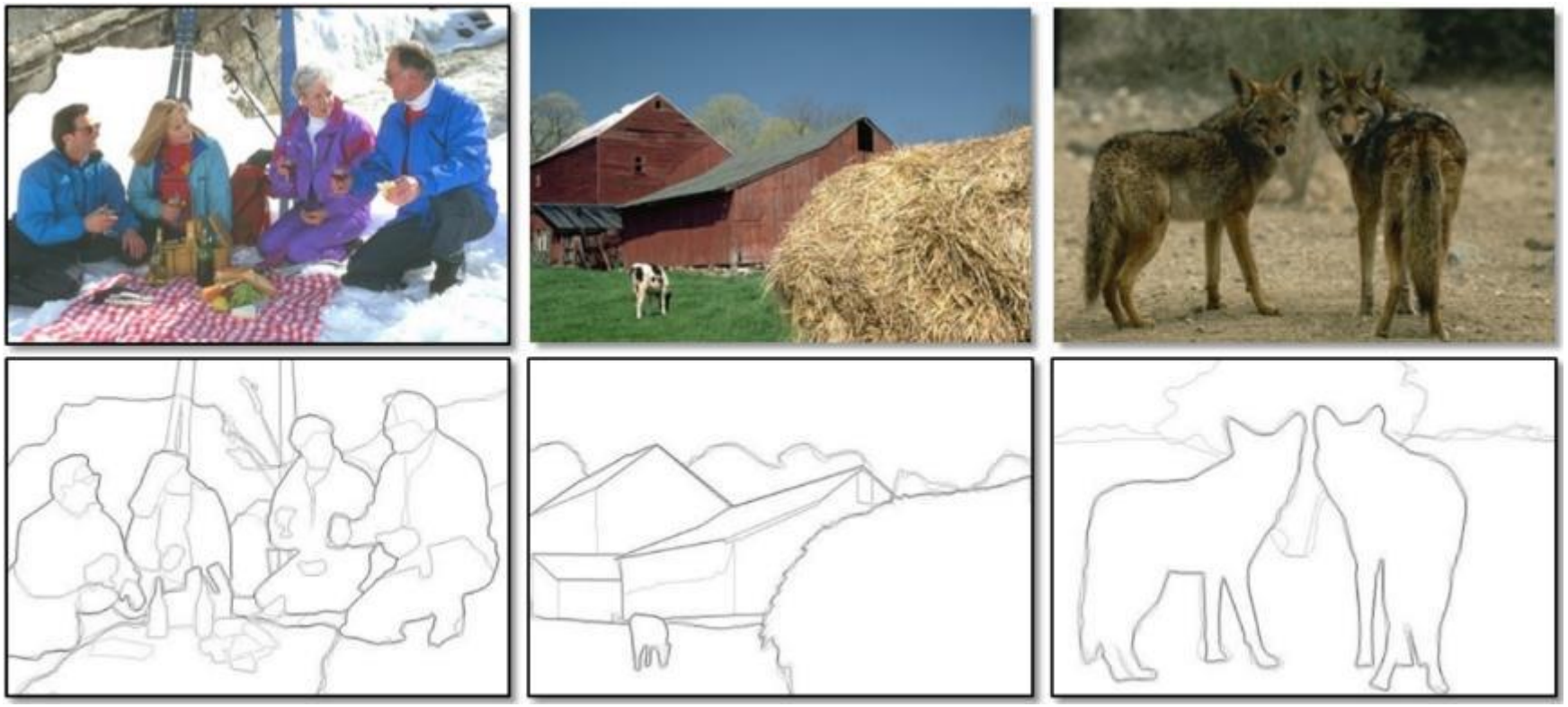
Edge Detection



Computer Vision

Jia-Bin Huang, Virginia Tech

Many slides from D. Hoiem and K. Grauman

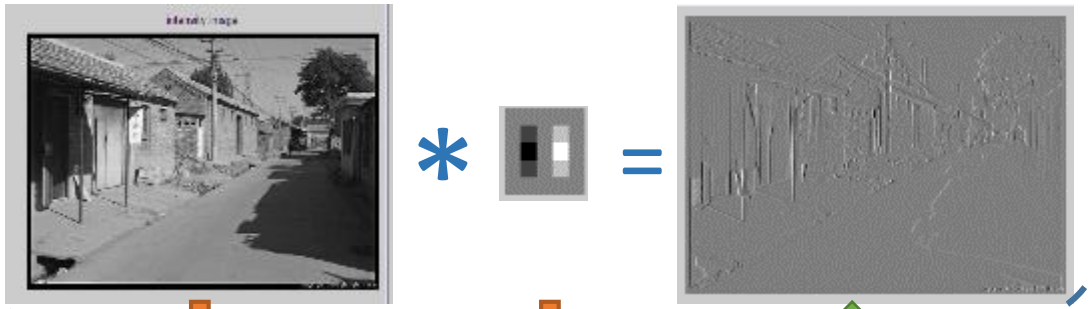
Administrative Stuffs

- HW 1 posted, due 11:59 PM Sept 19
  - Submission through Canvas
- Questions about HW?
  - Ask me/TA after classes
  - Post questions on Piazza (no emails)
  - Attend office hours

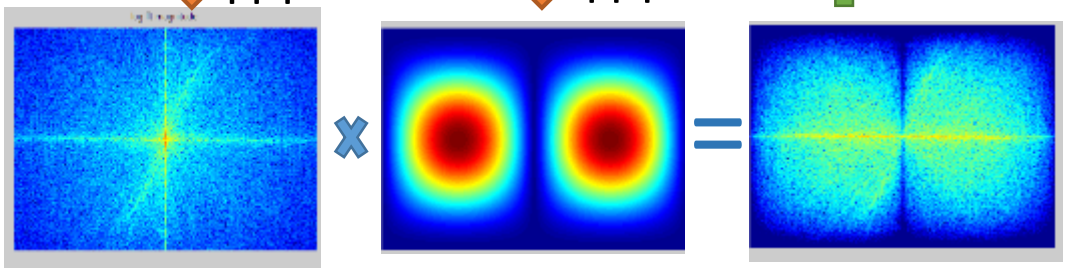
Previous three classes: Image Filtering

Spatial domain

- Smoothing, sharpening, measuring texture



FFT FFT Inverse FFT



Frequency domain

- Denoising, sampling, image compression

Template matching

- Find a template in an image

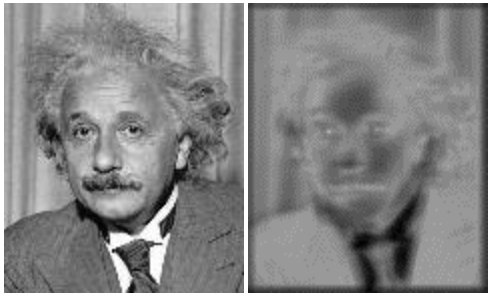


Image pyramid

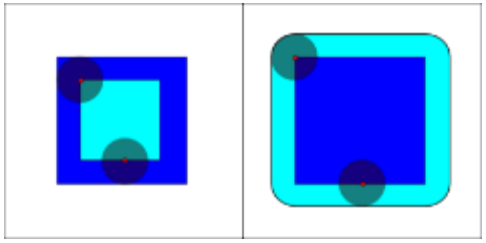
- Coarse-to-fine search
- Multi-scale detection

Today's class

- Detecting edges

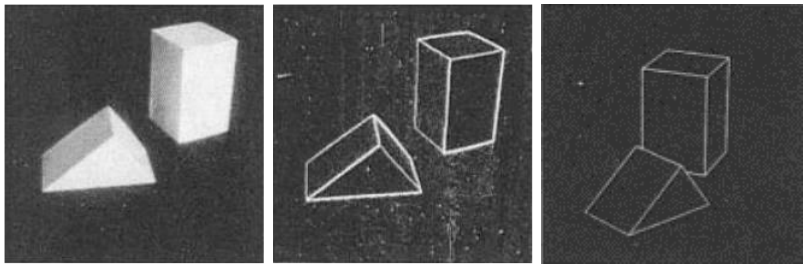
- Finding straight lines

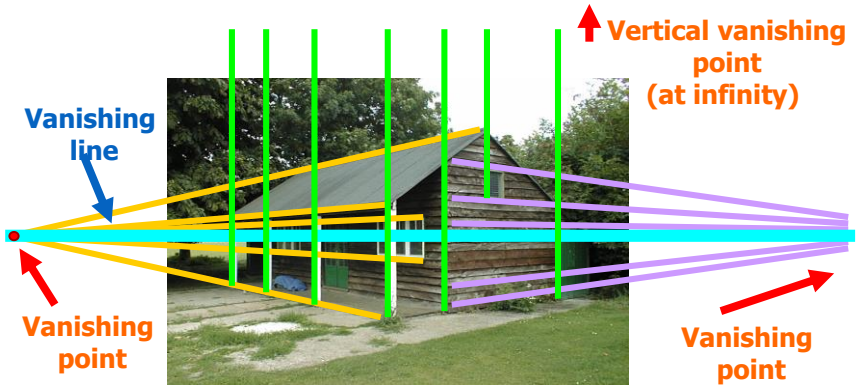
- Binary image analysis



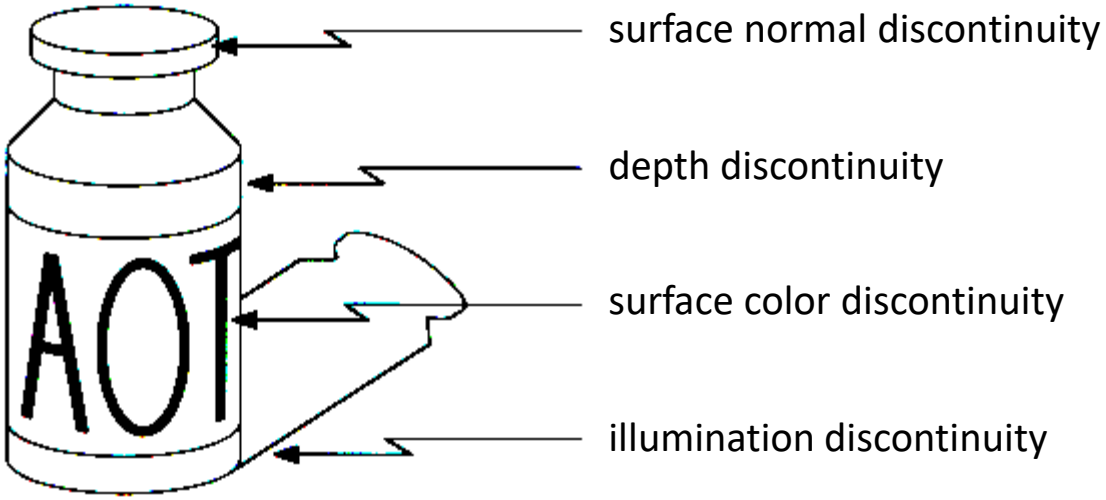
Why finding edges is important

- Cues for 3D shape
- Group pixels into objects or parts
- Guiding interactive image editing
- Recover geometry and viewpoint





Origin of Edges

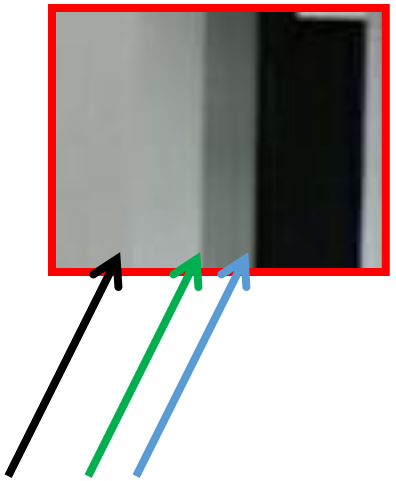


Edges are caused by a variety of factors

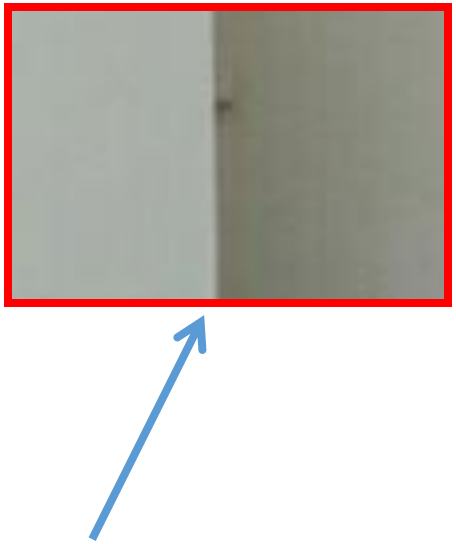
Source: Steve Seitz

Closeup of edges

Closeup of edges



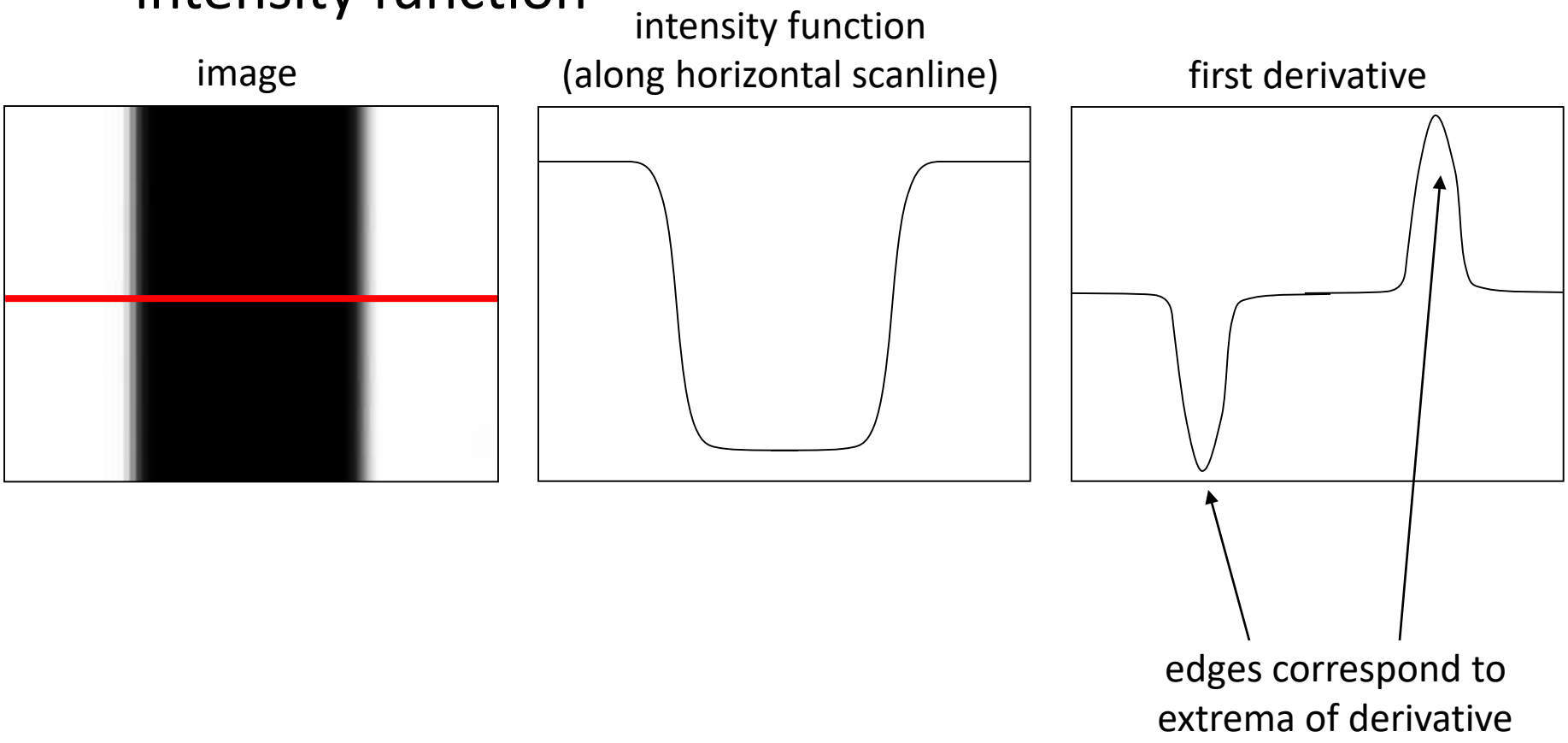
Closeup of edges



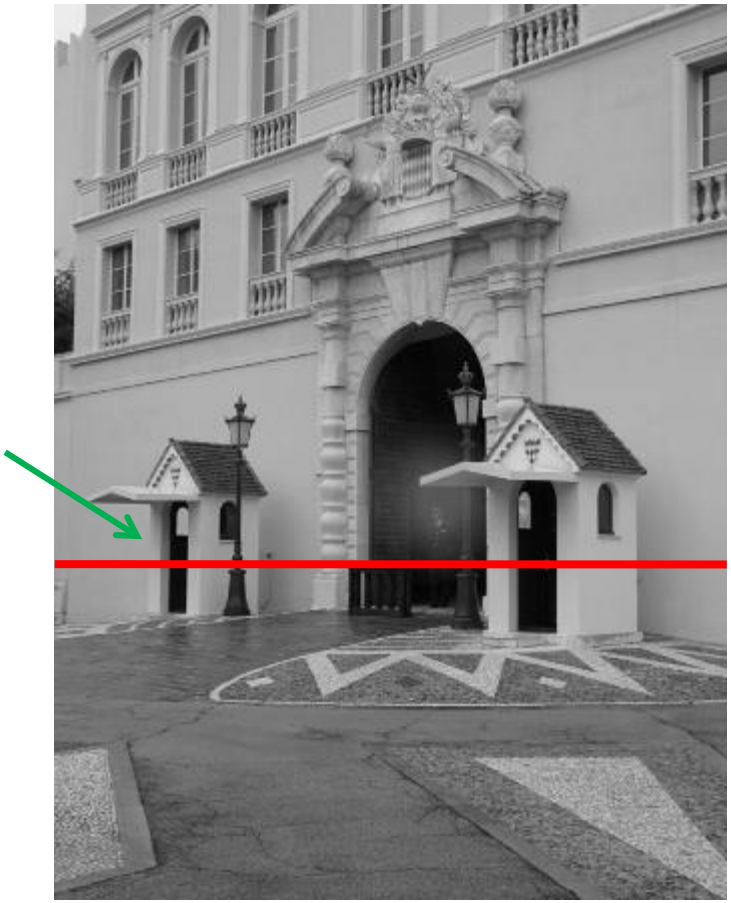
Closeup of edges

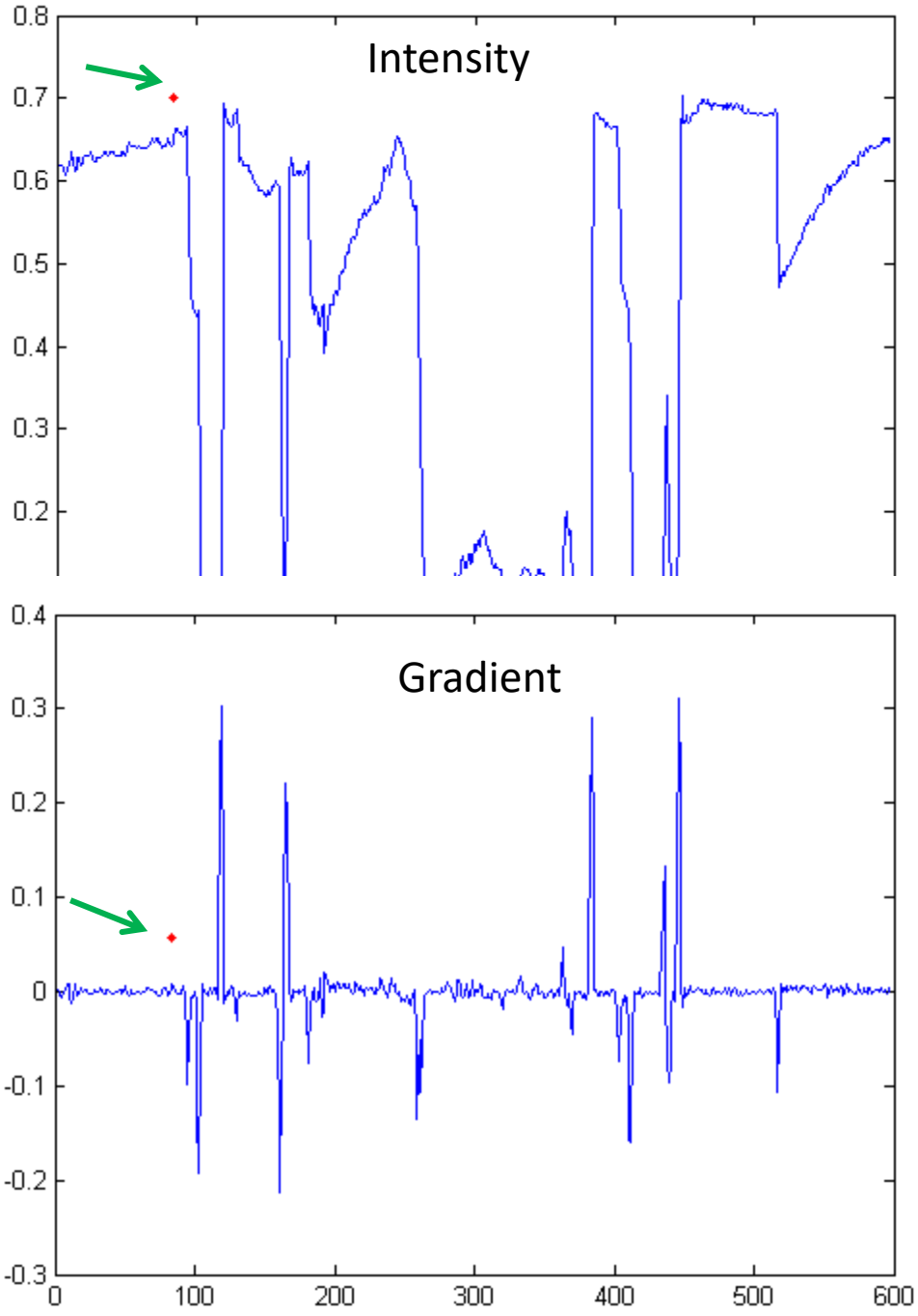
Characterizing edges

- An edge is a place of rapid change in the image intensity function

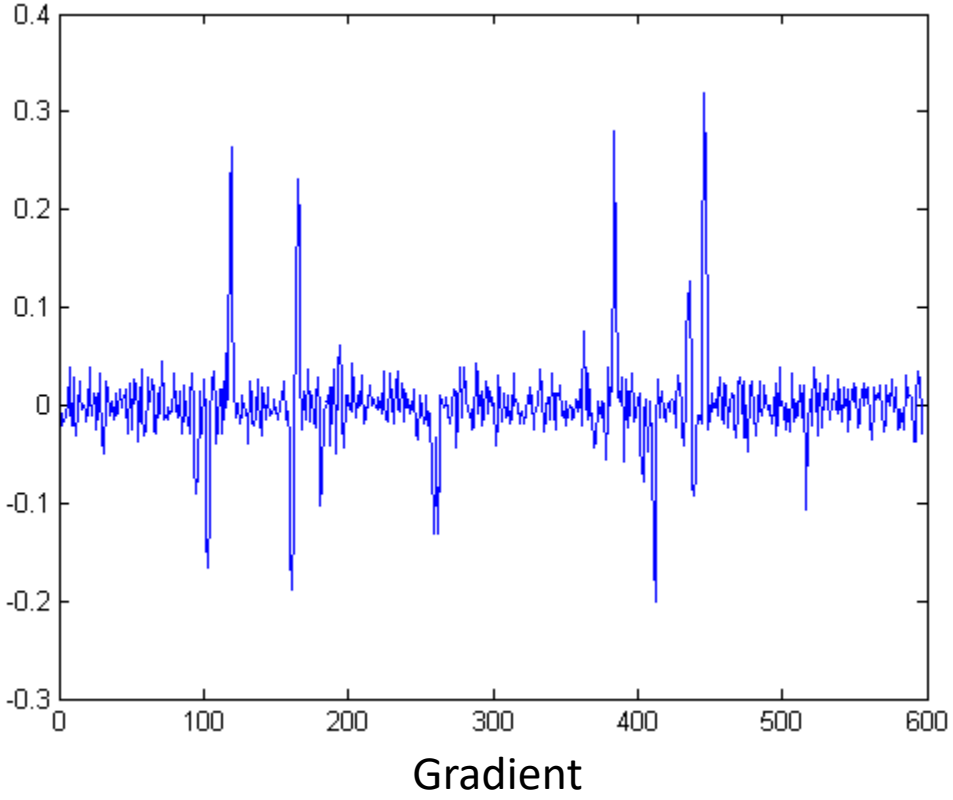


Intensity profile



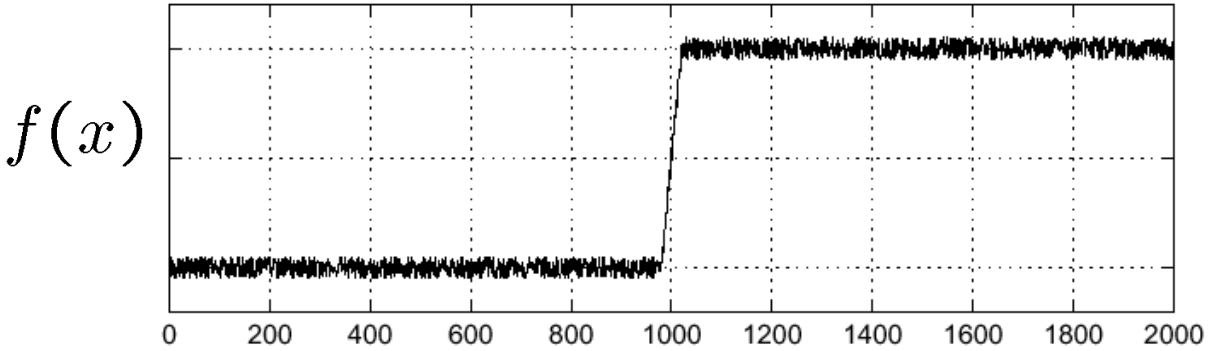


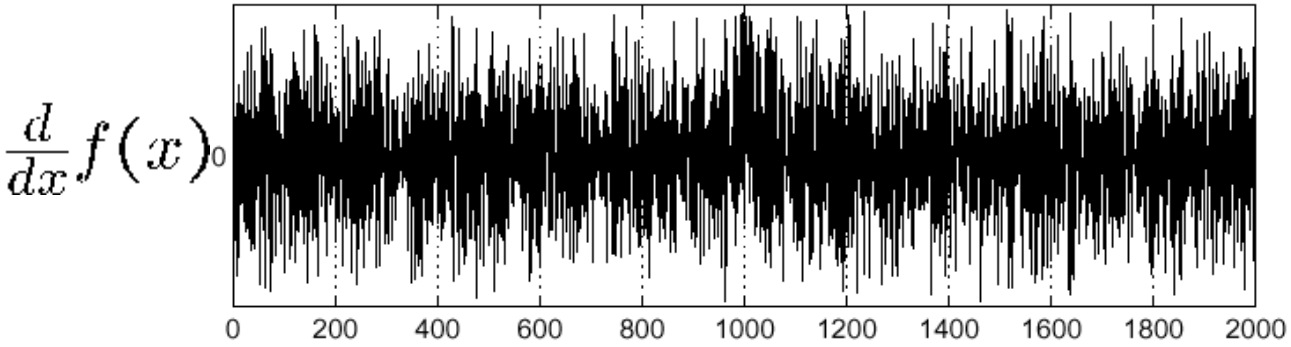
With a little Gaussian noise



Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

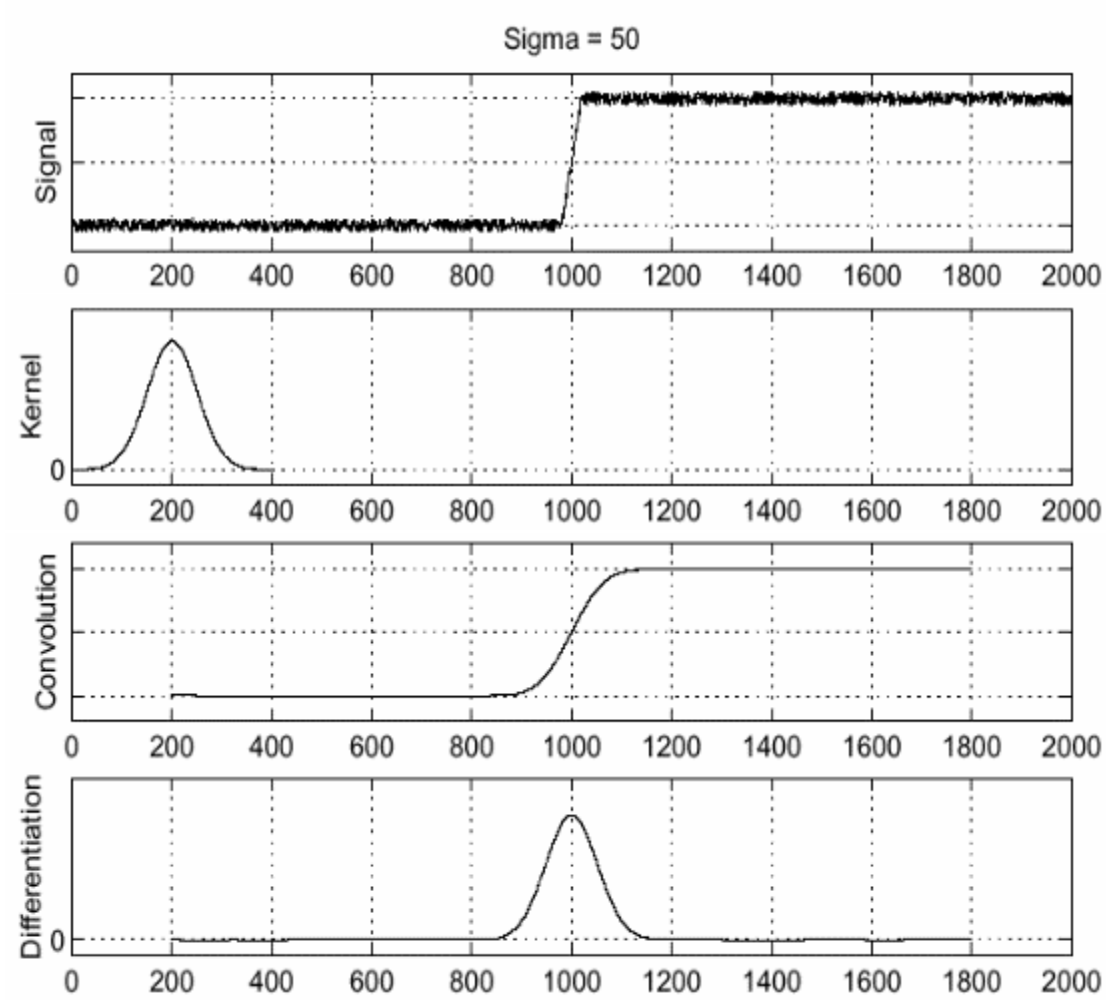
Source: S. Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

Solution: smooth first



f

g

$f * g$

$\frac{d}{dx}(f * g)$

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

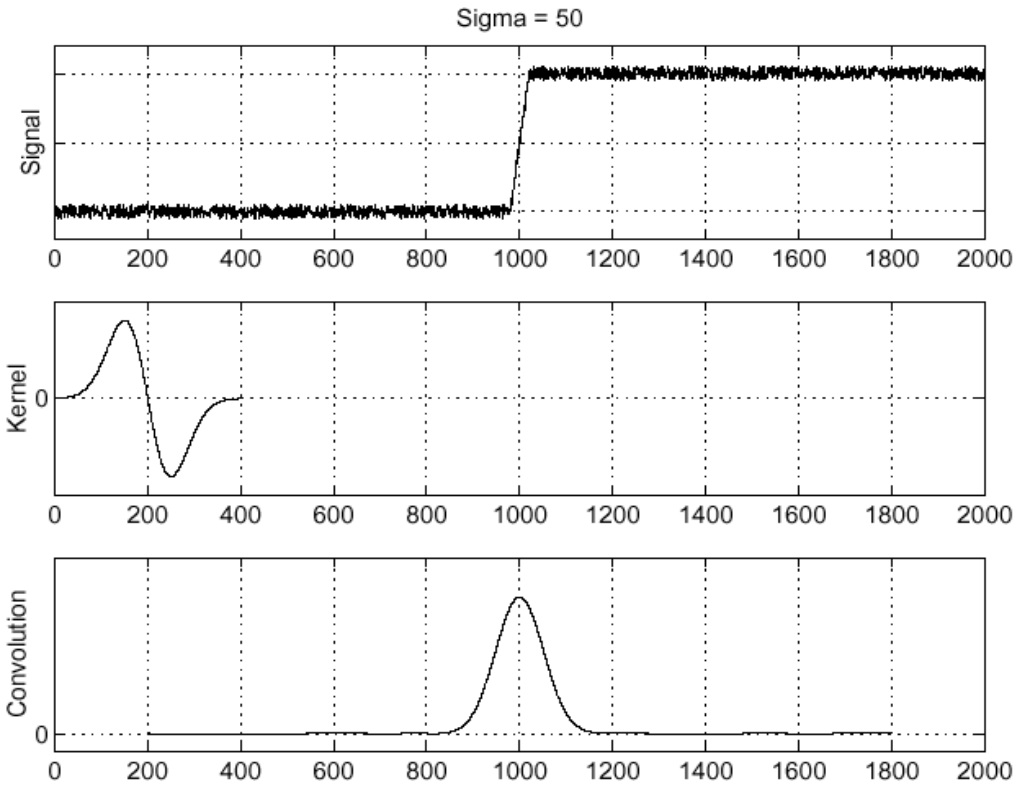
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

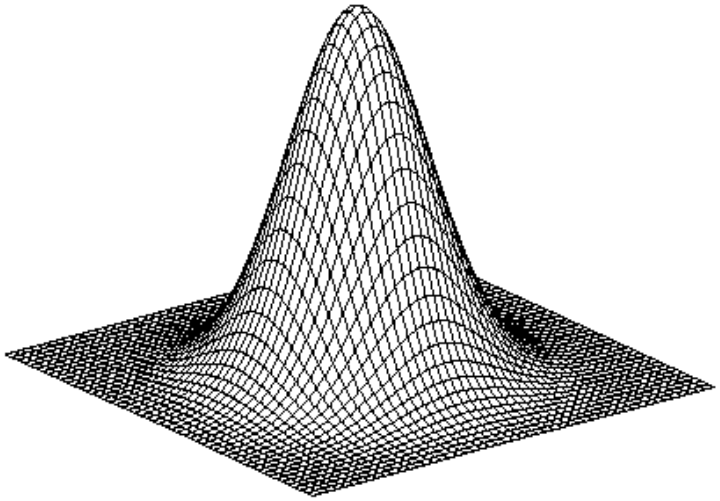
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

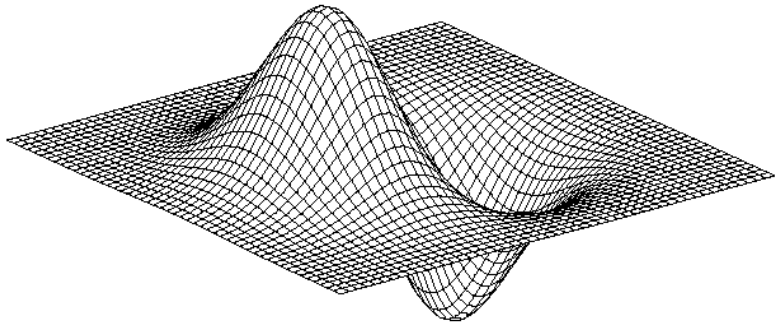


Source: S. Seitz

Derivative of Gaussian filter

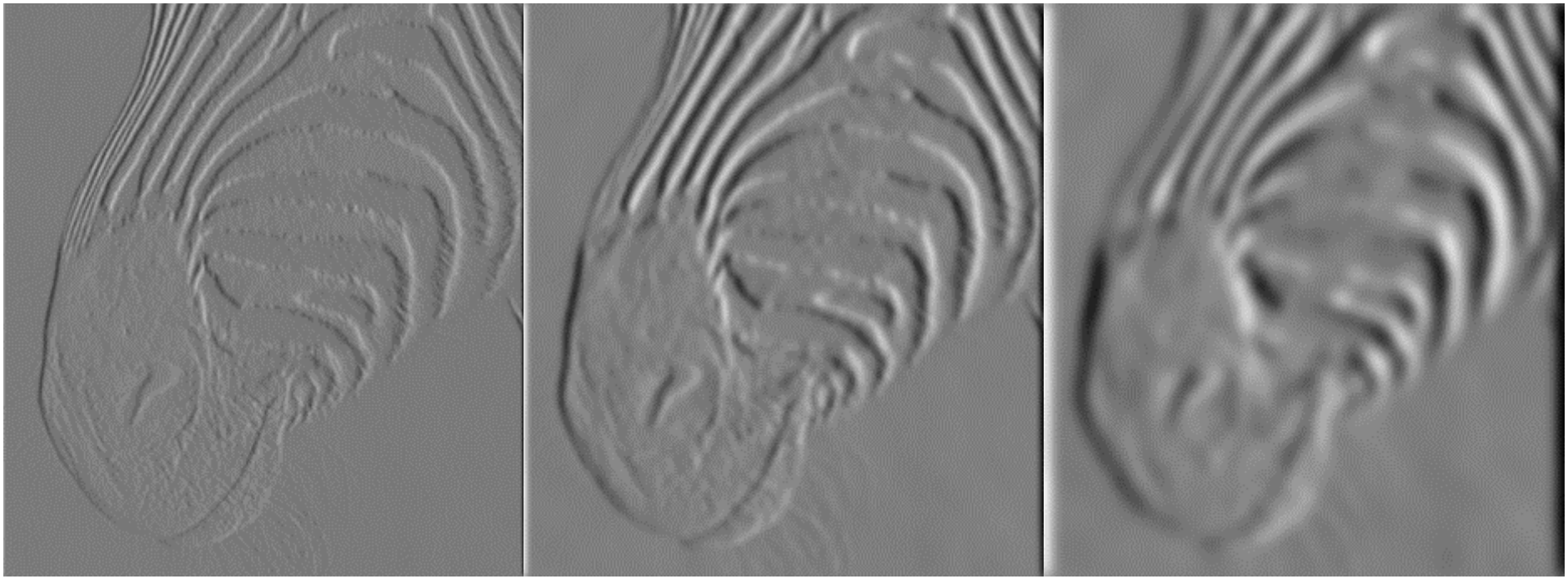


* [1 0 -1] =



- Is this filter separable?

Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Designing an edge detector

- Criteria for a good edge detector:
  - **Good detection:**
    - find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - detect edges as close as possible to the true edges
    - return one point only for each true edge point
- Cues of edge detection
  - Differences in color, intensity, or texture across the boundary
  - Continuity and closure
  - High-level knowledge

Source: L. Fei-Fei

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

A computational approach to edge detection
J Canny - IEEE Transactions on pattern analysis and machine …, 1986 - ieeexplore.ieee.org
Abstract-This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the ...
Cited by 24439 Related articles All 24 versions Import into BibTeX Save More

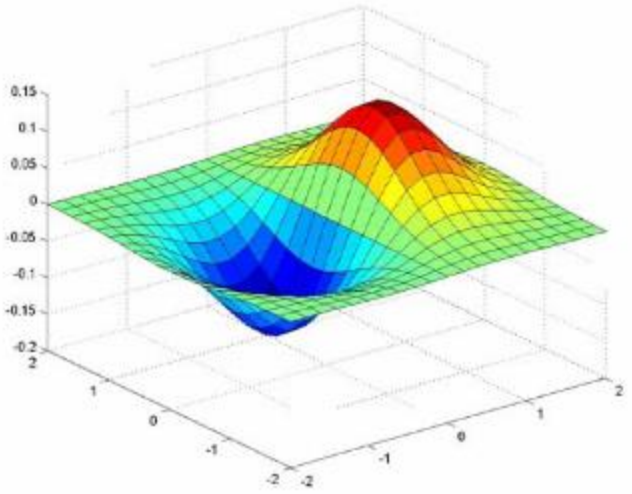
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

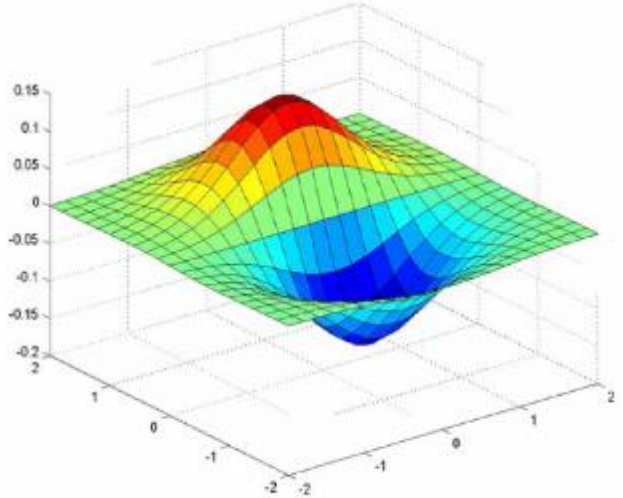
Example

input image (“Lena”)

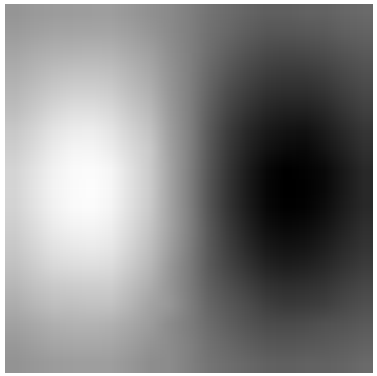
Derivative of Gaussian filter

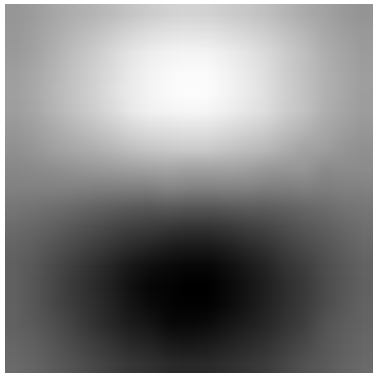


x-direction

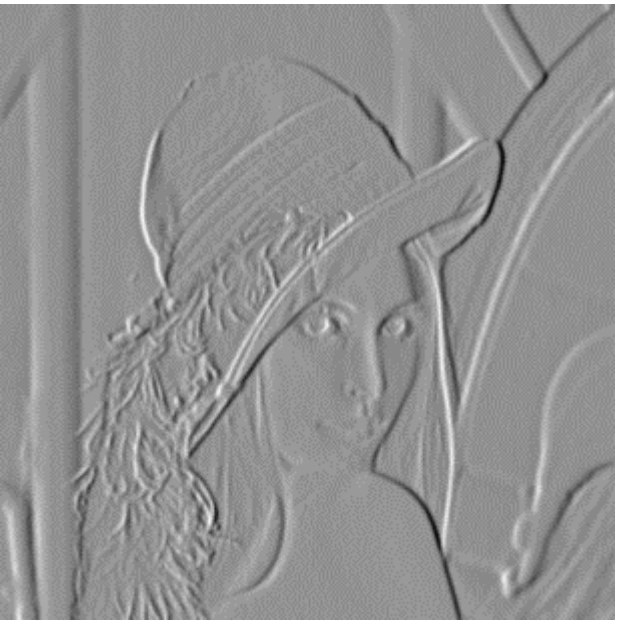


y-direction





Compute Gradients (DoG)



X-Derivative of Gaussian

Y-Derivative of Gaussian

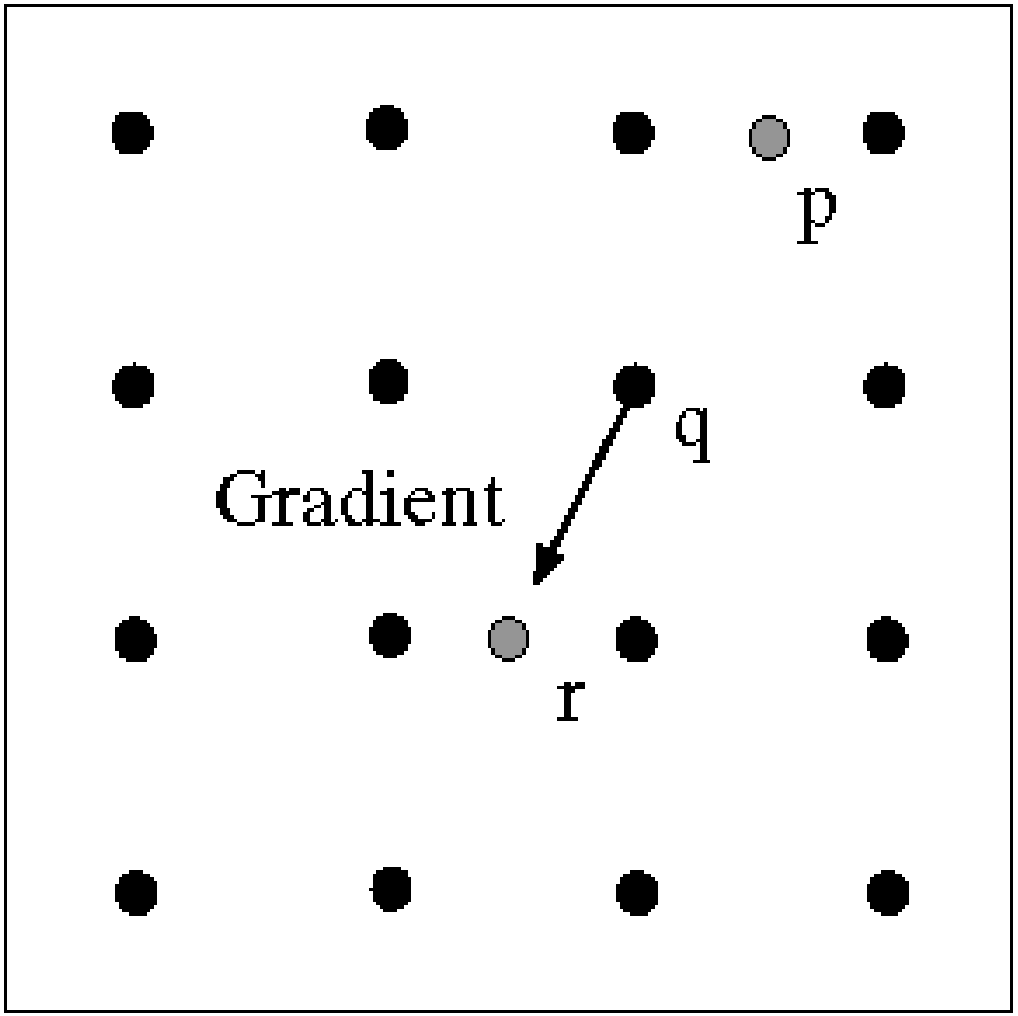
Gradient Magnitude

Get Orientation at Each Pixel

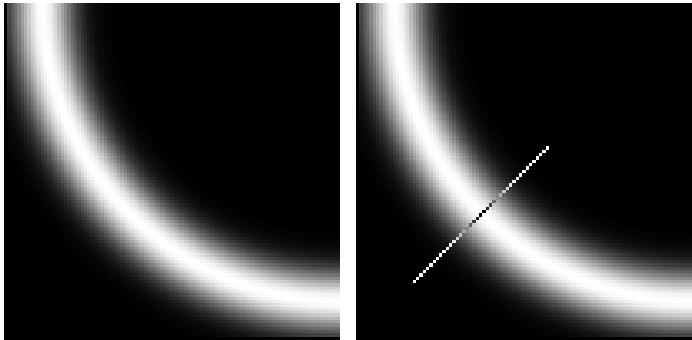
- Threshold at minimum level
- Get orientation

theta = atan2(-gy, gx)

Non-maximum suppression for each orientation



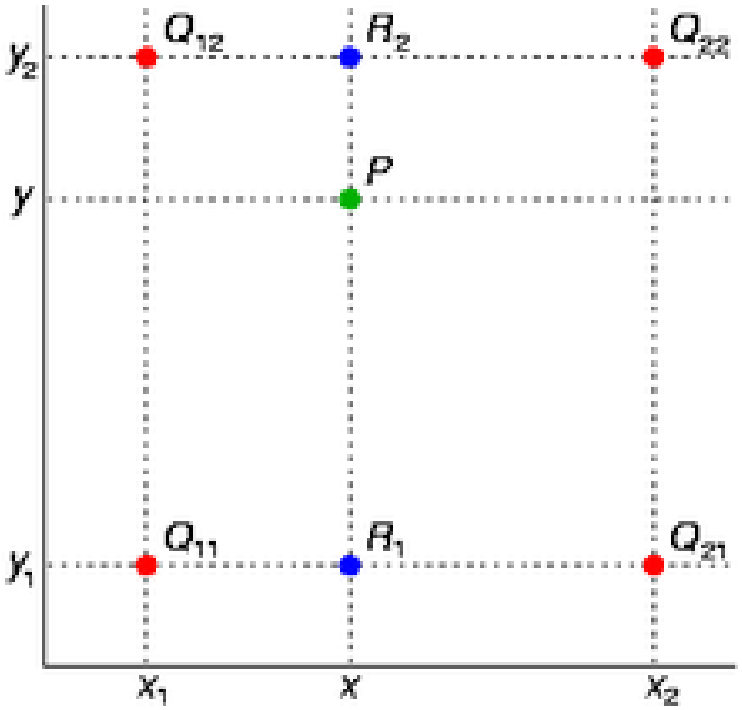
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

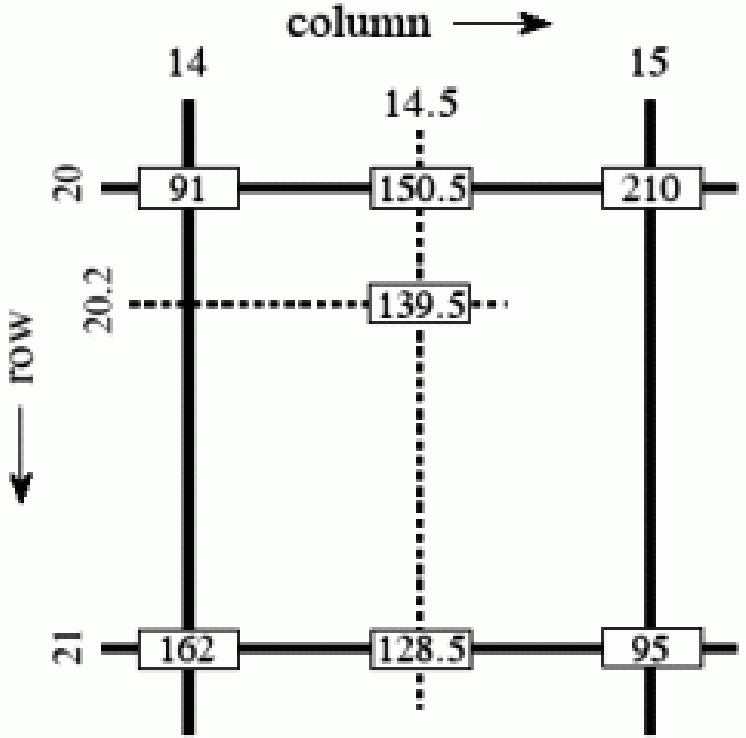


Source: D. Forsyth

Bilinear Interpolation

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$





http://en.wikipedia.org/wiki/Bilinear_interpolation

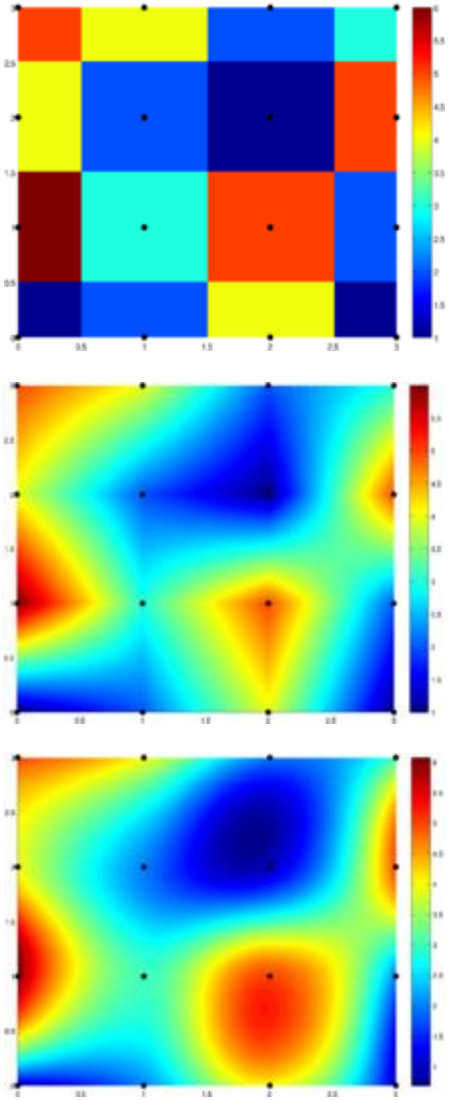
Sidebar: Interpolation options

- imx2 = imresize(im, 2, interpolation_type)

- ‘nearest’
  - Copy value from nearest known
  - Very fast but creates blocky edges

- ‘bilinear’
  - Weighted average from four nearest known pixels
  - Fast and reasonable results

- ‘bicubic’ (default)
  - Non-linear smoothing over larger area
  - Slower, visually appealing, may create negative pixel values



Examples from http://en.wikipedia.org/wiki/Bicubic_interpolation

Before Non-max Suppression

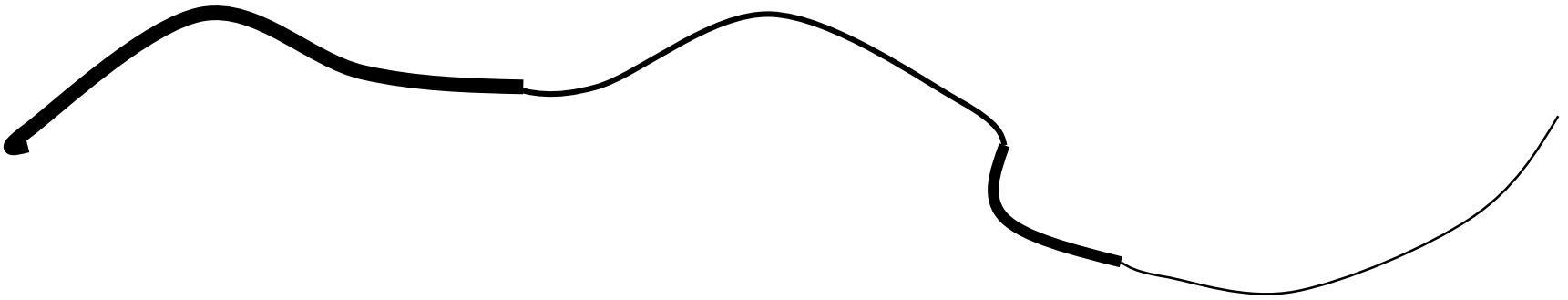
After non-max suppression

Hysteresis thresholding

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

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Final Canny Edges

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (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')`

Source: D. Lowe, L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

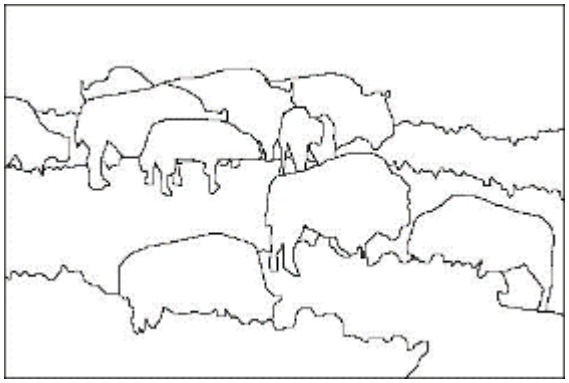
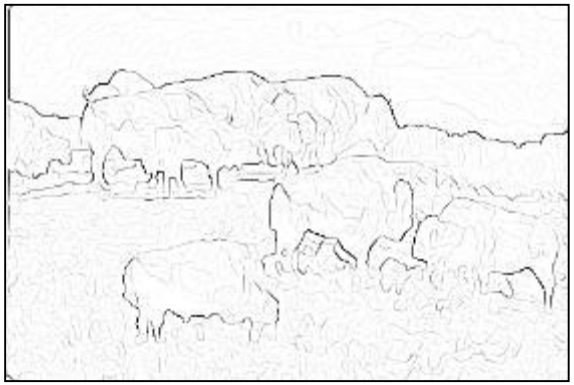
The choice of σ depends on desired behavior

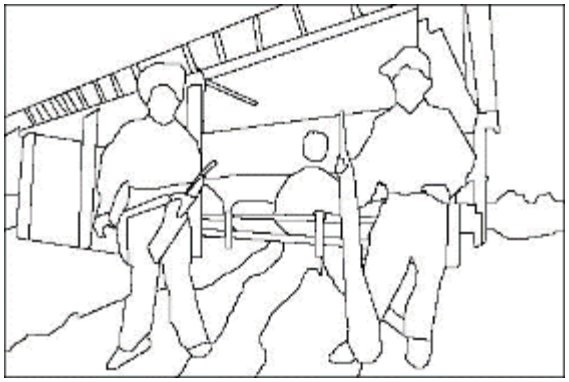
- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

Learning to detect boundaries

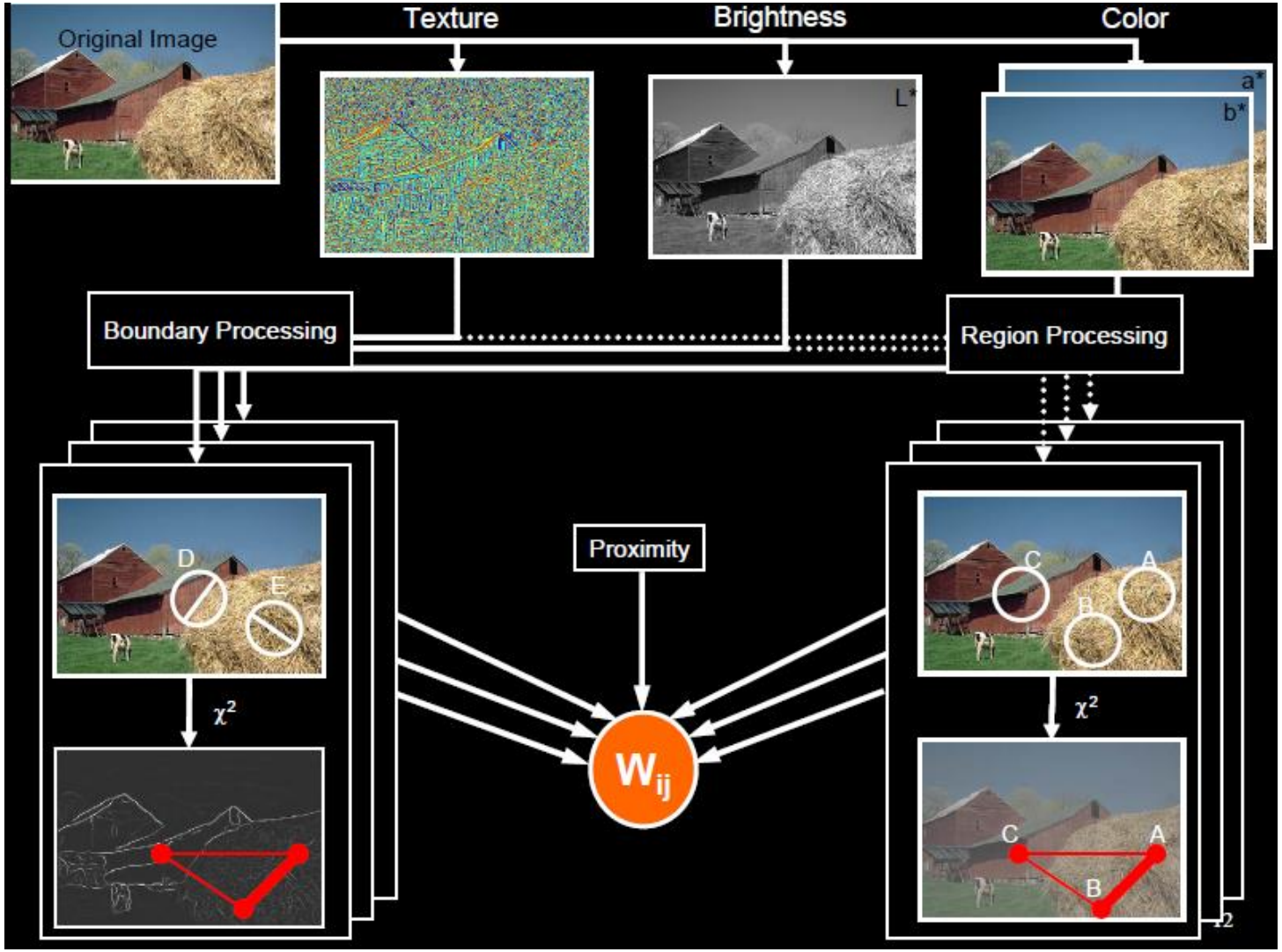
image | human segmentation | gradient magnitude




- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

pB boundary detector



Martin, Fowlkes, Malik 2004: Learning to Detection Natural Boundaries…

Figure from Fowlkes

pB Boundary Detector

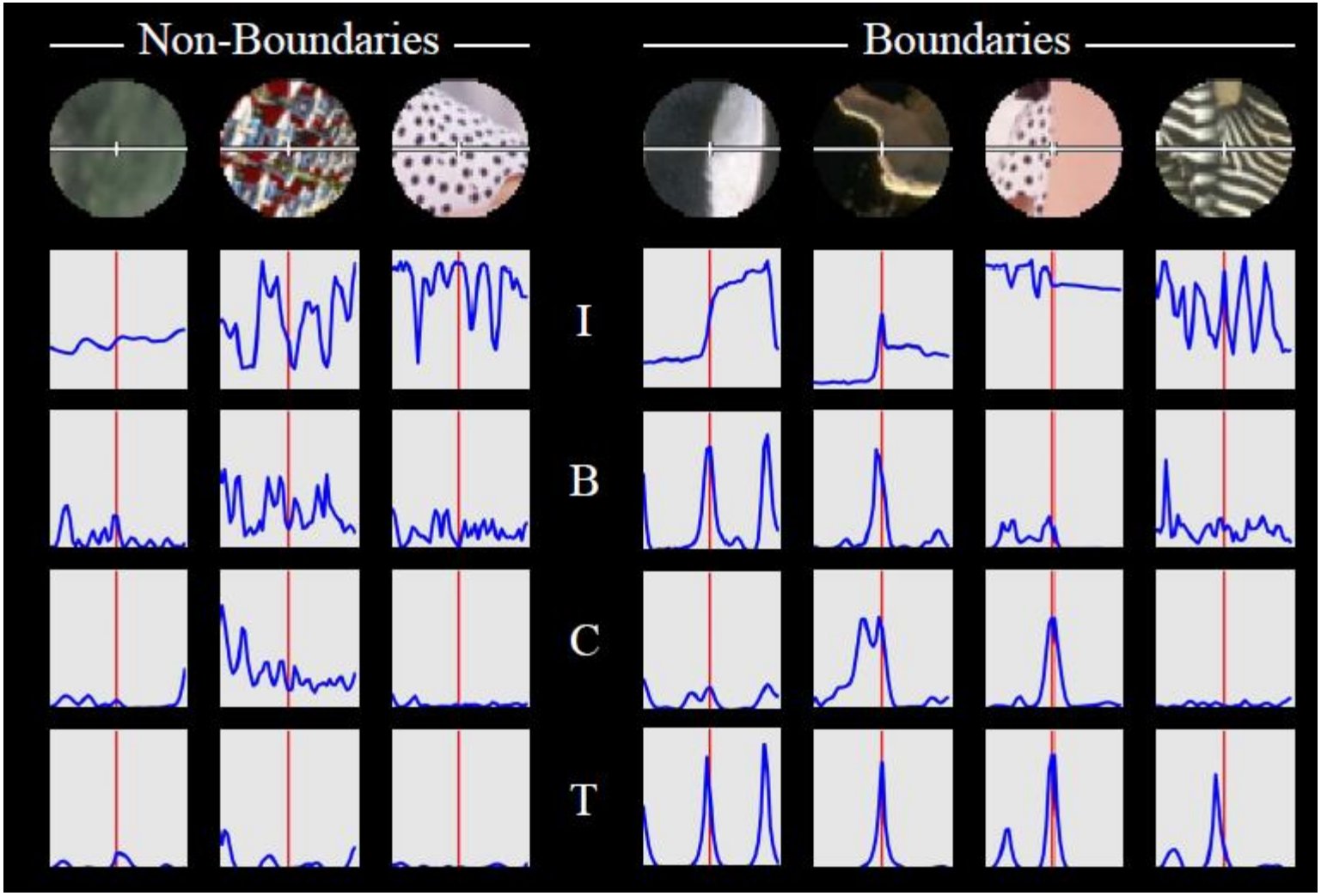


Figure from Fowlkes

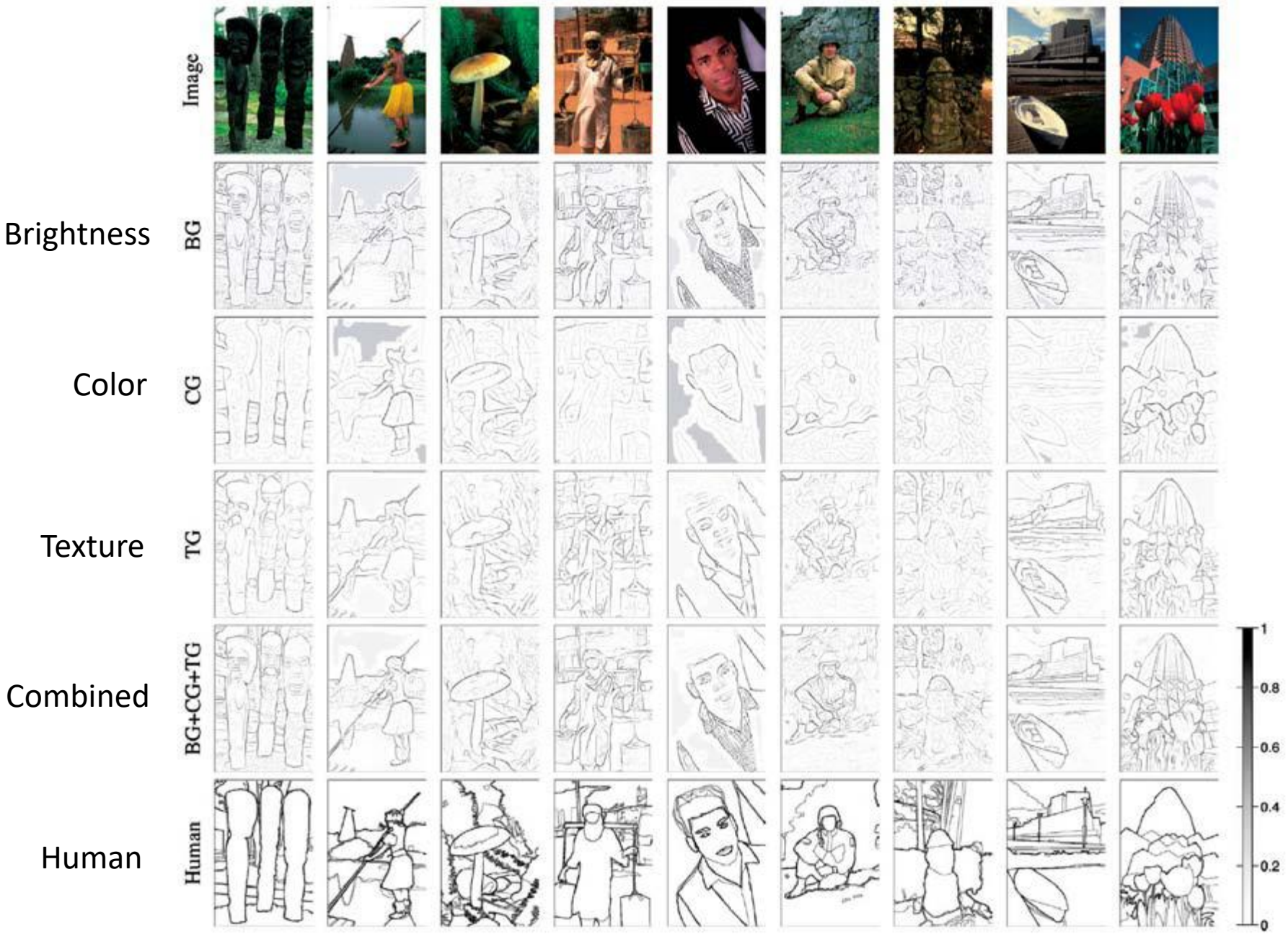
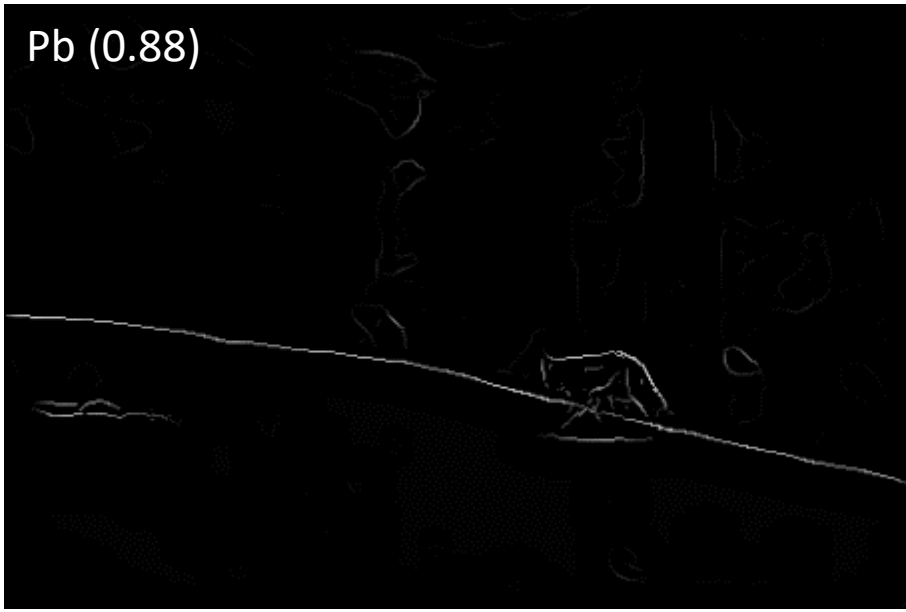
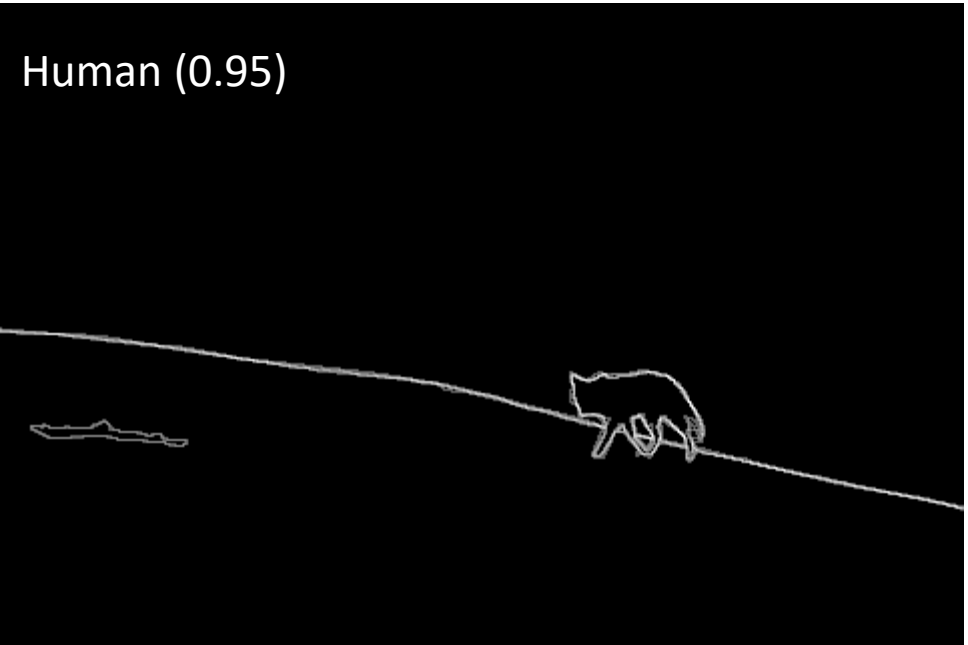


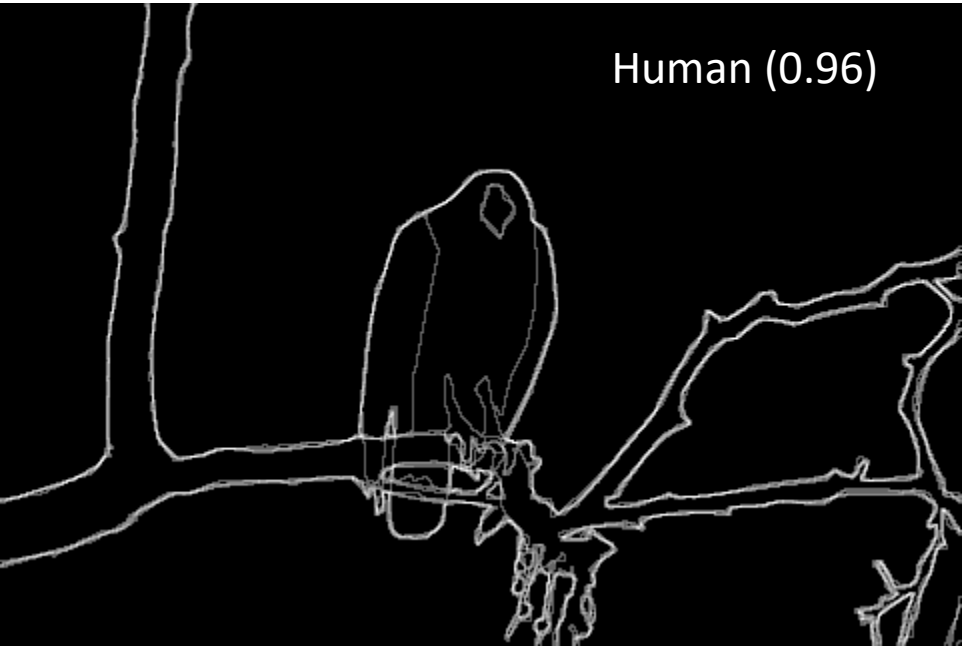
Image
BG
CG
TG
BG+CG+TG
Human
Brightness
Color
Texture
Combined
Human
1
0.8
0.6
0.4
0.2
0

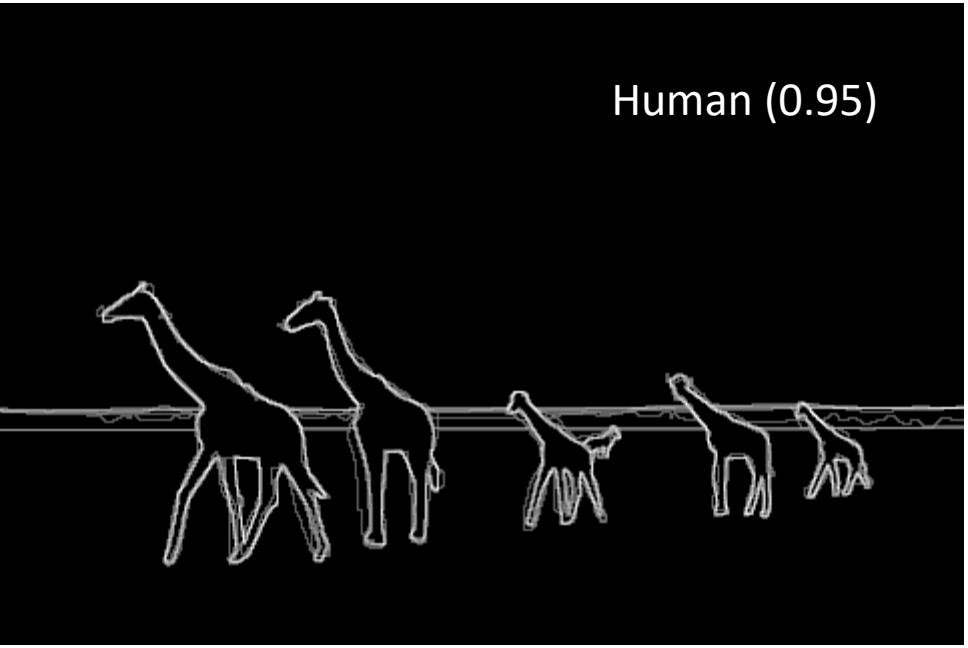
Results



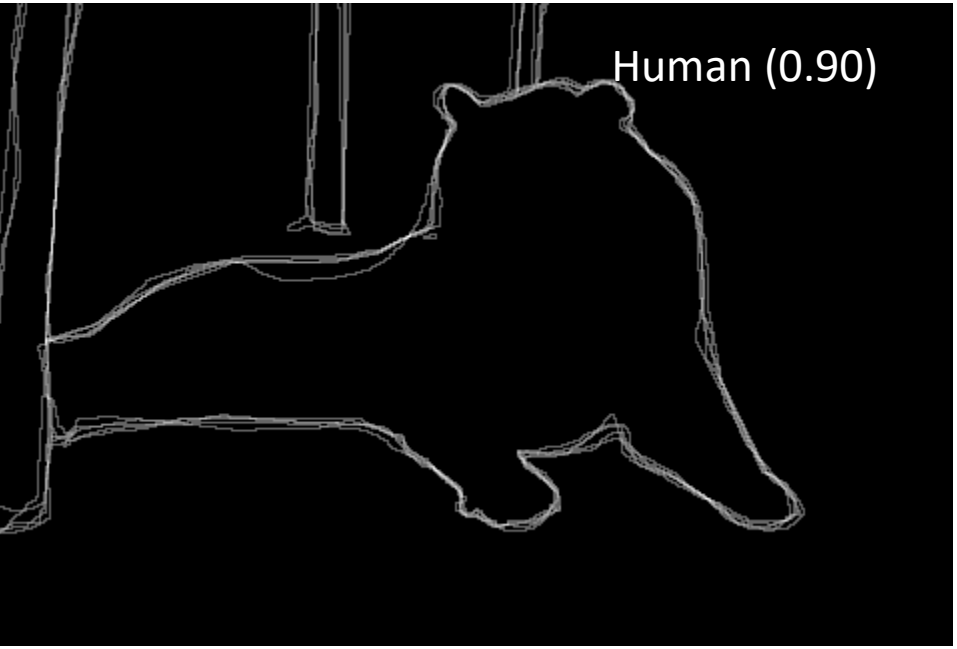


Results




Human (0.95)


Pb (0.63)



For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html

Global pB boundary detector

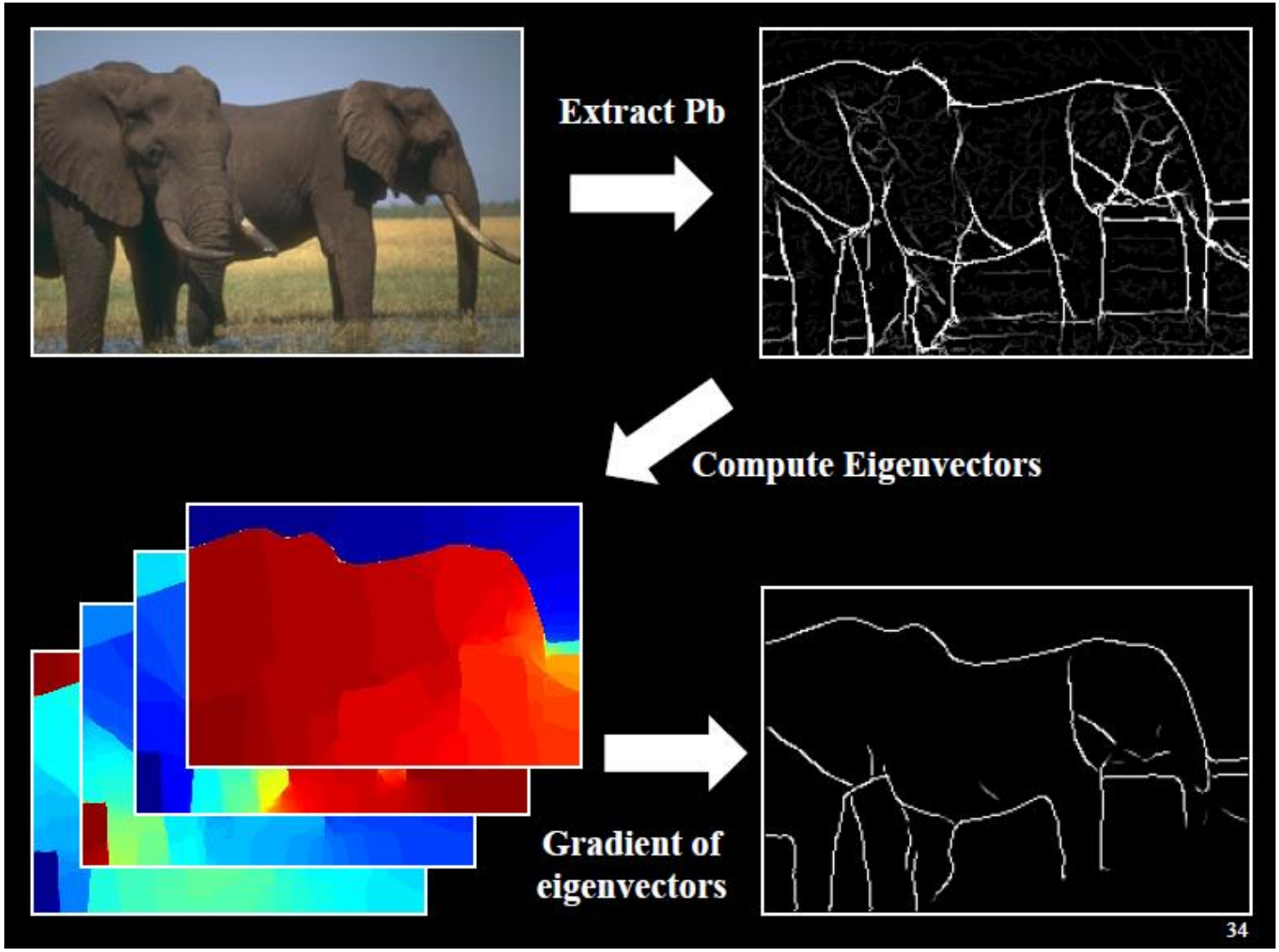


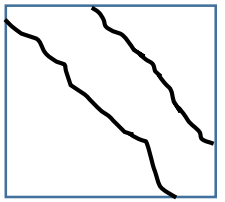
Figure from Fowlkes

Edge Detection with Structured Random Forests (Dollar and Zitnick ICCV 2013)

- Goal: quickly predict whether each pixel is an edge

- Insights
  - Predictions can be learned from training data
  - Predictions for nearby pixels should not be independent

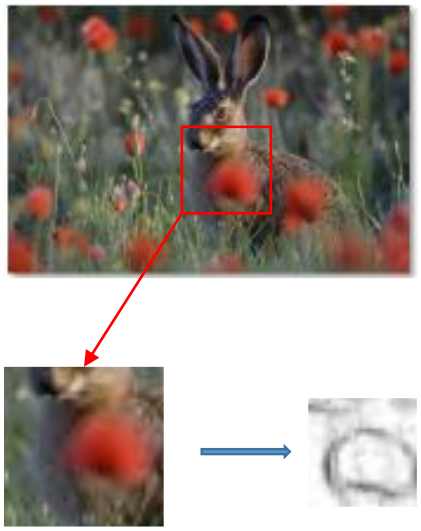
- Solution
  - Train structured random forests to split data into patches with similar boundaries based on features
  - Predict boundaries at patch level, rather than pixel level, and aggregate (average votes)



Boundaries in patch

http://research.microsoft.com/pubs/202540/DollarICCV13edges.pdf

Edge Detection with Structured Random Forests - Algorithm



1. Extract overlapping 32x32 patches at three scales
2. Features are pixel values and pairwise differences in feature maps (LUV color, gradient magnitude, oriented gradient)
3. Predict T boundary maps in the central 16x16 region using T trained decision trees
4. Average predictions for each pixel across all patches

Edge Detection with Structured Random Forests - Results

BSDS 500

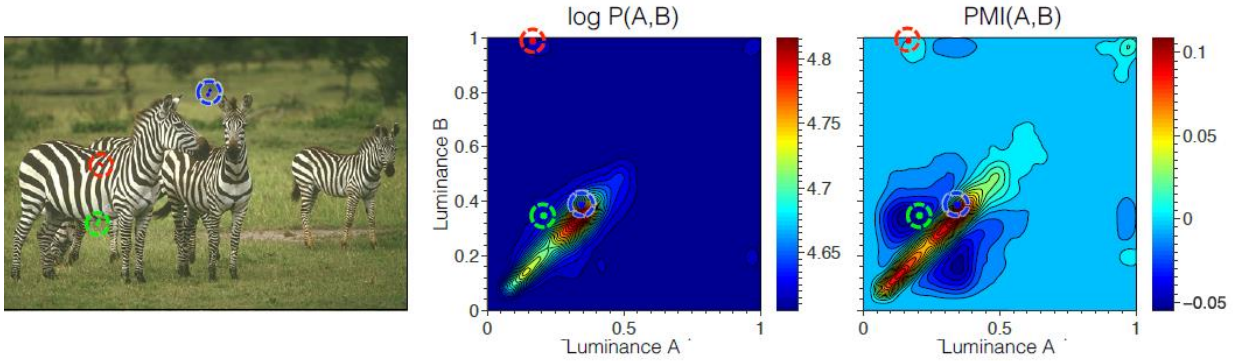
	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.60	.64	.58	15
Felz-Hutt [11]	.61	.64	.56	10
Hidayat-Green [16]	.62†	-	-	20
BEL [9]	.66†	-	-	1/10
gPb + GPU [6]	.70†	-	-	1/2‡
gPb [1]	.71	.74	.65	1/240
gPb-owt-ucm [1]	.73	**.76**	.73	1/240
Sketch tokens [21]	.73	.75	**.78**	1
SCG [31]	**.74**	**.76**	.77	1/280
SE-SS, T=1	.72	.74	.77	**60**
SE-SS, T=4	.73	.75	.77	30
SE-MS, T=4	**.74**	**.76**	**.78**	6

NYU Depth dataset edges

	ODS	OIS	AP	FPS
gPb [1] (rgb)	.51	.52	.37	1/240
SCG [31] (rgb)	.55	.57	.46	1/280
SE-SS (rgb)	.58	.59	.53	**30**
SE-MS (rgb)	**.60**	**.61**	**.56**	6
gPb [1] (depth)	.44	.46	.28	1/240
SCG [31] (depth)	.53	.54	.45	1/280
SE-SS (depth)	.57	.58	.54	**30**
SE-MS (depth)	**.58**	**.59**	**.57**	6
gPb [1] (rgbd)	.53	.54	.40	1/240
SCG [31] (rgbd)	.62	.63	.54	1/280
SE-SS (rgbd)	.62	.63	.59	**25**
SE-MS (rgbd)	**.64**	**.65**	**.63**	5

Edge Detection with Structured Random Forests

Crisp Boundary Detection using Pointwise Mutual Information (Isola et al. ECCV 2014)



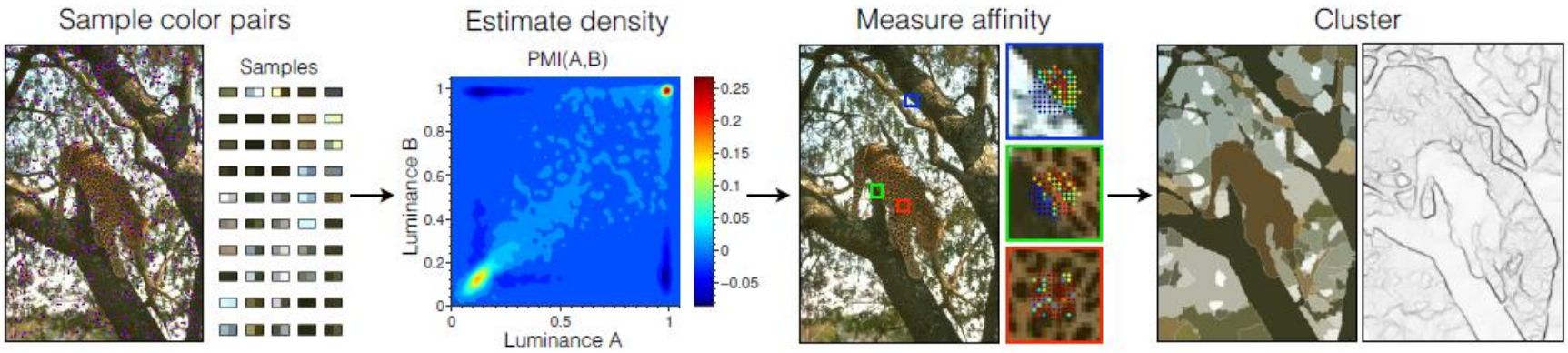
$$\mathrm{PMI}_\rho(A,B) = \log \frac{P(A,B)^\rho}{P(A)P(B)}$$

Pixel combinations that are unlikely to be together are edges

Algorithm:

Kernel density estimation

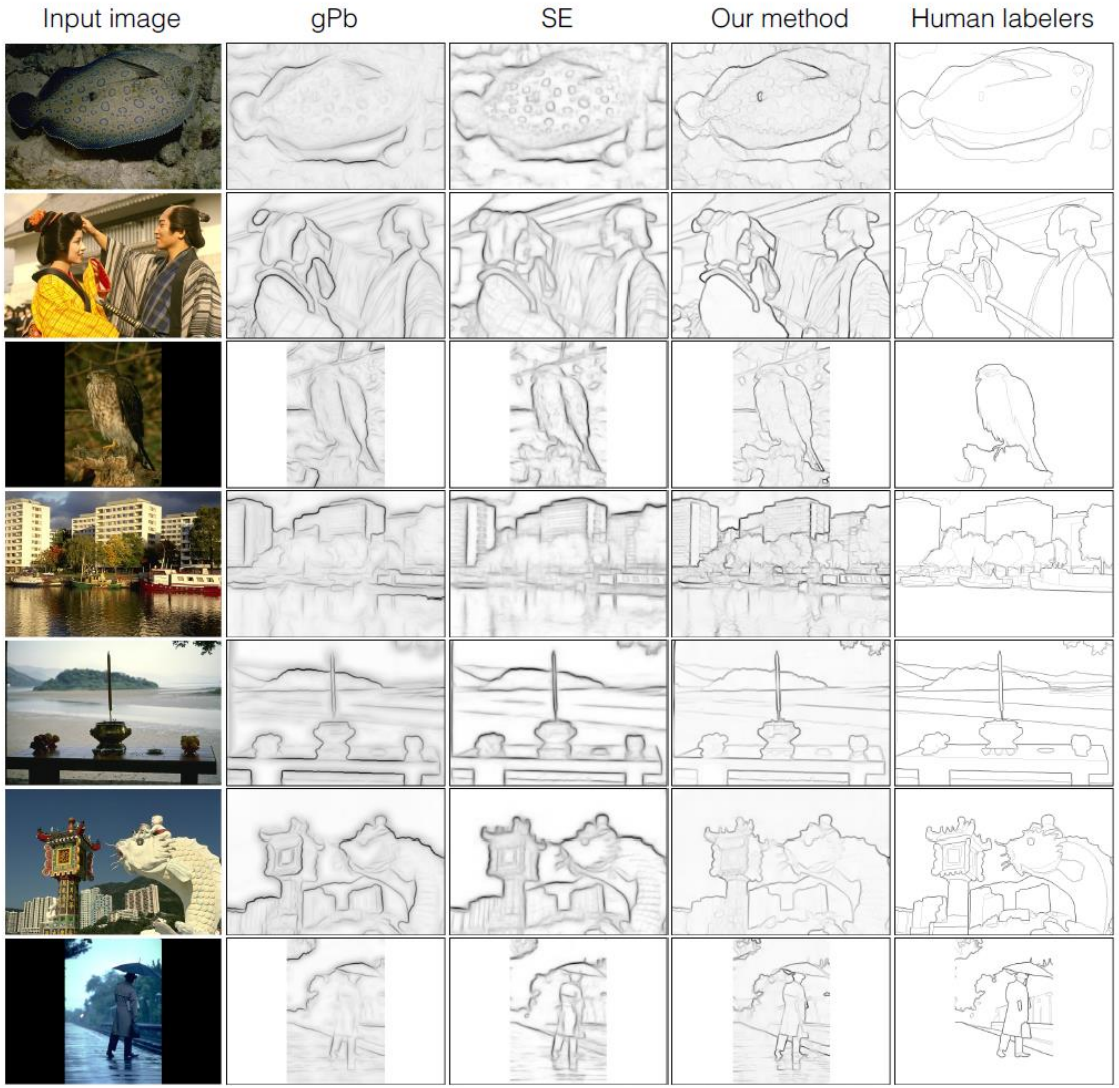
Spectral clustering



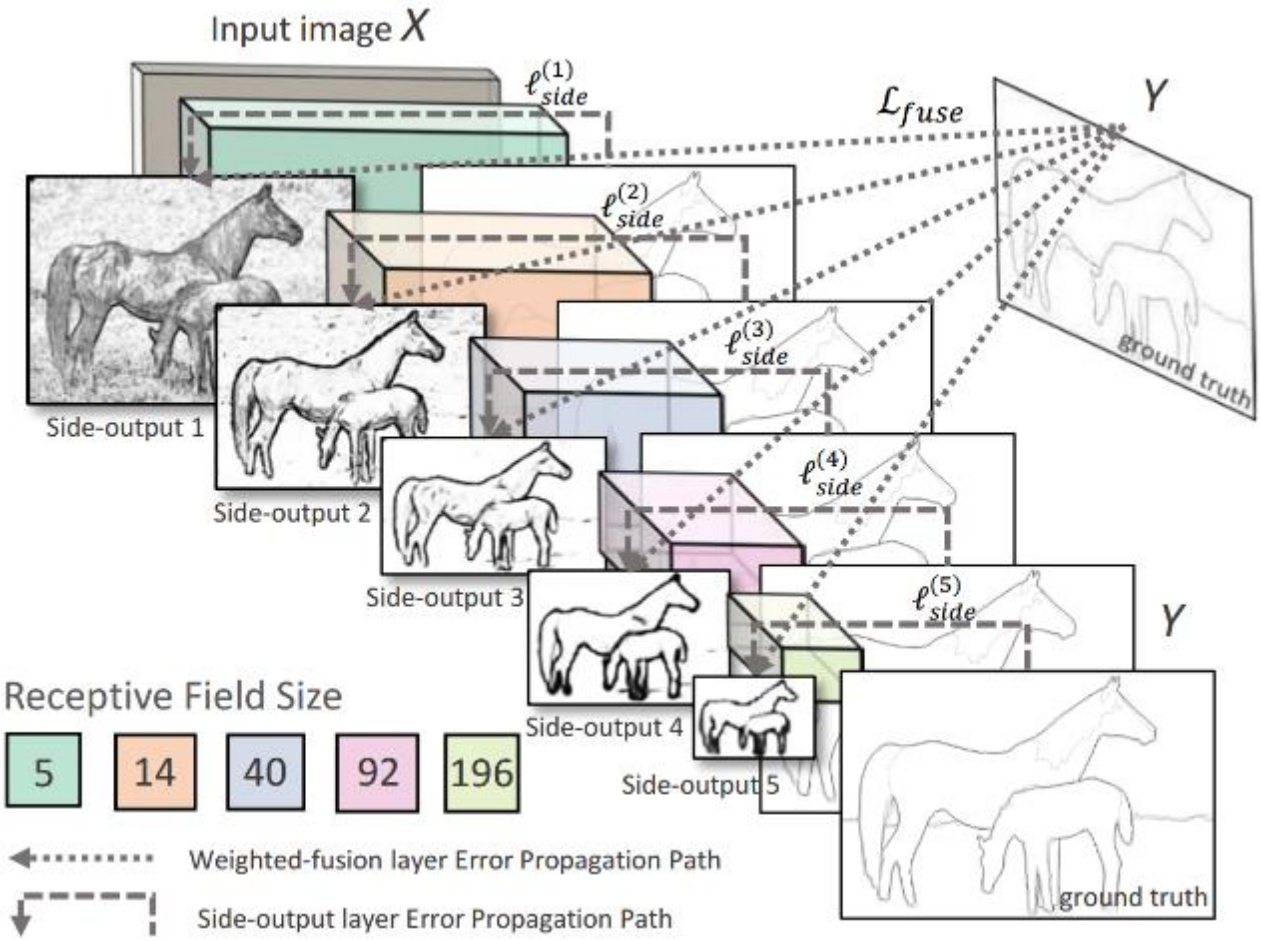
http://web.mit.edu/phillipi/www/publications/crisp_boundaries.pdf

Crisp Boundary Detection using Pointwise Mutual Information

Algorithm	ODS	OIS	AP
Canny [14]	0.60	0.63	0.58
Mean Shift [36]	0.64	0.68	0.56
NCuts [37]	0.64	0.68	0.45
Felz-Hutt [38]	0.61	0.64	0.56
gPb [1]	0.71	0.74	0.65
gPb-owt-ucm [1]	0.73	0.76	0.73
SCG [9]	**0.74**	0.76	0.77
Sketch Tokens [7]	0.73	0.75	0.78
SE [8]	**0.74**	0.76	0.78
Our method – SS, color only	0.72	0.75	0.77
Our method – SS	0.73	0.76	**0.79**
Our method – MS	**0.74**	**0.77**	0.78

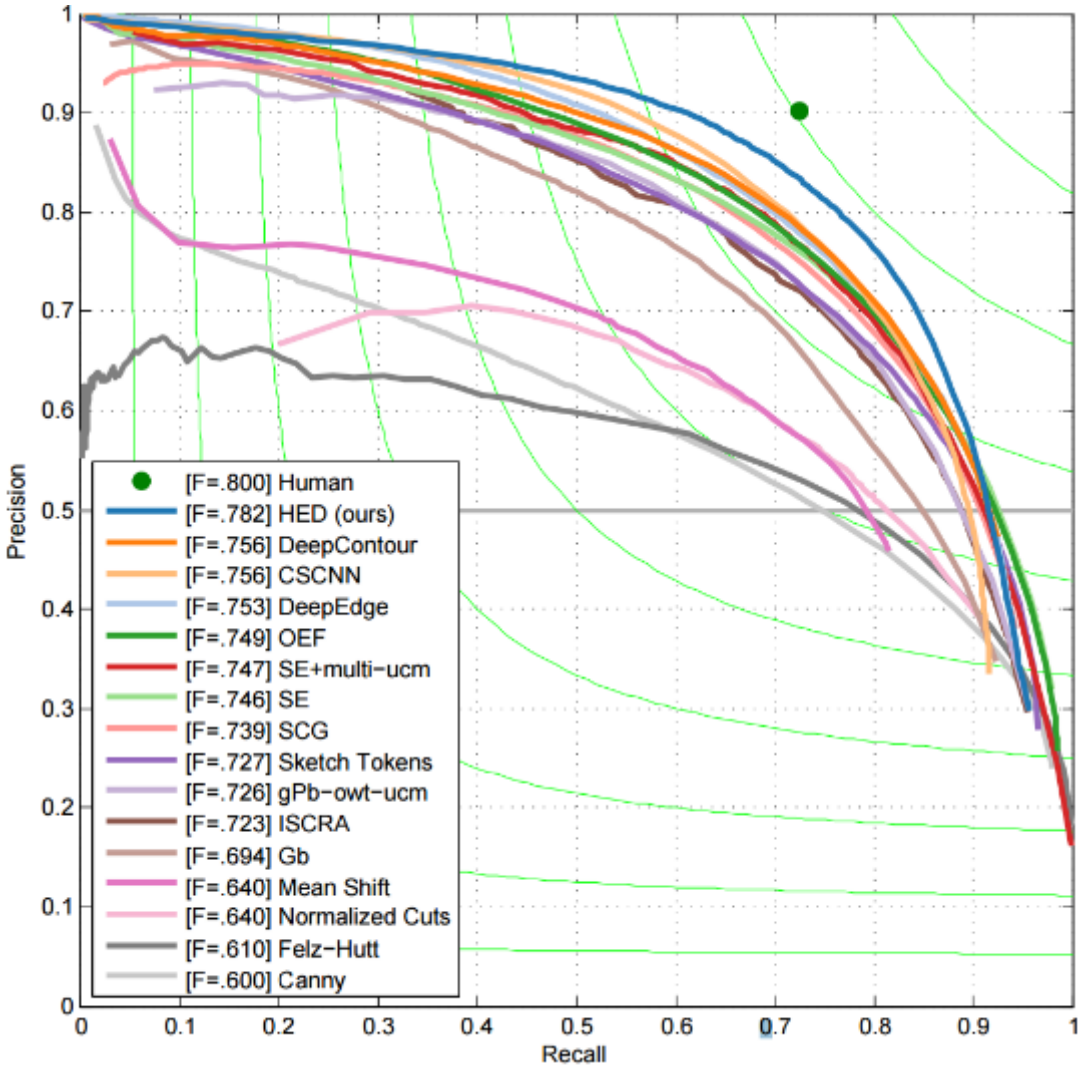


Holistically-Nested Edge Detection



Xie and Tu, Holistically-Nested Edge Detection, ICCV 2015

Holistically-Nested Edge Detection



Xie and Tu, Holistically-Nested Edge Detection, ICCV 2015

State of edge detection

- Local edge detection is mostly solved
  - Intensity gradient, color, texture
- Work on RGB-D edge detection is currently more active
- Some methods take into account longer contours, but could probably do better
- Often used in combination with object detectors or region classifiers

Finding straight lines

Finding line segments using connected components

1. Compute canny edges
   - Compute: gx, gy (DoG in x,y directions)
   - Compute: theta = atan(gy / gx)
2. Assign each edge to one of 8 directions
3. For each direction d, get edgelets:
   - find connected components for edge pixels with directions in {d-1, d, d+1}
4. Compute straightness and theta of edgelets using eig of x,y 2nd moment matrix of their points

$$\mathbf{M} = \begin{bmatrix} \sum (x-\mu_x)^2 & \sum (x-\mu_x)(y-\mu_y) \\ \sum (x-\mu_x)(y-\mu_y) & \sum (y-\mu_y)^2 \end{bmatrix} \qquad [v, \lambda] = \text{eig}(\mathbf{M})$$

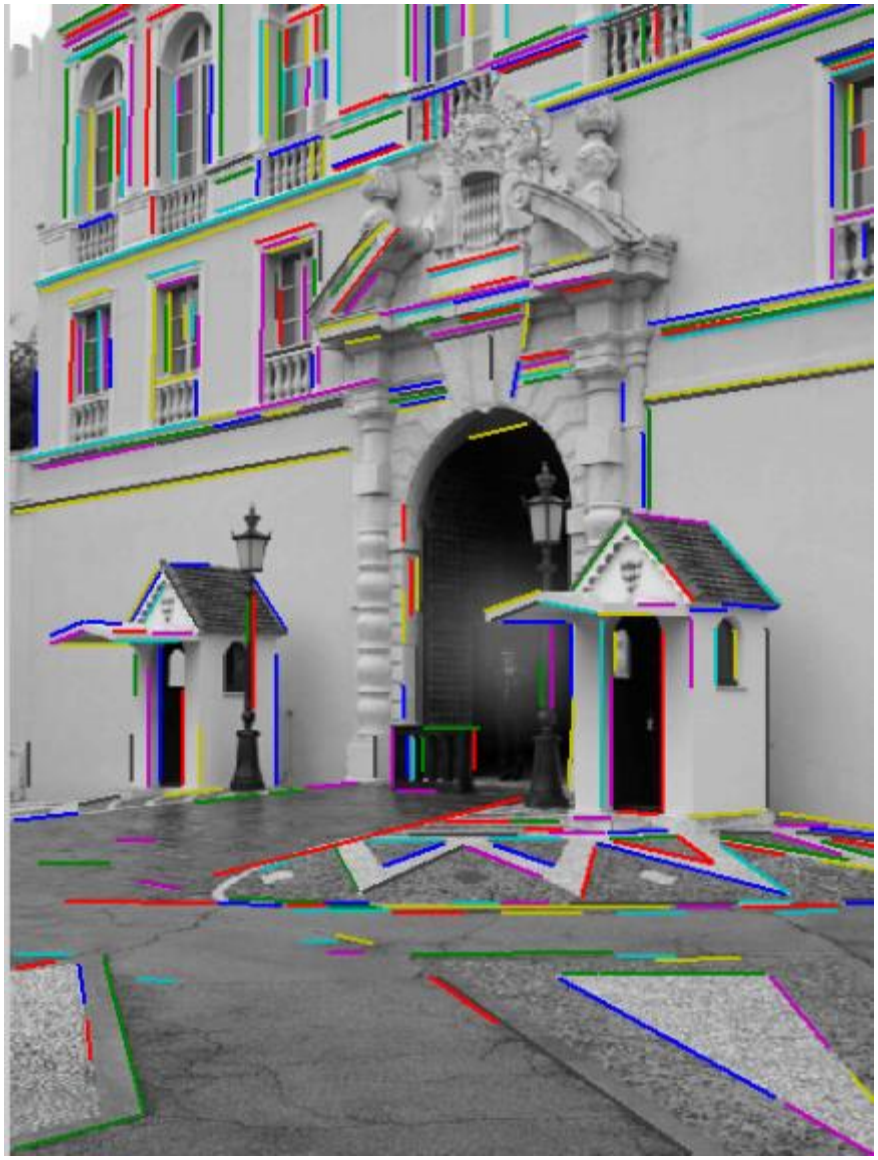
Larger eigenvector ↙

$$\theta = \text{atan}\,2(v(2,2), v(1,2))$$

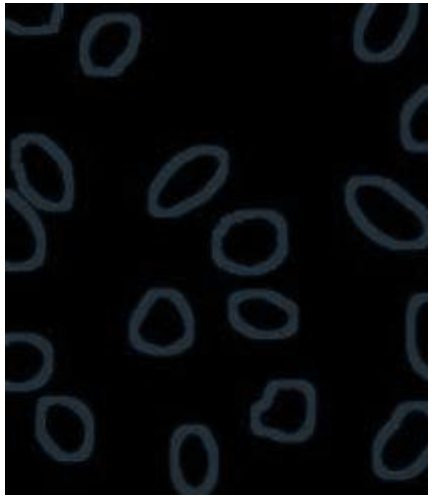
$$conf = \lambda_2 / \lambda_1$$

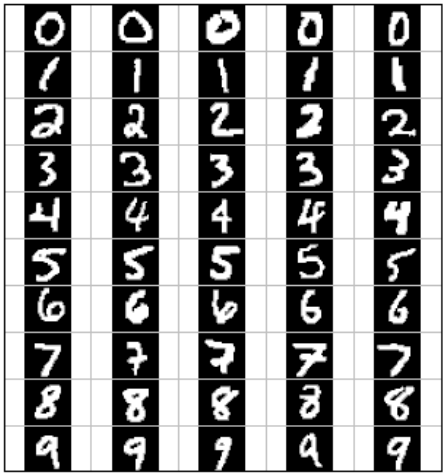
5. Threshold on straightness, store segment

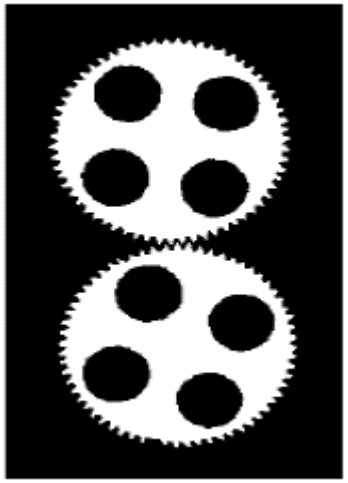
Canny lines → … →straight edges

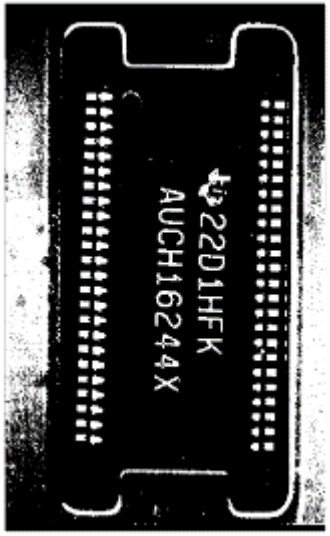


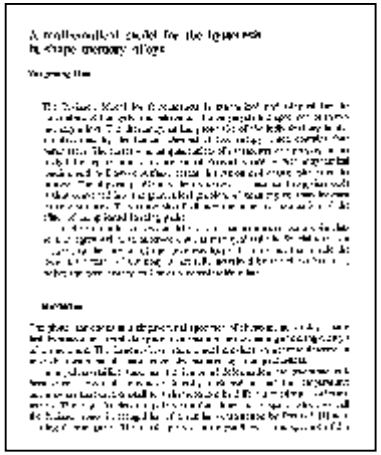
Binary images











Slide credit: Kristen Grauman

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

Slide credit: Kristen Grauman

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

Slide credit: Kristen Grauman

Thresholding

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

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

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

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

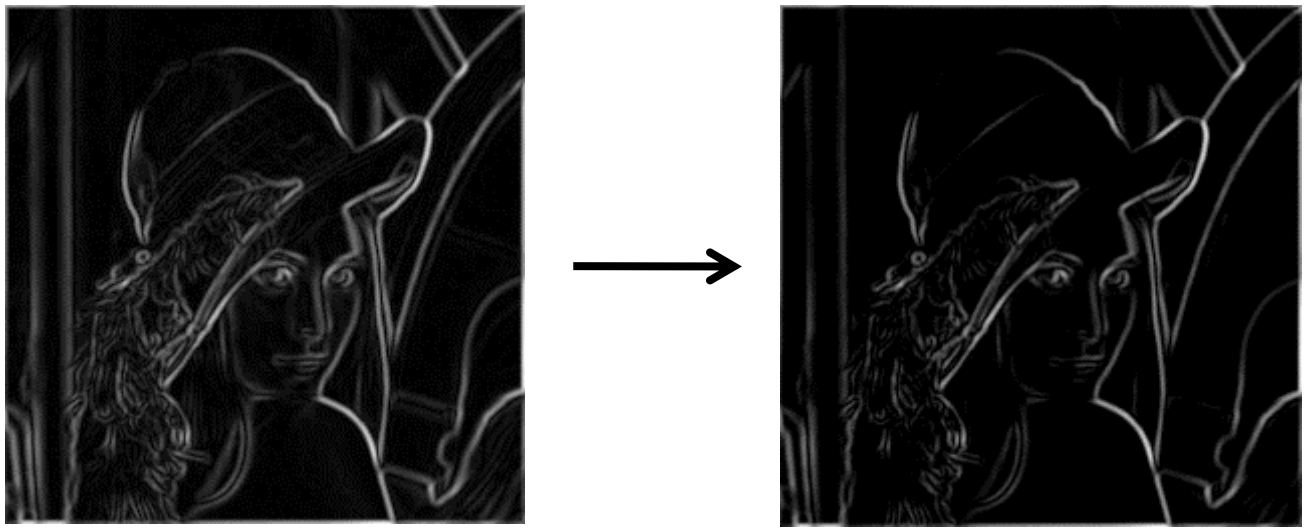
- Example http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FITZGIBBON/simplebinary.html

Slide credit: Kristen Grauman

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);
```

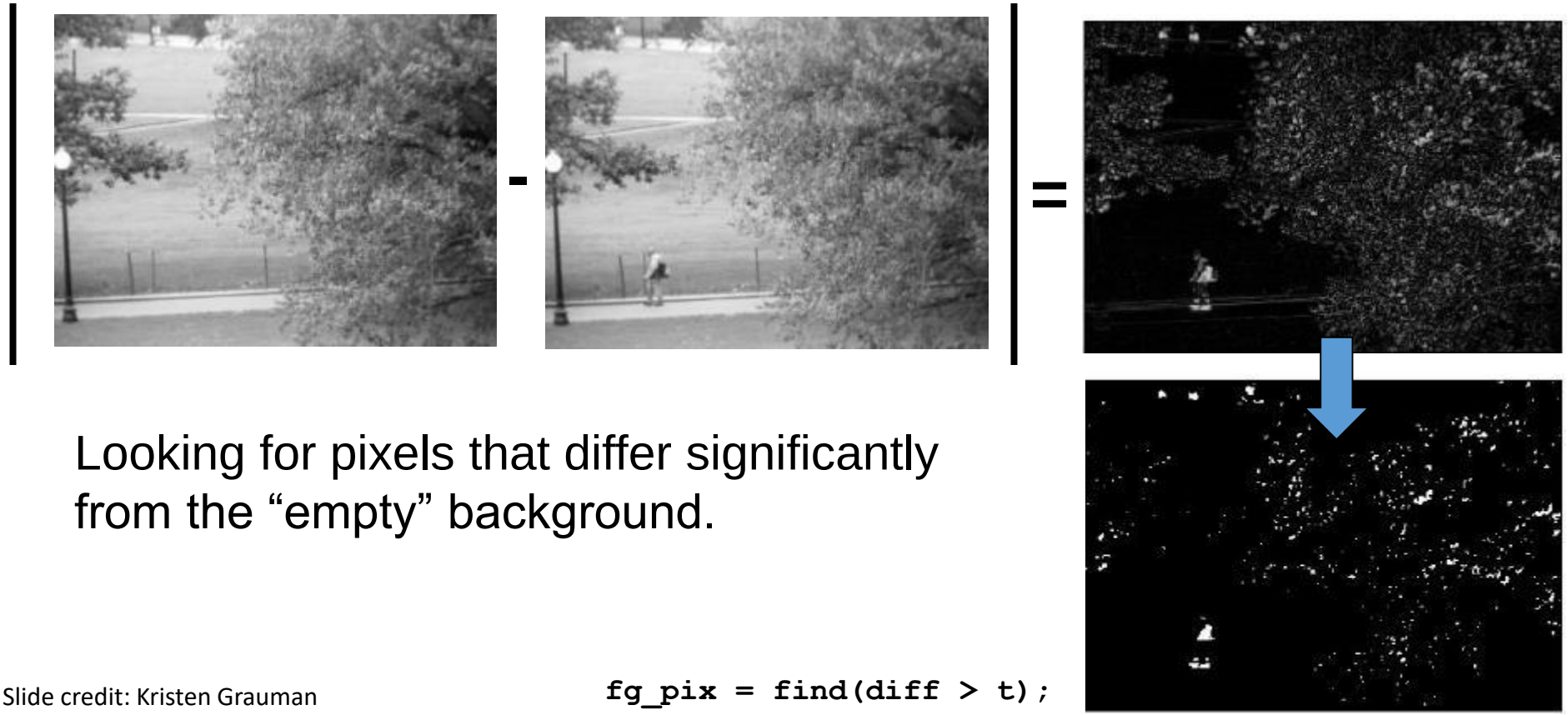
Looking for pixels where gradient is strong.

Slide credit: Kristen Grauman

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.

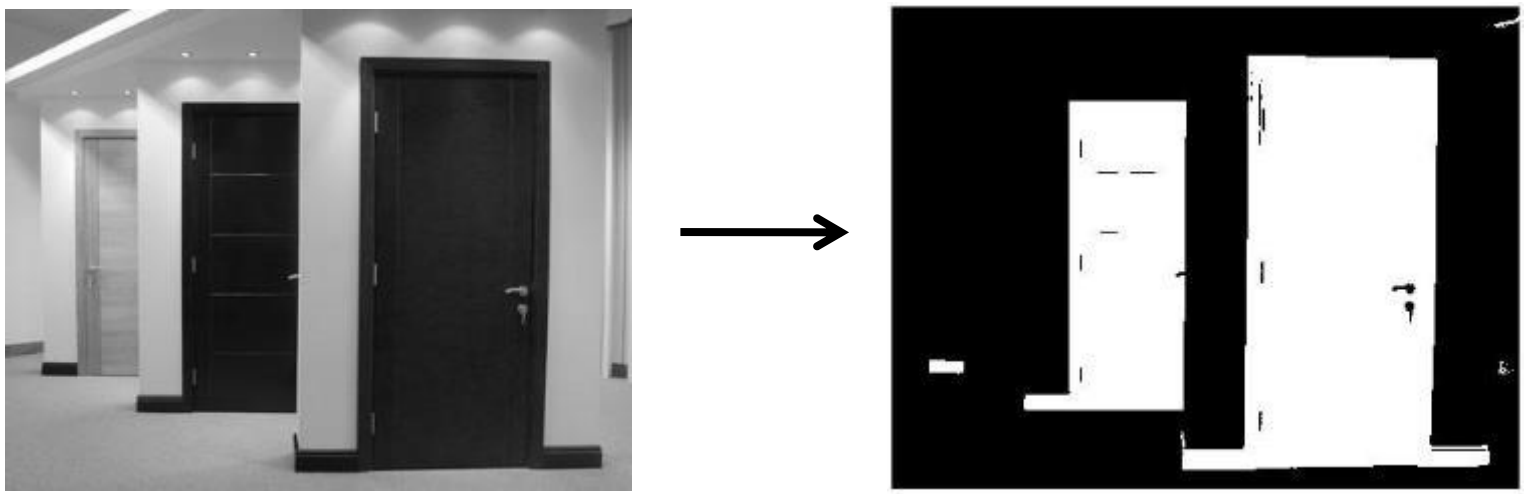
```
fg_pix = find(diff > t);
```

Slide credit: Kristen Grauman

Thresholding

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

Example: intensity-based detection



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

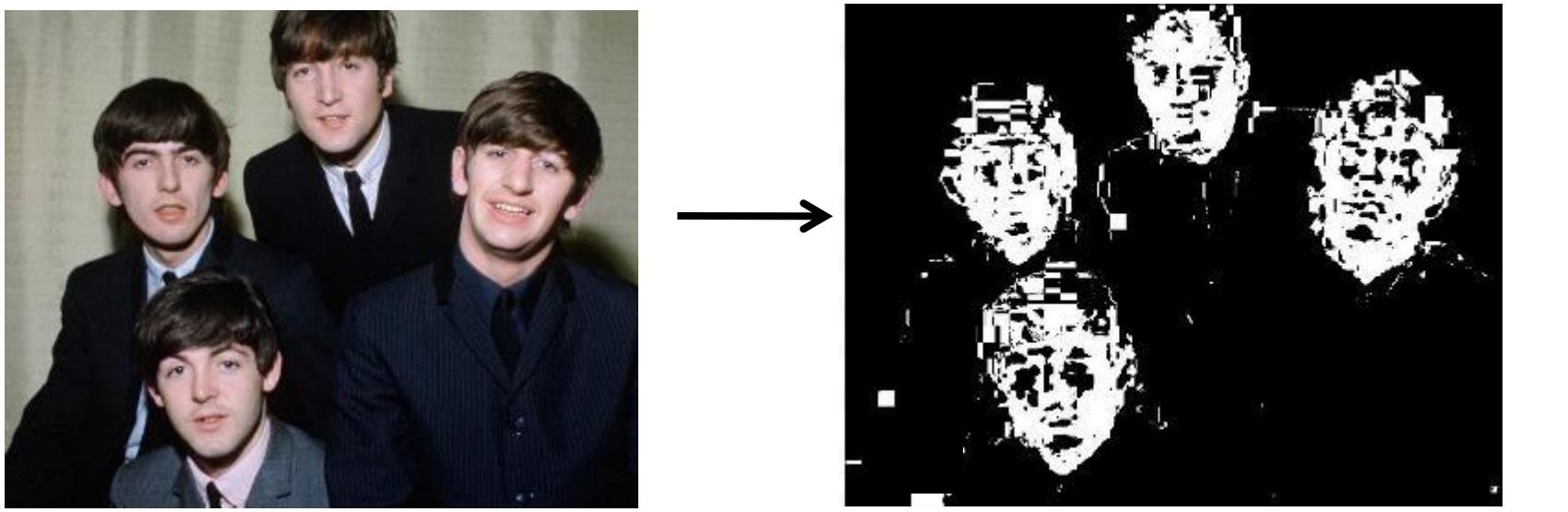
Looking for dark pixels

Slide credit: Kristen Grauman

Thresholding

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

Example: color-based detection

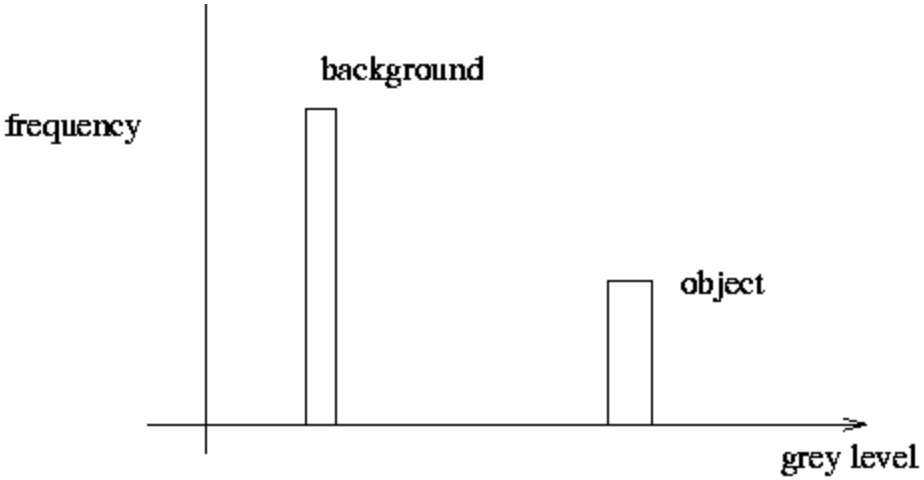


```
fg_pix = find(hue > t1 & hue < t2);
```

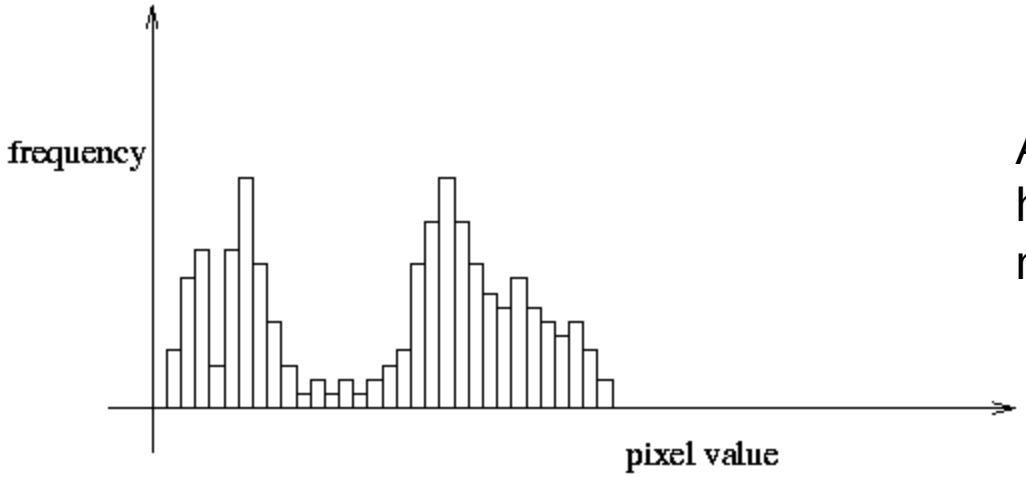
Looking for pixels within a certain hue range.

Slide credit: Kristen Grauman

A nice case: bimodal intensity histograms



Ideal histogram, light object on dark background

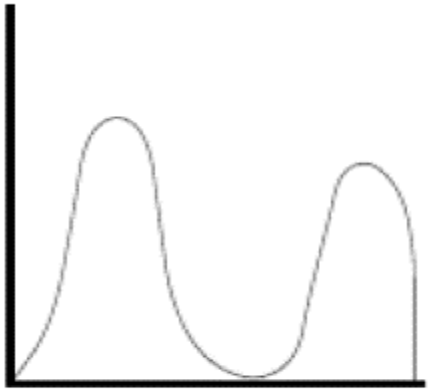


Actual observed histogram with noise

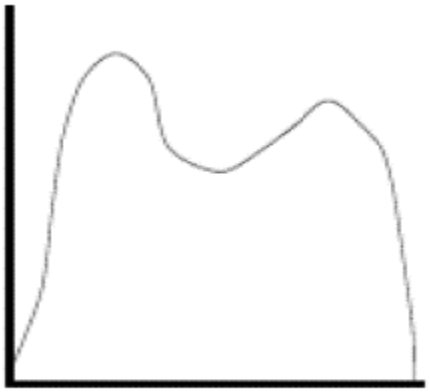
Images: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT2/node3.html

Slide credit: Kristen Grauman

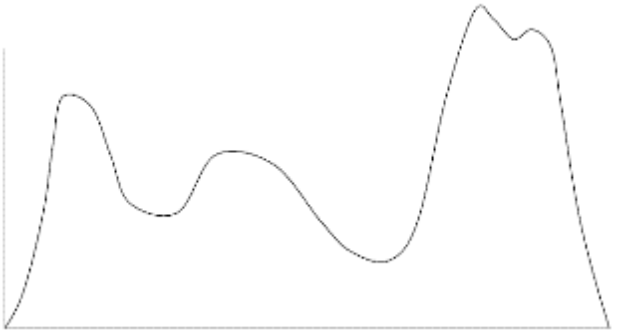
Not so nice cases



Two distinct modes



Overlapped modes

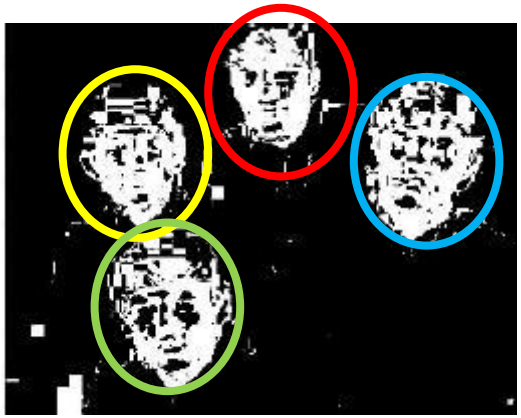


Slide credit: Shapiro and Stockman

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



Slide credit: Kristen Grauman

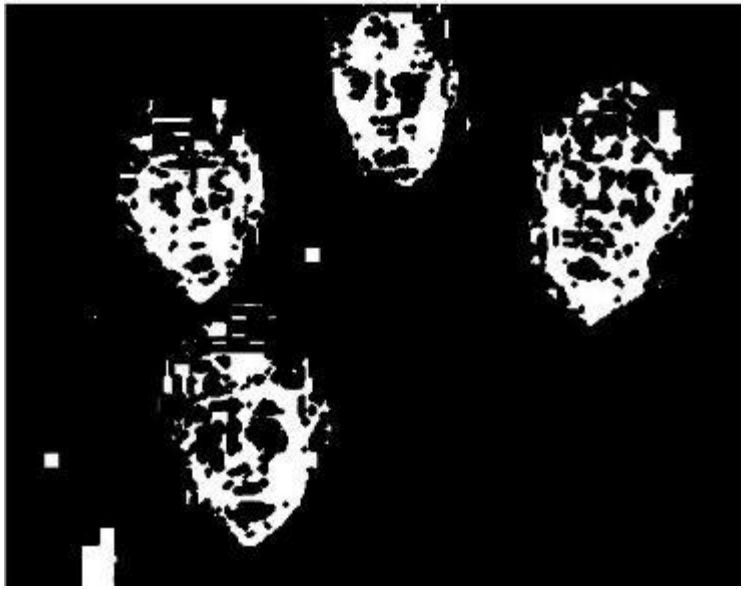
Morphological operators

- 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

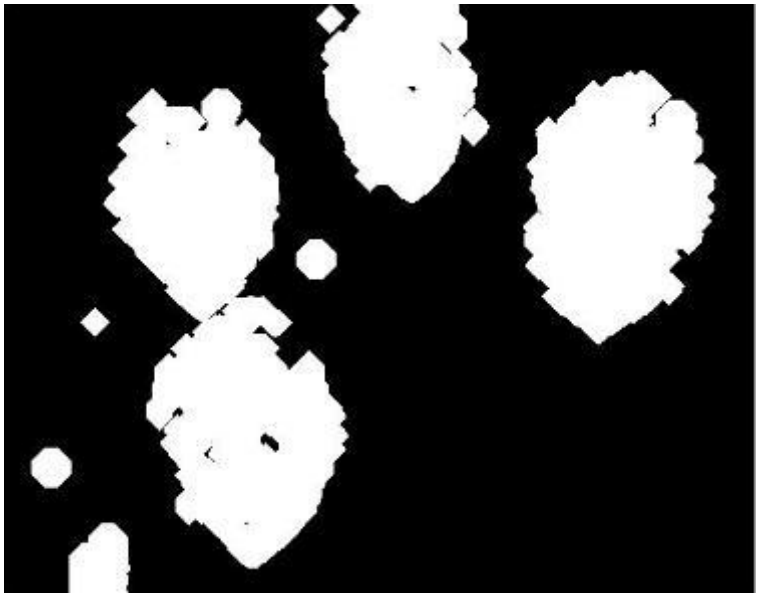
Slide credit: Kristen Grauman

Dilation

- Expands connected components
- Grow features
- Fill holes



Before dilation

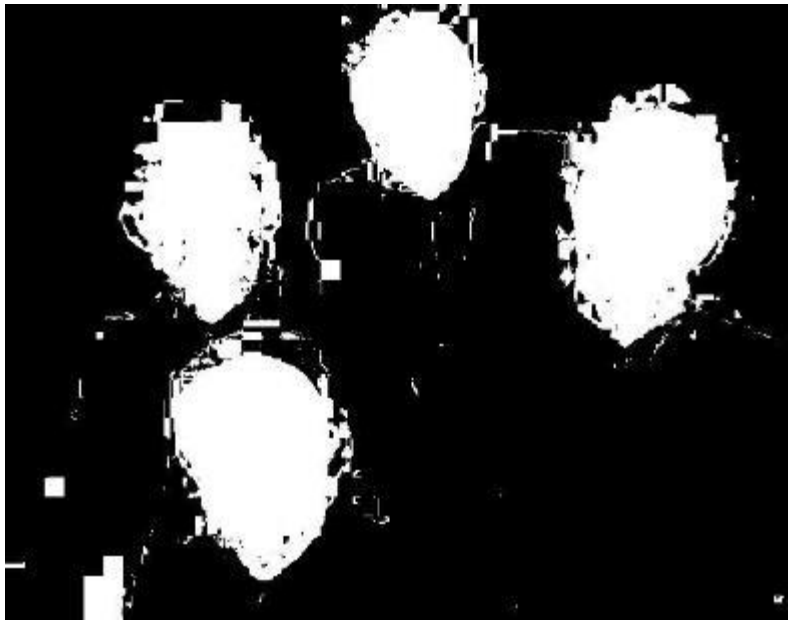


After dilation

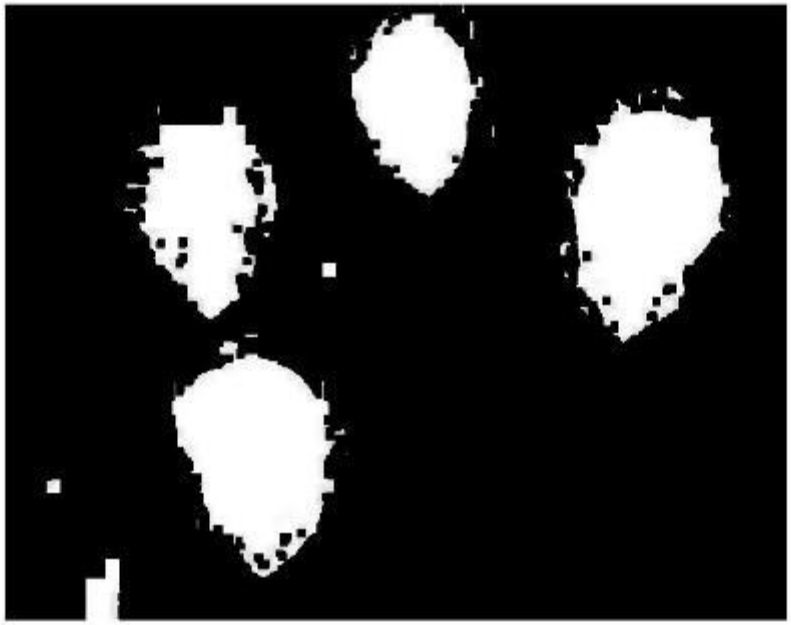
Slide credit: Kristen Grauman

Erosion

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



Before erosion

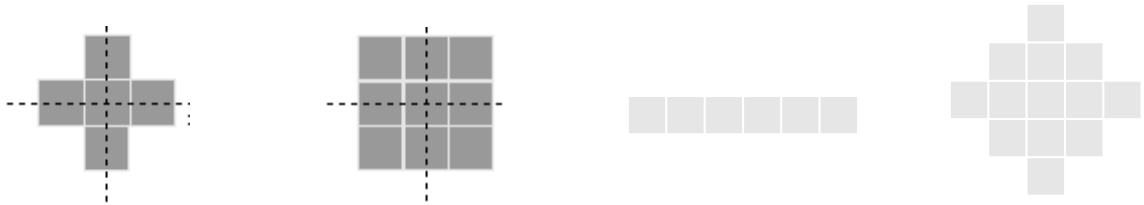


After erosion

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
```

Slide credit: Kristen Grauman

Dilation vs. Erosion

At each position:

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

Slide credit: Kristen Grauman

Example for Dilation

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								

Slide credit: Adapted by Kristen Grauman from T. Moeslund

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

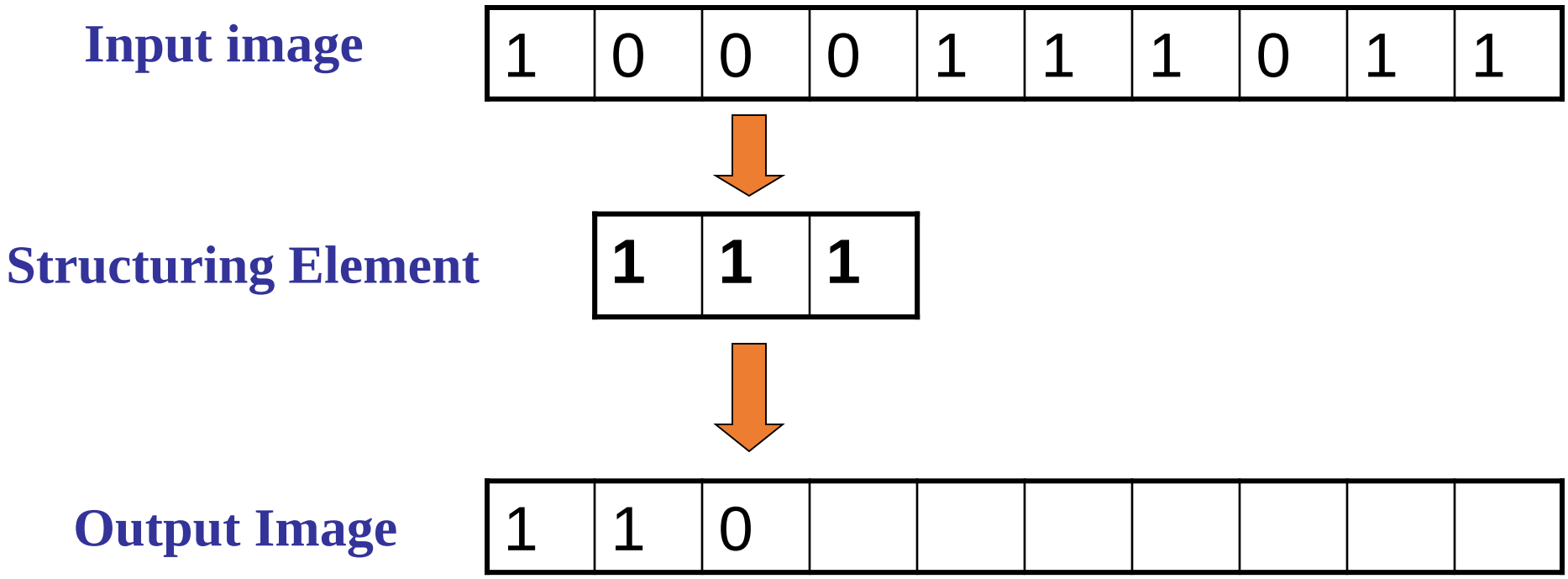
1	**1**	**1**

Output Image

1	1								

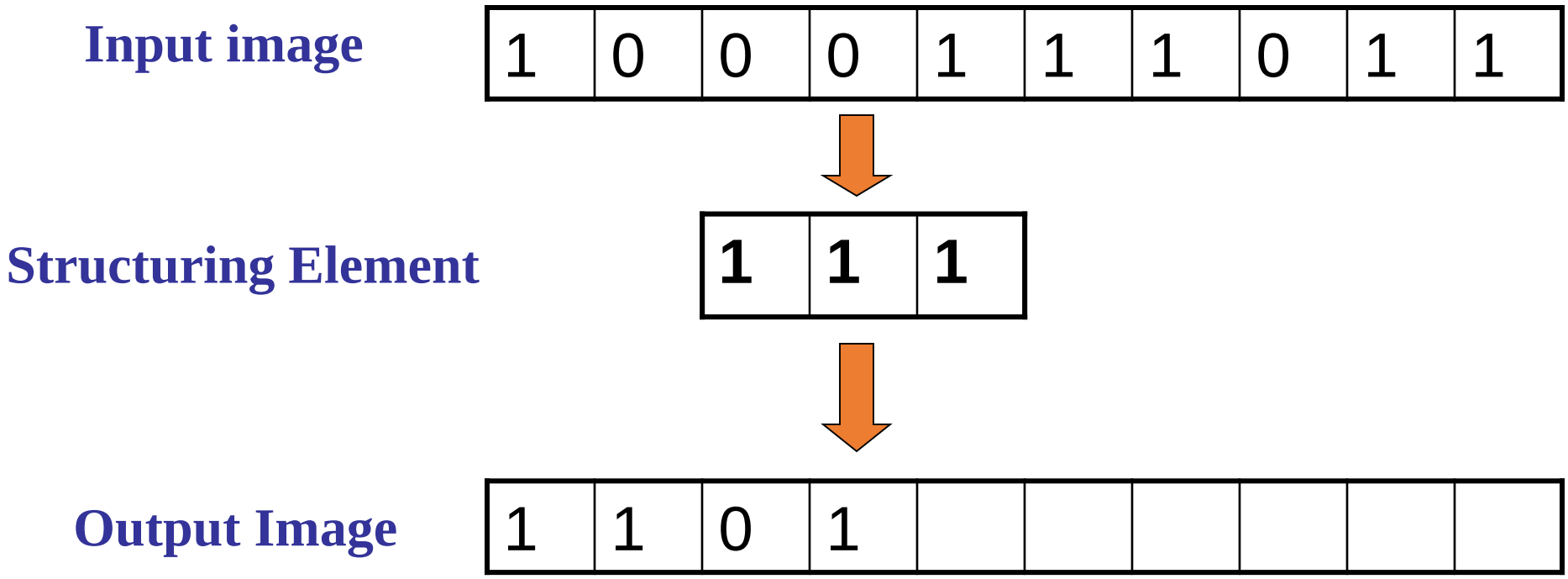
Slide credit: Kristen Grauman

Example for Dilation



Slide credit: Kristen Grauman

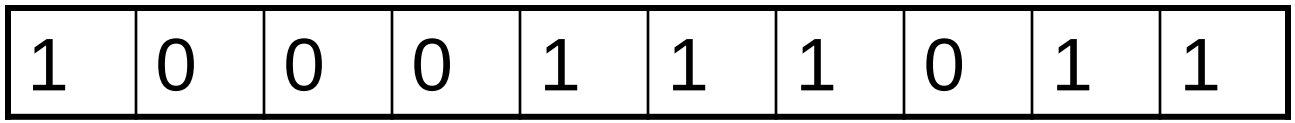
Example for Dilation



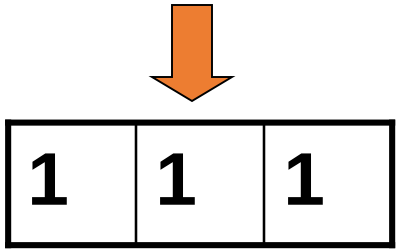
Slide credit: Kristen Grauman

Example for Dilation

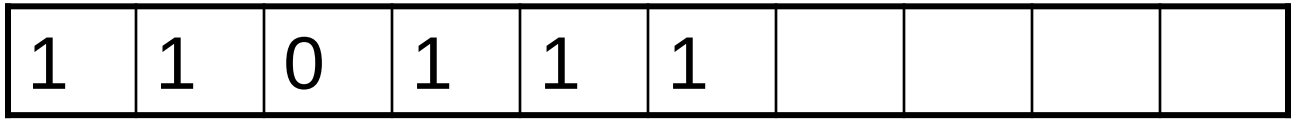
Input image



Structuring Element



Output Image



Slide credit: Kristen Grauman

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

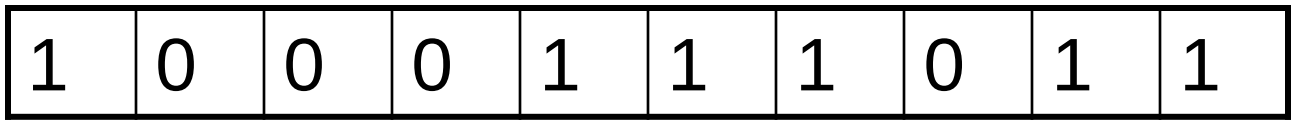
Output Image

1	1	0	1	1	1	1			

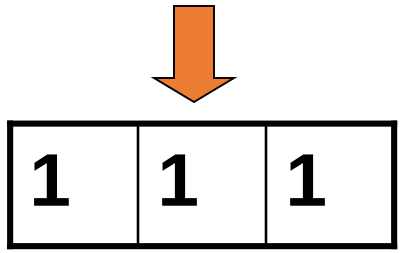
Slide credit: Kristen Grauman

Example for Dilation

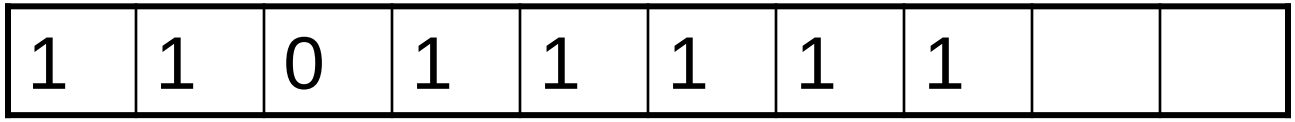
Input image



Structuring Element



Output Image



Slide credit: Kristen Grauman

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		

Slide credit: Kristen Grauman

Example for Dilation

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	**1**

Output Image

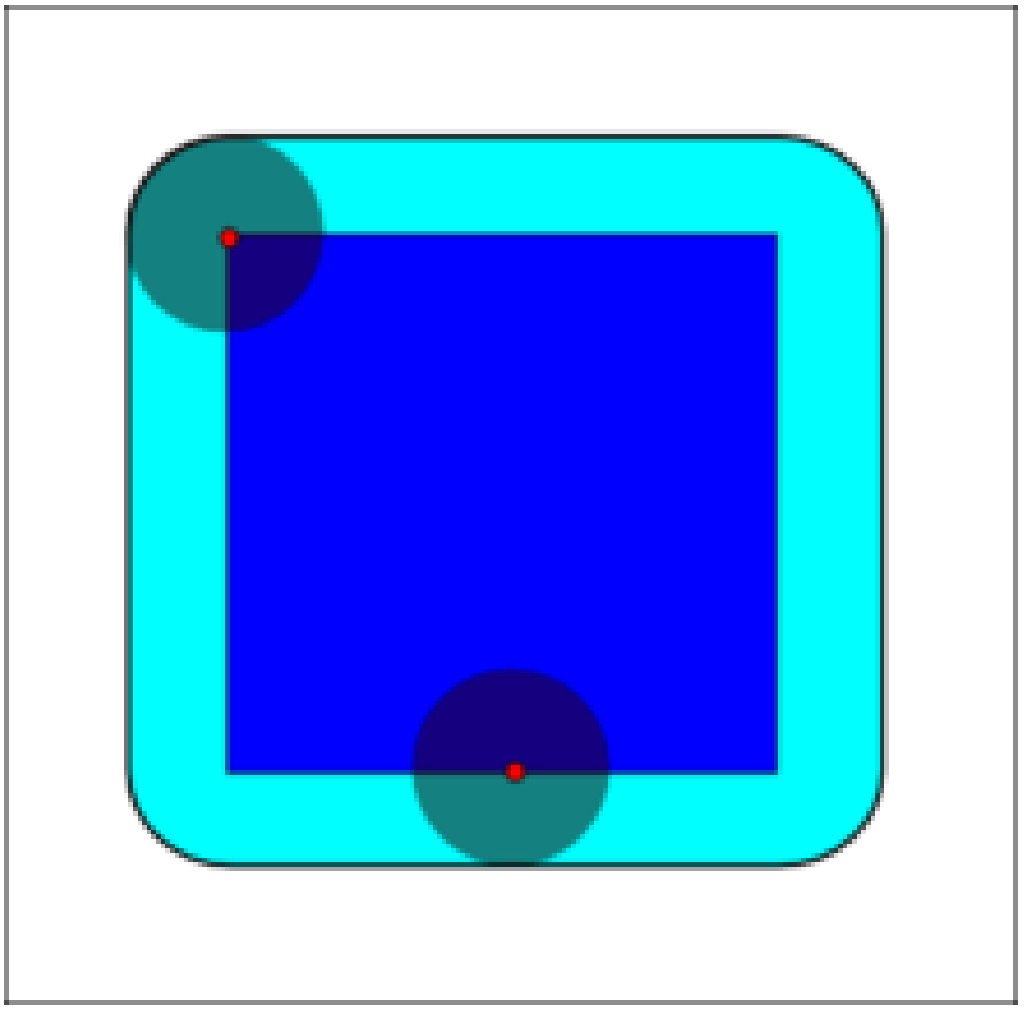
1	1	0	1	1	1	1	1	1	1

Note that the object gets bigger and holes are filled.

```
>> help imdilate
```

Slide credit: Kristen Grauman

2D example for dilation



Dilation vs. Erosion

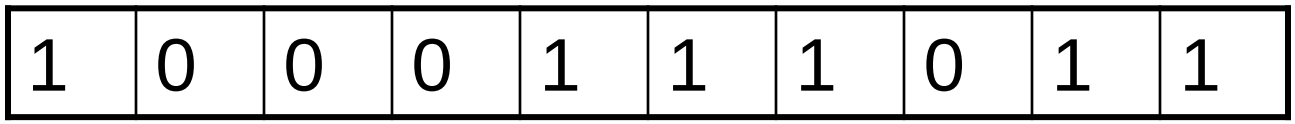
At each position:

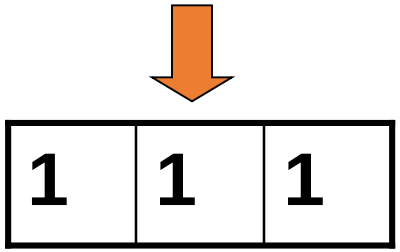
- **Dilation**: if **current pixel** is foreground, OR the structuring element with the input image.
- **Erosion**: if **every pixel** under the structuring element's nonzero entries is foreground, OR the current pixel with S.

Slide credit: Kristen Grauman

Example for Erosion (1D)

Input image





Structuring Element

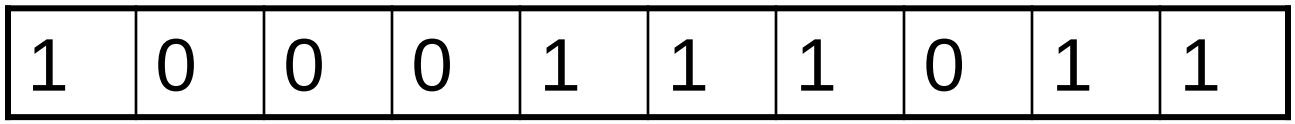
$$g(x) = f(x) \ominus SE$$

Output Image

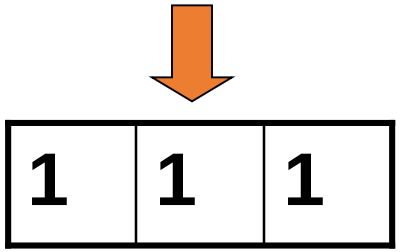
Slide credit: Kristen Grauman

Example for Erosion (1D)

Input image



Structuring Element

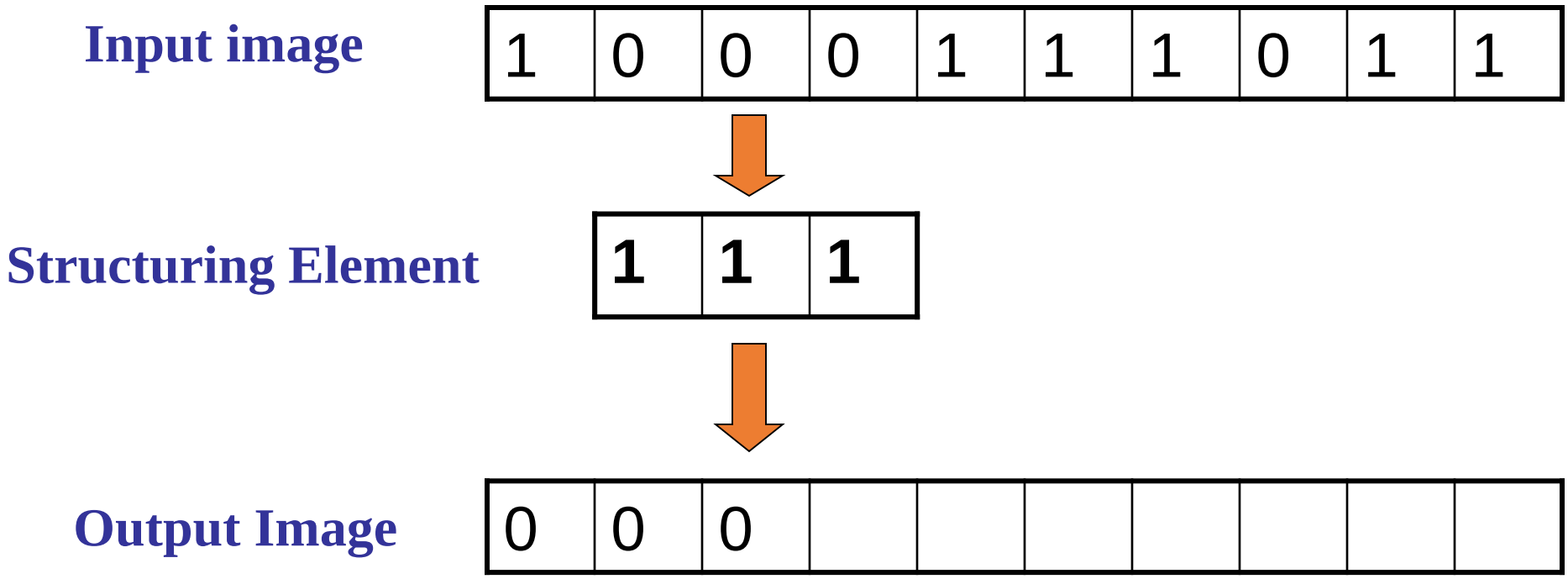


$$g(x) = f(x) \ominus SE$$

Output Image

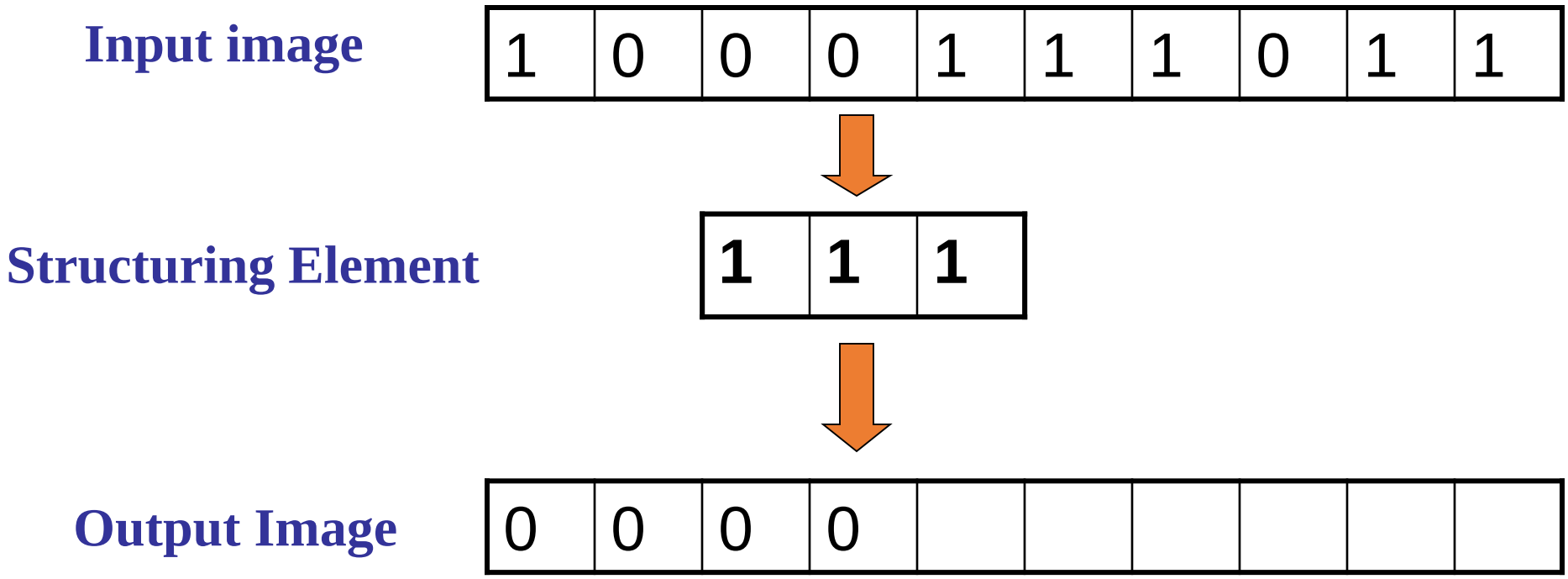
Slide credit: Kristen Grauman

Example for Erosion



Slide credit: Kristen Grauman

Example for Erosion



Slide credit: Kristen Grauman

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	1	1

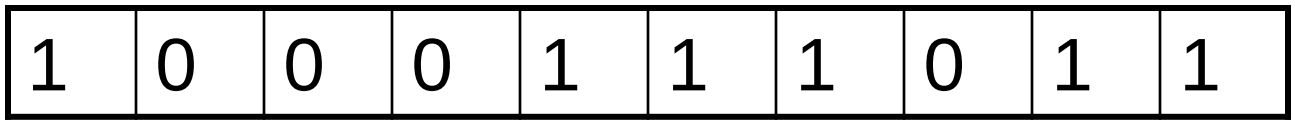
Output Image

0	0	0	0	0					

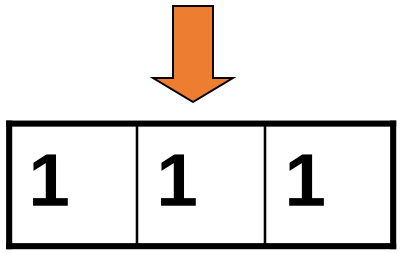
Slide credit: Kristen Grauman

Example for Erosion

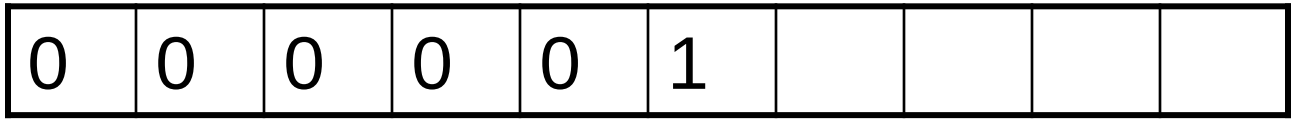
Input image



Structuring Element

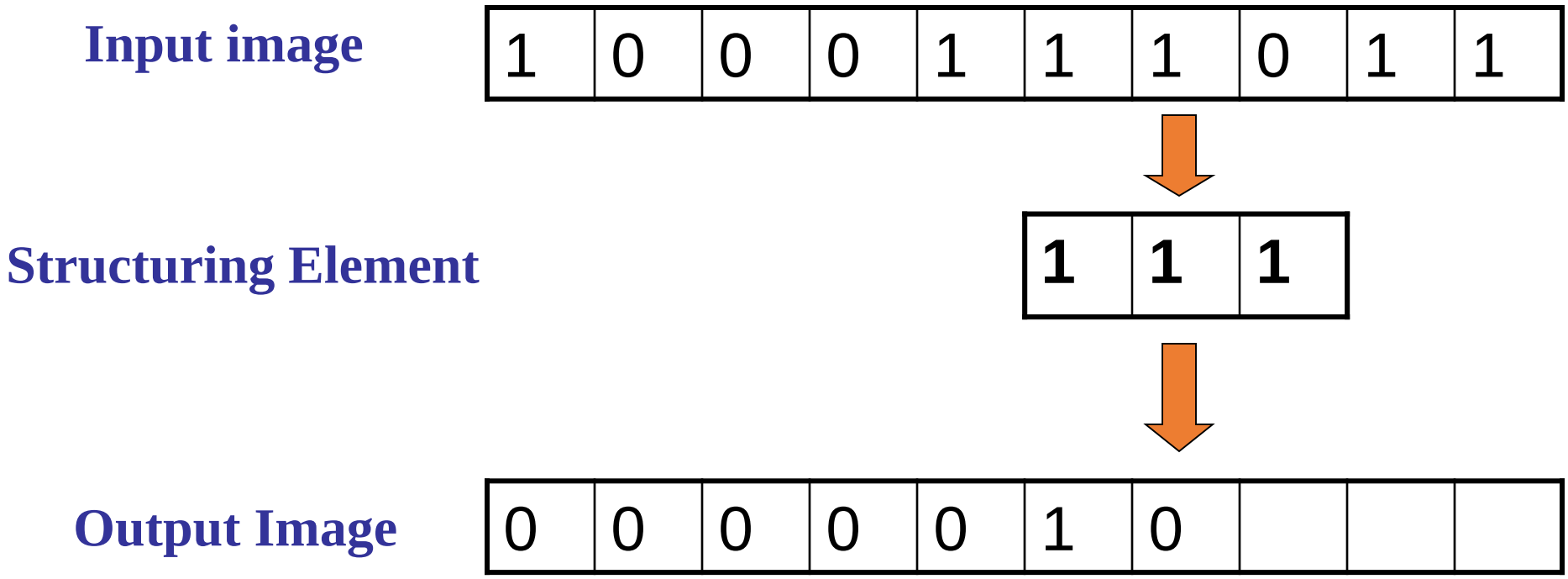


Output Image



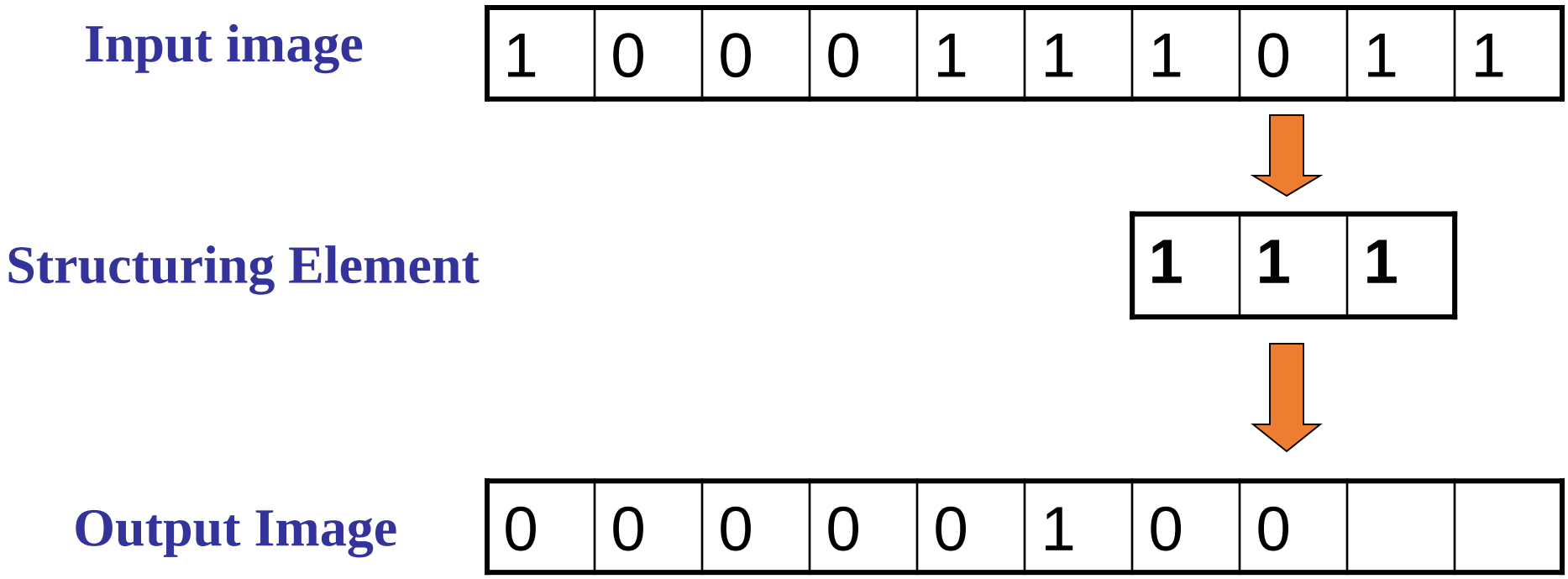
Slide credit: Kristen Grauman

Example for Erosion



Slide credit: Kristen Grauman

Example for Erosion



Slide credit: Kristen Grauman

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	

Slide credit: Kristen Grauman

Example for Erosion

Input image

1	0	0	0	1	1	1	0	1	1

Structuring Element

1	**1**	

Output Image

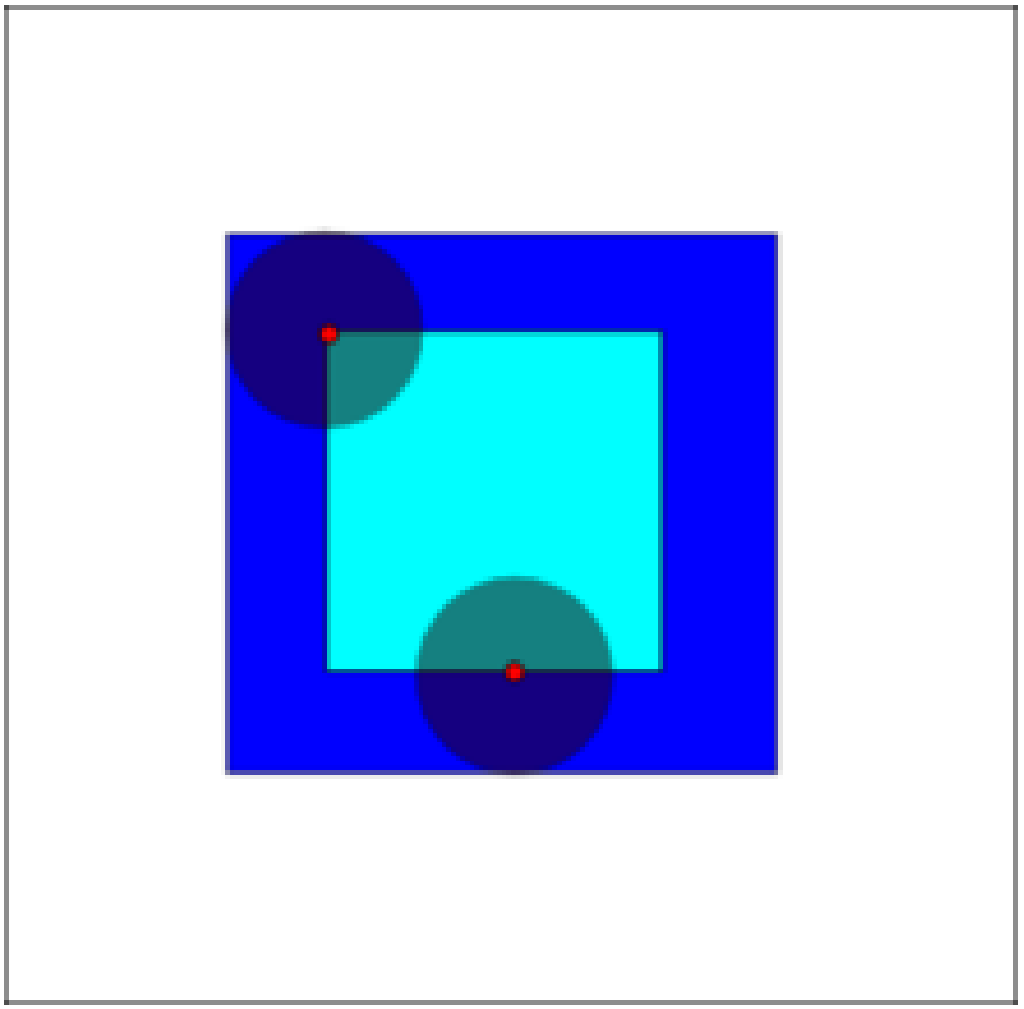
0	0	0	0	0	1	0	0	0	1

Note that the object gets smaller

```
>> help imerode
```

Slide credit: Kristen Grauman

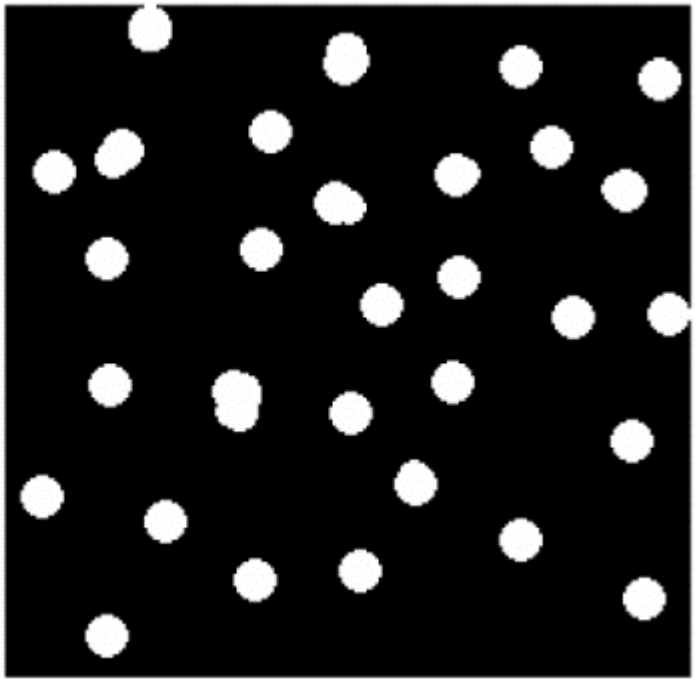
2D example for erosion



Opening

- First erode, then dilate
- Remove small objects, keep original shape

Before opening

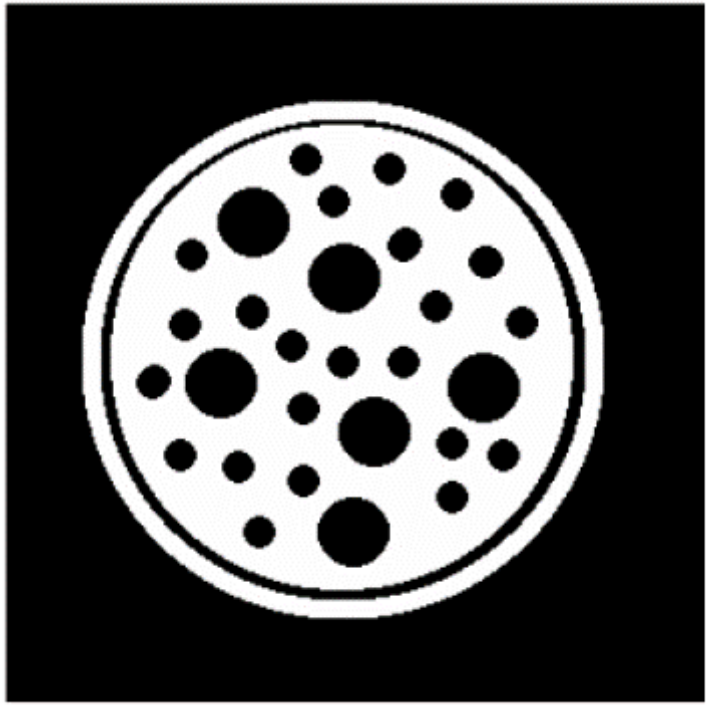


After opening

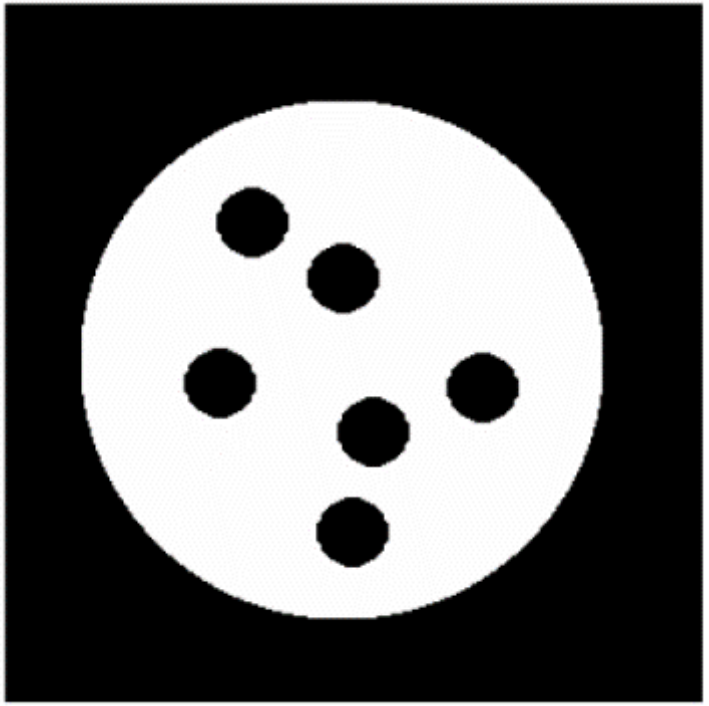
Slide credit: Kristen Grauman

Closing

- First dilate, then erode
- Fill holes, but keep original shape



Before closing



After closing

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

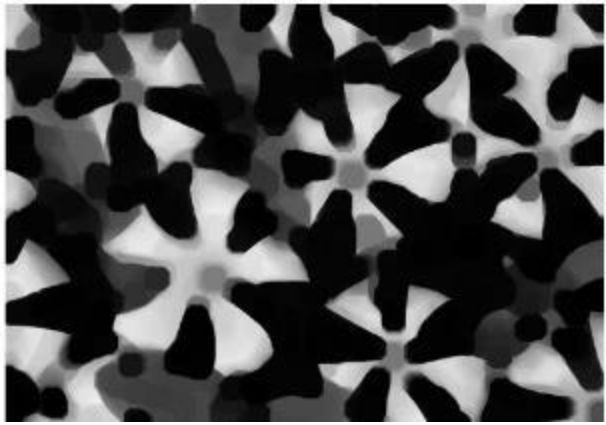
Slide credit: Kristen Grauman

Morphology operators on grayscale images

- Dilation and erosion typically performed on binary images.
- If image is grayscale:
  - Diation: take the neighborhood **MAX**
  - Erosion: take the neighborhood **MIN**

original

dilated



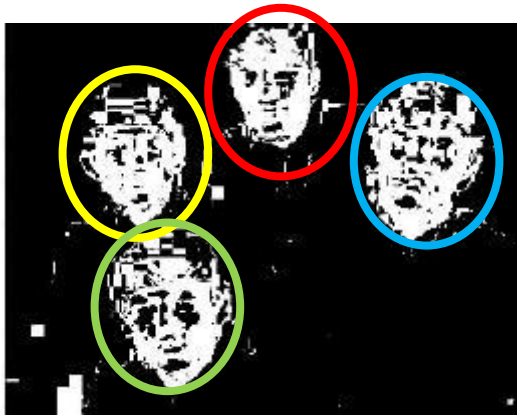
eroded

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

Connected components

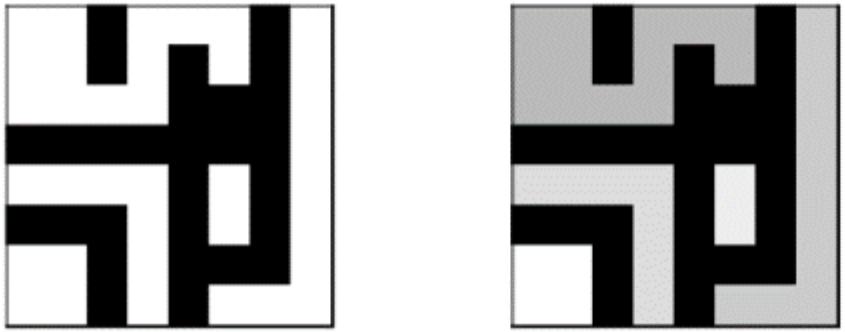
- Identify distinct regions of “connected pixels”

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

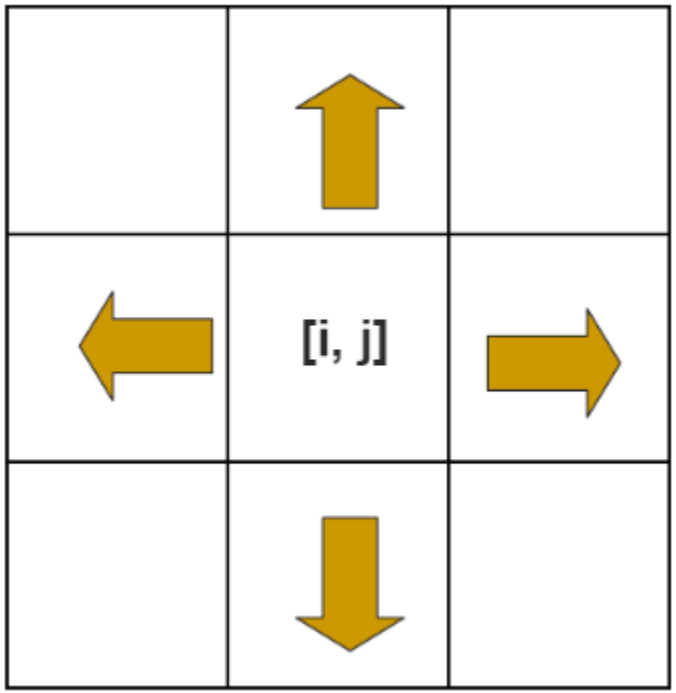


c) binary image and labeling, expanded for viewing

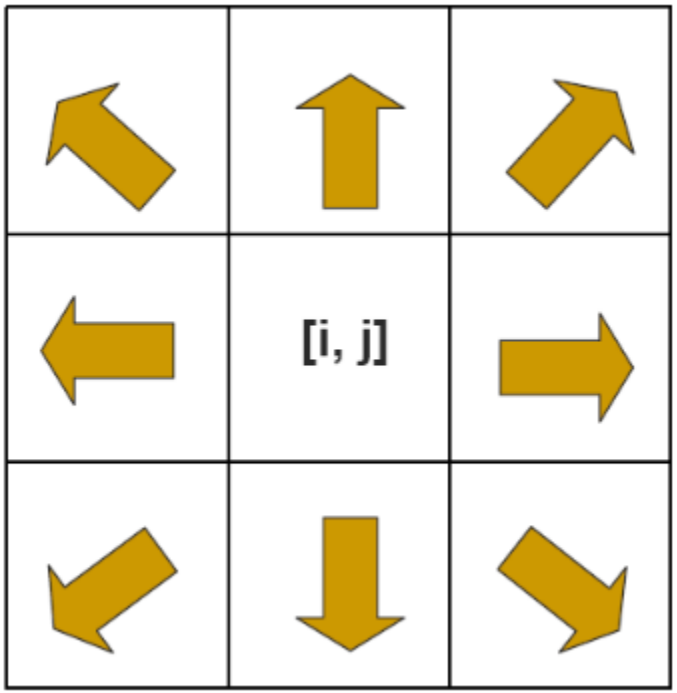
Slide credit: Shapiro and Stockman

Connectedness

- Defining which pixels are considered neighbors



4-connected



8-connected

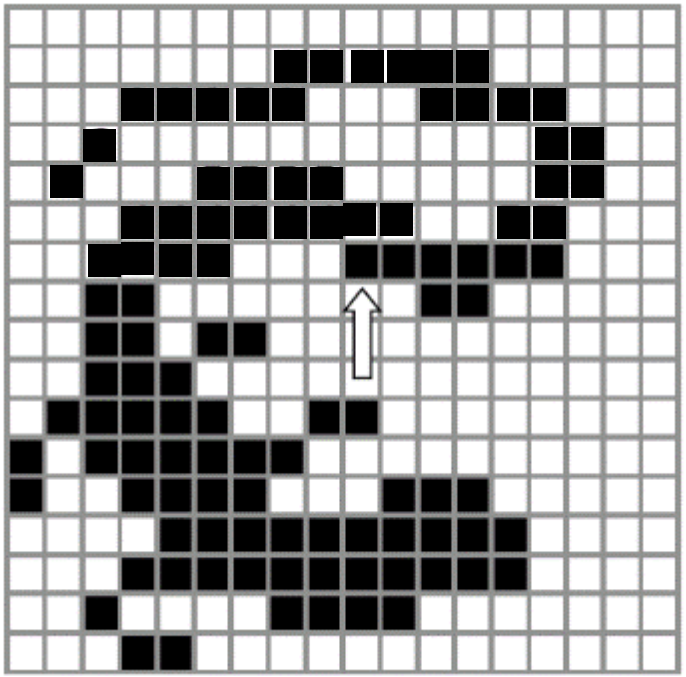
Slide credit: Chaitanya Chandra

Connected components

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

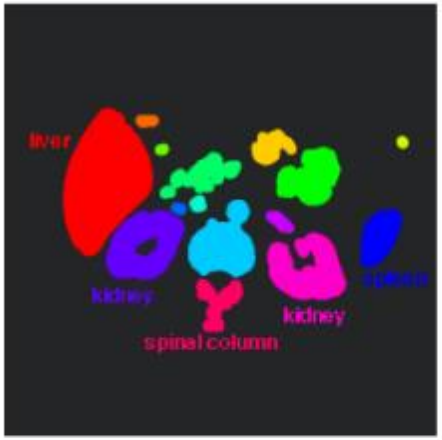
- **Input**: binary image
- **Output**: "label" image, where pixels are numbered per their component

- Note: foreground here is denoted with black pixels.

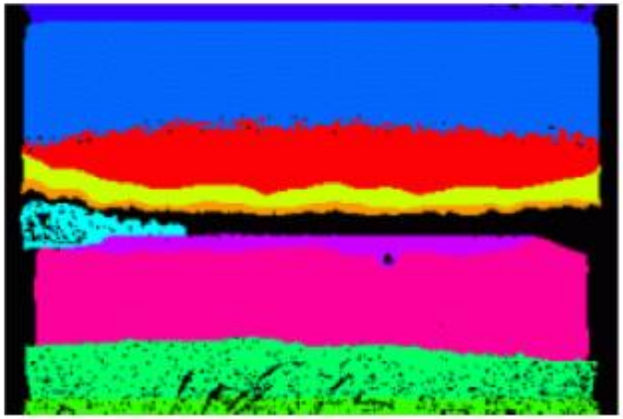


Slide credit: Kristen Grauman

Connected components



connected components of 1's from thresholded image

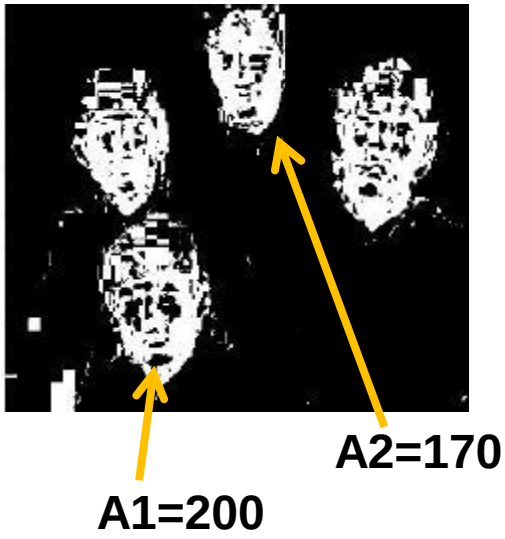


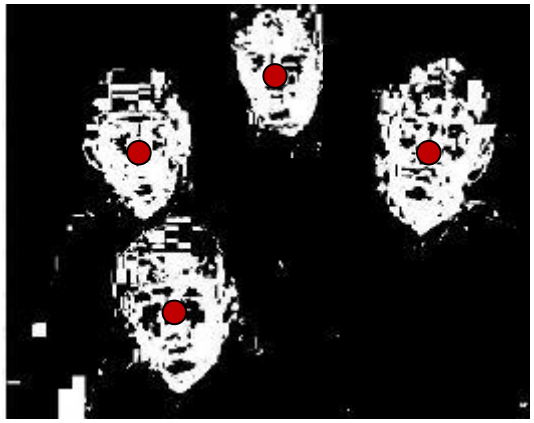
connected components of cluster labels

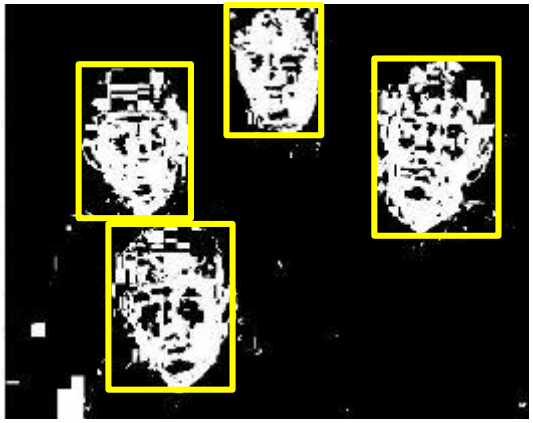
Slide credit: Pinar Duygulu

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)







Slide credit: Kristen Grauman

Circularity

- a second measure uses variation off of a circle

circularity(2):

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

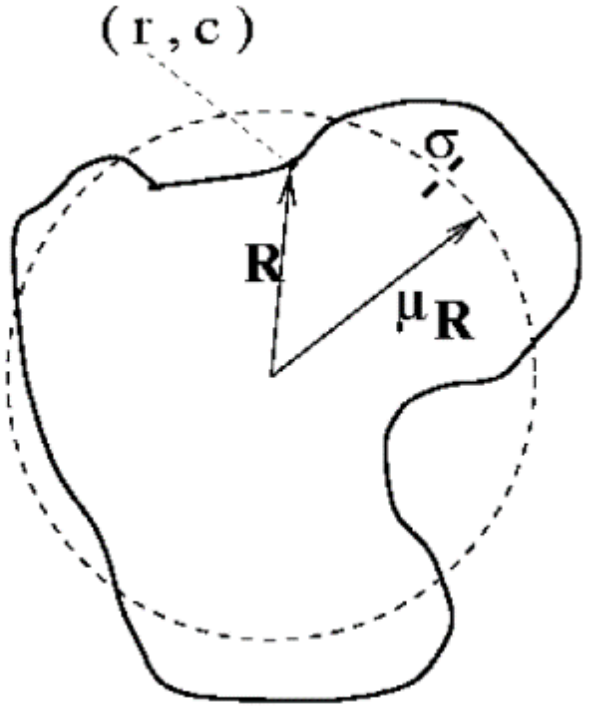
where μ_R and $\sigma_R{}^2$ are the mean and variance of the distance from the centroid of the shape to the boundary pixels (r_k, c_k).

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]

Slide credit: 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

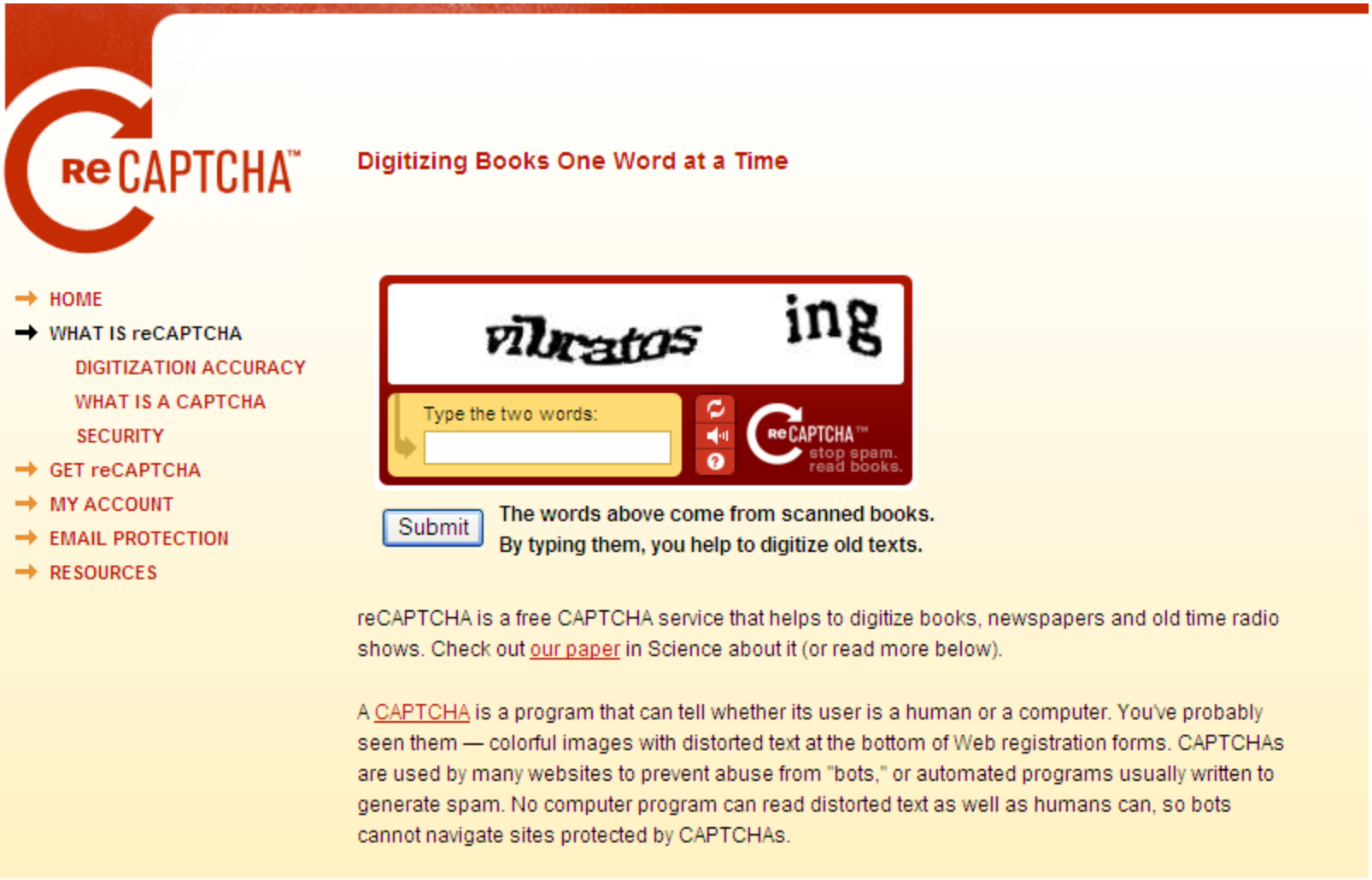
Slide credit: Kristen Grauman

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);`

Slide credit: Kristen Grauman

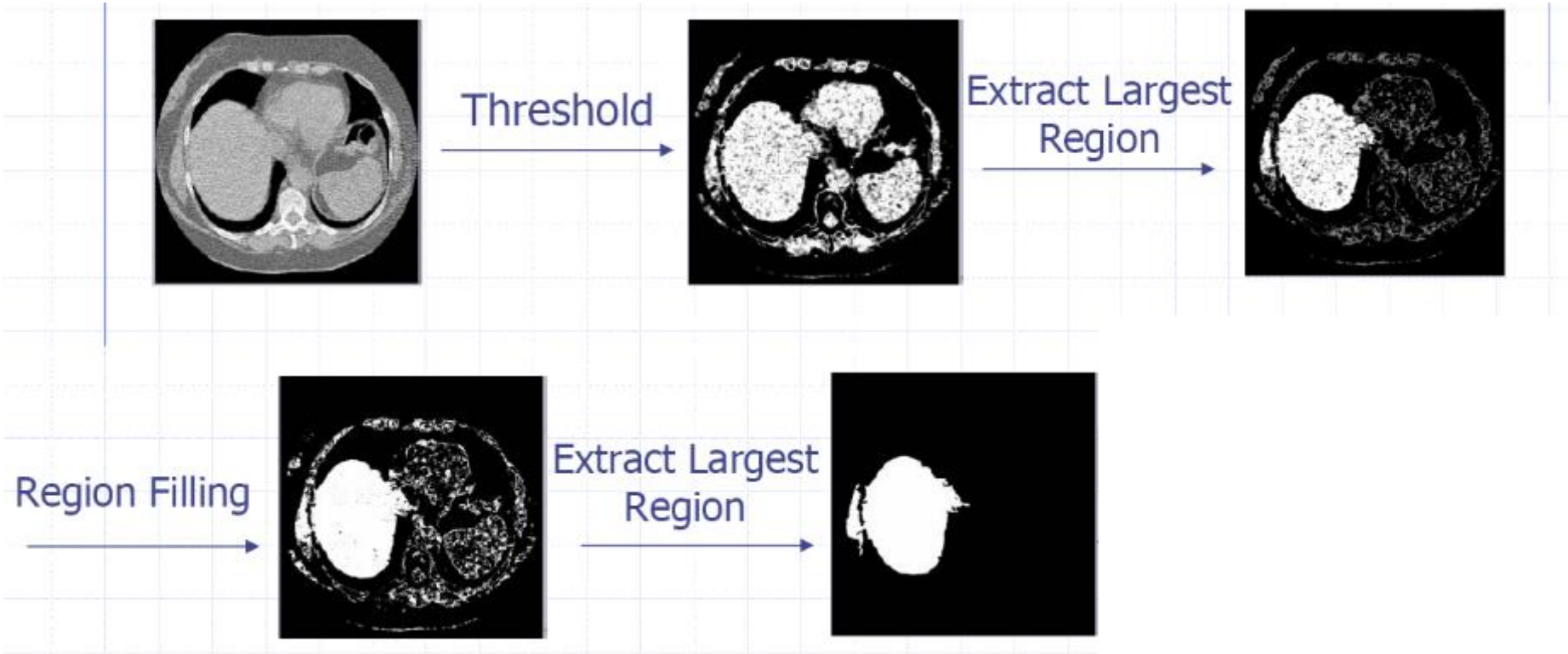
Example using binary image analysis: OCR



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

Slide credit: Kristen Grauman

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

University of Southern California
http://iris.usc.edu/Projects/detect/detection-result.html

Slide credit: Kristen Grauman

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

Slide credit: Kristen Grauman

OpenCV Demo

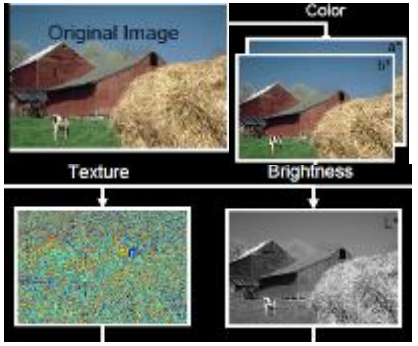
Things to remember

- Canny edge detector =
  smooth → derivative → thin → threshold → link

- Pb: learns weighting of gradient, color, texture differences
  More recent learning approaches give at least as good accuracy and are faster

- Straight line detector =
  canny + gradient orientations → orientation binning → linking → check for straightness

- Binary image analysis
  thresholding, dialation, erosion



Next classes: Correspondence and Alignment

- Detecting interest points
- Tracking points
- Object/image alignment and registration
  - Aligning 3D or edge points
  - Object instance recognition
  - Image stitching