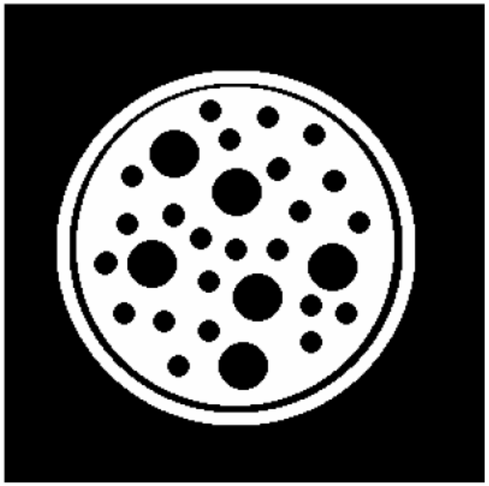
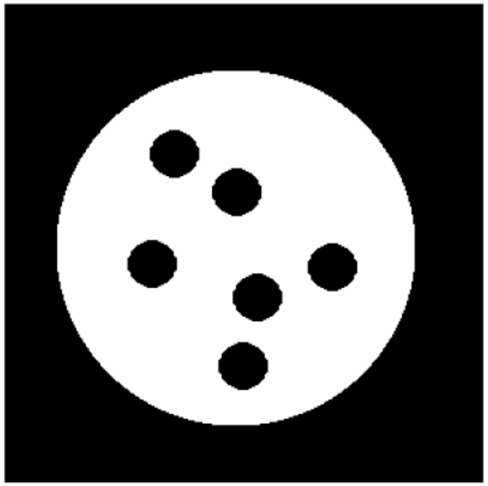
Edges and Binary Image Analysis



Notes Adapted from
Prof. Kristen Grauman



Previously

- Filters allow local image neighborhood to influence our description and features
  - Smoothing to reduce noise
  - Derivatives to locate contrast, gradient

Today

- Edge detection and matching
  - process the image gradient to find curves/contours
  - comparing contours

- Binary image analysis
  - blobs and regions

Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

Figure from D. Lowe

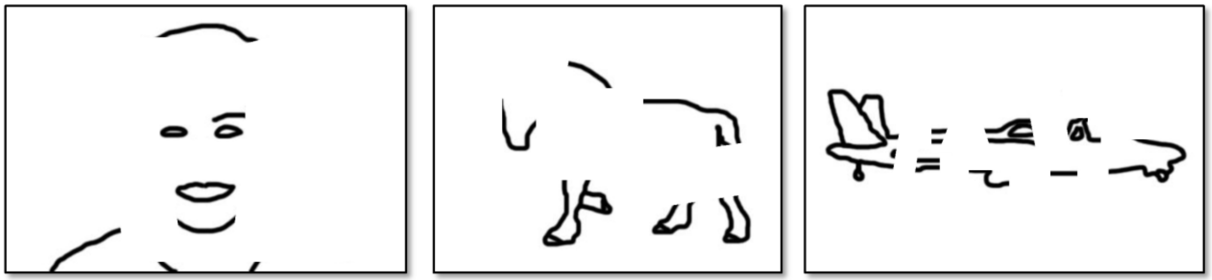


Figure from J. Shotton et al., PAMI 2007

- **Main idea**: look for strong **gradients**, post-process

Gradients -> edges

Primary edge detection steps:

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

   Determine which local maxima from filter output are actually edges vs. noise

   - Threshold, Thin

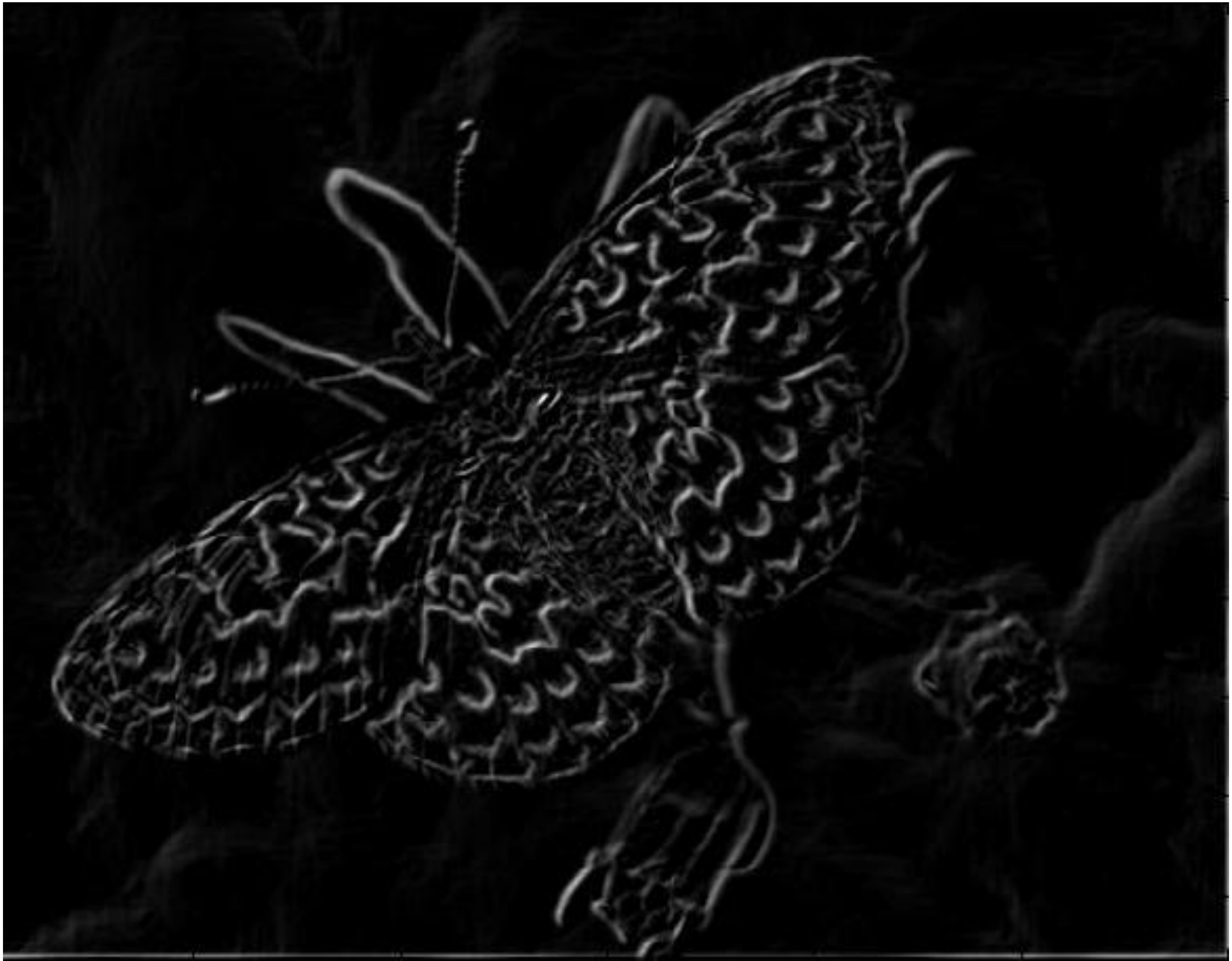
Kristen Grauman, UT-Austin

Thresholding

- Choose a threshold value t
- Set any pixels less than t to zero (off)
- Set any pixels greater than or equal to t to one (on)

Original image

Gradient magnitude image



Thresholding gradient with a lower threshold

Thresholding gradient with a higher threshold

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin wide “ridges” down to single pixel width
- **Linking and thresholding** (**hysteresis**):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: **`edge(image, ‘canny’);`**
- **`>>help edge`**

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

original image (Lena)

Slide credit: Steve Seitz

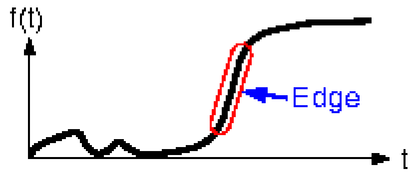
The Canny edge detector

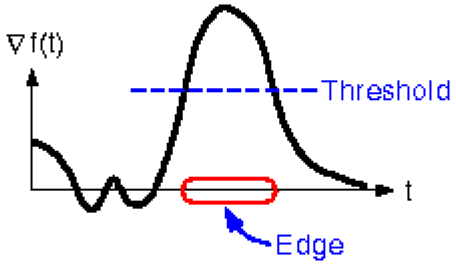
norm of the gradient

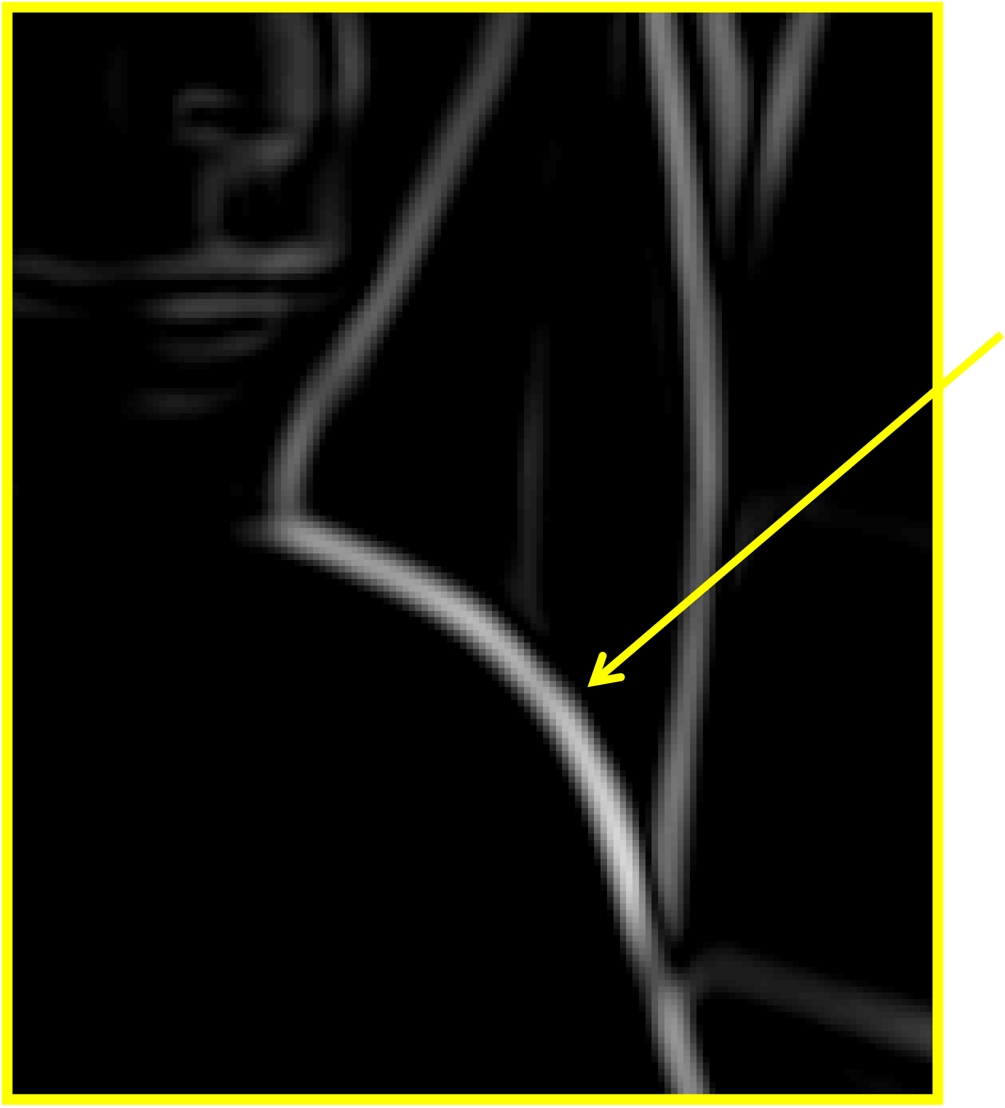
The Canny edge detector

thresholding

The Canny edge detector

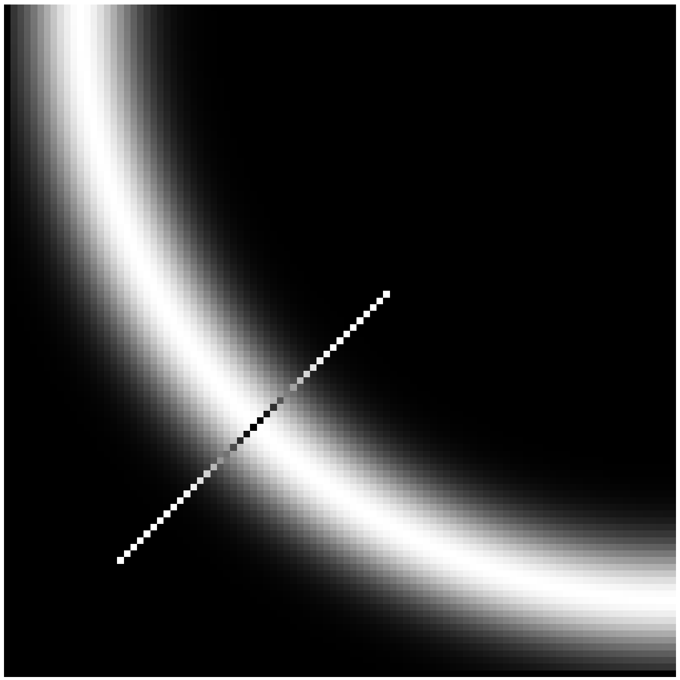


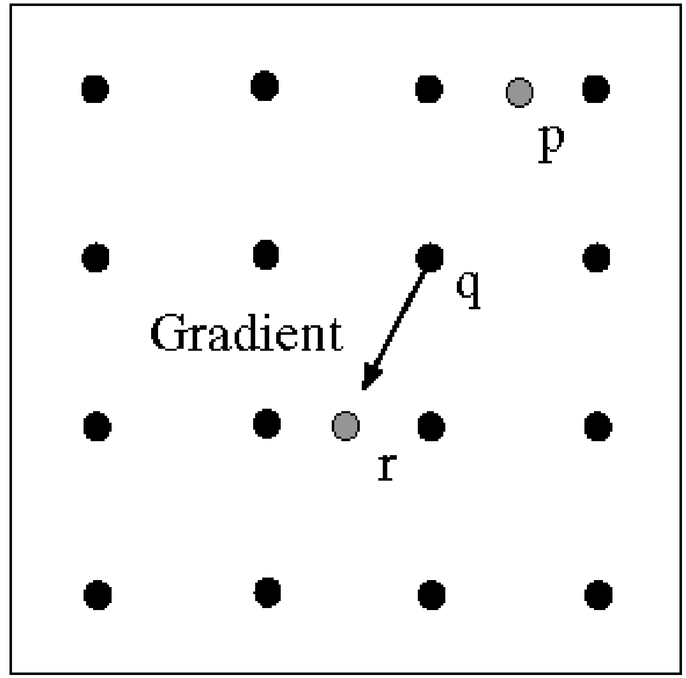




How to turn these thick regions of the gradient into curves?

Non-maximum suppression





Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

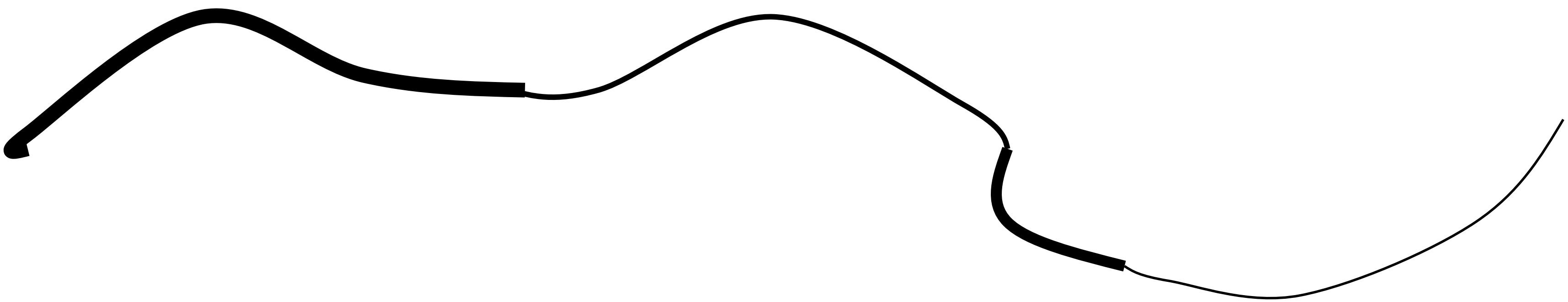
The Canny edge detector

Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them.



Source: Steve Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

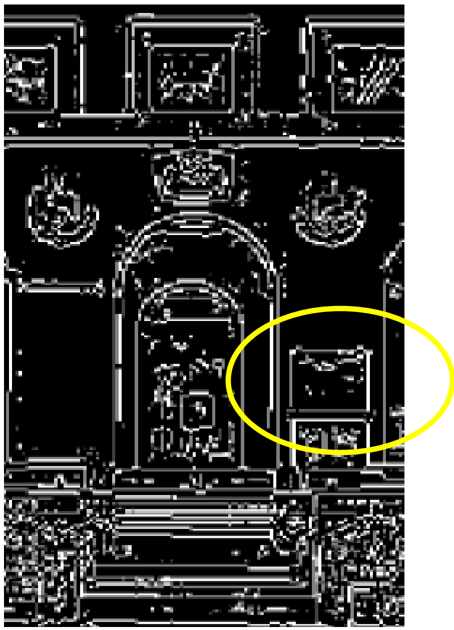
low threshold (weak edges)

hysteresis threshold

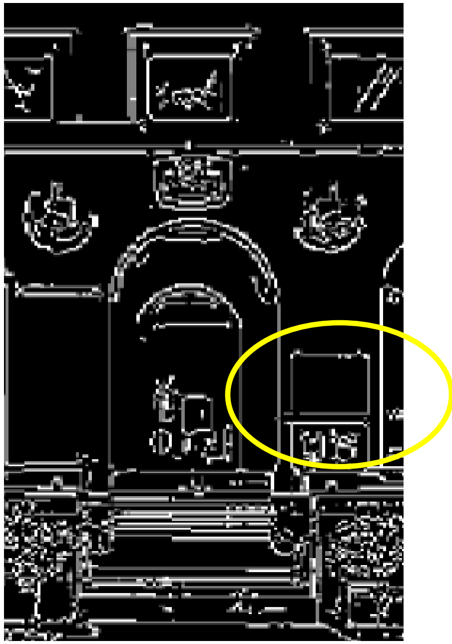
Source: L. Fei-Fei

Hysteresis thresholding

high threshold (strong edges)



low threshold (weak edges)



hysteresis threshold

Source: L. Fei-Fei

Recap: Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin wide “ridges” down to single pixel width
- **Linking and thresholding** (**hysteresis**):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

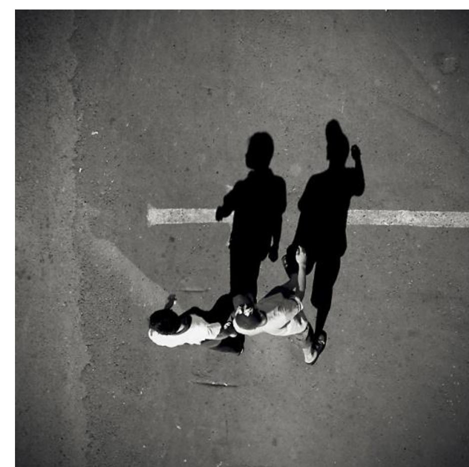
- MATLAB: **`edge(image, ‘canny’);`**
- **`>>help edge`**

Source: D. Lowe, L. Fei-Fei

Low-level edges vs. perceived contours

Background

Texture



Shadows

Kristen Grauman, UT-Austin

Low-level edges vs. perceived contours

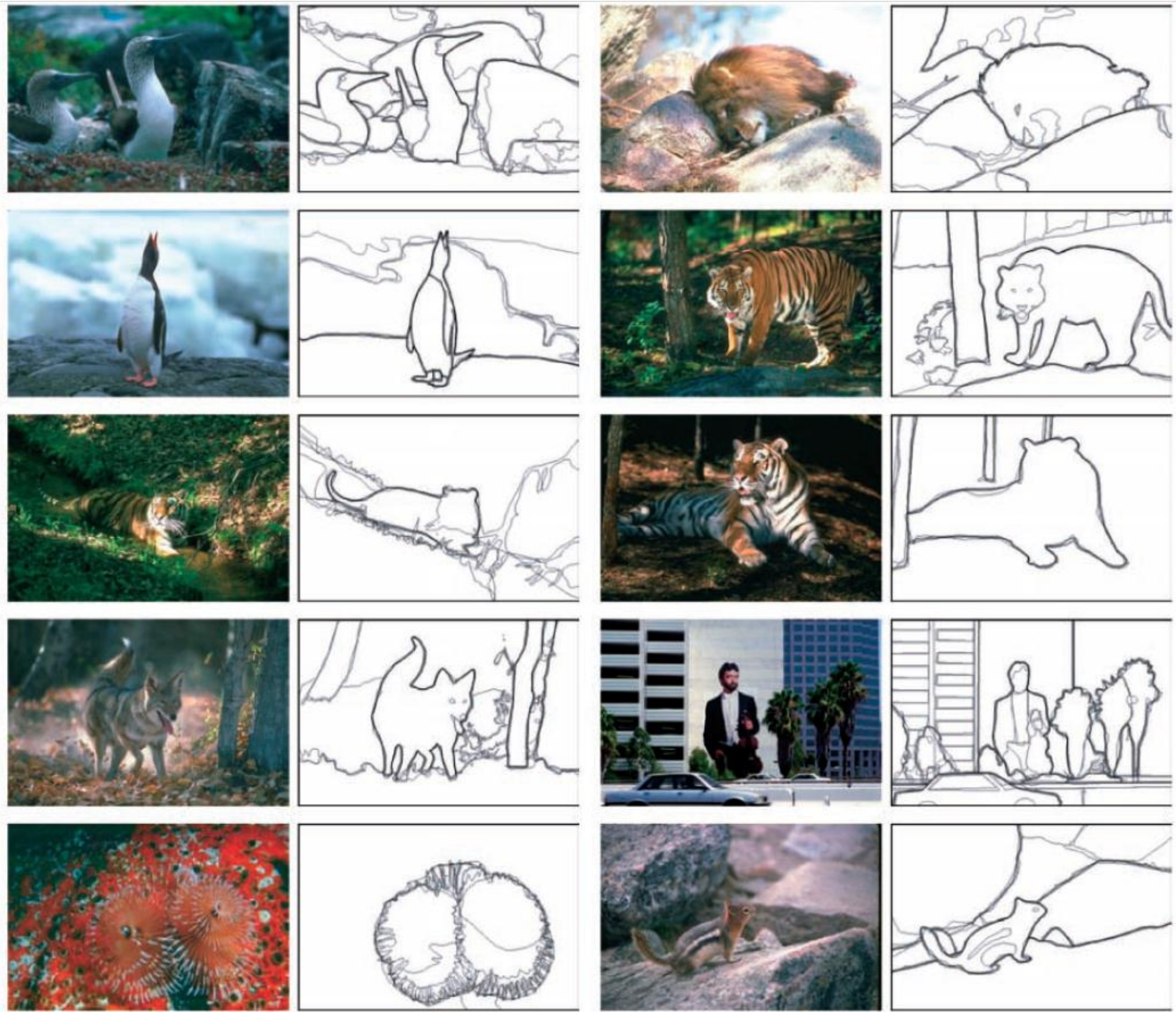
image **human segmentation** **gradient magnitude**

Berkeley segmentation database:

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

Source: L. Lazebnik

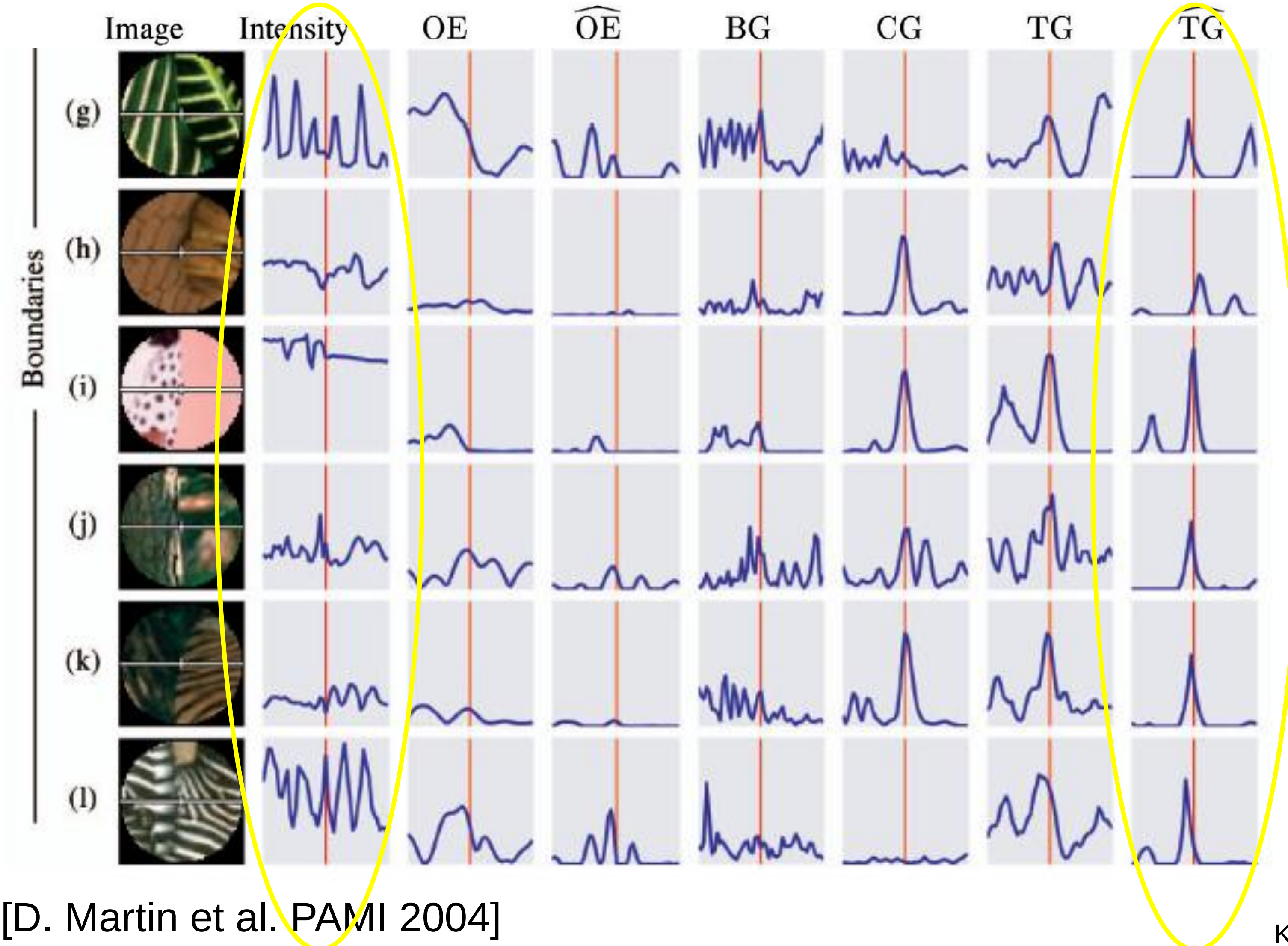
Learn from humans which combination of features is most indicative of a “good” contour?



[D. Martin et al. PAMI 2004]

Human-marked segment boundaries

What features are responsible for perceived edges?

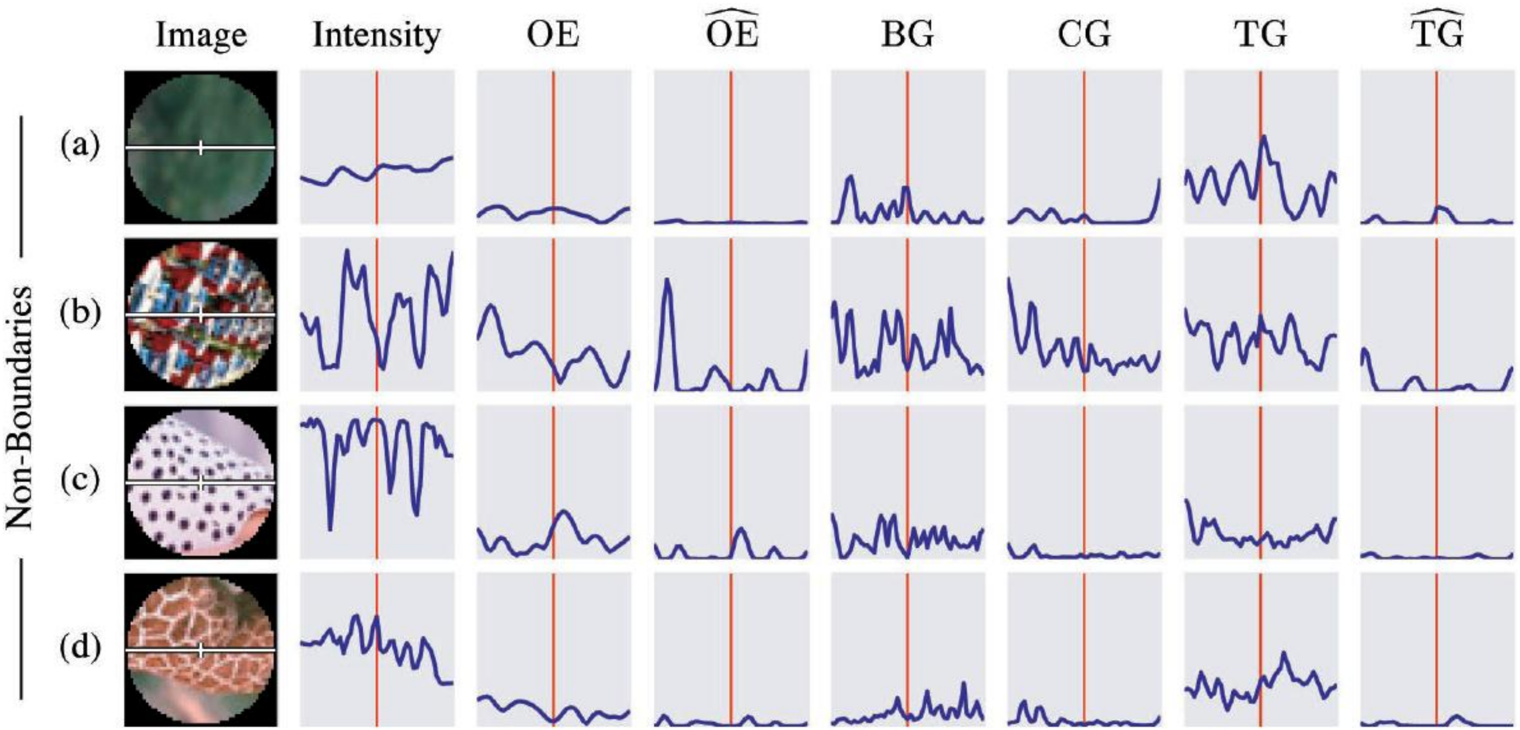


Feature profiles (oriented energy, brightness, color, and texture gradients) along the patch's horizontal diameter

[D. Martin et al. PAMI 2004]

Kristen Grauman, UT-Austin

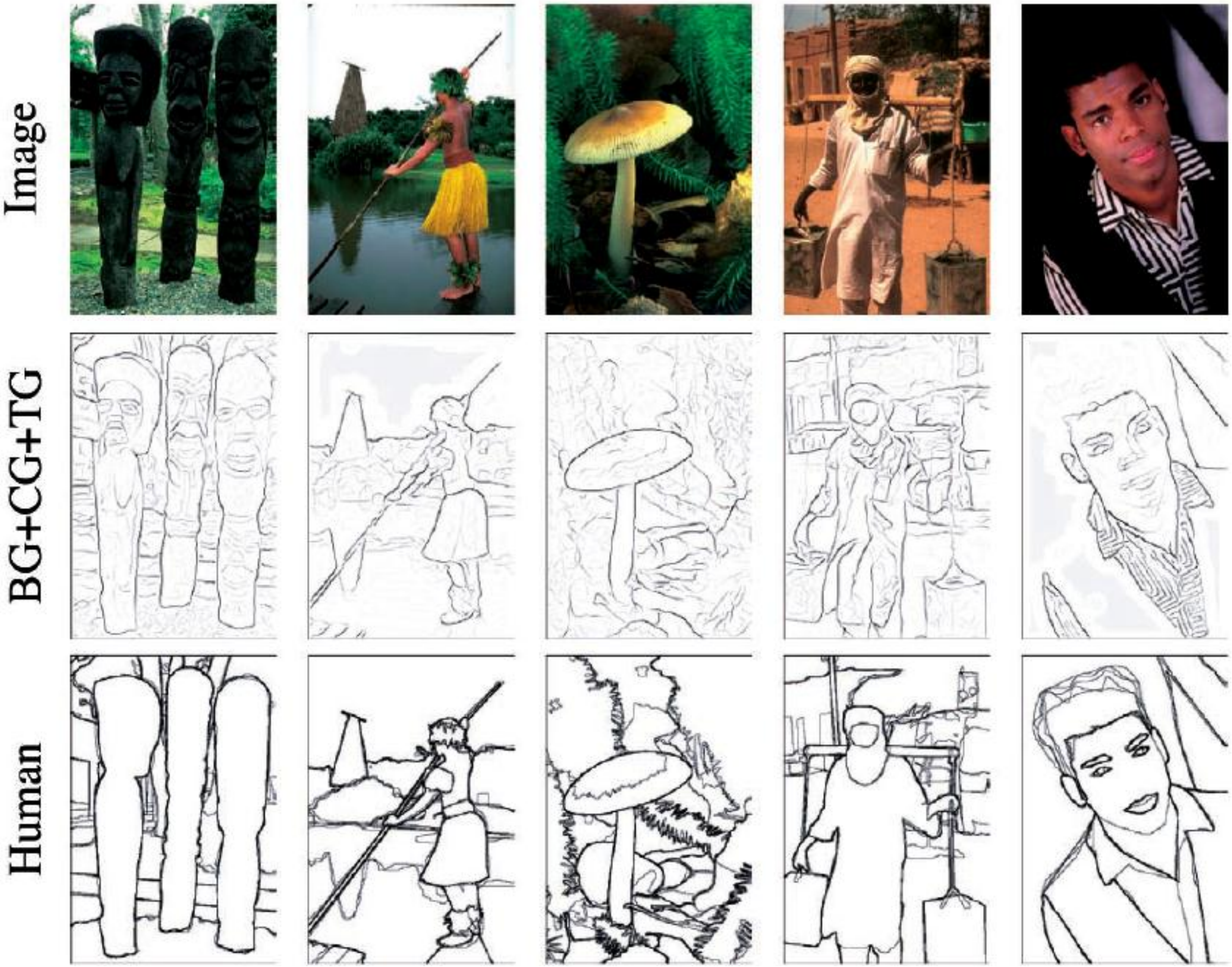
What features are responsible for perceived edges?



Feature profiles (oriented energy, brightness, color, and texture gradients) along the patch's horizontal diameter

[D. Martin et al. PAMI 2004]

Kristen Grauman, UT-Austin



[D. Martin et al. PAMI 2004]

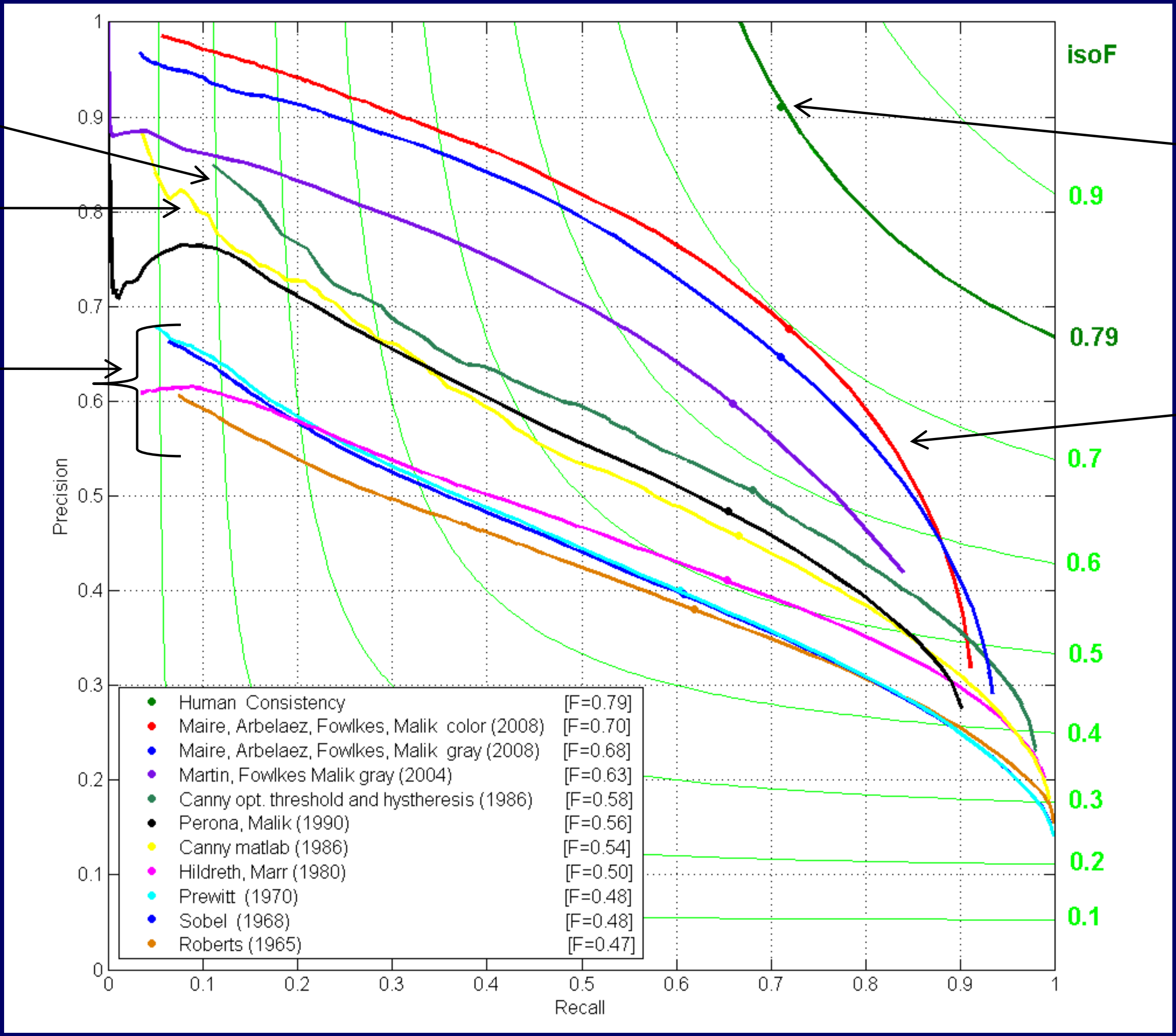
Kristen Grauman, UT-Austin

State-of-the-Art in Contour Detection

Canny+opt thresholds

Canny

Prewitt, Sobel, Roberts



Human agreement

Learned with combined features

Today

- Edge detection and matching
  - process the image gradient to find curves/contours
  - comparing contours
- Binary image analysis
  - blobs and regions

Fig. 1. Examples of two handwritten digits. In terms of pixel-to-pixel comparisons, these two images are quite different, but to the human observer, the shapes appear to be similar.

Figure from Belongie et al.

Chamfer distance

- Average distance to nearest feature

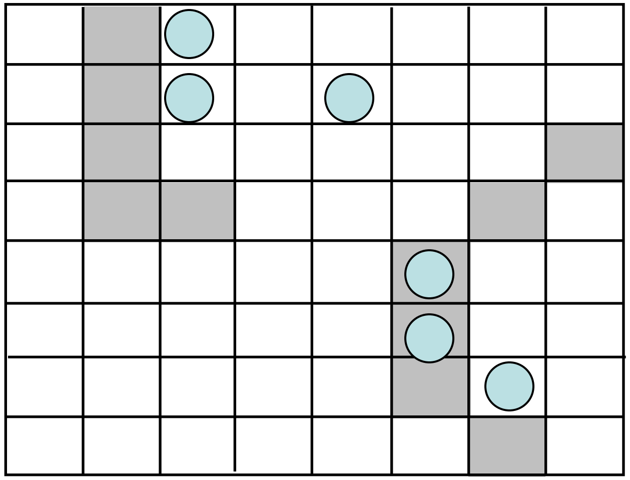
$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

$I =$ Set of points in image

$T =$ Set of points on (shifted) template

$d_I(t) =$ Minimum distance between point t and some point in I

Chamfer distance



$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

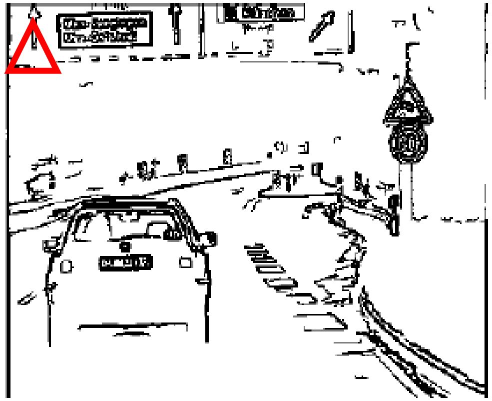
Chamfer distance

- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

How is the measure different than just filtering with a mask having the shape points?

How expensive is a naïve implementation?

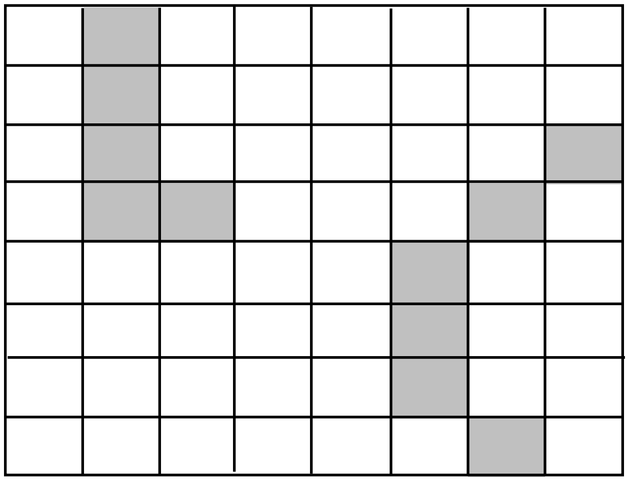


Edge image

Kristen Grauman, UT-Austin

Distance transform

Image features (2D)



Distance Transform

1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

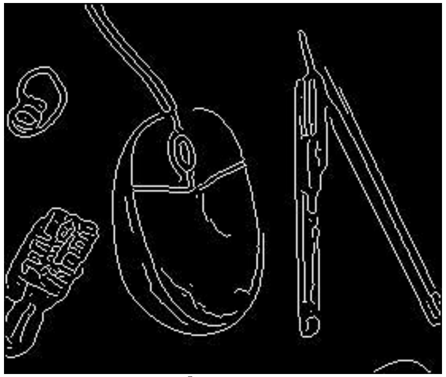
Distance Transform is a function $D(\cdot)$ that for each image pixel p assigns a non-negative number $D(p)$ corresponding to distance from p to the nearest feature in the image I

Features could be edge points, foreground points,…

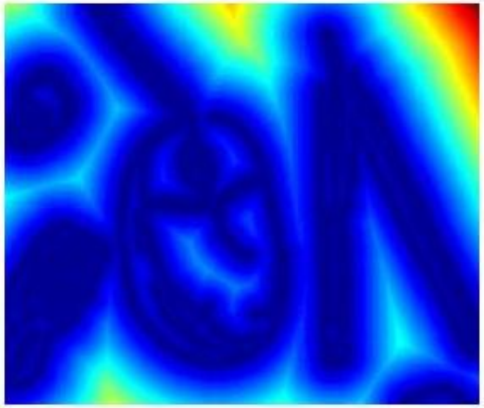
Source: Yuri Boykov

Distance transform

original



edges



distance transform

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

```
>> help bwdist
```

Distance transform (1D)

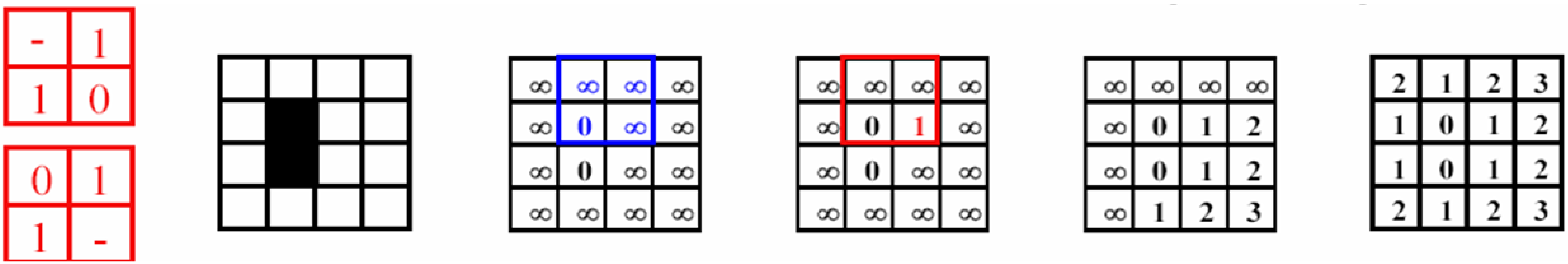
Two pass O(n) algorithm for 1D L_1 norm

1. <u>Initialize</u>: For all j
   $D[j] \leftarrow 1_{\mathbf{P}}[j]$ // 0 if j is in **P**, infinity otherwise

Adapted from D. Huttenlocher

Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right

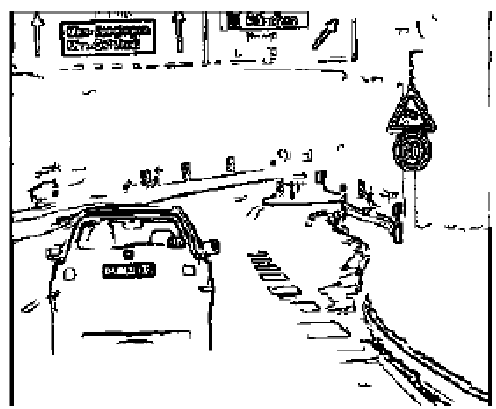


Adapted from D. Huttenlocher

Chamfer distance

- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

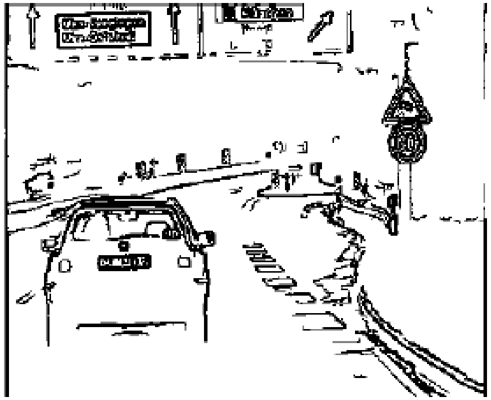


Edge image

Distance transform image

Kristen Grauman, UT-Austin

Chamfer distance



Edge image

Distance transform image

Fig from D. Gavrila, DAGM 1999

Chamfer distance: properties

- Sensitive to scale and rotation
- Tolerant of small shape changes, clutter
- Need large number of template shapes
- Inexpensive way to match shapes

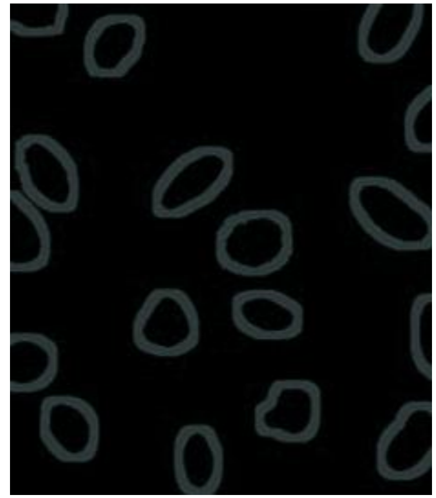
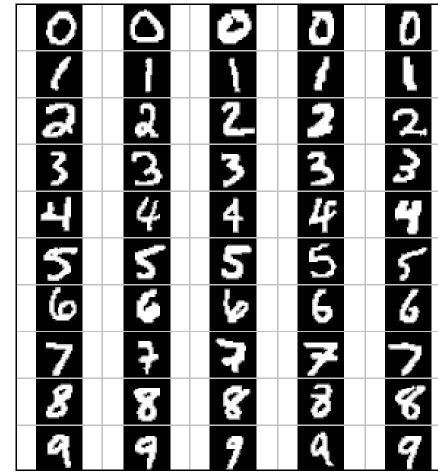
Today

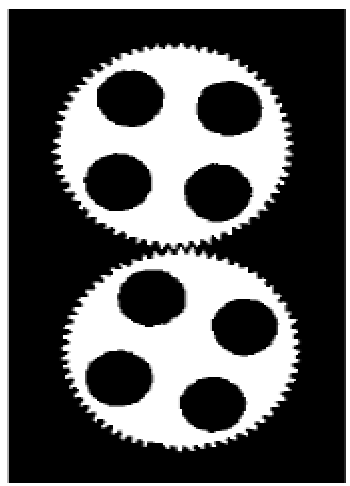
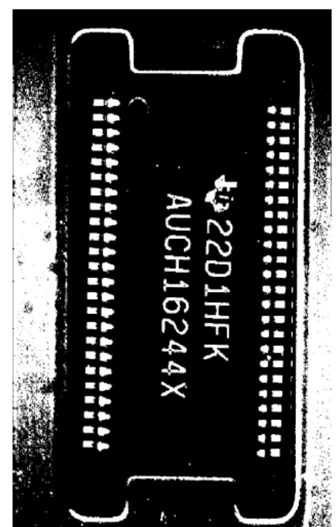
- Edge detection and matching
  - process the image gradient to find curves/contours
  - comparing contours

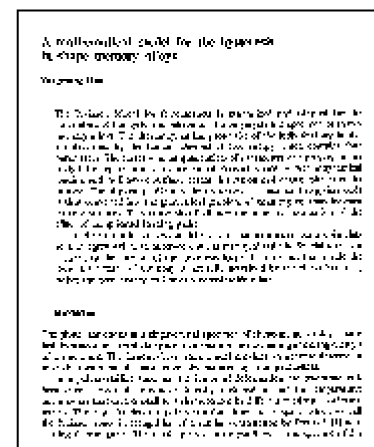
- Binary image analysis
  - blobs and regions

Binary images





Kristen Grauman, UT-Austin

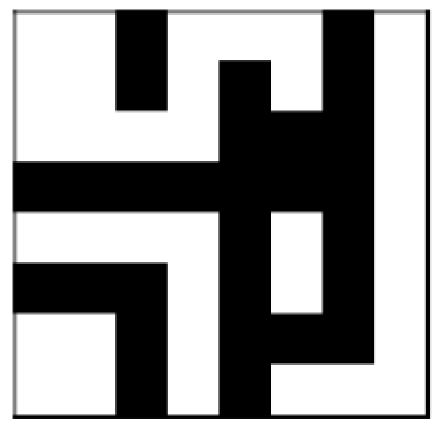
Binary image analysis: basic steps

- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract separate blobs
  - Connected components
- Describe the blobs with region properties

Binary images

- Two pixel values
  - Foreground and background
  - Mark region(s) of interest

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



Thresholding

- Grayscale -> binary mask
- Useful if object of interest’s intensity distribution is distinct from background

$$F_T[i,j] = \begin{cases} 1 & \textbf{if} \ \ F[i,j] \geq T \\ 0 & otherwise. \end{cases}$$

$$F_T[i,j] = \begin{cases} 1 & \textbf{if} \ \ T_1 \leq F[i,j] \leq T_2 \\ 0 & otherwise. \end{cases}$$

$$F_T[i,j] = \begin{cases} 1 & \textbf{if} \ \ F[i,j] \in Z \\ 0 & otherwise. \end{cases}$$

- [Example](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FITZGIBBON/simplebinary.html) http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FITZGIBBON/simplebinary.html

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: edge detection

Gradient magnitude

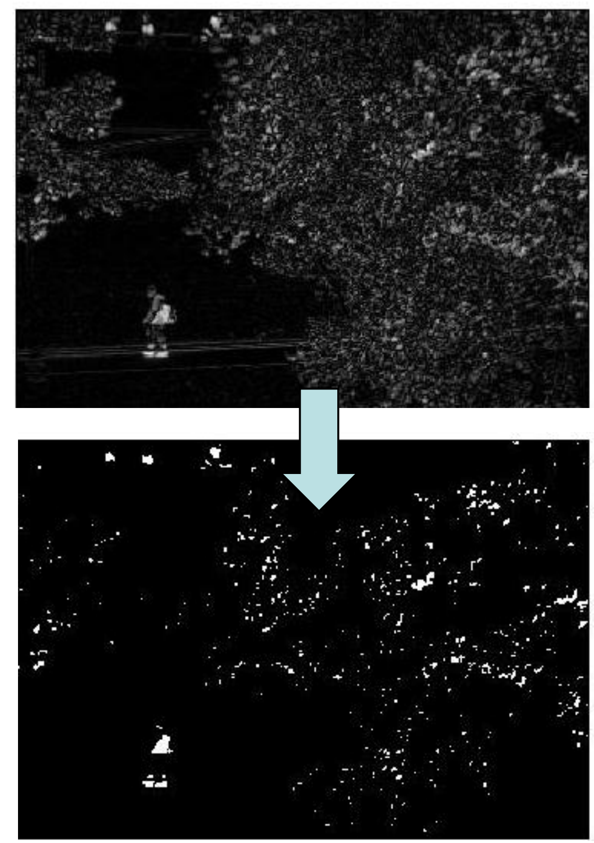
`fg_pix = find(gradient_mag > t);`

Looking for pixels where gradient is strong.

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: background subtraction



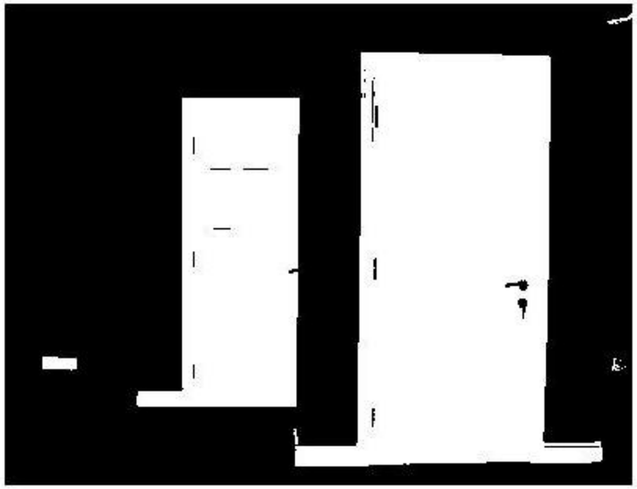
Looking for pixels that differ significantly from the “empty” background.

```
fg_pix = find(diff > t);
```

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

Example: intensity-based detection



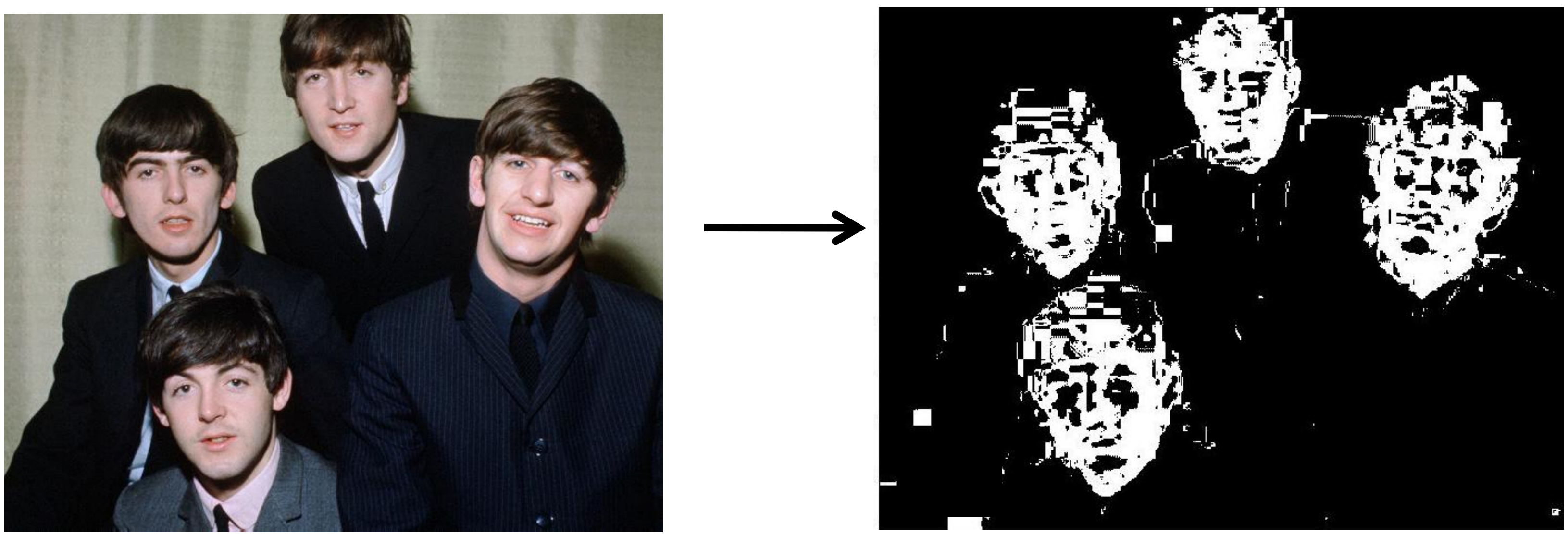
`fg_pix = find(im < 65);`

Looking for dark pixels

Thresholding

- Given a grayscale image or an intermediate matrix → threshold to create a binary output.

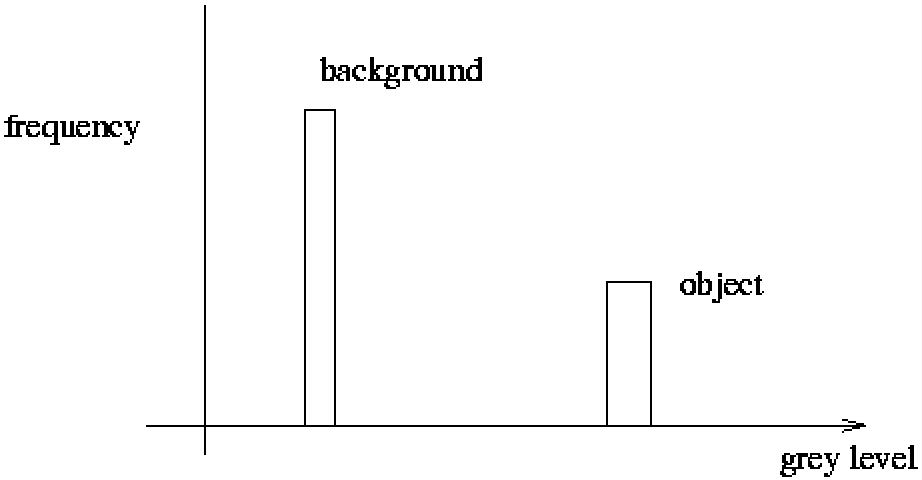
Example: color-based detection



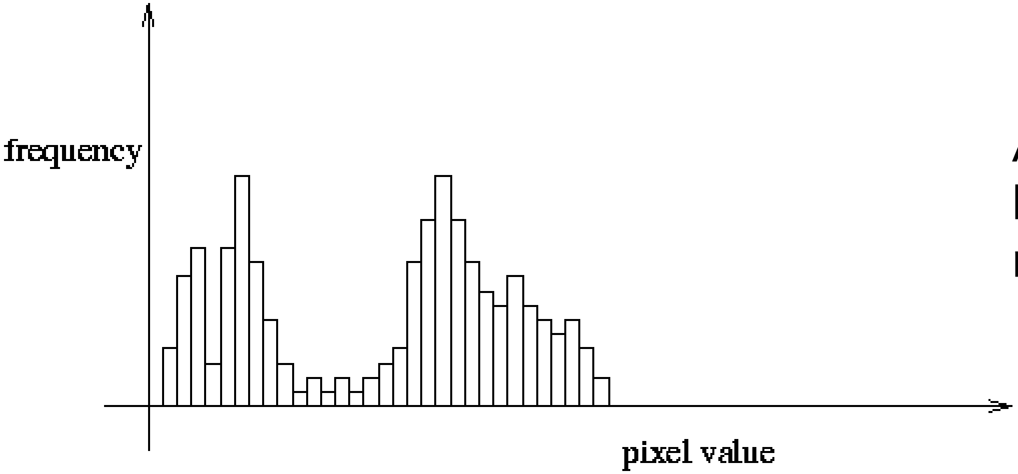
```
fg_pix = find(hue > t1 & hue < t2);
```

Looking for pixels within a certain hue range.

A nice case: bimodal intensity histograms



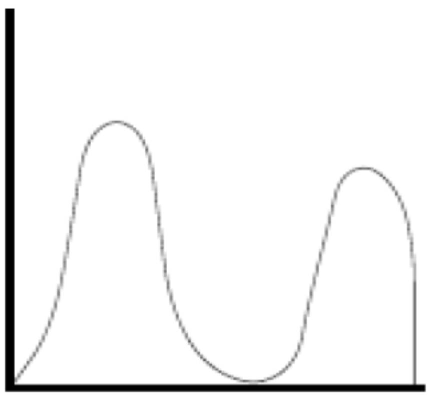
Ideal histogram, light object on dark background



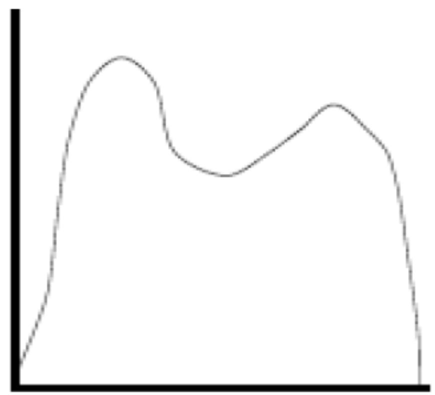
Actual observed histogram with noise

Images: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html

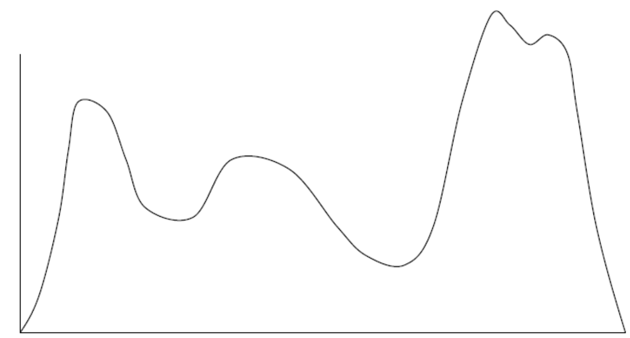
Not so nice cases



Two distinct modes



Overlapped modes

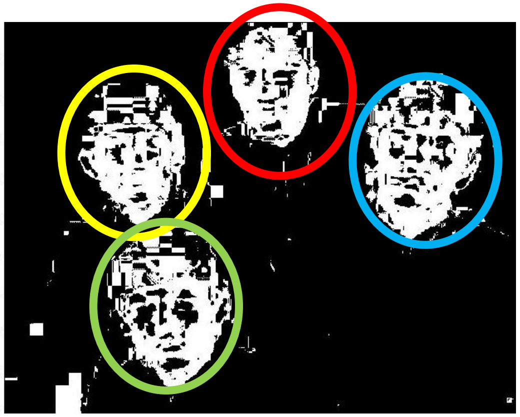


Shapiro and Stockman

Issues

- What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments

- How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object



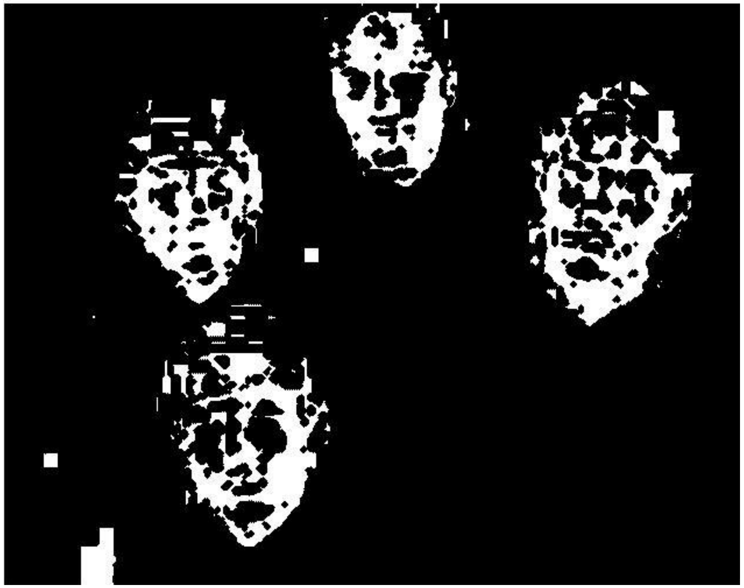
Kristen Grauman, UT-Austin

Morphological operators

- Change the shape of the foreground regions via intersection/union operations between a scanning structuring element and binary image.
- Useful to clean up result from thresholding
- Basic operators are:
  - Dilation
  - Erosion

Dilation

- Expands connected components
- Grow features
- Fill holes



Before dilation

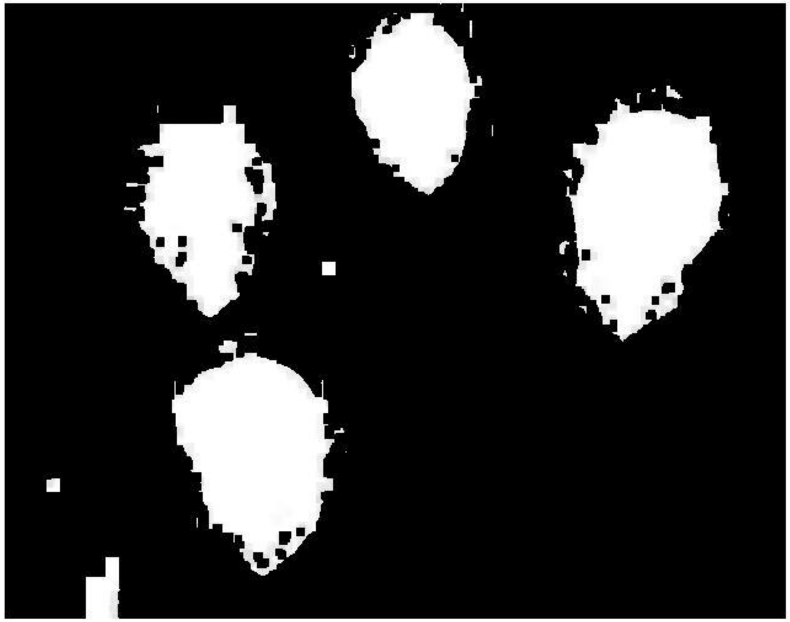
After dilation

Kristen Grauman, UT-Austin

Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise

Before erosion

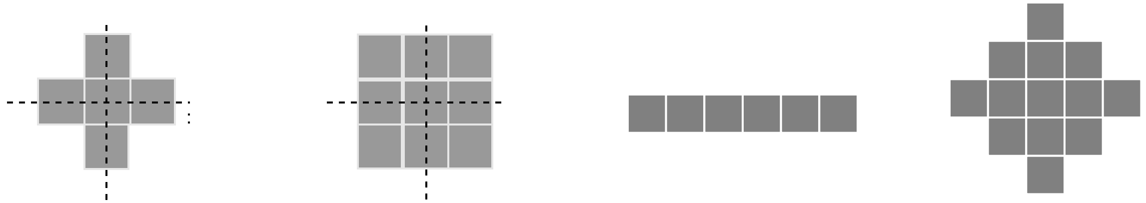


After erosion

Kristen Grauman, UT-Austin

Structuring elements

- **Masks** of varying shapes and sizes used to perform morphology, for example:



- Scan mask across foreground pixels to transform the binary image

```
>> help strel
```

Kristen Grauman, UT-Austin

Dilation vs. Erosion

At each position:

- **Dilation**: if current pixel is foreground, OR the structuring element with the input image.

Example for Dilation (1D)

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

$$g(x) = f(x) \oplus SE$$

Output Image

1	1								

Adapted from T. Moeslund

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1								

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0							

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	0						

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	1	1	1				

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	1	1	1	1			

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	1	1	1	1	1		

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	1	1	1	1	1		

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

1	1	0	1	1	1	1	1	1	1

Note that the object gets bigger and holes are filled.

```
>> help imdilate
```

2D example for dilation

1	1	1	1	1	1	1	
			1	1	1	1	
			1	1	1	1	
		1	1	1	1	1	
			1	1	1	1	
		1	1				

(*a*) Binary image **B**

1	1	1
1	1	1
1	1	1

(*b*) Structuring element **S**

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1	1	1	1
	1	1	1	1			

(*c*) Dilation $\mathbf{B} \oplus \mathbf{S}$

Shapiro & Stockman

Dilation vs. Erosion

At each position:

- **Dilation**: if **current pixel** is foreground, OR the structuring element with the input image.
- **Erosion**: if **every pixel** under the structuring element's nonzero entries is foreground, OR the current pixel with S.

Example for Erosion (1D)

Input image

1	0	0	0	1	1	1	0	1	1

1	**1**	**1**

Structuring Element

$$g(x) = f(x) \ominus SE$$

Output Image

0									

Example for Erosion (1D)

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

$$g(x) = f(x) \Theta SE$$

Output Image

0	0								

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0							

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0	0						

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0	0	0					

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0	0	0	1				

Example for Erosion

Input image	1	0	0	0	1	1	1	0	1	1
Structuring Element						**1**	**1**	**1**		
Output Image	0	0	0	0	0	1	0			

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0	0	0	1	0	0		

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

0	0	0	0	0	1	0	0	0	

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

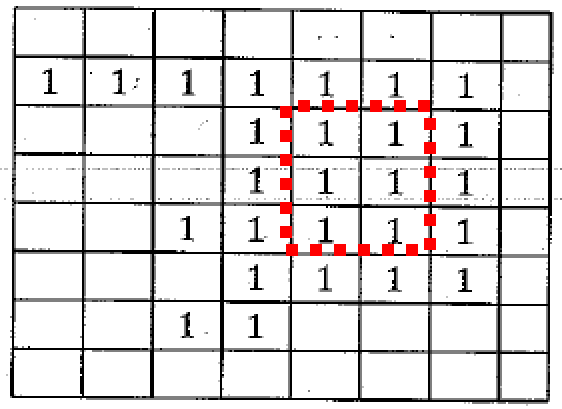
Output Image

0	0	0	0	0	1	0	0	0	1

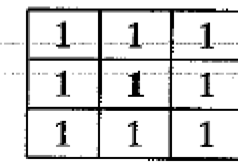
Note that the object gets smaller

```
>> help imerode
```

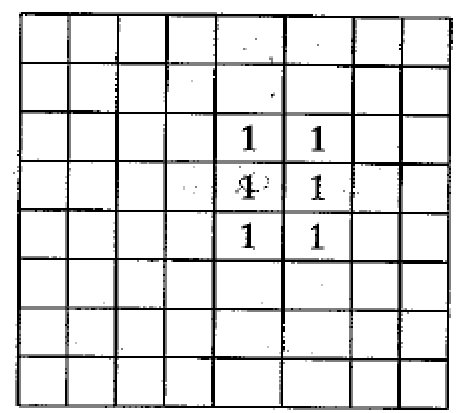
2D example for erosion



(*a*) Binary image **B**



(*b*) Structuring element **S**

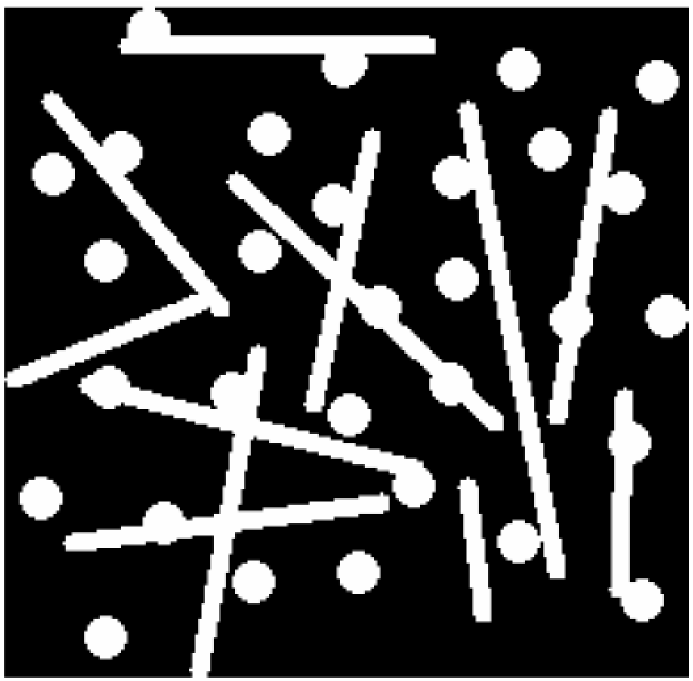


(*d*) Erosion $\mathbf{B} \ominus \mathbf{S}$

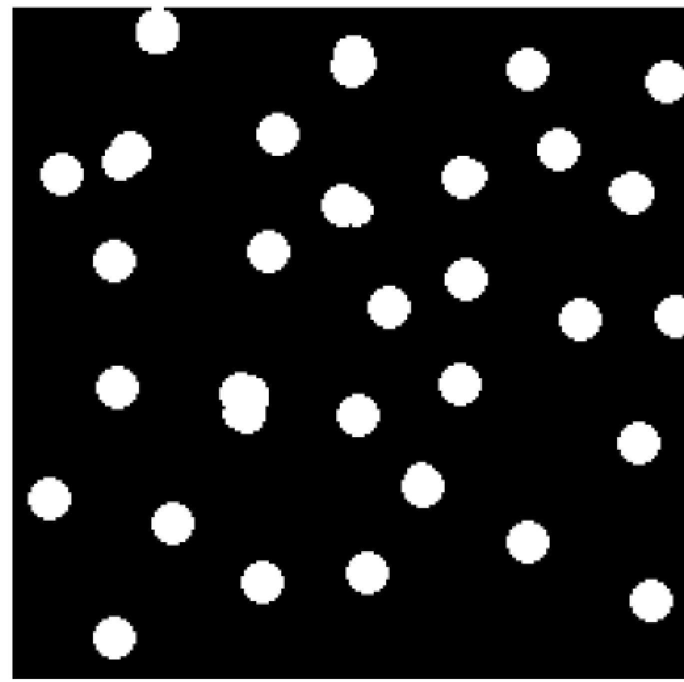
Shapiro & Stockman

Opening

- Erode, then dilate
- Remove small objects, keep original shape



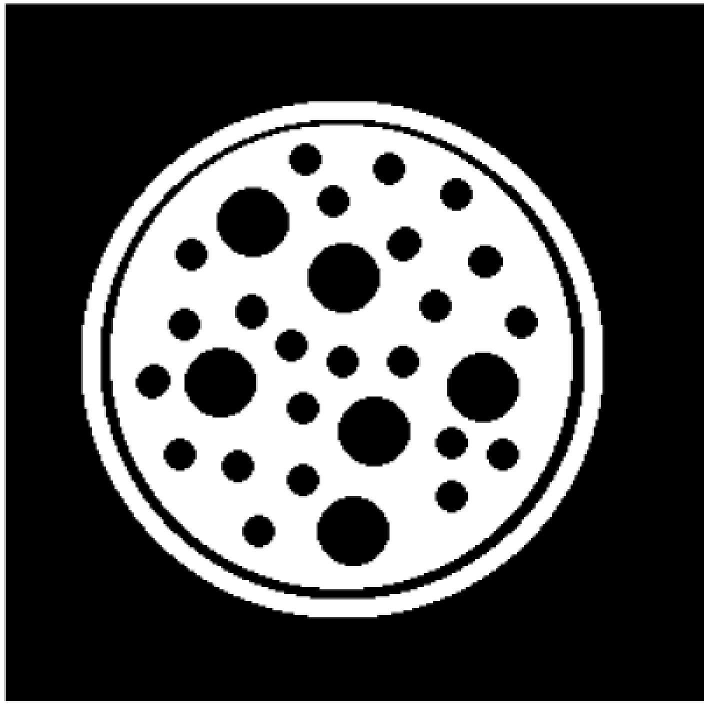
Before opening



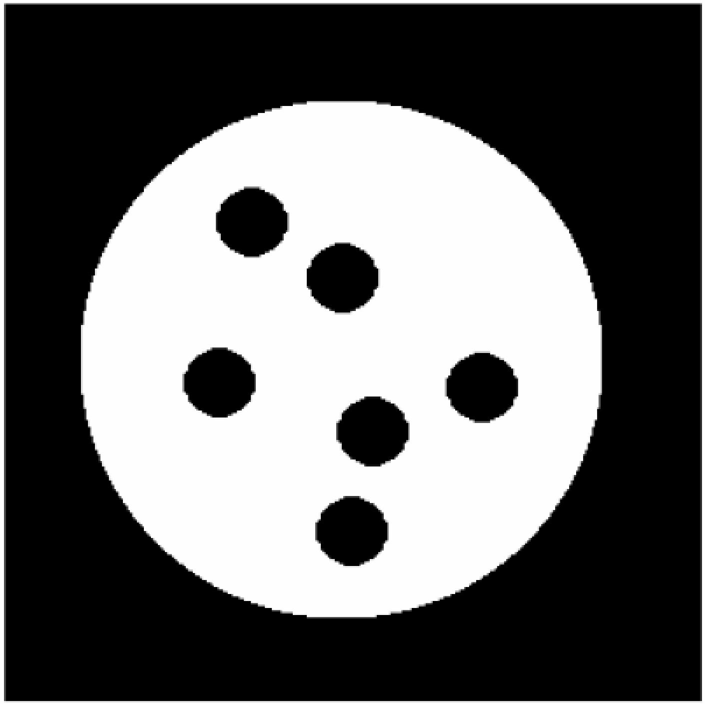
After opening

Closing

- Dilate, then erode
- Fill holes, but keep original shape



Before closing



After closing

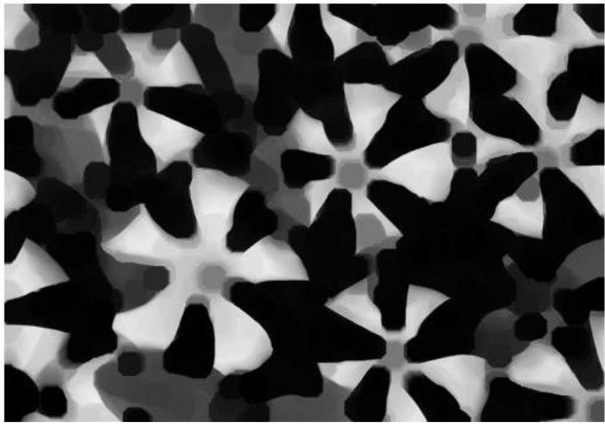
Applet: http://bigwww.epfl.ch/demo/jmorpho/start.php

Morphology operators on grayscale images

- Dilation and erosion typically performed on binary images.
- If image is grayscale: for dilation take the neighborhood **max**, for erosion take the **min**.

original

dilated



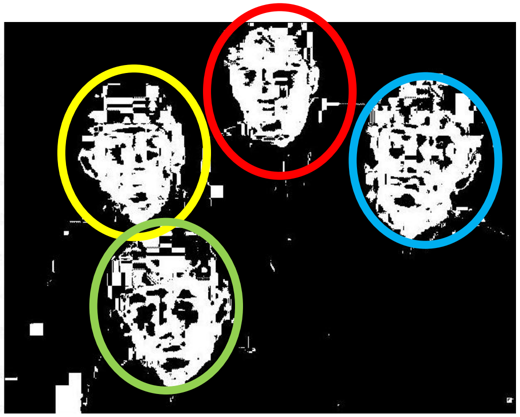
eroded

Kristen Grauman, UT-Austin

Issues

- What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments

- How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object



Kristen Grauman, UT-Austin

Connected components

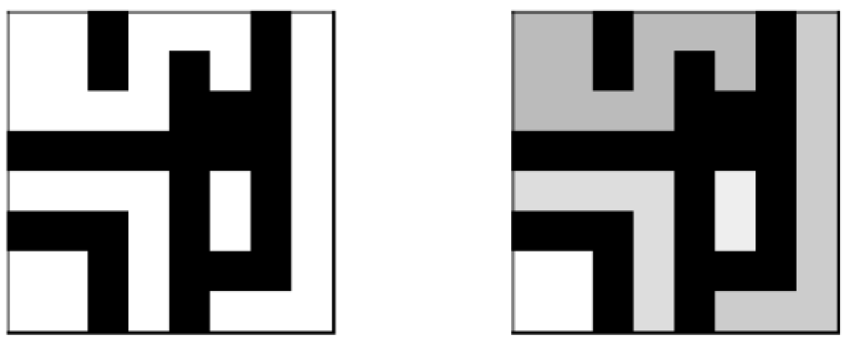
- Identify distinct regions of "connected pixels"

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

a) binary image

1	1	0	1	1	1	0	2
1	1	0	1	0	1	0	2
1	1	1	1	0	0	0	2
0	0	0	0	0	0	0	2
3	3	3	3	0	4	0	2
0	0	0	3	0	4	0	2
5	5	0	3	0	0	0	2
5	5	0	3	0	2	2	2

b) connected components labeling

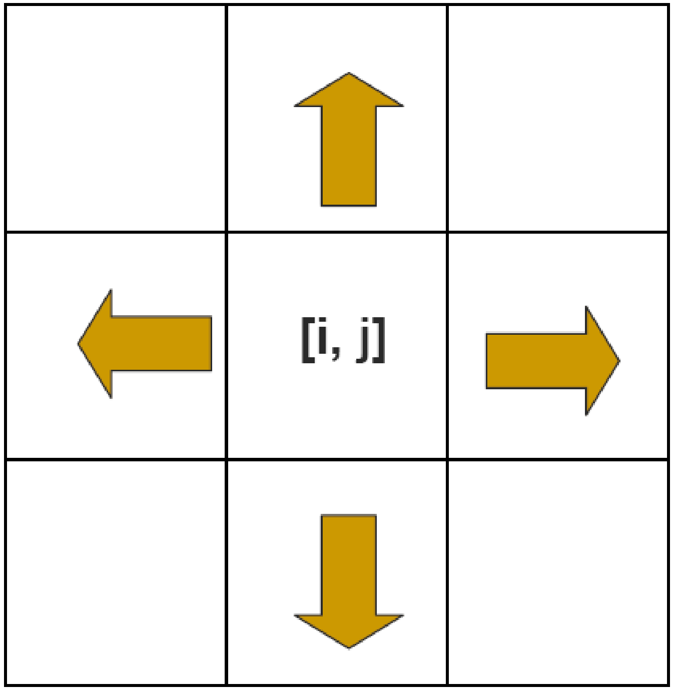


c) binary image and labeling, expanded for viewing

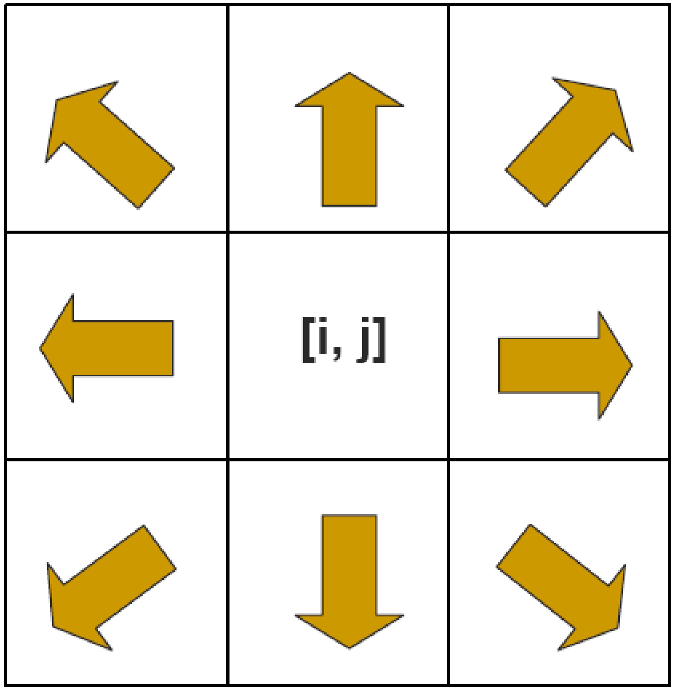
Shapiro and Stockman

Connectedness

- Defining which pixels are considered neighbors



4-connected



8-connected

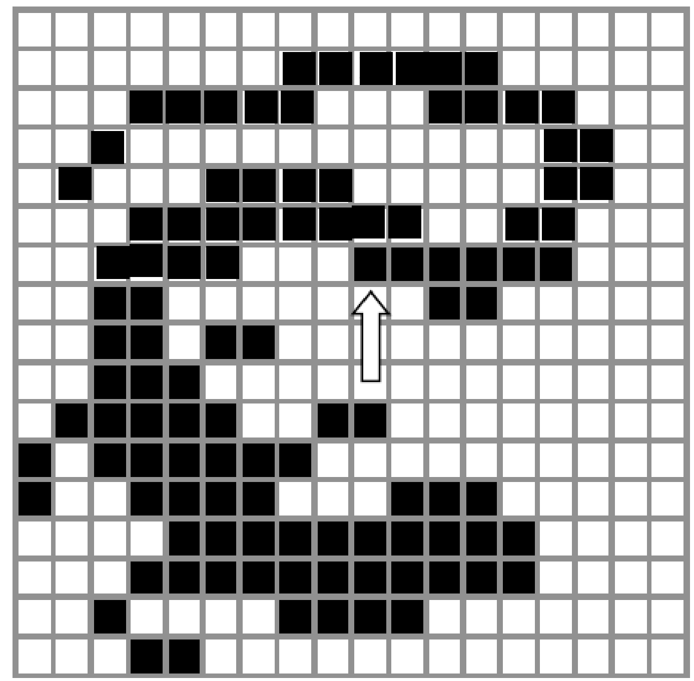
Source: Chaitanya Chandra

Connected components

- We'll consider a sequential algorithm that requires only 2 passes over the image.

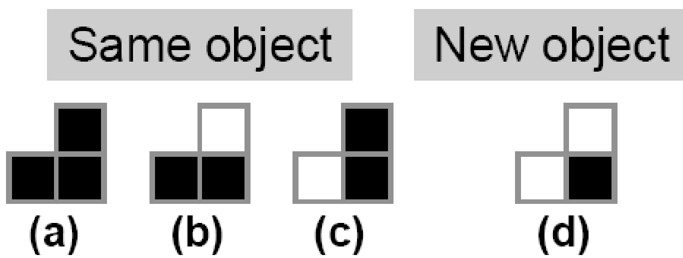
- **Input**: binary image
- **Output**: "label" image, where pixels are numbered per their component

- Note: foreground here is denoted with black pixels.



Sequential connected components

- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).



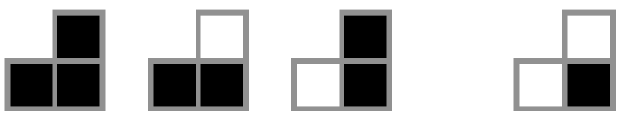
What happens in these cases?

Adapted from J. Neira

Sequential connected components

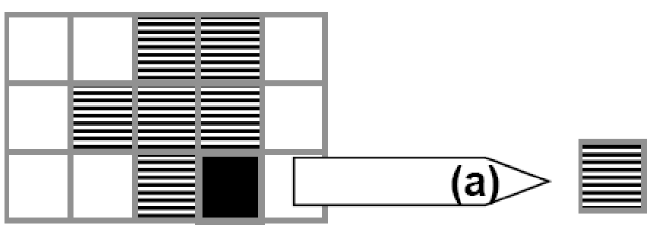
- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

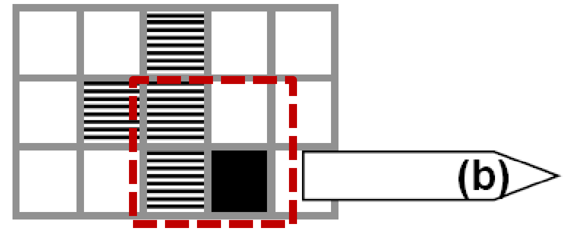
Same object | New object

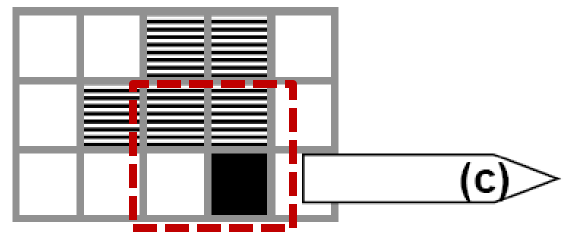


(a) (b) (c) (d)

What happens in these cases?



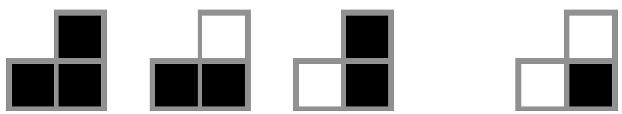




Sequential connected components

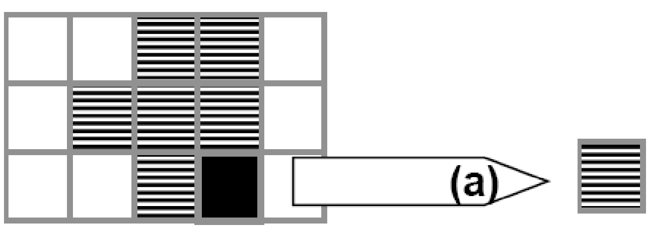
- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

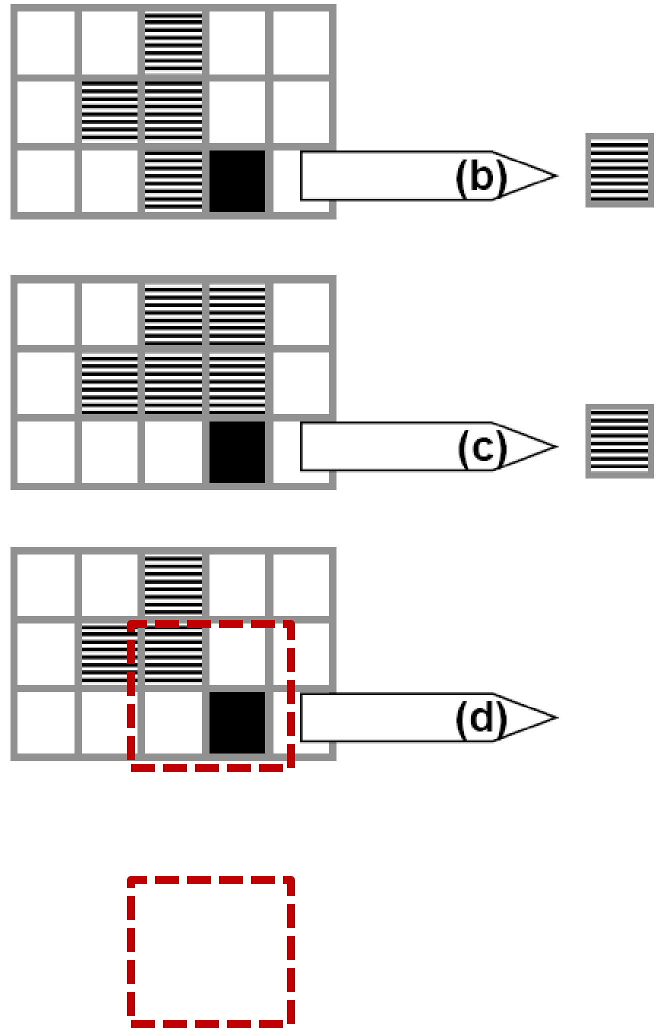
Same object | New object



(a) (b) (c) (d)

What happens in these cases?

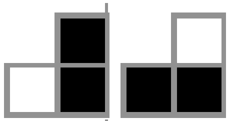




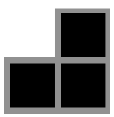
Sequential connected components

- Process the image from left to right, top to bottom.
  1. If the next pixel to process is 1-pixel:

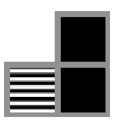
     Already processed

     1. If only one of its neighbors (superior or left) is 1-pixel, copy its label.

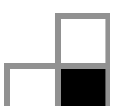
     2. If both are, and have the same label, copy it.

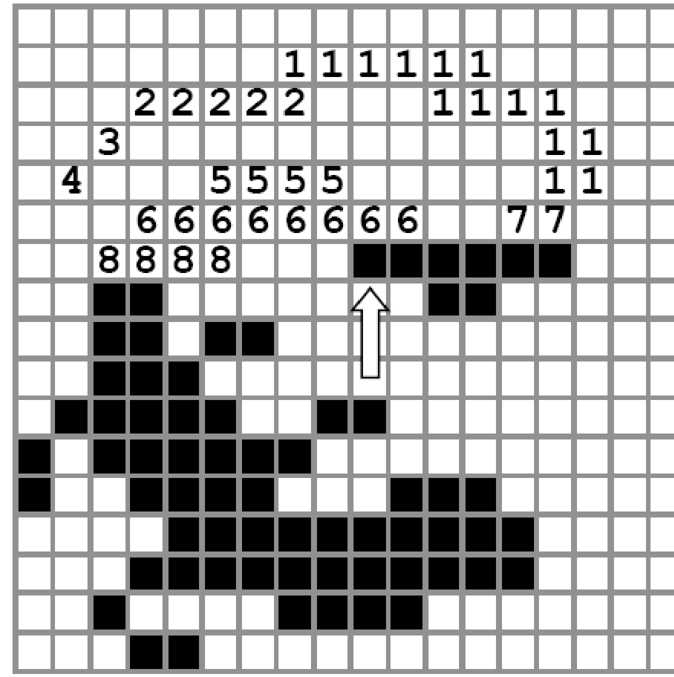
     3. If they have different labels:

        superior? smallest?

        1. Copy the label from the prior.
        2. Reflect the change in the table of equivalences.

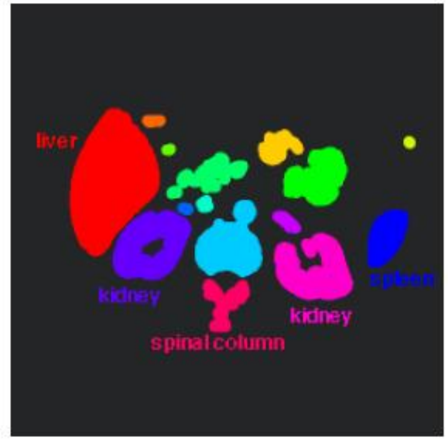
     4. Otw, assign a new label.
  2. More pixels? Go to step 1.



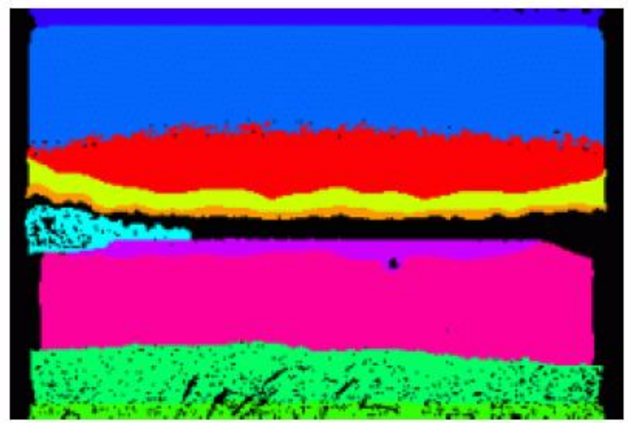
{1, 2, 7}
{3}
{4}
{5, 6, 8}
{}

- Re-label with the smallest of equivalent labels.
- Pixels of the same segment always have the same label.

Connected components



connected components of 1's from thresholded image

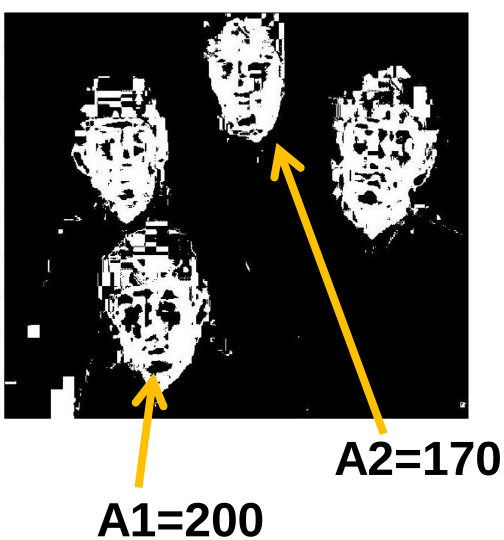


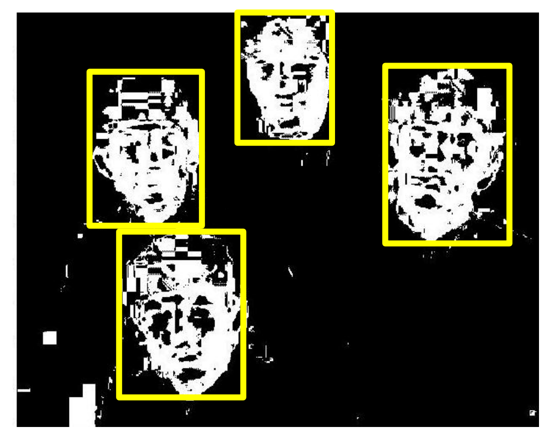
connected components of cluster labels

Slide credit: Pinar Duygulu

Region properties

- Given connected components, can compute simple features per blob, such as:
  - Area (num pixels in the region)
  - Centroid (average x and y position of pixels in the region)
  - Bounding box (min and max coordinates)
  - Circularity (ratio of mean dist. to centroid over std)





Kristen Grauman, UT-Austin

Circularity

- a second measure uses variation off of a circle

circularity(2):

$$C_2 = \frac{\mu_R}{\sigma_R}$$

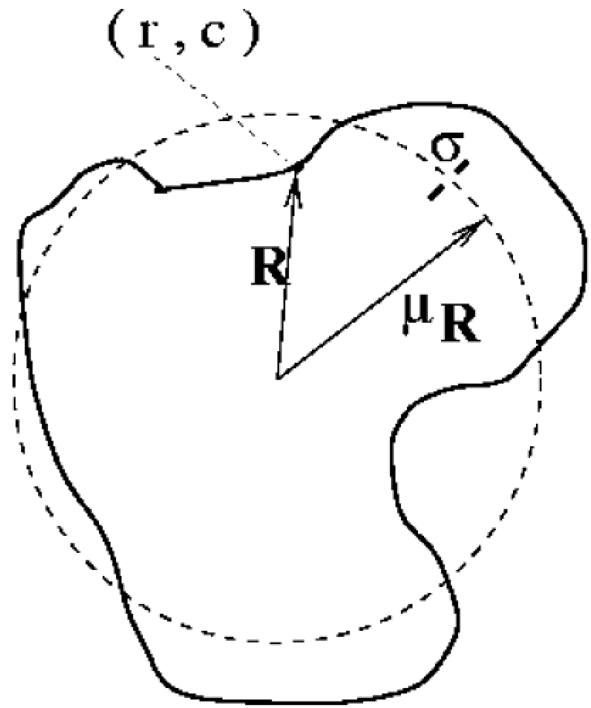
where μ_R and $\sigma_R{}^2$ are the mean and variance of the distance from the centroid of the shape to the boundary pixels (r_k, c_k).

mean radial distance:

$$\mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \|(r_k, c_k) - (\bar{r}, \bar{c})\|$$

variance of radial distance:

$$\sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [\|(r_k, c_k) - (\bar{r}, \bar{c})\| - \mu_R]^2$$



[Haralick]

Shapiro & Stockman

Binary image analysis: basic steps (recap)

- Convert the image into binary form
  - Thresholding
- Clean up the thresholded image
  - Morphological operators
- Extract separate blobs
  - Connected components
- Describe the blobs with region properties

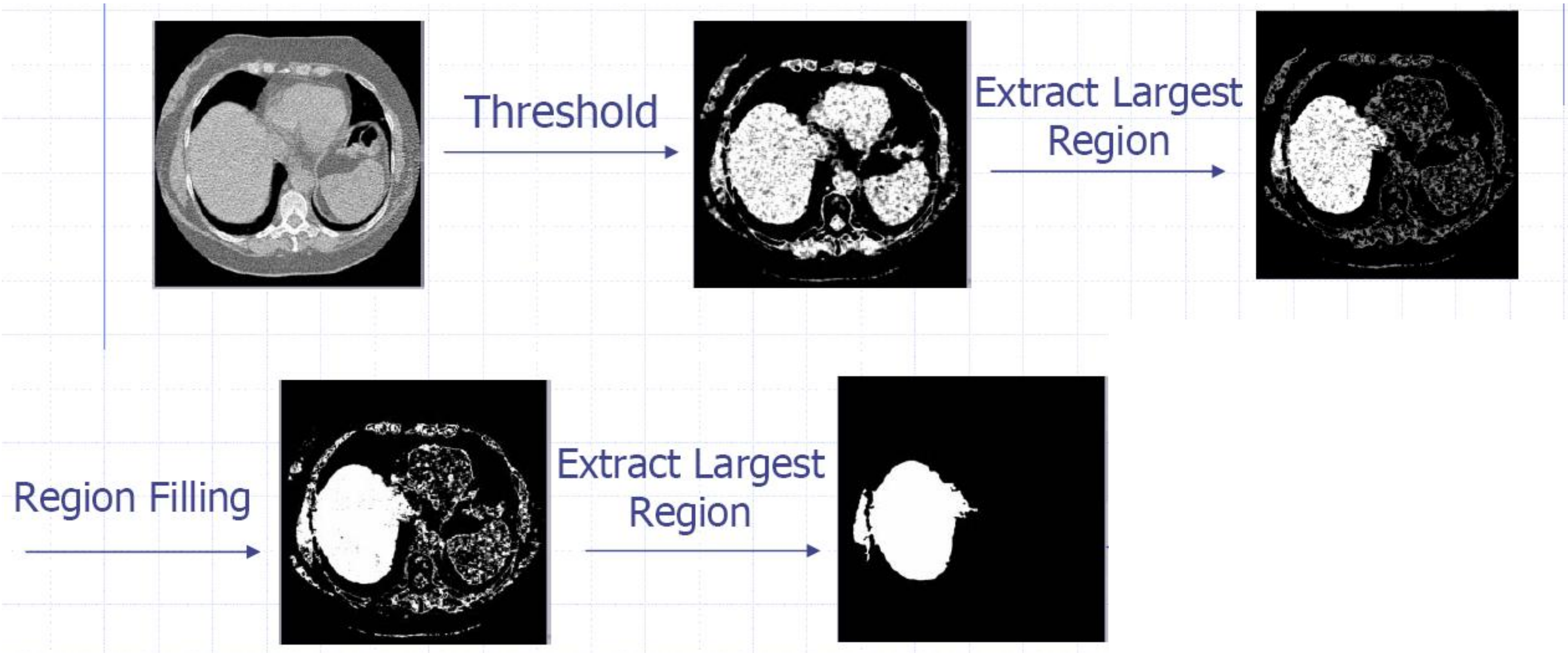
Matlab

- **`N = hist(Y,M)`**
- **`L = bwlabel (BW,N);`**
- **`STATS = regionprops(L,PROPERTIES) ;`**
  - **`'Area'`**
  - **`'Centroid'`**
  - **`'BoundingBox'`**
  - **`'Orientation`, …`**
- **`IM2 = imerode(IM,SE);`**
- **`IM2 = imdilate(IM,SE);`**
- **`IM2 = imclose(IM, SE);`**
- **`IM2 = imopen(IM, SE);`**

Example using binary image analysis: OCR

[Luis von Ahn et al. http://recaptcha.net/learnmore.html]

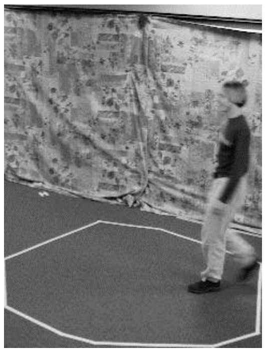
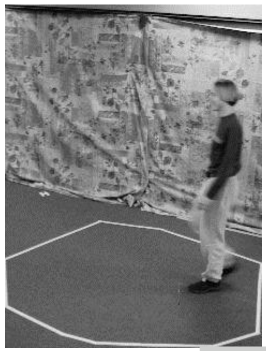
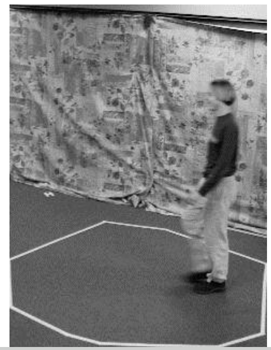
Example using binary image analysis: segmentation of a liver



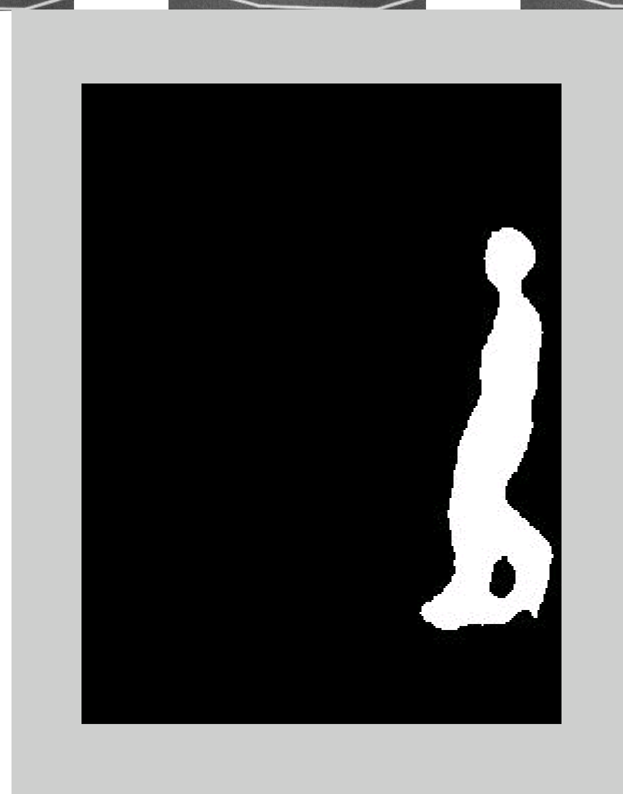
Slide credit: Li Shen

Application by Jie Zhu, Cornell University

Example using binary image analysis: Bg subtraction + blob detection

...



Example using binary image analysis: Bg subtraction + blob detection

University of Southern California
http://iris.usc.edu/~icohen/projects/vace/detection.htm

Binary images

- Pros
  - Can be fast to compute, easy to store
  - Simple processing techniques available
  - Lead to some useful compact shape descriptors

- Cons
  - Hard to get “clean” silhouettes
  - Noise common in realistic scenarios
  - Can be too coarse of a representation
  - Not 3d

Summary

- Operations, tools

Derivative filters

Smoothing, morphology

Thresholding

Connected components

Matched filters

Histograms

- Features, representations

Edges, gradients

Blobs/regions

Local patterns

Textures (next)

Color distributions

Next Class

Features – Corner Detection and SIFT