Image features

Francesco Isgrò

October 17, 2023

What are image features?

- a global property of an image
  - average grey level
  - area
- a part of the image with some special properties
  - a circle
  - a line
  - a textured region
  - a border

Definition

Image features are local, meaningful, detectable parts of the image

- Meaningful: the feature is associated to interesting scene elements via the image formation process.
- Detectable: a location algorithm must exist, otherwise the feature is of no use.

Meaningful feature

- sharp intensity variations, created by object contours
- regions with uniform grey levels, images of planar surfaces

Meaningful feature

- sharp intensity variations, created by object contours
- regions with uniform grey levels, images of planar surfaces

Keep in mind that...

feature may or may not correspond to parts of the scene.

Detectable

- different features are associated to different detection algorithms
- output is a feature descriptor: position plus other essential properties
- Line feature descriptor: central point, length, orientation

What are image features for?

- Feature extraction is an intermediate step
- For instance line extraction can be for:
  - navigate a robot in a corridor
  - decide whether an image contains a certain object
  - perform camera calibration

It does not make much sense to pursue *perfect* feature extraction per se. The adequacy of a feature detection algorithm should be ultimately assessed in the context of a complete system.
General performance criteria can and should be applied to test feature extraction modules independently.

Edge detection

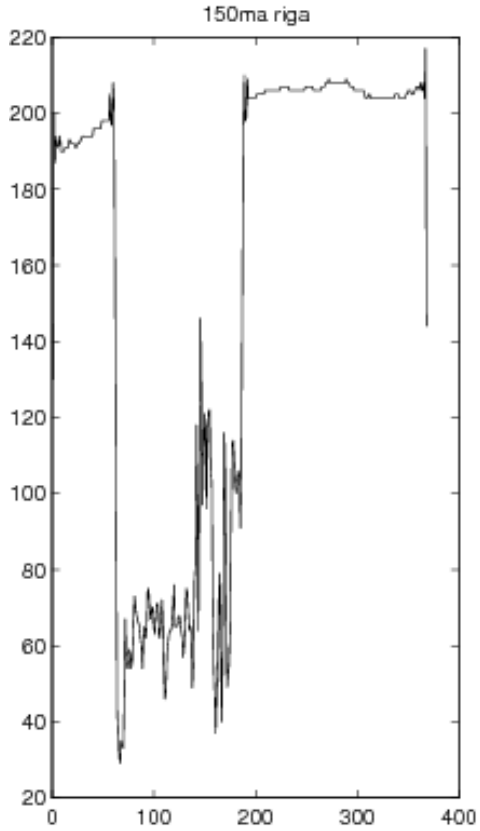
Definition of edges

Edge points are pixels at or around which the image values undergo a sharp variation.
The term edge is also used to refer to connected chains of edge points: contour fragments.

Edge detection

Definition of edges

Edge points are pixels at or around which the image values undergo a sharp variation.
The term edge is also used to refer to connected chains of edge points: contour fragments.



- Edges are significant local changes in the image. They are points in the image where there is a high variation in the intensity values.
- They typically occur on the boundary between two different regions in the image.
- They do not necessarily correspond to *real* contours, but could be caused by shadows, reflections, ...
- How do we study these variations?

Image gradient

A measure of this variation is the *gradient*. Given a function $f(x, y)$, its gradient is defined as

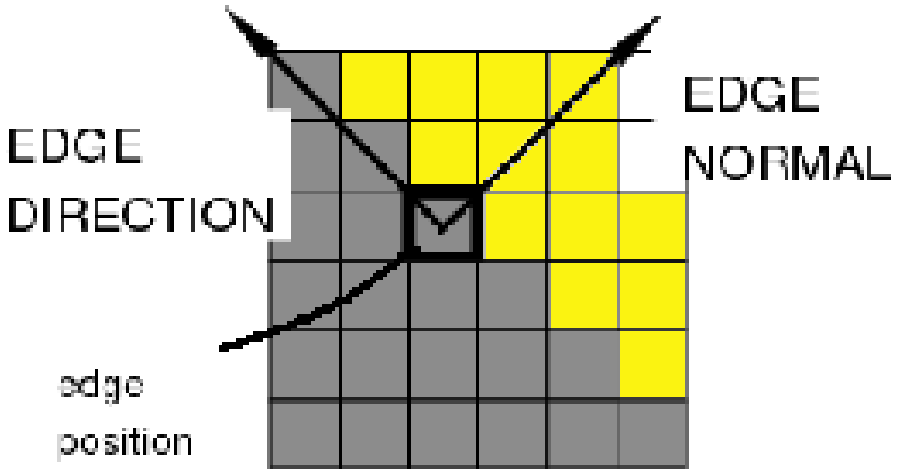
$$\nabla(f(x,y)) = \begin{bmatrix} J_x \\ J_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- gradient magnitude: $\sqrt{J_x^2 + J_y^2}$
- gradient direction: $\arctan \frac{J_y}{J_x}$

Ix

Iy

Essential edge descriptor



- Edge normal
- Edge direction
- Edge position
- Edge strength

Roberts cross edge detection

function RobertsEdgeDetect(I, τ)
Apply noise smoothing (e.g. Gaussian filter)
obtaining the image I_s
Filter I_s with the masks

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

obtaining the images I_x and I_y
Estimate the gradient magnitude $G(i,j) = \sqrt{\mathsf{I}_x(i,j)^2 + \mathsf{I}_y(i,j)^2}$
Make as edges all pixels such that $G(i,j) \geq \tau$

Original image

$G(i, j)$ image

Original image

$G(i,j)$ image $\tau = 80$

- Operator very fast since the kernel is small
- Output image presents spurious bright dots
- Operator is susceptible to noise
- Only the strongest edges have been detected with any reliability
- Output values from can overflow the maximum allowed pixel value: a solution is to set overflowing pixels to the maximum value.

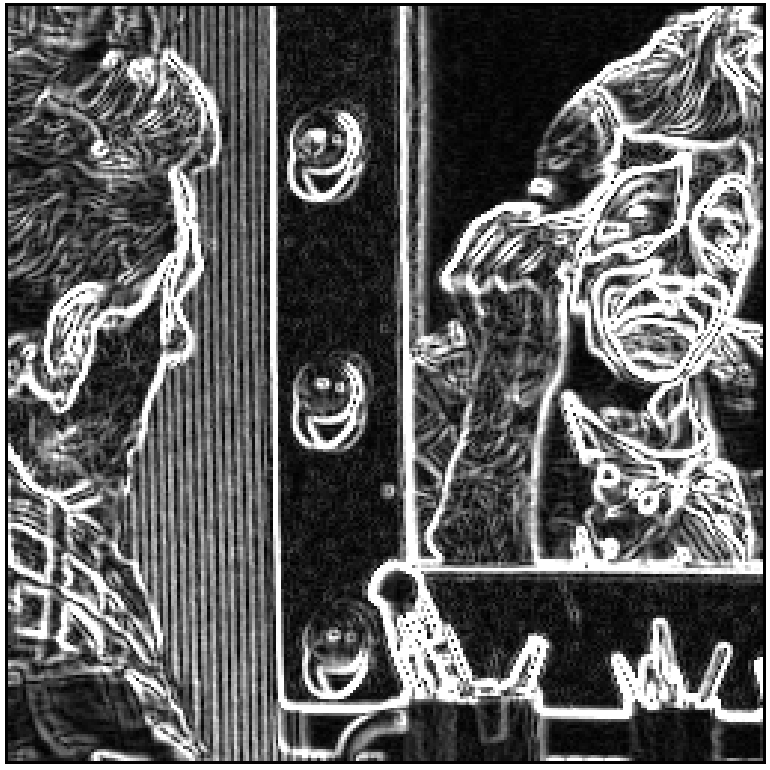
Sobel edge detection

function SobelEdgeDetect(I, τ)
Apply noise smoothing (e.g. Gaussian filter)
obtaining the image I_s
Filter I_s with the masks

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

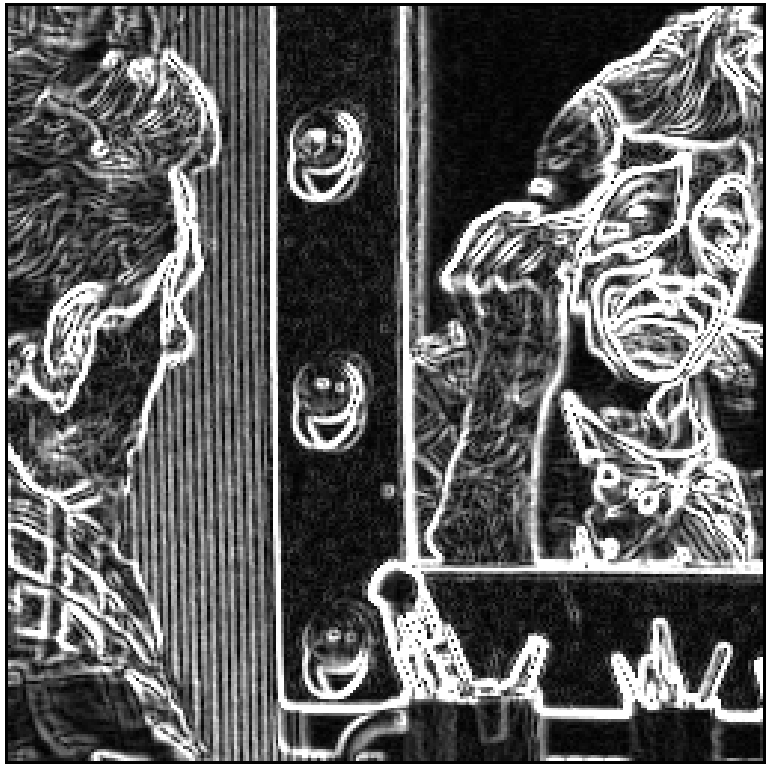
obtaining the images I_x and I_y
Estimate the gradient magnitude $G(i,j) = \sqrt{I_x(i,j)^2 + I_y(i,j)^2}$
Make as edges all pixels such that $G(i,j) \geq \tau$

Original image



$G(i, j)$ image

$G(i,j)$ Roberts



$G(i,j)$ Sobel

With respect to Roberts Cross operator

- Spurious noise is less intense
- Lines are thicker
  - Perform thinning: make position more precise

Edge detection

A typical algorithm for edge detection contains the following steps:

- Noise smoothing: when no information use a Gaussian filter
- Edge enhancement: design a filter responding to edges, so that edges are local maxima
- Edge localisation: Decide which local maxima are edges
  - thresholding
  - thinning

Canny's edge detector

- It is the most used edge detector.
- It is optimal in a precise mathematical sense
- It formulates a quantitative performance criteria
- It formalises desirable properties for the detector

Criteria for optimal edge detection

Good detection The optimal detector must minimise the probability of false positives, and that of missing real edges

Good localisation The edges detected must be as close as possible to the true edges

Single response constraint The detector must return one point for each true edge point

Edge detection and linear filters

A large linear filter

- improves detection
- worsen localisation

Localisation-detection trade-off

We can reach an optimal compromise between the localisation and detection criteria by adjusting the filter's scale, but we cannot improve both criteria simultaneously.

Canny's detector algorithm

1. `Edge enhancement`
2. `Non-maxima suppression`
3. `Hysteresis thresholding`

Edge enhancement

1. Apply Gaussian smoothing to image I, obtaining $\mathrm{J} = \mathrm{I} \star \mathrm{G}$
2. For each pixel (i, j)
   - Estimate the gradient components J_x and J_y
   - Compute the edge strength

$$\mathrm{E}_s(i,j) = \sqrt{J_x^2(i,j) + J_y^2(i,j)}$$

   - Estimate the orientation of the edge normal

$$\mathrm{E}_o(i,j) = \arctan \frac{J_y}{J_x}$$

Non-maxima suppression

1. For each (i, j) and for a given set D of directions:
   - find the direction $\hat{d}$ in D that best approximates $\mathrm{E}_o(i, j)$;
   - if $\mathrm{E}_s(i, j)$ is smaller than its two neighbours along the $\hat{d}$ direction then set $\mathrm{E}_s(i, j)$ to zero
2. a typical set D includes $\{0°, 45°, 90°, 135°\}$

Hysteresis thresholding

Given two threshold values $\tau_1 > \tau_2$, scan E_s and

1. if $\mathrm{E}_s(i, j) > \tau_1$ then
2. follow the connected local maxima, in both directions perpendicular to the edge normal, as long as $\mathrm{E}_s(h, k) > \tau_2$; mark the visited points and save a list of the locations of all points visited.

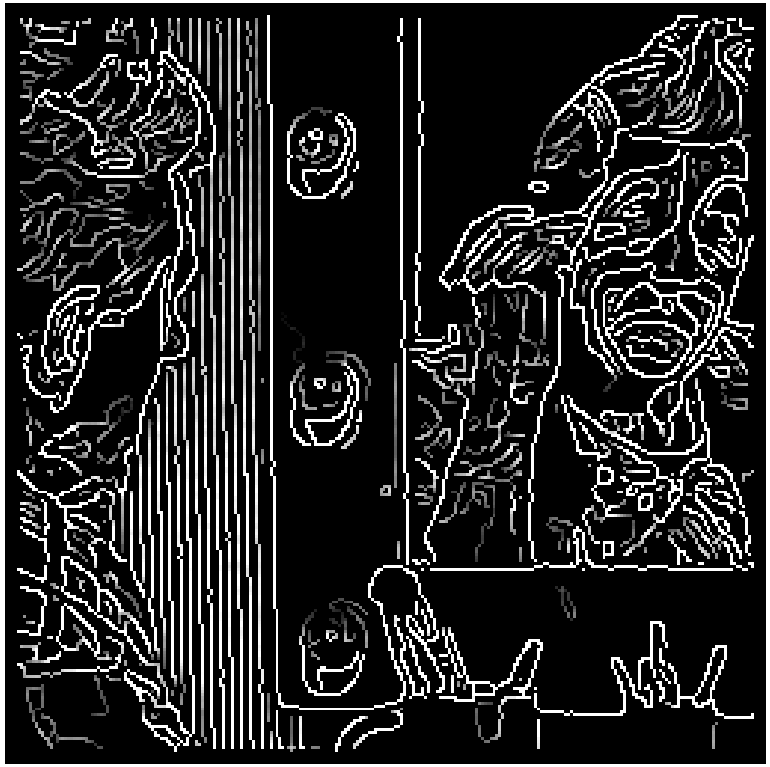
Original image

$\sigma = 1$, $\tau_1 = 255$, $\tau_2 = 1$

Original image

$\sigma = 1$, $\tau_1 = 255$, $\tau_2 = 220$

Original image



$\sigma = 1$, $\tau_1 = 128$, $\tau_2 = 1$

Original image

$\sigma = 2$, $\tau_1 = 128$, $\tau_2 = 1$

Sub-pixel precision

- All the edge detectors described identify the edge pixel
- The centre could be anywhere within the pixel
- In some applications half a pixel may correspond to high error
- May be important to locate edge at sub-pixel precision
- The easiest way is to interpolate a parabola in the enhanced image

For each edge pixel p in the enhanced image

1. Consider the edge normal θ_p
2. Identify the two neighbours pixels, p_{-1} and p_{+1} along θ_p
3. Interpolate a parabola using the three values in the enhanced image
4. Find the vertex of the parabola

Evaluation of edge detectors

Specific edge detectors can be evaluated:

- theoretically
- experimentally, estimating performance indexes on synthetic images
  - probability of false detection:
  - probability of misses:
  - the average and RMS errors of edge positions

Evaluation of edge detectors

Specific edge detectors can be evaluated:

- theoretically
- experimentally, estimating performance indexes on synthetic images
  - probability of false detection: the number of spurious edges
  - probability of misses:
  - the average and RMS errors of edge positions

Evaluation of edge detectors

Specific edge detectors can be evaluated:

- theoretically
- experimentally, estimating performance indexes on synthetic images
  - probability of false detection: the number of spurious edges
  - probability of misses: number of true edges missed
  - the average and RMS errors of edge positions

Line and curve detection

- Images often contain man-made objects
- Contours of these objects are very regular
  - line
  - circle
  - ellipse

Line detection: template matching

Look for peaks of the convolution between the image and a set of masks M_θ tuned to detect the presence of lines of a particular width and at a particular orientation θ.
A point p is on a line of orientation θ if

- The response of M_θ is the highest among all masks
- The response of M_θ is higher than a given threshold

a)

-1	-1	-1
2	2	2
-1	-1	-1

b)

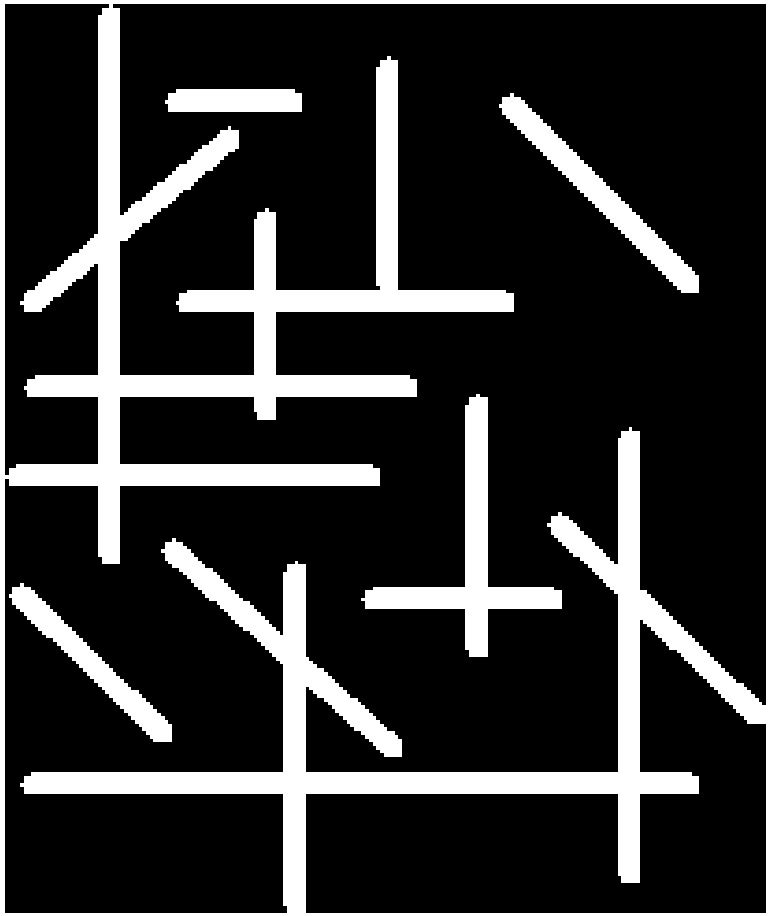
-1	2	-1
-1	2	-1
-1	2	-1

c)

-1	-1	2
-1	2	-1
2	-1	-1

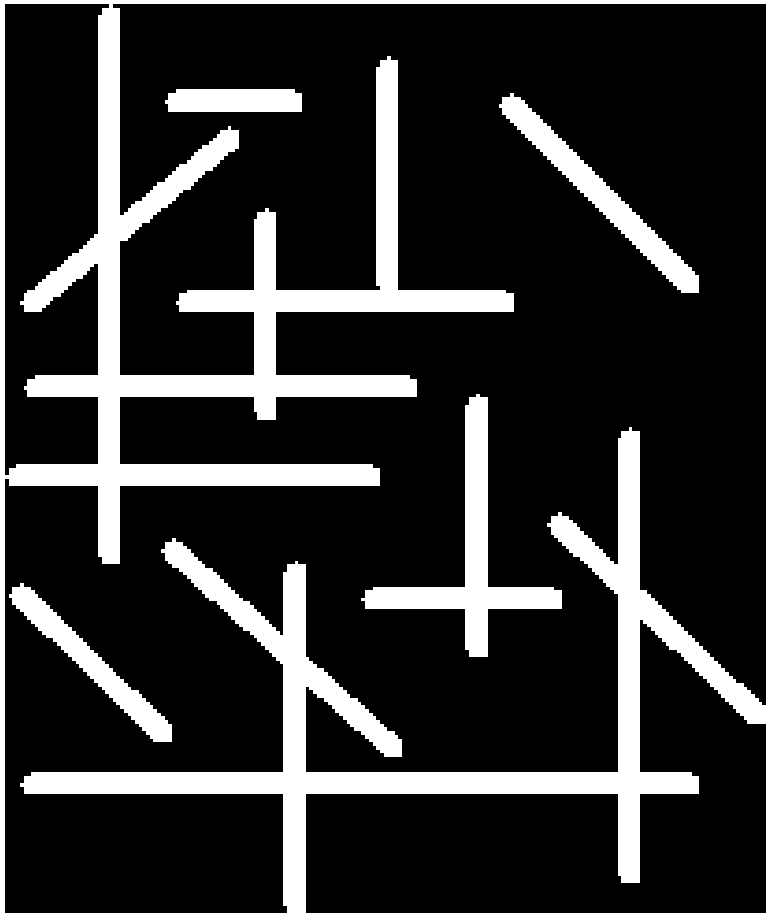
d)

2	-1	-1
-1	2	-1
-1	-1	2

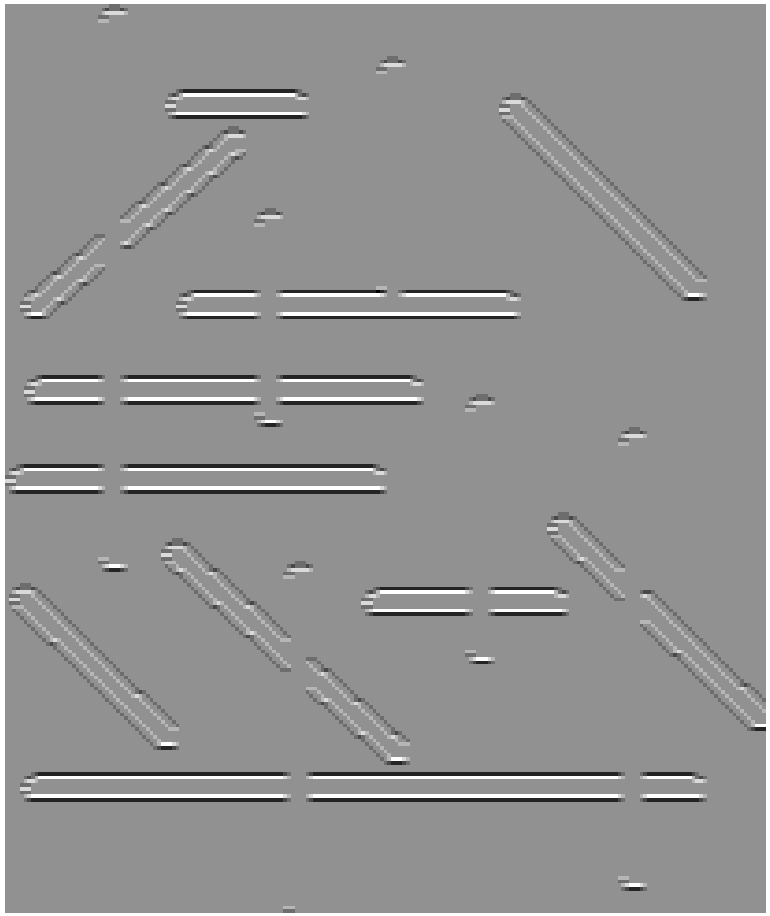


Input image

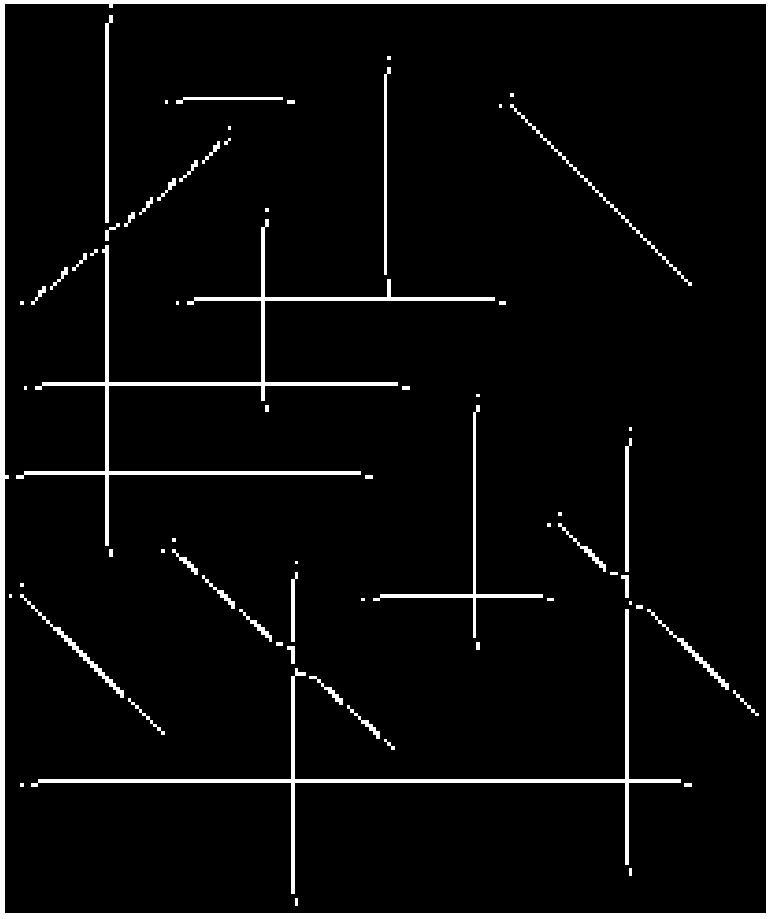
Filters response



Input image

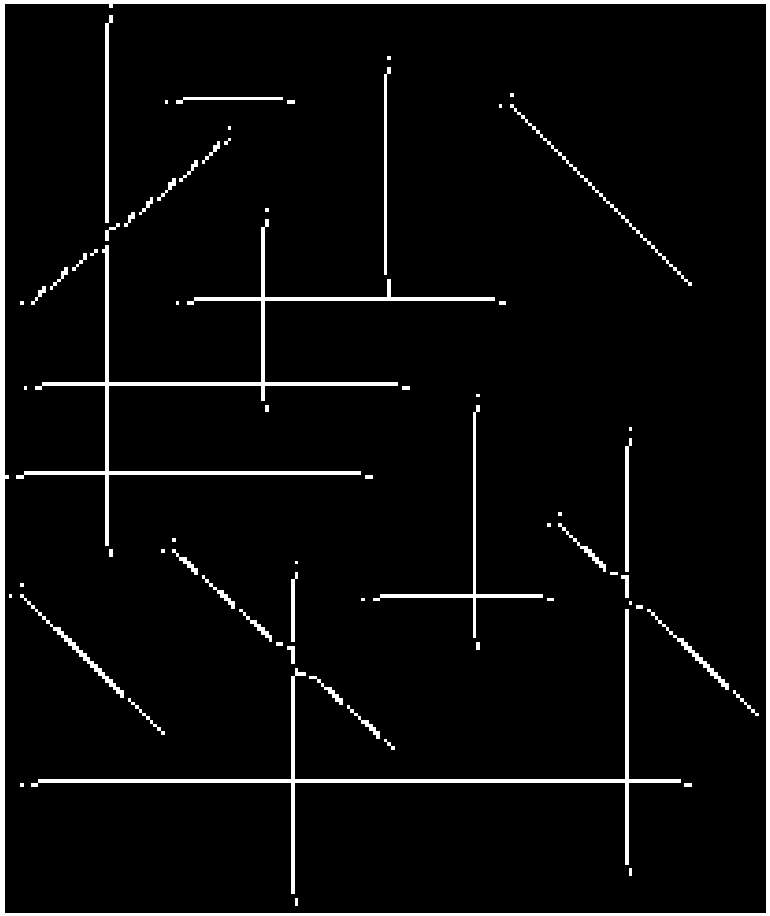


Filters response

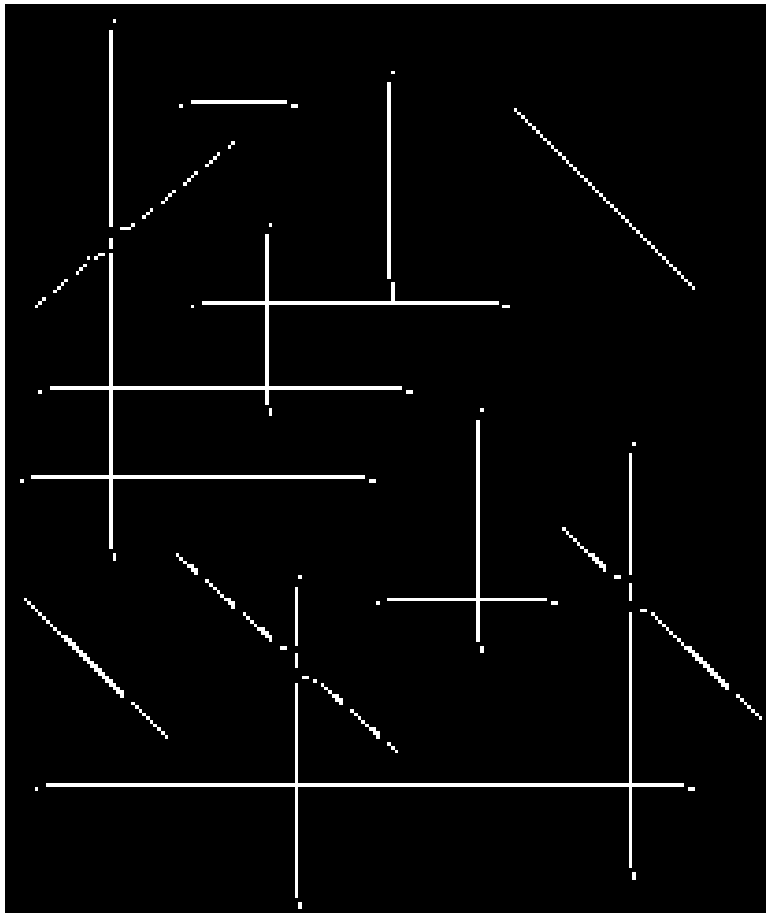


Skeletonised input image

Filters response

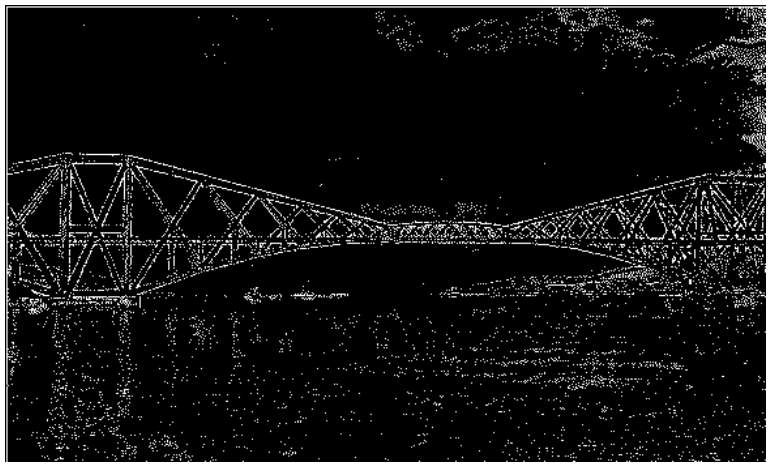


Skeletonised input image

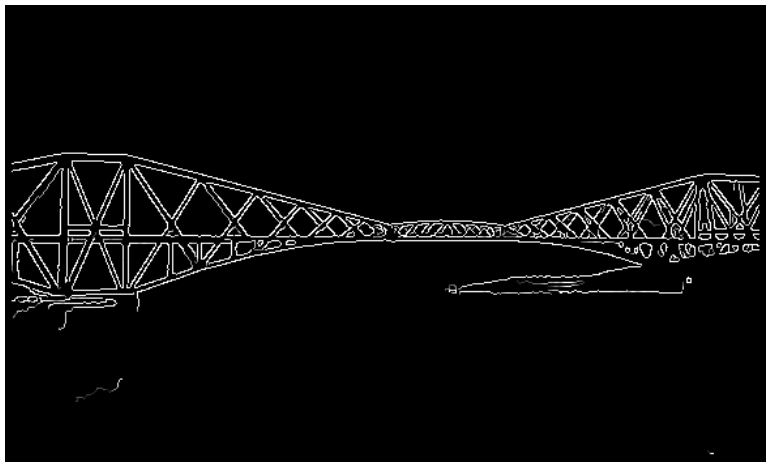


Filters response

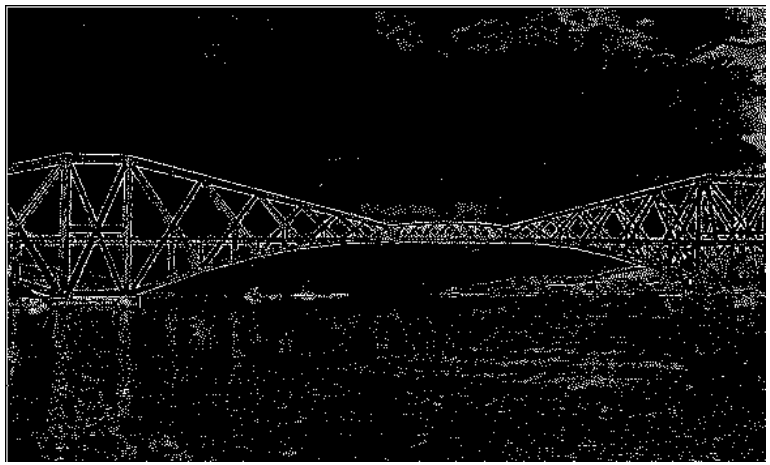
Input image



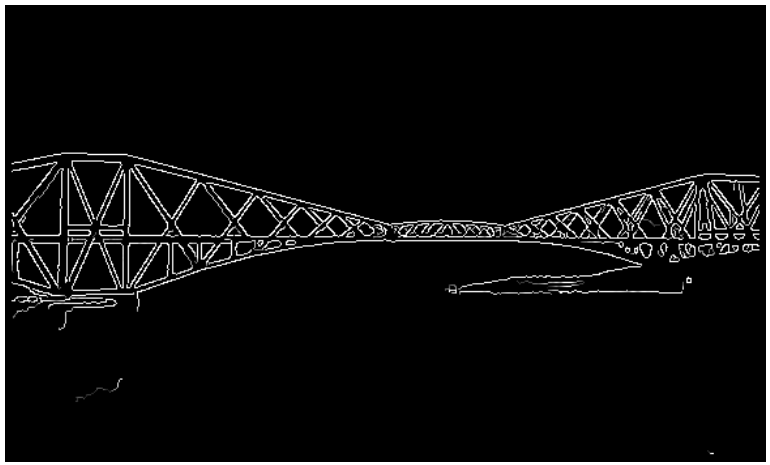
Outupt image



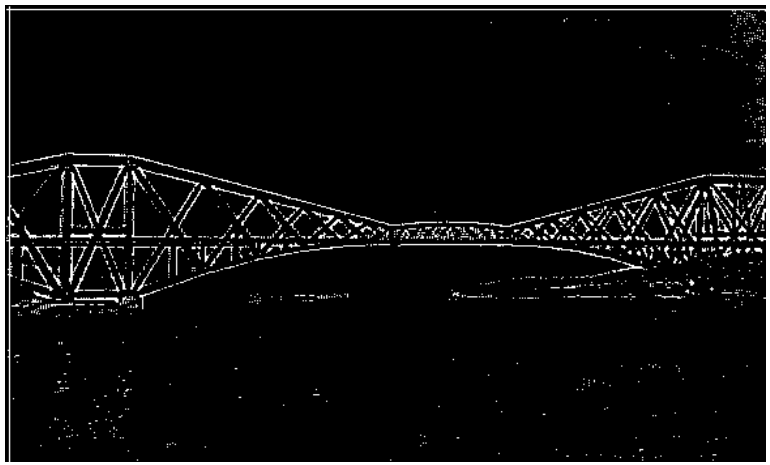
Output of Canny



Output image



Output of Canny



Gaussian Filter + Line detector

- accurate line detection
  - requires a large set of masks
  - the whole process is computationally expensive
- if lines wider than masks
  - return edges of lines
  - wrong results of two parallel lines
  - skeletonisation reduce accuracy

Line and curves detection

- Images often contain man-made objects
- Contours of these objects are very regular
  - line
  - circle
  - ellipse

Line and curves detection

- Images often contain man-made objects
- Contours of these objects are very regular
  - line
  - circle
  - ellipse

Problem statement

Given the output of an edge detector, find all the instances of a given curve or parts thereof.

The Hough transform

- It is a technique used to isolate features of a particular shape within an image.
- It requires that the desired features be specified in some parametric form

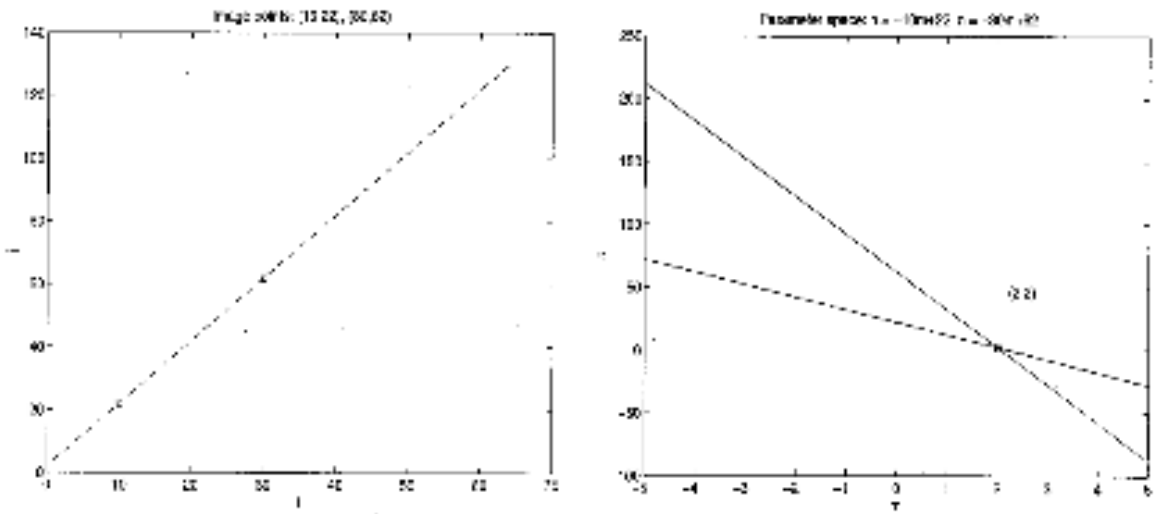
Map a difficult pattern detection problem into a simple peak detection problem in the space of the parameters of the curve.

The Hough transform for lines

1. Transform line detection into a line intersection problem
   - a line $y = mx + n$ is identified by (m, n)
   - a point $\mathrm{p} = (x, y)$ corresponds to the line $n = (-x)m + y$ in the parameter space
   - A line identified by $\mathrm{p}_1, \cdots, \mathrm{p}_n$ is identified by the intersection of n lines.
2. Transform line intersection into a peak detection problem

The Hough transform for lines

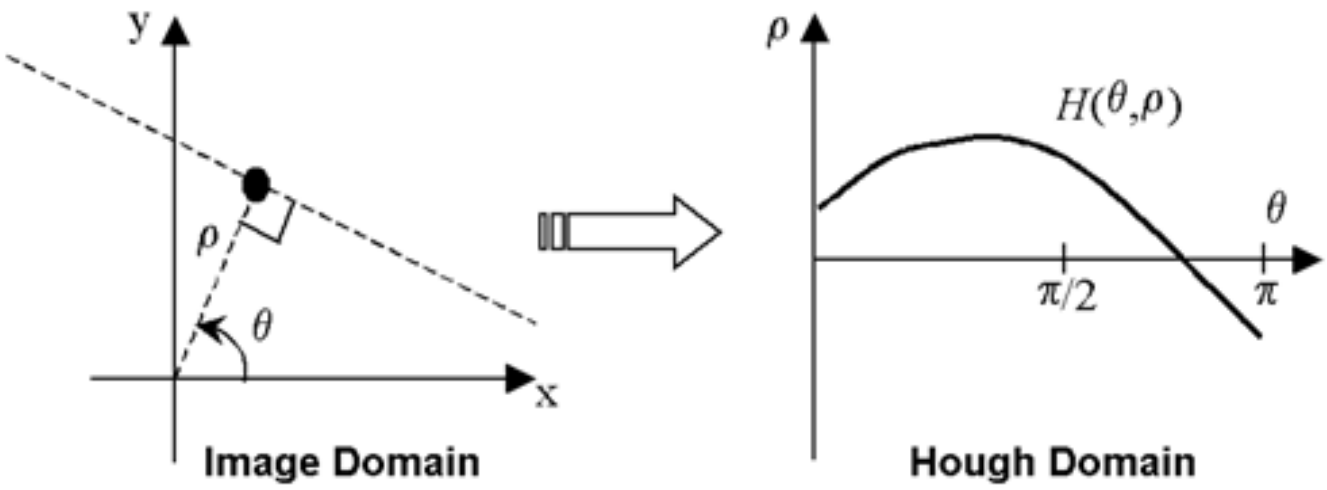
1. Transform line detection into a line intersection problem
   - a line $y = mx + n$ is identified by (m, n)
   - a point $\mathrm{p} = (x, y)$ corresponds to the line $n = (-x)m + y$ in the parameter space
   - A line identified by $\mathrm{p}_1, \cdots, \mathrm{p}_n$ is identified by the intersection of n lines.
2. Transform line intersection into a peak detection problem



The Hough transform for lines

1. Transform line detection into a line intersection problem
   - a line $y = mx + n$ is identified by (m, n)
   - a point $\mathrm{p} = (x, y)$ corresponds to the line $n = (-x)m + y$ in the parameter space
   - A line identified by $\mathrm{p}_1, \cdots, \mathrm{p}_n$ is identified by the intersection of n lines.
2. Transform line intersection into a peak detection problem
   - Divide the parameter space into a finite grid of cells
   - Associate a counter to each cell
   - Increment the counter for each line passing through the cell
   - Determine the peak

- Both m and n vary in $[-\infty, \infty]$
  - We cannot sample the whole parameters space
  - We cannot capture $x = k$
- We can use the polar representation
  - $\rho = x \cos\theta + y \sin\theta$
  - the intervals o variation for ρ and θ are finite
  - $\rho \varepsilon [0, \sqrt{M^2 + N^2}]$, where $M \times N$ is the image size
  - $\theta \varepsilon [0, \pi]$


y
ρ
θ
x
Image Domain
ρ
H(θ,ρ)
θ
π/2
π
Hough Domain

Nonlinear contours

- Not all edge points belong to lines
- Spread of random values throughout the parameters space
- Multitude of noisy peaks
- Problem: discriminate between noisy peaks and good peaks
  - Threshold the counters

Noise

- Pixelisation and limited accuracy in edge detection
  - not all lines in the parameters space (points of the same line) intersect at the same point
- Determine peak as local maxima
- Determine peak as weighted average in a neighbourhood of the local maxima

Algorithm

function $[I_l]$ = HoughLines$(I, \delta\rho, \delta\theta, \tau)$

$M_H = \sqrt{M^2 + N^2}/\delta\rho, \;\; N_H = \pi/\delta\theta$

A = zeros(M_H, N_H)

for each (i, j) such that $I(i, j) = 1$

  for h=0 .. N_H

    $\rho = round(i \cos\theta(h) + j \sin\theta(h))$

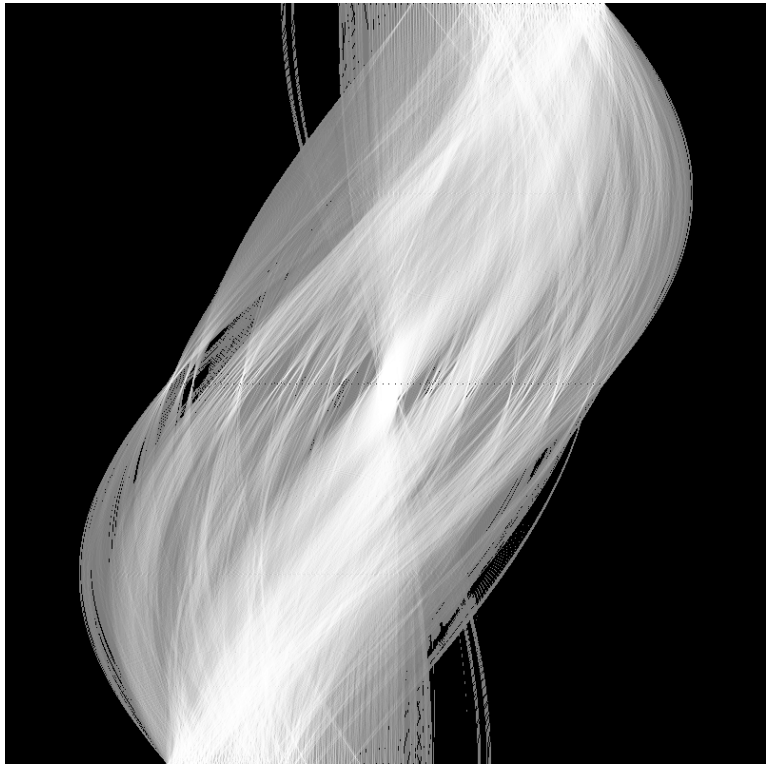
    A(ρ, h) += 1

Find all local maxima in A, such that A$(\rho, h) > \tau$

Input image

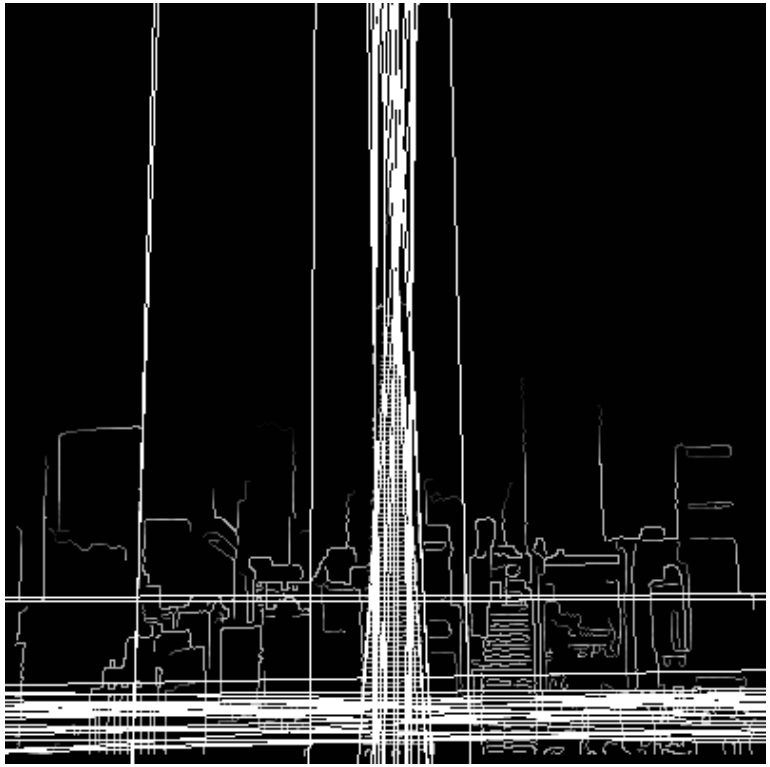
Output of Canny

Input image



Hough transform

Input image



Line detected

Hough transform for circles

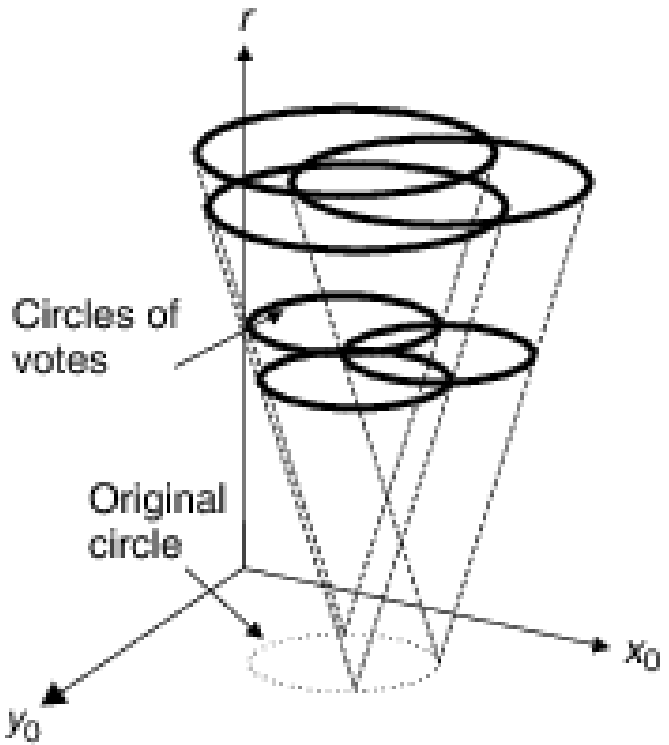
The equation of a circle is

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Hough transform for circles

The equation of a circle is

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$



(c) 3D accumulator space

The equation of a circle can be written in parametric form as

$$x = x_0 + r\cos\theta$$
$$y = y_0 + r\sin\theta$$

The equation of a circle can be written in parametric form as

$$x = x_0 + r\cos\theta$$
$$y = y_0 + r\sin\theta$$

therefore

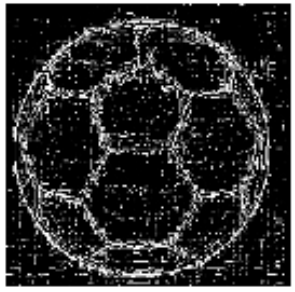
$$x_0 = x - r\cos\theta$$
$$y_0 = y - r\sin\theta$$

x_0 and y_0 depend on the radius r.

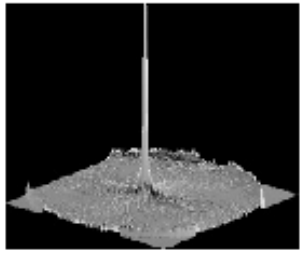
```
%Hough Transform for Circles

function HTCircle(inputimage,r)

%image size
[rows,columns]=size(inputimage);

%accumulator
acc=zeros(rows,columns);

%image
for x=1:columns
  for y=1:rows
    if(inputimage(y,x)==0)
      for ang=0:360
        t=(ang*pi)/180;
        x0=round(x-r*cos(t));
        y0=round(y-r*sin(t));
        if(x0<columns & x0>0 & y0<rows & y0>0)
          acc(y0,x0)=acc(y0,x0)+1;
        end
      end
    end
  end
end
```

(a) Circle

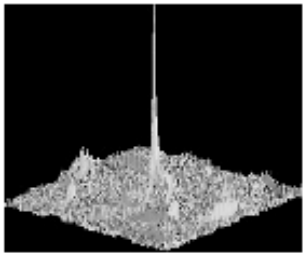
(b) Soccer ball edges



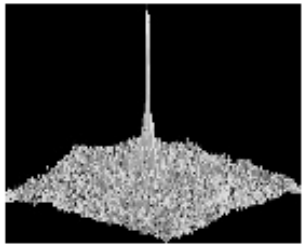
(c) Noisy soccer ball edges



(d) Accumulator for (a)



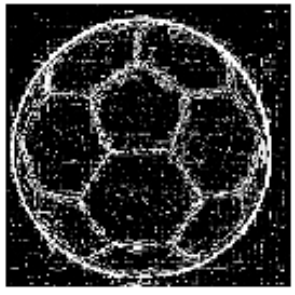
(e) Accumulator for (b)



(f) Accumulator for (c)

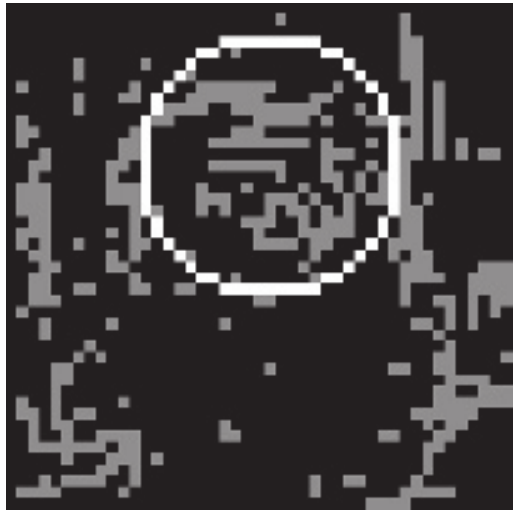
(g) Circle from (d)

(h) Circle from (e)



(i) Circle from (f)





The Hough transform for curves

- The algorithm can be generalised to generic curves
  - $y = f(x, \mathrm{a})$
  - $\mathrm{a} = [a_1, \cdots, a_T]^t$
- The parameters space grows exponentially with T
- Computation time can be too high
- Partial solution
  - Multi-resolution
  - Increase resolution only where needed, i.e. where there are local maxima

The Hough transform: key points

- The HT is a voting algorithm
- The counters in parameters space can be regarded as an histogram
- A counter of coordinates m indicates the relative likelihood of the hypothesis *a curve* m *exists in the image*
- It can cope with occlusions: points are processed independently
- It is robust to noise as outliers are unlikely to contribute to a single bin
- Major limitations:
  - search time increases with number of parameters
  - non-target shapes can produce spurious peaks (e.g., line detection can be disturbed by low-curvature circles).

Point features

Two main approaches

- Peaks of local curvature are feature points
  - Corners extraction
  - Enough when small changes between images
  - Start failing when strong affine transformation
- Using region or patch-based analysis
  - Looking for affine invariance
  - Can cope with largest changes between images

Definition of curvature

Curvature

Intuitively it is the of change in edge direction. Points where it changes rapidly are corners.

A curve in parametric form is ($U_x = [1,0]^t$, $U_y = [0,1]^t$)

$$v(t) = x(t)U_x + y(t)U_y$$

Changes in the position vector are given by the tangent vector

$$\frac{dv(t)}{dt} = \frac{dx(t)}{dt}U_x + \frac{dy(t)}{dt}U_y = \dot{x}(t)U_x + \dot{y}(t)U_y$$

A point moves with speed

$$|\dot{v}(t)| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}$$

in the direction

$$\phi(t) = \tan^{-1}\frac{\dot{y}(t)}{\dot{x}(t)}$$

Curvature

The curvature at a point $v(t)$ describes the change in the direction $\phi(t)$ with respect to changes in arc length s

$$\kappa(t) = \frac{d\phi(t)}{ds}$$

Curvature

The curvature at a point $v(t)$ describes the change in the direction $\phi(t)$ with respect to changes in arc length s

$$\kappa(t) = \frac{d\phi(t)}{ds}$$

$$\kappa(t) = \frac{d\phi(t)}{dt}\frac{dt}{ds}$$

Curvature

The curvature at a point $v(t)$ describes the change in the direction $\phi(t)$ with respect to changes in arc length s

$$\kappa(t) = \frac{d\phi(t)}{ds}$$

$$\kappa(t) = \frac{d\phi(t)}{dt}\frac{dt}{ds}$$

$$\frac{ds}{dt} = |\dot{v}(t)| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}$$

Curvature

The curvature at a point $v(t)$ describes the change in the direction $\phi(t)$ with respect to changes in arc length s

$$\kappa(t) = \frac{d\phi(t)}{ds}$$

$$\kappa(t) = \frac{d\phi(t)}{dt}\frac{dt}{ds}$$

$$\frac{ds}{dt} = |\dot{v}(t)| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}$$

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{[\dot{x}(t)^2 + \dot{y}(t)^2]^{3/2}}$$

For a straight line is $\ddot{x}(t) = \ddot{y}(t) = 0$ then

For a straight line is $\ddot{x}(t) = \ddot{y}(t) = 0$ then

$$\kappa(t) = 0$$

For a straight line is $\ddot{x}(t) = \ddot{y}(t) = 0$ then

$$\kappa(t) = 0$$

For a circle of radius r is

$$\begin{aligned} \dot{y}(t) &= r\cos(t) & \dot{x}(t) &= -r\sin(t) \\ \ddot{y}(t) &= -r\sin(t) & \ddot{x}(t) &= -r\cos(t) \end{aligned}$$

For a straight line is $\ddot{x}(t) = \ddot{y}(t) = 0$ then

$$\kappa(t) = 0$$

For a circle of radius r is

$$\begin{aligned} \dot{y}(t) &= r\cos(t) & \dot{x}(t) &= -r\sin(t) \\ \ddot{y}(t) &= -r\sin(t) & \ddot{x}(t) &= -r\cos(t) \end{aligned}$$

$$\kappa(t) = \frac{1}{r}$$

In digital images curves are discrete data. There three ways to compute $\kappa(t)$

- Directly compute the difference between angular directions of consecutive edge pixels
- Derive a measure of curvature changes in image intensity
- Obtain a measure of curvature by correlation
  - Moravec's detector
  - Harris detector

Computing differences in edge direction

- Early approach (70s)
- It requires precomputed edges
- It uses the gradient direction $\theta(t)$

For each edge pixels is

$$\kappa(t) = \theta(t+1) - \theta(t-1)$$

The values of $\theta(t)$ are

- defined at discrete points
- might not vary smoothly

The values of $\theta(t)$ are

- defined at discrete points
- might not vary smoothly

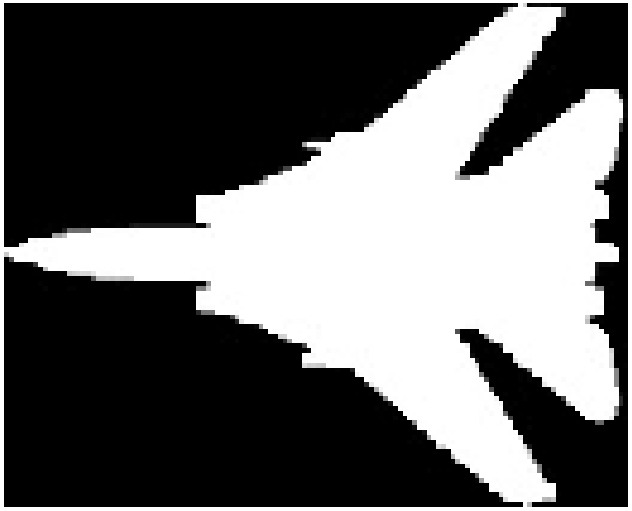
It can be smoothed considering the difference in mean angular direction of n pixels

$$\kappa(t) = \frac{1}{n}\sum_{i=1}^{n}\theta(t+i) - \frac{1}{n}\sum_{i=1}^{n}\theta(t-i)$$

It can be smoothed considering a weighted average (e.g. Gaussian smoothing) of n pixels

$$\kappa(t) = \frac{1}{n}\sum_{i=1}^{n}w(i)\theta(t+i) - \frac{1}{n}\sum_{i=1}^{n}w(i)\theta(t-i)$$

```
%Curvature detection
function outputimage=CurvConnect(inputimage)

        [rows,columns]=size(inputimage);   %Image size
        outputimage=zeros(rows,columns);   %Result image
        [Mag,Ang]=Edges(inputimage);       %Edge Detection
        Mag=MaxSupr(Mag,Ang);              %Maximal Suppression
        Next=Cont(Mag,Ang);                %Next connected pixels


        %Compute curvature in each pixel
        for x=1:columns-1
          for y=1:rows-1
            if Mag(y,x)~=0
              n=Next(y,x,1); m=Next(y,x,2);
               if(n~=-1 & m~=-1)
                [px,py]=NextPixel(x,y,n);
                [qx,qy]=NextPixel(x,y,m);
              outputimage(y,x)=abs(Ang(py,px)-Ang(qy,qx));
            end
           end
         end
        end
```



(a) Image

(b) Detected corners

This approach does not provide reliable results

- It assumes that corner information is in the detected edge map
- It does not use any corner structure
- The measurements can be affected by quantisation error
- Accuracy is limited

Measuring curvature by changes in intensity

- Curvature is derived as a function of changes in image intensity
- Not necessary to compute edges

Measuring curvature by changes in intensity

- Curvature is derived as a function of changes in image intensity
- Not necessary to compute edges

Let us assume

- (x, y) is and edge point
- $\phi(x, y)$ is the function representing the edge direction

Measuring curvature by changes in intensity

- Curvature is derived as a function of changes in image intensity
- Not necessary to compute edges

Let us assume

- (x, y) is and edge point
- $\phi(x, y)$ is the function representing the edge direction

Around (x, y) the edge curve can be approximated in parametric form as

$$x(t) = x + t\cos(\phi(x, y))$$
$$y(t) = y + t\sin(\phi(x, y))$$

$[\cos(\phi(x,y)), \sin(\phi(x,y))]^t$ is the unit vector in the edge direction, therefore t is the arc length

$$\kappa(x,y) = \frac{d\phi(x,y)}{dt} = \frac{\partial\phi(x,y)}{\partial x}\frac{dx}{dt} + \frac{\partial\phi(x,y)}{\partial y}\frac{dy}{dt}$$

where

$$\frac{dx}{dt} = \cos(\phi(x,y)) \quad \frac{dy}{dt} = \sin(\phi(x,y))$$

$[\cos(\phi(x,y)), \sin(\phi(x,y))]^t$ is the unit vector in the edge direction, therefore t is the arc length

$$\kappa(x,y) = \frac{d\phi(x,y)}{dt} = \frac{\partial\phi(x,y)}{\partial x}\frac{dx}{dt} + \frac{\partial\phi(x,y)}{\partial y}\frac{dy}{dt}$$

where

$$\frac{dx}{dt} = \cos(\phi(x,y)) \quad \frac{dy}{dt} = \sin(\phi(x,y))$$

What is $\phi(x,y)$?

$[\cos(\phi(x,y)), \sin(\phi(x,y))]^t$ is the unit vector in the edge direction, therefore t is the arc length

$$\kappa(x,y) = \frac{d\phi(x,y)}{dt} = \frac{\partial\phi(x,y)}{\partial x}\frac{dx}{dt} + \frac{\partial\phi(x,y)}{\partial y}\frac{dy}{dt}$$

where

$$\frac{dx}{dt} = \cos(\phi(x,y)) \quad \frac{dy}{dt} = \sin(\phi(x,y))$$

What is $\phi(x,y)$?
$\phi(x,y)$ is orthogonal to the image gradient $[\mathsf{I}_x, \mathsf{I}_y]^t$

$$\phi(x,y) = \arctan\left(-\frac{\mathsf{I}_x}{\mathsf{I}_y}\right)$$

Therefore it is

Therefore it is

$$\frac{dx}{dt} = \cos(\phi(x,y)) = \frac{-I_y}{I_x^2+I_y^2} \quad \frac{dy}{dt} = \sin(\phi(x,y)) = \frac{I_x}{I_x^2+I_y^2}$$

Therefore it is

$$\frac{dx}{dt} = \cos(\phi(x,y)) = \frac{-I_y}{I_x^2+I_y^2} \quad \frac{dy}{dt} = \sin(\phi(x,y)) = \frac{I_x}{I_x^2+I_y^2}$$

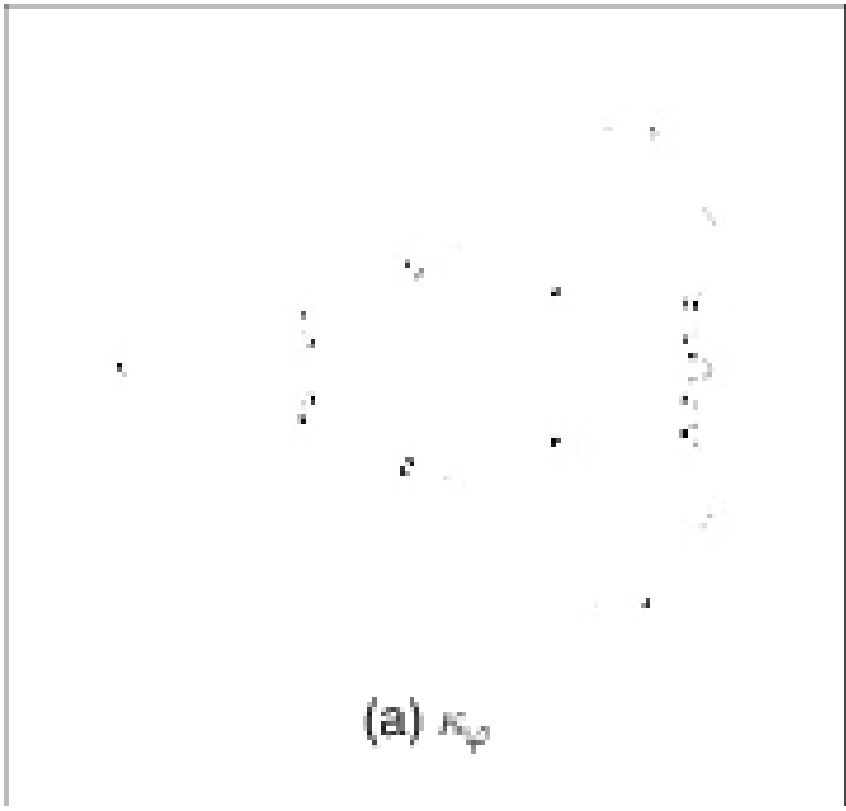
$$\kappa(x,y) = \frac{1}{I_x^2 + I_y^2}(I_y^2 I_{xx} - I_x I_y I_{yx} + I_x^2 I_{yy} - I_x I_y I_{xy})$$

This is a $\phi(x, y)$ forward measure of κ.
We can measure curvature along other directions, as $-\phi(x, y)$

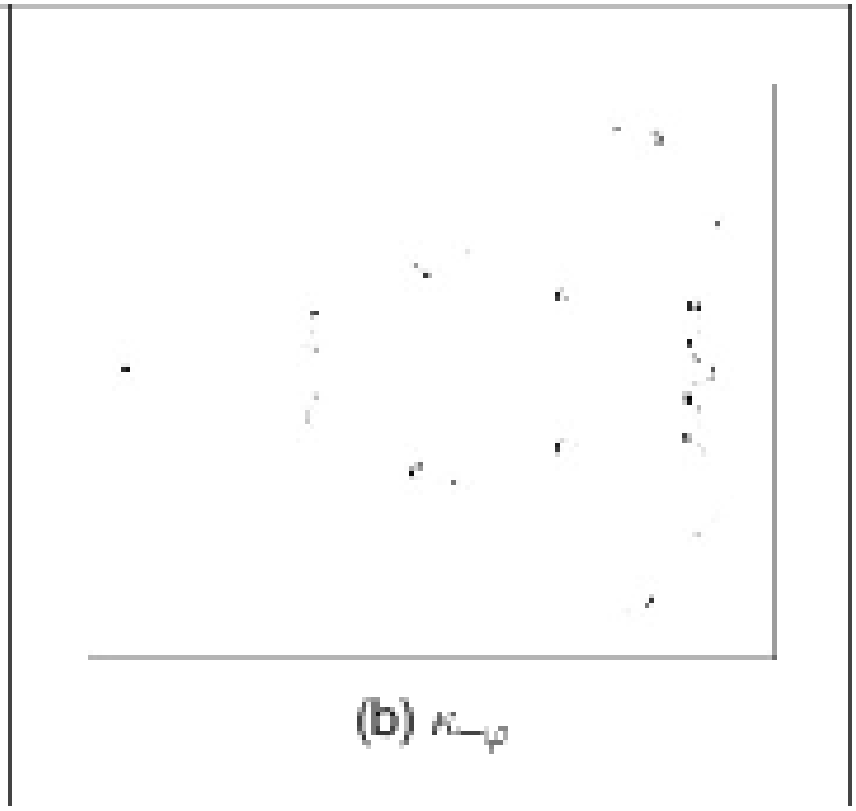
$$x(t) = x + t\cos(-\phi(x, y))$$
$$y(t) = y + t\sin(-\phi(x, y))$$

with

$$\kappa(x, y) = \frac{1}{I_x^2 + I_y^2}(I_y^2 I_{xx} - I_x I_y I_{yx} - I_x^2 I_{yy} + I_x I_y I_{xy})$$



(a) κ_φ



(b) $\kappa_{-\varphi}$

It is not a totally reliable method of getting the curvature.
Partly due to

- higher order differentiation
- local approximation of the curve

Moravec detector

Basic idea

A measure of curvature can be obtained considering changes along a particular direction in the image.

Moravec detector

Basic idea

A measure of curvature can be obtained considering changes along a particular direction in the image.

- It computes average changes in image intensity
- it shifts a $2a+1 \times 2a+1$ window in several directions
- Fixed a pixel (x, y) computes

$$E_{uv}(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} (\mathrm{I}(x+i, y+j) - \mathrm{I}(x+i+u, y+j+v))^2$$

A measure of curvature is given by

$$\kappa(x, y) = \min_{u,v} E_{u,v}(x, y)$$

where (u, v) varies in
$\{(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)\}$

A measure of curvature is given by

$$\kappa(x, y) = \min_{u,v} E_{u,v}(x, y)$$

where (u, v) varies in
$\{(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)\}$
The minimum because

A measure of curvature is given by

$$\kappa(x, y) = \min_{u,v} E_{u,v}(x, y)$$

where (u, v) varies in
$\{(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)\}$
The minimum because

- if (x, y) is in a flat area
  - $E_{u,v}$ is always small

A measure of curvature is given by

$$\kappa(x, y) = \min_{u,v} E_{u,v}(x, y)$$

where (u, v) varies in
$\{(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)\}$
The minimum because

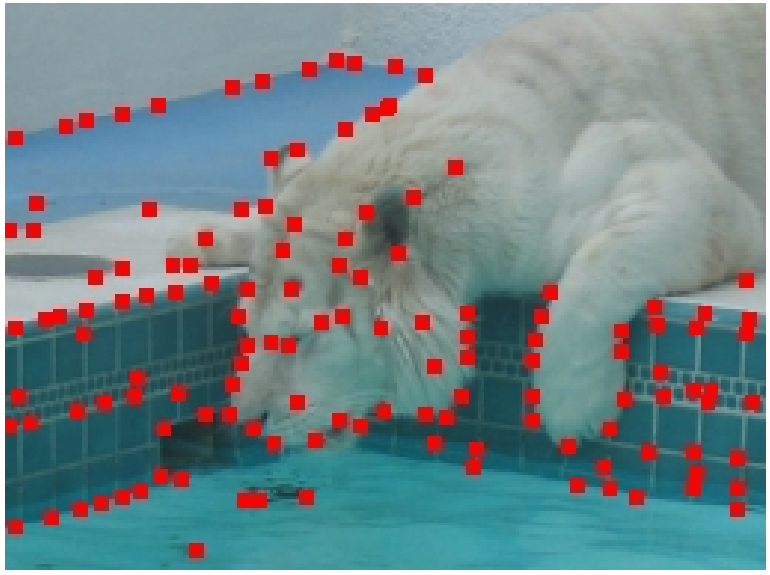
- if (x, y) is in a flat area
  - $E_{u,v}$ is always small
- if (x, y) is a proper edge
  - the curvature is small
  - $E_{u,v}$ is small for a shift along a straight edge
  - $E_{u,v}$ is large for a shift along the normal

A measure of curvature is given by

$$\kappa(x, y) = \min_{u,v} E_{u,v}(x, y)$$

where (u, v) varies in
$\{(1, 0), (1, 1), (0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1)\}$
The minimum because

- if (x, y) is in a flat area
  - $E_{u,v}$ is always small
- if (x, y) is a proper edge
  - the curvature is small
  - $E_{u,v}$ is small for a shift along a straight edge
  - $E_{u,v}$ is large for a shift along the normal
- if (x, y) is a corner point
  - all the shifts produce a large value



Main problem with Moravec's detector are

- The response is anisotropic: it considers only a small discrete sets of shifts
- The response is noisy: it uses a rectangular binary window
- It considers only the minimum of E, so it responds too readily to edges

Most of the problems solved with Harris detector.

Harris corner detector

- it gives an analytic expression for $(\mathsf{I}(x+i, y+j) - \mathsf{I}(x+i+u, y+j+v)$
- uses a Gaussian weighting mask
- changes the corner measure $\kappa(x, y)$

Harris corner detector

- it gives an analytic expression for $(\mathrm{I}(x+i, y+j) - \mathrm{I}(x+i+u, y+j+v)$
- uses a Gaussian weighting mask
- changes the corner measure $\kappa(x, y)$

Considering the first order Taylor expansion around (x, y)

$$\mathrm{I}(x+u, y+v) = \mathrm{I}(x, y) + \mathrm{I}_x(x, y)u + \mathrm{I}_y(x, y)v$$

Harris corner detector

- it gives an analytic expression for $(\mathrm{I}(x+i, y+j) - \mathrm{I}(x+i+u, y+j+v)$
- uses a Gaussian weighting mask
- changes the corner measure $\kappa(x, y)$

Considering the first order Taylor expansion around (x, y)

$$\mathrm{I}(x+u, y+v) = \mathrm{I}(x, y) + \mathrm{I}_x(x, y)u + \mathrm{I}_y(x, y)v$$

This also holds for $(x+i, y+j)$

$$\mathrm{I}(x+i+u, y+j+v) = \mathrm{I}(x+i, y+j) + \mathrm{I}_x(x+i, y+j)u + \mathrm{I}_y(x+i, y+j)v$$

$$E_{u,v}(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} w(i,j)(\mathrm{I}_x(x+i,y+j)u + \mathrm{I}_y(x+i,y+j)v)^2$$

where $w(i,j)$ is a Gaussian mask.

$$E_{u,v}(x,y) = \sum_{i=-a}^{a} \sum_{j=-a}^{a} w(i,j)(\mathrm{I}_x(x+i,y+j)u + \mathrm{I}_y(x+i,y+j)v)^2$$

where $w(i,j)$ is a Gaussian mask.
After some algebra

$$E_{u,v} = A(x,y)u^2 + 2C(x,y)uv + B(x,y)v^2$$

where

$$A(x,y) = \sum_{i,j=-a}^{a} \mathrm{I}_x(x+i,y+j)^2 \quad B(x,y) = \sum_{i,j=-a}^{a} \mathrm{I}_y(x+i,y+j)^2$$

$$C(x,y) = \sum_{i,j=-a}^{a} \mathrm{I}_x(x+i,y+j)\mathrm{I}_y(x+i,y+j)$$

- The problem is now to find when $E_{u,v}$ is minimised.

- The problem is now to find when $E_{u,v}$ is minimised.
- E_{uv} is a quadratic function and therefore it has 2 principal axes (Note that isocurves are conics)

- The problem is now to find when $E_{u,v}$ is minimised.
- E_{uv} is a quadratic function and therefore it has 2 principal axes (Note that isocurves are conics)
- We can rotate the function so that the axes are parallel to the coordinate system

$$\mathrm{R}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mu \\ \nu \end{bmatrix}$$

- The problem is now to find when $E_{u,v}$ is minimised.
- E_{uv} is a quadratic function and therefore it has 2 principal axes (Note that isocurves are conics)
- We can rotate the function so that the axes are parallel to the coordinate system

$$\mathrm{R}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \mu \\ \nu \end{bmatrix}$$

- $E_{\mu,\nu} = \alpha(x,y)^2\mu^2 + \beta(x,y)^2\nu^2$

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small
- if (x, y) is a edge

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small
- if (x, y) is a edge
  - smallest variation (smallest E) along the edge direction
  - largest variation (largest E) along the normal

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small
- if (x, y) is a edge
  - smallest variation (smallest E) along the edge direction
  - largest variation (largest E) along the normal
  - edge direction and normal are the principal axes
  - $\alpha(x, y)$ large and $\beta(x, y)$ small, or vice-versa

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small
- if (x, y) is a edge
  - smallest variation (smallest E) along the edge direction
  - largest variation (largest E) along the normal
  - edge direction and normal are the principal axes
  - $\alpha(x, y)$ large and $\beta(x, y)$ small, or vice-versa
- if (x, y) is a corner

The values of $\alpha(x, y)$ ad $\beta(x, y)$ are proportional to $E_{\mu,\nu}(x, y)$

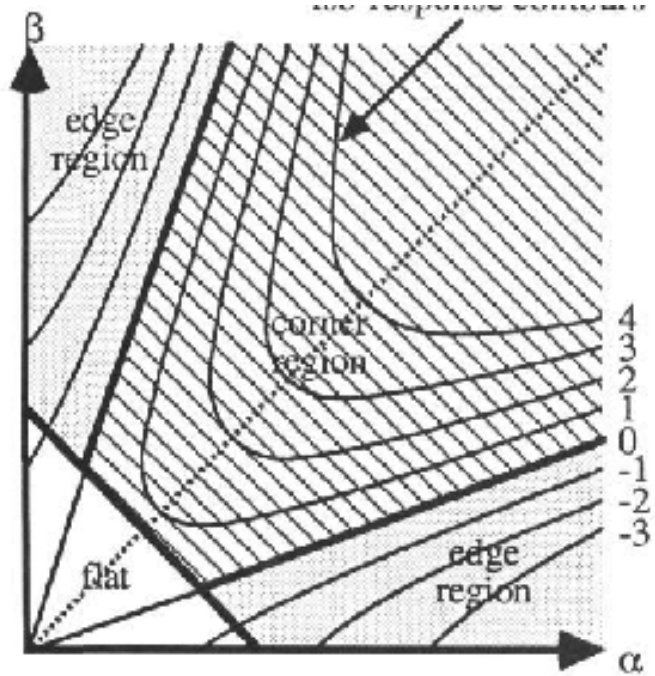
- if (x, y) is in a flat region
  - $E_{\mu,\nu}$ is small for each (μ, ν)
  - both $\alpha(x, y)$ and $\beta(x, y)$ are small
- if (x, y) is a edge
  - smallest variation (smallest E) along the edge direction
  - largest variation (largest E) along the normal
  - edge direction and normal are the principal axes
  - $\alpha(x, y)$ large and $\beta(x, y)$ small, or vice-versa
- if (x, y) is a corner
  - large variation along all directions
  - both $\alpha(x, y)$ and $\beta(x, y)$ are large

The curvature (corner response) is defined as

$$\kappa(x, y) = \alpha(x, y)\beta(x, y) - K(\alpha(x, y) + \beta(x, y))^2$$

κ is

- small in flat areas
- positive for corners
- negative for edges



How do we get the response?

- $$\mathrm{M} = \begin{bmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{bmatrix}$$
- $D = [u, v]^t$
- $E_{uv}(x, y) = D^t \mathrm{M} D$
- $E_{\mu,\nu}(x, y) = (\mathrm{R}D)^t \mathrm{M}\mathrm{R}D = D^t \mathrm{Q} D$
- $$\mathrm{Q} = \begin{bmatrix} \alpha(x,y) & 0 \\ 0 & \beta(x,y) \end{bmatrix}$$

How do we get the response?

- $$\mathrm{M} = \begin{bmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{bmatrix}$$
- $D = [u, v]^t$
- $E_{uv}(x, y) = D^t \mathrm{M} D$
- $E_{\mu,\nu}(x, y) = (\mathrm{R}D)^t \mathrm{M} \mathrm{R} D = D^t \mathrm{Q} D$
- $$\mathrm{Q} = \begin{bmatrix} \alpha(x,y) & 0 \\ 0 & \beta(x,y) \end{bmatrix}$$

- α and β are the eigenvalues of M
- We can
  - Explicitly compute the eigenvalues
  - Determine $\alpha\beta$ and $\alpha + \beta$

Harris follows the second approach that is faster

Harris follows the second approach that is faster

- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$

Harris follows the second approach that is faster

- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$

Harris follows the second approach that is faster

- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$
- $\alpha\beta = \det(Q) = \det(M)$

Harris follows the second approach that is faster

- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$
- $\alpha\beta = \det(Q) = \det(M)$
- $\alpha\beta = AB - C^2$

Harris follows the second approach that is faster

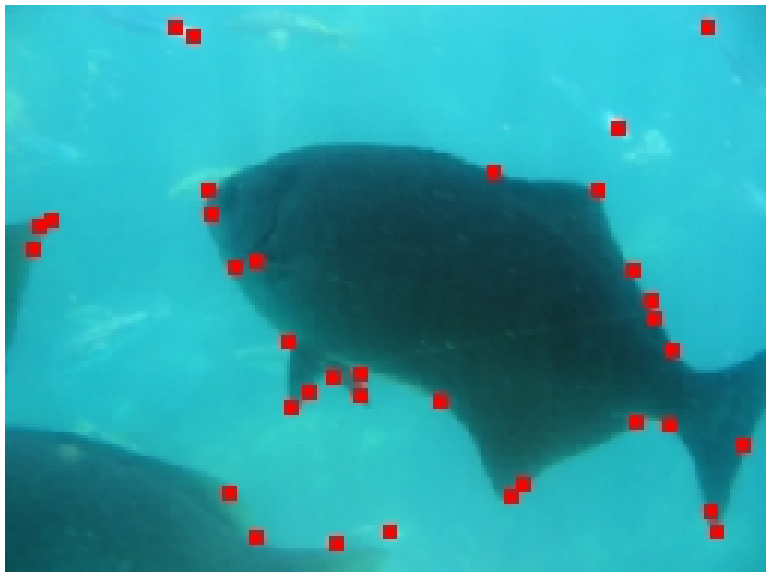
- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$
- $\alpha\beta = \det(Q) = \det(M)$
- $\alpha\beta = AB - C^2$
- The trace is invariant then $\alpha + \beta = \mathit{trace}(M)$

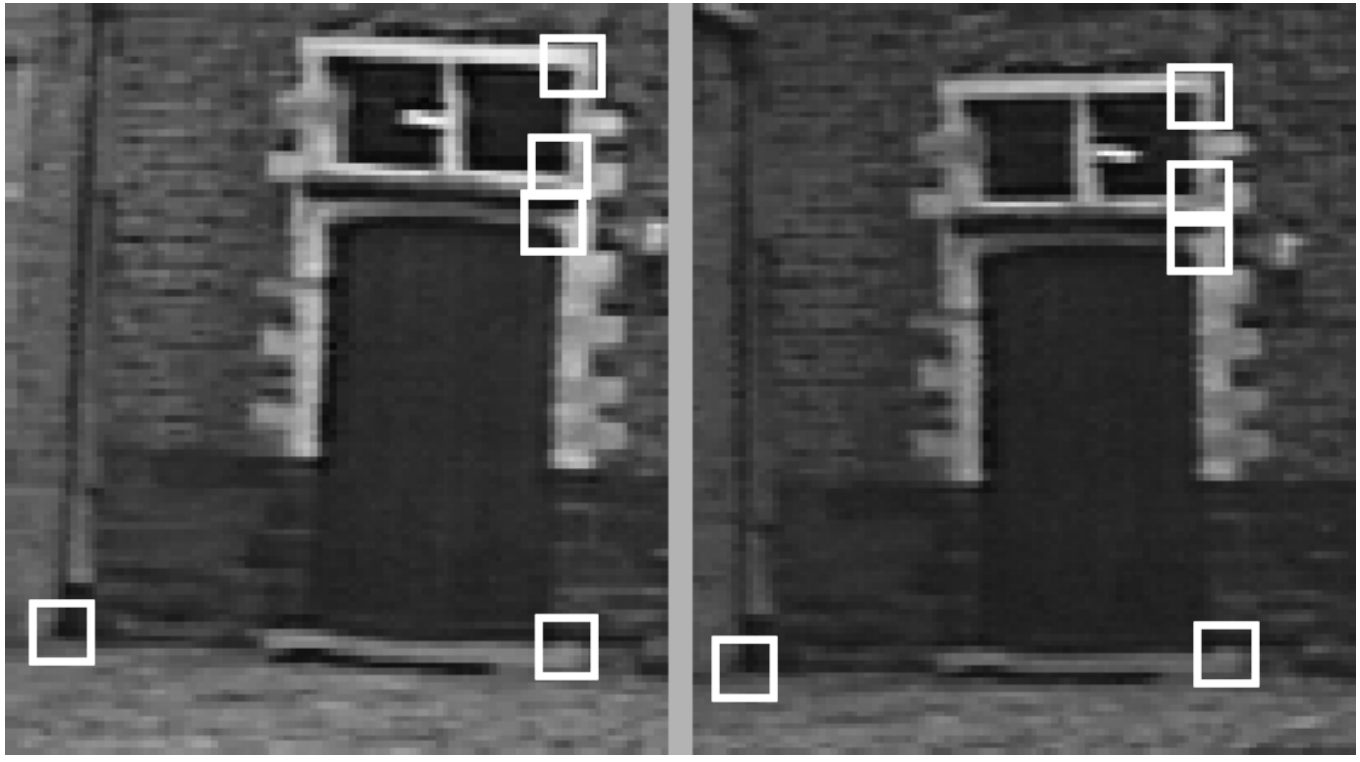
Harris follows the second approach that is faster

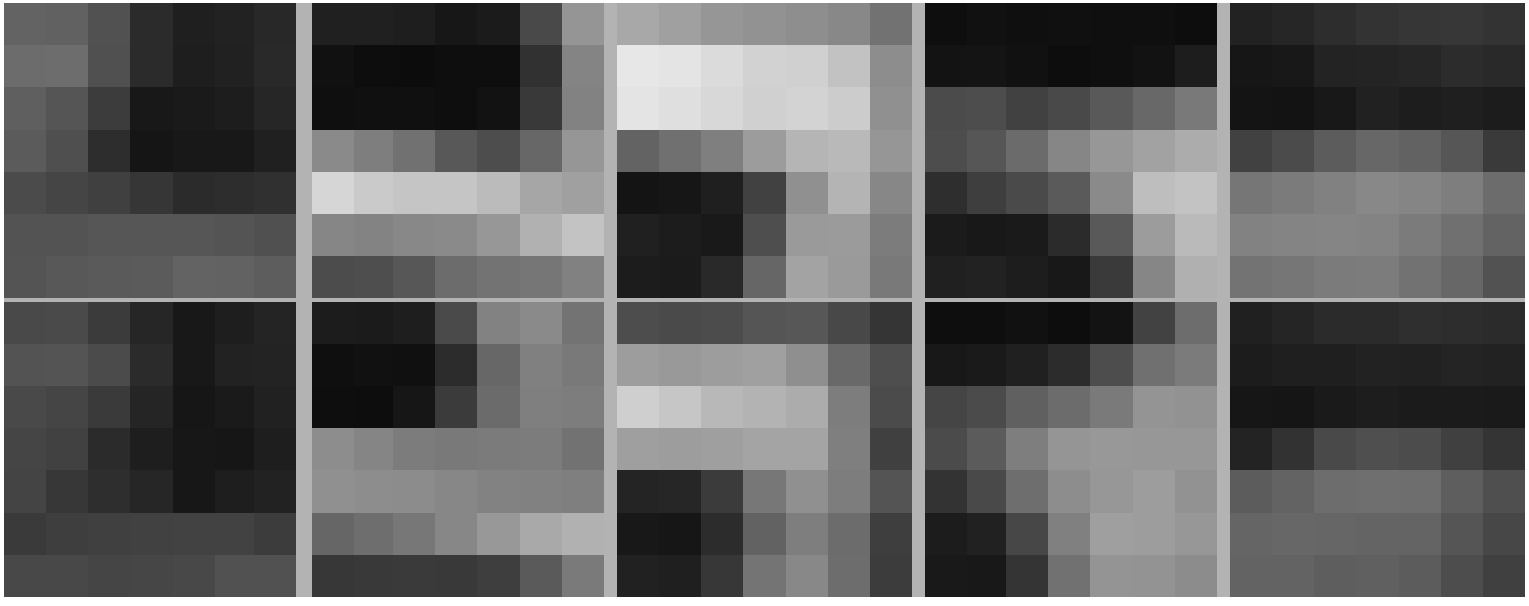
- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$
- $\alpha\beta = \det(Q) = \det(M)$
- $\alpha\beta = AB - C^2$
- The trace is invariant then $\alpha + \beta = trace(M)$
- $\kappa(x, y) = A(x, y)B(x, y) - C(x, y)^2 - K(A(x, y) + B(x, y))^2$

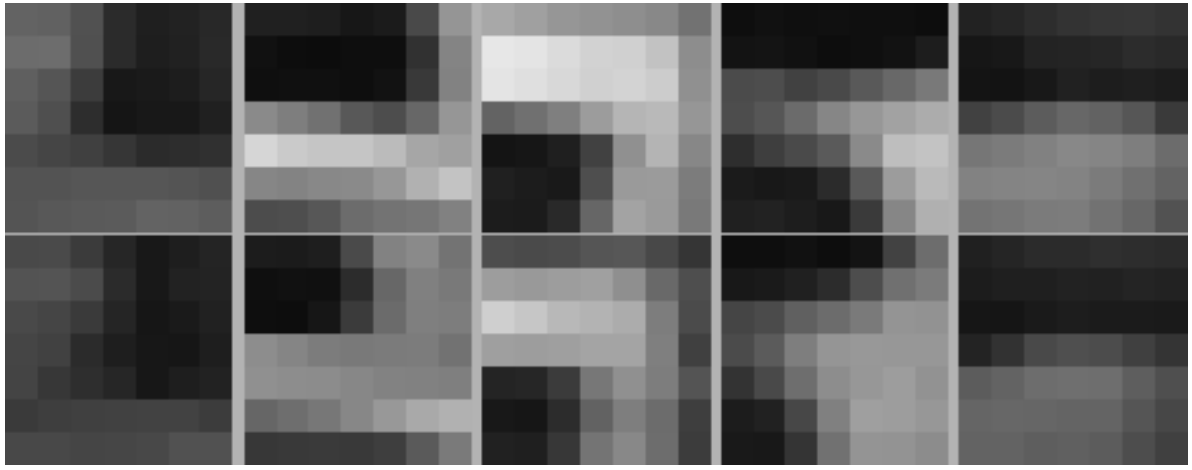
Harris follows the second approach that is faster

- $\det(Q) = \det(R^t M R) = \det(R^t)\det(M)\det(R)$
- $\det(R) = 1$
- $\alpha\beta = \det(Q) = \det(M)$
- $\alpha\beta = AB - C^2$
- The trace is invariant then $\alpha + \beta = trace(M)$
- $\kappa(x, y) = A(x, y)B(x, y) - C(x, y)^2 - K(A(x, y) + B(x, y))^2$
- K is empirically set to values between 0.004 and 0.15









0.9639	-0.3994	-0.1627	-0.3868	0.1914
-0.0533	**0.7503**	-0.4677	0.5115	0.7193
-0.1826	-0.3905	**0.7730**	0.1475	-0.7457
-0.2724	0.4878	0.1640	**0.7862**	0.2077
0.0835	0.5044	-0.4541	0.2802	**0.9876**

Intensity cross-correlation values for all possible combinations of the 5 corners

Tomasi-Kanade detector

Tomasi and Kanade showed that

- when affine deformations between images Harris $\kappa(x, y)$ is not a good measure
- they maintain the same algorithm
- $\kappa(x, y) = \min(\alpha(x, y), \beta(x, y))$

The scale space

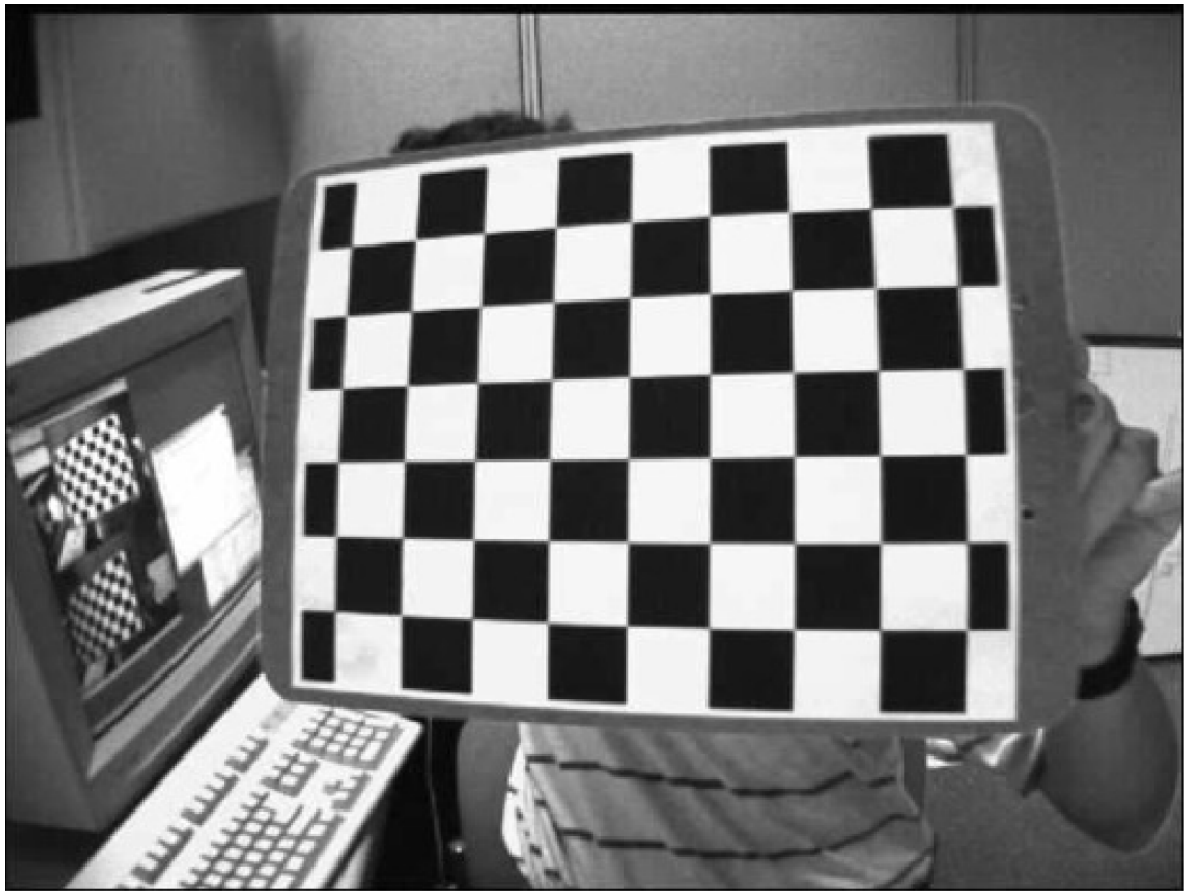
So far the content of an image is described by

- edge
- corners

The scale space

So far the content of an image is described by

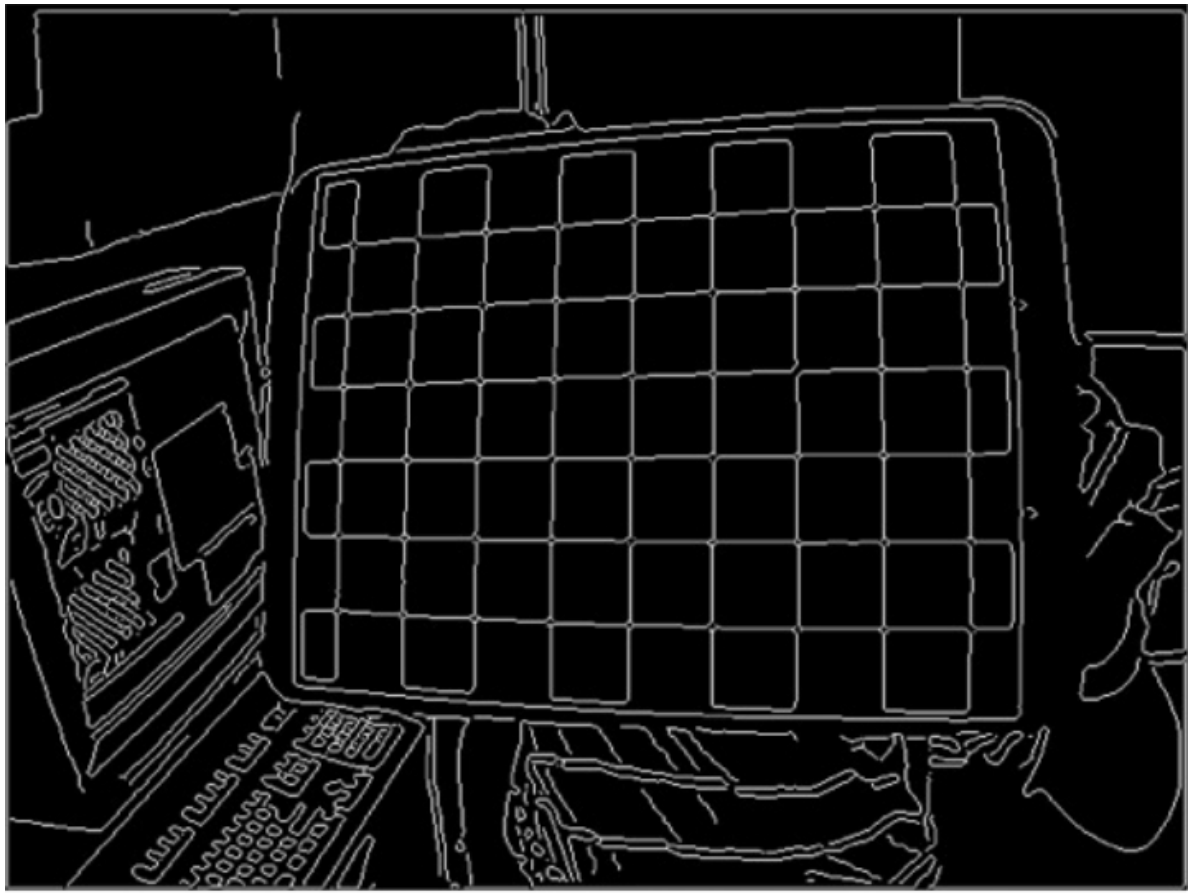
- edge
- corners



The scale space

So far the content of an image is described by

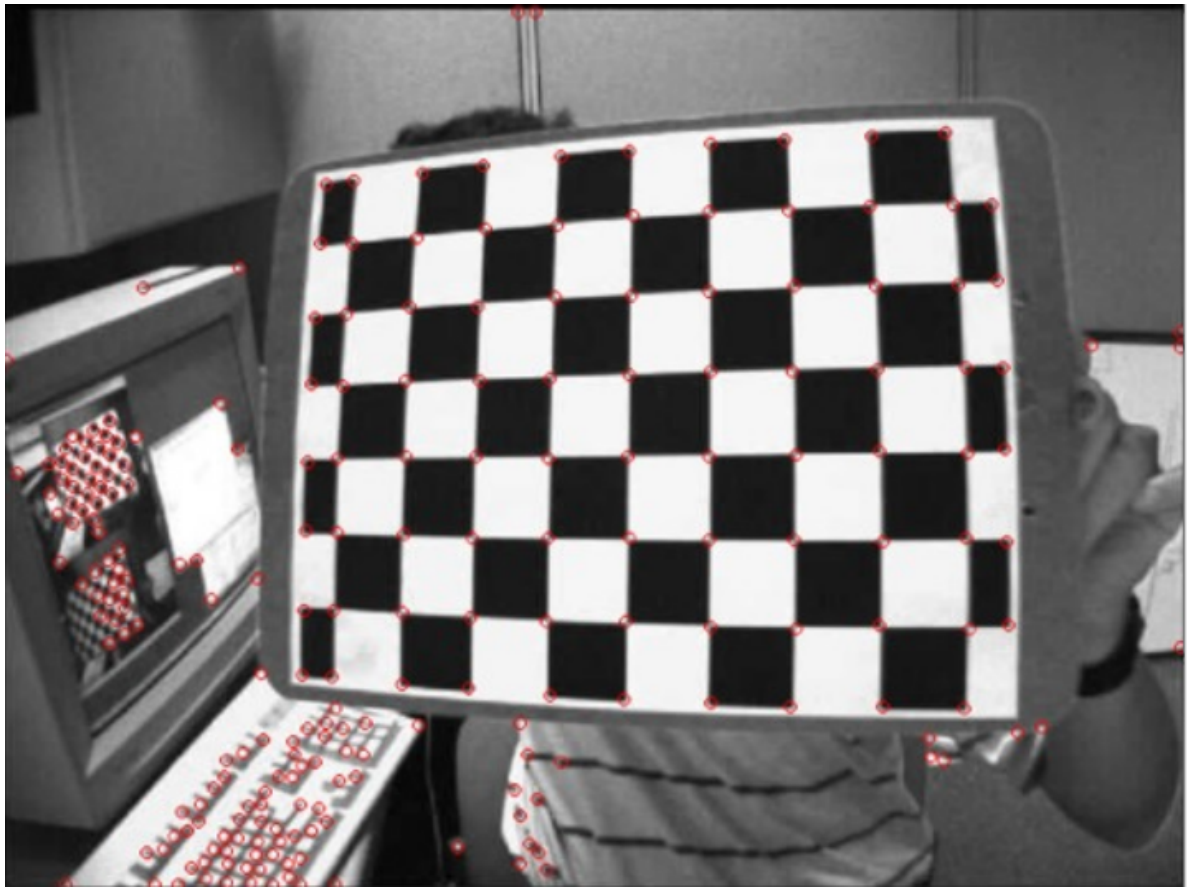
- edge
- corners



The scale space

So far the content of an image is described by

- edge
- corners



- We perceive some details depending on our distance from the objects
- Objects in an image are meaningful only if they are observed at a certain scale
- The description we associate to an objects depends strongly on the scale used for modelling the object

- We perceive some details depending on our distance from the objects
- Objects in an image are meaningful only if they are observed at a certain scale
- The description we associate to an objects depends strongly on the scale used for modelling the object

- We perceive some details depending on our distance from the objects
- Objects in an image are meaningful only if they are observed at a certain scale
- The description we associate to an objects depends strongly on the scale used for modelling the object

Examples

Examples

Examples

What is scale?

It captures the information about the spatial extension of a structure in the image.

What is scale?

It captures the information about the spatial extension of a structure in the image.

What is scale?

It captures the information about the spatial extension of a structure in the image.

What is scale?

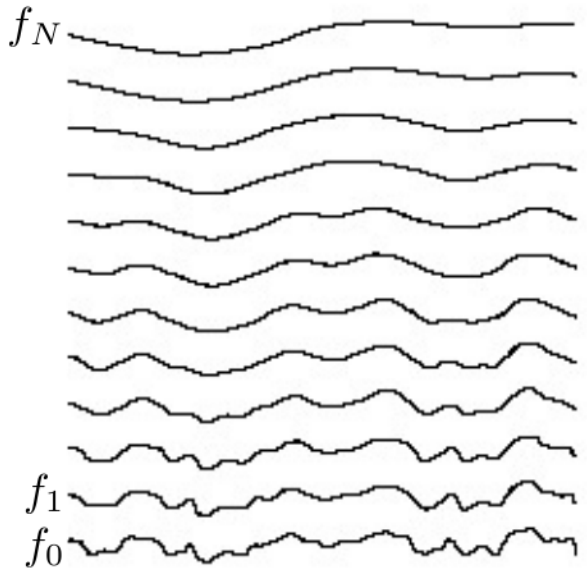
It captures the information about the spatial extension of a structure in the image.

Multi-scale representation

- In general we cannot say which one is the interesting scale
  - We do not know what is in the image.
- It is necessary a method for the automatic selection of the scale
- Different solutions
  - Gaussian scale space [Lindeberg 1994]
  - DoG and SIFT [Lowe 1999]
  - Harris multi-scale [Mikolajczyk 2002]

Scale space: definition

- The original signal is represented by a parametric family of signals
- All the signals are derived from the original one
- Finer structures are suppressed
- No new structures must be created
- Question: how do we move from f_i to f_{i+1}?



Given a signal

$$f : \boldsymbol{R}^n \longrightarrow \boldsymbol{R}$$

Its scale space representation is given by

$$\mathcal{L}(\mathrm{x}, t) = \int f(\mathrm{x} - \alpha) g(\alpha)\, d\alpha$$

where

Given a signal

$$f : \boldsymbol{R}^n \longrightarrow \boldsymbol{R}$$

Its scale space representation is given by

$$\mathcal{L}(\mathrm{x}, t) = \int f(\mathrm{x} - \alpha) g(\alpha)\, d\alpha$$

where

- t is a scale parameter

Given a signal

$$f : \boldsymbol{R}^n \longrightarrow \boldsymbol{R}$$

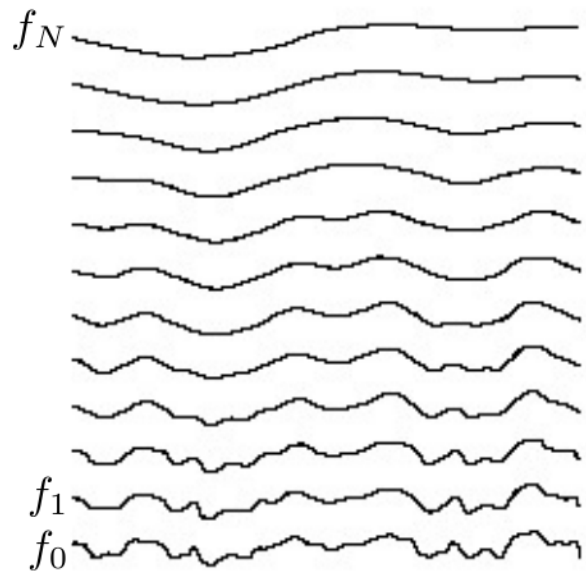
Its scale space representation is given by

$$\mathcal{L}(\mathrm{x}, t) = \int f(\mathrm{x} - \alpha) g(\alpha)\, d\alpha$$

where

- t is a scale parameter
- g is a Gaussian with $\sigma = \sqrt{t}$

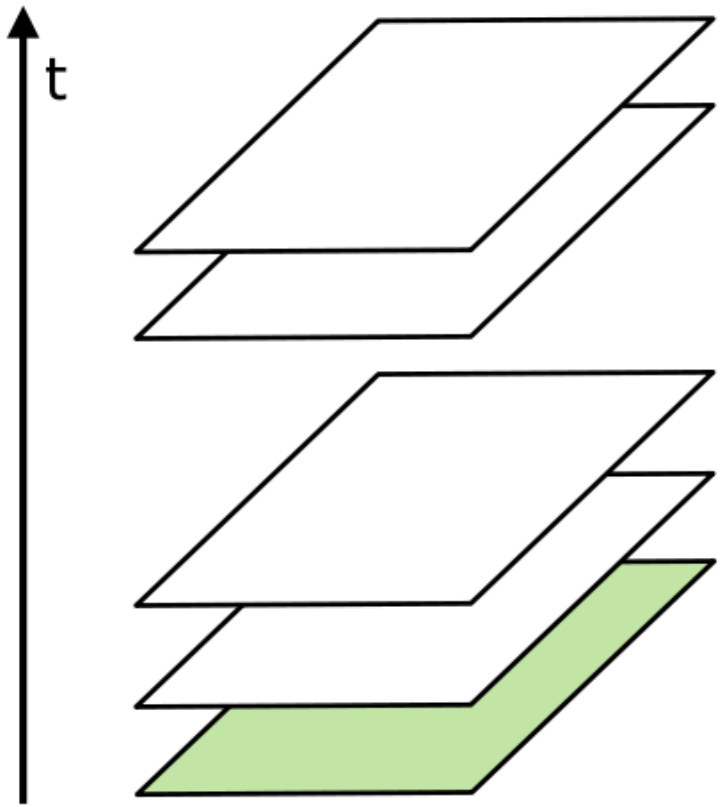
$$f_i = f_0 \star g(\sigma_i)$$

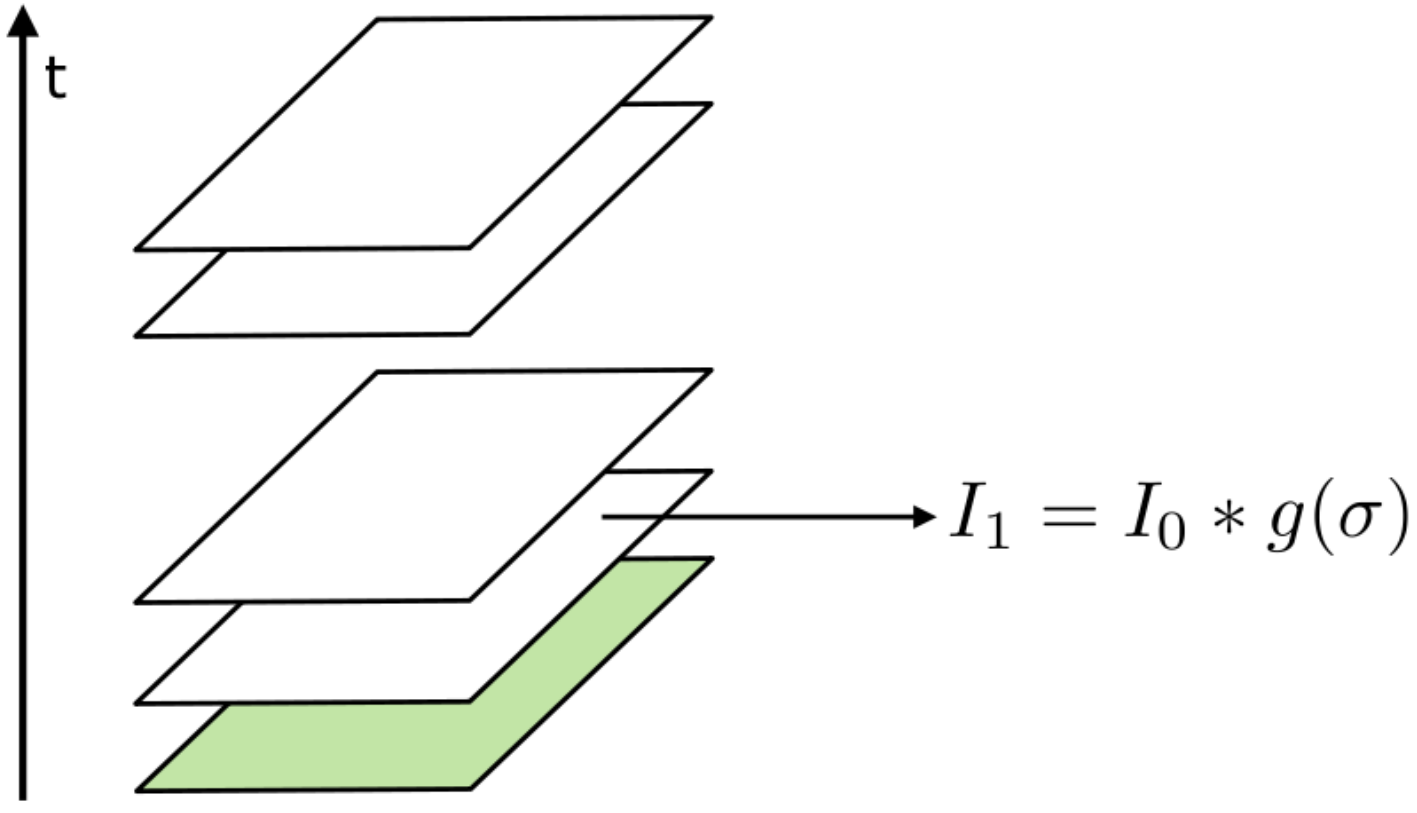


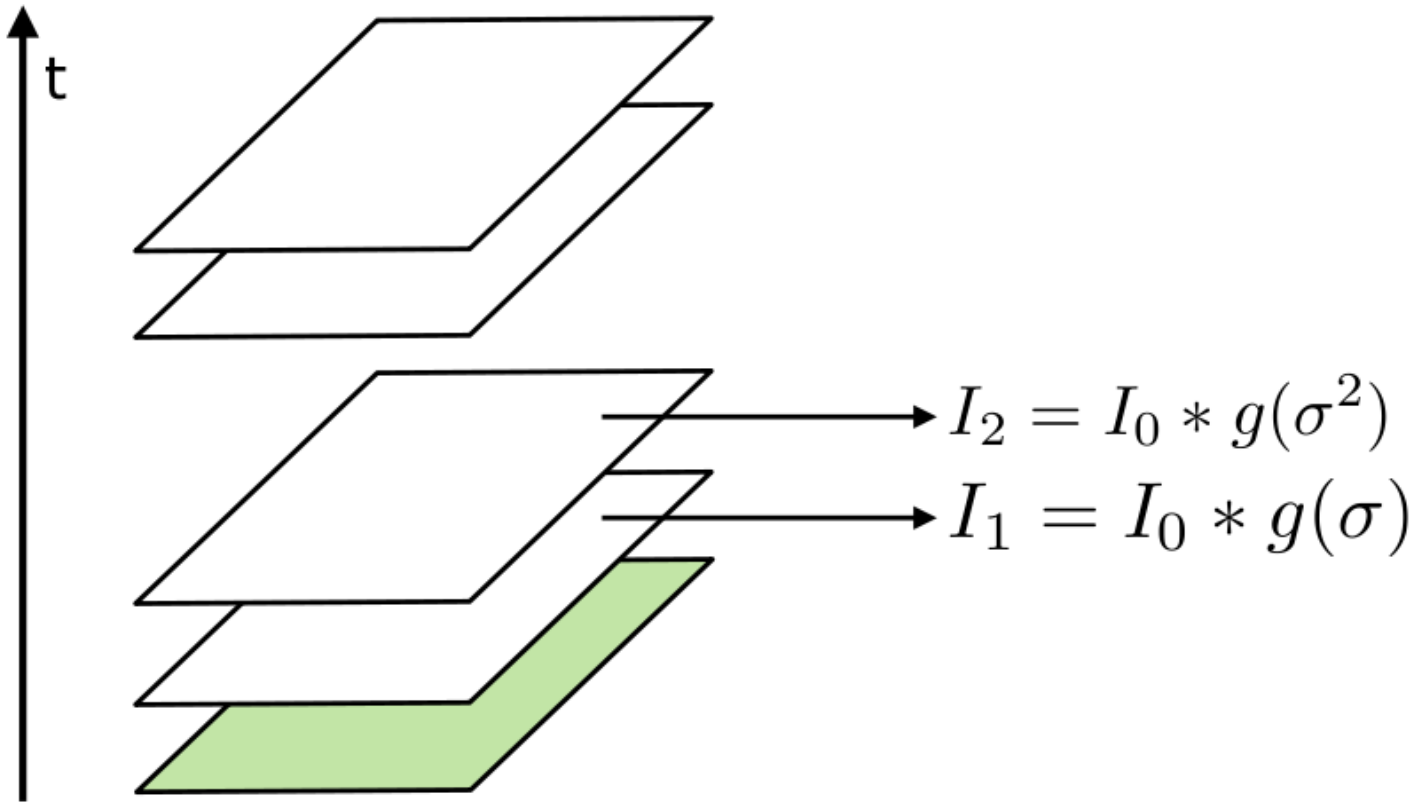
What for an image?

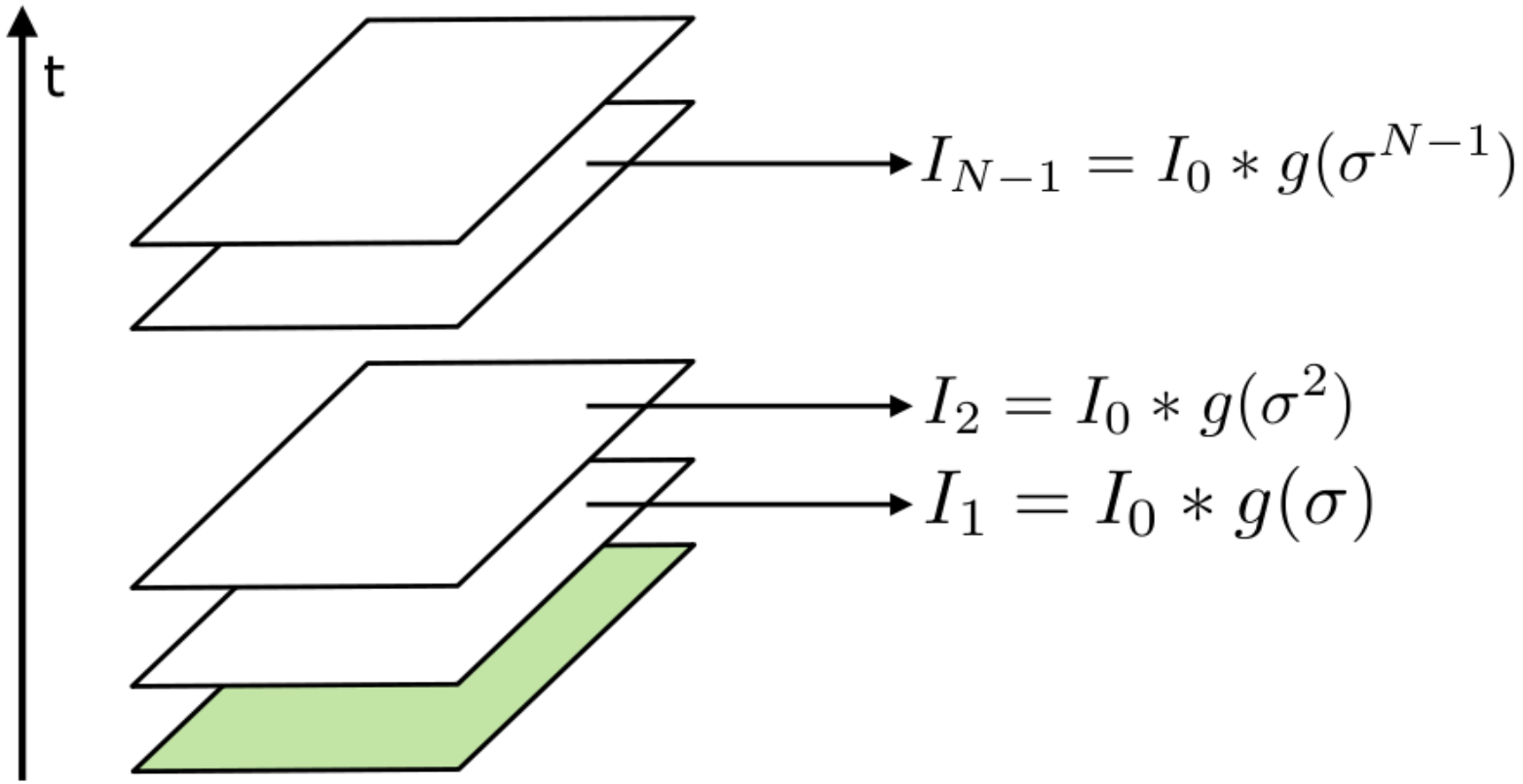
- structures start disappearing with scale
- no new structures are created

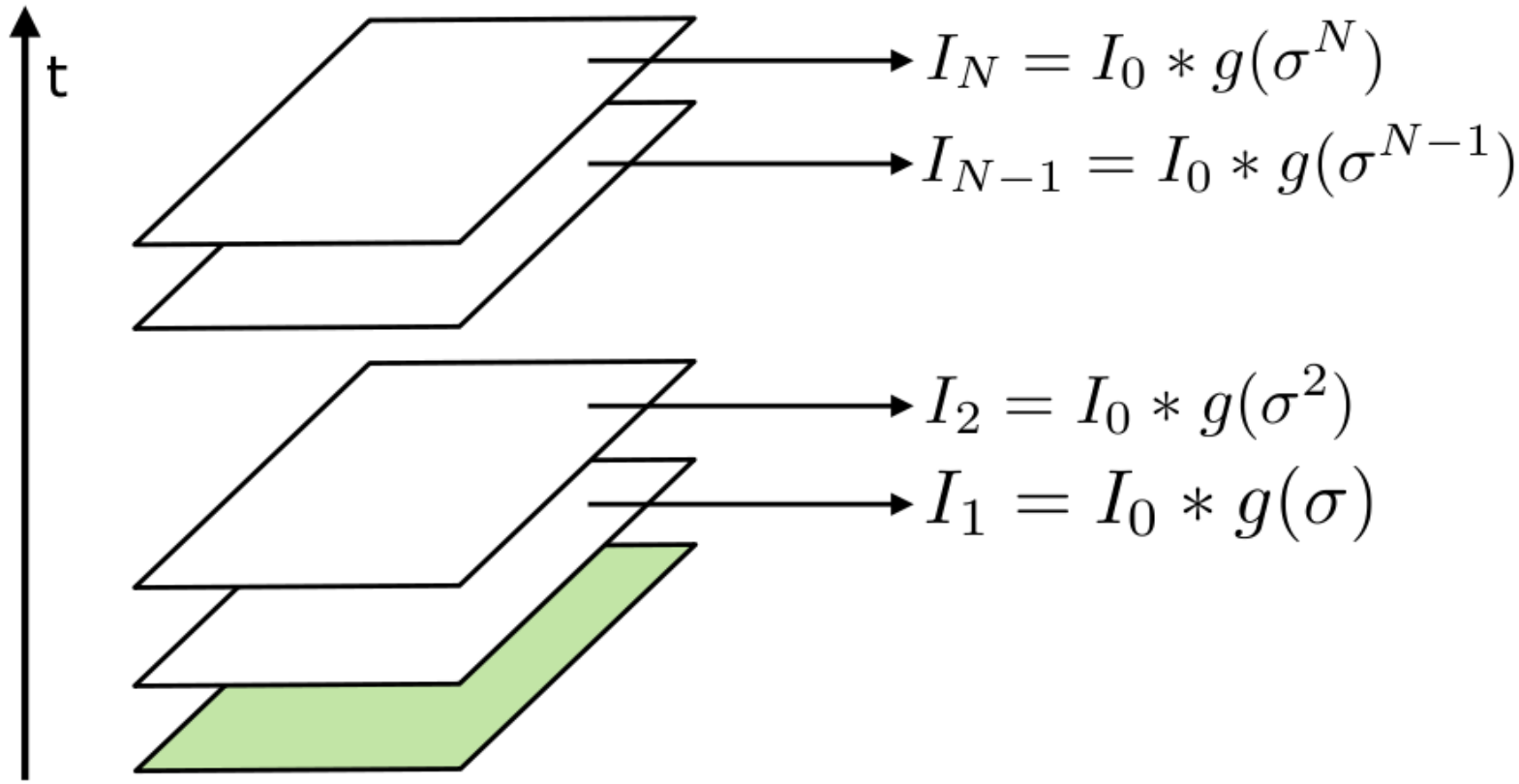
- Smallest details are identified only at finer scale (i.e., higher resolution)
  - We wouldn't see them from far away
  - They tend to disappear as the scale parameter is increasing
- Largest structure are identified at the coarsest scale (i.e., low resolution)


t


t
$I_1 = I_0 * g(\sigma)$


t
$I_2 = I_0 * g(\sigma^2)$
$I_1 = I_0 * g(\sigma)$

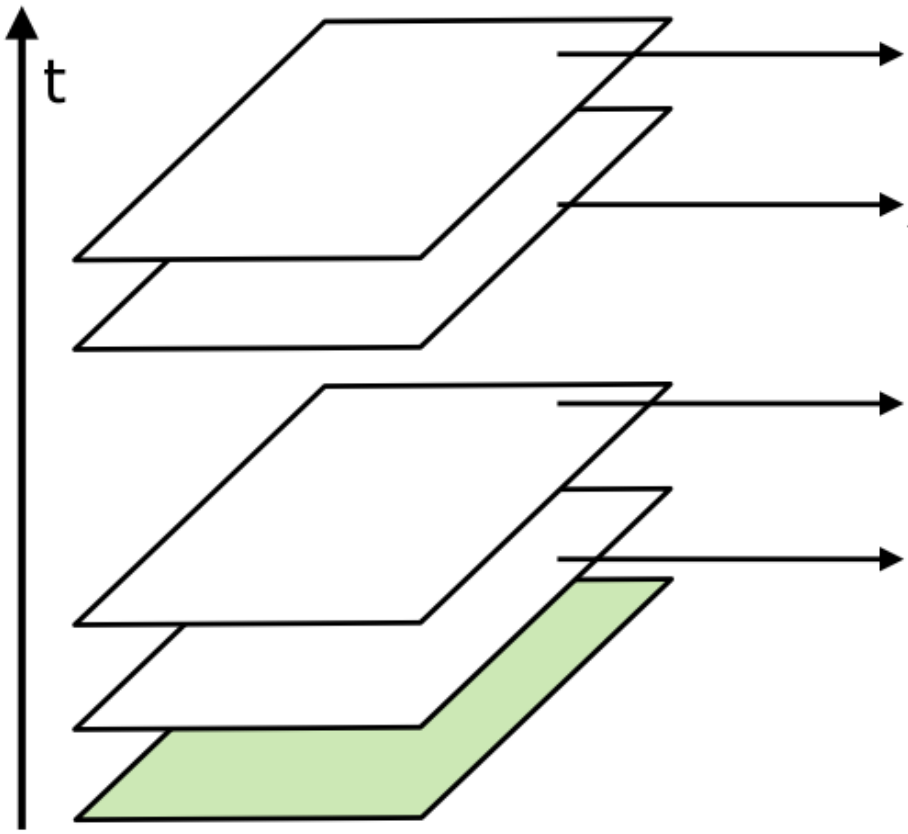

t
$I_{N-1} = I_0 * g(\sigma^{N-1})$
$I_2 = I_0 * g(\sigma^2)$
$I_1 = I_0 * g(\sigma)$


t
$I_N = I_0 * g(\sigma^N)$
$I_{N-1} = I_0 * g(\sigma^{N-1})$
$I_2 = I_0 * g(\sigma^2)$
$I_1 = I_0 * g(\sigma)$

- $\sigma = \sqrt{2}$ is taken as a good value for initial scale

- $\sigma = \sqrt{2}$ is taken as a good value for initial scale
- Is it computationally a good idea to filter the original image for each scale?

- $\sigma = \sqrt{2}$ is taken as a good value for initial scale
- Is it computationally a good idea to filter the original image for each scale?
- We can filter the image at the previous scale.



$$I_N = I_{N-1} * g(\frac{\sigma^N}{\sqrt{N+1}})$$

$$I_{N-1} = I_{N-2} * g(\frac{\sigma^{N-1}}{\sqrt{N}})$$

$$I_2 = I_1 * g(\frac{\sigma^2}{\sqrt{3}})$$

$$I_1 = I_0 * g(\frac{\sigma}{\sqrt{2}})$$

$$\frac{\sigma^i}{\sqrt{i}}$$

Pyramidal representation

- All the images in the scale space have the same size
- Another approach uses a pyramidal representation
  - At each level the size is reduced (typically by half[2])
  - A mean filter can be used for the down-sampling

Feature detection in the scale space

- We deal with N filtered images
- The main idea is
  - Extract interesting points at each scale
  - Select the most adequate scale
- Different features:
  - edge
  - corners
  - blob
  - ...
- Algorithm depends on the feature
- We focus on corners/points

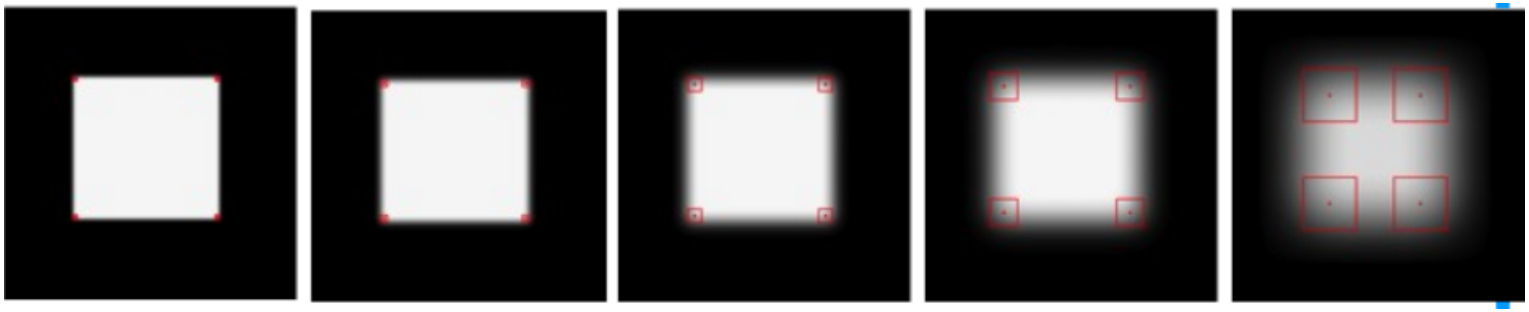
Tomasi-Kanade in the scale space

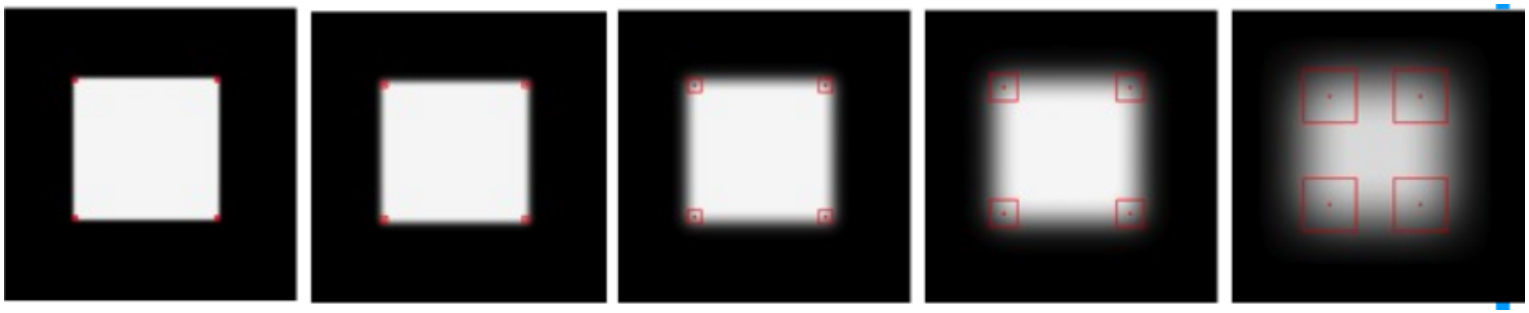
The extraction is, mutatis mutandis, the same as for the single scale case.

- Compute the gradient
- For each point
  - Compute the matrix M
  - Compute the smallest eigenvalue λ
  - If λ is large then mark the point
- Non-maxima suppression

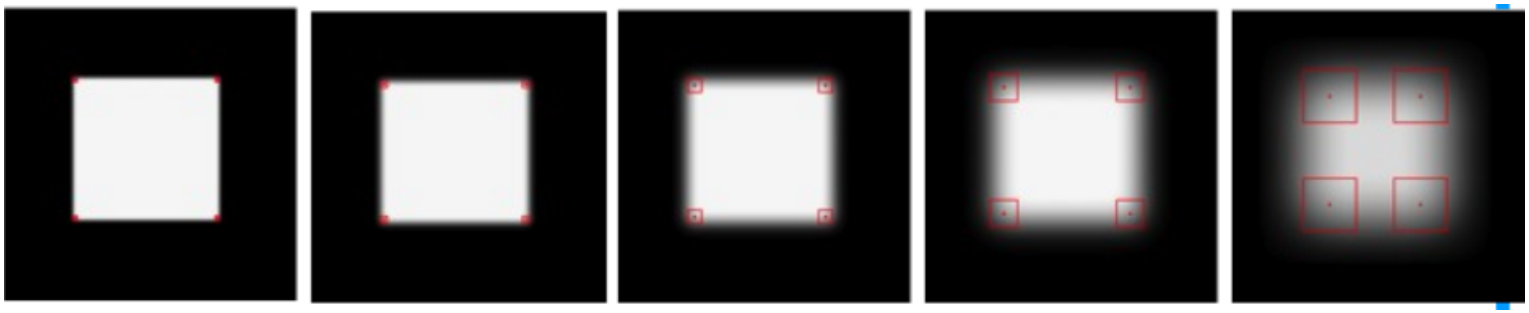
Each detected point will be identified by

- coordinates
- scale



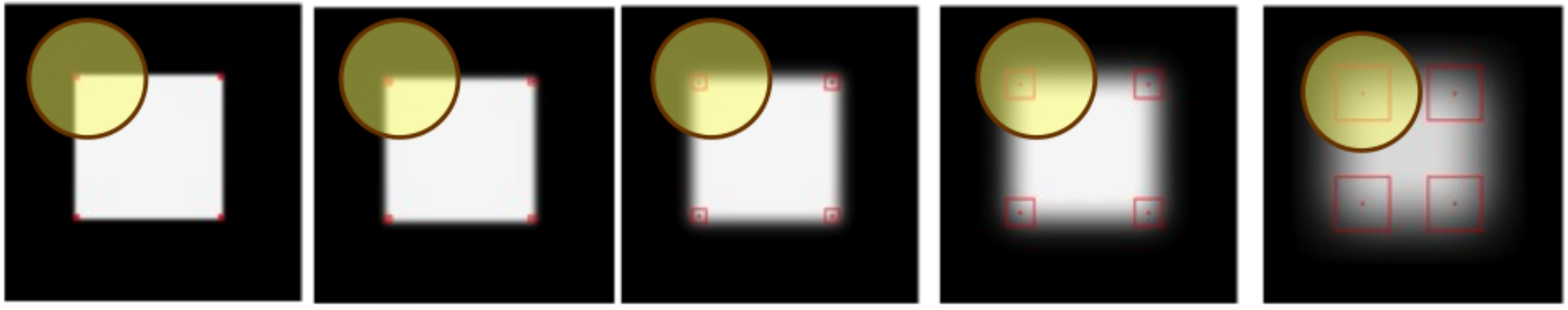


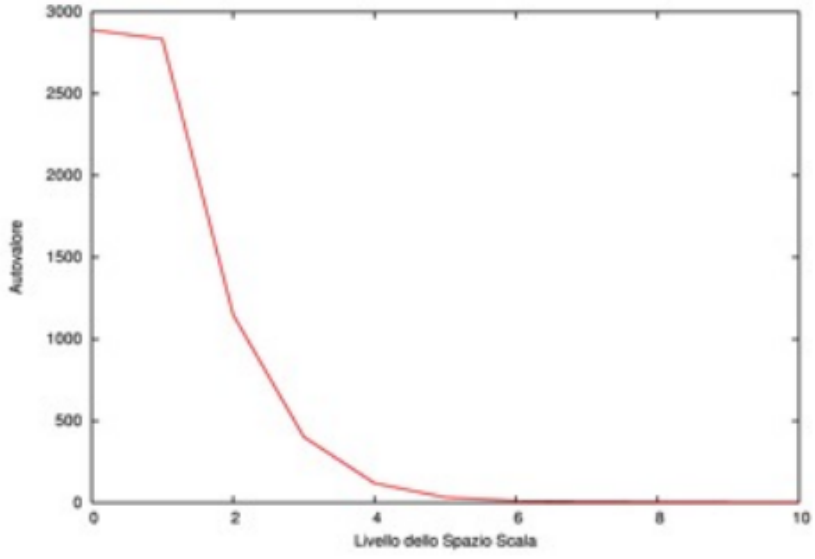
- Localisation is less accurate as scale increases



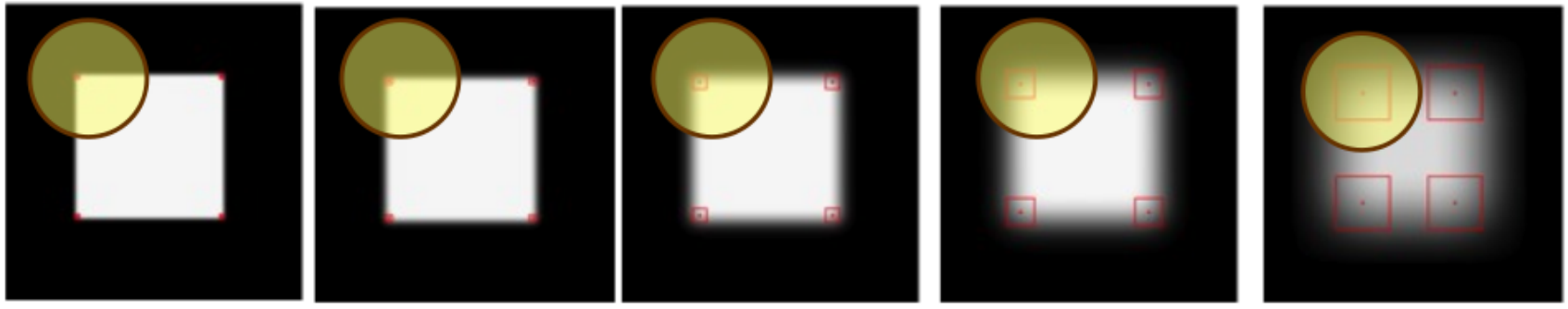
- Localisation is less accurate as scale increases
- Corners moves away in the direction of the gradient

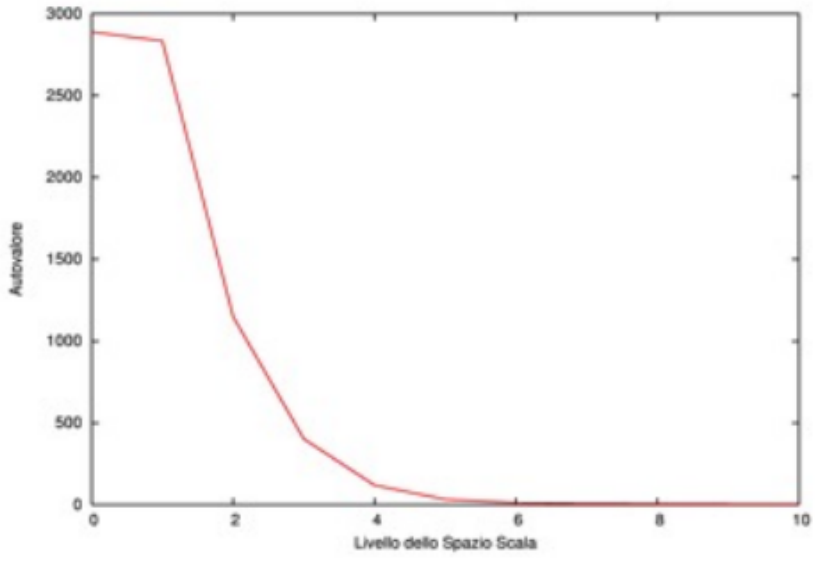
How does λ change with scale?





How does λ change with scale?

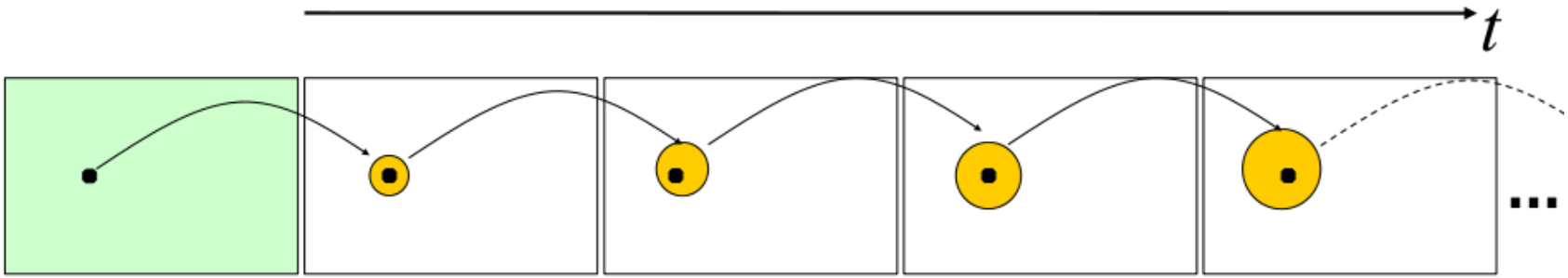




Therefore λ cannot help in selecting the best scale

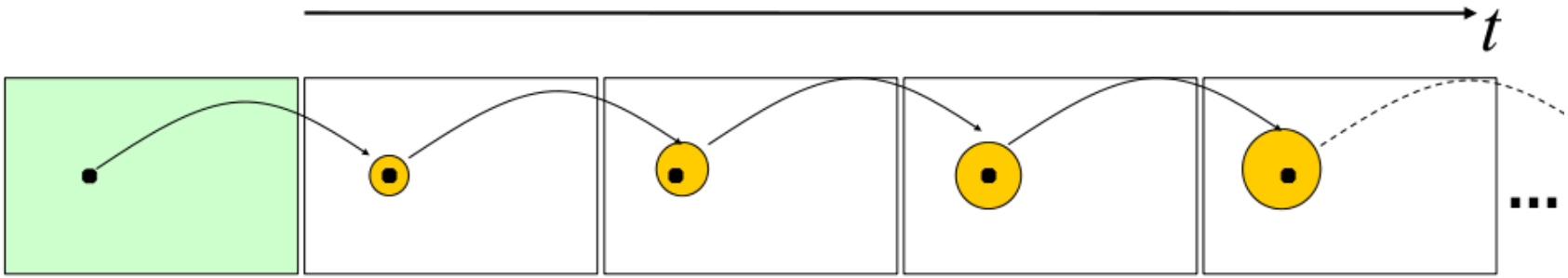
Automatic scale selection

We can reconstruct the path of a corner in the scale space.



Automatic scale selection

We can reconstruct the path of a corner in the scale space.

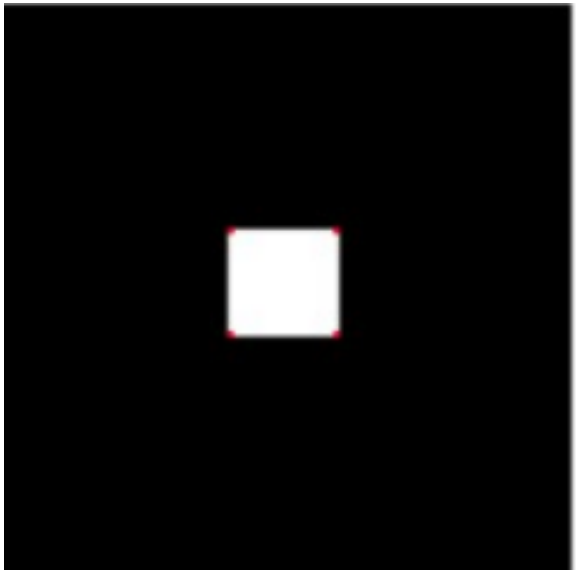


- We use a corner measure, defined by Lindeberg
- Given a corner path c_1,..., c_k

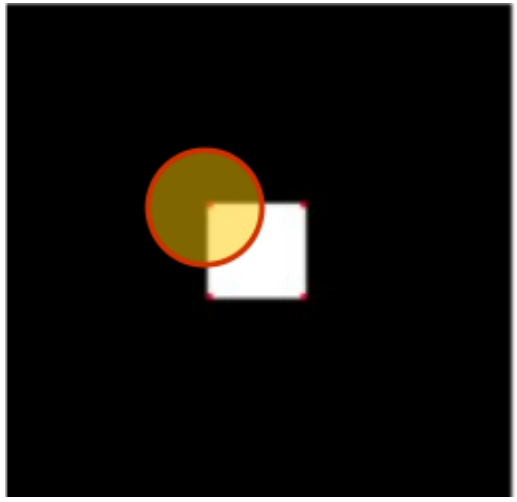
$$s(c_i) = t^2[I_y^2(c_i)I_{xx}(c_i) - 2I_x(c_i)I_y(c_i)I_{xy}(c_i) + I_x^2(c_i)I_{yy}(c_i)]$$

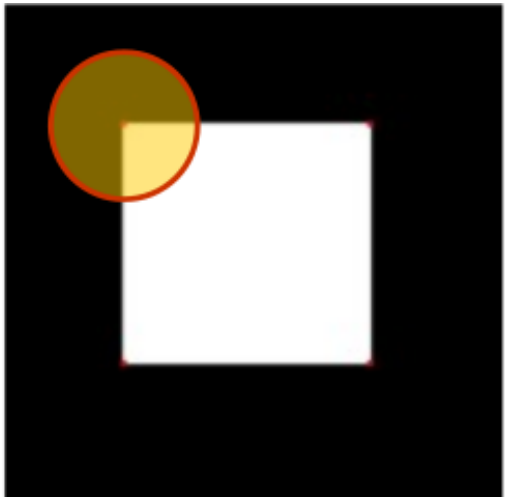
- The right scale is the one maximising the s on the path.
- A second iterative step is introduced to improve the localisation [Lindeberg 1998]

Example

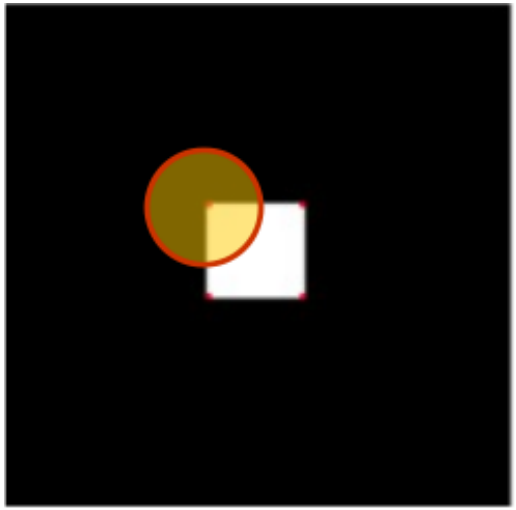


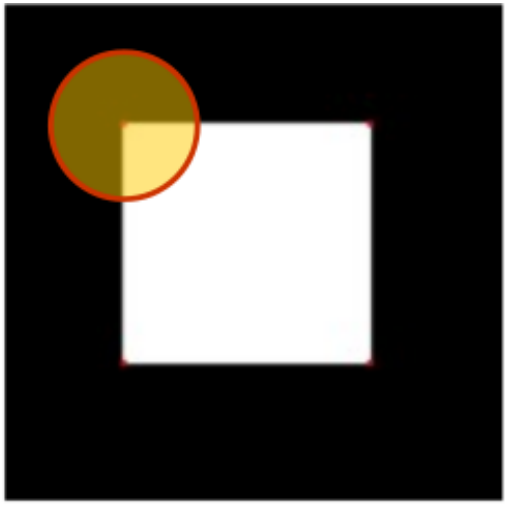
Example

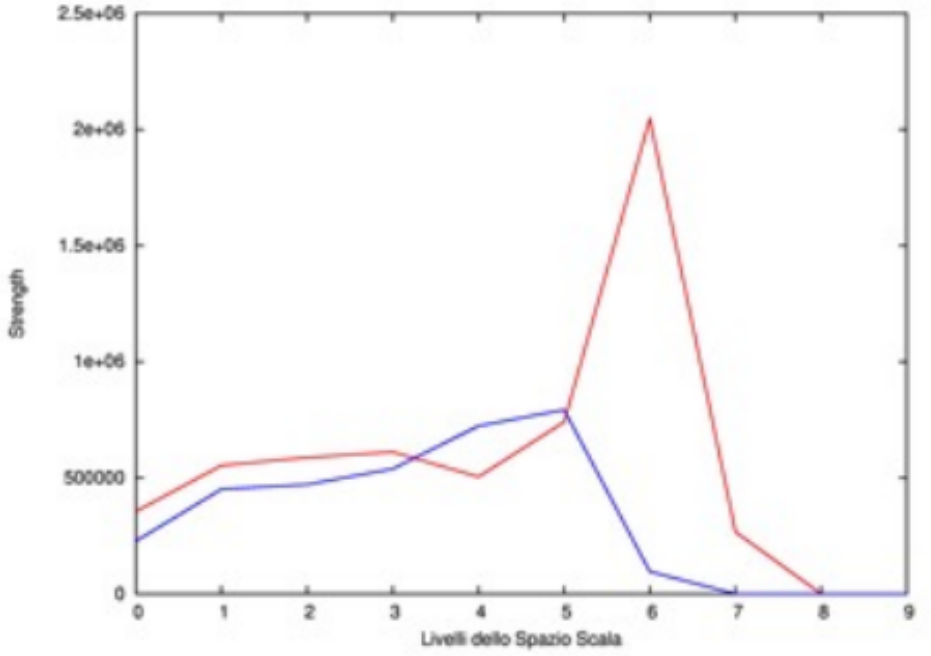




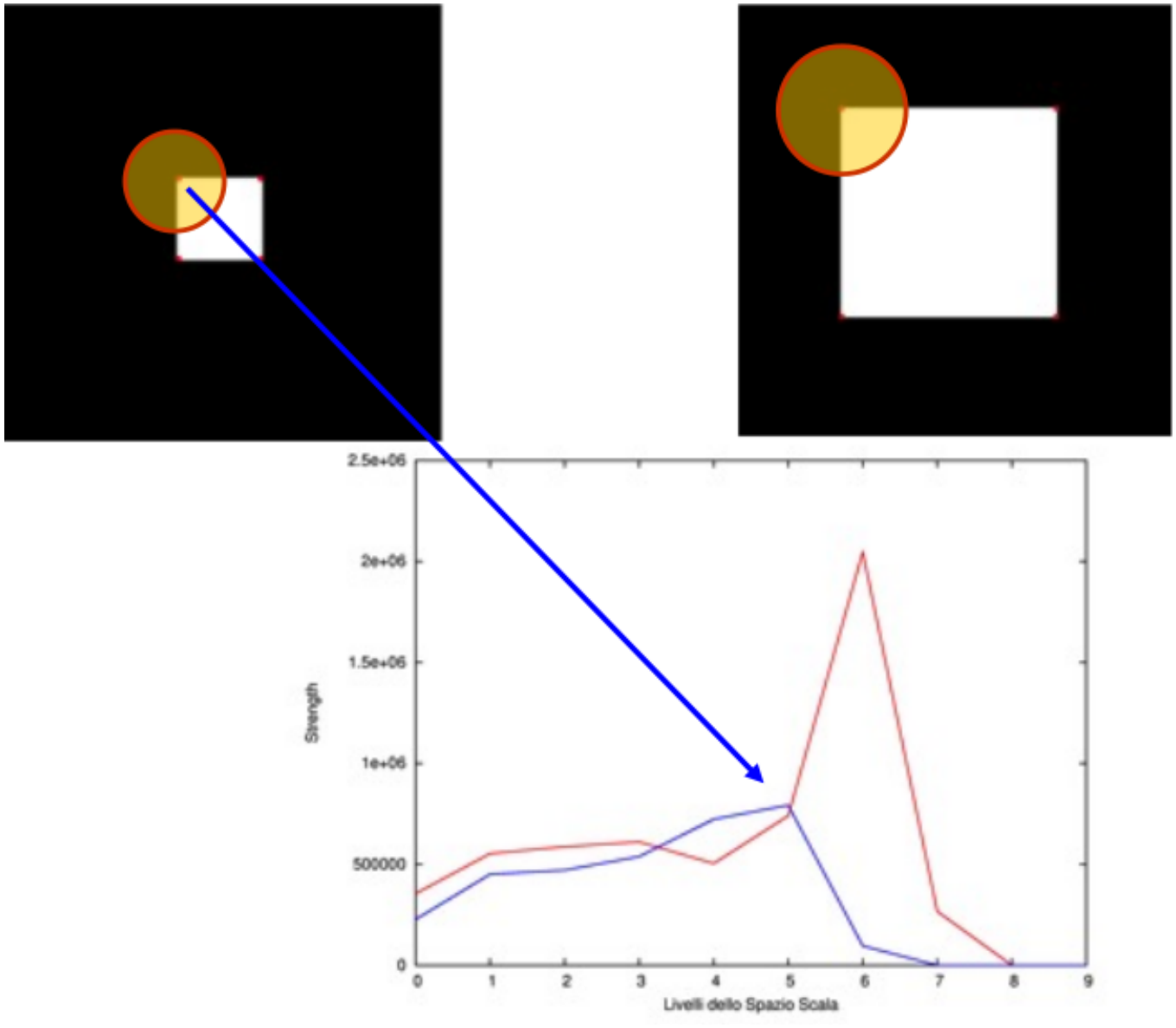
Example



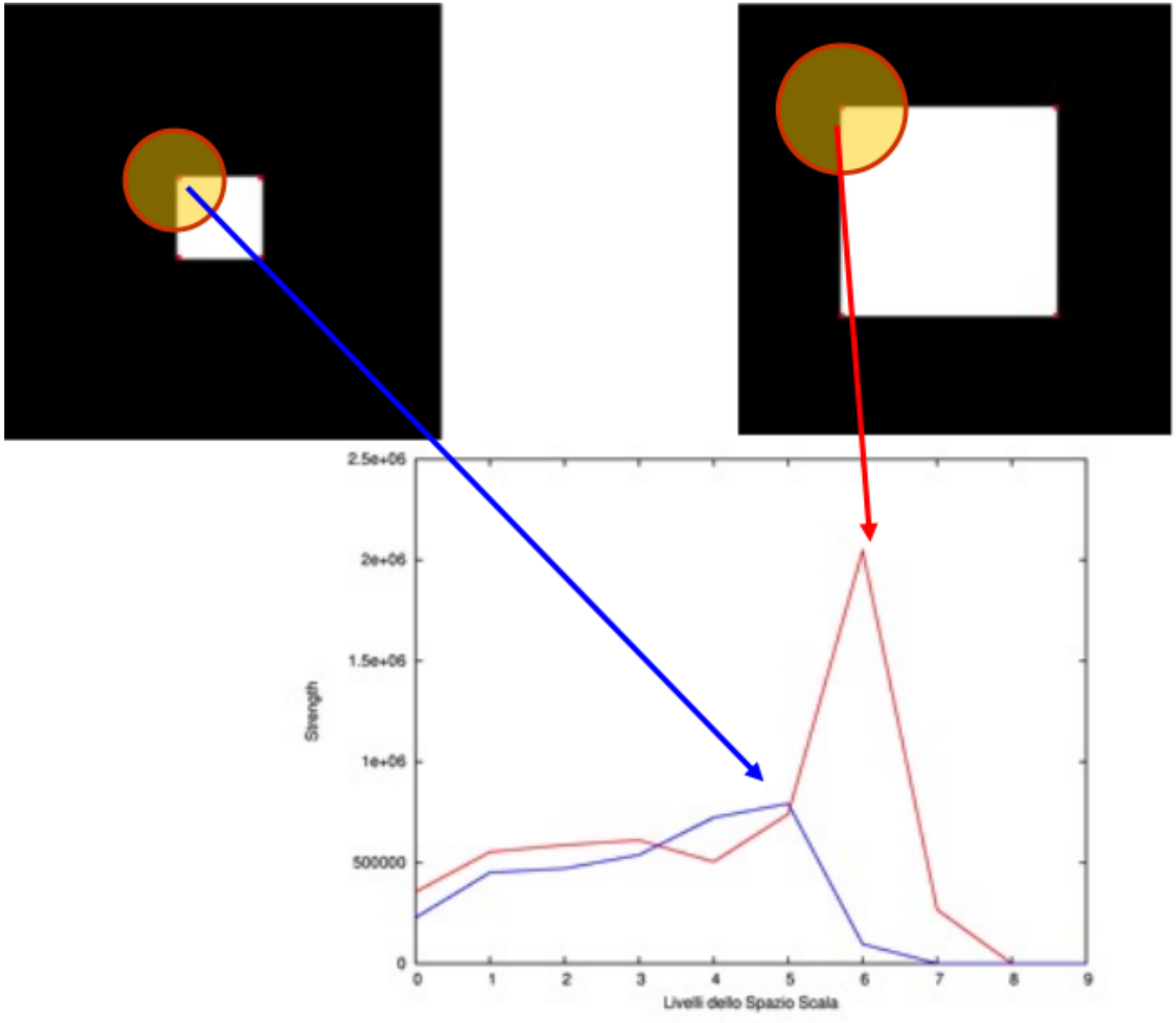



2.5e+06
2e+06
1.5e+06
1e+06
500000
0
Strength
0
1
2
3
4
5
6
7
8
9
Livelli dello Spazio Scala

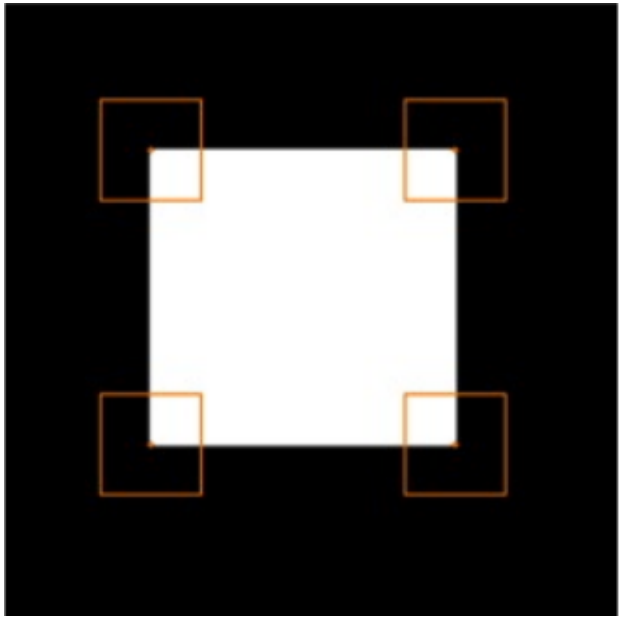
Example

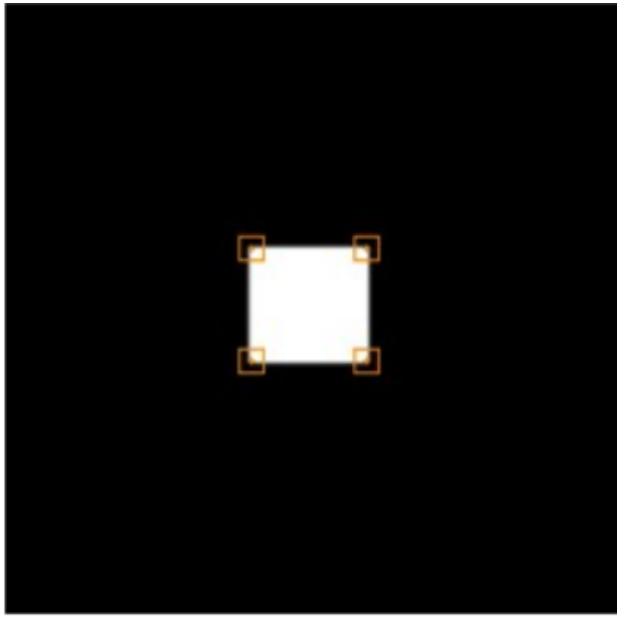

2.5e+06
2e+06
1.5e+06
1e+06
500000
0
Strength
0
1
2
3
4
5
6
7
8
9
Livelli dello Spazio Scala

Example

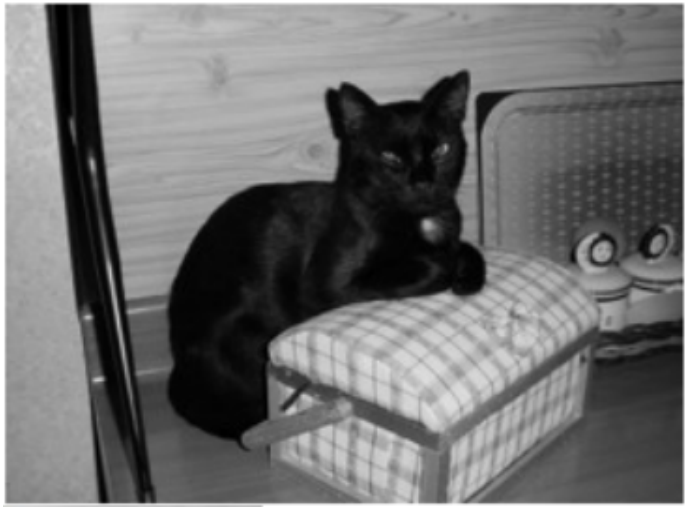

2.5e+06
2e+06
1.5e+06
1e+06
500000
0
Strength
0
1
2
3
4
5
6
7
8
9
Livelli dello Spazio Scala

Example



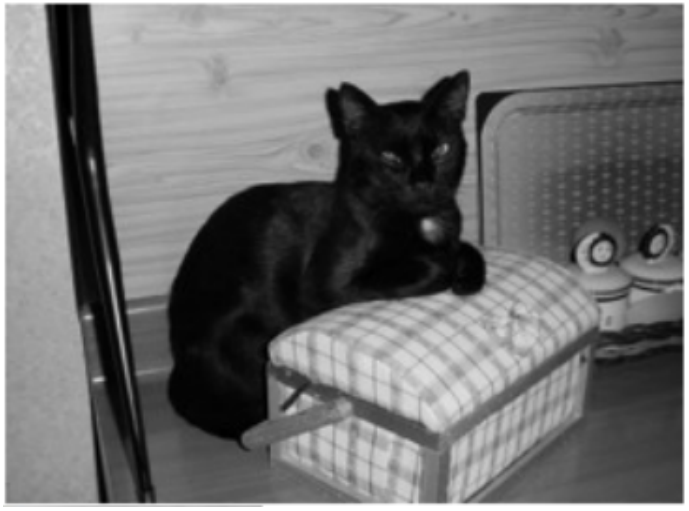


Another example

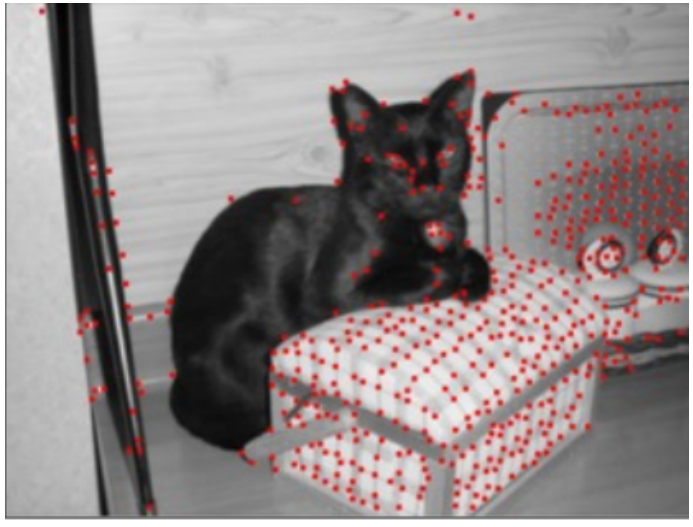


Original image

Another example

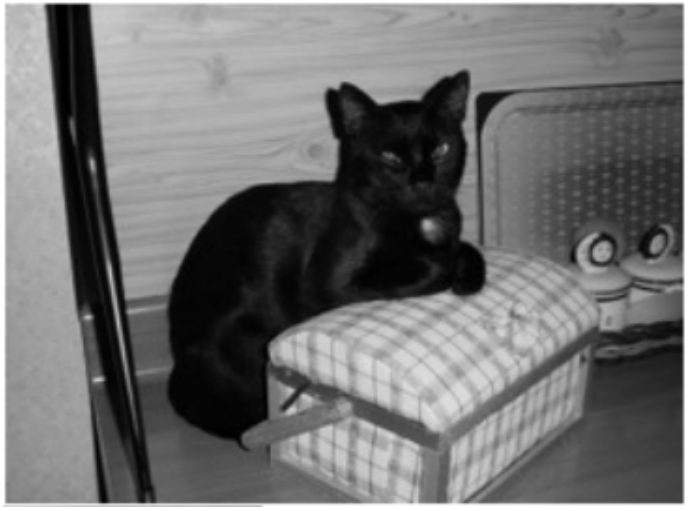


Original image



Corners at level 0

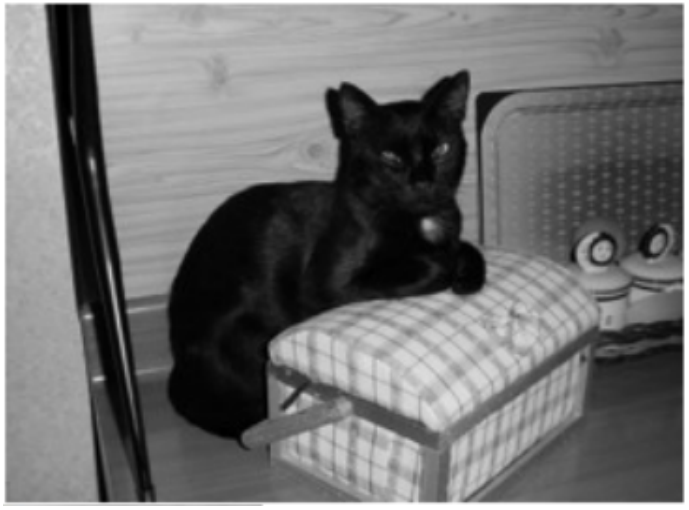
Another example



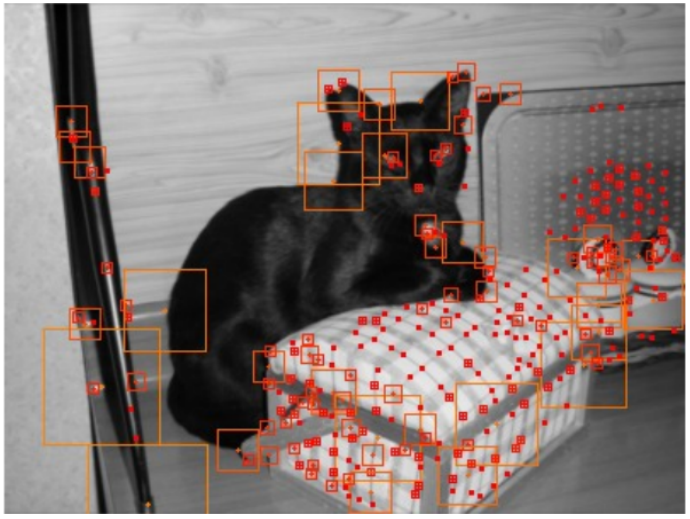
Original image

Corners at level 5

Another example



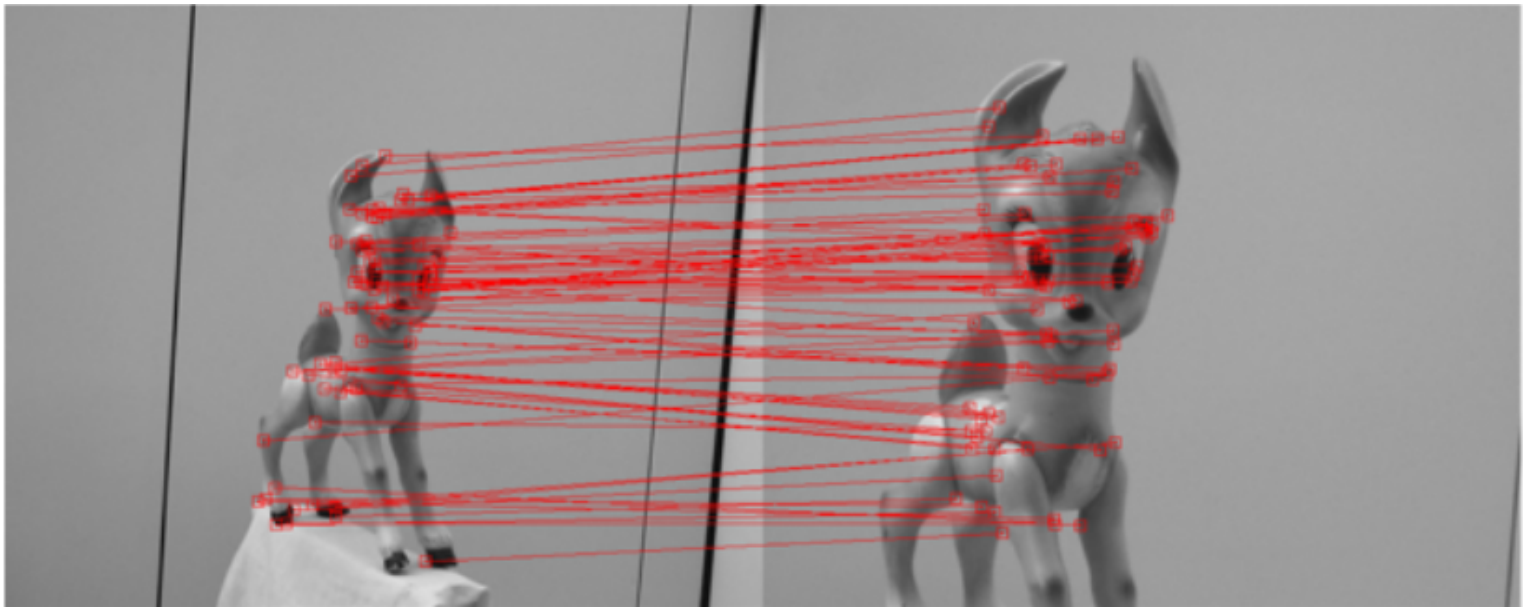
Original image



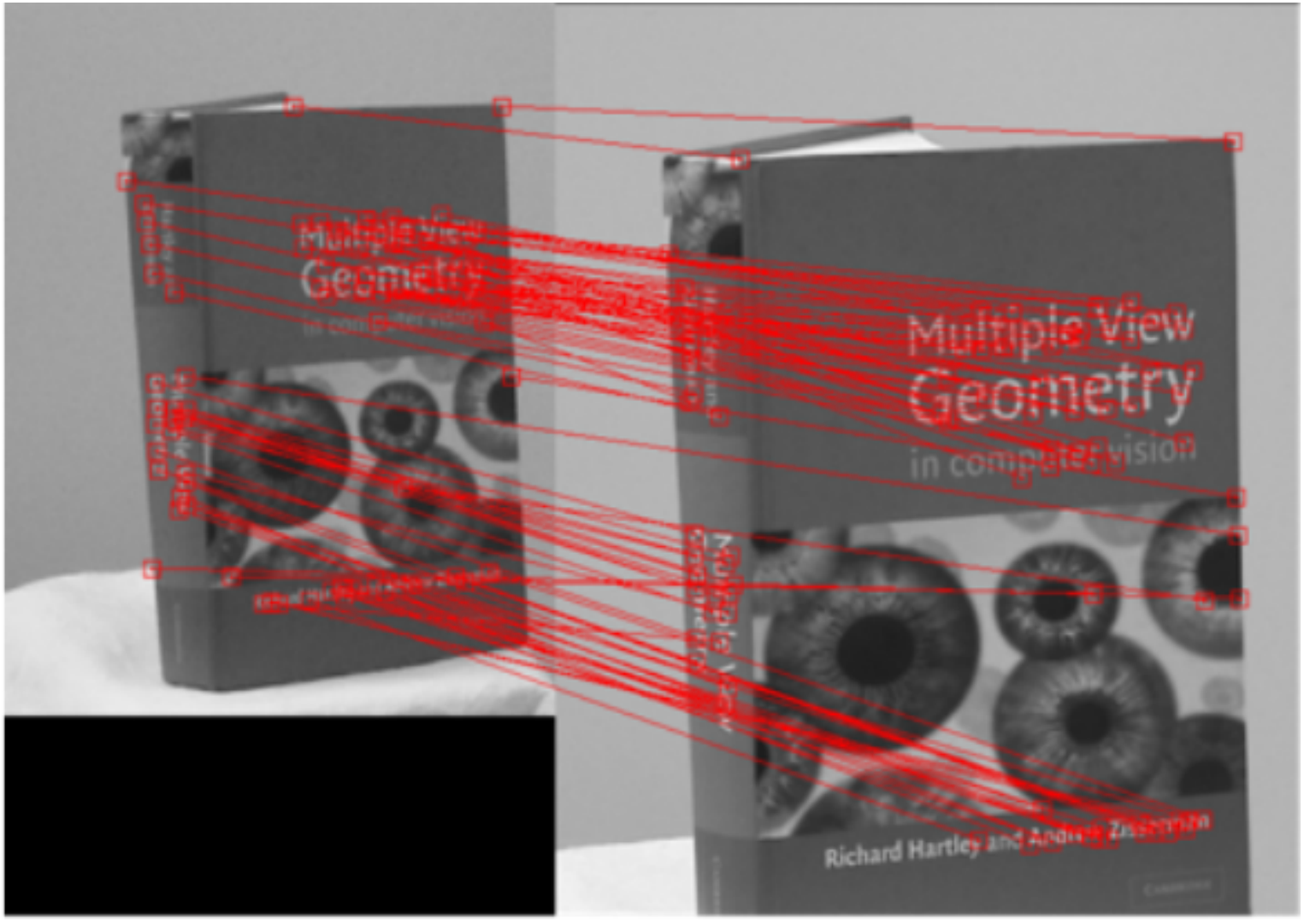
Corners extracted

Application: matching of different resolution images

Application: matching of different resolution images



Application: matching of different resolution images



SIFT features

- SIFT [Lowe 1999] are local features
- Created for object recognition tasks
- They show some invariance to
  - scale change
  - rotation
  - affine transformation
  - illumination changes
  - view point changes

Main steps

1. Determine extrema in the scale space
2. Localisation of feature points
3. Associate a direction to each feature point
4. Compute a description vector for each feature point

The DoG pyramid

The Difference of Gaussians is defined as

$$\mathcal{D}(x, y, \sigma) = \mathcal{L}(x, y, k\sigma) - \mathcal{L}(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \star \mathsf{I}(x, y)$$

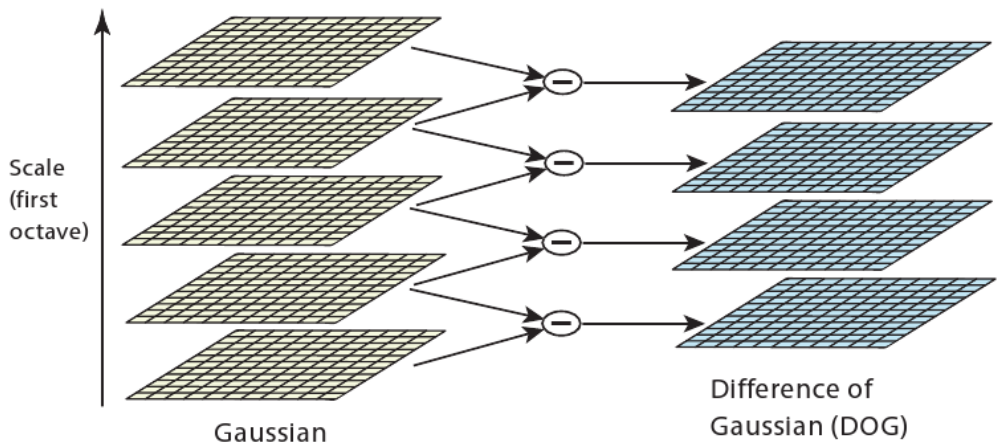
where k is a constant multiplicative factor.

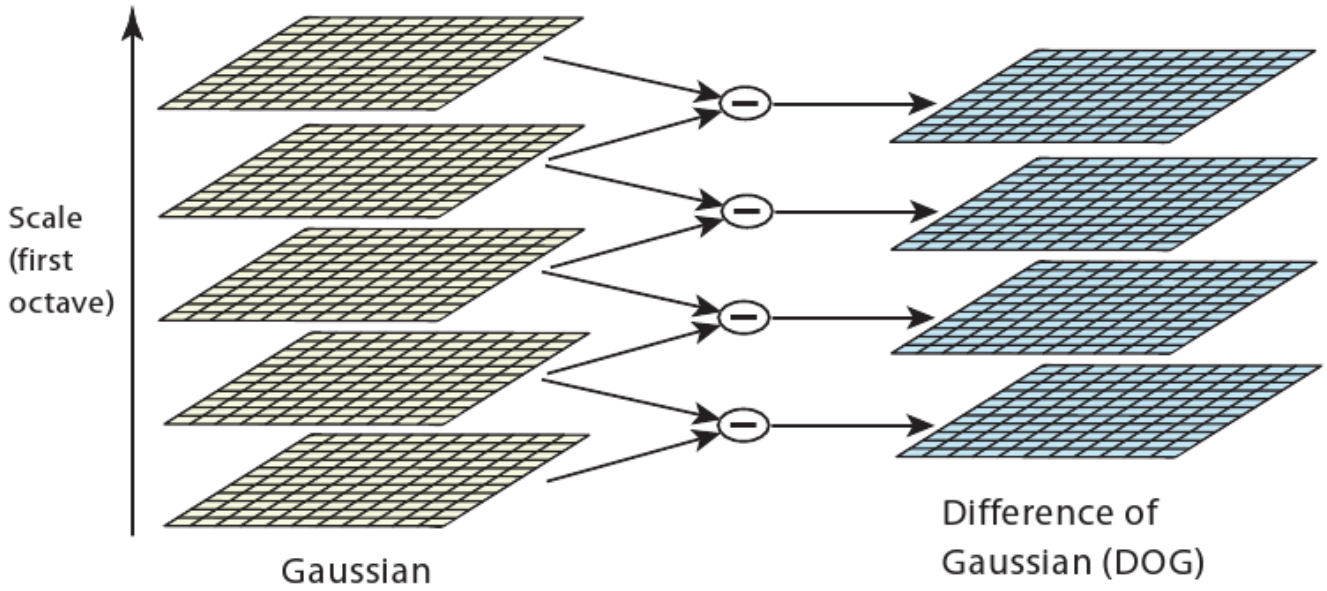
The DoG pyramid

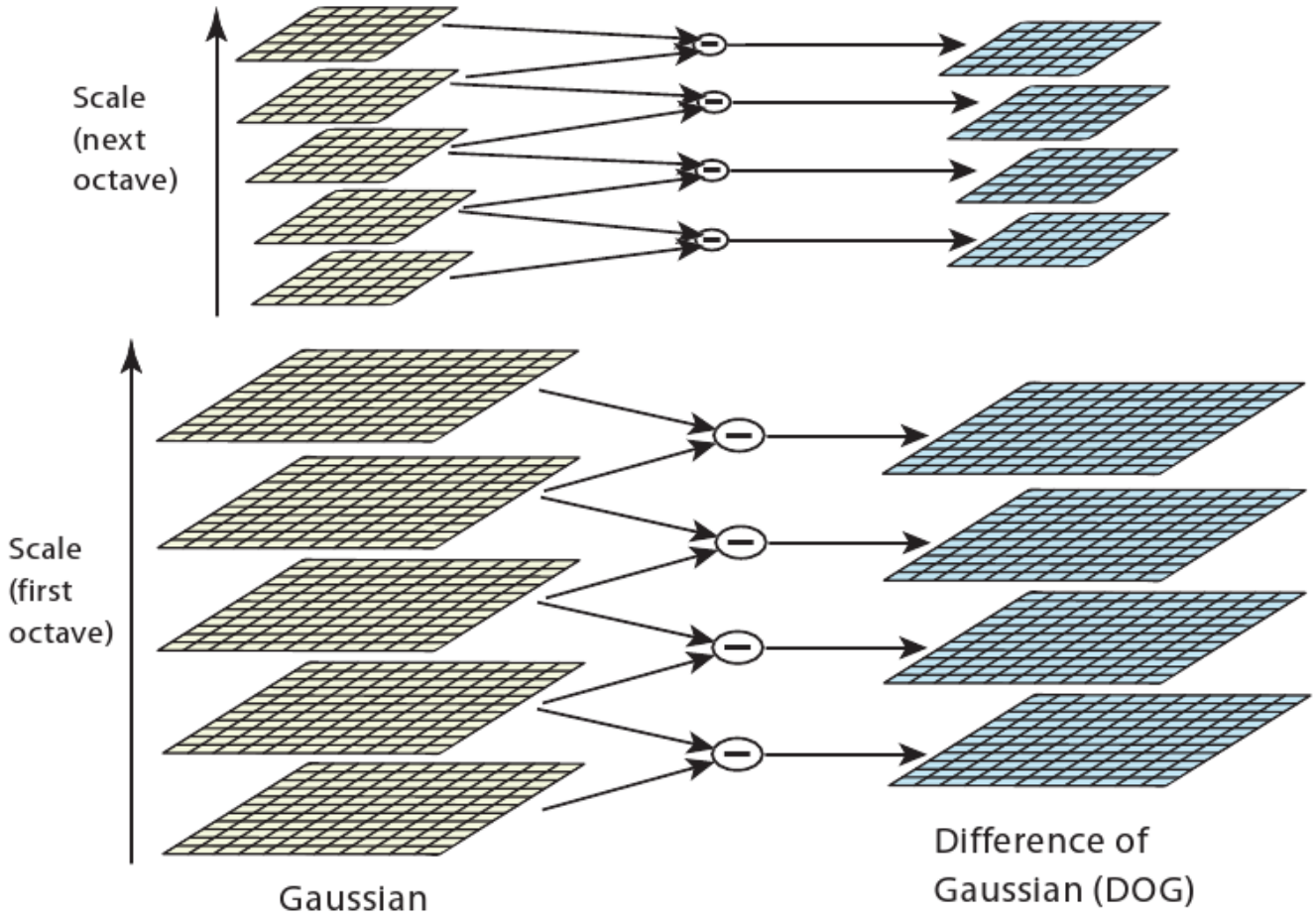
The Difference of Gaussians is defined as

$$\mathcal{D}(x, y, \sigma) = \mathcal{L}(x, y, k\sigma) - \mathcal{L}(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \star \mathsf{I}(x, y)$$

where k is a constant multiplicative factor.

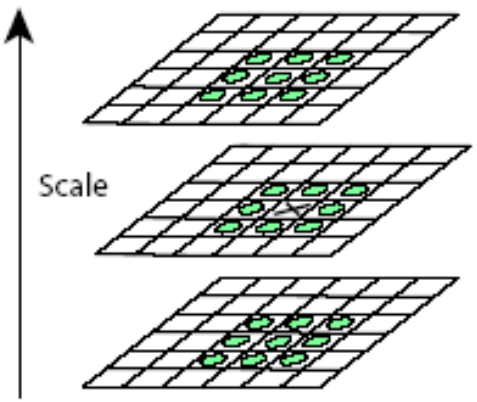



Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)


Scale
(next
octave)
Scale
(first
octave)
Gaussian
Difference of
Gaussian (DOG)

Determine extrema in DoG pyramid

- Determine local maxima or minima in $\mathcal{D}(x, y, \sigma)$
- It must be an extrema for a 3D neighbourhood



Feature point localisation

- Local extrema are candidate feature points
- Reject
  - points with low contrast
  - localised along an edge (unstable to noise and not invariant)
- More accurate localisation of points by fitting a 3D quadratic function

Eliminating edges

- α and β eigenvalues of the M from the Harris corner algorithm

$$\mathrm{M} = \begin{bmatrix} A(x,y) & C(x,y) \\ C(x,y) & B(x,y) \end{bmatrix}$$

- if (x, y) is a edge: α large and β small, or vice-versa
- $\alpha + \beta = \mathit{trace}(\mathrm{M})$ and $\alpha\beta = \mathit{det}(\mathrm{M})$
- if $\alpha = r\beta$ $(\alpha > \beta)$

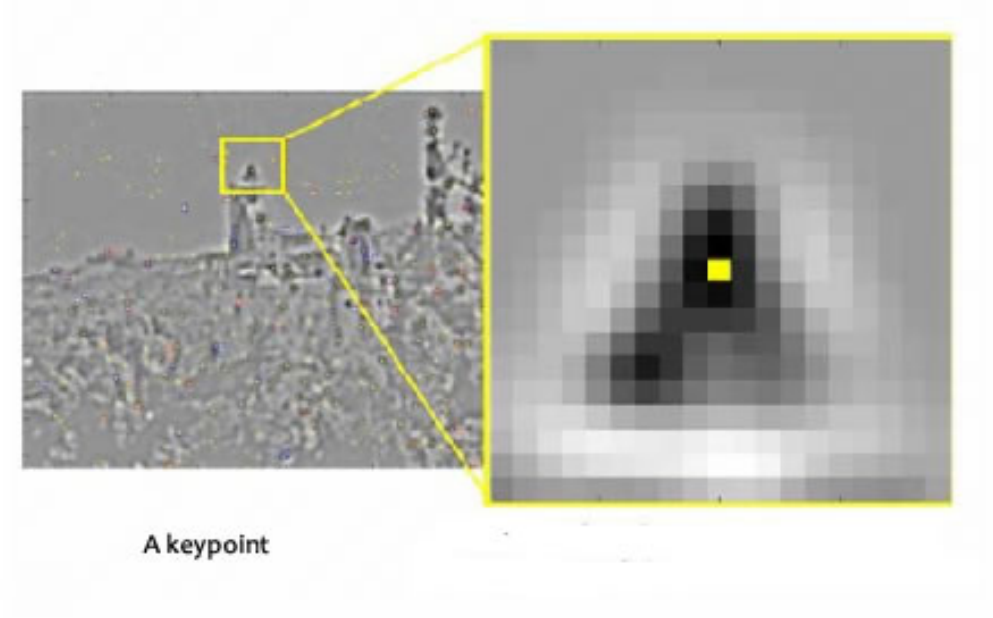
$$\frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

- $\frac{(r+1)^2}{r}$ minimum when $\alpha = \beta$ and increases with r
- We want to check when r is large, that is fixing a threshold r_0

$$\frac{\mathit{trace}(\mathrm{M})}{\mathit{det}(\mathrm{M})} < \frac{(r_0+1)^2}{r_0}$$

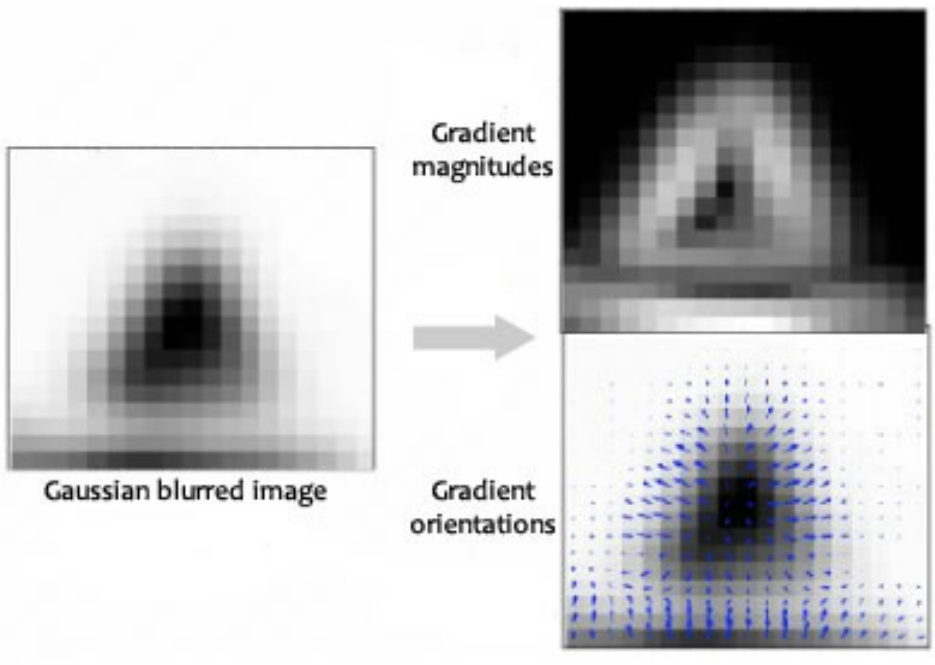
Principal direction

- Assign a principal direction to each feature point allows to have rotation invariant descriptors



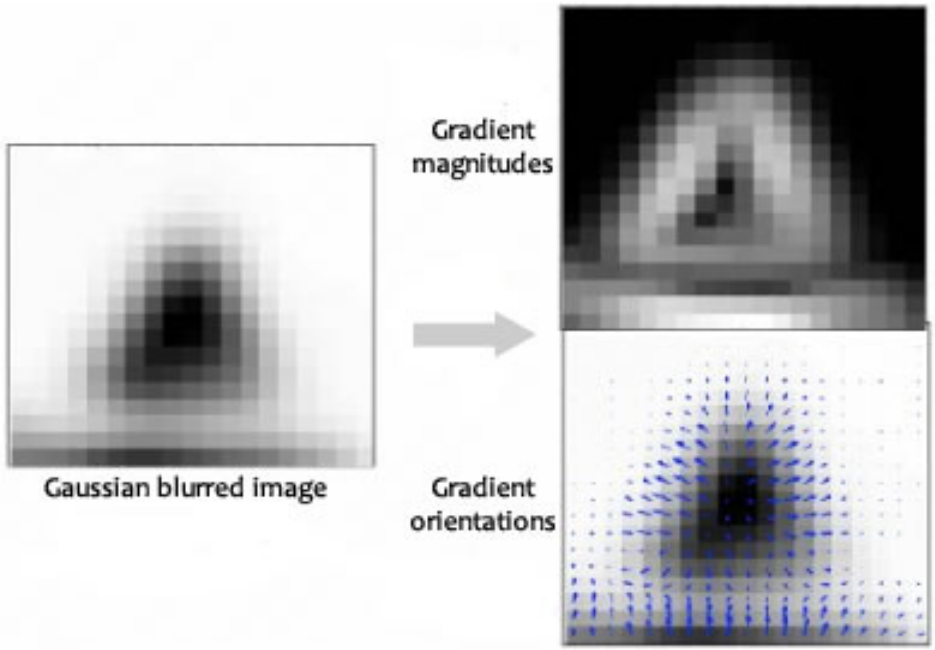
Principal direction

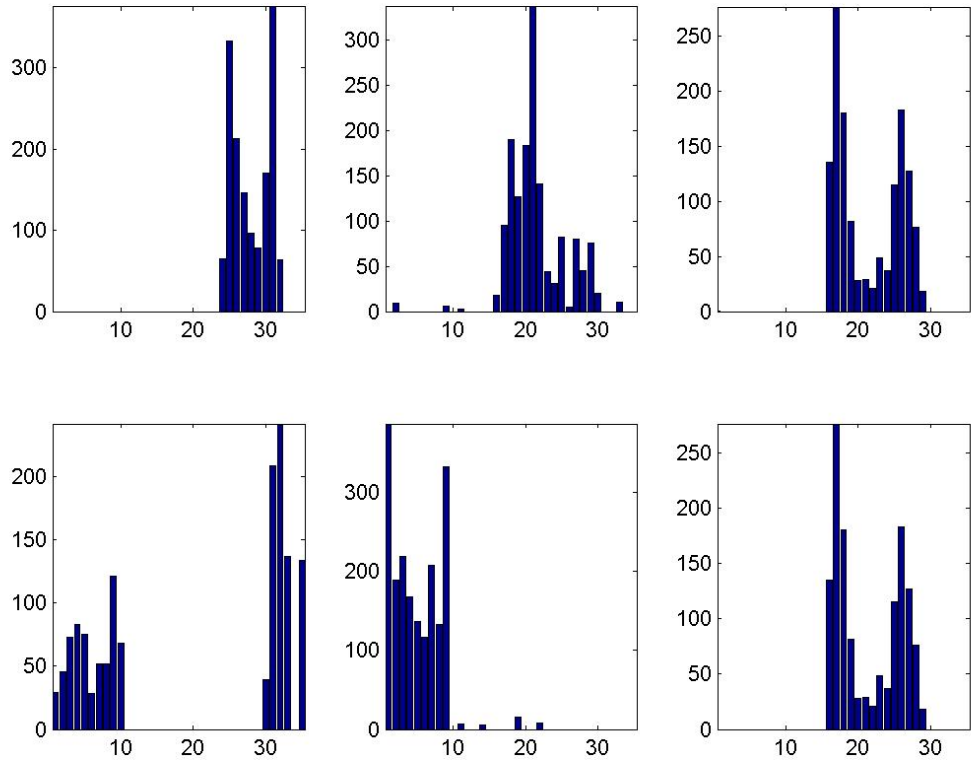
- Assign a principal direction to each feature point allows to have rotation invariant descriptors
- Compute magnitude and direction for each point in the scale space (not in the DoG)



Principal direction

- Assign a principal direction to each feature point allows to have rotation invariant descriptors
- Compute magnitude and direction for each point in the scale space (not in the DoG)
- The principal direction of a feature point is determined as a peak in the orientations histogram calculated for all pixels around the keypoint




300
200
100
0
10
20
30
300
250
200
150
100
50
0
10
20
30
250
200
150
100
50
0
10
20
30
200
150
100
50
0
10
20
30
300
200
100
0
10
20
30
250
200
150
100
50
0
10
20
30

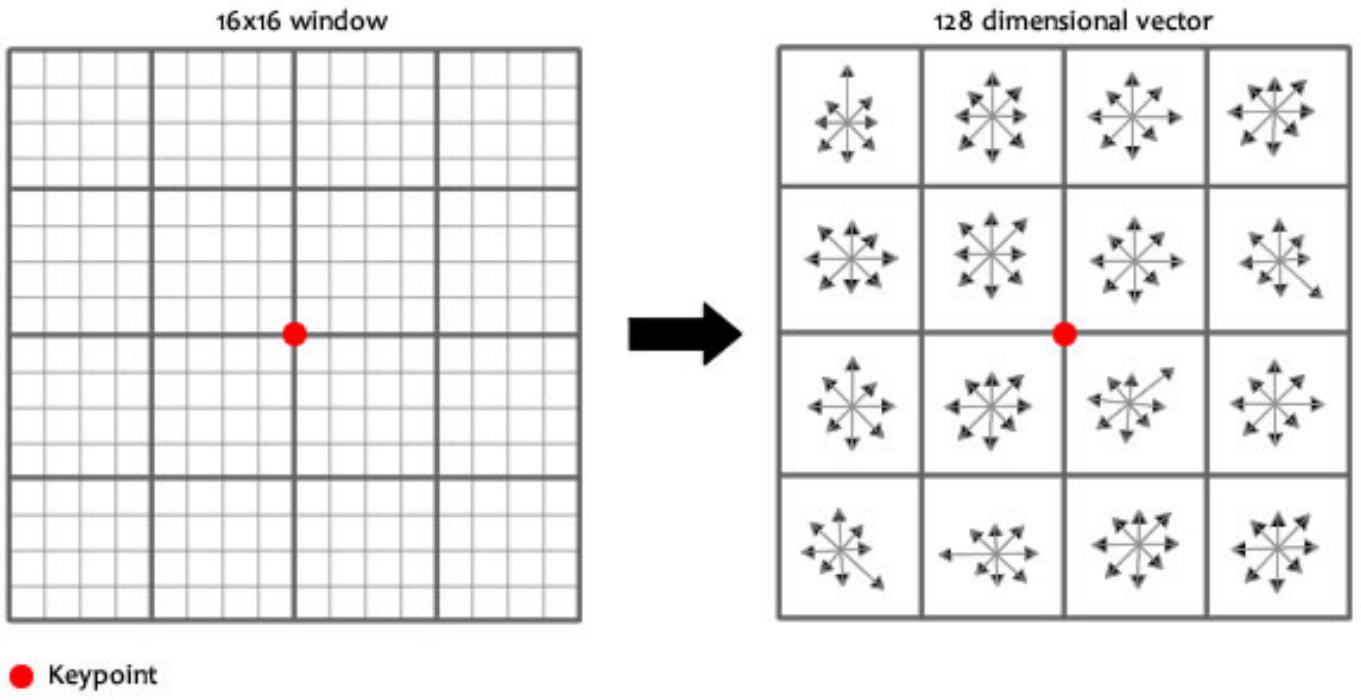
What if more than one peak?

What if more than one peak?

- any peaks above 80% of the highest peak are converted into a new keypoint
- this new keypoint has the same location and scale as the original
- its orientation is equal to the other peak
- orientation can split up one keypoint into multiple keypoints

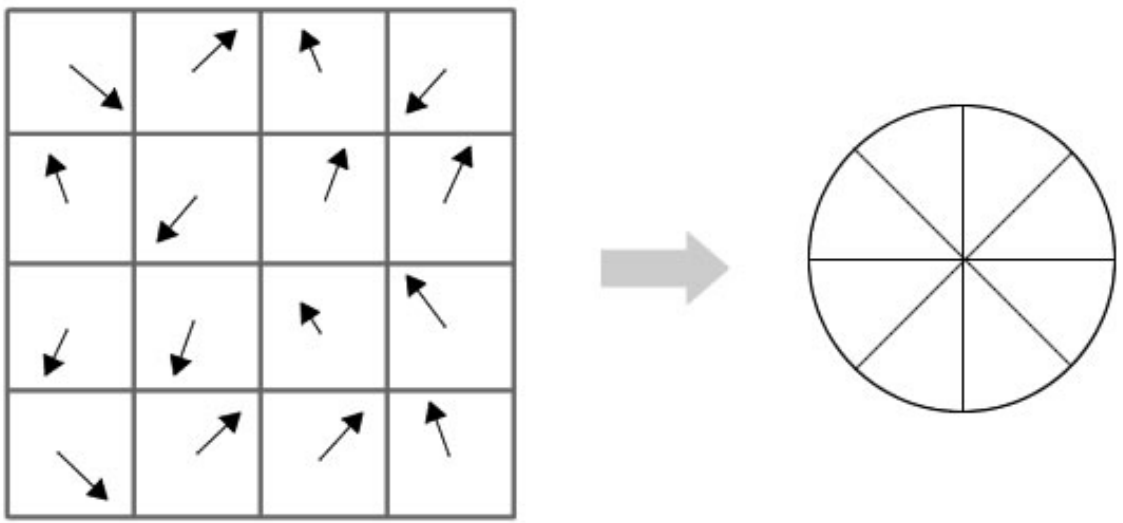
Descriptor

16×16 window around the keypoint. This window is broken into 16 4×4 windows.



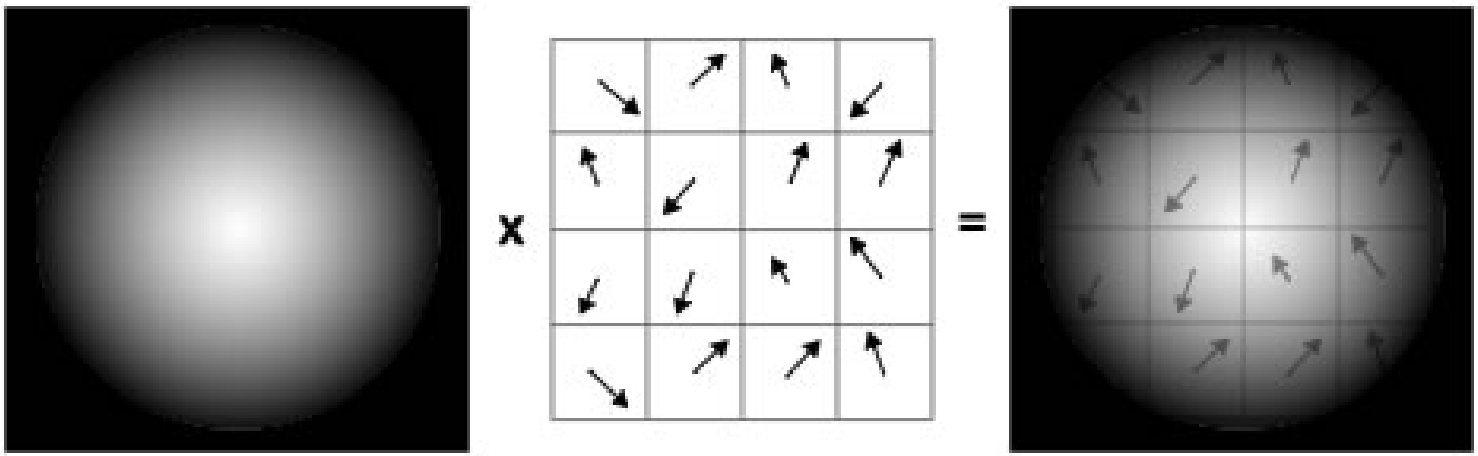
Descriptor

Within each 4×4 window, gradient magnitudes and orientations are calculated.
These orientations are put into an 8 bin histogram.
Any gradient orientation in the range 0-44 degrees add to the first bin. 45-89 add to the next bin. And so on.
The amount added to the bin depends on the magnitude of the gradient.



Descriptor

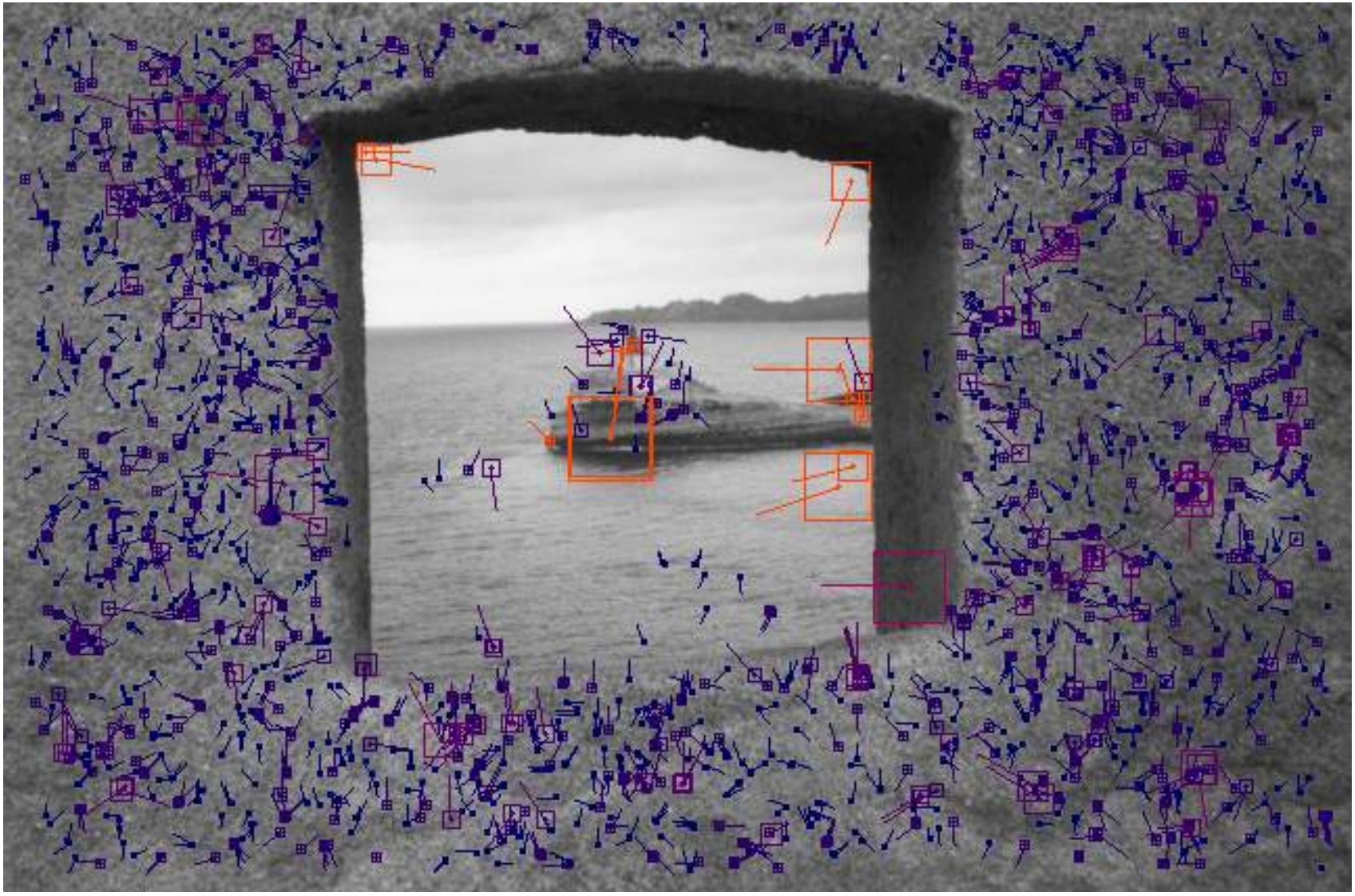
The amount added also depends on the distance from the keypoint. Gradients that are far away from the keypoint will add smaller values to the histogram.
This is done using a Gaussian weighting function.



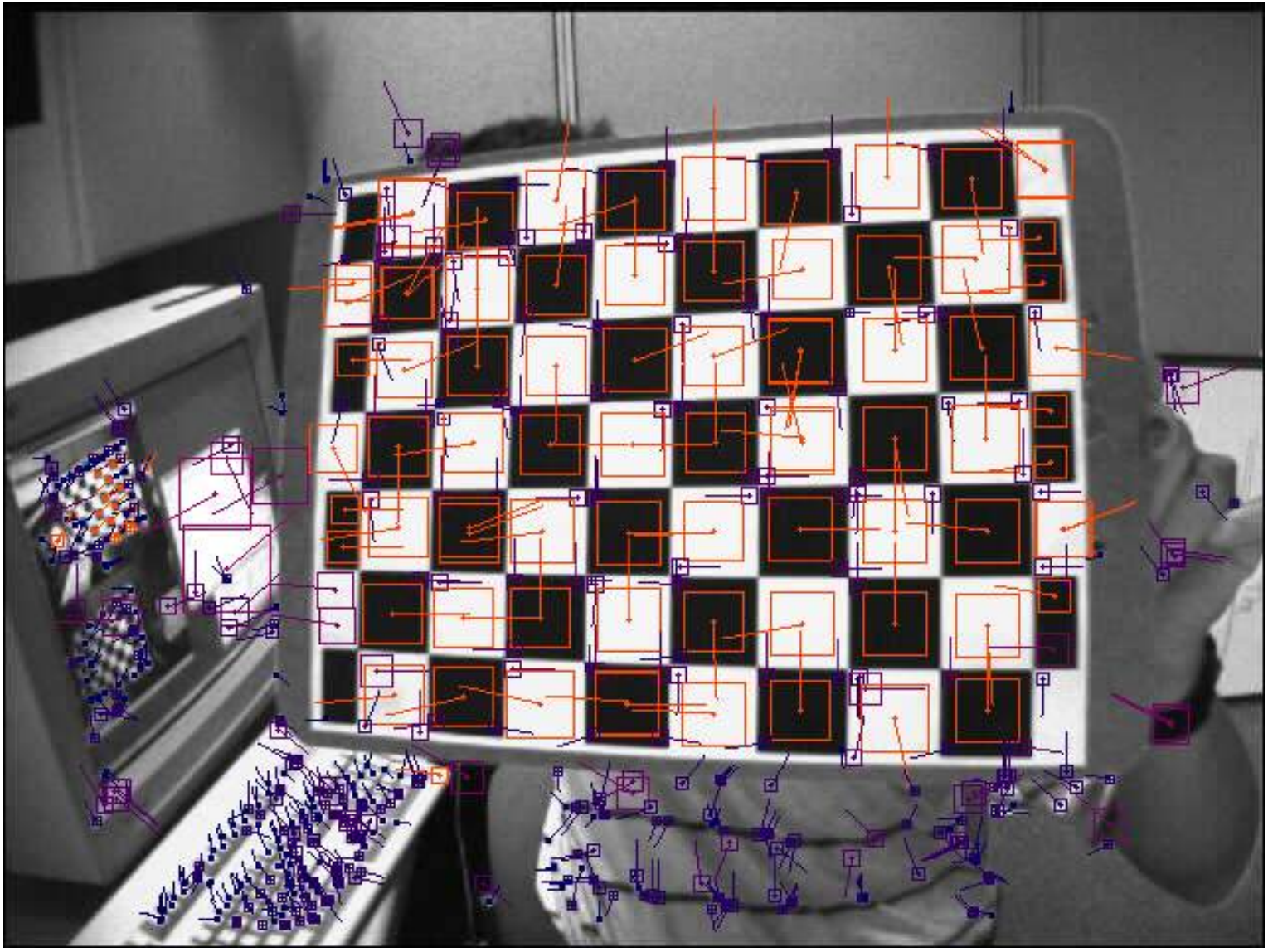
Descriptor

This is done for all sixteen 4×4 regions.
In the end we get a feature vector of $4 \cdot 4 \cdot 8 = 128$ components.
The feature vector is normalised.
The keypoint is uniquely identified by this feature vector.

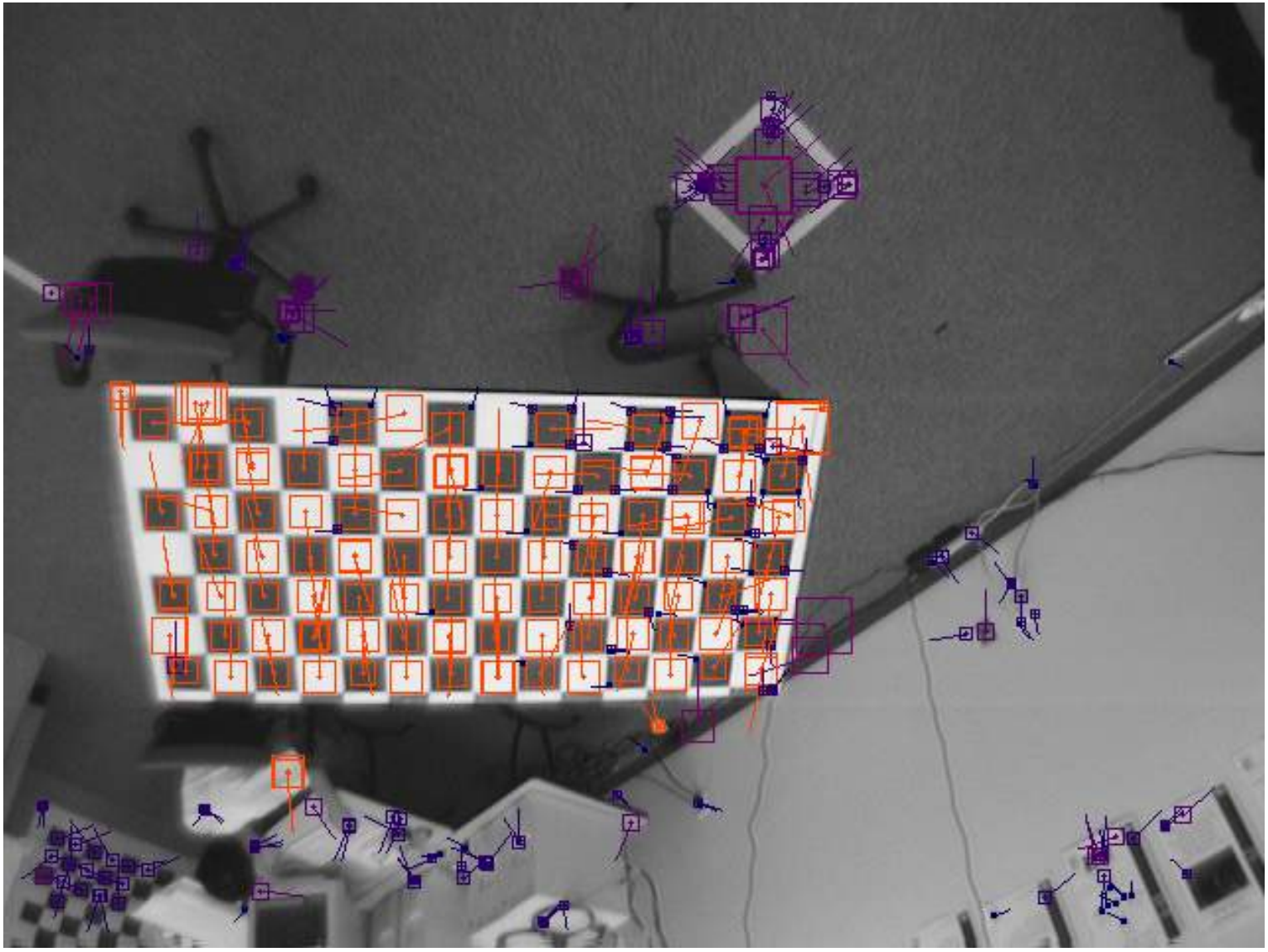
Examples



Examples



Examples



Applications

- object recognition
- robot localisation and mapping
- mosaicing
- 3D scene modelling, recognition and tracking
- analysing the Human Brain in 3D Magnetic Resonance Images
- 3D SIFT descriptors for human action recognition

Object recognition

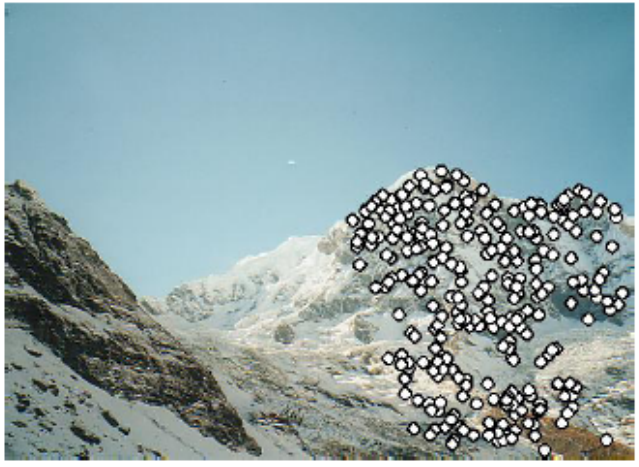
1. SIFT features are obtained from the input image
2. features are matched to the SIFT feature database
   - feature matching is done through a Euclidean-distance based nearest neighbour approach
   - matches for which the ratio of the nearest neighbour distance to the second nearest neighbour distance is greater than 0.8 are rejected
3. cluster those features that belong to the same object and reject the matches that are left out
   - Hough transform
   - clusters features that vote for the same object pose
4. Determine object using affine transform and a probabilistic method.





10
PACK
maxell
CD-R
700


10
PACK
maxell
CD-R
700



Robot localisation and mapping

- solution to the problem of robot localisation in unknown environments
- Use a trinocular system
- 3D position of SIFT points are computed
- robot localises itself using feature matches to an existing 3D map
- incrementally adds features to the map

Final remarks on SIFT

- SIFT are robust features.
- Relatively easy to compute
- Very slow
- New SIFT-like features proposed
  - PCA-SIFT
  - Speeded UP Robust Feature (SURF)
- Speed improved using exploiting GPUs