

# EDGES, CORNERS, AND BLOBS

# EDGE DETECTION

Slides from Svetlana Lazebnik

# Edge Detection



[Winter in Kraków photographed by Marcin Ryczek](#)

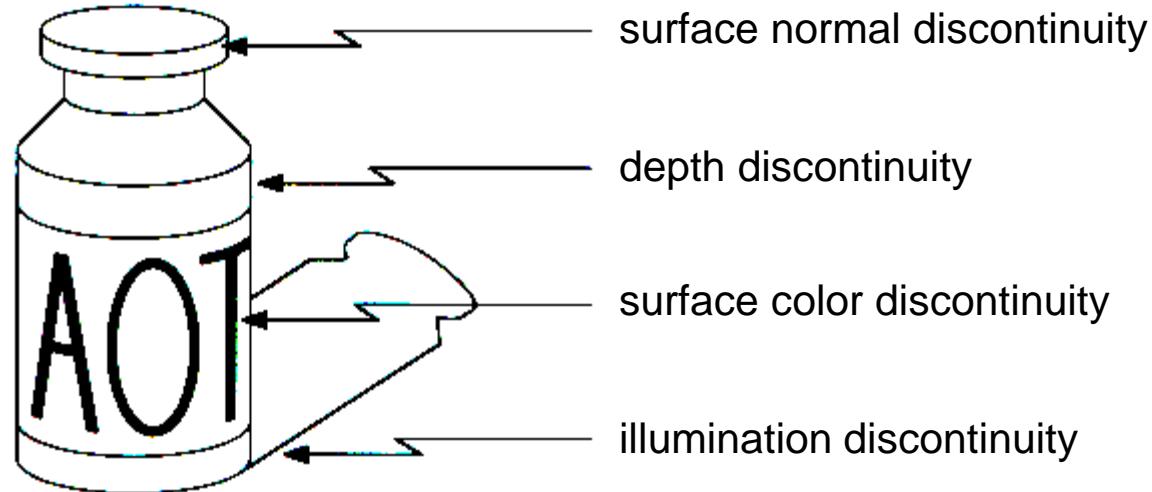
# Edge Detection

- **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)



# Origin of Edges

Edges are caused by a variety of factors:

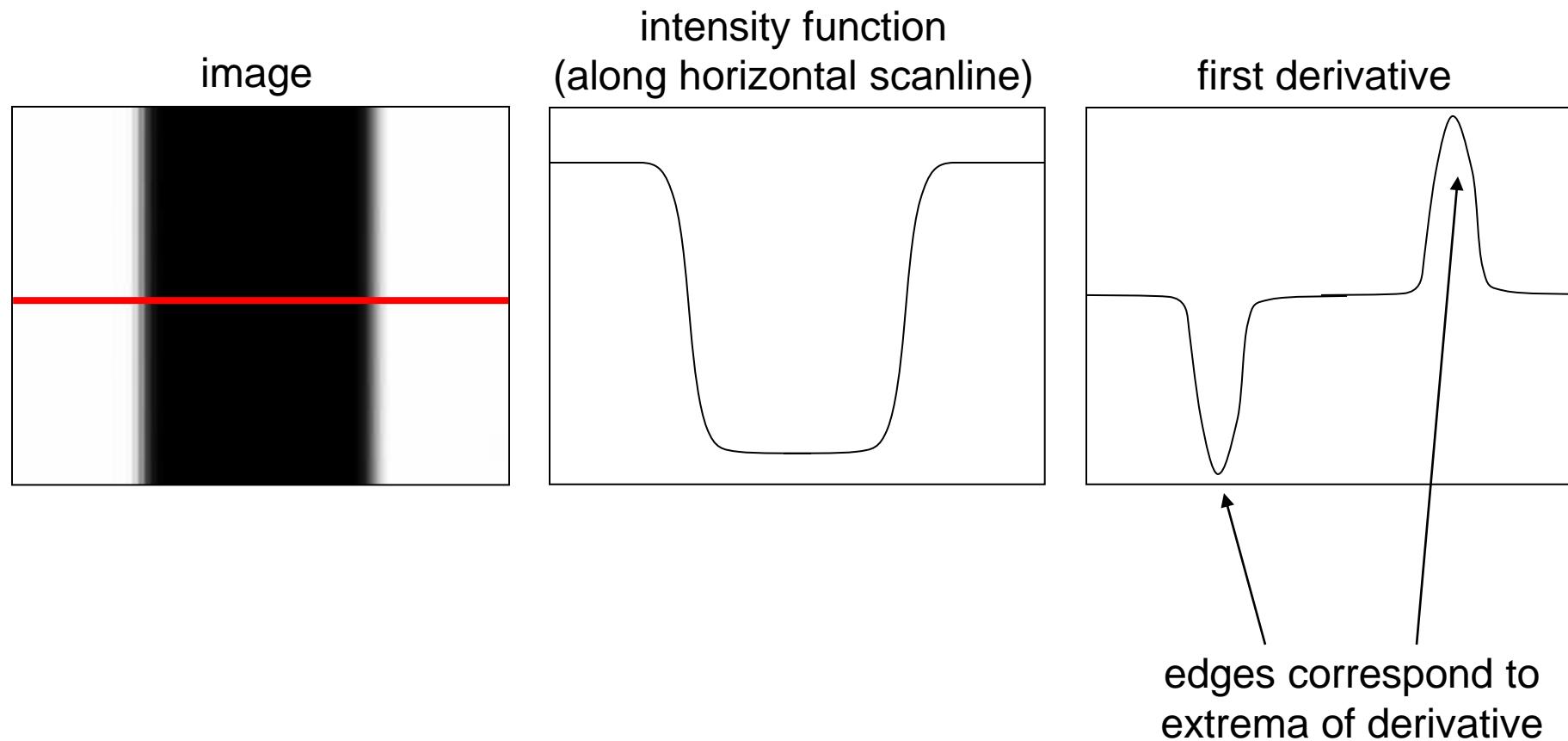


# Review: Linear Filtering

- Image blurring
- Separability
- Median filtering

# 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:

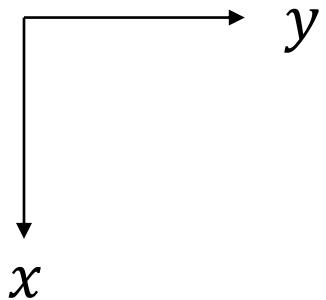
$$\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

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

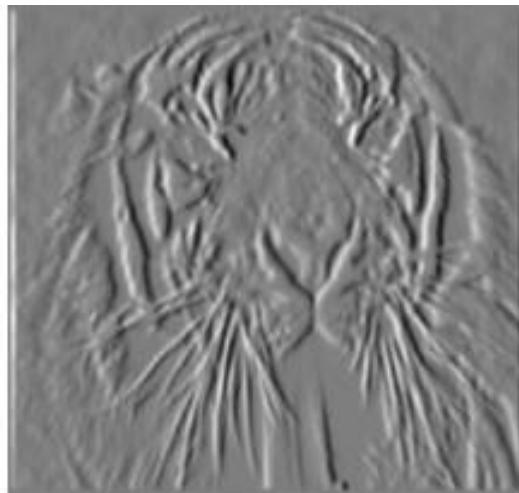
To implement the above as convolution, what would be the associated filter?

# Partial Derivatives of an Image

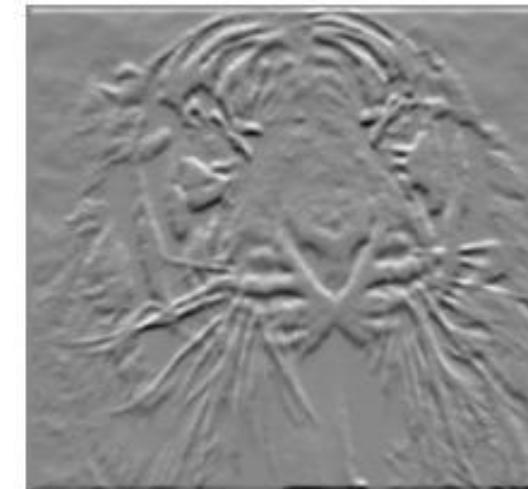


$$\frac{\partial f(x, y)}{\partial y}$$

-1	1
----	---



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



-1	1
or	
1	-1

Which shows changes with respect to *x*?

# Finite Difference Filters

Other approximations of derivative filters

Prewitt

$$M_x =$$

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

$$M_y =$$

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

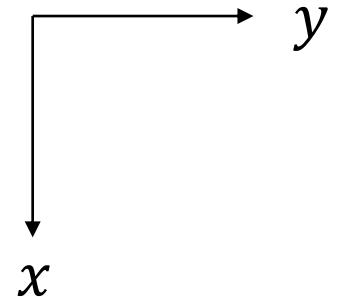
Sobel

$$M_x =$$

-1	-2	-1
0	0	0
1	2	1

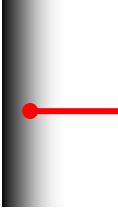
$$M_y =$$

-1	0	1
-2	0	2
-1	0	1

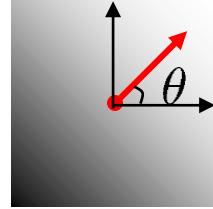


# Image Gradient

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


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


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


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

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

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

# Application: Gradient-Domain Image Editing

Goal: solve for pixel values in the target region to match gradients of the source region while keeping background pixels the same



sources/destinations



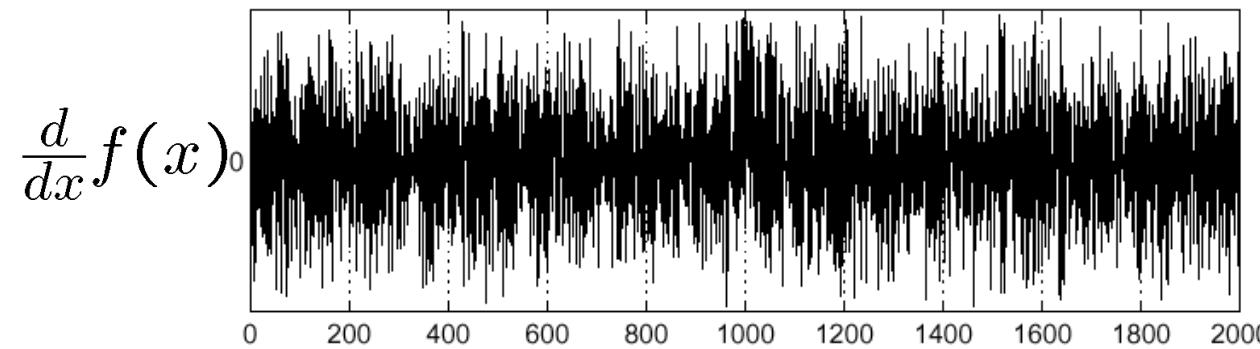
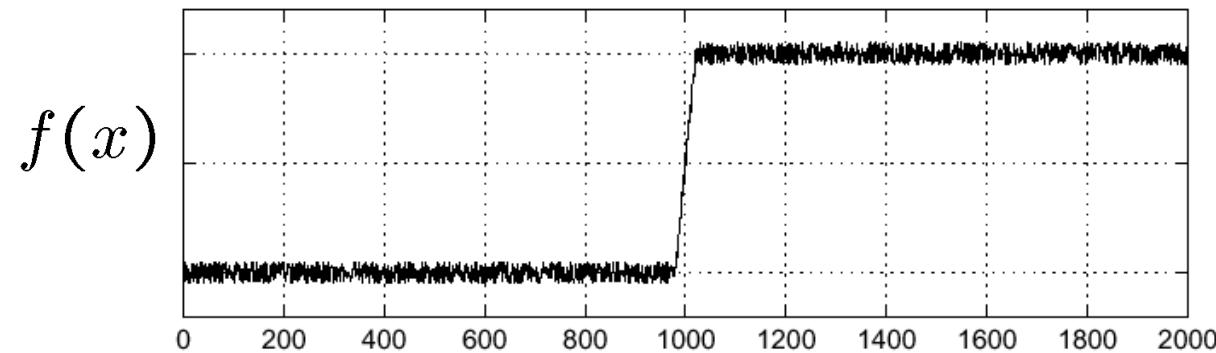
cloning



seamless cloning

# Effects of Noise

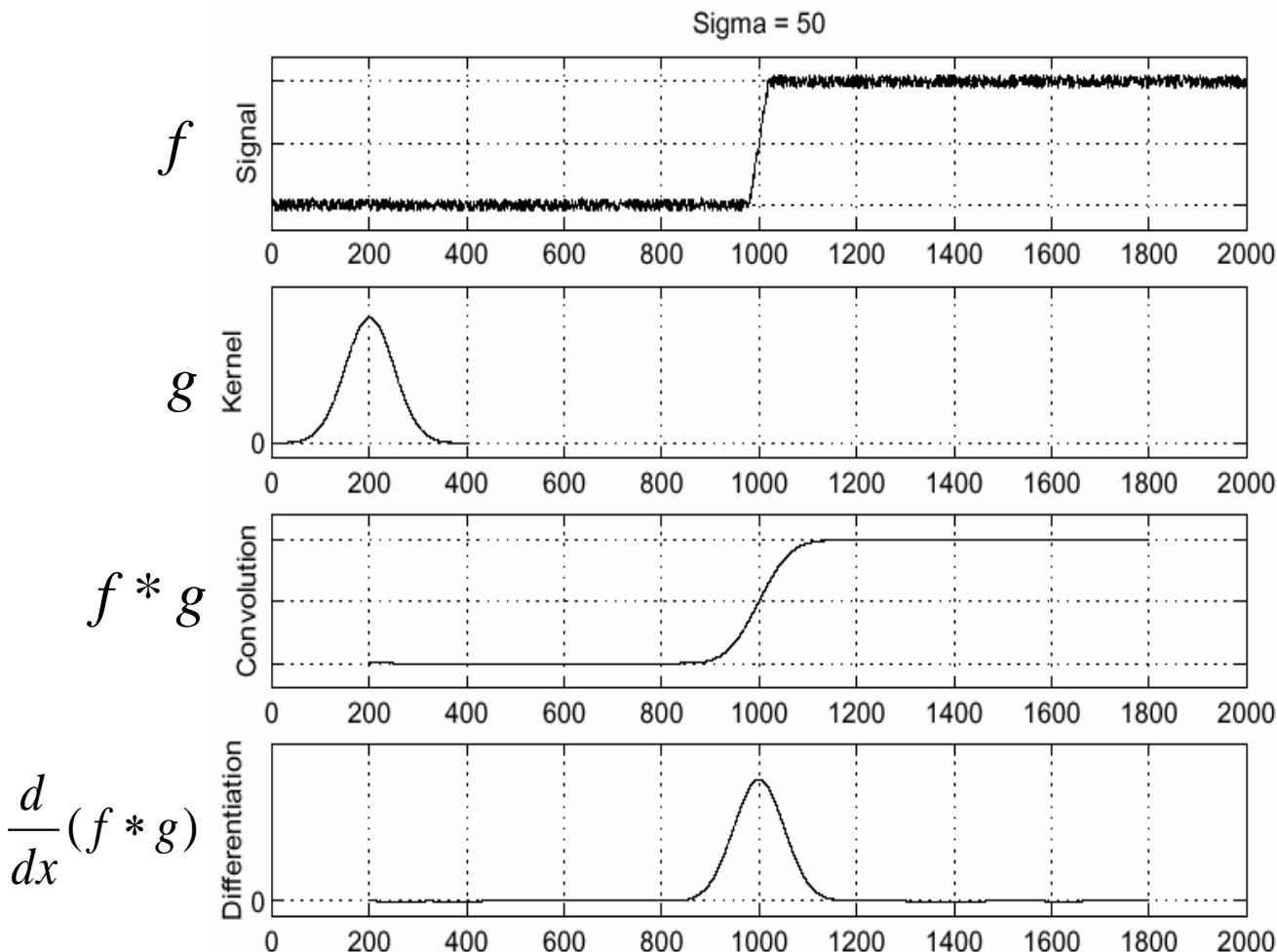
Consider a single row or column of the image



Where is the edge?

Source: S. Seitz

# Solution: Smooth First

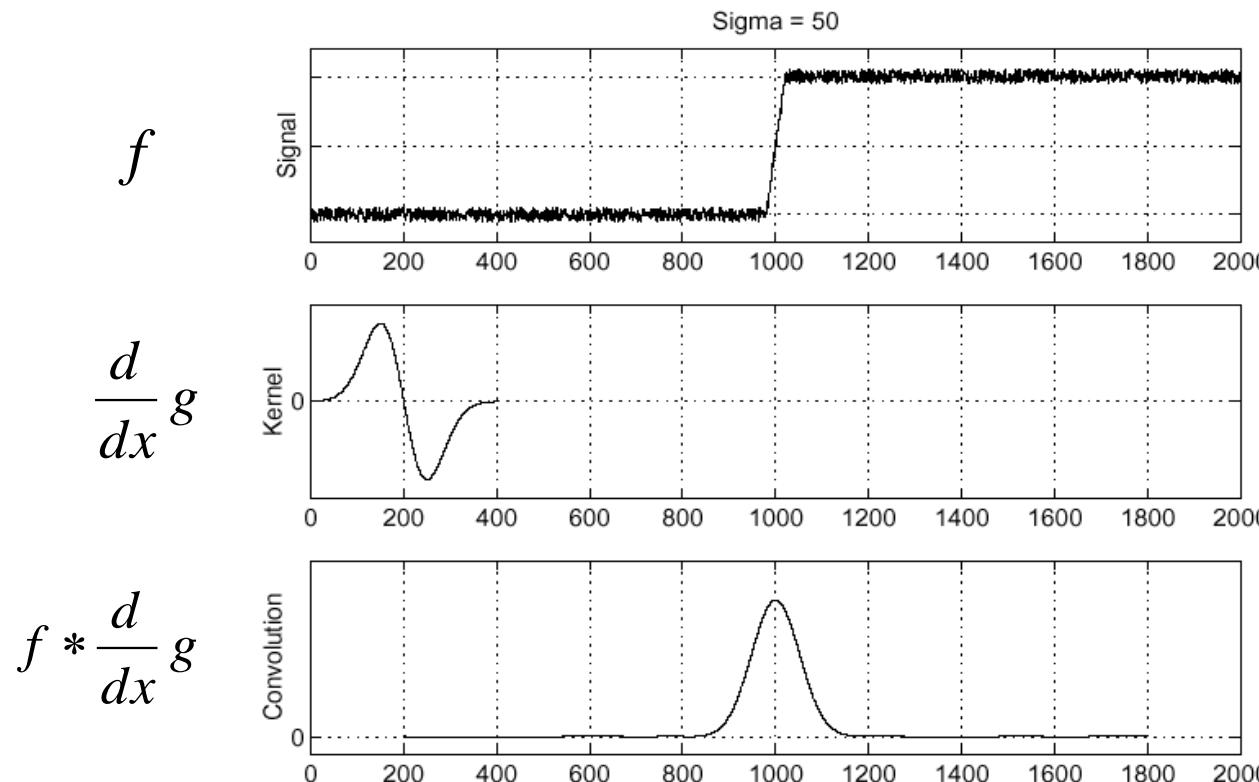


To find edges, look for peaks in  $\frac{d}{dx}(f * g)$

Source: S. Seitz

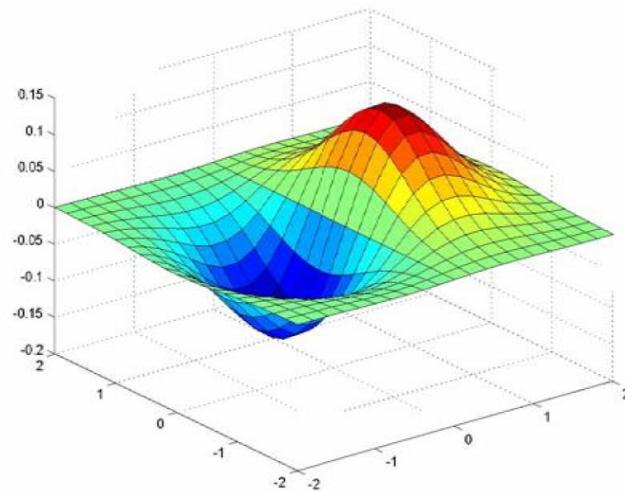
# Derivative Theorem of Convolution

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

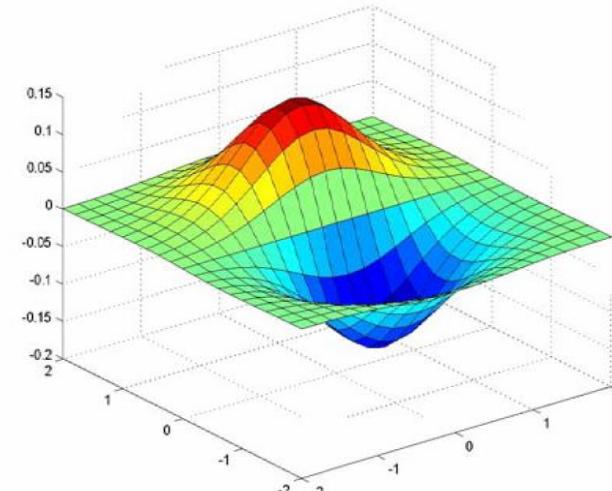


Source: S. Seitz

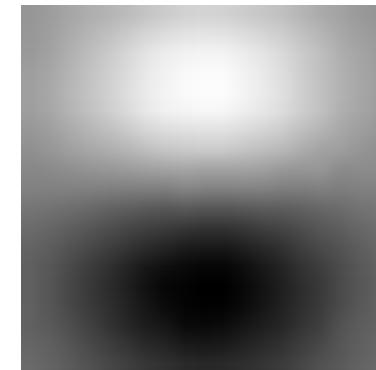
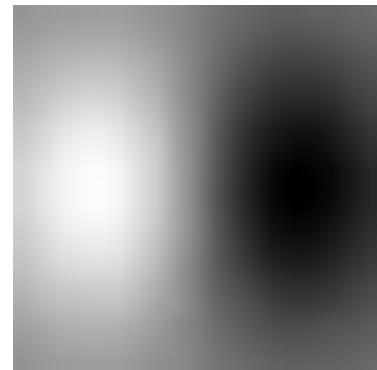
# Derivative of Gaussian Filters



x-direction

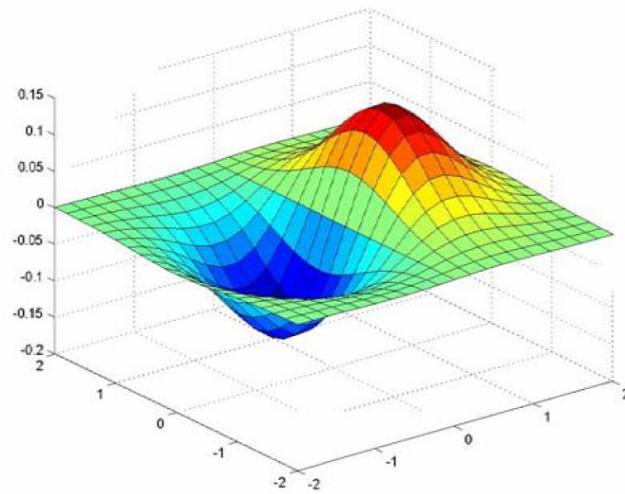


y-direction

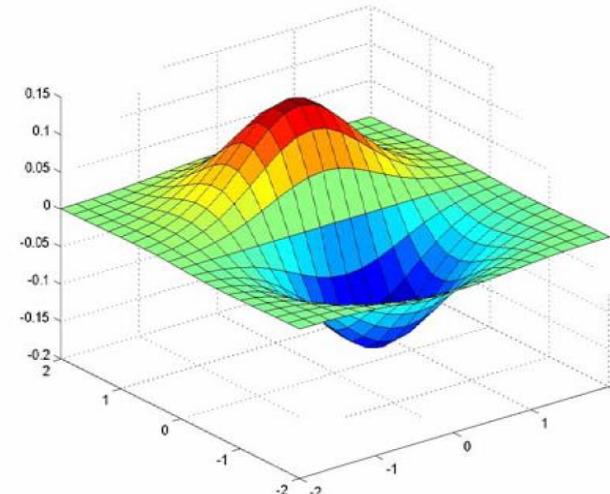


Which one finds horizontal/vertical edges?

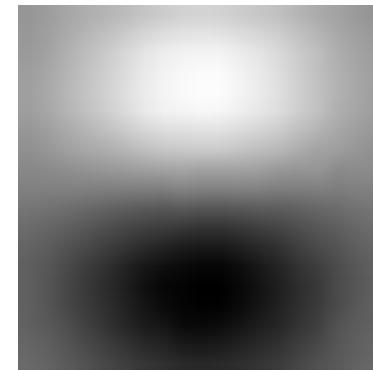
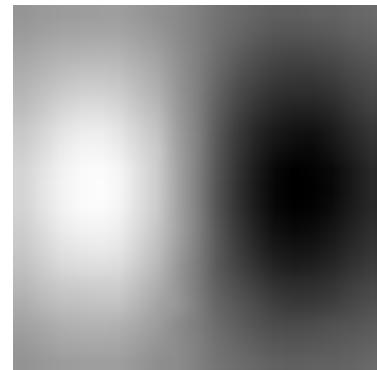
# Derivative of Gaussian Filters



x-direction



y-direction



Are these filters separable?

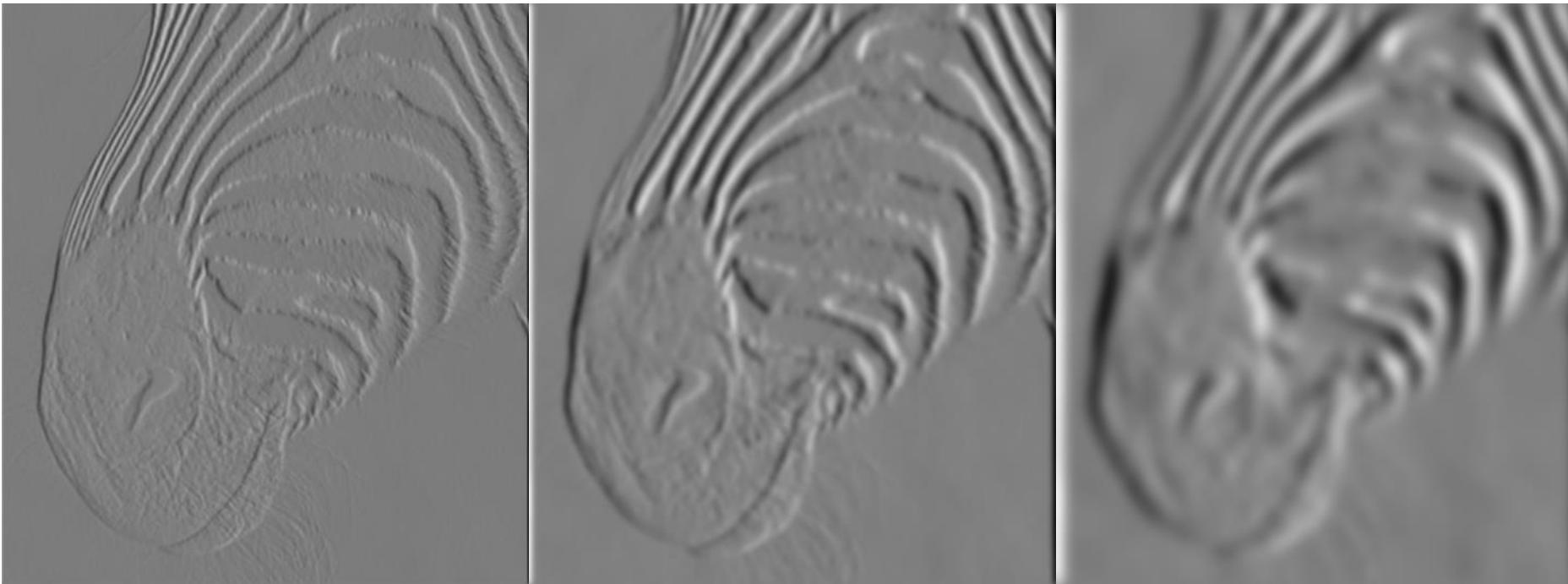
# Recall: Separability of the Gaussian Filter

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left( \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

The 2D Gaussian can be expressed as the product of two functions, one a function of  $x$  and the other a function of  $y$

In this case, the two functions are the (identical) 1D Gaussian

# Scale of Gaussian Derivative Filter



1 pixel

3 pixels

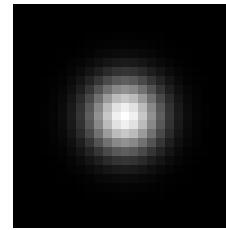
7 pixels

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

# Review: Smoothing vs. Derivative Filters

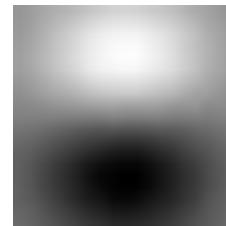
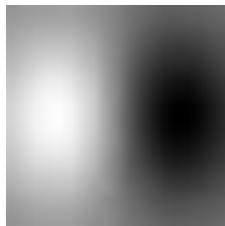
## Smoothing filters

- Gaussian: remove “high-frequency” components; “low-pass” filter
- Can the values of a smoothing filter be negative?
- What should the values sum to?
  - **One:** constant regions are not affected by the filter



## Derivative filters

- Derivatives of Gaussian
- Can the values of a derivative filter be negative?
- What should the values sum to?
  - **Zero:** no response in constant regions
- High absolute value at points of high contrast



# 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

# The Canny Edge Detector



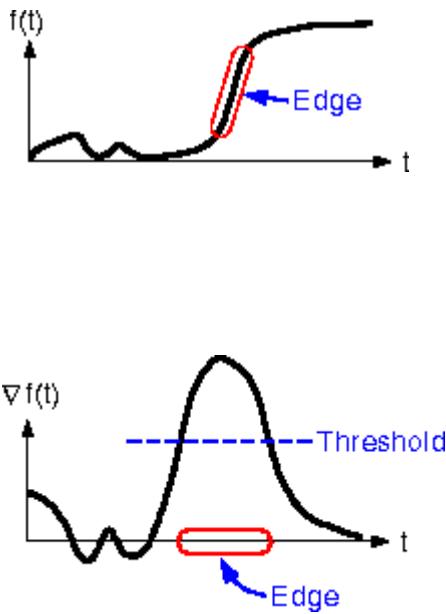
norm of the gradient

# The Canny Edge Detector



thresholding

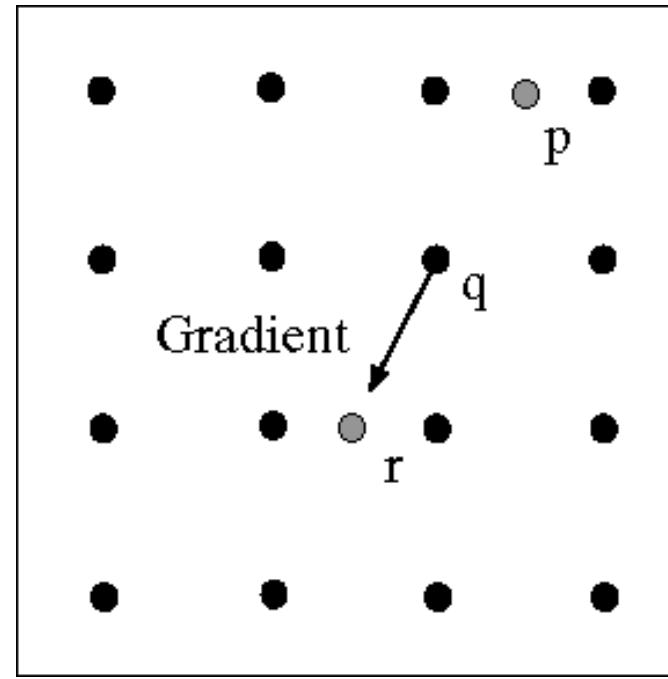
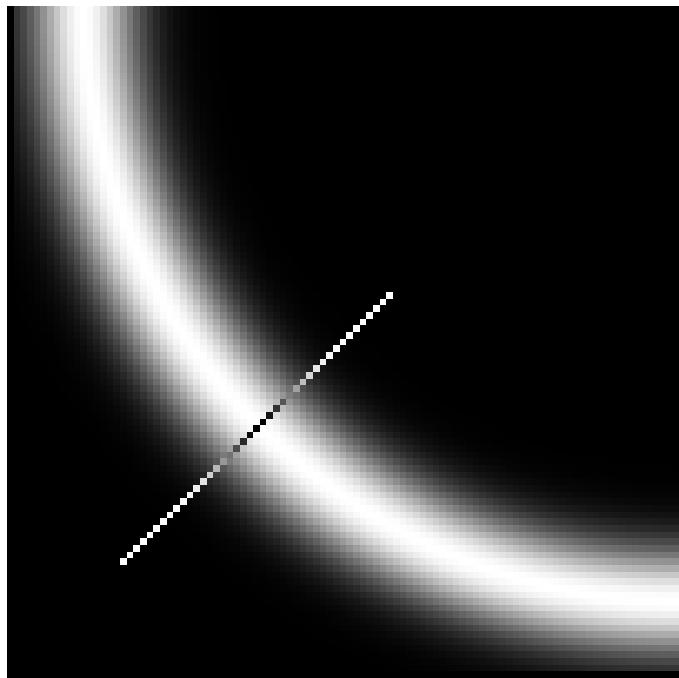
# The Canny Edge Detector



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

thresholding

# 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

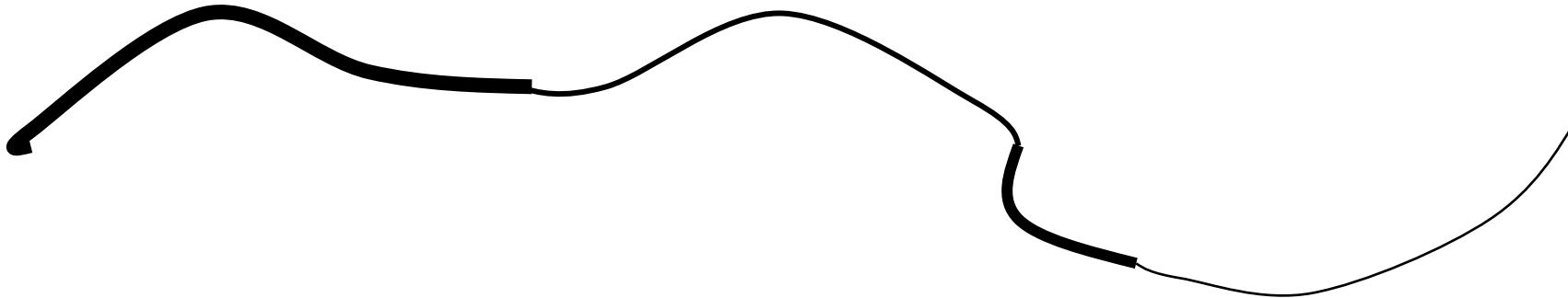


thinning  
(non-maximum suppression)

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

# Hysteresis Thresholding

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



# Hysteresis Thresholding



original image



high threshold  
(strong edges)



low threshold  
(weak edges)



hysteresis threshold

# Recap: Canny Edge Detector

1. Compute x and y gradient images
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

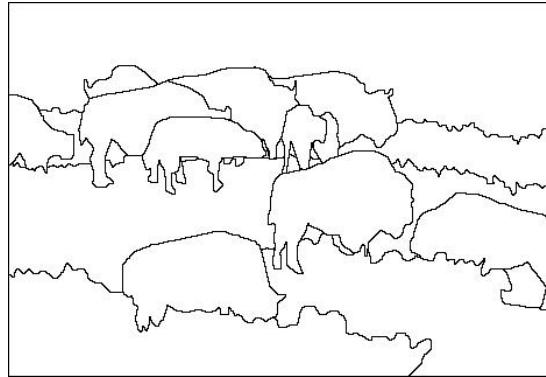
J. Canny, [\*A Computational Approach To Edge Detection\*](#), IEEE  
Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

# Image Gradients vs. Meaningful Contours

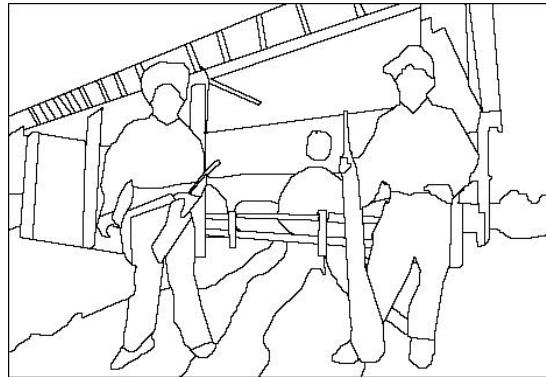
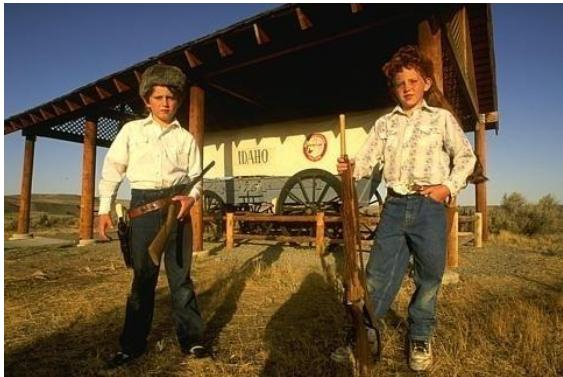
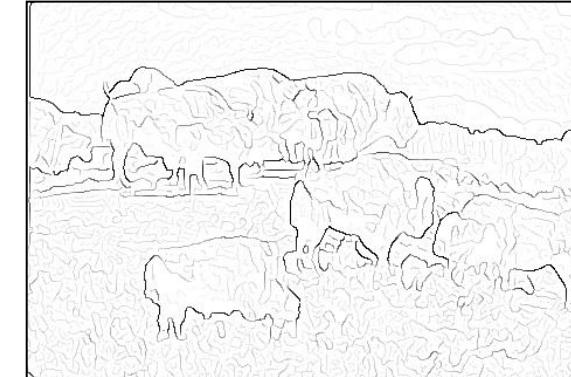
image



human segmentation



gradient magnitude

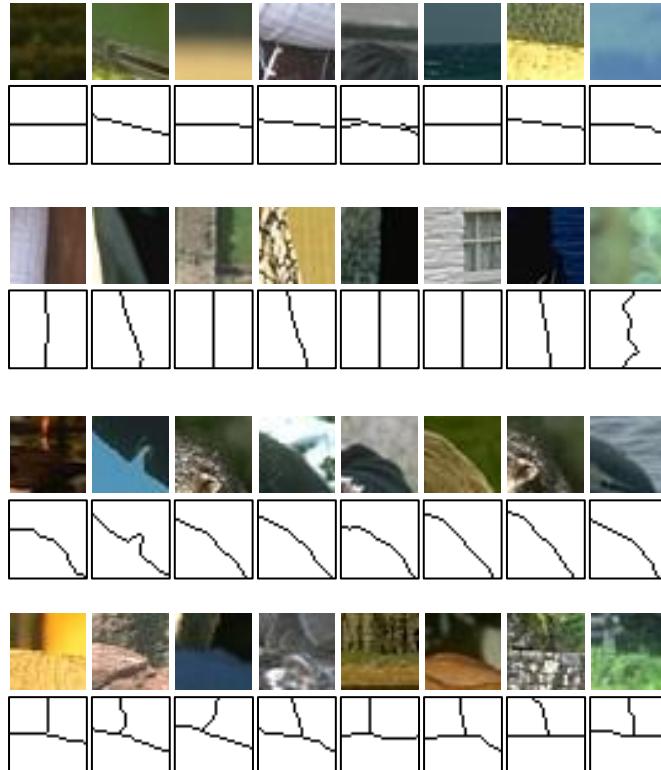


Berkeley segmentation database:

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

# Data-Driven Edge Detection

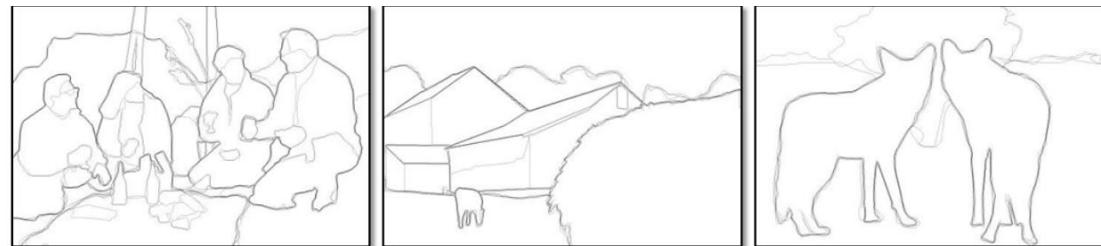
Training data



Input images



Ground truth



Output



# CORNERS

Slides are from Svetlana Lzebnik

# Feature Extraction: Corners



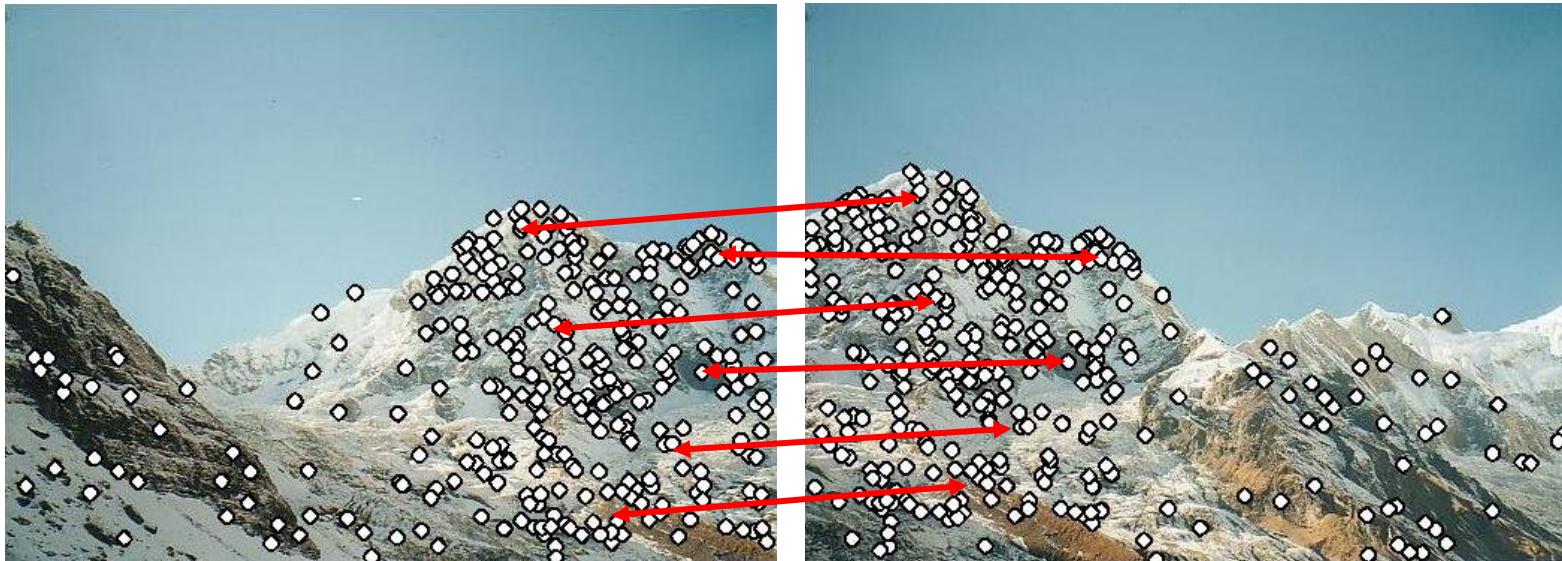
# Why Extract Features?

- 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?

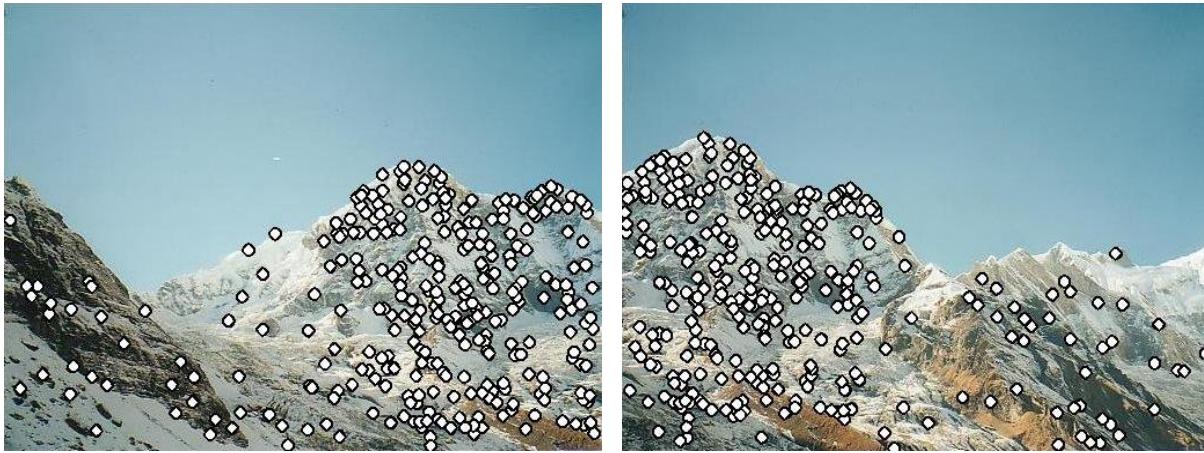


Step 1: extract features

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

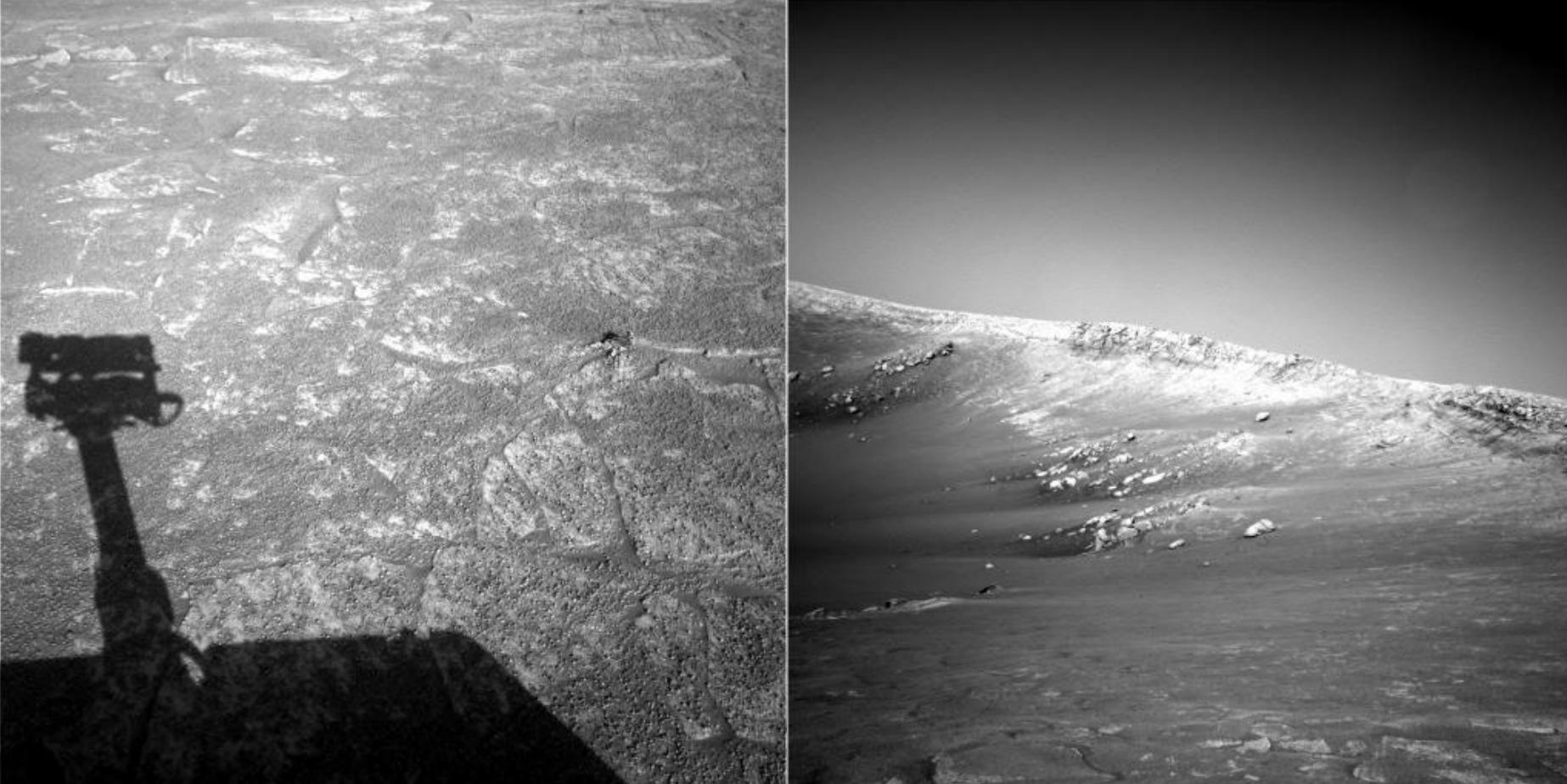
# Applications

Feature points are used for:

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

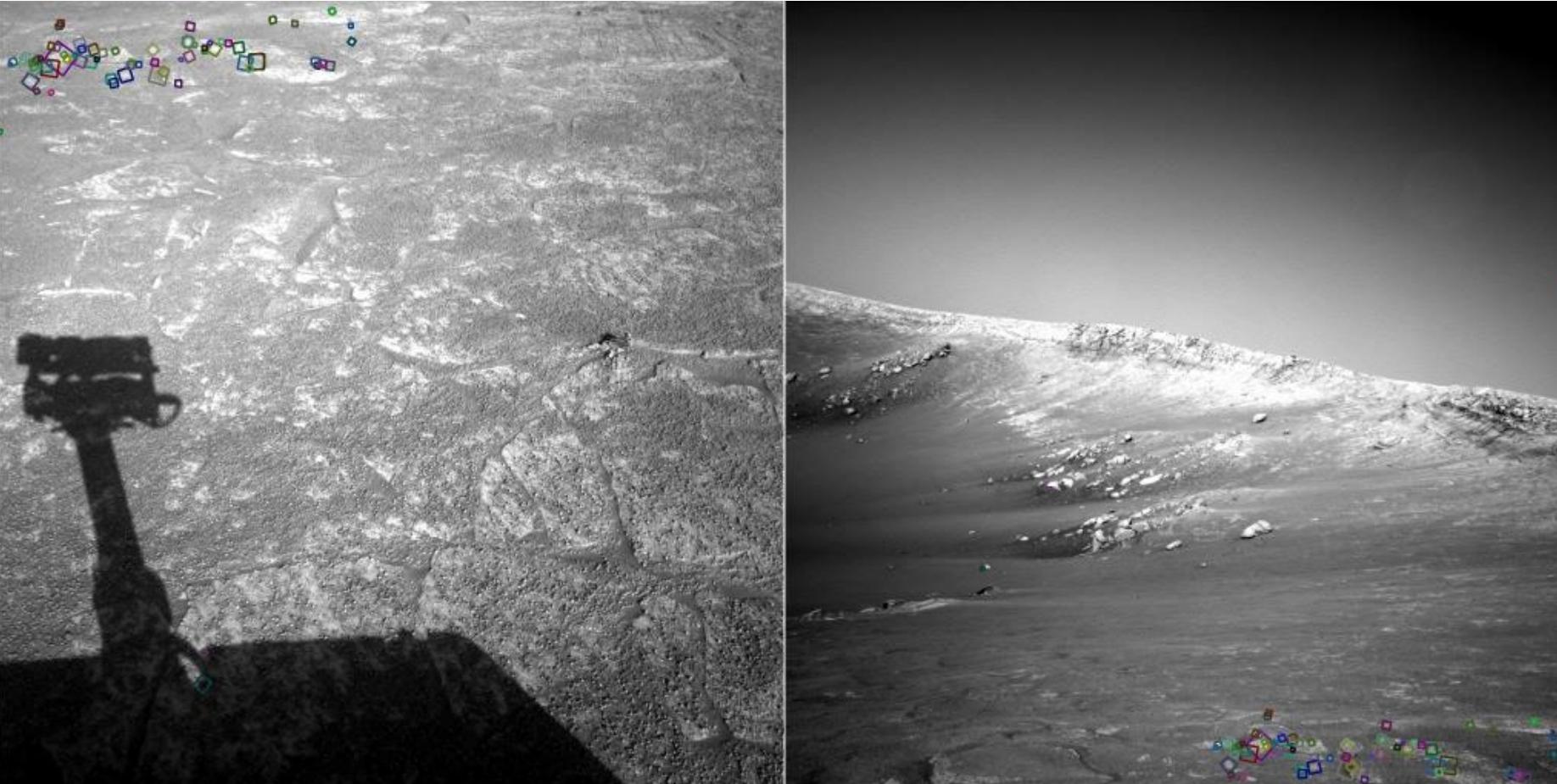


# A Hard Feature Matching Problem



NASA Mars Rover images

# Answer Below (look for tiny colored squares...)

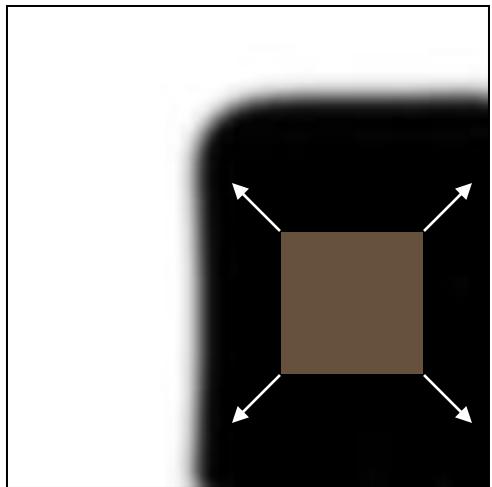


NASA Mars Rover images  
with SIFT feature matches  
Figure by Noah Snavely

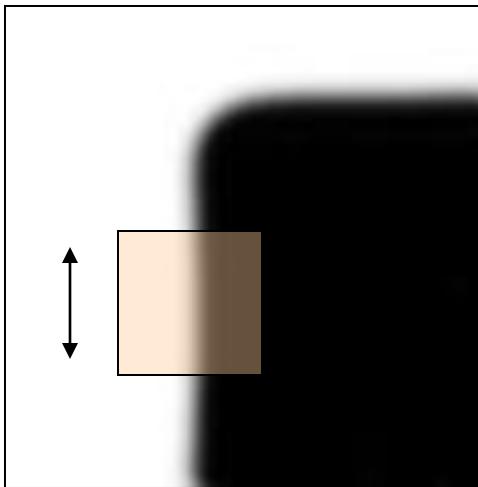
# Corner Detection: Basic Idea

We should easily recognize the point by looking through a small window

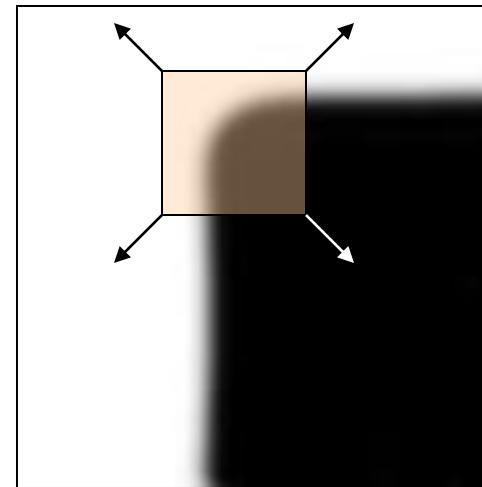
Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:  
no change in  
all directions



“edge”:  
no change  
along the edge  
direction

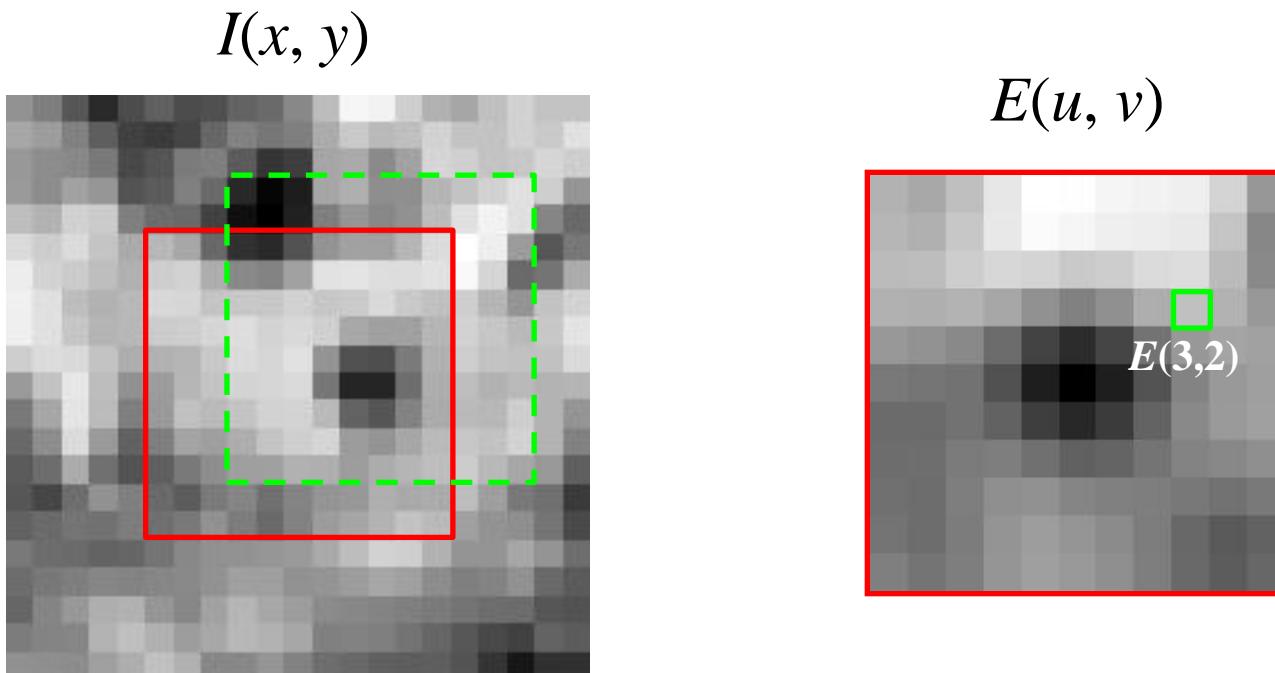


“corner”:  
significant  
change in all  
directions

# Corner Detection: Mathematics

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

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

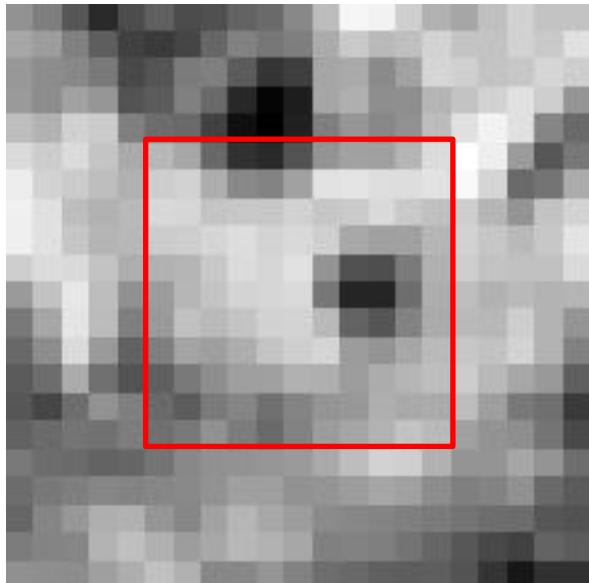


# Corner Detection: Mathematics

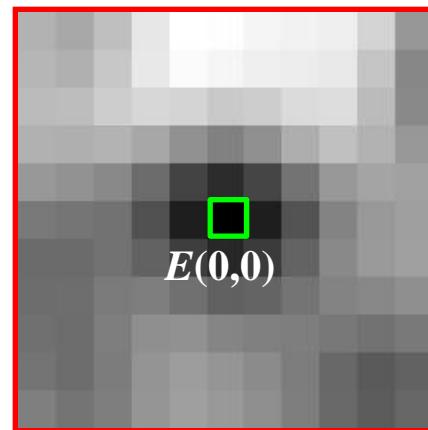
Change in appearance of window  $W$  for the shift  $[u, v]$ :

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

$I(x, y)$



$E(u, v)$



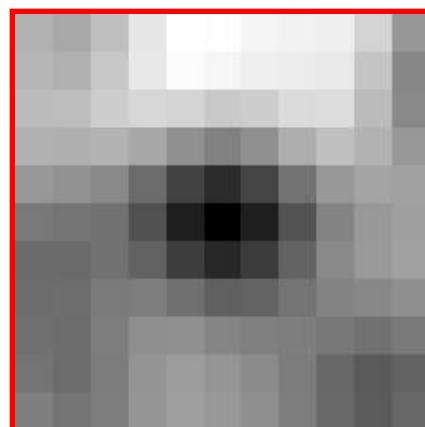
# Corner Detection: Mathematics

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

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

We want to find out how this function behaves for small shifts

$$E(u, v)$$



# Corner Detection: Mathematics

- First-order Taylor approximation for small motions  $[u, v]$ :

$$I(x + u, y + v) \approx I(x, y) + I_x u + I_y v$$

- Let's plug this into  $E(u, v)$ :

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

# Corner Detection: Mathematics

The quadratic approximation can be written as

$$E(u, v) \approx [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

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

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

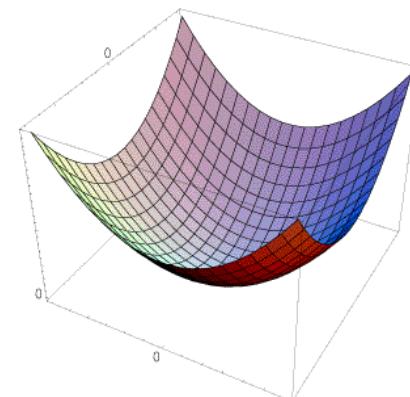
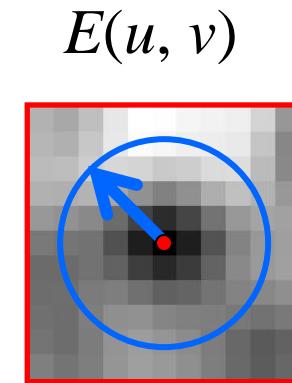
(the sums are over all the pixels in the window  $W$ )

# Interpreting the Second Moment Matrix

- The surface  $E(u, v)$  is locally approximated by a quadratic form. Let's try to understand its shape.
- Specifically, in which directions does it have the smallest/greatest change?

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

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



# Interpreting the Second Moment Matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

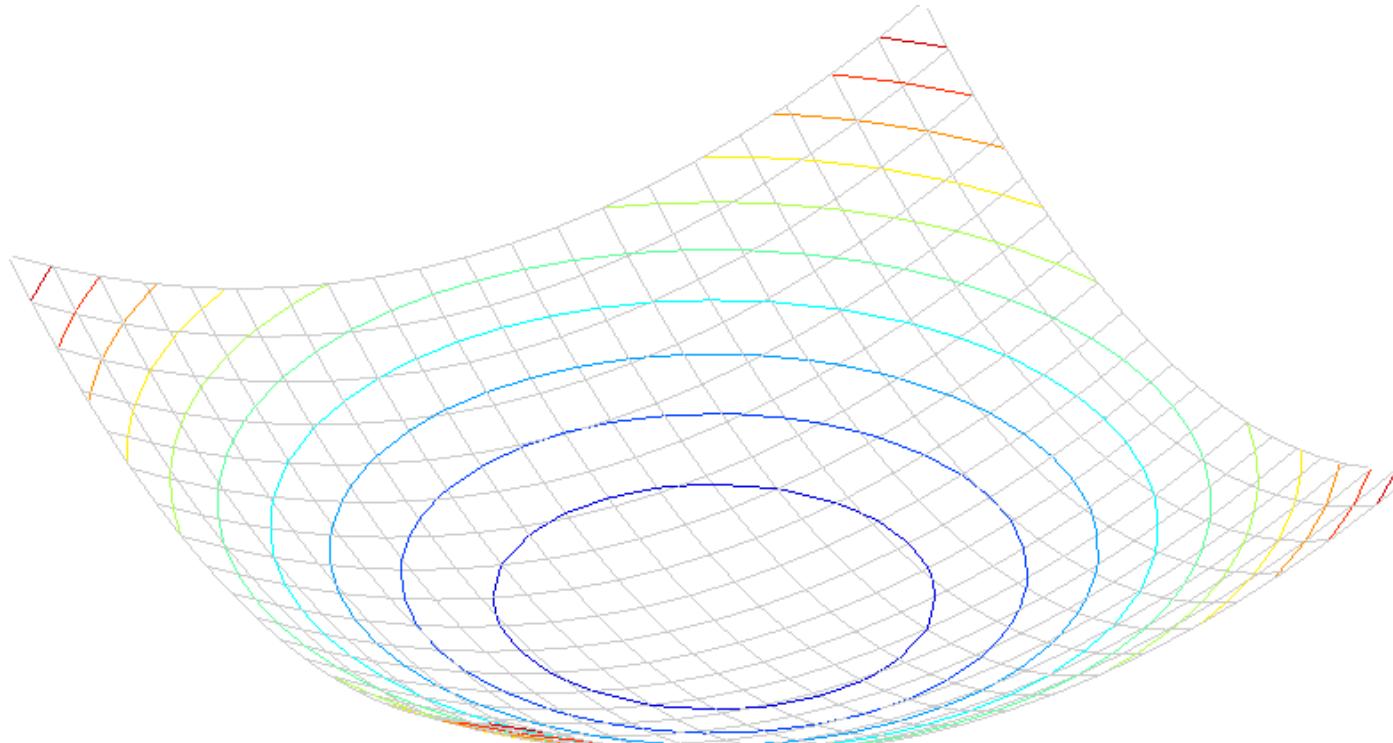
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

If either  $a$  or  $b$  is close to 0, then this is **not** a corner, so look for locations where both are large.

# 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.



# Interpreting the Second Moment Matrix

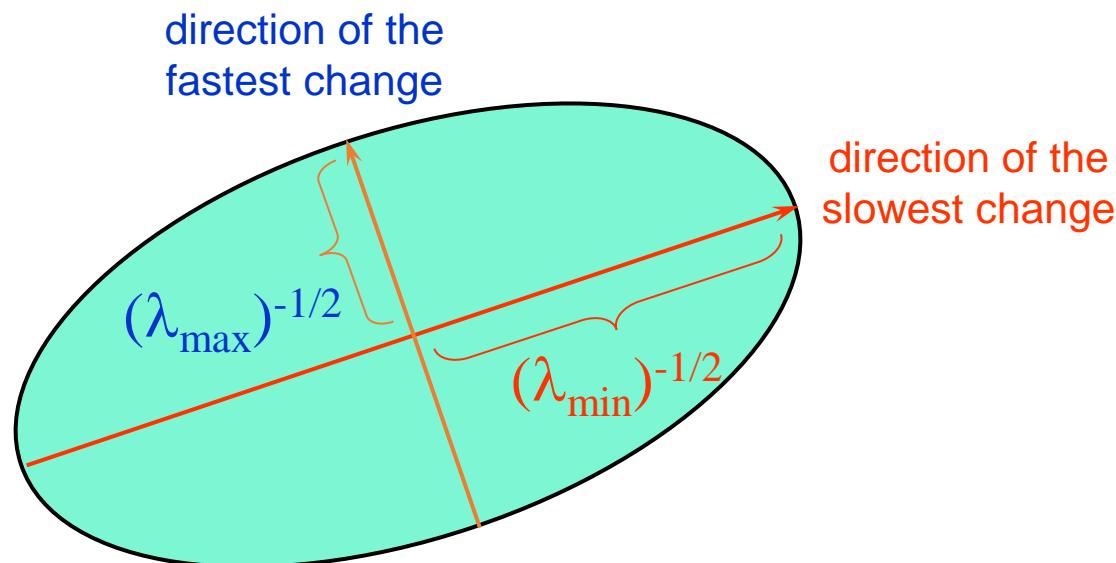
Consider a horizontal “slice” of  $E(u, v)$ :  $[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.

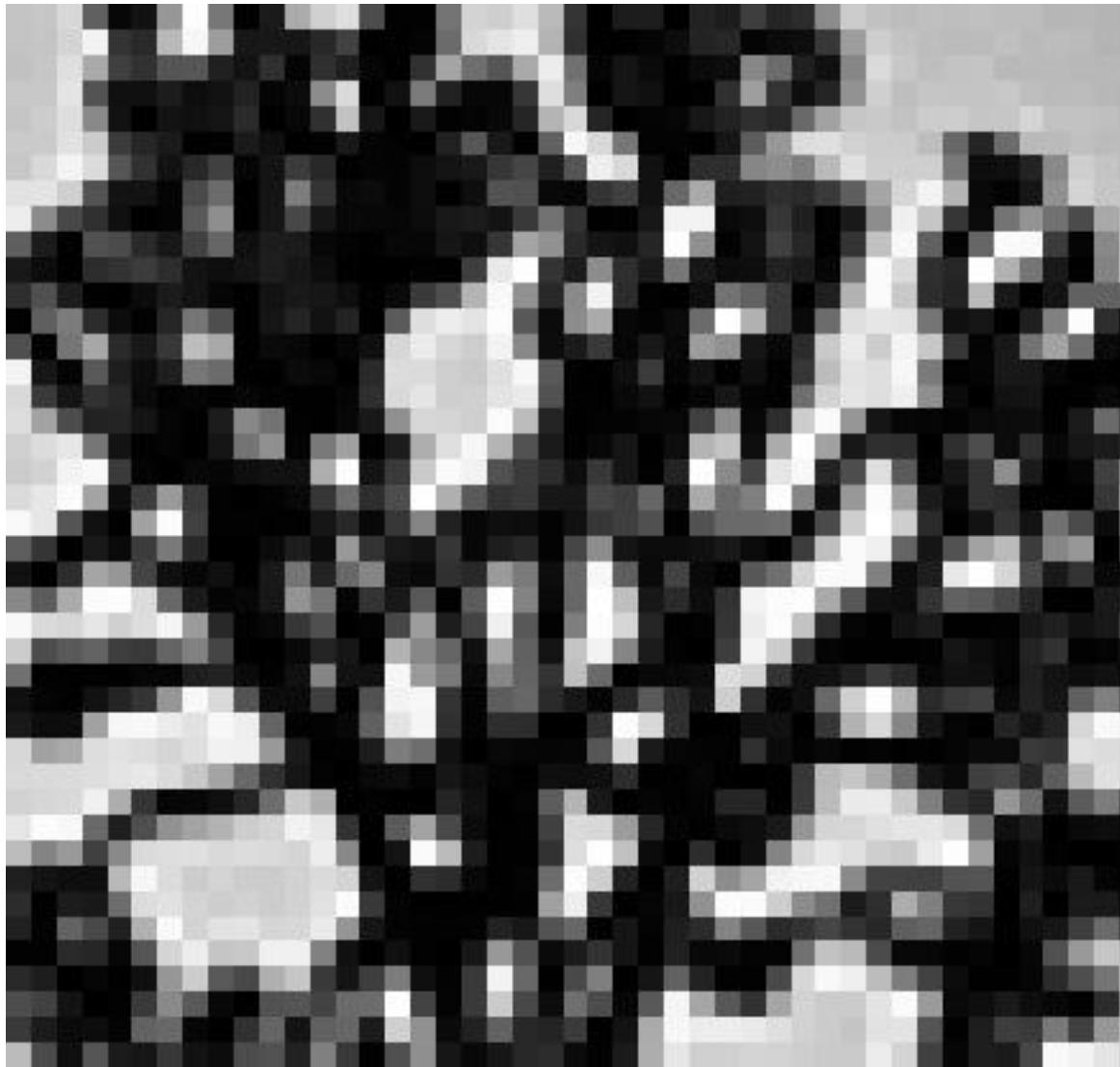
Diagonalization of  $M$ :

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

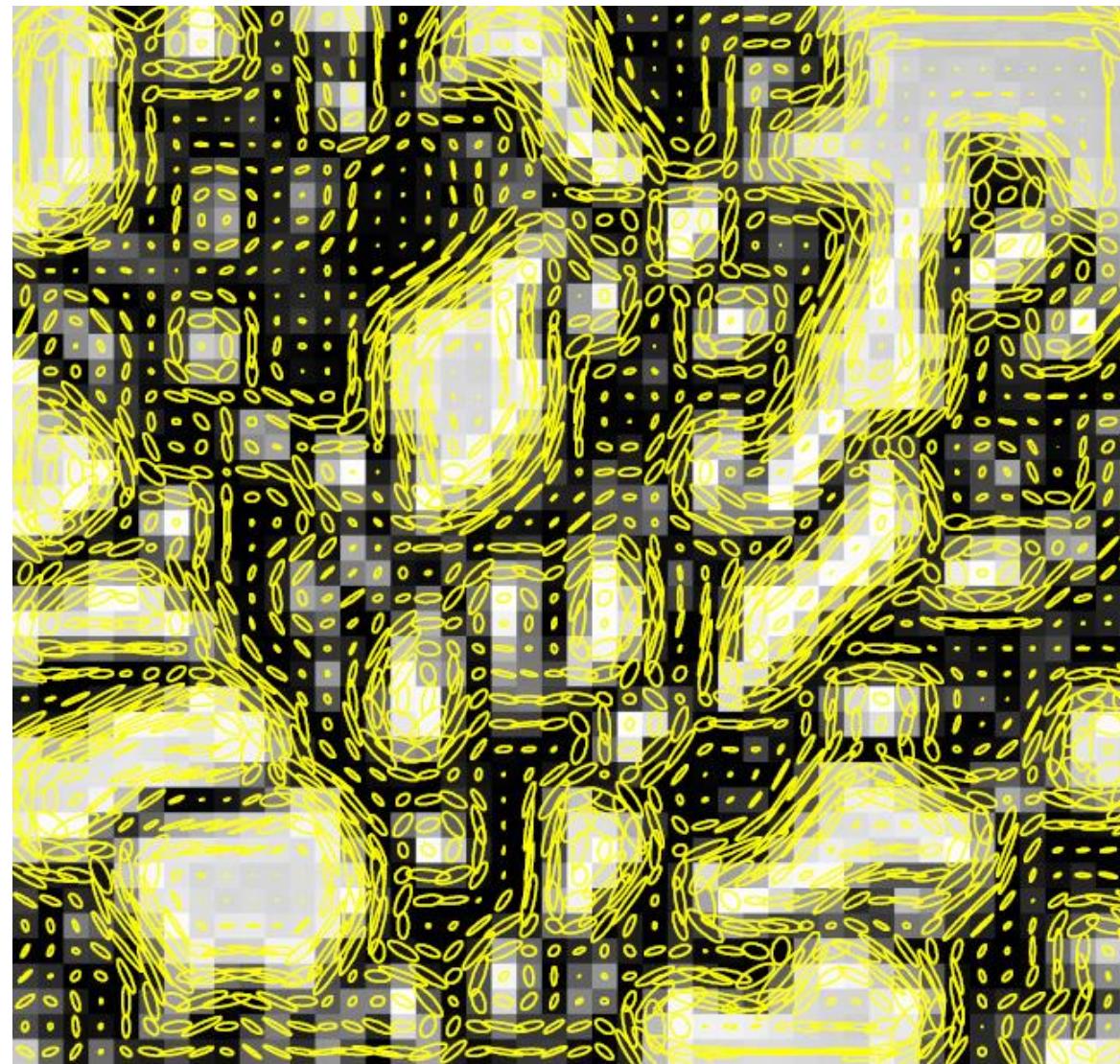
The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by  $R$



# Visualization of Second Moment Matrices

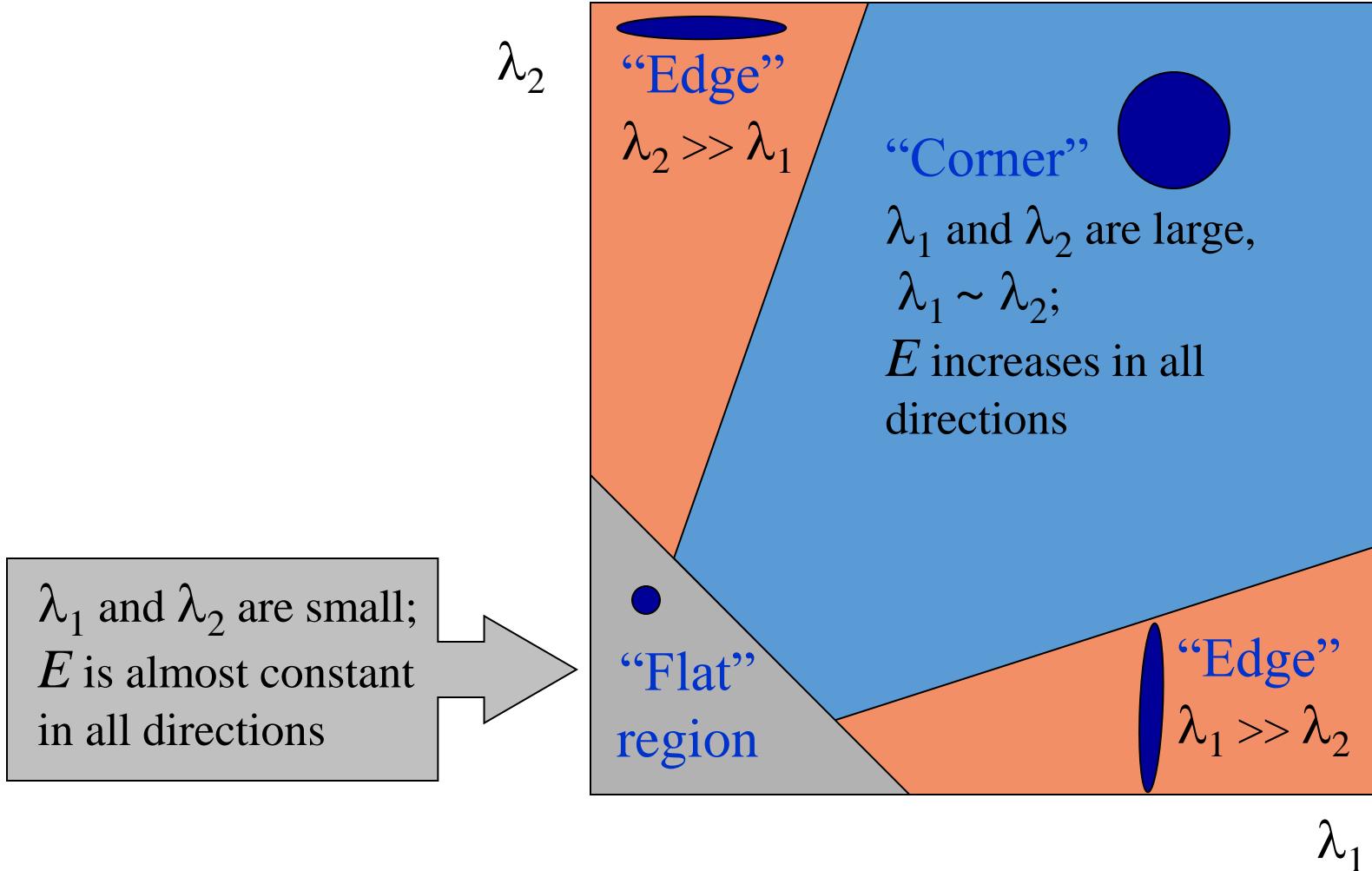


# Visualization of Second Moment Matrices



# Interpreting the Eigenvalues

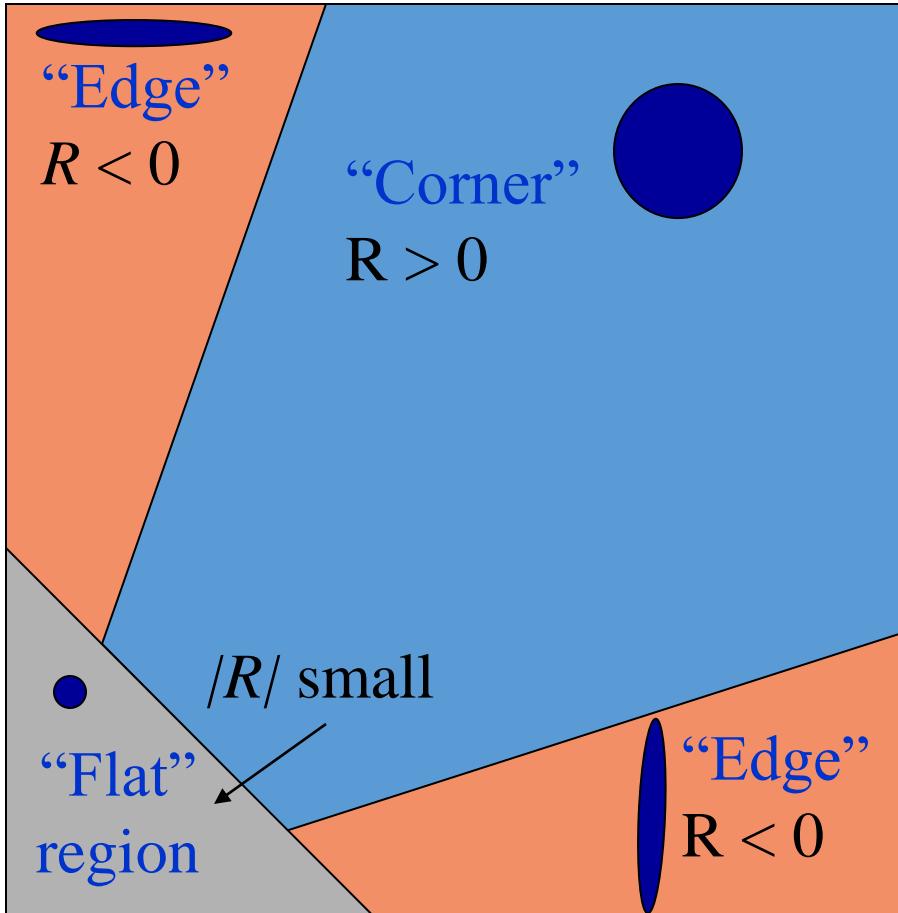
Classification of image points using eigenvalues of  $M$ :



# Corner Response Function

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

$\alpha$ : constant (0.04 to 0.06)



# The Harris Corner Detector

1. Compute partial derivatives at each pixel
2. Compute second moment matrix  $M$  in a Gaussian window around each pixel:

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

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

# 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  $R$

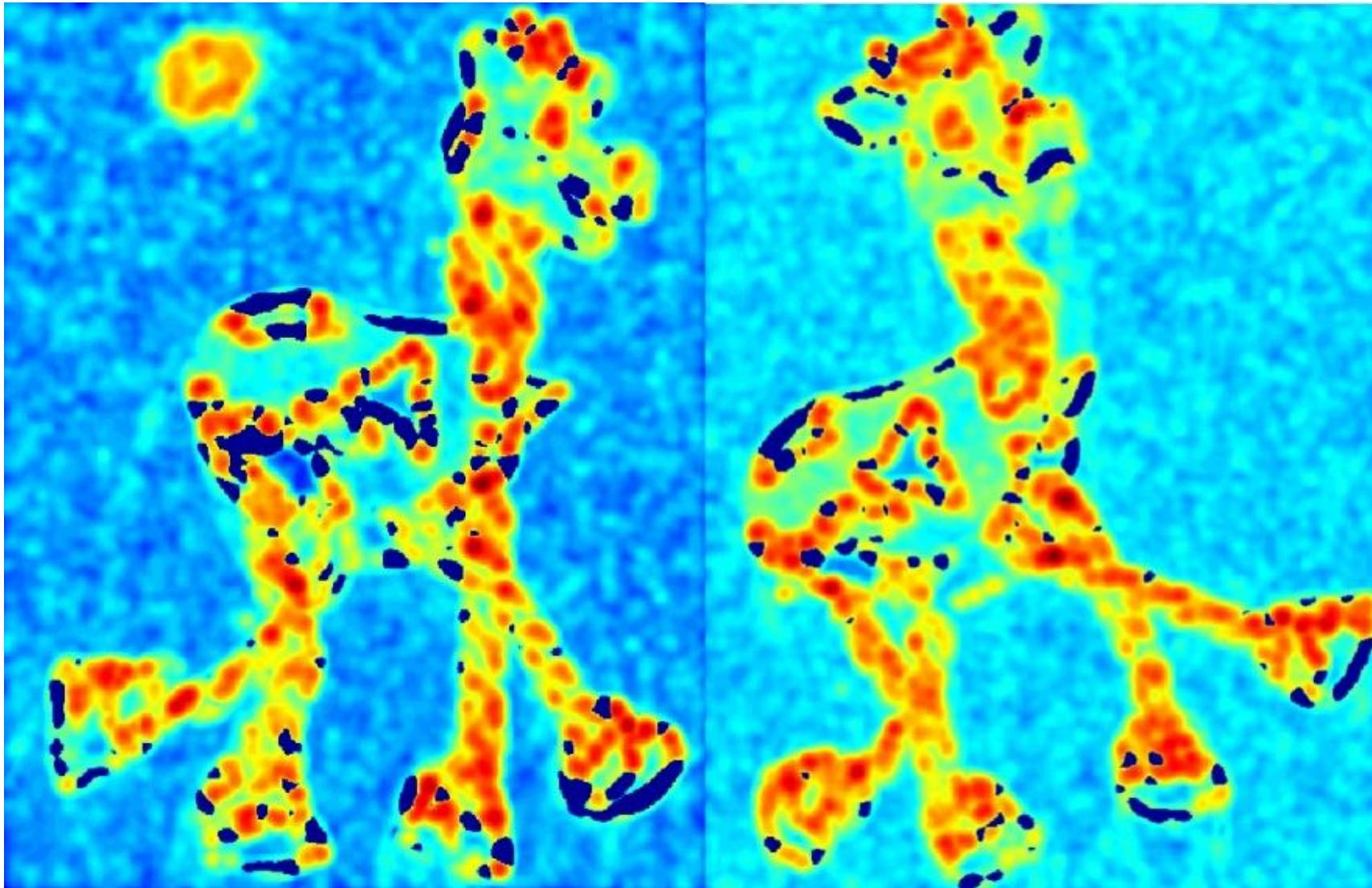
C.Harris and M.Stephens. "[A Combined Corner and Edge Detector.](#)"  
*Proceedings of the 4th Alvey Vision Conference:* pages 147—151, 1988.

# Harris Detector: Steps



# Harris Detector: Steps

Compute corner response  $R$



# 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.

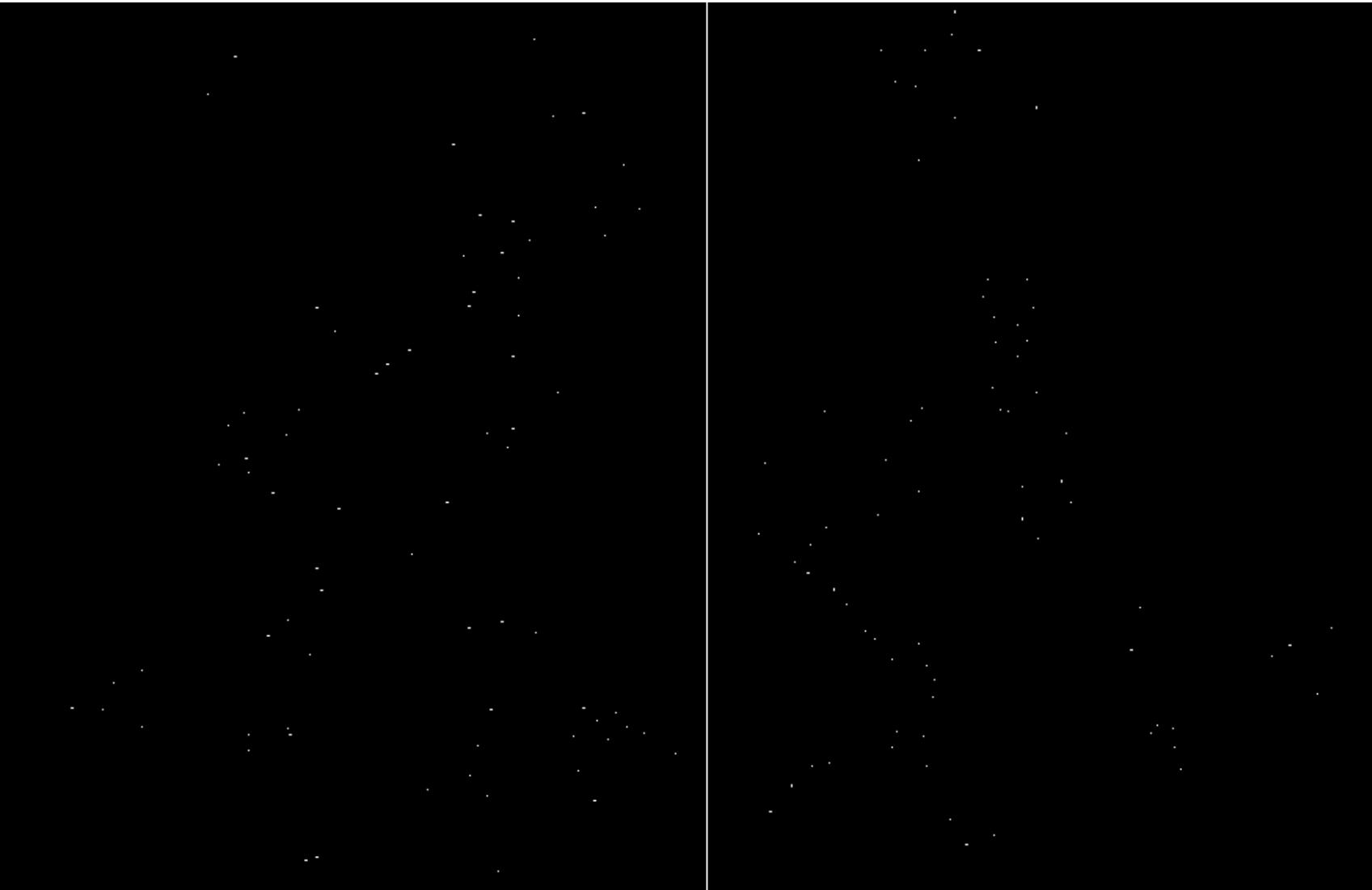
# Harris Detector: Steps

Find points with large corner response:  $R > \text{threshold}$



# Harris Detector: Steps

Take only the points of local maxima of  $R$



# 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



# Activity

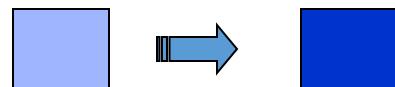
Discuss in pairs and decide:

I may randomly ask a pair to respond.

Is Harris corner detection invariant to

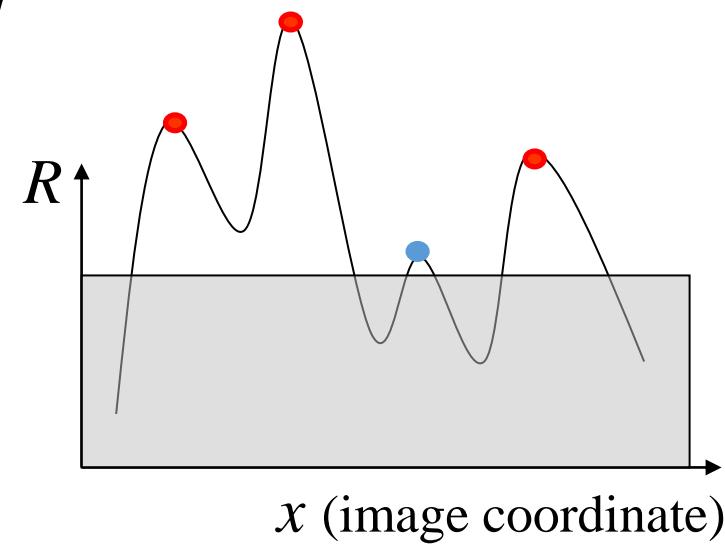
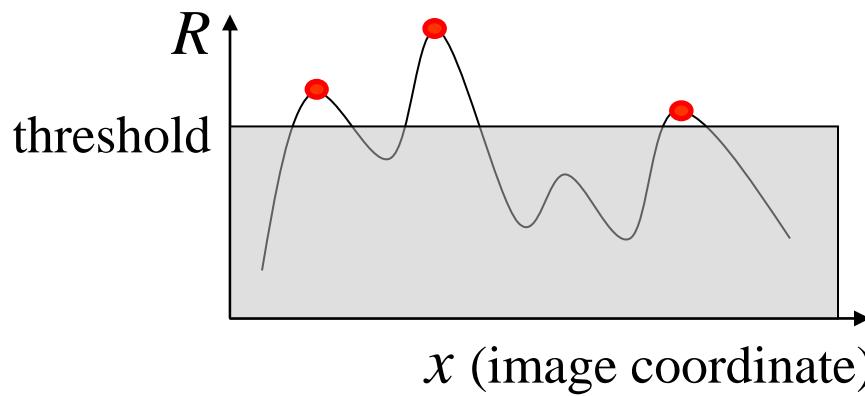
1. Affine intensity changes  $I \rightarrow aI + b$
2. Image translation
3. Image rotation
4. Image scaling

# Affine Intensity Change



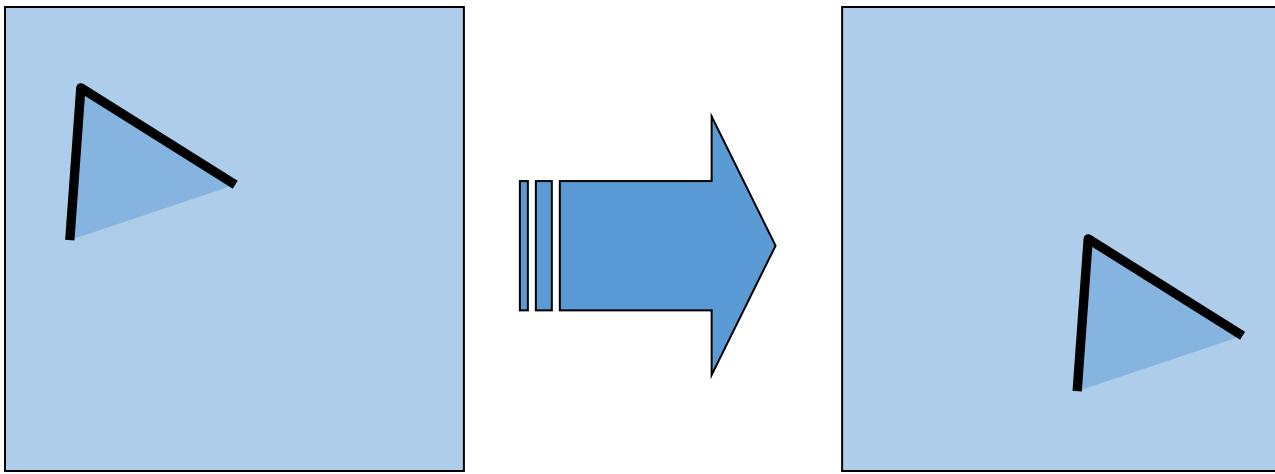
$$I \rightarrow aI + b$$

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



*Partially invariant to affine intensity change*

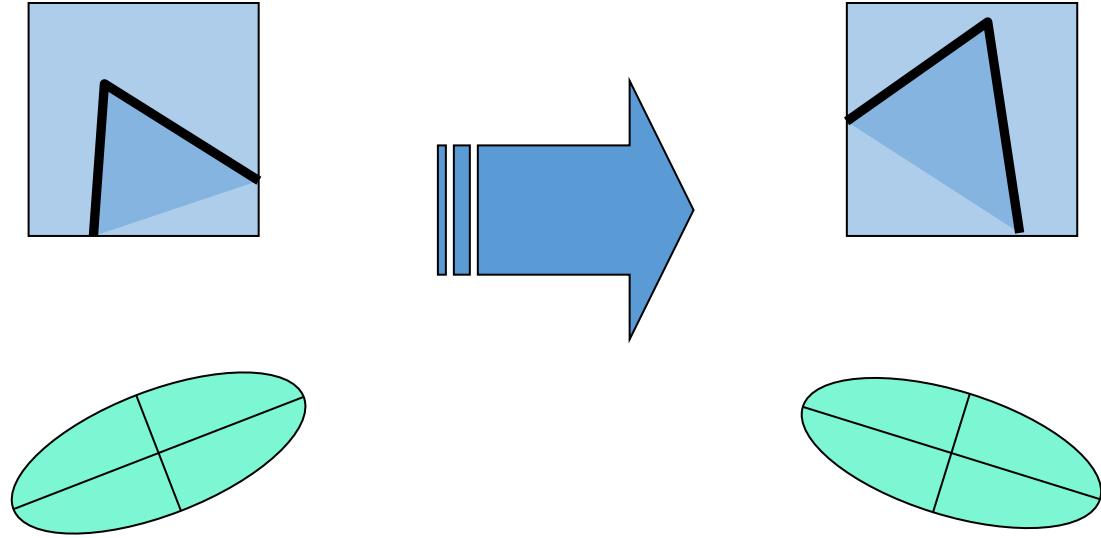
# Image Translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

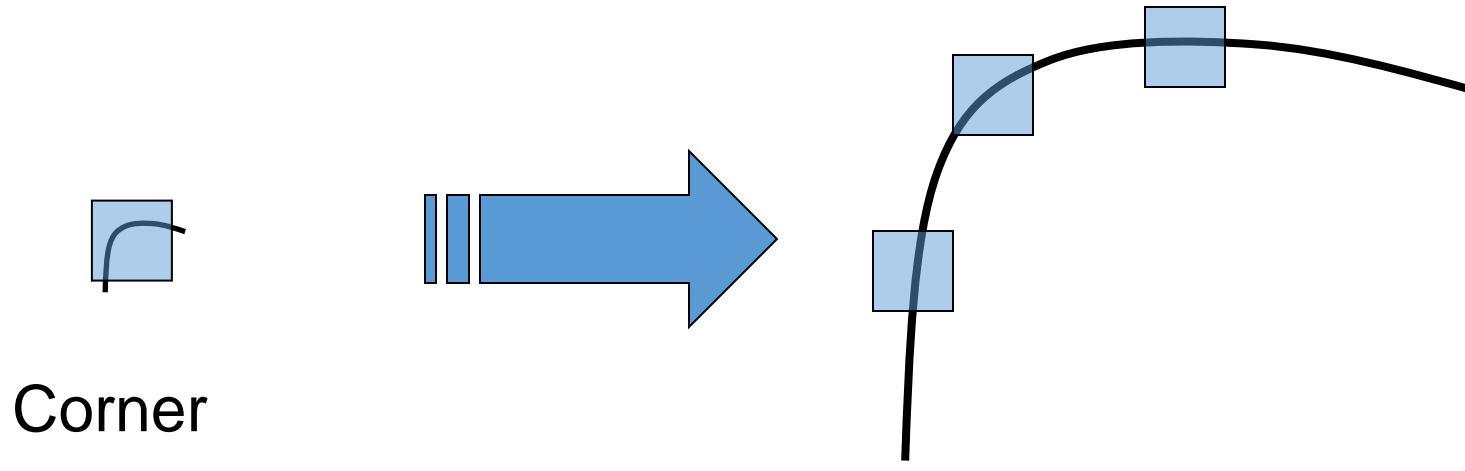
# Image Rotation



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

Corner location is covariant w.r.t. rotation

# Scaling



Corner

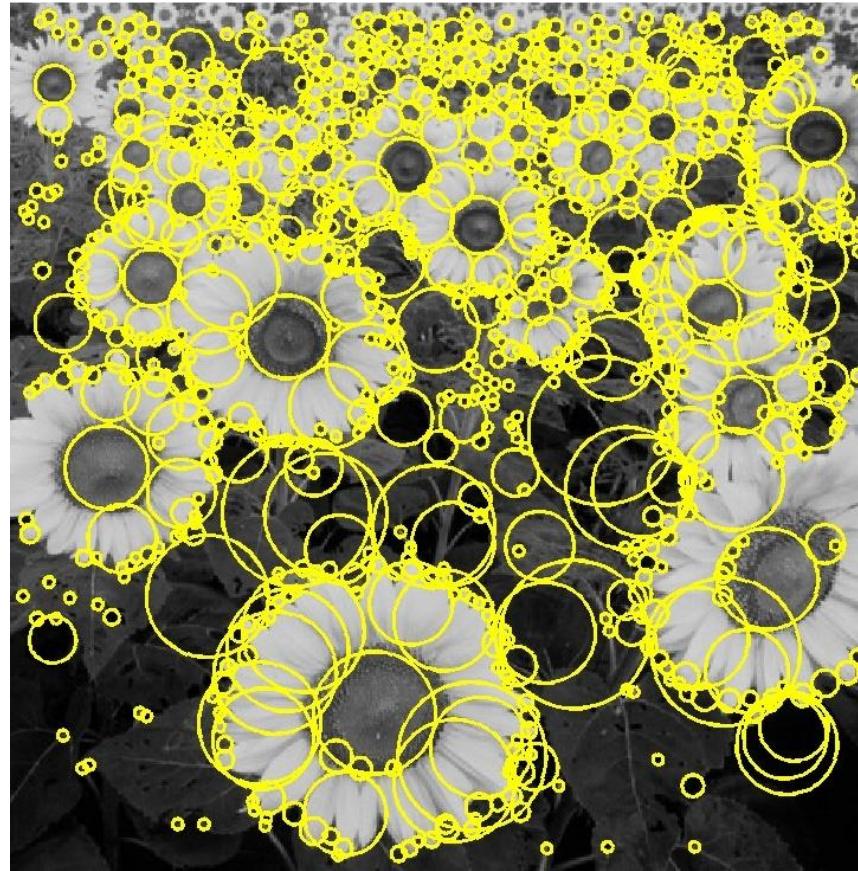
All points will be  
classified as  
edges

Corner location is not covariant to scaling!

# BLOB DETECTION

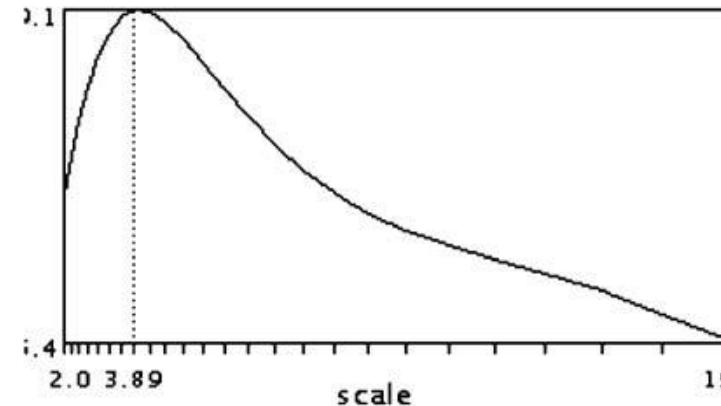
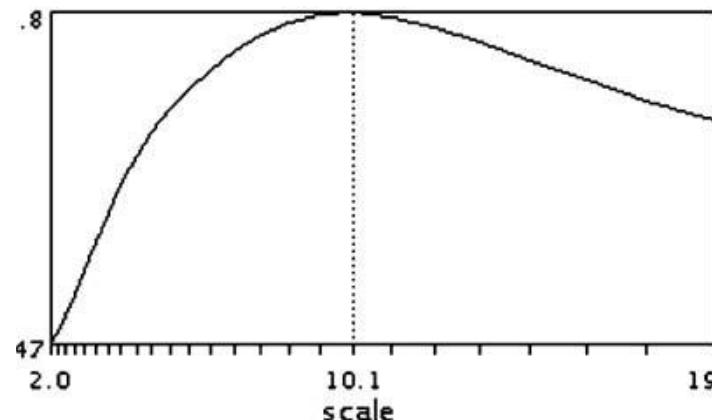
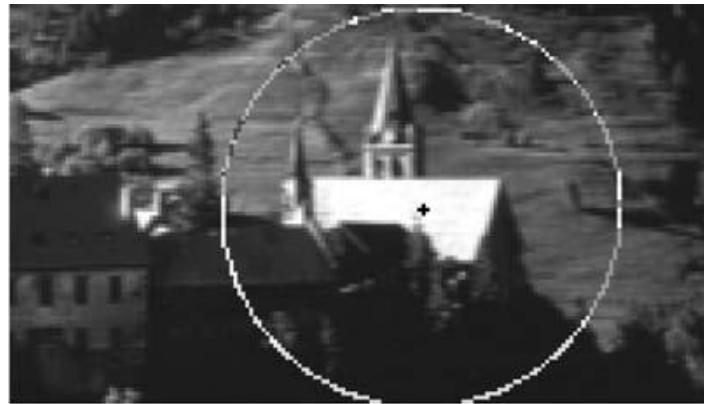
Slides from Svetlana Lazebnik

# Blob Detection



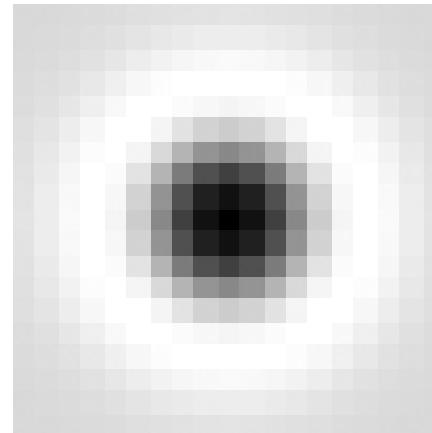
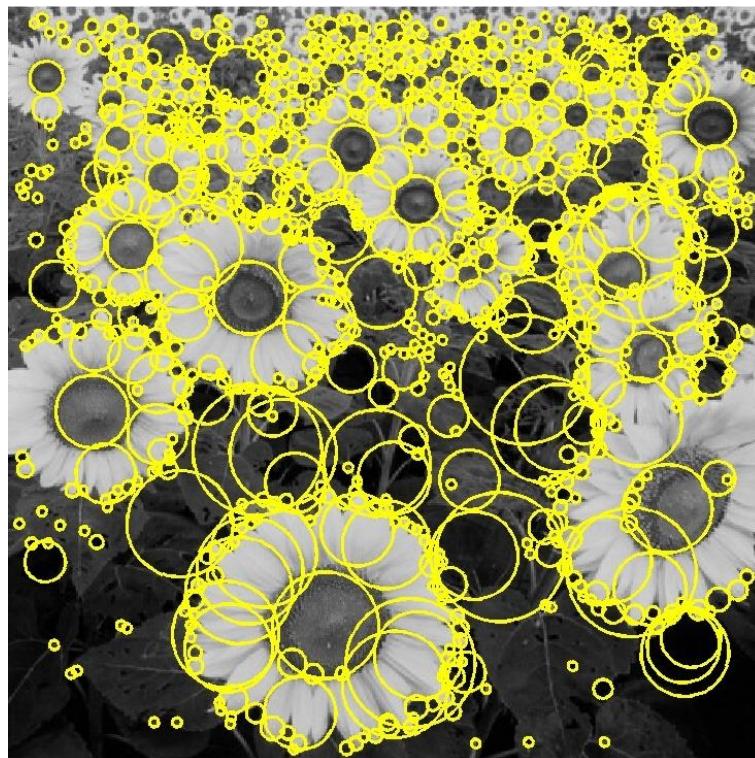
# Feature Detection with Scale Selection

We want to extract features with characteristic scale that is *covariant* with the image transformation



# Blob Detection: Basic idea

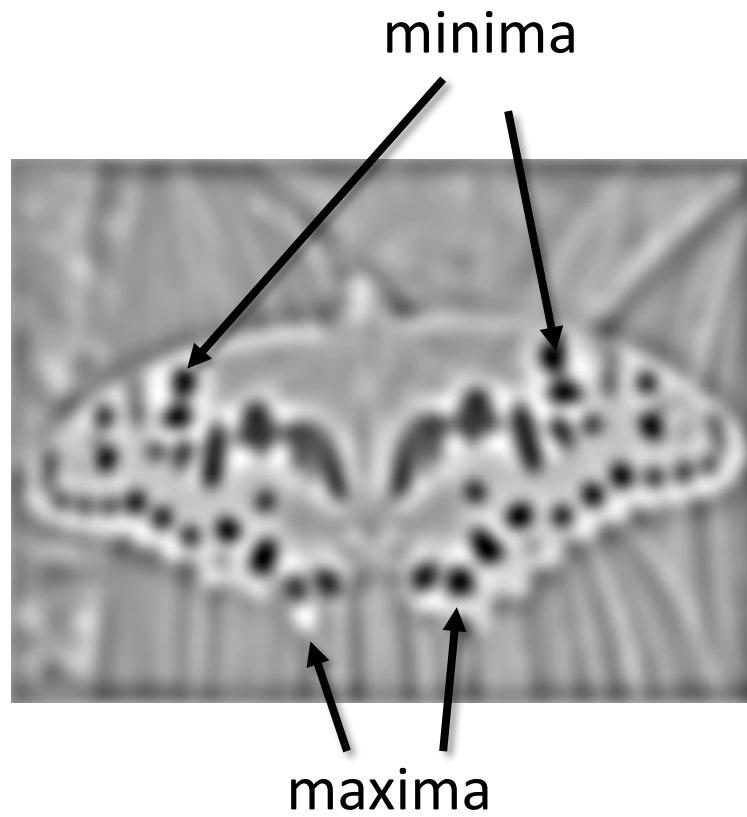
- To detect blobs, convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*.



# Blob Detection: Basic Idea



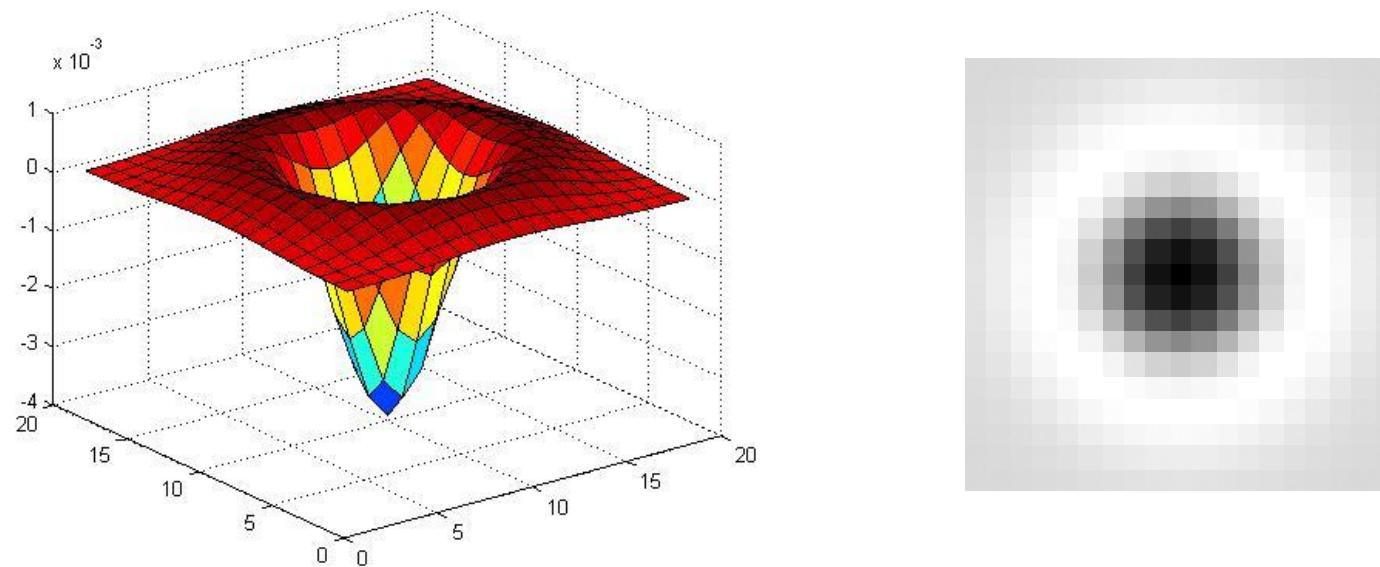
$$* \quad \bullet =$$



Find maxima *and minima* of blob filter response  
in space *and scale*

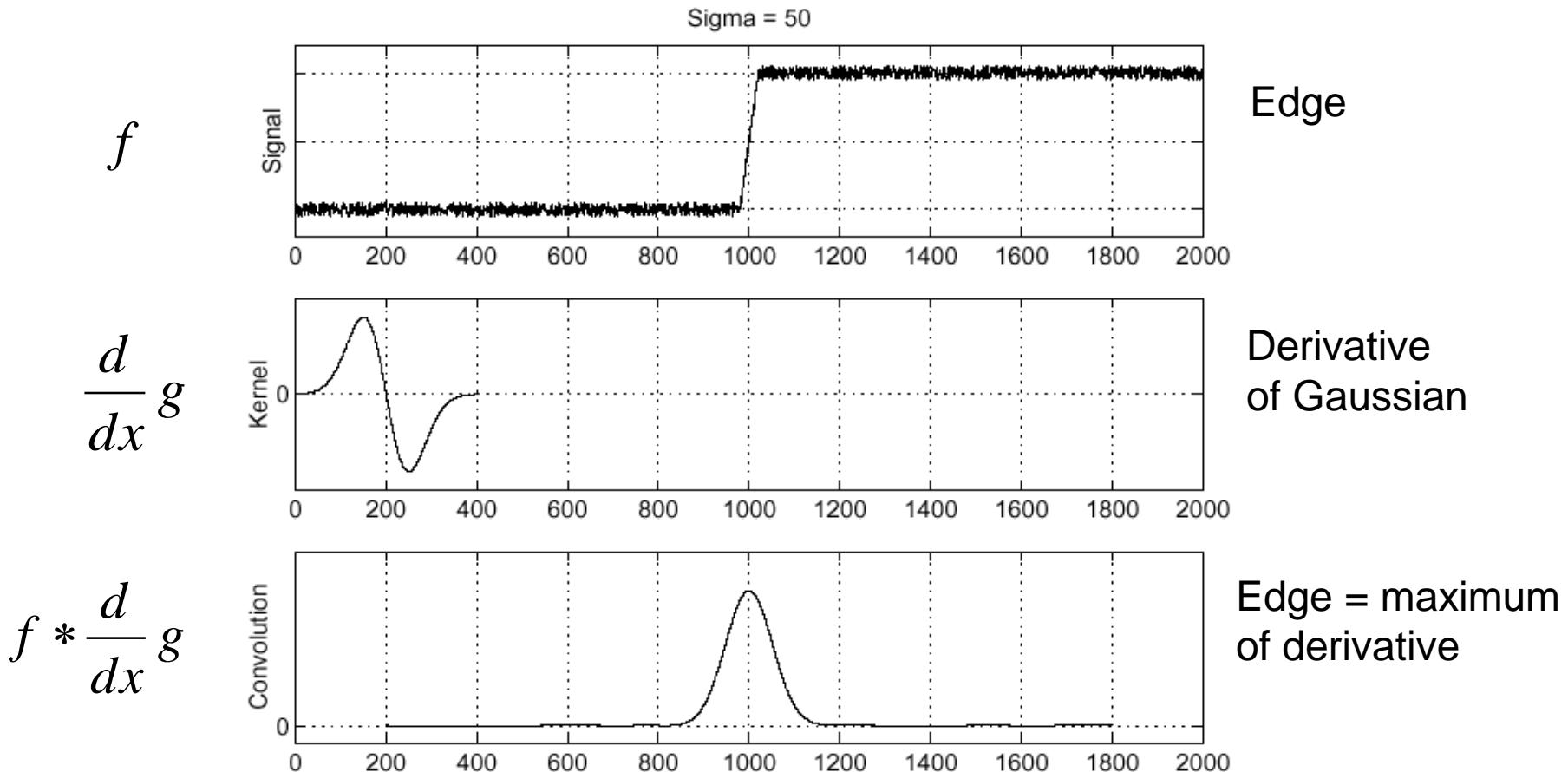
# Blob Filter

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

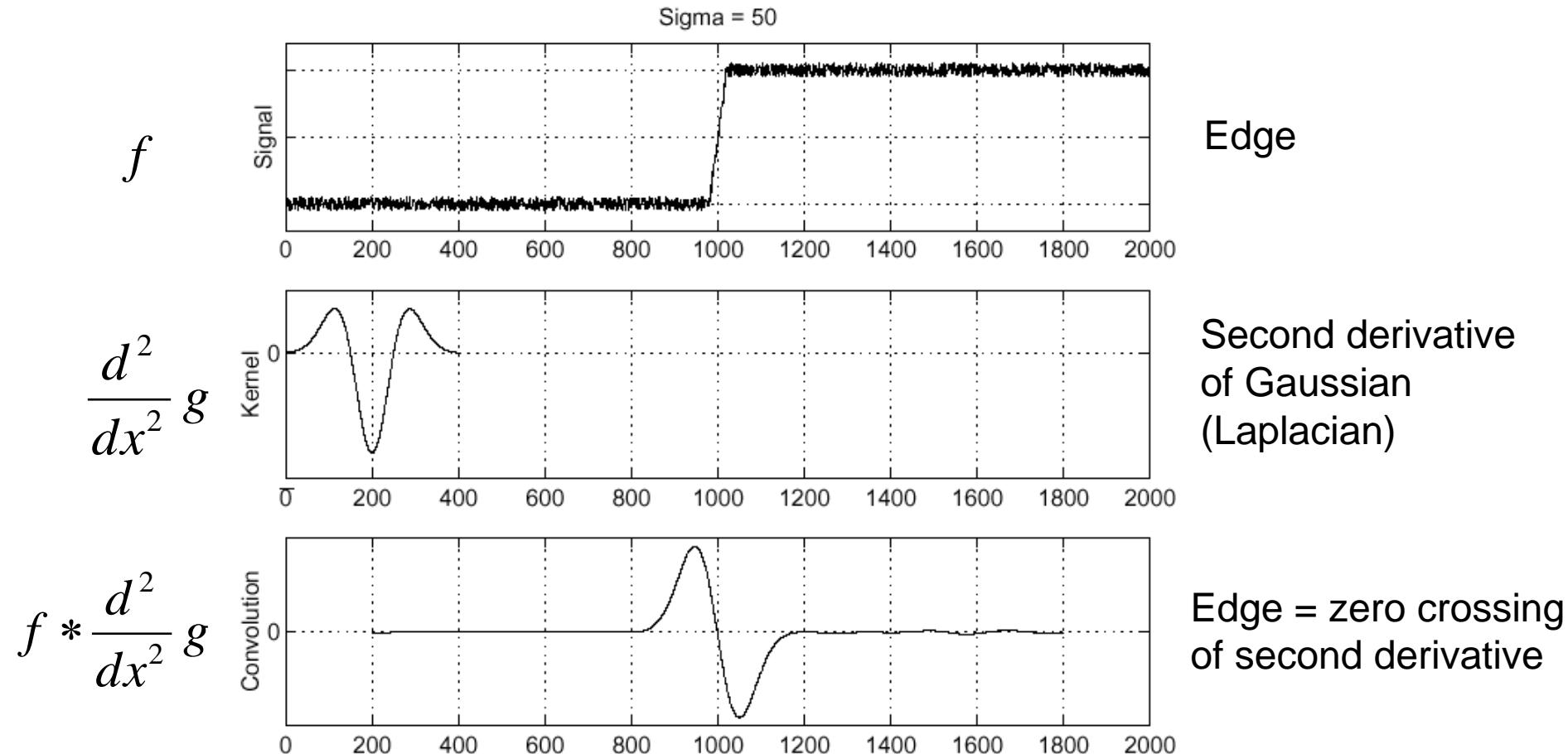


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

# Recall: Edge Detection

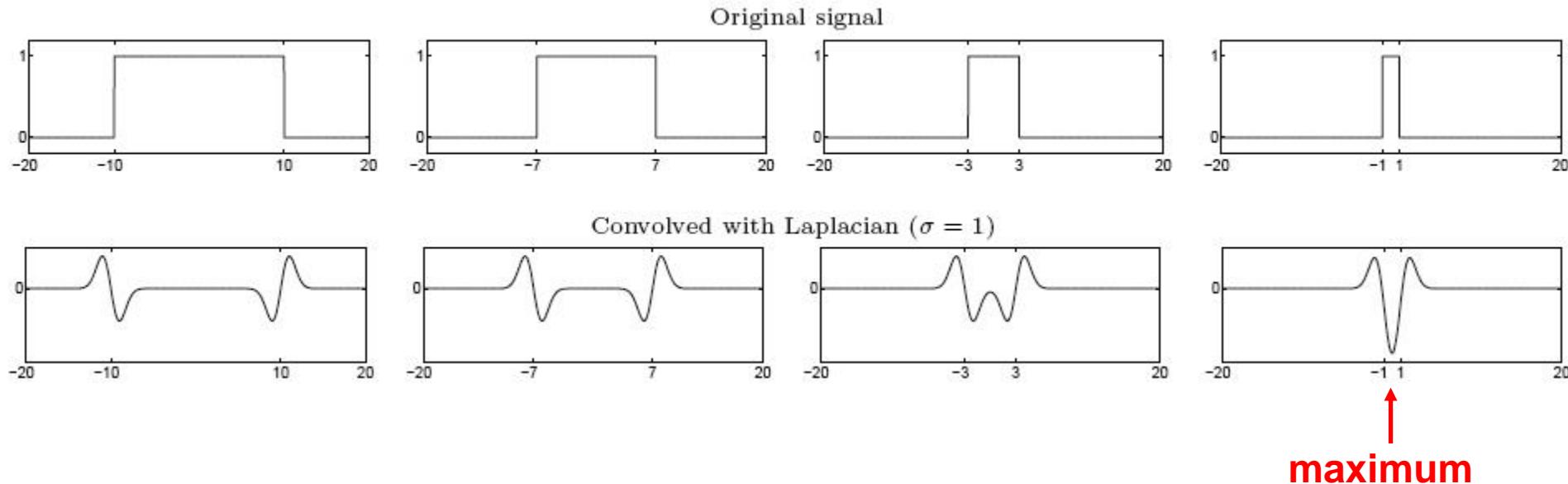


# Edge Detection, Take 2



# From Edges to Blobs

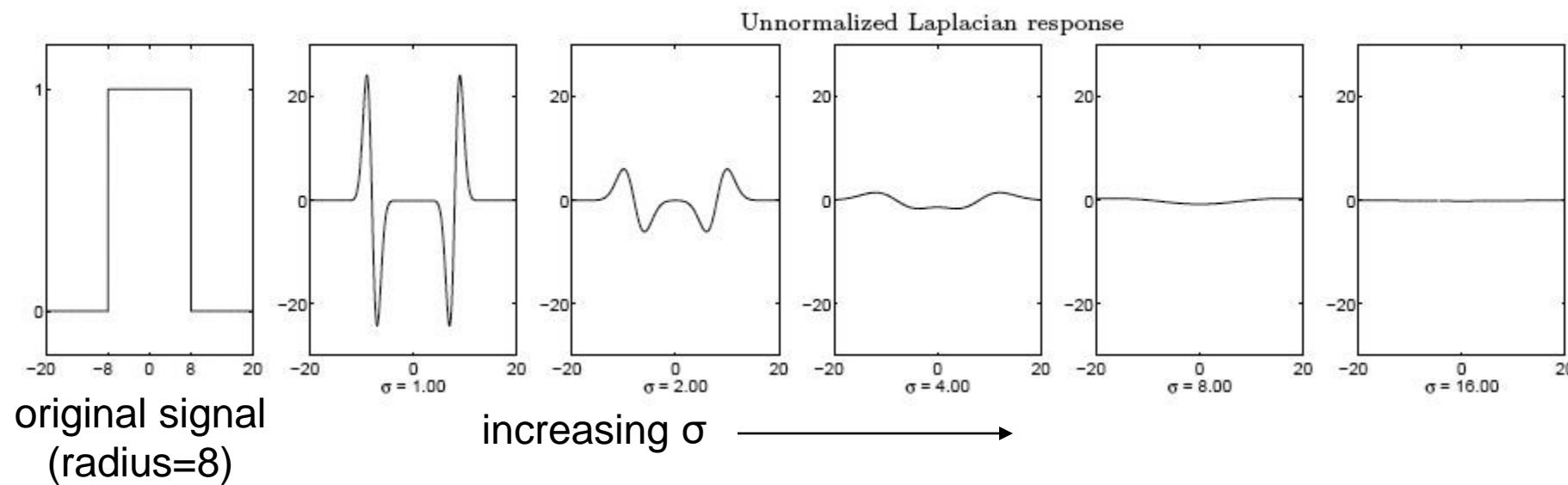
- Edge = ripple
- Blob = superposition of two ripples



**Spatial selection:** the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is “matched” to the scale of the blob

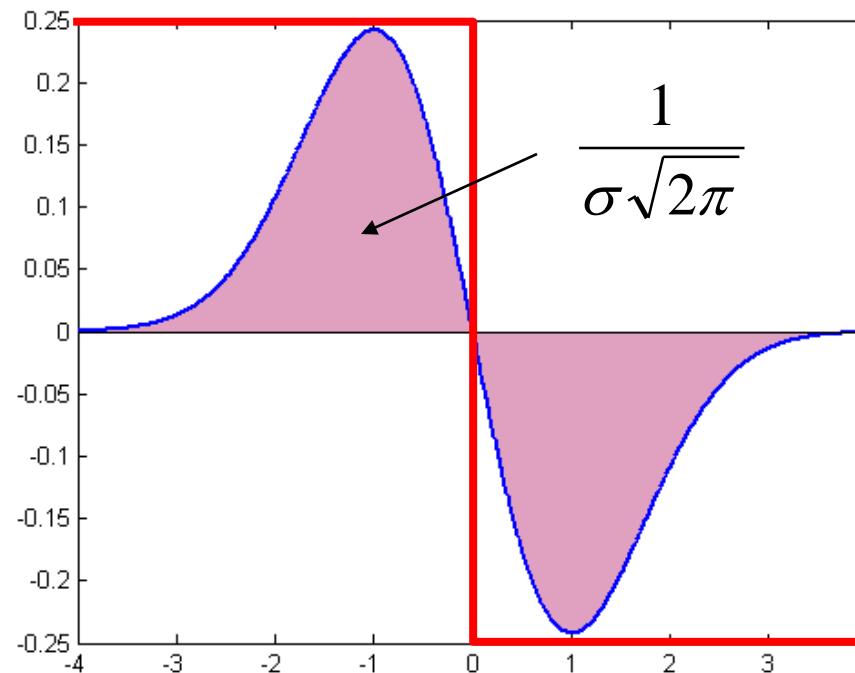
# Scale Selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response
- However, Laplacian response decays as scale increases:



# Scale Normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as  $\sigma$  increases

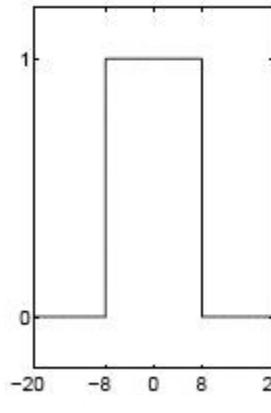


# Scale Normalization

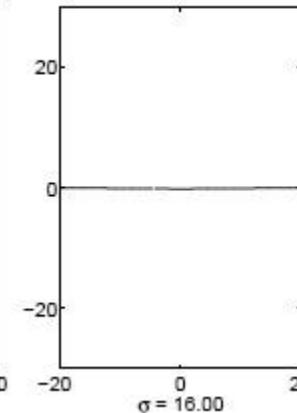
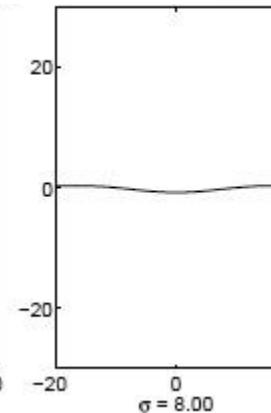
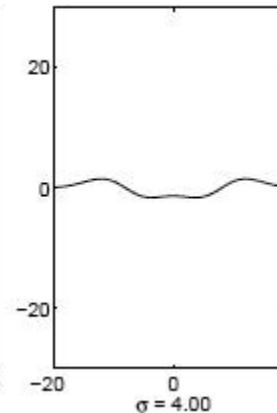
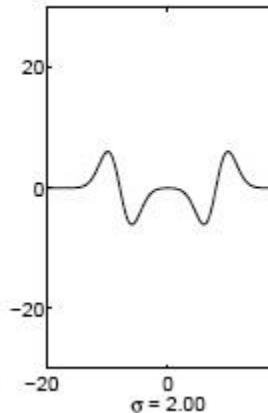
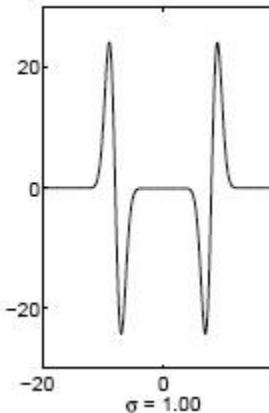
- The response of a derivative of Gaussian filter to a perfect step edge decreases as  $\sigma$  increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by  $\sigma$
- Laplacian is the second Gaussian derivative, so it must be multiplied by  $\sigma^2$

# Effect of Scale Normalization

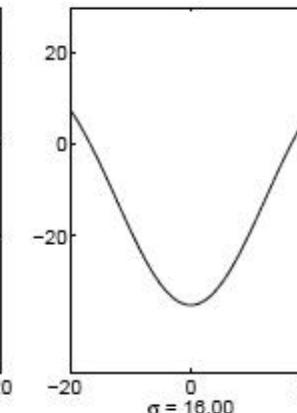
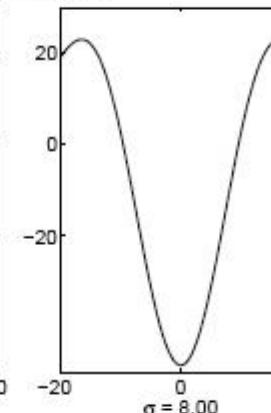
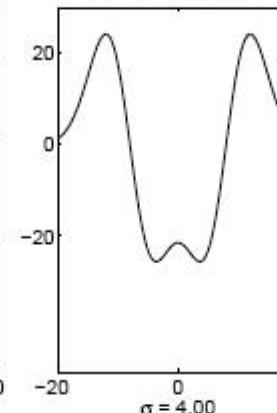
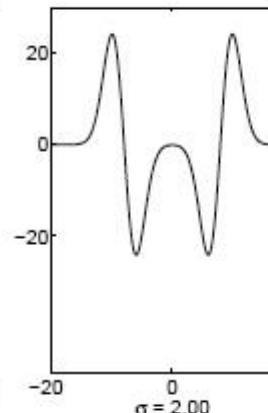
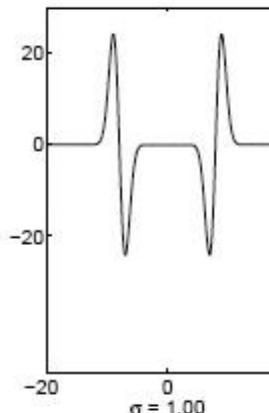
Original signal



Unnormalized Laplacian response



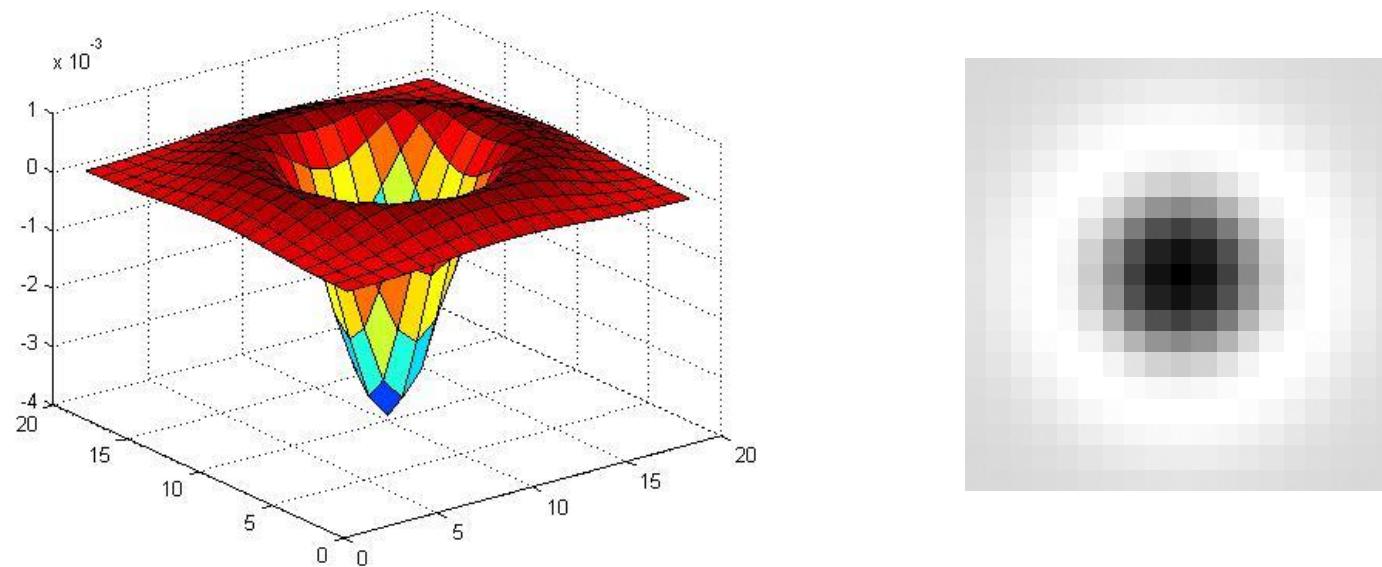
Scale-normalized Laplacian response



maximum

# Blob Detection in 2D

Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

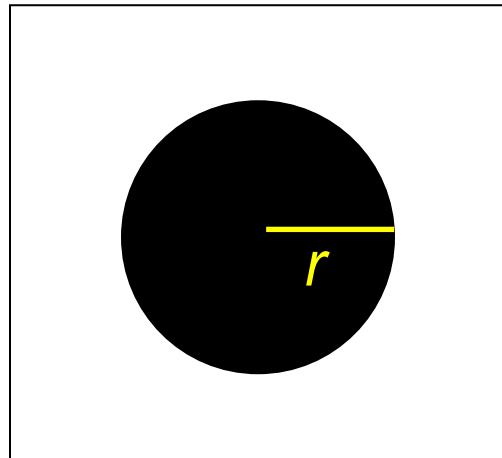


Scale-normalized:

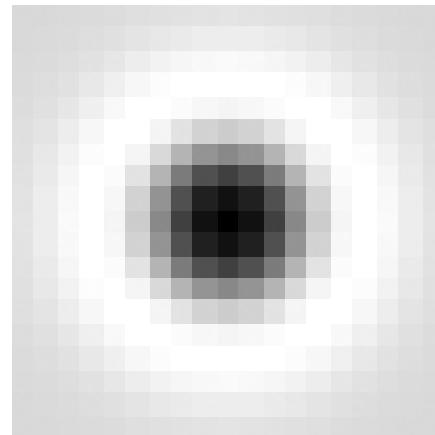
$$\nabla_{\text{norm}}^2 g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

# Scale Selection

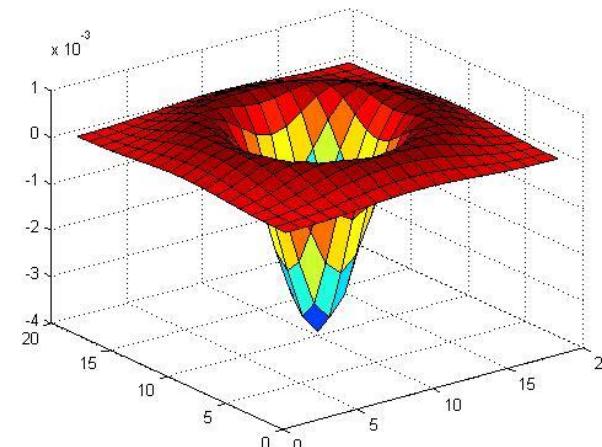
At what scale does the Laplacian achieve a maximum response to a binary circle of radius  $r$ ?



image



Laplacian



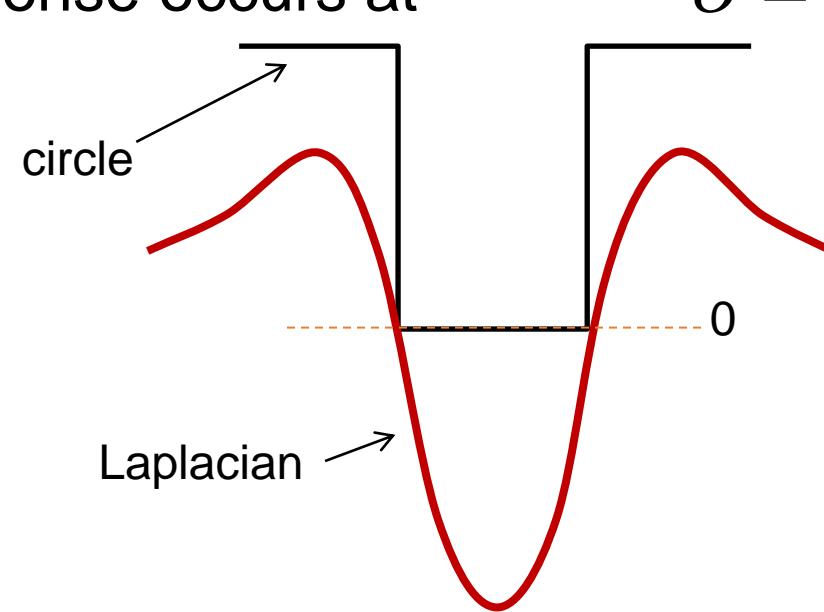
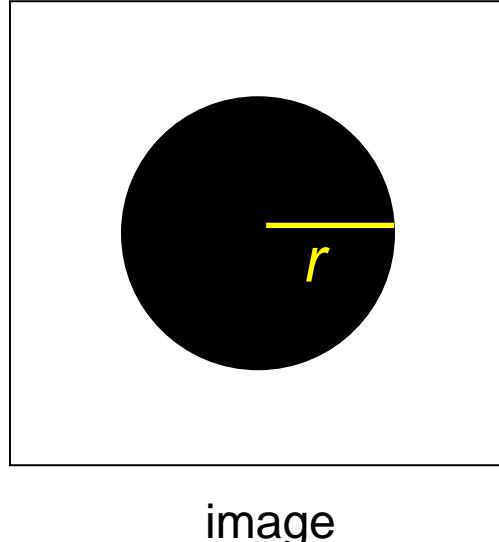
# Scale Selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius  $r$ ?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

$$(x^2 + y^2 - 2\sigma^2) e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at

$$\sigma = r / \sqrt{2}.$$



$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

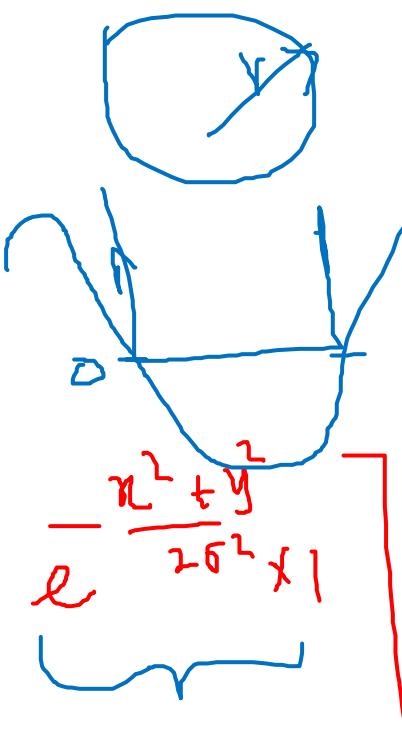
$$\frac{\partial g}{\partial x} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \times \left( -\frac{x}{\sigma^2} \right)$$

$$= \frac{-1}{2\pi\sigma^4} \frac{\pi}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial^2 g}{\partial x^2} = \frac{-1}{2\pi\sigma^4} \left[ \frac{\pi}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \times \left( -\frac{1}{\sigma^2} \right) + \frac{\pi^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \times 1 \right]$$

$$= \frac{\pi^2}{2\pi\sigma^4} \left[ \frac{x^2}{\sigma^2} - 1 \right]$$

$$\frac{\partial^2 g}{\partial y^2} = \checkmark$$



$$\nabla^2_{norm} = \sigma^2 \left[ \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right]$$

$$\nabla^2_{norm} =$$

$$= \frac{\pi^2 + y^2}{2\sigma^2} \left[ \frac{x^2}{\sigma^2} + \frac{y^2}{\sigma^2} - 1 \right]$$

$$\nabla^2_{norm} = D \quad ?$$

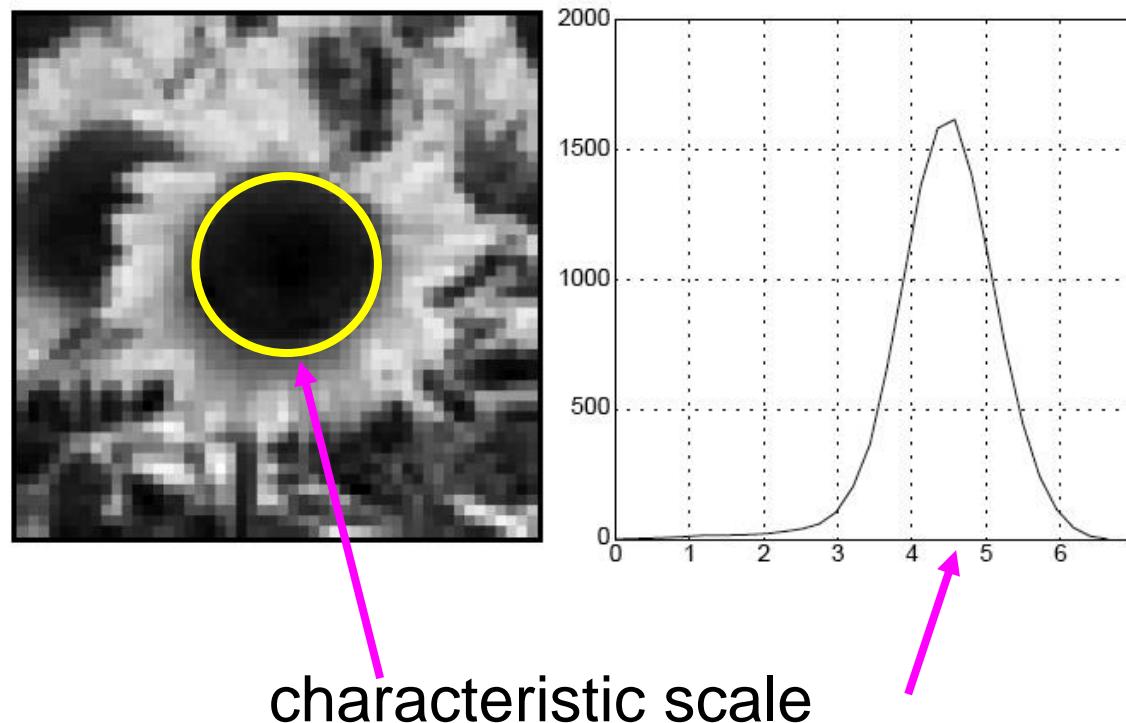
$$\Rightarrow \frac{\pi^2}{\sigma^2} + \frac{y^2}{\sigma^2} - 1 = D$$

$$\frac{\pi^2}{\sigma^2} = 1$$

$$\sigma = \sqrt{2\pi} \approx 2.5$$

# Characteristic Scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center



T. Lindeberg (1998). "[Feature detection with automatic scale selection.](#)"  
*International Journal of Computer Vision* **30** (2): pp 77--116.

# Scale-Space Blob Detector

1. Convolve image with scale-normalized Laplacian at several scales

# Scale-Space Blob Detector: Example



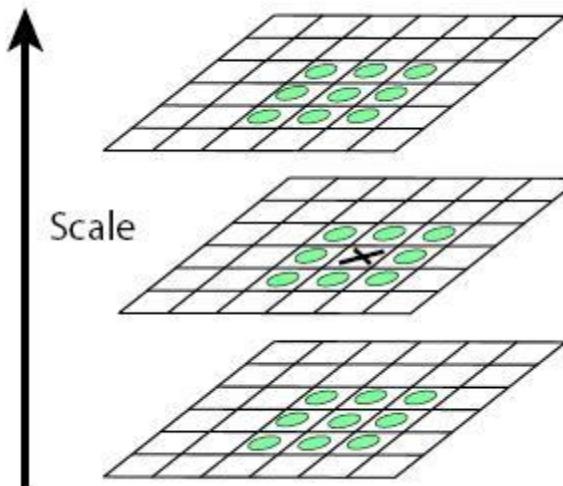
# Scale-Space Blob Detector: Example



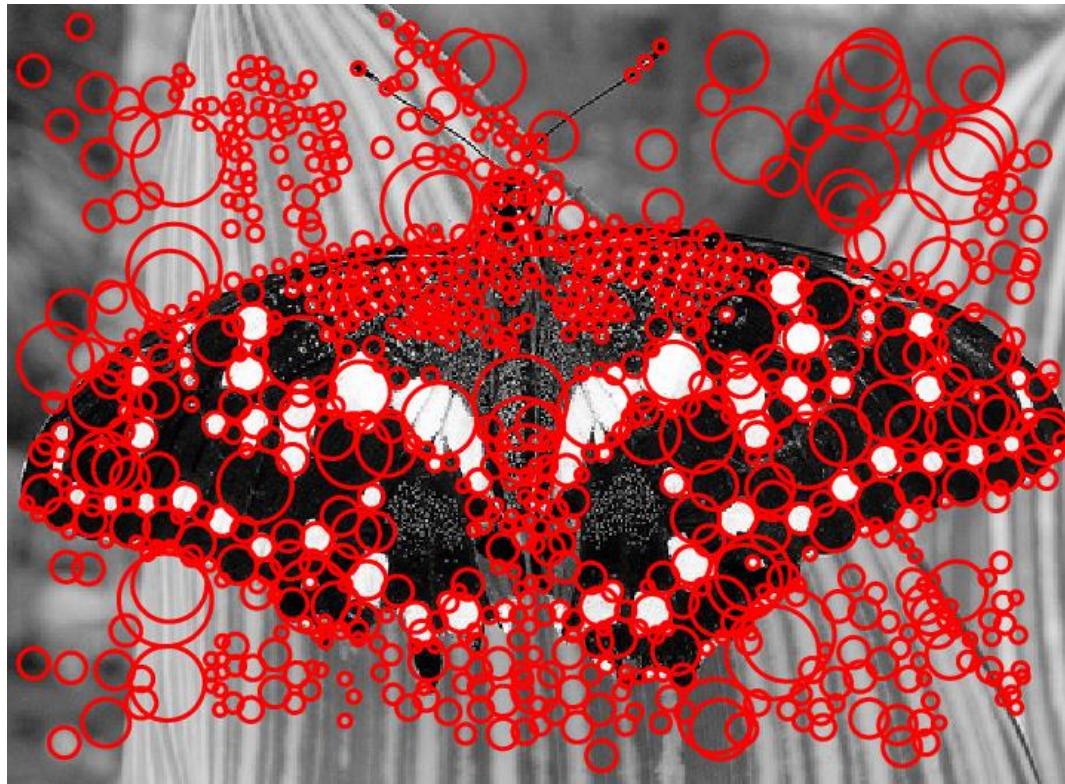
sigma = 11.9912

# Scale-Space Blob Detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space



# Scale-Space Blob Detector: Example



# Efficient Implementation

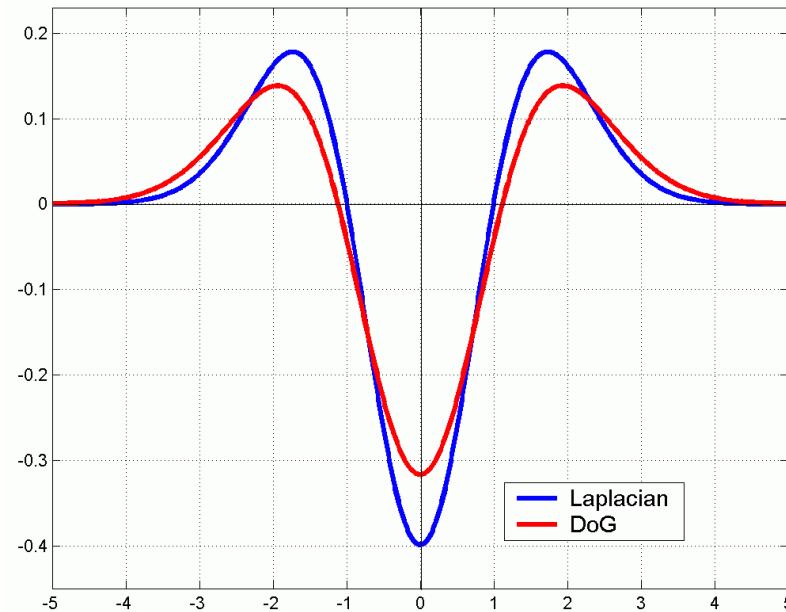
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left( G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

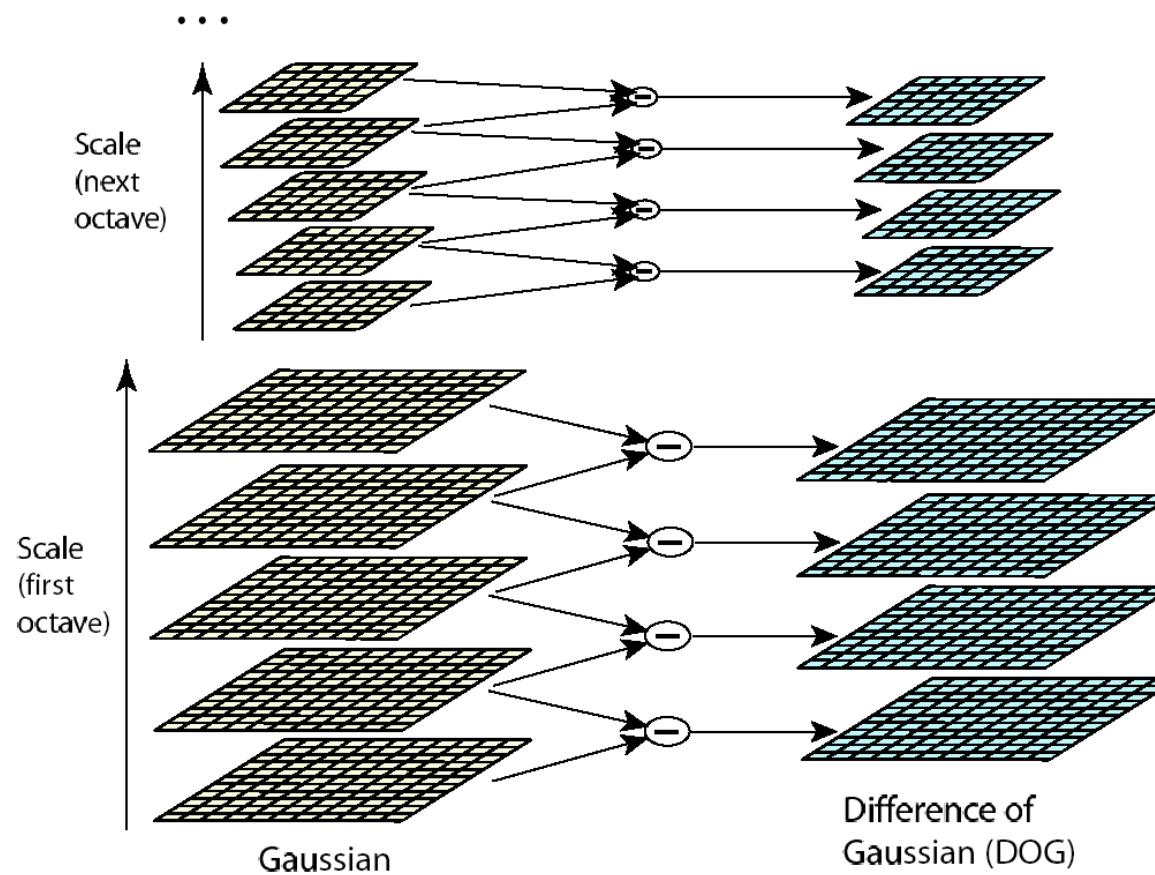
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)



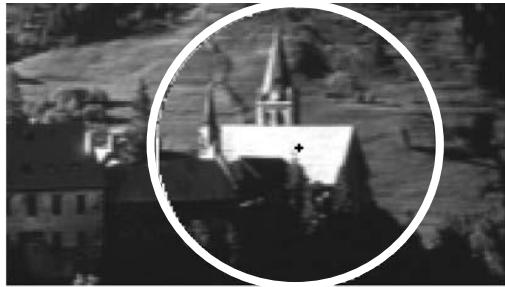
# Efficient Implementation



David G. Lowe. [\*\*"Distinctive image features from scale-invariant keypoints."\*\*](#) IJCV 60 (2), pp. 91-110, 2004.

# From Feature Detection to Feature Description

- Scaled and rotated versions of the same neighborhood will give rise to blobs that are related by the same transformation
- What to do if we want to compare the appearance of these image regions?
  - *Normalization*: transform these regions into same-size circles
  - Problem: rotational ambiguity



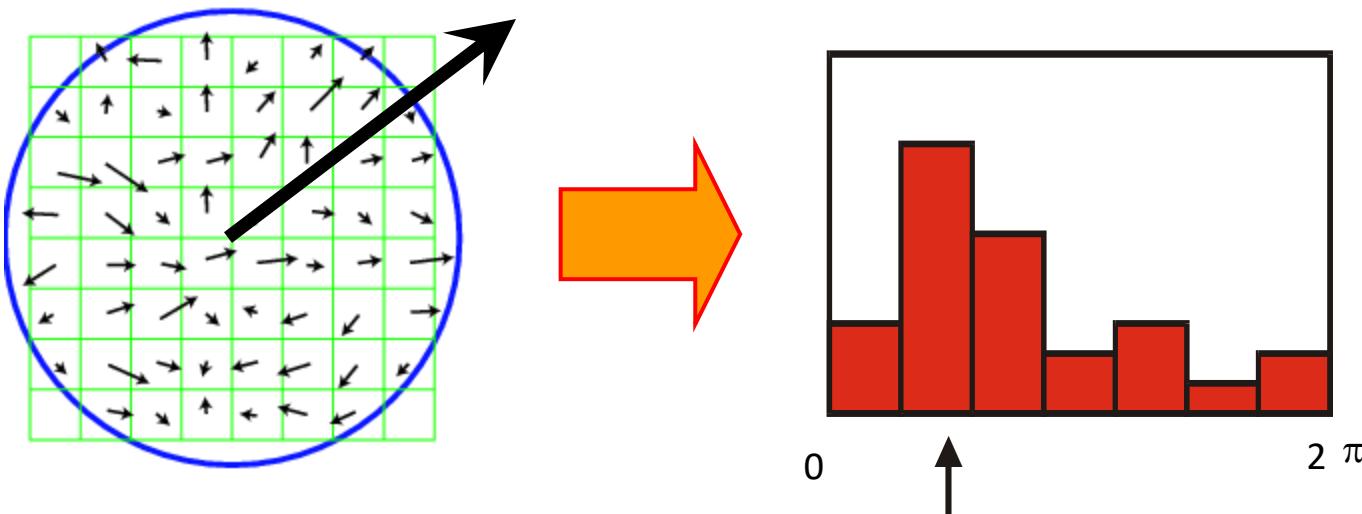
A grayscale image of the same landscape as the first image, but with a smaller white circle highlighting a different region in the foreground, likely a different view or scale of the same feature.

A circular crop of the textured foreground from the first image, centered on the white circle. It shows vertical, wavy patterns, possibly grass or reeds.

A circular crop of the textured foreground from the second image, centered on the white circle. It shows similar vertical, wavy patterns to the first crop.

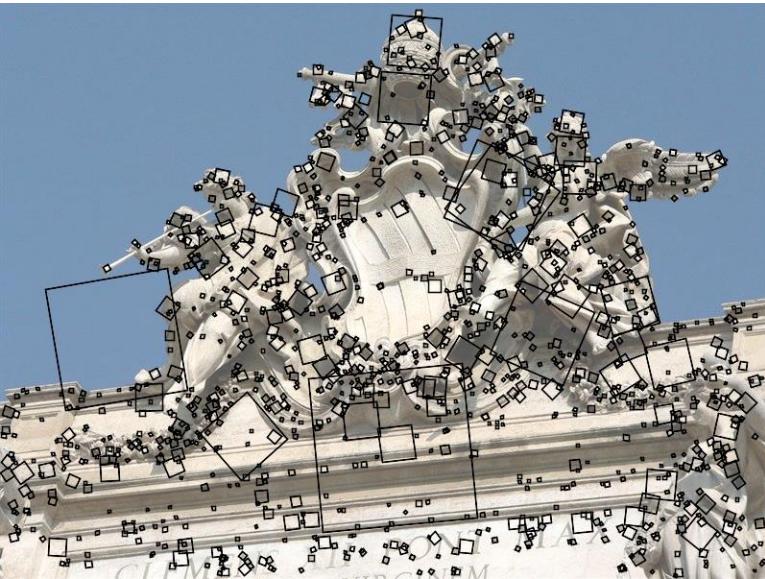
# Eliminating Rotation Ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



# SIFT Features

- Detected features with characteristic scales and orientations:



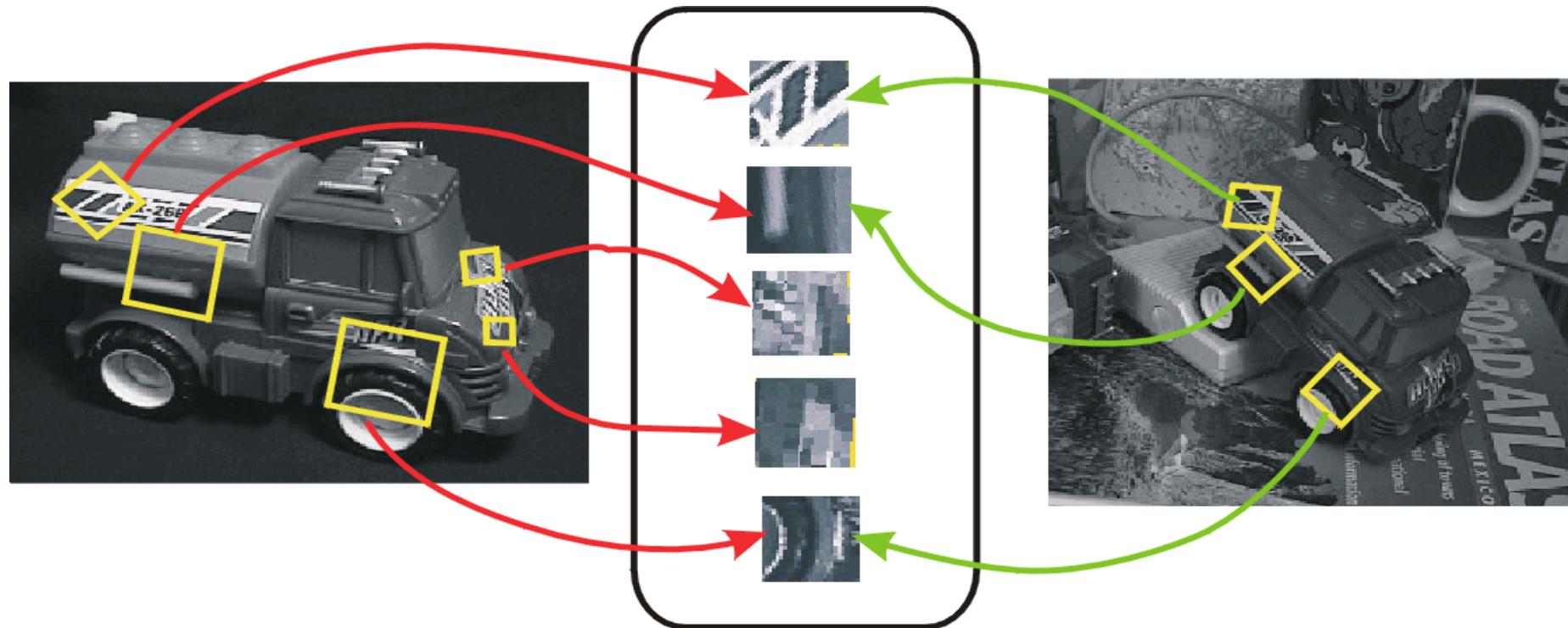
David G. Lowe. "[Distinctive image features from scale-invariant keypoints.](#)" IJCV 60 (2), pp. 91-110, 2004.

# From Feature Detection to Feature Description

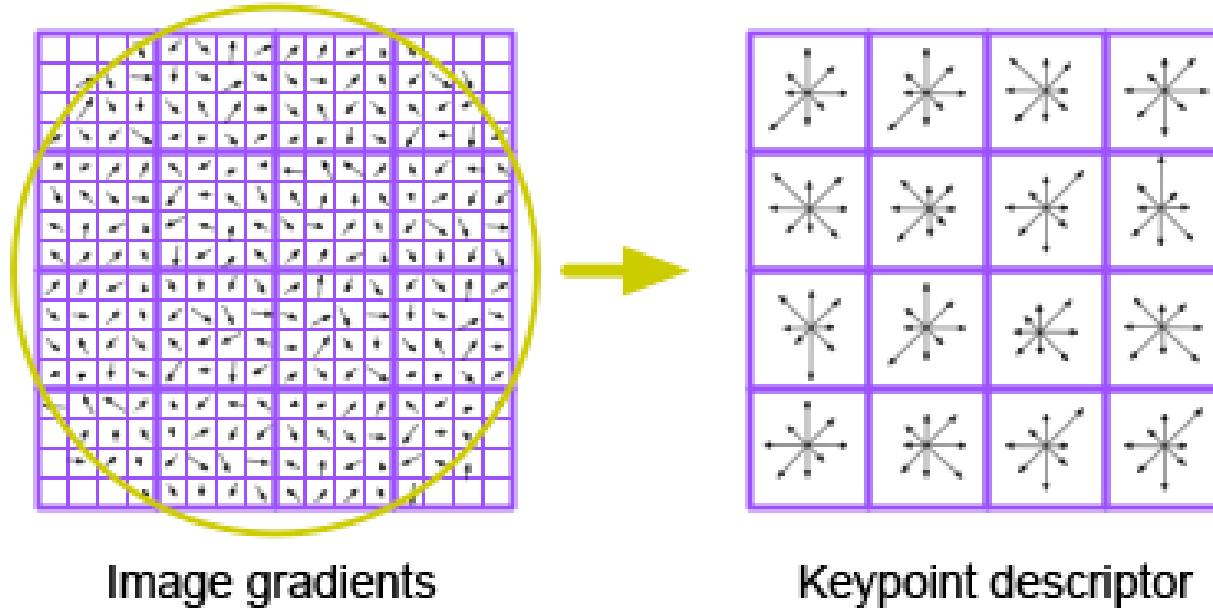
Detection is *covariant*:

$$\text{features}(\text{transform}(\text{image})) = \text{transform}(\text{features}(\text{image}))$$

Description is *invariant*:



# SIFT Descriptors

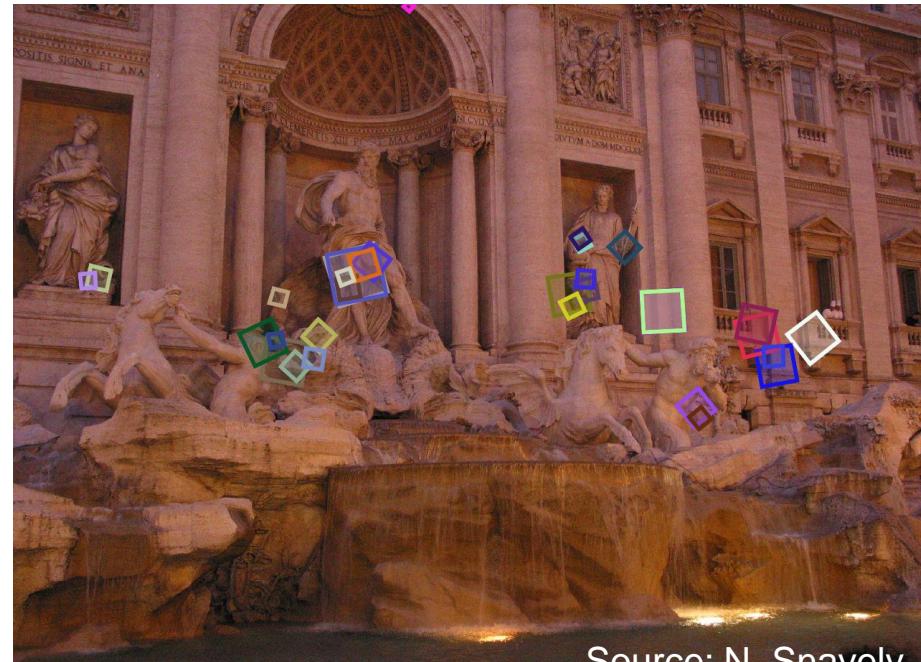


David G. Lowe. "[Distinctive image features from scale-invariant keypoints.](#)" IJCV 60 (2), pp. 91-110, 2004.

# Properties of SIFT

Extraordinarily robust detection and description technique

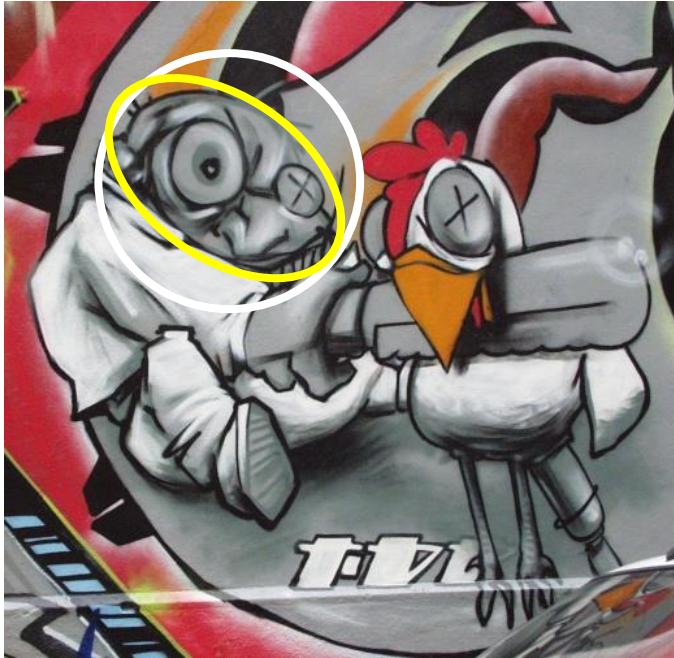
- Can handle changes in viewpoint
  - Up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Fast and efficient—can run in real time
- Lots of code available



Source: N. Snavely

# Affine adaptation

Affine transformation approximates viewpoint changes for roughly planar objects and roughly orthographic cameras



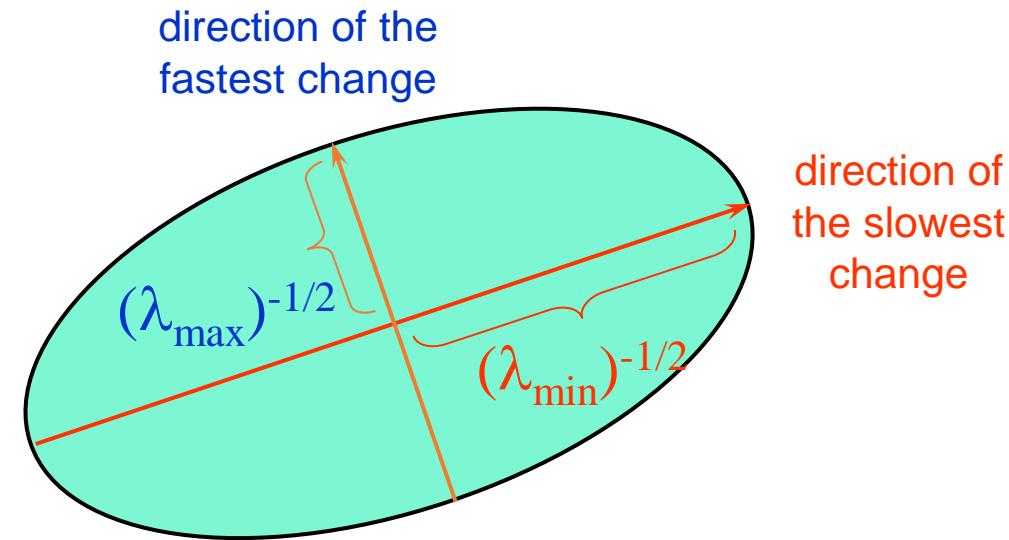
# Affine Adaptation

Consider the second moment matrix of the window containing the blob:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

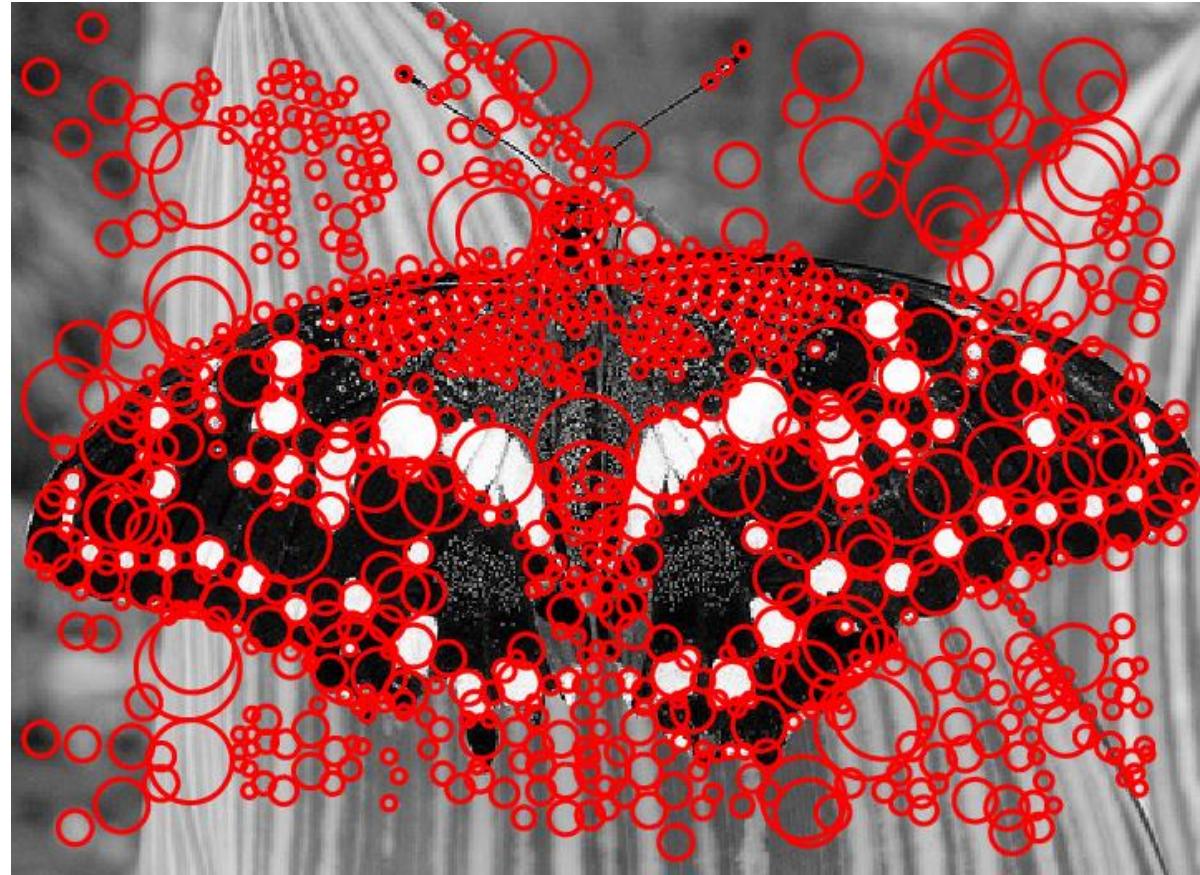
Recall:

$$[u \ v] M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$



This ellipse visualizes the “characteristic shape” of the window

# Affine Adaptation Example



Scale-invariant regions (blobs)

# Affine Adaptation Example



Affine-adapted blobs