
©1994 Encyclopaedia Britannica, Inc.

Visual Computing:

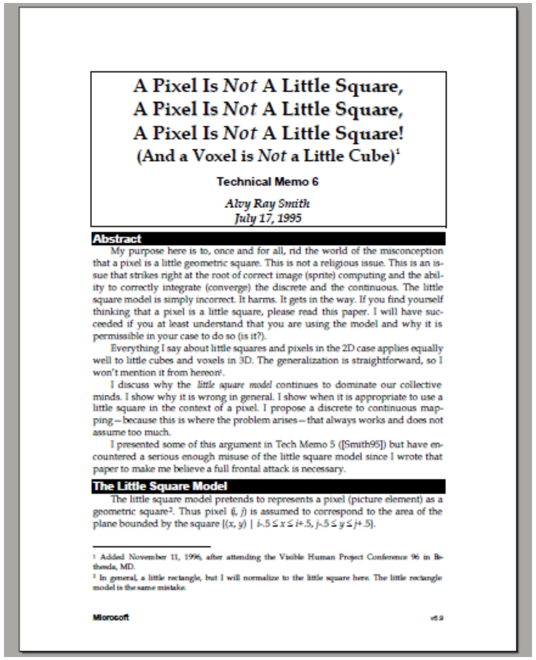
Image Segmentation

Prof. Marc Pollefeys

ETH

inf | Informatik
Computer Science

Recap



A Pixel Is *Not* A Little Square,
A Pixel Is *Not* A Little Square,
A Pixel Is *Not* A Little Square!
(And a Voxel is *Not* a Little Cube)[1]

Technical Memo 6

Alvy Ray Smith
July 17, 1995

Abstract

My purpose here is to, once and for all, rid the world of the misconception that a pixel is a little geometric square. This is not a religious issue. This is an issue that strikes right at the root of correct image (sprite) computing and the ability to correctly integrate (converge) the discrete and the continuous. The little square model is simply incorrect. It harms. It gets in the way. If you find yourself thinking that a pixel is a little square, please read this paper. I will have succeeded if you at least understand that you are using the model and why it is permissible in your case to do so (is it?).

Everything I say about little squares and pixels in the 2D case applies equally well to little cubes and voxels in 3D. The generalization is straightforward, so I won't mention it from hereon[1].

I discuss why the *little square model* continues to dominate our collective minds. I show why it is wrong in general. I show when it is appropriate to use a little square in the context of a pixel. I propose a discrete to continuous mapping—because this is where the problem arises—that always works and does not assume too much.

I presented some of this argument in Tech Memo 5 ([Smith95]) but have encountered a serious enough misuse of the little square model since I wrote that paper to make me believe a full frontal attack is necessary.

The Little Square Model

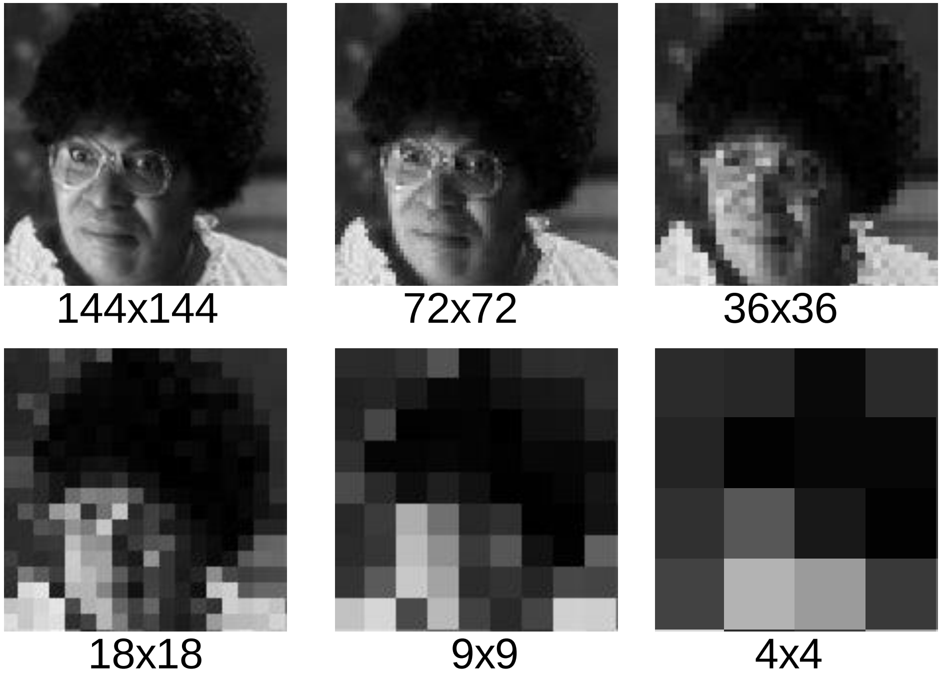
The little square model pretends to represents a pixel (picture element) as a geometric square[2]. Thus pixel (i, j) is assumed to correspond to the area of the plane bounded by the square $\{(x, y) \mid i\text{-}.5 \le x \le i\text{+}.5,\ j\text{-}.5 \le y \le j\text{+}.5\}$.

[1] Added November 11, 1996, after attending the Visible Human Project Conference 96 in Bethesda, MD.

[2] In general, a little rectangle, but I will normalize to the little square here. The little rectangle model is the same mistake.

Microsoft v5.9

Geometric resolution



Radiometric resolution

Aliasing and SNR

- What is the disadvantage of low sampling resolution?
- What is the disadvantage of high sampling resolution?
- Lossless vs. Lossy
  - Name some formats?

Unassessed Assignment

Use python to change the geometric and radiometric quantization resolution in one of your images. For each level of sampling and quantization, plot the image function, as in slides 71 & 72, and compare the approximations to the true intensity function that you get at each level.

Usual quantization intervals

- Grayscale image
  - 8 bit = 2^8 = 256 grayvalues
- Color image RGB (3 channels)
  - 8 bit/channel = 2^24 = 16.7M colors
- 12bit or 16bit from some sensors
- Nonlinear, for example log-scale

Photo: Paulo Barcellos Jr.

ETH

Image Noise

- A common model is *additive Gaussian* noise:

$$I(x,y) = f(x,y) + c$$

where $c \sim N(0,\sigma^2)$. So that $p(c) = (2\pi\sigma^2)^{-1} e^{-c^2/2\sigma^2}$

- Poisson noise: (shot noise) $p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$

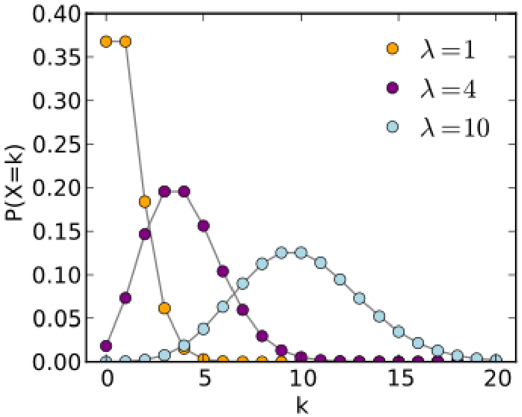
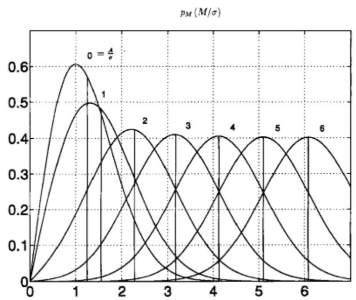


Image Noise



- Rician noise: (appears in MRI)

$$p(I)=\frac{I}{\sigma^2}\exp\left(\frac{-(I^2+f^2)}{2\sigma^2}\right)I_0\left(\frac{If}{\sigma^2}\right)$$

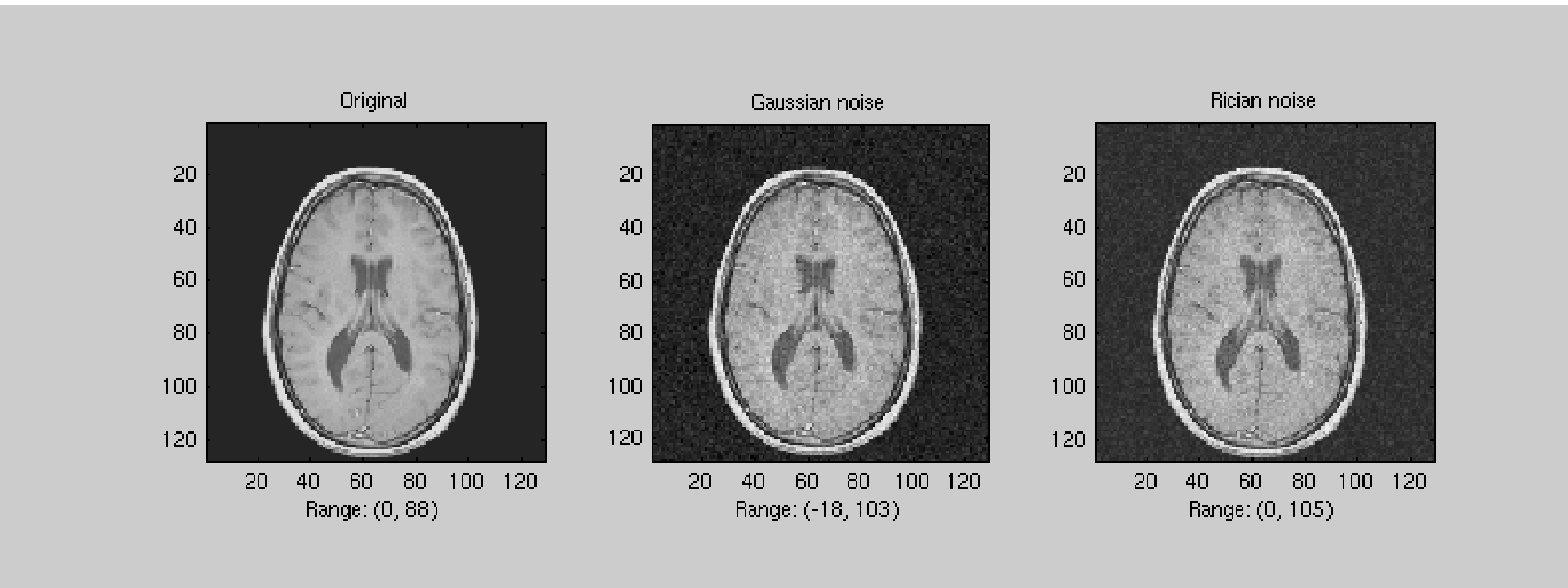
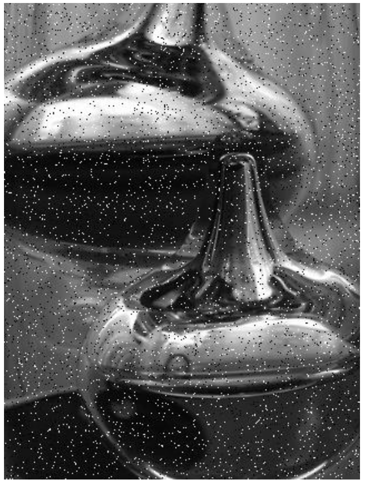


Image Noise

- Multiplicative noise:

$$I = f + fc$$



- Quantization errors
- Impulse "salt-and-pepper" noise

- The *signal to noise ratio (SNR) s* is an index of image quality

$$s = \frac{F}{\sigma}, \text{ where } F = \frac{1}{XY}\sum_{x=1}^{X}\sum_{y=1}^{Y} f(x,y)$$

Often used instead: *Peak Signal to Noise Ratio* (PSNR) $s_{peak} = \frac{F_{\max}}{\sigma}$

Colour Images

R

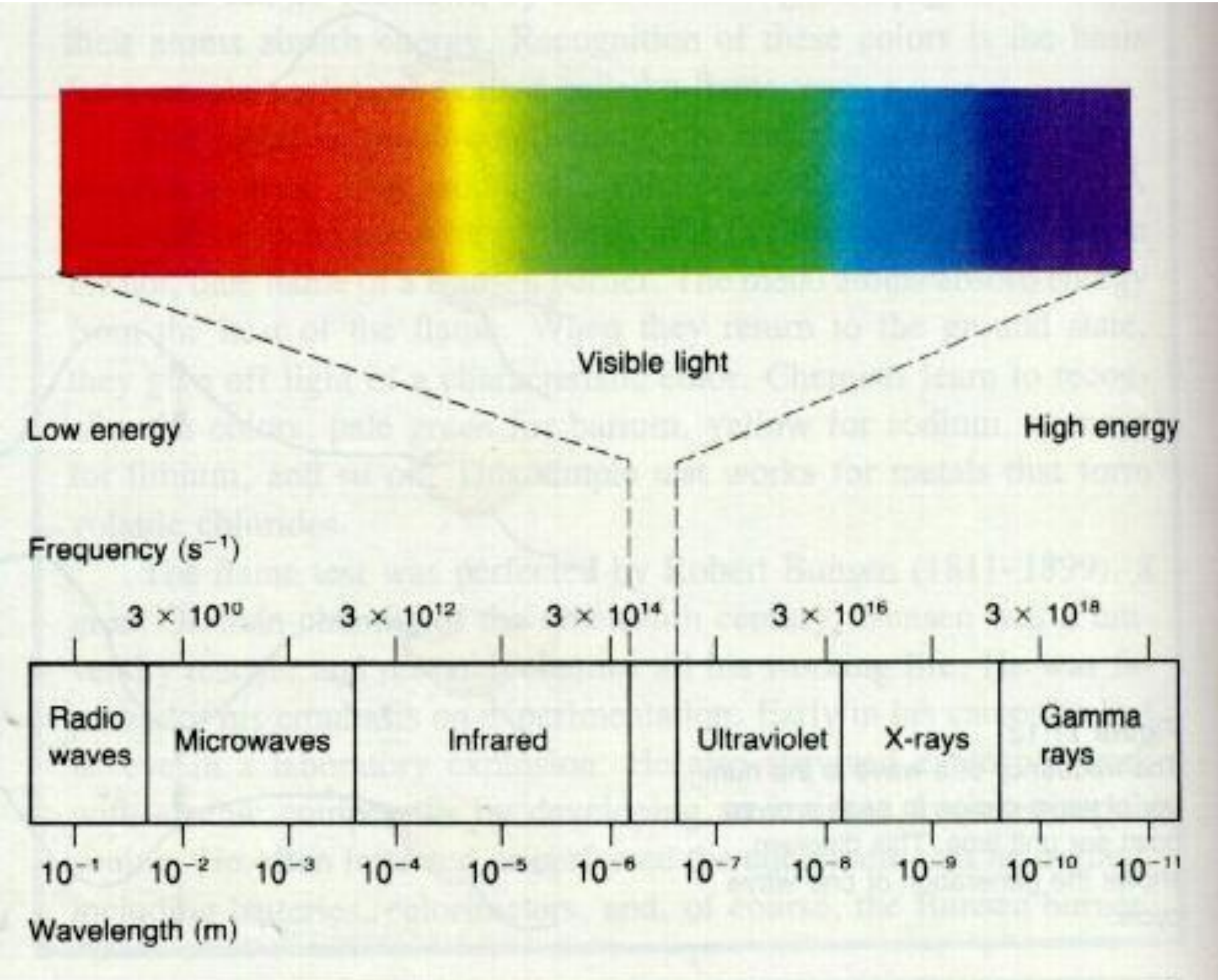
+

G

+

B

=



Visible light
Low energy
High energy
Frequency (s⁻¹)
3 × 10¹⁰
3 × 10¹²
3 × 10¹⁴
3 × 10¹⁶
3 × 10¹⁸
Radio waves
Microwaves
Infrared
Ultraviolet
X-rays
Gamma rays
10⁻¹
10⁻²
10⁻³
10⁻⁴
10⁻⁵
10⁻⁶
10⁻⁷
10⁻⁸
10⁻⁹
10⁻¹⁰
10⁻¹¹
Wavelength (m)

Color cameras

We consider 3 concepts:

1. Prism (with 3 sensors)
2. Filter mosaic
3. Filter wheel

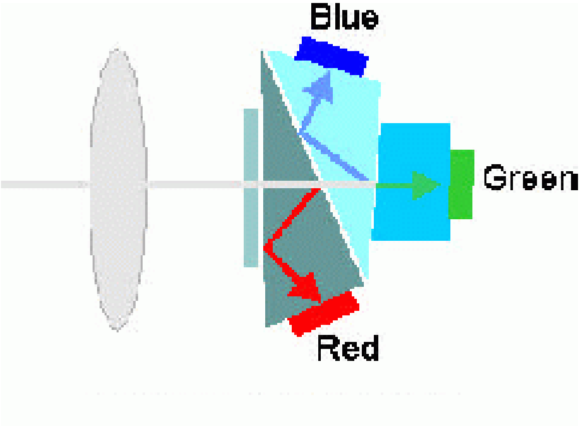
… and X3

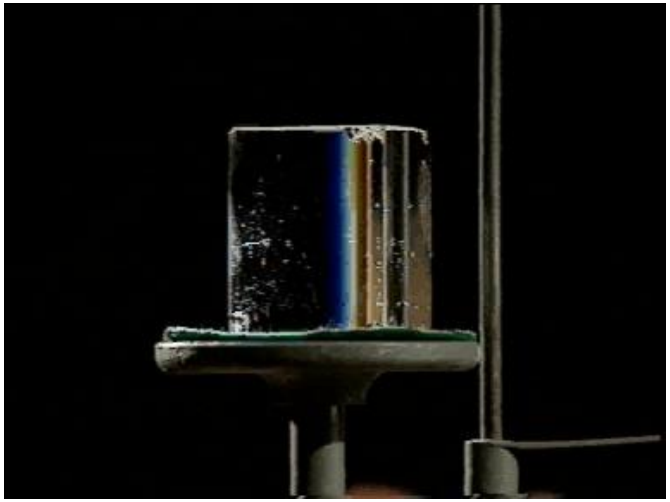
Prism color camera

Separate light in 3 beams using dichroic prism

Requires 3 sensors & precise alignment

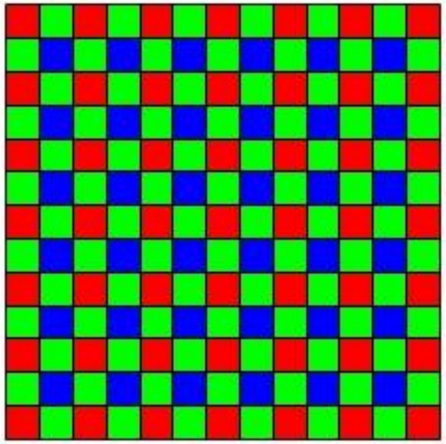
Good color separation



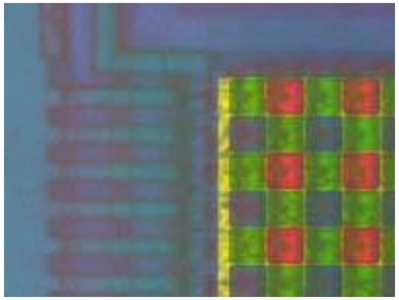


Filter mosaic

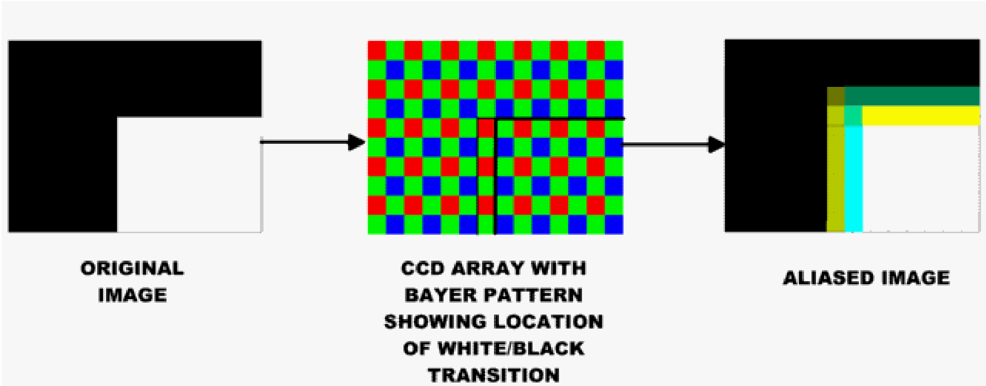
Coat filter directly on sensor



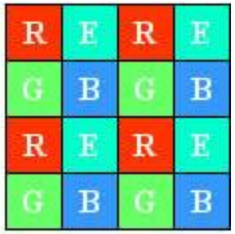
Bayer filter



Demosaicing (obtain full colour & full resolution image)



More colors:



Filter wheel

Rotate multiple filters in front of lens

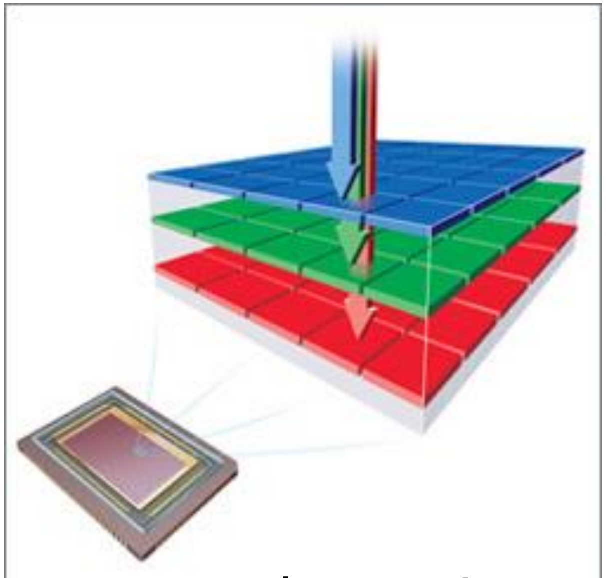
Allows more than 3 colour bands

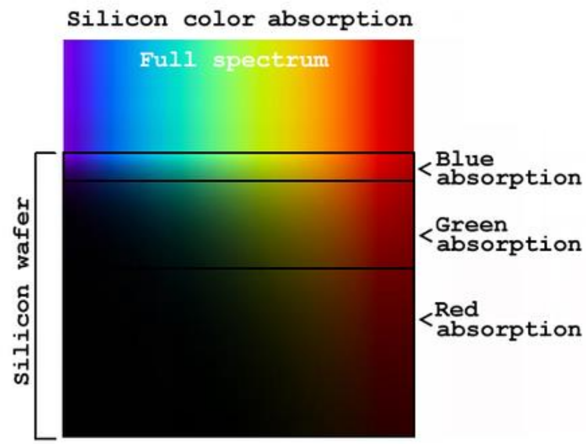
Only suitable for static scenes

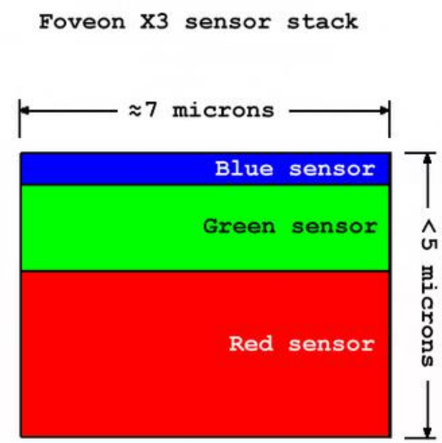
Prism vs. mosaic vs. wheel

approach	Prism	Mosaic	Wheel
# sensors	3	1	1
Separation	High	Average	Good
Cost	High	Low	Average
Framerate	High	High	Low
Artefacts	Low	Aliasing	Motion
Bands	3	3	3 or more
	High-end cameras	Low-end cameras	Scientific applications

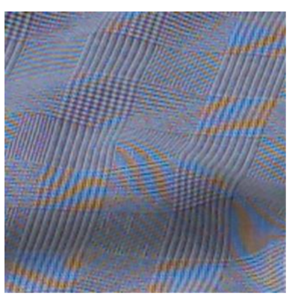
color CMOS sensor
Foveon's X3





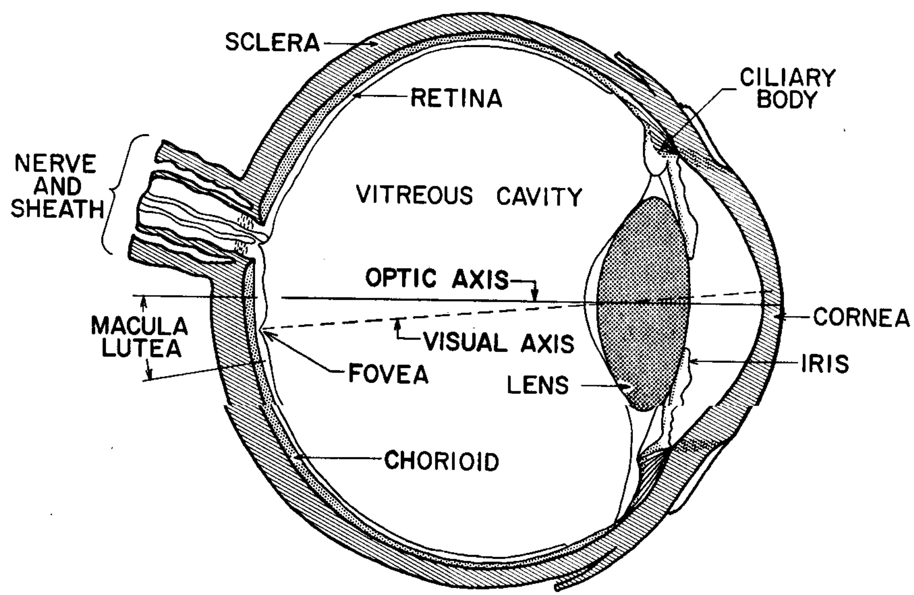


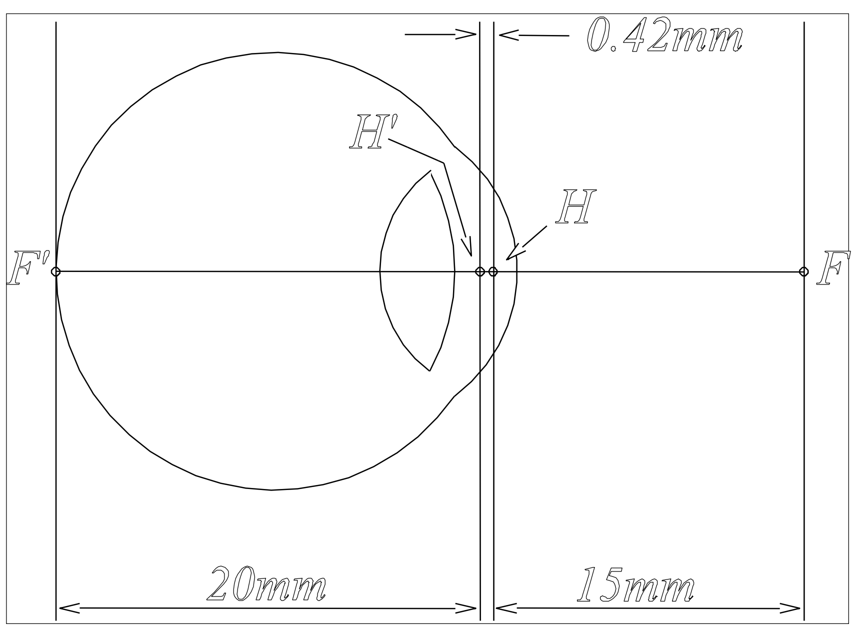
better image quality



Reproduced by permission, the American Society of Photogrammetry and Remote Sensing. A.L. Nowicki, “Stereoscopy.” Manual of Photogrammetry, Thompson, Radlinski, and Speert (eds.), third edition, 1966.

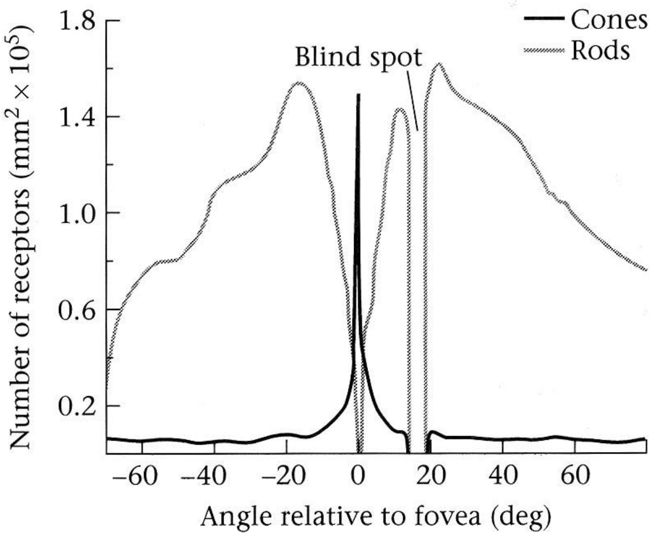
The Human Eye



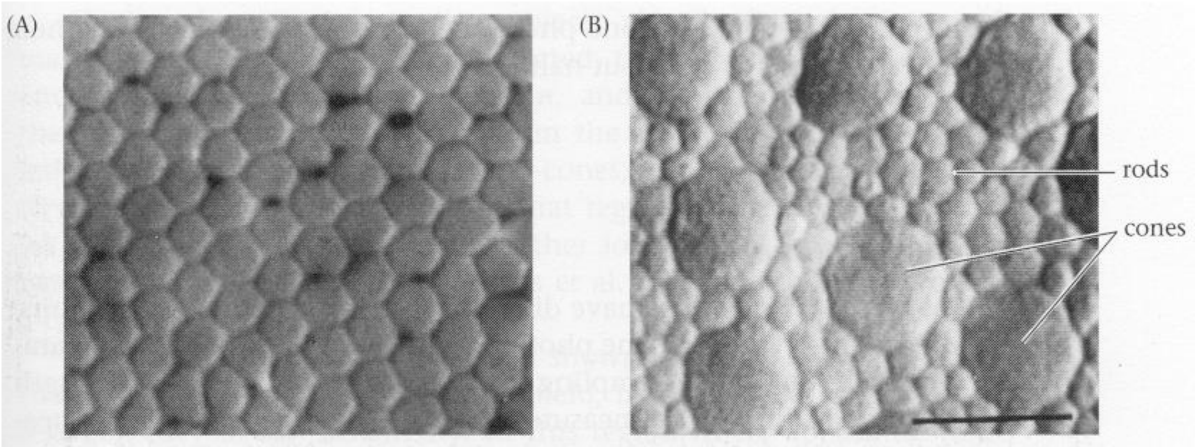


Helmoltz’s Schematic Eye

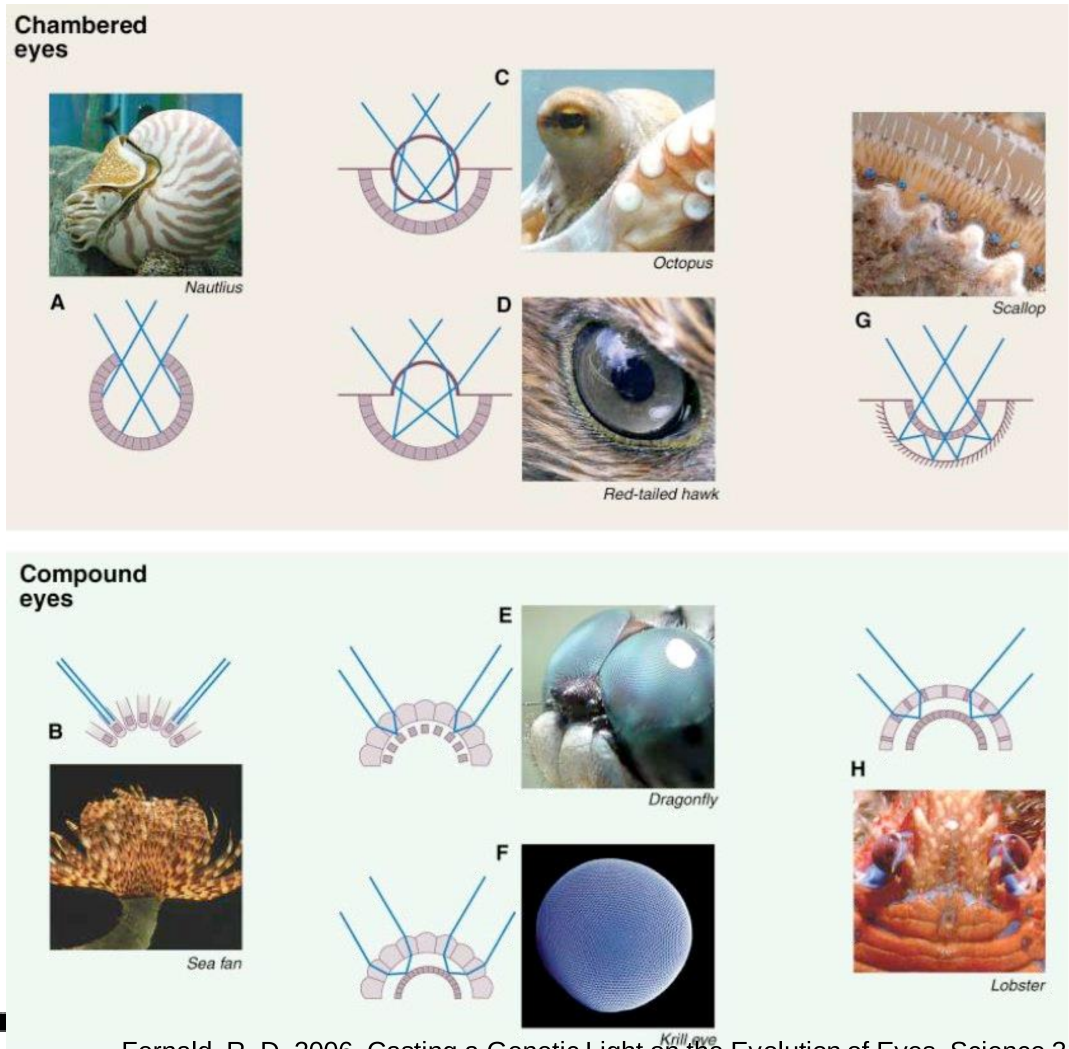
The distribution of rods and cones across the retina



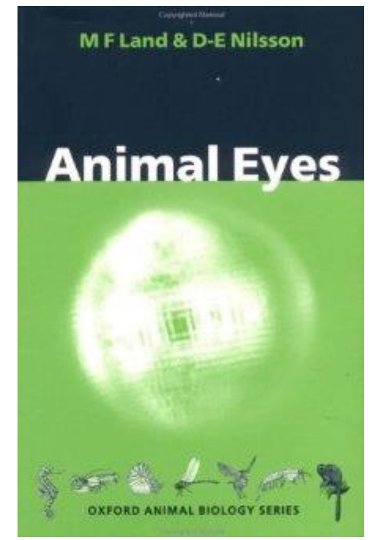
Cones in the fovea



More eyes in nature…



More info:



Fernald, R. D. 2006. Casting a Genetic Light on the Evolution of Eyes. Science 313, 1914-1918

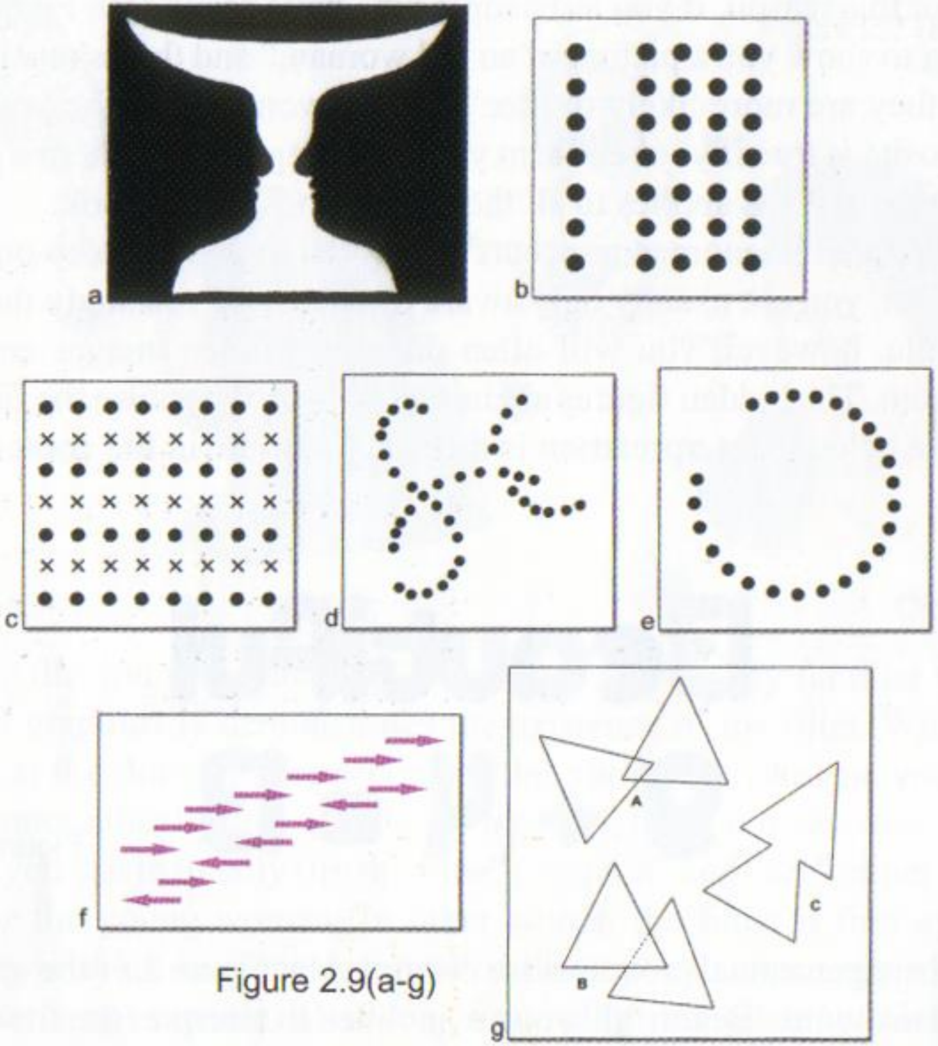


Figure 2.9(a-g)

Gestalt Phenomena

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

Young Lady or An Old Hag?

ETH

What is Image Segmentation?

Segmentation is the ultimate classification problem.
Once solved, Computer Vision is solved.

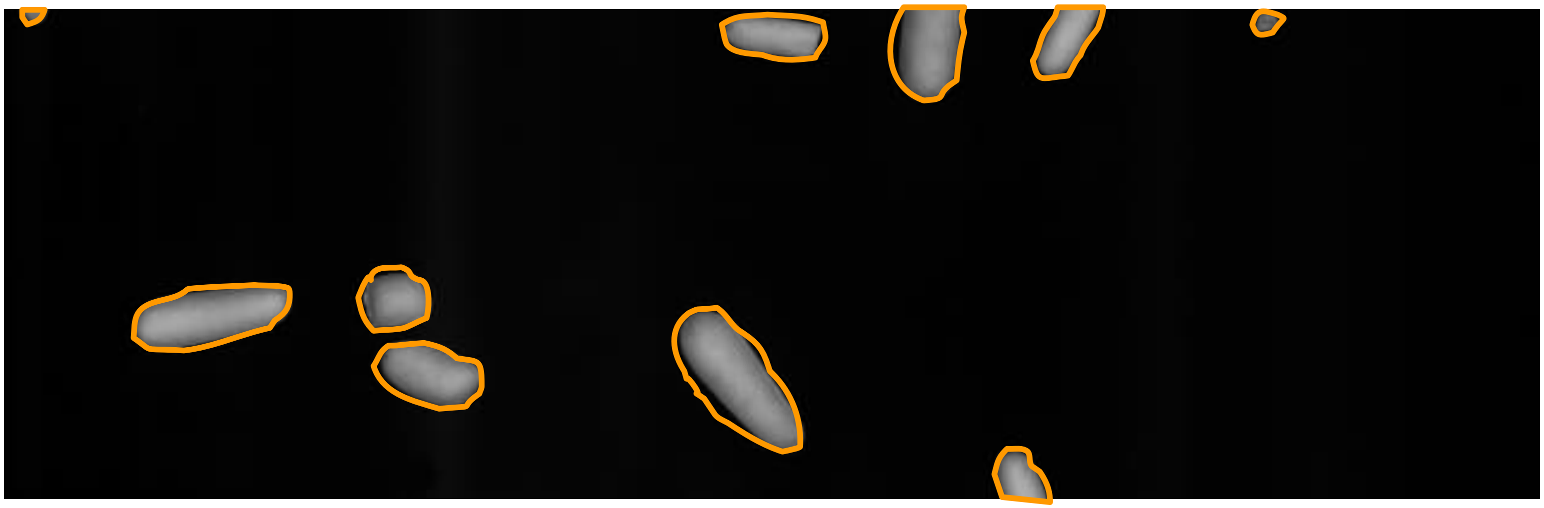
What is Image Segmentation?

- 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

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

ETH

How should I segment this?

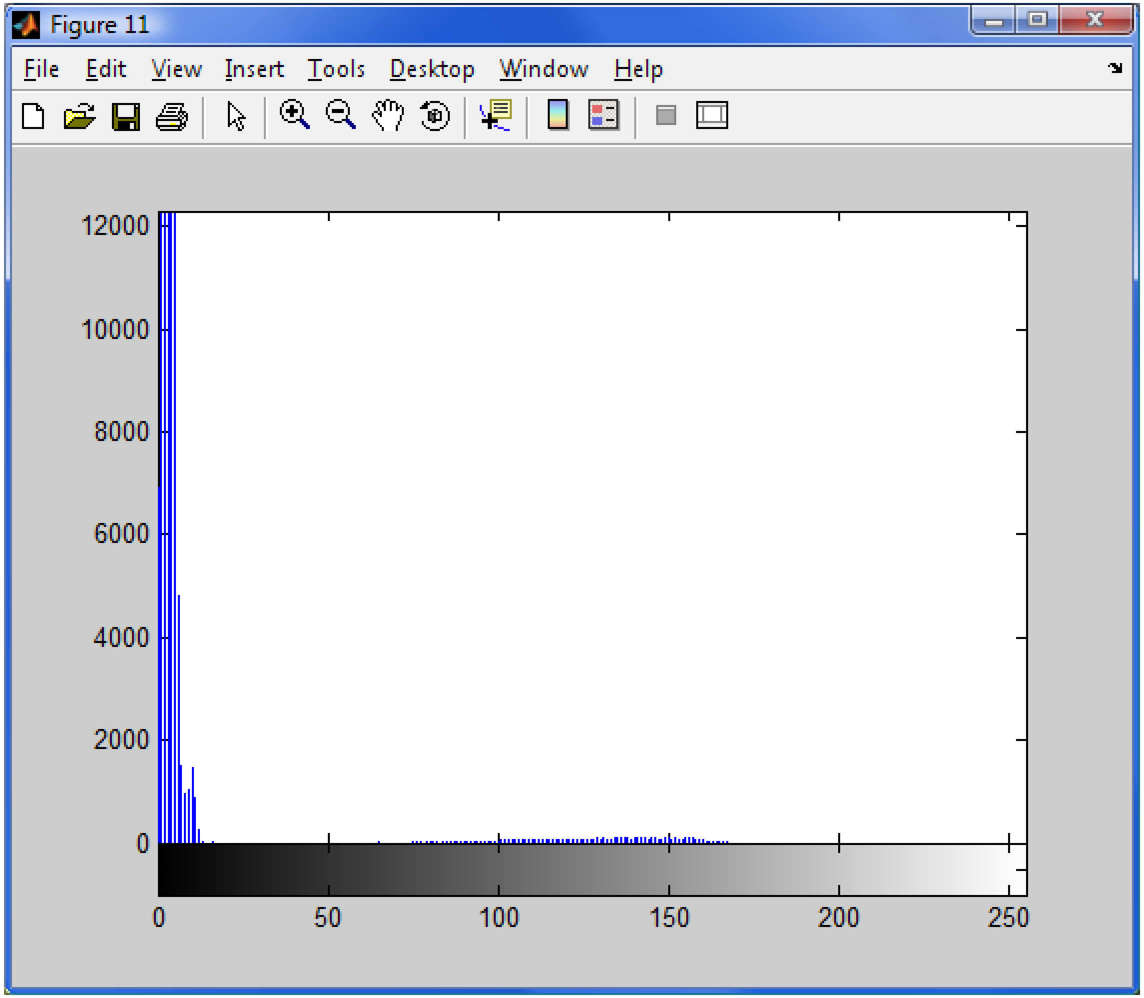


Exclude dark pixels?

```
img = cv2.imread('BlobsIP.png')
cv2.imshow('BlobsIP', img)
cv2.waitKey(0)
img.shape    --> [ 244   767  3 ]
hist = np.histogram(img, bins=256)
cv2.imshow('Histogram', hist)
cv2.waitKey(0)
cv2.imshow('Mask', img[:,:,1] > 20)
cv2.waitKey(0)
```

Histogram

Pixels



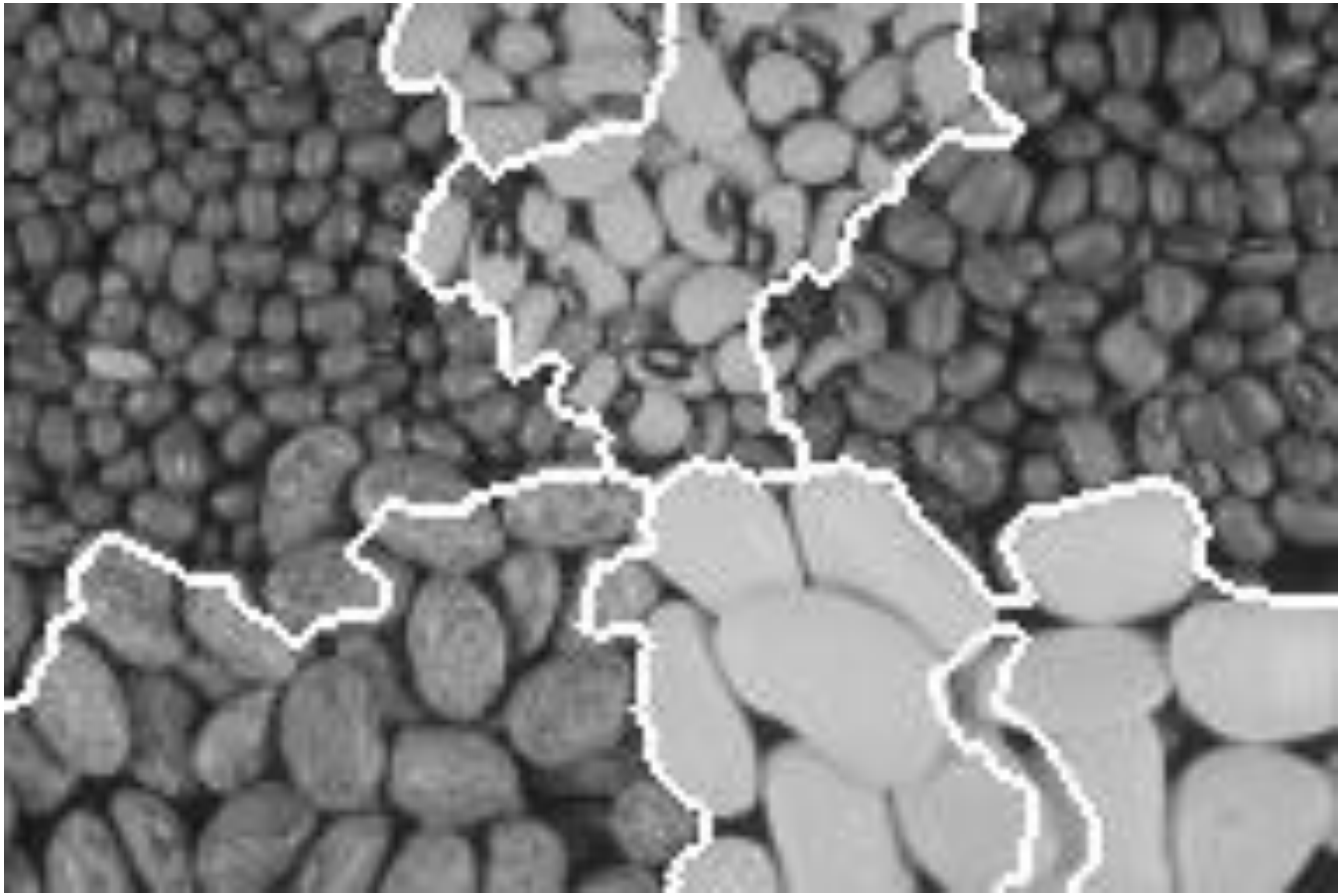
Grayscales [0,255]

How should I segment this?

Segmentation Quality

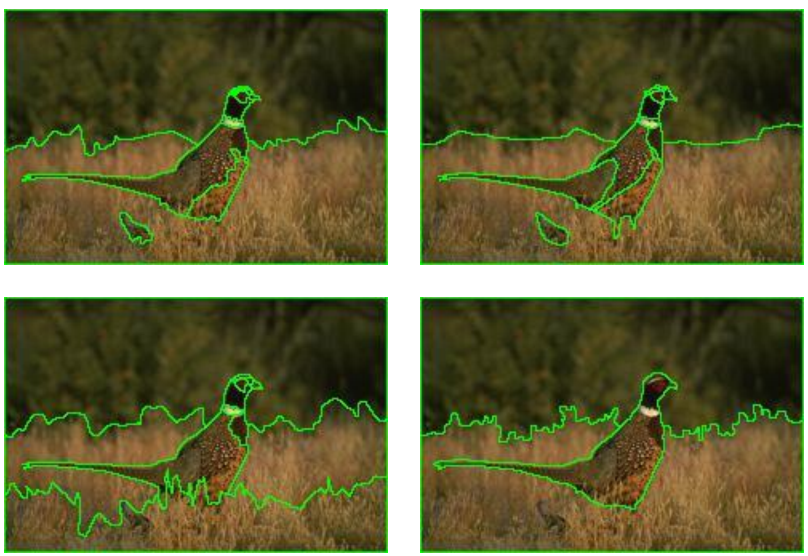
- The quality of a segmentation depends on what you want to do with it.
- Segmentation algorithms must be chosen and evaluated with an application in mind.

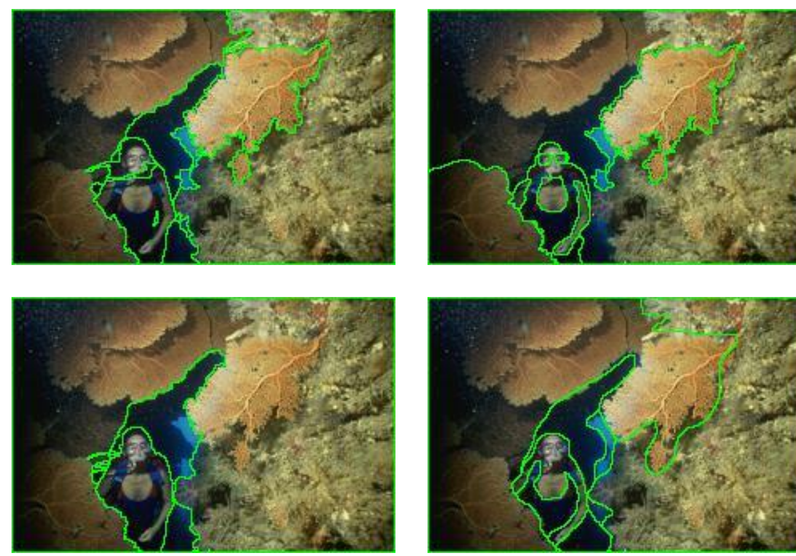
Segmentation example



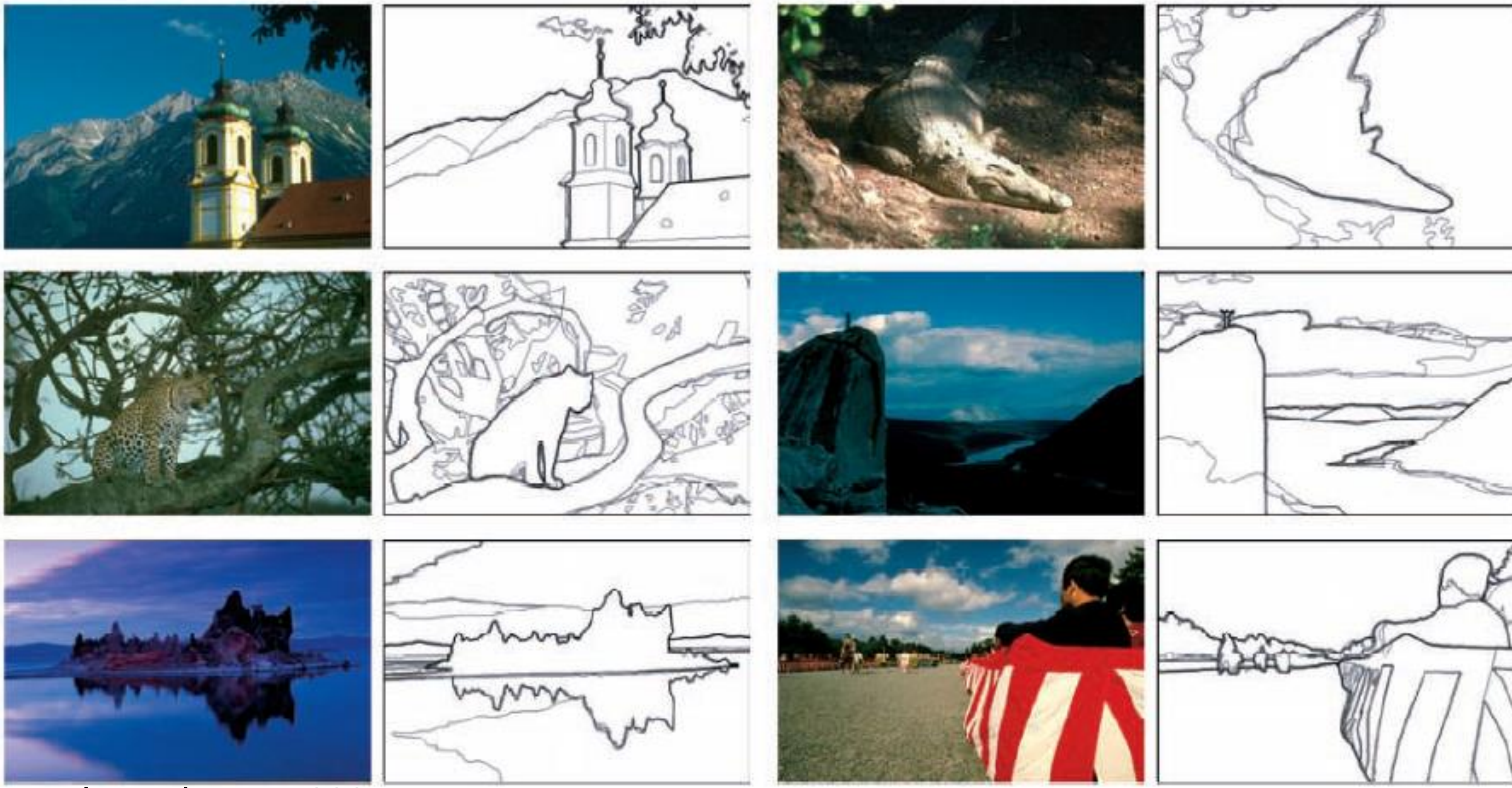
Berkeley Segmentation Database and Benchmark

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





Berkeley Segmentation Database and Benchmark



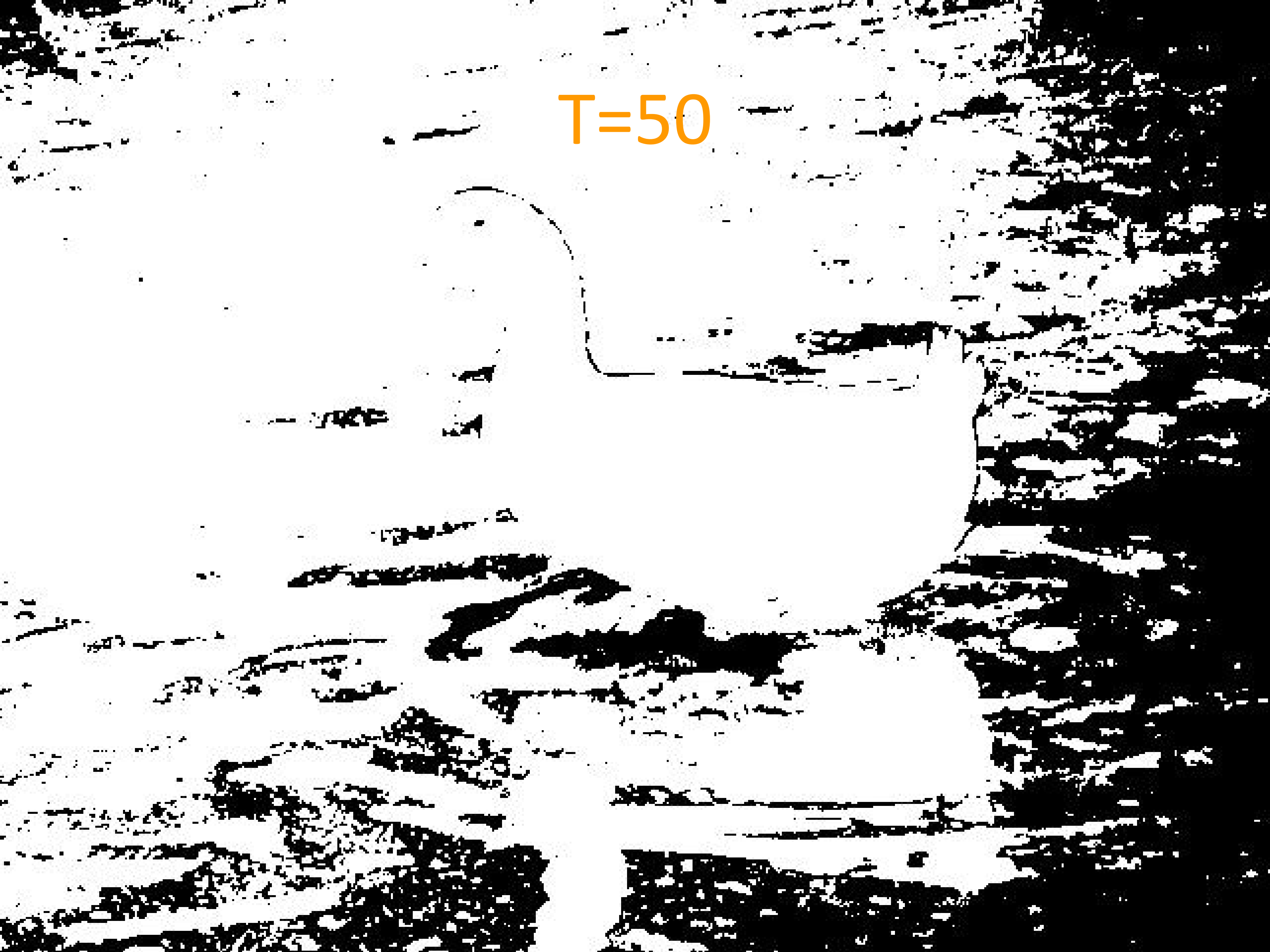
Martin et al. PAMI 2004

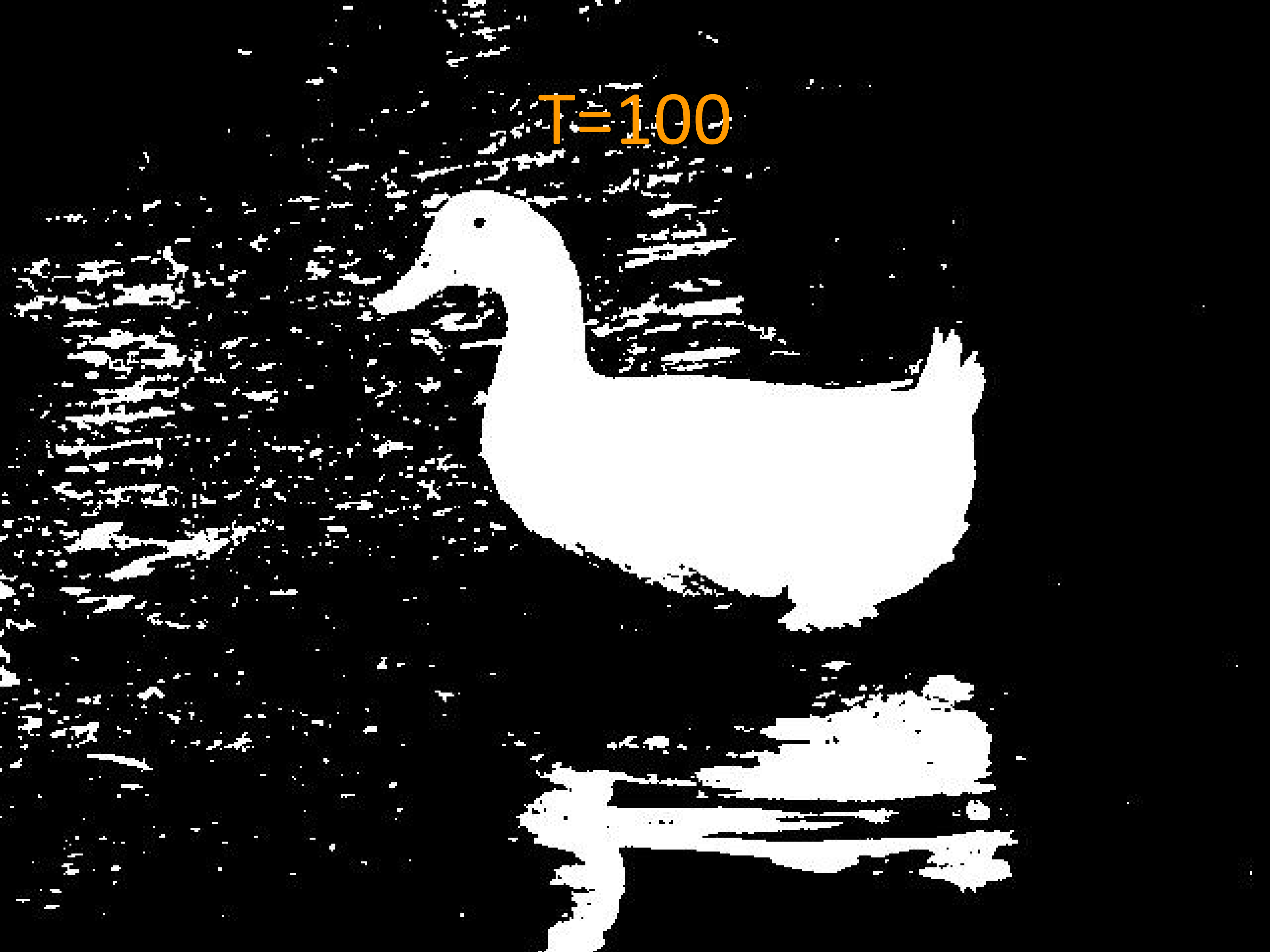
Thresholding

- 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*:

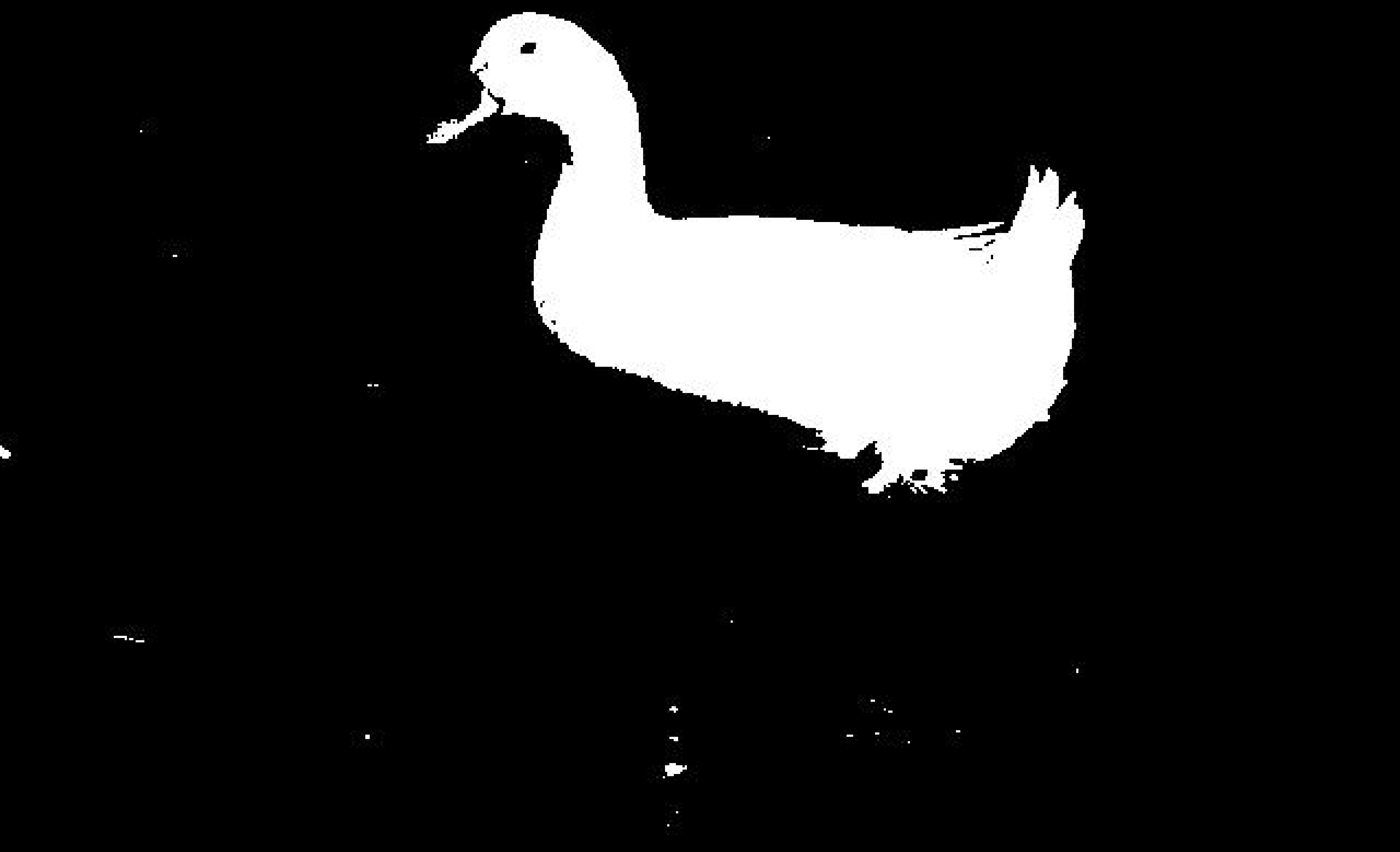
$$
\begin{aligned}
B(x,y) = {} & 1 \qquad \text{if } I(x,y) \geq T \\
& 0 \qquad \text{if } I(x,y) < T.
\end{aligned}
$$

Thresholding example

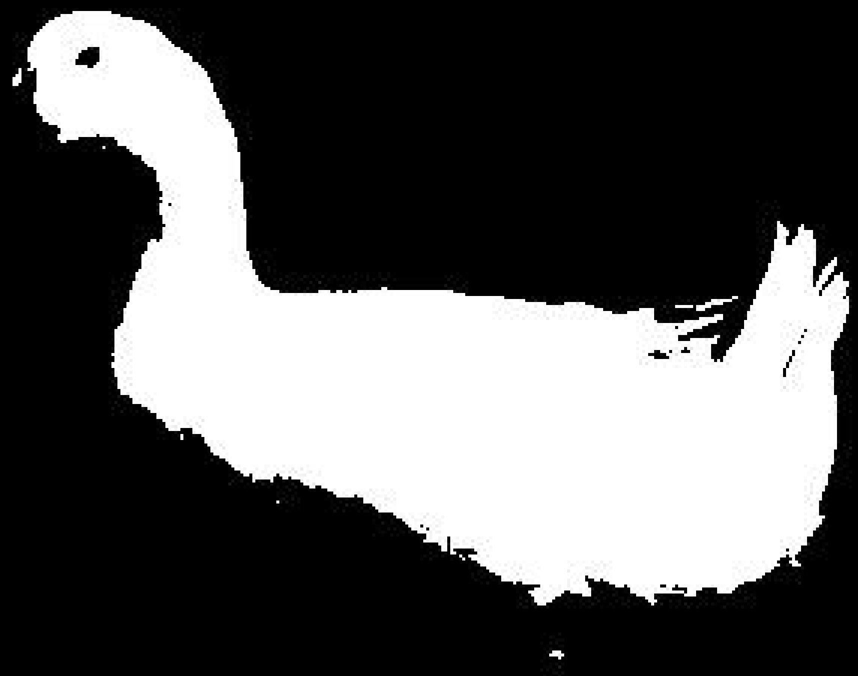

T=50


T=100

T=150



T=200

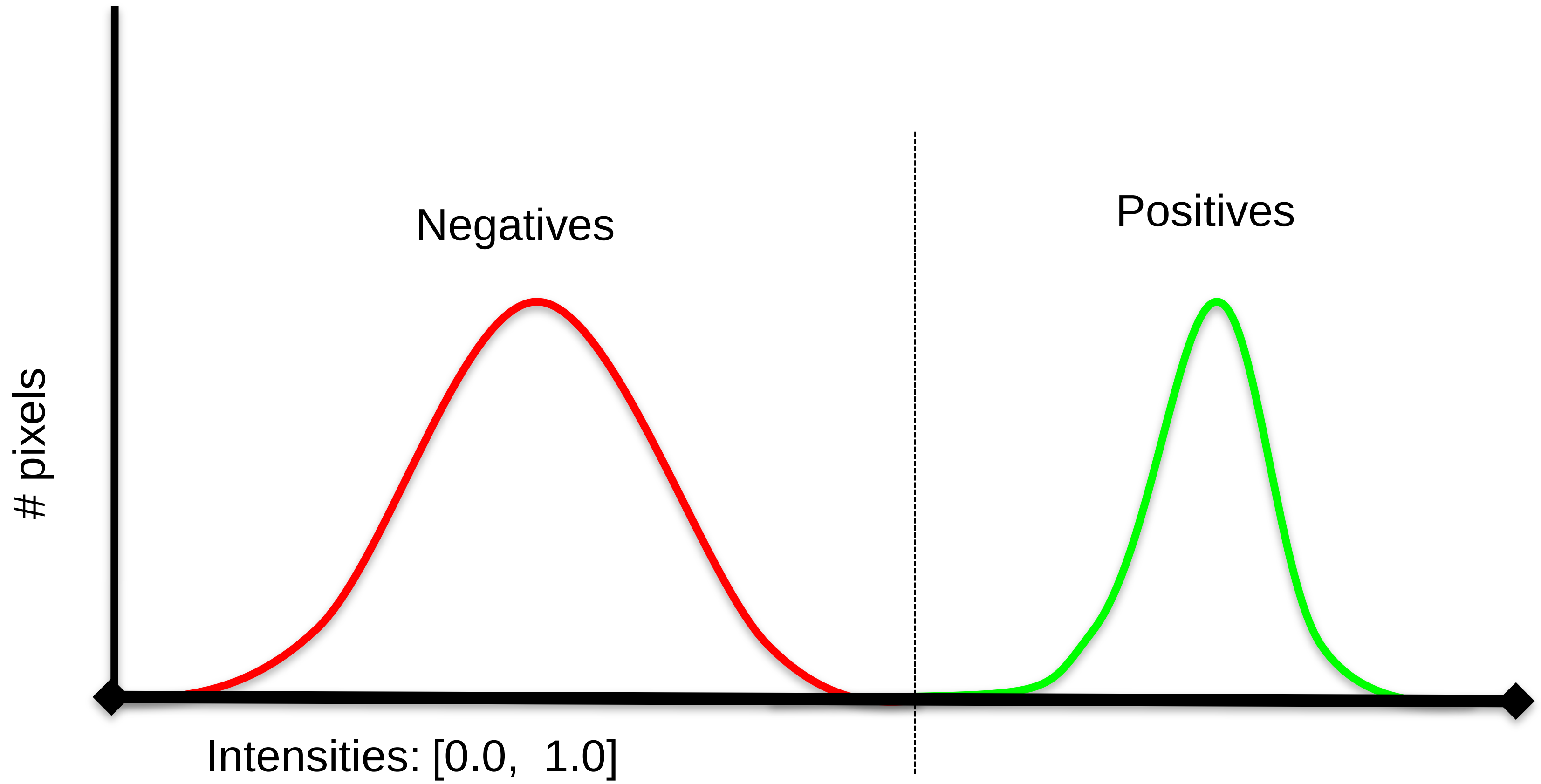


How do we choose T?

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

*= We’ll discuss ROC curves later

Wouldn't it be nice…



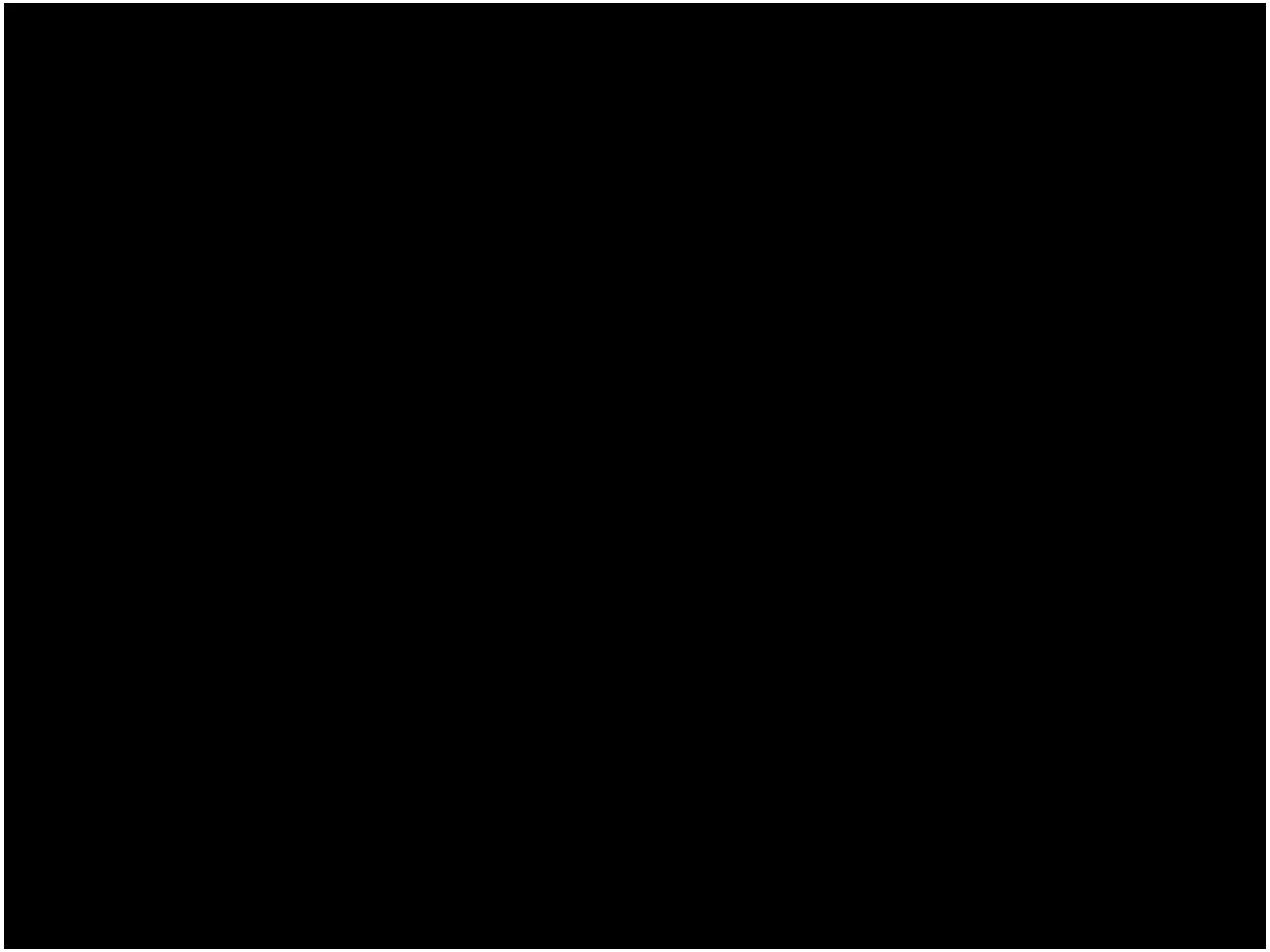
Planning to Segment?

Shraybman, HW1, 2003

Chromakeying: Control Lighting!

generalspecialist.com

Chrome keying example



Chromakeying

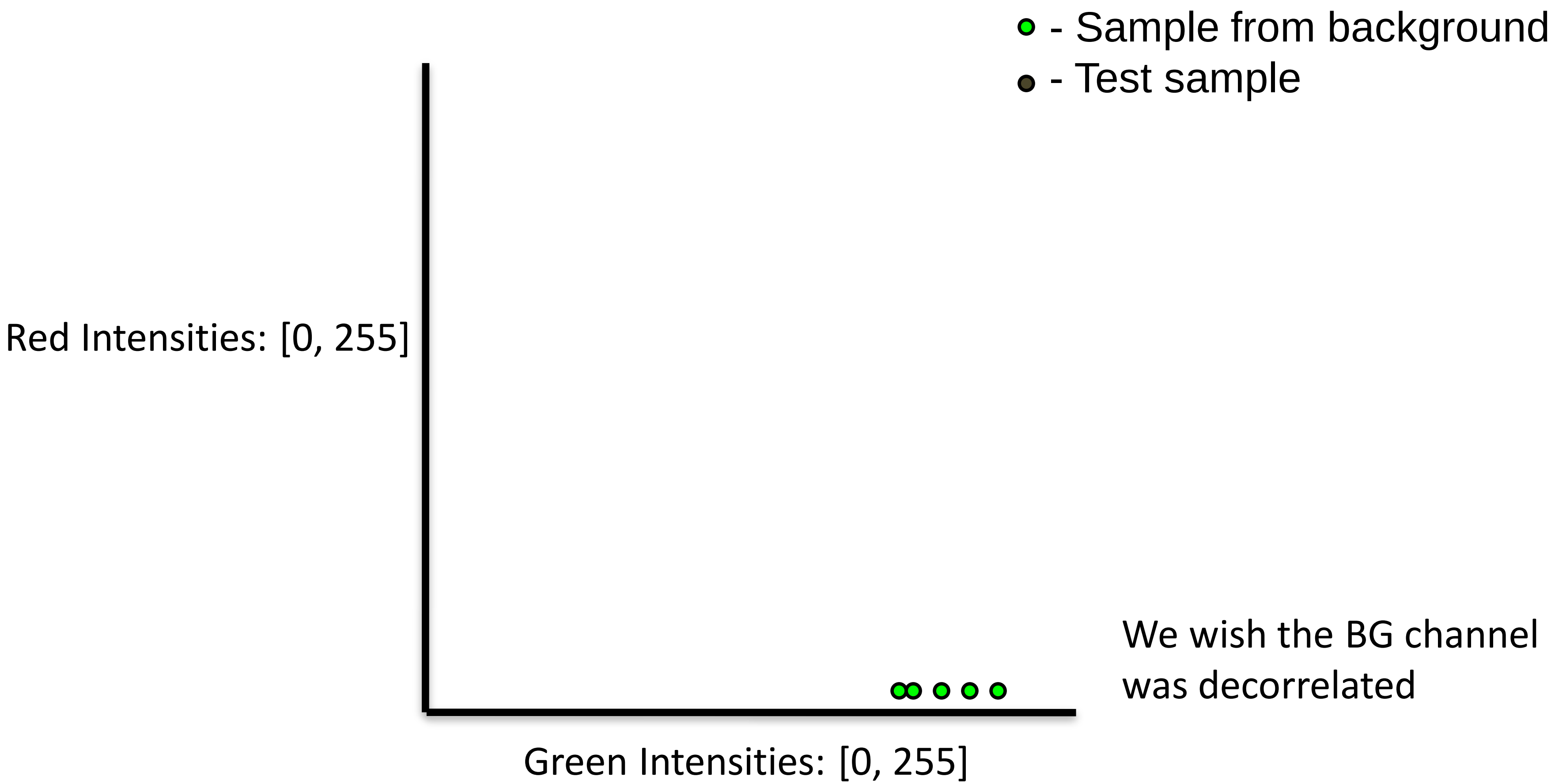
- "Plain" distance measure

$$\mathbf{I}_\alpha = \left|\mathbf{I} - \mathbf{g}\right| > T$$

T = ~20
$\mathbf{g}$ = (0 255 0) (for example)

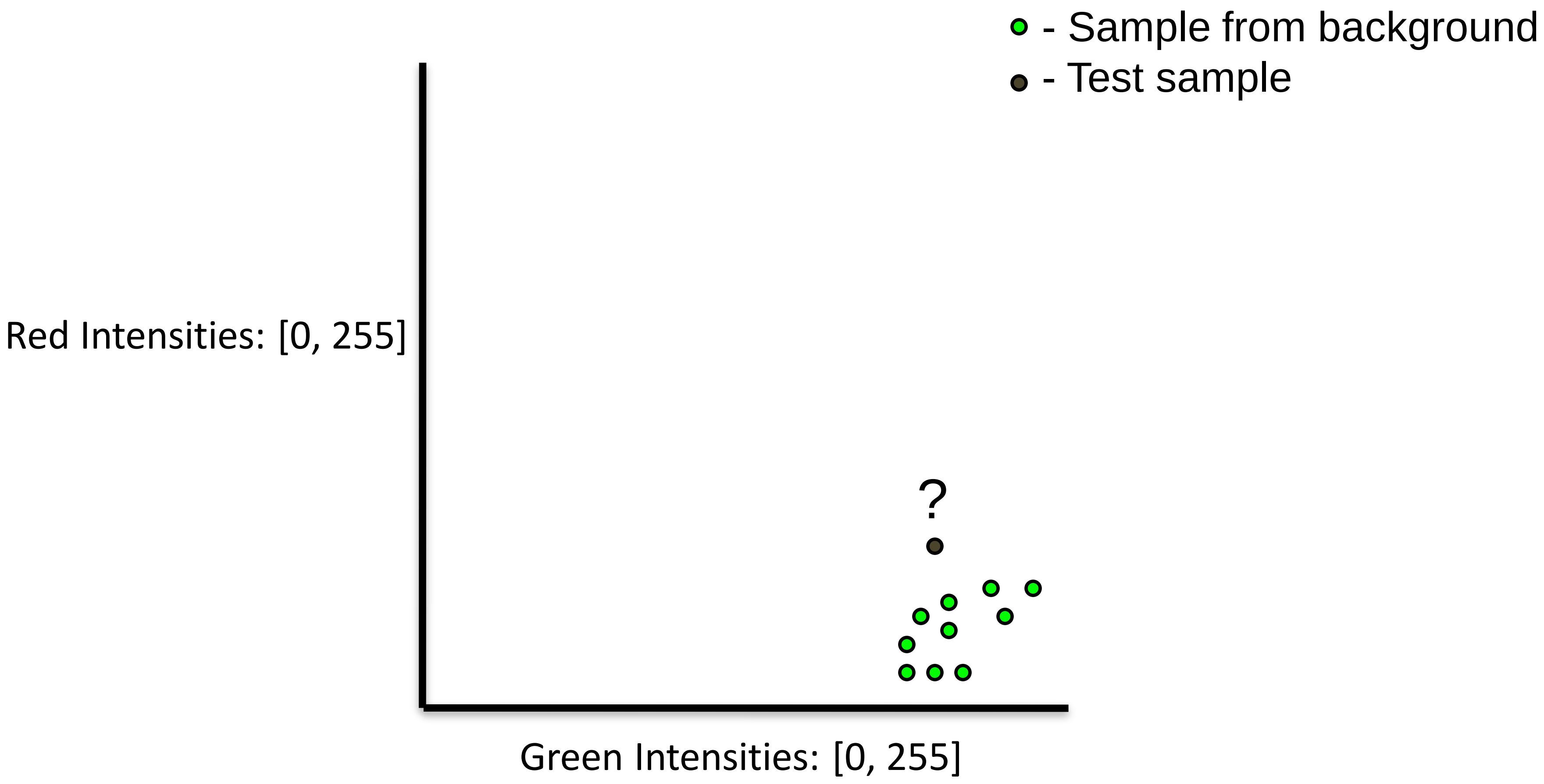
- Problems:
  - Variation is NOT same in all 3 channels
  - Hard alpha mask: $I_{comp} = I_\alpha I_a + (1-I_\alpha) I_b$

Background Color Variation

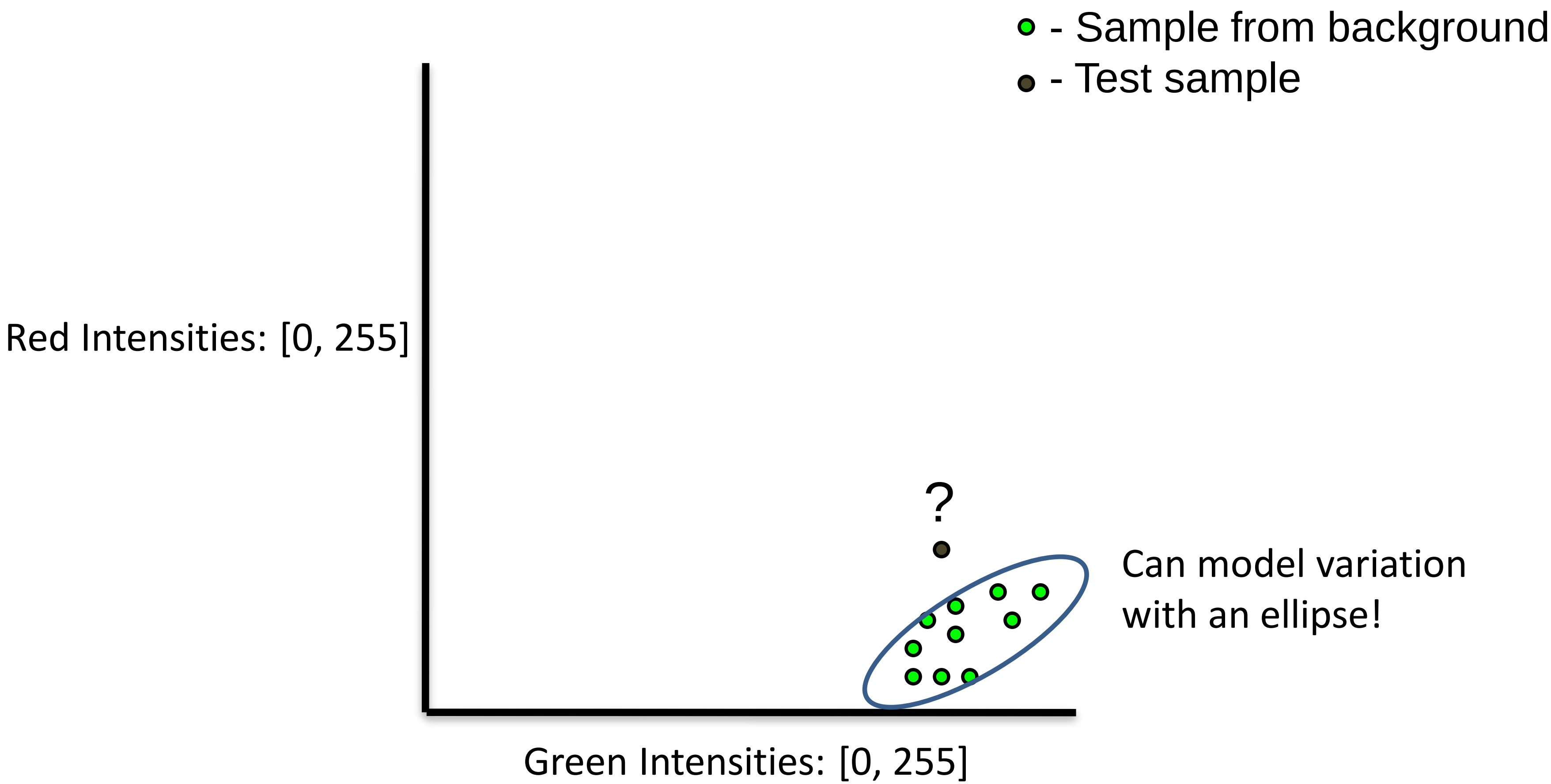


We wish the BG channel was decorrelated

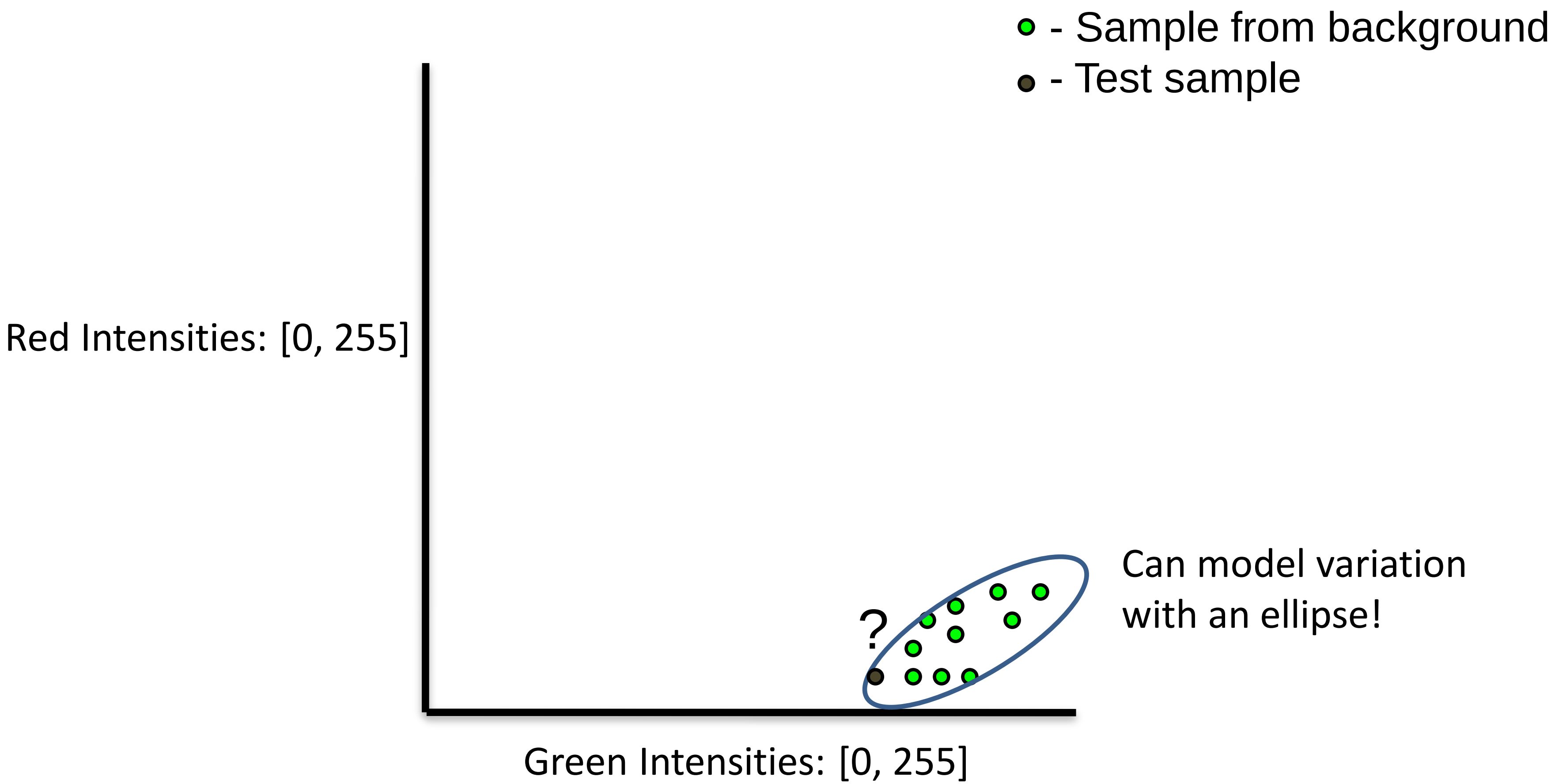
Background Color Variation



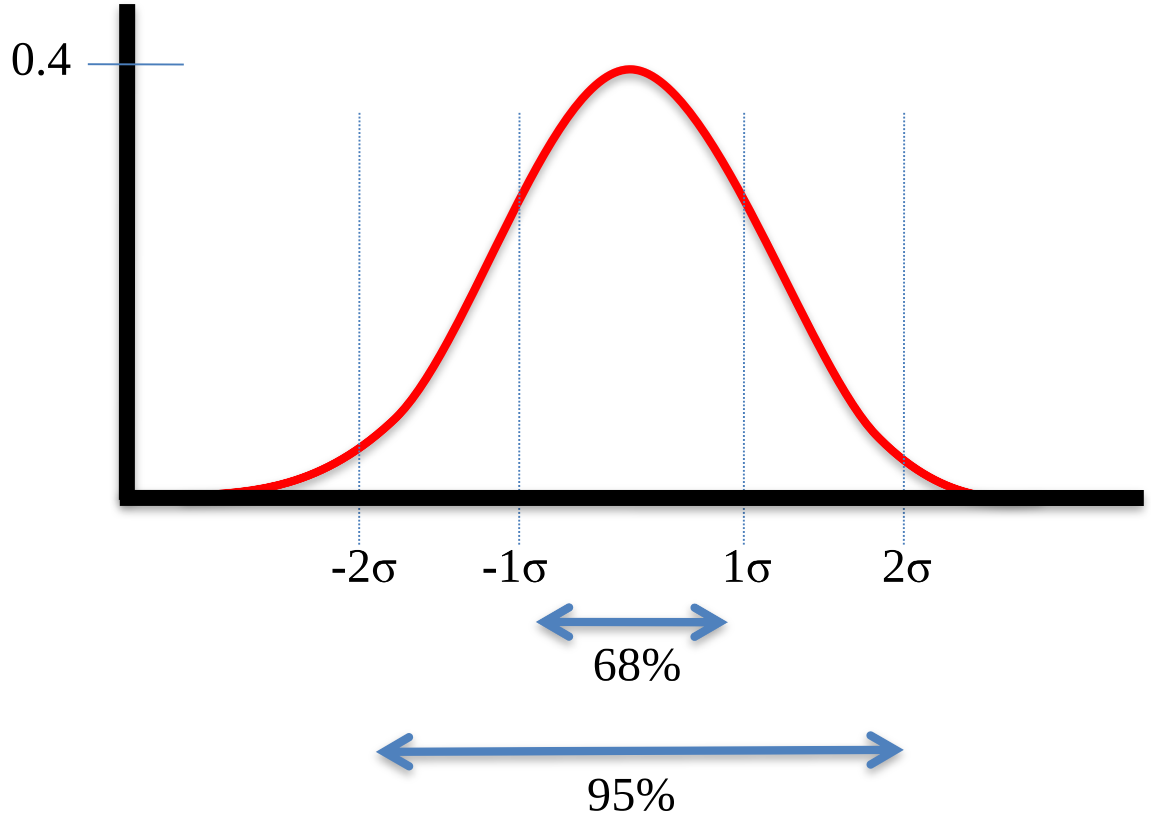
Background Color Variation



Background Color Variation



Use Gaussian to Explain Most Data

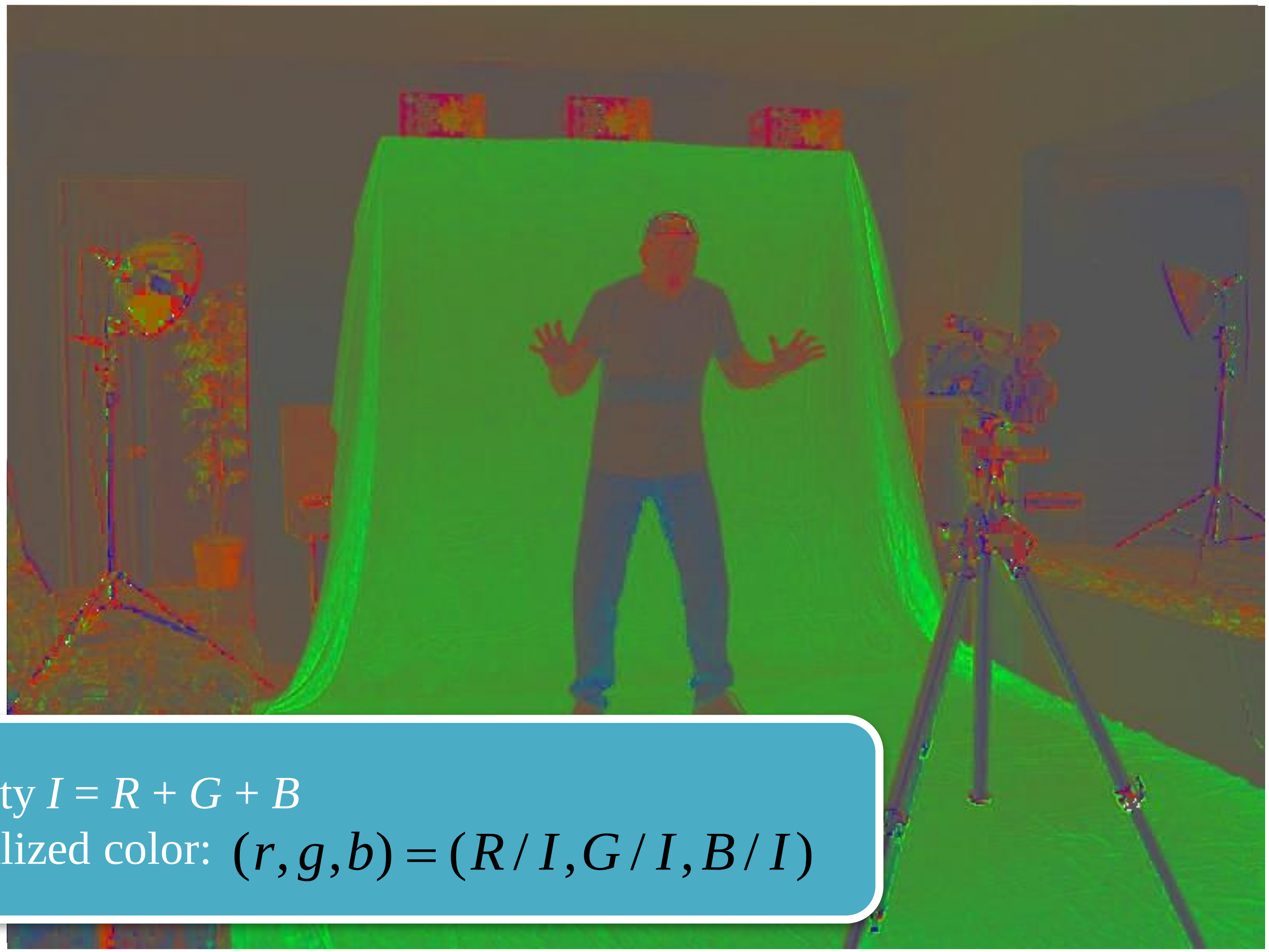


Green Variation in Reality

Making of "42 Story House"

3D RGB plot of just BG

Green Variation in Reality



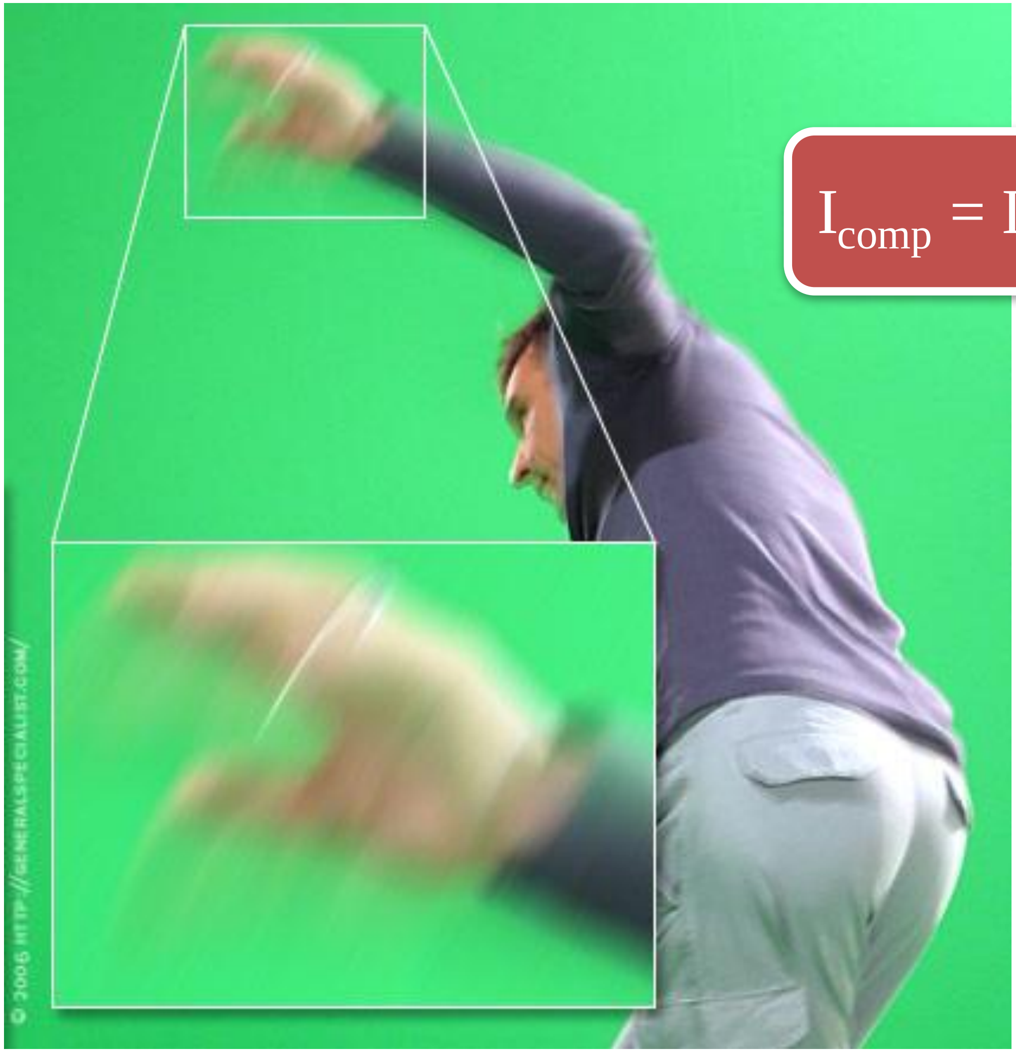
3D RGB plot of geometry-normalized BG

Intensity $I = R + G + B$
Normalized color: $(r, g, b) = (R/I, G/I, B/I)$

Making of “42 Story House”

ETH

Mixed Pixels:
Why I_α Should be Grayscale



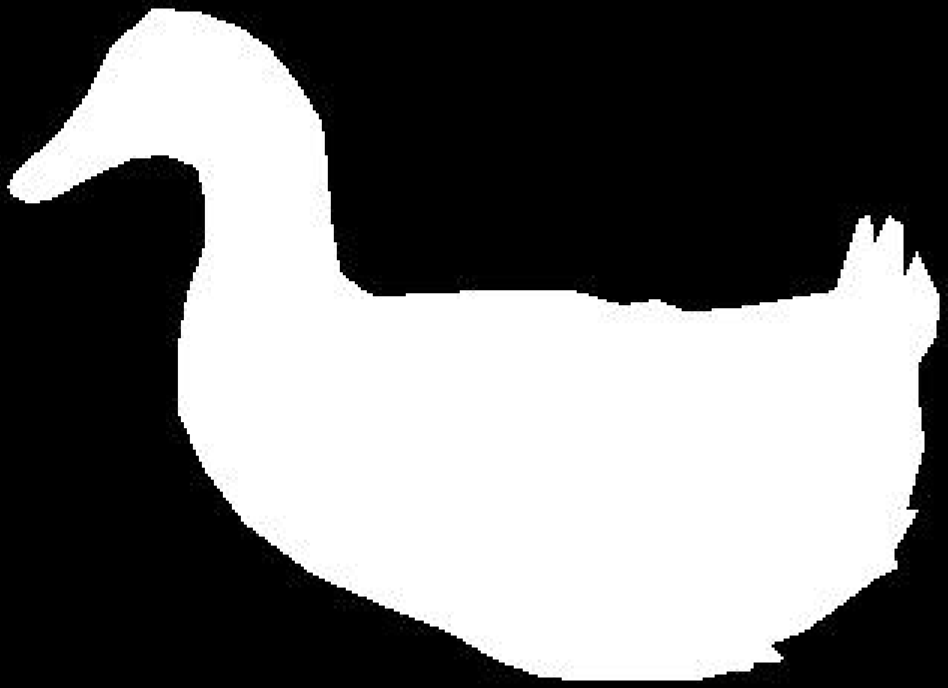
$$I_{comp} = I_\alpha \, I_a + (1-I_\alpha) \, I_b$$

generalspecialist.com

Segmentation Performance

- To use automatic analysis, one needs 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.

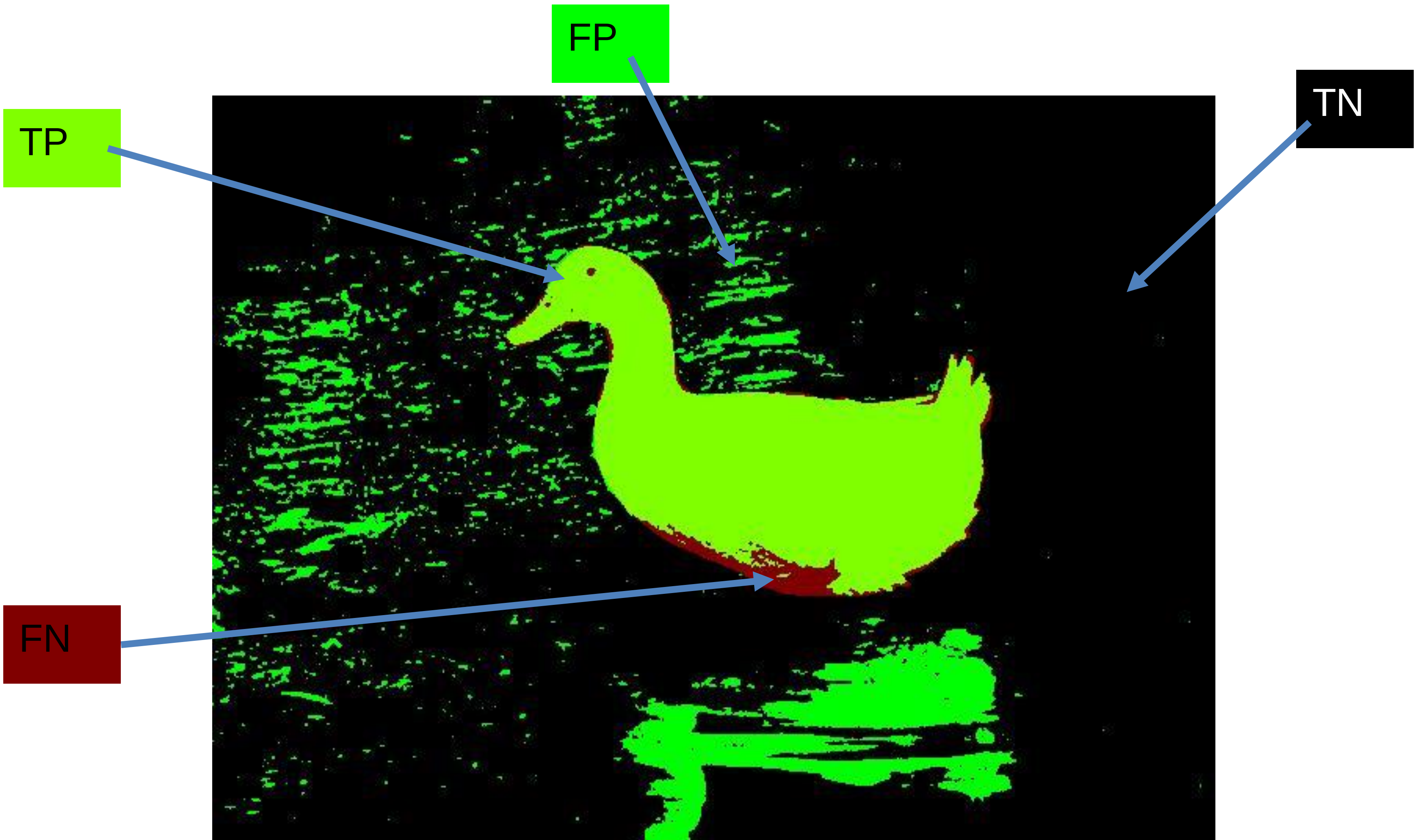
- A binary classifier distinguishes between two different types of thing. E.g.:
  - Healthy/afflicted patients – cancer screening
  - Pregnancy tests
  - Object detection
  - Foreground/background image pixels

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	

Classification outcomes



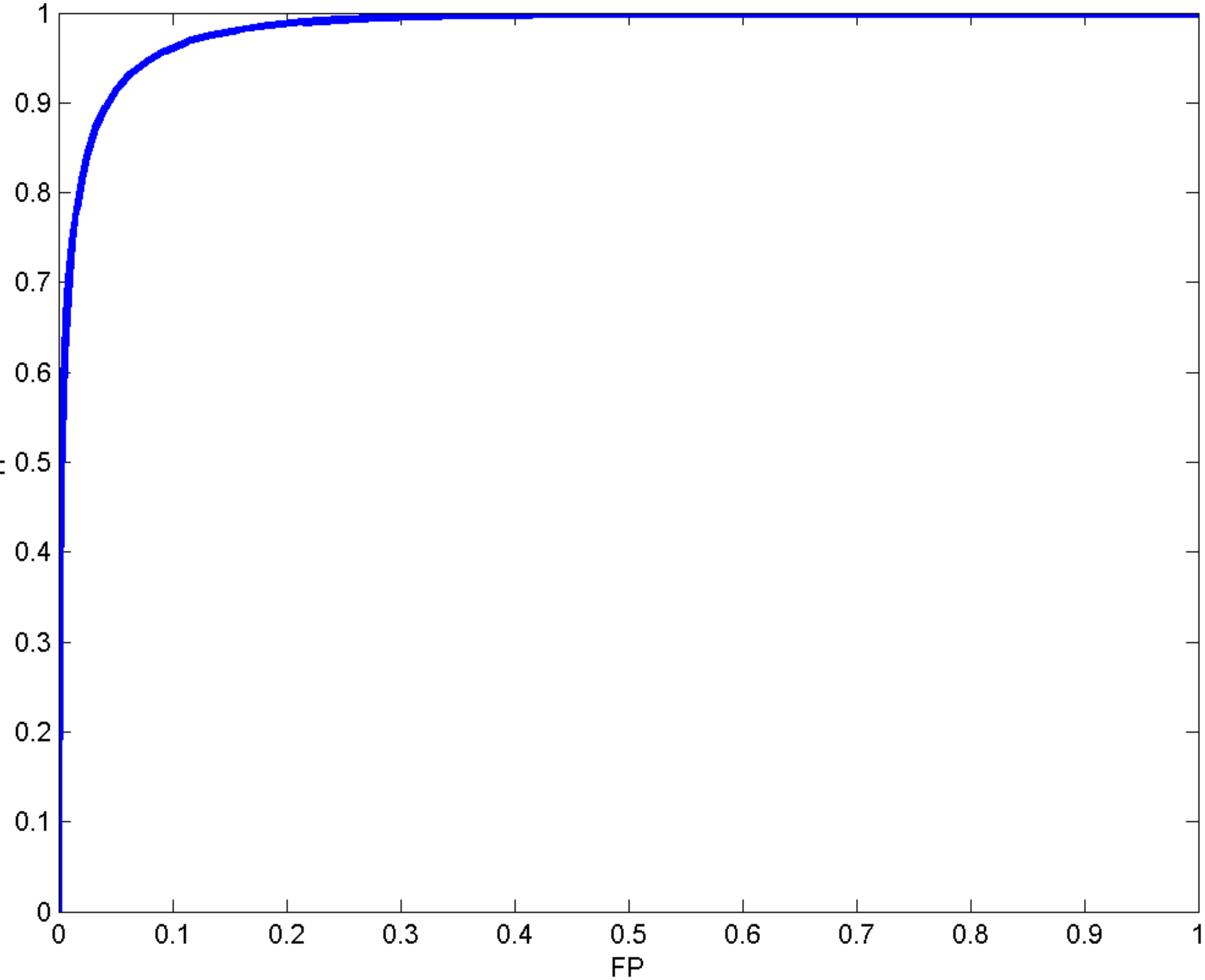
The ROC Curve

- 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)?

Threshold Sweep?

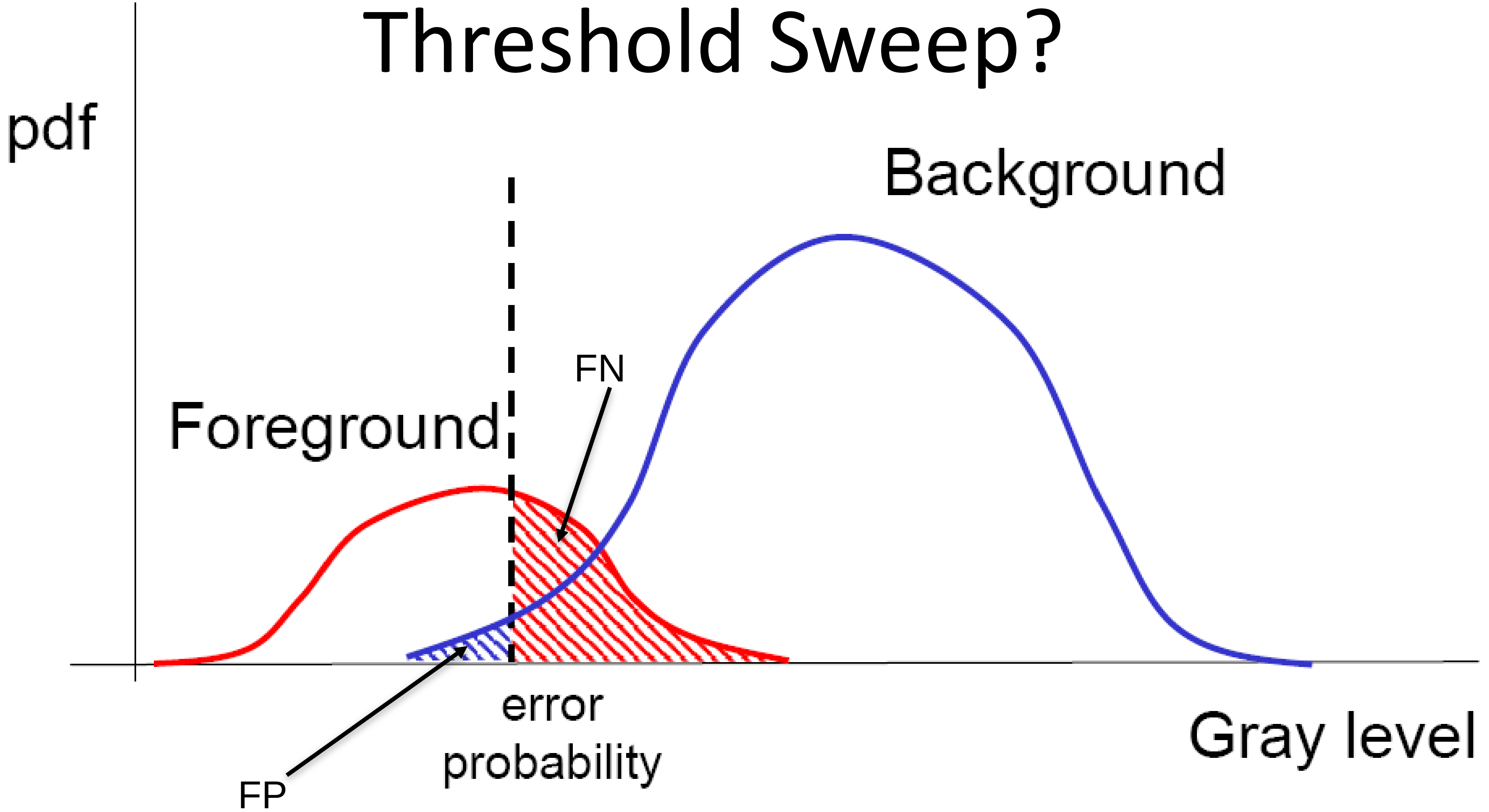
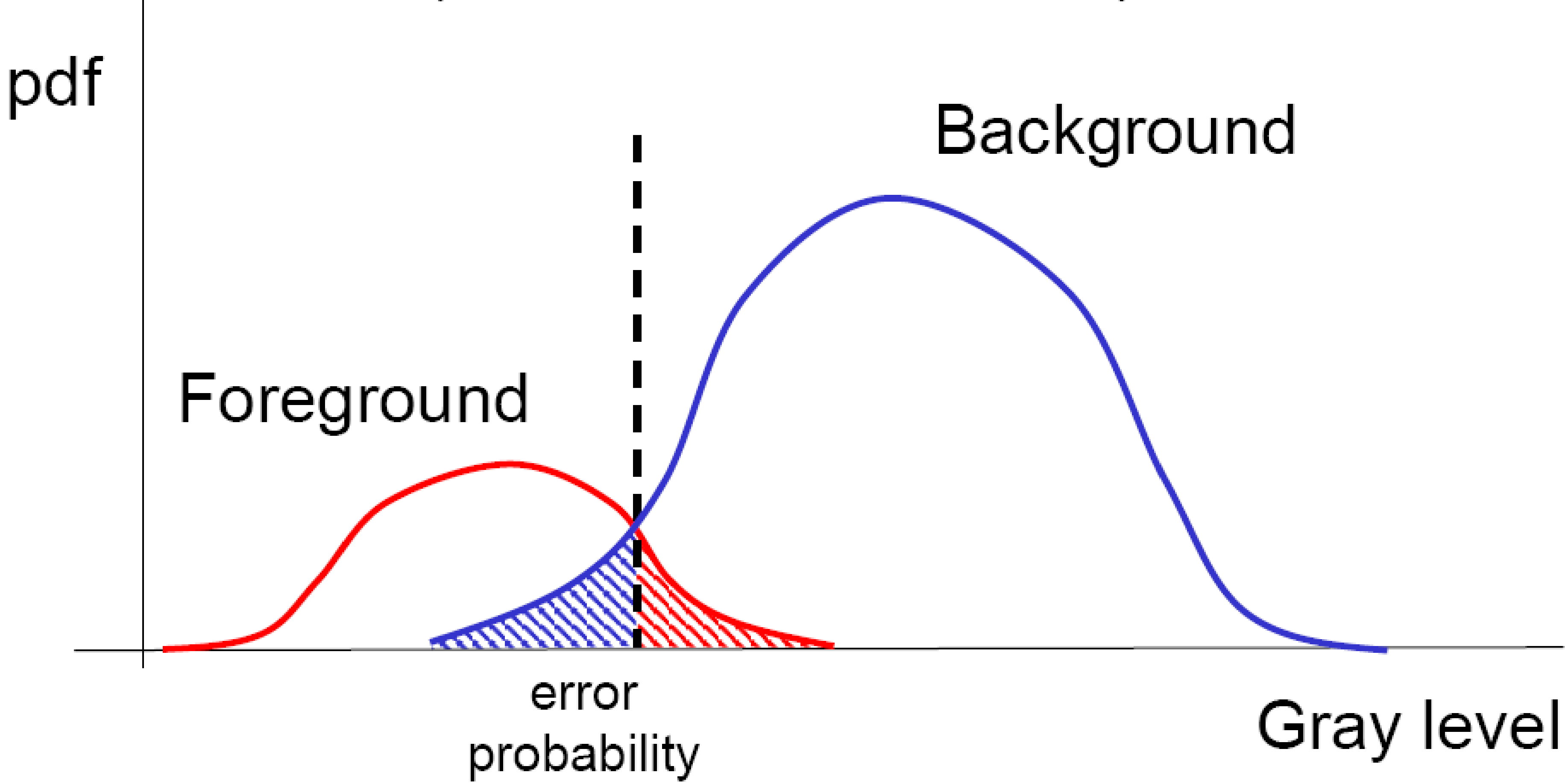


Figure: Bernd Girod

“MAP (Maximum A Posteriori) detector”

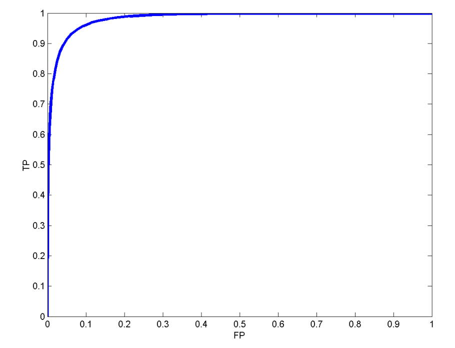


If different outcomes are associated with different costs:
more general “Bayes minimimum risk detector”

Figure: Bernd Girod

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

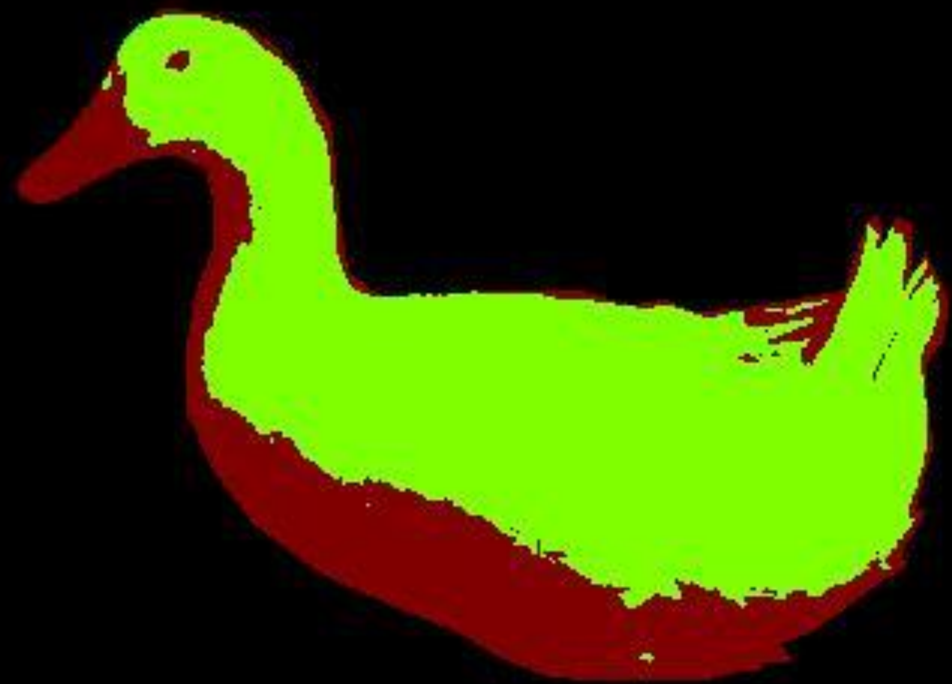


- Choose the point on the ROC curve with **gradient**

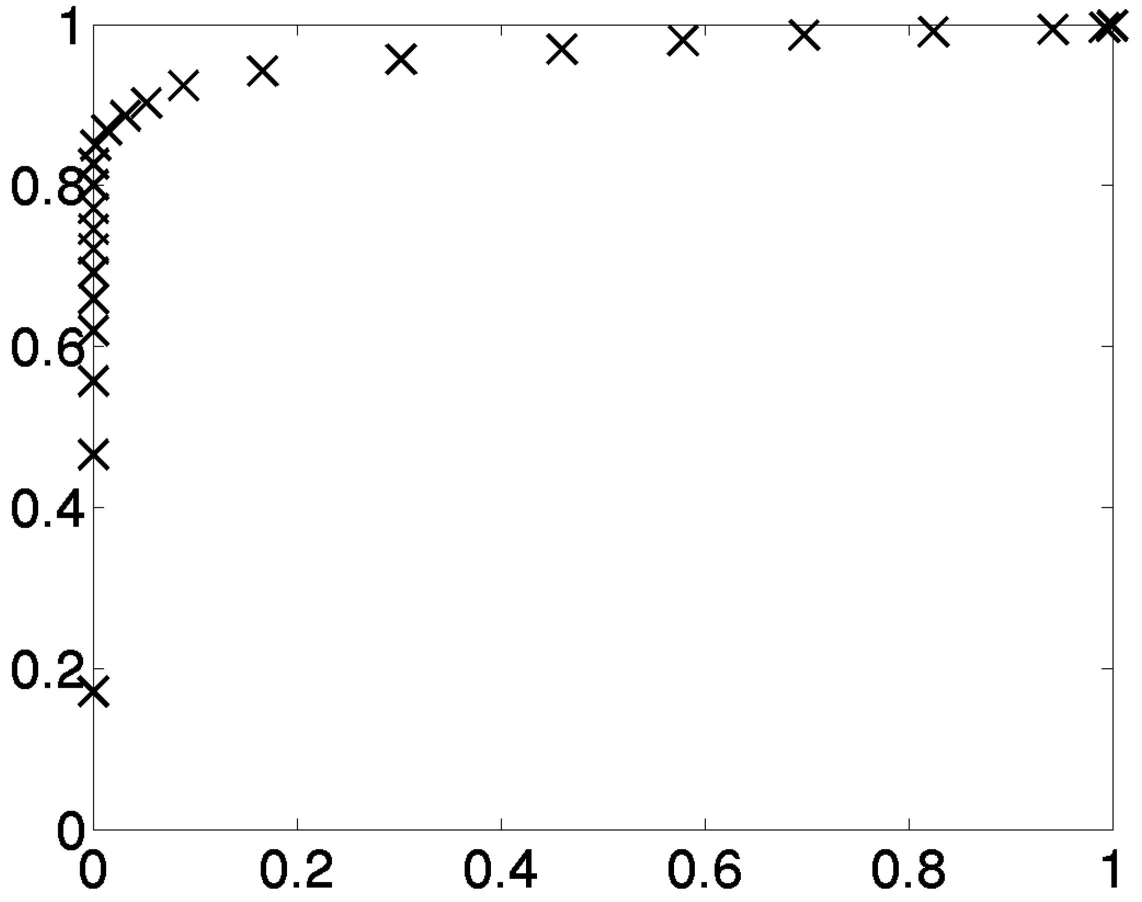
$$\beta = \frac{N}{P}\frac{V_{TN} + C_{FP}}{V_{TP} + C_{FN}}$$

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

Classification outcomes



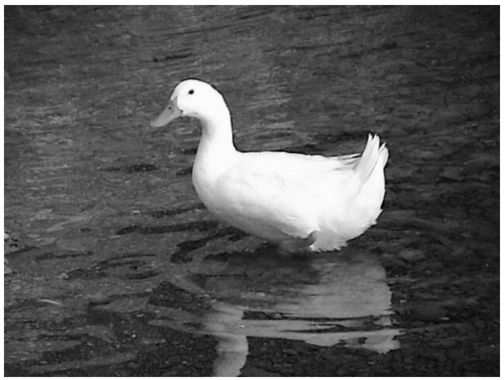
ROC curve

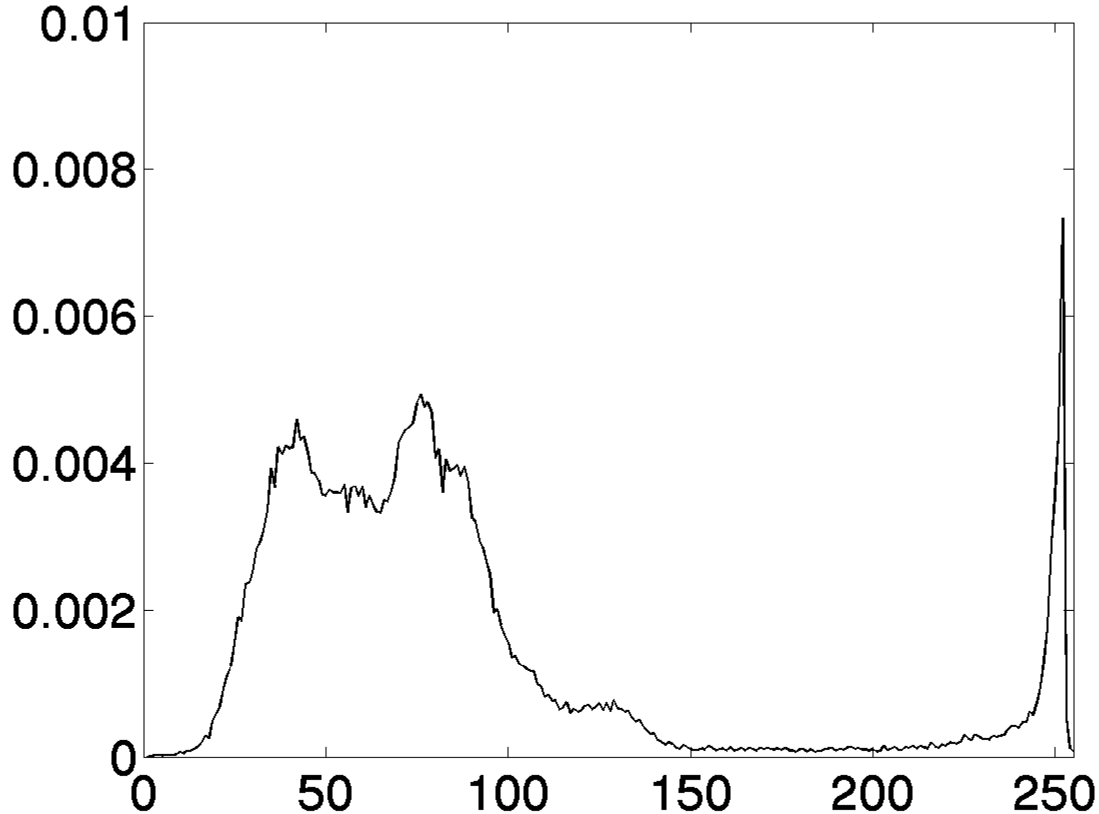


TP fraction

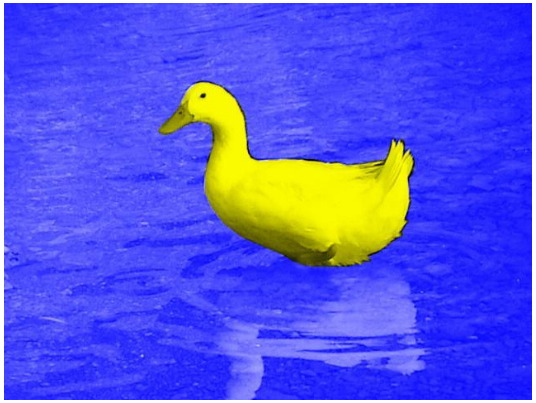
FP fraction

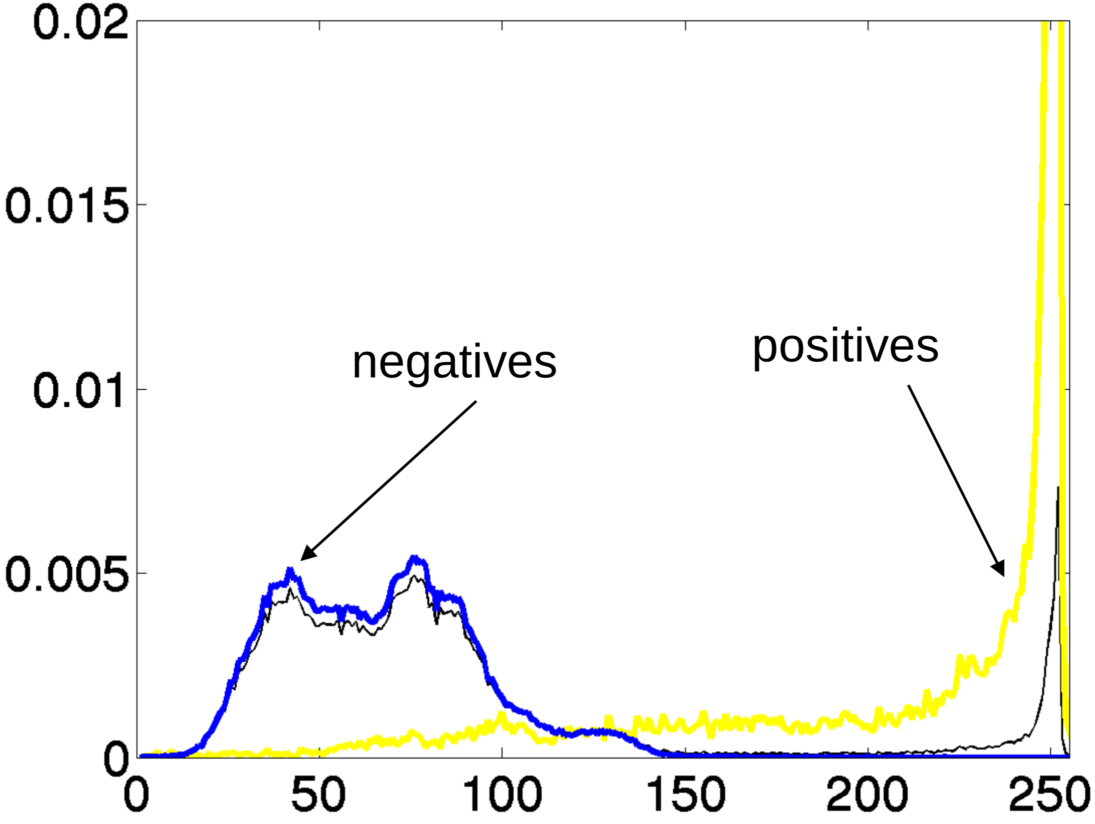
Greylevel Histograms





Positives and Negatives



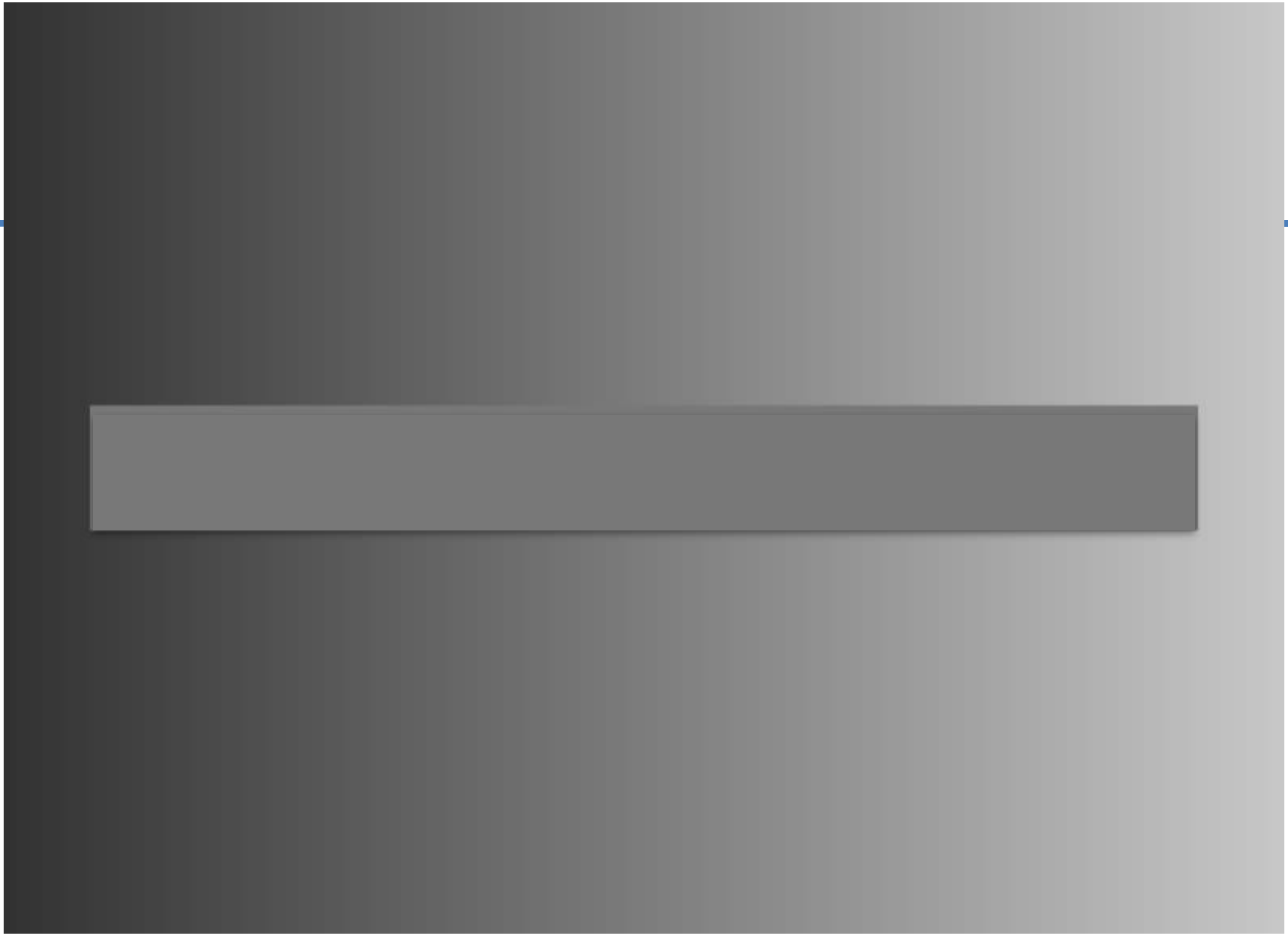


->22/09

Limitations of Thresholding

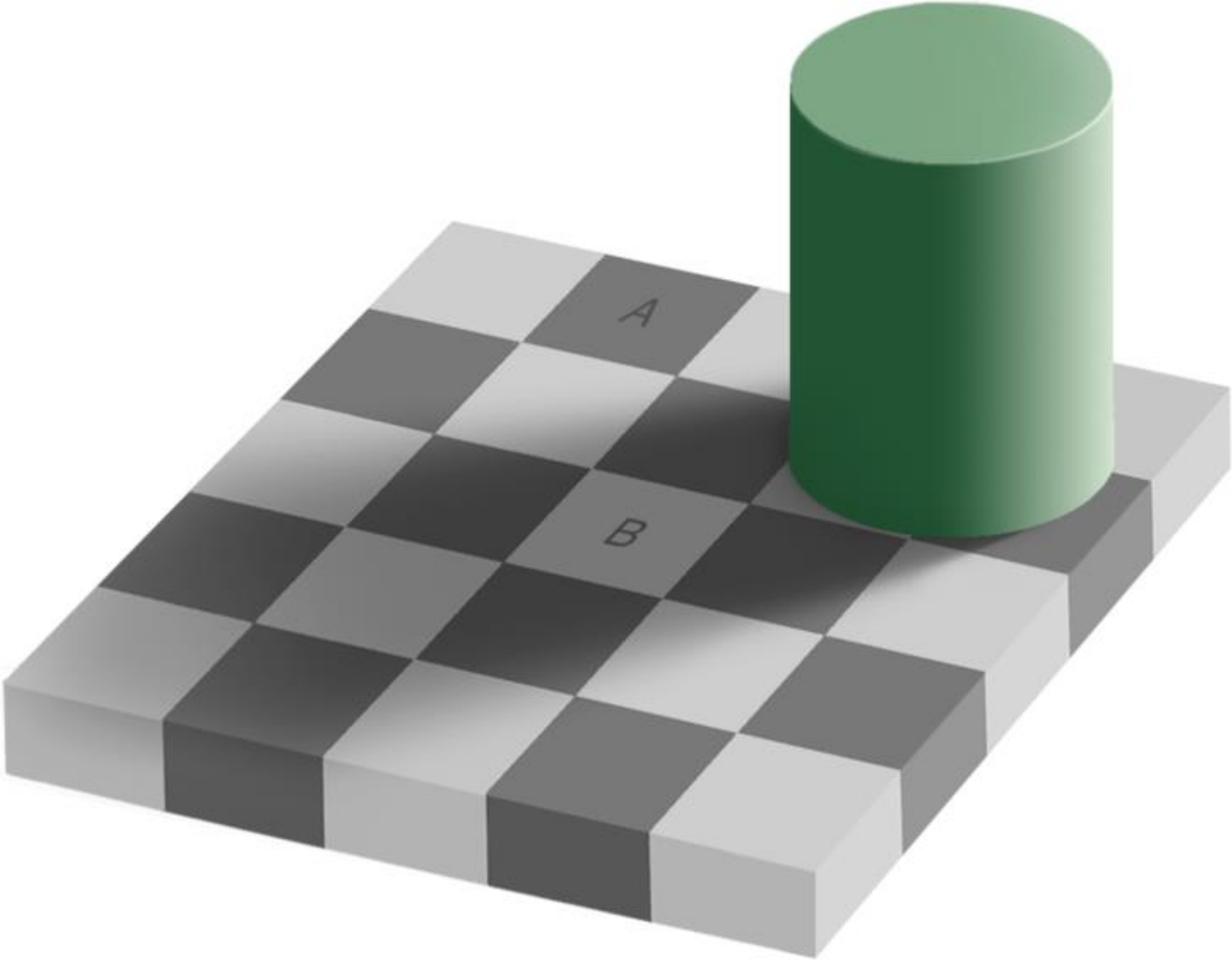
- Why can we segment images much better by eye than through thresholding processes?

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

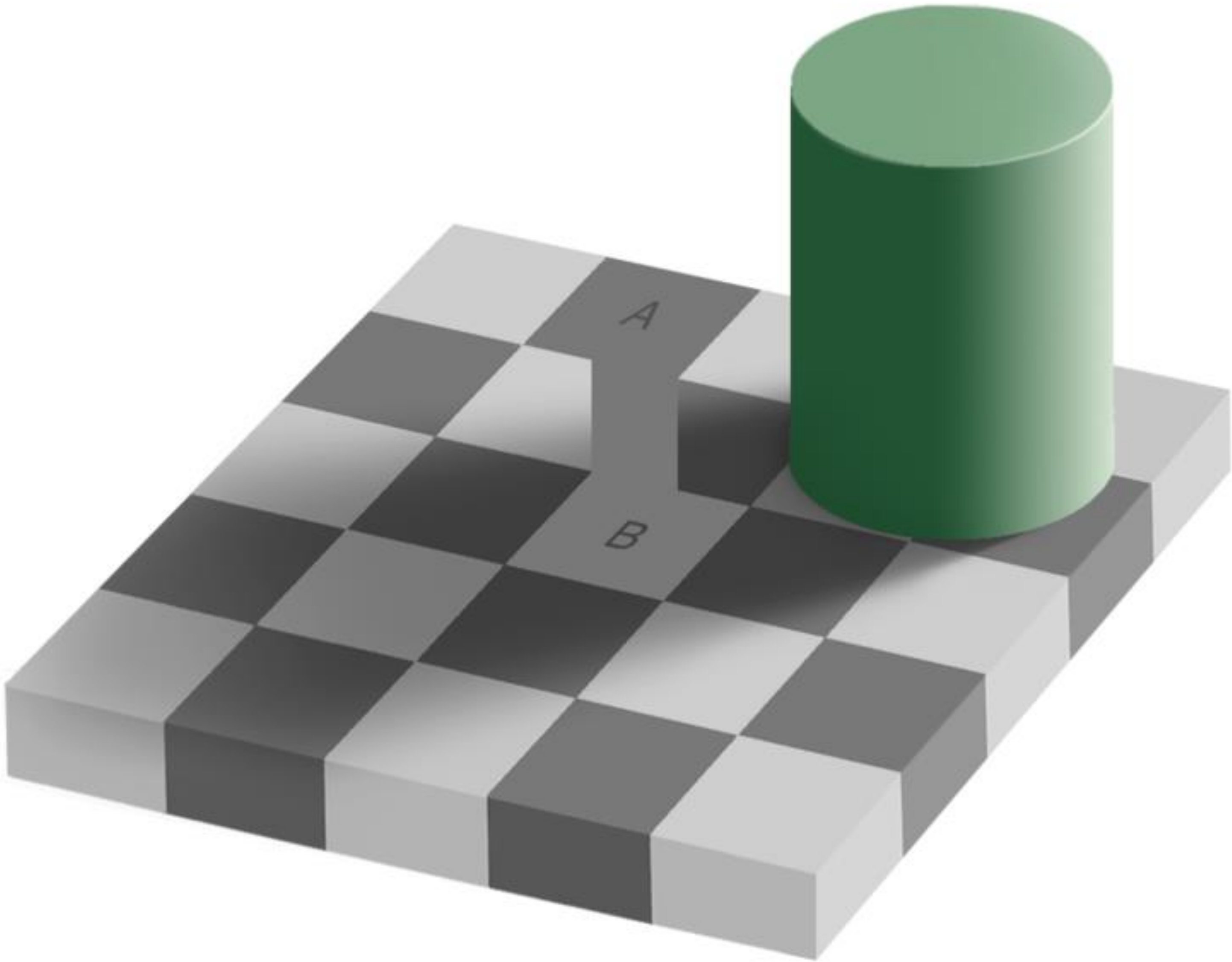


[Gradient.illusion.arp.jpg]

Aha! Humans are suckers for context!

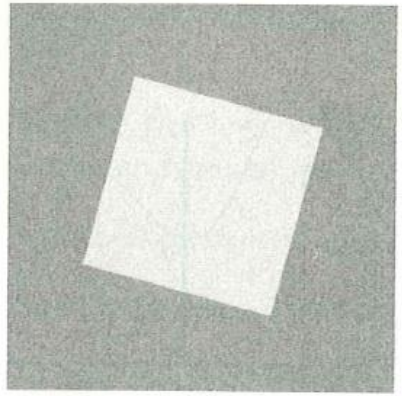


by Adrian Pingstone, based on the original created by Edward H. Adelson

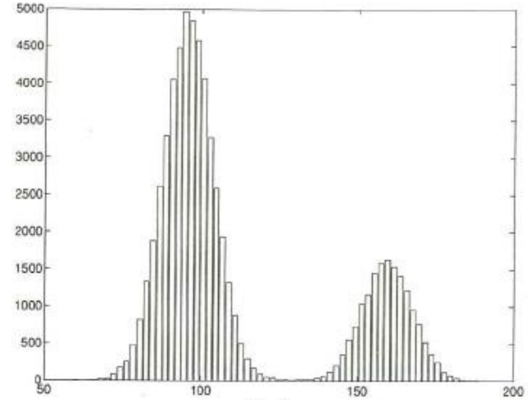


by Adrian Pingstone, based on the original created by Edward H. Adelson

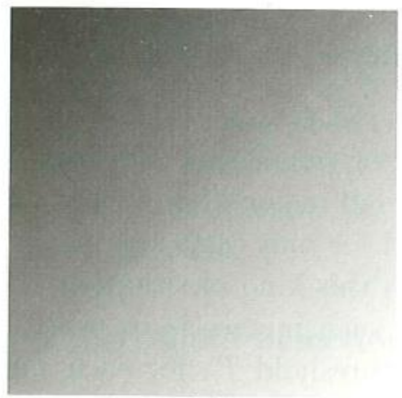
Chapter 3 in Machine Vision by Jain et al.



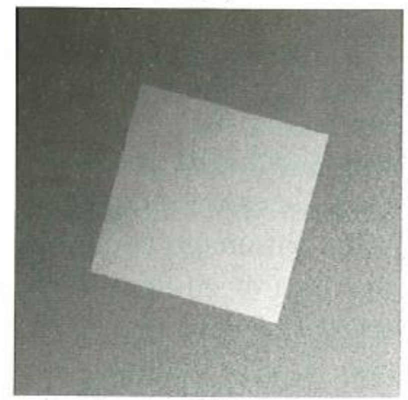
(a)



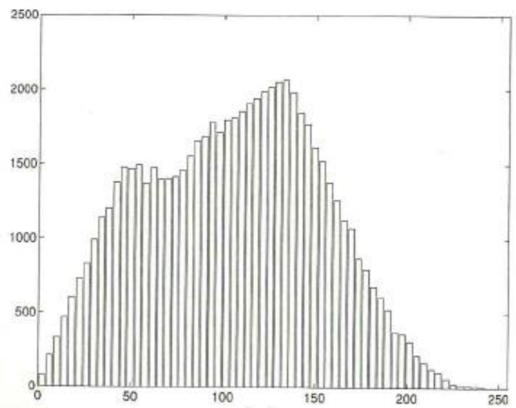
(b)



(c)

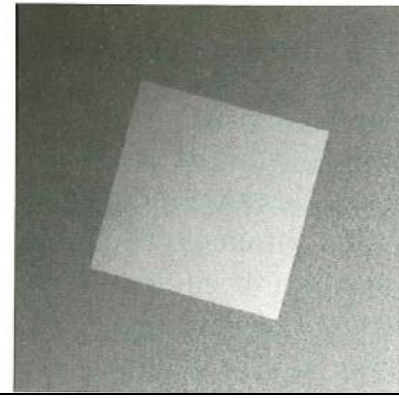


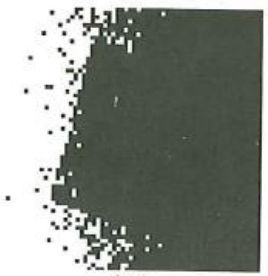
(d)



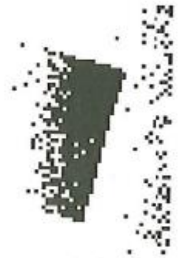
(e)

(f)

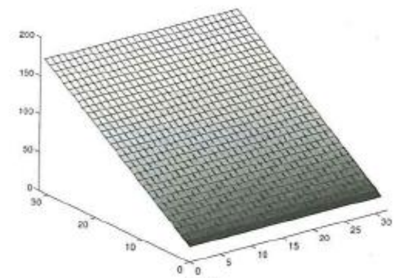




(d)

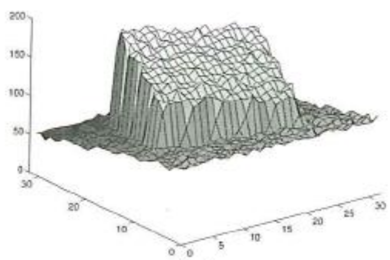


(e)



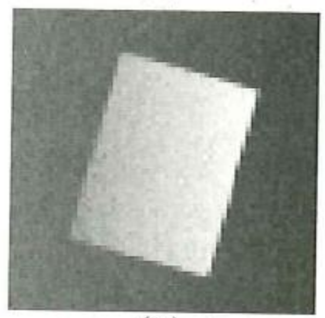
200
150
100
50
0
30
20
10
0 0
5
10
15
20
25
30

(f)

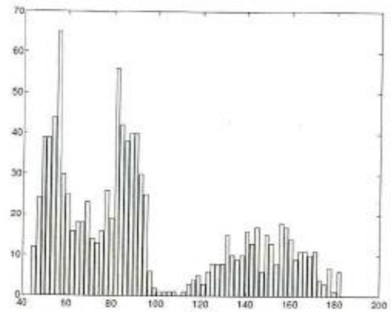


200
150
100
50
0
30
20
10
0 0
5
10
15
20
25
30

(g)



(h)



70
60
50
40
30
20
10
0
40
60
80
100
120
140
160
180
200

(i)

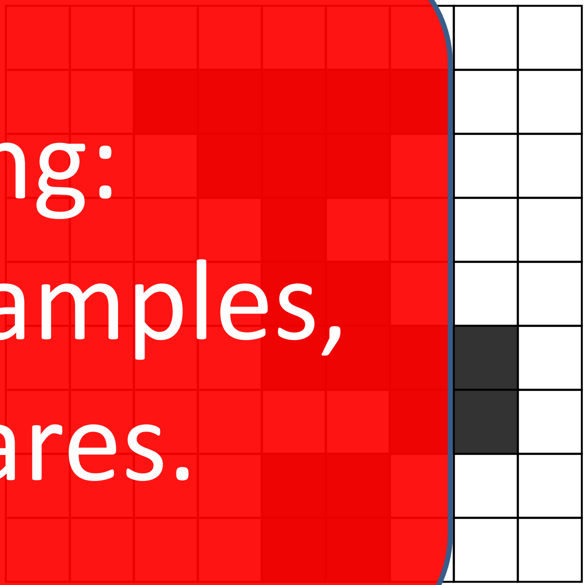
(j)

Note on Performance Assessment

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

Pixel connectivity

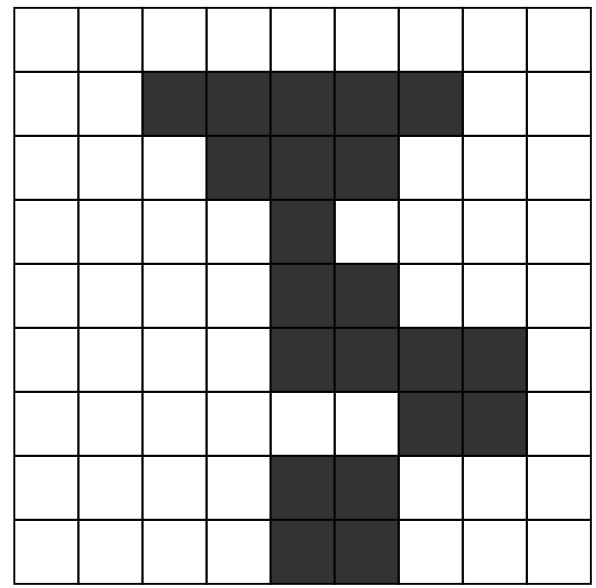
- We need to define which pixels are neighbors.
- Are the dark pixels in this array connected?



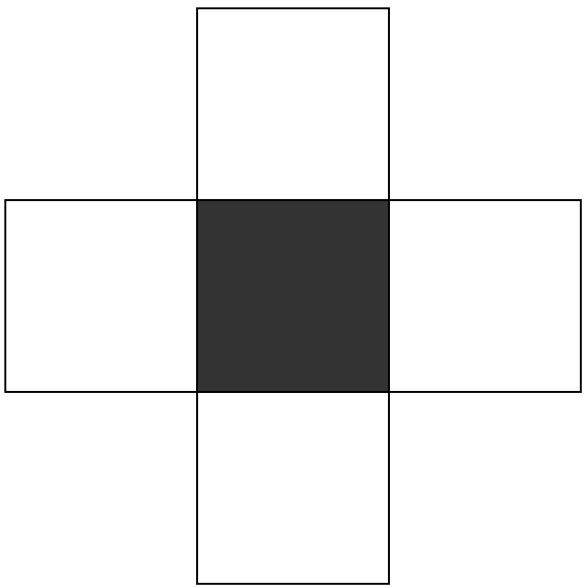
Warning:
Pixels are samples, not squares.

Pixel connectivity

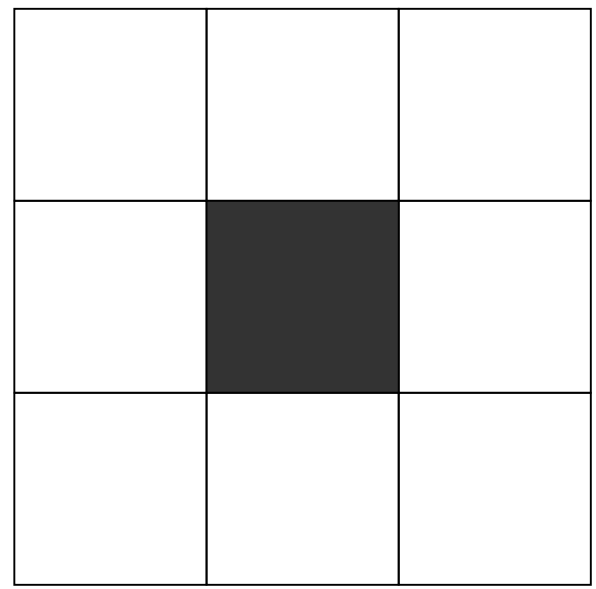
- We need to define which pixels are neighbors.
- Are the dark pixels in this array connected?



Pixel Neighborhoods



4-neighborhood



8-neighborhood

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-neighbor of p_{i+1}, i=1,…,n-1.

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

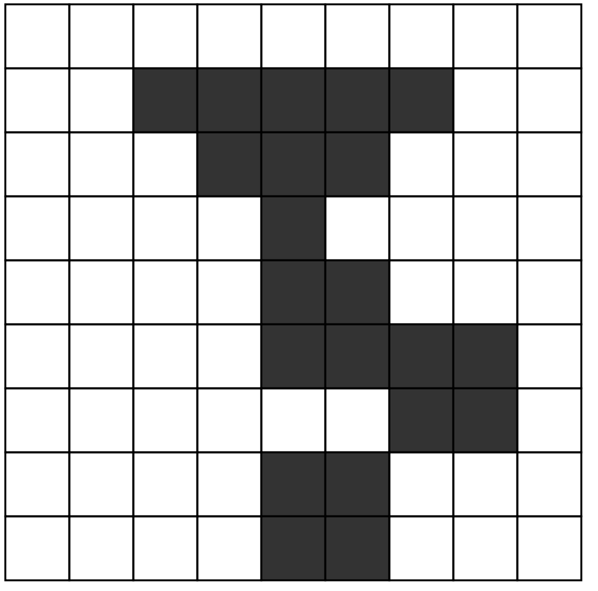
Connected regions

- 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.

ETH

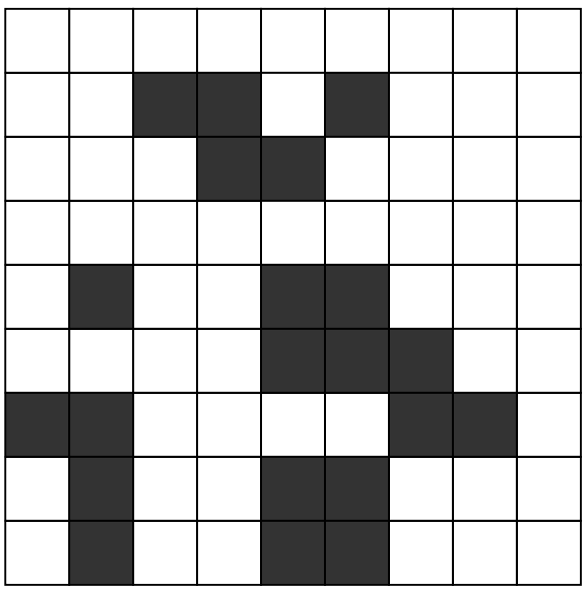
Connected regions

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



Connected components labelling

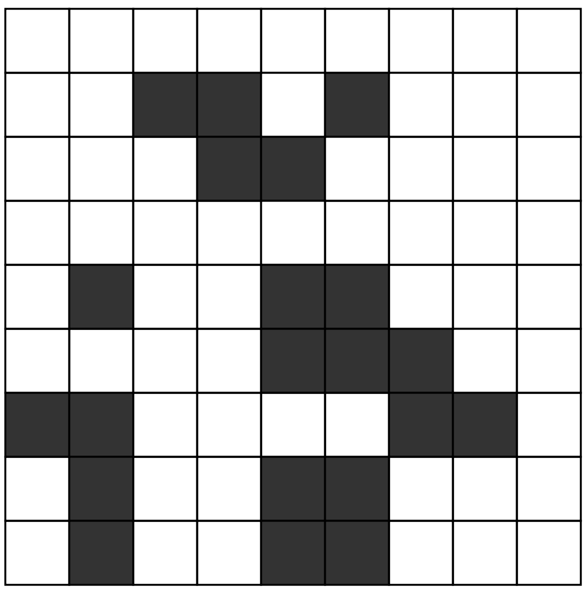
- Labels each connected component of a binary image with a separate number.



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

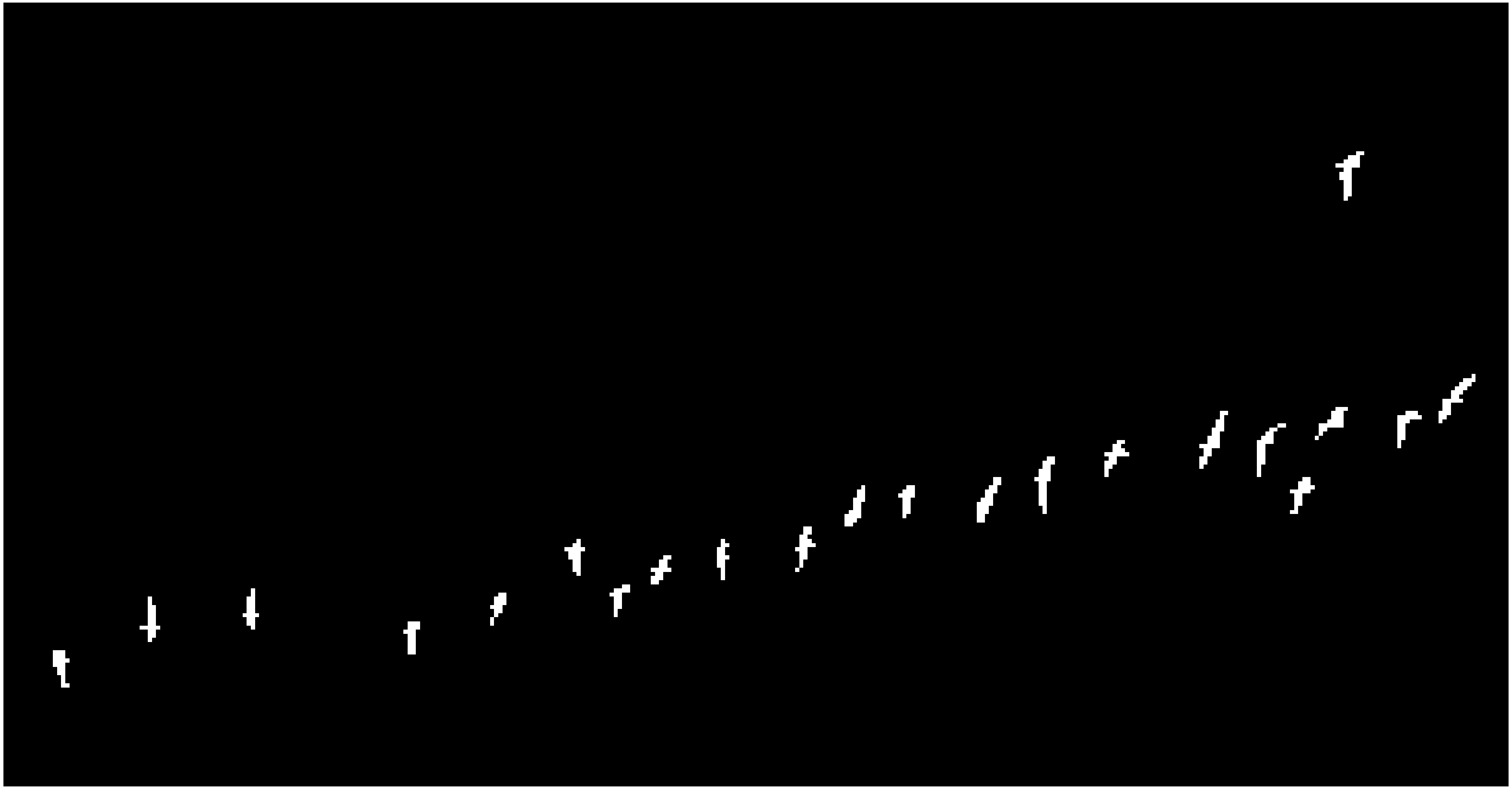
- Only extract the connected components of the foreground



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

Goose detector

Goose detector



Region Growing

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

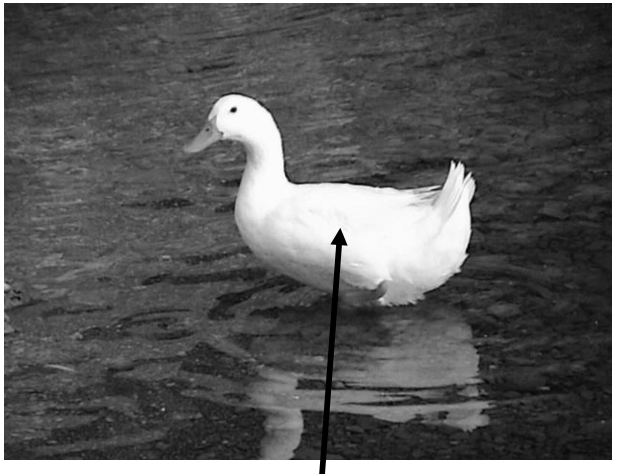
Region Growing

```
def regionGrow(I, seed):
    X, Y = I.shape
    visited = np.zeros((X,Y))
    visited[seed] = 1
    boundary = []
    boundary.append(seed)
    while len(boundary) > 0:
        nextPoint = boundary.pop()
        if include(nextPoint, seed):
            visited[nextPoint] = 2
            for (x, y) in neighbors(nextPoint):
                if visited[x,y] == 0:
                    boundary.append((x, y))
                    visited[x,y] = 1
```

Region Growing example

- Pick a single seed pixel
- Inclusion test is up to you:

```
def include(p, seed):
   test = ??
   return test
```




T=150

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.
- Color 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 neighborhood 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

Interim Summary

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

Thursday:
More segmentation