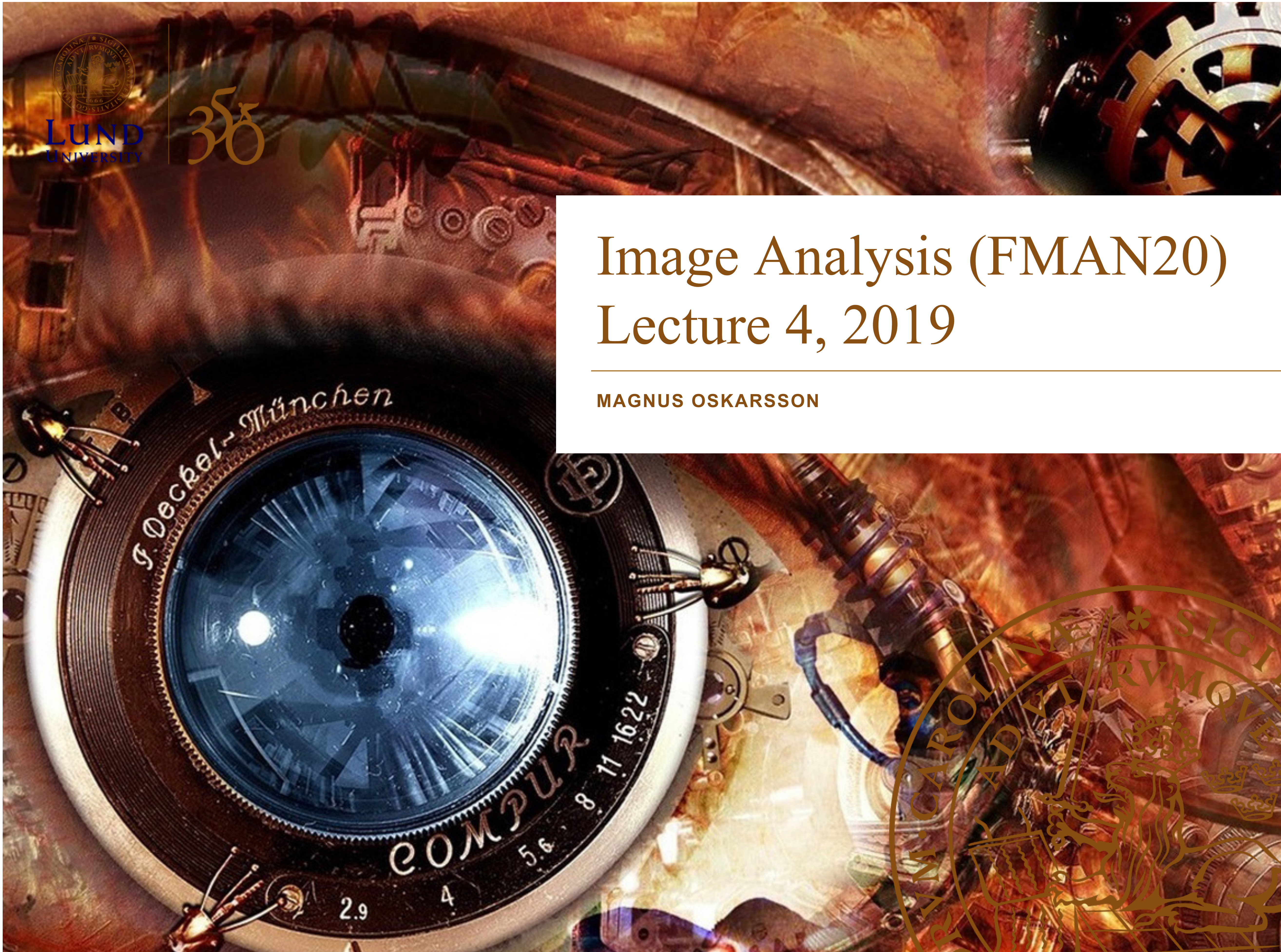
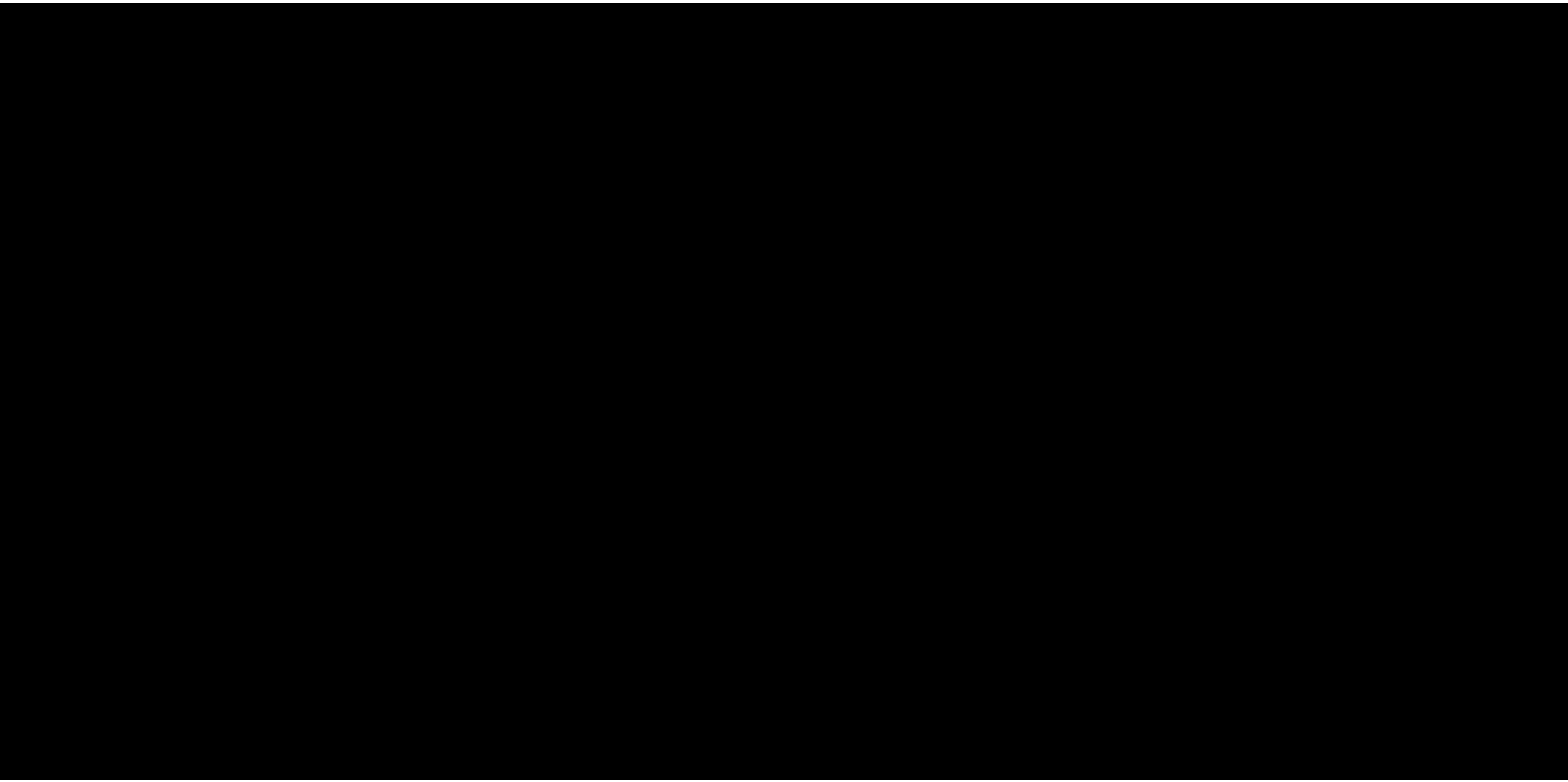


Image Analysis (FMAN20)
Lecture 4, 2019

MAGNUS OSKARSSON

Features - Motivation



ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras
Raul Mur-Artal, Juan D. Tardos

Overview –Feature Detection

- Techniques
  - Scale space theory
- Detectors
  - Edges
  - Ridges
  - Corners
  - SIFT
- Texture

Image data reduction

Image data reduction

Reduce number of colors (10 in this case) using clustering
(We will talk about clustering later on in the course)

Image data reduction

Half-toning can be used to print and send photographs.
The local intensity is coded using different sized blobs.

Image data reduction

Line drawings capture much of the content of an image.
How can we extract lines and edges from an image?

Local Features

- Goal: Find a low-dimensional description of image content
- Edges
- Corners
- Other features

Overview –Feature Detection

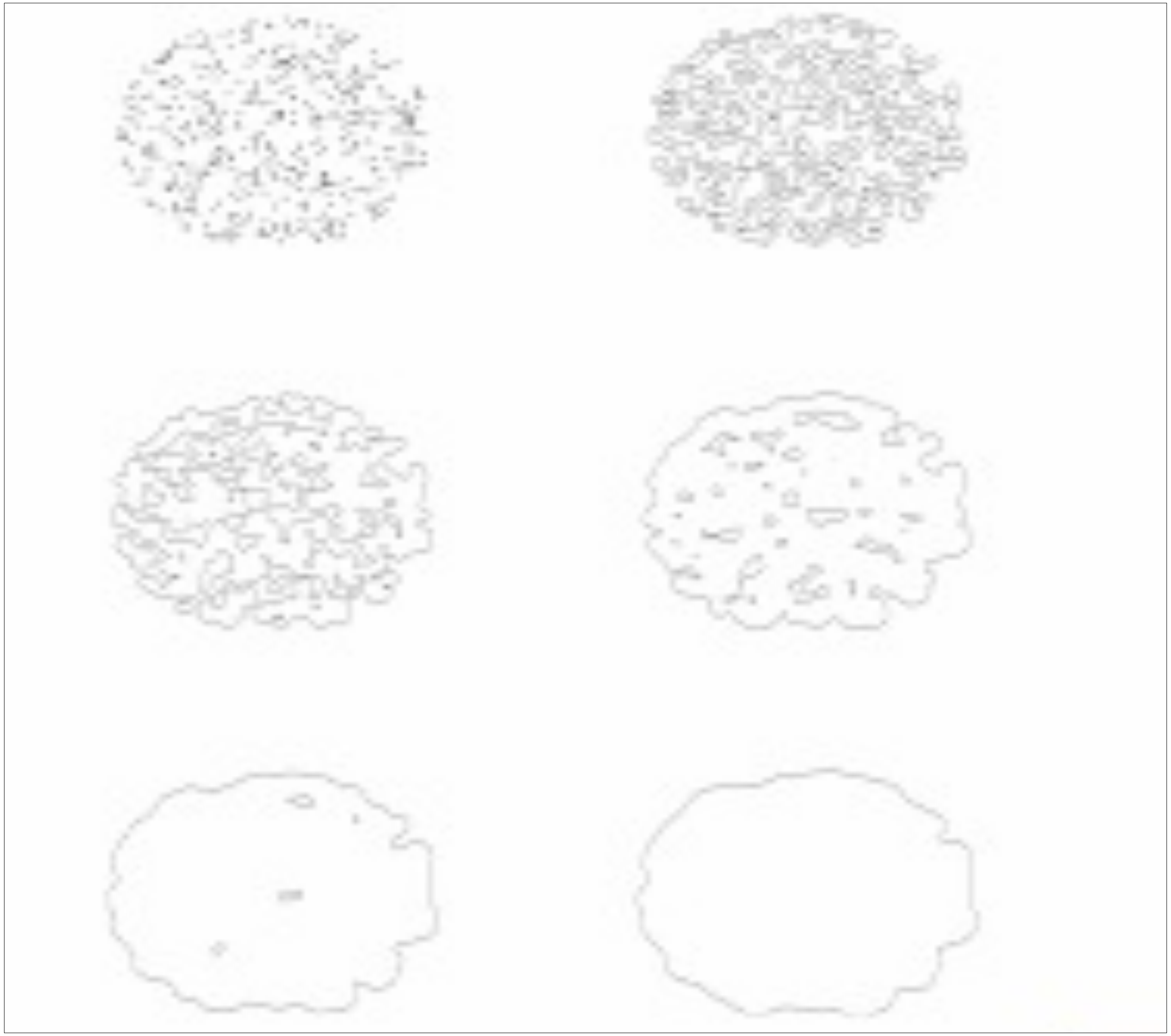
- Techniques
  - **Scale space theory**
- Detectors
  - Edges
  - Ridges
  - Corners
  - SIFT
- Texture

Scale Space

Example: What is a cloud?

- something in the sky
- Regions in the atmosphere, where the density of condensed H_2O is above $0.4gm^{-3}$ at a resolution of about 1 m.

Scale Space



Scale Space

Principal of causality
If $V_2 > V_1$ then $d(x, V_2)$ can be calculated from $d(x, V_1)$ but not vice versa.
We can go from a finer scale to a courser scale but not the other way!

Scale Space

The idea behind scale space theory is to every function $f : \mathbb{R}^n \to \mathbb{R}$ associate a family $\{T_t f | t \geq 0\}$ of gradually smoothed functions

$$T_t f : \mathbb{R}^n \to \mathbb{R} \quad .$$

The original signal corresponds to scale $t = 0$. Increasing scale simplifies the signal but should not introduce new features (e.g. new local minima or maxima).

Scale Space

Definition

The **Gaussian kernel** in two dimensions is defined as

$$G_b(x) = \frac{1}{2\pi b^2} e^{-|x|^2/2b^2}, \qquad x \in \mathbb{R}^2 .$$

Definition

The **Gaussian scale space** corresponding to the function $f : \mathbb{R}^2 \to \mathbb{R}$ is a family of functions $\{T_t f | t \geq 0\}$ parameterized by the variable t, where

$$T_t f = f * G_{\sqrt{t}} \ .$$

Scale Space

Theorem

An operator T_t with the following properties

- T_t *is a* **linear** *and* **translation invariant** *operator for every* t,
- **Scale invariance**. *If a function is scaled with a factor* λ, *i.e.* $g(x) = f(x/\lambda)$ *then there exists a scale* $t' = t'(t, \lambda)$ *such that* $T_t g(x) = (T_{t'} f)(x/\lambda)$,
- **Semi group property**: $T_{t_1}(T_{t_2} f) = T_{t_1+t_2} f$,
- **Positivity preserving**: $f > 0 \;\Rightarrow\; T_t f > 0$,

is given by

$$T_t f = f * G_{\sqrt{t}} \quad .$$

Scale Space

What does

$$f_t = T_t f_0 = f_0 * G_{\sqrt{t}} \ ?$$

There is no image with infinite resolution, i.e. the image at scale 0, f_0.
The only information we have about the image is an observation at one scale t_0, i.e. f_{t_0}.

Two popular uses of Scale Space

- **The coarse to fine principle**. In many applications it is useful to first search through the image on a coarse scale and then refine the search on a finer scale in the most interesting regions.
- **Scale space analysis:** Many features (e.g. edges) can be defined on all scales. Using the whole scale space representation one can construct robust detectors. Often features are detected on a coarser scale and positioned more precisely on a finer scale.

Scale Space Pyramid

- Fast implementations can be made using scale space pyramids
- After scale space smoothing one does not need to save all pixels and can subsample the image, usually in steps of two.

512 256 128 64 32 16 8

A bar in the big images is a hair on the zebra's nose; in smaller images, a stripe; in the smallest, the animal's nose

Overview –Feature Detection

- Techniques
  - Scale space theory
- Detectors
  - **Edges**
  - Ridges
  - Corners
  - SIFT
- Texture

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
- This is where most information in an image is coded
- **Example:** line drawings

What causes an edge?

- Depth discontinuity
- Surface orientation discontinuity
- Changes in surface properties
- Light discontinuities (e.g. shadows)

Edge detection

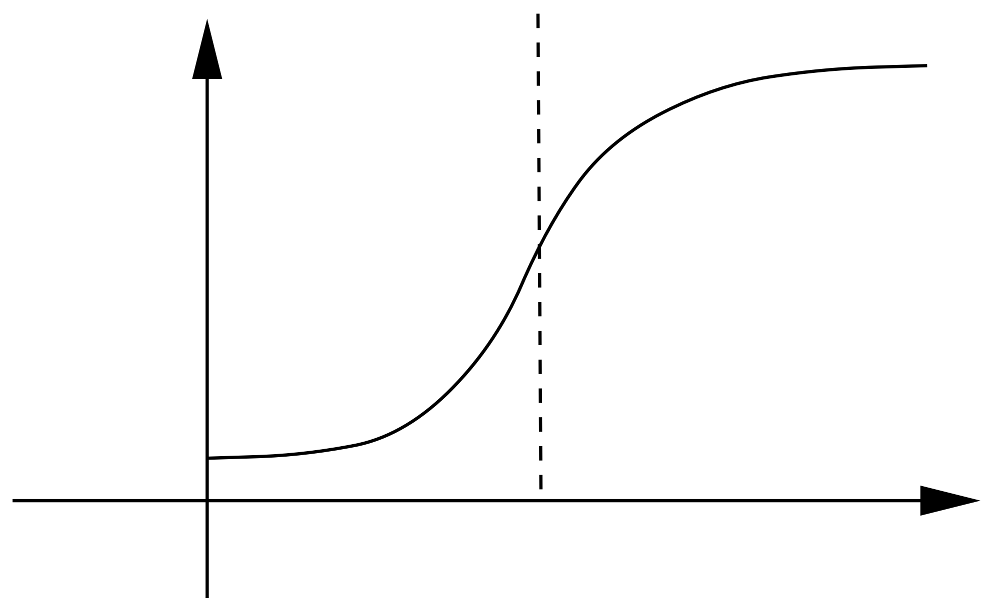
Edge detection is based on finding points in the image, where the first order derivatives are large.

Two main approaches

- Find points where the second derivative (in some sense) is zero (Laplacian methods).
- Find points where the first derivative is large (gradient

Laplacian methods

Define the edge as the inflexion point. $\Leftrightarrow$ second derivative $= 0$



Find zeros of $\Delta f = 0$ or to $\Delta G_a * f = 0$, where G_a is the Gaussian function.

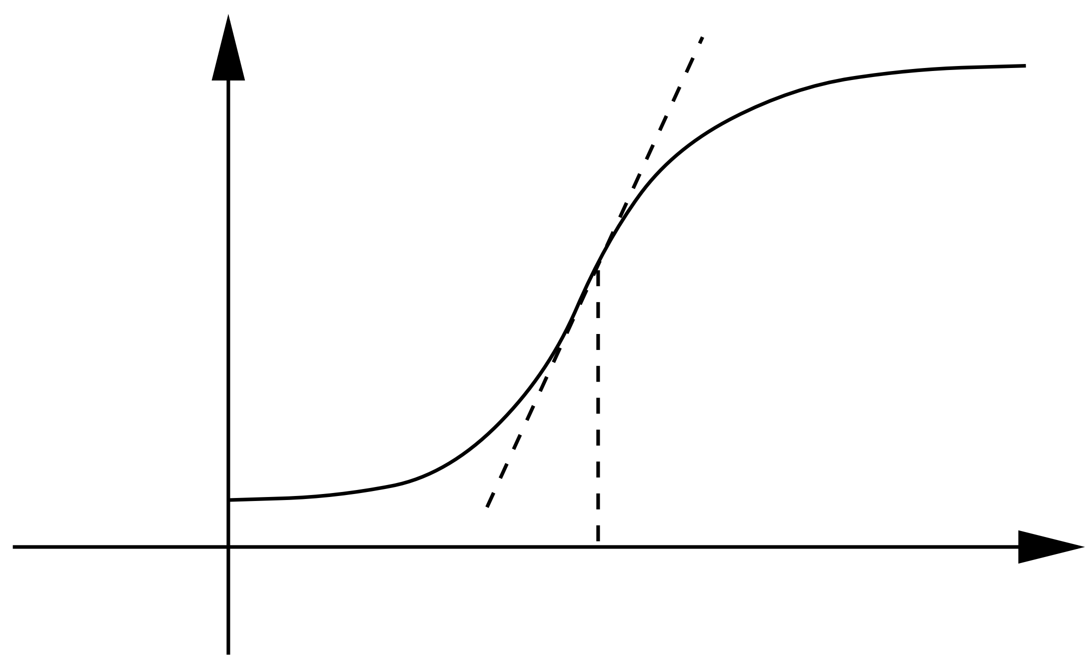
Laplacian methods

Laplacian methods have been used, but they have several disadvantages

- The Laplace filter is un-oriented
- The result is sometimes strange at sharp corners
- The result is strange where 3 or more intensities/colours intersect

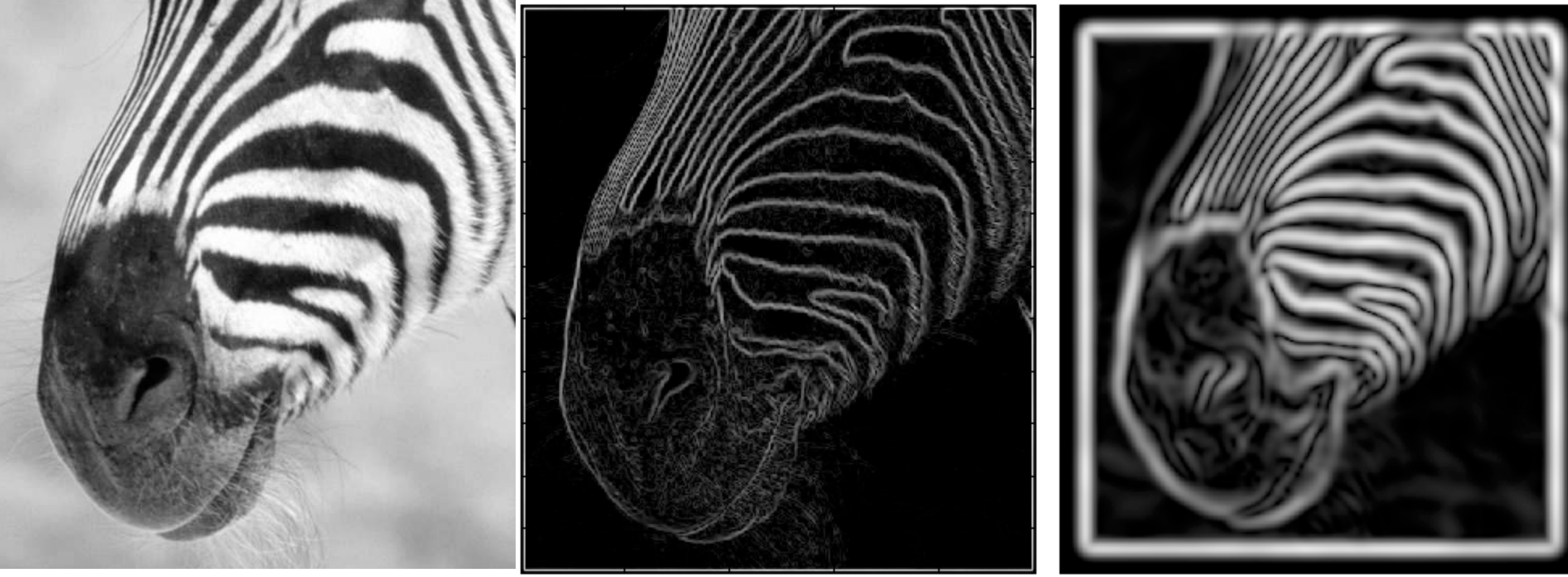
Gradient methods

One dimension:



Model of an edge: *Maximum of derivative = position of edge*
Two dimensions: Use discrete approximation of

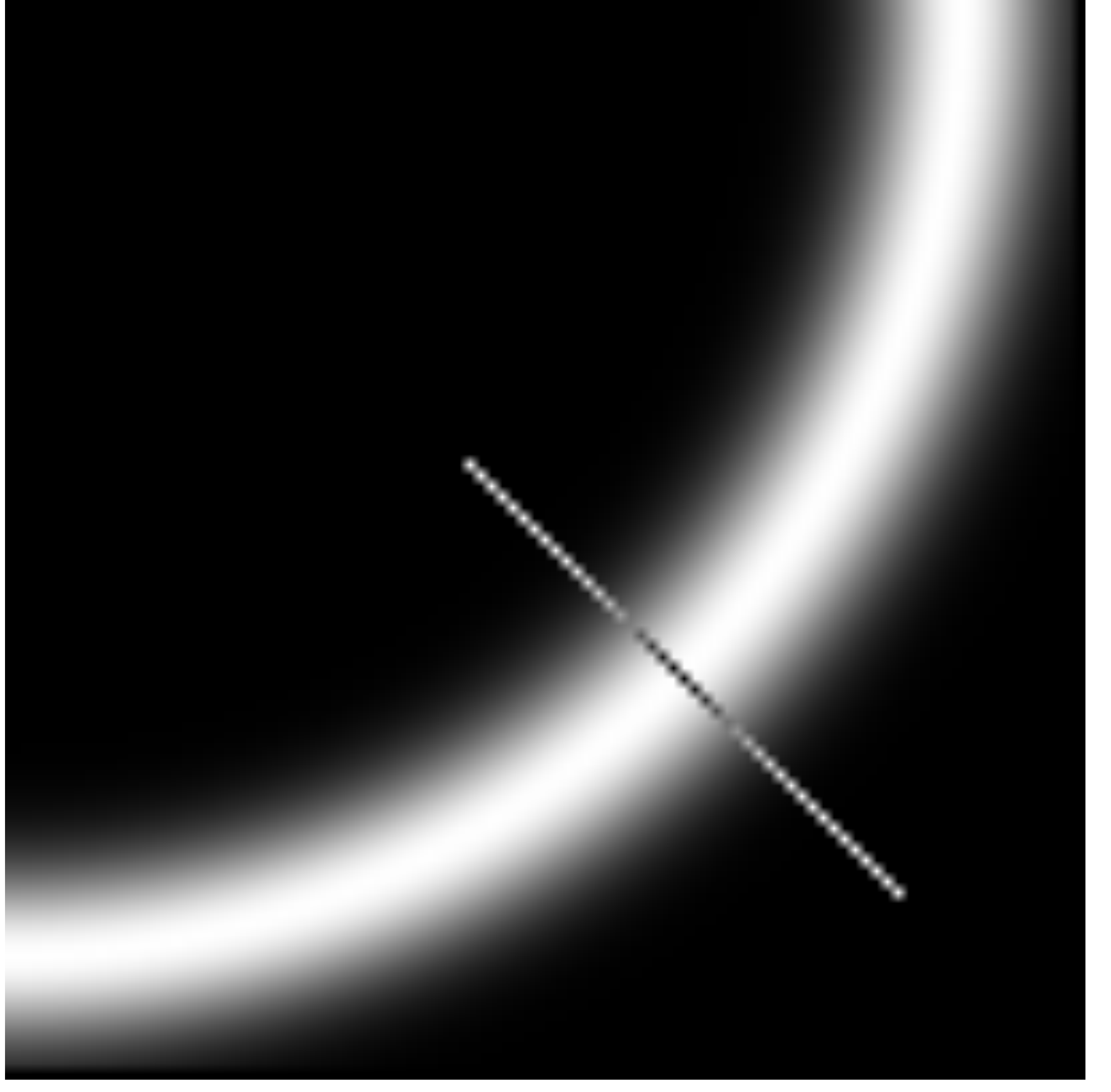
$$\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 = |\nabla f|^2 .$$



Scale

Increased scale:

- Eliminates noisy edges
- Makes edges smoother and thicker
- Removes fine details



Suppression of non-maxima:

Choose the local maximum point along a perpendicular cross section of the edge.

Example: Suppression of non-maxima

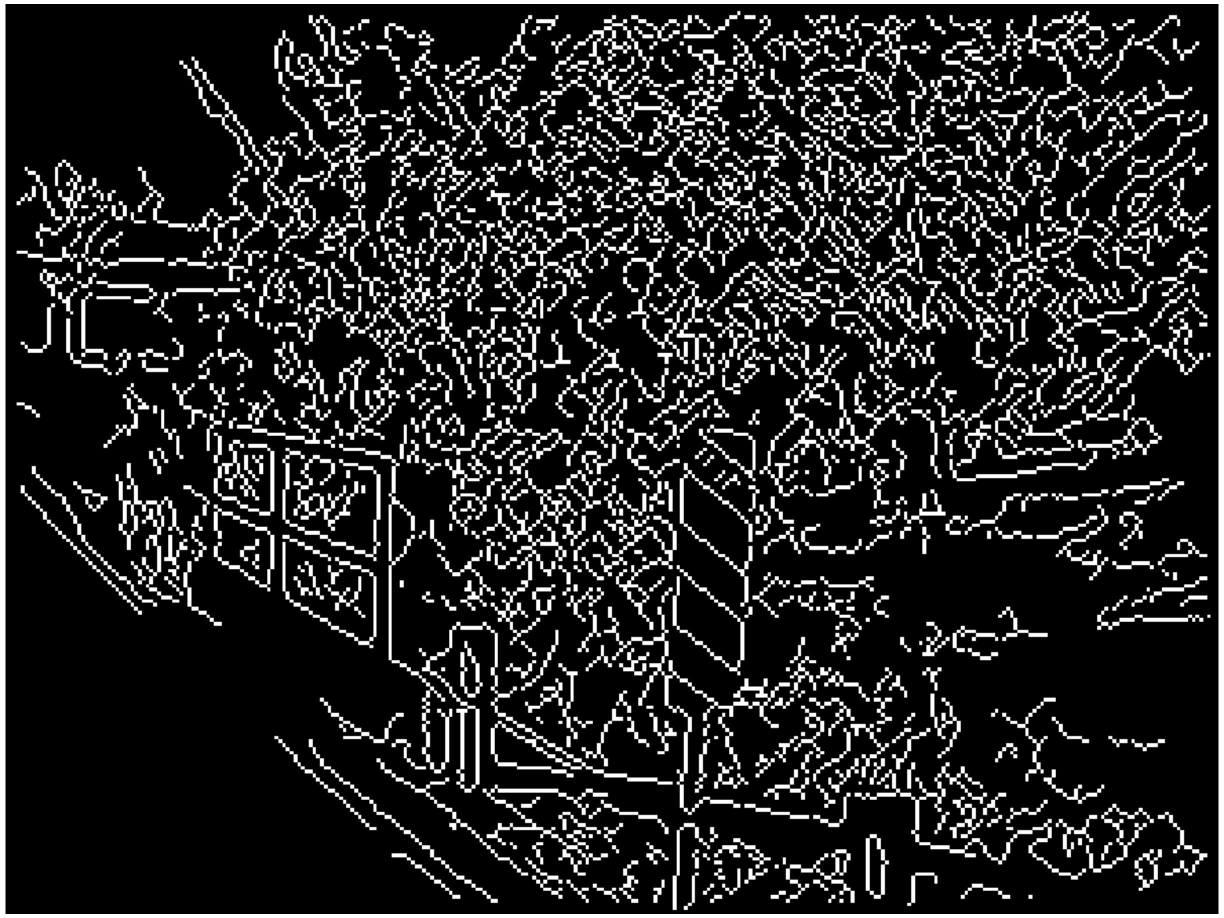
courtesy of G. Loy

Original image

Gradient magnitude

Non-maxima suppressed

Example: Canny Edge Detection



Using Matlab with default thresholds

Overview –Feature Detection

- Techniques
  - Scale space theory
- Detectors
  - Edges
  - **Ridges**
  - Corners
  - SIFT
- Texture

Ridge detection

Example from Masters thesis project in medical image analysis

Calculated smoothed second derivatives

$$\frac{\partial^2 G_a}{\partial x^2} * f$$

Ridge detection

Different scales (smoothing) is used to find ridges of different scales (widths)
The second derivatives in an arbitrary direction can be calculated from a combination of the three second order derivatives.
Compare with gradient.

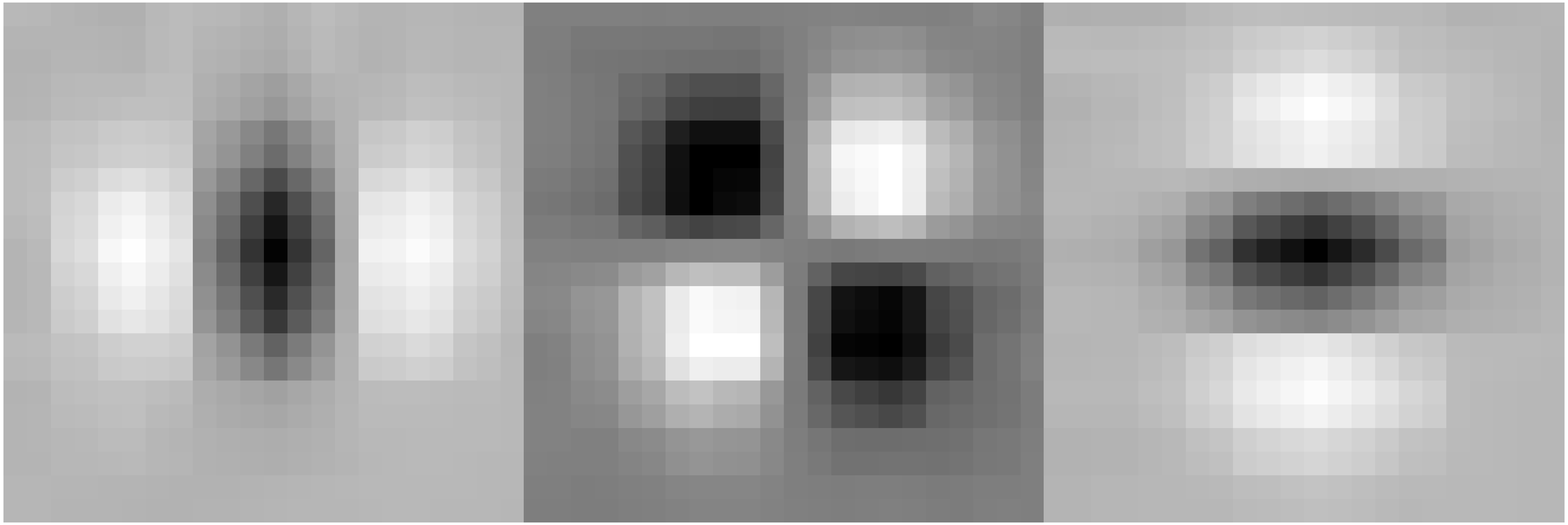
$$R_{xx} = \frac{\partial^2 G_a}{\partial x^2} * f$$

$$R_{xy} = \frac{\partial^2 G_a}{\partial x \partial y} * f$$

$$R_{yy} = \frac{\partial^2 G_a}{\partial y^2} * f$$

Ridge detection

The second order filters:

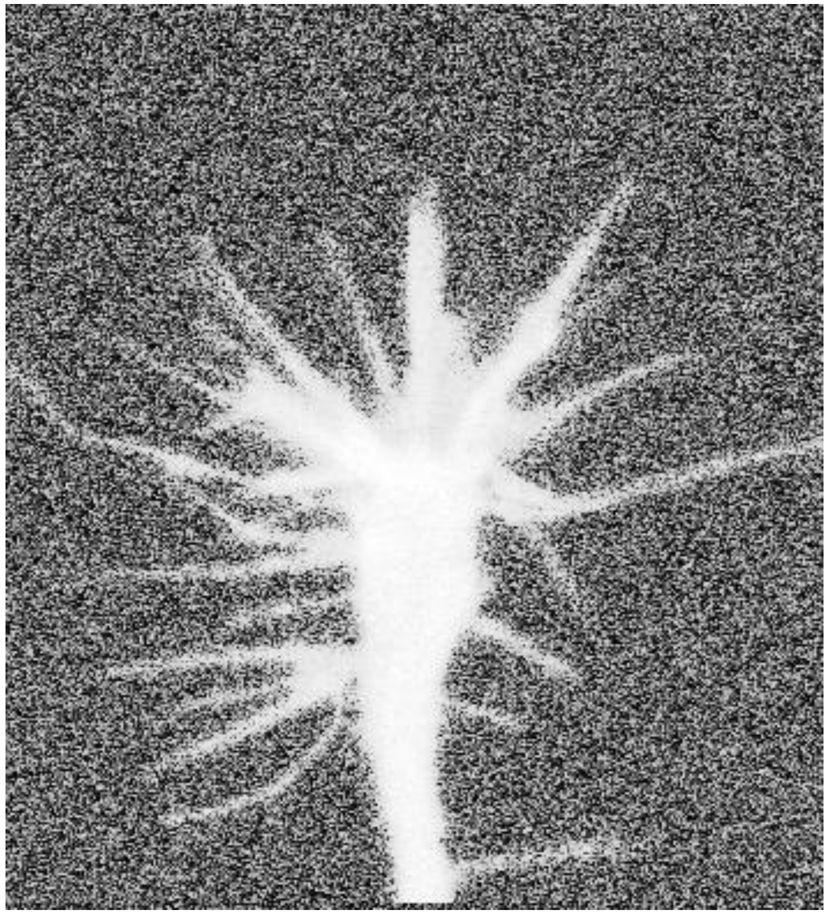


A filter in an arbitrary direction given by θ:

$$\begin{pmatrix}\cos(\theta) & \sin(\theta)\end{pmatrix}\begin{pmatrix}R_{xx} & R_{xy}\\ R_{xy} & R_{yy}\end{pmatrix}\begin{pmatrix}\cos(\theta)\\ \sin(\theta)\end{pmatrix}$$

Ridge detection

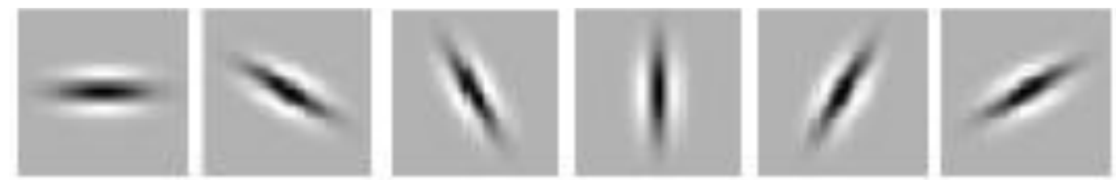
- Growing nerve-cell. Find the threadlike structures growing out from the growth cone



Histogram equalization

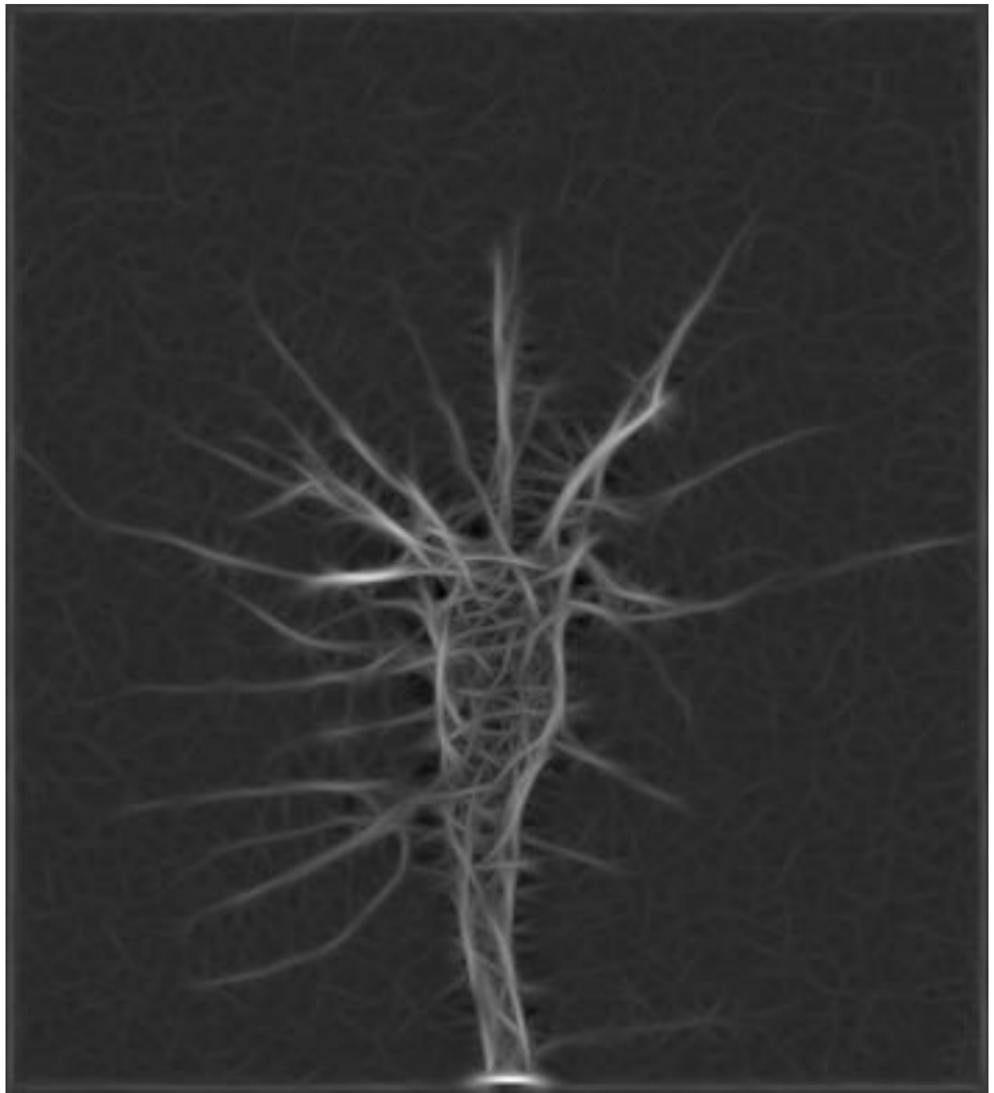
Ridge detection

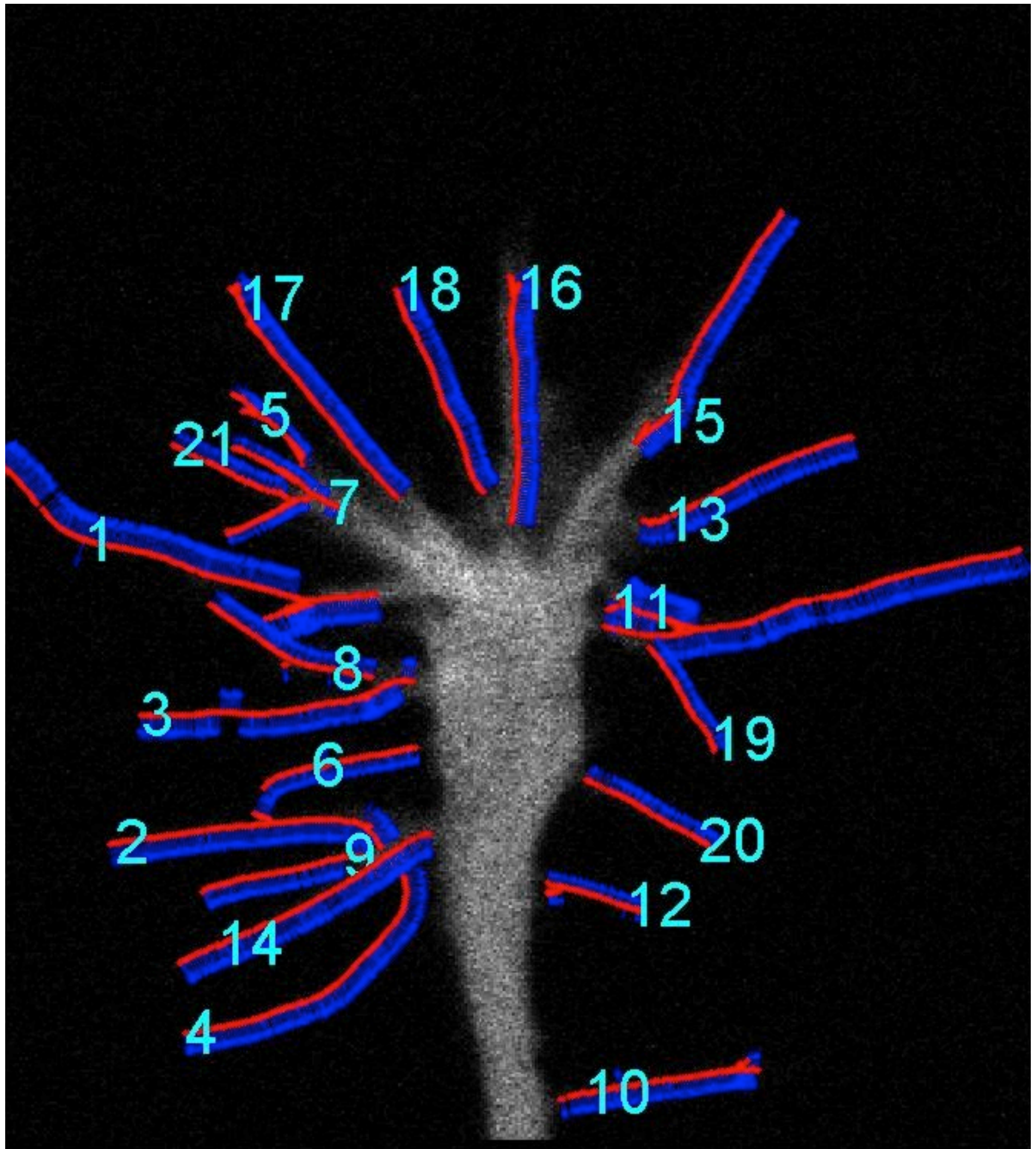
- Filter with elongated gaussians in different directions
- Filterbank with 16 directions



Ridge detection

- Filter with elongated gaussians in different directions

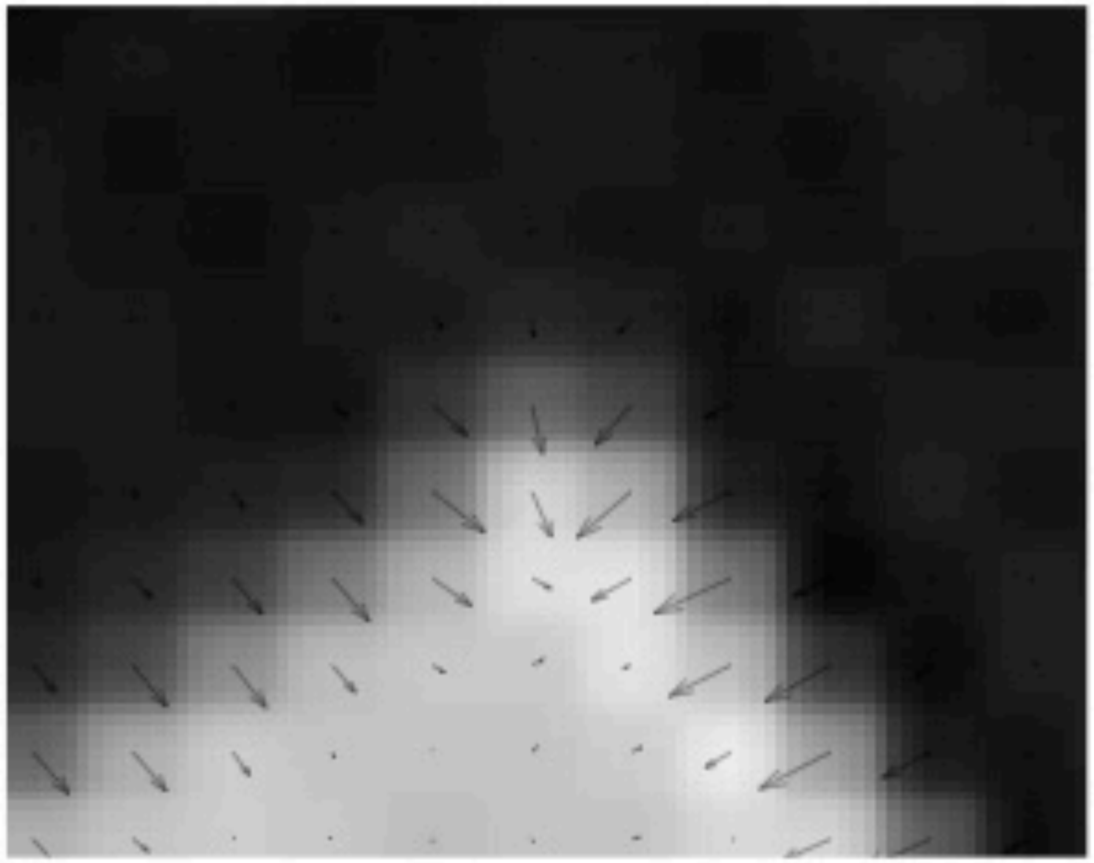




Overview –Feature Detection

- Techniques
  - Scale space theory
- Detectors
  - Edges
  - Ridges
  - **Corners**
  - SIFT
- Texture

Illustration of partial derivatives



Illustrations of the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$

Structure/Orientation Tensor

Construct the matrix

$$M = \begin{bmatrix} W_{xx} & W_{xy} \\ W_{xy} & W_{yy} \end{bmatrix} = \begin{bmatrix} (\frac{\partial f}{\partial x})^2 * G_b & (\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}) * G_b \\ (\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}) * G_b & (\frac{\partial f}{\partial y})^2 * G_b \end{bmatrix},$$

where G_b denotes the Gaussian function with parameter b.
M - orientation tensor.
Note: We construct a matrix for every pixel.

Structure Tensor

The matrix M has the following properties:

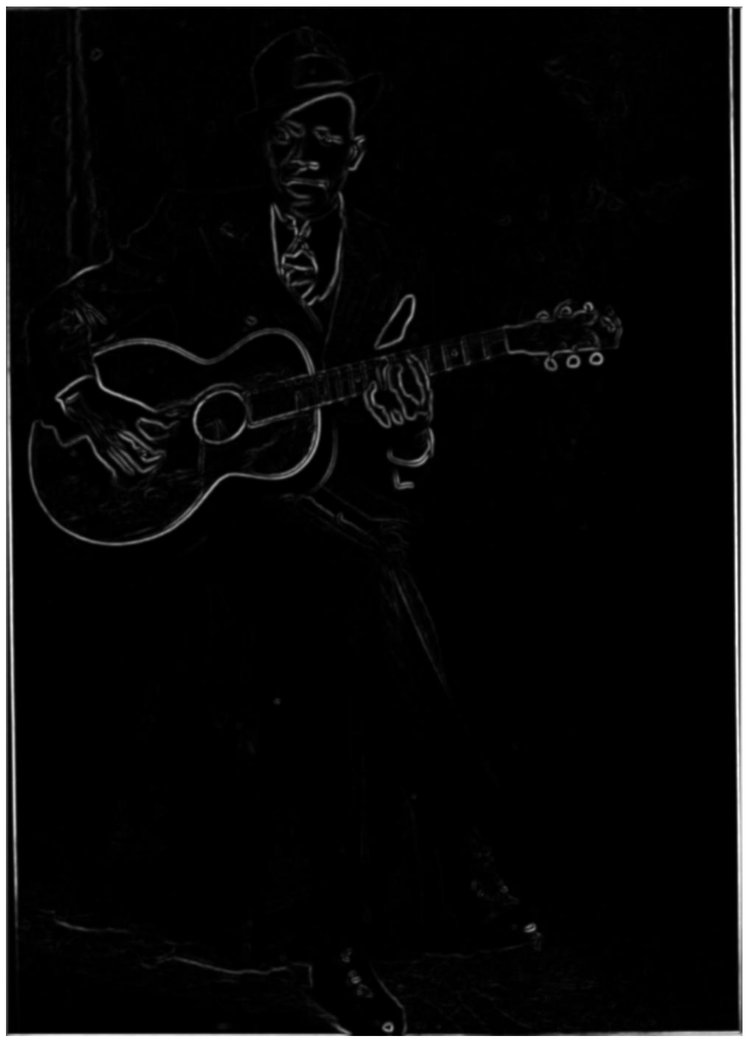
- (Flat) Two small eigenvalues in a region - flat intensity.
- (Flow) One large and one small eigenvalue - edges and flow regions.
- (Texture) Two large eigenvalues - corners, interest points, texture regions.

This can be used in algorithms for segmenting the image into (flat, flow, texture).

Corner Detector

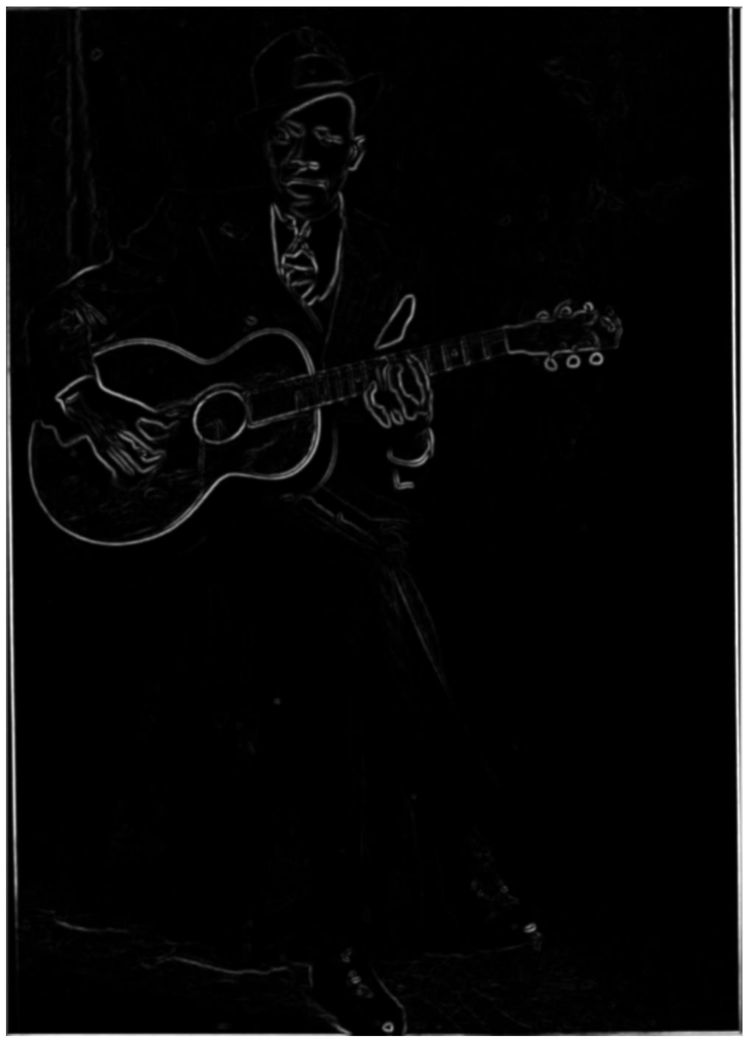
- Compute x- and y-derivatives with a Gaussian filter
- Form the orientation tensor M for every pixel
- Compute the product of eigenvalues, i.e. the determinant of M
- If both eigenvalues large (product is a local maximum), then it is a corner!

Harris Corner Detector

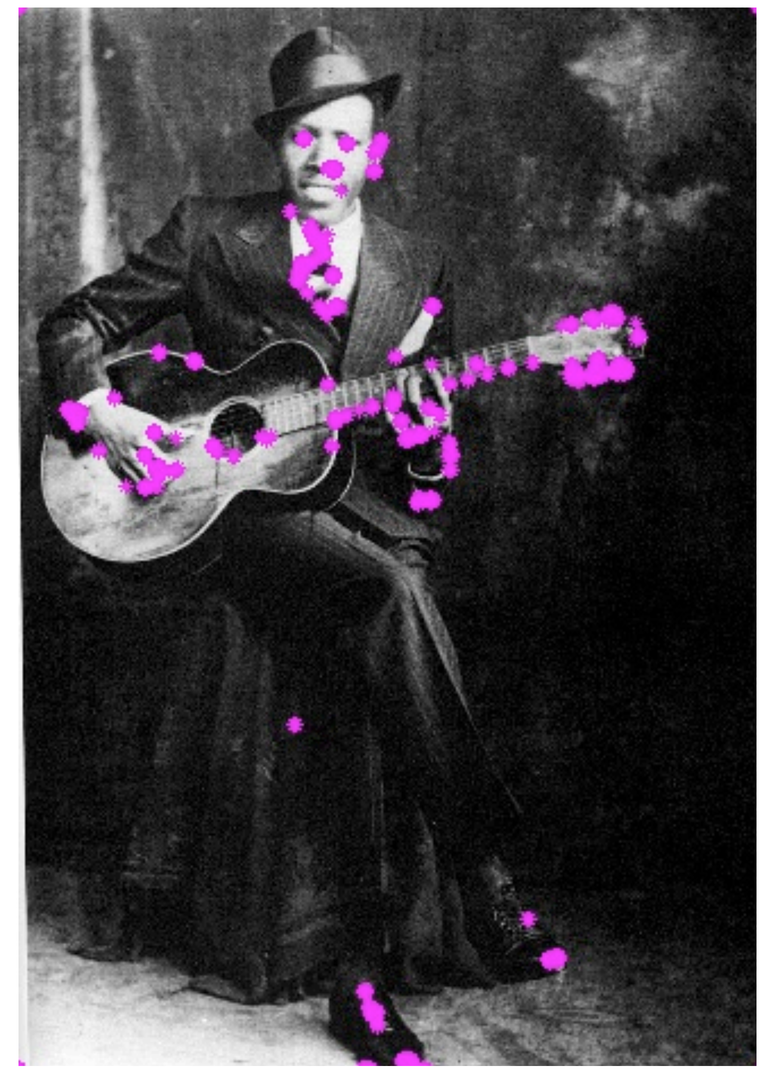


Eigenvalue two of
the orientation tensor

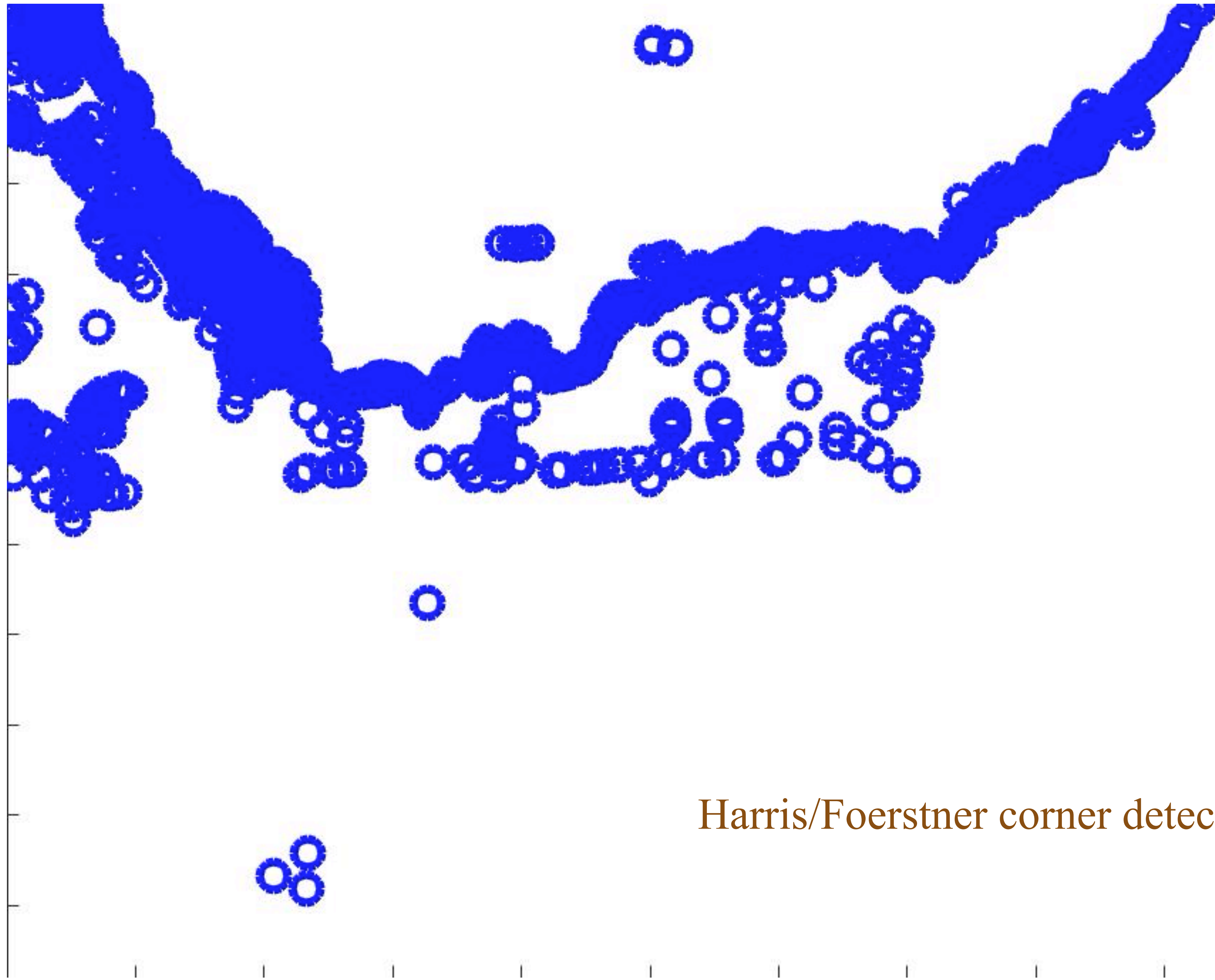
Harris Corner Detector



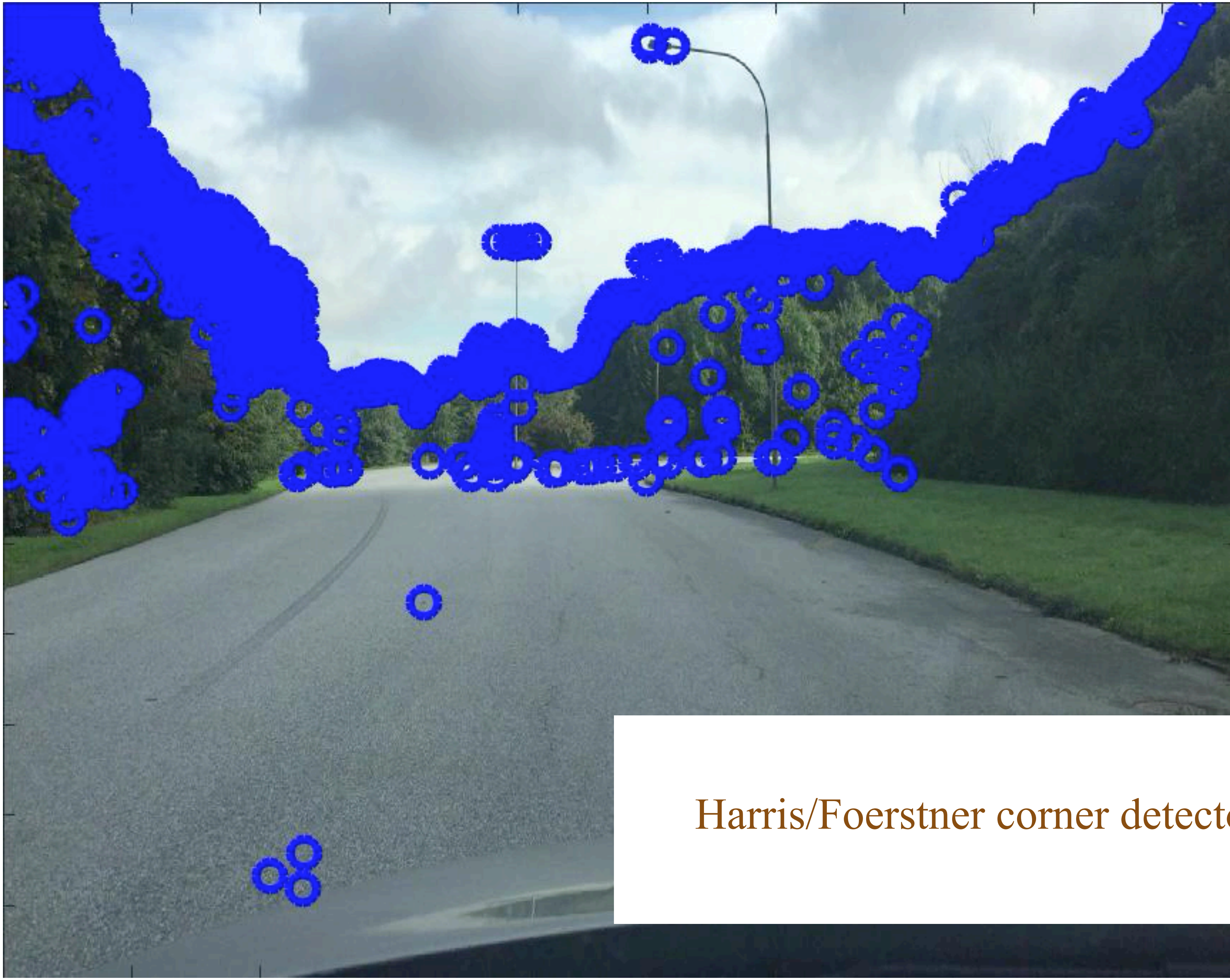
Eigenvalue two of
the orientation tensor



Two large Eigenvalues
Gives a corner



Harris/Foerstner corner detector

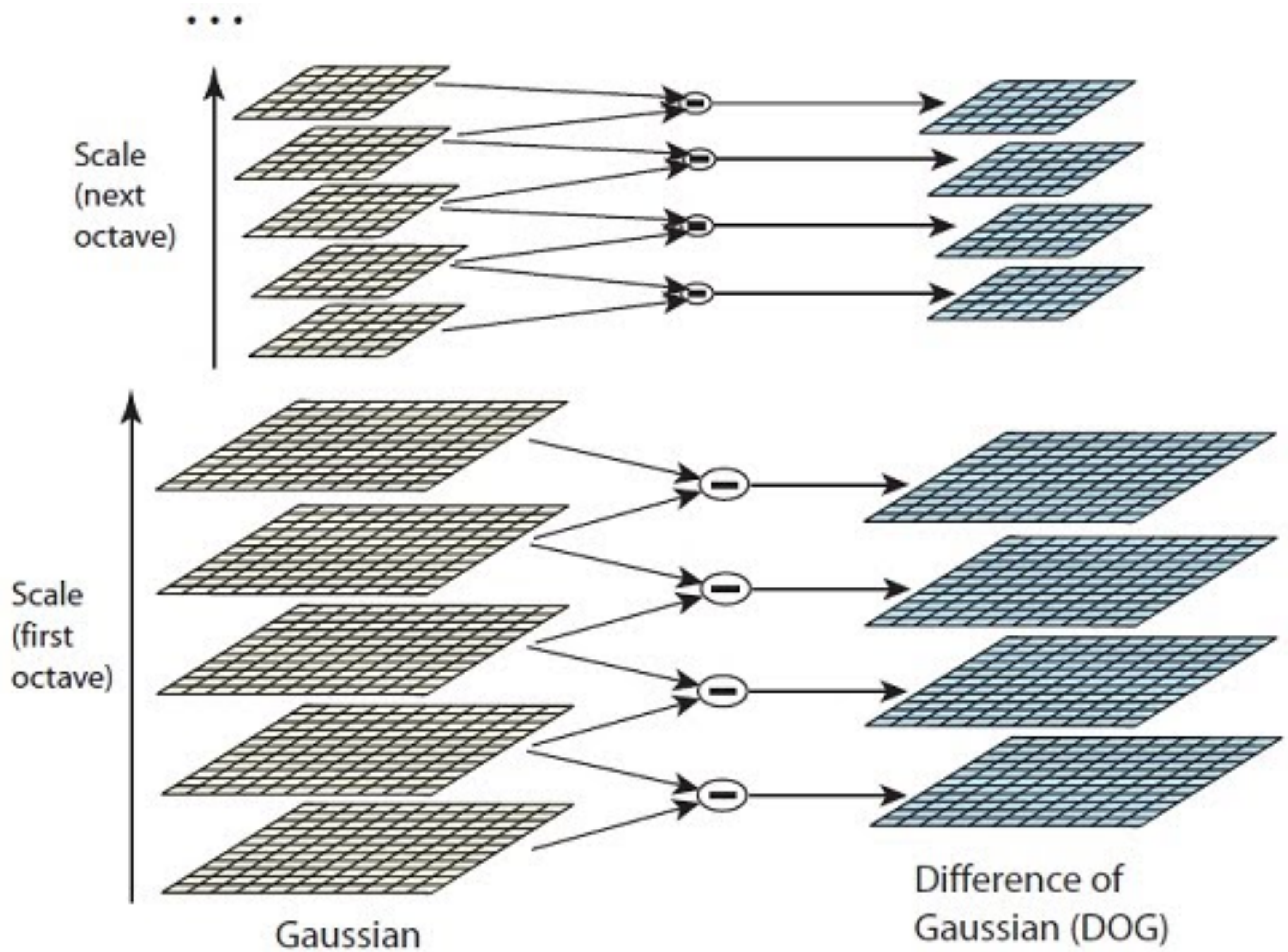


Harris/Foerstner corner detector

Overview –Feature Detection

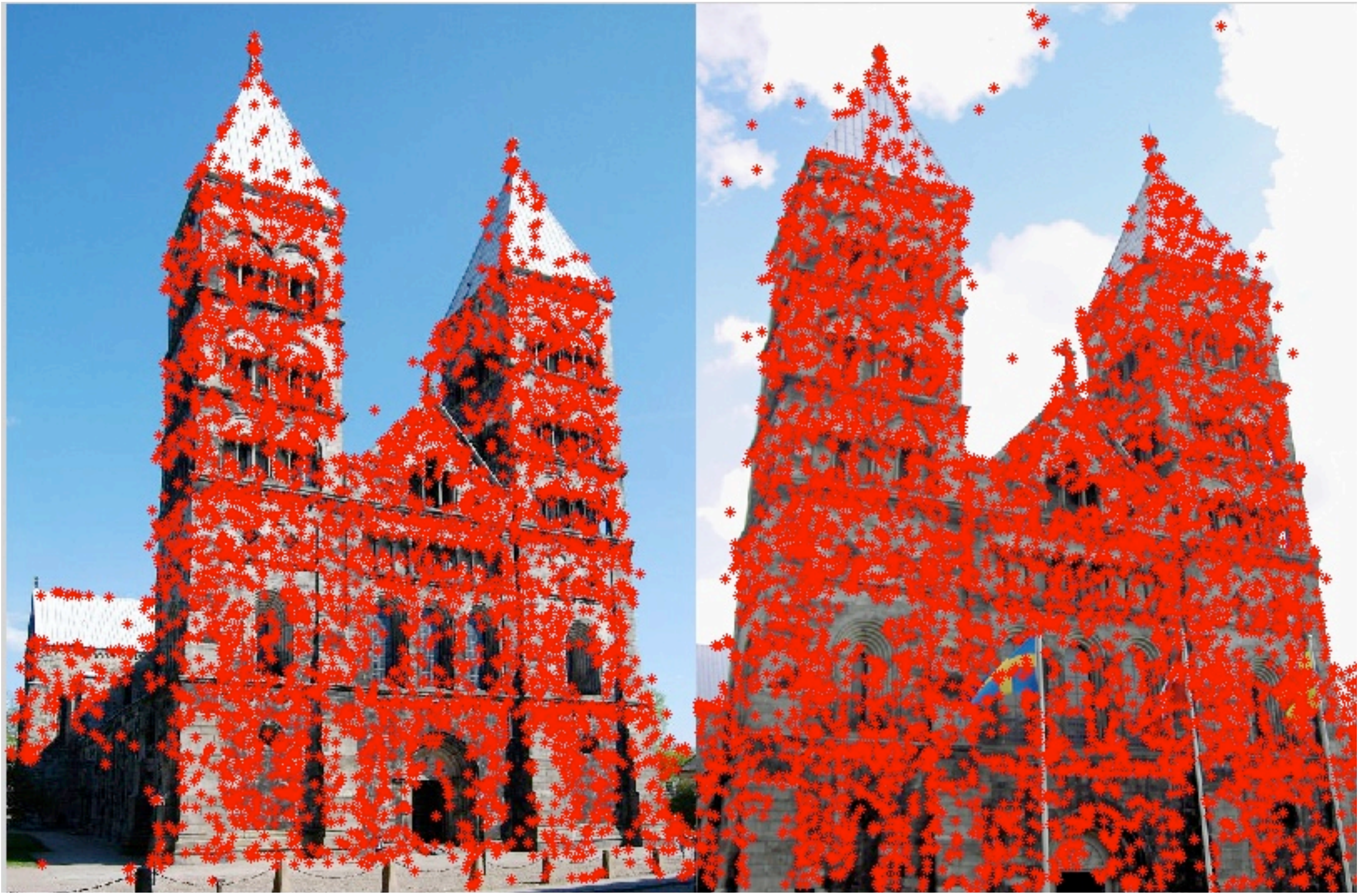
- Techniques
  - Scale space theory
- Detectors
  - Edges
  - Ridges
  - Corners
  - **SIFT**
- Texture

SIFT
(Scale Invariant Feature Transform)

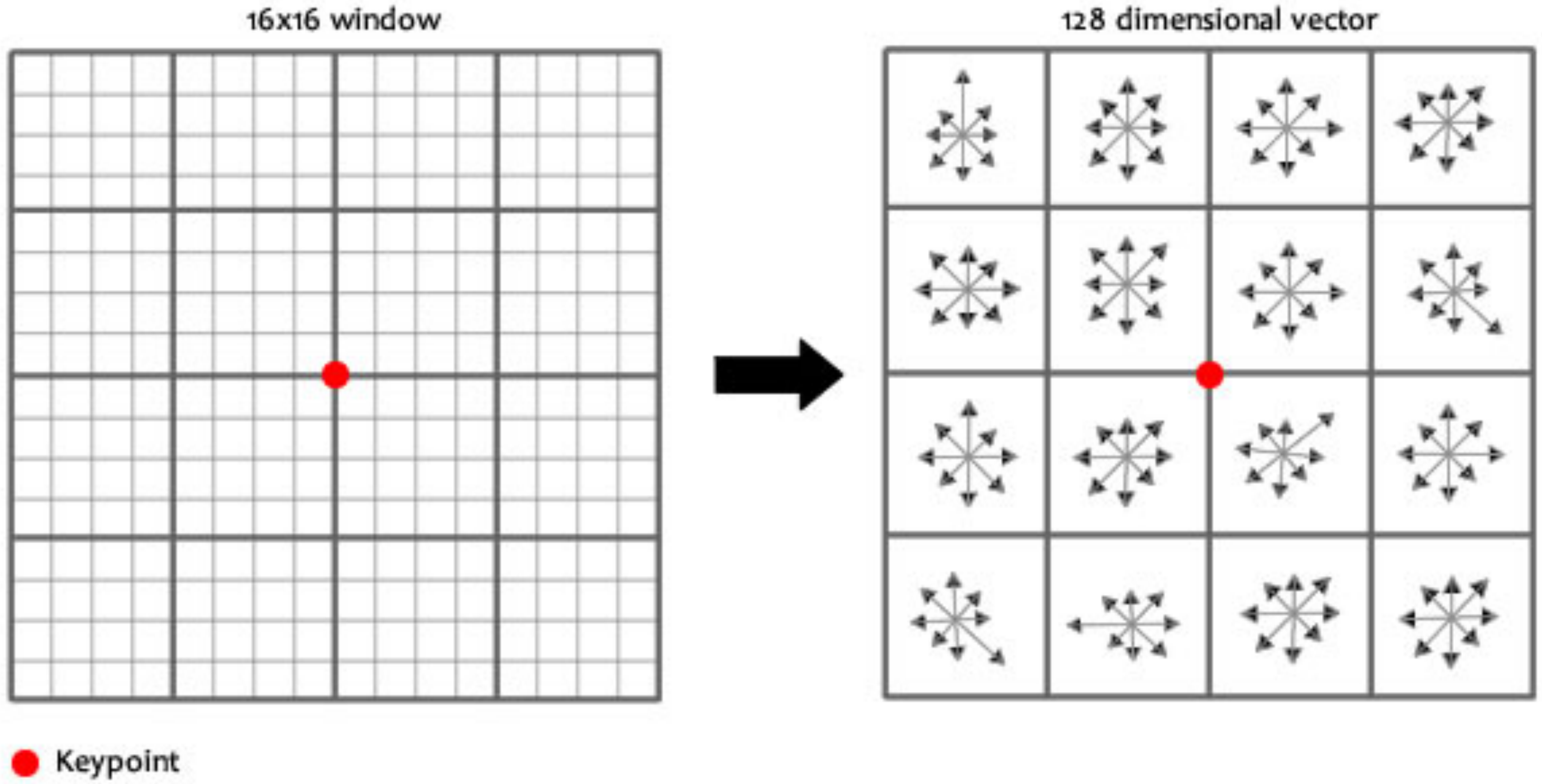


SIFT
(Scale Invariant Feature Transform)

SIFT
(Scale Invariant Feature Transform)

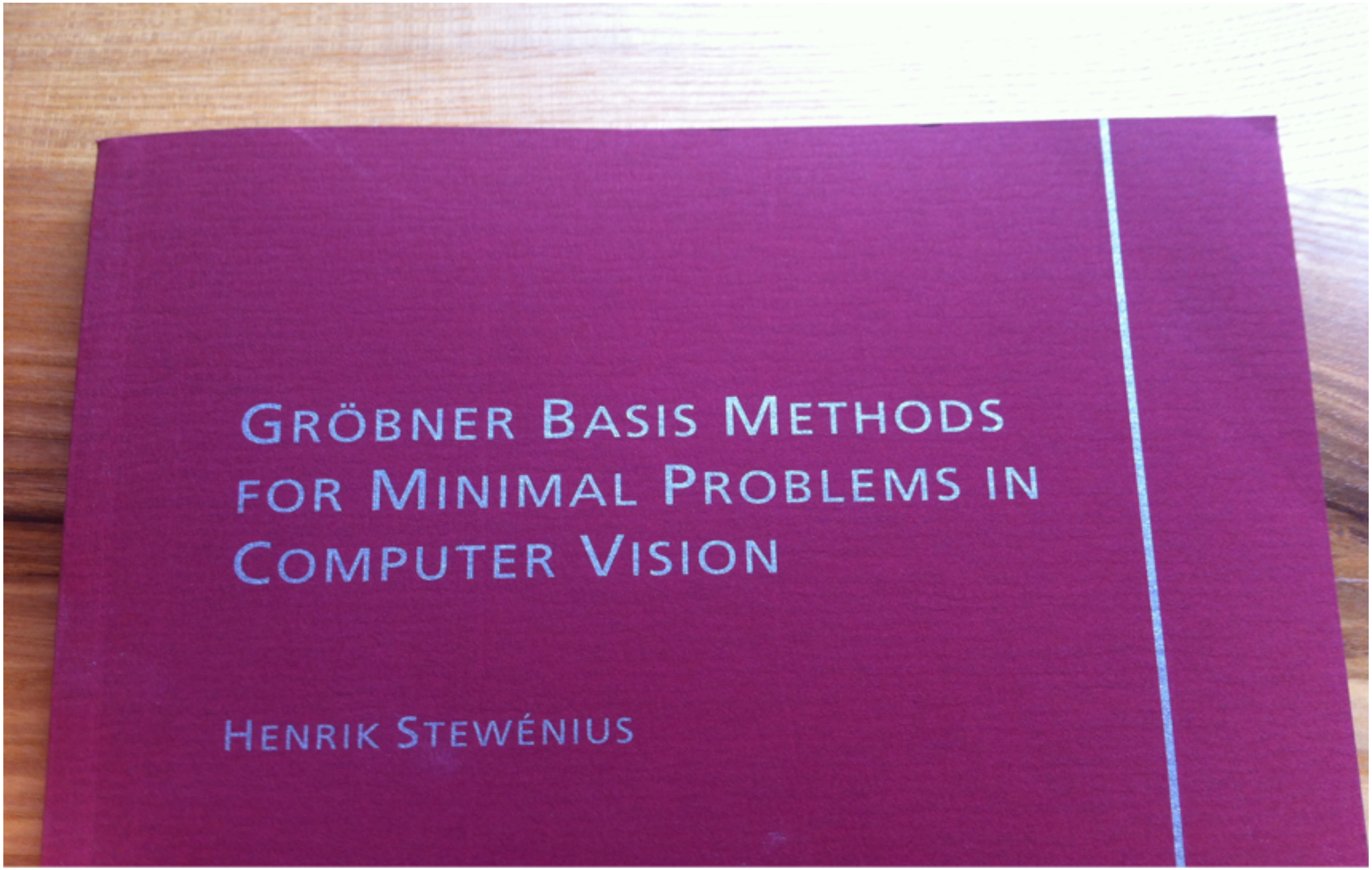


SIFT
Descriptor



SIFT examples

- Examples from image search
- Examples from 3D modelling


GRÖBNER BASIS METHODS
FOR MINIMAL PROBLEMS IN
COMPUTER VISION
HENRIK STEWÉNIUS

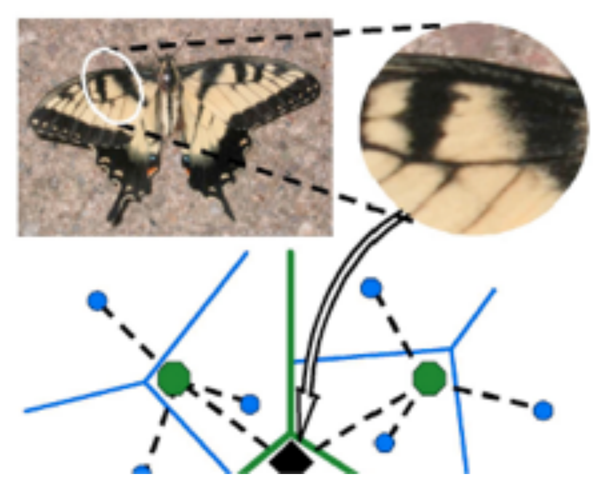
Scalable Recognition with a Vocabulary Tree

David Nistér and Henrik Stewénius
Center for Visualization and Virtual Environments
Department of Computer Science, University of Kentucky
http://www.vis.uky.edu/~dnister/ http://www.vis.uky.edu/~stewe/

Abstract

A recognition scheme that scales efficiently to a large number of objects is presented. The efficiency and quality is exhibited in a live demonstration that recognizes CD-covers from a database of 40000 images of popular music CD's.

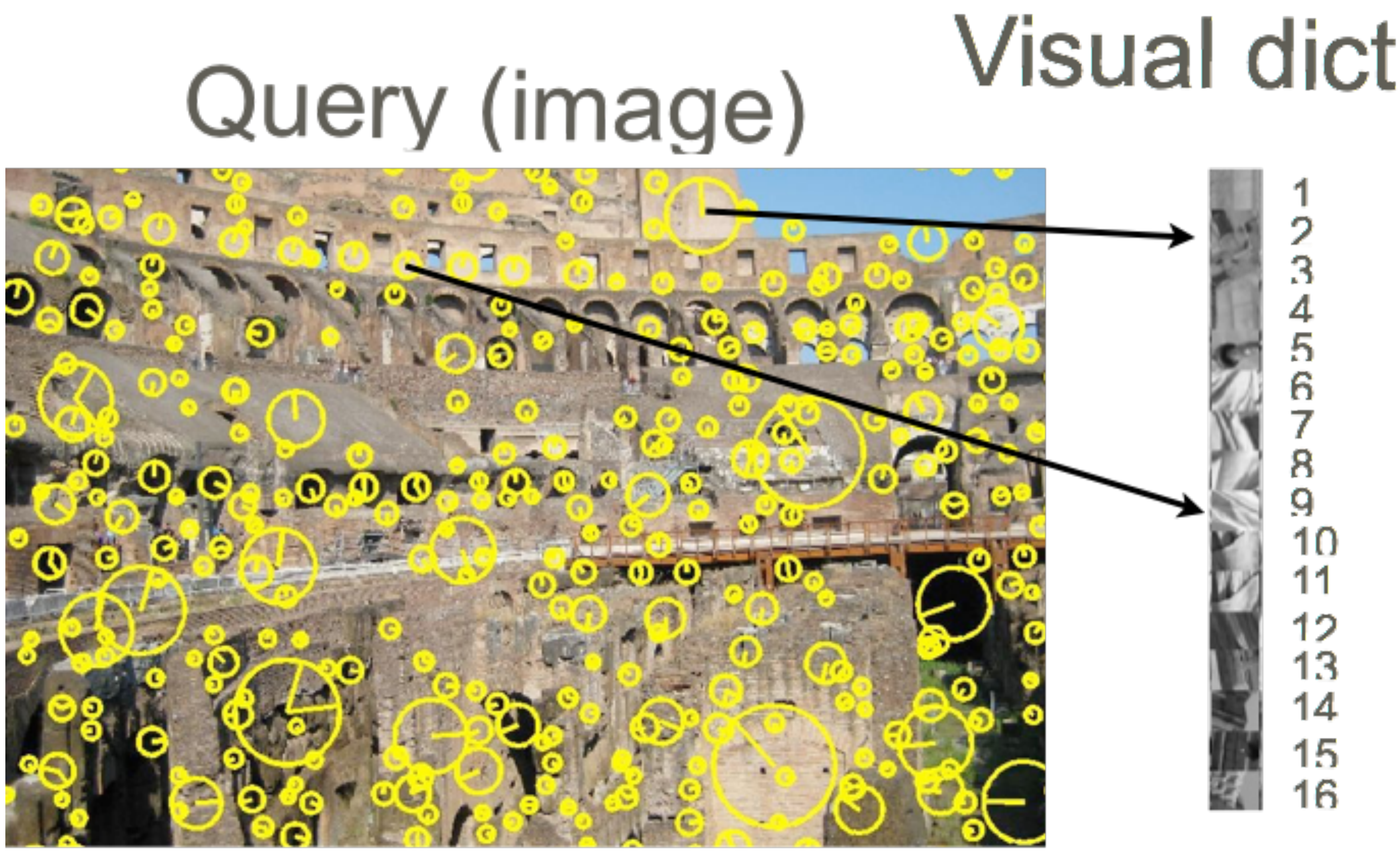
The scheme builds upon popular techniques of indexing descriptors extracted from local regions, and is robust to background clutter and occlusion. The local region descriptors are hierarchically quantized in a vocabulary tree. The vocabulary tree allows a larger and more discriminatory vocabulary to be used efficiently, which we show experimentally leads to a dramatic improvement in retrieval quality. The most significant property of the



- Query (text):
- ”wiki Lund University”

- Query contains ’words’: 9, 6, 8

- Dict.
  - ‘Aardvark’ - 1
  - ‘Abba’ - 2
  - ‘Conference’ - 3
  - ‘Eslöv’ - 4
  - ‘Lomma’ - 5
  - ‘Lund’ - 6
  - ‘Malmö’ - 7
  - ‘University’ - 8
  - ‘wiki’ - 9
  - ’Ös’ - 10



- Image contains 'visual words': 1, 8, ...

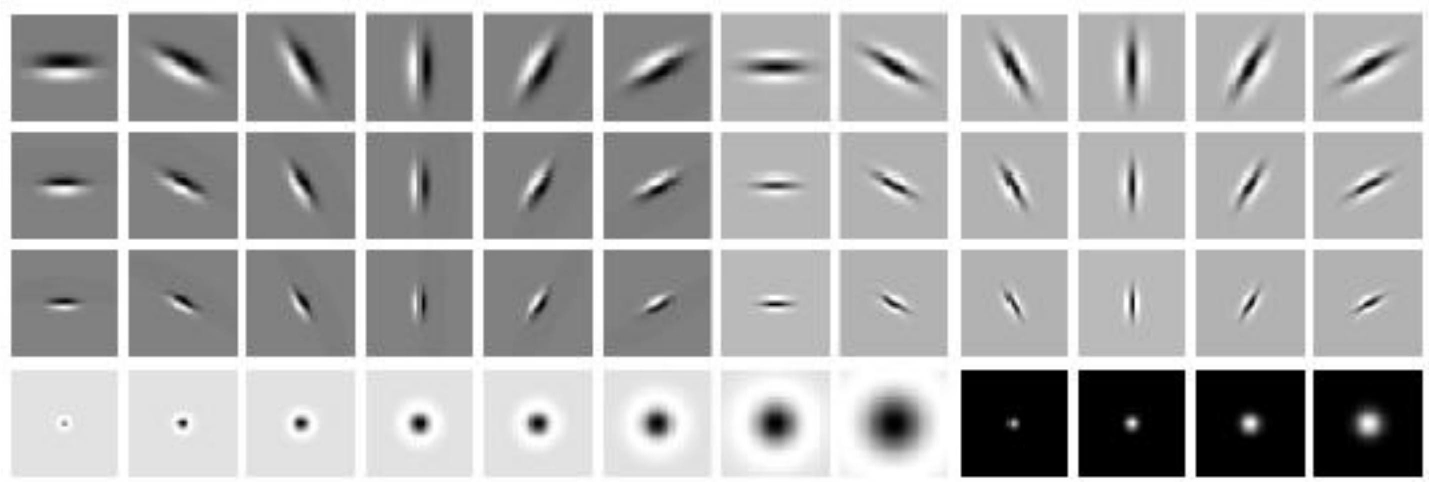
Overview –Feature Detection

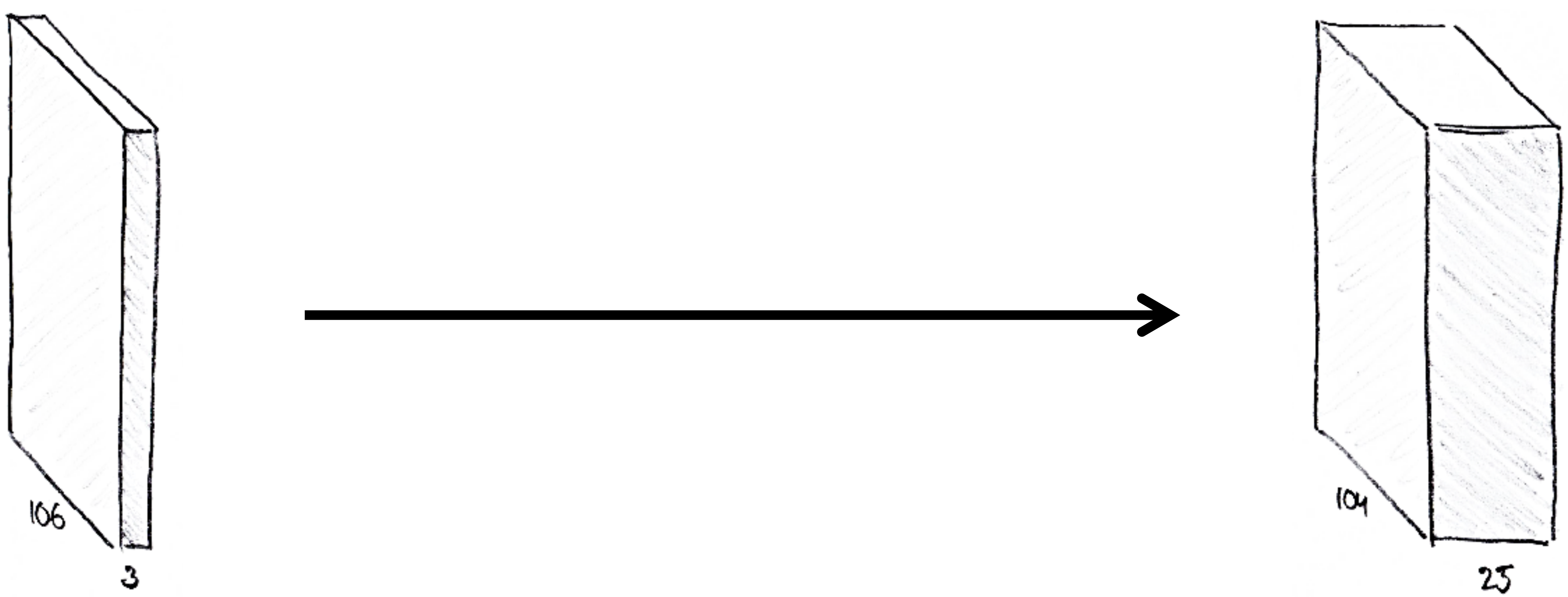
- Techniques
  - Scale space theory
- Detectors
  - Edges
  - Ridges
  - Corners
  - SIFT
- **Texture**

Filterbank

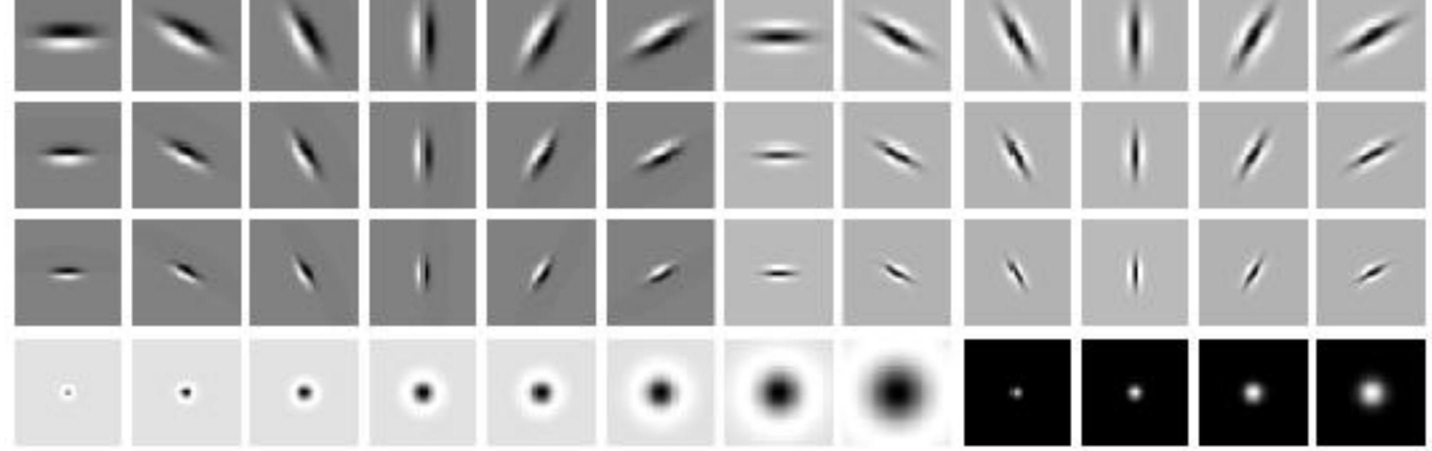
Convolve with each filter

Result is a block of data





Filterbank

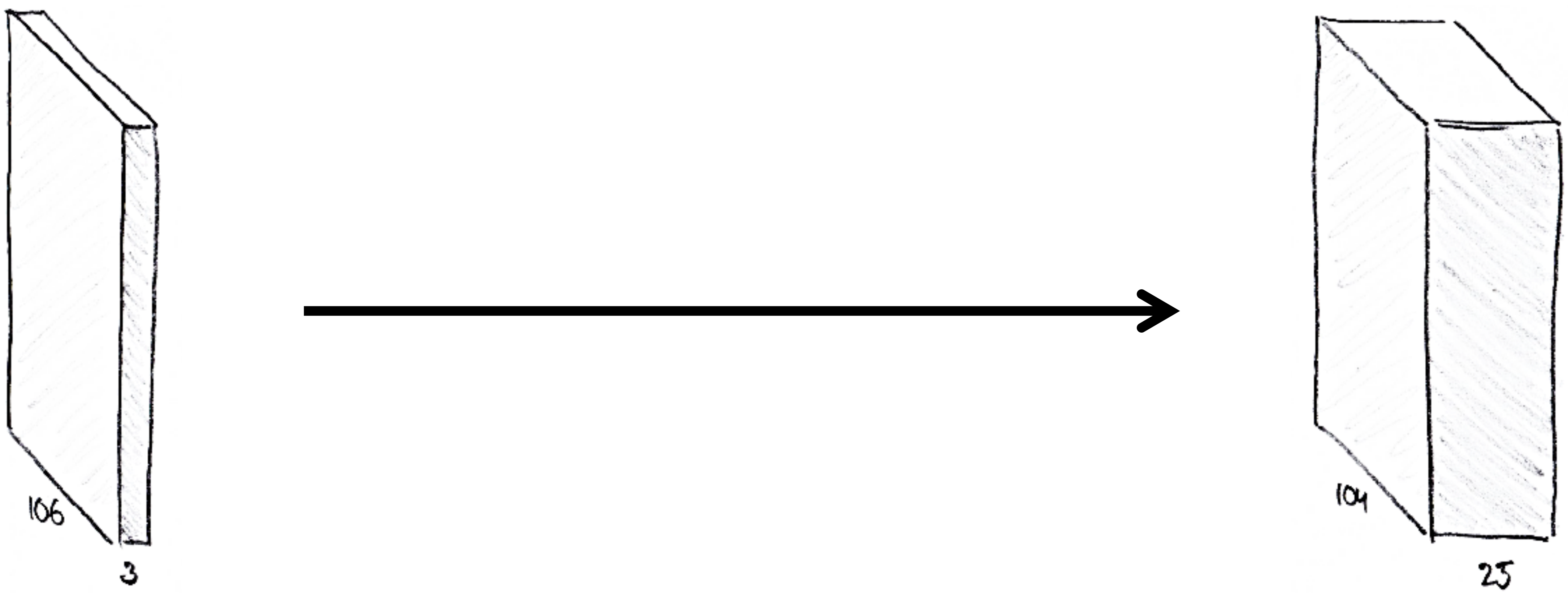


Convolution, filter h

$$g(i,j) = \sum_u \sum_v f(i-u, j-v) h(u,v)$$

Many convolutions, filterbank h with K filters

$$g(i,j.k) = \sum_u \sum_v f(i-u, j-v) h(u,v,k)$$



(Lecture 7 Deep Learning), CNN-Blocks - Convolutional layer

- Input: Data block x of size

$$m \times n \times k_1$$

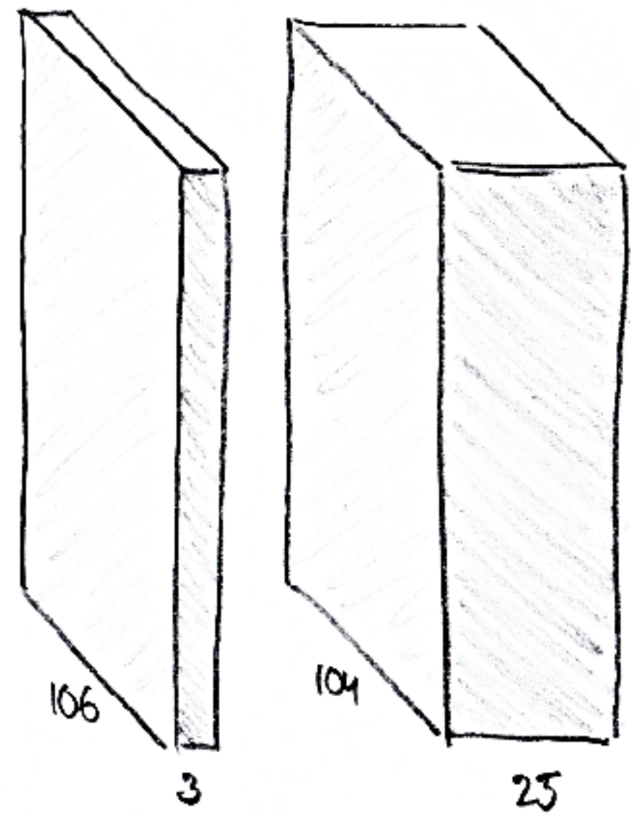
- Output: Data block y of size

$$m \times n \times k_2$$

- Filter: Filter kernal block w of size

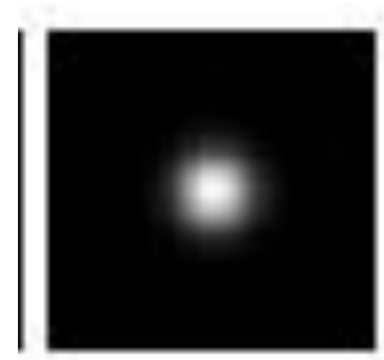
$$m_w \times n_w \times k_1 \times k_2$$

- Offsets: Vector w_o of length k_2

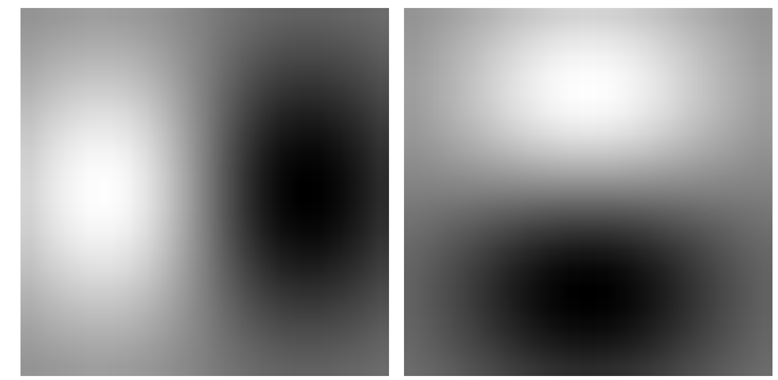


$$y(i,j,k) = w_o(k) + \sum_u \sum_v \sum_l x(i-u, j-v, l) w(u,v,l,k)$$

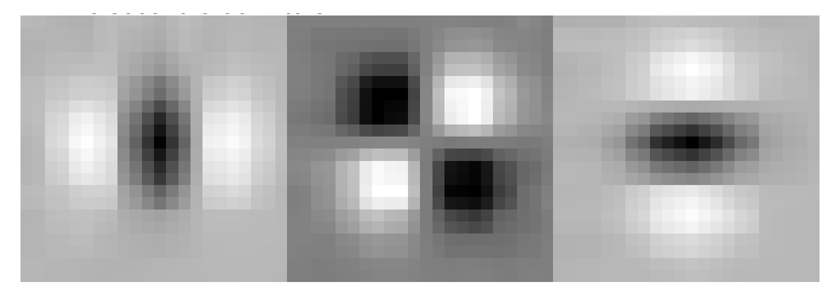
Filterbanks



- Blobdetection – uses one filter



- Edge detection – uses two filter (dx,dy)



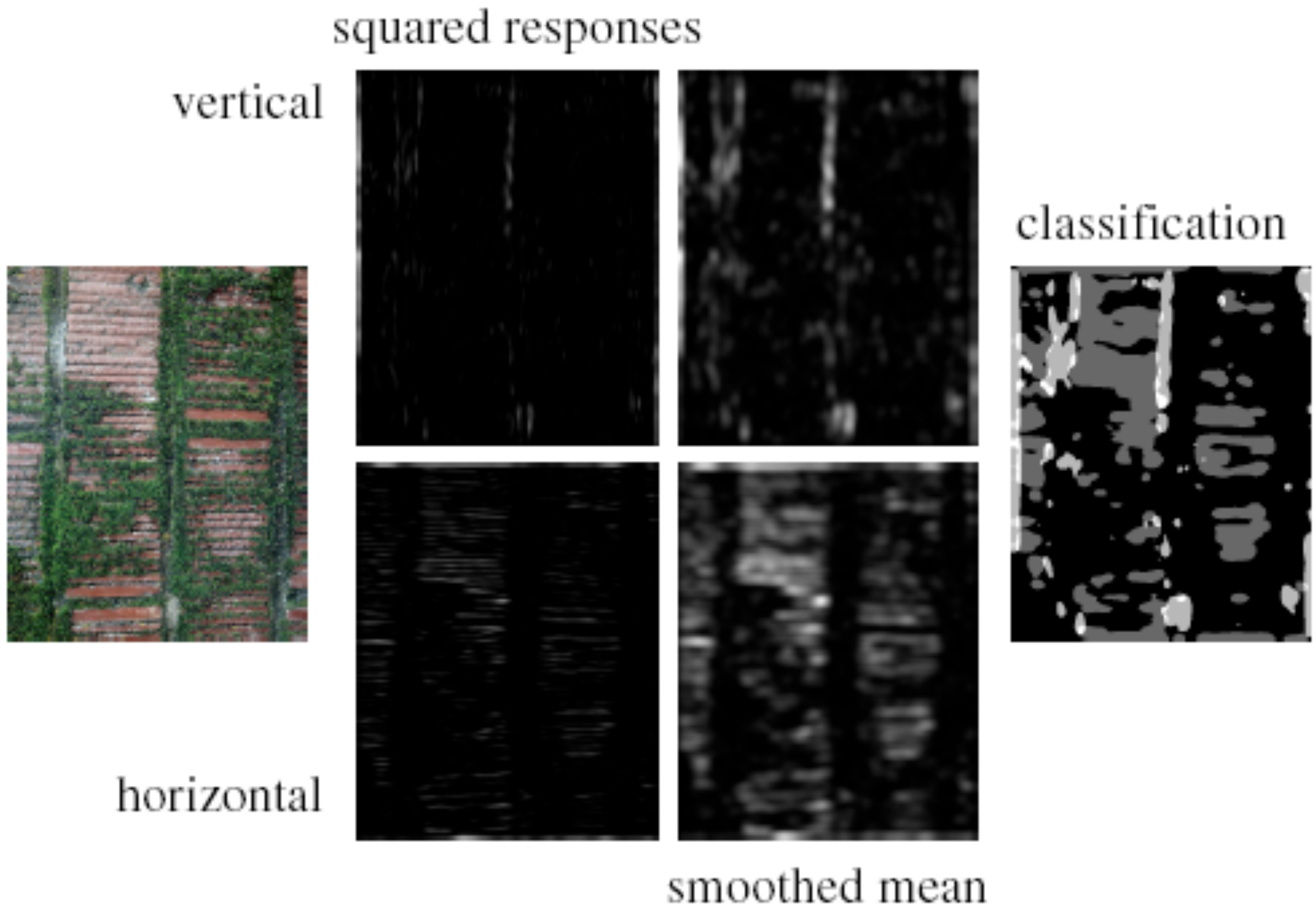
- Ridgedetection – uses three filters …

- ... Or more

Texture

Texture is easy to recognize, but difficult to explain.
A leaf is an object, but foliage is a texture.

- Texture recognition
- Texture synthesis
- Shape from texture



Computer Vision - A Modern Approach
Set: Pyramids and Texture
Slides by D.A. Forsyth

What is texture?

- An image obeying some statistical properties
- Similar structures repeated
- Often some degree of randomness

Segmentation and texture

Background/foreground

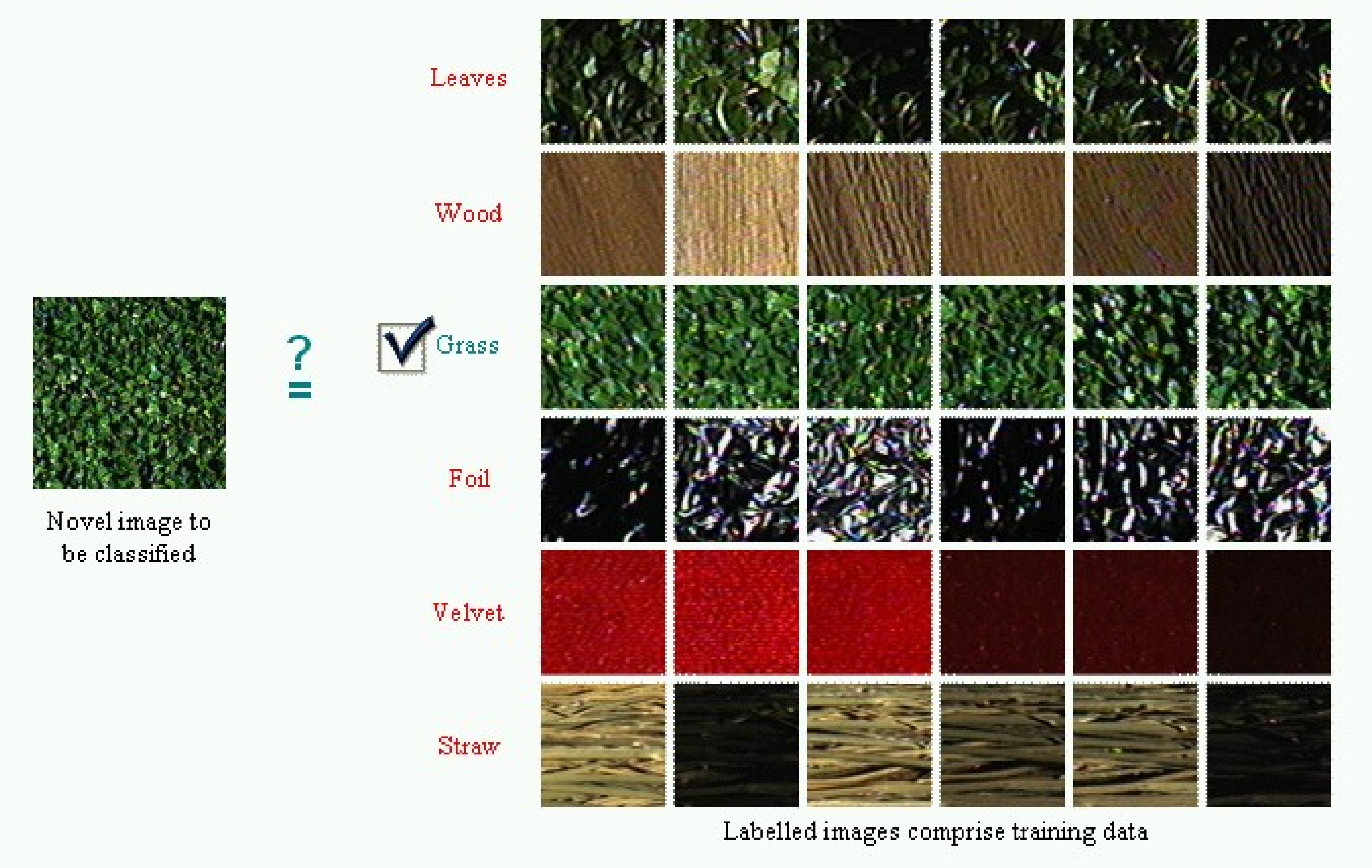
Segmentation and texture

Different texture

Segmentation and texture

Different objects

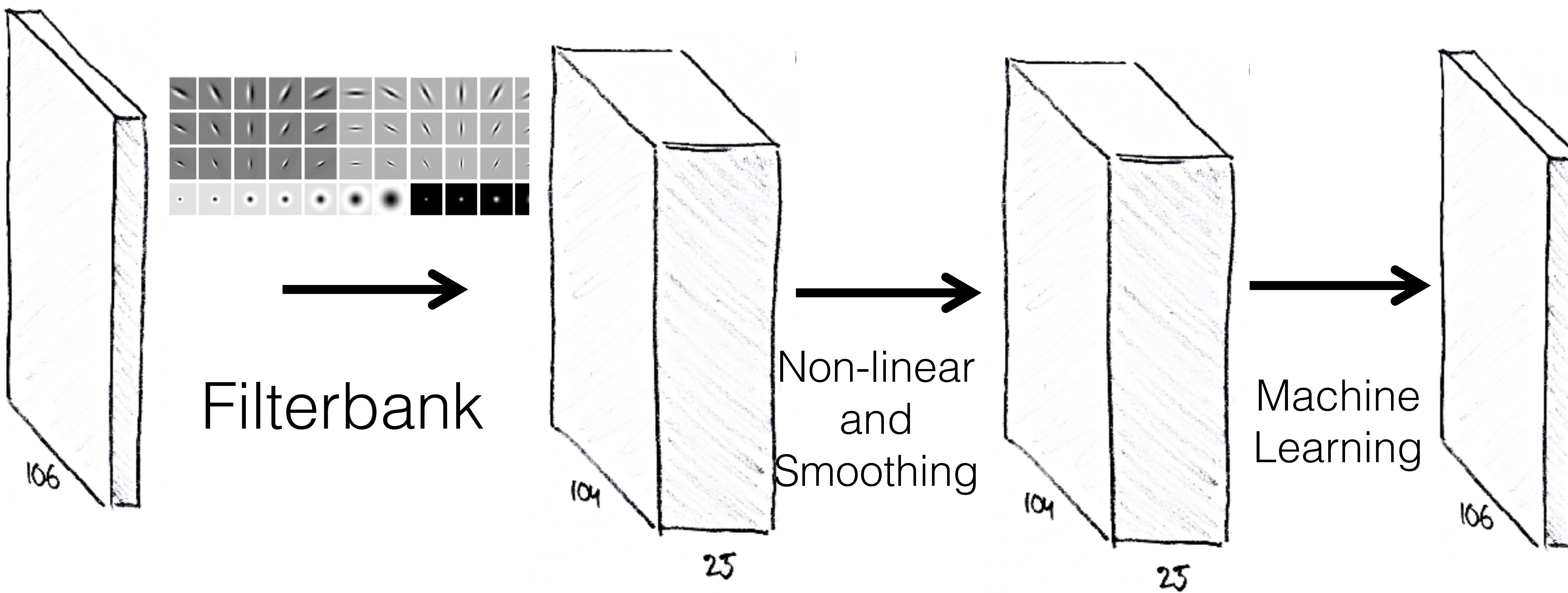
Texture



Images taken from: `http://www.robots.ox.ac.uk/~vgg/research/texclass/`

Texture – main ideas

- Several filters $(f * h_1, \ldots, f * h_n)$
- What filters $h_1, \ldots, h_n$ should we use?
- Non-linear transformation, e.g. squares, absolute values, taking the positive or negative part of a signal, thresholding.
- Use machine learning - classification on filter responses.



Texture

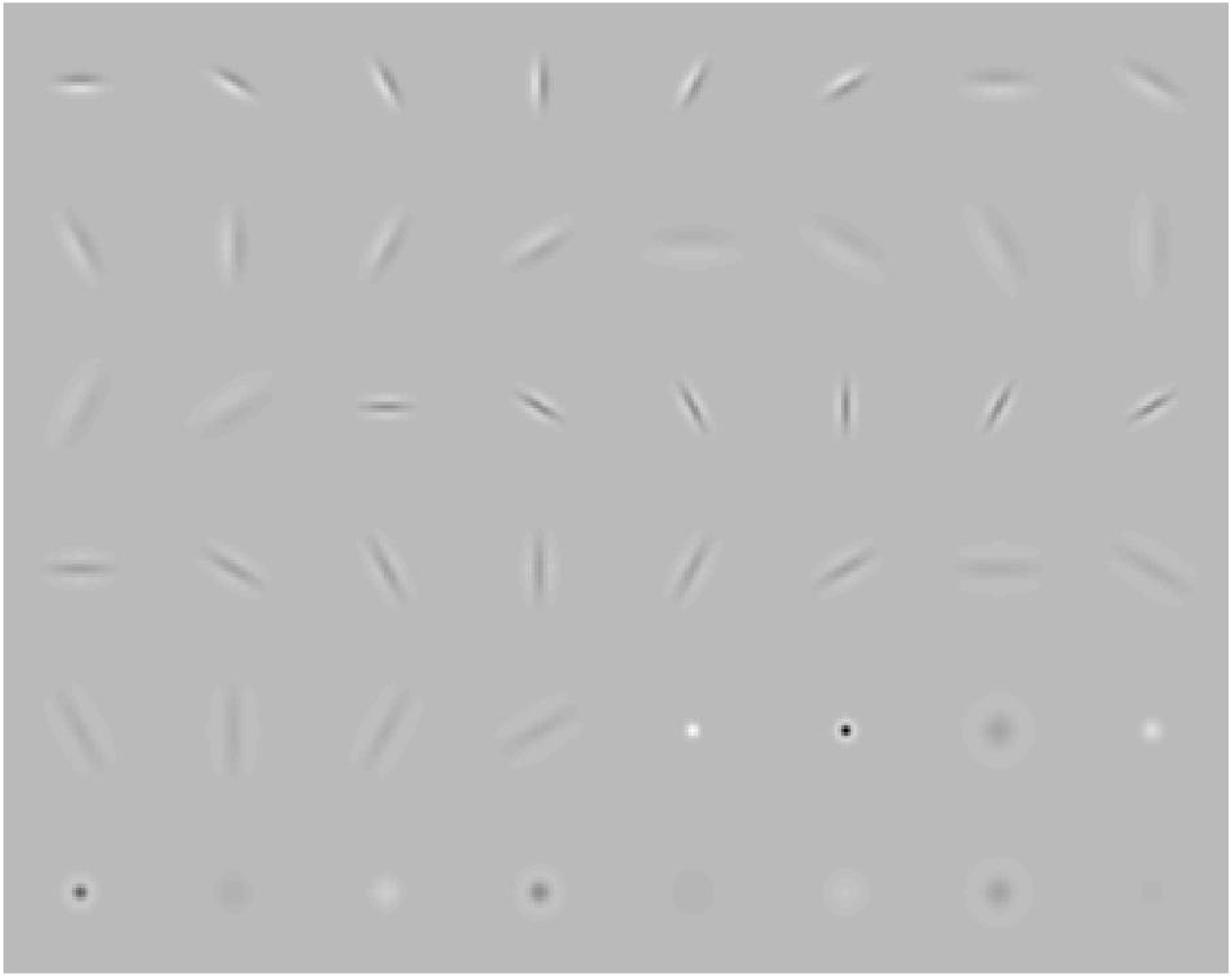
http://www.ux.uis.no/~karlsk/tct/

http://www.alceufc.com/2013/09/texture-classification.html

Texture – filter banks

- **Spots:** Gaussian filters
- **Spots:** Difference of Gaussian filters
- **Bars:** Elongated Gaussians
- **Edges:** Derivatives of Gaussians and of elongated Gaussians
- **Ridges:** Second derivatives of Gaussians and of elongated Gaussians
- **Gabor filters:**

Texture – filter banks (Gabor)



Texture – non-linear transformations

You can try several non-linear transformations, e.g.

- Squaring, polynomials
- Absolute value
- Thresholding (in particular taking the positive and negative parts of a signal).

After non-linear transformation, often it is a good idea to form the mean over a region, e.g. using mean value filtering. Similar to what we did with the orientation tensor. Alternatively one could take the maximal value over a region.

Example application

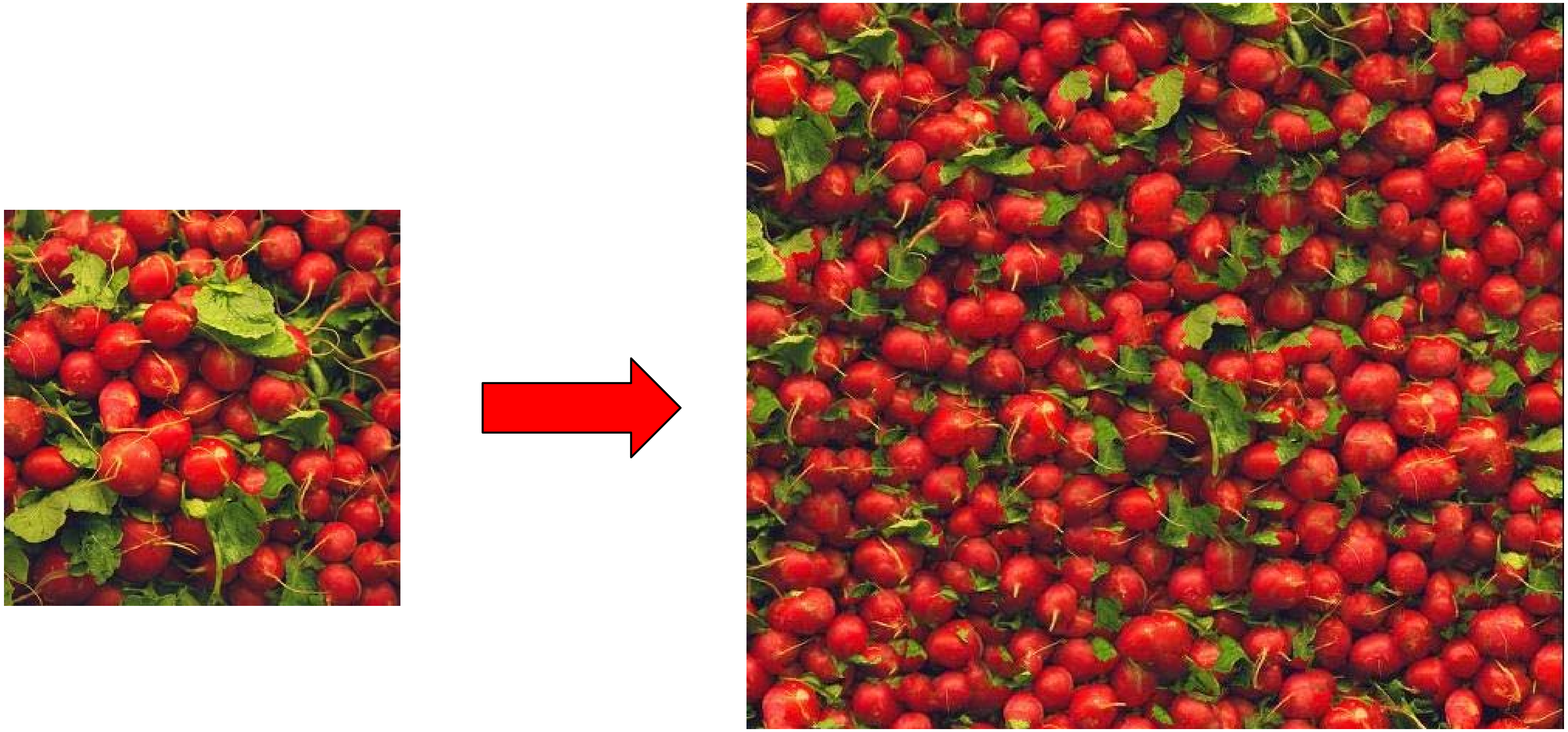
Texture Synthesis

Given a small sample

Example application

Texture Synthesis

Given a small sample, generate larger realistic versions of the texture



LUND UNIVERSITY

350