



LUND
UNIVERSITY

350

Image Analysis (FMAN20)

Lecture 16, 2019

MAGNUS OSKARSSON

5 Deckel-München

COMPU

2.9

4

8

11

16

22



After the course

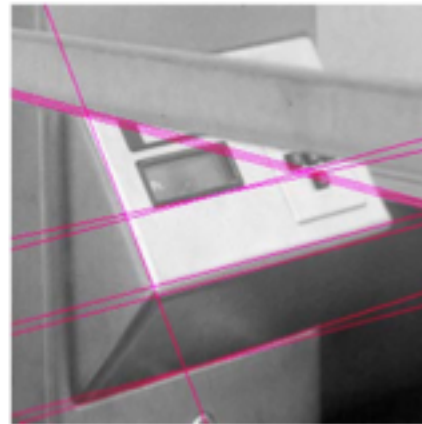
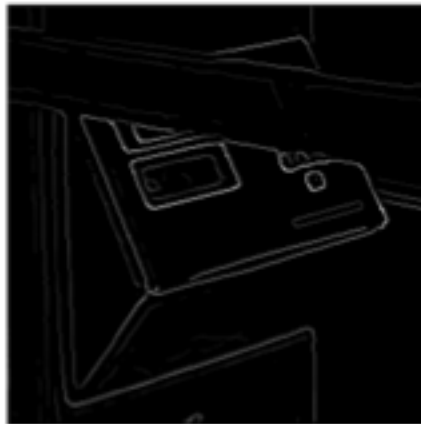
- You should be able to develop and test your own image analysis system
- You should have tools for understanding and working with big data
- You should have improved your skills in programming and modelling.

In this course

- Tools:
 - Basics of image analysis, sampling, interpolation, histogram equalisation, thresholding, connected components
 - Linear Algebra, Projections, PCA, Linear System Theory, Convolution, FFT
 - Mathematical Statistics, Distributions, Parametric models
 - Machine Learning, clustering, k-means, classification, bayes theorem, SVM, LR, Deep Learning
 - Segmentation, Graph-based methods, fitting, RANSAC, Hough
- System development
 - Based on the tools
 - Ground truth, evaluation, benchmarking

Fitting

We want to **associate** a **model** with **observed features**



[Fig from Marszalek & Schmid, 2007]

For example, the model could be a line, a circle, or an arbitrary shape. We use a model to produce compact representations that capture the relevant image structures we seek.

Hough Transform

- Goal: Finding linear structures in images
- Used on edge data

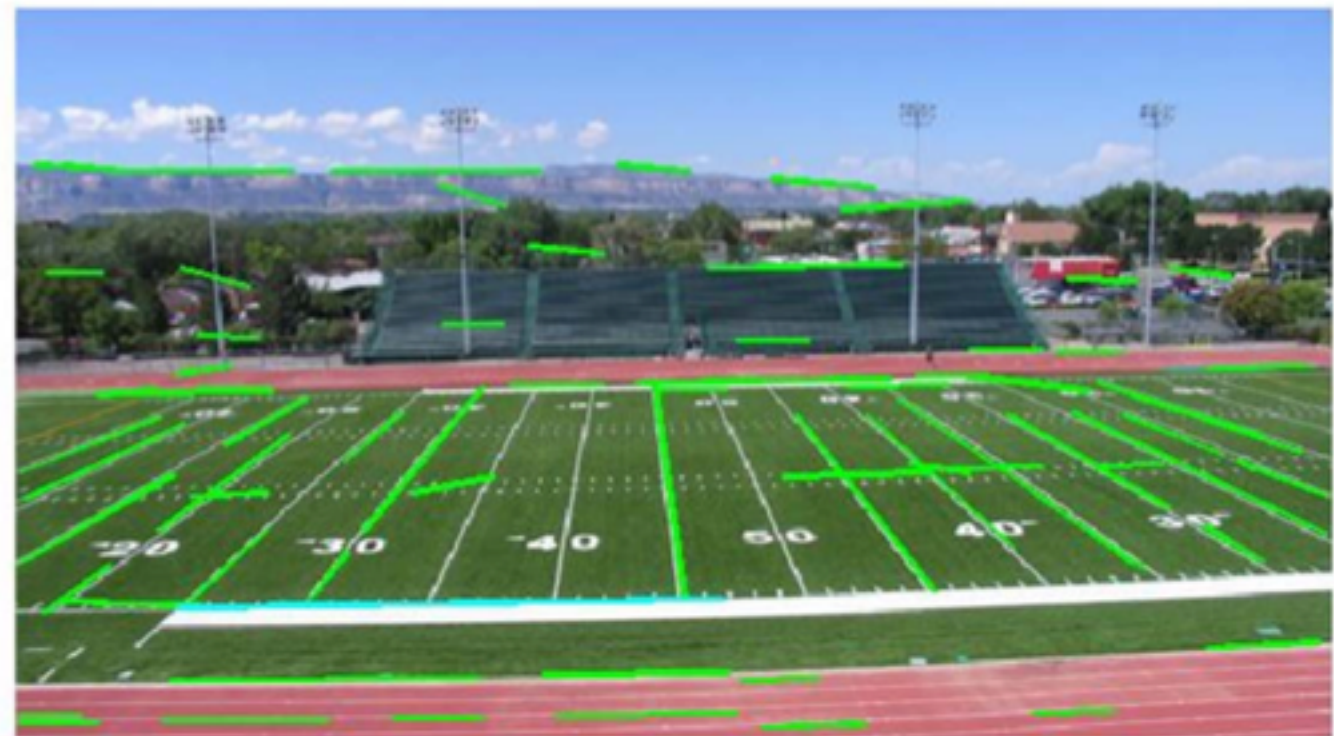
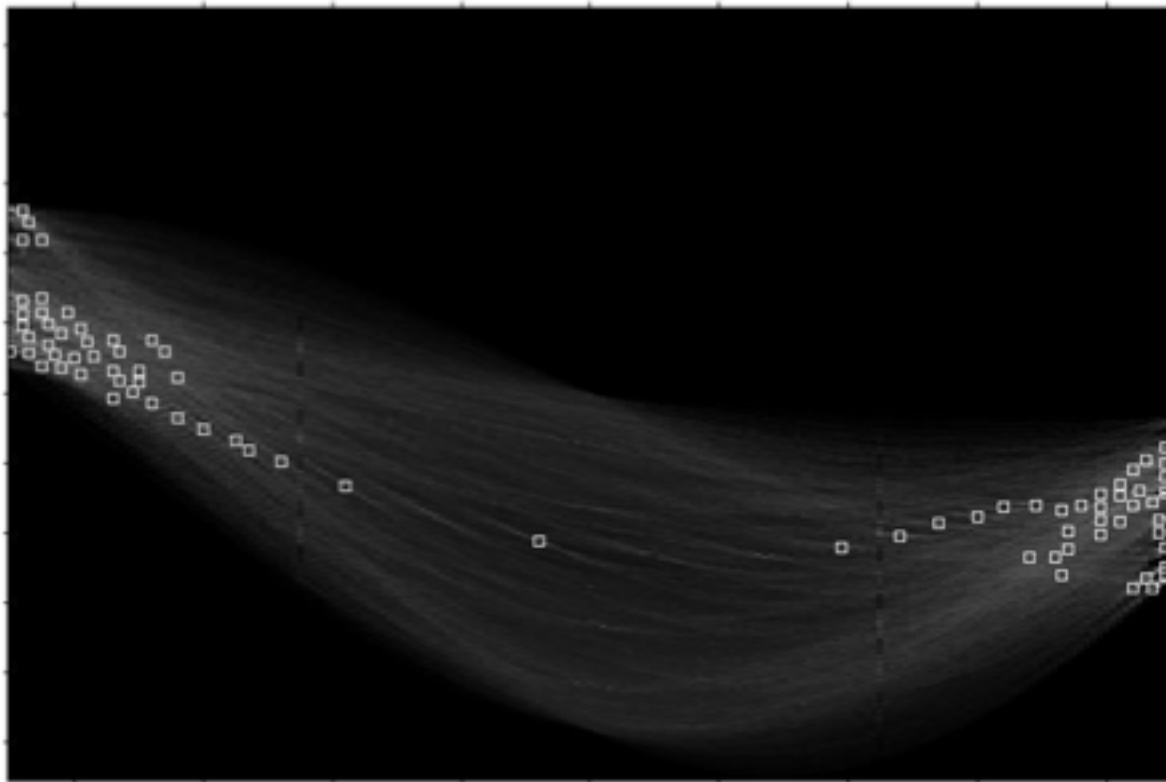
$$l_{a,b} : \quad ax + by = 1 \quad (\text{Assume } 0 \notin l)$$

$$(x_k, y_k) \in l_{a,b} \quad \Leftrightarrow \quad ax_k + by_k = 1$$

- Study the set

$$\Lambda_k = \{ (a, b) \mid (x_k, y_k) \in l_{a,b} \}$$

- This forms a line in the ab -plane.



Showing longest segments found

The least squares method

Line fitting

Assume that the points (x_i, y_i) are measured. Then

$$y_i = kx_i + l$$

for line parameters (k, l) .

Assume that:

- ▶ that the errors are only in the y -direction.
- ▶ the line is not vertical (since we are counting only vertical offsets from the line as errors, near vertical lines lead to quite large values of the error)

Line Fitting

Then

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} k \\ l \end{pmatrix} + \mathbf{n} = Ap + \mathbf{n}$$

If the errors \mathbf{n} are independent and Gaussian distributed, then it is reasonable to solve $y = Ap$ in least squares sense, i.e. minimizing $\|y - Ap\|$.

Line fitting

This least squares problem was studied in Lecture 2 (and in other courses).

The solution is

$$p = (A^T A)^{-1} A^T y$$

Write this out to obtain

$$\begin{pmatrix} k \\ l \end{pmatrix} = \begin{pmatrix} \bar{x}^2 & \bar{x} \\ \bar{x} & N \end{pmatrix}^{-1} \begin{pmatrix} \bar{x}\bar{y} \\ \bar{y} \end{pmatrix}$$

Least squares in Matlab

In matlab the least squares solution can be obtained using the slash function

$$p = A \backslash y$$

Read the help text 'help slash' for more information about how the 'slash'-operator works.

Total least squares

- ▶ One problem with the idea above lies in the two assumptions.
- ▶ It cannot handle vertical lines.
- ▶ For lines that are close to vertical the assumption that the errors are only in the y-direction gives sub-optimal estimates of the line.
- ▶ It is better to minimize the distance between the point and the line.

Solution

$$\begin{pmatrix} \bar{x}^2 - \frac{1}{N}\bar{x}\bar{x} & \bar{x}\bar{y} - \frac{1}{N}\bar{x}\bar{y} \\ \bar{x}\bar{y} - \frac{1}{N}\bar{x}\bar{y} & \bar{y}^2 - \frac{1}{N}\bar{y}\bar{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \lambda \begin{pmatrix} a \\ b \end{pmatrix}$$

is an eigenvalue problem.

There are two solutions, up to scale. Can be obtained in closed form.

The two solutions are orthogonal. One maximizes the likelihood, the other minimizes it.

It is straightforward to test which one of the two minimize $f(a, b, c)$.

Curve fitting

Similar ideas can be used to fit conics to points

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

or even higher order algebraic curves.

Outliers: M-estimators

A common method is to use an error function which is quadratic for small errors, but large for larger errors.

Then large errors (outliers) will not affect the fitting as much.

Instead of minimizing

$$\sum_i (ax_i + by_i + c)^2$$

we minimize

$$\sum_i \rho(ax_i + by_i + c, \sigma)$$

where e.g. one could use

$$\rho(u, \sigma) = \frac{u^2}{\sigma^2 + u^2}$$

Outliers: RANSAC

Another popular method to deal with outliers is RANSAC. It is an alternative to M-estimators (where we modified the underlying noise model to have heavier tails):

1. Randomly choose a minimal set of points needed for fitting.
2. Study how many points that now lie close to the line.
3. If there are sufficiently many, stop
4. Iterate 1-3 until stop, but at most k times.

Gray level transformations

Pixelwise operations

A simple method for image enhancement

Definition

Let $f(x, y)$ be the intensity function of an image. A **gray-level transformation**, T , is a function (of one variable)

$$g(x, y) = T(f(x, y))$$
$$s = T(r) \text{ ,}$$

that changes from gray-level f to gray-level g . T usually fulfils

- ▶ $T(r)$ increasing in $L_{min} \leq r \leq L_{max}$,
- ▶ $0 \leq T(r) \leq L$.

In many examples we assume that $L_{min} = 0$ och $L_{max} = L = 1$. The requirements on T being increasing can be relaxed, e.g. with inversion.

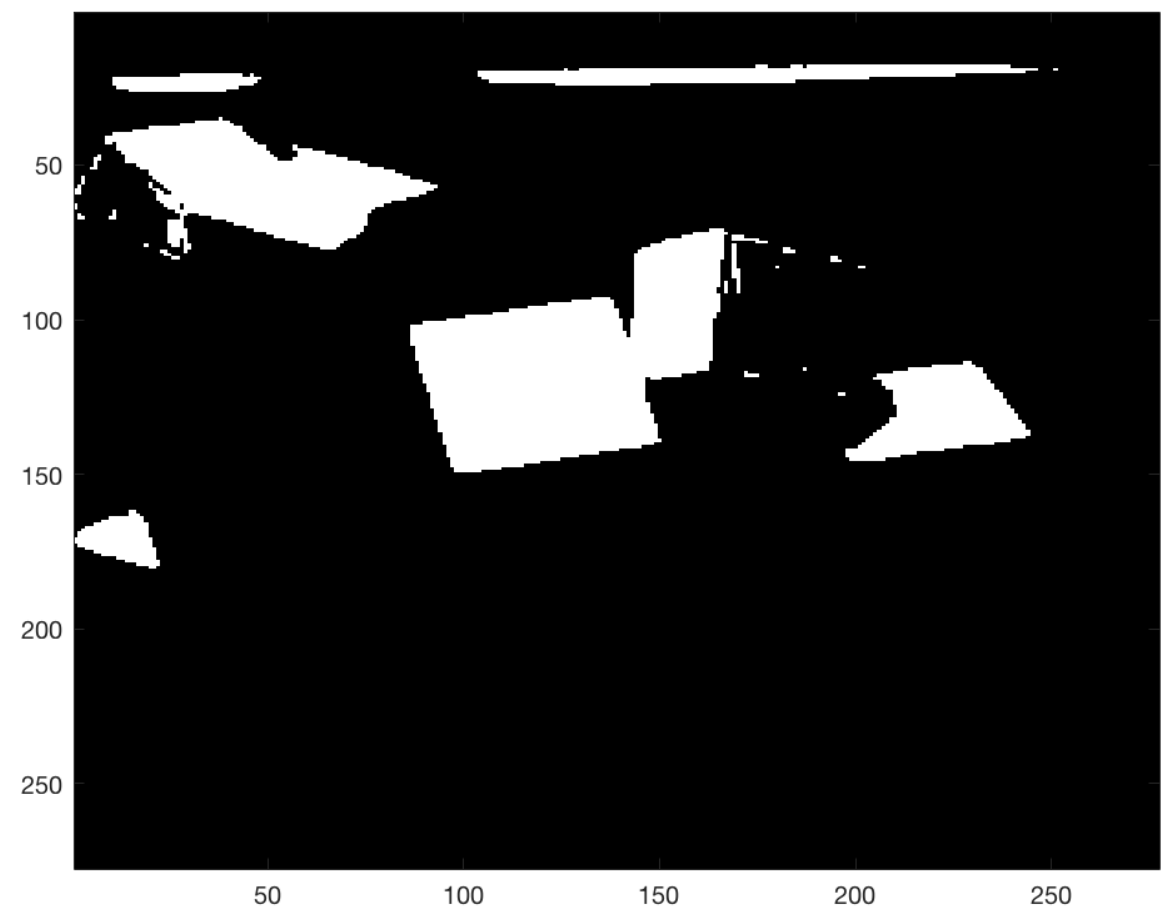
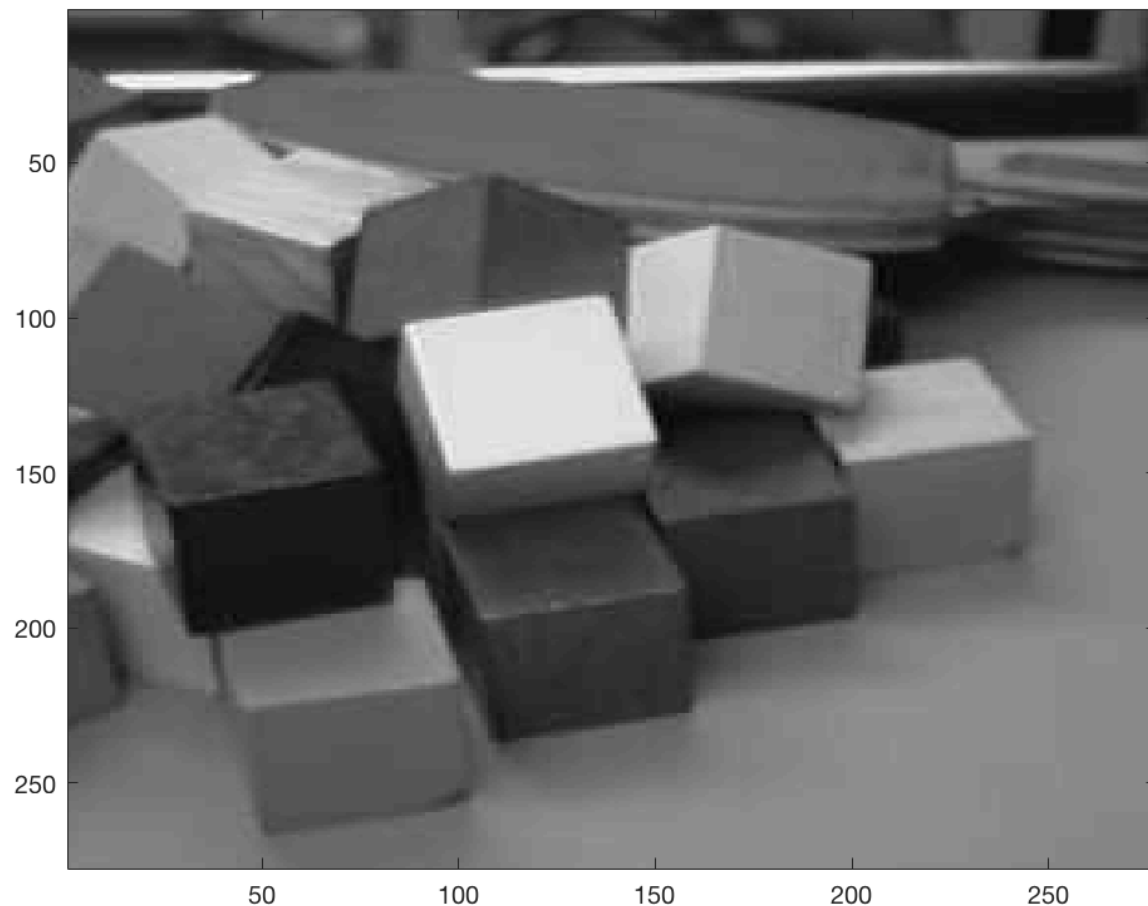
Gray level transformations

Pixelwise operations

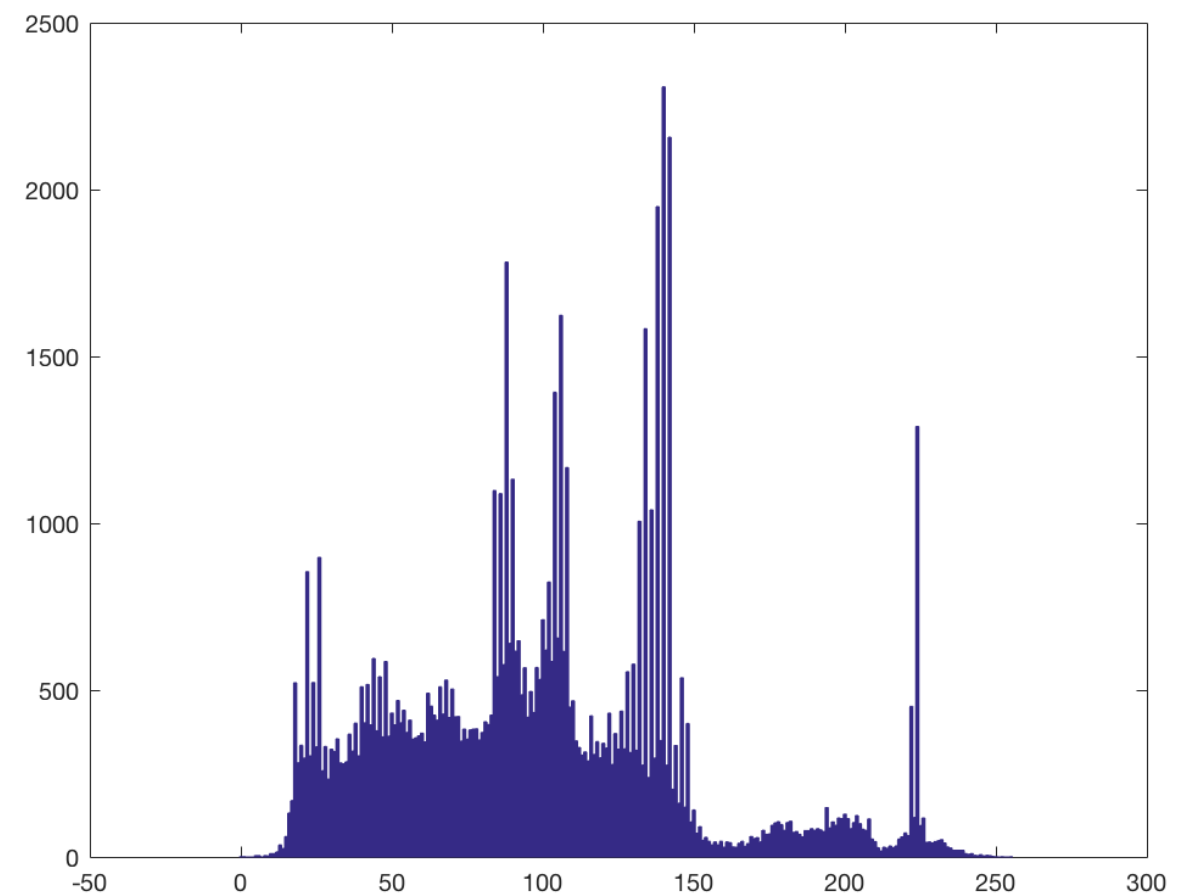
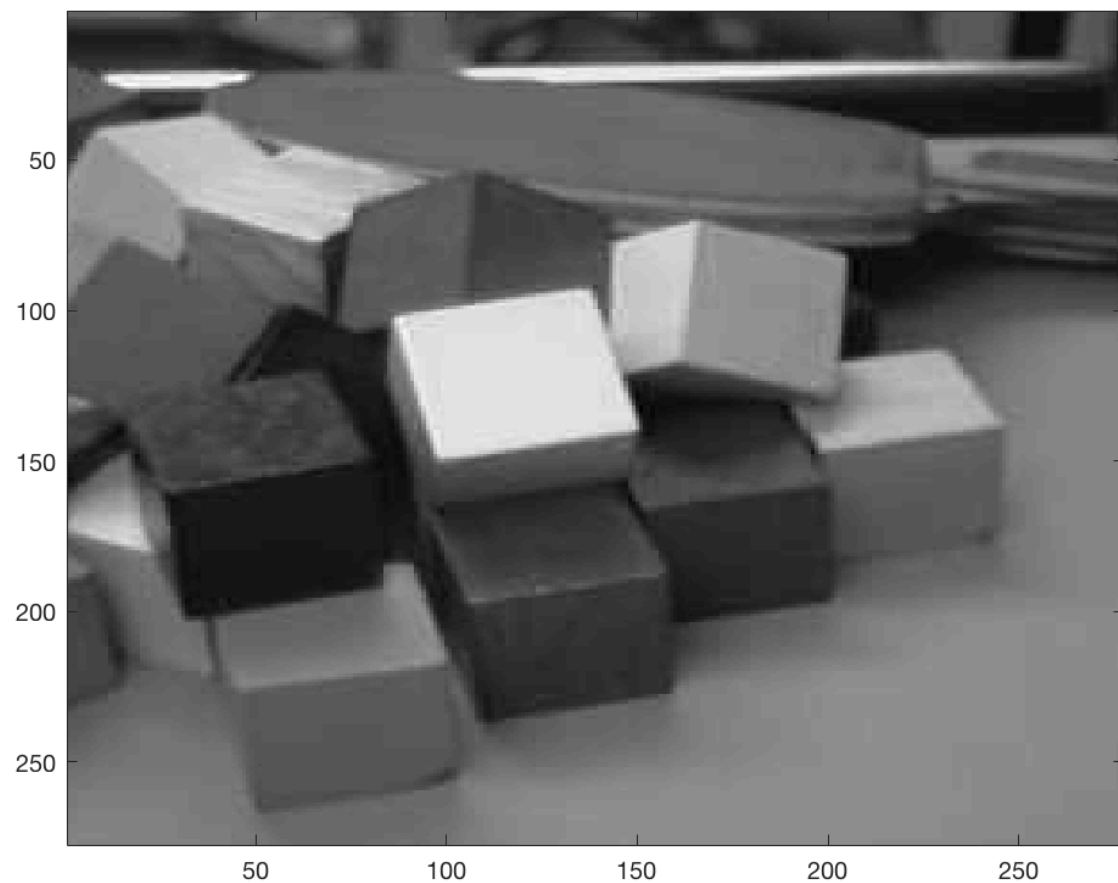
Let

$$T(r) = \begin{cases} 0 & r \leq m \\ 1 & r > m, \end{cases}$$

for some $0 < m < 1$.



Histograms



Histograms

- Let $s = T(r)$ be a gray level transformation
- Let p_r be the histogram before the transformation
- Let p_s be the histogram after the transformation
- Assume that T is a monotonically increasing function.
- The pixels that were darker than level r before are darker than s after.

It follows that

$$\int_0^s p_s(t) dt = \int_0^r p_r(t) dt.$$

Histograms

$$\int_0^s p_s(t) dt = \int_0^r p_r(t) dt.$$

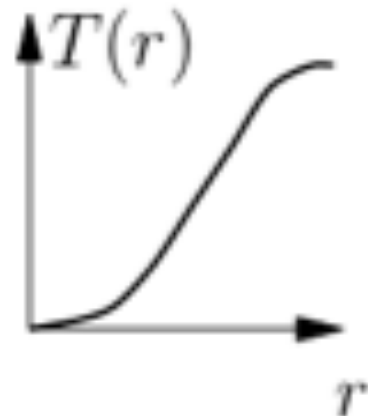
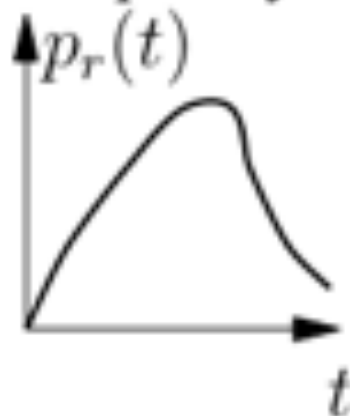
Take T so that $p_s(s) = 1$ (constant).

$$\int_0^r p_r(t) dt = \int_0^s 1 dt = s \Rightarrow s = T(r) = \int_0^r p_r(t) dt$$

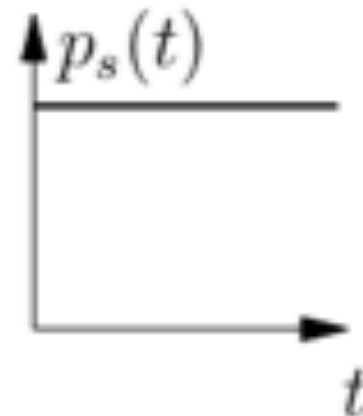
or

$$\frac{ds}{dr} = p_r(r)$$

frequency funct. transformation



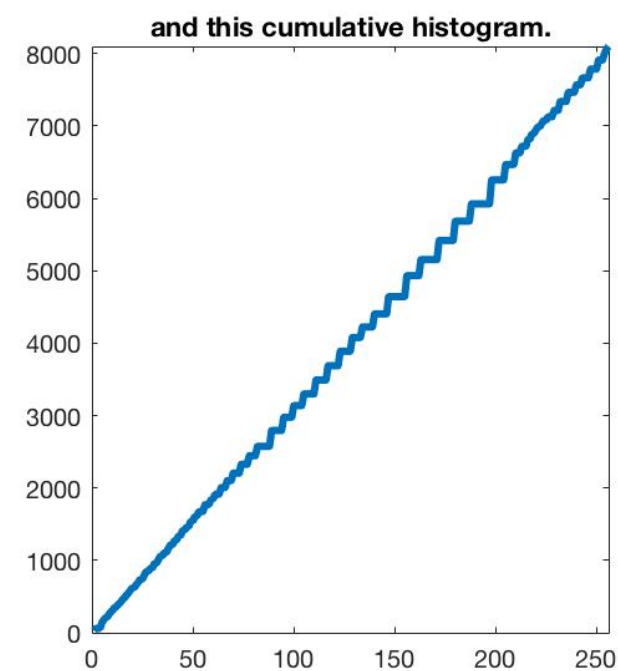
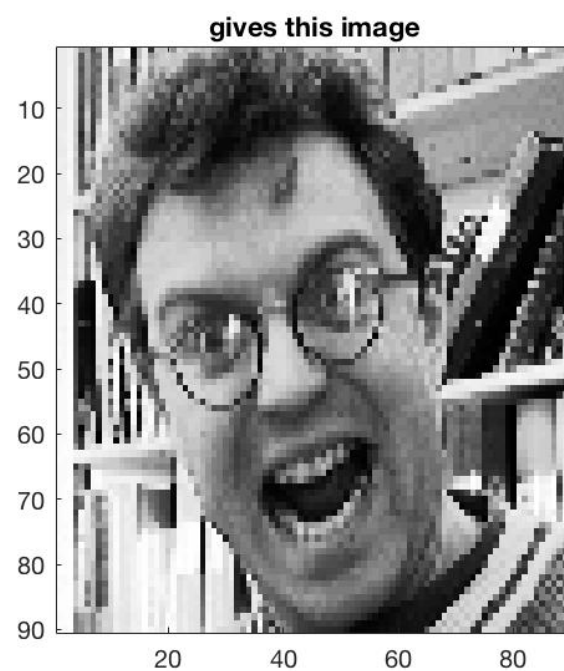
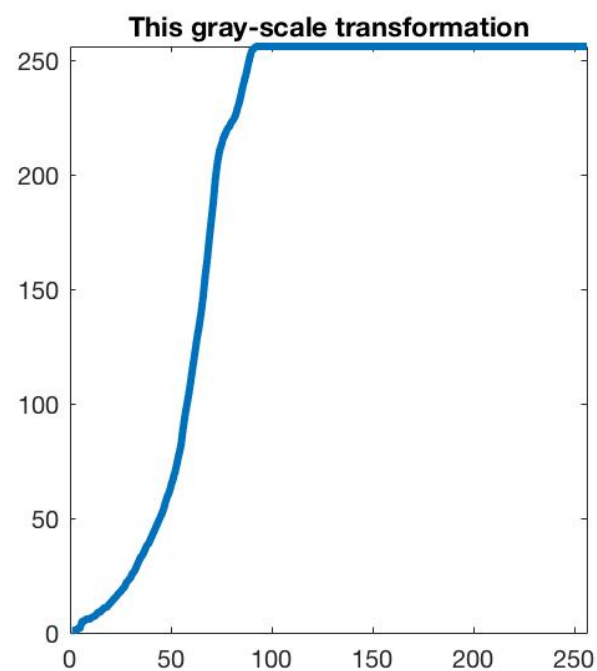
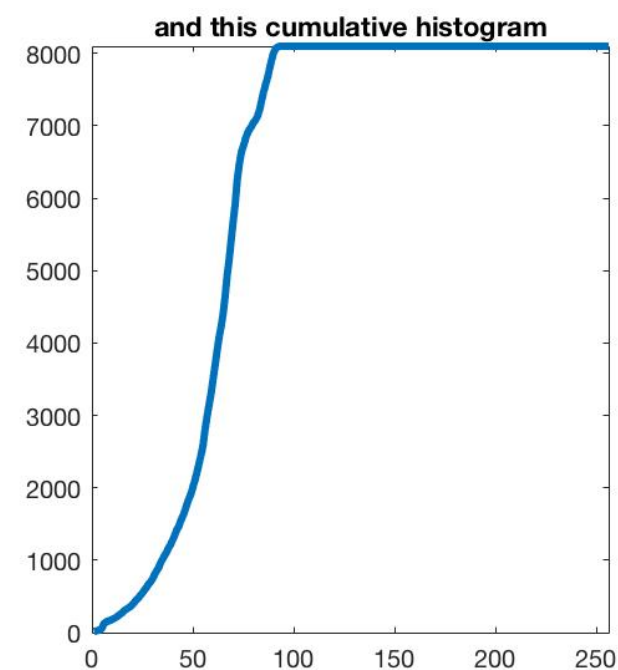
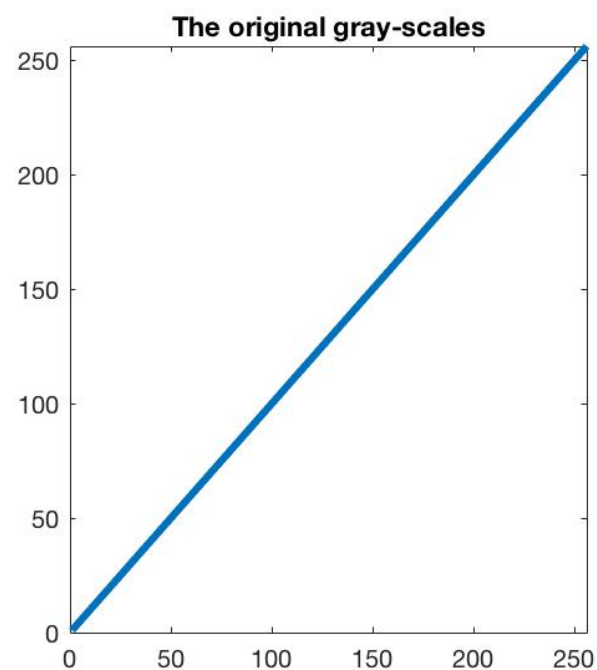
\Rightarrow

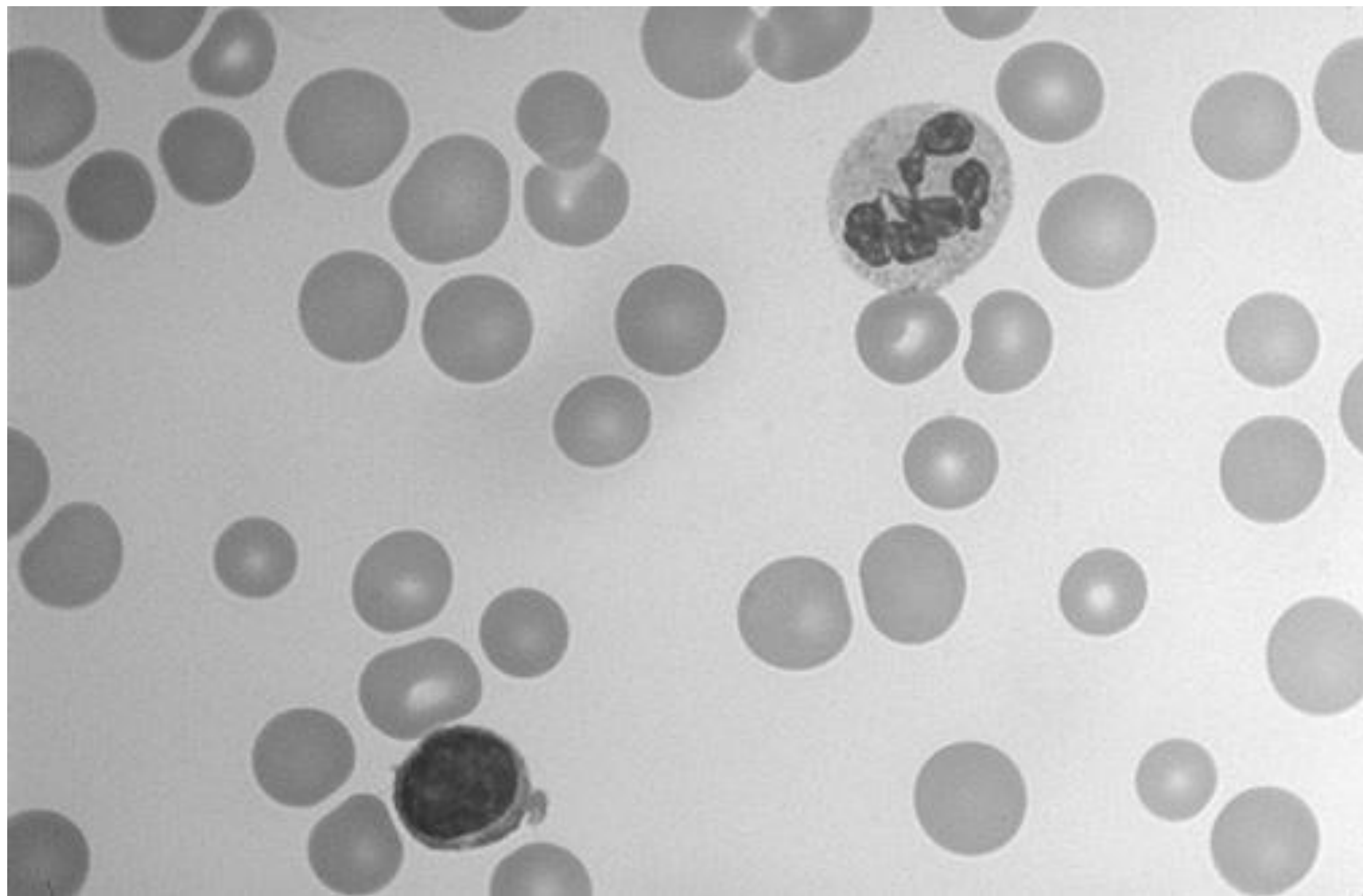


Histograms

$$\int_0^s p_s(t) dt = \int_0^r p_r(t) dt.$$

$$T(r) = \int_0^r p_r(t) dt$$





Mathematical Morphology

Definition

The **opening** of A with B is defined by

$$A \circ B = (A \ominus B) \oplus B .$$

Opening = first erosion, then dilation.

- ▶ Gives smoother contours.
- ▶ Removes narrow passages.
- ▶ Eliminates thin parts.

Mathematical Morphology

Definition

The **Closing** of A with B is defined by

$$A \cdot B = (A \oplus B) \ominus B .$$

Closing = first dilation, then erosion.

- ▶ Gives smoother contours.
- ▶ Fills up small parts.
- ▶ Fills up holes.

Image dilation



Image erosion

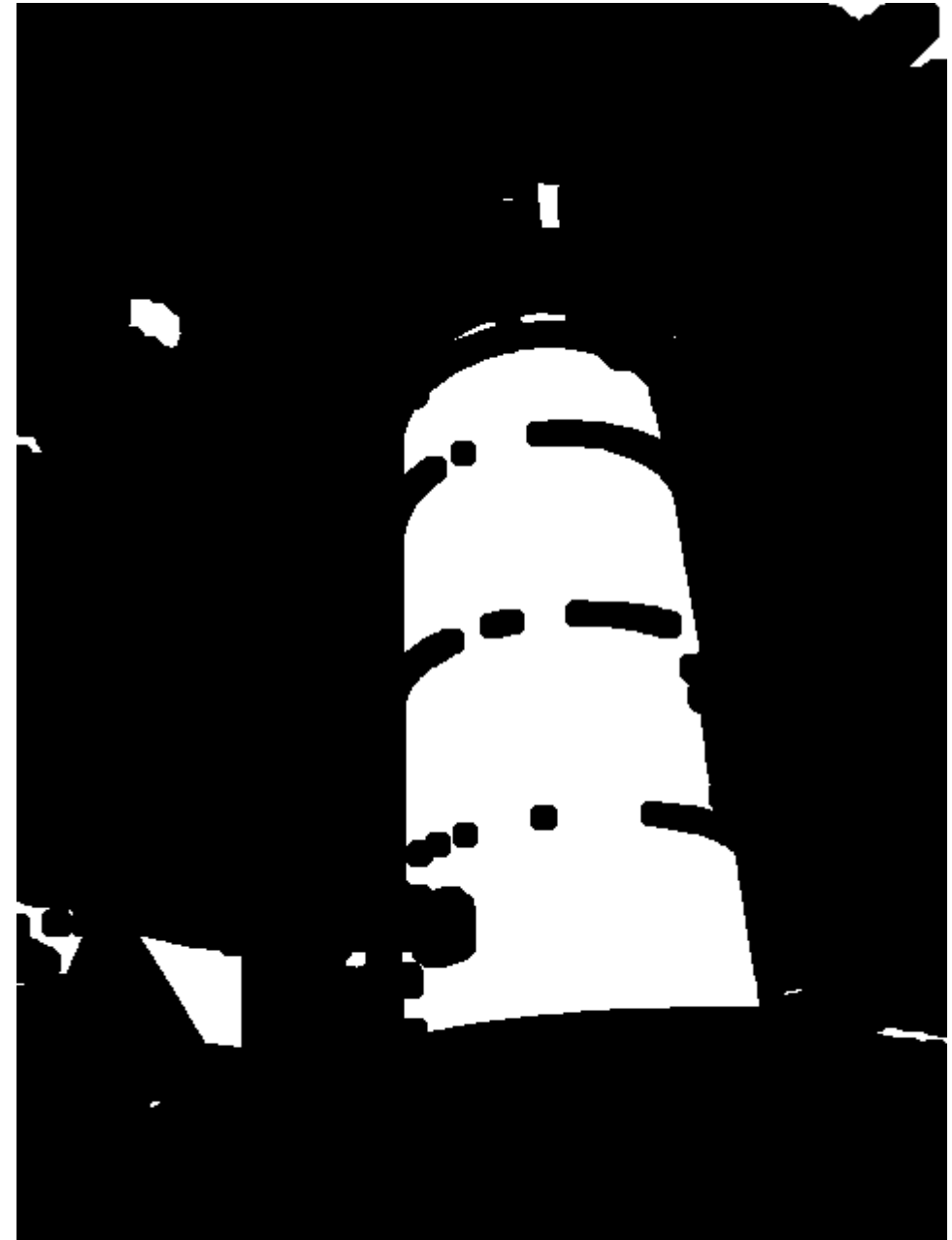


Image close

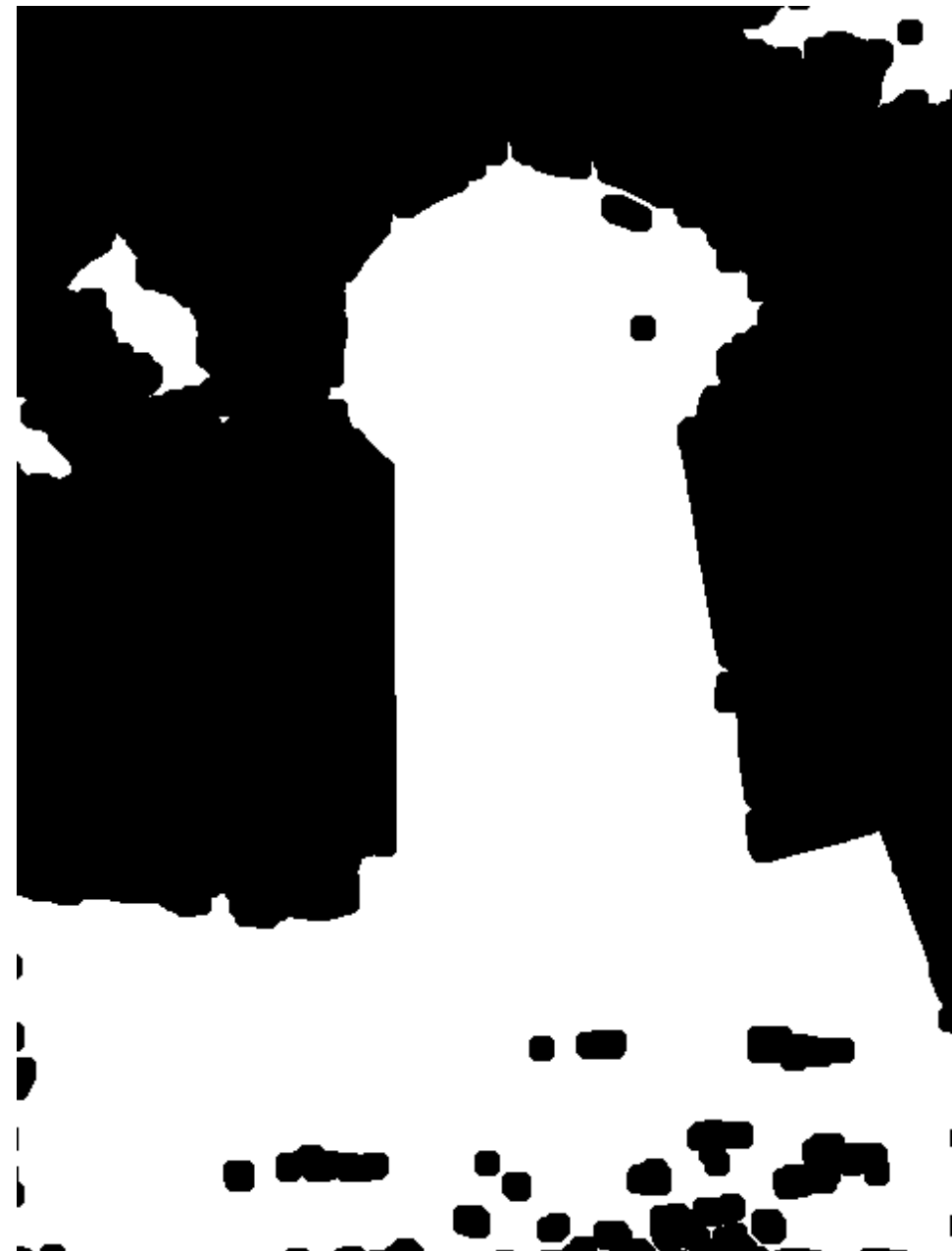
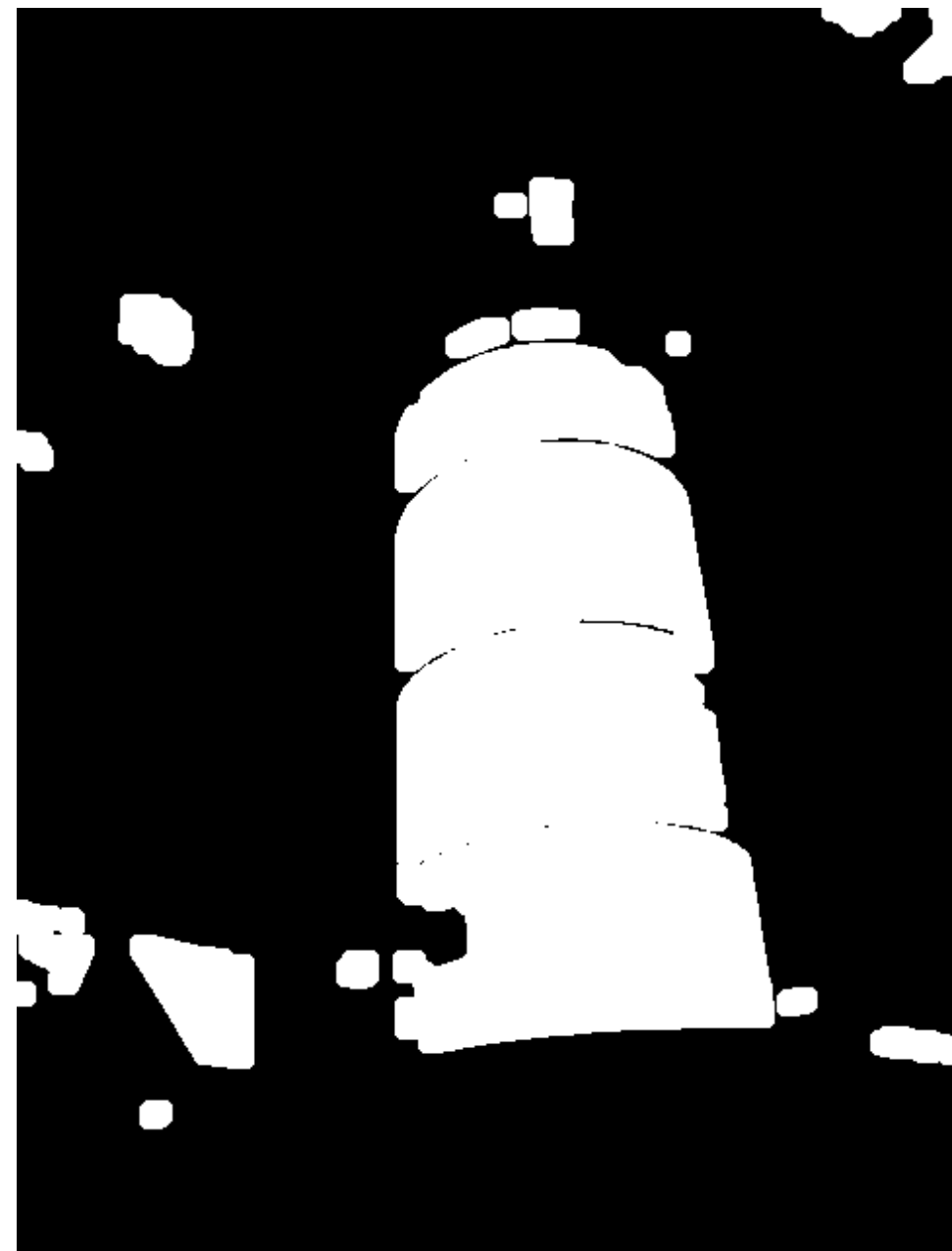
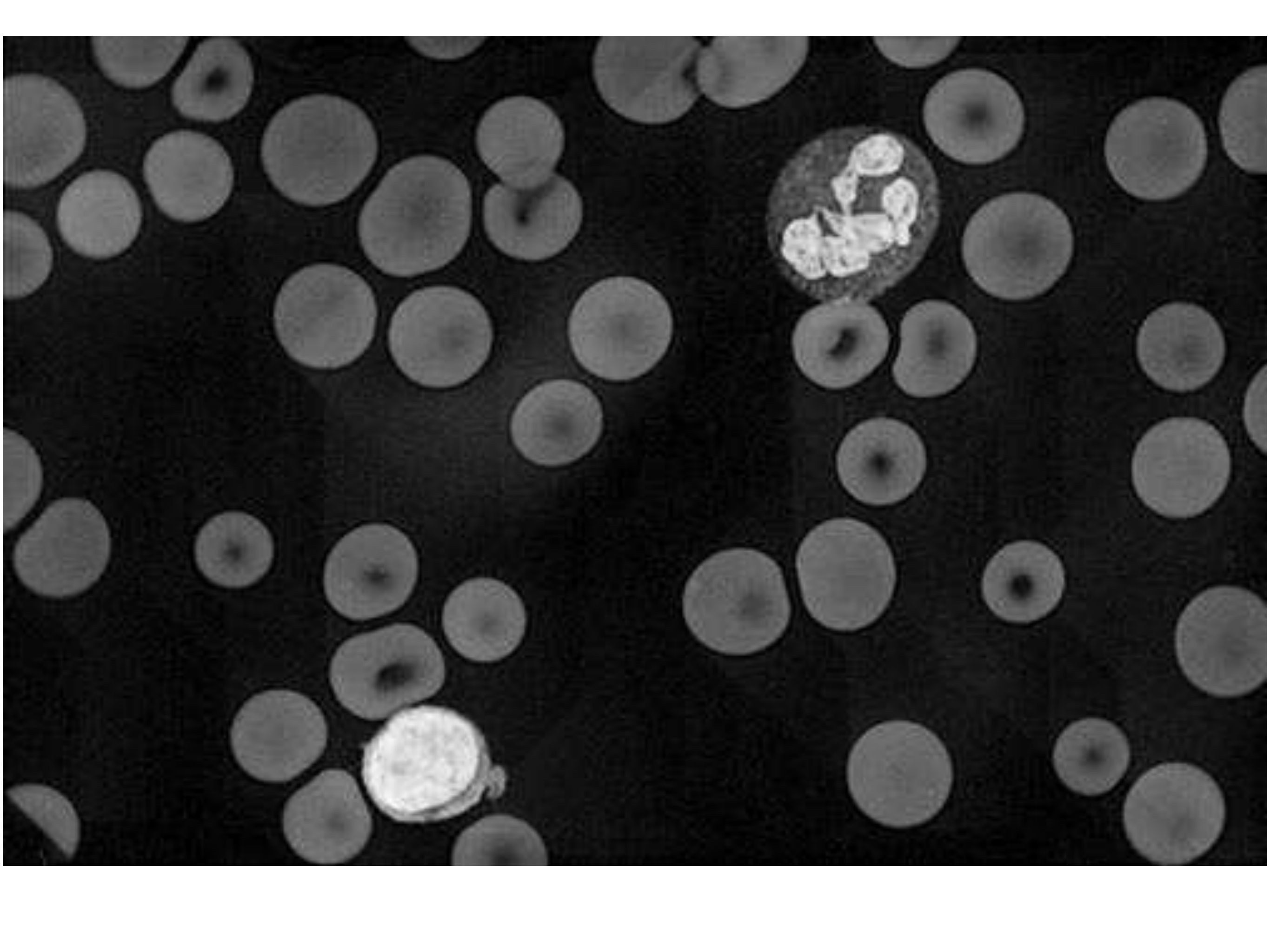
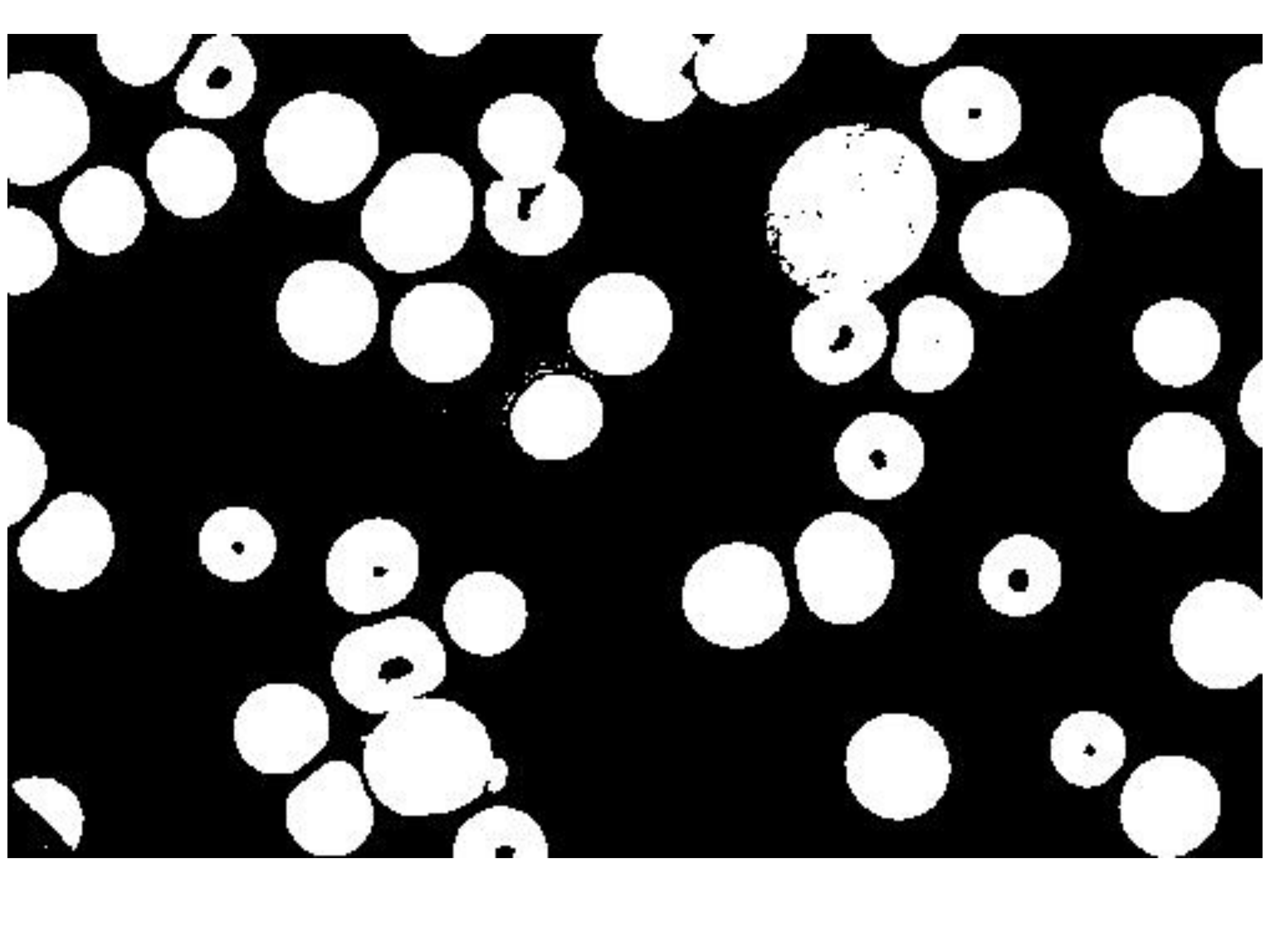


Image open









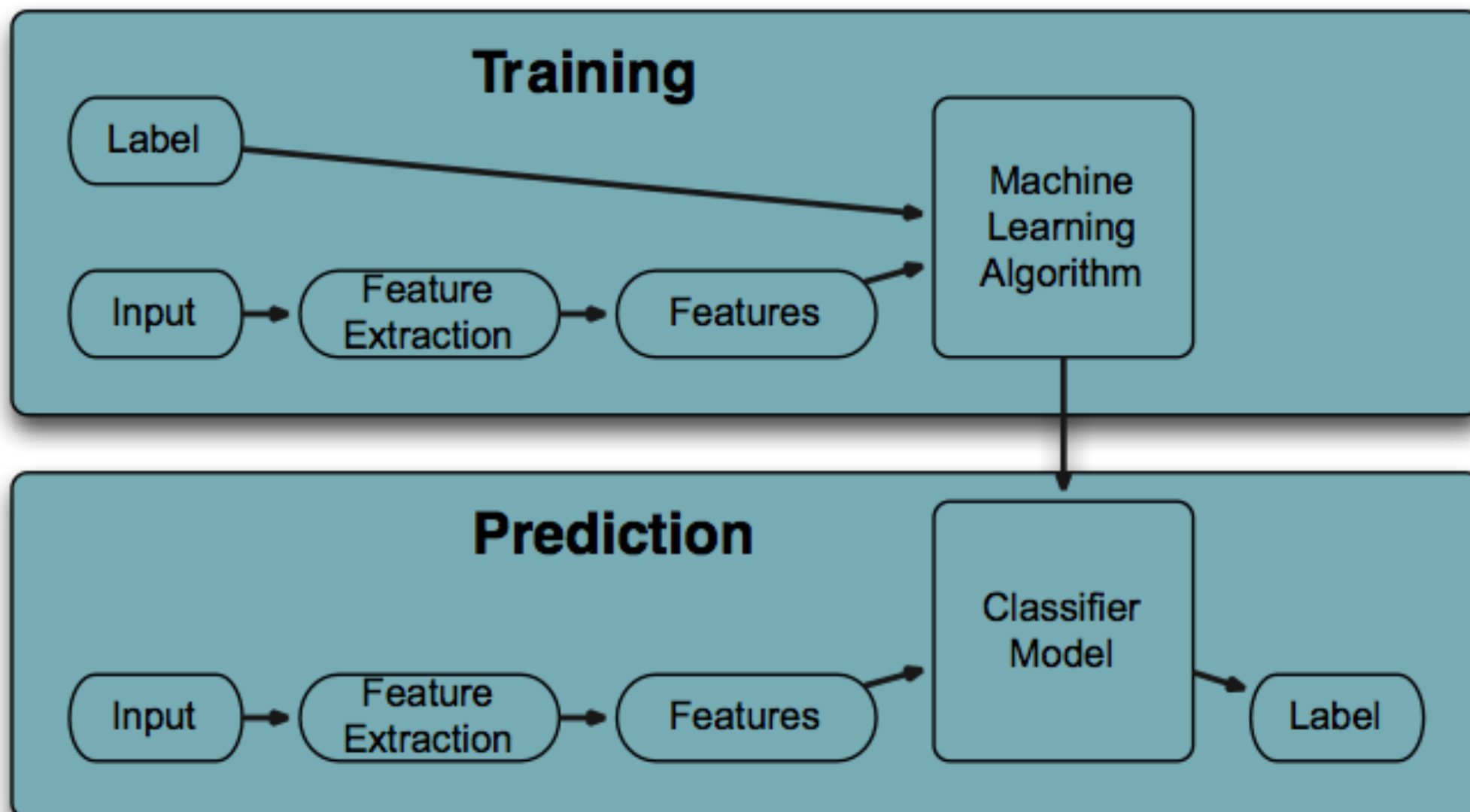






- Machine Learning typically has two phases
- Phase 1 – Training
 - A training dataset is used to estimate model parameters. Store these parameters. Code usually assumes that input are vectors
- Phase 2 – Prediction
 - Once the parameters have been estimated, we can use the model to classify future data

Input	Label
	Pear
	Apple
	Pear
	Pear
	Apple
	Apple



Machine Learning – classify

All of these classification problems have in common:

- ▶ data - \mathbf{x} (after segmentation, extract features)
- ▶ A number of classes

One would like to determine a class for every possible feature vector.

Here we will assume that the features are represented as a column vector, i.e. $\mathbf{x} \in \mathbb{R}^n$,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

One would like to compare the feature vector \mathbf{x} with those that one usually gets with a number of classes. Let y denote the class index, i.e. the classes are $y \in \omega_y = \{1, \dots, M\}$ where M denotes the number of classes.

Typical system: Image - filtering - segmentation - features - classification

Machine Learning – Bayes rule

Assume that one feature vector \mathbf{x} and class y are drawn from a joint probability distribution. If one can calculate the probability that the class is $y = j$ given the measurements \mathbf{x} , i.e. the so called **posterior probability**.

$$P(y = j|\mathbf{x})$$

The **maximum a posteriori classifier** is obtained as selecting the class j that maximizes the posterior probability, i.e.

$$j = \operatorname{argmax}_k P(y = k|\mathbf{x}).$$

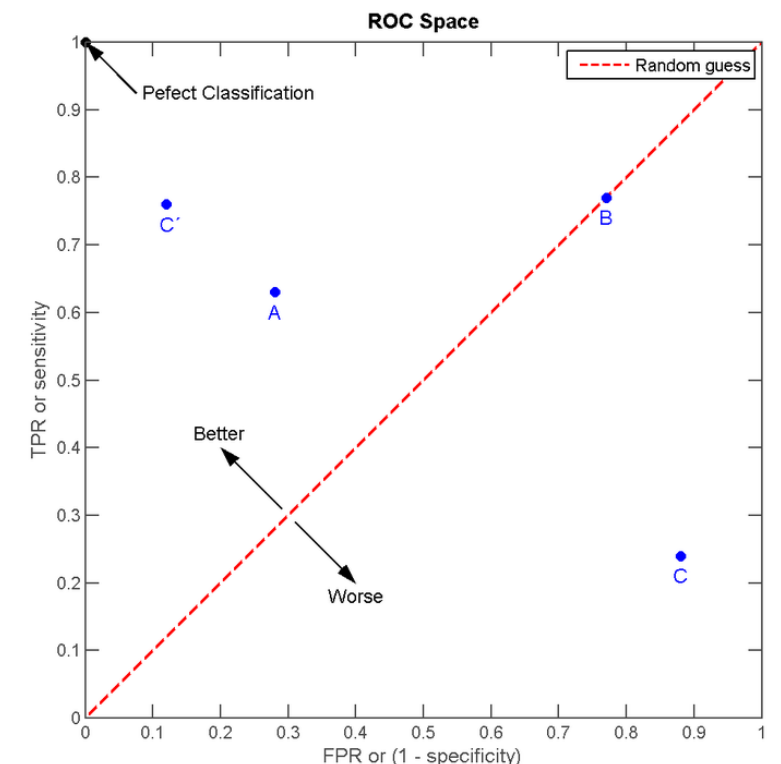
It is often easier to model and estimate the **likelihood** $P(\mathbf{x}|y = j)$ and to model the **prior** $p(y = j)$. The a posteriori probabilities can then be calculated using the Bayes rule,

$$p(y = j|\mathbf{x}) = \frac{p(\mathbf{x}|y = j)p(y = j)}{p(\mathbf{x})}.$$

False Positives, False Negatives

ROC - Curve

- For two class problems - Negatives and Positives
 - Negatives that are classified as negatives – True Negatives (TN)
 - Positives that are classified as positives – True Positives (TP)
 - Negatives that are classified as positives – False Positives (FP)
 - Positives that are classified as negatives – False Negatives (FN)
-
- False Positive Rate $FPR = FP/(FP+TN)$ -> x-axis
 - True Positive Rate $TPR = TP/(TP+FN)$ -> y-axis

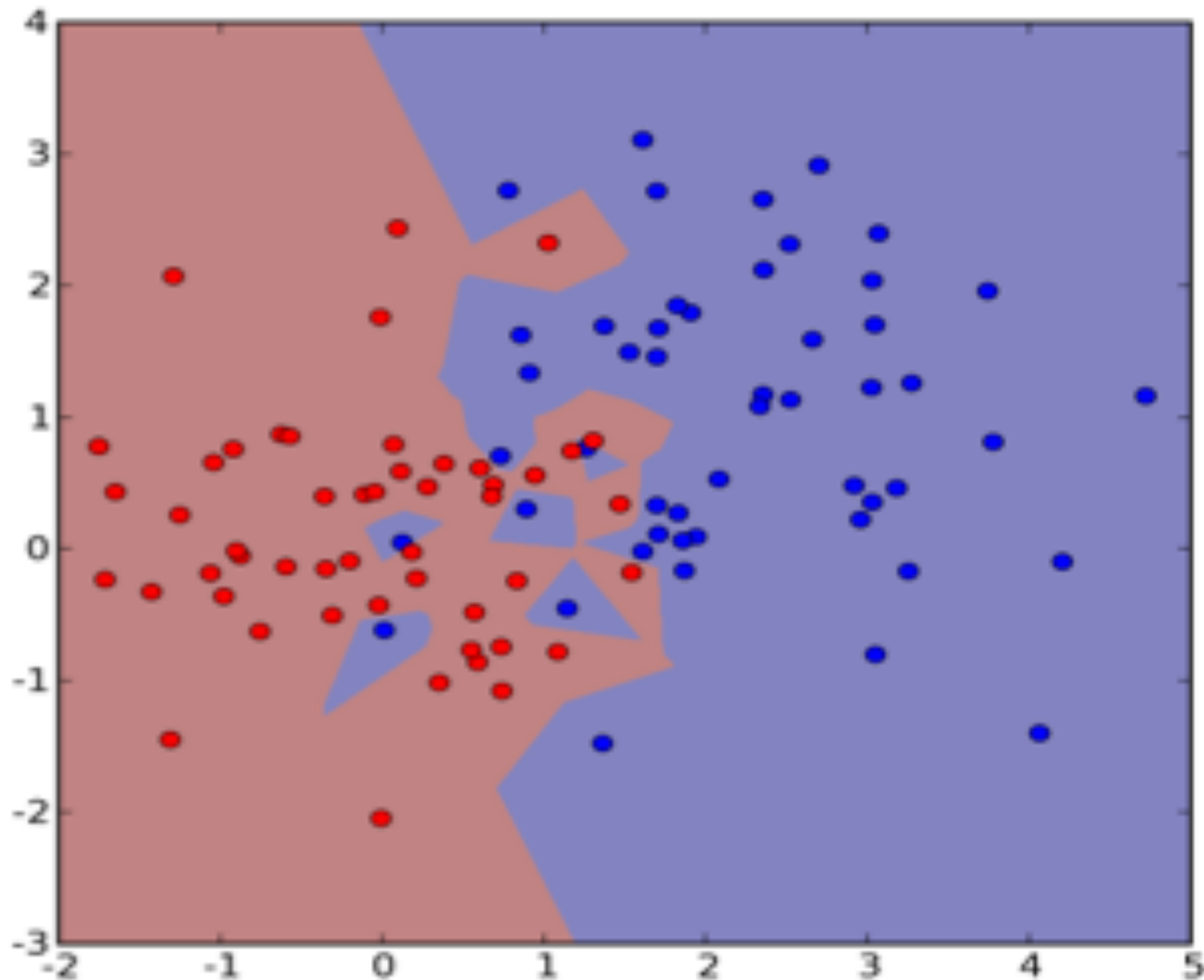


Nearest Neighbour Classification

NN and K-NN

- Classify using training data (x_i, y_i)
- NN: Use the label of the nearest neighbour
- KNN: Use the label of the majority of the k nearest neighbours
- Regression: Use the average of the value of the k nearest neighbours
- Easy to implement and understand
- Can use arbitrary distance functions between images
- Converges to the optimum
- Slow when using lots of data, need to store all training data, not smooth regression

Nearest Neighbour Classification (discussion)



Logistic regression

- Linear logistic regression
- Estimate the posterior
- As linear function followed by standard logistic function
- Convex optimization problem

$$\min_{\mathbf{w}} \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \log(1 + e^{-y_i \mathbf{w}^T \mathbf{x}_i}).$$

- Then classify according to

$$P(Y = y | X = x) = s(\mathbf{w}^T x + b)$$

Standard
logistic function

$$s(z) = \frac{1}{1 + e^{-x}}$$

Single Layer Neural Networks

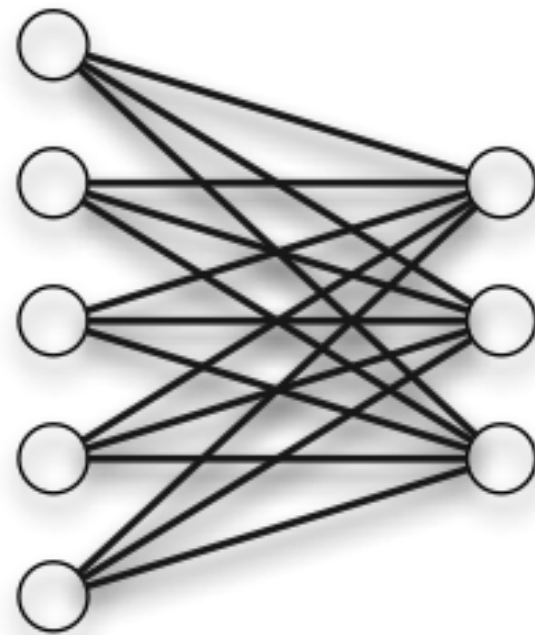
Several Neurons

- Several parallel neurons

$$x \in R^d, y \in R^k, B \in R^d, W - k \times d \text{ matrix}$$

$$y = s(Wx + B)$$

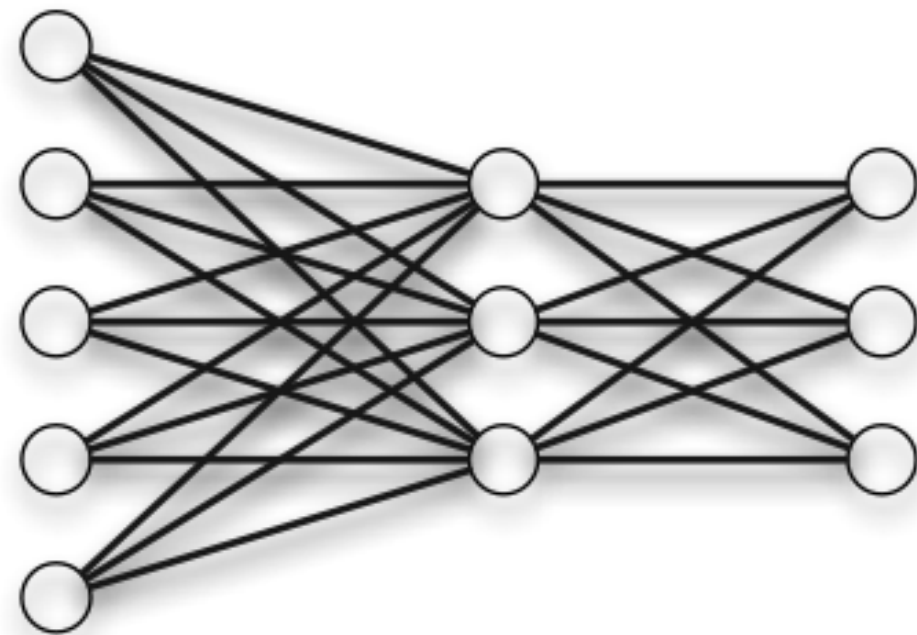
- Elementwise smooth thresholding – s



Artificial Neural Networks

One hidden layer

- Idea: Apply a new set of neurons on the output of the first set of neurons.
- Two layers
- Multi-class classification
- One hidden layer
- Not a convex optimization problem
- Can get stuck in local minima
- Trained by back-propagation
- Popular since the 1990ies

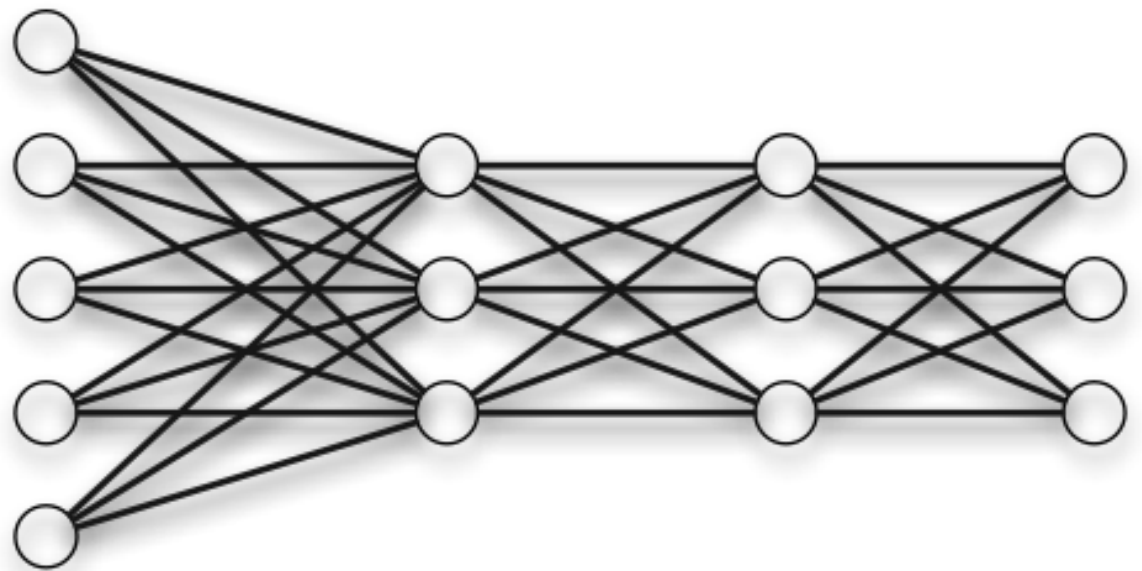


$$y = s(Wx + B)$$

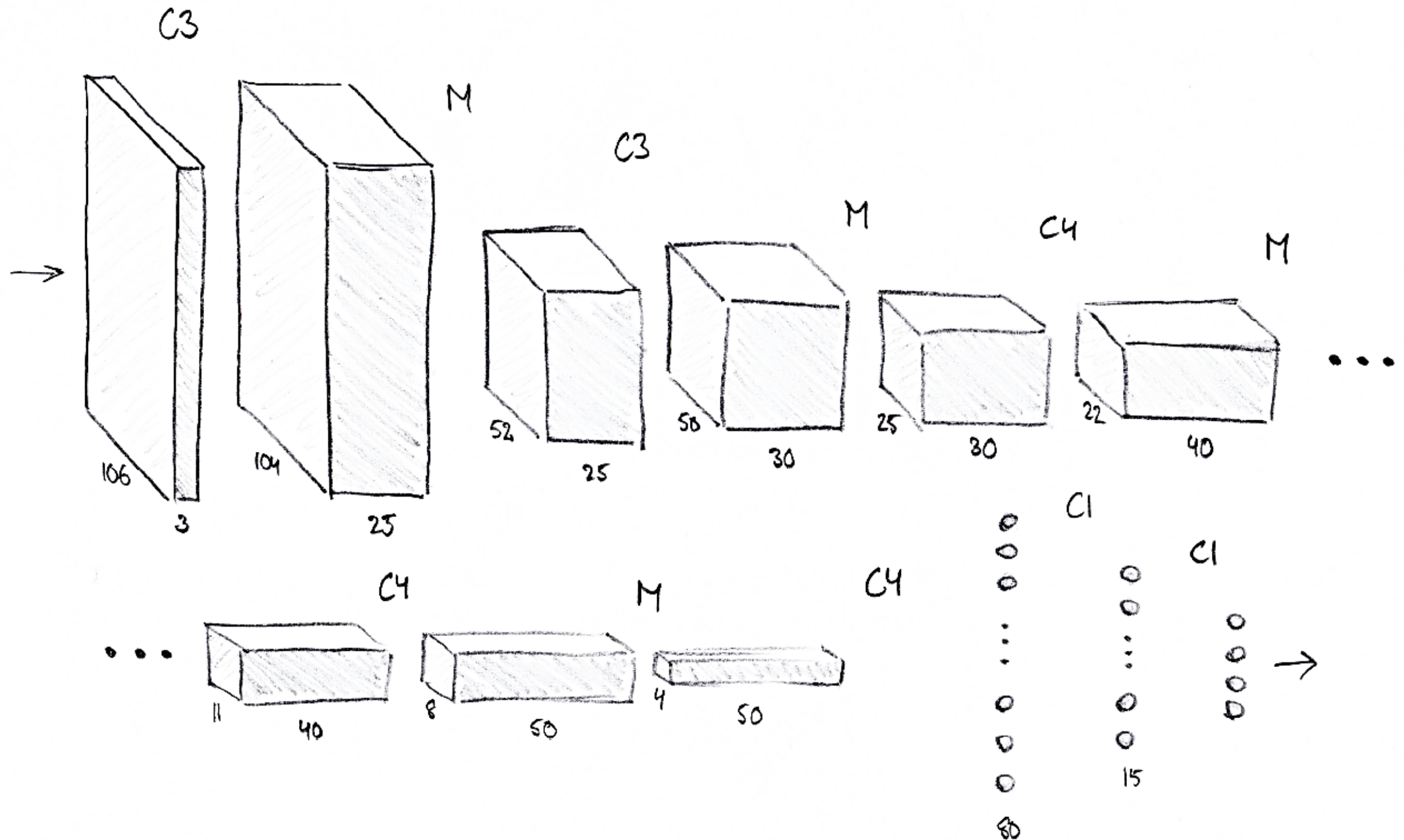
Deep Neural Networks

Many layers

- Deeper networks. Apply a third layer on the output of the first two.
- Naively implemented would give too many parameters
- Example
- 1M pixel image
- 1M hidden layers
- 10^{12} parameters between each pairs of layers

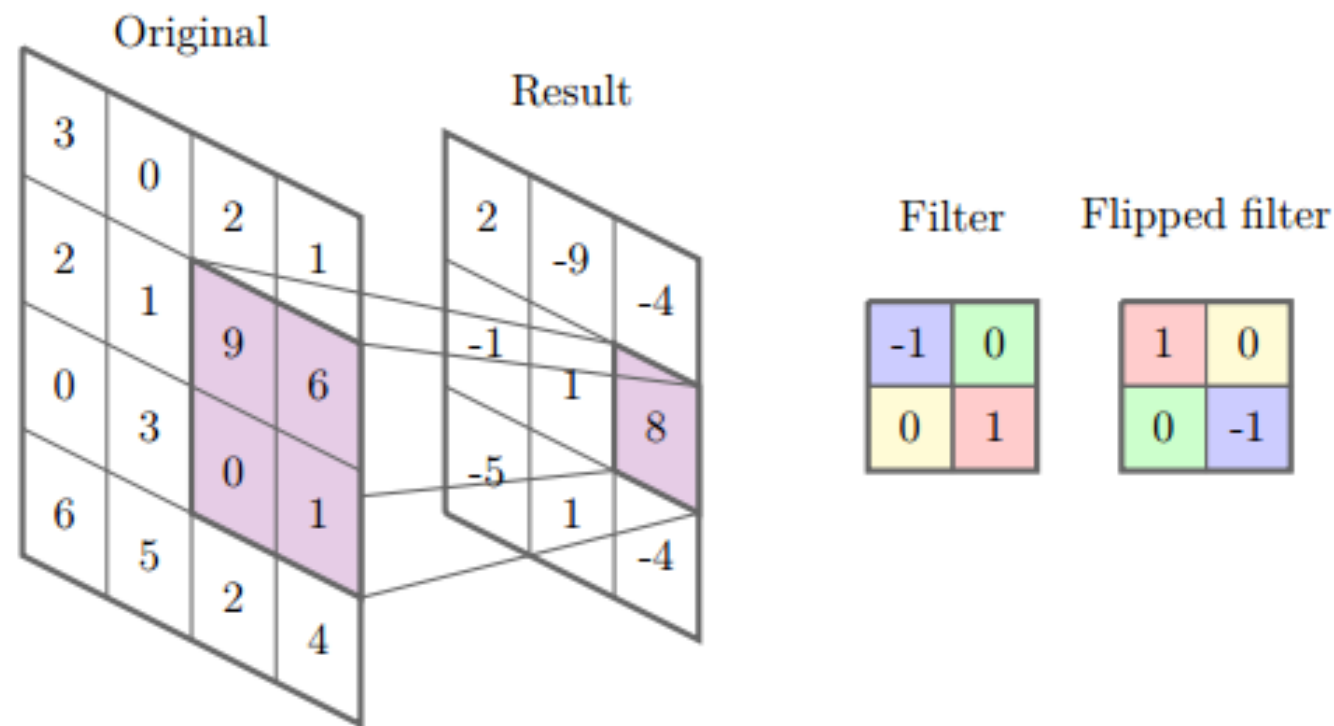


Convolutional Neural Networks

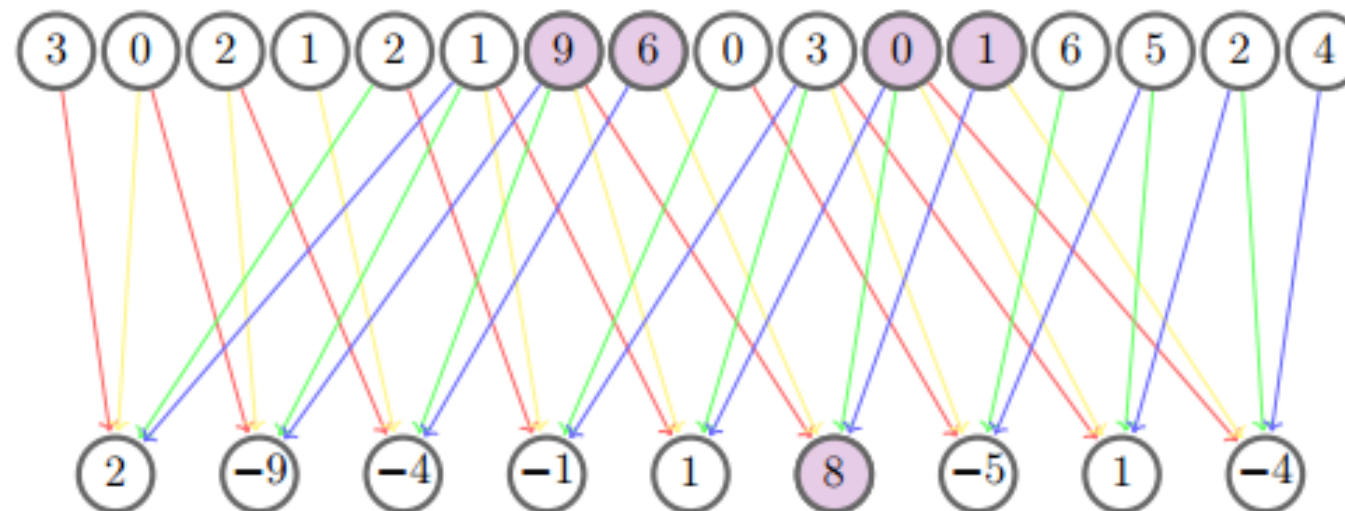


CNN-Blocks - Convolutional layer

Convolution of an image as a filter-operation.

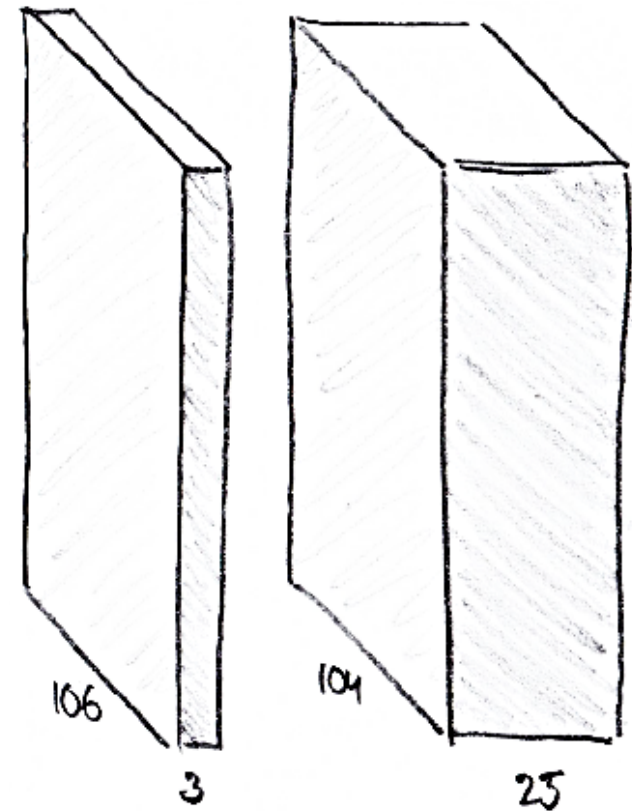


Convolution of an image represented as a sparsely connected ANN.



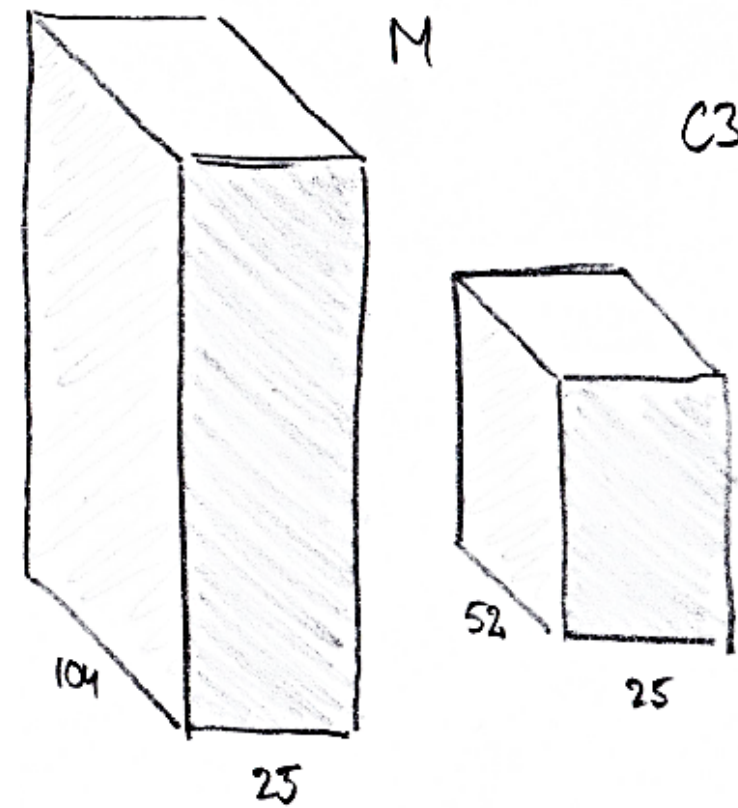
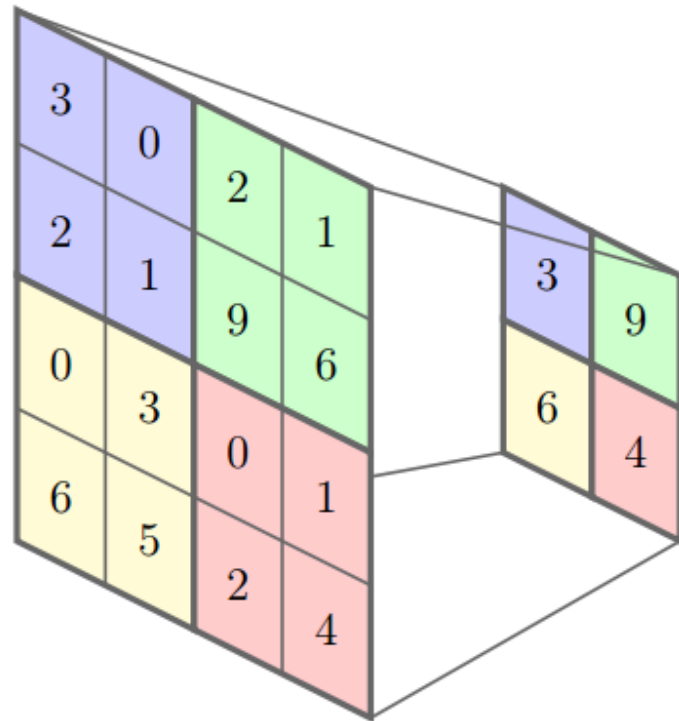
CNN-Blocks - Convolutional layer

- 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 kernel block w of size $m_w \times n_w \times k_1 \times k_2$
- Offsets: Vector w_o of length k_2



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

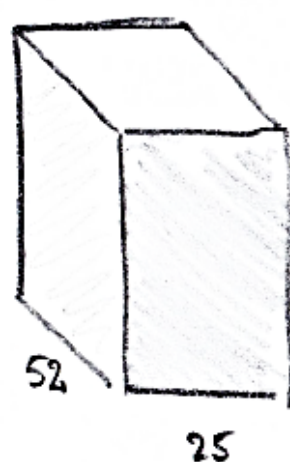
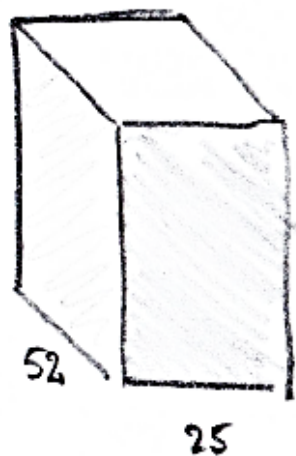
CNN-Blocks - Max-pooling



CNN-Blocks - RELU

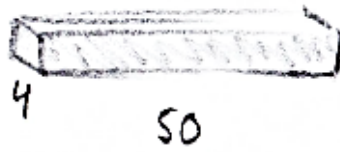
$$f(x) = \max(0, x)$$

$$y(i, j, k) = \max(x(i, j, k), 0)$$



CNN-Blocks – Softmax
(convert from 'log probabilities' d_j to
'probabilities' that sum to 1)

$$p_j = \frac{e^{d_j}}{\sum_{k=1}^m e^{d_k}}$$

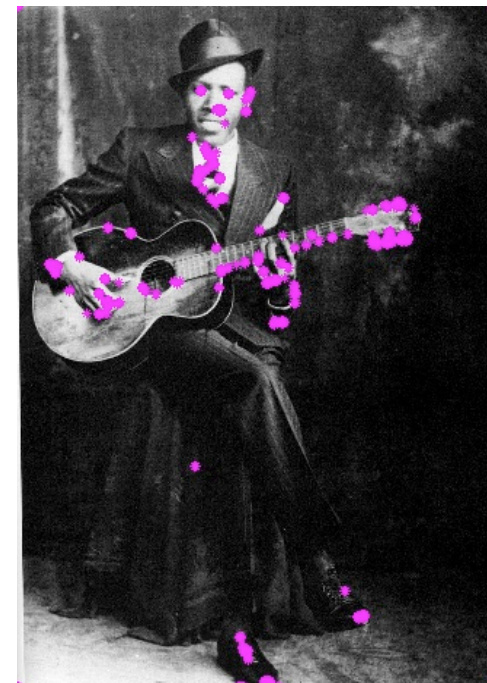


Testing your system

- Image analysis systems
 - Often complex and varying data
 - Often a system of systems
- Important to test your system
- Questions
 - Obtain **data**
 - Obtain '**ground truth**' ('Gold Standard')
 - Construct **benchmark scripts**
 - **Visualize** the results
- Address these questions **early** in a project

Local Features

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



Structure Tensor

The matrix M has the following properties:

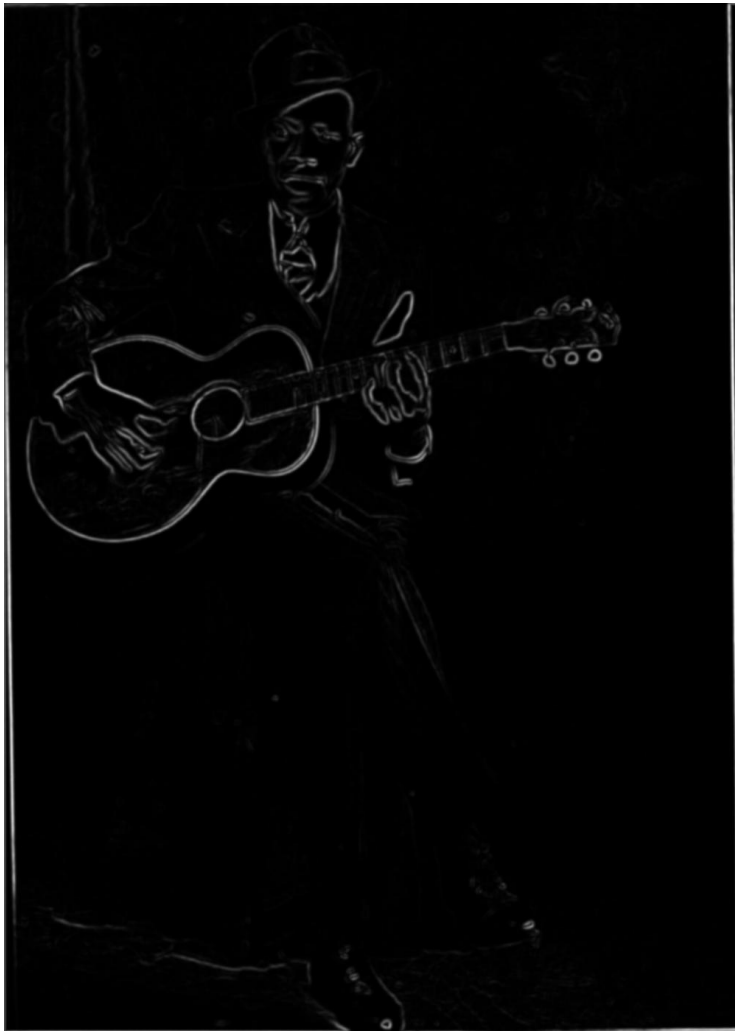
- ▶ (Flat) Two small eigenvalues in a region - flat intensity.
- ▶ (Flow) One large and one small eigenvalue - edges and flow regions.
- ▶ (Texture) Two large eigenvalues - corners, interest points, texture regions.

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

Corner Detector

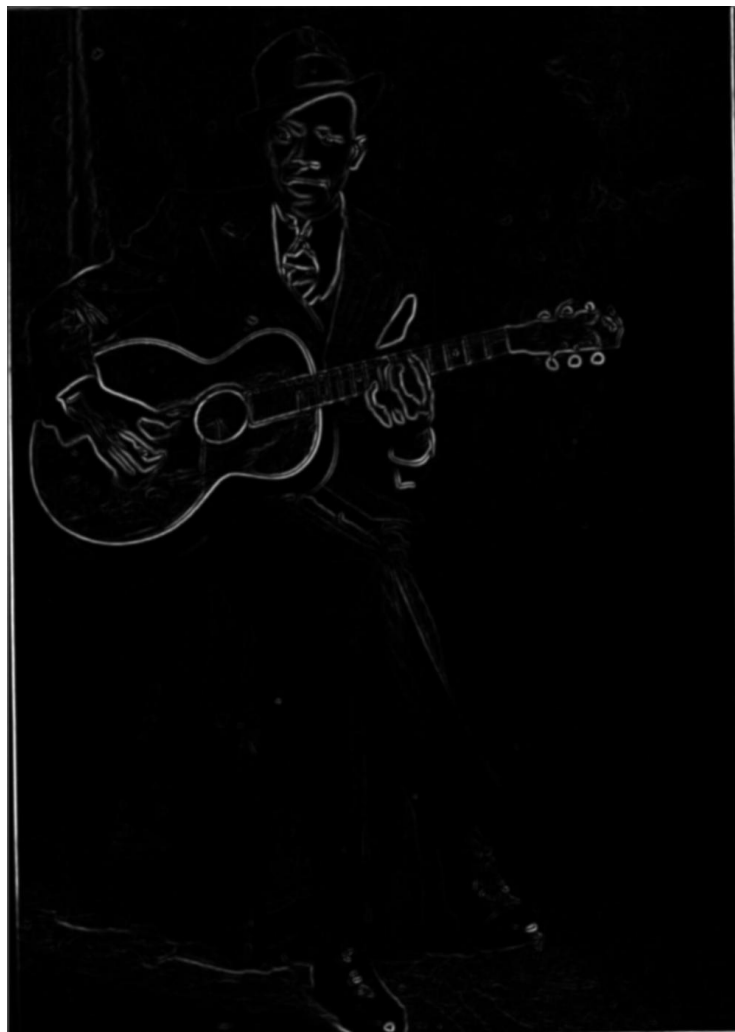
- Compute x- and y-derivatives with a Gaussian filter
- Form the orientation tensor M for every pixel
- Compute the product of eigenvalues, i.e. the determinant of M
- If both eigenvalues large (product is a local maximum), then it is a corner!

Harris Corner Detector



Eigenvalue two of
the orientation tensor

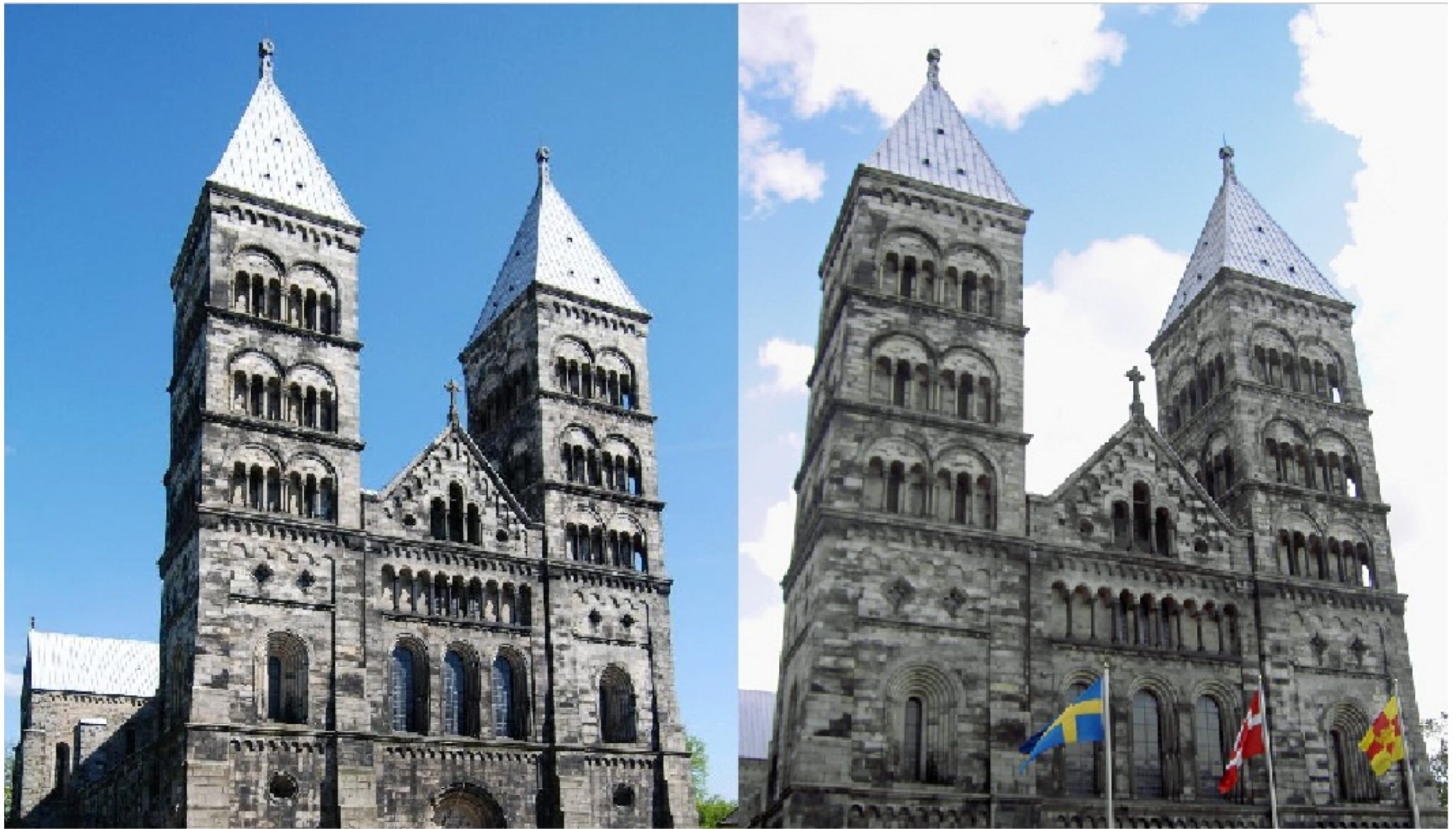
Harris Corner Detector



Eigenvalue two of
the orientation tensor



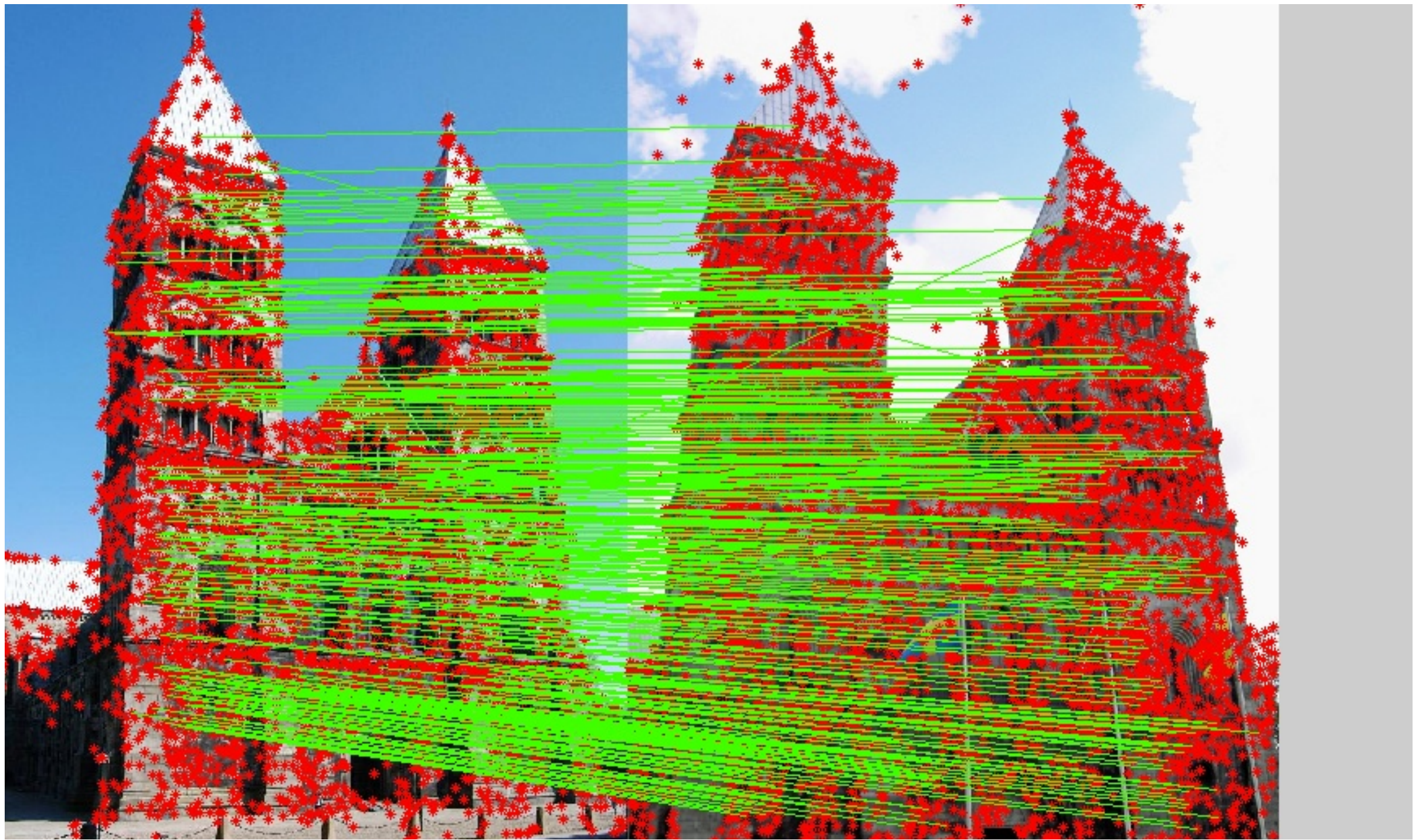
Two large Eigenvalues
Gives a corner



Building a three-dimensional model.
Start with two images.



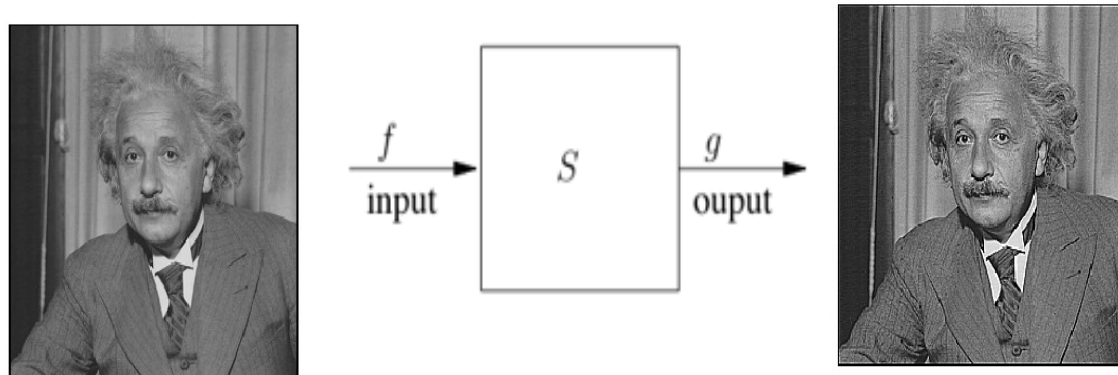
Building a three-dimensional model.
Find keypoints, using e.g SIFT



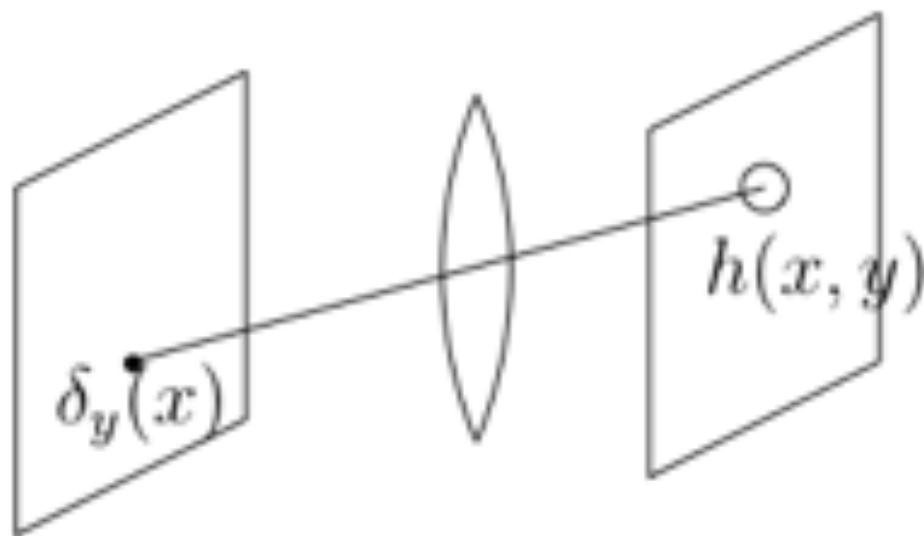
Building a three-dimensional model.
Match keypoints using e.g. RANSAC.

Convolutions and linear systems

Any linear and translation invariant system can be represented as a convolution.



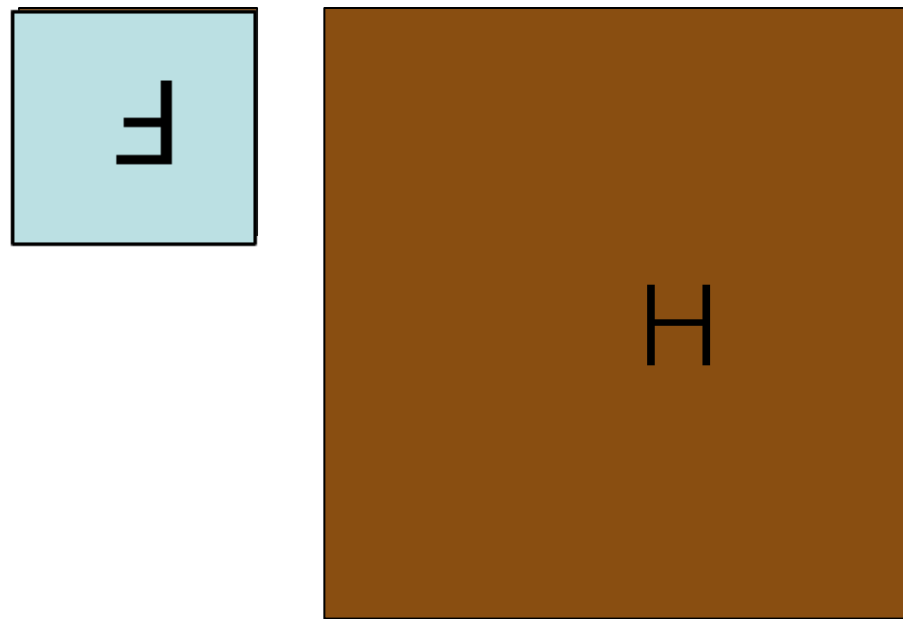
$$g = f * h$$



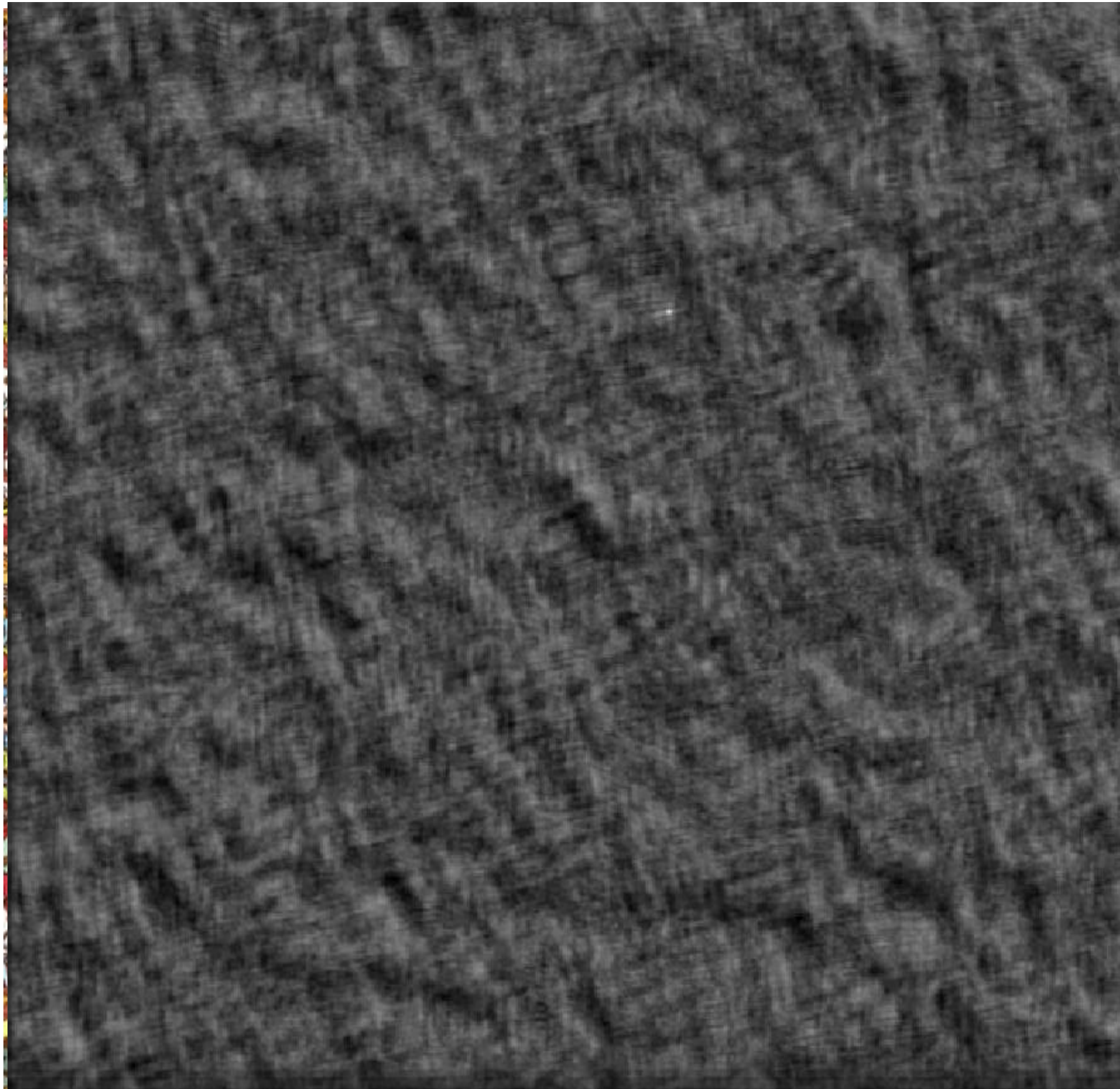
$$h = \delta * h$$

Convolution - repetition

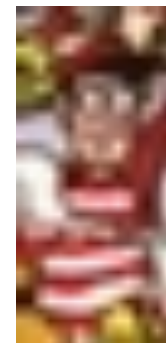
- Convolution:
 - Flip the filter in both dimensions (bottom to top, right to left)
 - Then apply cross-correlation
 - Produces scalar product of flipped filter at every position!



Where's Waldo?



Scene

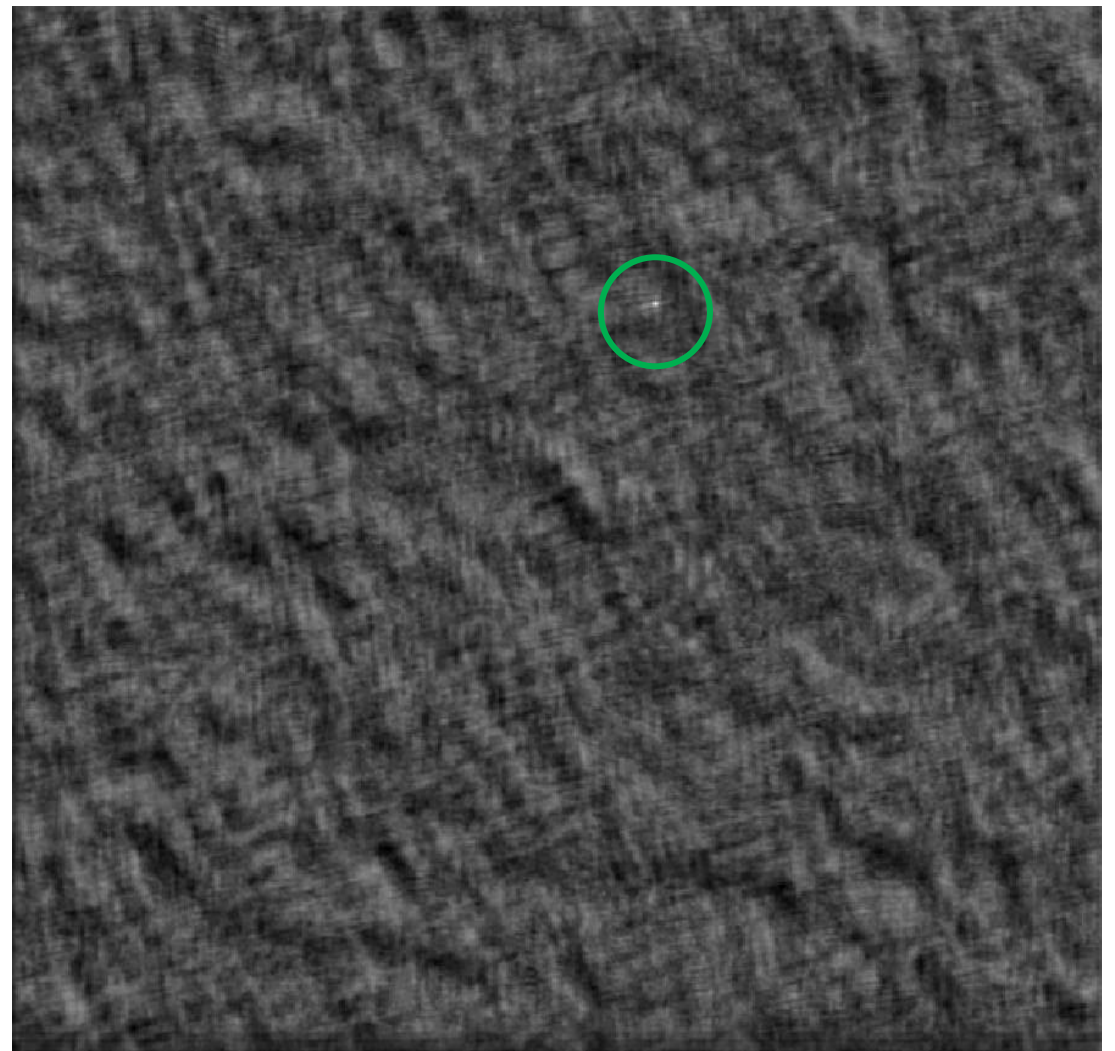


Template

Where's Waldo?



Detected
template



Correlation map

Clustering

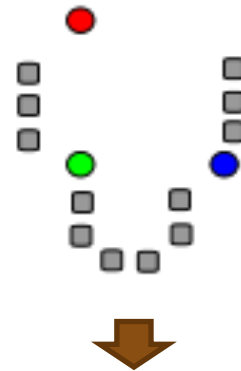
Clustering: group together similar points and represent them with a single token

Key Challenges:

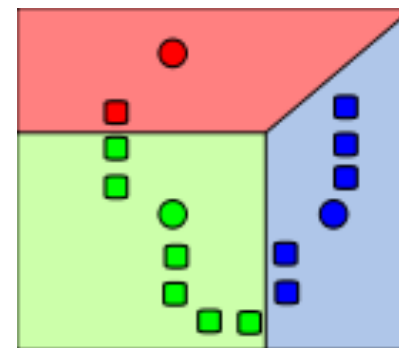
- 1) What makes two points/images/patches similar?
- 2) How do we compute an overall grouping from pairwise similarities?

K-means algorithm

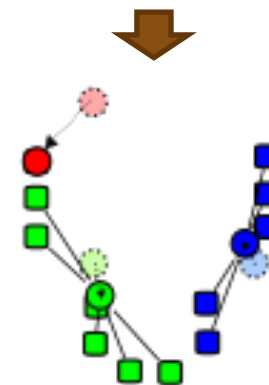
1. Randomly select K centers



2. Assign each point to nearest center

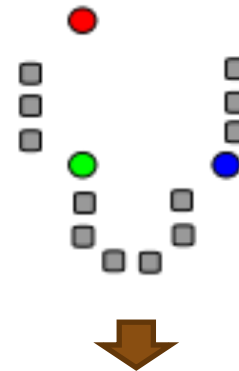


3. Compute new center (mean) for each cluster



K-means algorithm

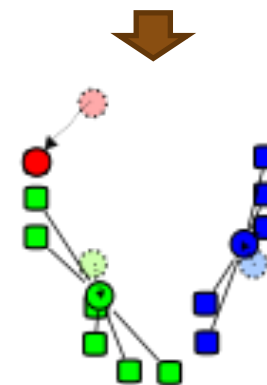
1. Randomly select K centers



2. Assign each point to nearest center



3. Compute new center (mean) for each cluster



Back to 2

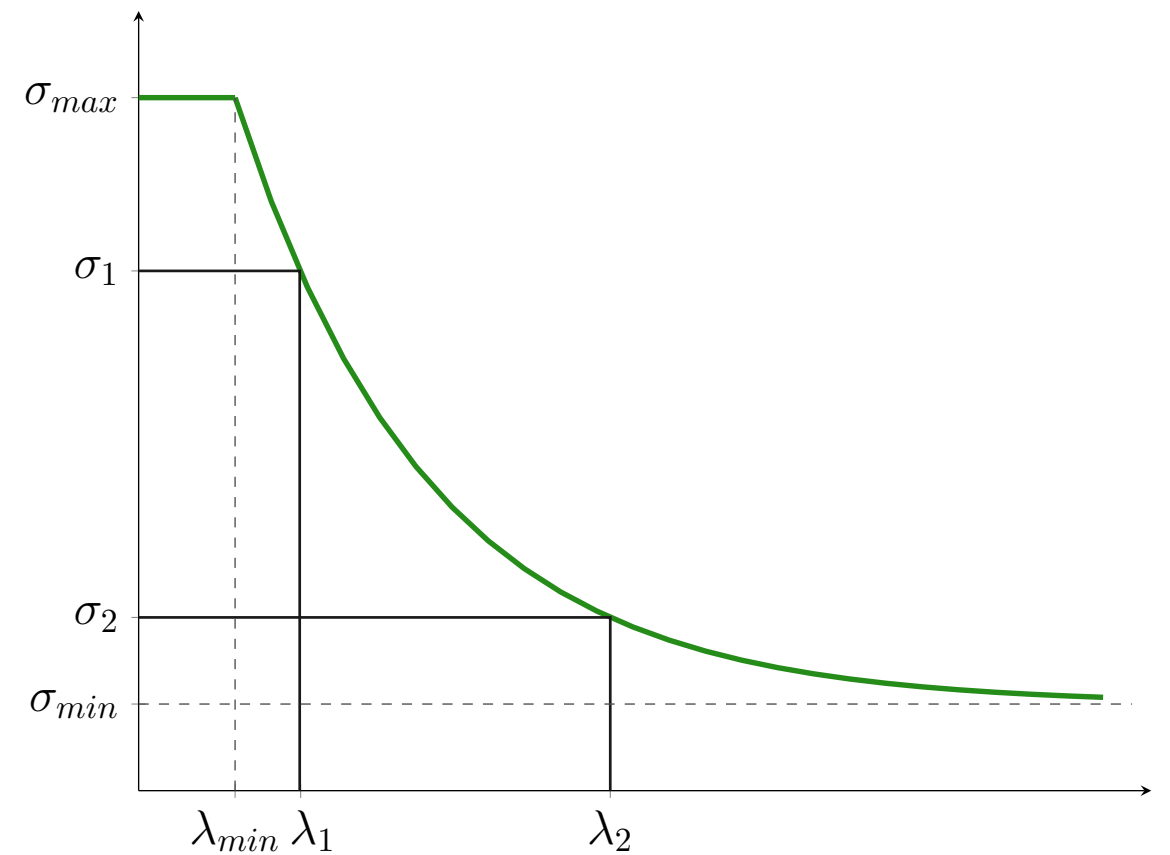
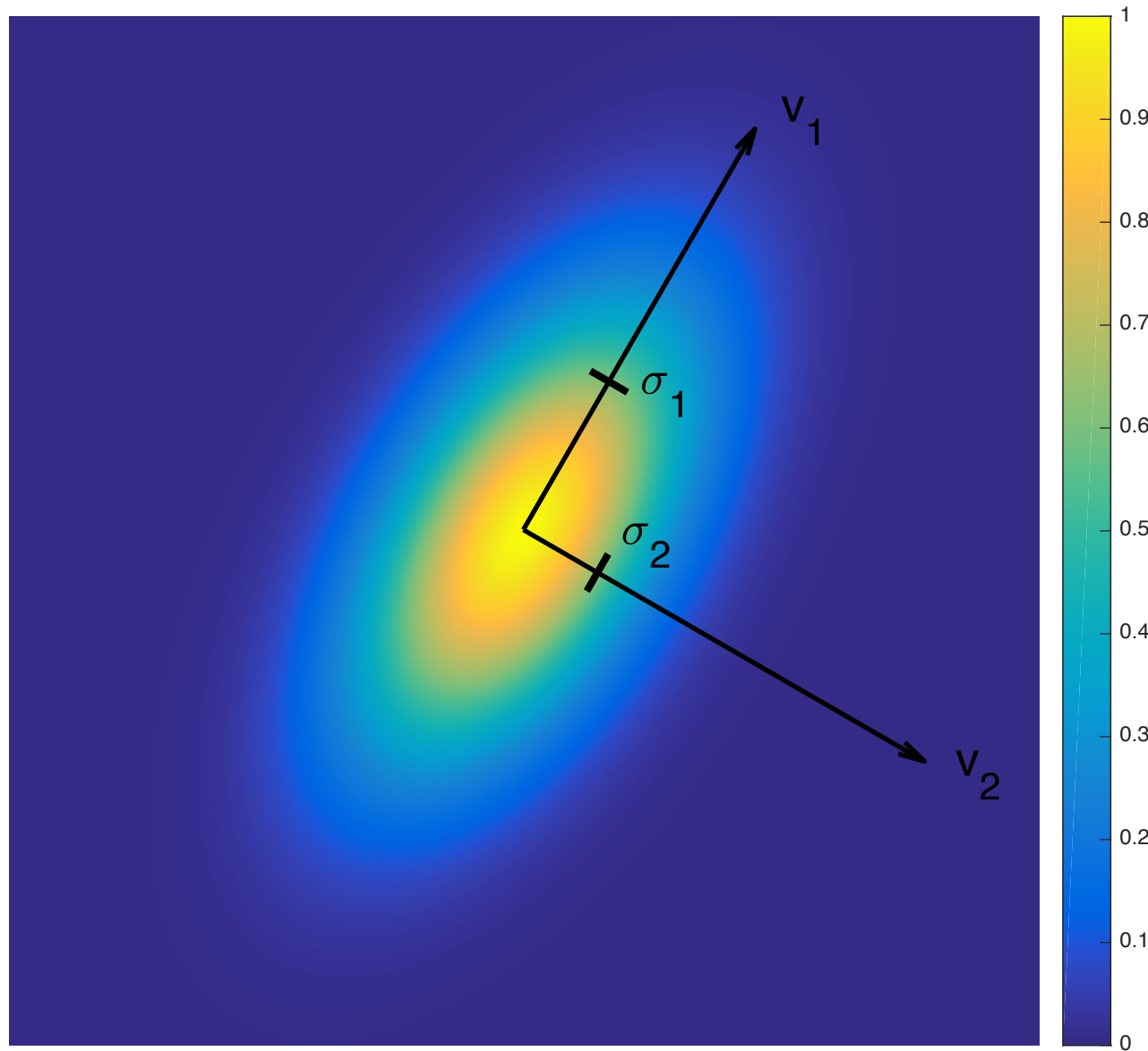
Structure Tensor

The matrix M has the following properties:

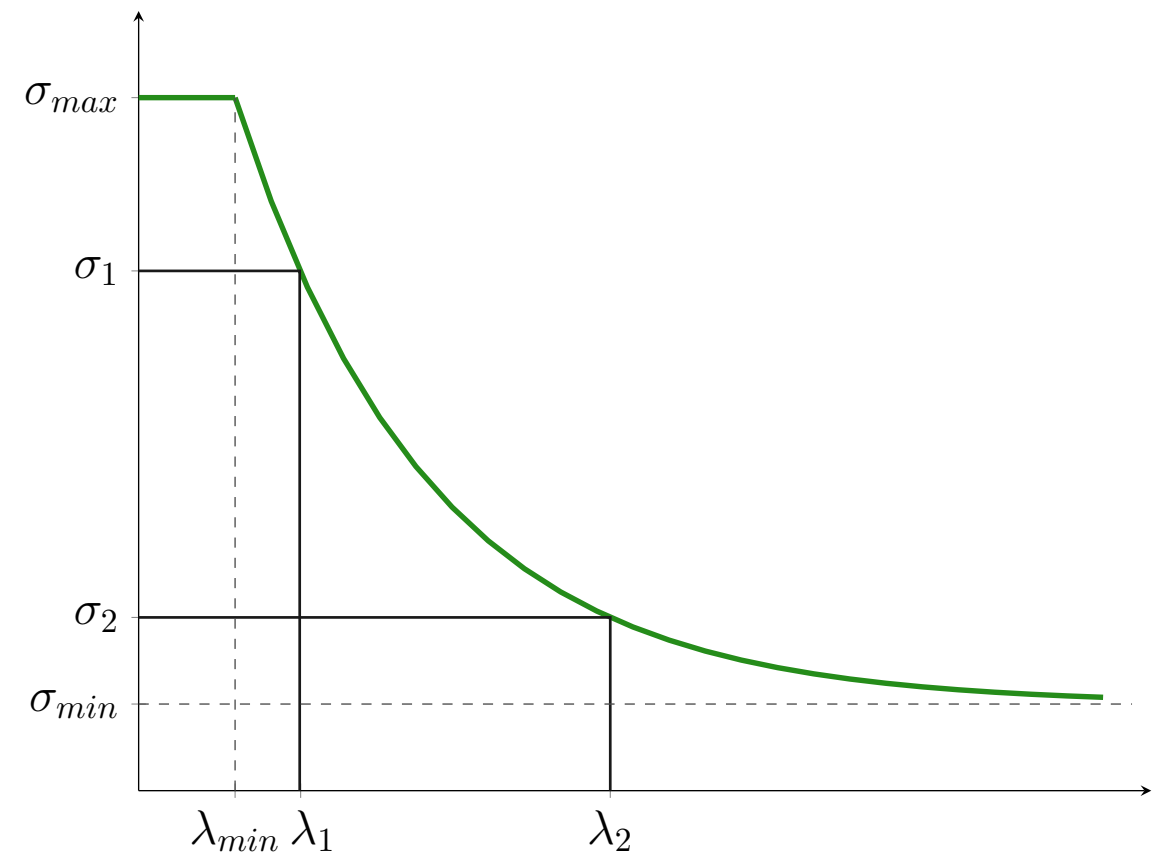
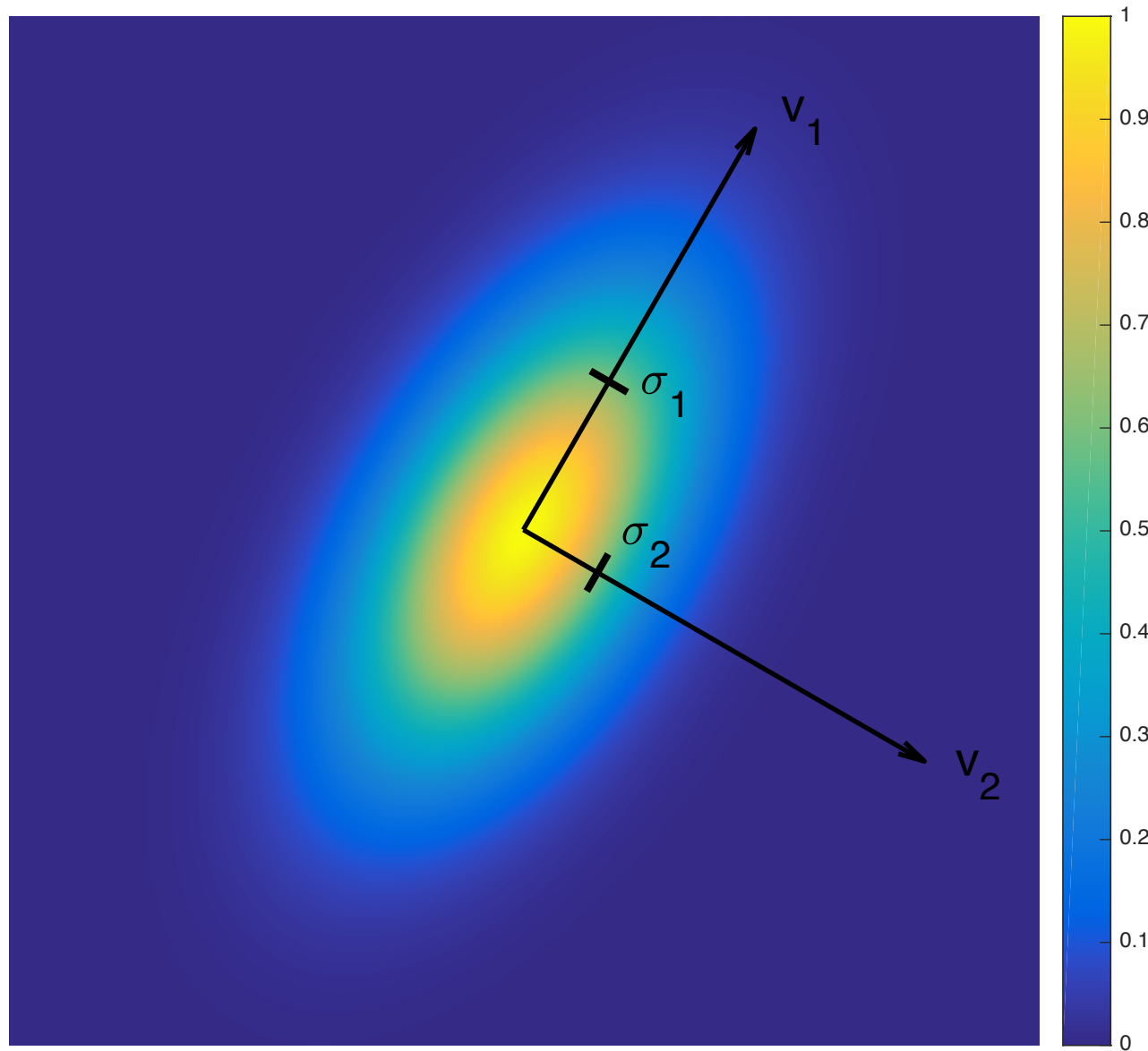
- ▶ (Flat) Two small eigenvalues in a region - flat intensity.
- ▶ (Flow) One large and one small eigenvalue - edges and flow regions.
- ▶ (Texture) Two large eigenvalues - corners, interest points, texture regions.

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

Rotate Gaussian kernels using eigenvectors of the structure tensor

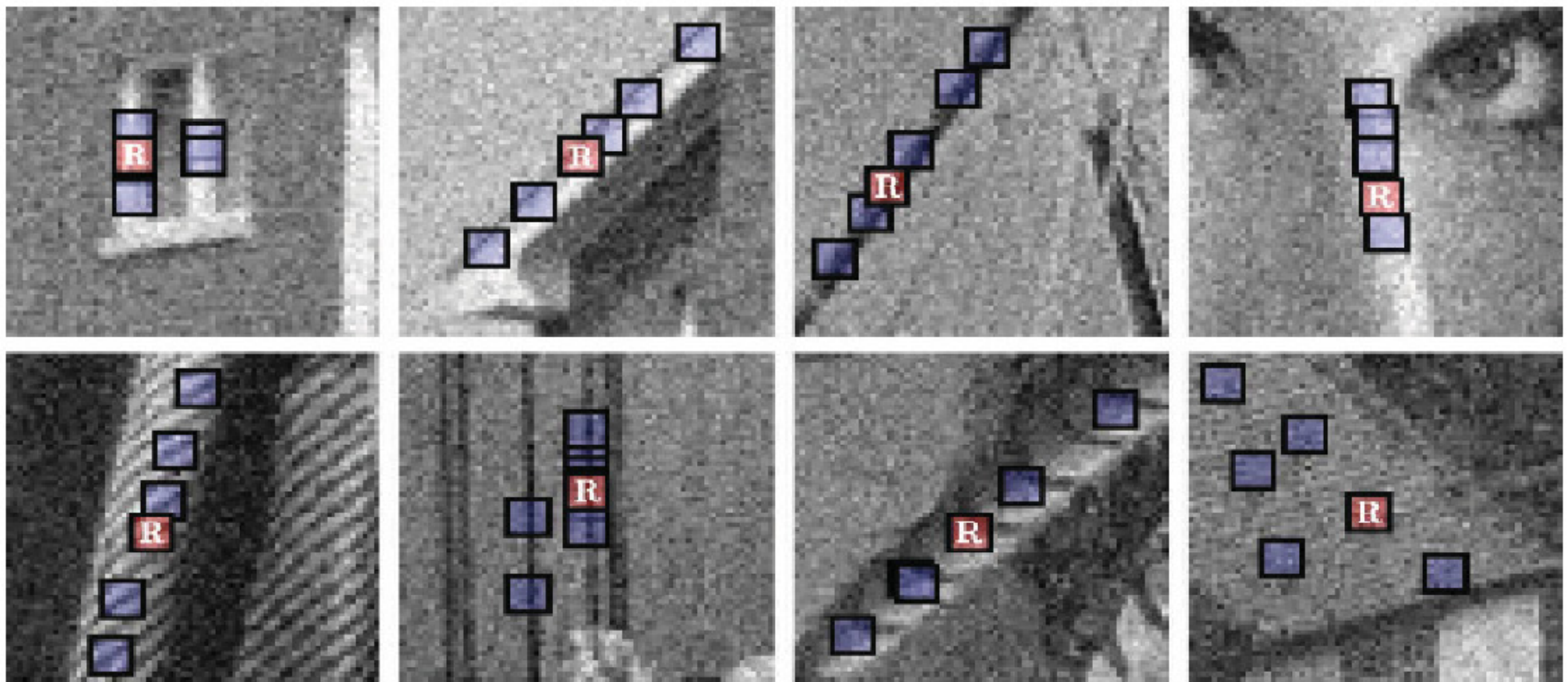


Scale Gaussian kernels using a function of the eigenvalues of the structure tensor



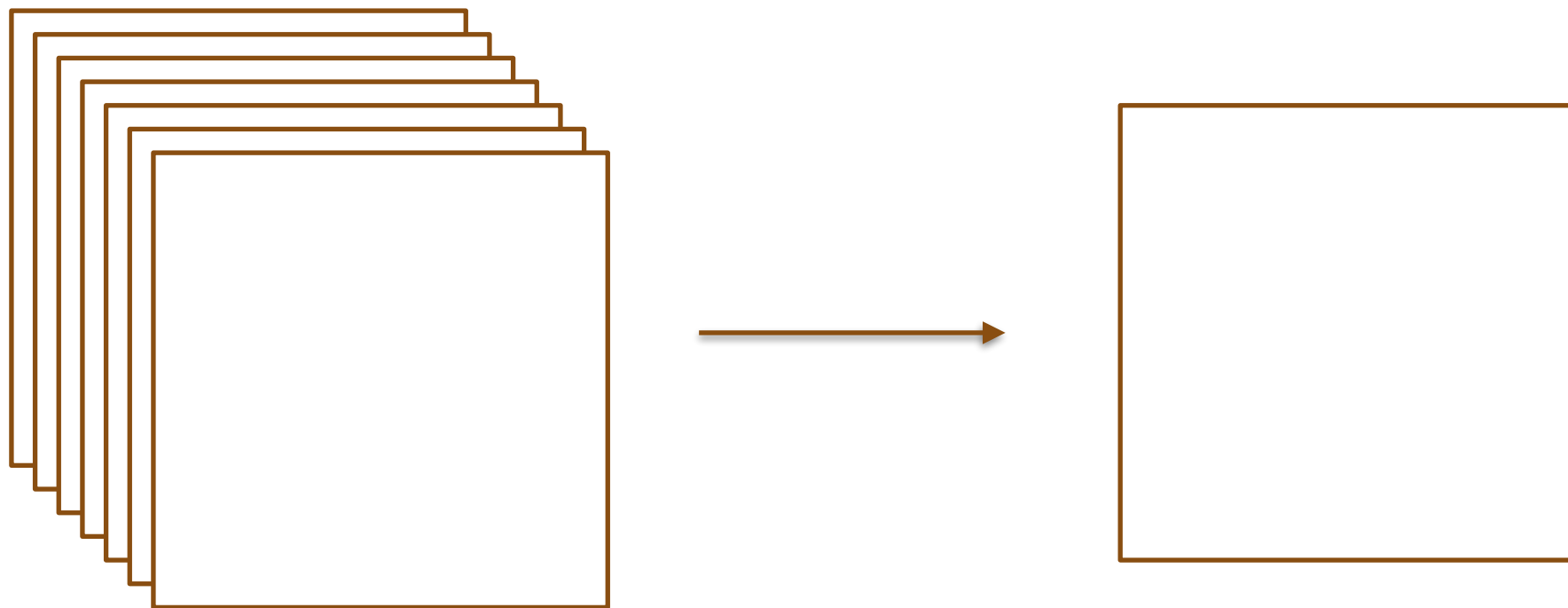
Block matching

many structures are repeated at multiple locations in an image



Collaborative filtering

Do hard filtering on each set of similar matched blocks



Example enhancement and filtering

Home Exam 2017, exercise 2: Enhance moon image



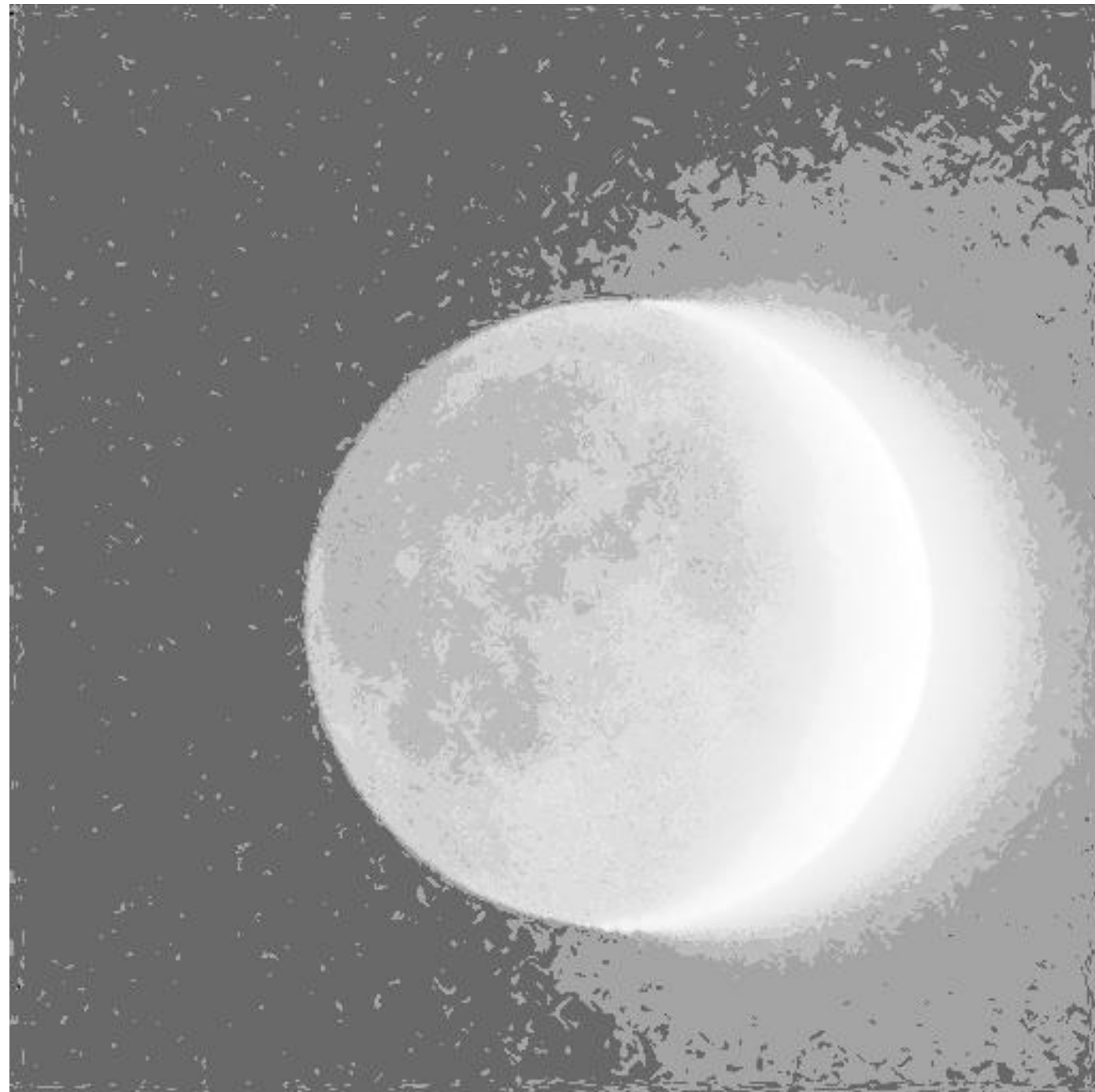
Amplify

Home Exam 2017, exercise 2: Enhance moon image



Anisotropic smoothing

Home Exam 2017, exercise 2: Enhance moon image



Visualize using color

Home Exam 2017, exercise 2: Enhance moon image





Segmentation



Pixels, clustering, segmentation

- At each pixel one could define a feature vector
 - Intensity $f(i,j)$
 - RGB colour channel (r,g,b)
 - Multispectral channel (Guest lecture)
 - Position (i,j)
 - Response from a filter bank
- Use machine learning to define a mapping from pixel feature vector to segment
- Either supervised (using lots of old examples) ...
- ... unsupervised (k-means, other clustering methods)

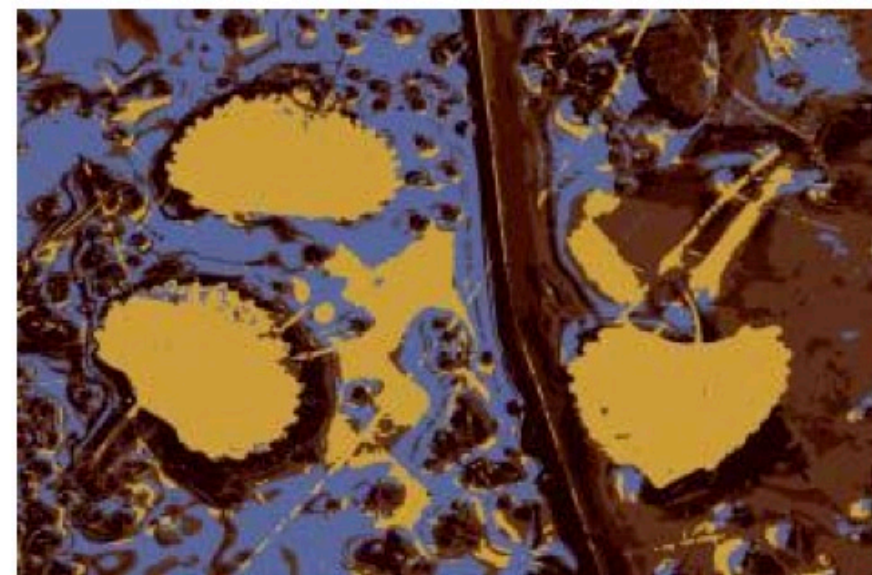
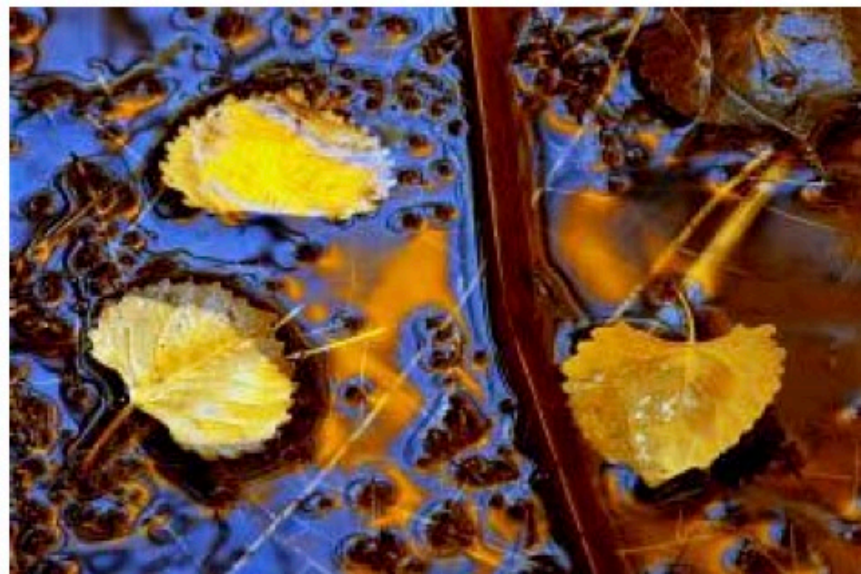
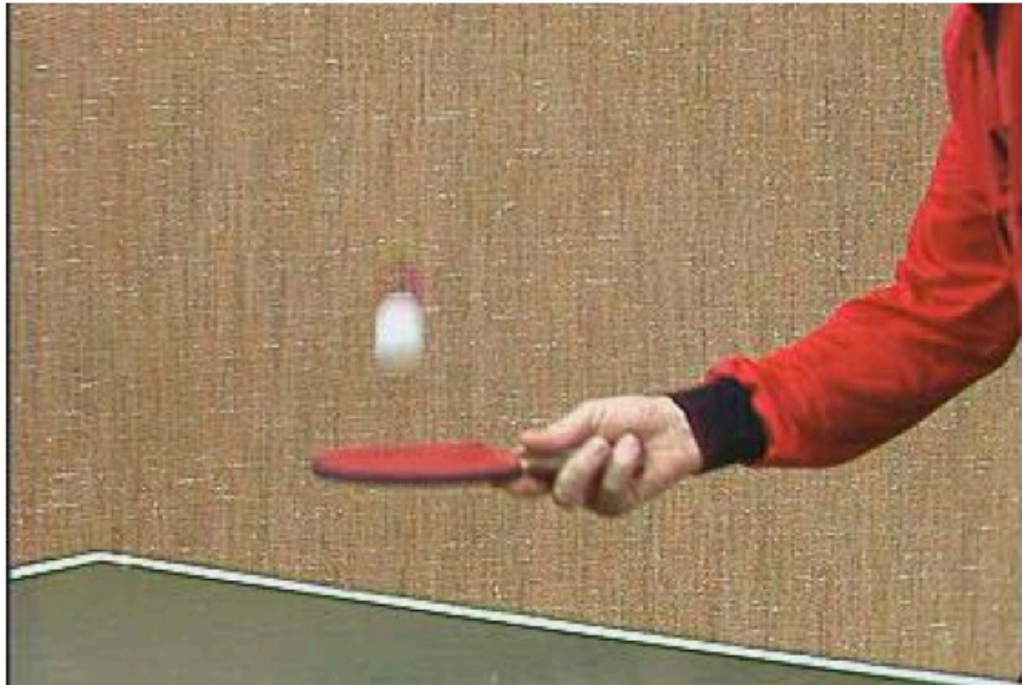
Mumford-Shah functional

- Let g differentiable on $\mathcal{U}R_i$ and allowed to be discontinuous across Γ .

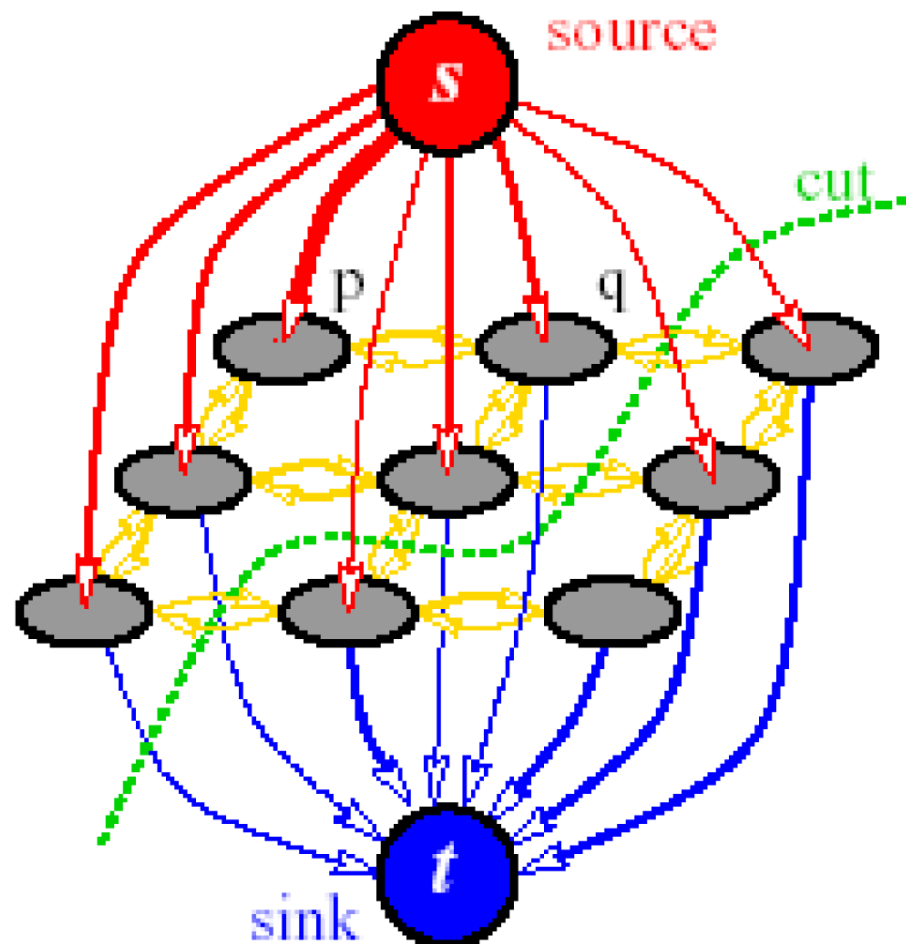
$$E(g, \Gamma) = \mu^2 \int_R (g - f)^2 dx dy + \int_{R-\Gamma} ||\nabla g||^2 dx dy + \nu |\Gamma|$$

- The smaller E , the better (g, Γ) segments f
 1. g approximates f
 2. g (hence f) does not vary much on R_i s
 3. The boundary Γ be as short as possible
- Dropping any term would cause $\inf E=0$.

Cartoon Image example



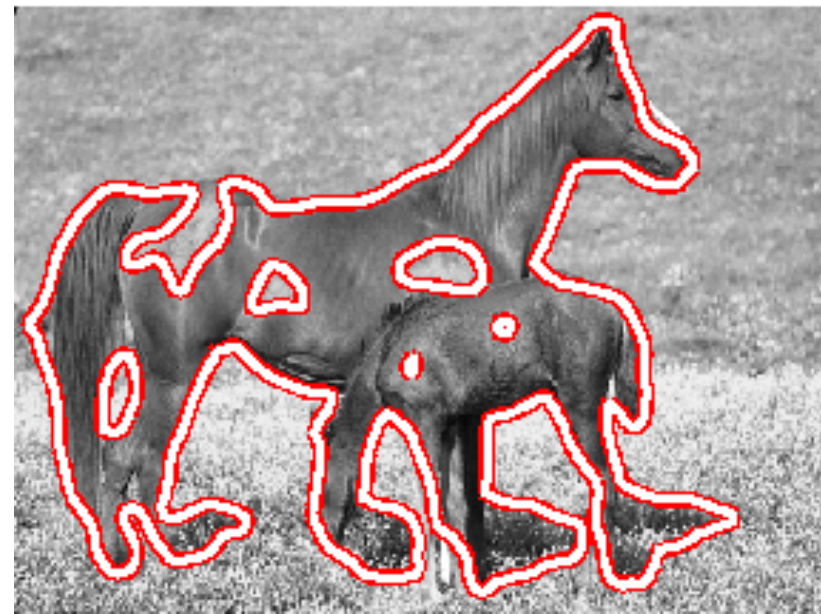
Graph Cuts



Definition: A *cut* (or *s-t cut*) in a graph $G=(V,E)$ is a subset of edges E_c such that there is no path from s to t when E_c is removed.

Definition: The *cost* of a cut is the sum of all edge weights for the edges in the cut.

Results of Two-Class Segmentation



P. Strandmark, F. Kahl, [Optimizing Parametric Total Variation Models](#),
International Conference on Computer Vision, Sep., Kyoto, Japan 2009.

Going to Object-class Segmentation Directly?

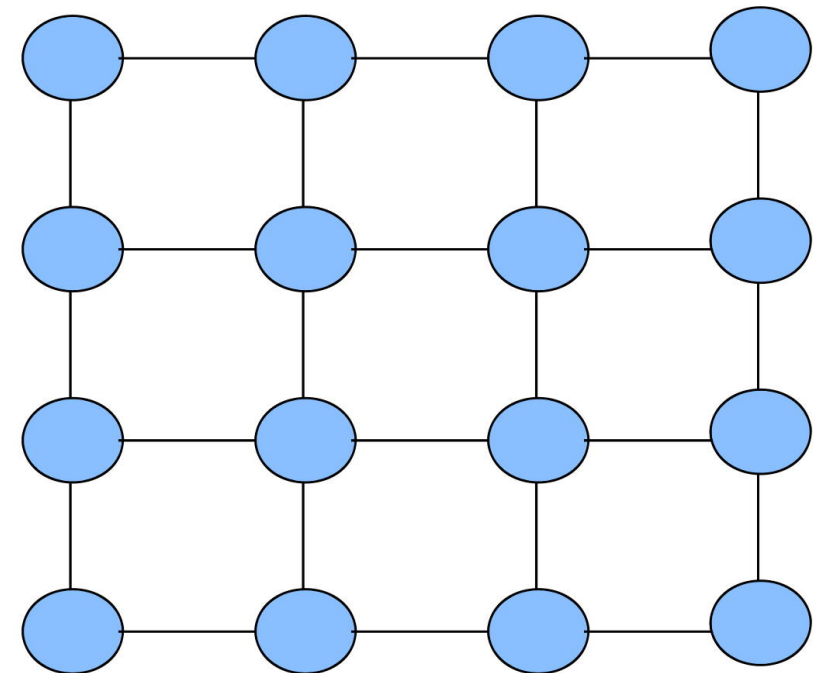
Slide by L. Ladický

Pairwise CRF over pixels



Input image

CRF
construction
→



↓
Training of
Potentials



Final segmentation

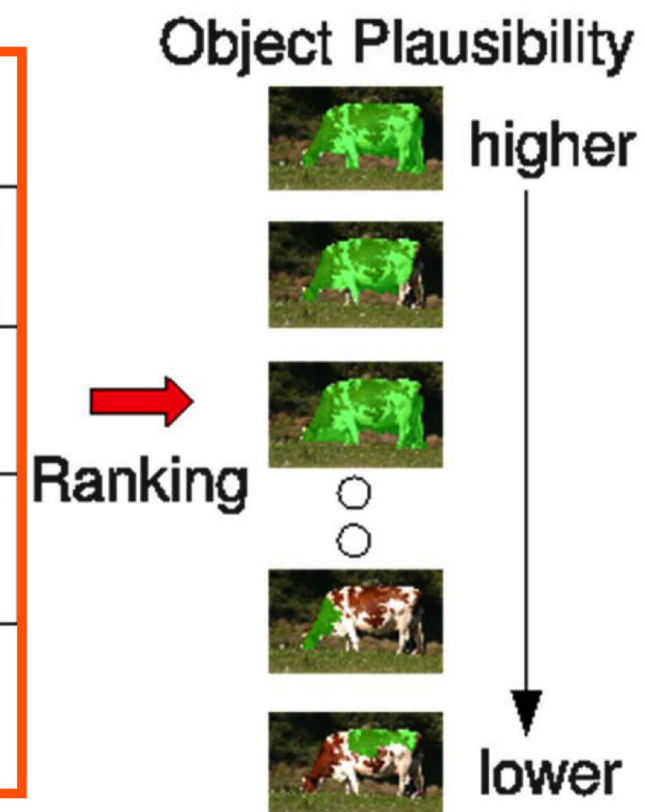
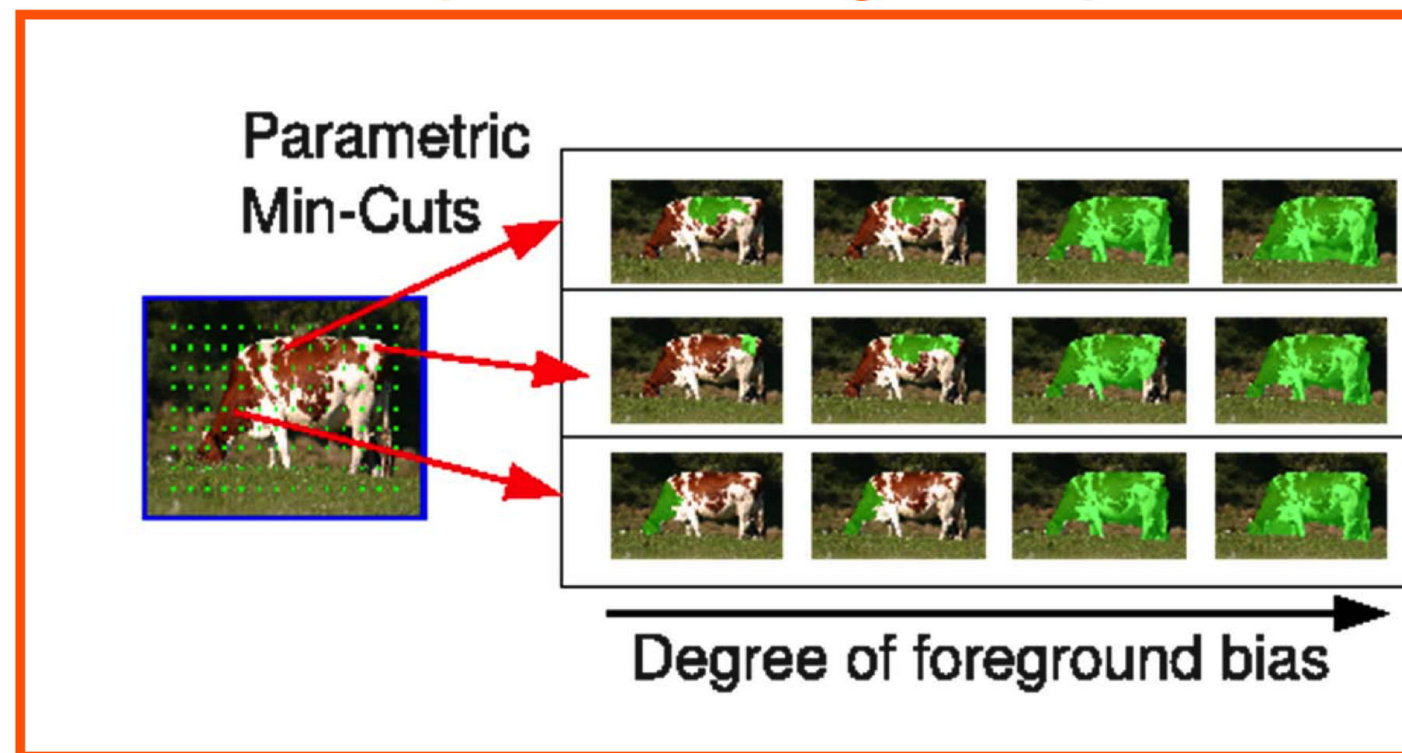
←
MAP

$$E(\mathbf{x}) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j)$$

Shotton et al. ECCV06

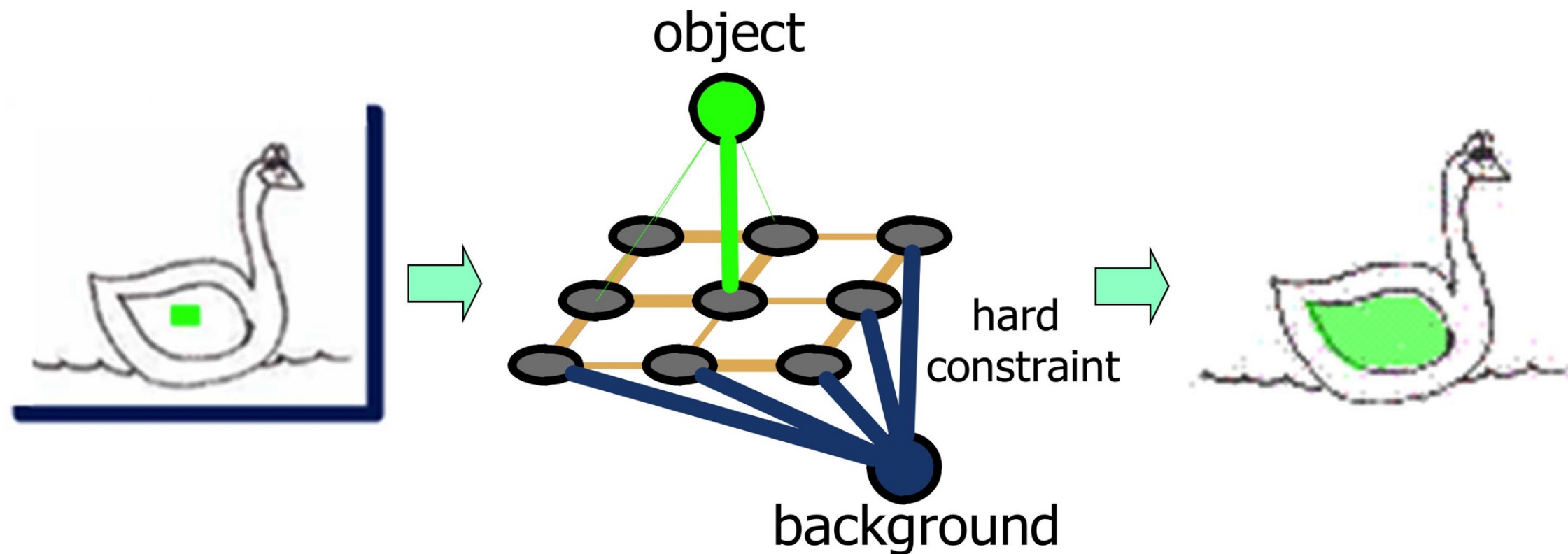
CPMC: Constrained Parametric Min-Cuts for Automatic Object Segmentation

First step: create segment pool



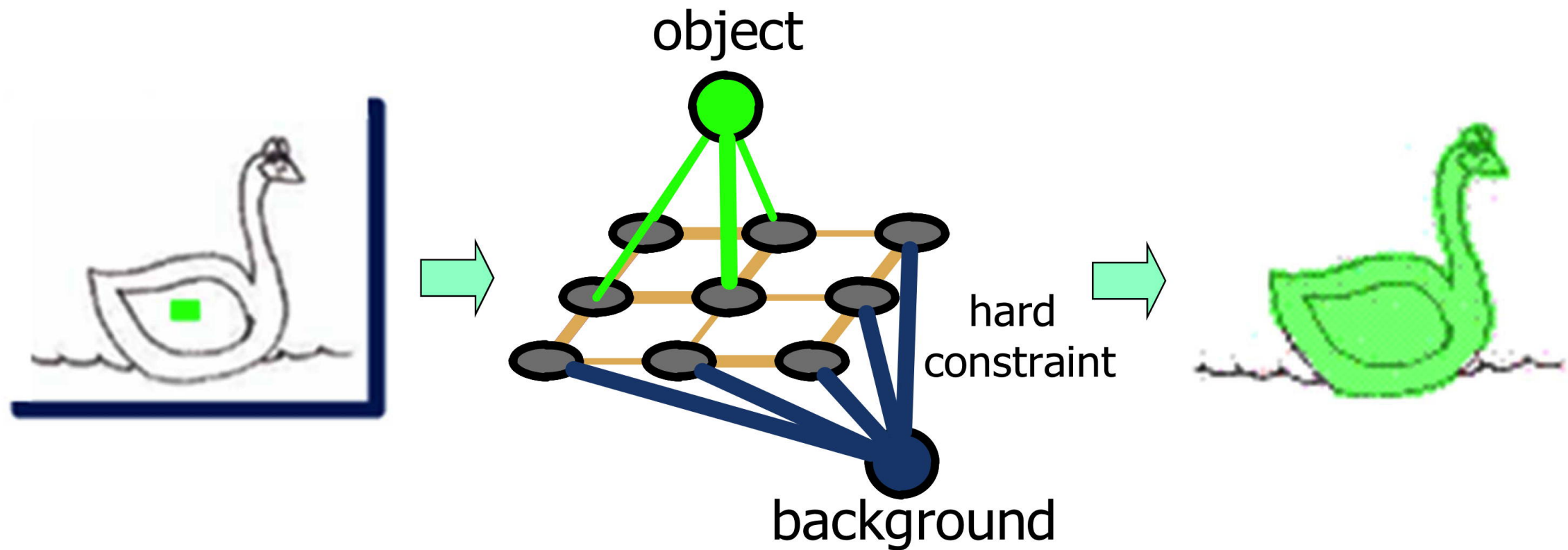
Generating a segment pool: constrained min-cut

$$E_{\lambda}(x) = \sum_{u \in V} D(x_u, \lambda) + \sum_{(u,v) \in E} V_{uv}(x_u, x_v)$$

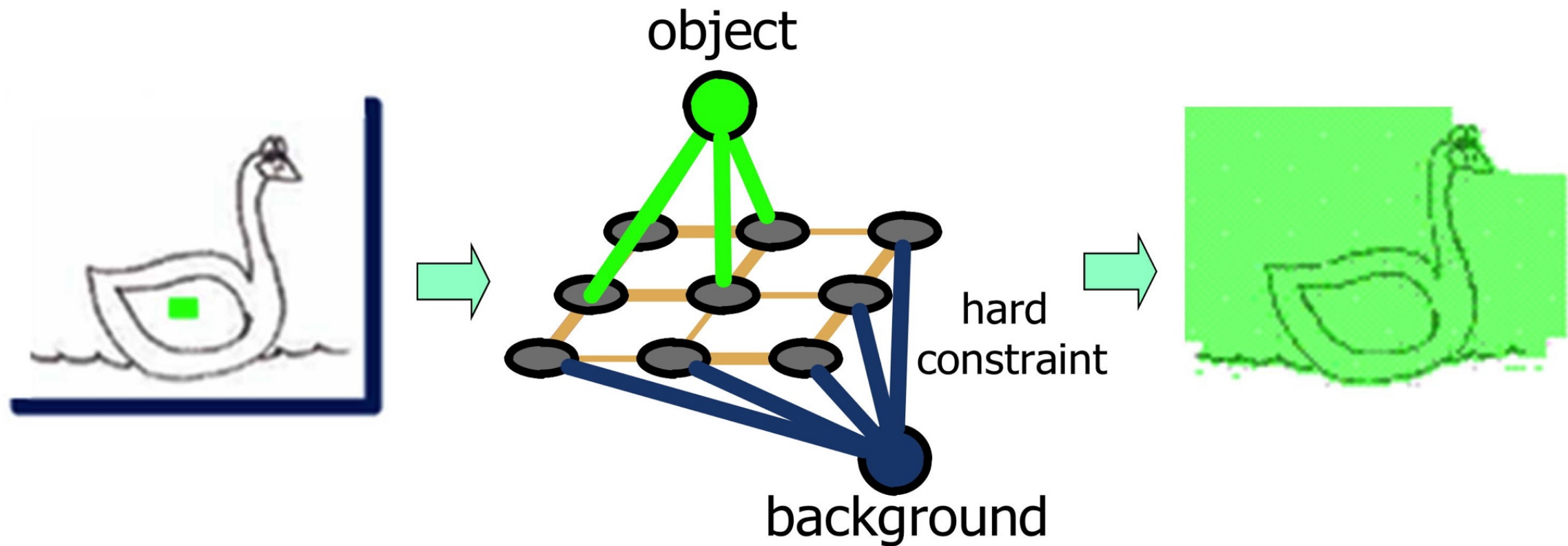


Generating a segment pool: constrained *parametric* min-cuts

$$E_{\lambda}(x) = \sum_{u \in V} D(x_u, \lambda) + \sum_{(u,v) \in E} V_{uv}(x_u, x_v)$$



Generating a segment pool: constrained *parametric* min-cuts





LUND
UNIVERSITY

