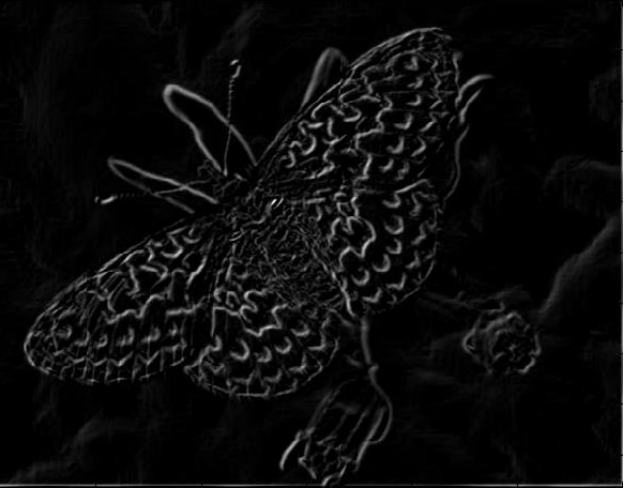
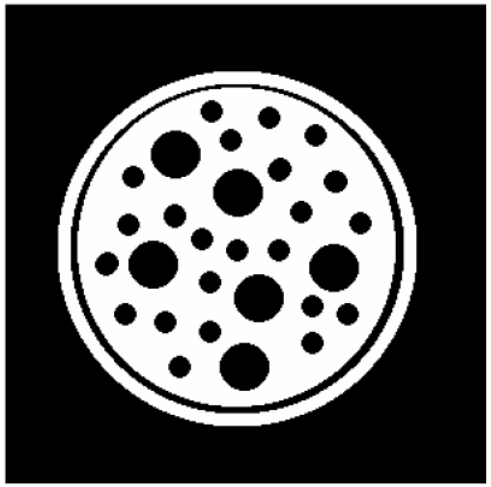
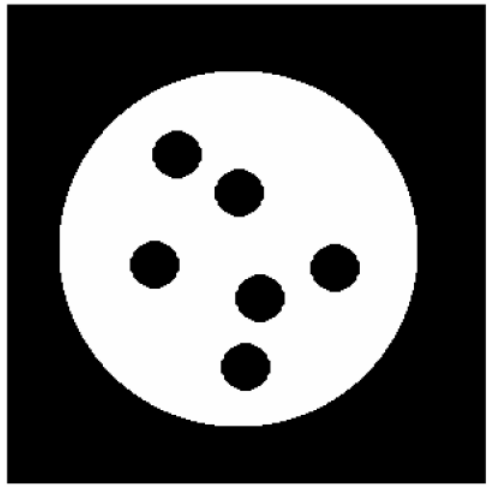


Edges and binary images

Thursday, Sept 15, 2020

Richang Hong, Hefei University of Technology





Outline

- Last time
  - Image gradients
  - Seam carving – gradients as “energy”
  - Gradients → edges and contours
  - Template matching
    - Image patch as a filter
- Today
  - Binary image analysis
    - Blobs and regions

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

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

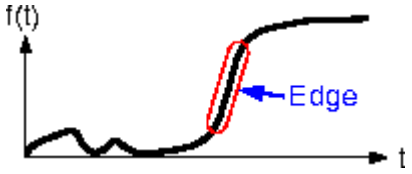
original image (Lena)

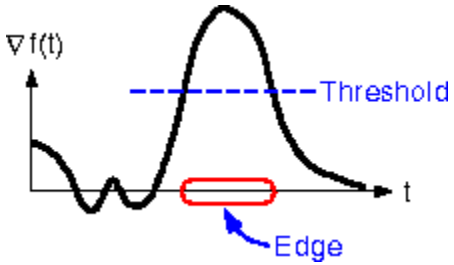
Slide credit: Steve Seitz

The Canny edge detector

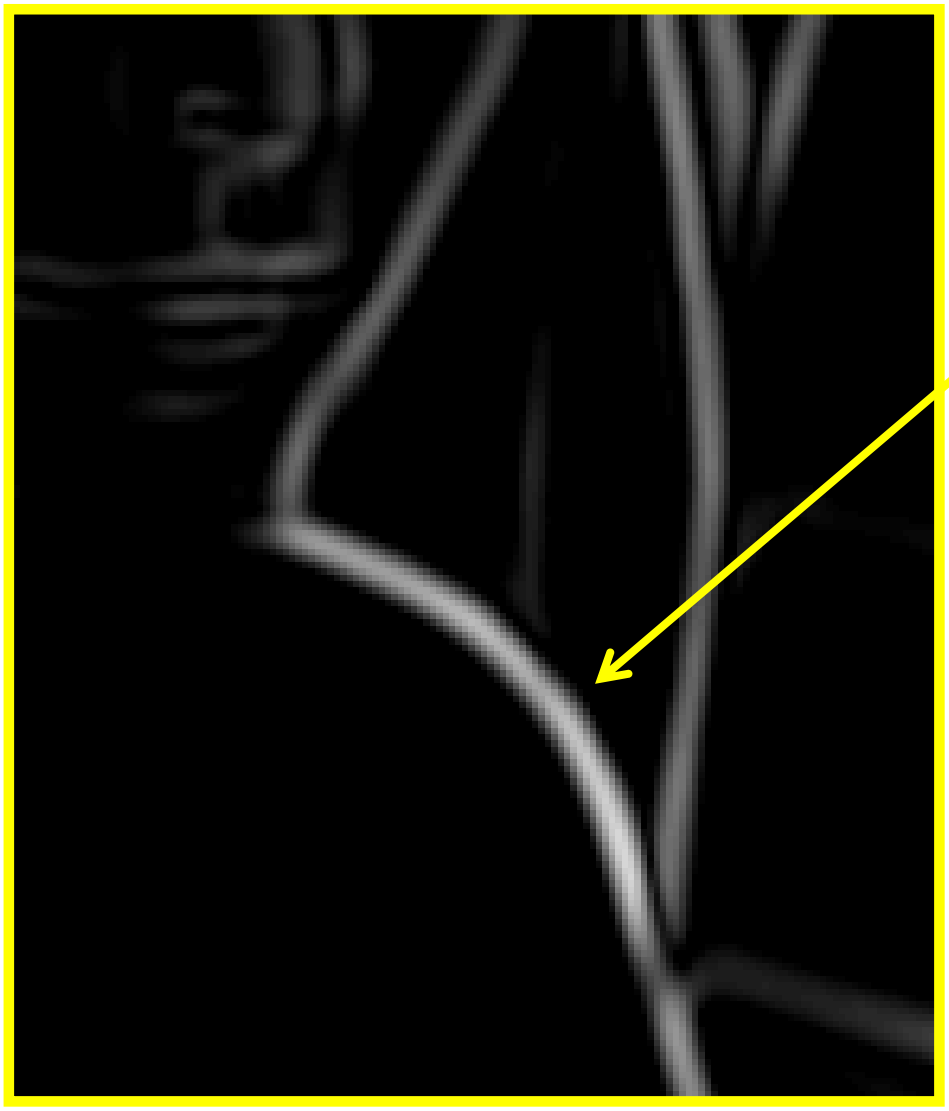
norm of the gradient

The Canny edge detector



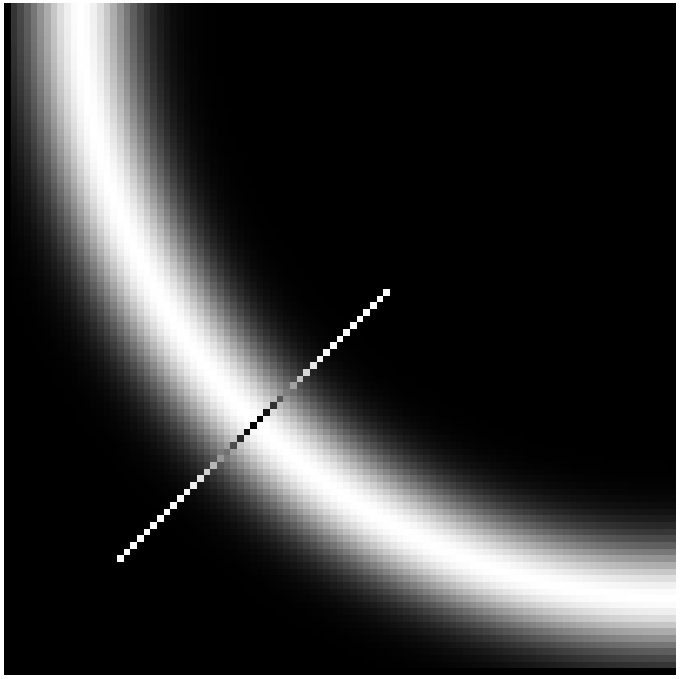


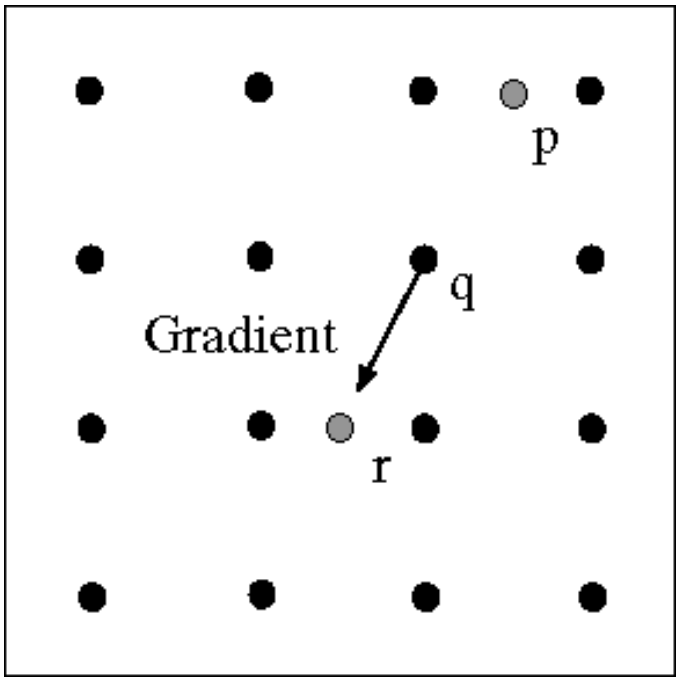
Non-maximum suppression : Thin wide “ridges” down to single pixel width



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)

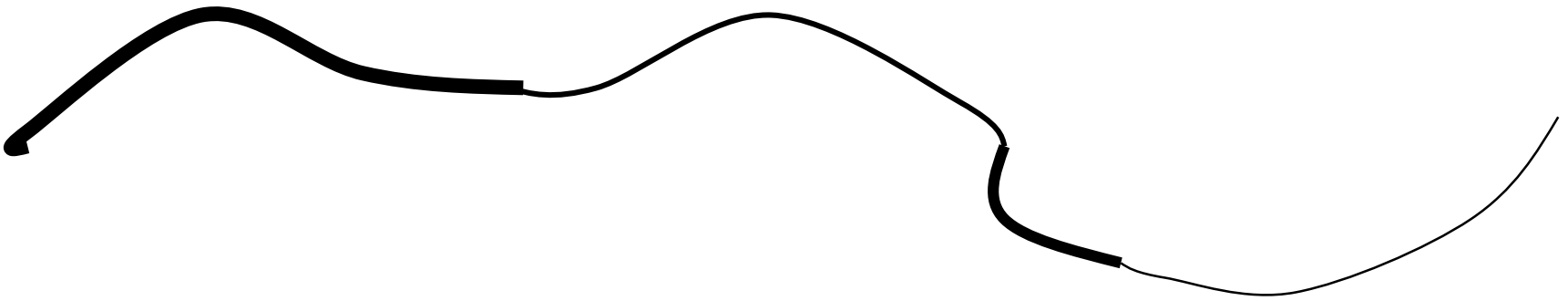
Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Credit: James Hays

Hysteresis thresholding

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



Source: Steve Seitz

Final Canny Edges

Credit: James Hays

Recap: Canny edge detector

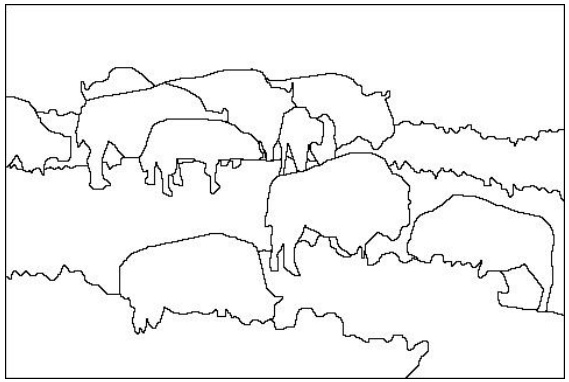
- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin wide “ridges” down to single pixel width
- **Linking and thresholding** (**hysteresis**):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves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

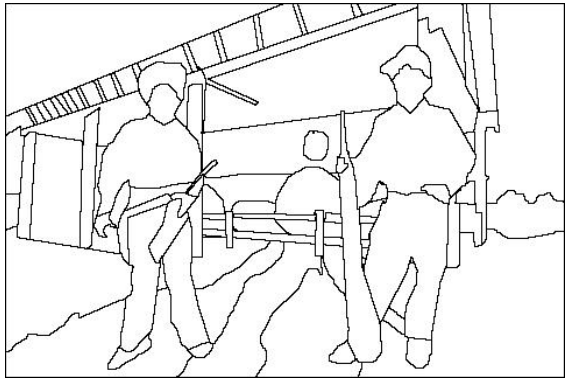
- MATLAB: `edge(image, 'canny');`
- `>>help edge`

Source: D. Lowe, L. Fei-Fei

Low-level edges vs. perceived contours

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

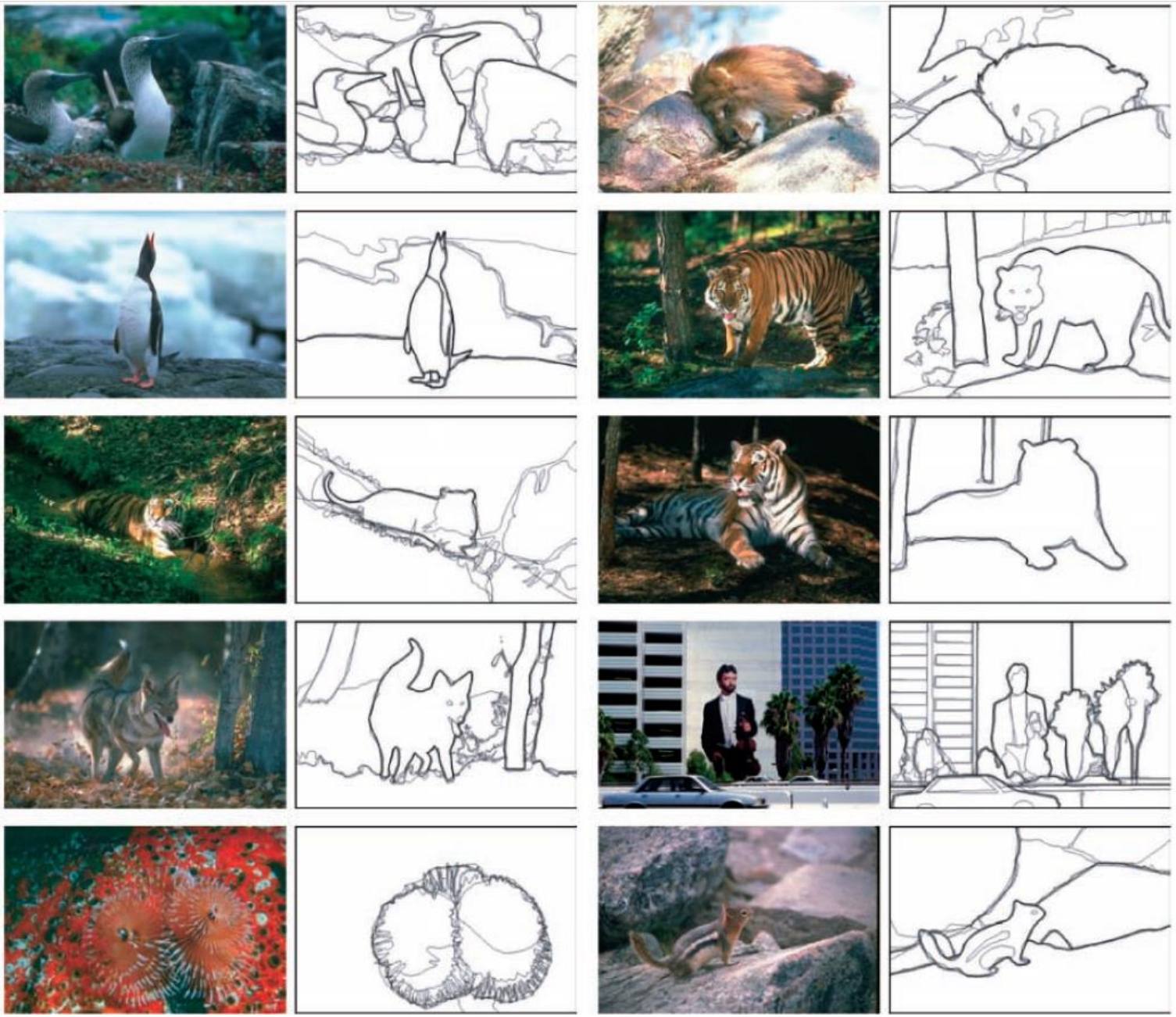



Berkeley segmentation database:

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

Source: L. Lazebnik

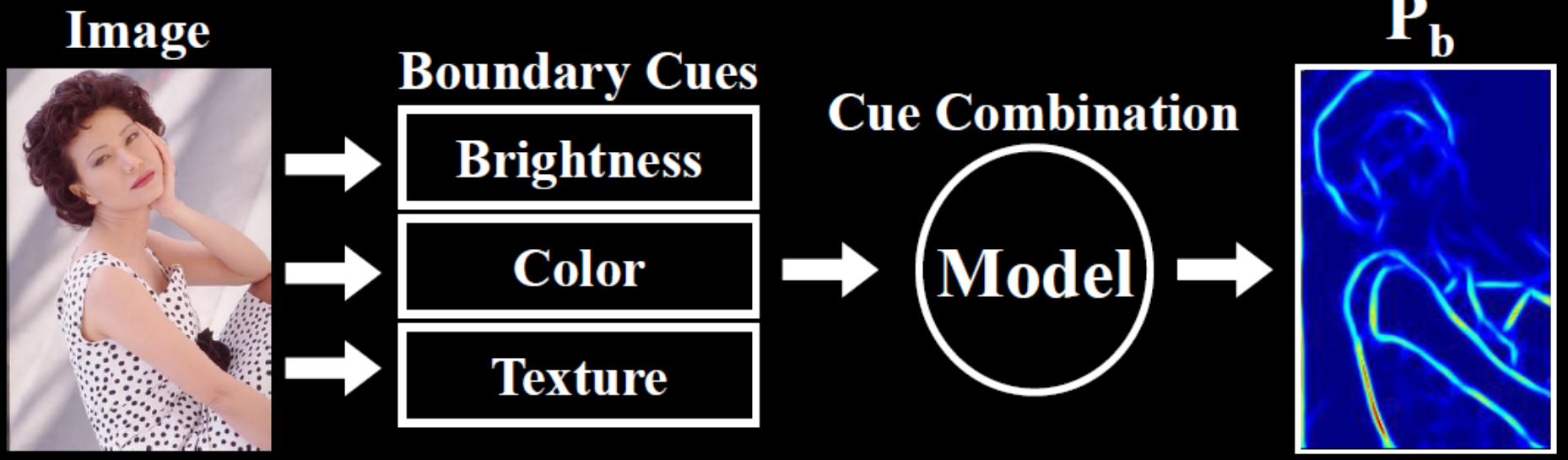
Learn from humans which combination of features is most indicative of a “good” contour?



[D. Martin et al. PAMI 2004]

Human-marked segment boundaries

Dataflow



Challenges: texture cue, cue combination

Goal: learn the posterior probability of a boundary $P_b(x,y,\theta)$ from local information only

[D. Martin et al. PAMI 2004]

Recall: image filtering

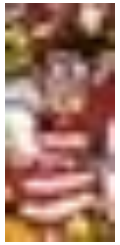
- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Adapted from Derek Hoiem

Where's Waldo?

Scene

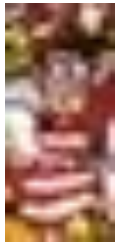


Template

Slide credit: Kristen Grauman

Where's Waldo?

Detected template

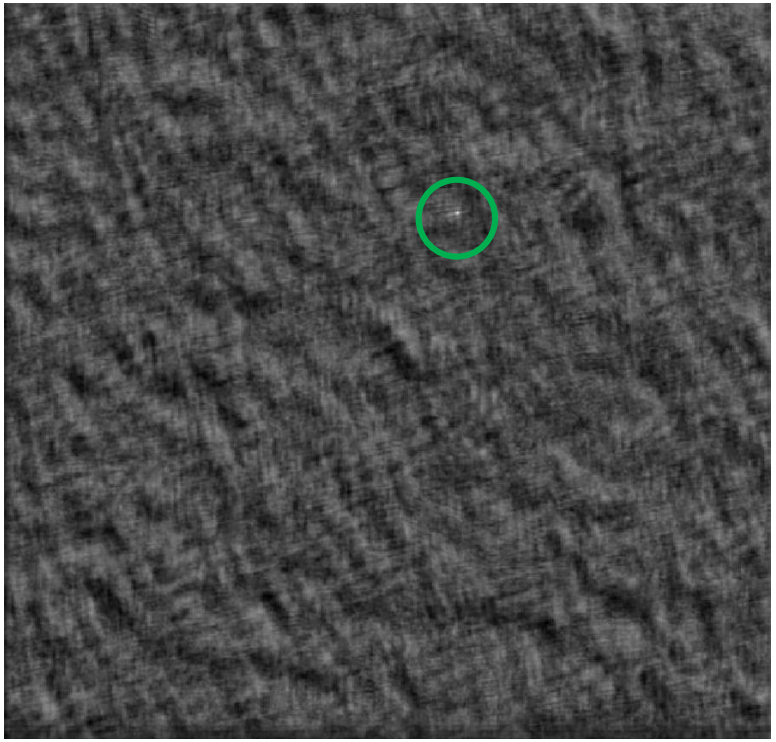


Template

Slide credit: Kristen Grauman

Where's Waldo?

Detected template



Correlation map

Slide credit: Kristen Grauman

Recap: Mask properties

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 → no response in constant regions
  - High absolute value at points of high contrast
- Filters act as templates
  - Highest response for regions that “look the most like the filter”
  - Dot product as correlation

Summary so far

- Image gradients
- Seam carving – gradients as “energy”
- Gradients → edges and contours
- Template matching
  - Image patch as a filter

Next

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

Motivation

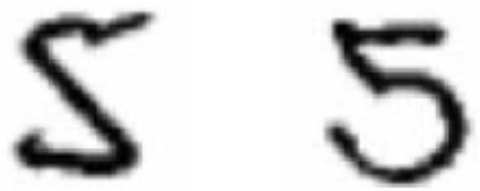


Fig. 1. Examples of two handwritten digits. In terms of pixel-to-pixel comparisons, these two images are quite different, but to the human observer, the shapes appear to be similar.

Figure from Belongie et al.

Chamfer distance

- Average distance to nearest feature

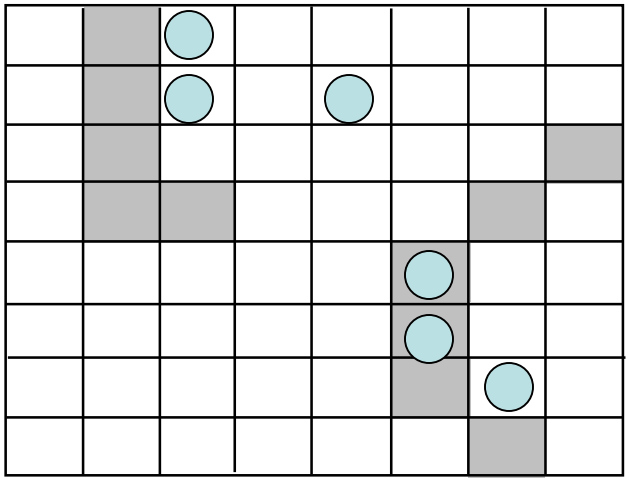
$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

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

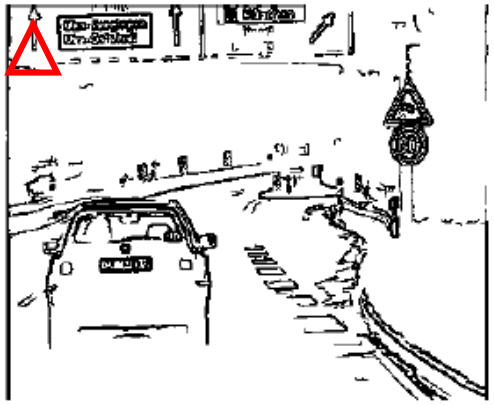


$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

Chamfer distance

- Average distance to nearest feature

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



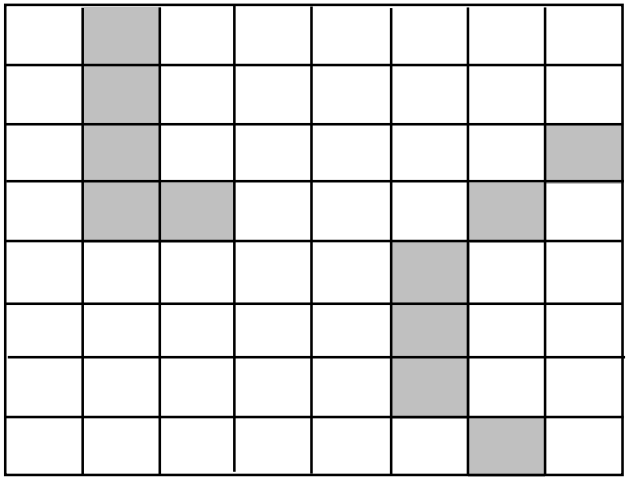
Edge image

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

How expensive is a naïve implementation?

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

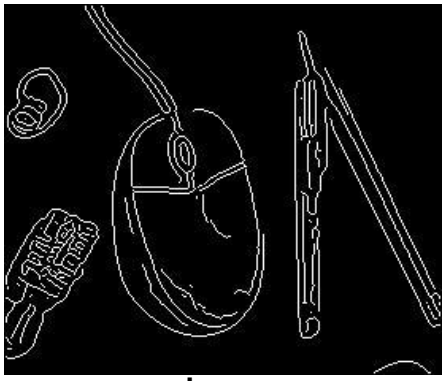
Distance Transform is a function $D(\cdot)$ that for each image pixel p assigns a non-negative number $D(p)$ corresponding to distance from p to the nearest feature in the image I

Features could be edge points, foreground points,…

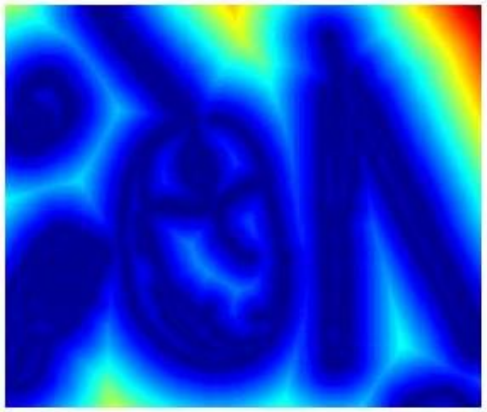
Source: Yuri Boykov

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
```

Slide credit: Kristen Grauman

Distance transform (1D)

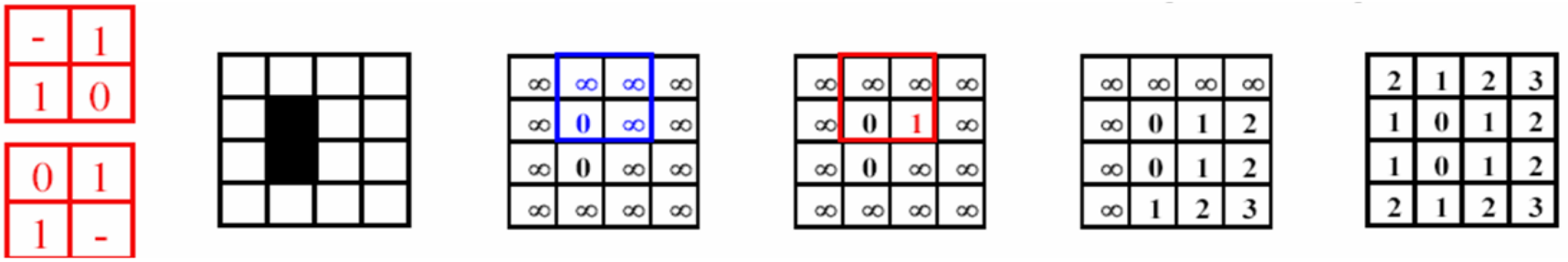
Two pass O(n) algorithm for 1D L_1 norm

1. Initialize: 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

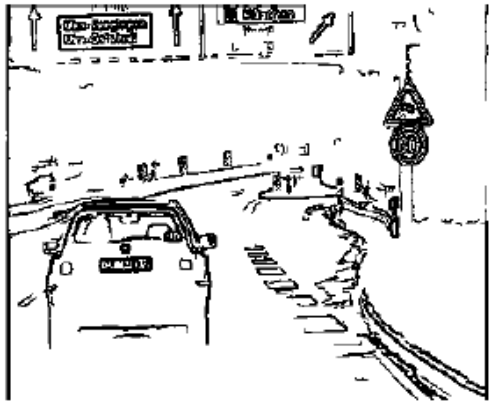


Adapted from D. Huttenlocher

Chamfer distance

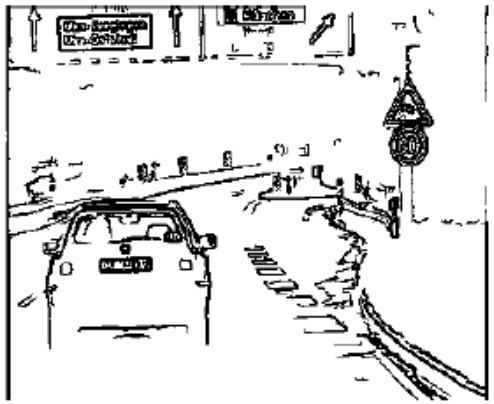
- Average distance to nearest feature

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



Edge image **Distance transform image**

Chamfer distance



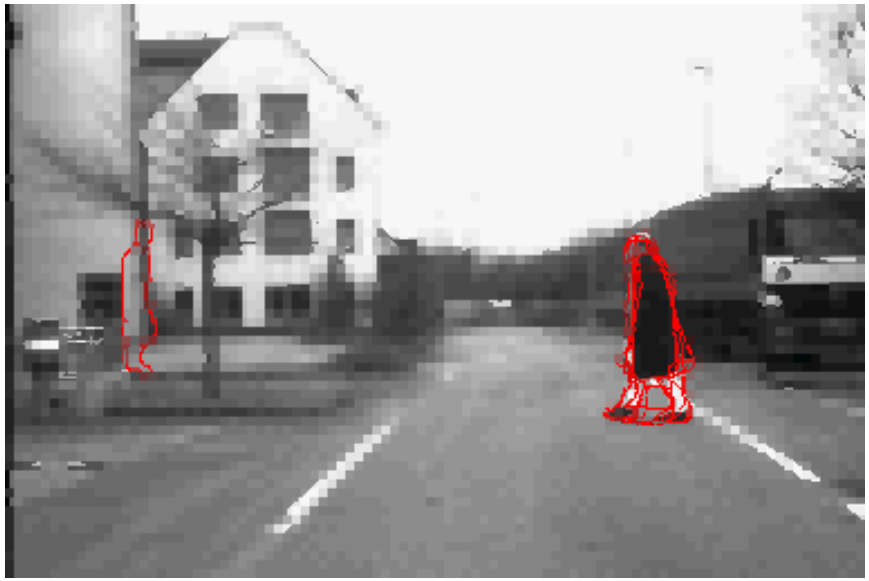
Edge image **Distance transform image**

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

Chamfer matching system



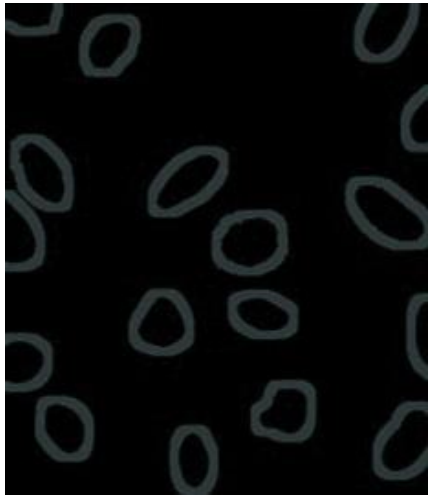
- Gavrila et al. http://gavrila.net/Research/Chamfer_System/chamfer_system.html

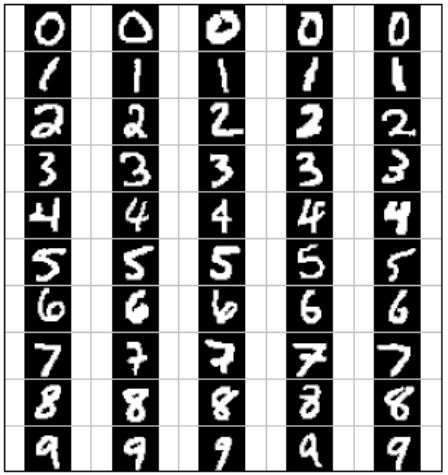
Today

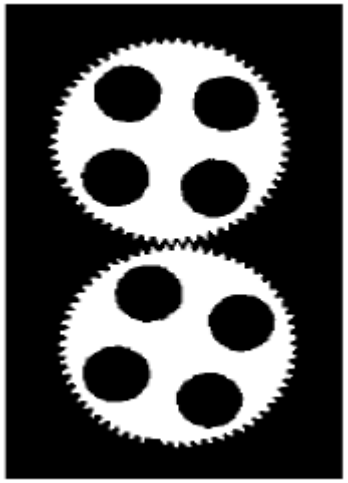
- Edge detection and matching
  - process the image gradient to find curves/contours
  - comparing contours

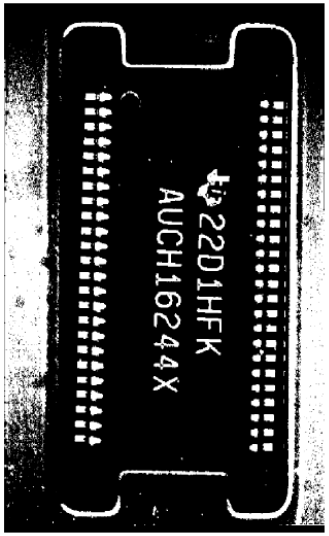
- Binary image analysis
  - blobs and regions

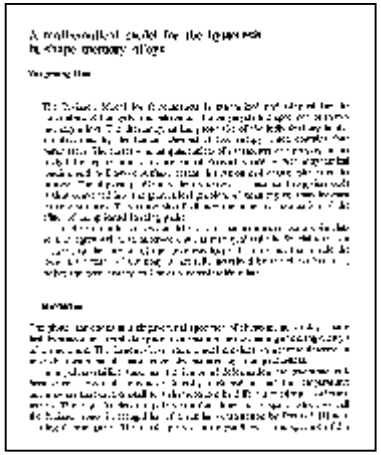
Binary images











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

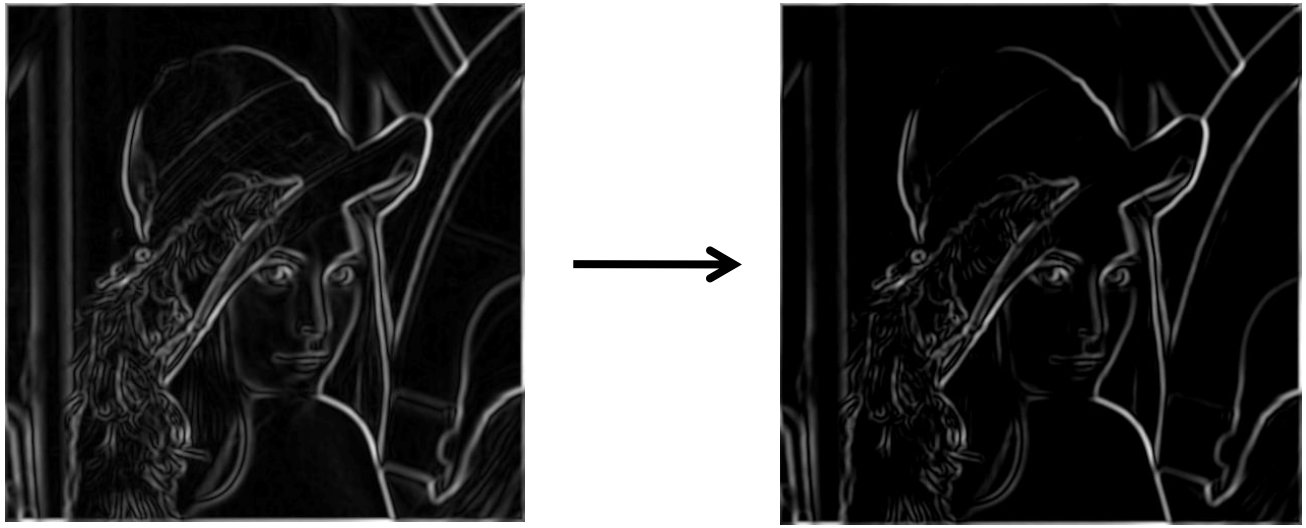
- Two pixel values
  - Foreground and background
  - Mark region(s) of interest

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

- 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

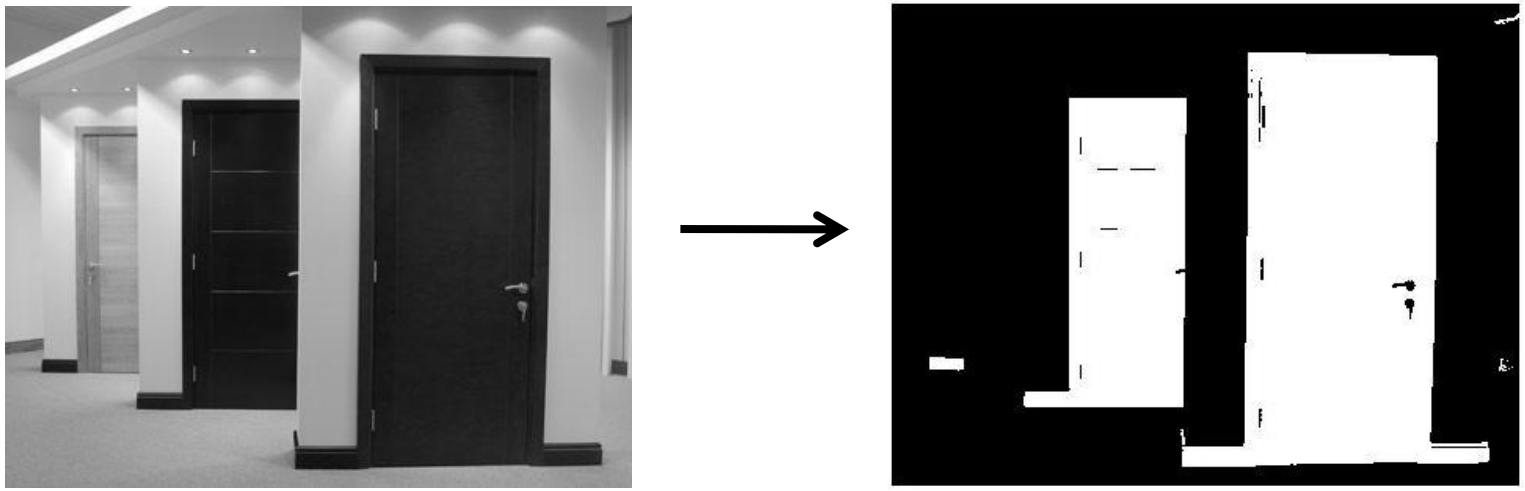
Looking for pixels that differ significantly from the “empty” background.

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

Thresholding

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

Example: intensity-based detection



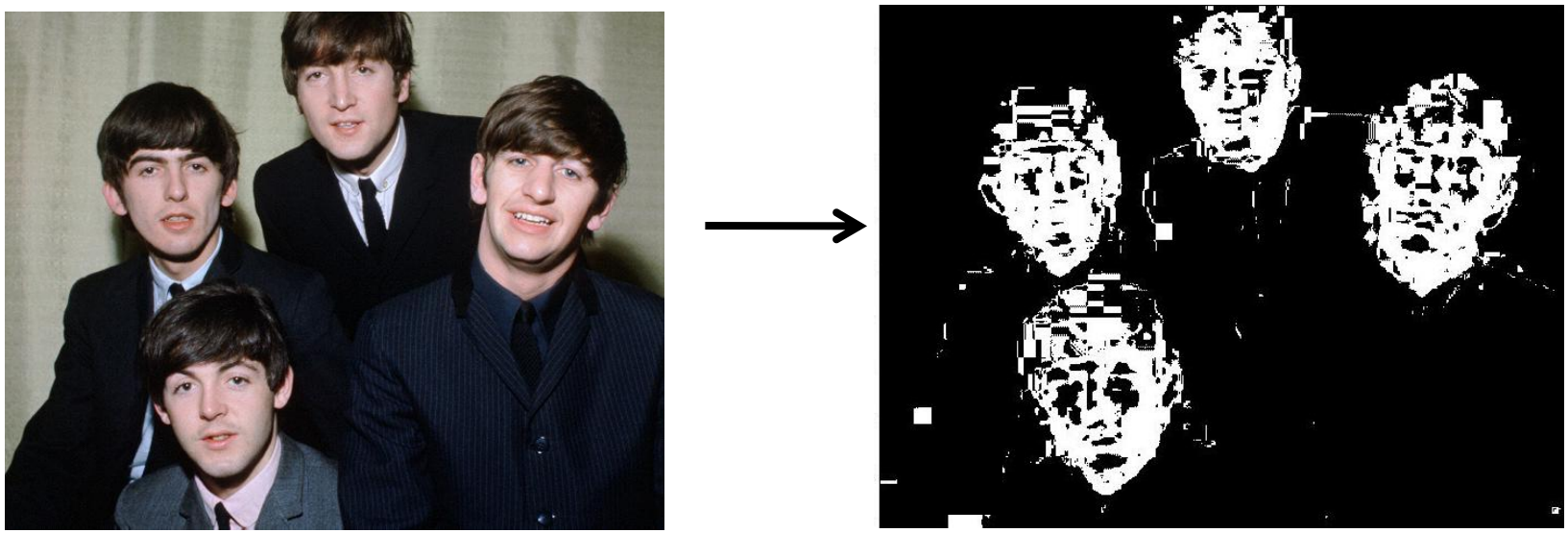
```
fg_pix = find(im < 65);
```

Looking for dark pixels

Thresholding

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

Example: color-based detection



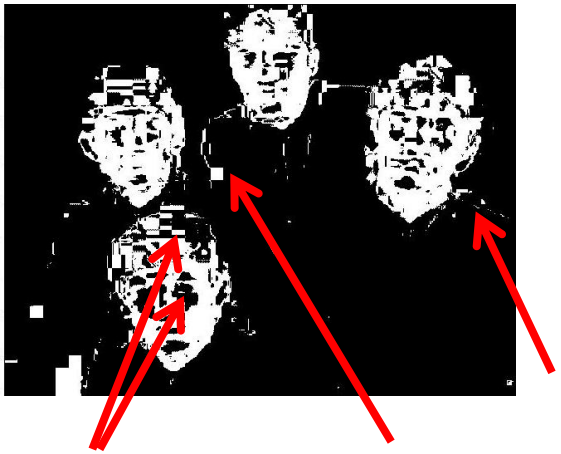
```
fg_pix = find(hue > t1 & hue < t2);
```

Looking for pixels within a certain hue range.

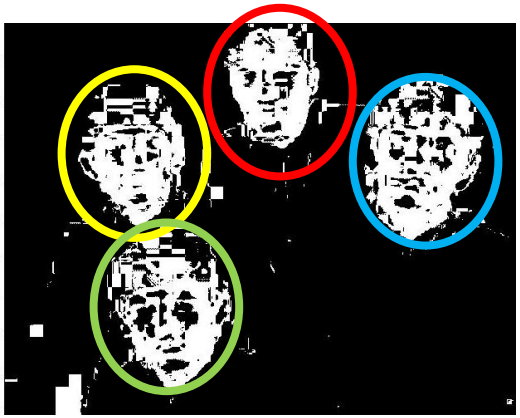
Slide credit: Kristen Grauman

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



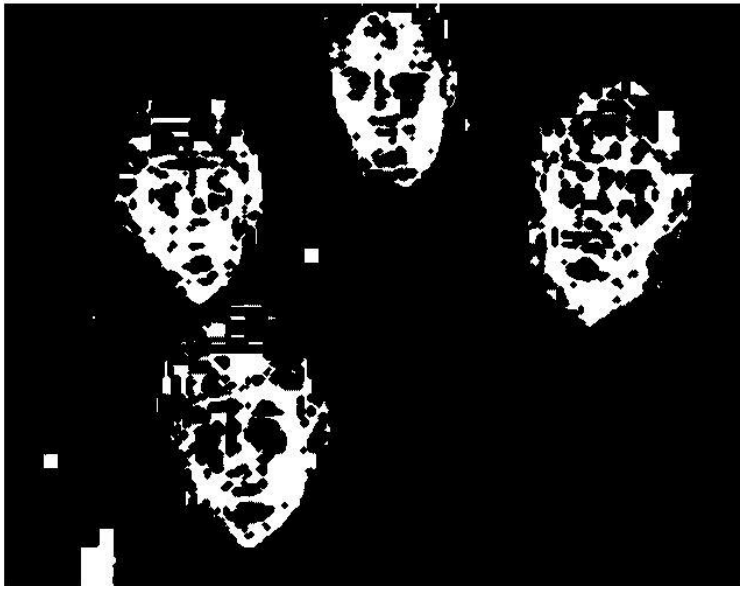
Slide credit: Kristen Grauman

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

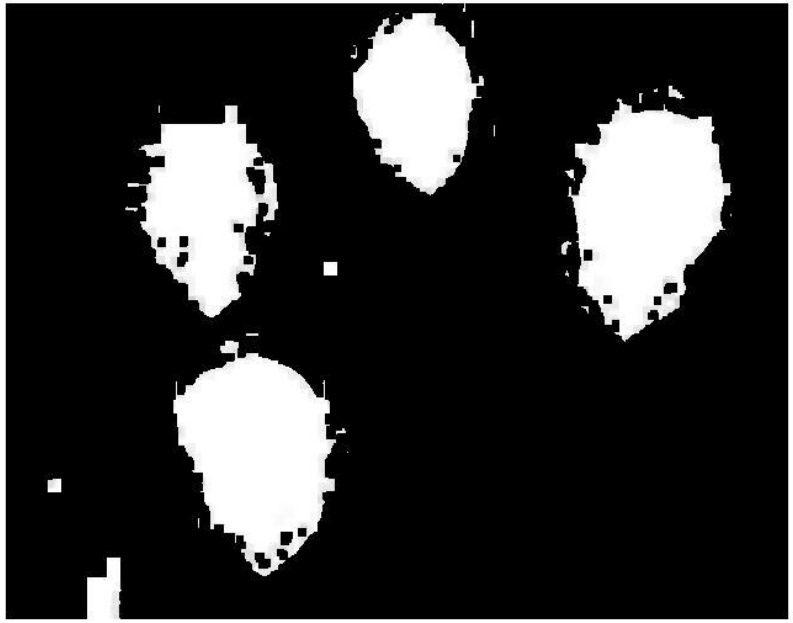
After dilation

Slide credit: Kristen Grauman

Erosion

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

Before erosion

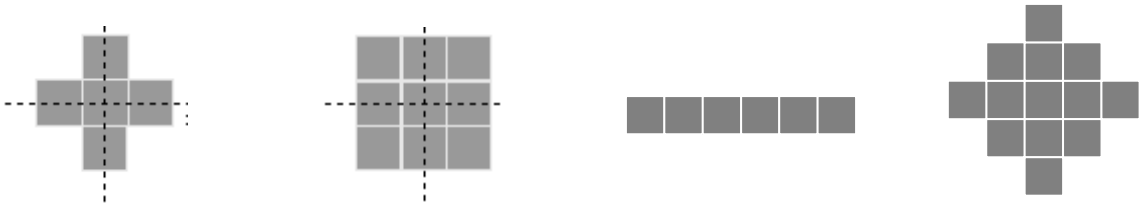


After erosion

Slide credit: Kristen Grauman

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
```

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**

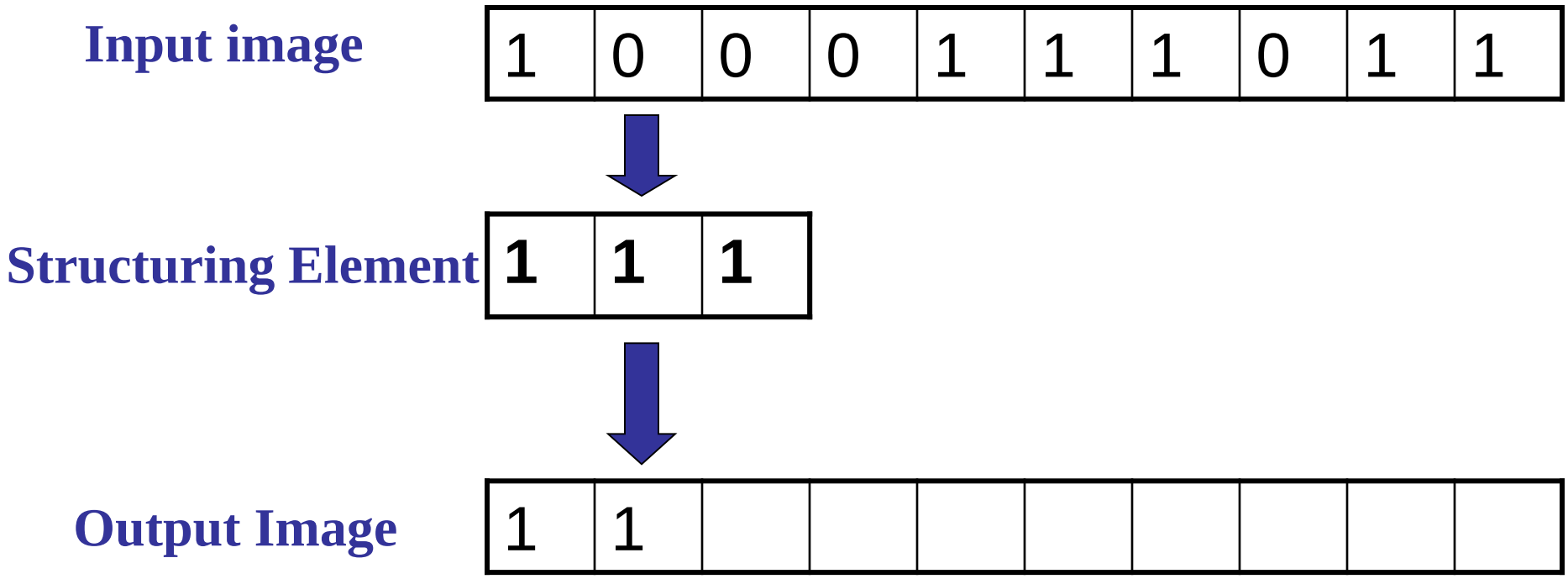
$$g(x) = f(x) \oplus SE$$

Output Image

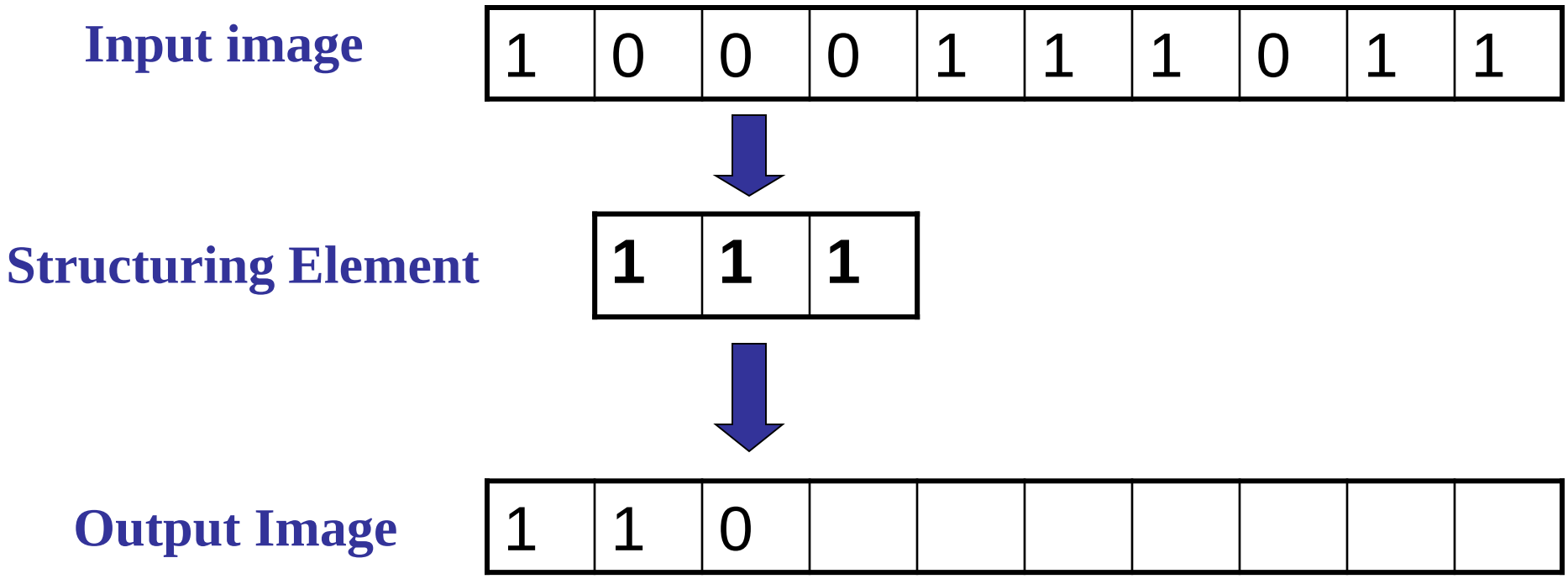
1									

Adapted from T. Moeslund

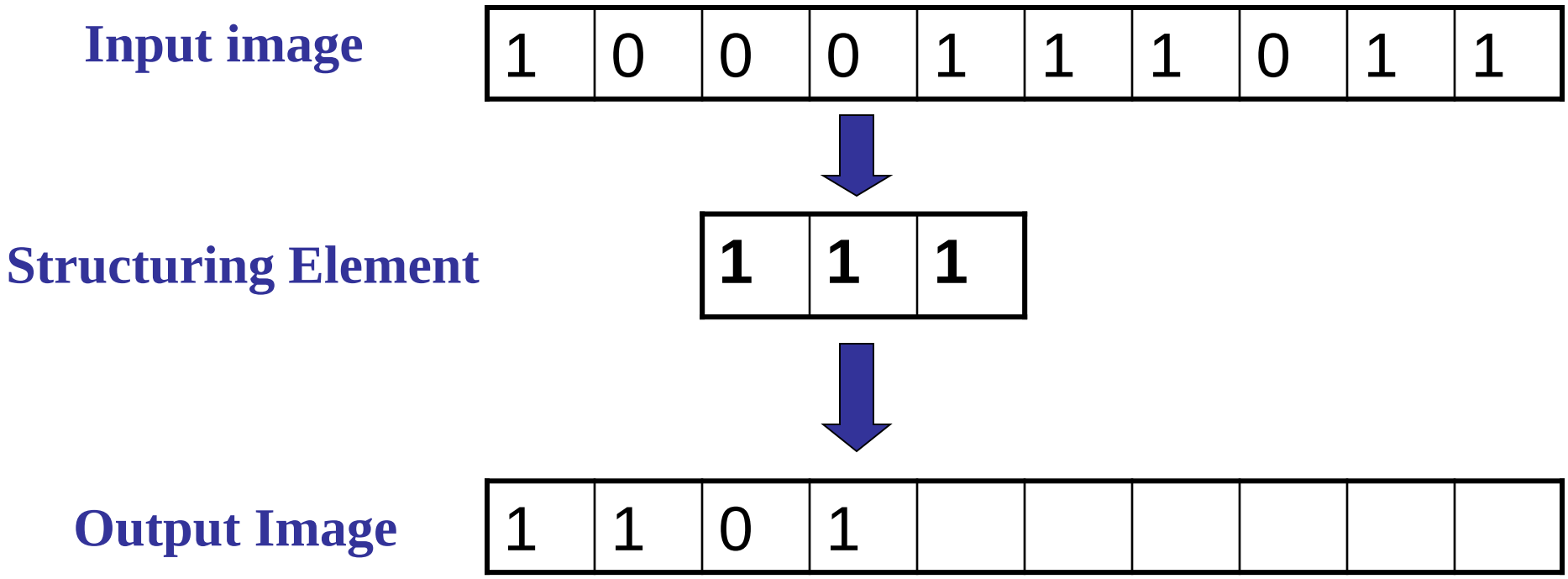
Example for Dilation



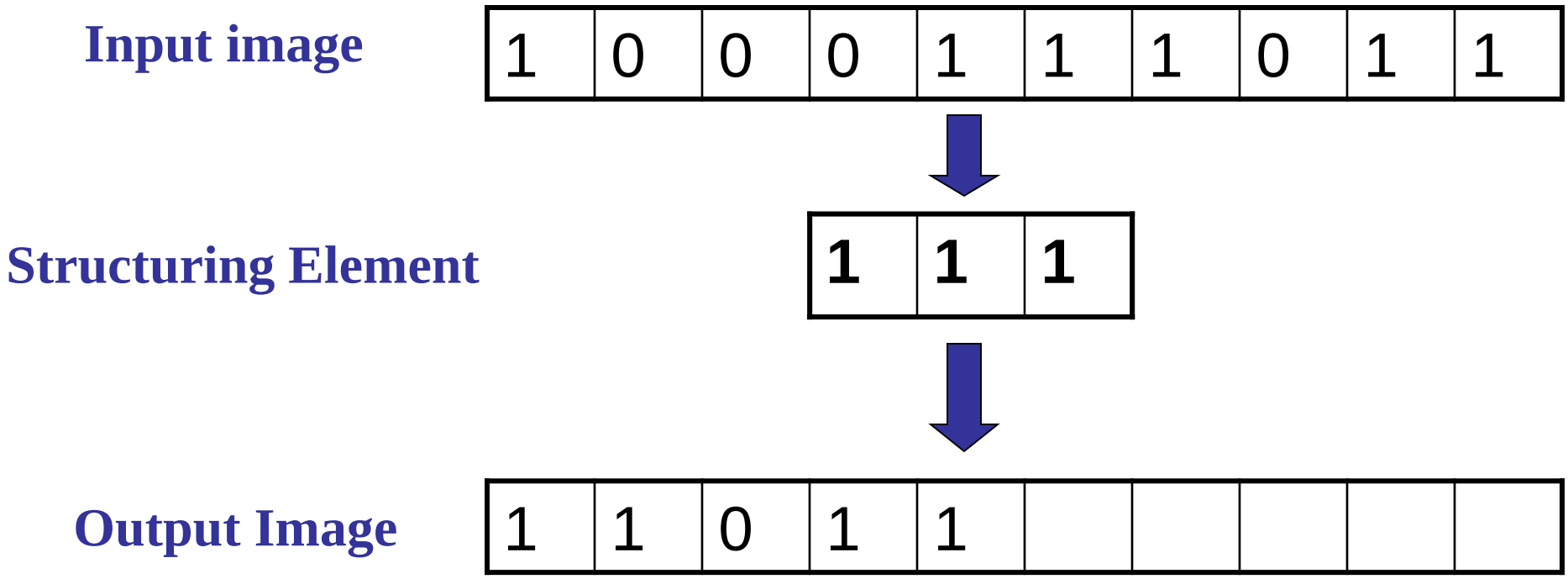
Example for Dilation



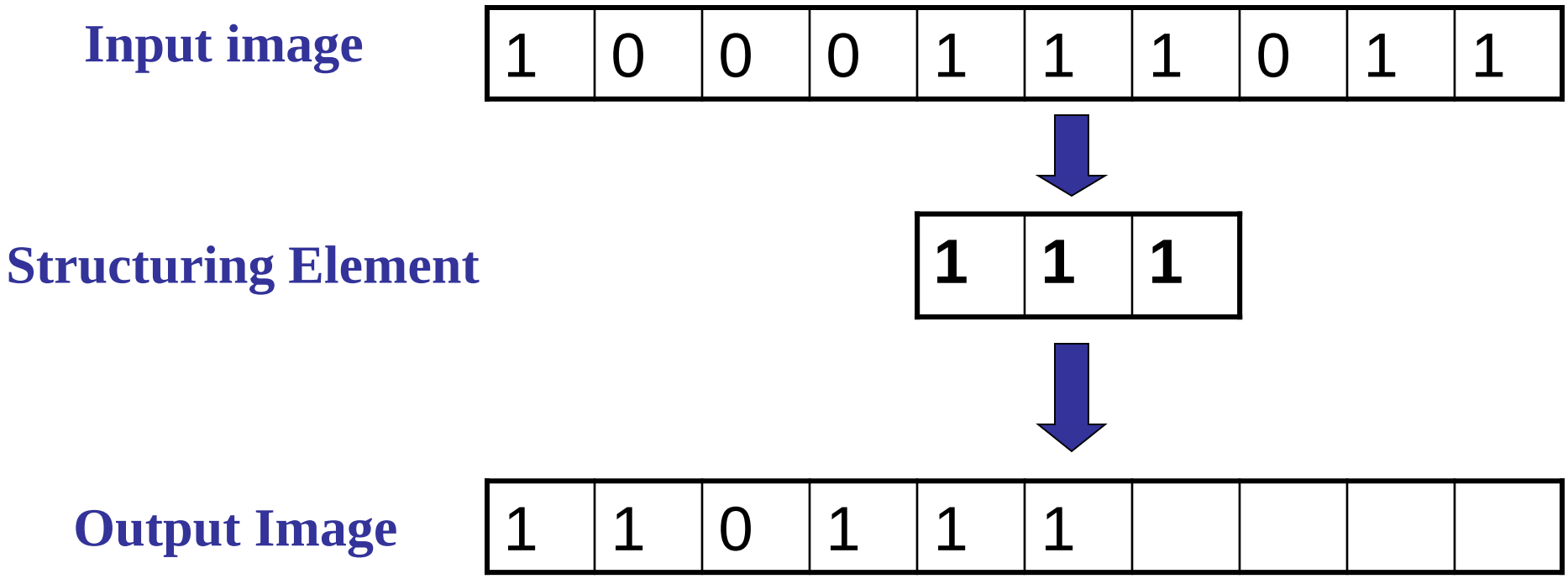
Example for Dilation



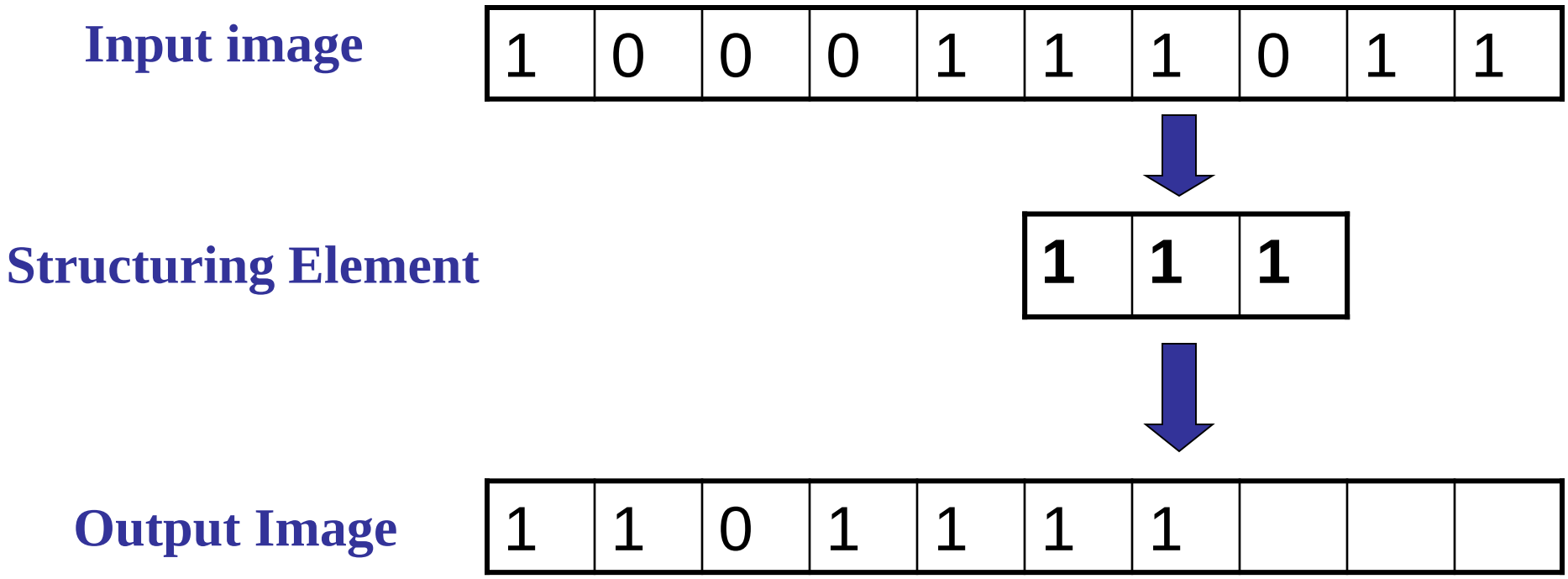
Example for Dilation



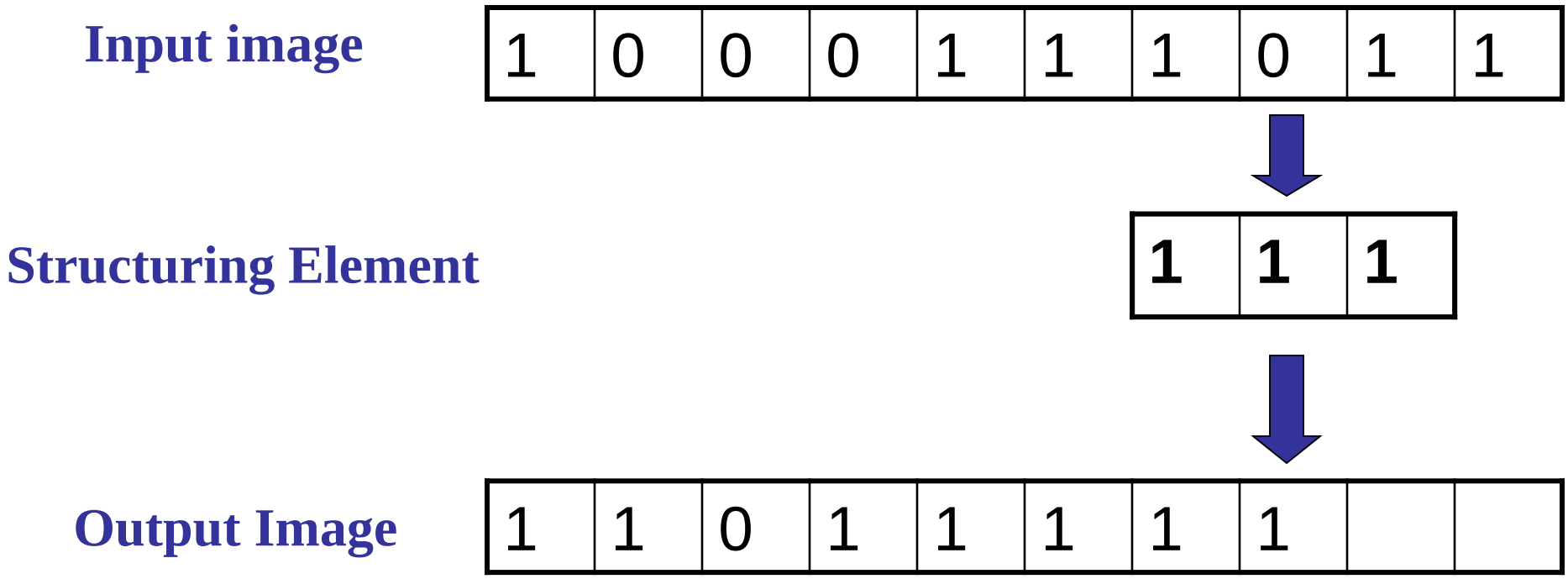
Example for Dilation



Example for Dilation



Example for Dilation



Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

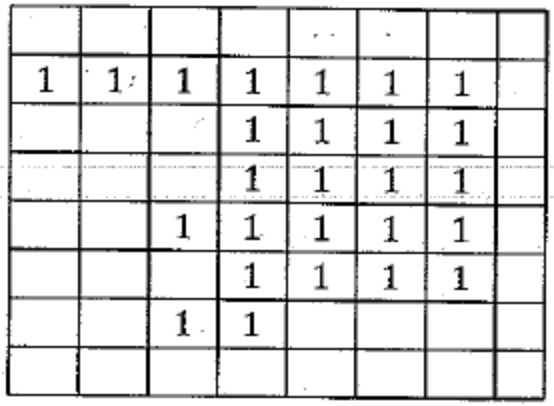
Output Image

1	1	0	1	1	1	1	1	1	

Note that the object gets bigger and holes are filled.

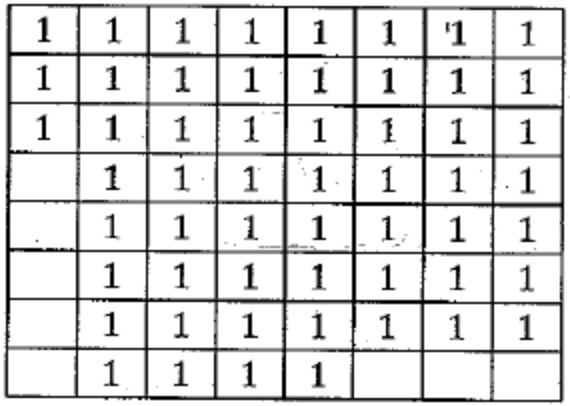
```
>> help imdilate
```

2D example for dilation



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

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



(*c*) Dilation **B** ⊕ **S**

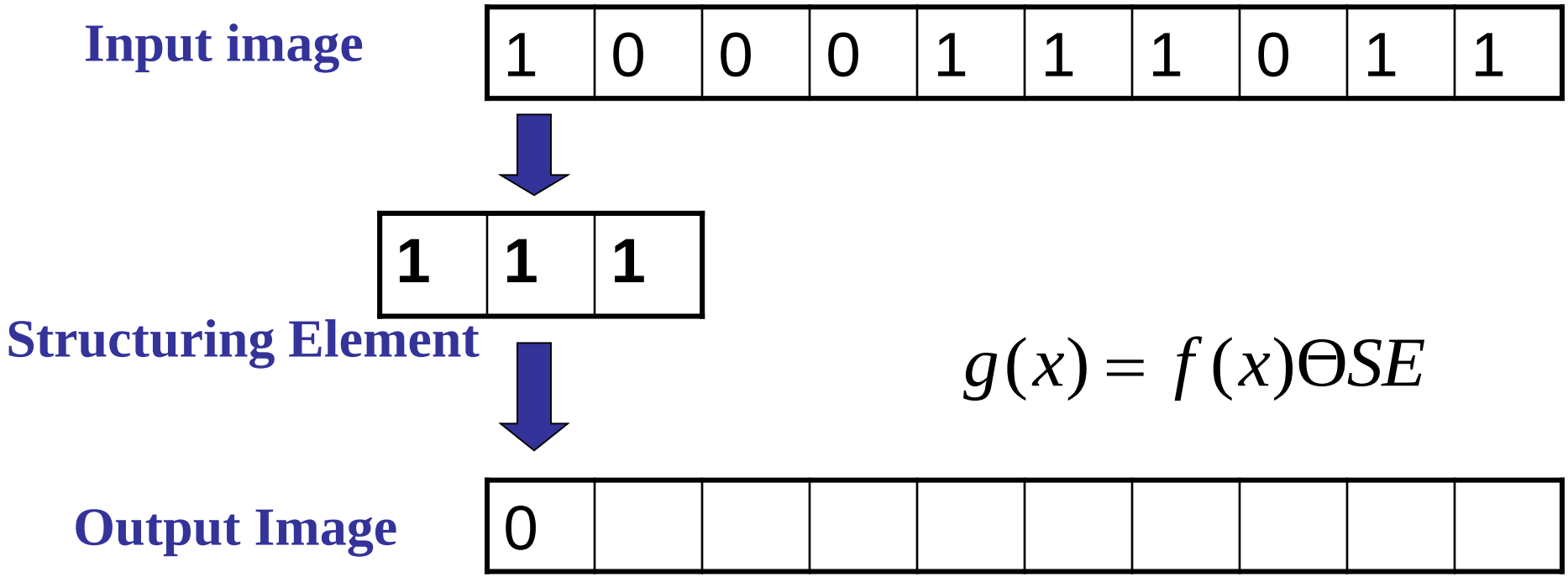
Shapiro & Stockman

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)



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

Example for Erosion (1D)

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

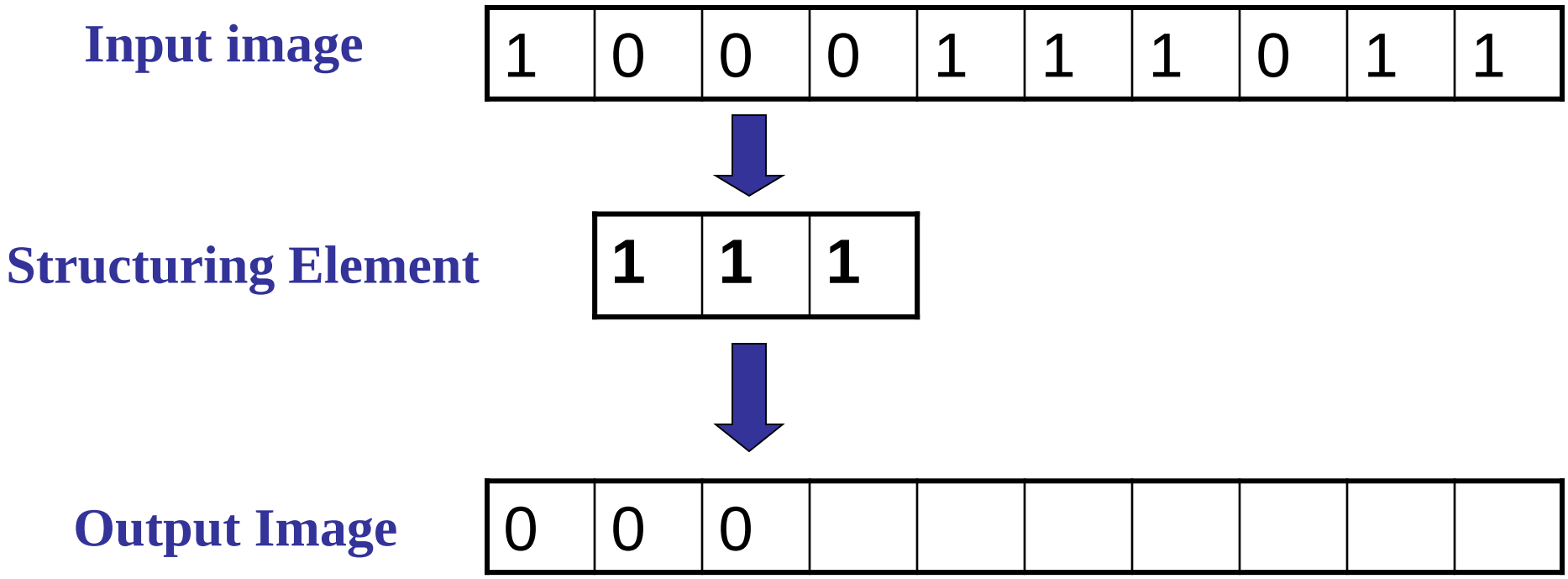
1	**1**	**1**

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

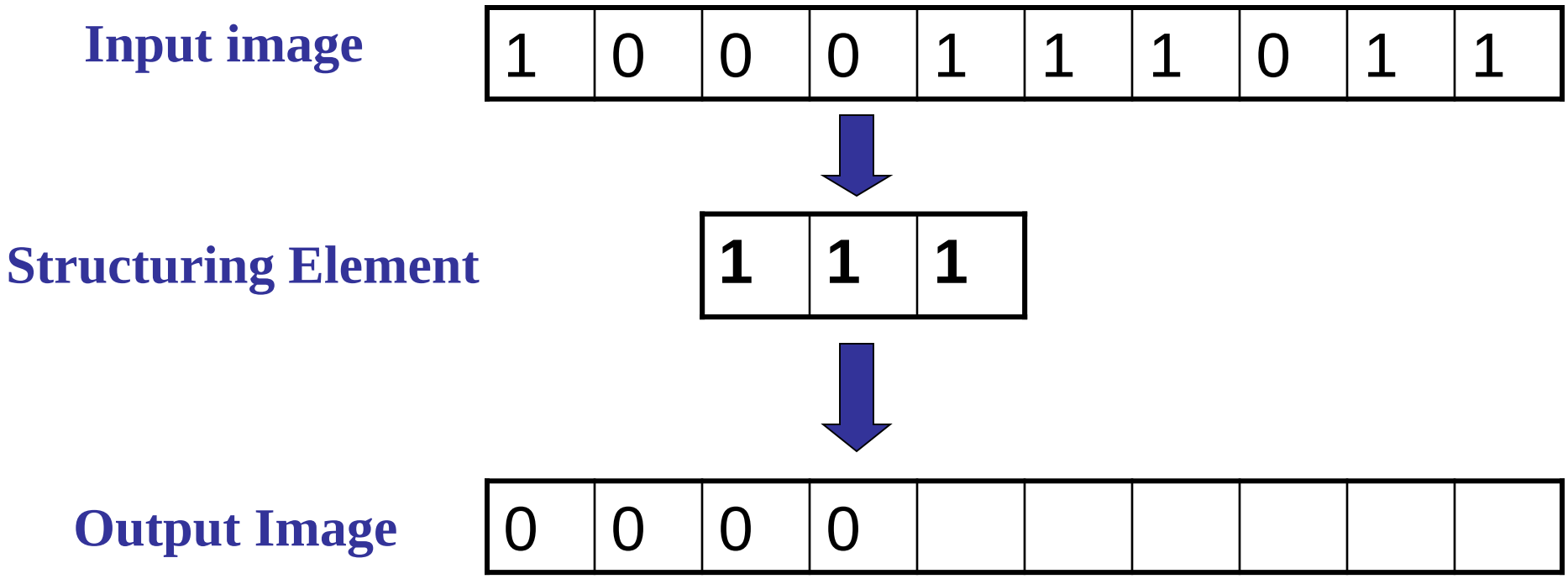
Output Image

0	0								

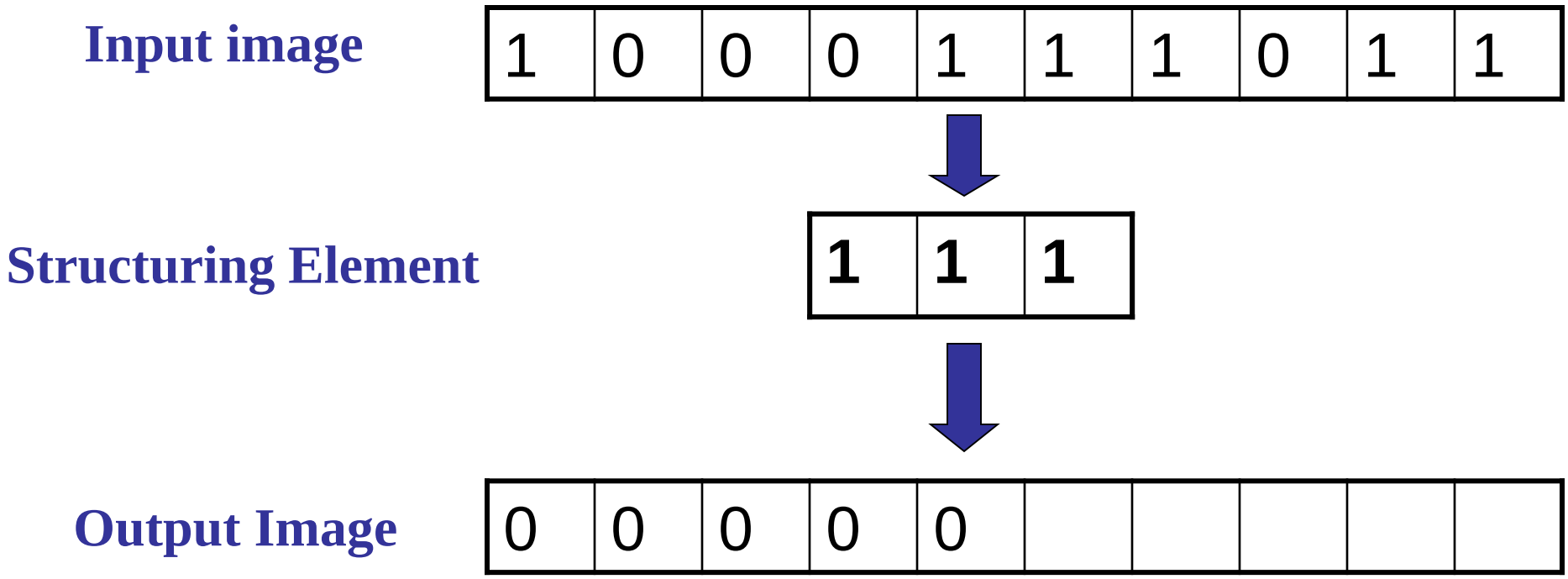
Example for Erosion



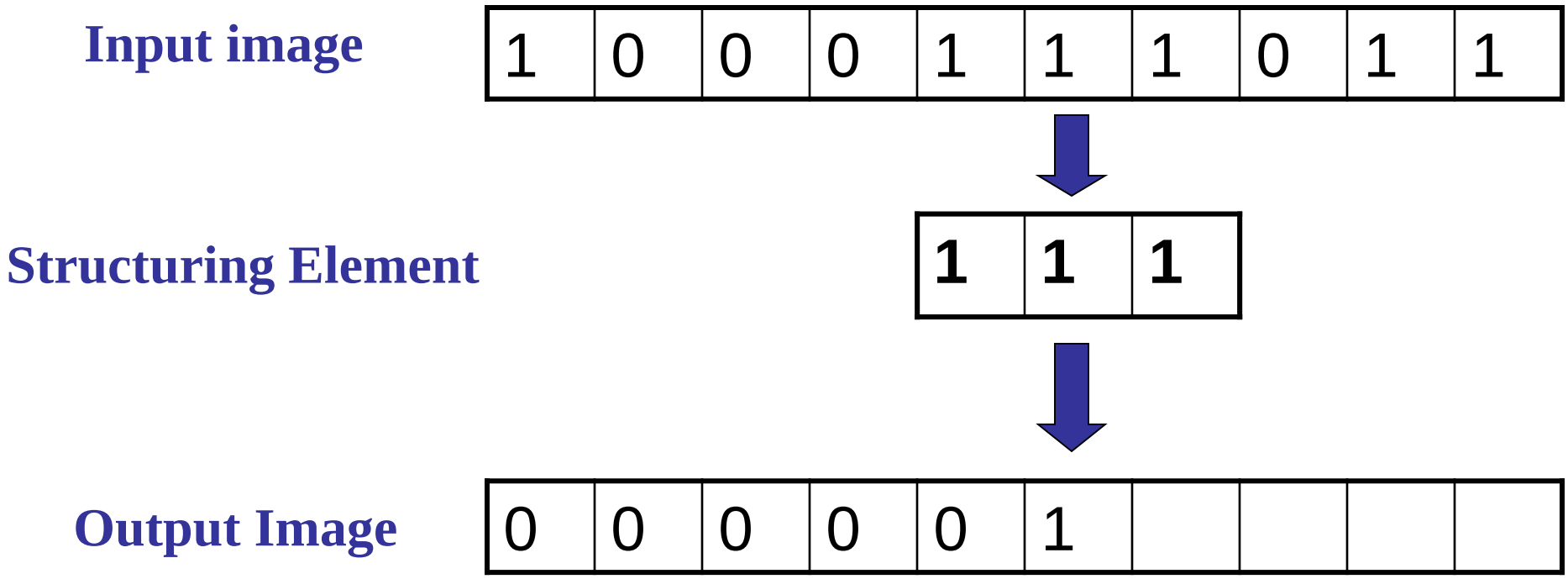
Example for Erosion



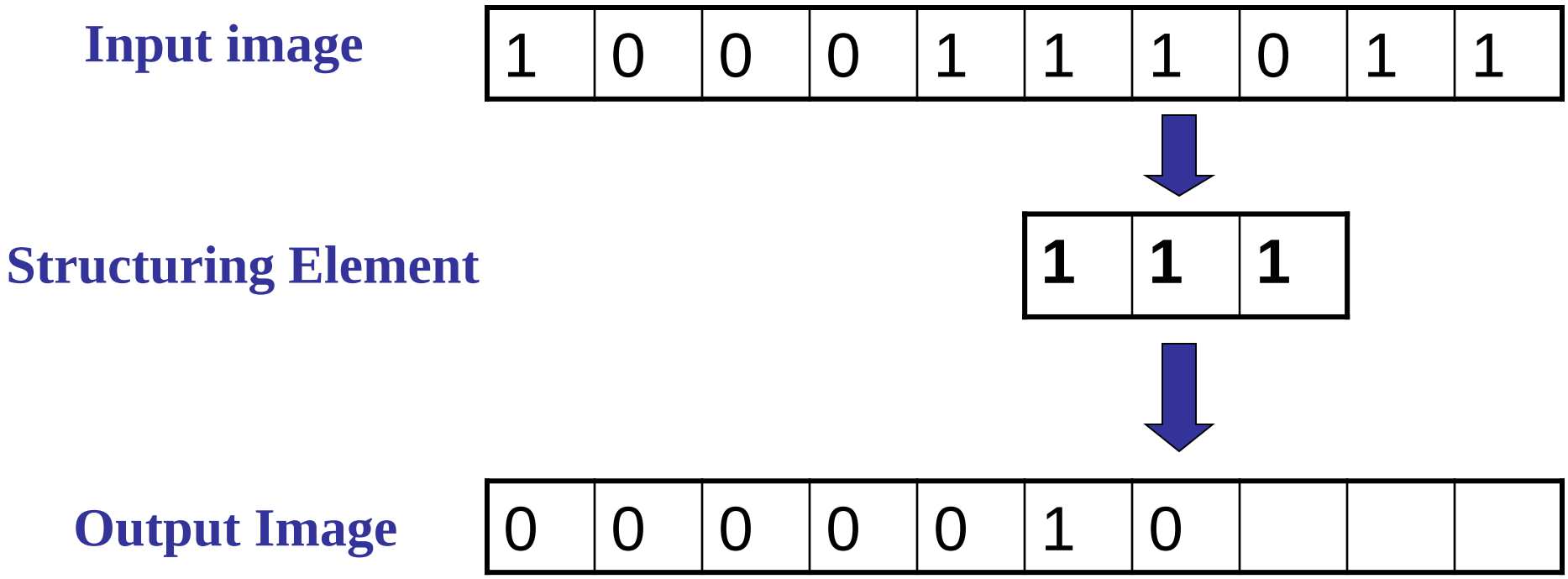
Example for Erosion



Example for Erosion



Example for Erosion



Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

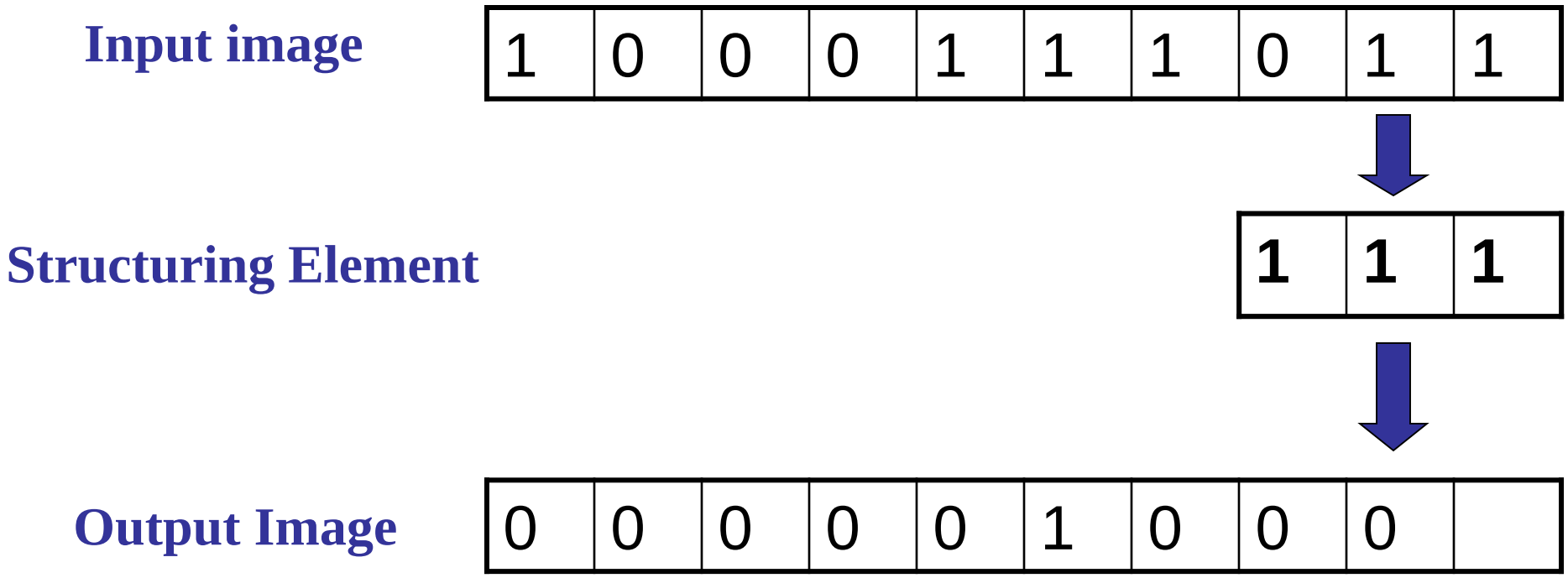
Structuring Element

1	**1**	**1**

Output Image

0	0	0	0	0	1	0	0		

Example for Erosion



Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**

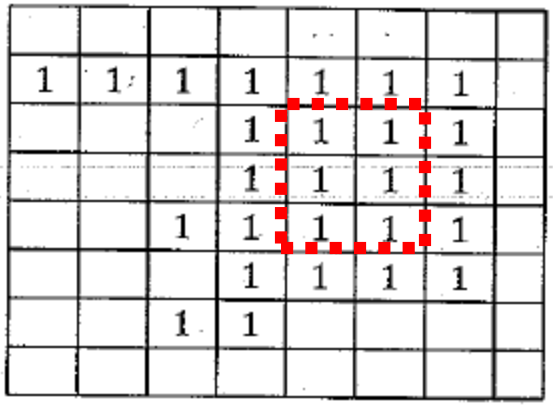
Output Image

0	0	0	0	0	1	0	0	0	1

Note that the object gets smaller

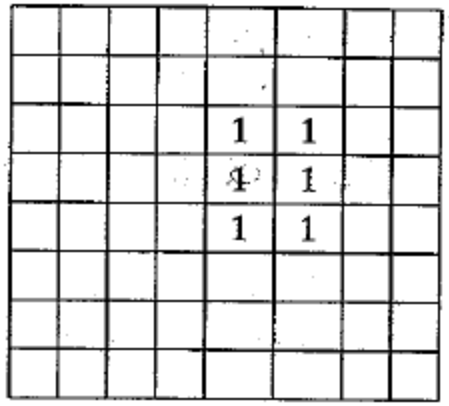
```
>> help imerode
```

2D example for erosion



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

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

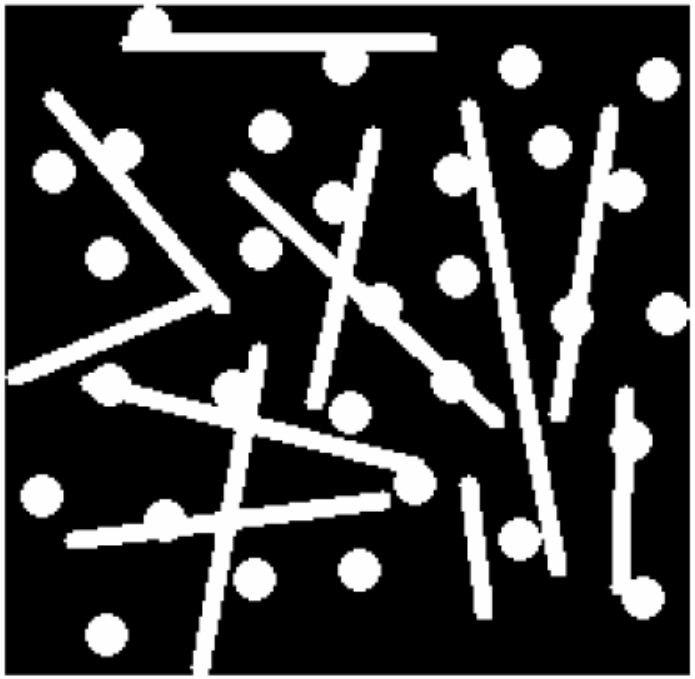


(*d*) Erosion **B** ⊖ **S**

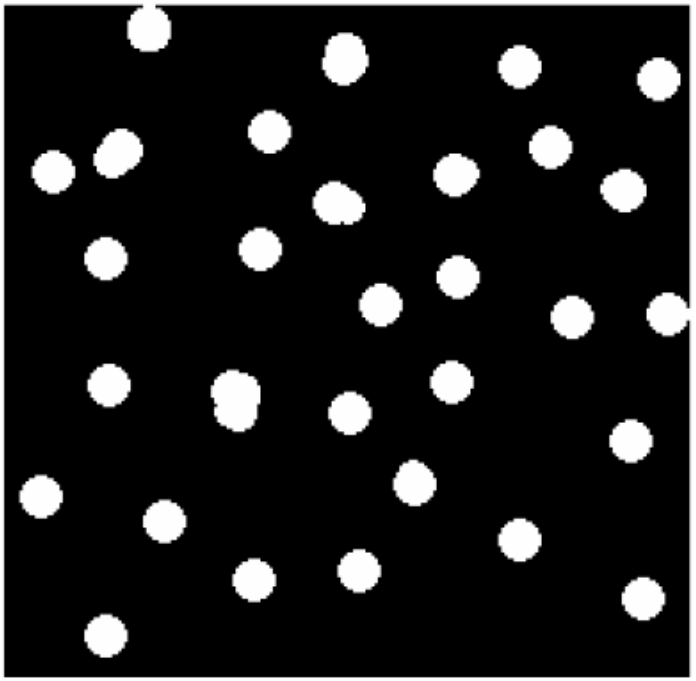
Shapiro & Stockman

Opening

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



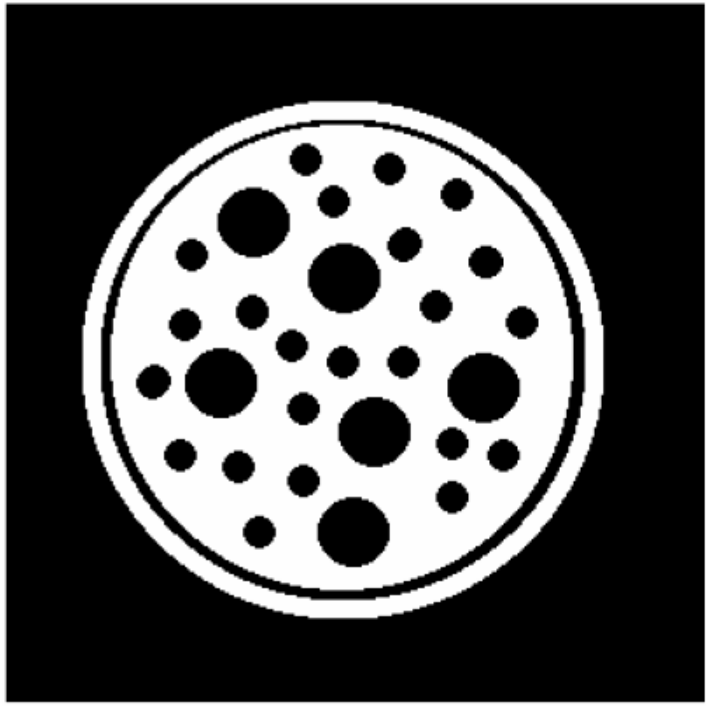
Before opening



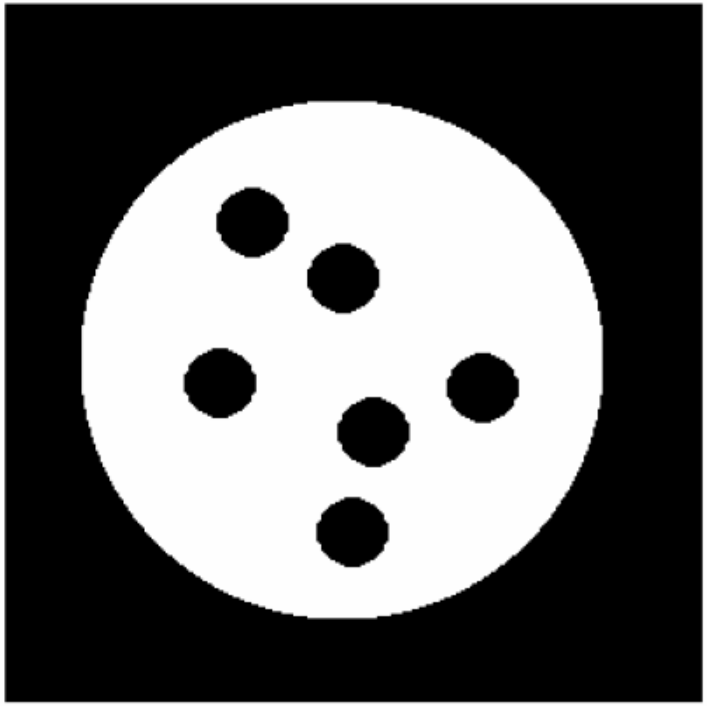
After opening

Closing

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



Before closing

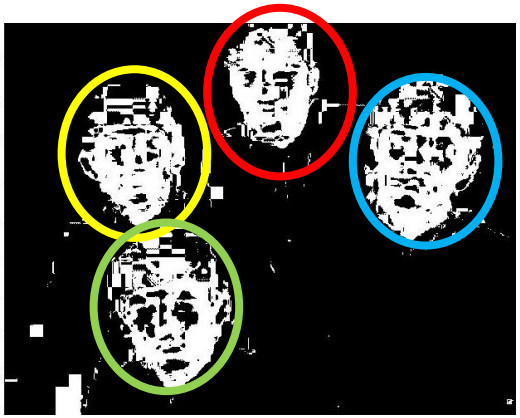


After closing

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



Slide credit: Kristen Grauman

Connected components

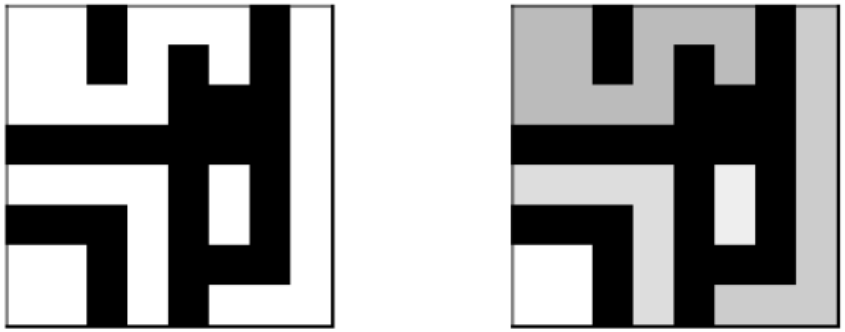
- Identify distinct regions of “connected pixels”

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

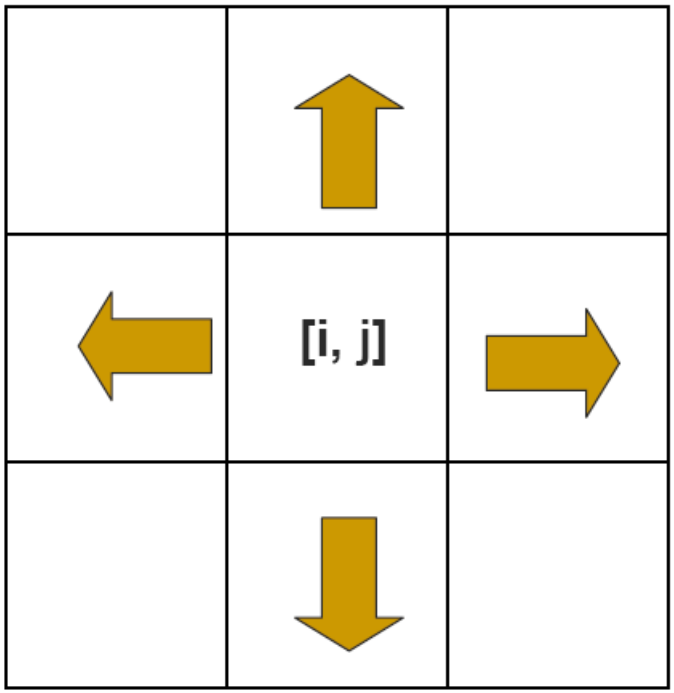


c) binary image and labeling, expanded for viewing

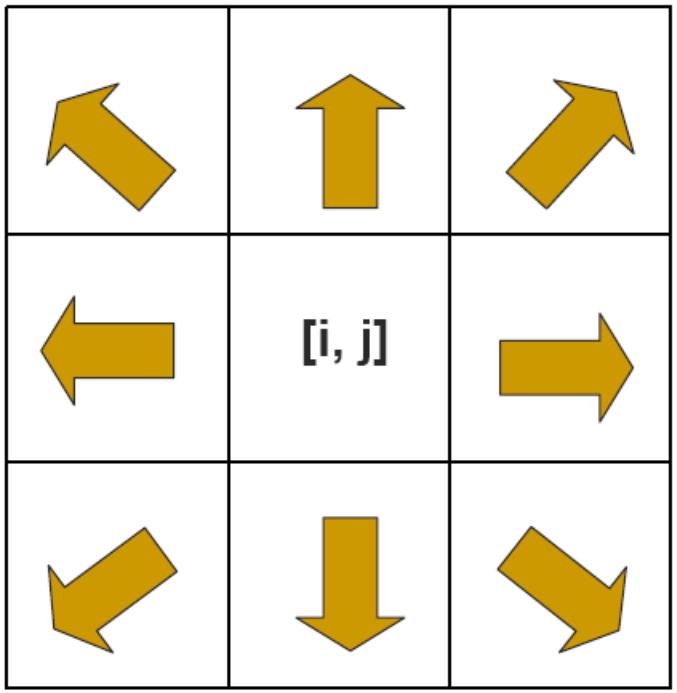
Shapiro and Stockman

Connectedness

- Defining which pixels are considered neighbors



4-connected



8-connected

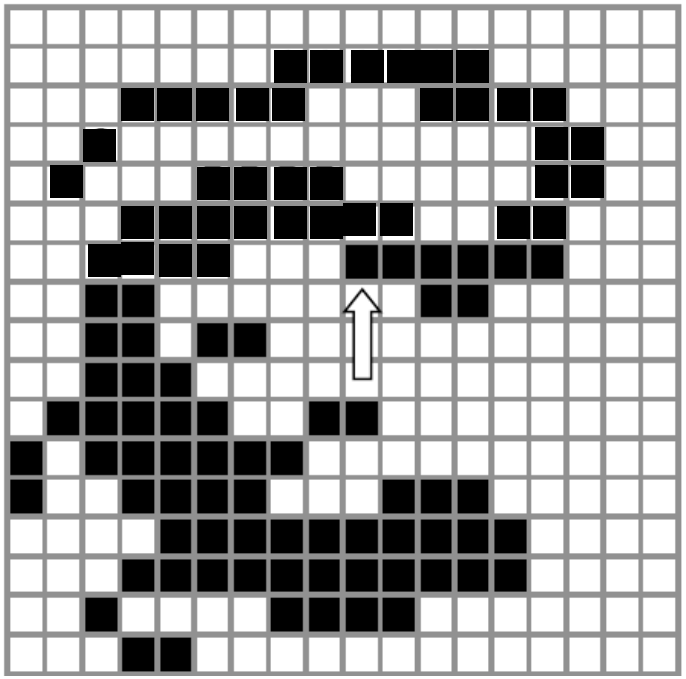
Source: Chaitanya Chandra

Connected components

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

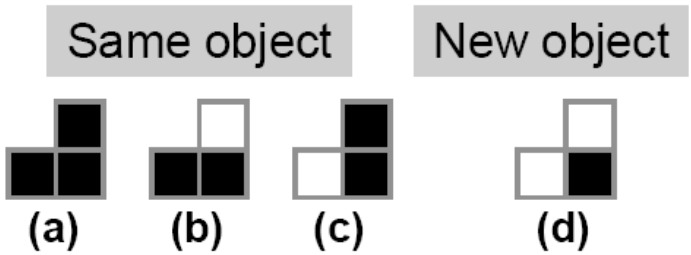
- **Input**: binary image
- **Output**: "label" image, where pixels are numbered per their component

- Note: foreground here is denoted with black pixels.

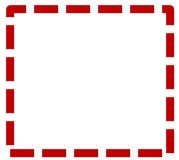


Sequential connected components

- 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).



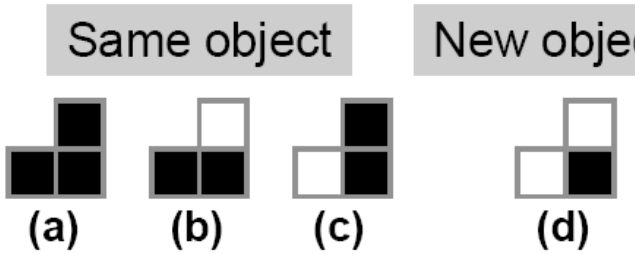
What happens in these cases?

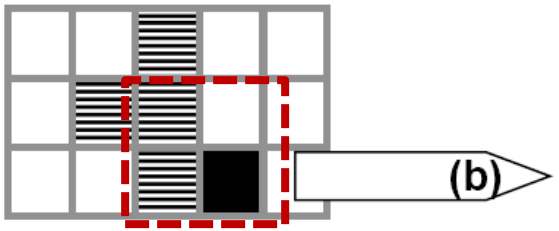


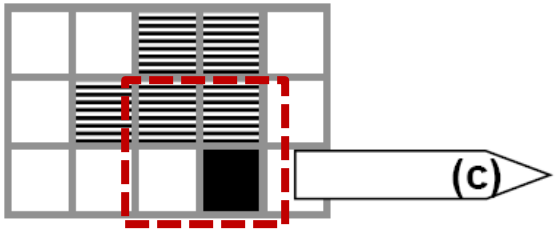
Adapted from J. Neira

Sequential connected components

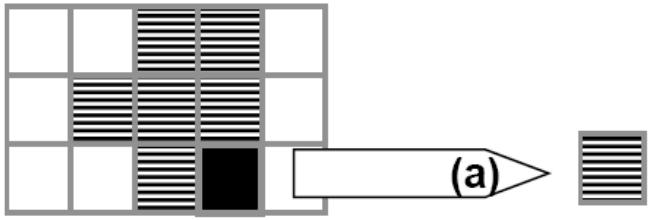
- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).





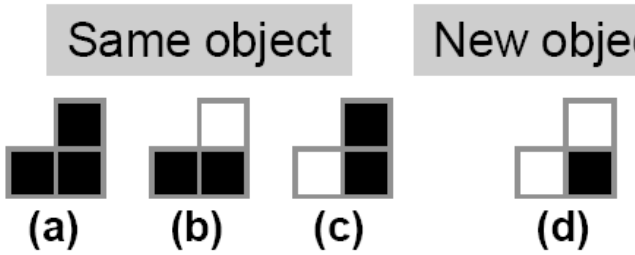


What happens in these cases?

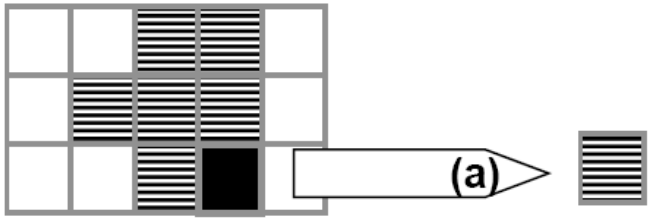


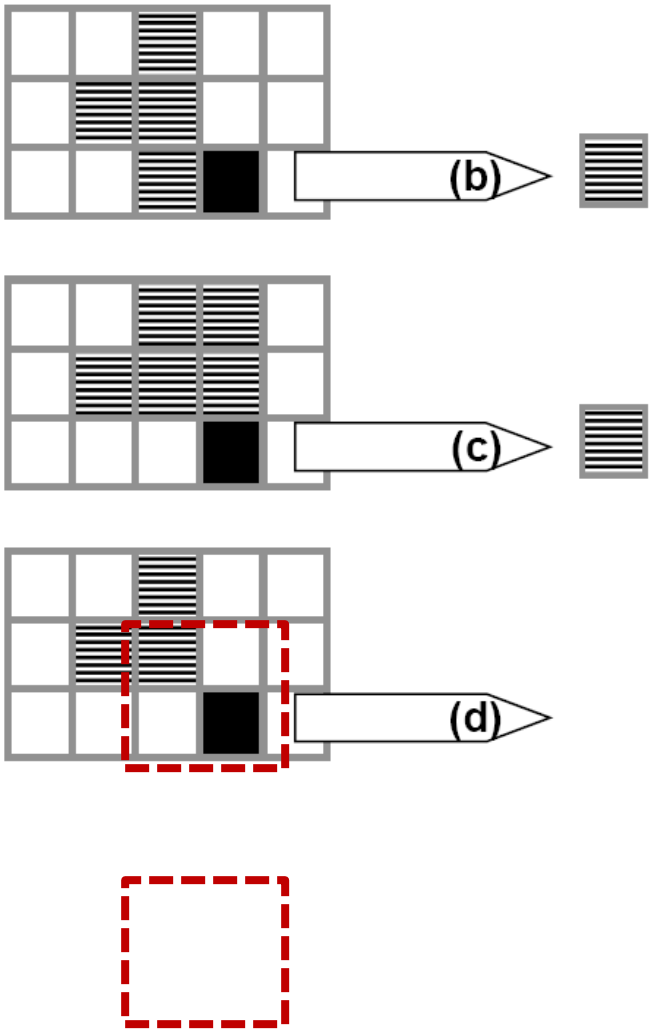
Sequential connected components

- 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).



What happens in these cases?



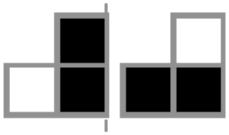


Sequential connected components

- 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.

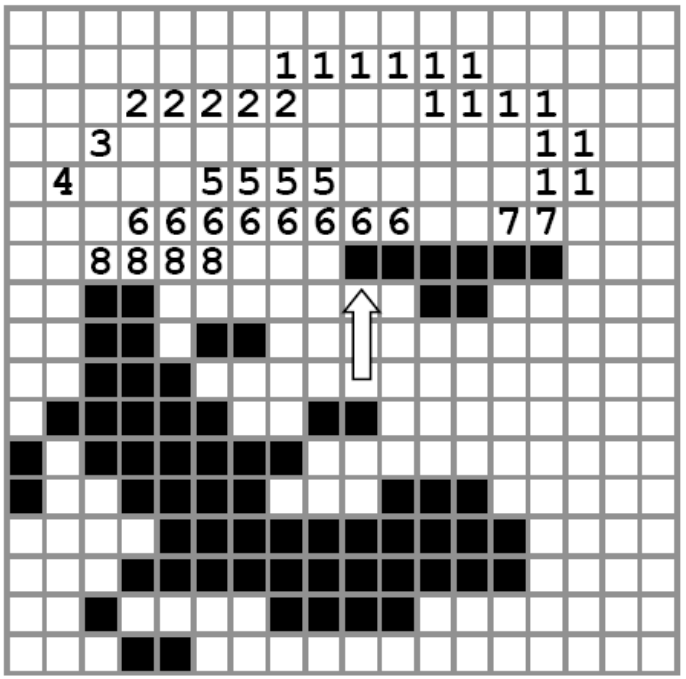
3. If they have different labels:

superior? smallest?

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

4. Otw, assign a new label.

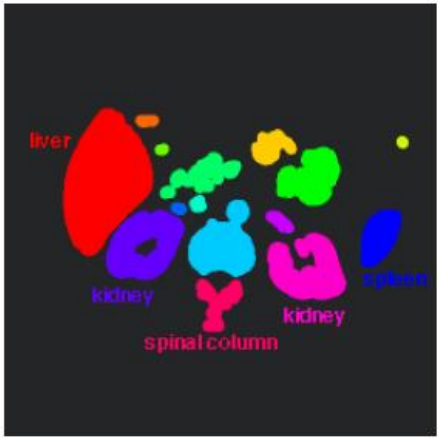
2. More pixels? Go to step 1.



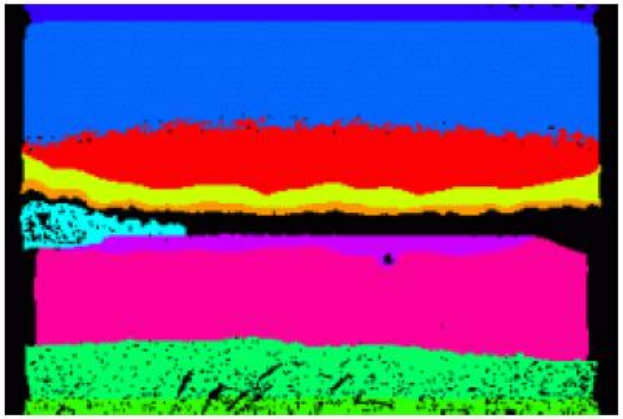

```
{1, 2, 7}
{3}
{4}
{5, 6, 8}
{}
```

- 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

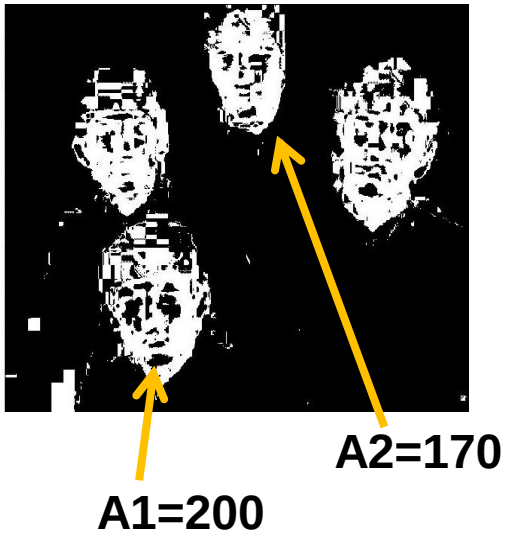


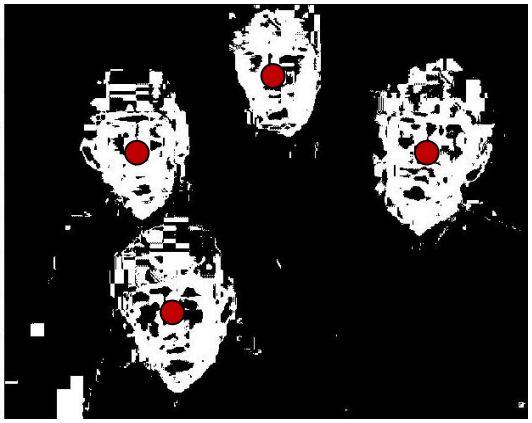
connected components of cluster labels

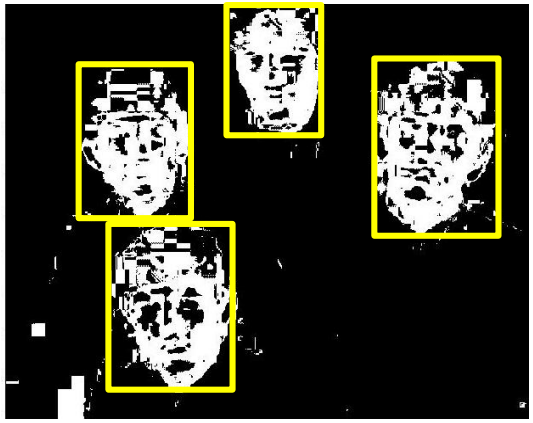
Slide credit: Pinar Duygulu

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)







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

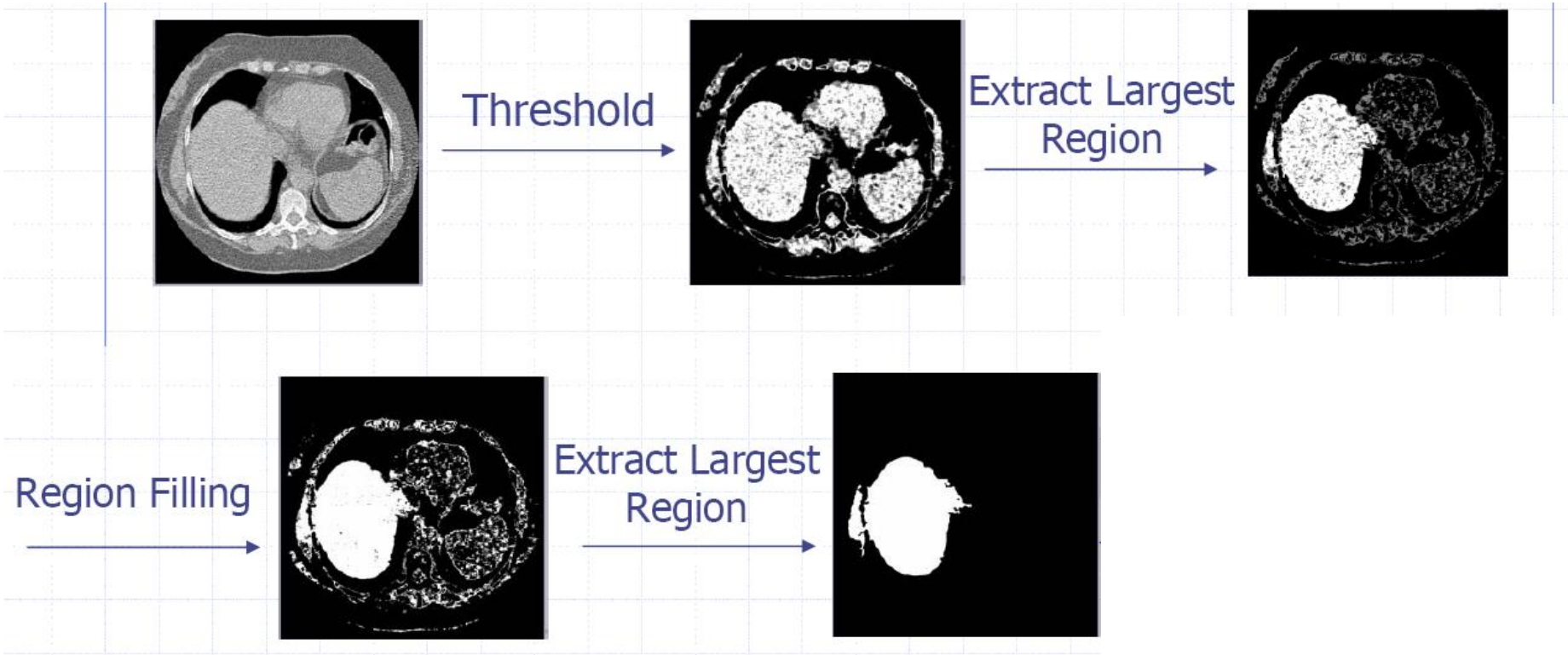
Matlab

- **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

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

New concepts today

- Gradients -> edge maps
- Template matching
- Chamfer distance
- Distance transform
- Erosion/dilation for binary images
- Connected components
- Region properties

Assignments

- 了解概念：二值图像连通区域。编写“种子填充法” 标记二值图像的连通区域的伪代码，并讨论其算法复杂度。

- 编程实现二值图像连通区域标记，并计算其面积等统计量。