

Computer Vision

CS-E4850, 5 study credits

Lecturer: Juho Kannala

Lecture 3: Image features

- Low-level features (corners, edges, texture patches) are needed for many tasks:
 - Image matching and registration
 - Structure-from-motion and image-based 3D modeling
 - Image segmentation
 - Object recognition
 - Image retrieval
- Relevant reading:
 - Chapter 4 of Szeliski's book
 - David Lowe's article (2004)
<http://www.cs.ubc.ca/~lowe/keypoints/>

Acknowledgement: many slides from Svetlana Lazebnik, Steve Seitz, David Lowe, Kristen Grauman, and others (detailed credits on individual slides)

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)

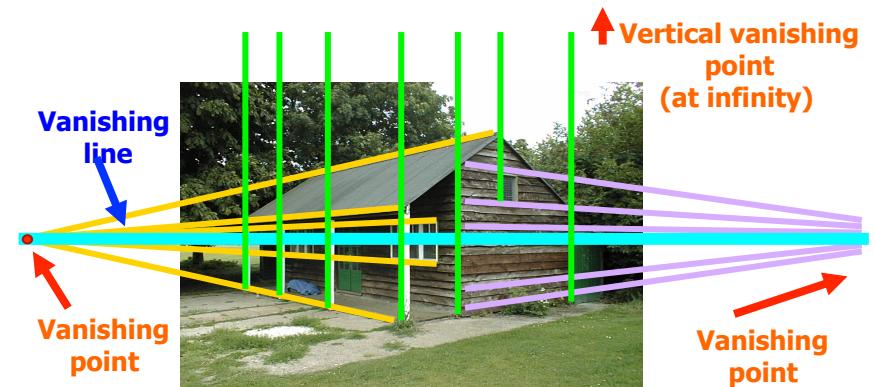


Why do we care about edges?

Extract information, recognize objects

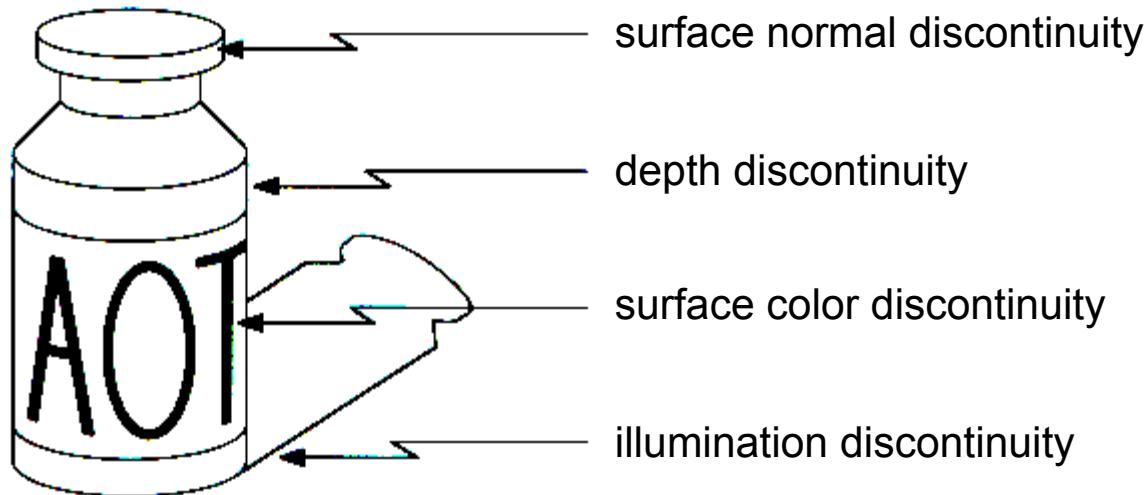


Recover geometry and viewpoint



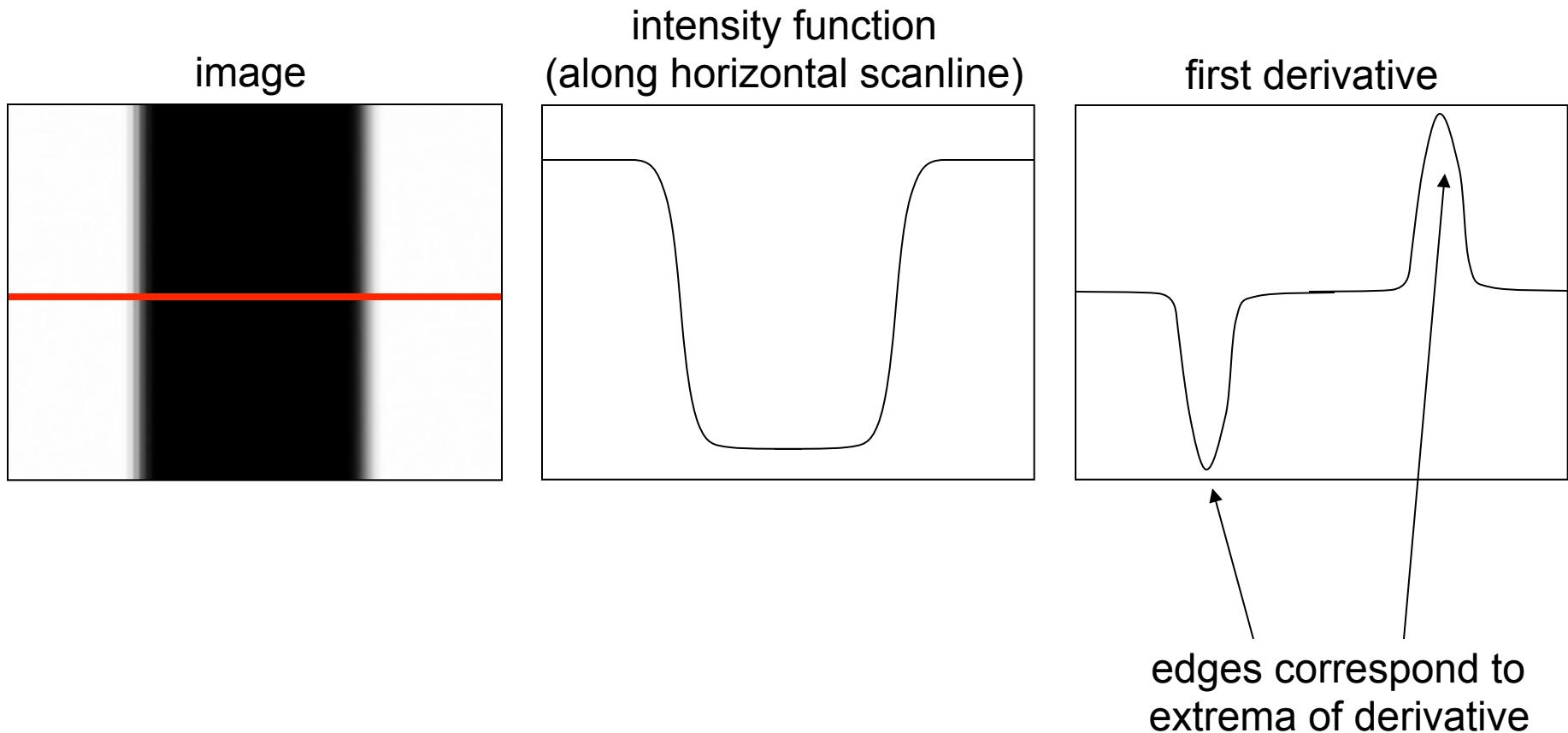
Origin of edges

Edges are caused by a variety of factors:



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:

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

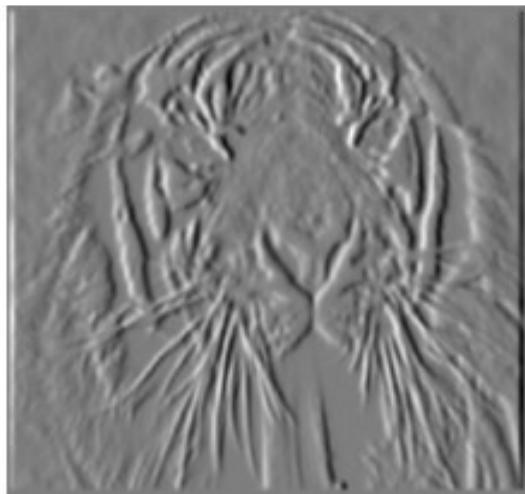
To implement the above as convolution, what would be the associated filter?

Partial derivatives of an image



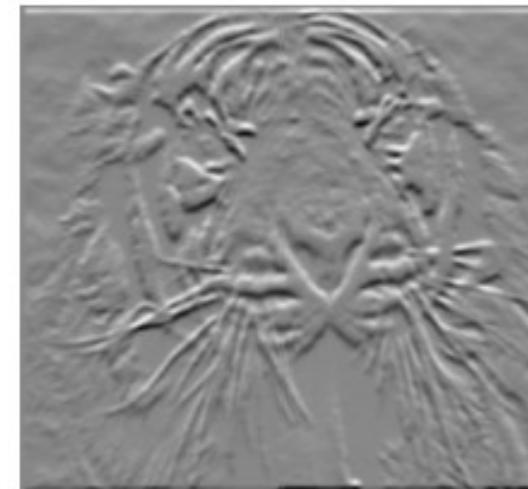
$$\frac{\partial f(x, y)}{\partial x}$$

-1	1
----	---



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

-1	or	1
1		-1



Which shows changes with respect to x?

Finite difference filters

Other approximations of derivative filters exist:

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

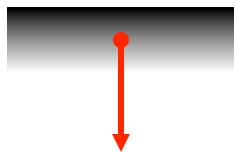
Sobel: $M_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

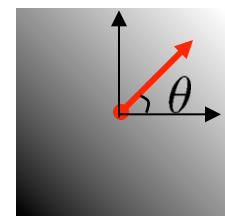
Roberts: $M_x = \begin{array}{|c|c|} \hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

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?

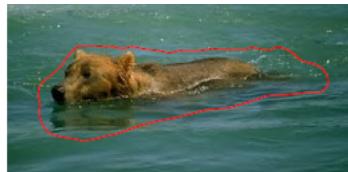
The gradient direction is given by $\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

The edge strength is given by the gradient magnitude

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

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

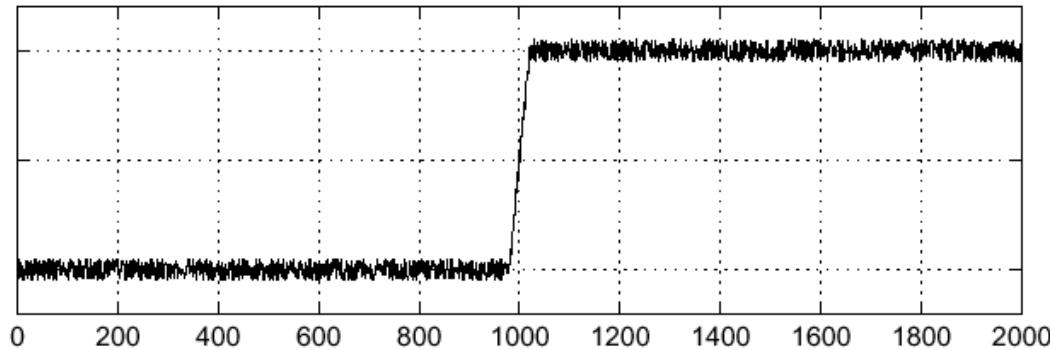


seamless cloning

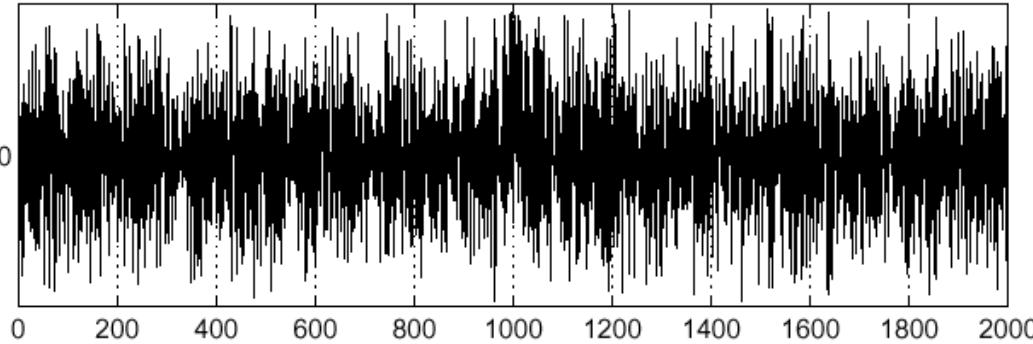
Effects of noise

Consider a single row or column of the image

$$f(x)$$

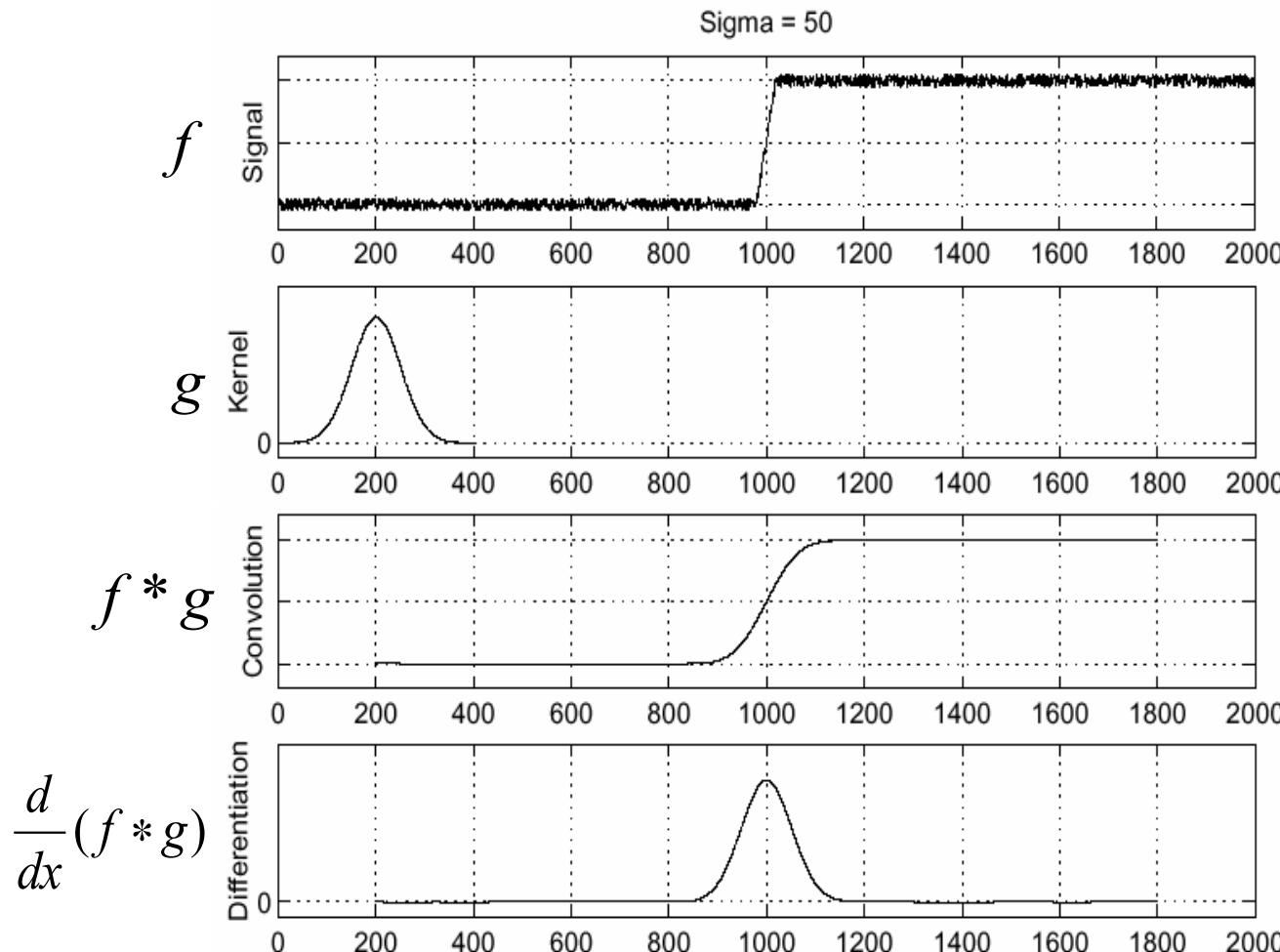


$$\frac{d}{dx} f(x)_0$$



Where is the edge?

Solution: smooth first

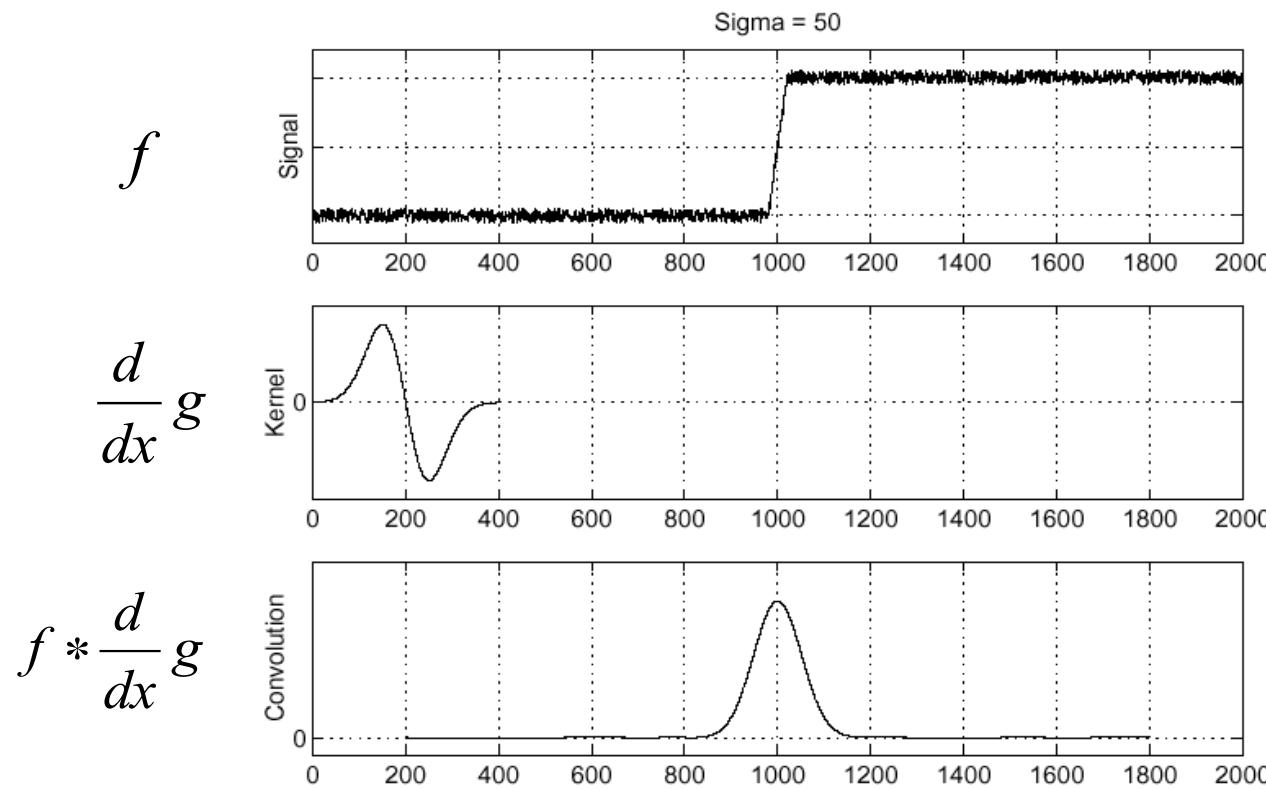


- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

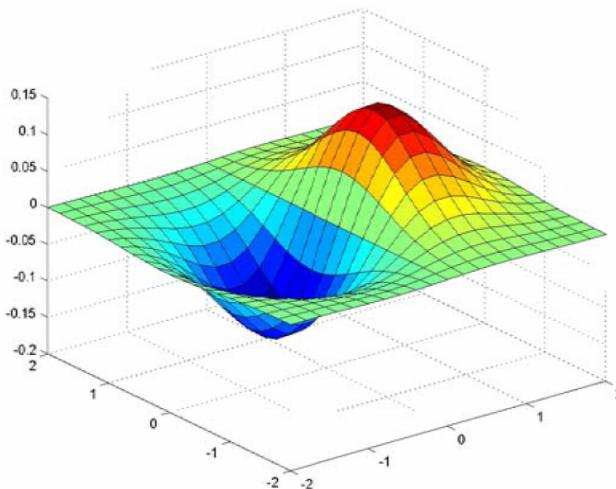
Source: S. Seitz

Derivative theorem of convolution

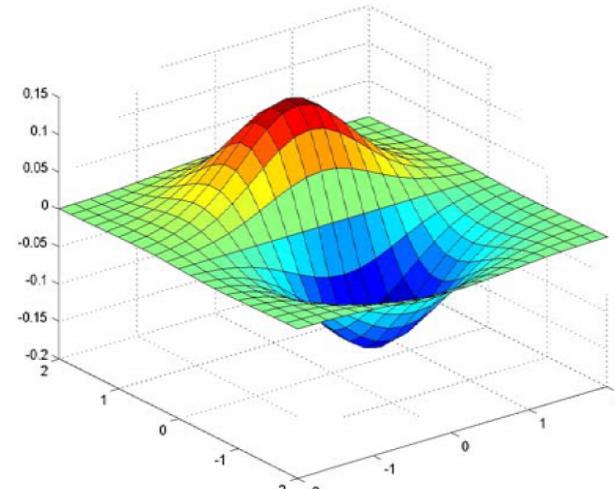
- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$
- This saves us one operation:



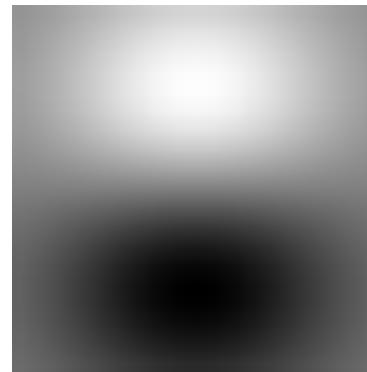
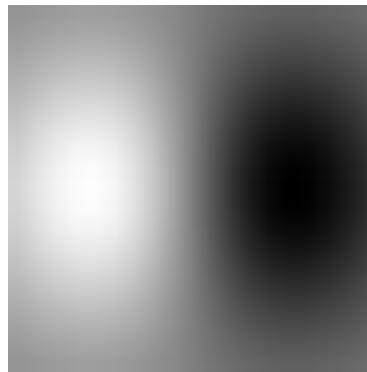
Derivative of Gaussian filters



x-direction

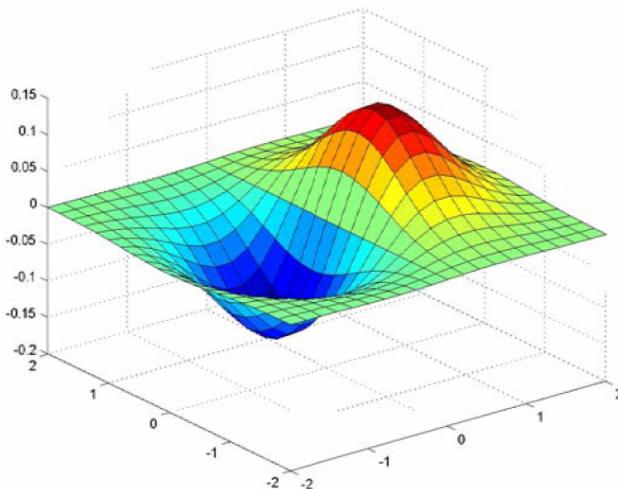


y-direction

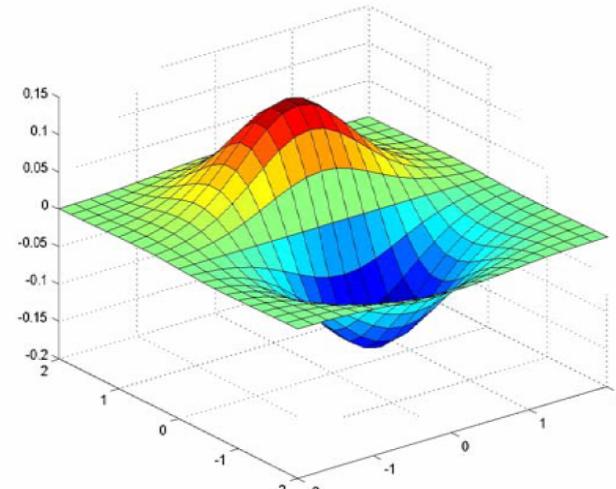


Which one finds horizontal/vertical edges?

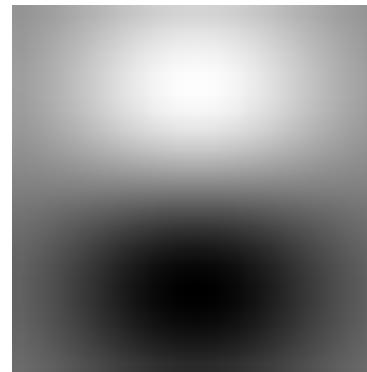
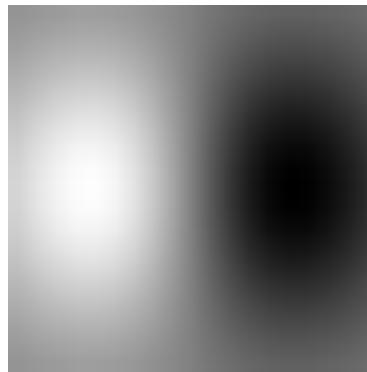
Derivative of Gaussian filters



x-direction



y-direction



Are these filters separable?

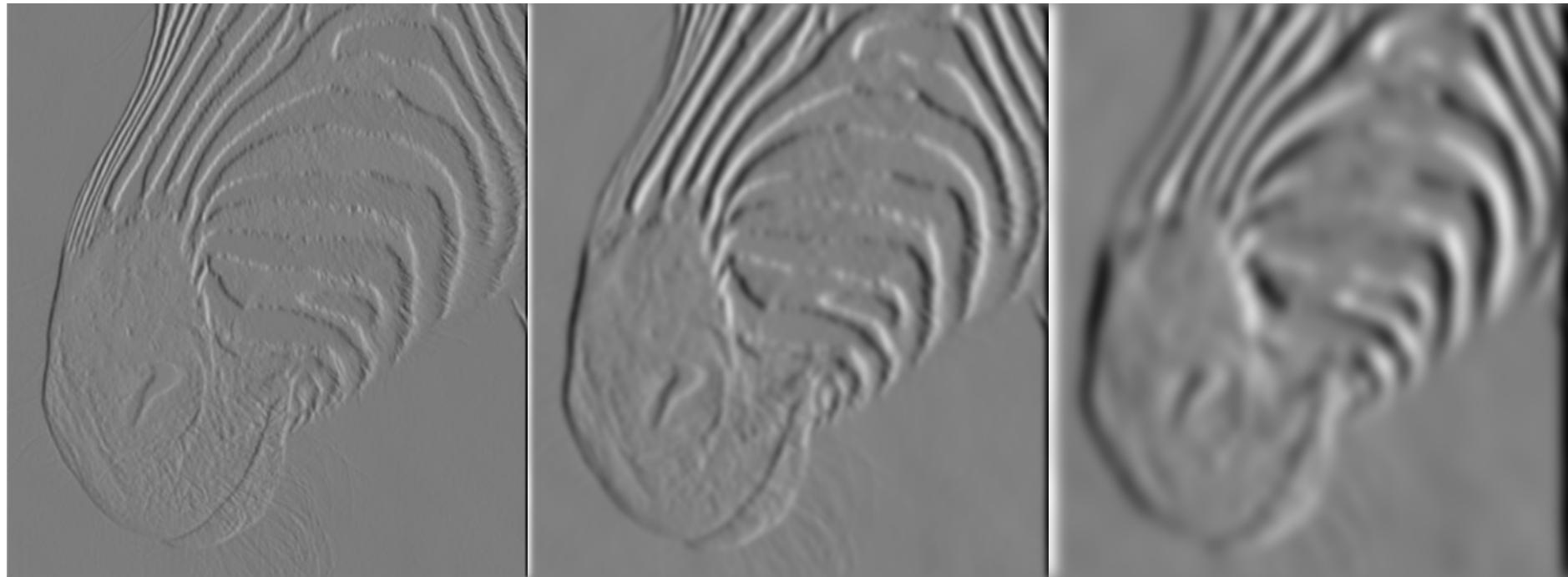
Recall: Separability of the Gaussian filter

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

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

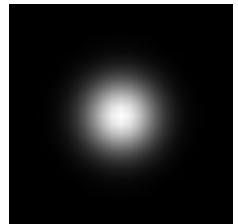
7 pixels

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

Review: Smoothing vs. derivative filters

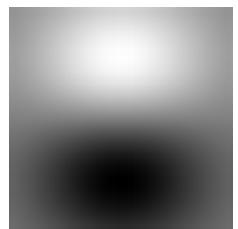
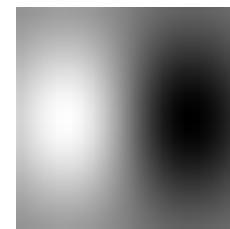
Smoothing filters

- 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



Building an edge detector

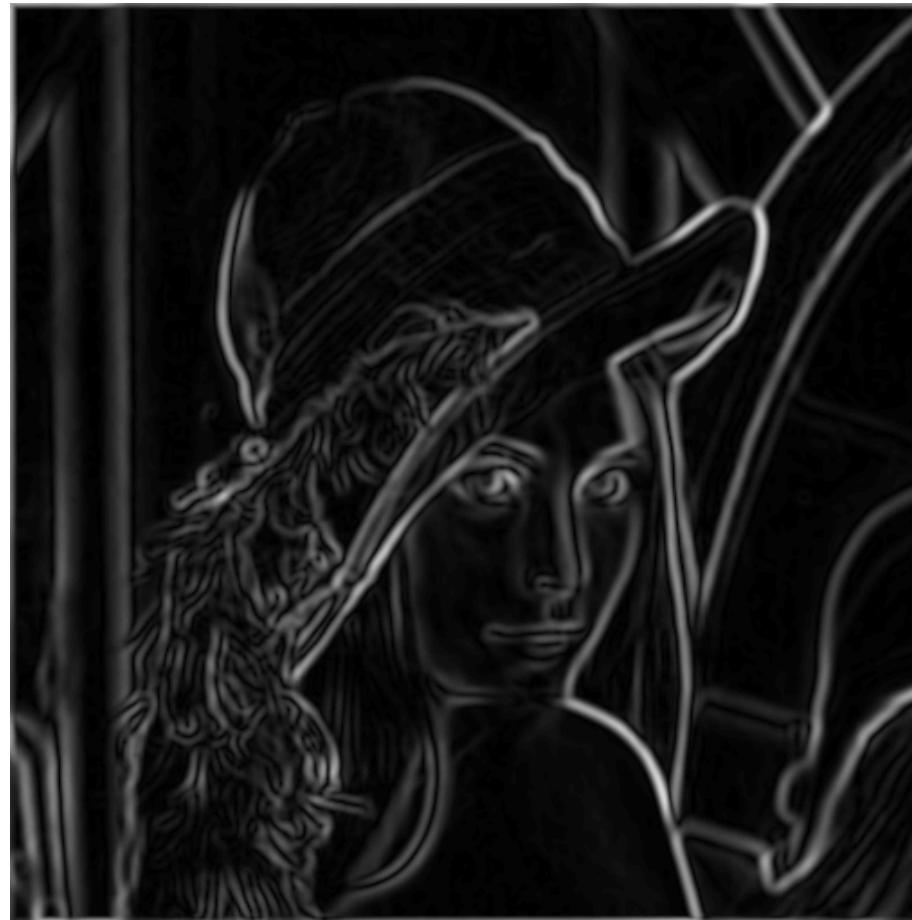


original image



final output

Building an edge detector



norm of the gradient

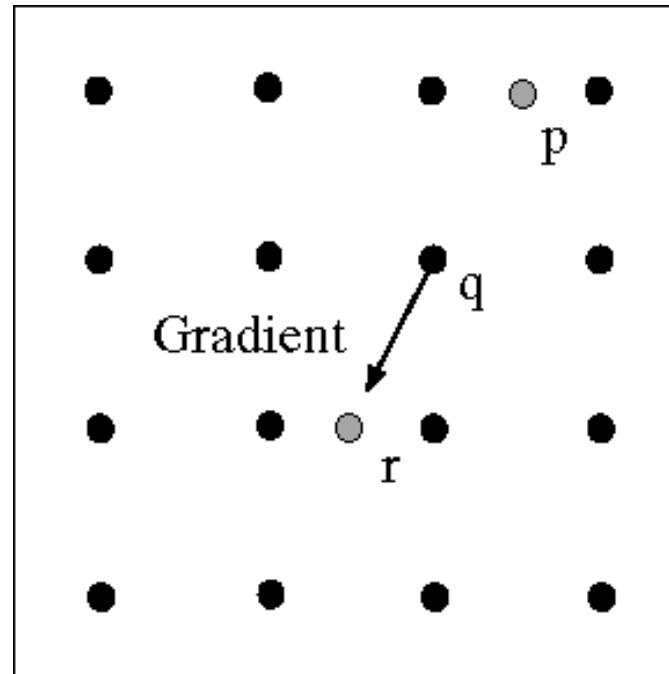
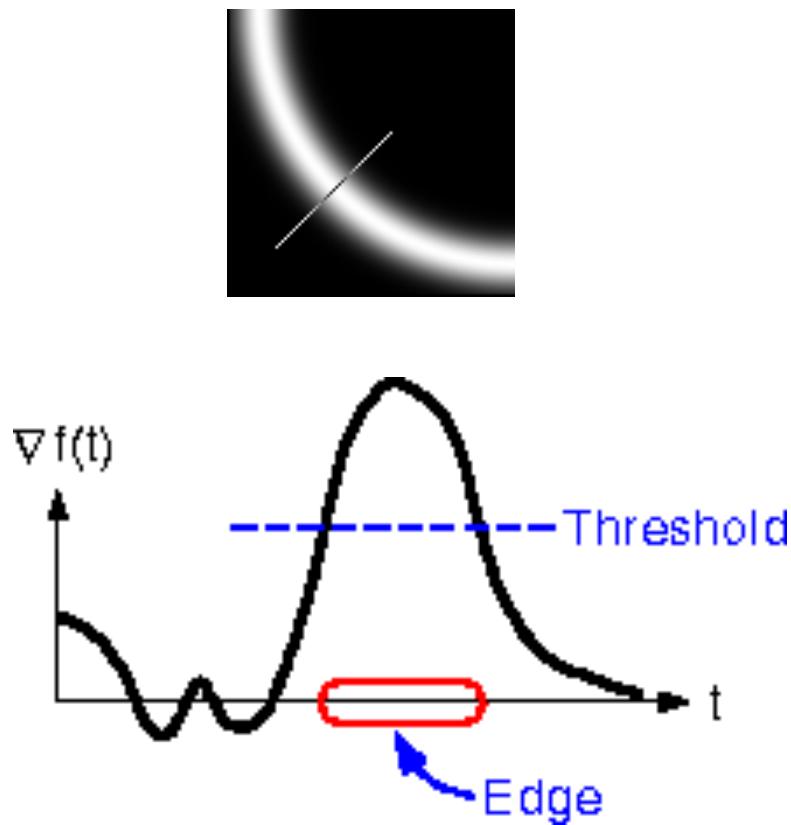
Building an edge detector



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

Thresholded norm of the gradient

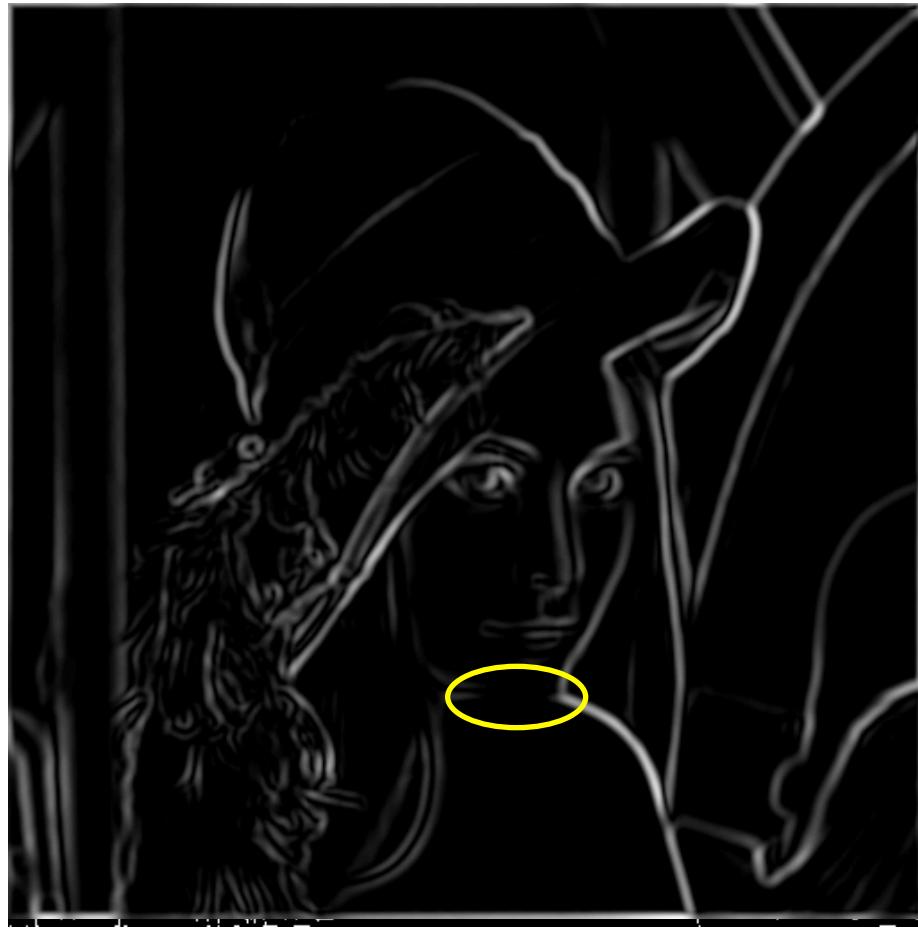
Non-maximum suppression



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

- requires checking interpolated pixels p and r

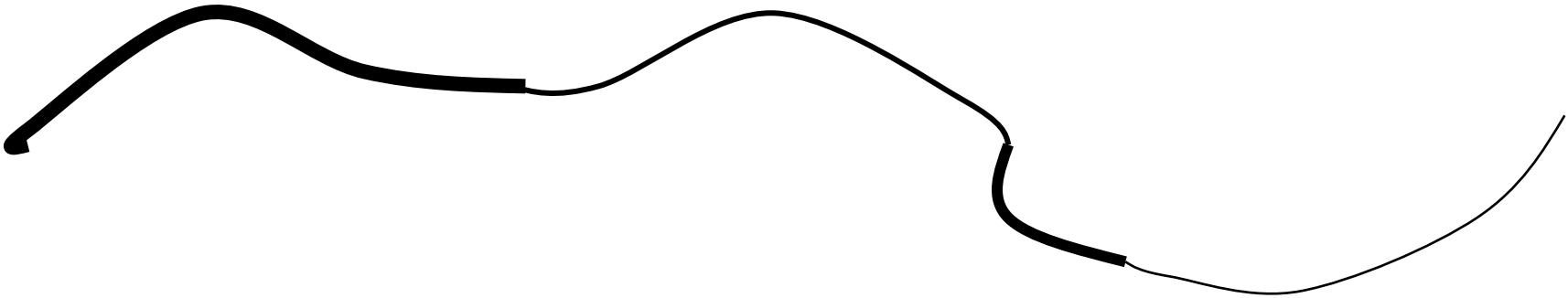
Non-maximum suppression



Another 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

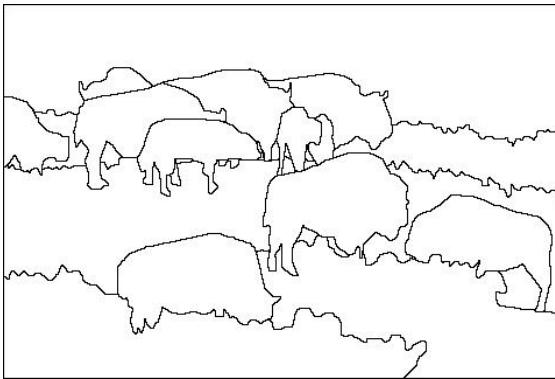
MATLAB: `edge(image, 'canny');`

Image gradients vs. meaningful contours

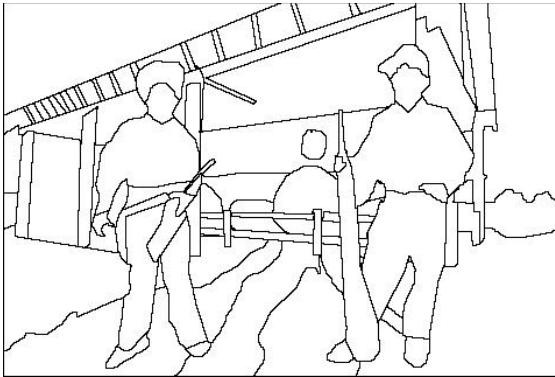
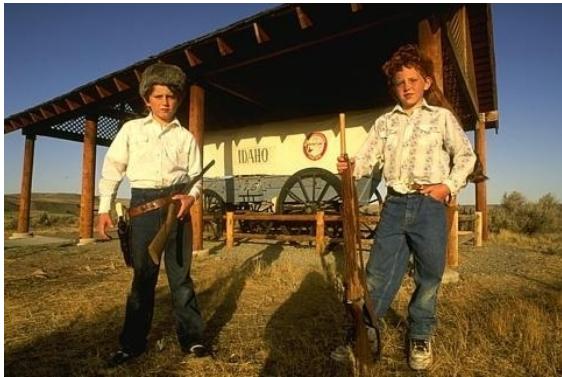
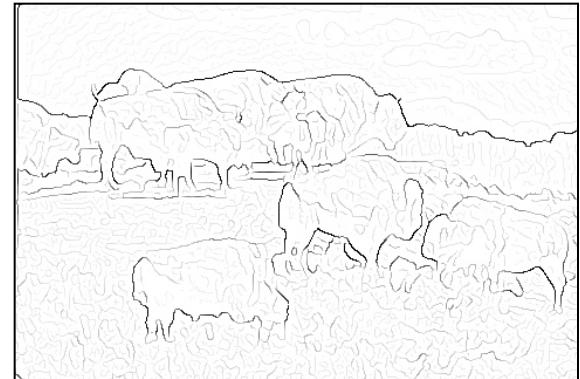
image



human segmentation



gradient magnitude

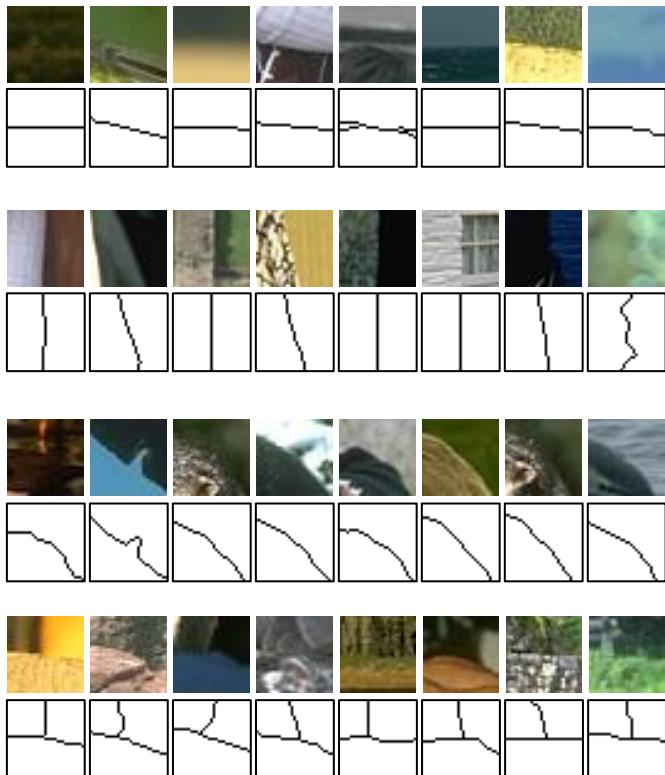


Berkeley segmentation database:

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

Data-driven edge detection

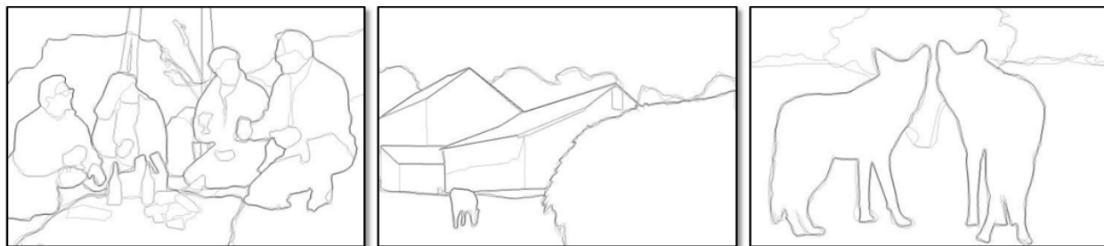
Training data



Input images



Ground truth



Output



Keypoint extraction: Corners



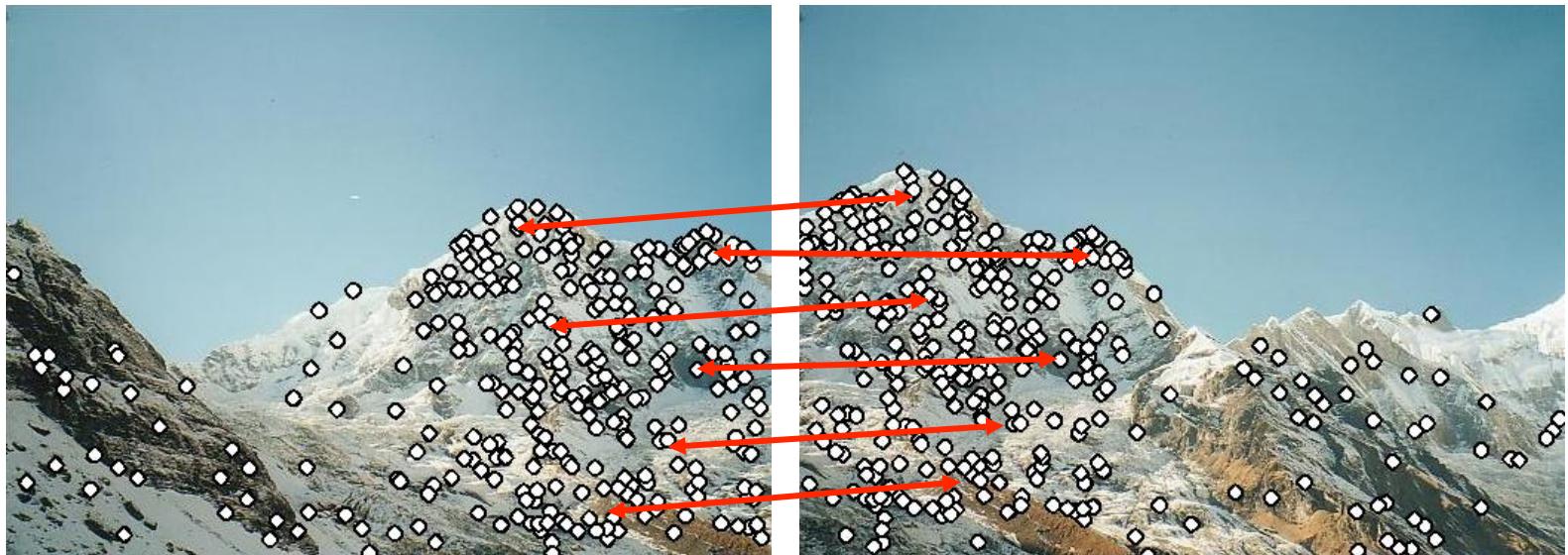
Why extract keypoints?

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



Why extract keypoints?

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



Step 1: extract keypoints

Step 2: match keypoint features

Why extract keypoints?

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

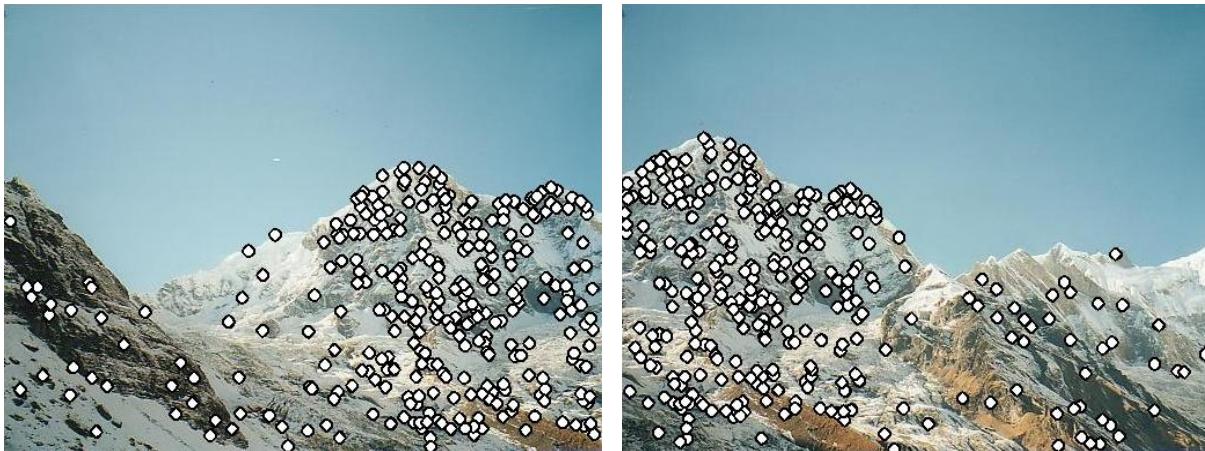


Step 1: extract keypoints

Step 2: match keypoint features

Step 3: align images

Characteristics of good keypoints



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

Applications

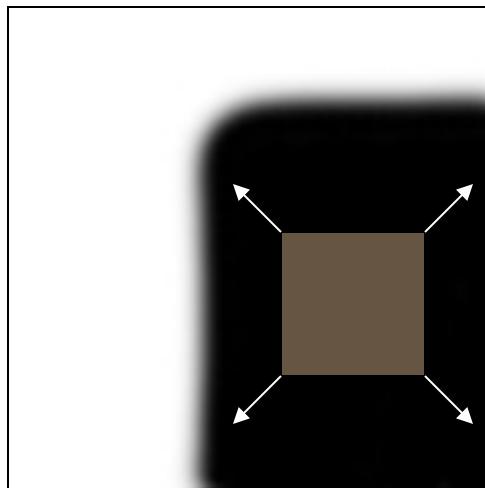
Keypoints are used for:

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

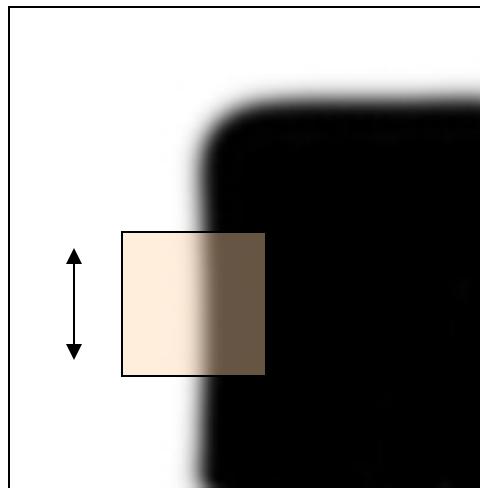


Corner Detection: Basic Idea

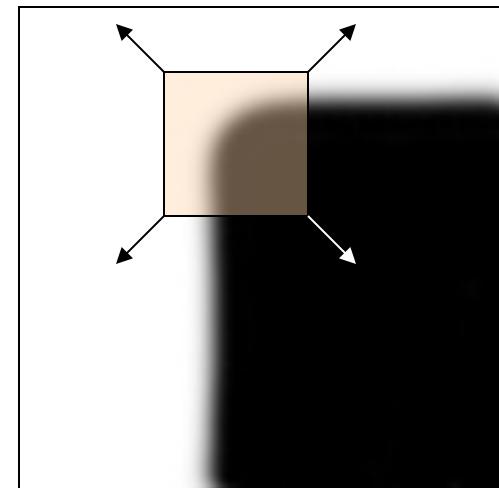
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give a *large change* in intensity



“flat” region:
no change in
all directions



“edge”:
no change along
the edge
direction

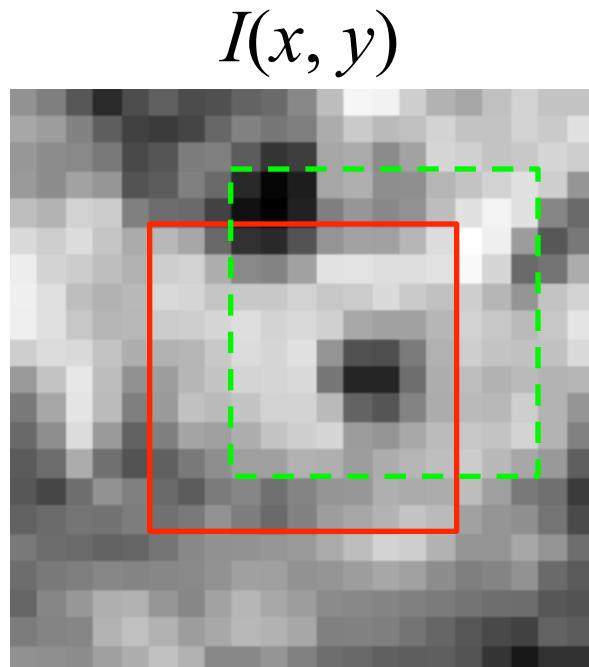


“corner”:
significant
change in all
directions

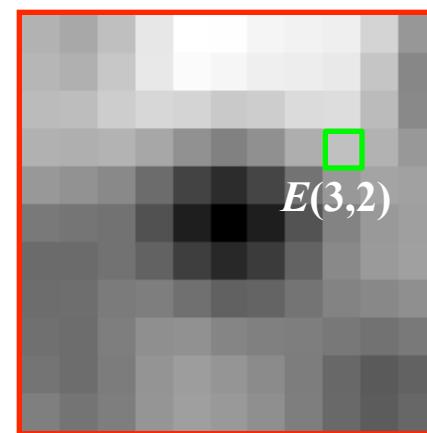
Corner Detection: Mathematics

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

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



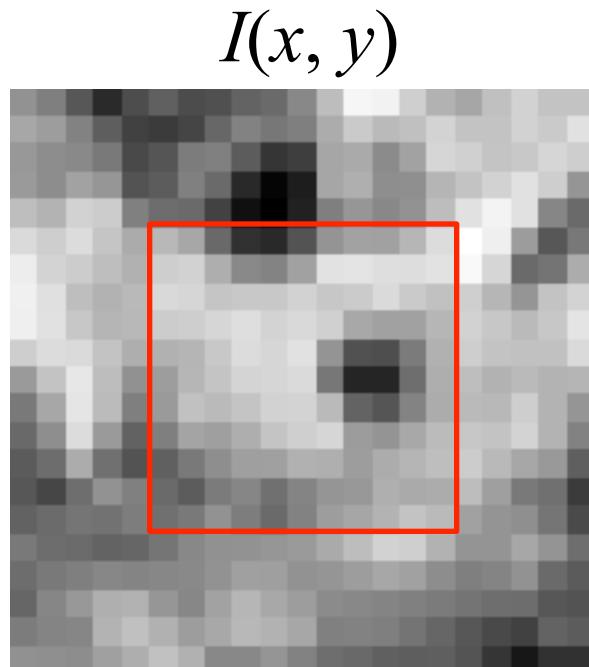
$$E(u, v)$$



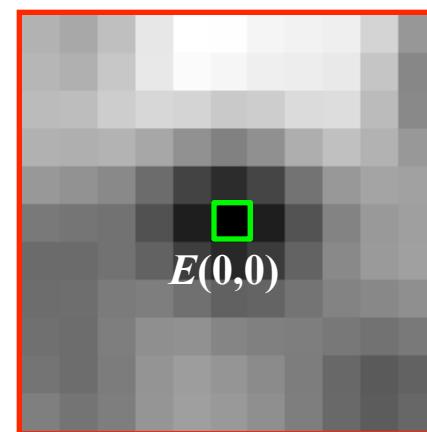
Corner Detection: Mathematics

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

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



$E(u, v)$



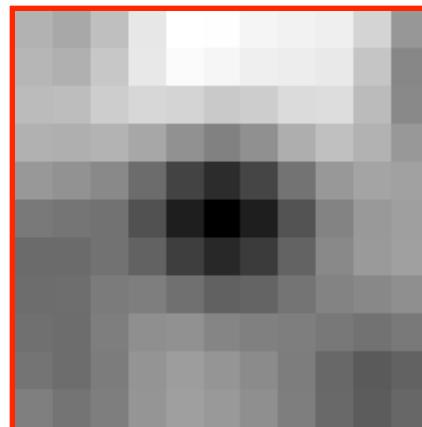
Corner Detection: Mathematics

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 $E(u, v)$:

$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I(x, y) + I_x u + I_y v - I(x, y)]^2 \\ &= \sum_{(x,y) \in W} [I_x u + I_y v]^2 = \sum_{(x,y) \in W} I_x^2 u^2 + 2I_x I_y uv + I_y^2 v^2 \end{aligned}$$

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:

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

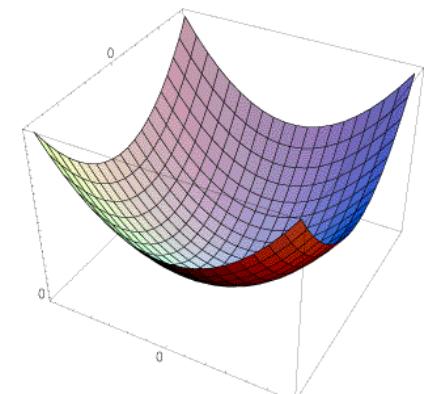
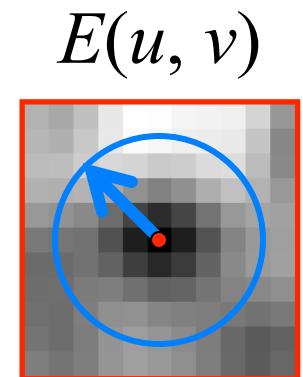
(the sums are over all the pixels in the window W)

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?

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

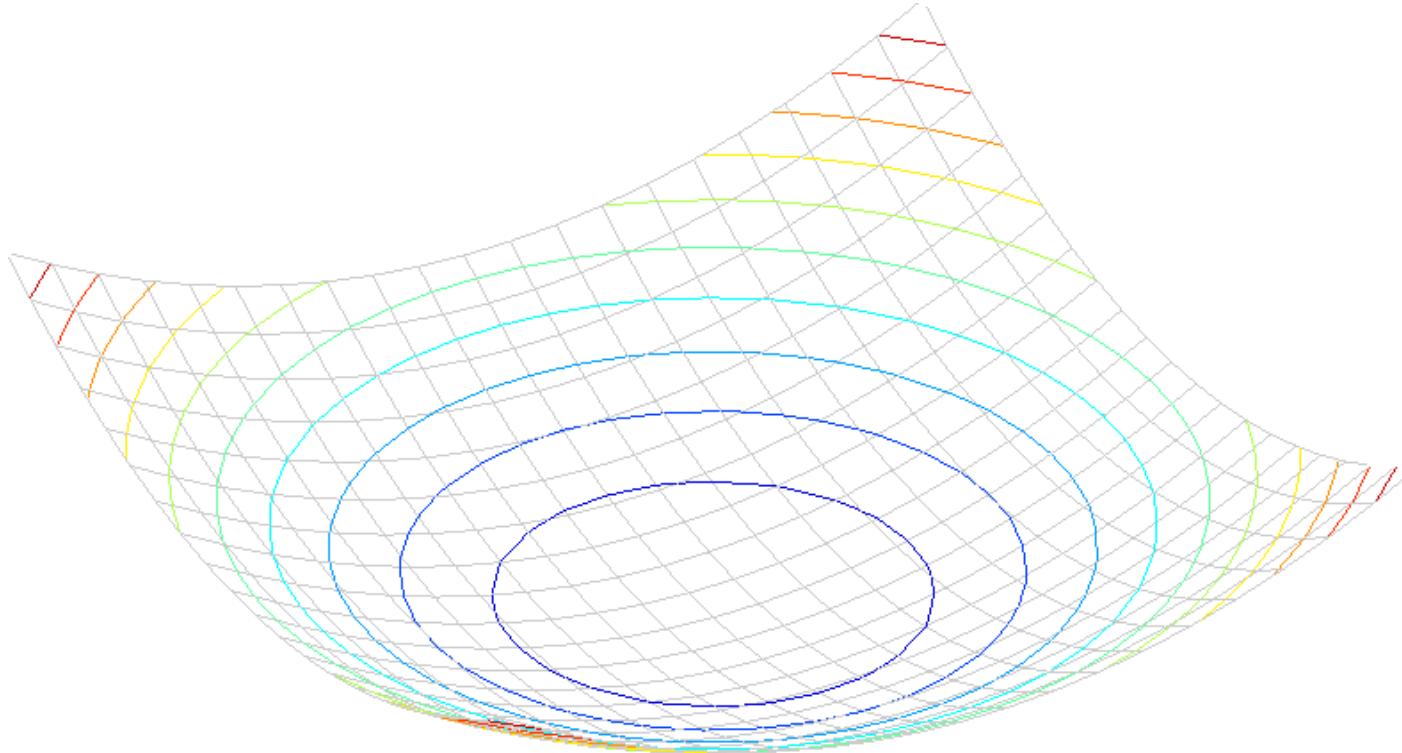
$$M = \begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix}$$



Interpreting the second moment matrix

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.



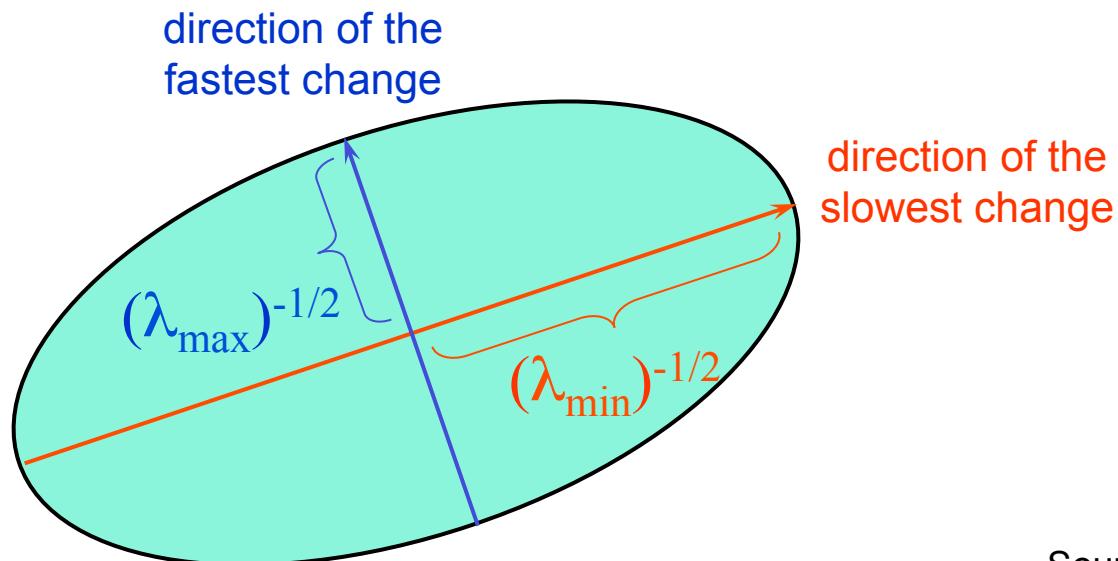
Interpreting the second moment matrix

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



Interpreting the second moment matrix

Consider the axis-aligned case (gradients are either horizontal or vertical)

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

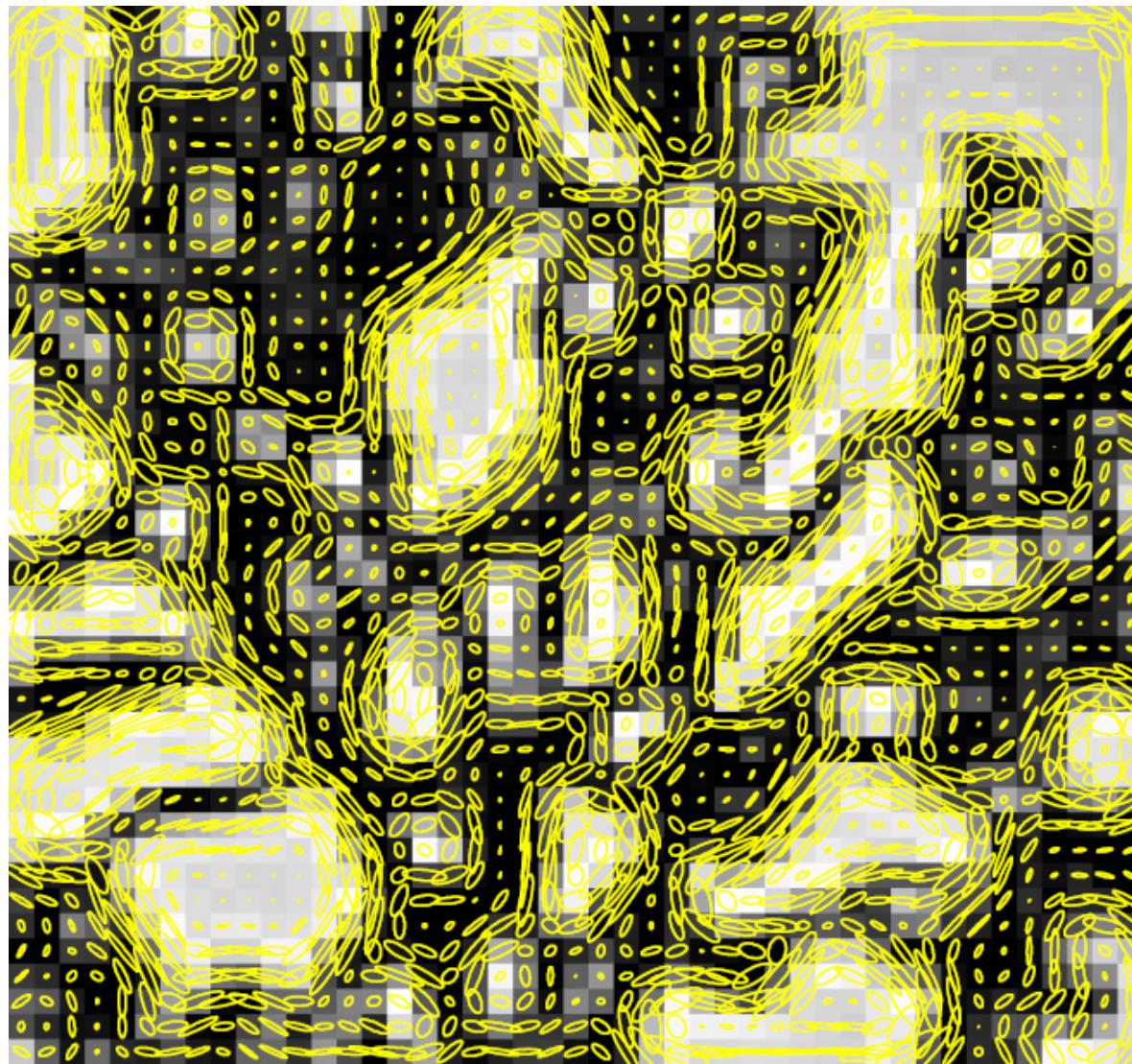
If either a or b is close to 0, then this is **not** a corner, so look for locations where both are large.

Visualization of second moment matrices



Source: S. Lazebnik

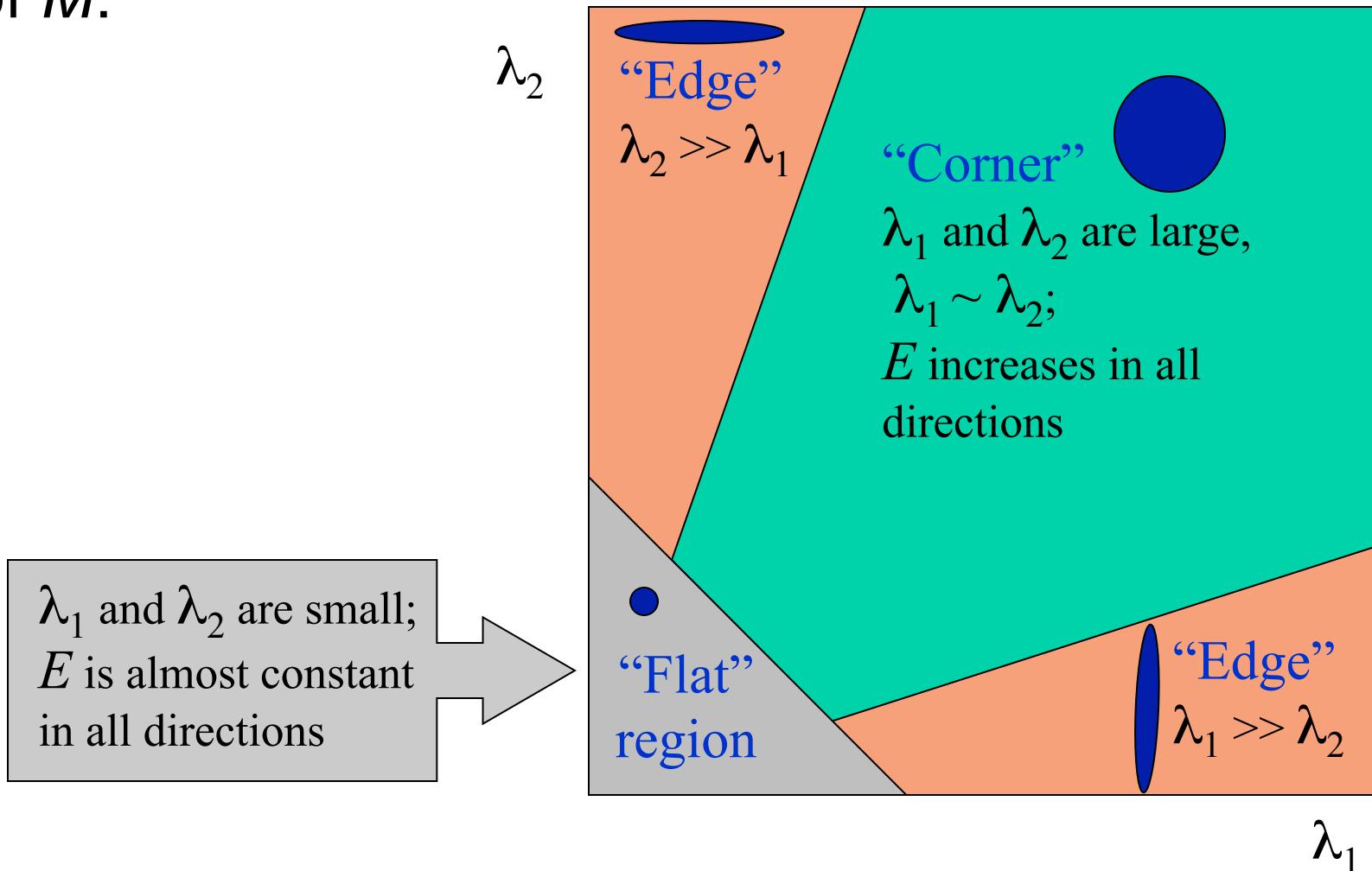
Visualization of second moment matrices



Source: S. Lazebnik

Interpreting the eigenvalues

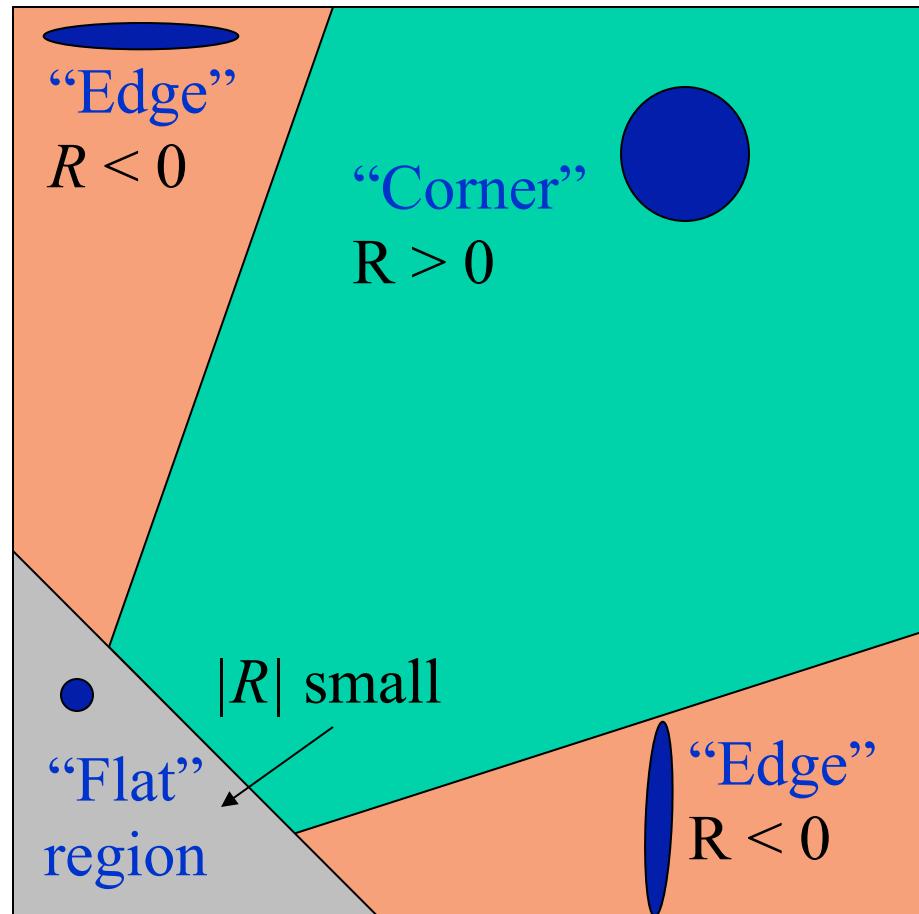
Classification of image points using eigenvalues of M :



Corner response function

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

α : 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.

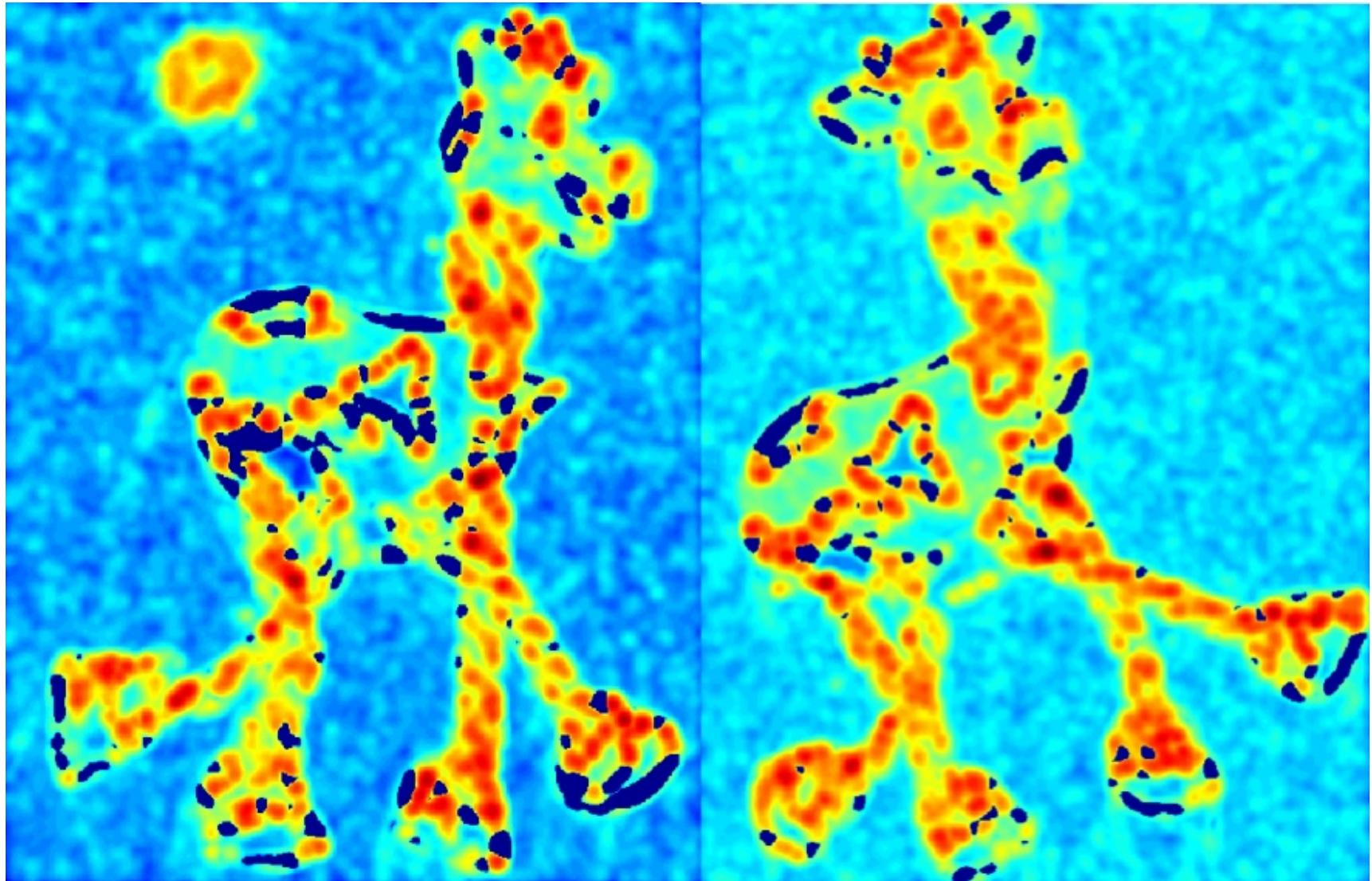
["A Combined Corner and Edge Detector."](#) *Proceedings of the 4th Alvey Vision Conference*: pages 147—151, 1988.

Harris Detector: Steps



Harris Detector: Steps

Compute corner response R



The Harris corner detector

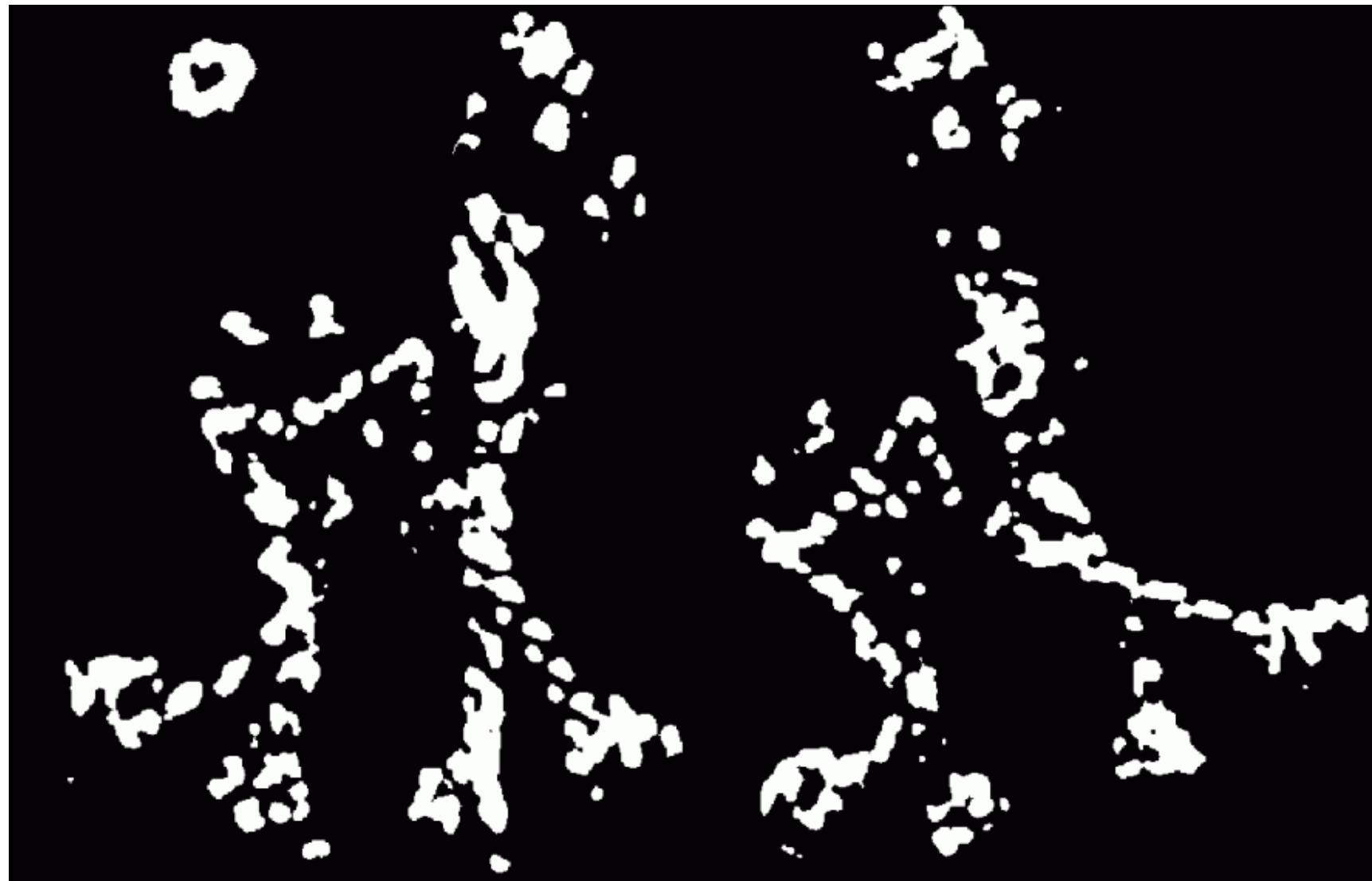
1. Compute partial derivatives at each pixel
2. Compute second moment matrix M in a Gaussian window around each pixel
3. Compute corner response function R
4. Threshold R
5. Find local maxima of response function
(nonmaximum suppression)

C.Harris and M.Stephens.

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

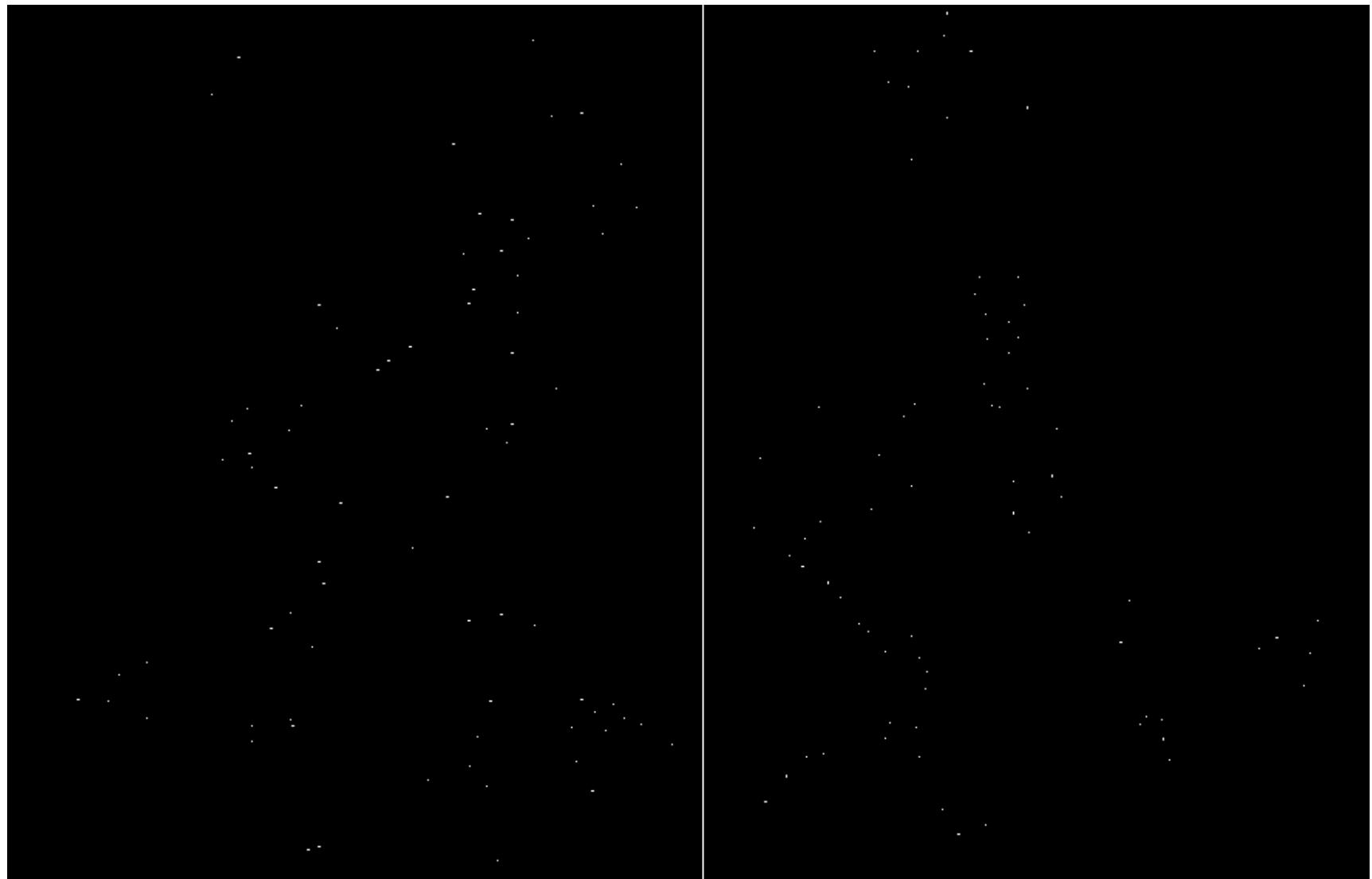
Harris Detector: Steps

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



Harris Detector: Steps

Take only the points of local maxima of R



Harris Detector: Steps

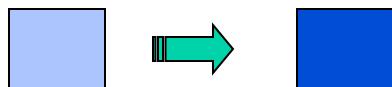


Robustness of corner features

- What happens to corner features when the image undergoes geometric or photometric transformations?

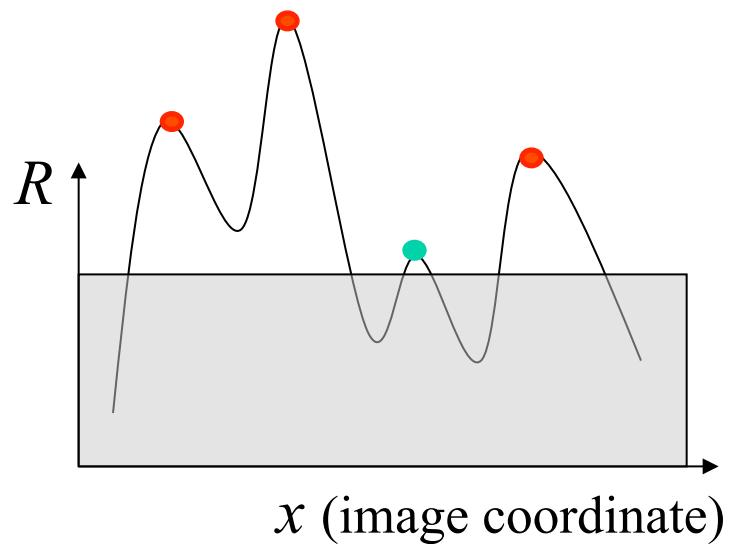
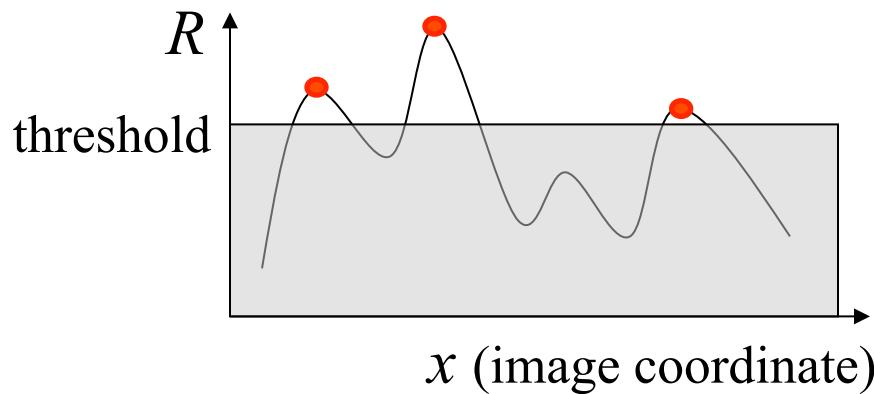


Affine intensity change



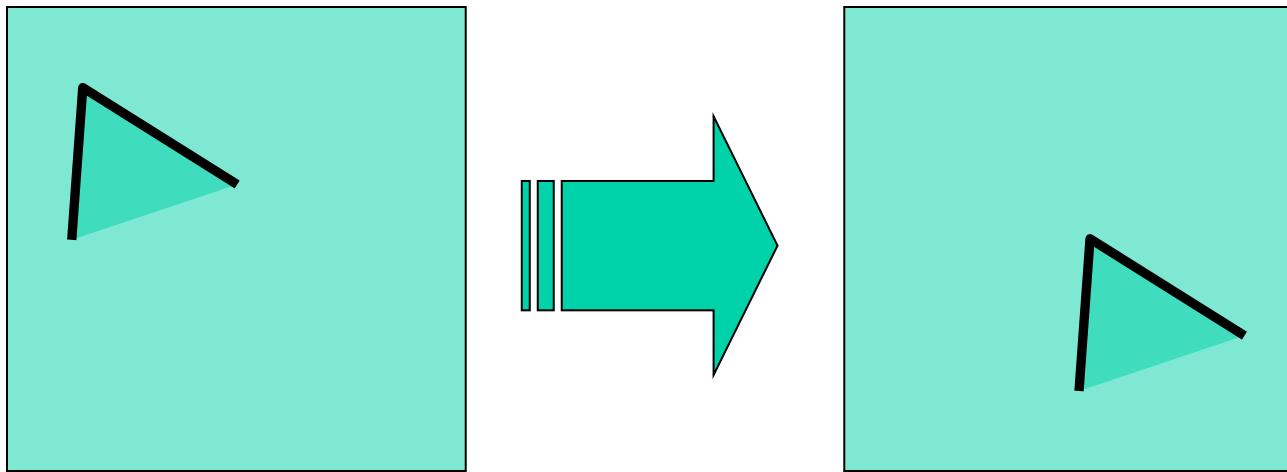
$$I \rightarrow a I + b$$

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



Partially invariant to affine intensity change

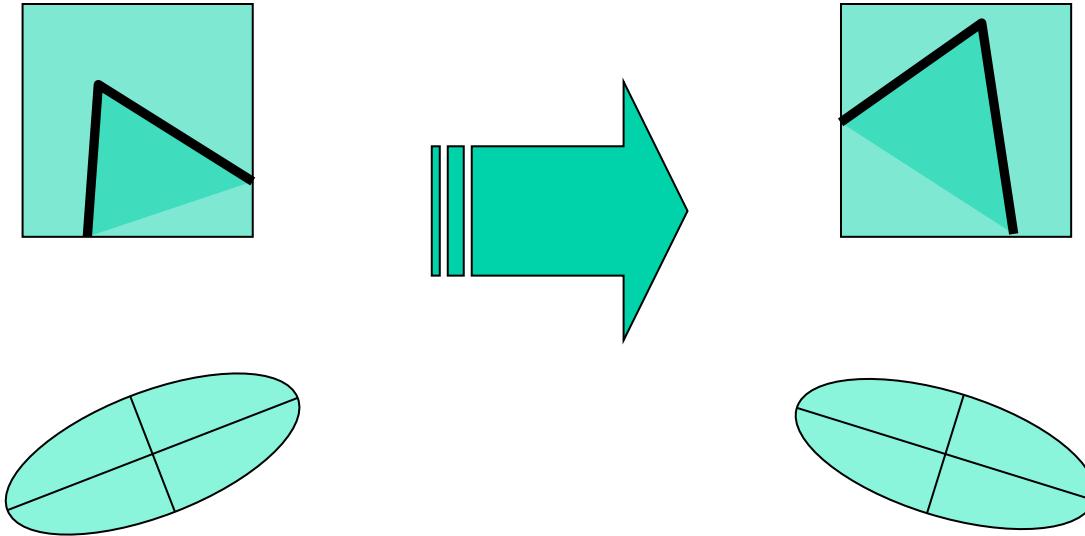
Image translation



- Derivatives and window function are shift-invariant

Corner location is *covariant* w.r.t. translation

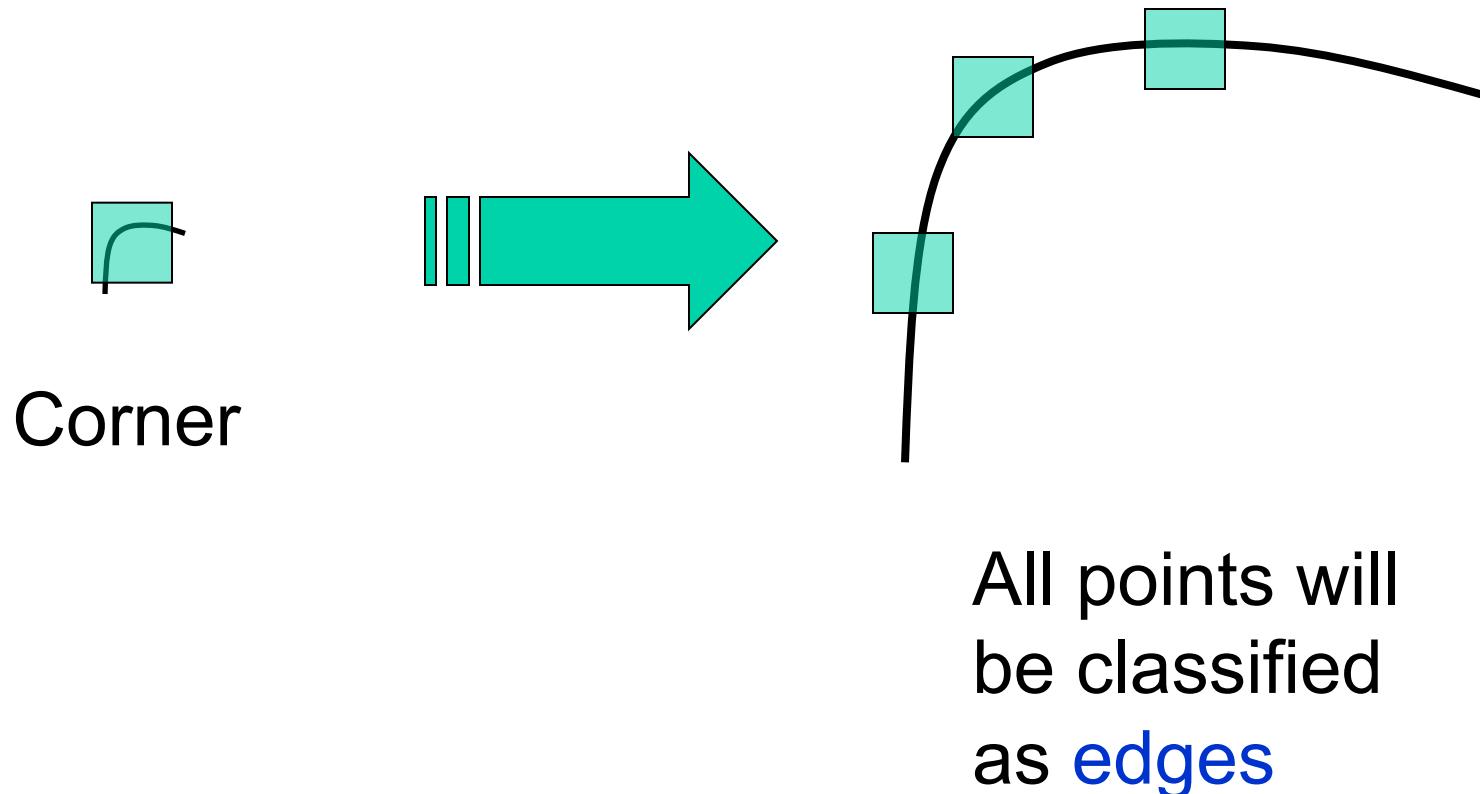
Image rotation



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

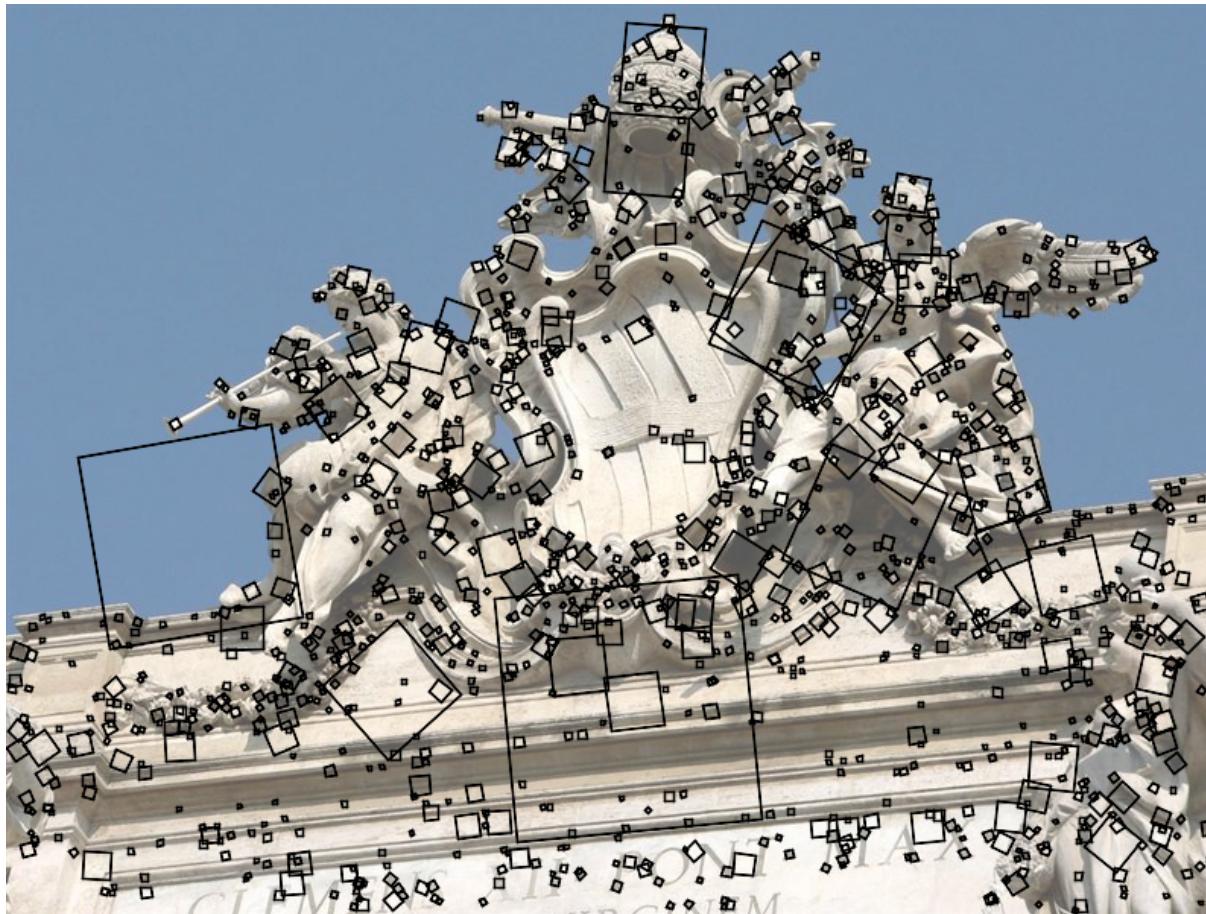
Corner location is covariant w.r.t. rotation

Scaling



Corner location is not covariant to scaling!

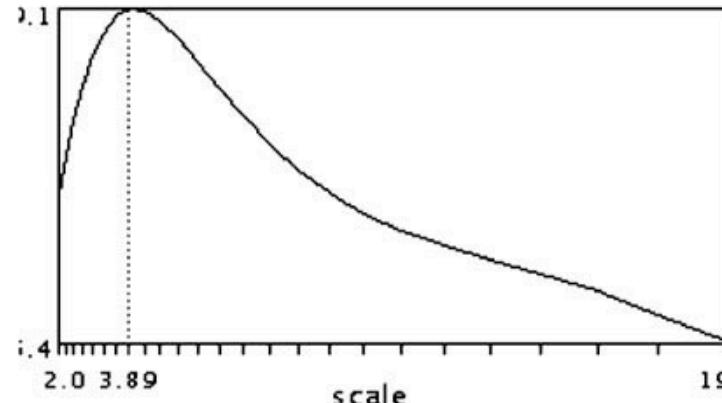
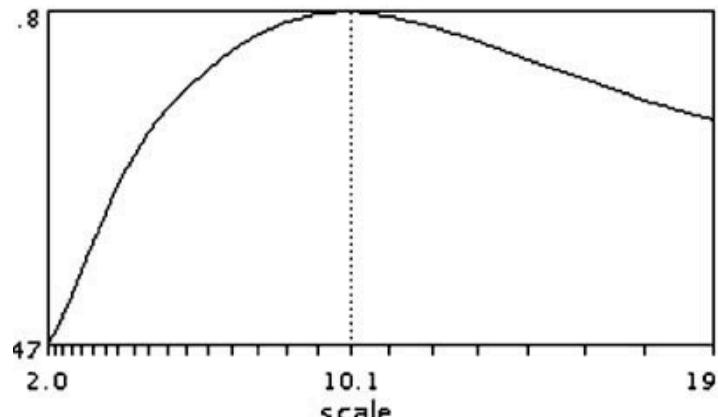
SIFT keypoint detection



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

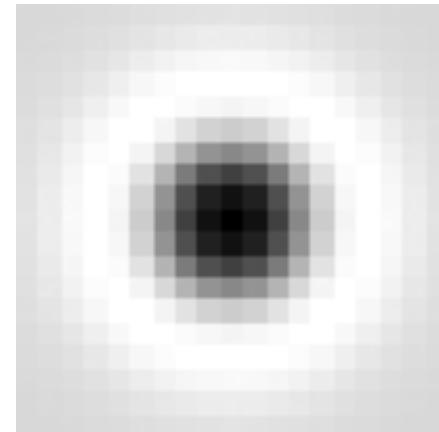
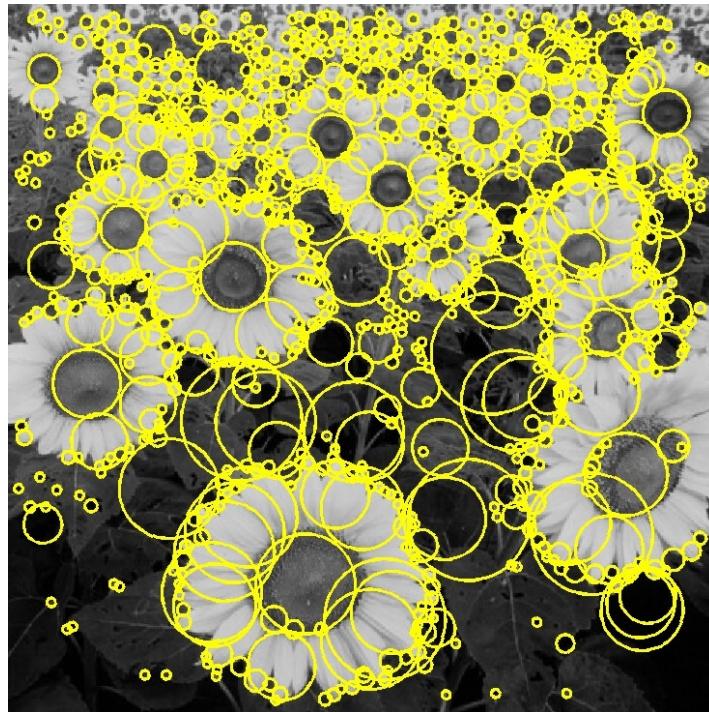
Keypoint detection with scale selection

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



Basic idea

- Convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting *scale space*

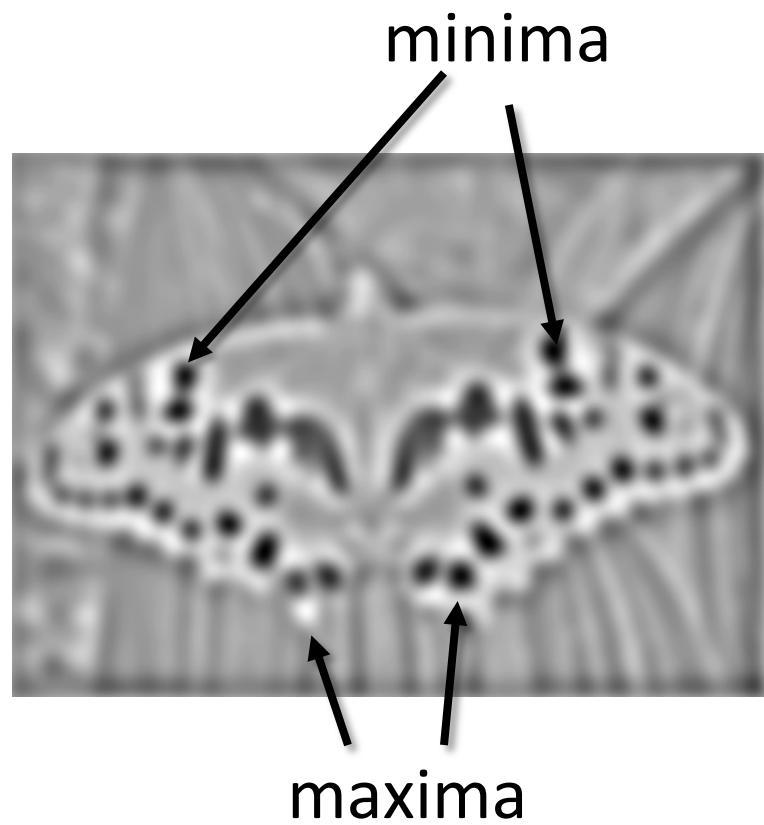


T. Lindeberg. [Feature detection with automatic scale selection.](#)
IJCV 30(2), pp 77-116, 1998.

Blob detection



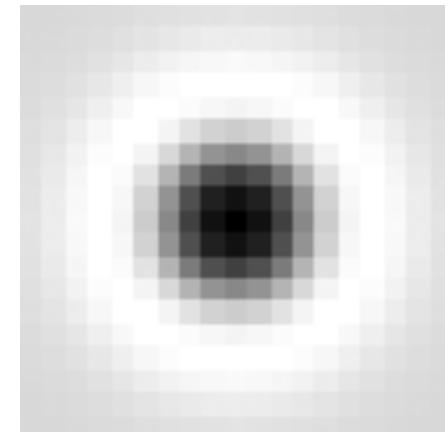
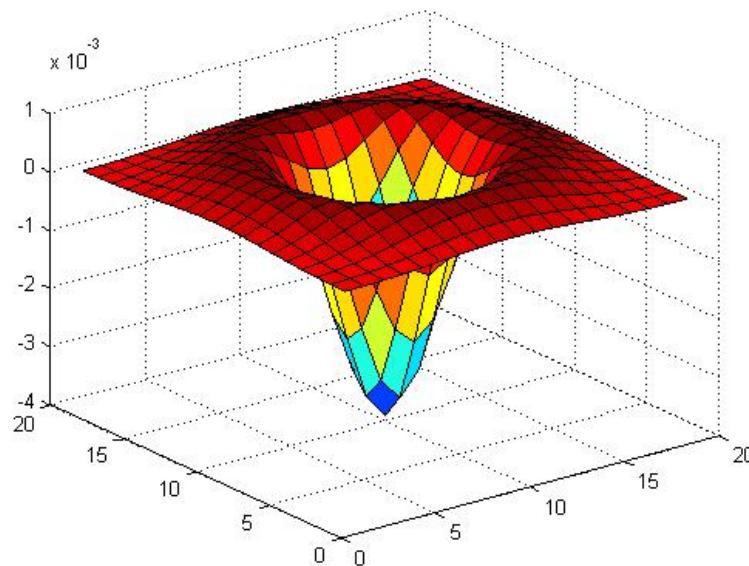
$$\ast \quad \bullet =$$



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

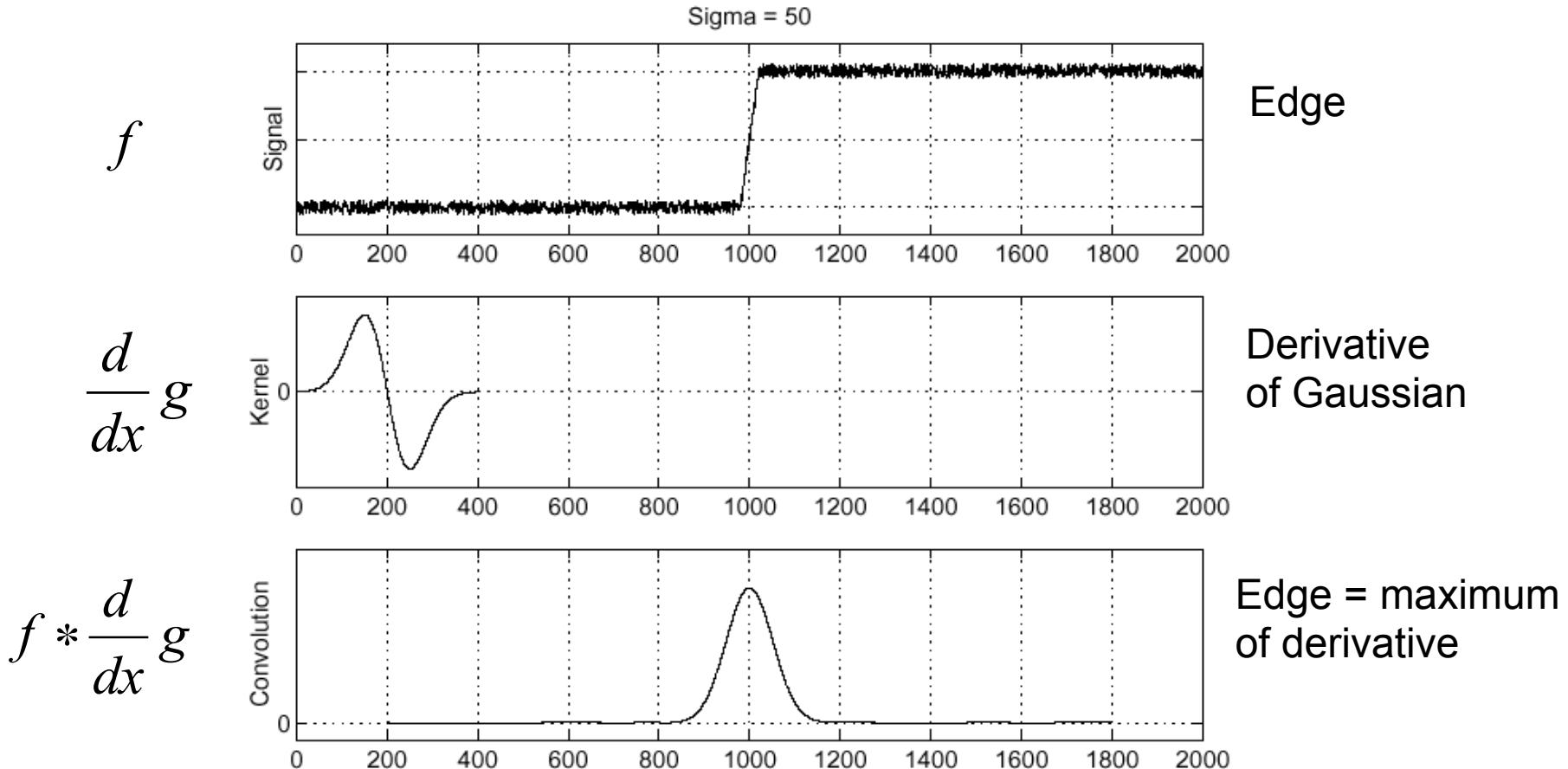
Blob filter

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

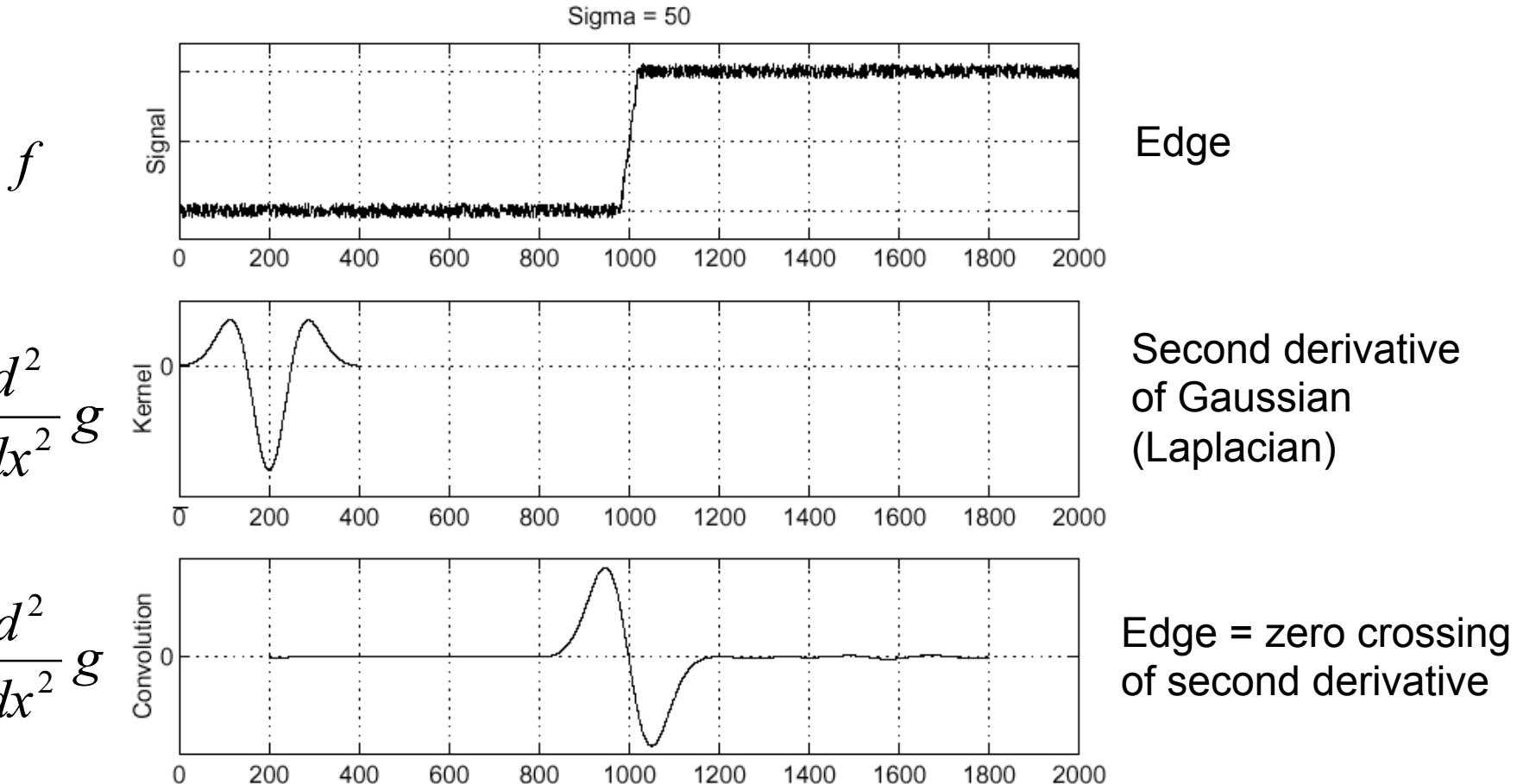


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Recall: Edge detection

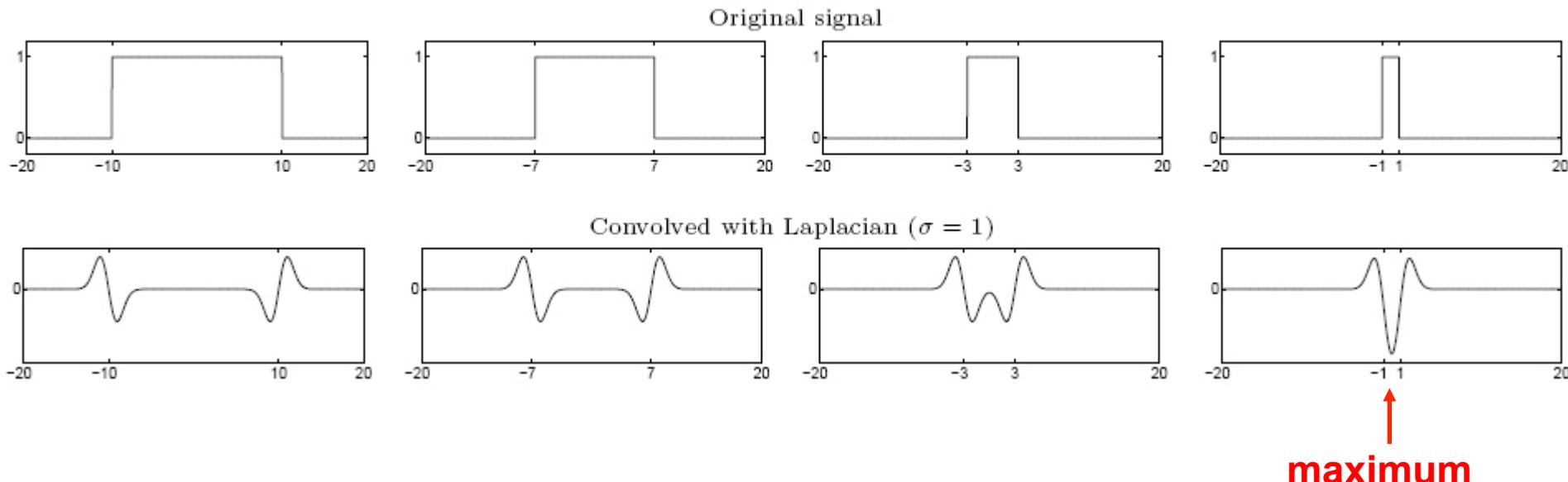


Edge detection, Take 2



From edges to blobs

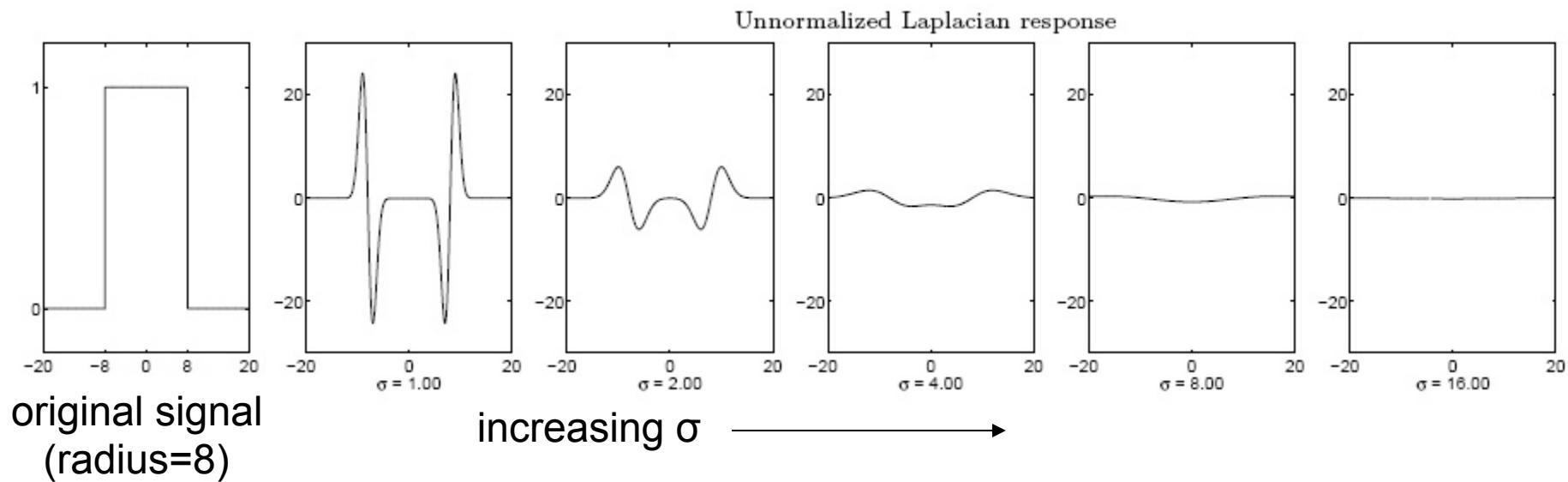
- Edge = ripple
- Blob = superposition of two ripples



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

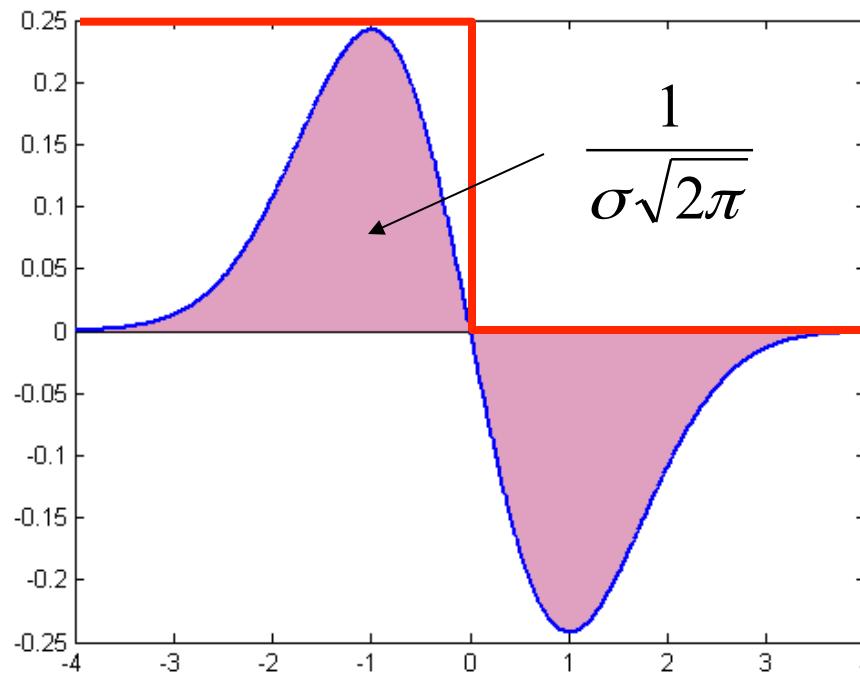
Scale selection

- 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 σ increases

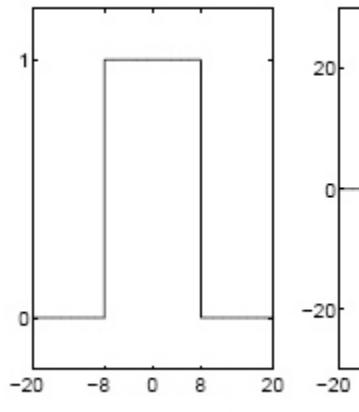


Scale normalization

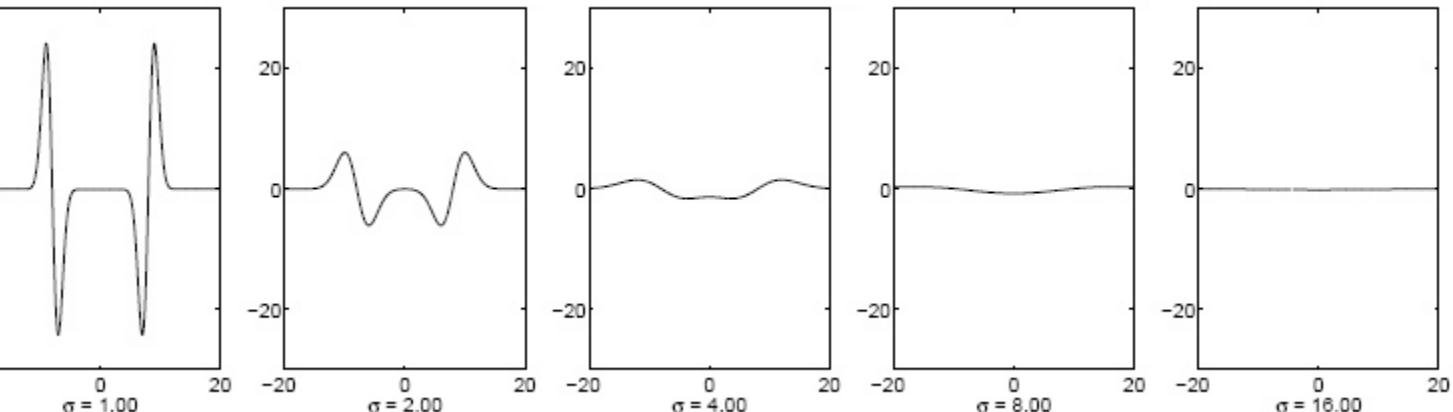
- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

Effect of scale normalization

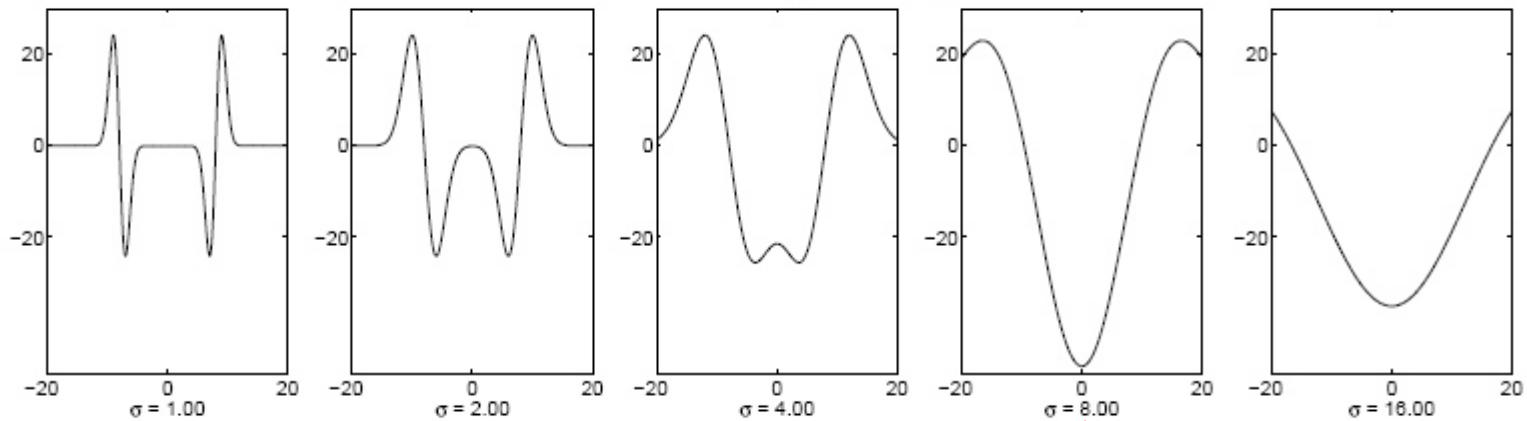
Original signal



Unnormalized Laplacian response



Scale-normalized Laplacian response

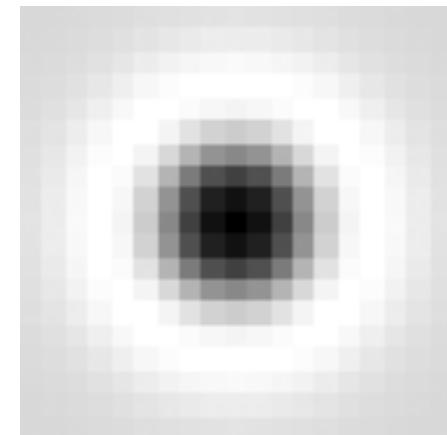
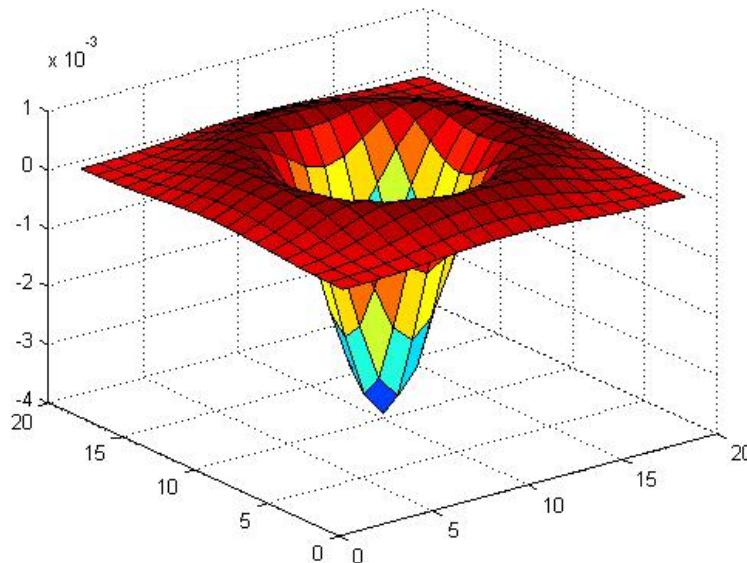


maximum

Blob detection in 2D

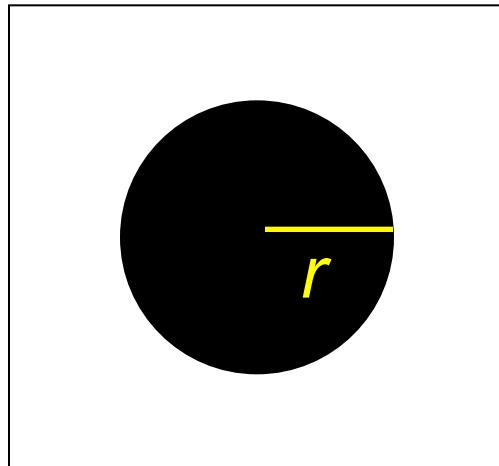
- *Scale-normalized Laplacian of Gaussian:*

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

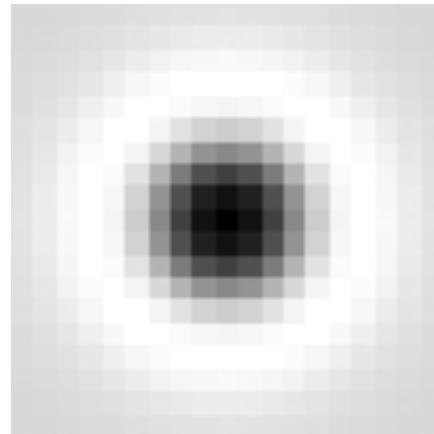


Blob detection in 2D

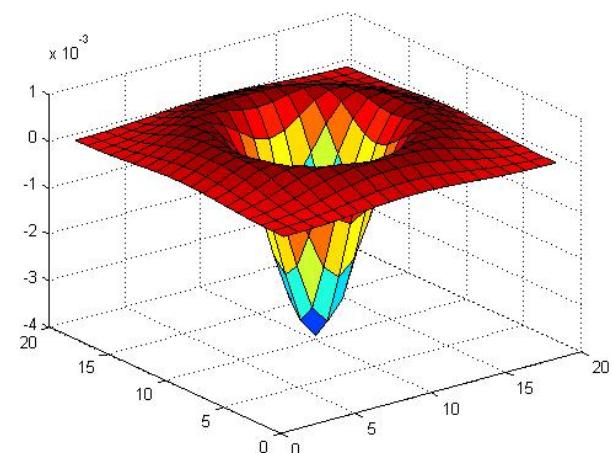
- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r ?



image

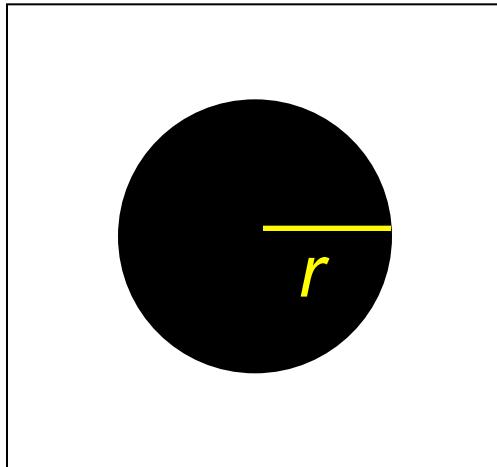


Laplacian

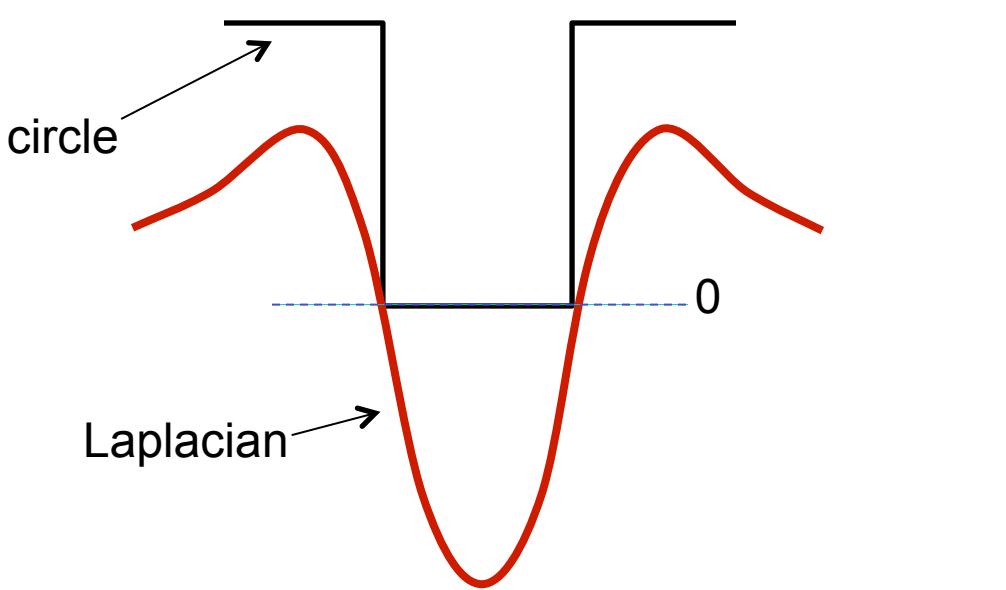


Blob detection in 2D

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



image



Scale-space blob detector

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

Scale-space blob detector: Example



Source: S. Lazebnik

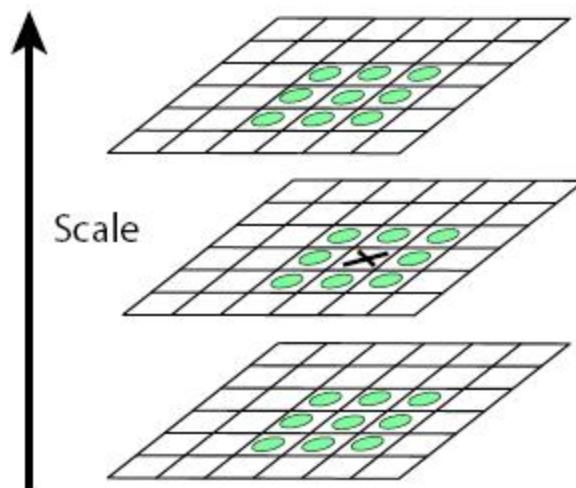
Scale-space blob detector: Example



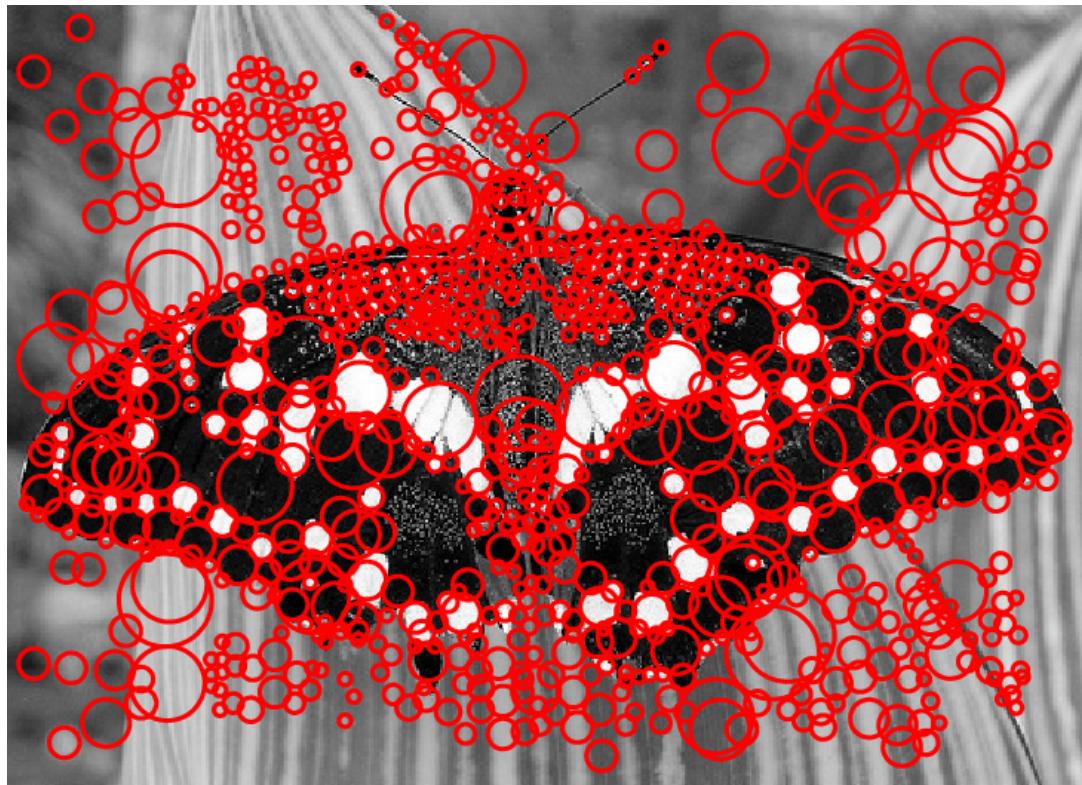
$\sigma = 11.9912$

Scale-space blob detector

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

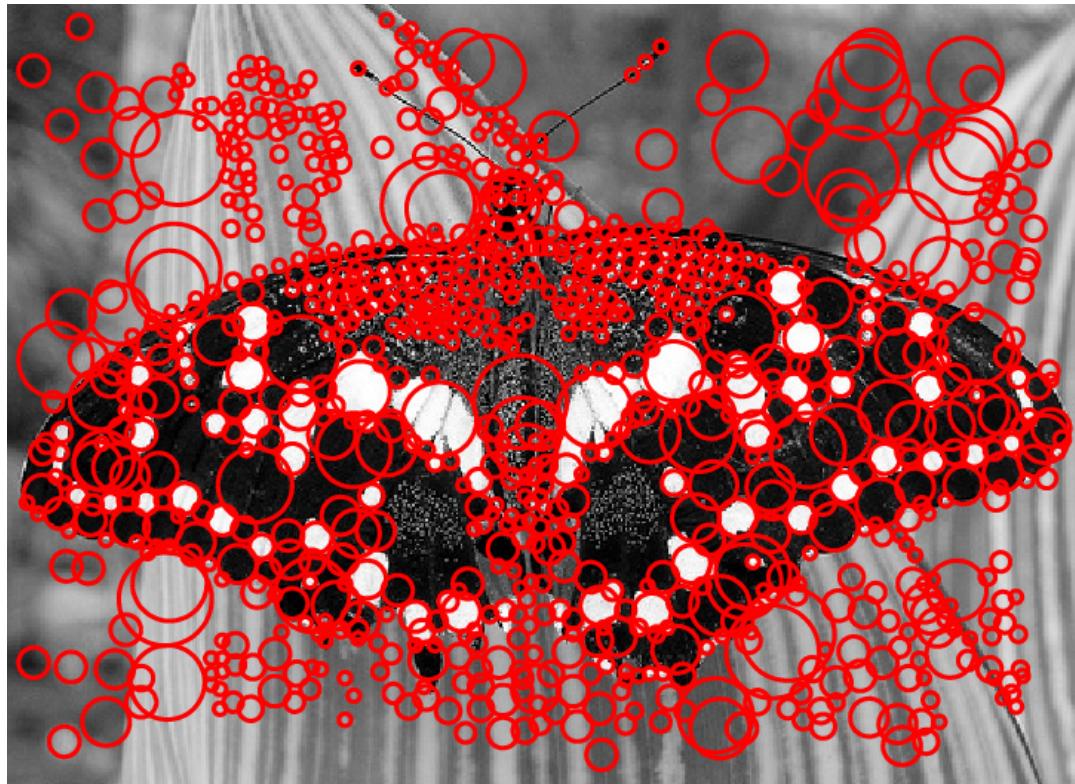


Scale-space blob detector: Example



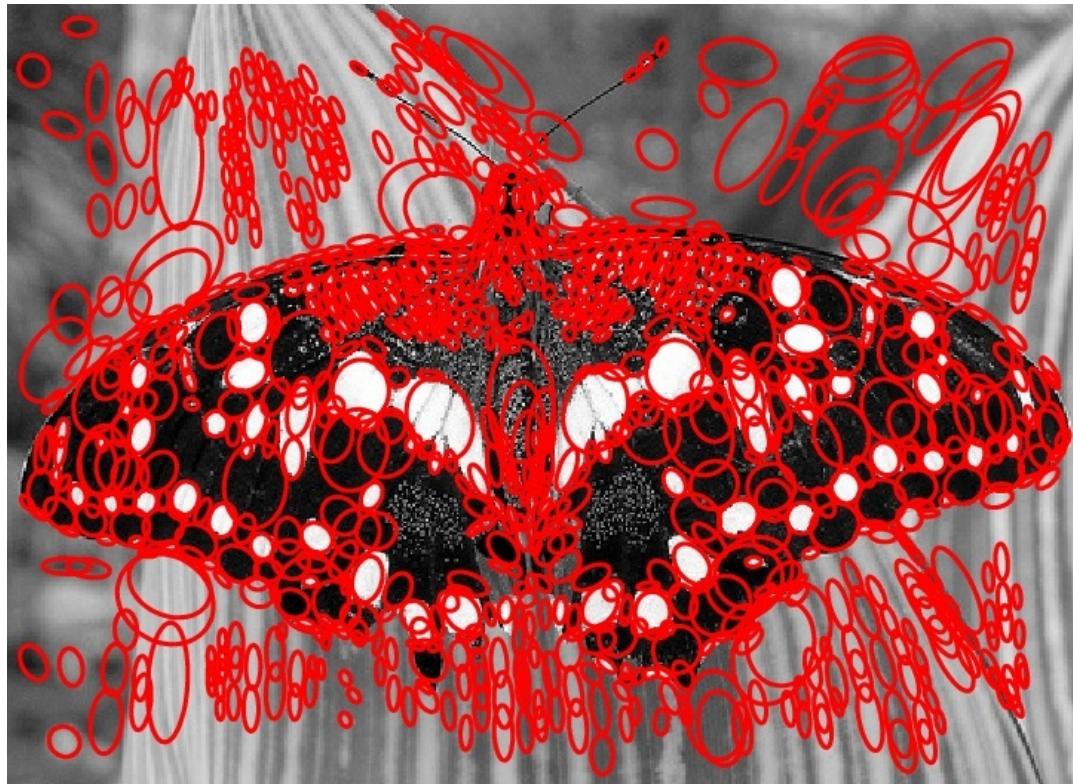
Eliminating edge responses

- Laplacian has strong response along edge



Eliminating edge responses

- Laplacian has strong response along edge



- Solution: filter based on Harris response function over neighborhoods containing the “blobs”

Efficient implementation

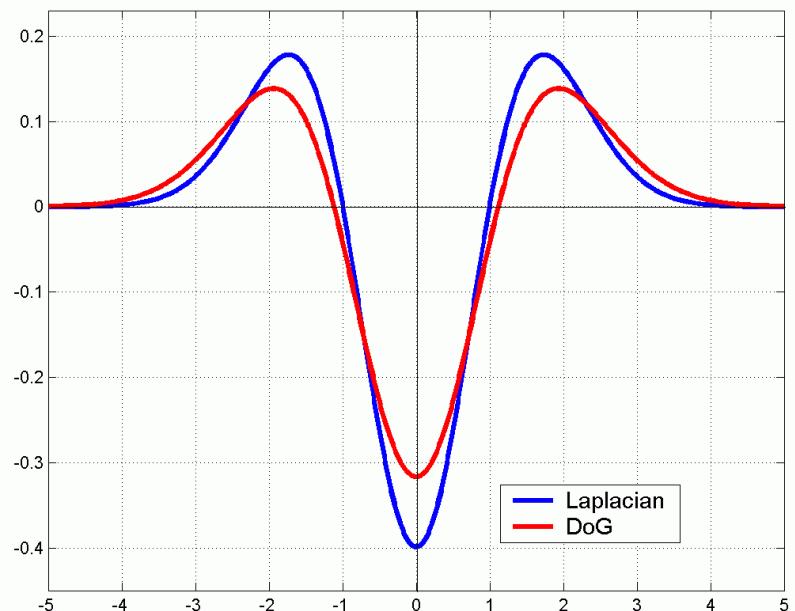
- Approximating the Laplacian with a difference of Gaussians:

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

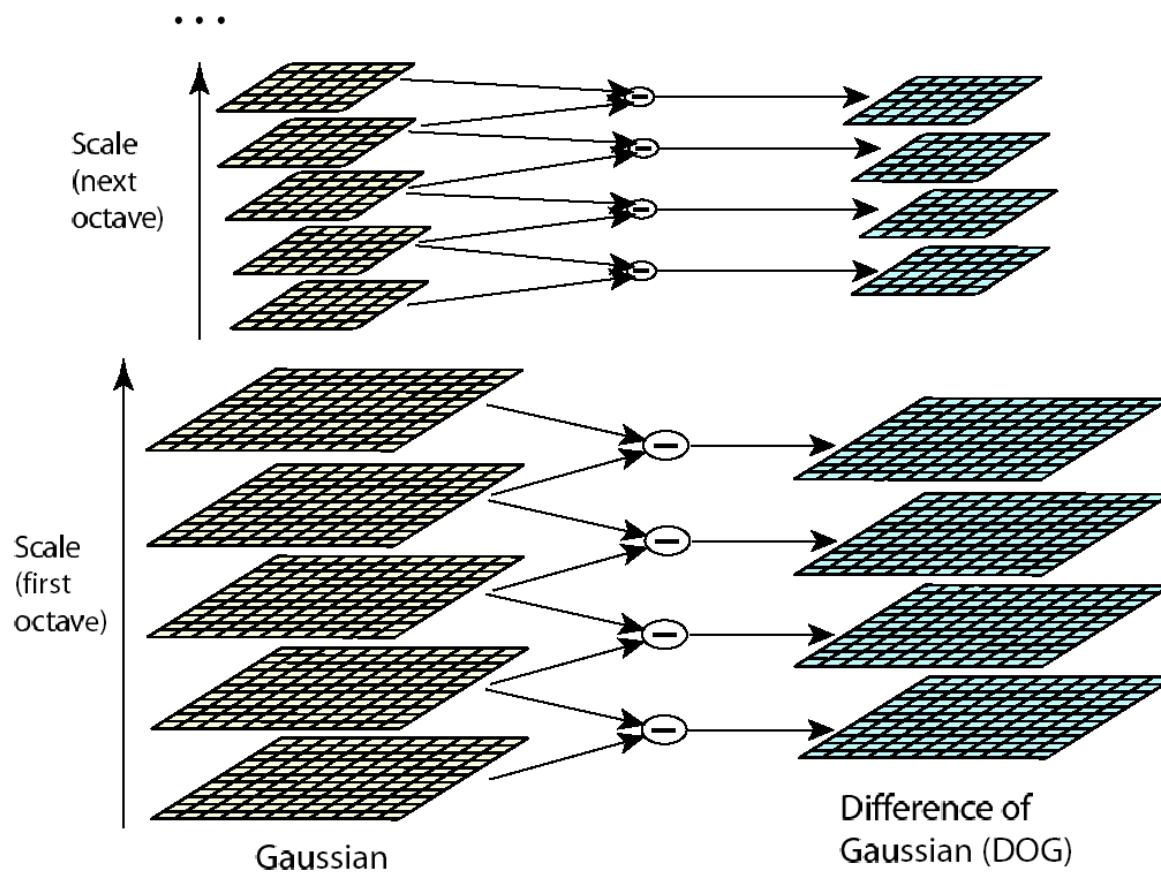
(Laplacian)

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

(Difference of Gaussians)



Efficient implementation



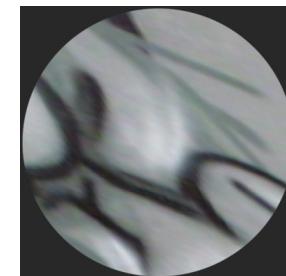
David G. Lowe.

"Distinctive image features from scale-invariant keypoints." IJCV 60
(2), pp. 91-110, 2004.

Source: S. Lazebnik

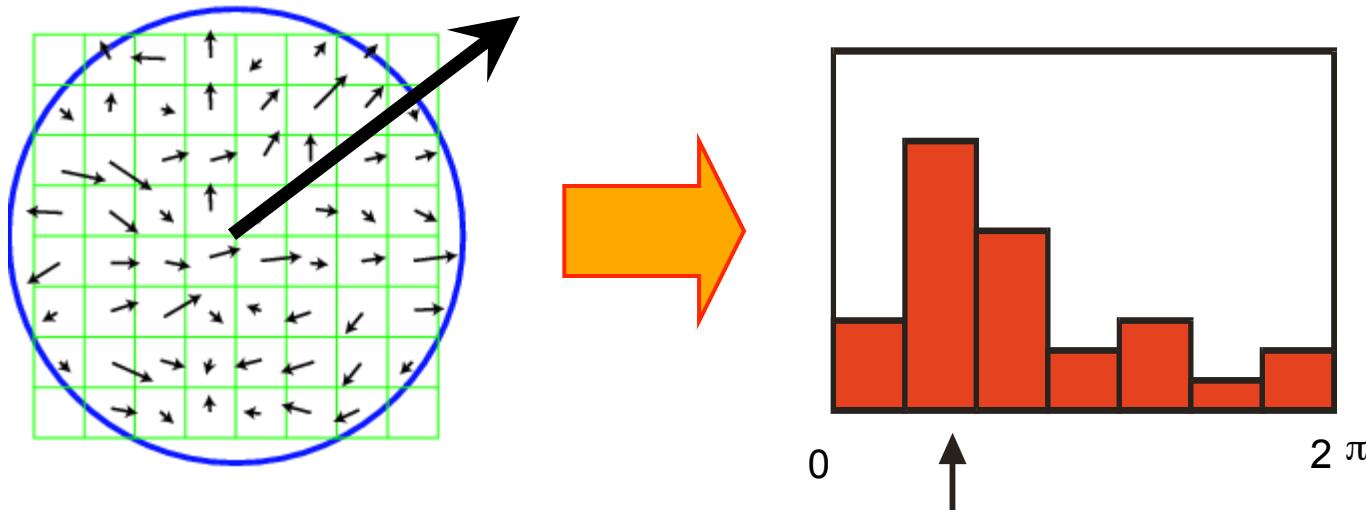
From feature detection to feature description

- 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



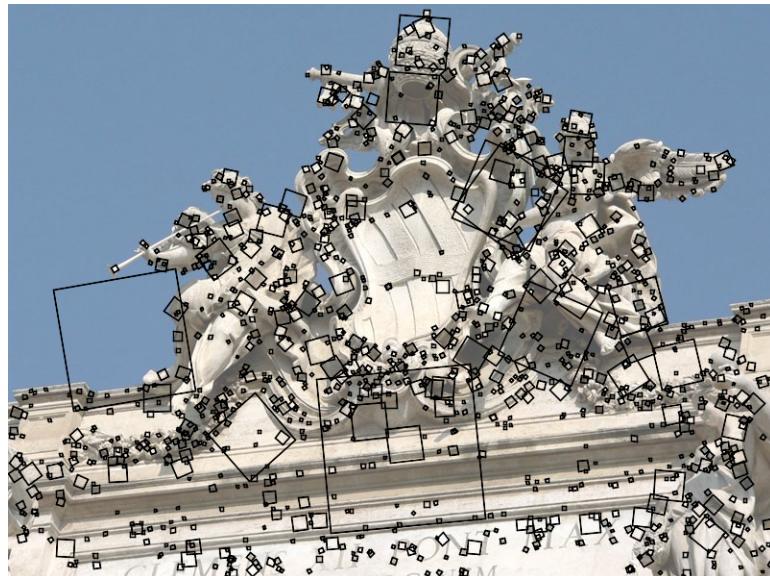
Eliminating rotation ambiguity

- 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:



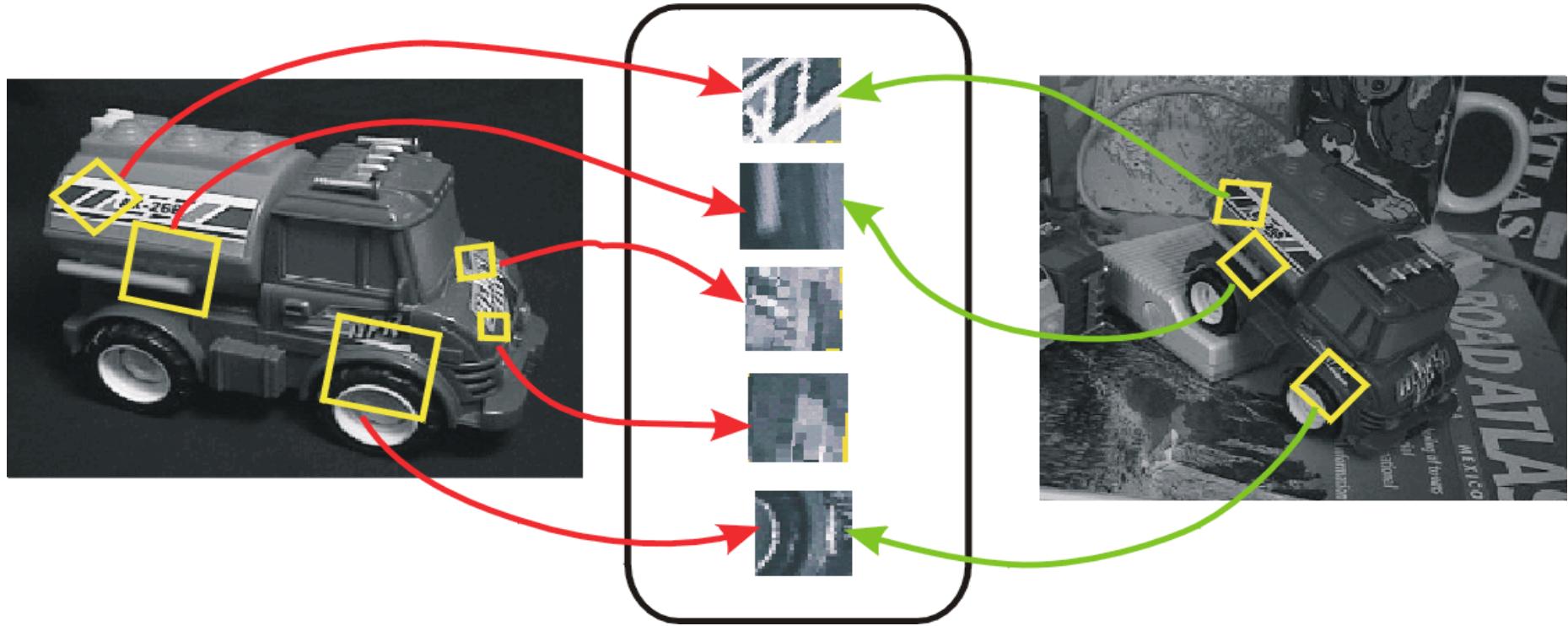
David G. Lowe.

["Distinctive image features from scale-invariant keypoints."](#) IJCV 60

(2), pp. 91-110, 2004.

Source: S. Lazebnik

From feature detection to feature description



Detection is *covariant*:

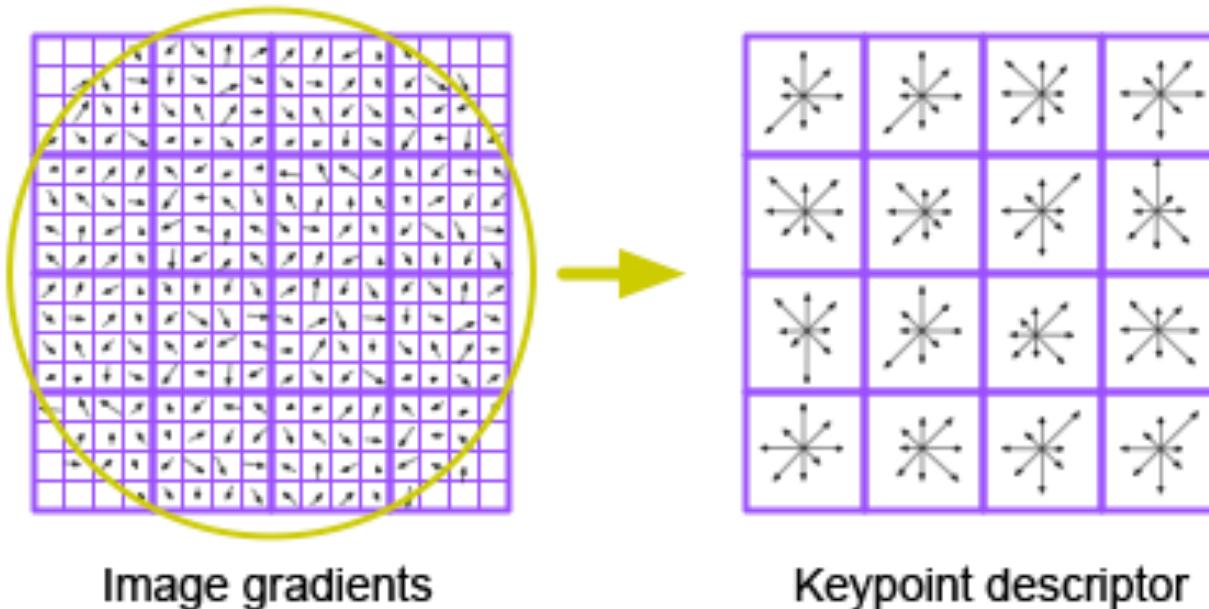
$$\text{features}(\text{transform}(\text{image})) = \text{transform}(\text{features}(\text{image}))$$

Description is *invariant*:

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

SIFT descriptors

- Inspiration: complex neurons in the primary visual cortex



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

Source: S. Lazebnik

SIFT descriptor computation

- use the **normalized** region about the keypoint
- compute gradient magnitude and orientation at each point in the region
- weight them by a **Gaussian window** overlaid on the circle
- create an **orientation histogram** over the 4×4 subregions of the window
- 4×4 descriptors over 16×16 sample array were used in practice. 4×4 times 8 directions gives a **vector of 128 values**.



Summary of SIFT feature detection and description

1. Scale-space extrema detection

Search over multiple scales and image locations.

2. Keypoint localization

Fit a model to determine location and scale.

Select keypoints based on a measure of stability.

3. Orientation assignment

Compute best orientation(s) for each keypoint region.

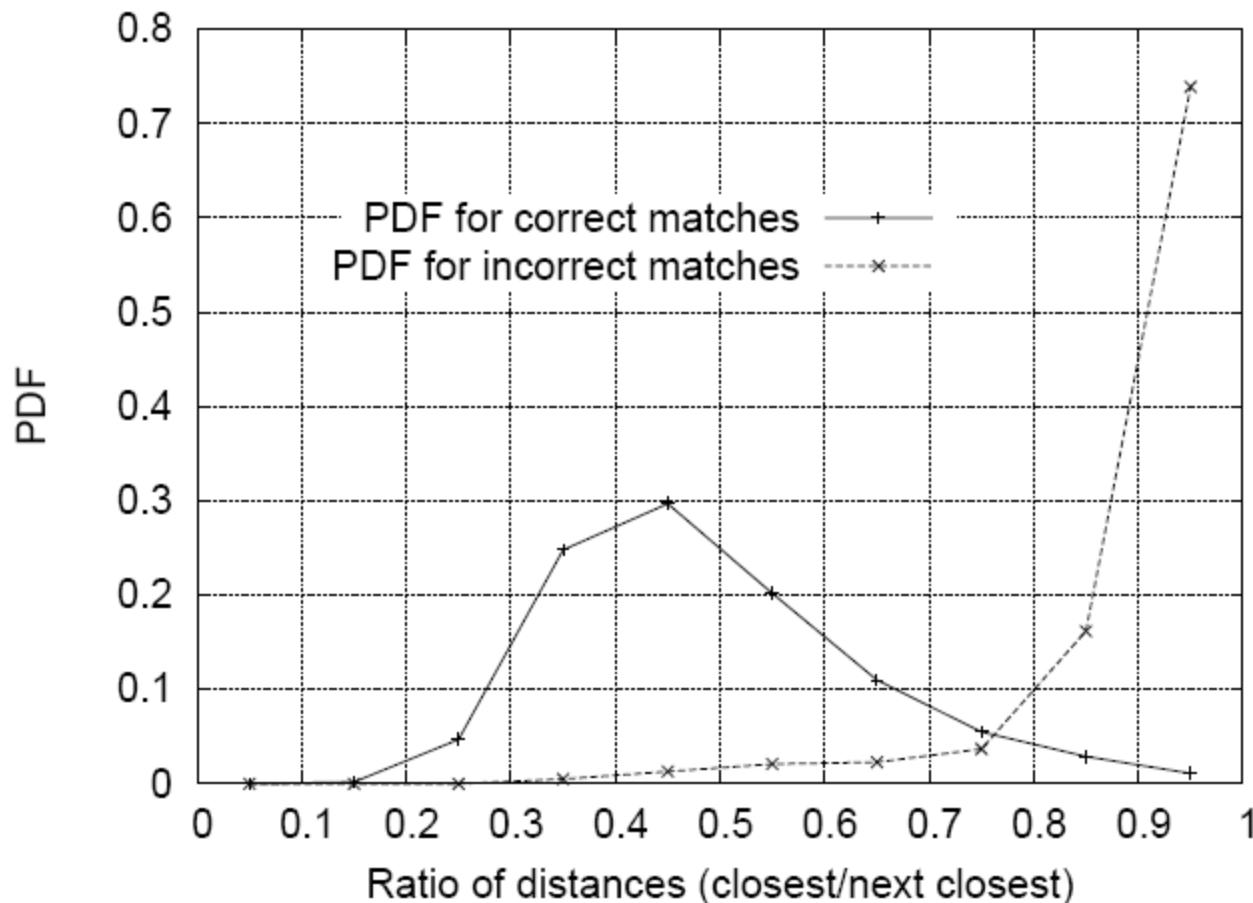
4. Keypoint description

Use local image gradients at selected scale and rotation to describe each keypoint region.

Matching SIFT Descriptors

Nearest neighbor (Euclidean distance)

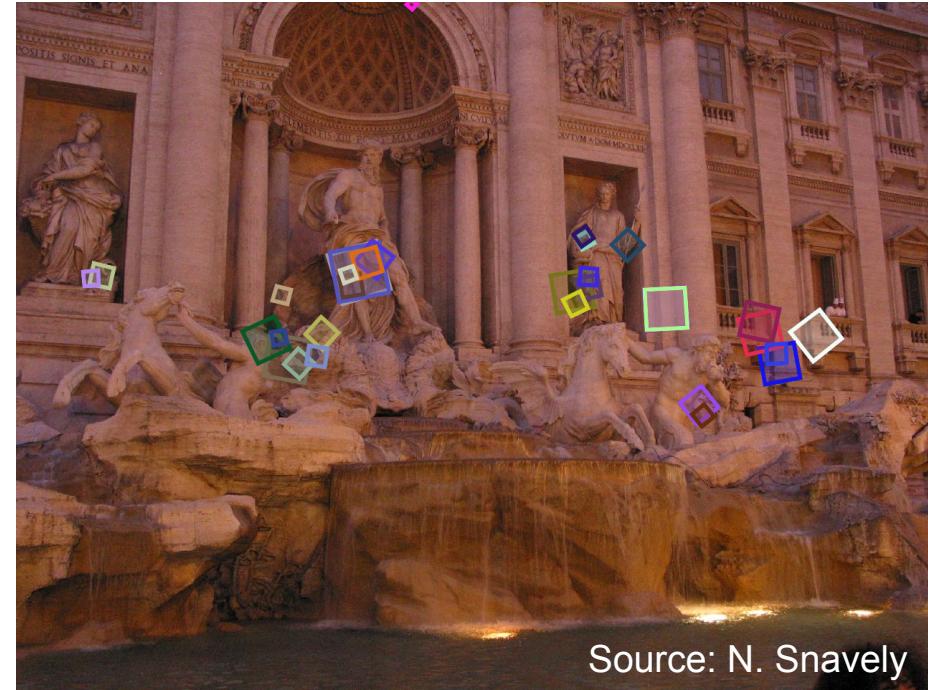
Threshold ratio of nearest to 2nd nearest descriptor



Properties of SIFT

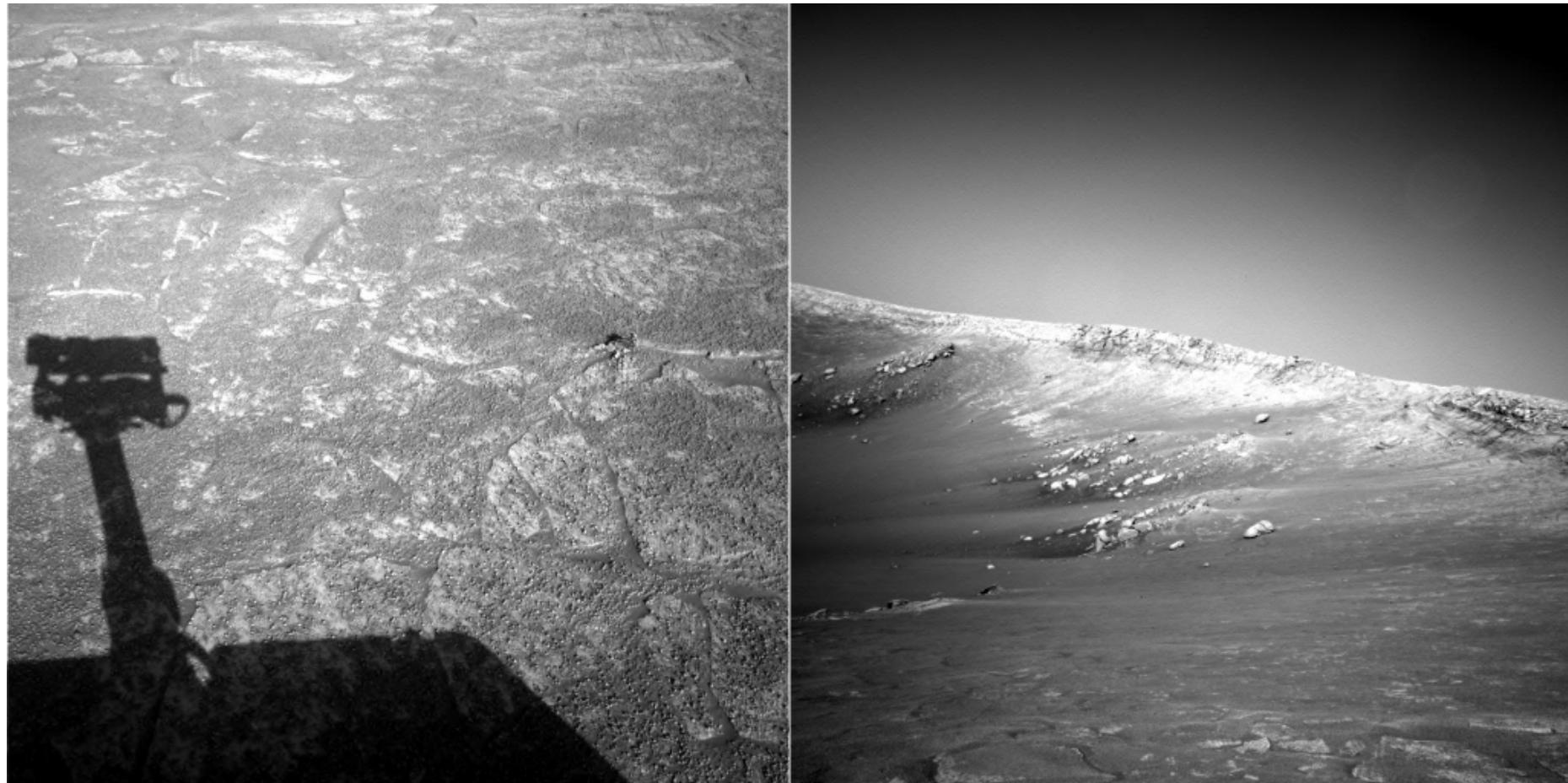
Extraordinarily robust detection and description technique

- 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



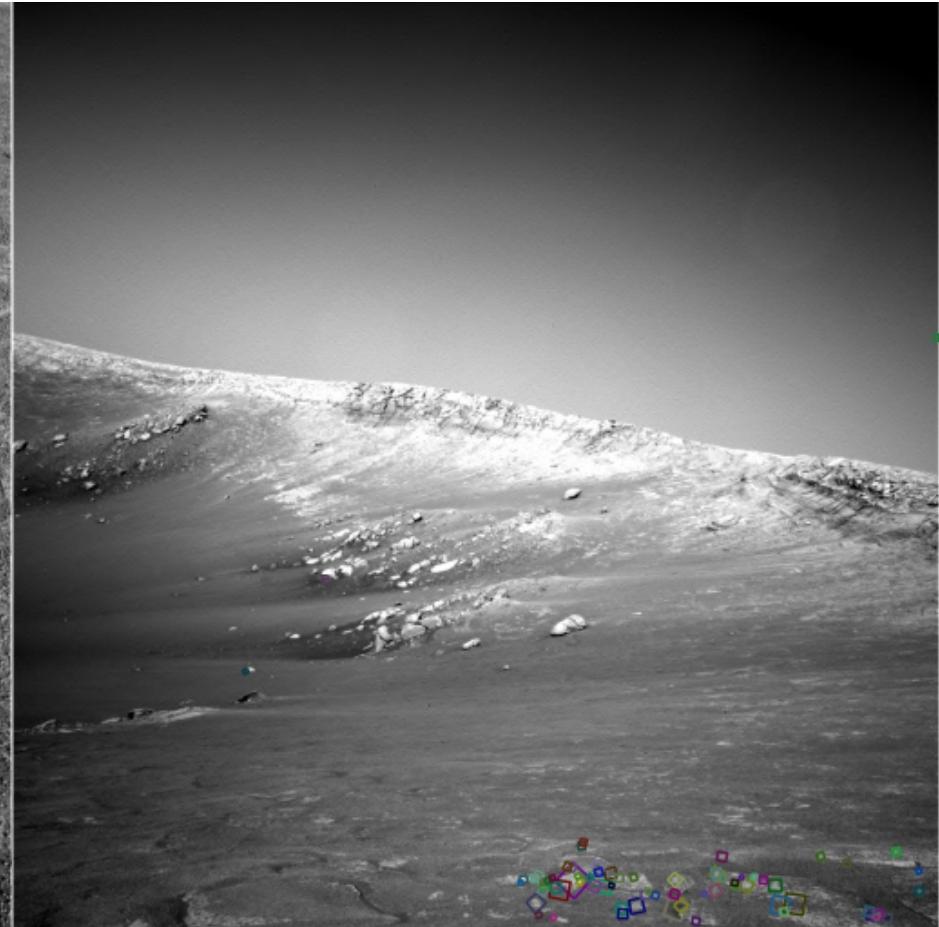
Source: N. Snavely

A hard keypoint matching problem



NASA Mars Rover images

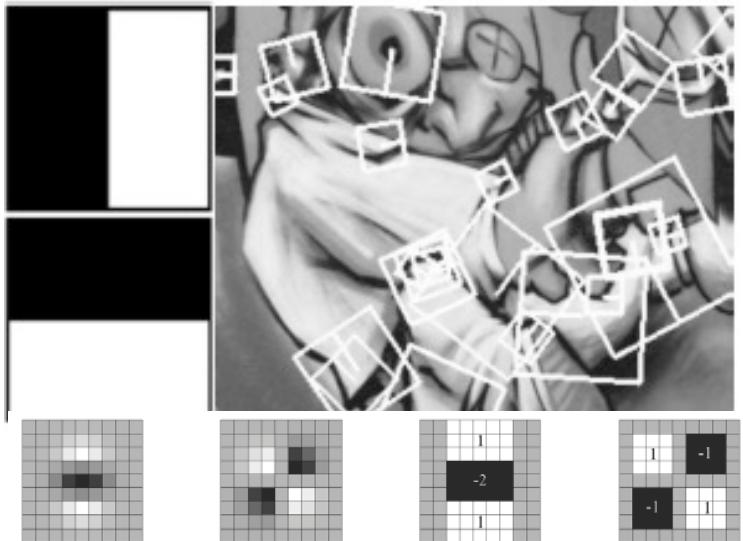
Answer below (look for tiny colored squares...)



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

Source: S. Lazebnik

Local Descriptors: SURF



Fast approximation of SIFT idea

**Efficient computation by 2D box filters
& integral images**

⇒ 6 times faster than SIFT

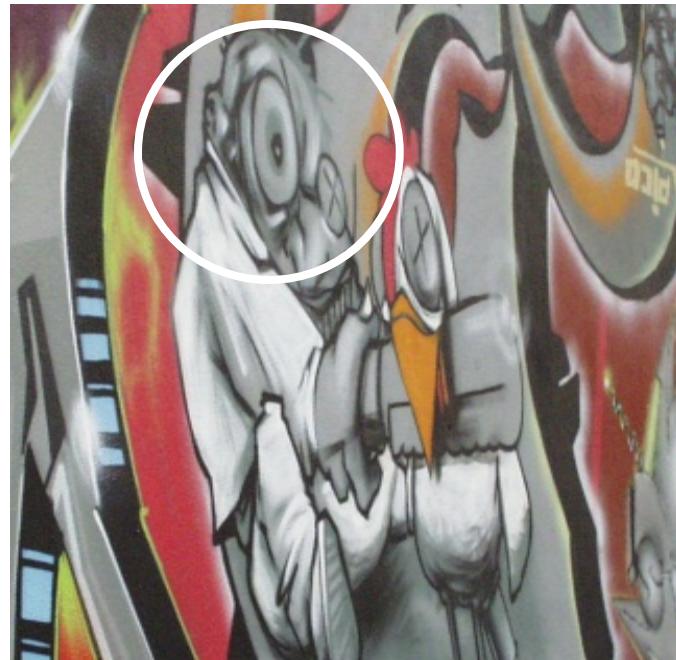
**Equivalent quality for object
identification**

Many other efficient descriptors
are also available

GPU implementation available

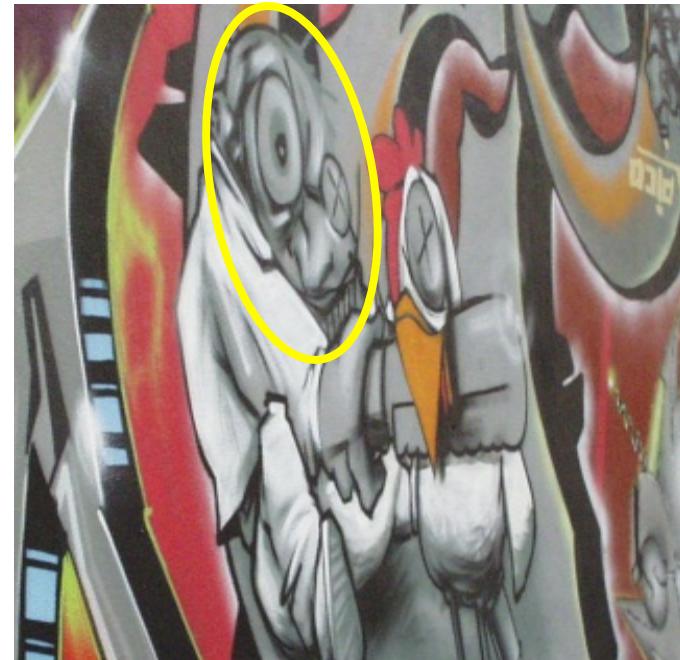
**Feature extraction @ 200Hz
(detector + descriptor, 640×480 img)**
<http://www.vision.ee.ethz.ch/~surf>

What about 3D rotations?



What about 3D rotations?

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



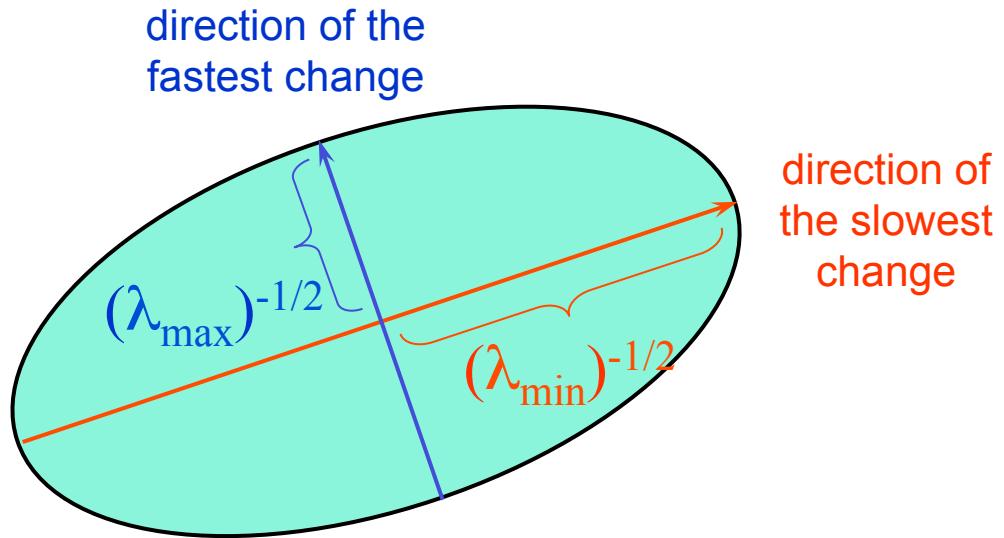
Affine adaptation

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

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

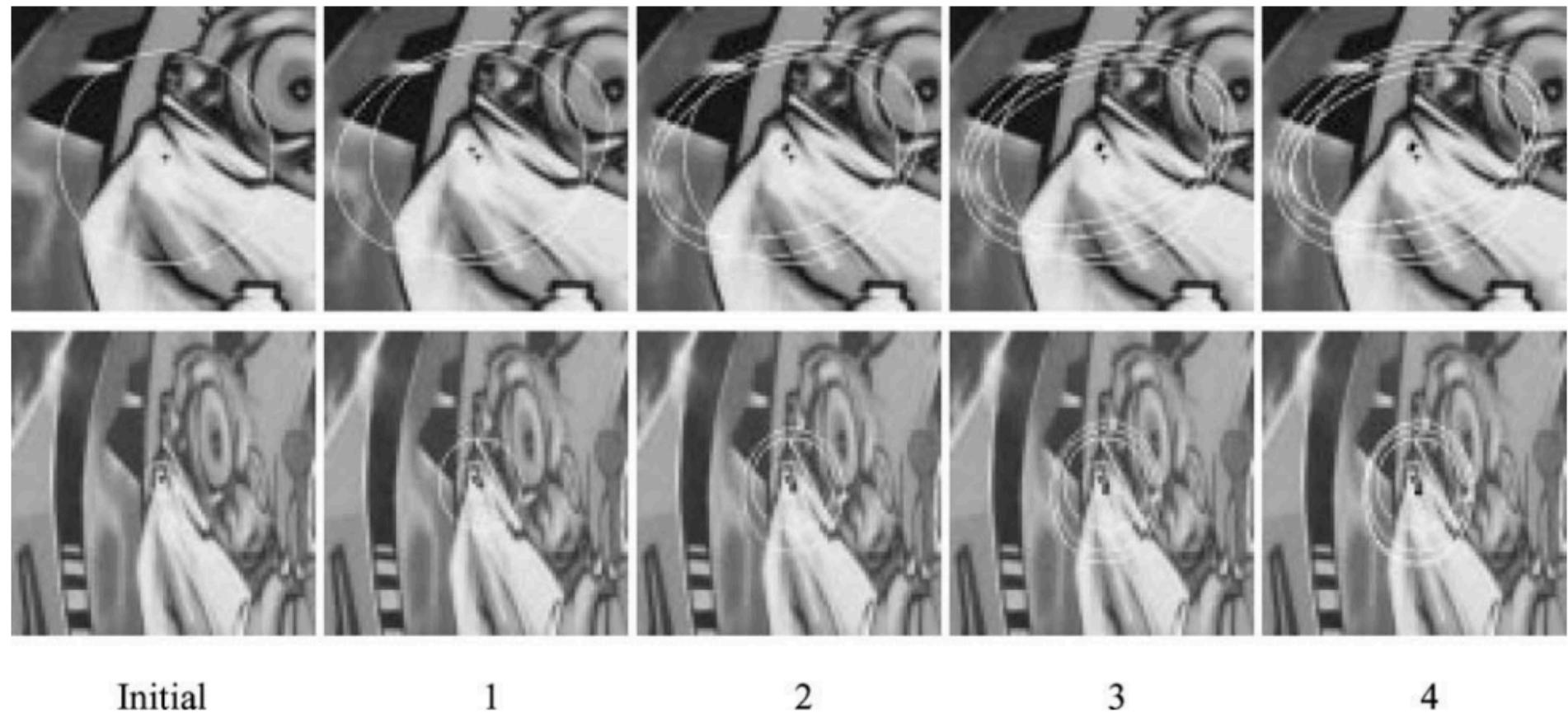
Recall:

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



This ellipse visualizes the “characteristic shape” of the window

Affine adaptation



K. Mikolajczyk and C. Schmid.

[Scale and affine invariant interest point detectors. IJCV 60\(1\):63-86, 2004.](#)

Thank you

Next lecture: Model estimation