Computer Vision

Technische Universiteit Delft

Lecture 2
Filtering and interest points

Lecturer:
Dr. Jan van Gemert (tud.computervision@gmail.com)

Recap of last time

- Course logistics
- Pinhole
- Reflection

- Questions?

Image Filtering

A bit of Image processing

How to get rid of noisy pixels?

Simple solution: replace pixel by neighbourhood average

Moving Neighborhood Average

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

Source: S. Seitz

Moving Neighborhood Average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Source: S. Seitz

Moving Neighborhood Average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Source: S. Seitz

Moving Neighborhood Average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

Source: S. Seitz

Moving Neighborhood Average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10							

Source: S. Seitz

Moving Neighborhood Average

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20						

Source: S. Seitz

Moving Neighborhood Average

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20	30					

Source: S. Seitz

Moving Neighborhood Average

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

10 pixels

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

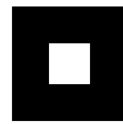
8 pixels

2 pixels lost to boundary (1 on each side)

How to prevent?

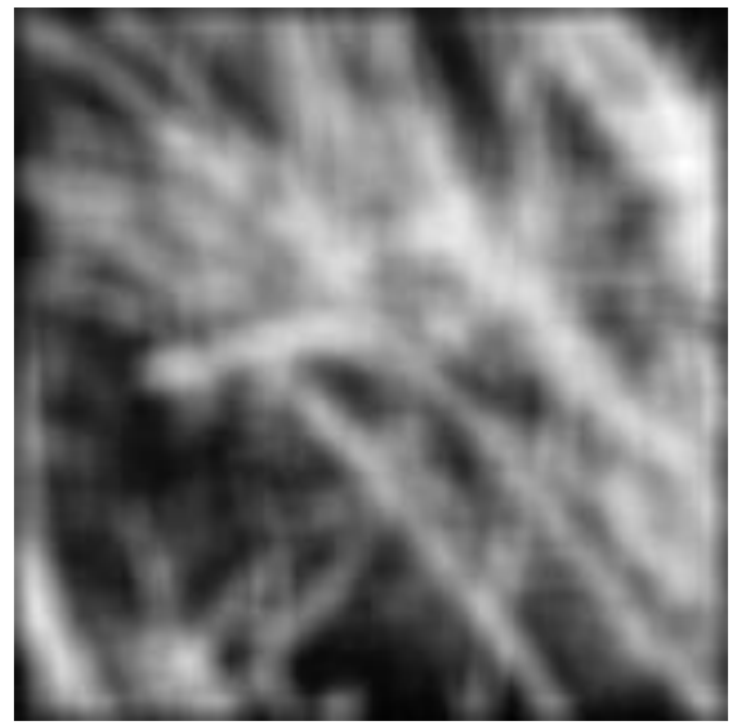
Source: S. Seitz

Smoothing by averaging



depicts box filter:
white = high value, black = low value

original

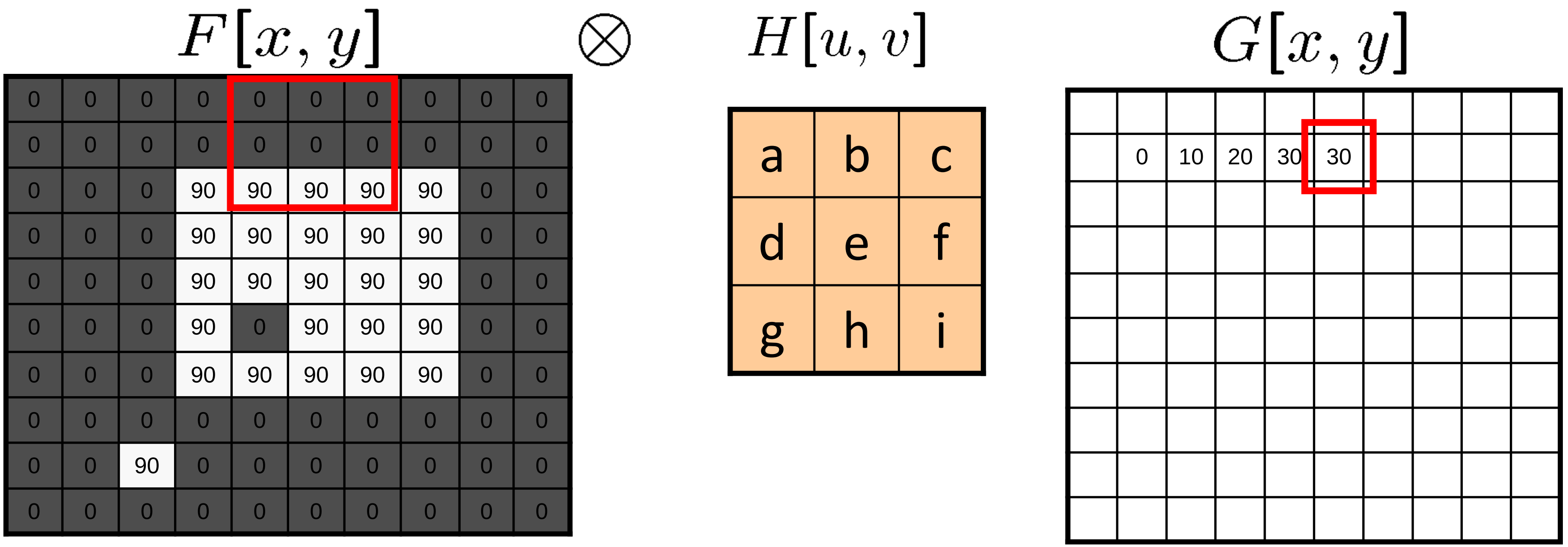


filtered

Averaging kernel

- Lets generalize to a kernel:

multiply weights per pixel and sum



- Which values in kernel *H* for the moving average example?

Averaging kernel

- Lets generalize to a kernel:

multiply weights per pixel and sum

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\otimes$

$H[u, v]$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

$G[x, y]$

	0	10	20	30	30				

- Which values in kernel *H* for the moving average example?

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Attribute uniform weight to each pixel *Loop over all pixels in neighborhood around image pixel F[i,j]*

Why 1/(2*k*+1)?

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Non-uniform weights

What happened to the 1/(2*k*+1)?

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" $H[u,v]$ is the prescription for the weights in the linear combination.

Questions?

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

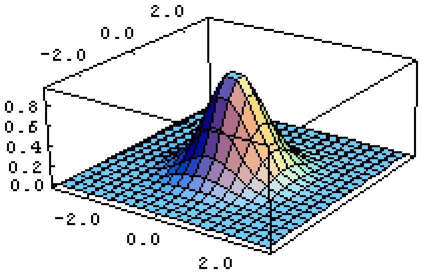
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

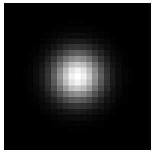
This kernel is an approximation of a Gaussian function:

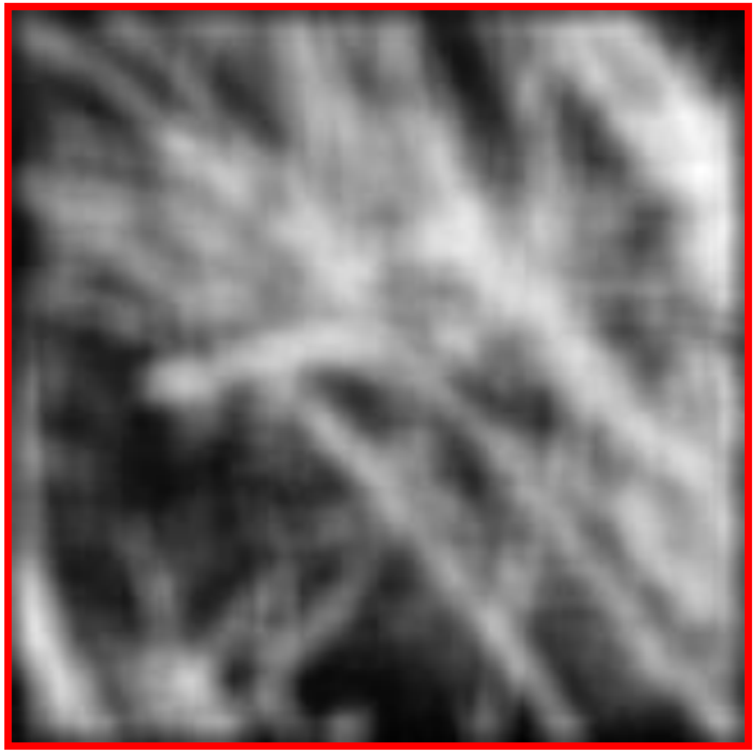
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Source: S. Seitz

Smoothing with a Gaussian

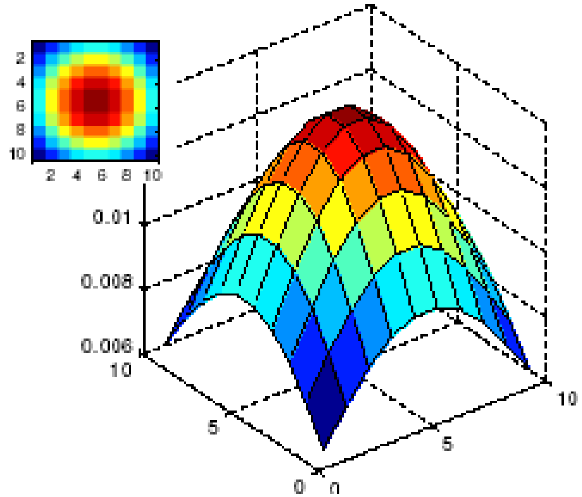




The Gaussian is the only function that does not introduce artifacts [koenderink, “The structure of images”, 1984]

Gaussian filters

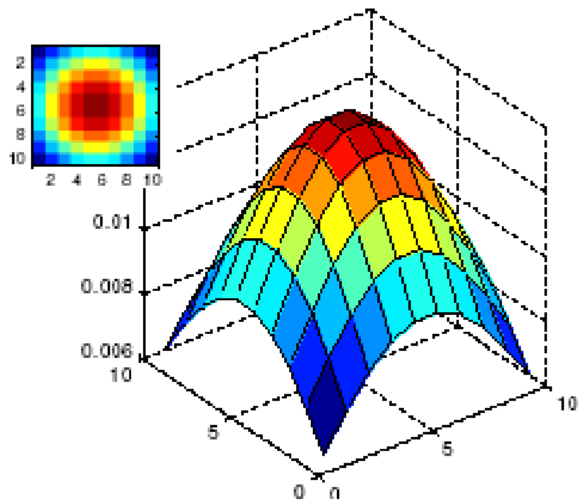
- What parameters matter here? $h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



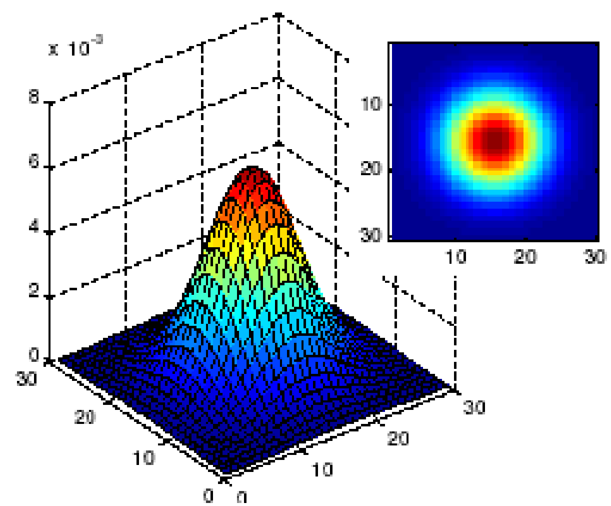
σ = 5 with 10 x 10 kernel

Gaussian filters

- What parameters matter here? $$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



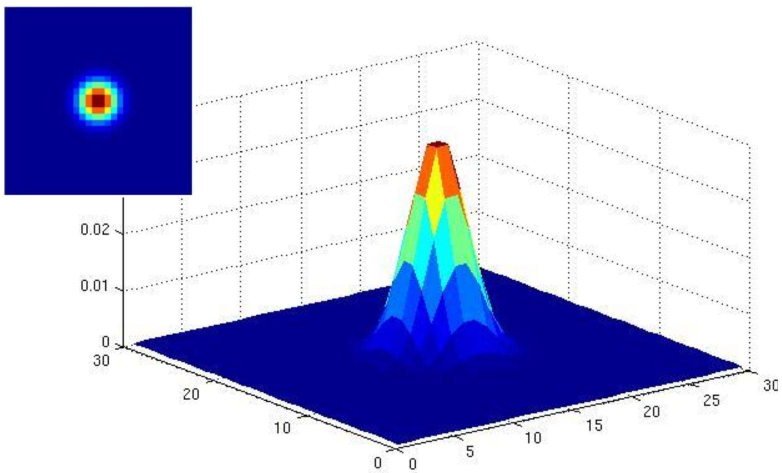
σ = 5 with 10 x 10 kernel



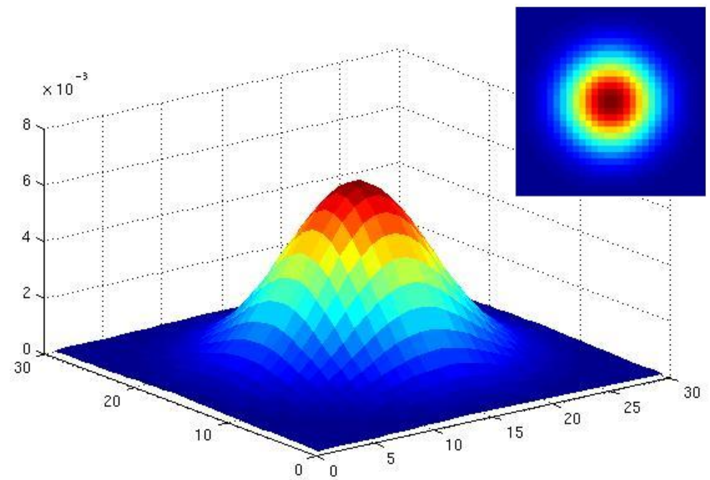
σ = 5 with 30 x 30 kernel

Gaussian filters

- **Variance** of Gaussian: determines extent of smoothing



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

- Rule of thumb: set extent to 3 σ

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Shifted left by 1 pixel with correlation

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

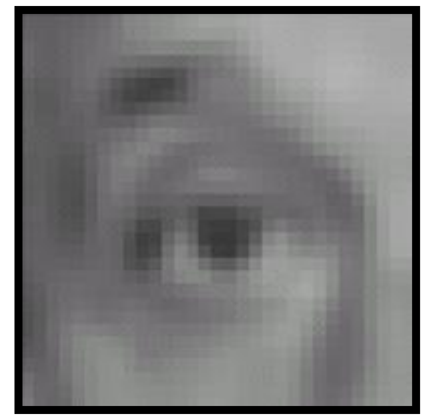
?

Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

Source: D. Lowe

Convolution

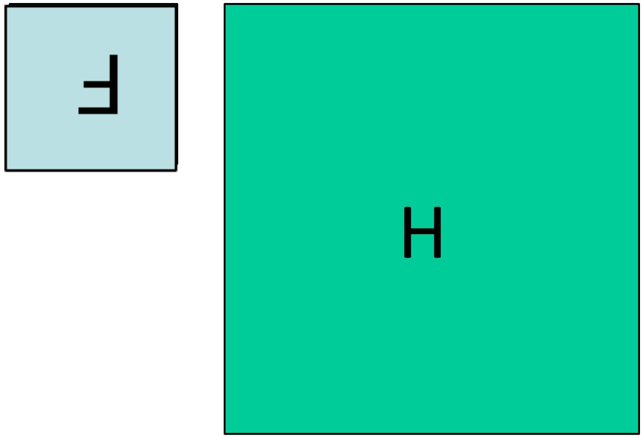
- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

How different from cross-correlation?

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

$$G = H \star F$$

Notation for convolution operator



Filtering an impulse signal

What is the result of convolving the impulse signal (image) *F* with the arbitrary kernel *H*?

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$F[x, y]$

$\star$

a	b	c
d	e	f
g	h	i

$H[u, v]$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	a	b	c	0	0
0	0	d	e	f	0	0
0	0	g	h	i	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

$G[x, y]$

Convolution vs. correlation

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

$$G = H \otimes F$$

For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Properties of convolution

- Linear & shift invariant
- Commutative:

  $f * g = g * f$

- Associative

  $(f * g) * h = f * (g * h)$

- Identity:

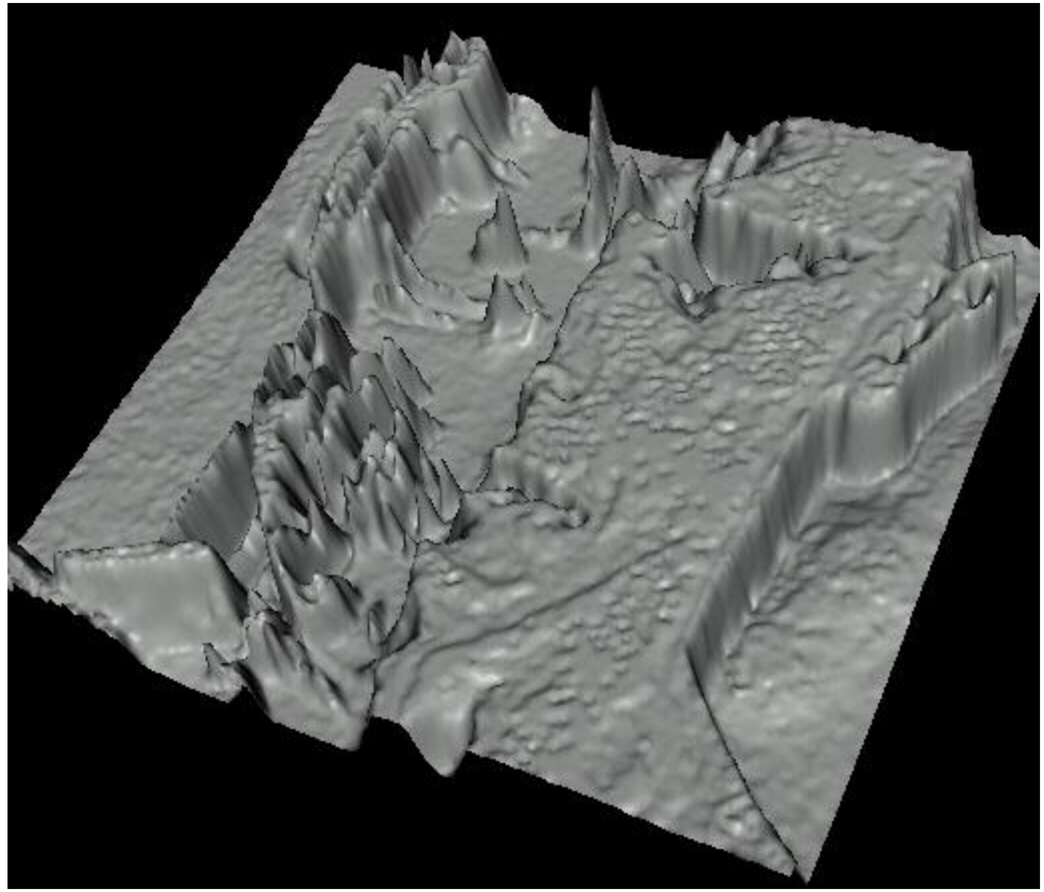
  unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$. $f * e = f$

- Differentiation:

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

Questions?

Images are functions



- Edges look like steep cliffs

Source: S. Seitz

Images as functions

- We can think of an image as a function, *f*, from R^2 to R:
  - $f(x, y)$ gives the intensity at position (x, y)
  - How does reality differ from this definition?
    - Only defined over a rectangle, with finite range:

$$f: [a,b] \times [c,d] \rightarrow [0, 1.0]$$

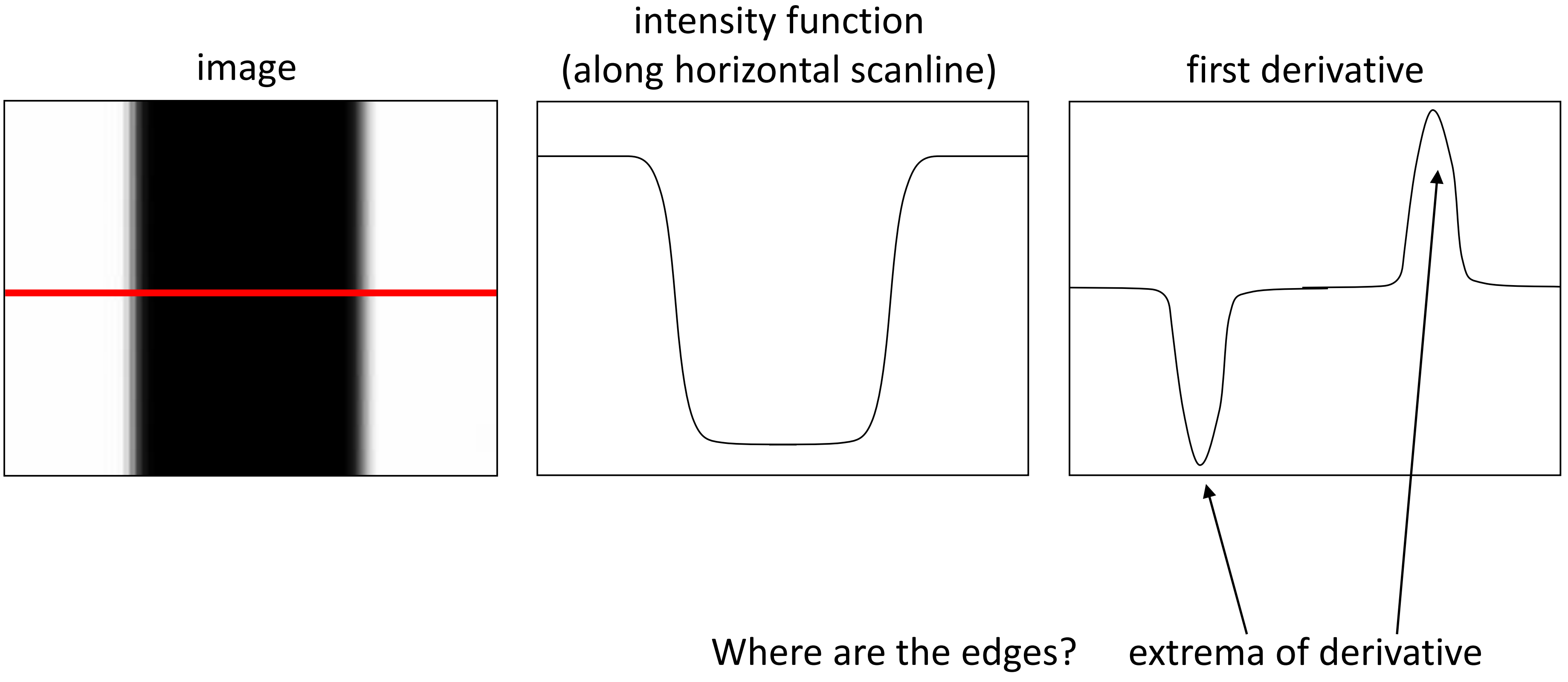
- How about color?
  - "Vector valued" function

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Source: S. Seitz

Derivatives and edges

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

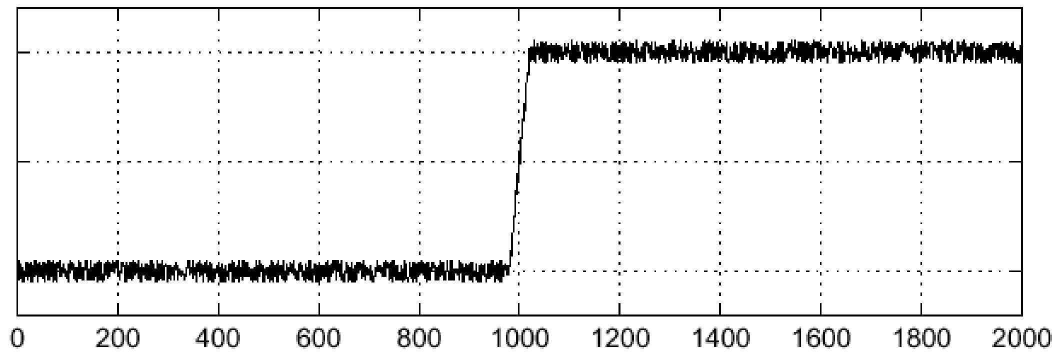


Source: L. Lazebnik

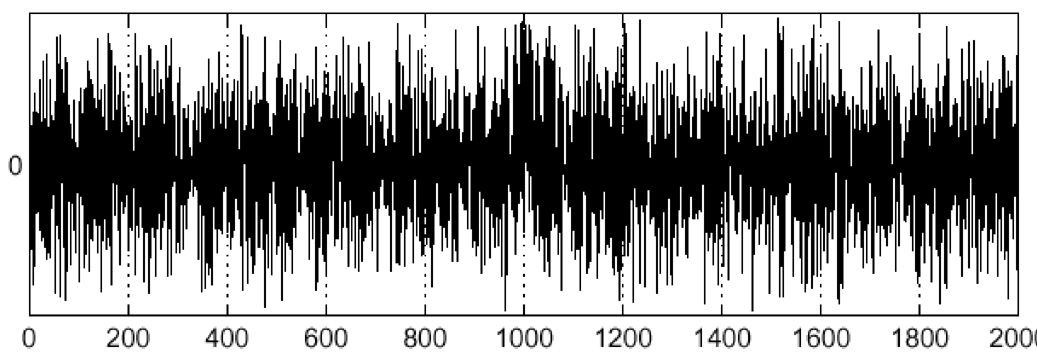
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



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



Where is the edge?

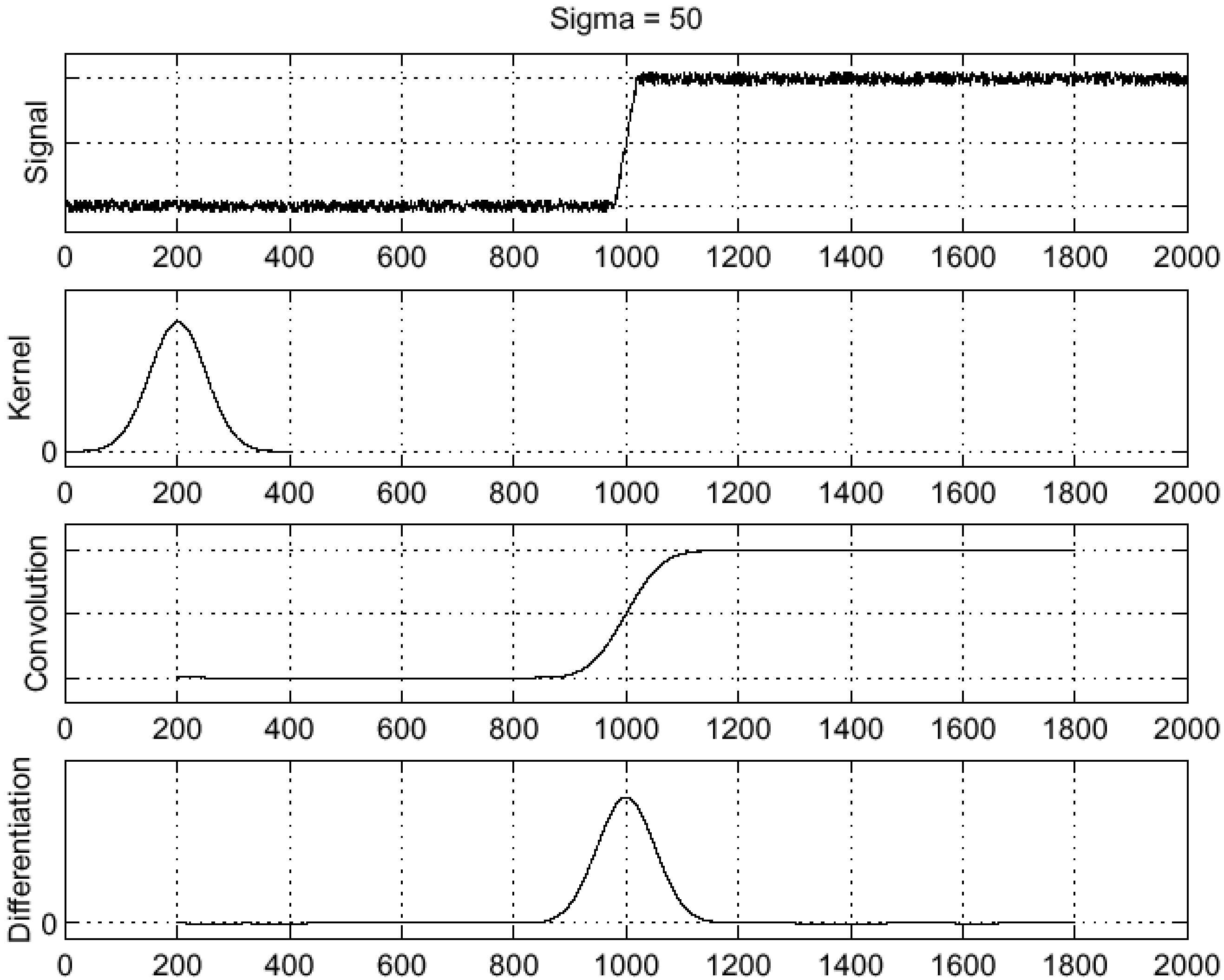
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

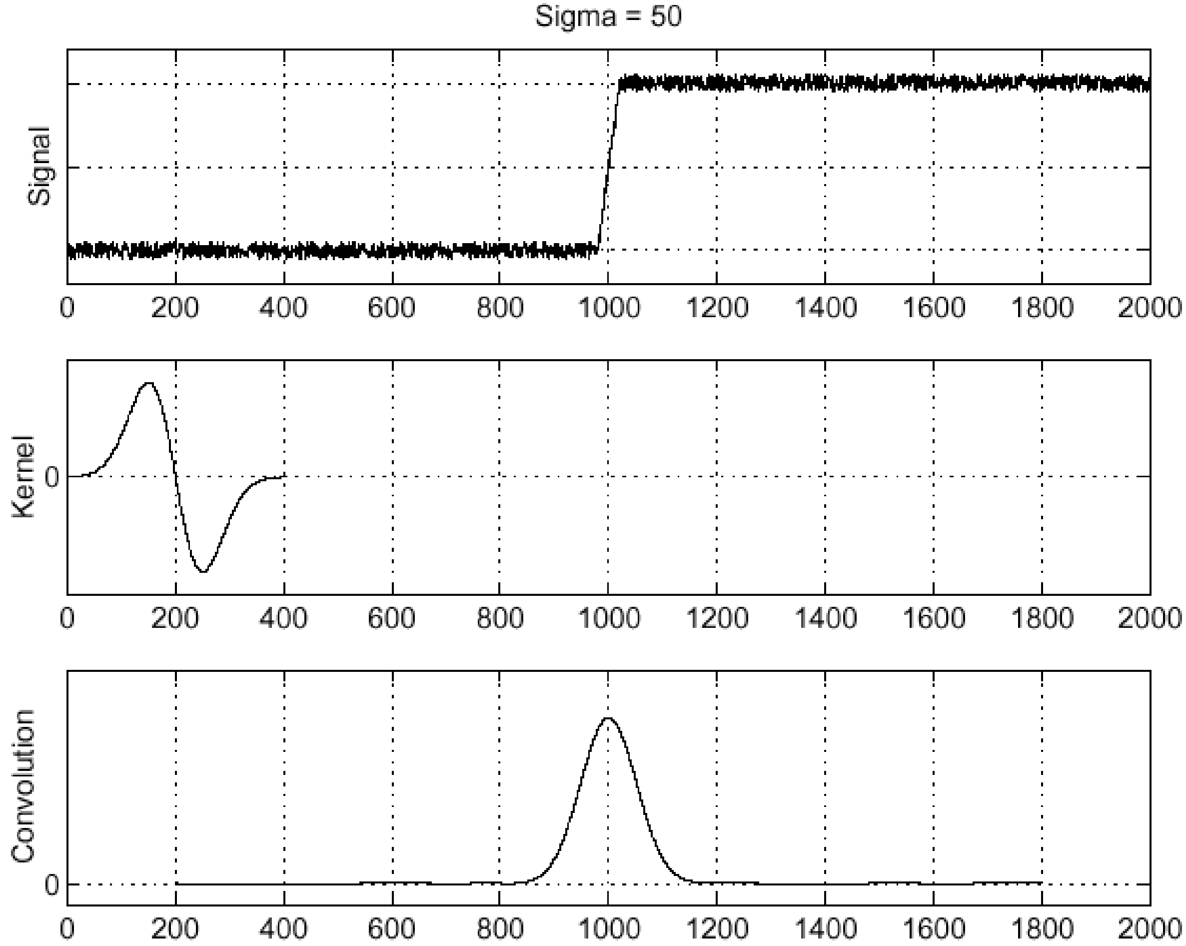
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

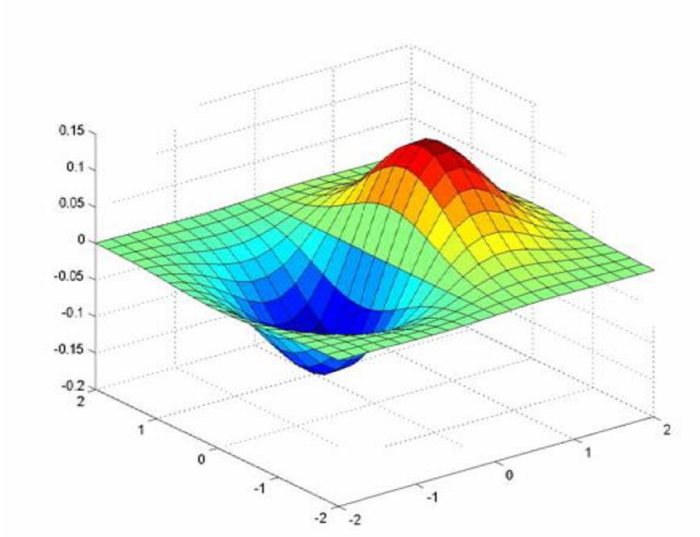
f

$\frac{\partial}{\partial x}h$

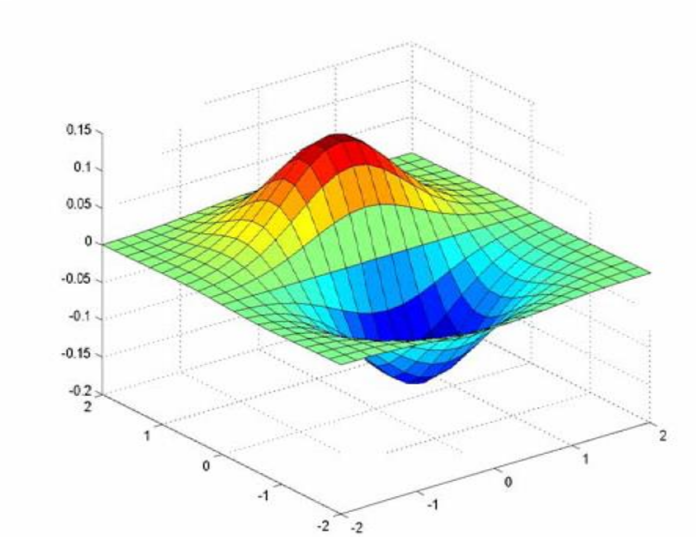
$(\frac{\partial}{\partial x}h) \star f$



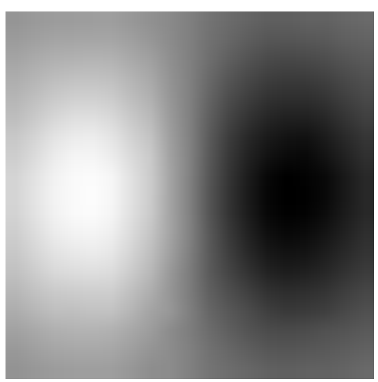
Derivative of Gaussian filters

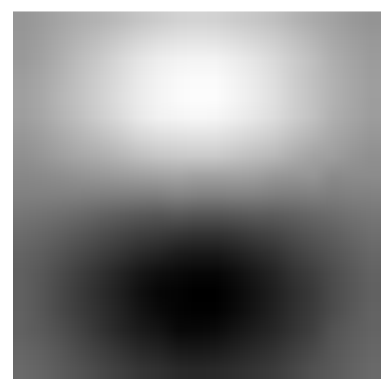


x-direction



y-direction





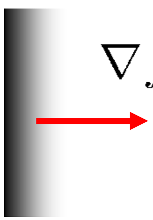
Source: L. Lazebnik

Image gradient

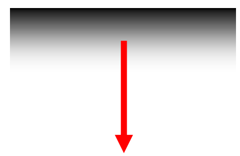
The gradient of an image:

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

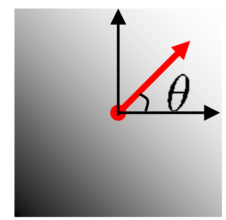
The gradient points in the direction of most rapid change in intensity



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



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



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

The gradient direction (orientation of edge normal θ) 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

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

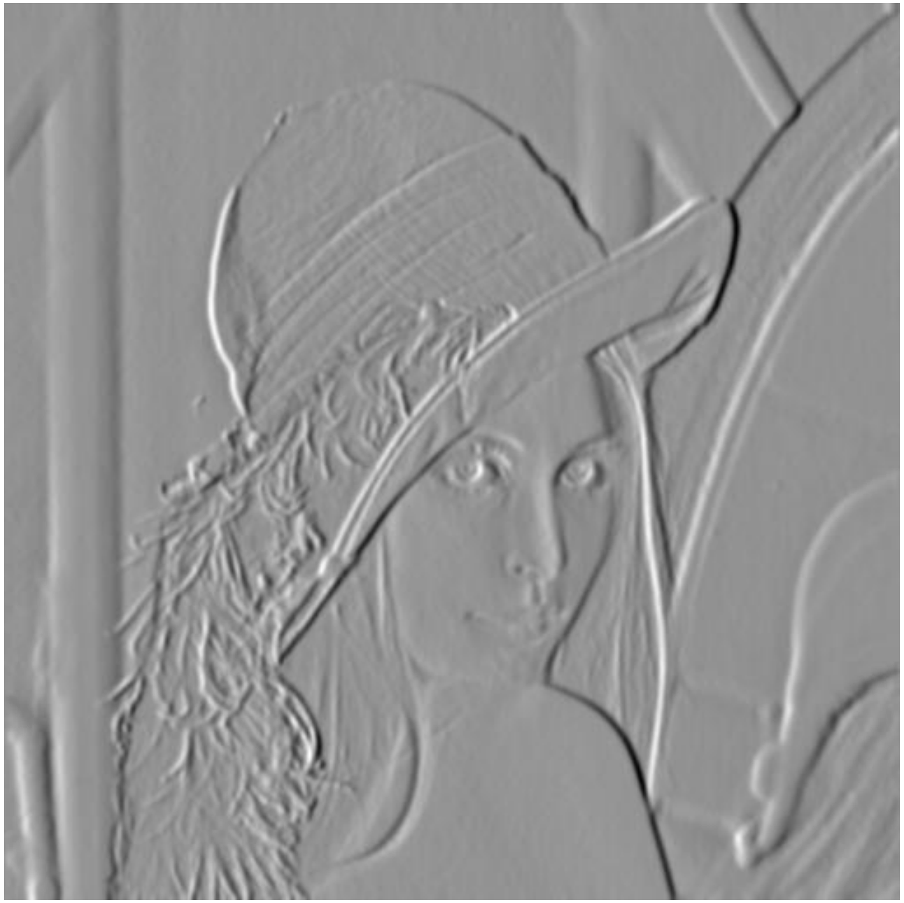
Slide credit S. Seitz

Example

original image (Lena)

Compute Gradients

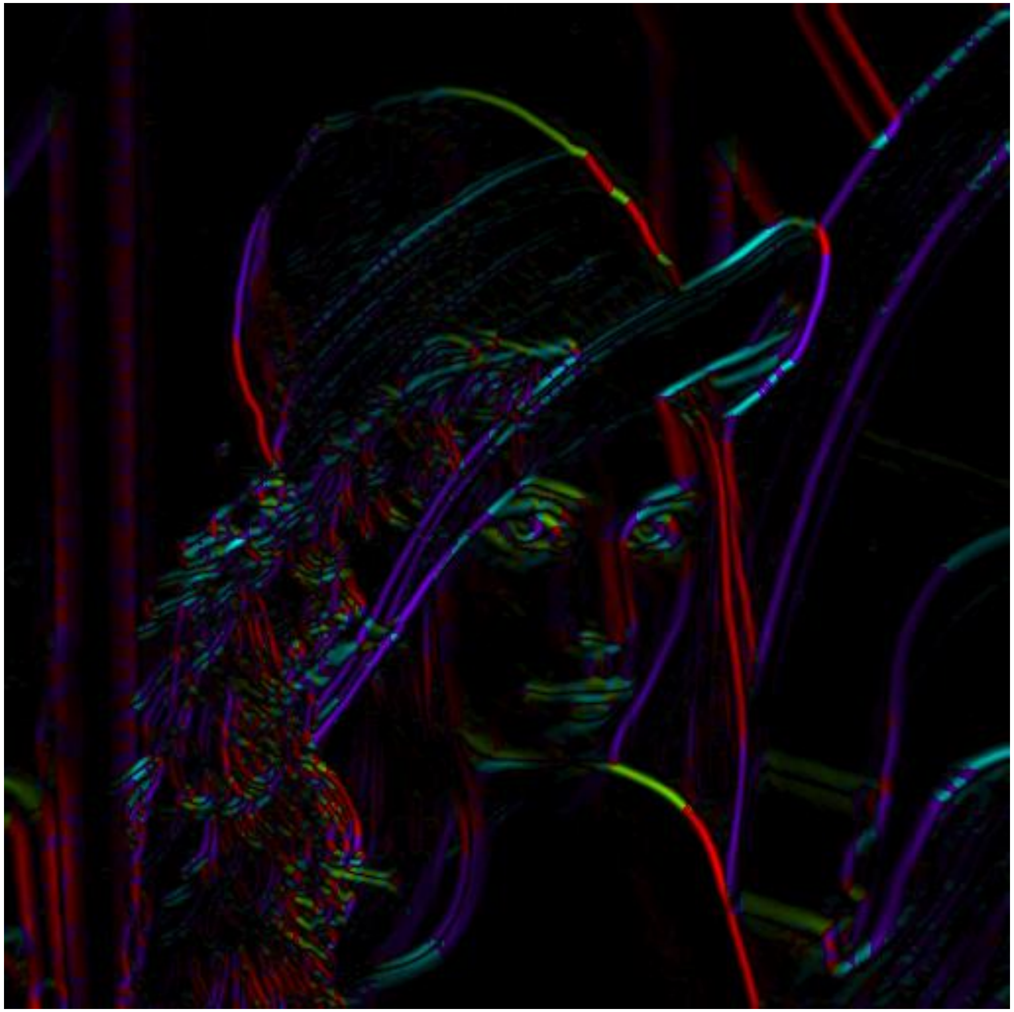
Derivative of Gaussian



Derivative of Gaussian

- Which one is the gradient in the x-direction (resp. y-direction)?

Gradient Orientation



Orientation of the gradient

theta = atan2(gy, gx)

Gradient magnitude

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

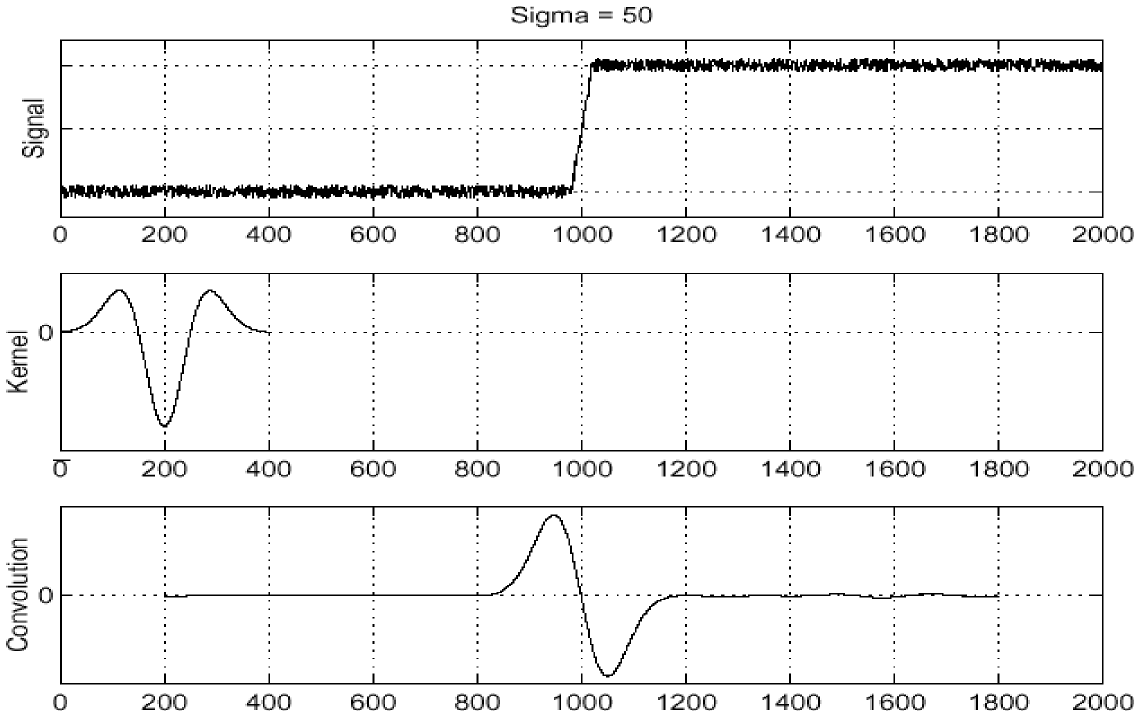
- Do you see a problem here?

Edge detection, Take 2

$$f$$

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

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



Edge

Second derivative of Gaussian (Laplacian)

Edge = zero crossing of second derivative

Source: S. Seitz

Questions?

- Next topic: interest point detection

Applications

- Feature points (blobs) are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition

Edge detection, Take 2

f

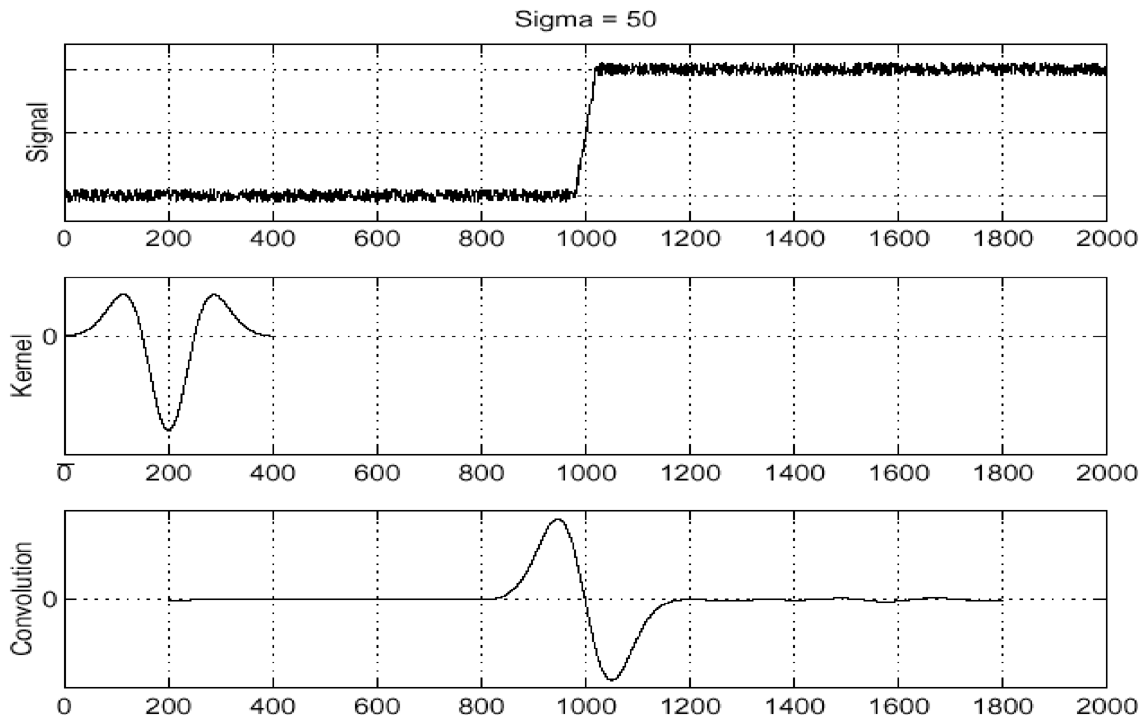
Edge

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

Second derivative of Gaussian (Laplacian)

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

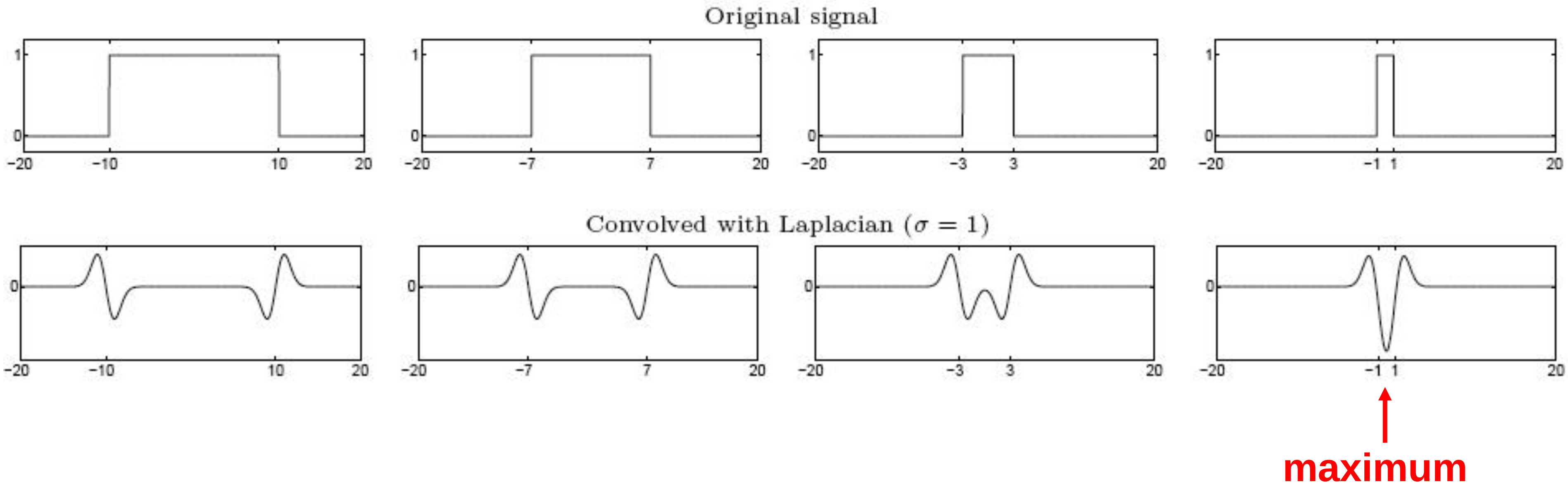
Edge = zero crossing of second derivative



Source: S. Seitz

From edges to blobs

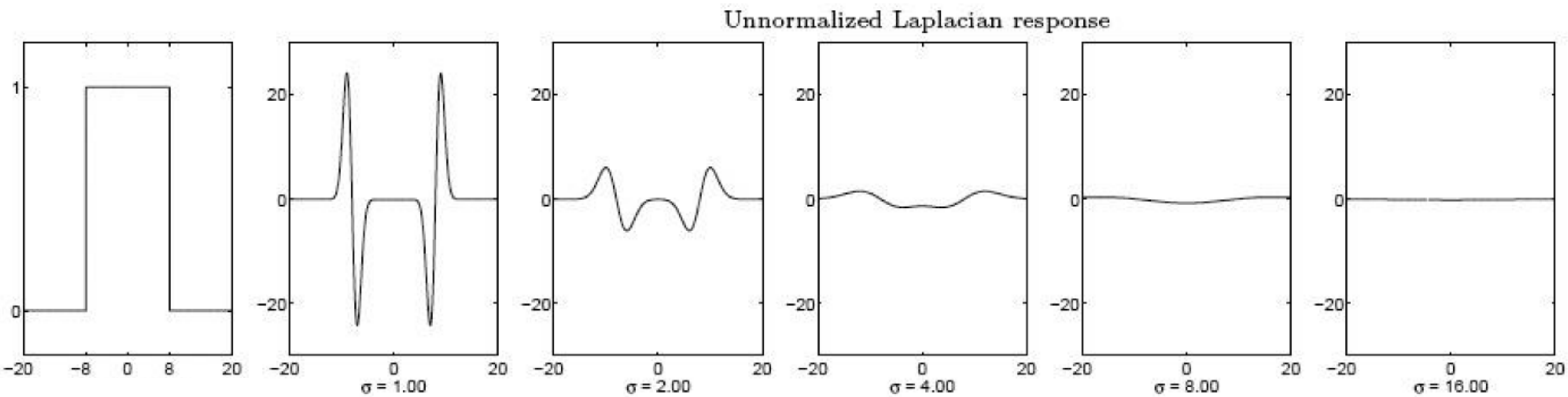
- Edge = ripple
- Blob = superposition of two ripples



Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the <u>*scale*</u> 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:



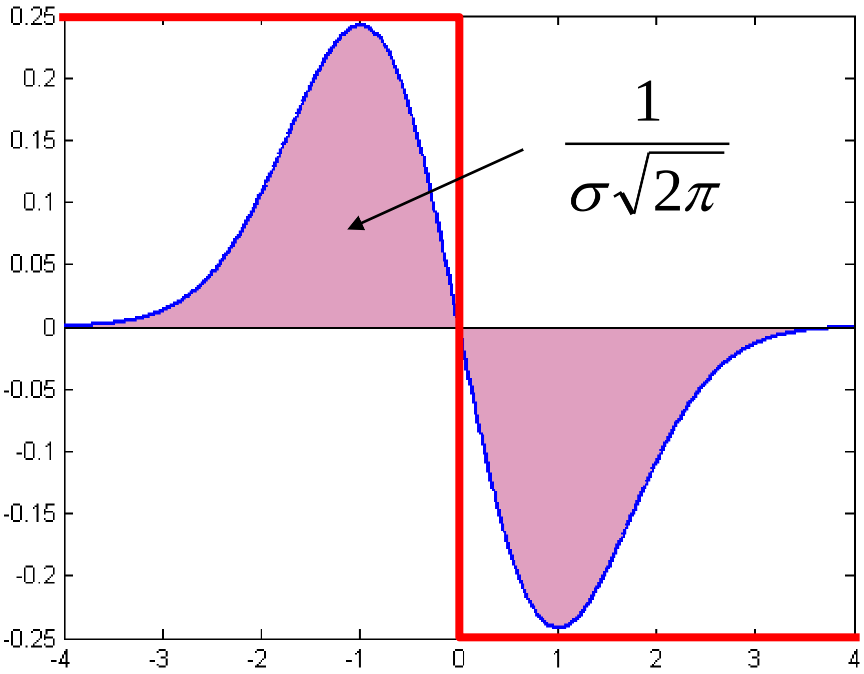
original signal (radius=8)

increasing σ

Why does this happen?

Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases



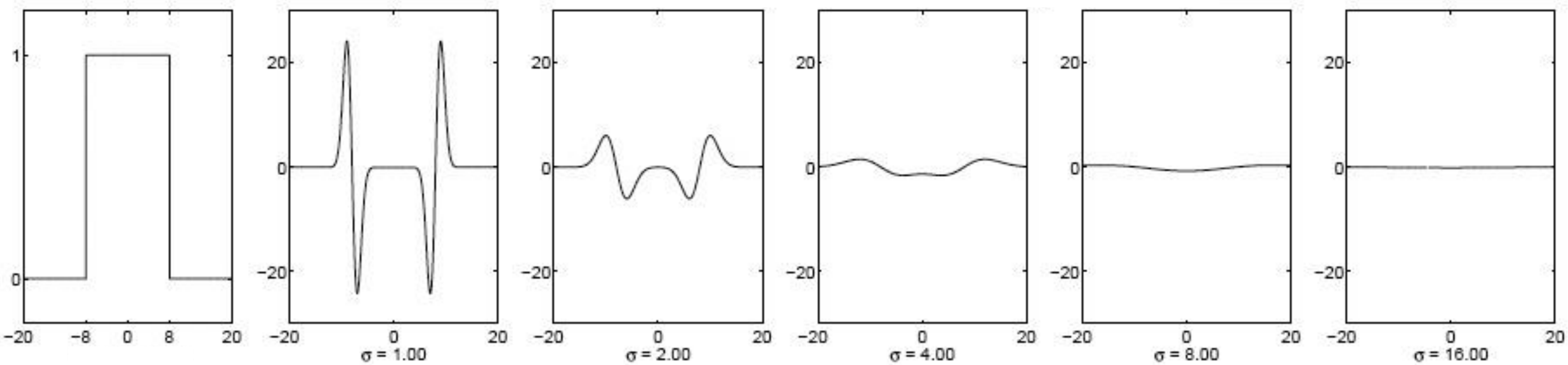
Scale normalization

- 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

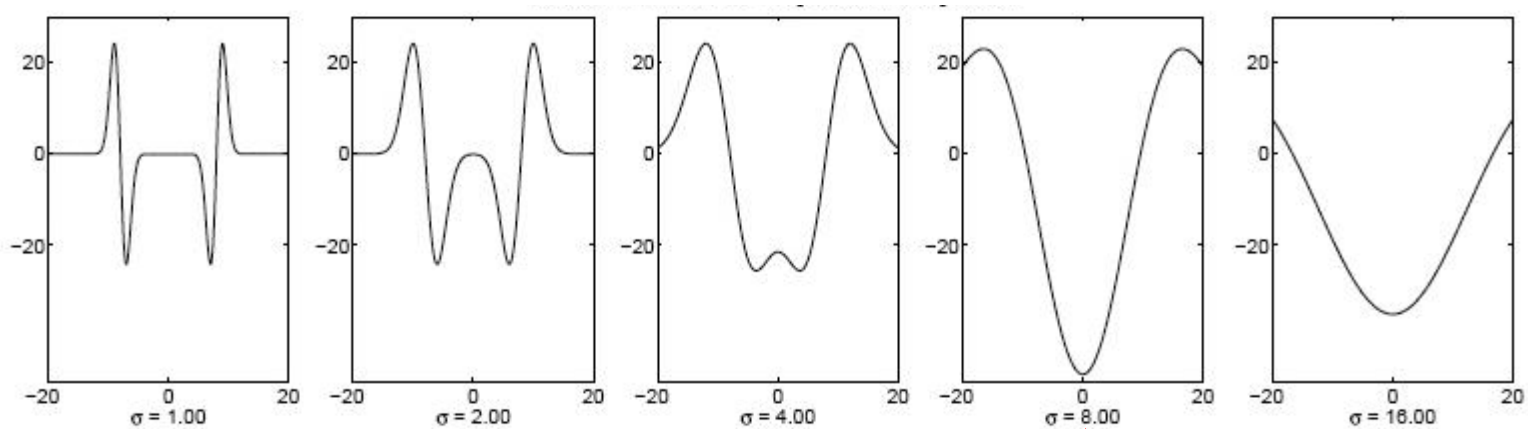
Effect of scale normalization

Original signal

Unnormalized Laplacian response



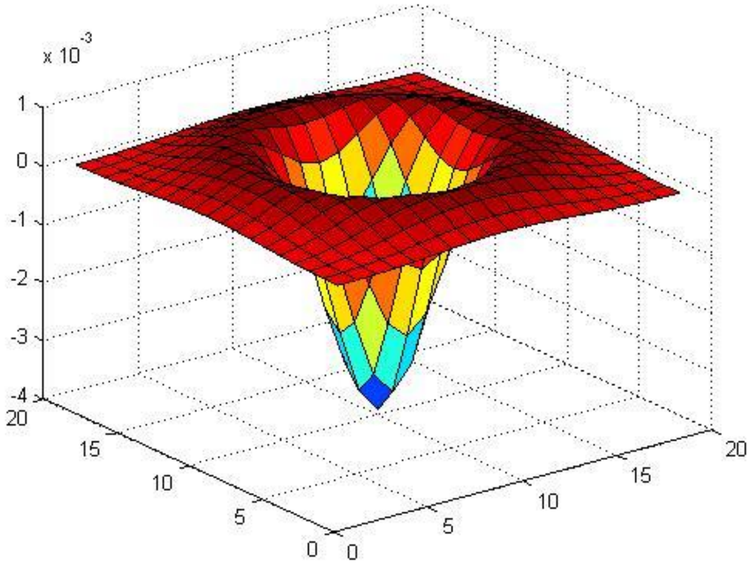
Scale-normalized Laplacian response

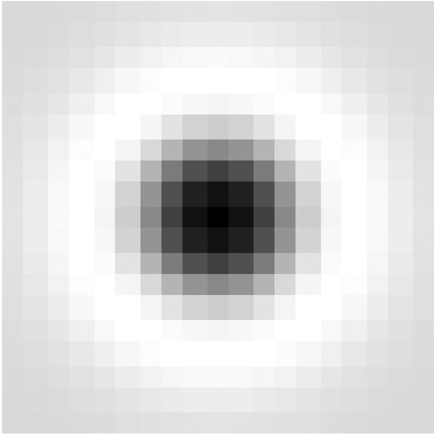


maximum

Blob detection in 2D

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

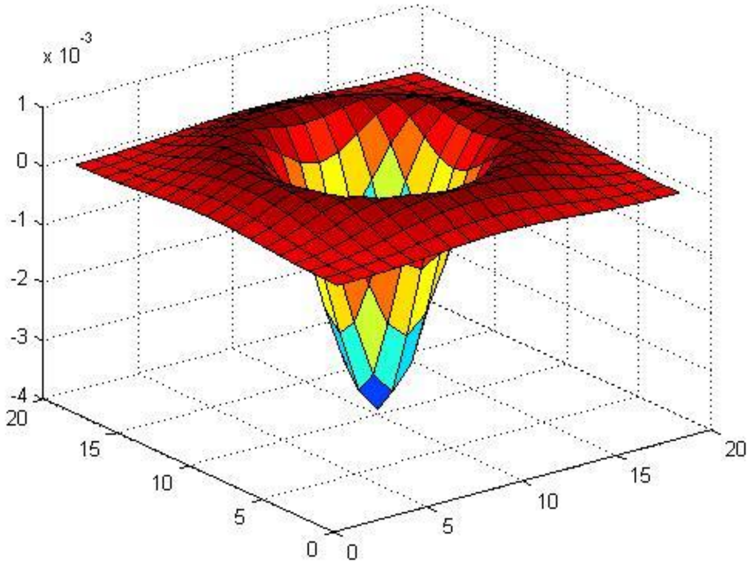


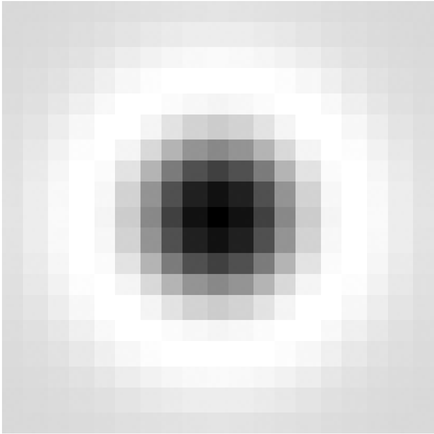


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Blob detection in 2D

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

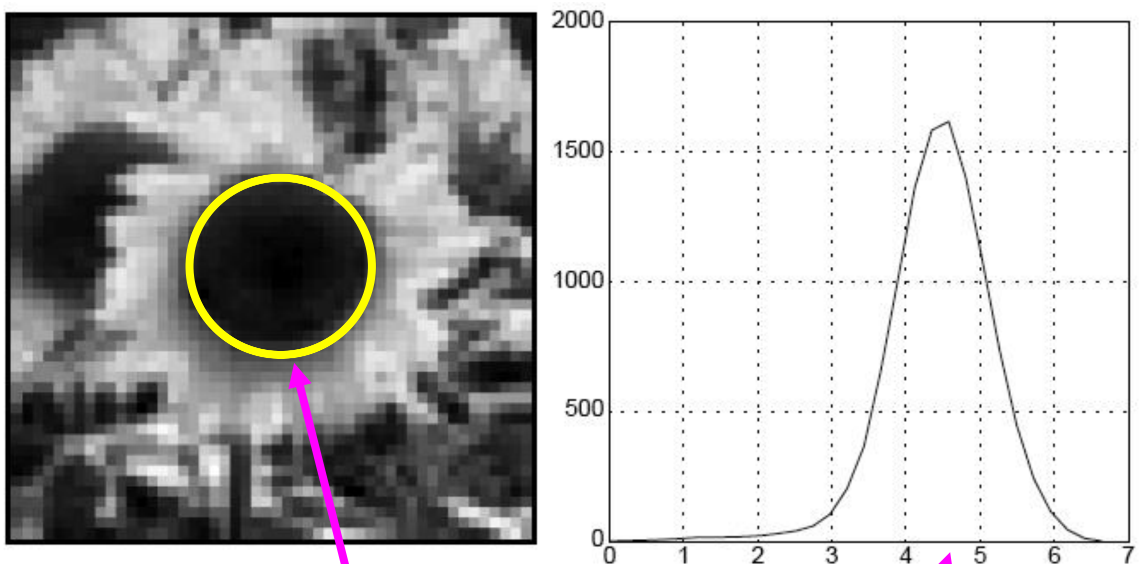




Scale-normalized: $$\nabla^2_{\text{norm}} g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

Characteristic scale

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

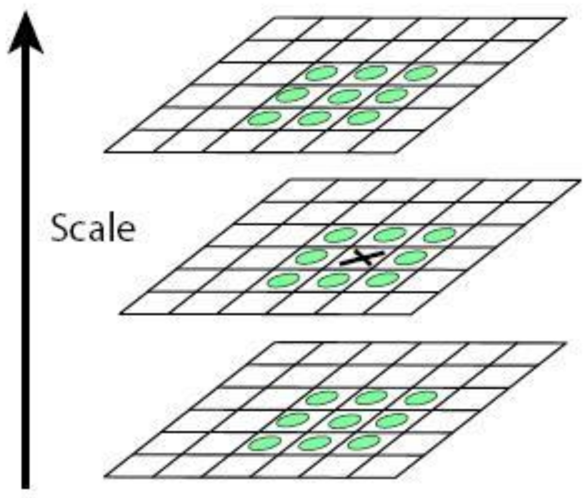


characteristic scale

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
2. Find maxima of squared Laplacian response in scale-space

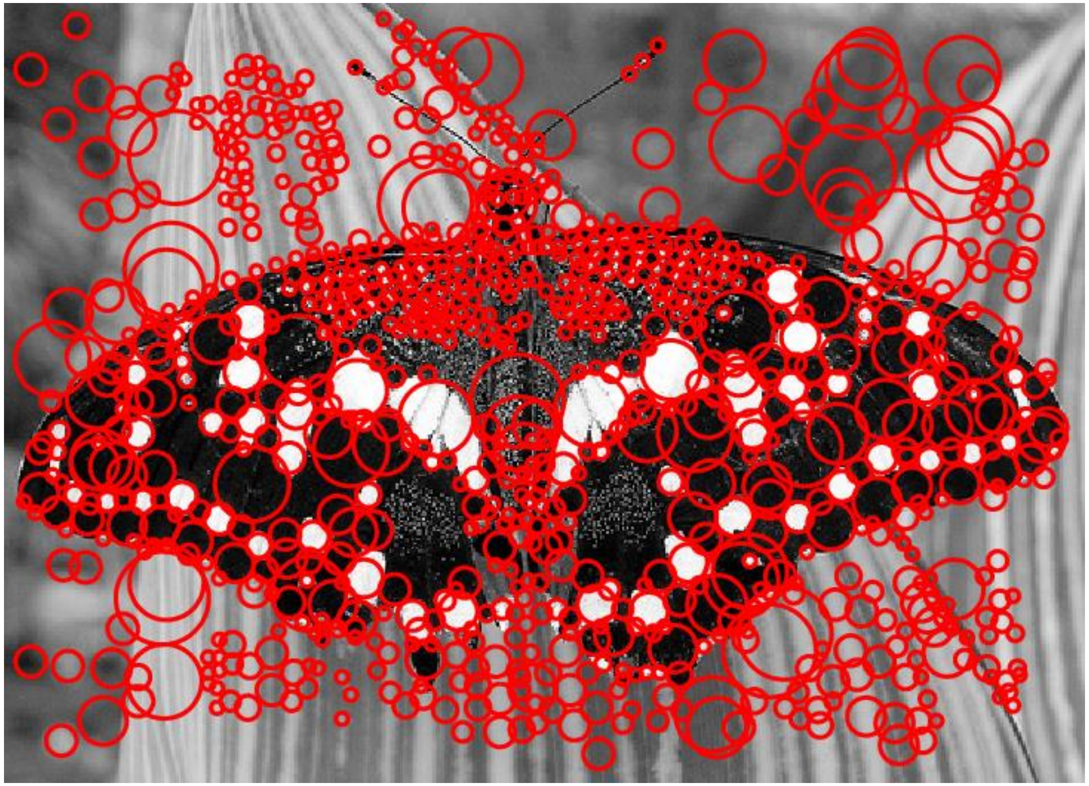


Scale-space blob detector: Example

Scale-space blob detector: Example

sigma = 11.9912

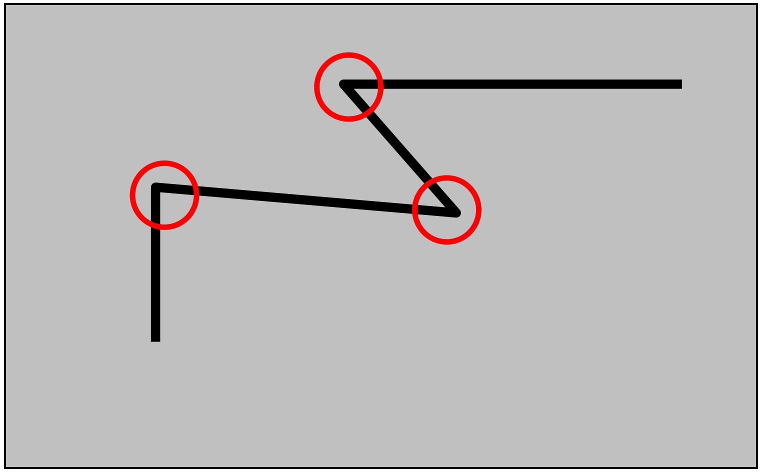
Scale-space blob detector: Example



Questions?

A feature detector example:

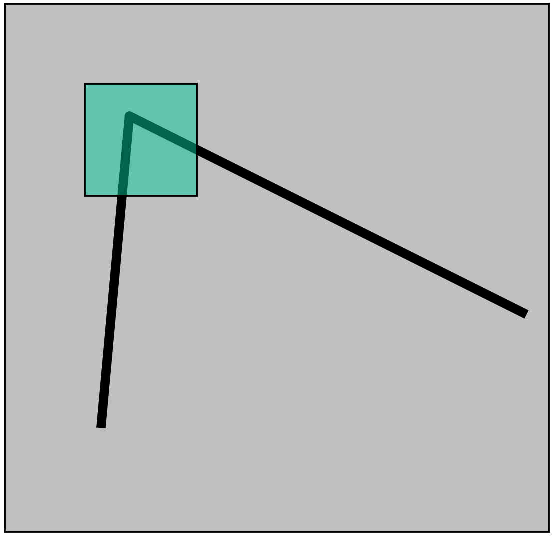
Harris corner detector



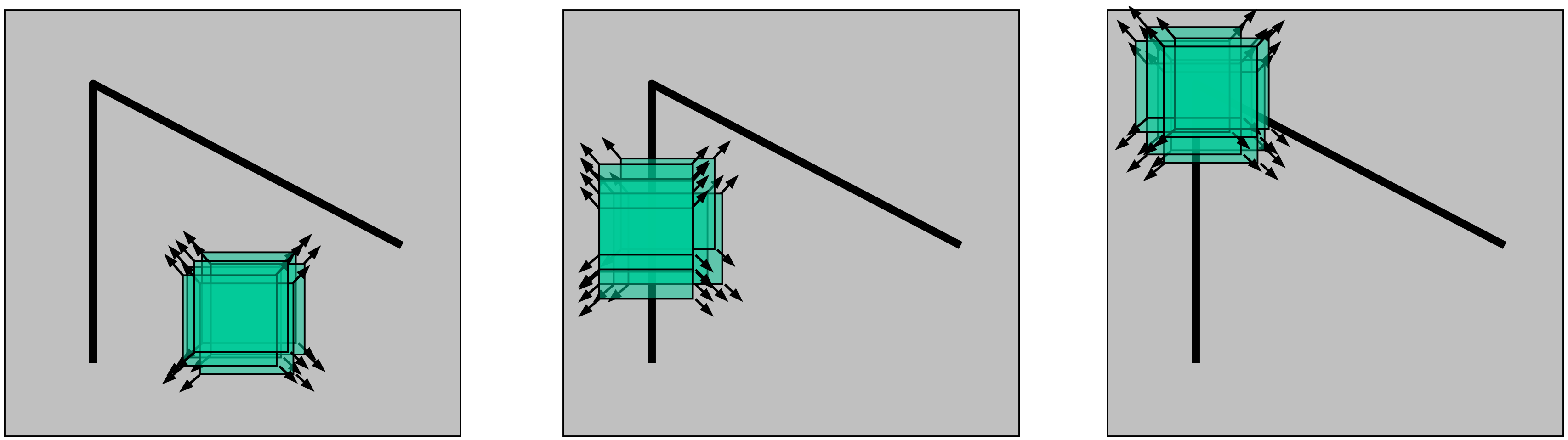
C.Harris, M.Stephens. “A Combined Corner and Edge Detector”. 1988

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

Harris Detector: Mathematics

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

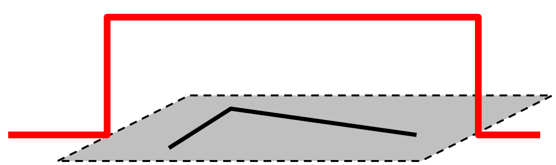
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

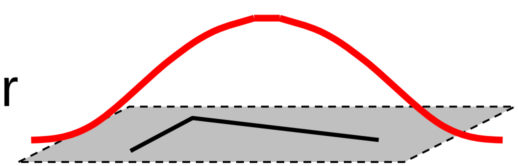
Shifted intensity

Intensity

Window function $w(x,y)$ =



or



1 in window, 0 outside

Gaussian

Taylor series approx to shifted image

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

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

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

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

Harris Detector: Mathematics

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

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

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"
or "second moment matrix"

Harris Detector: Mathematics

Intensity change in shifting window:

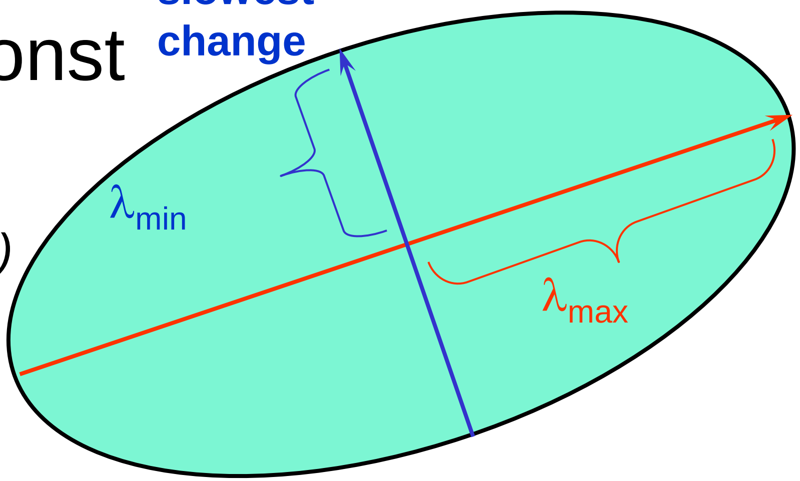
$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

Where is the largest change?

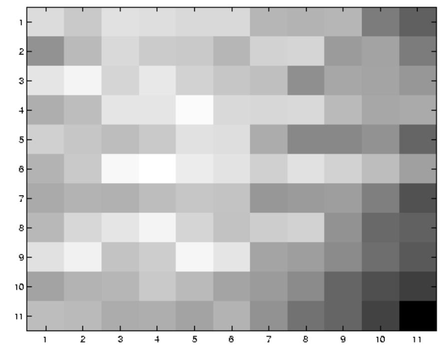
$\lambda_1, \lambda_2,$ eigenvalues of M

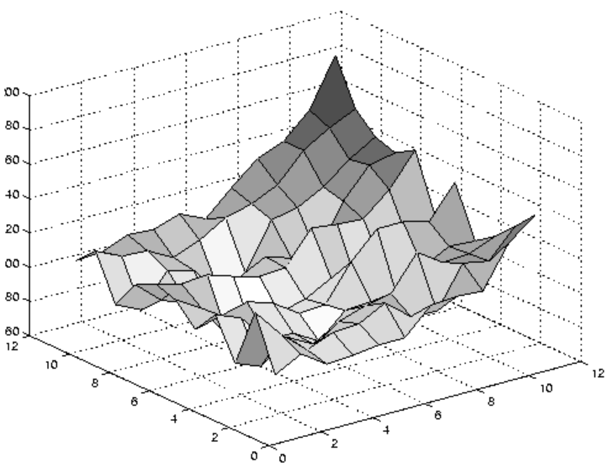
Ellipse $E(u,v)$ = const

Iso-intensity contour of $E(u,v)$



Selecting Good Features

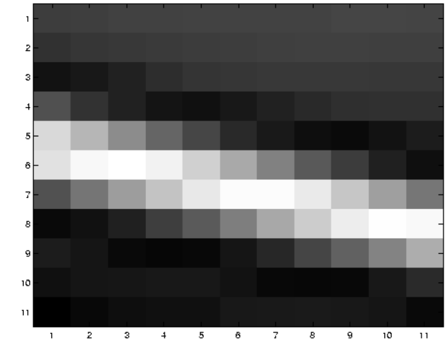


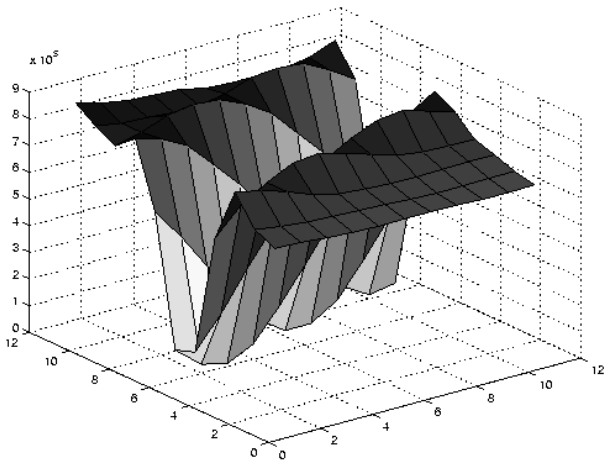


How large is each of the two eigenvalues?

small λ_1, small λ_2

Selecting Good Features

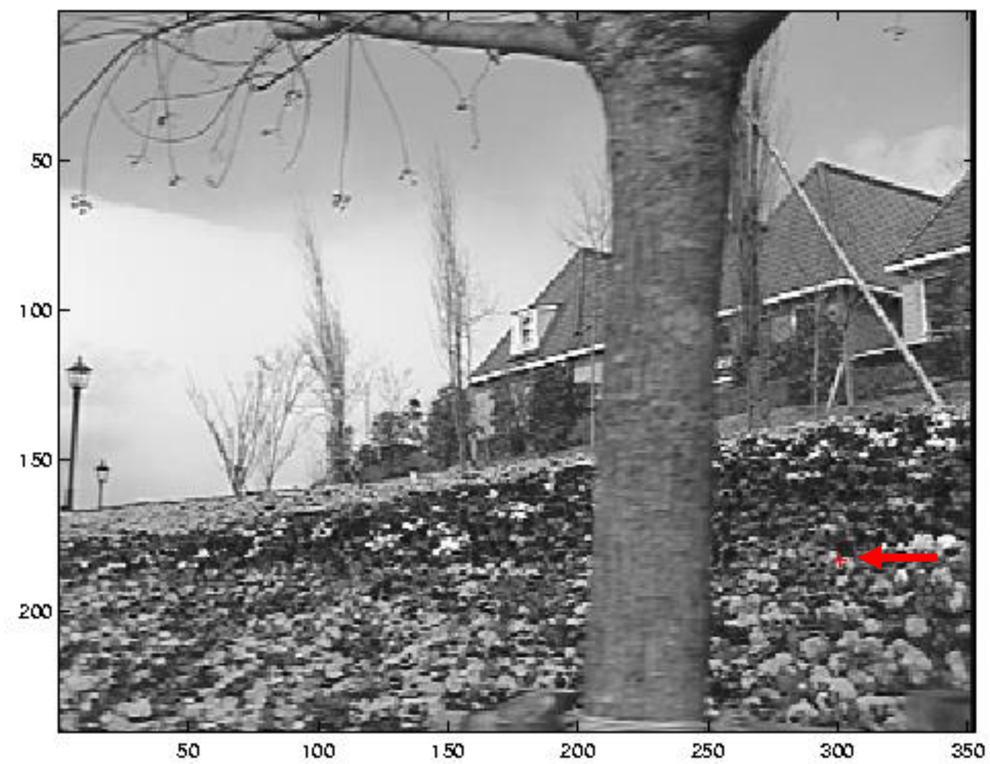


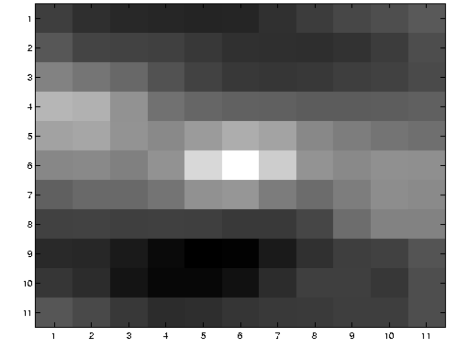


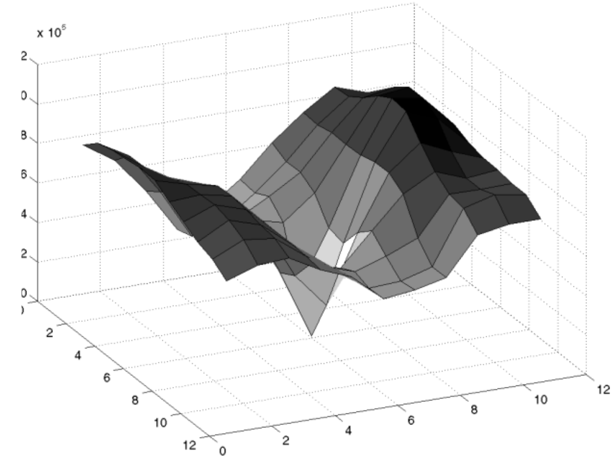
How large is each of the two eigenvalues?

large λ_1, small λ_2

Selecting Good Features







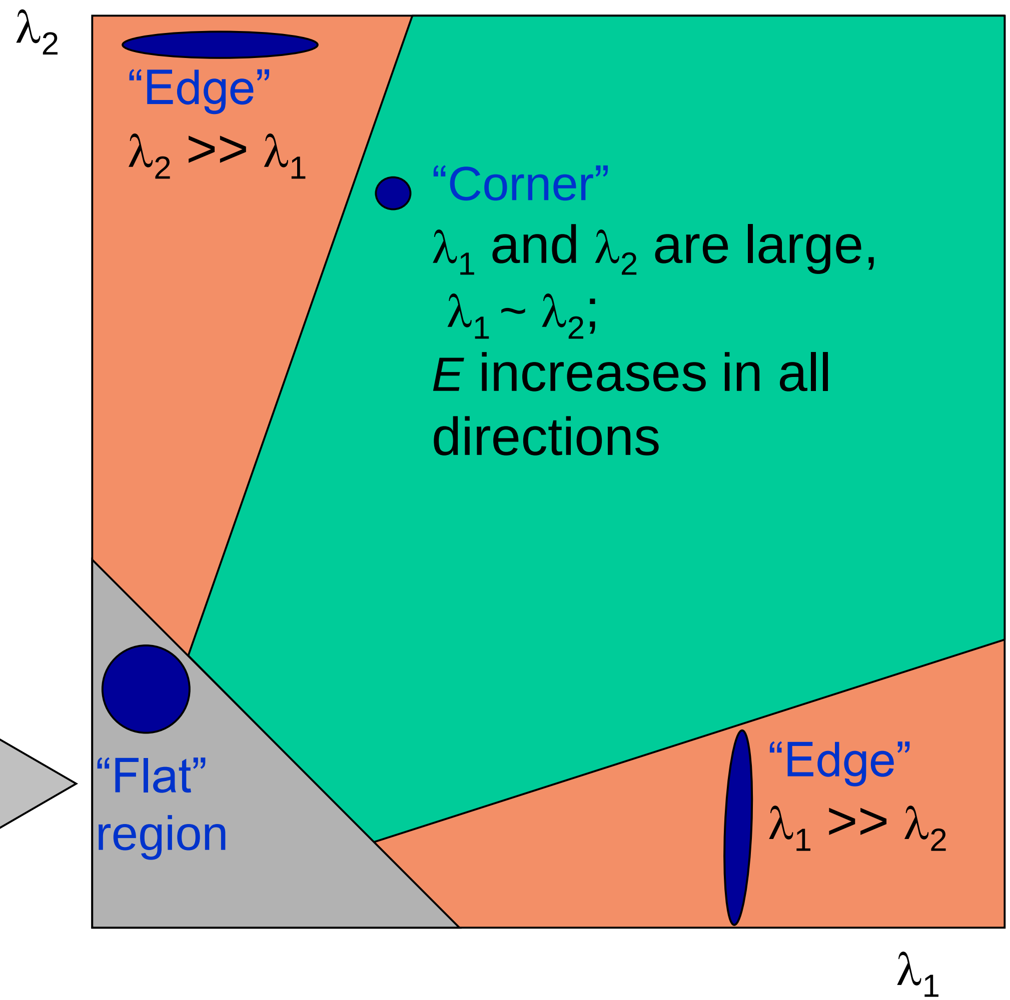
How large is each of the two eigenvalues?

λ_1 and λ_2 are large

Harris Detector: Mathematics

Classification of image points using eigenvalues of *M*:

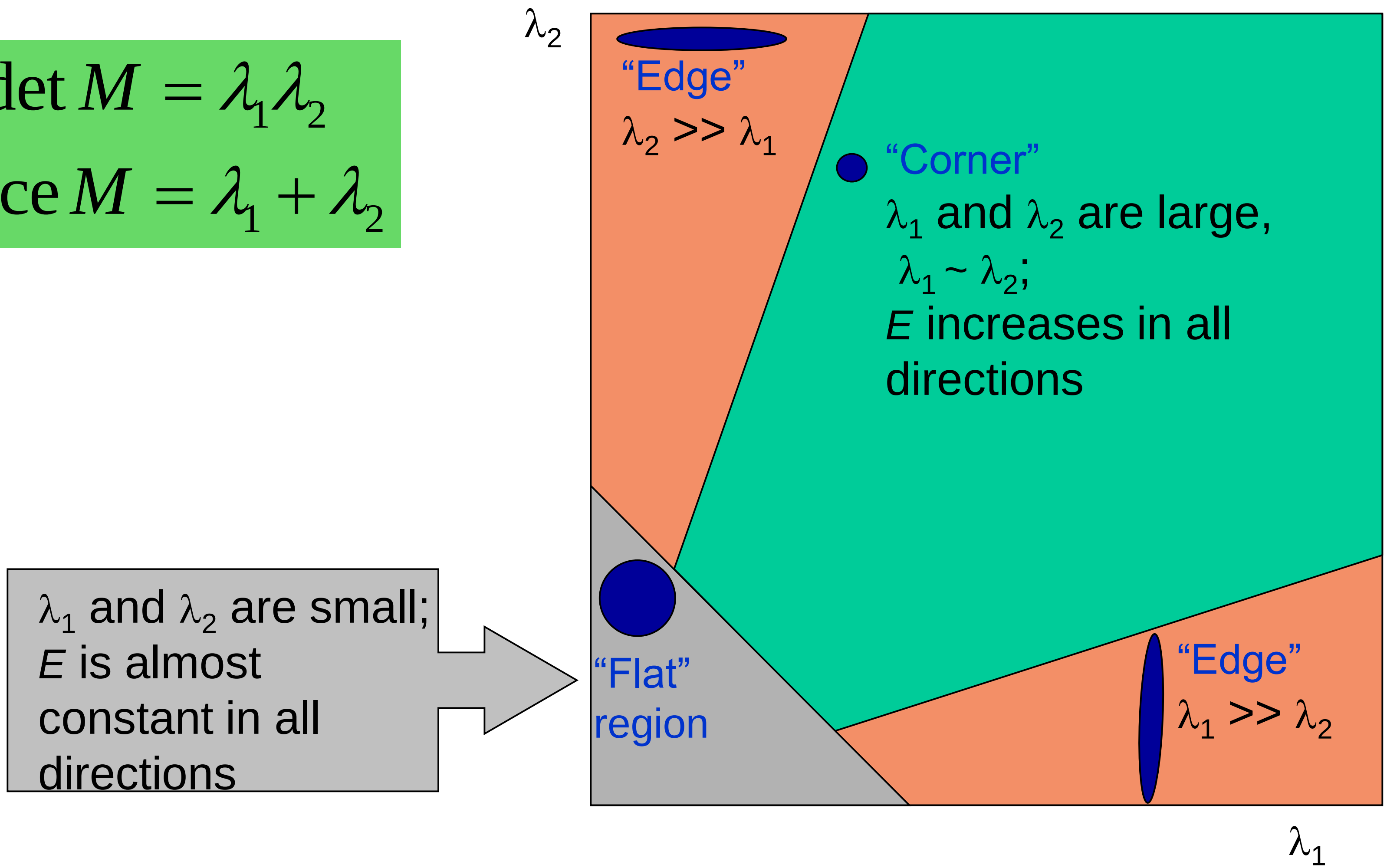
λ_1 and λ_2 are small; *E* is almost constant in all directions



Harris Detector: Mathematics

No need to compute eigenvalues:

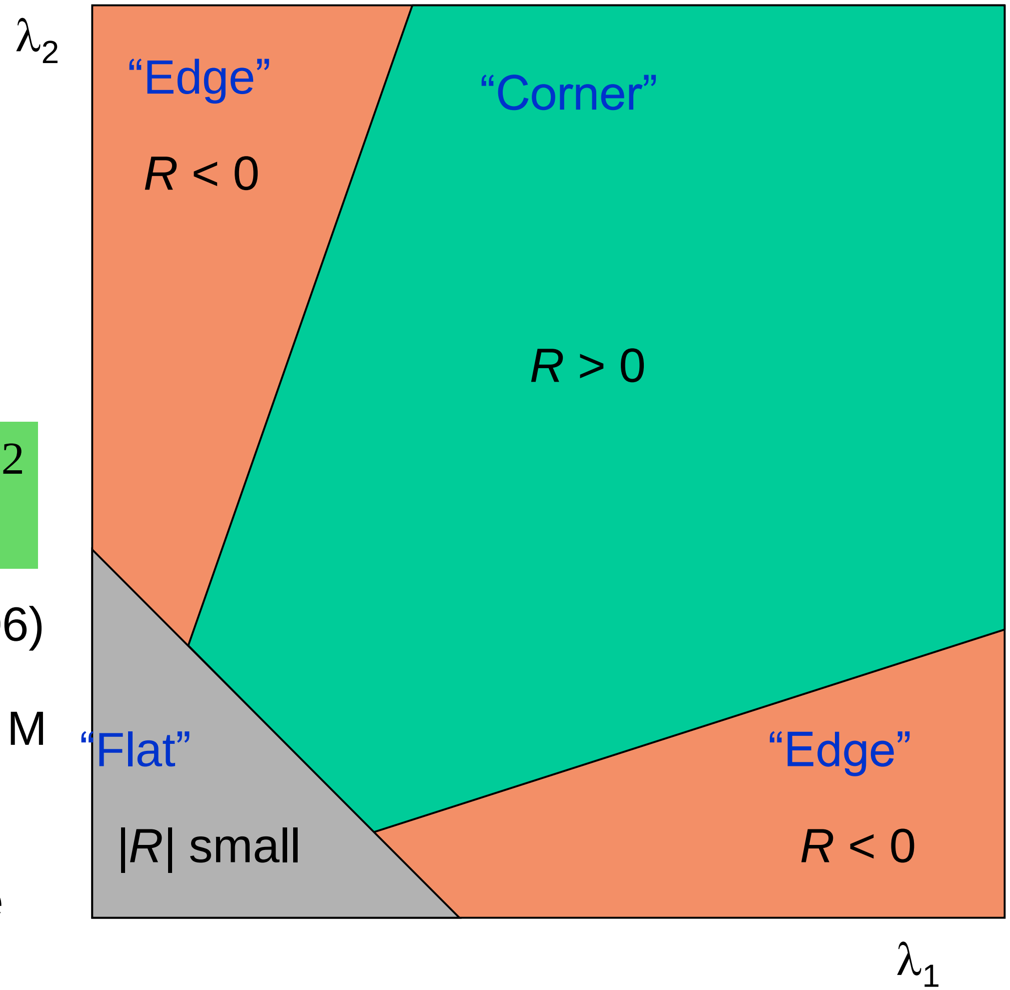
$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$



Harris Detector: Mathematics

No need to compute eigenvalues:

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$



Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

(k – empirical constant, k = 0.04-0.06)

- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

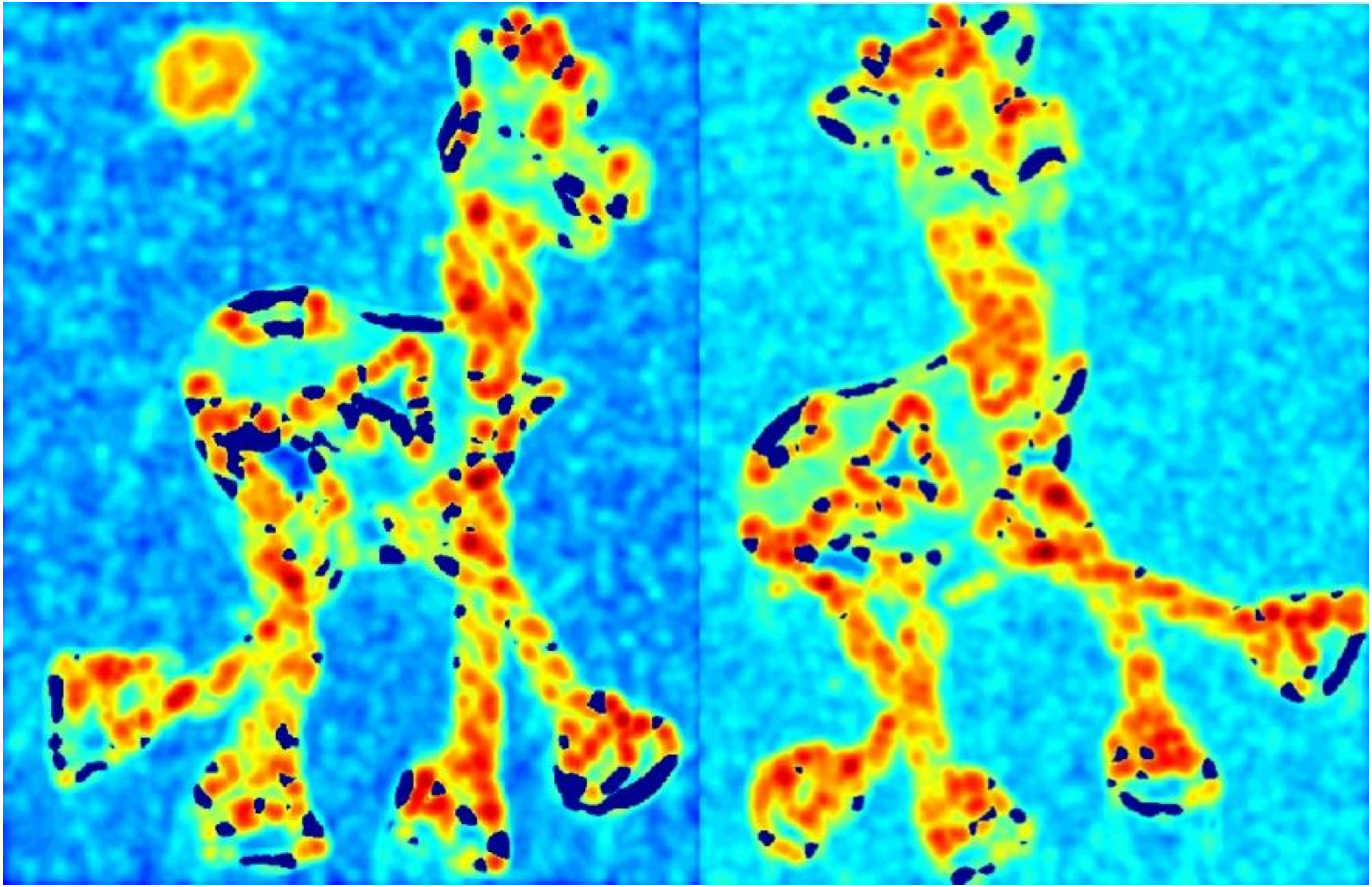
Harris Detector

- The Algorithm:
  - Compute each element of the structure tensor on the image level
  - Find points with large corner response function R (R > threshold)
  - Take the points of local maxima of R

Harris Detector: Steps

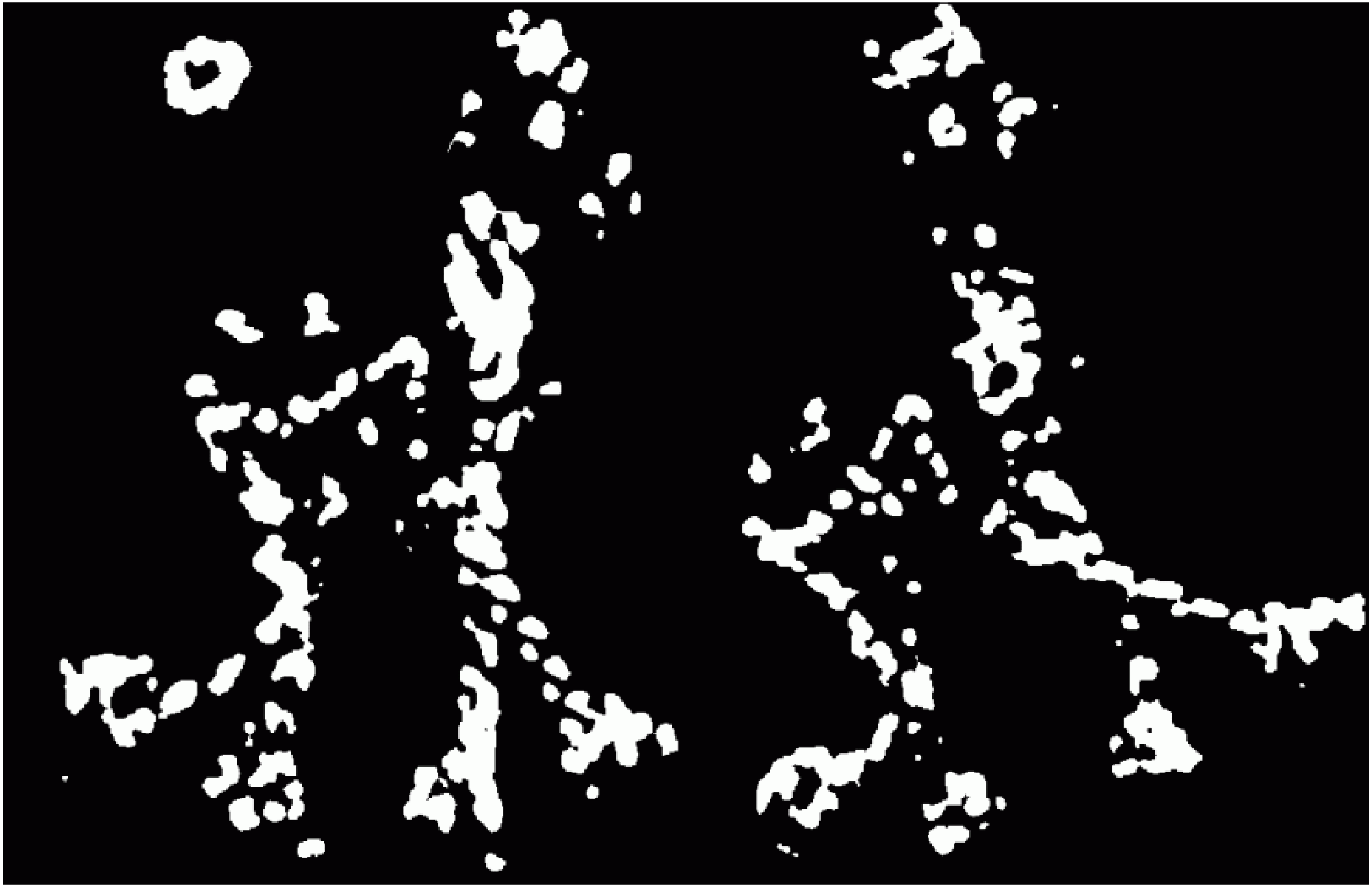
Harris Detector: Steps

Compute corner response R



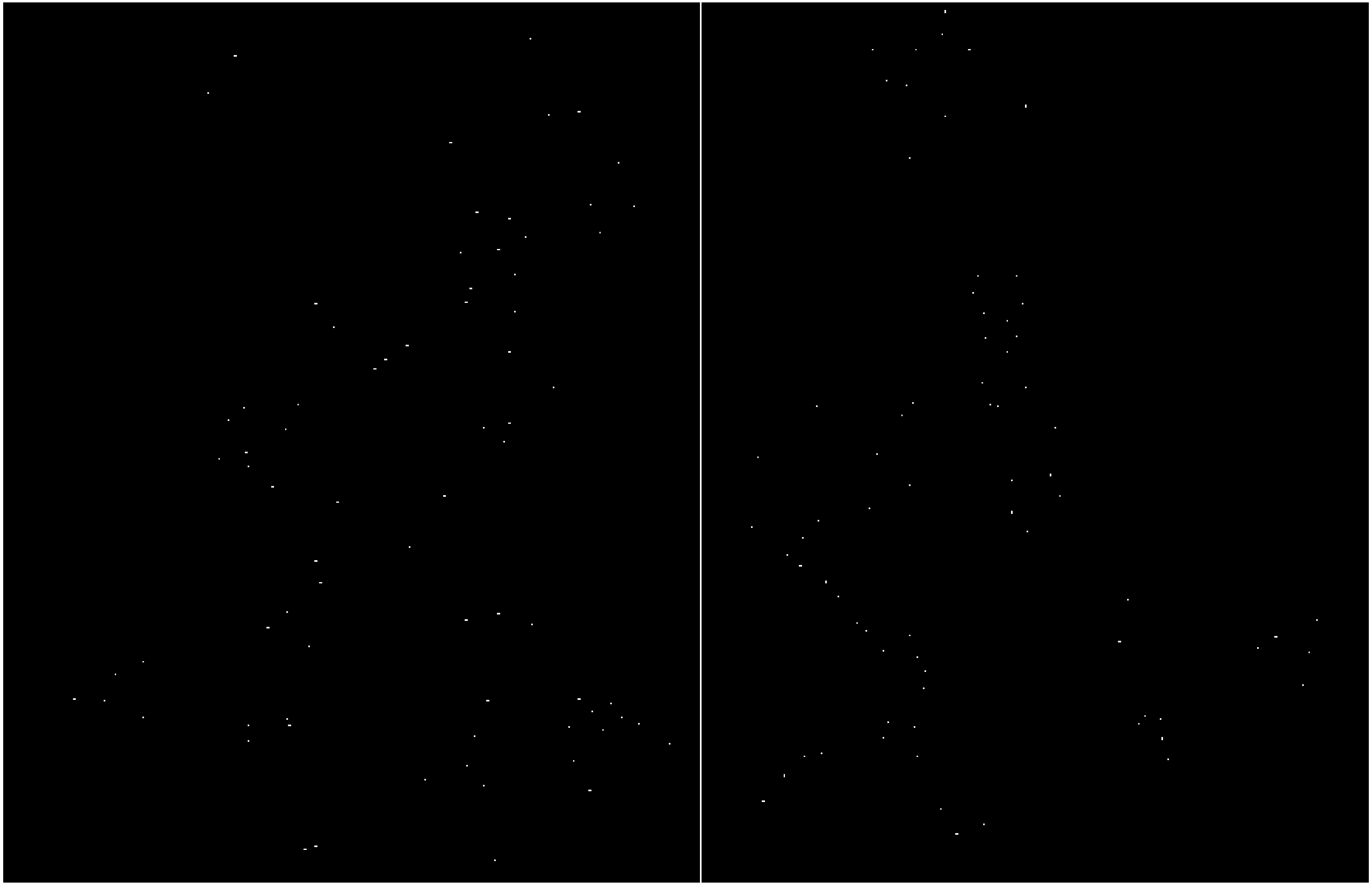
Harris Detector: Steps

Find points with large corner response: R>threshold



Harris Detector: Steps

Take only the points of local maxima of R



Harris Detector: Steps

Invariance and covariance

- We want features to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance:** image is transformed and features do not change
  - **Covariance:** if we have two transformed versions of the same image, features should be detected in corresponding locations

Harris Detector: Summary

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

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

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

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

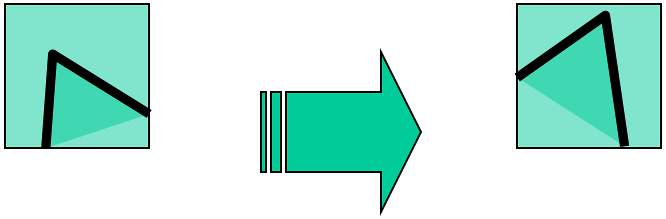
- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

Ideal feature detector

- Would always find the same point on an object, regardless of changes to the image.
- Ie, insensitive to changes in:
  - Scale
  - Lighting
  - Perspective imaging
  - Partial occlusion

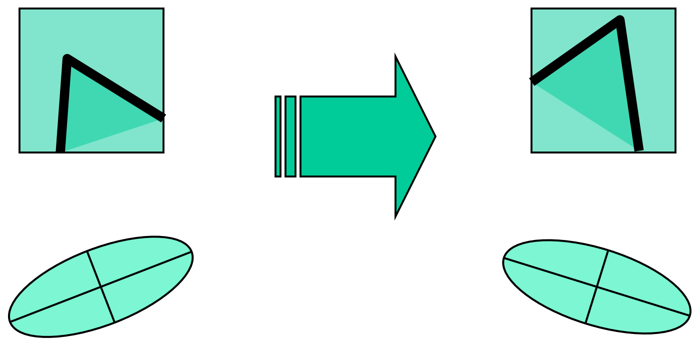
Harris Detector: Some Properties

- Rotation covariance?



Harris Detector: Some Properties

- Rotation invariance?



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner response R is invariant to image rotation

Harris Detector: Some Properties

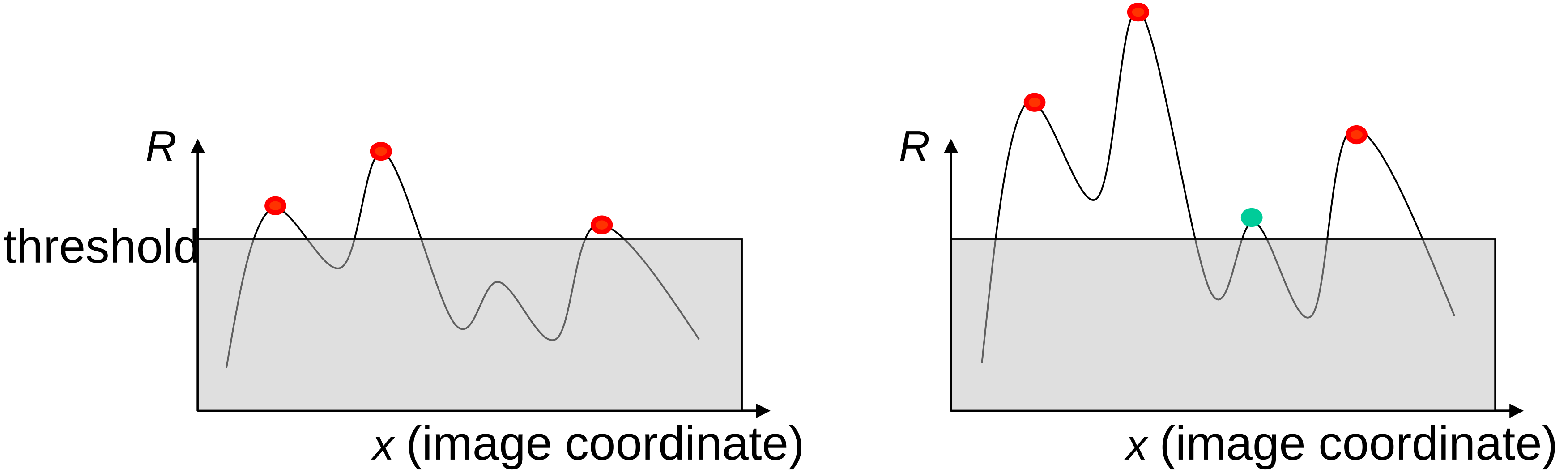
- Invariance to image intensity change?

Harris Detector: Some Properties

- Invariance to image intensity change?
- Partial invariance to additive and multiplicative intensity changes

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

✓ Intensity scale: $I \rightarrow a\,I$ Because of fixed intensity threshold on local maxima, only partial invariance to multiplicative intensity changes.

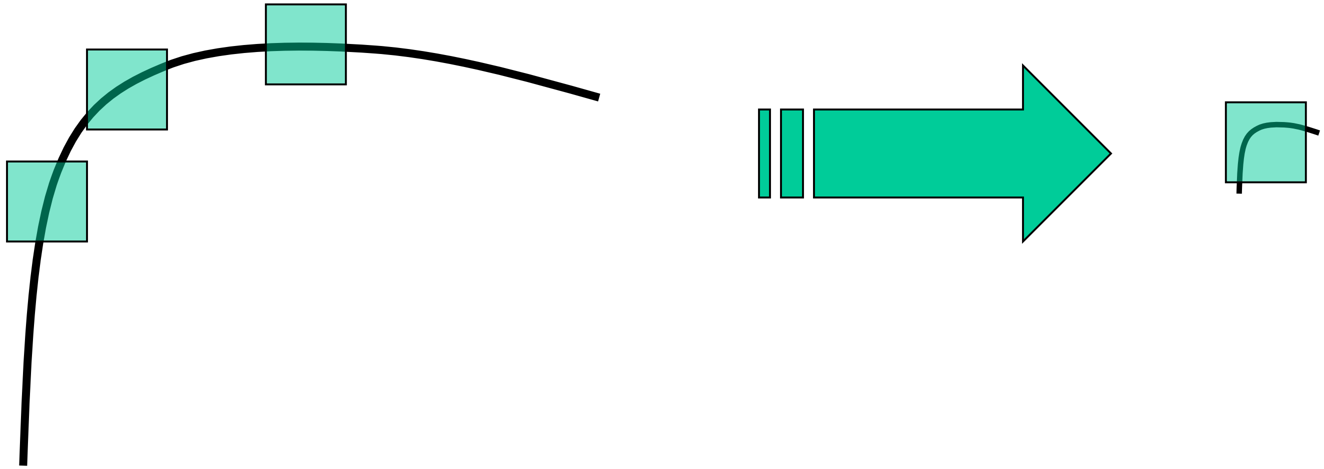


Harris Detector: Some Properties

- Invariant to image scale?

Harris Detector: Some Properties

- Not invariant to *image scale*!



All points will be classified as edges

Corner !

Questions?

- Scale invariant Harris detector?

Summary

- Filtering
- Convolution and cross-correlation
- Gaussian filtering
- Gradients
- Laplacian for blob detection
- Laplacian for scale selection
- Harris detector