Image Segmentation

Recap from last time

- Samples not squares
- Sensors are not perfect
- Quantization hurts
- Questions?

Overview

- What is image segmentation?
- Thresholding and thresholding algorithms
- Performance and the ROC curve
- Connectivity and connected components
- Region growing
- Split and merge algorithms
    - The region adjacency graph


©1994 Encyclopaedia Britannica, Inc.

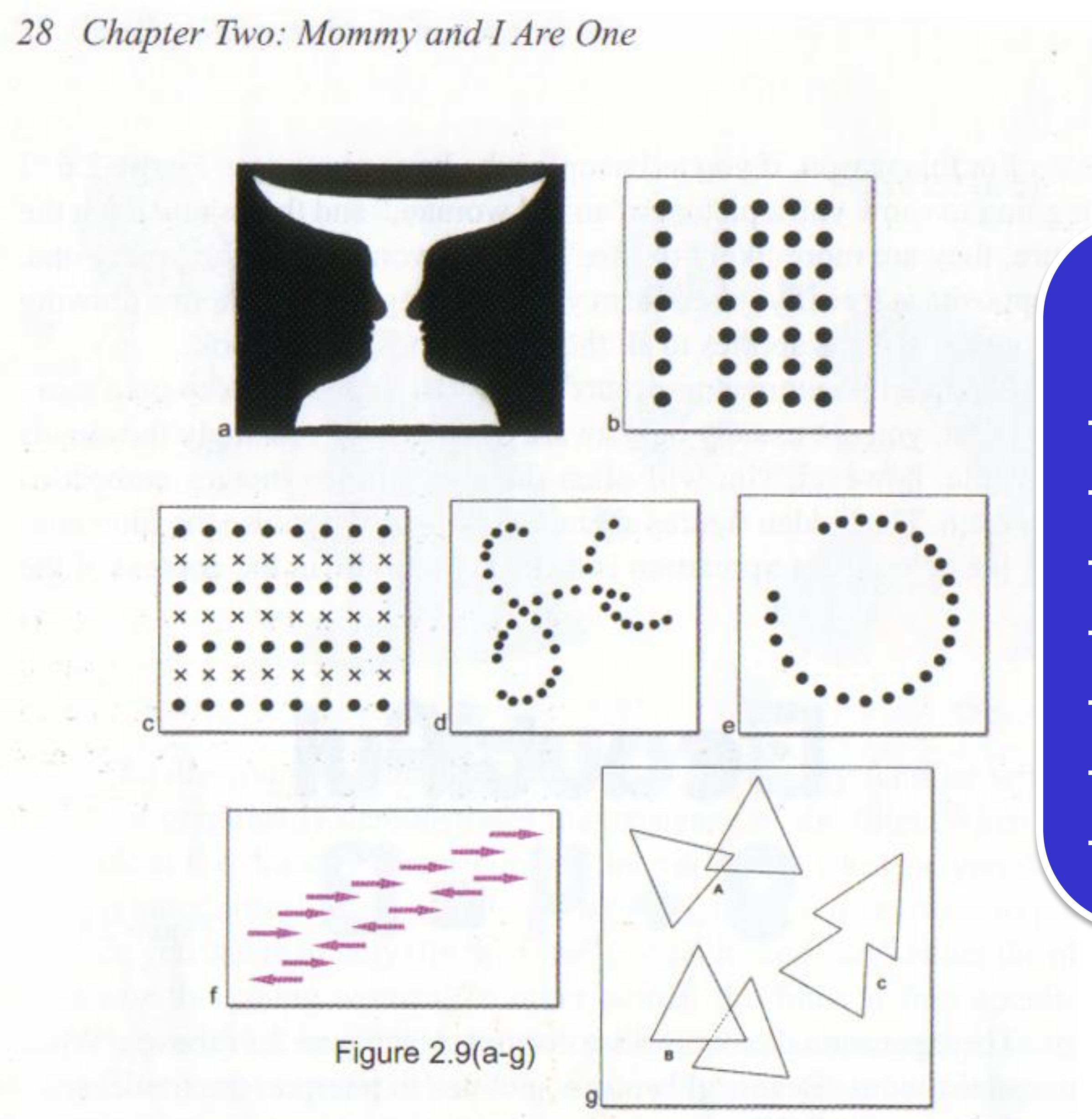


Figure 2.9(a-g)

Gestalt Phenomena

-Figure-ground
-Proximity
-Similarity
-Continuation
-Closure
-Common fate
-Symmetry

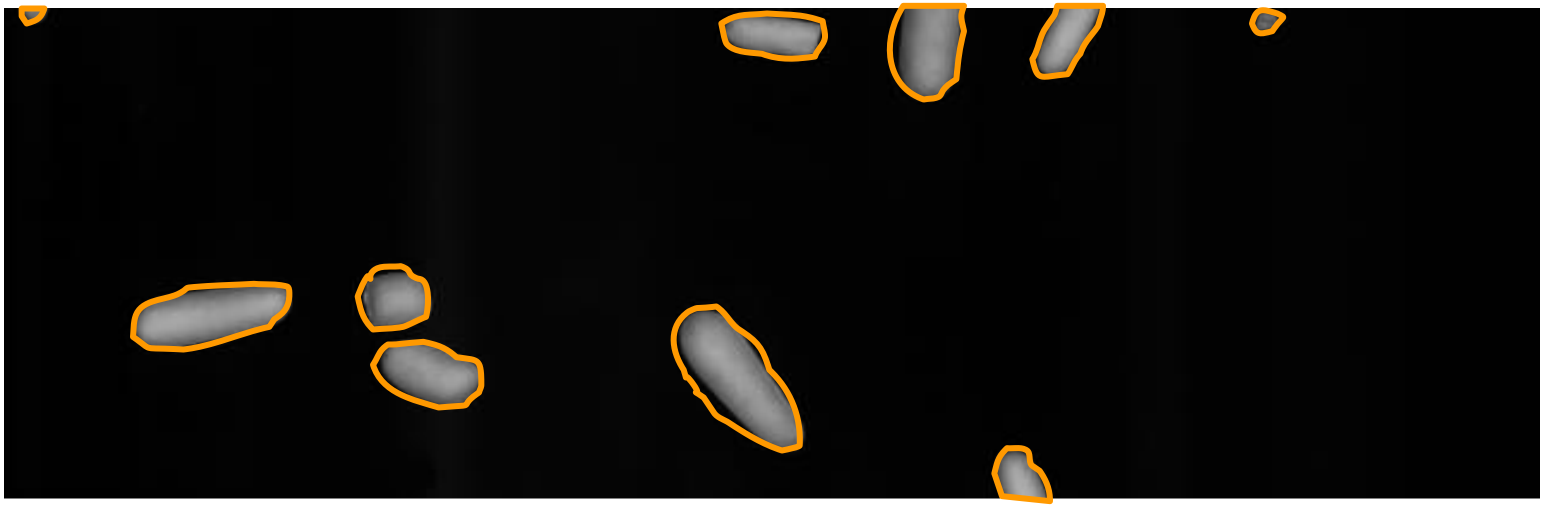
What is Image Segmentation?

- It partitions an image into regions of interest.

- The first stage in many automatic image analysis systems.
- A *complete segmentation* of an image I is a finite set of regions $R_1, \ldots, R_N$, such that

$$I = \bigcup_{i=1}^{N} R_i \text{ and } R_i \cap R_j = \phi \; \forall \, i \neq j.$$

How should I segment this?



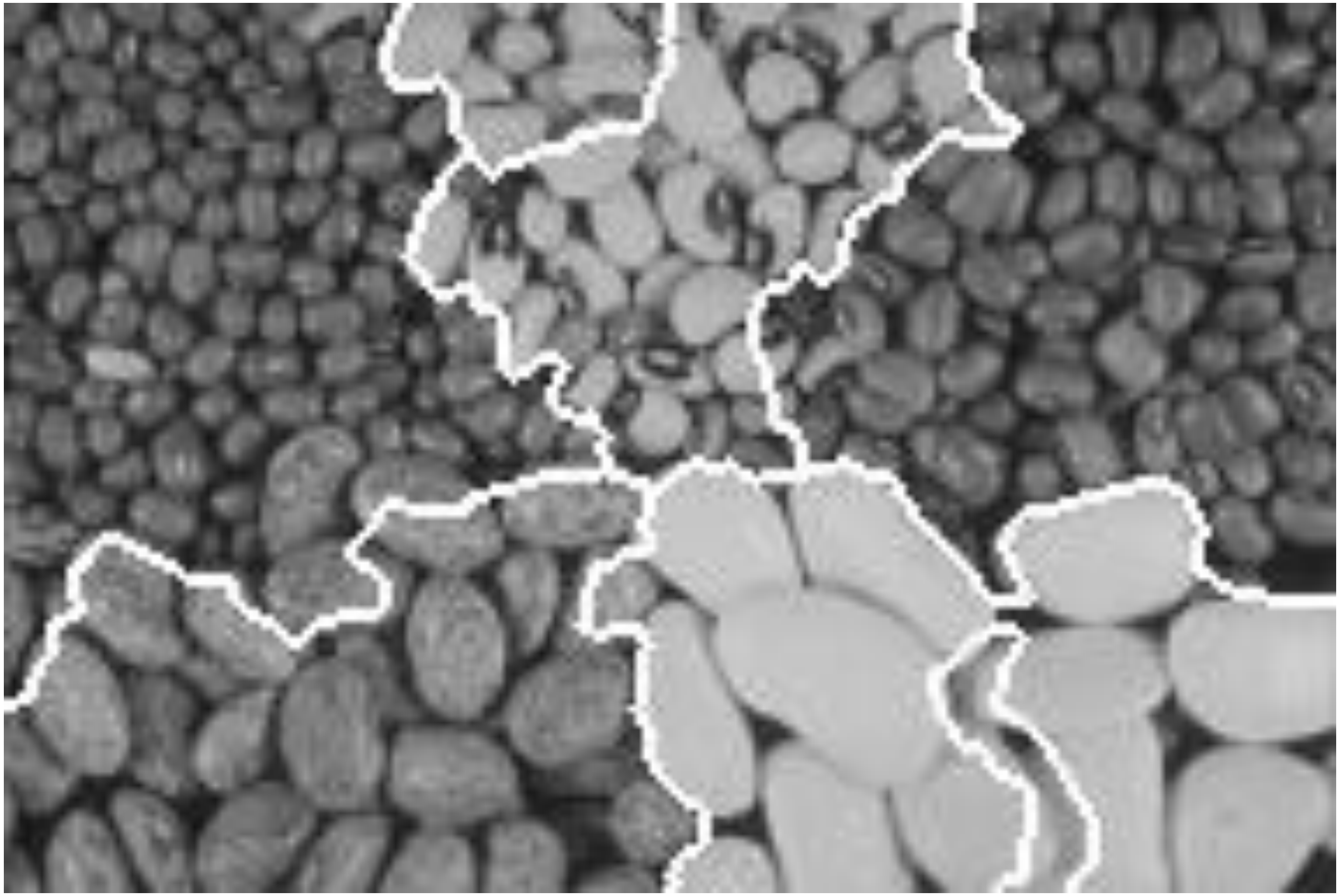
How should I segment this?

Segmentation Quality

- The quality of a segmentation depends on what you want to do with it.

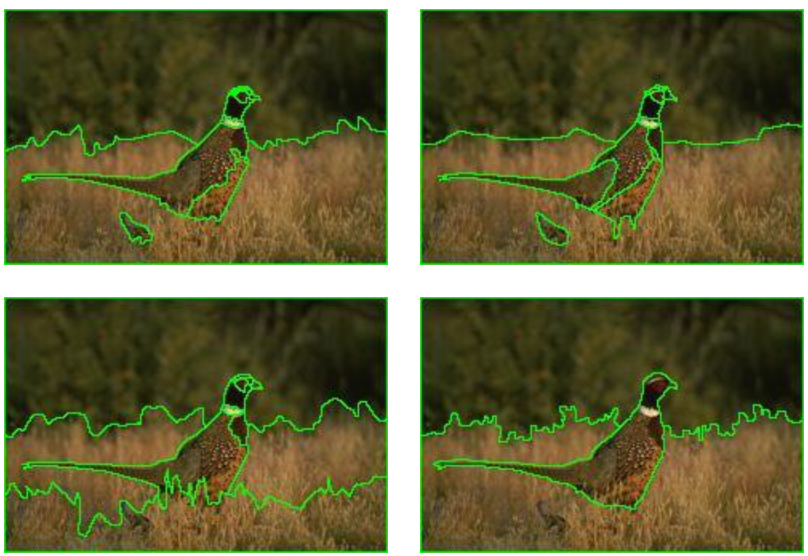
- Segmentation algorithms must be chosen and evaluated with an application in mind.

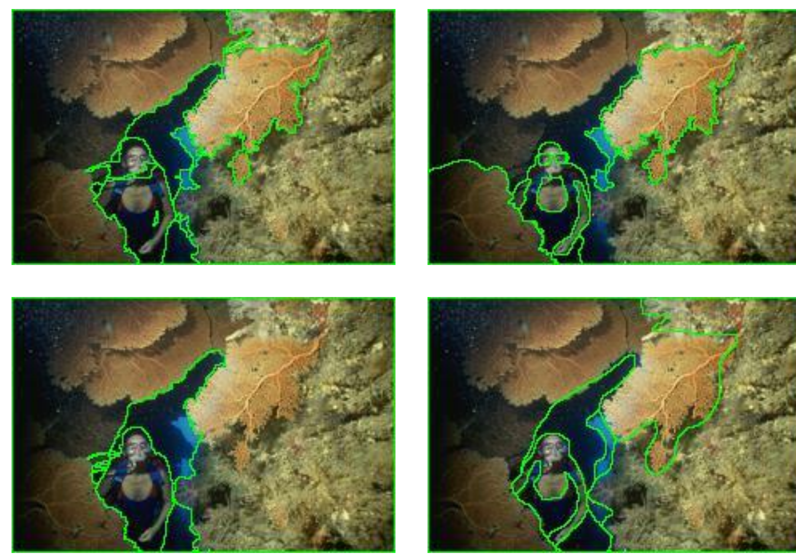
Segmentation example



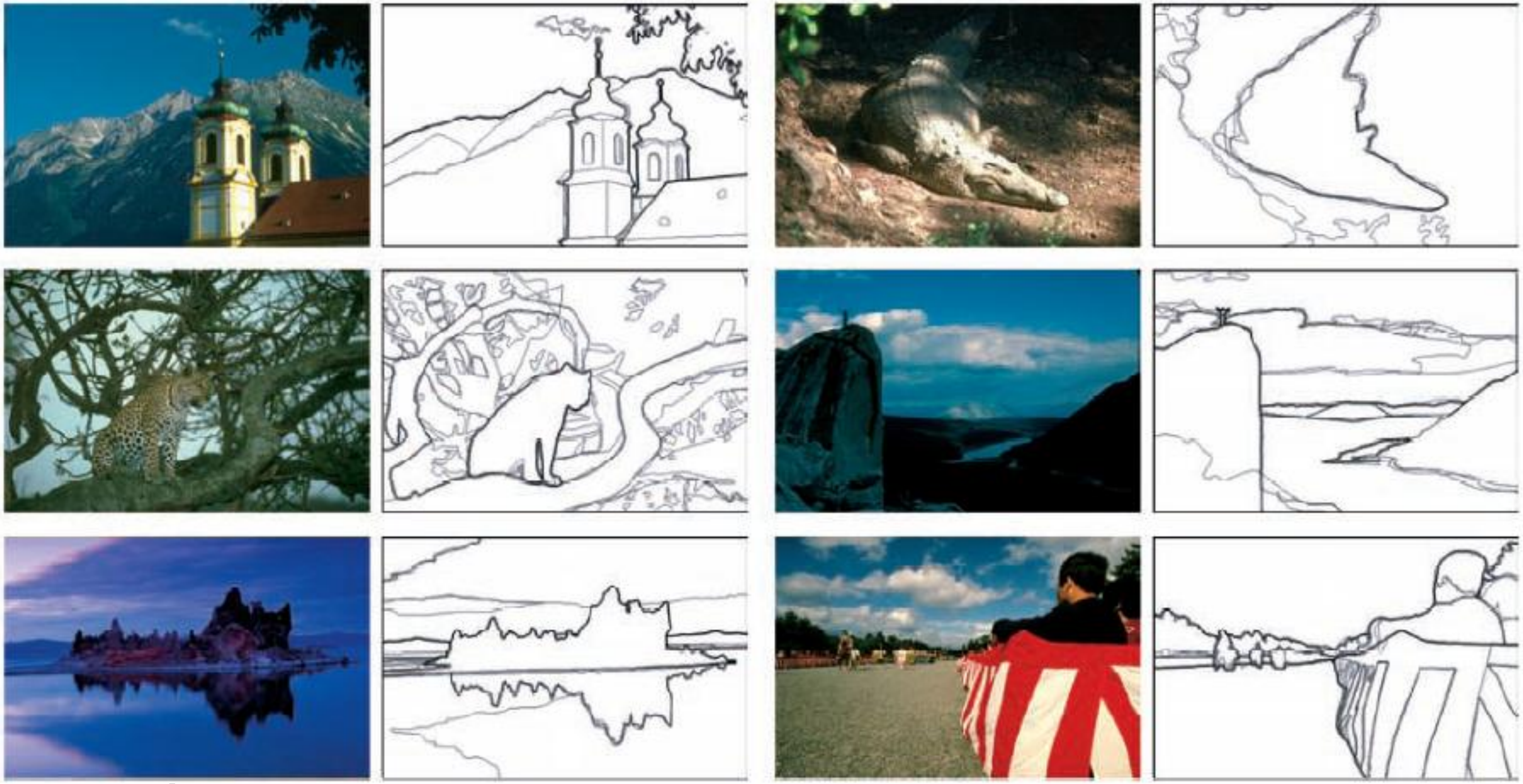
Berkeley Segmentation Database and Benchmark

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





Berkeley Segmentation Database and Benchmark



Martin et al. PAMI 2004

Thresholding

- Thresholding is a simple segmentation process.
- Thresholding produces a binary image *B*.
- It labels each pixel **in** or **out** of the region of interest by comparison of the greylevel with a threshold *T*:

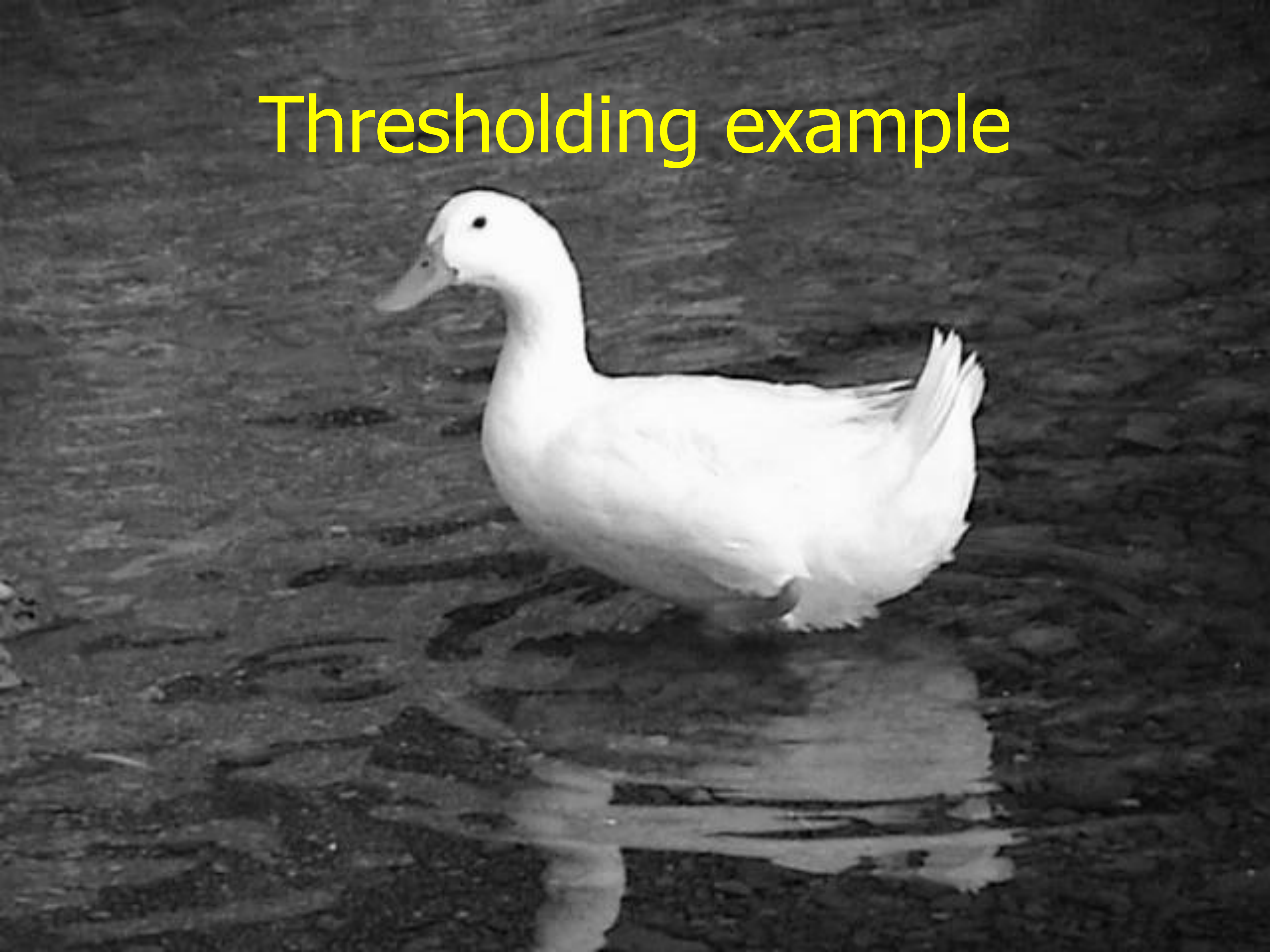
$$B(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{if } I(x,y) < T. \end{cases}$$

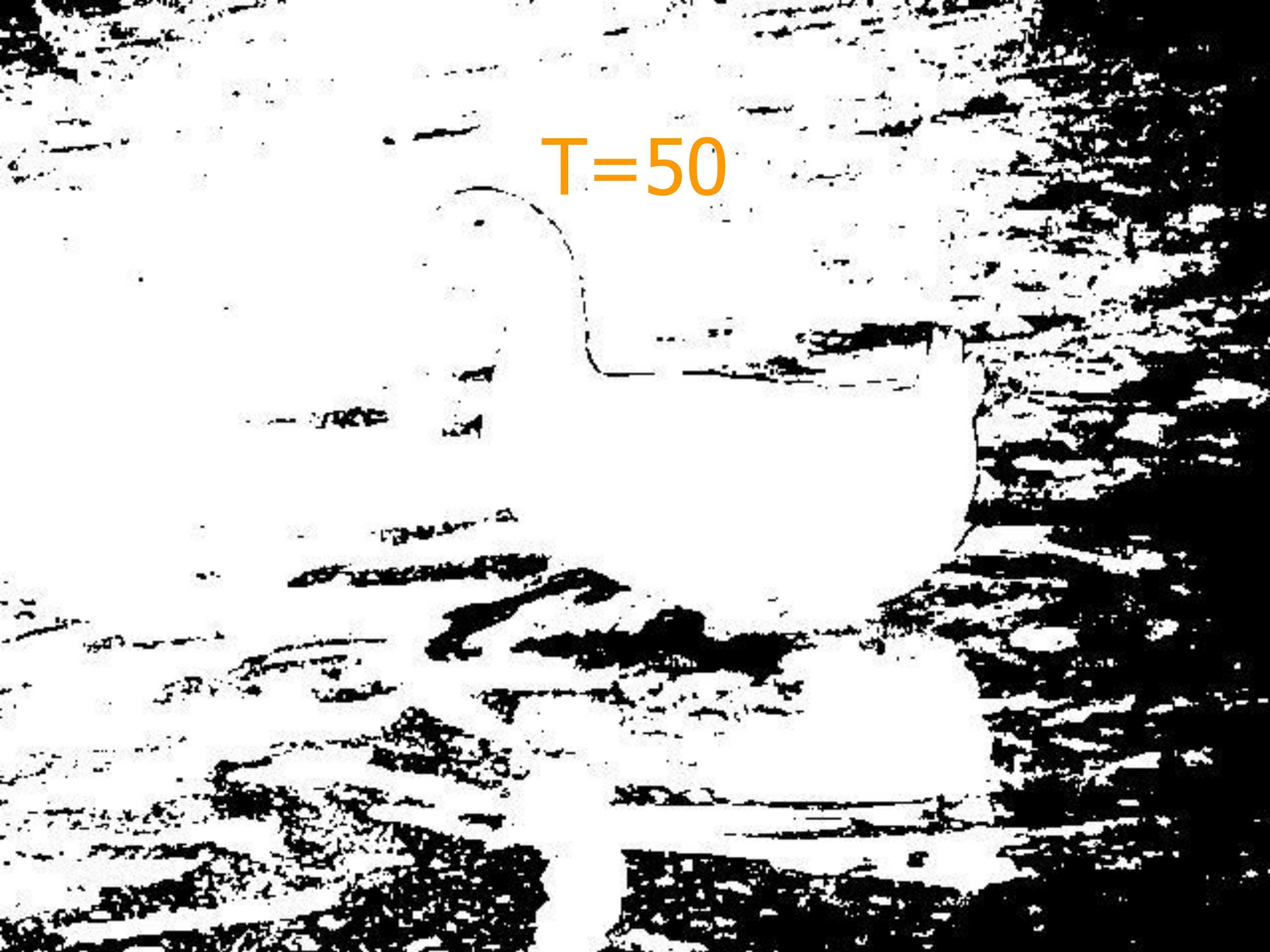
Basic Thresholding Algorithm

```
for x=1:X
    for y=1:Y
        B(x,y) = (I(x,y) >= T);
    end
end
```

Don't write it like this at home!

Thresholding example




T=50


T=100

T=150

T=200

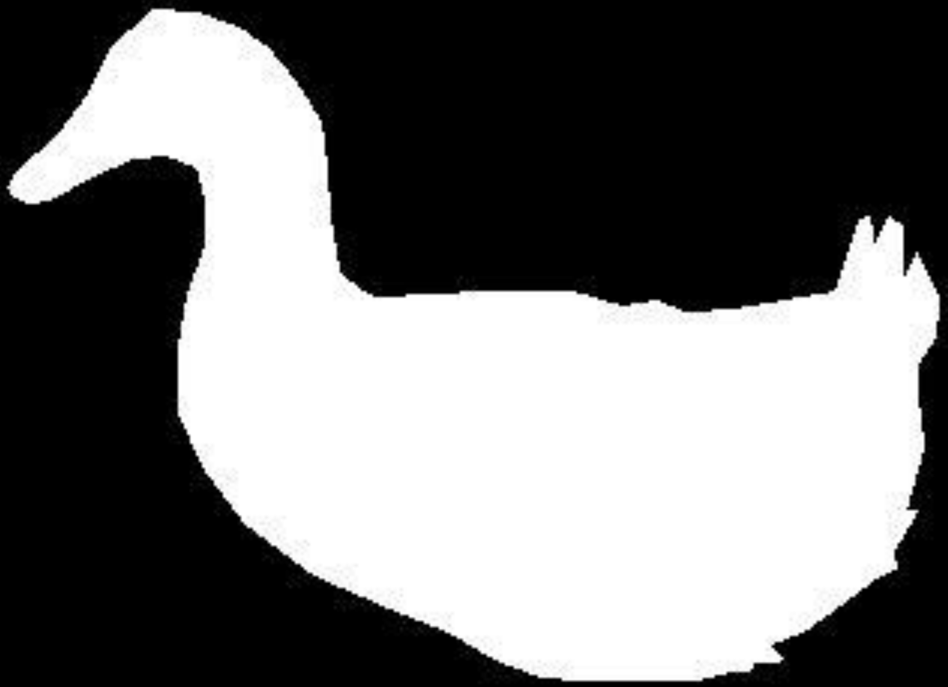
How do we choose T?

- Trial and error
- Compare results with ground truth
- Automatic methods

Segmentation Performance

- To use automatic analysis, we need to know the true classification of each test.
- We need to do the segmentation by hand on some example images.

Ground truth



ROC Analysis
(ROC = Receiver operating characteristic)

- An ROC curve characterizes the performance of a binary classifier.

- A binary classifier distinguishes between two different types of thing. E.g.:
  - Healthy/afflicted patients – cancer screening
  - Pregnancy tests
  - Foreground/background image pixels
  - Object detection

Classification Error

- Binary classifiers make errors.
- Two types of input to a binary classifier:
  - Positives
  - Negatives
- Four possible outcomes in any test:

True positive	False negative	P: Total # positives
True negative	False positive	N: Total # negatives

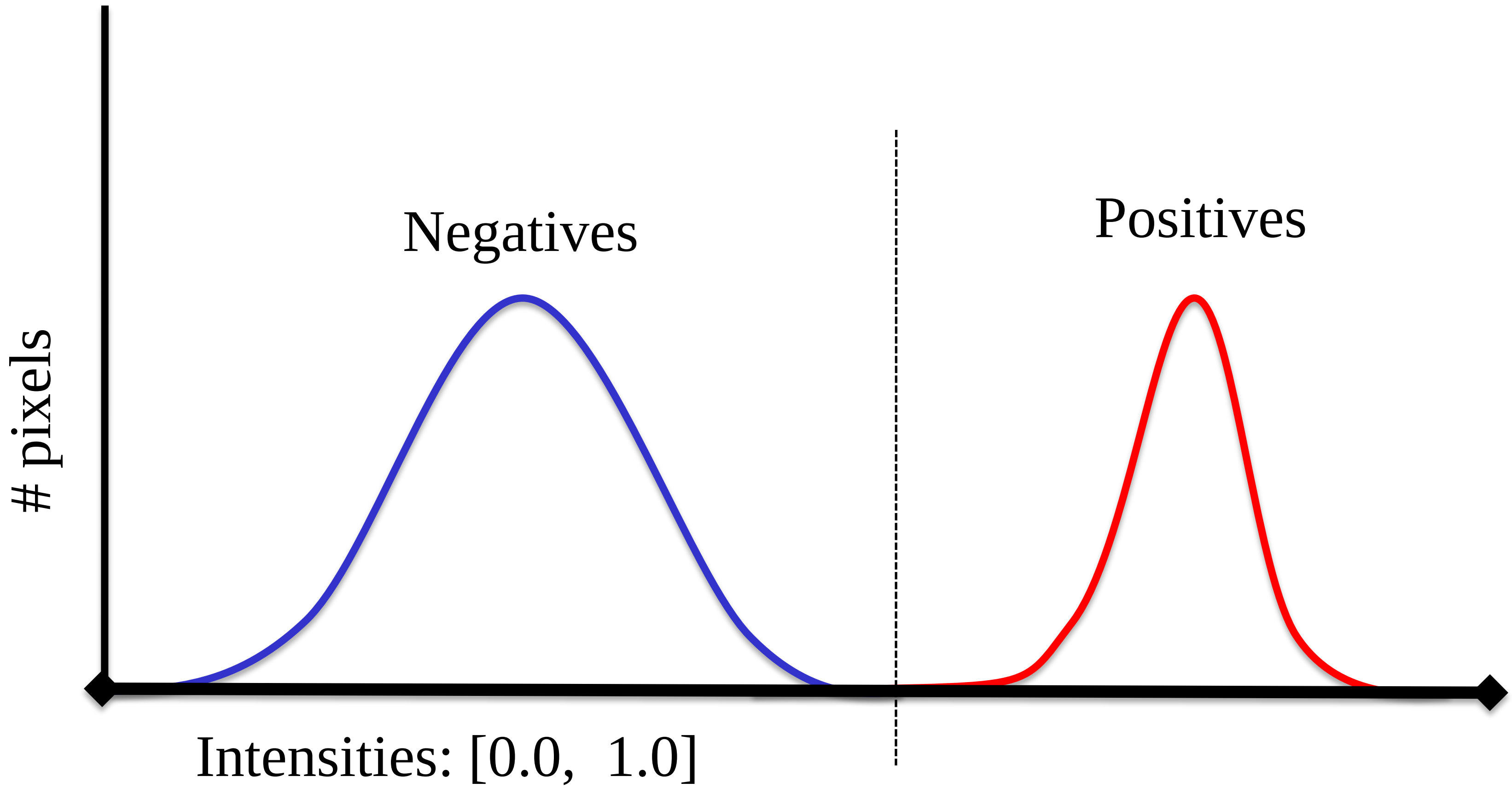
Classified: correctly | incorrectly

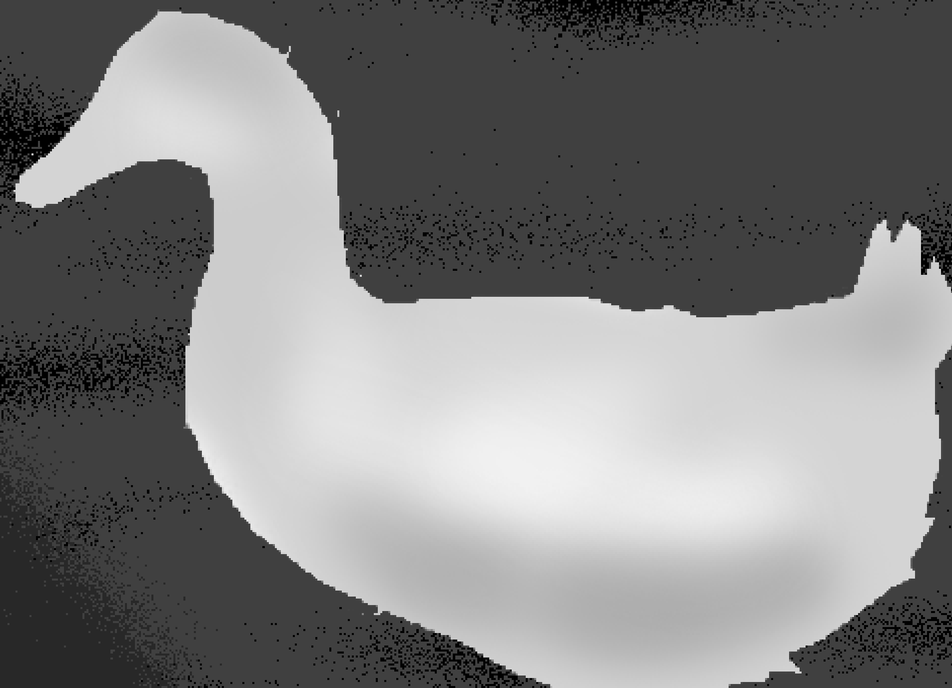
Where do errors come from?

- A binary classifier thresholds a measurement made on every case.

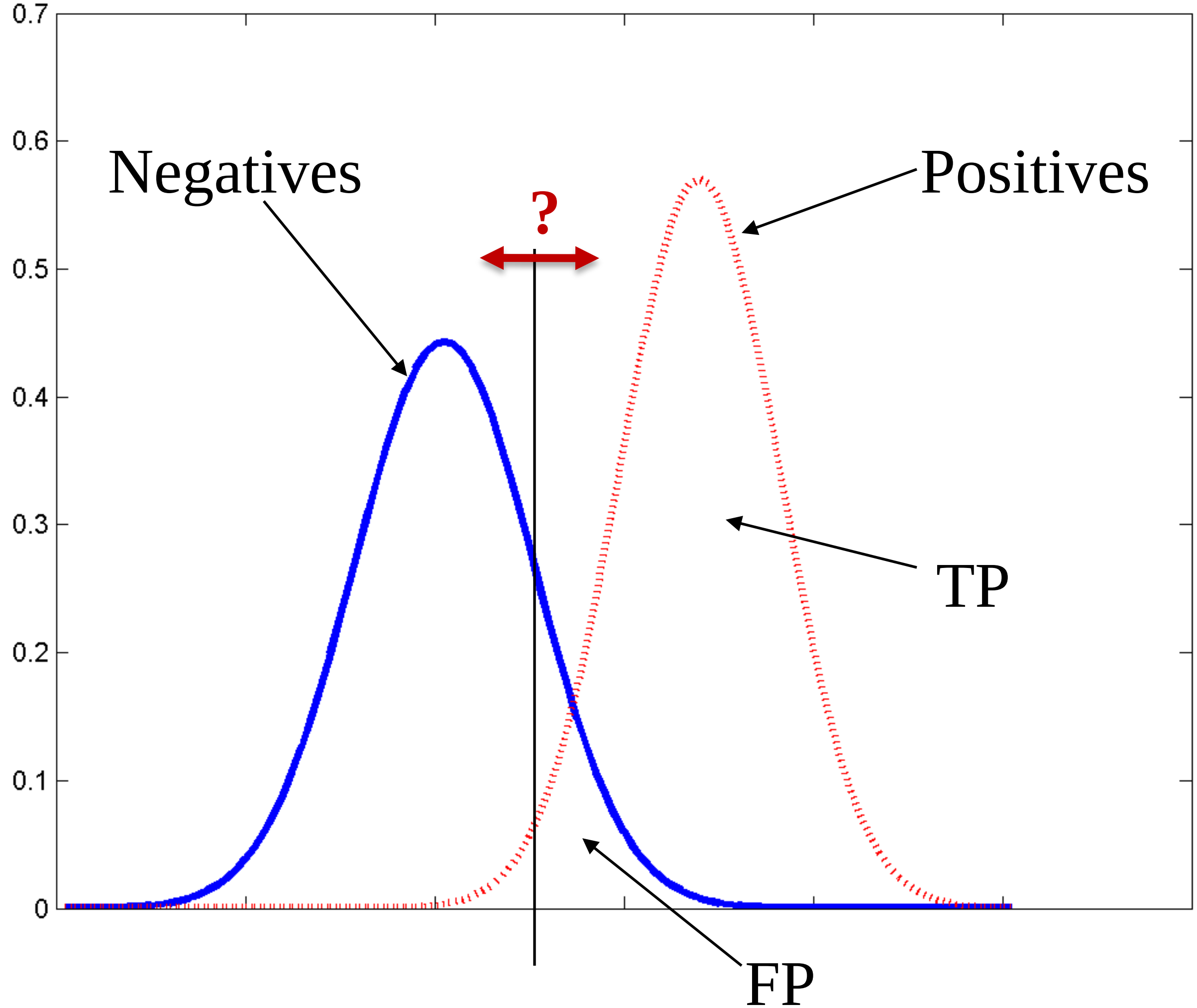
- The measurement may be explicit
  - Example: pixel grey-level: **1 scalar**
- Or implicit
  - The degree of certainty of a doctor examining a patient: **% confidence for EACH scalar →** probability distribution

Wouldn't it be nice…





Measurement Distributions



Classification outcomes

- As we change the threshold, FP and TP change.
- Notice that:
  - FP + TN = N (the total number of negatives)
  - TP + FN = P (total positives)

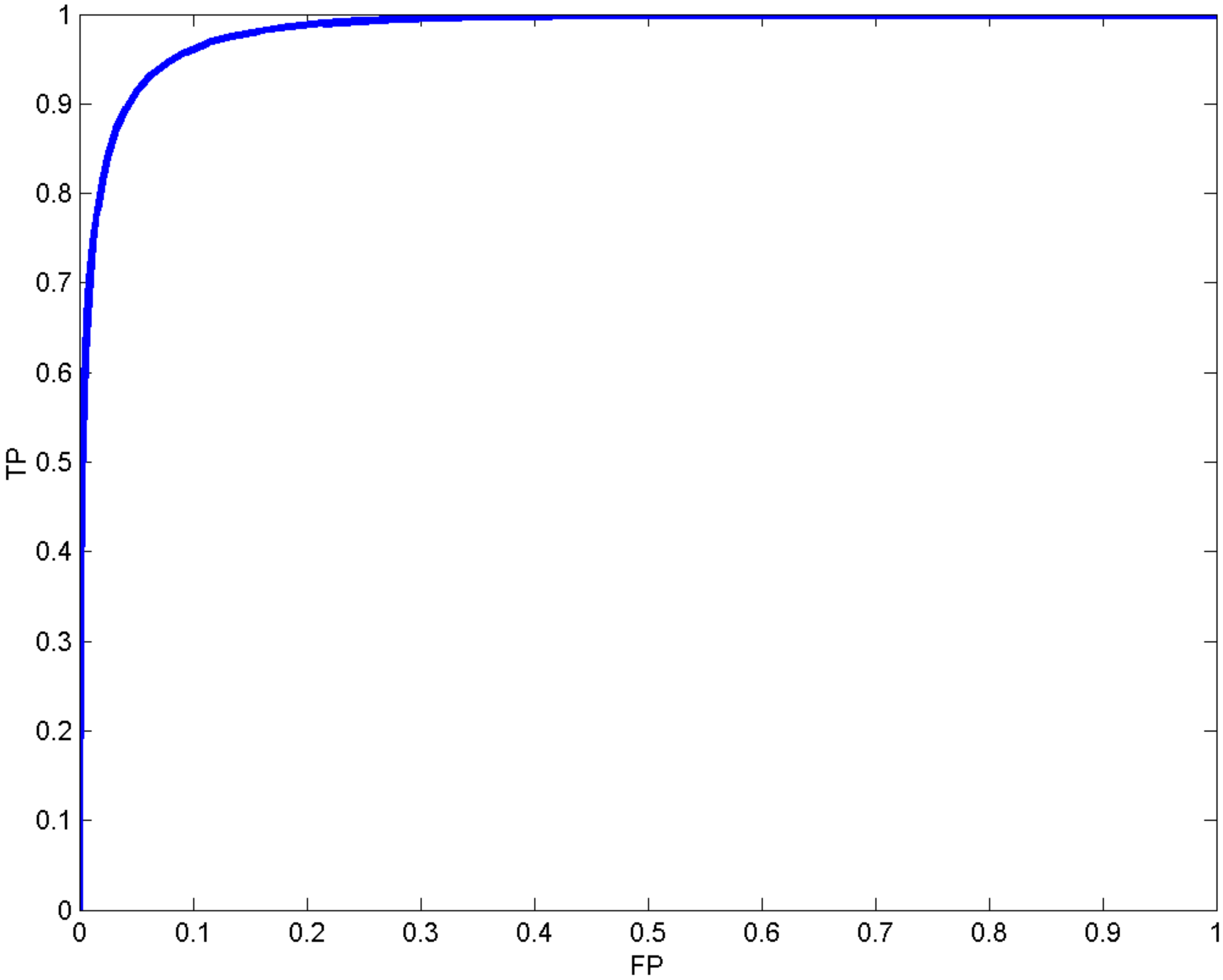
- How to evaluate performance?

The ROC curve

- Characterizes the error trade-off in binary classification tasks.
- It plots the TP fraction against FP fraction.
- TP fraction (*sensitivity*) is $\frac{\text{True positive count}}{P}$
- FP fraction (1-*specificity*) is $\frac{\text{False positive count}}{N}$

The ROC Curve

$$\frac{\text{True positive count}}{P = TP + FN}$$



$$\frac{\text{False positive count}}{N = FP + TN}$$

Properties of ROC curves

- An ROC curve always passes through (0,0) and (1,1).

- What is the ROC curve of a perfect system?
- What if the ROC curve is a straight line from (0,0) to (1,1)?

Area under the ROC curve

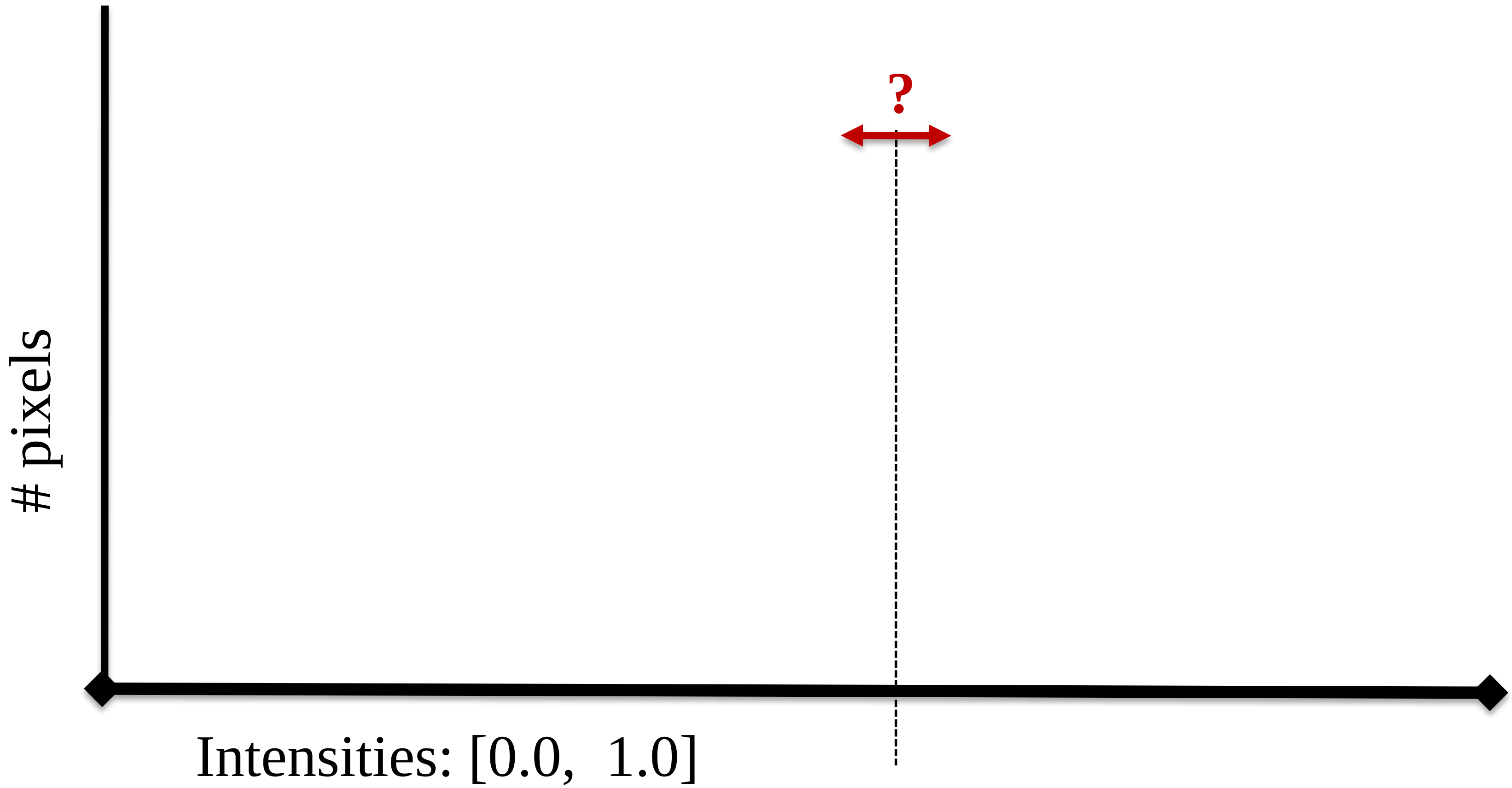
- The area A under the curve measures overall classification performance.
- If the distributions of measurements on positive and negative cases are $N(\mu_P, \sigma_P)$ and $N(\mu_N, \sigma_N)$, respectively, then

$$A = Z\left(\frac{|\mu_P - \mu_N|}{\left(\sigma_P^2 + \sigma_N^2\right)^{1/2}} \right)$$

- z is the cumulative normal distribution function

$$Z(x) = \int_{-\infty}^{x} z(t)\,dt$$

Auto-select a threshold?



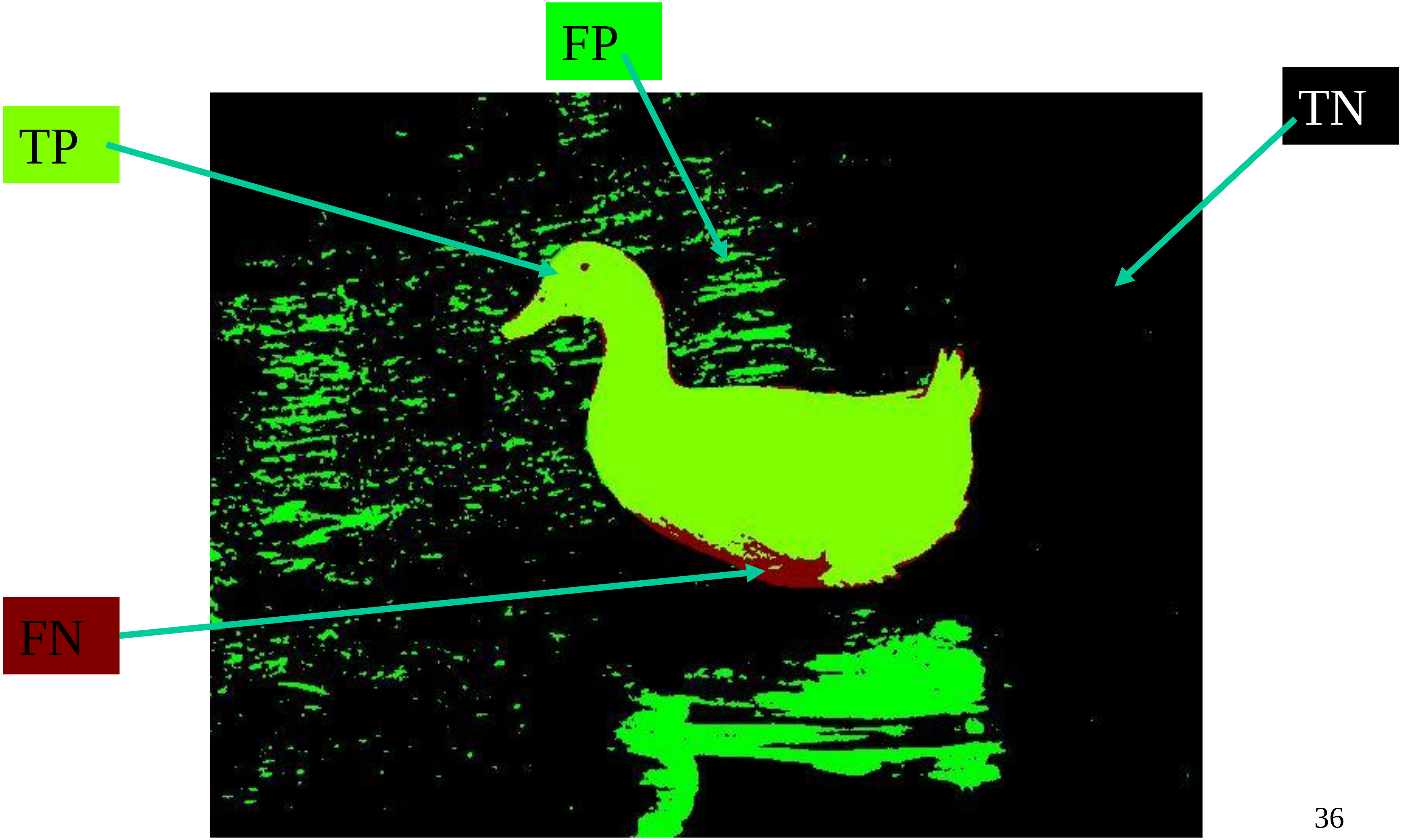
Operating points

- Choose an *operating point* by assigning relative costs and values to each outcome:
  - V_{TN} – value of true negative
  - V_{TP} – value of true positive
  - C_{FN} – cost of false negative
  - C_{FP} – cost of false positive
- Choose the point on the ROC curve with **gradient**

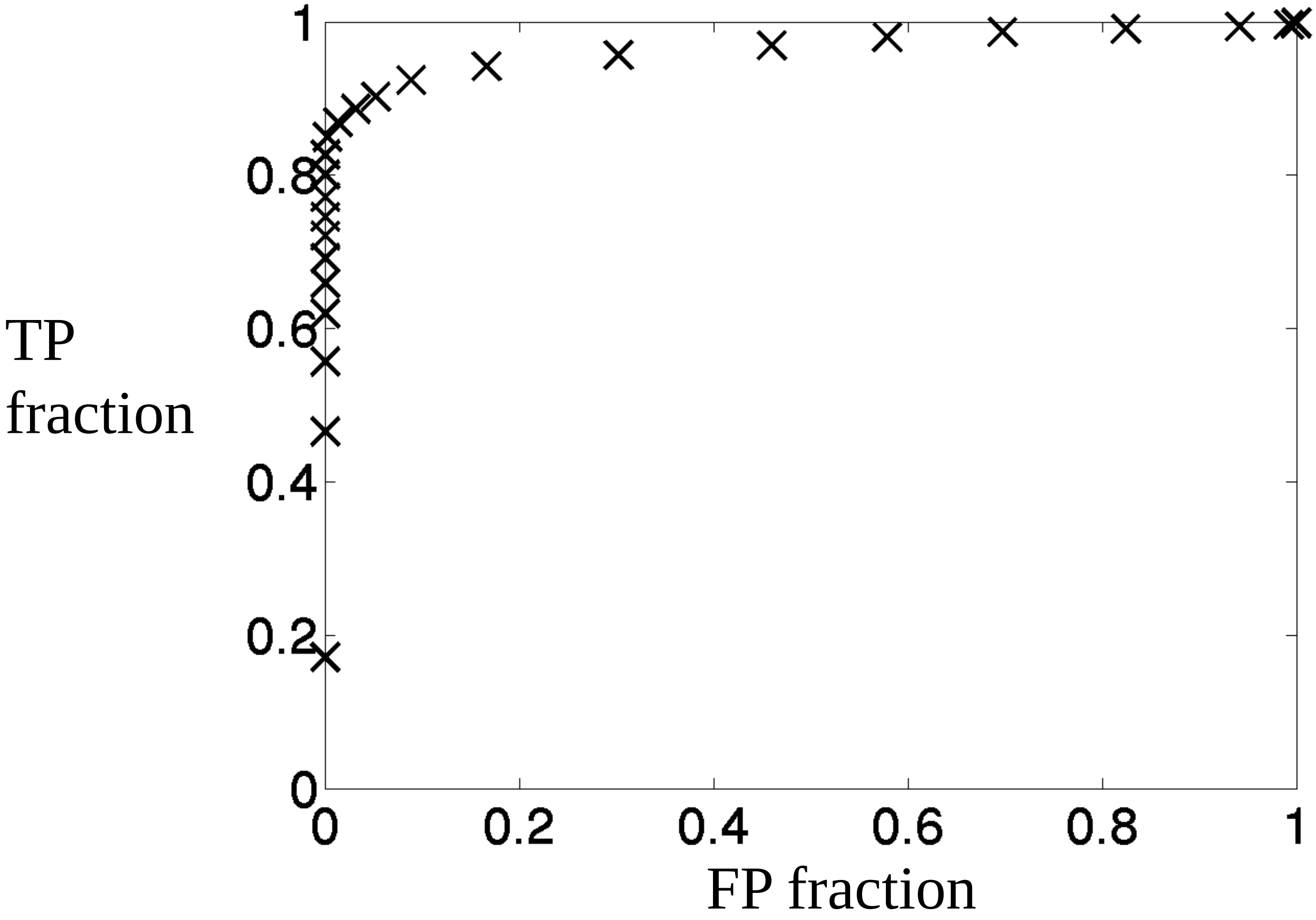
$$\beta = \frac{N}{P}\frac{V_{TN} + C_{FP}}{V_{TP} + C_{FN}}$$

- For simplicity, we often set $V_{TN} = V_{TP} = 0$.

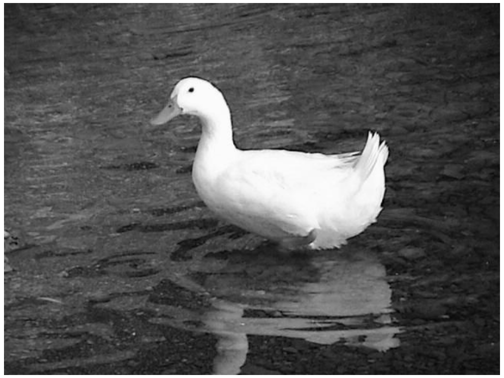
Classification outcomes

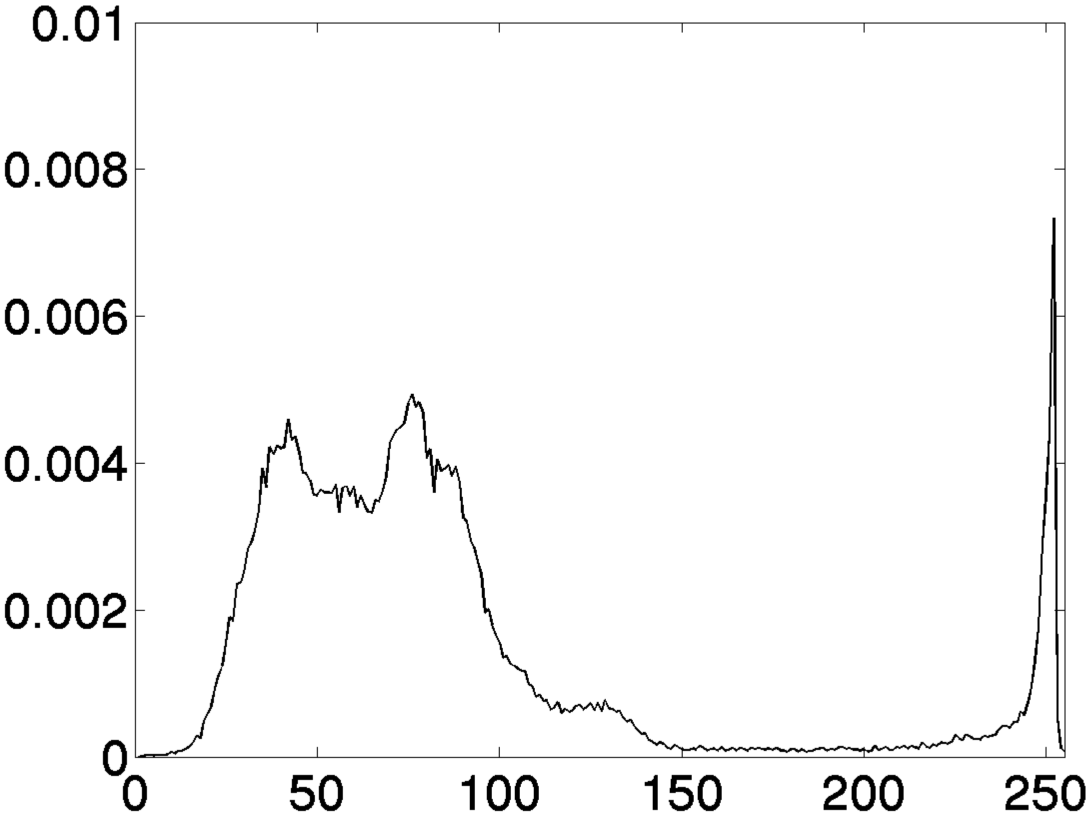


ROC curve

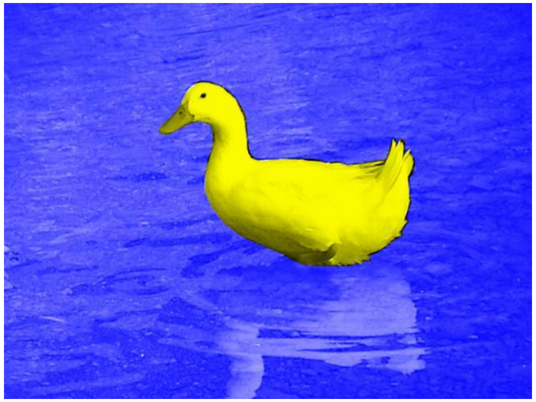


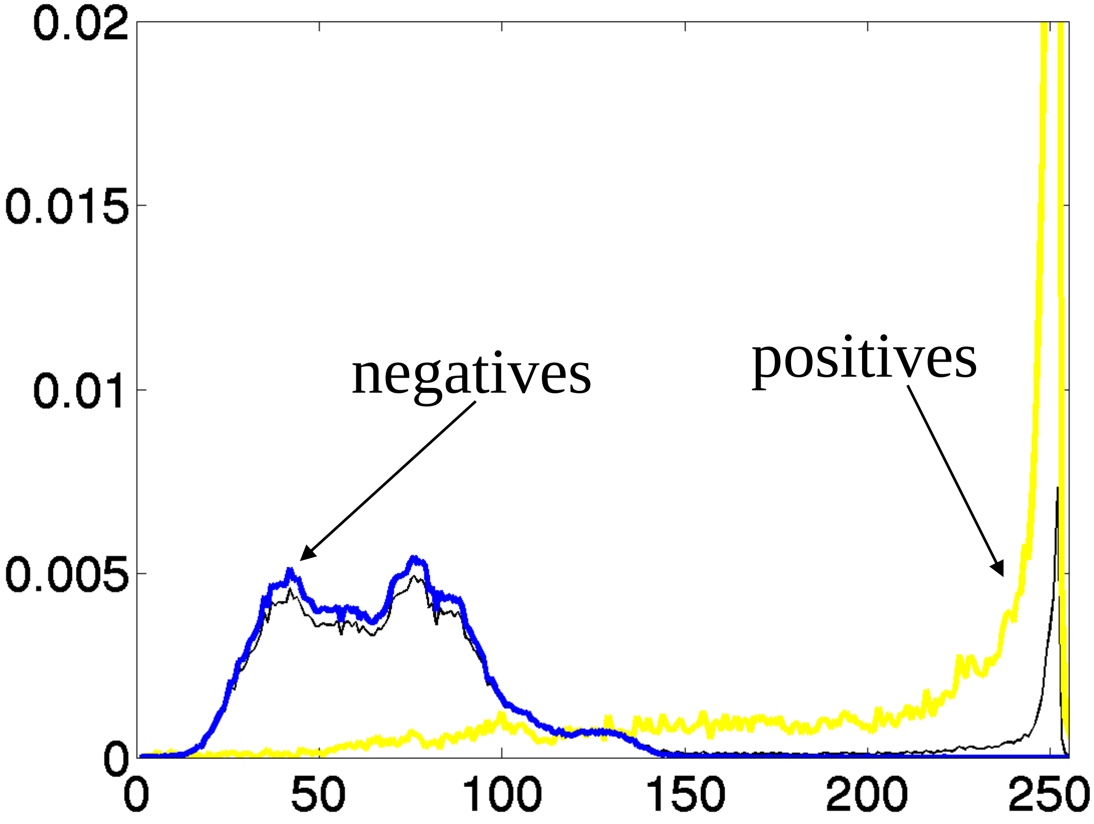
Greylevel Histograms





Positives and Negatives





Automatic Threshold Selection

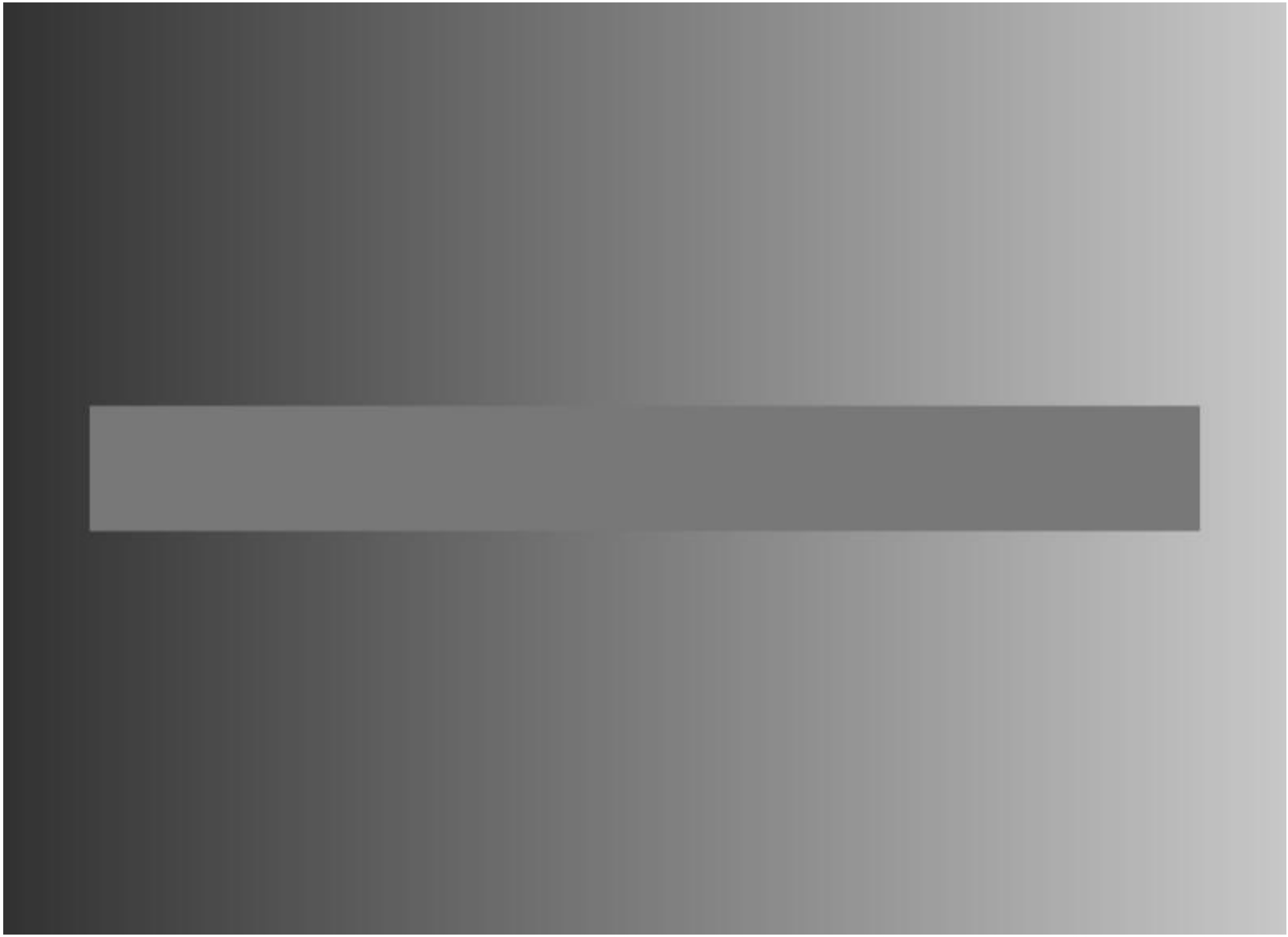
- Try to find a threshold between histogram peaks
- Maybe...model positive and negative region histograms by Gaussian distributions

K-means Clustering

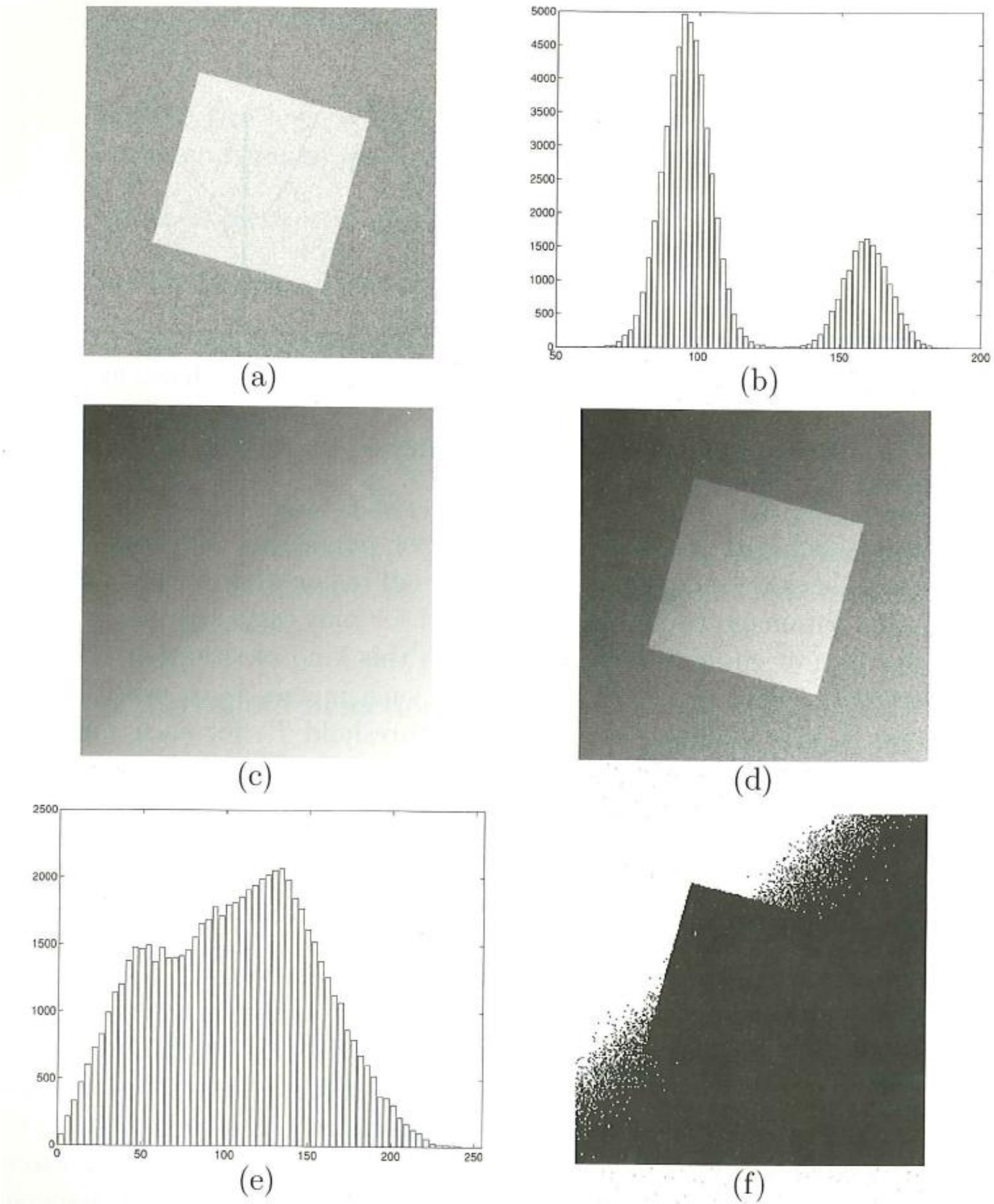
```
Initialize R_f and R_b
REPEAT
    % Find the mean greylevel in each region.
    m_f = mean(I(R_f));
    m_b = mean(I(R_b));
    % Pick threshold half way between.
    T = (m_f + m_b)/2;
    % Find new regions.
    R_f = {(x,y) | I(x,y)>=T};
    R_b = I\R_f;
UNTIL T remains unchanged.
```

Improvements?

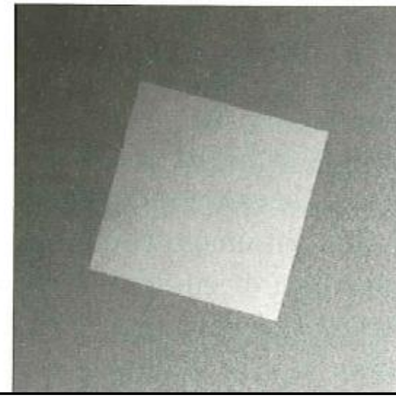
- Can you extend this to use the standard deviations of the regions as well?
- What problems might you encounter?
- Can one take advantage of special situations?

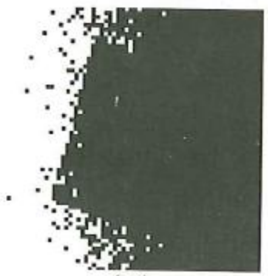


Gradient.illusion.arp.jpg


5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
50
100
150
200
2500
2000
1500
1000
500
0
0
50
100
150
200
250

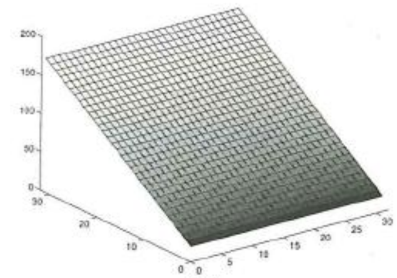
(a) (b) (c) (d) (e) (f)



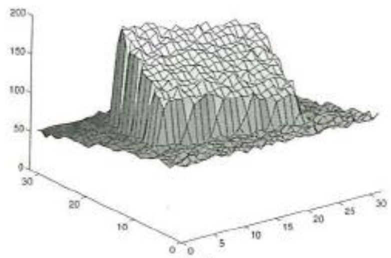


(d)

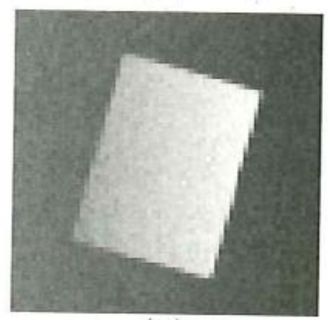
(e)



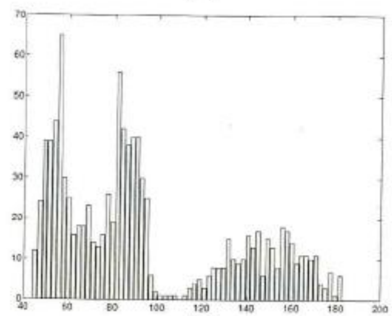
(f)



(g)



(h)



(i)

(j)

Recap of Thresholding (only)

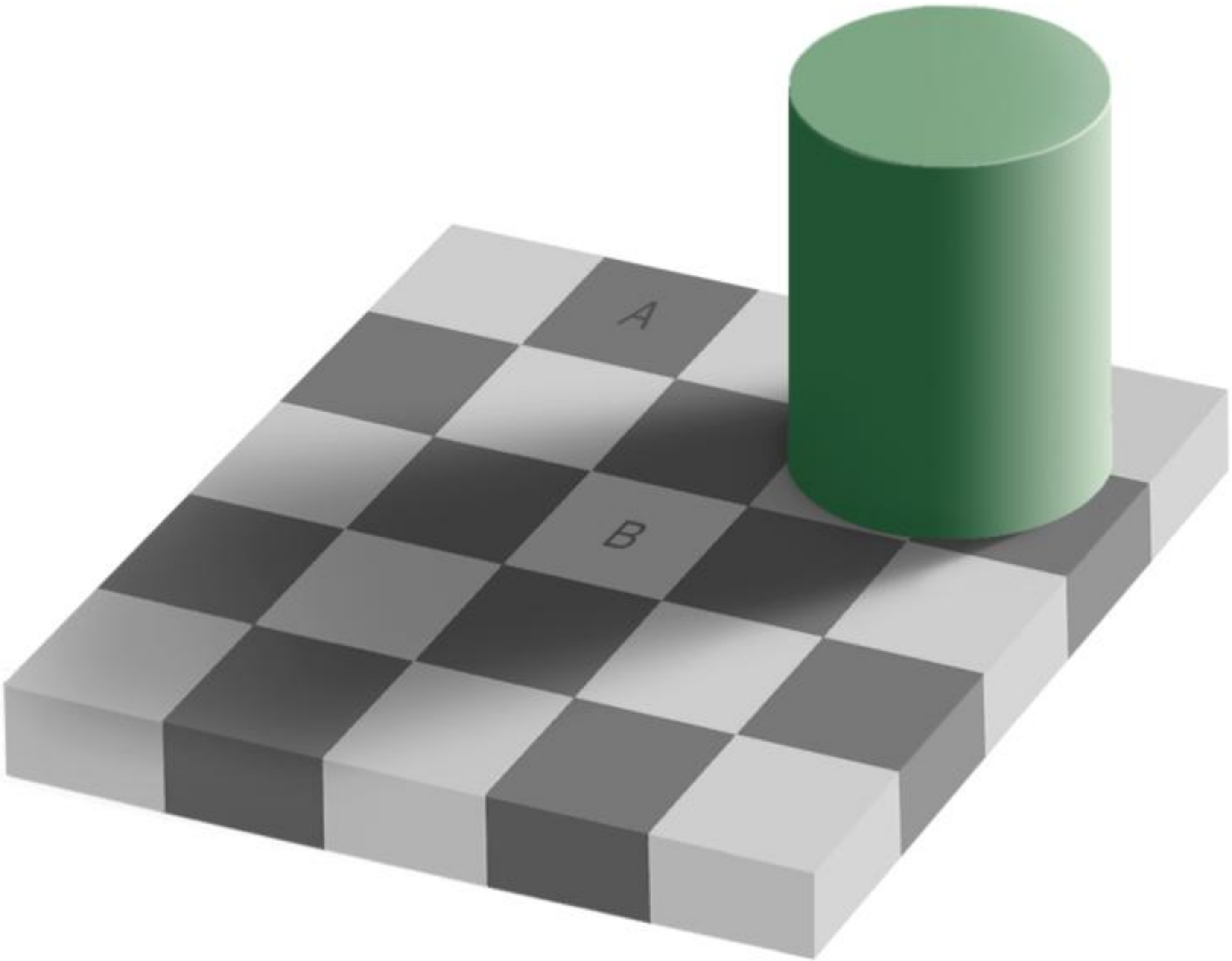
- Thresholding for binary segmentation
- Choosing the threshold can be difficult
- ROC analysis characterizes binary classification systems
- Can help choose a good threshold
- Simple algorithms can find good thresholds sometimes.

Note on Performance Assessment

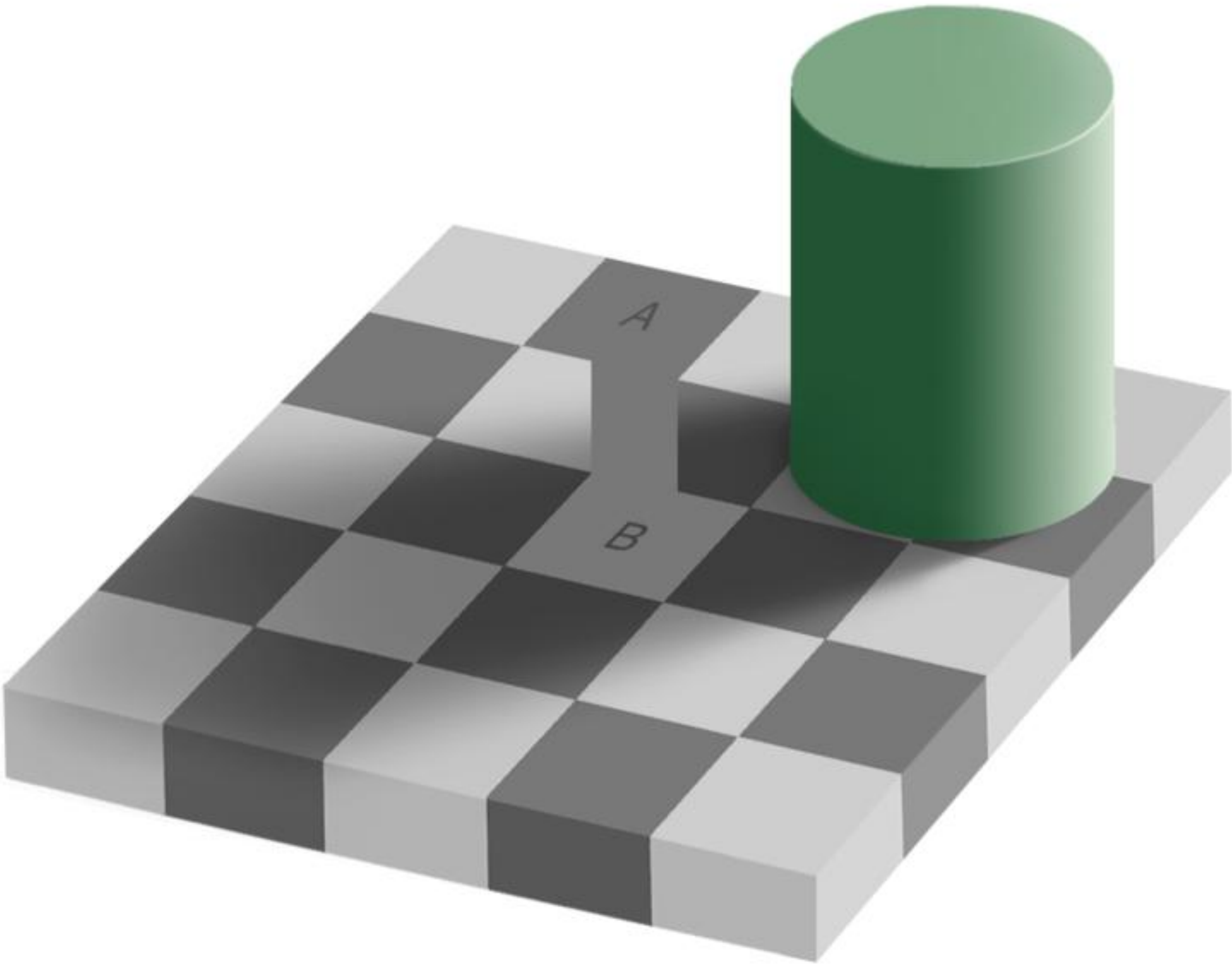
- In real-life, we use two or even three separate sets of test data:
  - A ***training set***, for tuning the algorithm.
  - An unseen ***test set*** to get a final performance score on the tuned algorithm.
  - A ***validation*** set for verifying the performance score obtained on the test set.

Limitations of Thresholding

- Why can we segment images much better by eye than through thresholding processes?



by Adrian Pingstone, based on the original created by Edward H. Adelson



by Adrian Pingstone, based on the original created by Edward H. Adelson

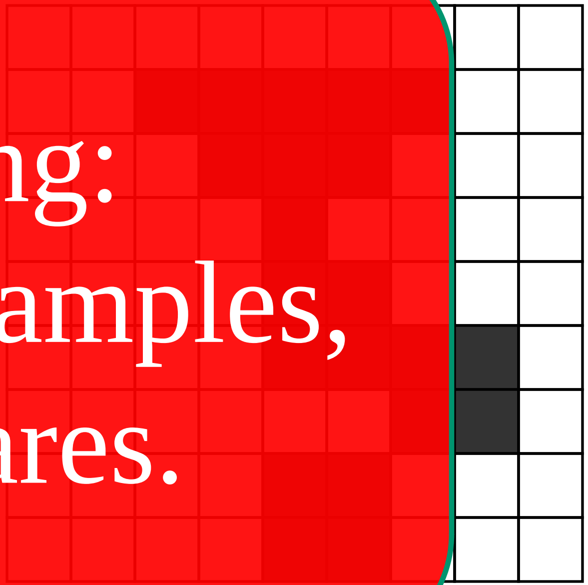
Limitations of Thresholding

- Why can we segment images much better by eye than through thresholding processes?

- We might improve results by considering image **context:** Surface Coherence

Pixel connectivity

- We need to define which pixels are neighbours.
- Are the dark pixels in this array connected?

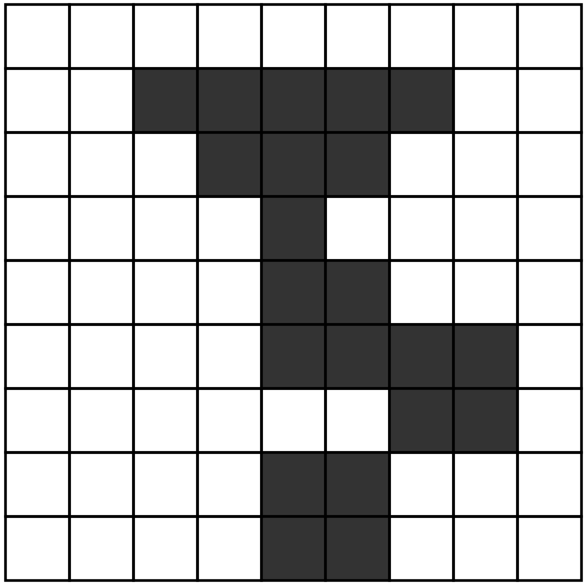


Warning:

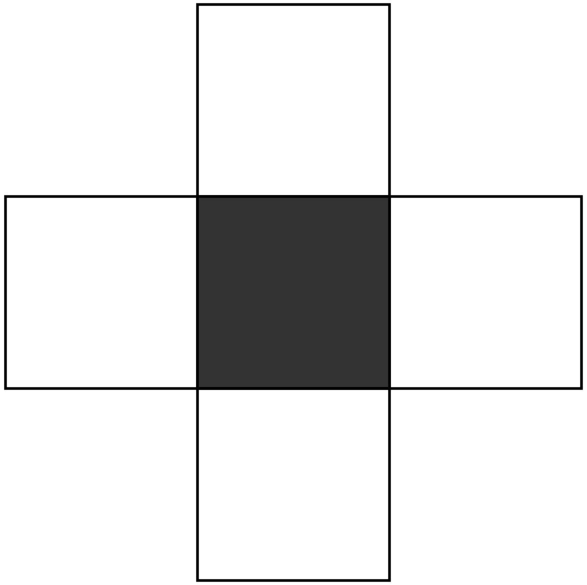
Pixels are samples, not squares.

Pixel connectivity

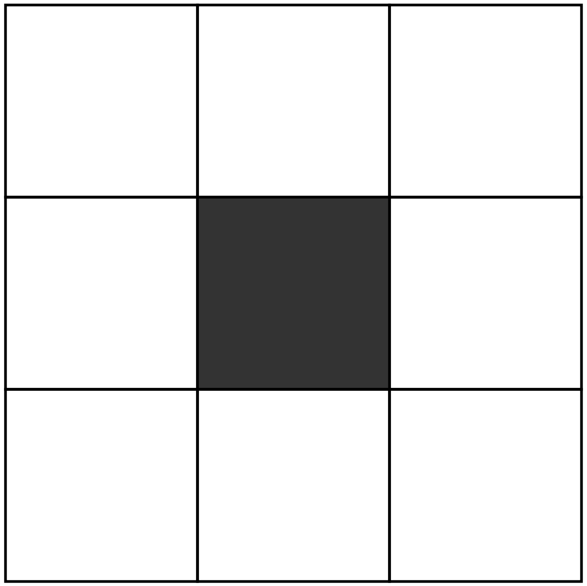
- We need to define which pixels are neighbours.
- Are the dark pixels in this array connected?



Pixel Neighbourhoods



4-neighbourhood



8-neighbourhood

Pixel paths

- A 4-connected path between pixels p_1 and p_n is a set of pixels $\{p_1, p_2, \ldots, p_n\}$ such that p_i is a 4-neighbour of p_{i+1}, $i=1,\ldots,n$-1.

- In an 8-connected path, p_i is an 8-neighbour of p_{i+1}.

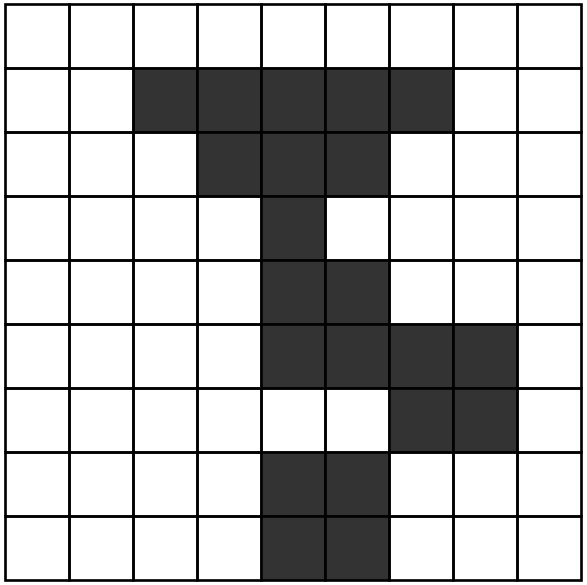
Connected regions

- A region is 4-connected if it contains a 4-connected path between any two of its pixels.

- A region is 8-connected if it contains an 8-connected path between any two of its pixels.

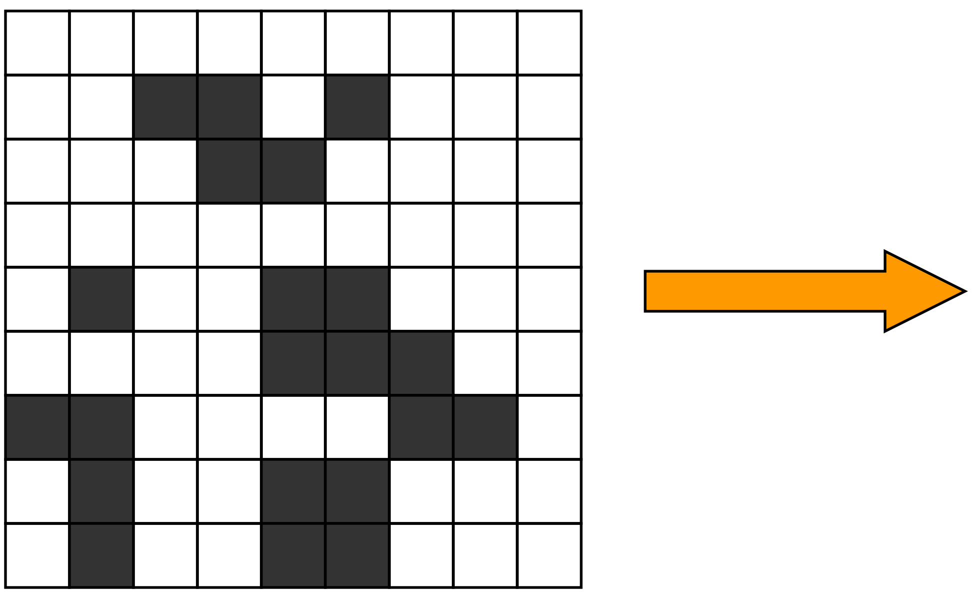
Connected regions

- Now what can we say about the dark pixels in this array?
- What about the light pixels?



Connected components labelling

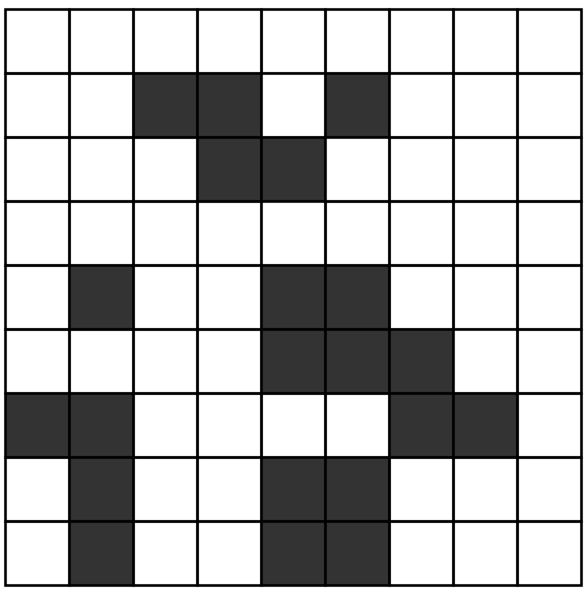
- Labels each connected component of a binary image with a separate number.



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

Foreground labelling

- Only extract the connected components of the foreground



1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1
1	1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1
1	0	1	1	0	0	1	1	1
1	1	1	1	0	0	0	1	1
0	0	1	1	1	1	0	0	1
2	0	1	1	0	0	1	1	1
2	0	1	1	0	0	1	1	1

Connected components

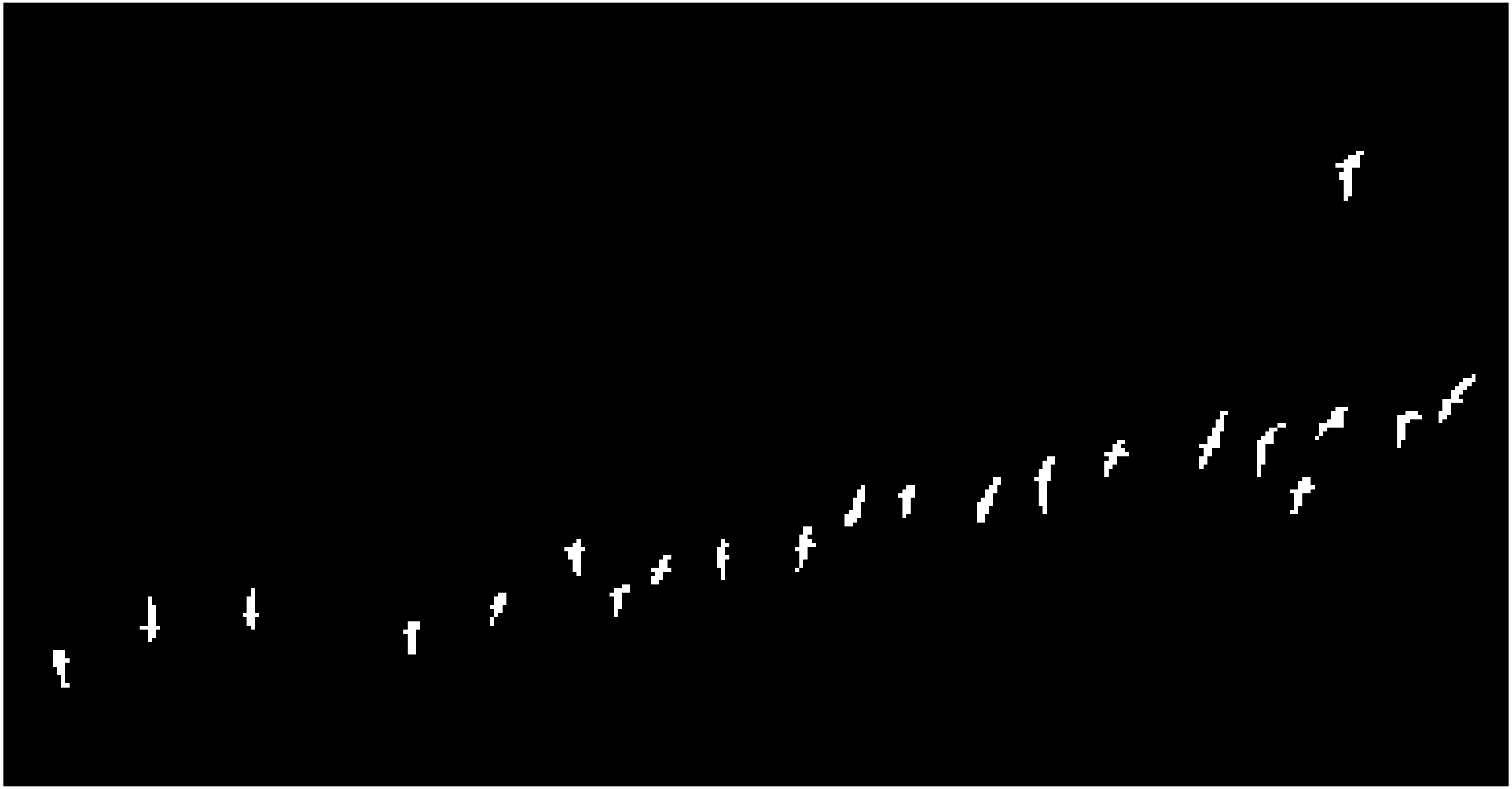
```
% B is the binary image input.  L is the labelled
% image output.
function L = ConnectedComponents(B)
[X,Y] = size(B);
L = zeros(X,Y);
n=1;
For each (x,y) in B
    if(B(x,y) & L(x,y)==0)
        label(x,y,n,B,L);
        n=n+1;
    end
end
```

Connected components

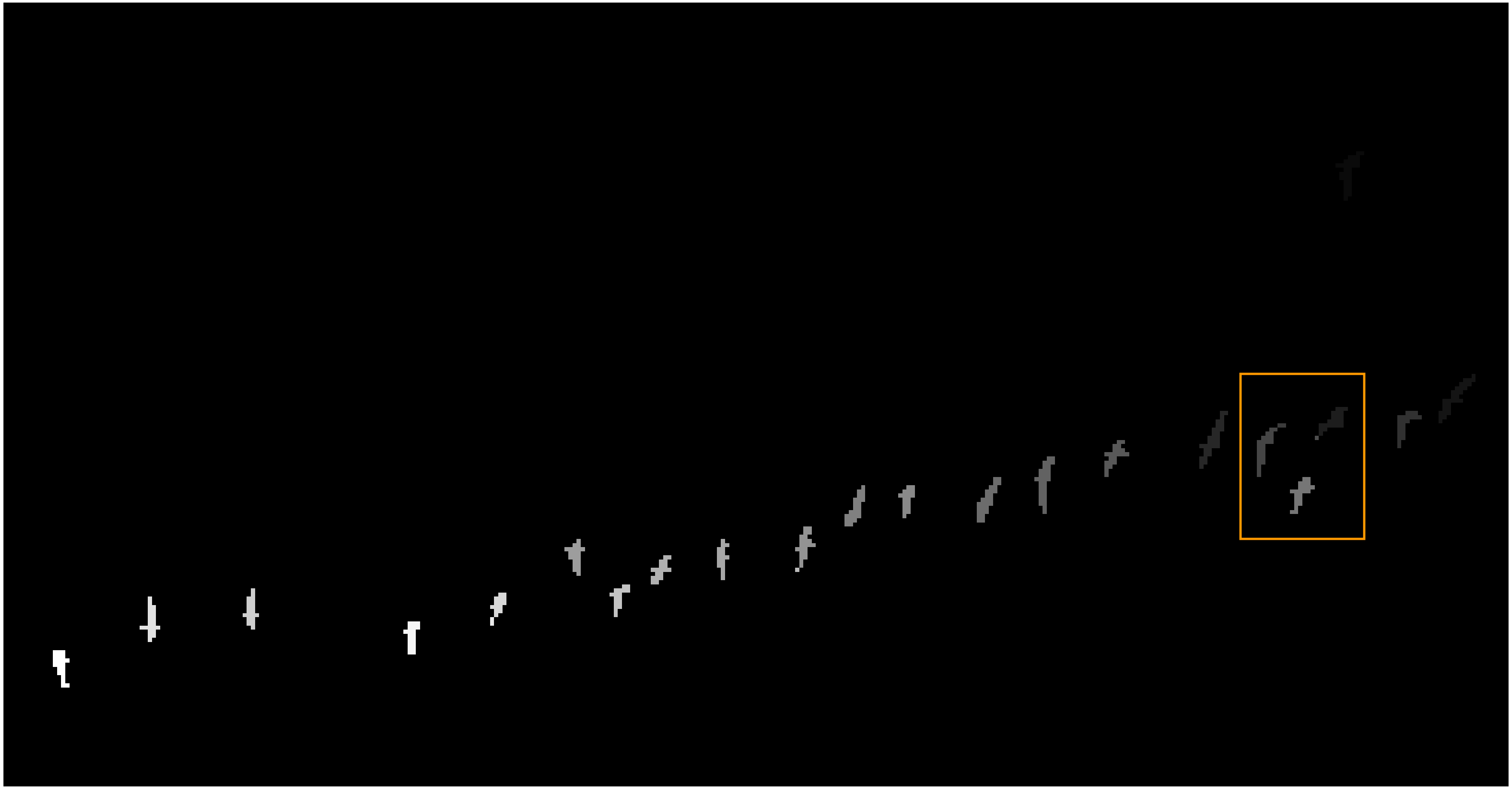
```
function label(x_start, y_start, n, B, L)
% Recursively give label n to this pixel
% and all its foreground neighbours.
L(x_start,y_start)=n;
For each (x,y) in N(x_start, y_start)
    if(L(x,y)==0 & B(x, y))
        label(x,y,n,B,L);
    end
end
```

Goose detector

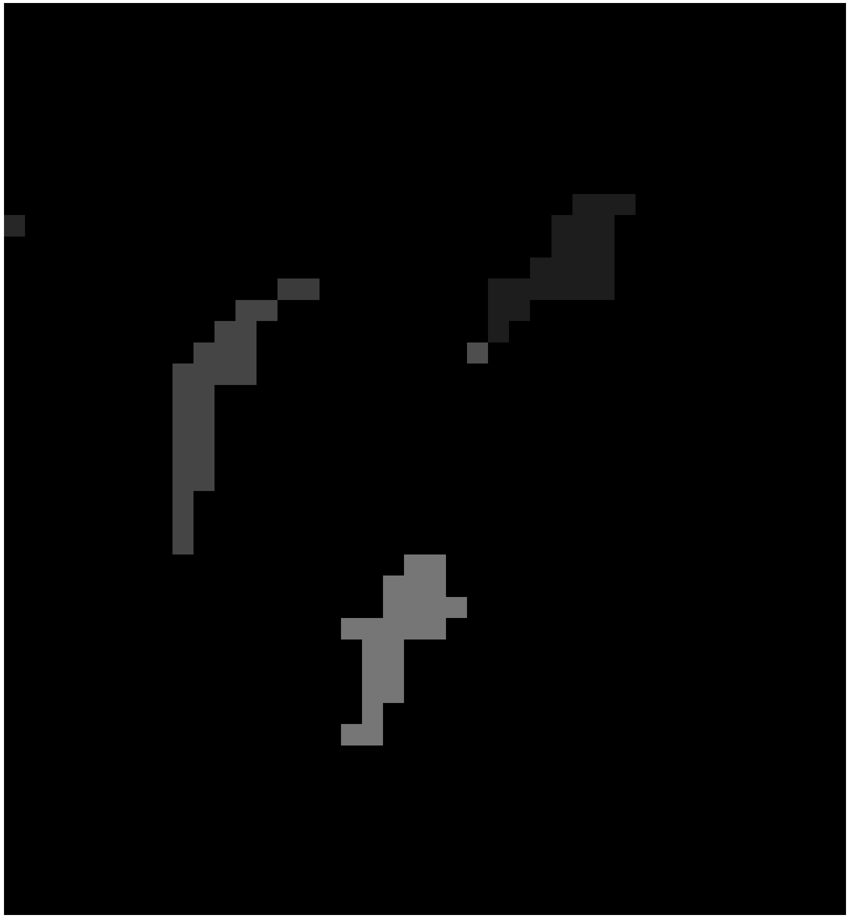
Goose detector



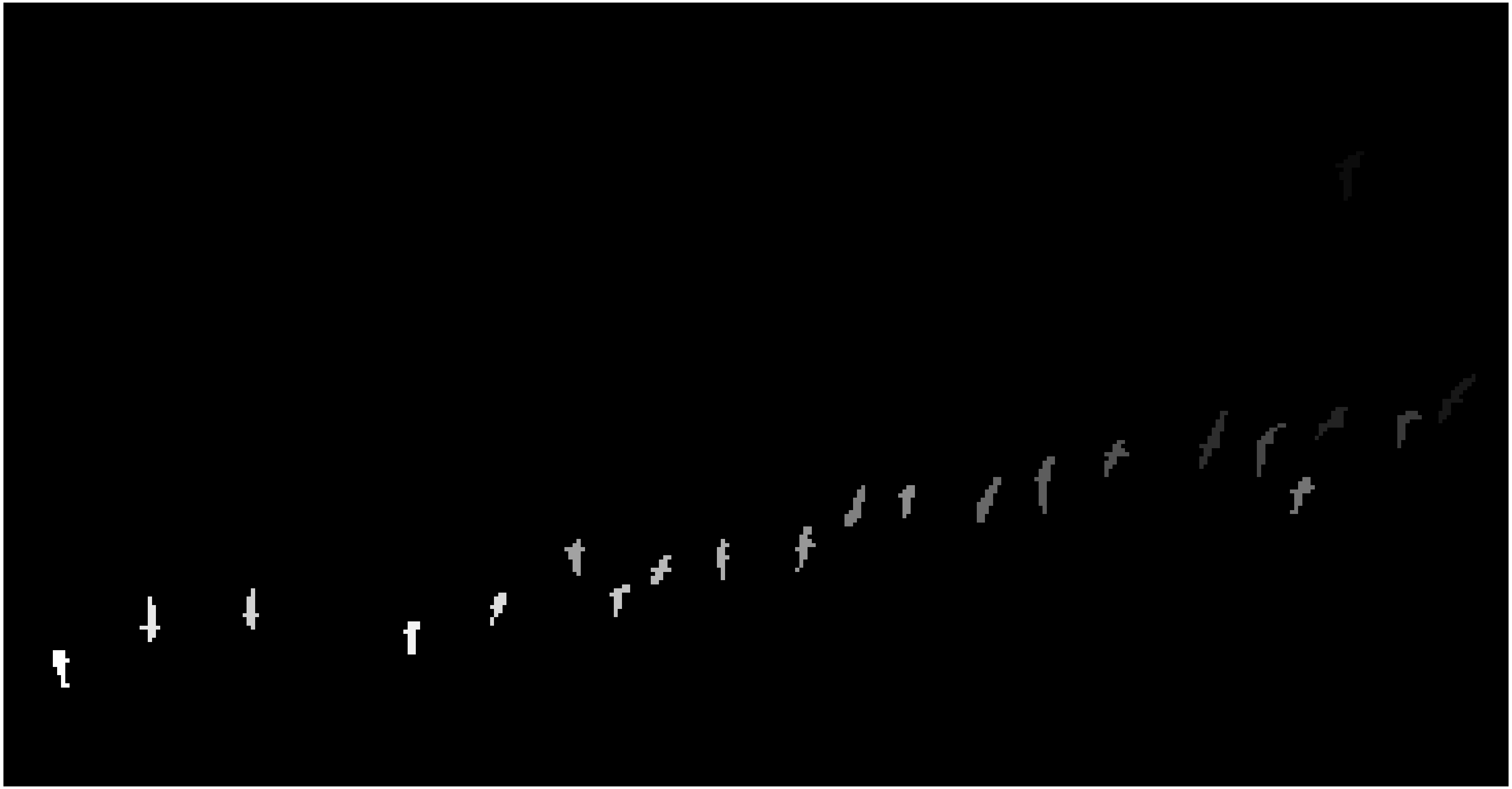
Goose 4-components (26)



Goose Error



Goose 8-components (22)



Connected components

- What happens if we use the connected components algorithm on this image?

- How might we improve our implementation?

Region Growing

- Start from a seed point or region.
- Add neighbouring pixels that satisfy the criteria defining a region.
- Repeat until we can include no more pixels.

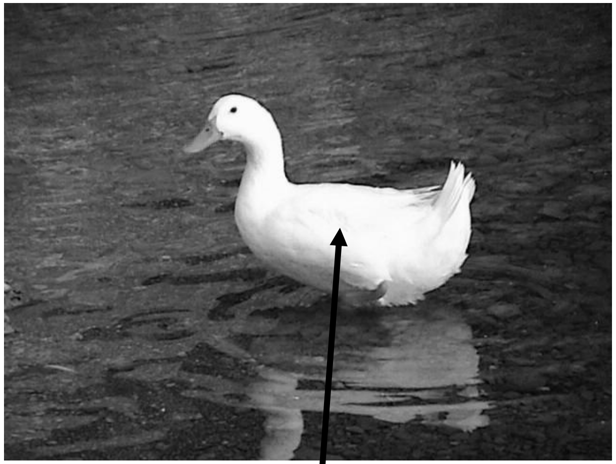
Region Growing

```
function B = RegionGrow(I, seed)

    [X,Y] = size(I);
    visited = zeros(X,Y);
    visited(seed) = 1;
    boundary = emptyQ;
    boundary.enQ(seed);

        while(~boundary.empty())
            nextPoint = boundary.deQ();
            if(include(nextPoint))
                visited(nextPoint) = 2;
                Foreach (x,y) in N(nextPoint)
                    if(visited(x,y) == 0)
                        boundary.enQ(x,y);
                        visited(x,y) = 1;
                    end
                end
            end
        end
```

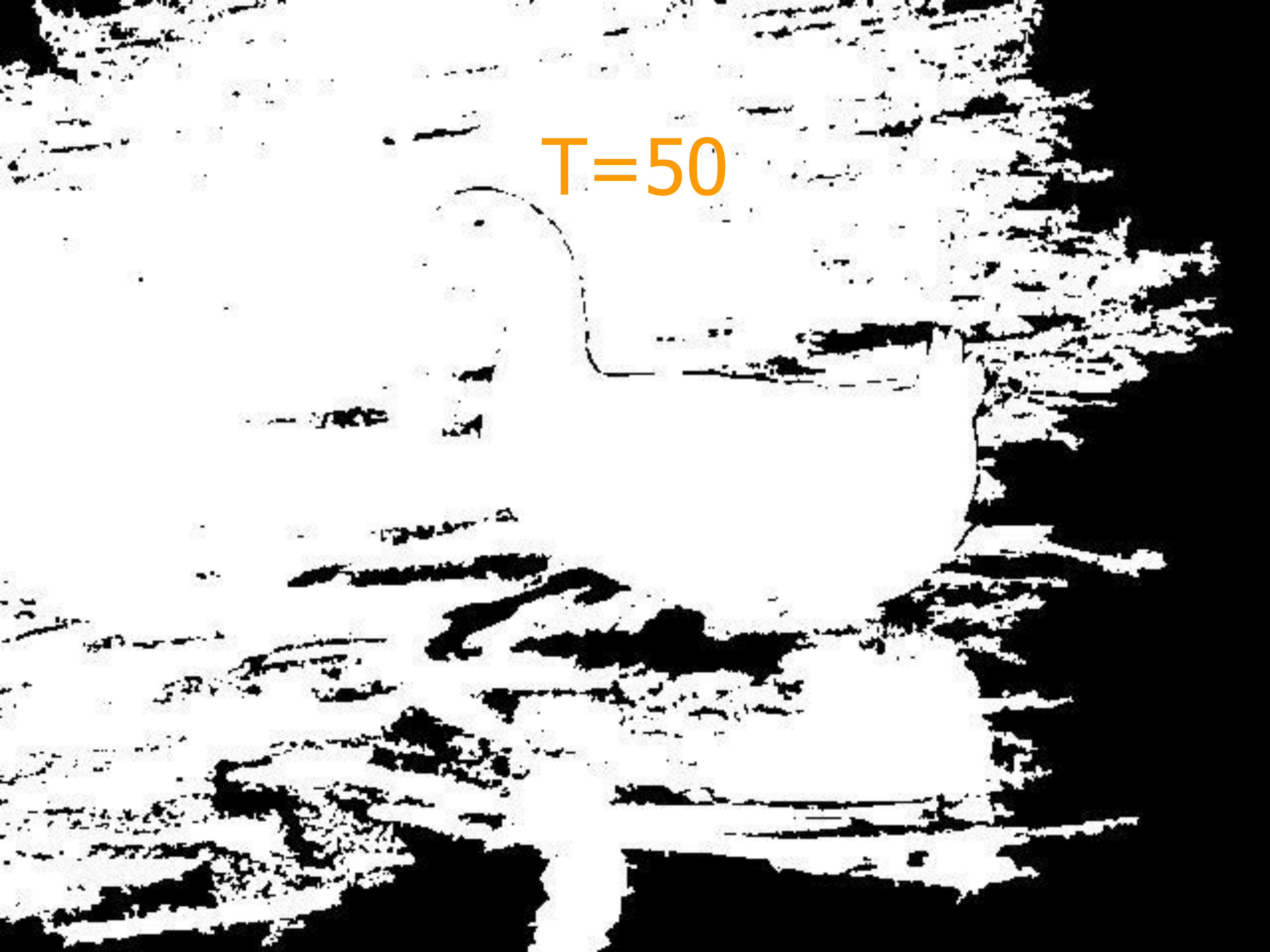
Region Growing example

- Pick a single seed pixel.
- Inclusion test is a simple grey level threshold:

```
function test = include(p)
    test = (p>=T);
```



Seed pixel


T=50

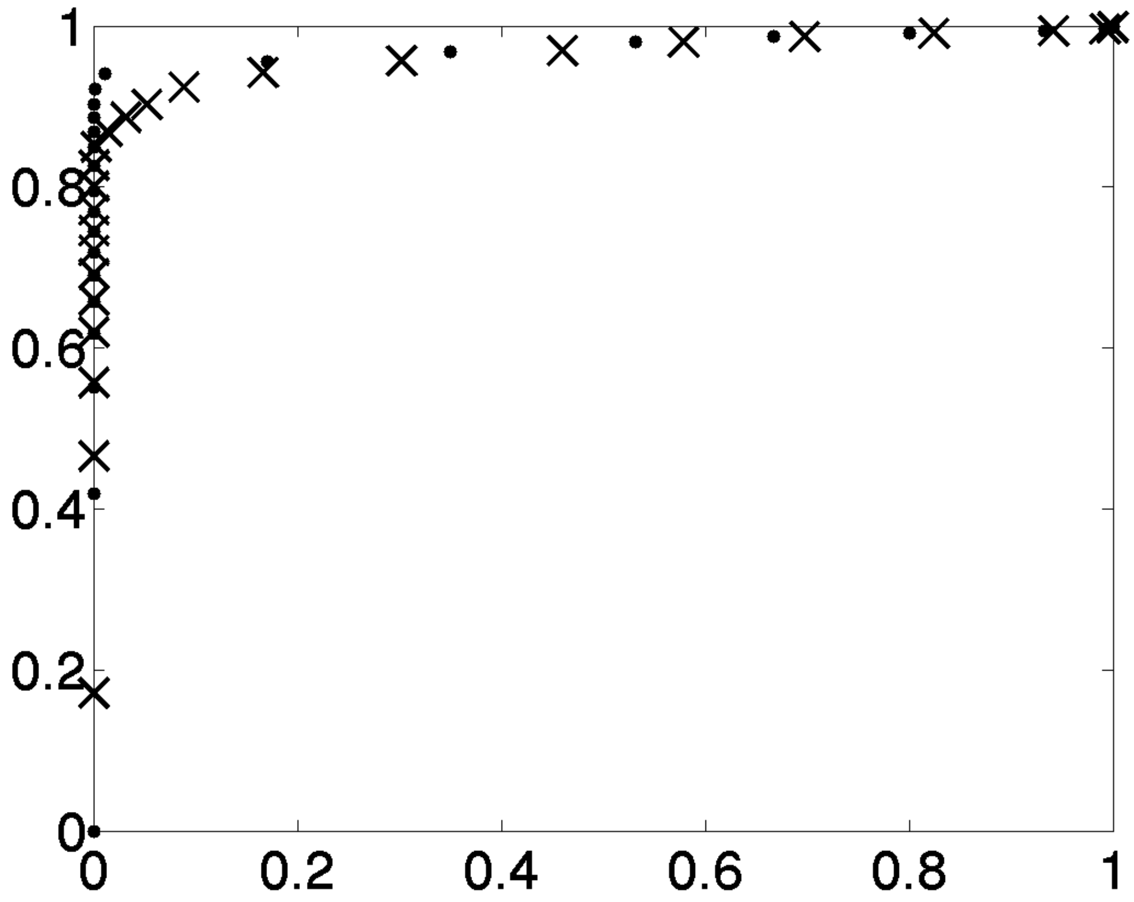
T=100

T=150

ROC Curve

• Region grower

× Thresholding



Implementation

- The region-growing algorithm above uses a *breadth-first search*.
- The connected-components algorithm above uses a *depth-first search*.
- Both algorithms can use either search procedure.
- Breadth-first search has more robust performance.

Variations

- Seed selection
- Inclusion criteria
- Boundary constraints and snakes

Seed selection

- Point and click seed point.
- Seed region
  - By hand
  - Automatically, e.g., from a conservative thresholding.
- Multiple seeds
  - Automatically labels the regions

Inclusion criteria

- Greylevel thresholding
- Greylevel distribution model
  - Use mean μ and standard deviation σ in seed region:
  - Include if $(I(x, y) - \mu)^2 < (n\sigma)^2$. Eg: $n = 3$.
  - Can update the mean and standard deviation after every iteration.
- Colour or texture information.

Snakes

- A snake is an *active contour*.
- It is a polygon, i.e., an ordered set of points joined up by lines.
- Each point on the contour moves away from the seed while its image neighbourhood satisfies an inclusion criterion.
- Often the contour has smoothness constraints.

Snakes

- The algorithm iteratively minimizes an energy function:

- $E = E_{tension} + E_{stiffness} + E_{image}$

- See Kass, Witkin, Terzopoulos, IJCV 1988

Example

Example program…

Split and Merge Algorithms

- This is a *global* rather than *binary* segmentation algorithm

Gray Level Variance

$$\frac{1}{N-1} \sum_{(r,c) \in Region} [I(r,c) - \overline{I}]^2$$

Split and Merge Algorithms

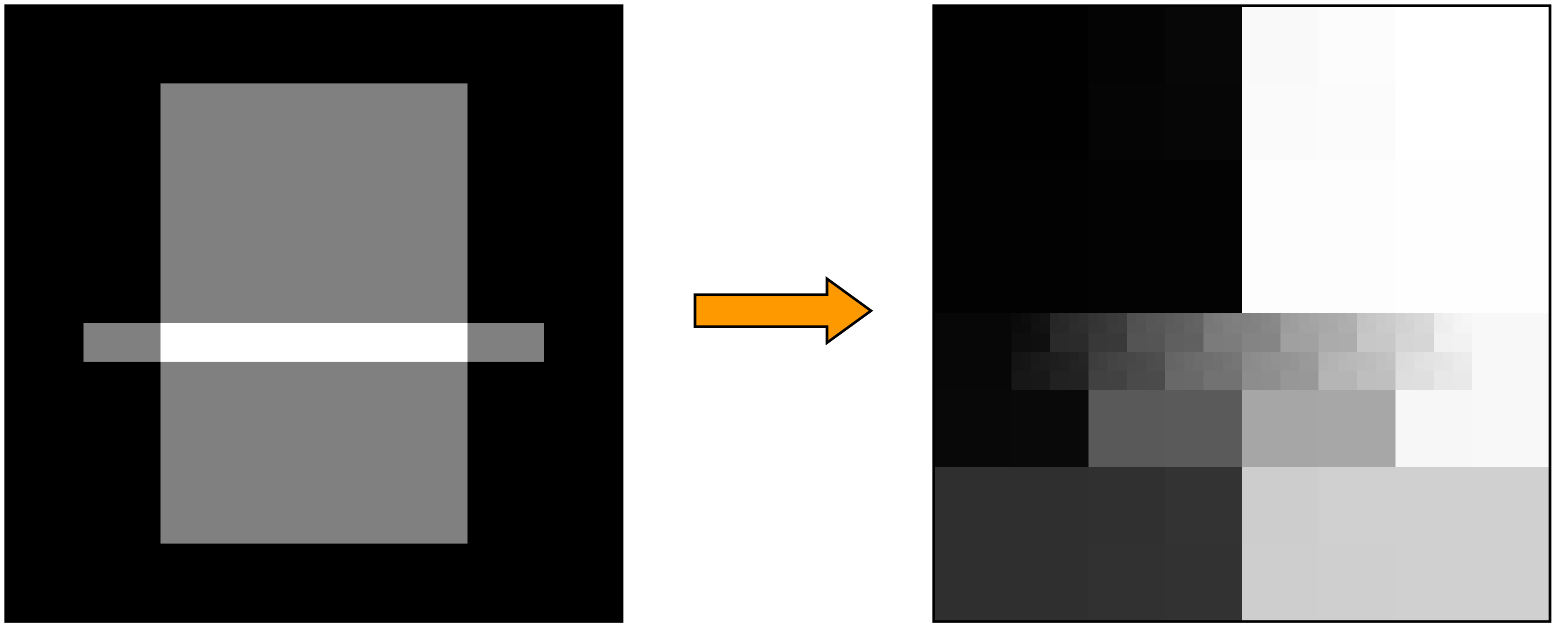
- This is a *global* rather than *binary* segmentation algorithm
- Recursively divide the image into homogeneous regions
- Merge adjacent regions that combine to a homogeneous region.

Splitting

```
% Splits an image I into homogeneous regions
% labelled in L.
function L = ImageSplit(I)
% Initialize.
[X,Y] = size(I);
L = ones(X,Y);
n = 2;
% Call recursive splitting subroutine
L = split(I, L, 1, X, 1, Y, n)
```

```
function L = split(I, L, xmin, xmax, ymin, ymax, n)
if(~homogeneous(I(xmin:xmax, ymin:ymax)))
    xSplit = (xmin+xmax)/2;  ySplit = (ymin+ymax)/2;
    % Top left
    L = split(I, L, xmin, xSplit, ymin, ySplit, n);
    % Top right
    L((xSplit+1):xmax, ymin:ySplit) = n;
    n = n + 1;
    L = split(I, L, (xSplit+1), xmax, ymin, ySplit,
n);
    % Bottom left
    L(xmin:xSplit, (ySplit+1):ymax) = n;
    n = n + 1;
    L = split(I, L, xmin, xSplit, (ySplit+1), ymax,
n);
    % Bottom right… similar…
end
```

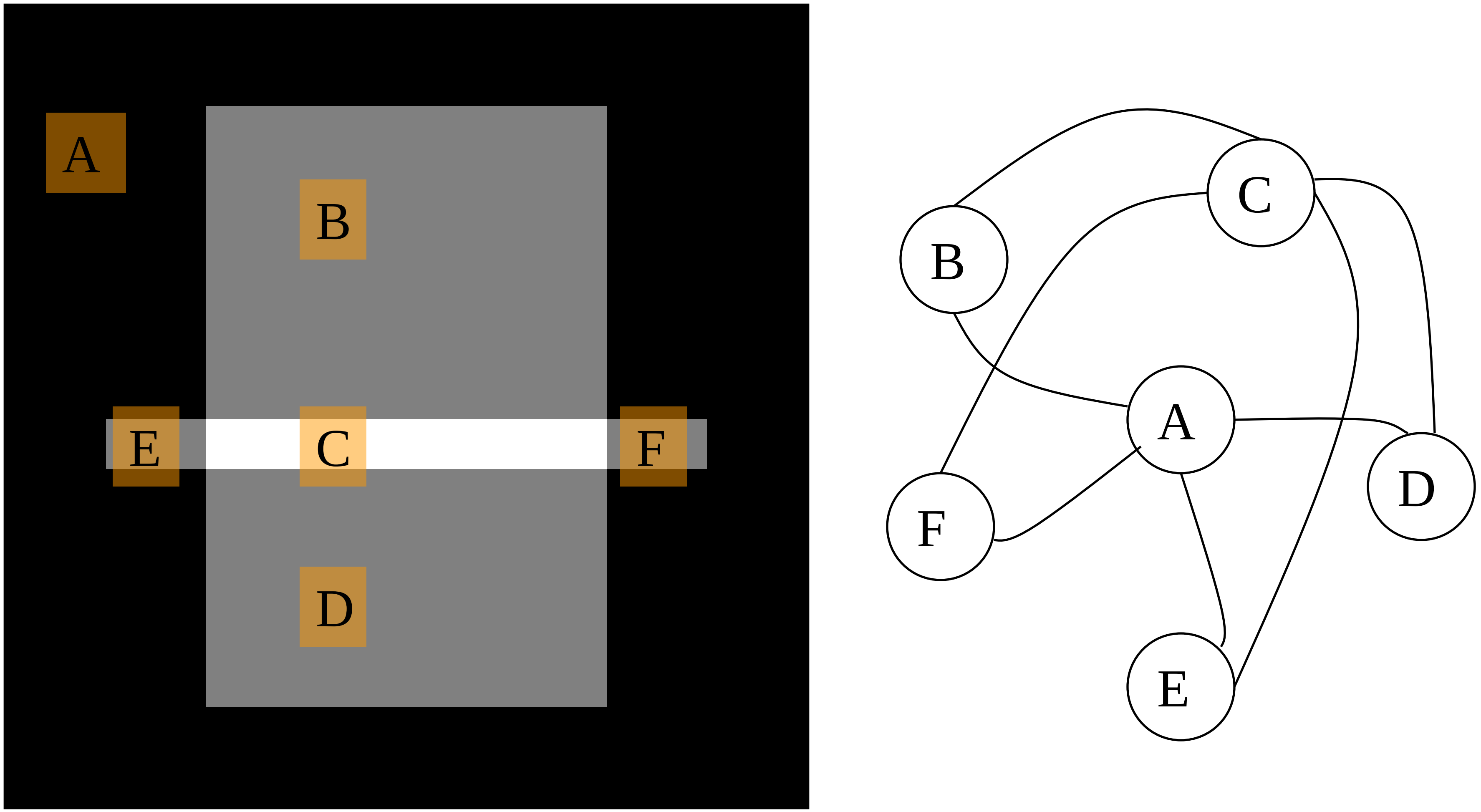
Splitting Example



Merging

- Build the *Region Adjacency Graph* (RAG).
- Merge adjacent regions that are homogeneous.
- Update RAG.
- Repeat to convergence

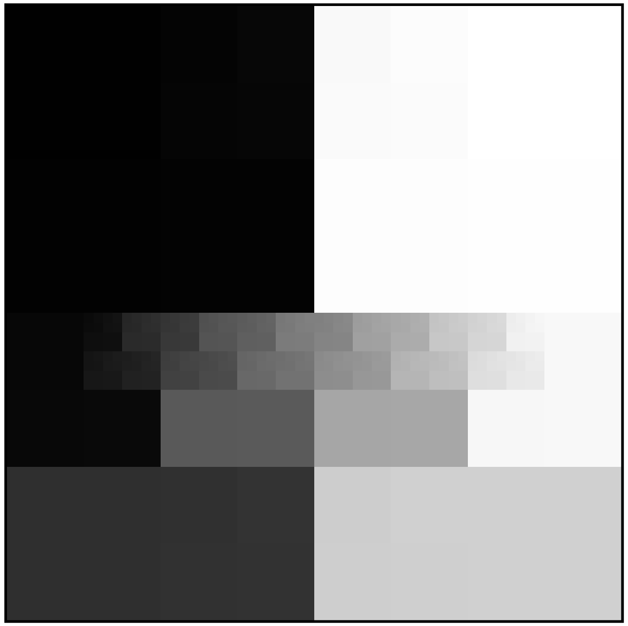
RAG Example



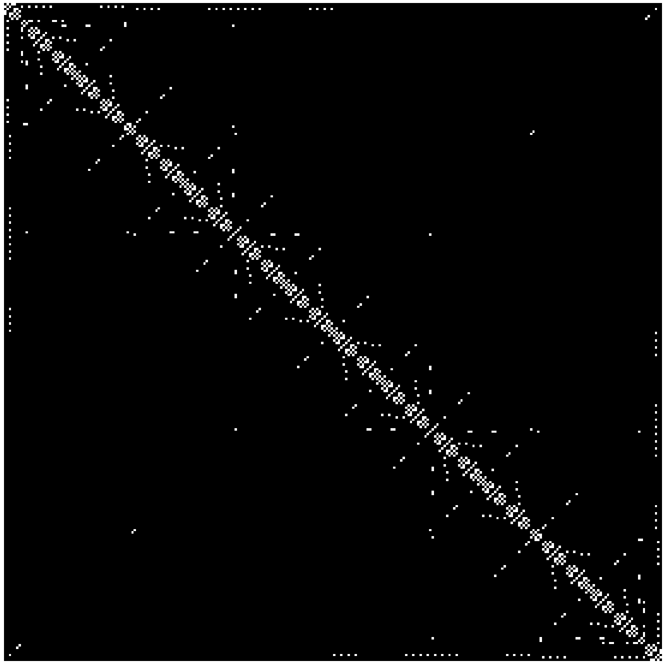
Computing the RAG

```
function rag = RegionAdjacencyGraph(L)
numRegions = max(max(L));  [X,Y] = size(L);
rag = zeros(numRegions, numRegions);
For each (x1, y1) in L
    r1 = L(x1, y1);
    For each (x2, y2) in N(x1, y1)
        r2 = L(x2, y2);
        if(r1 ~= r2)
            rag(r1, r2) = rag(r2, r1) = 1;
        end
    end
end
```

RAG Example



274 Regions



Region Merging

```
numRegions = max(max(L));
done = 0;
while(~done)
    done = 1;
    for i=1:(numRegions-1)
        for j=(i+1):numRegions
            if(rag(i,j))
                combinedR = I(find(L == i | L == j));
                if(homogeneous(combinedR))
                    [L, rag] = merge(i, j, rag, L);
                    done = 0;
                end
… end
```

Region Merging

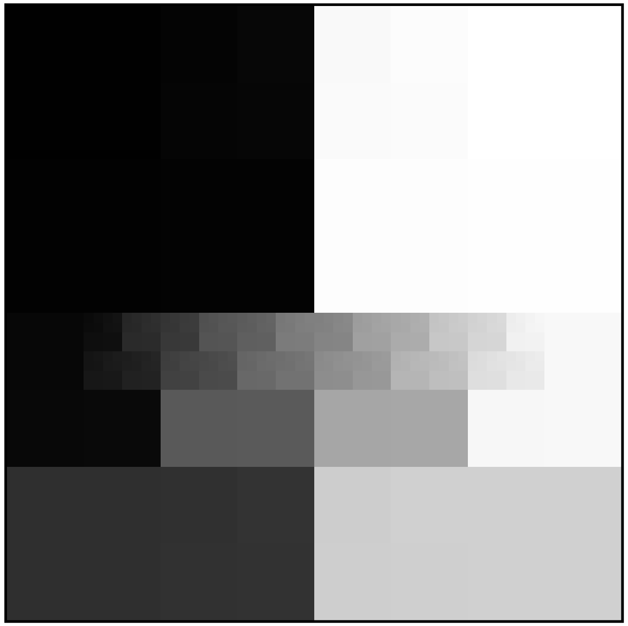
```
function [L, rag] = merge(label1, label2, rag, L)
% Merge the regions in the labelled image.
L(find(L == label2)) = label1;
% Combine the adjacency of the two regions.
rag(label1,:) = rag(label1,:) | rag(label2,:);
rag(:,label1) = rag(:,label1) | rag(:,label2);
% Make sure the combined region1 is not self-adjacent.
rag(label1, label1) = 0;
% Remove all adjacency to region2
rag(label2,:) = 0;
rag(:,label2) = 0;
```

Re-labelling

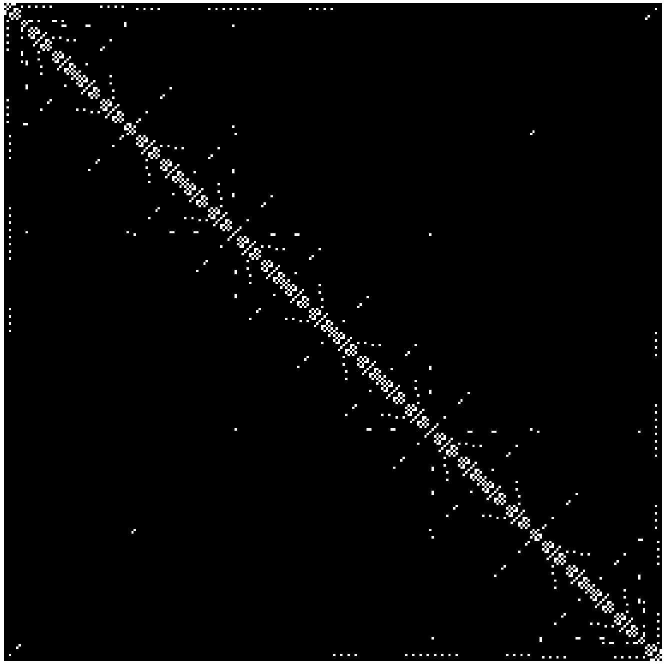
- Final step relabels L.

- Change each integer region label so that the set of labels is a contiguous set of labels starting at 1.

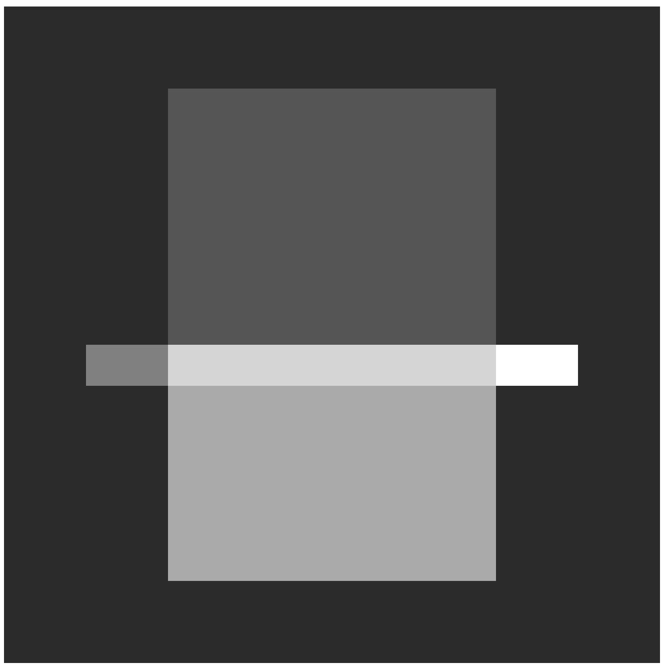
Merging Example - Before



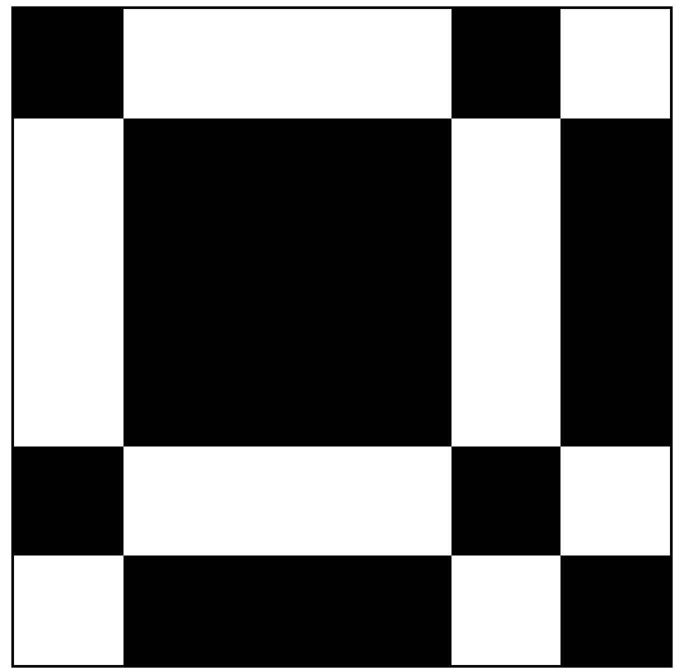
274 Regions



Merging Example - After



6 Regions



Split and merge example

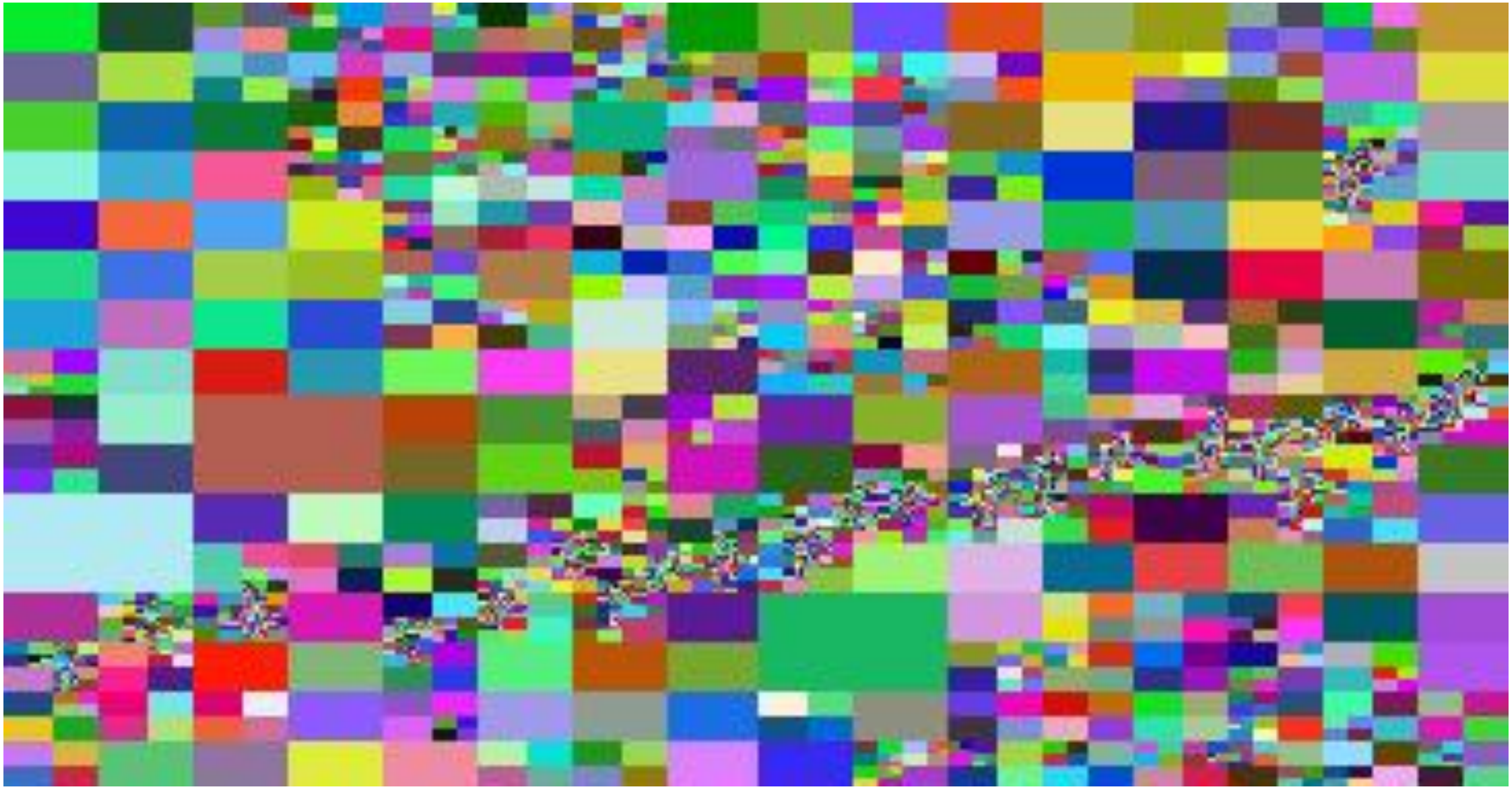
Split and merge example

- Regions are homogeneous if

Max. greylevel – min. greylevel < T.

- T = 20 for split.
- T = 100 for merge.

Split and merge example

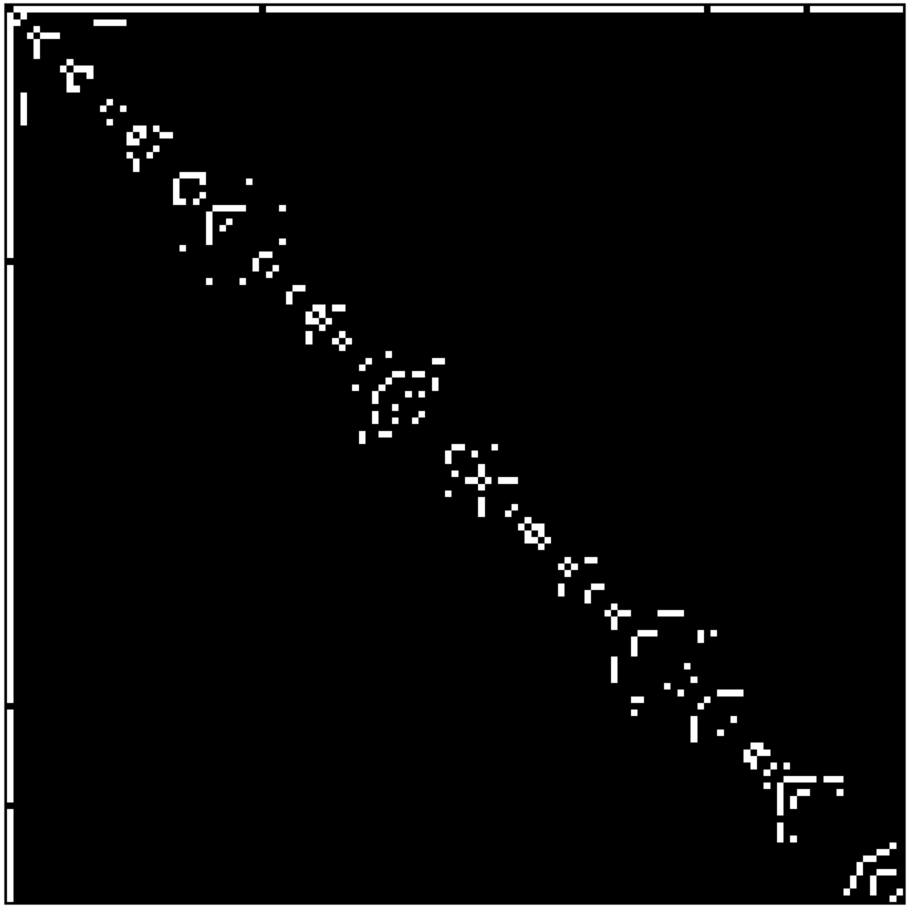


4087 regions

Split and merge example

135 regions

Split and merge example



Problems and Improvements

- Inefficiency
- Alternative data structures?

- Consistency.

- Homogeneity criteria
    - Statistical region similarity measures

But if this pixel is foreground…

Markov Random Fields

- In a Markov Random Field, the label in each pixel depends on the label in neighbouring pixels.
- We assign a probability to each configuration of pixels in a neighbourhood.
- We maximize the probability of the segmentation given the observed image.

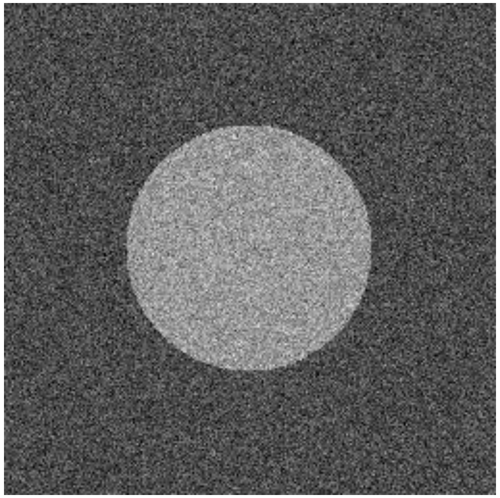
Neighborhood Probability

- Simple model for 4-neighbourhood.

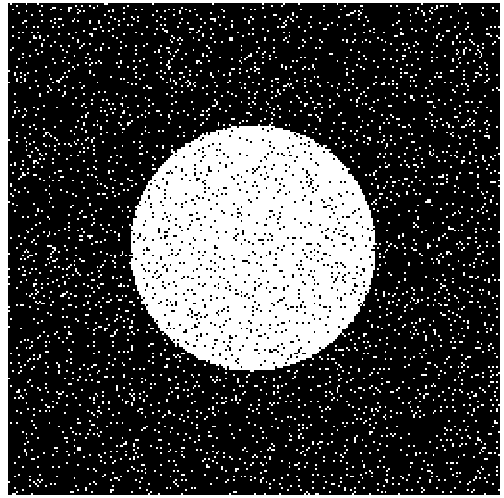
- $$P(x_1, x_2) = \begin{cases} 0.99 & x_1 = x_2 \\ 0.01 & x_1 \neq x_2 \end{cases}$$

- $$P(x \mid N(x)) = P(x, x_1, x_2, x_3, x_4)$$
  $$= P(x, x_1)P(x, x_2)P(x, x_3)P(x, x_4)$$

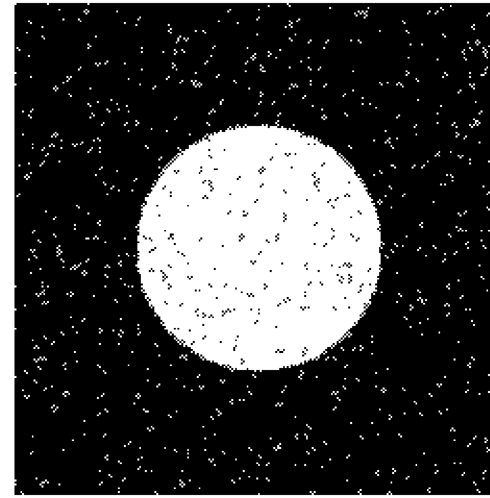
MRF Segmentation: Iter. Cond. Modes



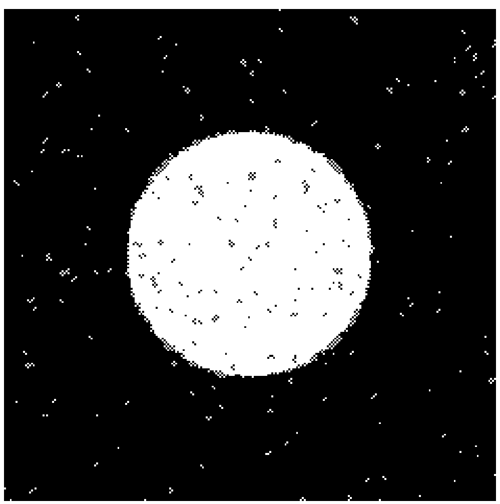
Image



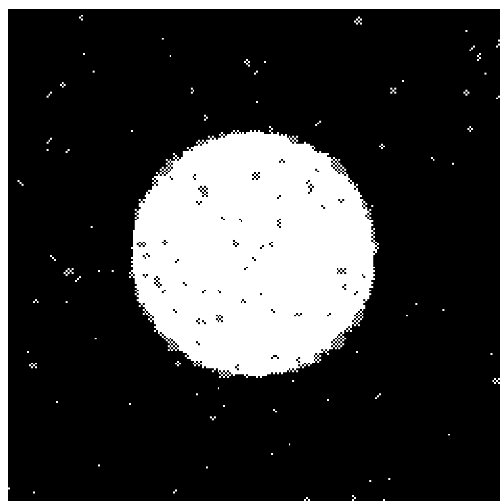
Initial segmentation



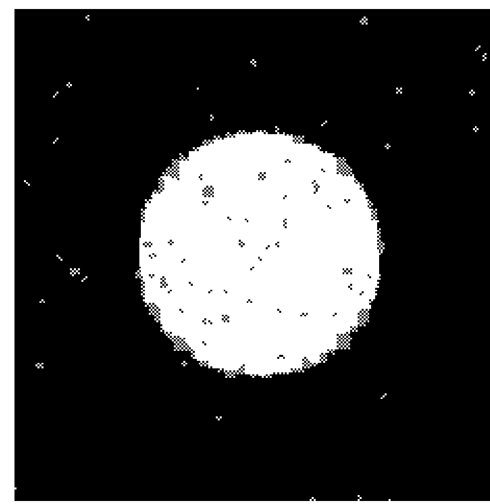
ICM Iter 1



ICM Iter 2



ICM Iter 4



ICM Iter 8

Model vs. Algorithm

- Find the set of labels that maximizes:
  $$P(X \mid I) = \prod P(y \mid x) P(x) P(x \mid N(x))$$

- $L(X) = \sum \log P(y \mid x) + \log P(x) + \log P(x \mid N(x))$
- Initialize with thresholded image
- Iteratively replace each label if it increases $L(I)$. "Greedy" algorithm.
- MCMC, Simulated Annealing, or Graph Cuts better.

MRF Extension

- Learn the within class probabilities using an EM algorithm (see Machine Vision class).
- Extends naturally to more than two classes.

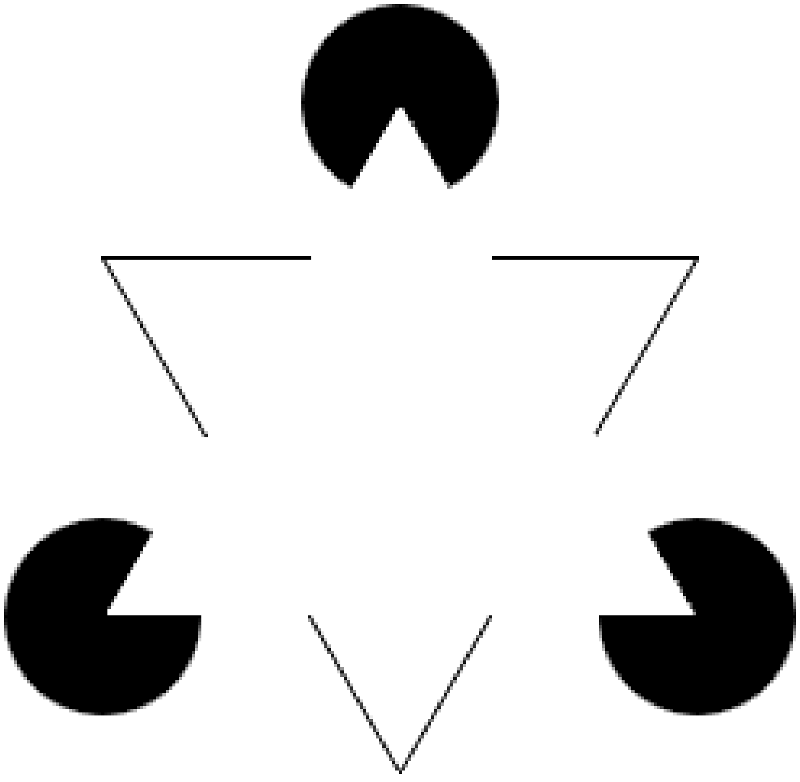
Summary

- We have looked at several segmentation algorithms:
  - Thresholding and connected components.
  - Region growing
  - Split and merge
  - Snakes and MRFs
- Used ROC analysis to compare the performance of simple systems.

Summary

- Segmentation is hard!
- But it is easier if you know what you are doing!
  - Is the segmentation task binary or global?
  - What are the regions of interest?
  - How accurately must the algorithm locate the region boundaries?
- Research problems remain!

Kanizsa Triangle



Illusory Contours: Kanizsa, G. (1955), Rivista di Psicologia 49(1): 7–30