Edge and corner detection

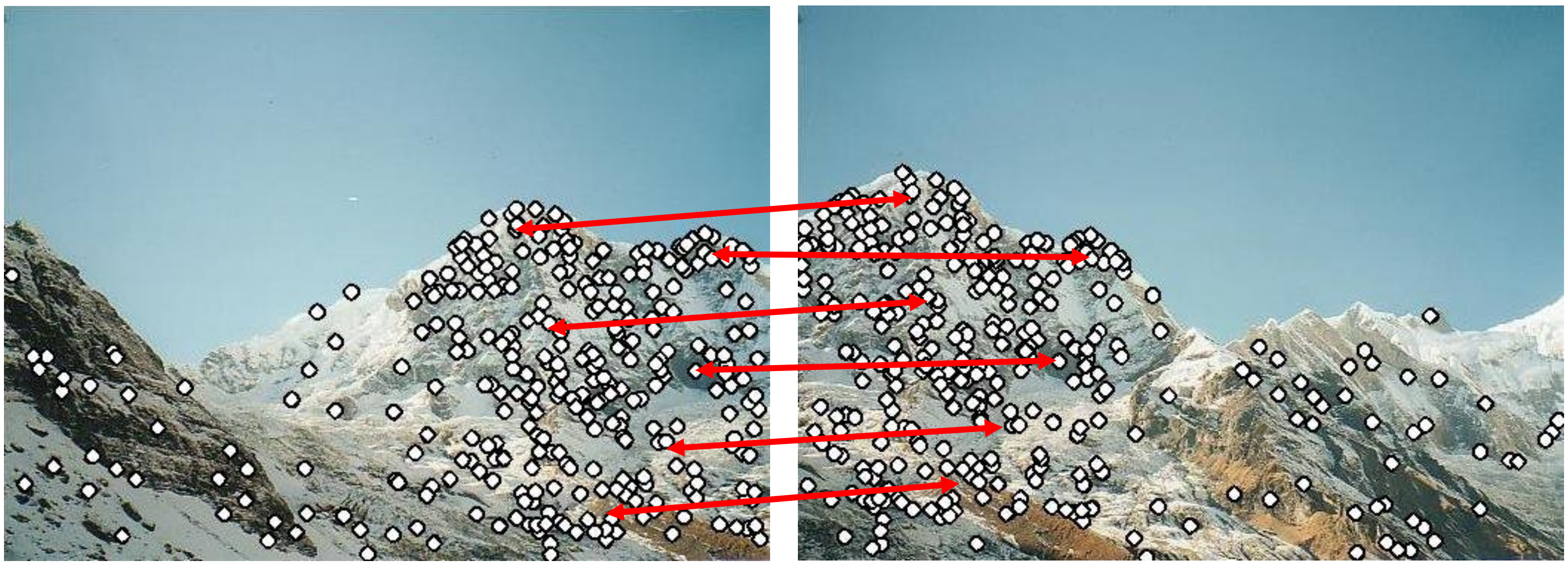
Book: Szeliski 4.1.1, 4.2, Forsyth 5.1, 5.2, 5.3

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Why extract features?

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features

Step 2: match features

Why extract features?

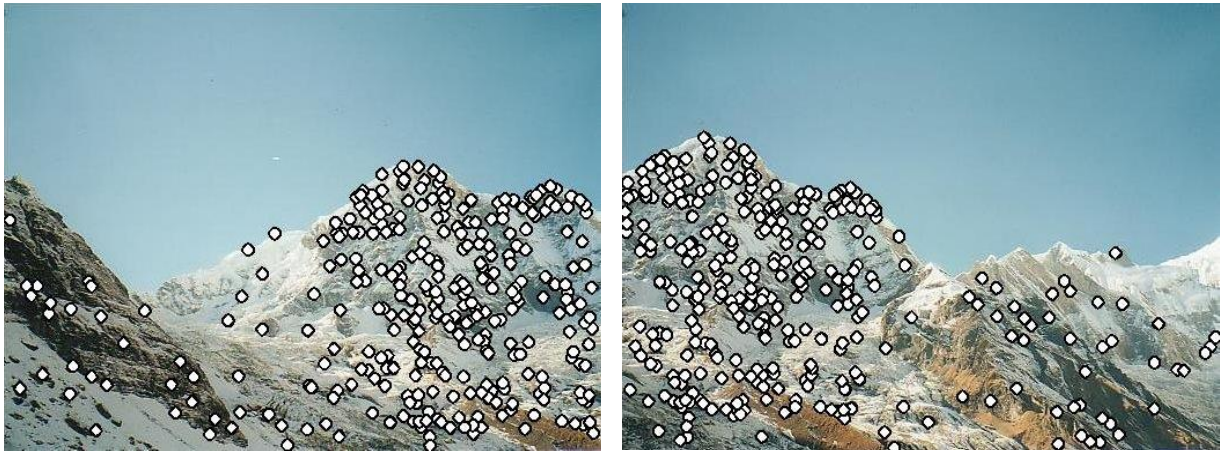
- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract features

Step 2: match features

Step 3: align images

Characteristics of good features



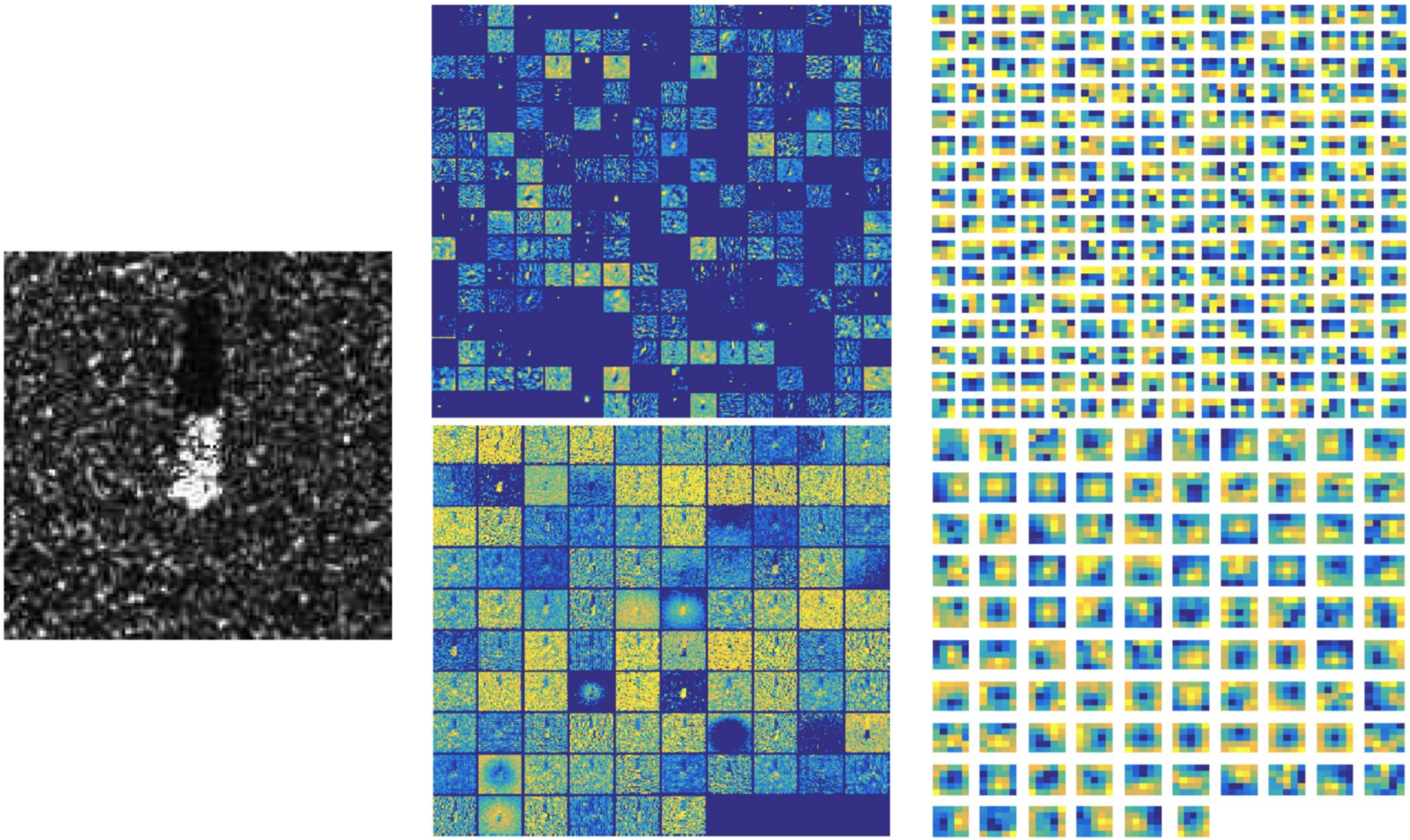
- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature is distinctive
- Compactness and efficiency
  - Many fewer features than image pixels
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

Feature points are used for:

- Image alignment
- 3D reconstruction
- Motion tracking
- Robot navigation
- Indexing and database retrieval
- Object recognition

Applications - currently



Source: [Deep feature extraction and combination for synthetic aperture radar target classification]()

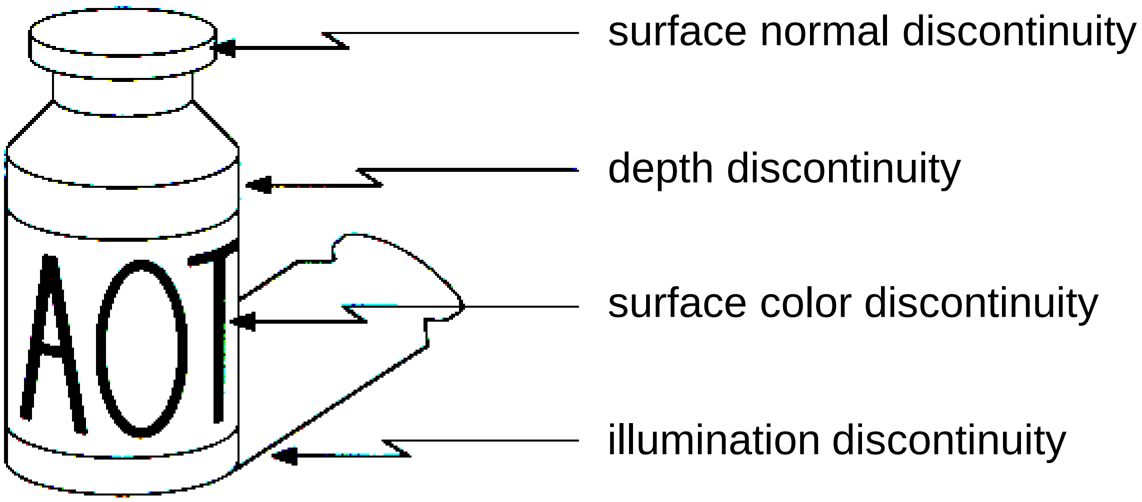
Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

Origin of edges

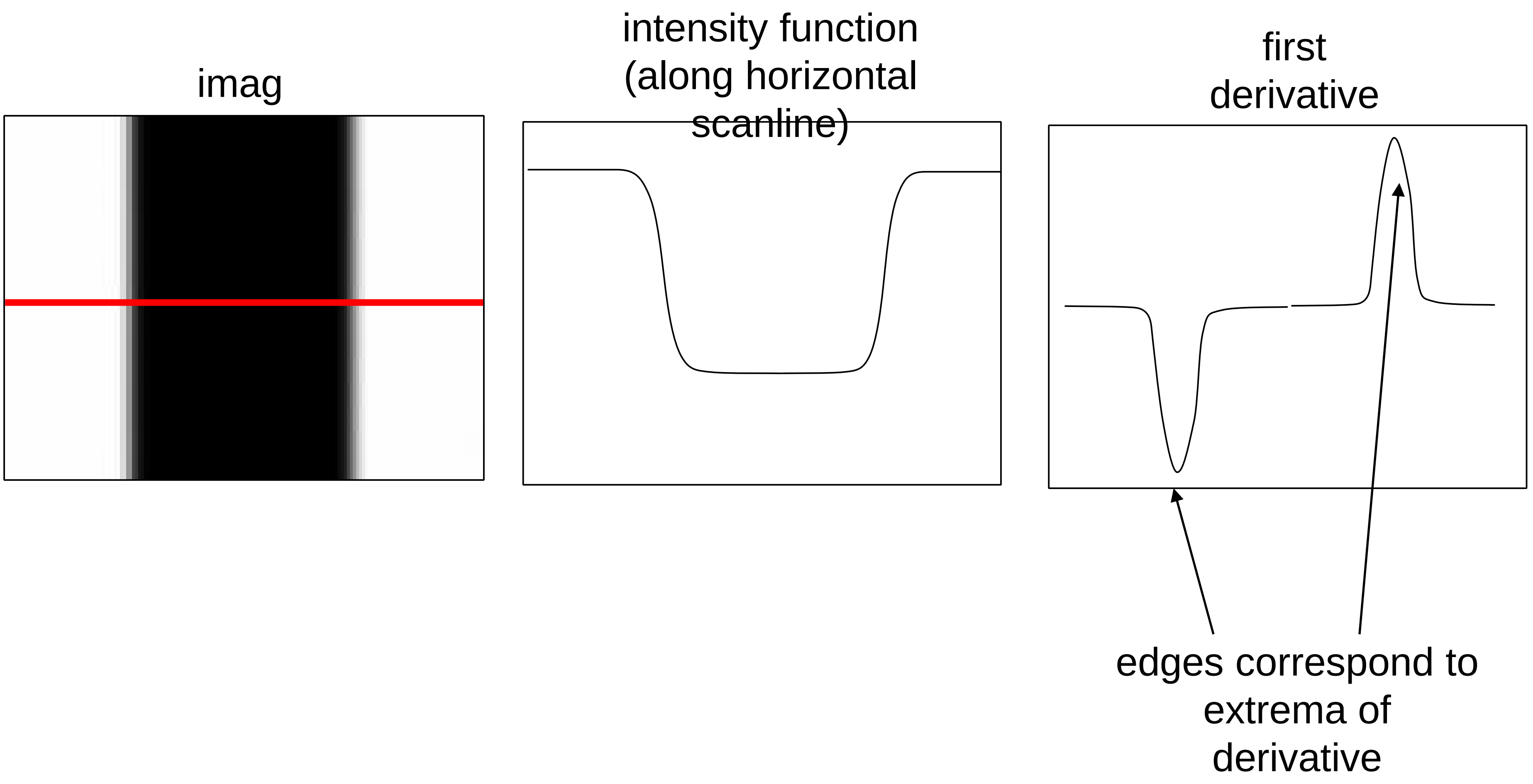
Edges are caused by a variety of factors:



Source: Steve Seitz

Edge detection

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



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:

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

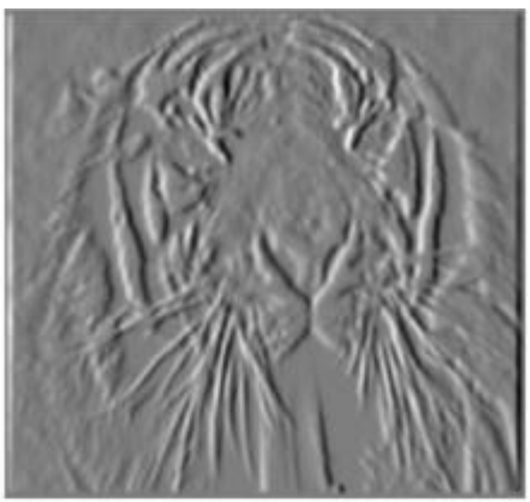
To implement the above as convolution, what would be the associated filter?

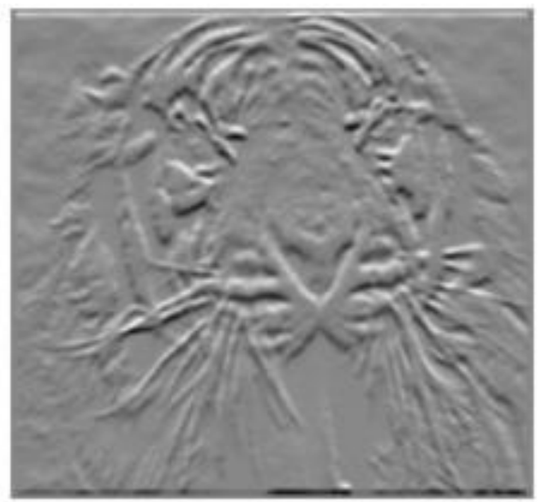
Source: K. Grauman

Partial derivatives of an image

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

-1	**1**





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

-1
1

Finite difference filters

Other approximations of derivative filters exist:

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

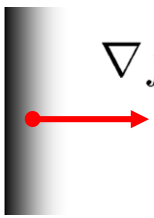
Sobel: $M_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

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

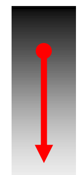
Source: K. Grauman

Image gradient

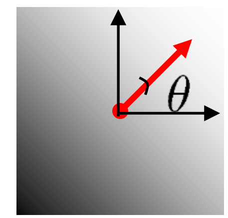
The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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



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



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

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

- How does this direction relate to the direction of the edge?

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

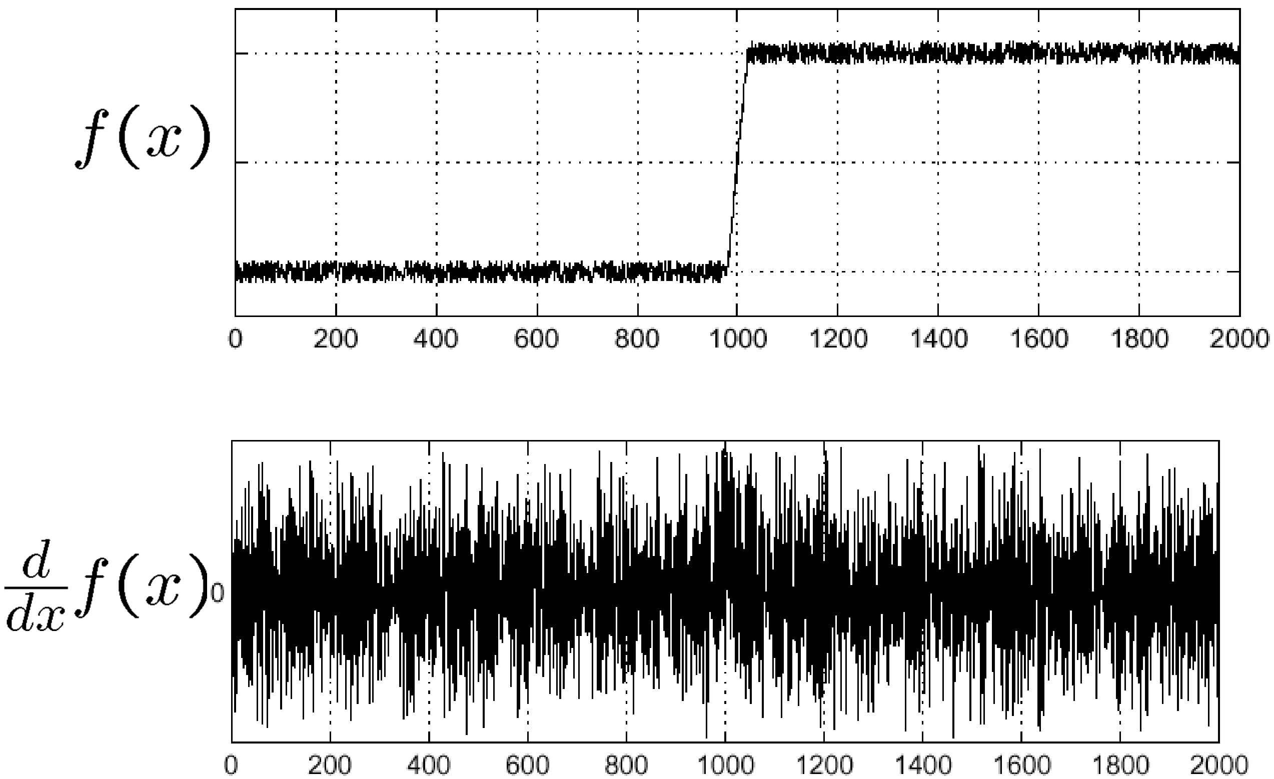
The edge strength is given by the gradient magnitude

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

Source: Steve Seitz

Effects of noise

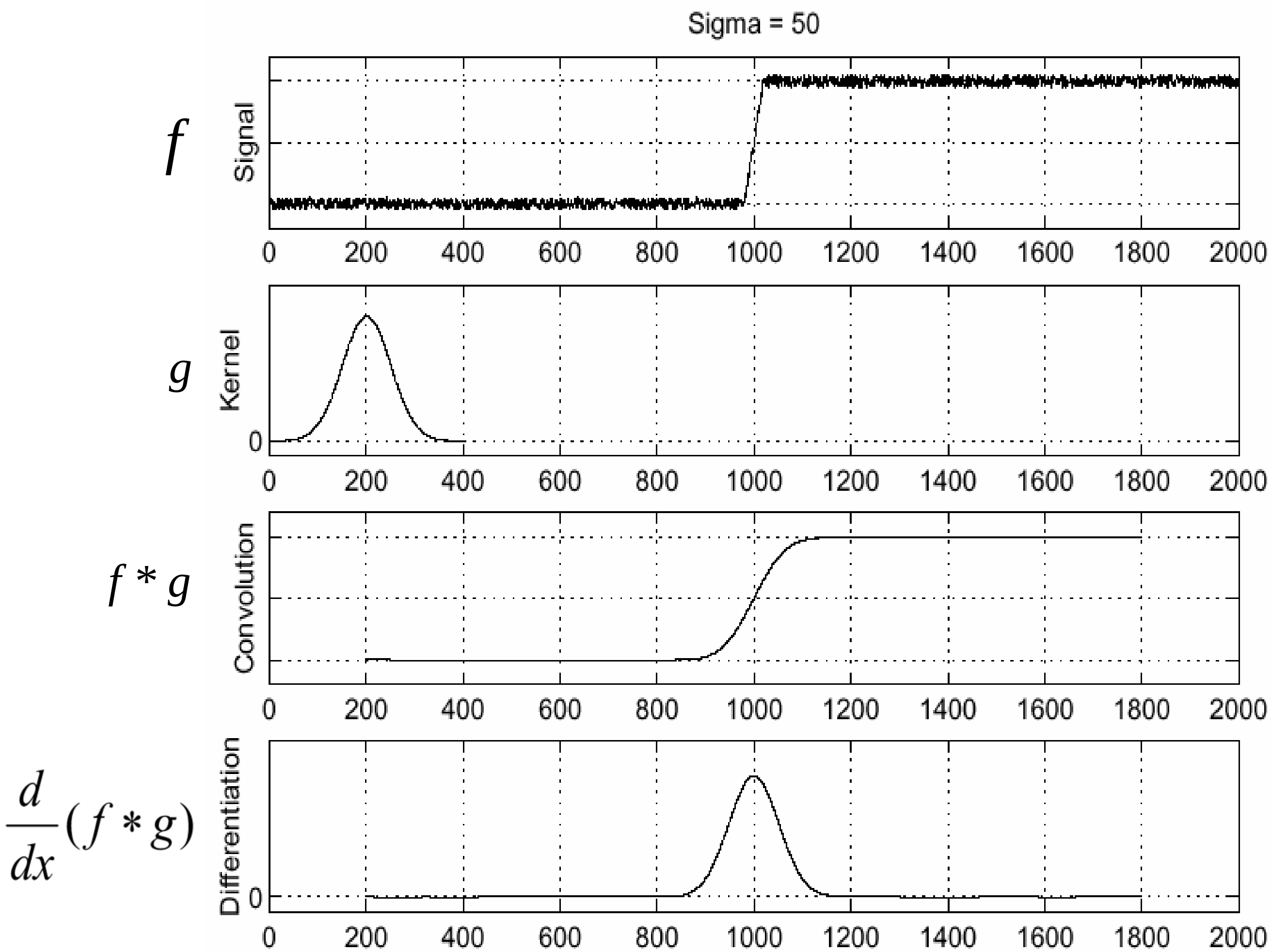
Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

Solution: smooth first



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

Source: S. Seitz

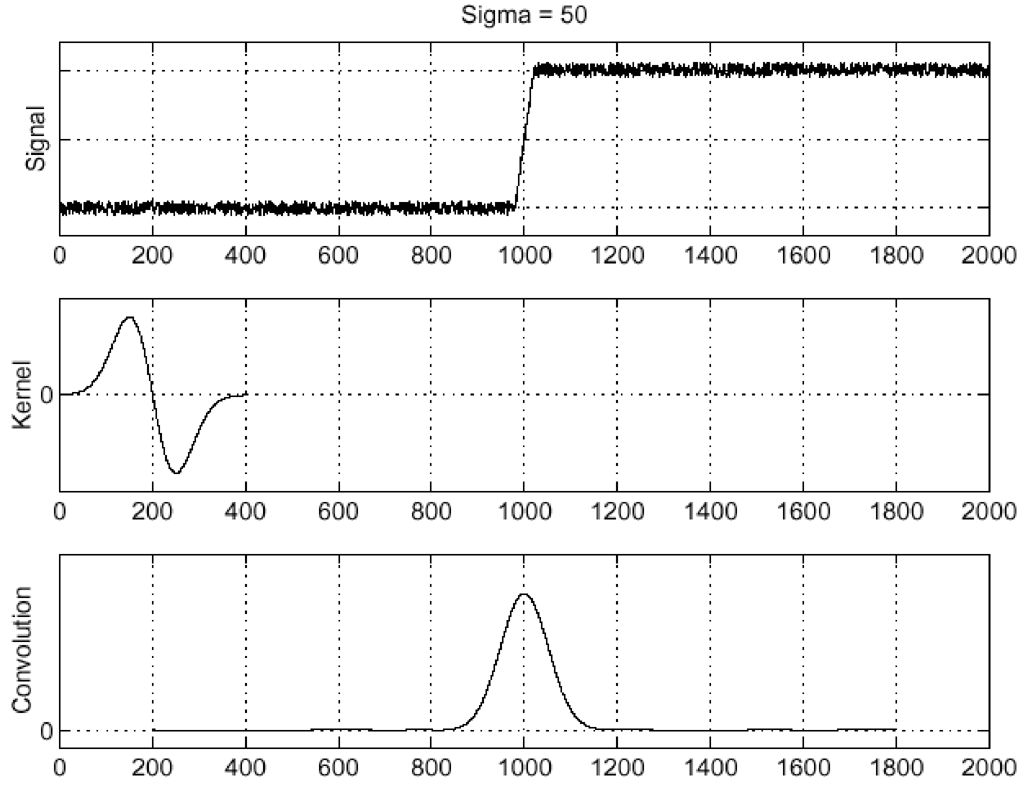
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:
$$\frac{d}{dx}(f*g) = f*\frac{d}{dx}g$$
- This saves us one operation:

f

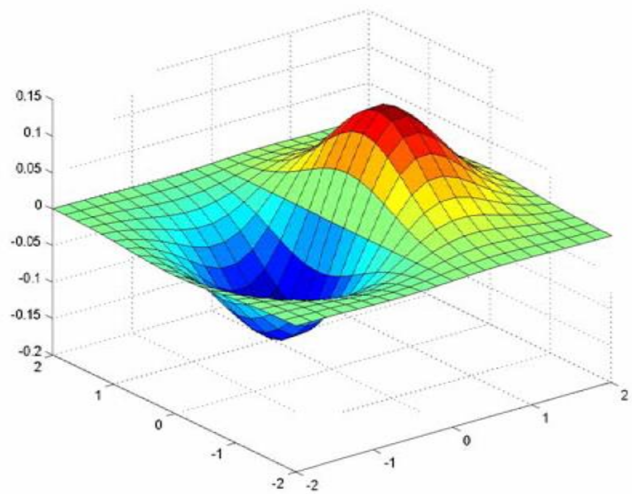
$\frac{d}{dx}g$

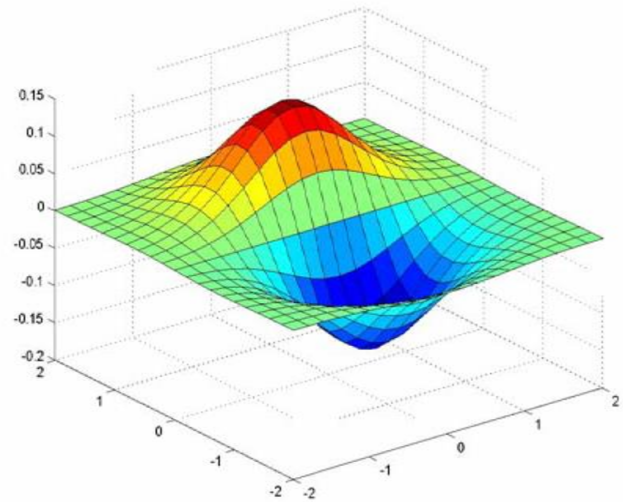
$f*\frac{d}{dx}g$



Source: S. Seitz

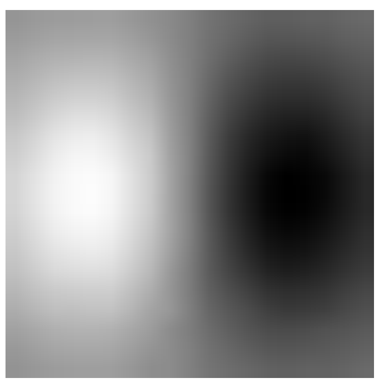
Derivative of Gaussian filters

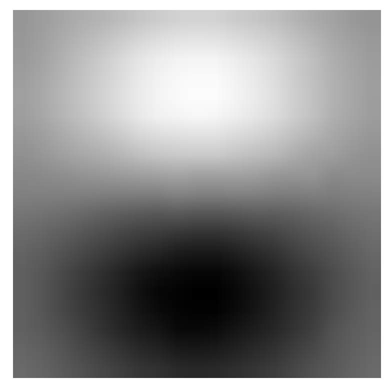




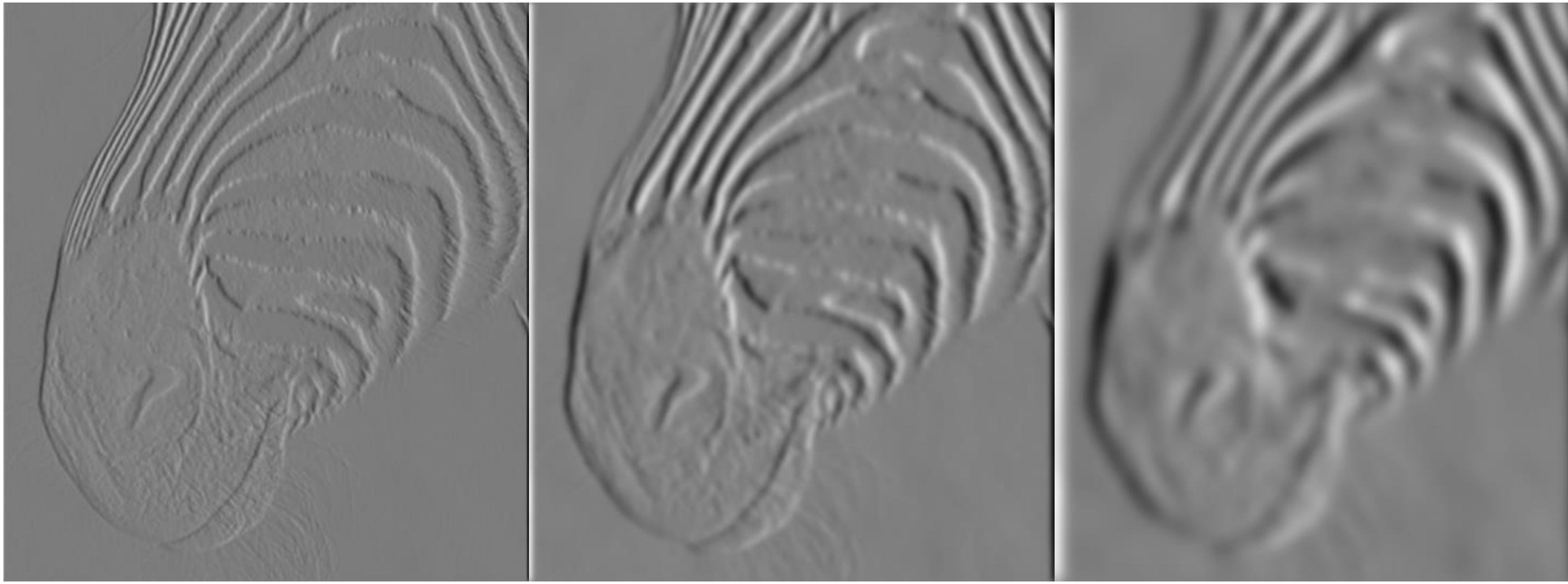
x-direction

y-direction





Scale of Gaussian derivative filter



1 pixel | 3 pixels | 7 pixels

Smoothed derivative removes noise, but blurs edge. Also finds edges at different "scales"

Source: D. Forsyth

The Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
    - Thin wide “ridges” down to single pixel width
4. **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

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

The Canny edge detector

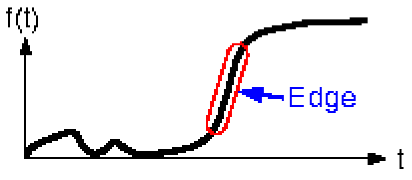
original image

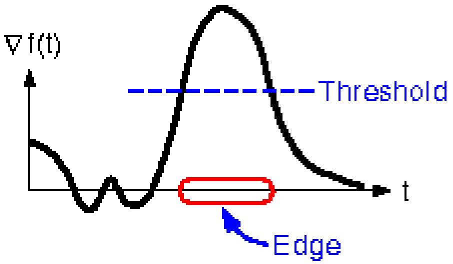
Slide credit: Steve Seitz

The Canny edge detector

norm of the gradient

The Canny edge detector

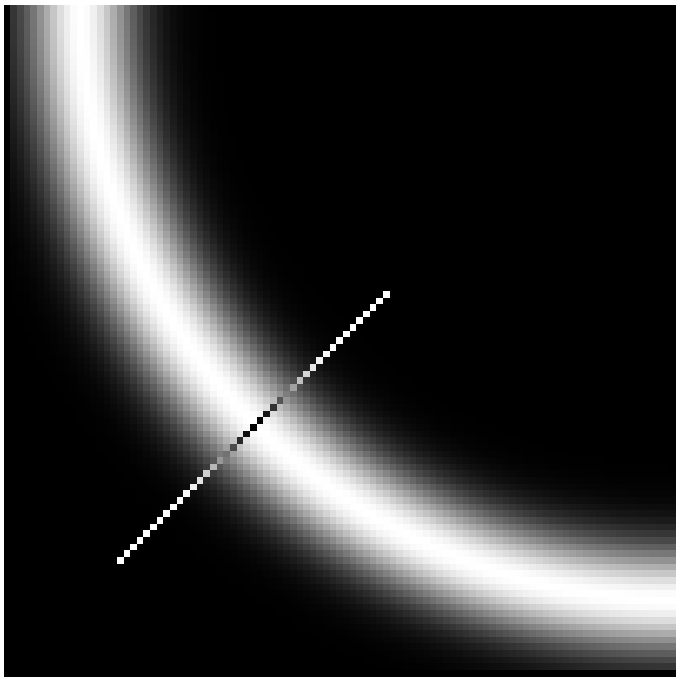


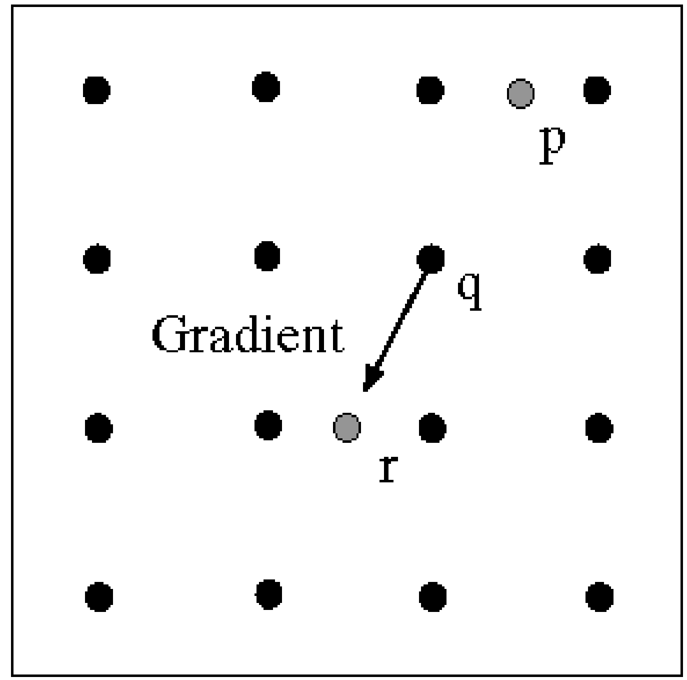


thresholding

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

Non-maximum suppression





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

- requires checking interpolated pixels p and r

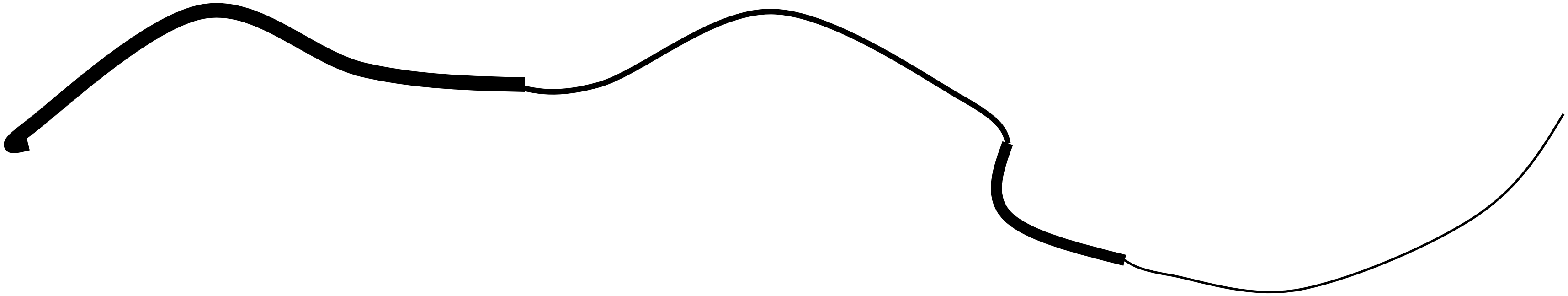
The Canny edge detector

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

thinning
(non-maximum suppression)

Hysteresis thresholding

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



Source: Steve Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Image gradients vs. meaningful contours

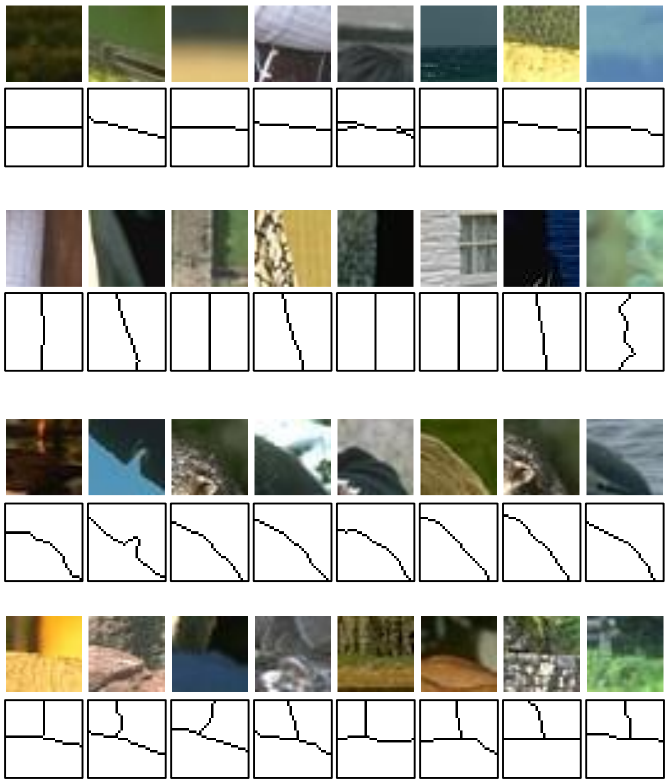
image | human segmentation | gradient magnitude

Berkeley segmentation database:

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

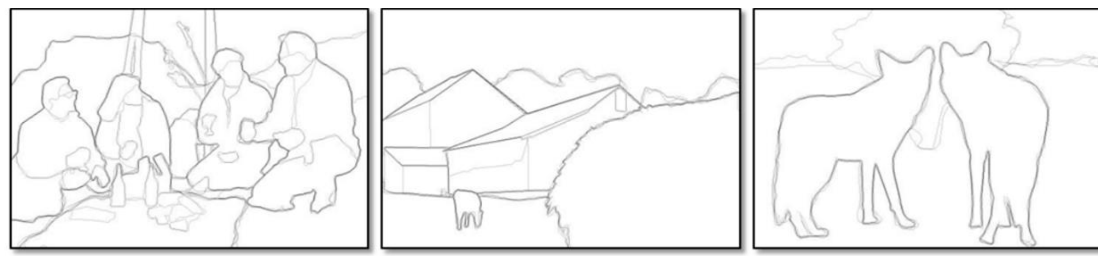
Data-driven edge detection

Training data



Input images

Ground truth

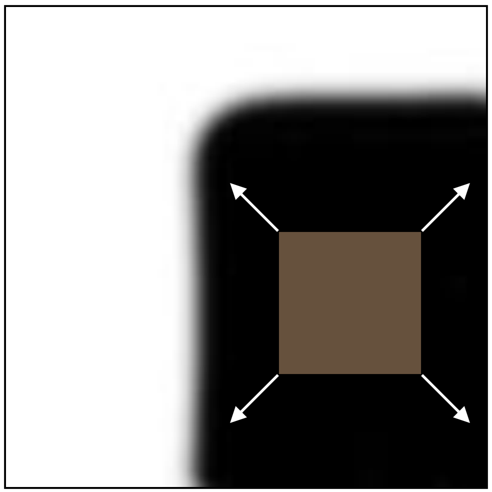


Output

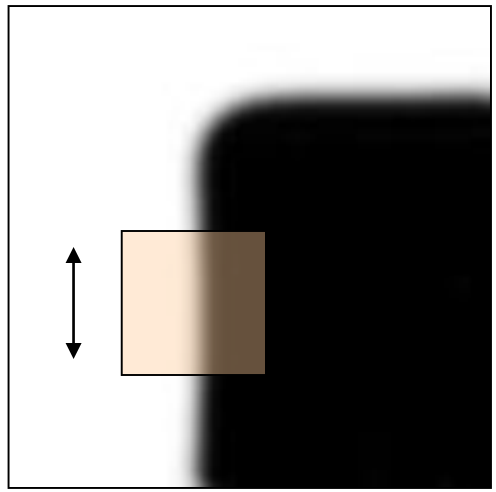
P. Dollar and L. Zitnick, [Structured forests for fast edge detection](), ICCV 2013

Corner Detection: Basic Idea

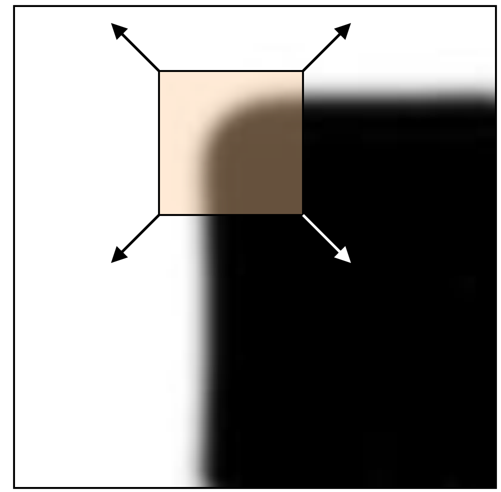
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction



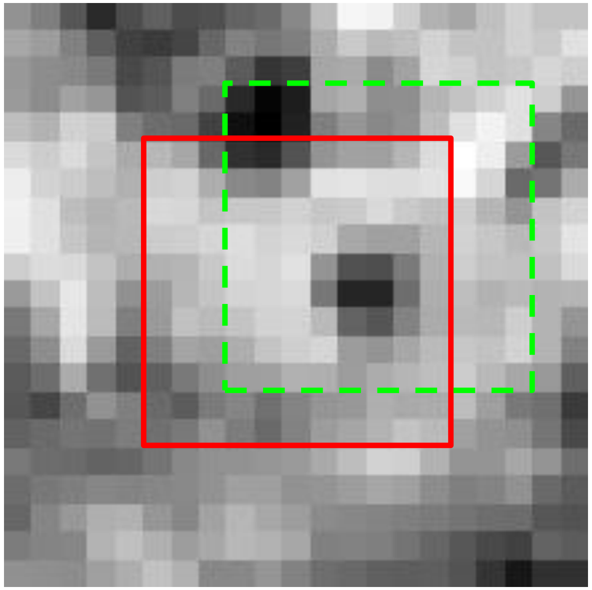
"corner":
significant change in all directions

Corner Detection: Mathematics

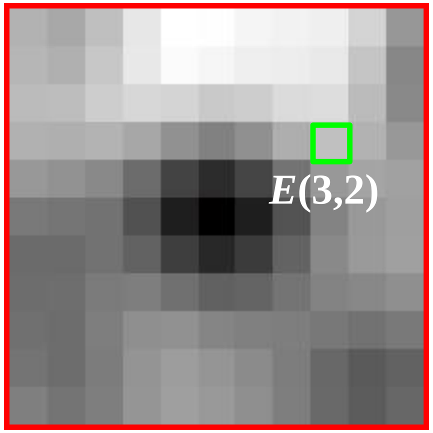
Change in appearance of window *W* for the shift [*u,v*]:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$I(x, y)$



$E(u, v)$

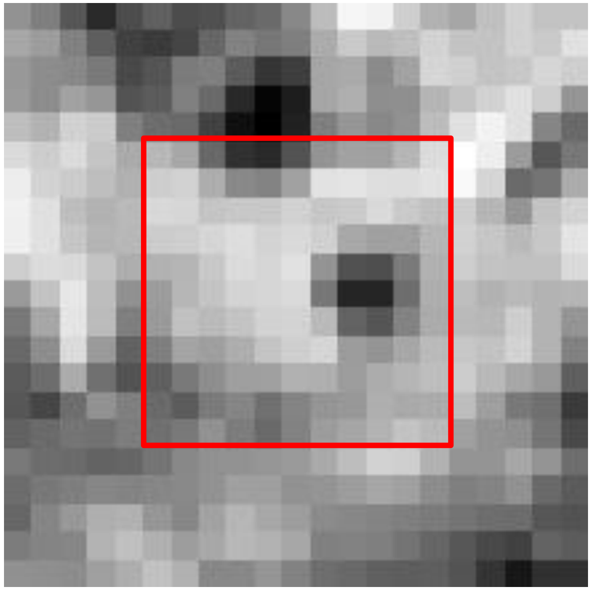


Corner Detection: Mathematics

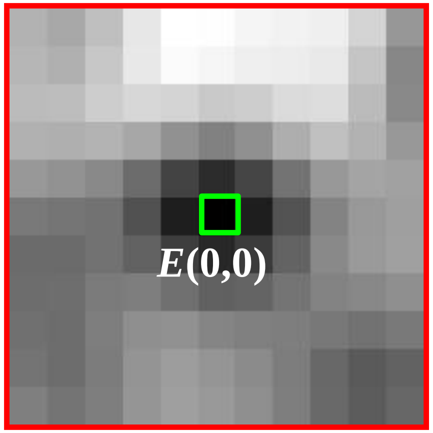
Change in appearance of window *W* for the shift [*u,v*]:

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$I(x, y)$



$E(u, v)$



Corner Detection: Mathematics

The quadratic approximation can be written as

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a *second moment matrix* computed from image derivatives:

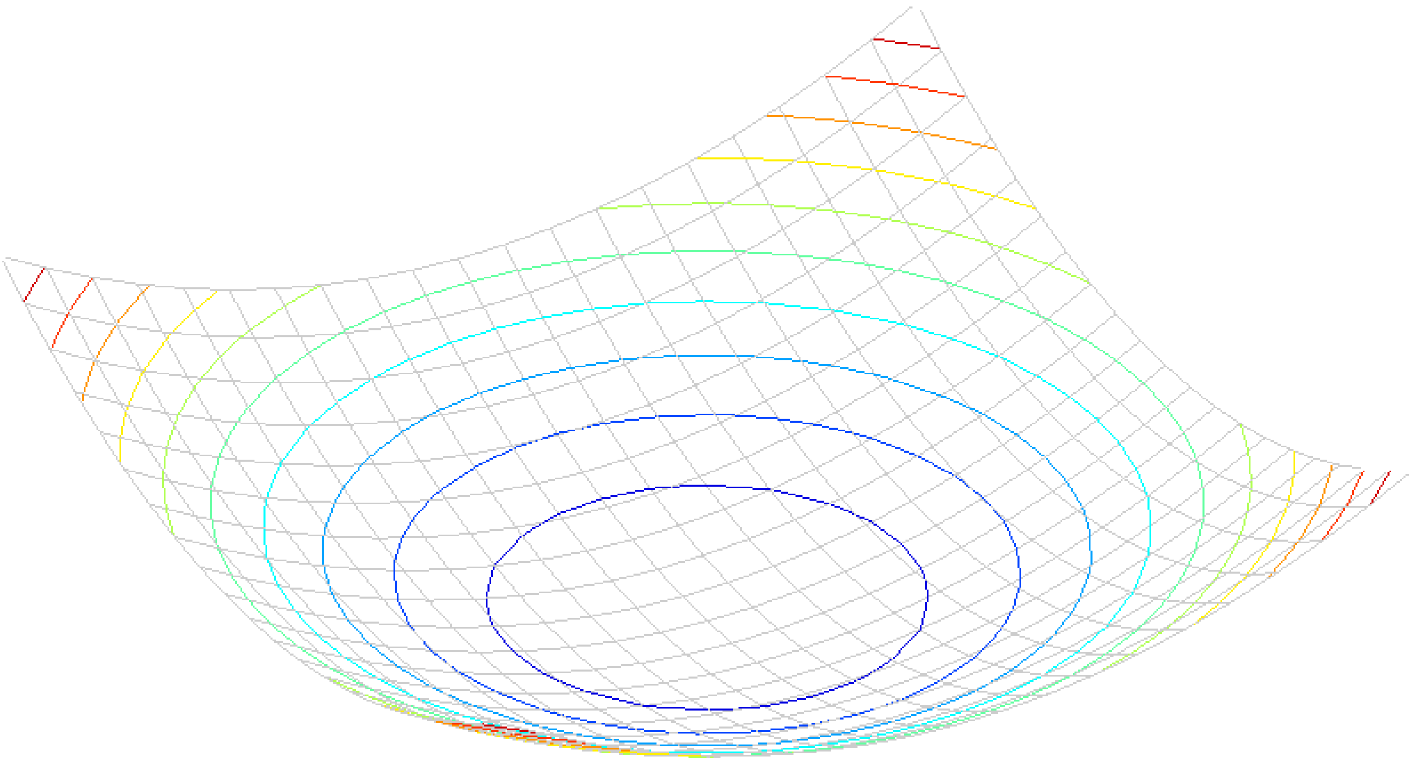
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$

(the sums are over all the pixels in the window *W*)

Quadratic Form Approximation

Consider a horizontal "slice" of $E(u, v)$: $[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.



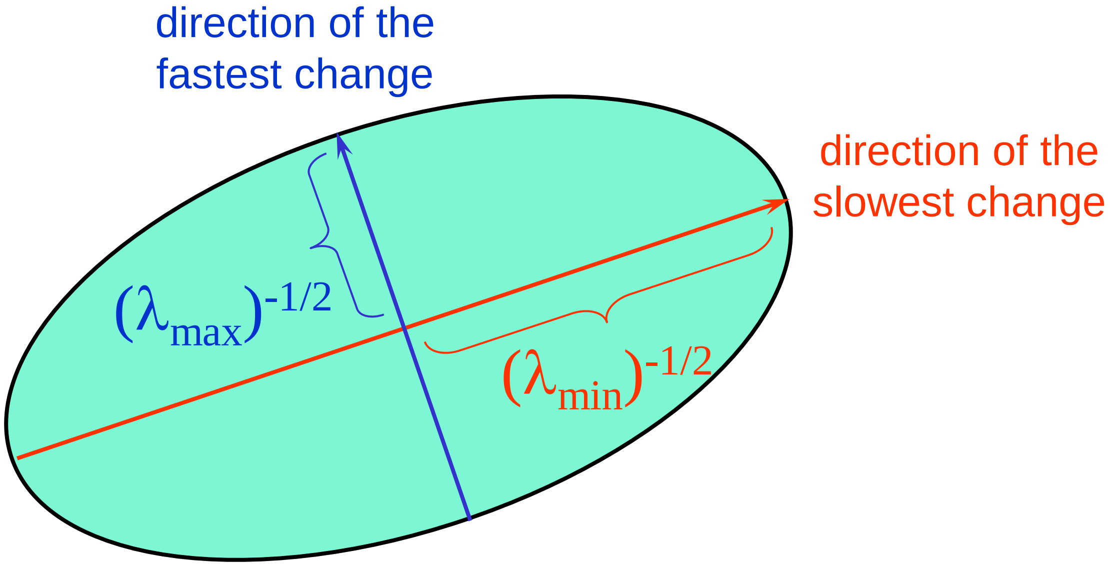
Interpreting the second moment matrix

Consider a horizontal "slice" of $E(u, v)$: $$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

This is the equation of an ellipse.

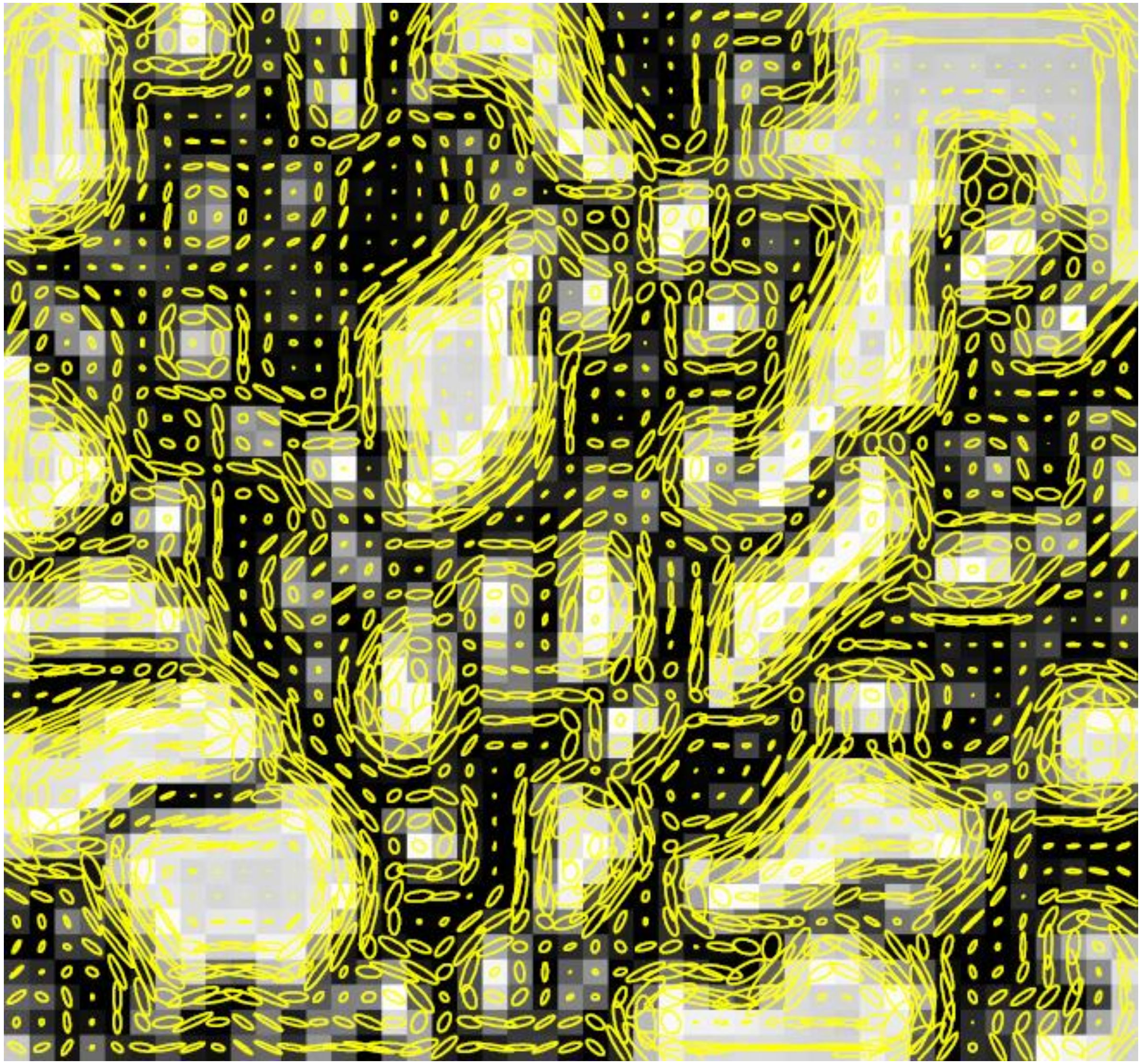
Diagonalization of M: $$M = P^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} P$$

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by *P*



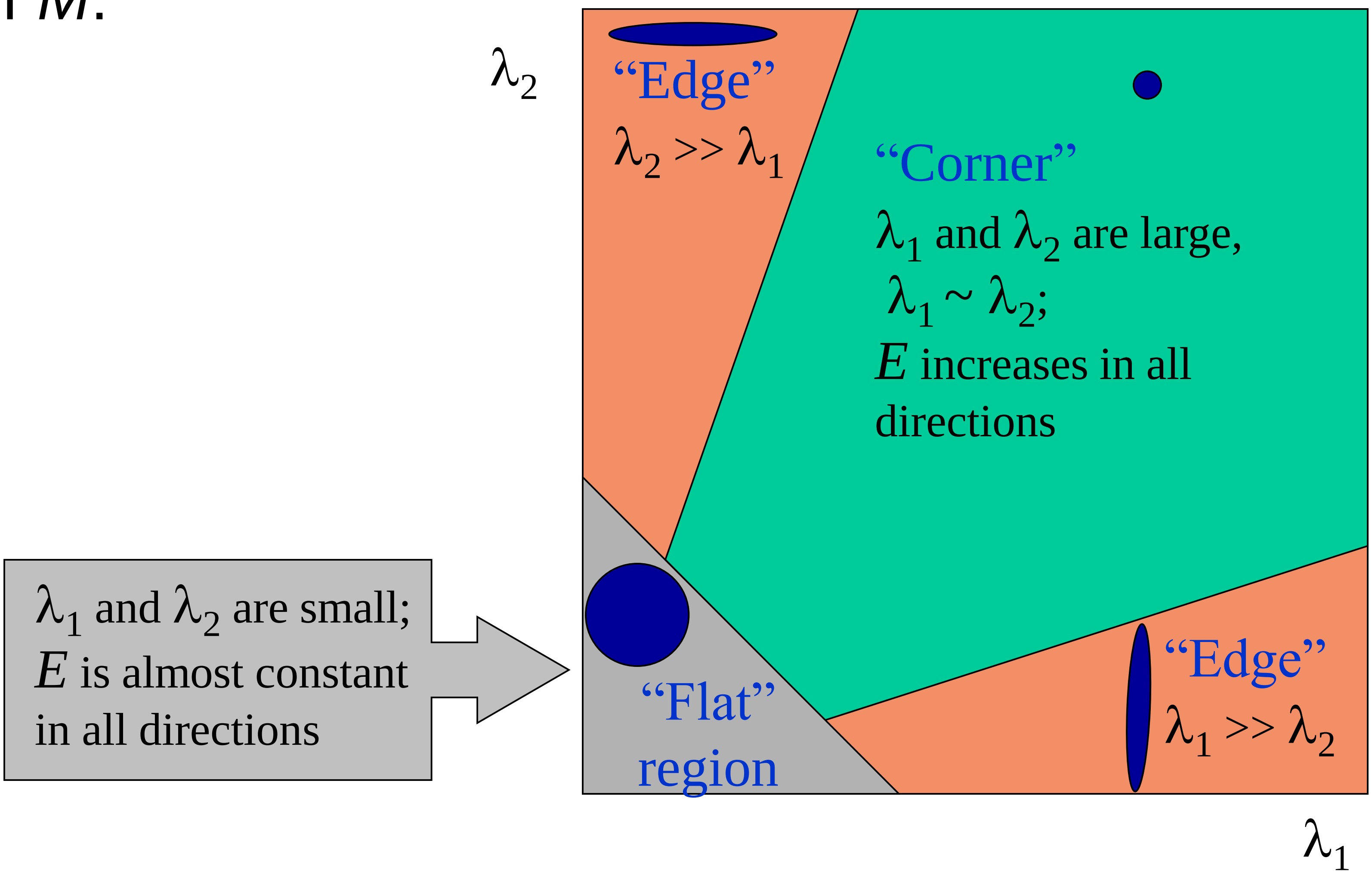
Visualization of second moment matrices

Visualization of second moment matrices



Interpreting the eigenvalues

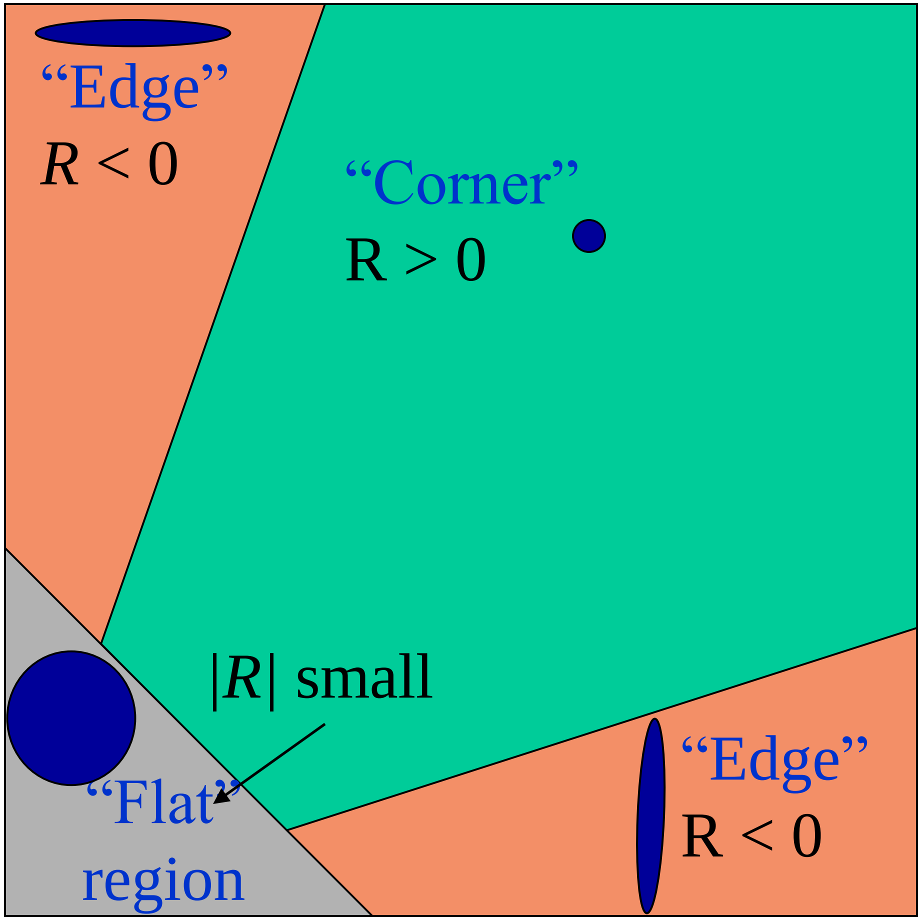
Classification of image points using eigenvalues of *M*:



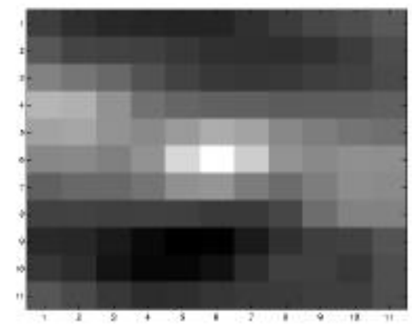
Corner response function

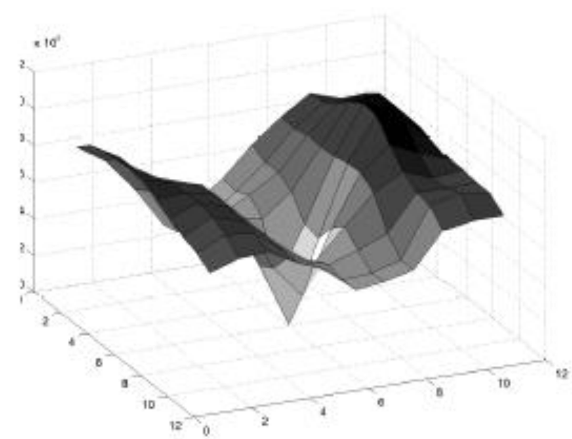
$$R = \det(M) - \alpha\, \mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



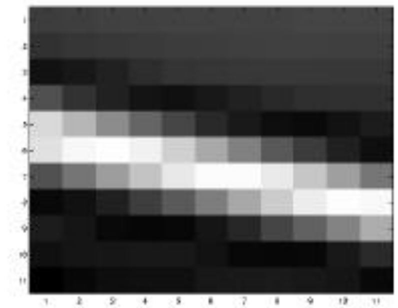
Harris corner detector

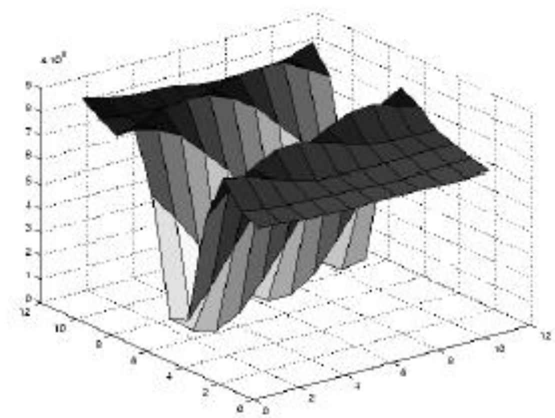




λ_1 and λ_2 are large29

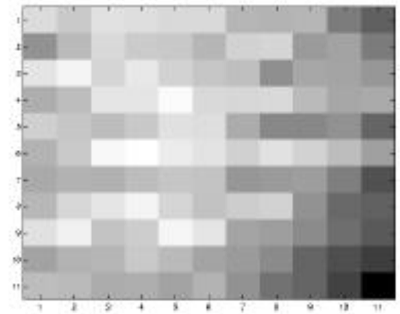
Harris corner detector

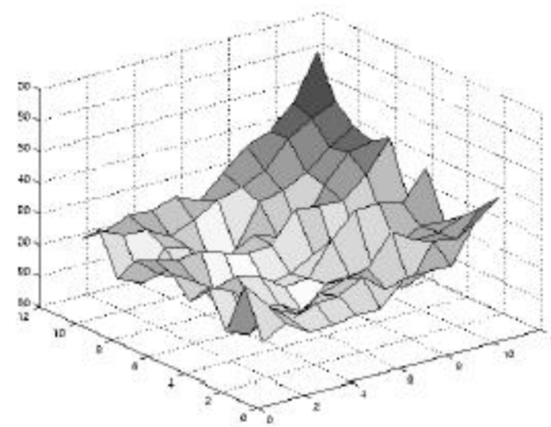




large λ_1, small λ_2 30

Harris corner detector





small λ_1, small λ_2 31

The Harris corner detector

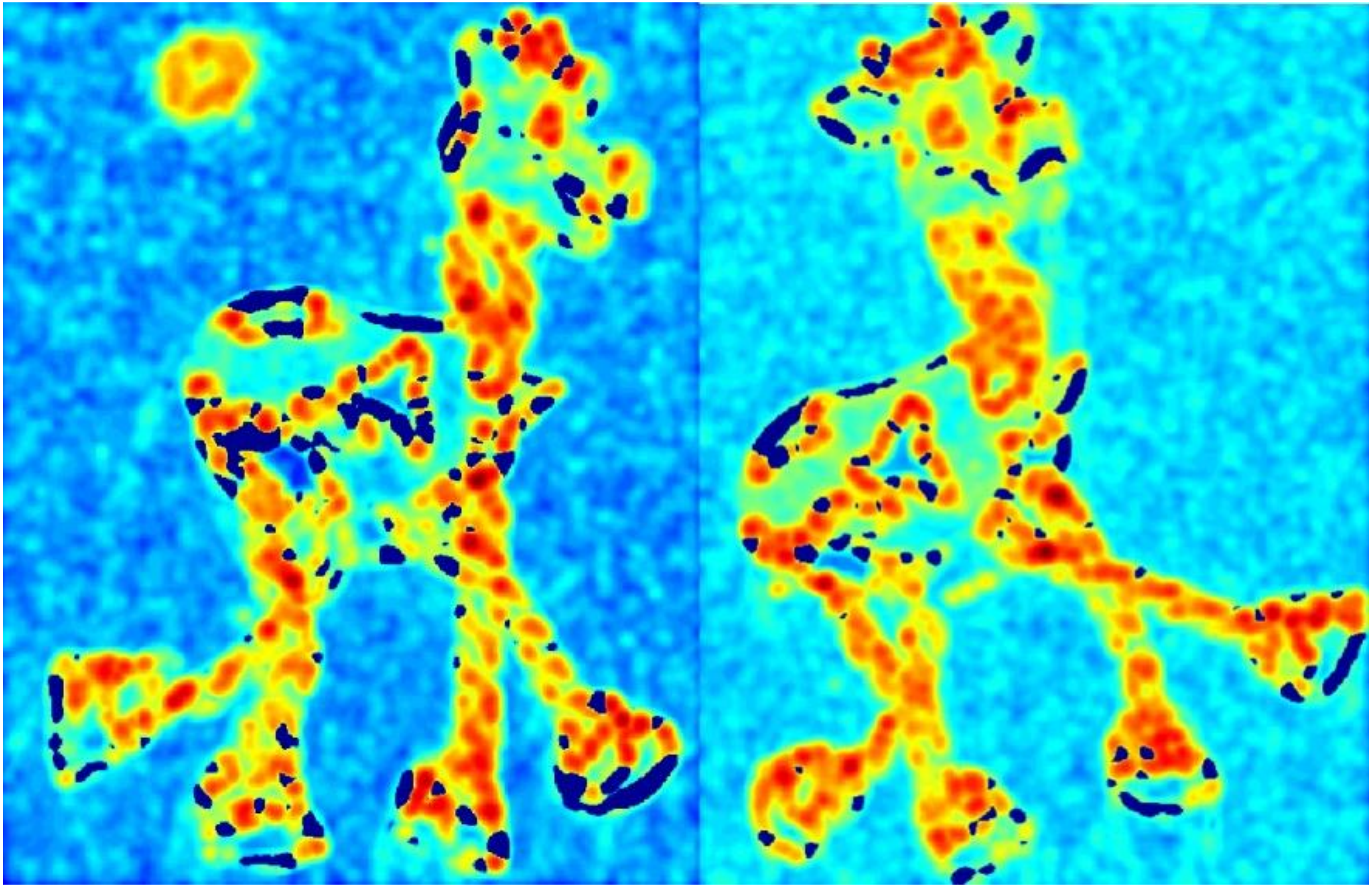
1. Compute partial derivatives at each pixel
2. Compute second moment matrix *M* in a Gaussian window around each pixel
3. Compute corner response function *R*
4. Threshold *R*
5. Find local maxima of response function (nonmaximum suppression)

C.Harris and M.Stephens. [“A Combined Corner and Edge Detector.”]
Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

Harris Detector: Steps

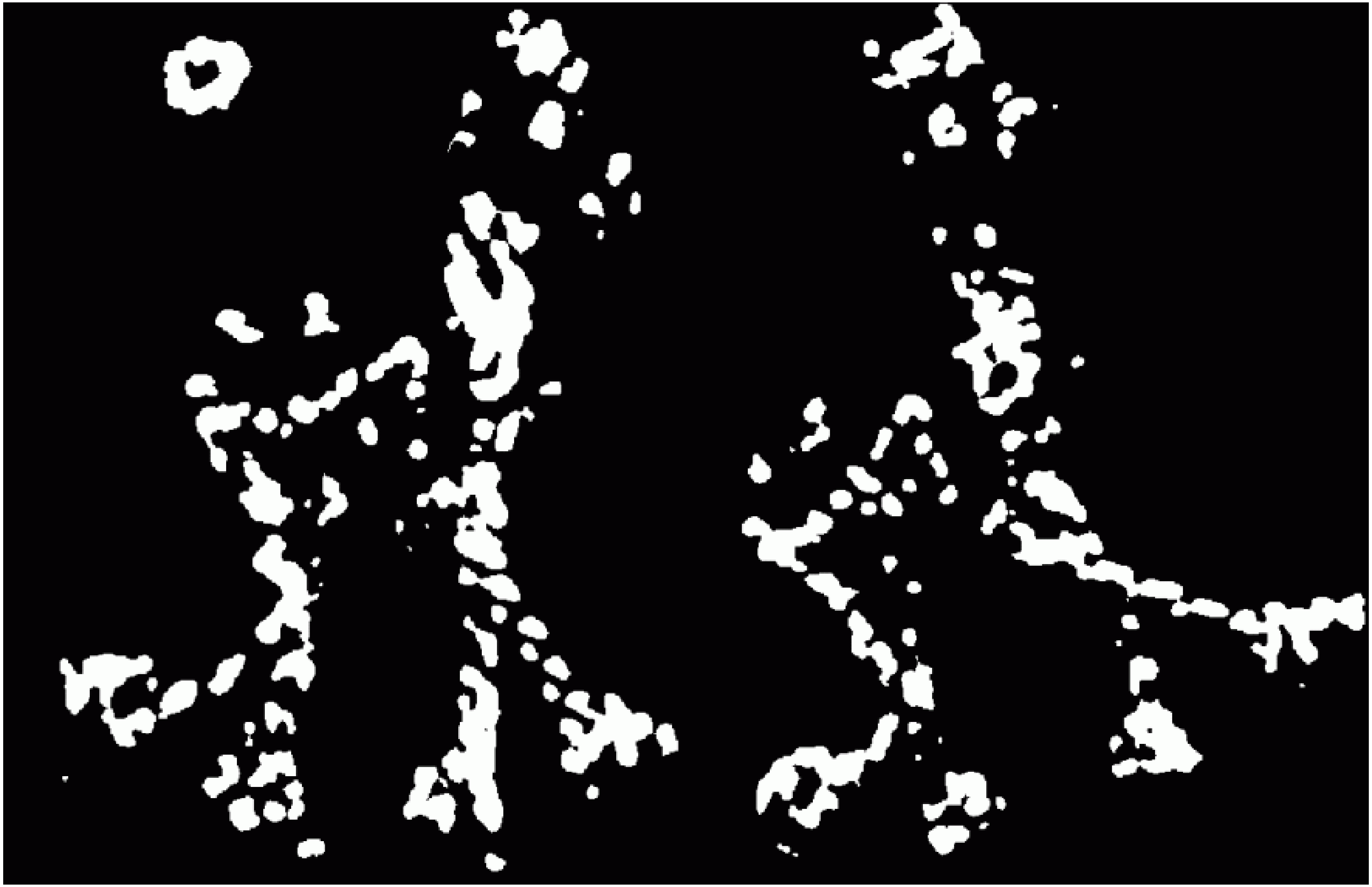
Harris Detector: Steps

Compute corner response R



Harris Detector: Steps

Find points with large corner response: $R >$ threshold



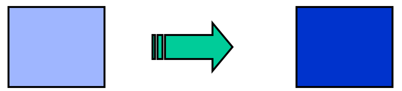
Harris Detector: Steps

Invariance and covariance

We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations

- **Invariance:** image is transformed and corner locations do not change
- **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

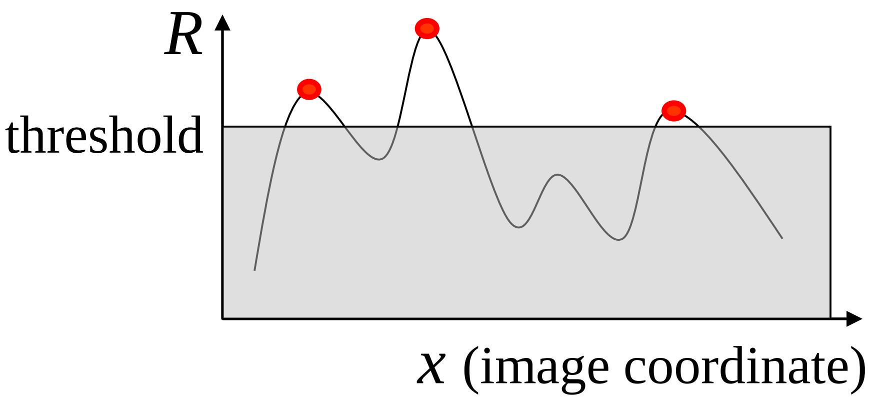
Affine intensity change

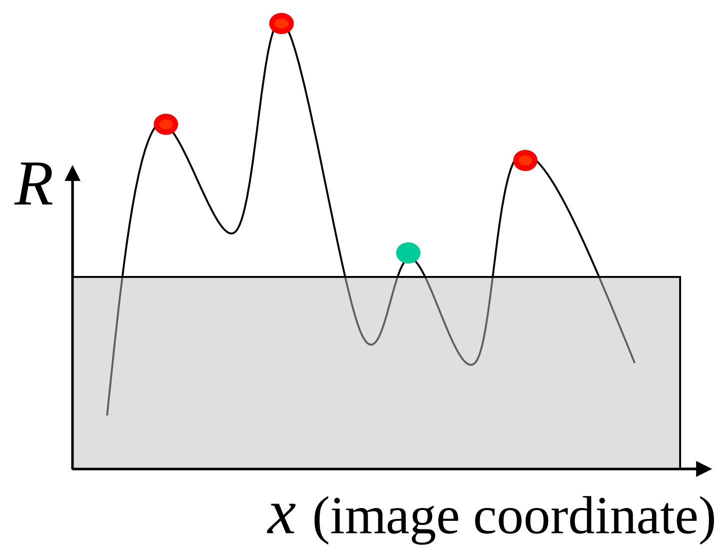


$$I \rightarrow a\,I + b$$

- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

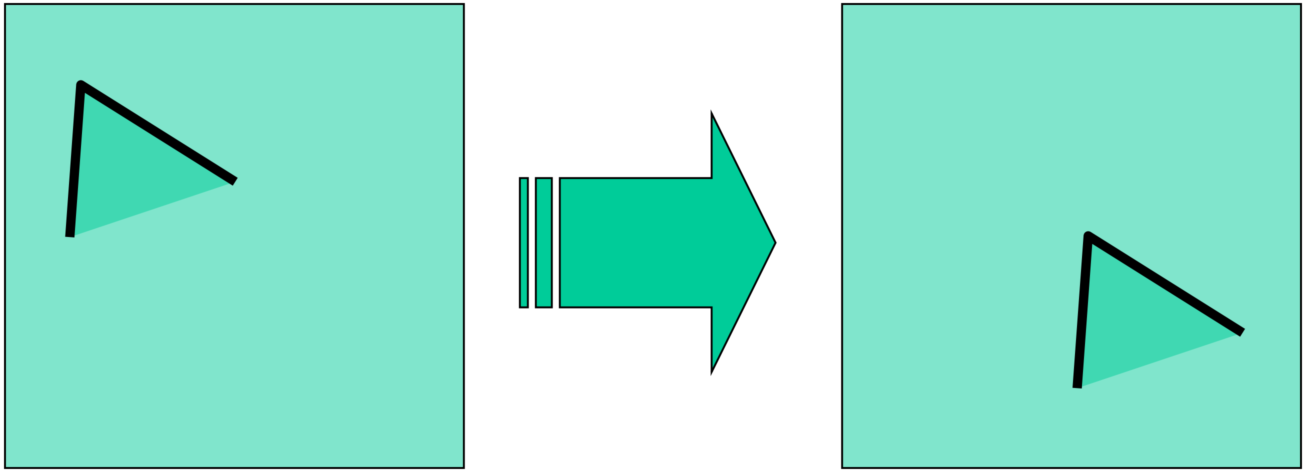
- Intensity scaling: $I \rightarrow a\,I$





Partially invariant to affine intensity change

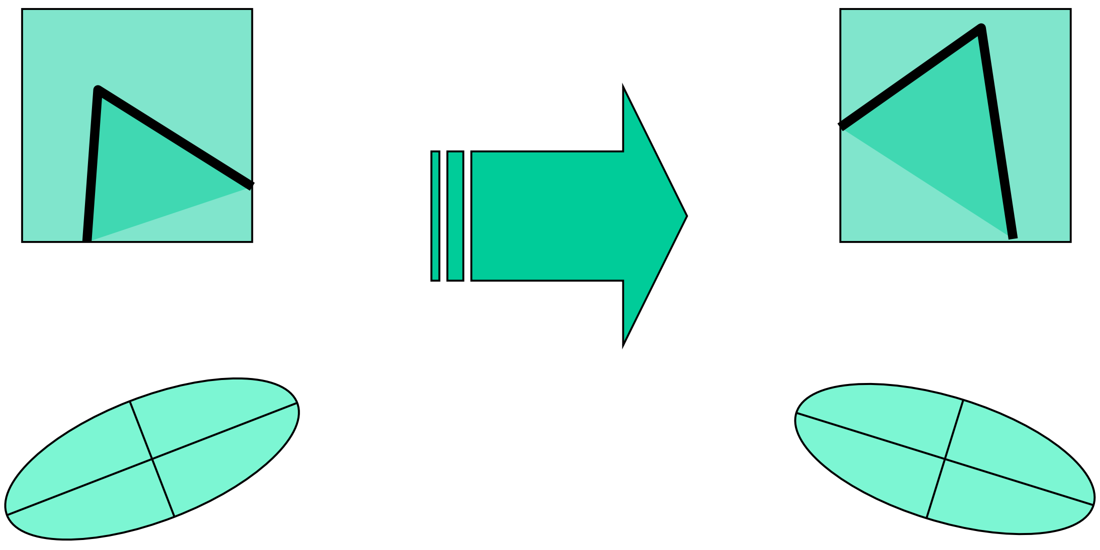
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

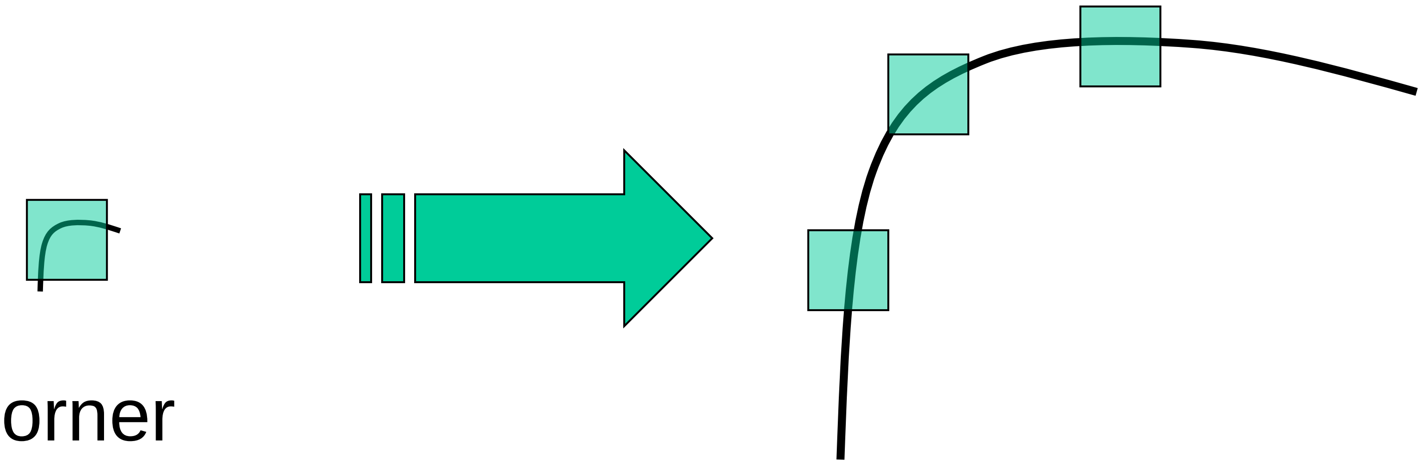
Image rotation



Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

Scaling



Corner

All points will be classified as edges

Corner location is not covariant to scaling!