Computer Vision I

Basic Image Processing (continued)

03.11.2014

Credits:

Slides partly adapted from:
Bernt Schiele
Michael Black

Stefan Roth

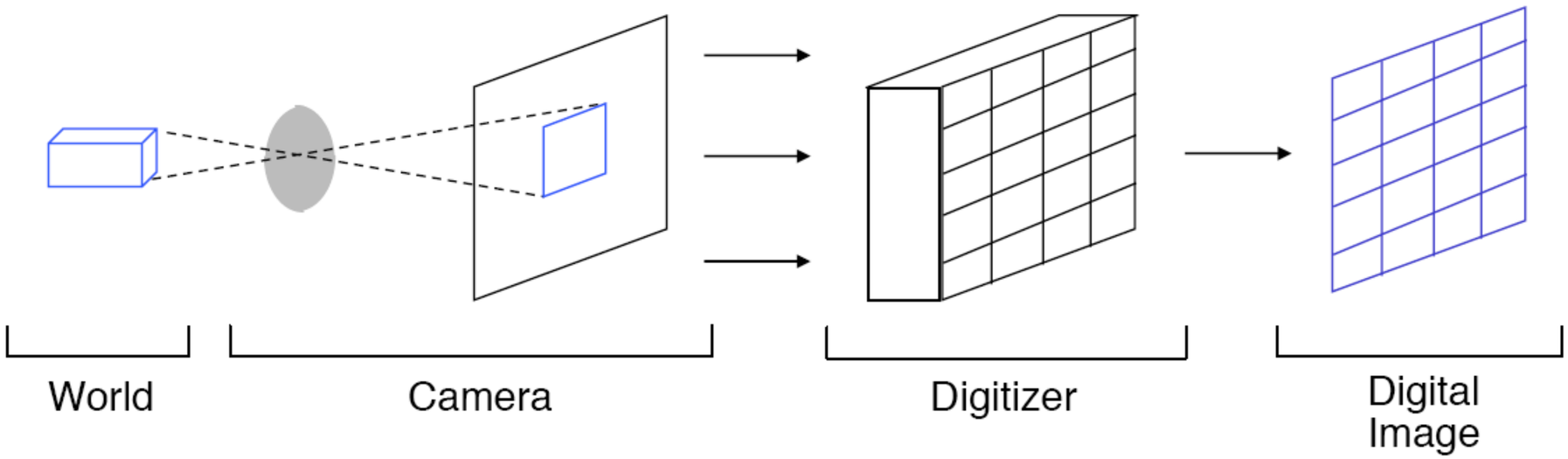
Quick Announcements

- Please pick up key token / account after the lecture
  - … if you have requested one.

- Homework assignment 1
  - Reminder: Deadline postponed
  - Monday, November 10

Overview

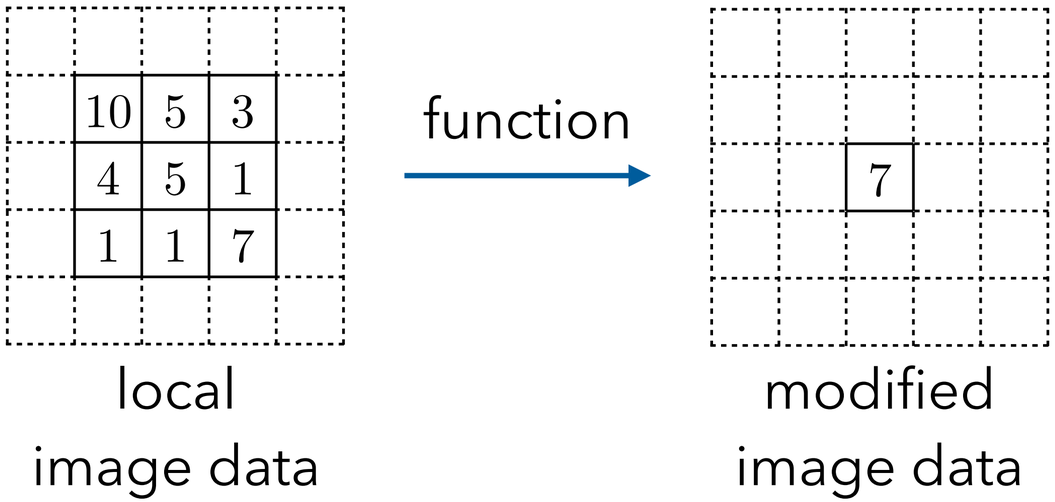
- Computer Vision: “invert” the imaging process
  - Today: 2D (2-dimensional) digital image processing
  - Later: ‘pattern recognition’, image understanding



Digital Image Processing

- Some basics
  - (digital signal processing, FFT, …)
  - Image filtering

- Image filtering
  - to reduce noise
  - to fill-in missing values/information
  - to extract image features (e.g. edges/corners)
  - …



Today: Basics of Digital Image Processing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - Motivation: Recognition using line drawings
  - Image derivatives (1st and 2nd order)

Special case: Binary Images

- Morphological operations:
  - Perform convolution with a "structuring element" s
  - This is a binary mask (often circle or square)
  - Then perform thresholding to recover a binary image:

$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases}$$

Note: As defined the morphological operations apply only to binary images!

Generalization does exist…

original

dilation
$\theta(f * s, 1)$

erosion
$\theta(f * s, S)$

# of pixels in s

Morphological filters – Side note

- Generalization to grayscale images:
  - Perform min/max-convolution with a "structuring element" s
  - We get our previous behavior with flat structuring element $s(\mathbf{x}) = \begin{cases} 0 & \mathbf{x} \in B \\ -\infty & \text{otherwise} \end{cases}$

original

dilation

$$\max_{\mathbf{y}}[f(\mathbf{x}) + s(\mathbf{x} - \mathbf{y})]$$

erosion

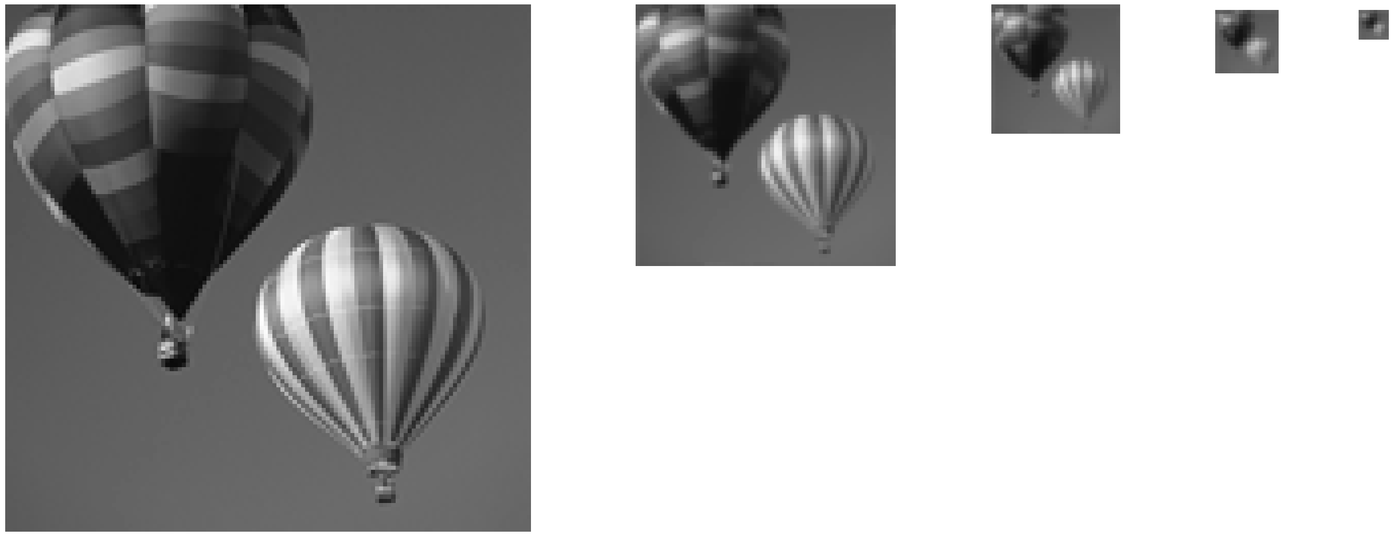
$$\min_{\mathbf{y}}[f(\mathbf{x}) - s(\mathbf{y} - \mathbf{x})]$$

Today: Basics of Digital Image Processing

- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - Motivation: Recognition using line drawings
  - Image derivatives (1st and 2nd order)

Image pyramids

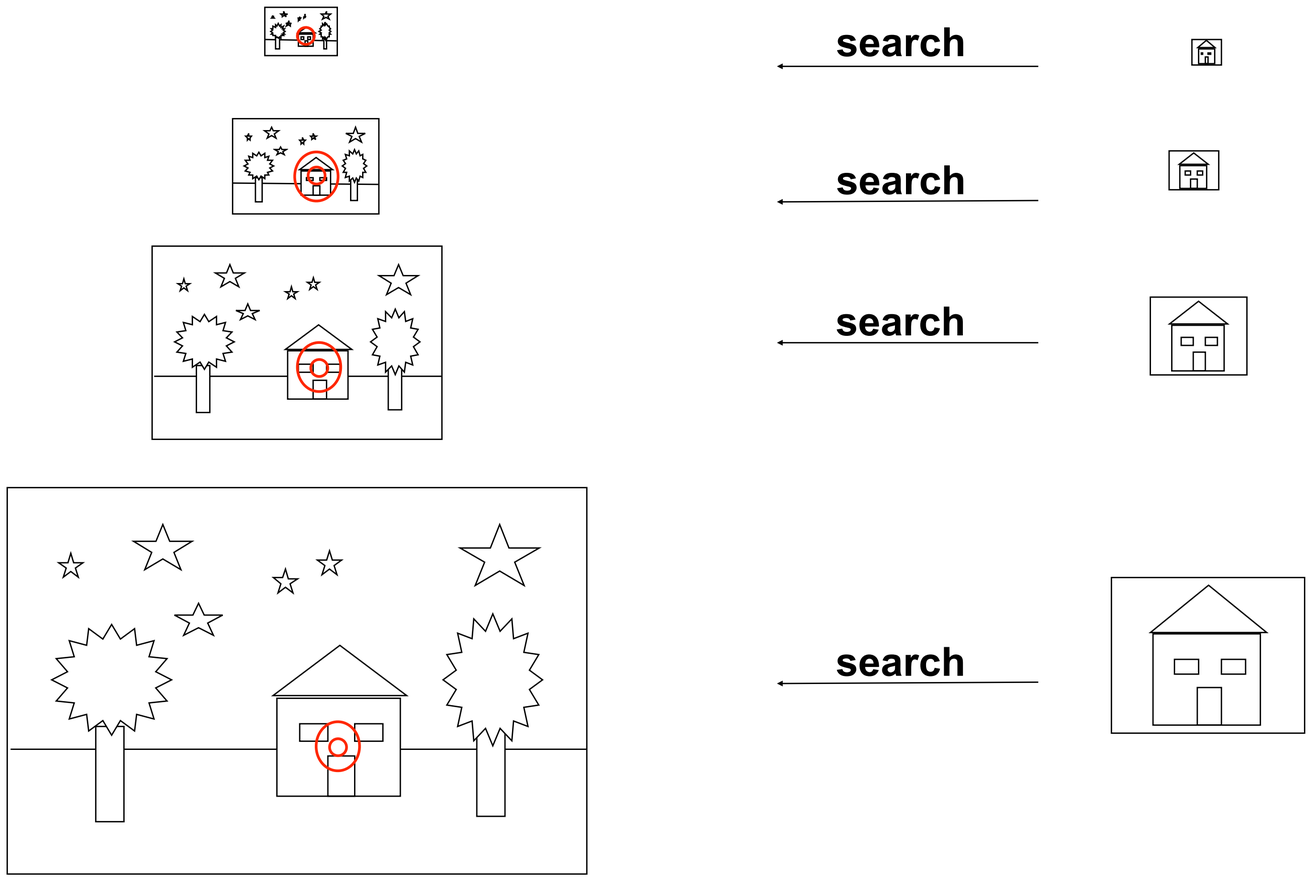
- Represent images at multiple scales (resolutions)
  - Stacked up like a pyramid.
  - E.g., Gaussian pyramid:



high resolution → low resolution

[Irani & Basri]

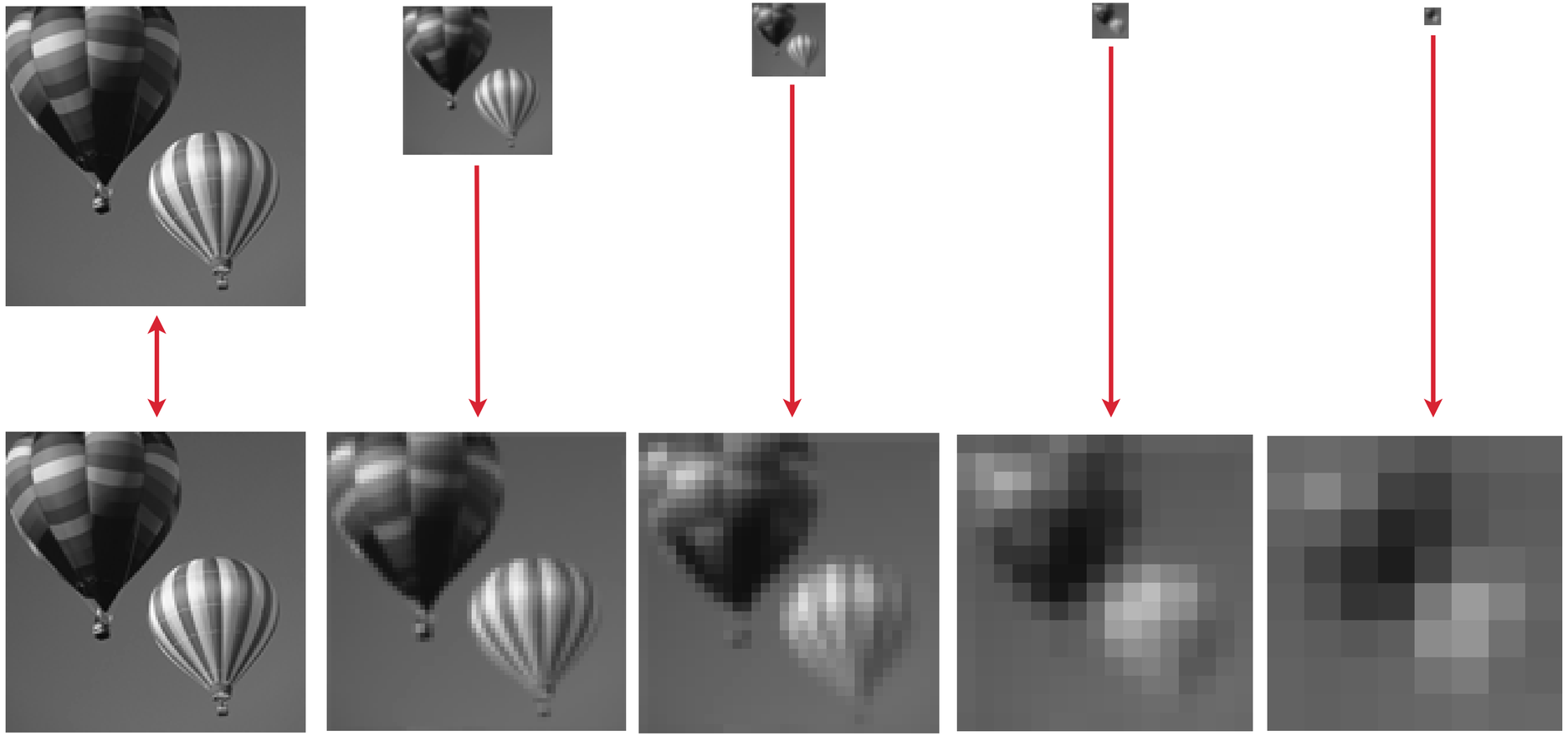
Motivation: Search



[Irani & Basri]

Gaussian pyramids

- Represent images at multiple scales (resolutions)
  - Stacked up like a pyramid.
  - E.g. Gaussian pyramid:



[Irani & Basri]

Another Example

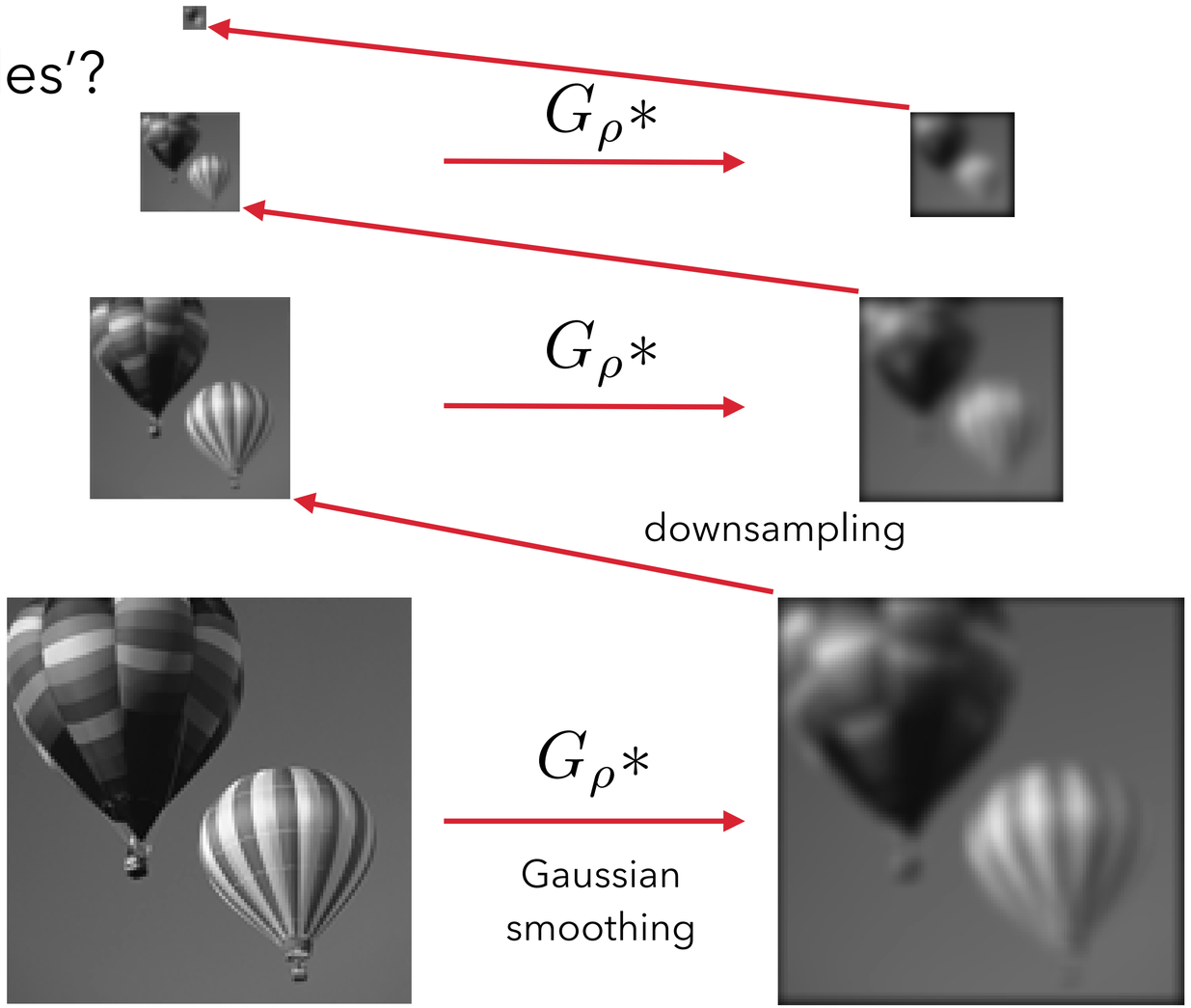
- A bar
  - in the big images is a hair (on the zebra's nose)
  - in smaller images, a stripe
  - in the smallest image, the animal's nose

512 256 128 64 32 16 8

[Ponce & Forsyth]

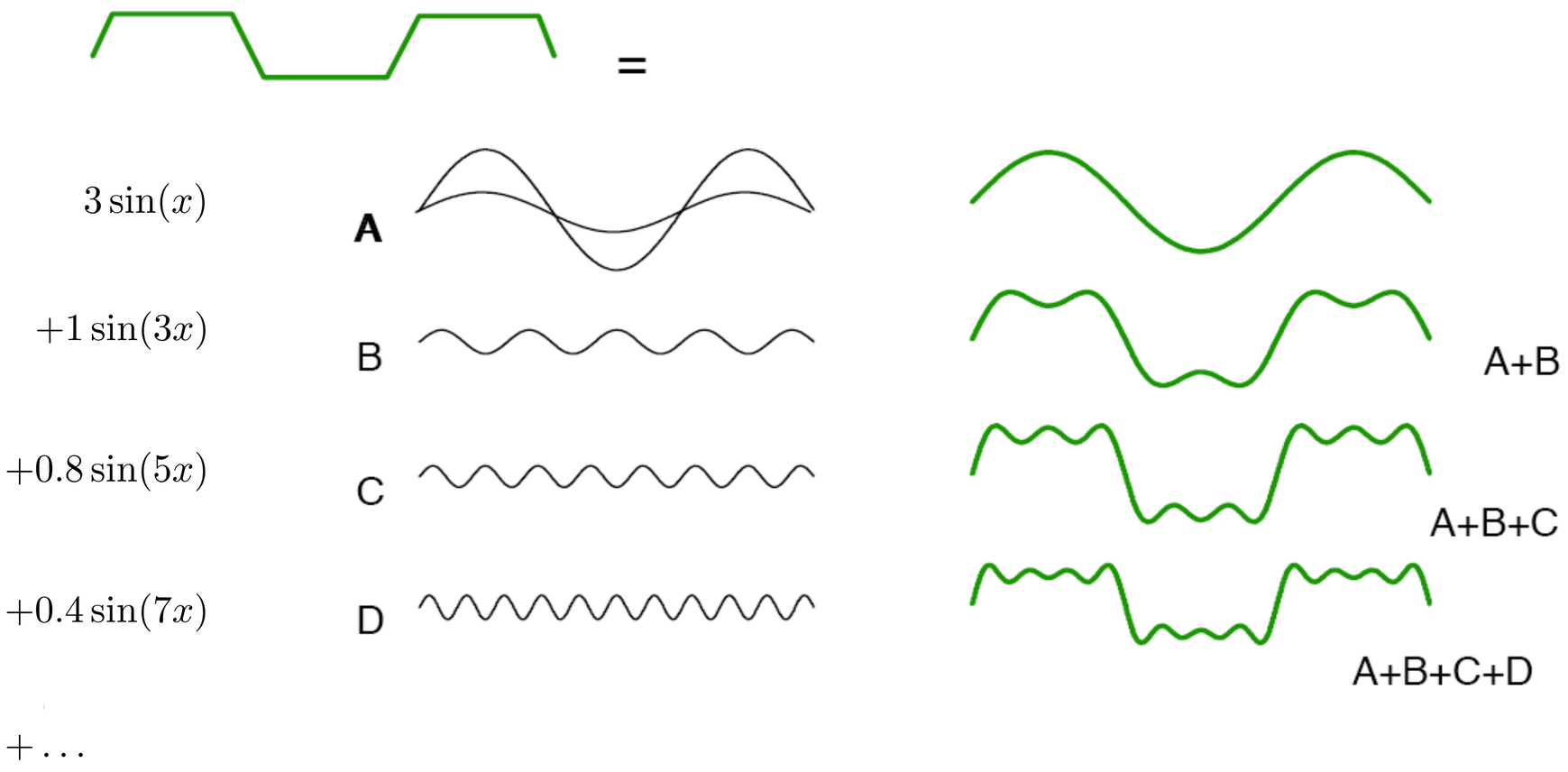
Gaussian pyramids

- Which information is preserved over ‘scales’?
- Which information is lost over ‘scales’?

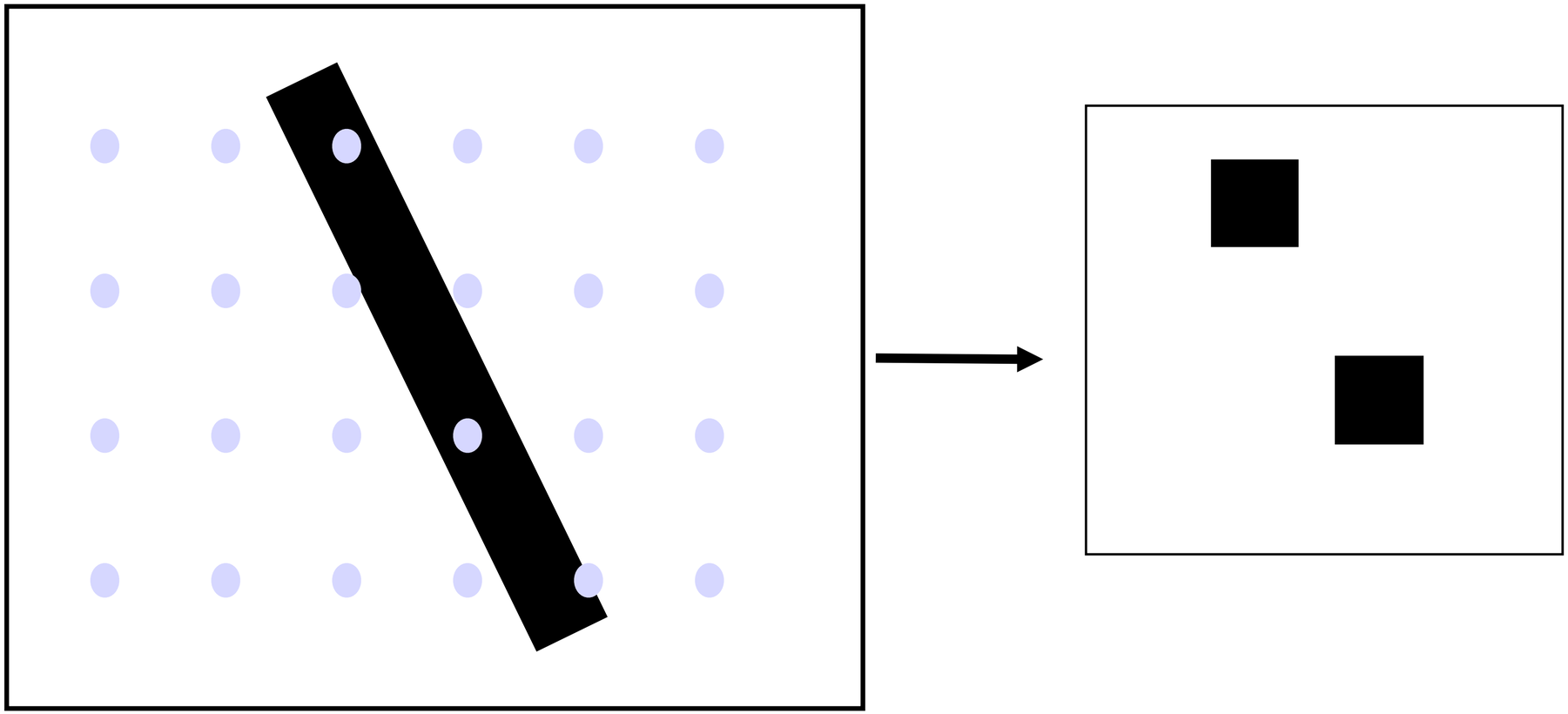


Transforming to the Frequency Domain in Pictures

- A tiny bit about spatial frequencies… (cf. HCS)



Subsampling

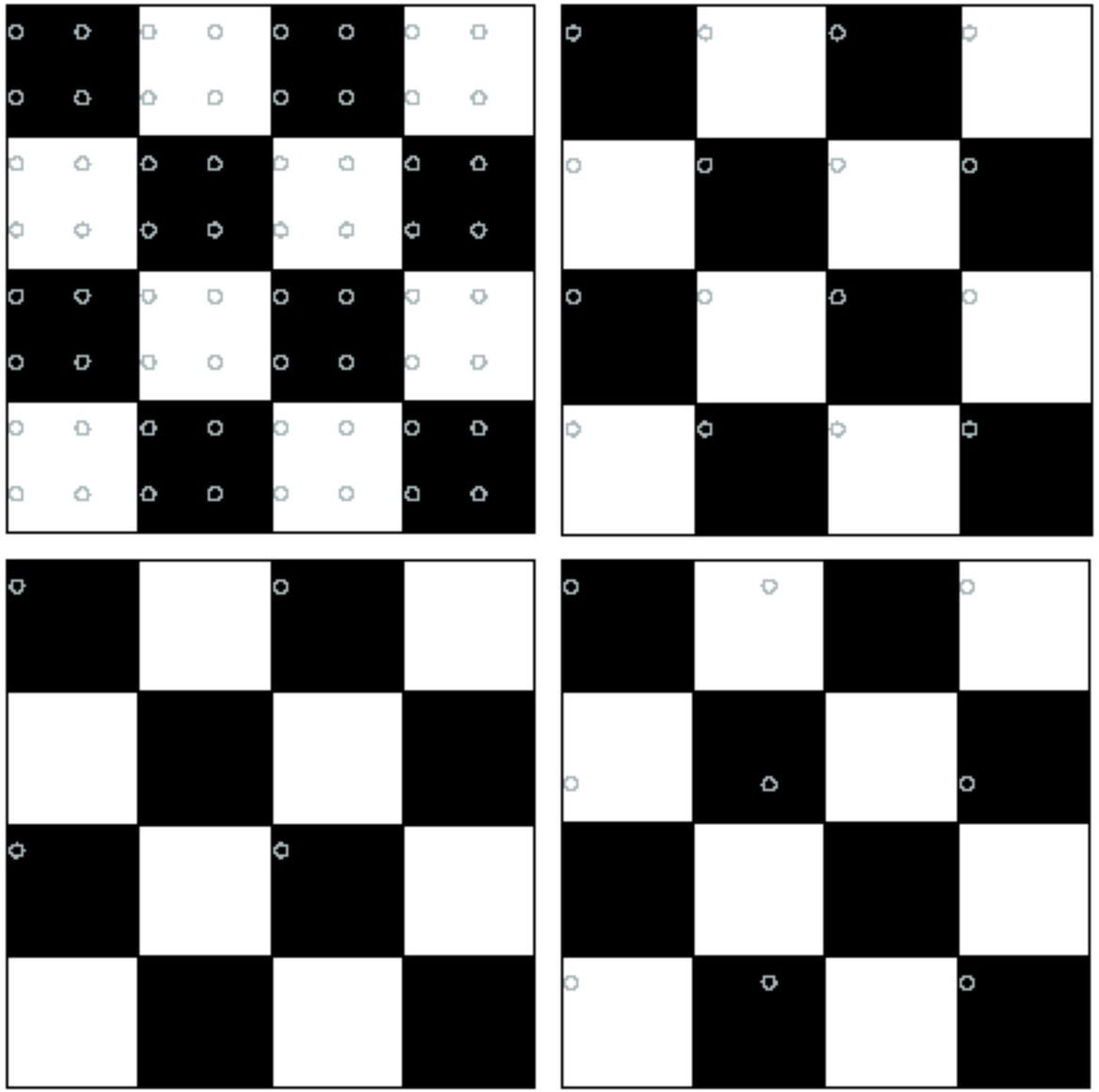


Aliasing

- Cannot shrink an image by taking every second pixel.
  - High frequencies cannot be represented anymore.

- If we do that anyway, characteristic errors appear:
  - Spatial frequencies are misinterpreted (aliasing)
  - Typically, small phenomena look bigger; fast phenomena can look slower.
  - Common examples:
    - Wagon wheels rolling the wrong way in movies
    - Checkerboards misrepresented in ray tracing
    - Striped shirts look strange on color television.

Aliasing



Resample the checkerboard by taking one sample at each circle.

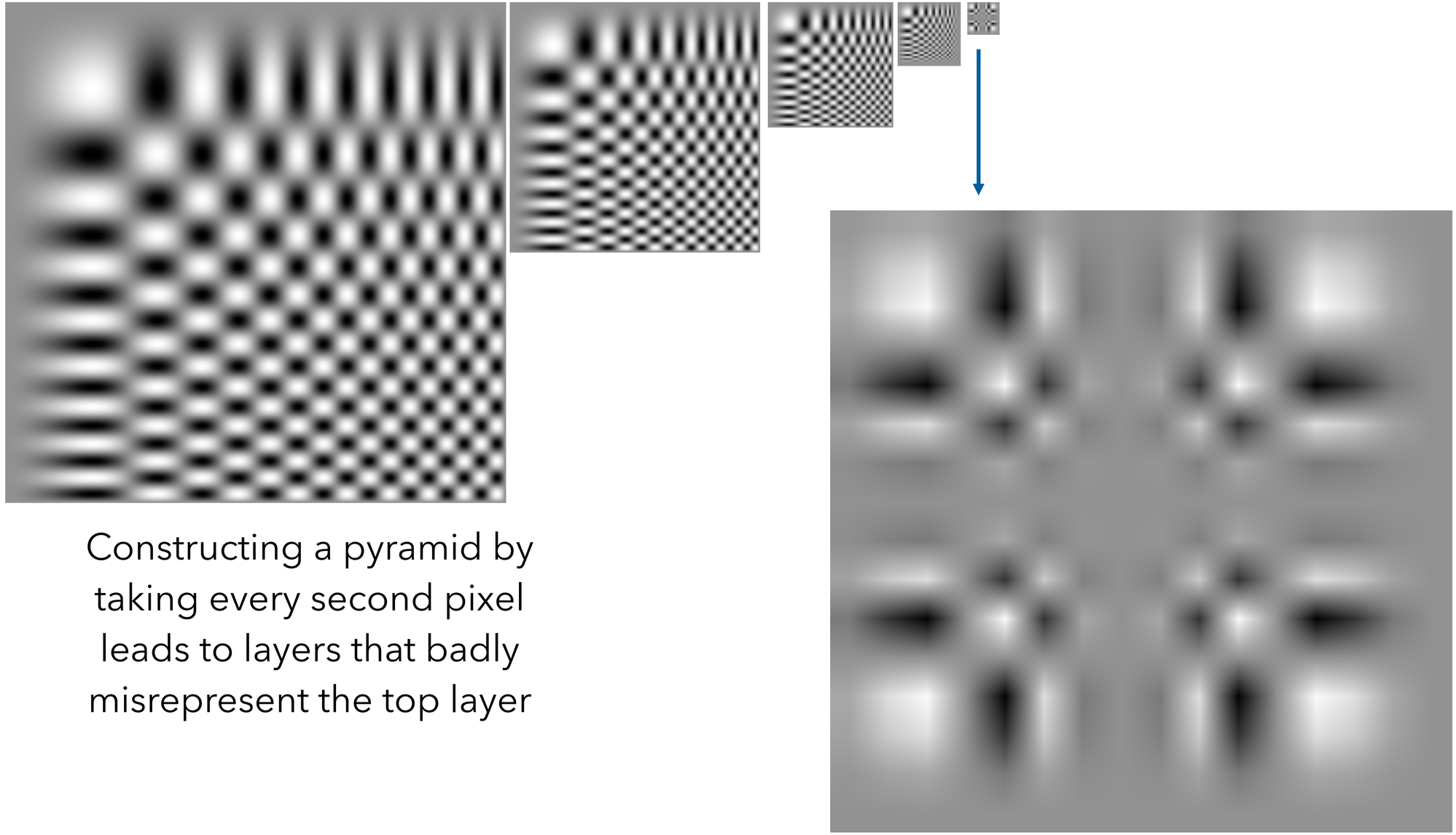
In the case of the top left board, new representation is reasonable.

Top right also yields a reasonable representation.

Bottom left is all black (dubious) and bottom right has checks that are too big.

[Ponce & Forsyth]

Aliasing



Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

[Ponce & Forsyth]

Today: Basics of Digital Image Processing

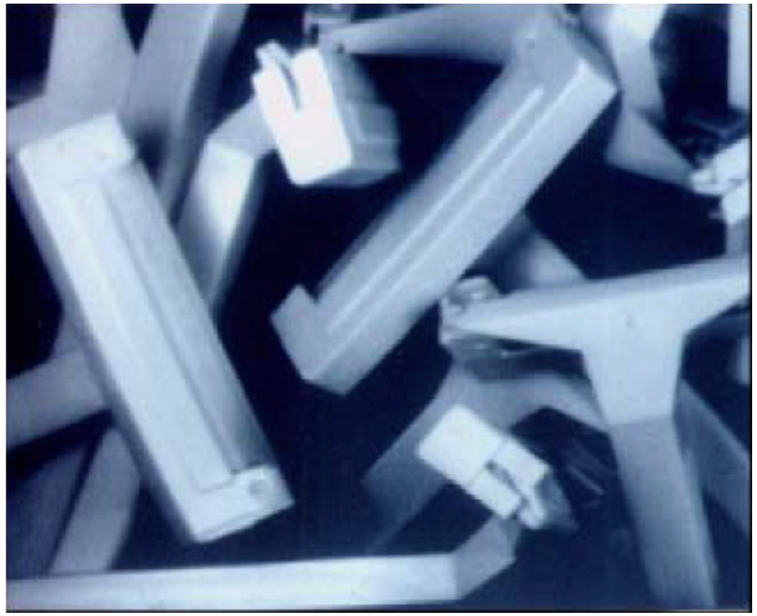
- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - Motivation: Recognition using line drawings
  - Image derivatives (1st and 2nd order)

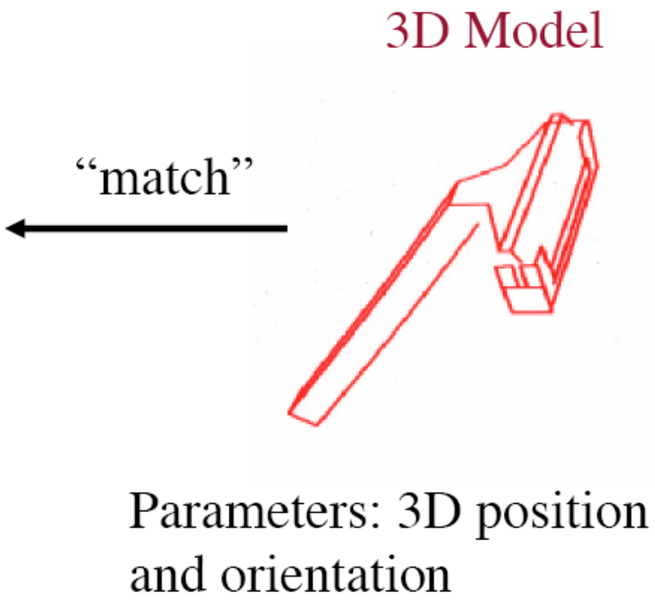
Line Drawings: Good Starting Point for Recognition?

- We humans can recognize an object through a line drawing
- Are edges important for recognition?

Example of Recognition & Localization

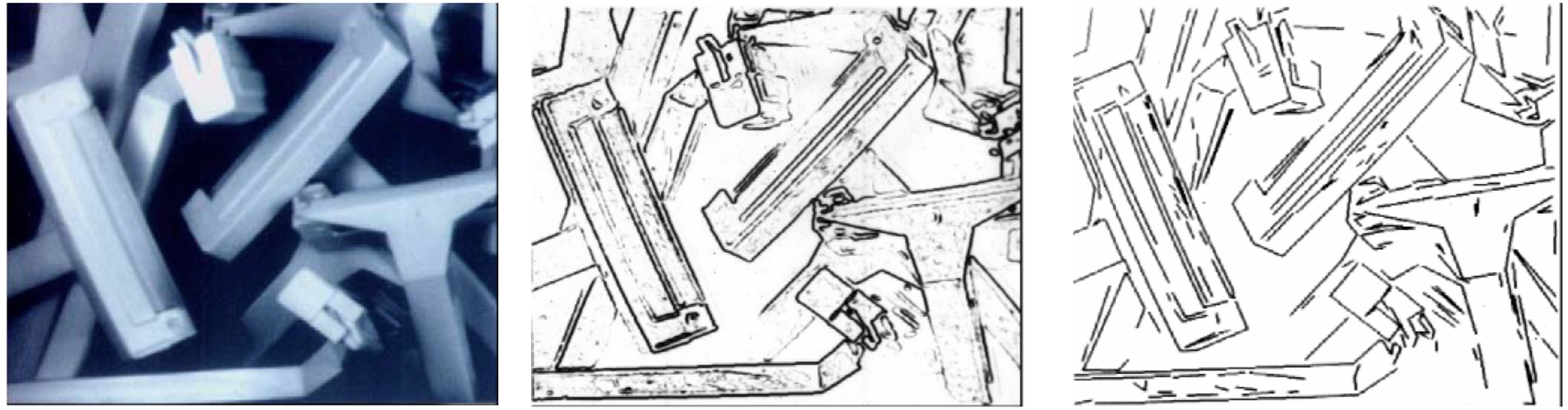
- Simple approach [David Lowe]





Example of Recognition & Localization

- Simple approach [David Lowe]
  - 1. “Filter” image to find brightness changes
  - 2. “Fit” lines to the raw measurements

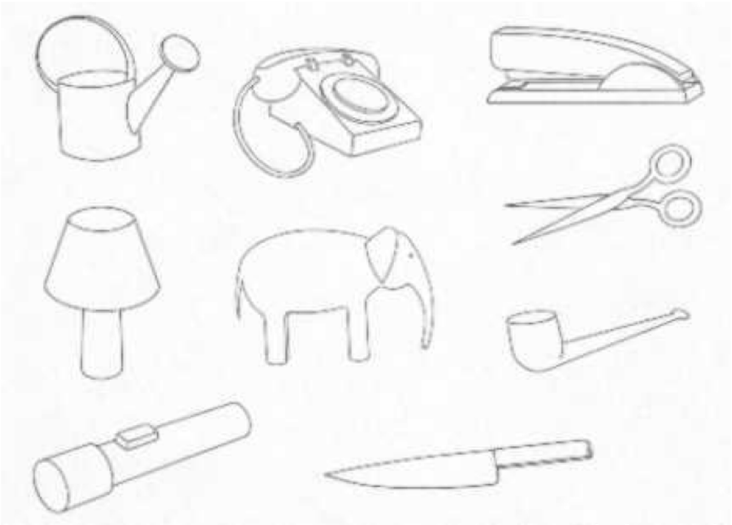


Example of Recognition & Localization

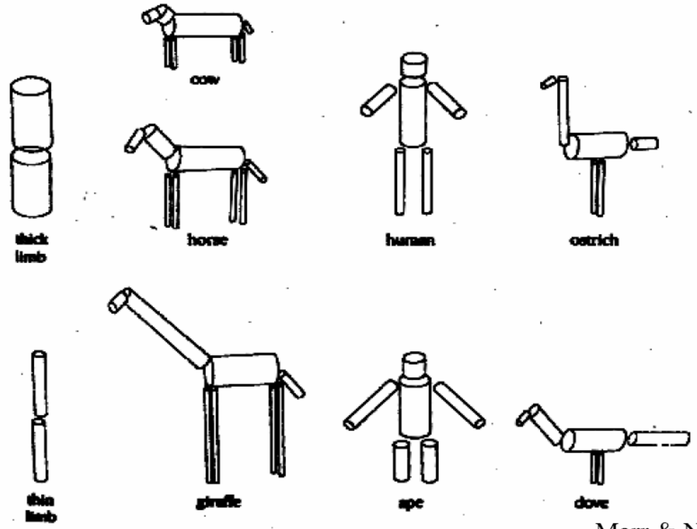
- Simple approach [David Lowe]
  - 3. “Project” model into the image and “match” to lines (solving for 3D pose)

Object Models

- (Mostly) Historical idea & approach
  - Matching of models (wire-frame/geons/generalized cylinders…) to edges and lines



Biederman's Geons

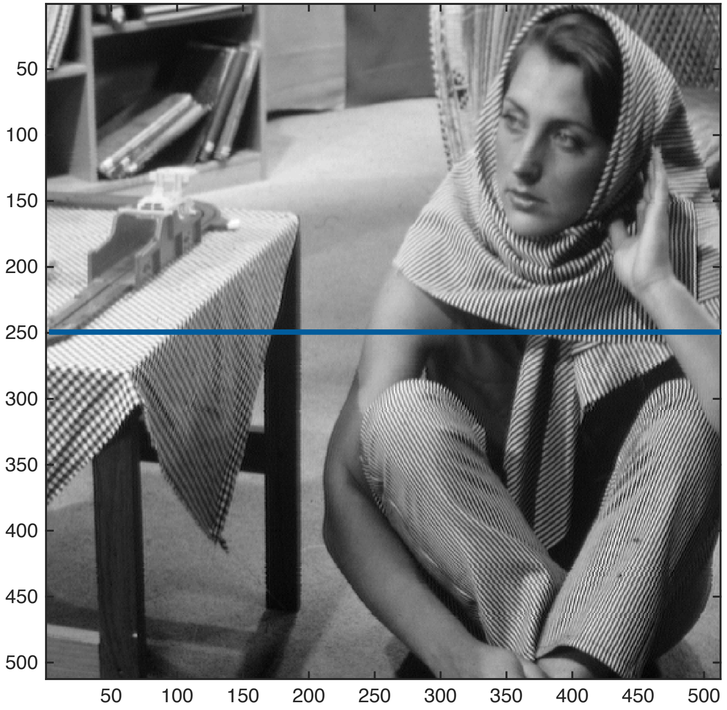


Marr & Nishihara

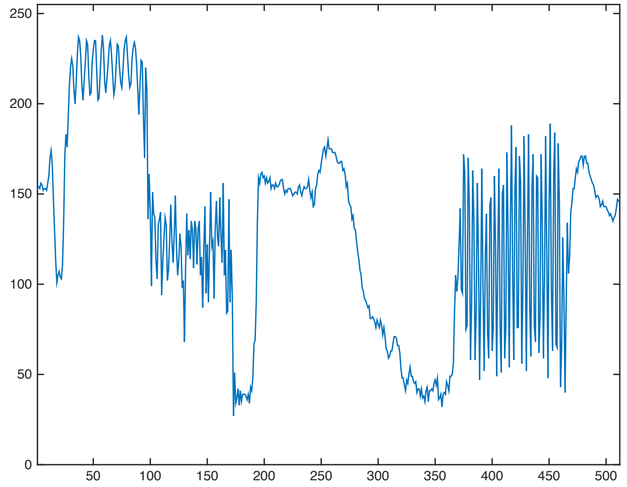
- The "only" remaining problem to solve is:
  - reliably extract lines & edges that can be matched to these models…

Image scanline

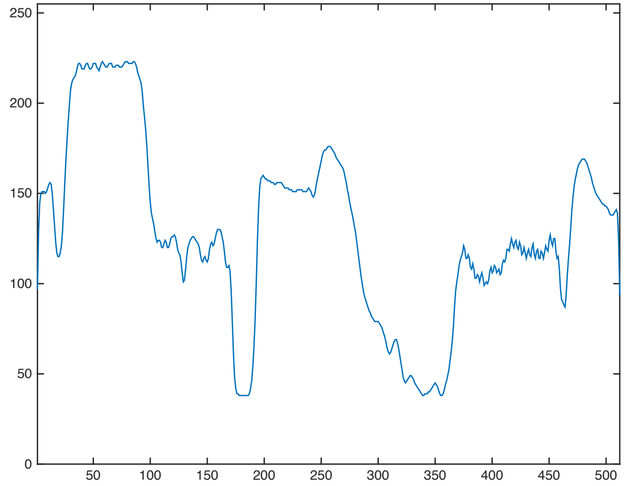
- Barbara image:
  - entire image



line 250:

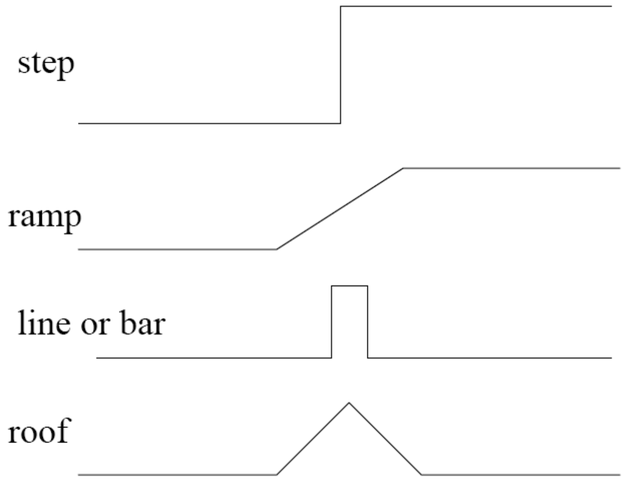


line 250 smoothed with a Gaussian:

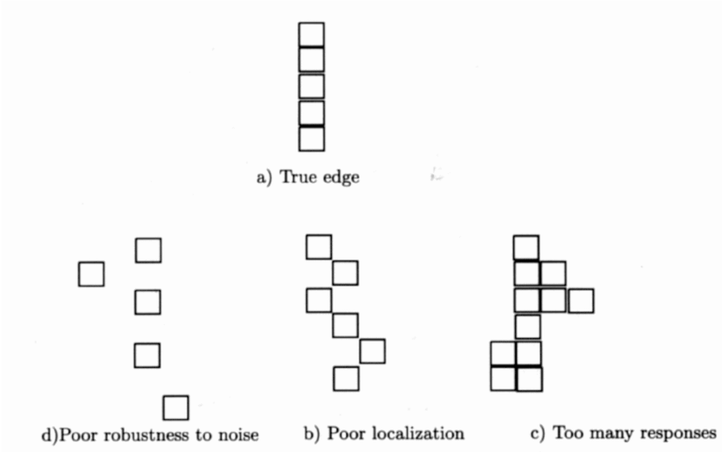


What are “edges” (1D)

- Idealized edge types:



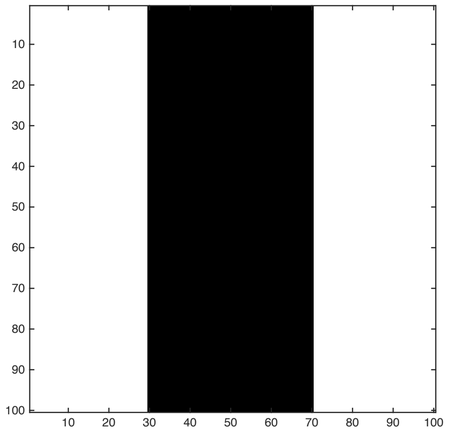
- Goals of edge detection:
  - *good detection:* filter responds to edge, not to noise
  - *good localization:* detected edge near true edge
  - *single response:* one per edge



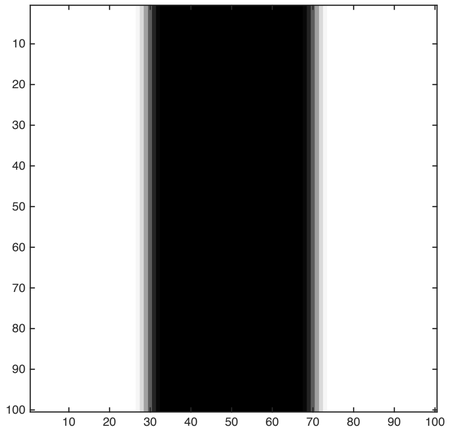
Edges

- Correspond to fast changes,
- where the magnitude of the derivative is large

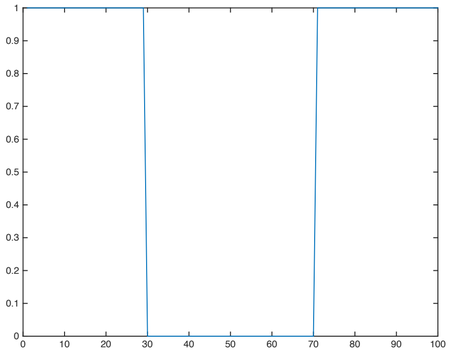
"image" of 2 step-edges

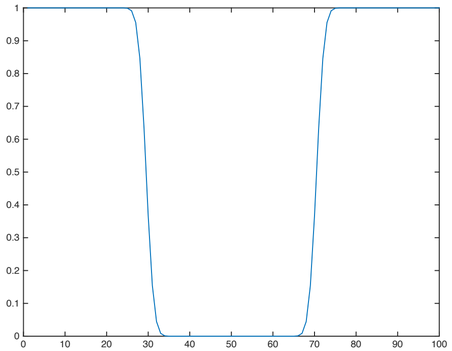


smoothing

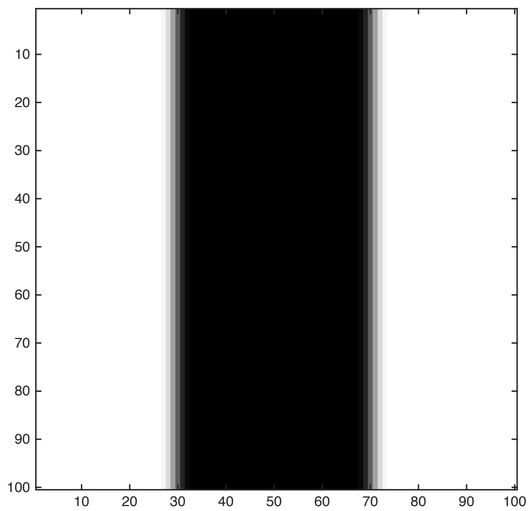


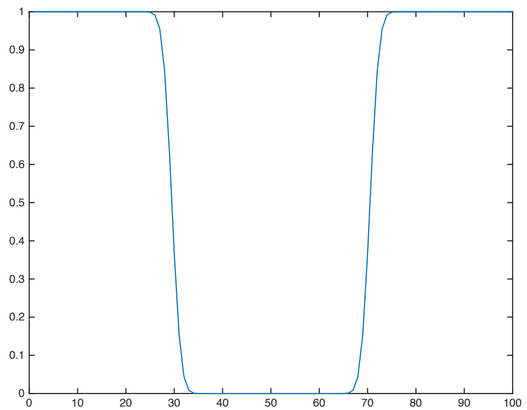
single line of "image"





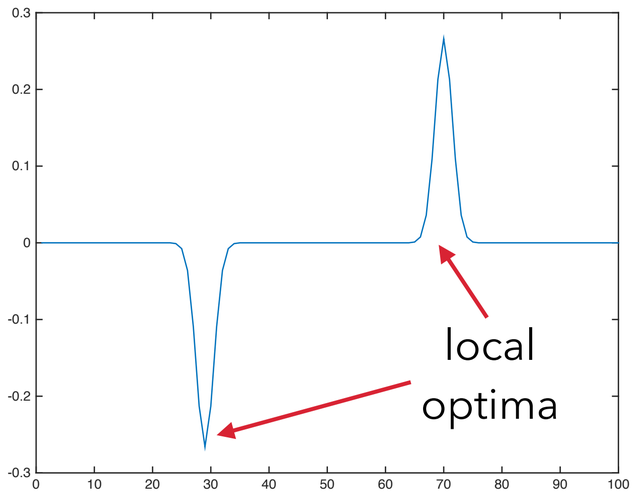
Edges & Derivatives…

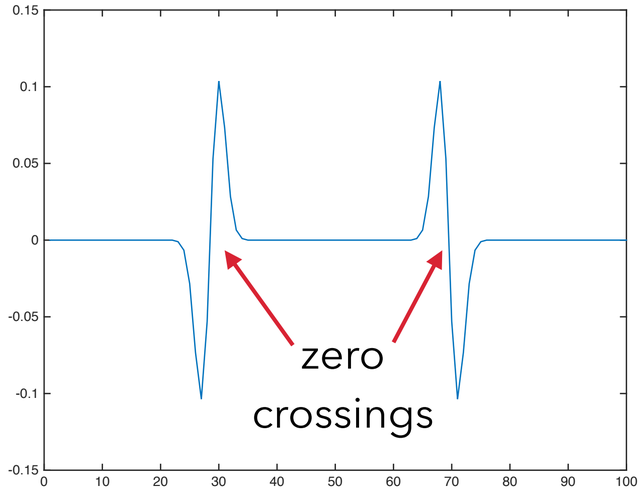




1st derivative

2nd derivative





Compute Derivatives

$$\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x)$$

- We can implement this as a linear filter:
  - forward differences:
    (convolution filter)

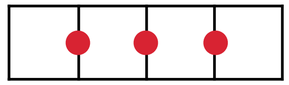
1	-1

  - or central differences:

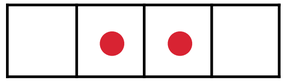
$$\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} \approx \frac{f(x+1) - f(x-1)}{2}$$

$$\frac{1}{2} \cdot \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline \end{array}$$

Where is the derivative computed?



in between pixels



at pixels

Edge Detection

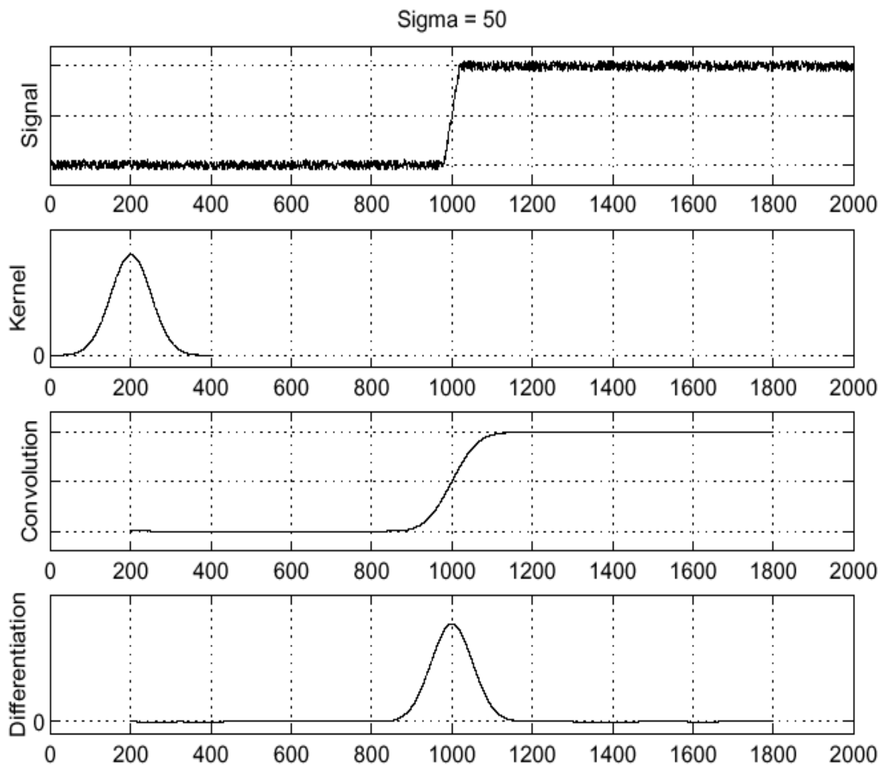
- based on 1st derivative:
  - smooth with Gaussian
  - calculate derivative
  - finds its local optima

f

g

$g * f$

$$\frac{\mathrm{d}}{\mathrm{d}x}(g * f)$$



Edge Detection

- Simplification:

$$\frac{\mathrm{d}}{\mathrm{d}x}(g * f) = \left(\frac{\mathrm{d}}{\mathrm{d}x} g\right) * f$$

  - saves one operation

derivative of Gaussian

f

$\frac{\mathrm{d}}{\mathrm{d}x} g$

$\left(\frac{\mathrm{d}}{\mathrm{d}x} g\right) * f$

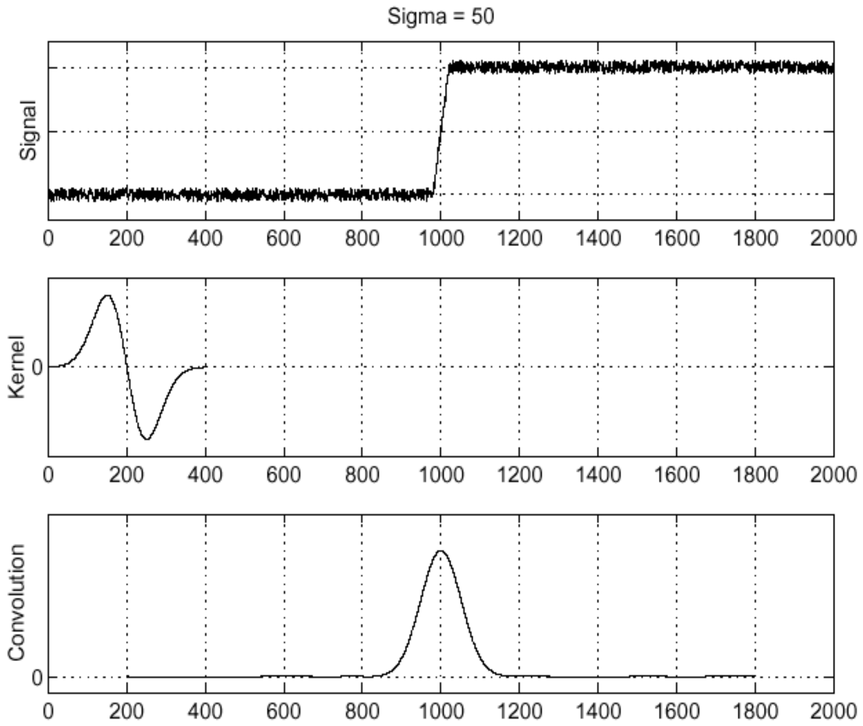
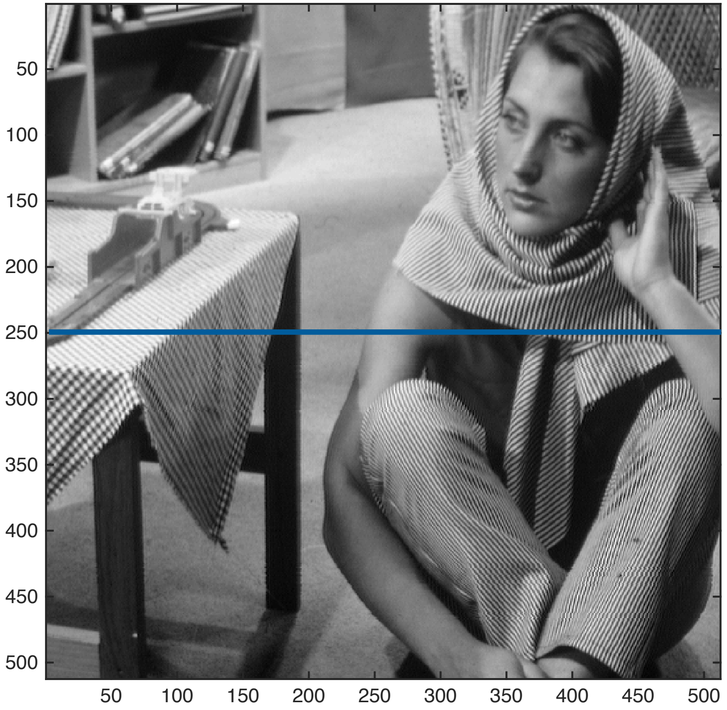
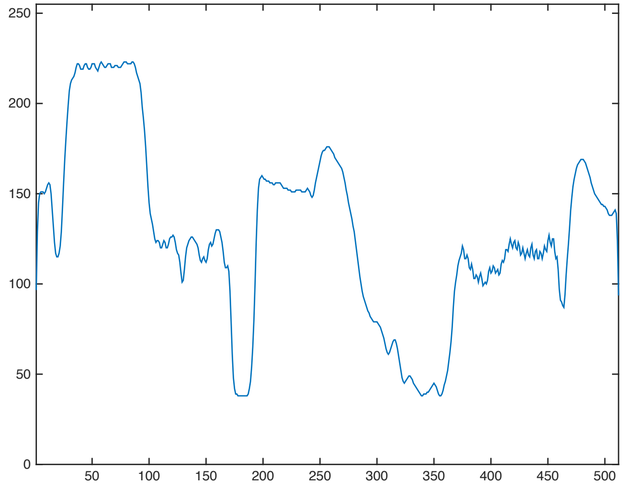


Image scanline

- Barbara image:
  - entire image



line 250 (smoothed):



1st derivative

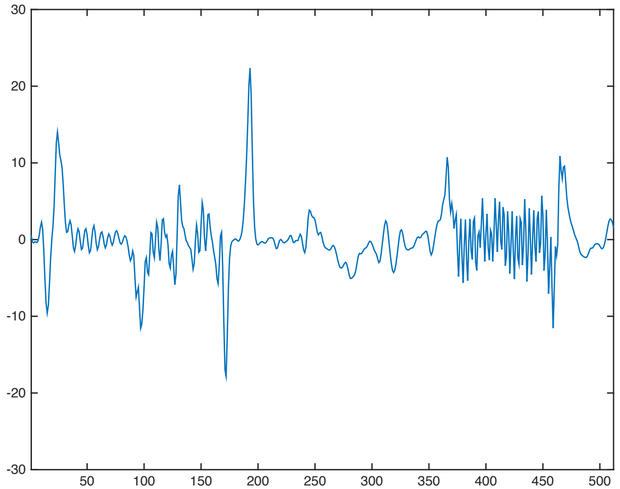
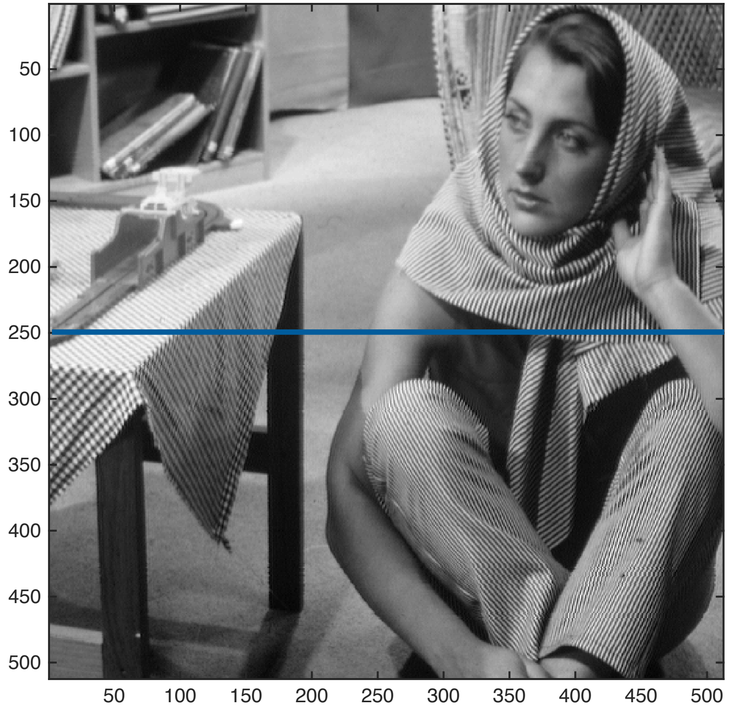


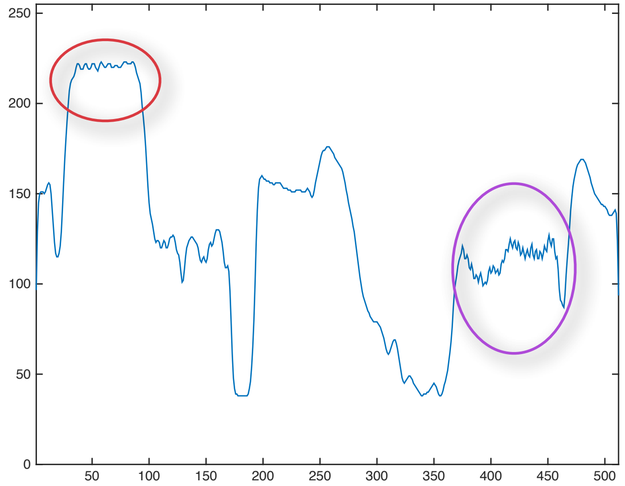
Image scanline

- Barbara image:
  - entire image

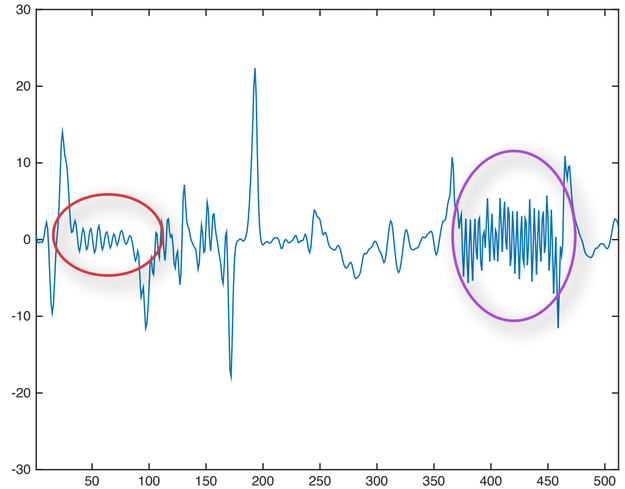


Amplification of small variations!

line 250 (smoothed):



1st derivative



Implementing 1D edge detection

- Algorithmically:
  - find peaks in the 1st derivative

- But
  - should be a local optimum
  - should be 'sufficiently' large

- Use 2 thresholds (called hysteresis)
  - high threshold (of absolute value) to start edge curve
  - low threshold to continue them
  - (really only makes sense in 2D… hold on)

Extension to 2D Edge Detection: Partial Derivatives

- Partial derivatives
  - in x direction:

$$\frac{\mathrm{d}}{\mathrm{d}x} I(x,y) = I_x \approx D_x * I$$

in y direction:

$$\frac{\mathrm{d}}{\mathrm{d}y} I(x,y) = I_y \approx D_y * I$$

- often approximated with simple filters (finite differences):
  - first try

$$D_x = \frac{1}{6} \cdot \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

$$D_y = \frac{1}{6} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Extension to 2D Edge Detection: Partial Derivatives

- Partial derivatives
  - in x direction:

$$\frac{\mathrm{d}}{\mathrm{d}x} I(x,y) = I_x \approx D_x * I$$

in y direction:

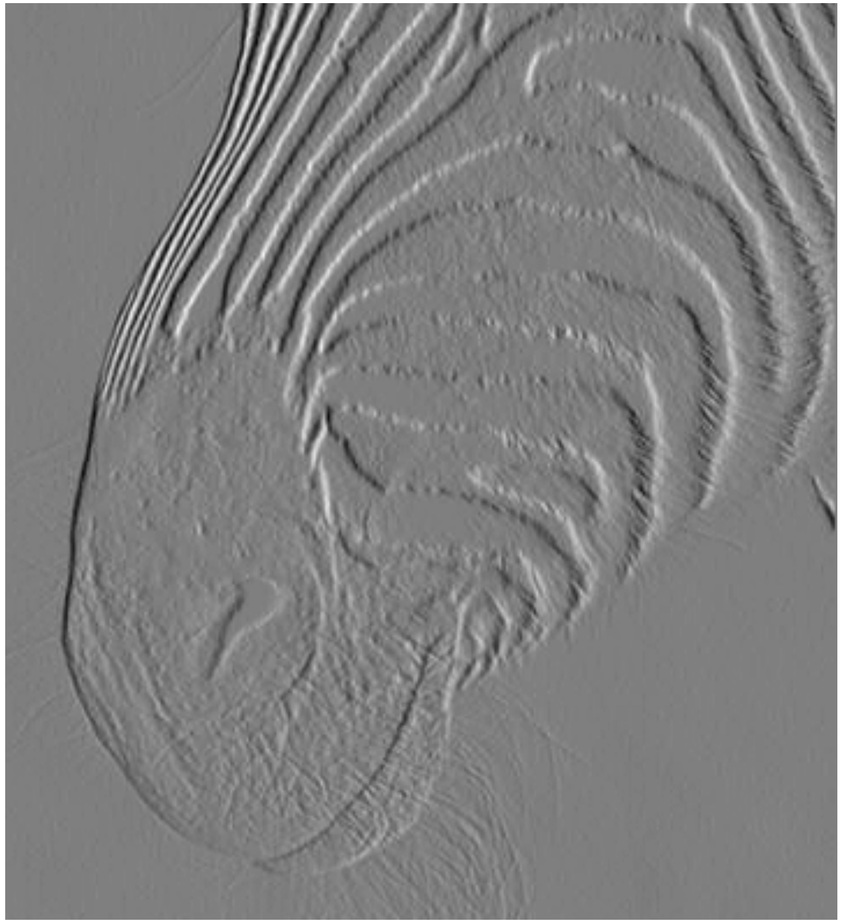
$$\frac{\mathrm{d}}{\mathrm{d}y} I(x,y) = I_y \approx D_y * I$$

- often approximated with simple filters (finite differences):
  - second try: Sobel filters (Gaussian smoothing in opposite direction)

$$D_x = \frac{1}{8} \cdot \begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array}$$

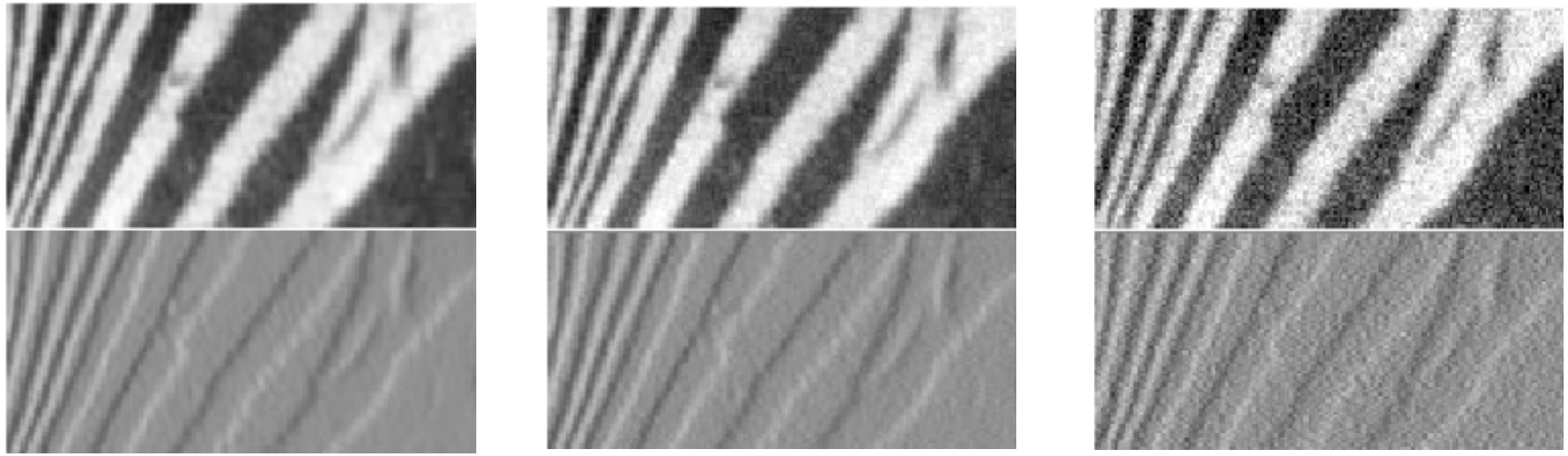
$$D_y = \frac{1}{8} \cdot \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Finite Differences



Finite differences responding to noise

- Increasing noise level (from left to right)
  - noise: zero mean additive Gaussian noise

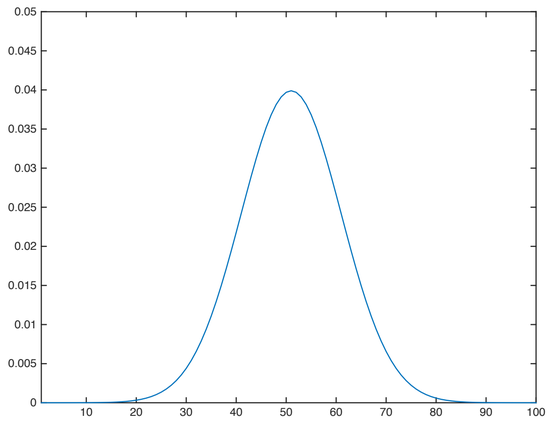


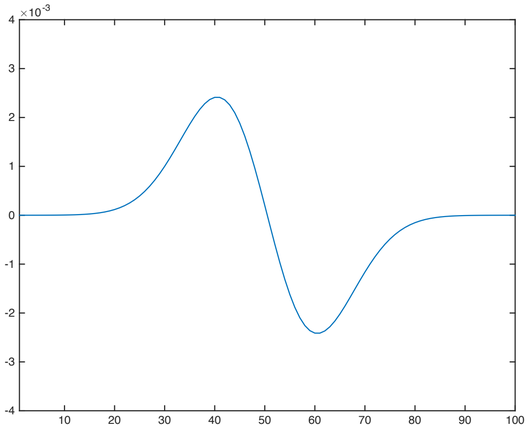
Again: Derivatives and Smoothing

- Derivative in x-direction:

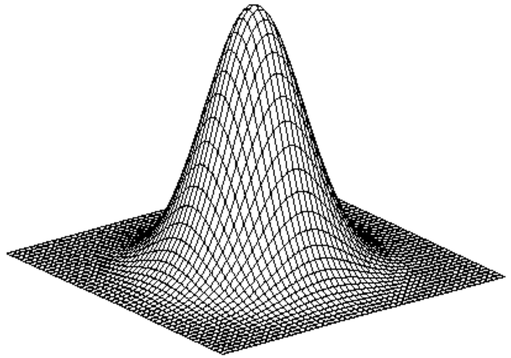
$$D_x * (G * I) = (D_x * G) * I$$

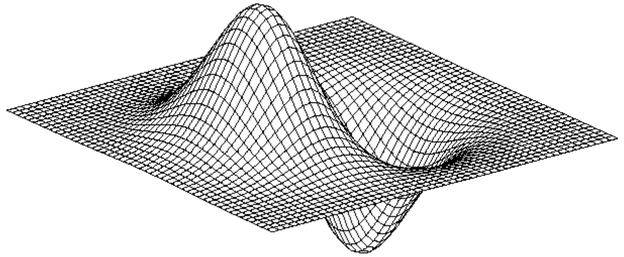
  - in 1D:





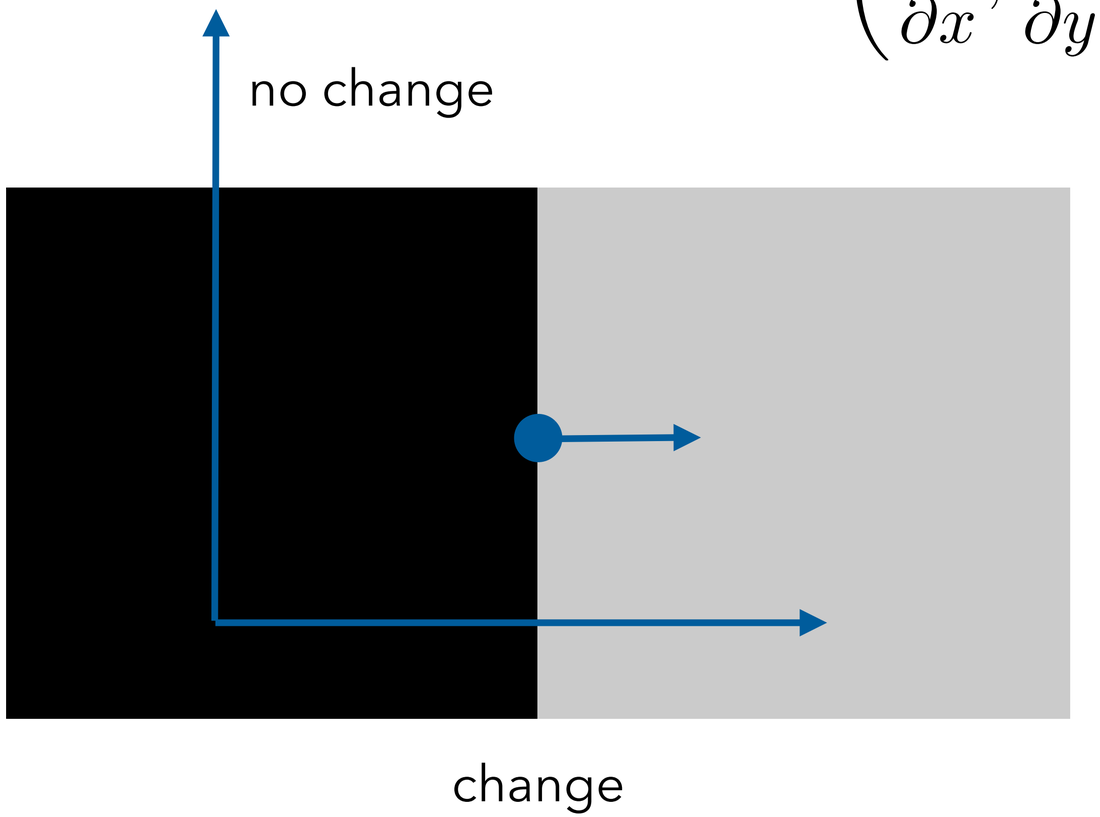
  - in 2D:





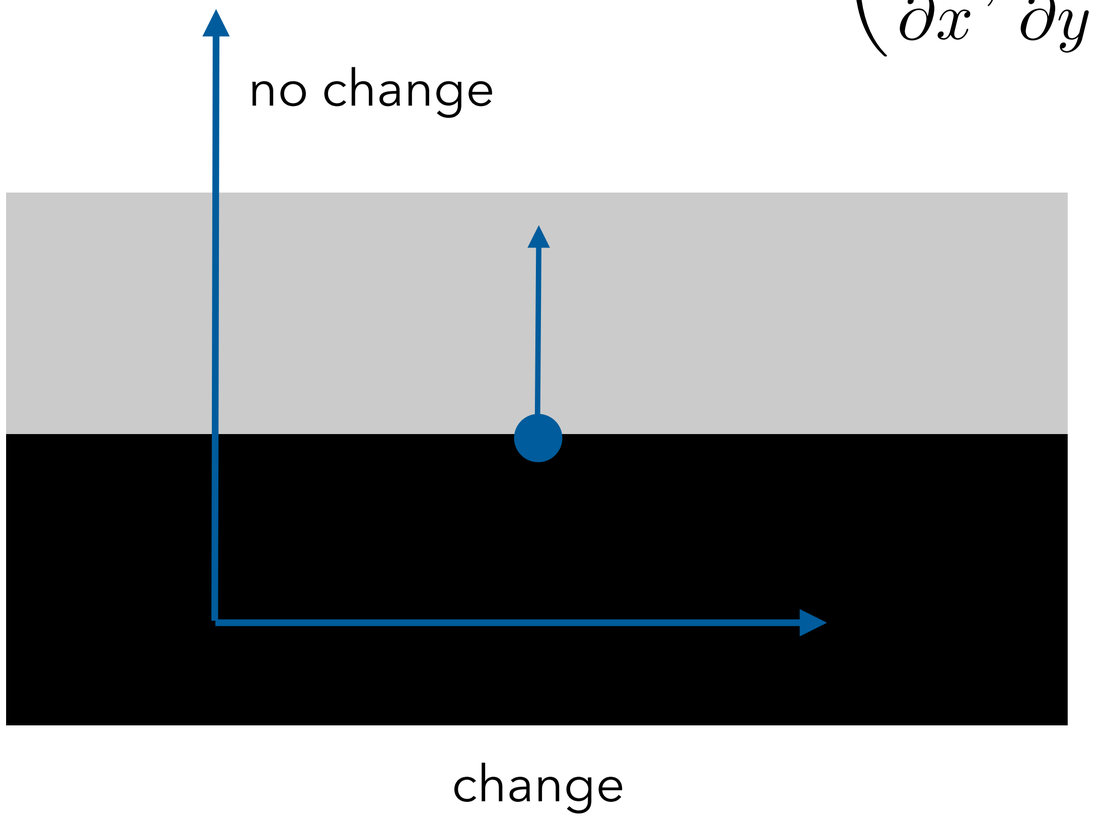
What is the gradient?

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k, 0)$$



What is the gradient?

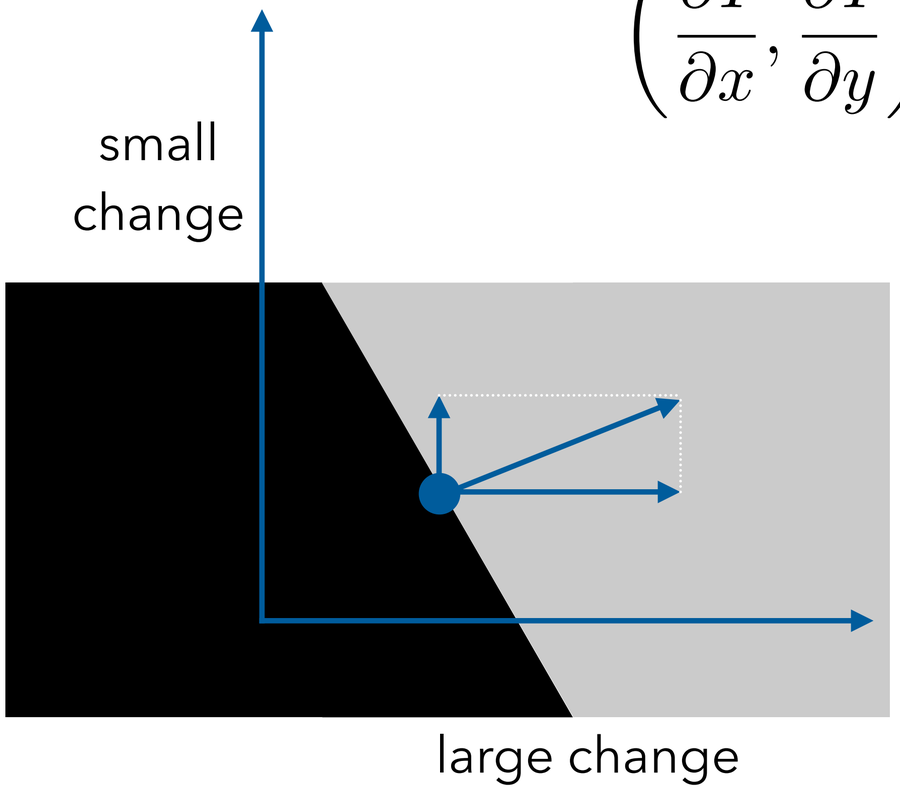
$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (0, k)$$



What is the gradient?

- gradient direction is *perpendicular* to edge
- gradient magnitude measures *edge strength*

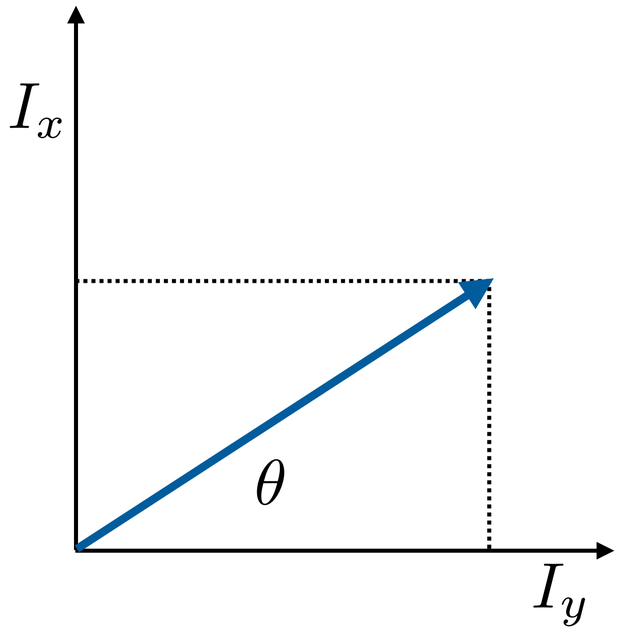
$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k_x, k_y)$$



2D Edge Detection

- Calculate derivative
  - use the **magnitude** of the gradient
  - the gradient is:

$$\nabla I = (I_x, I_y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$



  - the magnitude of the gradient is:

$$||\nabla I|| = \sqrt{I_x^2 + I_y^2}$$

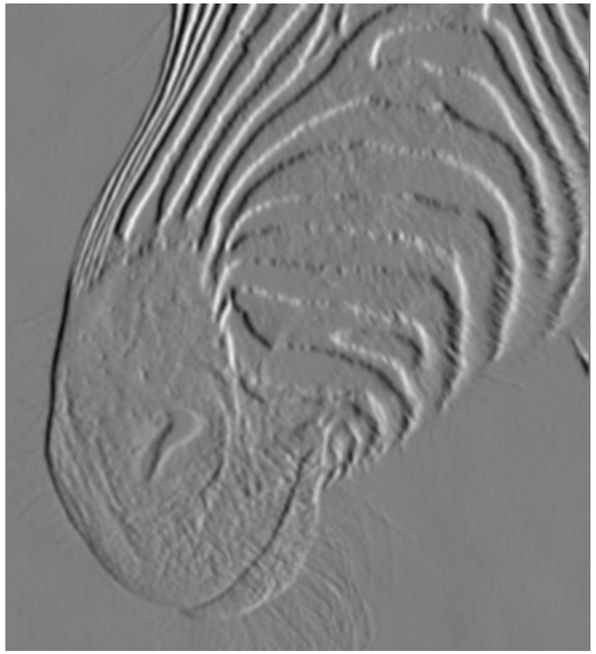
  - the direction of the gradient is:

$$\theta = \mathrm{atan}(I_y, I_x)$$

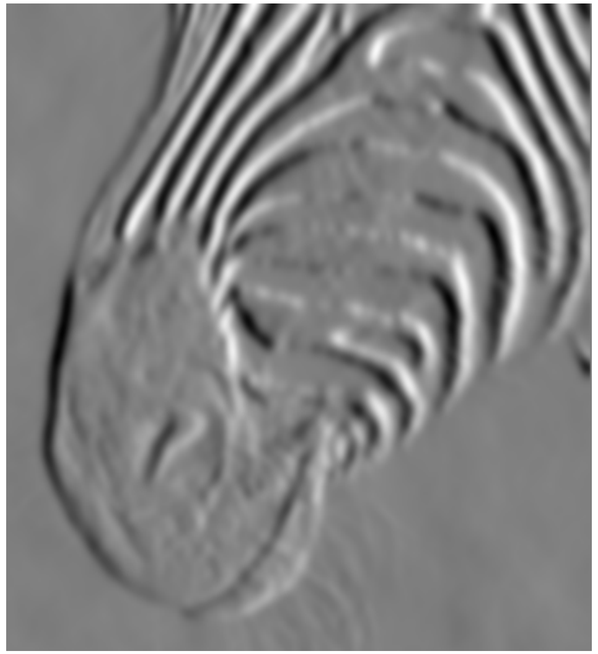
2D Edge Detection

- The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered
  - Note: strong edges persist across scales

1 pixel



3 pixels



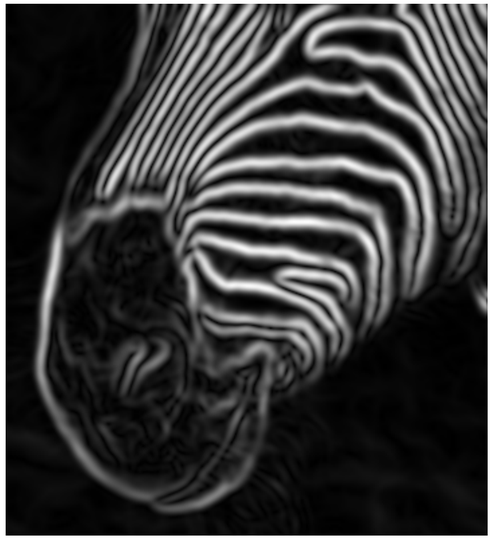
7 pixels

‘Optimal’ Edge Detection: Canny

- Assume:
  - linear filtering
  - additive i.i.d. Gaussian noise
- Edge detection should have:
  - *good detection:* filter responds to edge, not to noise
  - *good localization:* detected edge near true edge
  - *single response:* one per edge
- Then: Optimal detector is approximately *derivative of Gaussian* [Canny 1986]
- Detection/localization tradeoff:
  - more smoothing improves detection
  - and hurts localization

The Canny edge detector

original image



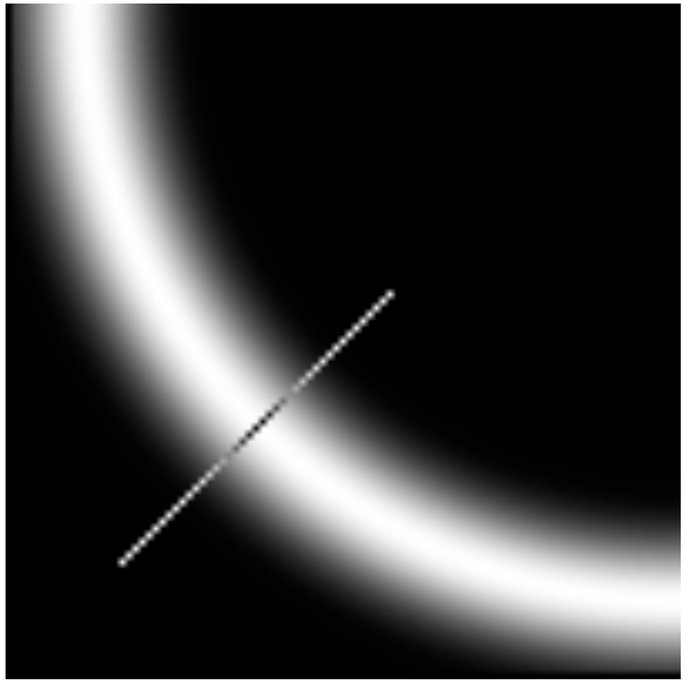
norm of the gradient

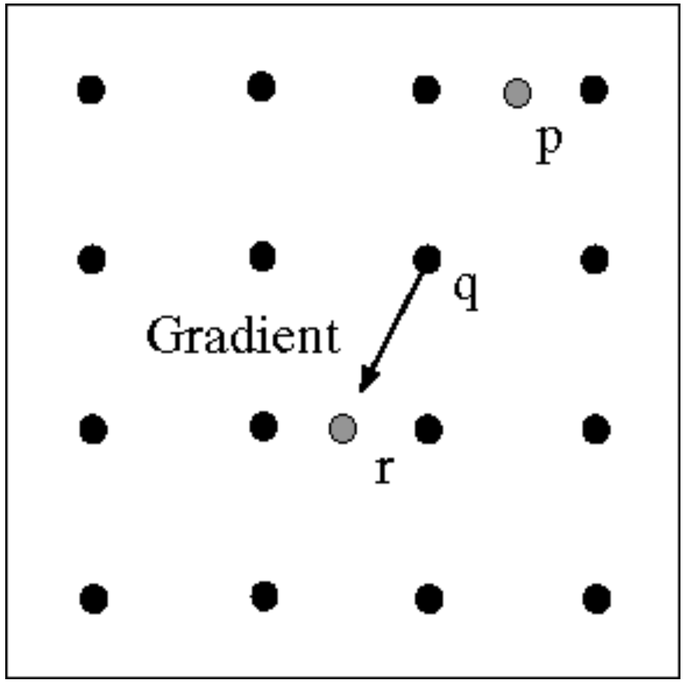
thinning: (non-maximum suppression)

thresholding

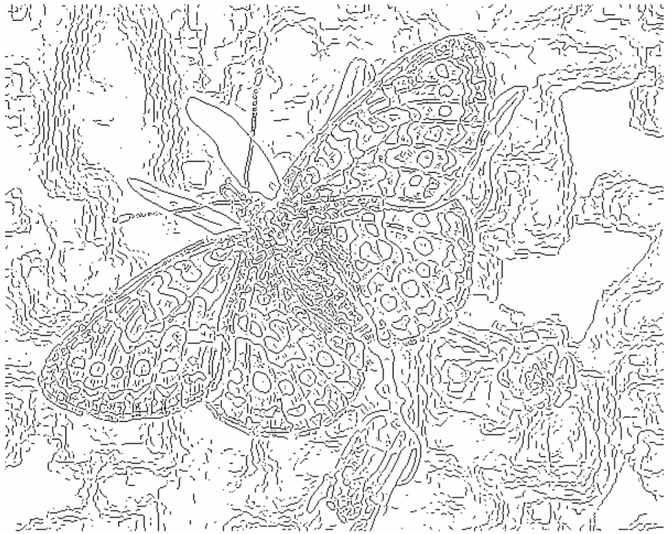
Non-maximum suppression

- Check if pixel is local maximum along gradient direction
  - choose the largest gradient magnitude along the gradient direction
  - requires checking interpolated pixels p and r





Example



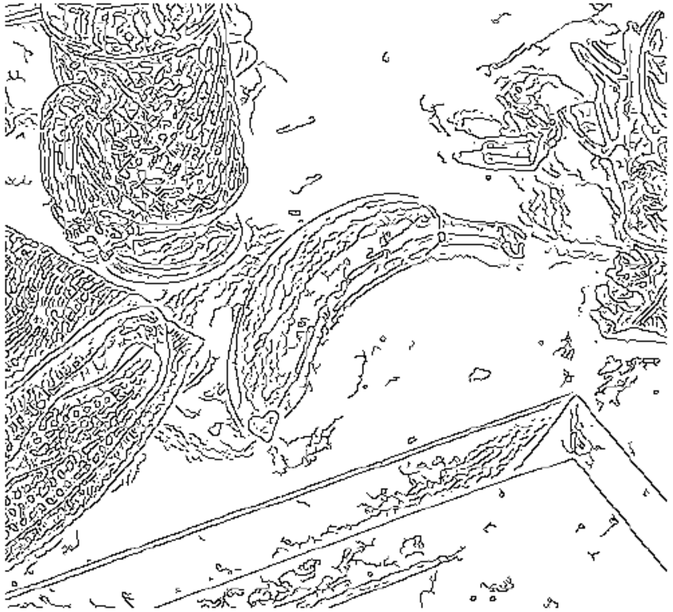
fine scale
high
threshold

coarse
scale
low
threshold

coarse
scale,
high
threshold

[Ponce & Forsyth]

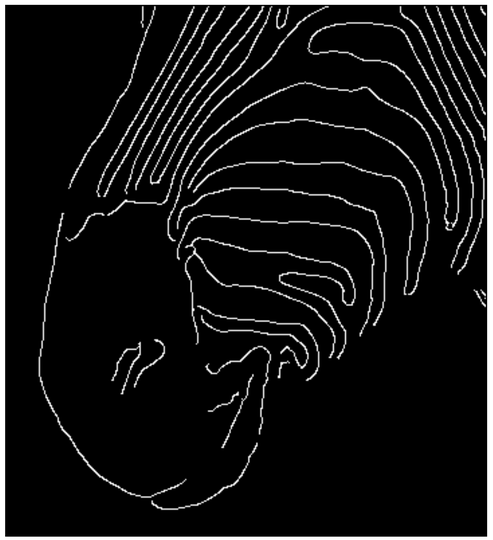
Line drawing vs. edge detection



Hysteresis

- Initially use high threshold to *start edges*
  - noise will typically have lower gradient magnitudes, thus not identified as an edge.
- Label all pixels as edge that
  - 1. – have gradient magnitudes above a second, lower threshold
  - 2. – are connected to a pixel above higher threshold

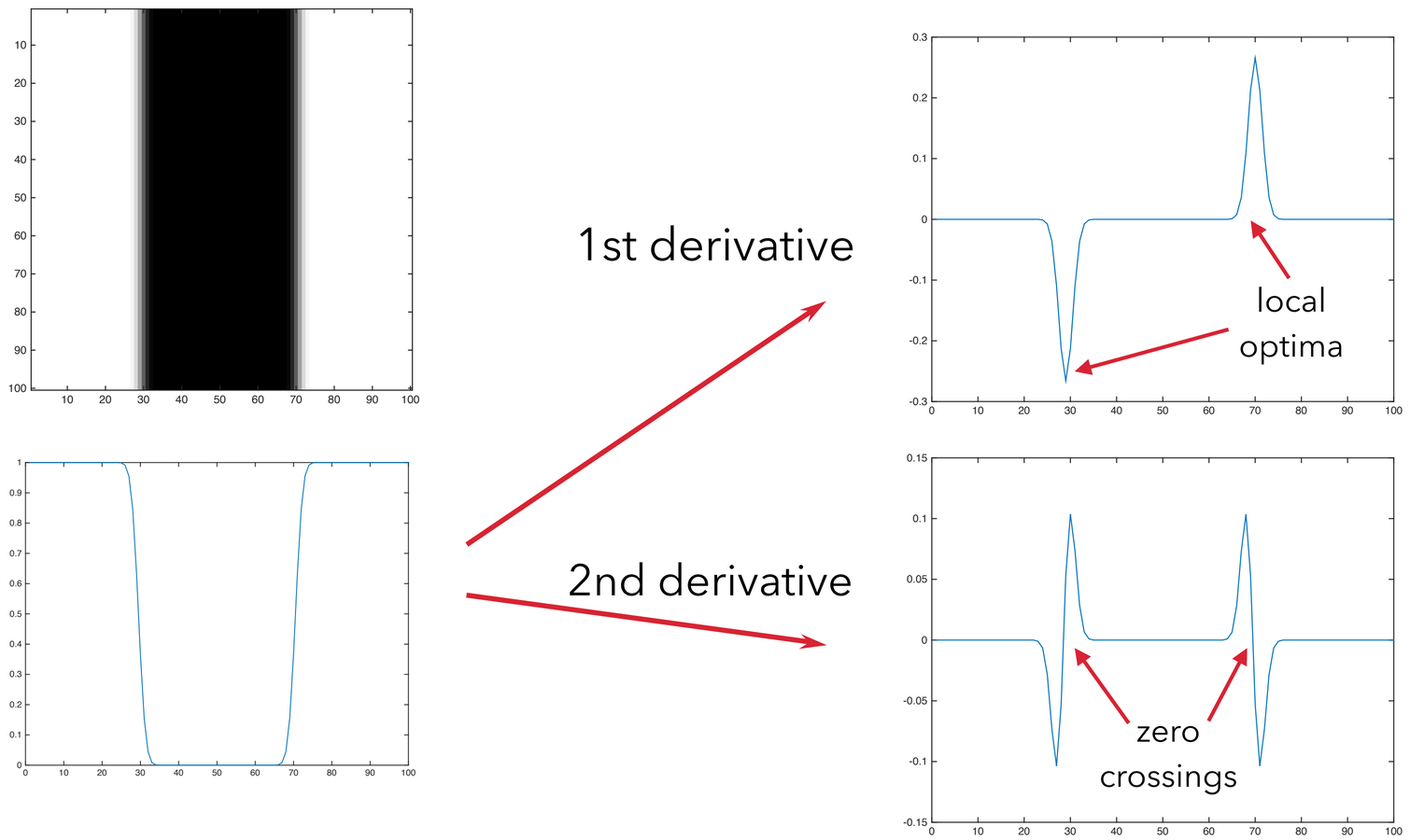
without hysteresis



with hysteresis

Edges & Derivatives…

- Recall:
  - the *zero-crossings of the second derivative* also tell us the location of edges



Compute 2nd order derivatives

- 1st derivative:

$$\frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h} \approx f(x+1)-f(x)$$

- 2nd derivative:

$$\frac{d^2}{dx^2}f(x) = \lim_{h\to 0}\frac{\frac{d}{dx}f(x)-\frac{d}{dx}f(x-h)}{h} \approx \frac{d}{dx}f(x)-\frac{d}{dx}f(x-1)$$

$$\approx f(x+1)-2f(x)+f(x-1)$$

- mask for
  - 1st derivative: $\boxed{1 \mid -1}$ 2nd derivative: $\boxed{1 \mid -2 \mid 1}$

The Laplacian

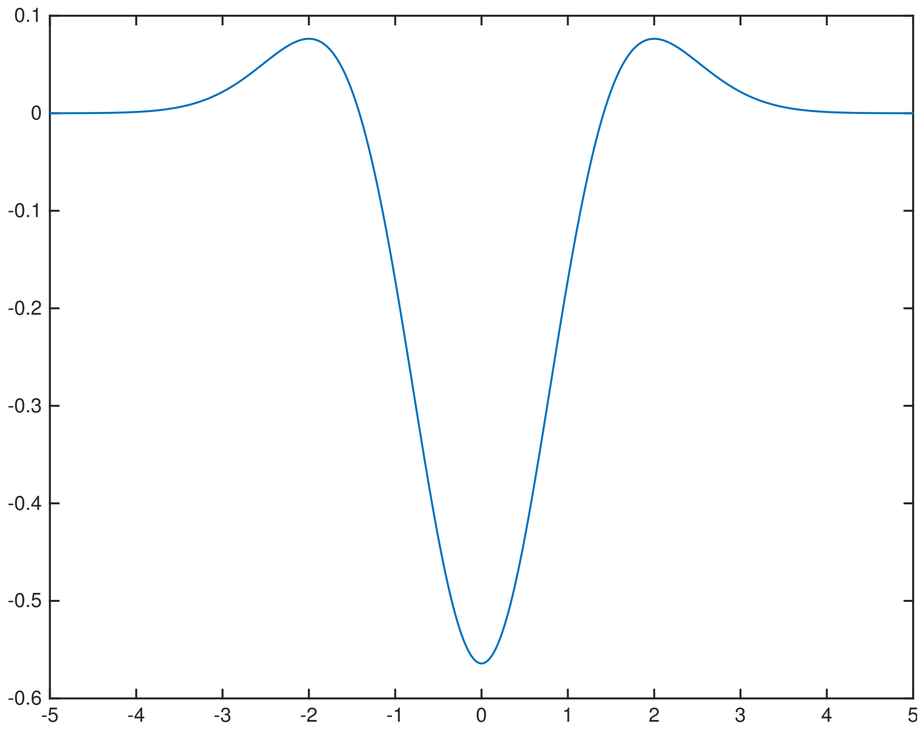
- The Laplacian:

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

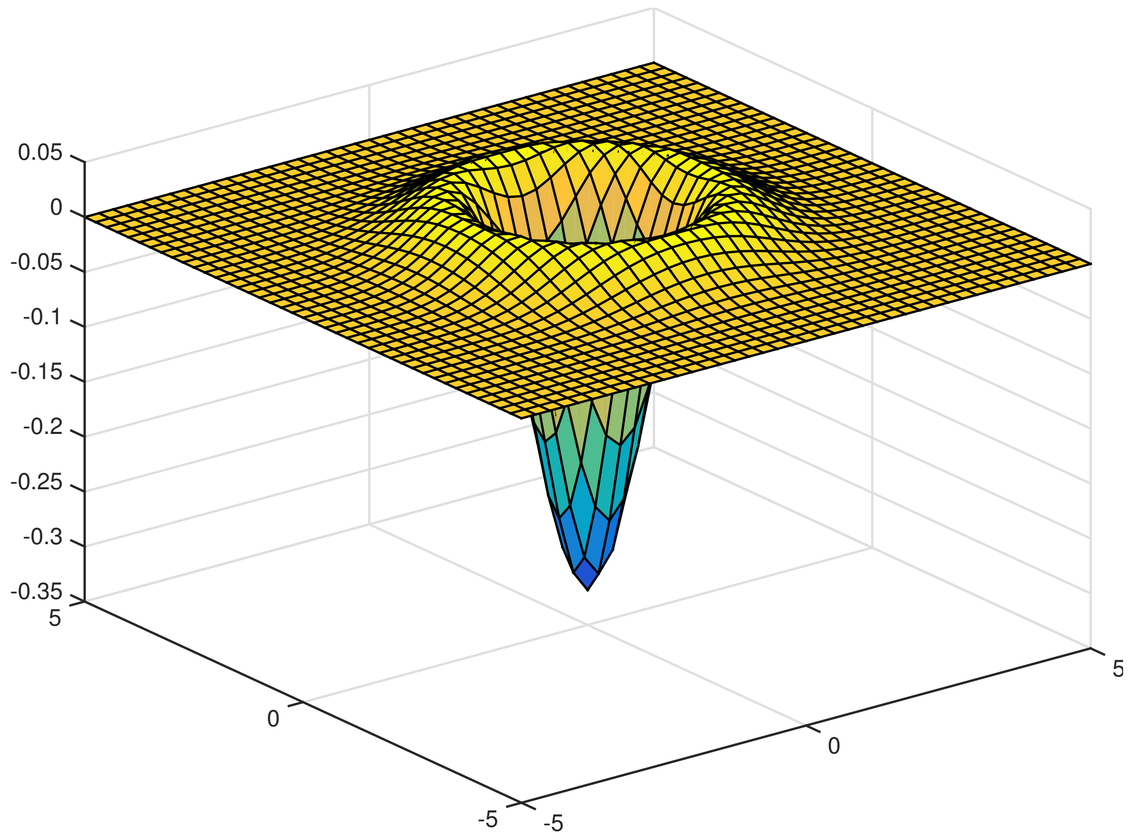
- just another linear filter
  - Laplacian of Gaussian (LoG): $\nabla^2 (G * I) = (\nabla^2 G) * I$

Second Derivative of Gaussian

- in 1D:



in 2D ("mexican hat function"):



1D edge detection

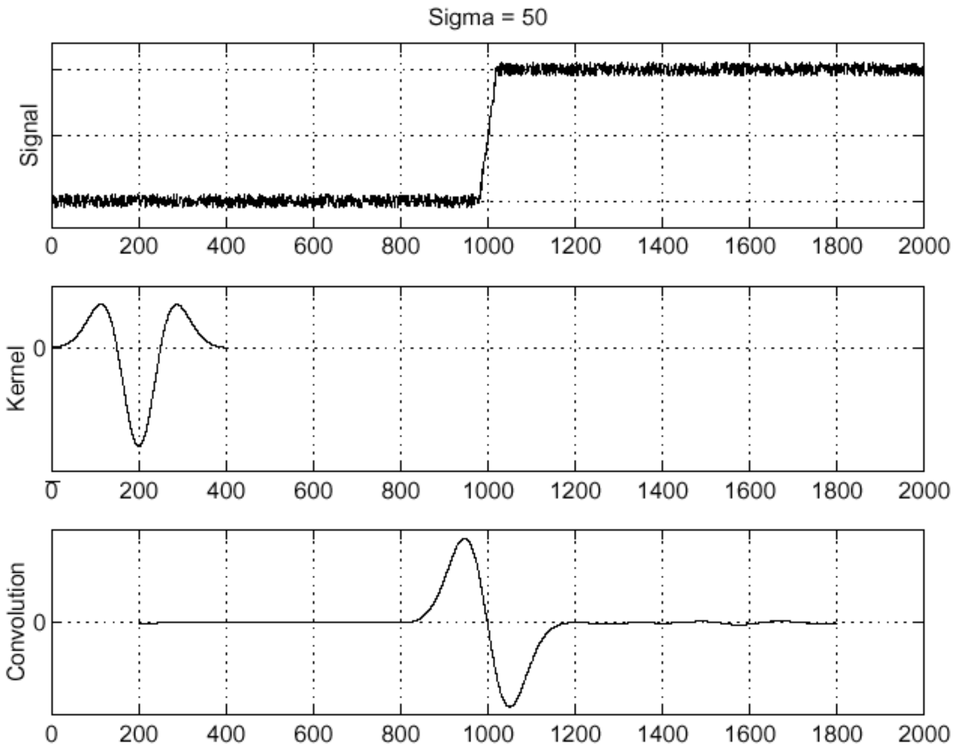
- using Laplacian

LoG - Laplacian of Gaussian operator

$$f$$

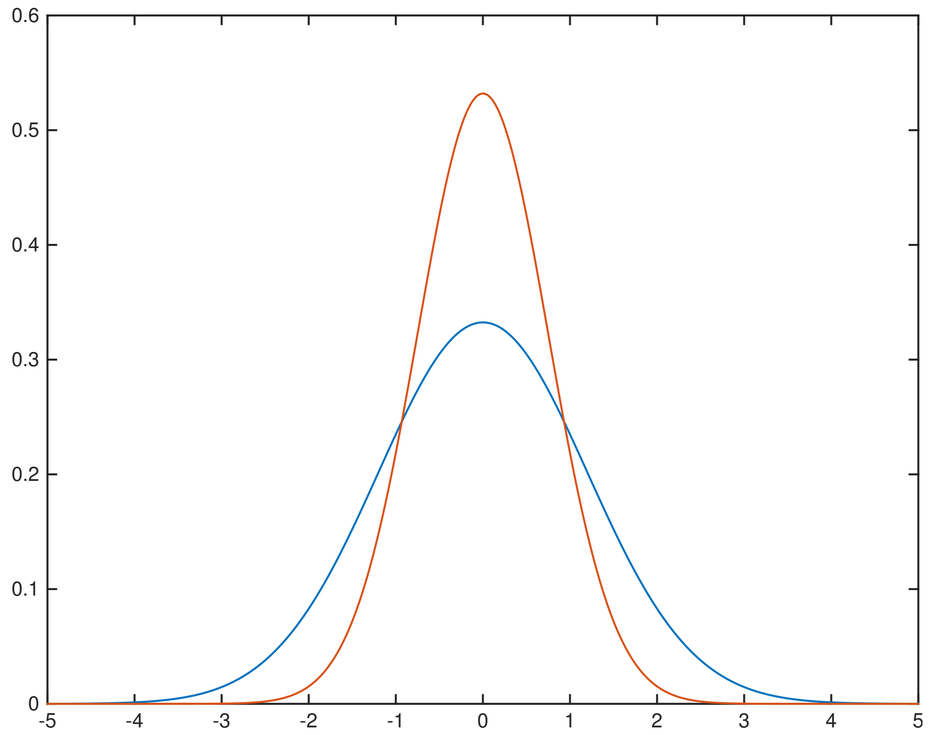
$$\frac{d^2}{dx^2} g$$

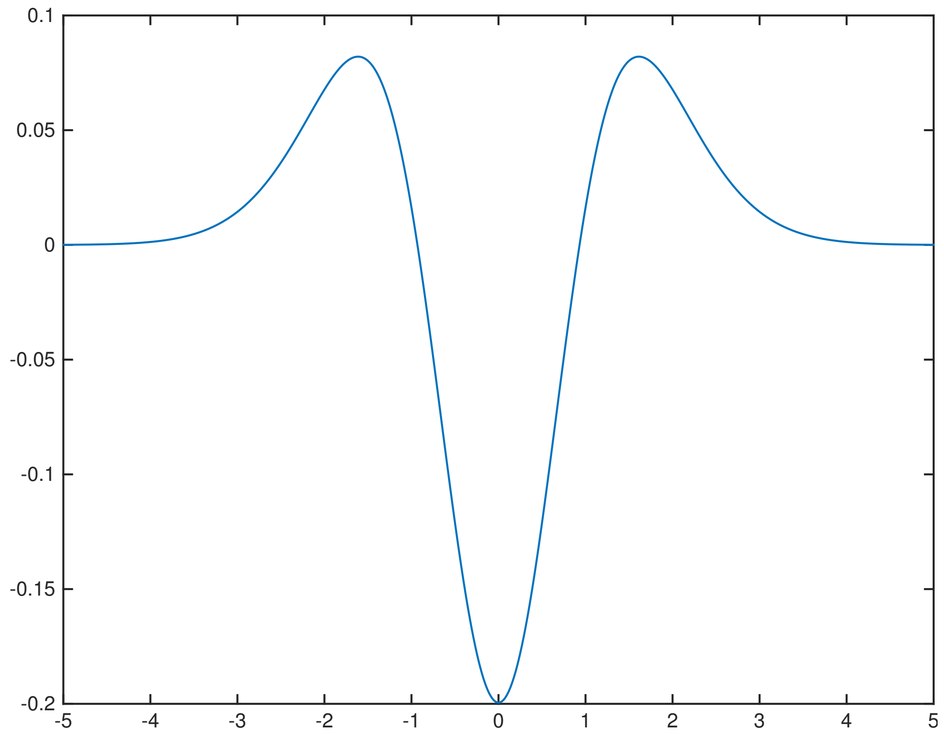
$$\left(\frac{d^2}{dx^2} g \right) \otimes f$$



Approximating the Laplacian

- Approximate the LoG by a DoG…
- Difference of Gaussians (DoG) at different scales:

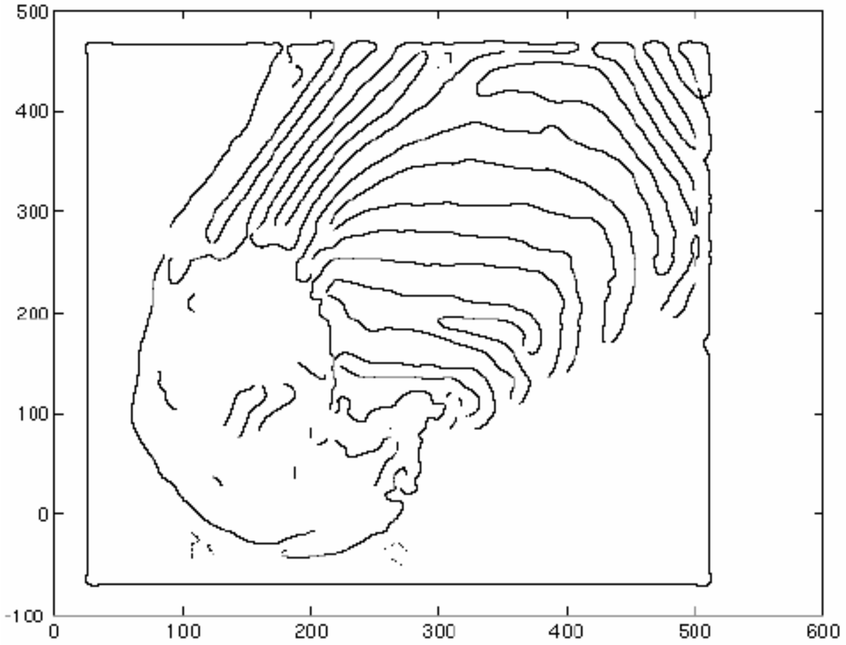




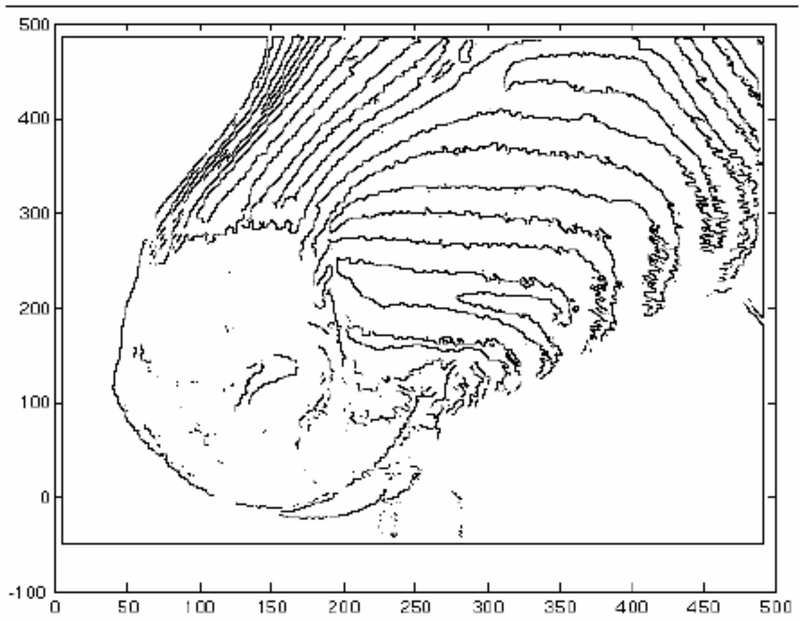
$\sigma_2 = 1.6 \cdot \sigma_1$ (other choices possible)

Edge Detection with Laplacian

$\sigma = 4$



$\sigma = 2$

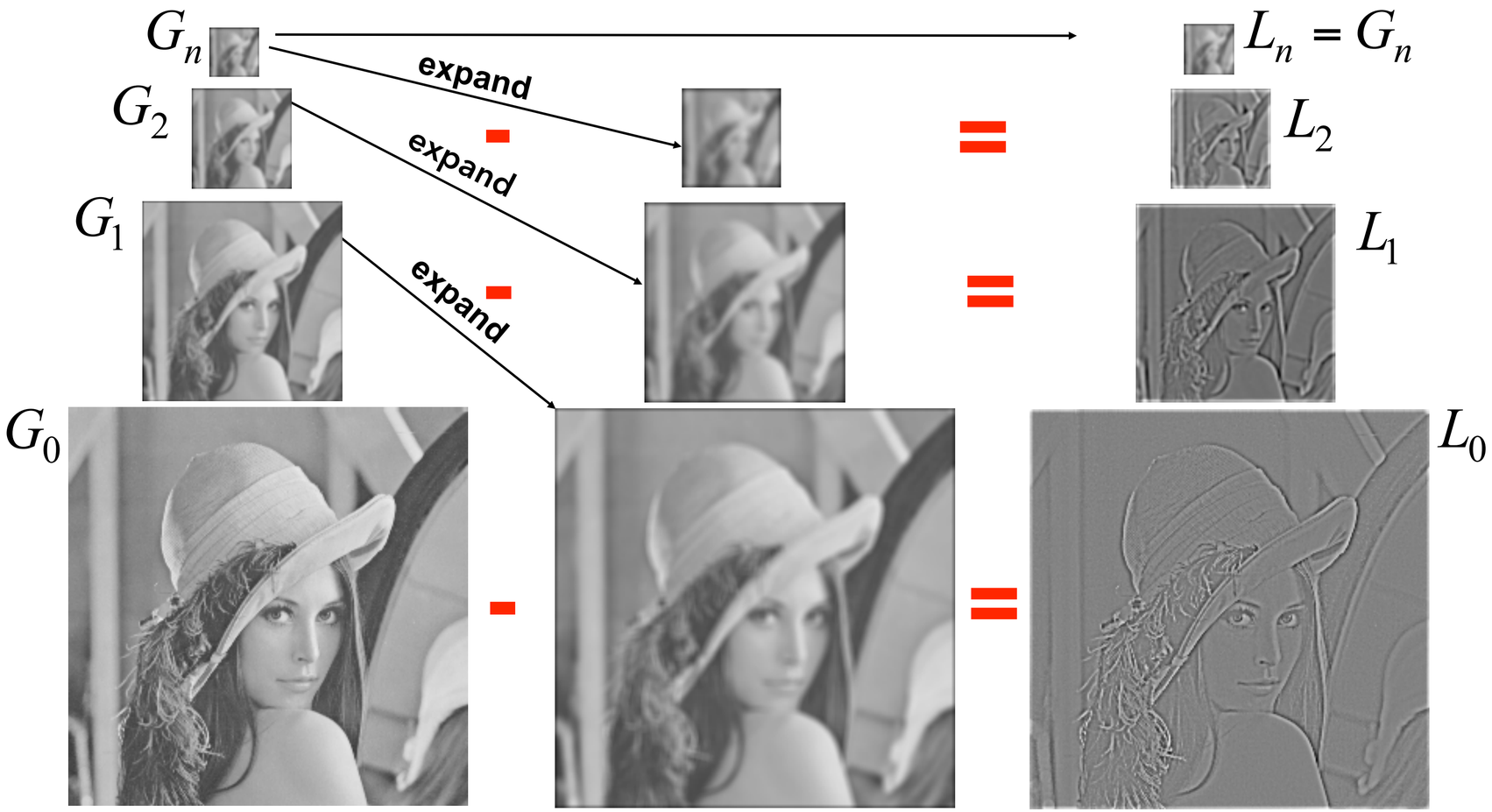


Today - Basics of Digital Image Processing

- Images
- Filtering
  - Linear filtering
  - Non-linear filtering & morphology
- Multi-scale image representation
  - Gaussian pyramid
  - Laplacian pyramid
- Edge detection
  - Motivation: Recognition using line drawings
  - Image derivatives (1st and 2nd order)

Laplacian pyramid

$$L_i = G_i - \text{expand}(G_{i+1}) \qquad G_i = L_i + \text{expand}(G_{i+1})$$



Gaussian Pyramid

Laplacian Pyramid

Laplacian pyramid

- The Laplacian pyramid provides a simple frequency decomposition into subbands.
- Each level / subband contains only image structure of a particular range of spatial frequencies:
  - E.g., the finest level contains all the high-frequency detail.
- Observation: We can get back the original image by reversing the decomposition.

- Example application:
  - *Image sharpening* by amplifying high-frequency components

Next: Interest Points

- Edges are important basic structures to find
- Next task:
  - Suppose we have two images depicting the same scene
  - How do we find corresponding points?
  - Which points should we even look at?

by Diva Sian

by swashford

[Szeliski & Seitz]

Next: Interest Points

- Edges are important basic structures to find
- Next task:
  - Suppose we have two images depicting the same scene
  - How do we find corresponding points?
  - Which points should we even look at?

harder:
different viewpoint

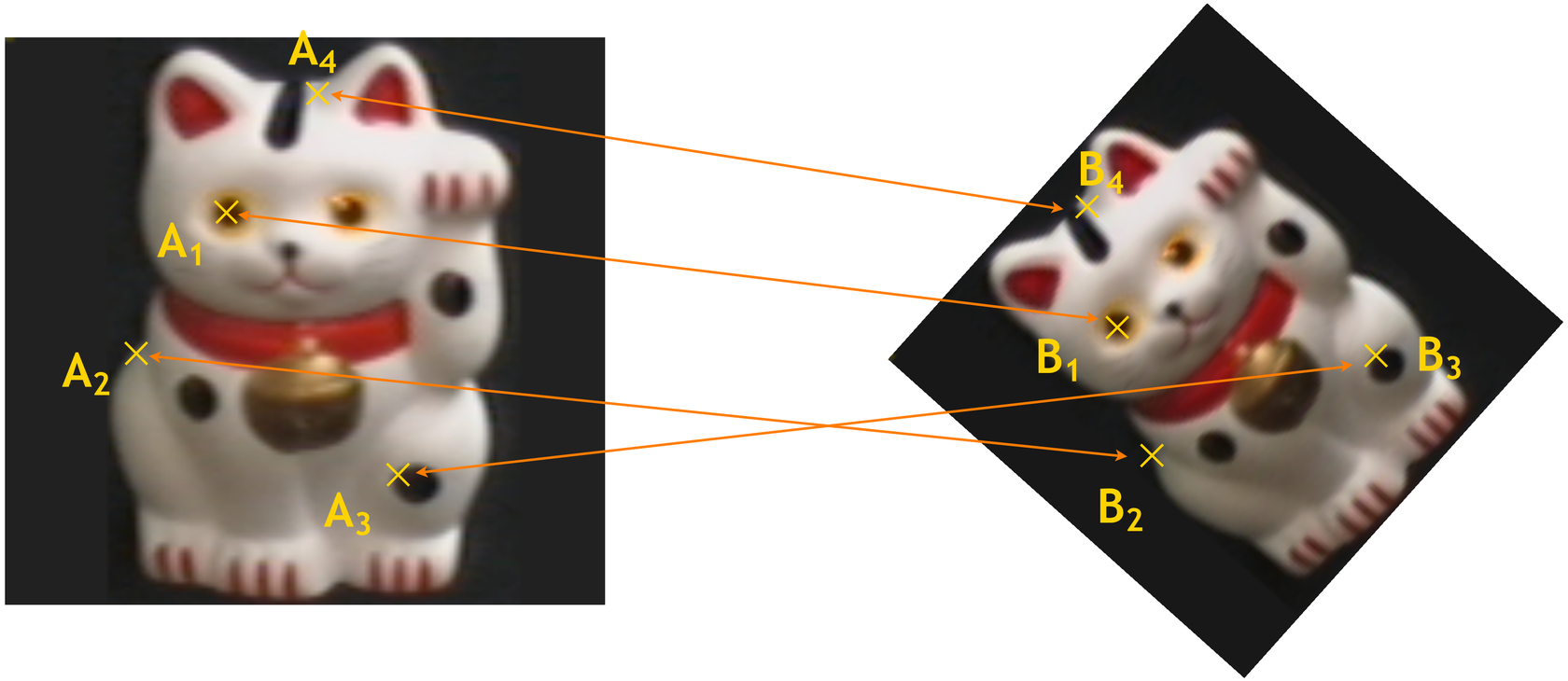
by Diva Sian

by scgbt

[Szeliski & Seitz]

Finding Corresponding Points

- General idea of detecting corresponding points:
  - 1. – Find important, i.e. “interesting” points in the images (today)
  - 2. – Find which interest points correspond to one another



What makes good interest points?

[Szeliski & Seitz]

Want uniqueness

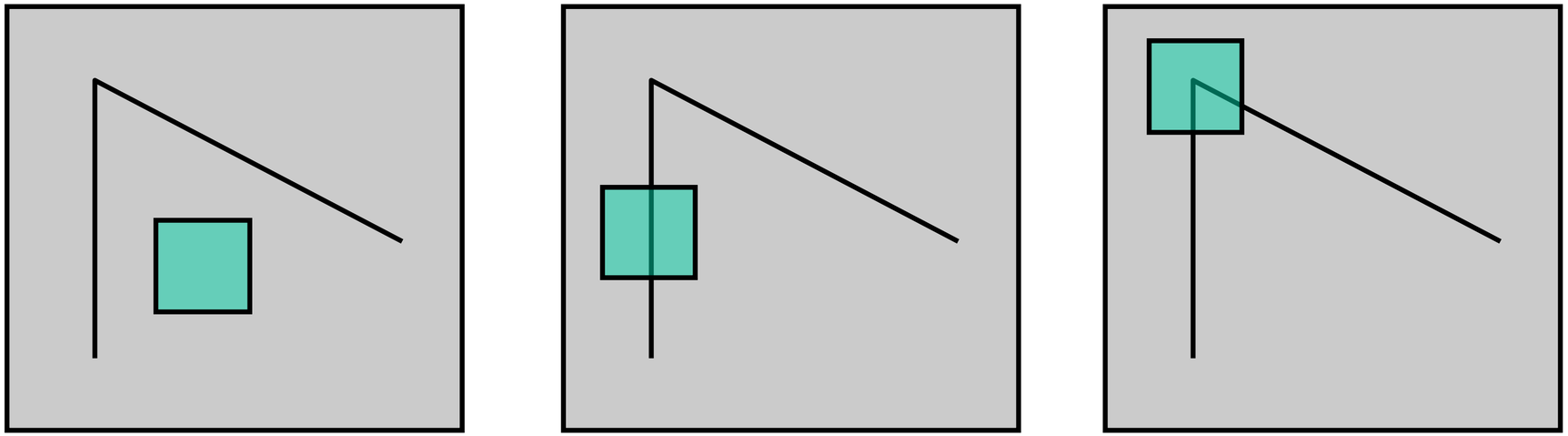
- Look for image regions that are unusual.
- Lead to unambiguous matches in other images

- How to define “unusual”?

[Szeliski & Seitz]

Local measures of uniqueness

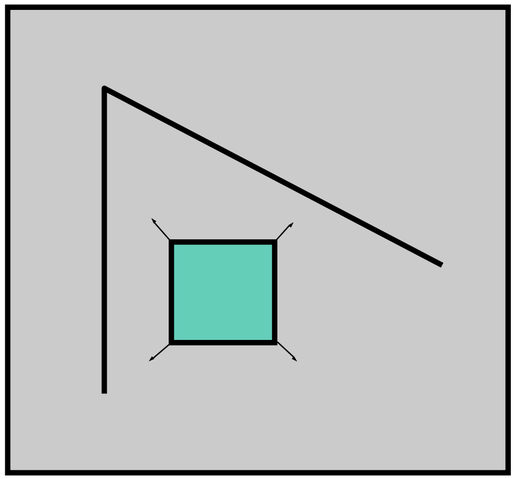
- Suppose we only consider a small window of pixels
  - What defines whether a feature is a good or bad candidate?



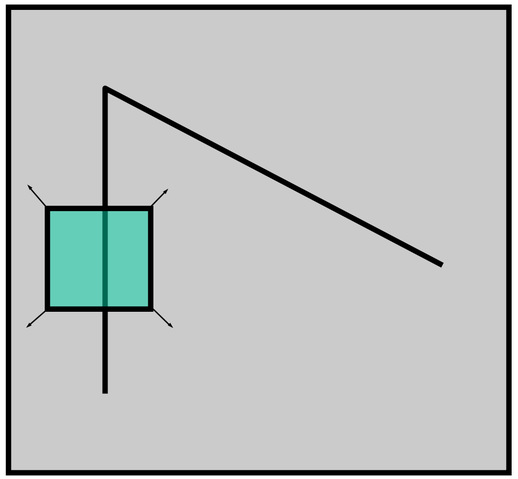
[Szeliski & Seitz]

Interest point detection

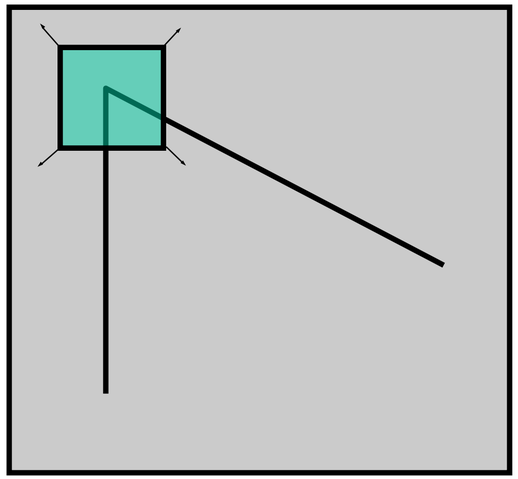
- Local measure of interest point uniqueness
  - How does the window change when you shift it?
  - Shifting the window in any direction causes a big change



"flat" region:
no change in all directions



"edge":
no change along the edge direction

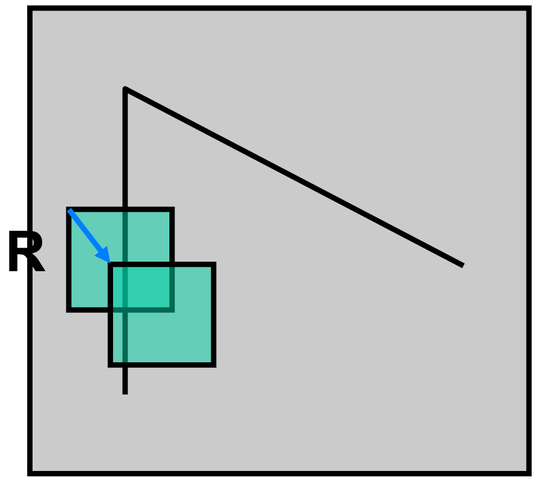


"corner":
significant change in all directions

[Szeliski & Seitz]

Interest point detection: Derivation

- Consider shifting the window R by (u,v)
  - How do the pixels in R change?
  - Compare each pixel before and after by *summing up the squared differences* (SSD)
  - This defines an SSD "error"/energy of $E(u,v)$:



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

[Szeliski & Seitz]

Can we approximate this?

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

Taylor series approximation:

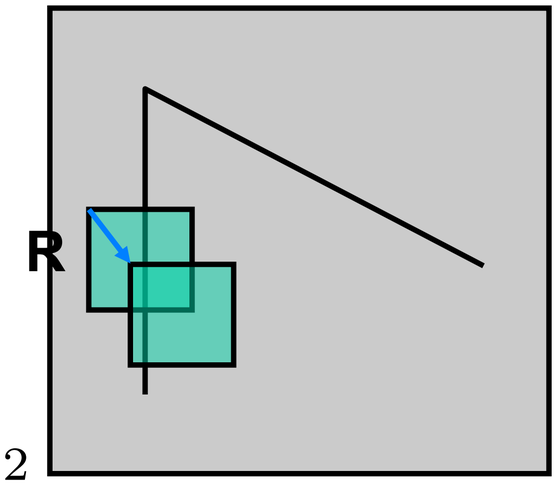
$$I(x,y) + u\frac{\partial}{\partial x} I(x,y) + v\frac{\partial}{\partial y} I(x,y) + \epsilon(u^2, v^2)$$

Approximation error

$$E(u,v) \approx \sum_{(x,y)\in R} \left(u\frac{\partial}{\partial x} I(x,y) + v\frac{\partial}{\partial y} I(x,y) \right)^2$$

Interest point detection: Derivation

- Consider shifting the window R by (u,v)
  - How do the pixels in R change?
  - Compare each pixel before and after by *summing up the squared differences* (SSD)
  - This defines an SSD "error"/energy of $E(u,v)$:



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

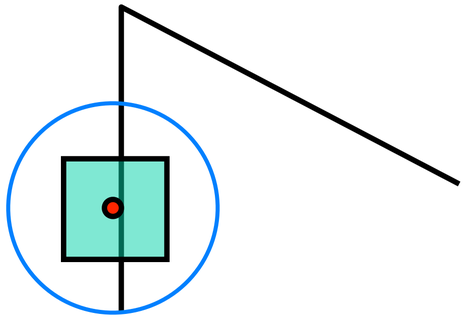
$$\approx \sum_{(x,y)\in R} \big(u \cdot I_x(x,y) + v \cdot I_y(x,y)\big)^2$$

$$= \sum_{(x,y)\in R} (u,v)\nabla I(x,y)\nabla I(x,y)^T \begin{pmatrix} u \\ v \end{pmatrix} \qquad \nabla I = \begin{pmatrix} I_x \\ I_y \end{pmatrix}$$

[Szeliski & Seitz]

Interest point detection: Derivation

$$E(u,v) \approx (u,v) \underbrace{\left[\sum_{(x,y)\in R} \nabla I(x,y) \nabla I(x,y)^T \right]}_{H} \begin{pmatrix} u \\ v \end{pmatrix}$$



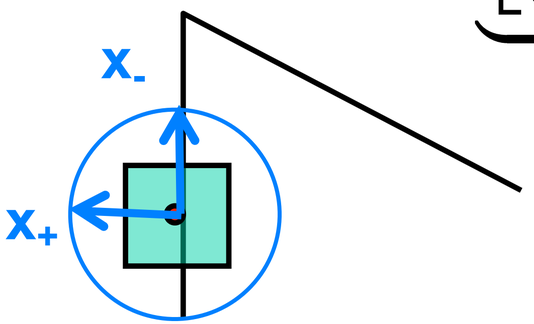
structure tensor

- Suppose you can move the center of the green window to anywhere on the blue unit circle
  - Which directions will result in the largest and smallest energy values E?
- We can find these directions by looking at the eigenvectors of H
  - Recall homogeneous least squares!

[Szeliski & Seitz]

Interest point detection: Derivation

$$E(u,v) \approx (u,v) \underbrace{\left[\sum_{(x,y)\in R} \nabla I(x,y) \nabla I(x,y)^T \right]}_{H} \begin{pmatrix} u \\ v \end{pmatrix}$$



structure tensor

- Eigenvalues and eigenvectors of H
  - Define shifts with the smallest and largest change (E value)
  - x_+ = direction of largest increase in E.
  - λ_+ = amount of increase in direction x_+
  - x_- = direction of smallest increase in E.
  - λ_- = amount of increase in direction x_-

$$\begin{aligned} Hx_+ &= \lambda_+ x_+ \\ Hx_- &= \lambda_- x_- \end{aligned}$$

[Szeliski & Seitz]

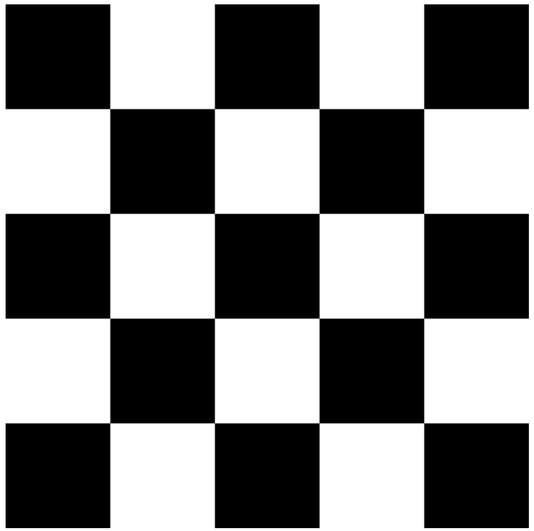
Interest point detection

- How are λ_+, x_+, λ_-, and x_- relevant for interest point detection?
  - What's our feature scoring function?

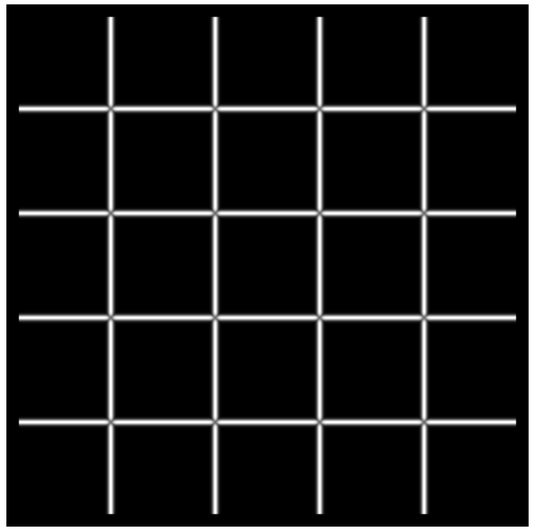
[Szeliski & Seitz]

Interest point detection

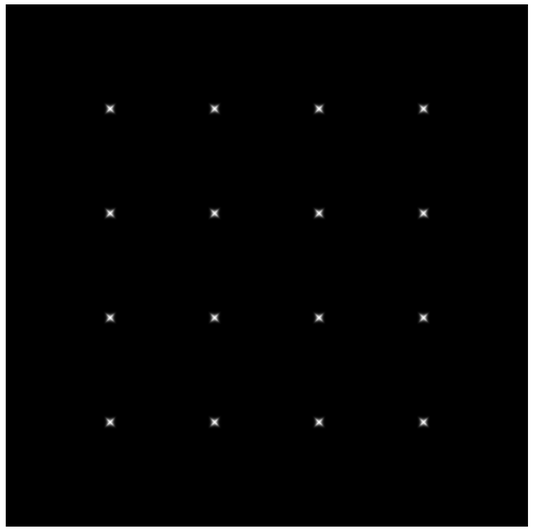
- How are λ_+, x_+, λ_-, and x_- relevant for interest point detection?
  - What's our feature scoring function?
- Want $E(u,v)$ to be large for small shifts in all directions
  - the minimum of $E(u,v)$ should be large, over all unit vectors $[u\ v]$
  - this minimum is given by the smaller eigenvalue (λ_-) of H



I



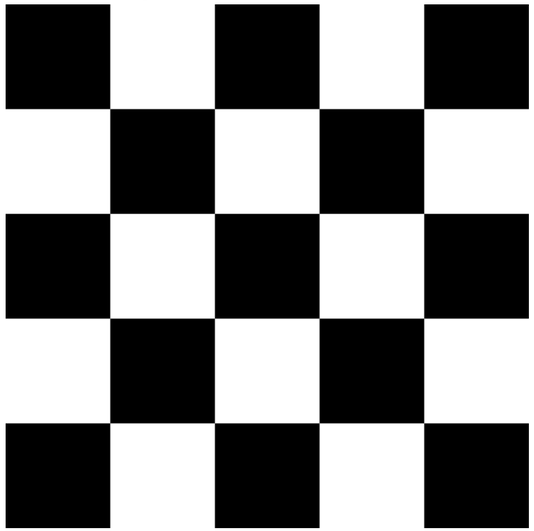
λ_+



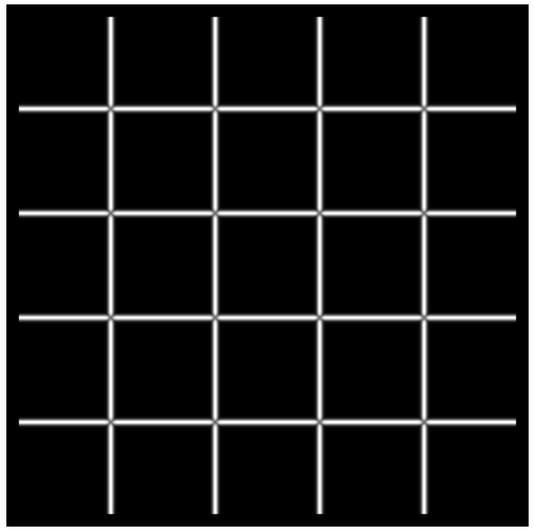
λ_-

Interest point detection

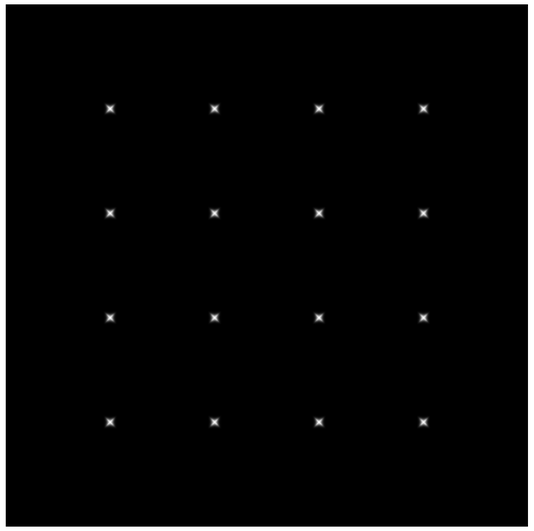
- Here's what you do:
  1. Compute the gradient at each point in the image
  2. Create the structure tensor H from the entries in the gradient
  3. Compute the eigenvalues.
  4. Find points with large response (λ_- > threshold)
  5. Choose those points where λ_- is a local maximum as interest points



I



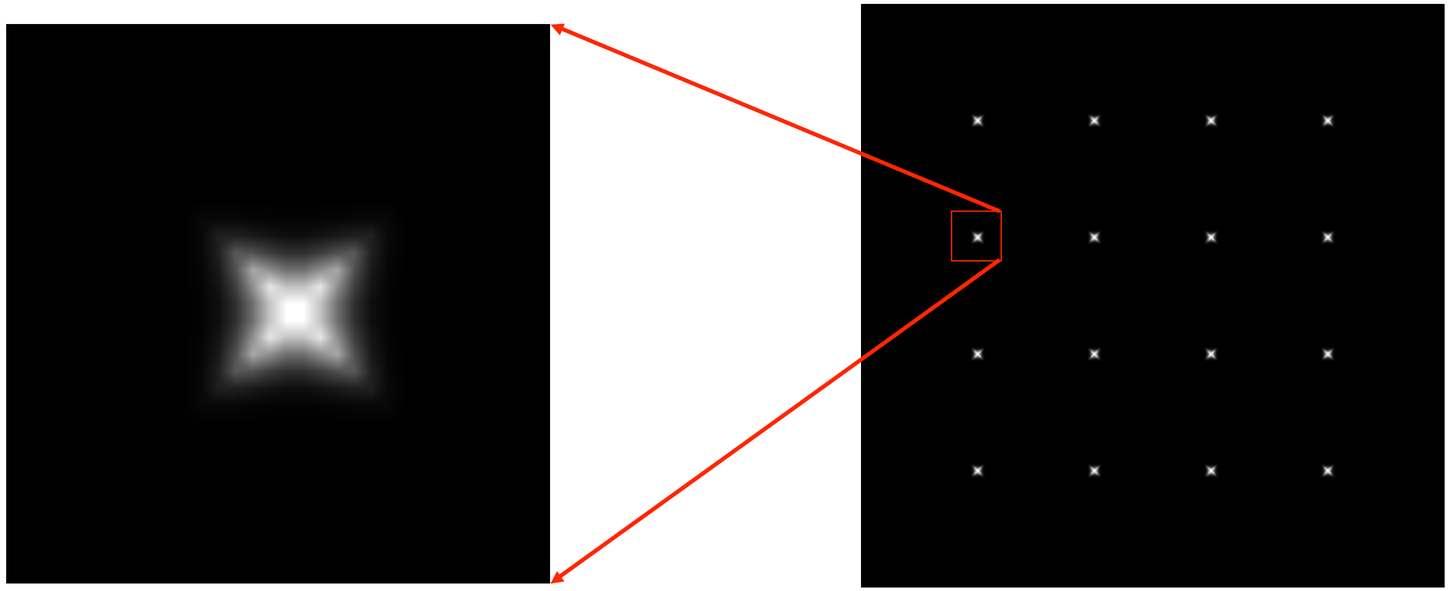
λ_+



λ_-

Interest point detection

- Here's what you do
  1. Compute the gradient at each point in the image
  2. Create the structure tensor H from the entries in the gradient
  3. Compute the eigenvalues.
  4. Find points with large response ($\lambda_$ > threshold)
  5. Choose those points where $\lambda_$ is a local maximum as interest points



The Harris operator

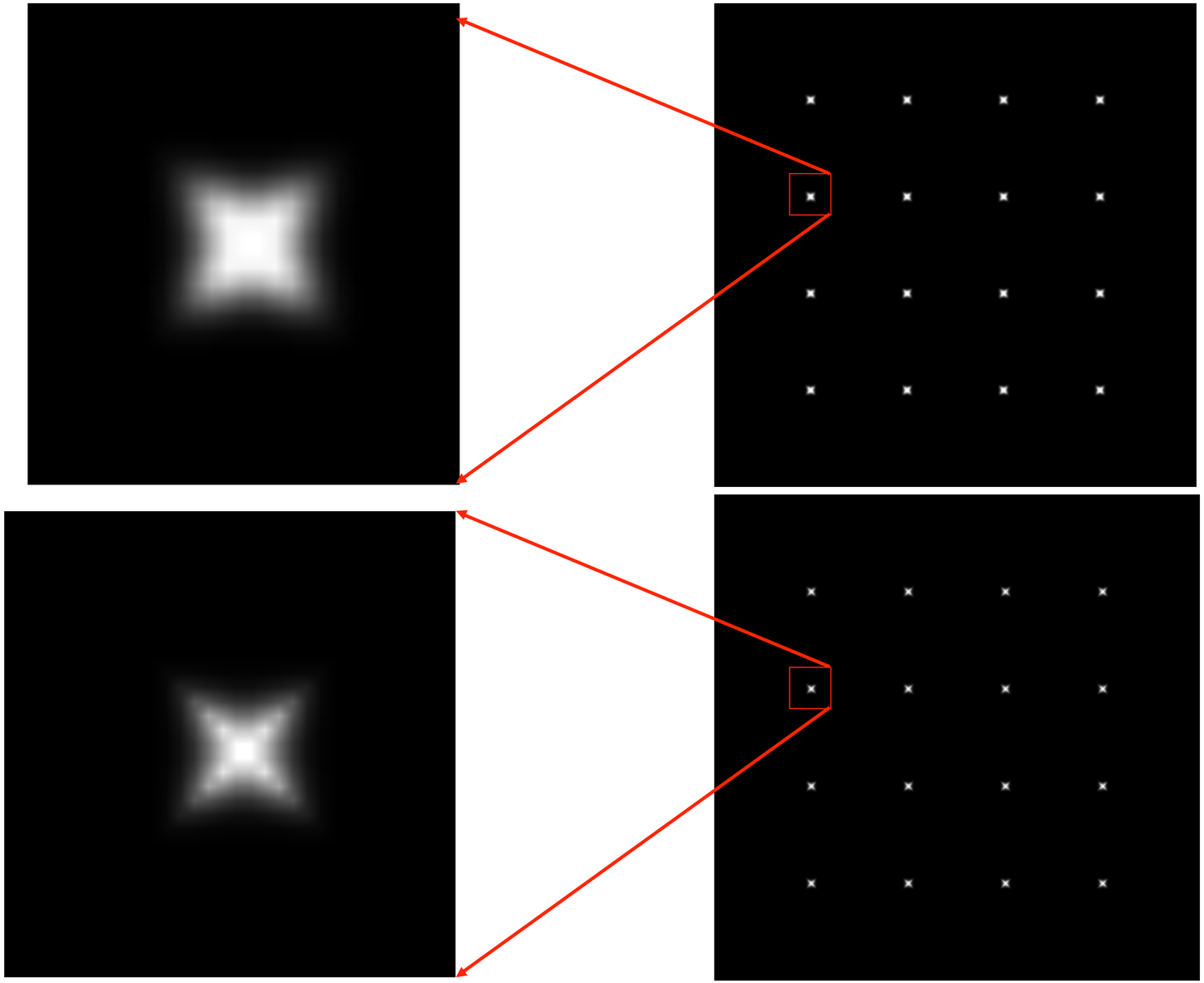
- λ_- is a variant of the "Harris operator" for interest point detection

$$
\begin{aligned}
f &= \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \\
&= \det(H) - \alpha \cdot \mathrm{trace}(A)^2
\end{aligned}
$$

  - The trace is the sum of the diagonals, i.e., $\mathrm{trace}(H) = h_{11} + h_{22}$
  - Very similar to λ_- but less expensive (no square root)
  - Called the "Harris Corner Detector" or "Harris Operator"
  - Lots of other detectors, this is one of the most popular

[Szeliski & Seitz]

The Harris operator



Harris operator

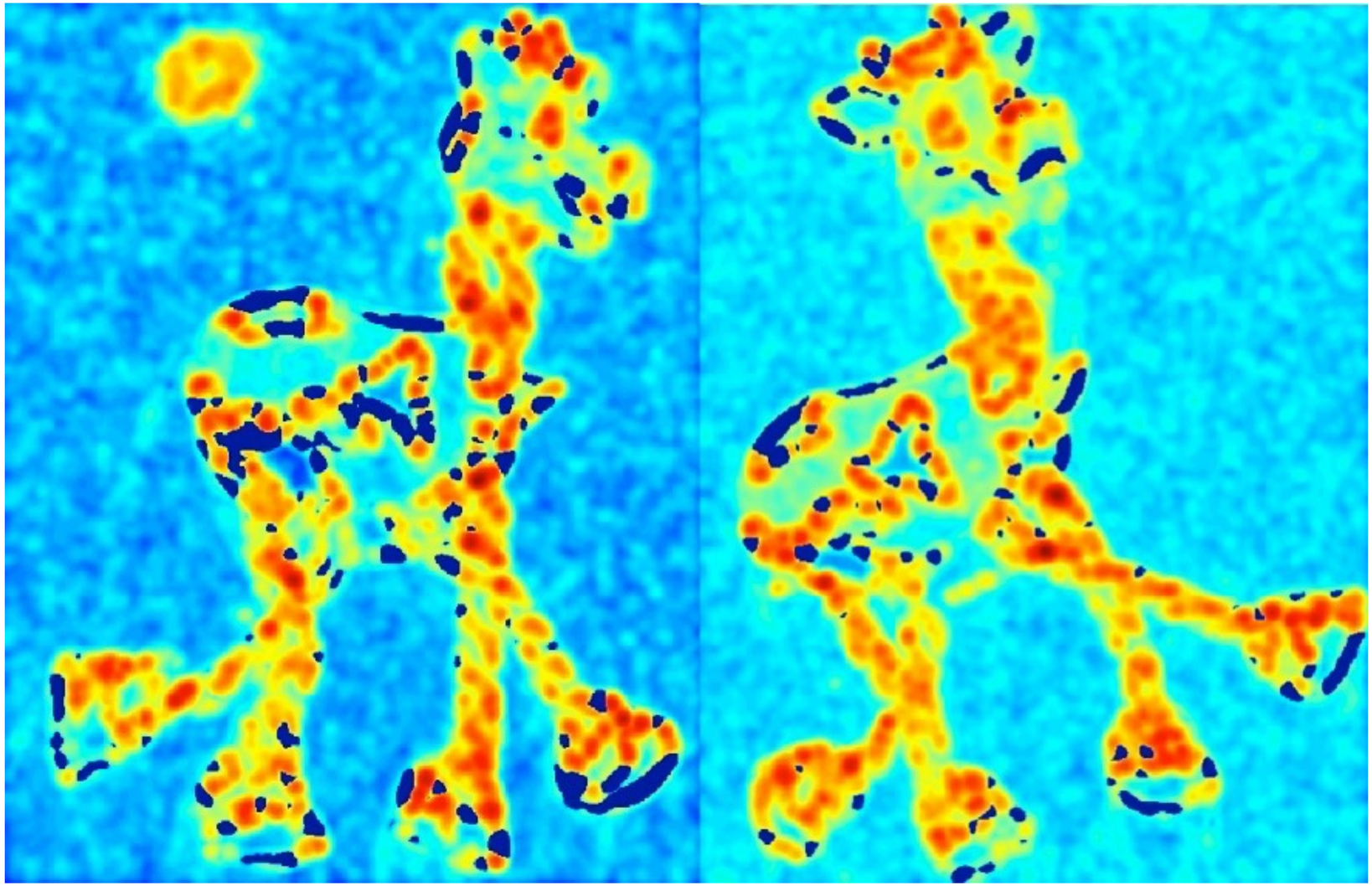
λ_-

[Szeliski & Seitz]

Harris detector example

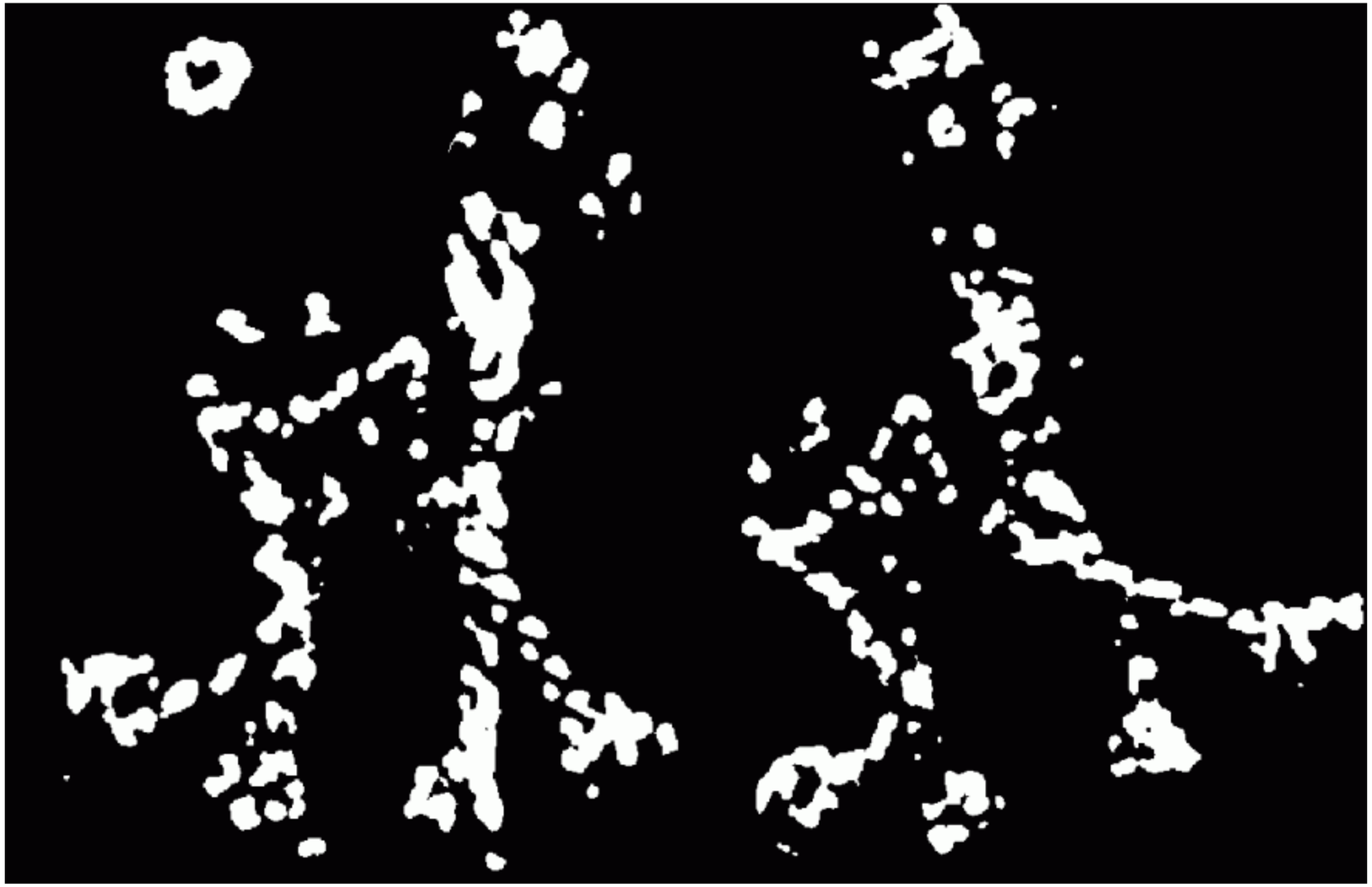
[Szeliski & Seitz]

f value (red high, blue low)



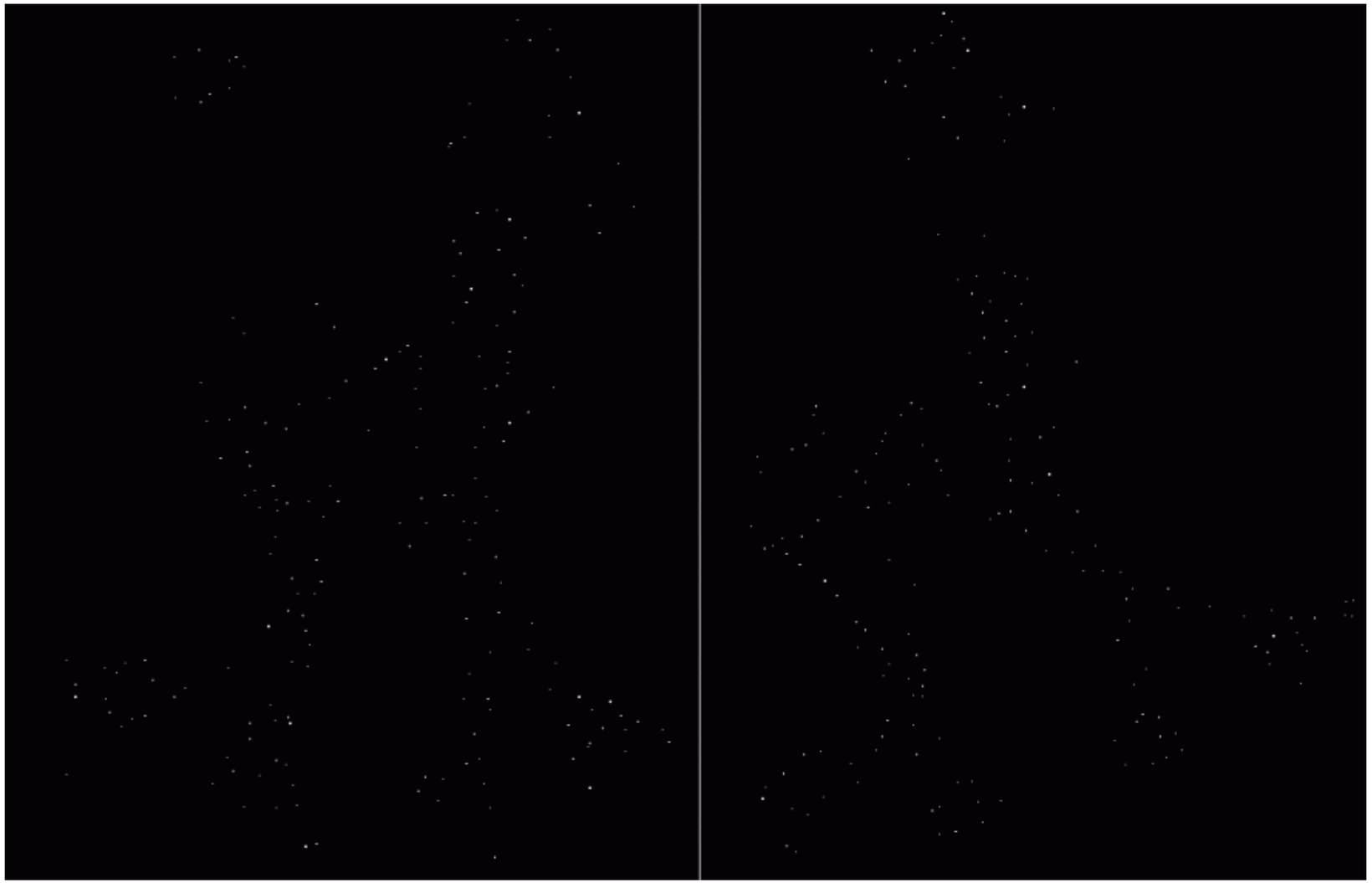
[Szeliski & Seitz]

Threshold (f > value)



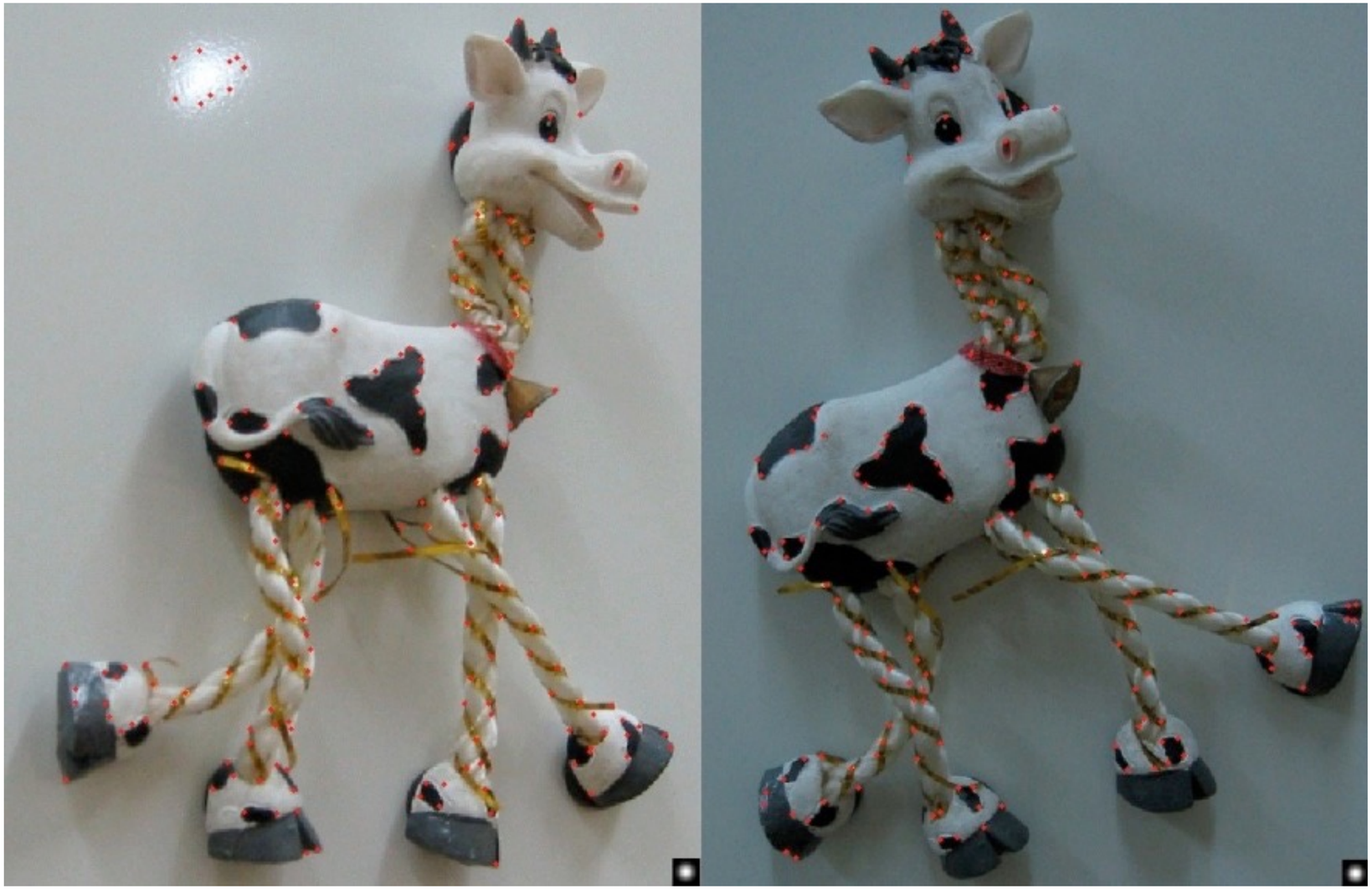
[Szeliski & Seitz]

Find local maxima of f



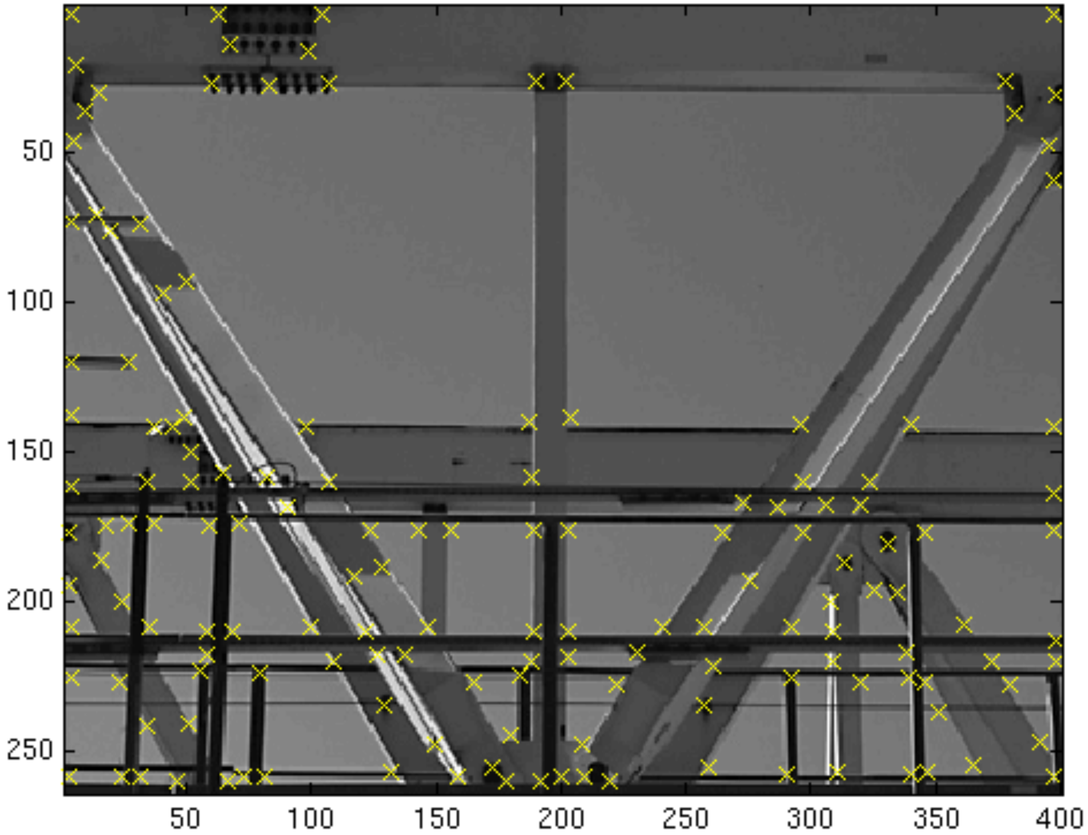
[Szeliski & Seitz]

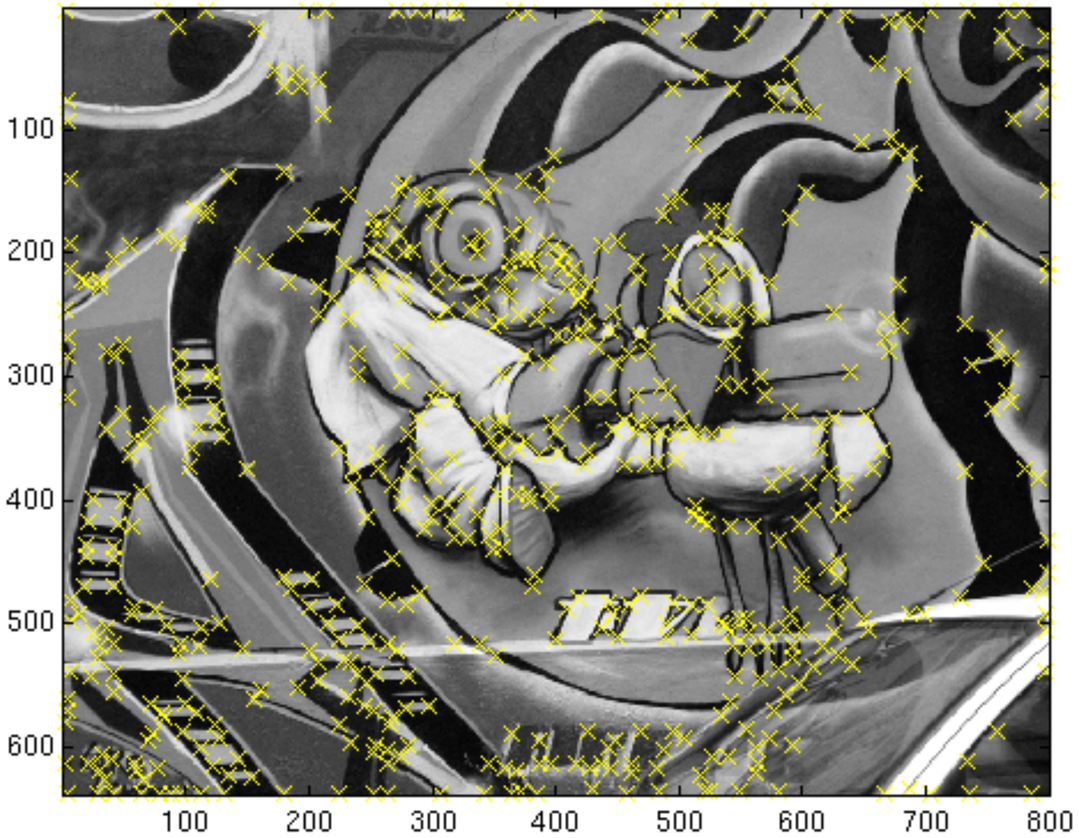
Harris features (in red)



[Szeliski & Seitz]

Harris interest points



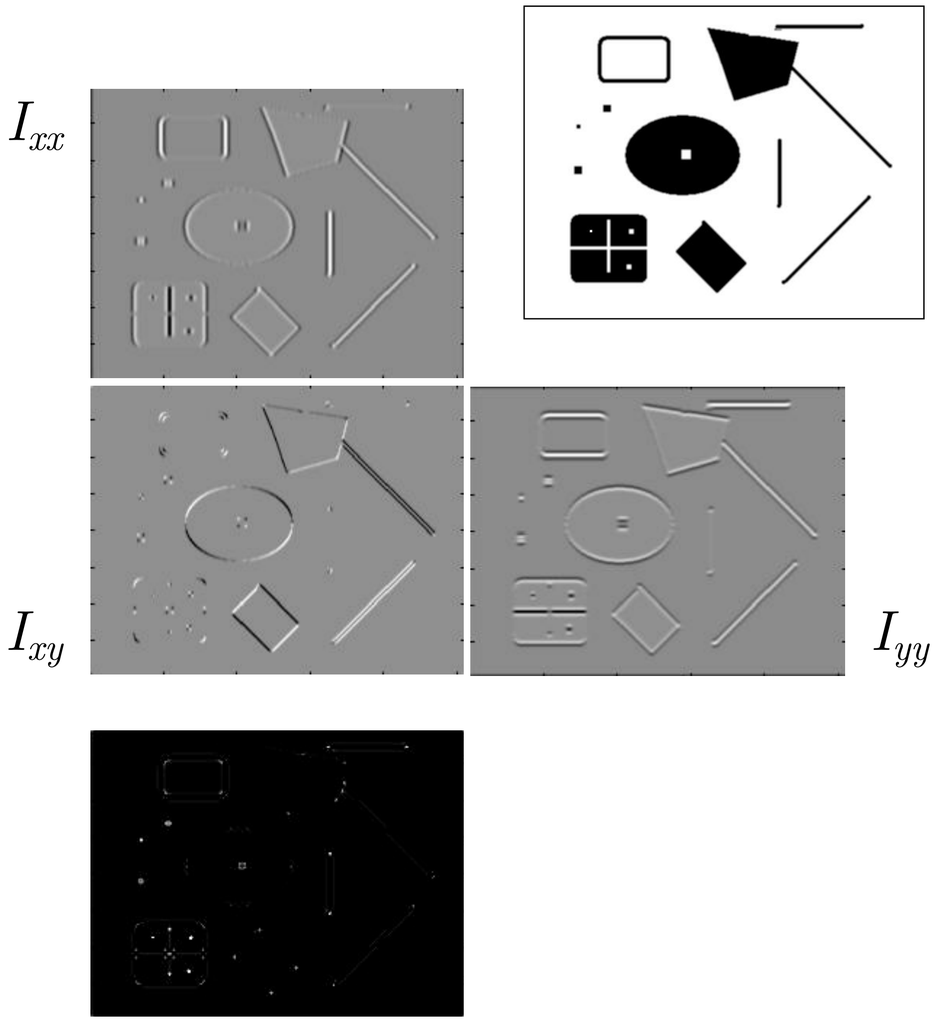


Other interest point detectors

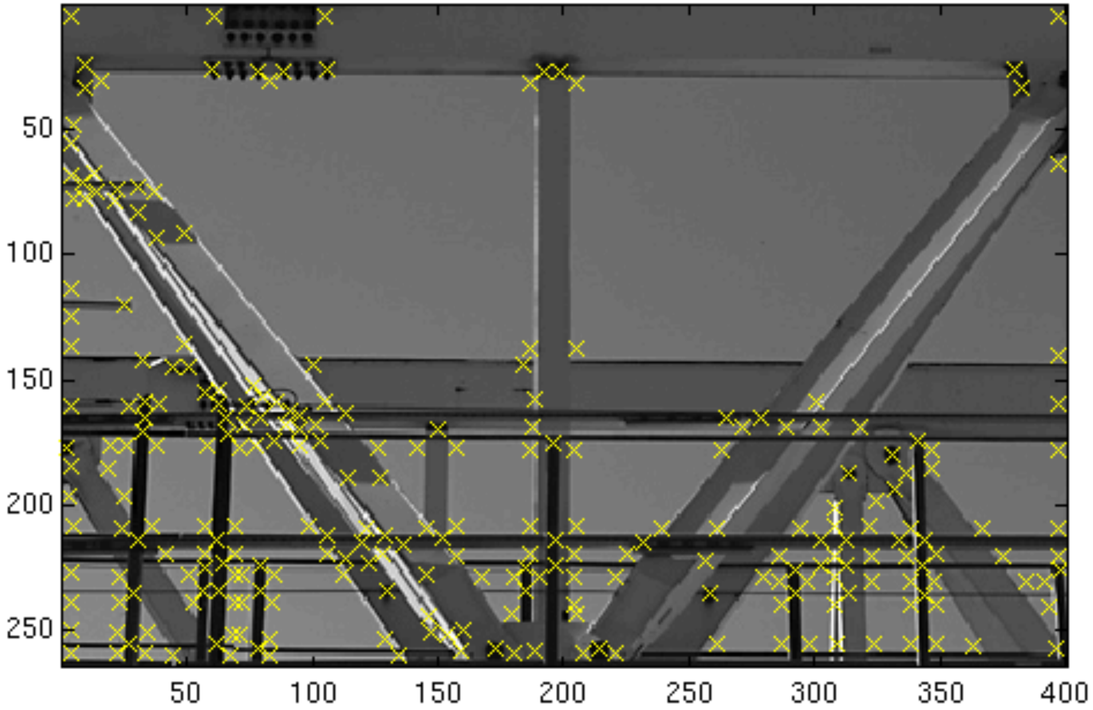
- Hessian determinant

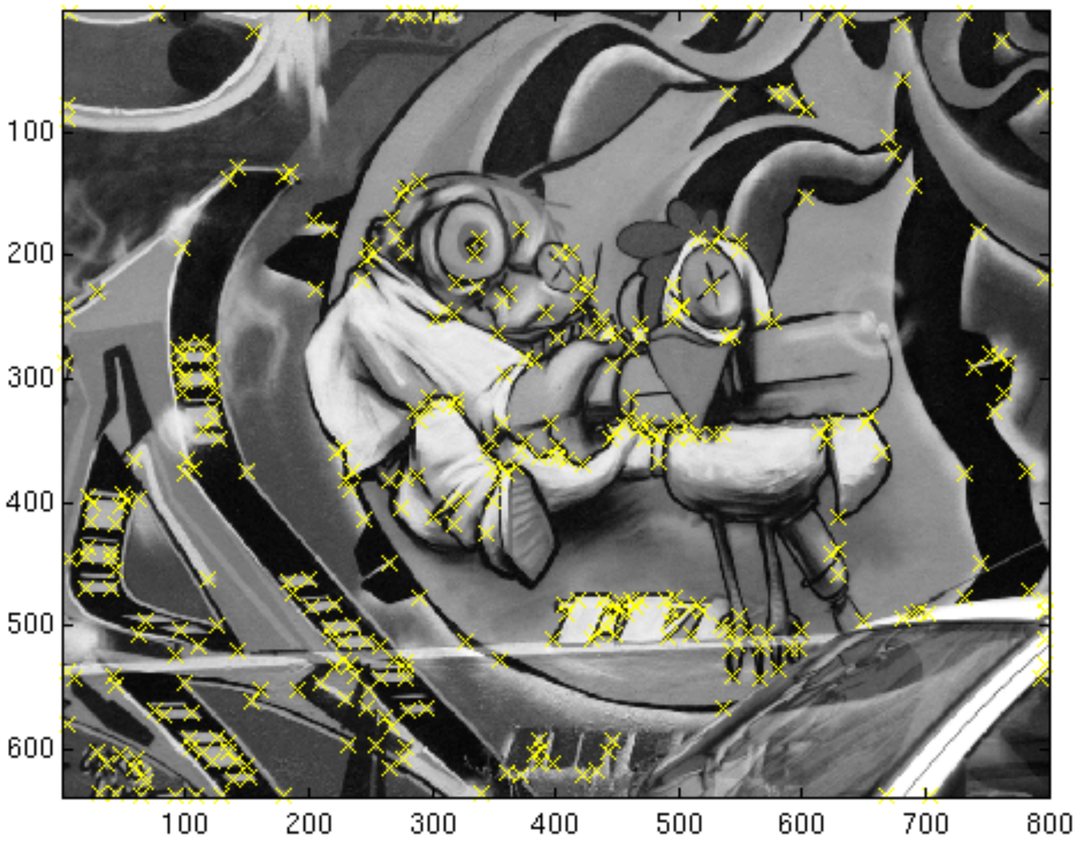
$$H(I) = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{pmatrix}$$

$$\det\left(H(I)\right) = I_{xx} I_{yy} - I_{xy}^2$$



Hessian interest points





Discussion

- Interest point detection so far:
  - Based on Harris or Hessian detector
  - Finds interesting, i.e. discriminative points
    (Harris detector was the “de-facto” standard for a long time)
  - Used for recognition, correspondence for stereo, sparse optical flow/motion, etc.

- To come: Clarify the goals of interest point detection
  - Distinctiveness vs. invariance to transformation
  - Harris & Hessian find distinctive points - but they are **not** invariant to scale, affine and projective transformations