BLG453E COMPUTER VISION

Fall 2021 Term

Week 9

İstanbul Technical University
Computer Engineering Department

Instructor: Prof. Gözde ÜNAL

Teaching Assistant: Yusuf Hüseyin ŞAHİN

Thanks: Some Slides in this presentation are from the lecture notes of Prof. Greg Slabaugh at City University of London

1

Learning Outcomes of the Course

Students will be able to:

1. Discuss the main problems of computer (artificial) vision, its uses and applications
2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images
3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images
4. Construct least squares solutions to problems in computer vision
5. Describe the idea behind dimensionality reduction and how it is used in data processing
6. Apply object and shape recognition approaches to problems in computer vision

2

Week 8-9: Image Segmentation

At the end of Week : Students will be able to (Learning Objectives of the week):

3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

3

Green Screen

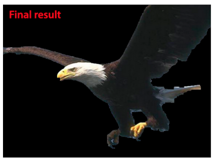
https://en.wikipedia.org/wiki/Chroma_key

[Prank: https://www.youtube.com/watch?v=wFa6NIgSRKQ](https://www.youtube.com/watch?v=wFa6NIgSRKQ)

4

Segmentation

- Segmentation has a variety of practical uses



cut out

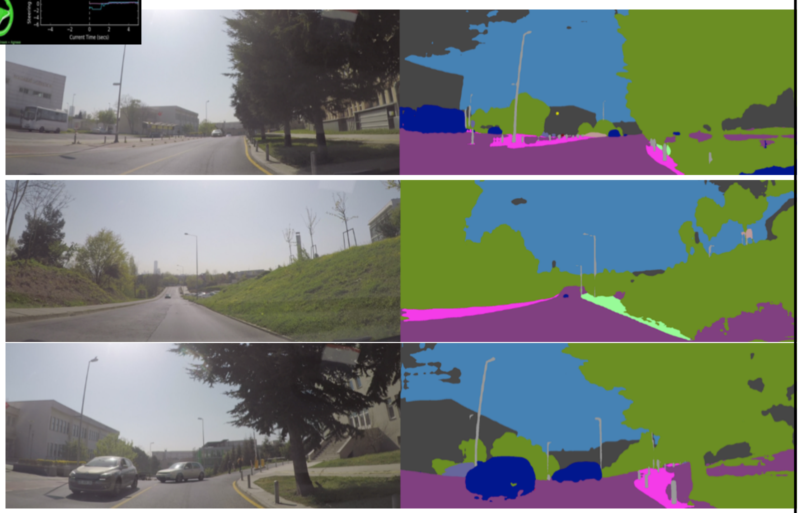
recolouring
(courtesy of Tommaso)

compositing

5

Segmentation

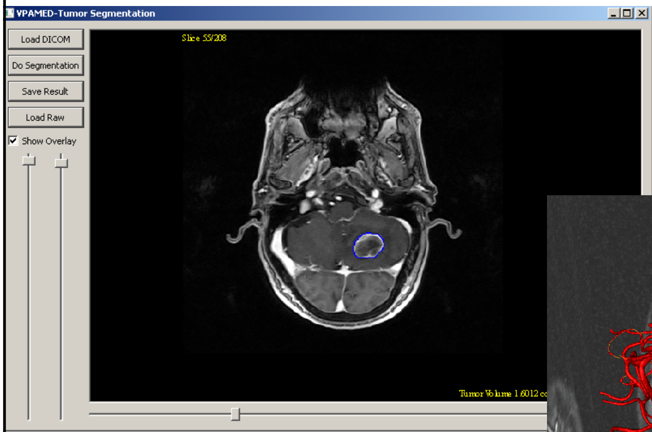
- Autonomous vehicles: Segmentation is a crucial early stage processing for later decision stages

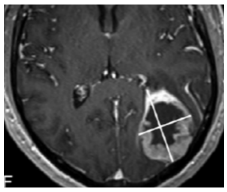


S.C. Yurtkulu, Y. Şahin, G. Unal. Semantic Segmentation with Extended DeepLabv3 Architecture, SİU 2019.

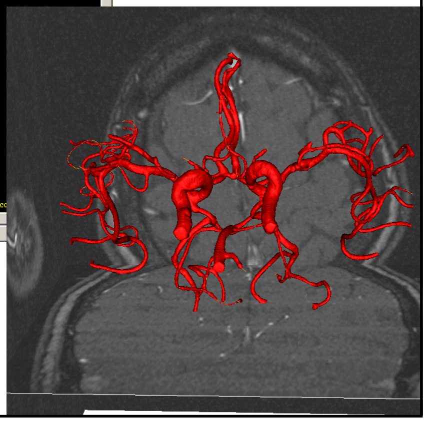
6

Segmenting anatomical structures





measurement



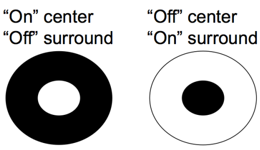
Some results from
Dr. Andac Hamamcı and Dr. Suheyla Cetin's PhD thesis results at Prof. G. Unal's group

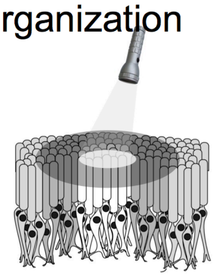
7

Center Surround Organization in the Eye Neurons

Center Surround Organization

Optimal stimuli (polka dots)





Some neurons receive net excitatory in the center and net inhibitory from the surround, and others receive net inhibitory in the center and net excitatory synapses in the surround.

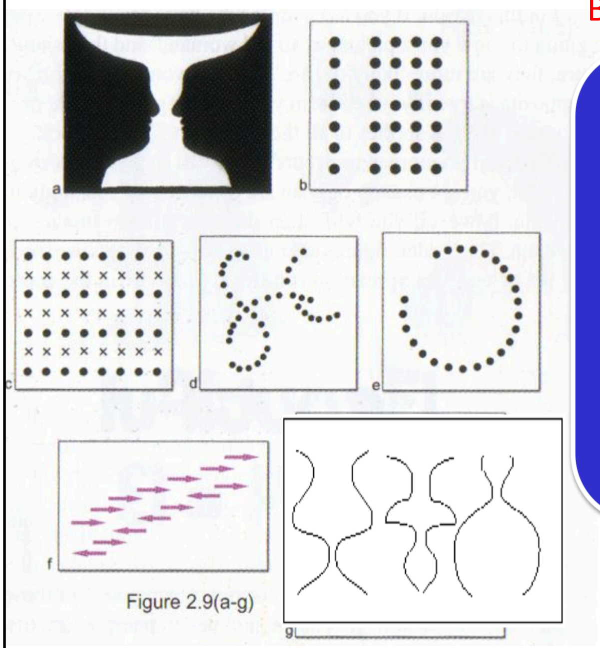
Video min: 1:04: http://www.youtube.com/watch?v=KE952yueVLA

As a result: retinal ganglion cell are more sensitive to patterns of light that are not uniform, but are changing in a way that, that matches their center surround organization
That can make them most responsive to light spots on a dark background, which is referred to as having an on center and off surround, or dark spots on a light background, which is referred to as having an off center and on surround.
This is a step towards identifying boundaries of objects (happens later in visual cortex)

Prof. J. Groh, Duke University

8

Biological basis:



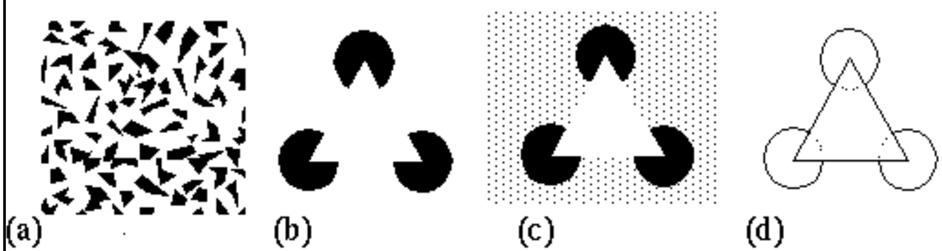
Gestalt Phenomena

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

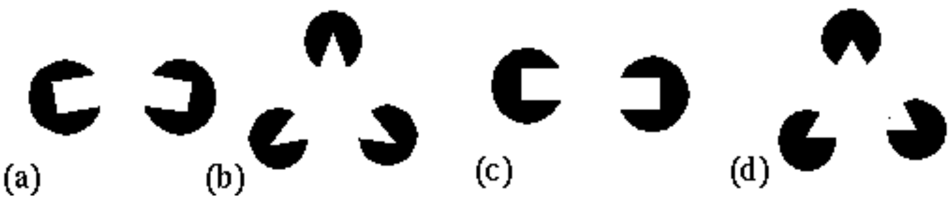
Gestalt psychology identifies several properties that result in grouping/segmentation

9

Consequence:
Groupings by Invisible Completions



Stressing the invisible groupings:



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

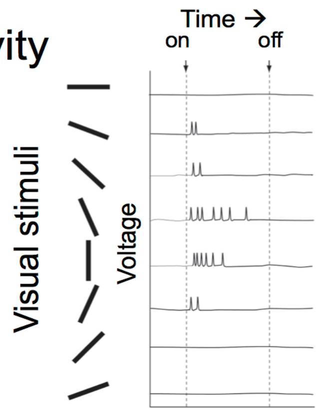
11

Brain Visual Maps

The brain's visual system (primary visual cortex) is sensitive to orientation

Orientation Selectivity

Hubel, D. H. and T. N. Wiesel (1959). J Physiol 148: 574-591.



Series of visual stimuli presented to a cat: and responses are recorded from a neuron in visual cortex

video (6:13 – 6:33) orientation and receptive field sizes

Listen to the sounds (each corresponds to a spike) recorded from a neuron in the visual cortex

http://www.youtube.com/watch?v=8VdFf3egwfg

Hubel and Wiesel: The Nobel Prize in Physiology or Medicine 1981 *for their discoveries concerning information processing in the visual system"*

12

And the famous invisible dog eating under a tree:

Slide: Bradski&Trun

13

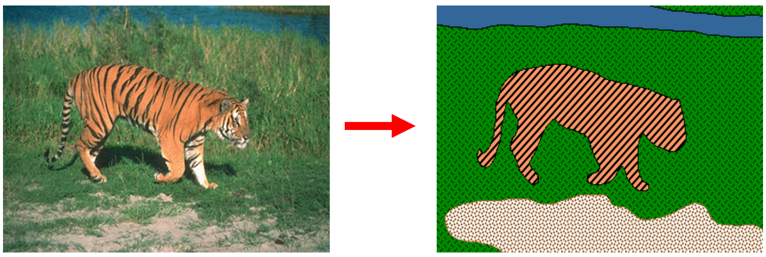
Segmentation

Essentially, segmentation is a grouping problem.

Depending on the image, this can be difficult!
Humans are adept at visual grouping.

14

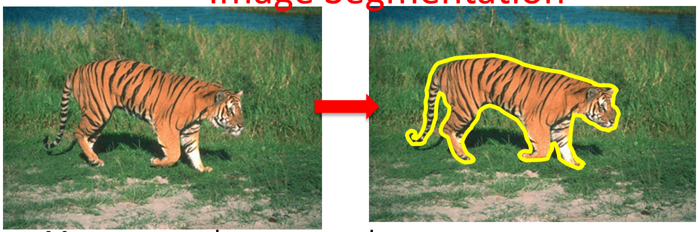
From Images to Objects/Regions: Segmentation



- Segmentation partitions an image into regions of interest.
- The first stage in many automatic image analysis systems.
- A complete segmentation of an image domain is a finite set of regions R_1, ..., R_N, such that $\Omega = \cup R_i$

15

Image Segmentation



How could we segment a scene into objects and others?

- Many approaches proposed
  - cues: color, contours, textural patterns
  - automatic vs. user-guided
  - no clear winner

- Is user-input required?
- * Our visual system is proof that automatic methods are possible, classical image segmentation methods are automatic
- * Argument for user-directed methods? only user knows desired scale/object of interest

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

16

Problem of Image Segmentation:

Segmentation: Labeling pixels in an image so that the image is partitioned into “meaningful” regions

Input: a grayscale or color image or image volume

Output: a binary (K-ary) image with labels

e.g. Labels = “foreground” vs “background”
or Labels = 0 or 1

Labels: 0, 1, 2, ... K (K objects) if more than two regions exist:

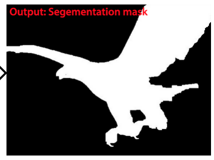
Image from Kohli, MICCAI 2009 tutorial

17

Segmentation

Image segmentation partitions an image into regions (aka *segments*).

Segmentation Algorithm



Input

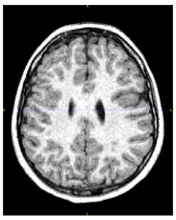
Output

You can think of image segmentation as assigning a label to each pixel in the image. In the case above, there are two labels: *bird* (foreground, white) and *non-bird* (background, black).

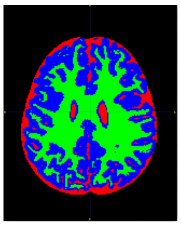
18

Segmentation

- There can be any number of labels

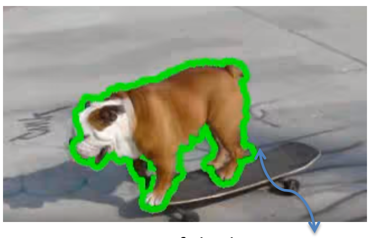


Segmentation Algorithm

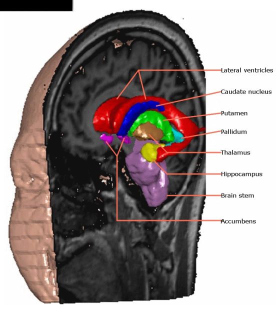


Four labels:
1. Non-brain
2. Grey matter
3. White matter
4. CSF

- Different types of data; more than two dimensions



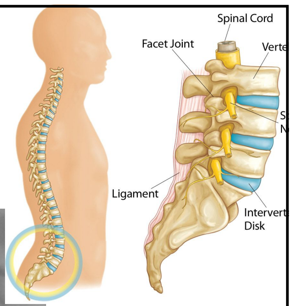
Green: Contours of the binary segmentation map

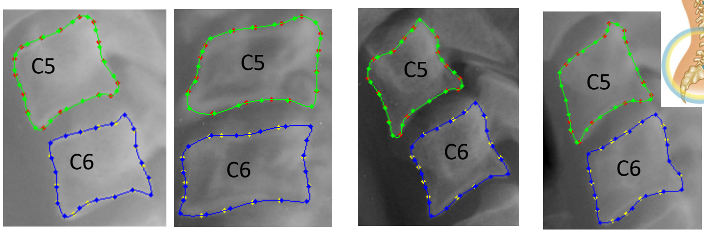


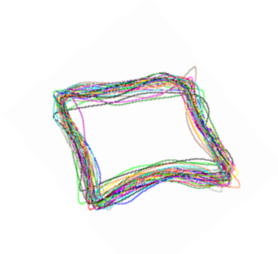
19

Segmentation

- The boundary of a segment provides a *shape*.







A collection of aligned C4 vertebra shapes. From this, we can do all sorts of interesting things:
- Population statistics
- Mean, variation (principal component analysis)
- (more)

20

Under and over segmentation

- An image that has been broken into too many segments is considered to be *over-segmented*

- Not enough segments is considered *under-segmented*

Pink flowers merged together

21

Bird counting

- How would you determine the number of birds in this image?

http://350sav.fotomaps.ru/flock-of-birds.php

- One way is to count the dark blobs. First we need to *find* the dark blobs.
- Let's load the image and convert to grayscale

```
I = cv2.imread("birds.png",0)
```

Or use a conversion from rgb2gray

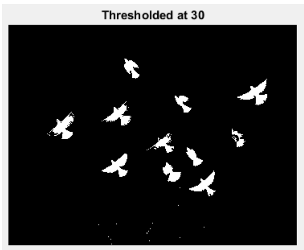
22

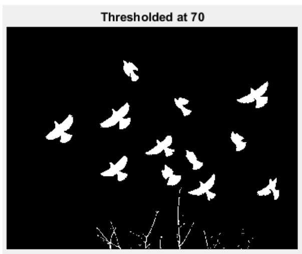
Thresholding

- The simplest form of segmentation is to *threshold* an image. This assigns a value of 1 to any pixel that satisfies the threshold, otherwise, 0. A *binary* image is produced

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

>T if looking for bright regions





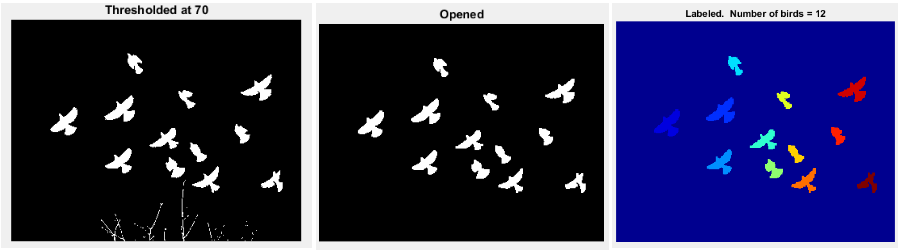

```
I = np.uint8((I < 30)
            * 255)
cv2.imshow('Thresholded'
           ' at 30', I)
cv2.waitKey(0)
```

```
I = np.uint8((I < 70)
            * 255)
cv2.imshow('Thresholded'
           ' at 70', I)
cv2.waitKey(0)
```

```
I = np.uint8((I < 120)
            * 255)
cv2.imshow('Thresholded'
           ' at 120', I)
cv2.waitKey(0)
```

23

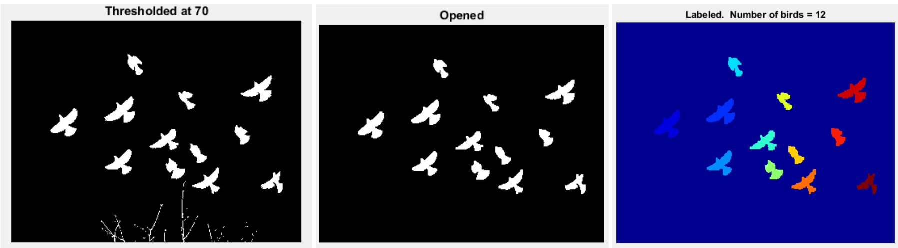
Bird counting




```
I = cv2.imread("birds.png",0)
ret,B=cv2.threshold(I, 70, 255, cv2.THRESH_BINARY_INV)
cv2.imshow("Thresholded    at 70",B)
J = cv2.morphologyEx(B, cv2.MORPH_OPEN, np.ones((3,3),np.uint8))
cv2.imshow('Opened',J)
labelCount, labels = cv2.connectedComponents(J)
# Map component labels to hue val
label_hue = np.uint8(179*labels/np.max(labels))
blank_ch = 255*np.ones_like(label_hue)
labeled_img = cv2.merge([label_hue, blank_ch, blank_ch])
```

24

Bird counting




```
# cvt to BGR for display
L = cv2.cvtColor(labeled_img, cv2.COLOR_HSV2BGR)
# set background label to black
L[label_hue==0] = 0
cv2.imshow('Labeled. Number of birds = ' + str(labelCount-1), L)
#backgorund is also labeled
cv2.waitKey(0)
cv2.destroyAllWindows()
```

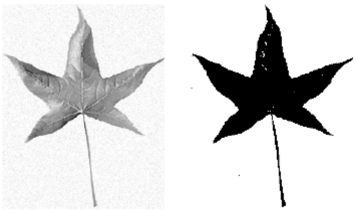
25

So, the simplest Segmentation approach is global thresholding

Simplest way to segment an image : create a binary image by separating the image domain into two regions

It labels each pixel **in** or **out** of the region of interest by comparison of the grey level with a threshold *T*:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{otherwise} \end{cases}$$



Basic Thresholding Algorithm

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

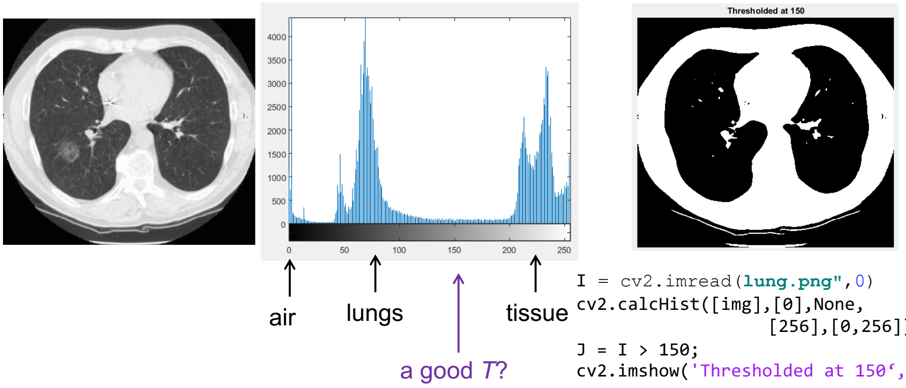
Don't write it like this at home!

Picture: Bryan Morse notes, BYU

26

Finding a good threshold ?

- The success of thresholding often depends on finding a good threshold *T*.
- Recall the histogram h(I) of an image. It counts how many pixels of a particular intensity are in the image.




```
I = cv2.imread(lung.png",0)
cv2.calcHist([img],[0],None,
                          [256],[0,256])
J = I > 150;
cv2.imshow('Thresholded at 150', J);
```

27

Recall Histogram:

Recall that a normalized histogram is a probability distribution:

$$p(I) = \frac{n(I)}{n}$$

That is, the number of pixels n(I) having grayscale intensity I as a fraction of the total number of pixels n.

28

Look at Histogram of the image (CT image slice from the brain)

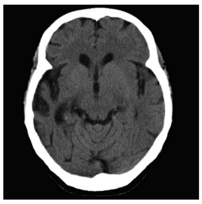
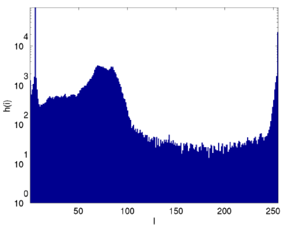
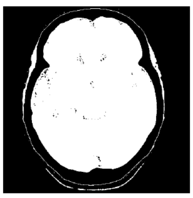


Image.



Histogram of CT brain image.
3 peaks: background, brain, skull.
Pick thresholds from histogram: 11 and 185.

Background.



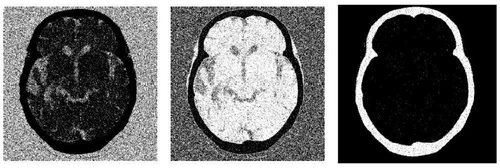
Brain.

Skull.

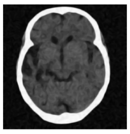
Slide: M. Niethammer

29

Thresholding
(noisy image)

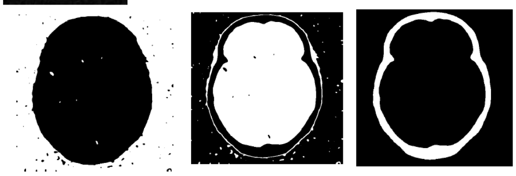


Background. Brain. Skull.



Thresholding
(smoothed image)

Smoothing helps local thresholding.

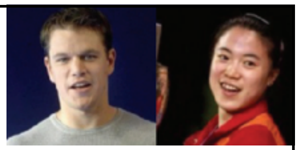


Background. Brain. Skull.

M. Niethammer

30

Skin segmentation



* Segmentation of skin a notoriously difficult problem, given the variation observed in practice. One can perform simple skin detectors using *colour thresholding*.

Examples below: Work in a color space (other than (R,G,B)): Y, U, and V values; Y stores the brightness (luminance), and the color (chrominance) stored as U and V values.

[Chai et al.]

Convert to YUV
Skin = (77 <= U <= 127) and
(133 <= V <= 173)

[Al-Tairi et al.]

Convert to YUV
Skin = (80 < U < 130) and
(136 < V < 200) and
(V > U) & (R > 80) and
(G > 30) & (B > 15) and
|R-G| > 15

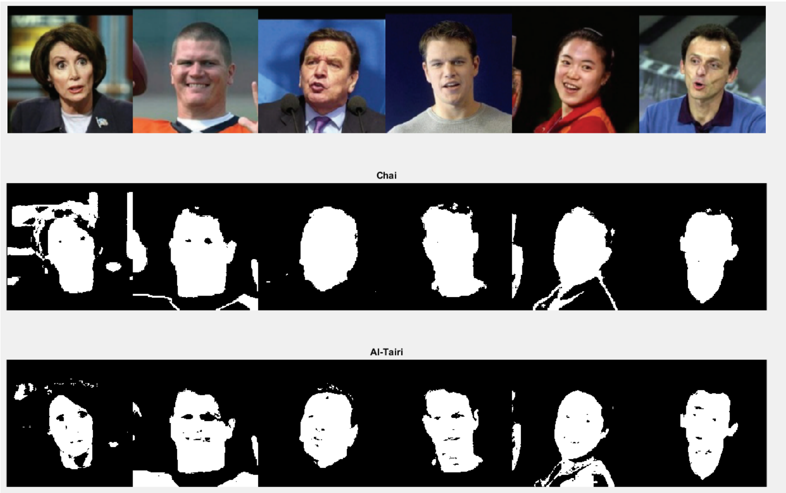
```
I = cv2.imread("faces.png");
cv2.imshow('Faces',I)
YUV= cv2.cvtColor(I, cv2.COLOR_RGB2YCR_CB);
U=YUV[:,:,1]; V=YUV[:,:,2]; R=I[:,:,2]
G=I[:,:,1]; B=I[:,:,0]
rows,cols,planes=I.shape

# Chai et al.
skin=np.zeros([rows,cols],dtype=np.uint8)
ind=(77<=U) & (U<=127) & (133<=V) & (V<= 173 )
skin[ind]=255; cv2.imshow('Chai',skin)

# Al-Tairi et al.
skin=np.zeros([rows,cols],dtype=np.uint8)
ind=(80    <  U) &  (U <  130)    &  \
    (136   <  V) &  (V <= 200     )& \
    (V >  U) &  (R >  80)     &  \
    (G >  30)     &  (B >  15 )& \
    (abs(R-G)  >  15)
skin[ind]=255; cv2.imshow('AL-Tairi',skin);
cv2.waitKey(0);cv2.destroyAllWindows()
```

31

Some results from the papers



Q. Can you implement a histogram-based threshold selection for skin segmentation?

32

Rule-based

- We have a set of rules that label a pixel as skin or non-skin.
- More abstractly, we can think of this as

colour → Set of rules → Skin (label = 1)
colour → Set of rules → Non-skin (label = 0)

⇨ This is a rule-based classifier

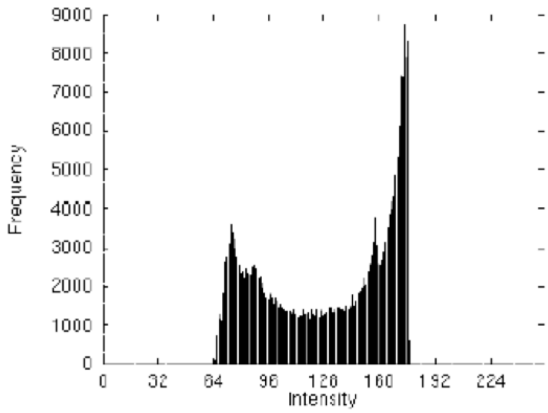
In this case, the rules have been generated by a human by looking at many datasets.

- An alternative is to use *machine learning* to develop a supervised approach to skin detection.
- Supervised learning is another way to perform segmentation, topic of another time.

33

Q : How to find thresholds automatically?

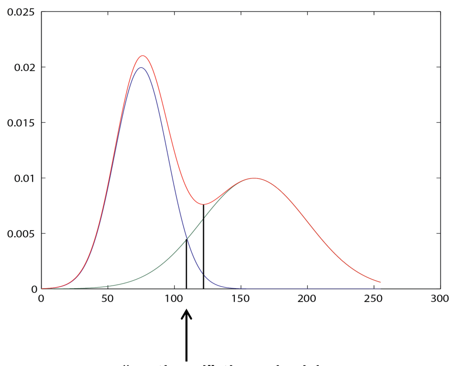
THQ: Histogram Threshold selection for segmentation



34

Automatic thresholding methods

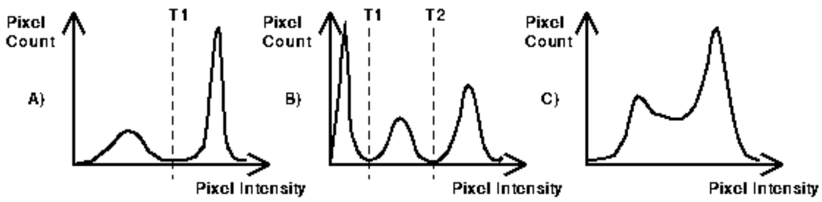
- Many automatic methods to determine the threshold assume a *bimodal* histogram, meaning there are two peaks (one for foreground, the other for background)



Always ask: "Optimal" in what sense?

35

Bimodal histogram Trimodal histogram:



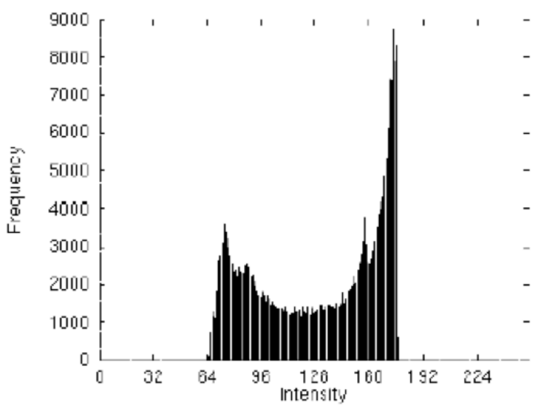
For instance, the segmentation thresholds T are set as the local minima of the histogram

Methods of threshold detection: Usually based on histogram analysis

36

Finding Modes of Histograms

Idea: find two peaks (modes) and the local minimum in between



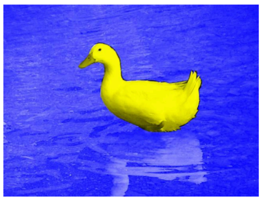
Problems :
Noisy Histograms
Are there better ways?

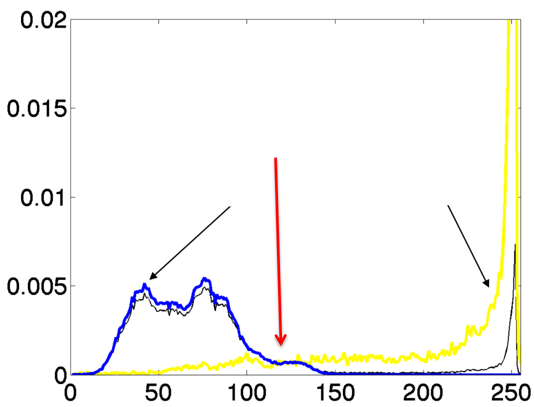
Bryan Morse, BYU

37

Optimal Thresholding

- Idea: the histogram is the sum of two overlapping distributions.
- Optimal threshold: the overlapping point of these distributions.





38

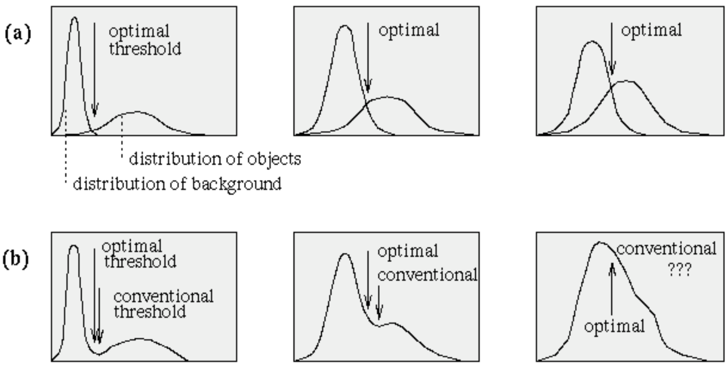


Figure 5.4 *Grey level histograms approximated by two normal distributions; the threshold is set to give minimum probability of segmentation error: (a) Probability distributions of background and objects, (b) corresponding histograms and optimal threshold.*

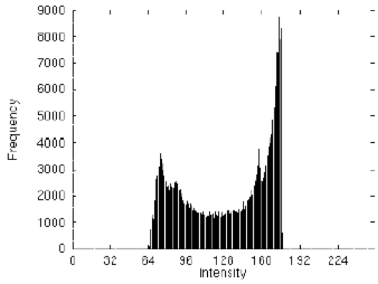
Problem 1: Don't know the distributions

Problem 2: "Optimal threshold" as the intersection of two distributions is not the same as the local minimum.

Figure: Bryan Morse, BYU

39

"Optimal" Thresholding
One approach was proposed by Otsu



One can express the total variance of image intensities as:

$$\sigma^2(T) = \underbrace{\sigma^2_{Within}(T)}_{\text{Within-class variance}} + \underbrace{\sigma^2_{Between}(T)}_{\text{Between-class variance}}$$

T: unknown threshold

Since the total variance is constant and independent of *T*, the effect of changing the threshold is merely to move the contributions of the two terms back and forth.

So, *minimizing the within-class variance is the same as maximizing the between-class variance with respect to "optimal" threshold T to be estimated.*

40

Digression:

Segmentation by Otsu Method: Based on the Law of Total Variance* from Probability Theory

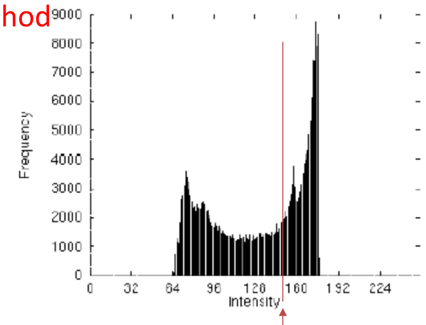
$$\mathrm{Var}(X) = \underbrace{E\,[\,\mathrm{Var}(X|Y)\,]}_{\text{Average Variability within each group}} + \underbrace{\mathrm{Var}(\,E\,[X|Y]\,)}_{\text{Variability between groups}}$$

- Conditioning on different Y's (= groups) in a conditional universe.
- E.g. Y= Foreground , Y= Background

* E.g. see Bertsekas et al's Book. Introduction to Probability Theory. Section 4.3.

41

"Optimal" Thresholding by Otsu Method



T: current threshold

Total variance is constant

$$\sigma^2(T) = \underbrace{\sigma^2_{Within}(T)}_{\text{Within-class variance}} + \underbrace{\sigma^2_{Between}(T)}_{\text{Between-class variance}}$$

T: unknown threshold

Minimize wrt T

or equivalently Maximize wrt T

The nice thing about this is that we can calculate the quantities in $\sigma^2_{Between}(T)$ *recursively* as we run through the range of *T* values when we compute the variance

42

Optimal thresholding (Otsu)

- Otsu's method finds the threshold *T* that minimises the expected value of the within-class variance, defined as

$$\sigma^2_{within} = w_f \sigma_f^2 + w_b \sigma_b^2$$

where σ_f^2 and σ_b^2 are the variances of the foreground and background, and

$$w_b = \sum_{I=0}^{T} p(I) \qquad w_f = \sum_{I=T+1}^{255} p(I)$$

are weights formed by summing the pdf over the background and foreground intensities, respectively.
i.e. those are probabilities of the background and the foreground regions, respectively.

In OpenCV, Otsu thresholding is implemented with the function `threshold with cv2.THRESH_OTSU parameter`, which returns a threshold in the range of 0 to 1.

43

Optimal thresholding (Otsu)

Easier way to calculate:

- If you subtract the "within-class" variance from the total variance of the combined distribution, you get the "between-class" variance:

$$\sigma^2_{Between}(T) = \sigma^2_{total} - \sigma^2_{within}(T)$$

$$= w_B(T)[\mu_B(T) - \mu]^2 + w_F(T)[\mu_F(T) - \mu]^2$$

where σ^2_{total} is the total variance and μ is the total mean.

* Since total variance is independent of the threshold, maximizing the between-class variance is the same as minimizing the within-class variance.

Substituting:

$$\mu = w_B(T)\mu_B(T) + w_F(T)\mu_F(T)$$

and simplifying, we get:

$$\sigma^2_{Between}(T) = w_B(T)w_F(T)[\mu_B(T) - \mu_F(T)]^2$$

44

"Optimal" Thresholding (Otsu's method)

Goal: Find T that maximizes the measure:

$$\sigma^2_{Between}(T) = w_B(T)w_F(T)[\mu_B(T) - \mu_F(T)]^2$$

ALGORITHM:

i. For each potential threshold T:

1. Separate the pixels into two clusters according to the threshold T
2. Find the mean of each cluster (good thing: you don't have to calculate the variance in this way – if you minimize within-class variance instead, you have to calculate the variances of each cluster)
3. Calculate the above measure

ii. Output: Choose the threshold T that gives the maximum measure.

45

Extra Slide 1: OTSU'S THRESHOLDING Method: Details for an even more efficient implementation
Note the notation change from w_ weights to n_ and subscripts B(background) and O(object)

Idea: select T to minimize the *within-class* variance—the weighted sum of the variances of each cluster:

$$\sigma^2_{\text{Within}}(T) = n_B(T)\,\sigma^2_B(T) + n_O(T)\,\sigma^2_O(T)$$

where

$$\begin{aligned} n_B(T) &= \sum_{i=0}^{T-1} p(i) \\ n_O(T) &= \sum_{i=T}^{N-1} p(i) \\ \sigma^2_B(T) &= \text{the variance of the pixels in the background } (< T) \\ \sigma^2_O(T) &= \text{the variance of the pixels in the foreground } (\geq T) \end{aligned}$$

and $[0, N-1]$ is the range of intensity levels.

46

Extra Slide 2: A very efficient implementation of Otsu

- Better Still: Update $n_B(T)$, $n_O(T)$, and the respective cluster means $\mu_B(T)$ and $\mu_O(T)$ with the pixels that move from one cluster to the other as T increases:

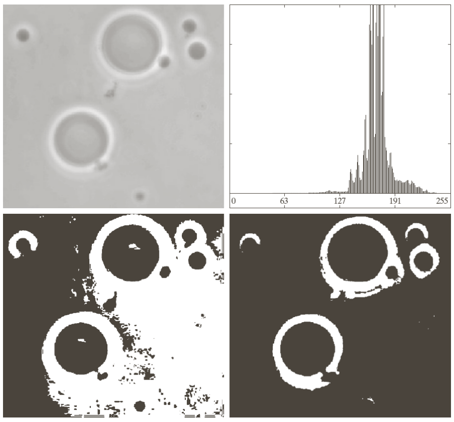
$$n_B(T+1) = n_B(T) + n_T$$
$$n_O(T+1) = n_O(T) - n_T$$

$$\mu_B(T+1) = \frac{\mu_B(T)\,n_B(T) + n_T T}{n_B(T+1)}$$

$$\mu_O(T+1) = \frac{\mu_O(T)\,n_O(T) - n_T T}{n_O(T+1)}$$

47

Result of Otsu



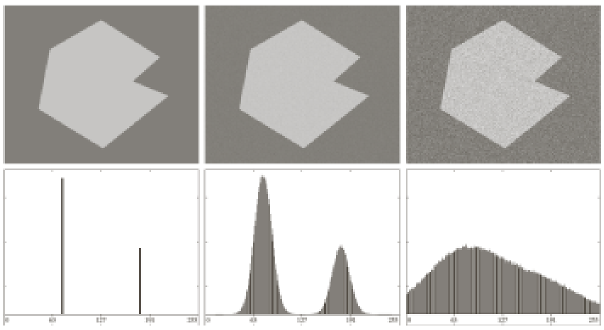
a b
c d

FIGURE 10.39 (a) Original image. (b) Histogram (high peaks were clipped to highlight details in the lower values). (c) Segmentation result using the basic global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

Digital Image Processing, Gonzalez and Woods

48

Effect of Noise in Thresholding



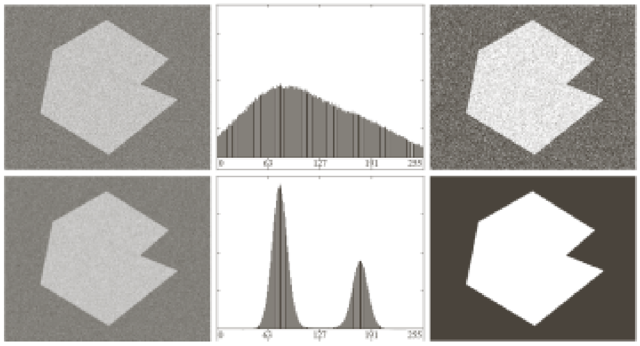
a b c
d e f

FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Digital Image Processing, Gonzalez and Woods

49

Effect of Noise and Smoothing in Thresholding



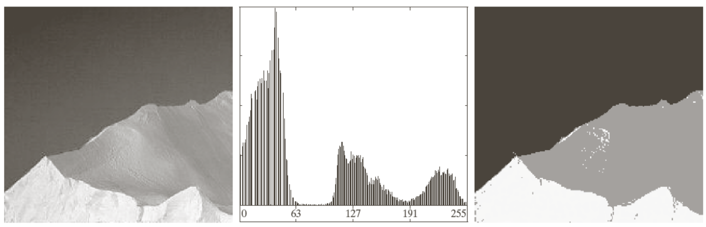
a b c
d e f

FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5 × 5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

Digital Image Processing, Gonzalez and Woods

50

Ternary Segmentation: i.e. Segment the image into 3 regions



a b c

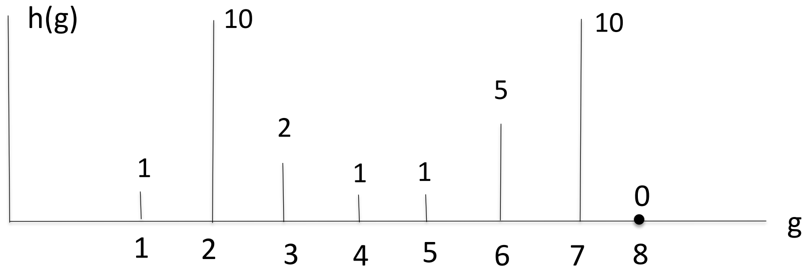
FIGURE 10.45 (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

- Note that Otsu's method can be extended to N regions, then we will need to search for N-1 thresholds to separate those N regions.
- E.g. N= 3 → We need 2 thresholds.

Digital Image Processing, Gonzalez and Woods

51

THQ: Threshold selection techniques



Use Otsu's Thresholding technique to find the "optimal" threshold (Only try for T=4 and T=5)

52

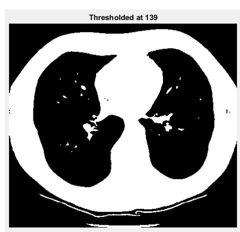
Iterative Thresholding method (Another method)

1. Set T to some initial value, and determine foreground and background pixels.
2. Compute m_b, the mean of the background, and m_f, the mean of the foreground, based on the current value of T.
3. Set $T = (m_b + m_f)/2$
4. Go to step 2 until convergence (when T no longer changes).

```
T=50
lastT=0
while abs(T-lastT) > 1 :
    lastT=T
    mf= np.mean(I[I>T])
    mb= np.mean(I[I<=T])
    T=0.5*(mb+mf)

cv2.imshow("Lungs",I)
```

In this example, the loop executes 5 times, with T starting at 50, and going to 81.9, 126.7, 138.2, 139.3, 139.4



An idea to set the initial Threshold: Start with m_b as the average of the four corner pixel, which is assumed to be the background, and m_f as the average of everything else. Go to step 3 of the algorithm above and continue.

53

Idea behind Iterative Threshold Selection

- Idea: pick a threshold such that each pixel on each side of the threshold is closer in intensity to the mean of all pixels on that side of the threshold than the mean of all pixels on the other side of the threshold.
- Let

$$\begin{aligned}\mu_B(T) &= \text{the mean of all pixels less than the threshold (background)}\\ \mu_O(T) &= \text{the mean of all pixels greater than the threshold (object)}\end{aligned}$$

We want to find a threshold such that the greylevels for the object are closest to the average of the object:

$$\forall g \geq T : |g - \mu_B(T)| > |g - \mu_O(T)|$$

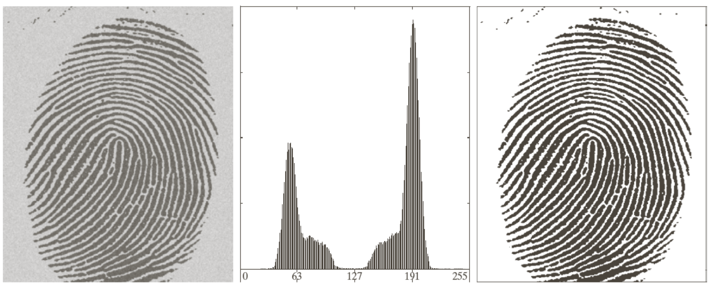
and the greylevels for the background are closest to the average of the background:

$$\forall g < T : |g - \mu_B(T)| < |g - \mu_O(T)|$$

Q: Can you recall what this algorithm is from LFD ?

A: This is exactly the K-means algorithm!

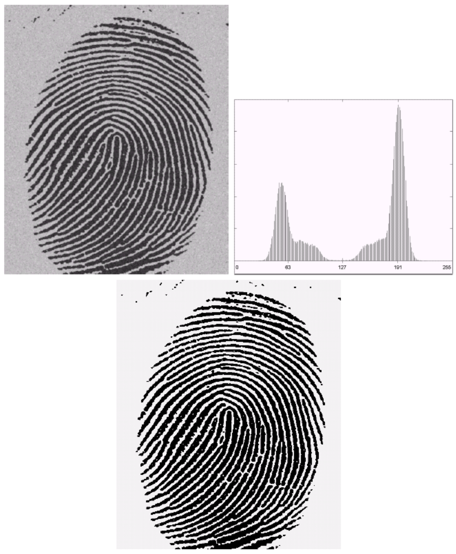
54



a b c

FIGURE 10.38 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)

55

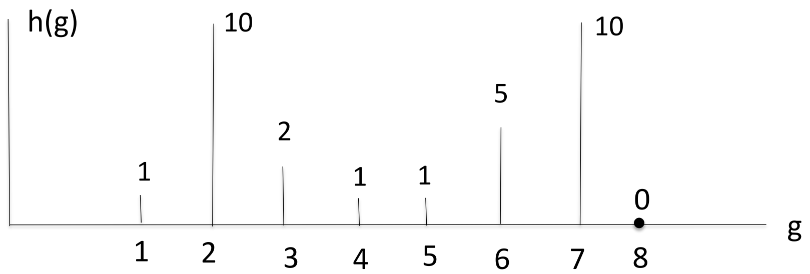


a b
c

FIGURE 10.29 (a) Original image. (b) Image histogram. (c) Result of segmentation with the threshold estimated by iteration. (Original courtesy of the National Institute of Standards and Technology.)

56

THQ: Threshold selection techniques



Use Iterative Thresholding technique to find the “optimal” threshold (Only try T=4 and T=5)

57

Gaussian Mixture Modeling

Idea: based on approximation of the histogram of an image using a weighted sum of two or more probability densities with normal distribution.

- Suppose that region intensities are each normal distributions (Gaussians).

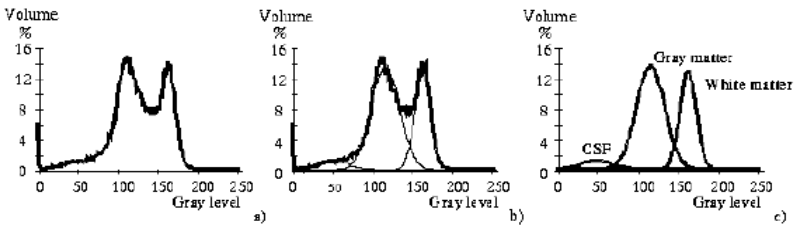


Figure 5.5 *Segmentation of 3-D T1-weighted MR brain image data using optimal thresholding: (a) Local gray level histogram, (b) fitted Gaussian distributions; global 3-D image fit, (c) Gaussian distributions corresponding to WM, GM, and CSF.*

Picture: Bryan Morse, BYU

58

Gaussian Mixture Modeling

- Each of the distributions has a mean (μ_i) and a standard deviation (σ_i) *independent of the threshold we choose.*
- Foreground/background case:

$$h_{\text{model}}(g) = n_B \; e^{-(g-\mu_B)^2/2\sigma_B^2} + n_O \; e^{-(g-\mu_O)^2/2\sigma_O^2}$$

- *Assumes that there exists two distributions and we must find them.* (Once we know parameters, determining the best threshold is easy.)
- Six unknown parameters: $\vec{\mathbf{x}} = n_B, n_O, \mu_B, \mu_O, \sigma_B$, and σ_O
- How well does the sum of the distributions approximate the histogram?

$$F(\vec{\mathbf{x}}) = \sum_{0}^{N-1} \left[h_{\text{model}}(g) - h_{\text{image}}(g) \right]^2$$

- Choose parameters to minimize the error in the fit (F). [We won't cover the algorithm for doing this, just the basic idea.]

Q: Can you suggest a way to minimize F(x) to solve for the parameters of the Gaussian mixture?

59

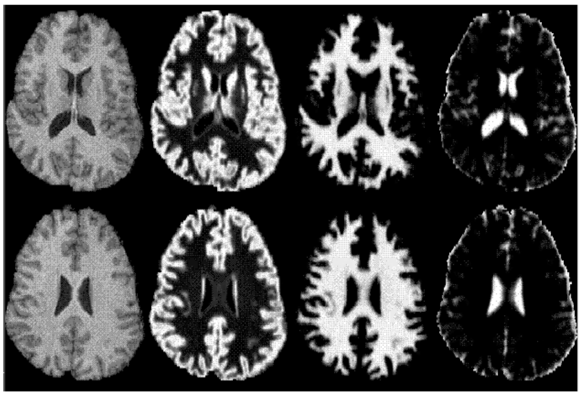


Figure 5.6 *Optimal MR brain image segmentation. Left column: Original T1-weighted MR images, two of 120 slices of the 3-D volume. Middle left: Partial-volume maps of gray matter. The brighter the voxel, the higher is the partial volume percentage of gray matter in the voxel. Middle right: Partial-volume maps of white matter. Right column: Partial-volume maps of cerebro-spinal fluid. Courtesy R.J. Frank, T.J. Grabowski, Human Neuroanatomy and Neuroimaging Laboratory, Department of Neurology, The University of Iowa.*

60

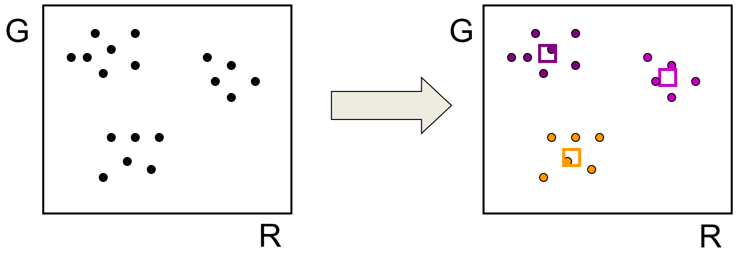
K means Clustering

- The optimal (iterative) thresholding method described on the previous slides is an example of the K means algorithm, where K = 2.
- K means is a clustering technique, in this case grouping pixels of similar intensity into two or more (K) groups, or *clusters*.
- K means can be used to cluster data into any number (K) classes. The data can be based on intensity, colour, or any attribute associated with a pixel (e.g., texture, depth, etc.)
- It requires K to be provided in advance.

61

Segmentation as a Clustering Problem

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



Objective

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

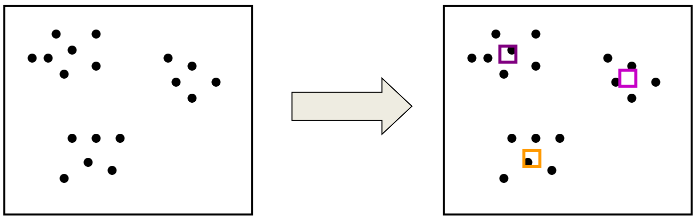
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

R. Szeliski Slides

63

Break it down into subproblems

- Suppose I tell you the cluster centers c_i
  - Q: how to determine which points to associate with each c_i?
  - A: for each point p, choose closest c_i



Suppose I tell you the points in each cluster

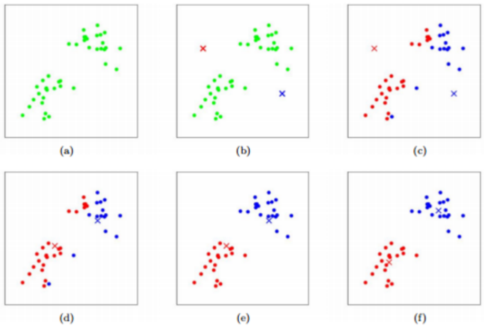
- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

R. Szeliski Slides

64

Algorithm: K-means clustering

1. Start with a set of N points (green points below)
2. Randomly initialize the K cluster centers, c_1, ..., c_K (x's below)
3. Assignment: Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
4. Update: Given points in each cluster, calculate new means c_i
   - Set c_i to be the mean of assigned points in cluster i
5. Go to 3 until convergence (e.g. means c_i no longer change)



Java demo (please check if the link still works)
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

65

K-means clustering

K-means minimizes the objective function: Sum of Within-Cluster Variance, which is also Mean Squared Error within each cluster.
Equivalently: it is based on Sum of Squared Distances (SSD) between points in a cluster and the mean point

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

Properties
- Will always converge to *some* solution
- Can be a “local minimum”
  - does not always find the global minimum of objective function
- Why?

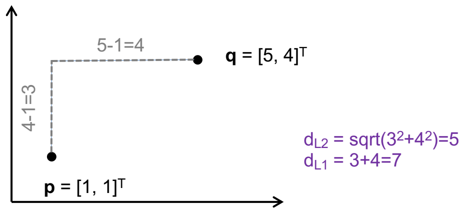
66

Issues with K means

- What is K?
  - In some applications, there is an obvious K. In others, it may be unclear.
- How to initialise the means? Potentially different results each time the algorithm is run
  - One can rerun the algorithm multiple times and merge the results
- Dealing with high dimensional data
  - Finding nearest neighbours can be slow (e.g. kd tree can help)
- How do you measure distance?
  - Normally the L2 norm (Euclidean distance) is selected, but there are other options, for example, the L1 distance (aka CityBlock distance).

$$d_{\mathrm{L2}}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_i (p_i - q_i)^2}$$

$$d_{\mathrm{L1}}(\mathbf{p},\mathbf{q}) = \sum_i |(p_i - q_i)|$$



67

K means

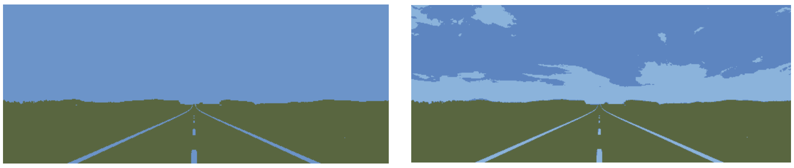
- OpenCV has support for K means using with the [kmeans] function. It requires an NxM matrix. Here, N is the number of pixels, and M are the features (colours) used in the grouping. This can be achieved using the reshape function.

```
img = cv2.imread('road_sky.jpg'); cv2.imshow('Original' ,img)
Z = img.reshape((-1,3))
# convert to np.float32
Z = np.float32(Z)
# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
K = 3; attemps=3
ret,label,center=cv2.kmeans(Z,K,None,criteria,attemps,cv2.KMEANS_RANDOM_CENTERS
# Now convert back into uint8, and make original image
center = np.uint8(center)
res = center[label.flatten()]; res2 = res.reshape((img.shape))
cv2.imshow('K = ' + str(K),res2);cv2.waitKey(0);cv2.destroyAllWindows()
```

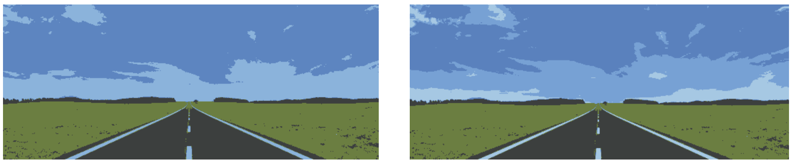
70

K means

- Different results for different K



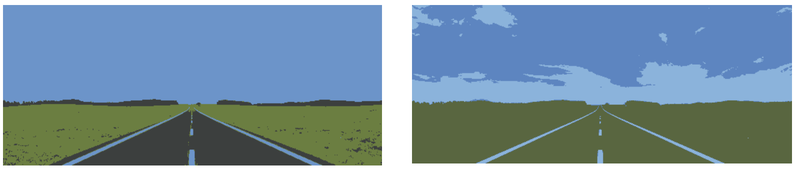
K = 2 K = 3



K = 4 K = 5

K means

- Different results for different *runs* due to random initialisation.



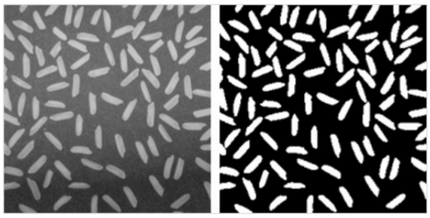
K = 3 K = 3

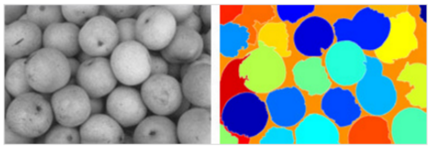
- Replicates (running the algorithm multiple times) can produce more consistent results

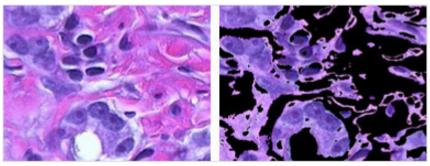
```
[clusterID, clusterCentre] = kmeans(R, 3, 'Replicates', 3);
```

OpenCV documentation and examples

- https://docs.opencv.org/3.4.3/d1/d5c/tutorial_py_kmeans_opencv.html







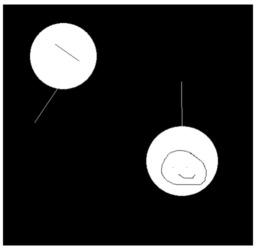
73

Mathematical morphology

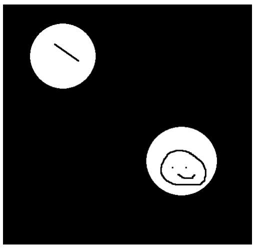
- A related class of nonlinear image processing techniques is known as mathematical morphology.
- These technique also use kernels, which are called *structuring elements* in the nomenclature of this branch of image processing.
- There are two basic operations, *erosion* and *dilation*. Erosion is like a min filter (over the structuring element extent on the image), and dilation is like a max filter. These operations are often used on binary images, but can be used on grayscale images too.
- A *closing* is dilation, followed by erosion. It fills holes smaller than the structuring element. An *opening* is erosion, followed by dilation. It removes objects smaller than the structuring element.
- Mathematical morphology may be useful in "cleaning up" segmentations.

75

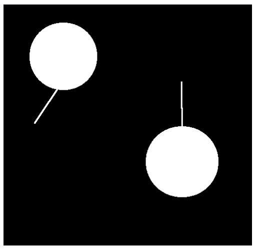
Erosion and Dilation



Original image



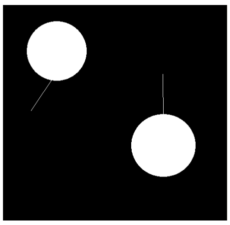
```
J = cv2.erode(I,np.ones((3,3),
              np.uint8))
cv2.imshow('erosion',J)
```



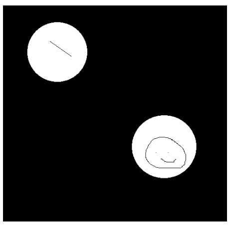
```
J = cv2.dilate(I,np.ones((3,3),
                          np.uint8))
cv2.imshow('dilation',J)
```

76

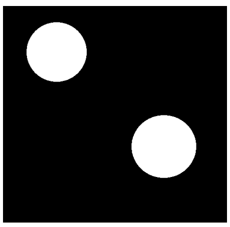
Closing and Opening



```
J = cv2.morphologyEx(I,
cv2.MORPH_CLOSE,
np.ones((3,3),np.uint8))
cv2.imshow('closing',J)
```



```
J = cv2.morphologyEx(I,
cv2.MORPH_OPEN,
np.ones((3,3),np.uint8))
cv2.imshow('opening',J)
```



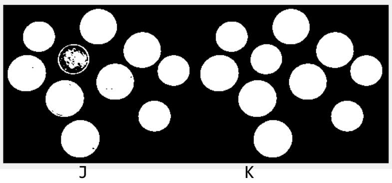
```
K = np.ones((3,3),np.uint8)
J1 = cv2.morphologyEx(I,
cv2.MORPH_OPEN, K)
J2 = cv2.morphologyEx(J1,
cv2.MORPH_CLOSE, K)
cv2.imshow('opening-
closing',J2)
```

77

Some other handy morphological functions

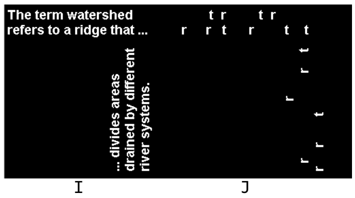
- imfill can be used to fill holes in each binary image region

```
I = imread('coins.png');
T = graythresh(I); % otsu threshold
J = I > 255*T;
K = imfill(J,'holes');
imshowpair(J,K,'montage');
```



- bwareafilt can filter regions in a binary image based on size (in pixels)

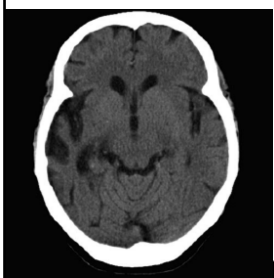
```
I = imread('text.png');
J = bwareafilt(I,[40 50]);
imshowpair(I,J,'montage');
```



- You can also use regionprops to get the properties of each region, like its area, perimeter, etc. Based on this, you can filter regions to keep those within prescribed limits

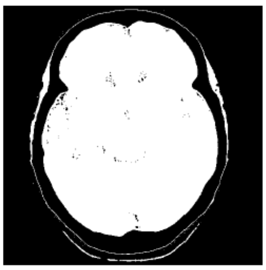
78

Recall Segmentation based on Thresholding



… needs to be cleaned up -> e.g., morphological operations.

Background.



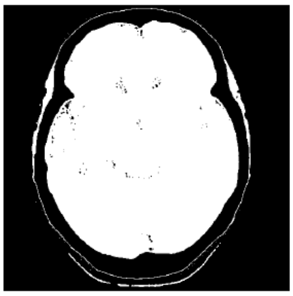
Brain.

Skull.

M. Niethammer

79

Extracting Connected Component



Original brain mask.

Largest connected component.

M. Niethammer

80

Dilation

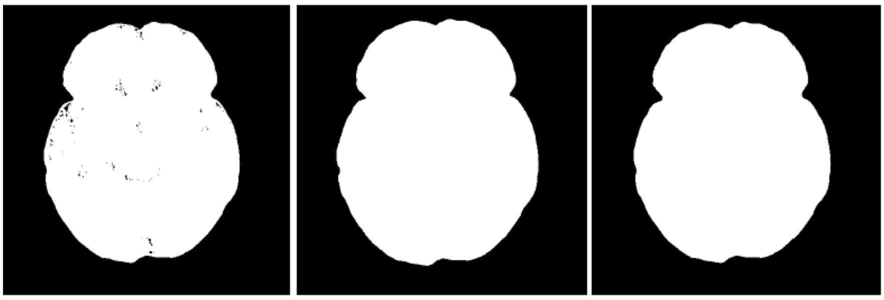
Original brain mask.

Dilated brain mask.
Fills in holes (5x5, SE).

M. Niethammer

81

Erode to Recover Original Size



Original. Dilated. Closed.

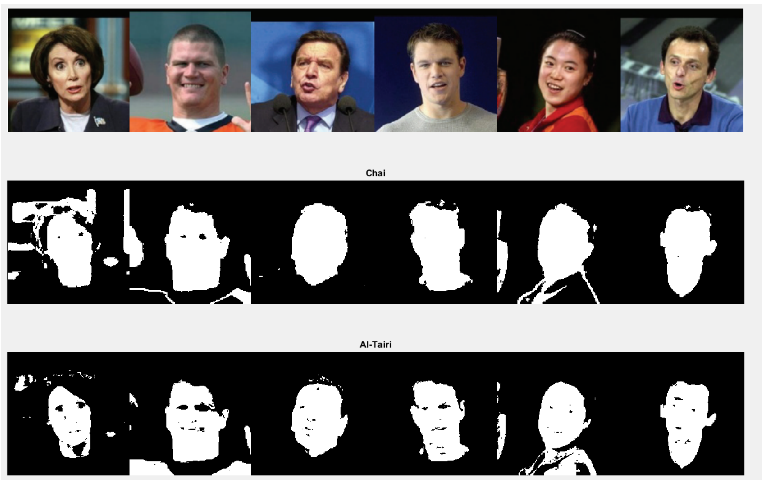
Dilation followed by Erosion = Closing

M. Niethammer

82

Evaluating segmentations

- What makes for a “good” segmentation?



83

Evaluating segmentations

- A common approach is to establish some ground truth. This can specify, for each pixel in the image if it should be part of the segmentation (or not).
- Example data for skin detection: http://web.fsktm.um.edu.my/~cschan/downloads_skin_dataset.html

84

The Segmentation Benchmark Datasets

http://host.robots.ox.ac.uk/pascal/VOC/

The PASCAL Visual Object Classes

- Provides standardised image data sets for object class recognition and segmentation
- Enables evaluation and comparison of different methods

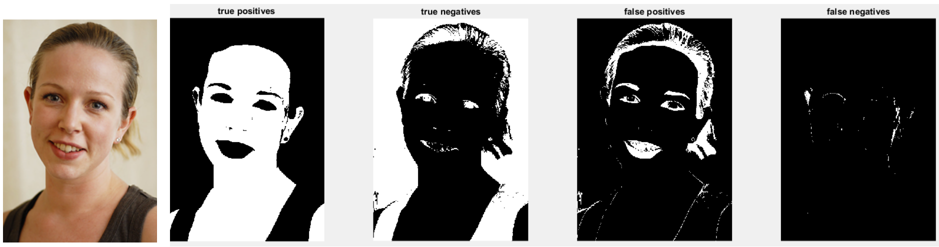
Figure from Arbelaez et al. Semantic Segmentation using Regions and Parts, 2012, CVPR. (figure 5)

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

85

Evaluating segmentations

- One can perform the segmentation, and using the ground truth, categorise pixels as:
  - True positives: correctly detected skin pixels
  - True negatives: correctly detected non-skin (background) pixels
  - False positives: non-skin pixels incorrectly detected as skin pixels
  - False negatives: skin pixels missed by the skin detector



86

Accuracy, sensitivity, specificity

- One can then produce quantitative measures such as:
  - Accuracy

    ACC = (TP + TN) / (TP + FP + FN + TN)

    This measures the number of correctly labelled pixels in the image

  - Sensitivity, or True Positive Rate (Recall)

    SENS = TP / P = TP / (TP + FN)

    This measures the percentage of positives correctly labelled

  - Specificity, or True Negative Rate

    SPEC = TN / N = (TN + FP)

    This measures the percentage of negatives correctly labelled

  - Precision or Positive Predictive Value

    PREC = TP / (TP+FP)

    This measures the percentage of positives among all predicted positives

where TP and TN are the number of true positives and true negatives, and FP and FN are the number of false positives and false negatives.

- A perfect result would have 100% accuracy, sensitivity, and specificity.

87

Example

```
I = cv2.imread("face.jpg")
T = cv2.imread("groundTruth.png",0)

YUV= cv2.cvtColor(I, cv2.COLOR_RGB2YCR_CB);
U=YUV[:,:,1]; V=YUV[:,:,2]; R=I[:,:,2]; G=I[:,:,1]; B=I[:,:,0]
rows,cols,planes=I.shape

# Al-Tairi et al.
skin=np.zeros([rows,cols],dtype=np.uint8)
ind=(80 < U)   & (U < 130) &  (136 < V)   & (V <= 200) & (V > U) & (R > 80) & (G
> 30)     & (B>15) & (abs(R-G) >    15)
skin[ind]=255

cv2.imshow('Image',I)
cv2.imshow('Ground Truth',T)
cv2.imshow('AL-Tairi',skin)


tpInd = (skin  == 255) & (T == 255)
tnInd = (skin  == 0)  & (T == 0)
fpInd = (skin  == 255) & (T == 0)
fnInd = (skin  == 0)  & (T == 255)
```

88

Example, part 2

```
tpImage = np.zeros([rows, cols],dtype=np.uint8)
tpImage[tpInd] = 255
tnImage = np.zeros([rows, cols],dtype=np.uint8)
tnImage[tnInd] = 255
fpImage = np.zeros([rows, cols],dtype=np.uint8)
fpImage[fpInd] = 255
fnImage = np.zeros([rows, cols],dtype=np.uint8)
fnImage[fnInd] = 255
cv2.imshow('true   positives',tpImage)
cv2.imshow('true   negatives',tnImage)
cv2.imshow('false  positives',fpImage)
cv2.imshow('false  negatives',fnImage)
tp = len(tpInd); tn = len(tnInd); fp = len(fpInd); fn = len(fnInd)
#  Compute    measures
accuracy = (tp + tn)/ (tp + tn + fp + fn)
sens = tp/(tp + fn); spec = tn/(tn + fp)
print('Accuracy = ' + str(accuracy) + ', sensitivity = ' + str(sens) + ',
specificity = ' +    str(spec))
cv2.waitKey(0); cv2.destroyAllWindows()
```

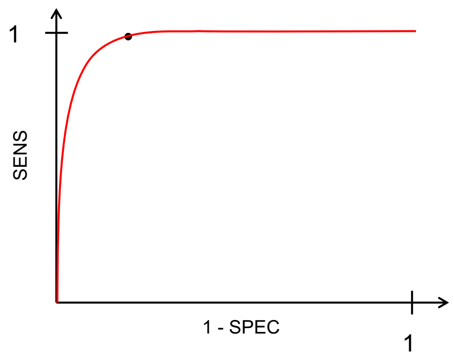
>> Accuracy = 0.8979 , sensitivity = 0.98949, specificity = 0.82425

Interpretation: Sensitivity is high so many skin pixels correctly identified. However, specificity is lower, due primarily to the false positives. Overall, roughly 90% of the pixels are correctly labelled.

89

ROC curve

- A Receiver Operating Characteristic curve plots the sensitivity as a function of 1 – specificity.
- In our previous example, SENS = 0.99, 1 – SPEC = 0.18
- One can sweep a parameter that trades of sensitivity vs specificity to generate a curve. The area under the curve gives another way to characterise the quality of a classification method.



90

Confusion matrix

- The confusion matrix describes the number of predicted labels vs the ground truth. In this example, we have skin and non-skin pixels.

		Predicted	
		Skin	Non-skin
Actual	Skin	TP	FN
	Non-skin	FP	TN

- A perfect result would be a diagonal matrix (no FPs or FNs)

- The confusion matrix generalises to multiple labels

		Predicted		
		Label 1	Label 2	Label 3
	Label 1			
Actual	Label 2			
	Label 3			

91

Segmentation based on Region Growing

Region Growing labels pixels into segmentation labels based on:

- predefined criteria for growth
- starting from a set of seeds

- Important : growth or similarity criterion selected according to the problem under consideration

- Can be based on color, image statistics etc.

- Stop the region growing when no more pixels change label

https://www.youtube.com/watch?v=WJGcaSmVE0E

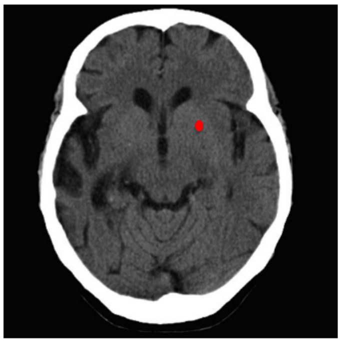
92

Region Growing

- Start with a single *seed pixel* p and expand from that.
- Define a similarity measure $S(i, j)$ for pixels i and j.
- Add adjacent pixel q to pixel p's region iff $S(p, q) > T$ for some threshold T.
- Proceed to the other neighbors of p and do likewise.
- We can now similarly consider the neighbors of q and add them likewise if they are similar enough.

93

Segmentation based on Region Growing



Region growing started with a seed point in the brain (e.g. the red dot shown)

For example, here adds neighboring points to the region if they are less than a threshold away from a running (re-calculated or updated every step) mean intensity

94

Similarity Measures

Can use intensity, texture, color (hue), etc.

- Compare to seed pixel
  Sensitive to noise, not transitive
- Compare to neighboring pixel
  Transitive, can drift easily
- Compare to region statistics
  Not transitive, doesn't drift as easily
- Multiple seed points
  Can get an idea not only of values but *variance*

from Bryan Morse, BYU

95

Region Growing

Design decisions:

1. How do we define similarity measure S?
2. What threshold T do we use? Does it change or stay constant?
3. If we wish to add q's neighbor r, do we use $S(p, r)$, $S(q, r)$, or something else?

Next: A list-based efficient processing in C++ for a fast implementation: recursive

96

Region Growing Algorithm ~ Connected Components

```
// Nice Simple Example C++ code for 3D segmentation:
// check it out; for questions: send an email to me

struct vector3Df {float x, y, z;
             };
//INITIALIZE YOUR LISTS
std::vector<vector3Df> *current;     //holds current region voxels
std::vector<vector3Df> *next;        //holds voxels to be analyzed on the next iteration
std::vector<vector3Df> *swap;        //pointer used in swapping the two lists
std::vector<vector3Df>::iterator it;
int Y = xSize;    int Z = xSize*ySize;       int x,y,z,i,j,k;
vector3Df  point3D;
point3D.x = seedX;  point3D.y = seedY;  point3D.z = seedZ; //  the seed location: a 3D vector

int OBJECT= 2;
int MARKED = 1;

next = new std::vector<vector3Df>;
current = new std::vector<vector3Df>;

next->clear();            //clear the two lists
current->clear();
//initialize the current list with the seed voxel
current->push_back(point3D);
```

97

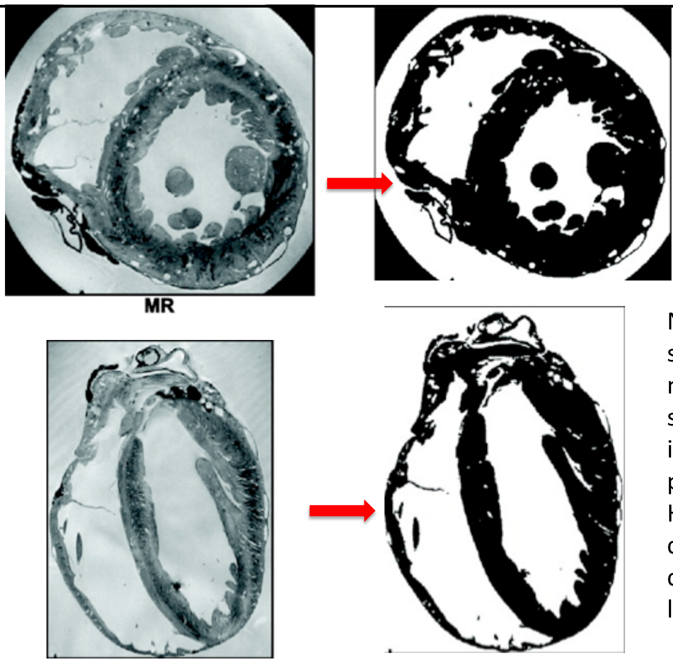
Region Growing Algorithm (cont'd)~ Connected Comp

```
while(!current->empty()) //while there are still current voxels
{for (it = current->begin(); it != current->end(); it++) { //for each current voxel
    x = (int)it->x;  y = (int)it->y;  z = (int)it->z;
    segMap[x+y*Y+z*Z] = OBJECT; //mark current voxel as object in the seg. map

    for(i=-1;i<=1;i++)for(j=-1;j<=1;j++)for(k=-1;k<=1;k++)
     { //******* for each nbr of current voxel
        // check if its in the volume
        if ( x+i<0 || x+i>=xSize || y+j<0 ||y+j>=ySize || z+k<0 || z+k>=zSize)  continue;

        //DESIGN:  Similarity Criterion
         if( (segMap[(x+i)+(y+j)*Y+(z+k)*Z] ~= OBJECT && (fabs(Image[x+y*Y+z*Z] -
Image[(x+i)+(y+j)*Y+(z+k)*Z]) < epsilon) ) {
          segMap[(x+i)+(y+j)*Y+(z+k)*Z] =  OBJECT;  //mark the nbr as object/connected
          point3D.x = x+i; point3D.y = y+j; point3D.z = z+k;   //add it to the "next" list
           next->push_back(point3D);          }
     } //*********end for each nbr of current voxel
   }//*******for each current voxel
  swap = current;    //swap "current" and "next" lists... old "next" is new "current"
  current = next;
   next = swap;    next->clear();
}// end while  //free allocated memory ;
delete next;  delete current;  ......
```

98



Next step after segmentation of multi-region scenes or objects is to label different parts separately. How? We can use connected component labeling

Figure from: "Development of an anatomically detailed MRI-derived rabbit ventricular model and assessment of its impact on simulations of electrophysiological function", American J Physiology, MJ Bishop et al 2010

99

Connected Component Labeling

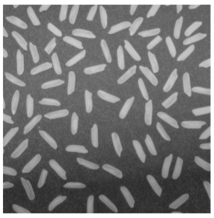
How to label (or color) separately connected segmented regions in an image ?

Binary Image

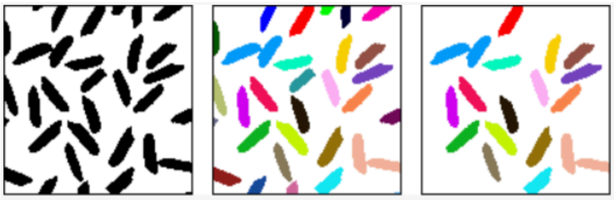
Connected Components

Compare to Semantic Segmentation

This is Instance Segmentation



First segment

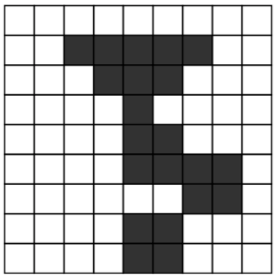


Example of area opening on a binary image. Left: original binary image. Middle: identification of connected components. Right: only the connected components with a sufficient size (defined by the area), have been retained.

http://imagej.net/MorphoLibJ

100

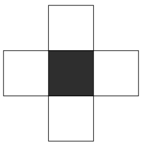
Pixel Connectivity



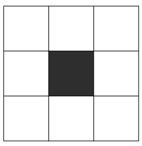
Q: are the dark pixels in this image connected?

To answer that: We need to define which pixels are counted as neighbors

Pixel Neighbourhoods



4-neighbourhood

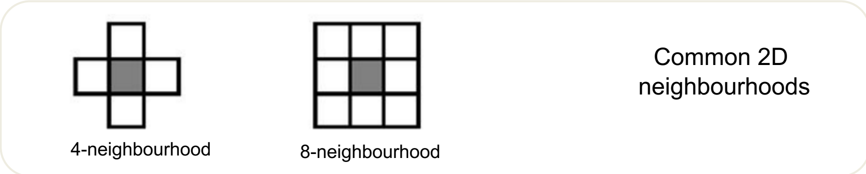


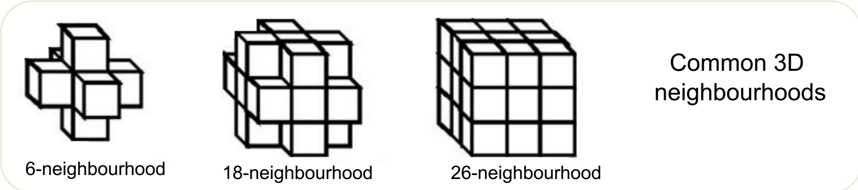
8-neighbourhood

101

Spatial Connectedness: going beyond color

- Colour (or intensity) is a powerful feature and typically used in image segmentation. However, typically there is additional contextual information we can use, often encoded *spatially*.
- When segmenting a pixel **p**, we might consider pixels in **p**'s neighbourhood.
- This require a definition what a neighbouring pixel is.





https://www.linkedin.com/pulse/20140506202304-14648565-3d-is-not-1-5-times-2d

102

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}, $i=1,\ldots,n-1$.

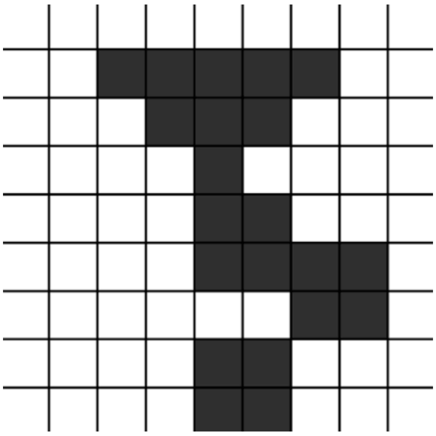
In an 8-connected path, p_i is an 8-neighbour of p_{i+1}.

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.

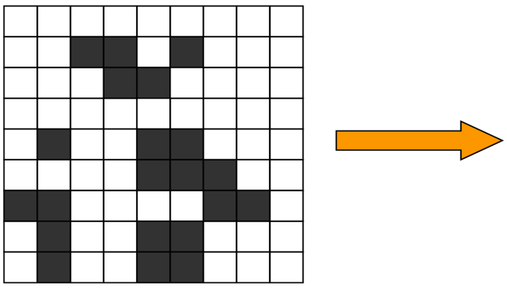
103

THQ: Connected regions: How many connected components for dark/light pixels? Using 4 or 8 connectivity ?



Connected Components Labeling

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

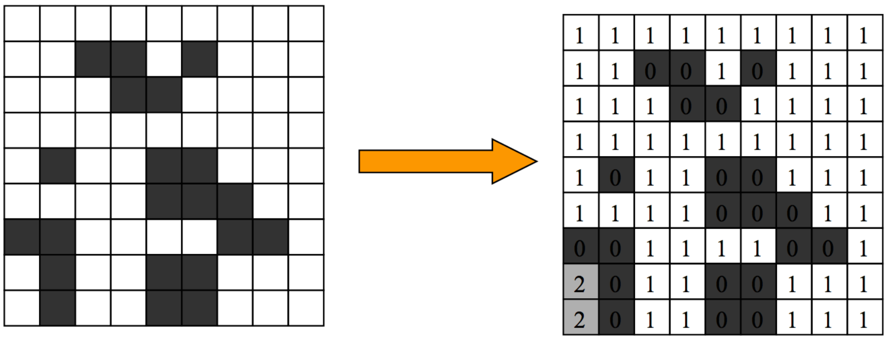


1	1	1	1	1	1	1	1	1
1	1	2	2	1	3	1	1	1
1	1	1	2	2	1	1	1	1
1	1	1	1	1	1	1	1	1
1	4	1	1	5	5	1	1	1
1	1	1	1	5	5	5	1	1
6	6	1	1	1	1	5	5	1
7	6	1	1	8	8	1	1	1
7	6	1	1	8	8	1	1	1

E.g. Use region growing starting from the top left corner of the image, set it to label L=1. Then find the first unlabelled pixel, set to L=2, use region growing, and continue in this way. Here, region growing algo, growing criteria: Neighbor pixel has the same label as the current pixel.

Foreground Connected Components Labeling

- Only extract the connected components of the foreground



Assuming that dark pixels are the background, others are foreground

106

Check this algorithm on connected component labeling

```
#B is the binary image input.
#L is the labeled image output
def ConnectedComponents(B):
    X,Y = B.shape
    L = np.zeros([X,Y])
    n=0
    for (y,x) in B:
        if B[y][x] and L[y][x]==0:
            label(x,y,n,B,L)
            n = n + 1
    return L
```

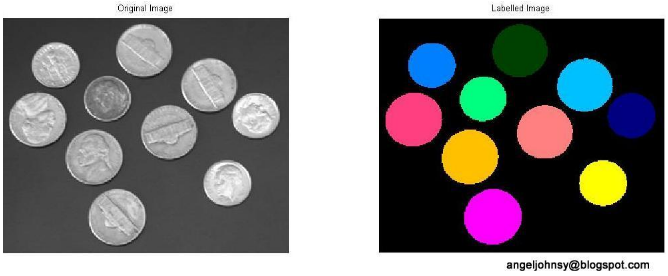
107

```python
# Recursively give label to this pixel
# and all its foreground neighbours.
def label(x_start,y_start,n,B,L):
    L[y_start][x_start]= n
    for (y,x) in N[y_start][x_start]:
        if L[y][x]==0 and B[y][x]:
            label(x,y,n,B,L)
```

* You need to define your neighborhood function N(.,.).

E.g. 4 nbhd: x,y should be one of: (xs+1,ys), (xs-1,ys), (xs,ys+1), (xs,ys-1)

108

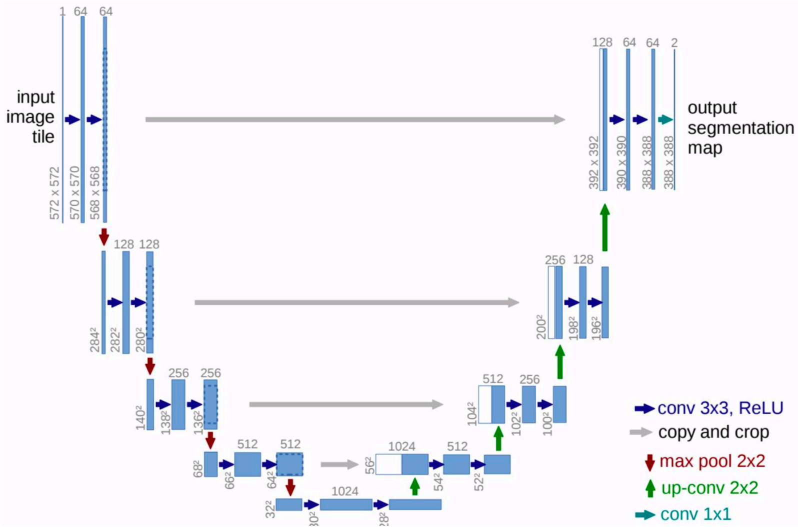


Example Result you would get from Connected Component Labeling

Think how you would perform the above operation : to go from the Original Image on the left to the Labeled image on the right.

109

Deep Neural Networks for Image Segmentation: For example



U-Net: Convolutional Neural Networks for Biomedical Image Segmentation, Ronnenberger et al, 2015.

There are many Semantic Segmentation Benchmarks: E.g. Cityscape, Pascal VOC, COCO

110

END OF LECTURE

Note that Segmentation has a vast literature, there are many more methods currently used in Computer Vision.
We just studied the basic methods, but they are important in understanding the idea of segmentation and data clustering.

Recall Learning objectives of the week: Students are able to:

3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

Reading Assignments:
[Klette Book]

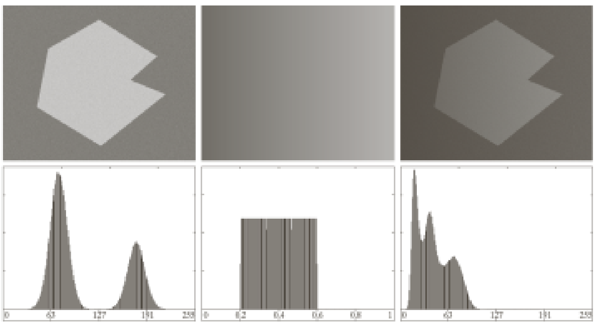
[Gonzalez and Woods] 10.3 : very limited
R. Szeliski Comp Vision Book: Section 5.3

111

Extra Material for those interested

112

Effect of Intensity Inhomogeneity in the Background



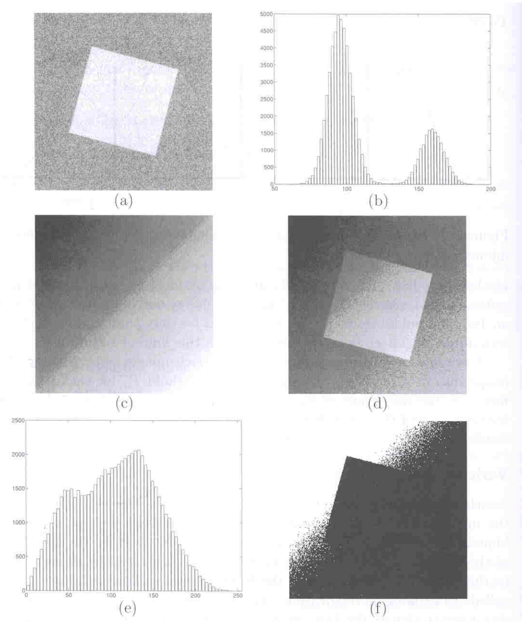
a b c
d e f

FIGURE 10.37 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.

Iterative threshold won't work!

113

Basic segmentation with e.g. Iterative thresholding won't work when there is background intensity variation in the image

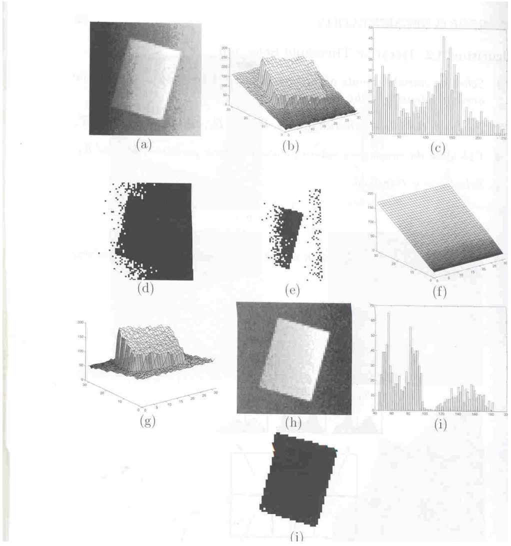


114

Background Normalization for better Thresholding

- Approximate the intensity values of the image by as simple function such as a plane.
- The function fit is determined in large part by the gray value of the background.
- Hence the name Background normalization: one obtains estimate of the variable background.
- Correct the image by subtracting the background estimate
- Now one can threshold the image appropriately

115



116

Background Light Intensity Problem in Historic Documents

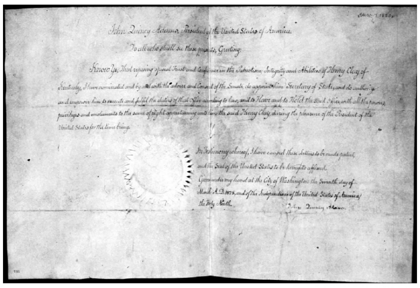


Figure 2: Historical handwritten document image with uneven background.

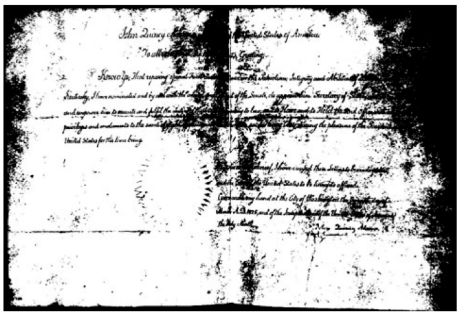


Figure 7: It is impossible for the original document image in Figure 2 to be segmented using a global threshold. The best global threshold we could manually find is at 200 and the binarized image shows significant parts of text obliterated.

"Historical Handwritten Document Image Segmentation Using Background Light Intensity Normalization" by Zhixin Shi and Venu Govindaraju

117

Background Intensity Normalization for historic documents

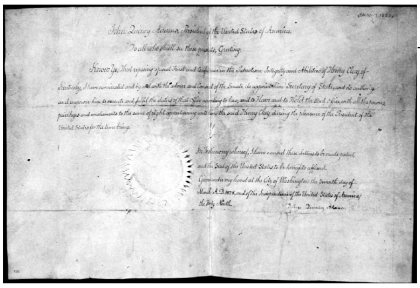


Figure 2: Historical handwritten document image with uneven background.

Threshold to find a mainly background image

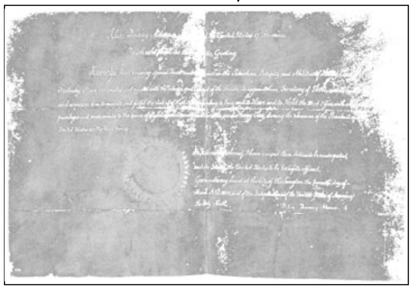


Figure 3: Pixels below the thresholding plane cover most of the background.

Plane fit to background image

"Historical Handwritten Document Image Segmentation Using Background Light Intensity Normalization" by Zhixin Shi and Venu Govindaraju

118

Background Intensity Normalization

Let us treat an image function as a 3D object (x,y,z): i.e. as a graph of (x,y) pixel coordinates, and z=I(x,y) image gray value.

Goal: find a plane H with an orientation over the x-y plane using the pixel coordinates (x_i,y_i) and their intensity values (z_i) coming mainly from the background (x_i,y_i,z_i) (e.g. One can do a simple tresholding to select mainly bg)

Using a plane equation: $$Ax + By - z + d = 0$$

We can minimize: $$\min \sum_i (Ax_i + By_i - z_i + d)^2$$

where the sum is for all the available points in the background image (thresholding result)

119

Background Intensity Normalization

"The best fit" linear plane is found (Note: one can also partition the image into smaller regions and find different planes for each sub-region)

That is, the parameters A, B, d.

Now the approximate background image value for each pixel coordinate (x_i,y_i) is:

$$z(x_i, y_i) = Ax_i + By_i + d$$

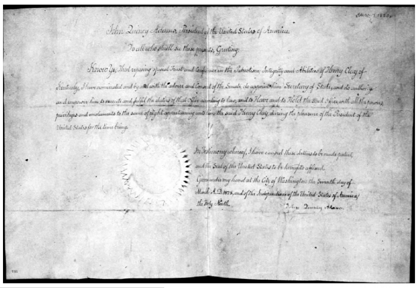
Background subtracted image can be found by:

$$\mathrm{I}_{\mathrm{corrected}}(\mathrm{x}, \mathrm{y}) = \mathrm{I}_{\mathrm{orig}}(\mathrm{x}, \mathrm{y}) - \mathrm{z}(\mathrm{x}, \mathrm{y}) + \mathrm{const}$$

where one can appropriately scale the image (e.g. Either add a const such as 255 and normalize the estimated z values to (0,255).

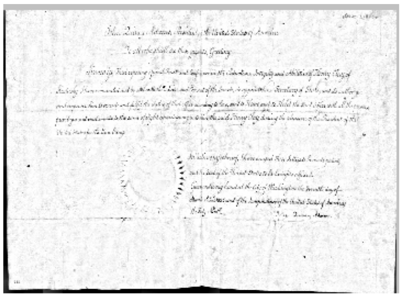
120

Background Intensity Normalization for historic documents



$\mathrm{I}_{\mathrm{orig}}$

$\mathrm{I}_{\mathrm{corrected}}$



Now binarized with a global threshold

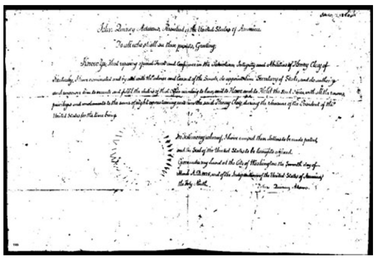
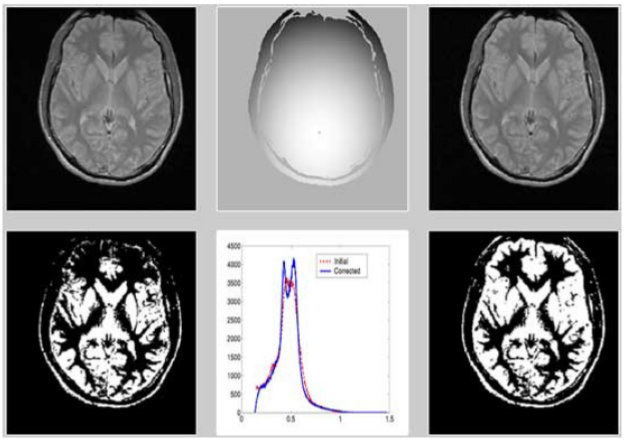


Figure 5: Normalized historical document image showing an even background.

Historical Handwritten Document Image Segmentation Using Background Light Intensity Normalization, Zhixin Shi and Venu Govindaraju

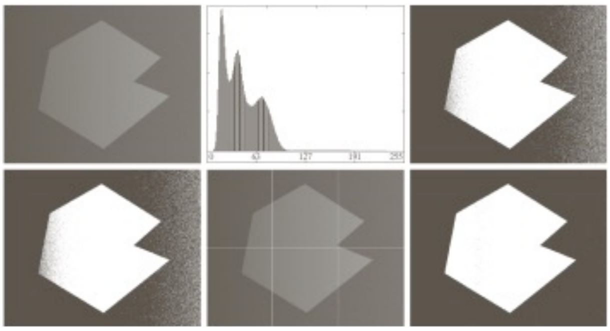
121

Bias Field in MR



Locally varying intensities make thresholding challenging. Image: University of Utah.

122



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

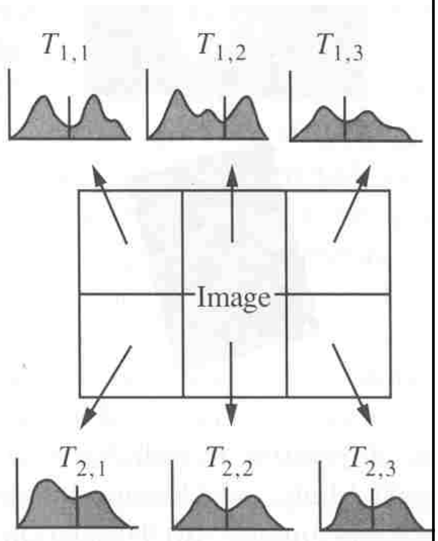
123

Variable Thresholding (Adaptive Thresholding)

Threshold value varies over the image as function of local image characteristics

- Image f is divided into sub-images f_c.
- A threshold is determined independently in each sub-image.
- If a threshold cannot be determined in some sub-image, it can be interpolated from thresholds determined in neighboring sub-images
- Each sub-image is then processed with respect to its local threshold.

$T=T(f, f_c)$



124