Edge Detection and Corner Detection

Computer Vision I

CSE 252A

Lecture 6

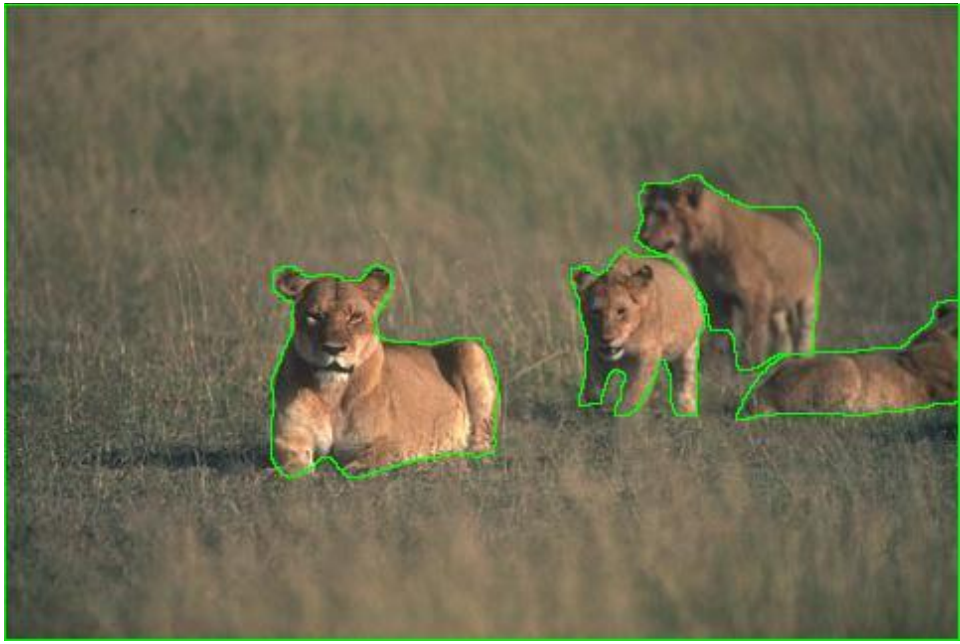
Announcements

- Assignment 1 is due Oct 25, 11:59 PM
- Assignment 2 will be released Oct 25
  - Due Nov 8, 11:59 PM

Image Segmentation and Edges

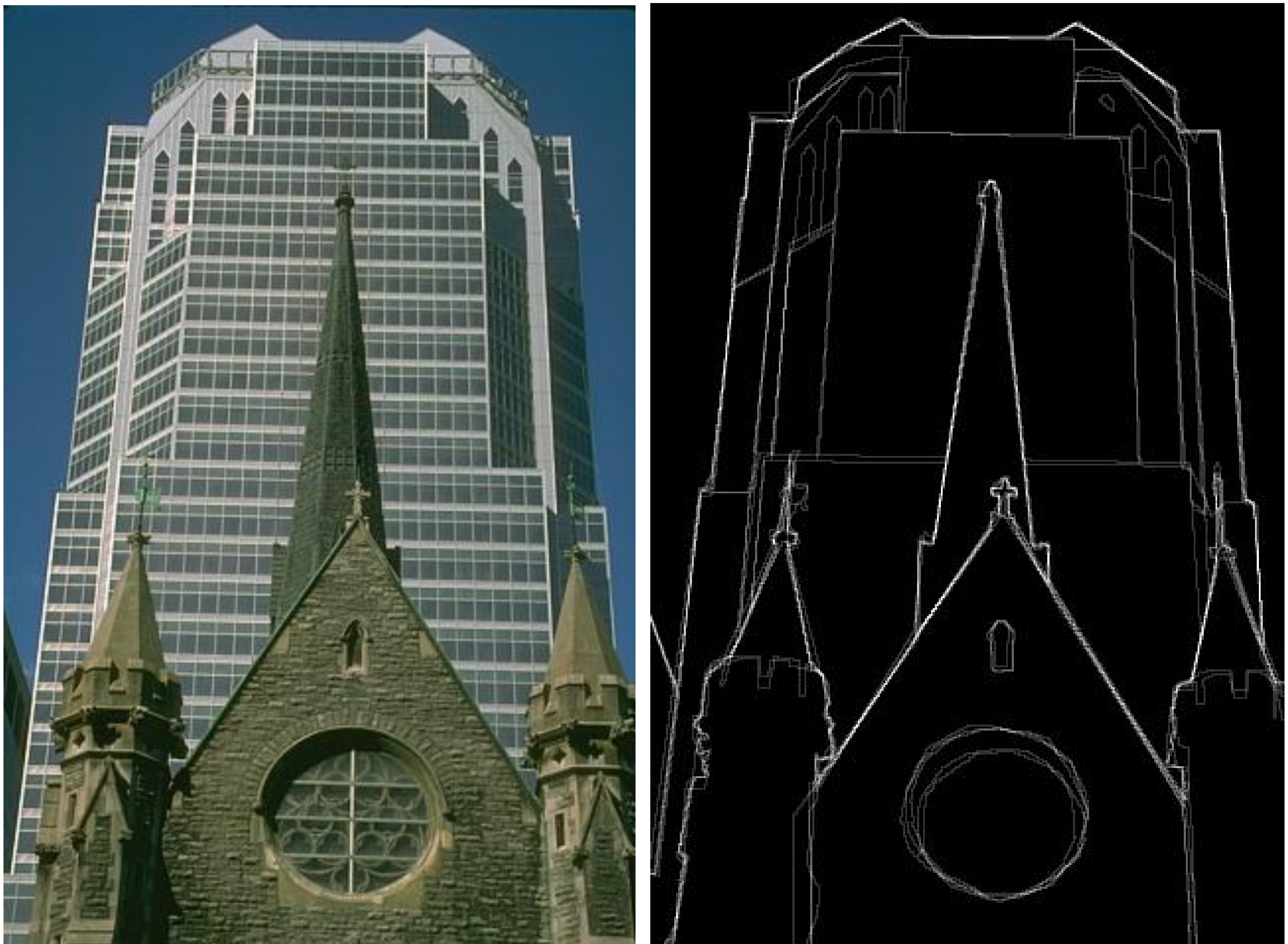
- Image Segmentation is the process of dividing an image into connected regions such that pixels within a region share certain characteristics (color, texture , brightness, etc.)
- Boundaries or edges divide segmented regions.

[From Berkeley Segmentation Dataset]



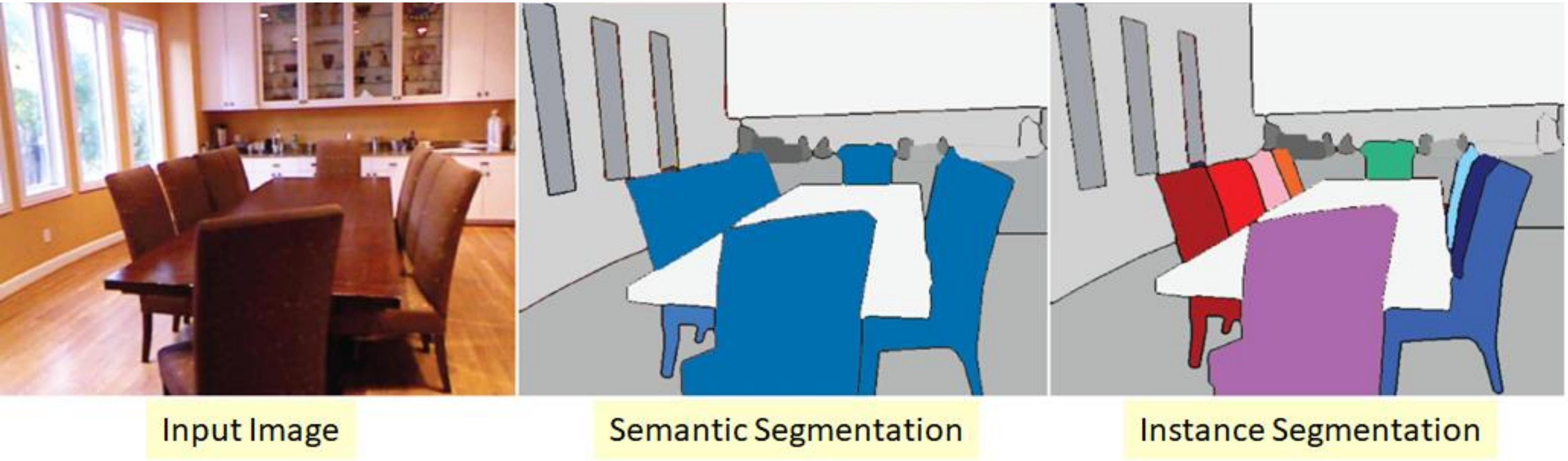
13 Regions

Image Segmentation



[From Berkeley Segmentation Dataset]

Related Topics:
Semantic and Instance Segmentation

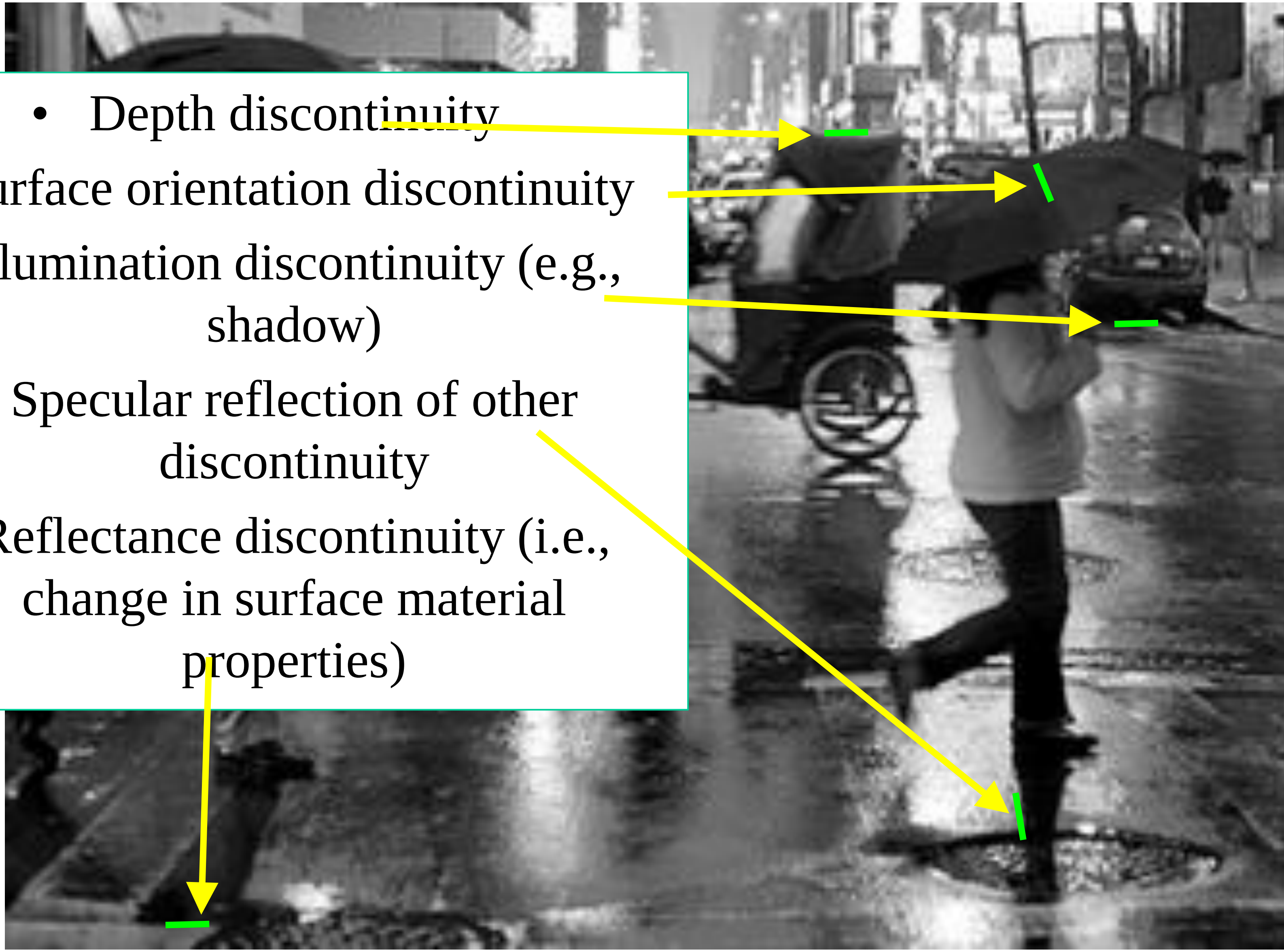


Edges in Natural Images

Source: Photografr.com

What Causes an Edge?

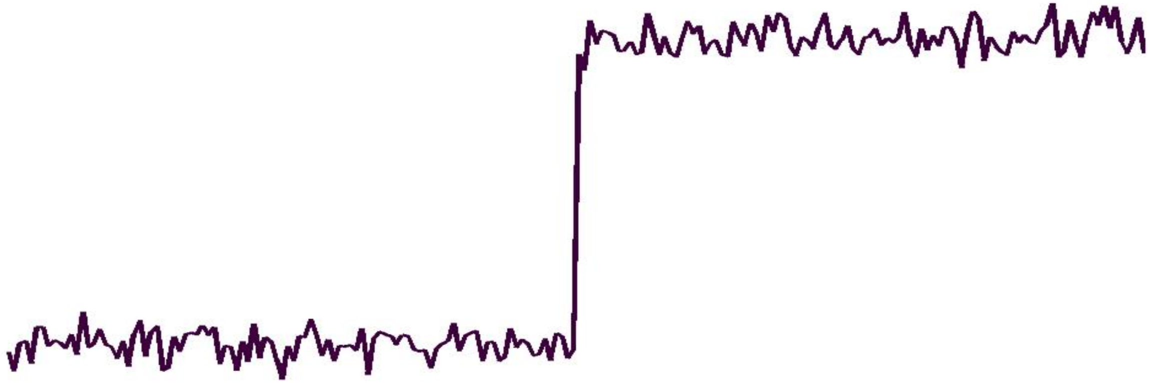
- Depth discontinuity
- Surface orientation discontinuity
- Illumination discontinuity (e.g., shadow)
- Specular reflection of other discontinuity
- Reflectance discontinuity (i.e., change in surface material properties)



Source: Photografr.com

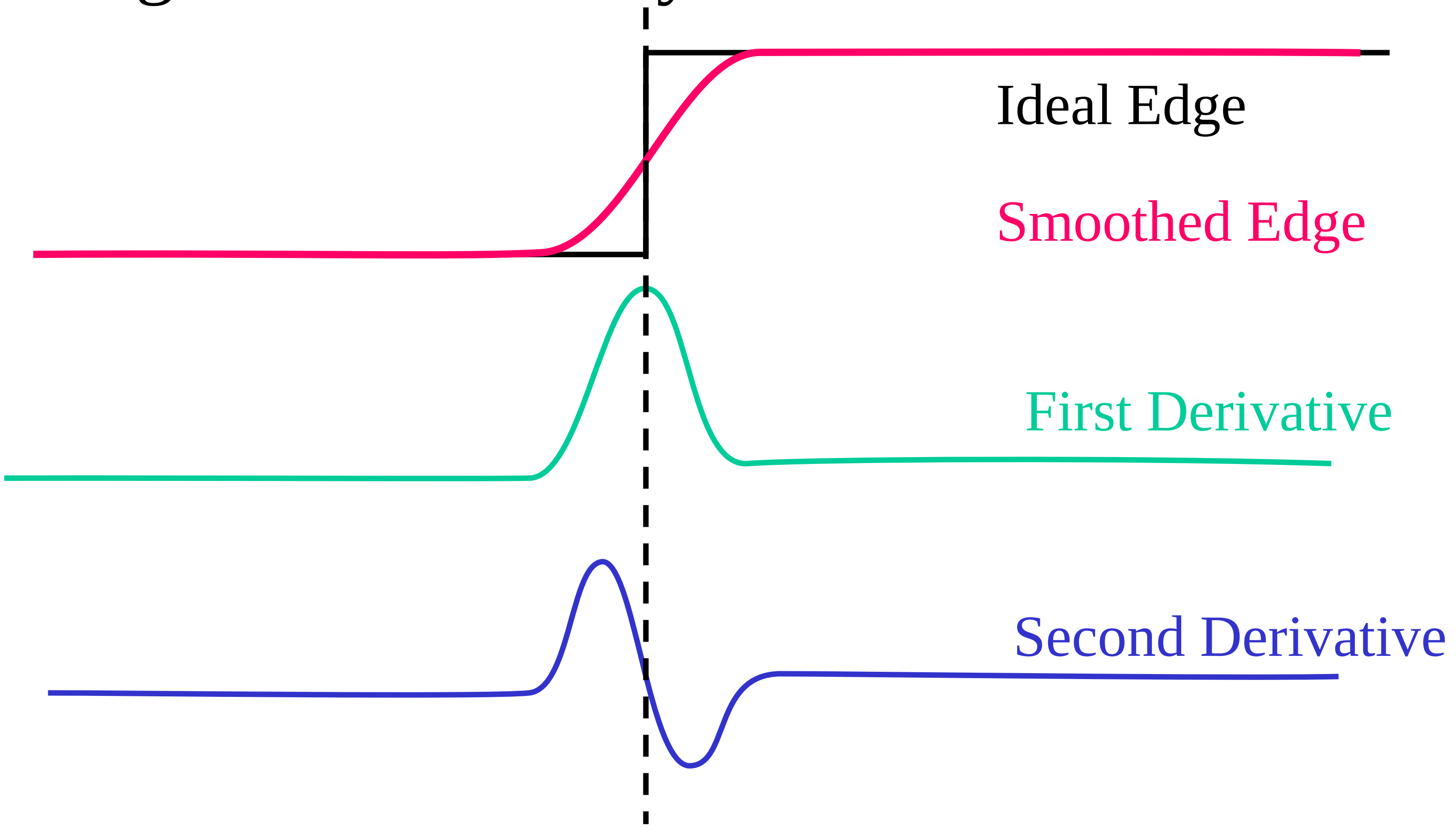
Noisy 1D Step Edge

- Derivative is high everywhere.
- Must smooth before taking gradient.



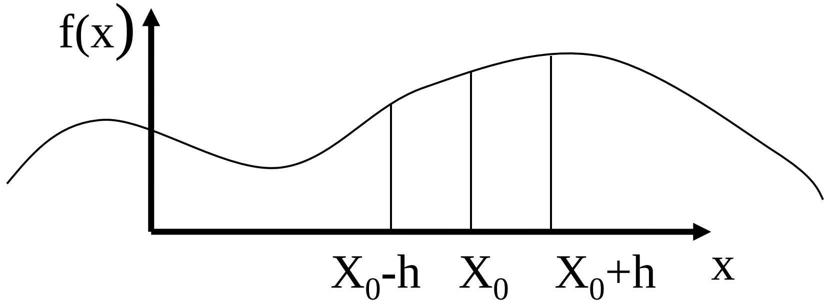
Edge is Where Change Occurs: 1-D

- Change is measured by derivative in 1D



- Biggest change, first derivative has maximum magnitude
- Or second derivative is zero

Numerical Derivatives of Sampled Signal



Take Taylor series expansion of f(x) about x_0

$$f(x) = f(x_0)+f'(x_0)(x-x_0) + \tfrac{1}{2} f''(x_0)(x-x_0)^2 + \cdots$$

Consider samples taken at increments of h and first two terms of the expansion, we have

$$f(x_0+h) = f(x_0)+f'(x_0)h+ \tfrac{1}{2} f''(x_0)h^2$$

$$f(x_0-h) = f(x_0)-f'(x_0)h+ \tfrac{1}{2} f''(x_0)h^2$$

Subtracting and adding $f(x_0+h)$ and $f(x_0-h)$ respectively yields

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

$$f''(x_0) = \frac{f(x_0+h) - 2f(x_0) + f(x_0-h)}{h^2}$$

Correlate with
First Derivative: [-1/2h 0 1/2h]
Second Derivative: [$1/h^2$ $-2/h^2$ $1/h^2$]

Numerical Derivatives

Kernel
First Derivative: [-1/2h 0 1/2h]
Second Derivative: [$1/h^2$ $-2/h^2$ $1/h^2$]

- With images, units of h is pixels, so h=1
  - First derivative: [-1/2 0 1/2]
  - Second derivative: [1 -2 1]
- When computing derivatives in the x and y directions, use these (correlation) kernels:

$$\frac{d}{dx} = [-1/2 \quad 0 \quad 1/2] \qquad \frac{d}{dy} = \begin{bmatrix} -1/2 \\ 0 \\ 1/2 \end{bmatrix}$$

Implementing 1D Edge Detection

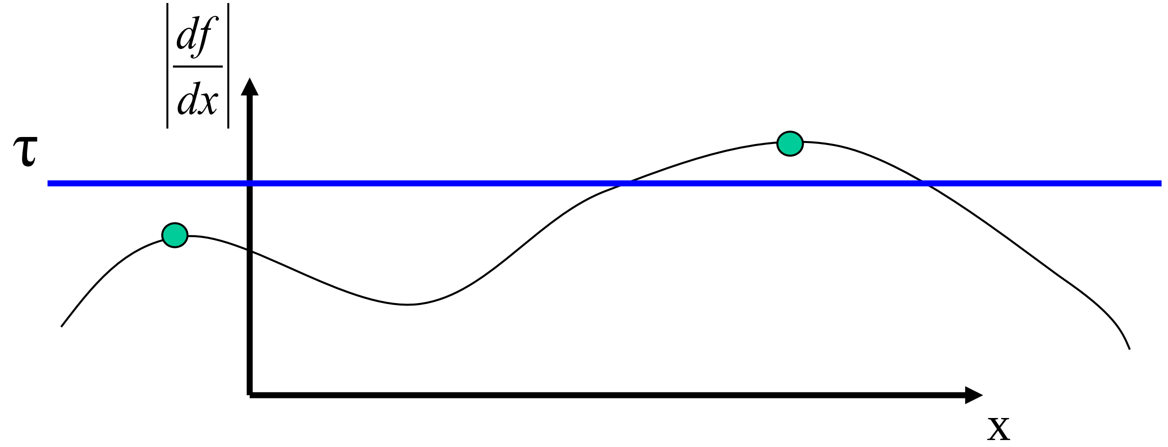
1. Filter out noise: convolve with Gaussian
2. Take a derivative: convolve with [-1/2 0 1/2] (correlation kernel)

   We can combine steps 1 and 2

3. Find the peaks of |df/dx|

   Two issues:

   - Should be a local maximum of |df/dx|
   - Should be greater than a threshold: $|df/dx| > \tau$



2D Edge Detection

1. Filter out noise
   – Use a 2D Gaussian Filter.
2. Take a derivative
   – Compute the magnitude of the gradient:

$$J = I \,\mathrm{A}\, G$$

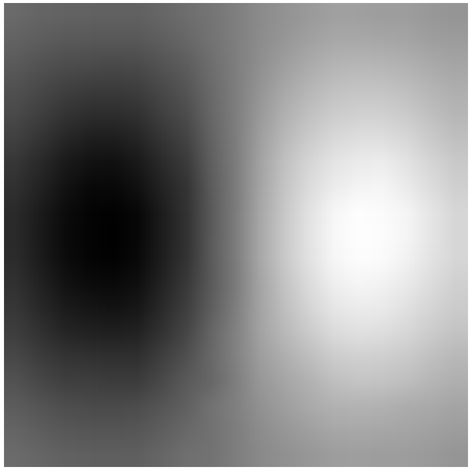
$$\nabla J = (J_x, J_y) = \left(\frac{\P J}{\P x}, \frac{\P J}{\P y} \right)$$ is the gradient

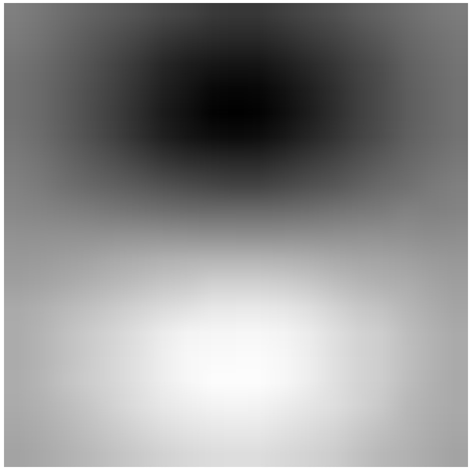
$$\left\| \nabla J \right\| = \sqrt{J_x^2 + J_y^2}$$ is the magnitude of the gradient

$$\tan^{-1}\left(\frac{J_y}{J_x}\right)$$ the direction of the gradient

Smoothing and Differentiation

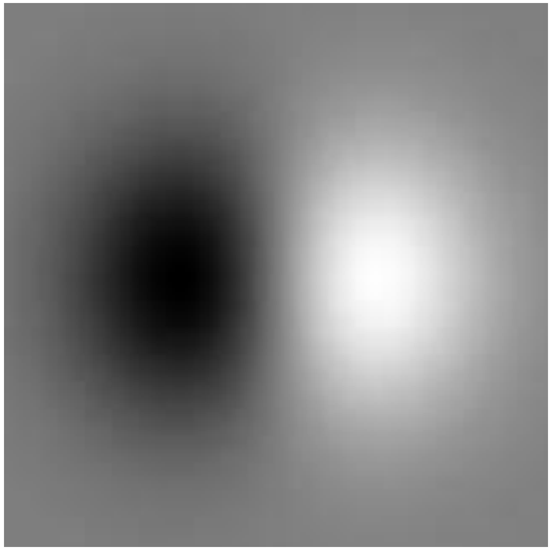
- Need two derivatives, in x and y direction.
- Filter with Gaussian and then compute Gradient, OR
- Use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative (shape full convolution is required)



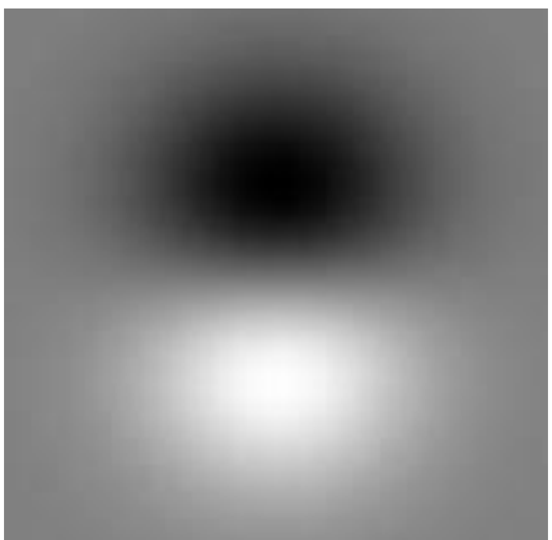


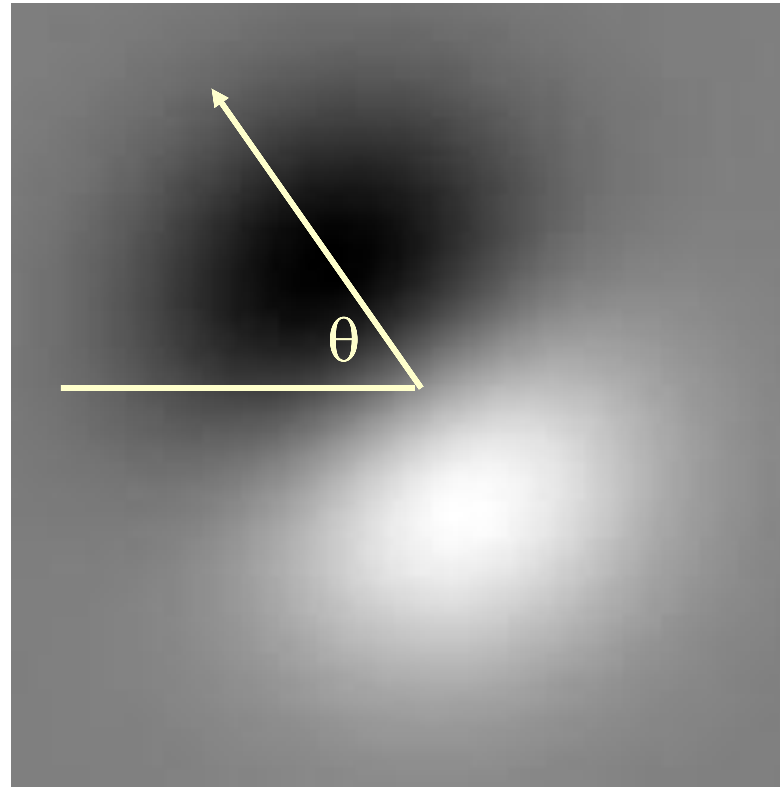
Directional Derivatives

$$\frac{\partial G_\sigma}{\partial x}$$



$$\frac{\partial G_\sigma}{\partial y}$$





$$\cos\theta \frac{\partial G_\sigma}{\partial x} + \sin\theta \frac{\partial G_\sigma}{\partial y}$$

Finding derivatives

Is this dI/dx or dI/dy?

$\sigma = 1$ $\sigma = 2$

There are three major issues:

1. The gradient magnitude at different scales is different; which scale should we choose?
2. The gradient magnitude is large along a thick trail; how do we identify the significant points?
3. How do we link the relevant points up into curves?

There is ALWAYS a tradeoff between smoothing and good edge localization!

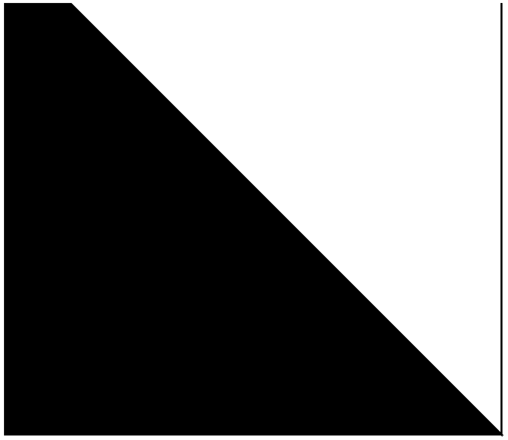
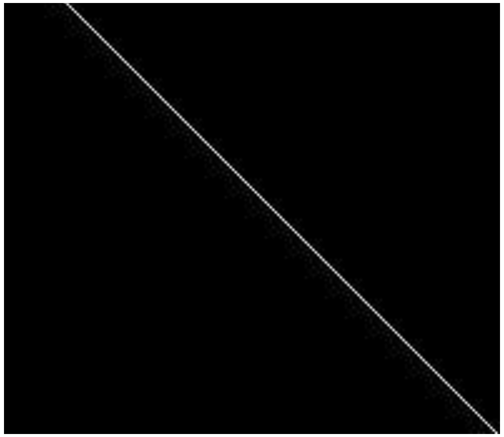


Image with Edge (No Noise)



Edge Location

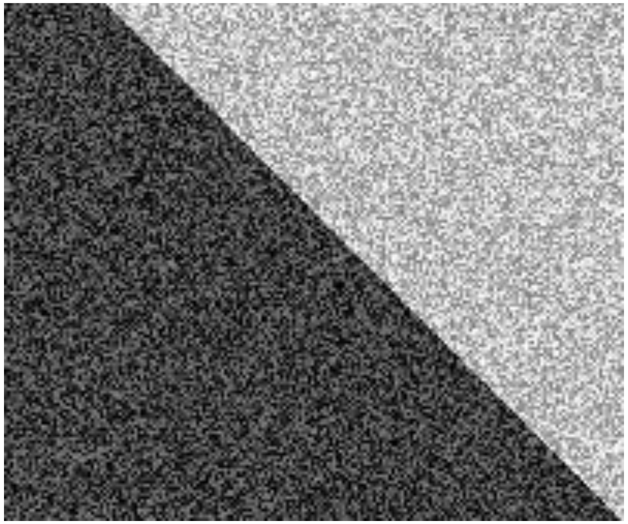
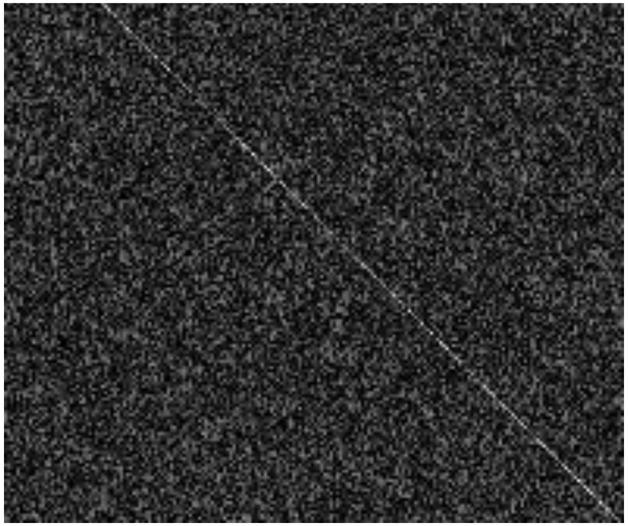
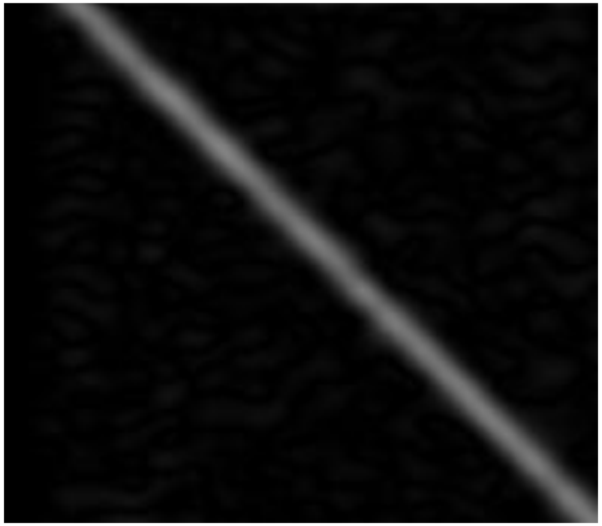


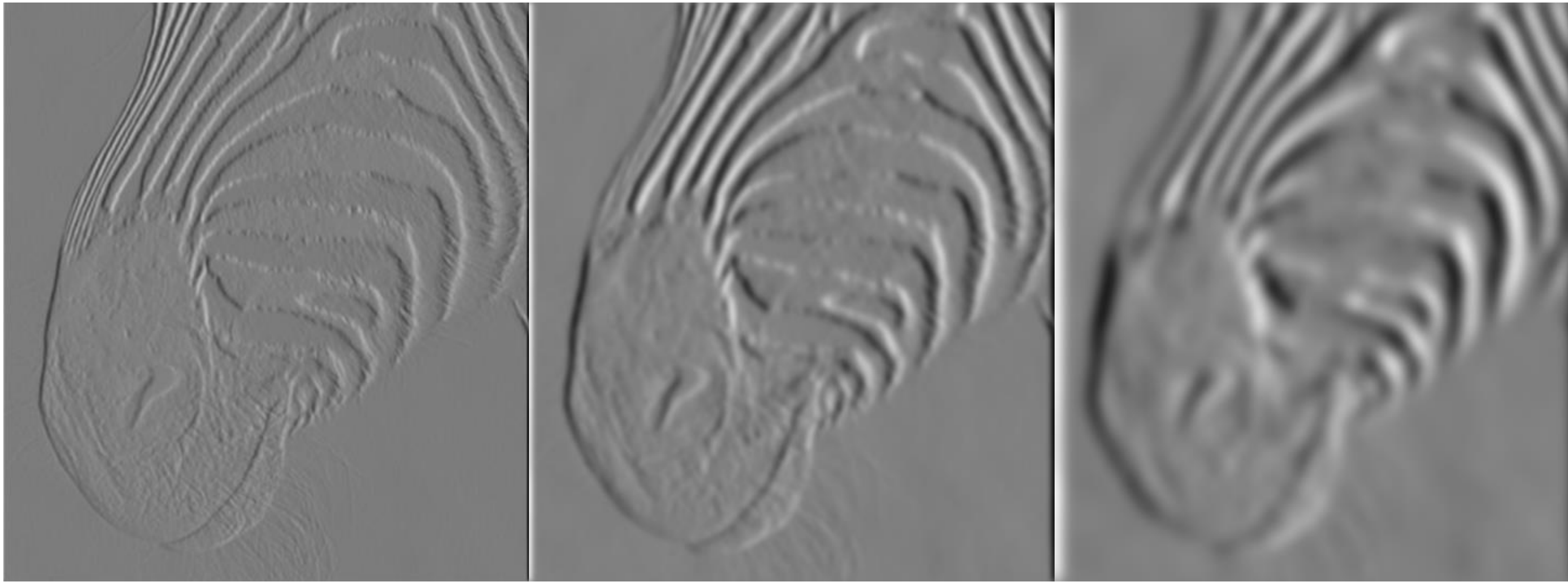
Image + Noise



Derivatives detect edge *and* noise



Smoothed derivative removes noise, but blurs edge

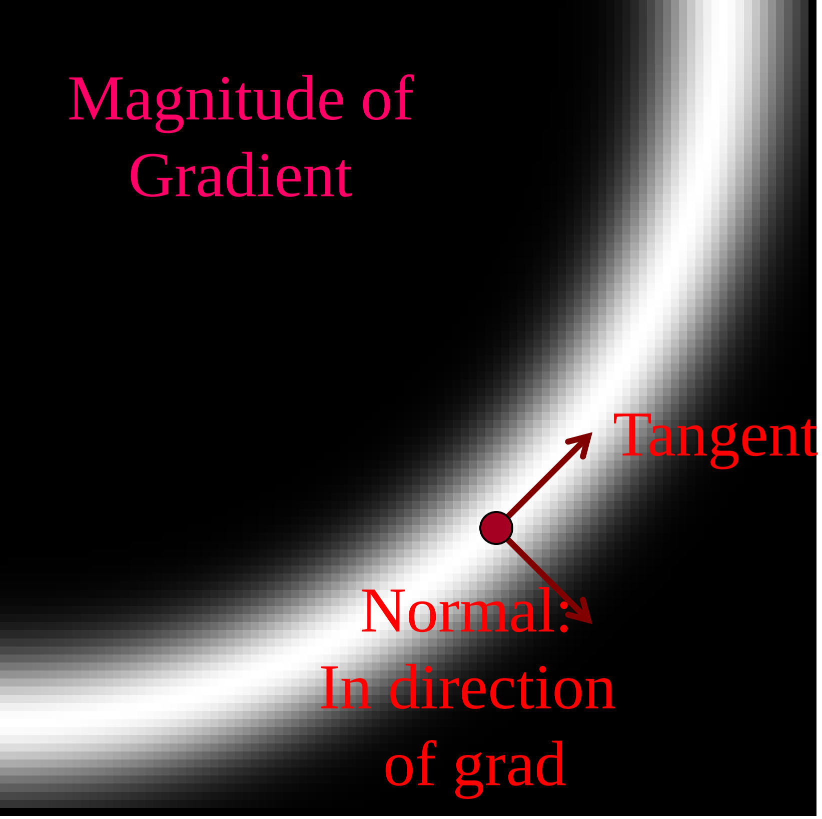


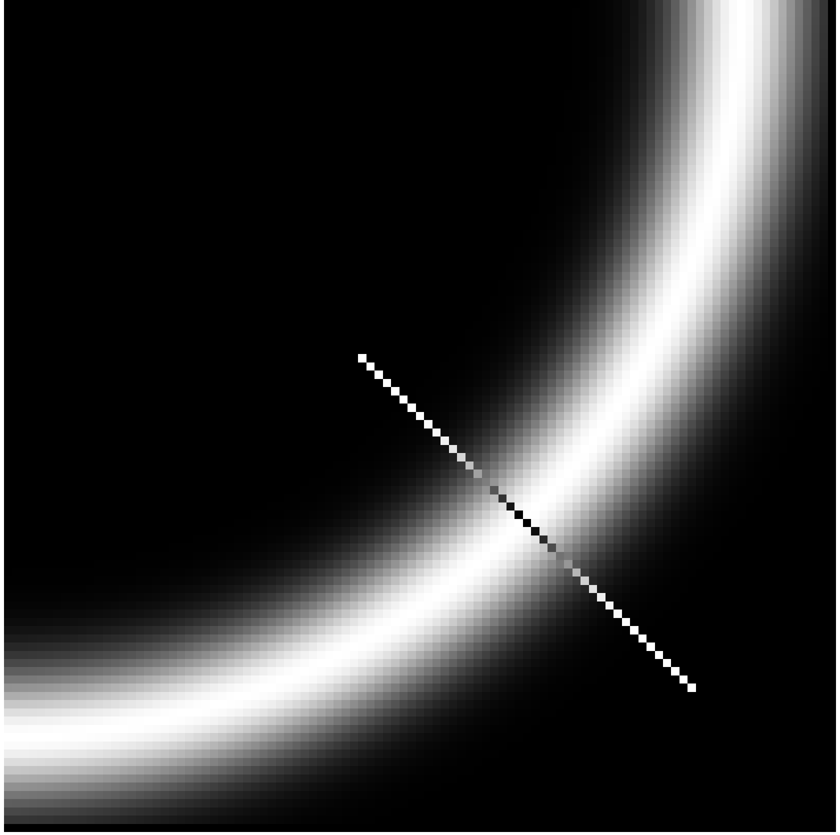
1 pixel

3 pixels

7 pixels

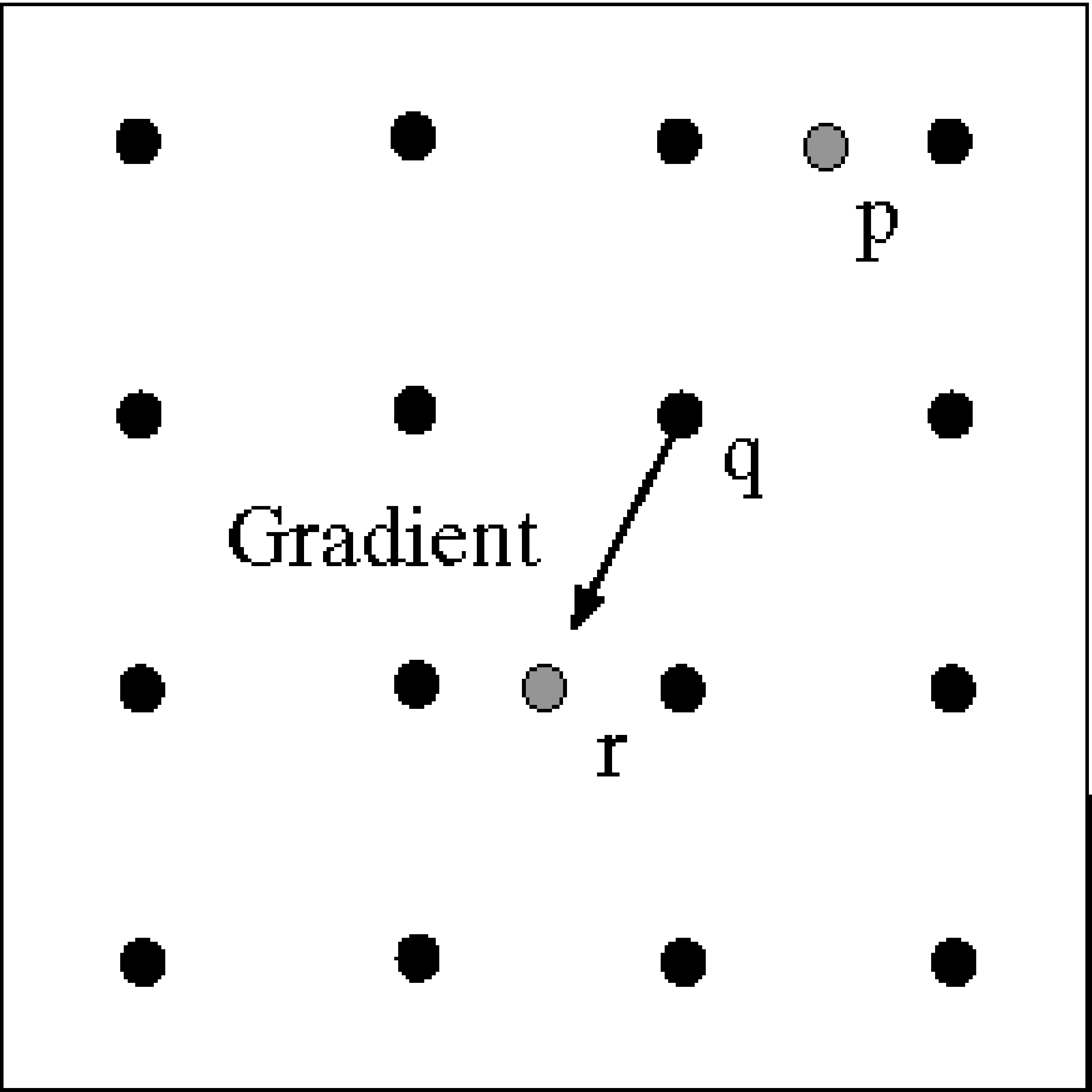
The scale of the smoothing filter affects derivative estimates





We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: which point is the maximum, and where is the next point on the curve?

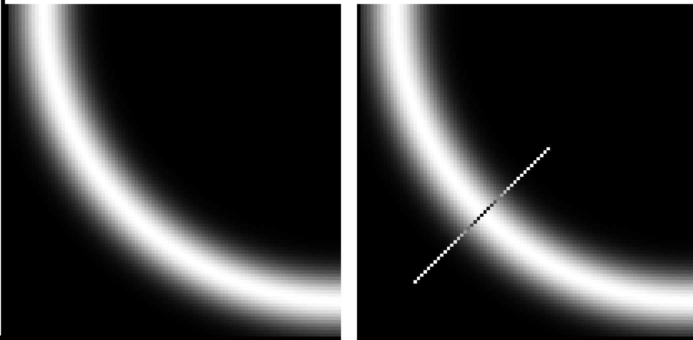
Non-maximum suppression



Using normal at q, find two points p and r on adjacent rows (or columns)

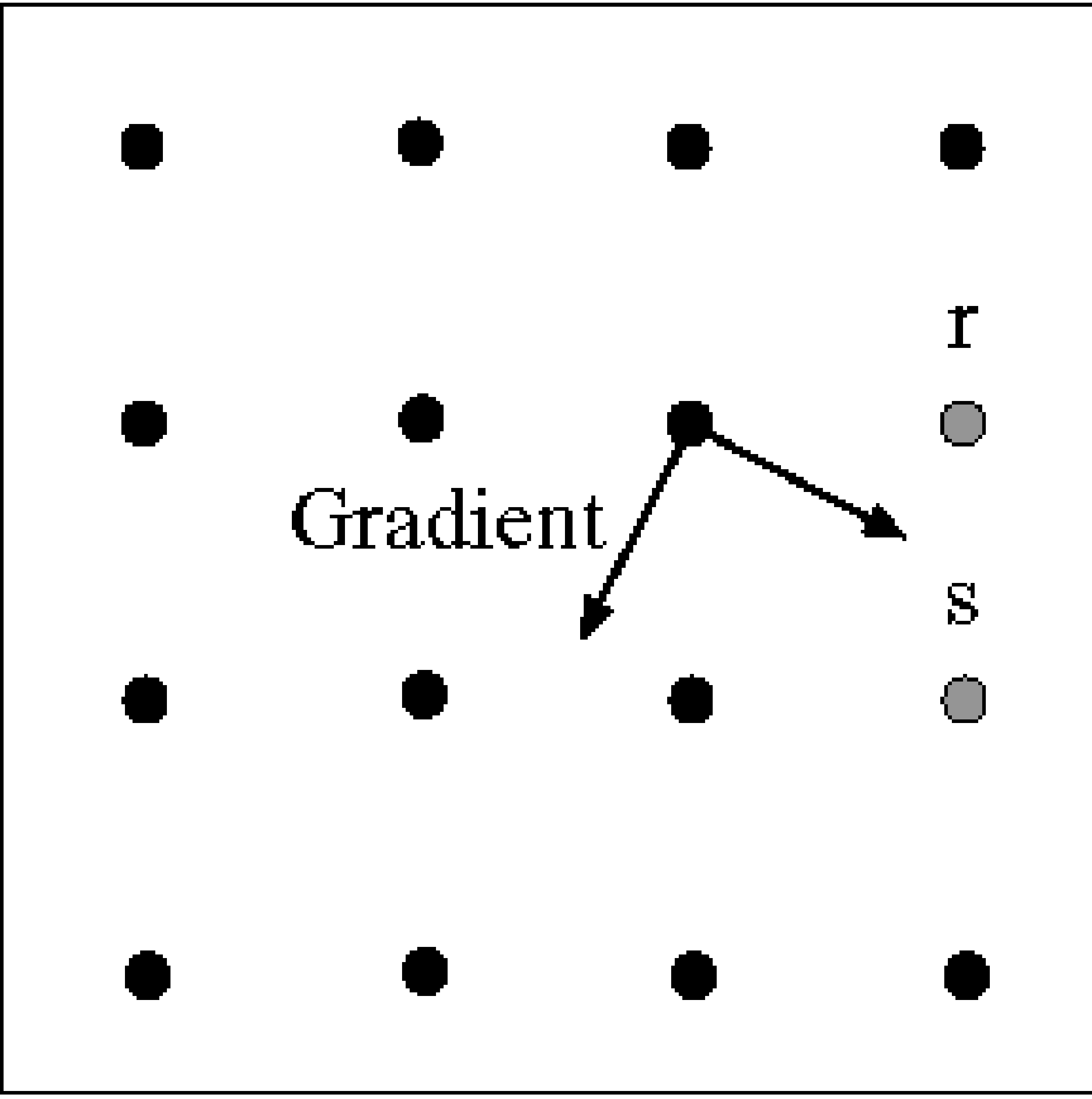
q is a maximum if $\left|\nabla J(q)\right|$ is larger than $\left|\nabla J(p)\right|$ and $\left|\nabla J(r)\right|$

Interpolate to get values

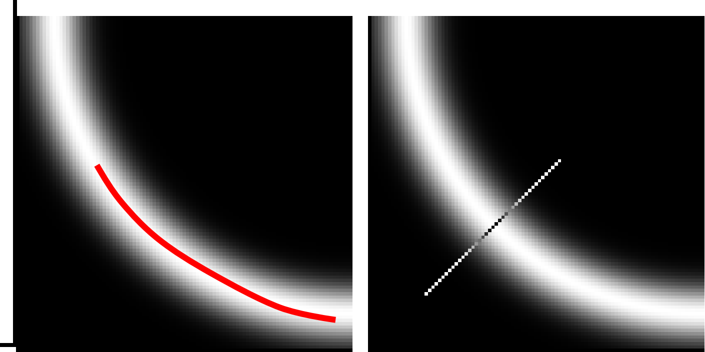


Non-maximum suppression

Predicting the next edge point



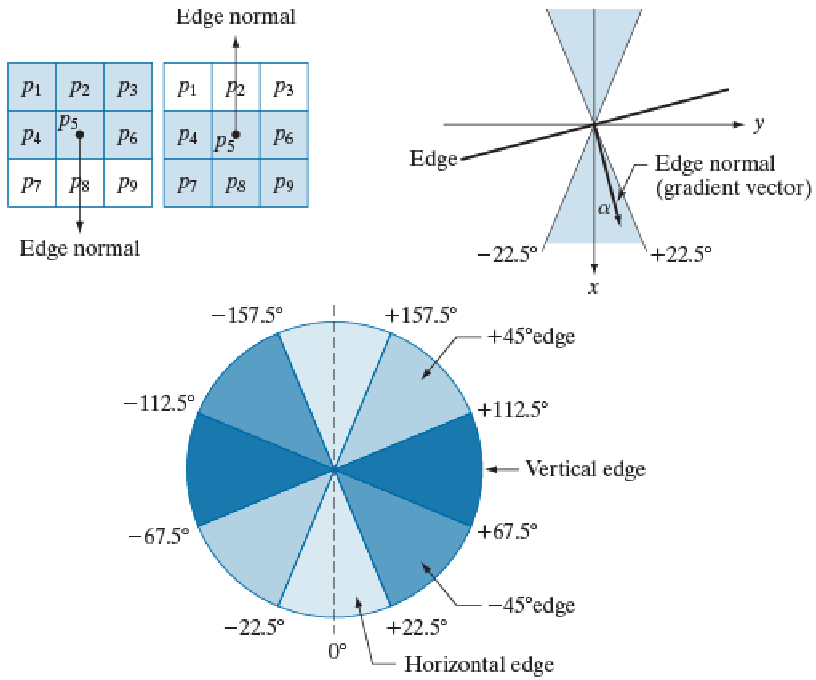
- The marked point is an edge point.
- From edge tangent (normal to gradient), predict next point along edge curve (here either r or s)
- Link together to create edge curve



Nonmaxima suppression (alternative method)

Specify a number of discrete orientations $d_1, d_2, \ldots$

1. Determine the direction d_k closest to $\alpha(x, y)$
2. Let K denote the value of $\|\nabla f\|$ at (x, y). If K is less than the value of $\|\nabla f\|$ at one or both of the neighbors of point (x, y) along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = K$.

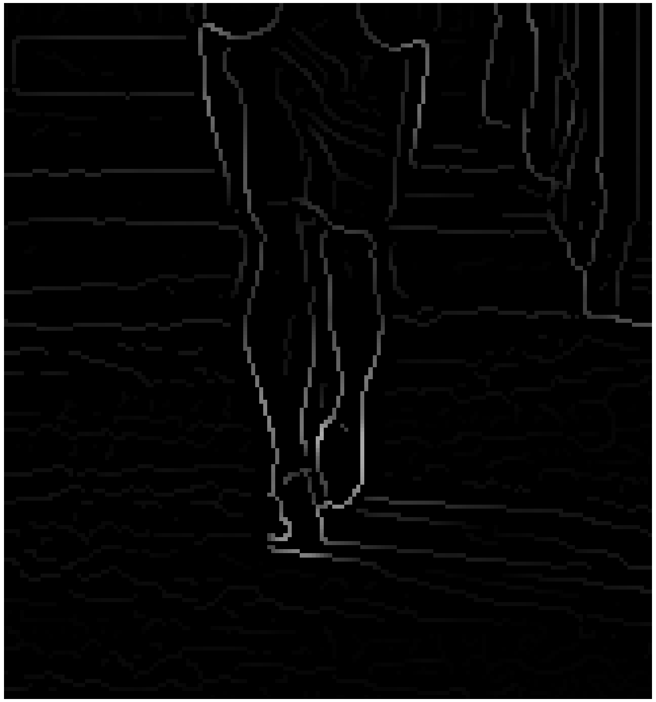


Every edge has two possible orientations

Before Non-max Suppression

Gradient magnitude (x4 for visualization)

After non-max suppression

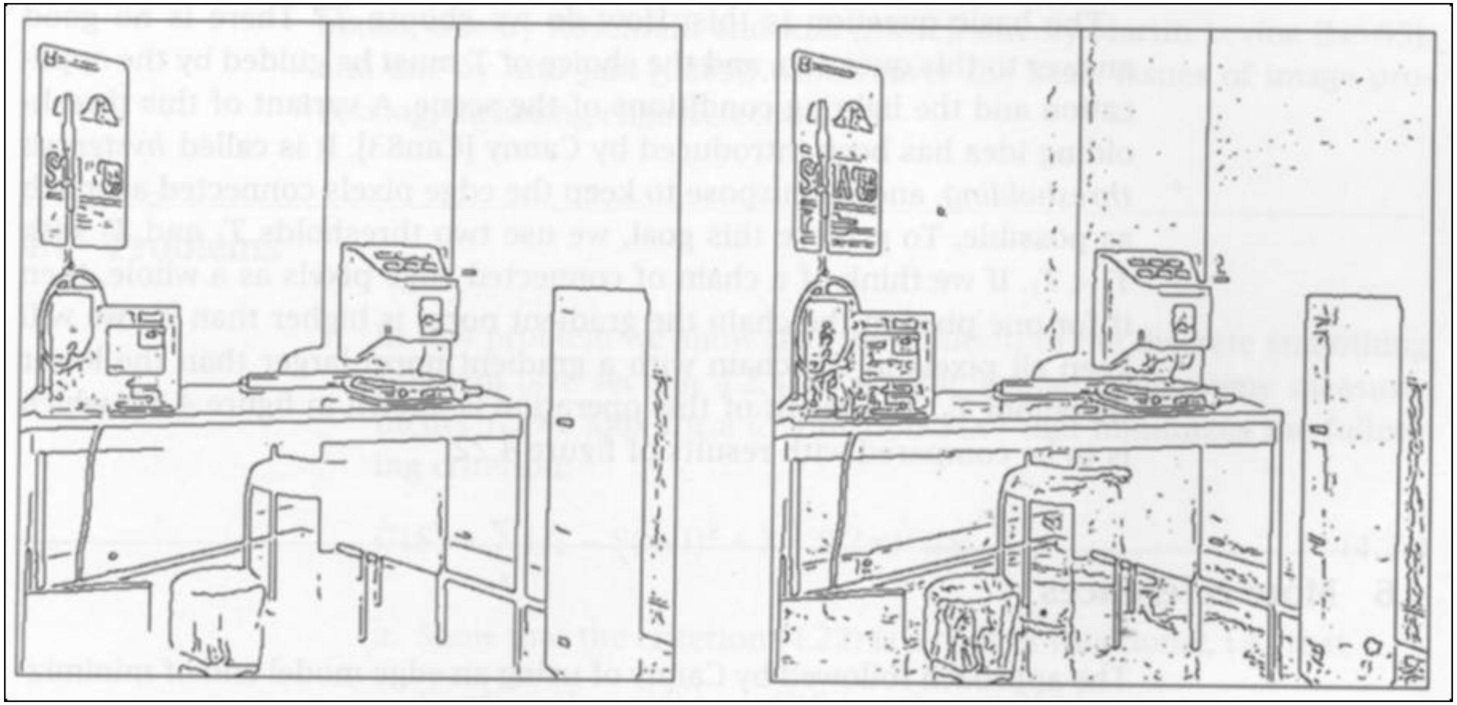


Gradient magnitude (x4 for visualization)

Input image

Single Threshold

T=15

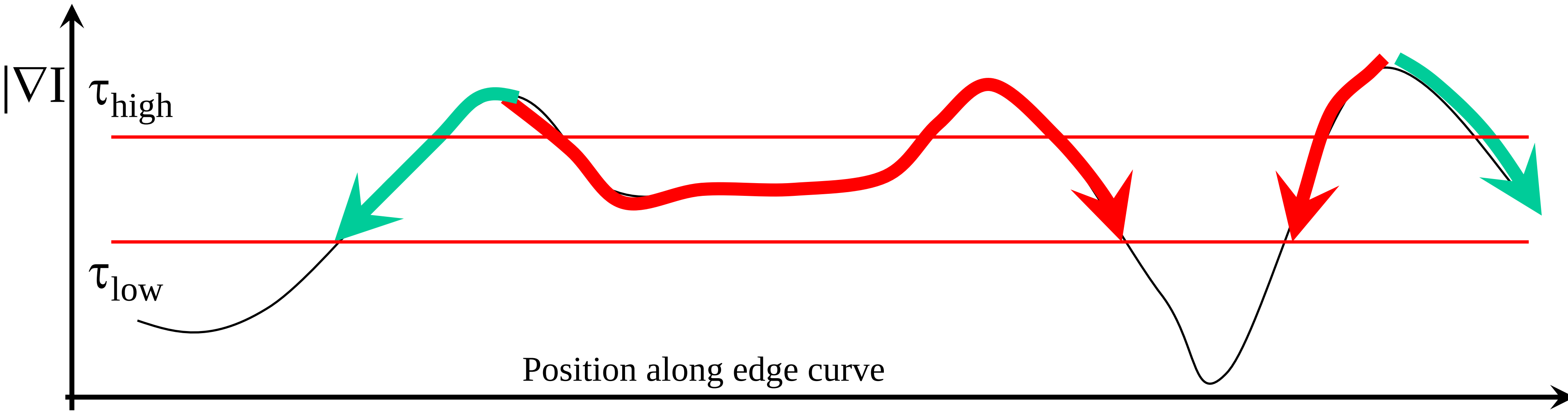


T=5

- When threshold is to high, important edges may be missed or be broken
- When threshold is too low, many extraneous edges, but non missed
- Hysteresis thresholding: Best of both

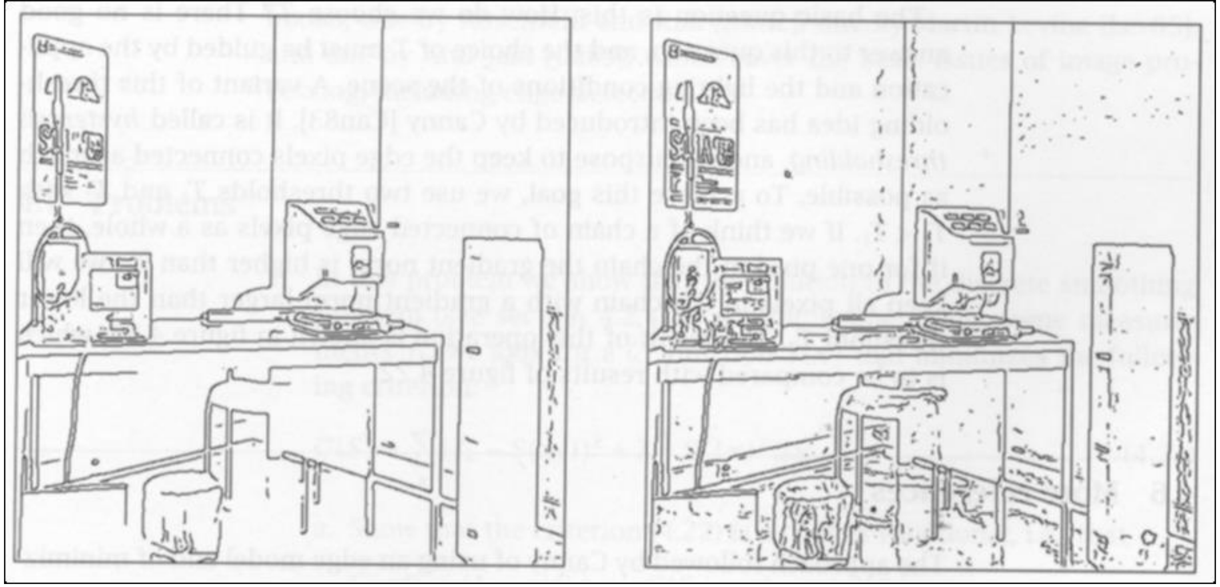
Hysteresis Thresholding

- Start tracking an edge chain at pixel location that is local maximum of gradient magnitude where gradient magnitude > τ_{high}.
- Follow edge in direction orthogonal to gradient.
- Stop when gradient magnitude < τ_{low}.
  - i.e., use a high threshold to start edge curves and a low threshold to continue them.

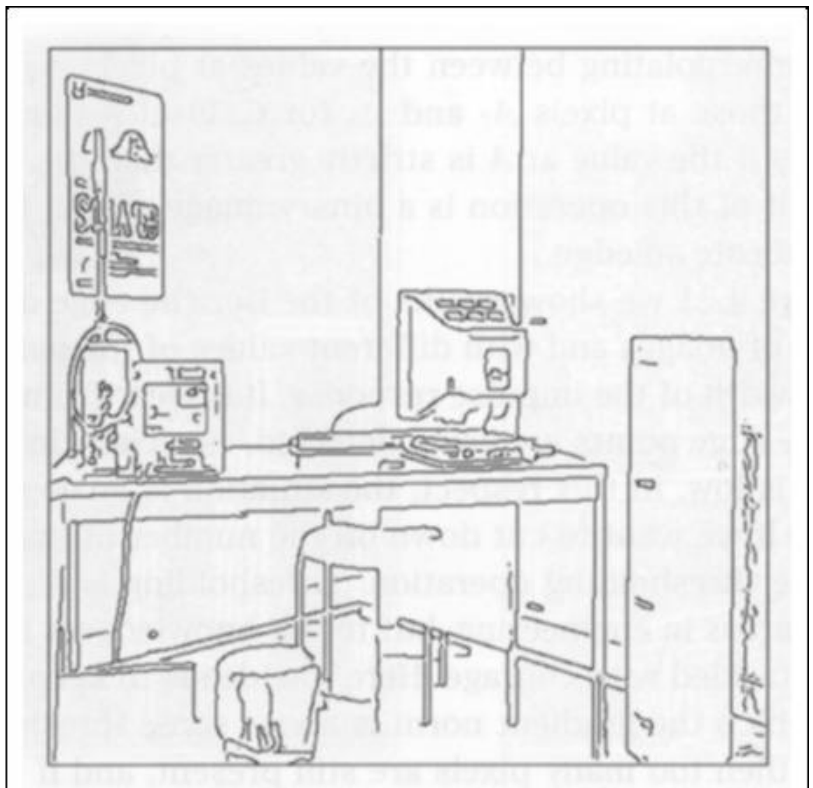


Single Threshold

T=15



T=5

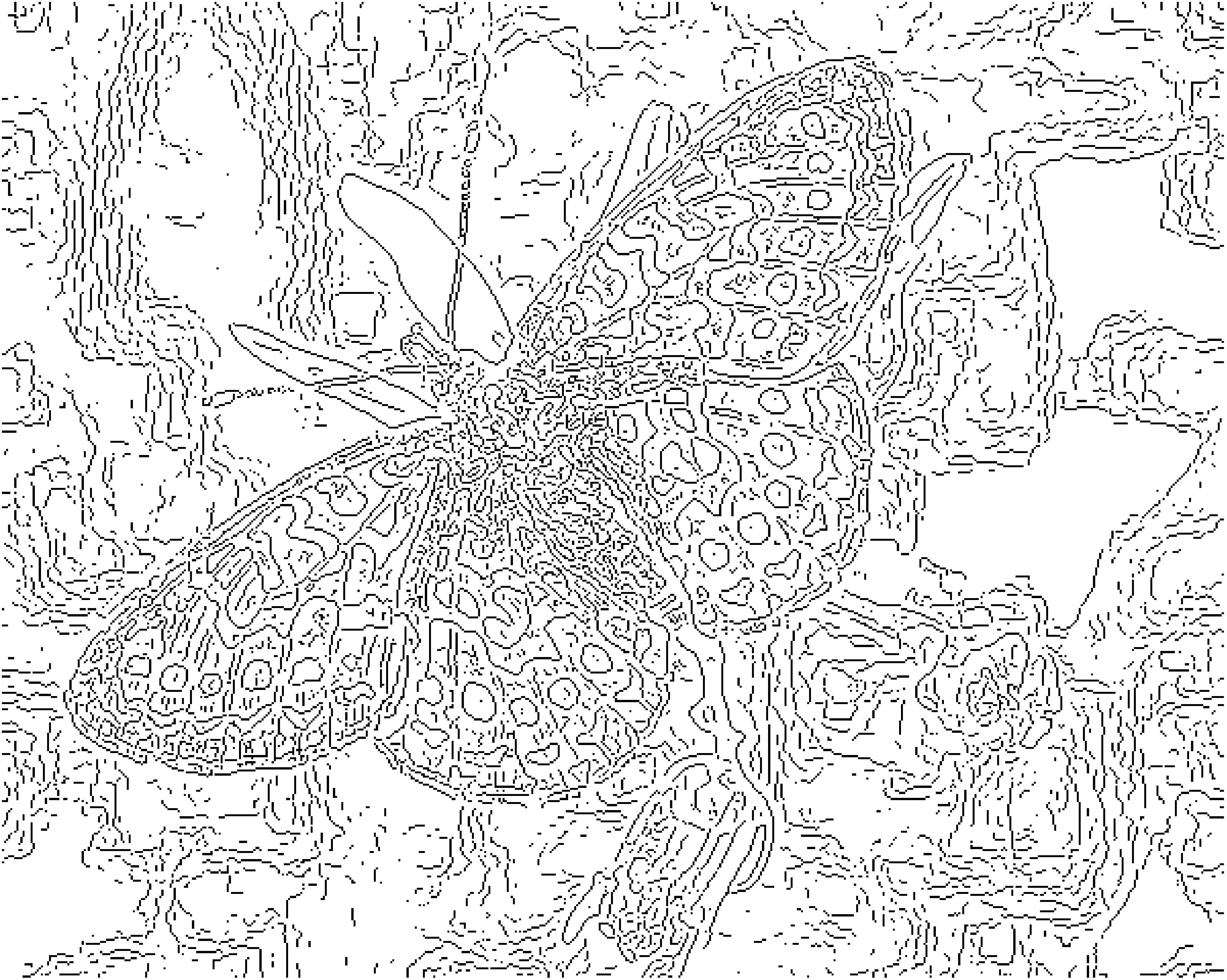


Hysteresis T_h=15 T_l = 5

Hysteresis thresholding

Canny Edge Detection Algorithm

1. Three parameters σ, τ_{high}, τ_{low}
2. Filter with symmetric Gaussian of width σ
3. Computer gradient, magnitude, direction
4. Non-maximal supression
5. Hysteresis thresholding using τ_{high}, τ_{low}



fine scale, high threshold

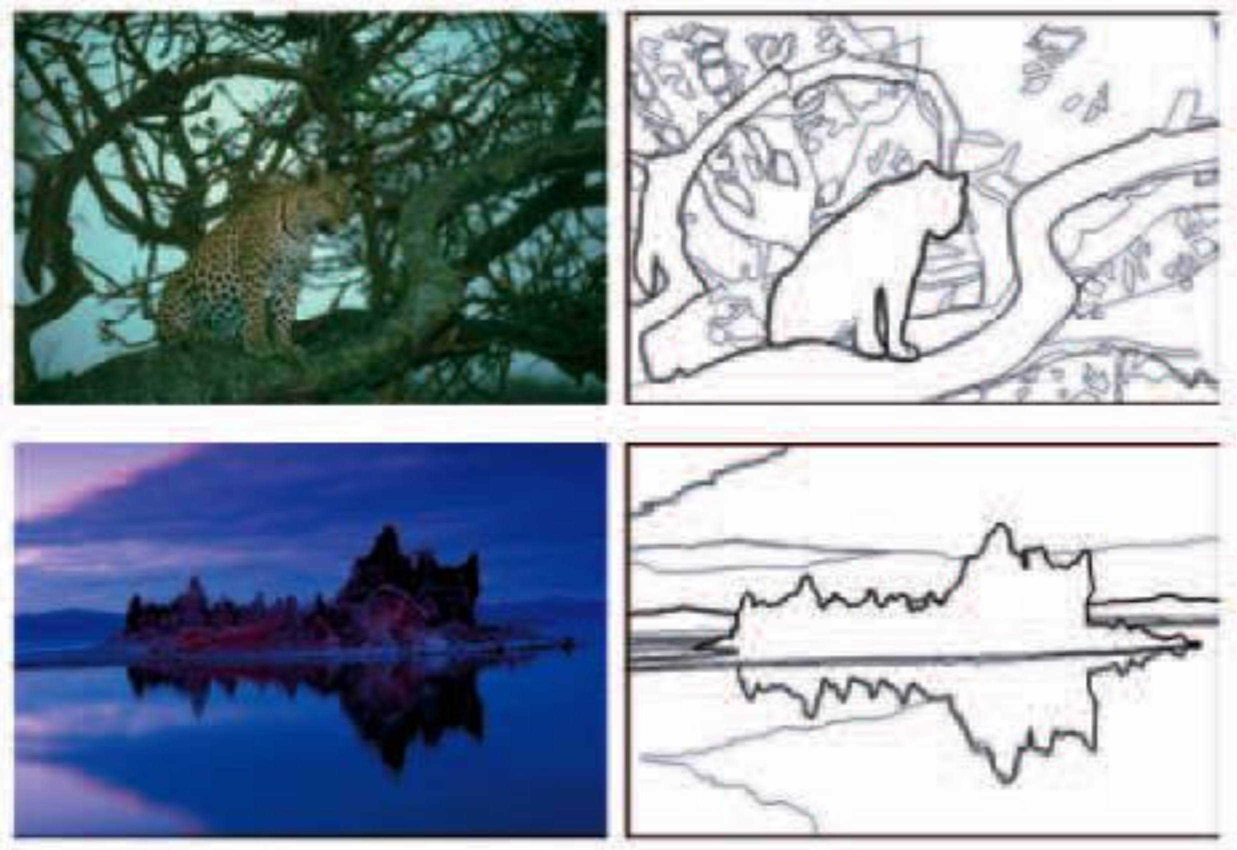
coarse
scale,
high high
threshold

coarse
scale,
low high
threshold

Why is Canny so Dominant

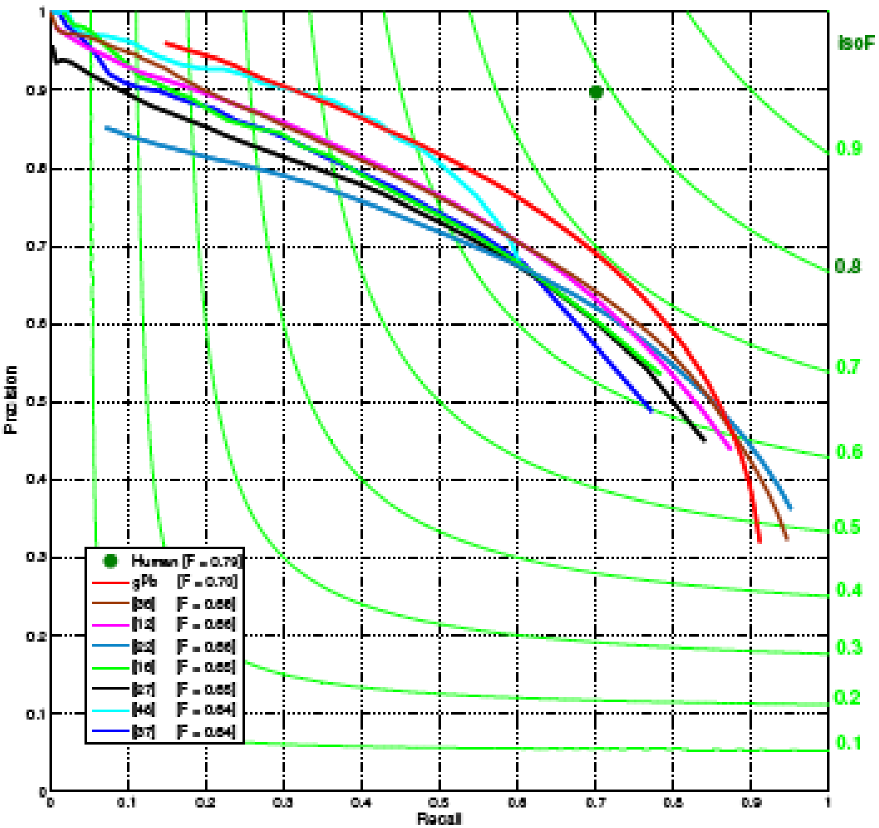
- Widely used for 30 years.
- Theory is nice
- Details are good
  - Magnitude of gradient,
  - Non-max supression
  - Hysteresis thresholding
- Most subsequent detectors weren't much better until learning-based detectors came along
- Code was distributed

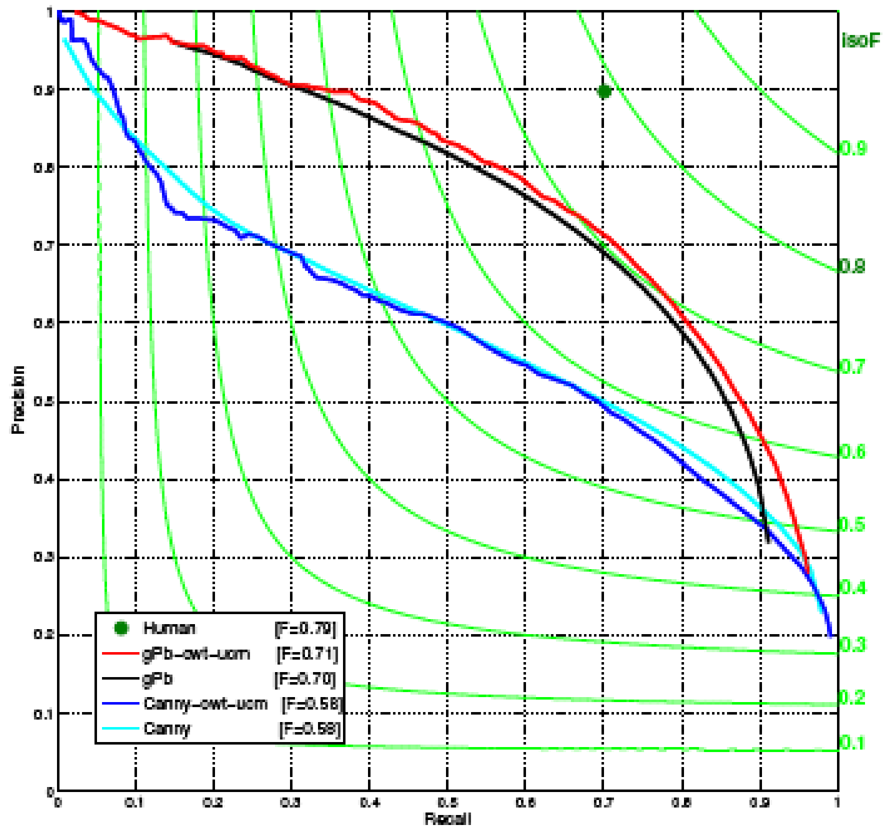
Learning-based detectors: Not edges, but boundaries



- **Brightness**
- **Color**
- **Texture**
- **Subjective contours**
- **Grouping**
- **Multiscale**

Boundary detection





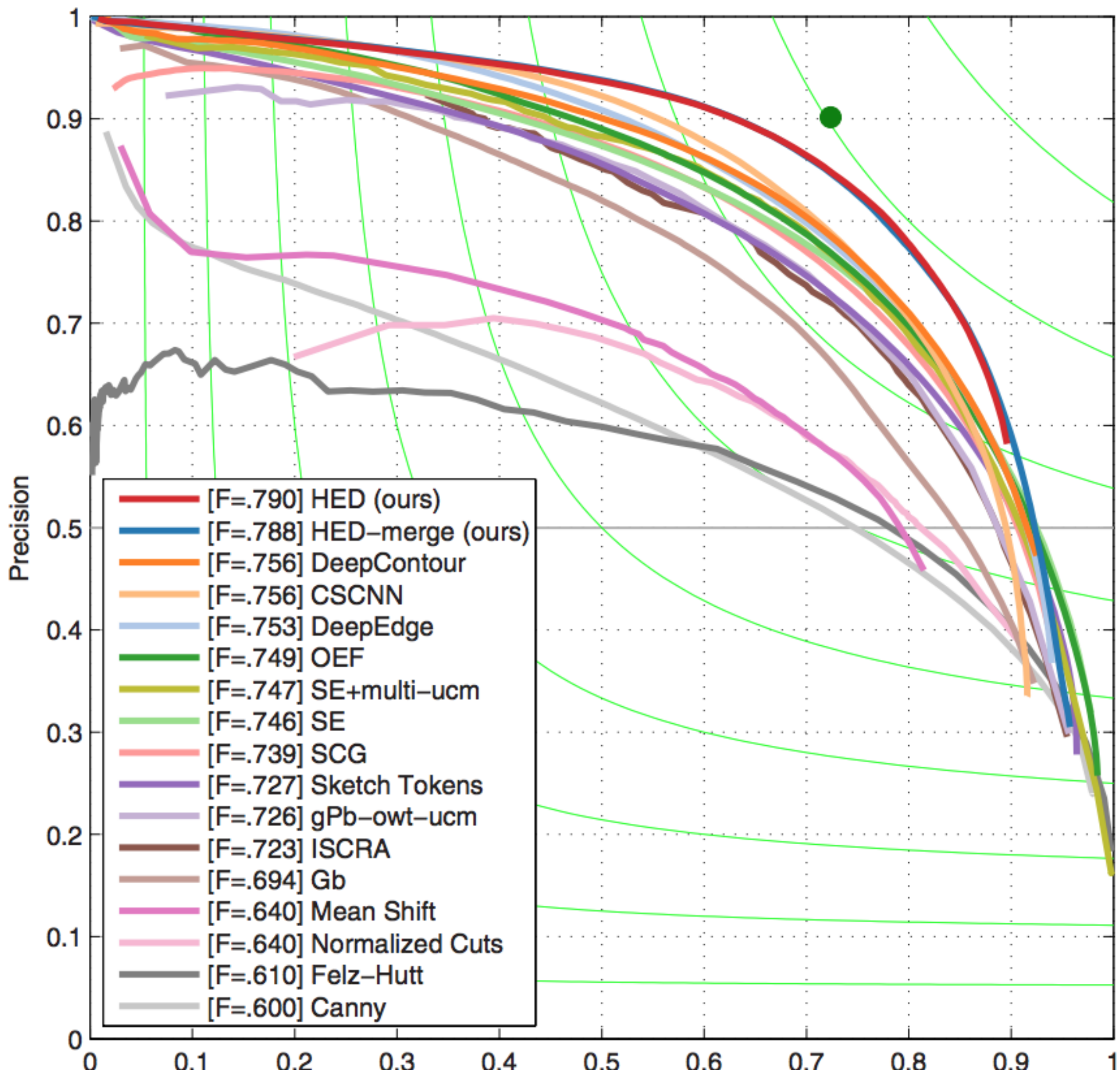
- Precision is the fraction of detections that are true positives rather than false positives, while recall is the fraction of true positives that are detected rather than missed.

- From Contours to Regions: An Empirical Evaluation, Arbelaez, M. Maire, C. Fowlkes, and J. Malik, CVPR 2008

Learned Edge Detectors

- Dollar, Piotr, Zhuowen Tu, and Serge Belongie. "Supervised learning of edges and object boundaries." Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Vol. 2. IEEE, 2006
- Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Proceedings of the IEEE International Conference on Computer Vision. 2013.
- Xie, Saining, and Zhuowen Tu. ." Proceedings of the IEEE international conference on computer vision. 2015.
- Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015

HED Performance



Xie and Tu. "Holistically-nested edge detection." ICCV 2015

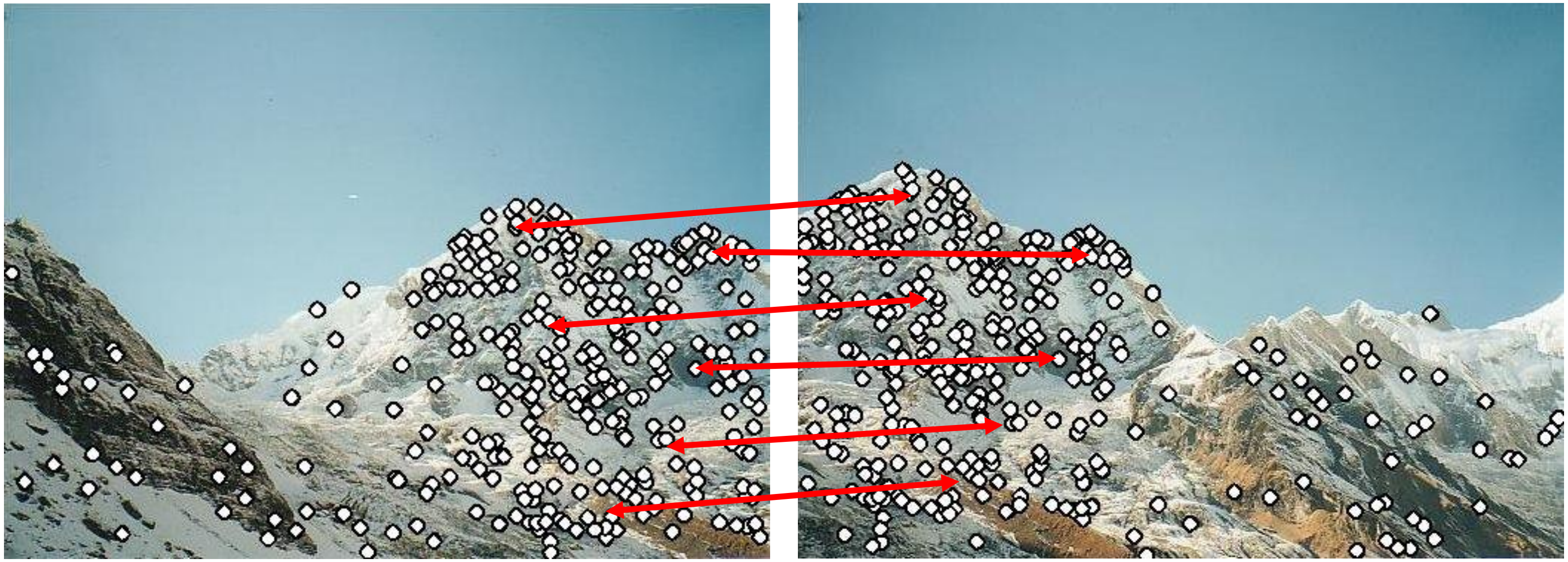
Corner Detection

Motivation: feature matching

- Panorama stitching
  - We have two images – how do we combine them?

Motivation: feature matching

- Panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features

Step 2: match features

Motivation: feature matching

- Panorama stitching
  - We have two images – how do we combine them?

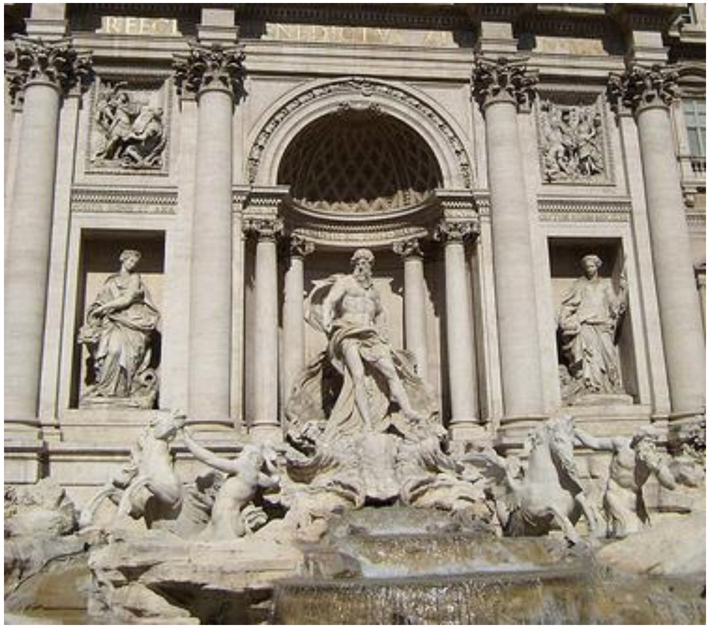
Step 1: Extract features in each image

Step 2: Match features across images

Step 3: Align images and determine a transformation

Image matching

by Diva Sian



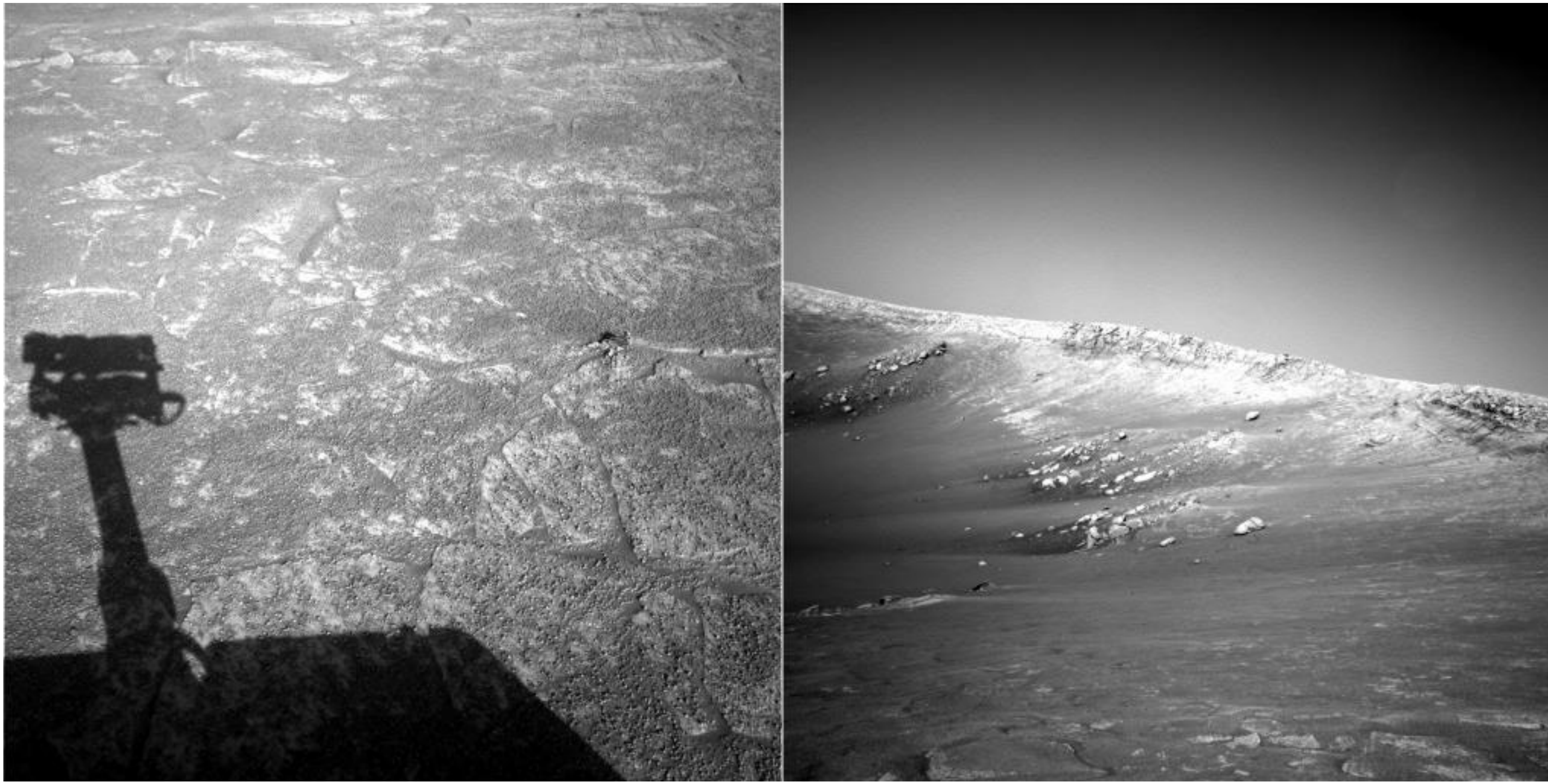
by swashford

Harder case

by Diva Sian

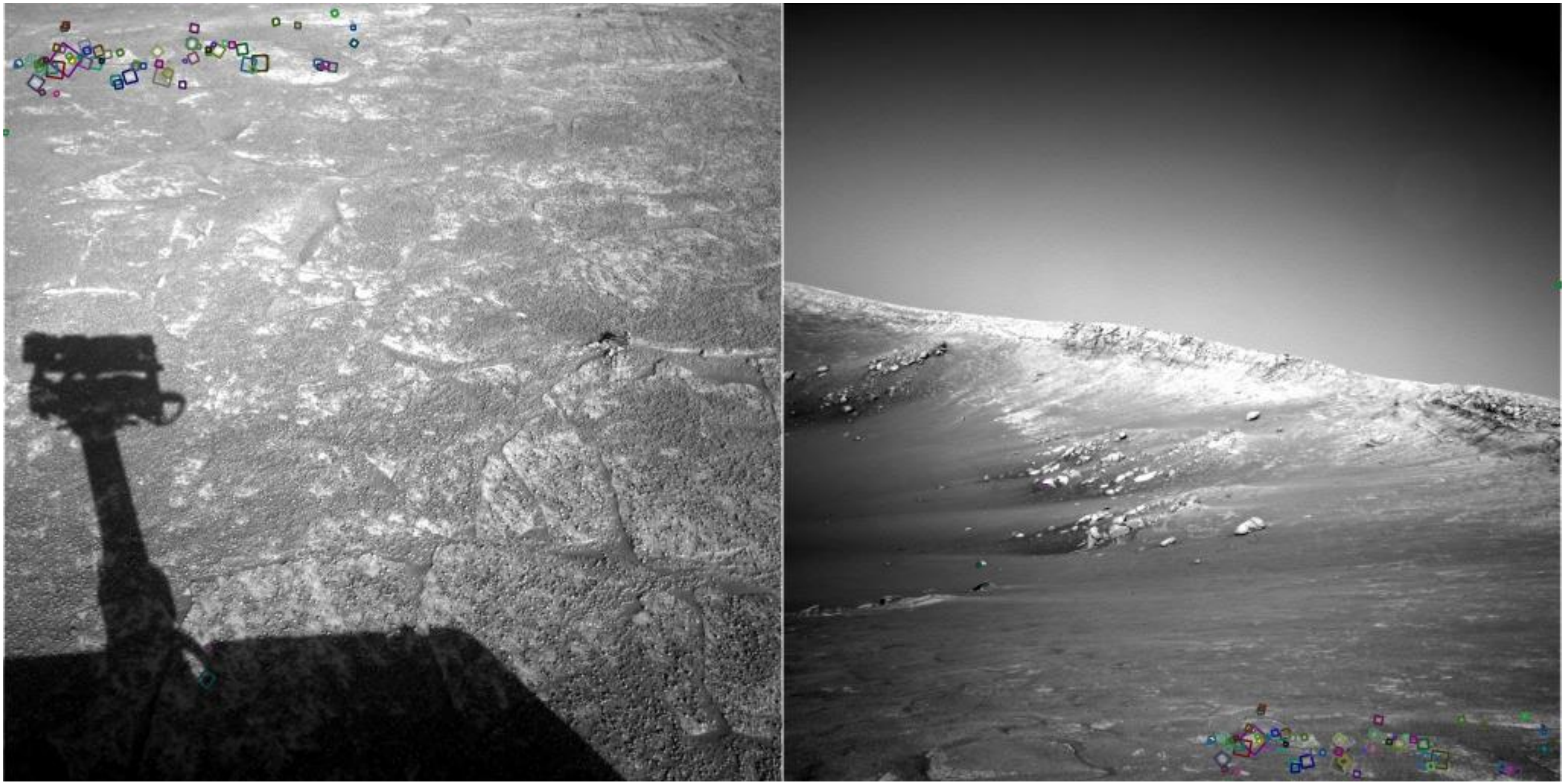
by scgbt

Harder still?



NASA Mars Rover images

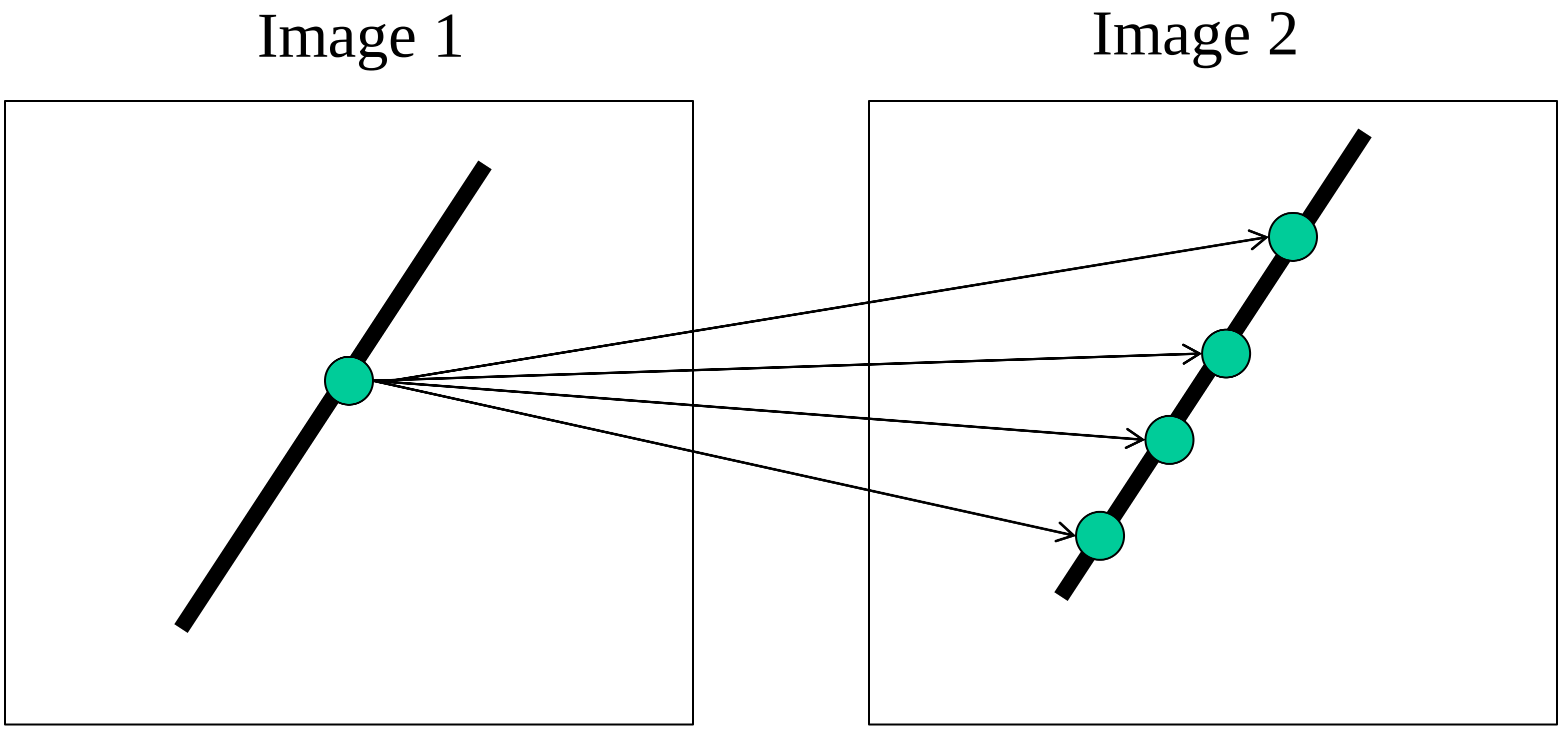
Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

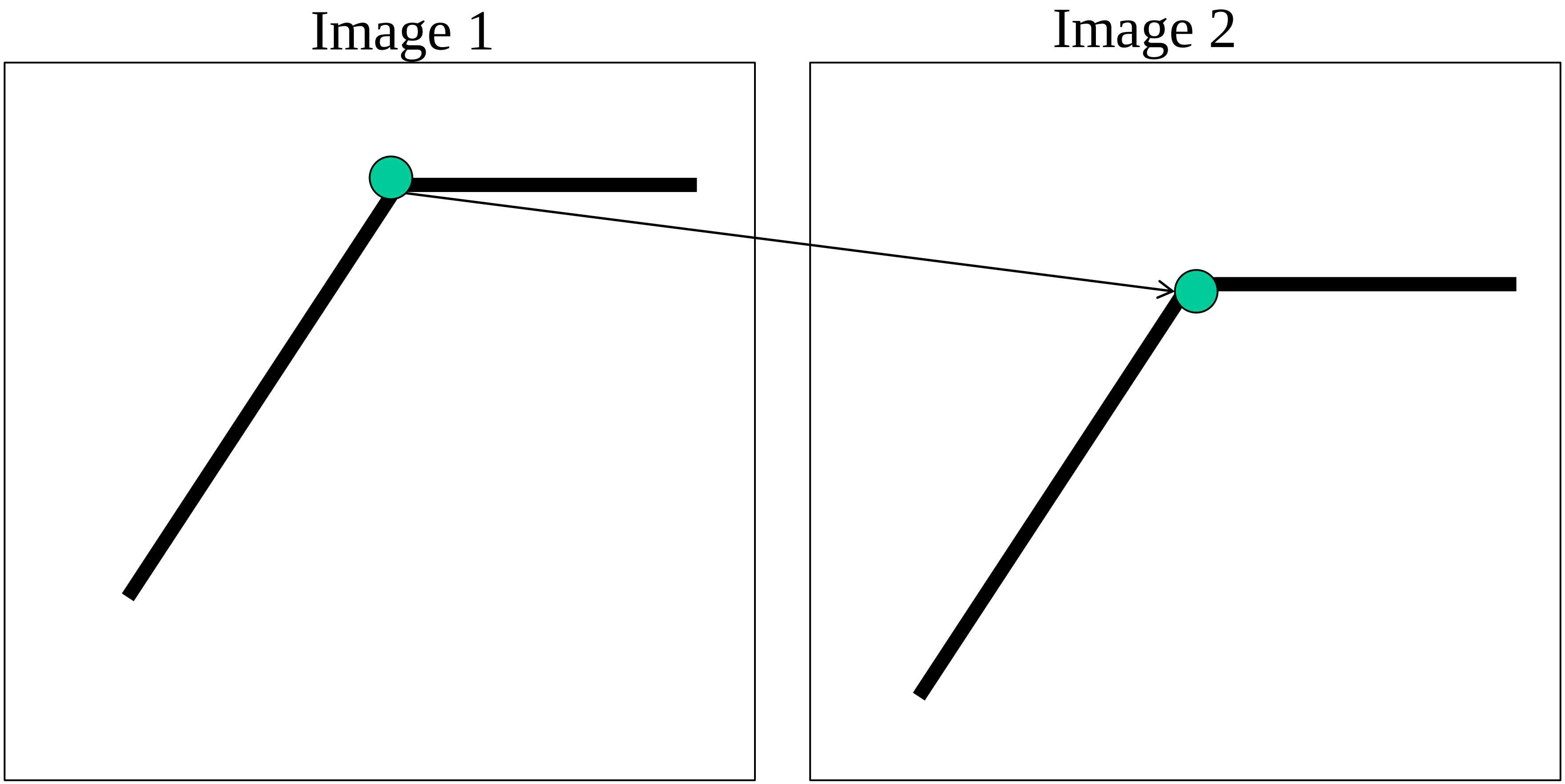
Corners contain more info than lines

- A point on a line is hard to match



Corners contain more info than lines

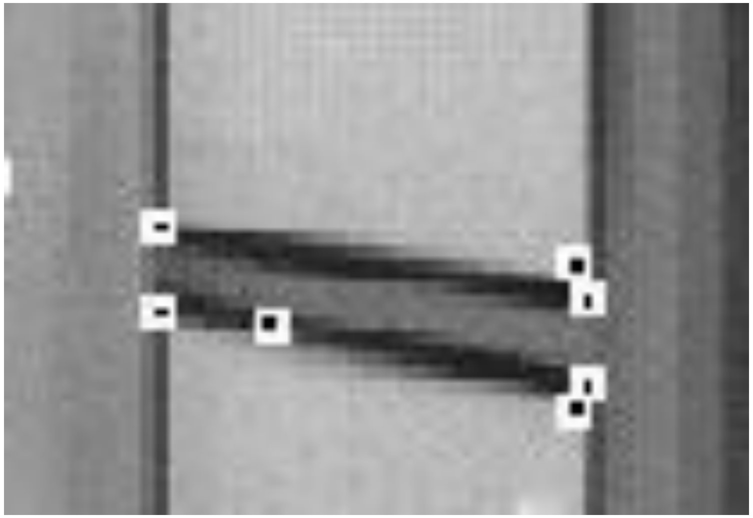
- A corner is easier to match



Detection of corner-like features

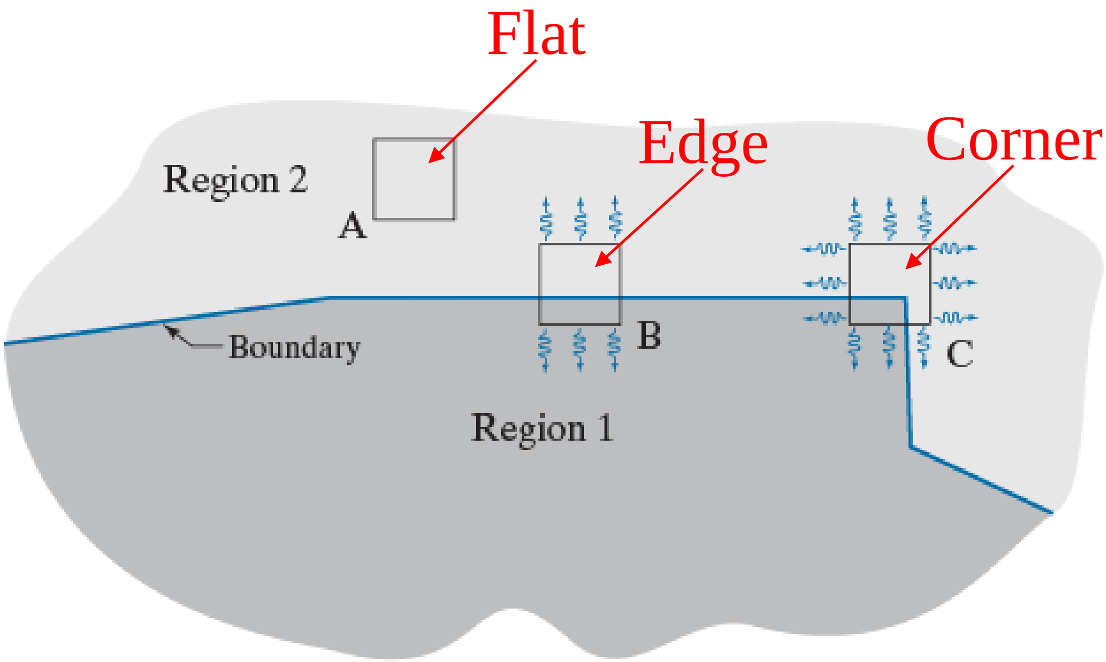
- Corner
  - A rapid change of direction in a curve
  - A highly effective feature
    - Distinctive
    - Reasonably invariant to viewpoint

Corners



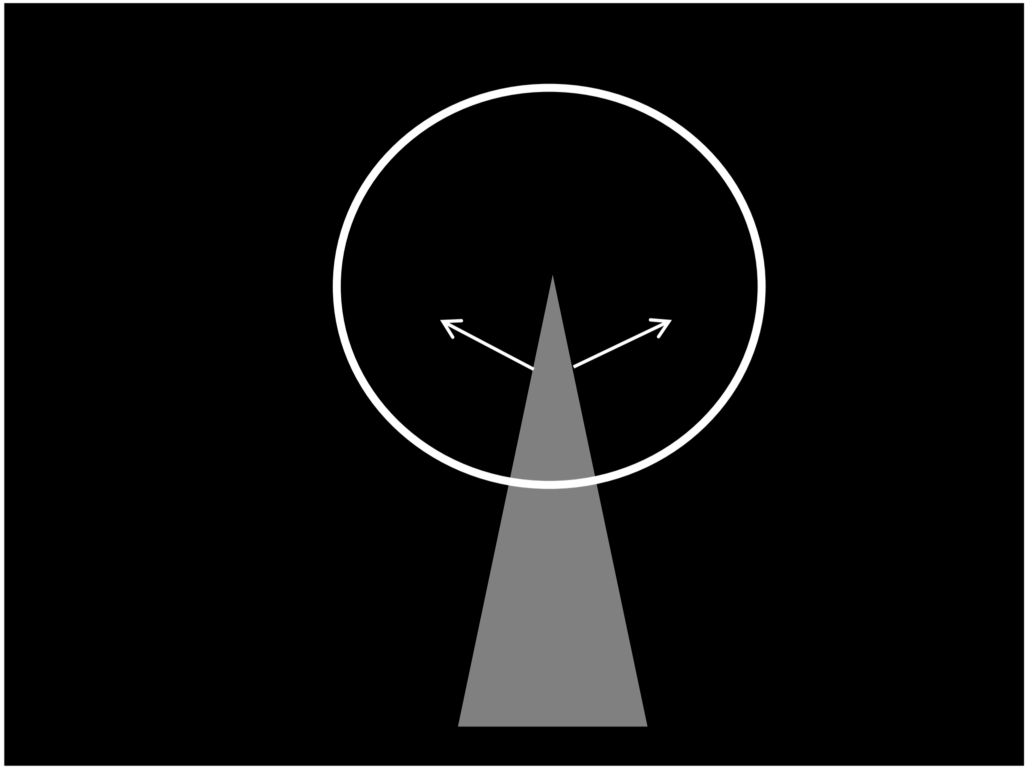
Detection of corner-like features

- Examine a small window over an image



The wiggly arrows indicate graphically a directional response in the detector as it moves in the three areas shown

Detection of corner-like features



Intuition:

- Right at corner, gradient is ill-defined.
- Near corner, gradient has two different values.

Detection of corner-like features

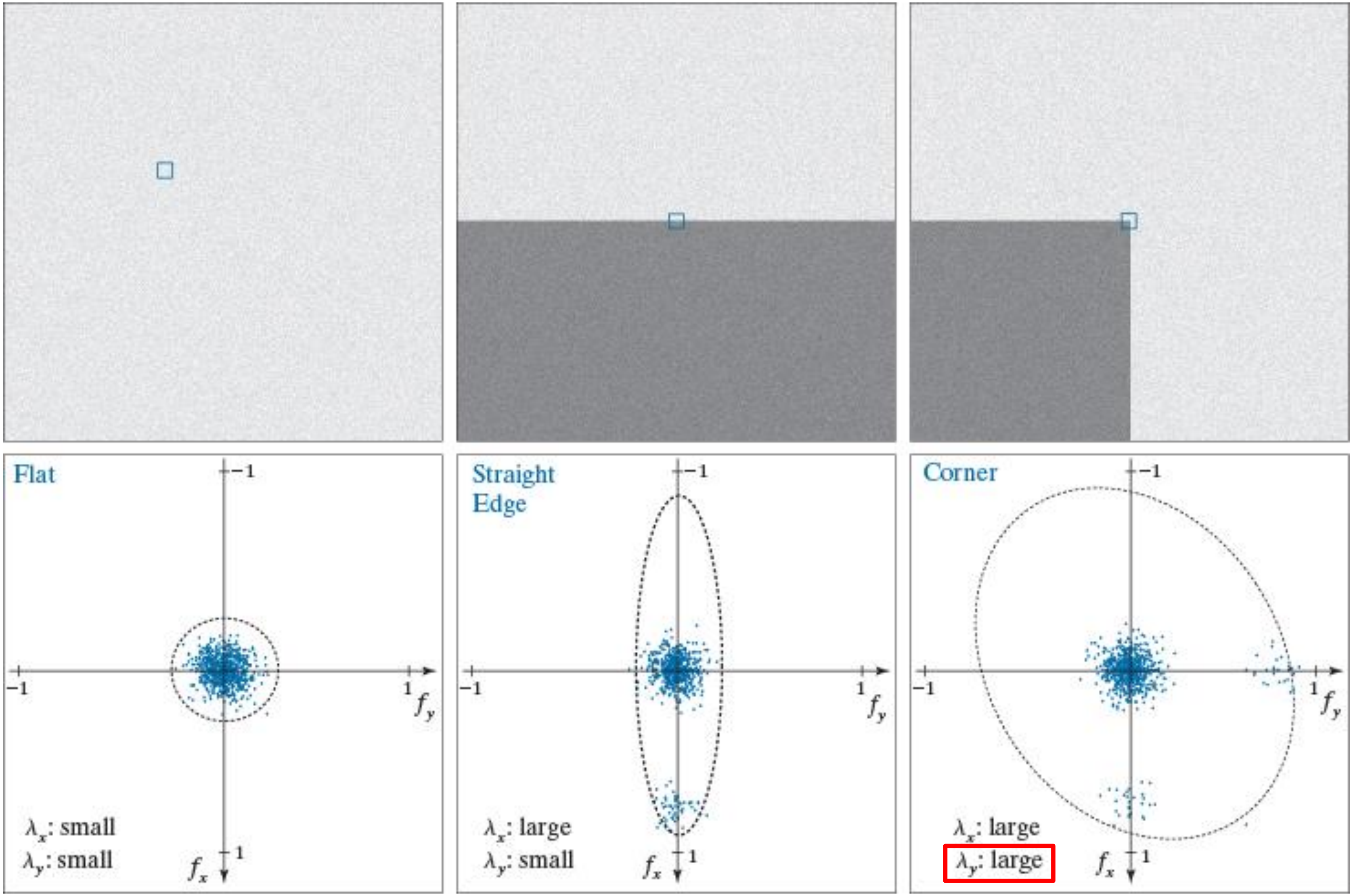
- For each window location, compute the spatial gradient matrix

$$\mathtt{M} = \begin{bmatrix} \sum_s \sum_t f_x^2(s,t) & \sum_s \sum_t f_x(s,t) f_y(s,t) \\ \sum_s \sum_t f_x(s,t) f_y(s,t) & \sum_s \sum_t f_y^2(s,t) \end{bmatrix}$$

  where f_x is the gradient in the *x*-direction and f_y is the gradient in the *y*-direction

- Then, compute eigenvalues of spatial gradient matrix

Eigenvalues of spatial gradient matrix



Detection of corner-like features

When computing gradients, use shape valid filtering followed by zero padding such that output image is same size as input image

- Shi-Tomasi corner detector
  - Run a small window over an image and compute spatial gradient matrix **M**
  - Compute the minor eigenvalue of **M** to compute measurement image *R*

$$\lambda_{\min} = \frac{\mathrm{Tr}(\mathtt{M}) - \sqrt{\mathrm{Tr}(\mathtt{M})^2 - 4\det(\mathtt{M})}}{2}$$

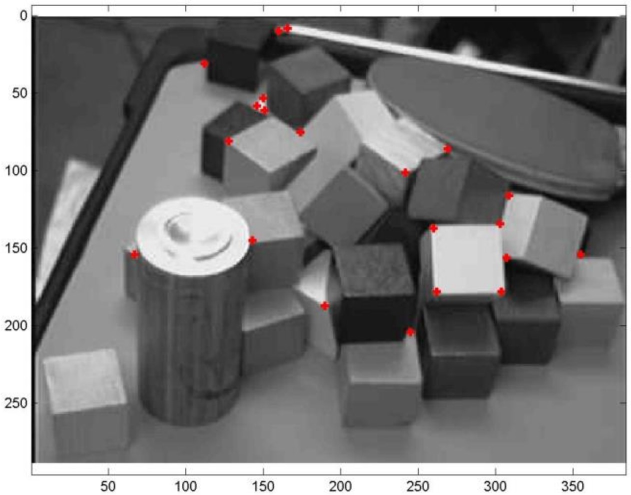
To mitigate floating-point numerical precision and accuracy issues, replace with

$$\sqrt{\max(0, \mathrm{Tr}(\mathtt{M})^2 - 4\det(\mathtt{M}))}$$

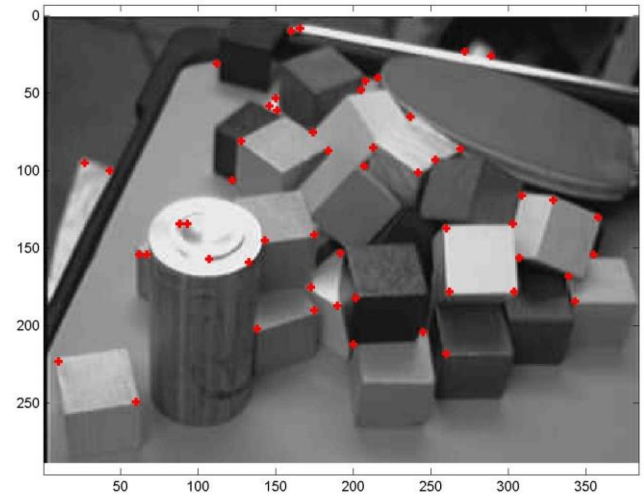
  - Apply nonmaximal suppression to the measurement image *R*
    - Prevents corners from being too close to each other
  - Threshold resulting image *R* using global threshold *T*
    - Corner at coordinates corresponding to *R* > *T*

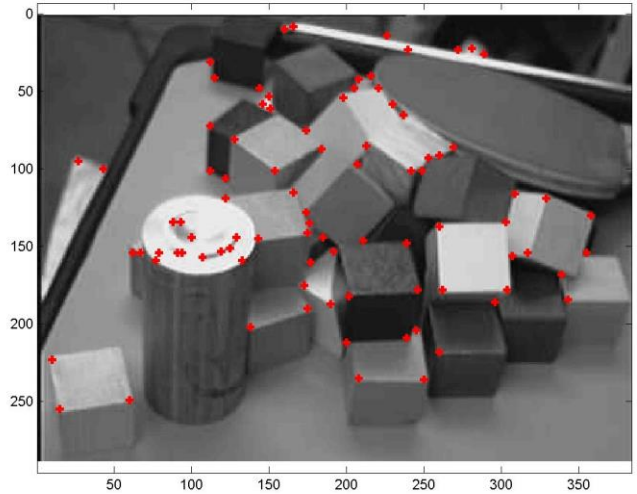
Corner Detection Sample Results

Threshold=25,000



Threshold=10,000





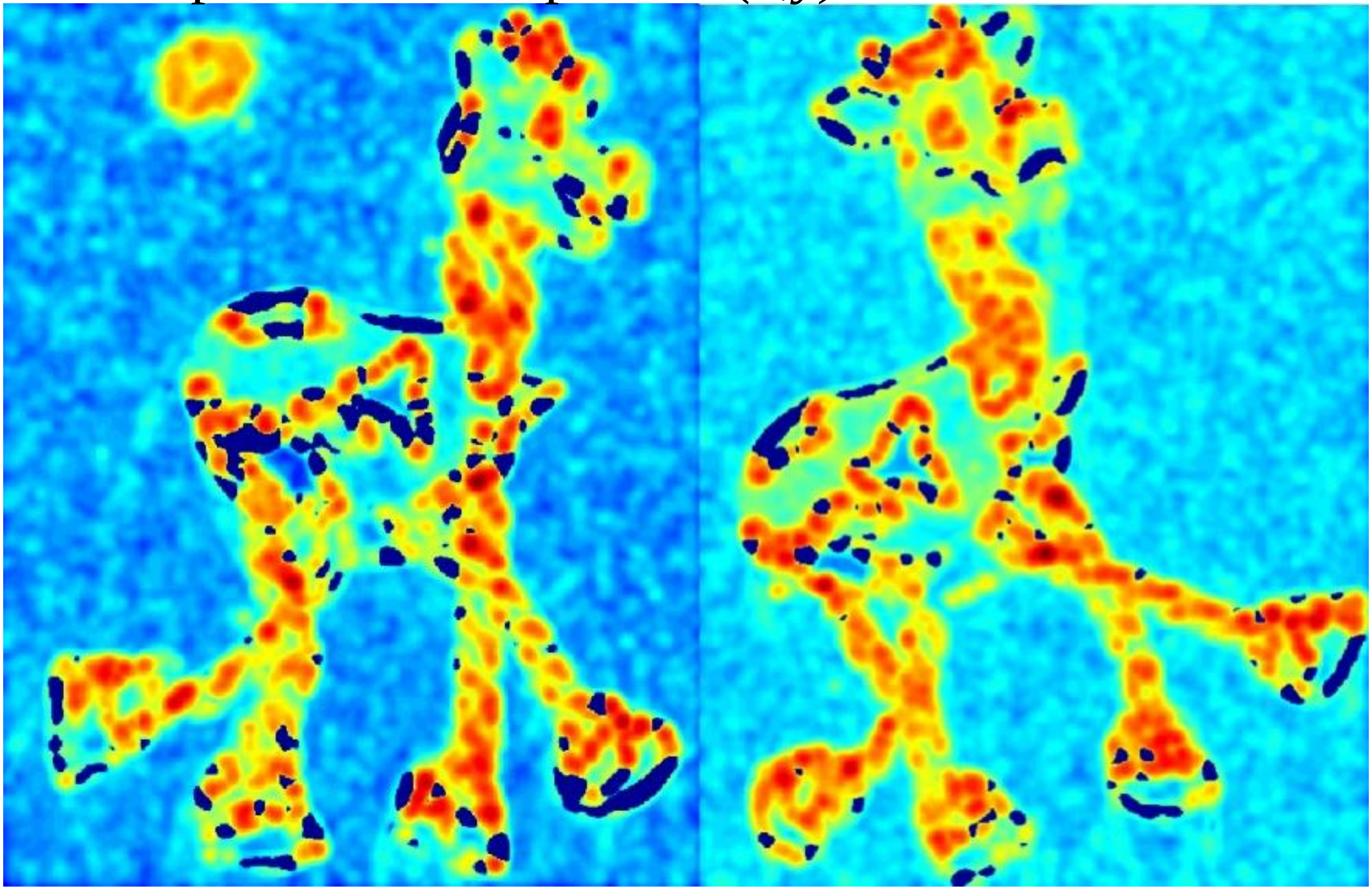
Threshold=5,000

Corner Detector: Workflow

Slide credit: http://vims.cis.udel.edu/~chandra/

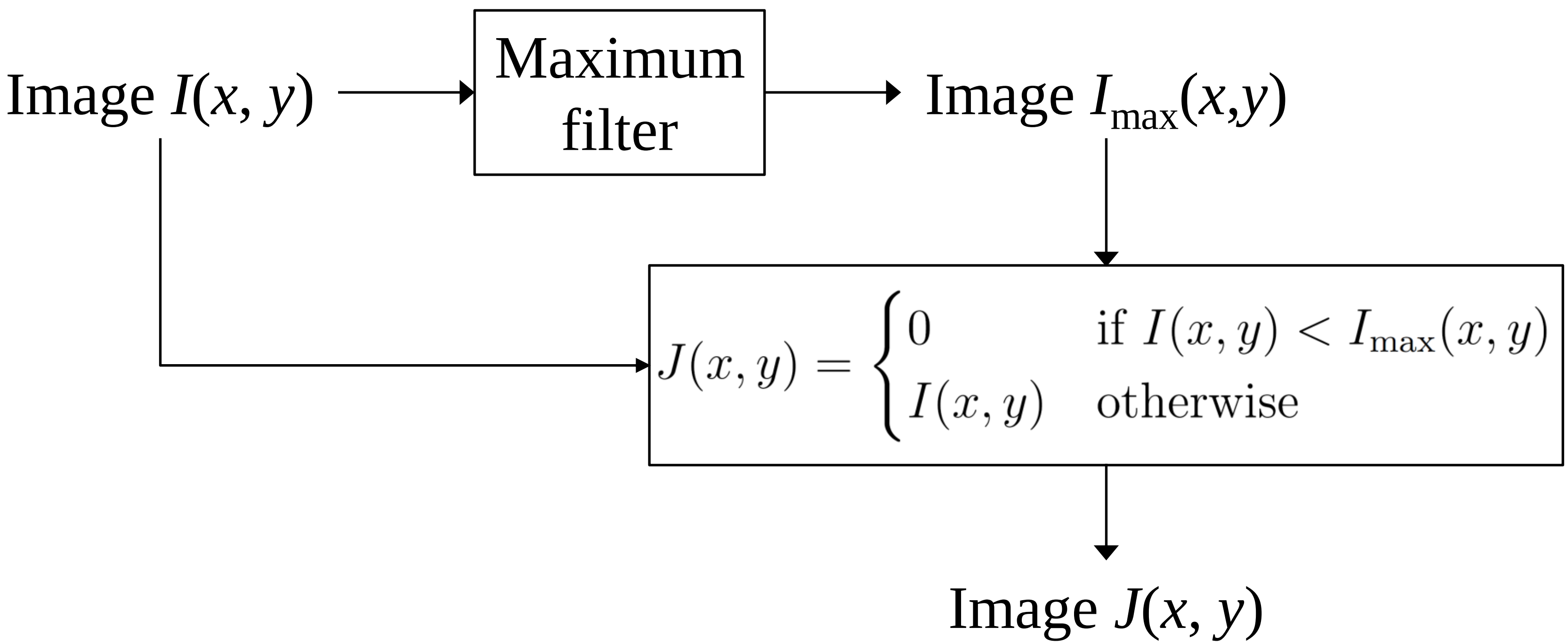
Corner Detector: Workflow

Compute corner response $R(x,y)$



Slide credit: http://vims.cis.udel.edu/~chandra/

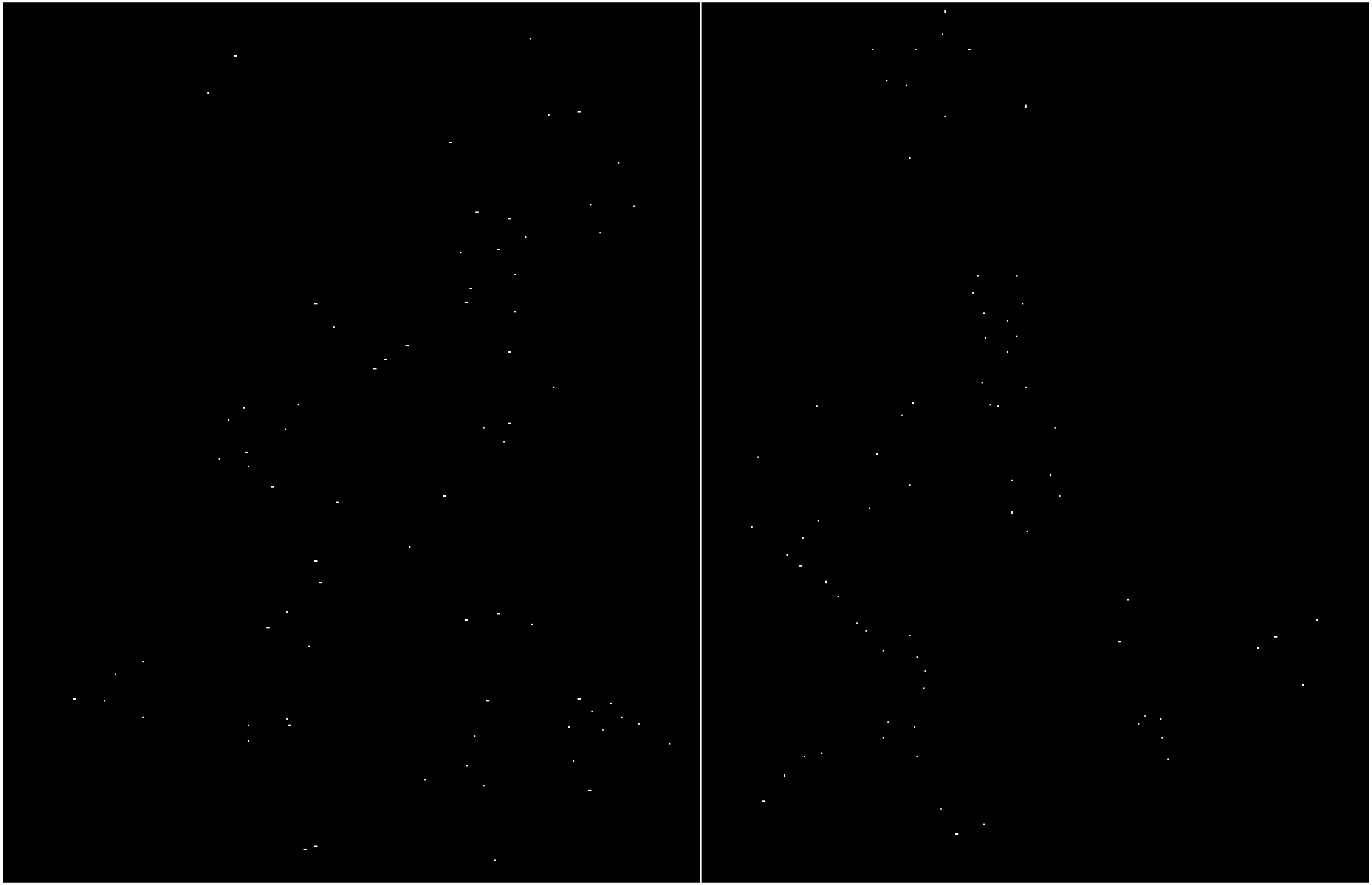
Nonmaximum suppression



- Then, find coordinates of pixels in image $J(x,y)$ with nonzero values

Corner Detector: Workflow

Take only the points of local maxima of $R(x,y)$ and threshold



Slide credit: http://vims.cis.udel.edu/~chandra/

Corner Detector: Workflow

Slide credit: http://vims.cis.udel.edu/~chandra/

Next Lecture

- Calibrated stereo