计算机视觉表征与识别
Chapter 5: Edges

王利民

媒体计算课题组

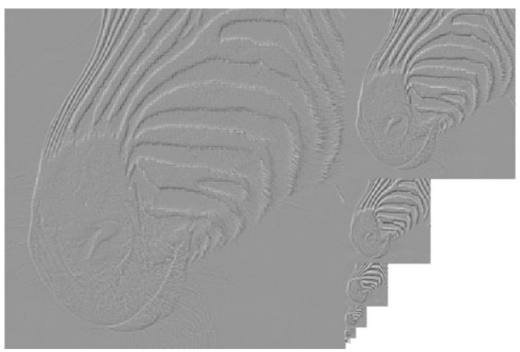
http://mcg.nju.edu.cn/

Image pyramids

- Gaussian

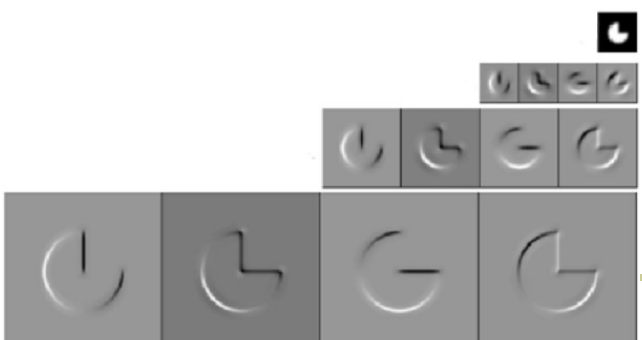
Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- Laplacian



Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- Steerable pyramid



Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis.

Linear Image Transforms

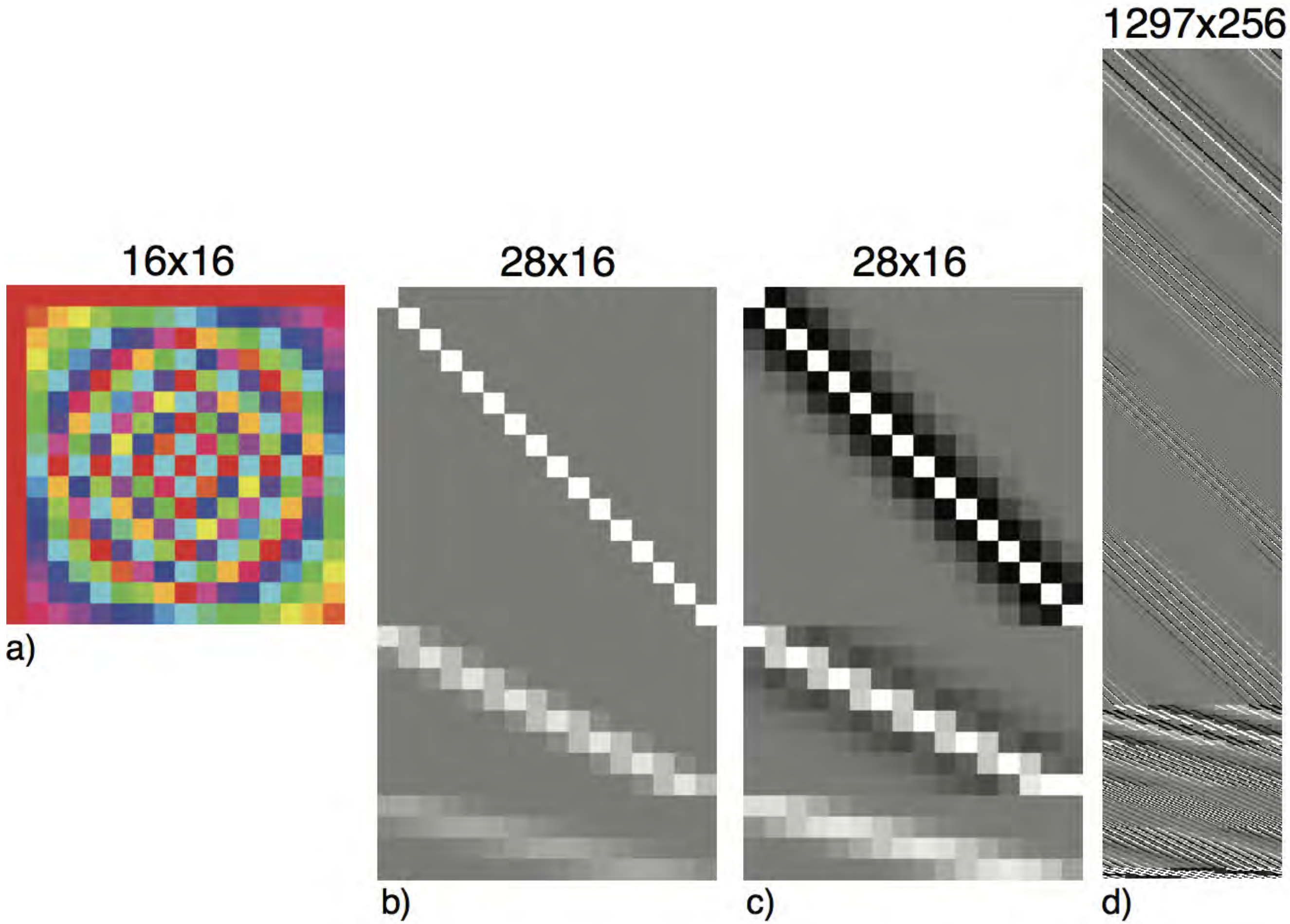


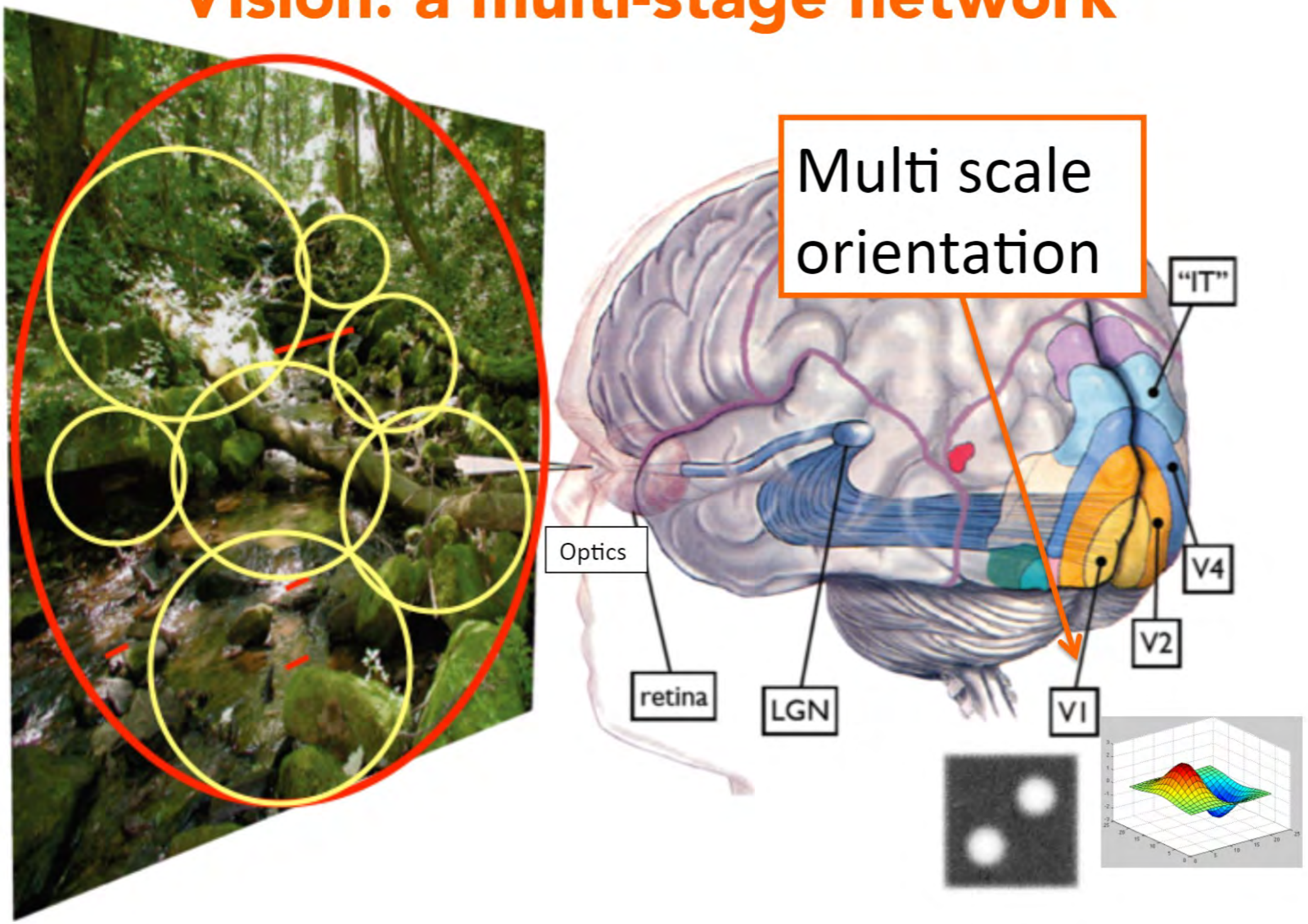
Image representation

- Pixels: great for spatial resolution, poor access to frequency

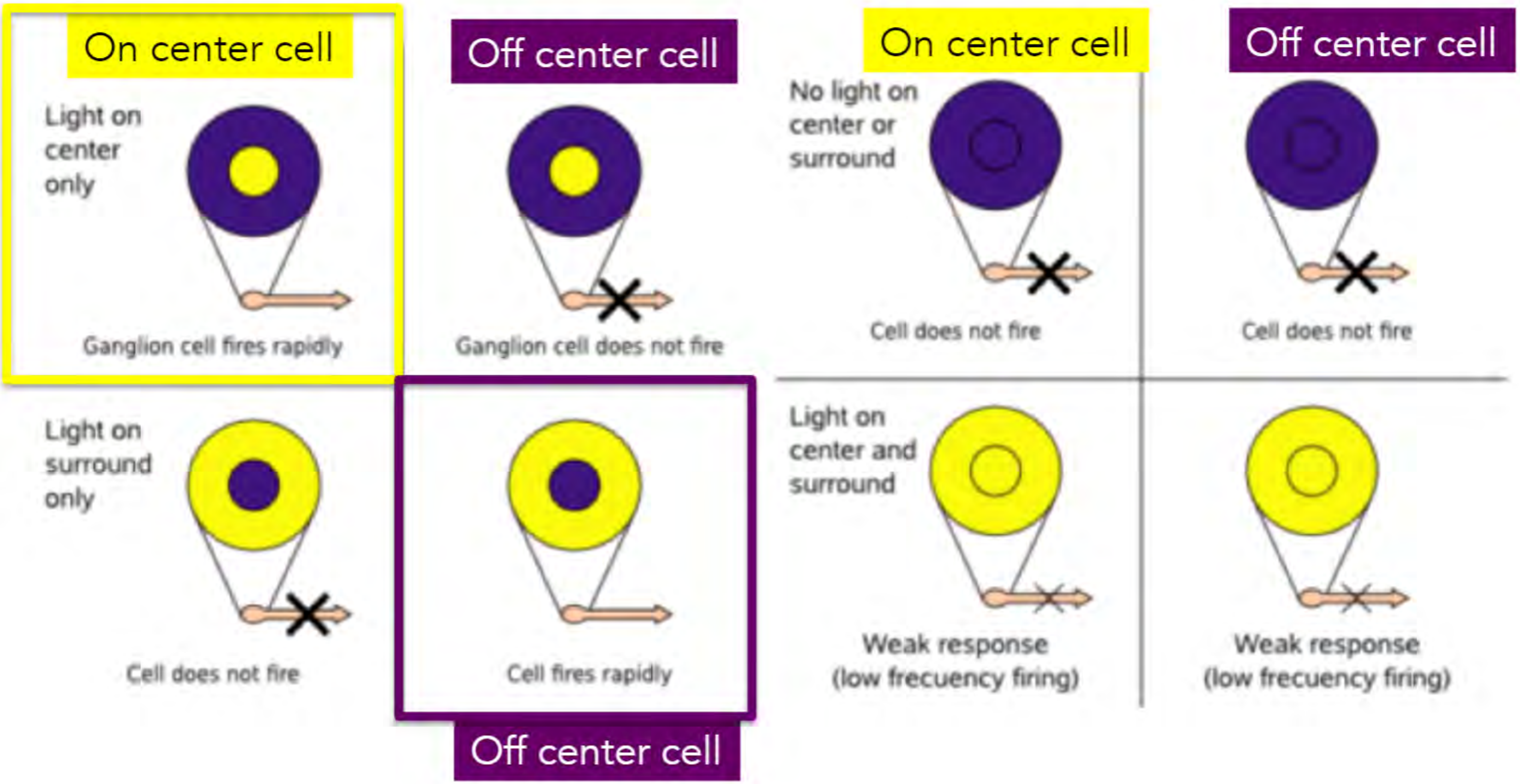
- Fourier transform: great for frequency, not for spatial information

- Pyramids/filter banks: balance between spatial and frequency information

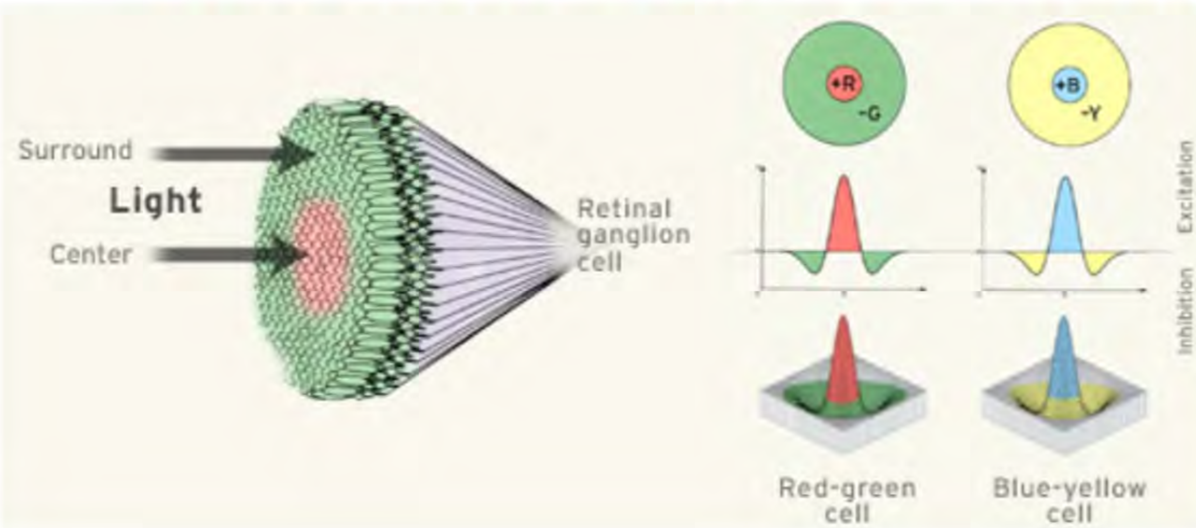
Vision: a multi-stage network

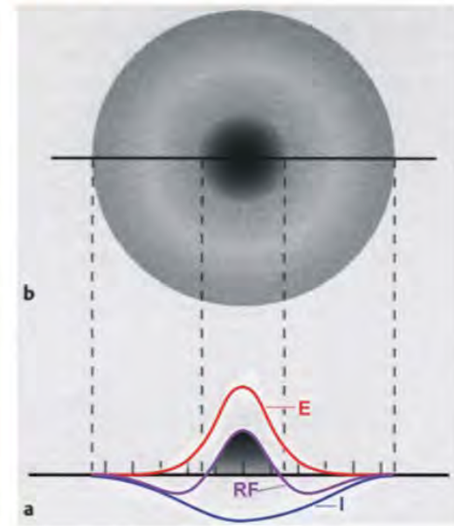


Model of Retinal and LGN cells



Difference of Gaussian Model

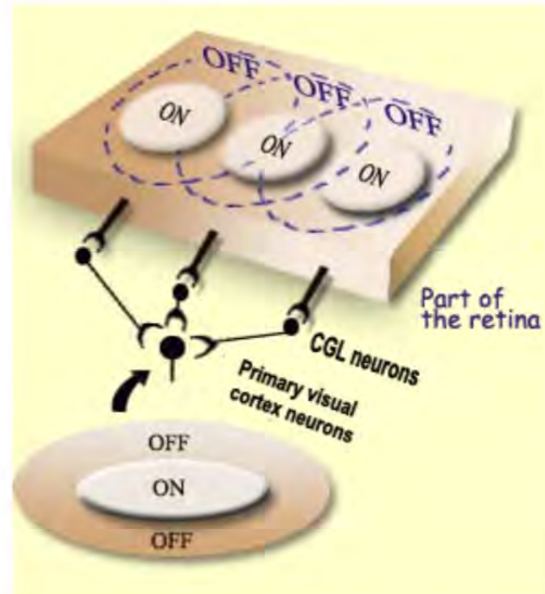


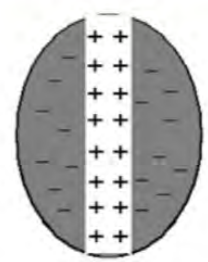


- The output of a retinal ganglion cell is a weighted sum of its inputs. You can treat the cell as a shift-invariant linear system
- **Difference of Gaussian (DoG):** A model of the operation performed by retinal and ganglion cells (LGN). The DoG model supposes that the neural response results from the combined signal of two separates mechanisms
- LGN and retinal neurons have circular receptive fields: **they respond equally well to all stimulus orientations, and at different spatial frequency (scale)**
- Multiple DoG is called a **Laplacian Pyramid**

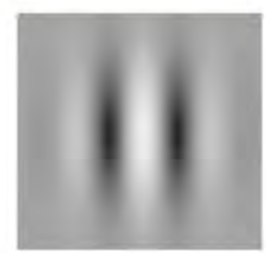
Oriented representation in V1

- Simple-cell receptive fields (RFs) are constructed from the output of LGN cells
- They are selective to **oriented contours** and edges





RECEPTIVE FIELD STRUCTURE

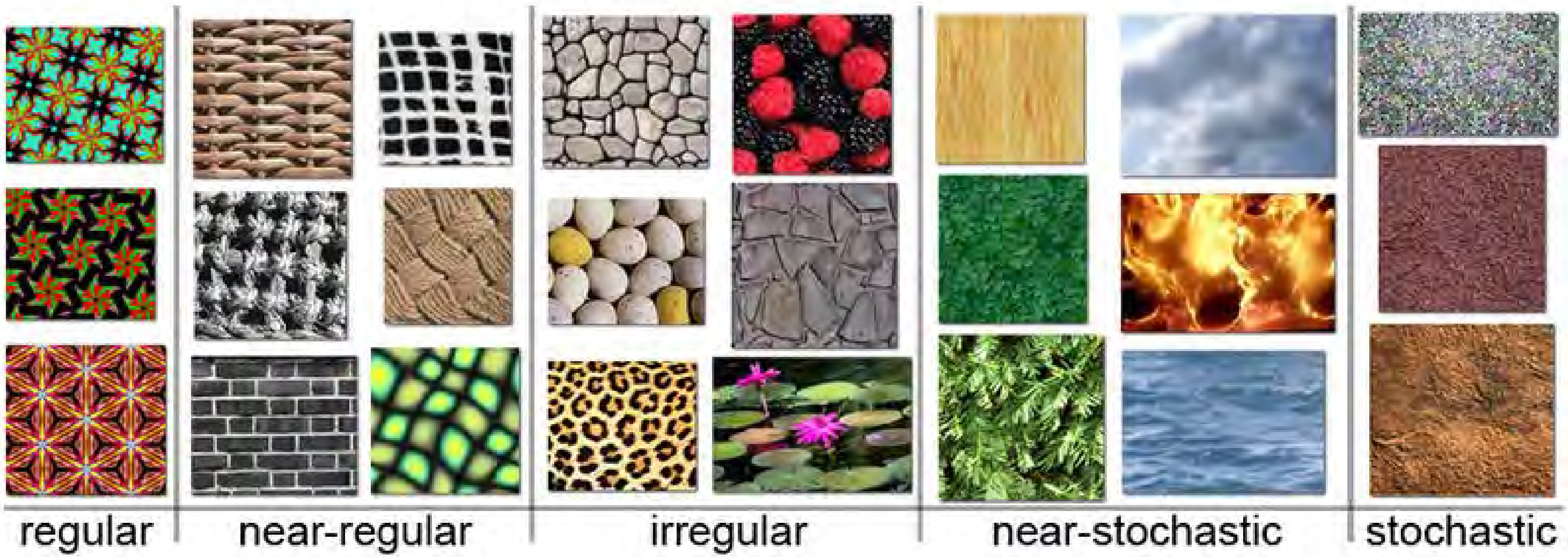


GABOR PATCH

What is texture?

- **Texture is a phenomenon that is widespread, easy to recognize, and hard to define.**
- Views of large numbers of small objects
- Regular or stochastic patterns caused by bumps, grooves, and/or markings
- Textures tend to show repetition: the same local patch appears again and again.

Texture overview



Why analyze texture?

Importance to perception:

- Often indicative of a material’s properties
- Can be important appearance cue, especially if shape is similar across objects
- Aim to distinguish between shape, boundaries, and texture

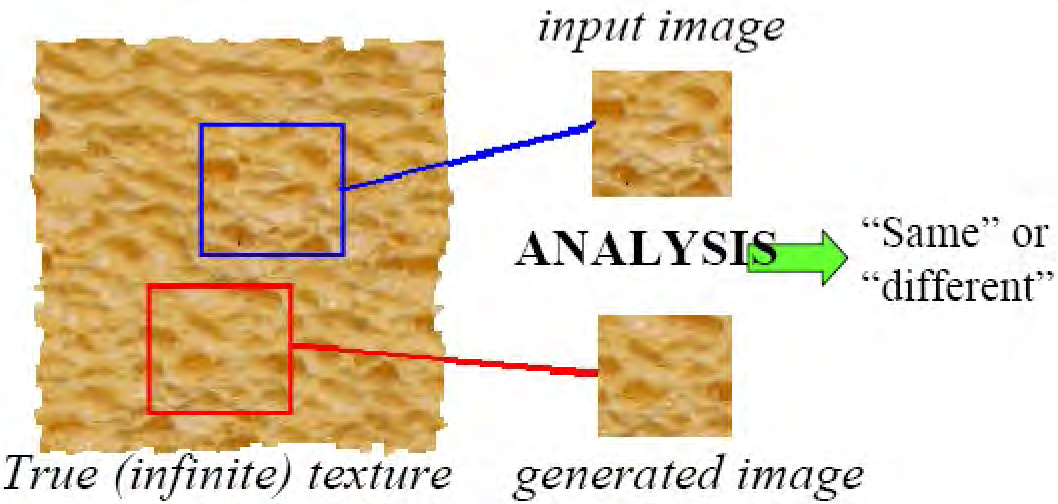
Technically:

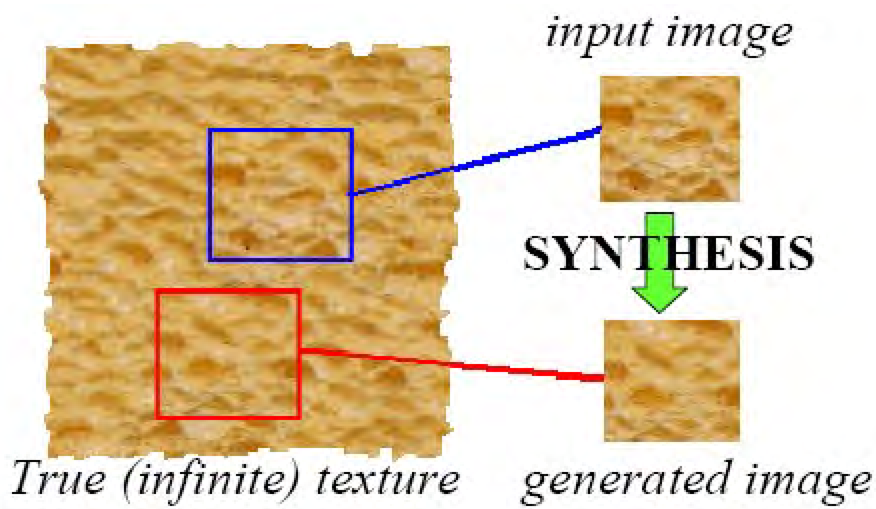
- Representation-wise, we want a feature one step above “building blocks” of filters, edges.

Texture-related tasks

- **Shape from texture**
  - Estimate surface orientation or shape from image texture
- **Segmentation/classification** from texture cues
  - Analyze, represent texture
  - Group image regions with consistent texture
- **Synthesis**
  - Generate new texture patches/images given some examples

Analysis vs. Synthesis





Images:Bill Freeman, A. Efros

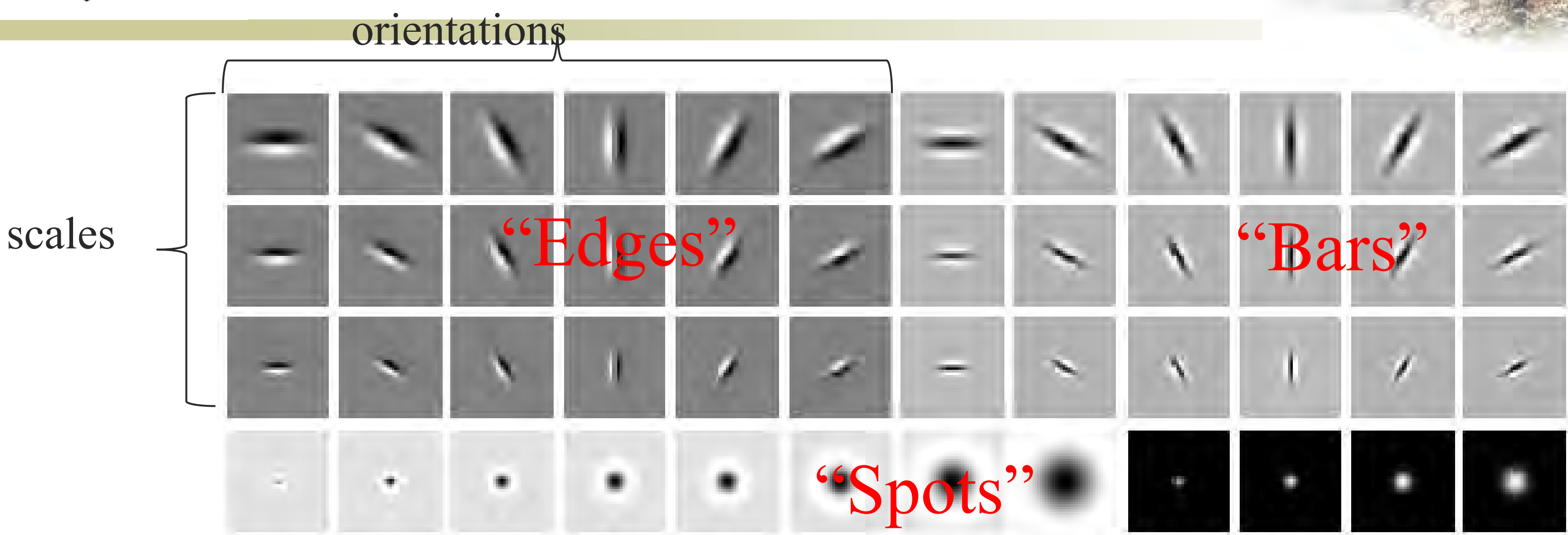
Texture representation

- Textures are made up of repeated local patterns, so:
  - Find the patterns
    - Use filters that look like patterns (spots, bars, raw patches…)
    - Consider magnitude of response
  - Describe their statistics within each local window, e.g.,
    - Mean, standard deviation
    - Histogram
    - Histogram of “prototypical” feature occurrences

Filter banks

- Our previous example used two filters, and resulted in a 2-dimensional feature vector to describe texture in a window.
  - x and y derivatives revealed something about local structure.
- We can generalize to apply a collection of multiple (*d*) filters: a “filter bank”
- Then our feature vectors will be *d*-dimensional.
  - still can think of nearness, farness in feature space

Filter banks



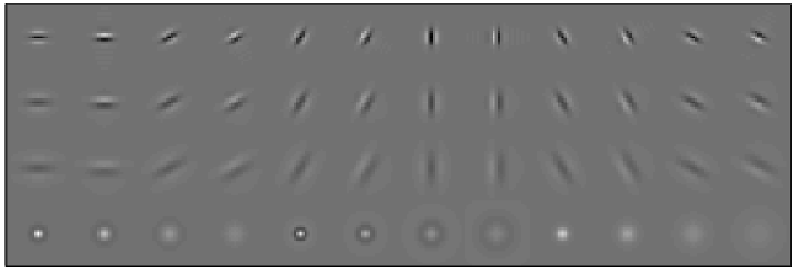
- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

2D Textons

- Goal: find canonical local features in a texture;

  1) Filter image with linear filters:

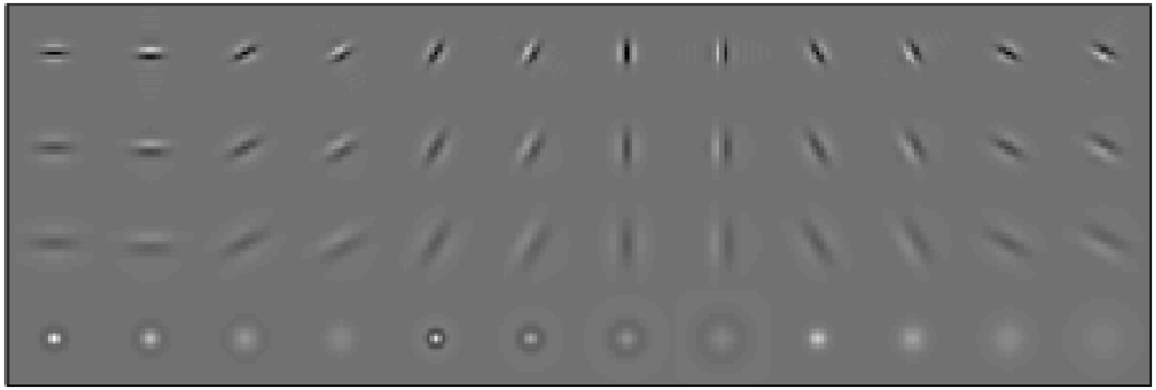


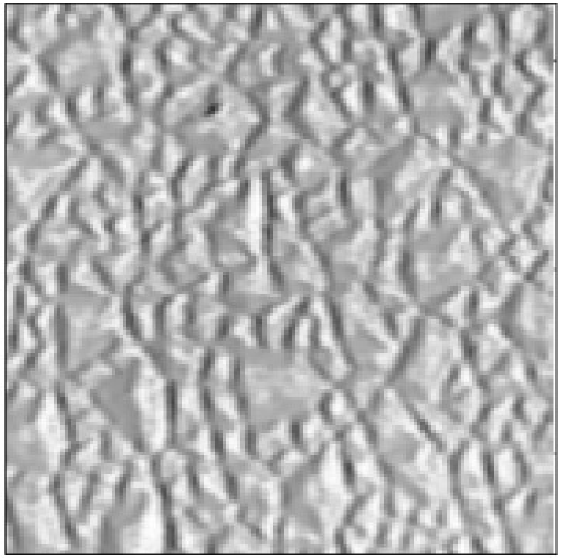
  2) Vector quantization (k-means) on filter outputs;

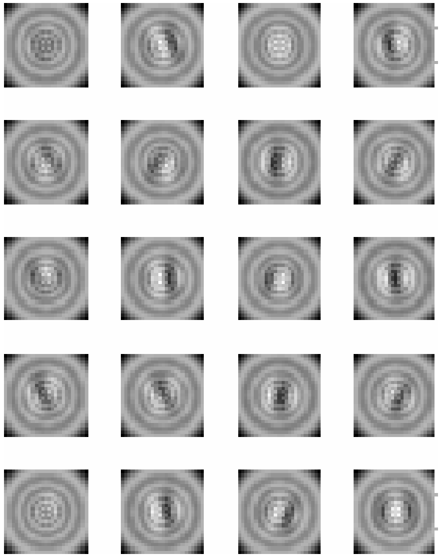
  3) Quantization centers are the textons.

- Spatial distribution of textons defines the texture;

2D Textons (cont’d)







Texture/Material Representation

- Each material is now represented as a spatial arrangement of symbols from the texton vocabulary;
- Recognition： ignore spatial arrangement,
  use histogram (K=100);

Similarity of materials

- Similarity between histograms measured using chi-square difference:

$$\chi^2(h_1, h_2) = \sum_{n=1}^{N} \frac{(h_1(n) - h_2(n))^2}{h_1(n) + h_2(n)}$$

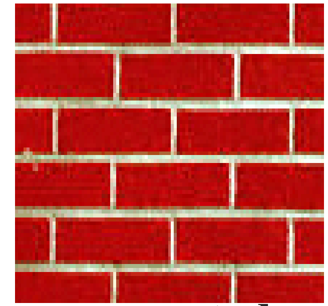
Texture synthesis

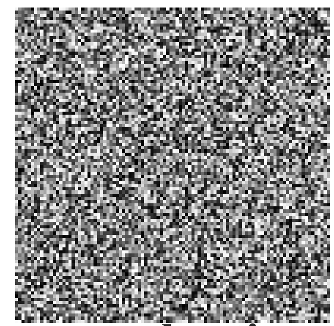
- Goal: create new samples of a given texture
- Many applications: virtual environments, hole-filling, texturing surfaces

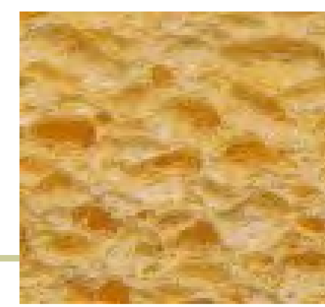
The Challenge

- Need to model the whole spectrum: from repeated to stochastic texture

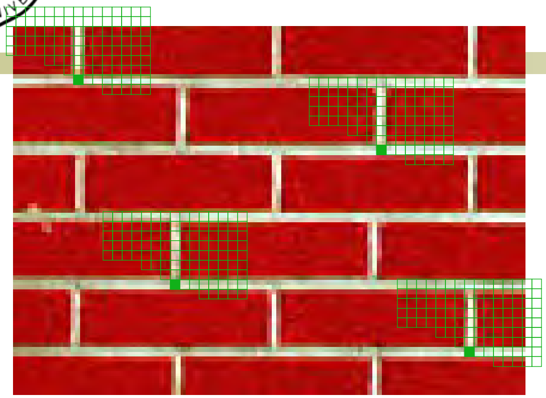
Alexei A. Efros and Thomas K. Leung, “Texture Synthesis by Non-parametric Sampling,” Proc. International Conference on Computer Vision (ICCV), 1999.


repeated

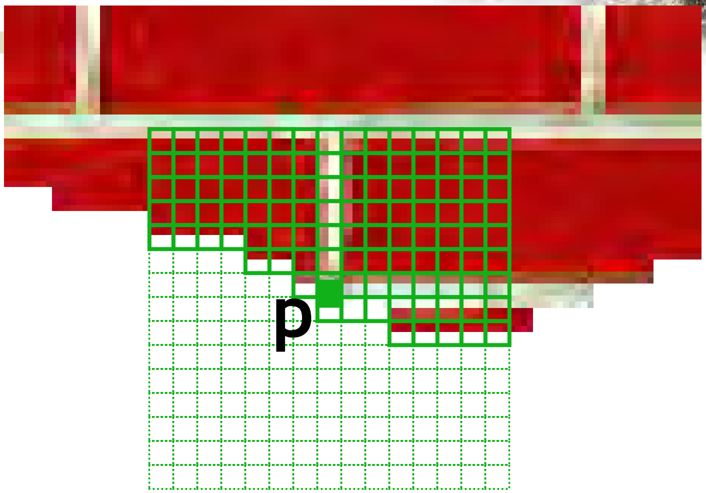

stochastic


Both?

Synthesizing One Pixel



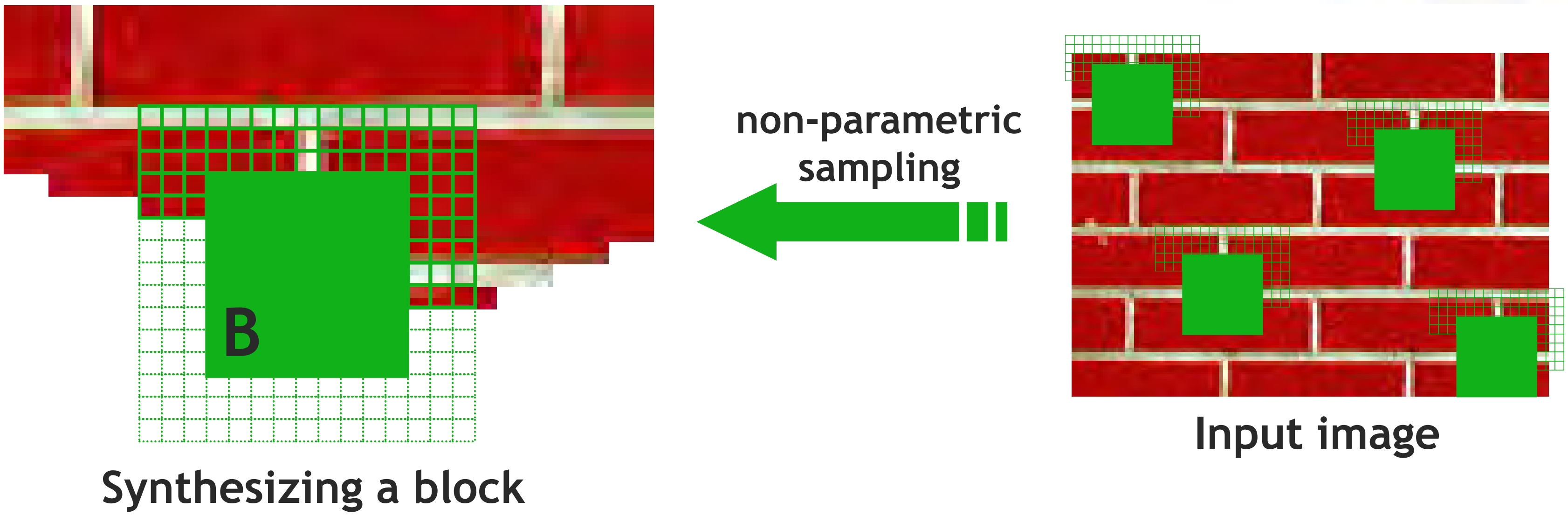
input image



synthesized image

- What is $P(\mathbf{x}|\text{neighborhood of pixels around x})$?
- Find all the windows in the image that match the neighborhood
- To synthesize **x**
  - pick one matching window at random
  - assign **x** to be the center pixel of that window

Image Quilting [Efros & Freeman 2001]



- Observation: neighbor pixels are highly correlated

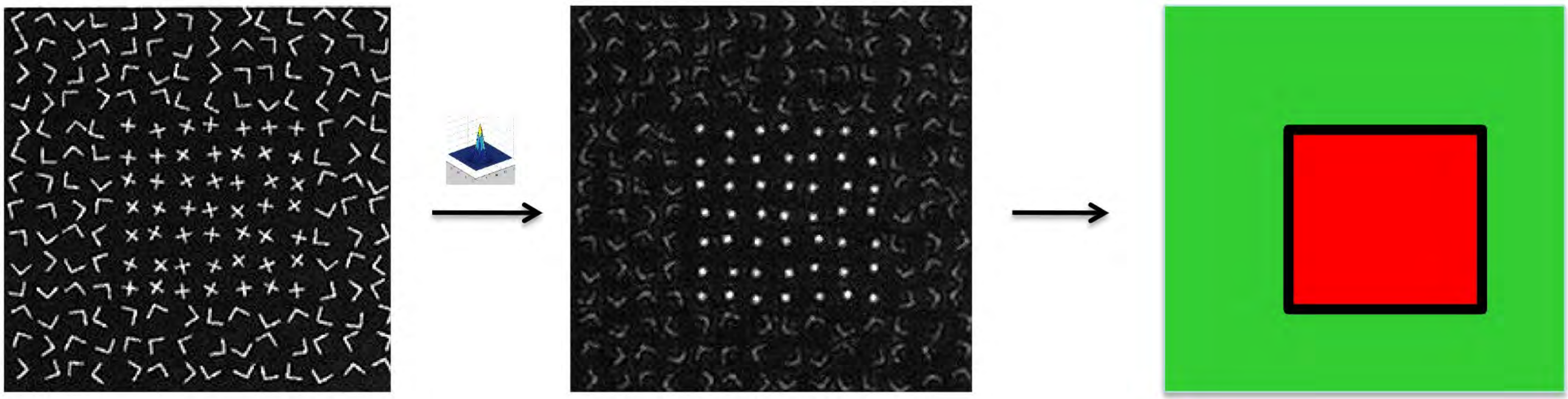
Idea: unit of synthesis = block

- Exactly the same but now we want P(**B**|N(**B**))
- Much faster: synthesize all pixels in a block at once

Today's class

- Introduction to edge detection.
- Gradients and edges.
- Canny edge detector.
- Object contour.
- Pb edge detector.
- Recent advances in edge detection.
- Straight line detection

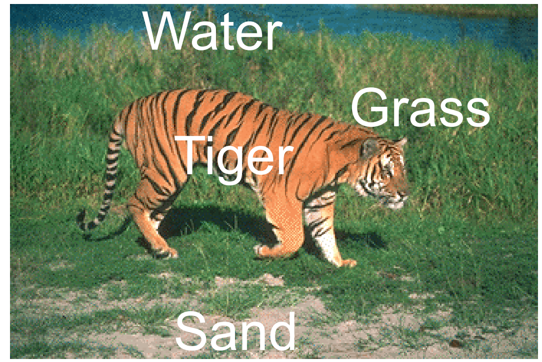
From texture to edge/segmentation

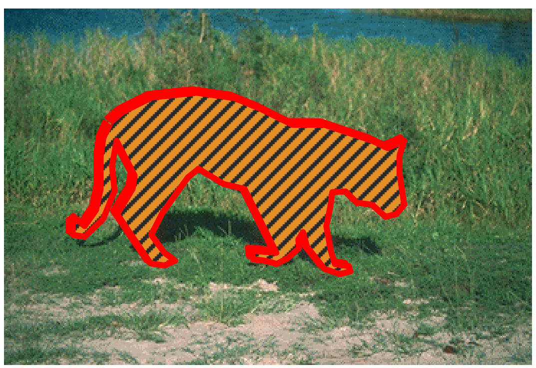


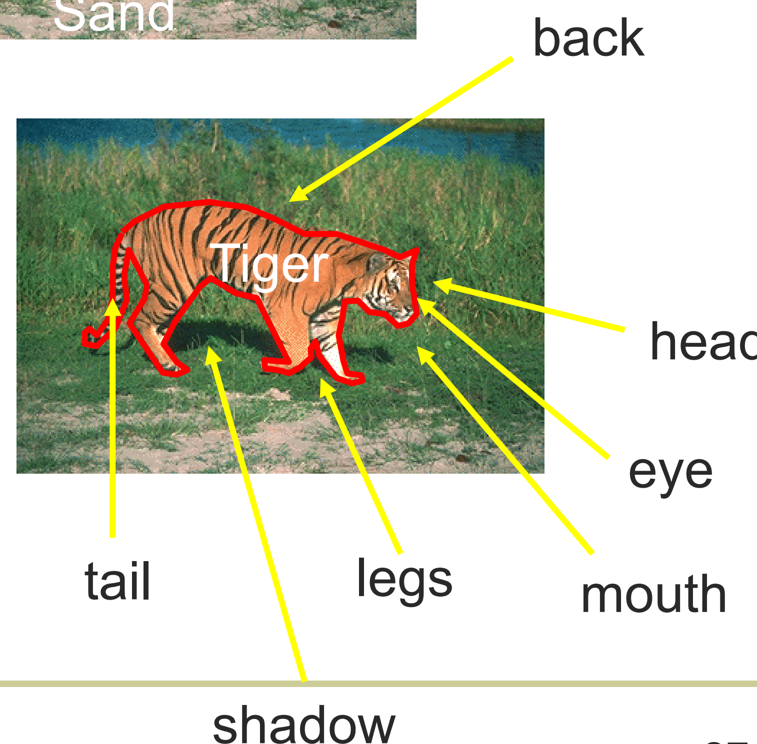
Low level Vision

Middle level Vision

From Pixels to Perception

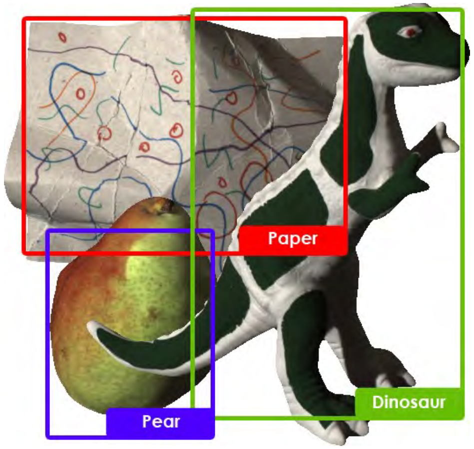




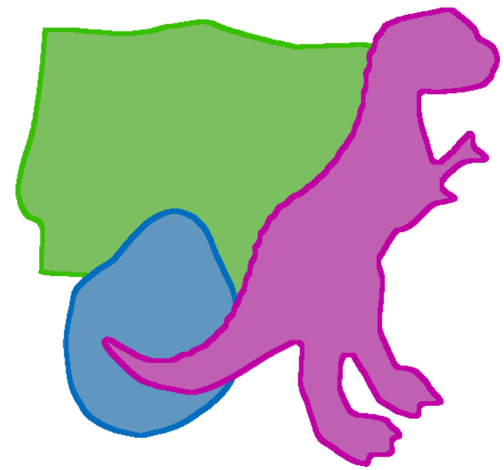


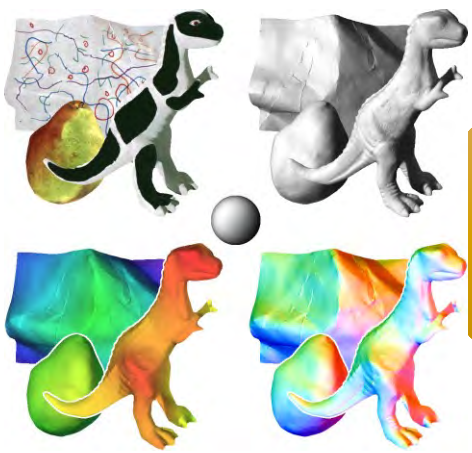
Mid-level operations of Segmentation and Grouping

Recognition, Reconstruction & Reorganization



Recognition





Reconstruction

Reorganization

Figure / Ground

Finding groups of pixels that go together (parts, objects, textures, holes)

Predicted scene categories:

forest - broadleaf (0.498), swimming hole (0.402), bayou (0.062)

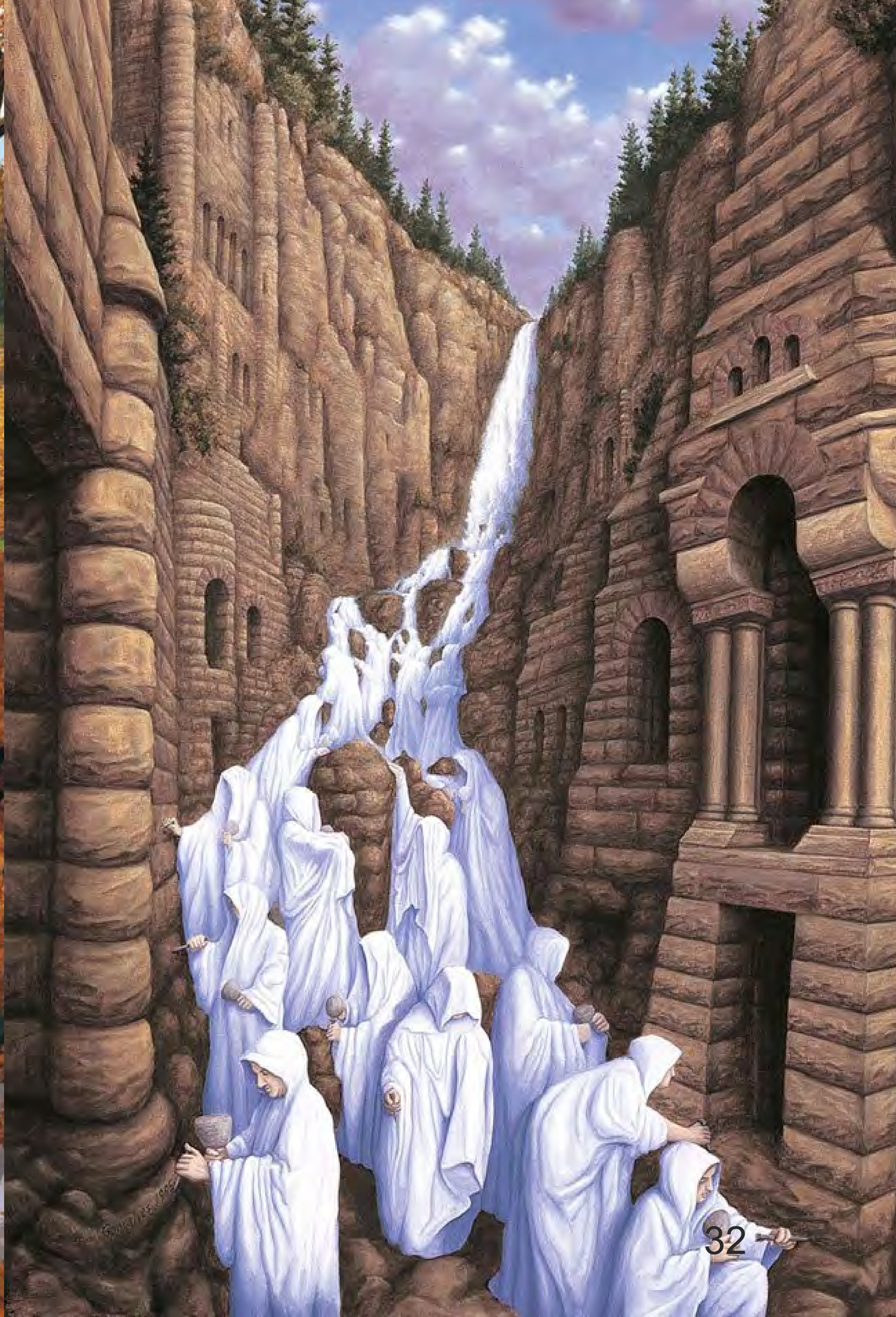
Predicted scene categories:

forest - broadleaf (0.979)

Figure / Ground

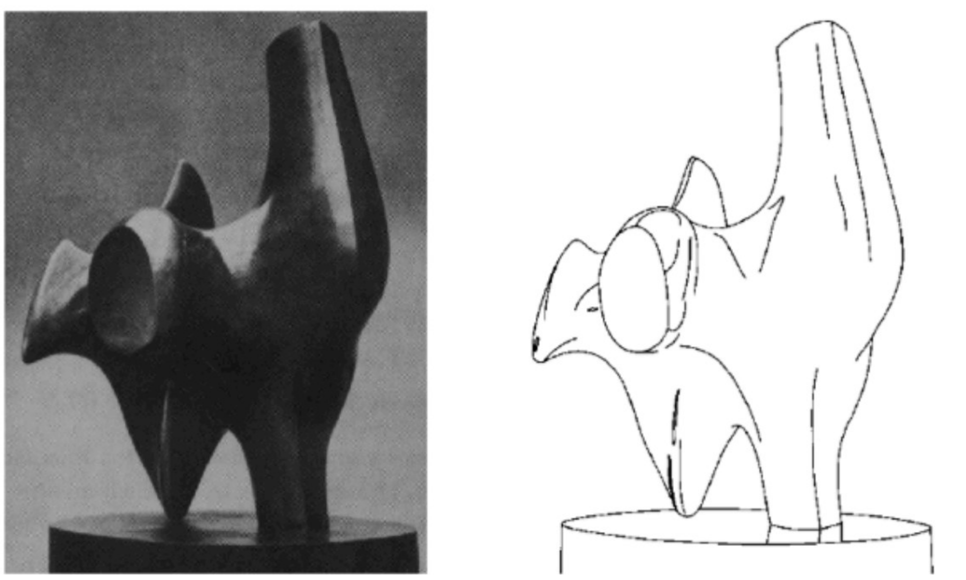
Figure / Ground

http://twistedsifter.com/2015/03/mind-bending-optical-illusion-paintings-by-rob-gonsalves/



First problem: edge detection

- Goal: compute something like a line drawing of a scene



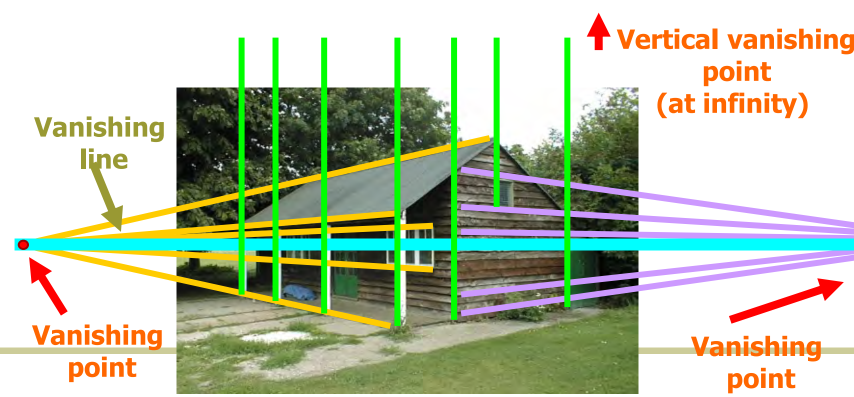
Why

- Edges reflect intrinsic properties of a scene
  - Capture shape information
  - Independent of illumination
- Our human visual system does something like this
- Good initial step for solving other problems
  - Recognition, tracking, etc.

Why

- Extract information, recognize objects

- Recover geometry and viewpoint



Issues

- No precise problem formulation
- Much harder than it seems to be
- Edge detectors usually work by detecting “big changes” in image intensity
- Boundary is contour in the image plane that represents a change in pixel ownership from object or surface to another.

Edge detection

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

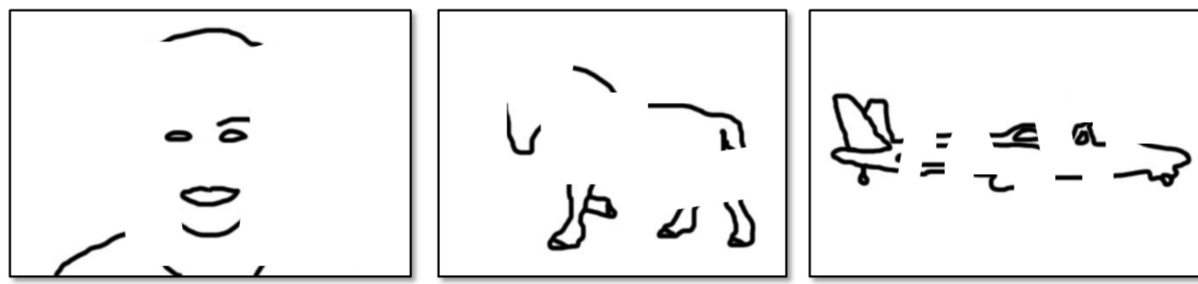
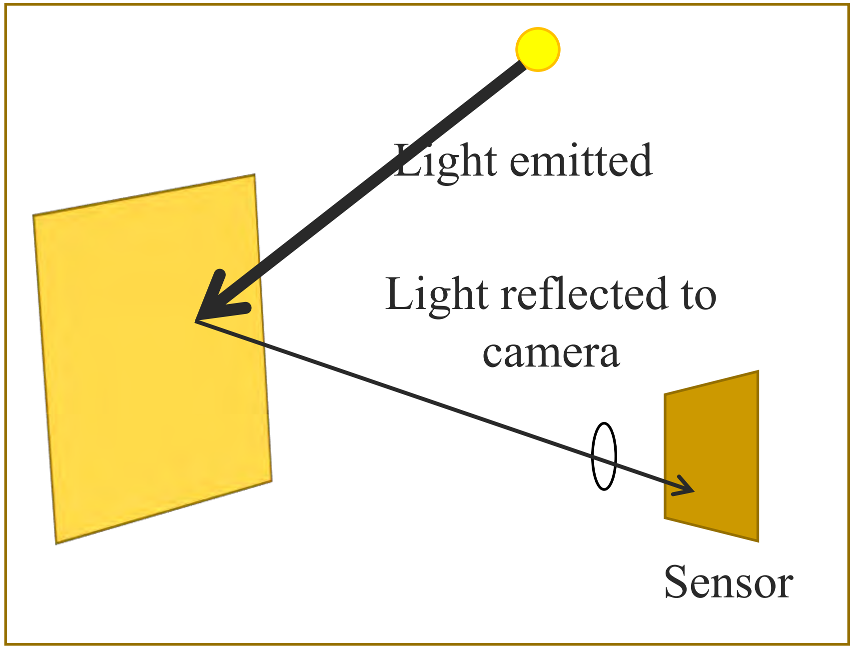


Figure from J. Shotton et al., PAMI 2007

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

Recall: How much light is recorded

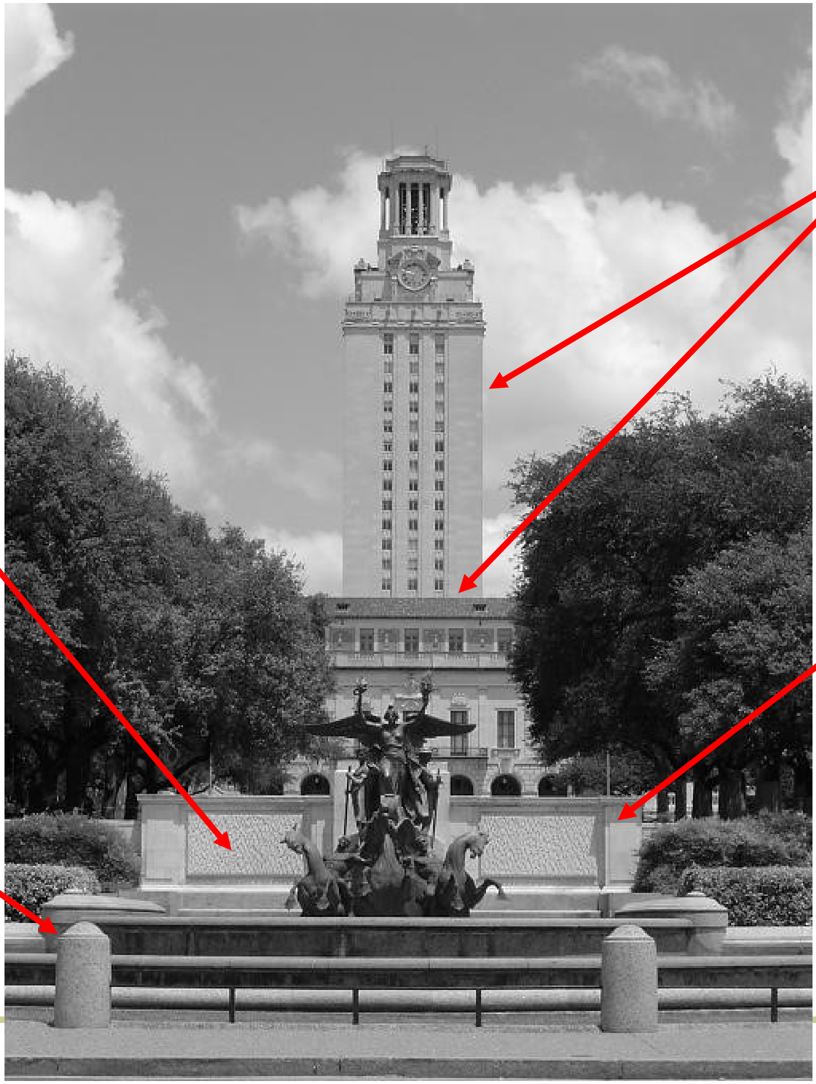
- Major factors
  - Illumination strength and direction
  - Surface geometry
  - Surface material
  - Nearby surfaces
  - Camera gain/exposure



What causes an edge?

Reflectance change: appearance information, texture

Change in surface orientation: shape

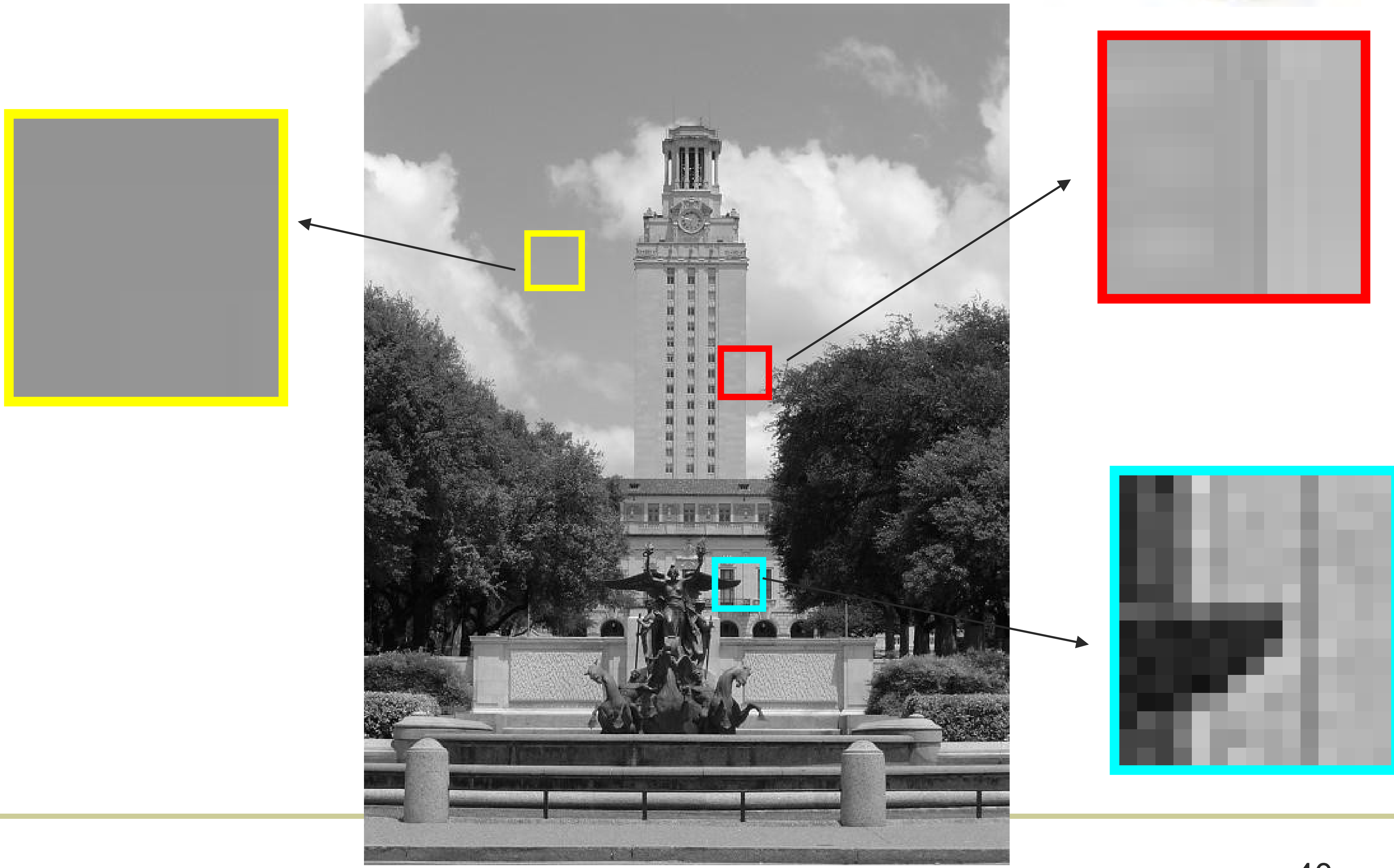


Depth discontinuity: object boundary

Cast shadows

Slide credit: Kristen Grauman

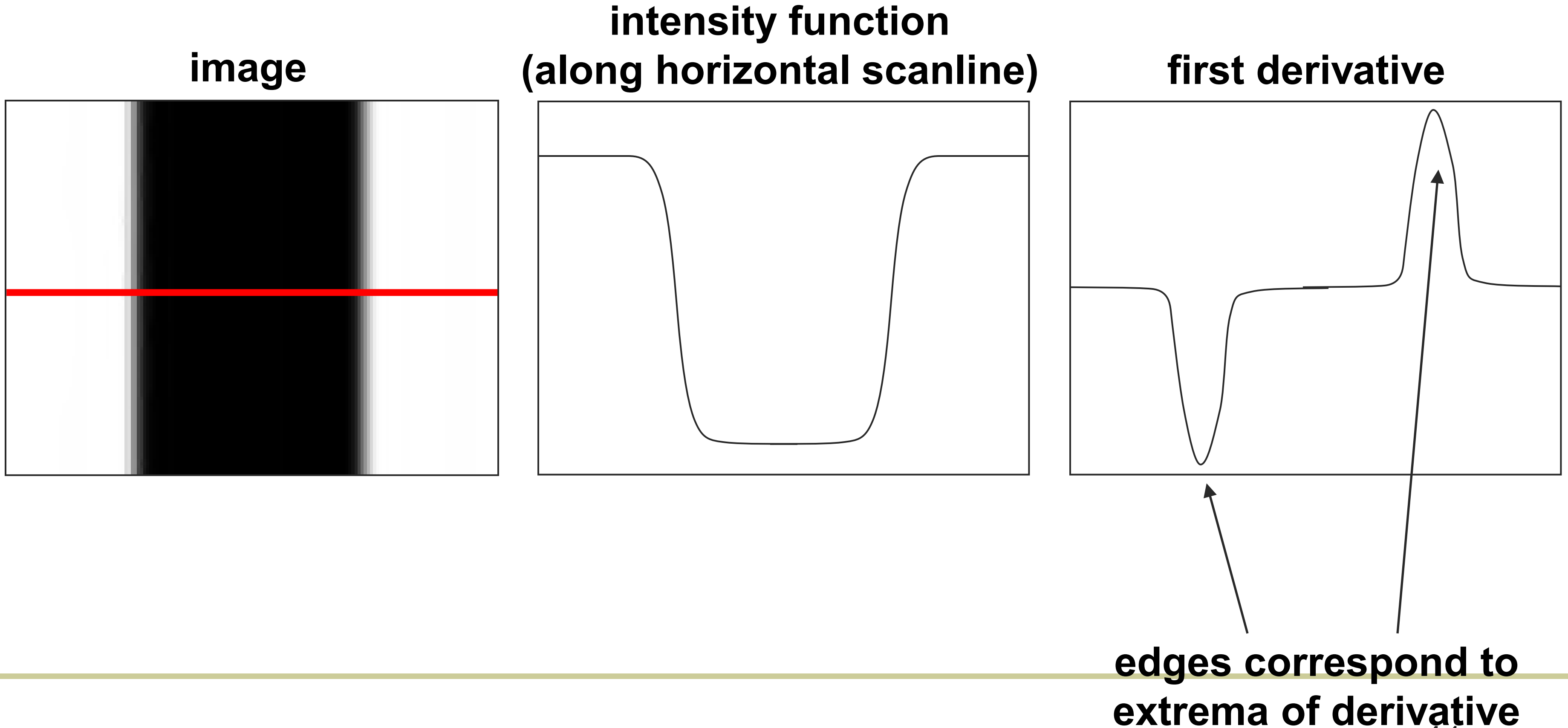
Edges/gradients and invariance



Slide credit:
Kristen Grauman

Derivatives and edges

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



Source: L. Lazebnik

Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

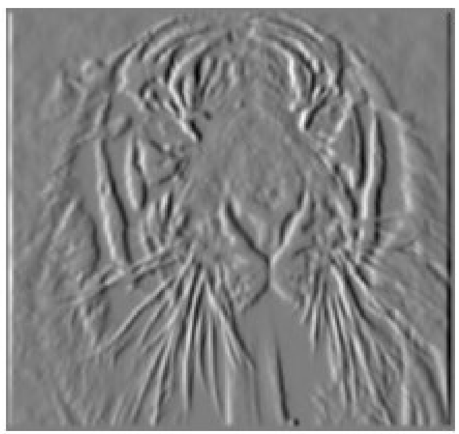
To implement above as convolution, what would be the associated filter?

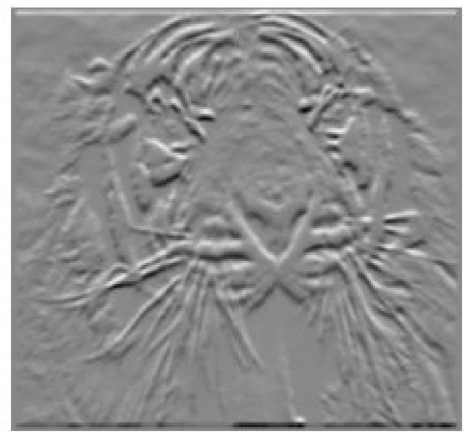
Slide credit: Kristen Grauman

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$





-1	1

-1	?	1
1	or	-1

Which shows changes with respect to x?

(showing filters for correlation)

Finite difference filters

- Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

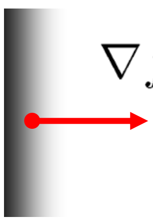
Source: K. Grauman

Image gradient

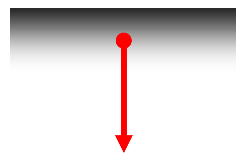
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

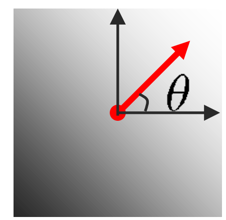
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



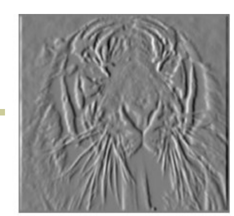
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

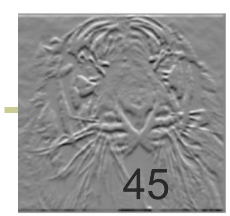
The **gradient direction** (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

The **edge strength** is given by the gradient magnitude

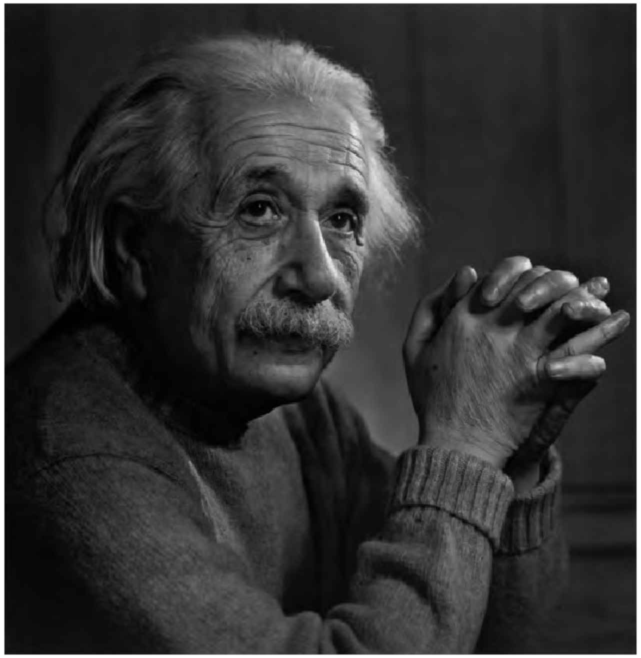
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

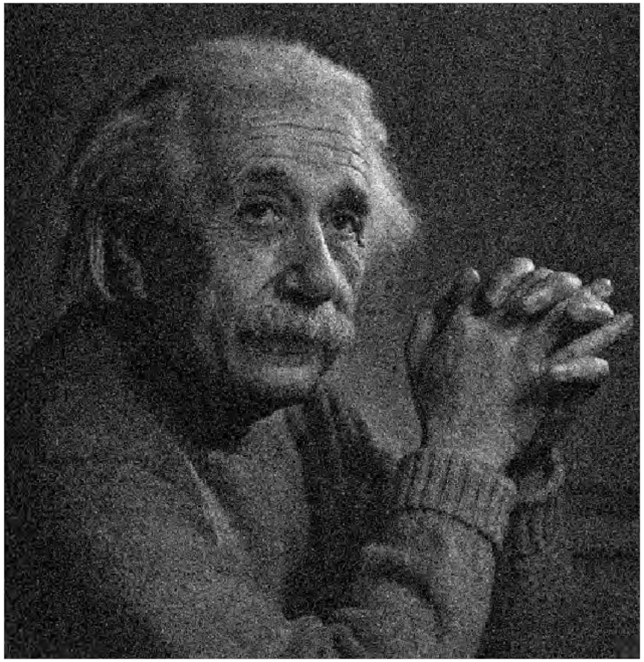




Slide credit Steve Seitz

Effects of noise



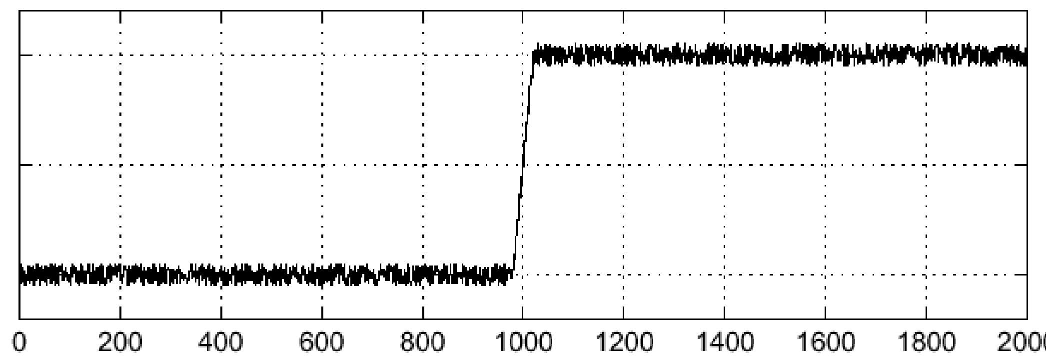


Effects of noise

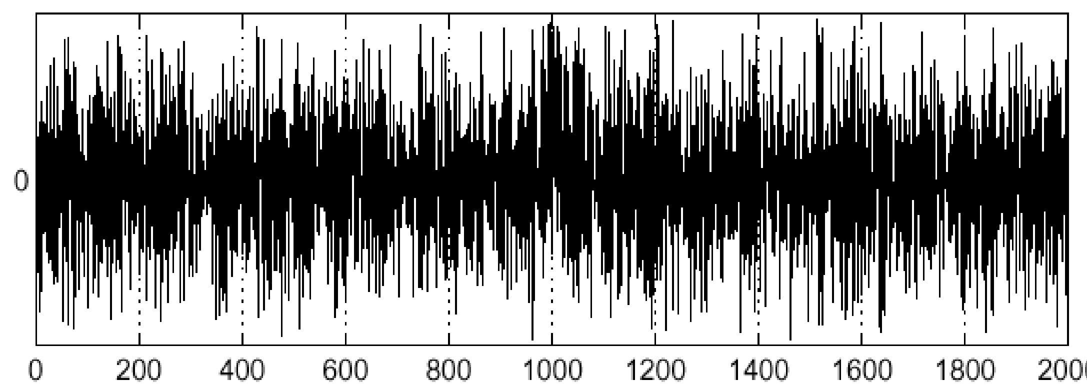
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$$f(x)$$



$$\frac{d}{dx}f(x)$$



Where is the edge?

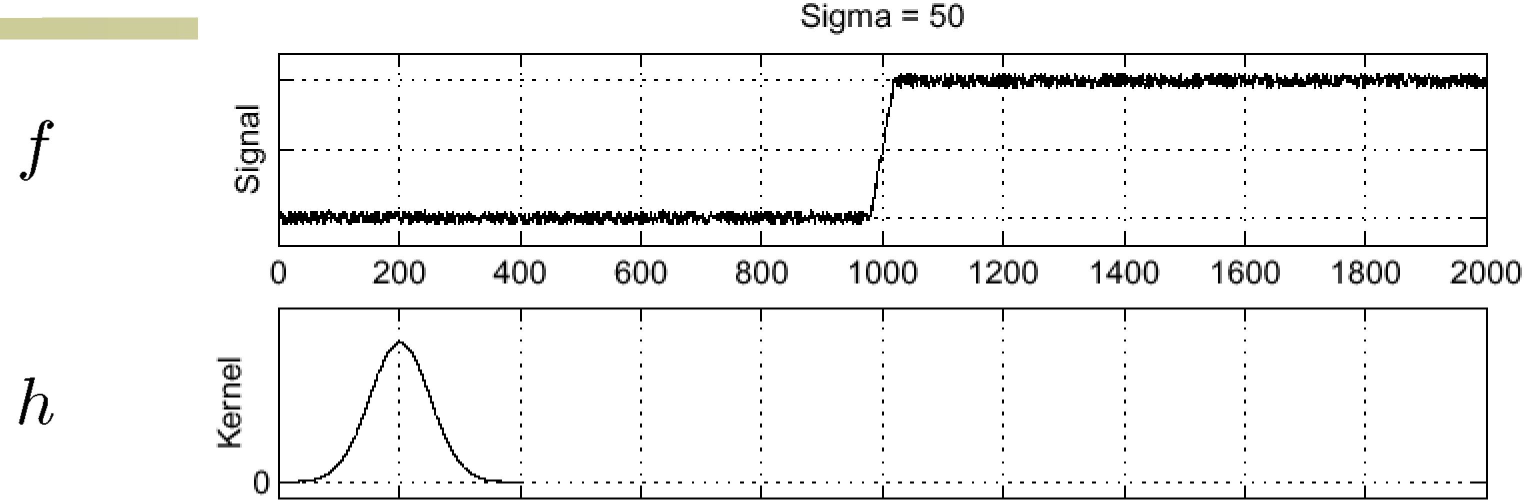
Slide credit Steve Seitz

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



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

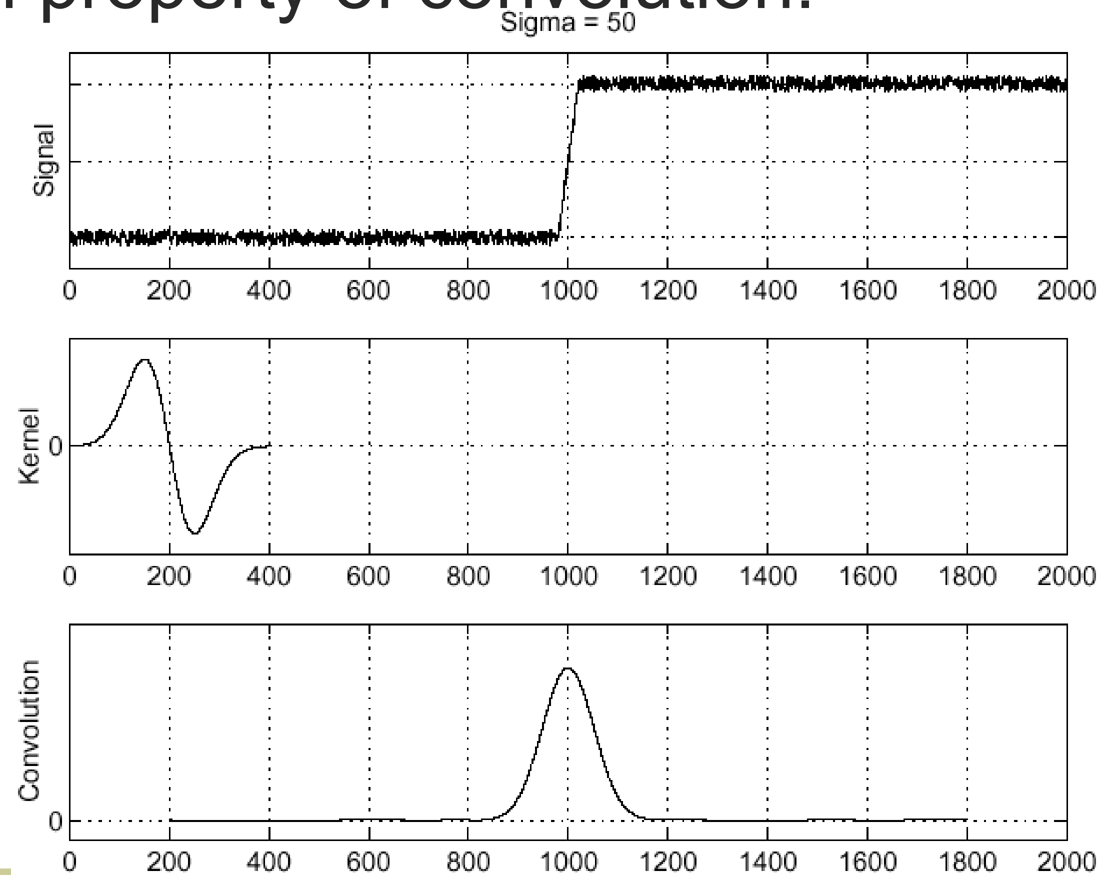
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

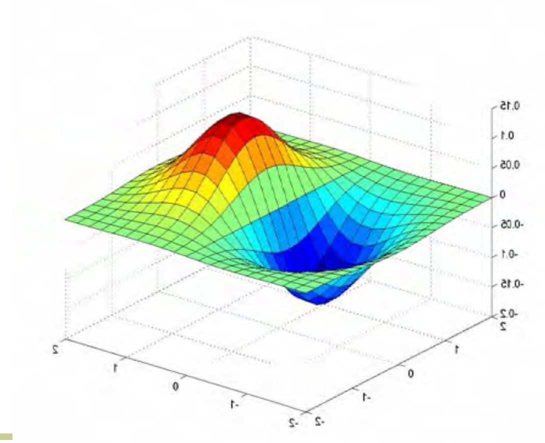


Slide credit Steve Seitz

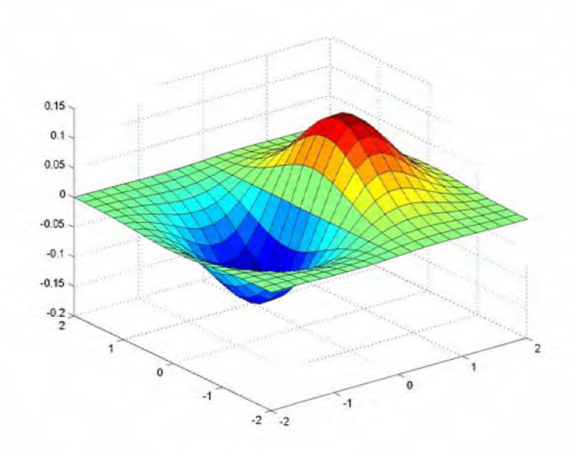
Derivative of Gaussian filters

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

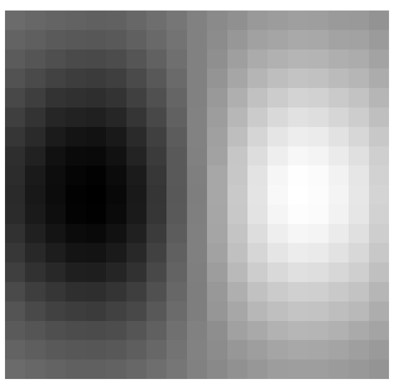
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$

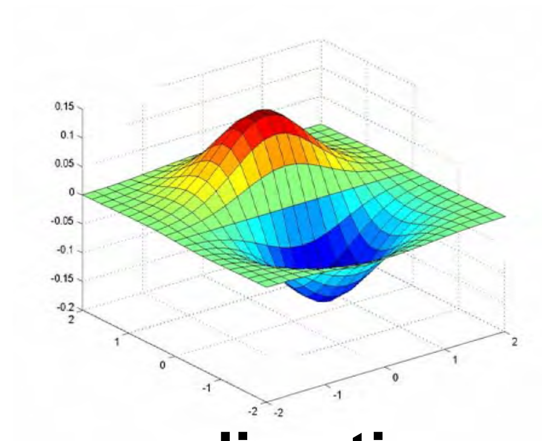


Derivative of Gaussian filters

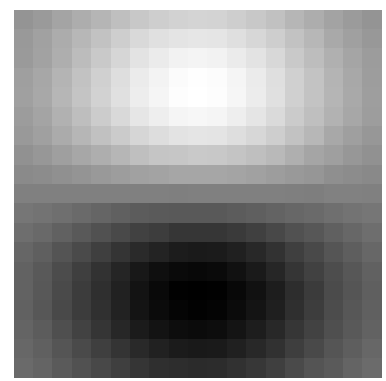


***x*-direction**





***y*-direction**



Source: L. Lazebnik

The Sobel Operator: A common approximation of derivative of gaussian

- Common approximation of derivative of Gaussian

$$s_x = \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad s_y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

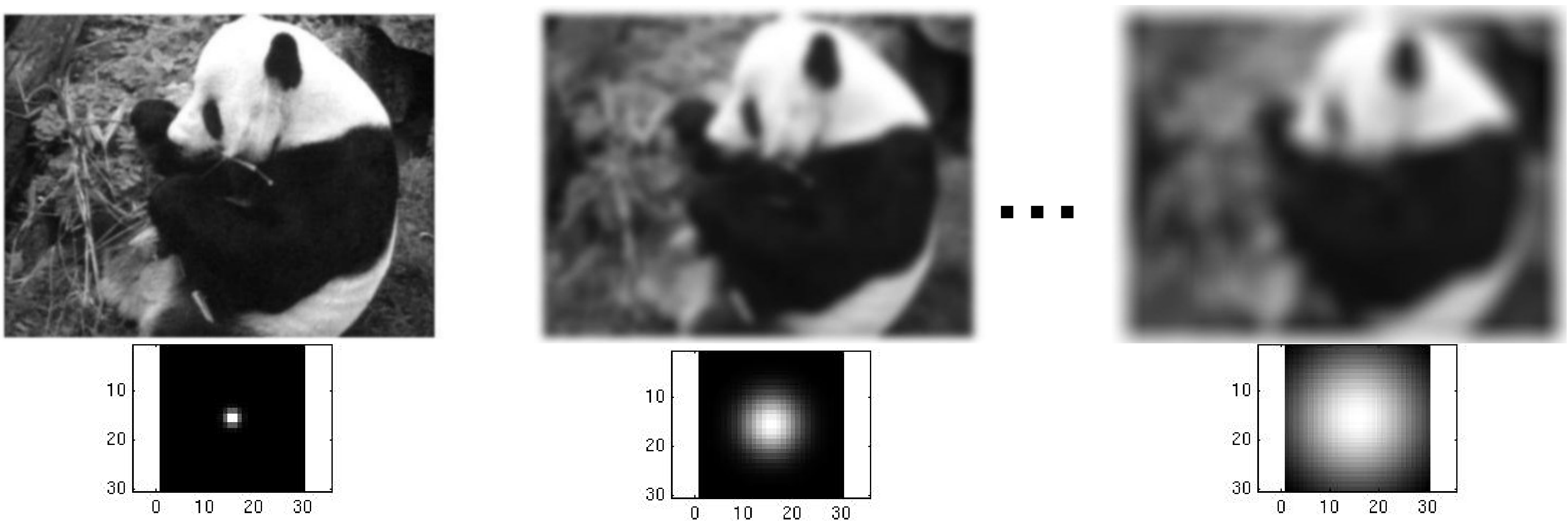
- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value

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
  - ___________ signs used to get high response in regions of high contrast
  - Sum to ___ → no response in constant regions
  - High absolute value at points of high contrast

Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives

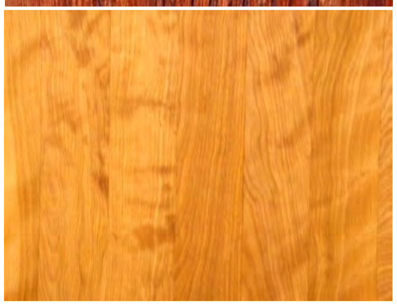
σ = 1 pixel σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.
Larger values: larger scale edges detected
Smaller values: finer features detected

Slide credit: Kristen Grauman

So, what scale to choose?

It depends what we're looking for.



Slide credit: Kristen Grauman

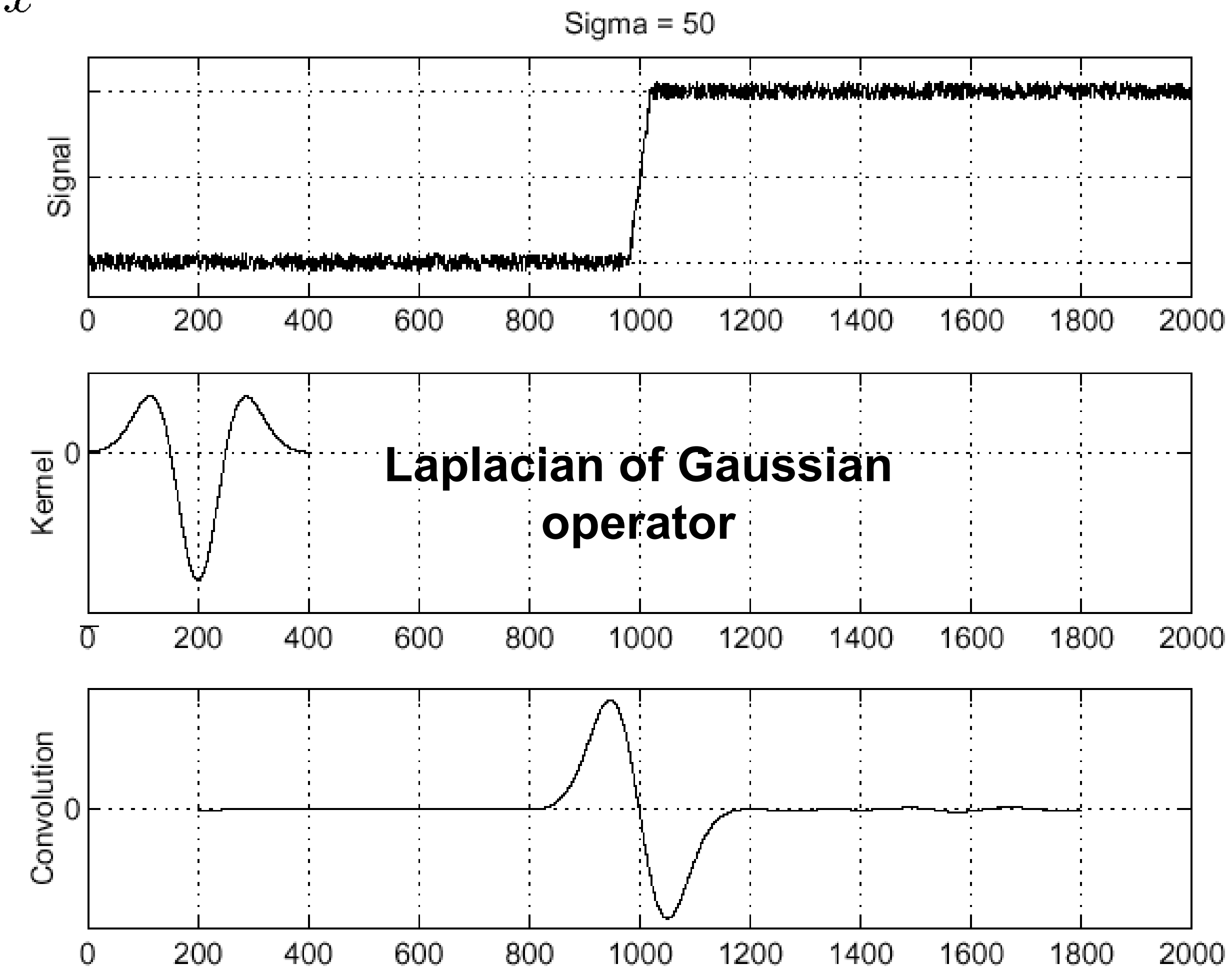
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$

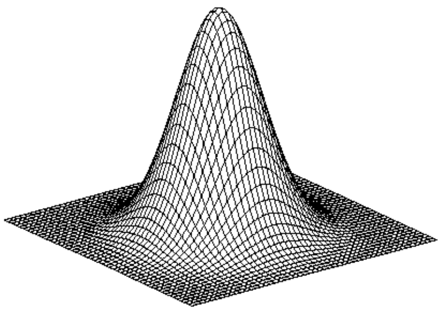


Where is the edge?

Zero-crossings of bottom graph

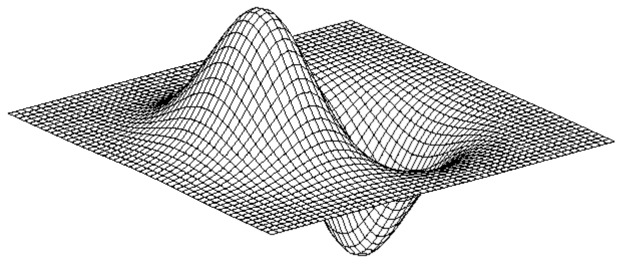
Slide credit: Steve Se

2D edge detection filters



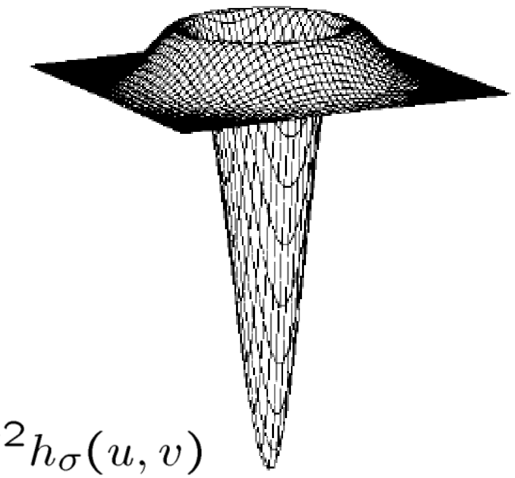
Gaussian

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$



$\nabla^2 h_\sigma(u,v)$

Laplacian of Gaussian

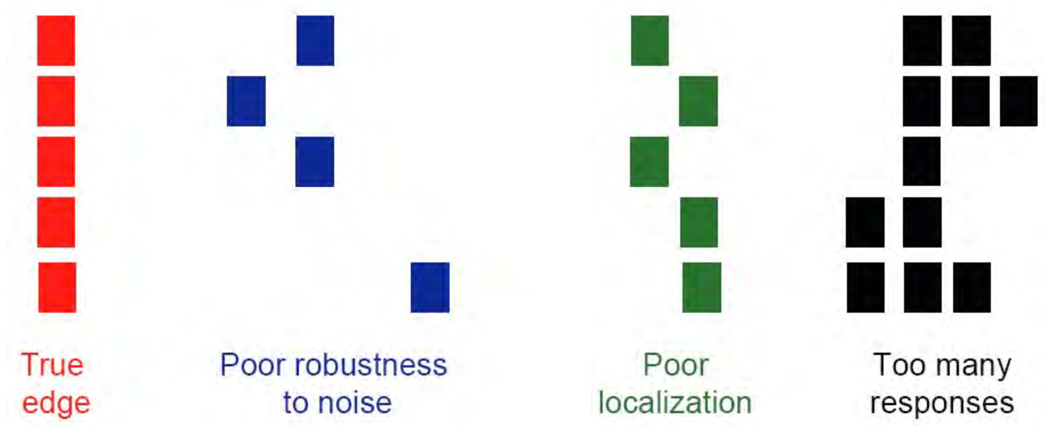
- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Slide credit: Steve Seitz

Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point;

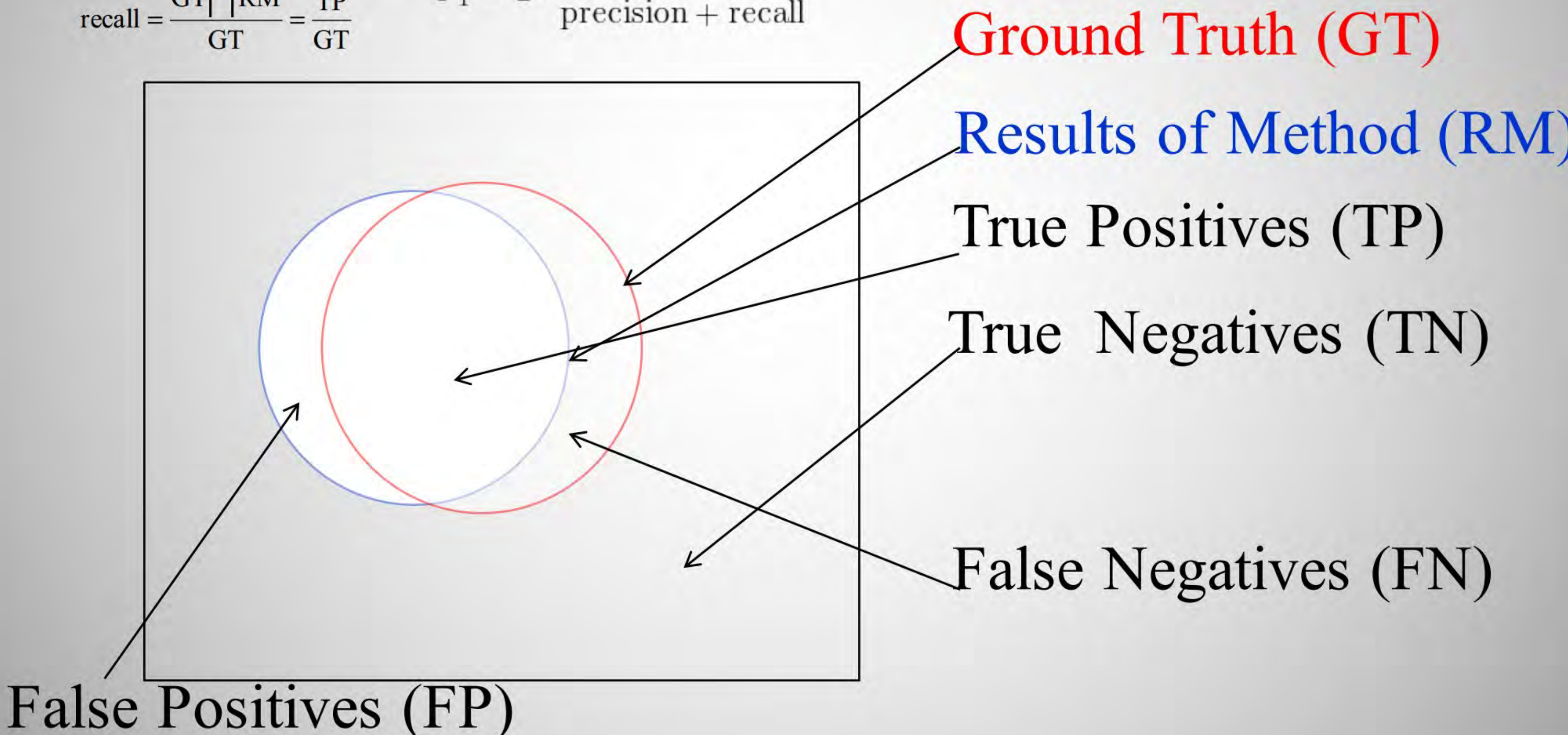


Evaluate Edge Detection

$$\text{precision} = \frac{GT \bigcap RM}{RM} = \frac{TP}{RM}$$

$$\text{recall} = \frac{GT \bigcap RM}{GT} = \frac{TP}{GT}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$



Basic Comparisons of Edge Operators

Gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

Gradients -> edges

Primary edge detection steps:

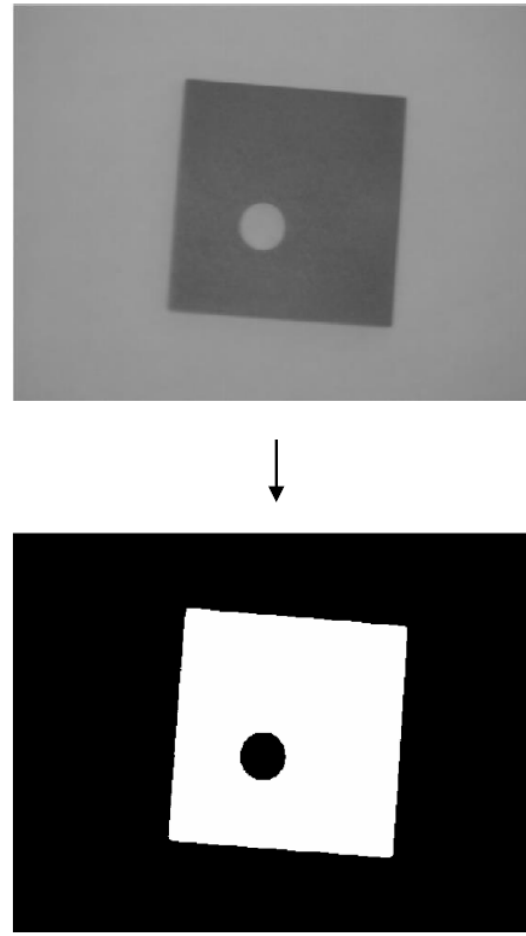
1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

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

    Thresholding and thinning

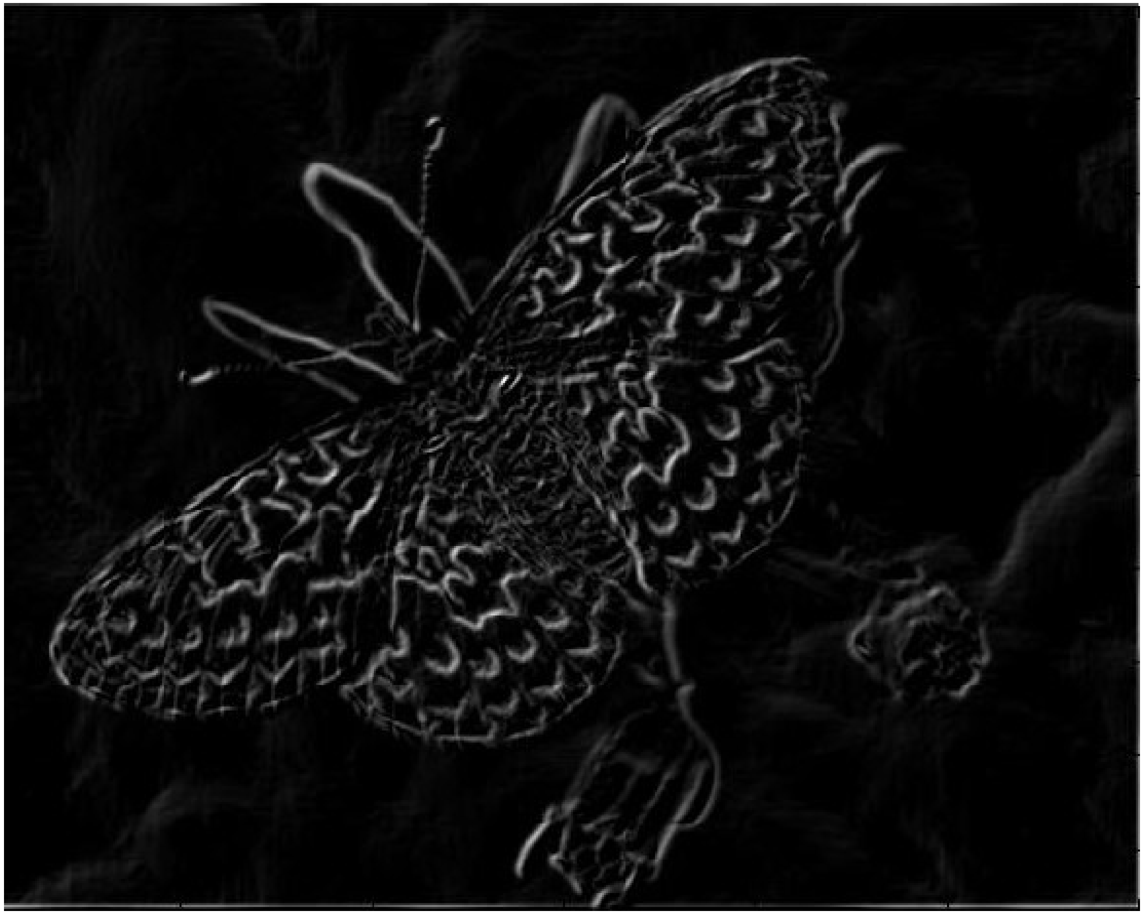
Thresholding

- 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



Lower threshold

Higher threshold

Example: Laplacian of Gaussian (LoG) and Canny Edge Detector

Marr and Hildreth Filtering, 1980.

- Smooth Image with Gaussian Filter
- Applying the Laplacian for a Gaussian-filtered image can be done in one step of convolution.
- Find zero-crossings
- Find slope of zero-crossings
- Apply threshold to slope and mark edges

J. Canny. 1986

- Smooth Image with Gaussian filter
- Compute Derivative of filtered image
- Find Magnitude and Orientation of gradient
- Apply Non-max suppression
- Apply Thresholding (Hysteresis)

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

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

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 Se

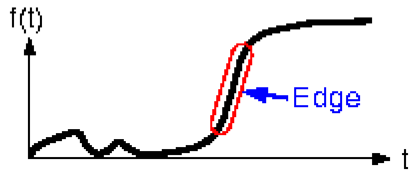
The Canny edge detector

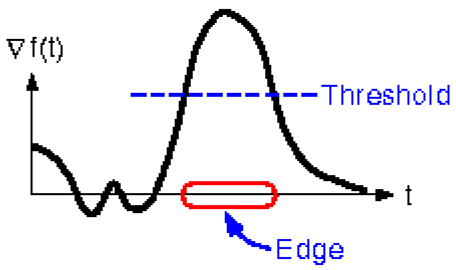
norm of the gradient

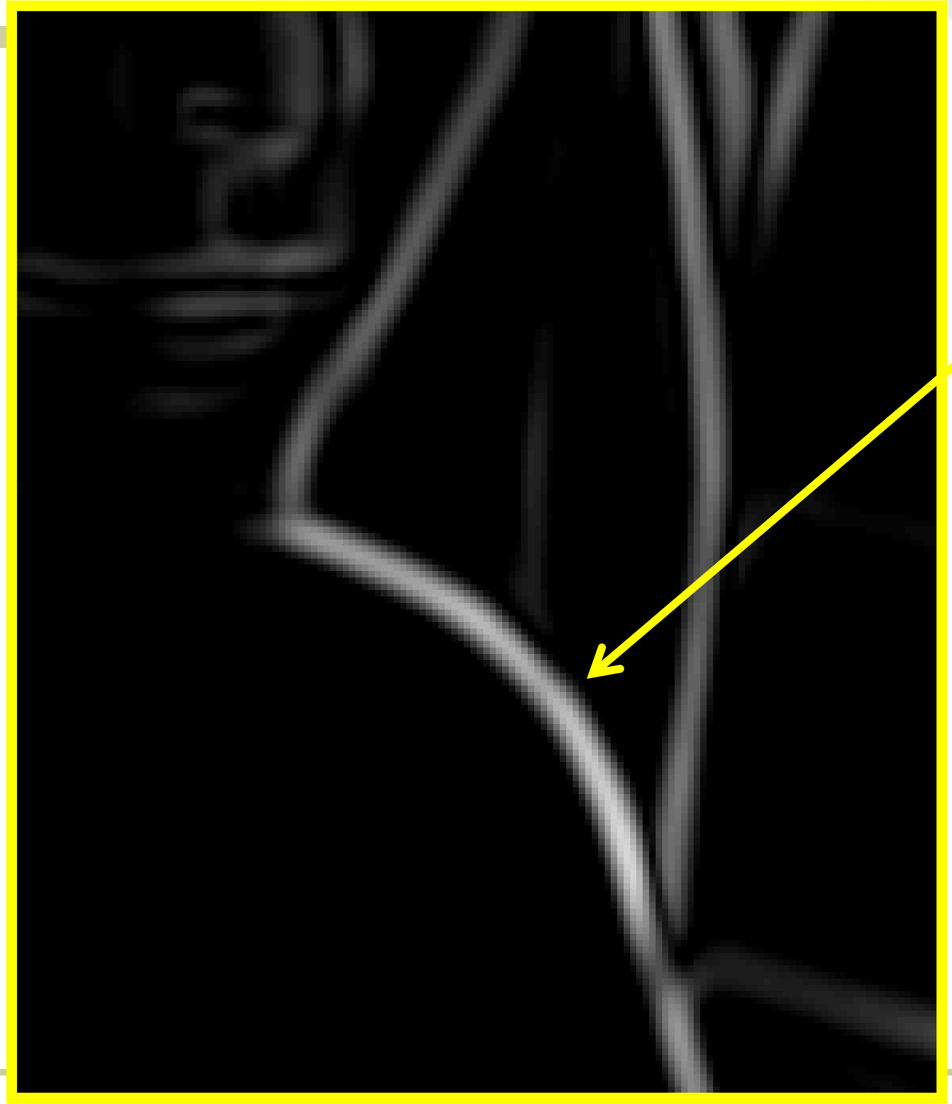
The Canny edge detector

thresholding

The Canny edge detector

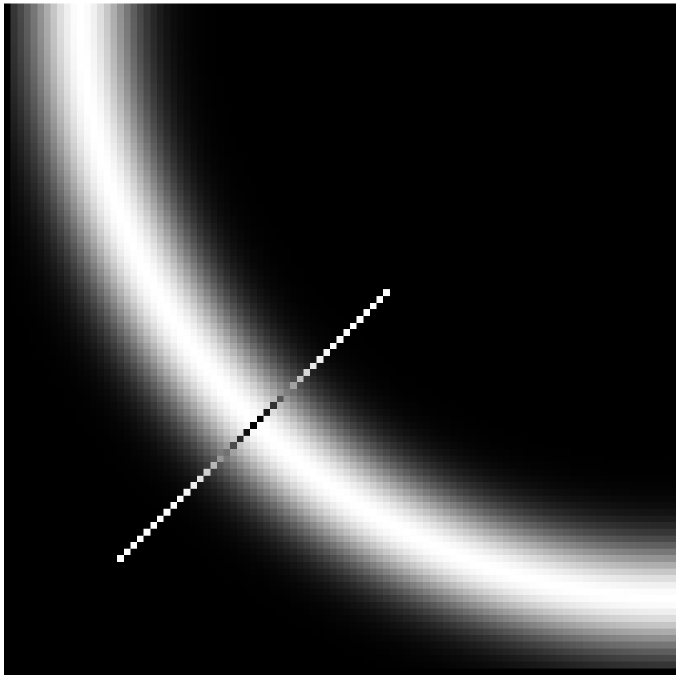


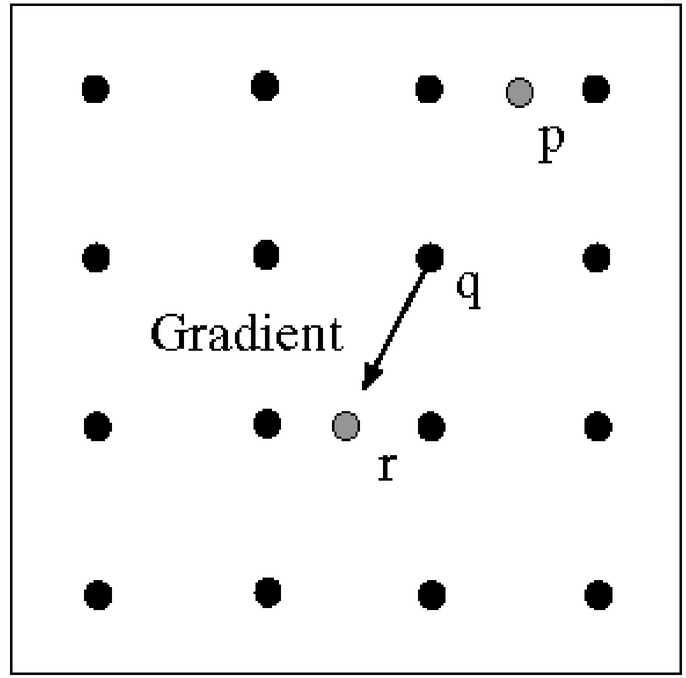




How to turn these thick regions of the gradient into curves?

Non-maximum suppression



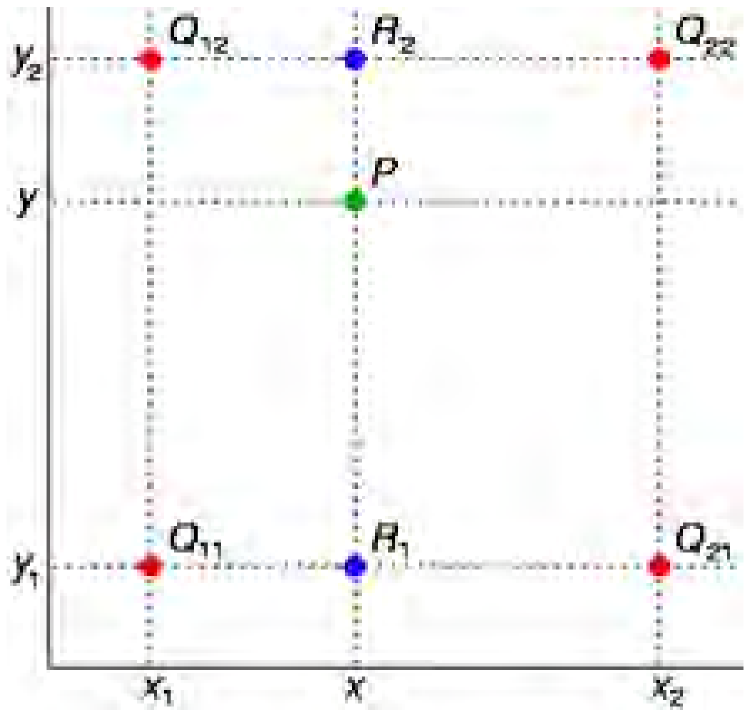


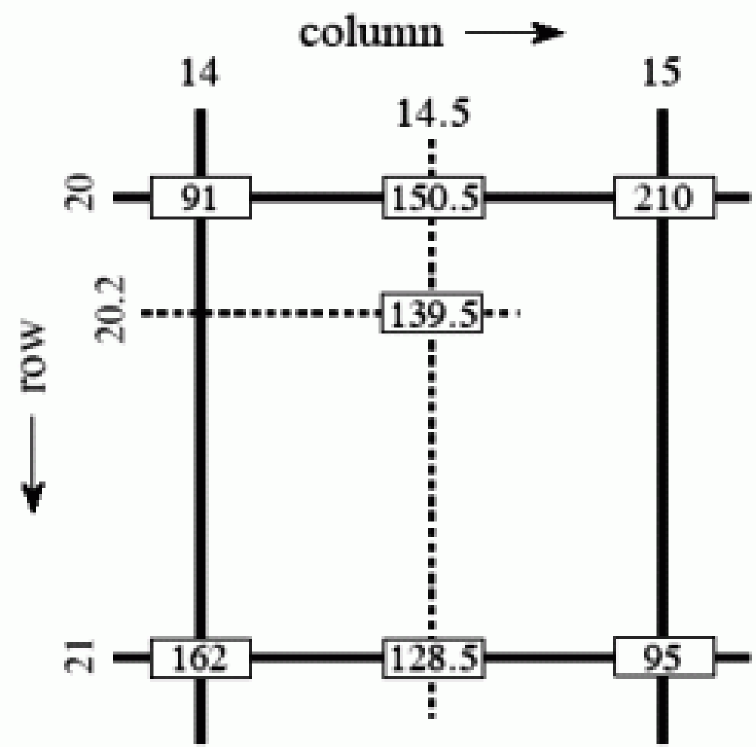
Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

Bilinear Interpolation

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

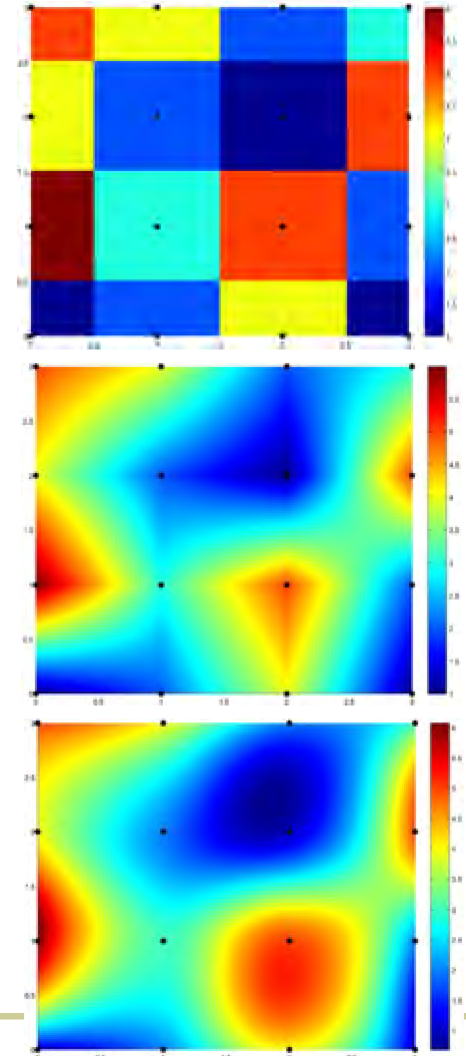




http://en.wikipedia.org/wiki/Bilinear_interpolation

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

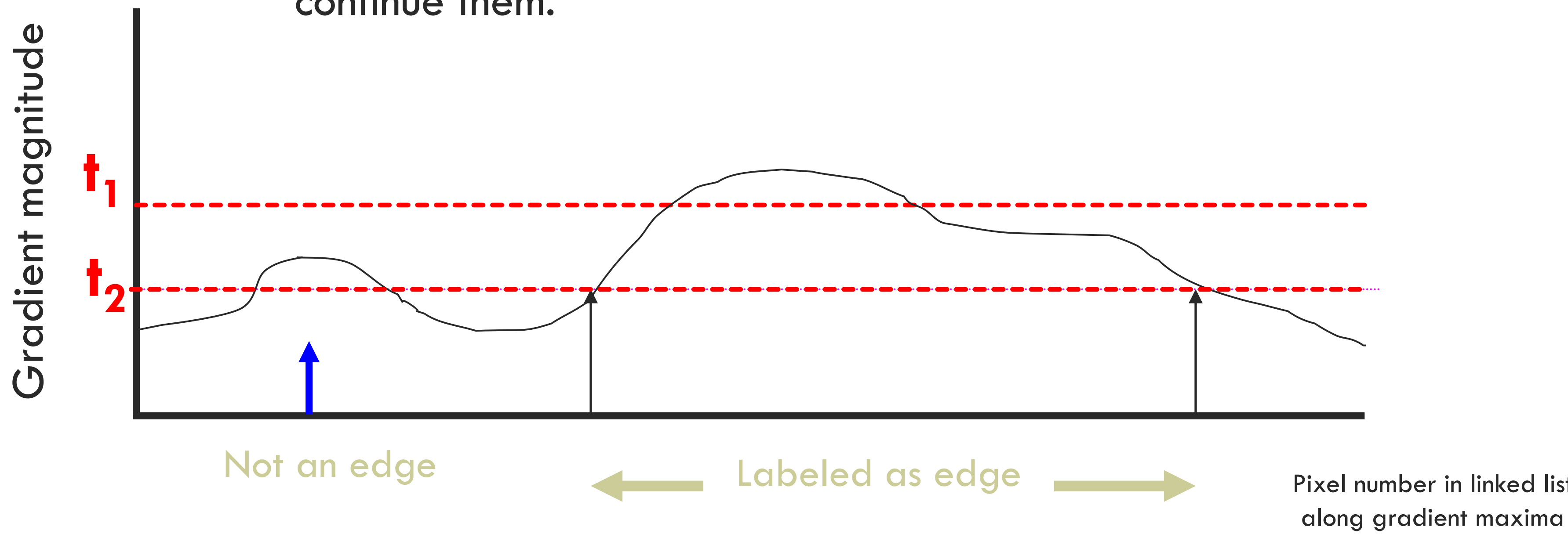
Problem: pixels along this edge didn't survive the thresholding

Hysteresis thresholding

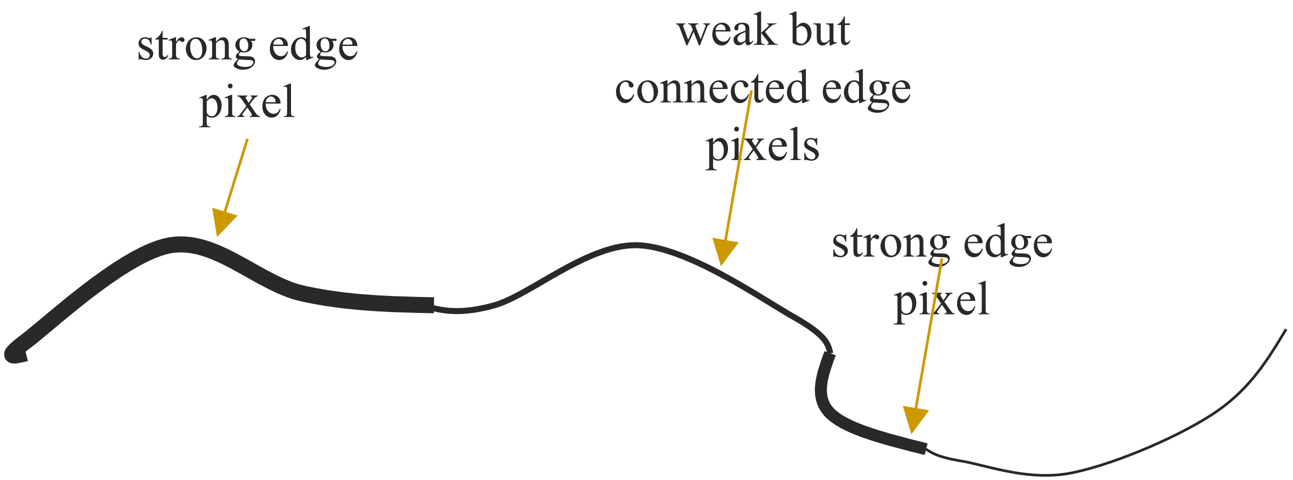
- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Closing edge gaps

- 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.



Hysteresis thresholding



Source: S. Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Final Canny Edges

Effect of σ (Gaussian kernel spread/size)

original

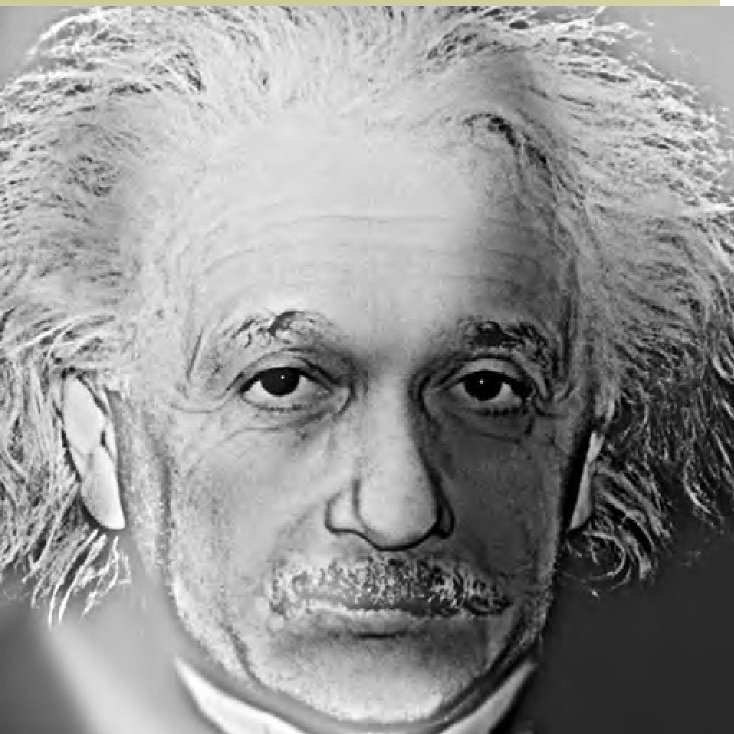
Canny with $\sigma = 1$

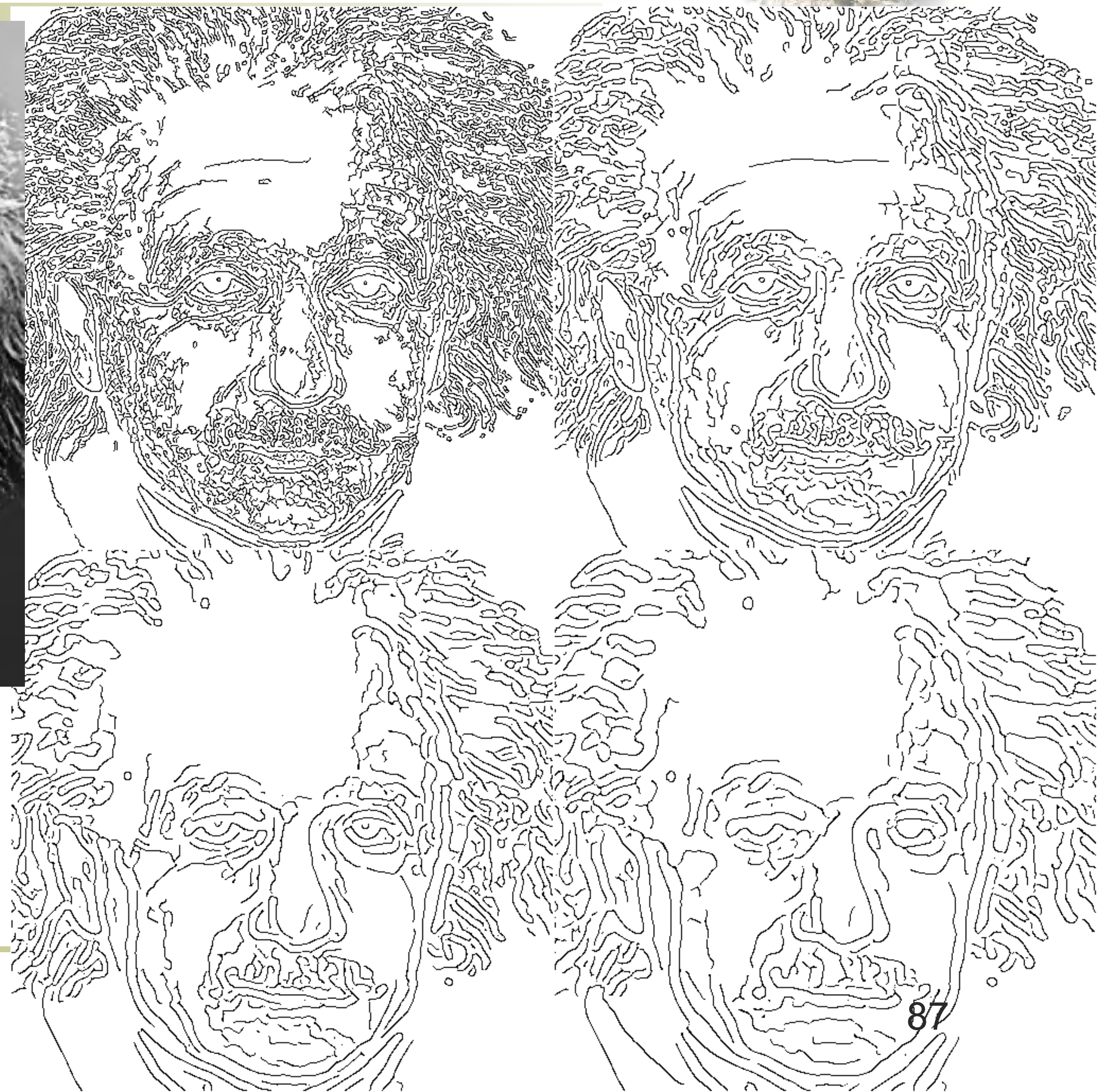
Canny with $\sigma = 2$

The choice of σ depends on desired behavior

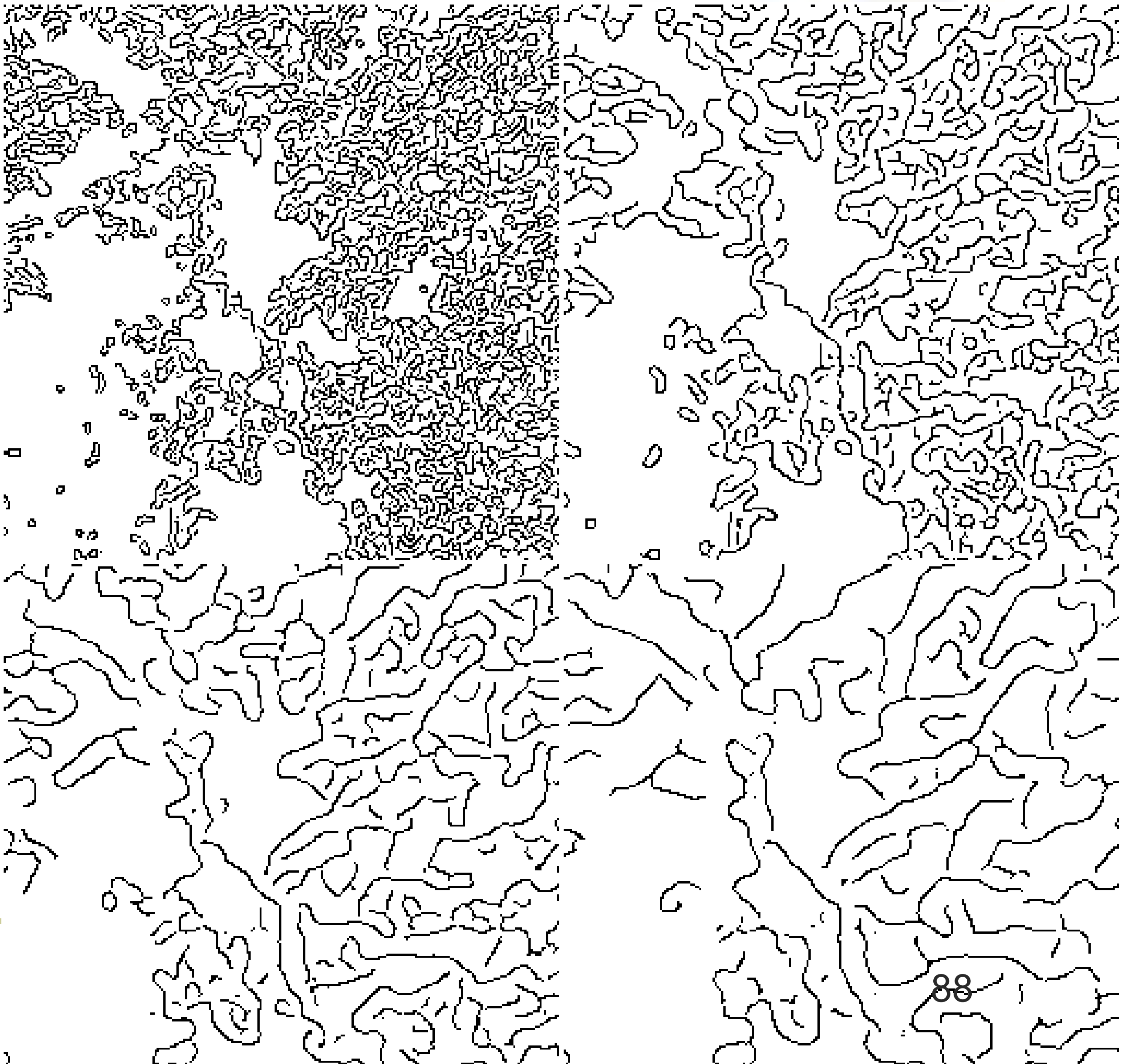
- large σ detects large scale edges
- small σ detects fine features

Example: Canny Edge Detection





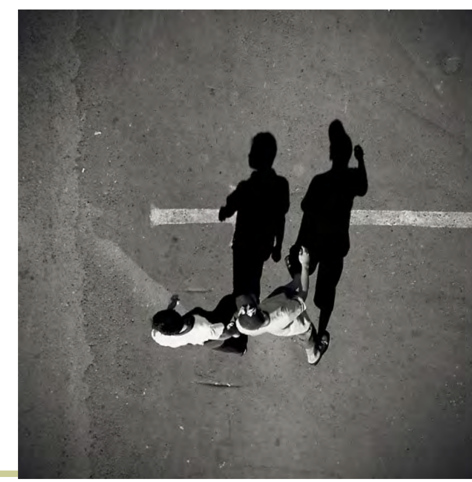
Example: Canny Edge Detection



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`**

Low-level edges vs. perceived contours



Background **Texture** **Shadows**





Low-level edges vs. perceived contours

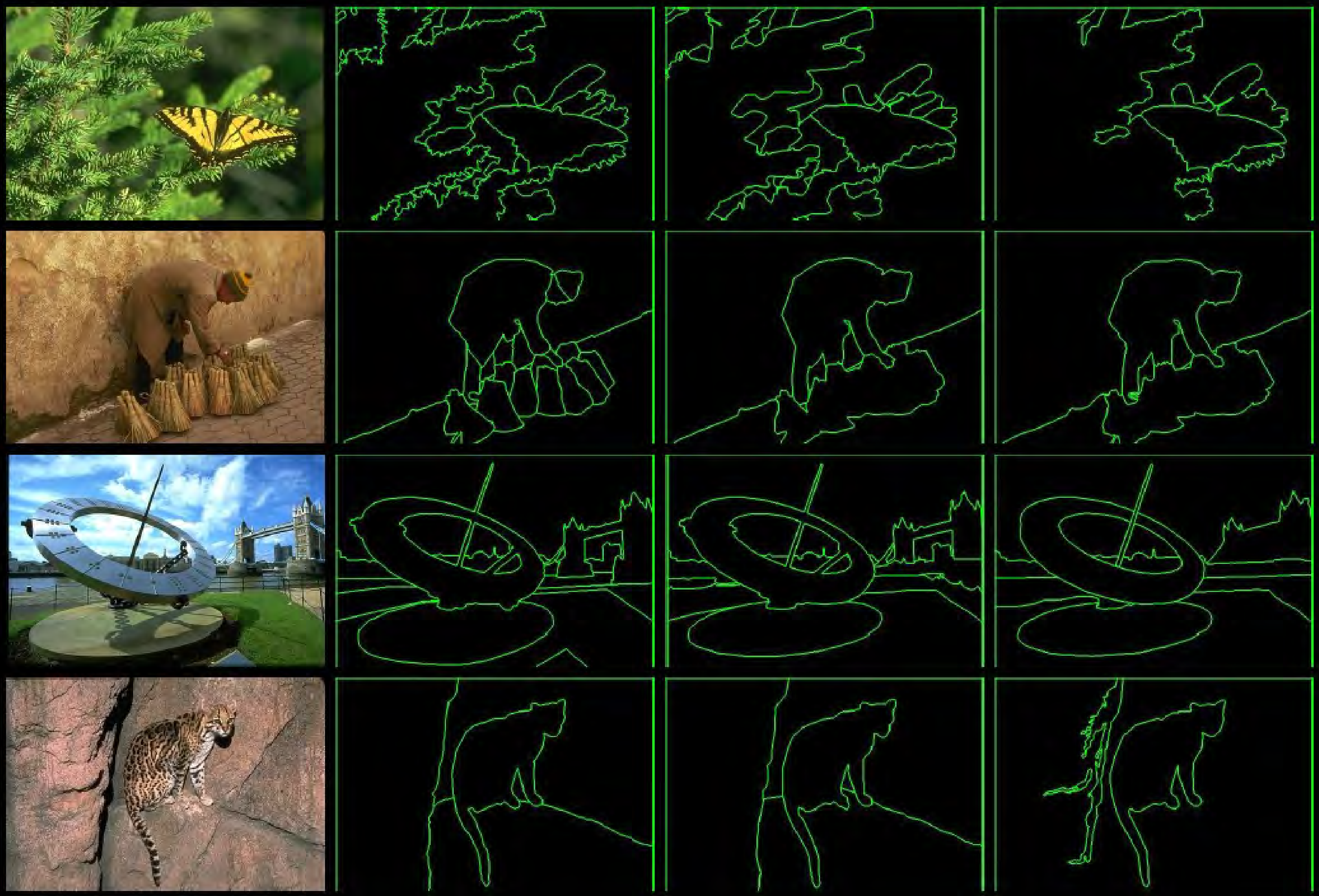
image **human segmentation** **gradient magnitude**

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

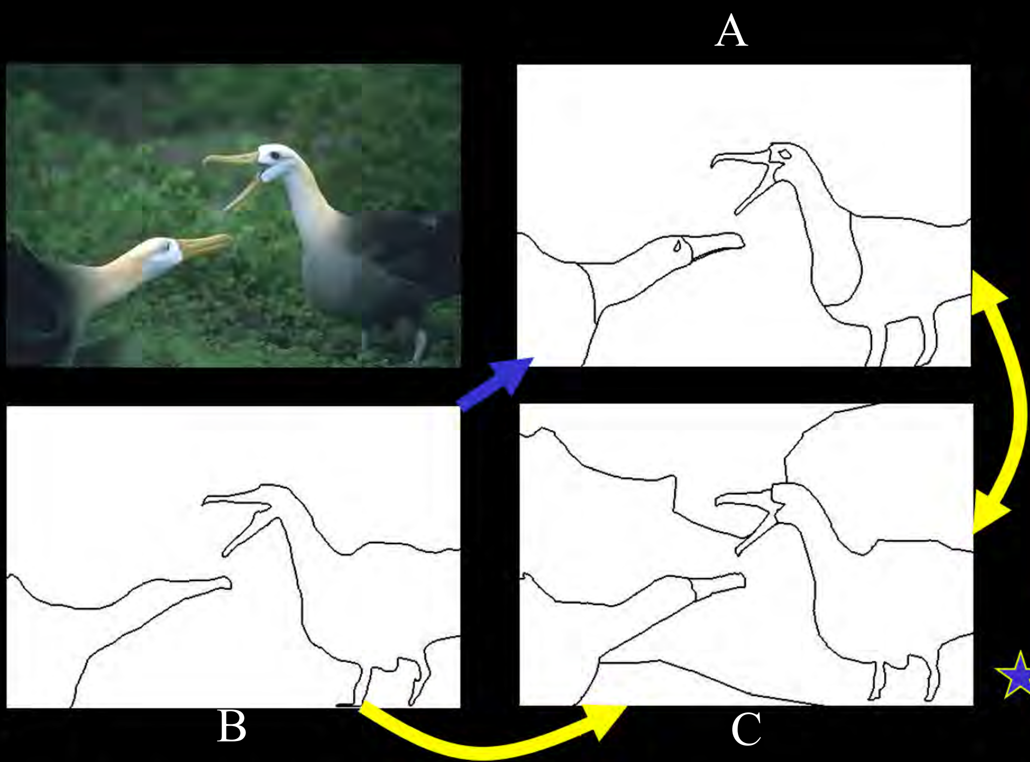
Protocol

You will be presented a photographic image. Divide the image into some number of segments, where the segments represent "things" or "parts of things" in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance.

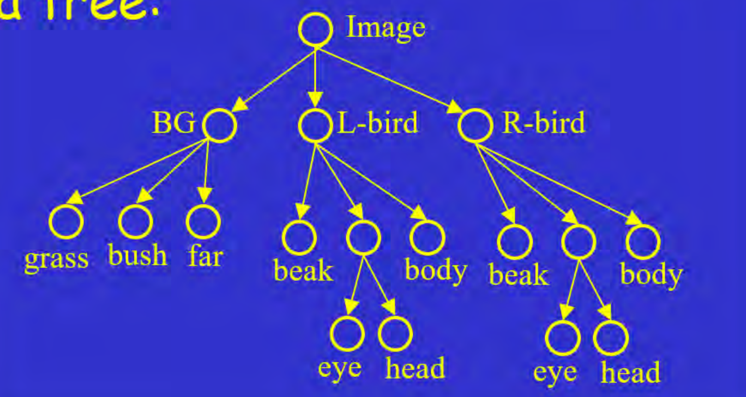
- Custom segmentation tool
- Subjects obtained from work-study program (UC Berkeley undergraduates)



Consistency



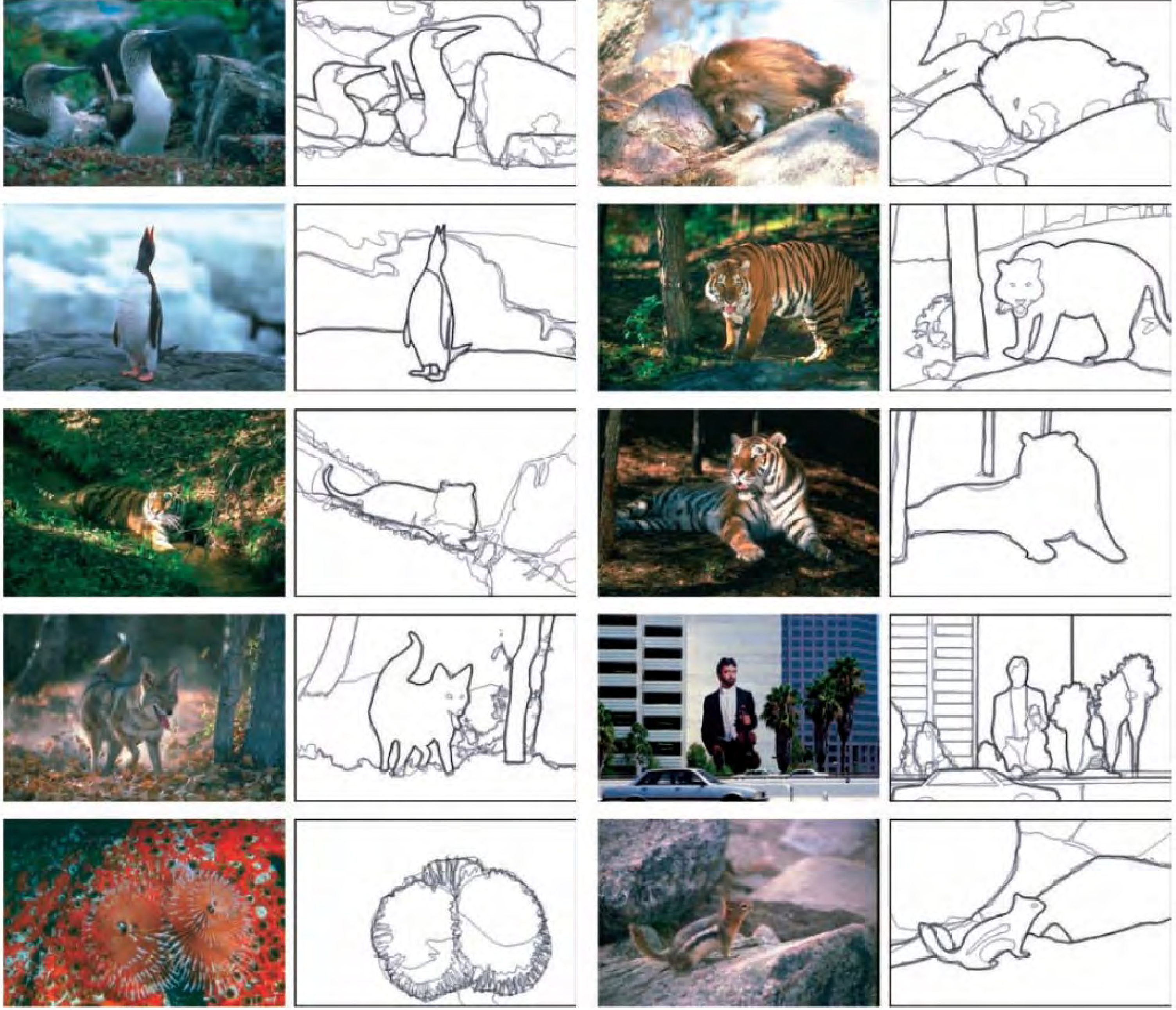
Perceptual organization forms a tree:



- A,C are refinements of B
- A,C are mutual refinements
- A,B,C represent the same percept
  - <u>Attention</u> accounts for differences

☆ Two segmentations are consistent when they can be explained by the same segmentation tree (i.e. they could be derived from a single perceptual organization).

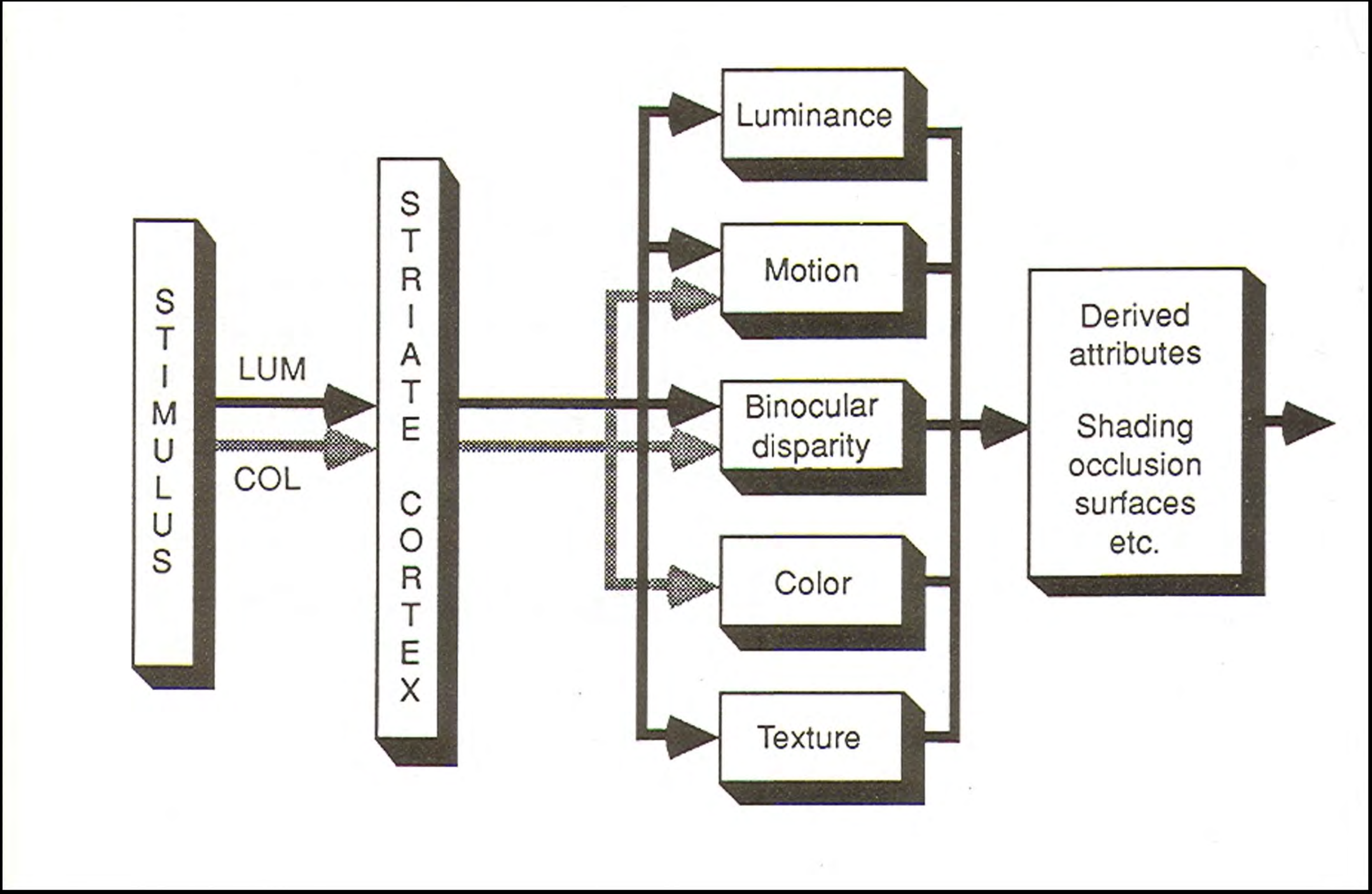
Learn from humans which combination of features is most indicative of a "good" contour?



[D. Martin et al. PAMI 2004]

Human-marked segment boundaries

Contours can be defined by any of a number of cues (P. Cavanagh)



Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues

David R. Martin, *Member, IEEE*, Charless C. Fowlkes, and Jitendra Malik, *Member, IEEE*

Abstract—The goal of this work is to accurately detect and localize boundaries in natural scenes using local image measurements. We formulate features that respond to characteristic changes in brightness, color, and texture associated with natural boundaries. In order to combine the information from these features in an optimal way, we train a classifier using human labeled images as ground truth. The output of this classifier provides the posterior probability of a boundary at each image location and orientation. We present precision-recall curves showing that the resulting detector significantly outperforms existing approaches. Our two main results are 1) that cue combination can be performed adequately with a simple linear model and 2) that a proper, explicit treatment of texture is required to detect boundaries in natural images.

Index Terms—Texture, supervised learning, cue combination, natural images, ground truth segmentation data set, boundary detection, boundary localization.

Dataflow

Image

Boundary Cues

Brightness

Color

Texture

Cue Combination

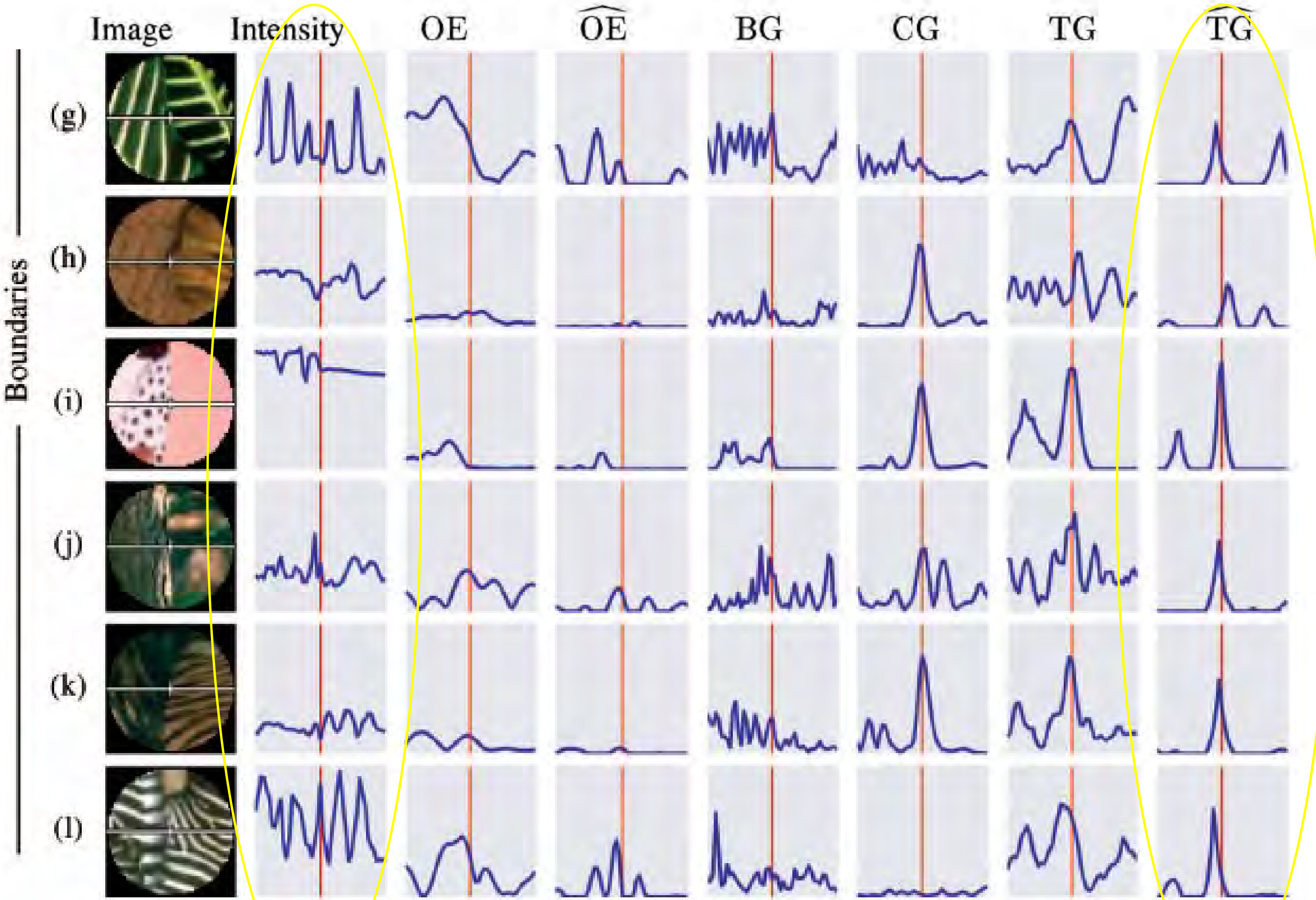
Model

P_b

Challenges: texture cue, cue combination

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

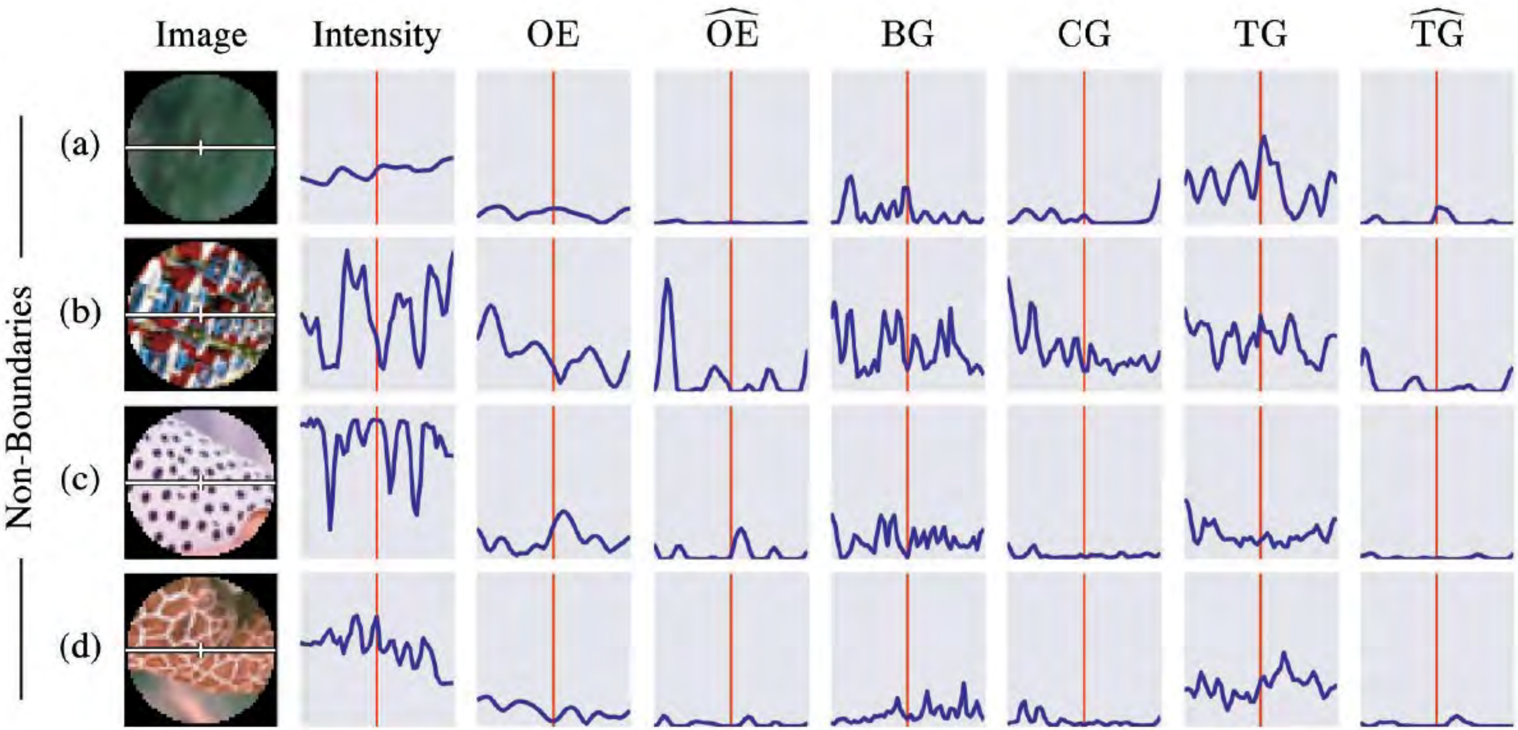
What features are responsible for perceived edges?



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

[D. Martin et al. PAMI 2004]

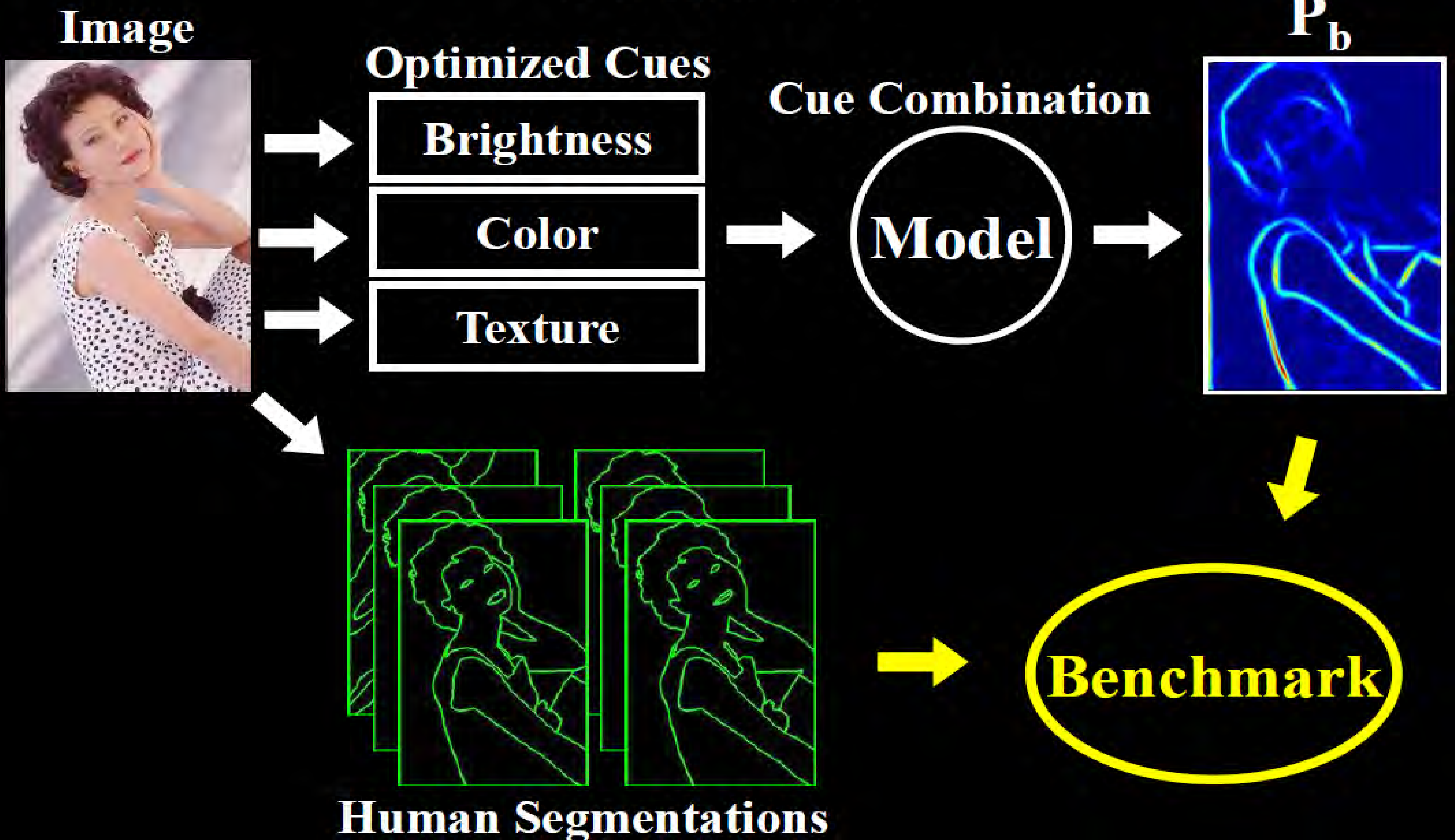
What features are responsible for perceived edges?



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

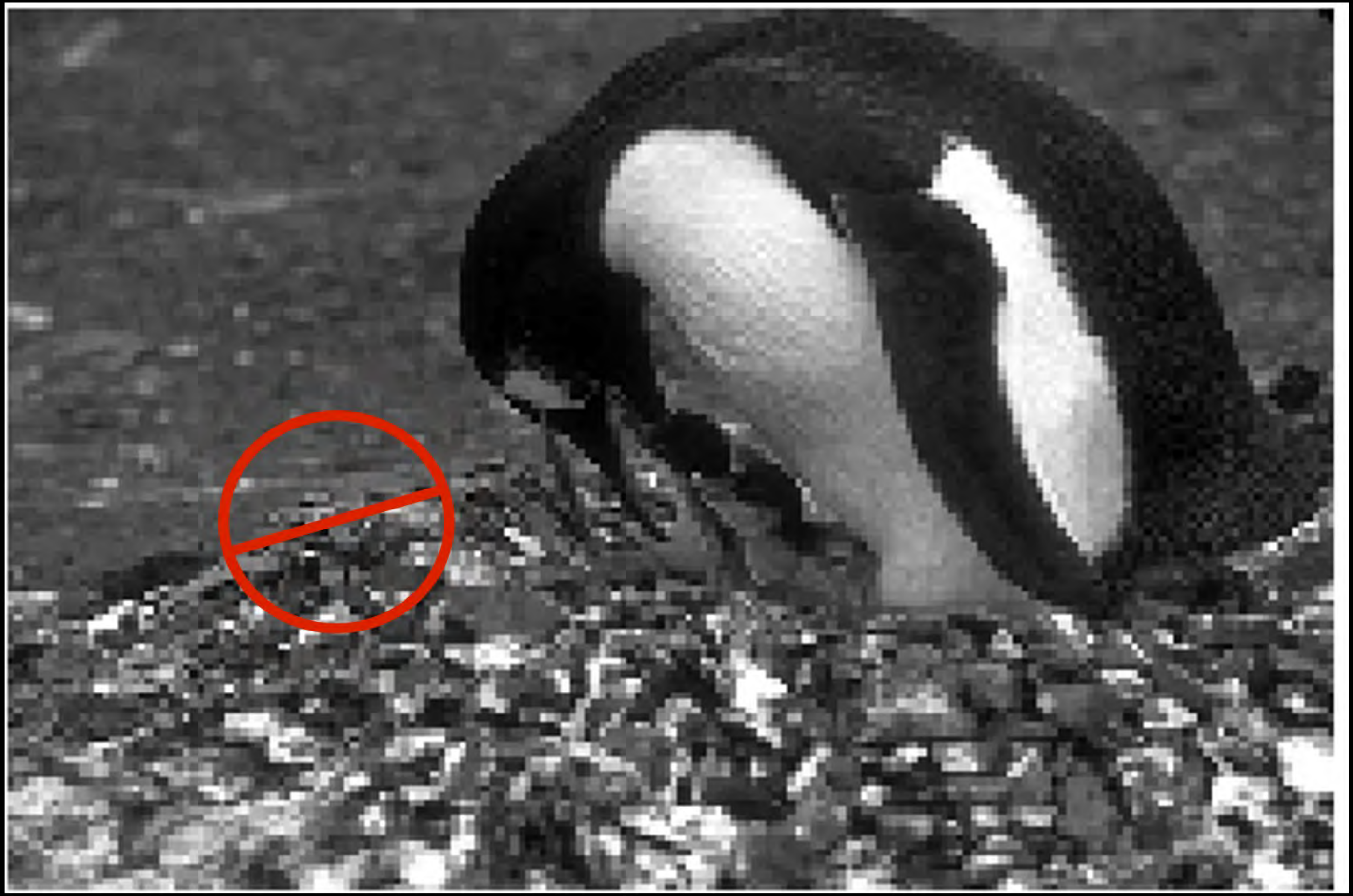
[D. Martin et al. PAMI 2004]

Dataflow



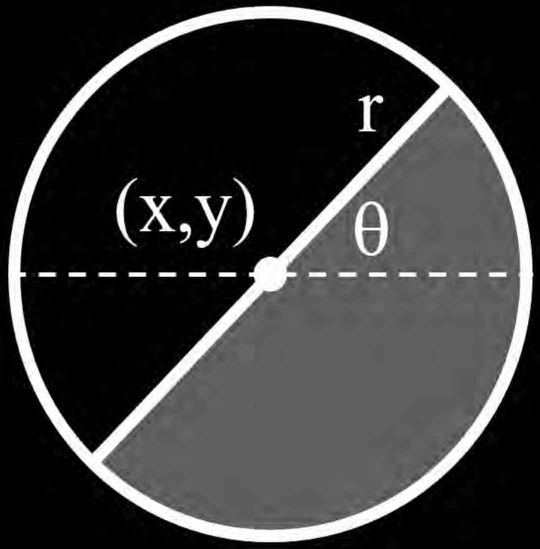
Credit: David Martin

Oriented Feature Gradient

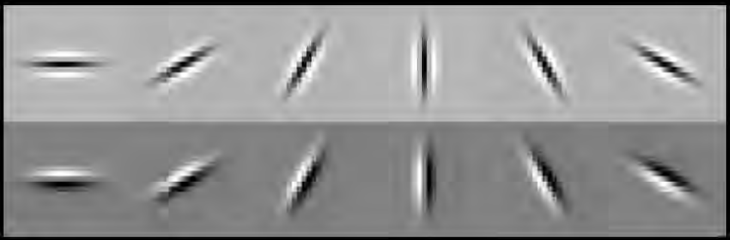


Individual Features

- 1976 CIE L*a*b* colorspace
- Brightness Gradient BG(x,y,r,θ)
  - Difference of L* distributions
- Color Gradient CG(x,y,r,θ)
  - Difference of a*b* distributions

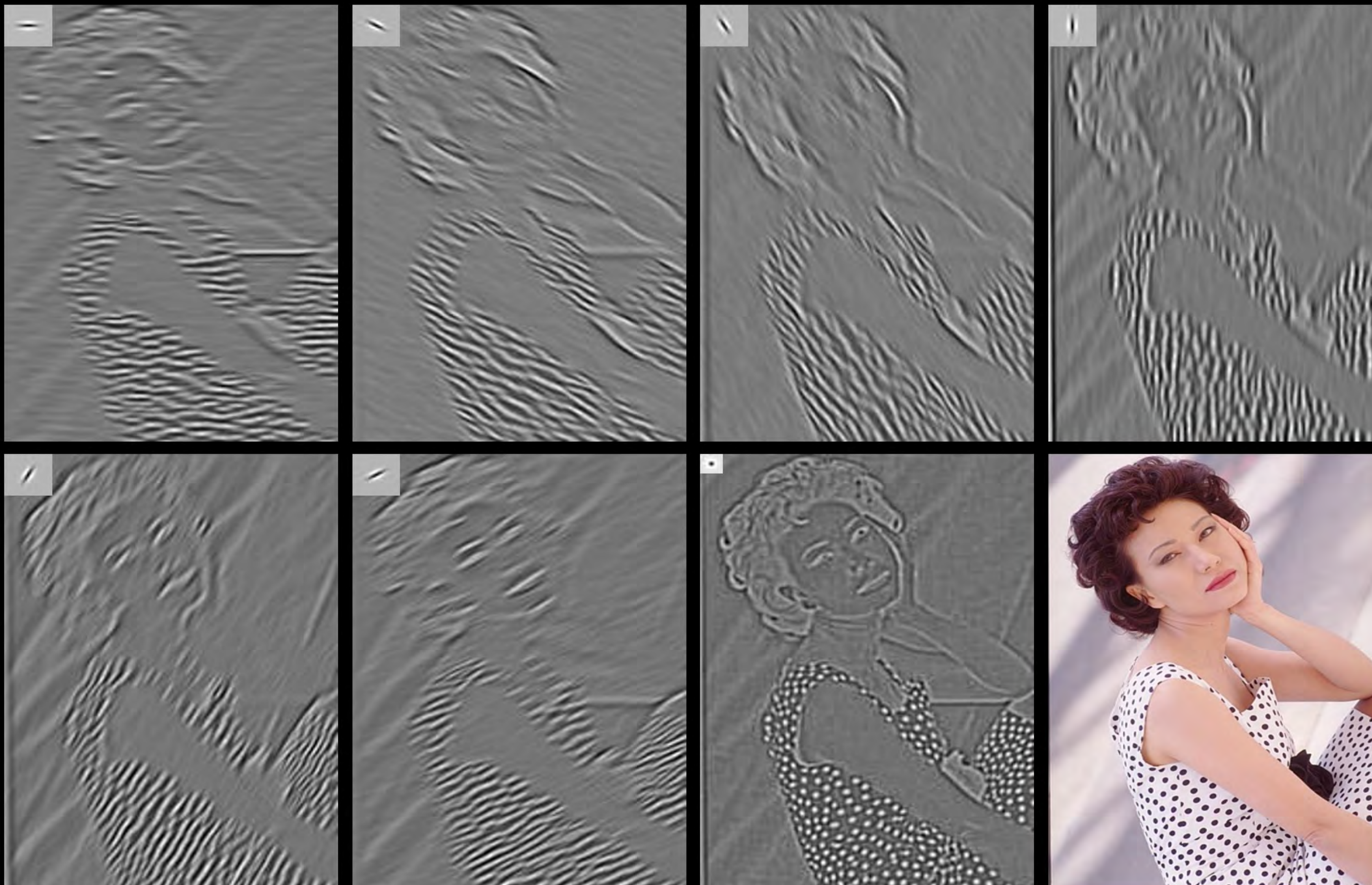


- Texture Gradient TG(x,y,r,θ)
  - Difference of distributions of V1-like filter responses



These are combined using logistic regression

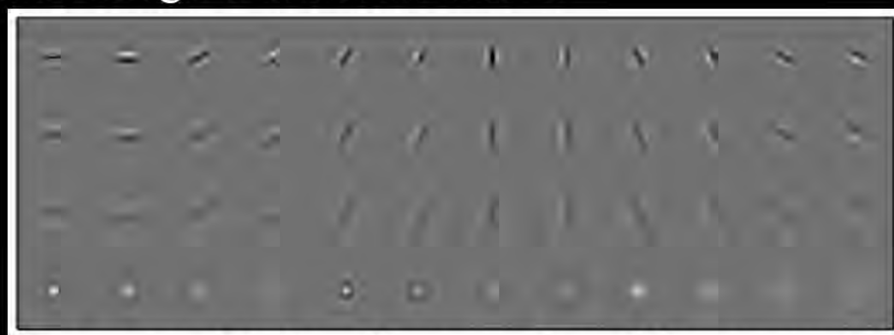
Filter Outputs



2D Textons

- Goal: find canonical local features in a texture;

  1) Filter image with linear filters:

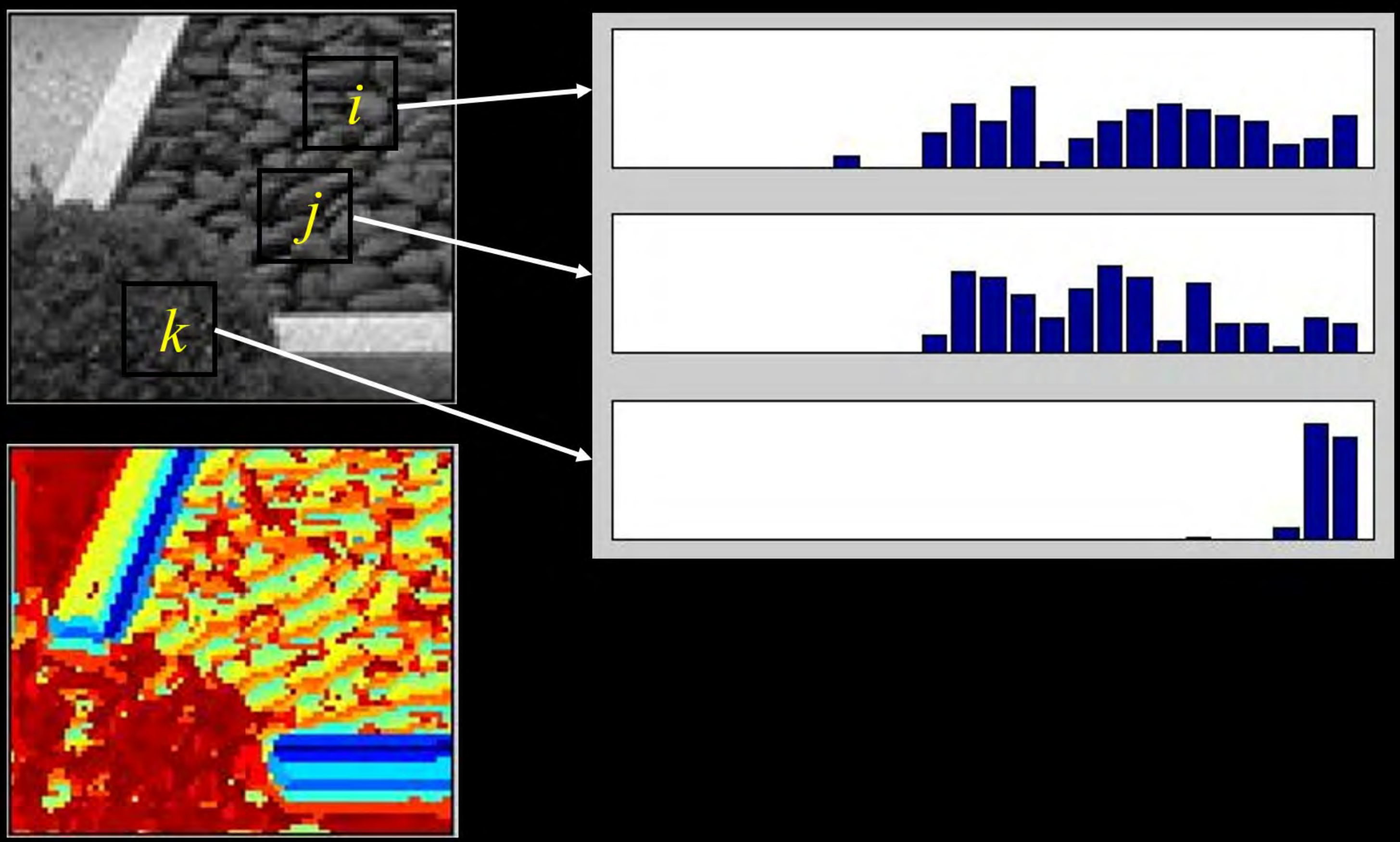


  2) Vector quantization (k-means) on filter outputs;

  3) Quantization centers are the textons.

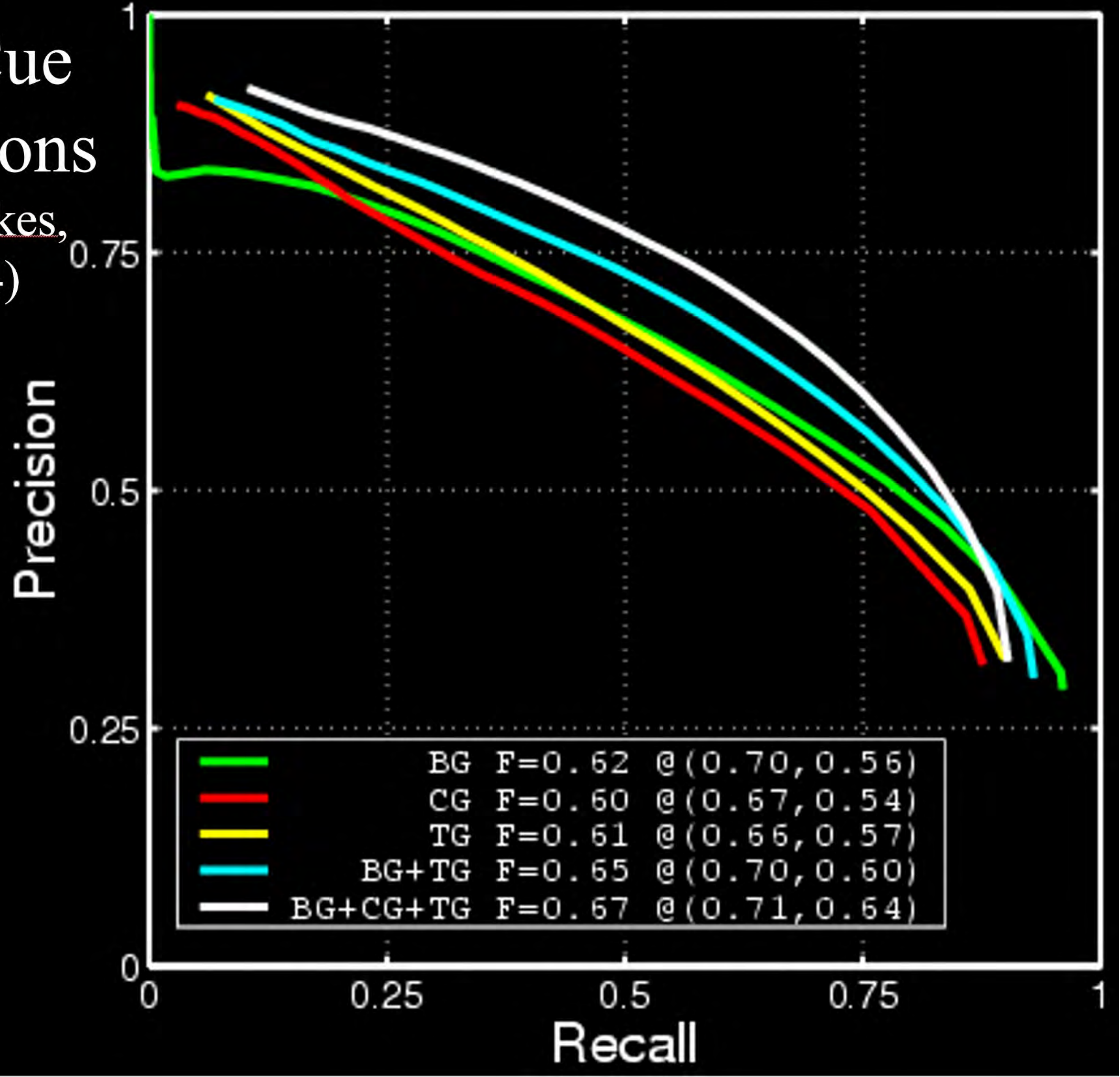
- Spatial distribution of textons defines the texture;

Texture gradient = Chi square distance between texton histograms in half disks across edge



Various Cue Combinations

(Martin, Fowlkes, Malik, 2004)



[D. Martin et al. PAMI 2004]

Kristen Grauman, UT-

Brightness

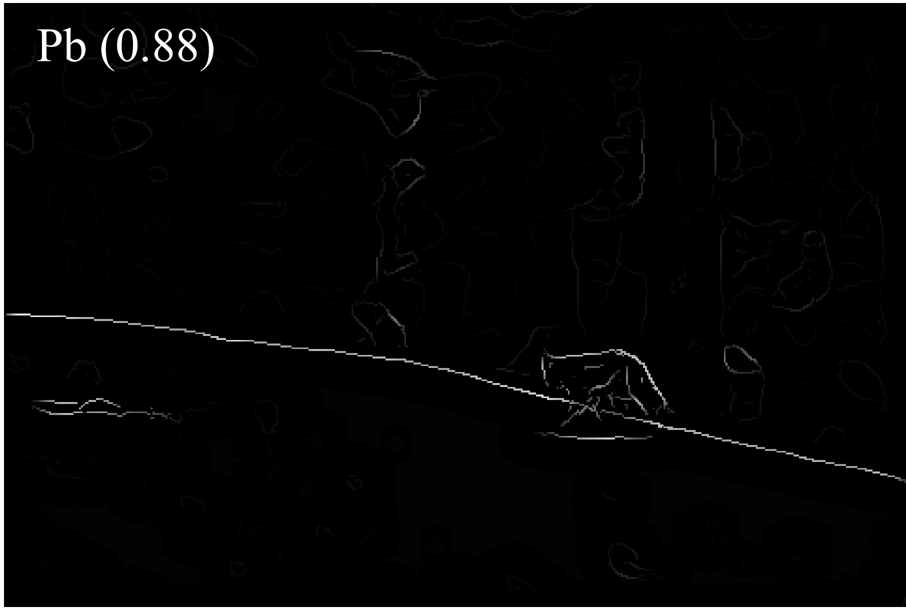
Color

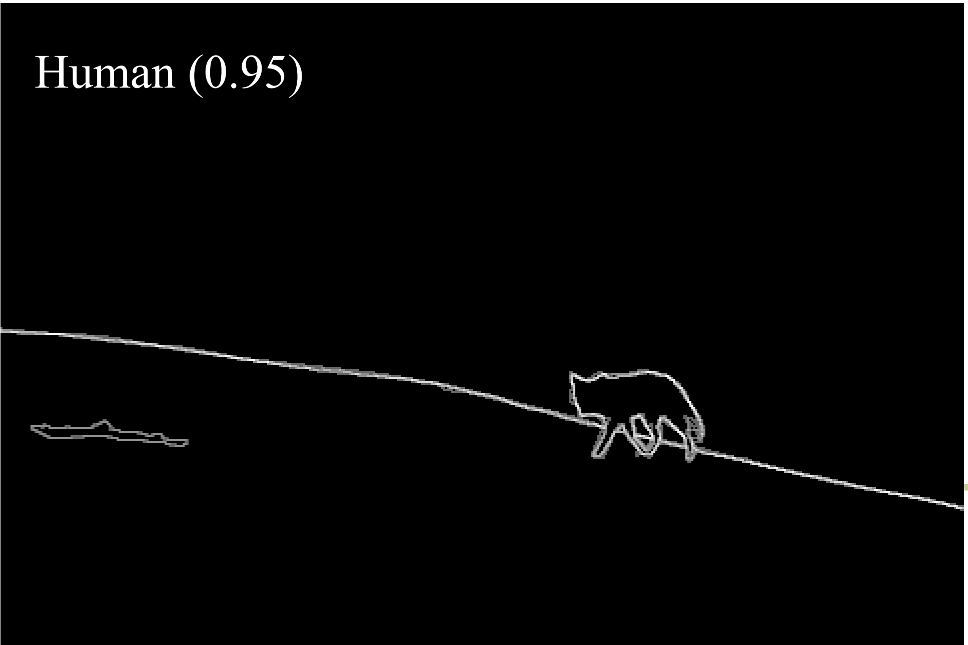
Texture

Combined

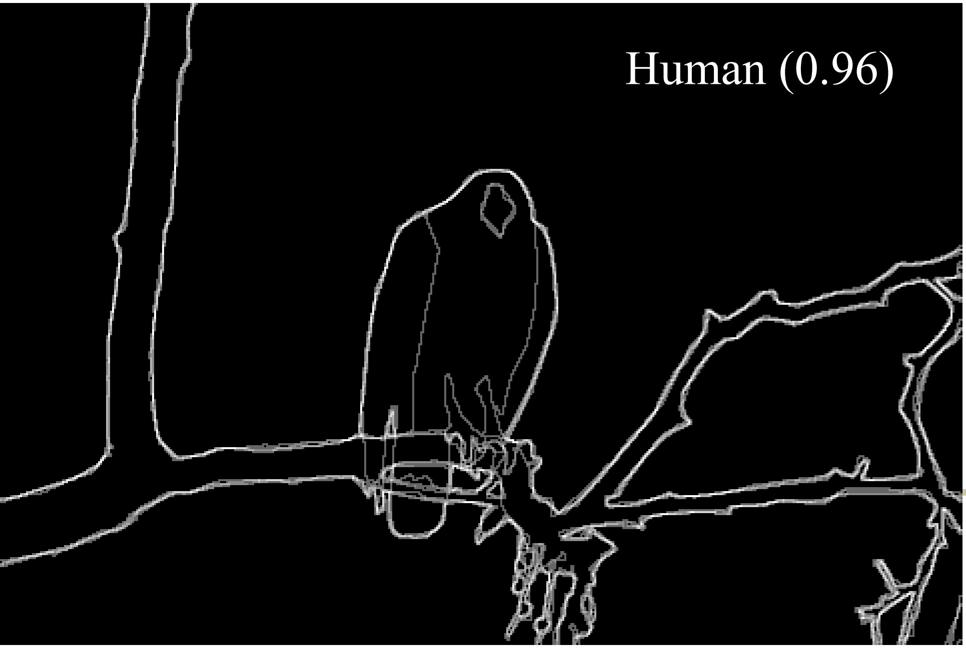
Human

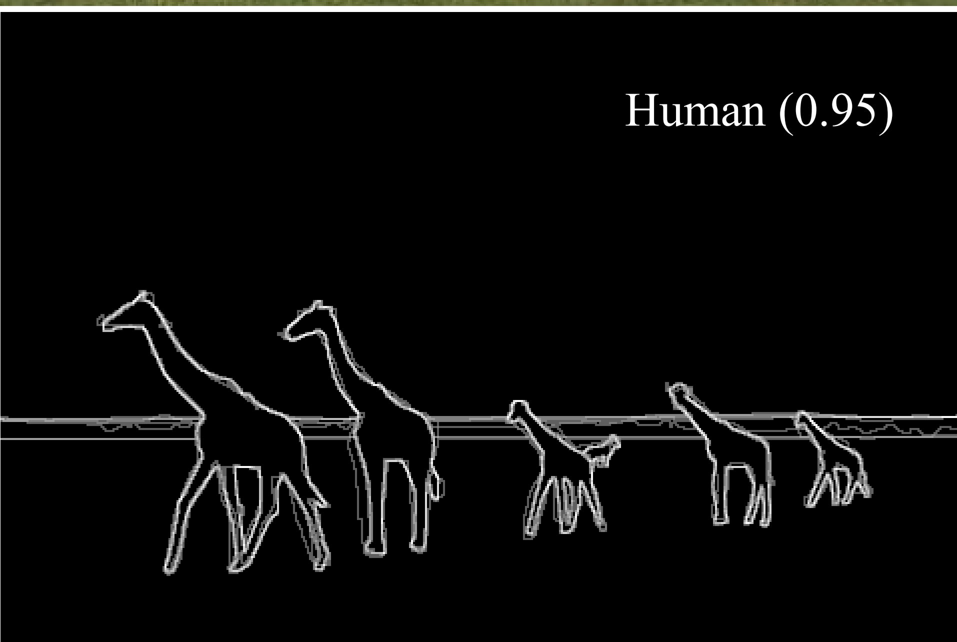
Results





Results




Human (0.95)


Pb (0.63)

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

Contour Detection and Hierarchical Image Segmentation

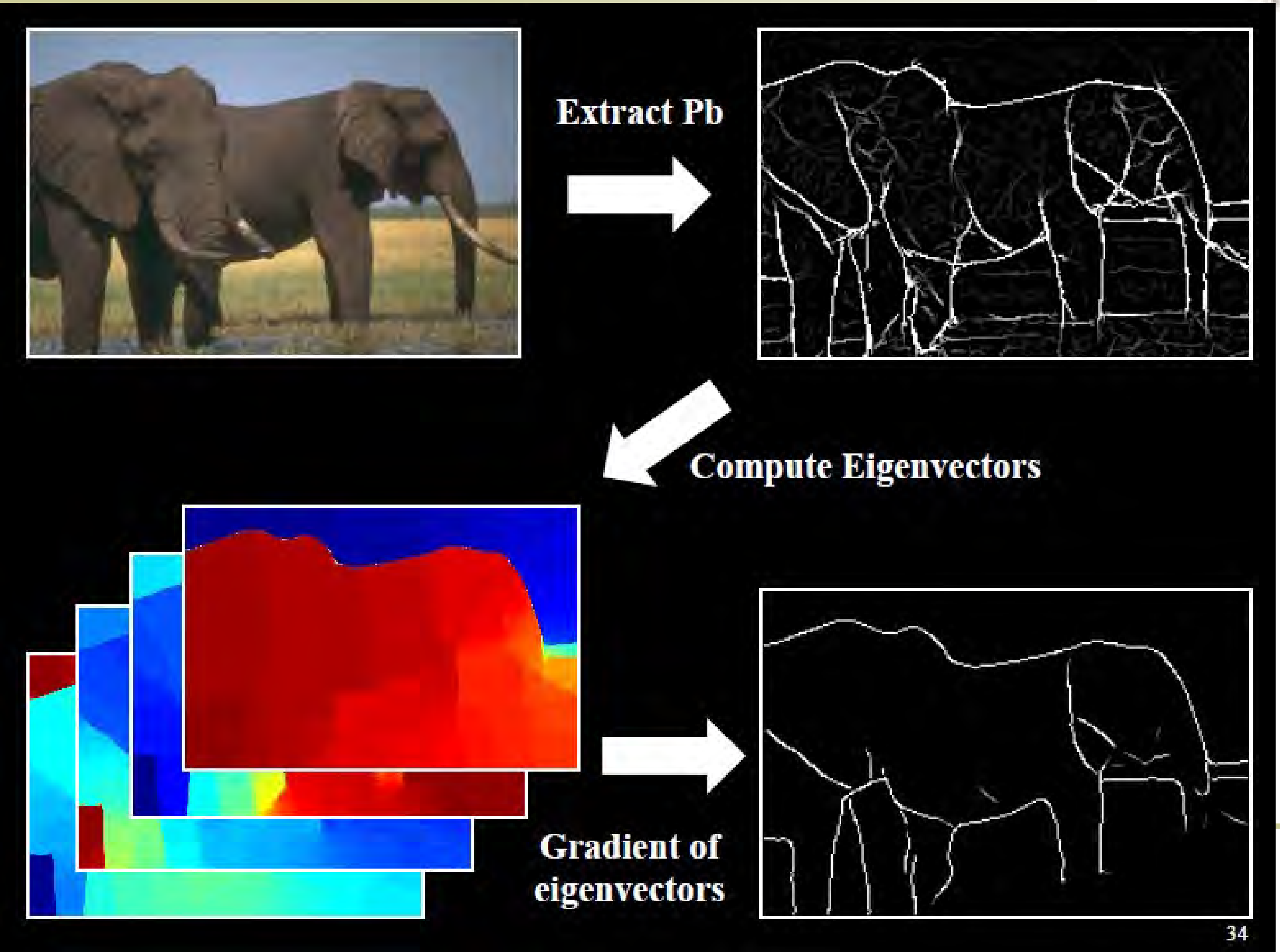
Pablo Arbeláez, *Member, IEEE,* Michael Maire, *Member, IEEE,*
Charless Fowlkes, *Member, IEEE,* and Jitendra Malik, *Fellow, IEEE.*

Abstract—This paper investigates two fundamental problems in computer vision: contour detection and image segmentation. We present state-of-the-art algorithms for both of these tasks. Our contour detector combines multiple local cues into a globalization framework based on spectral clustering. Our segmentation algorithm consists of generic machinery for transforming the output of any contour detector into a hierarchical region tree. In this manner, we reduce the problem of image segmentation to that of contour detection. Extensive experimental evaluation demonstrates that both our contour detection and segmentation methods significantly outperform competing algorithms. The automatically generated hierarchical segmentations can be interactively refined by user-specified annotations. Computation at multiple image resolutions provides a means of coupling our system to recognition applications.

Spectral Pb

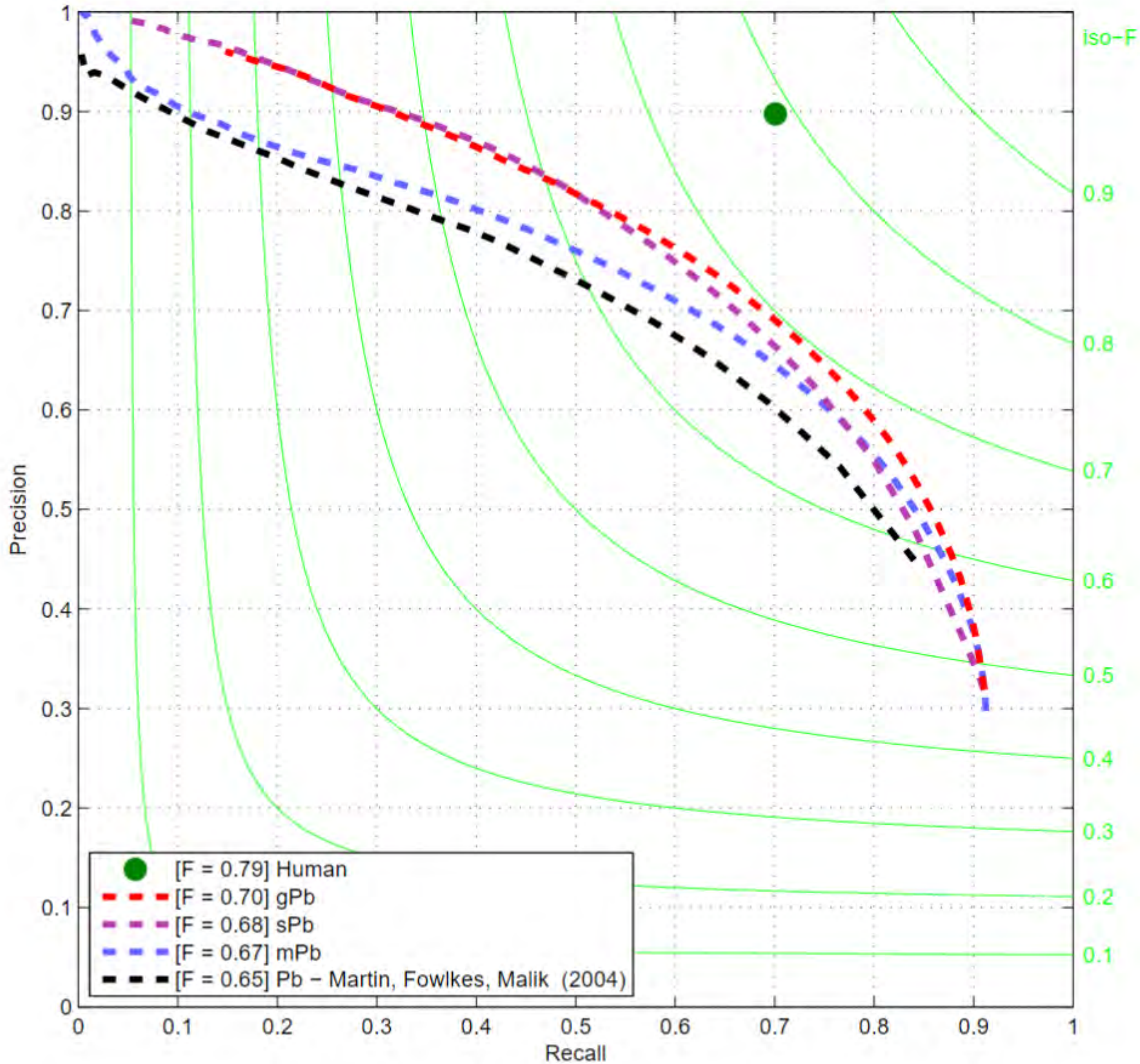
Fig. 7. **Spectral Pb.** ***Left:*** Image. ***Middle Left:*** The thinned non-max suppressed multiscale Pb signal defines a sparse affinity matrix connecting pixels within a fixed radius. Pixels i and j have a low affinity as a strong boundary separates them, whereas i and k have high affinity. ***Middle:*** First four generalized eigenvectors resulting from spectral clustering. ***Middle Right:*** Partitioning the image by running K-means clustering on the eigenvectors erroneously breaks smooth regions. ***Right:*** Instead, we compute gradients of the eigenvectors, transforming them back into a contour signal.

Global pB boundary detector

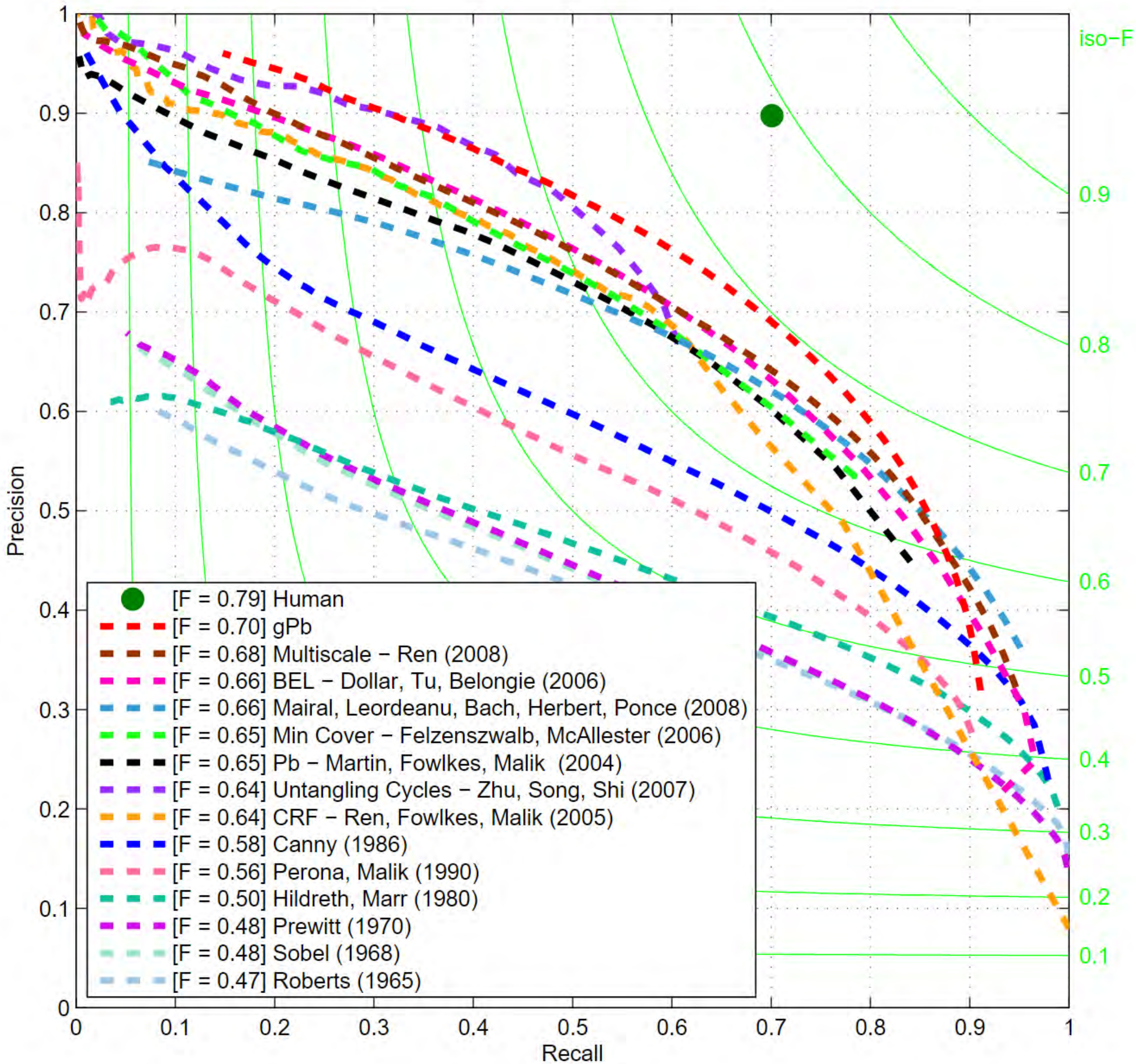


Global pB boundary detector

Fig. 9. **Benefits of globalization.** When compared with the local detector Pb, our detector gPb reduces clutter and completes contours. The thresholds shown correspond to the points of maximal F-measure on the curves in Figure 1.



iso-F
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Precision
Recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
[F = 0.79] Human
[F = 0.70] gPb
[F = 0.68] sPb
[F = 0.67] mPb
[F = 0.65] Pb − Martin, Fowlkes, Malik (2004)



iso-F
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Precision
Recall
[F = 0.79] Human
[F = 0.70] gPb
[F = 0.68] Multiscale − Ren (2008)
[F = 0.66] BEL − Dollar, Tu, Belongie (2006)
[F = 0.66] Mairal, Leordeanu, Bach, Herbert, Ponce (2008)
[F = 0.65] Min Cover − Felzenszwalb, McAllester (2006)
[F = 0.65] Pb − Martin, Fowlkes, Malik (2004)
[F = 0.64] Untangling Cycles − Zhu, Song, Shi (2007)
[F = 0.64] CRF − Ren, Fowlkes, Malik (2005)
[F = 0.58] Canny (1986)
[F = 0.56] Perona, Malik (1990)
[F = 0.50] Hildreth, Marr (1980)
[F = 0.48] Prewitt (1970)
[F = 0.48] Sobel (1968)
[F = 0.47] Roberts (1965)

Fast Edge Detection Using Structured Forests

Piotr Dollár and C. Lawrence Zitnick

Abstract—Edge detection is a critical component of many vision systems, including object detectors and image segmentation algorithms. Patches of edges exhibit well-known forms of local structure, such as straight lines or T-junctions. In this paper we take advantage of the structure present in local image patches to learn both an accurate and computationally efficient edge detector. We formulate the problem of predicting local edge masks in a structured learning framework applied to random decision forests. Our novel approach to learning decision trees robustly maps the structured labels to a discrete space on which standard information gain measures may be evaluated. The result is an approach that obtains realtime performance that is orders of magnitude faster than many competing state-of-the-art approaches, while also achieving state-of-the-art edge detection results on the BSDS500 Segmentation dataset and NYU Depth dataset. Finally, we show the potential of our approach as a general purpose edge detector by showing our learned edge models generalize well across datasets.

Index Terms—Edge detection, segmentation, structured random forests, real-time systems, visual features

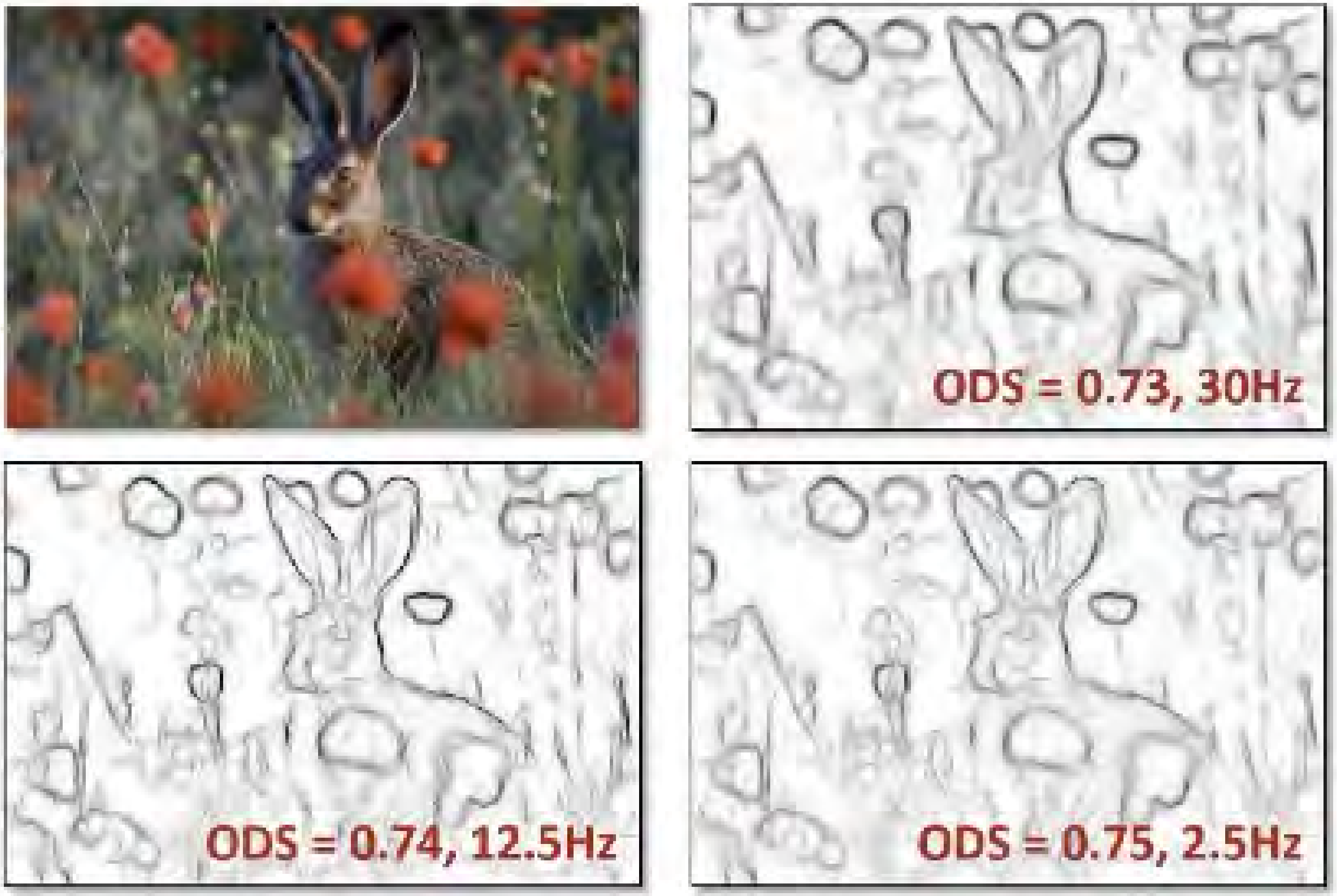


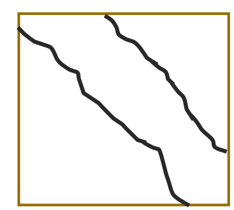
Fig. 1. Edge detection results using three versions of our Structured Edge (SE) detector demonstrating tradeoffs in accuracy vs. runtime. We obtain realtime performance while simultaneously achieving state-of-the-art results. ODS numbers were computed on BSDS [1] on which the popular gPb detector [1] achieves a score of .73. The variants shown include SE, SE+SH, and SE+MS+SH, see §4 for details.

Edge Detection with Structured Random Forests

- 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

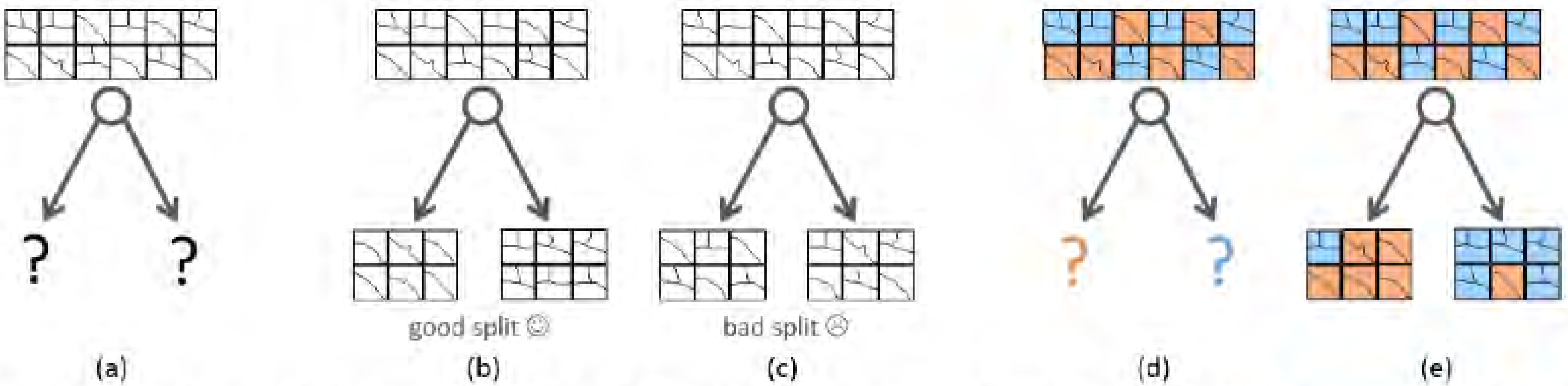
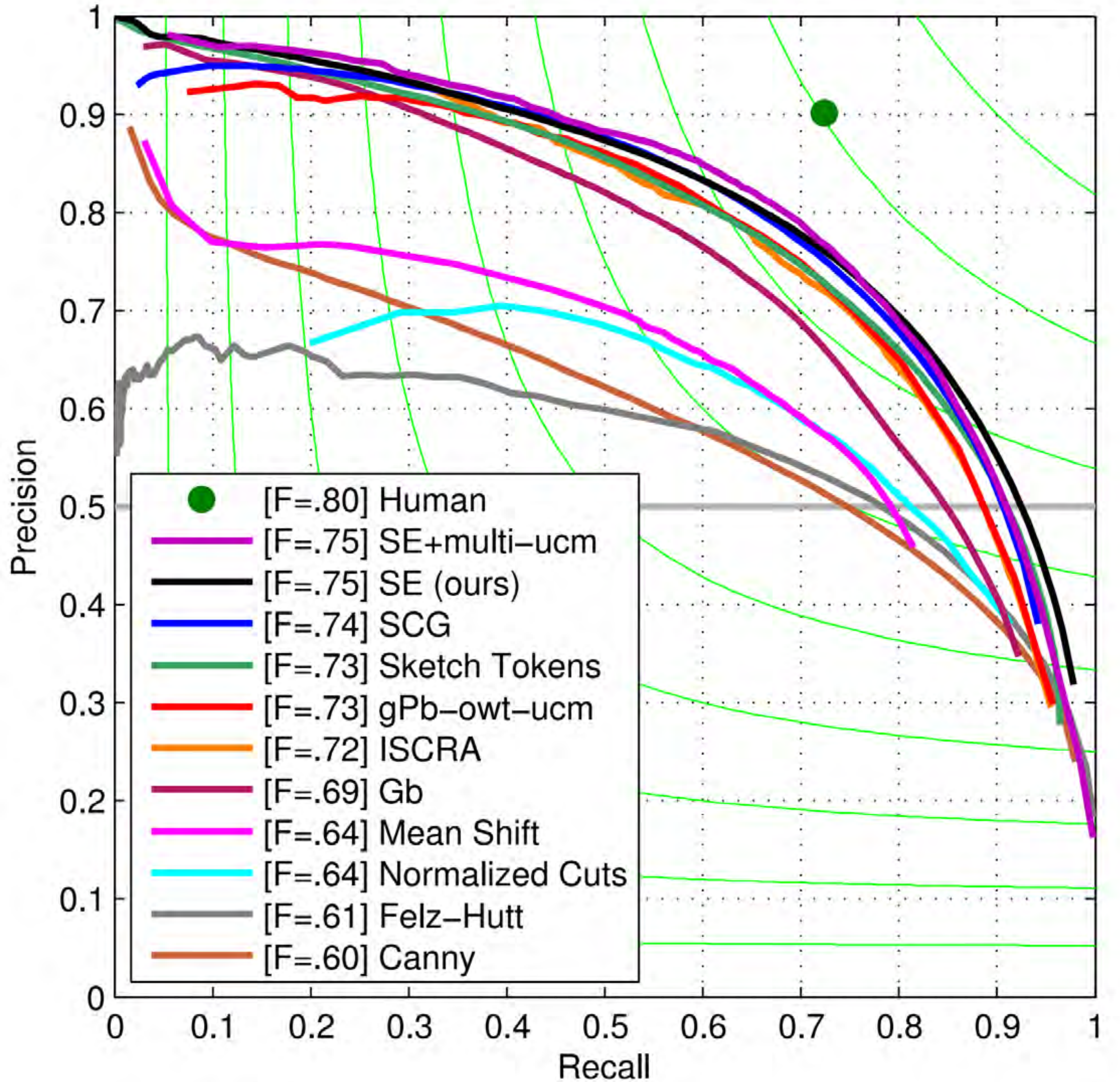


Fig. 2. Illustration of the decision tree node splits: (a) Given a set of structured labels such as segments, a splitting function must be determined. Intuitively a good split (b) groups similar segments, whereas a bad split (c) does not. In practice we cluster the structured labels into two classes (d). Given the class labels, a standard splitting criterion, such as Gini impurity, may be used (e).



Precision
Recall
[F=.80] Human
[F=.75] SE+multi−ucm
[F=.75] SE (ours)
[F=.74] SCG
[F=.73] Sketch Tokens
[F=.73] gPb−owt−ucm
[F=.72] ISCRA
[F=.69] Gb
[F=.64] Mean Shift
[F=.64] Normalized Cuts
[F=.61] Felz−Hutt
[F=.60] Canny

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

Holistically-Nested Edge Detection

Saining Xie
Dept. of CSE and Dept. of CogSci
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
s9xie@eng.ucsd.edu

Zhuowen Tu
Dept. of CogSci and Dept. of CSE
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093
ztu@ucsd.edu

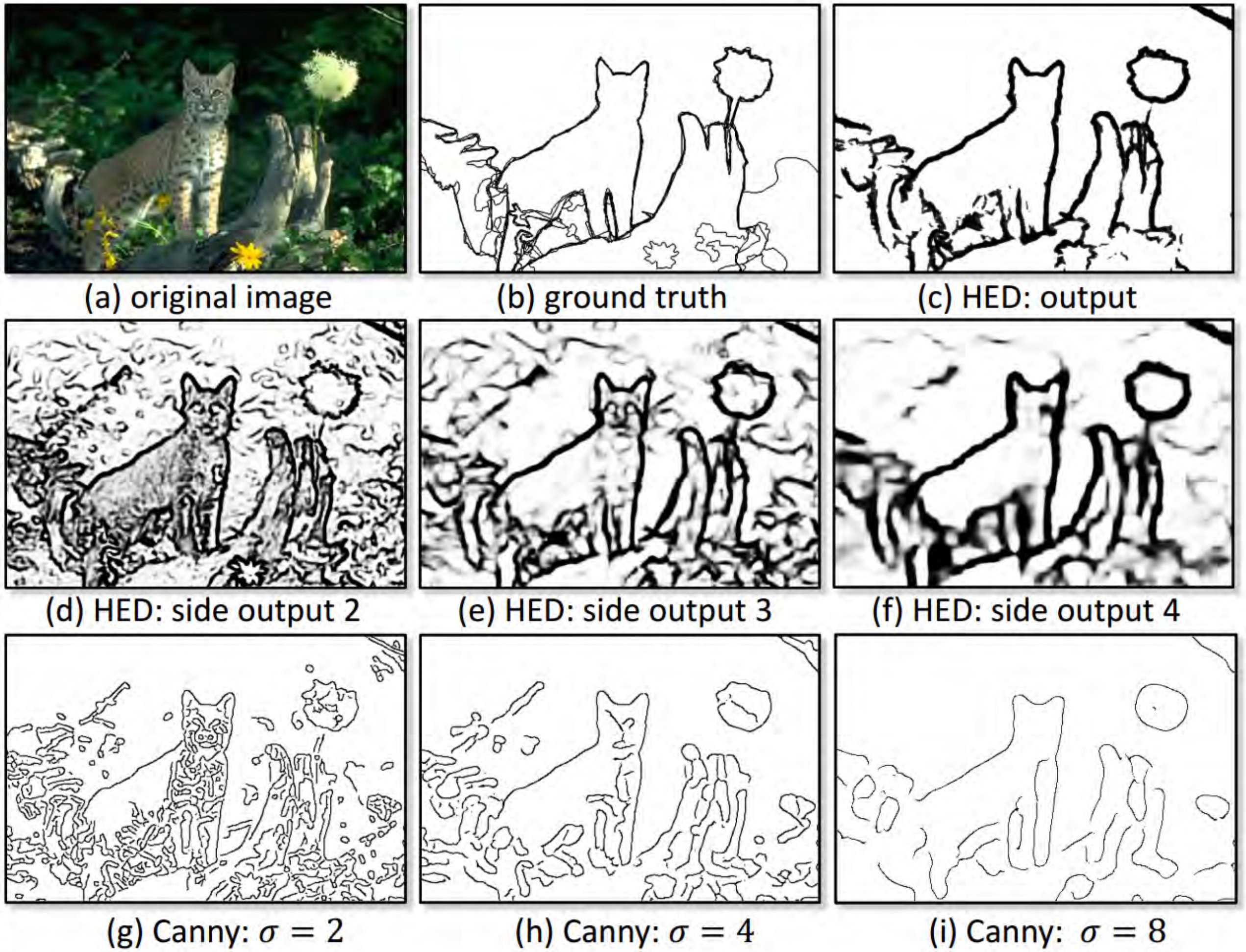


Figure 1. Illustration of the proposed HED algorithm. In the first row: (a) shows an example test image in the BSD500 dataset [28]; (b) shows its corresponding edges as annotated by human subjects; (c) displays the HED results. In the second row: (d), (e), and (f), respectively, show side edge responses from layers 2, 3, and 4 of our convolutional neural networks. In the third row: (g), (h), and (i), respectively, show edge responses from the Canny detector [4] at the scales $\sigma = 2.0$, $\sigma = 4.0$, and $\sigma = 8.0$. HED shows a clear advantage in consistency over Canny.

Holistically nested edge detection

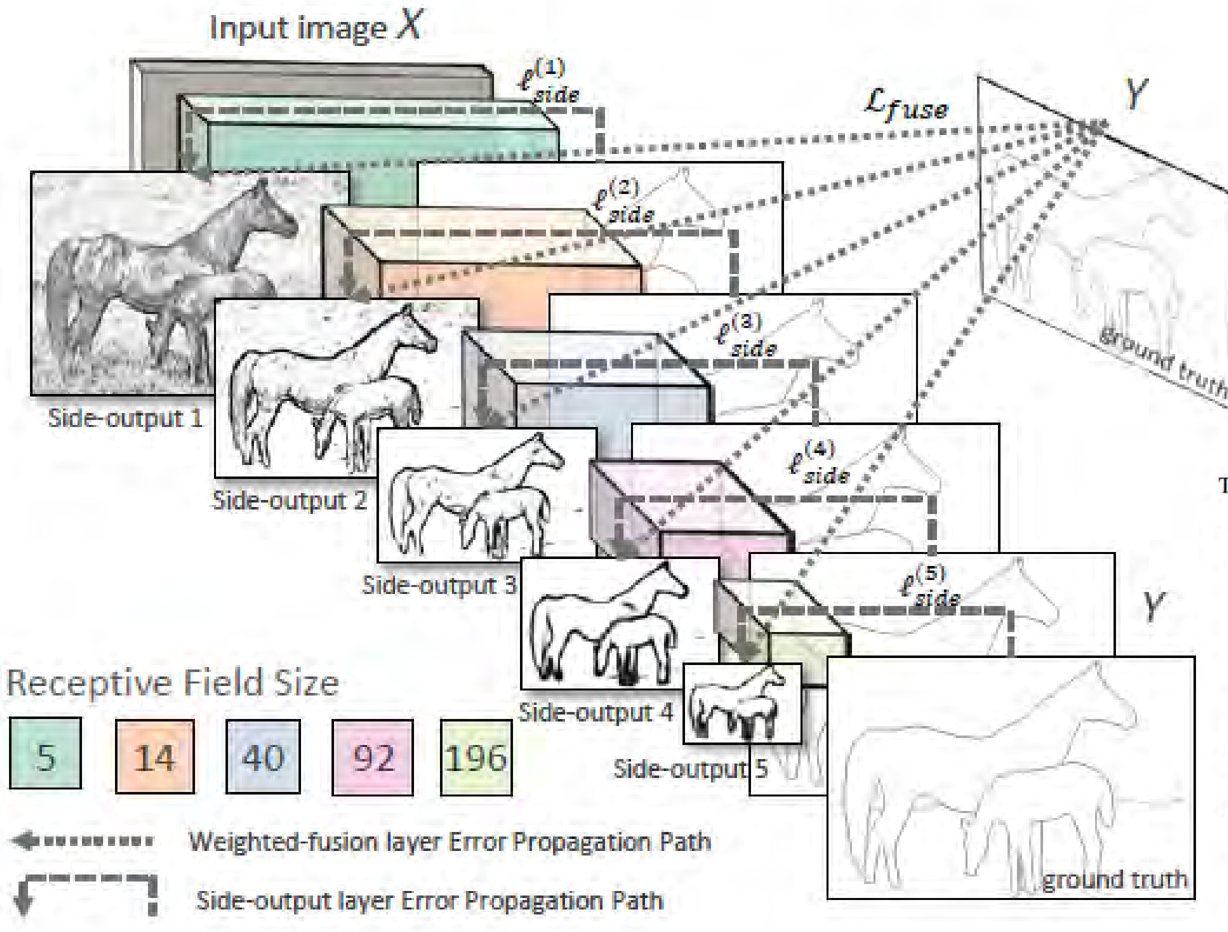
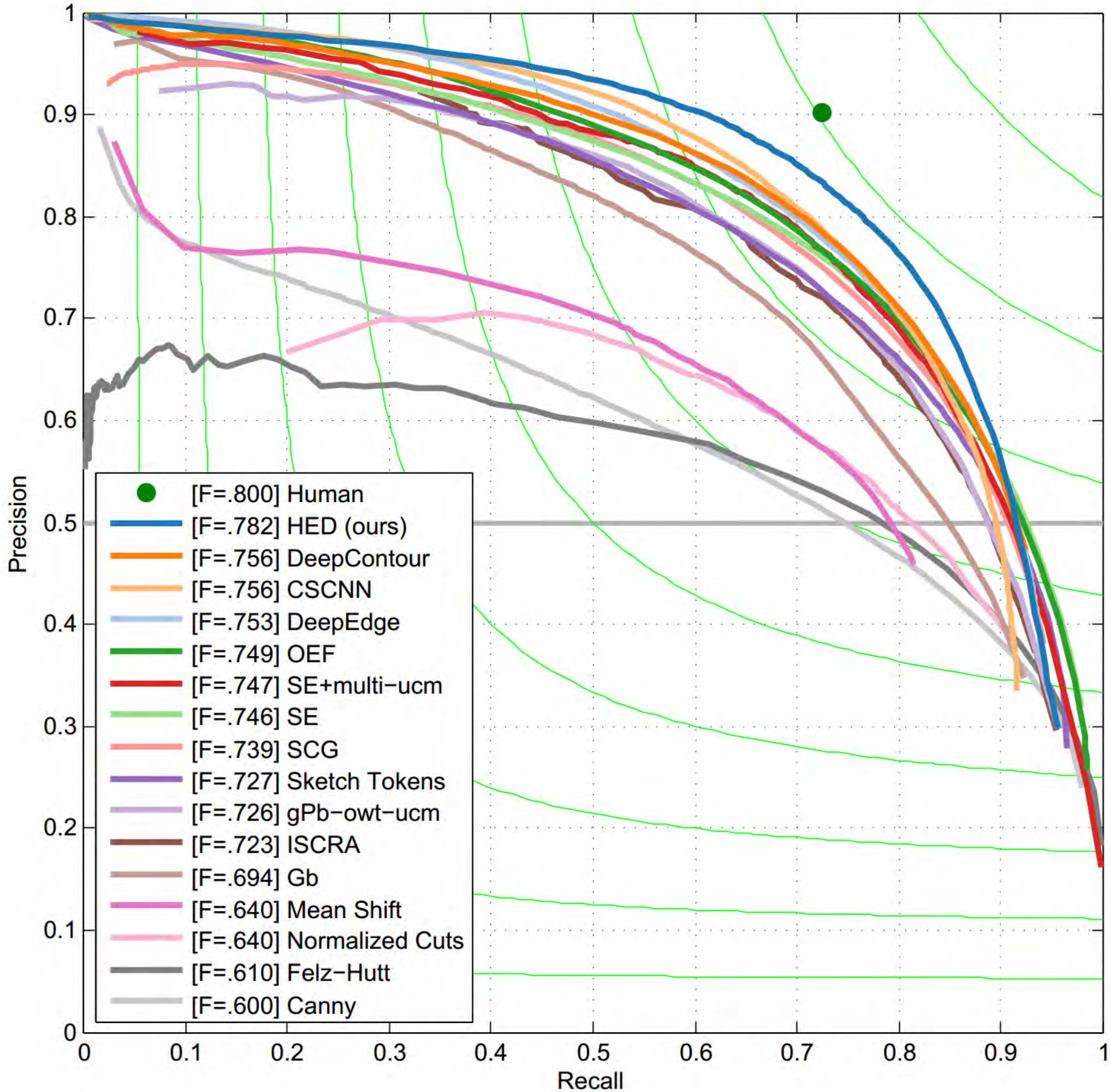


Table 4. Results on BSDS500. ∗BSDS300 results,†GPU time

	ODS	OIS	AP	FPS
Human	.80	.80	-	-
Canny	.600	.640	.580	15
Felz-Hutt [9]	.610	.640	.560	10
BEL [5]	.660∗	-	-	1/10
gPb-owt-ucm [1]	.726	.757	.696	1/240
Sketch Tokens [24]	.727	.746	.780	1
SCG [31]	.739	.758	.773	1/280
SE-Var [6]	.746	.767	.803	2.5
OEF [13]	.749	.772	.817	-
DeepNets [21]	.738	.759	.758	1/5†
N4-Fields [10]	.753	.769	.784	1/6†
DeepEdge [2]	.753	.772	.807	$1/10^3$†
CSCNN [19]	.756	.775	.798	-
DeepContour [34]	.756	.773	.797	1/30†
HED (ours)	**.782**	**.804**	**.833**	2.5†, 1/12

https://arxiv.org/pdf/1504.06375.pdf



Precision
Recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
[F=.800] Human
[F=.782] HED (ours)
[F=.756] DeepContour
[F=.756] CSCNN
[F=.753] DeepEdge
[F=.749] OEF
[F=.747] SE+multi−ucm
[F=.746] SE
[F=.739] SCG
[F=.727] Sketch Tokens
[F=.726] gPb−owt−ucm
[F=.723] ISCRA
[F=.694] Gb
[F=.640] Mean Shift
[F=.640] Normalized Cuts
[F=.610] Felz−Hutt
[F=.600] Canny

State of edge detection

- Local edge detection is mostly solved
  - Intensity gradient, color, texture
  - HED on BSDS 500 is near human performance
- Some room for improvement by taking advantage of higher-level knowledge (e.g., objects)
- Still hard to produce all objects within a small number of regions

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

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

Larger eigenvector

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

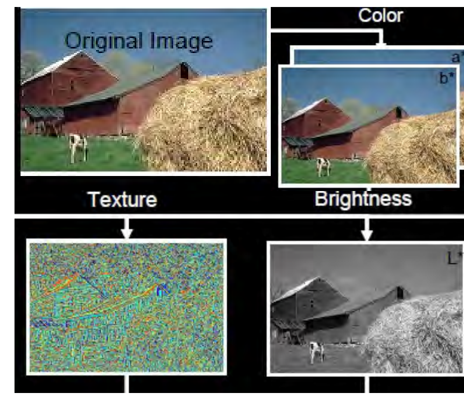
$$conf = \lambda_2 / \lambda_1$$

5. Threshold on straightness, store segment

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