Computer Vision I - Algorithms and Applications: Basics of Digital Image Processing – *Part 2*

Carsten Rother

06/11/2013

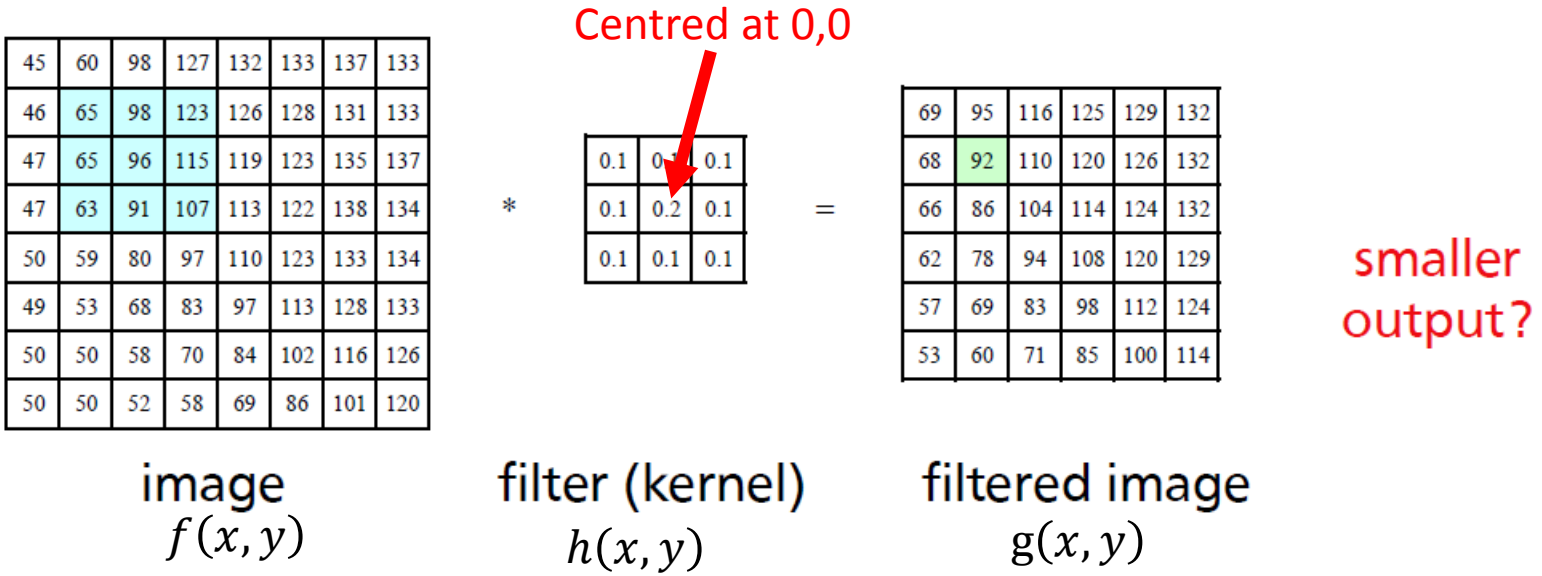
Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

Reminder: Convolution

- Replace each pixel by a linear combination of its neighbours and itself.
- 2D convolution (discrete)

$$g = f * h$$



$$g(x,y) = \sum_{k,l} f(x-k, y-l)h(k,l) = \sum_{k,l} f(k,l)h(x-k, y-l)$$

Reminder – Linear Filters

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$

(a) box, $K = 5$

(b) bilinear

(c) "Gaussian"

(d) Sobel

(e) corner

Roadmap: Basics of Digital Image Processing

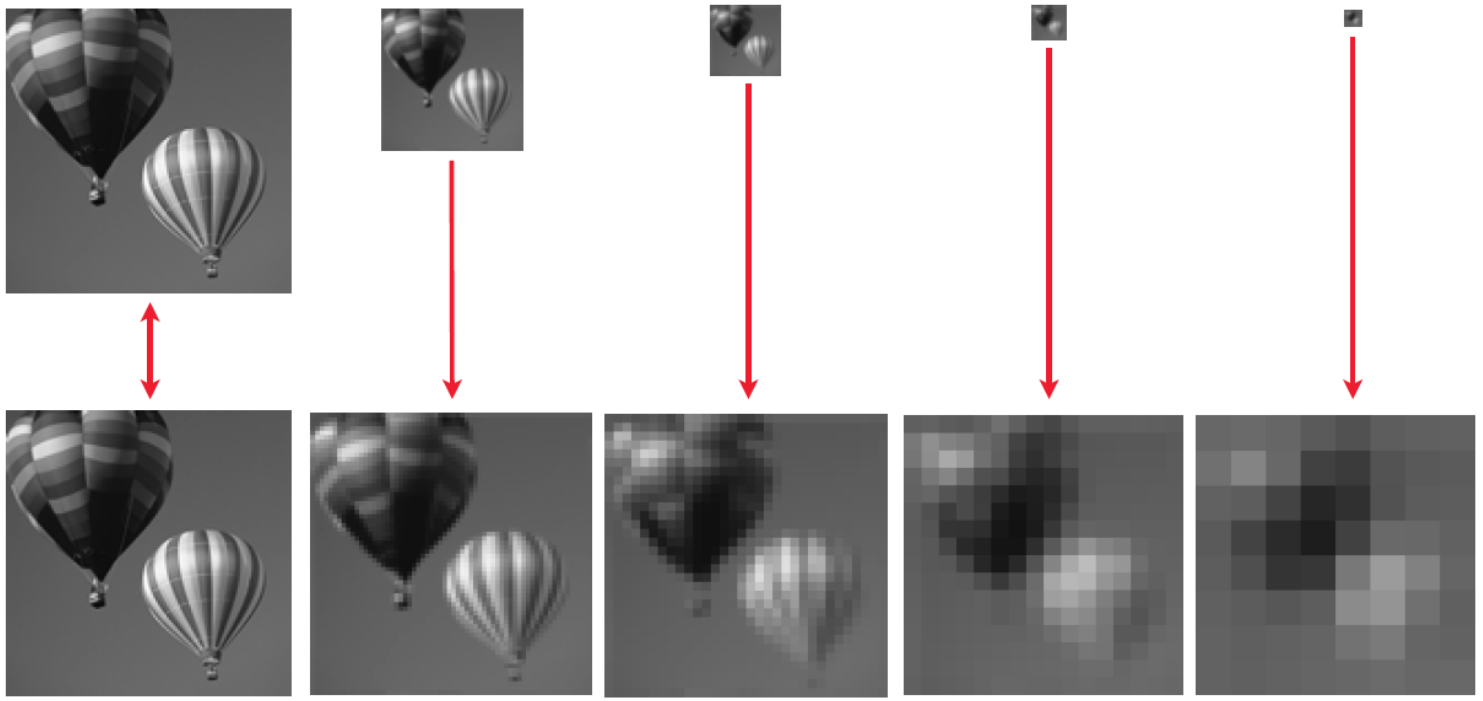
- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

Gaussian Image Pyramid

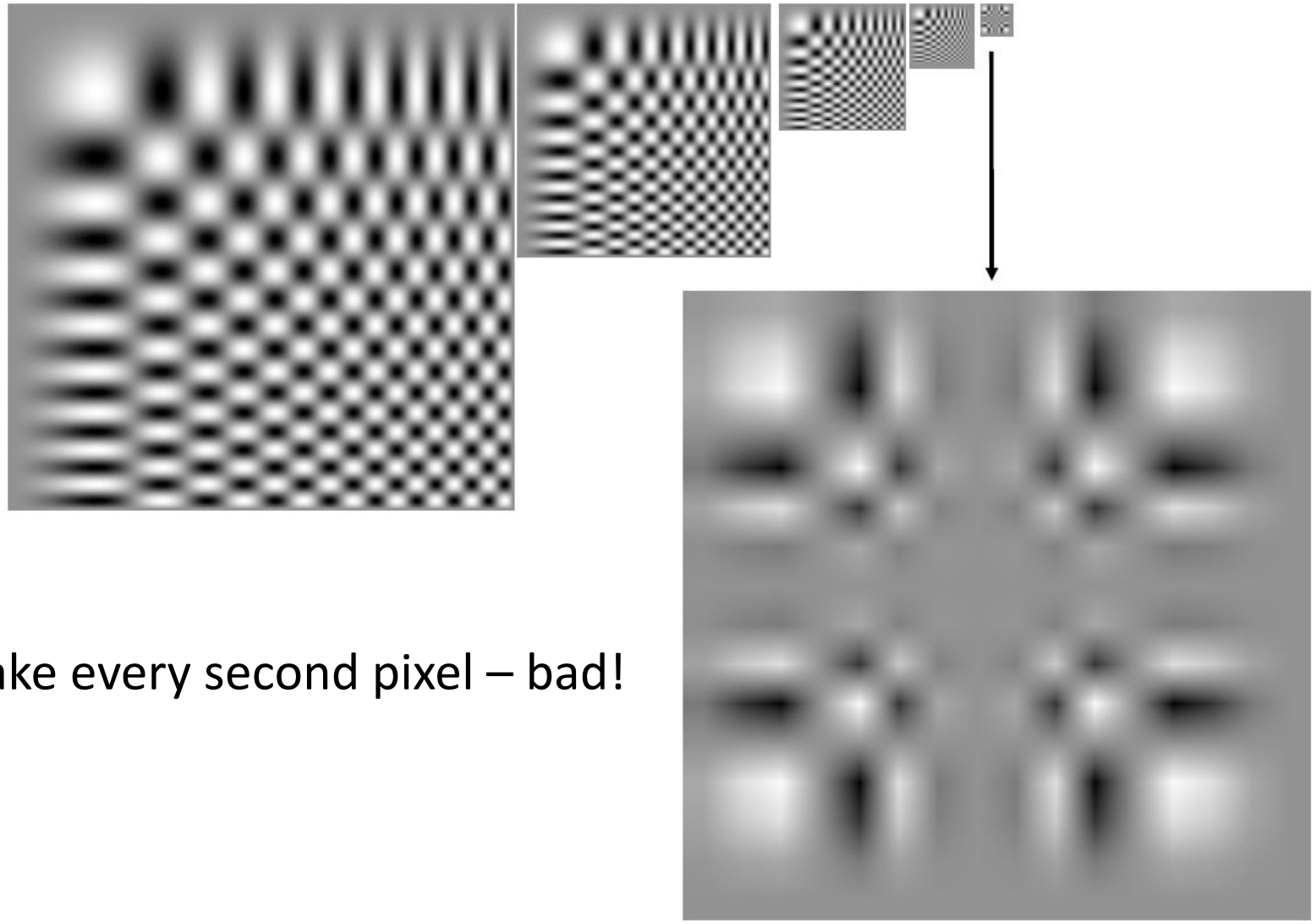
- Represent Image at multiple resolution

High resolution

Low resolution



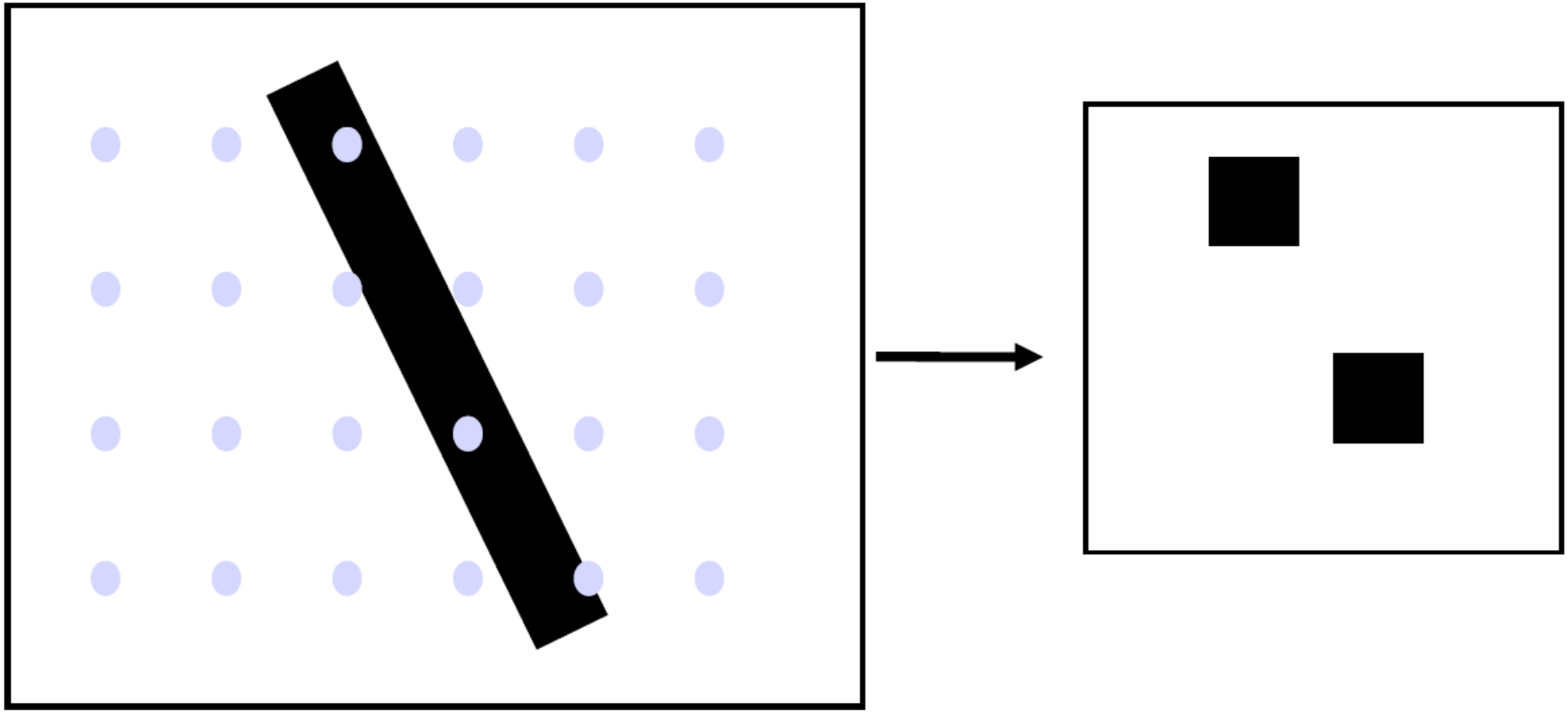
A naive approach



Take every second pixel – bad!

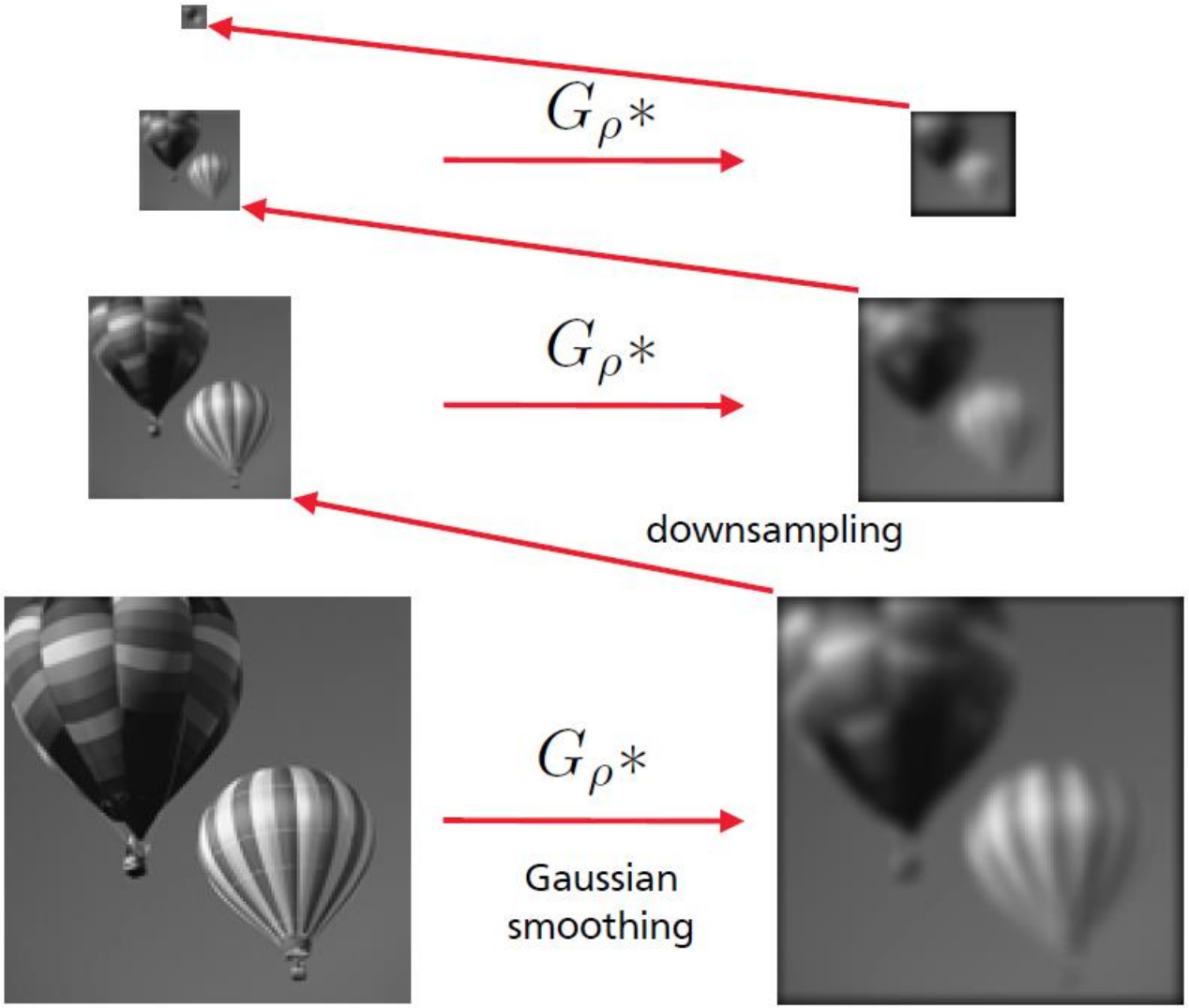
[From book: Computer Vision A modern Approach, Ponce and Forsyth]

Problem: Aliasing Effect



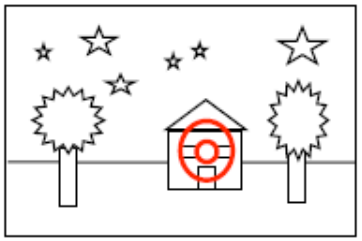
Problem: High frequencies (sharp transitions) are lost

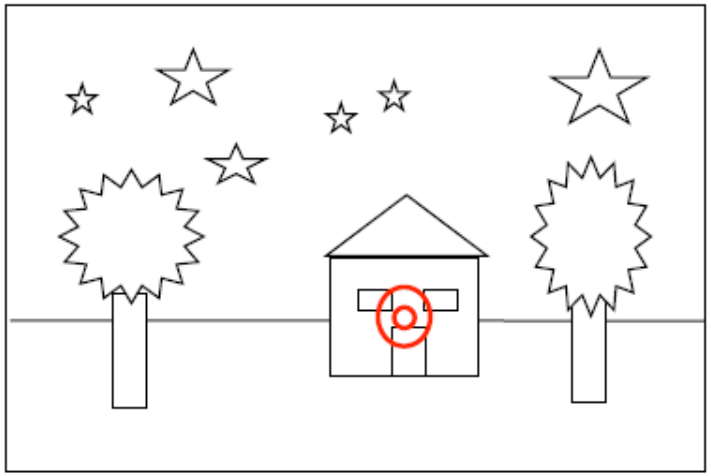
Solution: Smooth before downsampling



Application: template search

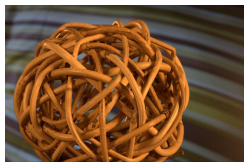
Search template: 





[from Irani, Basri]

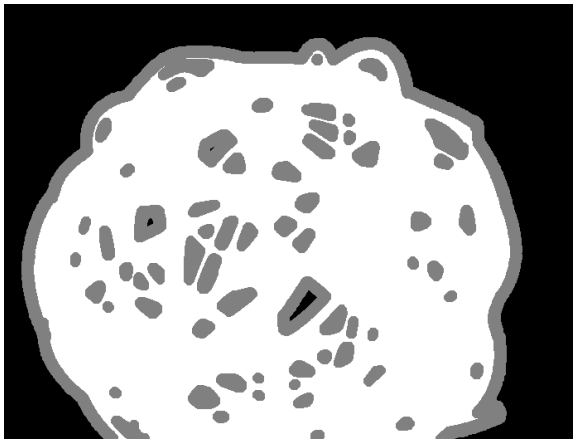
Application: Large Image Segmentation



Small image (100x100)

Small segmentation result

Large image, e.g. 10 MPixel



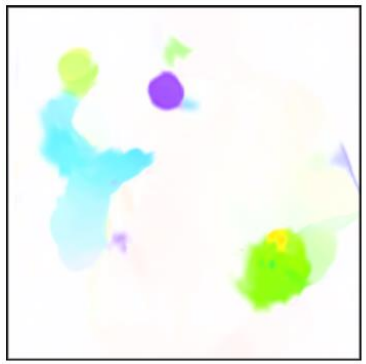
Trimap: created from small image

Segmentation large image

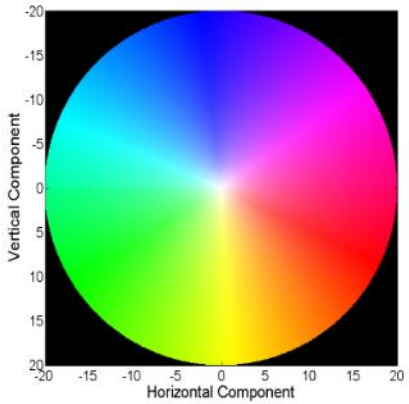
Banded Segmentation

Application: Large Label Space

2 images
(overlaid)



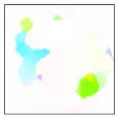
Motion
$200\ x\ 200$ possible
discrete movements
(40.000 labels)

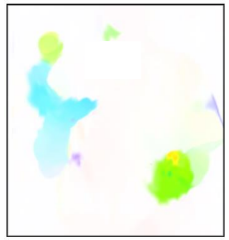


Color coding

Approach:

1. solve problem on small image ($x5$ downscale) with $40\ x\ 40$ label space **(coarse motion)**
   (1600 labels)
2. Do on full resolution only in $5x5$ neighbourhood around each solution (**add fine motion)**
   (25 labels)





(problem small objects can get lost)

Roadmap: Basics of Digital Image Processing

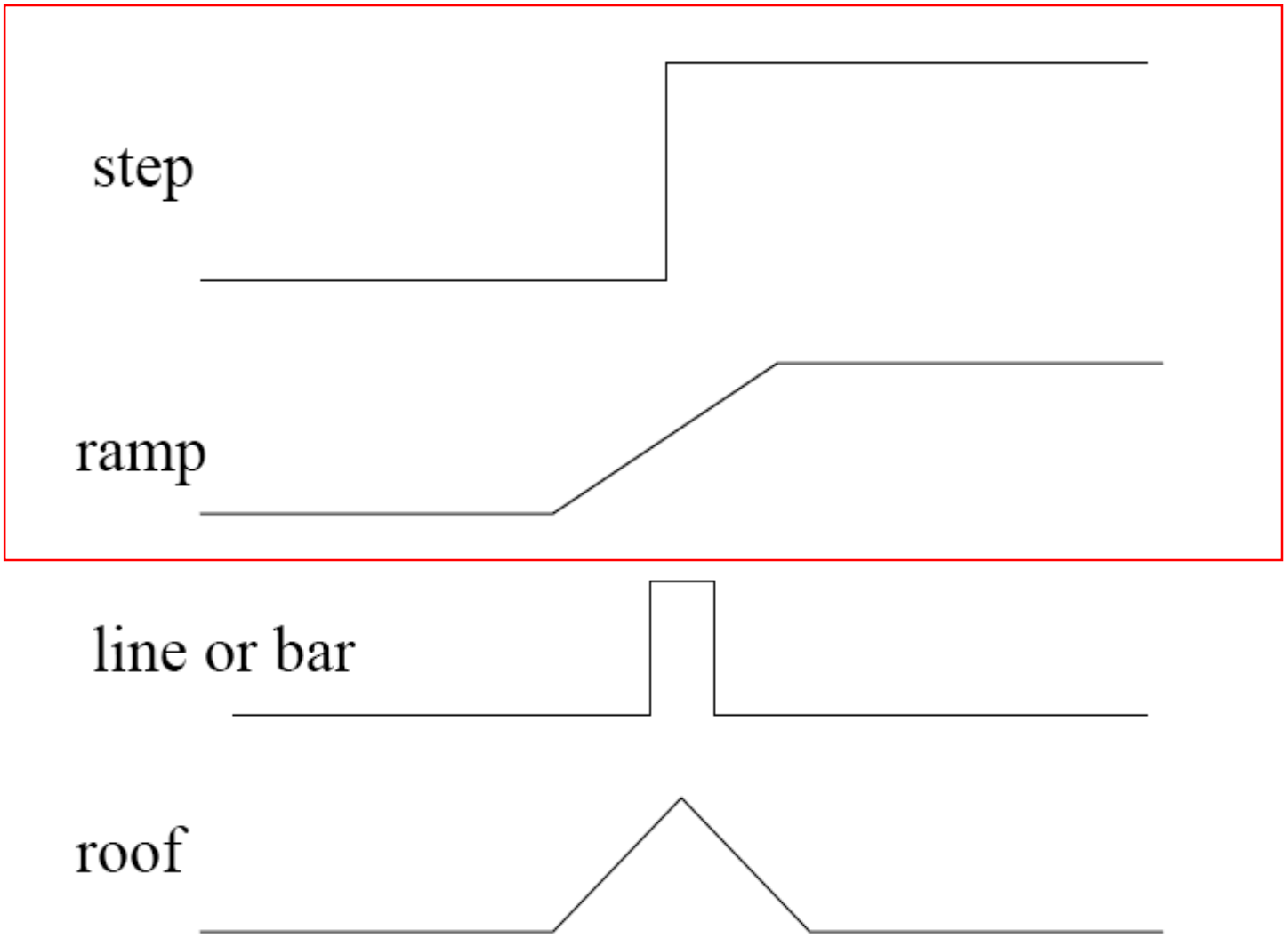
- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

Goal: Find long edge chains

What do we want:

- Good detection: we want to find edges not noise
- Good localization: find true edge
- Single response: one per edge (independent of edge sharpness)
- Long edge-chains

Idealized edge types



We focus on this

What are edges ?

- correspond to fast changes in the image
- The magnitude of the derivative is large

Image of 2 step edges

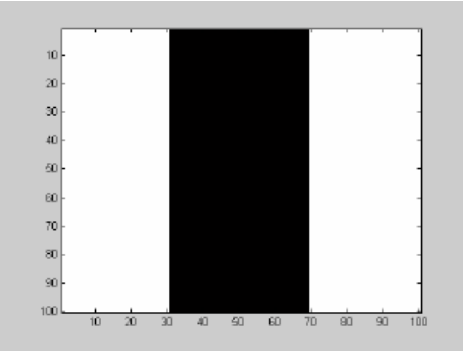
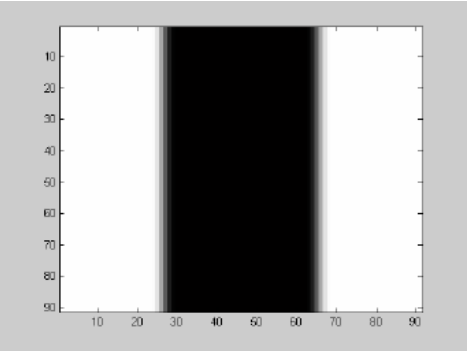
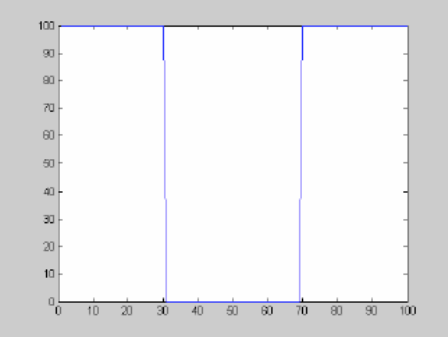


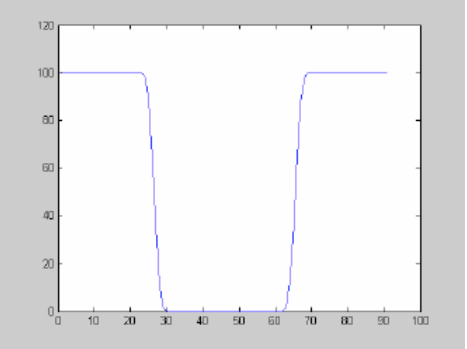
Image of 2 ramp edges



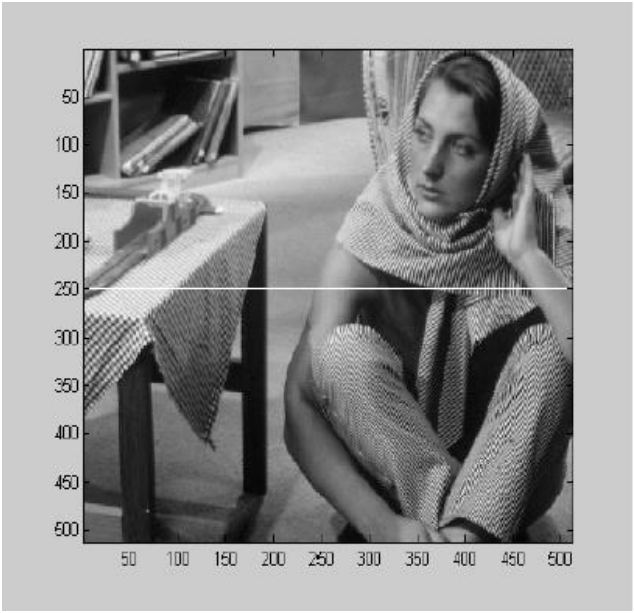
Slizce through the image



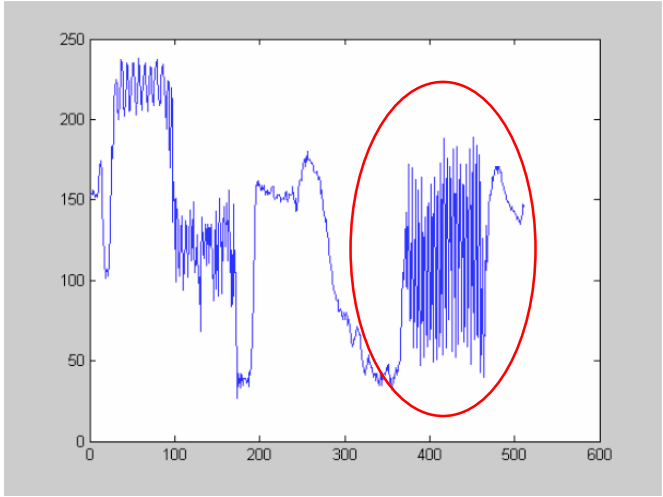
Slizce through the image



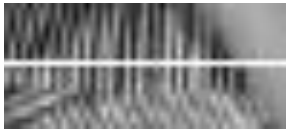
What are fast changes in the image?



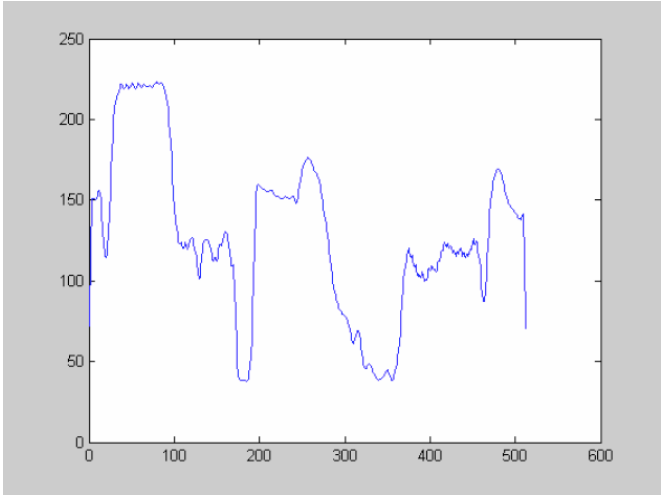
Image



Scanline 250



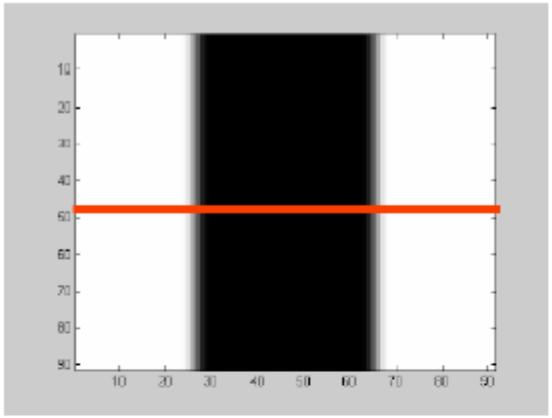
Texture or many edges?

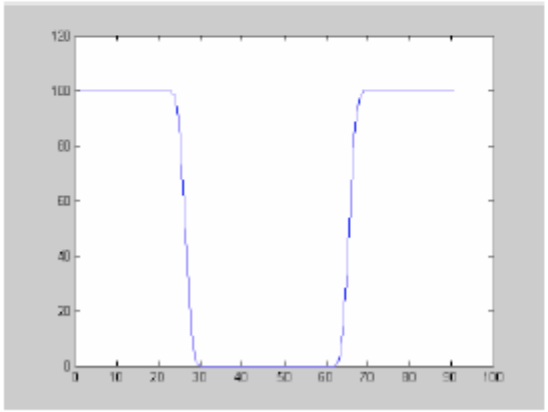


Edges defined after smoothing

Scanline 250 smoothed with Gausian

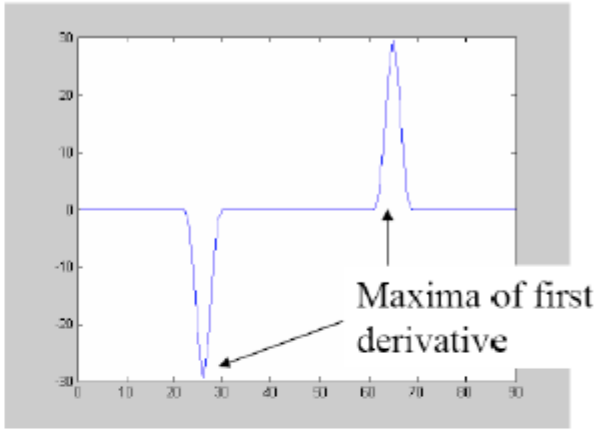
Edges and Derivatives

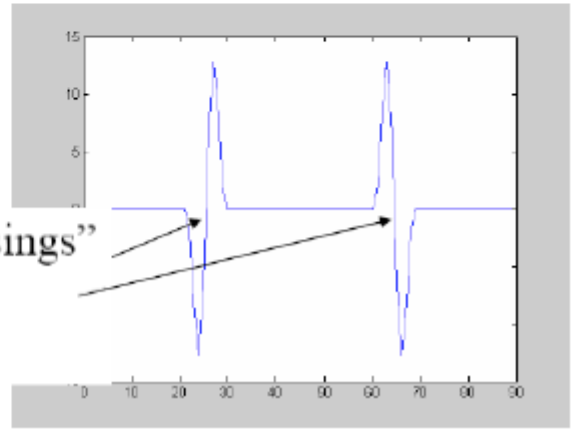




1st derivative

2nd derivative





We will look at this first

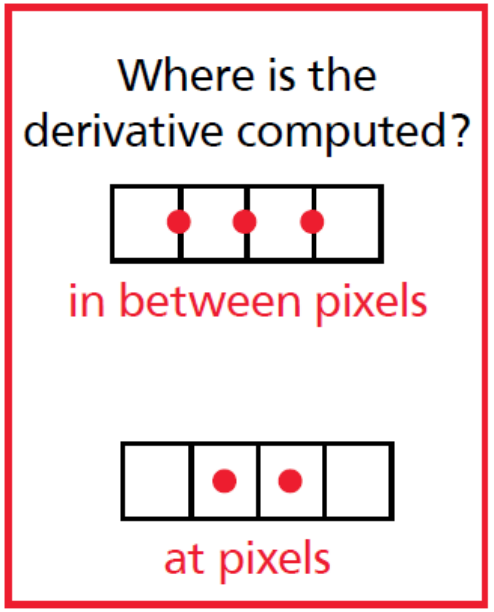
Edge filters in 1D

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

We can implement this as a linear filter:

Forward differences: 1/2 | -1 | 1 |

Central differences: 1/2 | -1 | 0 | 1 |



Reminder: Seperable Filters

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$$





(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

This is the centralized differencesoperator

Edge Filter in 1D: Example

Based on 1st derivative

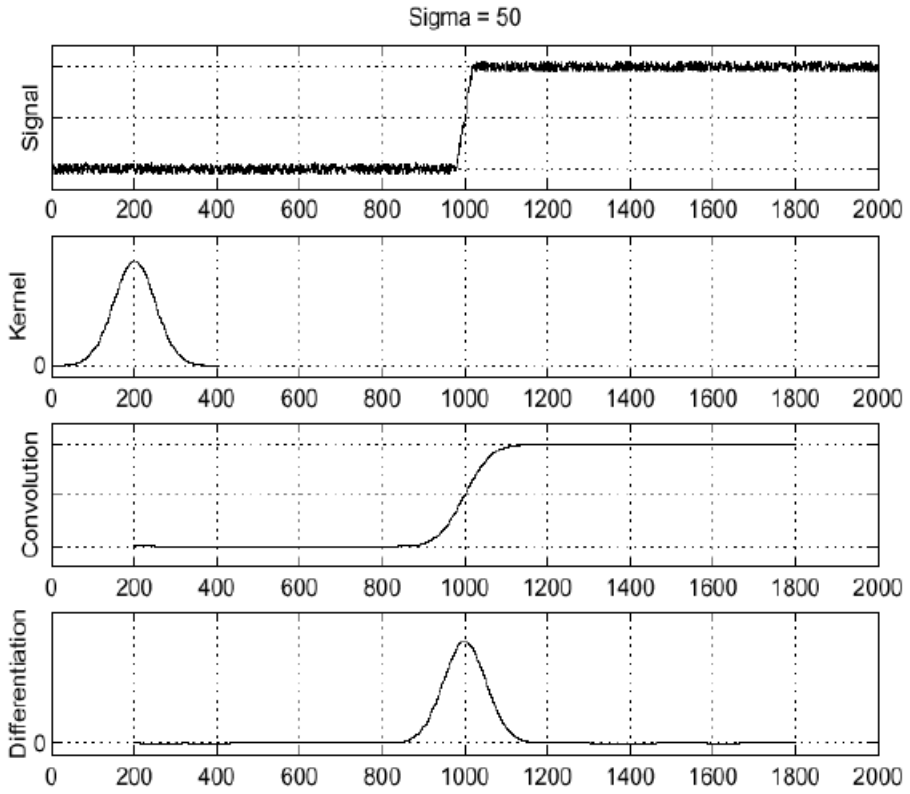
- Smooth with Gaussian – to filter out noise
- Calculate derivative
- Find its optima

f

g

$g * f$

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



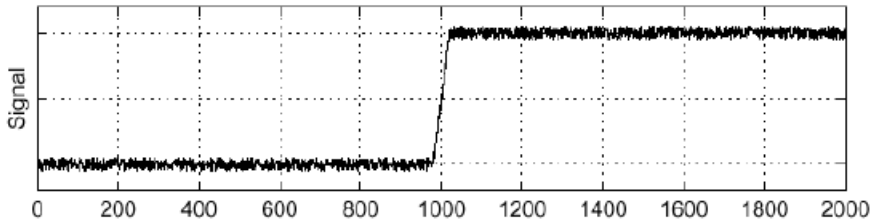
Edge Filtering in 1D

Simplification:
(saves one operation)

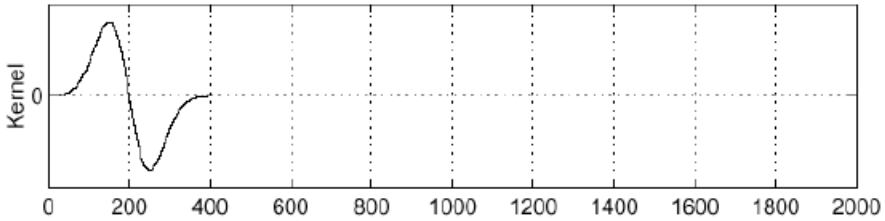
$$\frac{\mathrm{d}}{\mathrm{d}x}(g * f) = \left(\frac{\mathrm{d}}{\mathrm{d}x}g\right) * f$$

Derivative of Gaussian

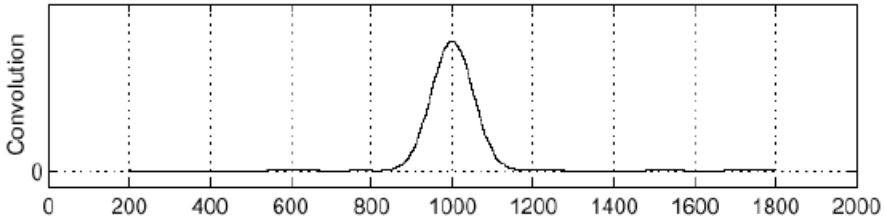
f



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

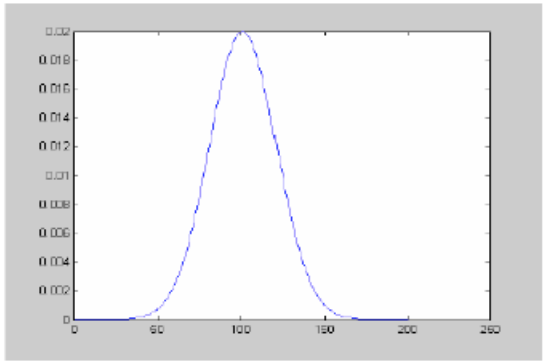


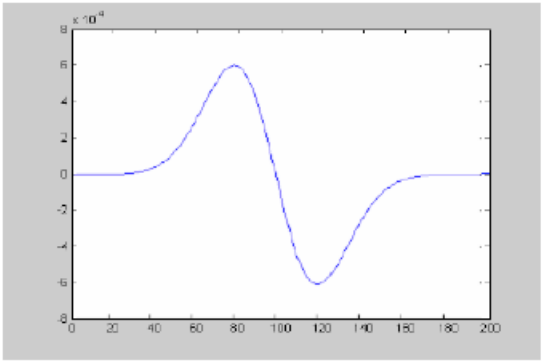
$\left(\frac{\mathrm{d}}{\mathrm{d}x}g\right) * f$



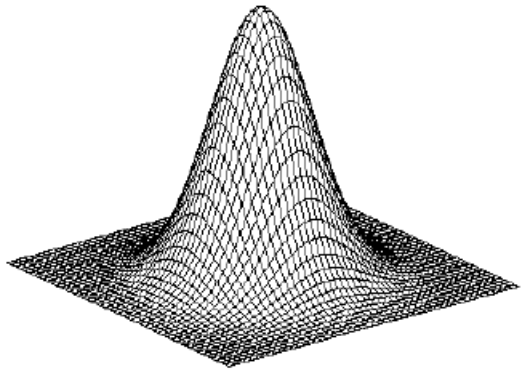
Edge Filtering in 2D

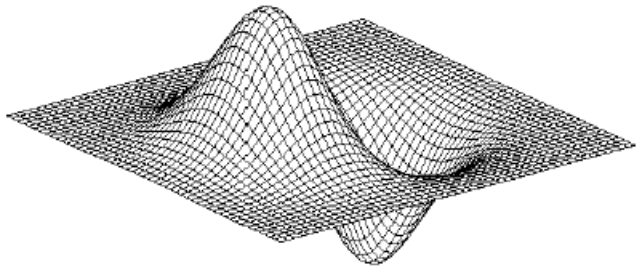
- Derivative in x-direction: $D_x * (G * I) = (D_x * G) * I$
  - in 1D:



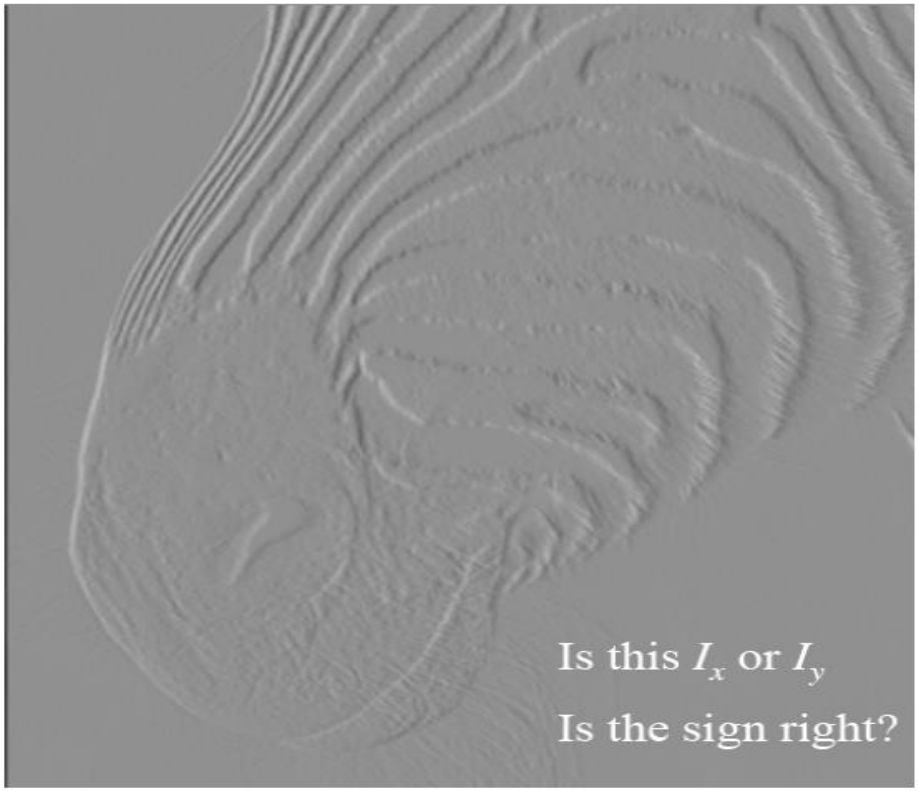


  - in 2D:



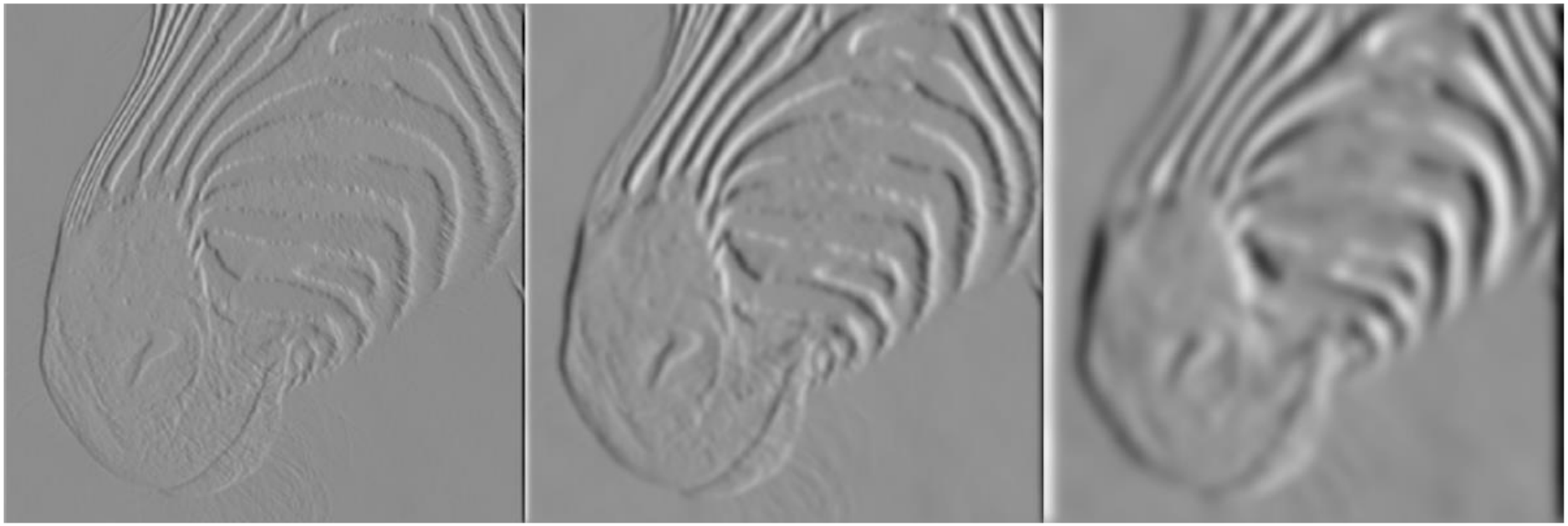


Edge Filter in 2D: Example

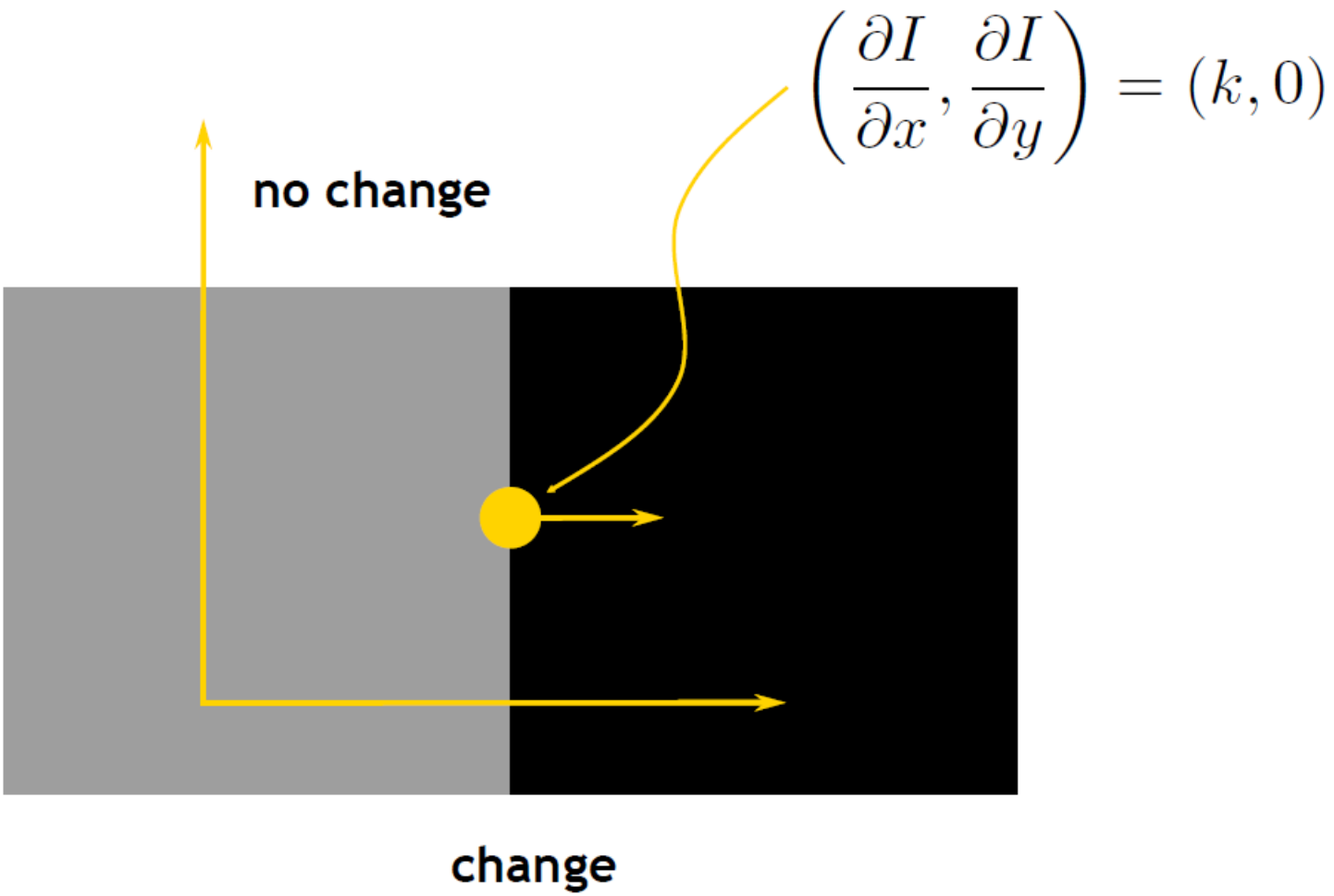


Edge Filter in 2D

x-derivatives with different Gaussian smoothing

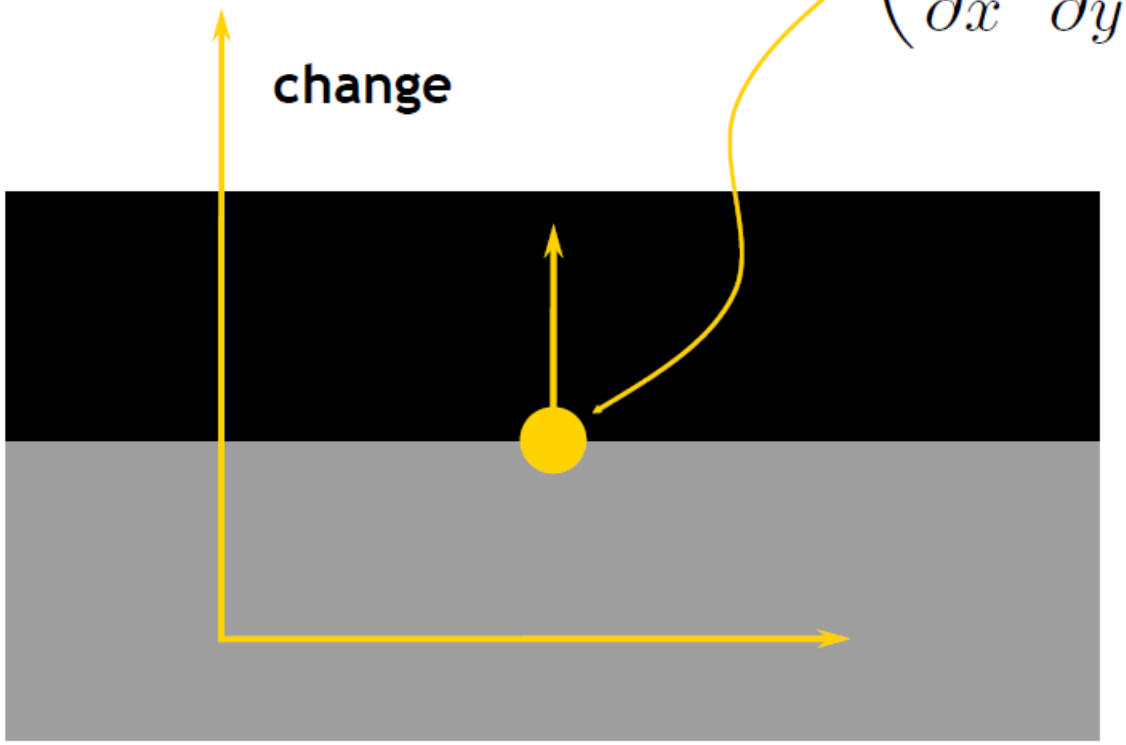


What is a gradient



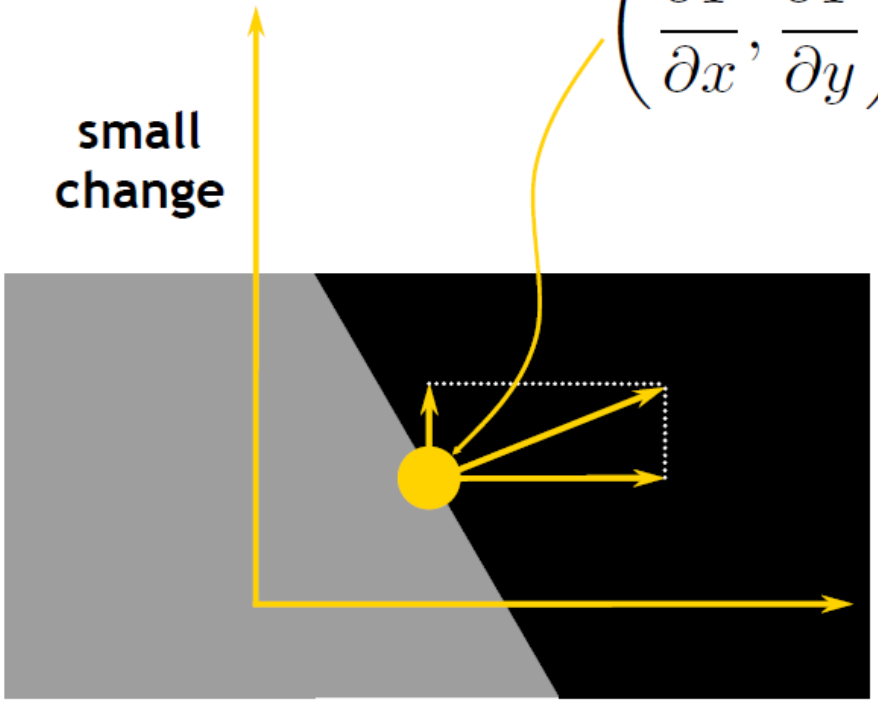
What is a gradient

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



What is a gradient

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

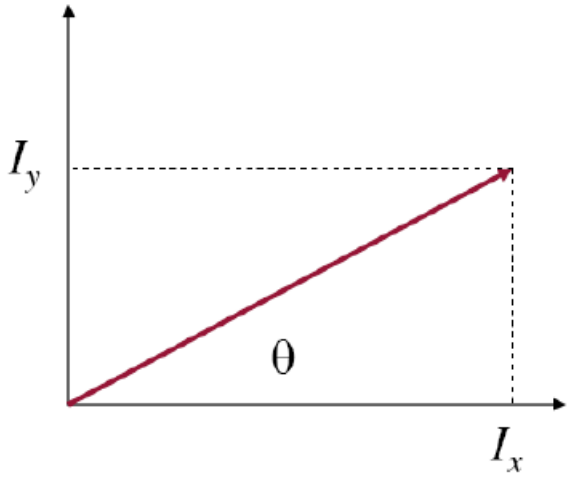


- gradient direction is perpendicular to edge
- gradient magnitude measures edge strength

What is a Gradient

- the gradient is:

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



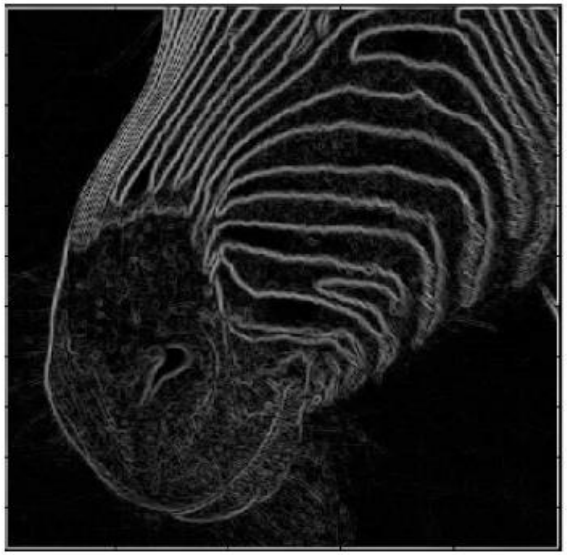
- the magnitude of the gradient is:

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

- the direction of the gradient is:

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

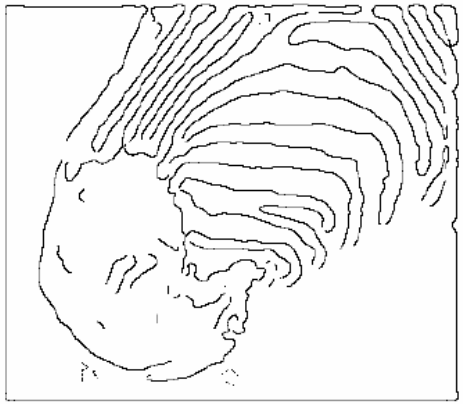
Example – Gradient magnitude



First smoothed with Gaussian

First smoothed with broad Gaussian

Our goal was to get thin edge chains?



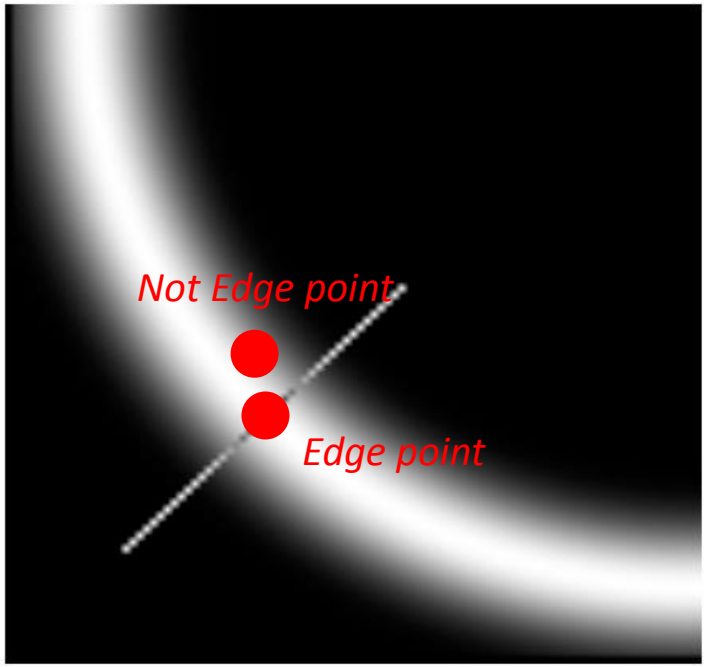
How to get edge chains

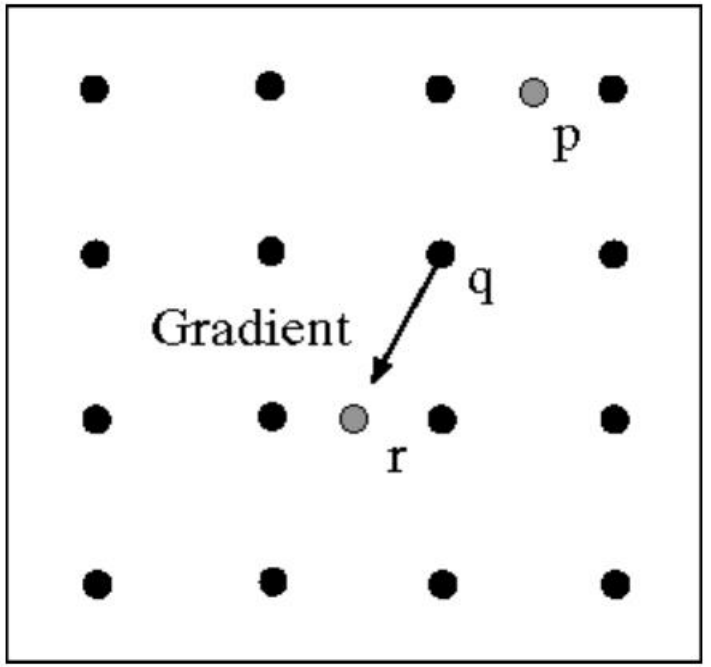
Rough Outline of a good edge detector (such as Canny)

1. Compute robust Gradient Image: $((D_x * G) * I , (D_y * G) * I)$
2. Find edge-points ("edgels"): non-maximum suppression
3. Link-up edge-points to get chains
4. Do hysteresis to clean-up chains

edge-points
or edgel

Non-maximum surpression

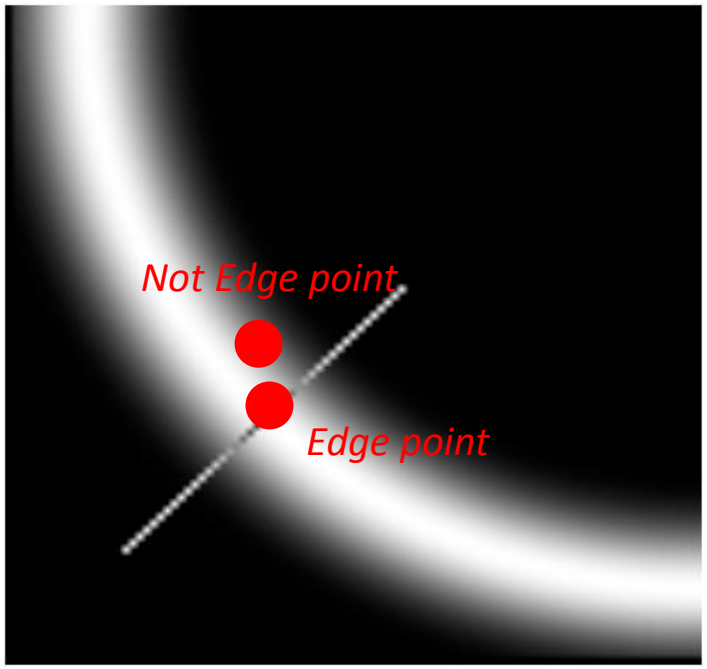


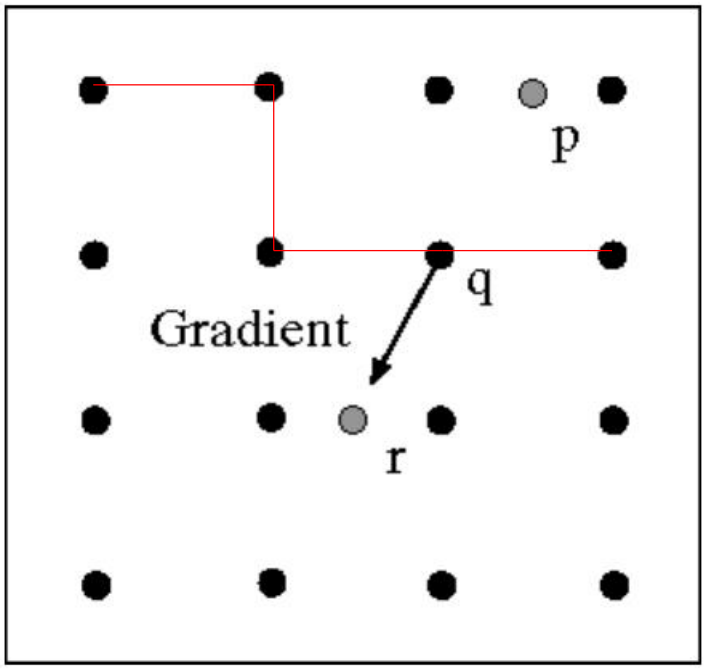


Wich pixel is an edge point ? (non-max surpression)

1. Check if pixel is local maximum at gradient orientation (interpolate values for p,r)
2. Accept edge-point if above a threshold

Link up edge-points to get chains

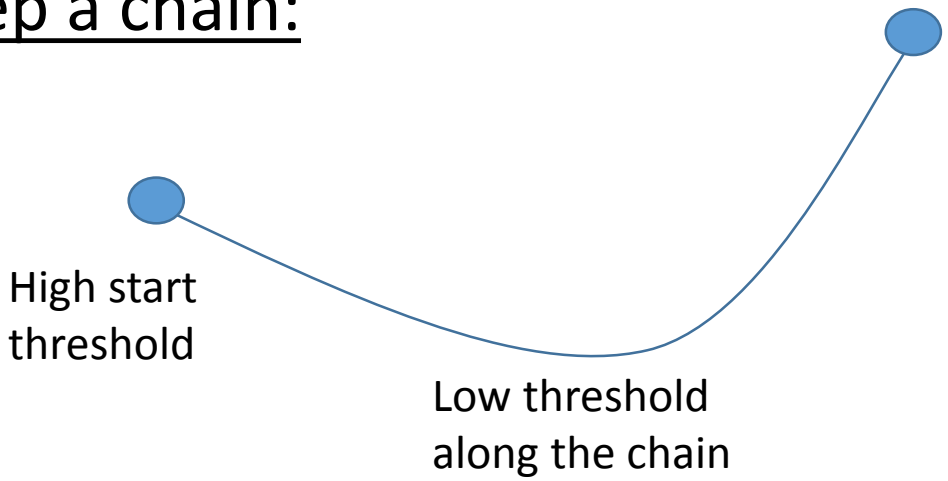




1. Link-up neighbouring pixels if both are edge-points.

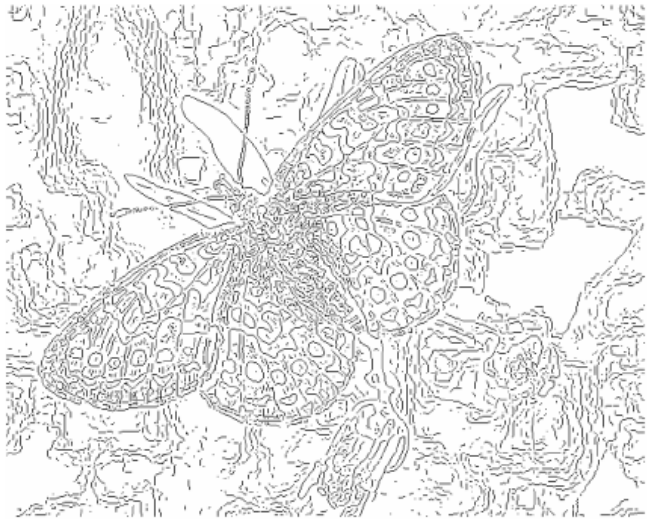
Clean up chains with Hysteresis

Keep a chain:



Final Result

Image



Not much smoothing (fine scale)

much smoothing (coarse scale)
small threshold

much smoothing (coarse scale)
large threshold

Alterative: Edge detection with Laplacian Filter

- The Laplacian:

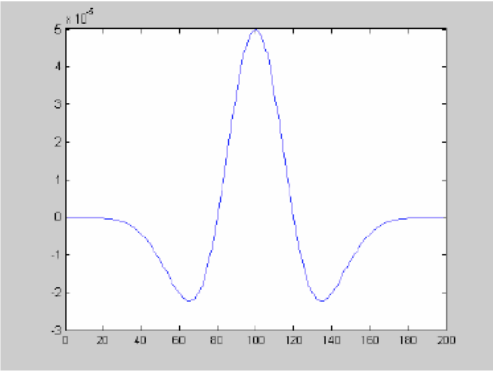
$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

- just another linear filter:

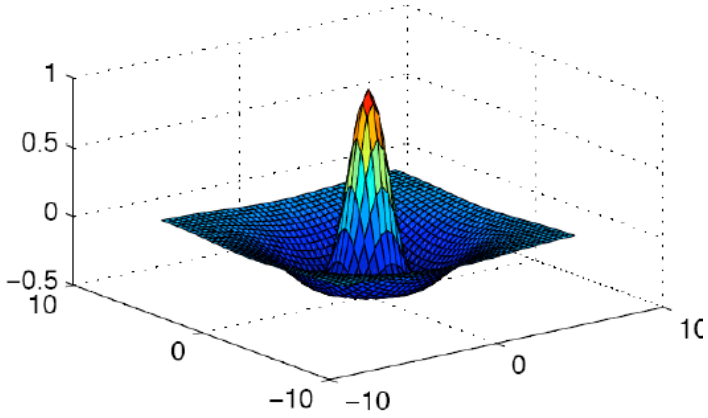
$$\nabla^2 (G * I) = (\nabla^2 G) * I$$

Called Laplacian of Gaussian (LoG)

- in 1D:



in 2D (mexican hat):



Laplacian example in 1D

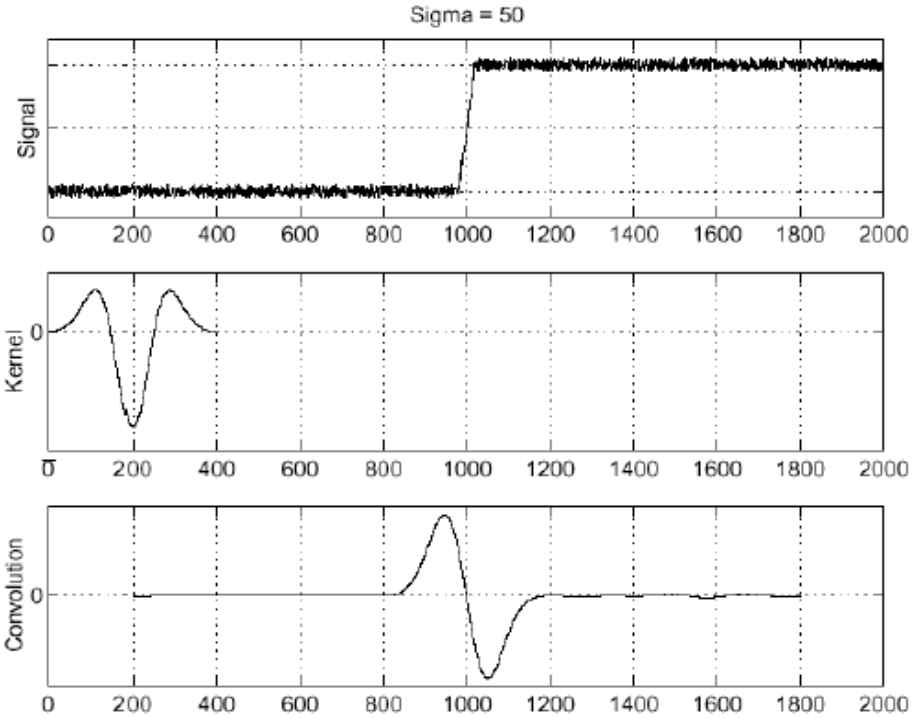
- using Laplacian

LoG - Laplacian of Gaussian operator

$$f$$

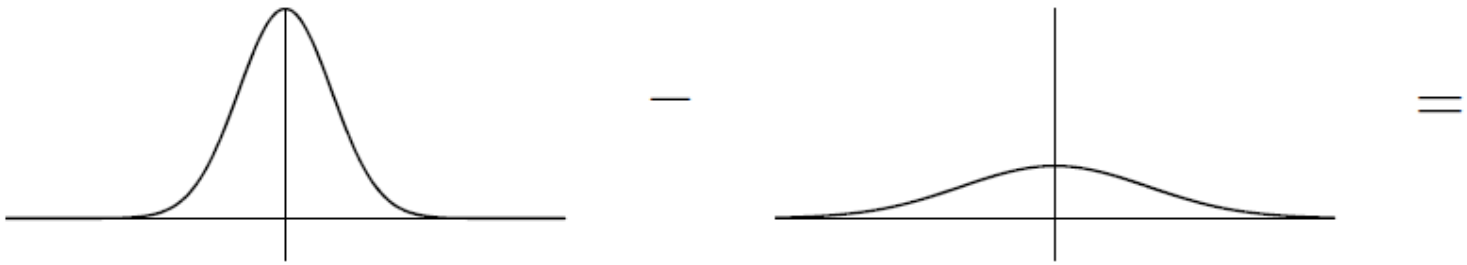
$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} g$$

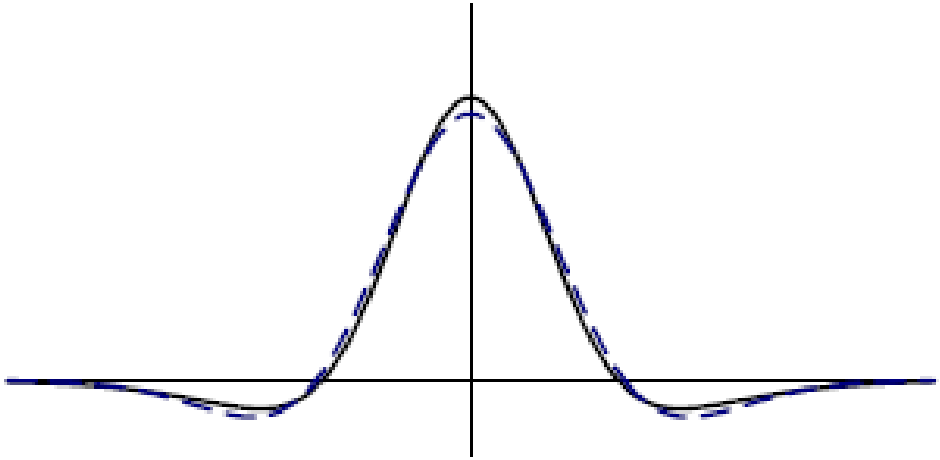
$$\left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} g \right) * f$$



Find zero-crossing

Approximate LoG with Difference of Gaussian (DoG)



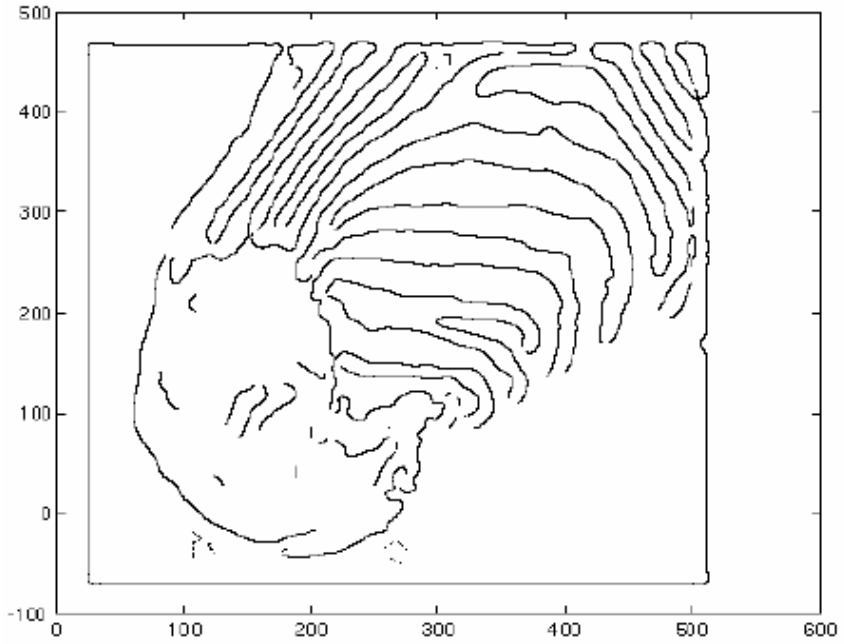


Solid Line:
Differnce of Gaussian (DoG)

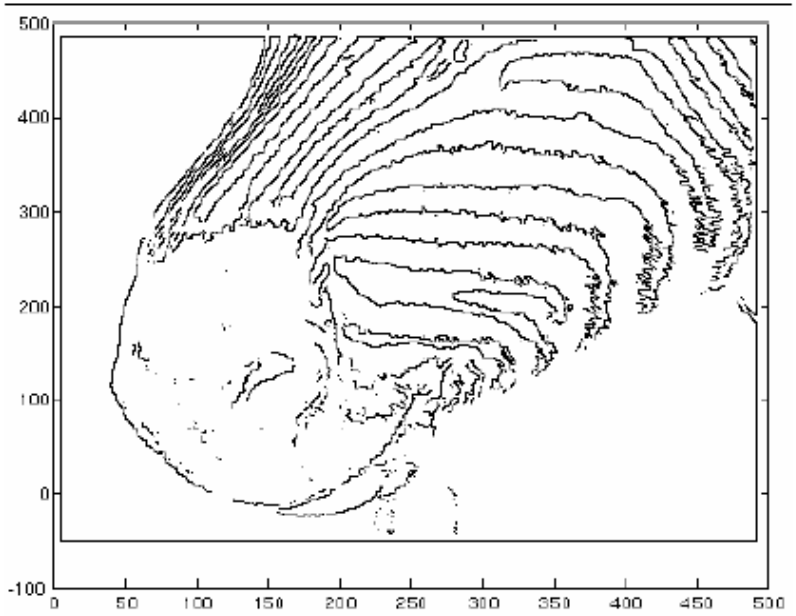
Dashed Line:
Laplacian of Gaussian

Laplacian example in 2D

sigma = 4

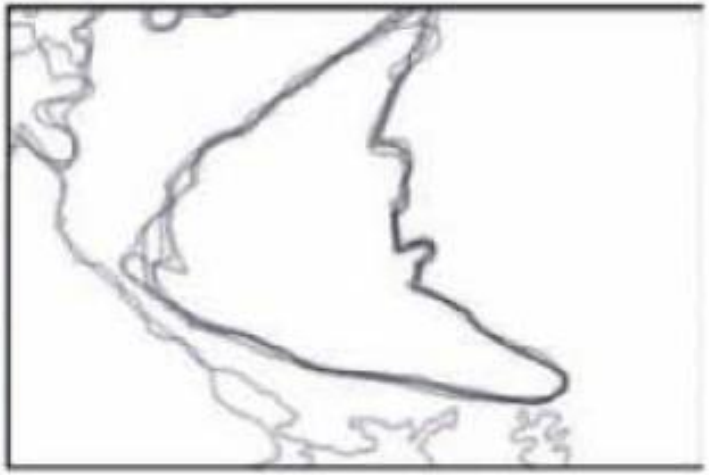


sigma = 2

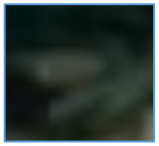


Future Lecture: Segmentation

- So far we looked at “jumps” in gray-scale images?
- Humans perceive edges very differently (edges depend on semantic)



“average human drawing”



Hard for computer vision method which operates only locally!

[from Martin, Fowlkes and Mail 2004]

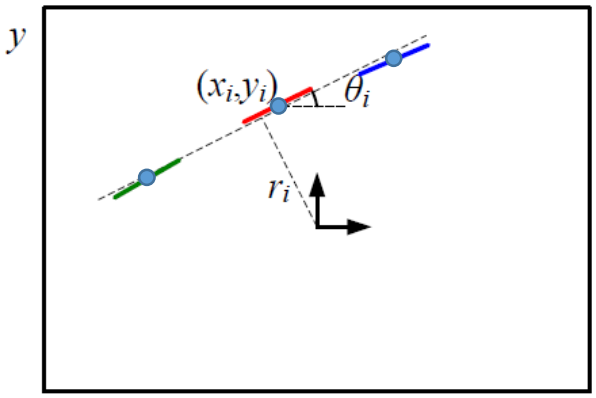
Try to learn semantically meaningful image edges

- Features: brightness gradient; color gradient; texture gradient; weighted combination, etc.

Roadmap: Basics of Digital Image Processing

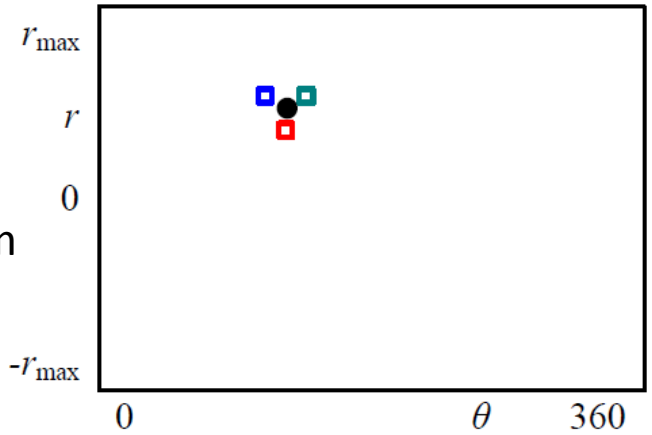
- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

Hough voting with edgels (edge-points)



3 edgels
(edge-points with direction)

Hough transform



Hough Space
(line is now a point)

Algorithm:

1. Empty cells in Hough Space
2. Put for each Edgel(θ, r) into a cell of the Hough Space
3. Find Peaks in Hough Space (use non-max suppression)
4. Re-fit all edgels to a single line

Hough Voting: original

Goal: find all lines

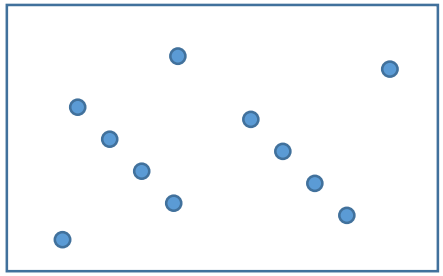


Image with points

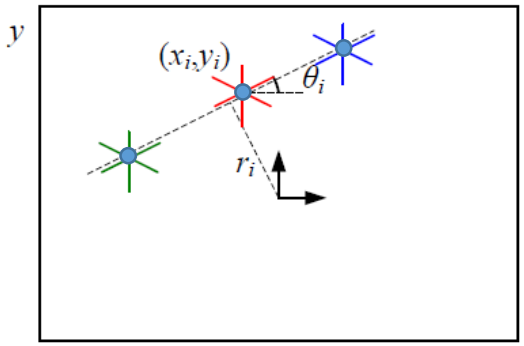
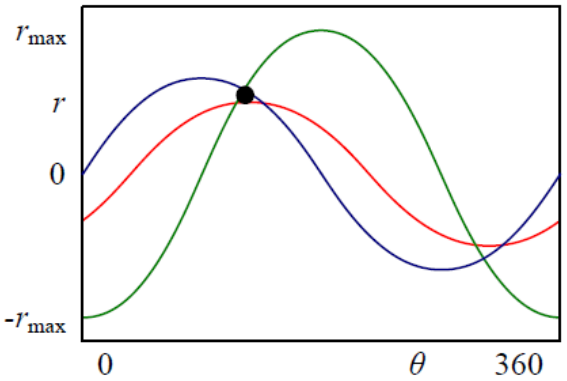


Image with just 3 points

Hough transform

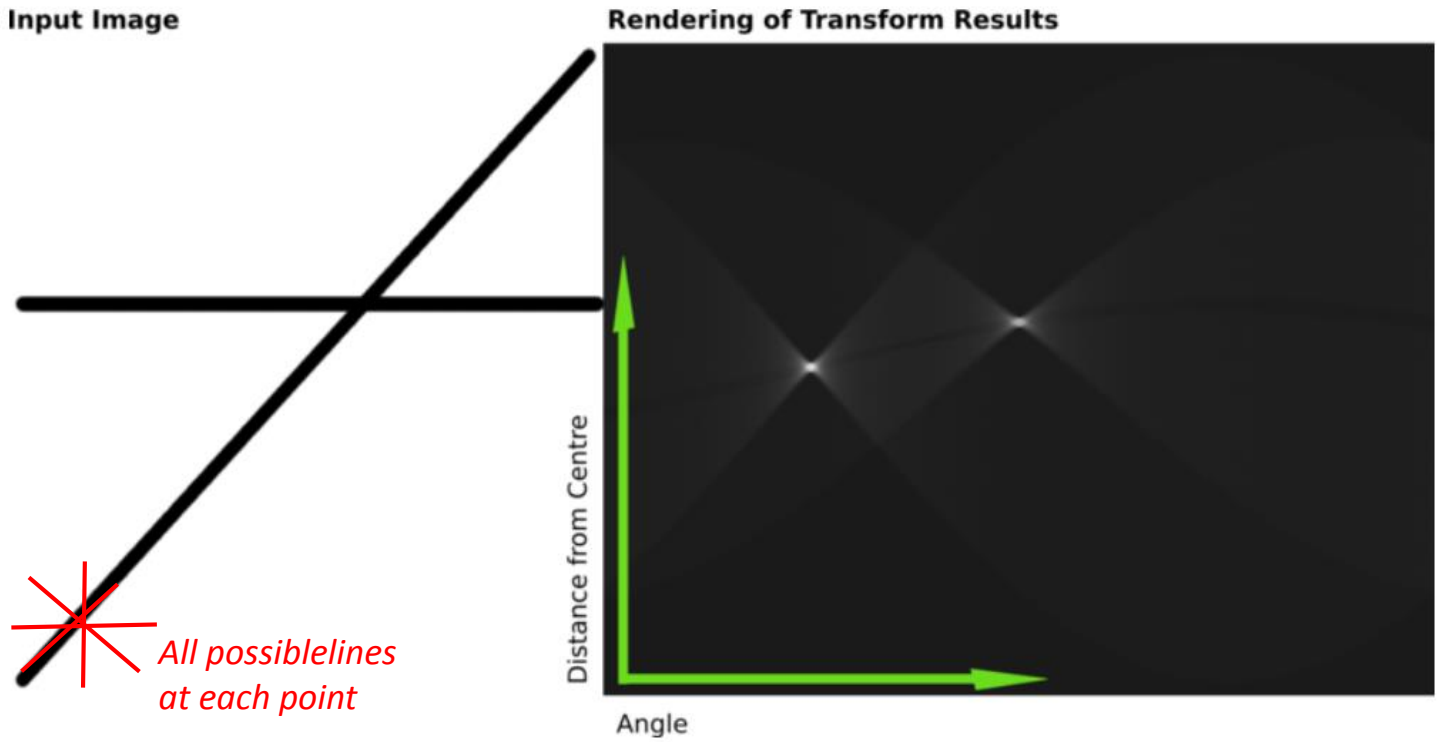
Goal: find all points with many "votes" in accumulator space



All lines that go through the 3 points

This idea of transformation to a voting space can be used for many scenarios

Hough transform: original



[From Wikipedia]

Example: Orthogonal Vanishing point detection

846 line segments found

Found 3 orthogonal vanishing points

Algorithm: RANSAC (explained later)
Application: Camera Calibration (see later)

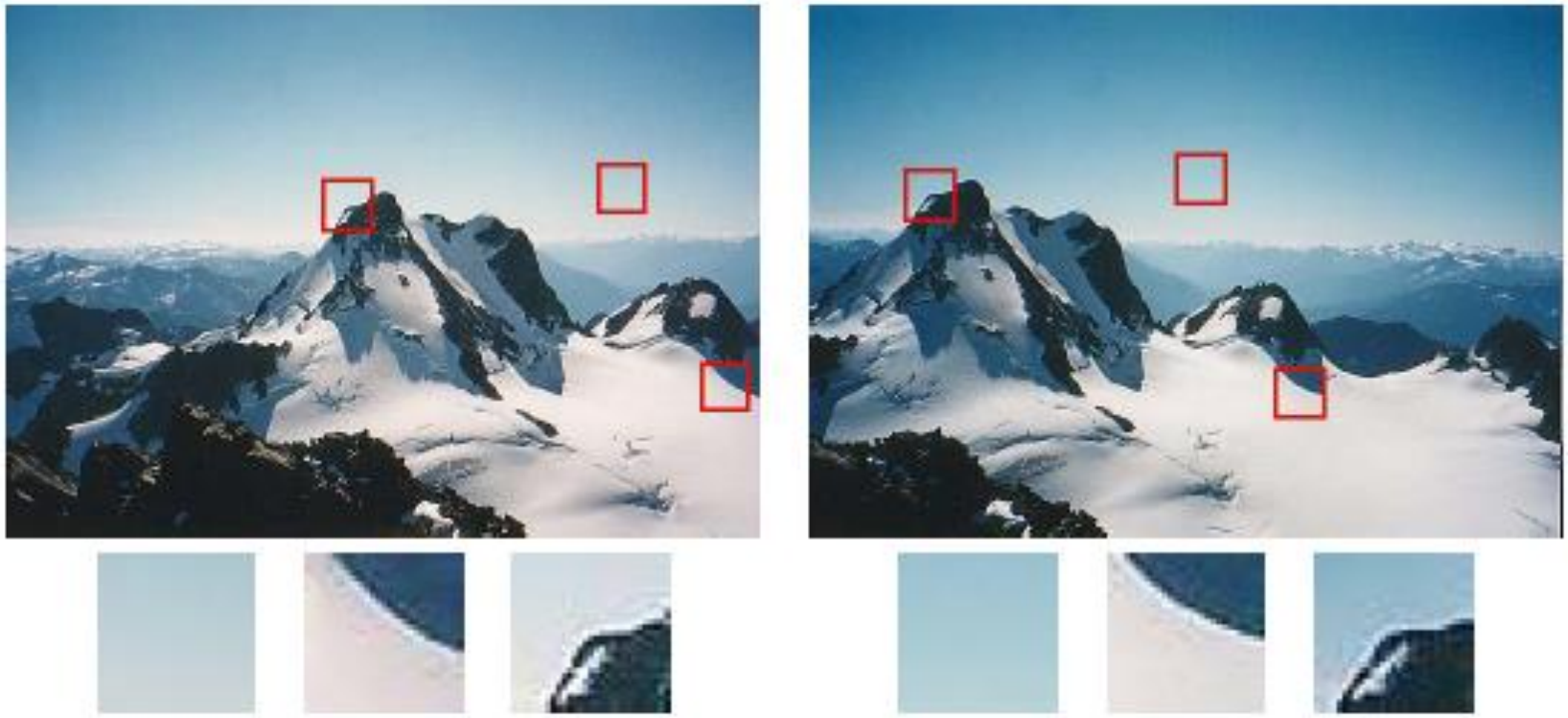
[Rother, Thesis '03]

Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

What region should we try to match?

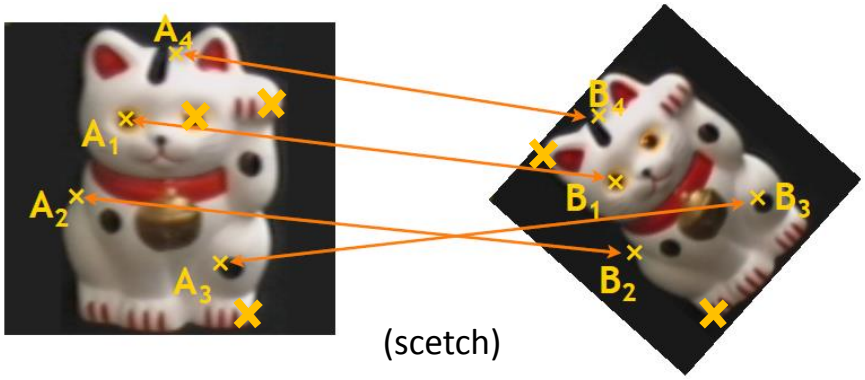
We want to find a few regions where this image pair matches: Applications later



Look for a region that is unique, i.e. not ambiguous

Goal: Interest Point Detection

- Goal: predict a few "interest points" in order to remove redundant data efficiently



(scetch)

- Should be invariant against:

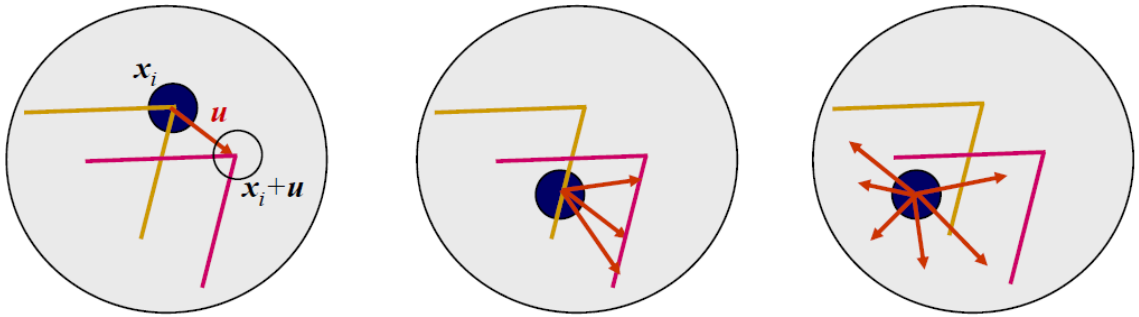
  a. Geometric transformation – scaling, rotation, translation, affine transformation, projective transformation etc.

  b. Color transformation – additive (lightning change), multiplicative (contrast), linear (both), monotone etc.;

  c. Discretization (e.g. spatial resolution, focus);

Points versus Lines

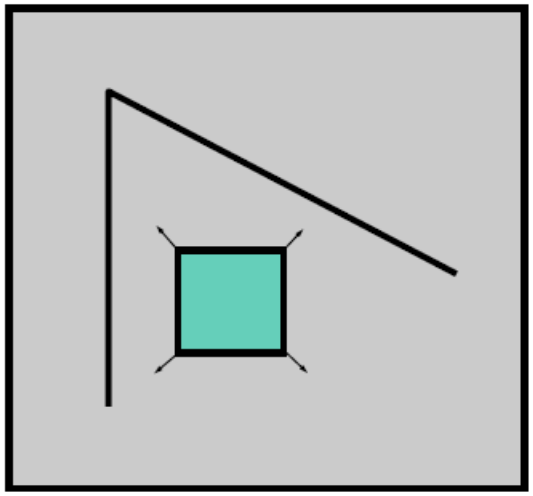
„Apeture problem“



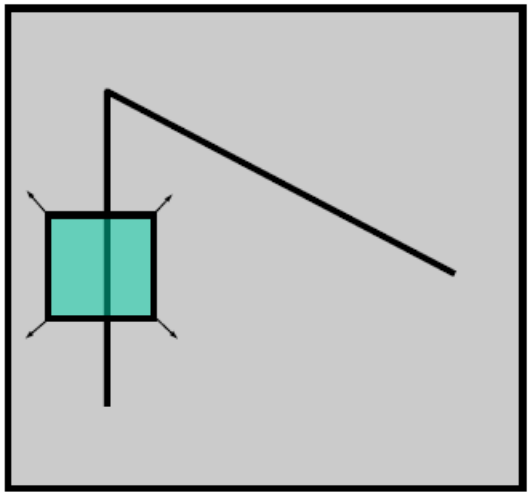
Lines are not as good as points

Harris Detector

Local measure of feature uniqueness:
Shifting the window in any direction: how does it change

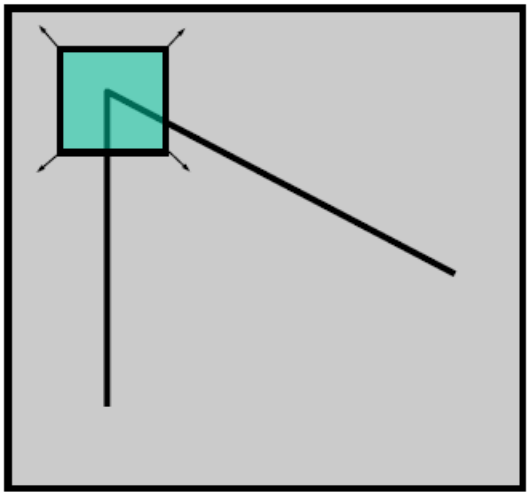


"flat" region:
no change in all directions



"edge":
no change along the edge direction

Shift left



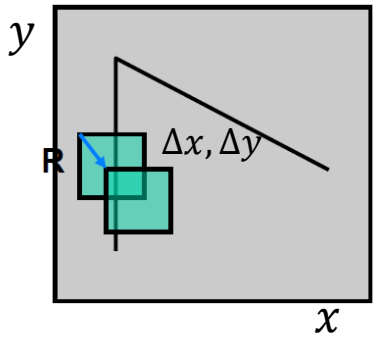
"corner":
significant change in all directions

Shift top,left

[Szeliski and Seitz]

Harris Detector

How similar is the image $I(x, y)$ to itself?



Autocorrelation function:

$$c(x, y, \Delta x, \Delta y) = \sum_{(u,v) \in W(x,y)} w(u, v)\big(I(u, v) - I(u+\Delta x, v+\Delta y)\big)^2$$

$W(x, y)$ is a small vicinity (window) around (x, y)

$w(u, v)$ is a convolution kernel, used to decrease the influence of pixels far from (x, y), e.g. the Gaussian $\exp\left[-\frac{(u-x)^2+(v-y)^2}{2\sigma^2}\right]$
For simplicity we use $w(u, v) = 1$

Harris Detector

$$c(x, y, \Delta x, \Delta y) = \sum_{(u,v) \in W(x,y)} \Big(I(u, v) - I(u+\Delta x, v+\Delta y) \Big)^2$$

One is interested in **properties** of $c(x, y, \Delta x, \Delta y)$ at each position (x, y)

Let us look at a linear approximation of $I(u+\Delta x, v+\Delta y)$
Taylor expansion around (u, v)

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

$$\approx I(u, v) + [I_x(u, v), I_y(u, v)] \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Gradient at (u, v)

Harris Detector

Put it together:

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

Q: Structure Tensor

with

$$
Q(x, y) = \begin{bmatrix} \sum_W I_x(u, v)^2 & \sum_W I_x(u, v) I_y(u, v) \\ \sum_W I_x(u, v) I_y(u, v) & \sum_W I_y(u, v)^2 \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}
$$

We compute this at any image location (x, y)

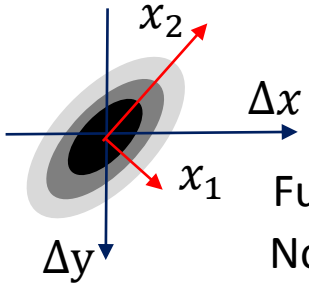
Harris Detector

The autocorrelation function

$$c(x, y, \Delta x, \Delta y) \approx [\Delta x, \Delta y] Q(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Function *c* is (after approximation) a **quadratic** function in Δx and Δy

- Isolines are ellipses ($Q(x, y)$ is symmetric and positive definite);
- Eigenvector x_1 with (larger) Eigenvalue λ_1 is the direction of fastest change in function c
- Eigenvector x_2 with (smaller) Eigenvalue λ_2 is direction of slowest change in function c

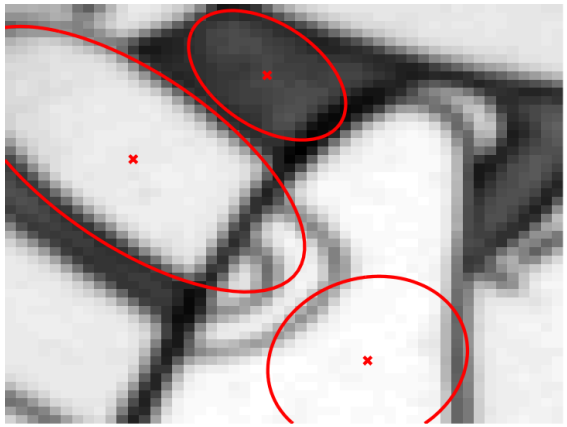


Function *c*

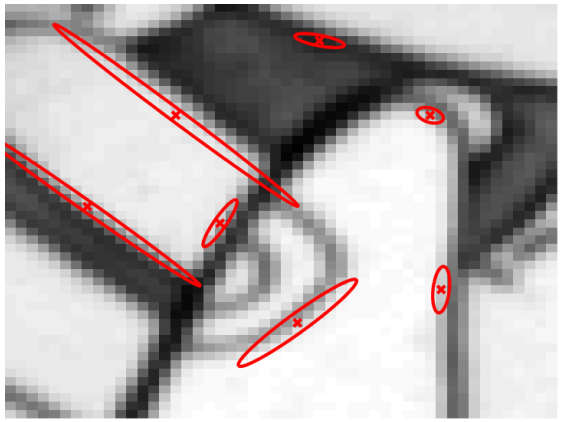
Note $c = 0$ for $\Delta x = \Delta y = 0$

Harris Detector

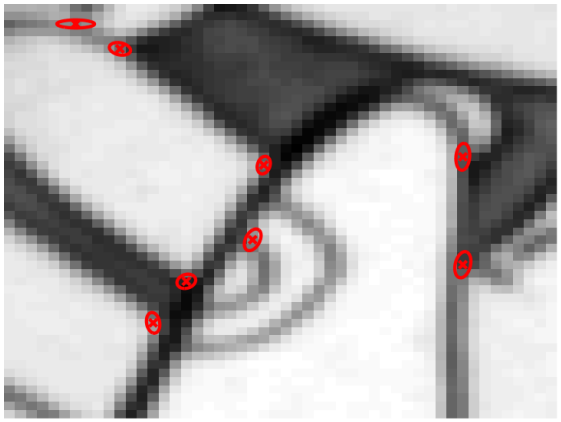
Some examples – isolines for $c(x, y, \Delta x, \Delta y) = 1$:



(a) Flat



(b) Edges



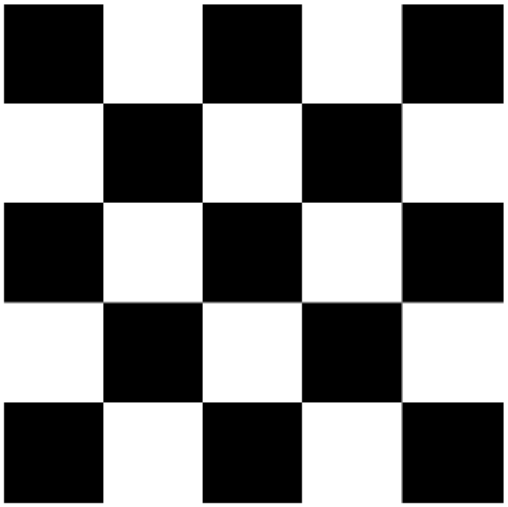
(c) Corners

a. Homogenous regions: both λ-s are small

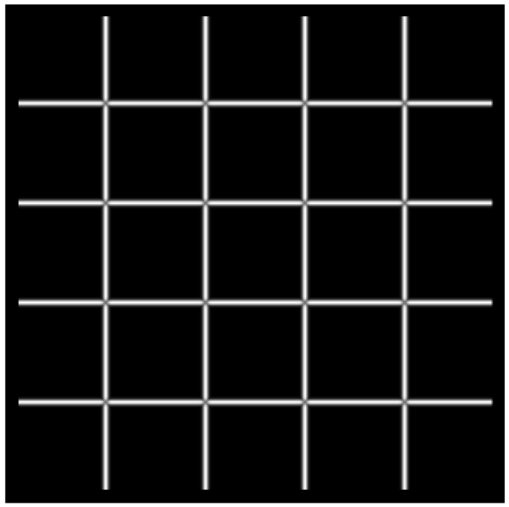
b. Edges: one λ is small the other one is large

c. Corners: both λ-s are large (this is what we are looking for!)

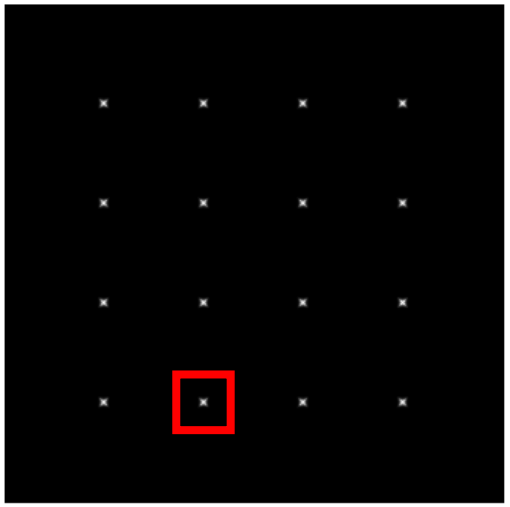
Harris Detector



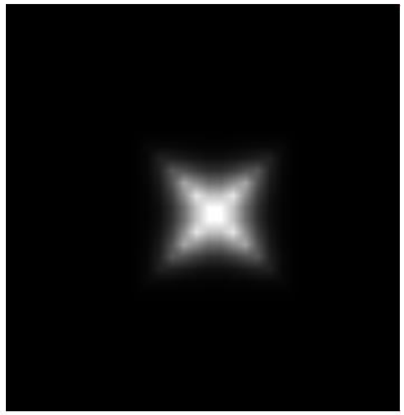
Image



λ_1 larger eigenvalue



λ_2 smaller eigenvalue



Zoom in

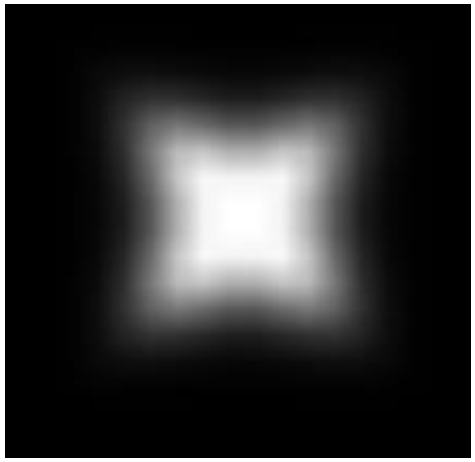
Harris detector

"**Cornerness**" is a characteristic of $Q(x, y)$

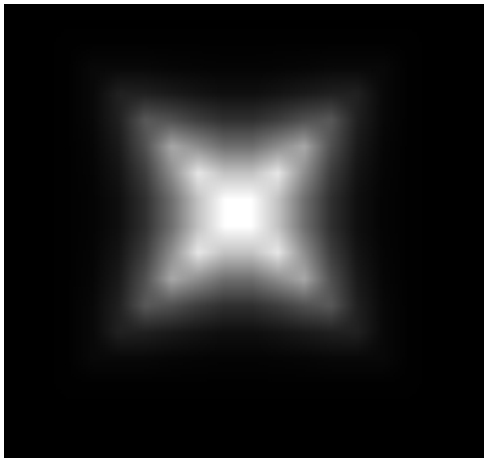
$$\lambda_1\lambda_2 = \det Q(x,y) = AC - B^2, \quad \lambda_1 + \lambda_2 = \text{trace}\, Q(x,y) = A + C$$

Proposition by Harris: $H = \lambda_1\lambda_2 - \underbrace{0.04(\lambda_1 + \lambda_2)^2}$

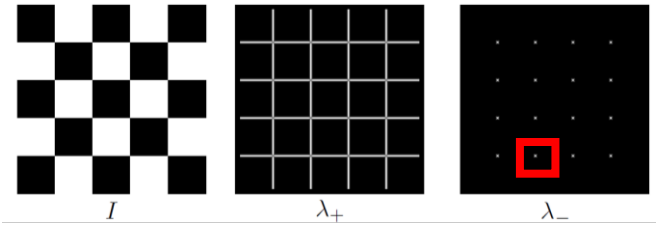
Downweights edges where $\lambda_1 \gg \lambda_2$



Harris Value

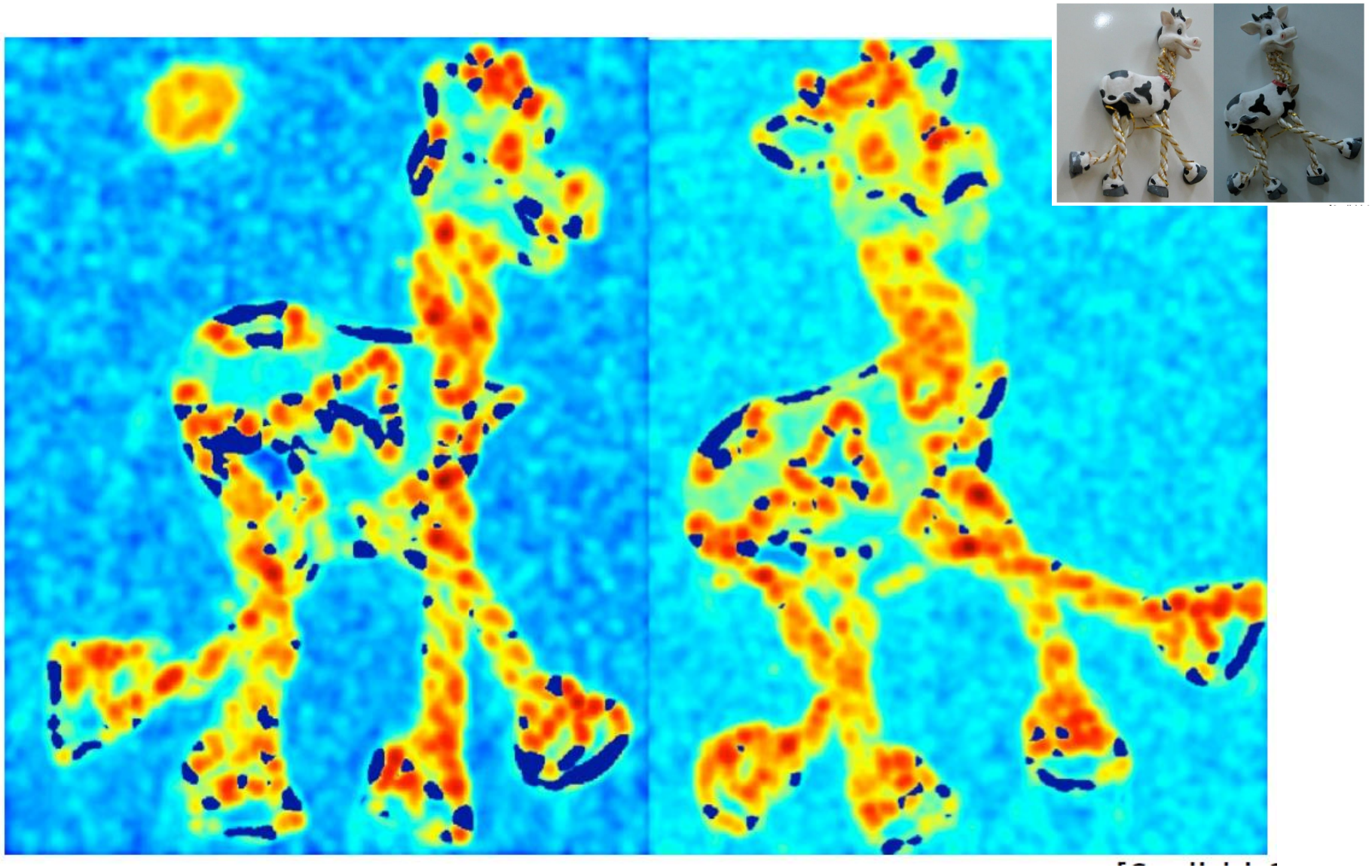


Smallest eigenvalue

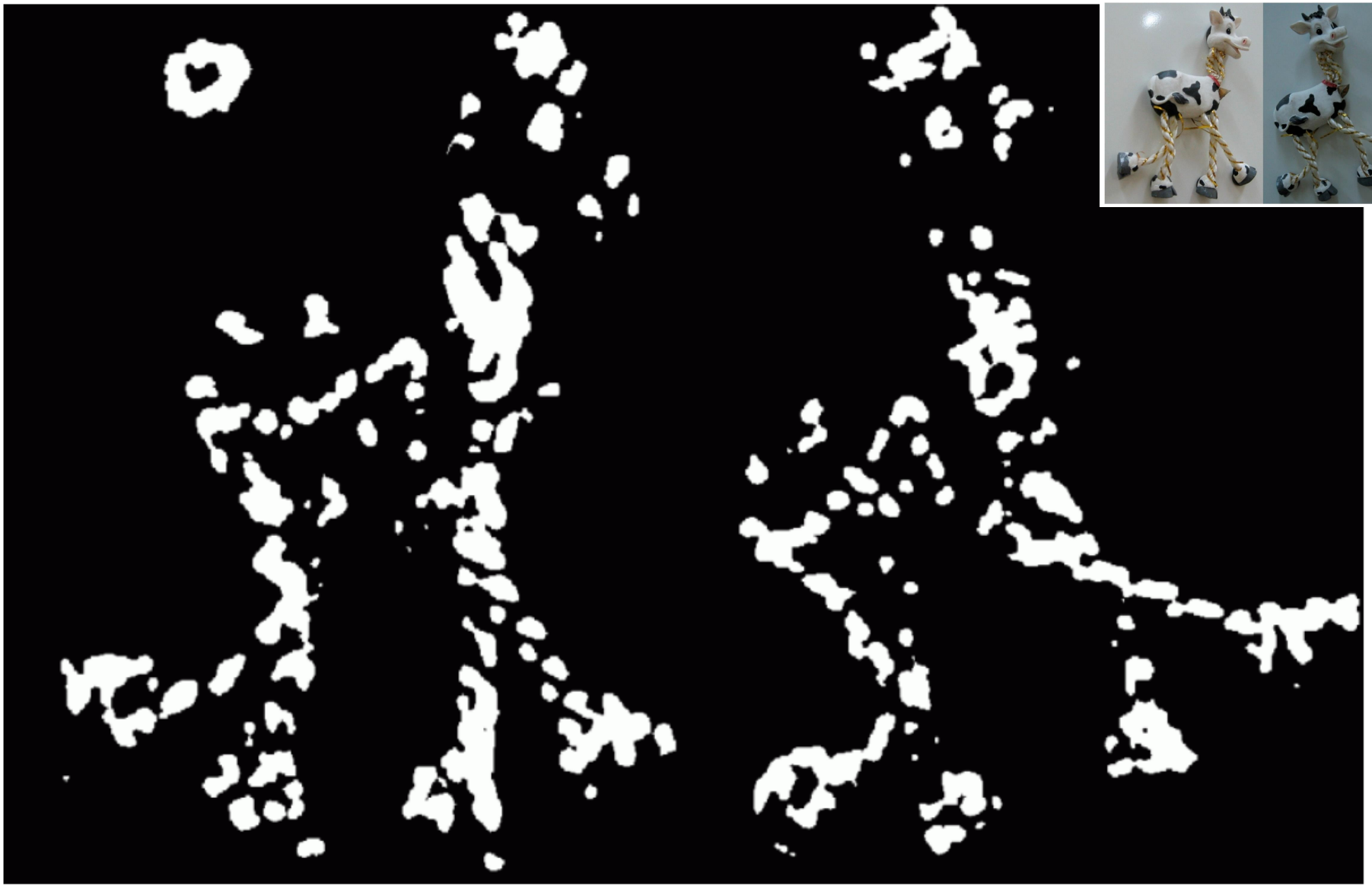


Harris Corners - example

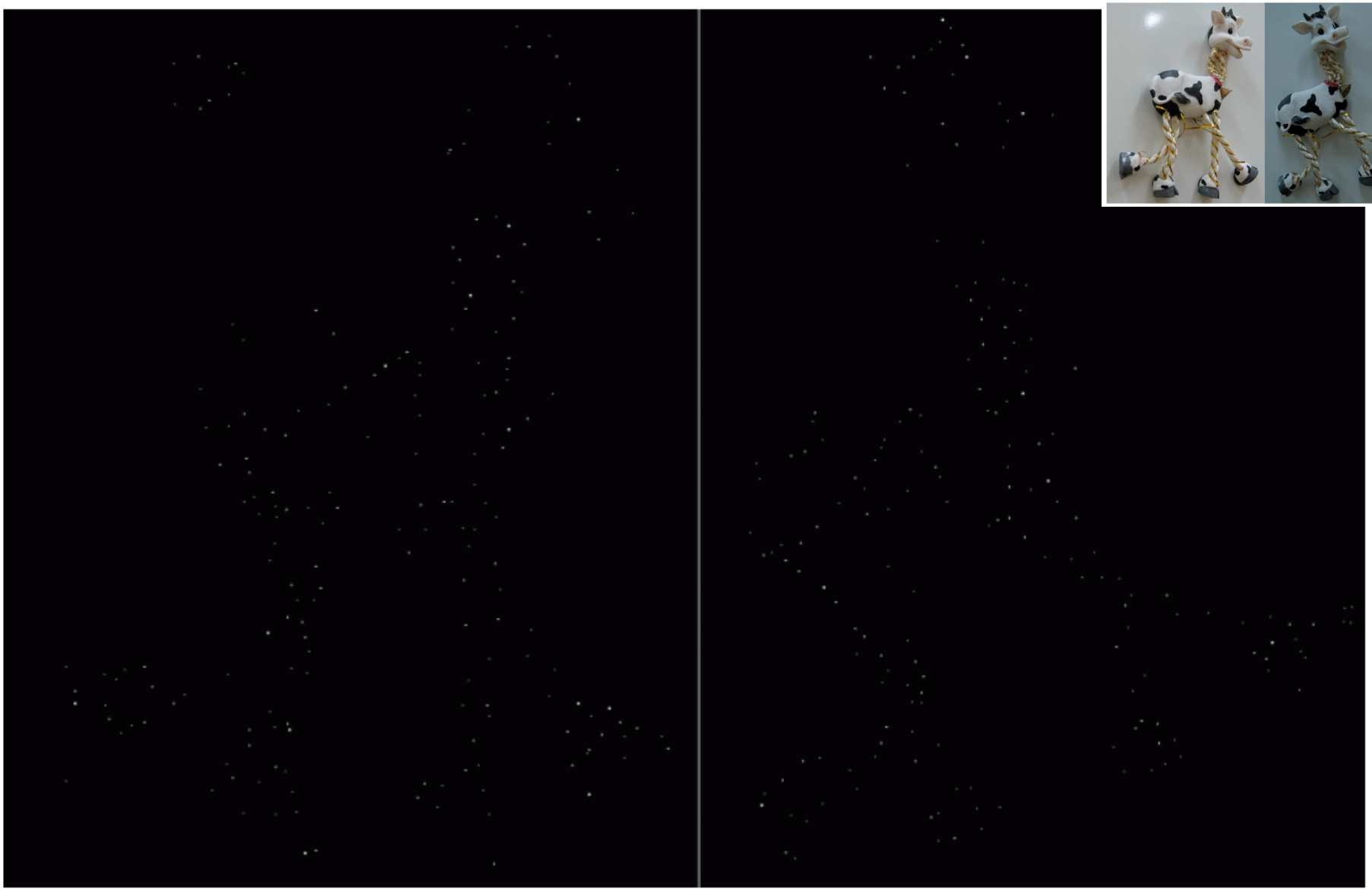
h-score (red- high, blue - low)



Threshold (H-score > value)

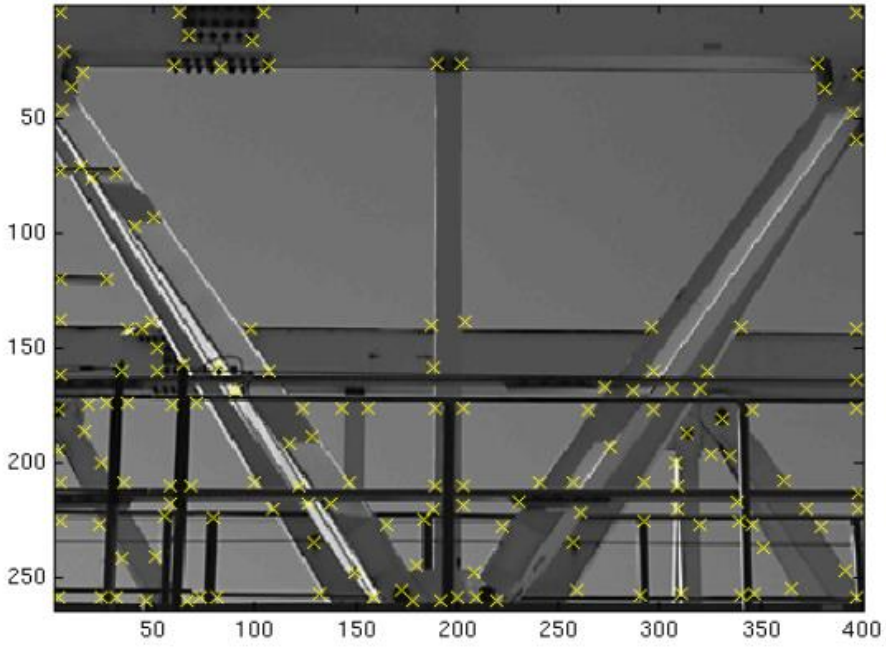


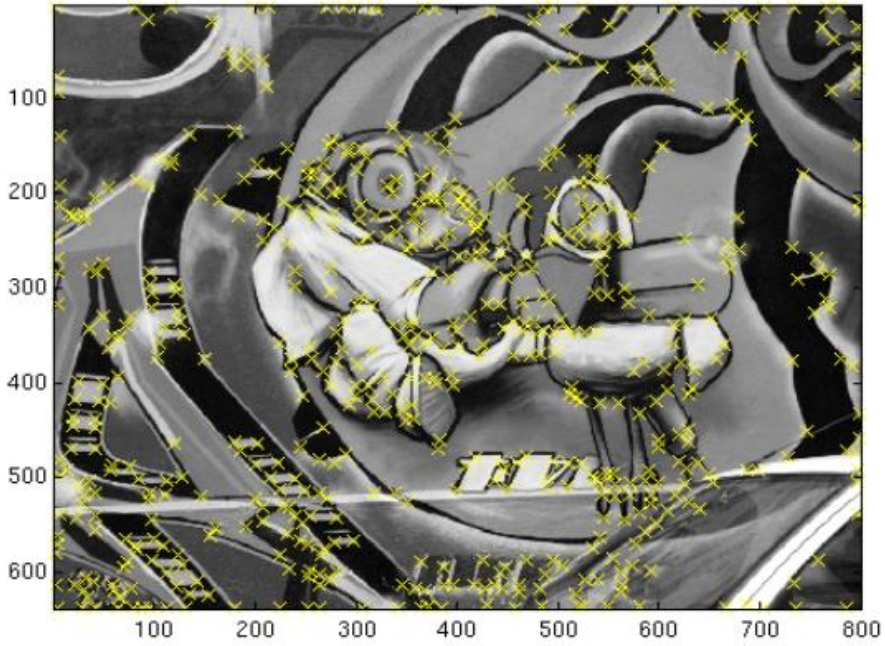
Non-maximum suppression



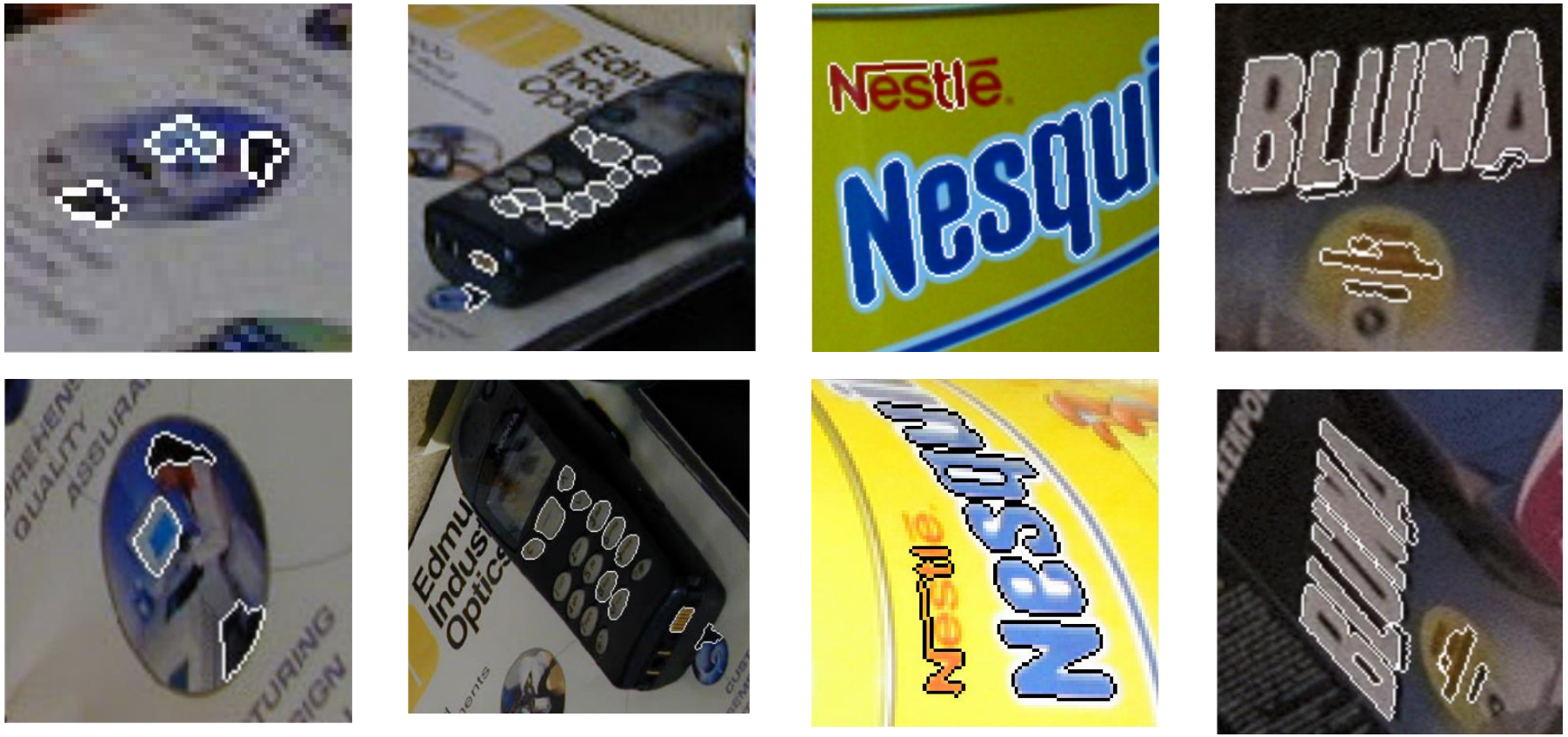
Harris corners in red

Other examples





Maximally stable extremal regions



- Invariant to affine transformation of gray-values
- Both small and large structures are detected

Literature

There is a large body of literature on detectors and descriptors (later lecture)

A comparison paper
(e.g. what is the most robust corner detectors):

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir: A Comparison of Affine Region Detectors (IJCV 2006)

Roadmap: Basics of Digital Image Processing

- Images
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4) – main focus
  - Linear filtering
  - Non-linear filtering
- Fourier Transformation (ch. 3.4)
- Multi-scale image representation (ch. 3.5)
- Edges detection and linking (ch. 4.2)
- Line detection (ch. 4.3)
- Interest Point detection (ch. 4.1.1)
- Using multiple Images: Define Challenges

Difference between Appearance and Geometry

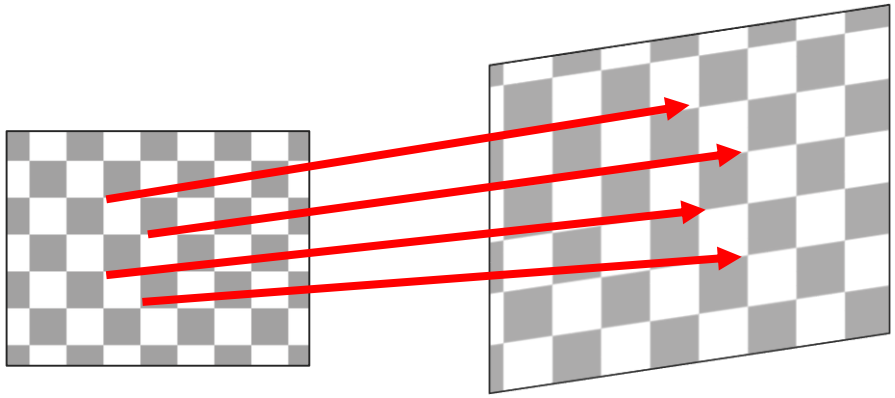


Image 1 Image 2

Does this look like a cotrrect match?

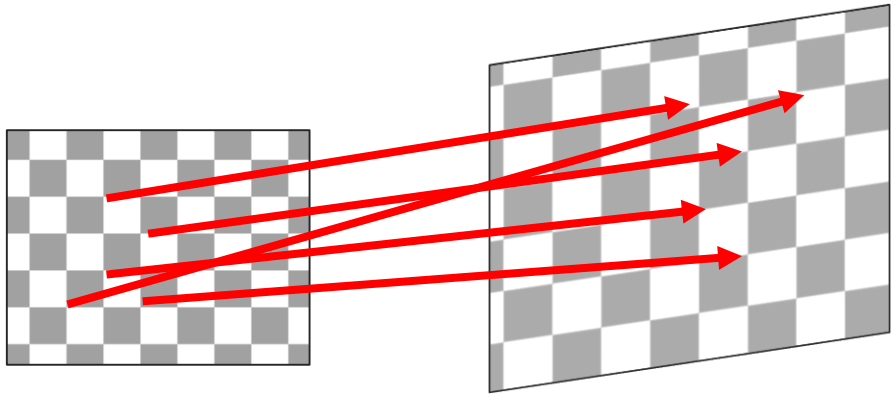


Image 1 Image 2

Does this look like a cotrrect match?

Appearance based matching:

Neighborhodd looks quite similiar (descriptor)

Geometry based matching:

1) Assume sensible camera model. We will see that: Given 4 matching points on a surface defines how other points on the surface will match

The 3D case

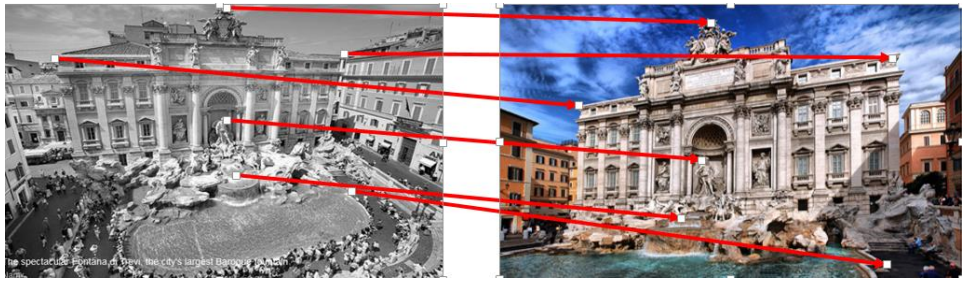
Illustration of the general 3D case

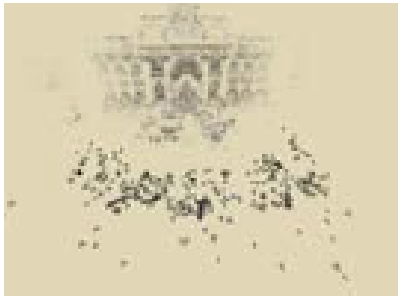
Appearance based matching:

Geometry based matching:

1) Assume sensible camera model. We will see that: Given 7 matching 3D points defines how other 3D points match.

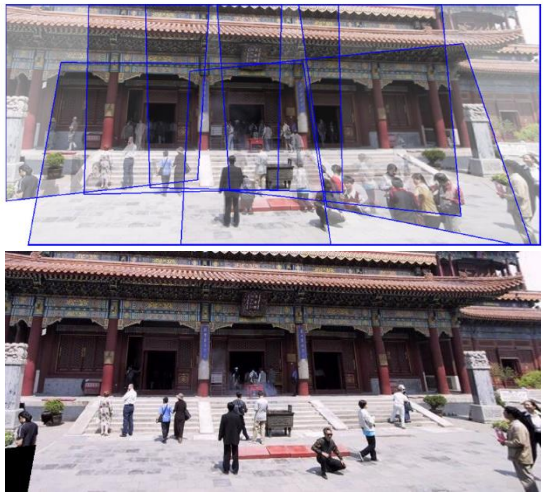
Sparse versus Dense Matching: Tasks and Applications





Tasks:

- Find places where we could match features (points, lines, regions, etc)
- Extract appearance - features descriptors
- Find all possible (putative) appearance matches between images
- Verify with geometry



For what applications is sparse matching enough:

- Sparse 3D reconstruction of a rigid scene
- Panoramic stitching of a rotating / translating camera

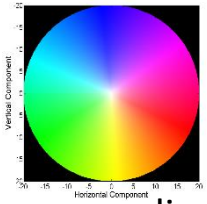
Building Rome on a cloudless day

The old city of Dubrovnik

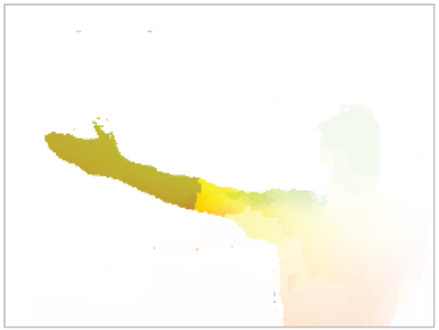
[Frahm et al. ECCV ´10]

Sparse versus Dense Matching

Kinect RGB and Depth data input



Flow encoding



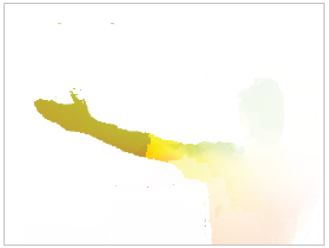
Dense flow:
frame 1->2

Dense flow:
frame 2->1

3D view interpolation

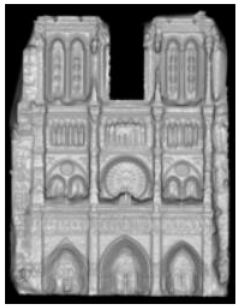
Sparse versus Dense Matching: Tasks and Applications





Tasks (all in one):

- Find for each pixel the 2D/3D/6D displacement (using both appearance and geometry)
- Find points which are occluded



[Goesele et al. ICCV 07]

For what applications is dense matching needed:

- Dense reconstruction of rigid scene
- 3D reconstruction of a non-rigid scene

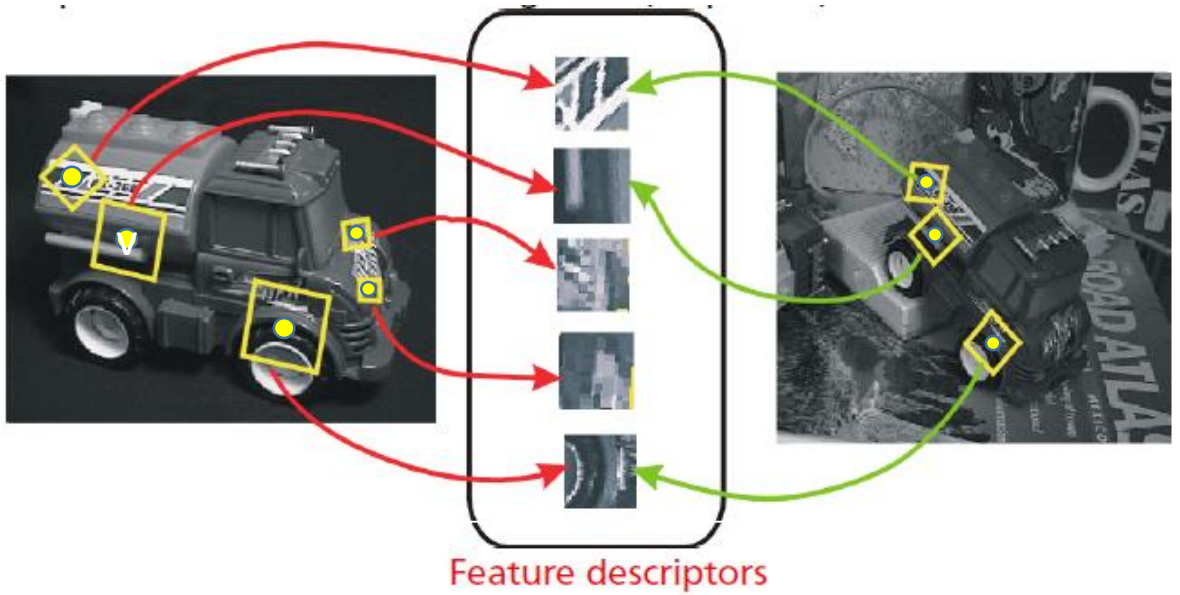
Using multiple Images: Define Challenges

A road map for the next five lectures

- L4: Geometry of a Single Camera
  and Image Formation Process
- L5: Sparse Matching two images: Appearance
- L6: Sparse Matching two images: Geometry
- L7: Sparse Reconstructing the world (Geometry of n-views)
- L8: Dense Geometry estimation
  (stereo, flow and scene flow, registration)

Outlook – matching 2 Images (appearance & geometry)

- Find interest points (including different scales)
- Find orientated patches around interest points to capture appearance
- Encode patch in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry



Reading for next class

This lecture:

- Chapter 3.5: multi-scale representation
- Chapter 4.2 and 4.3 - Edge and Line detection
- Chapter 4.1.1 Interest Point Detection

Next lecture:

- Chapter 2 (in particular: 2.1, 2.2) – Image formation process
- And a bit of Hartley and Zisserman – chapter 2