Feature Extraction

Computer Vision

Hubel DH (1988) Eye, Brain and Vision.
Olshausen & Field, 1997

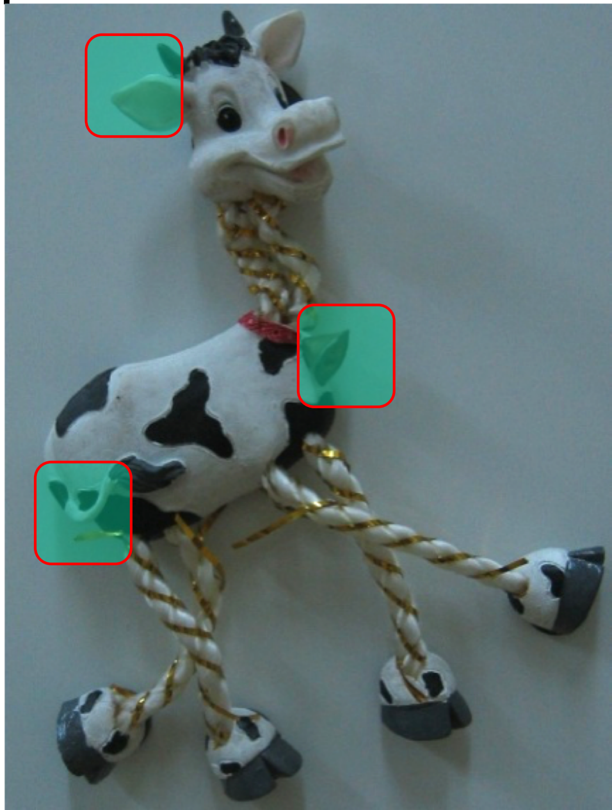
Computer Vision

Features and matching

Feature: description of a (part of) a pattern / object in the image, e.g. shape, texture, emitted heat if infrared…

Mathematically: Describe the pattern with a vector of values

$$\mathbf{f} = [f_1, \ldots, f_N]$$



Goal : efficient matching for

registration

correspondences for 3D,

tracking,

recognition,

…

Features and matching

Feature: description of a (part of) a pattern / object in the image, e.g. shape, texture, emitted heat if infrared…

Mathematically: Describe the pattern with a vector of values

$$\mathbf{f} = [f_1, \ldots, f_N]$$

Goal : efficient matching for

registration

correspondences for 3D,

tracking,

recognition,

…

Computer Vision

Examples for recognition

Examples for recognition

Computer Vision

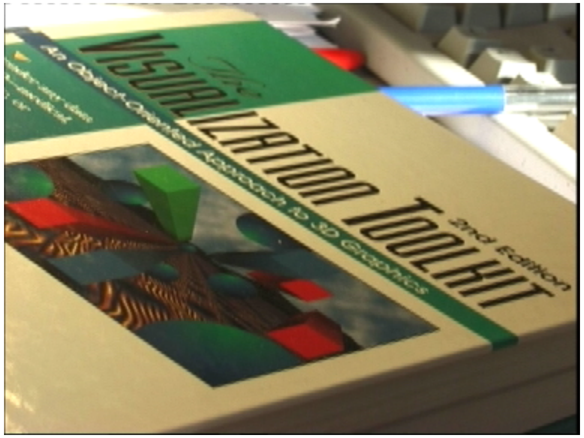
Matching is a challenging task

Re-iterating the difficulties with matching highlighted thus far:

- features to deal with **large variations** in
  - Viewpoint

Matching: a challenging task

- features to deal with **large variations** in
  - Viewpoint
  - Illumination



Matching: a challenging task

- features to deal with **large variations** in
  - Viewpoint
  - Illumination
  - Background

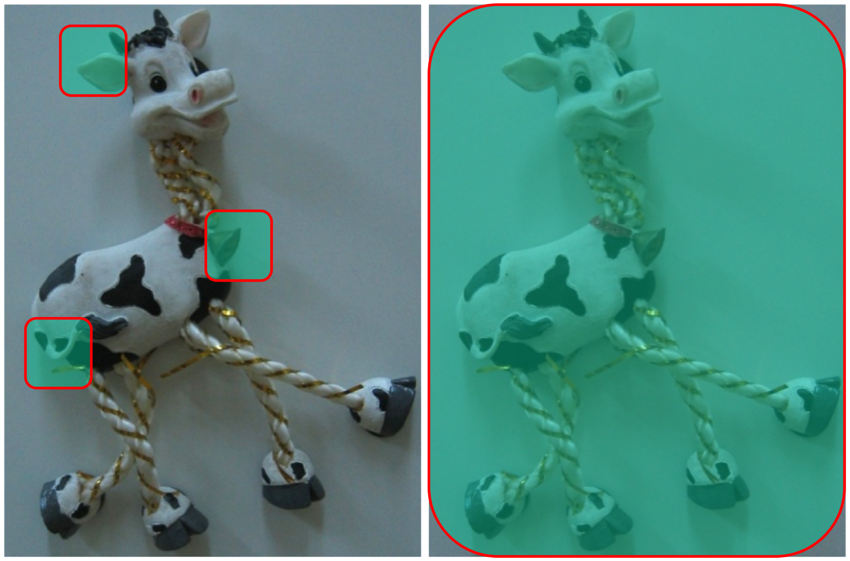
Matching: a challenging task

- features to deal with **large variations** in
  - Viewpoint
  - Illumination
  - Background
  - Occlusion

Computer Vision

Considerations when selecting features

- 1. Complete (describing pattern unambiguously) or not
- 2. Robustness of extraction
- 3. Ease / speed of extraction
- 4. Global vs. local



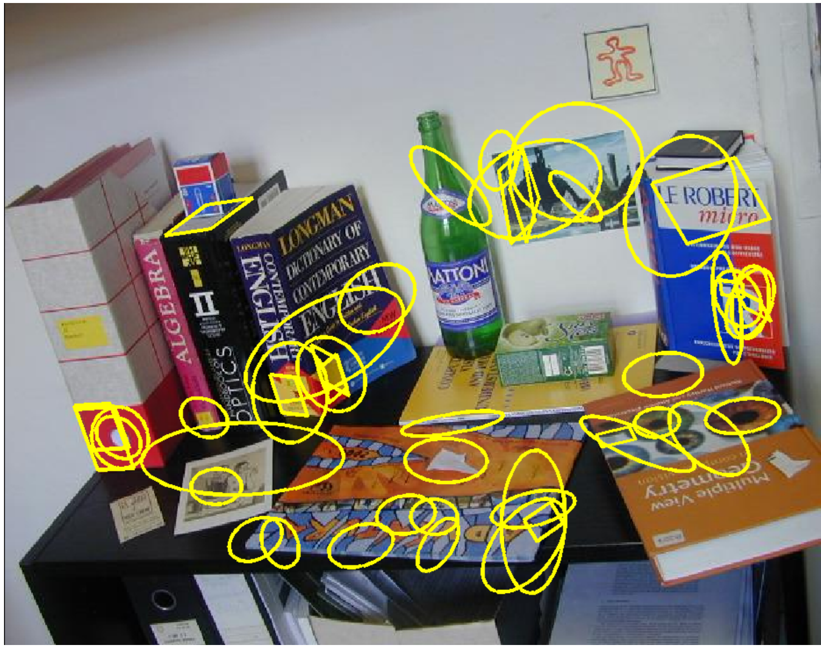
THUS:

Computer Vision

Strategy

1. Identifying points of interest
2. Extract local features, vectors, around those interest points

- Many features per image
- Representing different parts of objects



PART 1: Identifying “basic” points of interest

Interest points for localizable patches

A feature should capture something *discriminative* about a well *localizable* patch of a pattern

We start with the well localizable bit:

Shifting the patch a bit should make a big difference in terms of the underlying pattern

I should be able to get the same point of interest under pose / lighting variations

Outline

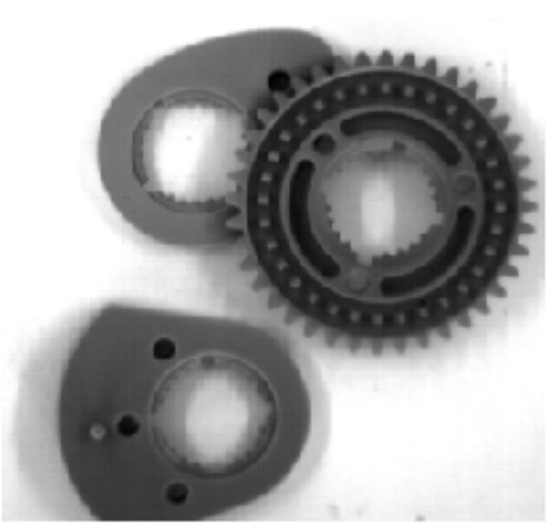
Identifying points of interest

1. Edge detection
   a. Gradient operators
   b. Zero-crossings of Laplacians
   c. Canny Edge Detector
2. Corner detection

Computer Vision

Hubel DH (1988) Eye, Brain and Vision.
Olshausen & Field, 1997

Edge Detection



edges arise from changes in :

- 1. reflectance
- 2. orientation
- 3. Illumination (e.g. shadows)

Thus, edges are not necessarily relevant to e.g. shape

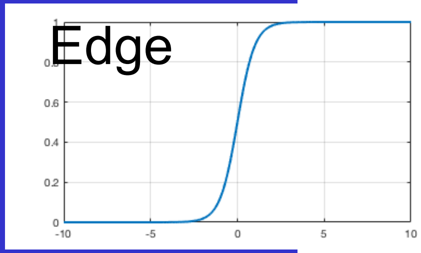
Methods introduced here are only 1st step, edge linking is the hard part

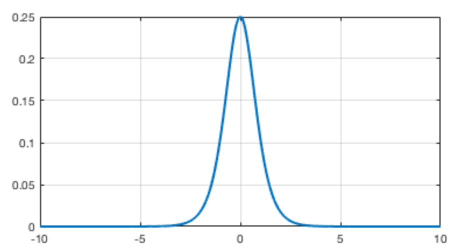
Edge detection methods

we investigate three approaches :

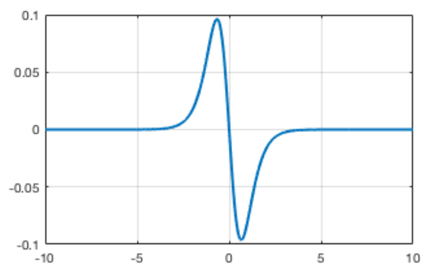
- ❑ 1. locating high intensity gradient magnitudes
- ❑ 2. locating inflection points in the intensity profile
- ❑ 3. signal processing view (optimal detectors)
  Canny edge detector

we will only consider isotropic operators

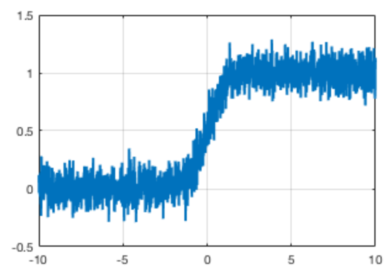




Gradient magnitude



Inflection points



Optimal detector

Outline

Identifying points of interest

1. Edge detection
   a. **Gradient operators**
   b. Zero-crossings of Laplacians
   c. Canny Edge Detector
2. Corner detection

Gradient operators : principle

image $f(x,y)$: locate edges at f's steep slopes

measure the gradient magnitude

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

easy to check that this operator is isotropic

the direction of steepest change : rotate coordinate frame and find θ that maximizes

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

differentiation w.r.t. θ yields

$$\theta_{xtr} = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

the corresponding magnitude is the one defined above

Gradient operators : implementation

Gradient magnitude is a non-linear operator

$\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ are linear and shift-invariant

they can thus be implemented as a convolution

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

-1	1

$\frac{\partial}{\partial y}$

-1
1

Prone to noise!
We want something that will smooth and compute gradients

Gradient operators : Sobel

discrete approximation (finite differences) :

$$\frac{\partial}{\partial x} \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{\partial}{\partial y} \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

these are the *Sobel masks*

Gradient operators : Sobel

one mask primarily for vertical and one for horizontal edges

combine their outputs :

- 1. take the square root of the sum of their squares
- 2. take arctan of their proportion to obtain edge orientation

these masks are separable, e.g.

$$(-1,0,1) \otimes (1,2,1)^T$$

easy to implement in hardware

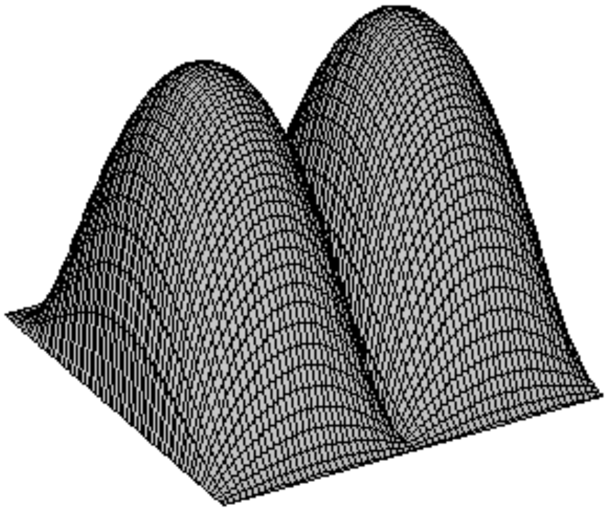
Gradient operators : MTF shows smoothing effect of Sobel mask

example : MTF of the vertical Sobel mask

$$(2i \sin 2\pi u)(2\cos 2\pi v + 2)$$

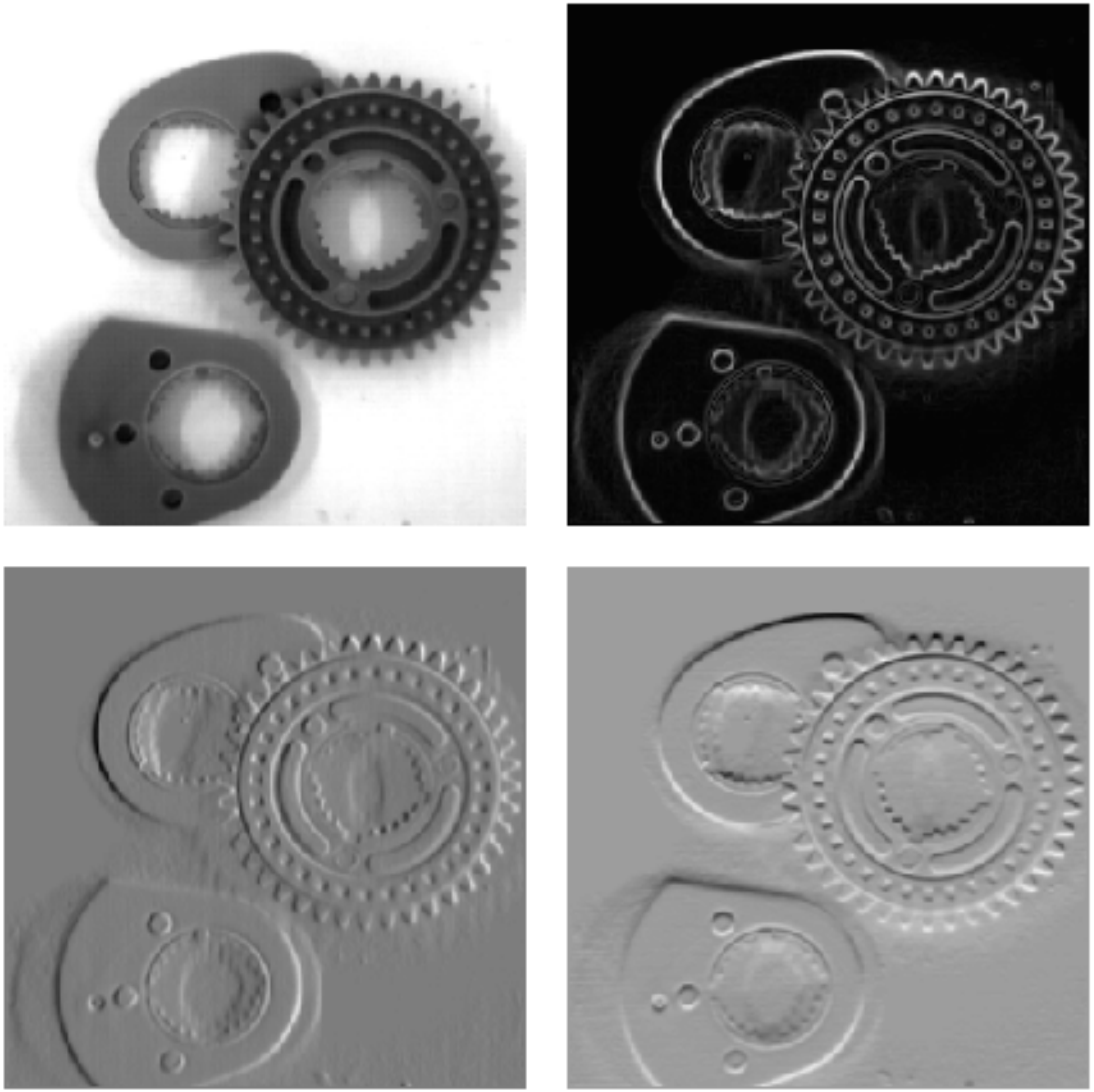
this is a pure imaginary function, resulting in $\pi/2$ phase shifts

power spectrum :

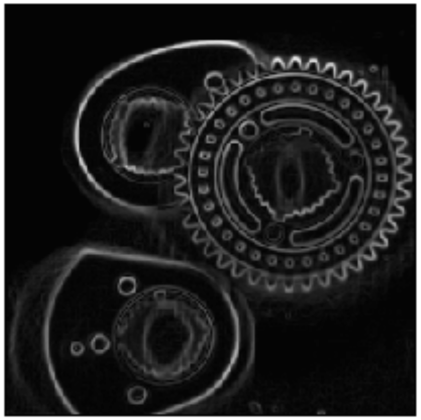


u-dir. : band-pass, v-dir. : low-pass

Gradient operators : example



Gradient operators : analysis



result far from a perfect line drawing :

1. gaps
2. several pixels thick at places
3. some edges very weak , whereas others are salient

Sobel masks are the optimal 3 x 3 convolution filters with integer coefficients for step edge detection

Outline

Identifying points of interest

1. Edge detection
   a. Gradient operators
   b. Zero-crossings of Laplacians
   c. Canny Edge Detector
2. Corner detection

Computer Vision

Zero-crossings : principle

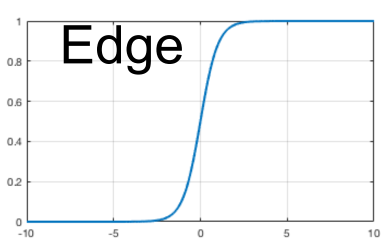
consider edges to lie at intensity inflections

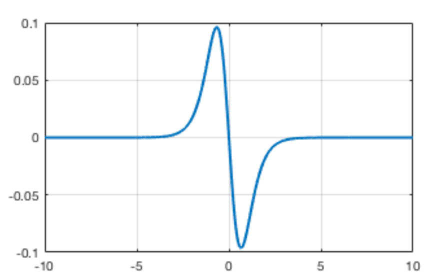
can be found at the zero-crossings of the Laplacian :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

= linear + shift-invariant $\Rightarrow$ convolution

= also isotropic





Inflection points

Computer Vision

Discrete approximations of the Laplacian

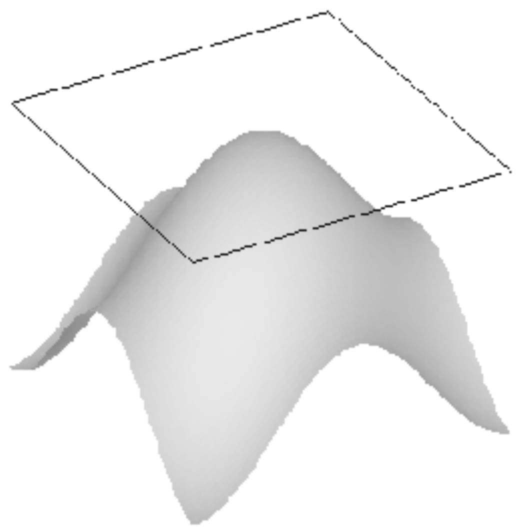
0	1	0
1	- 4	1
0	1	0

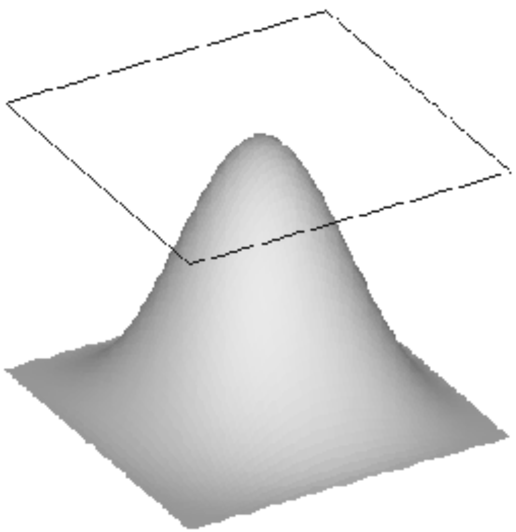
1	2	1
2	-12	2
1	2	1

MTF of left filter :

$$2\cos(2\pi u) + 2\cos(2\pi v) - 4$$

MTF's





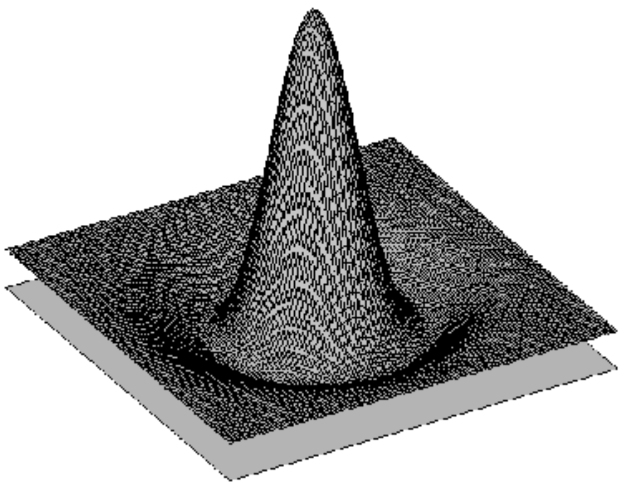
Zero-crossings : implementation

sensitive to noise (2nd order der.)

therefore combined with smoothing, e.g. a Gaussian :

$$L * (G * f) = (L * G) * f$$

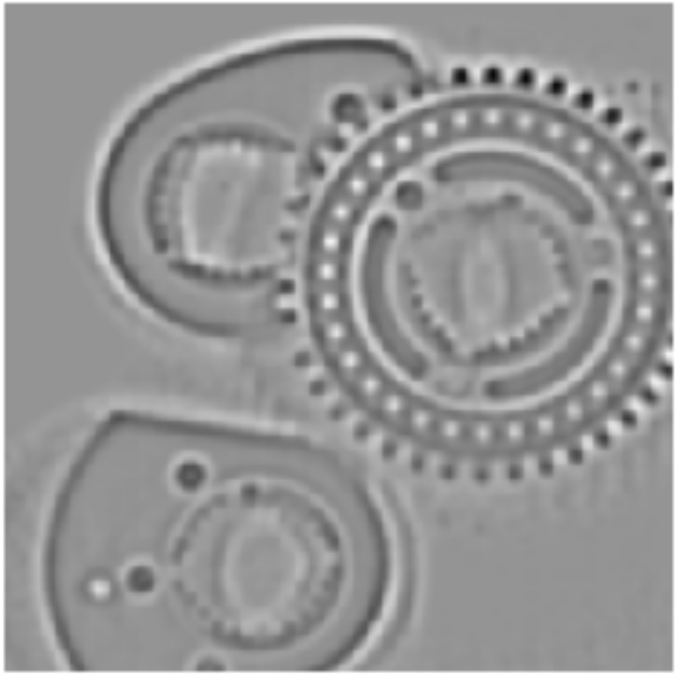
yields *"Mexican hat"* filter :

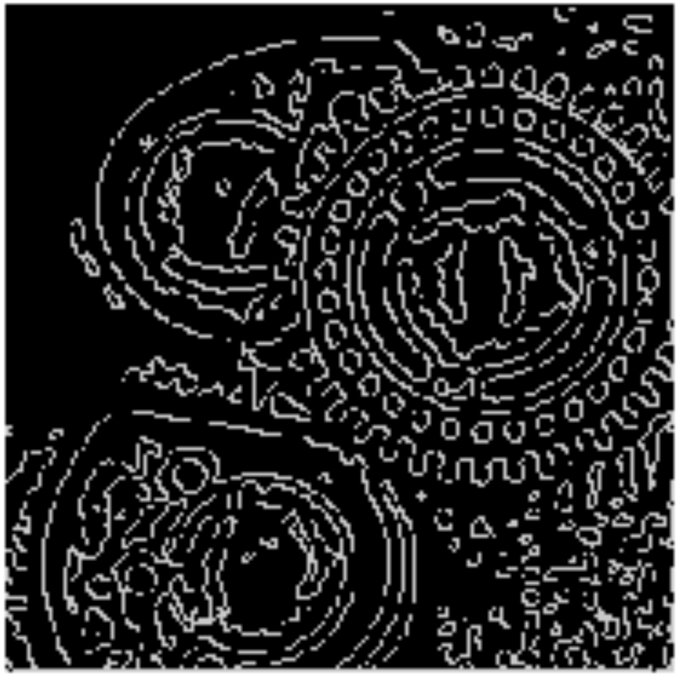


also implemented as DOG (difference of Gaussians)

Computer Vision

Zero-crossings : example





one-pixel thick edges
closed contours
yet not convincing

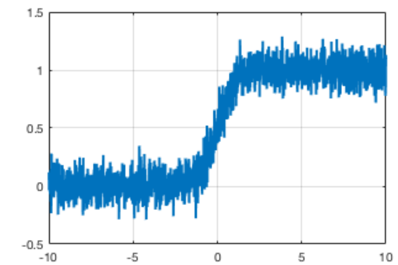
Outline

Identifying points of interest

1. Edge detection
    a. Gradient operators
    b. Zero-crossings of Laplacians
    c. Canny Edge Detector
2. Corner detection

Computer Vision

The Canny edge detector



Optimal detector

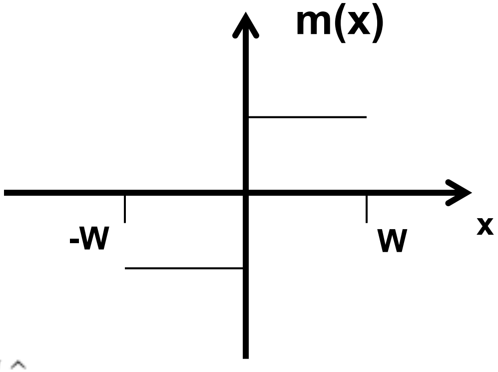
A (1D) signal processing approach

Looking for "optimal" filters

Optimality criteria

- Good SNR
  strong response to edges
  low (no) response to noise
- Good localization
  edges should be detected on the right position
- Uniqueness
  edges should be detected only once

Characterization of SNR



- Response of system ***h*** to signal
  deterministic signal model $m(x)$
  step edge at the origin

$$\boldsymbol{h}[m(x)] = \int_{-W}^{W} h(x-\hat{x})m(\hat{x})d\hat{x}$$

  at the edge position

$$\boldsymbol{h}[m](0) = \int_{-W}^{W} h(-\hat{x})m(\hat{x})d\hat{x}$$

- Response to noise $n(x)$
  noise is stochastic (Gaussian, white, uncorrelated)
  can be characterized by expected value

$$\left.\sqrt{E\left[\left(\int_{-W}^{W} h(\hat{x})n(x-\hat{x})d\hat{x}\right)^2\right]}\,\right|_{x=0} = \sigma\sqrt{\int_{-W}^{W} h^2(\hat{x})d\hat{x}}$$

Computer Vision

Characterization of SNR

$$SNR = \frac{\int_{-W}^{W} h(-\hat{x})m(\hat{x})d\hat{x}}{\sigma\sqrt{\int_{-W}^{W} h^2(\hat{x})d\hat{x}}}$$

Computer Vision

Characterization of localization

- Edge location: maximum of the system response
  extremum of ***h***$(m(x)+n(x))$ at x_0
  again stochastic (depending on the noise)
  will deviate from the ideal edge position at 0
- Quantification through expected value of the deviation from the real edge location $\sqrt{E\left[x_0^2\right]}$
- Localization measure

$$LOK = \frac{1}{\sqrt{E\left[x_0^2\right]}} = \frac{\left|\int_{-W}^{W} h'(\hat{x})m'(-\hat{x})d\hat{x}\right|}{\sigma\sqrt{\int_{-W}^{W}\left[h'(\hat{x})\right]^2 d\hat{x}}}$$

The matched filter

- Optimal filter ***h****(x)* for which

$$\max\left(\mathrm{SNR}\times\mathrm{LOC}\right)$$

- can be shown that

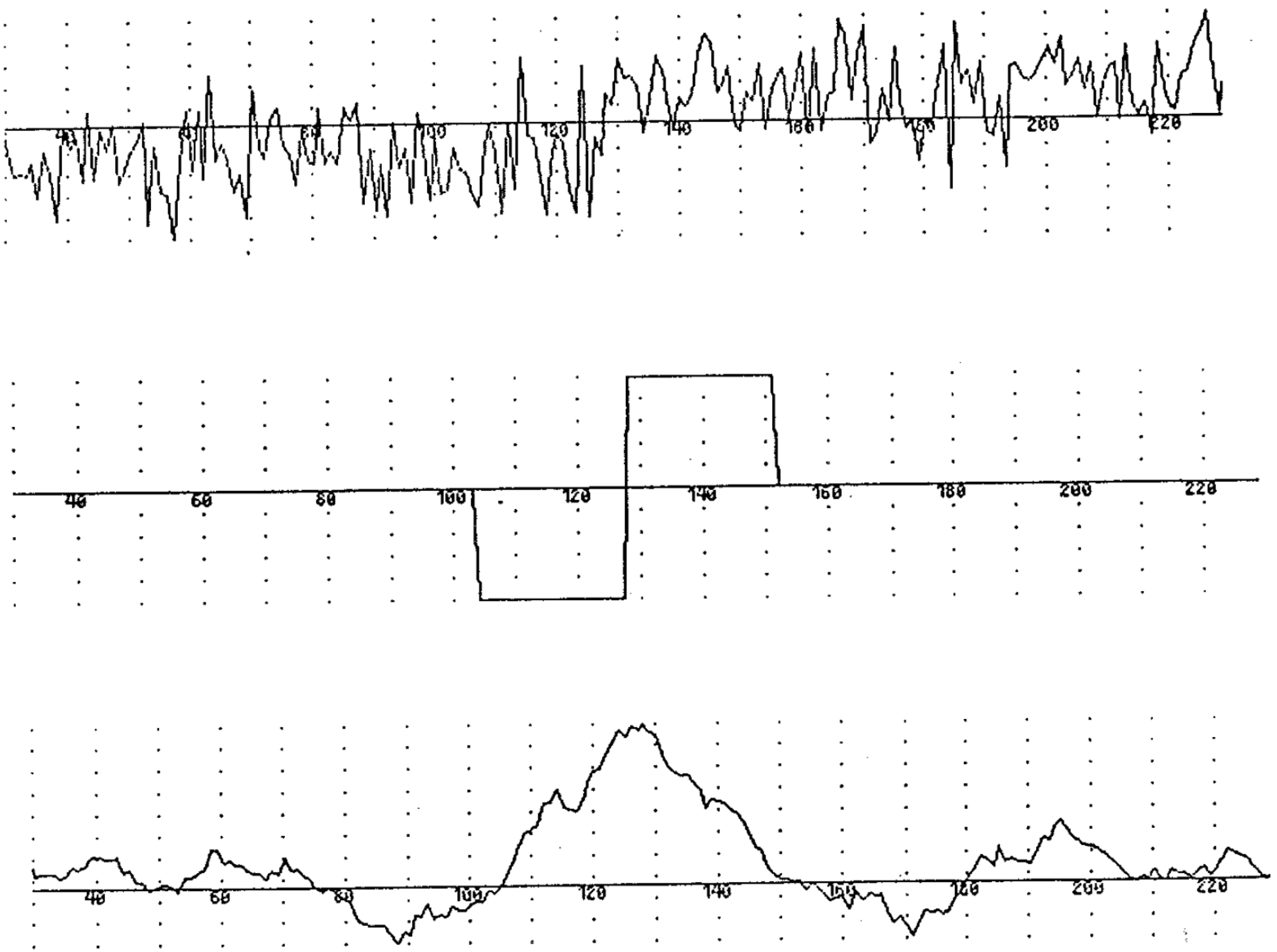
$$h(x) = \lambda m(x) \quad x \in [-W, W]$$

Essentially identical with the signal to be detected

- For the edge model used: difference of boxes DOB Filter

Computer Vision

The matched filter



Uniqueness

- Filtering with DOB generates many local maxima due to noise
- Hinders unique detection
- Remedy: minimize the number of maxima within the filter support
- Caused by noise (stochastic)
  Characterized by the average distance between subsequent zero crossings of the noise response derivative (*f* = ***h'***(*n*))
  Rice theorem

$$x_{ave} = \pi \sqrt{\frac{-\Phi_{ff}(0)}{\Phi''_{ff}(0)}}$$

Computer Vision

1D optimal filter

- Average number of maxima within filter support

$$N_{max} = \frac{2W}{x_{max}} = \frac{W}{x_{ave}}$$

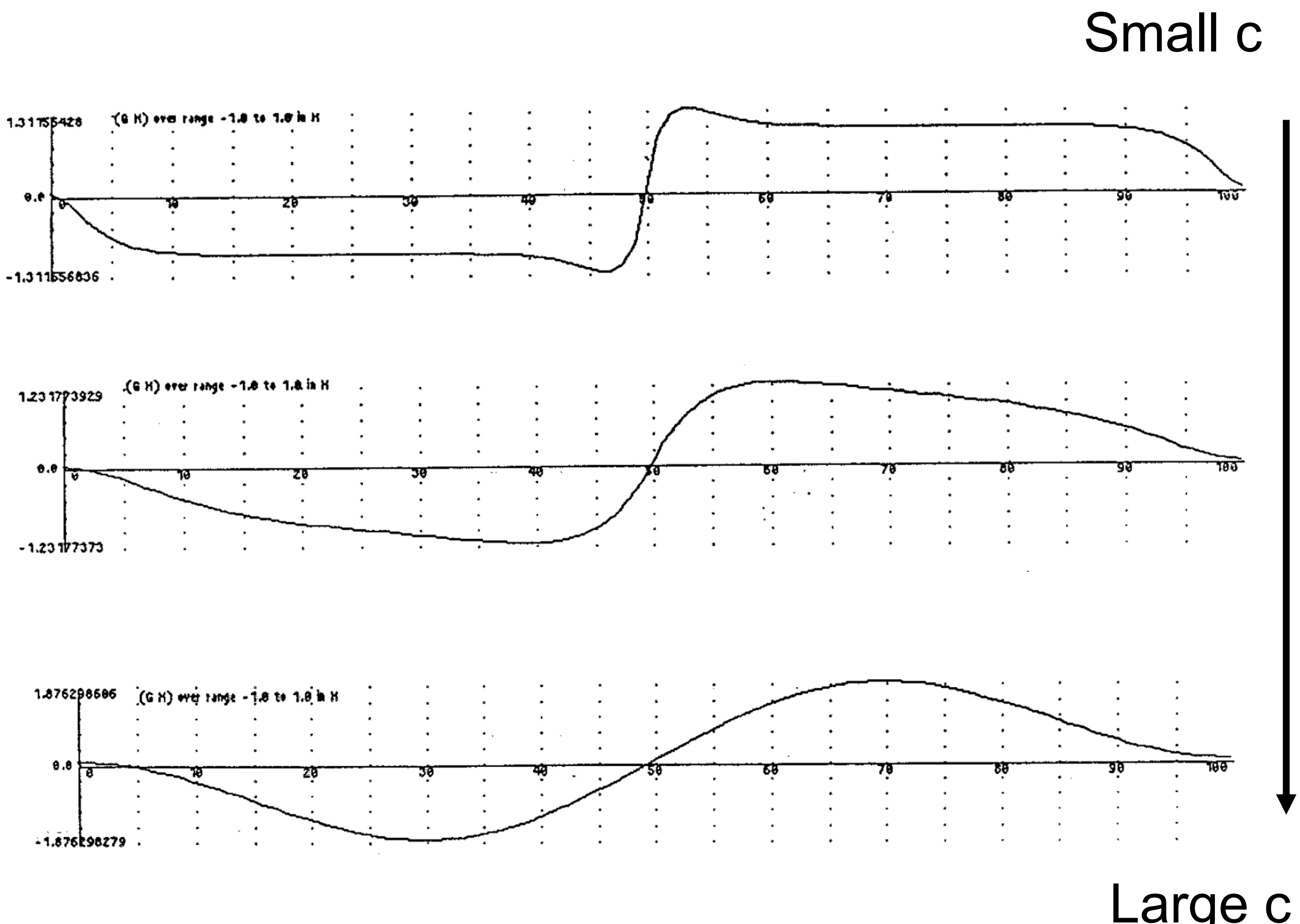
  should be minimized

- Overall goal function is a linear combination of the two criteria

$$\max\left(\mathrm{SNR} \times \mathrm{LOC} + c\frac{1}{N_{\max}}\right)$$

  the solution depends on c
  empirically selected

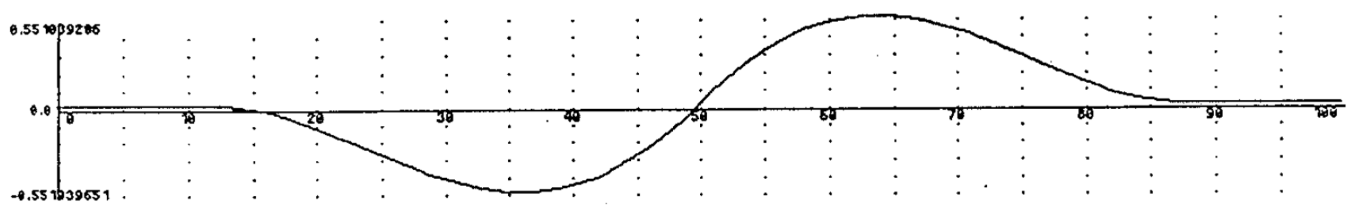
1D optimal filter



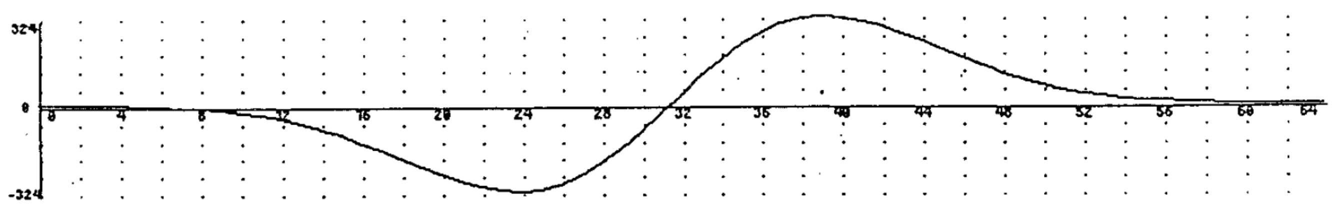
1D optimal filter

Resembles the first derivative of the Gaussian

Canny selection



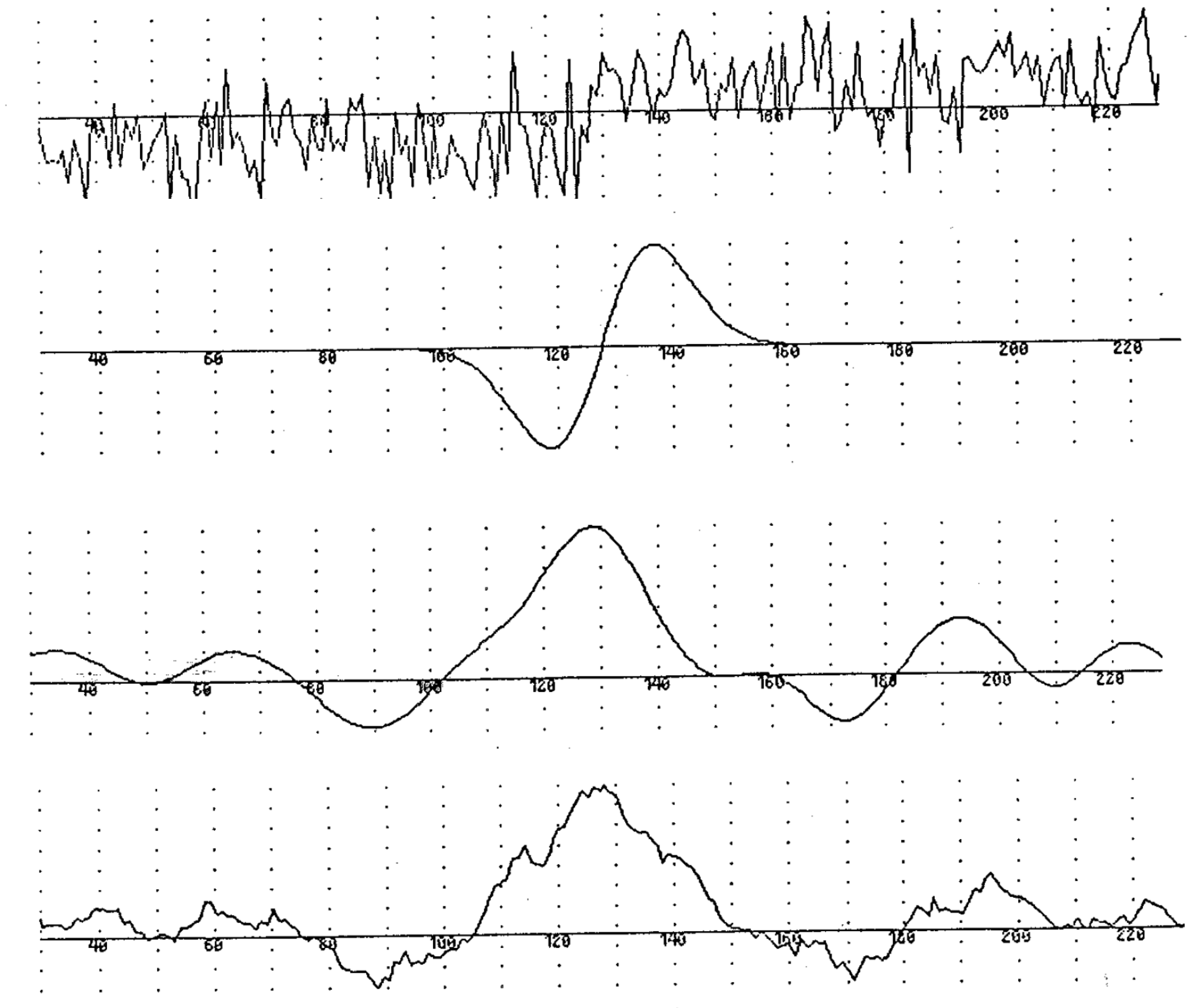
Gaussian derivative



Another first derivative based detector

Computer Vision

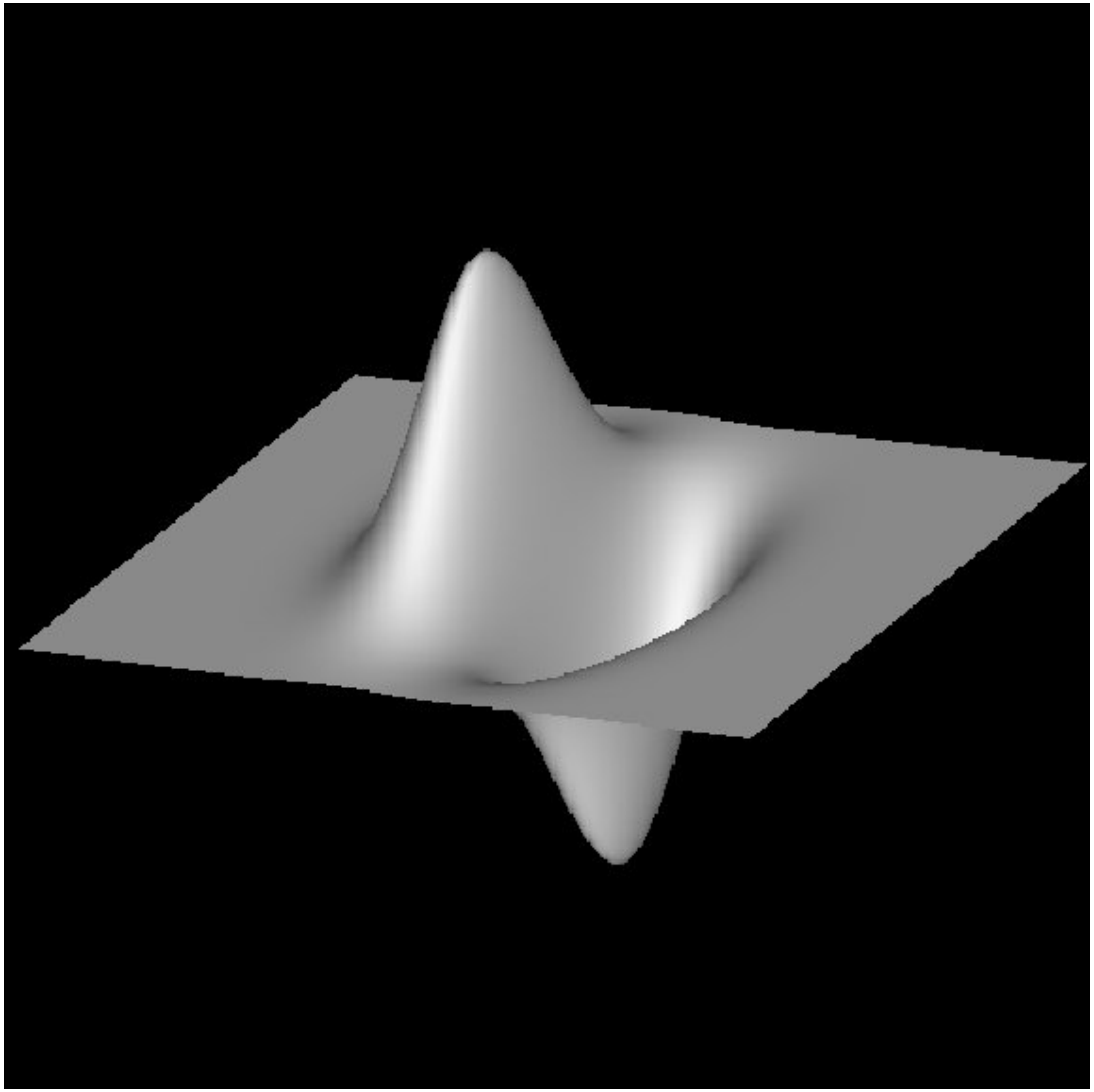
Optimal filter



Canny filter in nD

- The Canny filter is essentially 1D
- Extension to higher dimensions
  - simplified edge model
  - intensity variation only orthogonal to the edge
  - no intensity change along the edge
- Combination of two filtering principles
  - 1D Canny filter across the edge
  - (n-1)D smoothing filter along the edge
    Gaussian smoothing is used
- The effective filter is a directional derivative of Gaussian

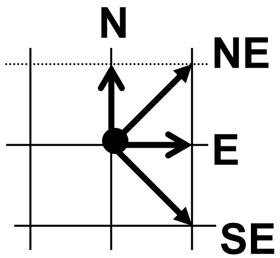
The 2D Canny filter



2D implementation on the discrete image raster

- Faithful implementation by selecting gradient direction: does not respect discretization

- Estimation of directional derivatives instead considering neighbours on the image raster



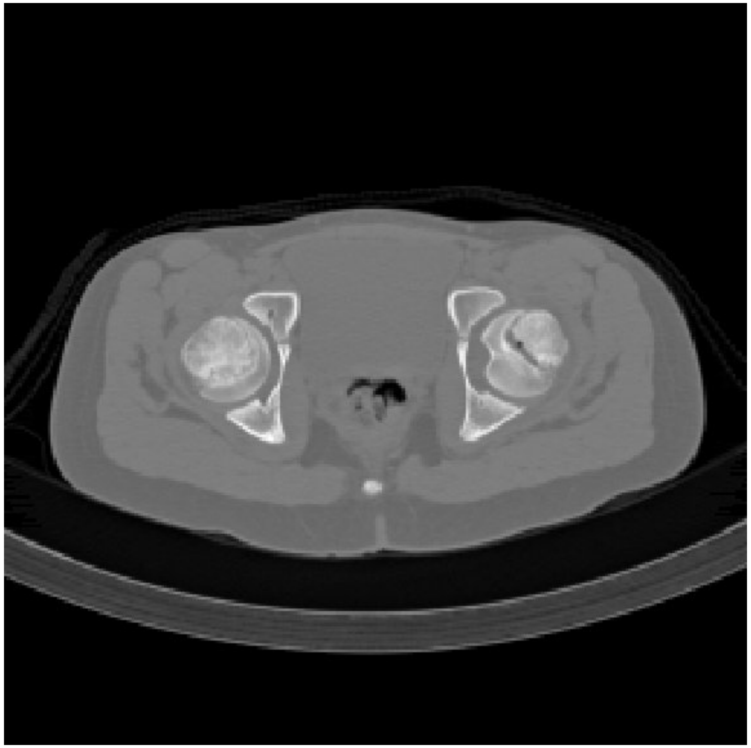
- Start with 2D Gaussian smoothing $f = G * I$
- Directional derivatives from discrete differences

$$f'_N = f(i, j+1) - f(i,j); f'_{NE}(i,j) = (f(i+1, j+1) - f(i,j))/\sqrt{2}$$

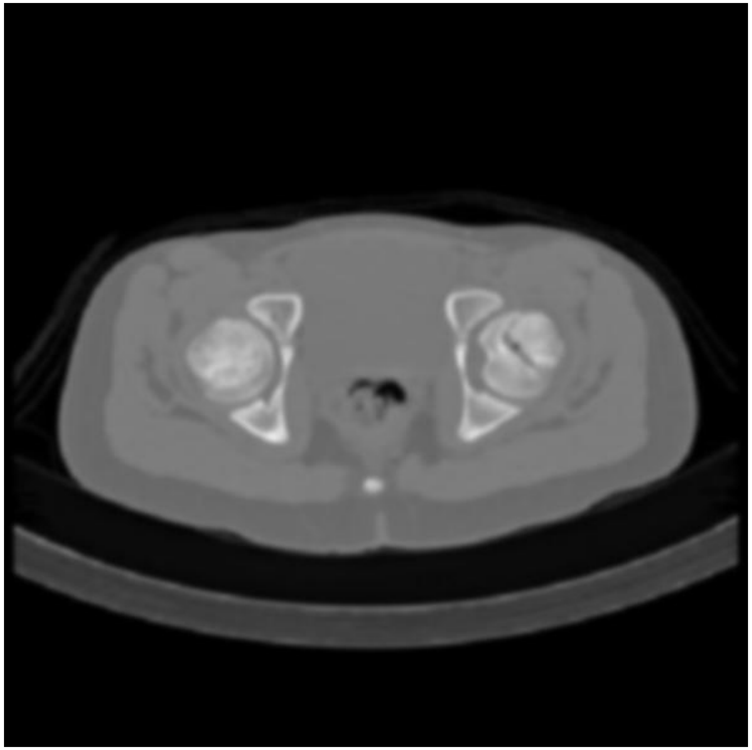
$$f'_E = f(i+1, j) - f(i,j); f'_{SE}(i,j) = (f(i+1, j-1) - f(i,j))/\sqrt{2}$$

- Selecting the maximum as gradient approximation

Canny 2D results



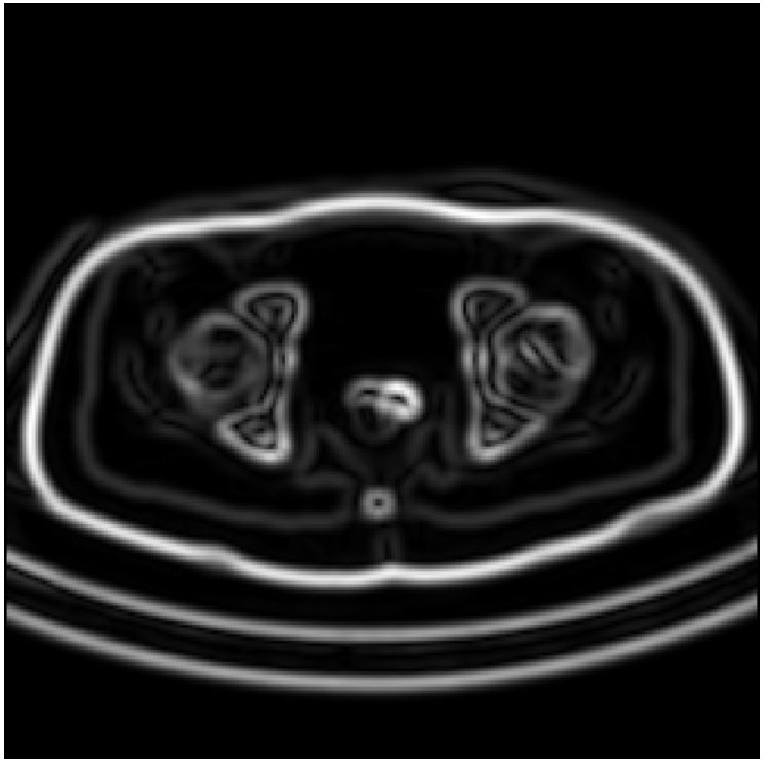
original image



Gaussian smoothing

Computer Vision

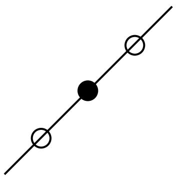
Canny 2D results



Gradient approximation

Post-processing steps

- Non-maximum suppression
  - Comparing derivatives at the two neighbours along the selected direction

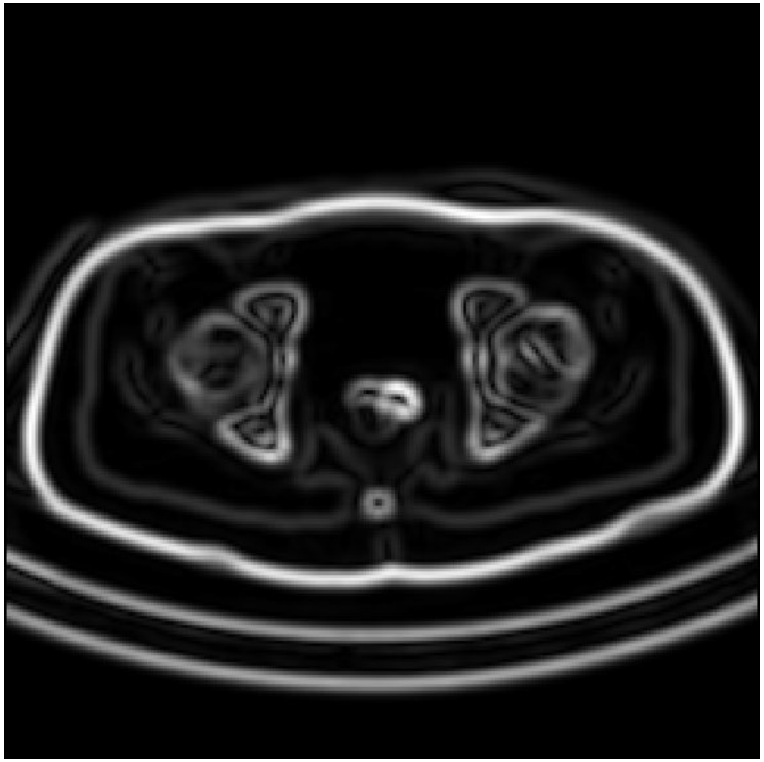


  - Keeping only values which are not smaller than any of them

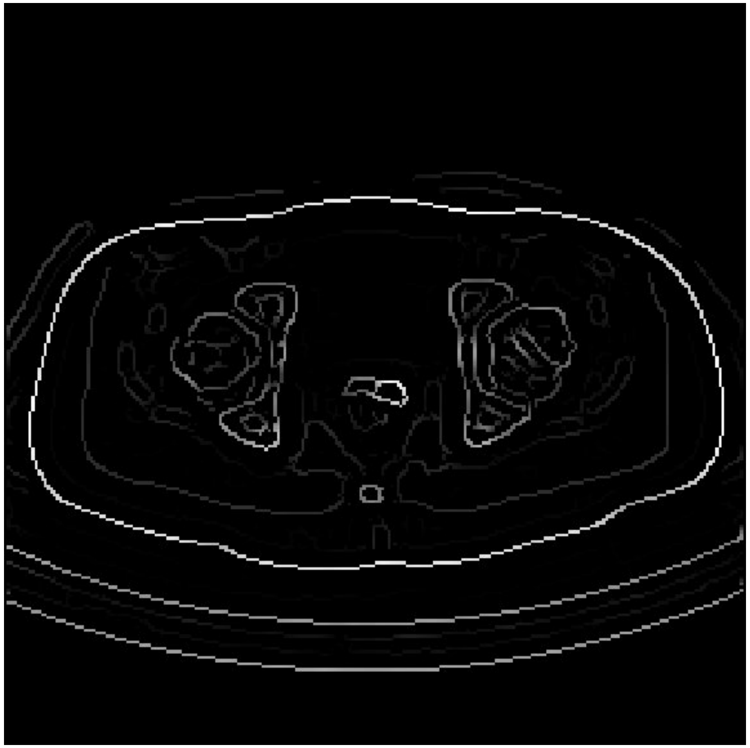
- Hysteresis thresholding
  - using two treshold values t_{low} and t_{high}
  - keep class 1 edge pixels for which $|f(i,j)| \geq t_{high}$
  - discard class 2 edge pixels for which $|f(i,j)| < t_{low}$
  - for class 3 edge pixels $t_{high} > |f(i,j)| \geq t_{low}$ keep them only if connected to class 1pixels through other class 3 pixels

Computer
Vision

Canny 2D results



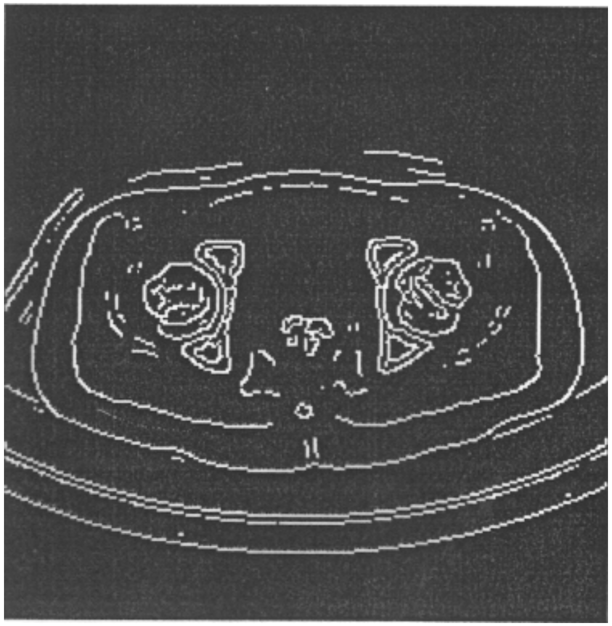
Gradient approximation



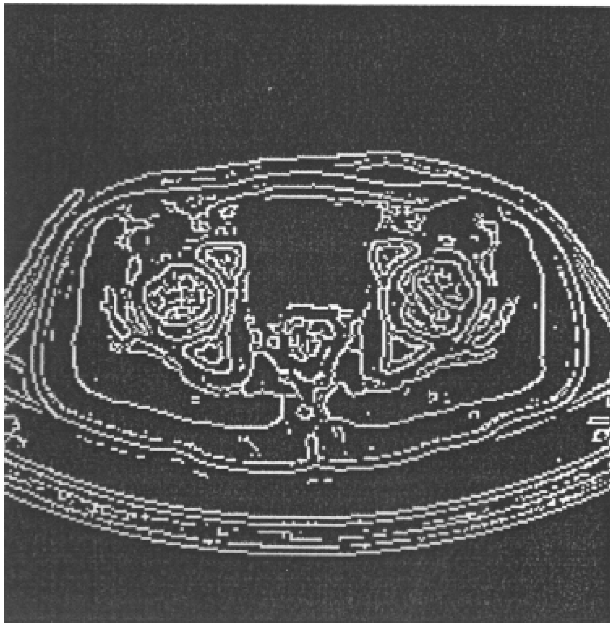
non-maximum suppresion

Computer Vision

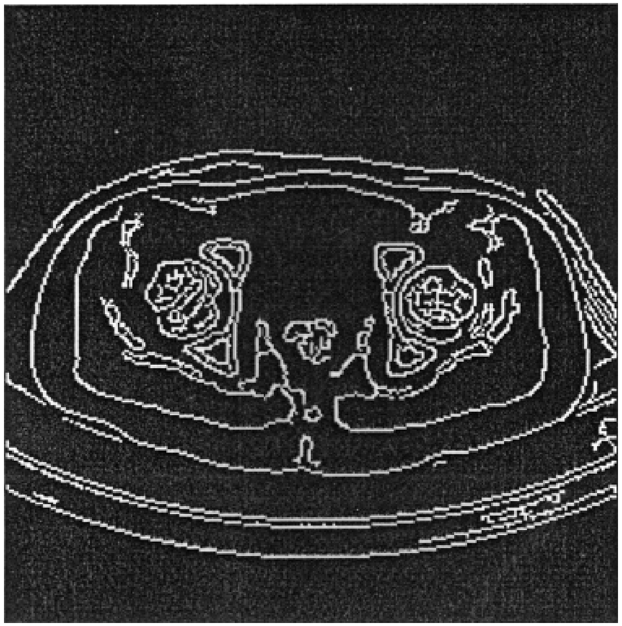
Canny 2D results



threshold t_{high}



threshold t_{low}



hysteresis thresholding

Remarks to the Canny filter

- The state-of-the-art edge detector even today
- Very efficient implementation
  no interpolation is needed as respecting the raster
- Post-processing is the major contribution
- Can be applied to any gradient-based edge detection scheme
- Fails where the simplified edge model is wrong
  - crossing, corners, …
  - gaps can be created
  - mainly due to non-maximum suppression
- Hysteresis thresholding is only effective in 2D

A Post Scriptum

e.g. Sobel filter reflects intensity pattern to be found

other example, line detector

$$\begin{array}{ccc} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{array}$$

in line with the *matched filter theorem*

Outline

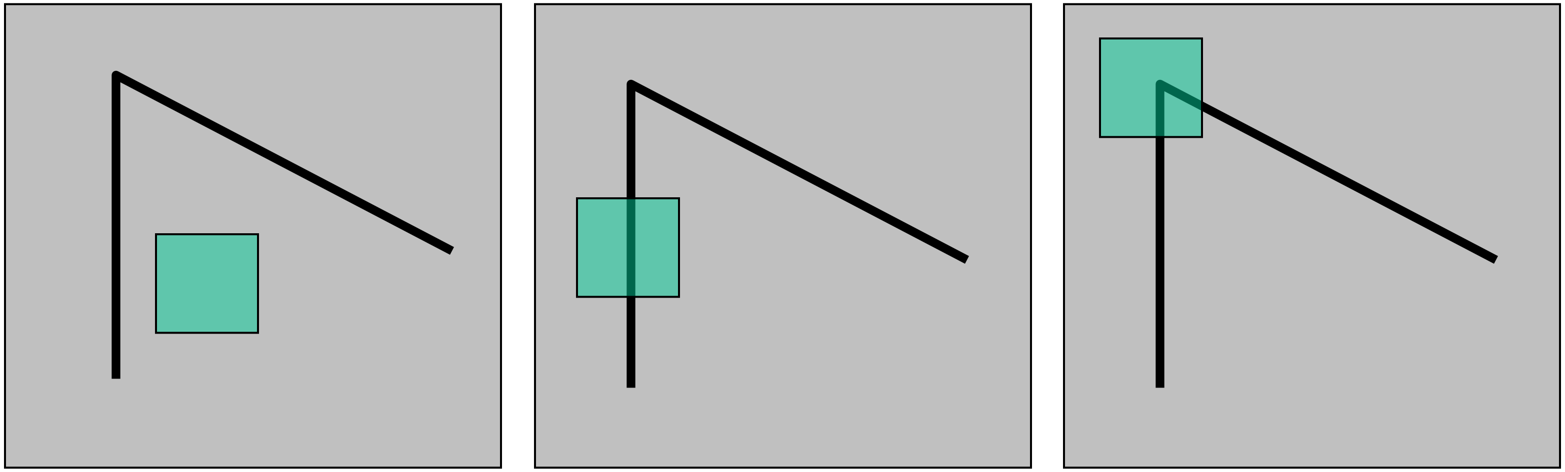
Identifying points of interest

1. Edge detection
   a. Gradient operators
   b. Zero-crossings of Laplacians
   c. Canny Edge Detector
2. **Corner detection**

Computer Vision

Uniqueness of a patch

consider the pixel pattern within the green patches:



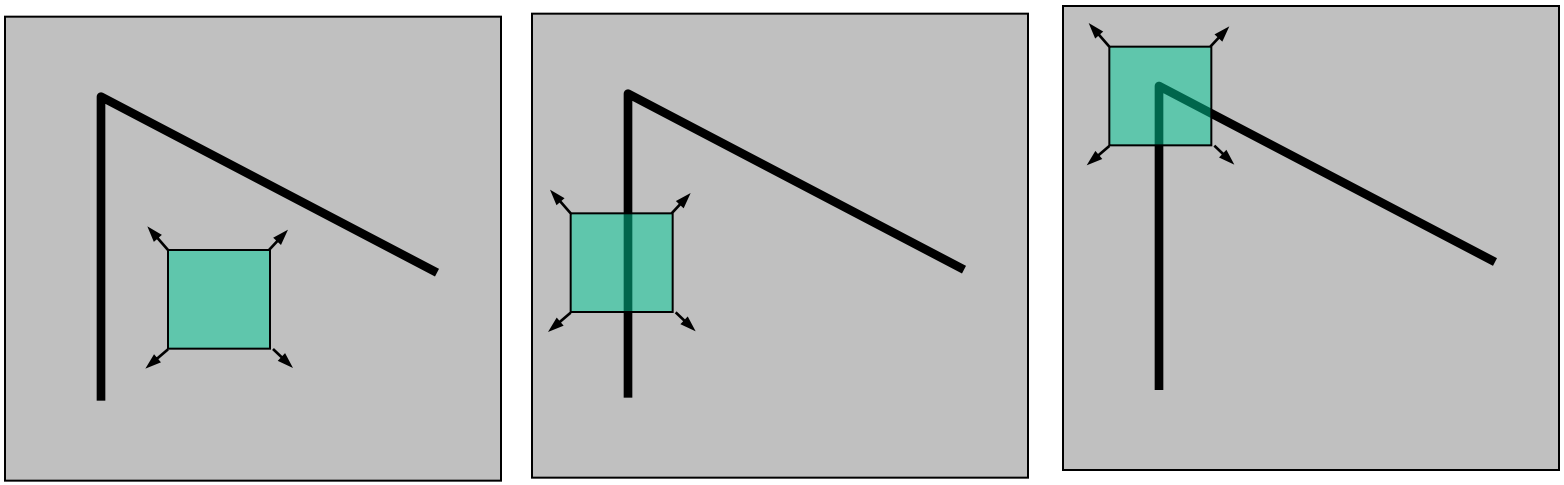
Think of the wedge as being darker than the background, i.e. not as drawn in the figure…

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

How do the patterns change upon a shift?

Is it well localizable?



"flat" region:
no change in
all directions

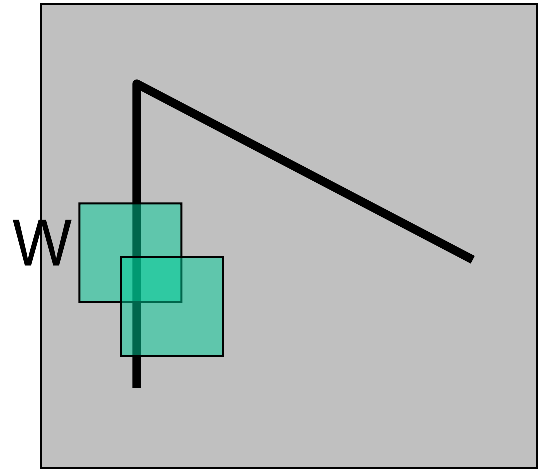
"edge":
no change along
the edge direction

"corner":
significant change
in all directions

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

Consider shifting the patch or `window' W by (u,v)



- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines the SSD "error" *E(u,v)*:

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

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

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

Taylor Series expansion of I:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (u,v) is small, then 1st order appr. is good

$$I(x+u, y+v) \approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

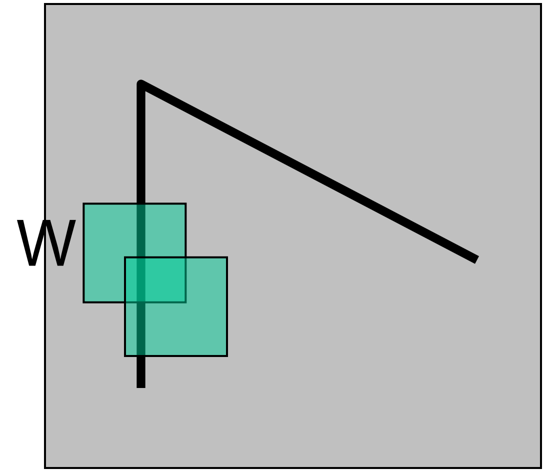
$$\approx I(x,y) + [I_x \ \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

Plugging this into the formula at the top…

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Computer Vision

Uniqueness of a patch



Then, with shorthand: $I_x = \frac{\partial I}{\partial x}$

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

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch

This can be rewritten further as:

$$E(u,v) = \sum_{(x,y)\in W} [u \ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Which directions *(u,v)* will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of *H*

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Computer Vision

Uniqueness of a patch

This can be rewritten further as:

$$E(u,v) = \sum_{(x,y)\in W} [u \ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

Eigenvalues and eigenvectors of H

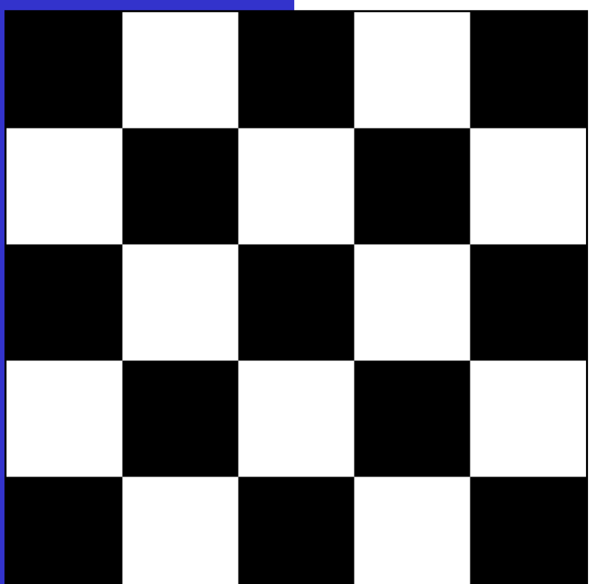
- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E
- λ_- = amount of increase in direction x_-

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

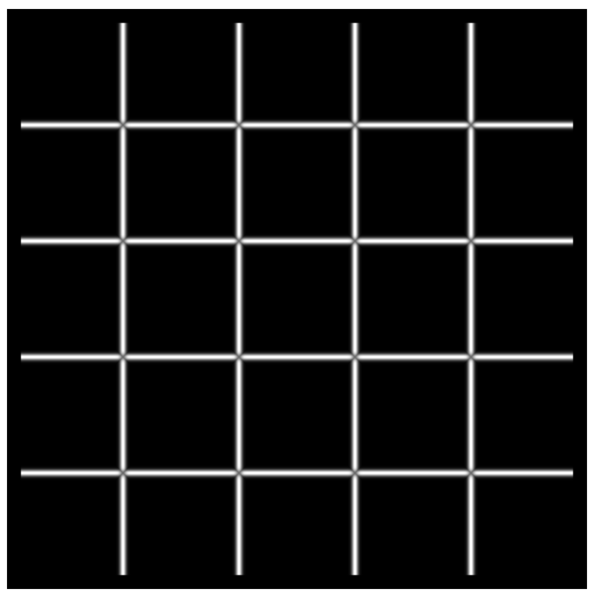
Uniqueness of a patch

Want *E(u,v)* to be *large* for small shifts in *all* directions

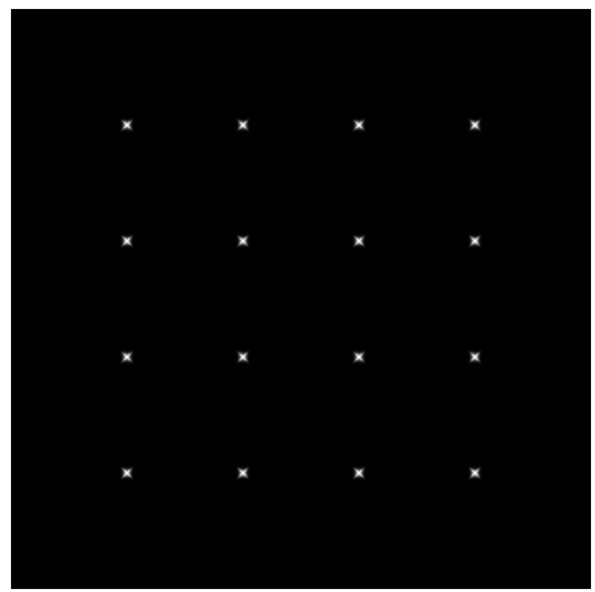
- the *minimum* of *E(u,v)* should be large, over all unit vectors $[u\ v]$
- this minimum is given by the smaller eigenvalue (λ_-) of *H*



example image



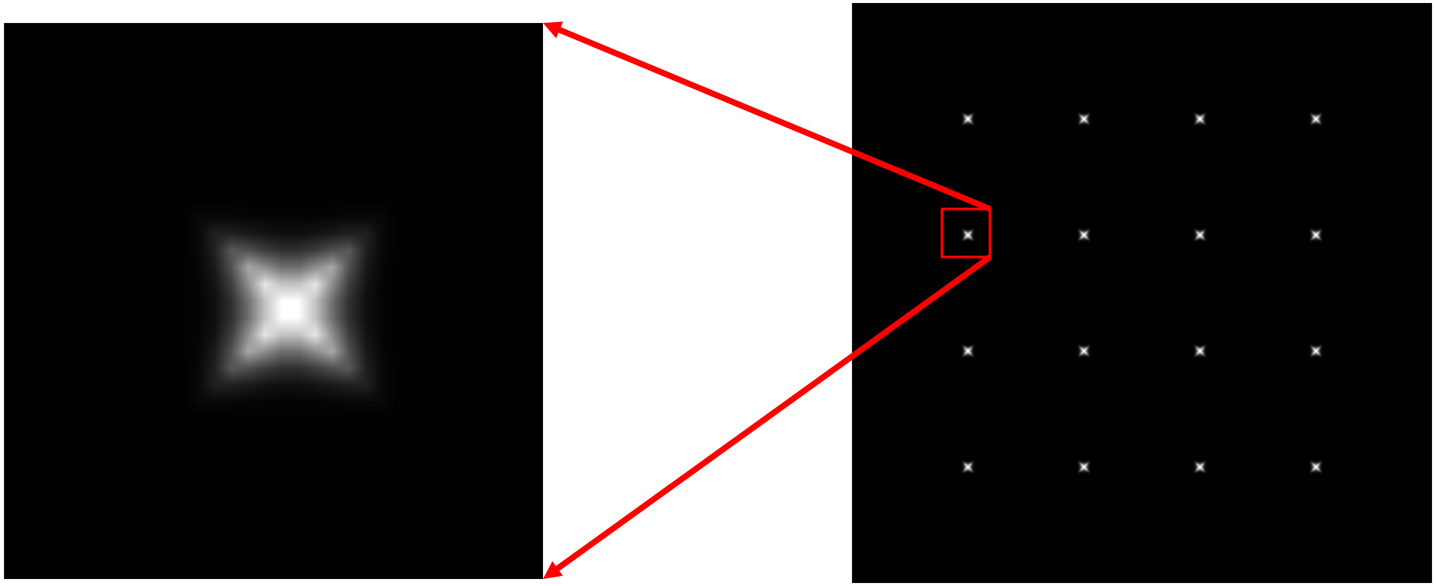
λ_+



λ_-

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Uniqueness of a patch



λ_-

Slide adapted from Darya Frolova, Denis Simakov, Deva Ramanan

Interest points

Corners are the most prominent example of so-called `Interest Points', i.e. points that can be well localized in different views of a scene

`Blobs' are another, as we will see… but also a blob is a region with intensity changes in multiple directions, similar to corners

The Harris corner detector

Goal : one approach that distinguishes

- *1. homogeneous areas*
- *2. edges*
- *3. corners*

Key : looking at intensity variations in different directions :

- *1. small everywhere*
- *2. large in one direction, small in the others*
- *3. Large in all directions*

The Harris corner detector

Homogeneous region   Edge region   Corner region

Approach : find the directions of minimal and maximal change

Second order moments of the intensity variations also called the structure tensor:

$$\begin{pmatrix} \left\langle \left(\frac{\partial f}{\partial x}\right)^2 \right\rangle & \left\langle \left(\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\right) \right\rangle \\ \left\langle \left(\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\right) \right\rangle & \left\langle \left(\frac{\partial f}{\partial y}\right)^2 \right\rangle \end{pmatrix}$$

Look for the *eigenvectors* and *eigenvalues*

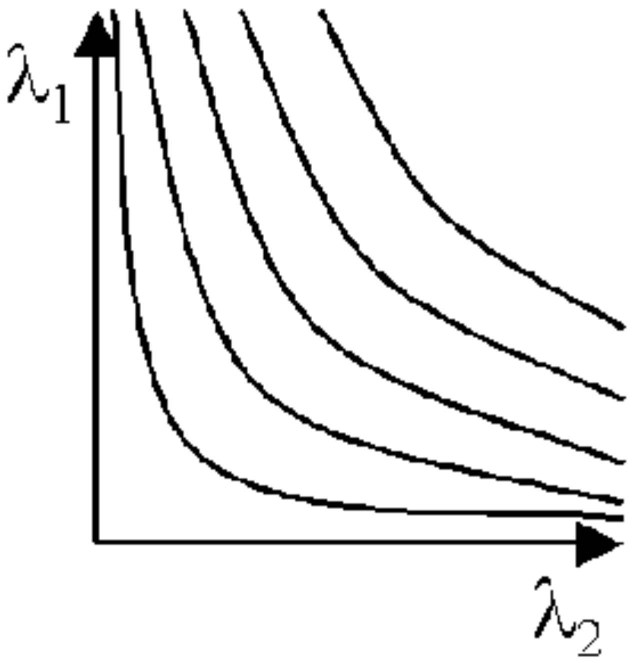
The Harris corner detector

The classification can be made as

- 1. two small eigenvalues
- 2. one large and one small eigenvalue
- 3. two large eigenvalues

First attempt : determinant of the 2nd-order matrix,

i.e. the product of the eigenvalues :

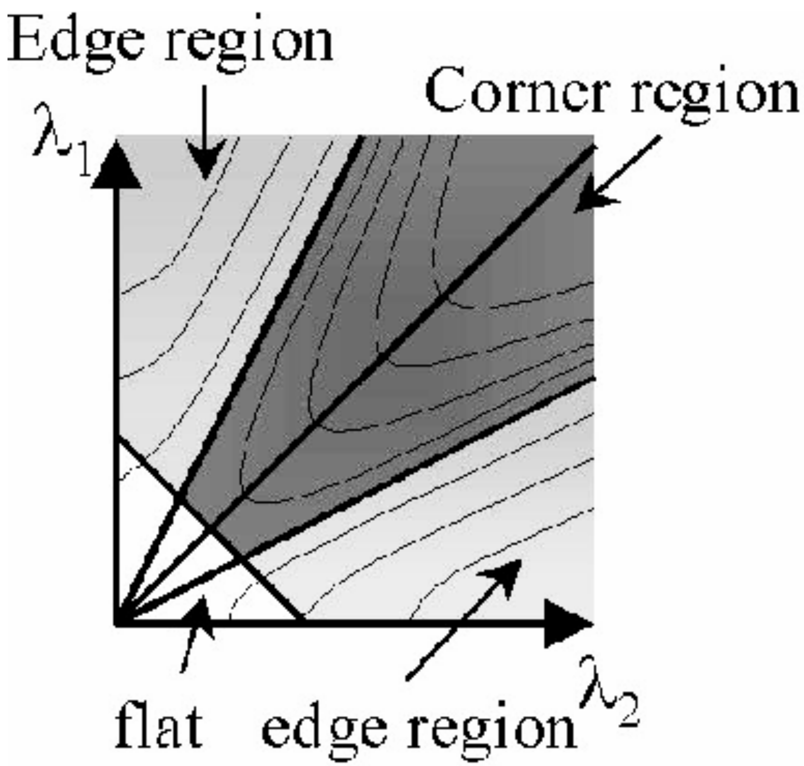


The Harris corner detector

Edges are now too vague a class : one very large eigenvalue can still trigger a corner response.

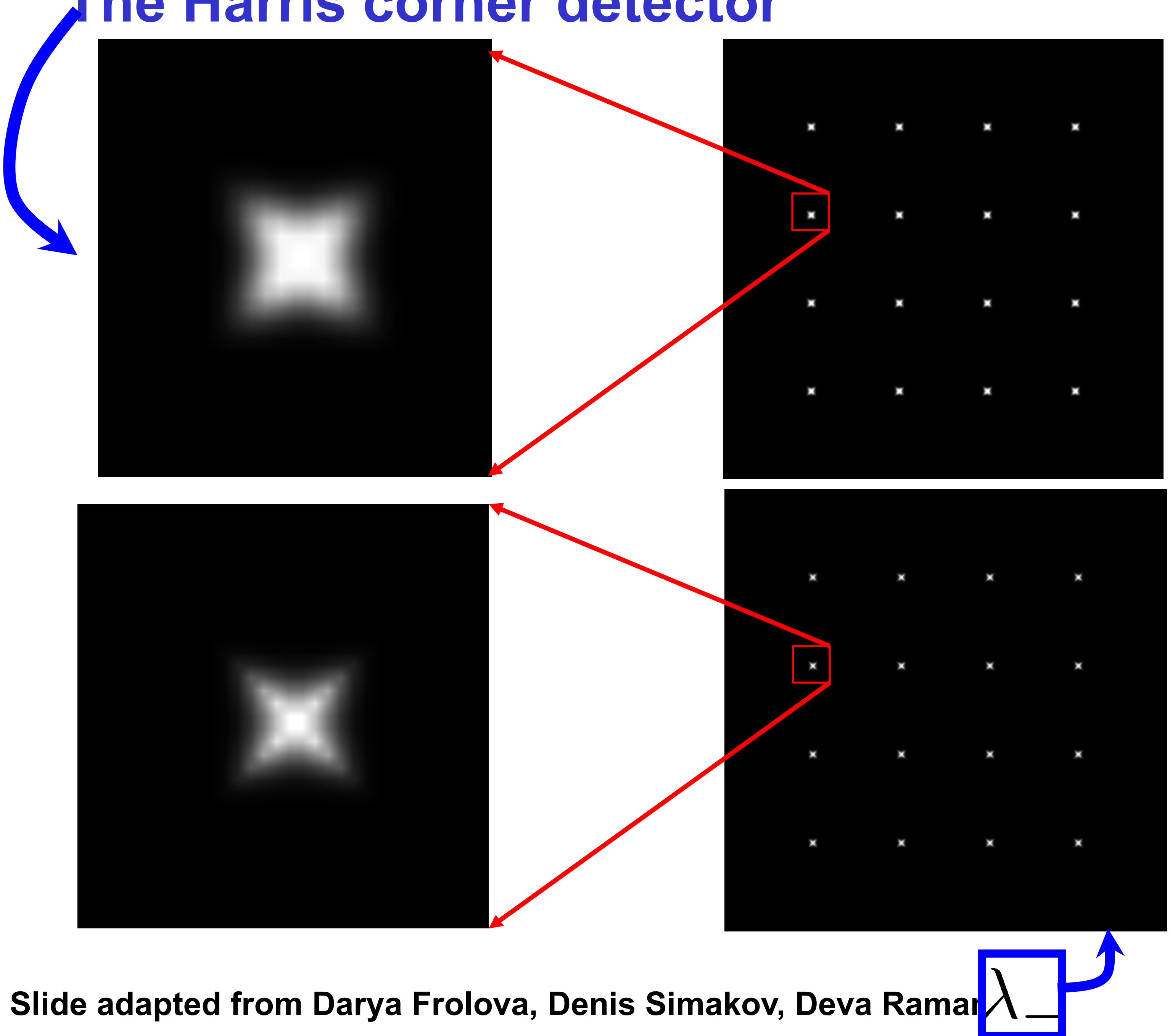
A refined strategy :

Use iso-lines for *Determinant - k (Trace)*^{2}.



Computer Vision

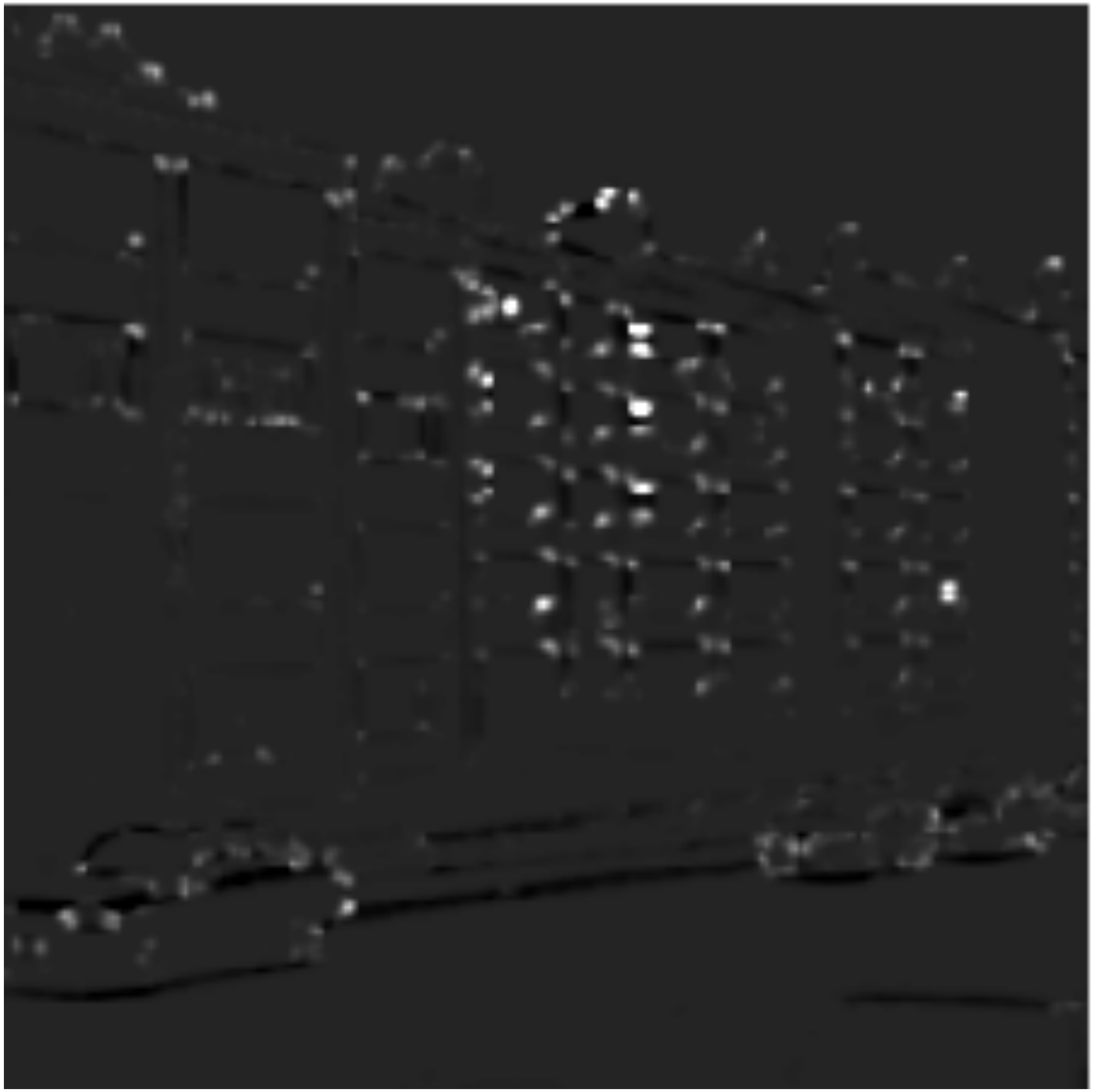
The Harris corner detector



Slide adapted from Darya Frolova, Denis Simakov, Deva Raman

The Harris corner detector

The Harris corner detector



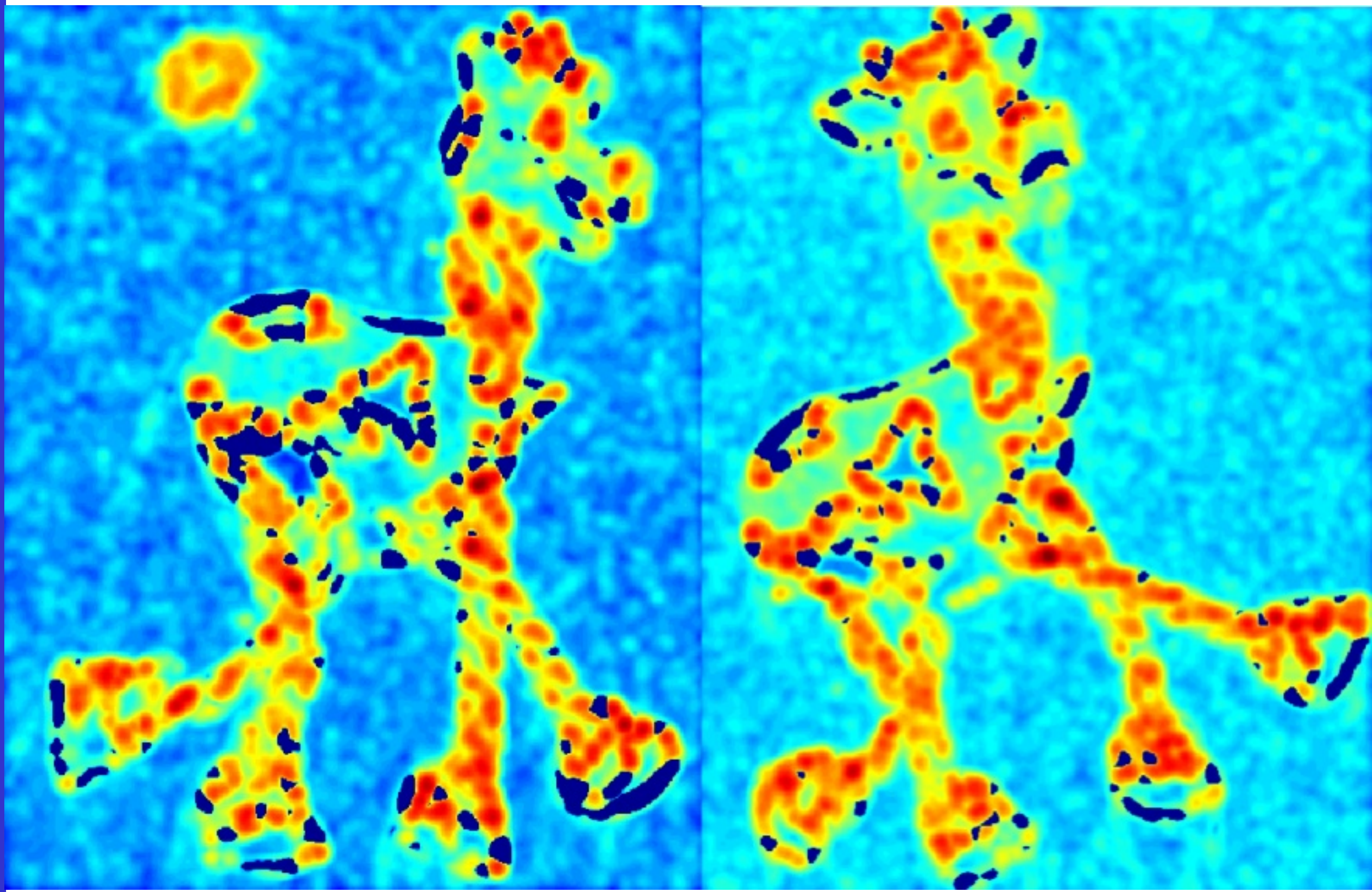
Computer Vision

The Harris corner detector

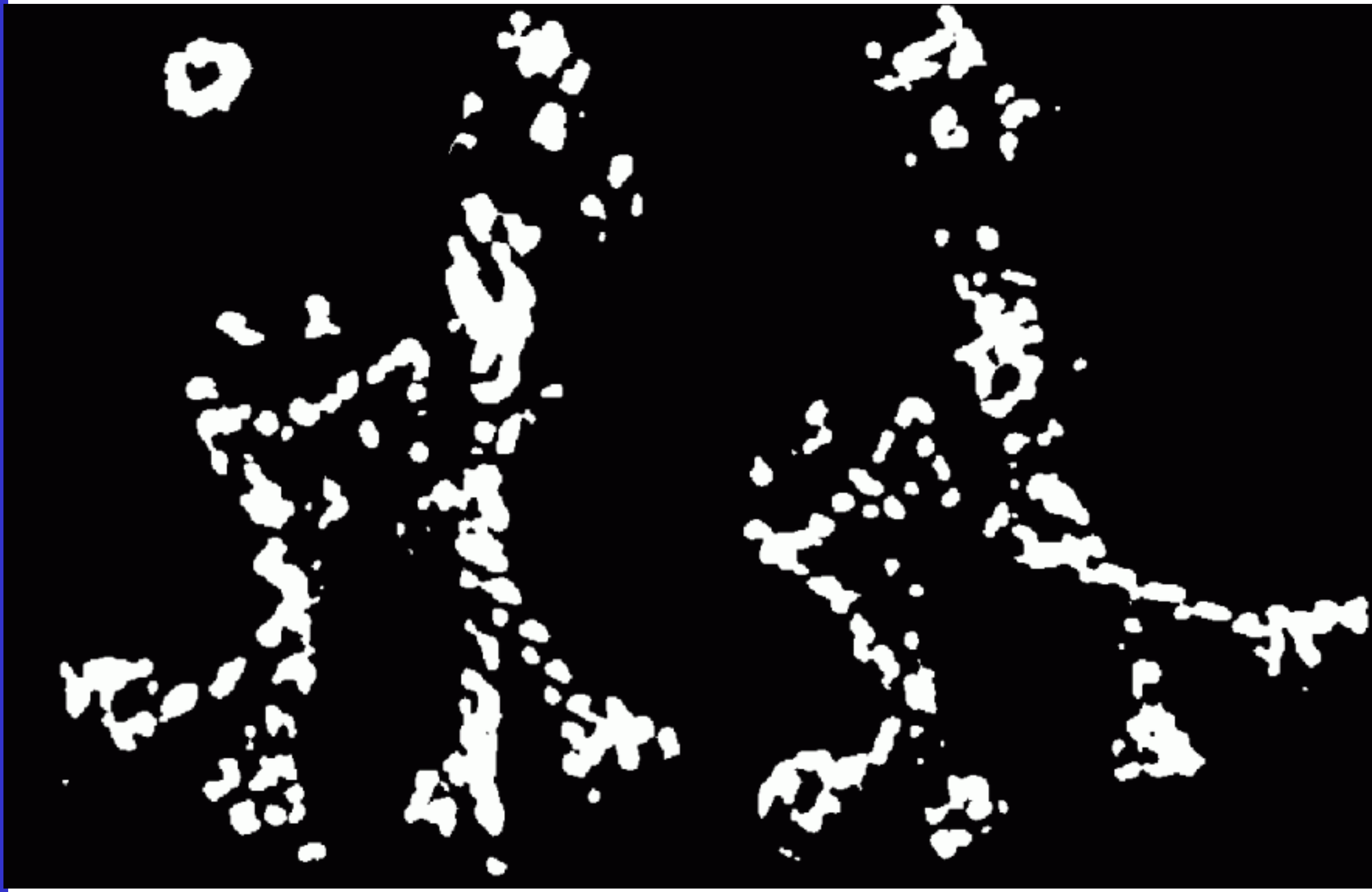
2 views of an object… are the corners stable?

Computer Vision

The Harris corner detector

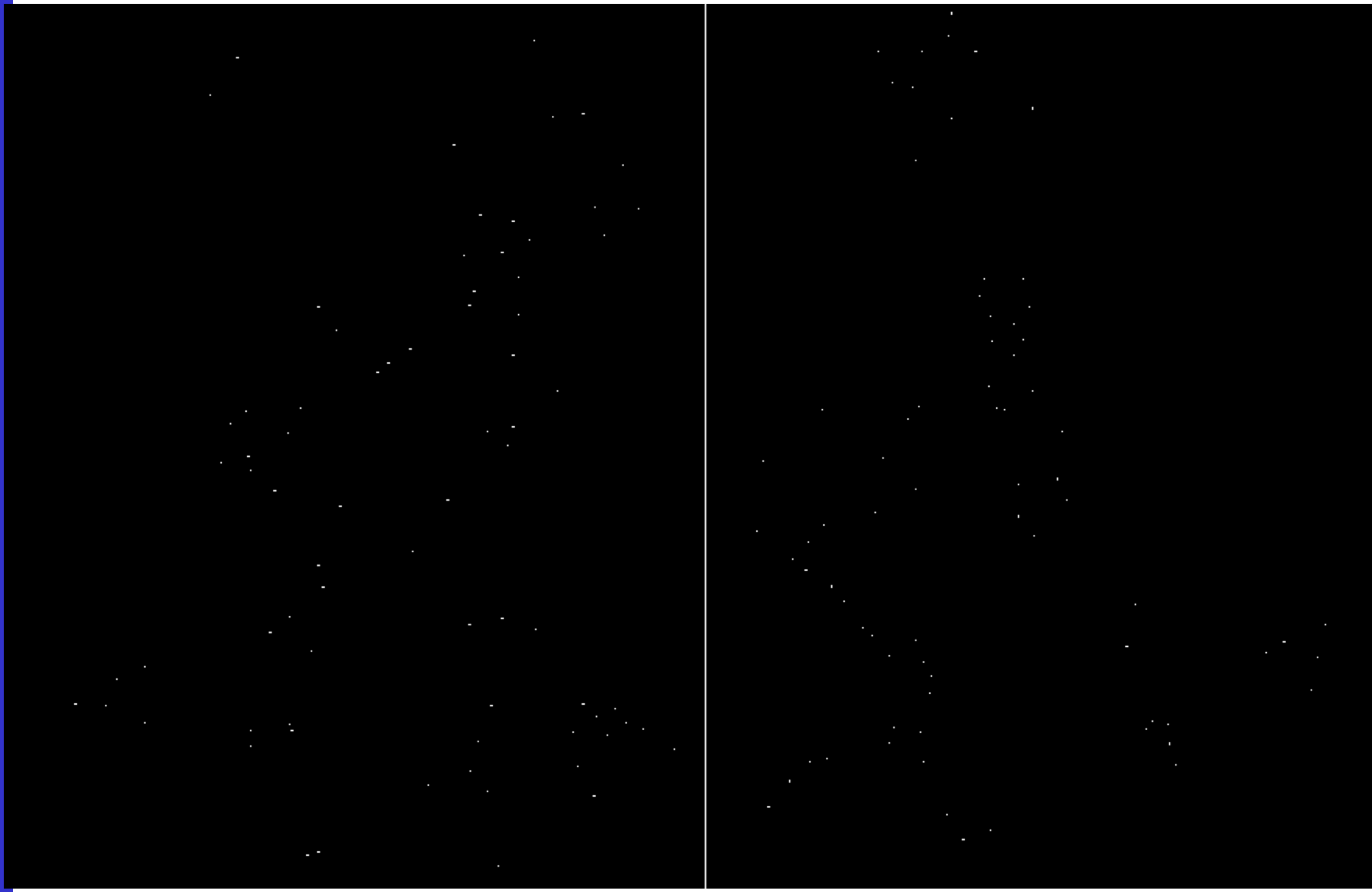


The Harris corner detector

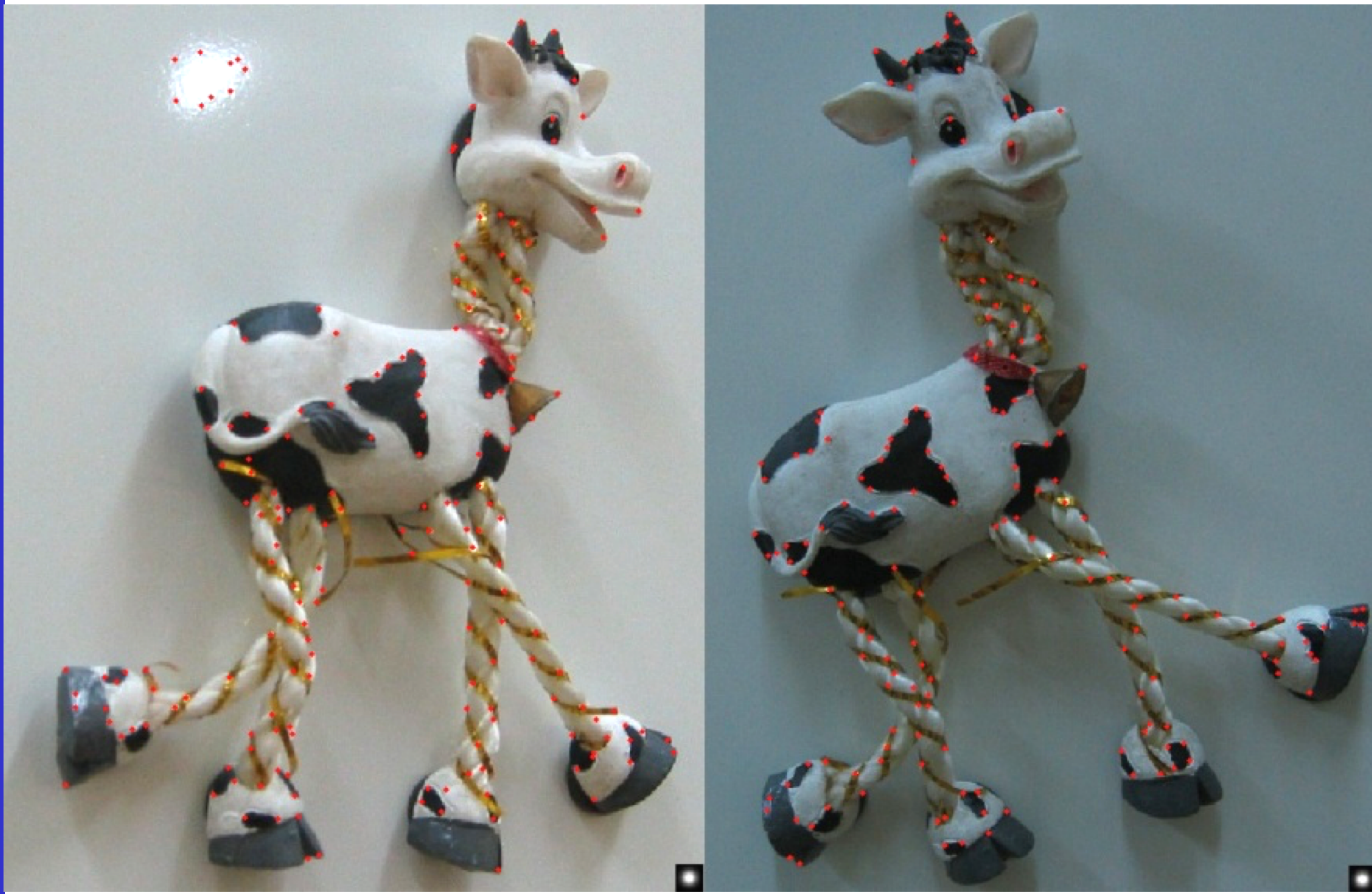


Computer Vision

The Harris corner detector



The Harris corner detector



PART 2: Extracting descriptors around interest points

Need for a descriptor:

A feature should capture something *discriminative* about a well *localizable* patch of a pattern

There are many corners coming out of our DETECTOR, but they still cannot be told apart

We need to describe their surrounding patch in a way that we can discriminate between them, i.e. we need to build a feature vector for the patch, a so-called DESCRIPTOR

During a MATCHING step, the descriptors can then be compared

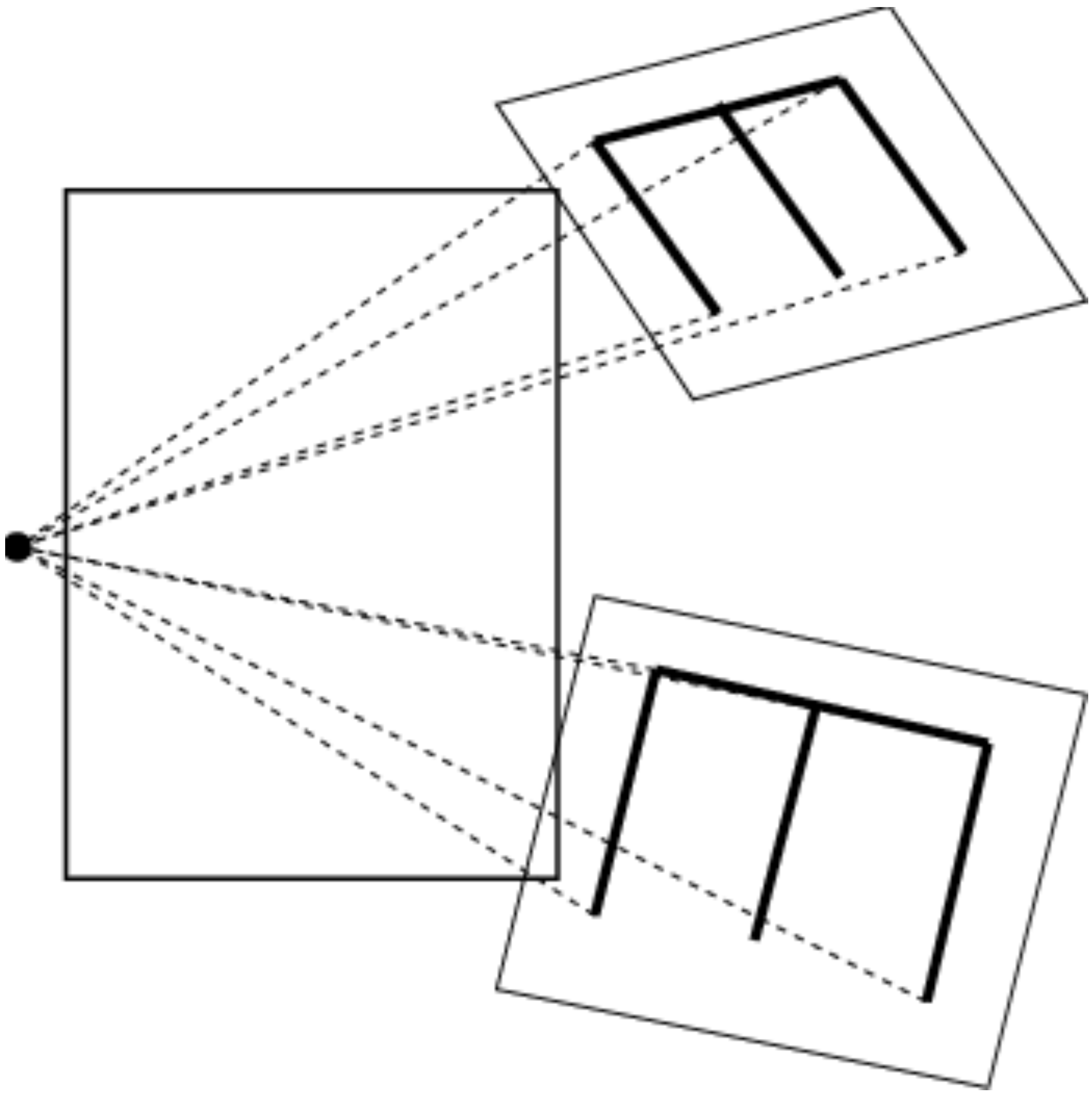
Additional requirement on the descriptor:

Invariance under geom./phot. changes

A local patch is small,
hence probably rather *planar.*

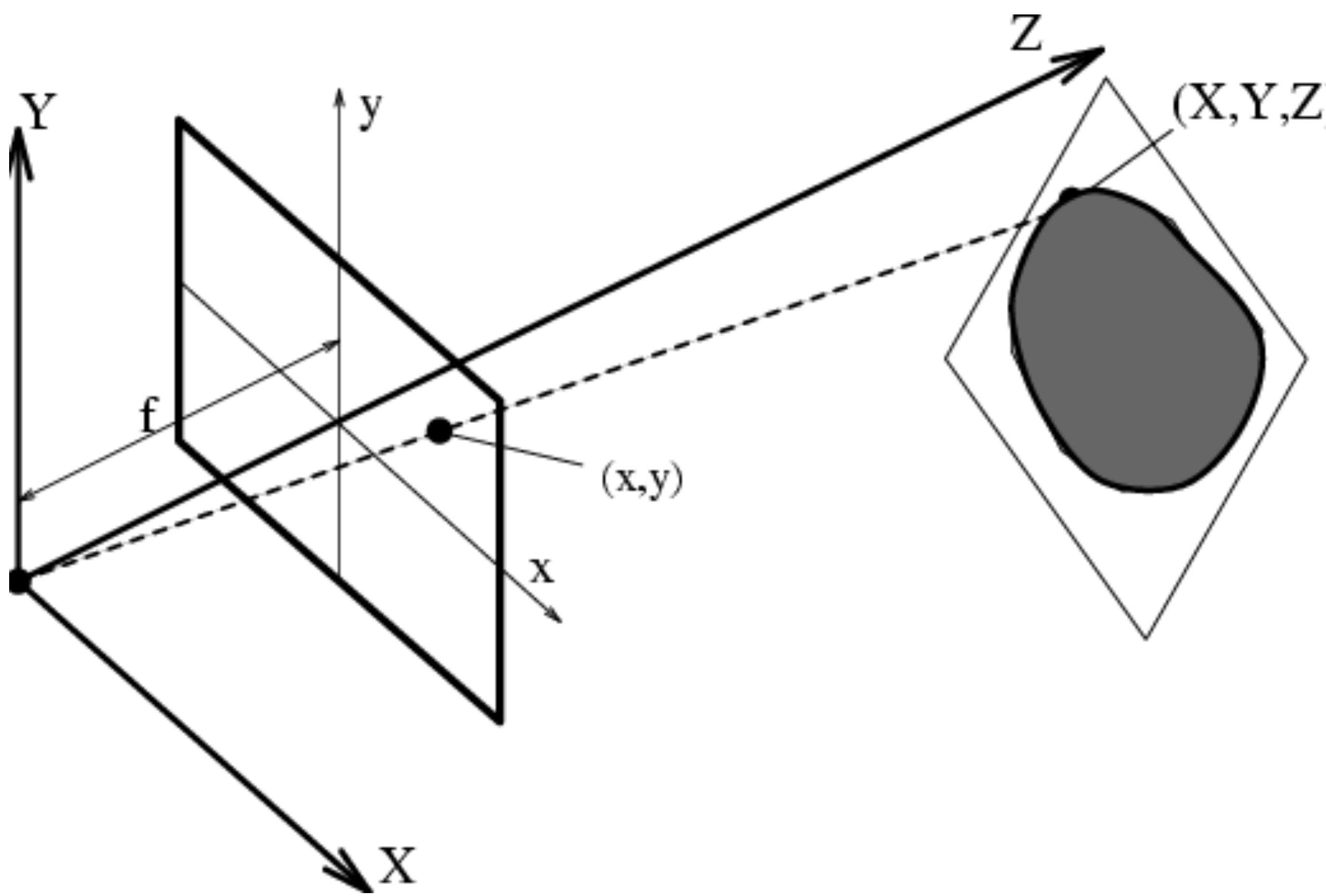
But how do planar patches deform when
looking at their image projection?
i.e. we determine the *geometric* changes
the descriptor should remain invariant under

Planar pattern projections to be compared



Projection : a camera model

Reversed center-image pinhole model :



Projection : equations

$$x = f\,\frac{X}{Z} \qquad y = f\,\frac{Y}{Z}$$

an approximation...

special cases :

1. Z constant, or
2. object small compared to its distance to the camera

$$\left.\begin{aligned} x &= kX \\ y &= kY \end{aligned}\right\} \quad \text{pseudo-perspective}$$

Deformations under projection

In particular, how do different views
of the same planar shape / contour differ ?

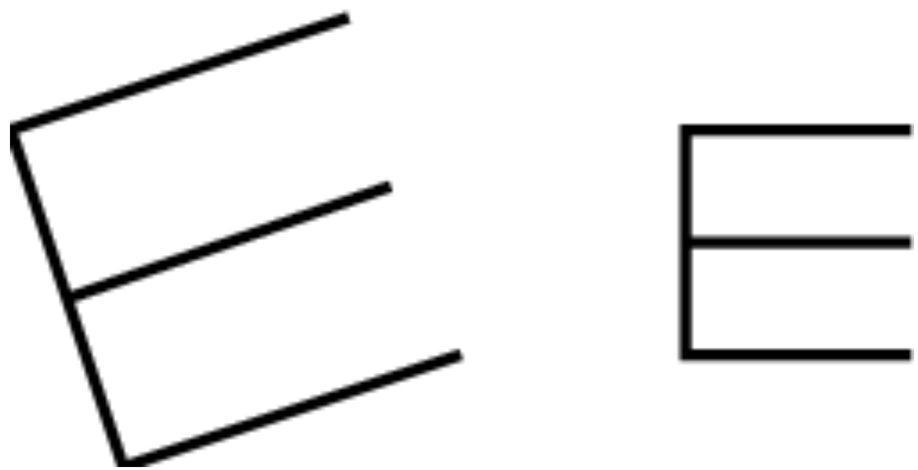
we study 3 cases :

1. viewed from a perpendicular direction
2. viewed from any direction but at sufficient distance to use pseudo - perspective
3. viewed from any direction and from any distance

Case 1

Case of fronto-parallel viewing :

SIMILARITY

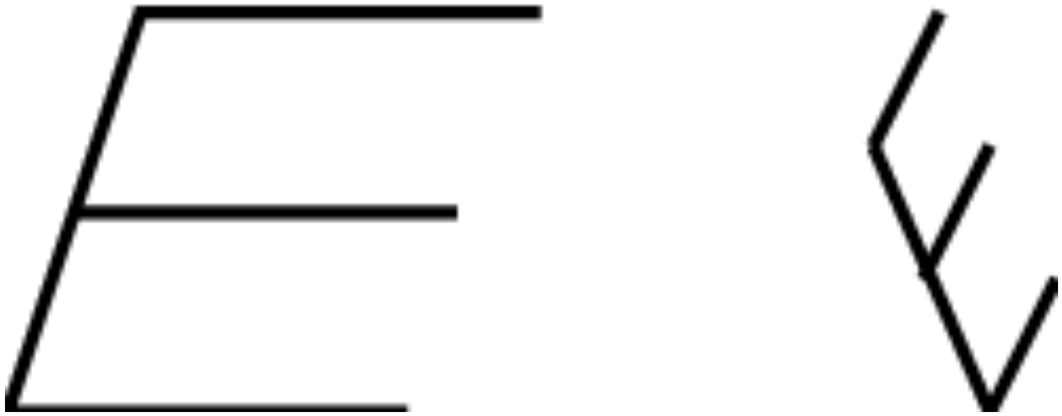


$$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_{\boldsymbol{x}} \\ t_{\boldsymbol{y}} \end{pmatrix}$$

Case 2

Viewing from a distance :

AFFINITY



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Case 3

General viewing conditions:

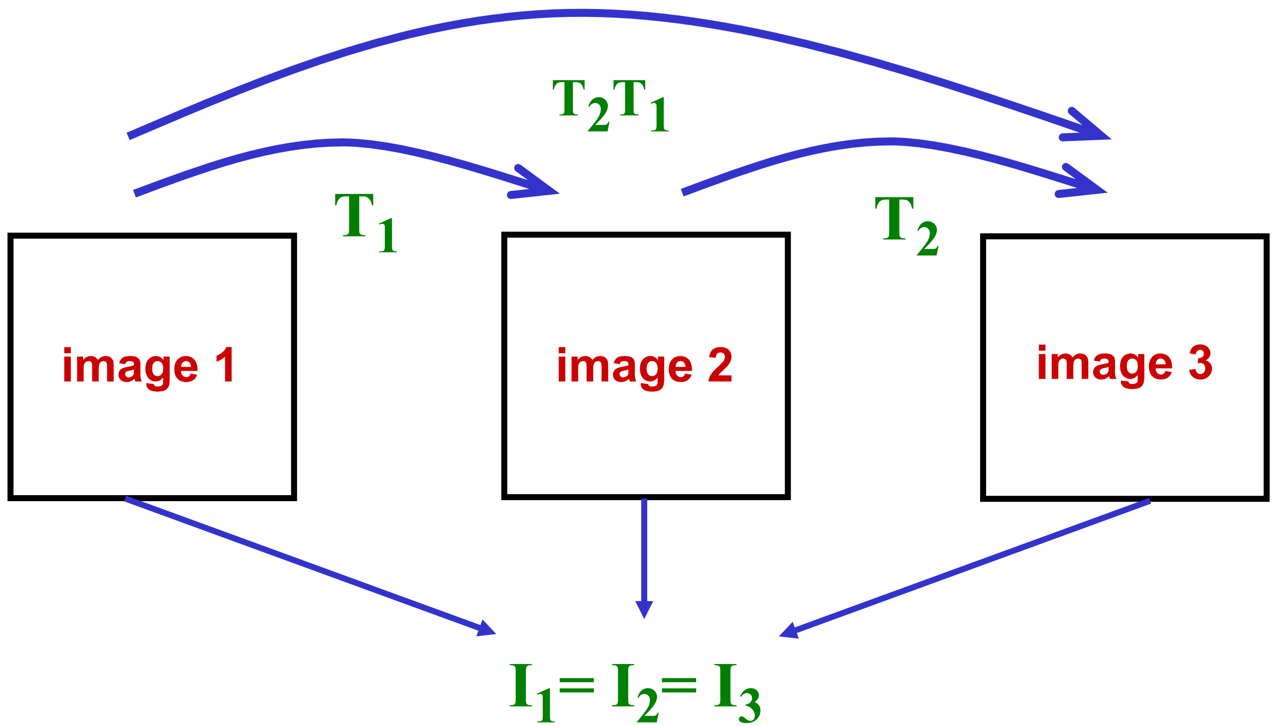
PROJECTIVITY

$$x' = \frac{p_{11}x + p_{12}y + p_{13}}{p_{31}x + p_{32}y + p_{33}}$$

$$y' = \frac{p_{21}x + p_{22}y + p_{23}}{p_{31}x + p_{32}y + p_{33}}$$

Invariance and groups

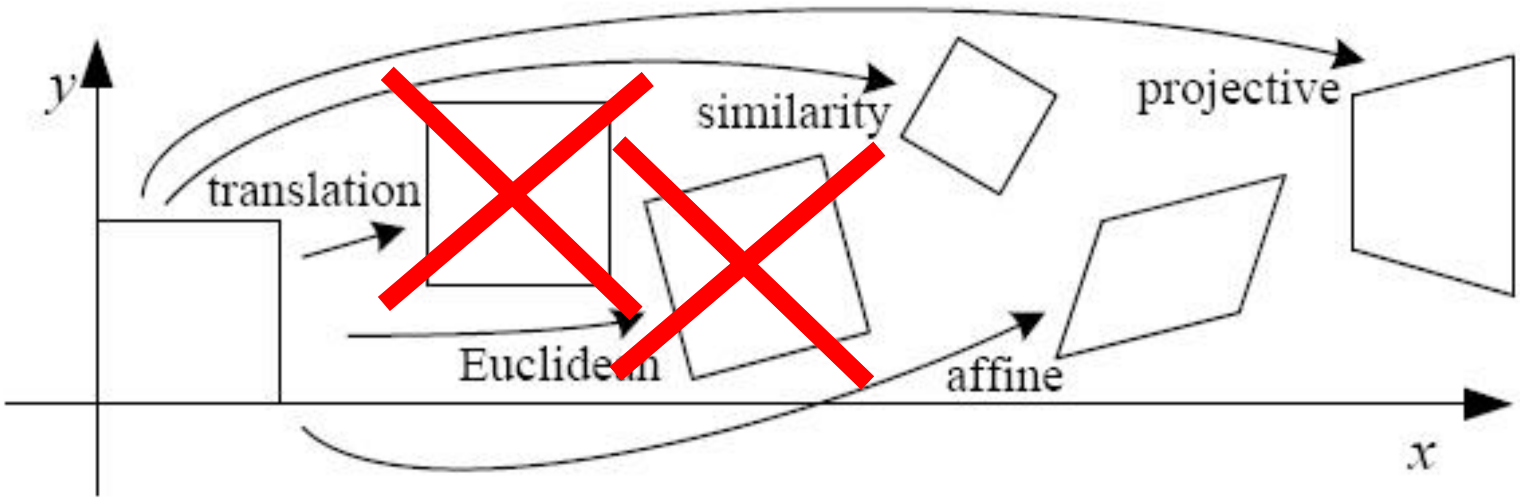
Invariance w.r.t. group actions



Invariance under transformations implies invariance under the smallest encompassing group

Remarks

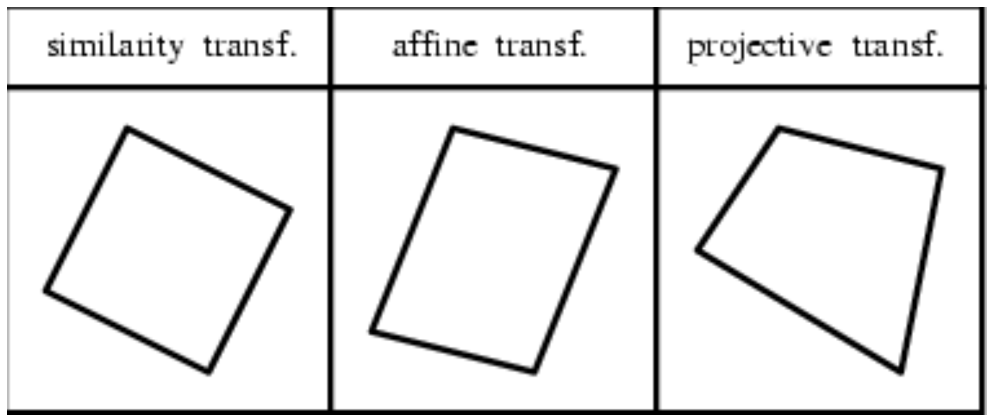
There are more groups, but the ones described seem the most relevant for us



Computer Vision

Remarks

Complexity of the groups goes up with the generality of the viewing conditions, and so does the complexity of the group's invariants



Fewer invariants are found going from left to right

Similarities $\subset$ affinities $\subset$ projectivities

Invar. Proj. $\subset$ invar. Aff. $\subset$ invar. Sim.

PART 3: Examples

Our goal

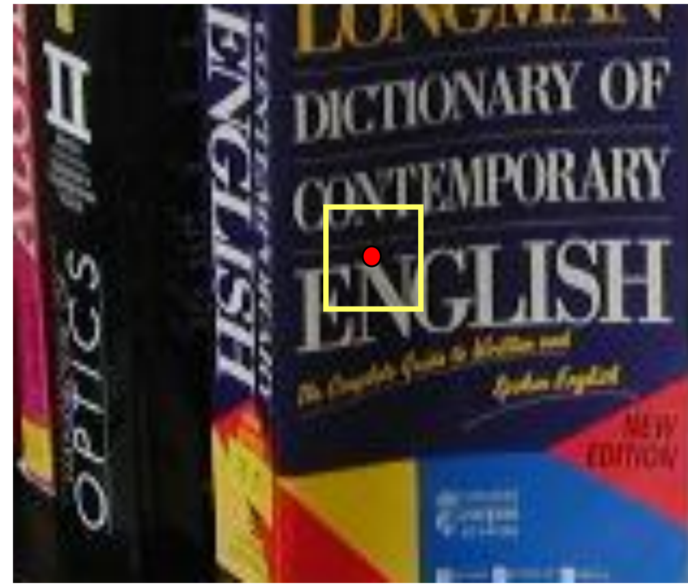
Define good interest points, i.e.

DETECTORS + DESCRIPTORS

The shape of the patch should change with viewpoint for invariance to geometric changes.

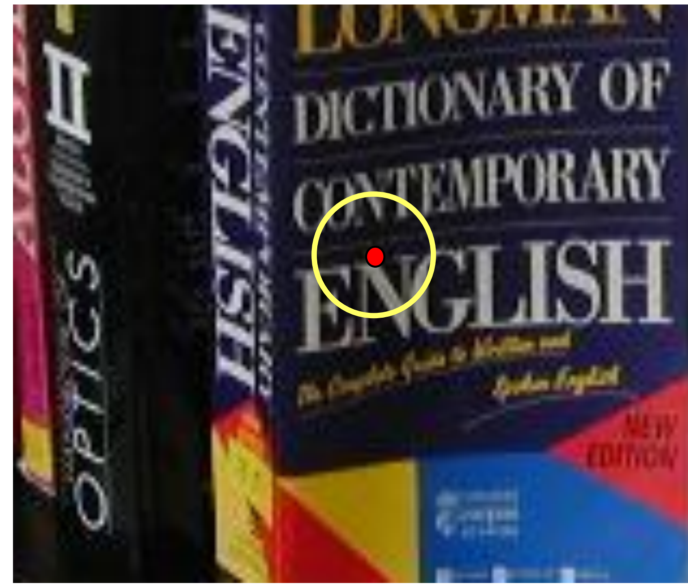
What we extract within the patch should be invariant to photometric and geometric changes

The need for variable patch shape



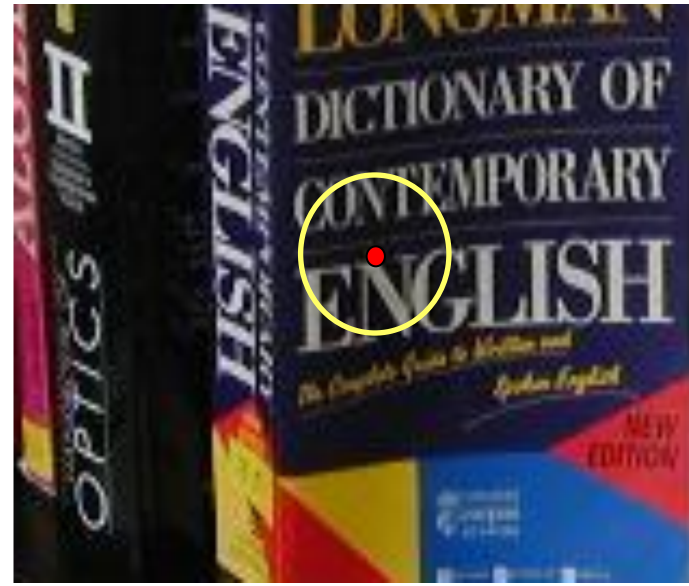
Taking the same square patch around corresponding interest points leads to a very different content of the patches… hence the matching will become hard.

The need for variable patch shape



Replacing the squares by identical circles does not really help much…

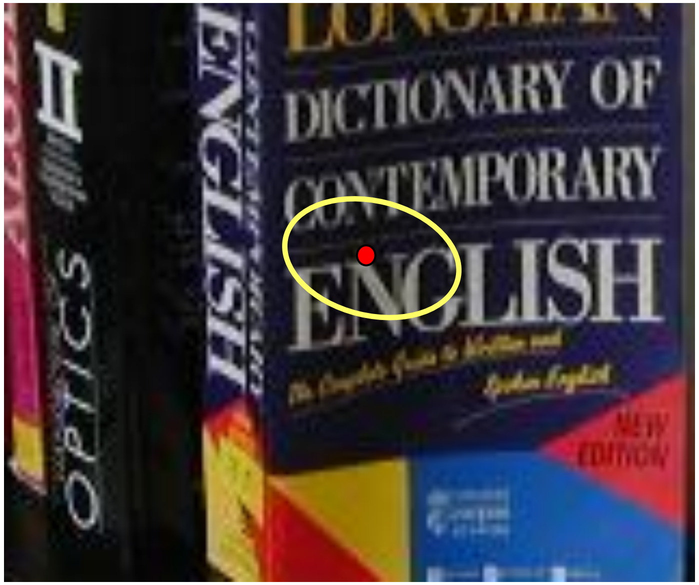
The need for variable patch shape



Allowing the diameters to differ helps somewhat, but contents still quite different (look at the top regions in the circles)

Computer Vision

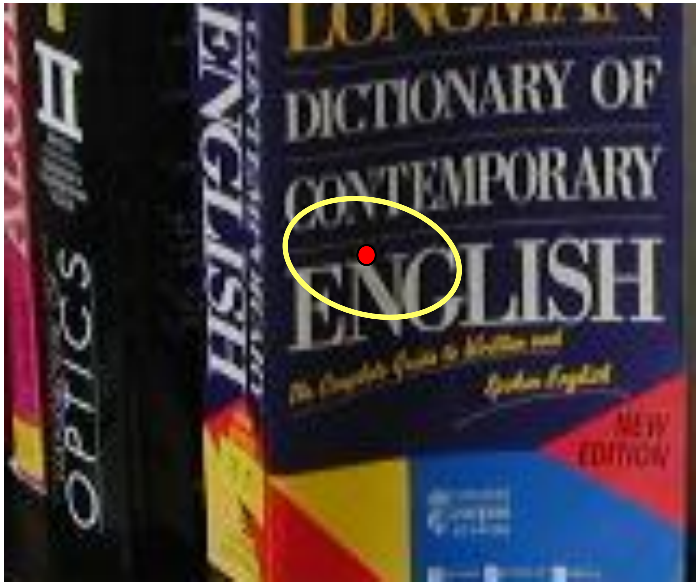
The need for variable patch shape



Using ellipses works much better: the circle on the left transforms into an ellipse under the affine transf. between the local view change

Computer Vision

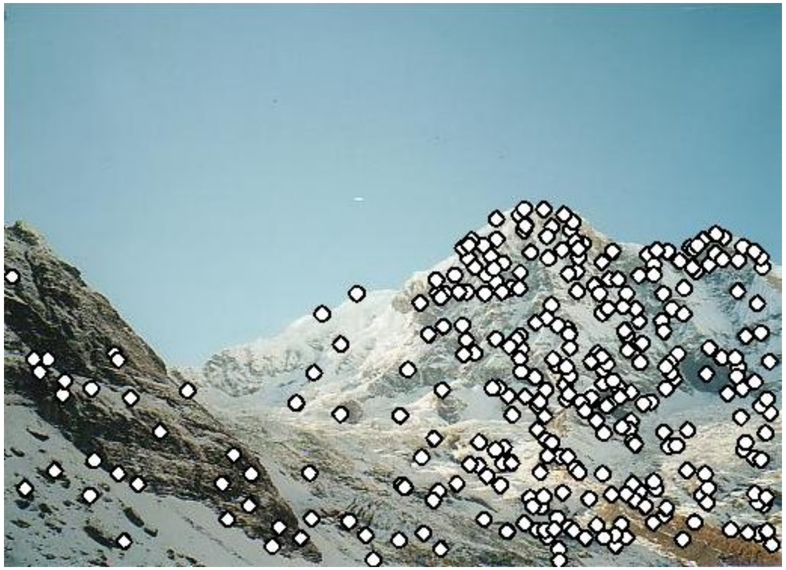
The need for variable patch shape



The important thing is to achieve such change in patch shape without having to compare the images, i.e. this should happen on the basis of information in only one image.

(Schmid and Mohr ‘97)

Example 1: Euclidean invariant features



Harris corner detector to identify corners as interest points.

Circular areas around each interest points.

For each interest point circular areas of different radii – to account for scale changes

Extract invariants under planar rotation from each area to form the descriptors.

Very successful model.

Example 1: Euclidean invariant features

Example (rotation) invariant:

$$G_x G_x + G_y G_y$$

Where G_x and G_y represent horizontal and vertical derivatives of intensity weighted by a Gaussian profile *(`Gaussian derivatives')*

2nd example invariant:

$$G_{xx} + G_{yy}$$

Where G_{xx} and G_{yy} represent 2nd order Gaussian derivatives

(Compute features for circles at different scales, i.e. take scale into account explicitly)

Question

- Is gradient rotationally invariant?

- Show that gradient magnitude is rotationally invariant.

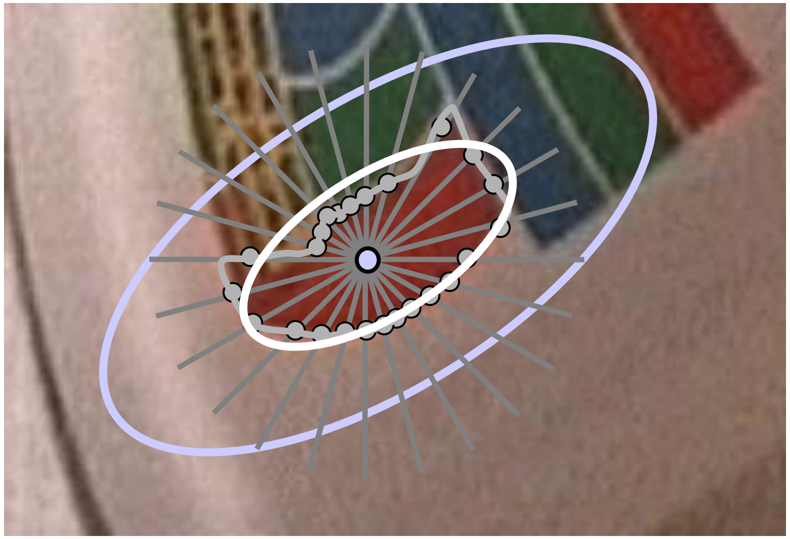
$$G_x G_x + G_y G_y$$

Computer Vision

Example 2: intensity extrema + affine moments

1. Search intensity extrema
2. Observe intensity profile along rays
3. Search maximum of invariant function f(t) along each ray
4. Connect local maxima
5. Fit ellipse
6. Double ellipse size
7. Describe elliptical patch with moment invariants

$$f(t) = \frac{abs(I_0 - I)}{\max\left(\frac{\int abs(I_0 - I)dt}{t}, d\right)}$$



Example 2: intensity extrema + affine moments

Geometric/photometric moment invariants based on generalised colour moments:

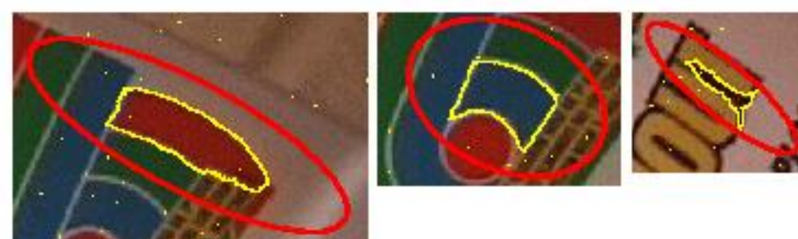
$$M_{p,q}^{a,b,c} = \int x^p y^q \, r^a(x,y) \, g^b(x,y) \, b^c(x,y) \, dxdy$$

M_{pq}^{abc} *are not invariant themselves, need to be combined*

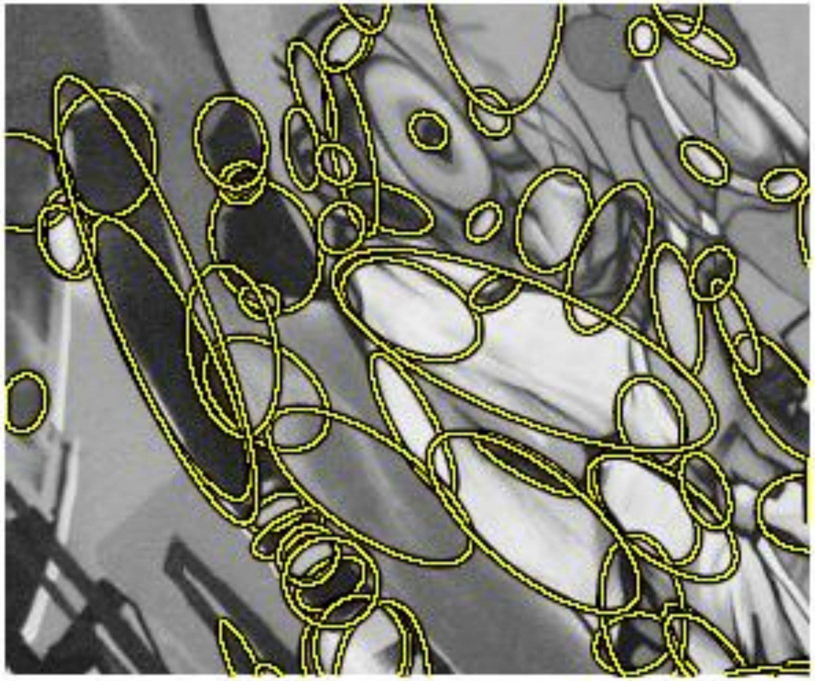
Example moment invariant from only 2 color bands:

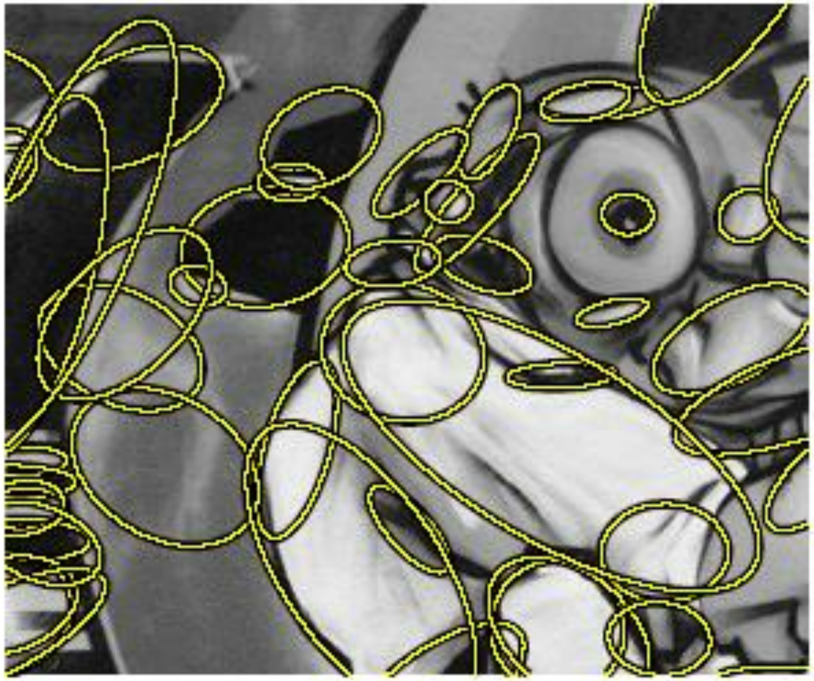
$$D_{02} = \frac{\left[M_{00}^{11} M_{00}^{00} - M_{00}^{10} M_{00}^{01}\right]^2}{\left[M_{00}^{20} M_{00}^{00} - (M_{00}^{10})^2\right] \left[M_{00}^{02} M_{00}^{00} - (M_{00}^{01})^2\right]}$$

Example 2: intensity extrema + affine moments



Example 2: intensity extrema + affine moments



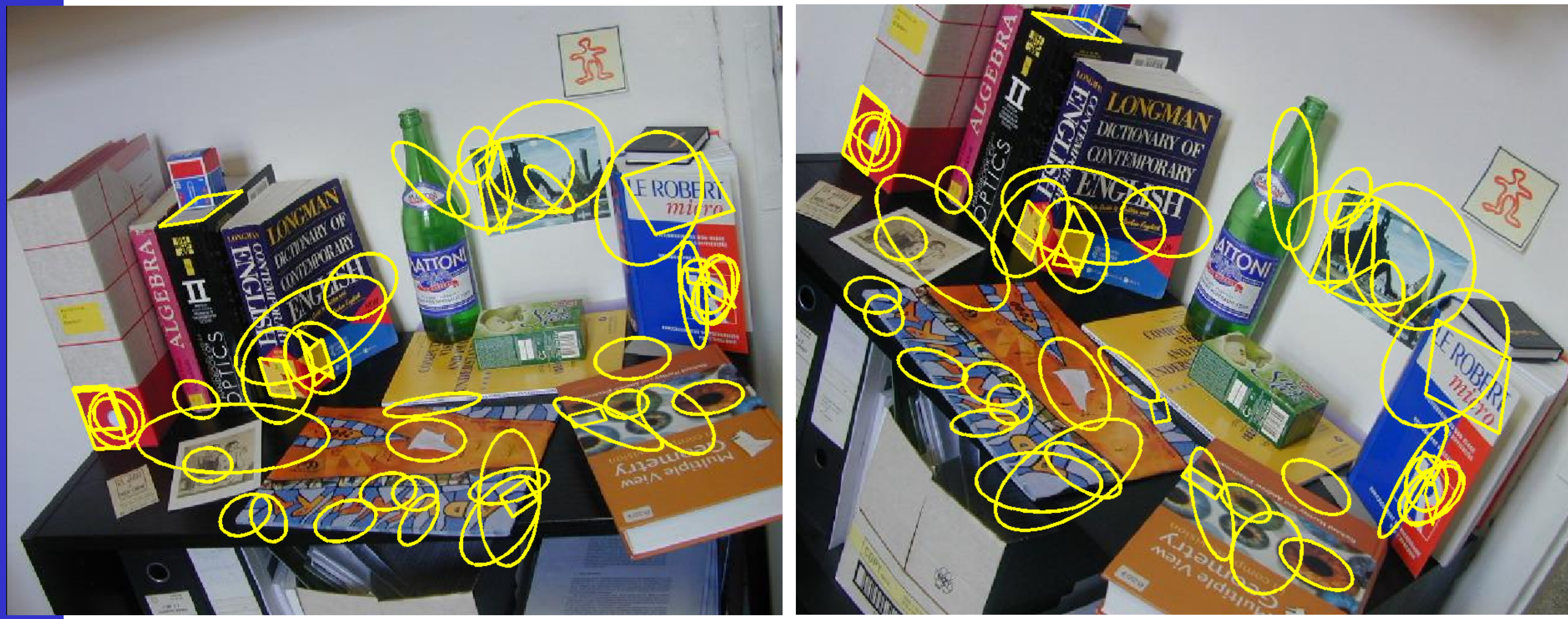


Remark

In practice different types of interest points are often combined

Wide baseline stereo matching example.

based on ex.1 and ex.2 interest points



PART 3: Advanced examples

MSER interest regions

MSER = Maximally Stable Extremal Regions

- Similar to the Intensity-Based Regions we just saw
- Came later, but is more often used
- Start with intensity extremum
- Then move intensity threshold away from its value and watch the super/sub-threshold region grow
- Take regions at thresholds where the growth is slowest (happens when region is bounded by strong edges)

Computer Vision

MSER

MSER

MSER

Computer Vision

MSER

MSER

Computer Vision

MSER

Computer Vision

MSER

Computer Vision

MSER

Computer Vision

MSER

MSER

MSER

MSER

MSER

Computer Vision

MSER

MSER

MSER

MSER

- ***Extremal region***: **region such that**

$$\forall p \in Q, \forall q \in \delta Q : \begin{matrix} I(p) > I(q) \\ I(p) < I(q) \end{matrix}$$

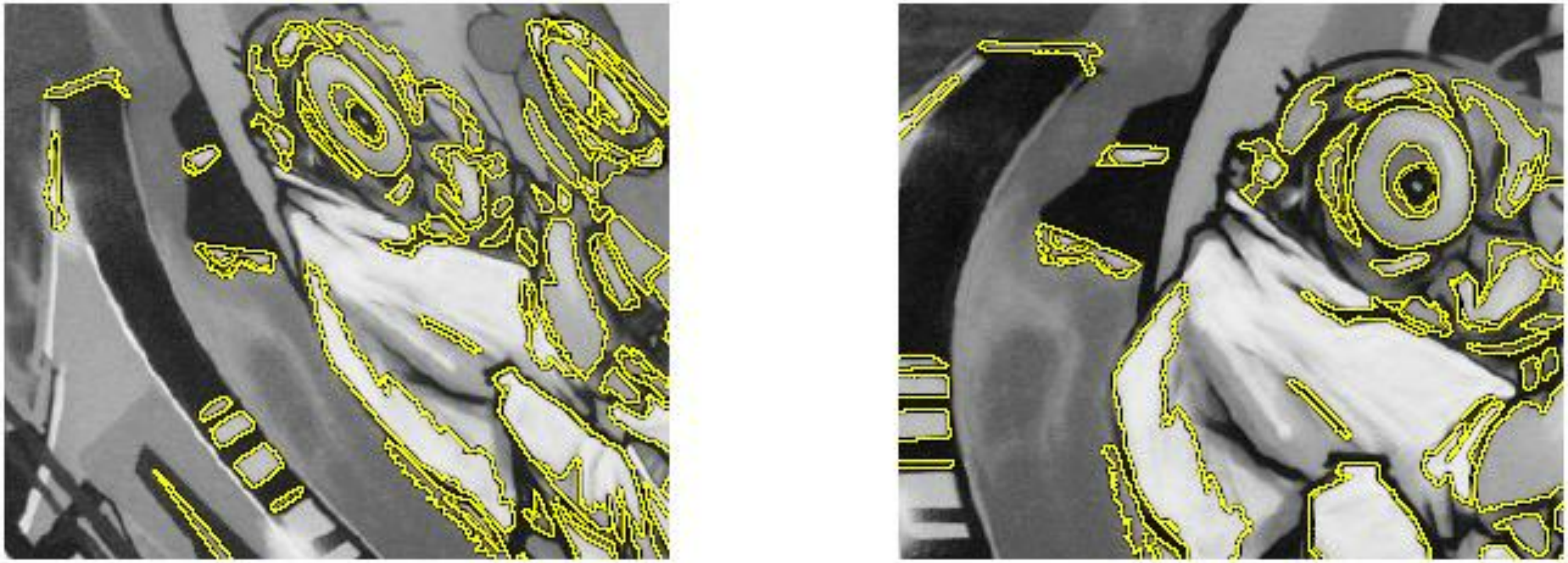
- **Order regions, following increasing or decreasing threshold**

$$Q_1 \subset \ldots \subset Q_i \subset Q_{i+1} \subset \ldots Q_n$$

- ***Maximally Stable Extremal Region***: **local minimum of**

$$q(i) = |Q_{i+\Delta} \setminus Q_{i-\Delta}| / Q_i$$

MSER



(a) MSER

SIFT

SIFT = Scale-Invariant Feature Transform

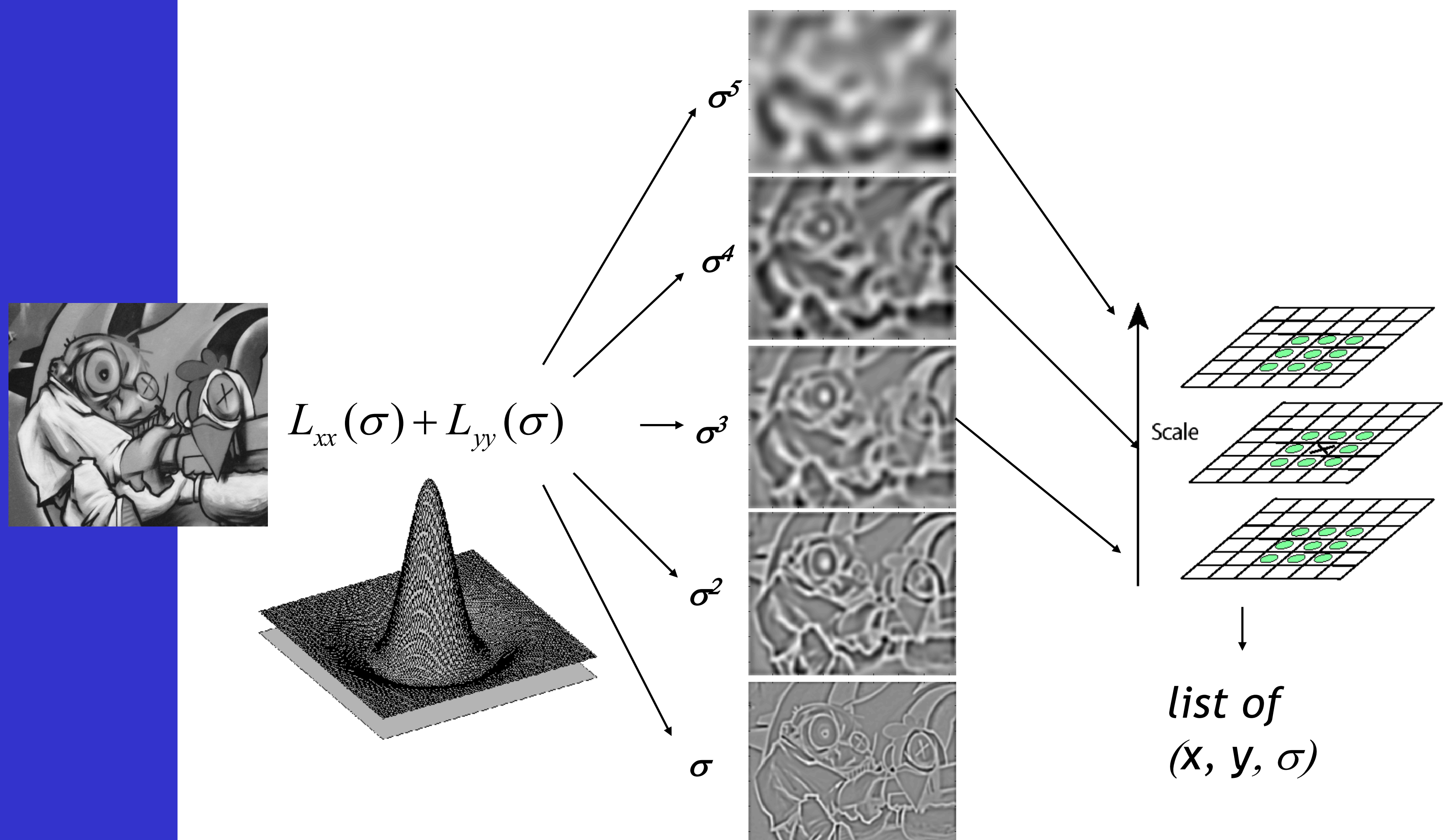
SIFT, developed by David Lowe (Un. British Columbia, Canada), is a carefully crafted interest point detector + descriptor, based on intensity gradients (cf. our comment on photometric invariance) and invariants under similarities, not affine like so far

Our summary is a simplified account!

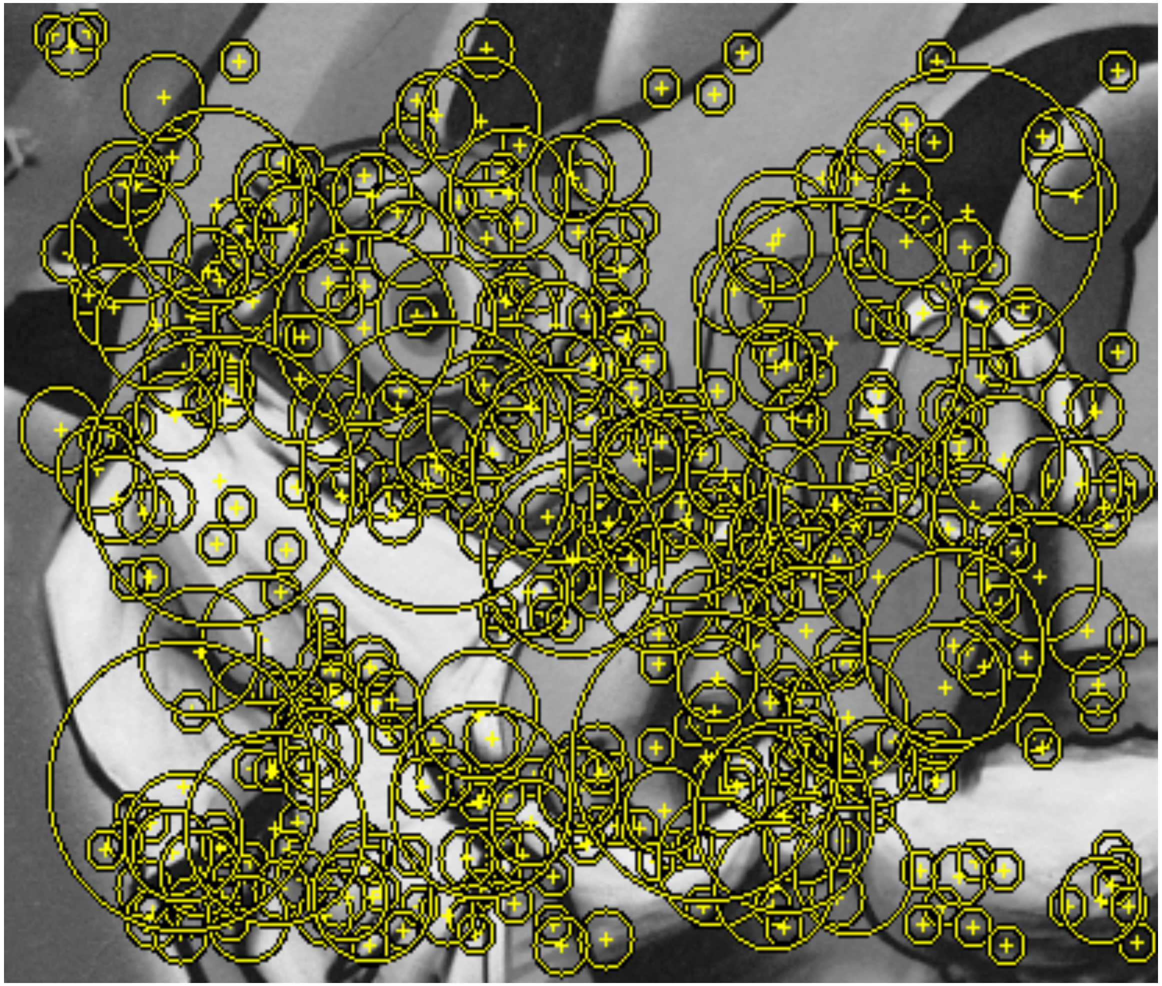
Computer Vision

SIFT

Descriptor is based on blob detection, at several scales, i.e. local extrema of the Laplacian-of-Gaussian, or LoG



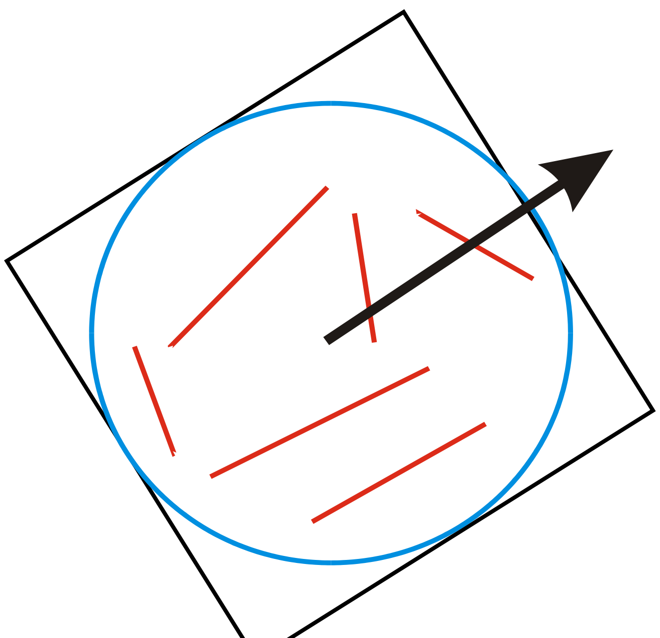
SIFT

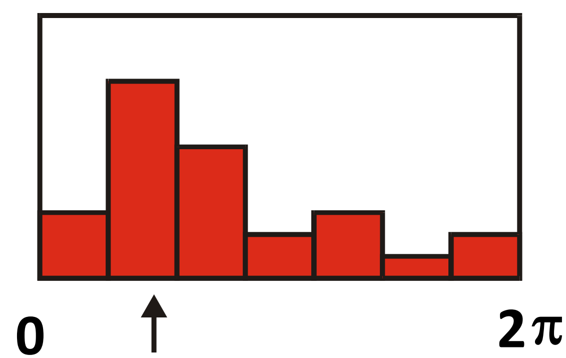


SIFT

Dominant orientation selection

- Compute image gradients
- Build orientation histogram
- Find maximum

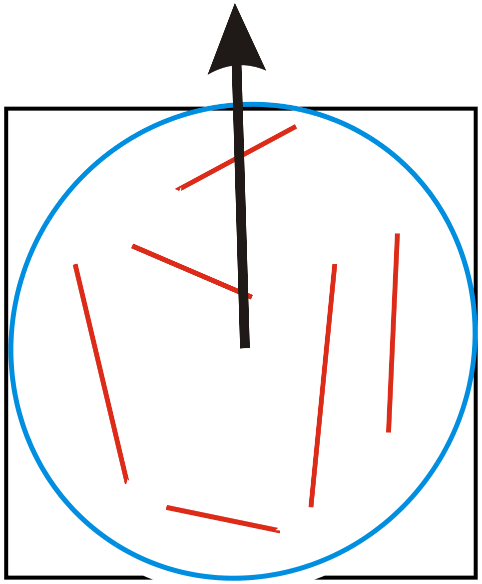


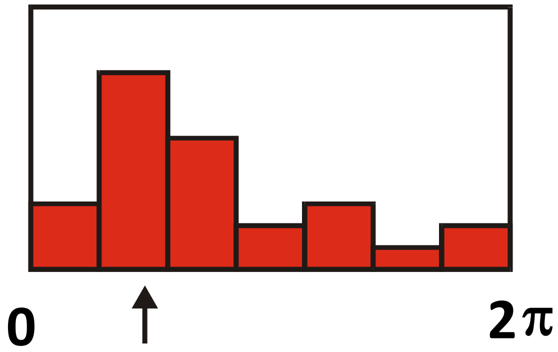


SIFT

Dominant orientation selection

- Compute image gradients
- Build orientation histogram
- Find maximum





SIFT

- Image gradients are sampled over a grid
- Create array of orientation histograms within blocks
- 8 orientations x 4x4 histogram array = 128 dimensions
- Apply weighting with a Gaussian located at the center
- Normalized to unit vector

(Fig. shows 2x2, actually 4x4)

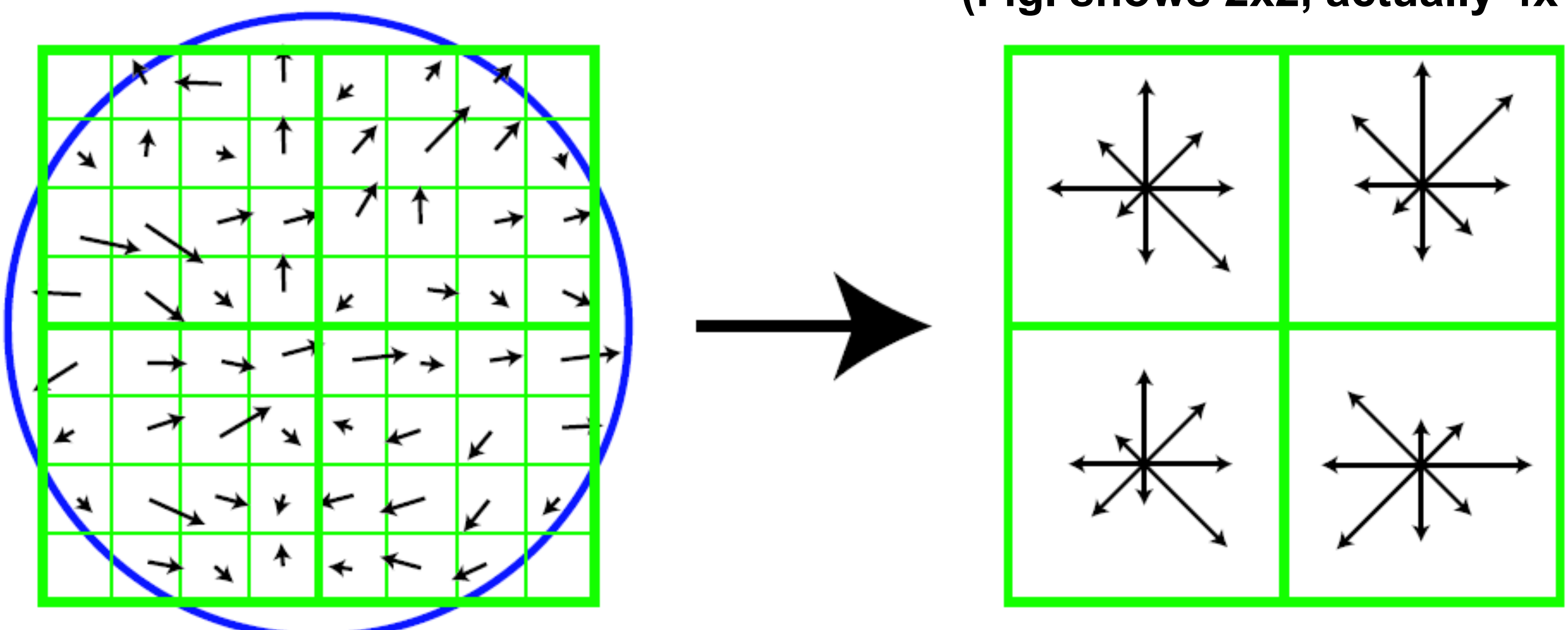
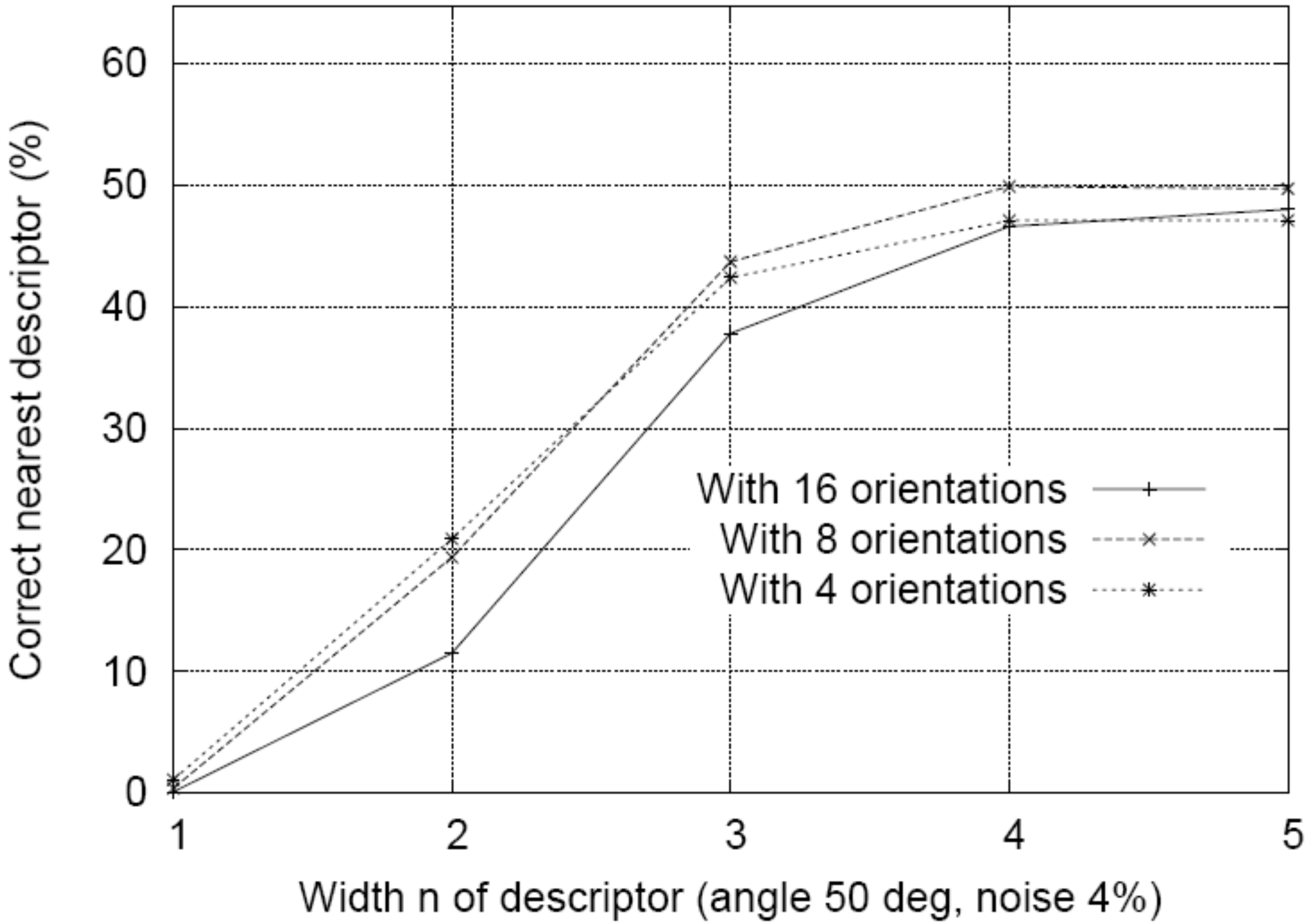


Image gradients

Keypoint descriptor

SIFT

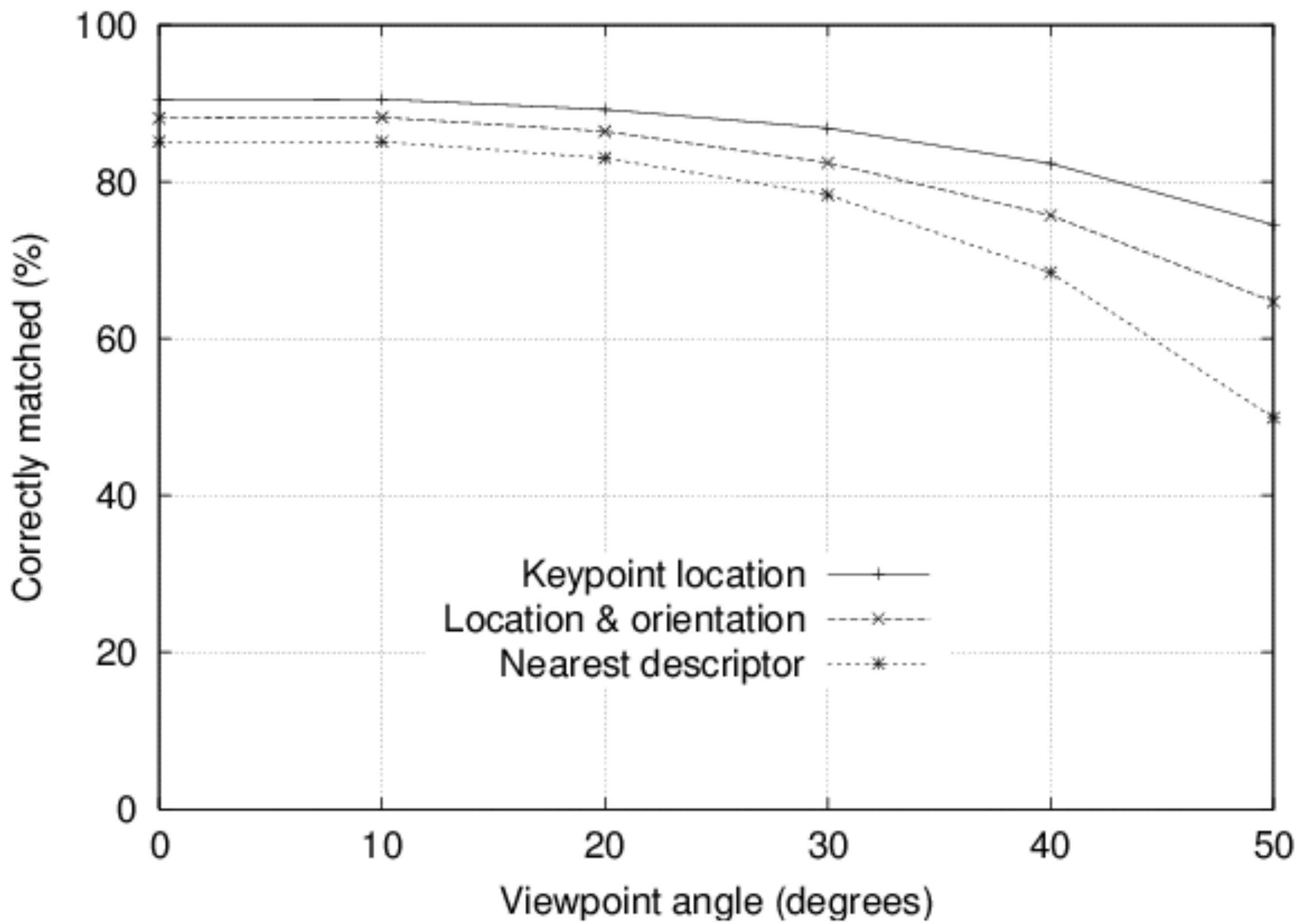
Carefully crafted… e.g. why 4 x 4 x 8 ?



Computer Vision

SIFT

Sensitivity to affine changes… quite good !!!



notes on matching

- Interest points are matched on the basis of their descriptors
- E.g. nearest neighbour, based on some distance like Euclidean or Mahalanobis; good to compare against 2nd nearest neighbour: OK if difference is big; or fuzzy matching w. multiple neighbours
- Speed-ups by using lower-dim. descriptor space (PCA) or through some coarse-to-fine scheme (very fast schemes exist to date!)
- Matching of individual points typically followed by some consistency check, e.g. epipolar geometry, homograpy, or topological