Computer Vision I - *Filtering and Feature detection*

Carsten Rother

30/10/2015

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

Reminder: Convolution

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

$$g = f * h$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

image
$f(x, y)$

$*$

Centred at 0,0

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

filter (kernel)
$h(x, y)$

$=$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

filtered image
$g(x, y)$

smaller output?

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

“the image f is implicitly mirrored”

Reminder: Application: Noise removal

- Noise is what we are not interested in:
  Sensor noise (Gaussian, shot noise), quantisation artefacts, light fluctuation, etc.

- Typical assumption is that the noise is not correlated between pixels

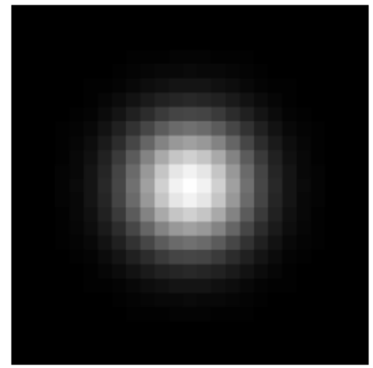
- Basic Idea: neighbouring pixel contain information about intensity

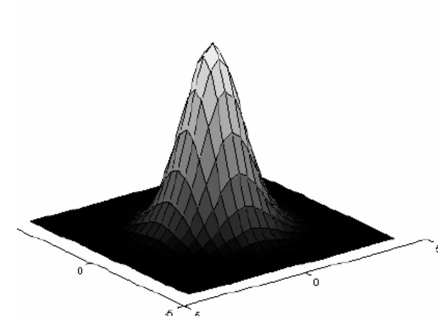
$$\begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 20 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 3 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

Reminder: Gaussian (Smoothing) Filters

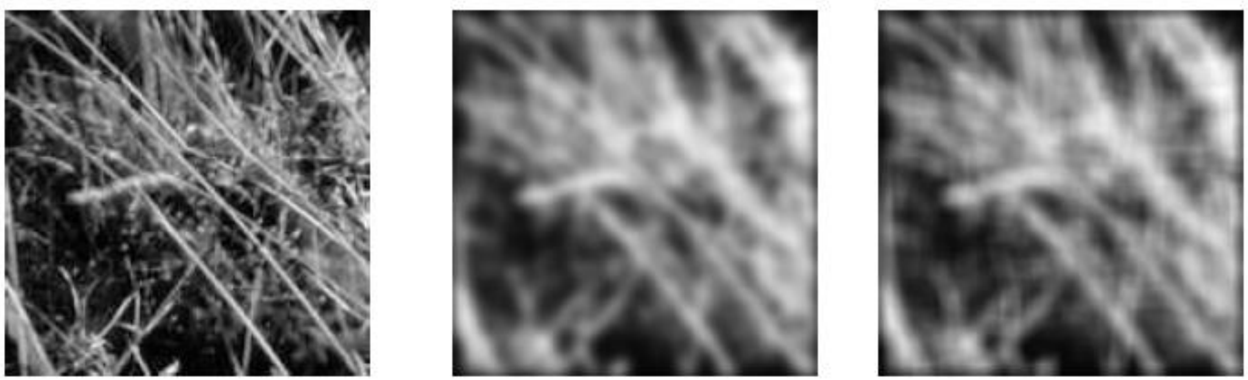
- Nearby pixels are weighted more than distant pixels
- Isotropic Gaussian (rotational symmetric)

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





Original Image Gaussian-filtered image Box-filtered image



The box filter does noise removal

- Box filter takes the mean in a neighbourhood

Image

Noise

Pixel-independent
Gaussian noise added

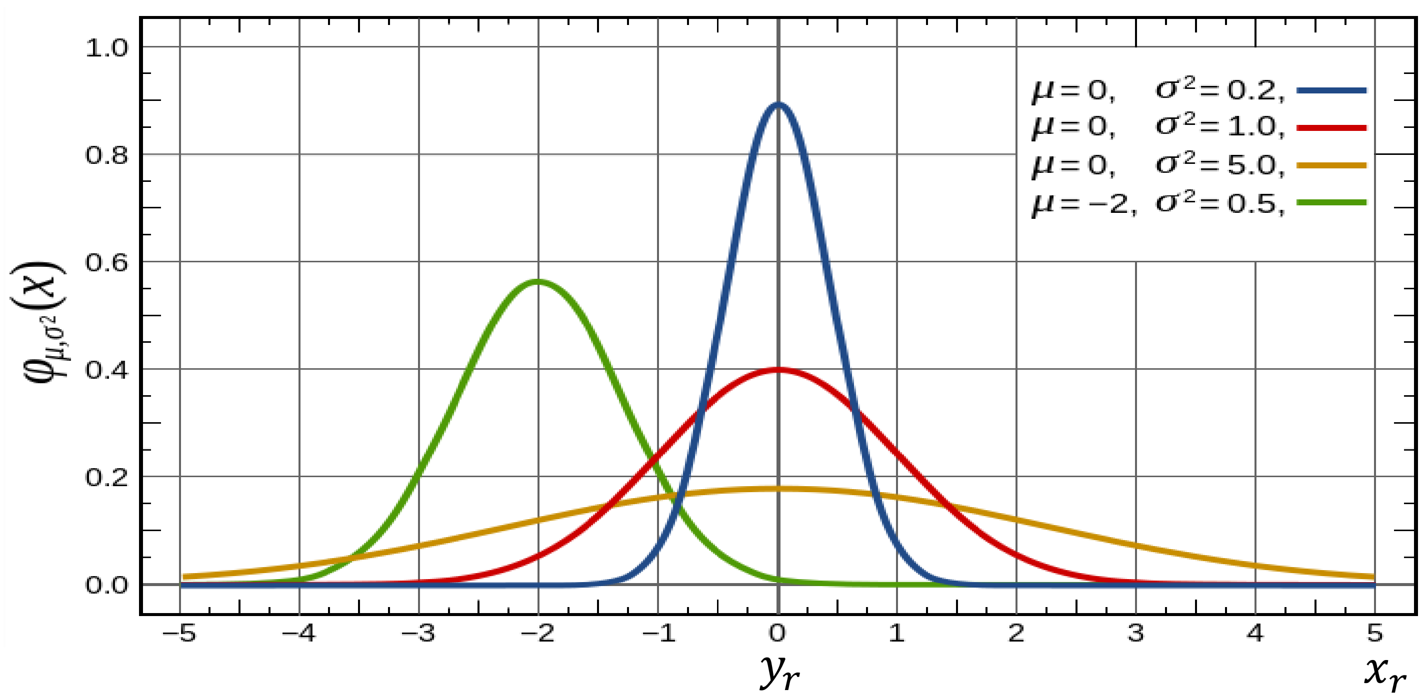
$$\frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Filtered Image

Derivation of the Box Filter

- y_r is true gray value (color)
- x_r observed gray value (color)
- Noise model: Gaussian noise

$$p(x_r|y_r) = N\,(x_r; y_r, \sigma) \sim \exp[-\frac{||x_r - y_r||^2}{2\sigma^2}]$$



Derivation of Box Filter

Comments:
1) $\sim$ means equal up to scale (i.e. $5x \sim x$)
2) $\|a\|^2 = \left(\sqrt{a^2}\right)^2 = a^2$ (i.e. squared L2 norm)
3) $|a|$ is norm of a (i.e. L1 norm)

Further assumption: independent noise

$$p(x|y) \sim \prod_r \exp[-\frac{\|x_r - y_r\|^2}{2\sigma^2}]$$

Find the most likely solution for the true signal y

Maximum-Likelihood principle (probability maximization):

$$y^* = argmax_y \underbrace{p(y|x)}_{posterior} = argmax_y \frac{\overbrace{p(x|y)}^{likelihood}\,\overbrace{p(y)}^{prior}}{p(x)}$$

$p(x)$ is a constant (drop it), assume (for now) uniform prior $p(y)$.

We get:

$$p(y|x) = p(x|y) \sim \prod_r \exp[-\frac{\|x_r - y_r\|^2}{2\sigma^2}]$$

➡ the solution is trivial: $y_r = x_r$ for all r ☹

➡ additional assumptions about the **signal y** are necessary !!!

Derivation of Box Filter

Assumption: not uniform prior $p(y)$ but …
in a small vicinity $W(r) \subset D$ the "true" signal y_r is nearly constant

Maximum-a-posteriori:

Only one y_r in a window $W(r)$

$$p(y|x) \sim \prod_{r} \prod_{r' \in W(r)} \exp[-\frac{\|x_{r'} - y_r\|^2}{2\sigma^2}]$$

For one pixel r :

$$y_r^* = argmax_{y_r} \prod_{r' \in W(r)} \exp[-\frac{\|x_{r'} - y_r\|^2}{2\sigma^2}]$$

take neg. logarithm:

$$y_r^* = argmin_{y_r} \sum \frac{\|x_{r'} - y_r\|^2}{2\sigma^2}$$

Note, Log-function is monotonically increasing hence minimum does not change, i.e. $x_1 < x_2$ then $\log x_1 < \log x_2$

Derivation of Box Filter

Let us minimize:

$$y_r^* = argmin_{y_r} \sum_{r' \in W(r)} \| x_{r'} - y_r \|^2$$

factor $1/2\sigma^2$ is irrelevant

Take derivative and set to 0:

$$F(y_r) = \sum_{r' \in W(r)} \| x_{r'} - y_r \|^2$$

$|W|$ means number of pixel in window W

$$\frac{\partial F}{\partial y_r} = -2 \sum_{r'} (x_{r'} - y_r) = -2 \left(\sum_{r'} x_{r'} - |W| \, y_r \right) \overset{!}{=} 0$$

$$\Rightarrow \quad y_r^* = \frac{1}{|W|} \sum_{r'} x_{r'}$$ (the average)

Box filter is optimal under pixel-independent, Gaussian Noise assumption and assuming that the signal is constant in a window.

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

Non-linear filters

- There are many different non-linear filters.
  (see Image Processing in SS16)

- We only look at the Median filter, Bilateral filter and Joint Bilateral Filter

Shot noise (Salt and Pepper Noise)

Gaussian
filtered

Original + shot noise
(a random number of
independent pixels have a
random value)

Median
filtered

Another example

Original

Noisy

Mean

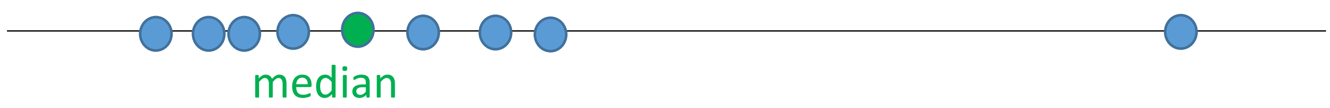
Median

Median Filter

Replace each pixel with the median in a neighbourhood:

5	6	5
4	20	5
4	6	5

5	6	5
4	5	5
4	6	5



Median filter: order the values and take the **middle** one

- No strong smoothing effect since values are not averaged
- Very good to remove outliers (shot noise)
- Used a lot for post processing of outputs (e.g. optical flow)

Median Filter: Derivation

Reminder, for Gaussian noise we did solve the following Maximum Likelihood Estimation Problem:

$$y_r^* = argmax_{y_r} \underbrace{\prod_{r' \in W(r)} \exp[-\frac{\|x_{r'} - y_r\|^2}{2\sigma^2}]}_{p(y|x)} = argmin_{y_r} \sum_{r' \in W(r)} \|x_{r'} - y_r\|^2 = 1/|W| \sum_{r' \in W(r)} x_{r'}$$

Mean and median are quite Different

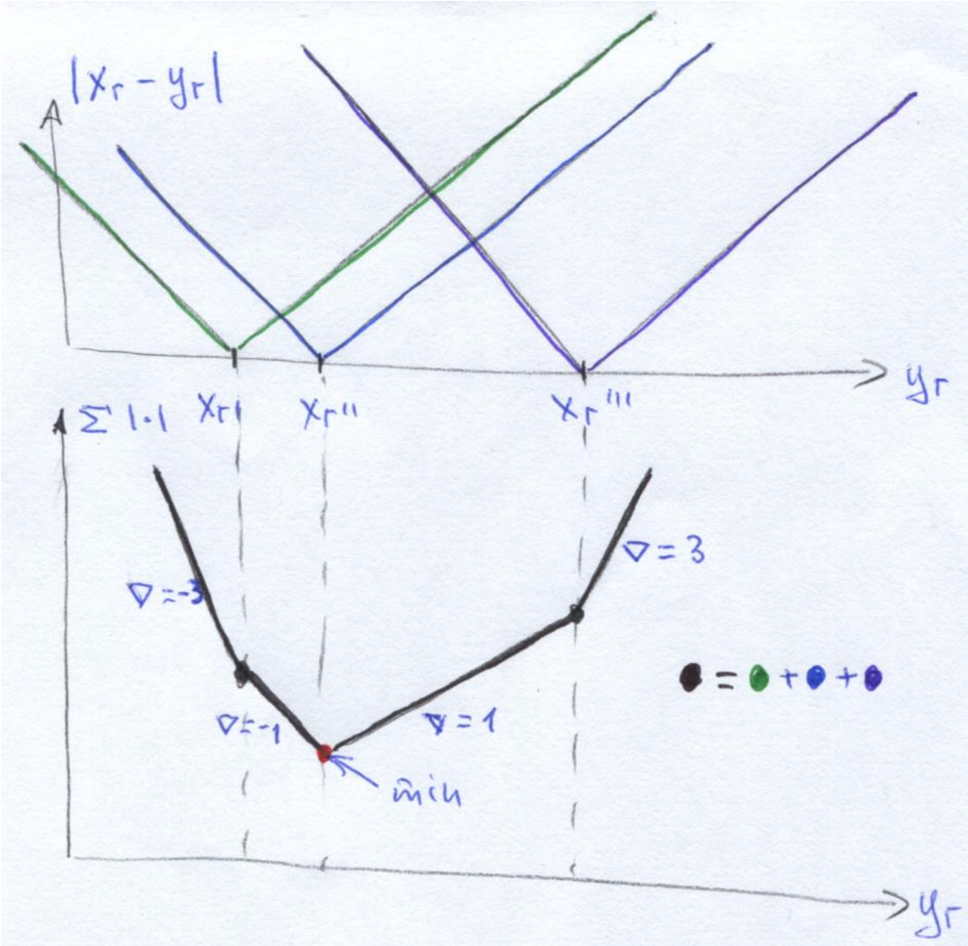
For Median we solve the following problem:

(see next slide)

$$y_r^* = argmax_{y_r} \prod_{r' \in W(r)} \exp[-\frac{|x_{r'} - y_r|}{2\sigma^2}] = argmin_{y_r} \sum_{r' \in W(r)} |x_{r'} - y_r| = Median\ (W(r))$$

This is a different noise distribution than Gaussian.

Median Filter Derivation



Optimal solution is the median of all values

minimize the following:

$$F(y_r) = \sum_{r' \in W(r)} |x_{r'} - y_r|$$

Problem: not differentiable ☹, good news: it is convex ☺

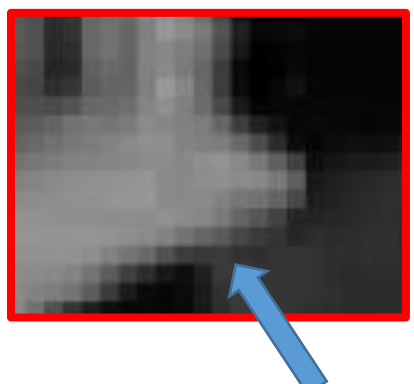
Bilateral Filter

More sharp

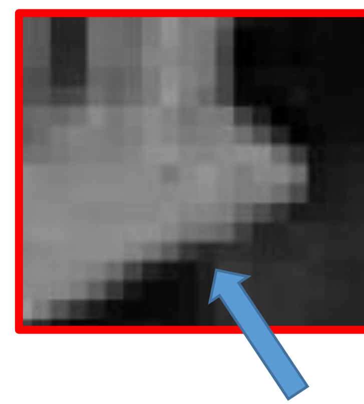
Original + Gaussian noise

Gaussian filtered

Bilateral filtered

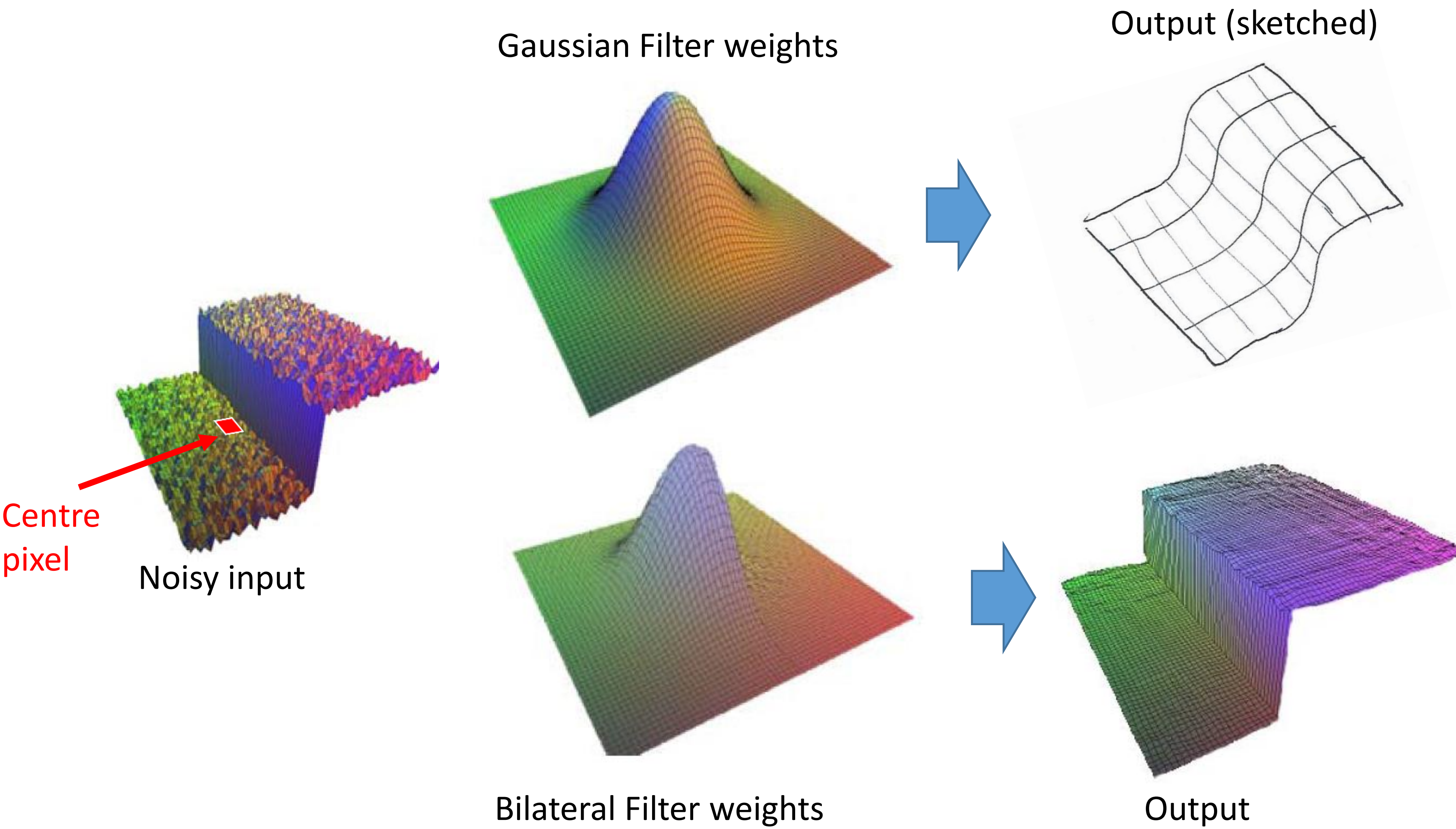


Edge over-smoothed



Edge not over-smoothed

Bilateral Filter – in Pictures



Bilateral Filter – in equations

Filters looks at: a) distance to surrounding pixels (as Gaussian)

b) Intensity of surrounding pixels

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

Linear combination

$$w(i,j,k,l) = \exp\left(-\underbrace{\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}}_{\text{Same as Gaussian filter}} - \underbrace{\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}}_{\text{Consider intensity}} \right)$$

Same as Gaussian filter *Consider intensity*

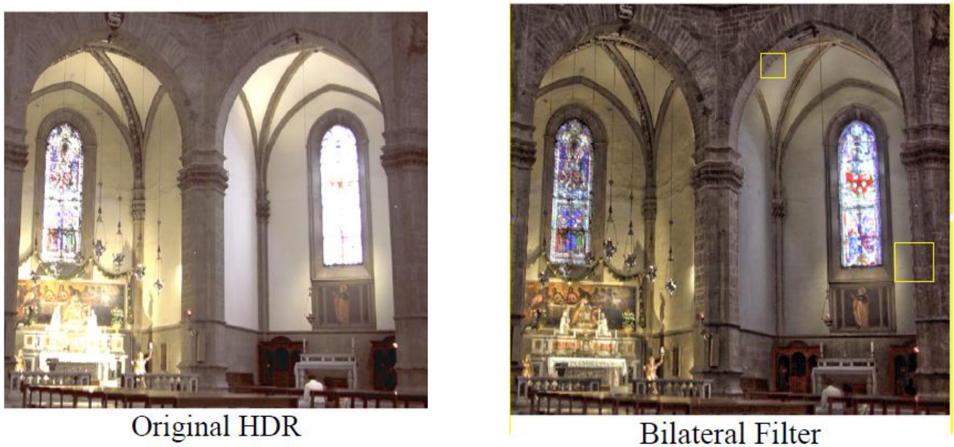
Problem: computation is slow $O(Nw)$; approximations can be done in $O(N)$

Comment: **Guided filter** is similar and can be computed exactly in $O(N)$

See a tutorial on: http://people.csail.mit.edu/sparis/bf_course/

Application: Bilateral Filter

Cartoonization



HDR compression (Tone mapping)

Joint Bilateral Filter

$$g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|\tilde{f}(i,j)-\tilde{f}(k,l)\|^2}{2\sigma_r^2}\right)$$

Same as Gaussian *Consider intensity*

f is the input image – which is processed

$\tilde{f}$ is a guidance image – where we look for pixel similarity

Application: Combine Flash and No-Flash

input image f

Guidance image $\tilde{f}$

Joint Bilateral Filter

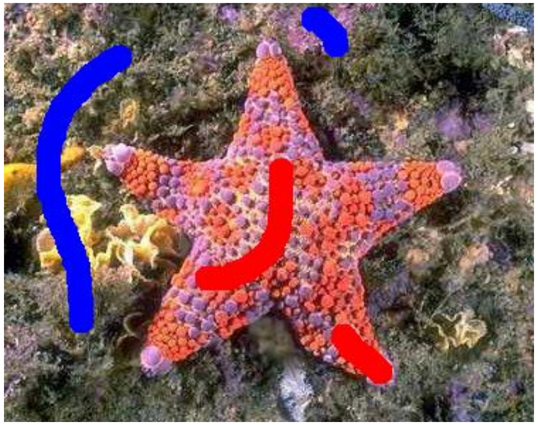
We don't care about absolute colors

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|\tilde{f}(i,j)-\tilde{f}(k,l)\|^2}{2\sigma_r^2}\right)$$

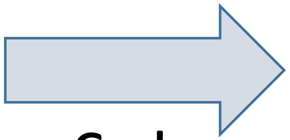
[Petschnigg et al. Siggraph '04]

Reminder: Model versus Algorithm

Example: Interactive Image Segmentation



$\boldsymbol{z} \ = \ (R, G, B)^n$



Goal

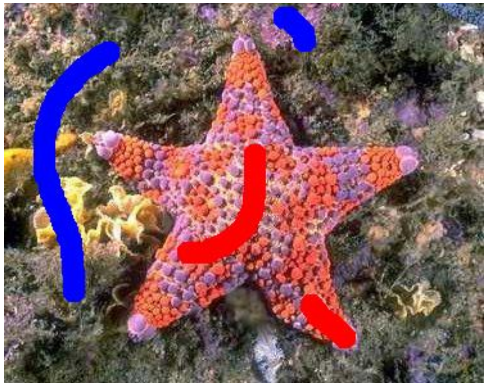
$\mathbf{x} = \{0,1\}^n$

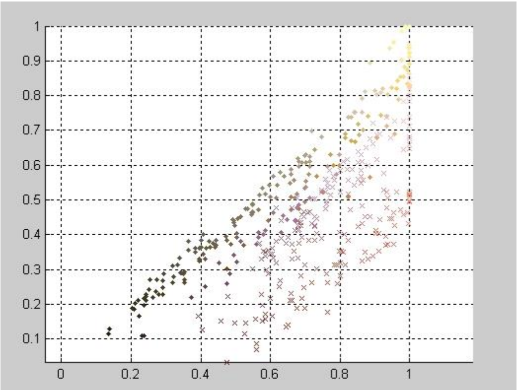
Given **z**; derive binary **x**:

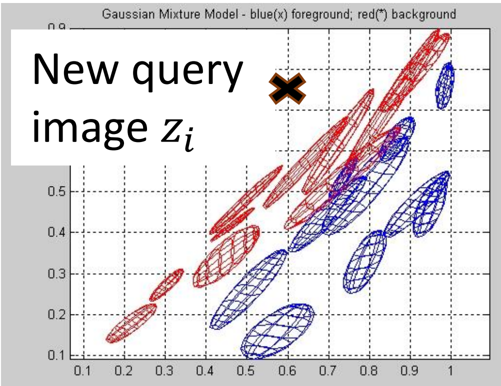
Model: Energy function $\boldsymbol{E}(\boldsymbol{x}) = \sum_i \theta_i(x_i) \ + \sum_{i,j} \theta_{ij}(x_i, x_j)$

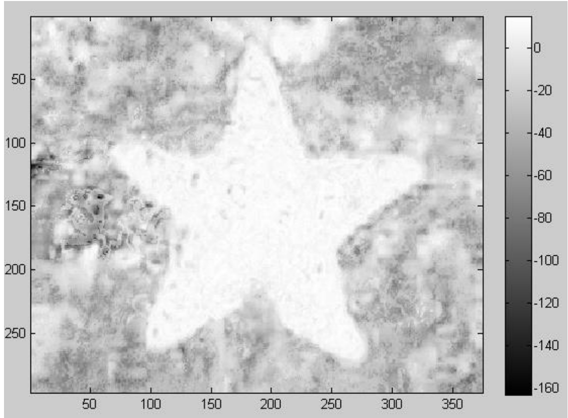
Algorithm to minimization: $\boldsymbol{x}^* = argmin_x \, E(\boldsymbol{x})$

Filtering for Binary Segmentation



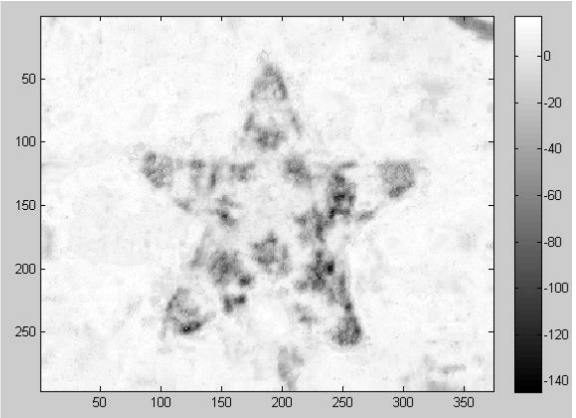






$\theta_i(x_i = 0)$

Dark means likely background

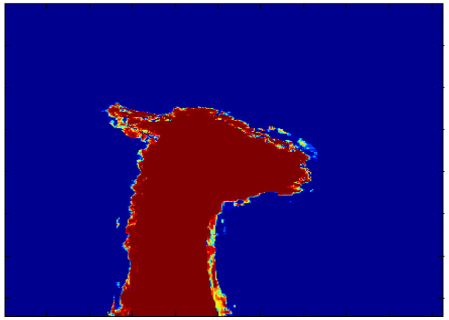


$\theta_i(x_i = 1)$

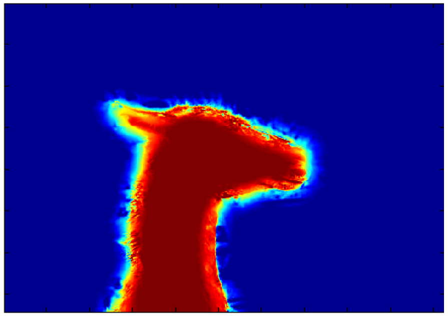
Dark means likely foreground

Optimum with unary terms only

Filtering for Binary Segmentation

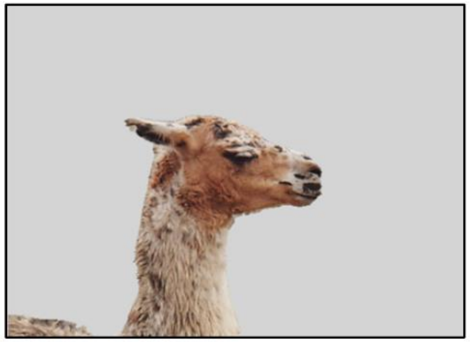


Ratio Image $\frac{\theta_i(x_i=1)}{\theta_i(x_i=0)}$ is the Input Image f

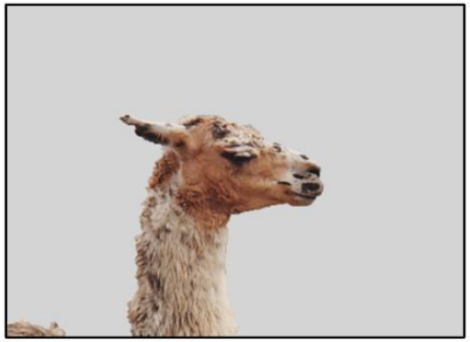


Filtered Ratio Image

Guidance Input Image $\tilde{f}$ (also shown user brush strokes)



Cost Volume filtering (Winner takes all Result)



Energy minimization

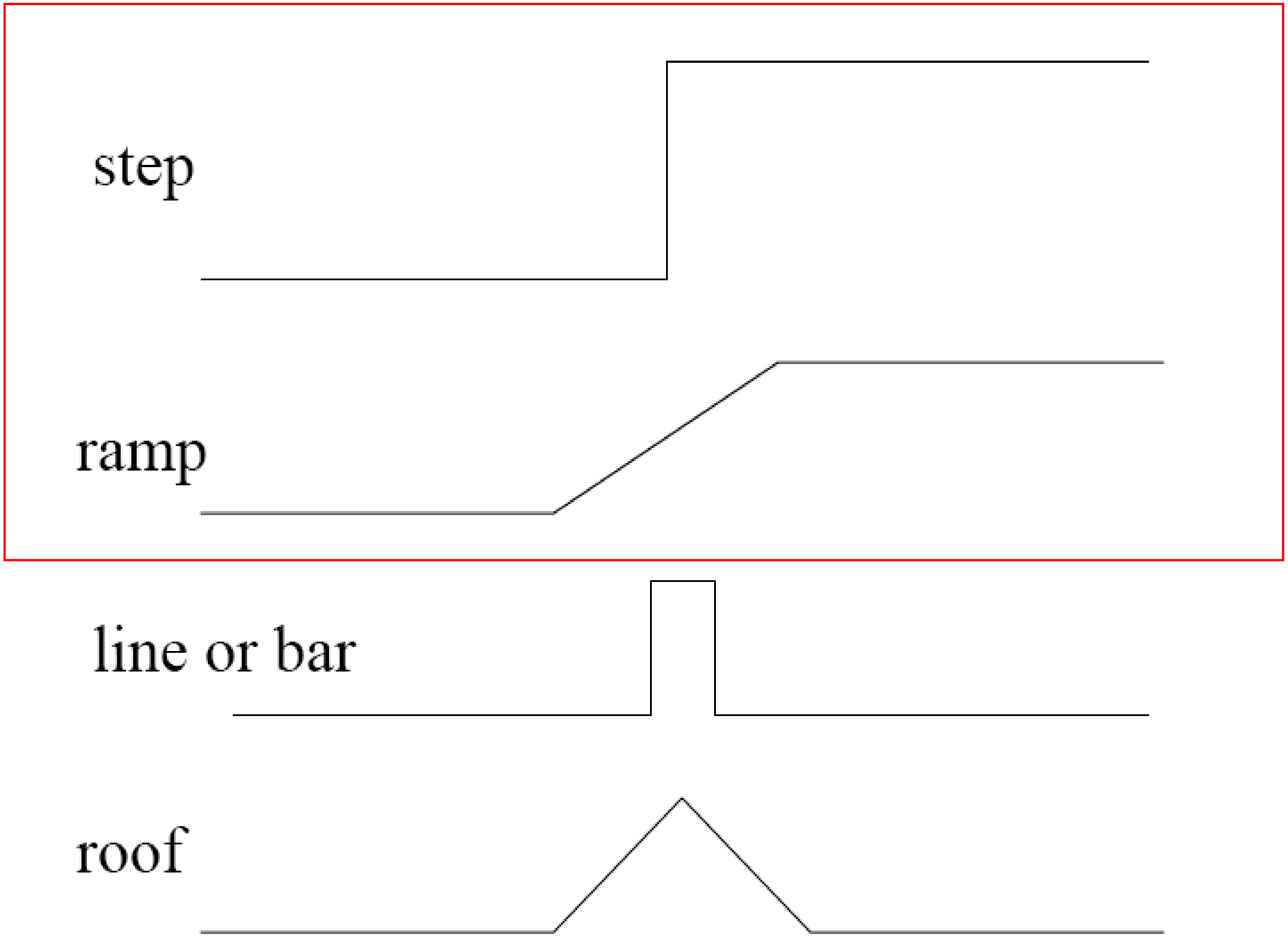
- Results are very similar: This is an alternative to energy minimization
- This can also be done for stereo, etc.

[C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, Fast Cost-Volume Filtering for Visual Correspondence and Beyond, CVPR 11]

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

Idealized edge types



We focus on this

What are edges ?

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

Image of 2 step edges

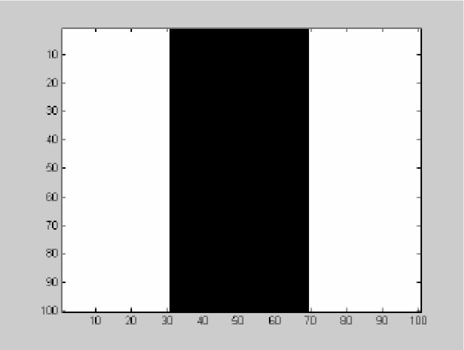
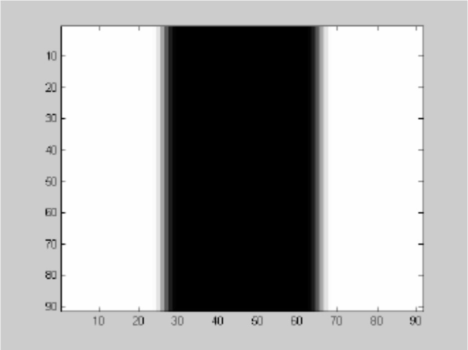
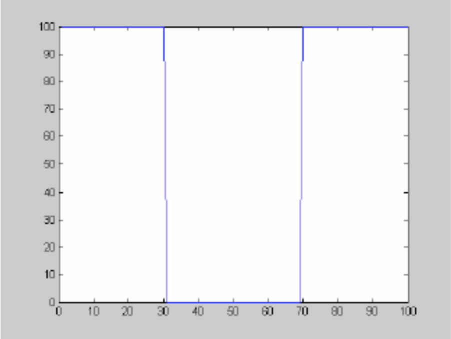


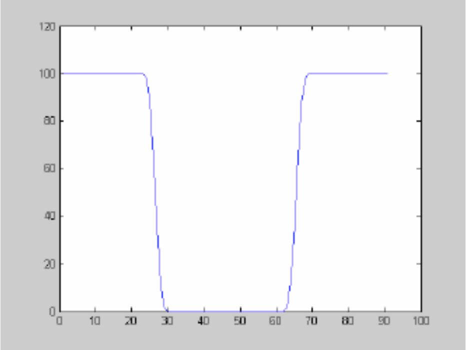
Image of 2 ramp edges



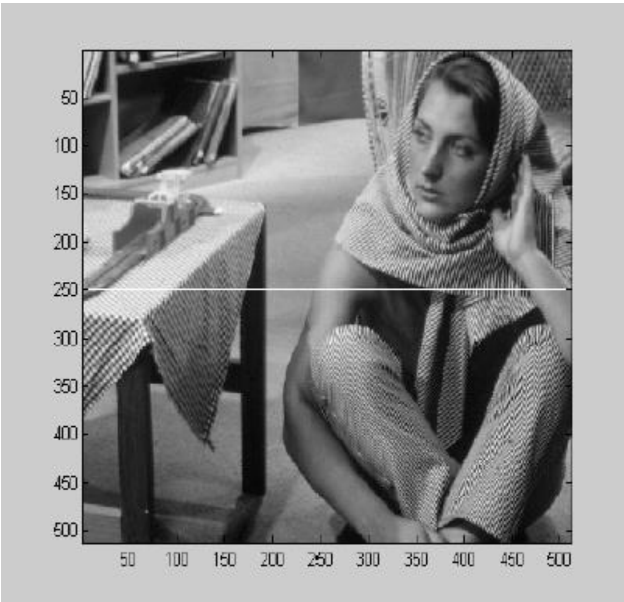
Slice through the image



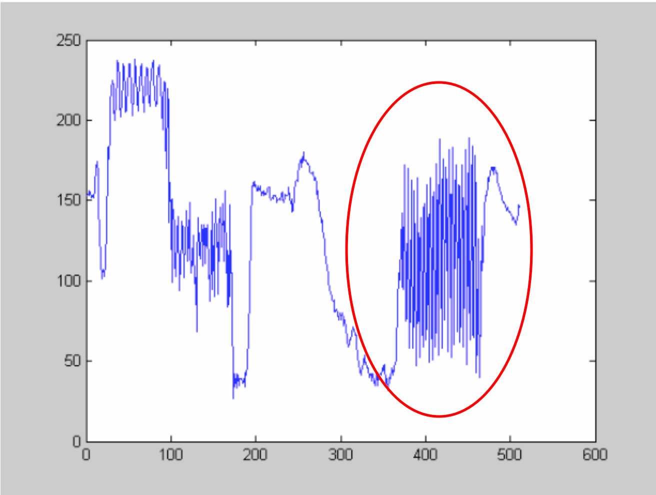
Slice through the image



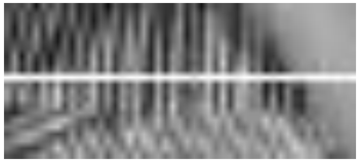
What are fast changes in the image?



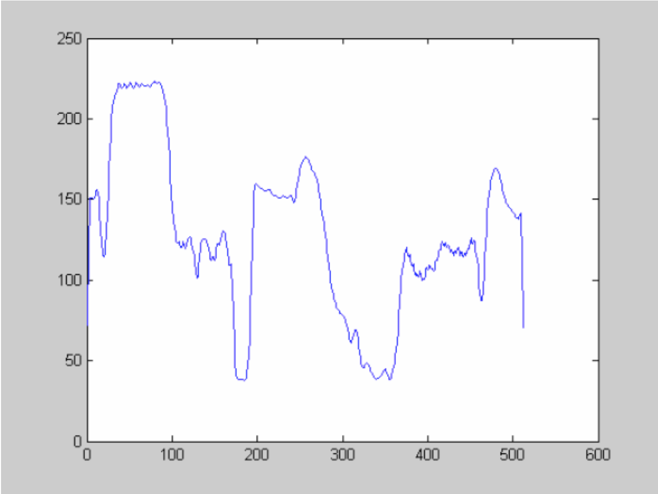
Image



Scanline 250



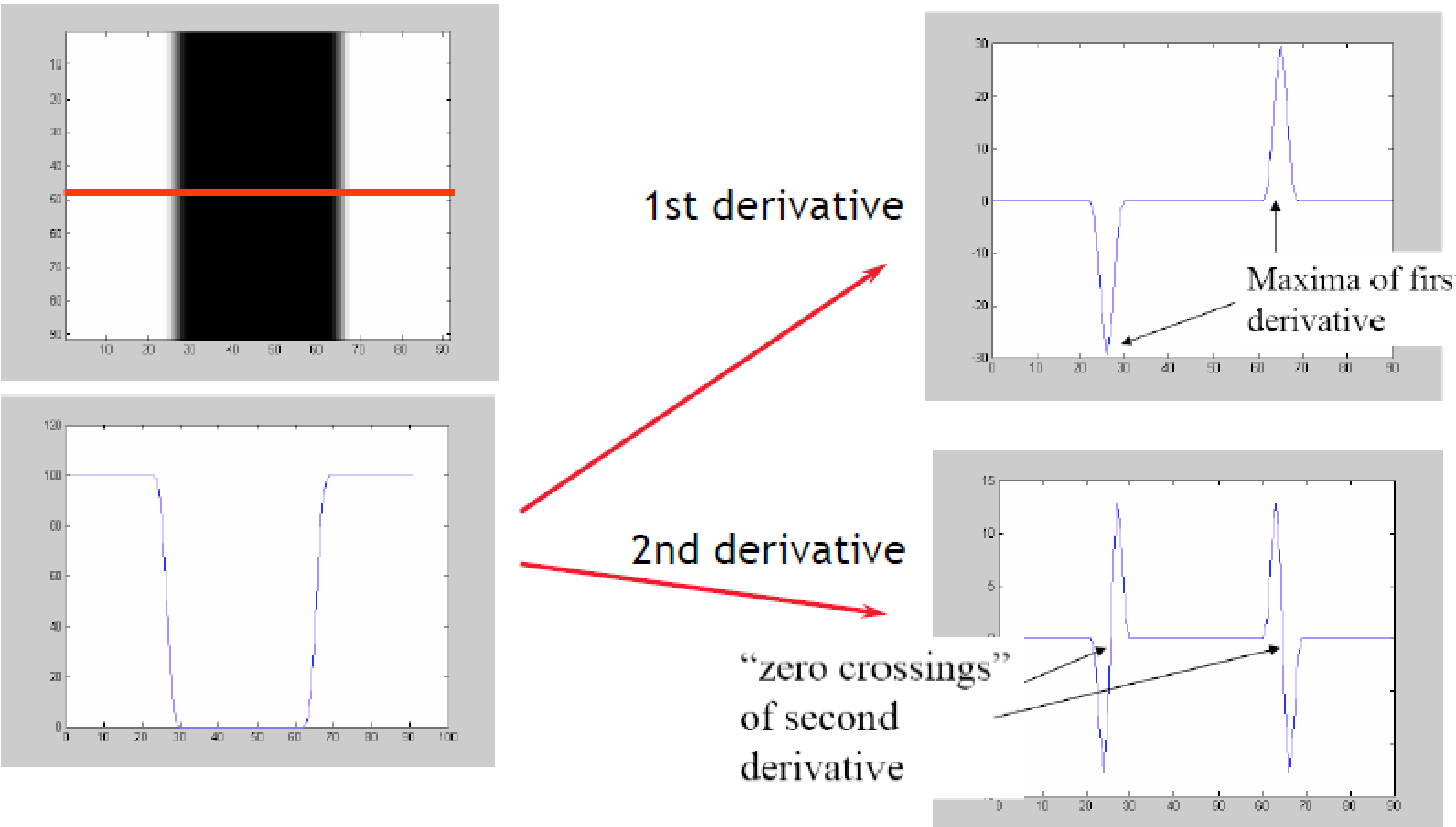
Texture or many edges?



Scanline 250 smoothed with Gausian

Edges defined after smoothing

Edges and Derivatives



We will look at this mainly

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:

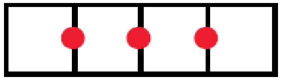
Forward differences: 1/2

-1	1

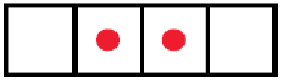
Central differences: 1/2

-1	0	1

Where is the derivative computed?



in between pixels



at pixels

Reminder: Seperable Filters

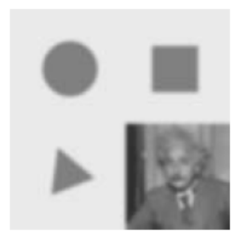
$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

(a) box, $K = 5$

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

(b) bilinear

$$\frac{1}{256}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 6 & 24 & 36 & 24 & 6 \\ \hline 4 & 16 & 24 & 16 & 4 \\ \hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array} \qquad \frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline \end{array}$$



(c) "Gaussian"

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array}$$

(d) Sobel

This is the centralized difference-operator

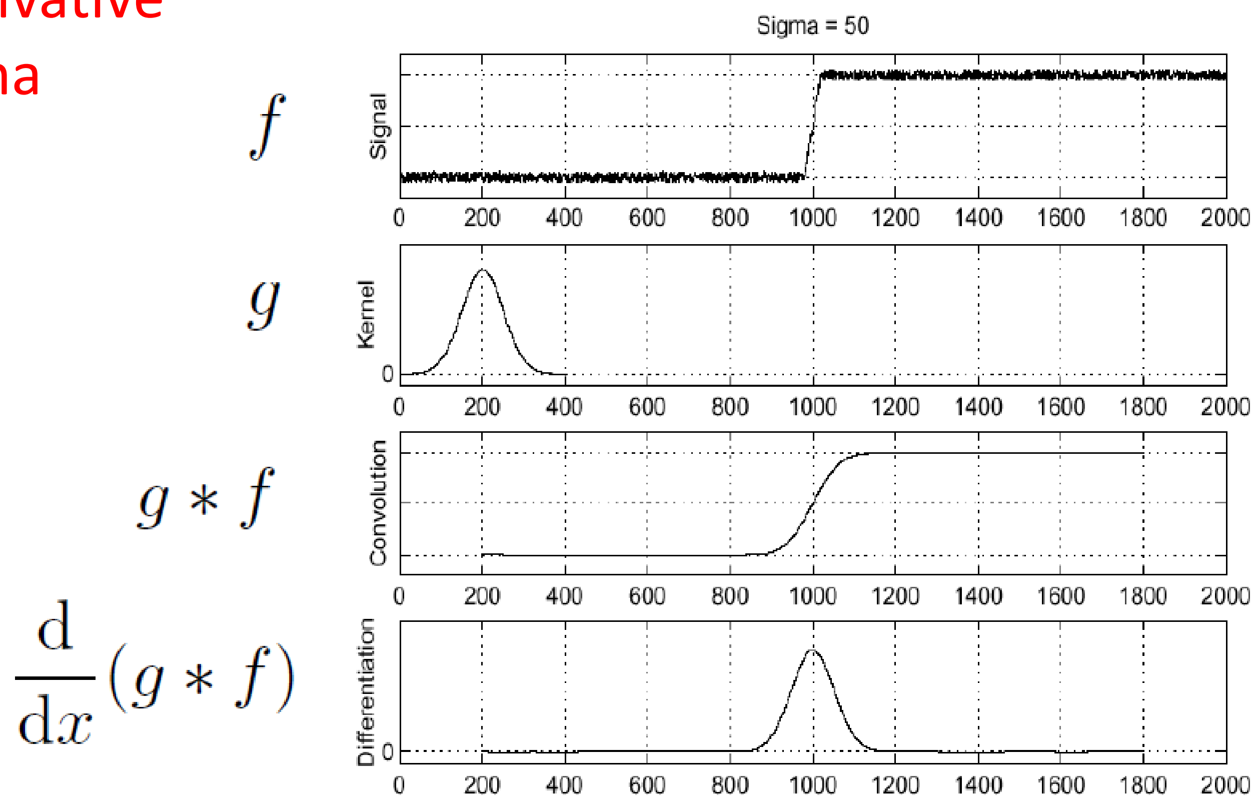
$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline -2 & 4 & -2 \\ \hline 1 & -2 & 1 \\ \hline \end{array} \qquad \frac{1}{2}\begin{array}{|c|c|c|}\hline 1 & -2 & 1 \\ \hline \end{array}$$

(e) corner

Edge Filter in 1D: Example

Based on 1st derivative

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



Edge Filtering in 1D

Simplification:
(saves one operation)

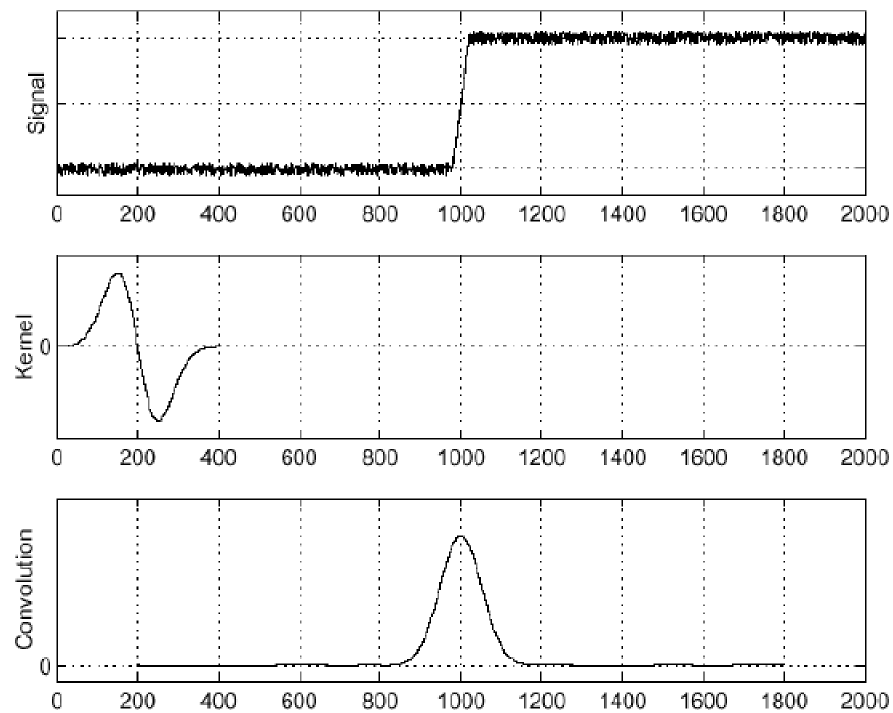
$$\frac{\mathrm{d}}{\mathrm{d}x}(g * f) = \left(\frac{\mathrm{d}}{\mathrm{d}x}g\right) * f$$

Derivative of Gaussian

f

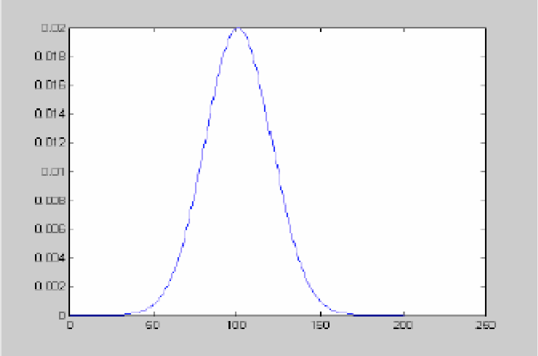
$\frac{\mathrm{d}}{\mathrm{d}x}g$

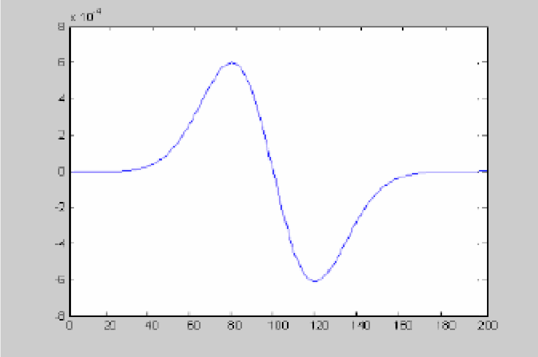
$\left(\frac{\mathrm{d}}{\mathrm{d}x}g\right) * f$



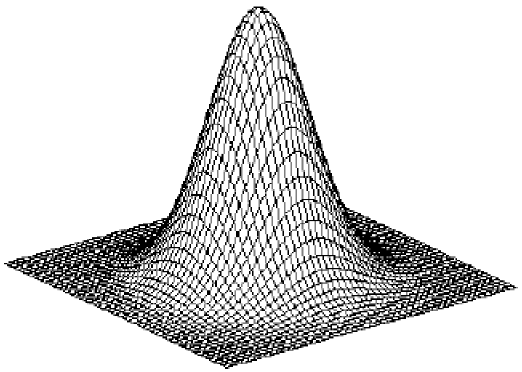
Edge Filtering in 2D

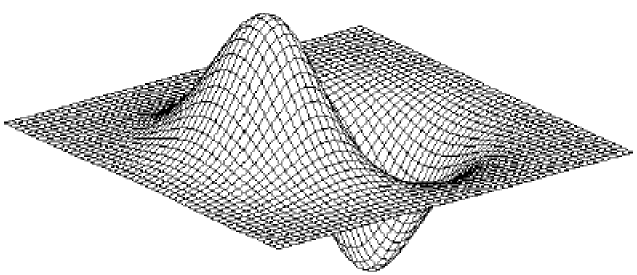
- Derivative in x-direction: $D_x * (G * I) = (D_x * G) * I$
  - in 1D:



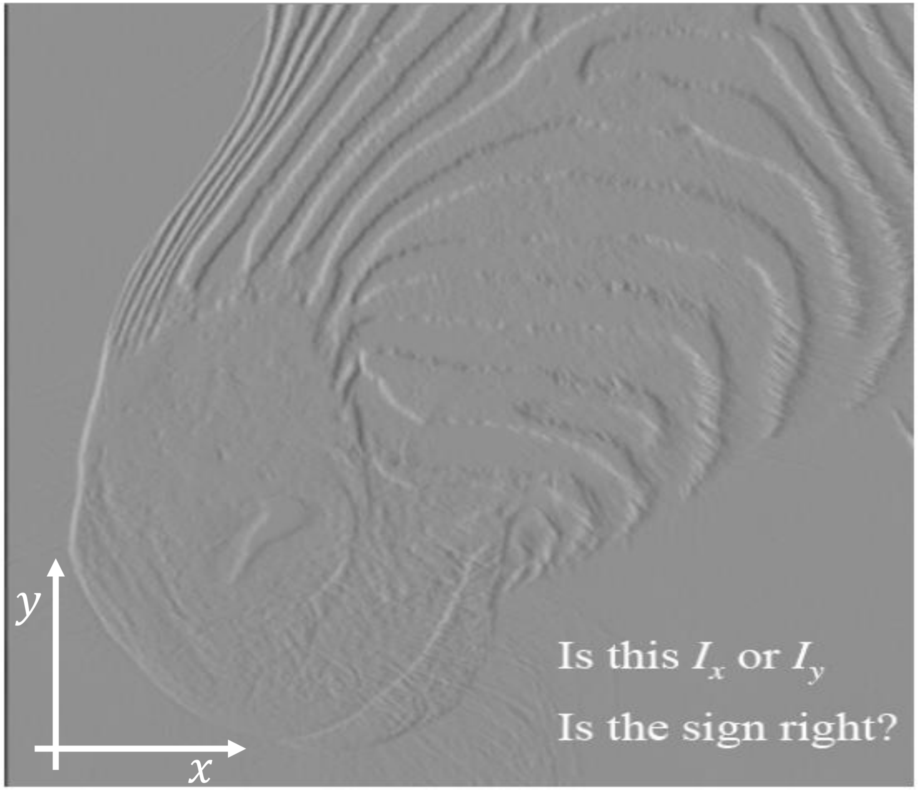


  - in 2D:



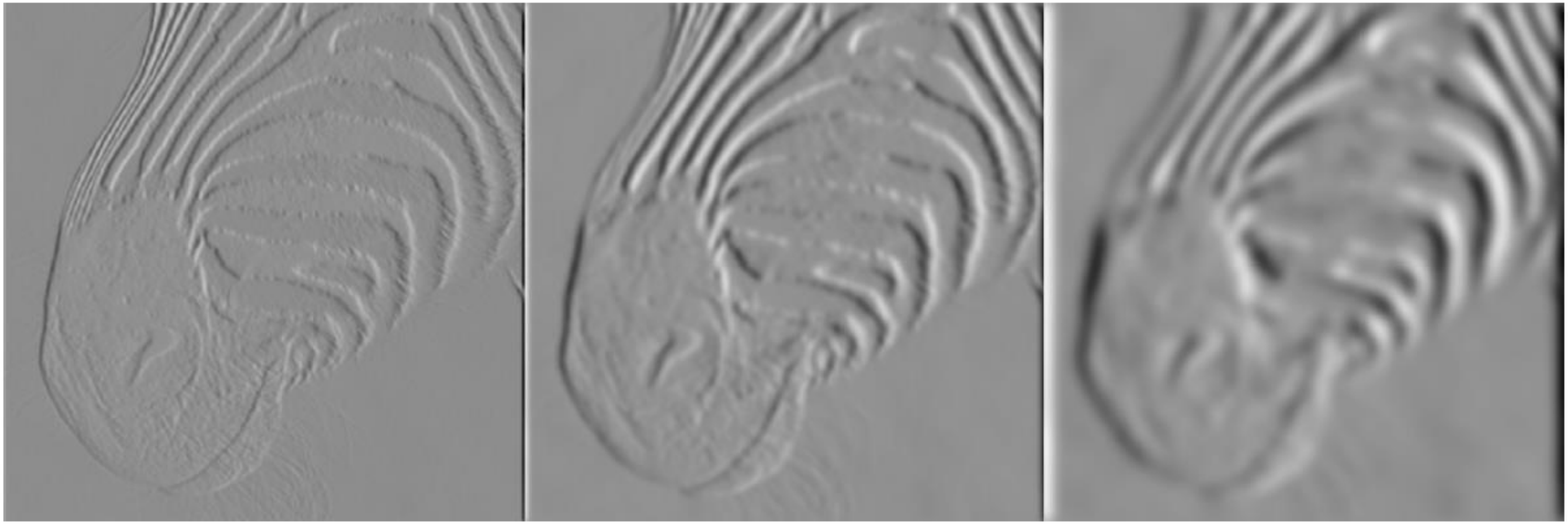


Edge Filter in 2D: Example

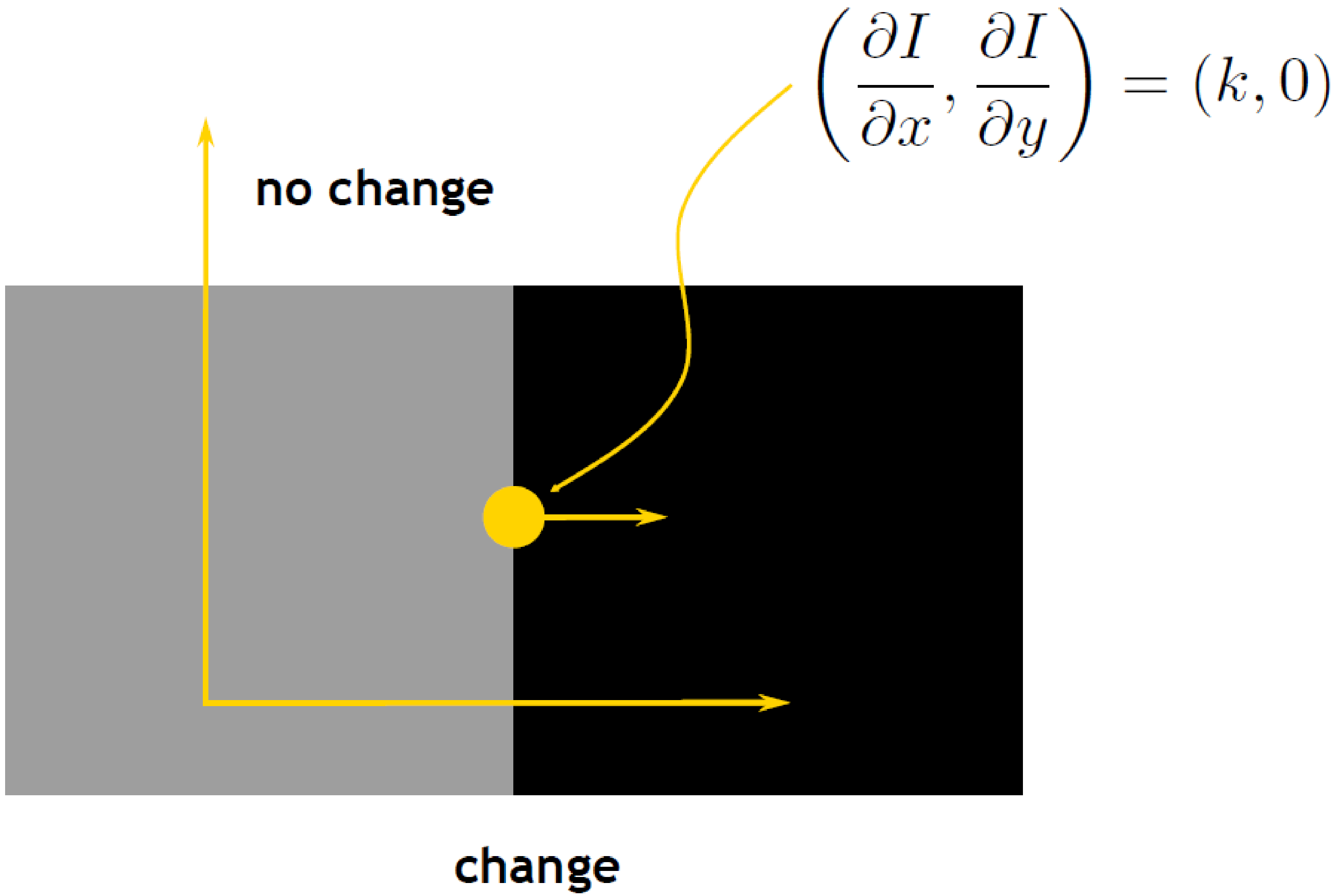


Edge Filter in 2D

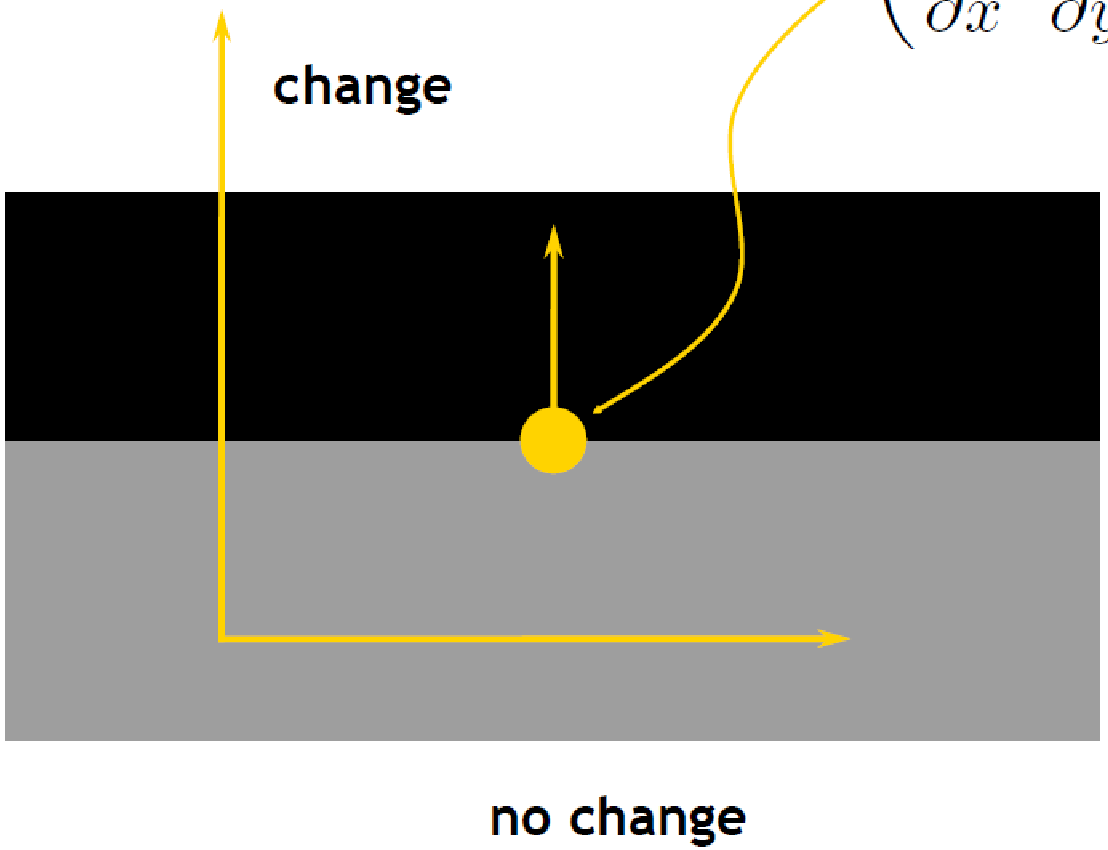
x-derivatives with different Gaussian smoothing



What is a gradient



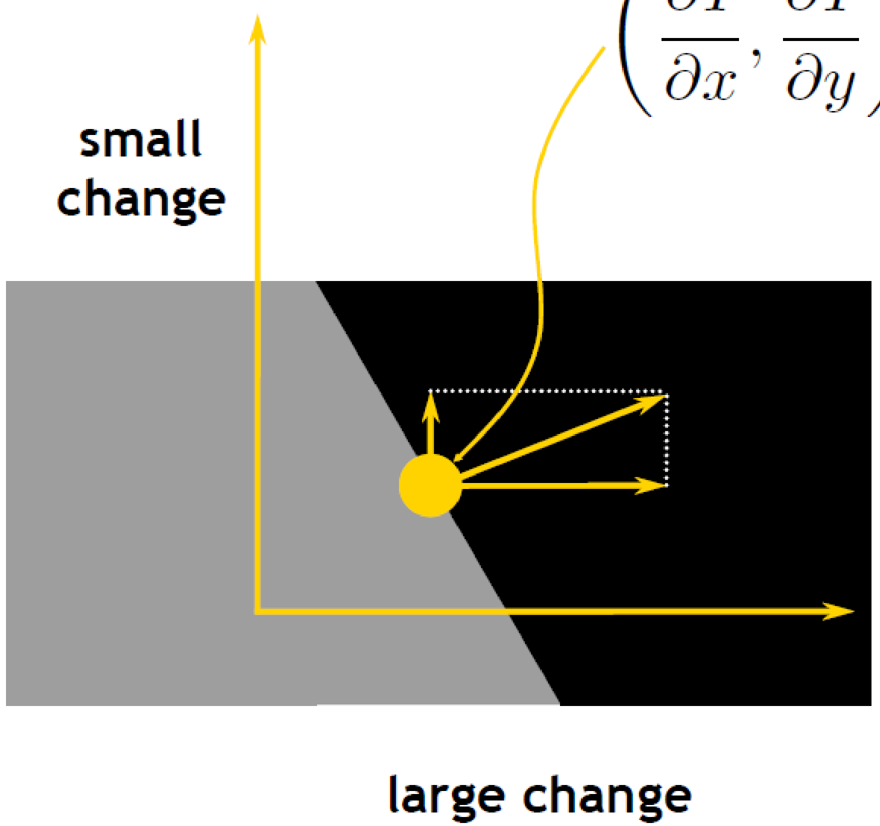
What is a gradient



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

What is a gradient

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

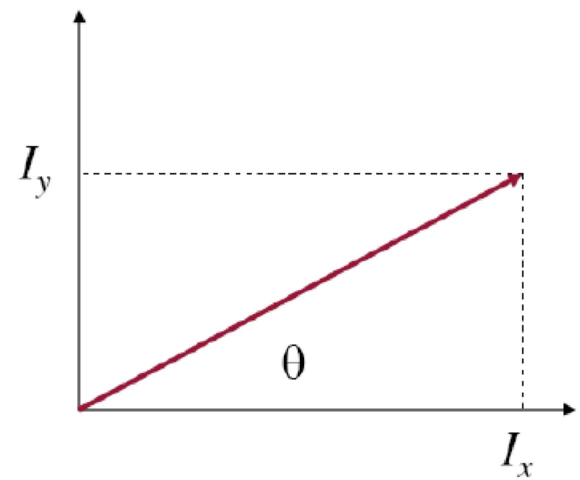


- gradient direction is perpendicular to edge
- gradient magnitude measures edge strength

What is a Gradient

- the gradient is:

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



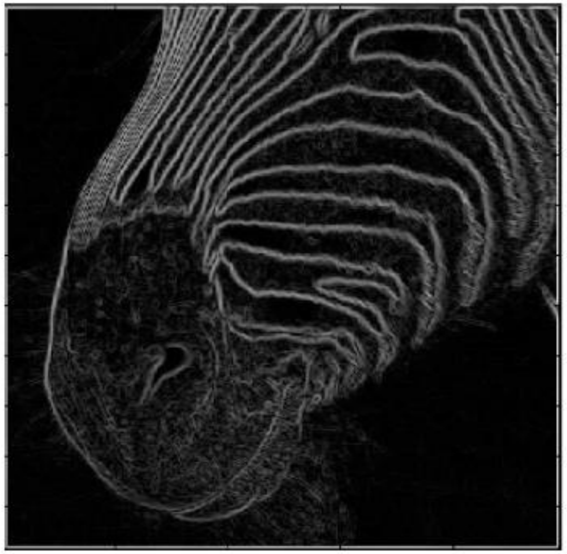
- the magnitude of the gradient is:

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

- the direction of the gradient is:

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

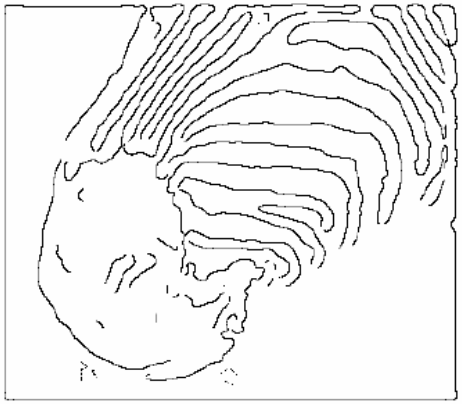
Example – Gradient magnitude image



First smoothed with Gaussian

First smoothed with broad Gaussian

In Image Processing Lecture we look at how to get edge chains?



Half-way Slide

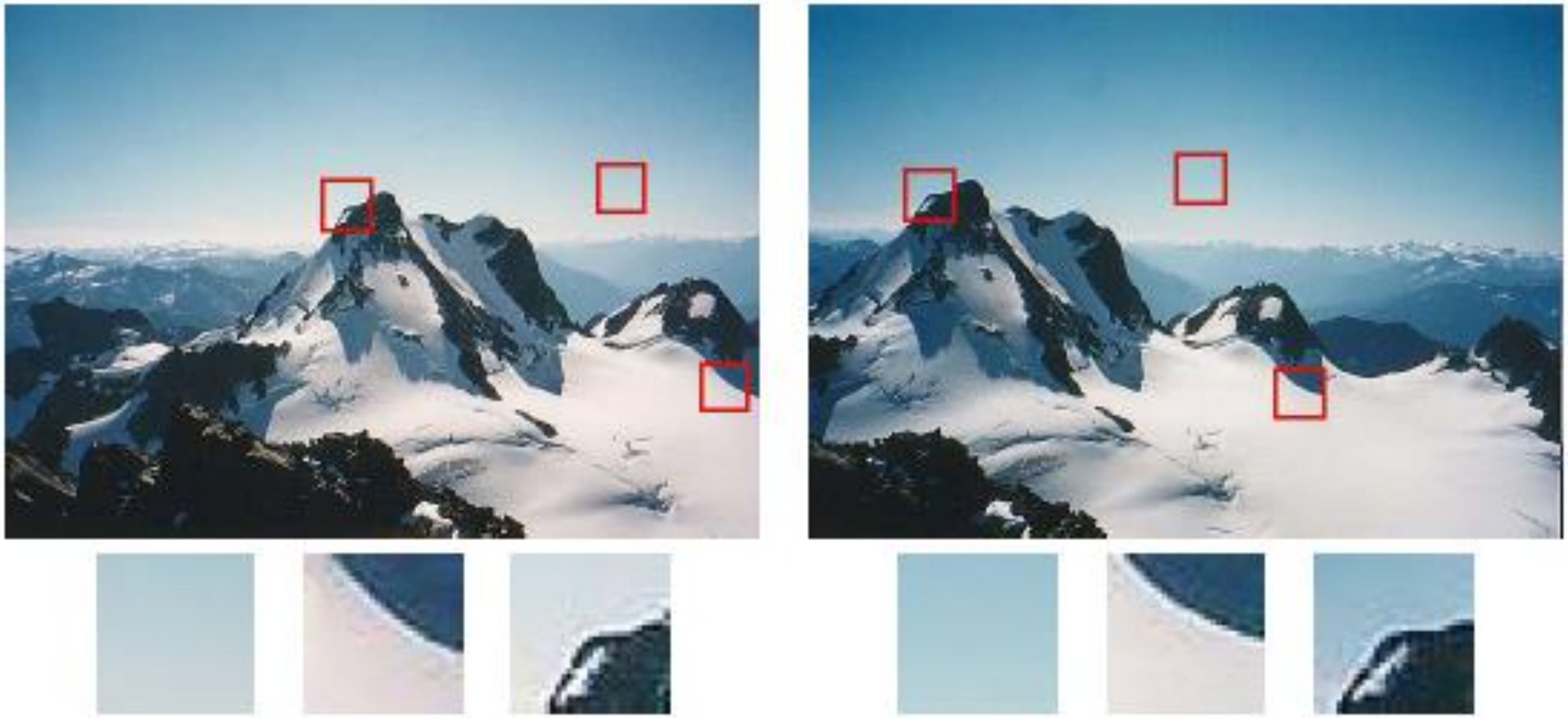
3 minutes break

Roadmap: Basics of Digital Image Processing

- What is an Image?
- Point operators (ch. 3.1)
- Filtering: (ch. 3.2, ch 3.3, ch. 3.4)
  - Linear filtering
  - Non-linear filtering
- Edges detection (ch. 4.2)
- Interest Point detection (ch. 4.1.1)

What region should we try to match?

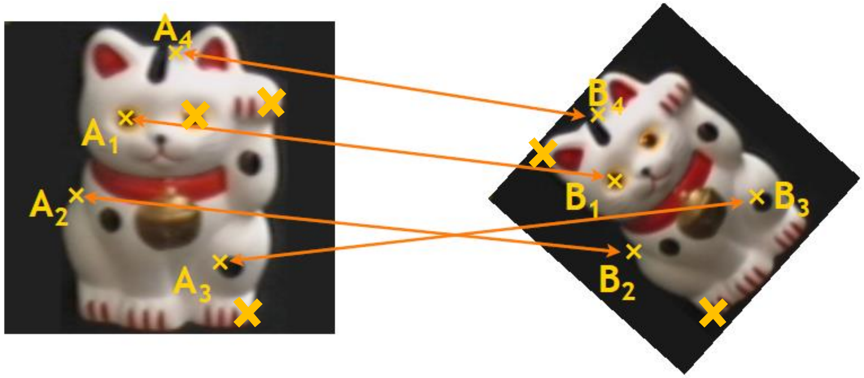
We want to find a few regions where this image pair matches (applications later)



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



Should be invariant against:

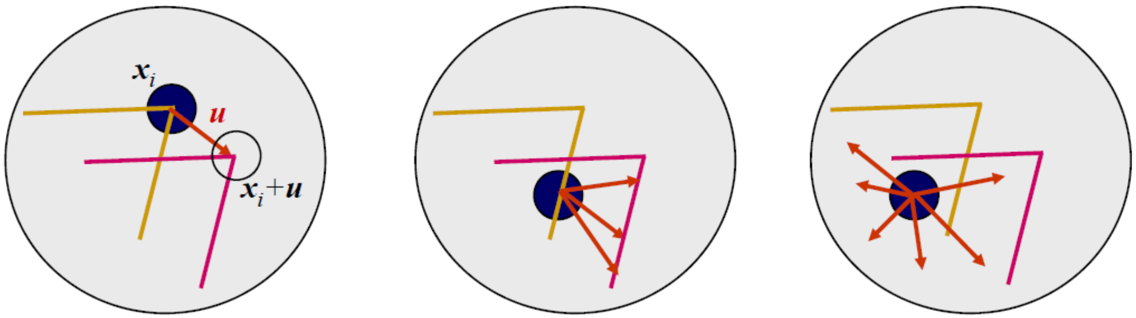
a. Geometric transformation – scaling, rotation, translation, affine transformation, projective transformation etc.

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

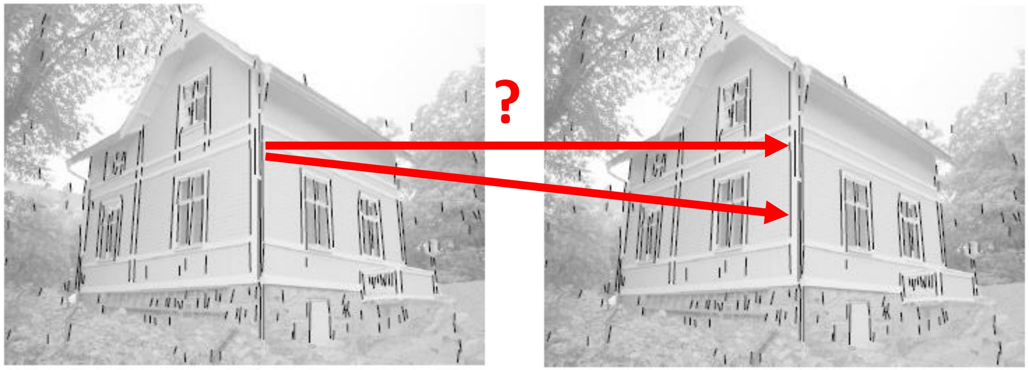
c. Discretization (e.g. spatial resolution, focus);

Points versus Lines

„Apeture problem“

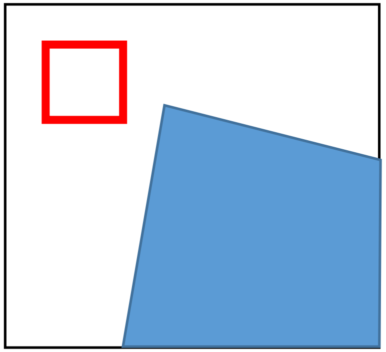


Lines are not as good as points

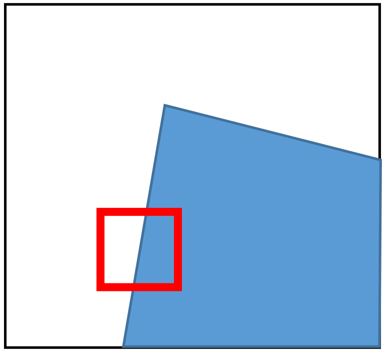


Harris Corner Detector – Basic Idea

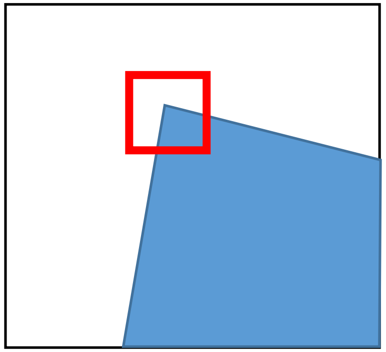
Local measure of feature uniqueness:
Shifting the window in any direction: How does it change?



"flat" region:
no change in all directions



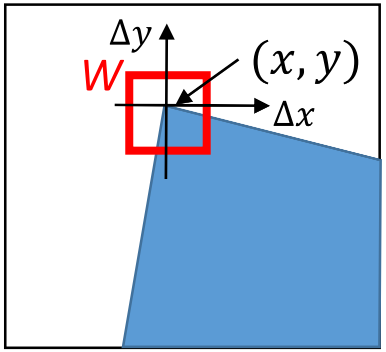
"edge":
no change along the edge direction



"corner":
significant change in all directions

[Szeliski and Seitz]

How similar is a local window with its neighbor?



Auto-correlation 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 window around (x, y)

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

Harris Corner Detector

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

One is interested in **properties** of $c(x, y, \Delta x, \Delta y)$ at each position (x, y)
We could evaluate $c(x, y, \Delta x, \Delta y)$ for all discrete shifts $\Delta x, \Delta y$ = +/-1.
But we would like to do smaller shifts and have a fast method.

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

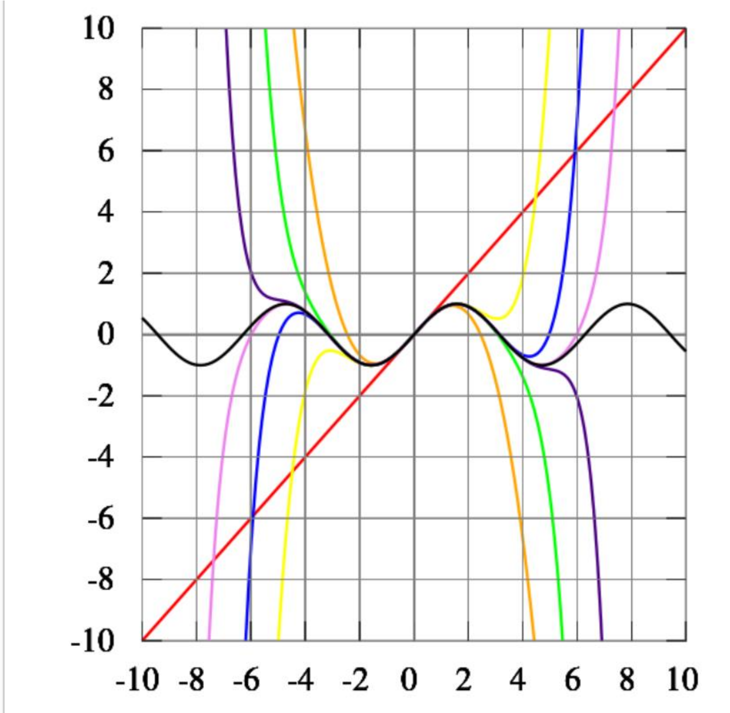
$$\begin{aligned} I(u+\Delta x, v+\Delta y) &= I(u, v) + \frac{\partial I(u, v)}{\partial x} \Delta x + \frac{\partial I(u, v)}{\partial y} \Delta y + \epsilon(\Delta x, \Delta y) \\ &\approx I(u, v) + \underbrace{[I_x(u, v), I_y(u, v)]}_{\text{Gradient at } (u,v)} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \end{aligned}$$

Reminder: Taylor Expansion

A function $f(x)$ is approximated by:

$$f(x) \approx \sum_n \frac{f^{(n)}(a)}{n!} (x-a)^n$$

The approximation is most accurate at point a



As the degree of the Taylor polynomial rises, it approaches the correct function. This image shows sin(*x*) and its Taylor approximations, polynomials of degree 1, 3, 5, 7, 9, 11 and 13.

Harris Corner Detector

Put it together:

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

Q is call the 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)

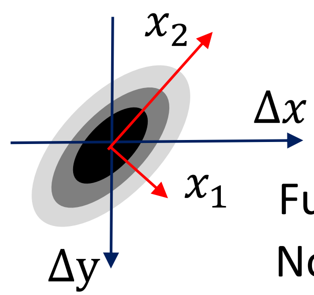
Harris Corner Detector

The auto-correlation 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

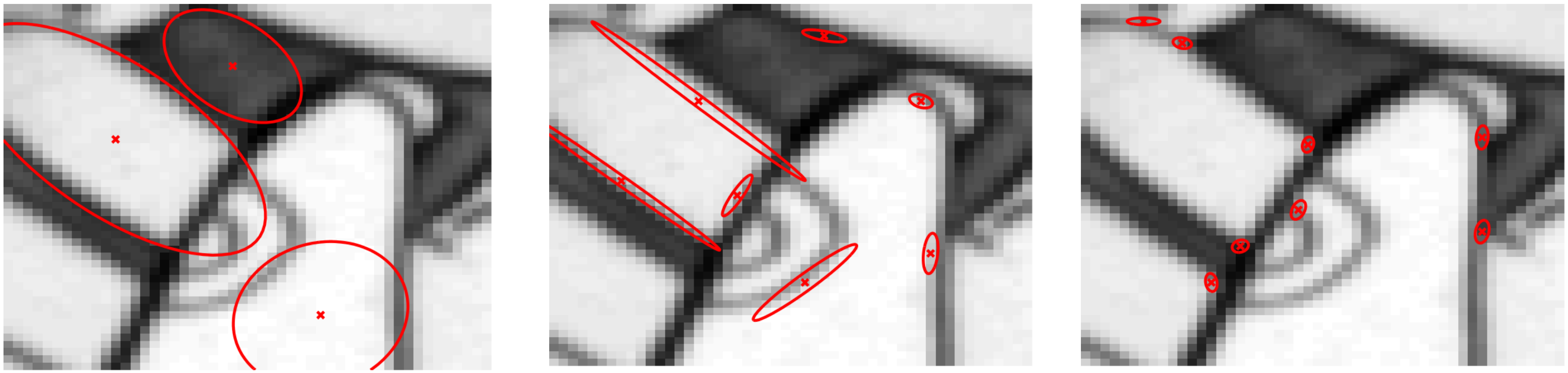


Function *c*

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

Harris Corner Detector

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



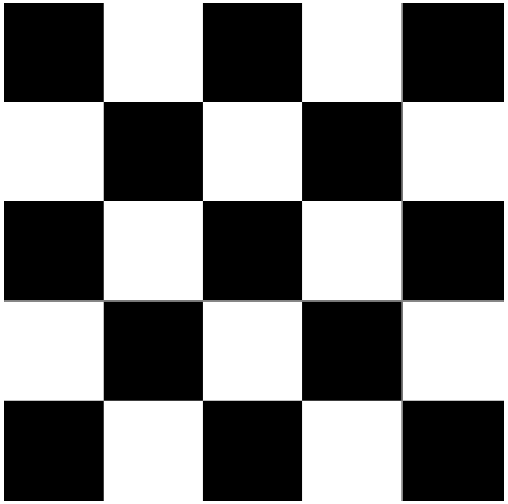
(a) Flat (b) Edges (c) Corners

a. Homogenous regions: both λ-s are small

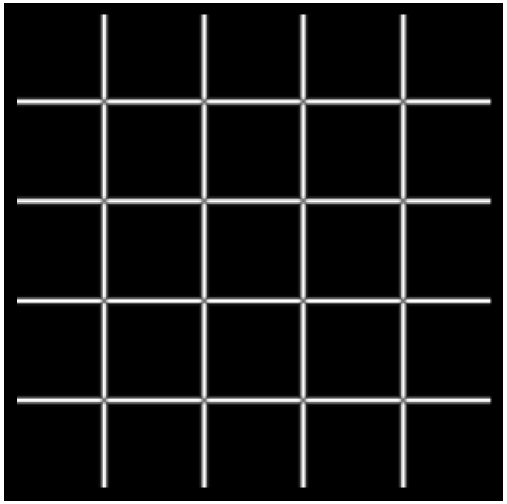
b. Edges: one λ is small the other one is large

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

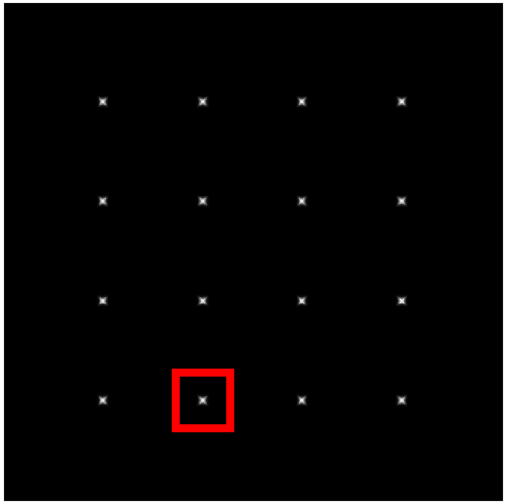
Harris Corner Detector



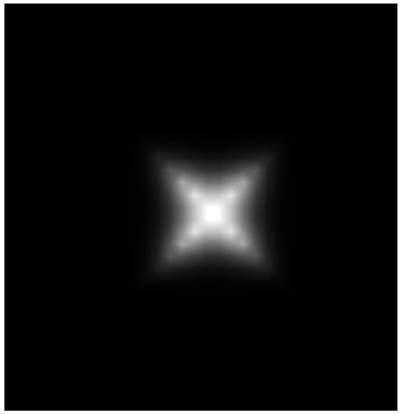
Image



λ_1 (larger eigenvalue)



λ_2 (smaller eigenvalue)



Zoom in

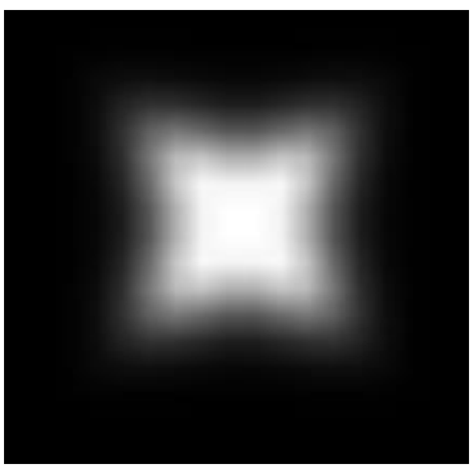
Harris Corner Detector

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

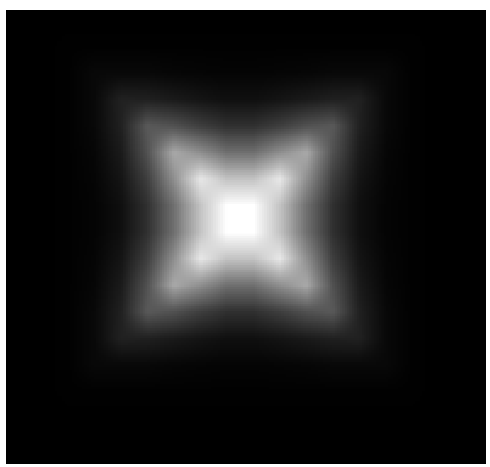
$$\lambda_1 \lambda_2 = \det Q(x,y) = AC - B^2, \quad \lambda_1 + \lambda_2 = \operatorname{trace} Q(x,y) = A + C$$

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

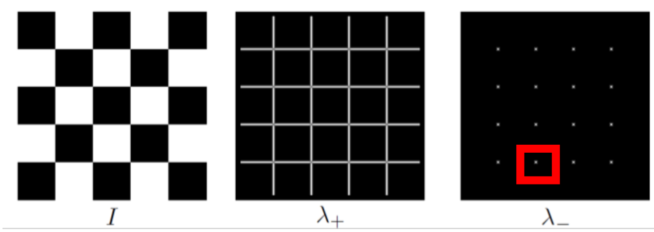
Downweights edges where $\lambda_1 \gg \lambda_2$



Harris Value

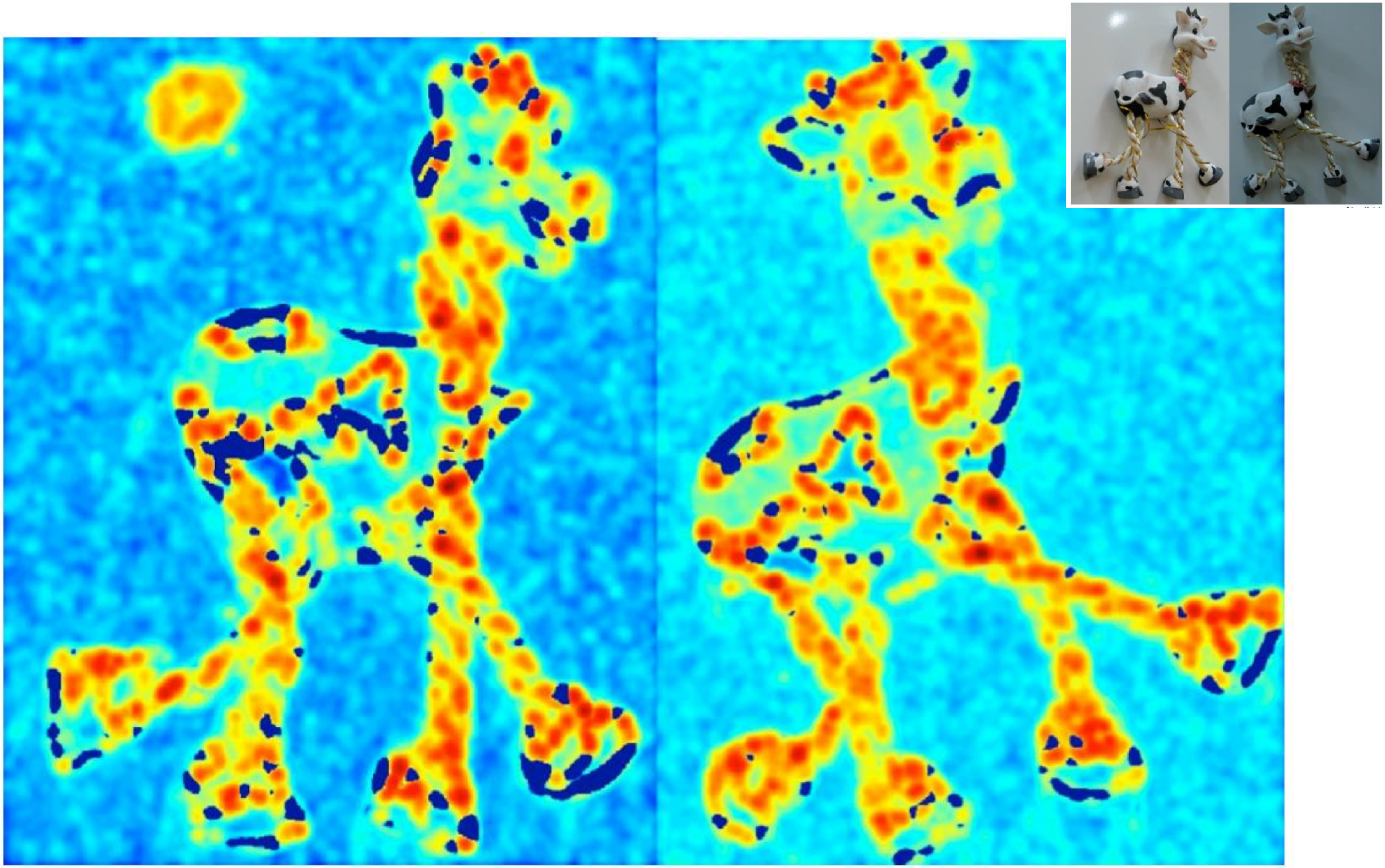


Smallest eigenvalue

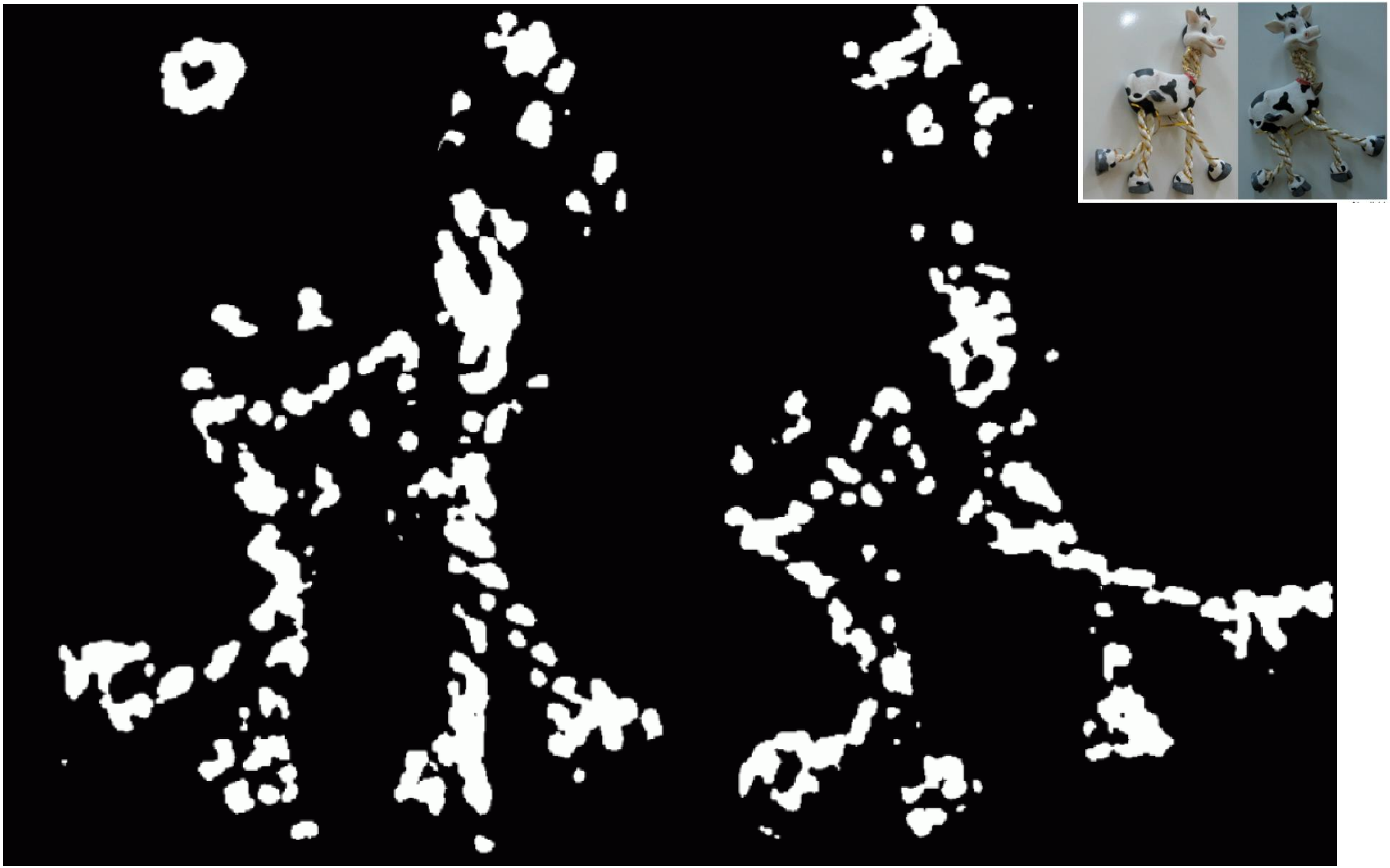


Harris Corners - Example

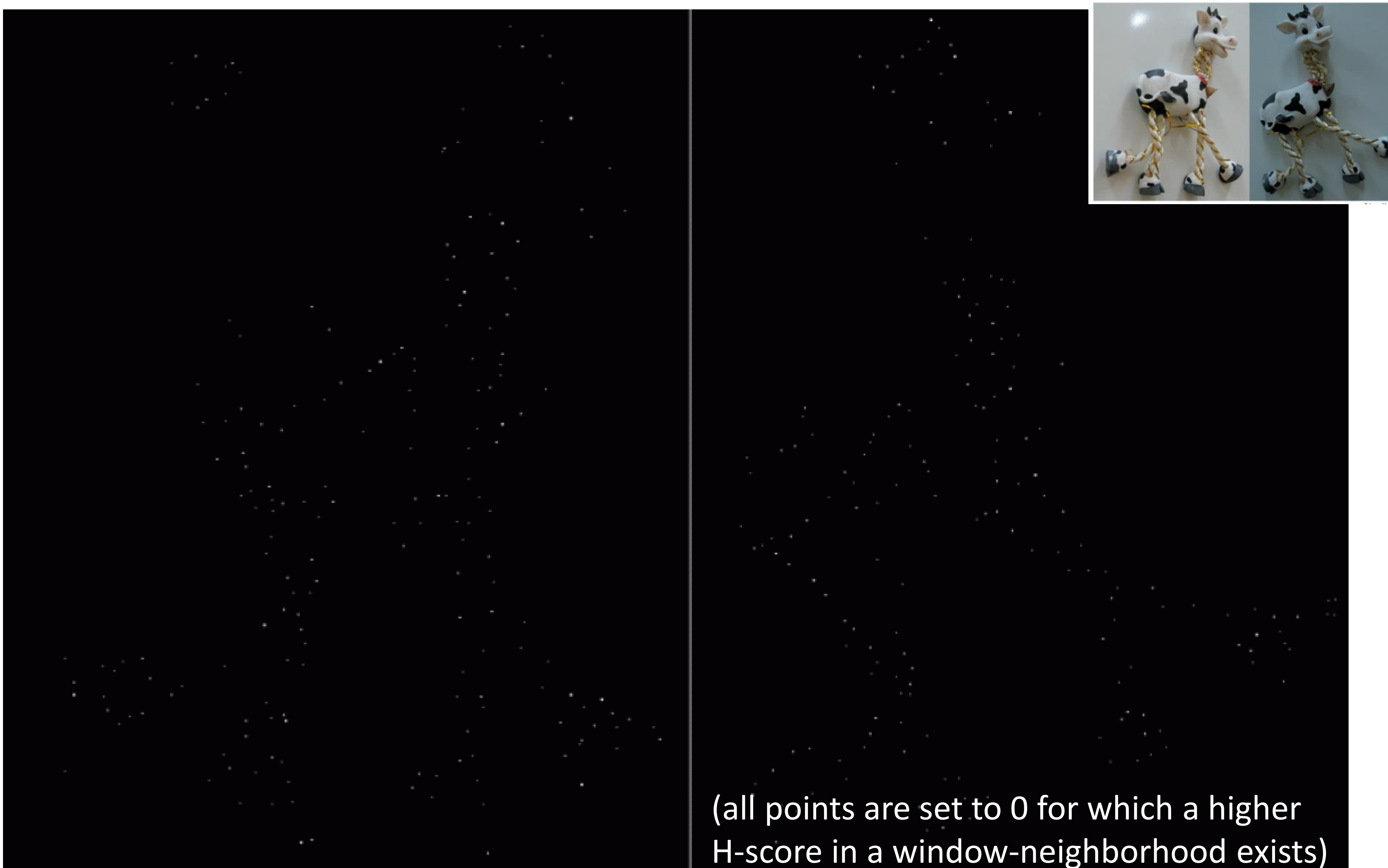
H-score (red- high, blue - low)



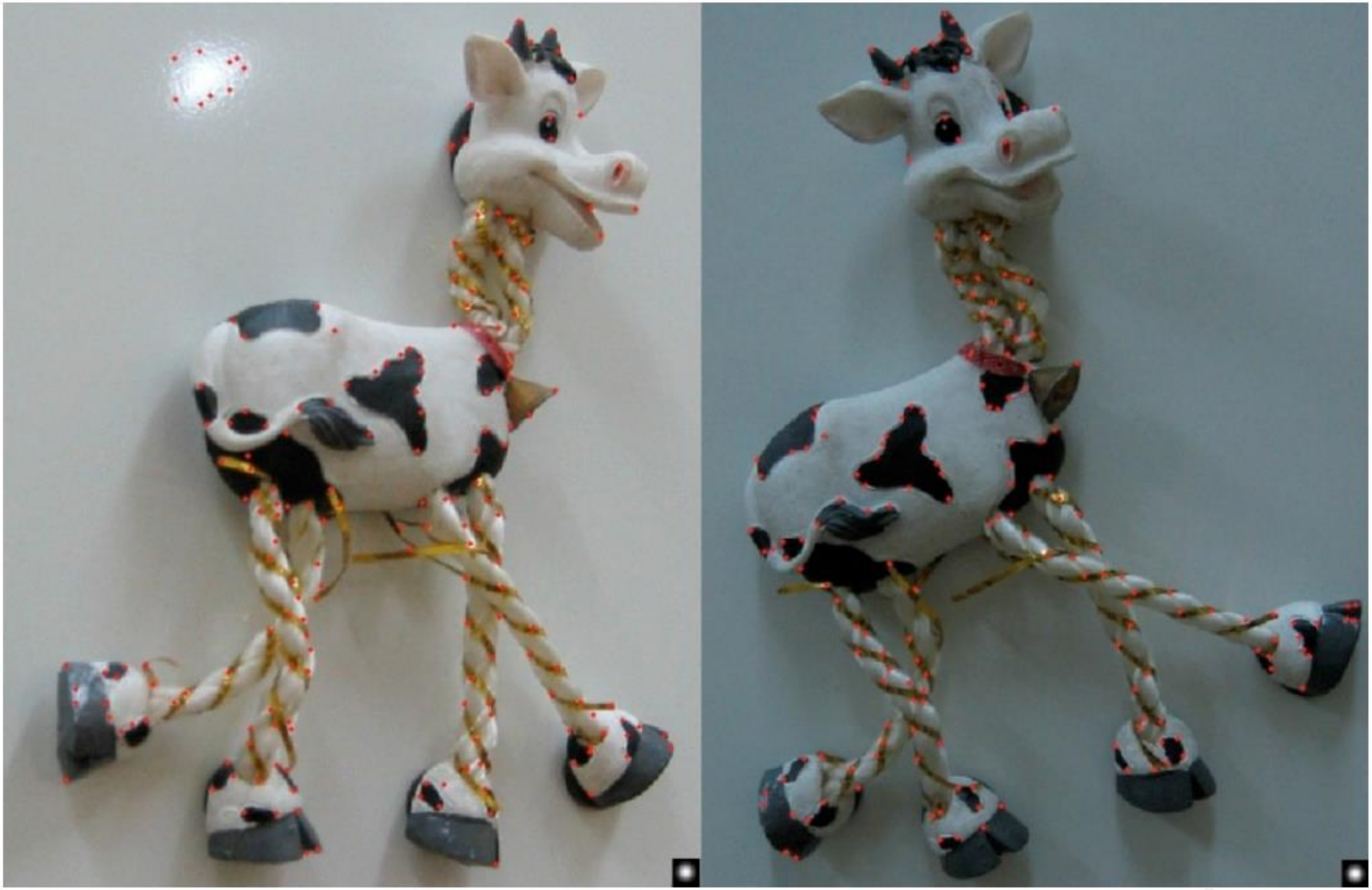
Threshold (H-score > value)



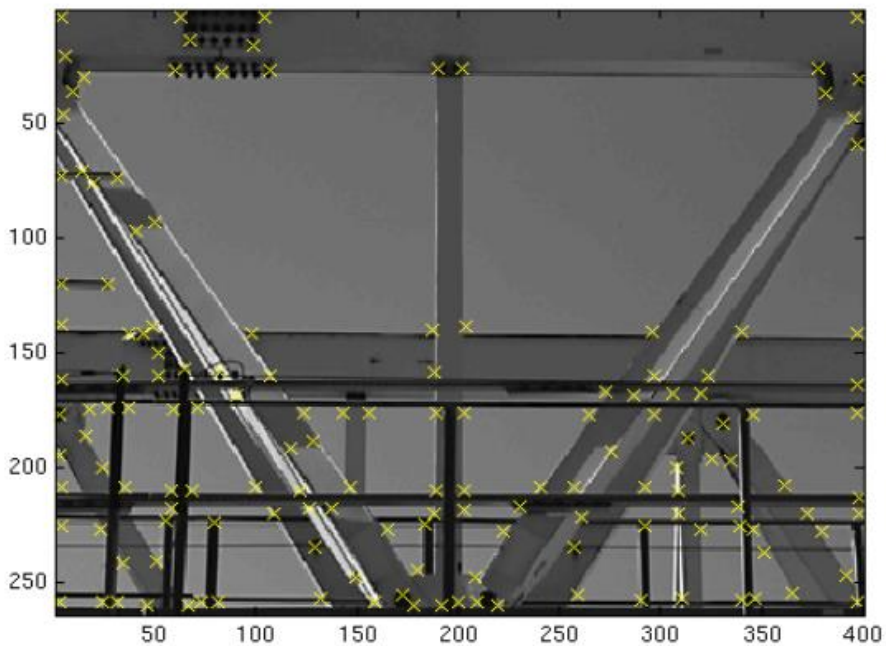
Non-maximum suppression

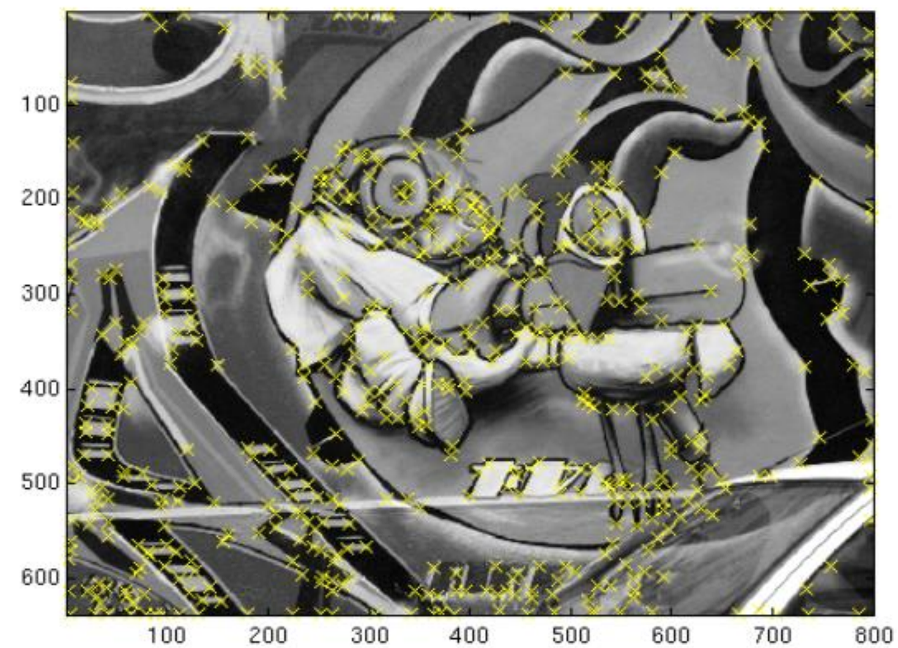


Harris Corners in Red

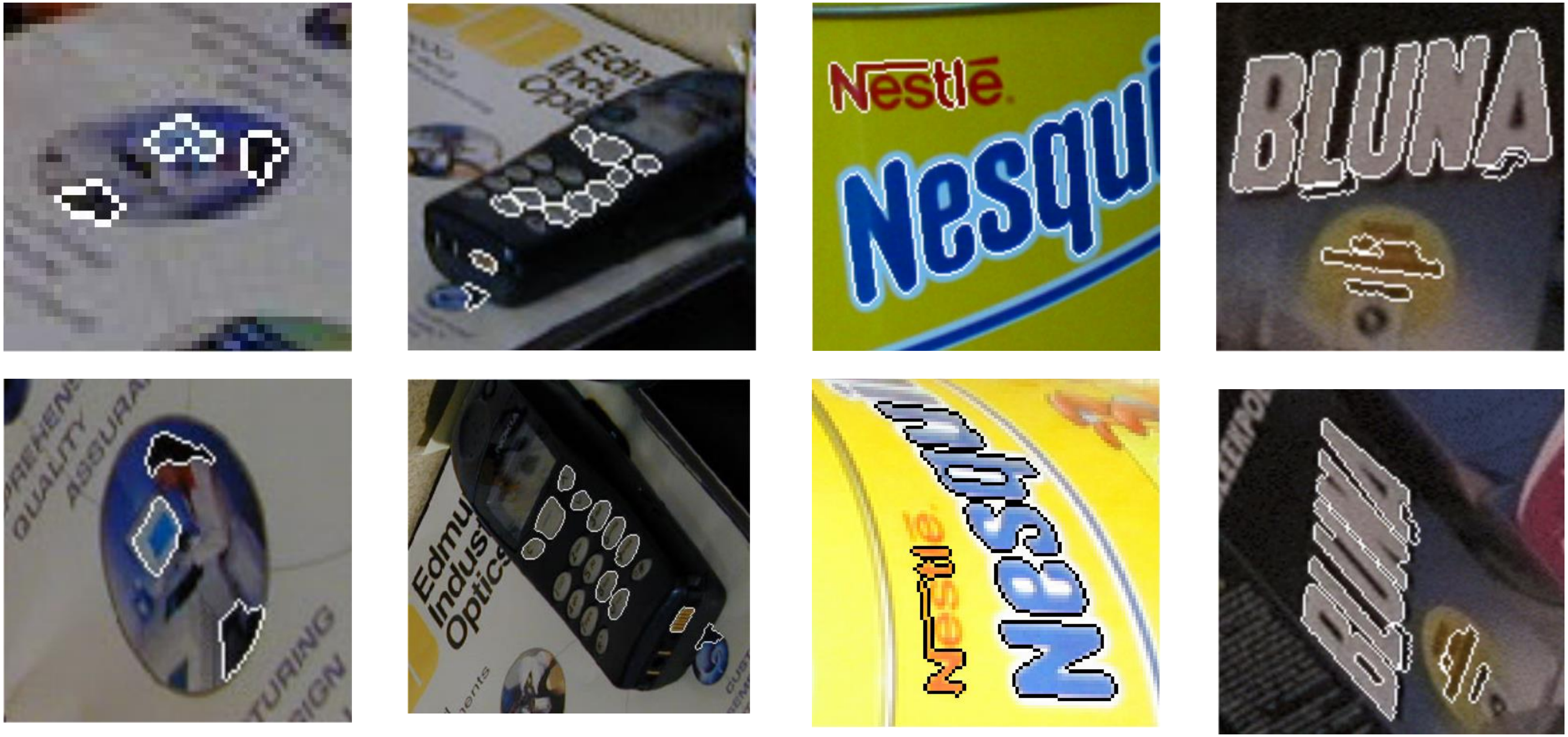


Other examples





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)