计算机视觉

Computer Vision

Lecture 5: Local Image Features (1)

张 超

信息科学技术学院 智能科学系

Local Image Features

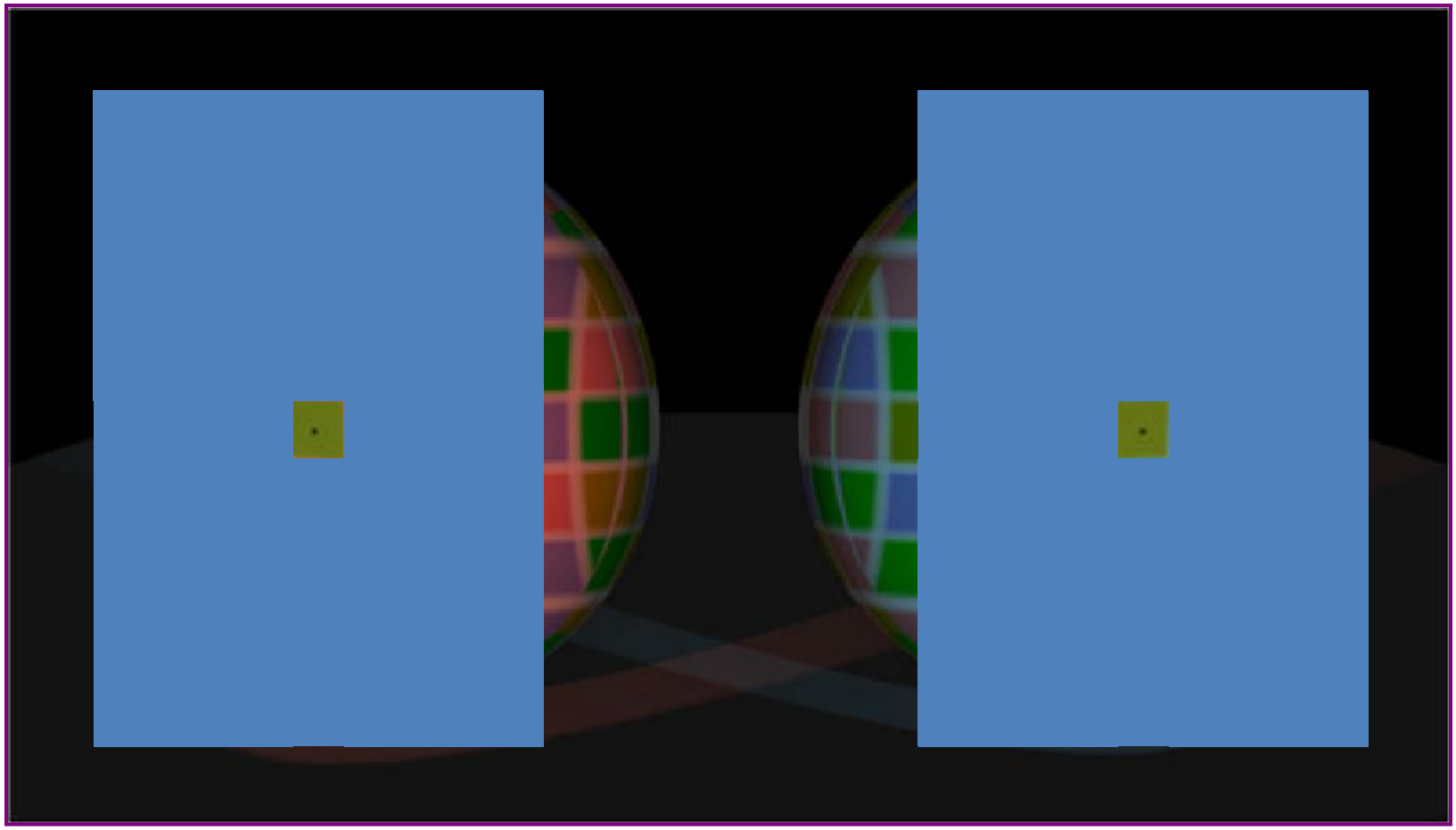
- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT

Today's Question

- What is a feature?
- How can we find edges?
- How can we find corners?
- How can we find blobs

- (How can we find cars in images?)

What is a Feature?

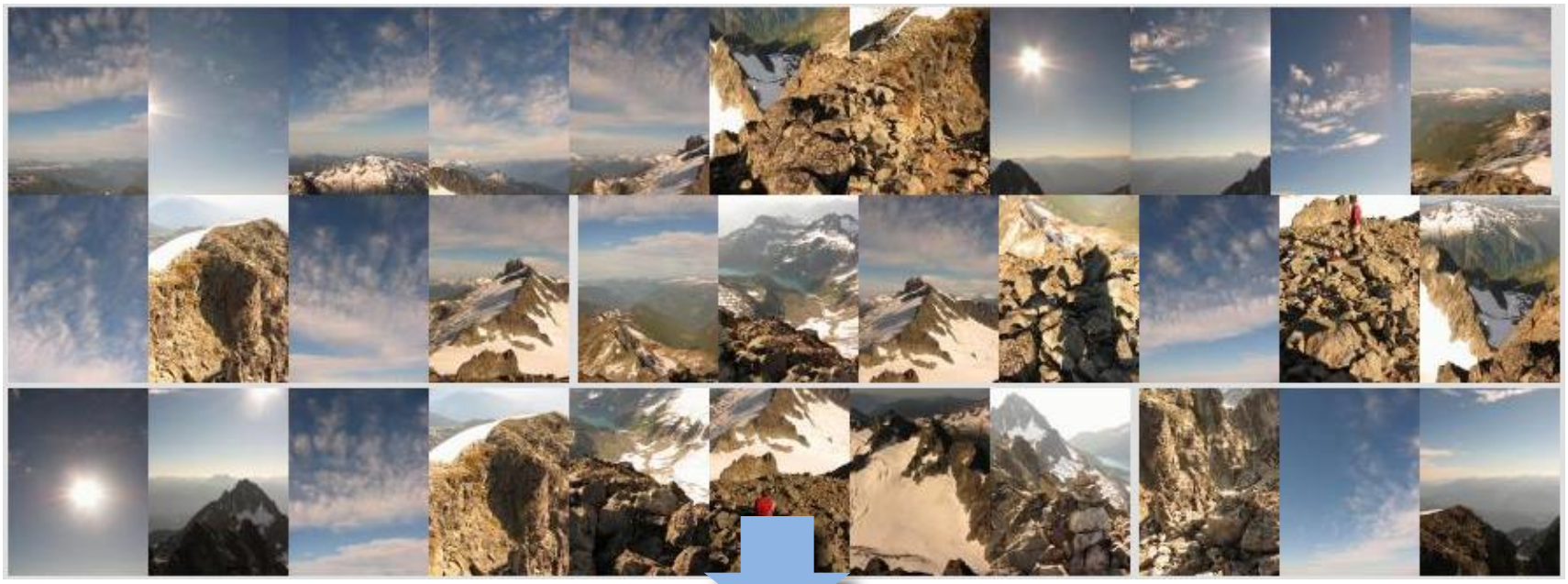


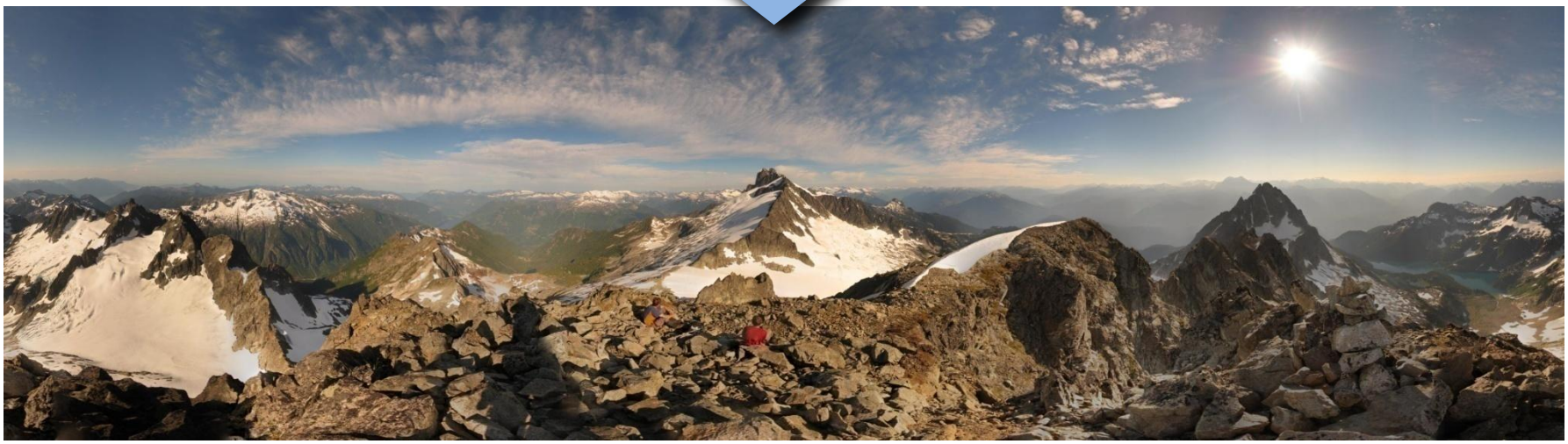
- **Local, meaningful, detectable parts of the image**

Features in Computer Vision

- **What is a feature?**
  - **Location of sudden change**
- **Why use features?**
  - **Information content high**
  - **Invariant to change of view point, illumination**
  - **Reduces computational burden**

Motivation: Automatic panoramas





Credit: Matt Brown

Motivation: Automatic panoramas

HD View
http://research.microsoft.com/en-us/um/redmond/groups/ivm/HDView/HDGigapixel.htm

Also see GigaPan:

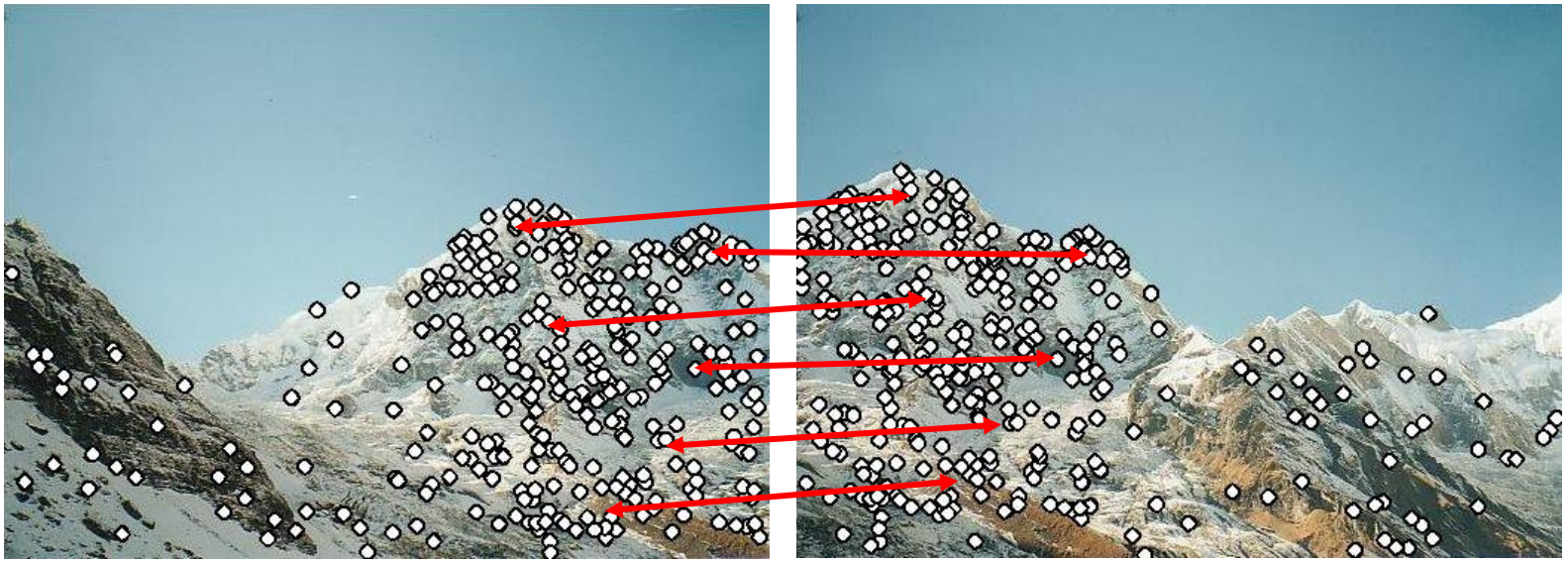
http://gigapan.org/

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

Image matching

by Diva Sian

by swashford

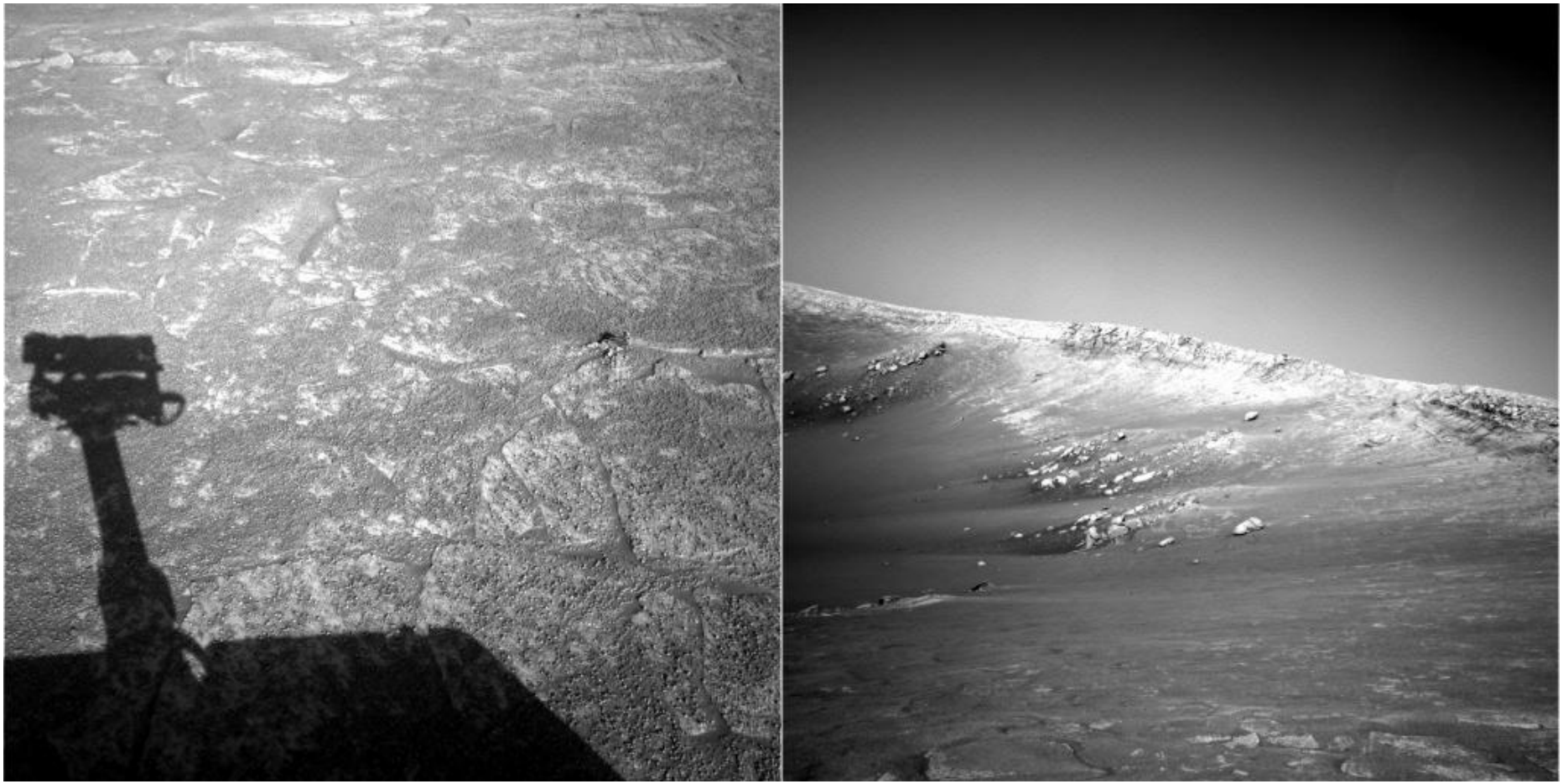
Trevi/Fontana di Trevi

Harder case

by Diva Sian

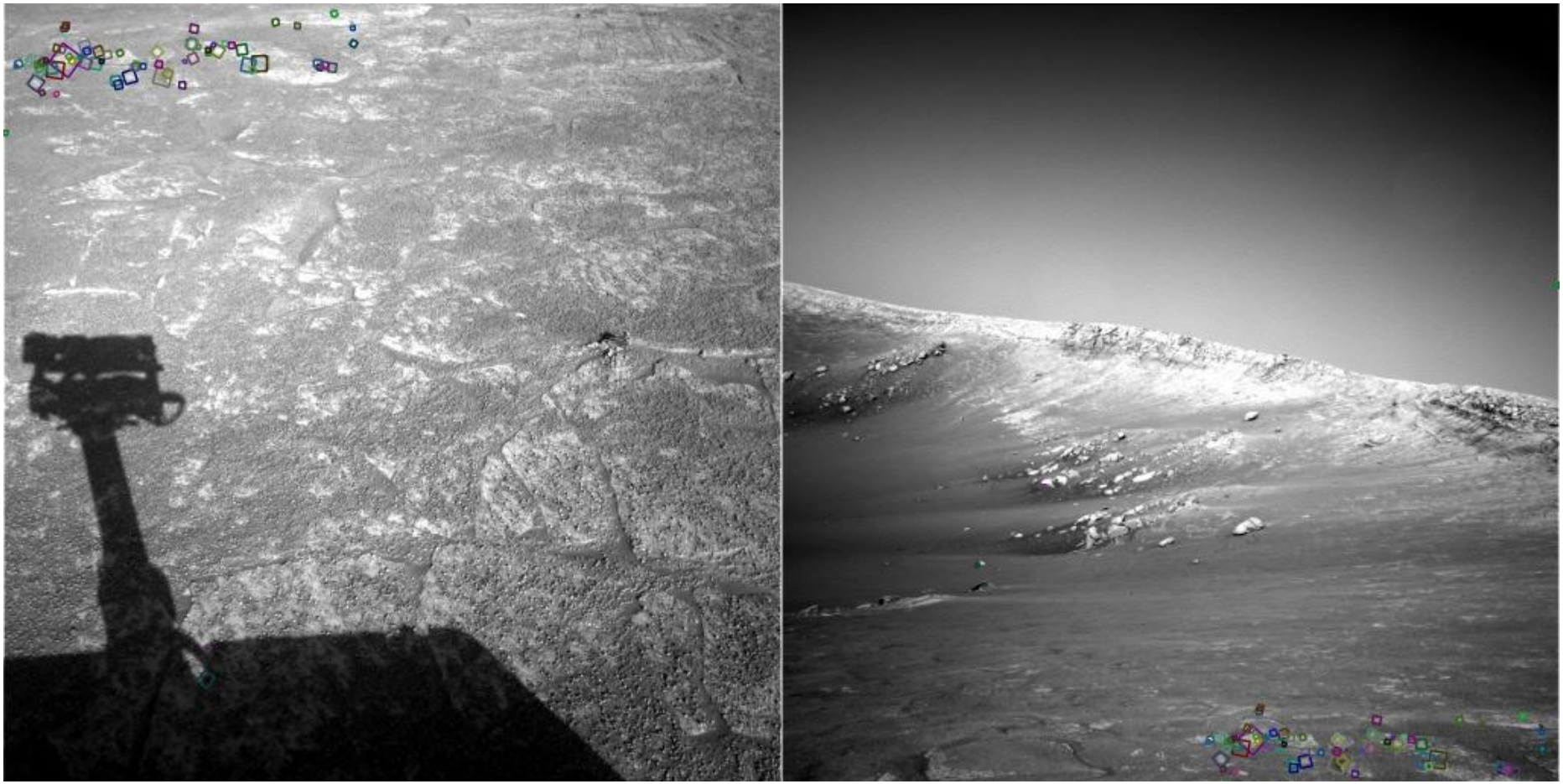
by scgbt

Harder still?



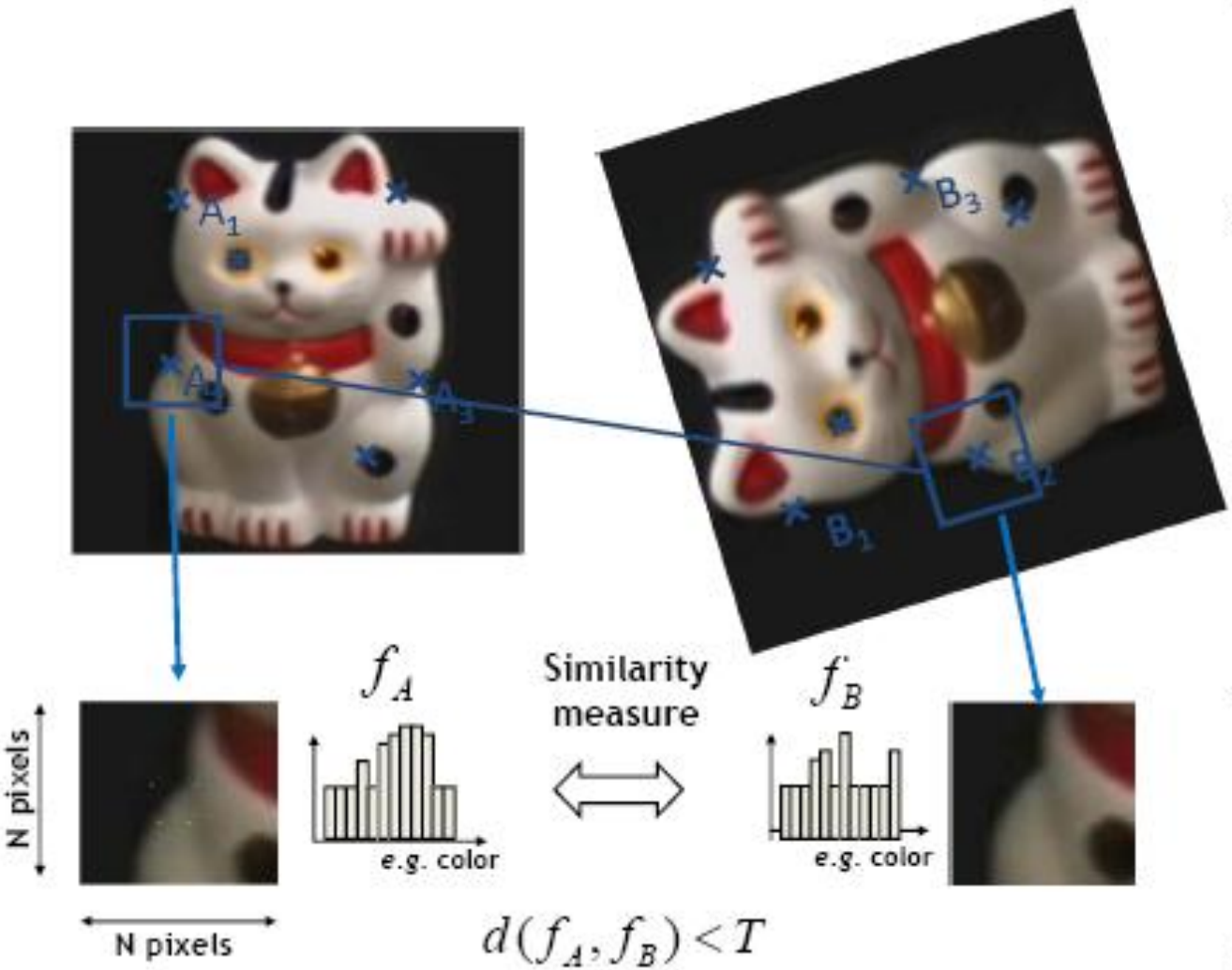
NASA Mars Rover images

Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches

General Approach



1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Common Requirements

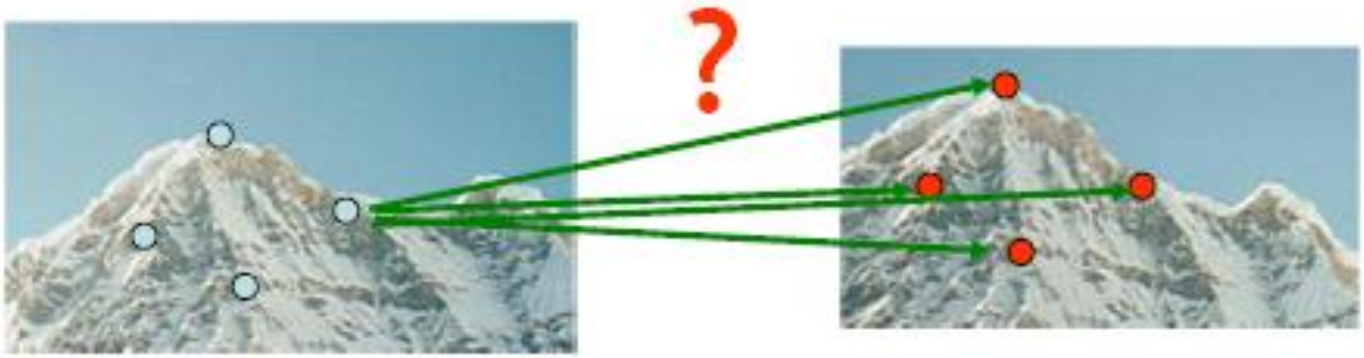
- Problem 1:
  - Detect the same point *independently* in both images

No chance to match!

We need a repeatable detector!

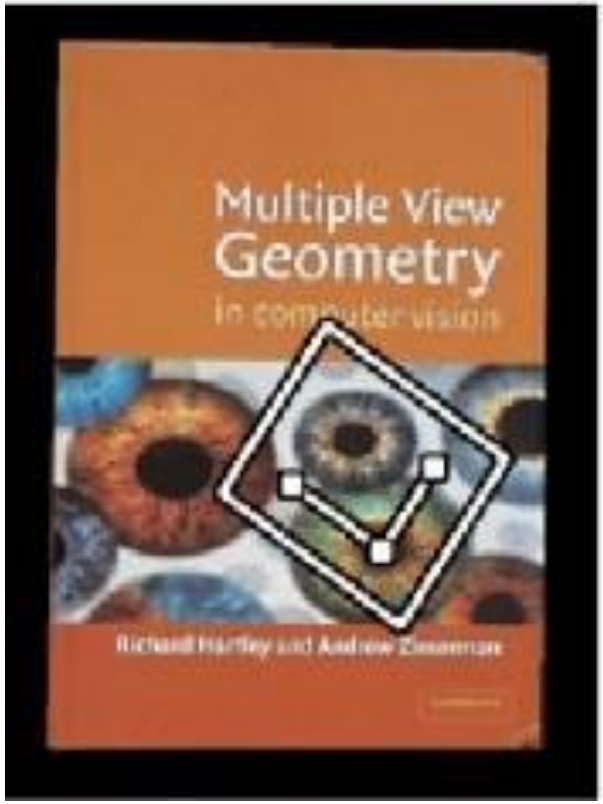
Common Requirements

- Problem 1:
  - Detect the same point *independently* in both images
- Problem 2:
  - For each point correctly recognize the corresponding one

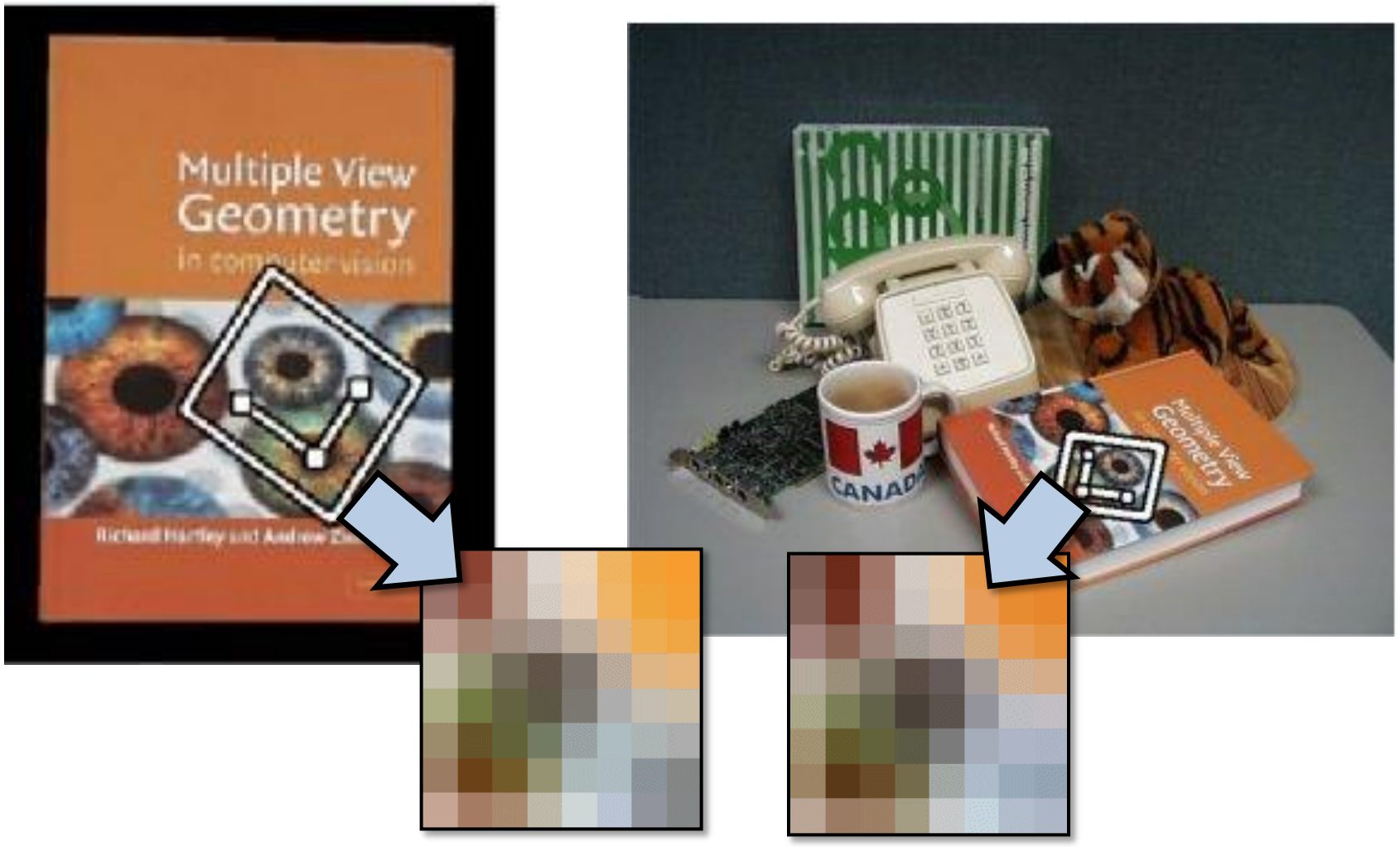


We need a reliable and distinctive descriptor!

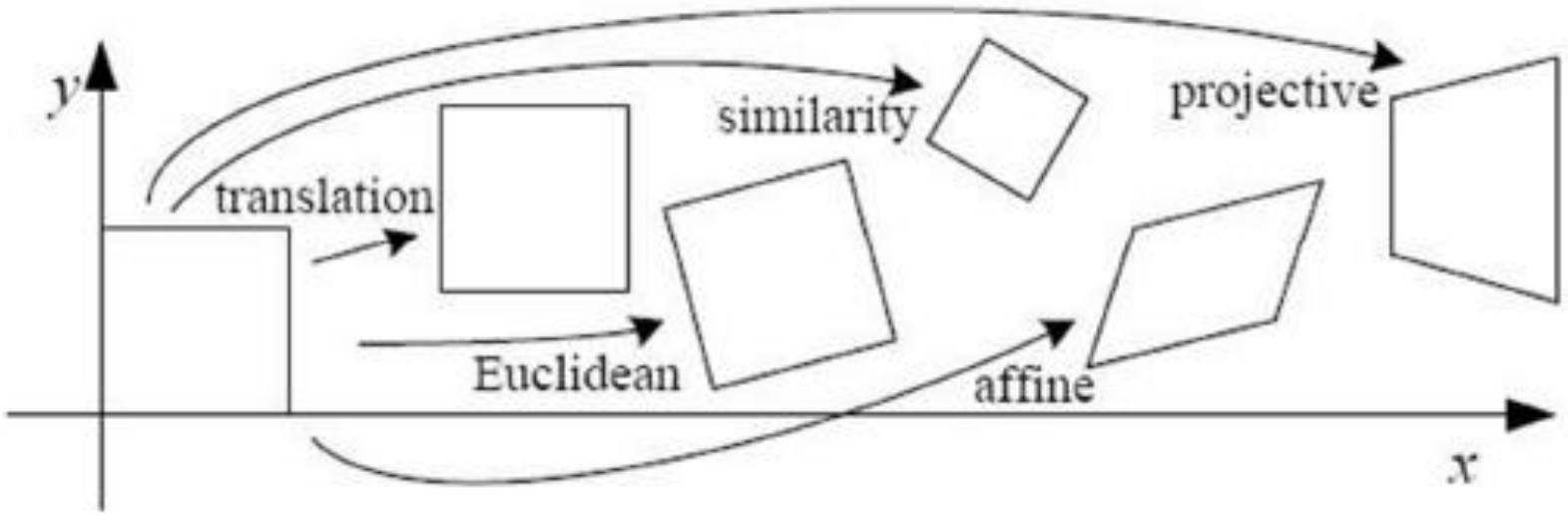
Invariance: Geometric Transformations



Invariance: Geometric Transformations



Levels of Geometric Invariance



Invariance: Photometric Transformatio

- Often modeled as a linear transformation:
  - Scaling + Offset

Requirements

- **Region extraction needs to be repeatable and accurate**
  - **Invariant to translation, rotation, scale changes**
  - **Robust to affine transformations**
  - **Robust to lighting variations, noise, blur, quantization**

- **Locality: Features are local, therefore robust to occlusion and clutter.**
- **Quantity: We need a sufficient number of regions to cover the object.**
- **Distinctiveness: The regions should contain "interesting" structure.**
- **Efficiency: Close to real-time performance.**

More motivation…

Feature points are used for:

- **Image alignment (e.g., mosaics)**
- **3D reconstruction**
- **Motion tracking**
- **Object recognition**
- **Indexing and database retrieval**
- **Robot navigation**
- **… other**

Local Image Features

- **Properties of detectors**
  - **Edge detectors**
  - **Corners**
  - **Blobs**
  - **Scale invariant detection**
- **Properties of descriptors**
  - **HOG**
  - **SIFT**

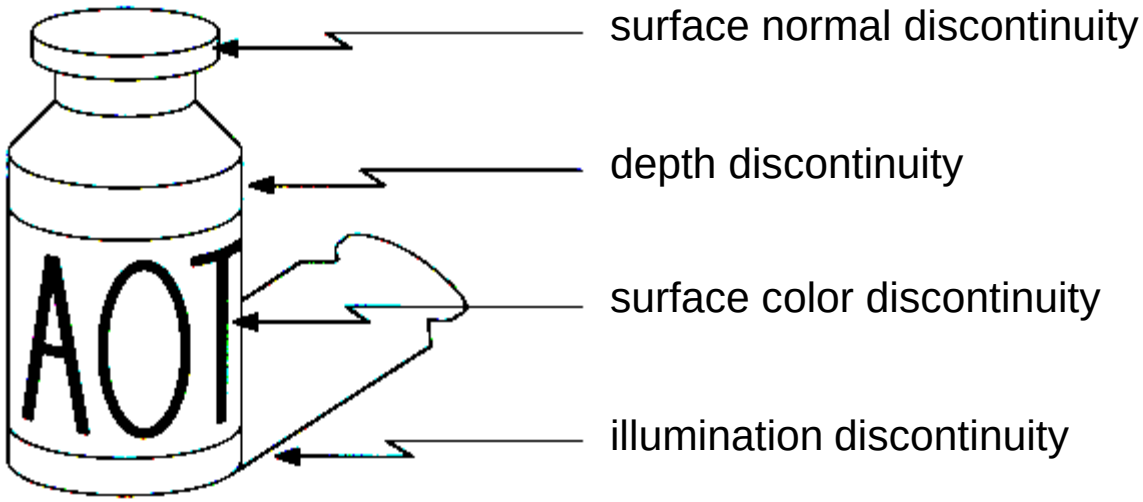
Edge detection

- **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)**

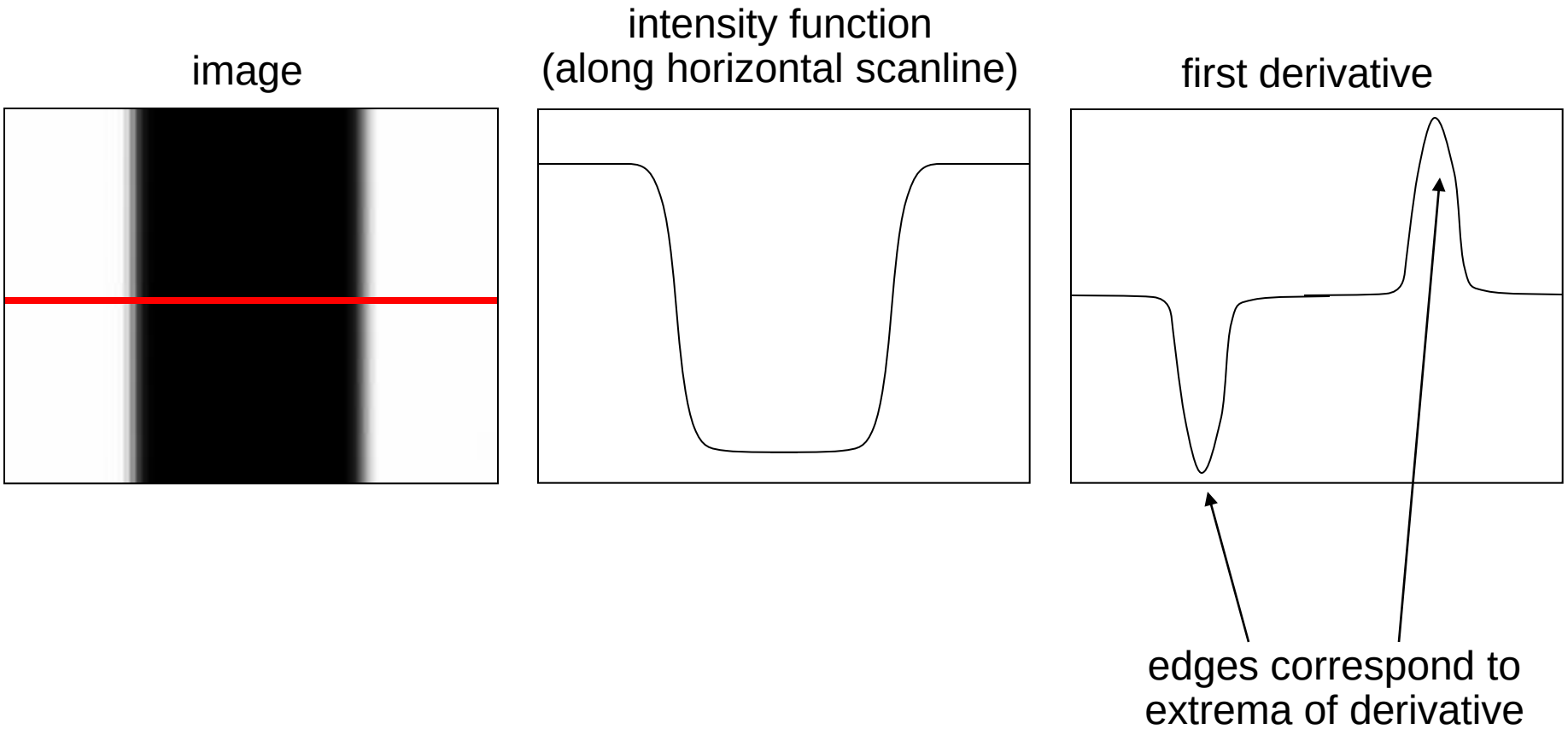
Origin of edges

- **Edges are caused by a variety of factors:**



Edge detection

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



Derivatives with convolution

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

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

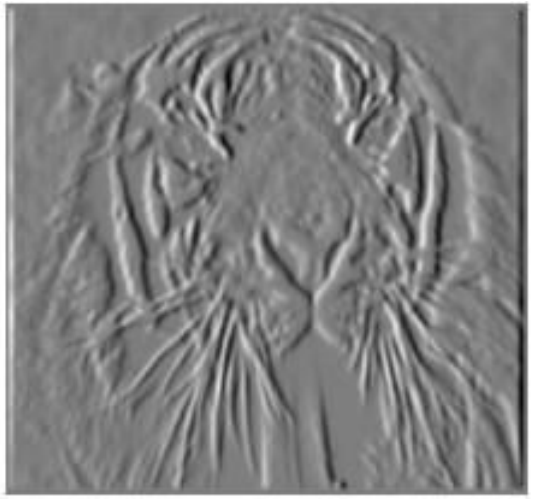
For discrete data, we can approximate using finite differences:

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

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

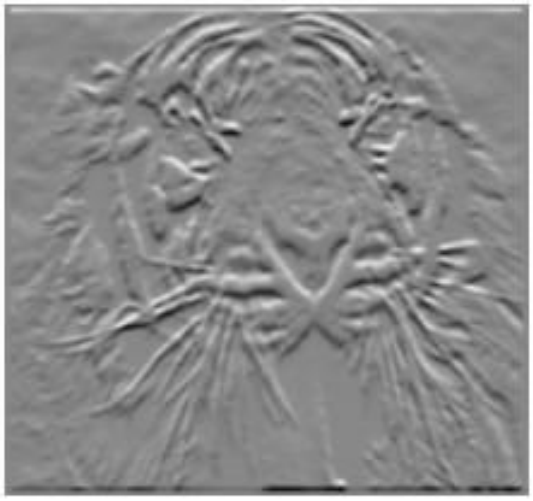
Partial derivatives of an image

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



-1	1

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



-1
1

or

1
-1

Which shows changes with respect to x?

Finite difference filters

-

Prewitt: $M_x =$

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

; $M_y =$

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

Sobel: $M_x =$

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

; $M_y =$

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

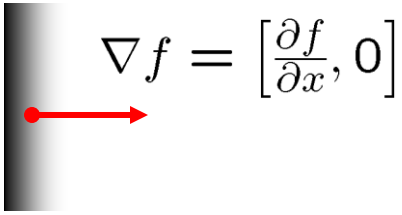
Roberts: $M_x =$

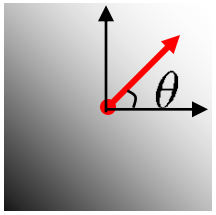
0	1
-1	0

; $M_y =$

1	0
0	-1

Image gradient

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



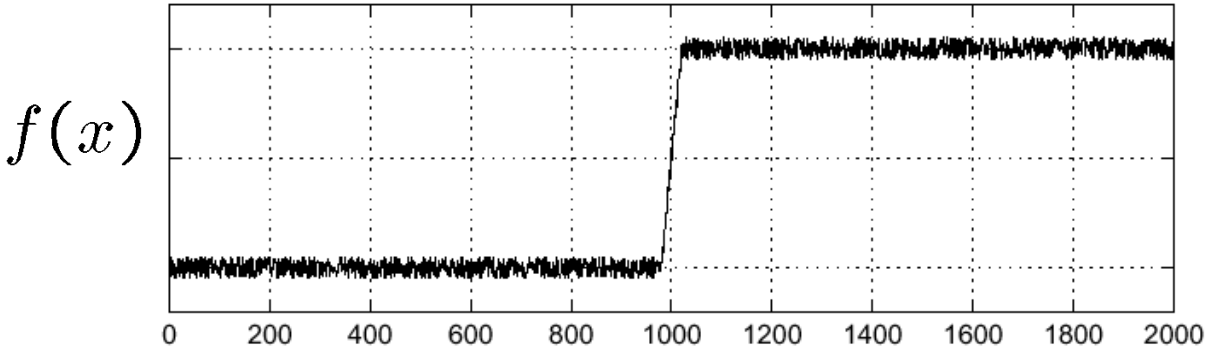
- The gradient points in the direction of most rapid increase in intensity
  - How does this direction relate to the direction of the edge?
- The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The edge strength is given by the gradient magnitude

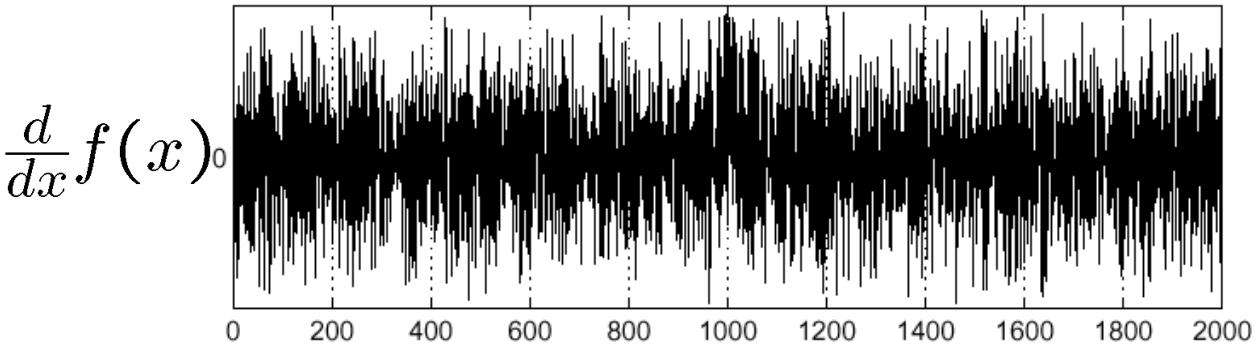
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Effects of noise

- **Consider a single row or column of the image**

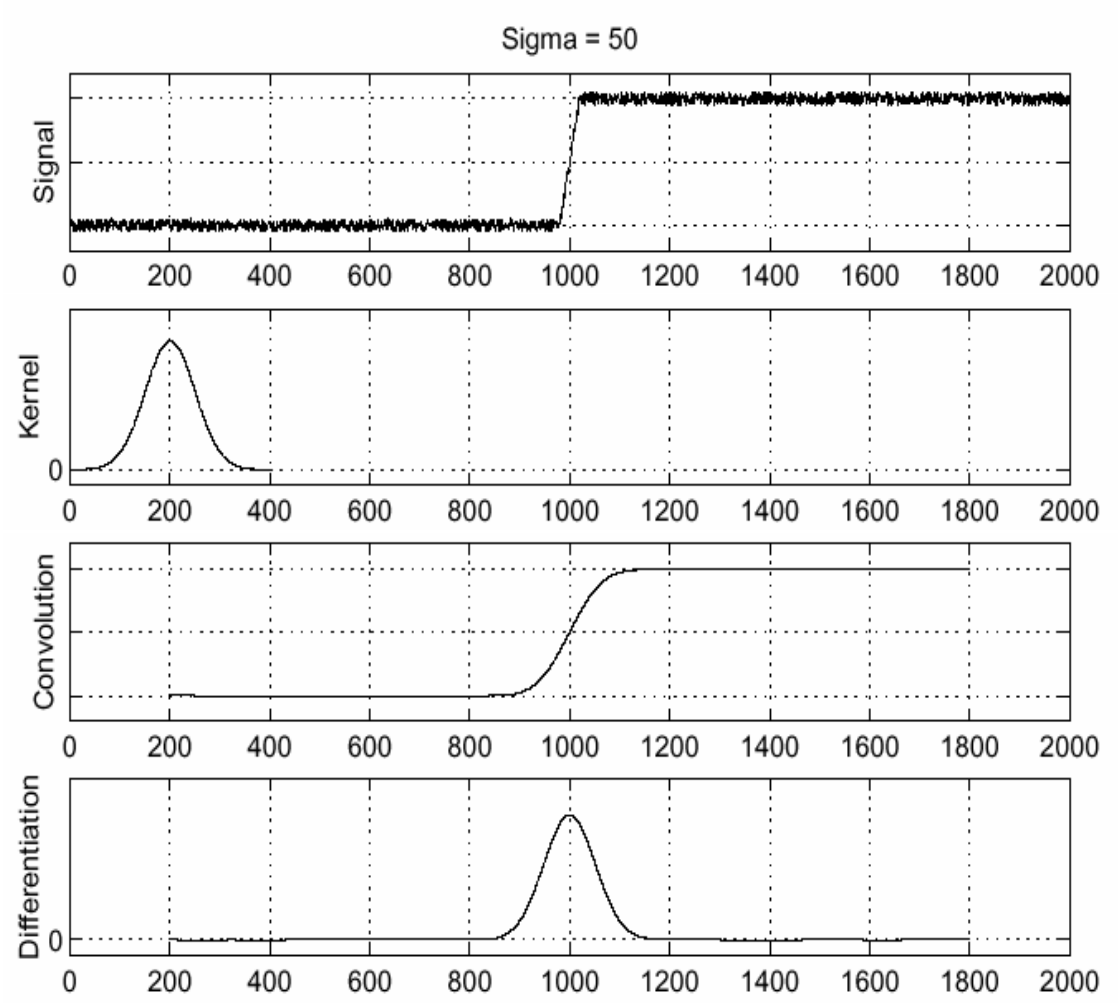




- Where is the edge?

Source: S. Seitz

Solution: smooth first



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

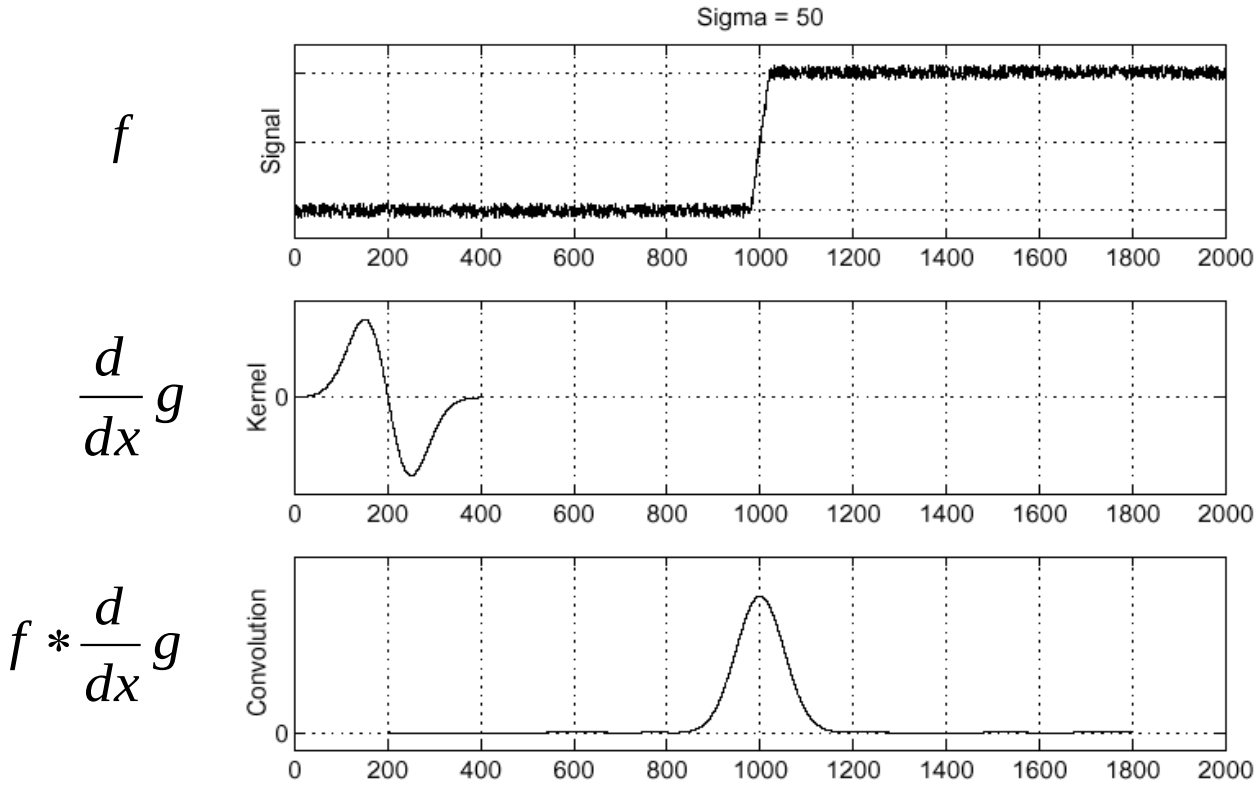
Source: S. Seitz

Derivative theorem of convolution

- **Differentiation is convolution, and convolution is associative:**

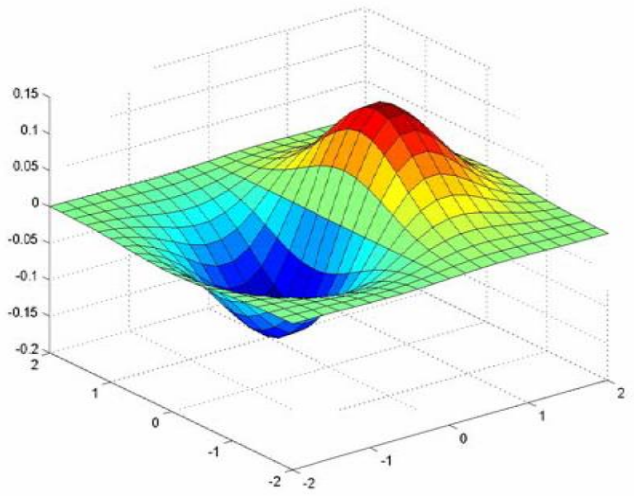
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- **This saves us one operation:**

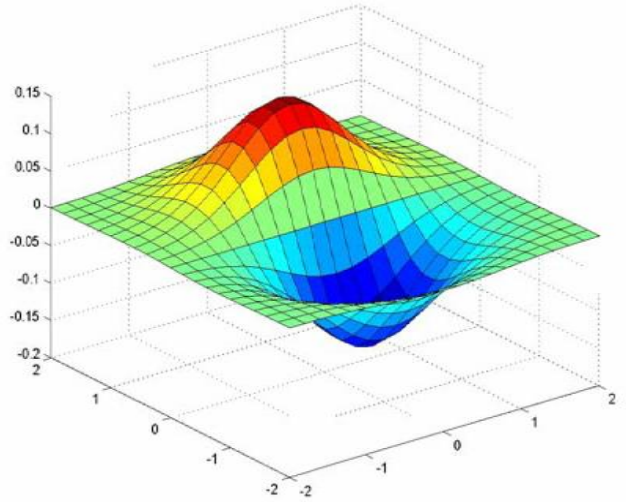


Source: S. Seitz

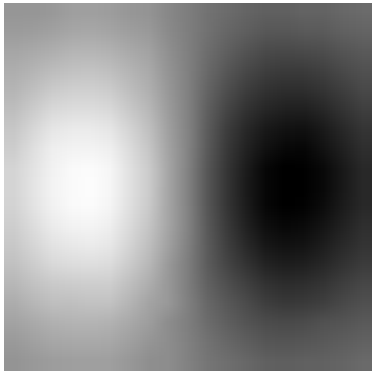
Derivative of Gaussian filters

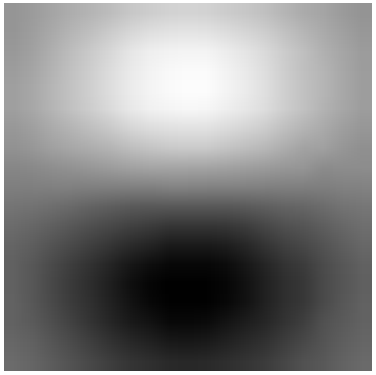


x-direction



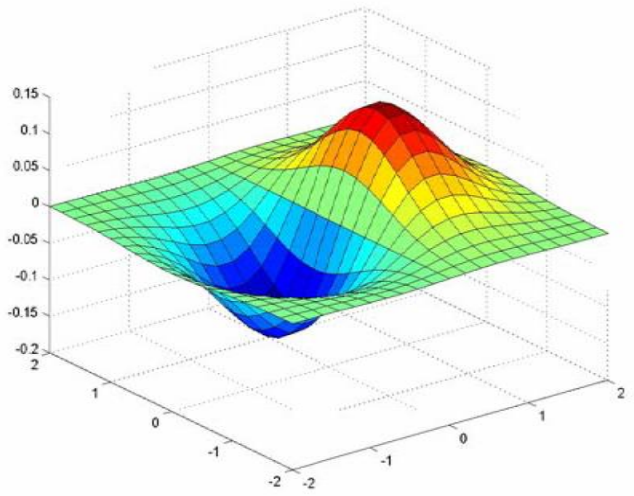
y-direction



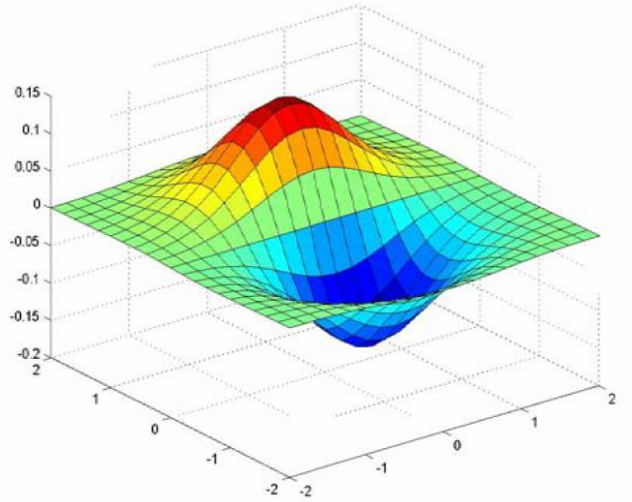


- **Which one finds horizontal/vertical edges?**

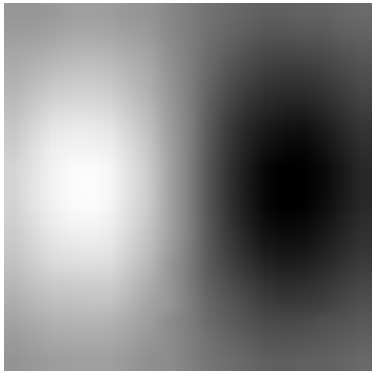
Derivative of Gaussian filters

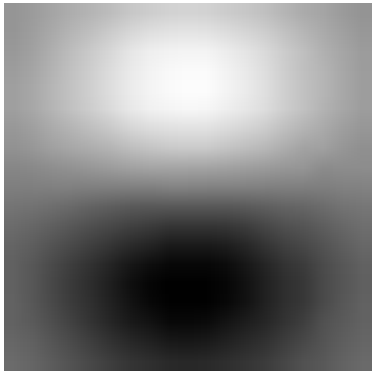


x-direction



y-direction





- **Are these filters separable?**

Recall: Separability of the Gaussian filter

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

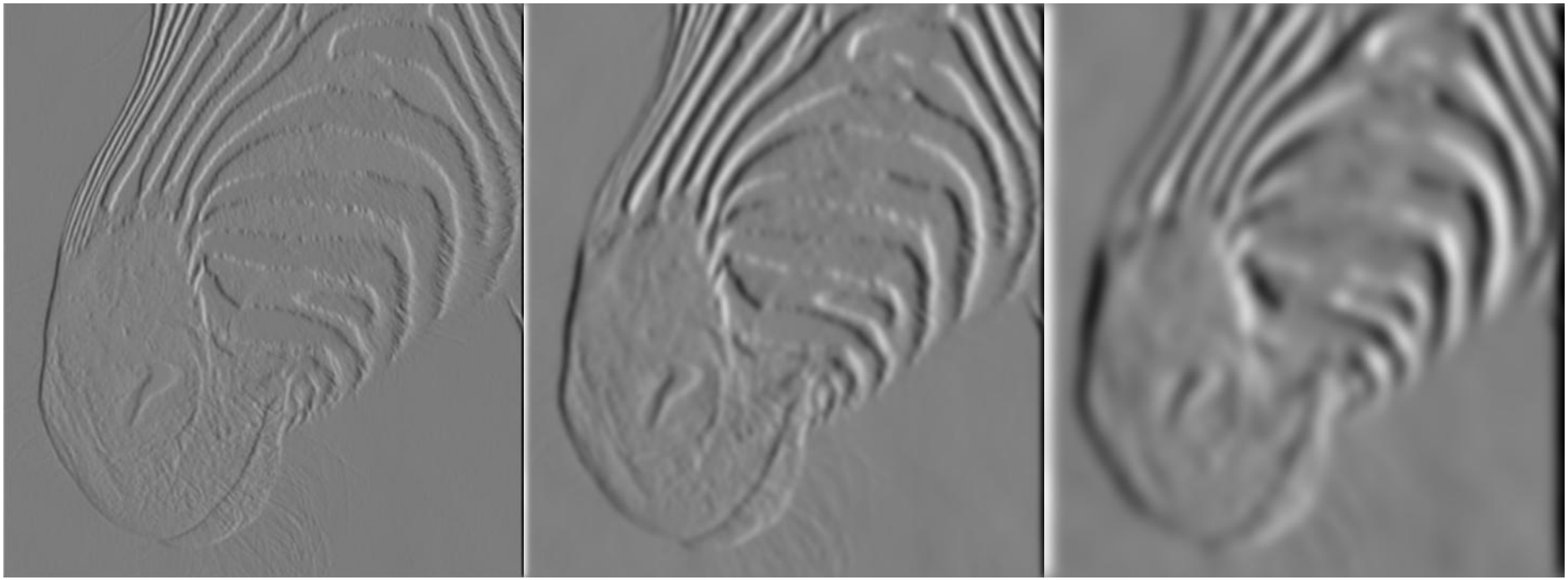
$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

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

Source: D. Lowe

Scale of Gaussian derivative filter



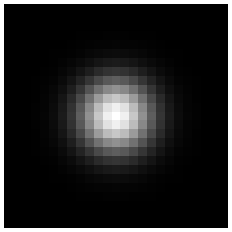
1 pixel 3 pixels 7 pixels

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

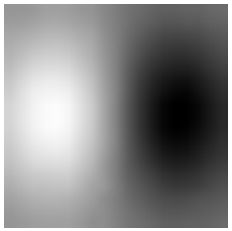
Source: D. Forsyth

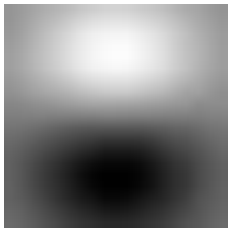
Review: Smoothing vs. derivative filters

- 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**

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

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

The Canny edge detector

original image

Slide credit: Steve Seitz

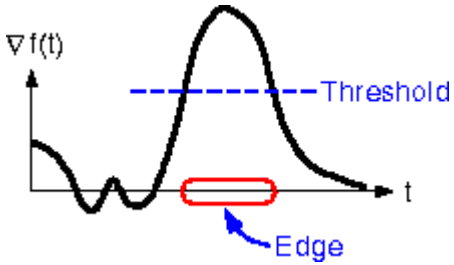
The Canny edge detector

norm of the gradient

The Canny edge detector

thresholding

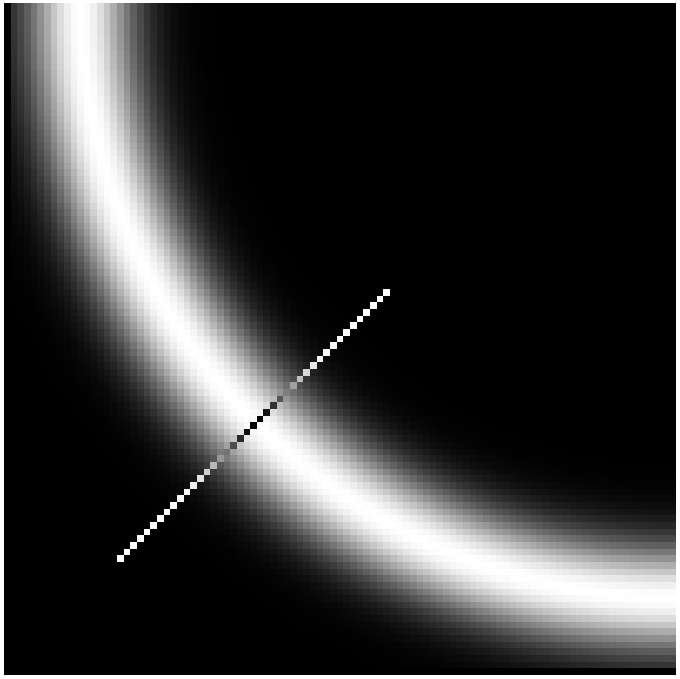
The Canny edge detector

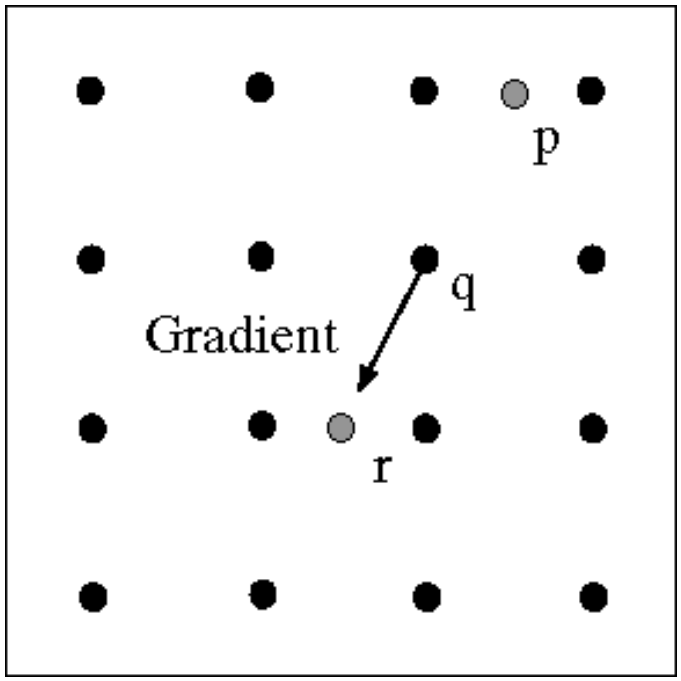


How to turn these thick regions of the gradient into curves?

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**

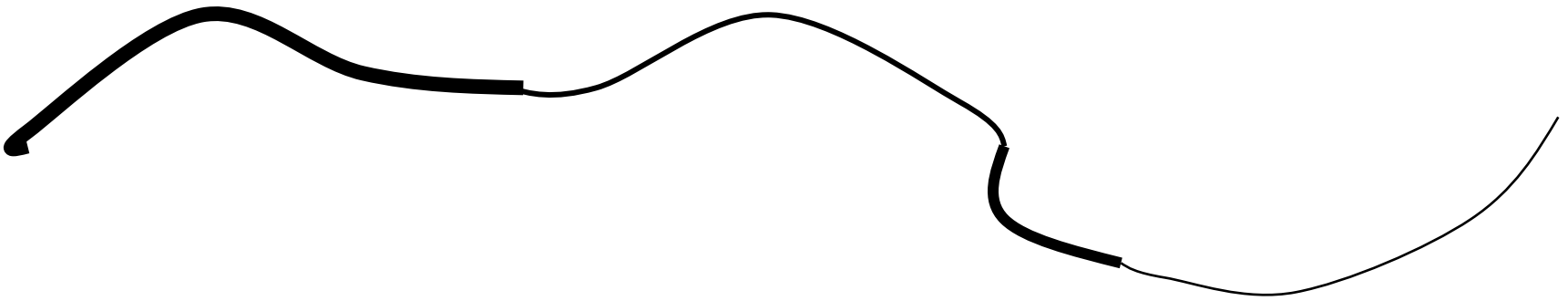
The Canny edge detector

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

thinning
(non-maximum suppression)

Hysteresis thresholding

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



Source: Steve Seitz

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)

hysteresis threshold

Source: L. Fei-Fei

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');`

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

Edges - crucial points

- Edges are caused by
  - changes in orientation
  - changes in albedo
  - object boundaries
  - other effects (specularities, lighting, etc.)

- No edge detector can discriminate between these cases
  - so edge maps can be hard to use

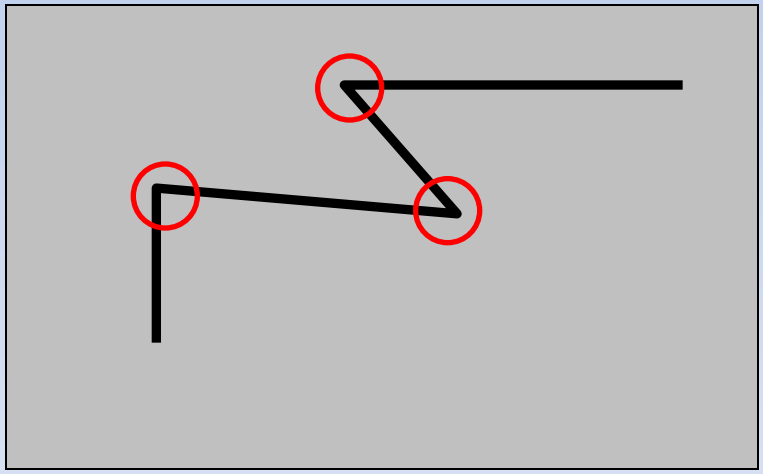
- There are numerous good edge-detectors available
  - Most are based on maximum of gradient magnitude

Local Image Features

- **Properties of detectors**
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- **Properties of descriptors**
  - HOG
  - SIFT

An introductory example:

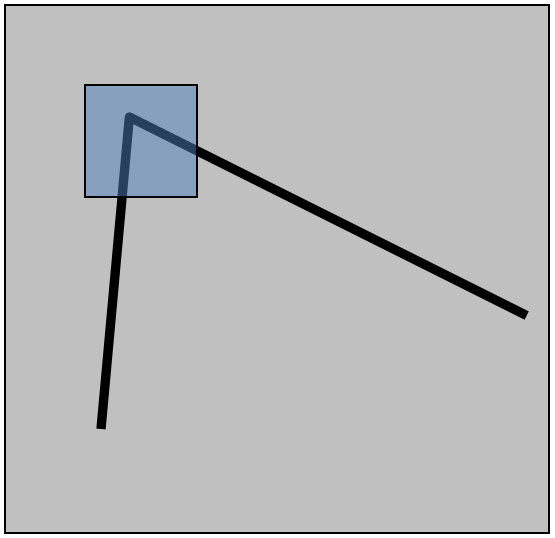
Harris corner detector



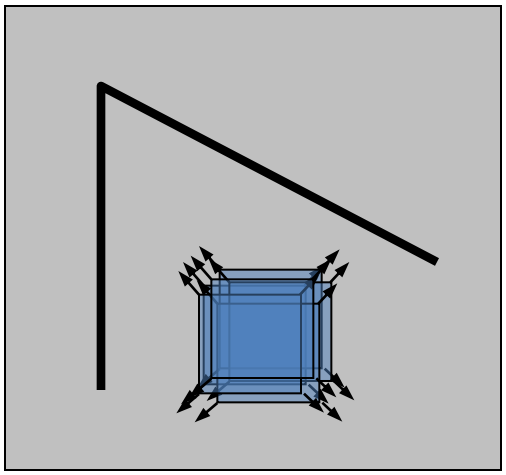
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

The Basic Idea

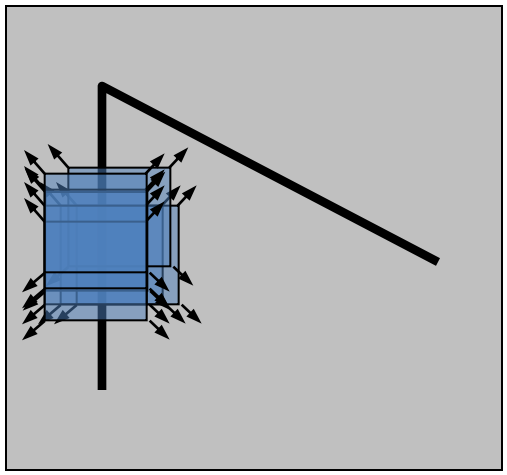
- **We should easily localize the point by looking through a small window**
- **Shifting a window in *any direction* should give *a large change* in intensity**



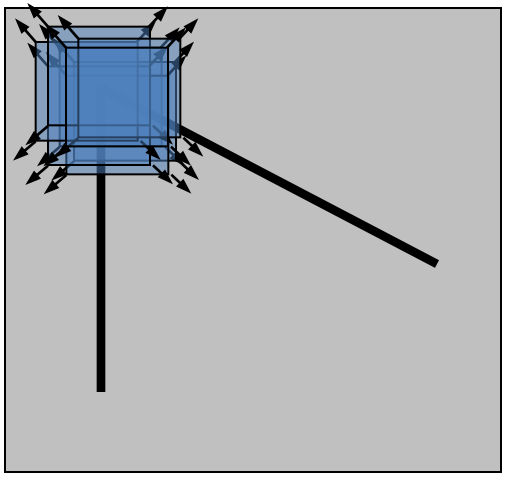
Harris Detector: Basic Idea



“flat” region:
no change as shift window in all directions



“edge”:
no change as shift window along the edge direction



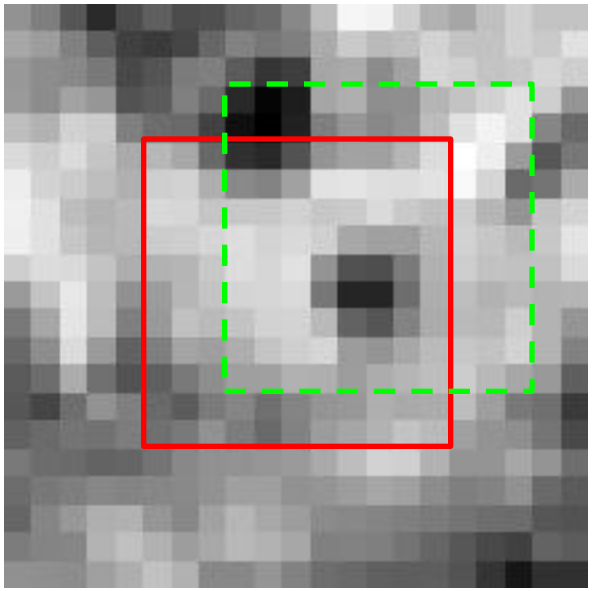
“corner”:
significant change as shift window in all directions

Corner Detection: Mathematics

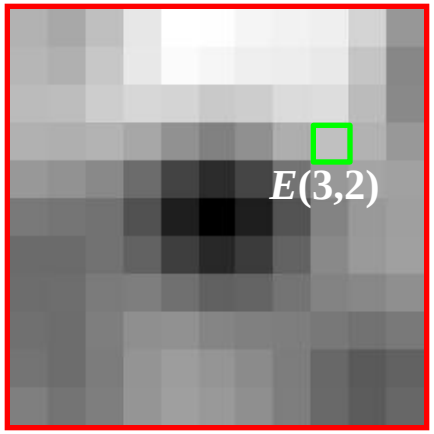
Change in appearance of window *W* for the shift [*u,v*]:

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

$I(x, y)$



$E(u, v)$

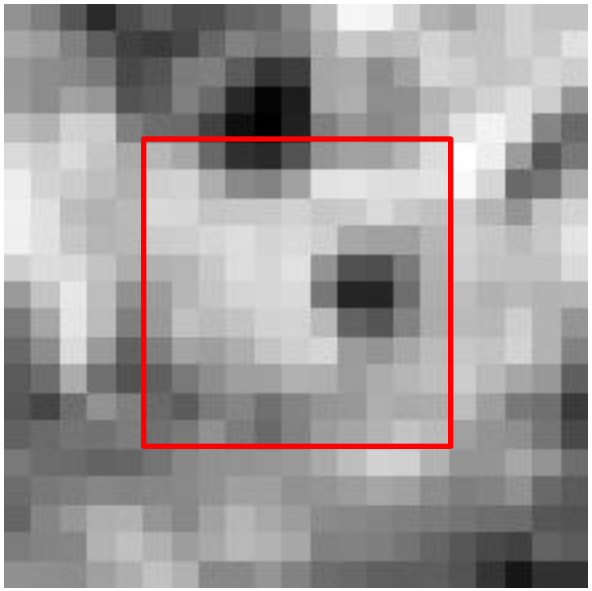


Corner Detection: Mathematics

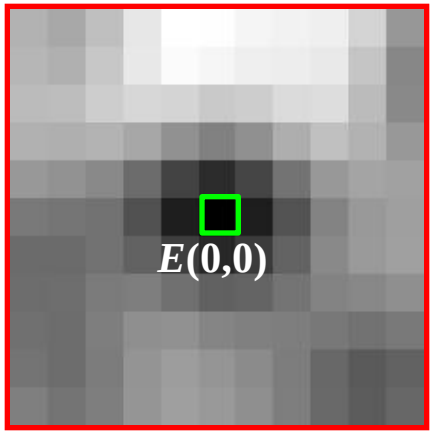
Change in appearance of window *W* for the shift [*u,v*]:

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

$I(x, y)$



$E(u, v)$



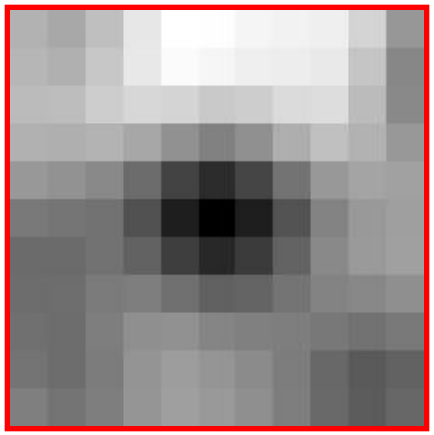
Corner Detection: Mathematics

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*]:**

$$I(x+u, y+v) = I(x,y) + I_x u + I_y v + \text{higher-order terms}$$

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

$$= I(x,y) + \begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

- **Let's plug this into**

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

Corner Detection: Mathematics

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

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

$$= \sum_{(x,y)\in W} \left(\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \right)^2$$

$$= \sum_{(x,y)\in W} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Corner Detection: Mathematics

The quadratic approximation simplifies to

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

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

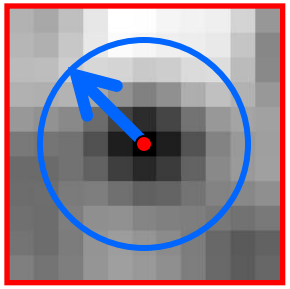
$$M = \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Interpreting the second moment matrix

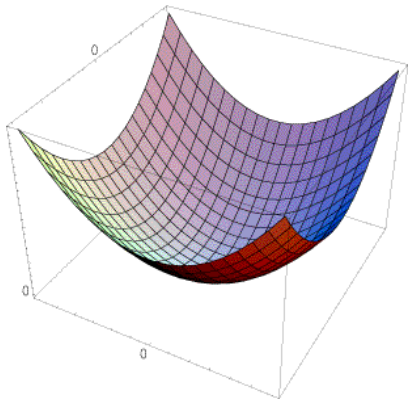
- The surface $E(u,v)$ is locally approximated by a quadratic form. Let's try to understand its shape.
  - Specifically, in which directions does it have the smallest/greatest change?

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

$E(u, v)$



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



Interpreting the second moment matrix

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

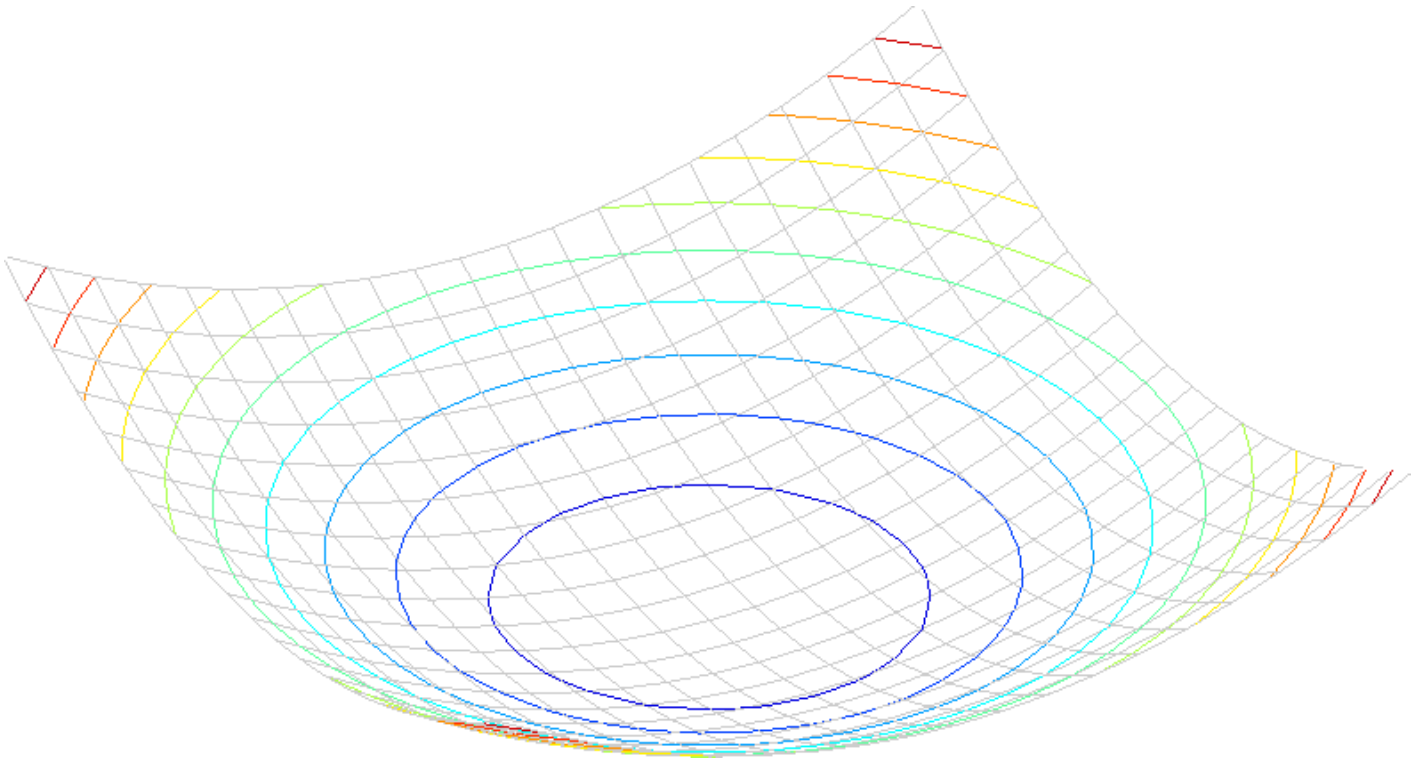
$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

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

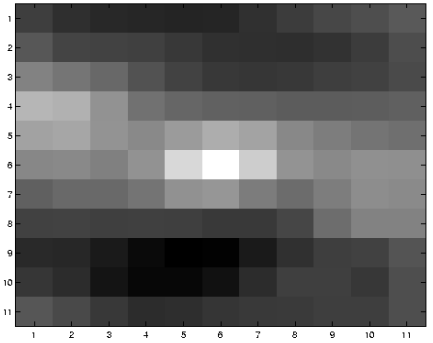
Interpreting the second moment matrix

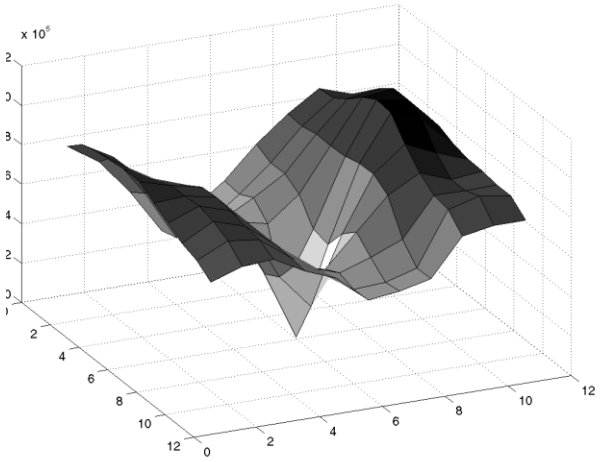
Consider a horizontal "slice" of $E(u, v)$: $\begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$

This is the equation of an ellipse.



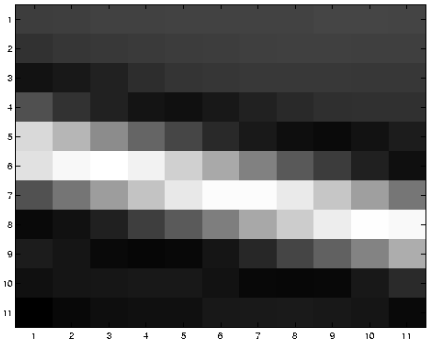
Selecting Good Features

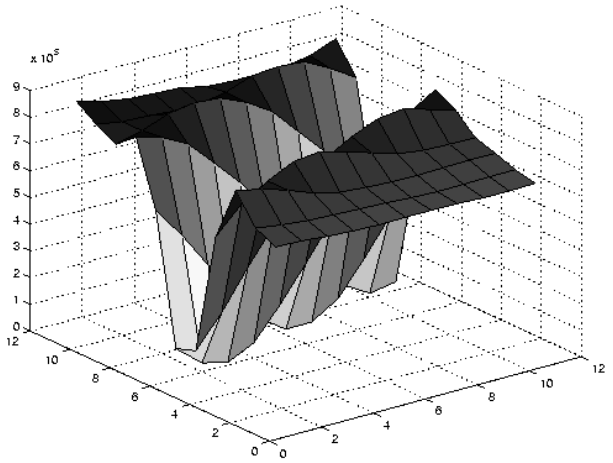




λ_1 and λ_2 are large

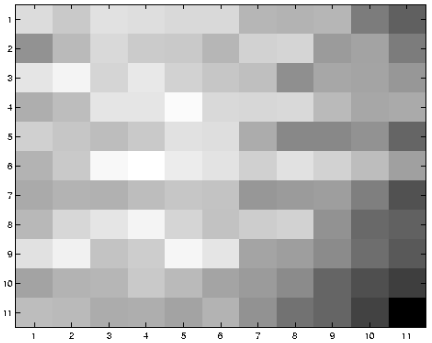
Selecting Good Features

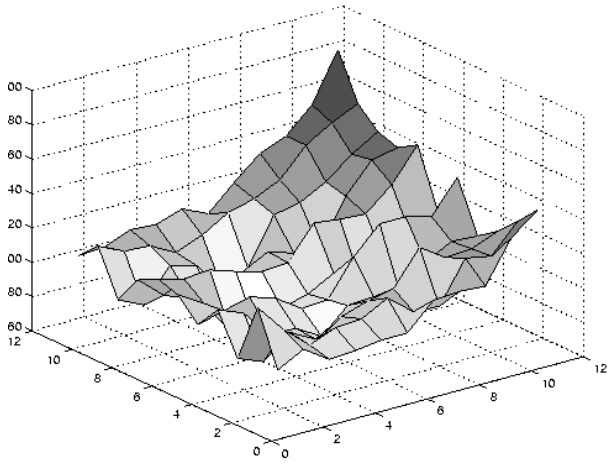




large λ_1, small λ_2

Selecting Good Features





small λ_1, small λ_2

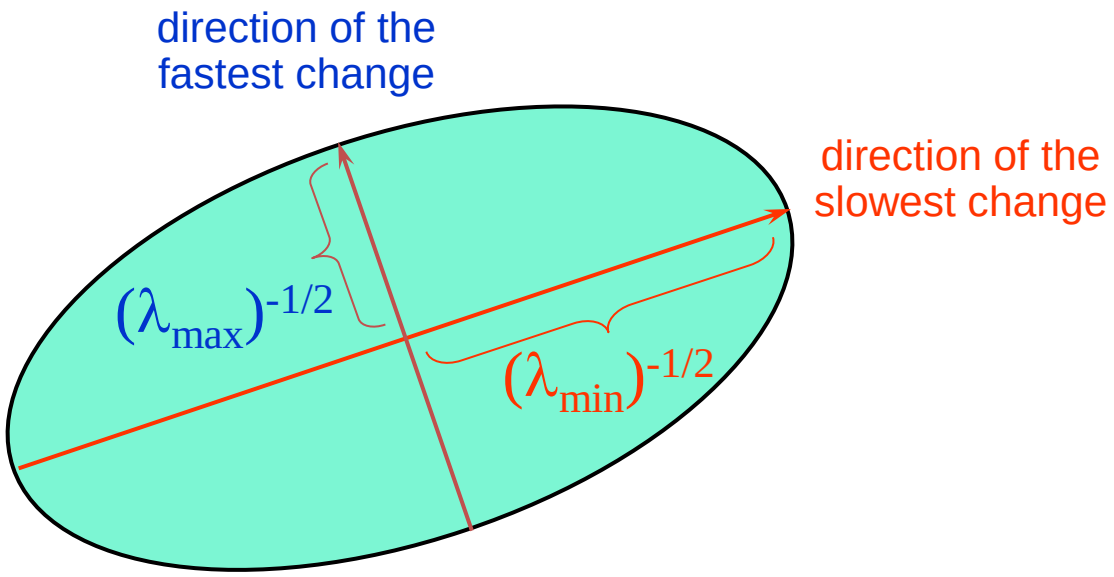
Interpreting the second moment matrix

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

This is the equation of an ellipse.

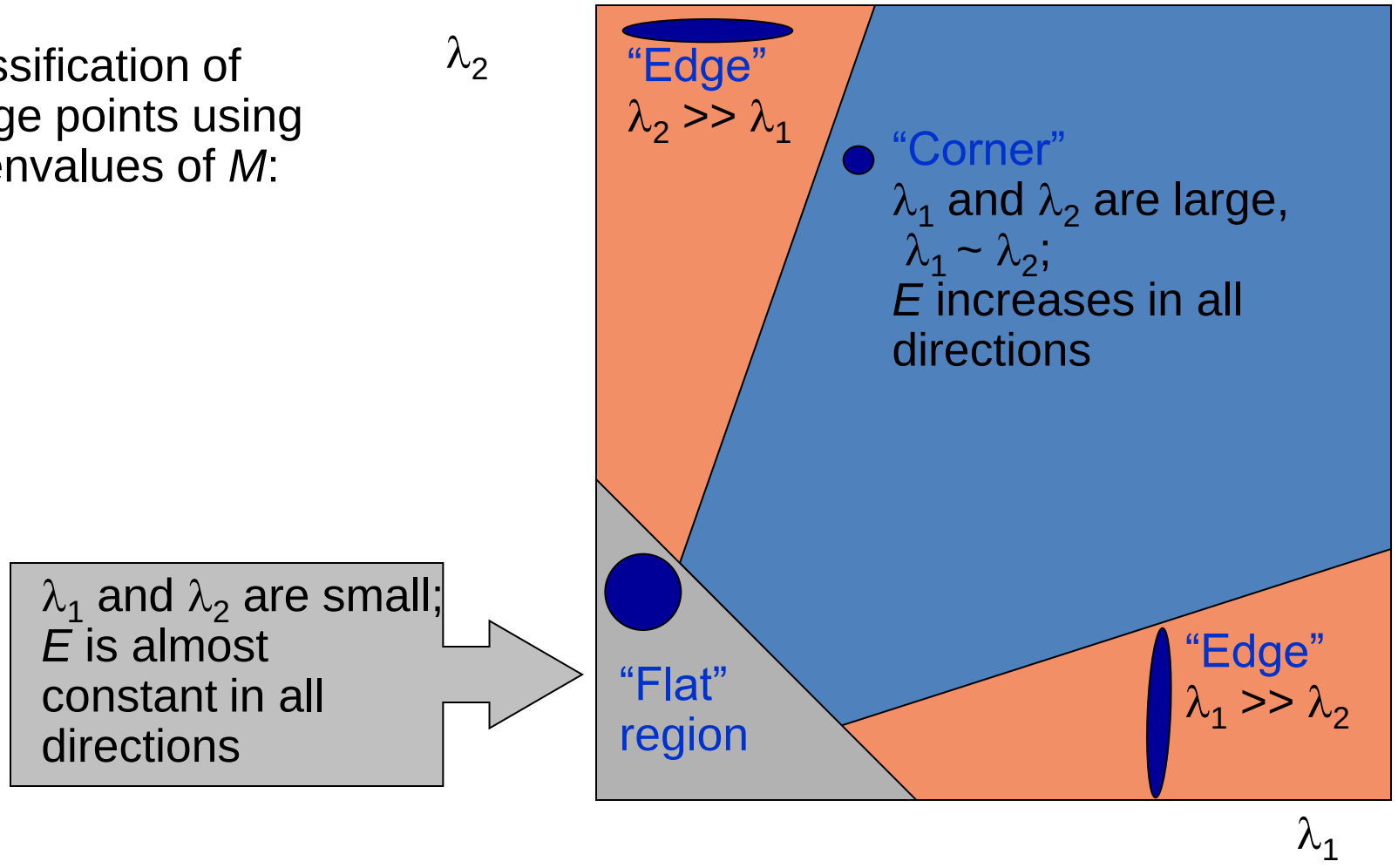
Diagonalization of M: $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$

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

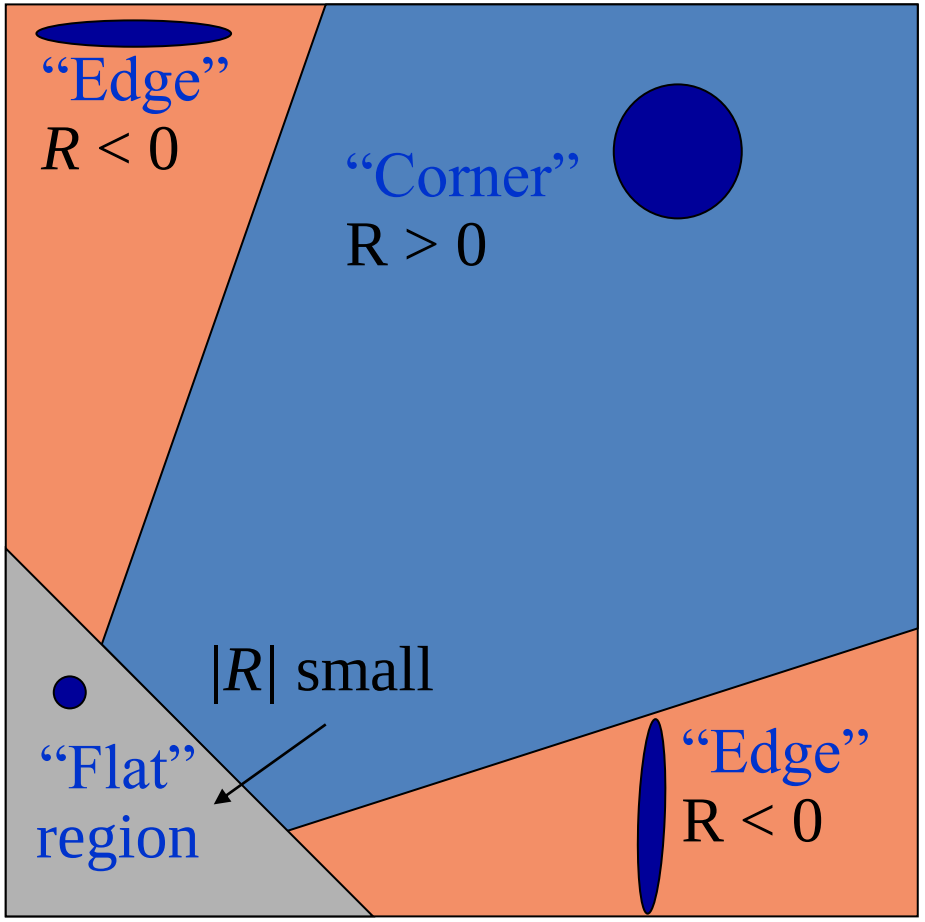


Harris Detector: Mathematics

Classification of image points using eigenvalues of *M*:



Corner response function



α: constant (0.04 to 0.06)

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

Harris Detector: Mathematics

Window-averaged change of intensity induced by shifting the image data by [*u,v*]:

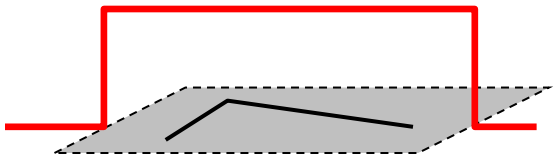
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

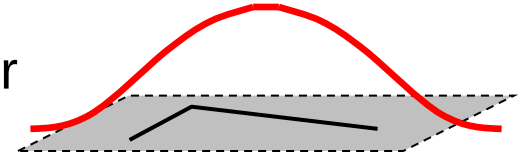
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Taylor series approx to shifted image

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= \sum_{x,y} w(x,y) \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Harris Detector: Mathematics

Expanding I(x,y) in a Taylor series expansion, we have, for small shifts [*u,v*], a *bilinear* approximation:

$$E(u,v) \cong \begin{bmatrix} u,v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

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

M is also called “structure tensor”

The Harris corner detector

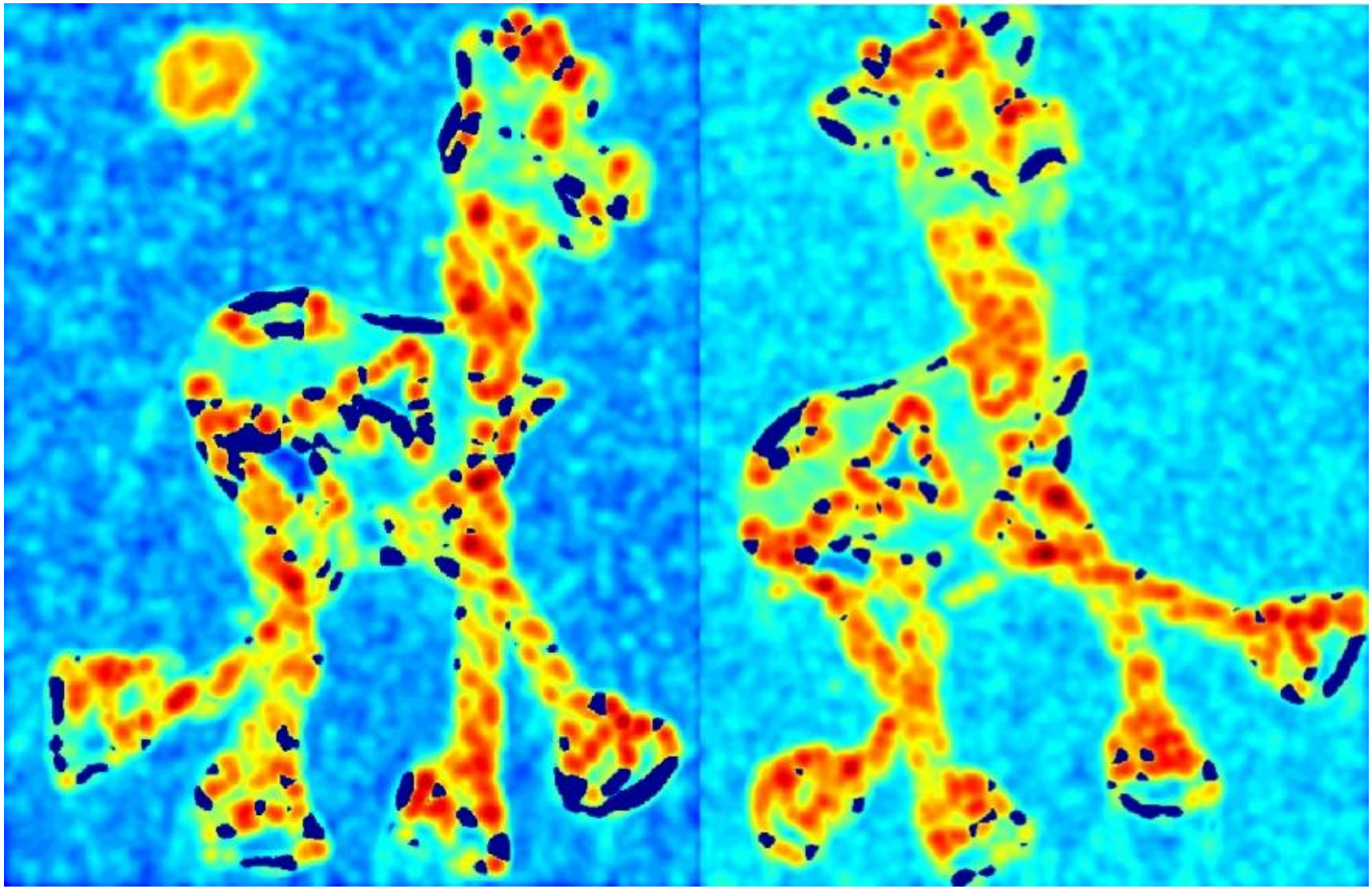
1. **Compute Gaussian derivatives at each pixel**
2. **Compute second moment matrix *M* in a Gaussian window around each pixel**
3. **Compute corner response function *R***
4. **Threshold *R***
5. **Find local maxima of response function (nonmaximum suppression)**

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

Harris Detector: Workflow

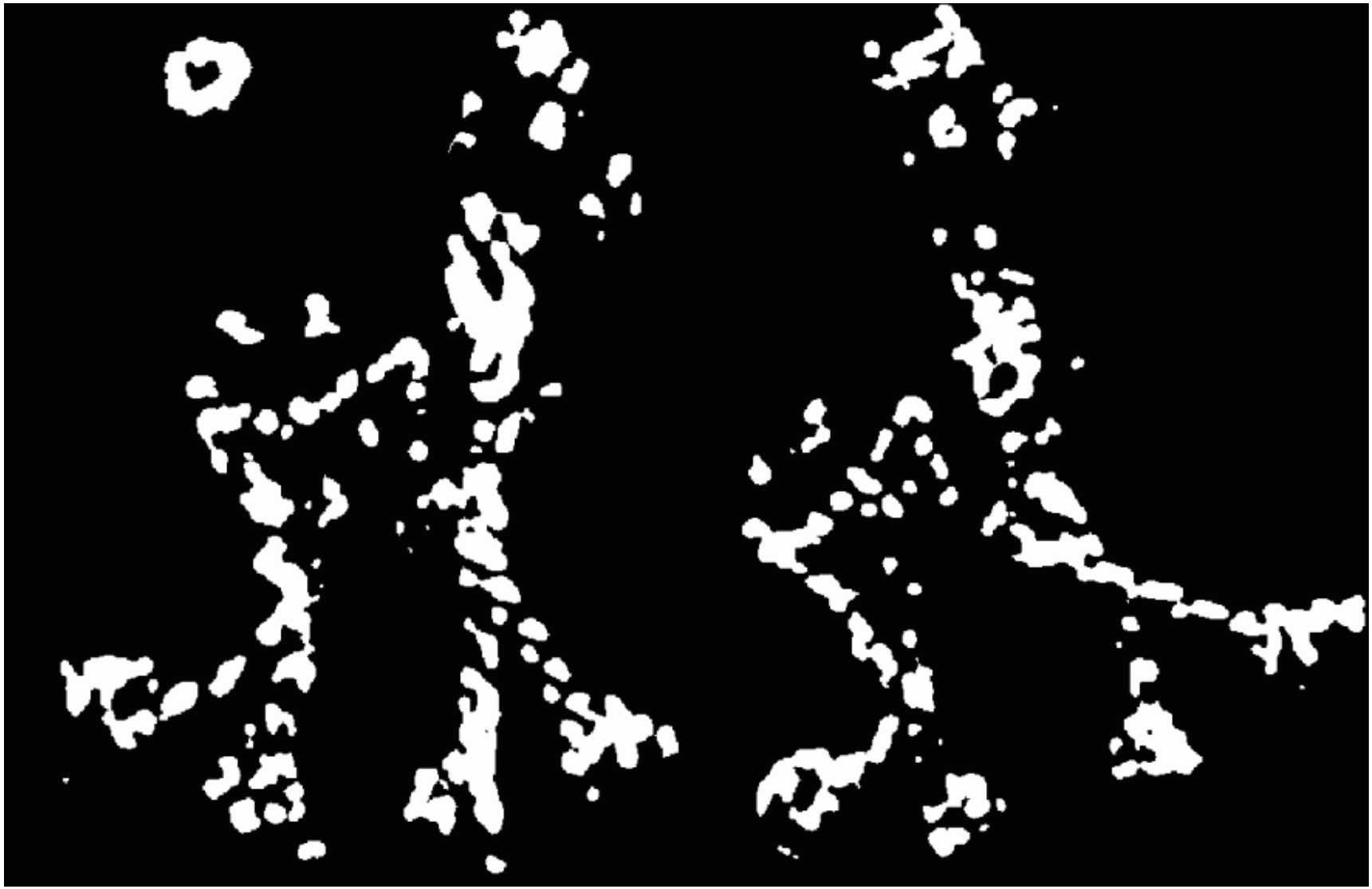
Harris Detector: Workflow

Compute corner response *R*



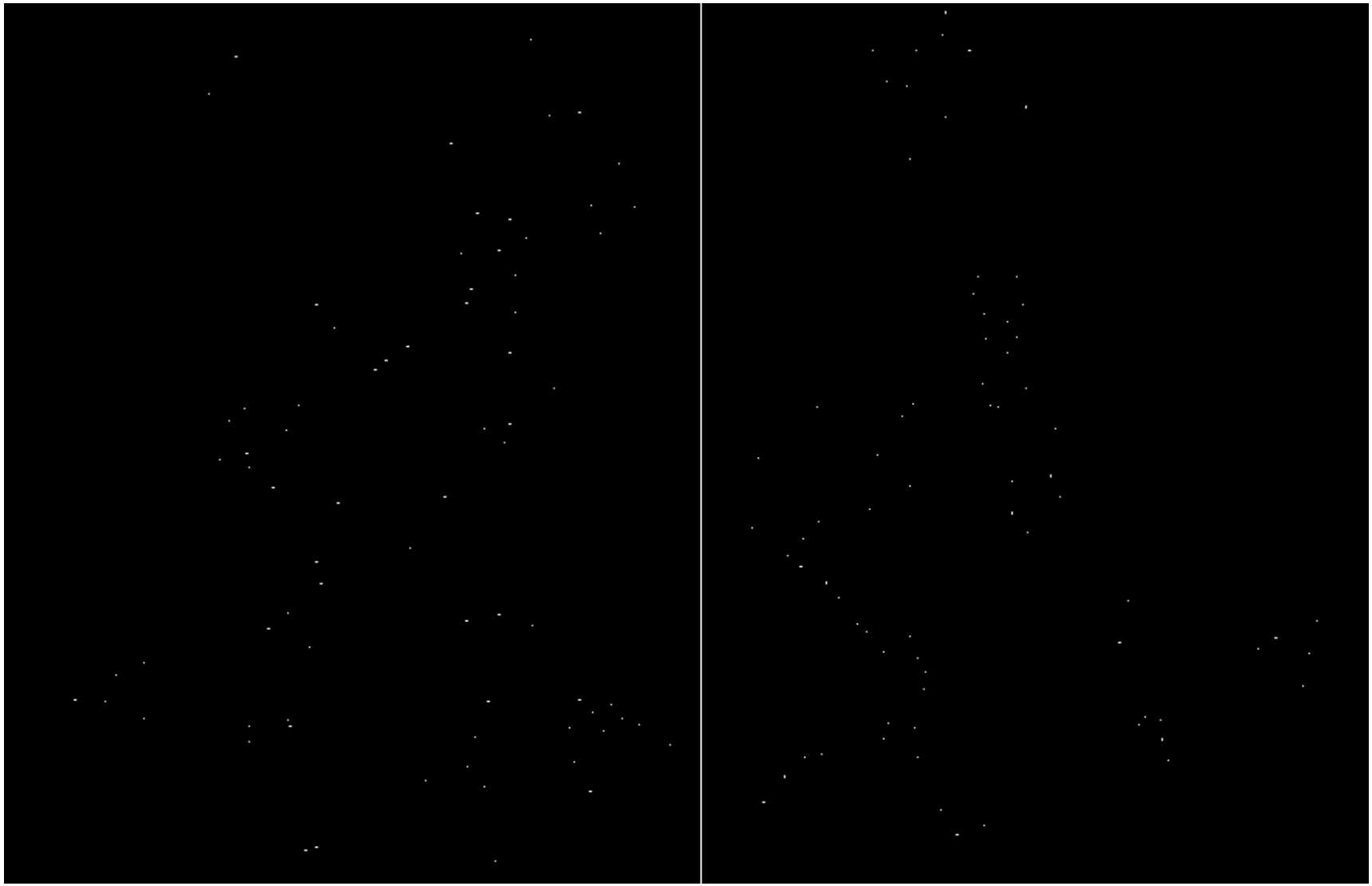
Harris Detector: Workflow

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



Harris Detector: Workflow

Take only the points of local maxima of *R*



Harris Detector: Workflow

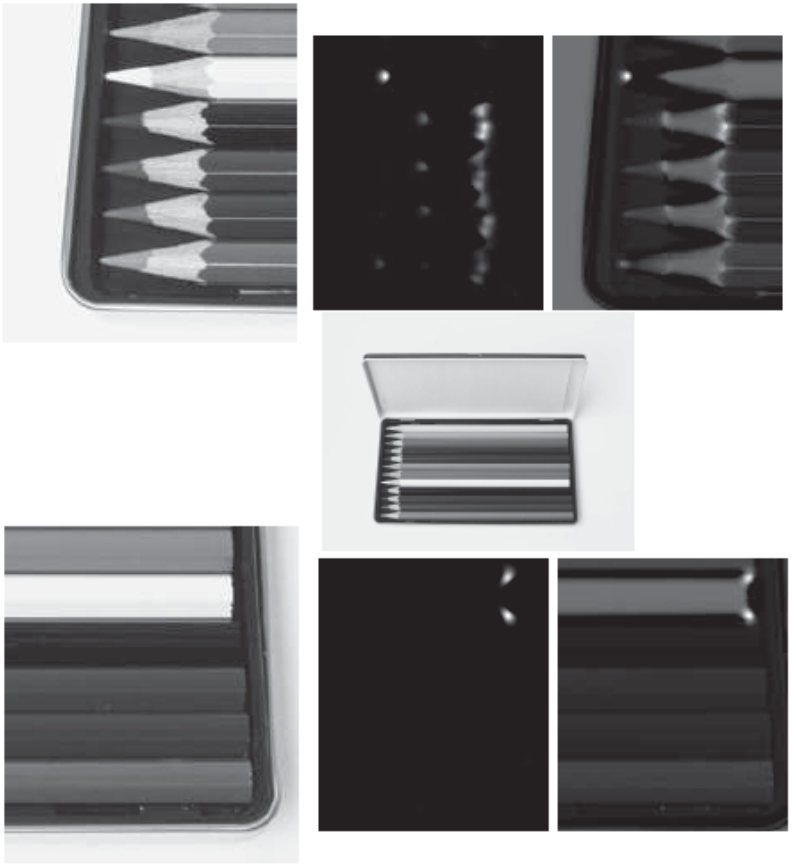


FIGURE 5.10: The response of the Harris corner detector visualized for two detail regions of an image of a box of colored pencils (**center**). **Top left**, a detail from the pencil points; **top center**, the response of the Harris corner detector, where more positive values are lighter. The **top right** shows these overlaid on the original image. To overlay this map, we added the images, so that areas where the overlap is notably dark come from places where the Harris statistic is negative (which means that one eigenvalue of $\mathcal{H}$ is large, the other small). Note that the detector is affected by contrast, so that, for example, the point of the mid-gray pencil at the top of this figure generates a very strong corner response, but the points of the darker pencils do not, because they have little contrast with the tray. For the darker pencils, the strong, contrasty corners occur where the lead of the pencil meets the wood. The **bottom** sequence shows corners for a detail of pencil ends. Notice that responses are quite local, and there are a relatively small number of very strong corners. *Steve Gorton © Dorling Kindersley, used with permission.*

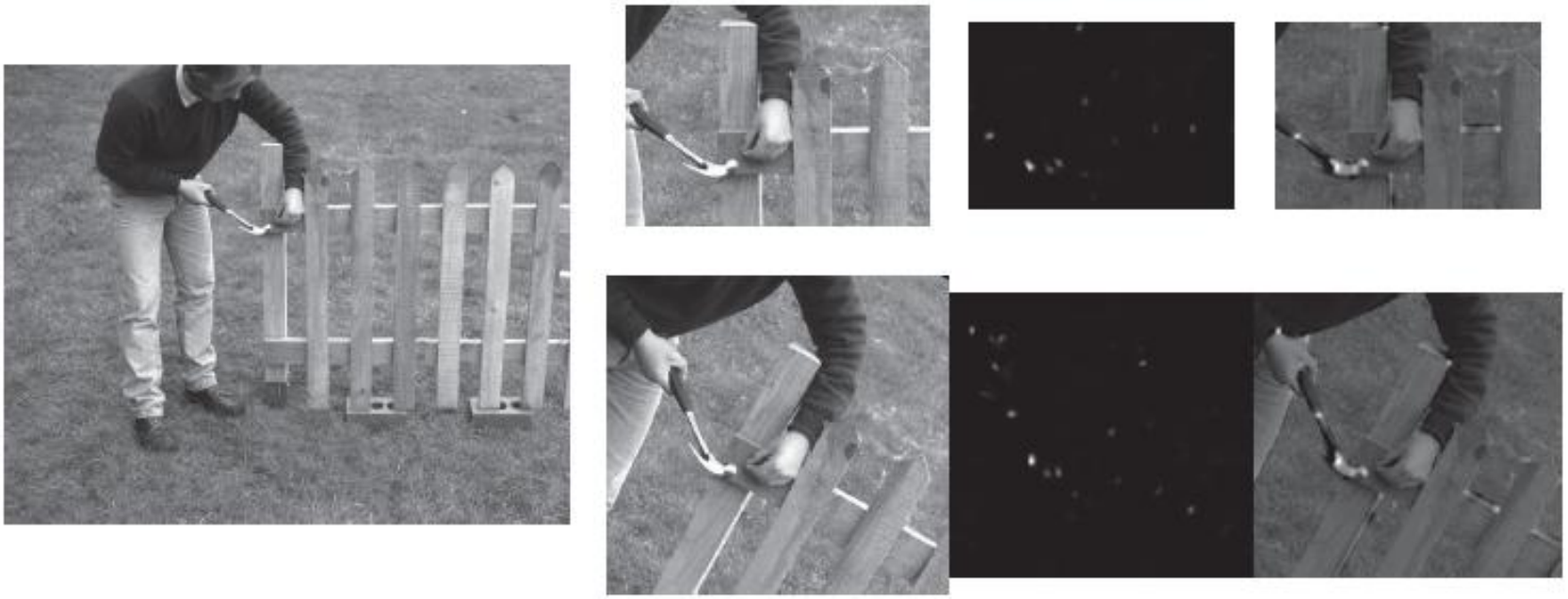


FIGURE 5.11: The response of the Harris corner detector is unaffected by rotation and translation. The **top row** shows the response of the detector on a detail of the image on the **far left**. The **bottom row** shows the response of the detector on a corresponding detail from a rotated version of the image. For each row, we show the detail window (**left**); the response of the Harris corner detector, where more positive values are lighter (**center**); and the responses overlaid on the image (**right**). Notice that responses are quite local, and there are a relatively small number of very strong corners. To overlay this map, we added the images, so that areas where the overlap is notably dark come from places where the Harris statistic is negative (which means that one eigenvalue of $\mathcal{H}$ is large, the other small). The arm and hammer in the top row match those in the bottom row; notice how well the maps of Harris corner detector responses match, too.

Harris Detector: Summary

- **Average intensity change in direction [*u,v*] can be expressed as a bilinear form:**

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

- **Describe a point in terms of eigenvalues of *M*: *measure of corner response***

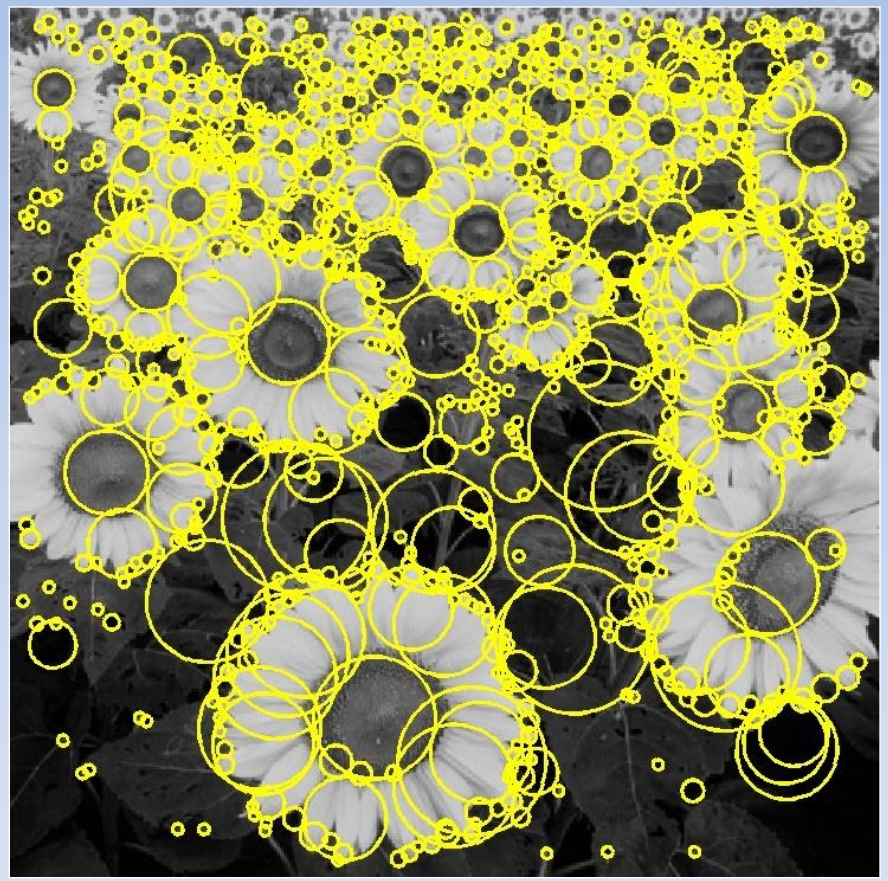
$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

- **A good (corner) point should have a *large intensity change* in *all directions*, i.e. *R* should be large positive**
- **The detector is unaffected by translation and rotation.**

Local Image Features

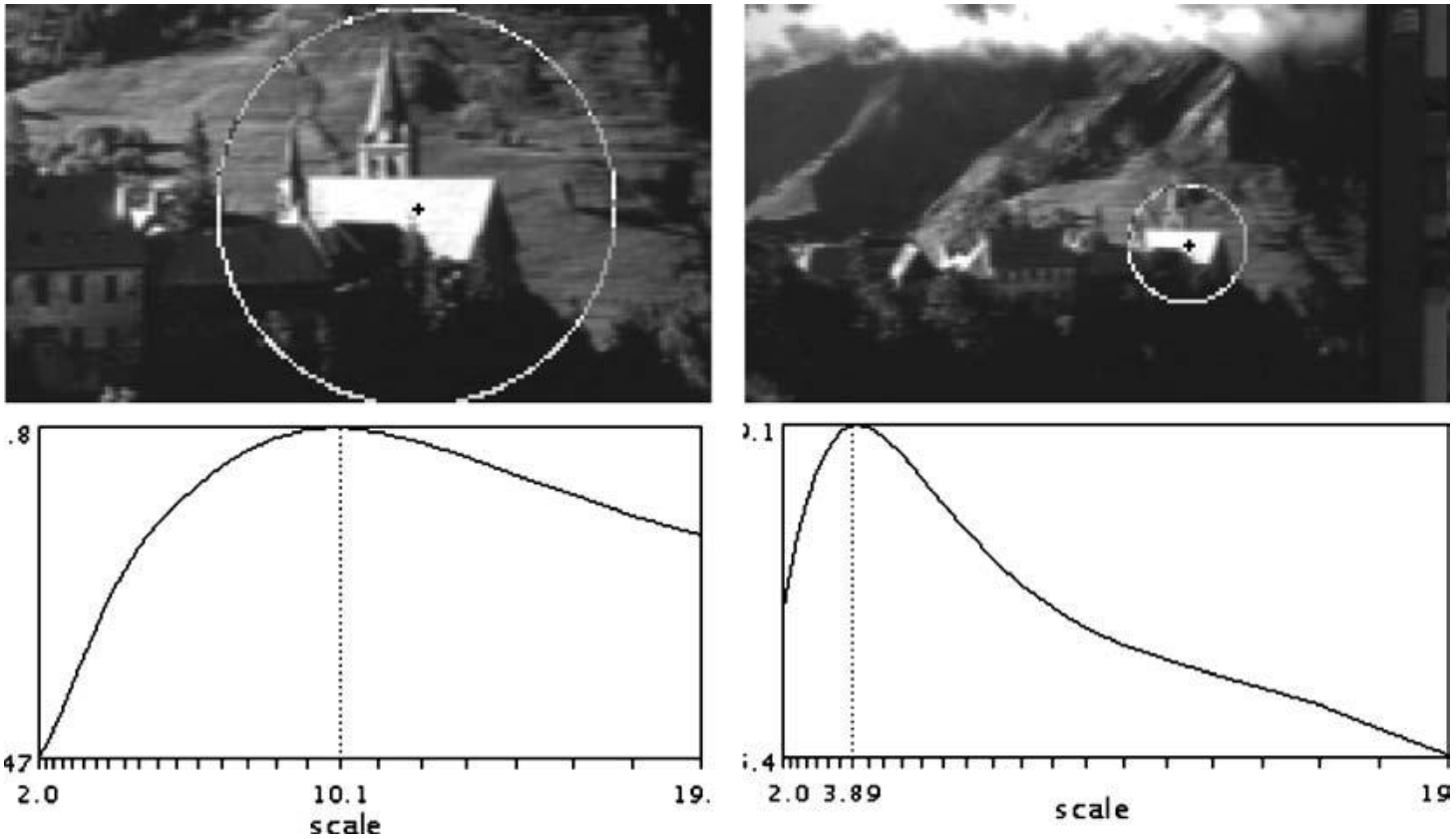
- **Properties of detectors**
  - Edge detectors
  - Corners
  - **Blobs**
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT

Blob detection



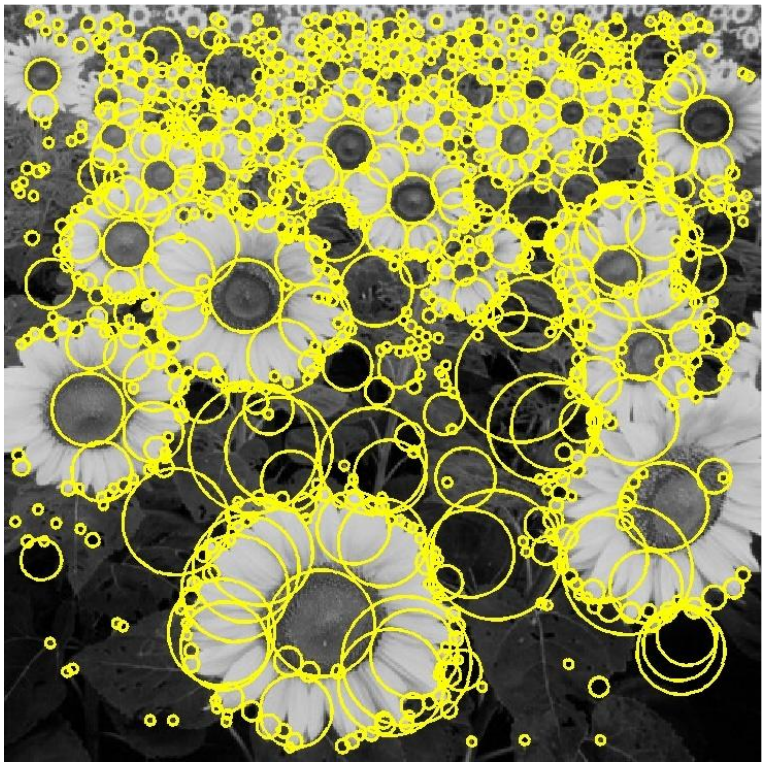
Feature detection with scale selection

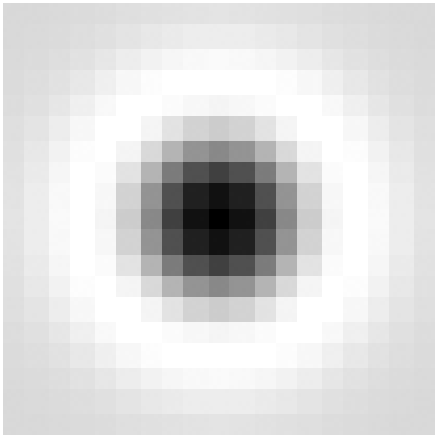
- **We want to extract features with characteristic scale that is *covariant* with the image transformation**



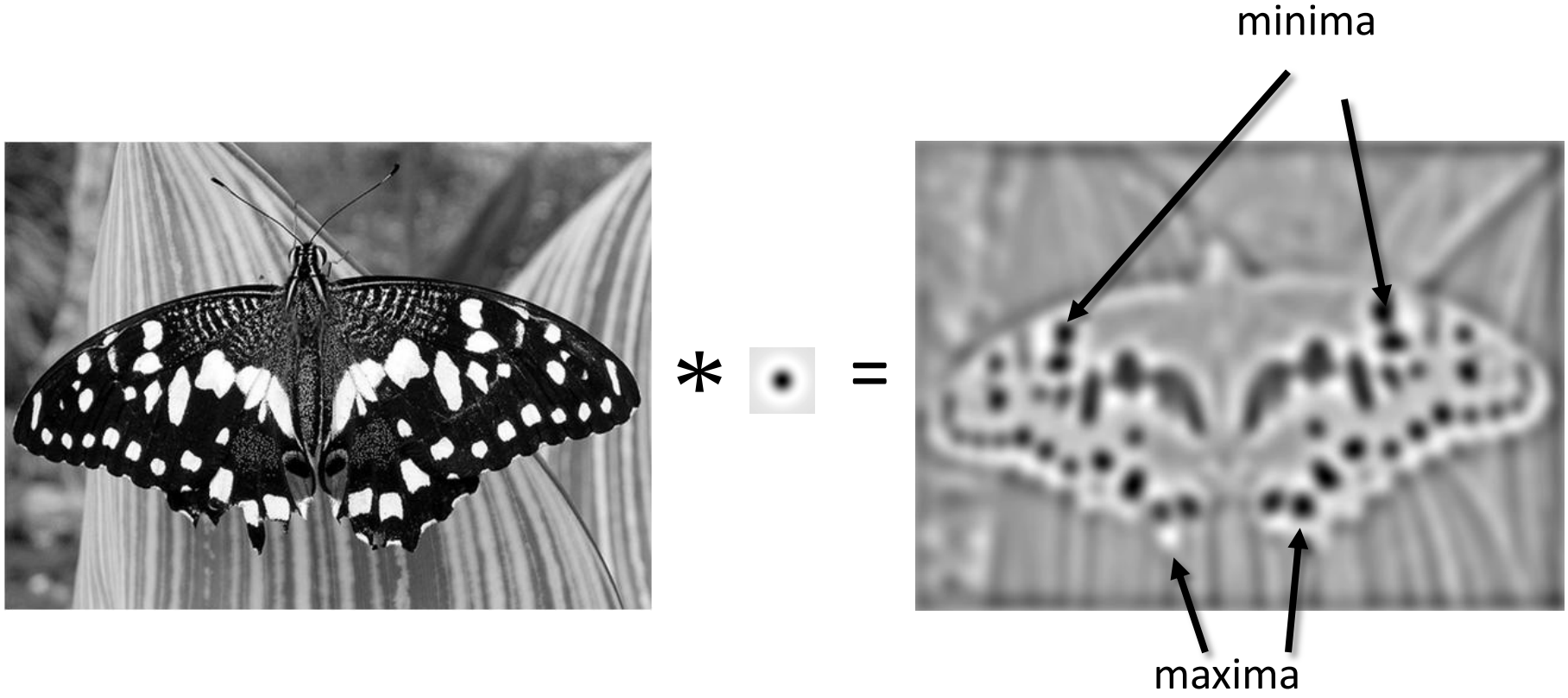
Blob detection: Basic idea

- **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

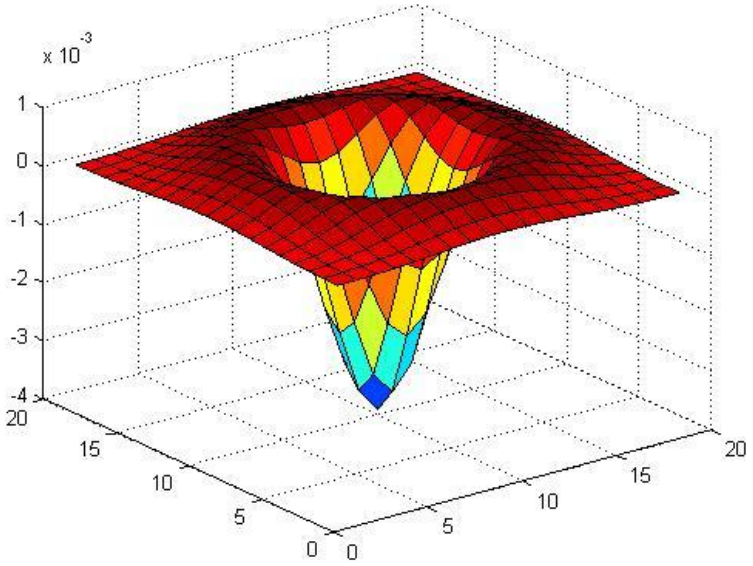


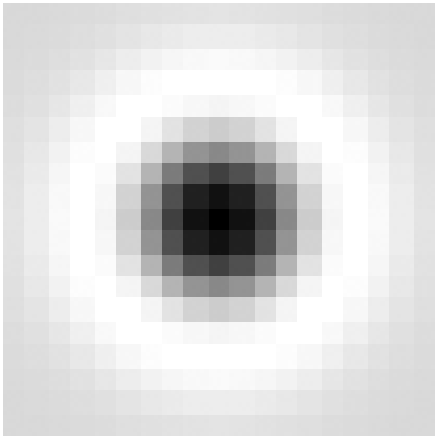
- **Find maxima *and minima* of blob filter response in space *and scale***

Source: N. Snavely

Blob filter

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

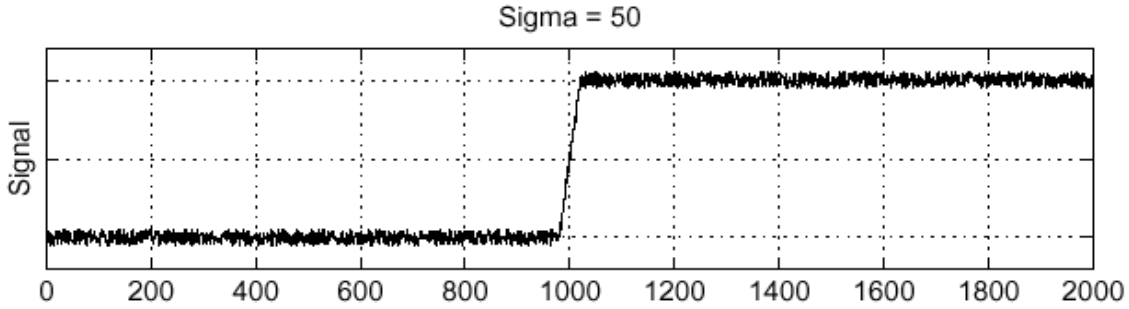




$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

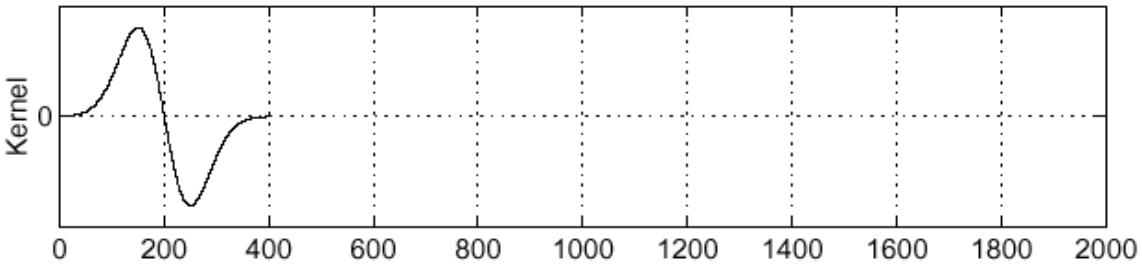
Recall: Edge detection

f



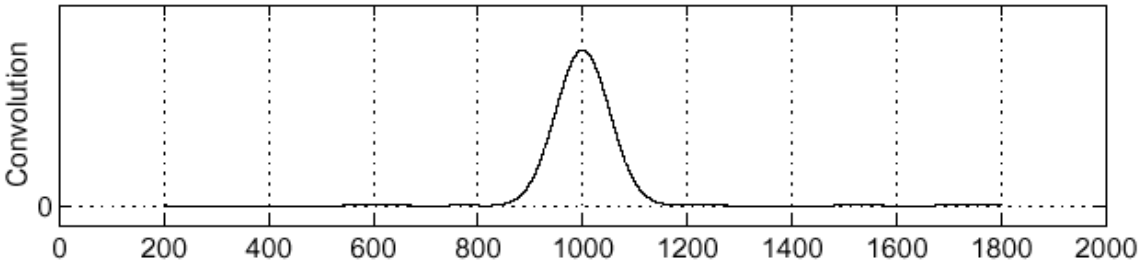
Edge

$\frac{d}{dx} g$



Derivative of Gaussian

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

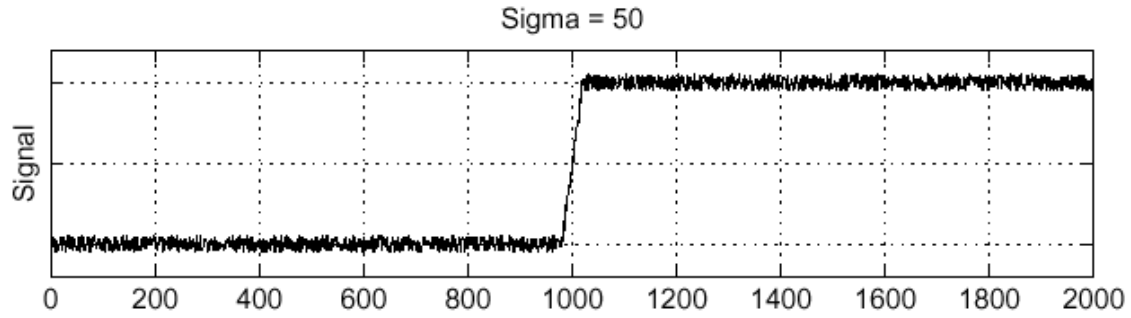


Edge = maximum of derivative

Source: S. Seitz

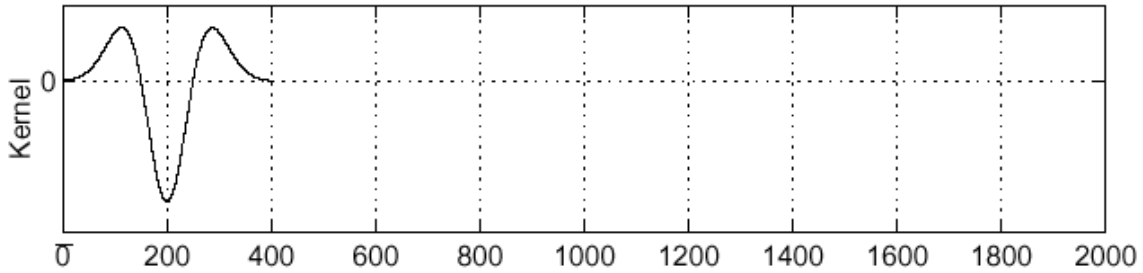
Edge detection, Take 2

f



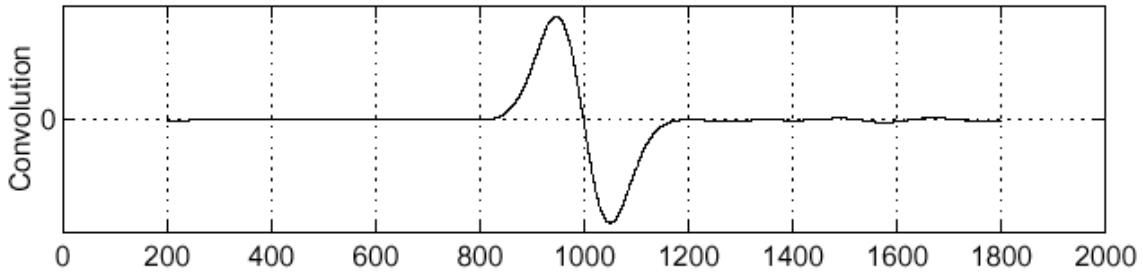
Edge

$$\frac{d^2}{dx^2} g$$



Second derivative of Gaussian (Laplacian)

$$f * \frac{d^2}{dx^2} g$$

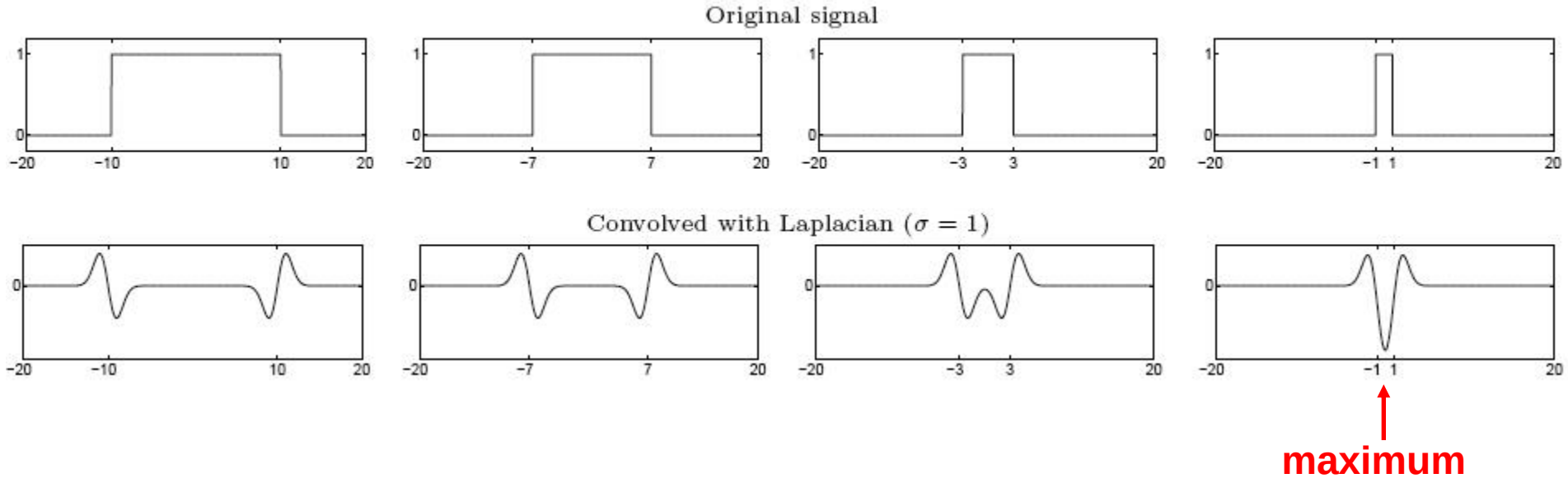


Edge = zero crossing of second derivative

Source: S. Seitz

From edges to blobs

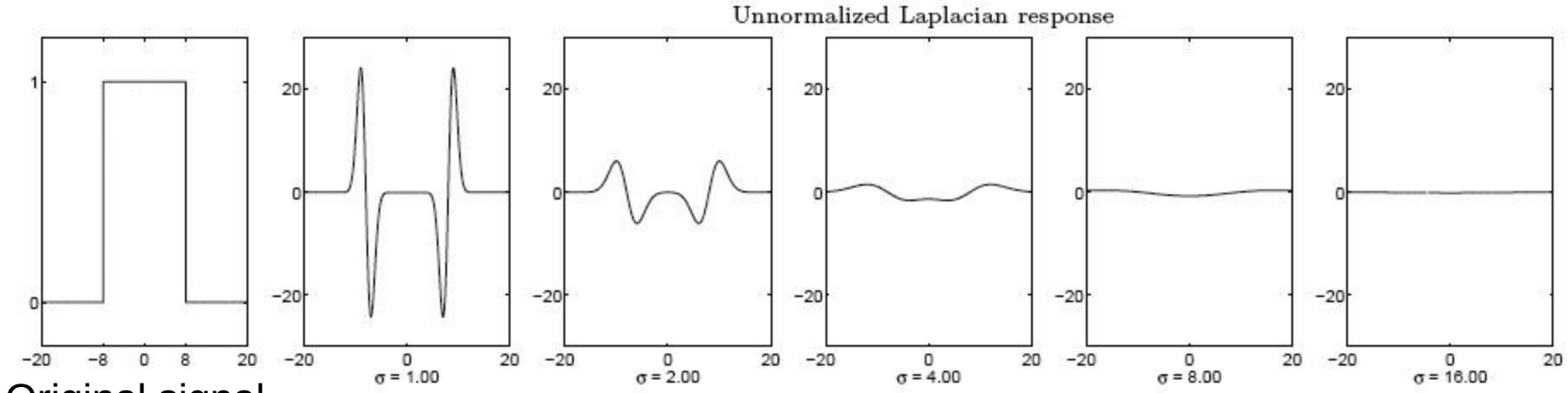
- **Edge = ripple**
- **Blob = superposition of two ripples**



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

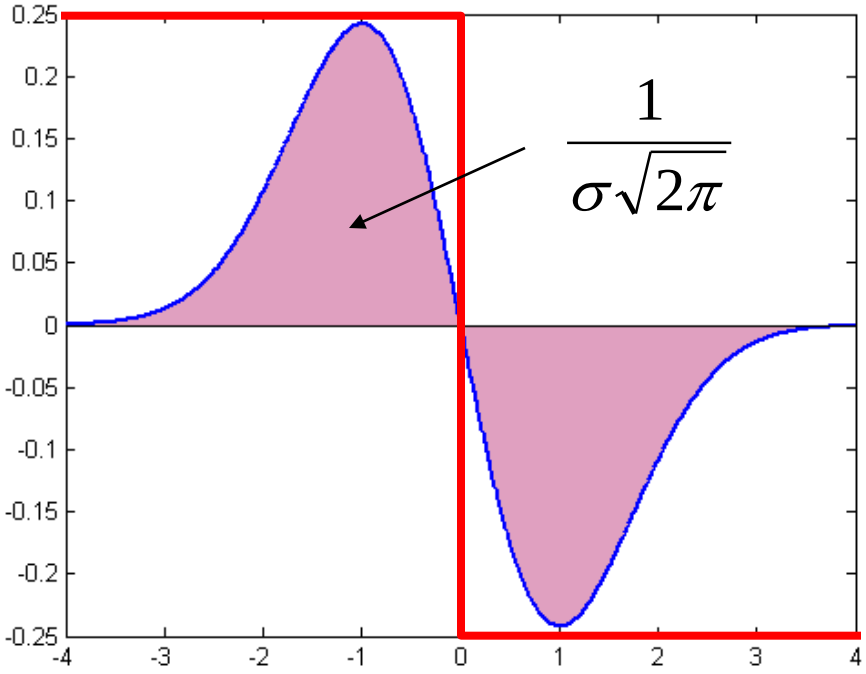
Scale selection

- **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**



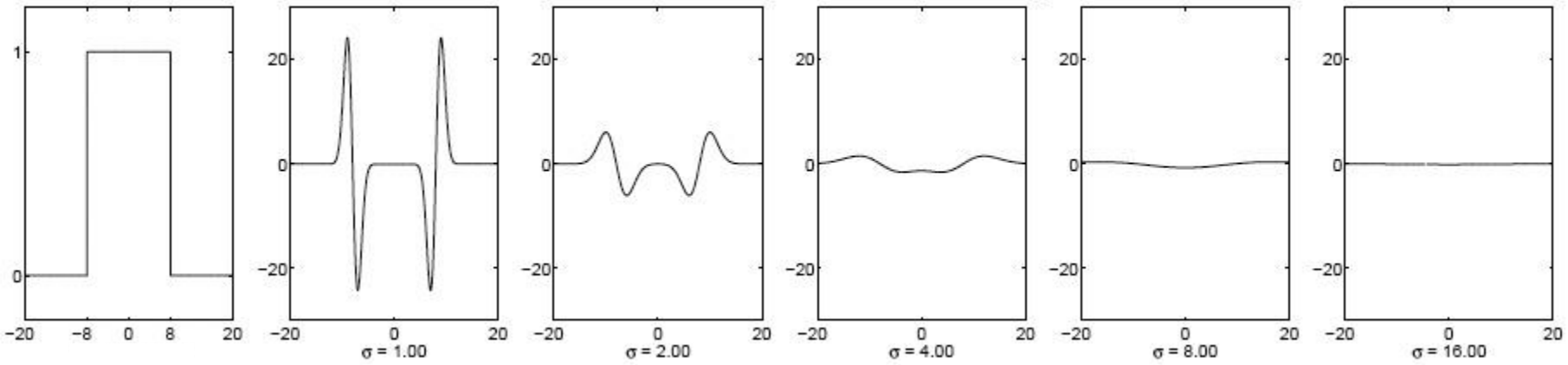
Scale normalization

- 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

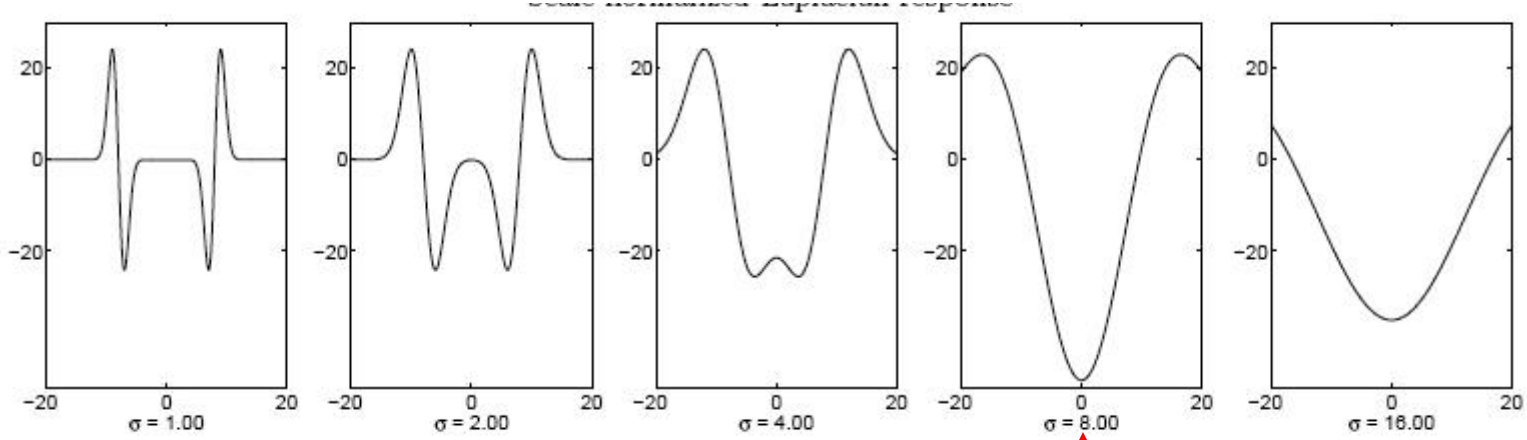
Effect of scale normalization

Original signal

Unnormalized Laplacian response



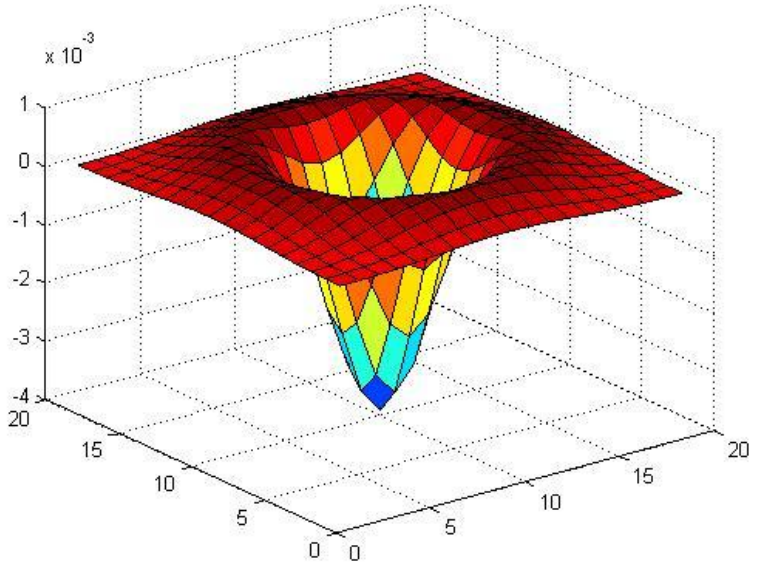
Scale-normalized Laplacian response

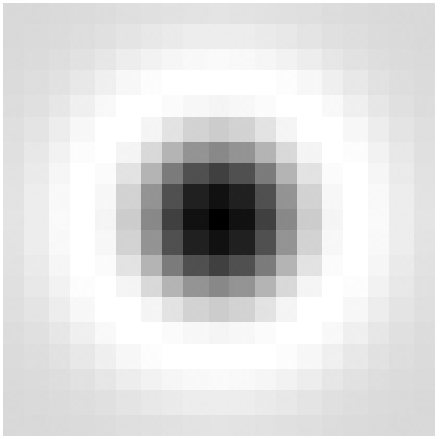


maximum

Blob detection in 2D

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



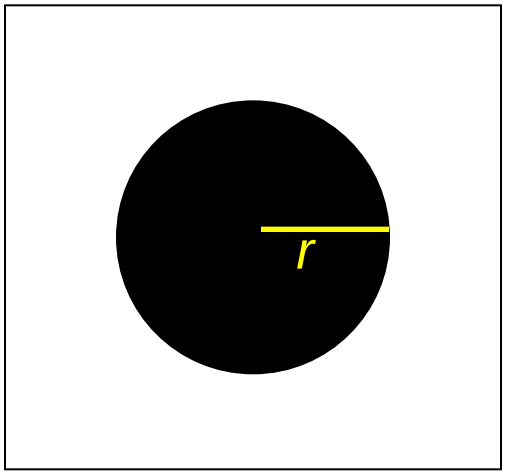


Scale-normalized:

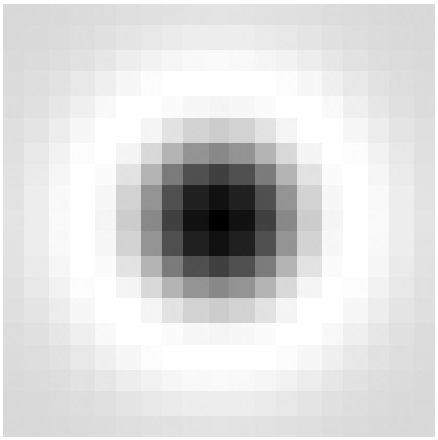
$$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

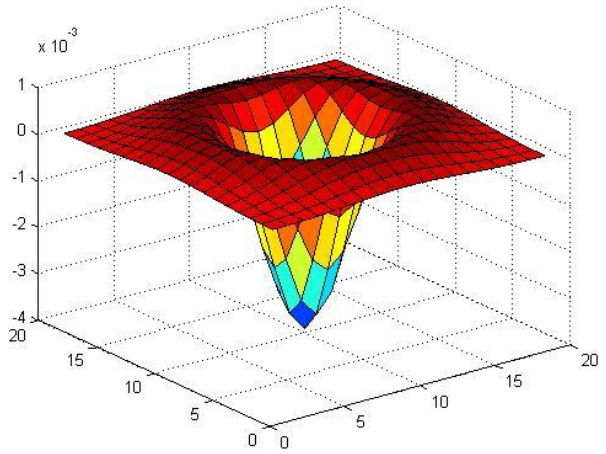
Scale selection

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



image





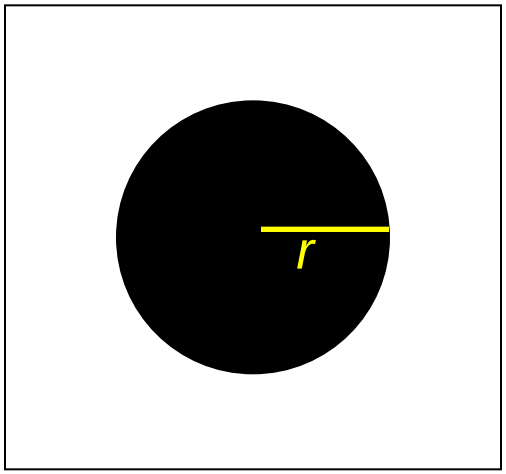
Laplacian

Scale selection

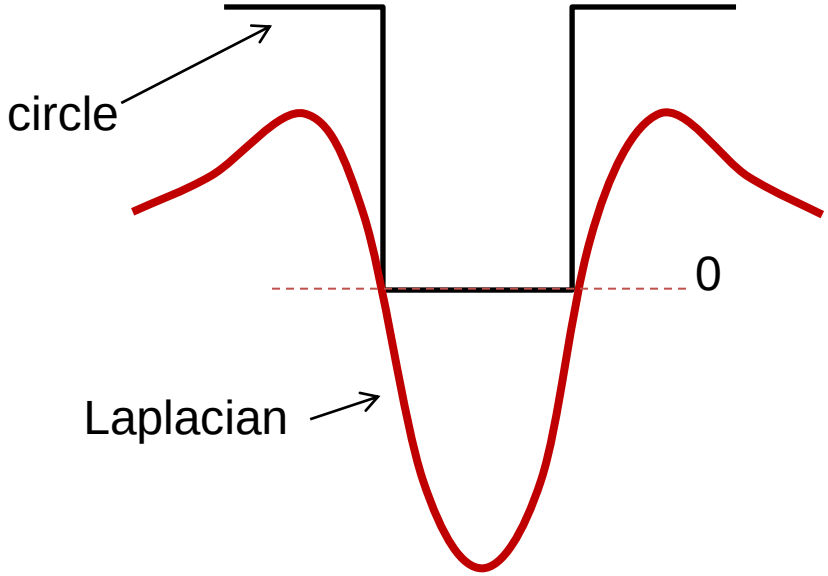
- **At what scale does the Laplacian achieve a maximum response to a binary circle of radius r?**
- **To get maximum response, the zeros of the Laplacian have to be aligned with the circle**
- **The Laplacian is given by (up to scale):**

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

- **Therefore, the maximum response occurs at** $\sigma = r / \sqrt{2}.$

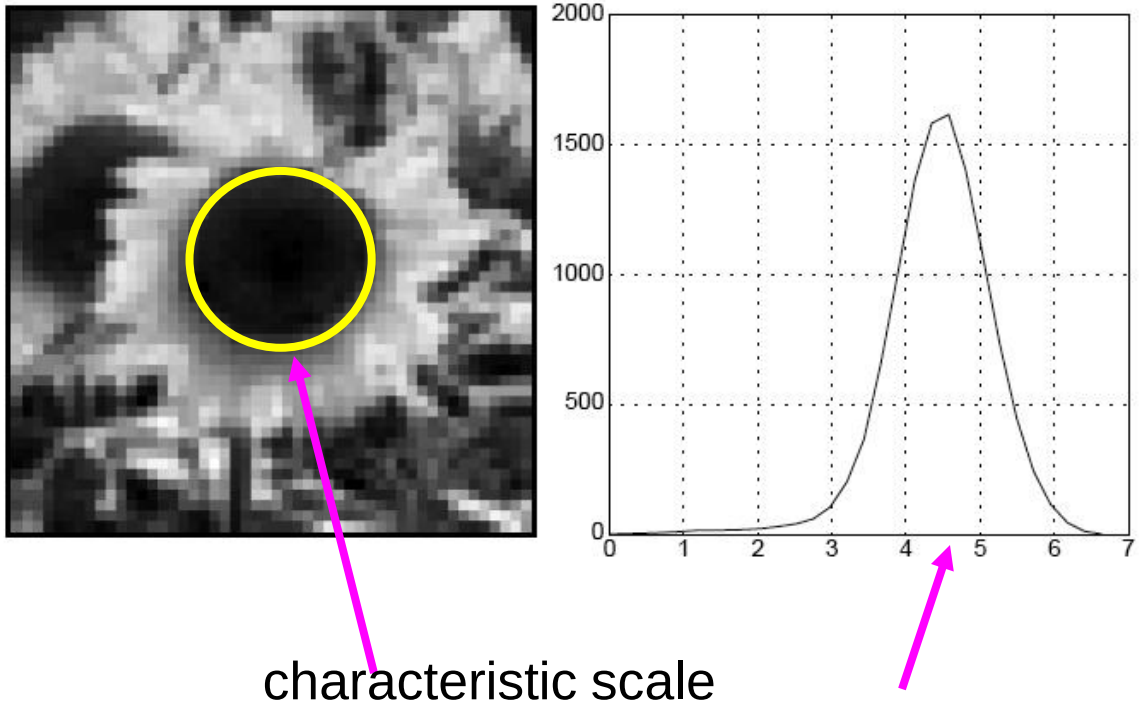


image



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

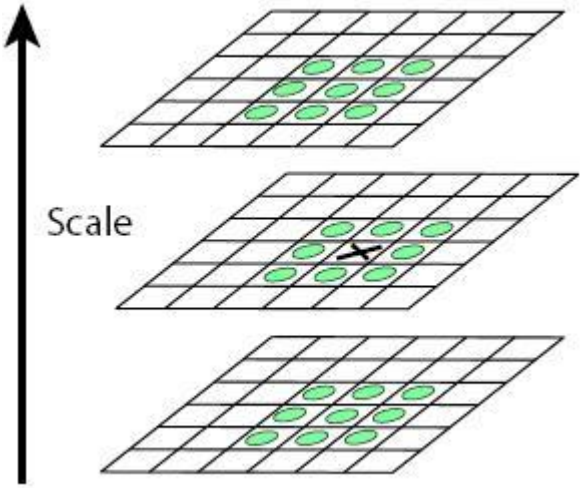
Scale-space blob detector: Example

Scale-space blob detector: Example

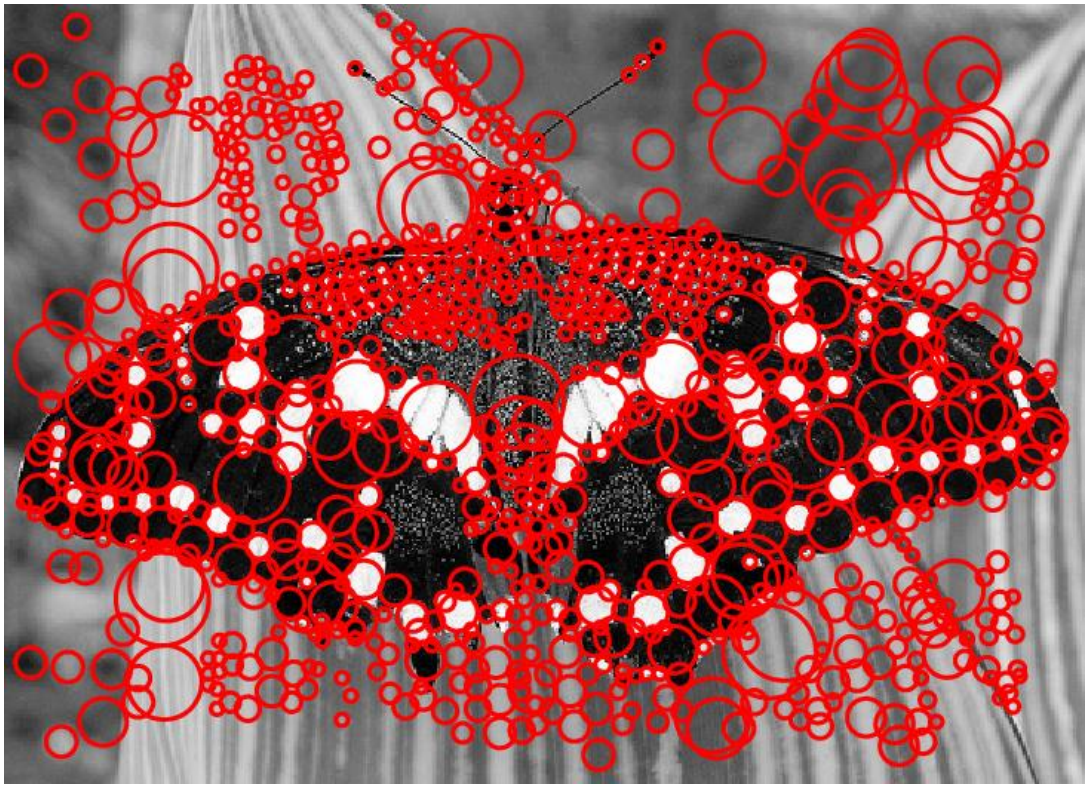
sigma = 11.9912

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

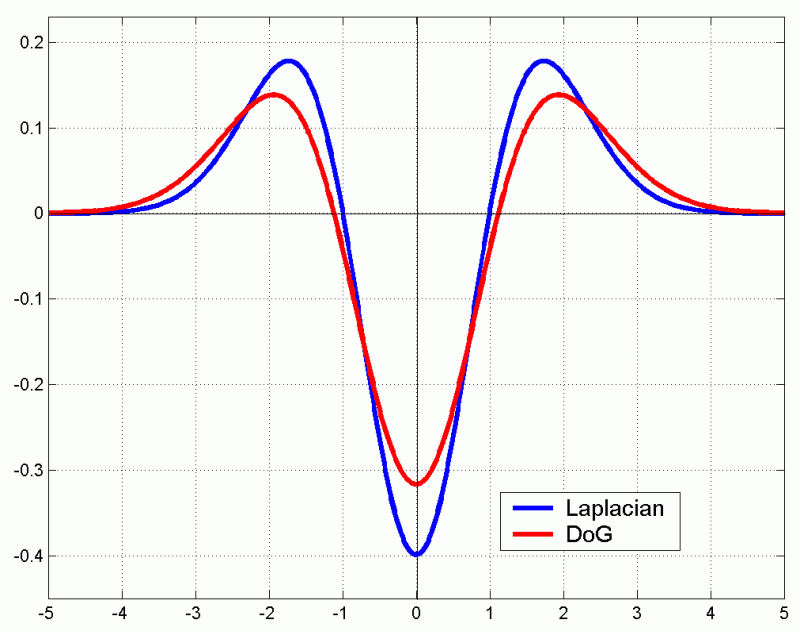
- **Approximating the Laplacian with a difference of Gaussians:**

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

(Laplacian)

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

(Difference of Gaussians)



Efficient implementation

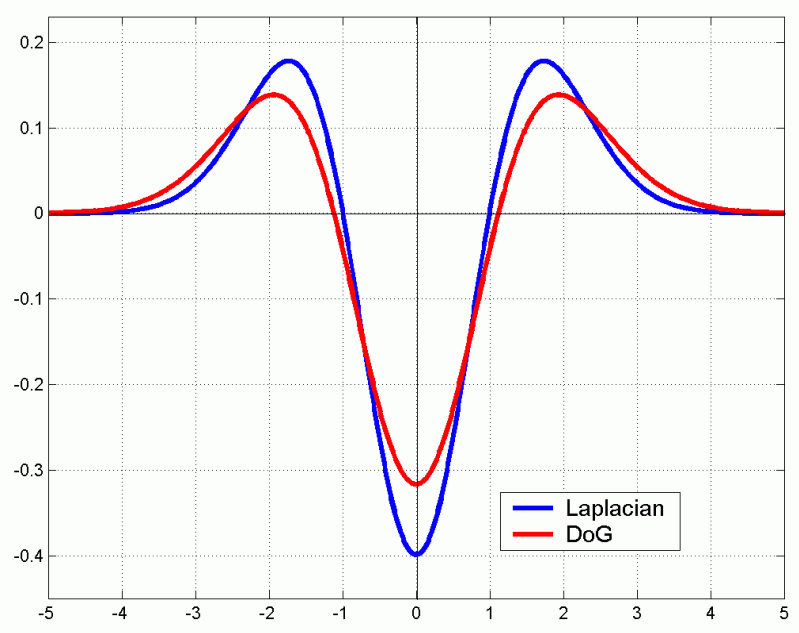
- **Approximating the Laplacian with a difference of Gaussians:**

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

(Laplacian)

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

(Difference of Gaussians)



Local Image Features

- **Properties of detectors**
  - Edge detectors
  - Corners
  - Blobs
  - **Scale invariant detection**
- **Properties of descriptors**
  - HOG
  - SIFT

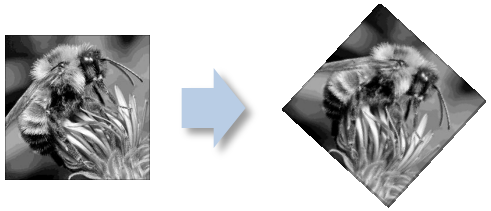
Invariance and covariance

- **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**

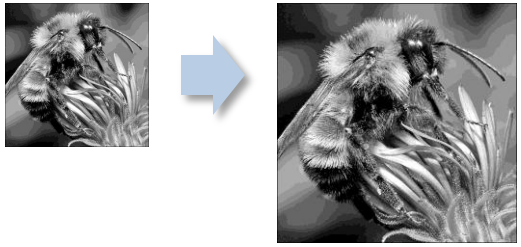
Image transformations

- **Geometric**

Rotation

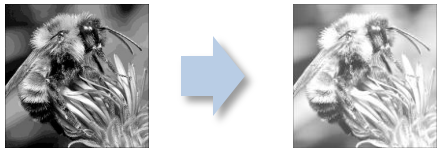


Scale

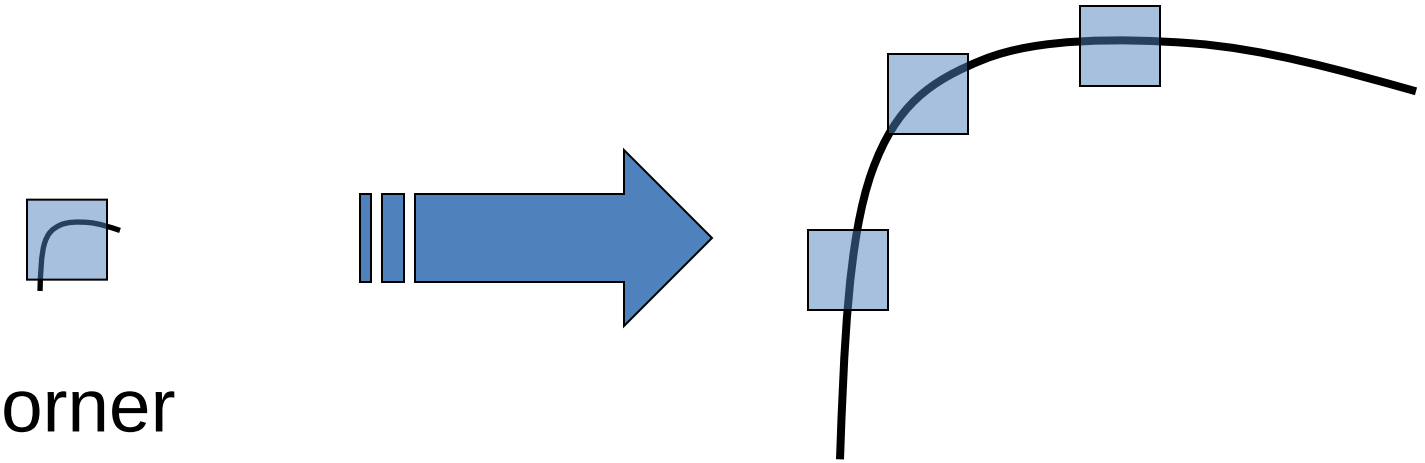


- **Photometric**

Intensity change



Scaling



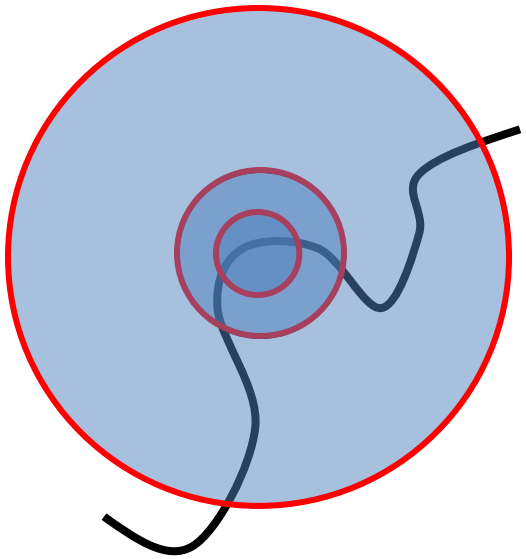
Corner

All points will be classified as edges

Corner location is not covariant to scaling!

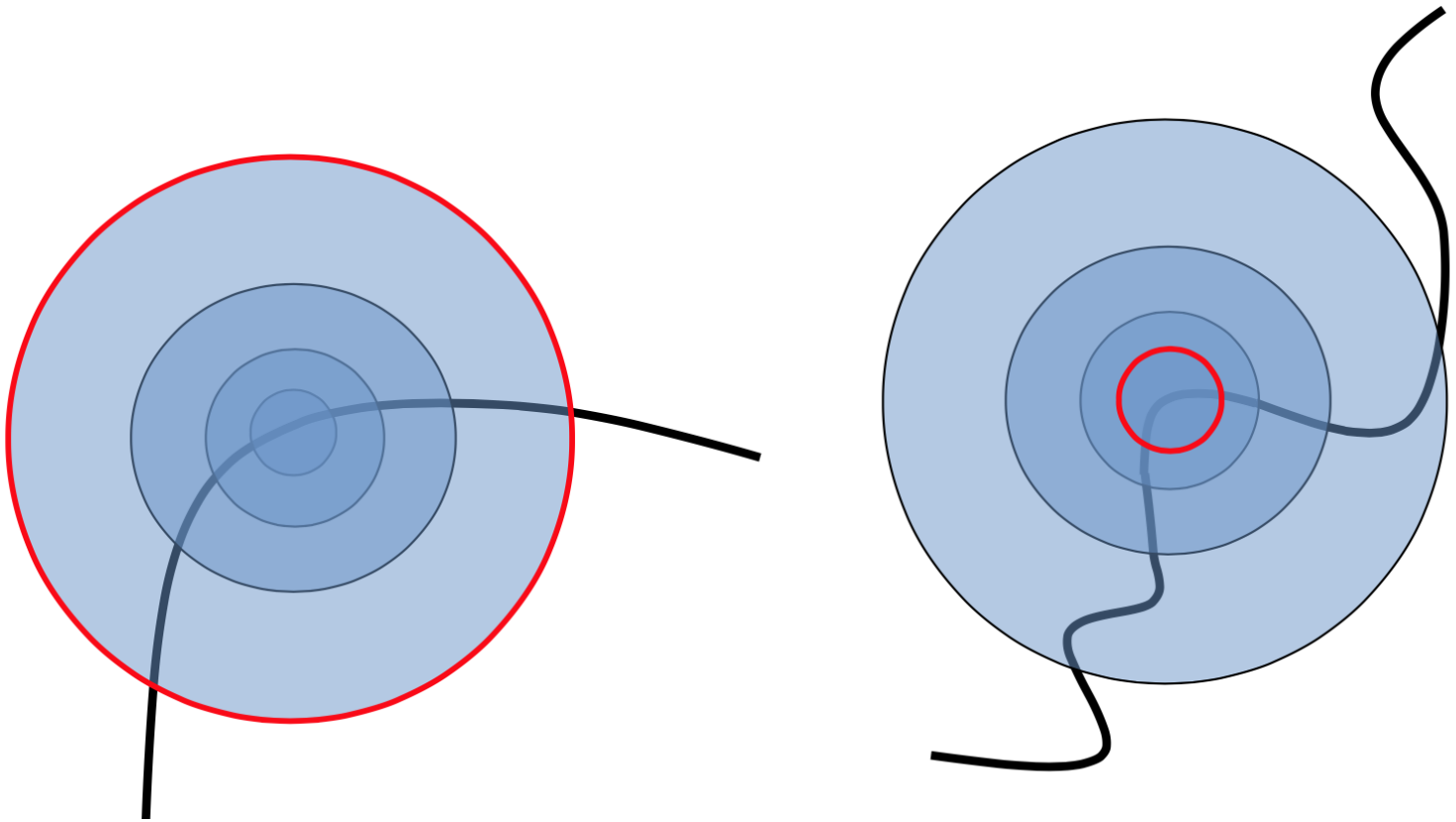
Scale invariant detection

Suppose you're looking for corners



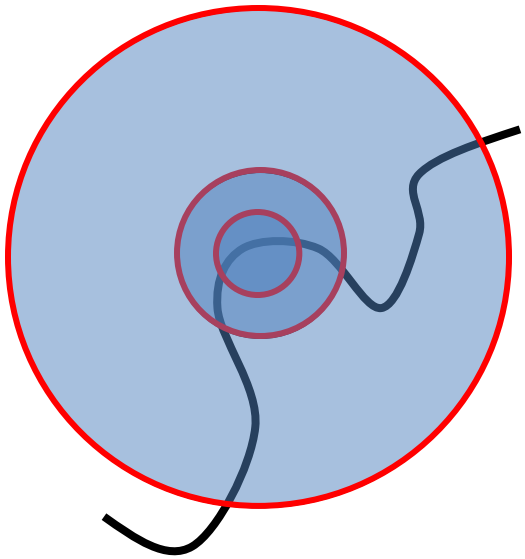
Q: How to find circle of right size?

- The problem: how do we choose corresponding circles ***independently*** in each image?



Scale invariant detection

Suppose you're looking for corners



Q: How to find circle of right size?

Key idea: find scale that gives local maximum of *f*

- **in both position and scale**
- **One definition of *f*: the Harris operator**

Solution

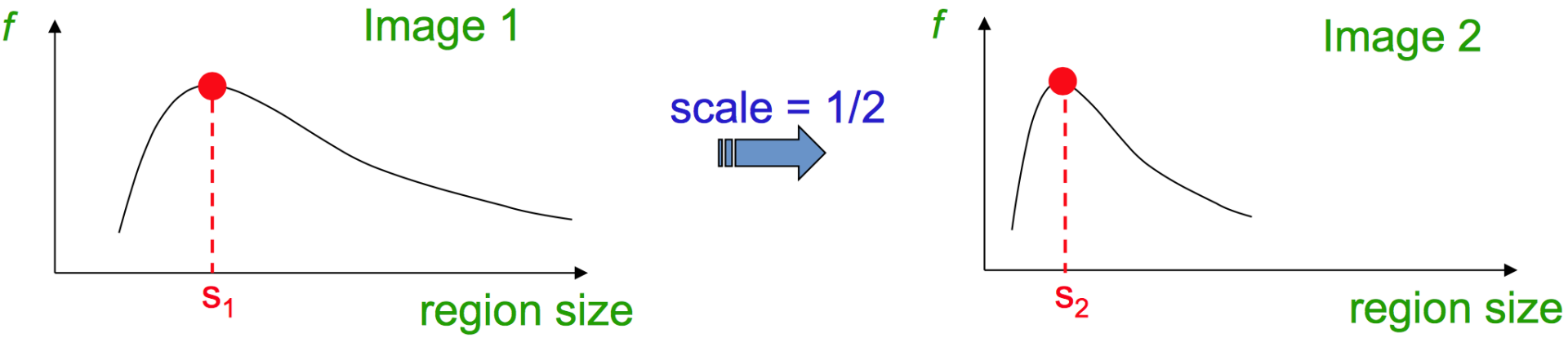
- **Design a function on the region (circle) which is "scale invariant"**
  - **i.e., the same for corresponding regions, even if at different scales**
  - **E.g., average intensity. Same even for different sizes**

- **For a point in one image, consider it as a function of region size (circle radius)**

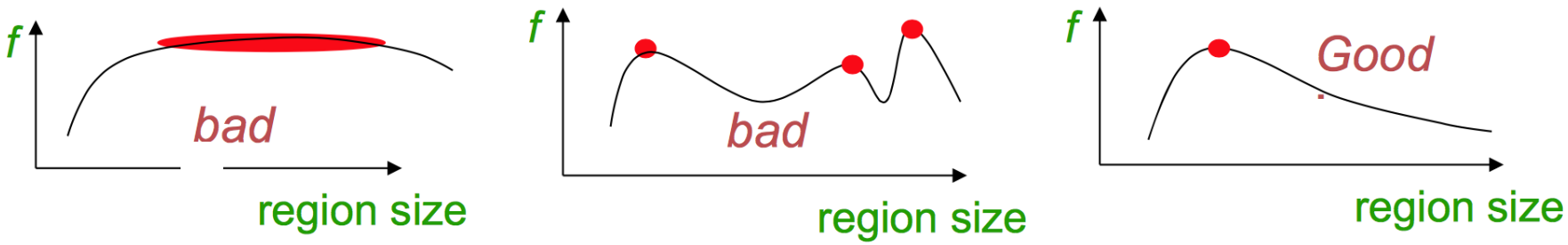
- Common approach:
  Take a local maximum of this function

- Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

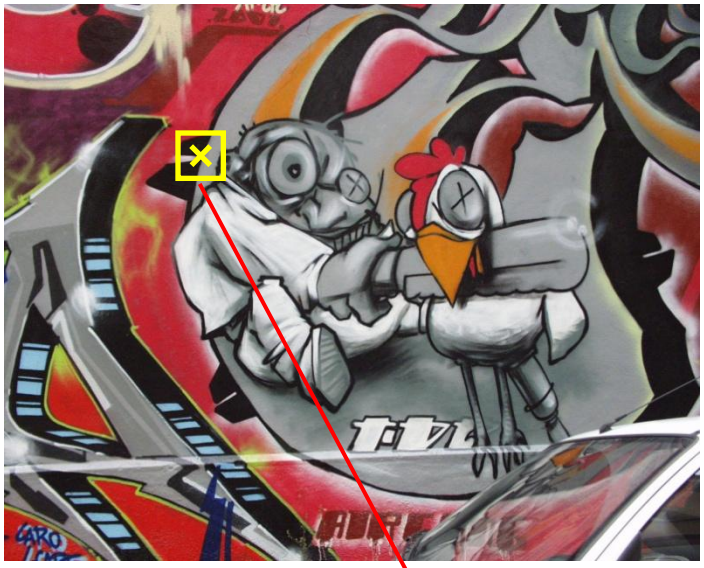
Important: this scale invariant region size is found in each image independently!



- A “good” function for scale detection:
  has one stable sharp peak



Automatic Scale Selection

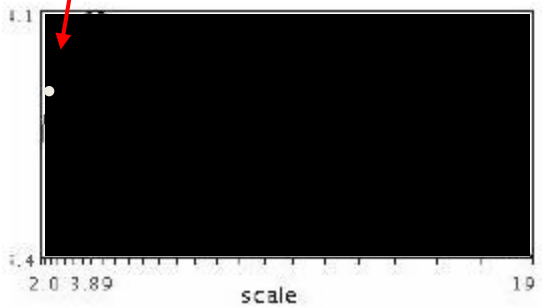


$$f(I_{i_1 \ldots i_m}(x,\sigma)) \;=\; f(I_{i_1 \ldots i_m}(x',\sigma'))$$

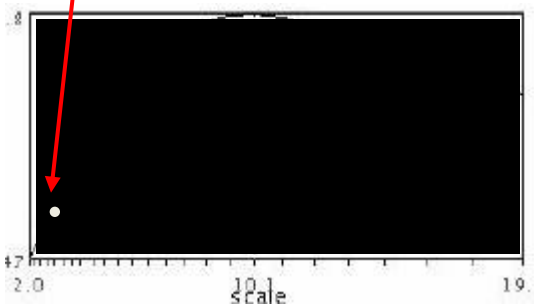
How to find corresponding patch sizes?

Automatic Scale Selection

- **Function responses for increasing scale (scale signature)**

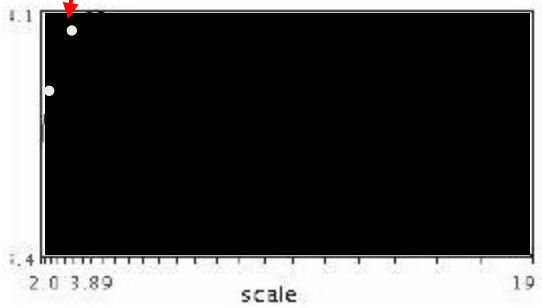


$f(I_{i_1 \ldots i_m}(x,\sigma))$

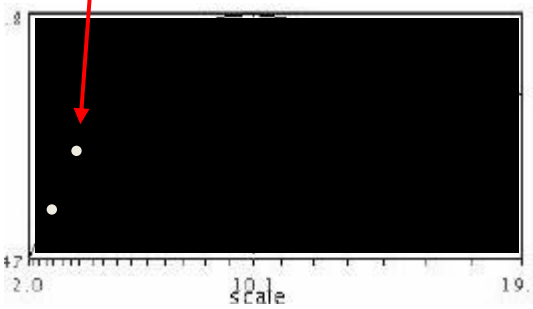


$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection

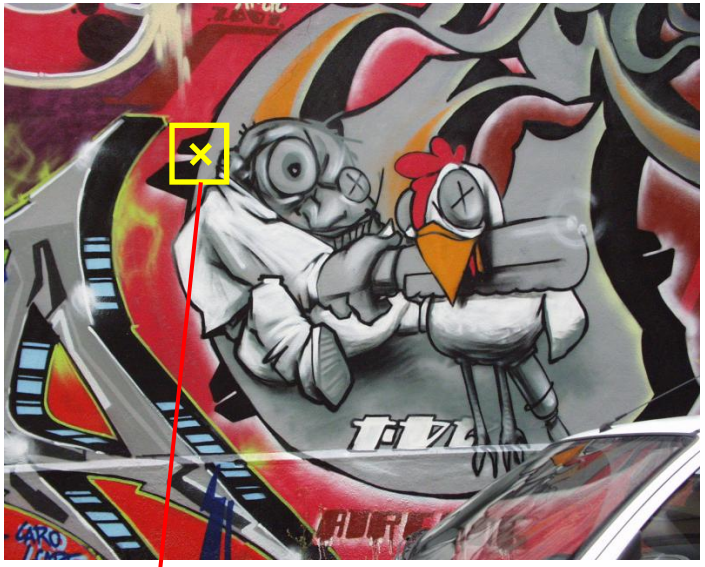


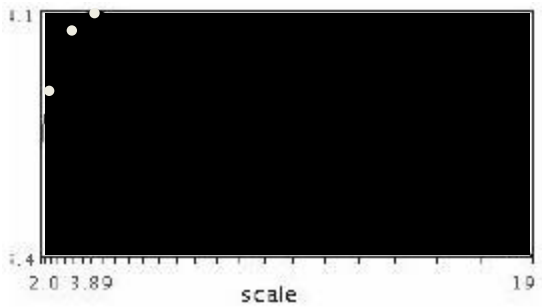
$f(I_{i_1 \ldots i_m}(x,\sigma))$



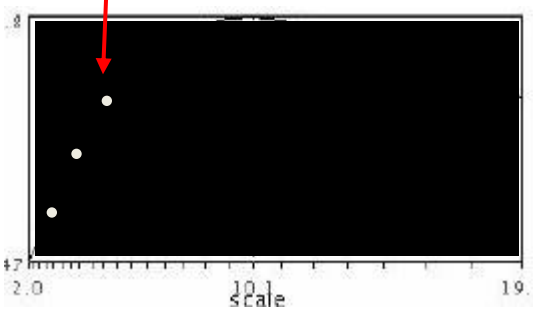
$f(I_{i_1 \ldots i_m}(x',\sigma))$

Automatic Scale Selection





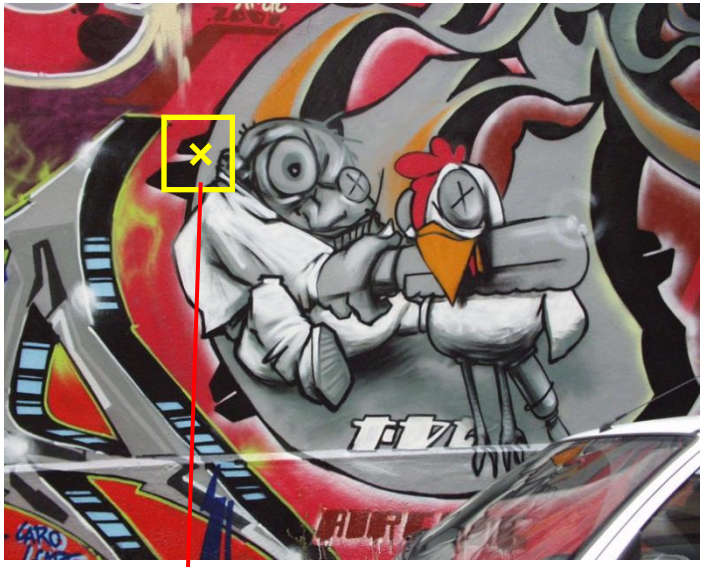
$f(I_{i_1 \ldots i_m}(x,\sigma))$

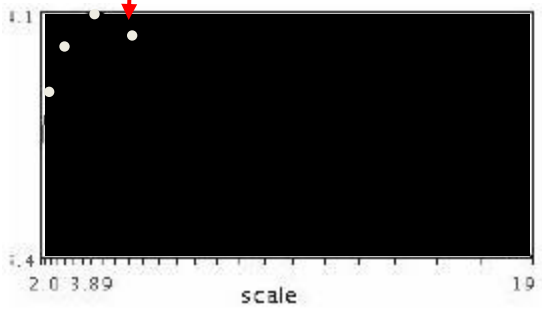


$f(I_{i_1 \ldots i_m}(x',\sigma))$

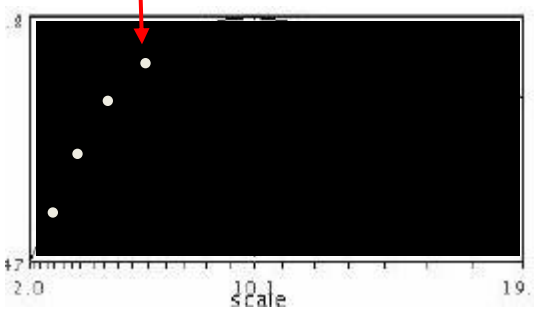
K. Grauman, B. Leibe

Automatic Scale Selection



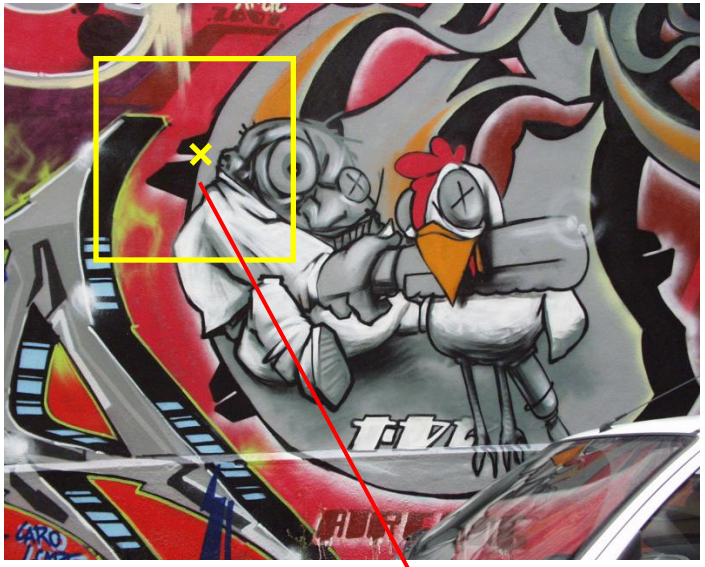


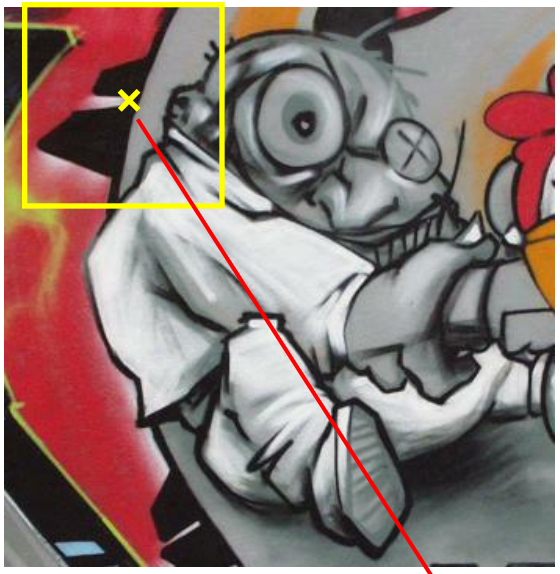
$f(I_{i_1 \ldots i_m}(x,\sigma))$

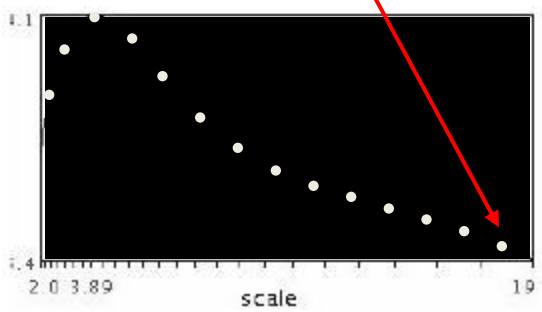


$f(I_{i_1 \ldots i_m}(x',\sigma))$

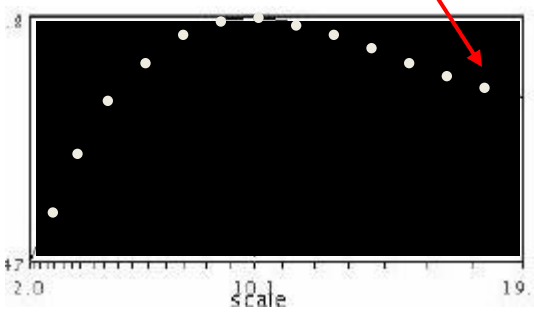
Automatic Scale Selection







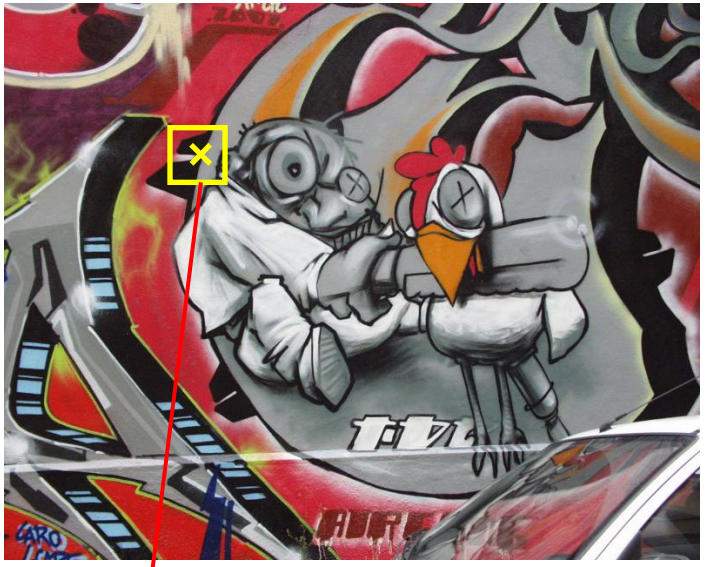
$f(I_{i_1 \ldots i_m}(x,\sigma))$

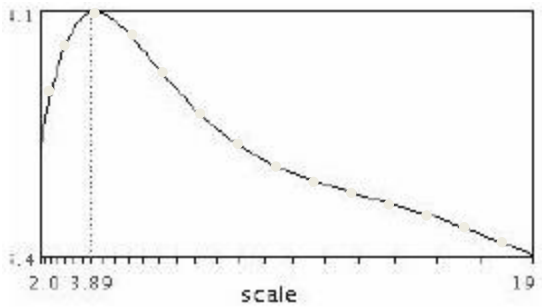


$f(I_{i_1 \ldots i_m}(x',\sigma))$

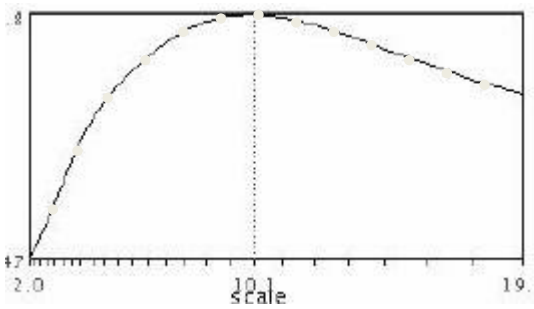
K. Grauman, B. Leibe

Automatic Scale Selection





$f(I_{i_1\ldots i_m}(x,\sigma))$

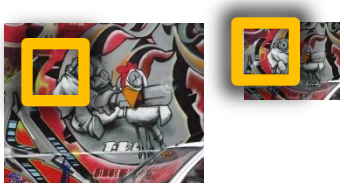


$f(I_{i_1\ldots i_m}(x',\sigma'))$

K. Grauman, B. Leibe

Implementation

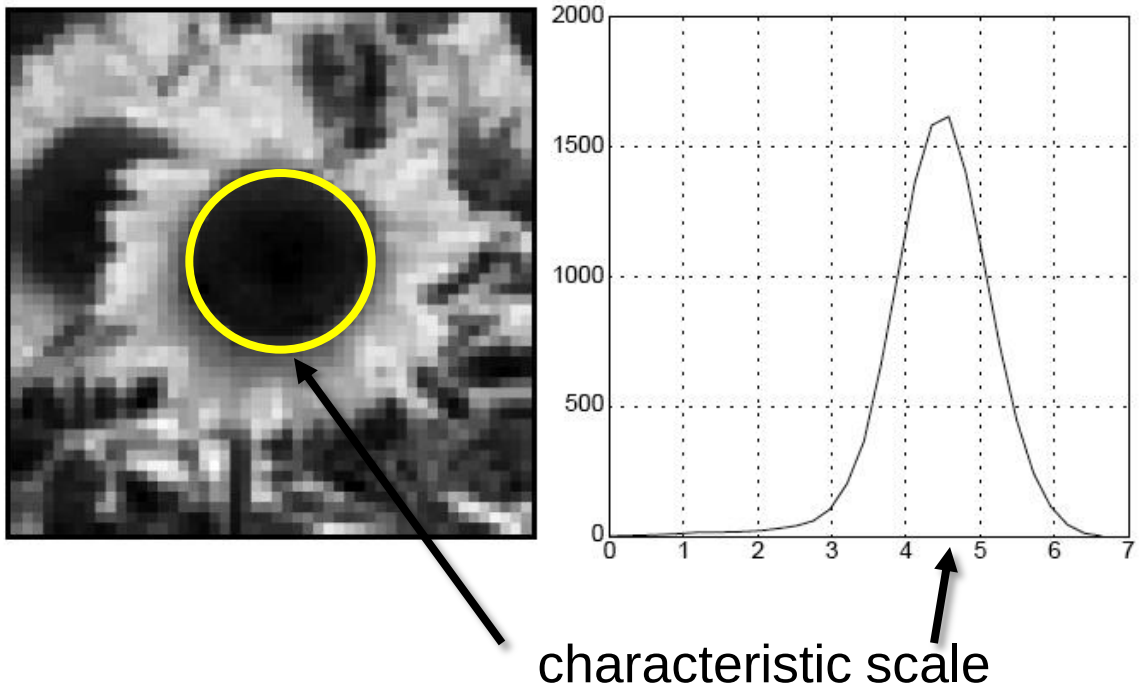
- **Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid**



(sometimes need to create in-between levels, e.g. a ¾-size image)

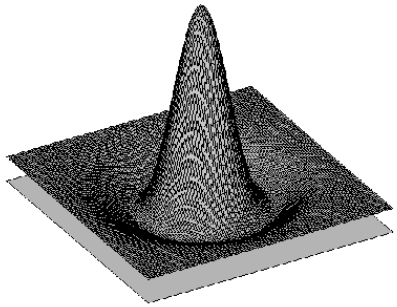
Characteristic scale

- **We define the characteristic scale as the scale that produces peak of Laplacian response**



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

Difference-of-Gaussian (DoG)



-

=

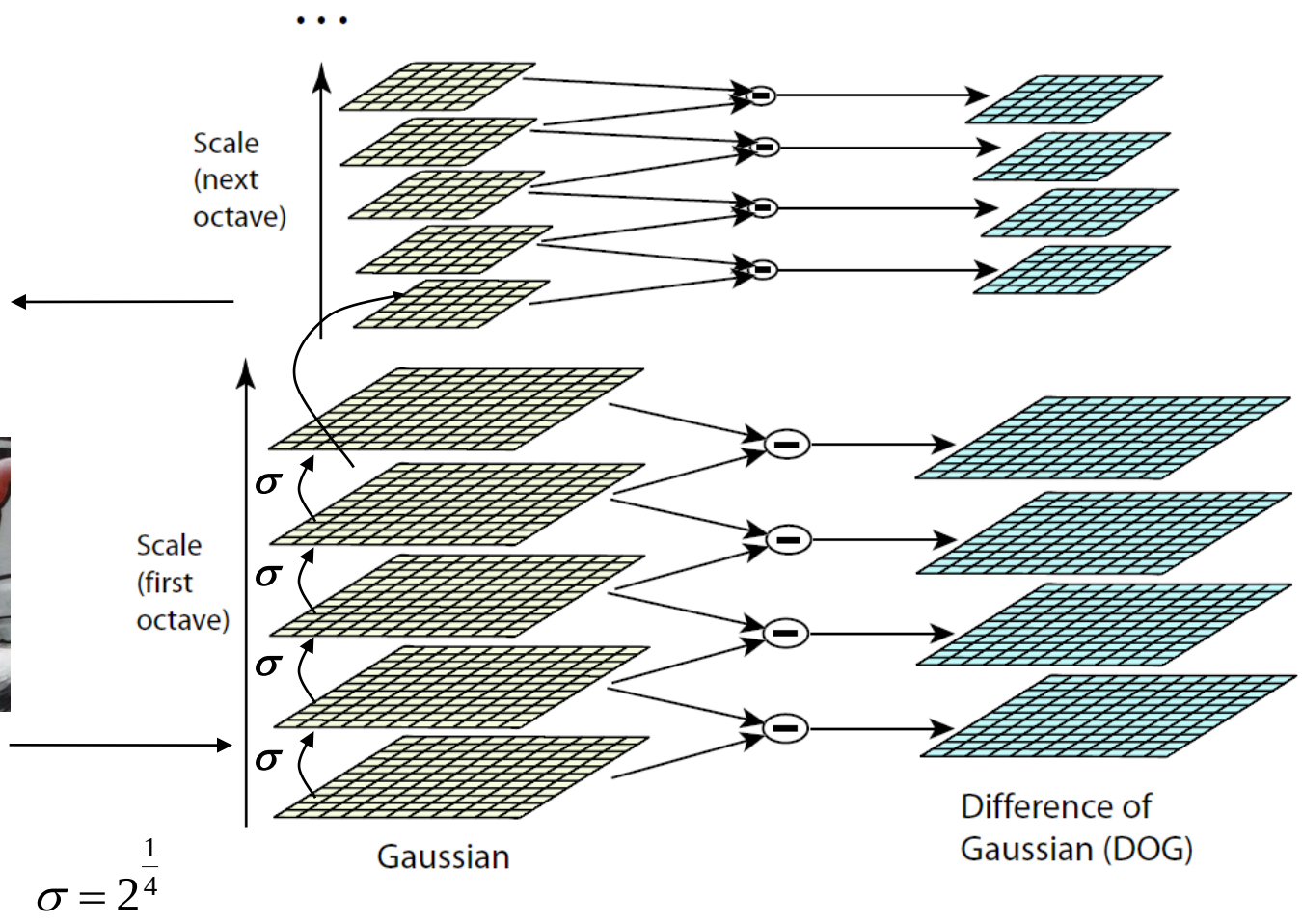
K. Grauman, B. Leibe

DoG – Efficient Computation

- **Computation in Gaussian scale pyramid**

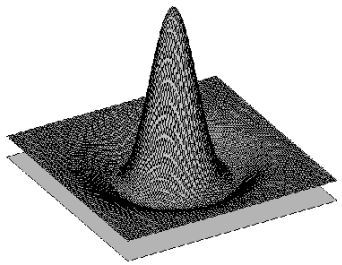
Sampling with step $\sigma^4 = 2$

Original image

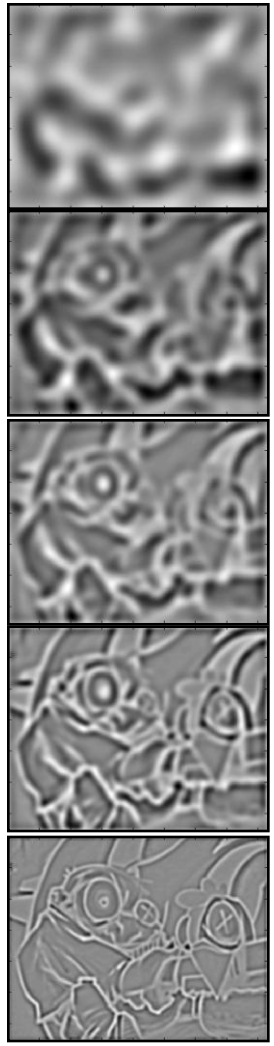


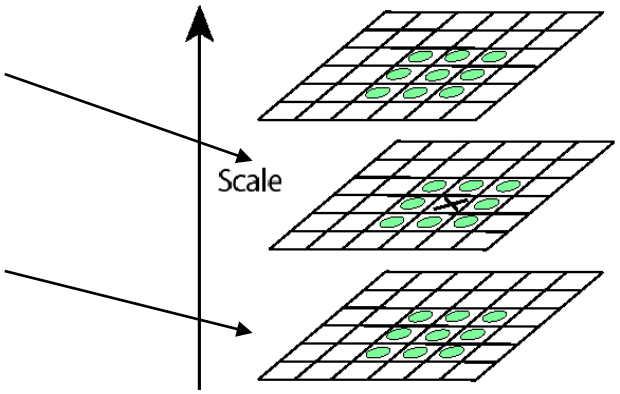
$$\sigma = 2^{\frac{1}{4}}$$

Find local maxima in position-scale space of Difference-of-Gaussian



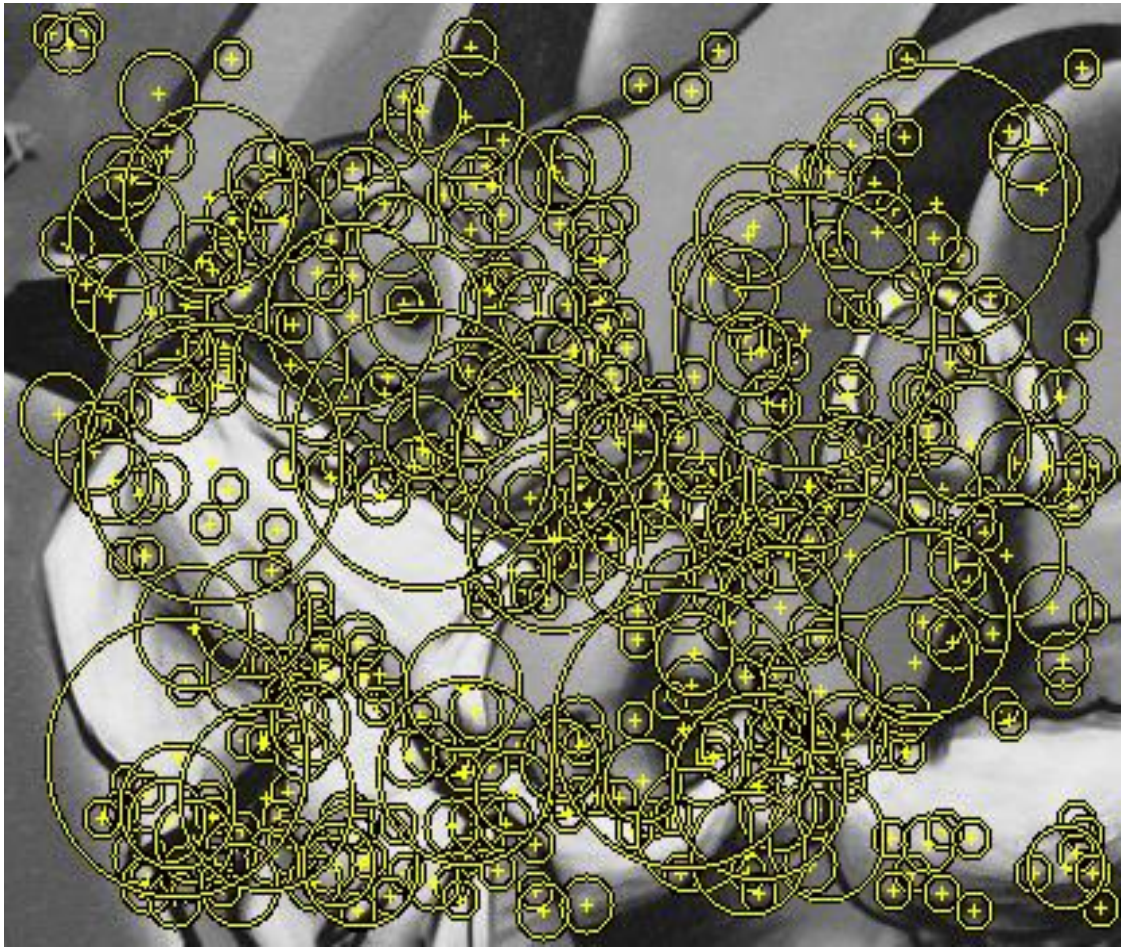
$$L_{xx}(\sigma) + L_{yy}(\sigma)$$





⇒ List of
(x, y, s)

Results: Difference-of-Gaussian



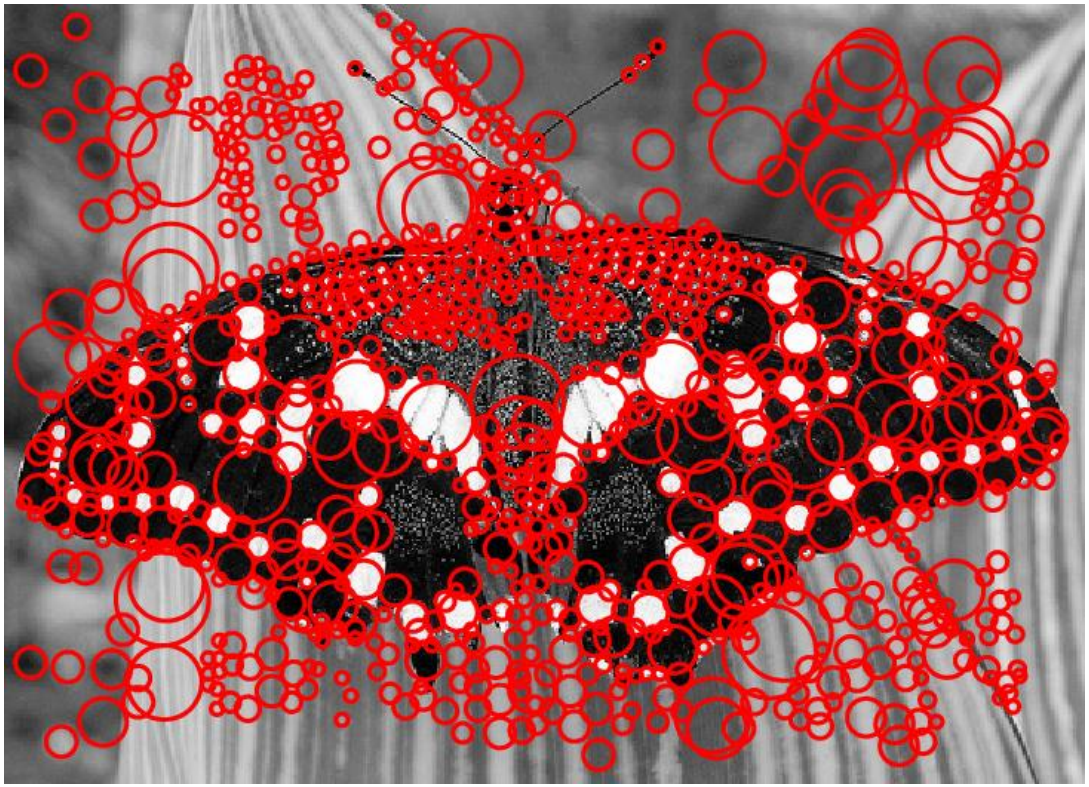
K. Grauman, B. Leibe

Scale-space blob detector: Example

Scale-space blob detector: Example

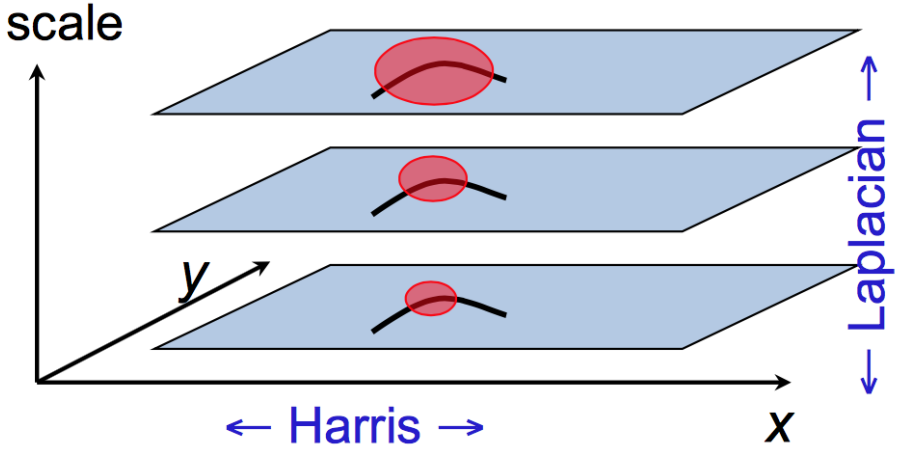
sigma = 11.9912

Scale-space blob detector: Example

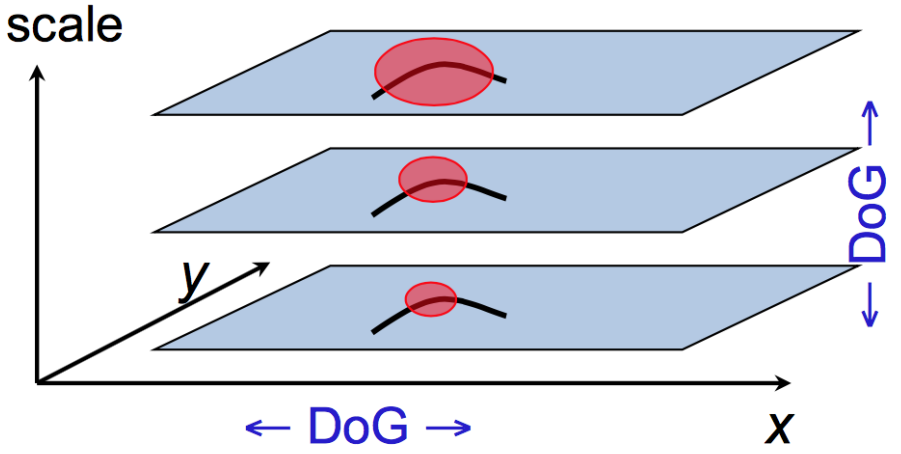


Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



[1] K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001
[2] D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". IJCV 2004

Comparison of Keypoint Detectors

Table 7.1 Overview of feature detectors.

Feature Detector	Corner	Blob	Region	Rotation invariant	Scale invariant	Affine invariant	Localization Repeatability	Localization accuracy	Robustness	Efficiency
Harris	√			√			+++	+++	+++	++
Hessian		√		√			++	++	++	+
SUSAN	√			√			++	++	++	+++
Harris-Laplace	√	(√)		√	√		+++	+++	++	+
Hessian-Laplace	(√)	√		√	√		+++	+++	+++	+
DoG	(√)	√		√	√		++	++	++	++
SURF	(√)	√		√	√		++	++	++	+++
Harris-Affine	√	(√)		√	√	√	+++	+++	++	++
Hessian-Affine	(√)	√		√	√	√	+++	+++	+++	++
Salient Regions	(√)	√		√	√	(√)	+	+	++	+
Edge-based	√			√	√	√	+++	+++	+	+
MSER			√	√	√	√	+++	+++	++	+++
Intensity-based			√	√	√	√	++	++	++	++
Superpixels			√	√	(√)	(√)	+	+	+	+

Tuytelaars Mikolajczyk 2008

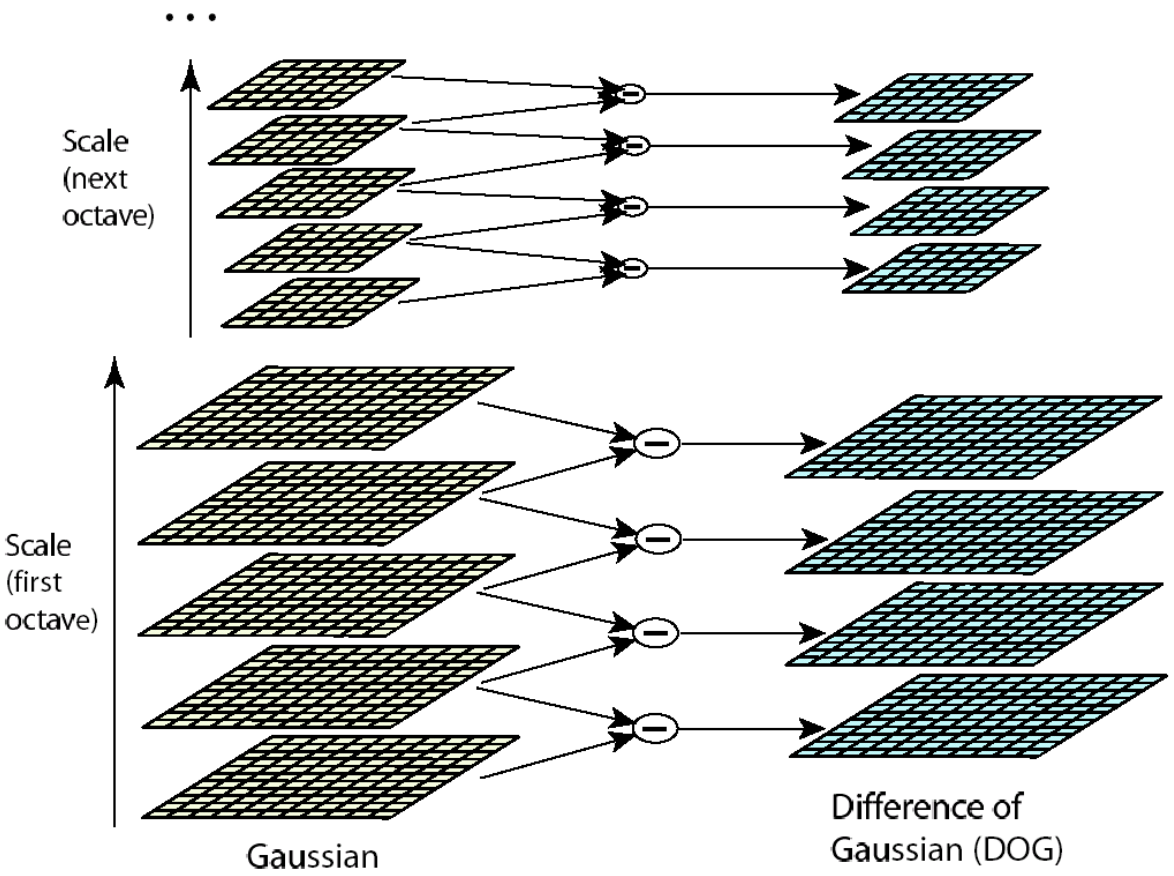
Scale Invariant Detection: Summary

- Given: two images of the same scene with a large *scale difference* between them
- Goal: find *the same* interest points *independently* in each image
- Solution: search for *maxima* of suitable functions in *scale* and in *space* (over the image)

Methods:

1. Harris-Laplacian [Mikolajczyk, Schmid]: maximize Laplacian over scale, Harris' measure of corner response over the image
2. SIFT [Lowe]: maximize Difference of Gaussians over scale and space

Efficient implementation



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

After the Lecture

- **Readings：**
  - **Chapter 5.1-5.3**
  - **David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.**
  - **T. Lindeberg (1998). "Feature detection with automatic scale selection." *International Journal of Computer Vision* 30 (2): pp 77--116.**