of Symmetry (Figure ground relationships)— Symmetrical images are perceived collectively, even in spite of distance.
  - Law of Continuity — The mind continues visual, auditory, and kinetic patterns.
  - Law of Common Fate — Elements with the same moving direction are perceived as a collective or unit.

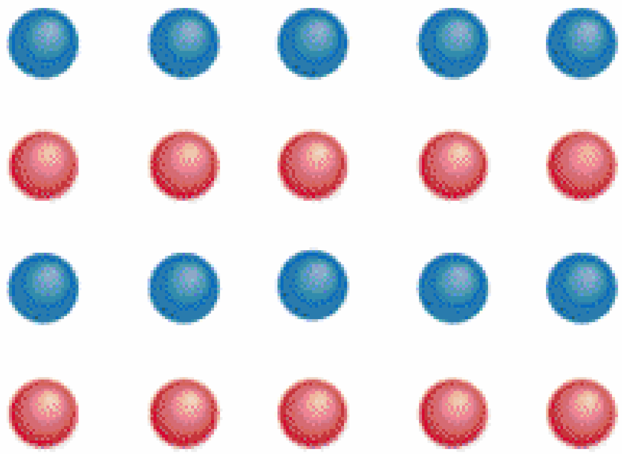
Gestalt Principles (格式塔原理)

- Proximity(接近律)

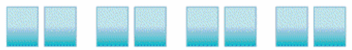
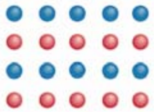


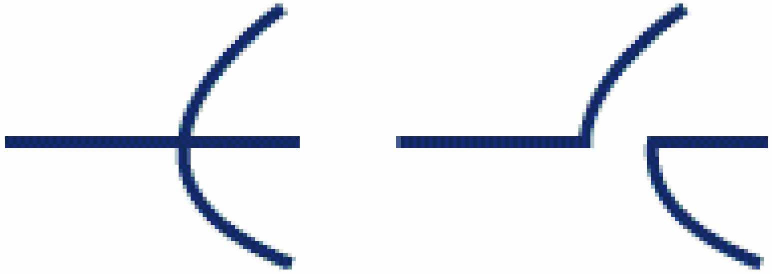
Gestalt Principles

- Proximity 
- Similarity (相似律)

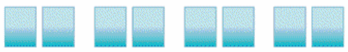
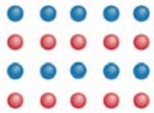


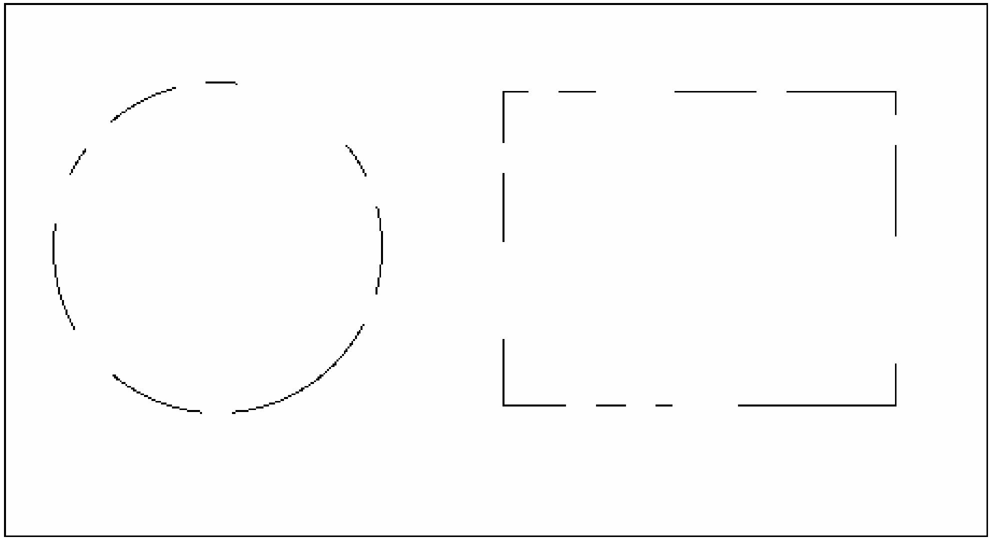
Gestalt Principles

- Proximity 
- Similarity 
- Continuity (连续律)

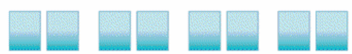
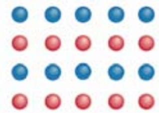


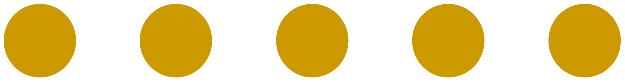
Gestalt Principles

- Proximity 
- Similarity 
- Continuity 
- Closure (闭合律)

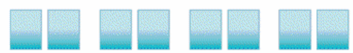
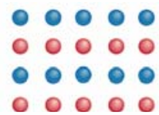
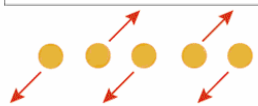


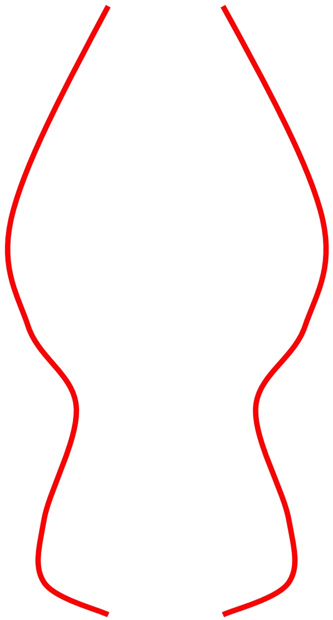
Gestalt Principles

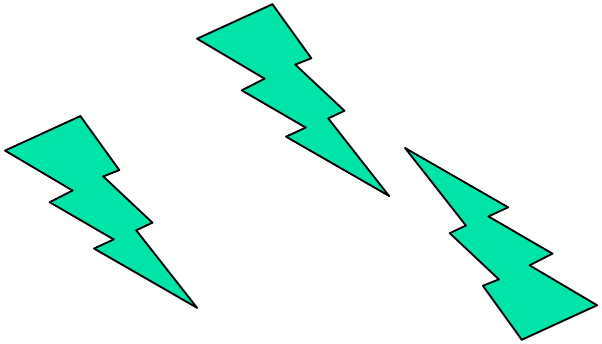
- Proximity 
- Similarity 
- Continuity 
- Closure 
- Common Fate (同步律)



Gestalt Principles

- Proximity 
- Similarity 
- Continuity 
- Closure 
- Common Fate 
- Symmetry (对称律)





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

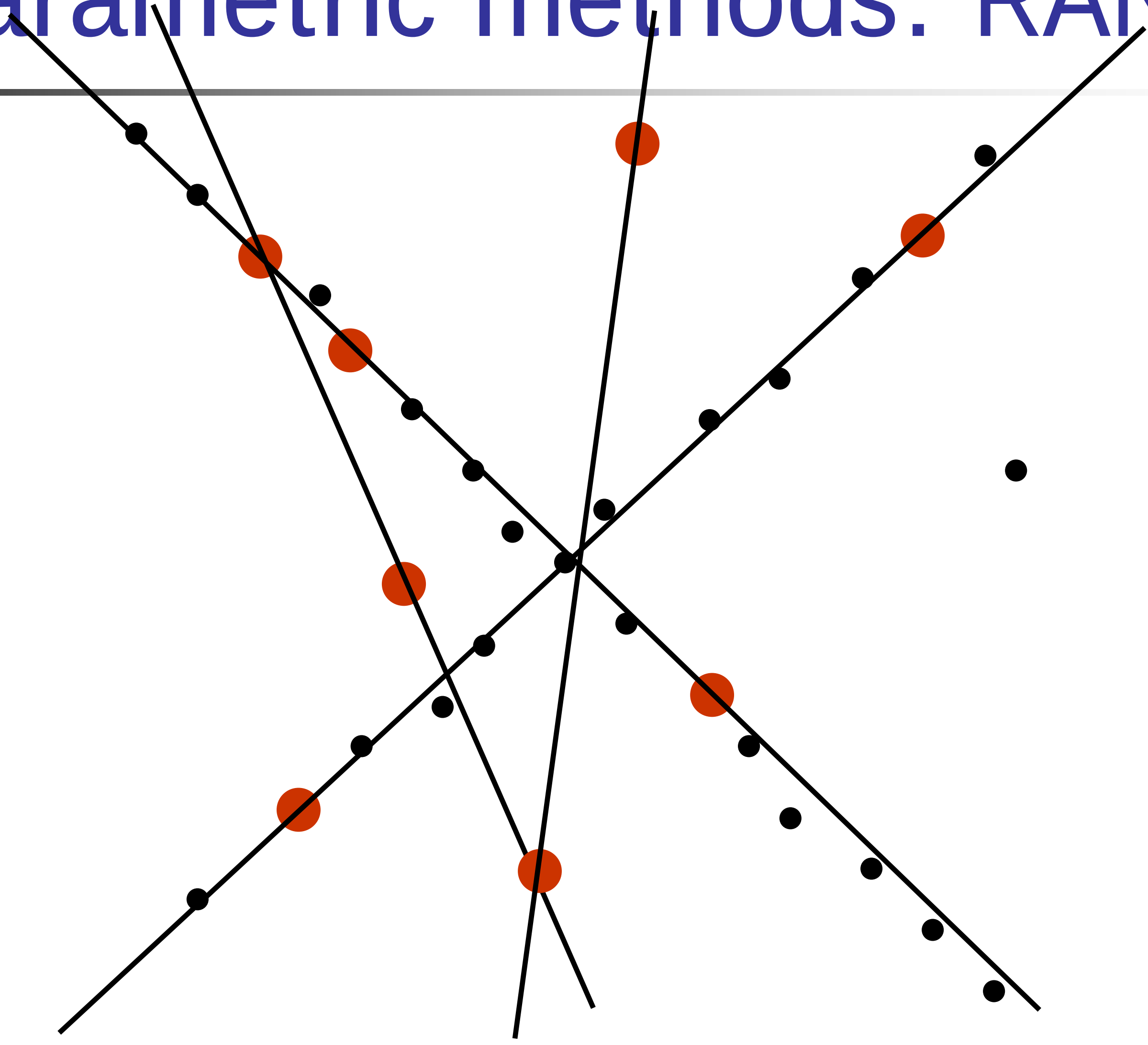
- The things that are grouped (pixels, points, surface elements, etc., etc.)

- 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

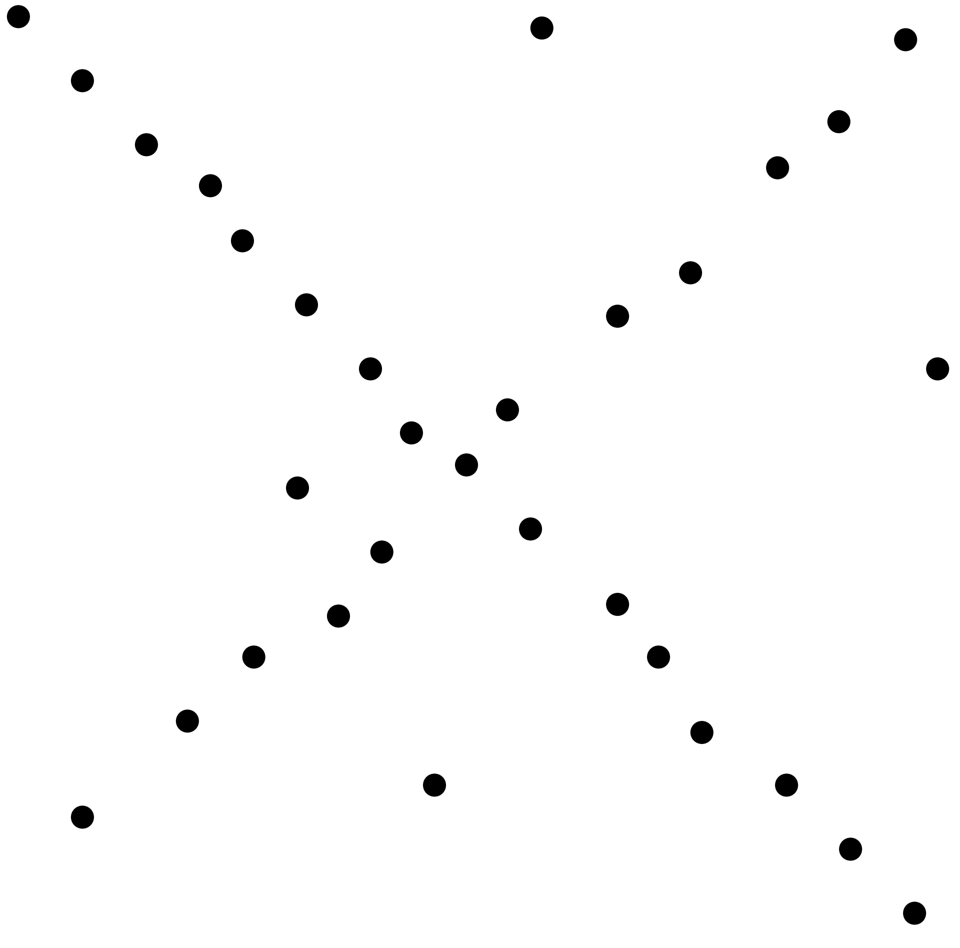
General Ideas

- 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?

Parametric methods: RANSAC



Hough Transform



Hough Transform

- Linear in the number of points
- Describe lines as

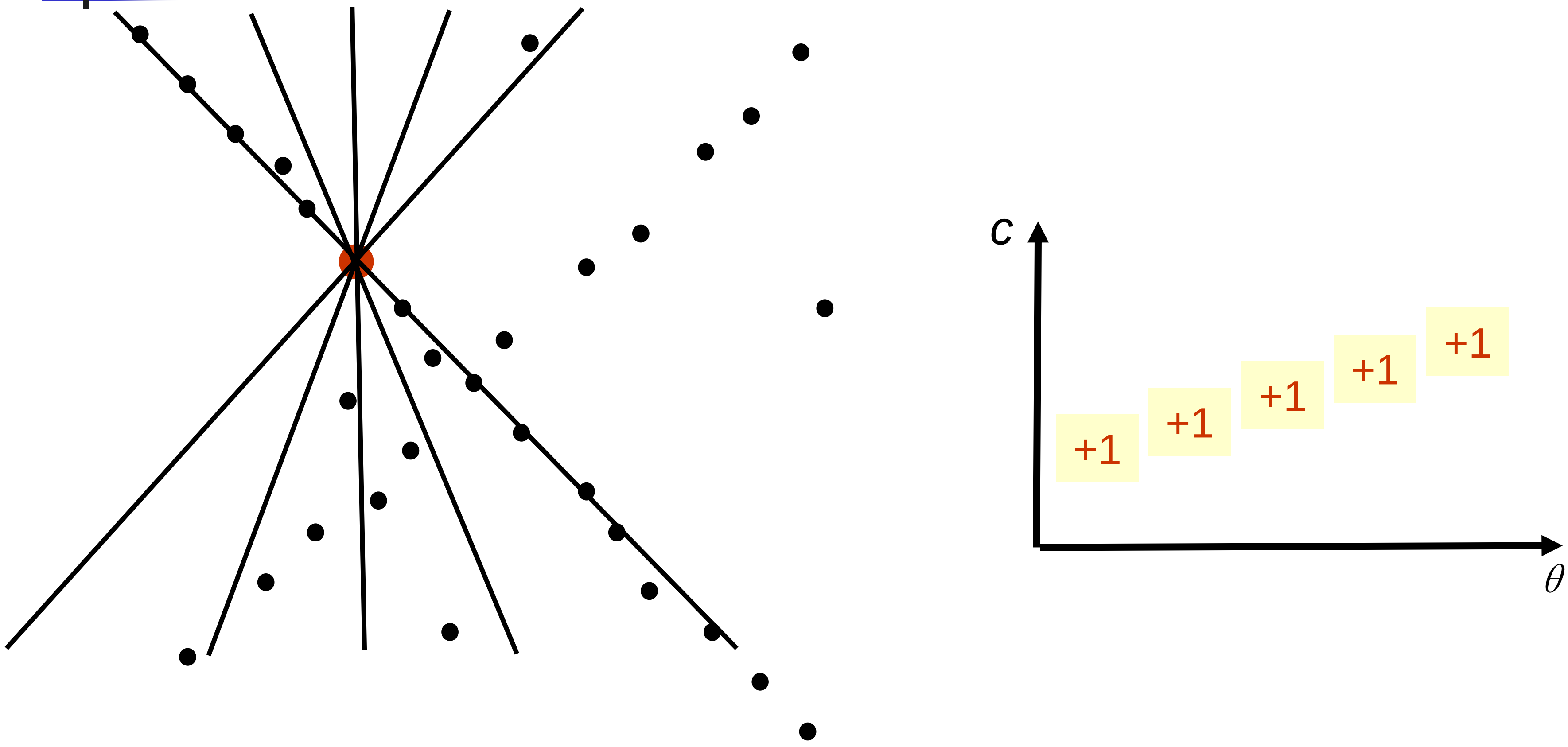
$$y = mx + n$$

- Or better

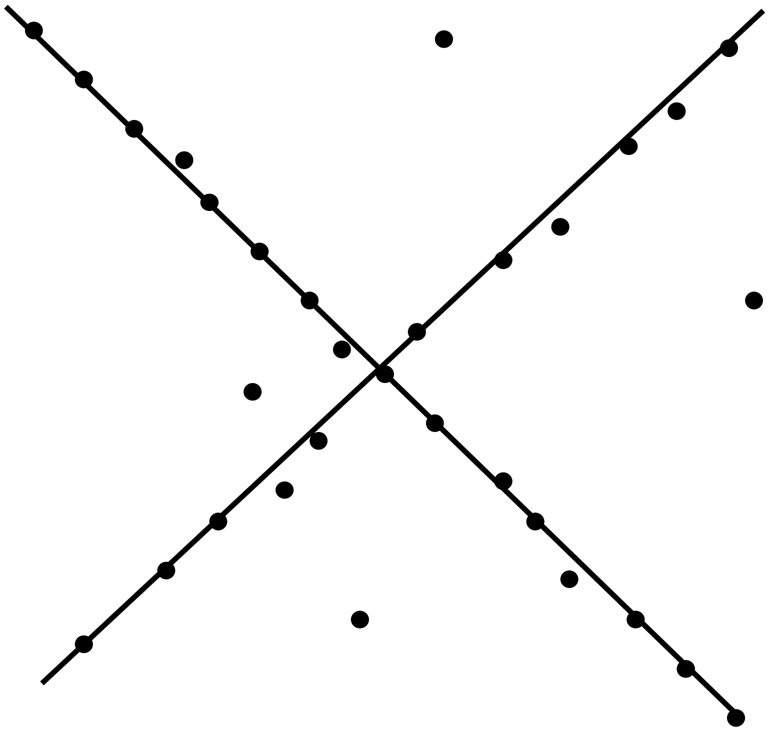
$$x\cos\theta + y\sin\theta = c$$

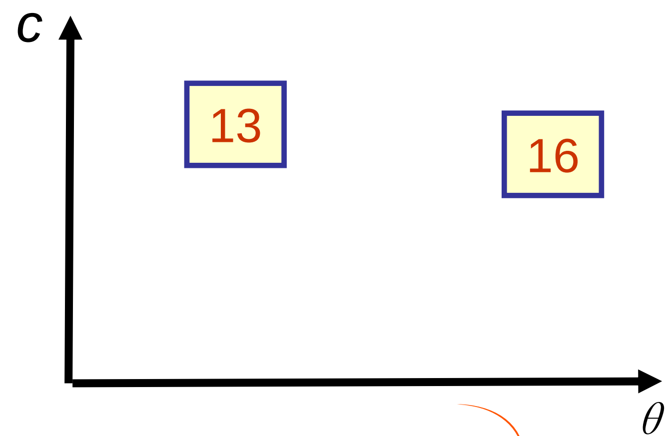
- Prepare a 2D table

Hough Transform



Hough Transform

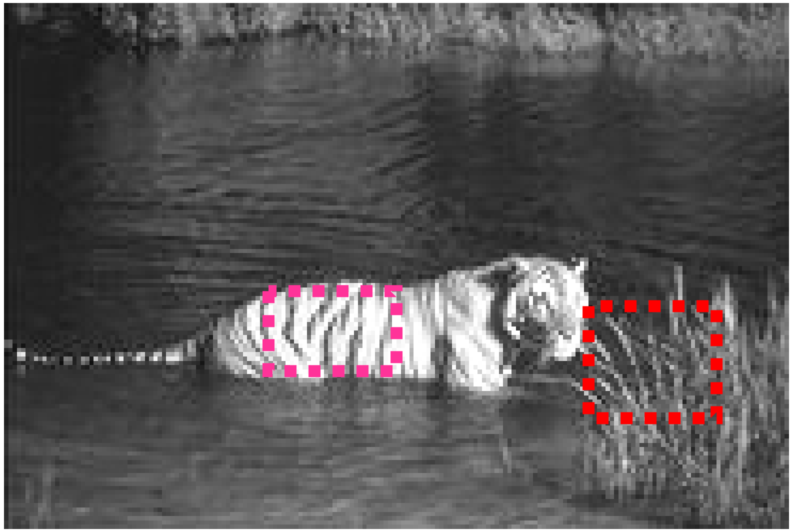


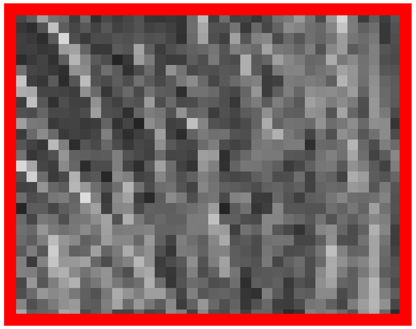


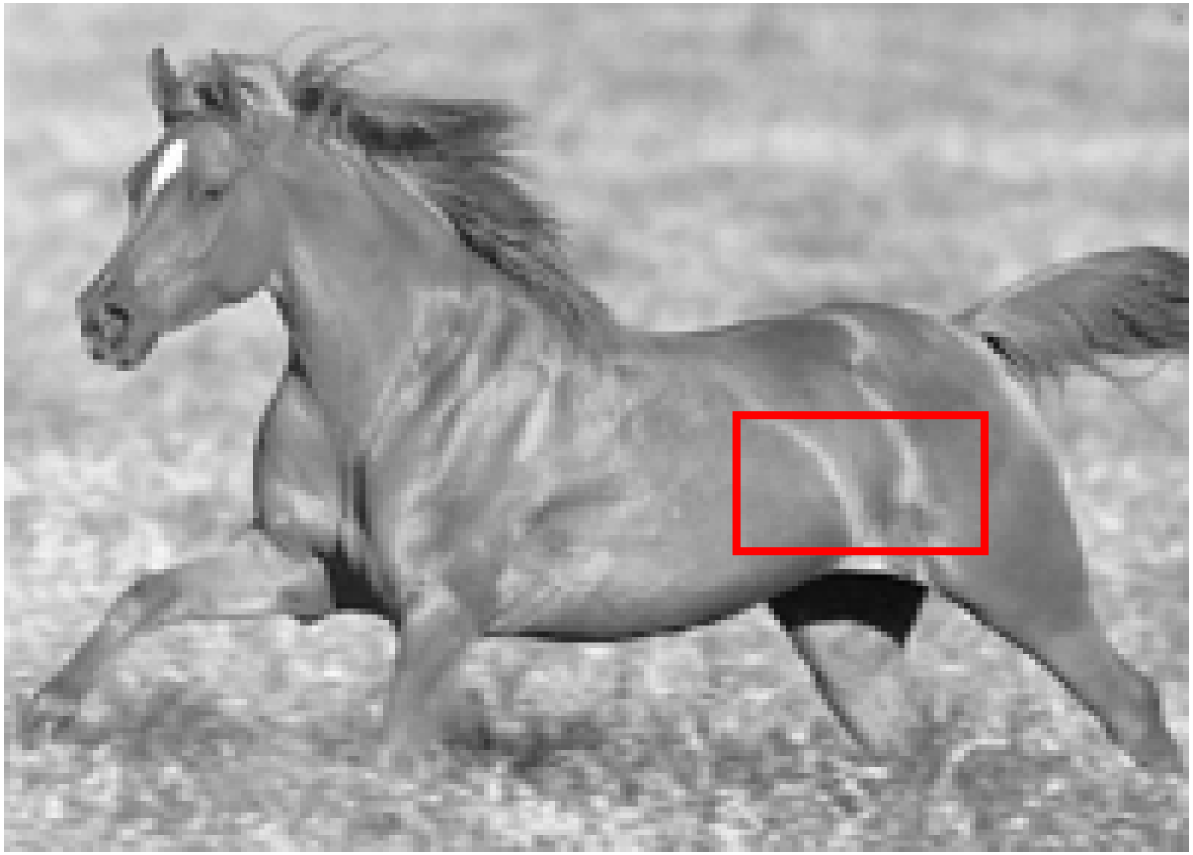
What if we want to find *circles*?

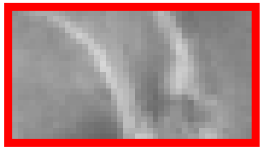
Note

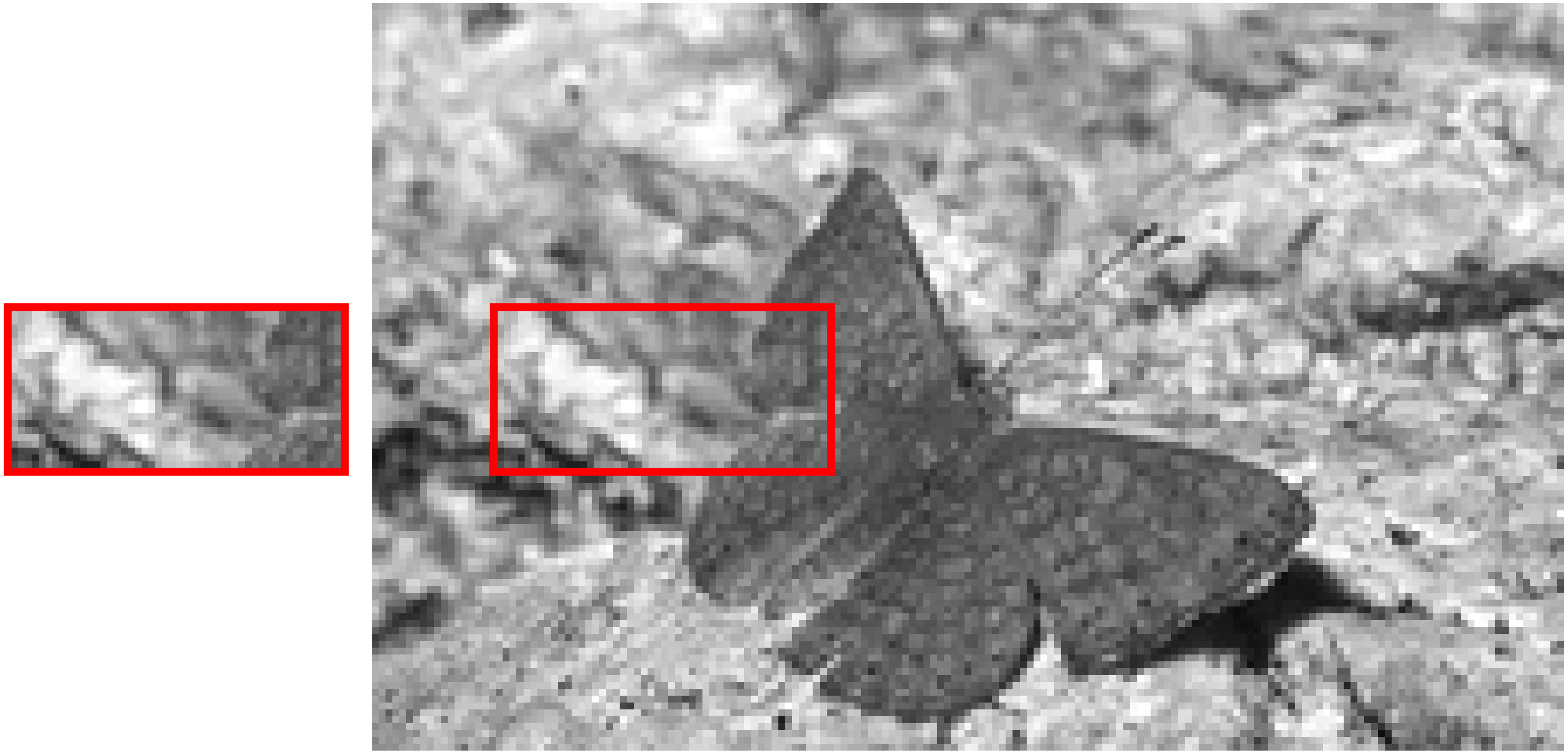
- Local processing is often insufficient to separate objects
- We reviewed several approaches for
  - curve extraction, completion
  - region segmentation





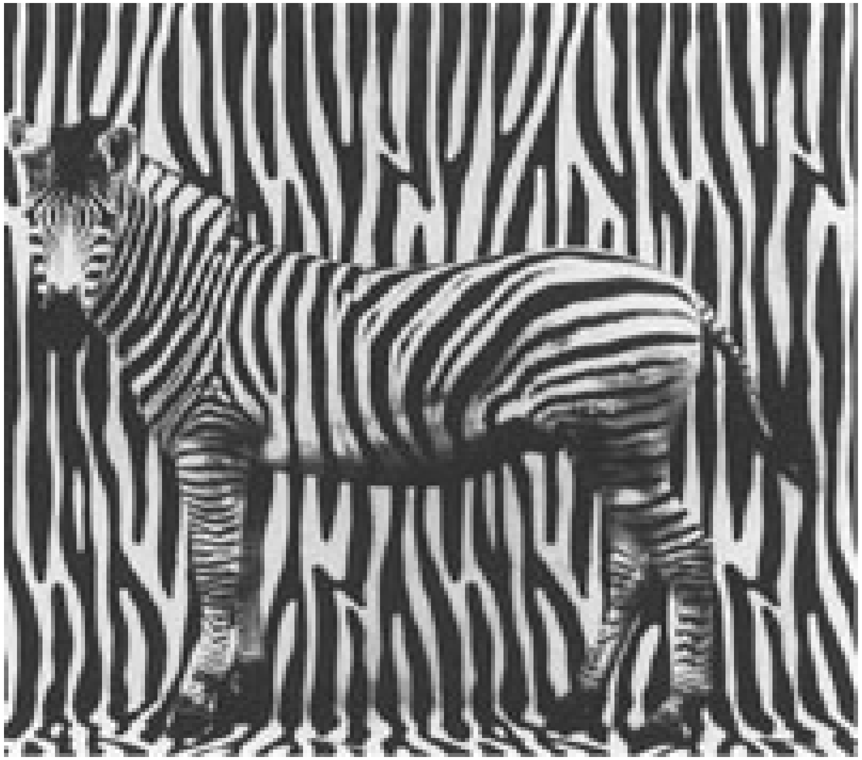






Camouflage(伪装)

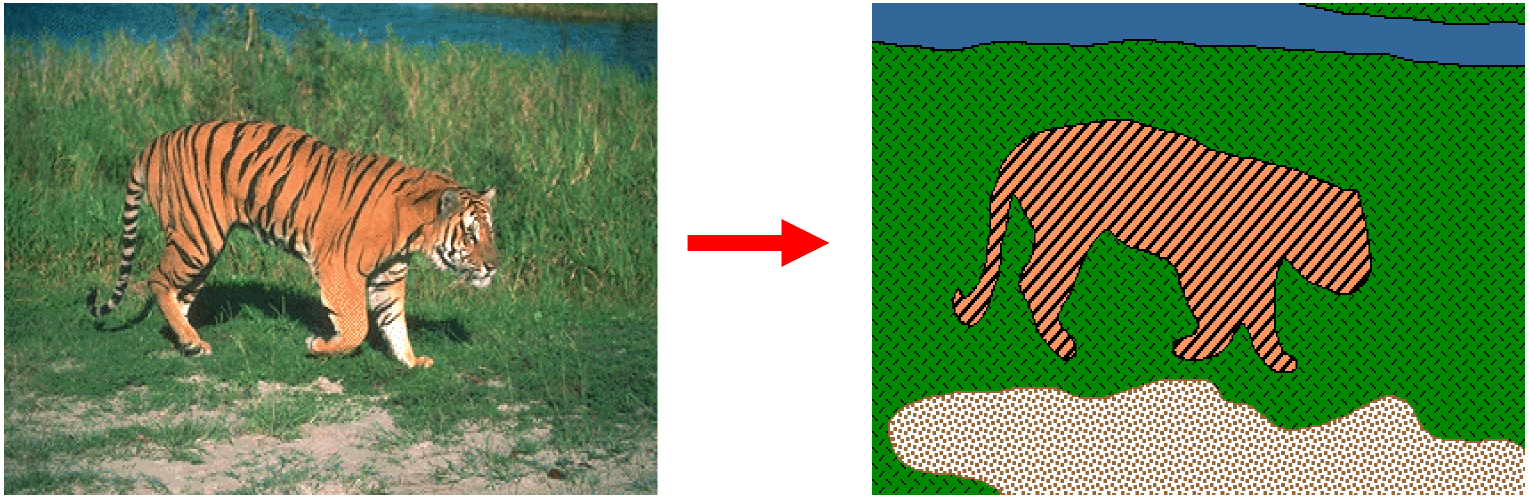
A Final Example



Outline

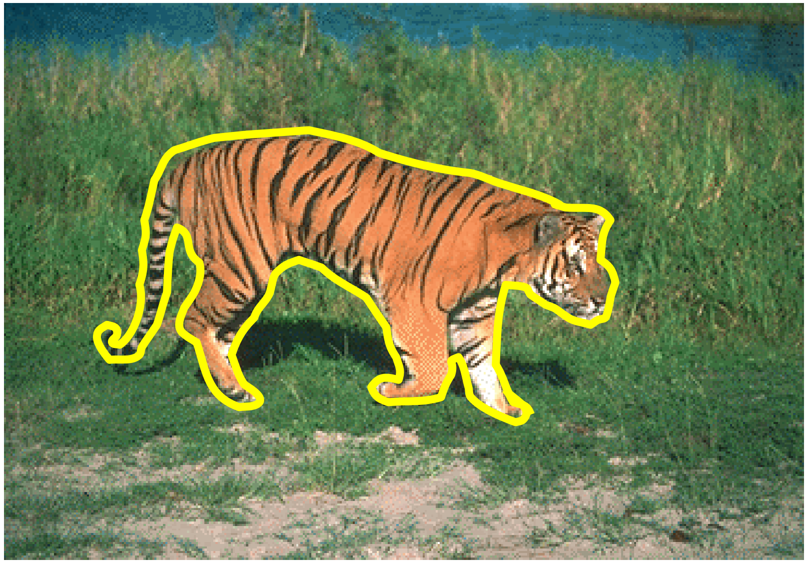
- Segmentation Challenges
- Segmentation Approaches
- Segmentation by Clustering
- Segmentation by Graph

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

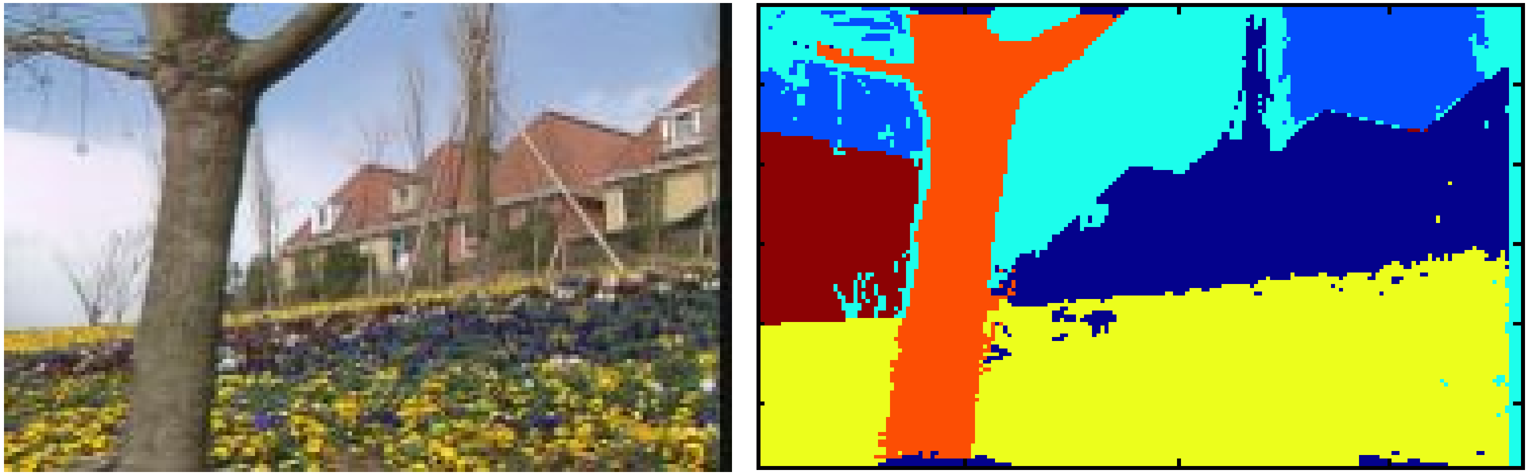
 

- How could this be done?

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



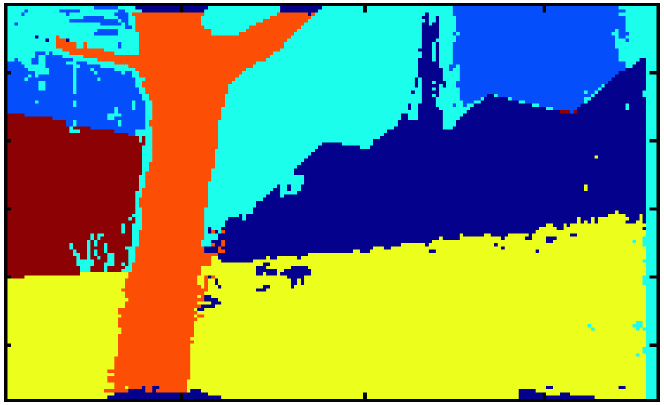
Layer Representation

Segmentation

- Find set of regions R_1, R_2, ….,R_n such that

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

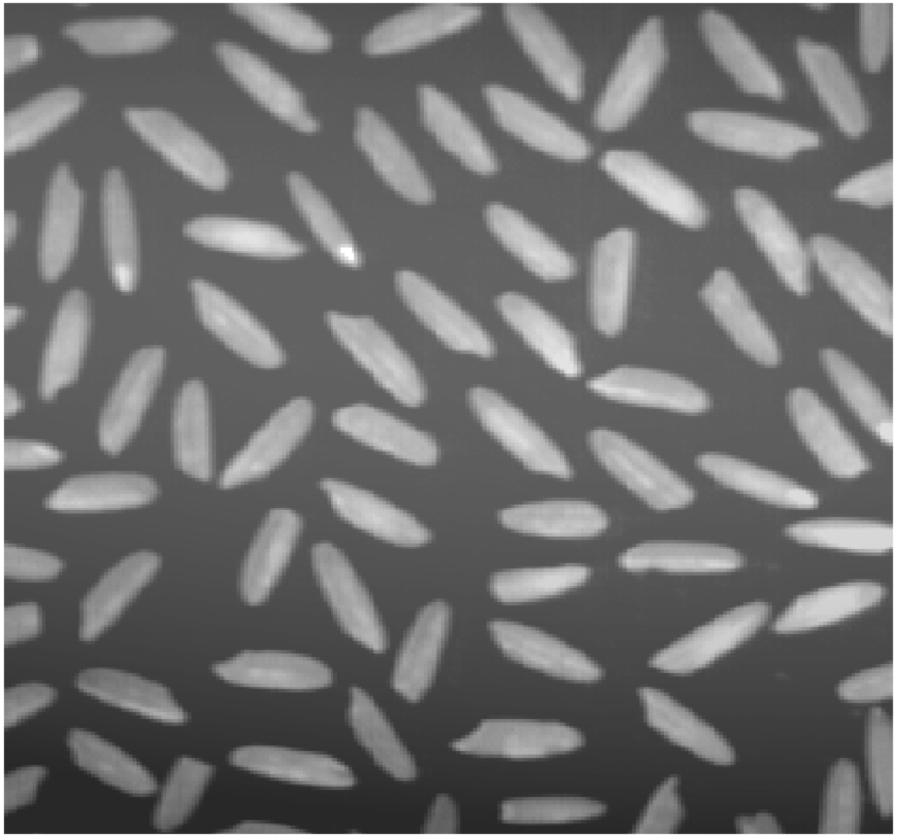
- All pixels in region i satisfy some similarity constraint



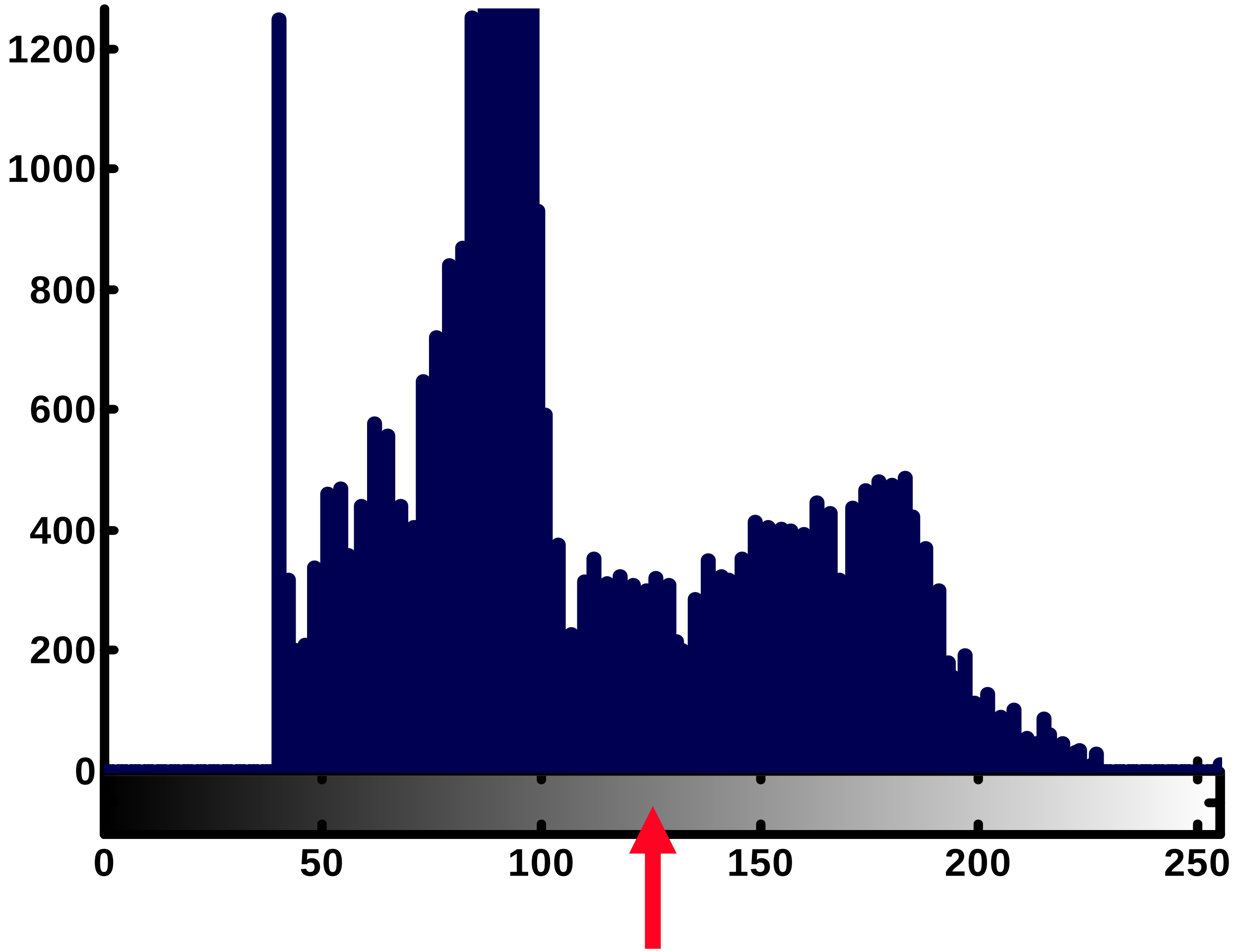
Similarity Constraints

- Pixels in any sub-image must have the **same** gray levels.
- Pixels in any sub-image **must not differ** more than some threshold
- 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.

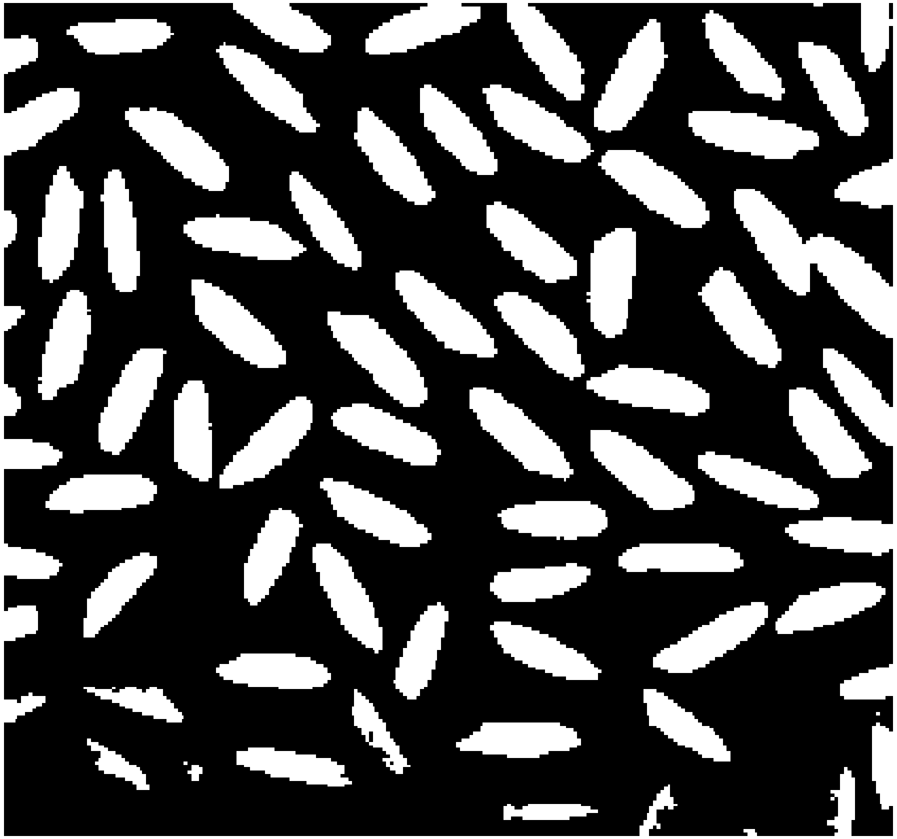
Image Segmentation: Thresholding



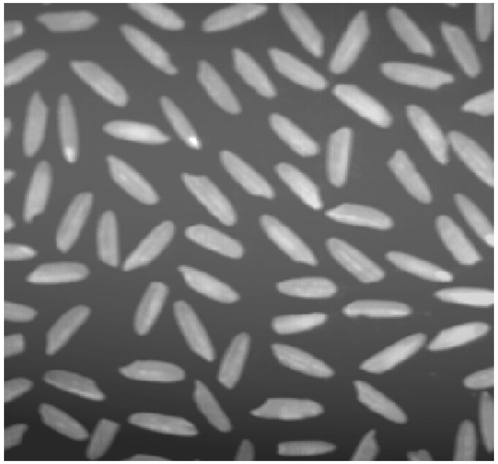
Histogram

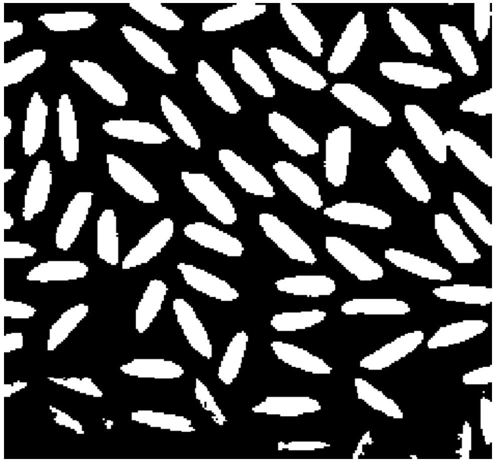


Thresholding



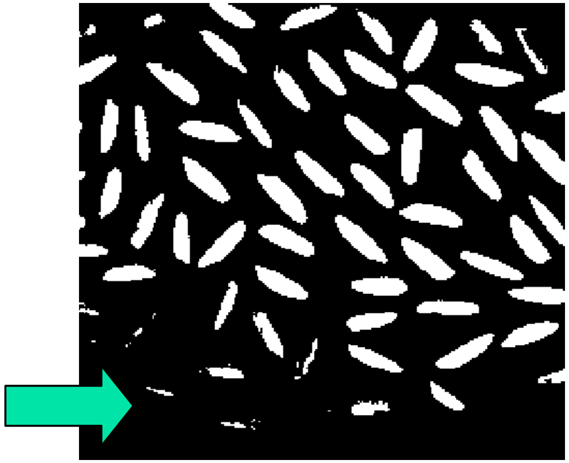
Thresholding

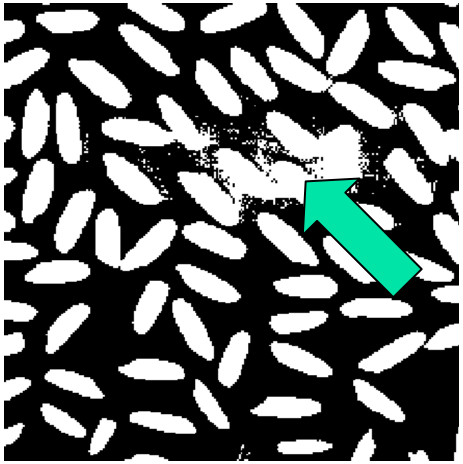




125

99

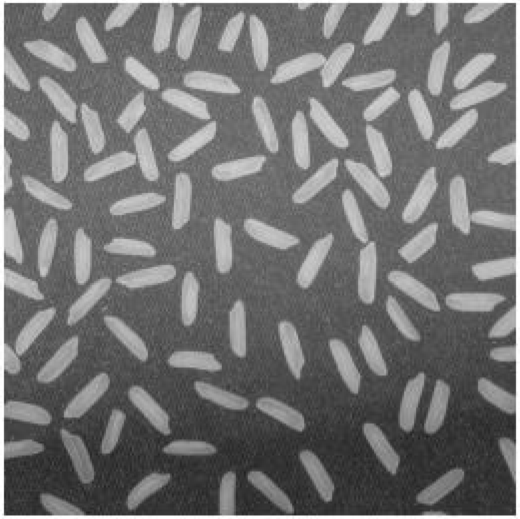


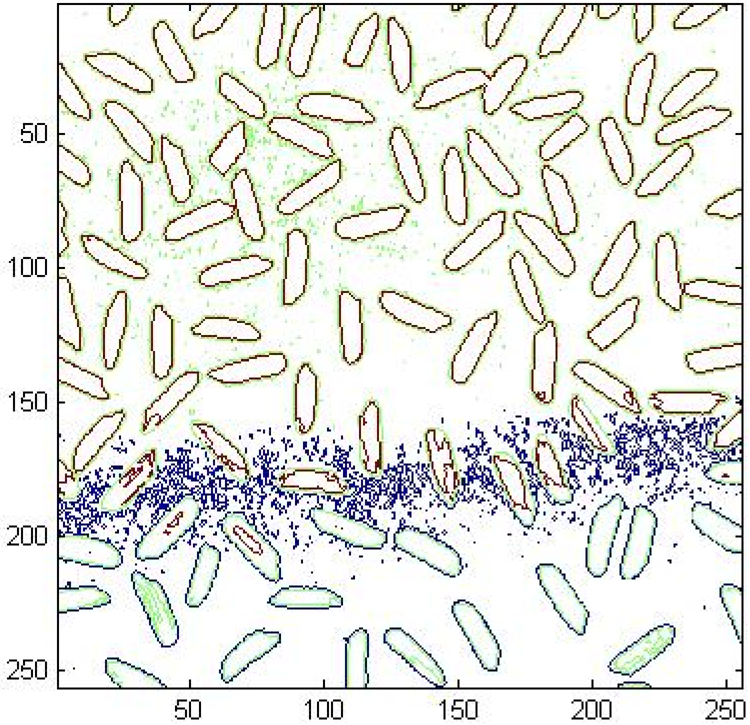


156

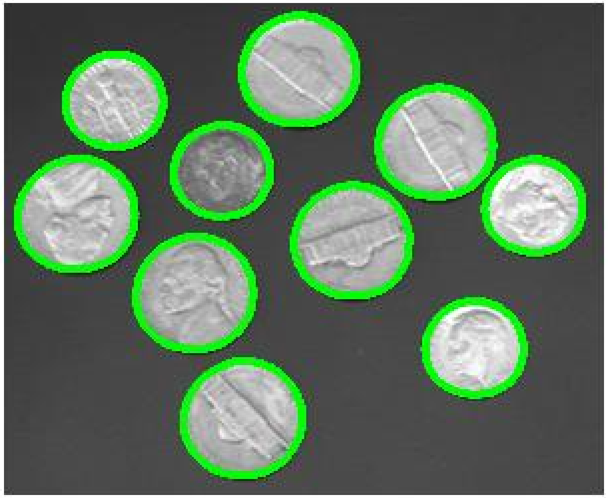
Matlab Demo 0: Contours

- 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
figure;imshow(BW_filled);
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

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

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

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

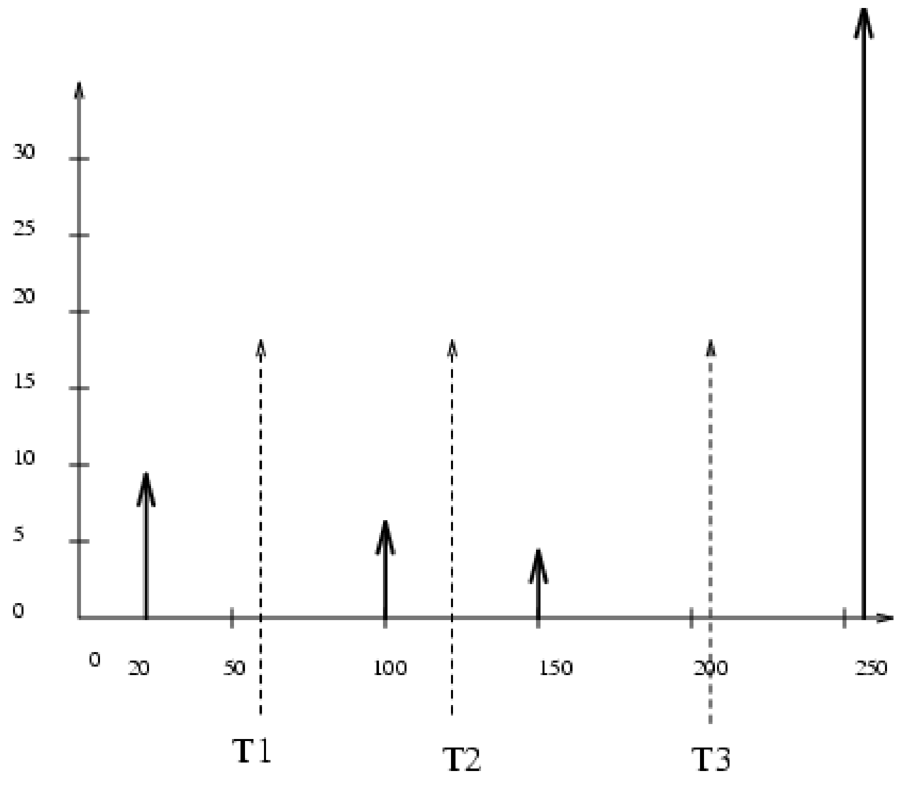
Image Histogram

- 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



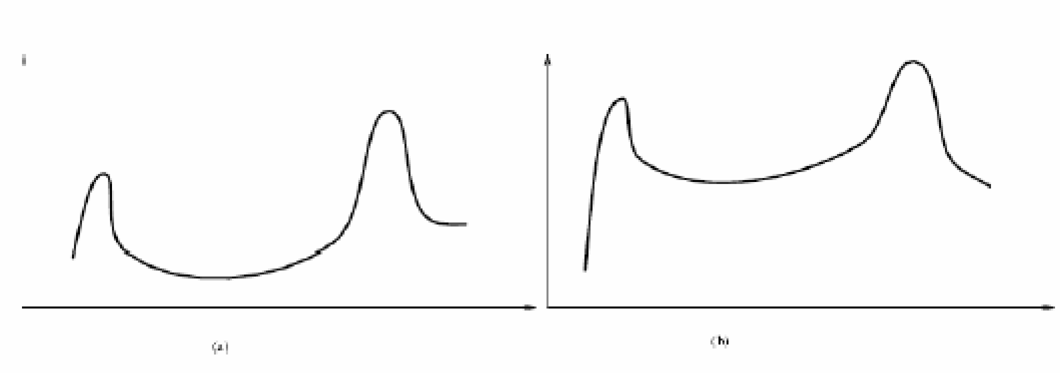
Segmentation Using Histogram Simple Case

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

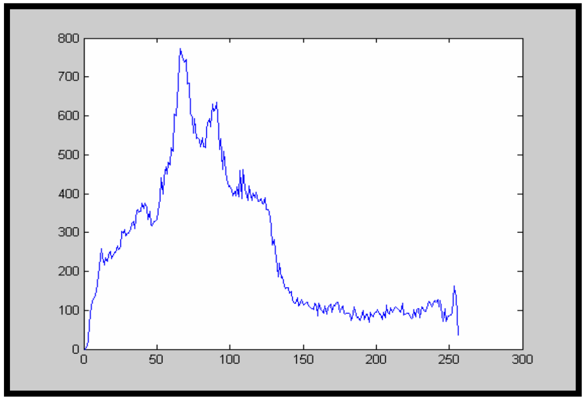
$$B_2(x,y) = \begin{cases} 1 & \text{if } T_1 < f(x,y) < T_2 \\ 0 & \text{Otherwise} \end{cases}$$

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

Realistic Histograms



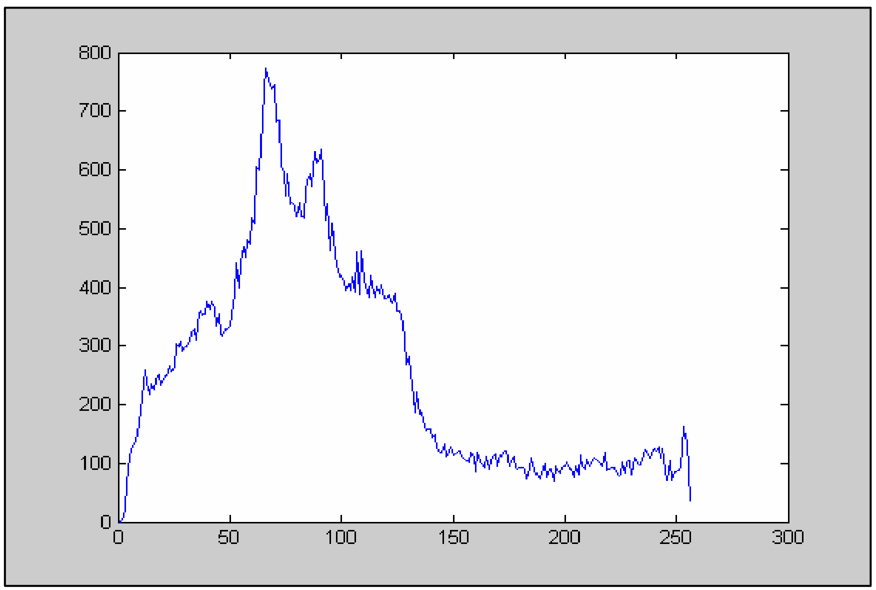
Not realistic

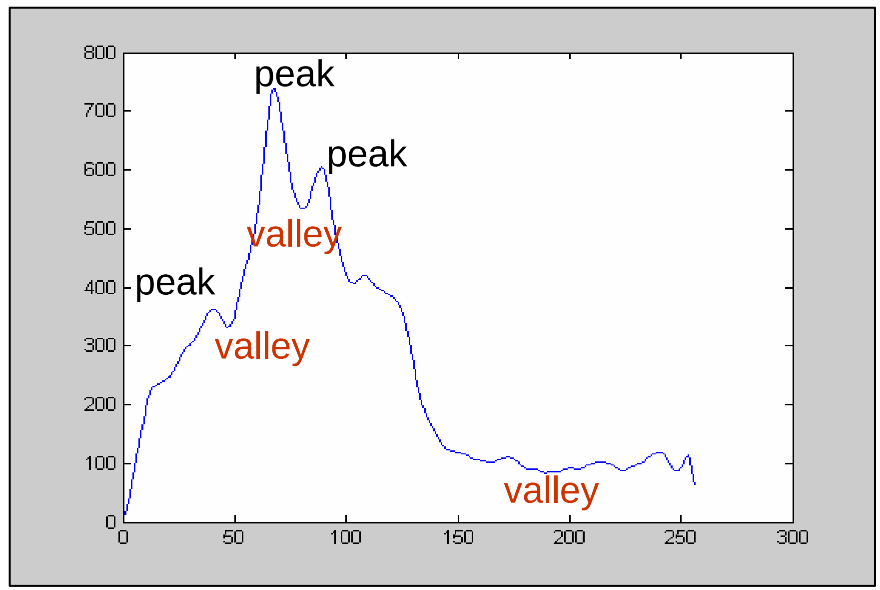


Real (noise)

Realistic Histograms

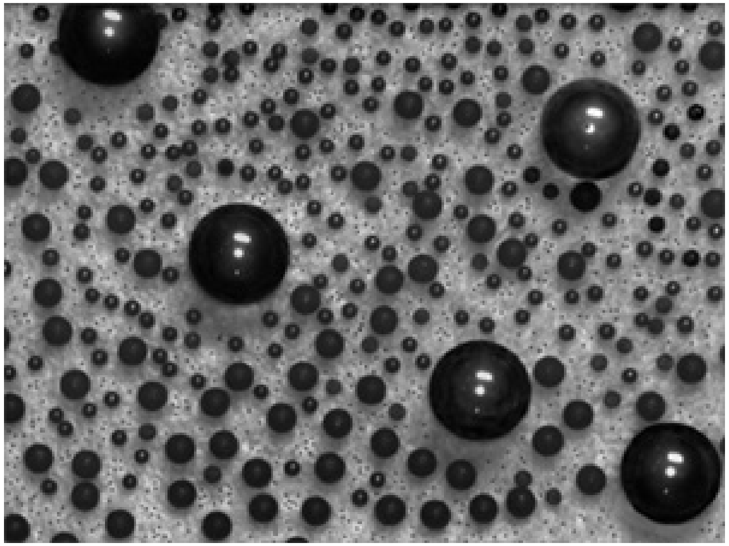
- Smooth out noise in the histogram
  - Convolve by averaging or 1D Gaussian filter

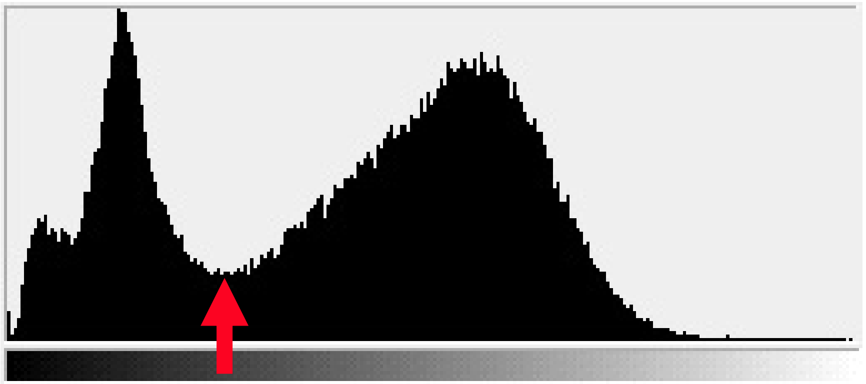




Histogram-based segmentation

- 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

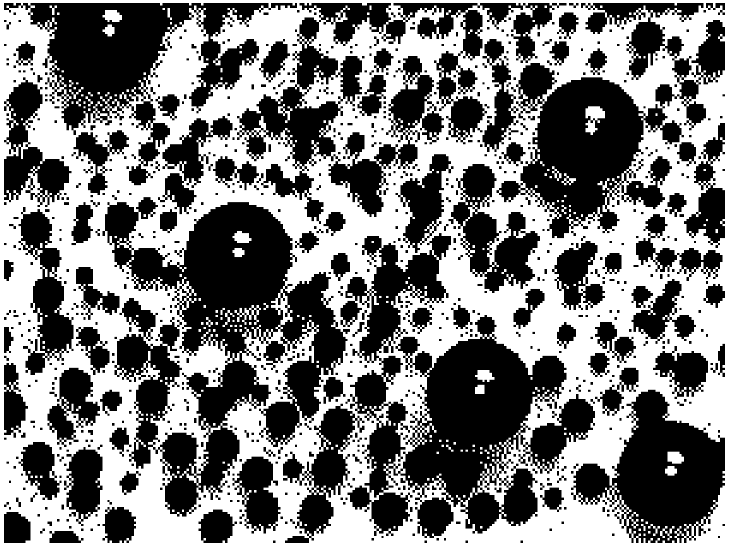


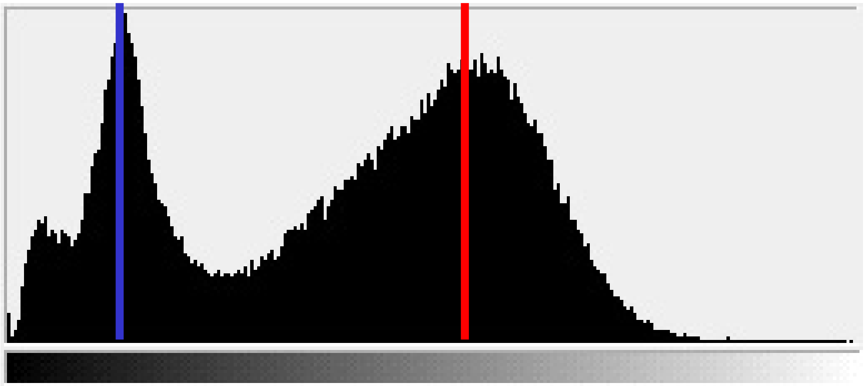


Cut here

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



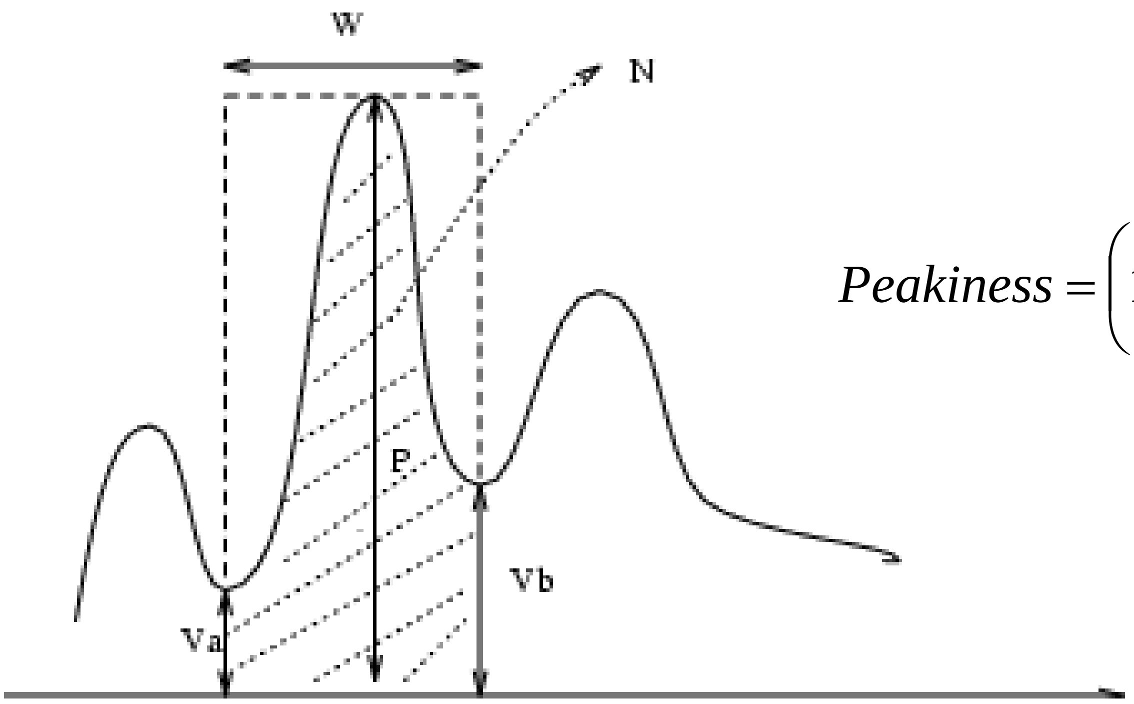


- Here's what it looks like if we use two colors

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 to find connected regions.

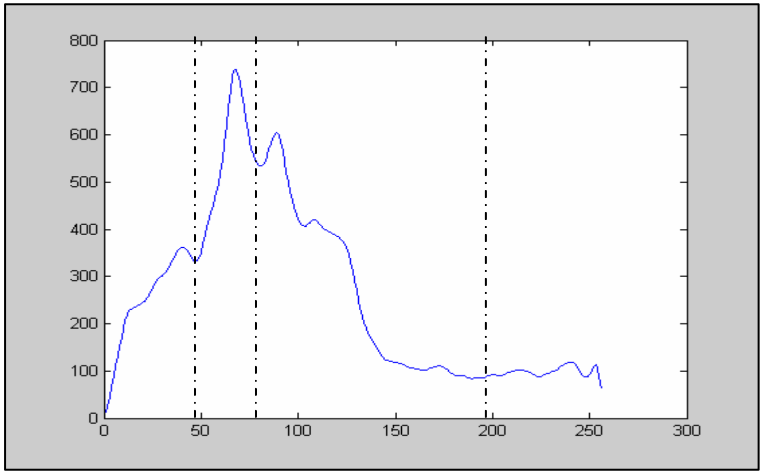
Good Peaks
Peakiness Test



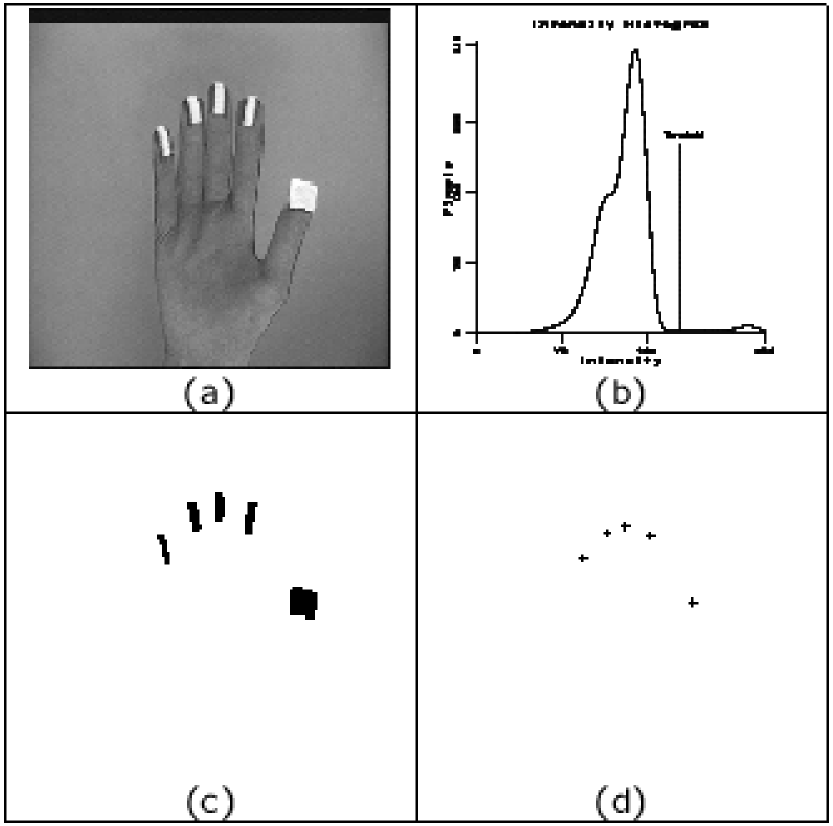
$$Peakiness = \left(1 - \frac{(V_a + V_b)}{2P}\right) . \left(1 - \frac{N}{(W.P)}\right)$$

Segmentation Using Histograms

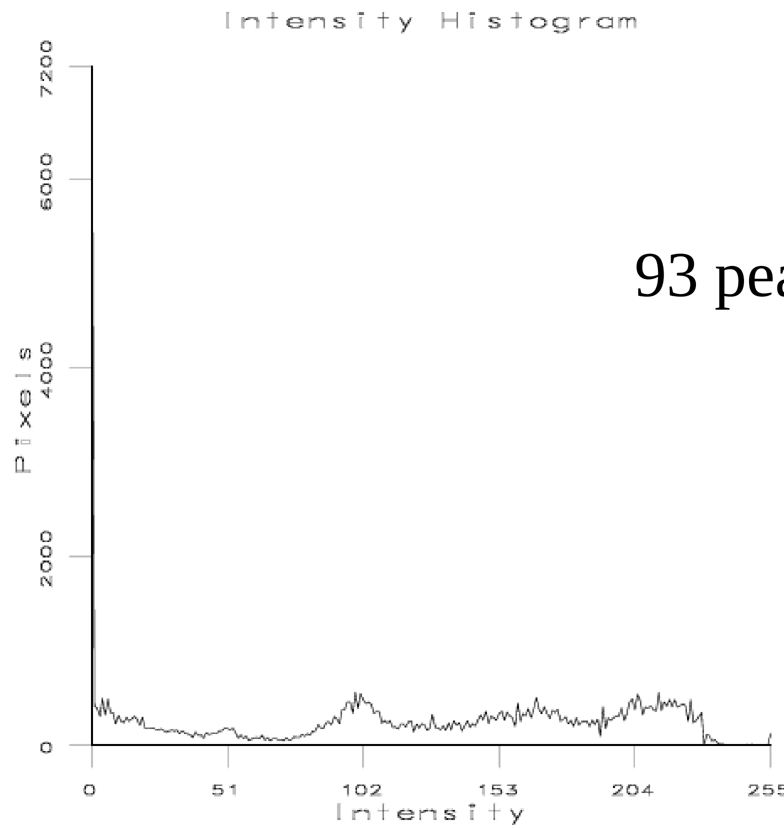
- Select the valleys as thresholds
  - Apply threshold to histogram
  - Label the pixels within the range of a threshold with same label, i.e., a, b, c … or 1, 2, 3 …



Example:
Detecting Finger Tips (marked white)

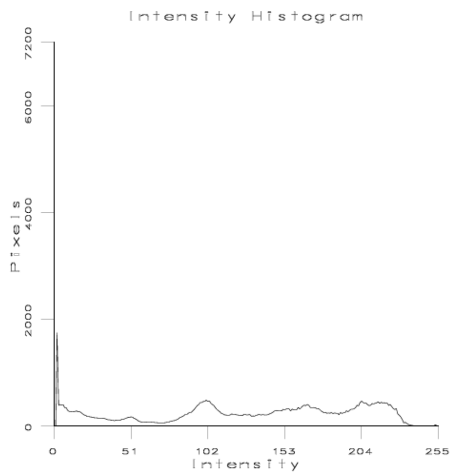


Example
Segmenting a bottle image

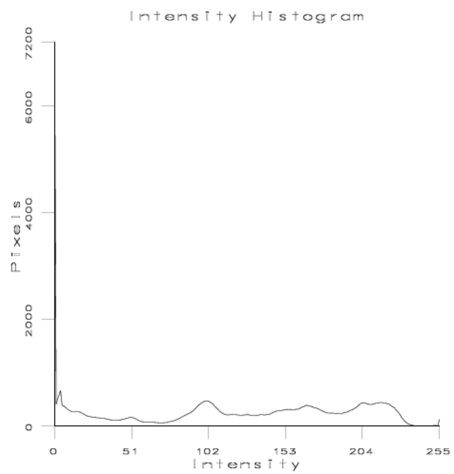


93 peaks

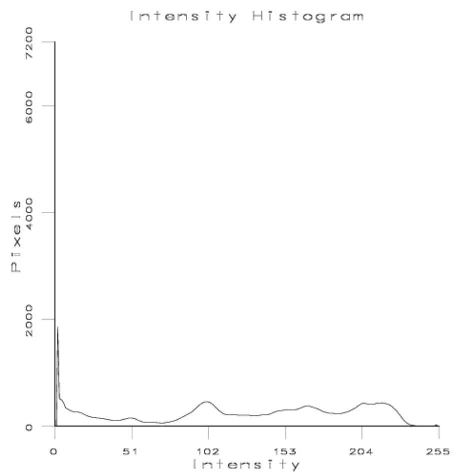
Example
Segmenting a bottle image



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)

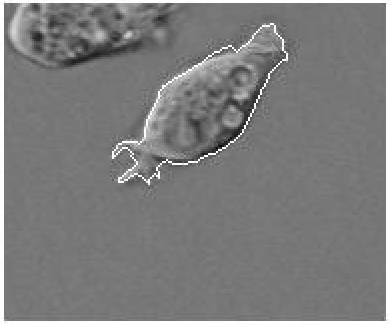
Difference Between *Segmentation* and *Edge Detection*

- 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');
```

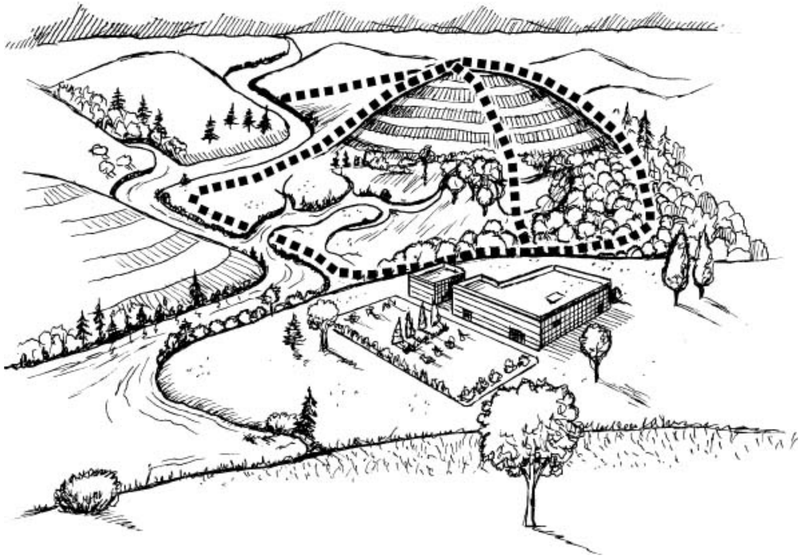
outlined original image



Watershed segmentation

Watershed Segmentation
分水岭算法

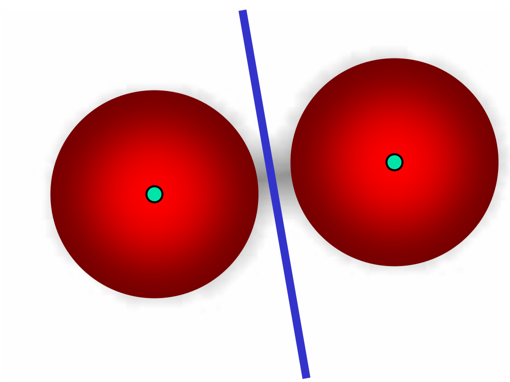
- Intensity of an image ~ elevation in a landscape
  - Flood from minima
  - Prevent merging of “catchment basins”
  - Watershed borders built at contacts between basins

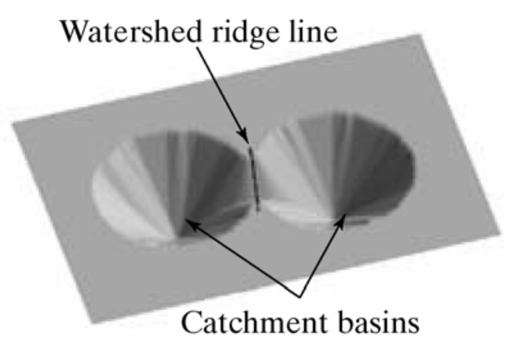


http://www.ctic.purdue.edu/KYW/glossary/whatisaws.html

Basic Definitions

- Three types of points
  - Points belonging to a regional minimum
  - Catchment basin / watershed of a regional minimum
    - Points at which a drop of water will certainly fall to a single minimum
  - Divide lines / Watershed lines
    - Points at which a drop of water will be equally likely to fall to more than one minimum
    - Crest lines on the topographic surface
- This technique is to identify all the third type of points for segmentation

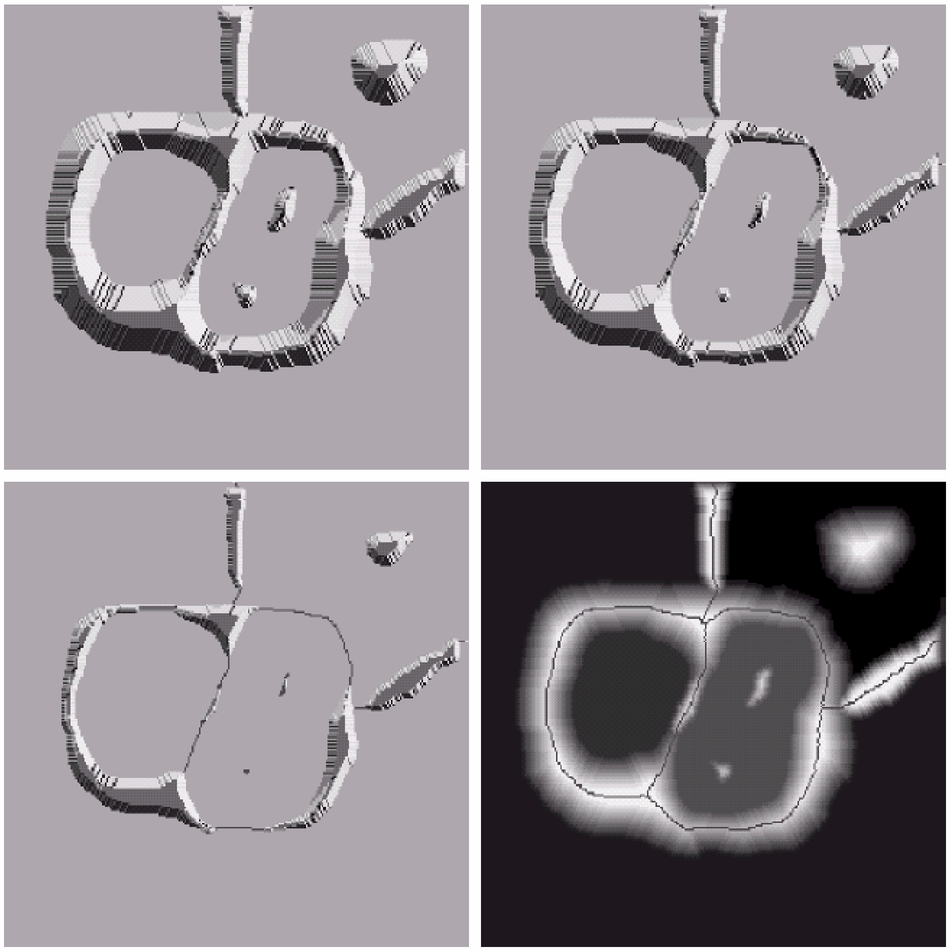




Basic Steps

1. Piercing holes in each regional minimum of I
2. The 3D topography(地形) is flooded from below gradually
3. When the rising water in distinct catchment basins is about to merge, a dam is built to prevent the merging

Watershed segmentation



e f
g h

FIGURE 10.44 *(Continued)* (e) Result of further flooding. (f) Beginning of merging of water from two catchment basins (a short dam was built between them). (g) Longer dams. (h) Final watershed (segmentation) lines. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

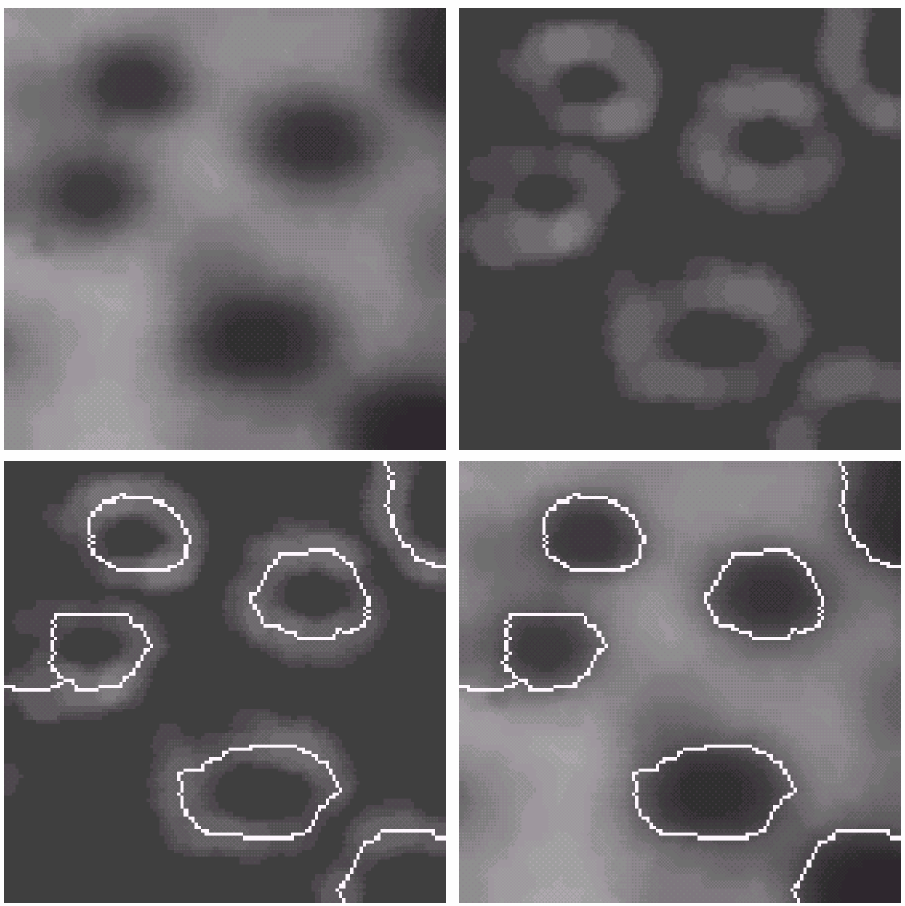
- The dam boundaries correspond to the watershed lines to be extracted by a watershed segmentation algorithm

- Eventually only constructed dams can be seen from above

Watershed Segmentation -- some tips

- Instead of working on an image itself, this technique is often applied on its gradient image.
- Smoothing is usually employed to prevent over-segmentation.

example watershed



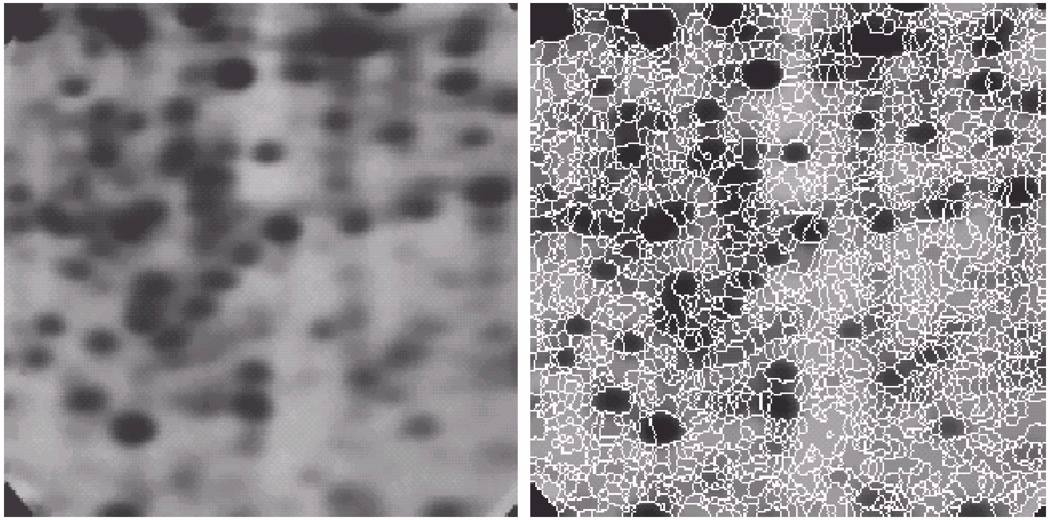
A B
C D

A: Image of blobs

B: gradient image

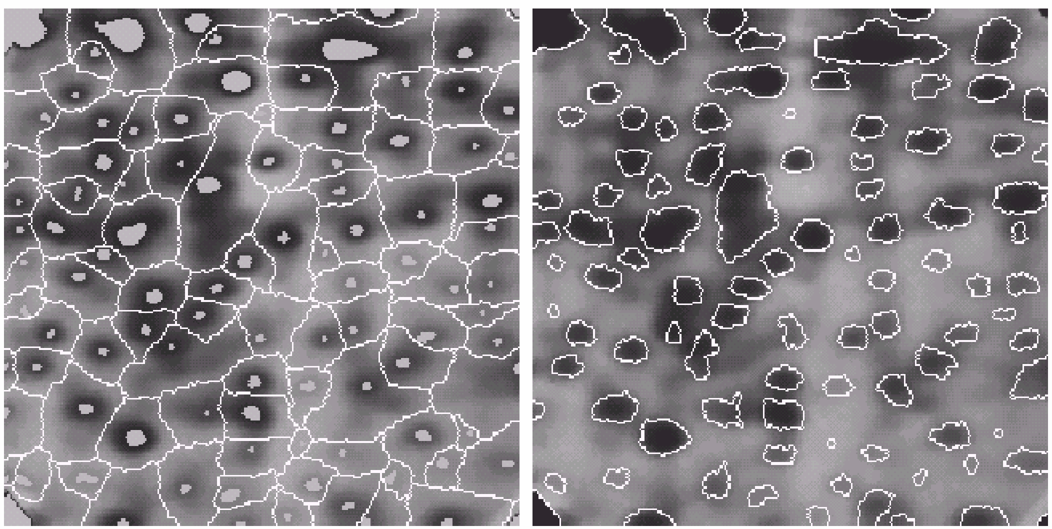
C: Watershed lines of image B

D: superimposed on origina;



a b

FIGURE 10.47 (a) Electrophoresis image. (b) Result of applying the watershed segmentation algorithm to the gradient image. Oversegmentation is evident. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)



a b

FIGURE 10.48 (a) Image showing internal markers (light gray regions) and external markers (watershed lines). (b) Result of segmentation. Note the improvement over Fig. 10.47(b). (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

Adding “markers”:

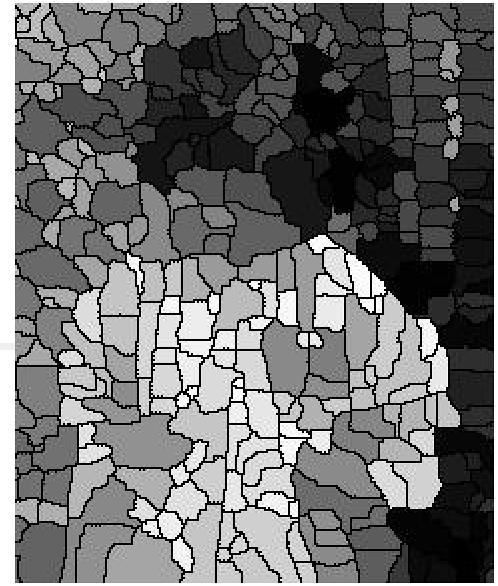
internal: belong to objects of interest

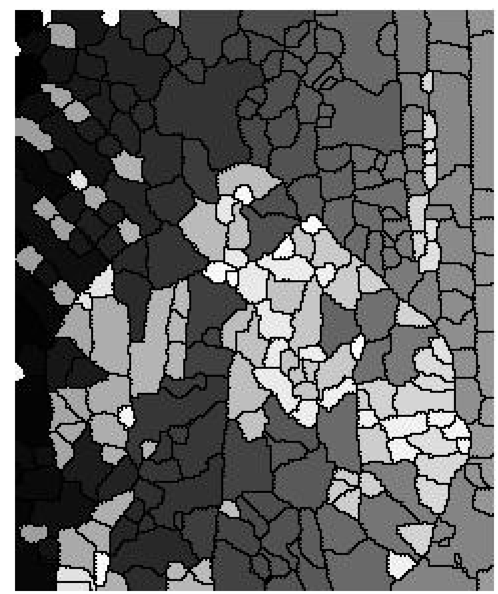
external: associated with the background

Smoothing is an important step

Matlab demo 3
--Watershed

```
I=imread('pout.tif');
figure, imshow(I);
%Smooth before watershed transform
Ir=I(:,:,1);
H = fspecial('disk',2);
blurredIr = imfilter(Ir,H,'replicate');
%First try original watershed transform
mask1=watershed(blurredIr);
figure,imshow(mask1,[]);
 %Then try gradient watershed transform
[height width]=size(Ir);
[gx gy]=gradient(double(Ir));
grad=uint8(sqrt(gx.^2+gy.^2));
%Smooth before watershed transform
H = fspecial('disk',3);
blurredgrad = imfilter(grad,H,'replicate');
mask2=watershed(blurredgrad);
figure,imshow(mask2,[]);
```



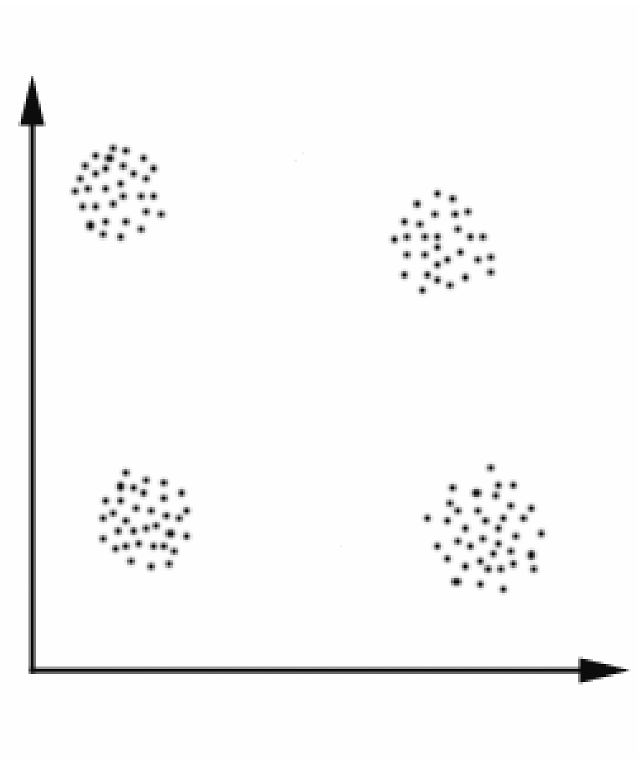


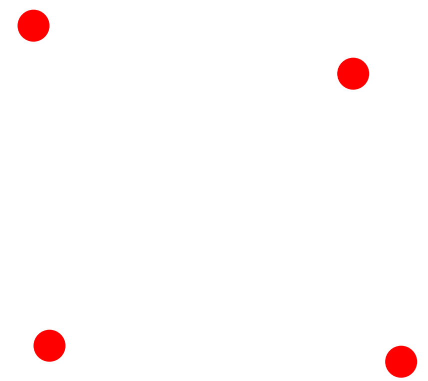
Which one is better?

Outline

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

Clustering (聚类) Principle





Example

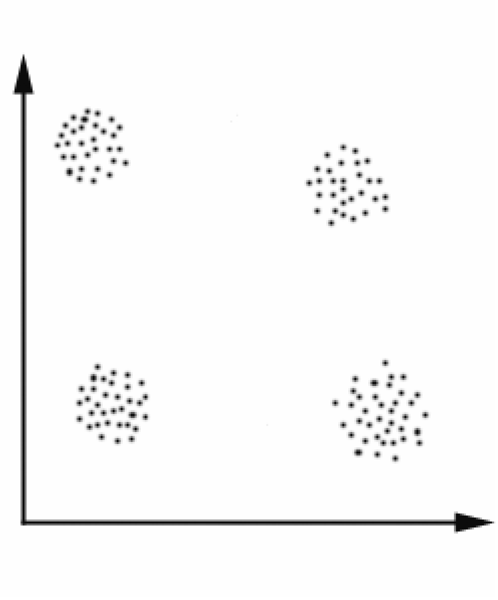
Image

Clusters on intensity

Clusters on color

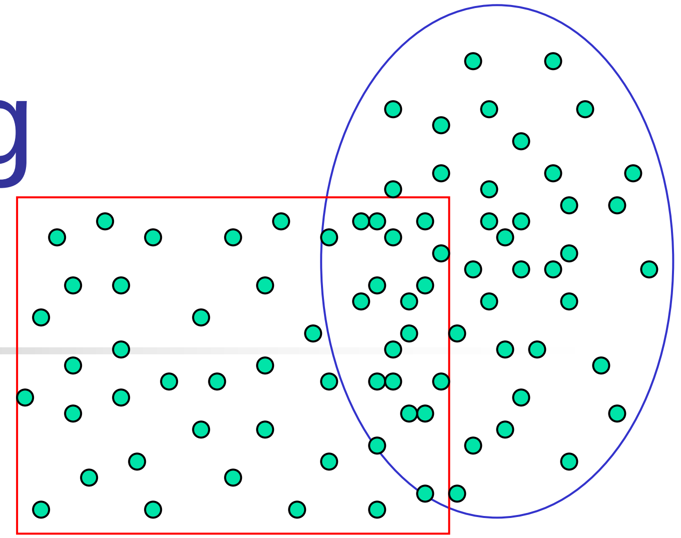
Clustering

- We want to group together some primitives



Seems easy, but…

Advanced Clustering



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

- 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,distribution…*
- 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…

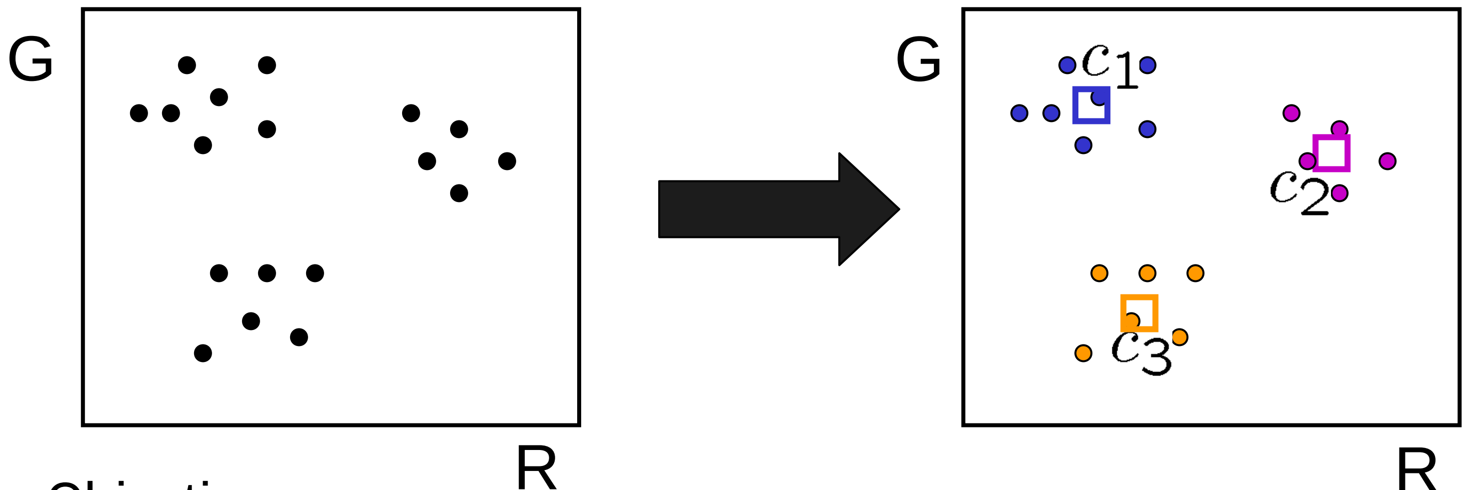
Clustering

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

Clustering

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

For each group, we define a representative point as its description

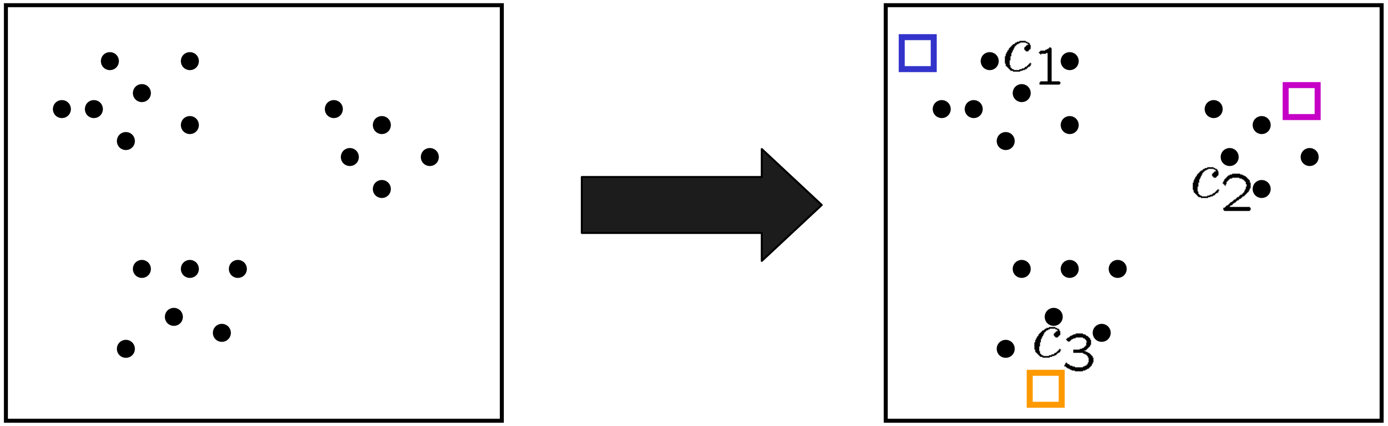


- Objective
  - Each point should be as close as possible to a cluster center
    - Minimize sum squared distance of each point to closest center

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

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 c_i?
  - A: choose c_i to be the mean of all points in the cluster

K-means clustering

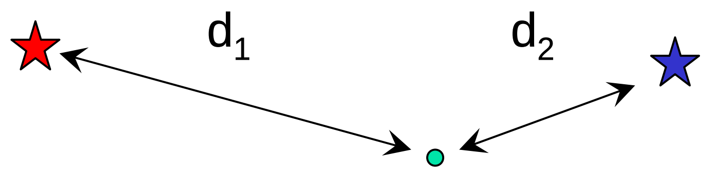
- K-means clustering algorithm
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_K$
  2. Given cluster centers, determine points in each cluster
     - For each point p, find the closest c_i. Put p into cluster i
  3. Given points in each cluster, solve for c_i
     - Set c_i to be the mean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

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

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

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

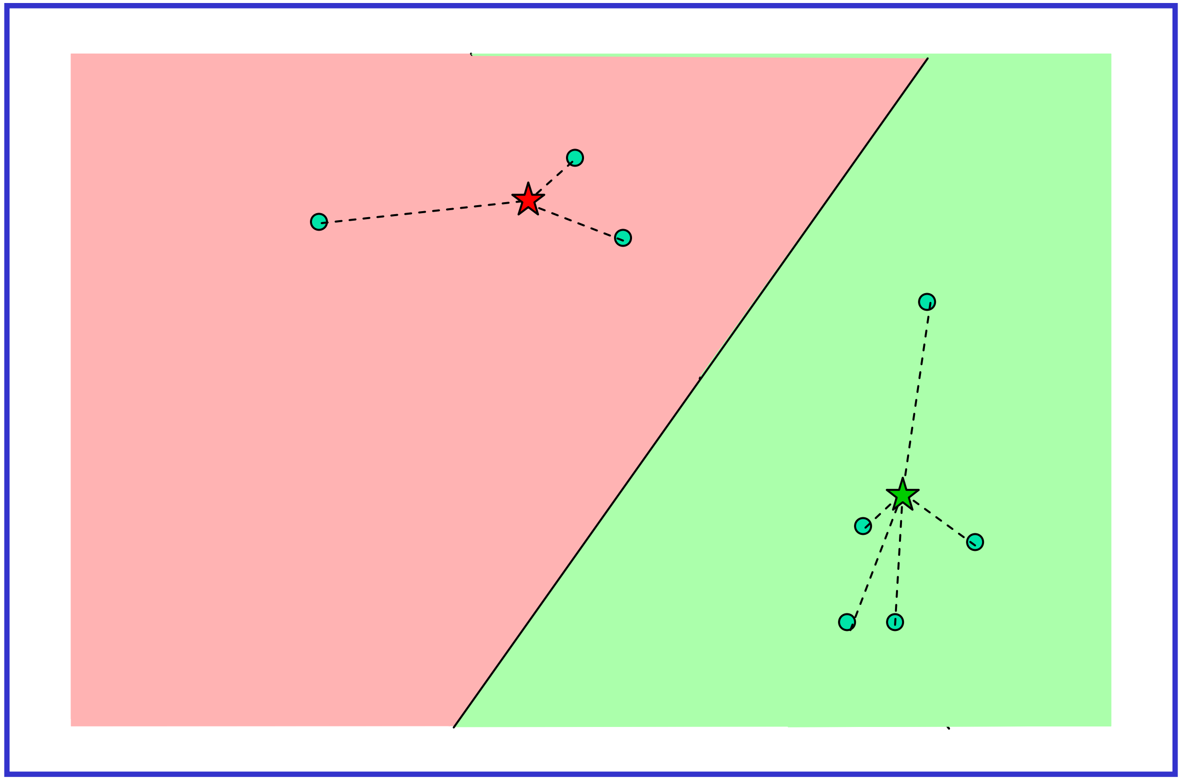
K-Means

- 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 exhaustive 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

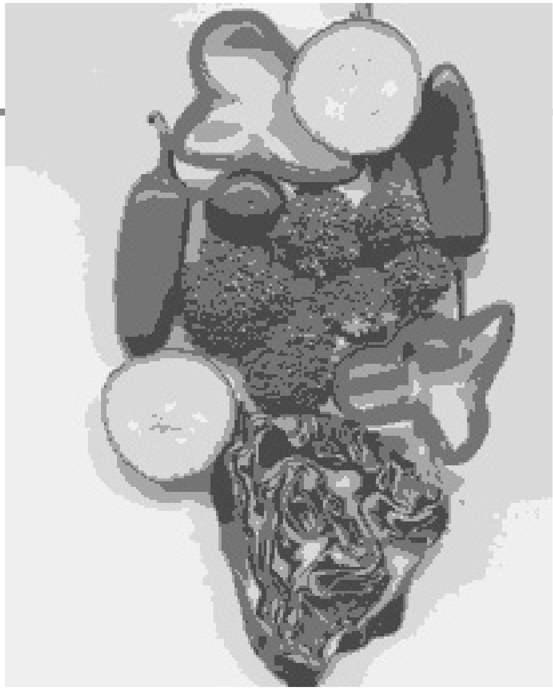
Image Segmentation by K-Means

- 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.
- 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

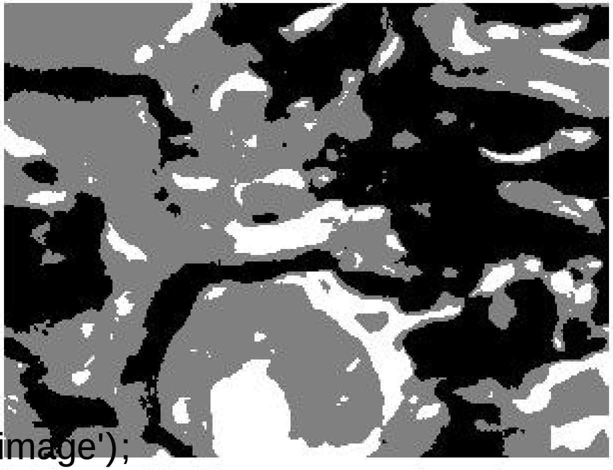
K-means clustering using intensity alone and color alone

* From Marc Pollefeys COMP 256 2003

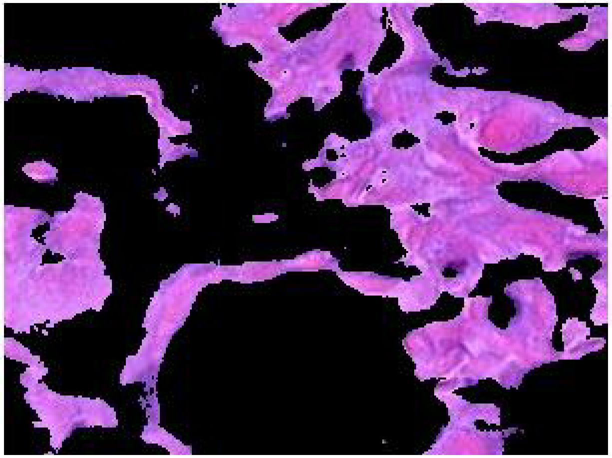
Expectation-Maximization

- 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 forecasting)
    - 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

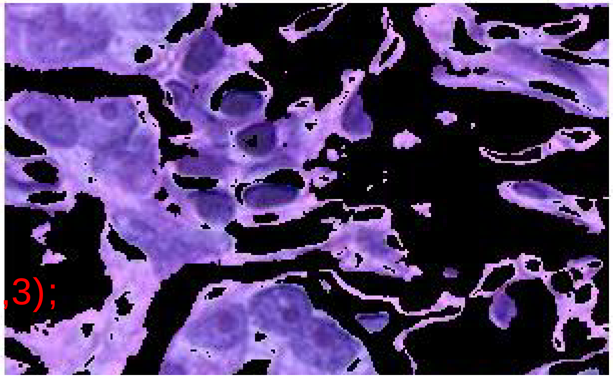
image labeled by cluster index



objects in cluster 1



objects in cluster 2



objects in cluster 3

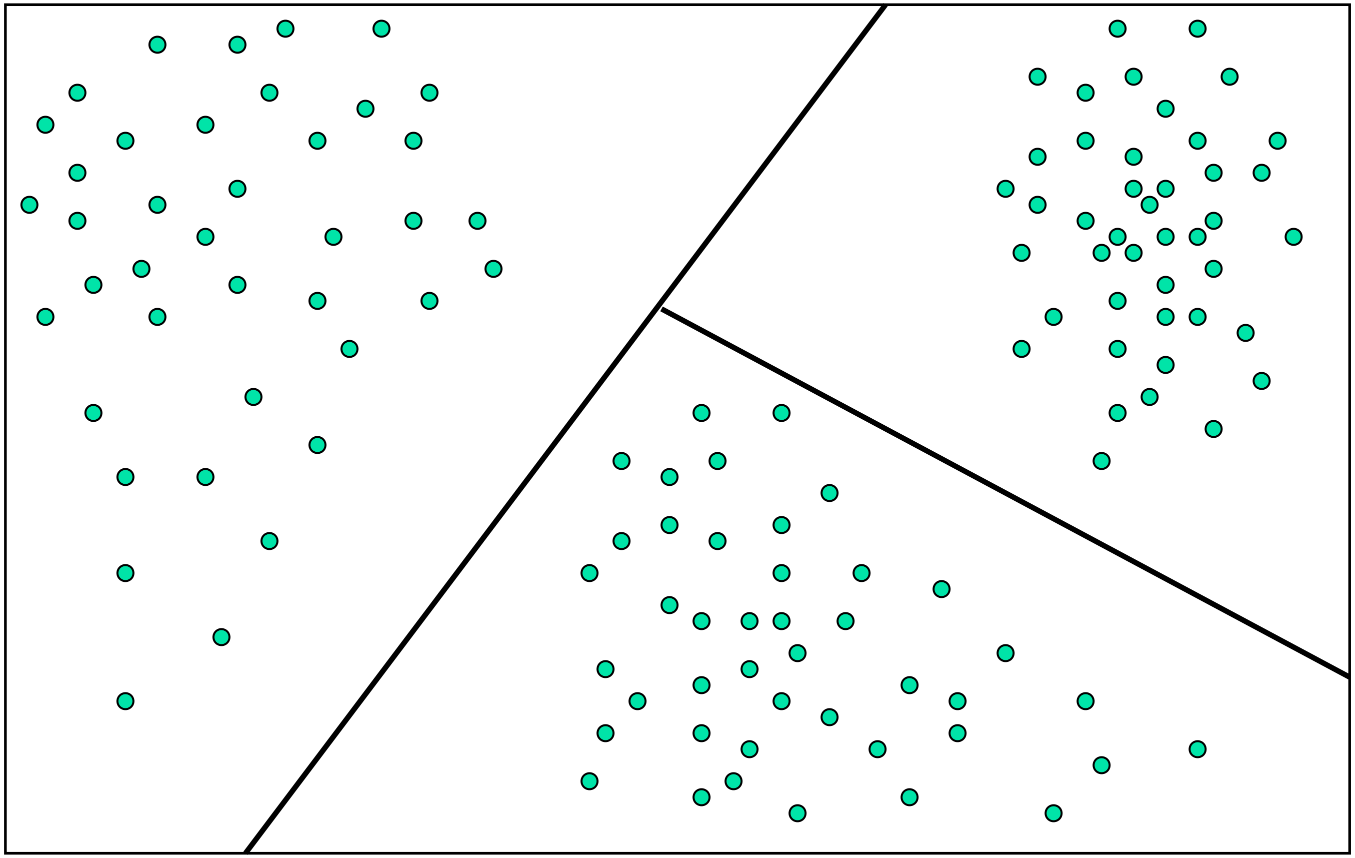
M

```
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;
[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');
```

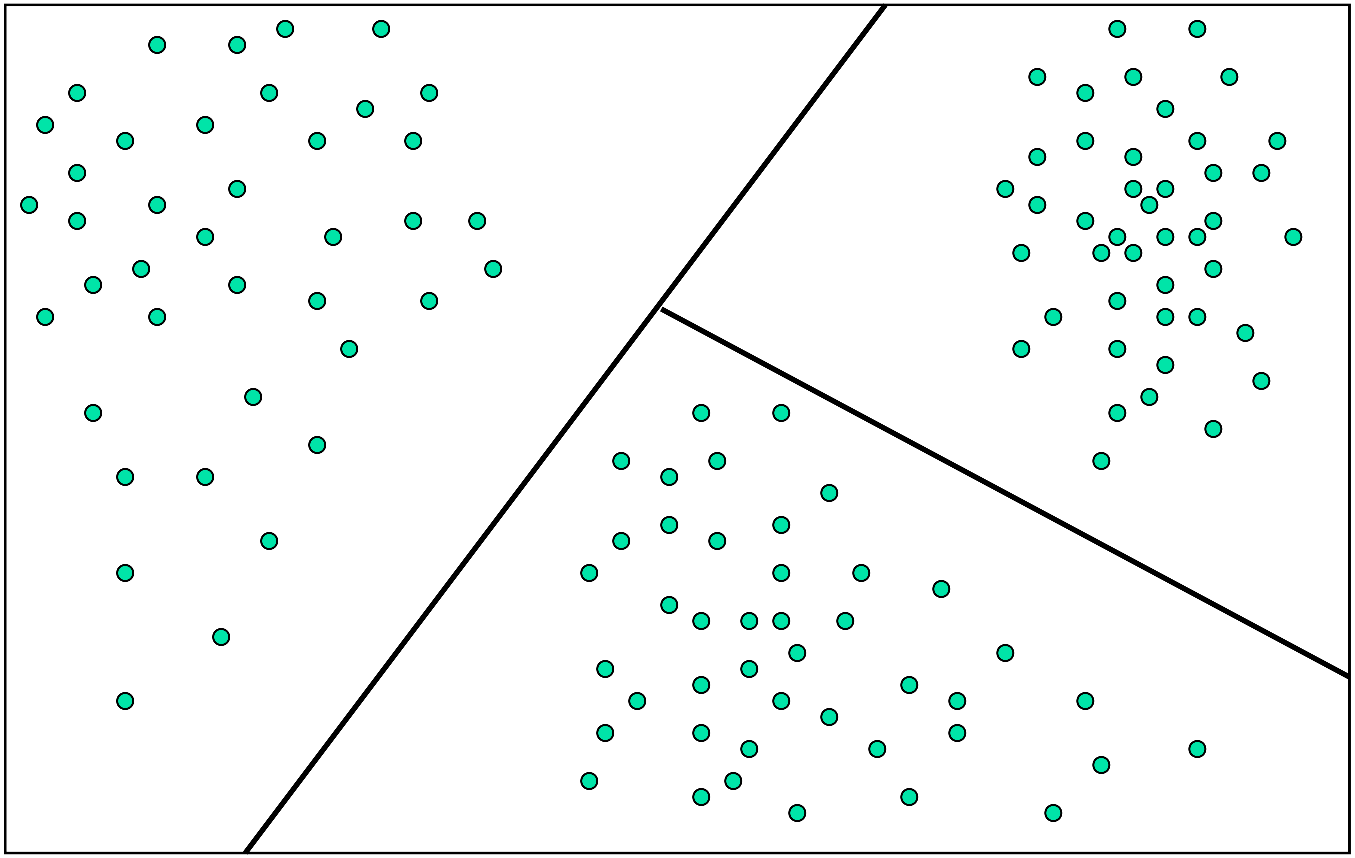
Outline

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

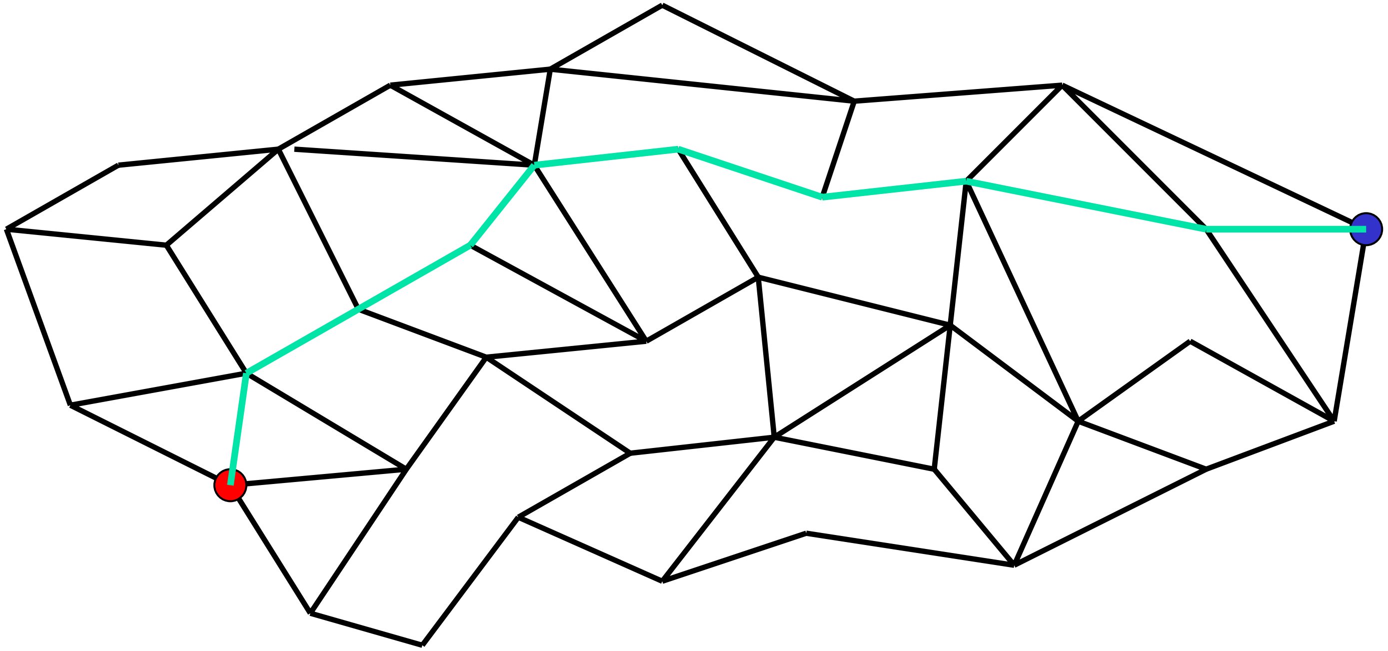
Clustering – Determine Regions



Graph –
Determine Boundaries



Preface—shortest path



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

Intelligent Scissors (demo)

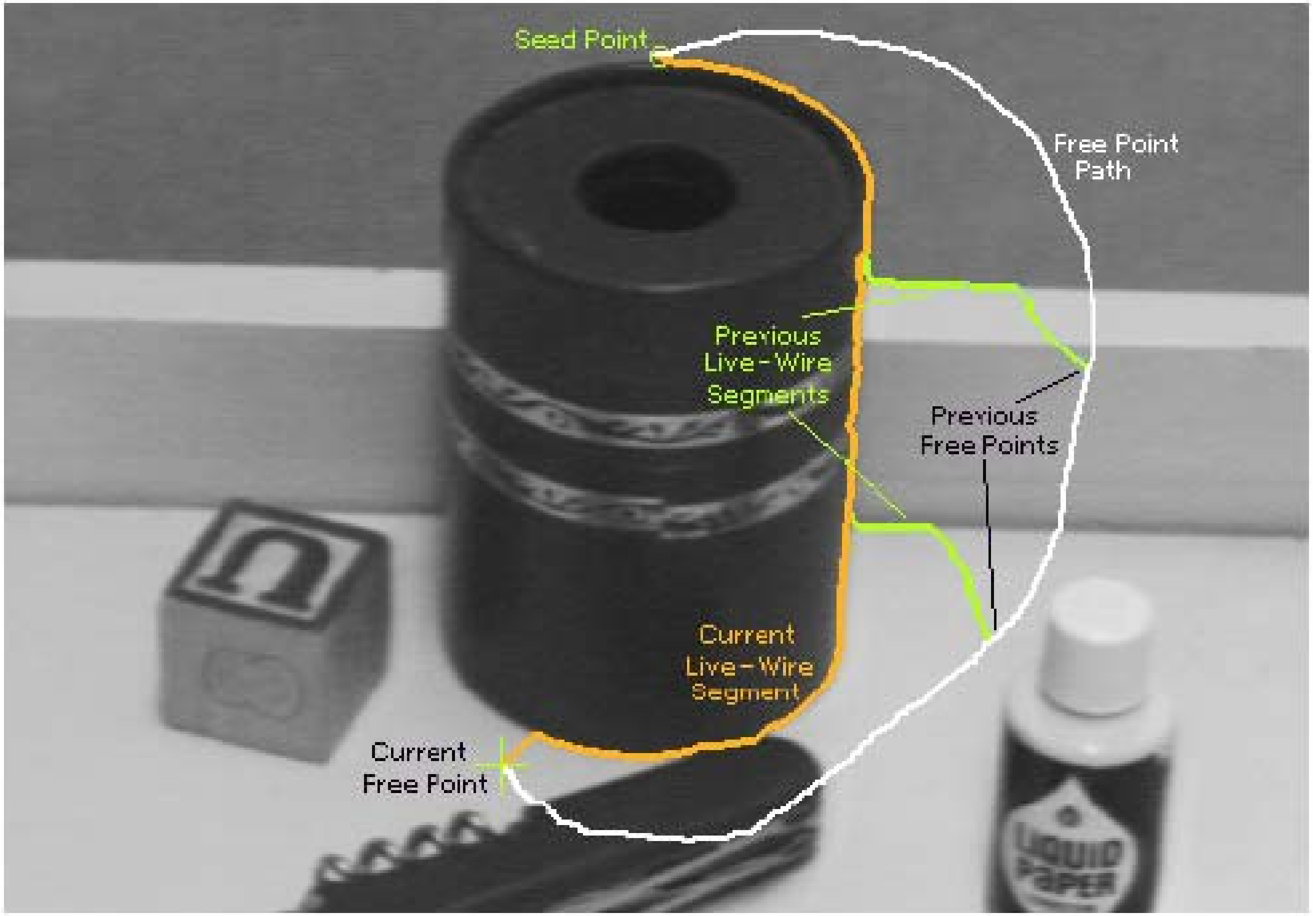


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

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?

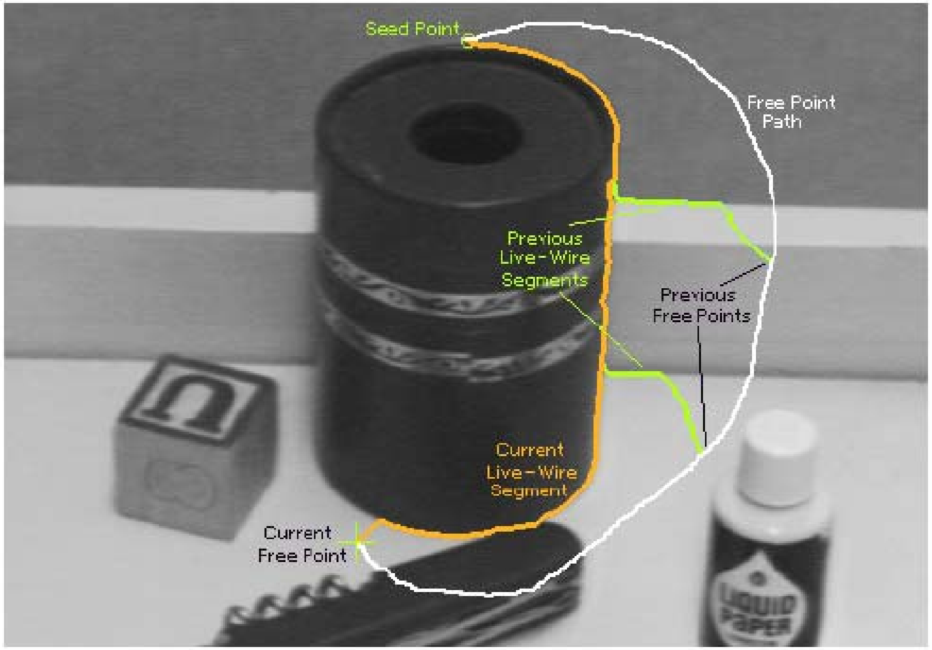
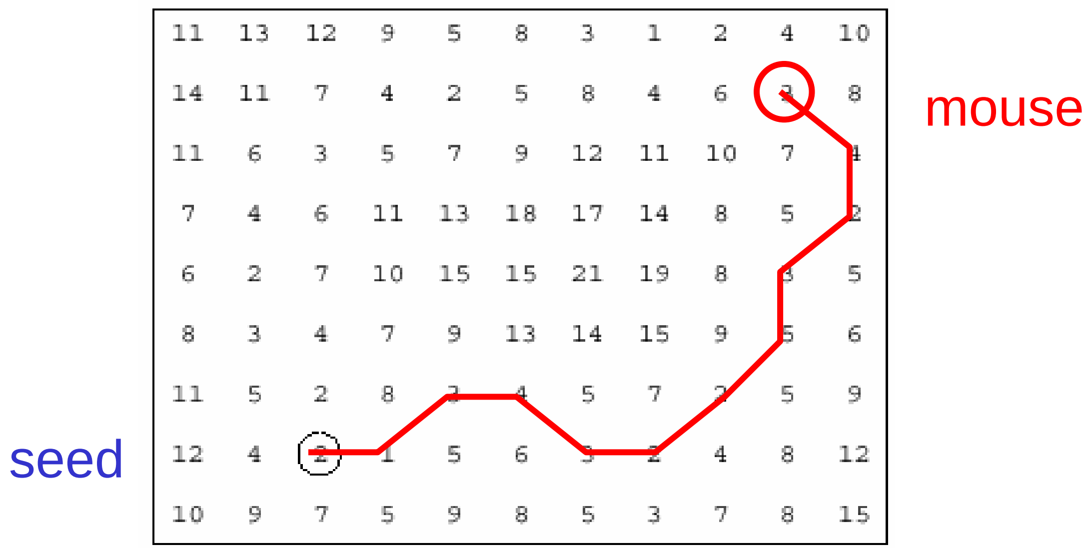


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors

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



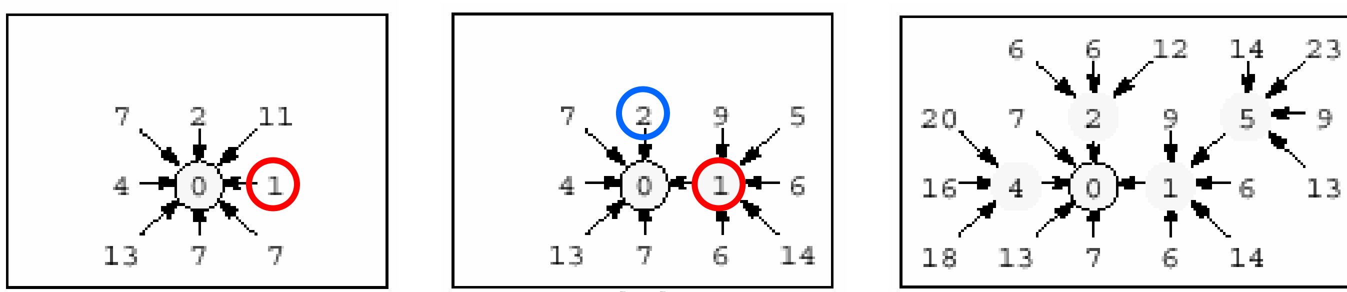
Questions

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

Path Search (basic idea)

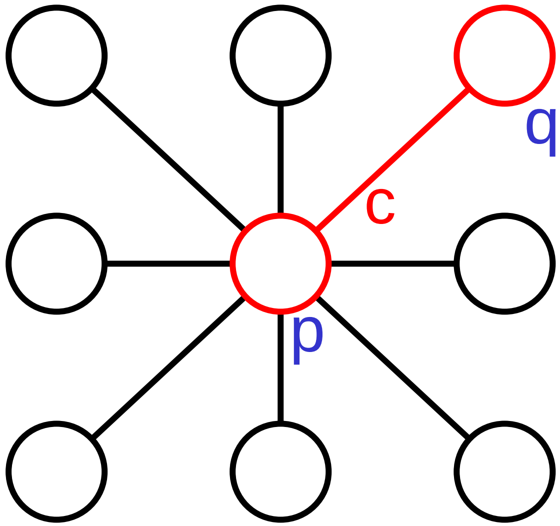
- Graph Search Algorithm
  - Computes minimum cost path from seed to *all other pixels*

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?

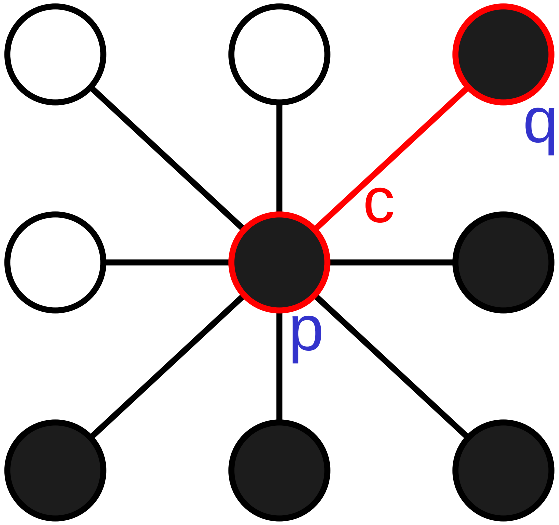
- Treat the image as a graph



- 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

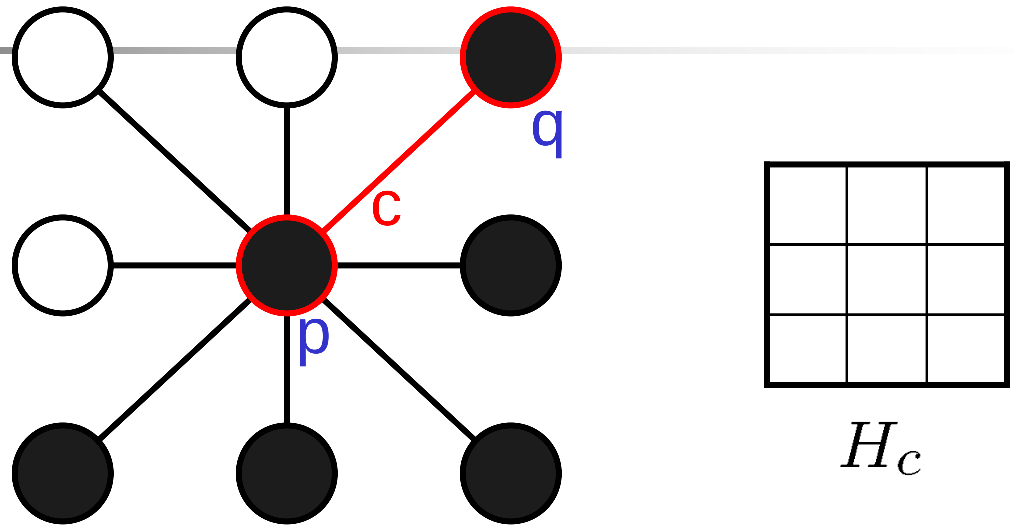
Defining the costs

- Treat the image as a graph



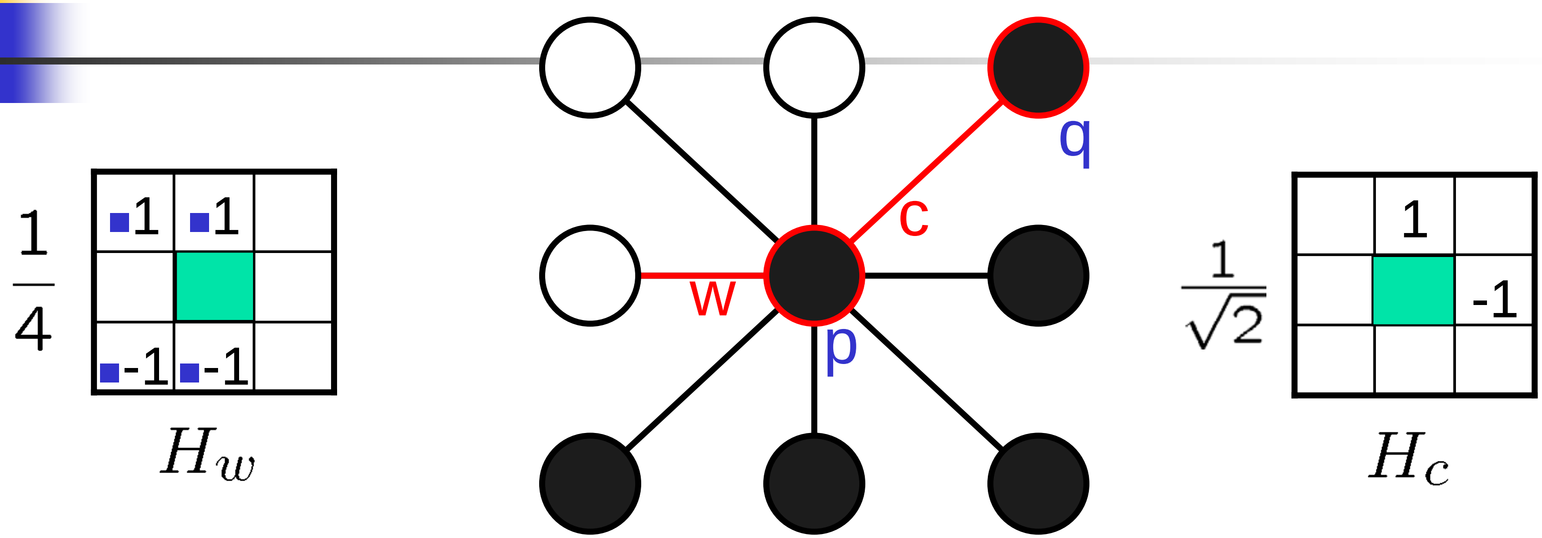
- Want to hug image edges: how to define cost of a link?
    - the link should follow the intensity edge
        - want intensity to change rapidly with respect to the link
    - c ≈ - |difference of intensity with respect to link|

Defining the costs



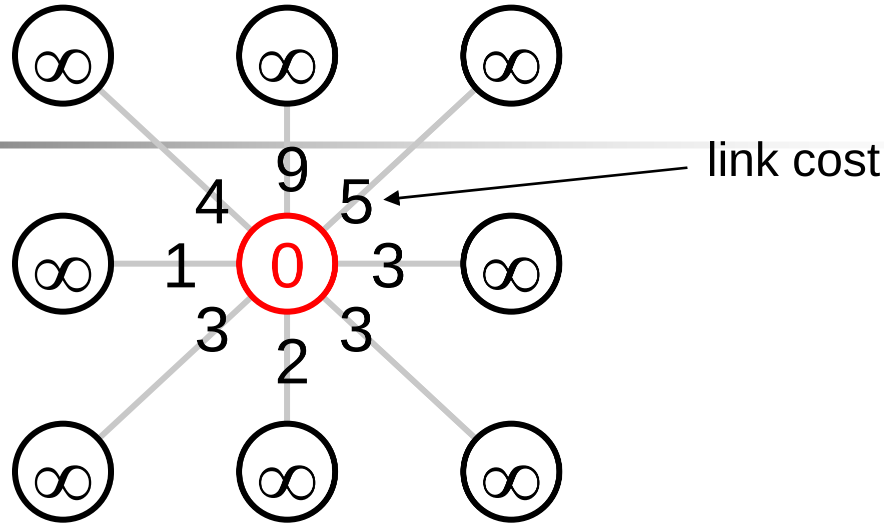
- c can be computed using a cross-correlation filter
  - assume it is centered at p
- Also typically scale c by its length
  - set c = (max-|filter response|)
    - where max = maximum |filter response| over all pixels in the image

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

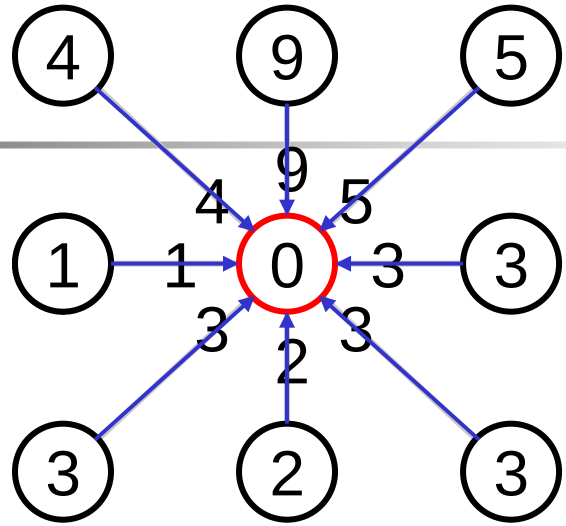


- Algorithm
  1. init node costs to ∞,
  2. Init an active set p = seed point, cost(p) = 0
  3. expand p as follows:

     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

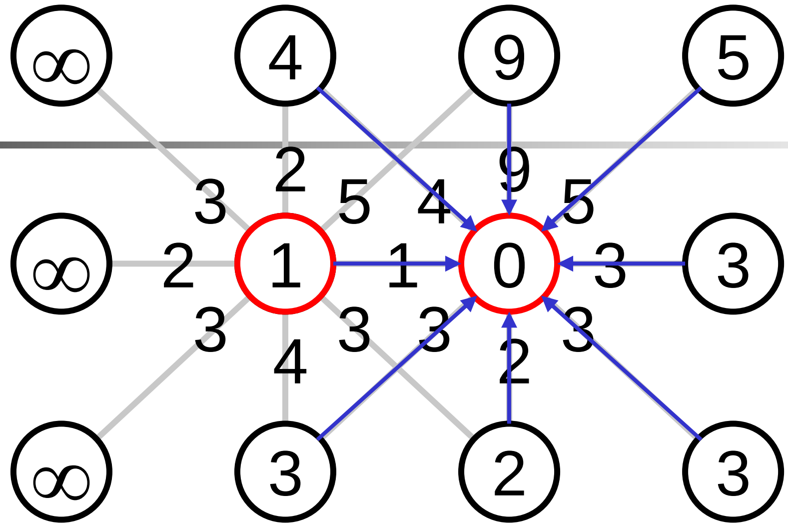


- Algorithm
  1. init active set p = seed point, cost(p) = 0
  2. expand p as follows:

     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

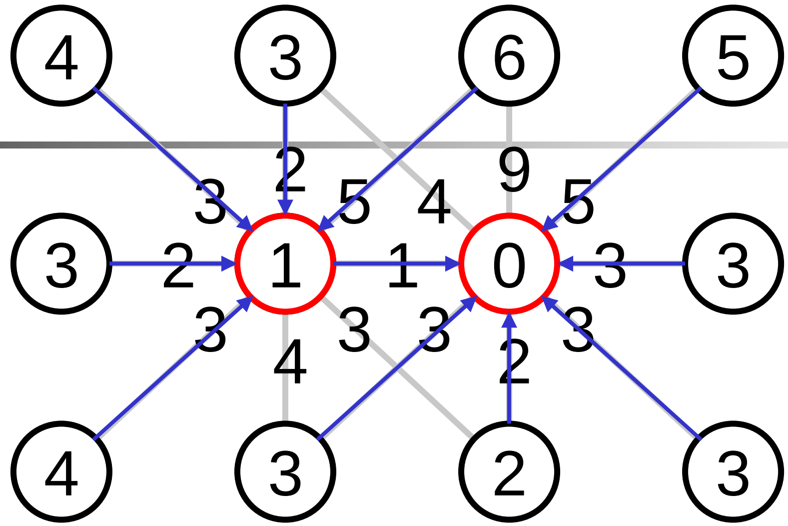


- Algorithm
  1. init node costs to ∞, set p = seed point, cost(p) = 0
  2. expand p as follows:

     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

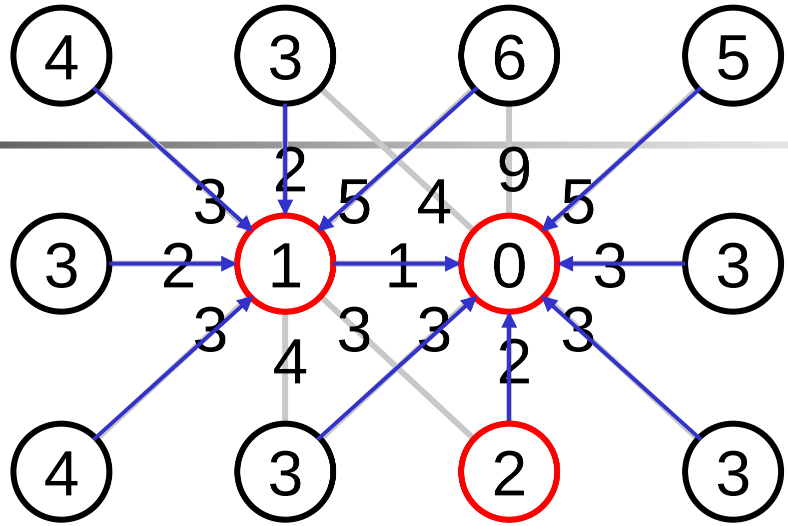
Dijkstra's shortest path algorithm



- 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

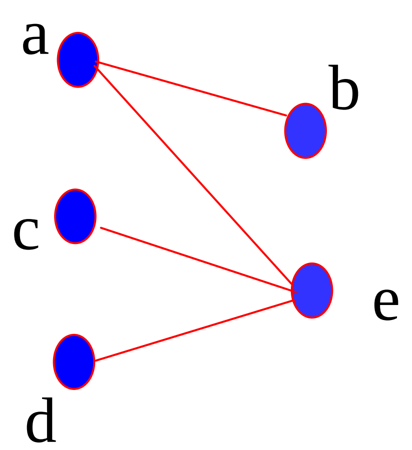
Let’s stop here

- Experience more by yourself…

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

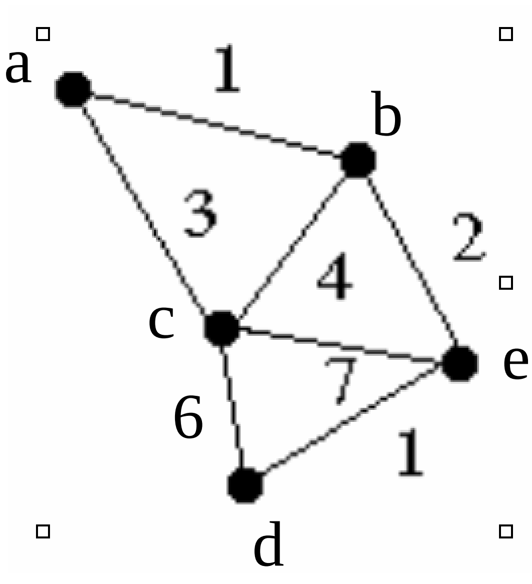
Graphs Representations



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

Adjacency Matrix: W

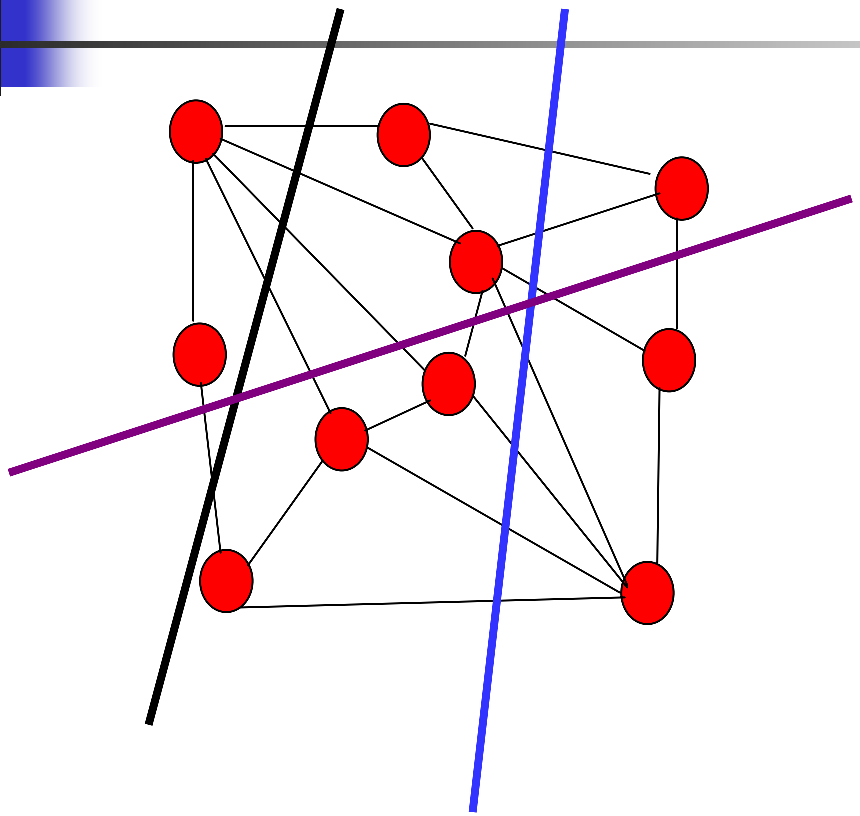
Weighted Graphs



$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 0 & 1 & 3 & 0 & 0 \\ 1 & 0 & 4 & 0 & 2 \\ 3 & 4 & 0 & 6 & 7 \\ 0 & 0 & 6 & 0 & 1 \\ 0 & 2 & 7 & 1 & 0 \end{bmatrix} \end{array}$$

Weight Matrix: W

Minimum Cut



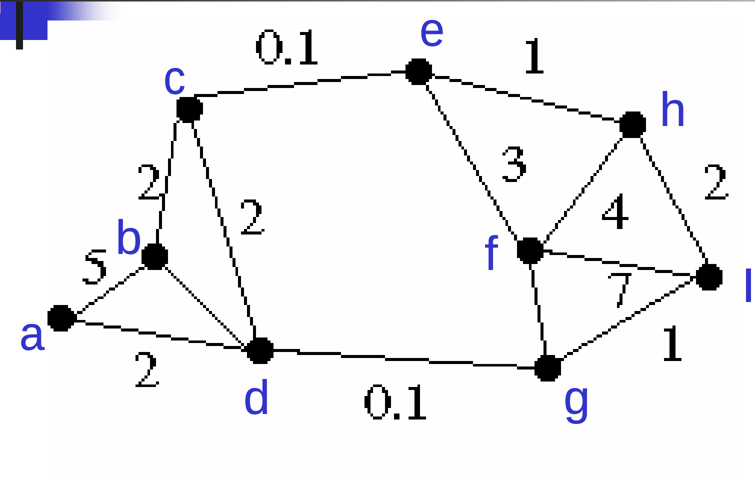
A cut of a graph G is the set of edges S such that removal of S from G disconnects G.

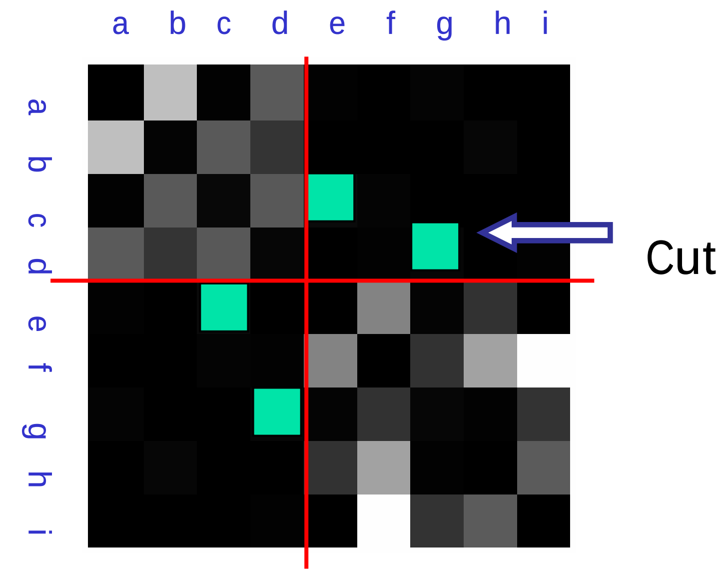
Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as

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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

Minimum Cut and Clustering





$$\begin{array}{c|ccccccccc} & a & b & c & d & e & f & g & h & i \\ \hline a & 0 & 5 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ b & 5 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ c & 0 & 2 & 0 & 2 & 0.1 & 0 & 0 & 0 & 0 \\ d & 2 & 1 & 2 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ e & 0 & 0 & 0.1 & 0 & 0 & 3 & 0 & 1 & 0 \\ f & 0 & 0 & 0 & 0 & 3 & 0 & 1 & 4 & 7 \\ g & 0 & 0 & 0 & 0.1 & 0 & 1 & 0 & 0 & 1 \\ h & 0 & 0 & 0 & 0 & 1 & 4 & 0 & 0 & 2 \\ i & 0 & 0 & 0 & 0 & 0 & 7 & 1 & 2 & 0 \end{array}$$

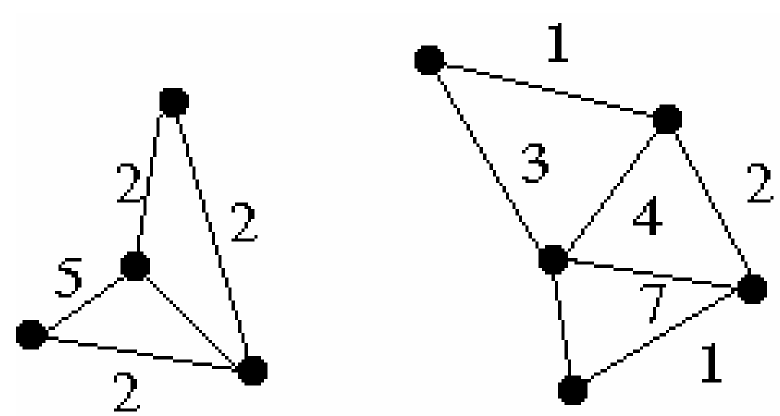
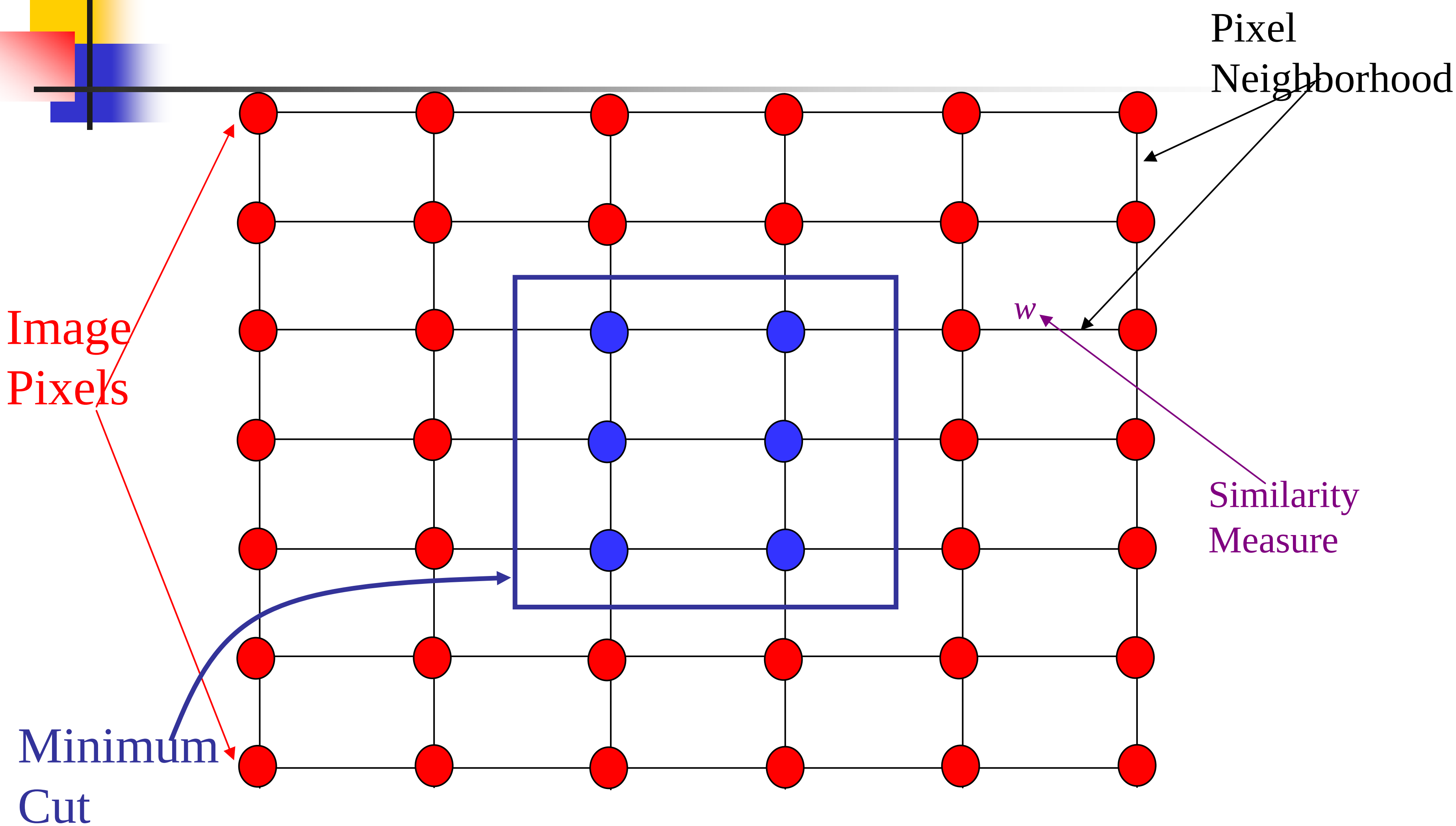
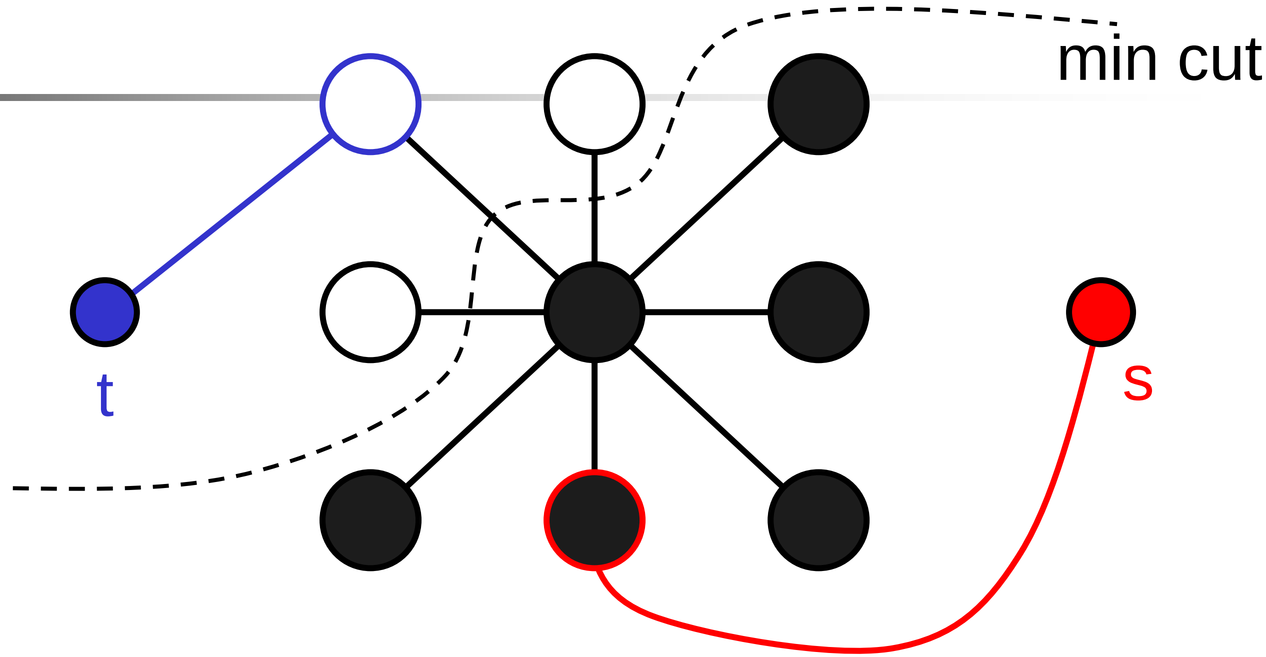


Image Segmentation & Minimum Cut

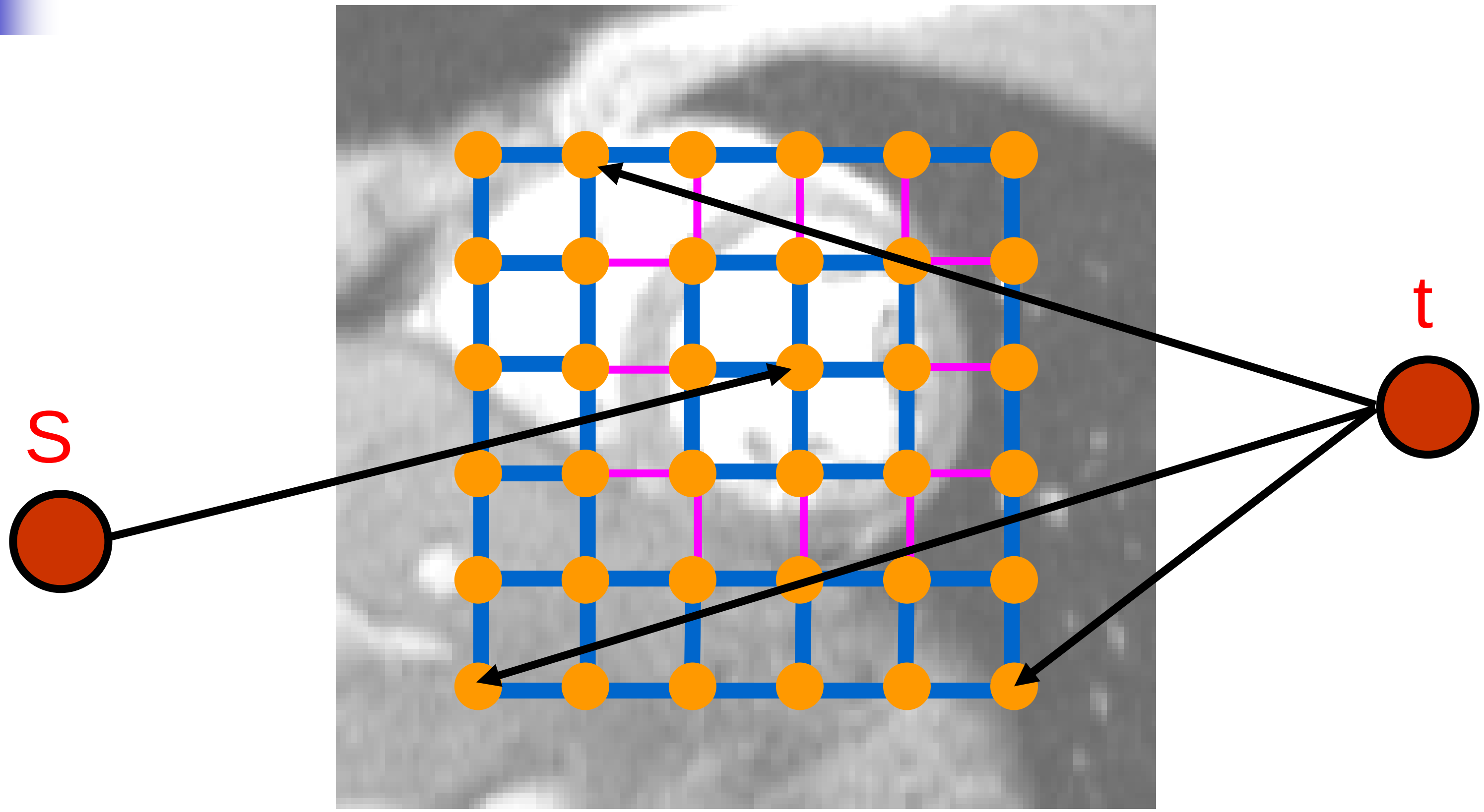


Segmentation by min (s-t) cut [Boykov 2001]

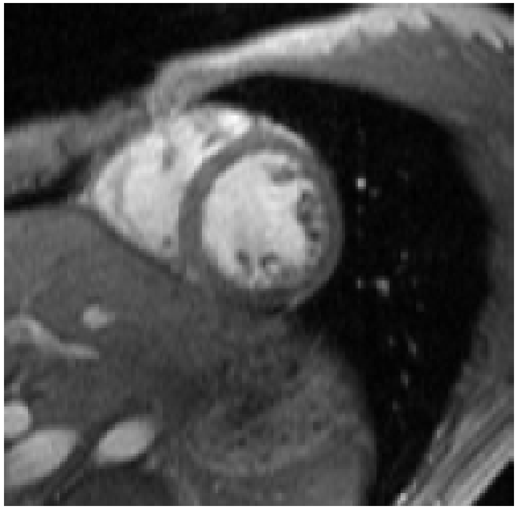


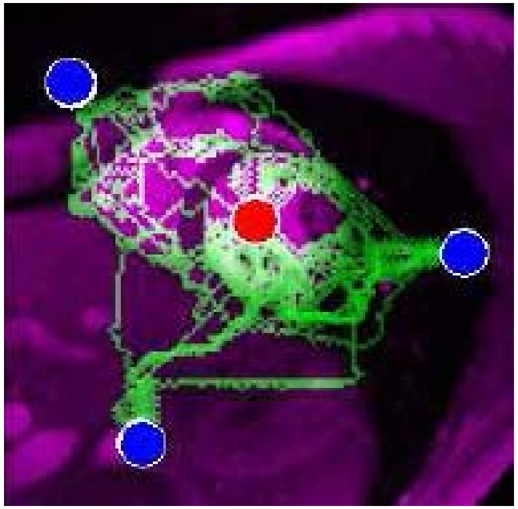
- Graph
  - node for each pixel, link between pixels
  - specify a few pixels as foreground and background
    - create an infinite cost link from each bg pixel to the “t” node
    - create an infinite cost link from each fg pixel to the “s” node
  - compute min cut that separates s from t
  - how to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

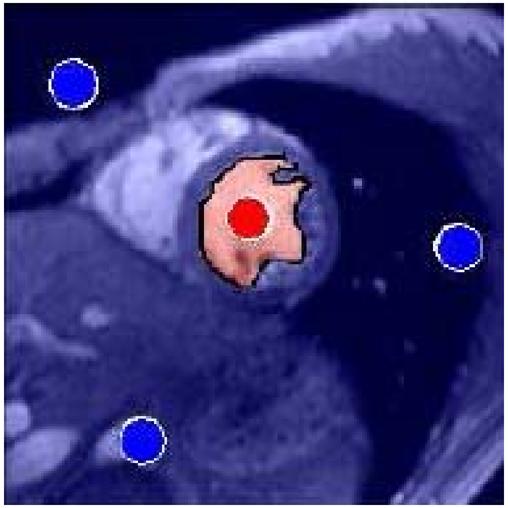
S-T Min-Cut/Max Flow



S-T Min-Cut/Max Flow





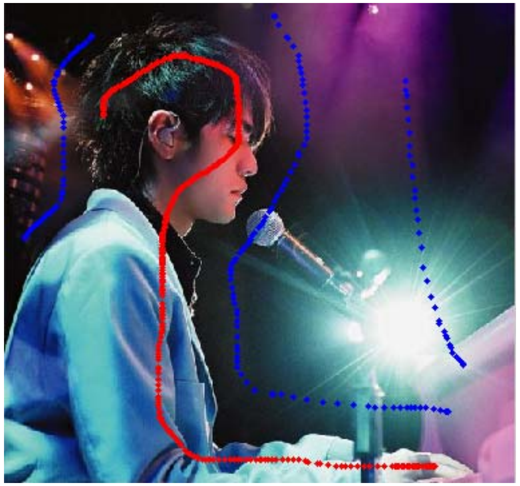


Welcome to the world of image segmentation!

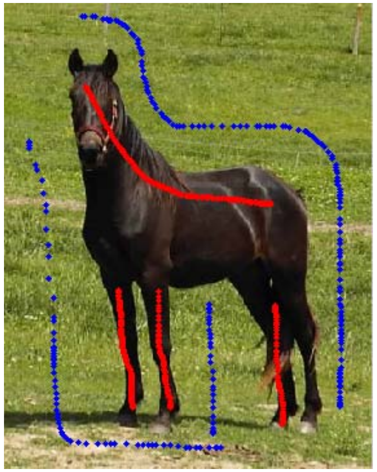
(A student's report)

漆黑中的追迹者

更多样例（1/3)

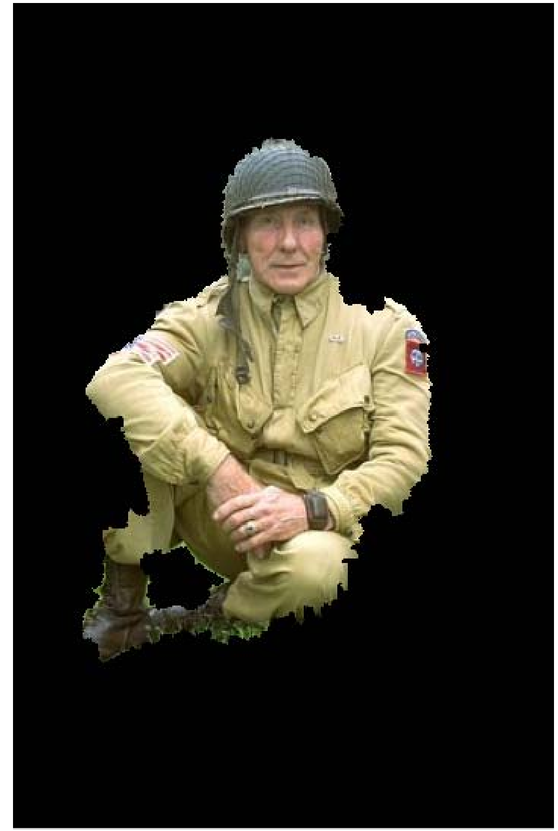


更多样例 （2/3)



更多样例 （3/3)



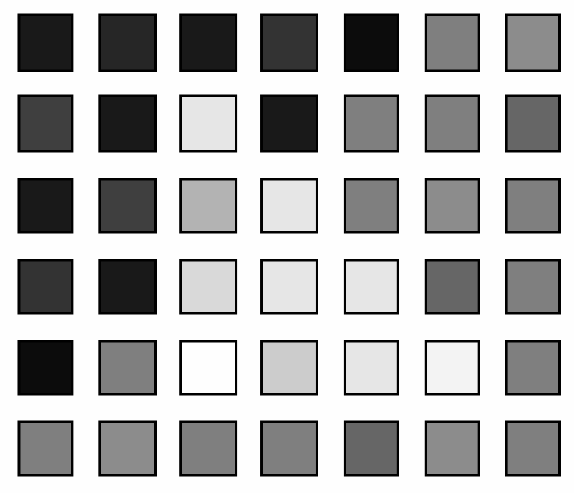


内容安排

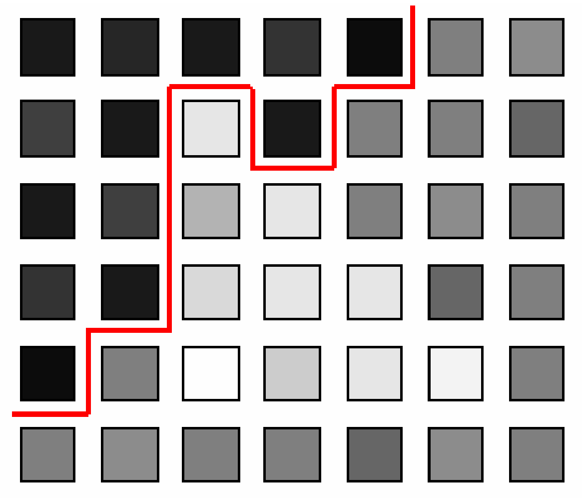
- 1.大致的想法 （3星）
  - 背景知识：流网络
- 2.算法流程
  - 彩色图到灰度图转换 （1星）
  - 预处理：分水岭算法 （2星）
  - GraphCut方法 （4星）
- 3.一些例子

大致的想法 (1/2)

- 要解决的问题是什么？



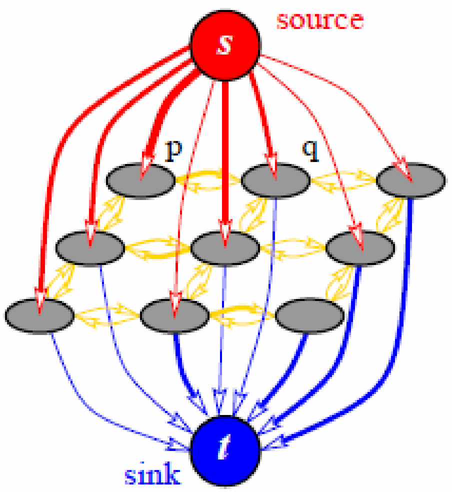
一幅灰度图



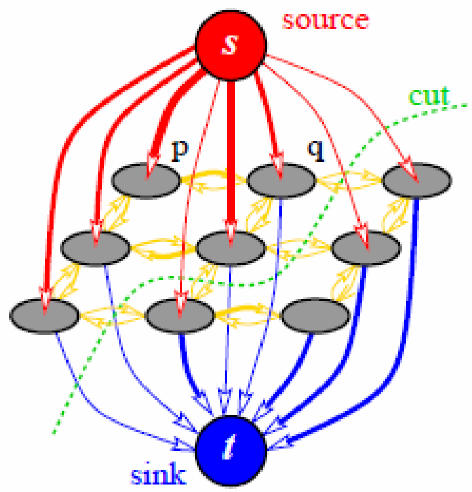
对应的一种分割

大致的想法 (2/2)

- 解决方案：抽象为图论模型



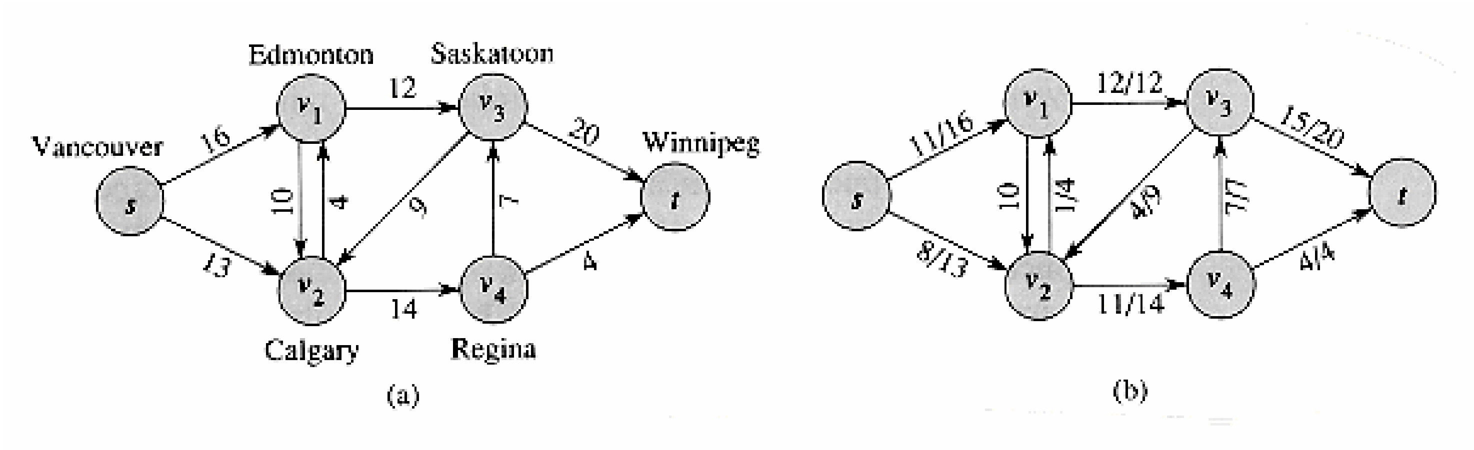
(a) A graph $\mathcal{G}$



(b) A cut on $\mathcal{G}$

背景知识：流网络（1/2）

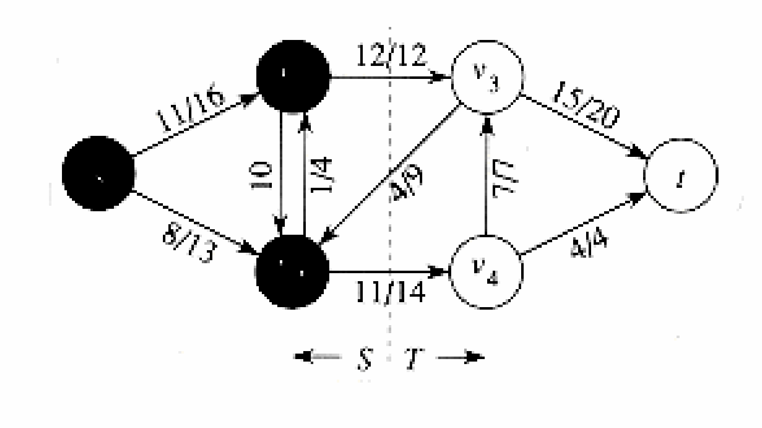
流网络（Flow Networks)模型：



流网络　　　　流网络中的流

- 流的值：从源点出发的总流

背景知识：流网络 （2/2）

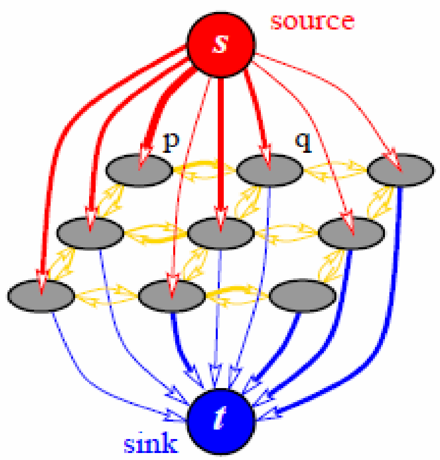


割：一个割将顶点集划分为S和T两部分

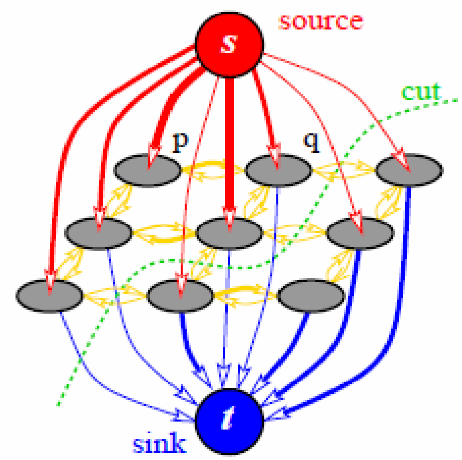
割的容量：割的从S到T的边集的容量和

最大流最小割定理：最大流的值=最小割的容量

最终想法



(a) A graph $\mathcal{G}$



(b) A cut on $\mathcal{G}$

算法流程 Step By step

彩色图像灰度化

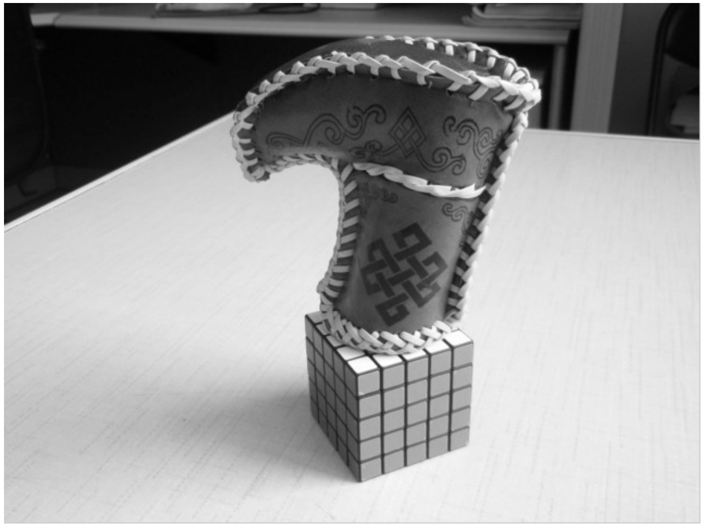
用Watershed进行区算法对图像域标记

把区域当做大像素，运行GraphCut算法

彩色图像灰度化　step1

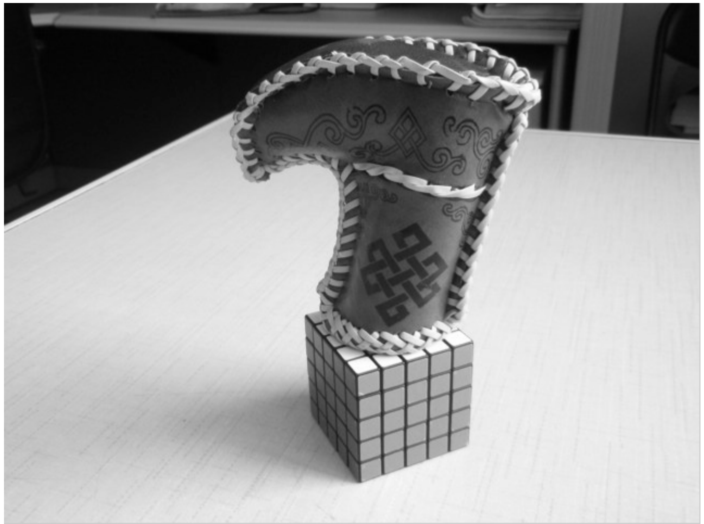
- I = imread(ImageName); Ig = rgb2gray(I);

原图像

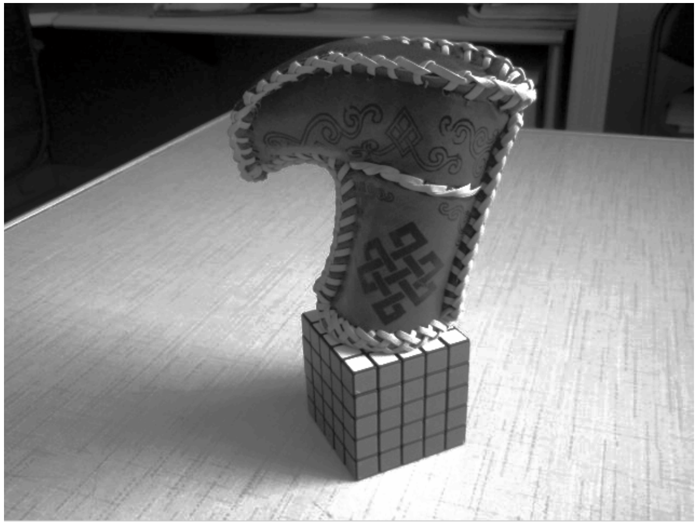


灰度图

直方图均衡化



灰度图



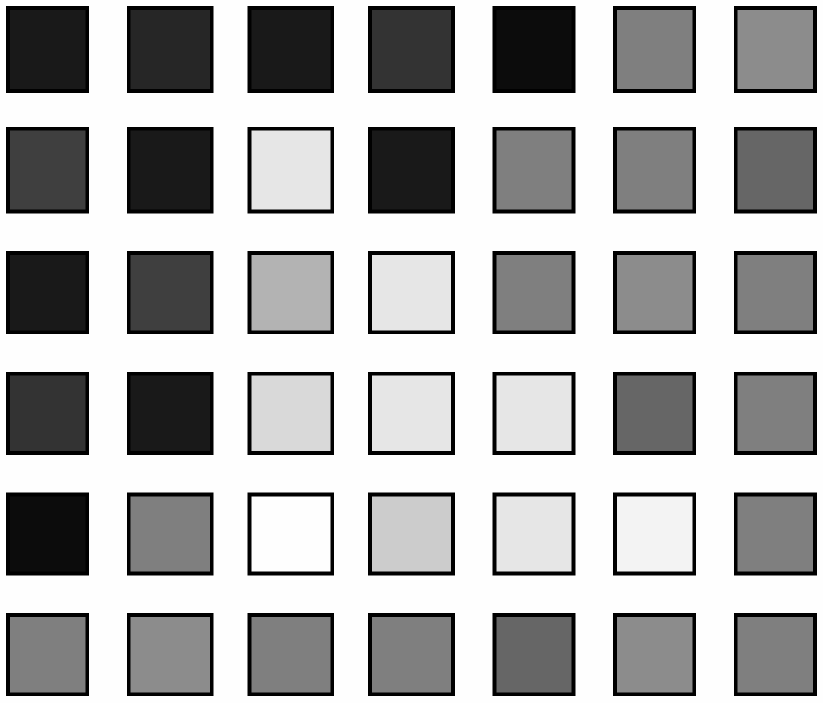
直方图均衡化后结果

分割结果的差异

未经直方图均衡化

经过直方图均衡化

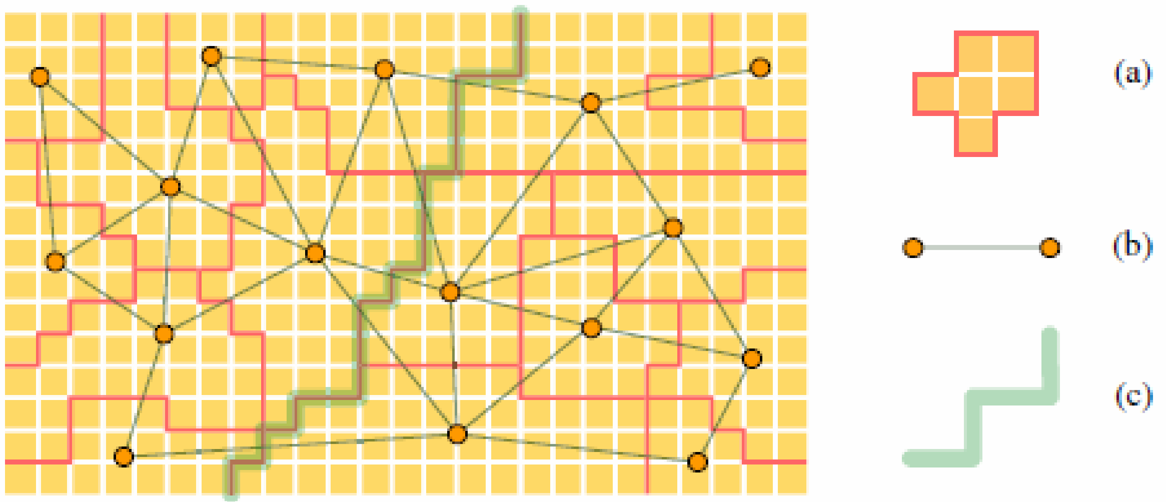
得到的结果:



一幅灰度图

分水岭算法 step2

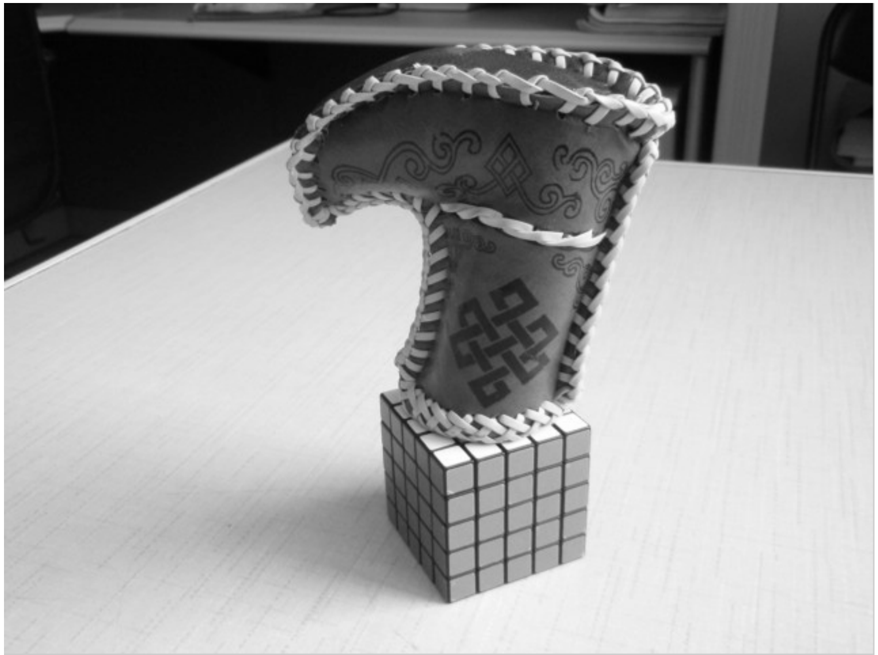
- 目的：以像素块为整体



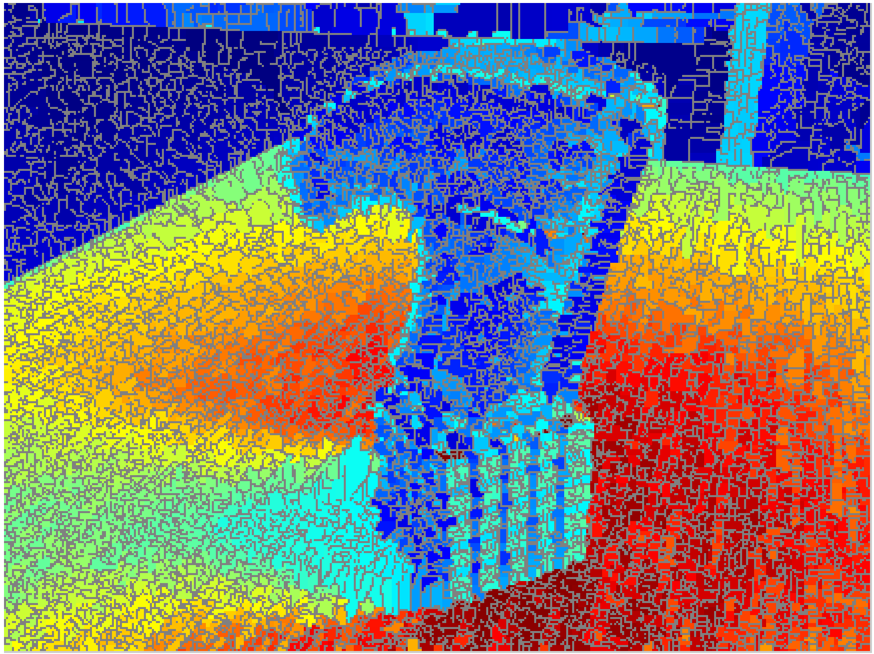
(a) A small region by the pre-segmentation. (b) The nodes and edges for the graph cut algorithm with pre-segmentation. (c) The boundary output by the graph cut segmentation.

分水岭算法

- 将灰度图像按灰度划分为多个区域



灰度图



Watershed算法处理结果

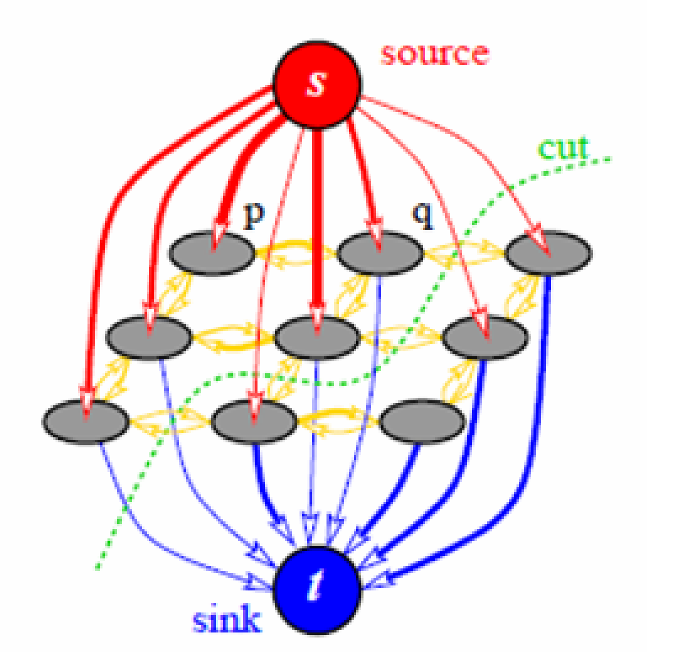
分水岭算法的结果

- 分水岭算法的结果：L = watershed(Ig);

1	1	0	2	2	2	2	0	5	5
1	1	0	2	2	2	2	0	0	0
1	1	1	0	2	2	0	6	6	0
0	0	0	0	2	0	0	0	6	0
8	8	8	0	2	0	7	7	0	7
8	8	0	4	0	3	0	7	7	7
8	8	0	0	0	3	3	0	7	7
8	8	8	8	8	0	3	3	0	7

分水岭算法的结果L数组

Graphcut方法 step



- 能量函数：

$$E(X) = \sum_{i \in \mathcal{V}} E_1(x_i) + \lambda \sum_{(i,j) \in \mathcal{E}} E_2(x_i, x_j)$$

- 权值安排：

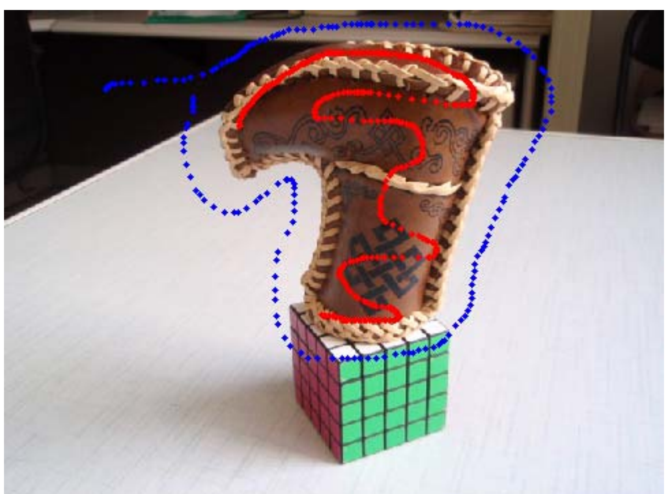
$$E_1(x_i = S) = \infty \qquad E_1(x_i = T) = 0 \qquad \forall i \in F$$

$$E_1(x_i = S) = 0 \qquad E_1(x_i = T) = \infty \qquad \forall i \in B$$

$$E_1(x_i = S) = \frac{d_i^B}{d_i^F + d_i^B} \qquad E_1(x_i = T) = \frac{d_i^F}{d_i^F + d_i^B} \qquad \forall i \in U$$

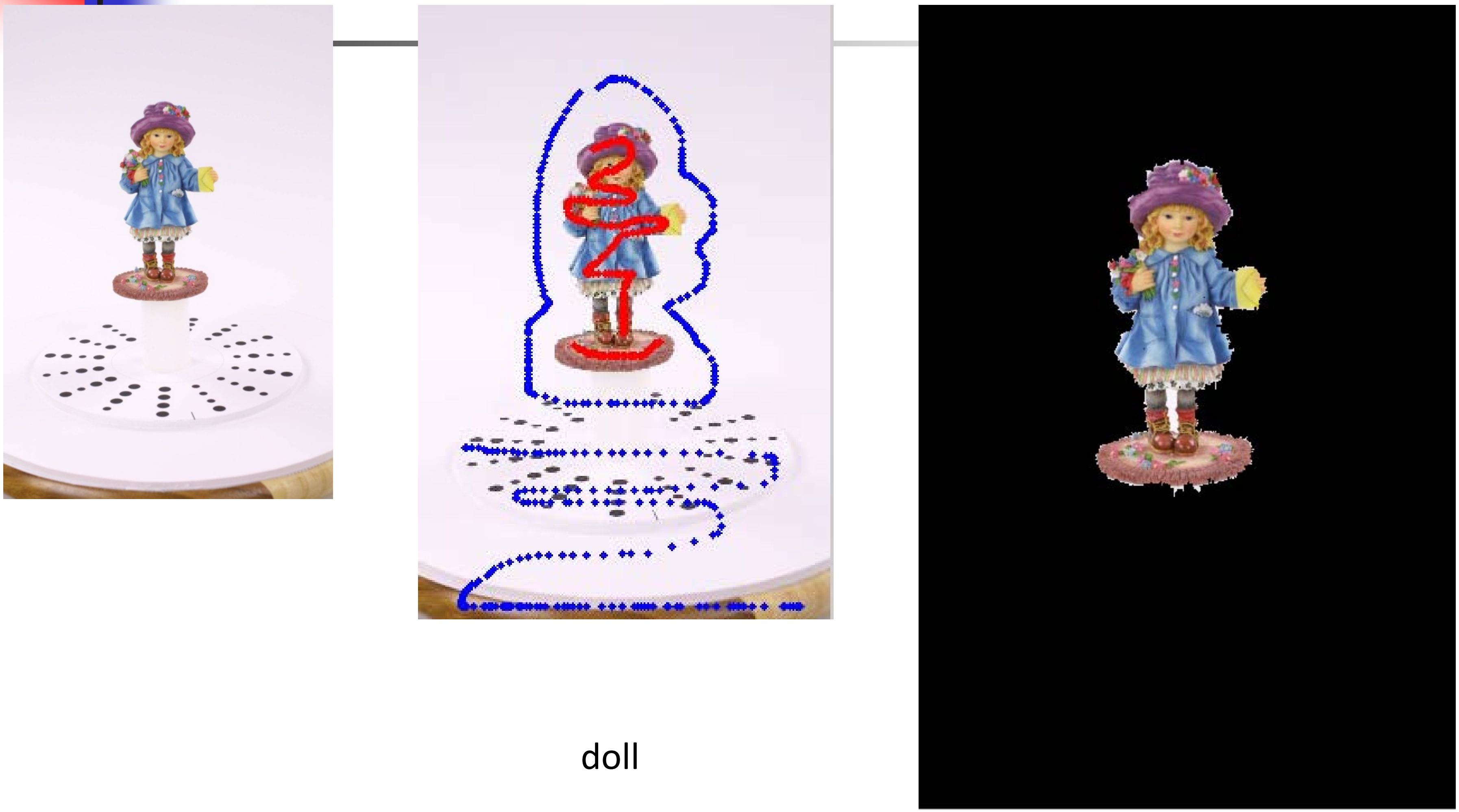
$$E_2(x_i, x_j) = |x_i - x_j| \cdot \frac{1}{1 + \|C(i) - C(j)\|^2}$$

更多样例 （1/4)



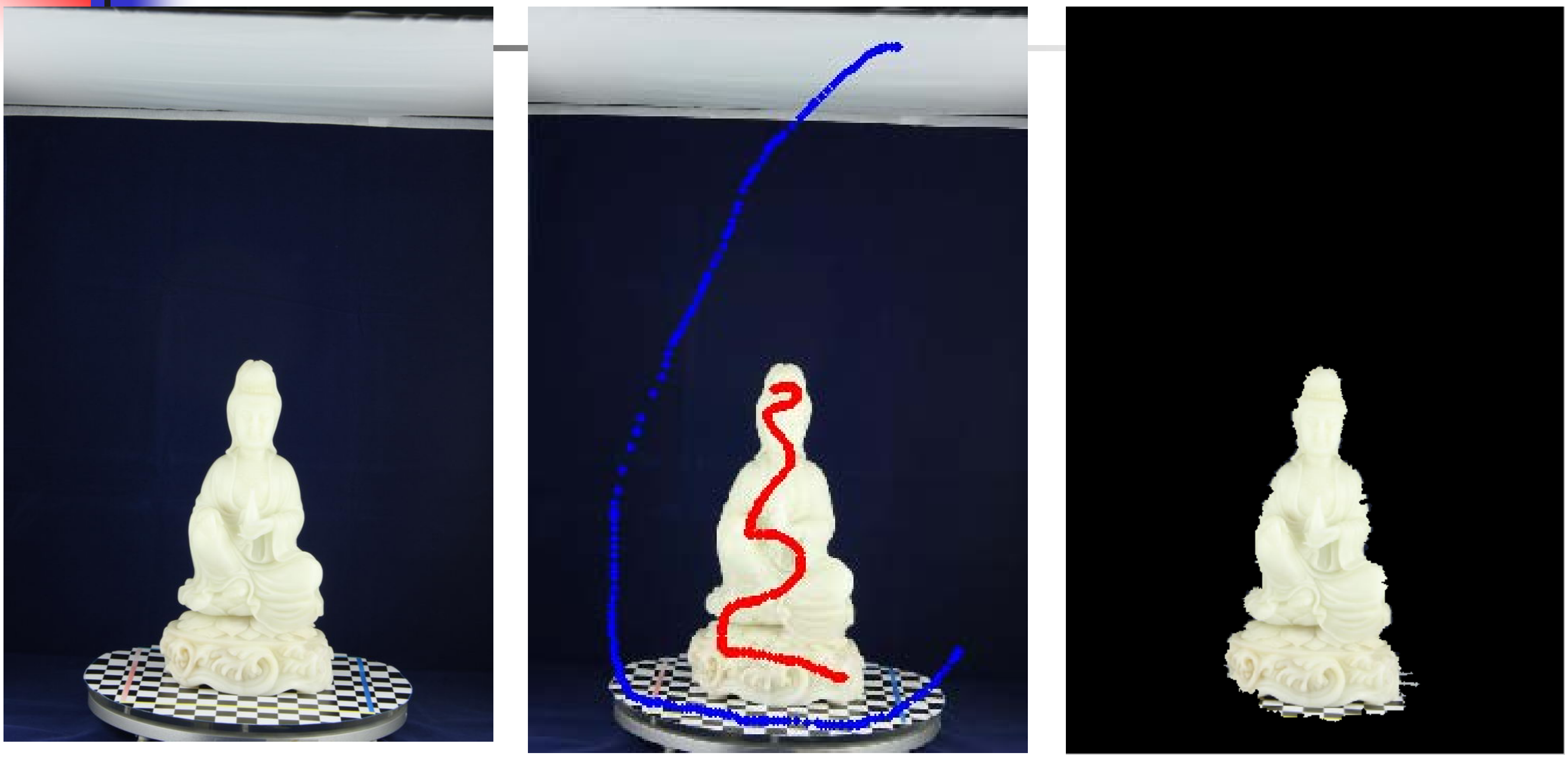
boot

更多样例（2/4)



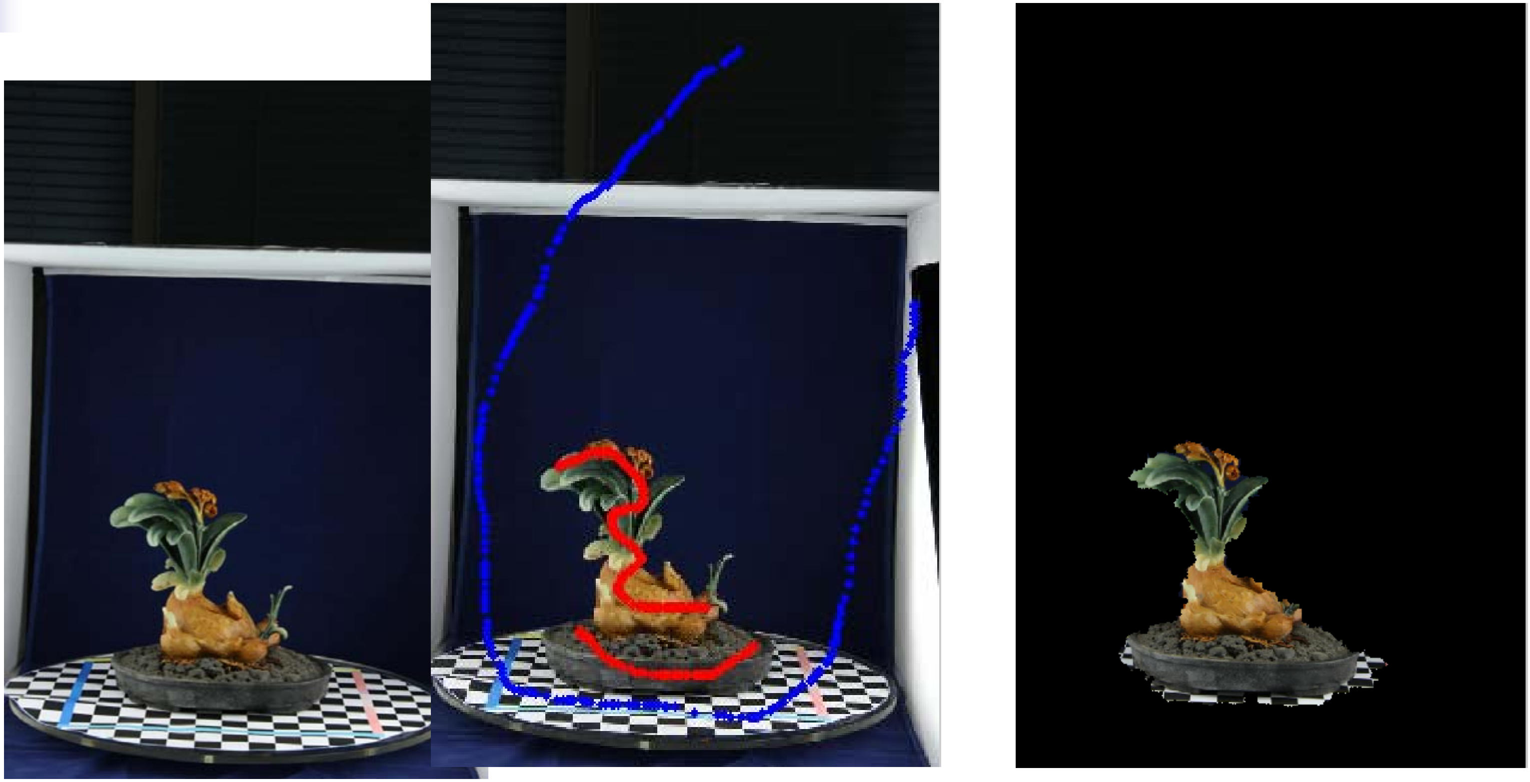
doll

更多样例（3/4)



guanyin

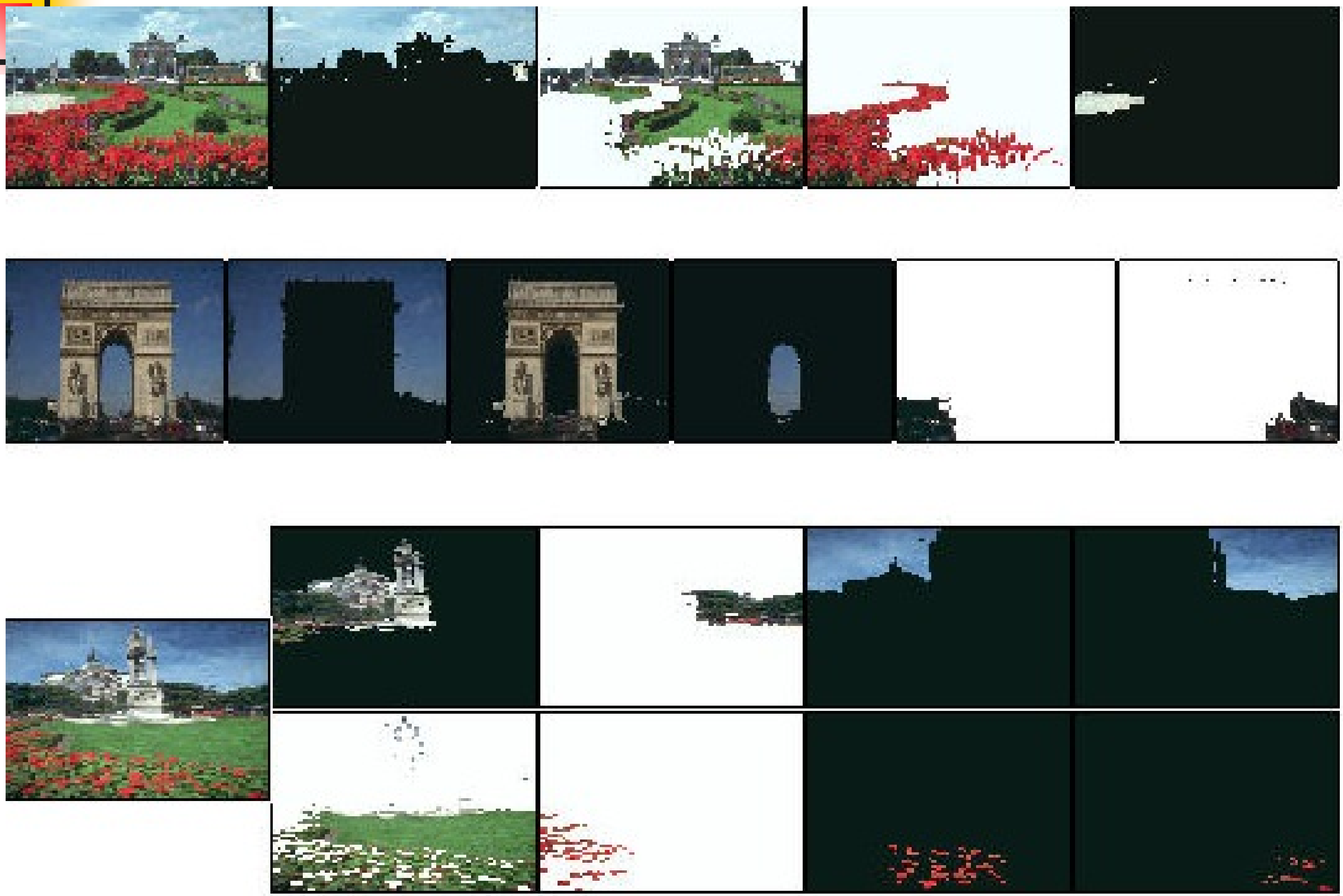
更多样例（4/4)



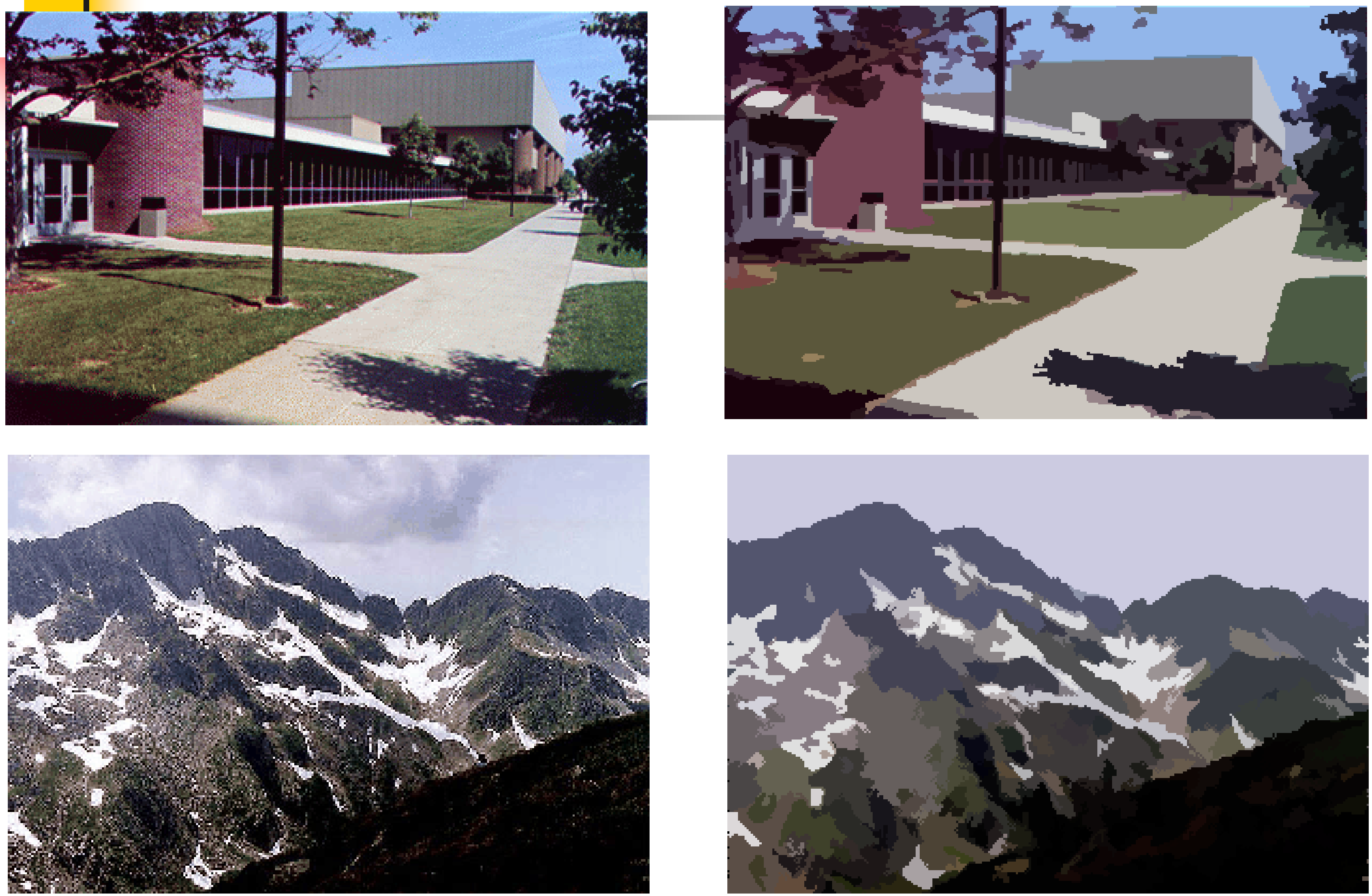
bonsai

Grabcut [Rother et al., SIGGRAPH 2004]

Color Image Segmentation

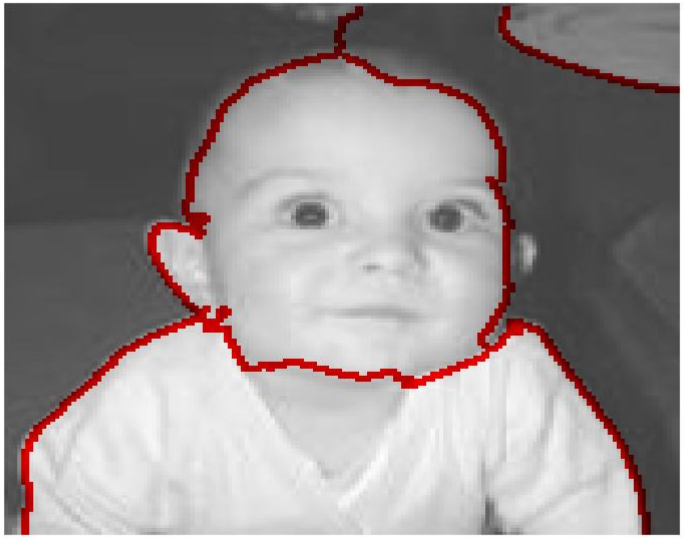


Mean Shift Segmentation



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

Normalized Cuts



Efficient Graph-Based Image Segmentation

Segmentation parameters: sigma = 0.5, K = 500, min = 50.

Segmentation parameters: sigma = 0.5, K = 1000, min = 100.

References

- Mortensen and Barrett, "Intelligent Scissors for Image Composition," Proc. *SIGGRAPH* 1995.
- Shi and Malik, "Normalized Cuts and Image Segmentation," Proc. CVPR 1997.
- Comaniciu and Meer, "Mean shift analysis and applications," Proc. *ICCV* 1999.
- Felzenszwalb and Huttenlocher. Efficient graph-based image segmentation. IJCV, 59(2):167–181, 2004.
- Boykov and Jolly, "Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images," Proc. ICCV, 2001.
- Rother, Kolmogorov, and Blake, "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph., 23(3):309–314, 2004.

Applications

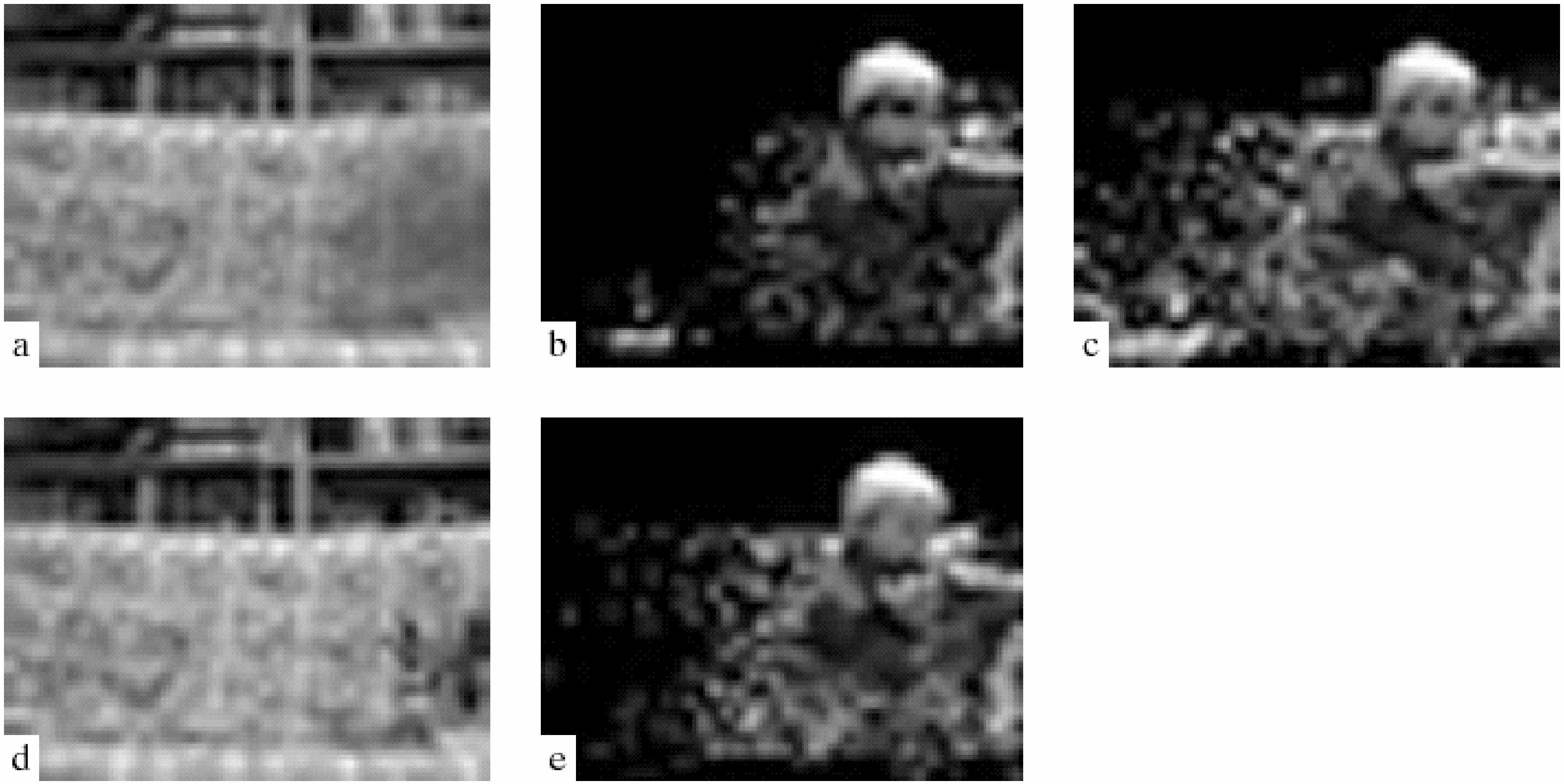
Shot Boundary Detection

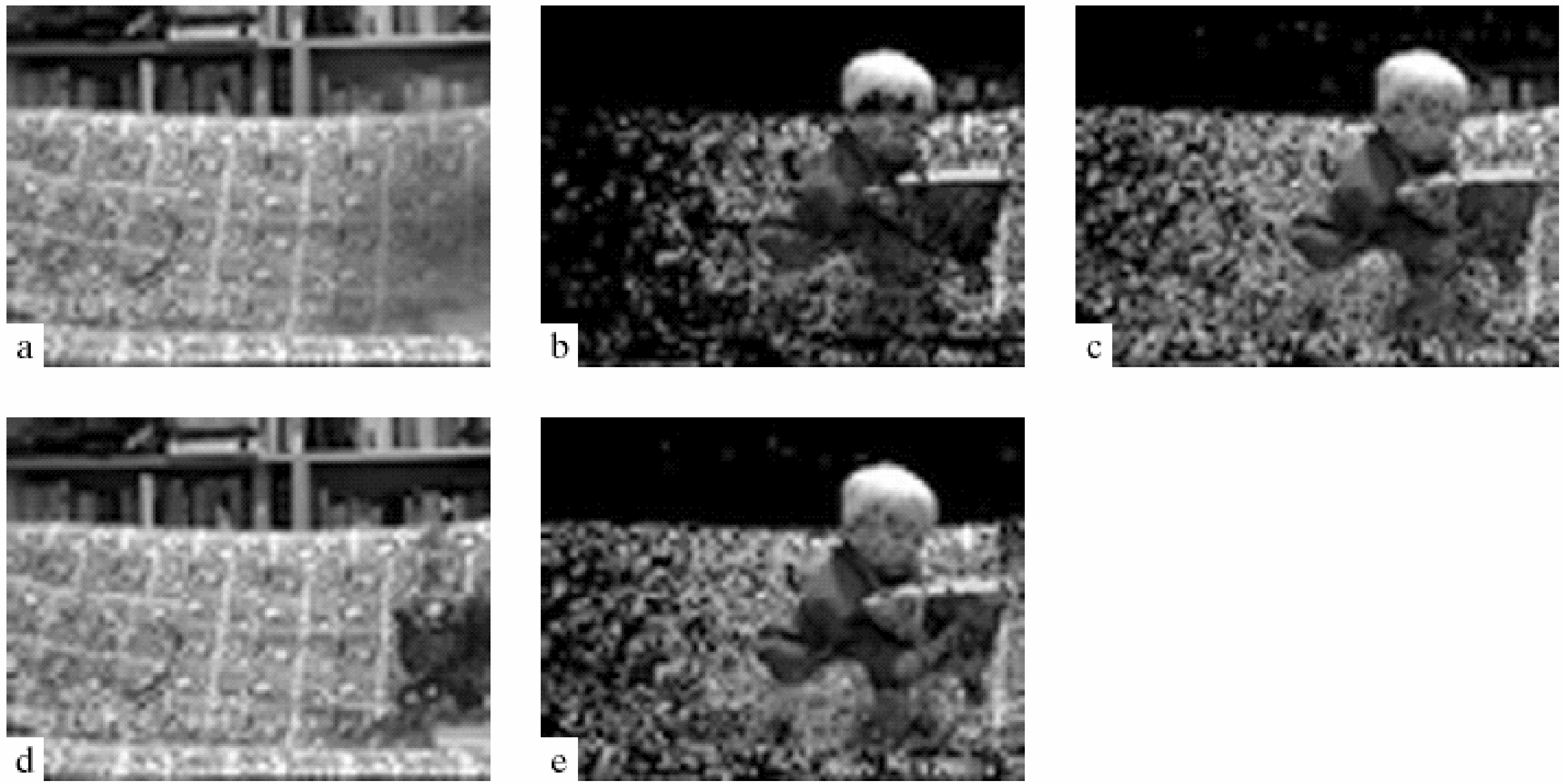
- Find the shots in a sequence of video
  - shot boundaries usually result in big differences between succeeding frames
- Strategy:
  - compute inter-frame distances
  - declare a boundary where these are big
- Possible distances
  - frame differences
  - histogram differences
  - block comparisons
  - 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.

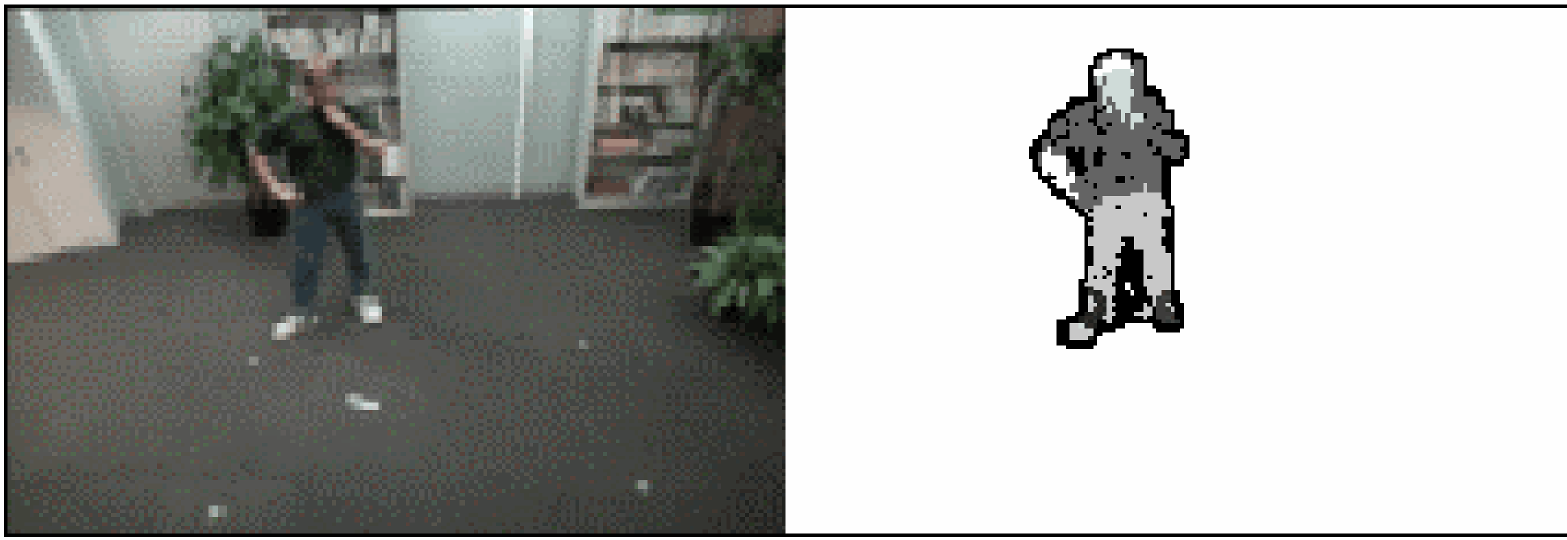
Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

Static Background Modeling Examples

[MIT Media Lab Pfinder / ALIVE System]

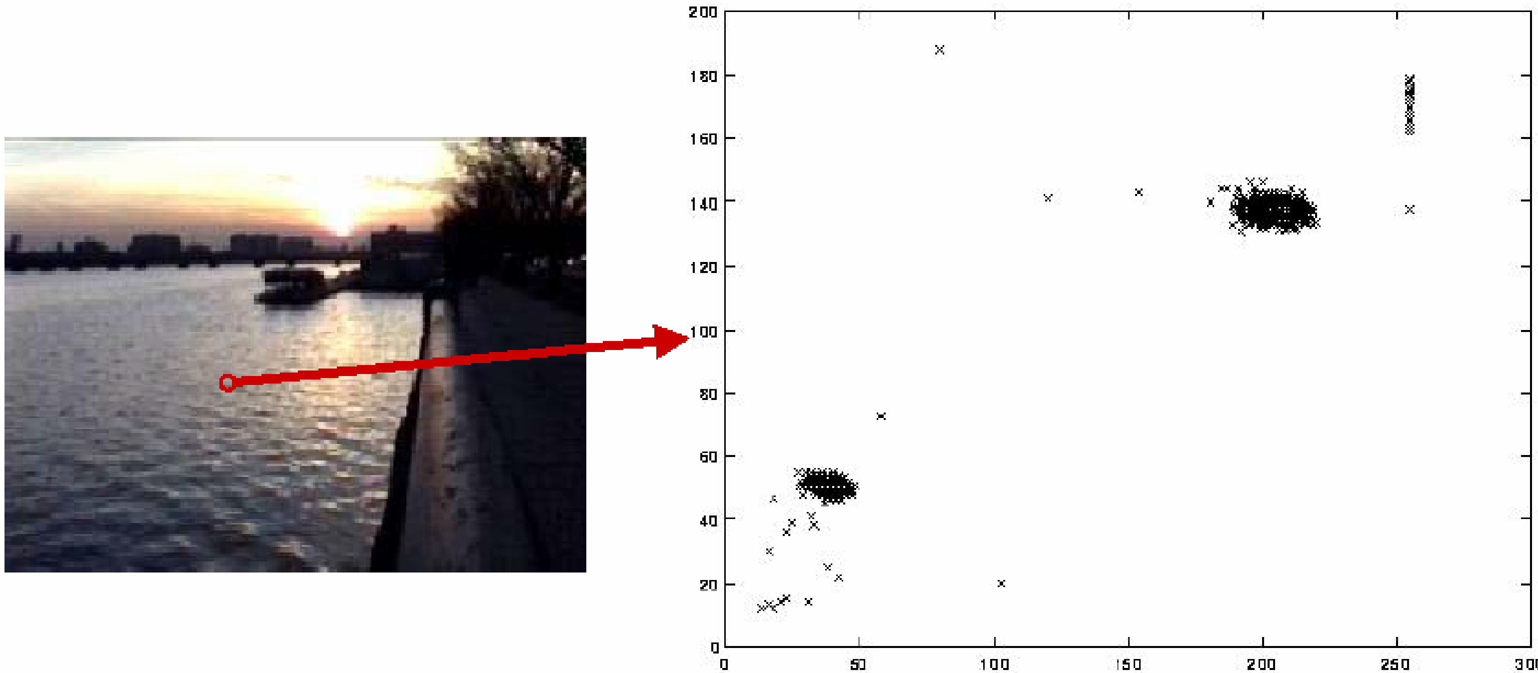
Static Background Modeling Examples



[MIT Media Lab Pfinder / ALIVE System]

Dynamic Background

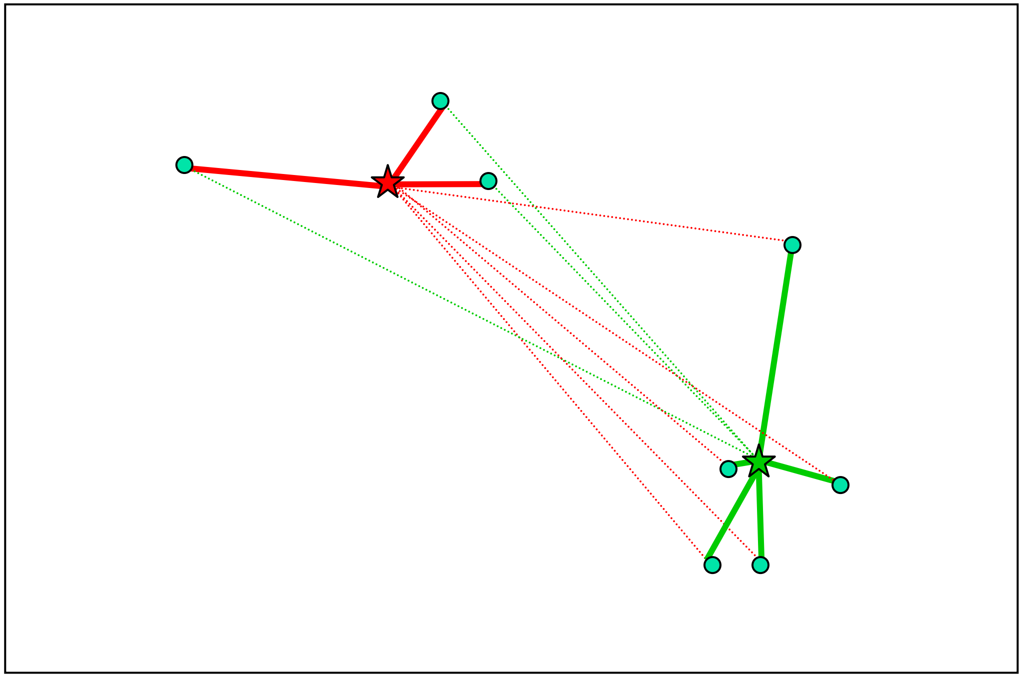
BG Pixel distribution is non-stationary:



Appendix

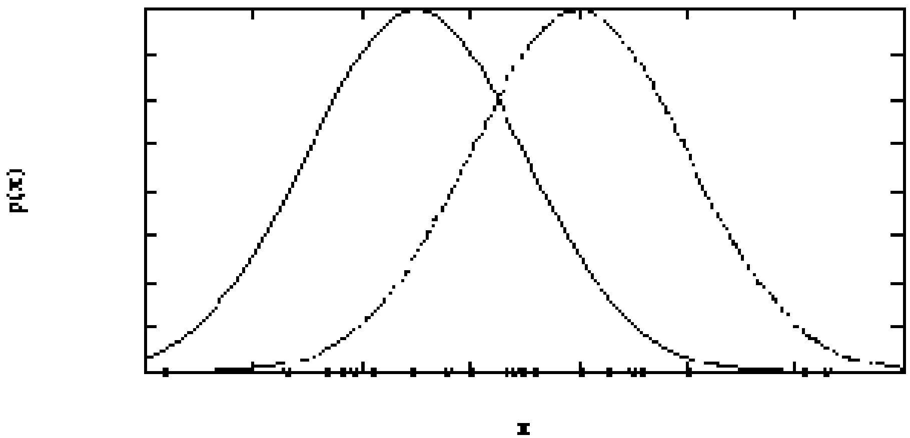
- Expectation-Maximization

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

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

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

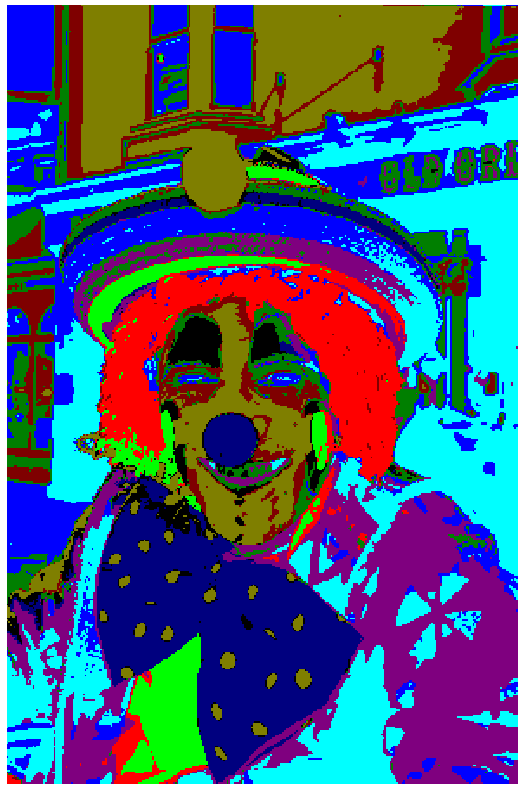
M-Step

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

Generalized K-Means

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

EM Clustering: Results

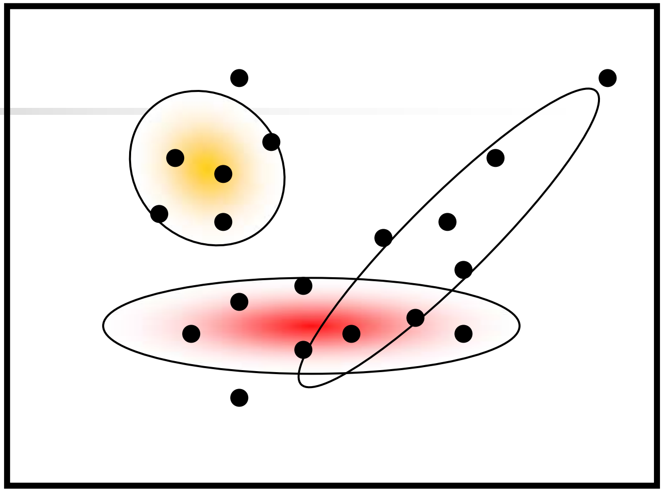


Probabilistic clustering

- 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

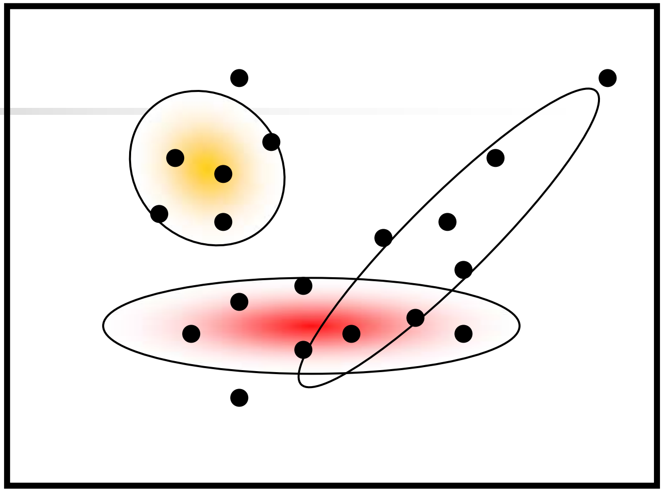


- One generative model is a mixture of Gaussians (MOG)
  - K Gaussian blobs with means μ_b covariance matrices $\mathbf{V}_b$, dimension d
    - blob *b* defined by: $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - blob *b* is selected with probability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

* Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003

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

$$W = N \times N \text{ symmetric matrix, where}$$

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

$$\|F_i - F_j\| = \text{Image feature similarity}$$

$$\|X_i - X_j\| = \text{Spatial Proximity}$$

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

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

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

- 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

The famous invisible dog eating under a tree: