INF 5520 – Digital Image Analysis

TEXTURE

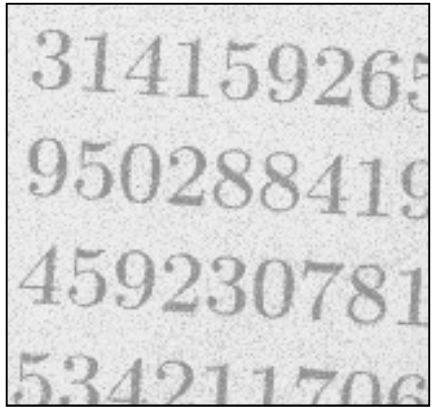
Fritz Albregtsen 26.08.2020

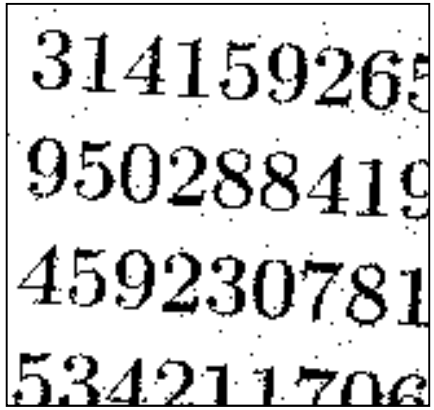
Plan for today

- G&W, "DIP4E", Ch. 11 (pages 846-857)
- Why texture, and what is it?
- Statistical descriptors
  - First order
    - Mean, variance, …, moments, …
  - Second order
    - Gray level co-occurrence matrices (GLCM)
  - Higher order
    - Fourier analysis
    - Gray level runlength matrices
    - Cooccurrence of gray level runs

What is segmentation?

- A process that splits the image into meaningful regions.
- One of the most important elements of a complete image analysis system.
- Segmentation gives us regions and objects that we may later describe and recognize.
- The simplest case: two classes:
  - Foreground
  - Background





”Simple” example:
find symbols for OCR

Segmentation problems

- Usually several objects in an image.
- Often several classes.
- Objects of same class
  are seldom completely alike.
- Reflection, color etc. may vary.
- Lighting may vary over image.

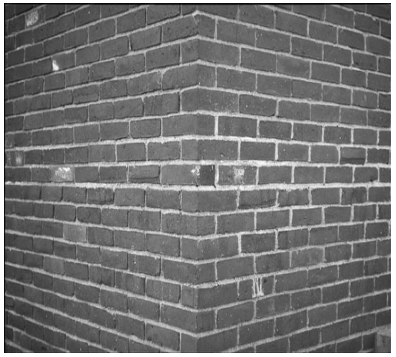
- We perceive and utilize
  - intensity (jnd≈1.7%=>N ≈ 60)
    & color, N(IHS)≈(60*100*60 =360 000)
    + **texture**.

A more complex example:
What and where is the object in this image?

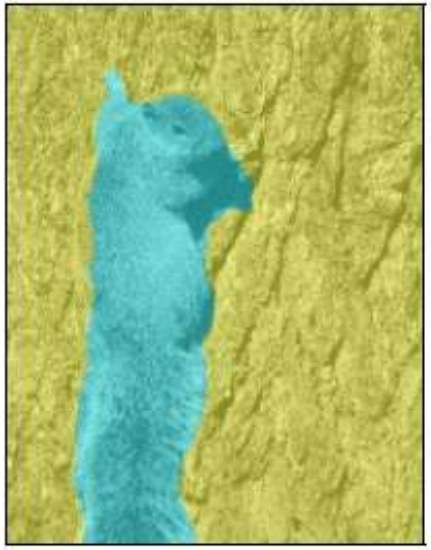
What is texture?

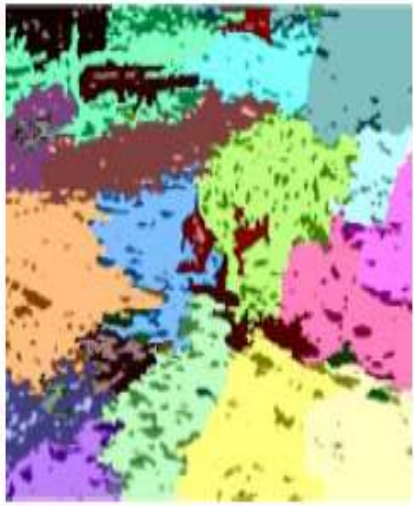
- Intuitively obvious,
  but no precise definition exists
  - ”fine, coarse, grained, smooth” etc
- Texture consists of texture primitives, texels,
  - a contiguous set of pixels with some tonal and/or regional property
- Texture can be characterized by
  - intensity (color) properties of texels
  - Structure & spatial relationships of texels
- A texel is the characteristic object that the texture consists of (the ”brick in the wall”)
- Textures are highly scale dependent.

“Spatially extended patterns of more or less accurate repetitions of some basic texture element, called texels.”



How do we segment these images?





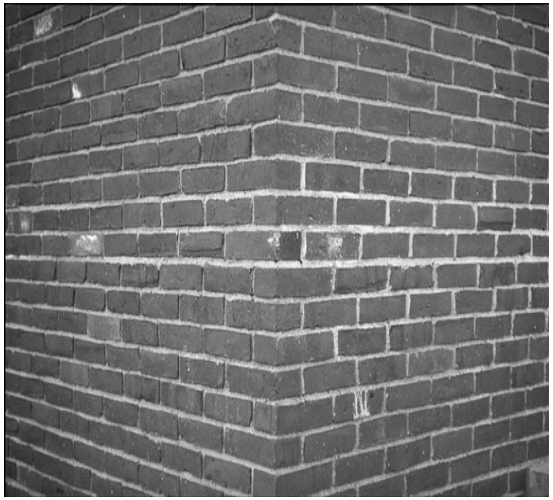
What is a texel?

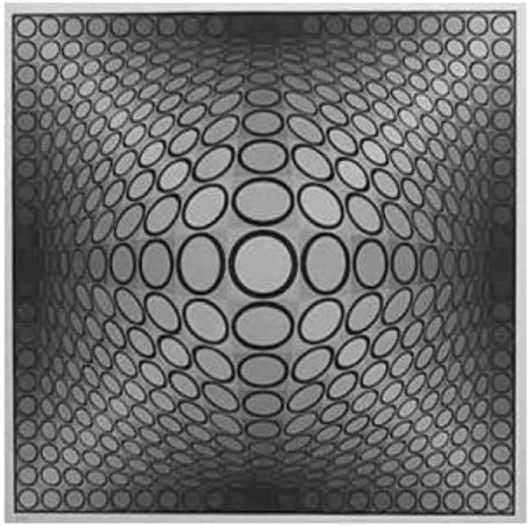
Texel = texture element, the fundamental unit of texture space.
Can be defined in a strict geometrical sense, or statistically.

Note that you can define texels in any image scale,
and that the best image scale for analysis is problem dependent.

Texture analysis applications

- Segment an image into regions with the same texture, i.e. as a complement to graylevel or color
- Recognize or classify objects in images based on their texture
- Find textural edges in an image, i.e., where the texture changes
- ”shape from texture” (Victor Vasarely) --->
- object detection, compression, synthesis
- Industrial inspection:
  - find defects in materials

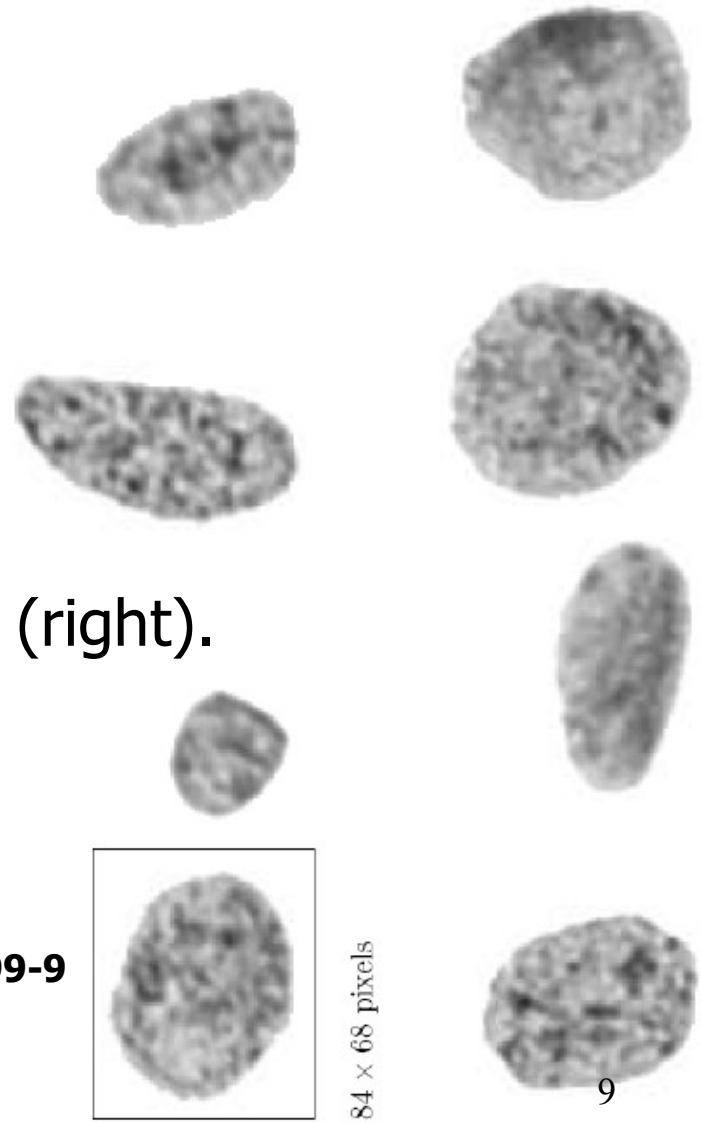




Texture analysis applications - II

- Reliable cancer prognostics.
- Microscopy images of monolayer cell nuclei from early stage ovarian cancer.
- Four monolayer cell nuclei from a good prognosis sample (left) and from a bad prognosis sample (right).
- Aim: small set of differentiating textural features.
  - See The Lancet Oncology (2018):

http://dx.doi.org/10.1016/S1470-2045(17)30899-9

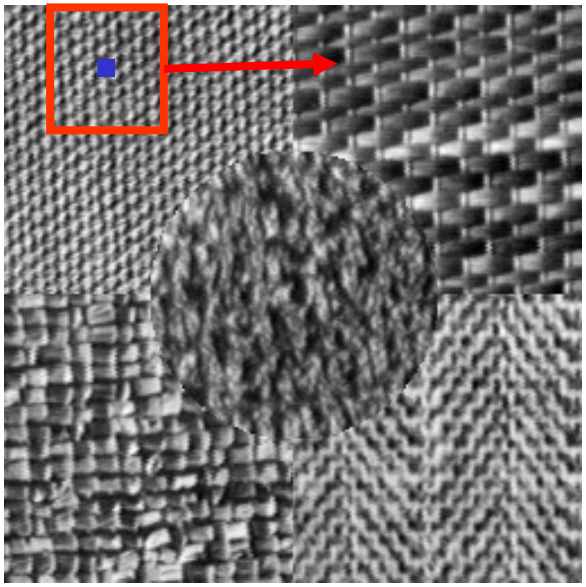


Features

- Gray level image features can be found from:
  - Edges
    - Gives the (incomplete ?) borders between image regions
  - Homogeneous regions
    - Mean and variance are useful for describing the contents of homogeneous regions
  - The texture of a local (sliding) window
    - Features that describe how the gray levels in a window varies, e.g. roughness, regularity, smoothness, contrast etc.

A simple approach to texture

- To be able to find *changes* in the texture of an image, a simple strategy is to perform texture measurements in a sliding window
- Most texture features can be summed up as a set of scalars, so we can assign features to each of the image pixels corresponding to window centers
- For each pixel, we now have a description of the ”texture” in its neighborhood
- Beware of image boundaries, artifacts will occur!

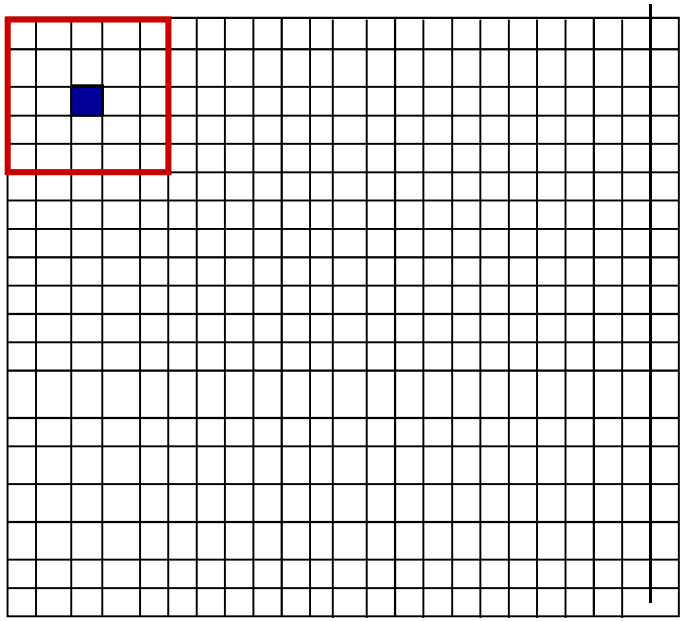


Compute a local texture feature in a local window.
Slide the window around in the image.
Each computed texture feature gives a new texture feature image!

Computing texture images

- Select a window size
  and select a texture feature

- For each pixel (i,j) in the image:
  - Center the window at pixel (i,j)
  - Compute the texture feature
  - Assign the computed value
    to the center pixel (i,j)
    in a new output image
    of the same size



- This is similar to filtering

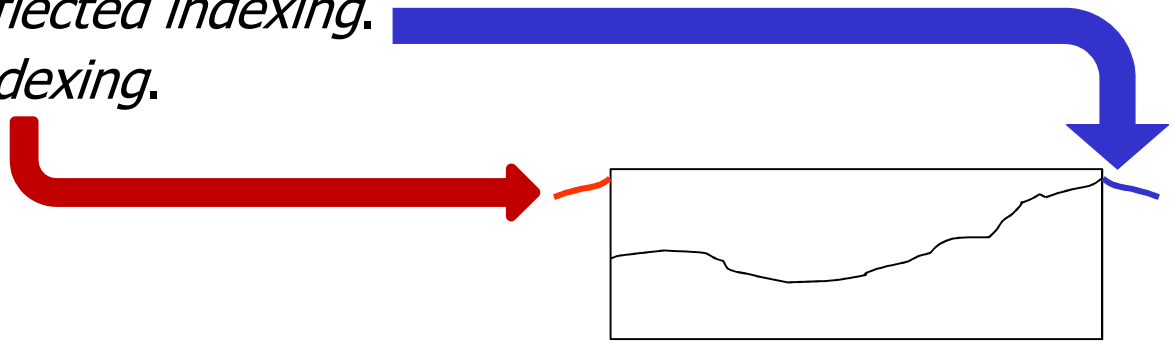
- Pixels close to the image border can be handled in the same manner as for filtering/convolution (reflected indexing, circular indexing)

What to do along image border?

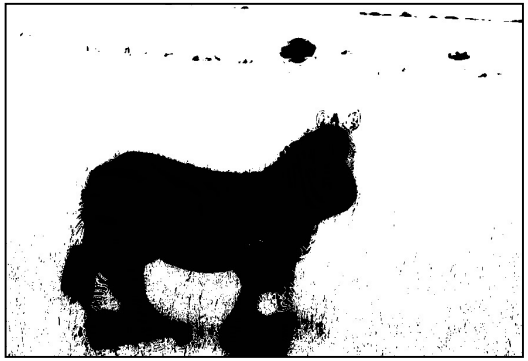
- Make larger in-image, to make filter fit inside in-image, to produce an out-image of same size as original.

- Padding of in-image:
  - Most common: *zero padding*.
  - Using some fixed value, e.g., image mean value
  - Using closest image pixel value (*replicate*).
  - Using *mirror-reflected indexing*.
  - Using *circular indexing*.

Do not conserve texture



Texture feature image example



Input image.

For each pixel, compute a local homogeneity measure in a local sliding window.

New homogeneity image.

Try to get an image where pixels belonging to the same texture type get similar values.

Segmented feature image.

”Texture” – description of regions

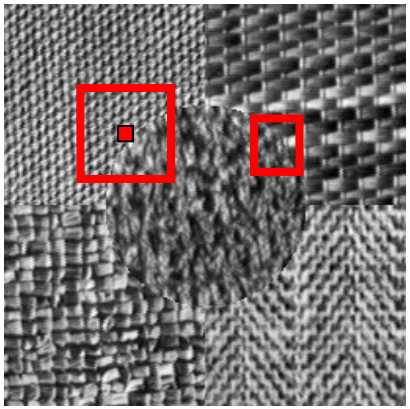
- Remember: we estimate local properties (features) to be able to isolate regions which are similar in an image (segmentation), usually with the goal of object description and possibly later identify these regions (classification),

- One can describe the ”texture” of a region by:
  - smoothness, roughness, regularity, orientation...

- Problem: we want the local properties to be as ”local” as possible

- Large region or window
  => Precise estimate of features
  => Imprecise estimate of location

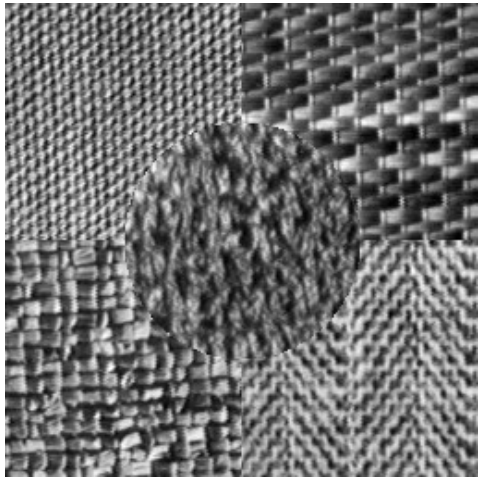
- Small window
  => Precise estimate of location
  => Imprecise estimate of feature values



Uncertainty relation

- Large region or window
  => More precise feature value,
  but imprecise boundaries between regions

- Small window
  =>More precise estimate of region boundaries,
  but imprecise feature value



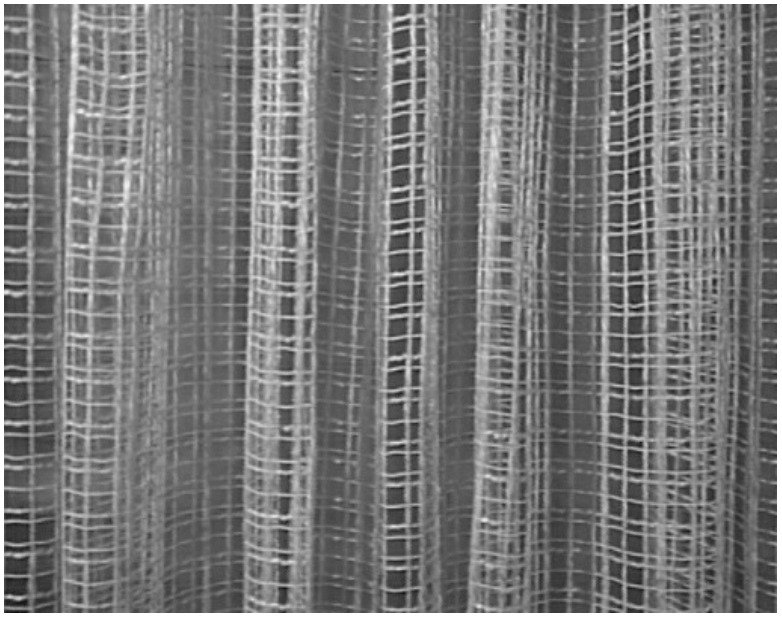
- Related to Heisenberg's uncertainty relation in physics:

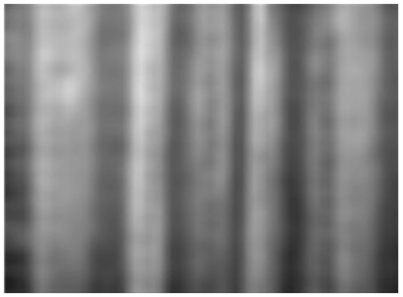
$$\sigma_x \sigma_p \approx h/(2\pi)$$

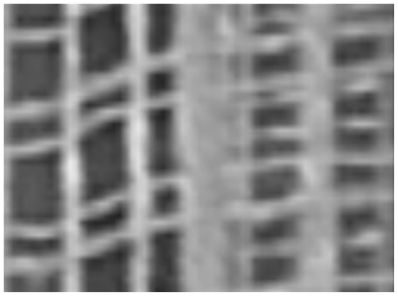
See: R. Wilson and G.H. Granlund,
"The Uncertainty Principle in Image Processing,
IEEE T-PAMI-6 , pp.758-767, 1984. DOI: 10.1109/TPAMI.1984.4767599

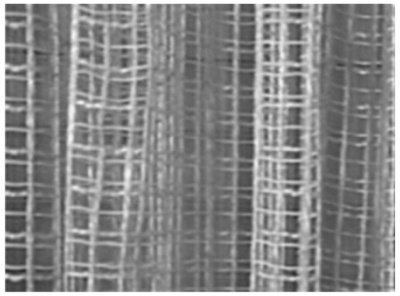
Texture description is scale dependent

- What is our goal for texture description in the image?
- Scale impacts the choice of texels, and vice versa
- The curtain can be described as
  - a repetition of single threads,
  - a configuration of meshes
  - a repetition of folds.

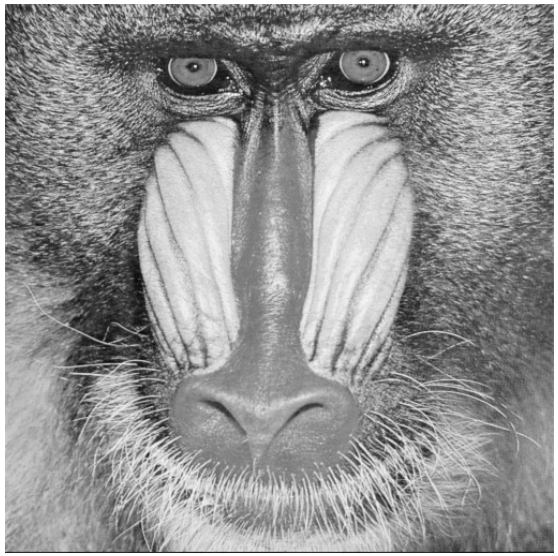








Example of scale dependence



Original image

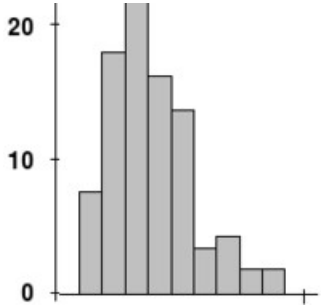
“Variance” feature computed in window of size 3x3

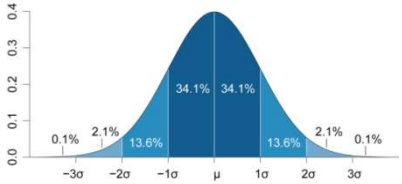
“Variance” feature computed in window of size 15x15

Statistical texture description

- Describe texture in a region by a vector of statistics (feature vector)
  - First order statistics from graylevel intensity histogram ***p(i)***
    - Mean, variance, 3. and 4. order moment, entropy, energy
  - Second order statistics,
    describing relation between pixel pairs
    - How does the gray levels of pixels ***i*** and ***j*** at a distance ***d*** depend on each other. Are they similar or different ?
  - Higher order statistics,
    - describe region by ***runs*** of similar pixels

First order statistics from histogram - I





- Mean (hardly a useful texture feature)

$$\mu = \frac{\sum_{i=0}^{G-1} ip(i)}{\sum_{i=0}^{G-1} p(i)} = \frac{\sum_{i=0}^{G-1} ip(i)}{n} = \sum_{i=0}^{G-1} iP(i)$$

- Variance (a more credible feature, measures region "roughness")

$$\sigma^2 = \sum_{i=0}^{G-1} (i-\mu)^2 P(i)$$

- Skewness (are the texel intensities usually darker/lighter than average?)

$$\gamma_3 = \frac{1}{\sigma^3} \sum_{i=0}^{G-1} (i-\mu)^3 P(i) = \frac{m_3}{\sigma^3}$$

- Kurtosis (how ”peaked" is the graylevel distribution?)

$$\gamma_4 = \frac{1}{\sigma^4} \sum_{i=0}^{G-1} (i-\mu)^4 P(i) - 3 = \frac{m_4}{\sigma^4} - 3$$

First order statistics from histogram - II

- Entropy, i.e., first order Shannon entropy
  (how uniform is the graylevel distribution?)

$$H = -\sum_{i=0}^{G-1} P(i) log_2 P(i)$$

- Energy
  (how non-uniform is the graylevel distribution?)

$$E = \sum_{i=0}^{G-1} [P(i)]^2$$

Using variance estimates

- Variance, σ^2, is directly a measure of ”roughness”
  - An unbounded measure ($\sigma^2 \geq 0$)

- A measure of ”smoothness” is

$$R = 1 - \frac{1}{1+\sigma^2}$$

  - A bounded measure ($0 \leq R \leq 1$)
    - R is close to 0 for homogenous areas
    - R tends to 1 as σ^2, ”roughness”, increases

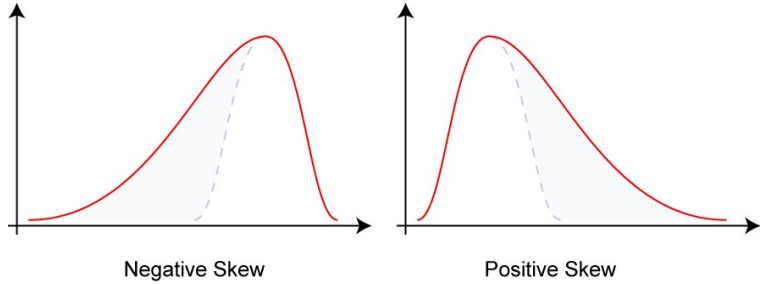
Using variance estimates

- "Coefficient of variation"

$$cv = \frac{\sigma_w(x,y)}{\mu_w(x,y)}$$

  - where σ_w and μ_w are computed within window w x w
  - *CV* is intensity scale invariant: $i' = Ai$
  - but not intensity shift invariant: $i' = i + B$
  - Alternatives (if assumption of normal distribution not valid):
    - use median instead of mean
    - interpercentile-distance instead of standard deviation
    - Also note "variance-to-mean" and "signal-to-noise"

Skewness



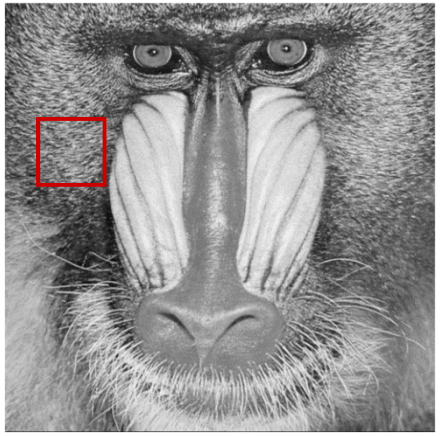
$$\gamma_3 = \frac{1}{\sigma^3}\sum_{i=0}^{G-1}(i-\mu)^3 P(i) = \frac{m_3}{\sigma^3}$$

- **Skewness** is a measure of the asymmetry of the probability distribution
- Measures if there is a "wider" range of either darker or lighter pixels

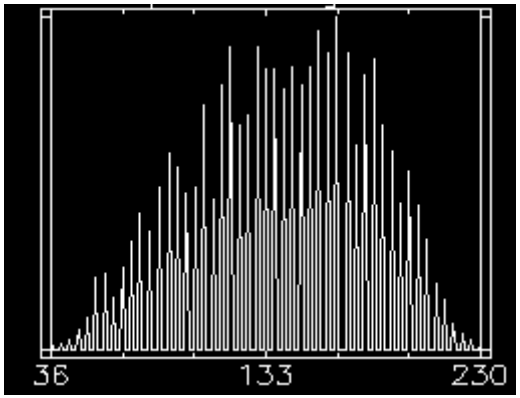
- **Negative skew**: The left tail is longer; the mass of the distribution is concentrated on the right of the figure.

- **Positive skew**: The right tail is longer; the *mass* of the distribution is concentrated on the left of the figure (more darker pixels than average).

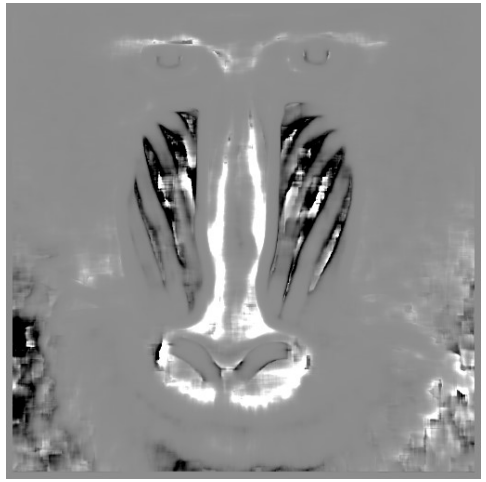
Skewness example



Region 1



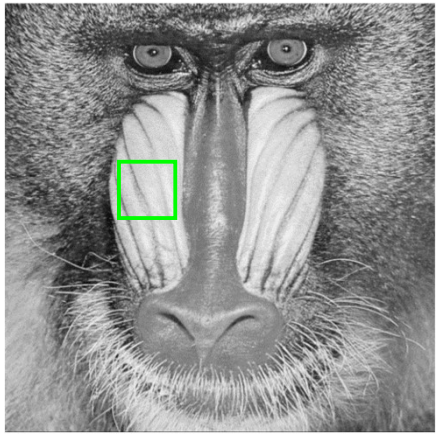
Histogram - Region 1



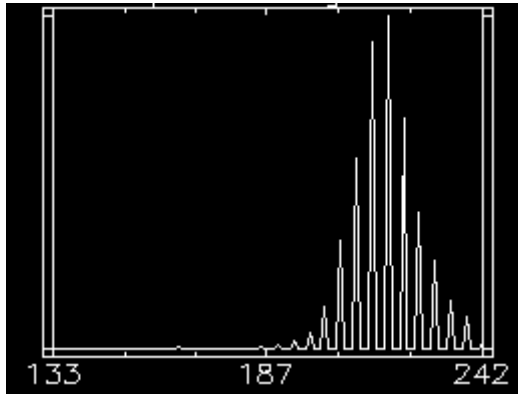
Skewness feature
Computed in 15x15 window

Region1: all gray levels occur
Histogram is fairly symmetric
Skewness is gray (average)

Region2: bright pixels more frequent. Histogram asymmetric.
Negative skew: Skewness is dark.

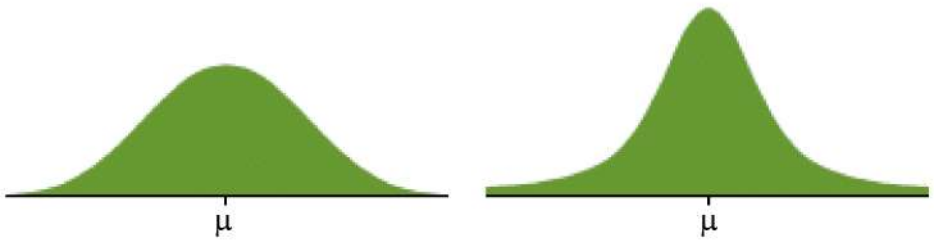


Region 2



Histogram - Region 2

Kurtosis



$$\gamma_4 = \frac{1}{\sigma^4}\sum_{i=0}^{G-1}(i-\mu)^4 P(i) - 3 = \frac{m_4}{\sigma^4} - 3$$

- Measure of "peakedness" of the probability distribution.
- Low kurtosis distribution has a more rounded peak with wider "shoulders"
- A high kurtosis distribution has a sharper "peak" and flatter "tails"

First order statistics - Entropy

- Entropy (how uniform is the graylevel distribution?)

$$H = -\sum_{i=0}^{G-1} P(i) log_2 P(i)$$

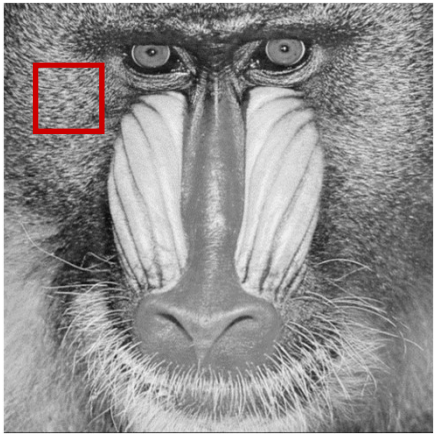
- If all pixel values are the same, H = 0 (since $\log_2(1) = 0$).
- If all pixel values are equally probable:
  - There are $G = 2^b$ gray levels, each having a probability $p(i) = 1/G = 1/2^b$, so:

$$H = -\sum_{i=0}^{2^b-1} \frac{1}{2^b} \log_2 \left(\frac{1}{2^b}\right) = -\log_2 \left(\frac{1}{2^b}\right) = b$$

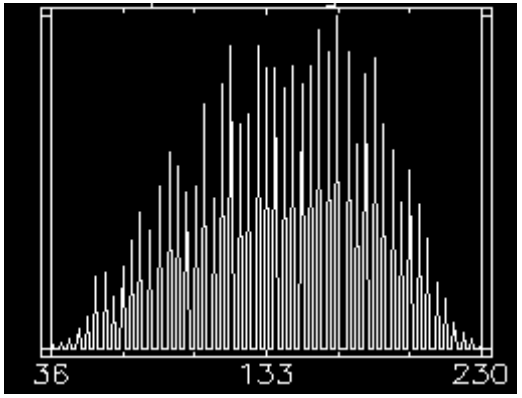
- We see that $0 \leq H \leq b$ (b = number of bits per pixel)

- Here, we use entropy as a texture feature, computed in a local window.

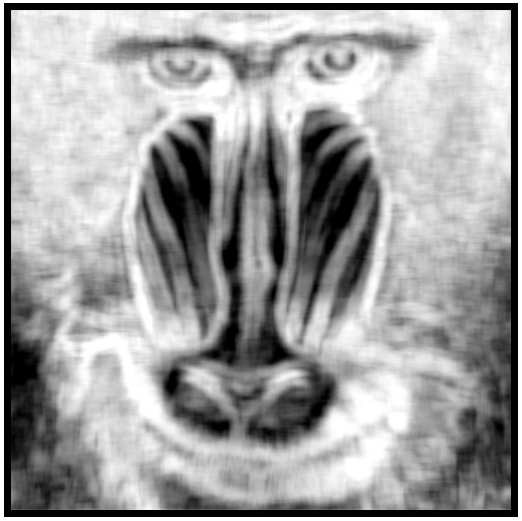
Entropy example

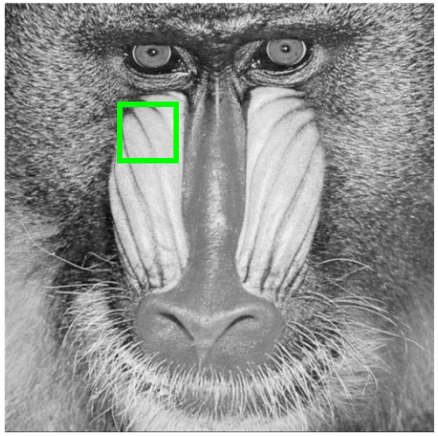


Region 1

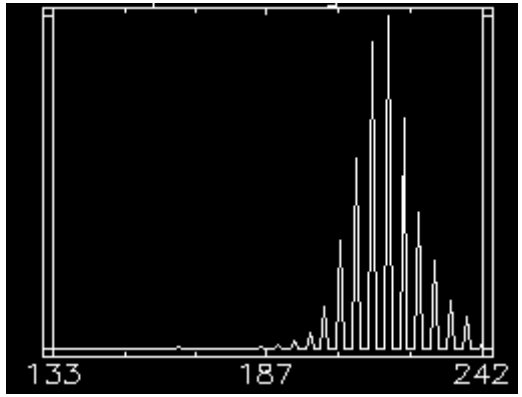


Histogram - Region 1





Region 2



Histogram - Region 2

Entropy feature
Computed in 15x15
Window

Region1: high entropy

Region2: low entropy

First order statistics - Energy

- Energy (how non-uniform?)

$$E = \sum_{i=0}^{G-1} [P(i)]^2$$

- A measure of homogeneity
- If all P(i) are equal (histogram is uniform), E=1/G
- If the image contains only one gray level:

  $E=(G-1)\times 0+1\times 1=1$

- **Thus, $1/G \leq E \leq 1$**

1. order statistics discussion

- 1. order statistics can separate two regions even if $\mu_1 = \mu_2$, as long as $\sigma^2_1 \neq \sigma^2_2$, or skewness or kurtosis differ
- The statistics of a pixel *(x, y)* is found in a local window

- Problems:
  - Edges around objects will be exaggerated
    - Solution: use adaptive windows

  - 1. order statistics does not describe geometry or context
    - Cannot discriminate between

    - Solution:
      - Use 2. or higher order statistics.

Second order statistics

- **Gray-level Co-Occurrence matrices**
  - Intensity-change-"histograms"
    as a function of distance and direction
  - By far the most popular texture description method due to its simplicity
  - The co-occurrence matrix is
    an estimate of the second order joint probability,
    - which is the probability of
      - going from gray level i to gray level j,
      - given the distance d between two pixels
      - along a given direction θ.

Gray Level Coocurrence Matrices (GLCM)

- Matrix element *P(i,j)* in a GLCM is 2. order probability of changing from graylevel *i* to *j* when moving a distance *d* in the direction θ of the image, or equivalent, (*Δx*, *Δy)*

- From a *M* × *N* image with *G* graylevels, and *f(m,n)* is the intensity. Then $P(i,j \mid \Delta x, \Delta y) = WQ(i,j \mid \Delta x, \Delta y)$, where

$$W = \frac{1}{(M-\Delta x)(N-\Delta y)} \quad , \quad Q(i,j \mid \Delta x, \Delta y) = \sum_{n=1}^{N-\Delta y} \sum_{m=1}^{M-\Delta x} A$$

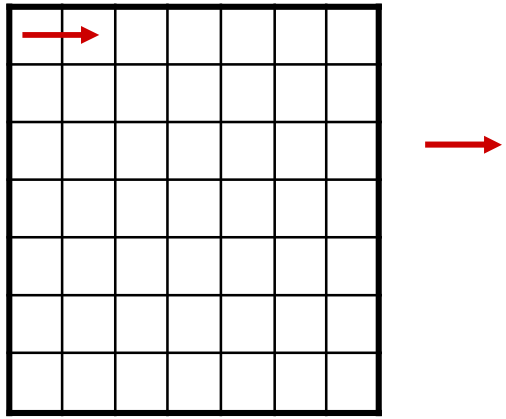
and

$$A = \begin{cases} 1 & \text{if } f(m,n) = i \text{ og } f(m+\Delta x, n+\Delta y) = j \\ 0 & \text{else} \end{cases}$$

- Alternative notation, dependent on distance and direction, *P(i,j* | *d,* θ*)*

GLCM

- From one window of size $w \times w$ we get one GLCM matrix for a given (d,θ)
- The dimension of the co-occurrence matrix is *GxG* if we have *G* gray-levels in the image.
- Choose a distance d and a direction θ



In this example, $d=1$ and $\theta=0$

- Check all pixel pairs with distance d and direction θ inside the window. $Q(i,j/d,\theta)$ is the number of pixel pairs where pixel 1 in the pair has pixel value i and pixel 2 has pixel value j.

GLCM

Image

0	1	1	2	3
0	0	2	3	3
0	1	2	2	3
1	2	3	2	2
2	2	3	3	2

d=1, θ=0 correspond to *dy=0, dx=1*

This row has no neighbors to the right. The number of pixel pairs that we can compute is N×(M-1) = 5 × (5-1) = 20

$Q(i,j|d,\theta)$

gray level ***j***

gray level ***i***

1	2	1	0
0	1	3	0
0	0	3	5
0	0	2	2

Cooccurrence Matrix

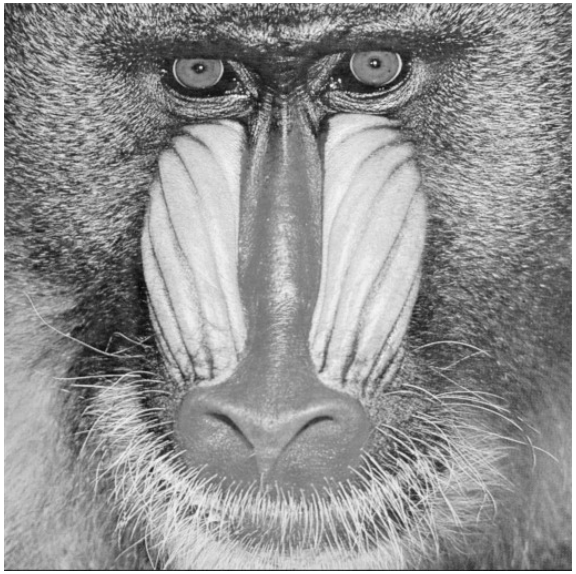
	j=0	1	2	3
i= 0	1/20	2/20	1/20	0
1	0	1/20	3/20	0
2	0	0	3/20	5/20
3	0	0	2/20	2/20

$P(i,j|d,\theta)$ is normalized by W, the number of pixel pairs inside the window.

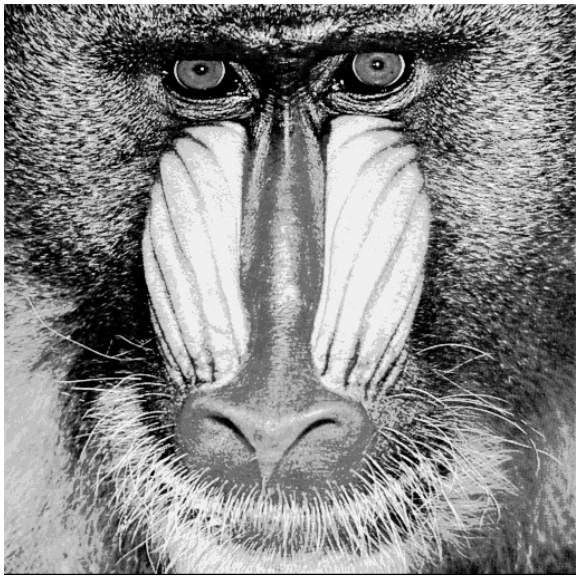
GLCM – practical issues

- The matrix must have a sufficient "average occupancy level" :
  - Reduce number of graylevels
    (Less precise description if the texture has low contrast)
    - Select **L** =number of gray levels
    - Rescale the image if necessary to use these levels using (histogram transform)
    - Requantize the scaled image from **G** to **L** gray levels before GLCM computation
  - Increase window size (Errors due to changes in texture)
- Heuristics:
  - *L = 16* graylevels is usually sufficient
  - window should be *31 x 31* – *51* x *51* pixels.

Preprocessing examples

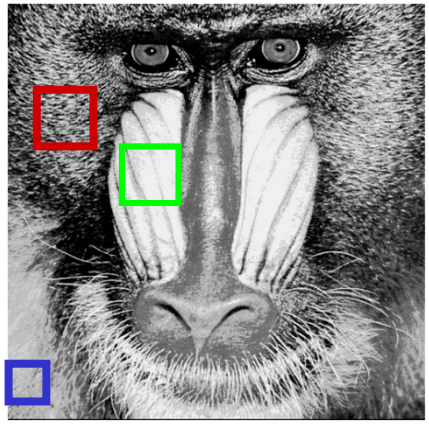


Original image



Histogram-equalized and requantized to 16 gray levels

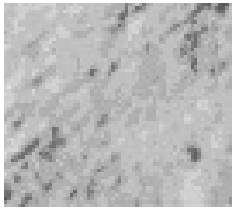
GLCM matrices for subregions



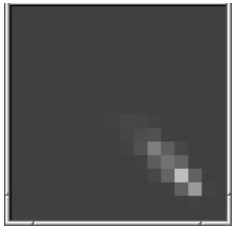
Subregions of image

Nose

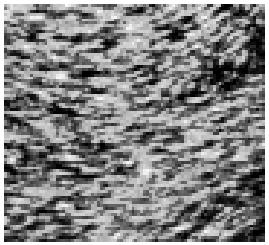
Fur2



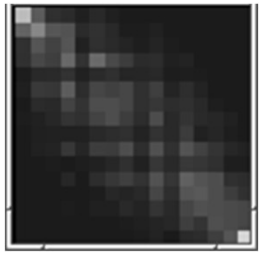
GLCM matrices
d=1, θ=0



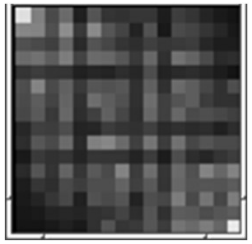
Fur1



GLCM matrix
d=1, θ=0



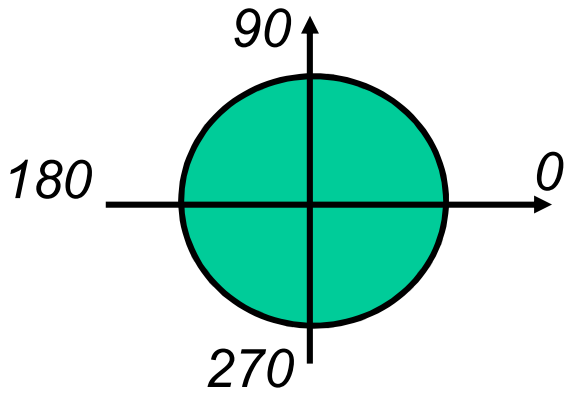
GLCM matrix
d=1, θ=90



GLCM

- Usually a good idea to reduce the number of (d,θ) combinations evaluated

- Simple pairwise relations:
  - $P(d,0^0) = P^t(d,180^0)$
  - $P(d,45^0) = P^t(d,225^0)$
  - $P(d,90^0) = P^t(d,270^0)$
  - $P(d,135^0) = P^t(d,315^0)$



  - *Symmetric GLCM:*
    - *Count "forwards" + "backwards"*

- Isotropic cooccurrence matrix by averaging

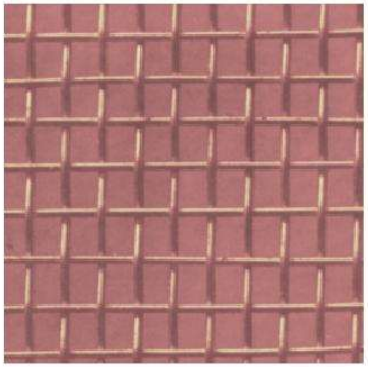
$$P(\theta),\ \theta \in \{0^o, 45^o, 90^o, 135^o\}$$

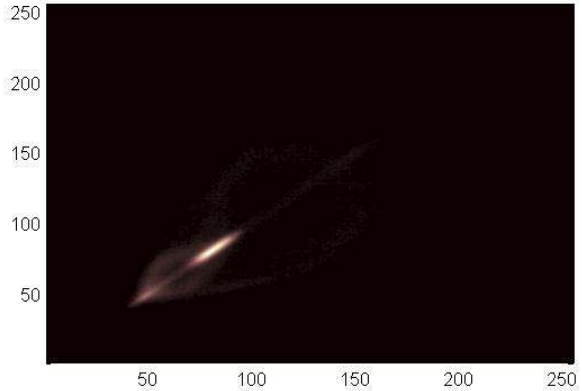
  - *Beware of differences in effective window size!*

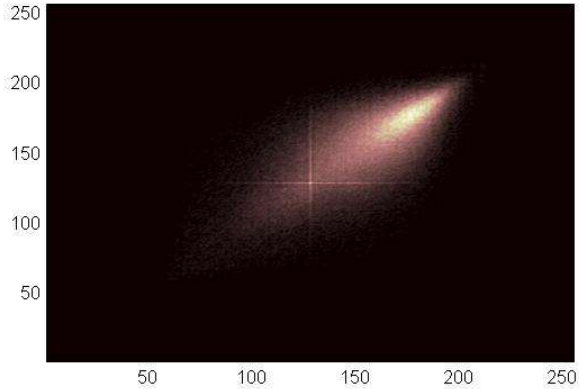
- An <u>isotropic texture</u> is equal in all directions

- If the texture has a clear orientation, we select θ according to this.

Isotropic GLCM example







How to use the GLCM

- Usually, used by extracting secondary features from GLCM
  - Haralick et al. and Conners et al.
  - Features are usually strongly correlated, using more than 4-5 simultaneously is not advisable
  - You may need to evaluate several distances *d*
    - *Would you perform anti-aliasing filtering for d>1 ?*
  - Optimal set of features is problem dependent

- It may be advisable to preprocess by histogram transform to remove effect of absolute gray level.

- Often, we want to make the features ”rotation” invariant by using the isotropic GLCM (remember different weights).

Classical GLCM features

- There are a number of *scalar* texture features that characterize the cooccurrence matrix directly and the image indirectly.

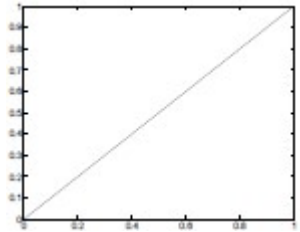
- Many of these GLCM features may be seen as a weighted sum of the cooccurrence matrix element values, where the weighting applied to each element is based on a given *weighting function.*

- By varying this weighting function, different types of information about the texture can be extracted.

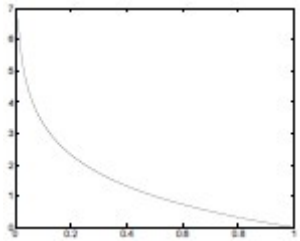
- The weighting functions fall into two general categories:
    1. Weighting based on the *value* of the GLCM element
    2. Weighting based on the *position* within the GLCM

Value-based GLCM Features

- Angular Second Moment, $ASM = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \{P(i,j)\}^2$
  - ASM is a measure of homogeneity of an image.
  - Homogeneous scene will contain a few gray levels, giving a GLCM with few but high values of *P(i,j)*.
  - Thus, the sum of squares will be high.



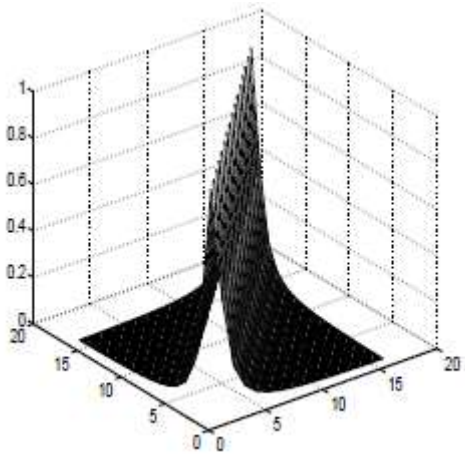
- Entropy, $E = -\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \times log(P(i,j))$
  - Inhomogeneous scenes have high entropy, while a homogeneous scene has a low entropy.
  - *Maximum Entropy is reached when all 2. order probabilities are equal.*
  - *Note that 0 ≤ P(i,j) ≤ 1.*



Position-based GLCM Features

- Inverse Difference Moment (also called homogeneity)

$$IDM = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \frac{1}{1+(i-j)^2} P(i,j)$$



  - So the weight function is

$$\mathrm{W} = \frac{1}{1+(i-j)^2}$$

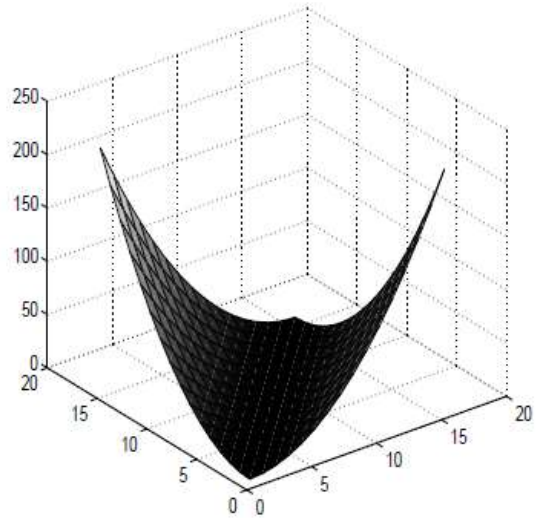
  - IDM is influenced by the homogeneity of the image.
  - Because of the weighting factor, IDM will get small contributions from inhomogeneous areas ($i \neq j$).
  - The result is a low IDM value for inhomogeneous images, and a relatively higher value for homogeneous images.

Position-based GLCM Features-2

- Inertia, also called Contrast:

$$\text{Inertia} = \sum_{i=1}^{G}\sum_{j=1}^{G}(i-j)^2 P(i,j)$$

$$W(i,j) = (i-j)^2$$

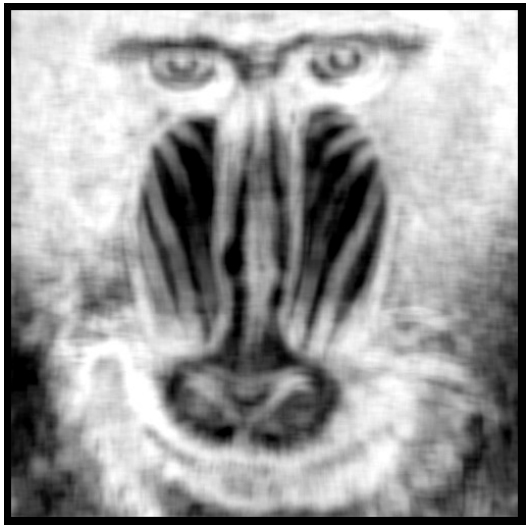


- The Inertia weighting function is zero along the diagonal (i = j), and increases towards $(G - 1)^2$ away from the diagonal.

- Thus, it will favor contributions from P(i, j) away from the diagonal (i ≠ j), i.e., give higher values for images with high local contrast.

GLCM feature image examples, w= 15

GLCM contrast



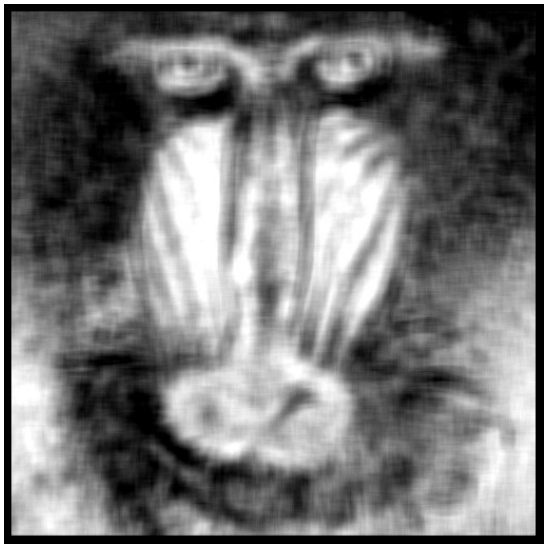
GLCM entropy

GLCM variance

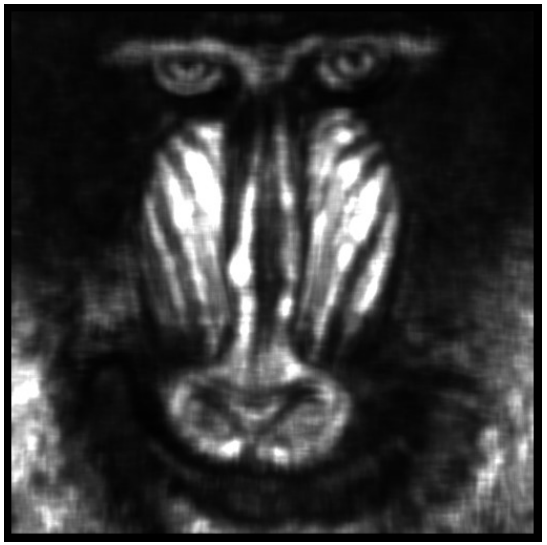
GLCM contrast is
- negative correlated with IDM
- positively correlated with variance

GLCM entropy is negatively correlated with ASM

GLCM feature image examples, w= 15



GLCM IDM



GLCM ASM

GLCM correlation

Aristotle and Occam

- Our search for models or hypotheses that describe the laws of nature is based on a ”minimum complexity principle”.
- Aristotle (384-322 BC), Physics, book I, chapter VI:
  ‘The more limited, if adequate, is always preferable’.
- William of Occam (1285-1349):
  ‘Pluralitas non est ponenda sine necessitate’.
- The simplest model that explains the data is the best.
- So far, “Occam’s Razor” has generally motivated the <u>search and selection</u> of reduced dimensionality feature sets.
- It should also motivate us to <u>generate</u> only a few powerful features.

The ”curse-of-dimensionality”

- Also called ”peaking phenomenon”.
- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- The implication is that:
- For a high measurement complexity, we will need large amounts of training data in order to attain the best classification performance.
- => 5-10 samples / feature / class.

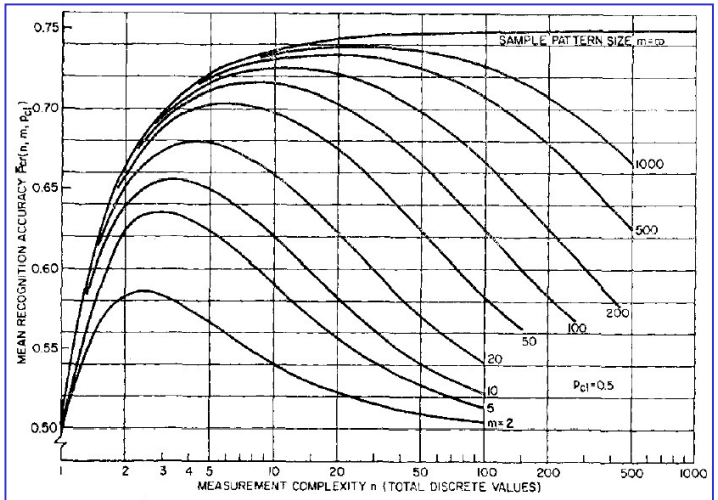


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.
Illustration from G.F. Hughes (1968).

Feature subsets

- A goal is to find the subset of observed features which
  - best characterizes the differences between groups
  - is similar within the groups
  - **Maximize the ratio of between-class and within-class variance.**

- If we want to perform an exhaustive search through D features for the optimal subset of the d ≤ m "best features", the number of combinations to test is

$$n = \sum_{d=1}^{m} \frac{D!}{(D-d)!\,d!}$$

- Impractical even for a moderate number of features!

m ≤ 5, D = 100 => n = 79.374.995

Which GLCM θ to use

- Remember the simple pairwise relations:
  - $P(d,0^0) = P^t(d,180^0)$
  - $P(d,45^0) = P^t(d,225^0)$
  - $P(d,90^0) = P^t(d,270^0)$
  - $P(d,135^0) = P^t(d,315^0)$

  - *Symmetric GLCM:*
    - *Count "forwards" + "backwards"*

- If you are to distinguish betw. oriented patterns, use a GLCM direction, θ, that optimizes differences, not similarities.

- If you are to distinguish betw. textures that differ in contrast or homogeneity, use θ and d that capture the differences, combined with features with weight functions like

  « Inertia / Contrast»: $(i-j)^2$

Low-D adaptive weighting functions

- The GLCM features described are *predefined and non-adaptive , and we need a number of them to describe the matrix content.*

- Instead, we could find a few weighting functions <u>adapted to the problem</u>.

- This would extract information from the parts of the GLCM matrix <u>that actually contain information about texture differences</u> between e.g., two different clinical classes (say, good and bad prognosis).

- Given a number of images of each of two classes, compute
  - the mean $P(i,j|\omega)$, for each class, $\omega=1,2$
  - the variance $\sigma^2(i,j|\omega)$ of P , for each class, $\omega=1,2$

  for each position (i,j) of the GLCM.

Low-D adaptive weighting functions

- Then for each position (i,j), compute two new matrices:
  - the class difference matrix value (difference between the means)

$$\Delta(i,j) = P(i,j|\omega=1) - P(i,j|\omega=2),$$

  - and the *Mahalanobis* class distance matrix (Q: what is this J(i,j)?)

$$J(i,j) = 2\,\Delta^2(i,j) \,/\, (\sigma^2(i,j|\omega=1) + \sigma^2(i,j|\omega=2)\,).$$

- *Then use the Malalanobis distance matrix as the weighting function on each GLCM matrix obtained from a (new) image, to produce only two features, one for the positive and one for the negative partition of Δ(i,j).*

See Kleppe, Albregtsen, Vlatkovic et al., The Lancet Oncology (2018)

"Chromatin organisation and cancer prognosis: A pan-cancer study»

Sum and difference histograms

- The sum histogram *S* is simply the histogram of the sums of pixels *dx* and *dy* apart
- For example, the gray level at *I(x,y)* is added to the gray level at *I(x+dx,y+dy)* and the histogram bin corresponding to that sum is incremented
- The difference histogram D is simply the histogram of the difference of pixels *dx* and *dy* apart

$$s_{\Delta x,\Delta y}(m,n) = f(m,n) + f(m+\Delta x, n+\Delta y)$$

$$d_{\Delta x,\Delta y}(m,n) = f(m,n) - f(m+\Delta x, n+\Delta y)$$

- The number of possible values of sum and difference histogram is *2G-1.*

Sum and difference histograms

- GLCM features can be derived from P_s and P_d
- Example:
  - Contrast from P_d

$$CON = \sum_{j=0}^{2G-2} j^2 P_d(j \mid \Delta x, \Delta y)$$

  - Contrast from GLCM

$$CON = \sum_{n=0}^{G-1} n^2 \{ \sum_{i=1}^{G} \sum_{j=1}^{G} P(i,j) \}, \qquad \mid i - j \mid = n$$

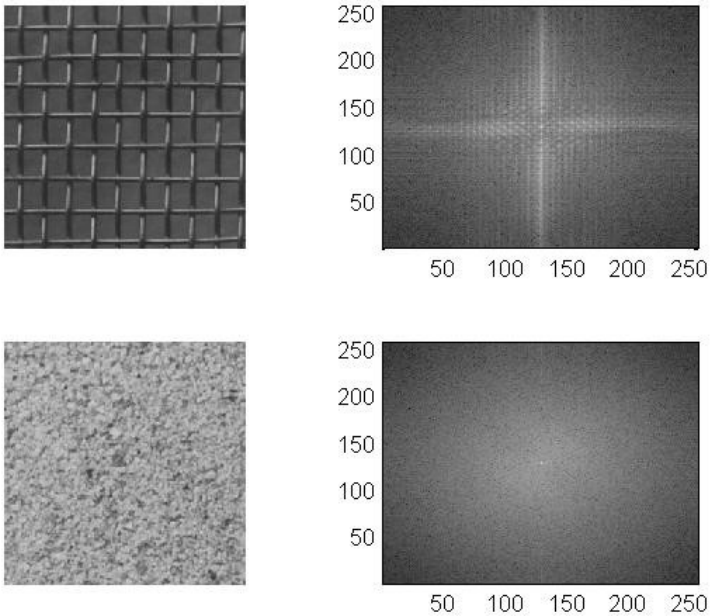
- Some of the features mentioned earlier is derived from the histograms
  - Sum Average, $AVE = \sum_{i=0}^{2G-2} P_s$
  - Sum Entropy, $SEN = -\sum_{i=0}^{2G-2} P_s(i) \log (P_s(i))$
  - Difference Entropy, $DEN = -\sum_{i=0}^{G-1} P_d(i) \log(P_d(i))$
  - Inverse Difference moment, $IDM = \sum_{i=0}^{G-1} \frac{1}{1+i^2} P_d(i)$

Fourier analysis

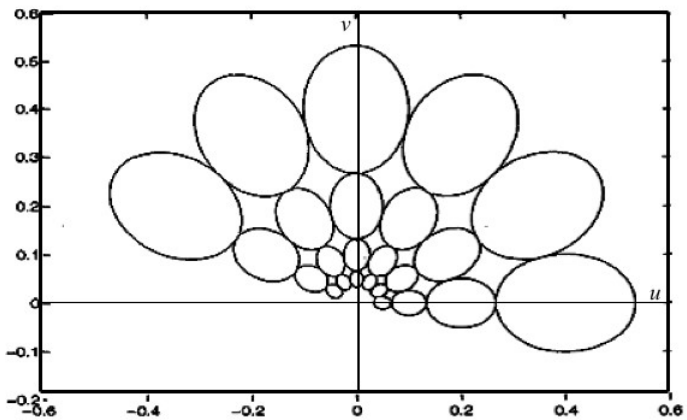
- The Fourier spectra give direction and frequency for periodic or near periodic 2D patterns
- Local FFT in windows
- Texture with a dominating direction will have peaks in the spectra along a line orthogonal to the texture orientation
- High frequency = fine texture
  = peaks in the spectra far from the origin
- Thus it is possible to separate fine and coarse spectra
- The spread in image frequencies = width of the peak in Fourier
- Isotropic textures with a defined frequency can be seen as rings in the spectra
- Scalar features can be extracted by integration over rings, wedges or from results of Gabor filtering (next slide)

Fourier analysis example

- Transform to Fourier domain, integrate over rings or wedges



Gabor filters combine estimate of orientation and frequency.



Gray level run length statistics

- The GLRLM method extracts *higher-order statistical texture information* from digital images.
- A set of consecutive pixels with the same gray-level, colinear in a given direction, constitute a gray-level run.
- The *run length* is the number of pixels in the run.
- *Run length value* is the number of occurrences of a run.
- The normalized GLRLM contains the run length values divided by the total number of runs in the image.
- A number of value- and position-dependent weightings.
- The adaptive approach has proven useful.

Cooccurrence of Gray Level Runs

- As an alternative to the GLCM and GLRLM methods, we introduced the cooccurrence of gray-level run length matrix (CGLRLM) method.

- The four-dimensional (4D) normalized matrix P(i, j, k, l) may be seen as a natural extension of the 2D gray-level run length matrix, containing the estimated probability of cooccurrence of two runs of *(gray-level, run length)* = (i, j) and (k, l).

- One could specify any geometrical relationship between the two runs. However, the most fruitful relationship to capture aspects of the texture primitives is to consider neighboring runs.

- The 4D normalized cooccurrence probability matrix P(i, j, k, l) may be replaced by its associated 2D *sum* and *difference run-length matrices*, and adaptive features may be extracted from these.

Learning goals - texture

- Understand what texture is, and the difference between first order and second order measures
- Understand the GLCM matrix, and be able to describe algorithm
- Understand how we go from an image to a GLCM feature image
  - Preprocessing, choosing d and θ, selecting some features that are not too correlated
- **There is no optimal texture features, it depends on the problem**
- **A good tutorial on texture: http://www.fp.ucalgary.ca/mhallbey/tutorial.htm**

Exercise & next lecture

- Exercise: implement your own GLCM function,
  and find the zebras in the images.
  Happy zebra hunting!, See the web page.

- Next lecture: Introduction to Hough transform:
  Using gradient information to detect & represent lines.
  Detecting circles and ellipses, triangles and polygons.

IN5520 / 9520 – Digital Image Analysis

REPETITION

•Central concepts from IN 2070 Digital Image Processing

Anne H.S. Solberg 31.08.2021

INF2310 – a brief repetition

- See https://www.uio.no/studier/emner/matnat/ifi/IN2070/v21/undervisningsplan/

- Topics covered in the course:
  - Image representation, sampling and quantization.
  - Compression and coding
  - Color imaging
  - Grey-level mapping
  - Geometrical operations

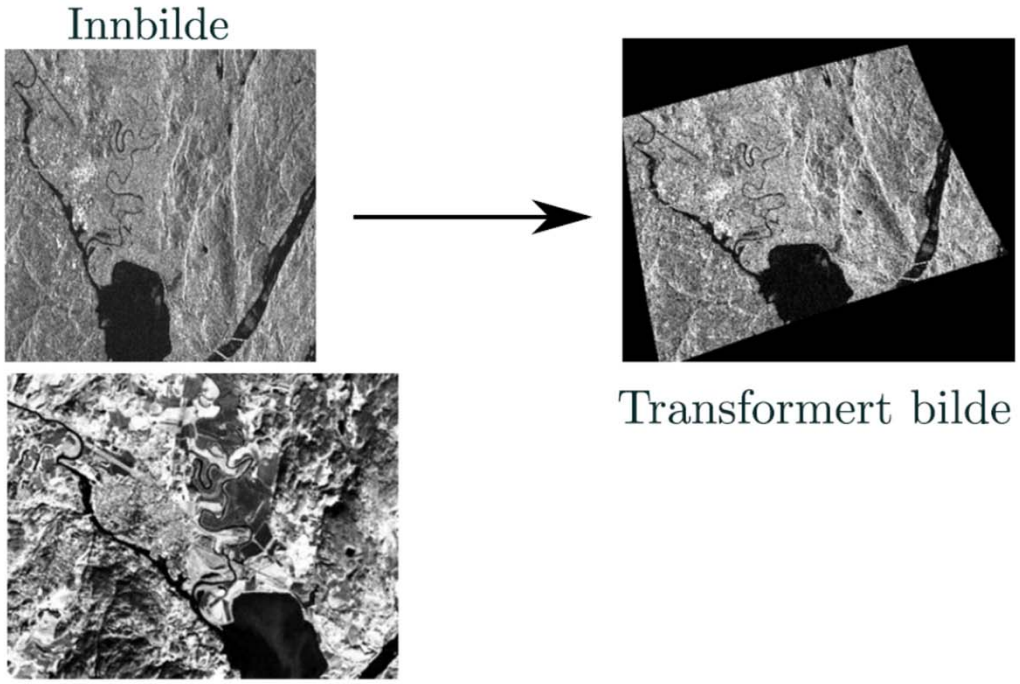
  Assumed known

  - Filtering and convolution in the image domain
  - Fourier transform
  - Segmentation by thresholding
  - Edge detection

  Good understanding needed

1 slide: geometrical operations

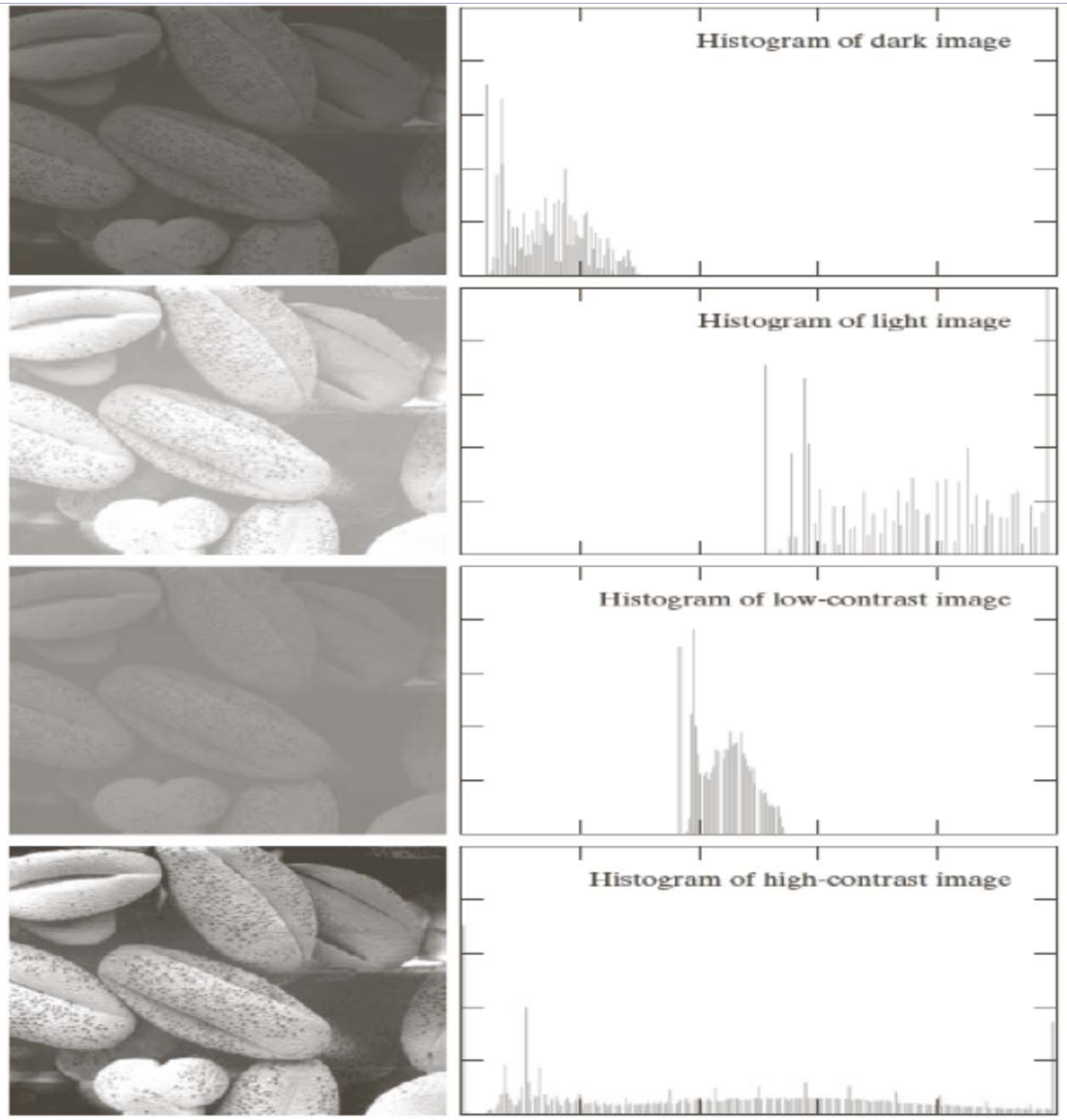
- Given 2 images, do affine transforms on the pixel coordinates for applications like:
  - Correcting geometrical errors from image aquisition.
  - Co-register 2 images of the same area, taken at different time and/or different sensors



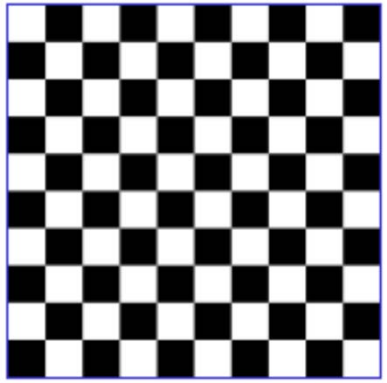
Gray-level mapping

- How to change the contrast in an image
- Normalize contrast across a data set

Image histograms



How does the histogram look like?



Typical application: image standarization

- Given a set of images to train and test a classifier.
- The trained classifier typically depends on image contrast, so we often need to standardize contrast over the image dataset.
- Most common normalization:
- Normalize to mean 0 and standard deviation 1 by subtracting the image mean and dividing by the standard deviation.

2-D convolution

- Input image f(x,y)
  The resulting image *g(x,y)* is given by

$$g(x,y) = \sum_{j=-w_1}^{w_1} \sum_{k=-w_2}^{w_2} h(j,k) f(x-j, y-k)$$

$$= \sum_{j=x-w_1}^{x+w_1} \sum_{k=y-w_2}^{y+w_2} h(x-j, y-k) f(j,k)$$

- *h* is a *m×n* filter with size $m=2w_1+1$, $n=2w_2+1$
- The result is a weighed sum of the input pixels surrounding pixel *(x,y).* The weights are given by *h(j,k).*
- The pixel value of the next pixel in the out image is found by moving the filter one position and computing again.

How to compute a convolution

$$g(x,y) = \sum_{s=x-a}^{x+a} \sum_{t=y-b}^{y+b} h(x-s, y-t) f(s,t)$$

- Rotate the filter 180 degrees (if filter not symmetric)
- Place the rotated filter with its origin at location (x,y) in the image f(s,t).
- Multiply each filter coefficient with the underlying image pixel values.
- Sum this responses over the filter to get the result g(x,y)
- To compute for all pixels in an image:
  - Move the filter around and compute the response for each pixel (x,y).

Size of output image

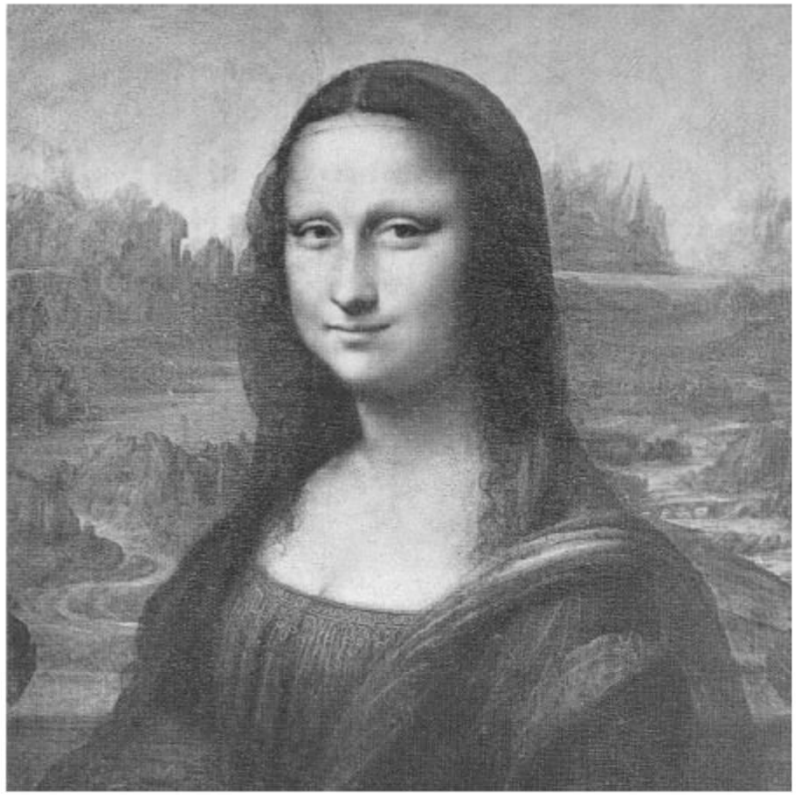
- The definition states that the convolution is computed at every location where the filter overlaps with the image.
- This will result in an output larger than the input.
- For normal filtering, we normally only compute the response for every pixel in the input image to get input and output of the same size.
- Exceptions: when convolving two filters to create a larger filter.

Separable filters

- Geometrical shapes: rectanglar and square
- Rectangular mean filters are separable.

$$h(i,j) = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

- Advantage: fast filtering
- Note: this is a low pass filter that will blur the image and remove noise.



Original

Filtrert med
3x3-middelverdifilteret

Filtrert med
9x9-middelverdifilteret

Filtrert med
25x25-middelverdifilteret

Non-uniform low pass filters

- 2D Gauss-filter:

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

- Parameter σ is standard deviation (width)
- Filter size must be set relative to σ

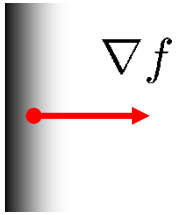
Digital gradient operators

- The gradient of *f(x)* is $\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$

- The (intensity) gradient of an image:

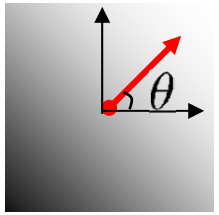
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient points in the direction of most rapid (intensity) change



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Gradient operators

- Prewitt-operator

$$H_x(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \; H_y(i,j) = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- Sobel-operator

$$H_x(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; H_y(i,j) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Frei-Chen-operator

$$H_x(i,j) = \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}, \; H_y(i,j) = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

Gradient direction and magnitude

- Horisontal edge component:
  - Compute $g_x(x,y)=H_x*f(x,y)$

  => Convolve with the horisontal filter kernel H_x
- Vertical edge component:
  - Compute $g_y(x,y)=H_y*f(x,y)$

  => Convolve with the vertical filter kernel H_y

The *gradient direction* is given by:

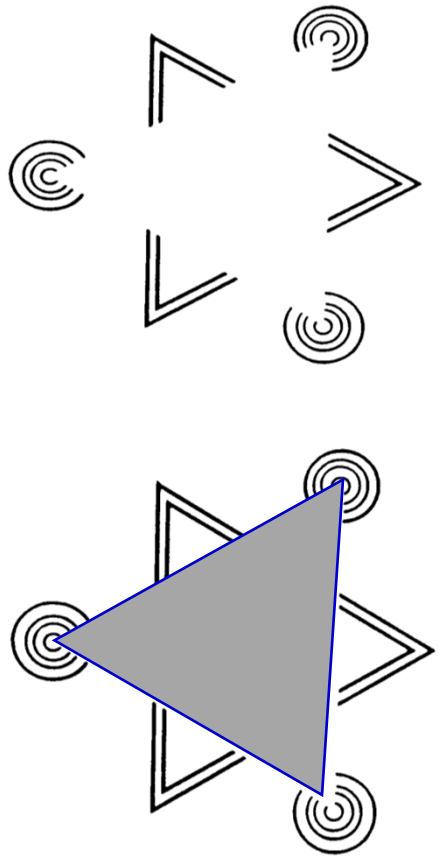
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

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

Edge extraction

- Several basic edge extraction techniques were taught in INF2310
- In this context edges are both edges in intensity, color and texture
- Edges are important for many reasons:
  - Much of the information in an image is contained in the edges. In many cases semantic objects are delineated by edges
  - We know that biological visual systems are highly dependent on edges



Edge extraction

- The standard operator is the so called Sobel operator.
- In order to apply Sobel on an image you convolve the two x- and y-direction masks with the image:

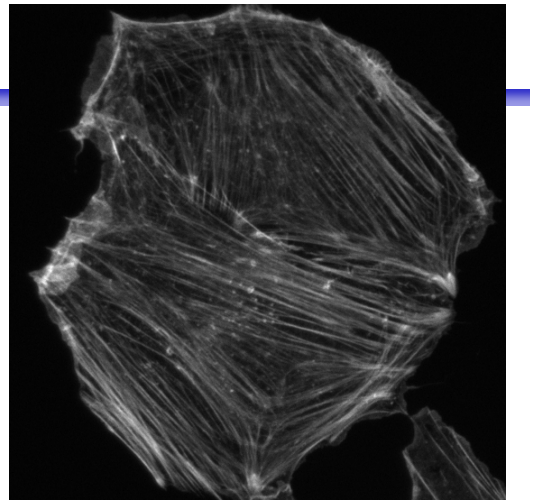
-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

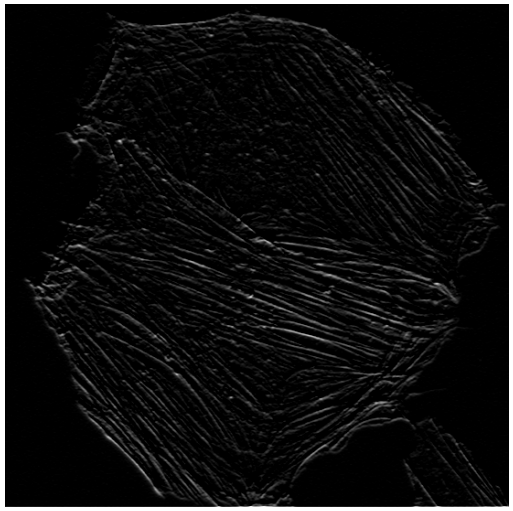
Edge extraction - Sobel

- This will give you two images, one representing the horizontal components of the gradient, one representing the vertical component of the gradient.

- Thus using Sobel you can derive both the local gradient magnitude and the gradient direction.



Grayscale image



«Horizontal» edges

Edge extraction - Laplace

- Another frequently used technique for edge detection is based on the use of discrete approximations to the *second derivative.*
- The *Laplace operator* is given by

$$\nabla^2(f(x,y)) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- This operator changes sign where f(x,y) has an inflection point, it is equal to zero at the exact edge position

Edge extraction - Laplace

- Approximating second derivatives on images as finite differences gives the following mask

$$\nabla^2(f(x,y)) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\approx -f(x-1,y) + 2f(x,y) - f(x+1-y)$$

$$- f(x,y-1) + 2f(x,y) - f(x,y+1)$$

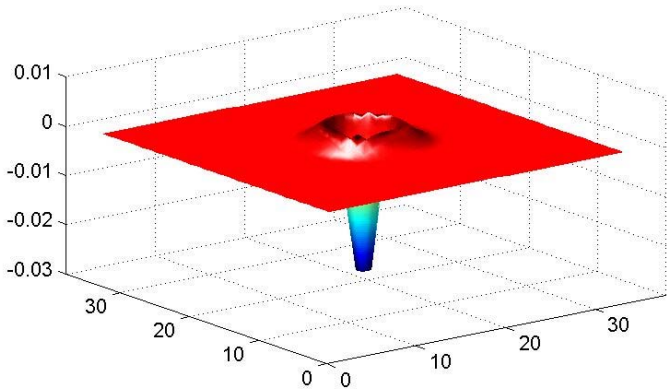
0	-1	0
-1	4	-1
0	-1	0

Edge extraction - LoG

- Since the Laplace operator is based on second derivatives it is extremely sensitive to noise.

- To counter this it is often combined with Gaussian pre-filtering in order to reduce noise.

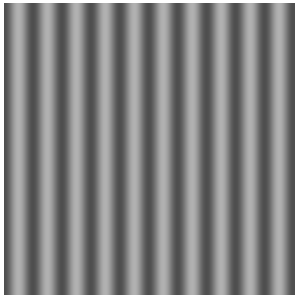
- This gives rise to the so called Laplacian-of-Gaussian (LoG) operator.



Sinusoids in images

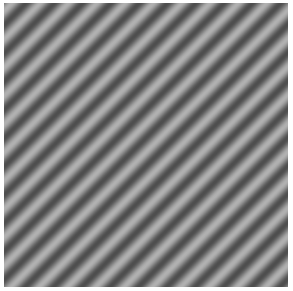
$$f(x,y) = 128 + A\cos\left(\frac{2\pi(ux+vy)}{N} + \phi\right)$$

A - amplitude
u - horisontal frequency
v - vertical frequency
ϕ - phase

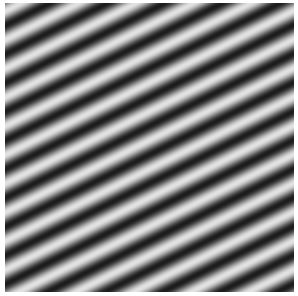


A=50, u=10, v=0

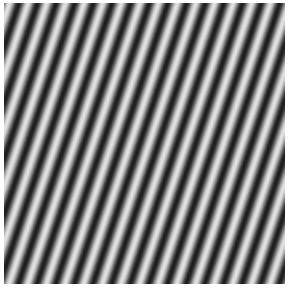
A=20, u=0, v=10



A=50, u=10, v=10



A=100, u=5, v=10



A=100, u=15, v=5

Note: u and v are the number of cycles (horisontally and vertically) in the image

2-D Discrete Fourier transform (DFT)

f(x,y) is a pixel in a *N*×*M* image

Definition:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

$$\boxed{e^{j\theta} = \cos\theta + j\sin\theta}$$

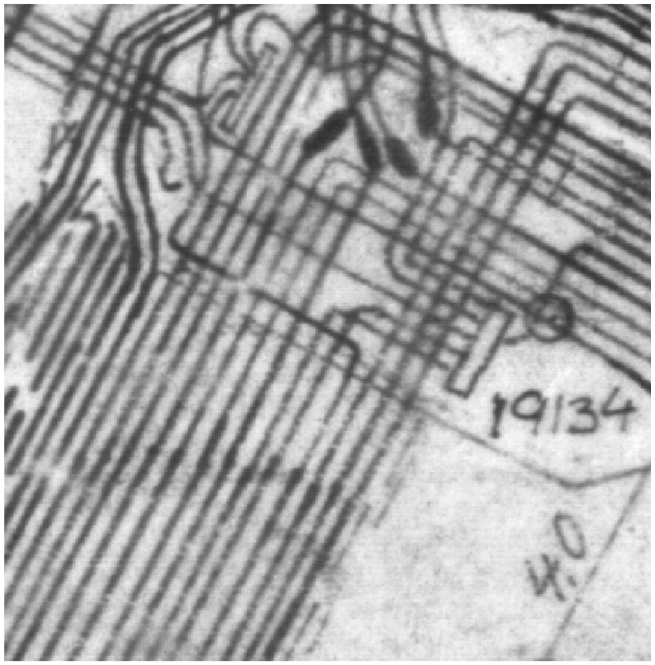
This can also be written:

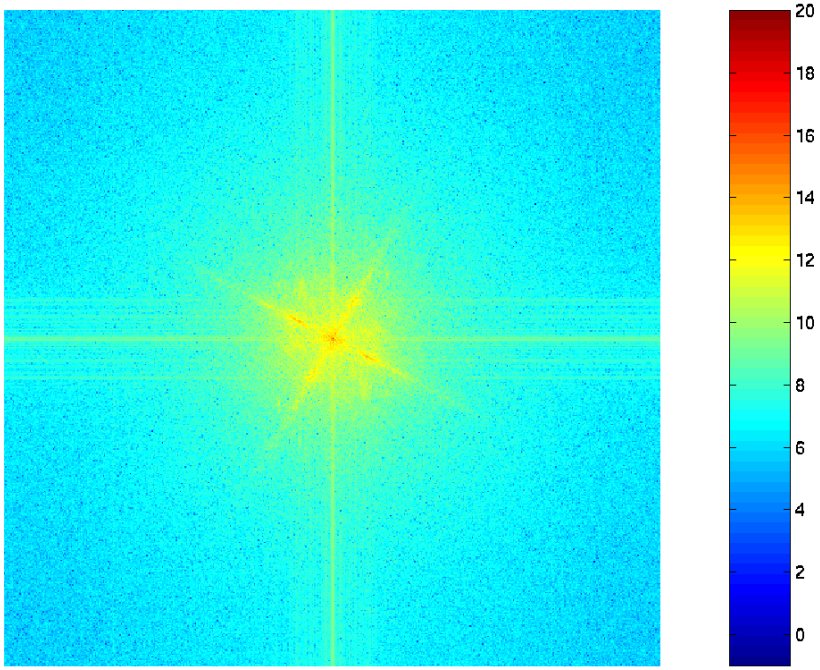
$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\left[\cos(2\pi(ux/M+vy/N)) - j\sin\left(2\pi(ux/M+vy/N)\right)\right]$$

Inverse transform:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

Example – oriented structure





The convolution theorem

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v) \cdot H(u,v)$$

Convolution in the image domain

$\Leftrightarrow$

Multiplication in the frequency domain

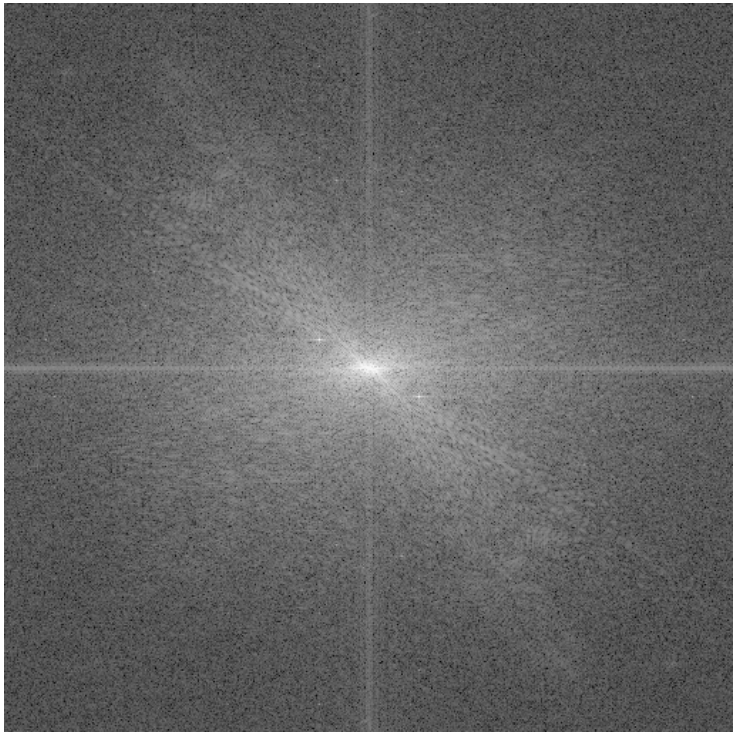
$$f(x,y) \cdot h(x,y) \Leftrightarrow F(u,v) * H(u,v)$$

Multiplication in the image domain

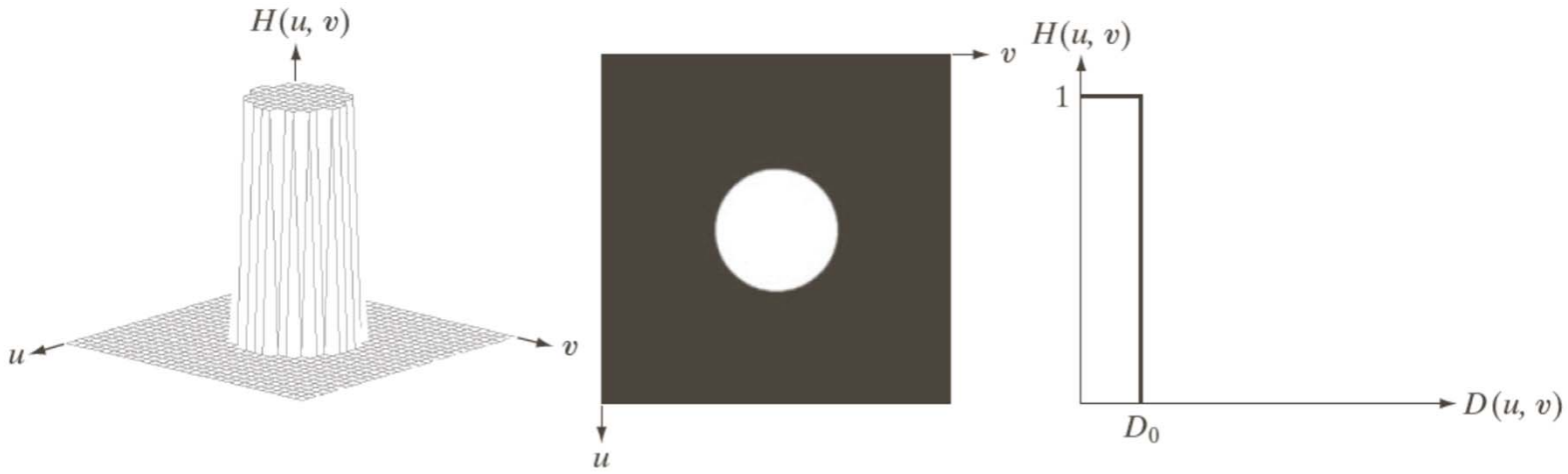
$\Leftrightarrow$

Convolution in the frequency domain

How do we filter out this effect?



The ”ideal” low pass filter



a b c

FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Example - ideal low pass

Original

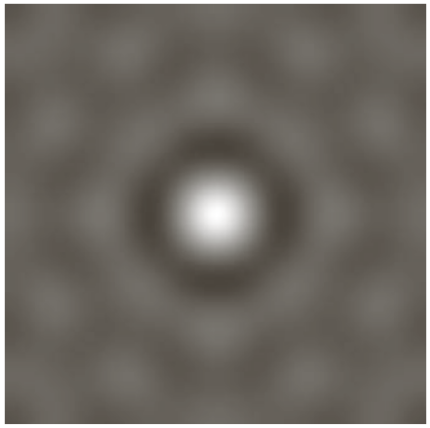
D_0=0.2

D_0=0.3

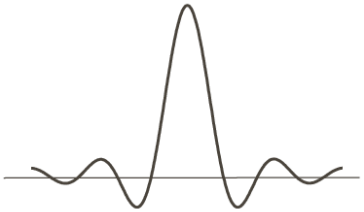
Look at these image in high resolution.
You should see ringing effects in the two rightmost images.

What causes the ringing effect?

Ideal lowpass in the image domain



fft of H(u,v)
for ideal lowpass

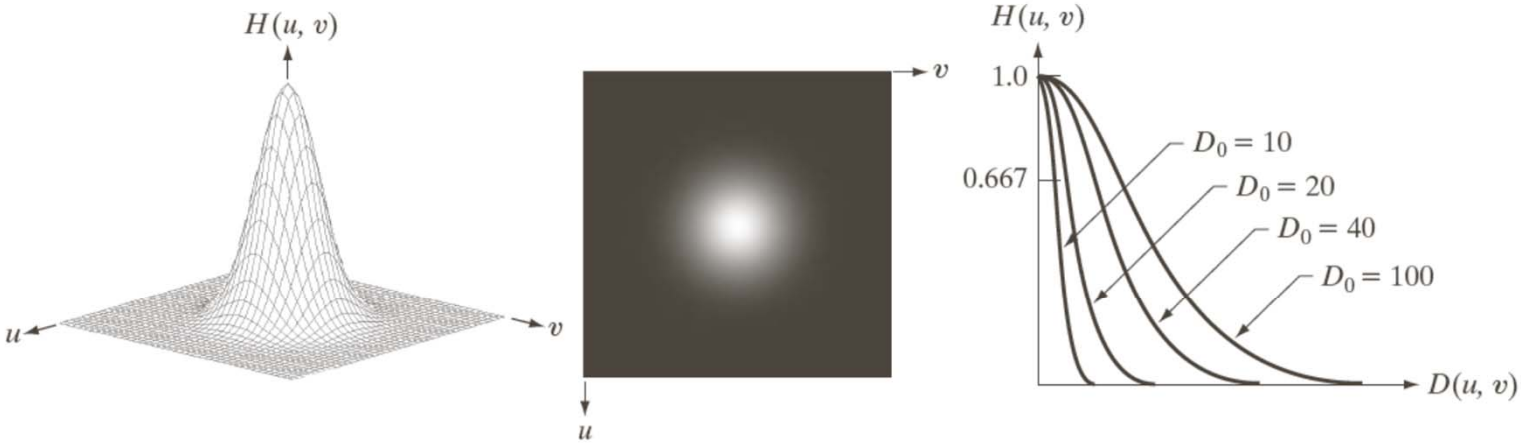


1D profile
for ideal lowpass

- Note that the filter profile has negative coefficients
- It has similar profile to a Mexican-hat filter (Laplace-of-Gaussian)
- The radius of the circle and the number of circles per unit is inversely proportional to the cutoff frequency
  - Low cutoff gives large radius in image domain

Gaussian lowpass filter

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$



a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

High pass filtering

- Simple (”Ideal”) high pass filter:

$$H_{hp}(u,v) = \begin{cases} 0, D(u,v) \le D_0, \\ 1, D(u,v) > D_0. \end{cases}$$

$$or$$

$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

- Gaussian high pass filter:

$$H_{hpG}(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

Ideal, Butterworth and Gaussian highpass filters

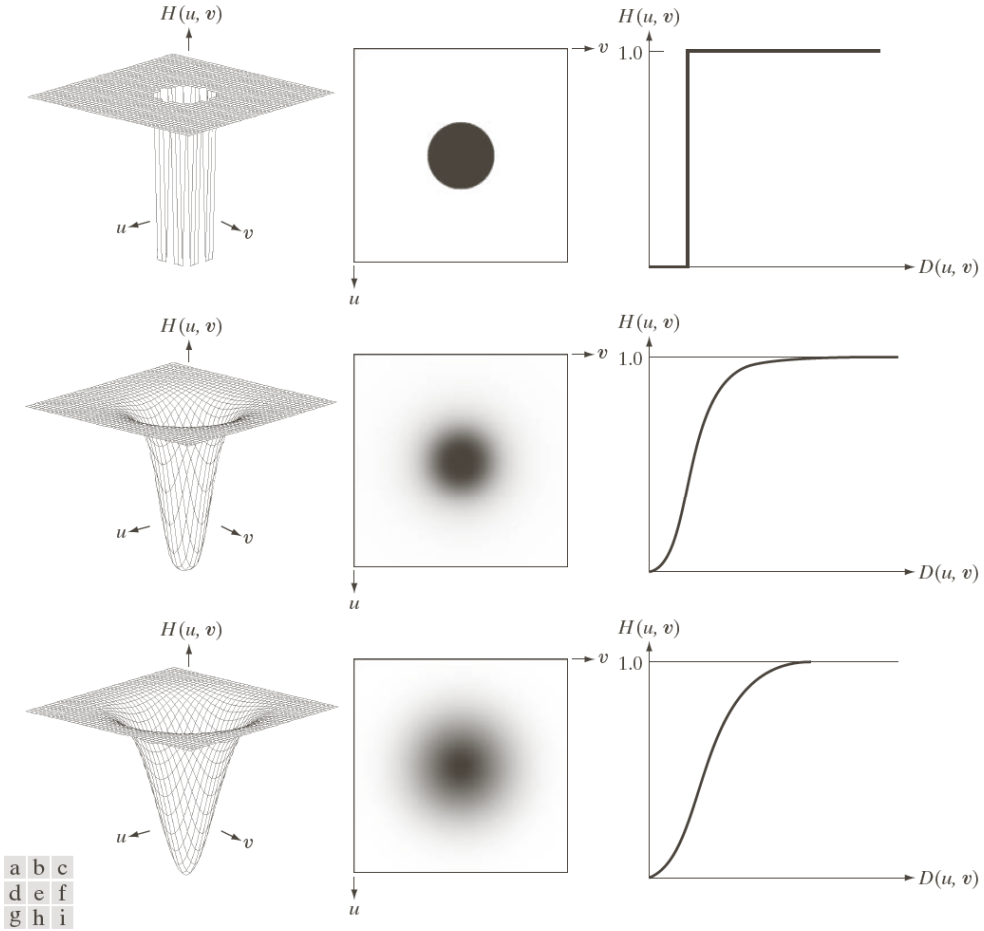


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.

Bandpass and bandstop filters

- Bandpass filter: Keeps only the energy in a given frequency band $<D_{low},D_{high}>$ (or $<D_0-W/2,D_0+W/2>$)

- W is the width of the band
- D_0 is its radial center.

- Bandstop filter: Removes all energy in a given frequency band $<D_{low},D_{high}>$

Bandstop/bandreject filters

- Ideal

$$H_{bs}(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \le D(u,v) \le D_0 + \frac{W}{2} \\ 1 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$$

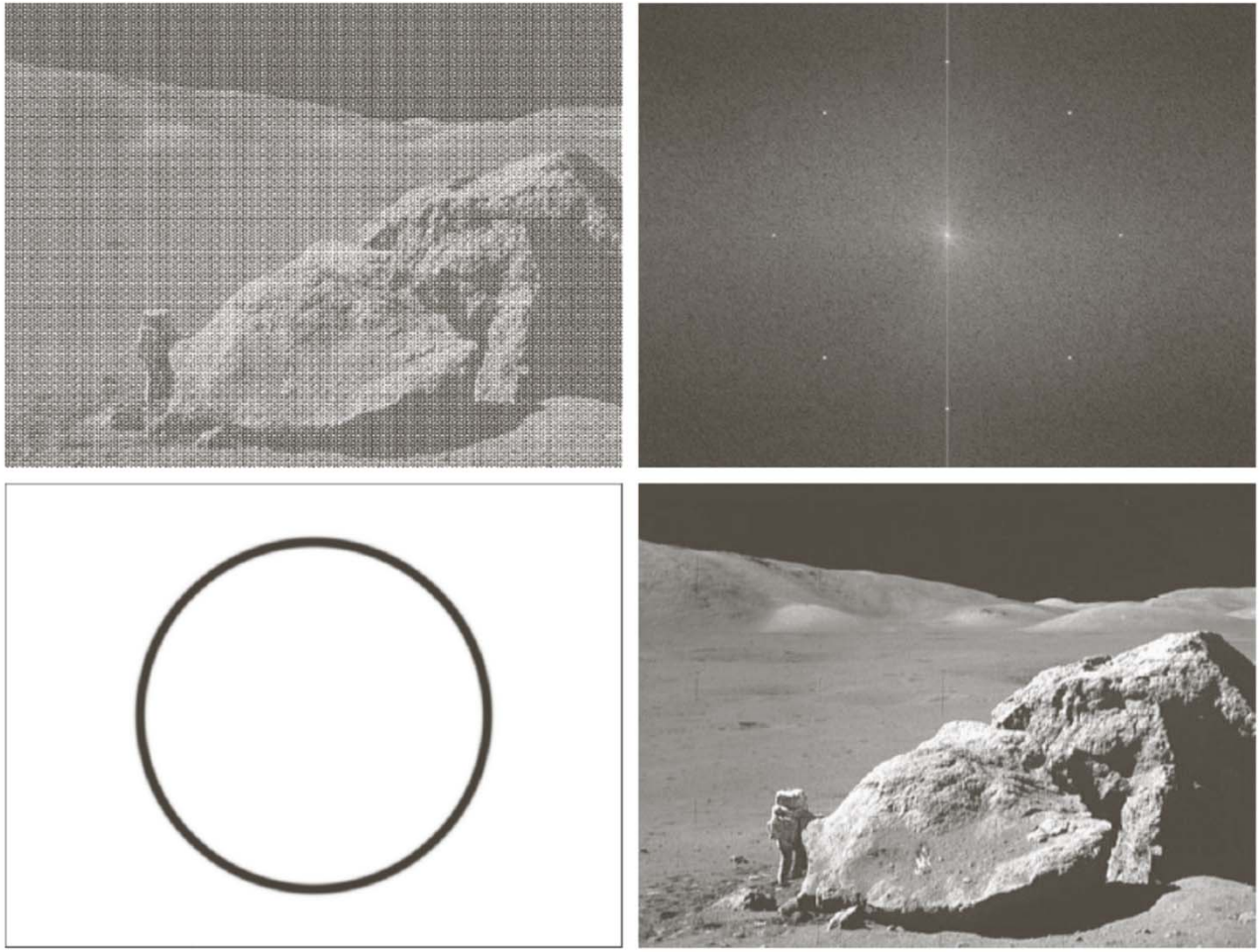
- Gaussian

$$H_{bsG}(u,v) = 1 - e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)W}\right]^2}$$

An example of bandstop filtering

a b
c d

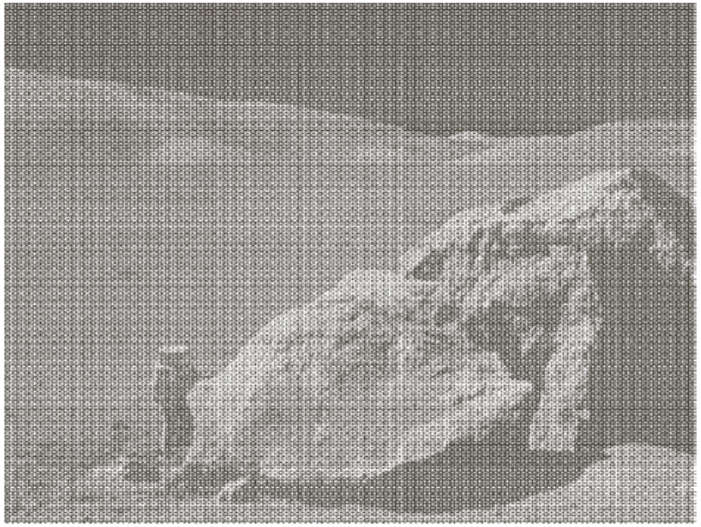
FIGURE 5.16 (a) Image corrupted by sinusoidal noise. (b) Spectrum of (a). (c) Butterworth bandreject filter (white represents 1). (d) Result of filtering. (Original image courtesy of NASA.)



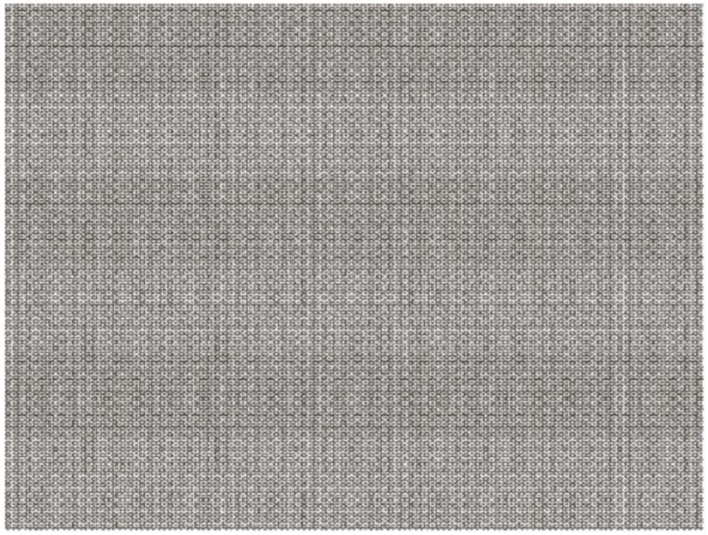
Bandpass filters

- Are defined by

$$H_{bp}(u,v) = 1 - H_{bs}(u,v)$$



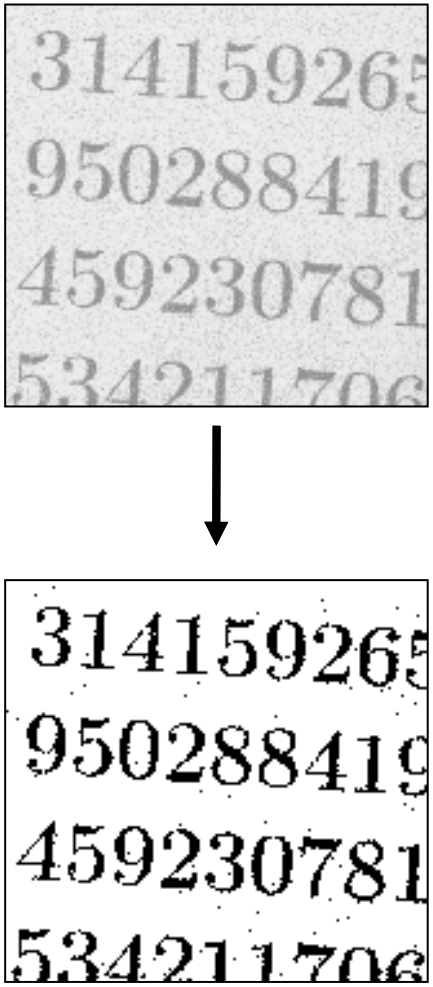
Original



Result after bandpass filtering

Segmentation and thresholding

- Segmentation
  - Function that labels each pixel in input image with a group label
  - Usually "foreground" and "background"
  - Each group shares some common properties
    - Similar color
    - Similar texture
    - Surrounded by the same edge
- Thresholding
  - One way of segmentation is by defining a threshold on pixel intensity



Segmentation and thresholding

Remember, regions that have semantic importance
do not always have any particular local visual distinction.

Segmentation and thresholding

- The only segmentation method taught in INF2310 was thresholding.
- Thresholding is a transformation of the input image *f* to an output (segmented) image *g* as follows:

$$g(i,j) = \begin{cases} 1, & f(i,j) \geq T \\ 0, & f(i,j) < T \end{cases}$$

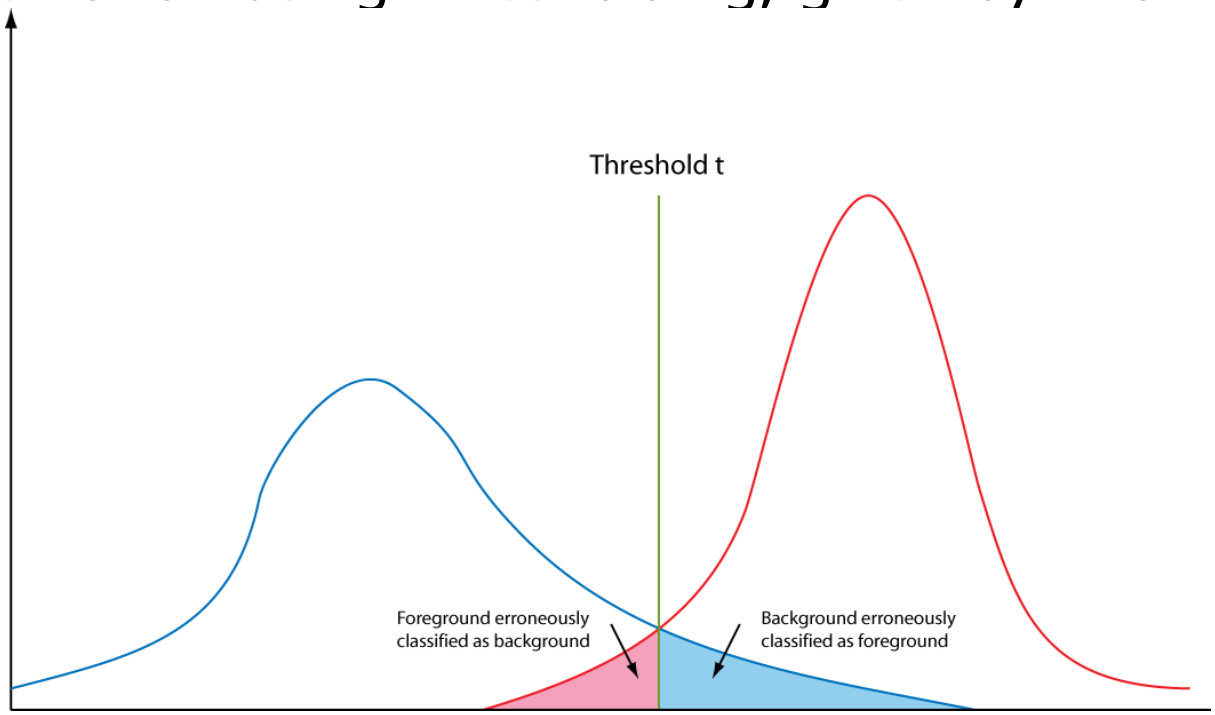
- Many variants of the basic definition …

Segmentation and thresholding

- This seemingly simple method must be used with care:
  - How do you select the threshold, manually or automatically?
  - Do you set a threshold that is global or local (on a sliding window or blockwise)?
  - Purely local method, no contextual considerations are taken
- Automatic threshold selection methods:
  - Otsu's method: a global "minimum-error" method
  - Ridler-Calvard's method: an iterative search for "optimal" threshold
- Local thresholding methods:
  - Local applications of Otsu and Ridler-Calvard
  - Niblack's method: $T(x,y) = \mu_w(x,y) + k\sigma_w(x,y)$
    - w=local window size, k=constant

Segmentation and thresholding

- Remember that you normally make an error performing a segmentation using thresholding, given by two integrals:



Segmentation and thresholding

- Assume that the histogram is the sum of two distributions $b(z)$ and $f(z)$, b and f are the normalized background and foreground distributions respectively, and z is the gray level.

- Let B and F be the prior probabilities for the background and foreground ($B+F=1$).

- The histogram can be written as a weighted sum of b & f

$$p(z)=Bb(z)+Ff(z).$$

Segmentation and thresholding

- The total thresholding error will be:

$$E(t) = F\int_{-\infty}^{t} f(z)dz + B\int_{t}^{\infty} b(z)dz$$

- Using Leibnitz's rule for derivation of integrals and by setting the derivative equal to zero you can find the optimal value for t:

$$\frac{E(t)}{dt} = 0 \Rightarrow Ff(T) = Bb(T)$$

Segmentation and thresholding

$$\frac{E(t)}{dt} = 0 \Rightarrow Ff(T) = Bb(T)$$

- This is a general solution !!!
- Does not depend on the type of distribution.

- In the case of *f* and *b* being Gaussian distributions, it is possible to solve the above equation explicitly.

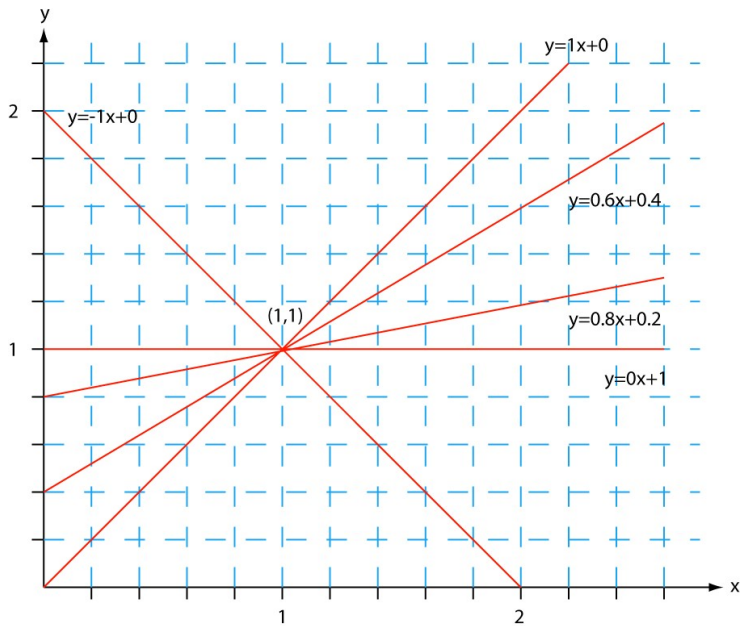
Segmentation and thresholding

- In INF2310 we introduced two methods
  (Ridler-Calvard and Otsu)
  for determining segmentation thresholds automatically.

- Region- and edge-based methods will be covered
  in detail in the IN5520 lectures.

IN5520 – Digital Image Analysis

HOUGH TRANSFORM



Fritz Albregtsen 02.09.2020

Plan for today

This lecture includes more details than

G&W, "DIP4E", Ch. 10 (pages 737-742):

Last section of 10.2:"Global Processing Using the Hough Transform"

- Introduction to Hough transform
- Using gradient information to detect lines
- Representing a line:
- [a,b]-representation & [ρ,θ]-representation
- Algorithms for detection of lines
- Detecting circles and ellipses
- Triangles and polygons

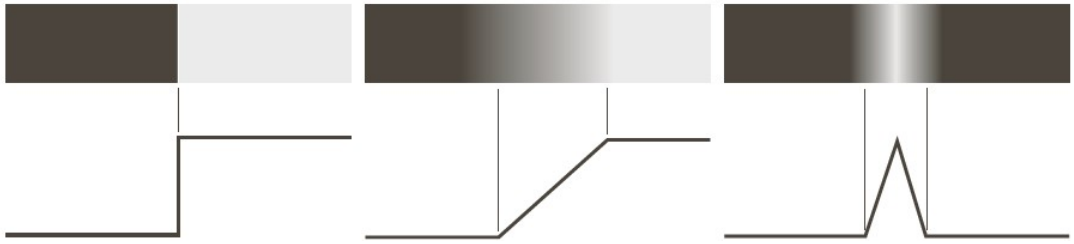
Introduction to HT

- The Hough transform (HT) can be used to detect
  - Lines
  - Circles and ellipses
  - Other parametric curves
- Introduced by Paul Hough (U.S. Patent 3,069,654, 1962)
- First used to find lines in images a decade later (Duda & Hart 1972).
- Our first goal is to find the location of line segments in images.
- This problem could be solved by e.g.
  - Morphology using a linear structuring element of a given direction
    - Then we would need to handle rotation, scaling, distortions etc
  - Fitting a line (regression)
    - Then we need to know which edge pixels to use as input, to handle one line segment at a time
- HT can give robust detection under noise and partial occlusion.

An image with linear structures

- Borders between the regions are straight line segments.
- These lines separate regions with different grey levels.
- Edge detection is often used as preprocessing to Hough transform.

Remember edge types?



a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

Step edge Ramp edge Roof edge

Hough-transform – the input

- The input image must be a thresholded edge image.
  - (See lecture notes from Introduction lecture, pp 28-36)

- The magnitude results computed by e.g. the Sobel operator can be thresholded and used as input.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Repetition - Basic edge detection

- A thresholded edge image is the starting point for HT.
- What does a Sobel filter produce?
- An approximation to the image gradient which is a vector quantity given by:

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Repetition – Edge magnitude

- The gradient is a measure of how the function f(x,y) changes as a function of changes in the arguments x and y.
- The gradient vector points in the direction of maximum change.
- The length of this vector indicates the size of the gradient:

$$|\nabla f| = \sqrt{g_x^2 + g_y^2}$$

G_x, G_y, and the gradient operator

- Horisontal edges:
  - Compute $g_x(x,y)=H_x*f(x,y)$
    - Convolve with the horisontal filter kernel H_x
- Vertical edges:
  - Compute $g_y(x,y)=H_y*f(x,y)$
    - Convolve with the vertical filter kernel H_y
- Compute the gradient parameters as:

$$G(x,y)=\sqrt{g_x^2(x,y)+g_y^2(x,y)}$$ Gradient magnitude

$$\theta(x,y)=\tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right)$$ Gradient-direction

- Note: from one input image, you produce four images:
  - $g_x(x,y)$, $g_y(x,y)$, $G(x,y)$, and $\theta(x,y)$.

More advanced edge detection

- The Canny edge detector (see G&W, "DIP4E", Ch 10, p 729-735)

  Lowpass filtering, & finding gradient magnitude and direction (Sobel).

  Thinning of gradient magnitude orthogonal to gradient direction.

  Hysteresis thresholding (two thresholds, T_h and T_l):

  a. Mark all pixels where $g(x,y) \geq T_h$

  b. For all pixels where $g(x,y) \in [T_l, T_h)$:

     - If (4 or 8)-neighbor of marked pixel, then mark this pixel too.

  c. Repeat from b until convergence.

- Laplacian of Gaussian (see G&W, "DIP4E", Ch 10, p 724-729)

  Find zero-crossings after convolving with LoG filter

$$-\nabla^2 G = \frac{1}{2\pi\sigma^4}\left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$

Input to Hough – thresholded edge image

Prior to applying Hough transform:

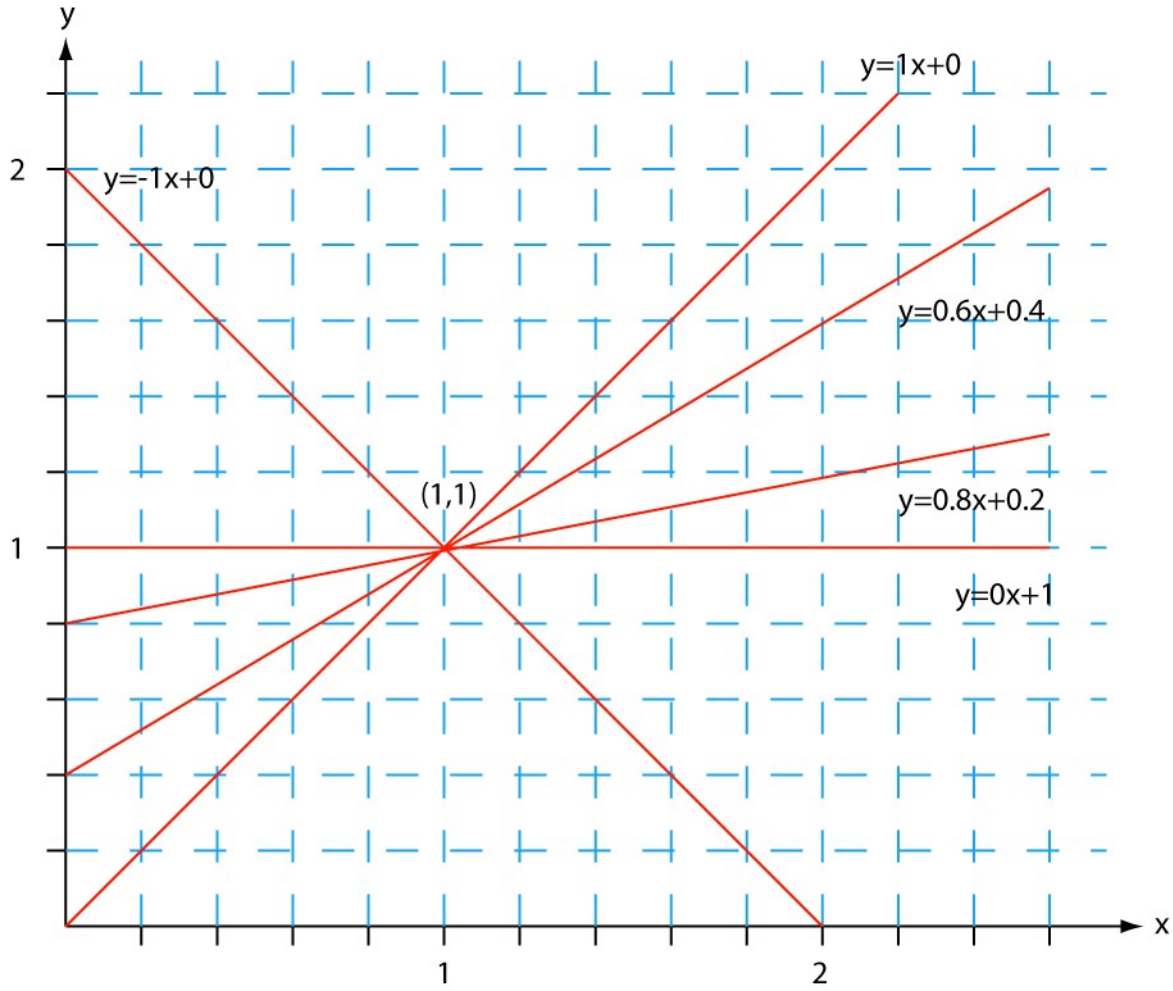
- Get edge magnitude from input image.
- As always with edge detection, simple lowpass filtering can be applied first.
- Thresholded gradient magnitude image.
- Note that such thresholding of gradient magnitude may be tricky!
  - Assumptions of e.g. Otsu may be wrong…
  - Gradient histogram is seldom bi-modal …

Hough-transform

- Assume that we have performed some edge detection, and a thresholding of the gradient magnitude image.
- We have *n* pixels that may partially describe the boundaries of some objects.
- We wish to find sets of pixels that make up straight line segments.
- Regard a point $(x_i; y_i)$

  and a straight line $y_i = ax_i + b$
  - Many lines may pass through the point (x_i, y_i).
  - Common to them is that they satisfy the equation above for varying values of the parameters (a,b).

Hough transform – basic idea



Hough transform – basic idea

- Given a line in the *(x,y)* – plane, with parameters *a* and *b*:

$$y = ax + b$$

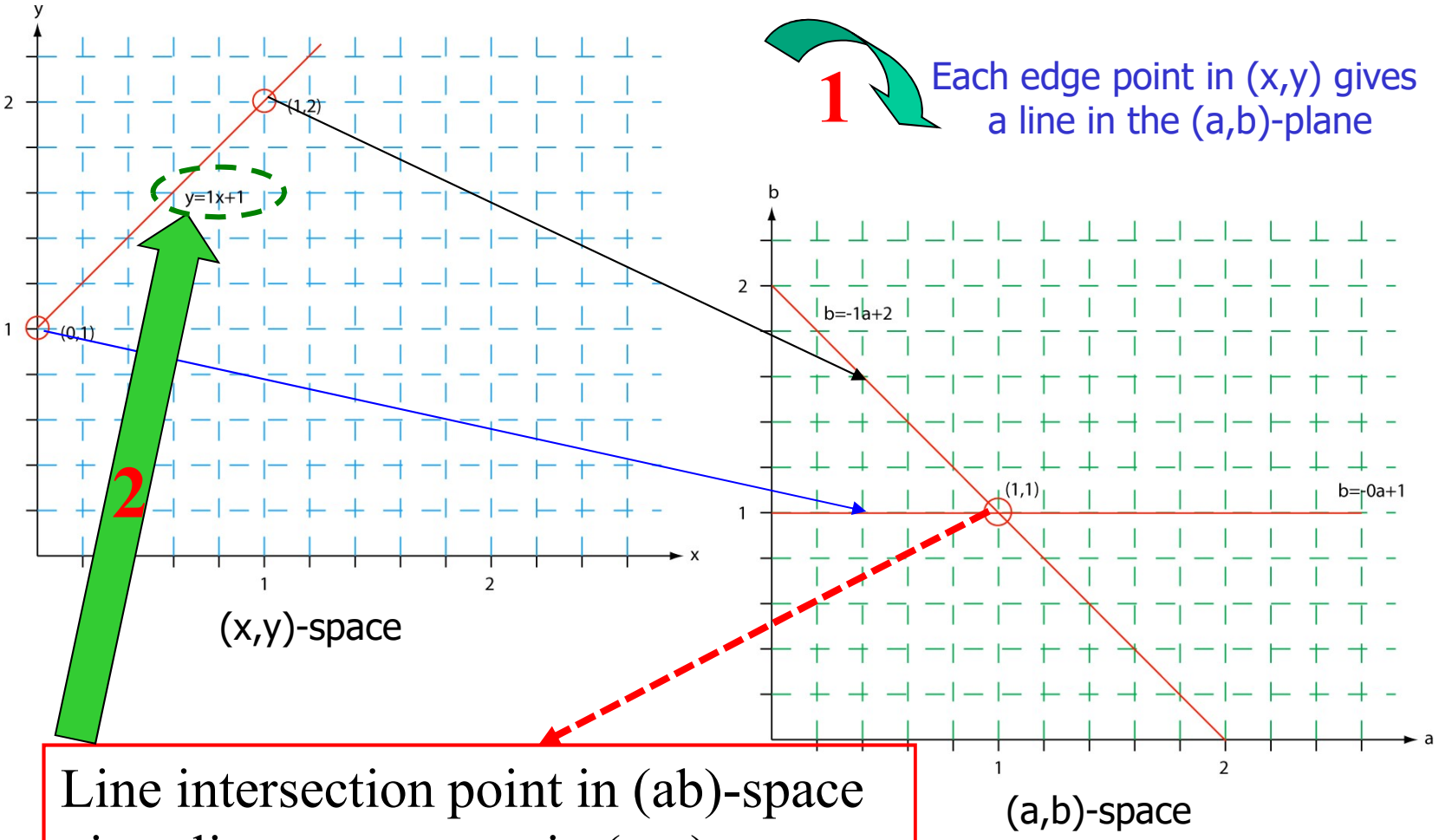
- This equation can obviously be rewritten as follows:

$$b = -xa + y$$

- We now consider (x ,y) as parameters and (a , b) as variables.

- This is a line in (a,b) - plane with parameters x and y.
  - So: a point (x_1,y_1) in xy - space gives a line in (a,b) space.
  - Another point (x_2,y_2) on the same line $y = ax + b$ in xy - space will give rise to another line in (a,b) - space.

Hough transform – basic idea



Line intersection point in (ab)-space gives line parameters in (x,y)-space

Hough transform – basic idea

- Two points (x,y) and (z,k) define a line in the (x,y) plane.
- These two points give two different lines in (a,b) space.

- In (a,b) space these lines will intersect in a point (a′,b′) where a′ is the slope and b′ the intercept of the line defined by (x,y) and (z,k) in (x,y) space.

- In fact, all points on the line defined by (x,y) and (z,k) in (x,y) space will parameterize lines that intersect in (a′,b′) in (a,b) space.

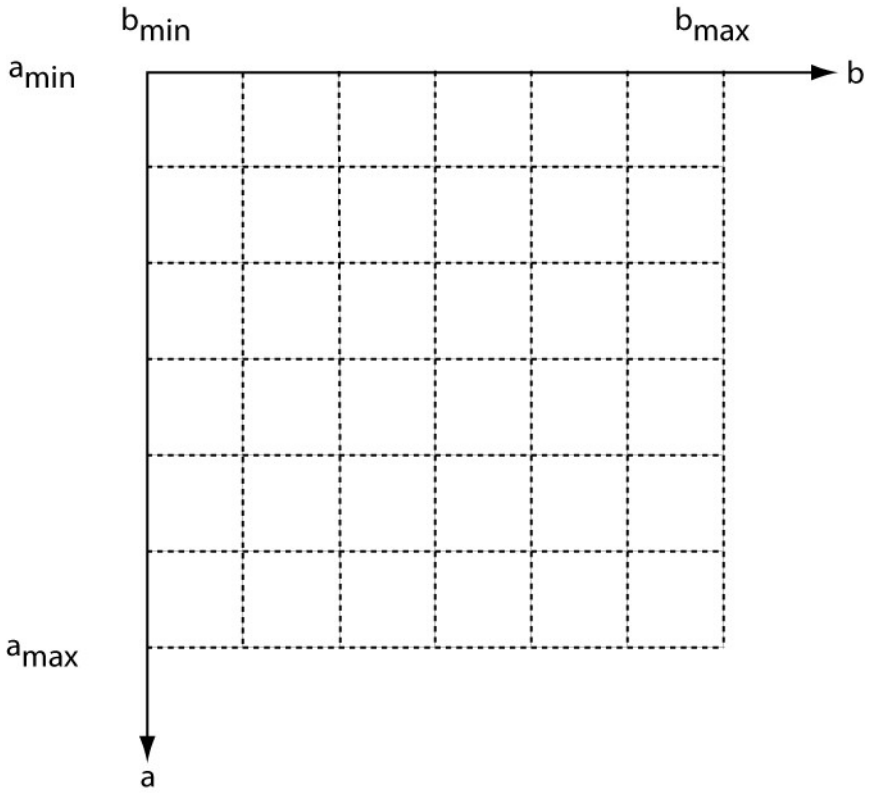
- Points that lie approximately on a line in the (x,y) space will form a "cluster of crossings" in the (a,b) space.

Hough transform – algorithm

- Quantize the parameter space (a,b), i.e., divide it into cells.
- This quantized space is often referred to as <u>the accumulator cells</u>.
- In the figure in the next slide a_{min} is the minimal value of a, etc.
- Count the number of times a line intersects a given cell.
  - For each point (x,y) with value 1 in the binary image, find the values of (a,b) in the range $[[a_{min},a_{max}],[b_{min},b_{max}]]$ defining the line corresponding to this point.
  - Increase the value of the accumulator for these [a',b'] point.
  - Then proceed with the next point in the image.

- Cells receiving more than a given number of "votes" are assumed to correspond to lines in (x,y) space.
  - Lines can be found as peaks in this accumulator space.

Hough transform - algorithm



Hough accumulator cells

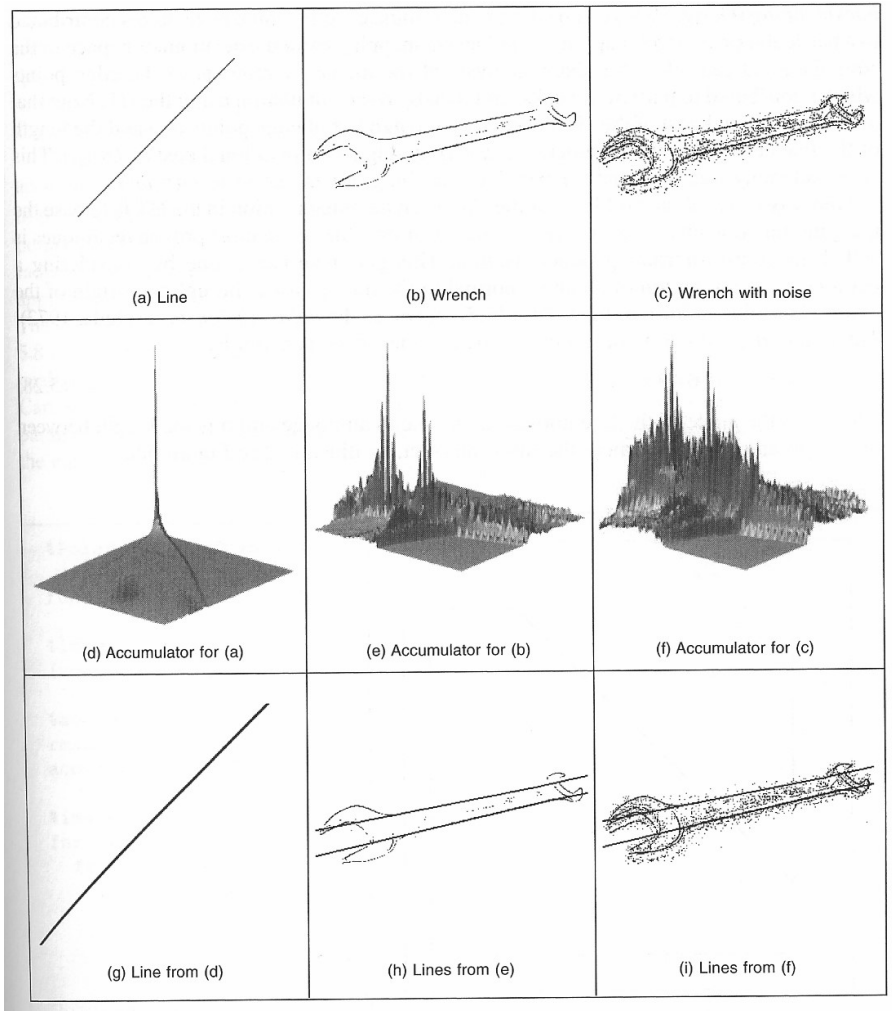
Example – images and accumulator space

Thresholded edge images

Visualizing the accumulator space

The height of the peak will be defined by the number of pixels in the line.

Thresholding the accumulator space and superimposing corresponding lines onto the edge image



Note how noise affects the accumulator. Even with noise, the largest peaks correspond to the major lines.

Polar representation of lines

- In practice, we do not use the equation

$$y = ax + b$$

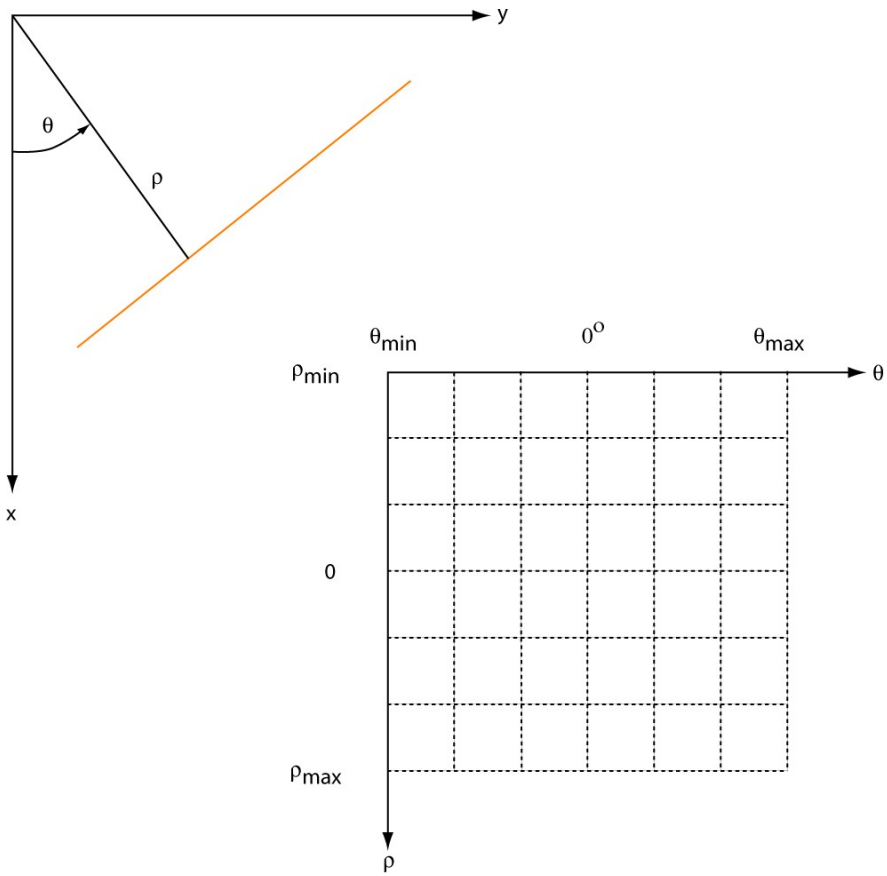
in order to represent lines (why?)

- Rather, we use the polar representation of lines:

$$x \cos\theta + y \sin\theta = \rho$$

  - where ρ is the length of the normal from origo to the line, and θ is the angle between this normal and the x-axis.
  - You should verify that this is true for any (x,y) on the line!

HT and polar representation of lines



Polar representation of lines

Hough transform and the polar representation

- The polar (also called normal) representation of straight lines is

$$x\cos\theta + y\sin\theta = \rho$$

- Each point (x_i, y_i) in the xy-plane gives a sinusoid in the $\rho\theta$-plane.

- M colinear points lying on the line

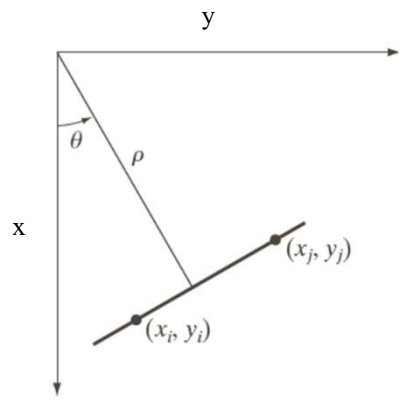
$$x_i\cos\theta + y_i\sin\theta = \rho$$

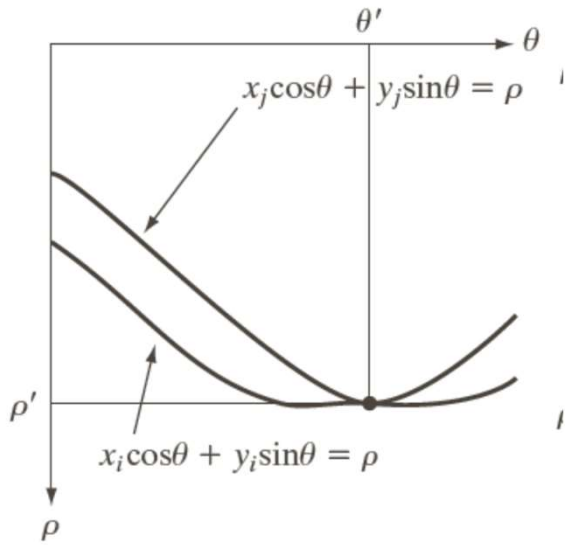
  will give M curves that intersect at (ρ_i, θ_j) in the parameter plane.

- Local maxima in parameter domain
  correspond to significant lines in the image domain.

Two points on a line …

- Each curve in the figure represents the familiy of lines that may pass through a particular point (x_i, y_i) in the xy-plane.
- The intersection point (ρ', θ') corresponds to the line that passes through two image points (x_i, y_i) and (x_j, y_j)
- A horizontal line will have $\theta=0$ and ρ equal to the intercept with the y-axis.
- A vertical line will have $\theta=90$ and ρ equal to the intercept with the x-axis.





HT with (ρ,θ)-representation of lines

- Partition the $\rho\theta$-plane into accumulator cells $A[\rho,\theta]$, $\rho \in [\rho_{min}, \rho_{max}]$; $\theta \in [\theta_{min}, \theta_{max}]$
- The range of θ is $\pm 90°$
  - Horizontal lines have $\theta=0°$, $\rho \geq 0$
  - Vertical lines have $\theta=90°$, $\rho \geq 0$
- The range of ρ is $\pm$ sqrt(M^2+N^2)
  if the image is of size MxN
- The discretization of θ and ρ must give acceptable precision and size of the parameter space.
  - An alternative could be the Muff transform (Exam 2015):
- **Wallace R.S.** "A Modified Hough Transform for Lines", *Proc. IEEE Computer Society Conf. on Comp. Vision and Patt. Recogni.* pp 665-667 (1985).

Algorithm continued

- The cell (i,j) corresponds to the square associated with parameter values (θ_j, ρ_i).
- Initialize all A(i,j) cells with value 0.

- For each foreground point (x_k,y_k) in thresholded edge image
  - For all possible θ-values
    - Solve for ρ using $\rho = x_k \cos \theta_j + y_k \sin \theta_j$
    - Round ρ to the closest cell value, ρ_q
    - Increment A(i,q) if the θ_j results in ρ_q

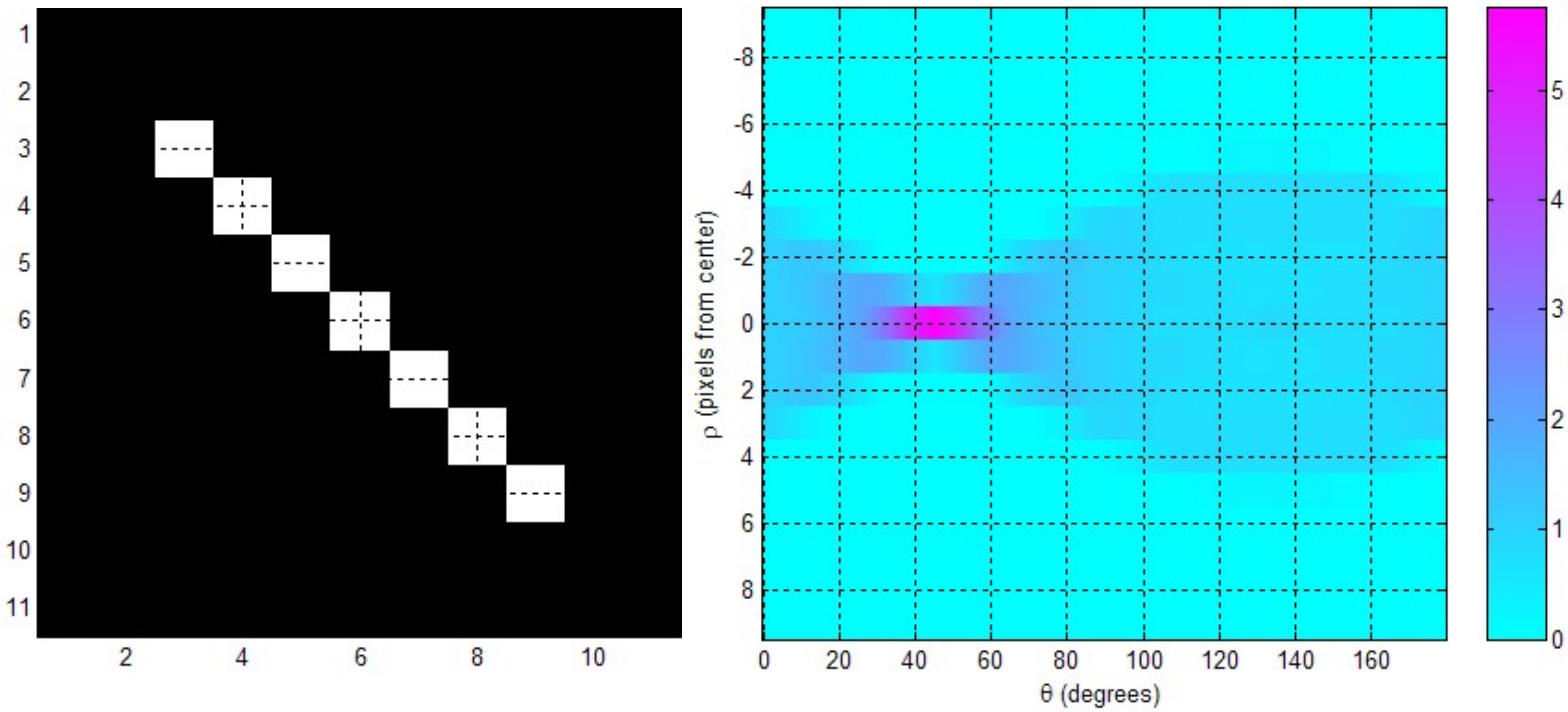
- A(i,j)=P means that P points in the xy-space lie on the line

$$\rho_j = x \cos \theta_j + y \sin \theta_j$$

- Find line candiates where A(i,j) peaks above a threshold.

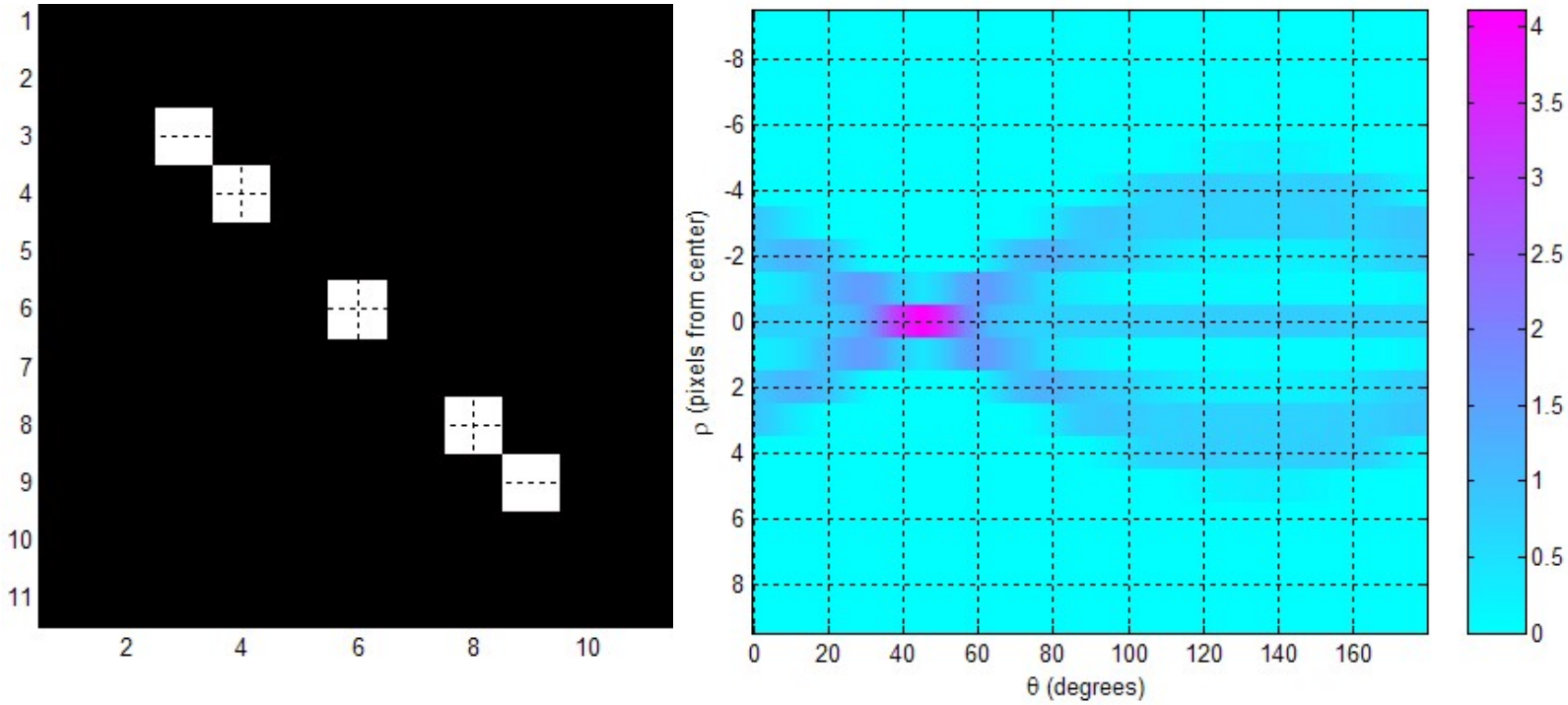
Hough transform – example 1

- Example 1: 11x11 image and its Hough transform:



Hough transform – example 2

- Example 2: 11x11 image and its Hough transform:



Hough transform – example 3

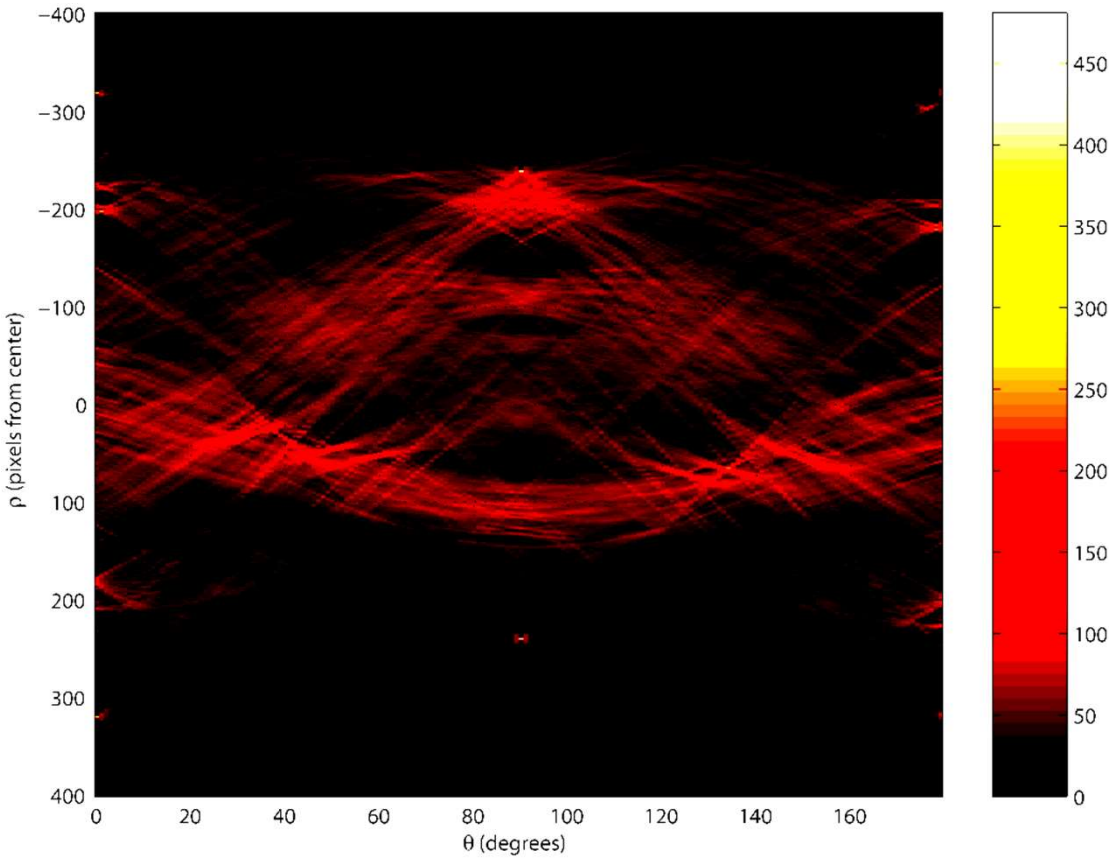
- Example 3: Natural scene & result of Sobel magnitude:

Hough transform – example 3

- Example 3: Natural scene and result of Sobel gradient magnitude followed by thresholding:

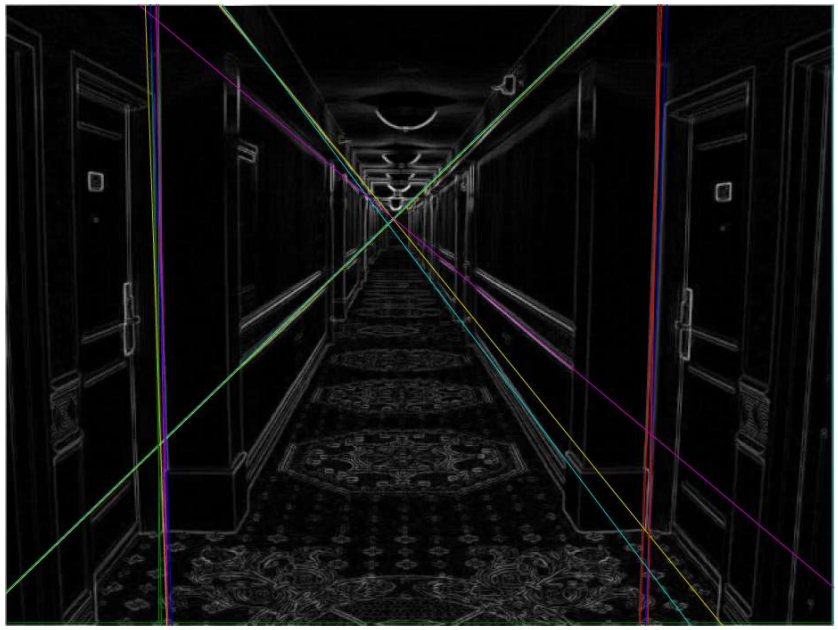
Hough transform – example 3

- Example 3: Accumulator matrix:



Hough transform – example 3

- Example 3: Original image (left) and 20 most prominent lines superimposed on gradient magnitude image (right):



HT - advantages

- Advantages:
  - Conceptually simple.
  - Easy implementation.
  - Handles missing and occluded data very gracefully.
  - Can be adapted to several types of shapes, not just lines.

HT - disadvantages

- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object.
  - Length and position of a line segment is not given directly.
  - Co-linear line segments are not separated directly.
    - Can be “fooled” by “apparent lines”.
    - Start and end of line segments must be identified by simple post-processing (linking: p35-36)

HT using the full gradient information

- Given a gradient magnitude image g(x,y) containing a line.
- Simple algorithm:

  for all $g(x_i,y_i)>T$ do

  for all θ do

  $\rho = x_i \cos\theta + y_i \sin\theta$

  find indexes (m,n) corresponding to (ρ, θ) and increment A(m,n);

- Better algorithm if we have both
  - The gradient magnitude g(x,y)
  - And the gradient components g_x and g_y
    - So we can compute gradient direction $\phi_g(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$
- The new algorithm:

  for all $g(x_i,y_i)>T$ do

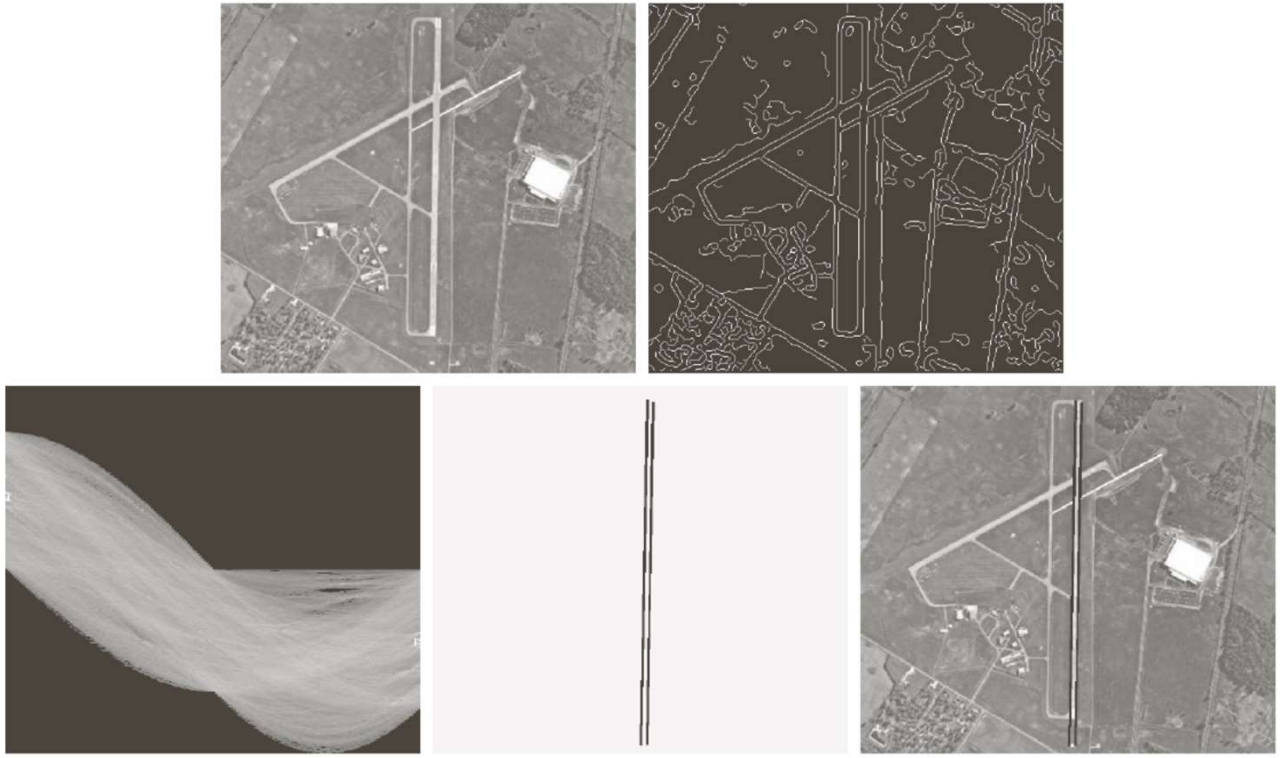
  $\rho = x_i \cos(\phi_g(x,y)) + y_i \sin(\phi_g(x,y))$

  find indexes (m, n) corresponding to $(\rho, \phi_g(x,y))$, increment A(m,n);

HT and line & edge linking

1. Obtain a thresholded edge image
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells (A) for high pixel concentrations (= length of line).
4. Examine the relationship (principally for continuity) between pixels corresponding to a chosen A-cell.
   - Continuity here normally means distance between disconnected pixels.
   - A gap in the line can be bridged if the length of the gap is less than a certain threshold.

Using edge linking



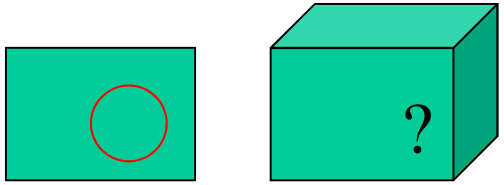
a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Hough transform for circles

- A circle in the xy-plane is given by

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$



- So we have a 3D parameter space. What size, what resolution?

- **A simple 3D accumulation procedure:**

  set all $A[x_c,y_c,r]=0$;

  for every (x,y) where g(x,y)>T

  for all x_c

  for all y_c

  $r = \mathrm{sqrt}((x-x_c)^2+(y-y_c)^2)$;

  $A[x_c,y_c,r] = A[x_c,y_c,r]+1$;

- **Better procedure(s)? 1 circle? Several circles?**
- **Applications …**

Hough transform for ellipses

- A general ellipse in the xy-plane has 5 parameters:
  - Position of center (x_c, y_c), semi-axes (a,b), and orientation (θ).
- Thus, we have a 5D parameter space.
- For large images and full parameter resolution,
  straight forward HT may easily overwhelm your computer!

- **Reducing accumulator dimensionality:**
  - Pick pixel pairs with opposite gradient directions
  - Accumulate 2D histogram of mid-points of such pairs
  - Peak histogram locations are center candidates.
  - Reduces HT-accumulator from 5D to 3D.

  - Disadvantage: Fails if there are lots of occlusions.

Hough transform for ellipses

- **Reducing accumulator dimensionality (2):**

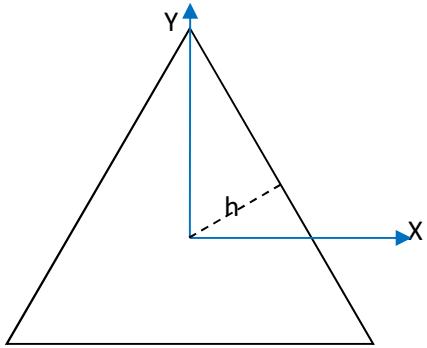
  1) Pick pixel pair (P,Q) with non-parallel tangents.
  - Tangents intersect at a point T.
  - Let M be mid-point of PQ.
  - Line TM goes through ellipse center.
  - Accumulate intersection of such lines to get (x_c, y_c).
  - Reduces HT-accumulator from 5D to 3D.

  2) Pick pixel pair (P,Q) with orthogonal tangents.
  - Tangents intersect at «director circle» of ellipse, giving (x_c, y_c).

- **Using other tricks …**

HT of rotated triangle - I

- An equilateral triangle of a given size is positioned so that the origin coincides with the center of mass of the triangle, as shown in the sketch.

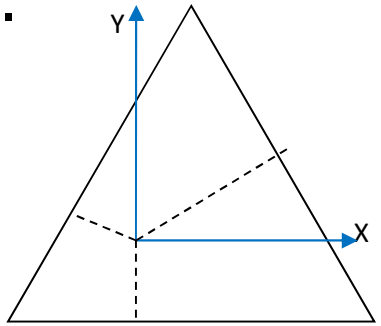


- *The three TH peaks will be π/3 apart.*
- *The peaks will have the same height.*
- *The normals onto the three sides are of the same length, => ρ=h*

- Q: What will happen in the Hough domain, if the triangle is rotated anti-clockwise around its centre? (let angular domain be [-π/2,π/2])
- *A: The three maxima in the (θ,ρ)-domain will slide in the positive θ-direction, keeping the 60 degree (π/3) distance between the maxima and a constant ρ-value, sliding out of the [-π/2,π/2]-domain at π/2 and reappearing at –π/2.*

HT of rotated triangle - II

- Given an equilateral triangle positioned so that the origin <u>does not</u> coincide with the centre of mass.

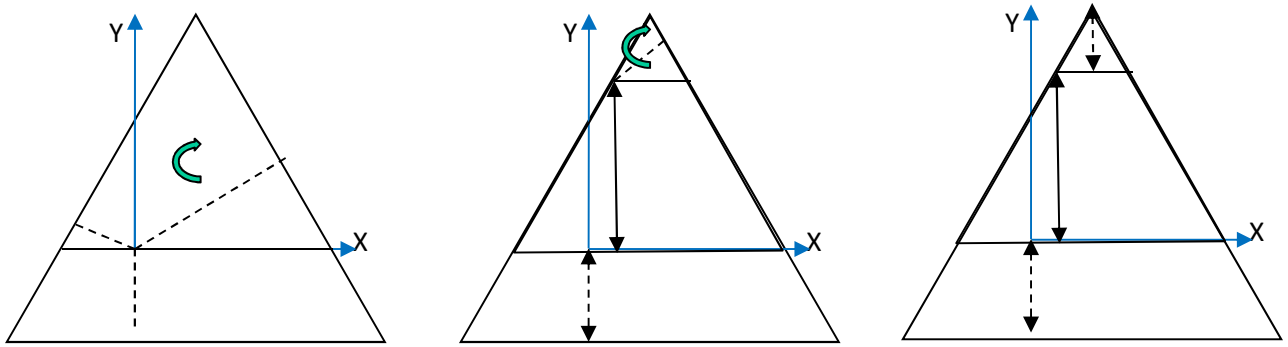


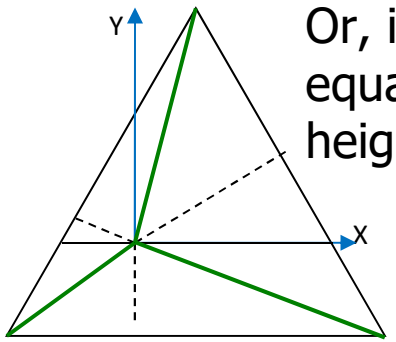
- *The three TH peaks will still be π/3 apart.*
- *The peaks will still have the same height.*
- *The normals onto the three sides are different.*

- If the triangle is now rotated anti-clockwise around the origin, the three (θ,ρ)-domain maxima will slide in the positive θ-direction, keeping the 60 degree (π/3) distance between the maxima. The three maxima will now have different, but constant ρ-values, sliding out of the [-π/2,π/2]-domain at π/2 and re-appearing at –π/2.
- *What is the sum of the three ρ values?*
- *Answer: <u>The sum is equal to the height of the equilateral triangle.</u>*
  (Viviani's Theorem, useful as smoothing constraint, valid for all regular polygons. (Regular = both equiangular and equilateral))

HT of rotated triangle - III

"Proof without words":





Or, if common side length = s, area of triangle is equal to sum of three triangles having base s and heights h_1, h_2, h_3.

Then $h_1+h_2+h_3$ is equal to height of equilateral triangle, independent of where inside the triangle (x,y) = (0,0) is.

- Valid for all *regular* polygons: The sum of the distance from an interior point to the N polygon sides is independent of the position of the point, and equal to N times the *apothem* (the distance from the center to any side).

HT of equiangular polygons - I

- Given an equiangular polygon positioned so that the origin <u>does not</u> coincide with the centre of mass.

- *The N HT peaks will be 180*(1-2/N)° apart.*
- *The normals onto the N sides are in general of different length.*

=>The peaks will in general not have the same height.

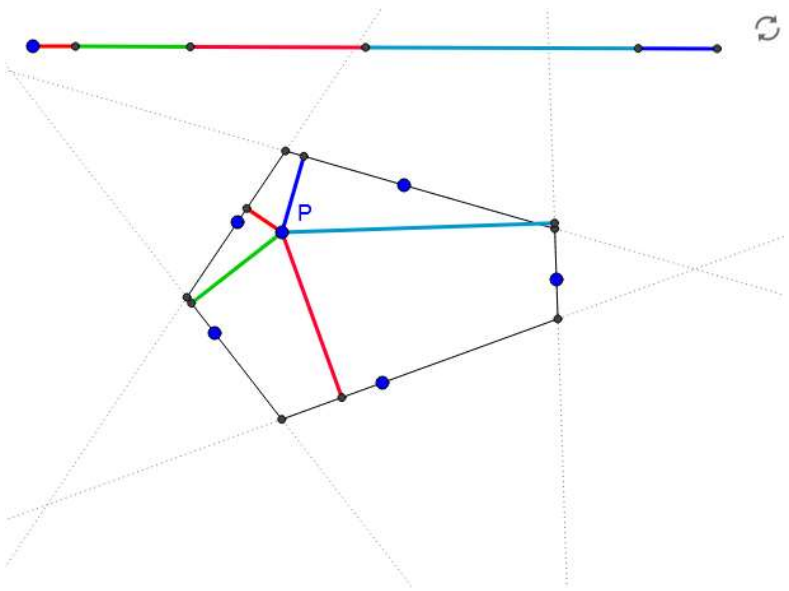
- *Viviani's theorem also holds for equiangular polygons.*[1]

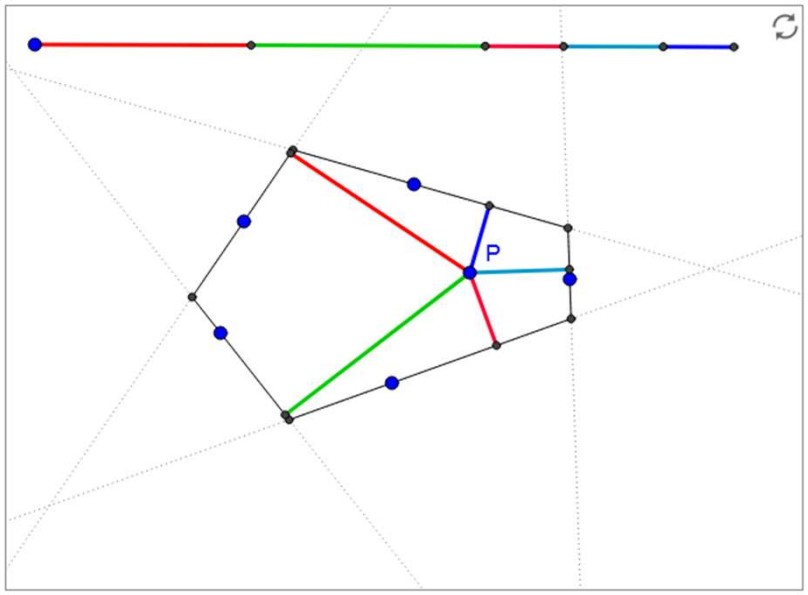
- *"The sum of distances from an interior point to the sides of an equiangular polygon does not depend on the location of the point, and is that polygon's invariant."*
- *Example on next page!*

1. E. Abboud: "On Viviani's Theorem and its Extensions", 2009.

HT of equiangular polygons - II

- Equiangular polygon: sum of pedal lengths is invariant to point position within polygon.





HT of any convex polygon

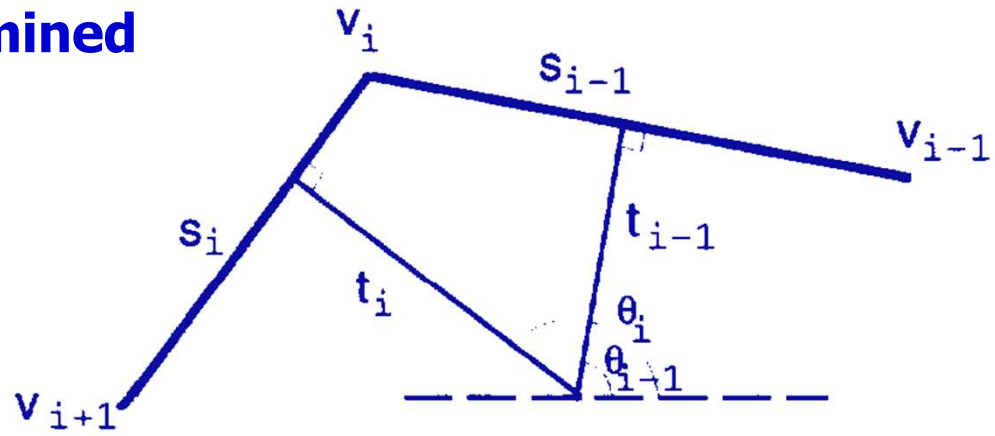
- **A convex polygon is determined by its HT:**

  For (x,y) on s_{i-1}

  $t_{i-1} = x\cos(\theta_{i-1}) + y\sin(\theta_{i-1})$

  For (x,y) on s_i

  $t_i = x\cos(\theta_i) + y\sin(\theta_i)$



- **Each vertex belongs to exactly two sides.**
- Given the HT peaks of these two sides, we can write

$$\begin{bmatrix} t_{i-1} \\ t_i \end{bmatrix} = \begin{bmatrix} \cos(\theta_{i-1}) & \sin(\theta_{i-1}) \\ \cos(\theta_i) & \sin(\theta_i) \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

- Which can be solved for the vertex $v_i = [x_i, y_i]^T$
- Given $0 \le \theta_1 < \theta_2 \ldots < \theta_N \le 2\pi$, we can solve for every vertex, and reconstruct the polygon from the *N* HT-peaks.

Hough transform – exercise 1

- Familiarize with Matlab function for line detection:
  - Functions hough(), houghpeaks(), and houghlines()

- Next exercise:
  - Test Hough transform for equal size circles on the coins image.

Hough transform – exercise 2

Next exercise: The randomized line-detection Hough transform.

- Simple idea (line case):
  - From the edge image ($g(x,y)>T$), pick two random points.
  - Find the ρ and θ corresponding to this set of points.
  - Increment the indicated (ρ,θ) cell.
  - Once a cell reaches a certain (low) count,
    assume that an edge is present in the image.
  - Verify this.
  - If truly present:
    - Store line parameters
    - Erase this line from the edge image
  - Continue until no more points or until the number of iterations between two detections is too high.
  - Orders of magnitude faster than the ordinary transform.

Learning goals - HT

- Understand the basic Hough Transform, and its limitations
- Understand that the normal representation is more general

- Be able to implement line detecting HT with accumulator matrix
  - Understand how this may be improved by gradient direction information.

- Be able to implement simple HT for circles of given size, in order to find position of circular objects in image.
  - Understand how this may be improved.

- Understand simple HT to detect ellipses
  - Understand how this may be improved.

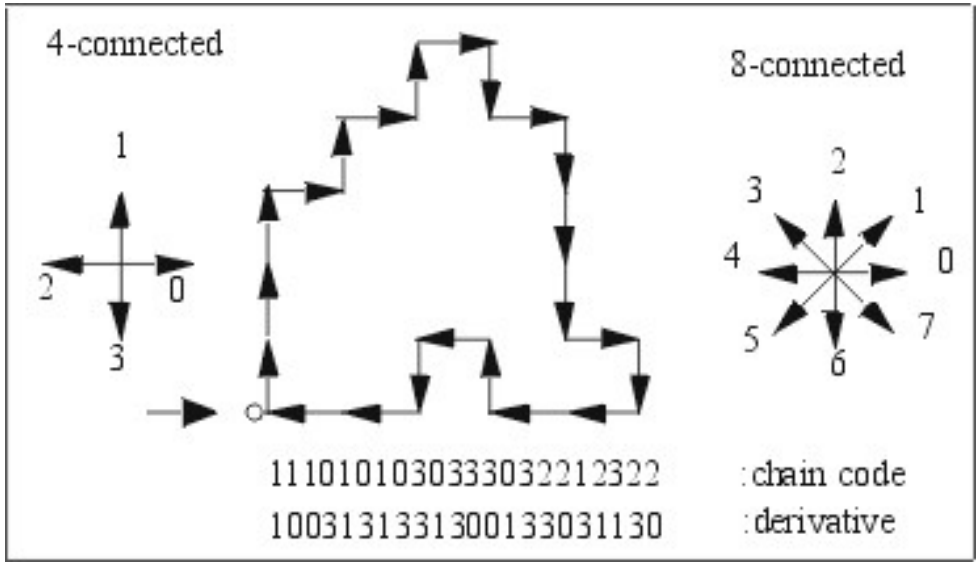
- Understand the basic random Hough transform.

- Do the exercises!

IN 5520 – Digital Image Analysis

OBJECT REPRESENTATION

FROM PIXELS TO REGIONS



Fritz Albregtsen 16.09.2020

Today

G&W, "DIP4E", Ch. 11 + some from Ch. 8

- Curriculum includes these lecture notes.

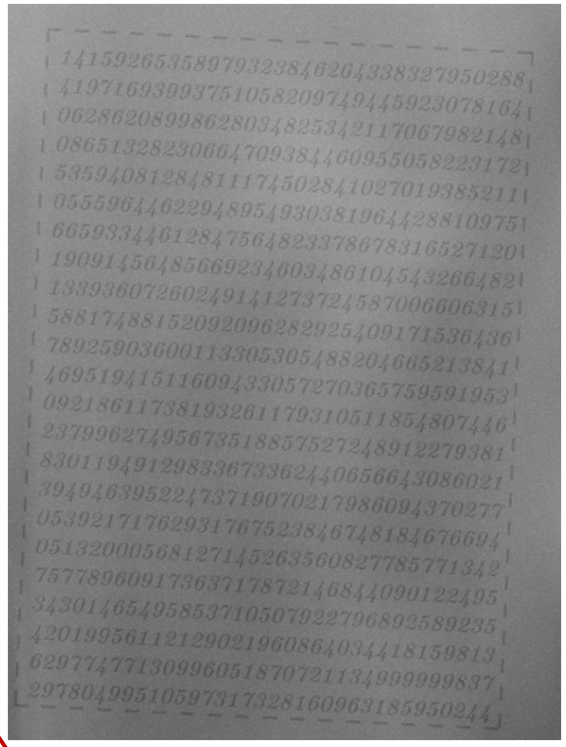
We cover the following:	DIP3E	DIP4E
Boundary following/tracing (self study)	*11.1.1*	*11.2 pp 814-816*
Chain Codes	11.1.2	11.2 pp 816-18
Polygonal Approximations using MPP,(omitted)	*11.1.3*	
Recursive boundary splitting	11.1.4	see lecture notes
Sequential polygonization		see lecture notes
Boundary Signatures	11.1.5	11.2 pp 826-828
Boundary Segments (convex hull)	11.1.6	see lecture notes
Skeletons	11.1.7	11.2 pp 828-831
Fourier representation	11.2.3	11.2 pp 835-839
Run Length Encoding	8.2.7	8.8 pp 566-572
Bit plane representation (& Gray Code)	8.2.7	8.8 pp 575-576

An example – Creating a program that recognizes the numbers

- Goal: get the series of digits, e.g. 1415926535897……

Steps in the program:

1. Segment the image to find digit pixels.
2. Find angle of rotation and rotate back.
3. Create region objects – one object pr. digit or connected component.
4. Compute features describing the shape of the digits (next week).
5. Train a classifier on many objects of each digit.
6. Classify a new object as the one with the highest probability.



Focus of this lecture

Shape representations vs. descriptors

- After the segmentation of an image,
  its regions or edges are represented and described
  in a manner appropriate for further processing.

- **Shape representation:**
  methods to represent and store image objects
  - **Perimeter (and all the methods based on perimeter)**
  - **Interior (and all the methods …)**

- Shape descriptor (next lecture):
  recipe to extract feature characterizing object shape.
  - The resulting feature value should be useful for discrimination between different object types.

Assumptions for shape representation

- The image objects can be represented as:
- Whole regions
  - grey level or color image
  - compressed image
  - binary image
- Contours (region boundaries)
  - in cartesian coordinates
  - in polar coordinates
  - in some other coordinates
  - through a chain code / transform
  - as coefficients of some transform (e.g. Fourier)
  - through a run length code / transform

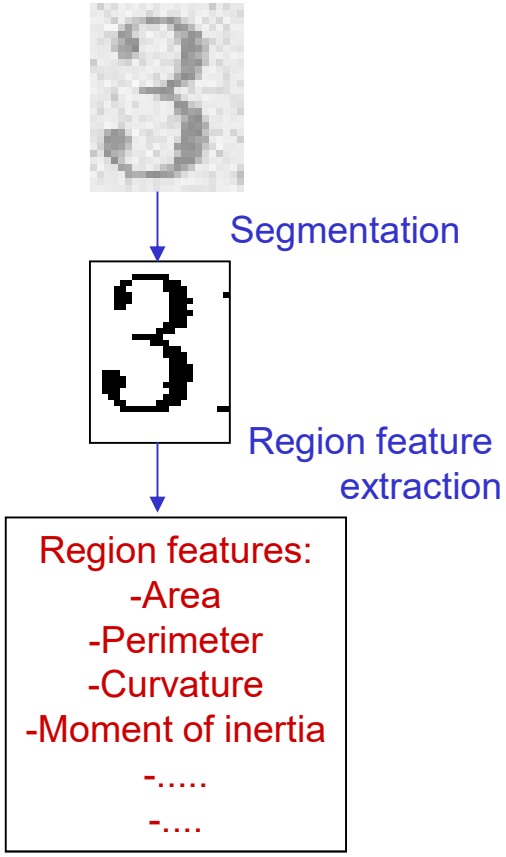
Considerations

- Input representation form, boundaries or whole regions?
- Object reconstruction ability?
- Incomplete shape recognition ability?
- Local/global description?
- Mathematical or heuristic techniques?
- Statistical or syntactic object description?
- Robustness of description to
  translation, rotation, and scale transformations?
- Shape description properties in different resolutions?
  - description changes discontinuously.
- Robustness against
  - image noise
  - geometric sampling (discretization)
  - intensity quantization

From pixels to features – text recognition

- Input to the classifier is normally a set of features derived from the image data, not the image data itself.

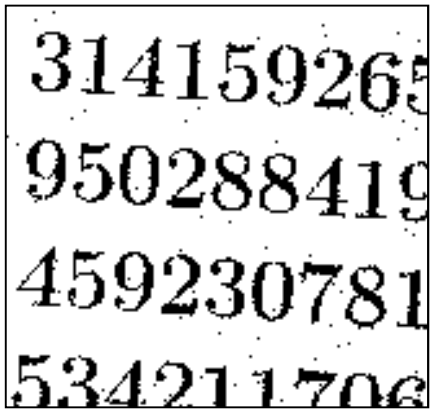
- Why can't we just use all the grey level pixels as they are for text recognition?
  - Objects corresponds to regions. We need the spatial relationship between the pixels.
  - For text recogntion: information is in shape, not in the grey levels.



Characteristics of a good segmentation

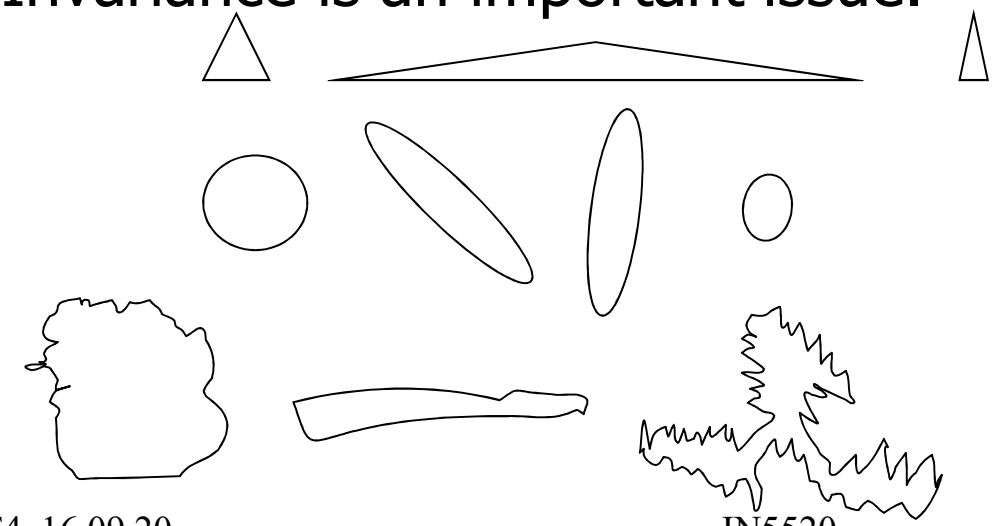
- Each object that is to be described has been identified during segmentation.
  - Ideally, one region in the segmented image should correspond to one object.
  - An object should not be fragmented into several non-connected regions.
  - Some small noise objects will often occurr, these can often be removed later.

  - This lecture and the next will describe how to find connected region objects and how to find scalars or vectors representing their shapes.



What is ”shape”?

- A numerical description of the spatial configuration.
- No generally accepted methodology of shape description.
- <u>High curvature points give essential information.</u>
- Many useful, application-dependent heuristics.
- Shape is often defined in a 2D image, but its usefulness in a 3D world depends on 3D -> 2D mapping.
- Invariance is an important issue.

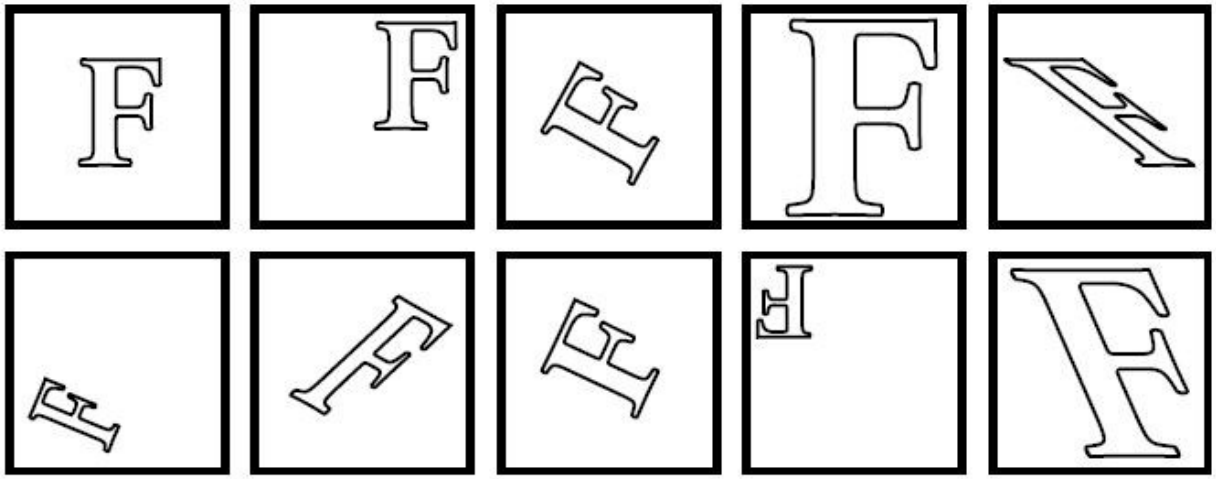


Three curvature points

Ellipses with different size and orientation

Objects with more irregular shape

Is invariance needed?



- Translation invariance
- Scale invariance
- Rotation invariance, but what about e.g., 6 and 9?
- Warp invariance
- You should check INF 2310 on geometrical operations

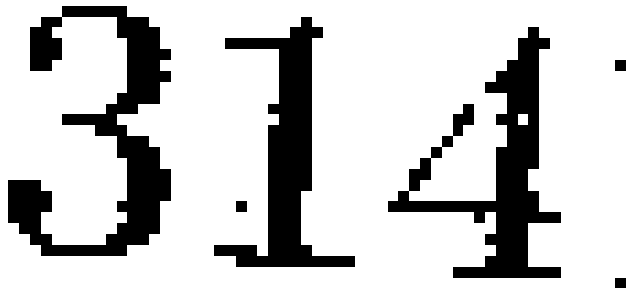
- Gray-level images: invariance to contrast and mean gray level

Shape invariants

- Shape descriptors depend on viewpoint,

  => object recognition may often be impossible if object or observer changes position.
- Shape description invariance is important
  - shape invariants represent properties which remain unchanged under an appropriate class of transforms.
- Stability of invariants is a crucial property which affects their applicability.
- The robustness of invariants to image noise and errors introduced by image sensors is of prime importance.

Creating region objects from segmented image

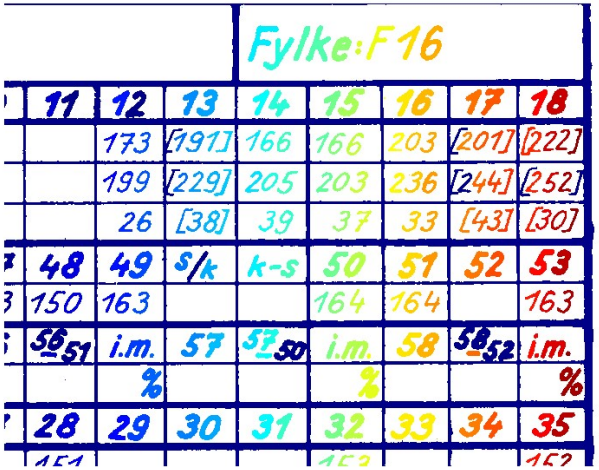


- Start in upper left corner.
- Find first object pixel.
- Traverse the border of this object (Ch. 11.2; p 814)
  - (recursive programming efficient)
- Continue with other pixels not yet visited
- Implementing this is time consuming
  - (but may be interesting)

Region identification

- After a complete segmentation, the regions must be labeled.
- Search image pixel by pixel, sequentially number each foreground pixel you find according to the labeling of its neighbors.
- The algorithm is basically the same in 4-connectivity and 8-connectivity.
- Result of region identification is a matrix same size as image, with integers representing region label.
- This description of regions will be the input to our shape descriptors

- Matlab function implementing this is *bwlabel*



All pixels
in a foreground object
(connected component)
get the same label value.

Pixels in the next object
get a different label etc.

Generating regions from segmented image

- Matlab: Functions bwlabel and regionprops are the starting points:

B1=bwlabel(B); B is a binary image, obtained by segmentation;

Useful for visualizing regions:

```
CC = bwconncomp(B1);
L = labelmatrix(CC);
RGBim = label2rgb(L); imshow(RGBim)
```

D=regionprops(B1,'area','boundingbox'); Compute area and boundingbox for all regions in the image.

aa=[D.area];

See help regionprops for a list of all descriptors.

Typical use: get a list of pixels for each region

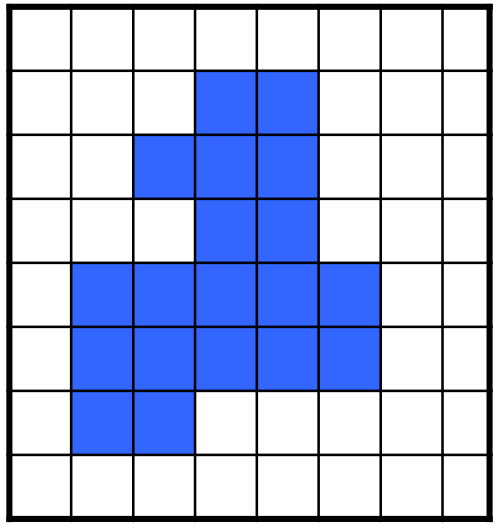
g=bwperim(B,8); Find perimeter pixels of all region in binary image B

Contour representation

- Goal of contour methods is to describe objects in images
- Hopefully, our contour detection method delivers a complete sequence of pixel coordinates in the image!
- The contour can be represented as:
  - Cartesian coordinates
  - Polar coordinates from a reference point (usually image origin)
  - Chain code and a starting point
    - Connectivity: 4- or 8-neighbors
    - Note: **chain code** is very sensitive to noise, image resolution and object rotation.
    - Not well suited for classification directly, but useful for computation of other features (next lecture).

Chains

- Chains represent objects within the image or borders between an object and the background

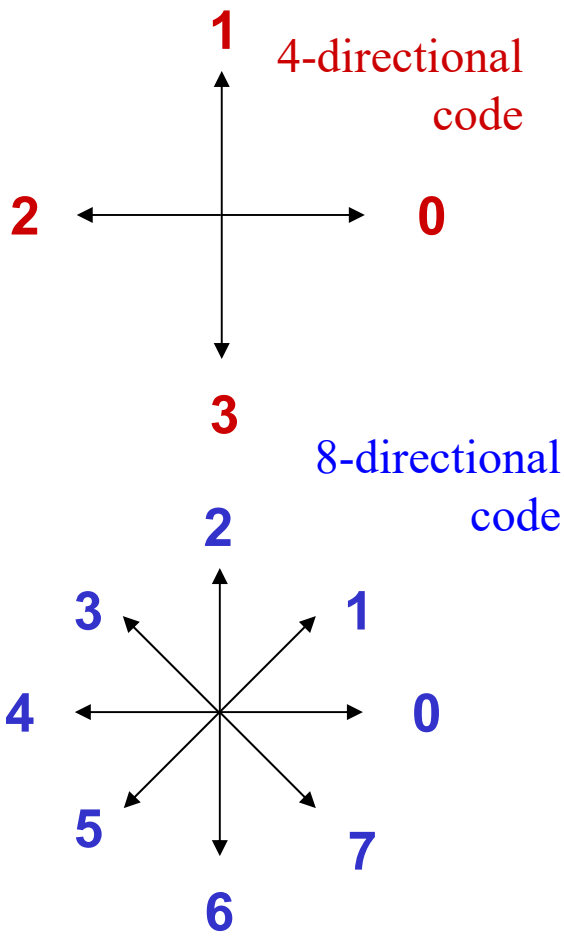


Object pixel

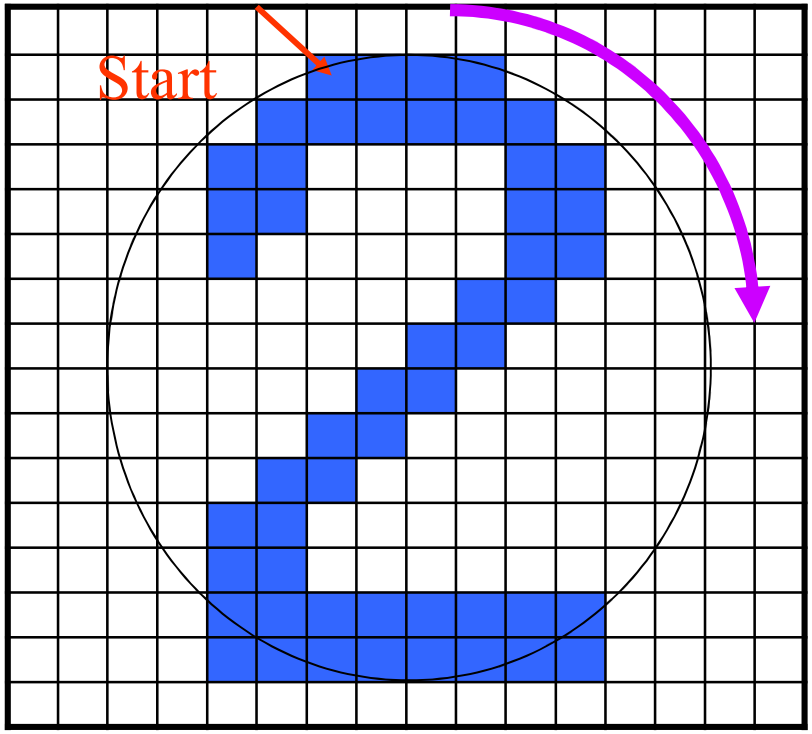
- How do we define border pixels?
  - 4-neighbors
  - 8-neighbors
- In which order do we check the neighbors?
- Chains represent the object pixel, but not their spatial relationship directly.

Chain codes

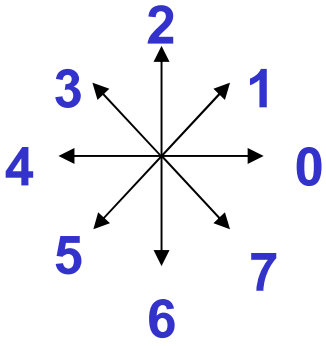
- Chain codes represent the boundary of a region.
- Chain codes are formed by following the boundary in a given direction (e.g. clockwise) with 4-neighbors or 8-neighbors.
- A code is associated with each direction.
- A code is based on a starting point, often the upper leftmost point of the object.



Absolute chain codes



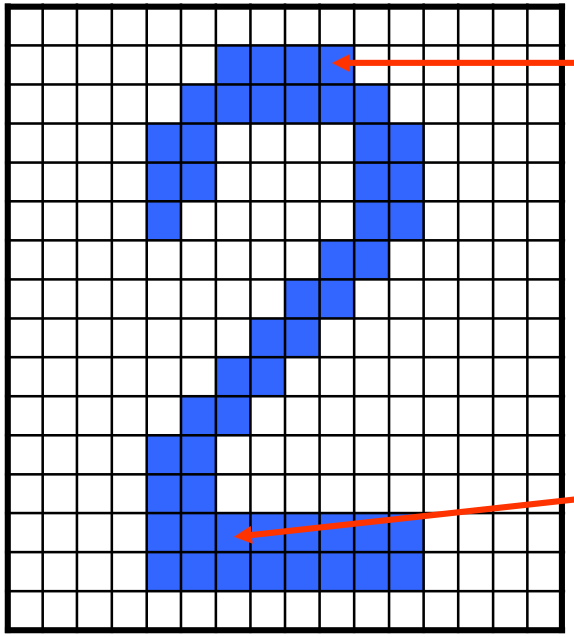
Search direction: look to the left first and check neighbors in clockwise direction



Chain code in clockwise direction:
0007766555555670000064444444222111112234445652211

Start point invariance

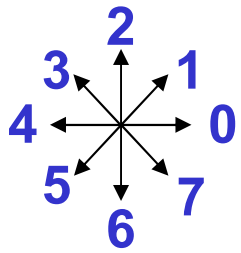
- The chain code depends on the starting point.
- It can be made invariant of start point by treating it as a circular/periodic sequence, and redefining the start point so that the resulting number is of «minimum magnitude».



Example:
77665555556700000644444444
22211111122344456522110 00

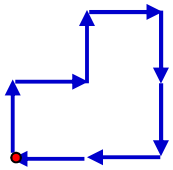
becomes
00000644444442221111112 23
44456522110007766555555 67
which starts here

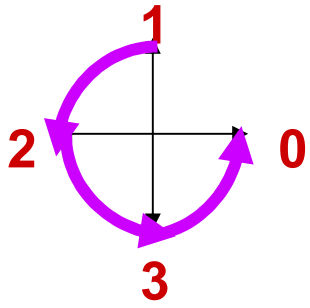
Sorting, not computing!



Rotation invariance

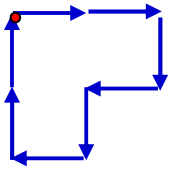
- We can also normalize for rotation by using the first difference of the chain code:
  (i.e., direction changes between code elements)





  - Example Chain code: 10103322 (in N_4)
  - First difference (counterclockwise): 33133030
  - To find first difference, look at the code and count counterclockwise directions.
  - Treating the curve as circular, add the 3 for the first point.
  - Minimum circular shift of first difference: **<u>03033133</u>**

- 00323211 is the same object, only rotated 180 degrees.
  First difference is 30331330, min.circ.shift is 03033133.



- This invariance is only valid if the boundary itself is invariant to rotation (and eventually scale).

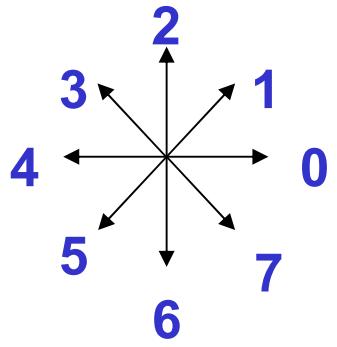
Relative chain code

- Here, directions are defined **in relation to** a moving perspective.

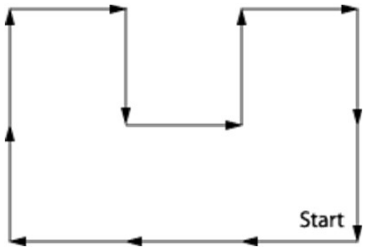
- Example: Orders given to a blind driver:

("F", "B", "L", "R").

- *The directional code representing any section of the contour is relative to the directional code of the preceding section.*



Note: rotate the code table so that 2 is forward from your position !!!

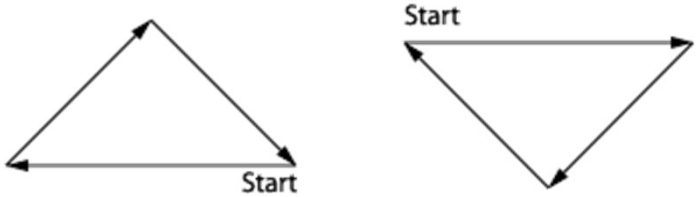


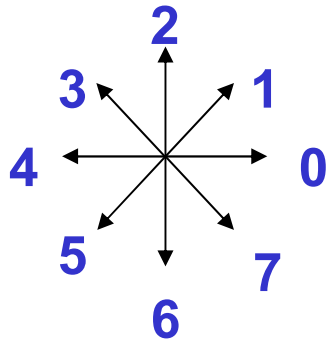
•*Why is the relative code:*
R,F,F,R,F,R,R,L,L,R,R,F ?

Relative chain code

- The absolute chain code for the triangles are

4,1,7 and 0, 5, 3

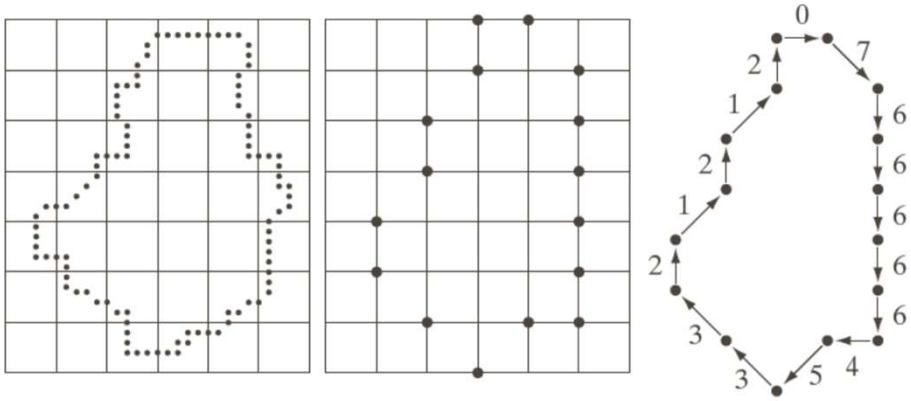




Note: rotate the code table so that 2 is forward from your position

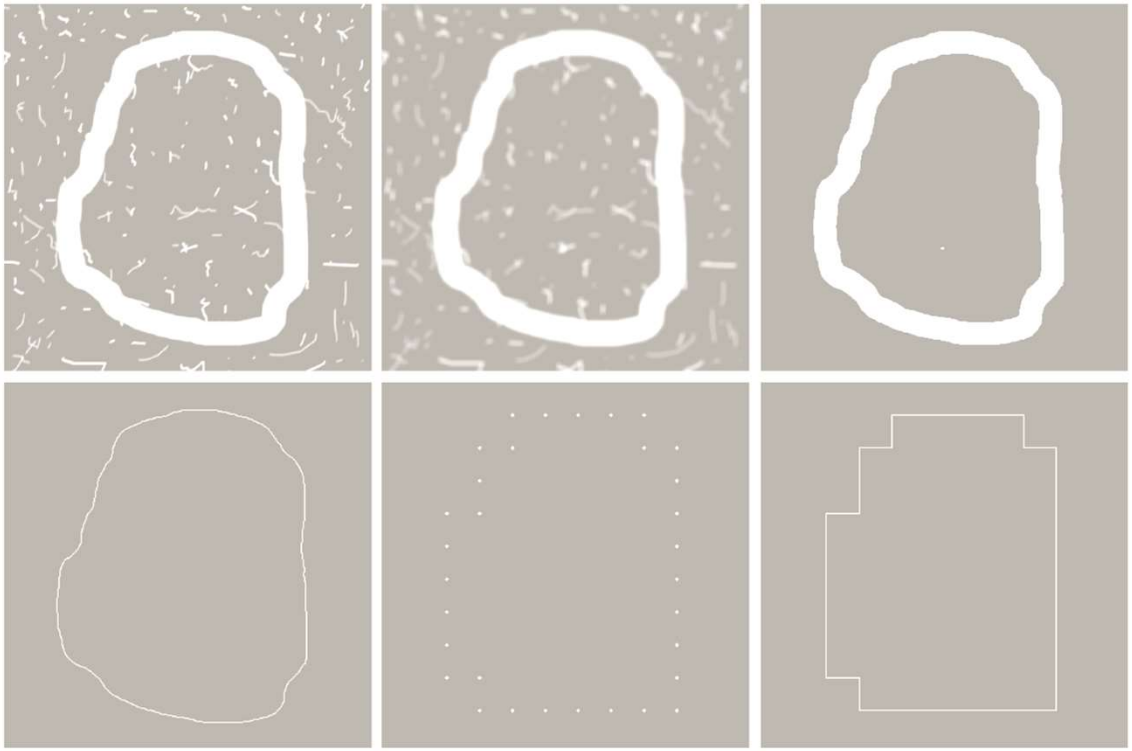
- The relative codes are 7, 7, 0. *(Remember "Forward" is 2)*
- So, relative code is invariant to rotation, as long as starting point remains fixed on object.
- Start-point invariance by "Minimum circular shift".
- Note: To find the first element, we look back one step, to the end of the contour.

Using all border point vs. resampling the border



- If chain codes are generated from every point on the border, the chain will be long and can be sensitive to small disturbances along the border.
- The border is often resampled to a larger grid before chain coding is done. The accuracy of the chain code depends on the grid size.

Chain code example with smoothing



a b c
d e f

FIGURE 11.5 (a) Noisy image. (b) Image smoothed with a 9×9 averaging mask. (c) Smoothed image, thresholded using Otsu's method. (d) Longest outer boundary of (c). (e) Subsampled boundary (the points are shown enlarged for clarity). (f) Connected points from (e).

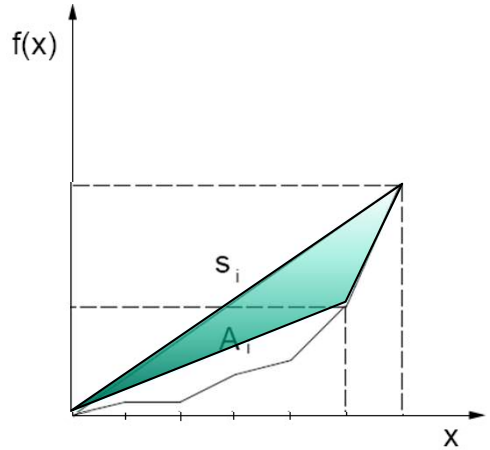
Polygonal boundary approximation: Recursive boundary splitting

- Draw straight line between contour points that are farthest apart. These two points are the initial breakpoints.

1. For each intermediate point:
2. Compute the point-to-line distance
3. Find the point with the greatest distance from the line.
4. If this distance is greater than given threshold, we have a new breakpoint.
5. The previous line segment is replaced by two, and 1-4 above is repeated for each of them.

- The procedure is repeated until all contour points are within the threshold distance from a corresponding line segment.
- The resulting ordered set of breakpoints is then the set of vertices of a polygon approximating the original contour
- This is probably the most frequently used polygonization.
- Since it is recursive, the procedure is fairly slow.

Sequential polygonization



- Start with any contour point as first "breakpoint".
- Step through ordered sequence of contour points.
- Using previous breakpoint as the current origin, area between contour and approximating line is:

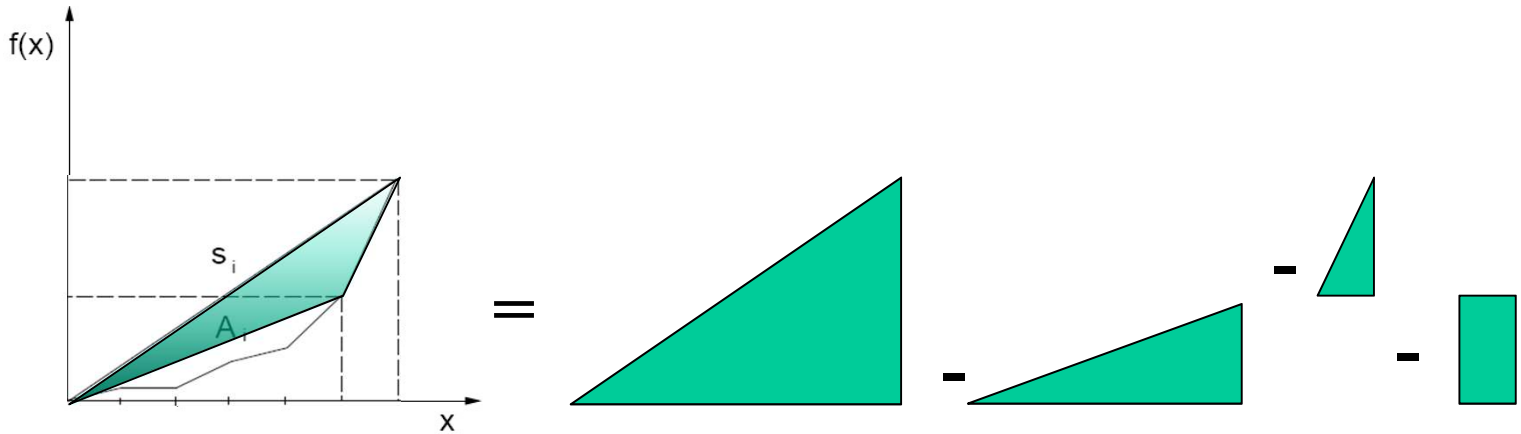
$$A_i = A_{i-1} + \frac{1}{2}\left(y_i\, x_{i-1} - x_i\, y_{i-1}\right), \quad s_i = \sqrt{x_i^2 + y_i^2}$$

- Let previous point be new breakpoint if
  - area deviation ***A*** per unit length ***s*** of approximating line segment exceeds a specified tolerance, ***T***.

- If $|A_i|/s_i < T$, **i** is incremented and (A_i, s_i) is recomputed.
- Otherwise, the previous point is stored as a new breakpoint, and the origin is moved to new breakpoint.

- This method is purely sequential and very fast.
- Reference: K. Wall and P.E. Danielsson, A Fast Sequential Method for Polygonal Approximation of Digital Curves, Computer Vision, Graphics, and Image Processing, vol. 28, 1984, pp. 220-227.

Sequential polygonization



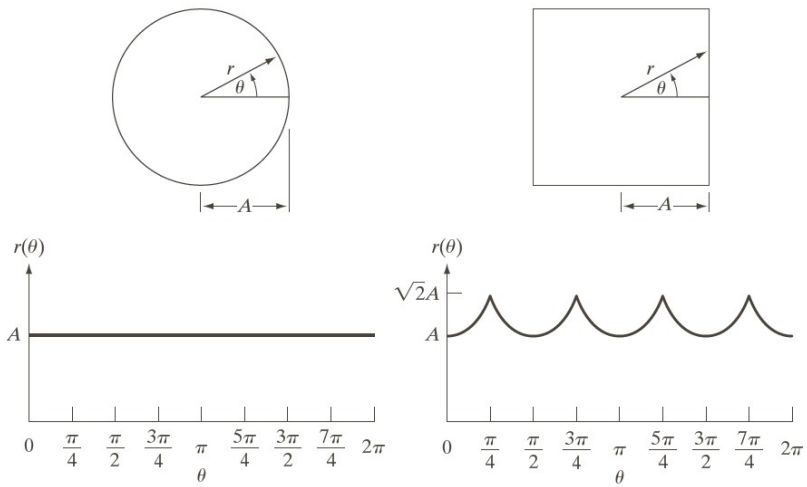
$$\Delta A_i = \boxed{\frac{1}{2}(x_i y_i)} - \boxed{\frac{1}{2}(x_{i-1} y_{i-1})} - \boxed{\frac{1}{2}(x_i - x_{i-1})(y_i - y_{i-1})} - \boxed{(x_i - x_{i-1}) y_{i-1}}$$

$$= \cancel{\frac{1}{2} x_i y_i} - \cancel{\frac{1}{2} x_{i-1} y_{i-1}} - \cancel{\frac{1}{2} x_i y_i} + \frac{1}{2} x_{i-1} y_i + \frac{1}{2} x_i y_{i-1} - \cancel{\frac{1}{2} x_{i-1} y_{i-1}} - x_i y_{i-1} + \cancel{x_{i-1} y_{i-1}}$$

$$= \frac{1}{2} x_{i-1} y_i - \frac{1}{2} x_i y_{i-1} = \underline{\underline{\frac{1}{2}(x_{i-1} y_i - x_i y_{i-1})}}$$

Signature representations

- A signature is a 1D functional representation of a 2D boundary.
- It can be represented in several ways.
- Simple choice: radius vs. angle:



a b

FIGURE 11.10 Distance-versus-angle signatures. In (a) $r(\theta)$ is constant. In (b), the signature consists of repetitions of the pattern $r(\theta) = A \sec\theta$ for $0 \le \theta \le \pi/4$ and $r(\theta) = A \csc\theta$ for $\pi/4 < \theta \le \pi/2$.

- Invariant to translation.
- Not invariant to starting point, rotation or scaling.

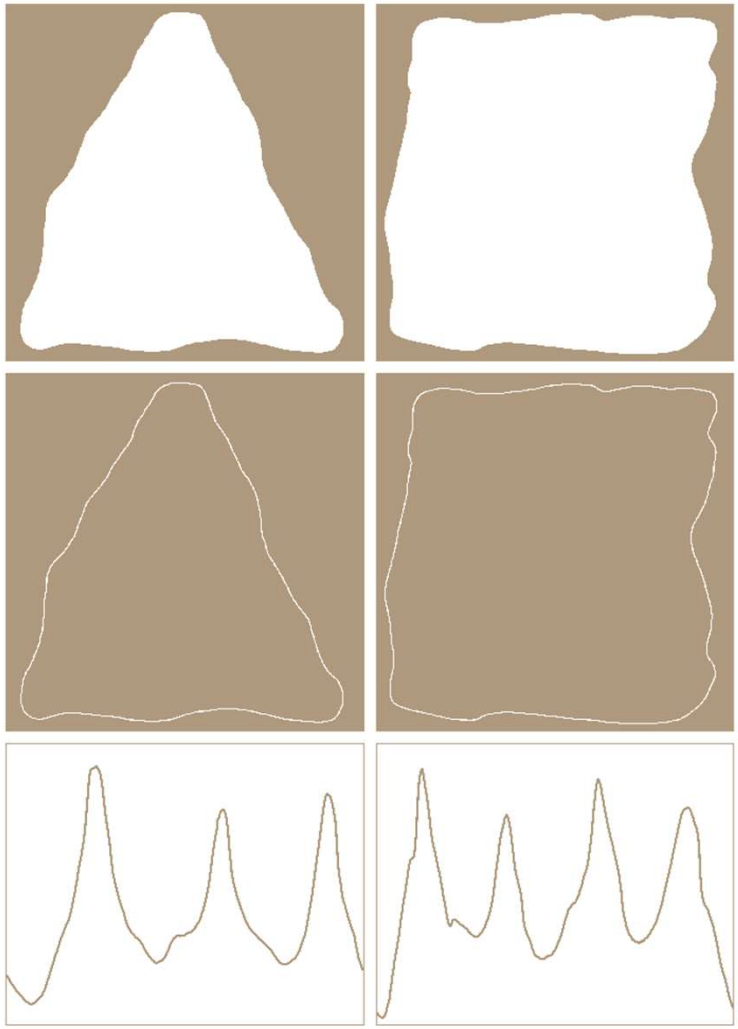
Normalization of (r,θ) signature

- Normalization for starting point and rotation:
  - Find unique point on the boundary according to some criteria:
    - Farthest away from the center (may not be unique)
    - Obtain the chain code and normalize as described for chain codes.

- Normalization for scale:
  - If the boundary is sampled at equal intervals in angle, changes in size only affect the amplitude (r).
    - The function can be scales so that it always spans the range [0,1].
      - This is not robust for extreme values of min/max.
    - Scaling by subtracting the mean and dividing by the standard deviation is more robust.

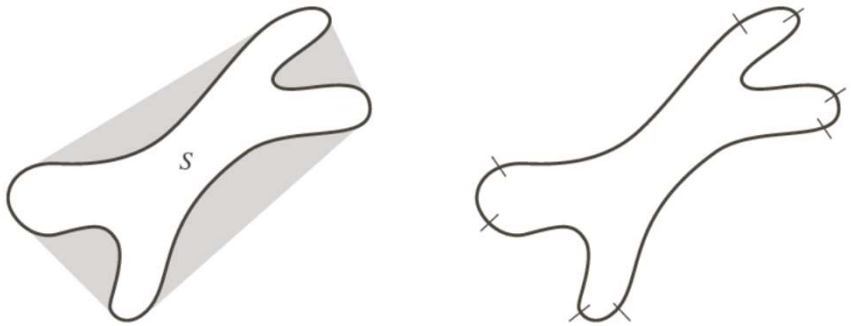
$$\frac{r-\mu}{\sigma}$$

Signature example



Boundary segments from convex hull

- The boundary can be decomposed into segments.
  - Useful to extract information from concave parts of the objects.
- **Convex hull** H of set S is the smallest convex set containing S.
- The set difference H-S is called the convex deficiency D.
- If we trace the boundary and identify the points where we go in and out of the convex deficiency, these points can represent important border points charaterizing the shape of the border.
- Border points are often noisy, and smoothing can be applied first.
  - Smooth the border by moving average of *k* boundary points.
  - Use polygonal approximation to boundary.
  - Simple algorithm to get convex hull from polygons.



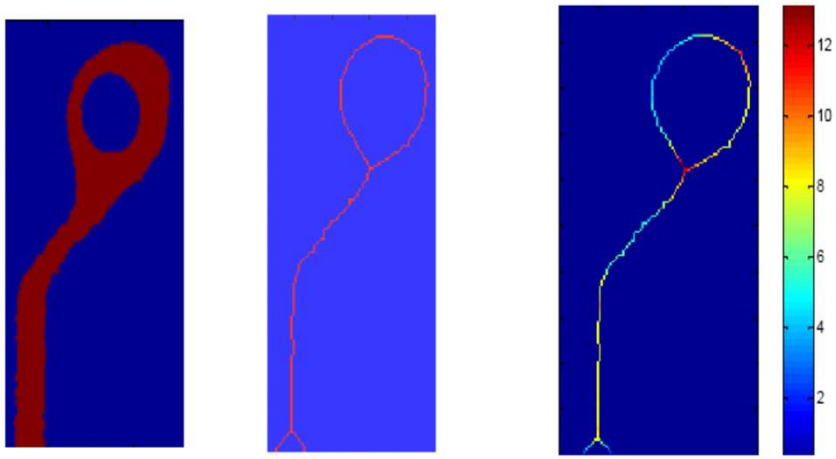
Descriptors extracted from Convex Hull

Useful features for shape characterization can be e.g.:

- Area of object and area of convex hull (CH)
  - CH ”solidity” aka ”convexity” = (object area)/(CH area)
    = The proportion of pixels in CH also in the object
  - Better than ”extent” = (object area)/(area of bounding box)
- Number of components of convex deficiency
  - Distribution of component areas
- Relative location of
  - points where we go in and out of the convex deficiency.
  - points of local maximal distance to CH.

Skeletons

- The skeleton of a region is defined by the medial axis transform:
  For a region R with border B, for every point p in R,
  find the closest neighbor in B.
- If p has more than one such neighbor, it belongs to the medial axis.
- The skeleton S(A) of an object is the axis of the object.
- The medial axis transform (MAT) gives the distance to the border,
  see color-coded left figure below.



Thinning algorithm to find the skeleton

- There are many algorithms in the literature, here is just one:
- Assume that region points have value 1 and background points 0.
- Define a contour point as a point that has value 1 (=> inside region) and at least one 8-neighbor with value 0 (=> outside region).
- Apply successive passes of two steps to contour points of region.
- Step 1: *Flag* contour point p_1 for deletion if:

  a) $2 \leq N(p_1) \leq 6$ (N=1 => end point)

  b) $T(p_1) = 1$ (T=1 => 1 pixel thick)

  c) $p_2 \cdot p_4 \cdot p_6 = 0$

  d) $p_4 \cdot p_6 \cdot p_8 = 0$

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Where $N(p_1)$ is the number of nonzero neighbors of p_1 ($=\Sigma p_i$)

$T(p_1)$ is the number of 0-1 transitions in the ordered sequence $p_2, p_3, \ldots, p_8, p_9, p_2$.

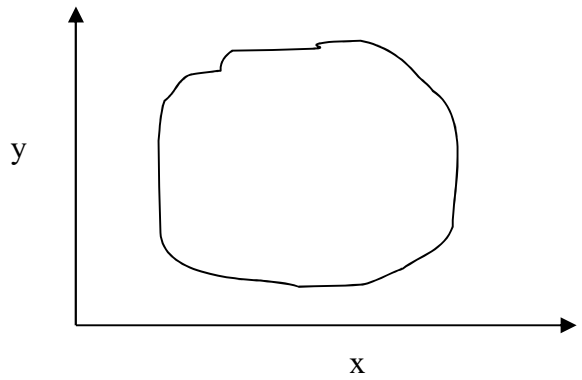
Thinning algorithm – cont.

- Step 2: change condition c) and d) to

  c') $p_2 \cdot p_4 \cdot p_8 = 0$ d') $p_2 \cdot p_6 \cdot p_8 = 0$
- Step 1 is applied to every border pixel in the region under consideration. If any of the conditions a)-d) are violated, the value of the point is unchanged.
  - If all conditions are satisfied, the point is flagged.
  - The point is not deleted until all points have been satisfied.
  - After completion of step 1, the flagged points are deleted.
- Step 2 is then applied in the same manner as step 1.
- Step1+Step2 defines one iteration of the algorithm.
- More iterations are done until convergence, i.e. until no further points are deleted.
- *Does not remove end points, break connectivity or cause excessive erosion of the region.*

Introduction to Fourier descriptors

- Suppose that we have an object *S*, and that we are able to find the length of its contour. The contour should be a closed curve.

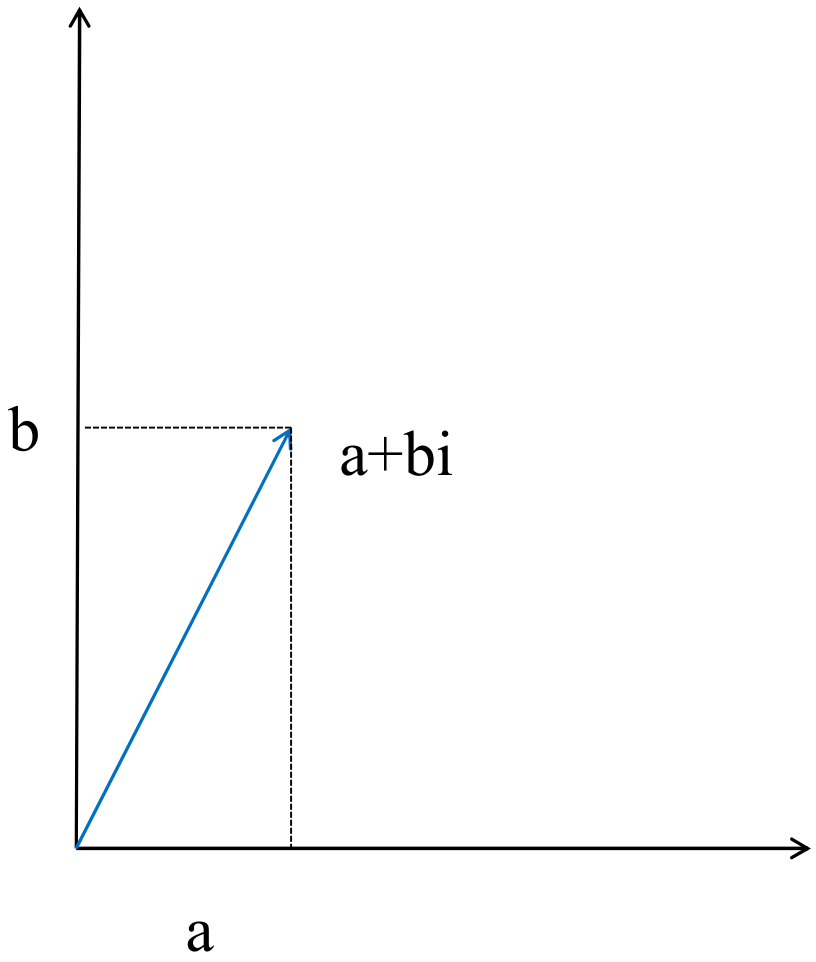


- We partition the contour into *M* segments of equal length, and thereby find *M* equidistant points along the contour of *S*.
- Traveling anti-clockwise along contour at constant speed, we can collect pairs of coordinates, x(k) and y(k). Any 1D signal representation can be used for these.
- If the speed is such that one circumnavigation of the object takes 2π, x(k) and y(x) will we periodic with period 2π.

GH Granlund, "Fourier preprocessing for hand print character recognition", IEEE Trans. Comput., C-21:195–201, 1972.

Reminder: complex numbers

- a+bi

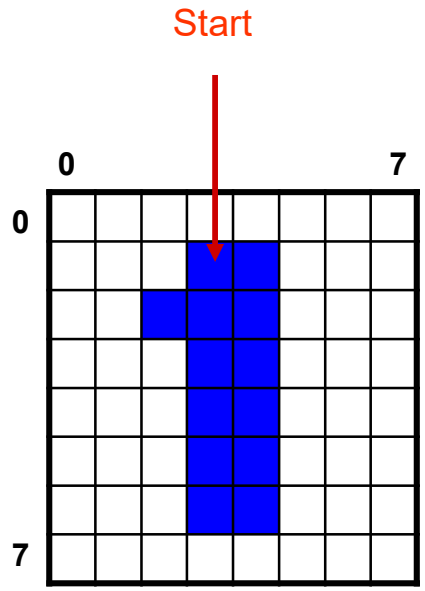


Parametric contour representation using 1D Fourier transform

- The coordinates (x,y) of these M points are then put into a complex vector s :

  $$s(k)=x(k)+iy(k), \quad k\in[0,M-1]$$

- We choose a direction (e.g. anti-clockwise)
- We view the x-axis as the real axis and the y-axis as the imaginary one for a sequence of complex numbers.
- The representation of the object contour is changed, but all the information is preserved.
- We have transformed the contour problem from 2D to 1D.



$$s(1) = 3 + 1i$$
$$s(2) = 2 + 2i$$
$$s(3) = 3 + 3i$$
$$s(4) = 3 + 4i$$
$$\vdots$$

x = 3 2 3 3 3 3 4 4 4 4 4 4 3

y = 1 2 3 4 5 6 6 5 4 3 2 1 1

Fourier-coefficients from *s(k)*

- We perform a 1D forward Fourier transform

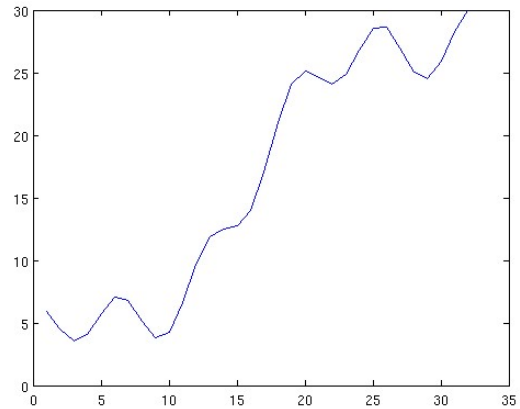
$$a(u) = \frac{1}{M}\sum_{k=0}^{M-1} s(k)\exp\left(\frac{-2\pi iuk}{M}\right) = \frac{1}{M}\sum_{k=0}^{M-1} s(k)\left(\cos\left(\frac{2\pi uk}{M}\right) - i\sin\left(\frac{2\pi uk}{M}\right)\right), \qquad u \in [0, M-1]$$

- Complex coefficients *a(u)* are the Fourier representation of boundary.
- *a(0)* contains the center of mass of the object.
- Exclude *a(0)* as a feature for object recognition.
- a*(1)*, a*(2),....,a(M-1)* will describe the object in increasing detail.
- These depend on rotation, scaling and starting point of the contour.

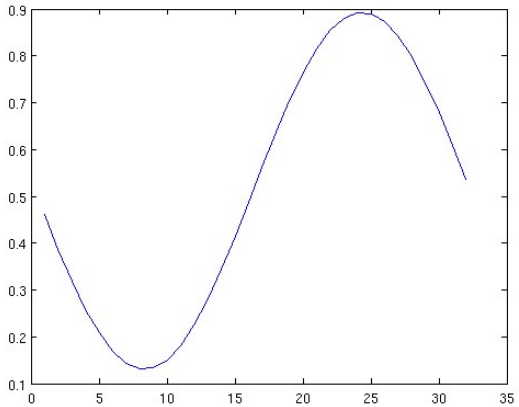
- For object recognition, use only the N first coefficients (a*(N), N<M*)
- This corresponds to setting a*(k)=0, k>N-1*

Approximating a curve with Fourier coefficients in 1D

- Do the Fourier transform of the signal of length M
- Keep only N (<M/2) first coefficients (set the others to 0 in amplitude).
- Compute the inverse 1D Fourier transform of the modified signal.
- Display the reconstructed M-point signal based on only N coefficients.

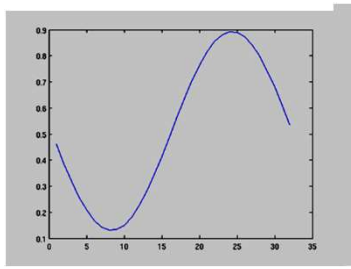


Original signal of length N=36

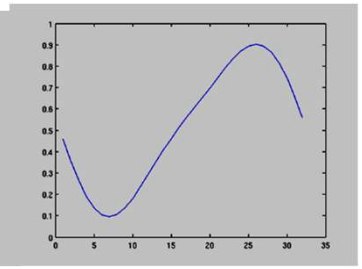


Reconstructed signal using only 2 Fourier coefficients

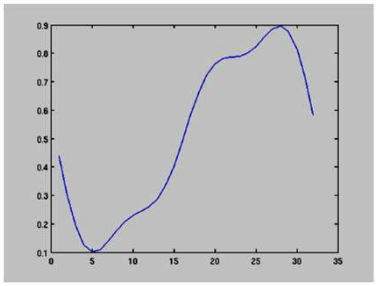
Approximating in increasing detail



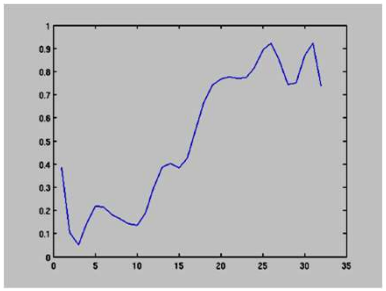
n=2



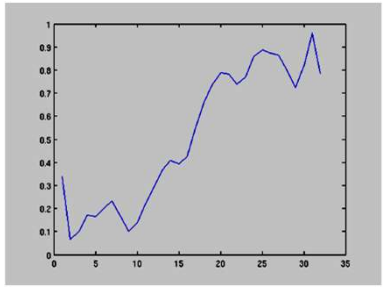
n=3



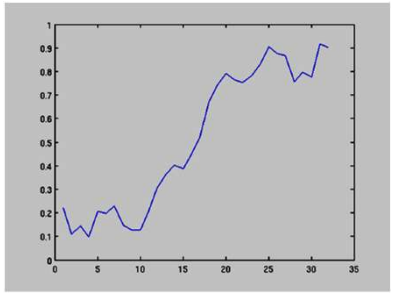
n=4



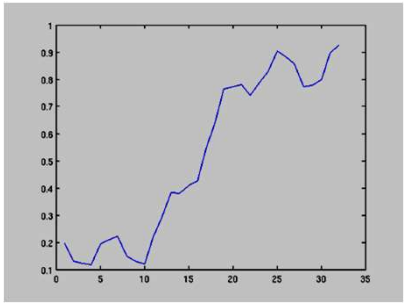
n=8



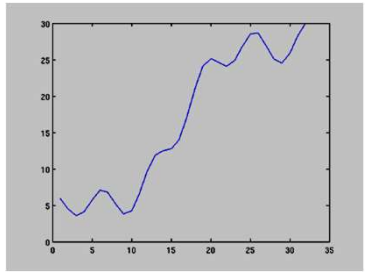
n=10



n=13

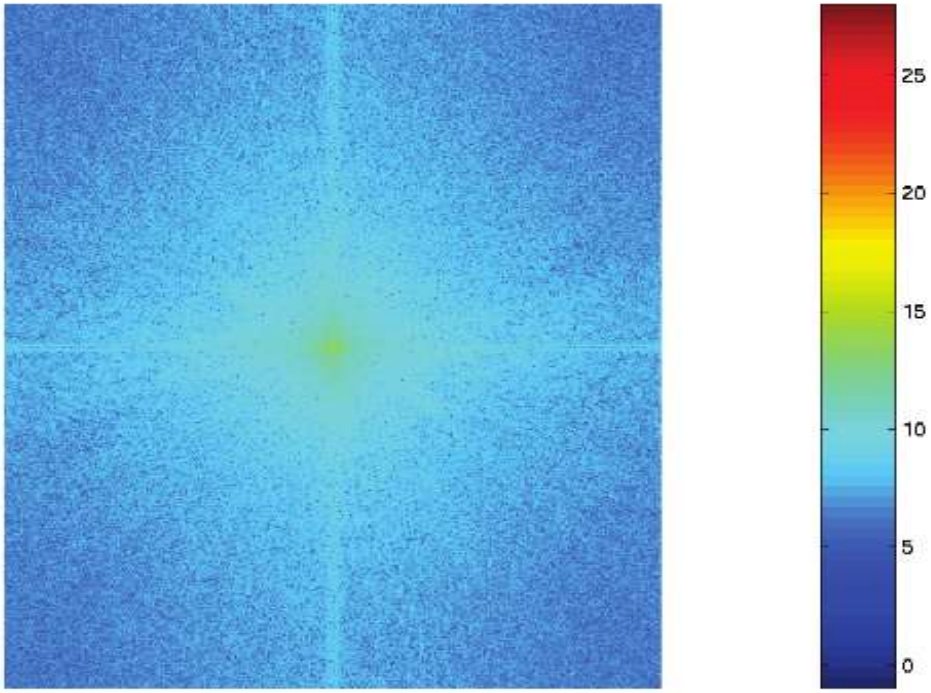


n=15



Original

Look back to 2D Fourier spectra (INF2310)



Most of the energy is concentrated along the lowest frequencies. We can reconstruct the image with an increasing accuracy by starting with the lowest frequencies and adding higher frequencies.

Fourier Symbol reconstruction

- Inverse Fourier transform gives an approximation to the original contour

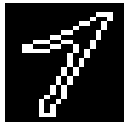
$$\hat{s}(k) = \sum_{k=0}^{N-1} a(u) \exp\left(\frac{2\pi i u k}{M}\right), \qquad k \in [0, M-1]$$

- We have only used N features to reconstruct each component of $\hat{s}(k)$.
- The number of points in the approximation is the same (M), but the number of coefficients (features) used to reconstruct each point is smaller ($N<M$).

- Use an even number of descriptors.
- The first 10-16 descriptors are found to be sufficient for character description. They can be used as features for classification.
- The Fourier descriptors can be invariant to translation and rotation if the coordinate system is appropriately chosen.
- All properties of 1D Fourier transform pairs (scaling, translation, rotation) can be applied.

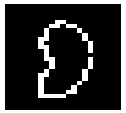
Fourer descriptor example

Image, 26x21 = 546 pixels

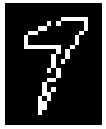


Boundary

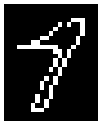
2 coefficients



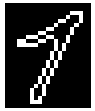
4 coefficients



6 coefficients



8 coefficients



20 coefficients

Matlab DIPUM Toolbox:

```
b=boundaries(f);
b=b{1};
%size(b) tells that the contour is 65
    pixels long
bim=bound2im(b,26,21);
%must tell image dimension
z=frdescp(b);
zinv2=ifrdesc(z,2);
z2im=bound2im(zinv2,26,21);
imshow(z2im);
```

Fourier coefficients and invariance

- Translation affects only the center of mass (a(0)).
- Rotation only affects the phase of the coefficients.
- Scaling affects all coefficients in the same way, so ratios $a(u_1)/a(u_2)$ are not affected.
- The start point affects the phase of the coefficients.

- Normalized coefficients can be obtained, but is beyond the scope of this course.
  - *See e.g., ØD Trier, A Jain and T Taxt, «Feature extraction methods for character recognition – a survey». Pattern Recognition, vol. 29, no. 4, pp. 641-662, 1996.*

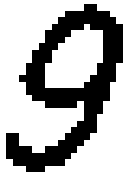
Examples from Trier et al. 1996

Figure 18: Character '4' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, ...10; 15, 20, 30, 40, 50, and 100, respectively.

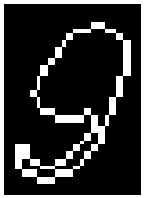
Figure 19: Character '5' reconstructed by elliptic Fourier descriptors of orders up to 1, 2, ...10; 15, 20, 30, 40, 50, and 100, respectively.

Remark about Fourier descriptors

Note that the contour of



is



Objects with holes, like 8, 9, 6 will only by described by the outer contour.

The length of a(u) will depend on the number of boundary points (and the size of the object).

We should resample the contour by selecting M evenly spaced points along the contour.

Run Length Encoding of Objects

- See also G&W, "DIP3E", 8.2.7, G&W, "DIP4E", 8.8
- Sequences of adjacent pixels are represented as "runs".
- Absolute notation of foreground in binary images:
  - Run_i = …;<row_i, $column_i$, $runlength_i$>; …
- Relative notation in graylevel images:
  - …;($graylevel_i$, $runlength_i$); …
- This is used as a lossless compression transform.
- Relative notation in binary images:

  Start value, $length_1$, $length_2$, …, eol,

  …

  Start value, $length_1$, $length_2$, …, eol,eol.
- This is also useful for representation of image bit planes.
- RLE is found in TIFF, GIF, JPEG, …, and in fax machines.

"Gray code"

Is the conventional binary representation of graylevels optimal?

- Consider a single band graylevel image having b bit planes.
- We desire a minimum complexity in each bit plane
  - Because the run-length transform will then be most efficient.
- Conventional binary representation gives high complexity:
  - If the graylevel value fluctuates between 2^k-1 and 2^k, k+1 bits will change value. Example: 127 = 01111111 while 128 = 10000000
- In "Gray Code" only one bit changes if graylevel is changed by 1.
- The transition from binary code to "Gray Code" is a reversible transform, while both binary code and "Gray Code" are codes.

Binary vs Gray Code transforms *

- ”Binary Code” to ”Gray Code”:
  1. Start by MSB in BC and keep all 0 until you hit 1
  2. 1 is kept, but all following bits are complemented until you hit 0
  3. 0 is complemented, but all following bits are kept until you hit 1
  4. Go to 2.

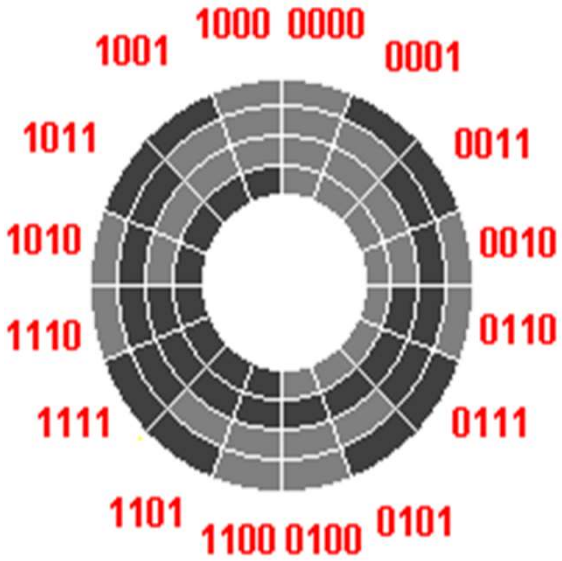
- ”Gray Code” to ”Binary Code”:
  1. Start by MSB in GC and keep all 0 until you hit 1
  2. 1 is kept, but all following bits are complemented until you hit 1.
  3. 1 is complemented, but all following bits are kept until you hit 1.
  4. Go to 2.

  * (This is just to demonstrate that there are simple transforms)

Encoding angular position of a shaft as a binary number

4 bits codes

Decimal	Binary	Gray
0_d	0000_b	0000_g
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

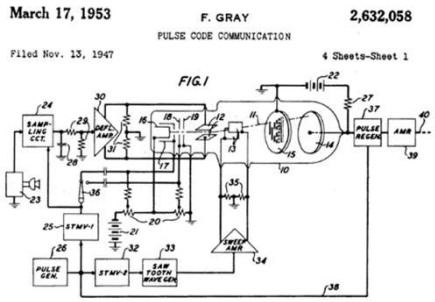


”Gray code shaft encoder”

Gives secure reading of angle.

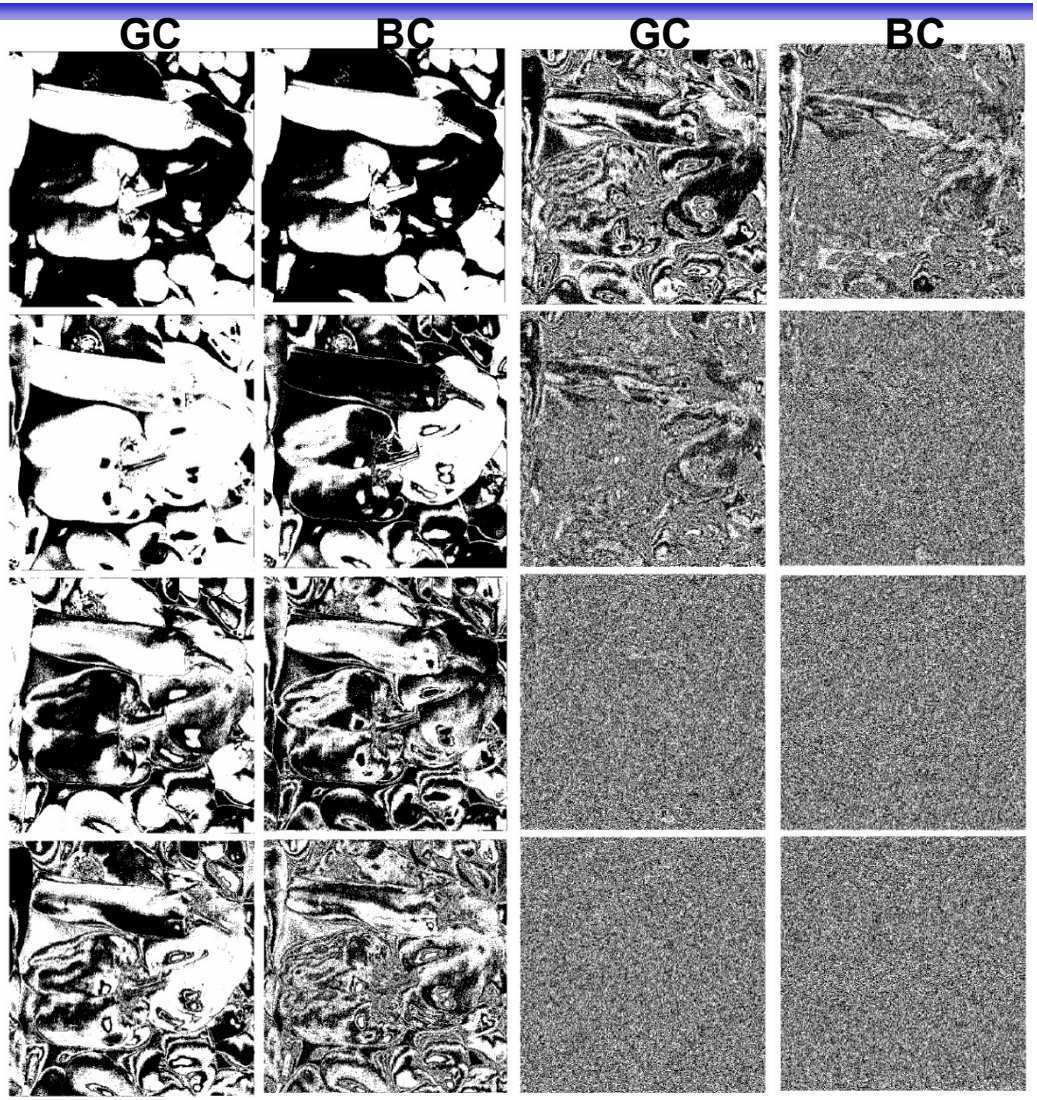
Émile Baudot’s telegraph 1878.

Code patented by Frank Gray in 1953.



Gray code in graylevel images

- MSB is the same in the two representations.
- Larger homogeneous areas in each bitplane in Gray Code than in natural binary code.
- Less noisy bitplanes in Gray Code.
- More compact run-length coding of bitplanes in Gray Code.

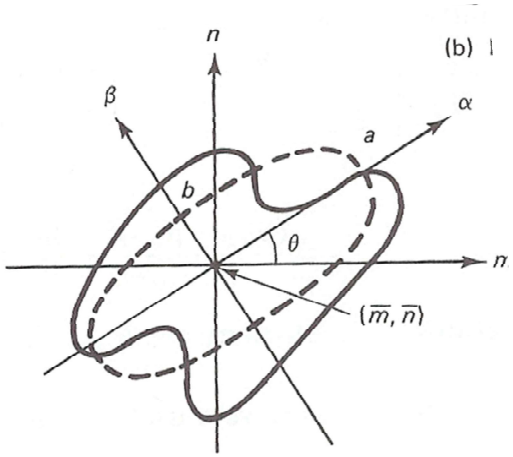


Learning goals - representation

- Understand difference between representation and description.
- Chain coding of contours
  - Absolute code invariants:
    - Start point: Min circular shift. Rotation: First difference + min circular shift
  - Relative code invariants
    - Rotation: inherent! Start point: Minimum circular shift
- Polygonization of contours
  - Recursive and sequential
- Signatures of contours
- Convex hull of contours
- Skeletons: Thinning
- Fourier descriptors of contours
- Run Length Encoding of objects
- Less complex bit planes in Gray coded images

INF 4300 – Digital Image Analysis

OBJECT DESCRIPTION – SHAPE FEATURE EXTRACTION



(c) Best fit ellipse

Fritz Albregtsen 23.09.2020

Today

From the textbook:	DIP3E	DIP4E
Boundary (Feature) Descriptors	11.2 (815-822)	11.3 (831-840)
Regional (Feature)Descriptors	11.3 (822-842)	11.4 (840-859)

Albregtsen: "Object shape descriptors - Area, perimeter, compactness, and spatial moments"

Curriculum also includes these lecture notes.

Today we cover the following :

1. Introduction
2. Topological features
3. Projections
4. Geometric features
5. Statistical shape features
6. Moment-based geometric features
7. Finding the best feature subset

What is feature extraction?

- Devijver and Kittler (1982):

  "Extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class variability".

  – Within-class pattern variability: AAAAAAAAAAAA
  variance between objects belonging to the same class.

  – Between-class pattern variability: A B C D E F G H I J
  variance between objects from different classes.

Feature extraction

- We will discriminate between different object classes based on a set of features.
- The features are often chosen given the application.

- Normally, a large set of different features is investigated.
- Classifier design also involves feature selection
  - selecting the best subset out of a larger feature set.
- Given a training data set of a certain size, the dimensionality of the feature vector must be limited.

- Careful selection of an optimal set of features is the most important step in image classification!

Feature extraction methods

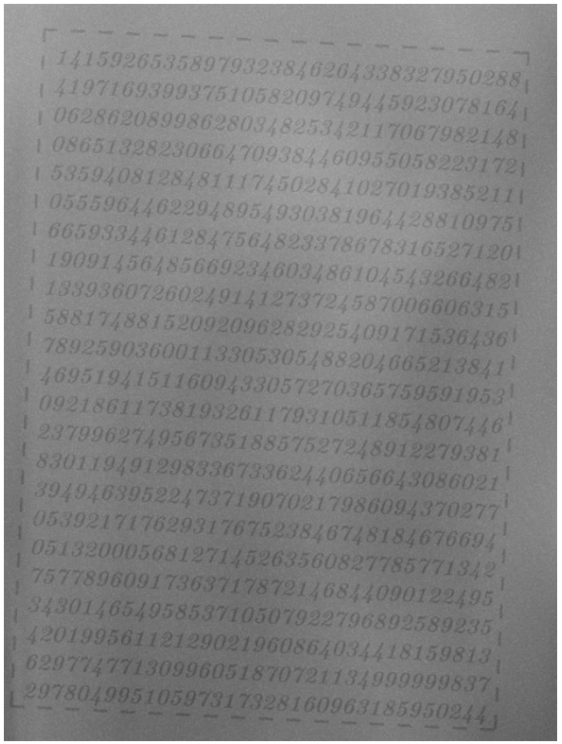
- There are a lot of different feature extraction methods, you will only learn some in this course.
- The focus of this lecture is on **features for describing the shape of an object**.
- Features can also be extracted in local windows around each pixel, e.g.
  - texture descriptors,
  - colour features,
  - or other metods.
- The features will later be used for object recognition and/or classification.

Example: Recognize printed numbers

- Goal: get the series of digits,
  e.g. 14159265358979323846……

Steps in the program:

1. Segment the image to find digit pixels.
2. Find angle of rotation and rotate back.
3. Create region objects – one object pr. digit or connected component.
4. Compute features describing shape of objects
5. Train a classifier on many objects of each digit.
6. Assign a class label to each new object,
   i.e., the class with the highest probability.



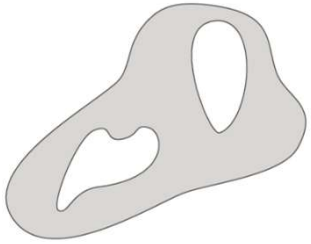
Focus of this lecture

Typical image analysis tasks

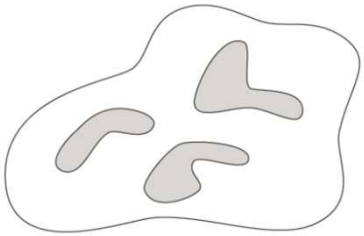
- Preprocessing/noise filtering
- Segmentation
- Feature extraction
  - Are the original image pixel values sufficient for classification, or do we need additional features?
  - What kind of features do we use in order to discriminate between the object classes involved?
- Exploratory feature analysis and selection (next lecture)
  - Which features separate the object classes best?
  - How many features are needed?
- Classification (following three/four lectures)
  - From a set of object examples with known class, decide on a method that separates objects of different types.
  - For new objects: assign each object/pixel to the class with the highest probability
- Testing and validation of classifier accuracy

Topologic features

- This is a group of invariant <u>integer</u> object shape features
  - Invariant to position, rotation, scaling, warping
- Features based on the object skeleton
  - Number of terminations (one line from a point)
  - Number of breakpoints or corners (two lines from a point)
  - Number of branching points (three lines from a point)
  - Number of crossings (> three lines from a point)
- Region features:
  - Number of holes in the object (H)
  - Number of components (C)
  - Euler number, E = C - H
    - Number of connected components – number of holes
  - Symmetry



Region with two holes



Regions with three connected components

1D Projection histograms

- For each row in the region,
  count the number of object pixels.

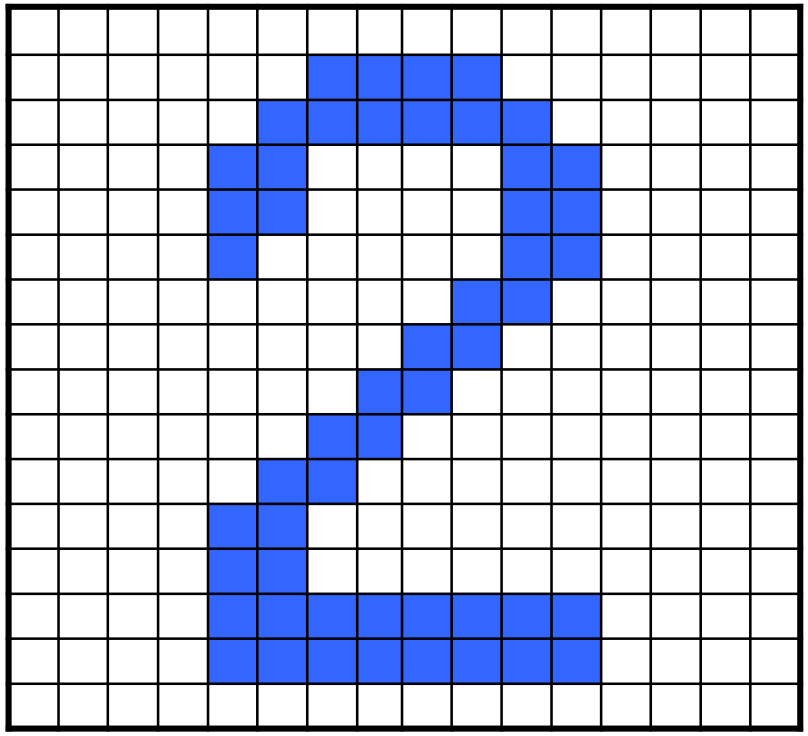
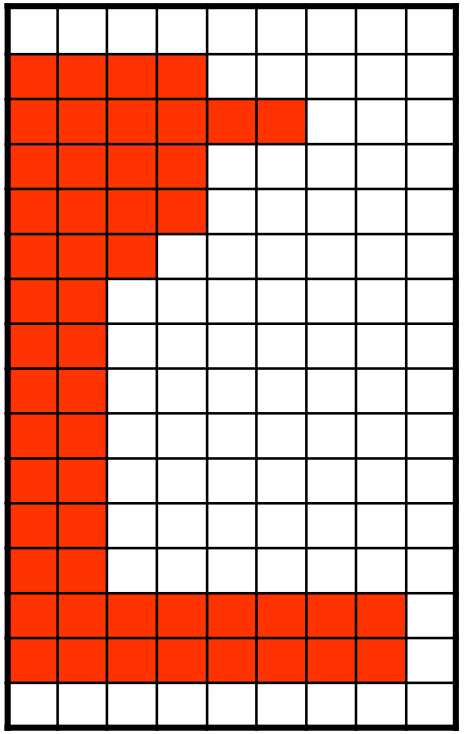


Image – binary region pixels



Row histogram

Projections

- 1D horizontal projection of the region (project on the x-axis):

$$p_h(x) = \sum_y f(x, y)$$

- 1D vertical projection of the region (project on the y-axis):

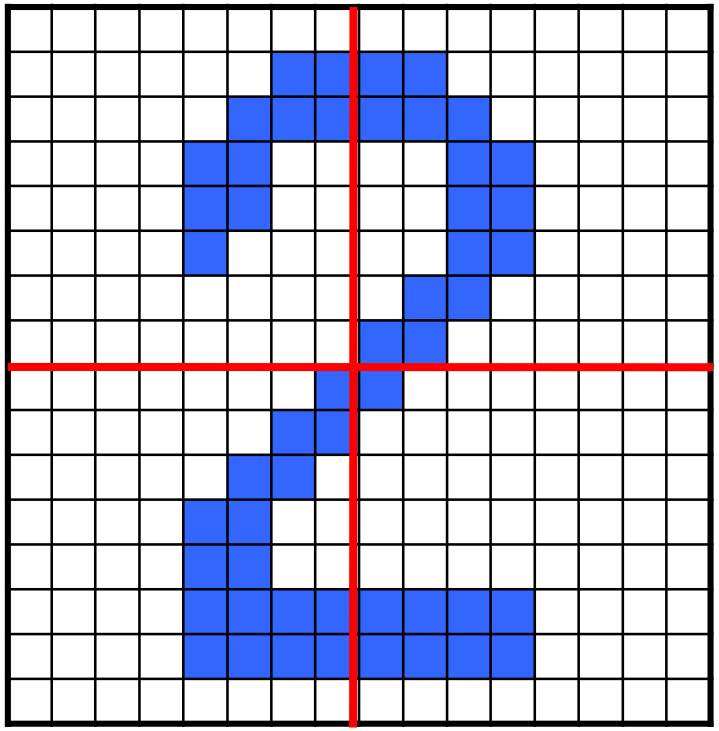
$$p_v(y) = \sum_x f(x, y)$$

- f(x,y) is normally the binary segmented image
- Can be made scale independent by using a fixed number of bins and normalizing the histograms.

- Radial projection in reference to centroid -> ”signature”, see previous lecture.

Use of projection histograms

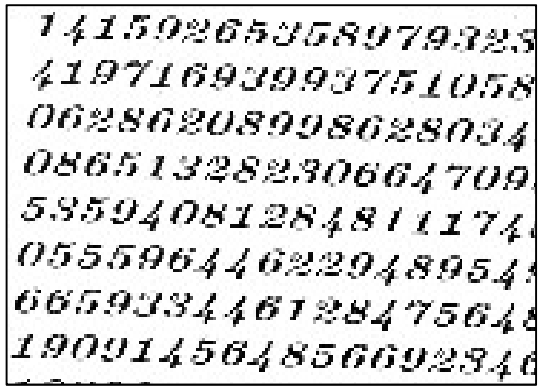
- Divide the object into different regions and compute projection histograms for each region.
  - How can we use this to separate 6 and 9?
- Compute features from the histograms.
  - E.g. mean and variance of the histograms.
- The histograms can also be used as features directly.

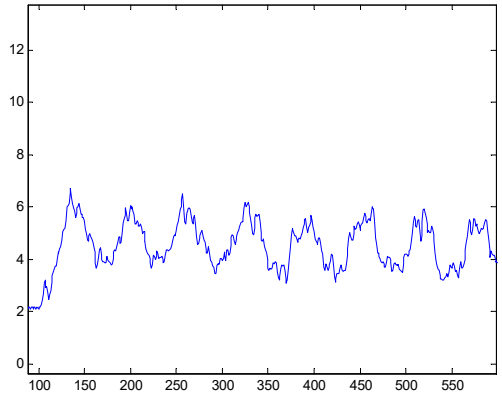


- Projections are also useful for preprocessing (e.g., finding and correcting angle of rotation).

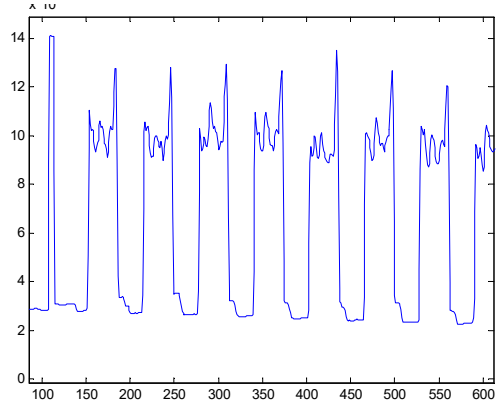
Use of projection histograms

- Check if a page with text is rotated



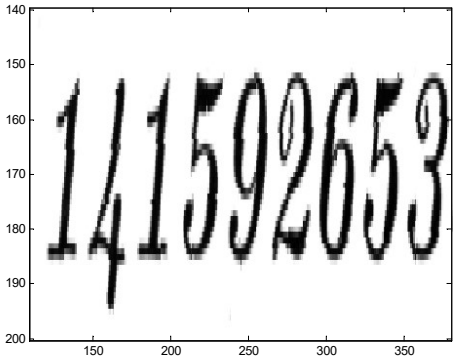


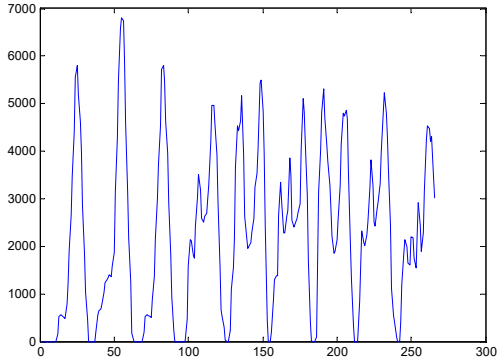
Y-axis projection of page



Y-axis projection after rotation correction

- Then detect lines, connected objects, single symbols, …





x-axis projection of page

Object area

- Generally, the area is defined as: $A = \int_X \int_Y I(x,y)\,dx\,dy$

  I(x,y) = 1 if the pixel is within the object, and 0 otherwise.

- In digital images: $A = \sum_X \sum_Y I(x,y)\,\Delta A$

  ΔA = area of one pixel. If ΔA = 1, area is simply measured in pixels.

- Area changes if we change the scale of the image
  - change is not perfectly linear, because of the discretization of the image.

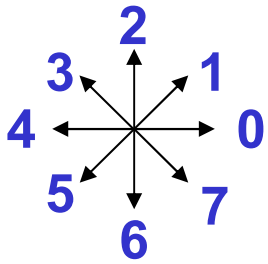
- Area ≈ invariant to rotation (except small discretization errors).

Geometric features from contours

- Boundary length/perimeter
- Area
- Curvature
- Diameter/major/minor axis
- Eccentricity
- Bending energy
- Basis expansion (Fourier – last week)

Perimeter length from chain code

- Distance measure differs when using 8- or 4-neighborhood
- Using 4-neighborhood, measured length ≥ actual length.
- In 8-neighborhood, fair approximation from chain code by:



$$P_F = n_E + n_O \sqrt{2}$$

where $\mathbf{n_E}$ and $\mathbf{n_O}$ are the number of even / odd chain elements .

- This overestimates real perimeters systematically.
- Freeman (1970) computed the area and perimeter of the chain by

$$A_F = \sum_{i=1}^{N} c_{ix} \left(y_{i-1} + {c_{iy}}/{2} \right), \quad P_F = n_E + n_O \sqrt{2}$$

- where **N** is the length of the chain, $\mathbf{c_{ix}}$ and $\mathbf{c_{iy}}$ are the **x** and **y** components of the **i**th chain element $\mathbf{c_i}$ ($\mathbf{c_{ix}}$, $\mathbf{c_{iy}}$ = {1, 0, -1} indicate the change of the **x**- and **y**-coordinates), $\mathbf{y_{i-1}}$ is the **y**-coordinate of the start point of the chain element $\mathbf{c_i}$.

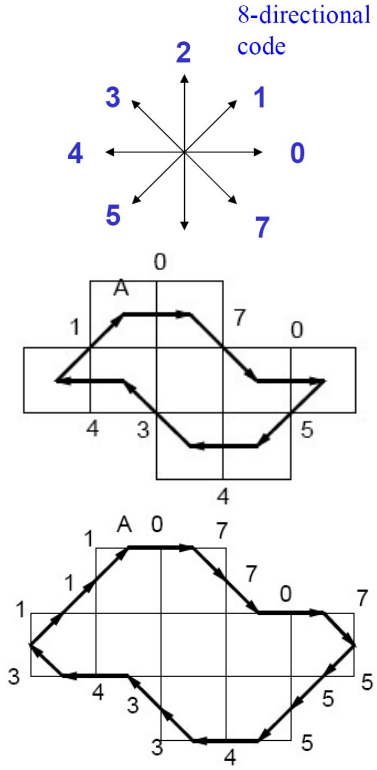
Perimeter from chain code

- Vossepoel and Smeulders (1982) improved perimeter length estimate by a corner count n_C, defined as the number of occurrences of unequal consecutive chain elements:

$$P_{VS} = 0.980\, n_E + 1.406 n_O - 0.091\, n_C$$

- Kulpa (1977) gave the perimeter as

$$P_K = \frac{\pi}{8}\left(1+\sqrt{2}\right)\left(n_E + \sqrt{2} n_O\right)$$



Pattern matching - bit quads

- Let n{Q} = number of matches between image pixels and pattern Q.
- Then area and perimeter of 4-connected object is given by:

$$A = n\{1\}, \quad P = 2n\{0 \quad 1\} + 2n\begin{Bmatrix} 0 \\ 1 \end{Bmatrix}$$

"Bit Quads" can handle 8-connected images:

$$Q_0: \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$Q_1: \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$Q_2: \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

$$Q_3: \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$Q_4: \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$Q_D: \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Gray (1971) gave area and the perimeter as

$$A_G = \frac{1}{4}\left[n\{Q_1\} + 2n\{Q_2\} + 3n\{Q_3\} + 4n\{Q_4\} + 2n\{Q_D\}\right], \quad P_G = n\{Q_1\} + n\{Q_2\} + n\{Q_3\} + 2n\{Q_D\}$$

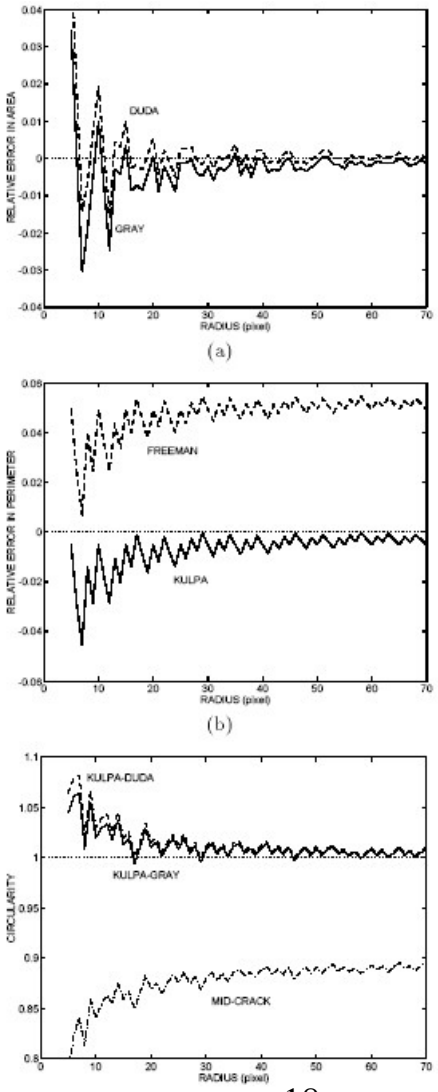
- More accurate formulas by Duda :

$$A_D = \frac{1}{4}\left[n\{Q_1\} + 2n\{Q_2\} + \frac{7}{2}n\{Q_3\} + 4n\{Q_4\} + 3n\{Q_D\}\right], \quad P_D = n\{Q_2\} + \frac{1}{\sqrt{2}}\left[n\{Q_2\} + n\{Q_3\} + 2n\{Q_D\}\right]$$

-

A comparison of methods

- We have tested the methods on circles, R ={5,…,70}.
- Area estimator :
  - Duda is slightly better than Gray.
- Perimeter estimator :
  - Kulpa is more accurate than Freeman.
- Circularity :
  - Kulpa's perimeter and Gray's area gave the best result.
- Errors and variability largest when R is small.
  - Test this yourself on this image: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$
- Best area and perimeter not computed simultaneously.
- Gray's area can be computed using discrete Green's theorem, suggesting that the two estimators can be computed simultaneously during contour following.



Object area from contour

- The surface integral over S (having contour C) is given by Green's theorem:

$$A = \iint_S dxdy = \int_C xdy$$

```
s := 0.0;
n := n + 1;

pkt[n].x := pkt[1].x;
pkt[n].y := pkt[1].y;
for i:=2 step 1 until n do
begin
        dy := pkt[i].y - pkt[i-1].y
        s := s + (pkt[i].x + pkt[i-1].x)/2 * dy;
end;
area := if (s > 0) then s else -s;
```

- The region can also be represented by n polygon vertices (see previous lecture)

$$\hat{A} = \frac{1}{2}\left|\sum_{k=0}^{N-1}\left(x_k\, y_{k+1} - x_{k+1}\, y_k\right)\right|$$

where the sign of the sum reflects the polygon orientation.

Compactness and circularity

- Compactness (very simple measure)
  - $\gamma = P^2 / (4\pi A)$, where P = Perimeter, A = Area,
  - For a circular disc, γ is minimum and equals 1.
  - Compactness attains high value for complex object shapes, but also for very elongated simple objects, like rectangles and ellipses where a/b ratio is high.

  => Compactness is not correlated with complexity!

- G&W defines
  - Compactness = P^2/A
  - Circularity ratio = $4\pi A/P^2$

Y=3.4
Circ=0.28

Y=10.1
Circ=0.09

Circularity and irregularity

- Circularity may be defined by $C = 4\pi A/P^2$.
- C = 1 for a perfect continuous circle; betw. 0 and 1 for other shapes.
- In digital domain, C takes its smallest value for a
  - digital octagon in 8-connectivity perimeter calculation
  - digital diamond in 4-connectivity perimeter calculation
- Dispersion may be given as the major chord length to area
- Irregularity of an object of area A can be defined as:

$$D = \frac{\pi \max\left((x_i - \bar{x})^2 + (y_i - \bar{y})^2\right)}{A}$$

  - where the numerator is the area of the centered enclosing circle.
- Alternatively, ratio of maximum and minimum centered circles:

$$I = \frac{\max\left(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}\right)}{\min\left(\sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}\right)}$$

Curvature

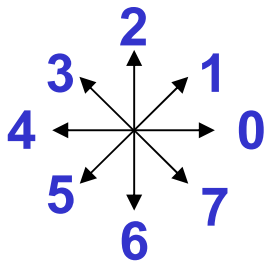
- In the continous case, curvature is the rate of change of slope.

$$|\kappa(s)|^2 = \left[\frac{d^2x}{ds^2}\right]^2 + \left[\frac{d^2y}{ds^2}\right]^2$$

- In the discrete case, difficult because boundary is locally ragged.

- Use difference between slopes of adjacent boundary segments to describe curvature at point of segment intersection.

- Curvature can be calculated from chain code.



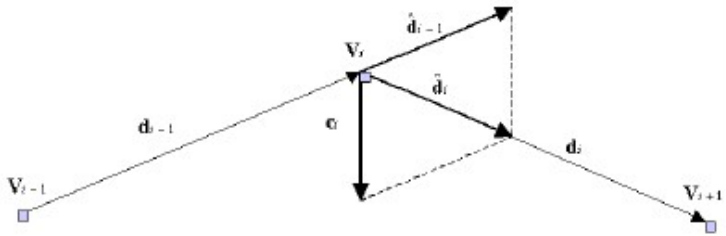
How to get from chain code to curvature, a simple example:

0, 0, 2, 0, 1, 0, 7, 6, 0, 0 --> 0, 2, -2, 1, -1, -1, -1, 2, 0

then square: <u>0, 4, 4, 1, 1, 1, 1, 4, 0</u>

Discrete computation of curvature

- Trace the boundary and insert vertices, at a given distance (e.g. 3 pixels apart), or by polygonization (previous lecture).



- Compute local curvature c_i as the difference between the directions of two edge segments joining a vertex:

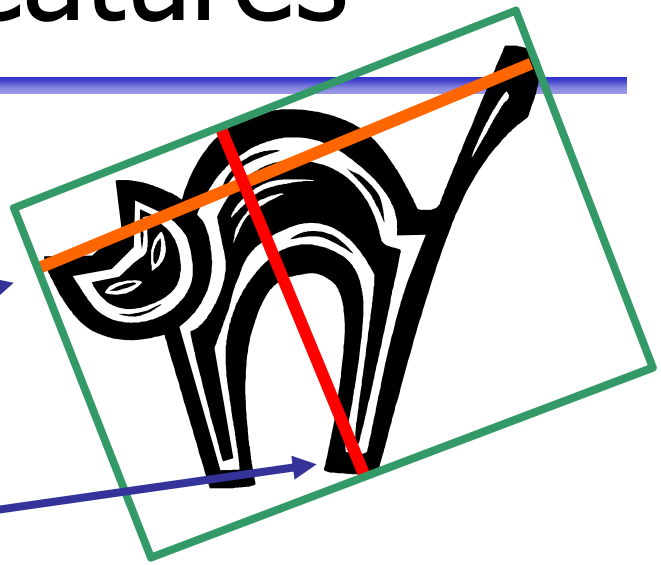
$$c_i = \vec{d}_i - \vec{d}_{i-1}$$

v_i: edge segment i
$d\hat{}_{t-1}$: unit vectors of edge segments d_{t-1} and dt
c_i: local curvature at point i

- Curvature feature: sum all local curvature measures along the border.

- More complex regions get higher curvature.

Contour based features



- Diameter = Major axis (a)

  Longest distance of a line segment

  connecting two points on the perimeter

- Minor axis (b)

  Computed along a direction perpendicular to the major axis. Largest length possible between two border points in the given direction.

Eccentricity: 4.7

Eccentricity: 1.8

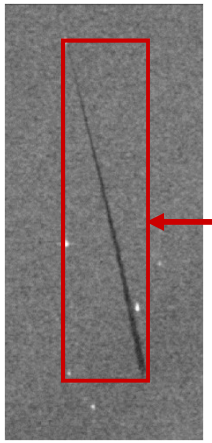
- So called "Eccentricity" of the contour (a/b)

Bounding box and CH features

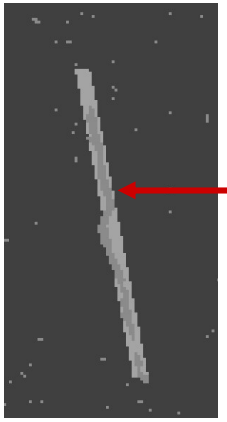
- Regular bounding box features:
  - Width/height of bounding box
  - Centre of mass position in box

- If the object's orientation is known, a bounding box can also be oriented along this direction. <u>More meaningful!</u>

- Extent = Area/(Area of bounding box)
  - But which type of bounding box?
- Solidity = Area/(Area of Convex Hull) (also termed "convexity")



Regular (image oriented) bounding box



Object-oriented bounding box

Moments

- Borrows ideas from physics and statistics.
- For a given continuous intensity distribution *g(x, y)* we define moments m_{pq} by

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q g(x, y) dx dy$$

- For sampled (and bounded) intensity distributions f(x, y) over a region R

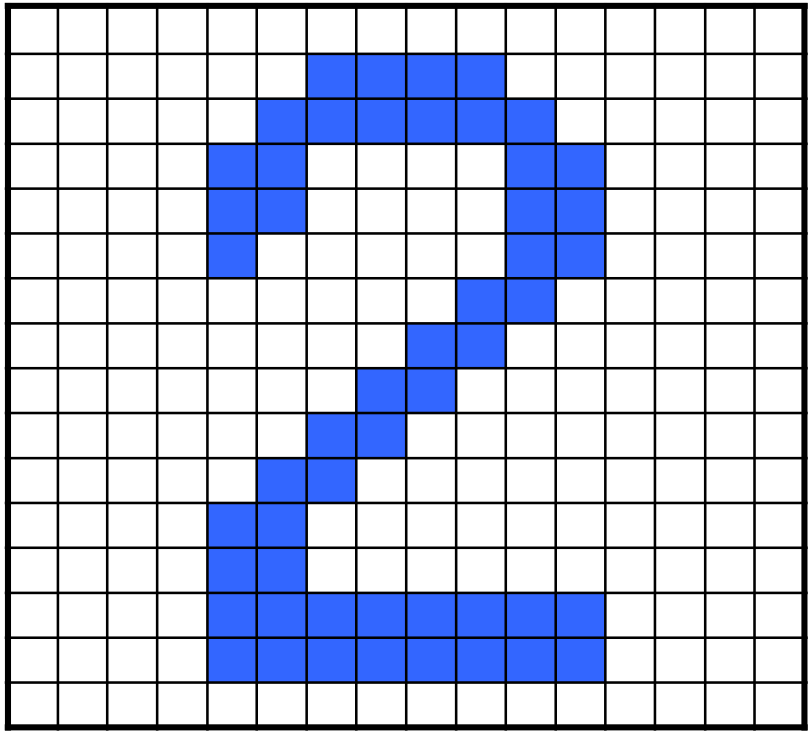
$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y)$$

- A moment m_{pq} is said to be of *order* p + q.

Moments from binary images - area

- For binary images, where
  - f (x, y) = 1 $\Rightarrow$ object pixel
  - f (x, y) = 0 $\Rightarrow$ background pixel
- Area

$$m_{00} = \sum_x \sum_y f(x, y)$$



Centroid/center of mass from moments

- For binary images, where

  f (x, y) = 1 $\Rightarrow$ object pixel

  f (x, y) = 0 $\Rightarrow$ background pixel

- Center of mass /"tyngdepunkt"

$$m_{10} = \sum_x \sum_y x f(x,y) = \bar{x} m_{00} \quad \Rightarrow \quad \bar{x} = \frac{m_{10}}{m_{00}}$$

$$m_{01} = \sum_x \sum_y y f(x,y) = \bar{y} m_{00} \quad \Rightarrow \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

gives the position of the object

Grayscale moments

- In gray scale images, we may regard *f(x,y)* as a discrete 2-D probability distribution over (x,y)

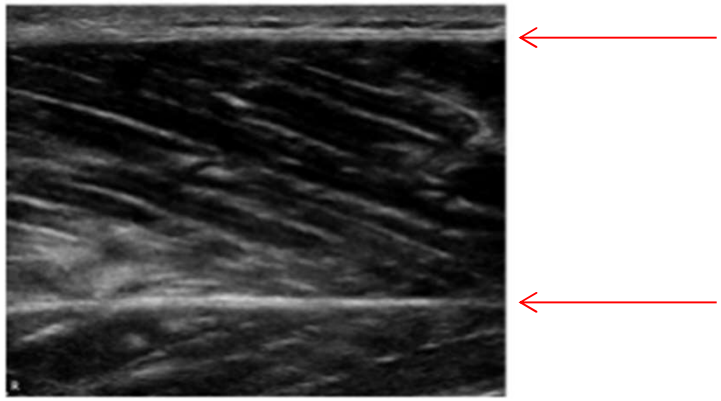
- For probability distributions, we should have

$$m_{00} = \sum_x \sum_y f(x,y) = 1$$

  – And if this is not the case we can normalize by

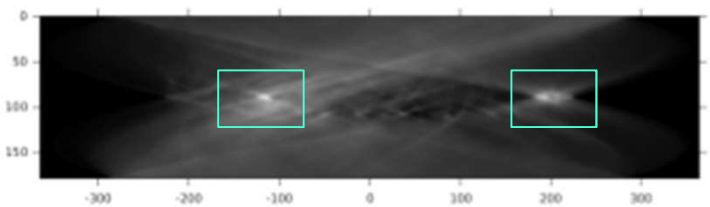
$$F(x,y) = f(x,y)/m_{00}$$

Example use of centroid of grayscale image



Ultrasound image of muscle fibers
Want to find the near-horisontal lines
using Radon-transform
(generalized Hough to grayscale images)

(a) Image before transformation.



Find peaks in the Radon domain
How do we robustly find the location
of a «peak» in the marked areas?

Central moments

- These are position invariant moments, defined by

$$\mu_{p,q} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

- where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$

- The total mass, and the center of mass coordinates are given by

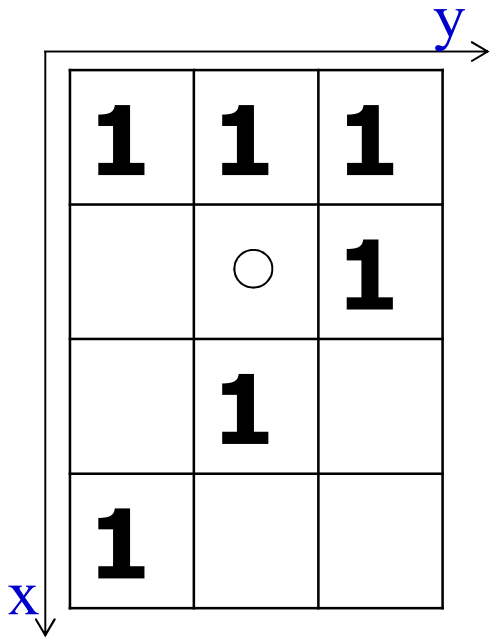
$$\mu_{00} = \sum_x \sum_y f(x, y), \qquad \mu_{10} = \mu_{01} = 0$$

- This corresponds to computing ordinary moments after having translated the object so that center of mass is in origo.

- Central moments are independent of position, but are not scaling or rotation invariant.

- Q: What is μ_{00} for a binary object?

A simple example: center of mass



$$m_{00} = \sum_x \sum_y f(x,y) \quad =6$$

$$m_{10} = \sum_x \sum_y x f(x,y) = \bar{x} m_{00} \quad \Rightarrow \quad \bar{x} = \frac{m_{10}}{m_{00}} \quad =2$$

$$m_{01} = \sum_x \sum_y y f(x,y) = \bar{y} m_{00} \quad \Rightarrow \quad \bar{y} = \frac{m_{01}}{m_{00}} \quad =2$$

2D Central moments μ_{pq} from ordinary moments m_{pq}

- Moments μ_{pq} ($p + q \leq 3$) are given by m_{pq} by:

$$\mu_{00} = m_{00}, \;\; \mu_{10} = 0, \;\; \mu_{01} = 0$$
$$\mu_{20} = m_{20} - \bar{x} m_{10}$$
$$\mu_{02} = m_{02} - \bar{y} m_{01}$$
$$\mu_{11} = m_{11} - \bar{y} m_{10}$$
$$\mu_{30} = m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10}$$
$$\mu_{12} = m_{12} - 2\bar{y} m_{11} - \bar{x} m_{02} + 2\bar{y}^2 m_{10}$$
$$\mu_{21} = m_{21} - 2\bar{x} m_{11} - \bar{y} m_{20} + 2\bar{x}^2 m_{01}$$
$$\mu_{03} = m_{03} - 3\bar{x} m_{02} + 2\bar{y}^2 m_{01}$$

Generalization to 3D moments from m_{pq}

- The 3D μ_{pqr}, are expressed by m_{pqr}:

$$\mu_{pqr} = \sum_{S=0}^{p} \sum_{t=0}^{q} \sum_{u=0}^{r} -1^{[D-d]} \binom{p}{s} \binom{q}{t} \binom{r}{u} \Delta x^{p-s} \, \Delta y^{q-t} \, \Delta z^{r-u} \, m_{stu}$$

- where

  D = (p + q + r); d = (s + t + u)

- and the binomial coefficients are given by

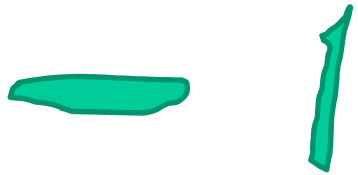
$$\binom{v}{w} = \frac{v!}{w! \, (v-w)!}, \quad w < v$$

Moments of inertia or Variance

- The two second order central moments measure the spread of points around the y- and x-axis through the centre of mass

$$\mu_{20} = \sum_x \sum_y (x - \bar{x})^2 f(x, y)$$

$$\mu_{02} = \sum_x \sum_y (y - \bar{y})^2 f(x, y)$$



- From physics: moment of inertia about an axis:
  how much energy is required to rotate the object about this axis:
  - Statisticans like to call this variance.

- The cross moment of intertia is given by

$$\mu_{11} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) f(x, y)$$

  - Statisticians call this covariance or correlation.

- Orientation of the object can be derived from these moments.
  - **This implies that they are not invariant to rotation.**

A simple example

$$\mu_{20} = \sum_x \sum_y (x - \bar{x})^2 f(x,y) \quad = 5$$

$$\mu_{02} = \sum_x \sum_y (y - \bar{y})^2 f(x,y) \quad = 5$$

$$\mu_{11} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) f(x,y) \quad = -5$$

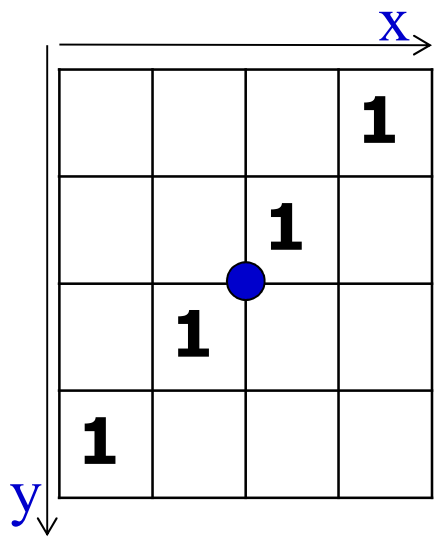


Image coordinates

$$\begin{bmatrix} 4,1 \\ 3,2 \\ 2,3 \\ 1,4 \end{bmatrix}$$

(Note that image coordinates are swapped)

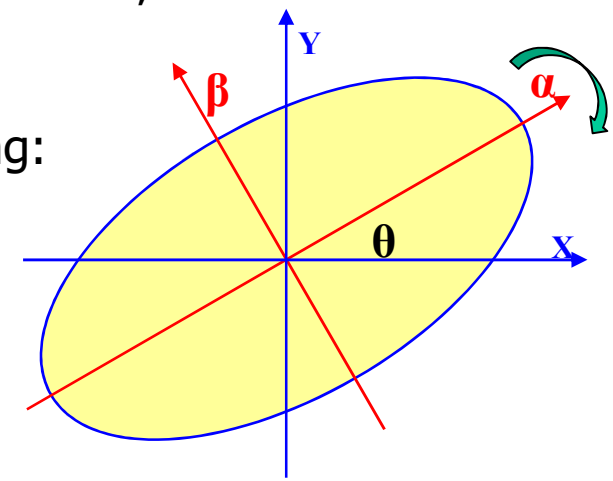
Object orientation - I

- Orientation is defined as the angle, relative to the X-axis, of an axis through the centre of mass that gives the lowest moment of inertia.
- Orientation θ relative to X-axis found by minimizing:

$$I(\theta) = \sum_{\alpha} \sum_{\beta} \beta^2 \, f(\alpha, \beta)$$

where the rotated coordinates are given by

$$\alpha = x\cos\theta + y\sin\theta, \qquad \beta = -x\sin\theta + y\cos\theta$$



- The second order central moment of the object around the α-axis, expressed in terms of x, y, and the orientation angle θ of the object is:

$$I(\theta) = \sum_{x} \sum_{y} \left[y\cos\theta - x\sin\theta \right]^2 f(x, y)$$

- We take the derivative of this expression with respect to the angle θ
- Set derivative equal to zero, and find a simple expression for θ :

Object orientation - II

- Second order central moment around the α-axis:

$$I(\theta)=\sum_x\sum_y\left[y\cos\theta-x\sin\theta\right]^2 f(x,y)$$

- Derivative w.r.t. Θ = 0 =>

$$\frac{\partial}{\partial\theta}I(\theta)=\sum_x\sum_y 2f(x,y)\left[y\cos\theta-x\sin\theta\right]\left[-y\sin\theta-x\cos\theta\right]=0$$

$$\Downarrow$$

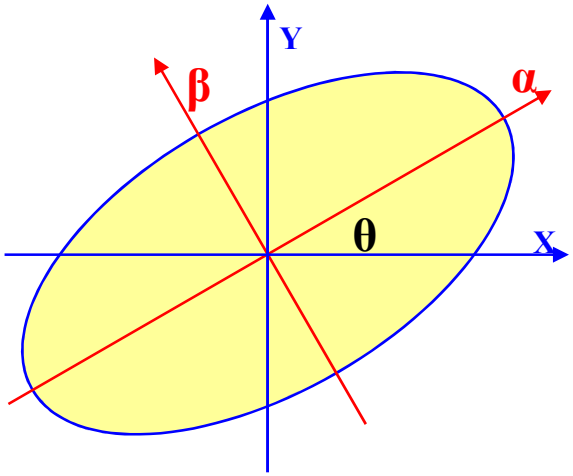
$$\sum_x\sum_y 2f(x,y)\left[xy\left(\cos^2\theta-\sin^2\theta\right)\right]=\sum_x\sum_y 2f(x,y)\left[x^2-y^2\right]\sin\theta\cos\theta$$

$$\Downarrow$$

$$2\mu_{11}\left(\cos^2\theta-\sin^2\theta\right)=2\left(\mu_{20}-\mu_{02}\right)\sin\theta\cos\theta$$

$$\Downarrow$$

$$\frac{2\mu_{11}}{\left(\mu_{20}-\mu_{02}\right)}=\frac{2\sin\theta\cos\theta}{\left(\cos^2\theta-\sin^2\theta\right)}=\frac{2\tan\theta}{1-\tan^2\theta}=\tan(2\theta)$$



- So the object orientation is given by:

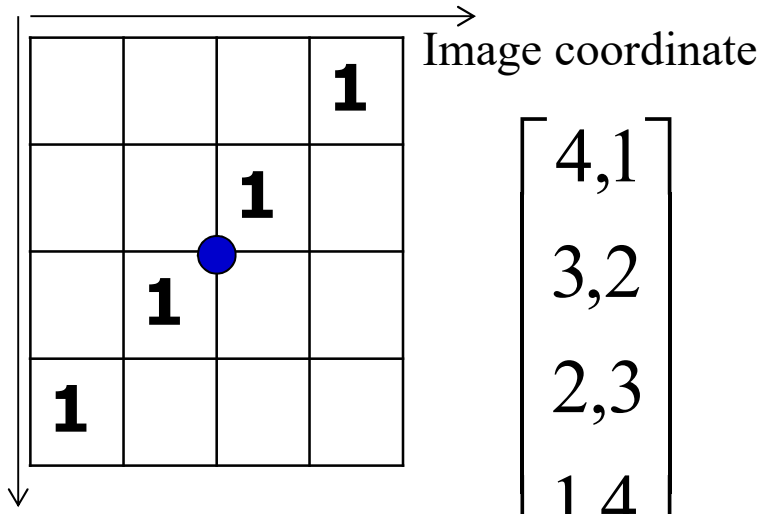
$$\underline{\underline{\theta=\frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{\left(\mu_{20}-\mu_{02}\right)}\right]}},\qquad where\ \ \theta\in\left[0,\pi/2\right] if\ \mu_{11}>0,\ \ \theta\in\left[\pi/2,\pi\right] if\ \mu_{11}<0$$

A simple example

$$\mu_{20} = \sum_x \sum_y (x - \bar{x})^2 f(x,y) \quad = 5$$

$$\mu_{02} = \sum_x \sum_y (y - \bar{y})^2 f(x,y) \quad = 5$$

$$\mu_{11} = \sum_x \sum_y (x - \bar{x})(y - \bar{y}) f(x,y) \quad = -5$$



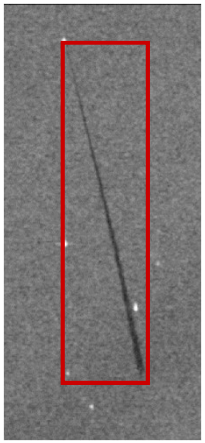
$$\begin{bmatrix} 4,1 \\ 3,2 \\ 2,3 \\ 1,4 \end{bmatrix}$$

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{(\mu_{20} - \mu_{02})}\right], \quad where \;\; \theta \in \left[0, \pi/2\right] if \;\; \mu_{11} > 0, \;\; \theta \in \left[\pi/2, \pi\right] if \;\; \mu_{11} < 0 \quad = -45 \text{degrees}$$

(Note that image coordinates are swapped)

Bounding box - again

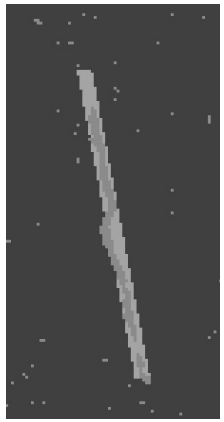
- Image-oriented bounding box:
  - The smallest rectangle around the object, having sides parallell to the edges of the image.
  - Found by searching for min and max x and y within the object (*xmin, ymin, xmax, ymax*)



- Object-oriented bounding box:
  - Smalles rectangle around the object, having one side parallell to the orientation of the object (θ).
  - The transformation

$$\alpha = x\cos\theta + y\sin\theta, \quad \beta = y\cos\theta - x\sin\theta$$

    is applied to all pixels in the object (or its boundary).
  - Then search for α_{min}, β_{min}, α_{max}, β_{max}



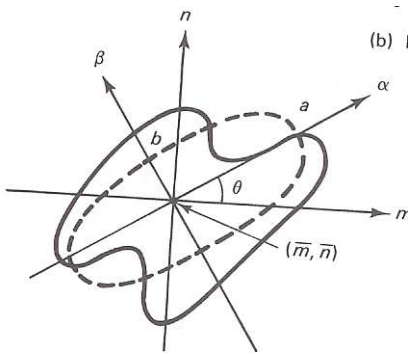
The best fitting ellipse

- Object ellipse is defined as the ellipse whose least and greatest moments of inertia equal those of the object.
- Semi-major and semi-minor axes are given by

$$\left(\hat{a},\hat{b}\right)=\sqrt{\frac{2\left[\mu_{20}+\mu_{02}\pm\sqrt{\left(\mu_{20}+\mu_{02}\right)^2+4\mu_{11}^2}\right]}{\mu_{00}}}$$

- Numerical eccentricity is given by

$$\hat{\varepsilon}=\sqrt{\frac{\hat{a}^2-\hat{b}^2}{\hat{a}^2}}$$



(c) Best fit ellipse

- Orientation invariant object features.
- Gray scale or binary object.

Radius of gyration, K

- The radius of a circle where we could concentrate all the mass of an object without altering the moment of inertia about its center of mass.
- For arbitrary object having a mass μ_{00} and a moment of inertia around the Z-axis, we may write

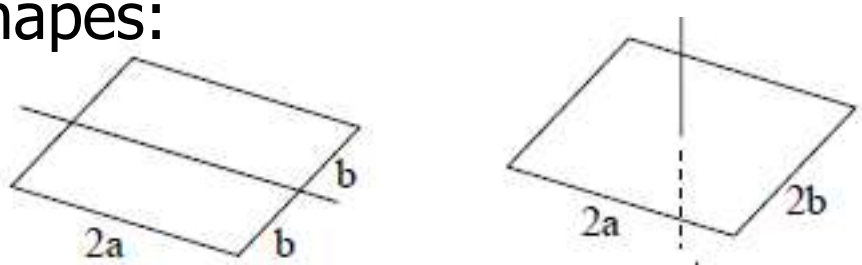
$$I = \mu_{00}\,\hat{K}^2 \quad \Rightarrow \hat{K} = \sqrt{\frac{I_Z}{\mu_{00}}} = \sqrt{\frac{I_X + I_Y}{\mu_{00}}} = \sqrt{\frac{\mu_{20} + \mu_{02}}{\mu_{00}}}$$

- This feature is obviously invariant to rotation.

Radius of gyration, K

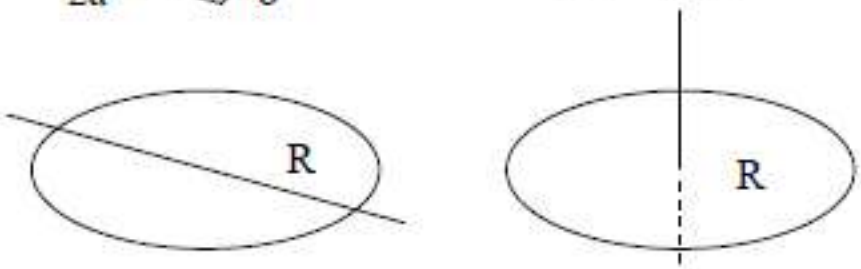
- For homogeneous objects, only determined by geometry.
- Thus, the squared radius of gyration may be tabulated for simple object shapes:

Rectangle: $K^2 = b^2/3$



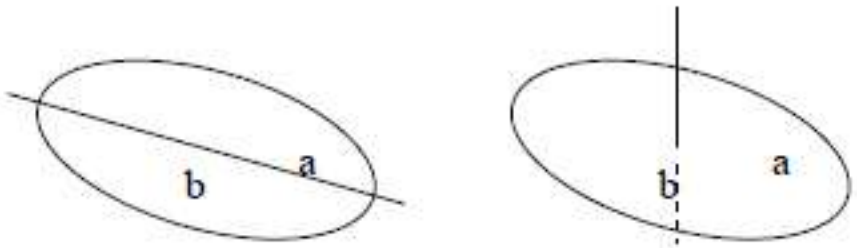
$K^2 = (a^2+b^2)/3$

Circular disk: $K^2 = R^2/4$



$K^2 = R^2/2$

Ellipse: $K^2 = b^2/4$



$K^2 = (a^2+b^2)/4$

What if we want scale-invariance?

- Changing the scale of *f(x,y)* by (α,β) gives a new image:

$$f'(x,y) = f\left(x/\alpha, y/\beta\right)$$

- The transformed central moments

$$\mu'_{pq} = \alpha^{1+p}\beta^{1+q}\mu_{pq}$$

- If α=β, scale-invariant central moments are given by the normalization:

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}, \quad \gamma = \frac{p+q}{2} + 1, \quad p+q \geq 2$$

Symmetry

- To detect symmetry about center of mass, use central moments.

- For invariance of scale, use scale-normalised central moments
  - **(η_{11}, η_{20}, η_{02}, η_{21}, η_{12}, η_{30}, η_{03}).**
- Objects symmetric about either x or y axis will produce **$\eta_{11} = 0$**.
- Objects symmetric about y axis will give **$\eta_{12} = 0$** and **$\eta_{30} = 0$**.
- Objects symmetric about x axis will give **$\eta_{21} = 0$** and **$\eta_{03} = 0$.**
- X axis symmetry: **$\eta_{pq} = 0$ for all p = 0, 2, 4, … ; q = 1, 3, 5, …**

	η_{11}	η_{20}	η_{02}	η_{21}	η_{12}	η_{30}	η_{03}
'M'	0	+	+	-	0	0	-
'C'	0	+	+	0	+	+	0
'O'	0	+	+	0	0	0	0

Rotation invariant moments

Method 1:

Find principal axes of object, rotate and compute moments.

This can break down if object has no unique principal axes.

Rotation invariant moments

Method 2 : Hu moments

The method of absolute moment invariants:

This is a set of normalized central moment combinations, which can be used for scale, position, and rotation invariant pattern identification.

- For second order (p+q=2),
  there are two invariants/Hu moments:

$$\varphi_1 = \eta_{20} + \eta_{02} \qquad \varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

Third order Hu moments

- For third order moments, (p+q=3), the invariants are:

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\varphi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\varphi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\varphi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\varphi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

φ_7 is skew invariant, and may help distinguish between mirror images.

- These moments are not independent, and do not comprise a complete set.

Hu's moments; a bit simplified notation

For second order moments (p+q=2), two invariants are used:

$$\varphi_1 = \eta_{20} + \eta_{02}$$
$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

For third order moments, (p+q=3), we can use

$$a = (\eta_{30} - 3\eta_{12}), \qquad b = (3\eta_{21} - \eta_{03}),$$
$$c = (\eta_{30} + \eta_{12}), \quad \text{and} \quad d = (\eta_{21} + \eta_{03})$$

and simplify the five last invariants of the set:

$$\varphi_3 = a^2 + b^2$$
$$\varphi_4 = c^2 + d^2$$
$$\varphi_5 = ac[c^2 - 3d^2] + bd[3c^2 - d^2]$$
$$\varphi_6 = (\eta_{20} - \eta_{02})[c^2 - d^2] + 4\eta_{11}cd$$
$$\varphi_7 = bc[c^2 - 3d^2] - ad[3c^2 - d^2]$$

Hu moments of simple objects

- In the continuous case, the two first Hu moments of a binary rectangular object of size 2a by 2b, are given by

$$\phi_1 = \frac{1}{12}\left(\frac{a}{b}+\frac{b}{a}\right), \qquad \phi_2 = \left(\frac{1}{12}\right)^2\left(\frac{a}{b}-\frac{b}{a}\right)^2$$

  while the remaining five Hu moments are all zero.

- Similarly, the two first Hu moments of a binary elliptic object with semi-axes a and b, are given by

$$\phi_1 = \frac{1}{4\pi}\left(\frac{a}{b}+\frac{b}{a}\right), \qquad \phi_2 = \left(\frac{1}{4\pi}\right)^2\left(\frac{a}{b}-\frac{b}{a}\right)^2$$

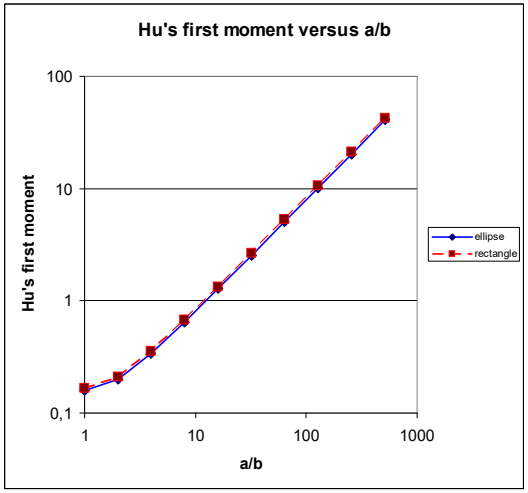
  while the remaining five Hu moments are all zero.

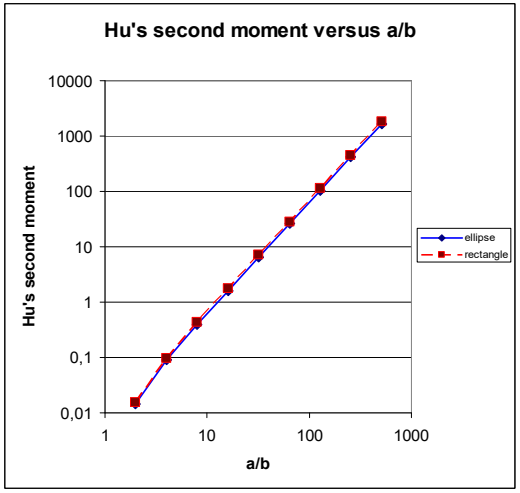
(See definitions of a, b, c and d and table of symmetry)

	η_{11}	η_{20}	η_{02}	η_{21}	η_{12}	η_{30}	η_{03}
'M'	0	+	+	-	0	0	-
'C'	0	+	+	0	+	+	0
'O'	0	+	+	0	0	0	0

Φ_1 and φ_2 versus a/b

- Only (φ1, φ2) are useful for these simple objects.
- Notice that even in the continuous case it may be hard to distinguish between an ellipse and its bounding rectangle using these two moments.
- Relative difference in φ_1 of ellipse and its object oriented bounding rectangle is constant, 4.5%.
- Relative difference in φ_2 of ellipse and its object oriented bounding rectangle is constant, 8.8%.
- Relative differences given above are also true when comparing an ellipse with a same-area rectangle having the same a/b ratio, regardless of the size and eccentricity of the ellipse.





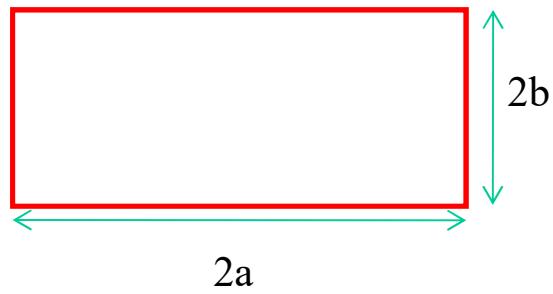
Moments as shape features

- The central moments are seldom used directly as shape descriptors.
- Major and minor axis, radius of gyration, and eccentricity are useful shape descriptors.
- Object orientation is normally not used directly, but to estimate rotation.
- The set of 7 Hu moments can be used as shape features. (Start with the first four, as the last half are often zero for simple objects).

Moments of inertia for simple shapes

- Rectangular object (2a×2b):

  $I_{20} = 4a^3b/3$, $I_{02} = 4ab^3/3$

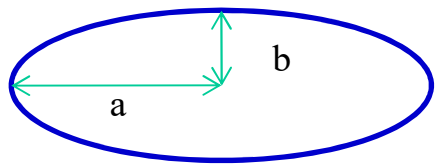


- Square (a×a):

  $I_{20} = I_{02} = a^4/12$

- Elliptical object, semi-axes (a,b):

  $I_{20} = \pi a^3b/4$, $I_{02} = \pi ab^3/4$



- Circular object, radius R:

  $I_{20} = I_{02} = \pi R^4/4$

Moments of an ellipse

- Assume that the ellipse has semimajor and semiminor axes (*a,b*), a>b. An ellipse where major axis is along x-axis is given by

$$(x/a)^2 + (y/b)^2 = 1 \;\Rightarrow y = \pm\frac{b}{a}\sqrt{a^2 - x^2}$$

The largest second order central moment (here called I_{20}) is given by

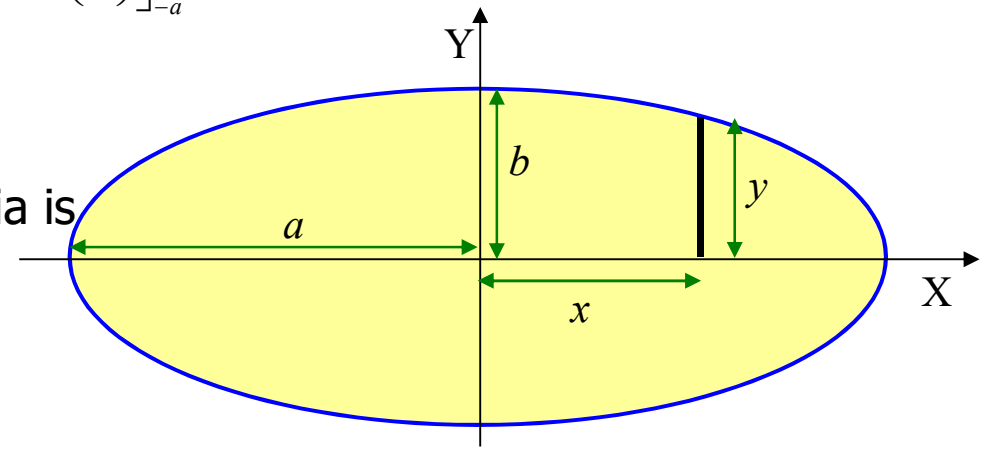
$$I_{20} = 2\int_{-a}^{a} x^2 y\,dx \;= 2\frac{b}{a}\int_{-a}^{a} x^2\sqrt{a^2 - x^2}\,dx$$

$$I_{20} = 2\frac{b}{a}\left[\frac{x}{8}\left(2x^2 - a^2\right)\sqrt{a^2 - x^2} + \frac{a^4}{8}\sin^{-1}\left(\frac{x}{a}\right)\right]_{-a}^{a}$$

$$I_{20} = 2\frac{b}{a}\left[\frac{a^4}{8}\left(\frac{\pi}{2} + \frac{\pi}{2}\right)\right] = \underline{\underline{\frac{\pi}{4}a^3 b}}$$

Similary, the smallest moment of inertia is

$$I_{\min} = \underline{\underline{\frac{\pi}{4}a\;b^3}}$$



Grayscale contrast invariants

- Abo-Zaid *et al.* have defined a normalization that cancels both scaling and contrast.
- The normalization is given by

$$\eta'_{pq} = \frac{\mu_{pq}}{\mu_{00}} \left(\frac{\mu_{00}}{\mu_{20} + \mu_{02}} \right)^{\frac{(p+q)}{2}}$$

- This normalization also reduces the dynamic range of the moment features, so that we may use higher order moments without having to resort to logarithmic representation.
- Abo-Zaid's normalization cancels the effect of changes in contrast, but not the effect of changes in intensity:

$$f'(x, y) = f(x, y) + b$$

- In practice, we often experience a combination:

$$f'(x, y) = cf(x, y) + b$$

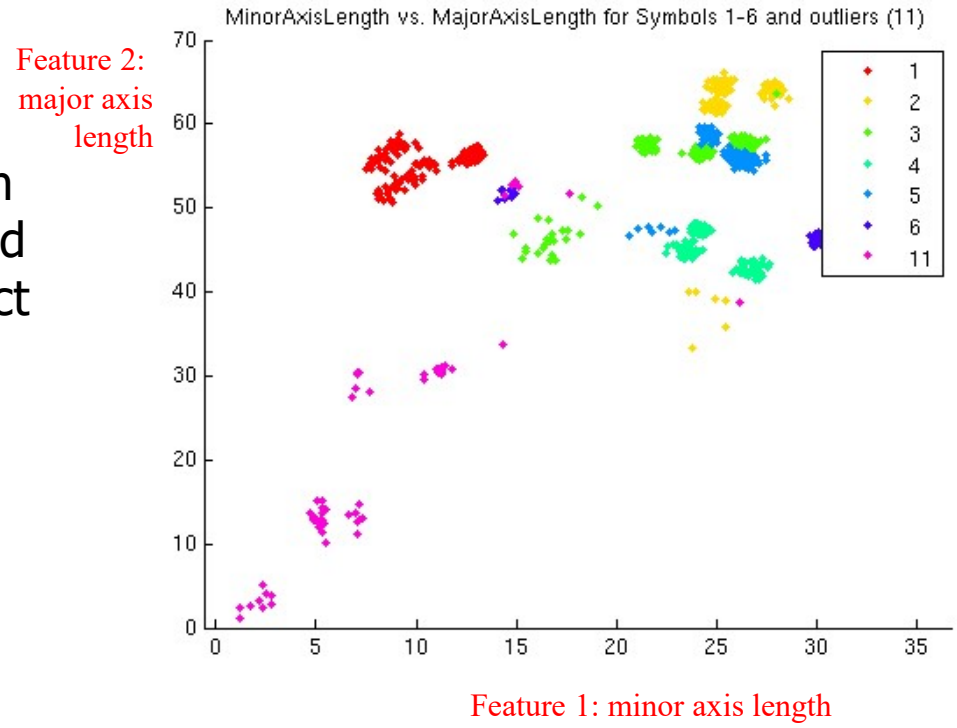
From features to discrimination between objects

- The following slide introduces simple tools like scatter plots to visualize how good a feature (or combination of 2-3 features) is in separating objects of different types/classes.

- To evaluate features, we use training data consisting of objects with KNOWN CLASS.

Scatter plots

- A 2D scatter plot is a plot of feature values for two different features. Each object's feature values are plotted in the position given by the features values, and with a class label telling its object class.
- Matlab: gscatter(feature1, feature2, labelvector)
- Classification is done based on more than two features, but this is difficult to visualize.
- Features with good class separation show clusters for each class, and different clusters should ideally be separated.



The "curse-of-dimensionality"

- Also called "peaking phenomenon".
- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.
- The implication is that:
- For a high measurement complexity, we will need large amounts of training data in order to attain the best classification performance.
- => 5-10 samples
  per feature per class.

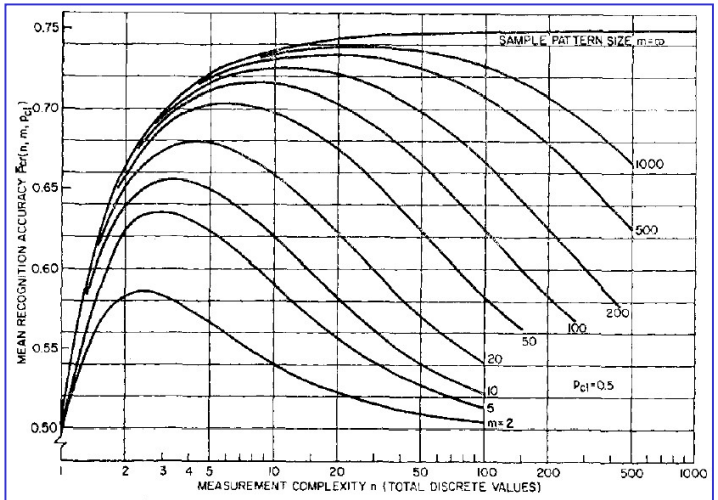


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

Illustration from G.F. Hughes (1968).

Finding best feature subset

- The goal: to find the subset of observed features which
  - best characterizes the differences between groups
  - is similar within the groups
  - Maximize the ratio of between-class and within-class variance.
- If we want to perform an <u>exhaustive search</u> through D features for the optimal subset of the d ≤ m "best features", the number of combinations to test is

$$n = \sum_{d=1}^{m} \frac{D!}{(D-d)!\,d!}$$

- Impractical even for a moderate number of features!

  d ≤ 5, D = 100 => n = 79.374.995

- There exist several <u>sub-optimal schemes</u> to search for the best sub-set.

A simulation study design

- *H.* Schulerud and F.Albregtsen: *"Many are called, but few are chosen. Feature selection and error estimation in high dimensional spaces",* Computer Methods and Programs in Biomedicine, 73, 91—99, 2004.

- Monte Carlo study, averaging 100 simulations per setting
- 2 classes, normally distributed, common covariance
- Up to 500 feature candidates
- Only 5 features are different between the classes

  For these 5, squared difference of class means = $\delta^2/\sqrt{5}$; $\delta^2 = 0, 1, 4$

  the rest of the continuous distributions are EQUAL!
- Stepwise forward-backward feature selection
- 20 - 1000 training samples
- 20 - 1000 test samples

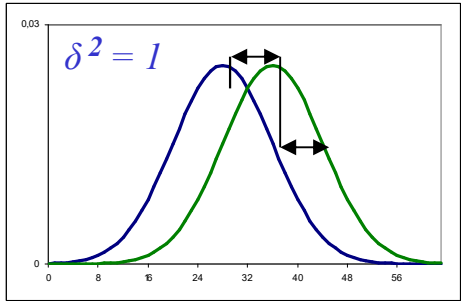
Probabilistic distance measures

- If the class-conditional probability distributions are Gaussian:

$$p(\xi \mid \omega_i) = \left[(2\pi)^d |\Sigma_i|\right]^{-1/2} \exp\left\{-\frac{1}{2}(\xi - \mu_i)^T \Sigma_i^{-1} (\xi - \mu_i)\right\}$$

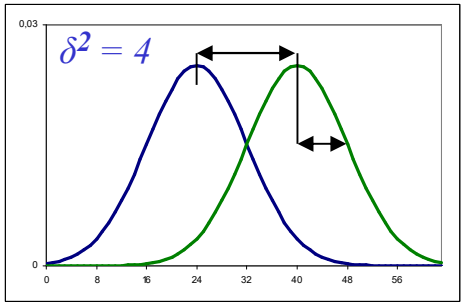
where μ_i and Σ_i are the mean vector and the covariance matrix of the *i*-th class distribution; the **Mahalanobis** distance is

$$\delta^2 = (\mu_2 - \mu_1)^T \Sigma^{-1} (\mu_2 - \mu_1), \ \ if\ \Sigma_1 = \Sigma_2 = \Sigma$$

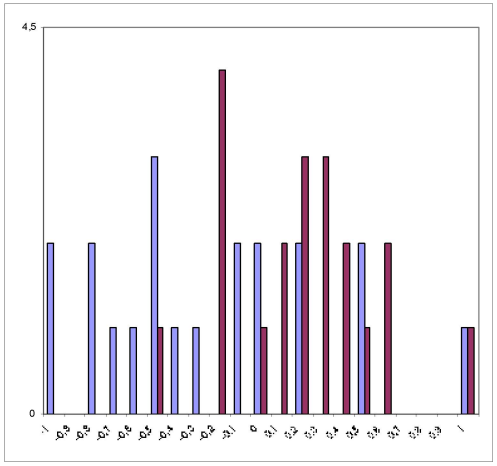


- The Bhattacharyya distance may also be useful:

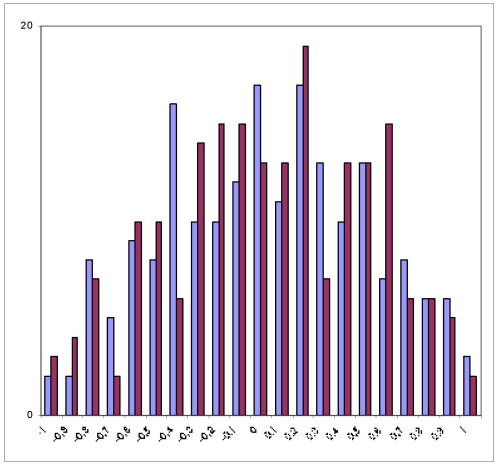
$$J_B = \frac{1}{4}(\mu_2 - \mu_1)^T [\Sigma_1 - \Sigma_2]^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \left[\frac{\left|\frac{1}{2}(\Sigma_1 + \Sigma_2)\right|}{\sqrt{|\Sigma_1||\Sigma_2|}} \right]$$



Samples from distributions



*Distribution of 2 independent sets of **20** samples from standardized normal distributions, $\delta^2 = 0$.*

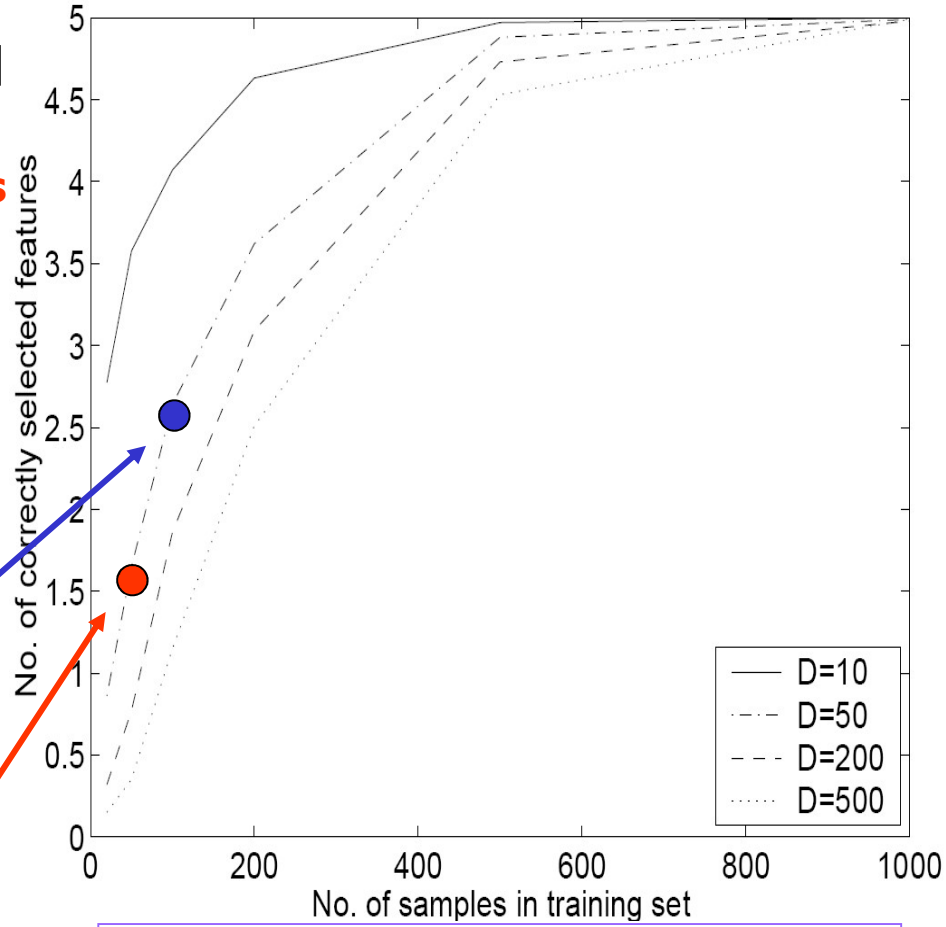


*Distribution of 2 independent sets of **200** samples from standardized normal distributions, $\delta^2 = 0$.*

- For small sample sets and small class distances, observations may indicate a separation of classes, while no real difference exists !!!

Simulation results - Feature selection II

- The number of correctly selected features increases with
  - **increasing # of training samples**
  - **decreasing # of candidates**
  - **(increasing class distance)**
- For small sample sizes the number of *candidates* features is of great importance :
  - For D = 50 and $\delta^2 = 1$, half of the 5 selected features will be noise if $n_{Tr} = 100$.
  - For D = 50, $\delta^2 = 1$, $n_{Tr} = 50$, **60% of the selected features will be noise!**
  - **So, Be Careful !!!**



The average number of correctly selected features, when the class Mahalanobis distance $\delta^2 = 1$.

Caveats of Cross Validation

- A simple simulation may demonstrate the effect of performing feature selection before a cross validation to estimate classification performance on the same data.

- If the classes are overlapping, the number of training samples is small, and the number of feature candidates are high, the common approach of performing feature selection before leave-one-out error estimation on the same data results in a highly biased error estimate of the true error.

- Performing feature selection and leave-one-out error estimation in one process gives an unbiased error estimate, but with high variance.

- See Figure 7 of

  H Schulerud and F Albregtsen, *"Many are called, but few are chosen. Feature selection and error estimation in high dimensional spaces",* Computer Methods and Programs in Biomedicine 73, 91—99, 2004.

Learning goals – object description

- Invariant topological features
- Projections and signatures – use and limitations
- Geometric features
  - Area, perimeter and circularity/compactness
  - Bounding boxes

- Moments, binary and grayscale
    - Ordinary moments and central moments
  - Moments of objects, object orientation, and best fitting ellipse
    - Focus on first- and second-order moments.
  - Invariance may be important

- Inspection of feature scatter plots
- Select your feature set with great care ! Validate correctly!

IN 5520
7.10.20
Introduction to classifiction

Anne Solberg (anne@ifi.uio.no)

- Introduction to classification

Based on handout from Pattern Recognition by Theodoridis

Lectures on zoom

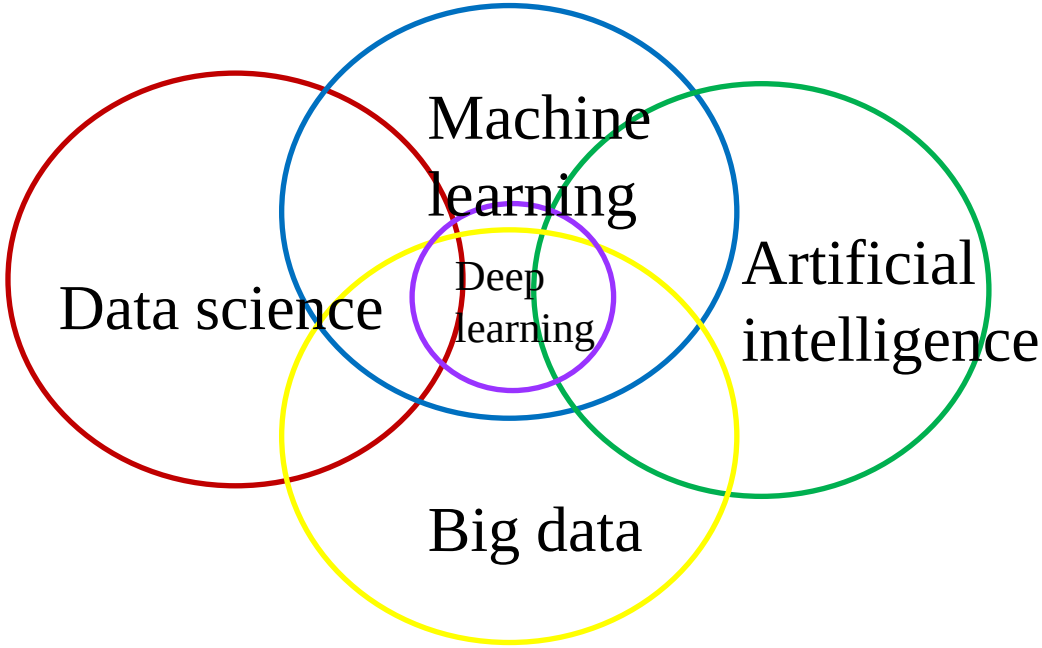
- The lectures will be recorded
- Don’t let this stop you to ask questions!!!
- During recording – ask your questions in chat
- At intervals marked «Questions?» I will repeat and answer these
- We take a short break in recording at halftime
  - This prevents the file to be too big
  - Ask oral questions during the break

- We are also open for oral questions after the lecture

Plan for this lecture:

- Introduction to machine learning/classification
- Explain the relation between thresholding and classification with 2 classes
- Background in probability theory
- Bayes rule
- K-nearest Neighbor classifier
- Classification with a Gaussian density and a single feature
  - Linear boundaries in feature space
- Minimum distance classifier
- Briefly: training and testing a classifier

Introduction to machine learning



The term **machine learning** is today used broadly and cover linear regression, statistical classification, neural networks ++

Keyworks: approaces that learn from data

Introduction to classification

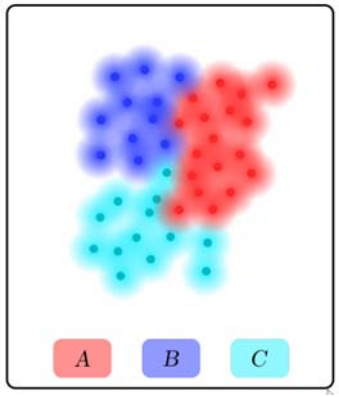
- Given a feature vector **x** computed from a data set
- We want to predict the value of an unkown variable y given **x**.
- If y is continuous, the simplest model could be linear regression.
- If y is categorical, we want to assign **x** into a set of predefined classes c=1,...C, and y=c if **x** belongs to class c.
- A classification problem can be solved using supervised, unsupervised or reinforcement learning.

Unsupervised learning

- Our training set consists of input $\mathbf{x}$ only:

- We do not have any labeled data. Our goal is to find an underlaying structure of the data.

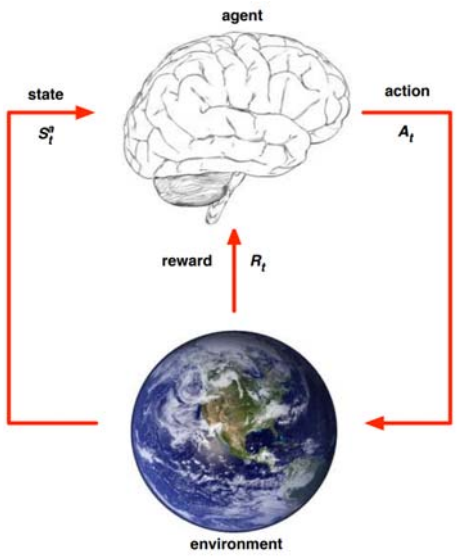
- Examples:
  - Data clustering
  - Anomality detection
  - Signal generation
  - Signal compression



Variational autoencoder (latent space z)

Reinforcement learning

- Reinforcement Learning ~ Science of decision making
- In RL an agent learns from the experiences it gains by interacting with the environment.
- The goal is to maximize an accumulated reward given by the environment.
- An agent interacts with the environment via states, actions and rewards.
- Example: games like chess or black jack.



RL Course by David Silver

Supervised learning

- Given a training set with input $\mathbf{x}$ and desired output y:

- The goal is to create a function f that "approximates" this mapping:

- Hope that this generalizes well to unseen examples:

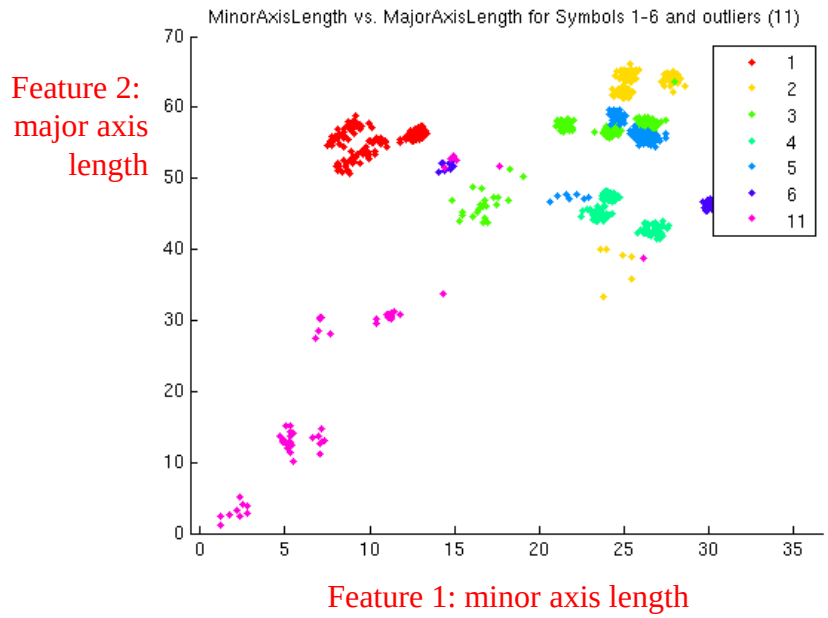
Examples:

- Classification, regression, object detection,
- Segmentation, image captioning.

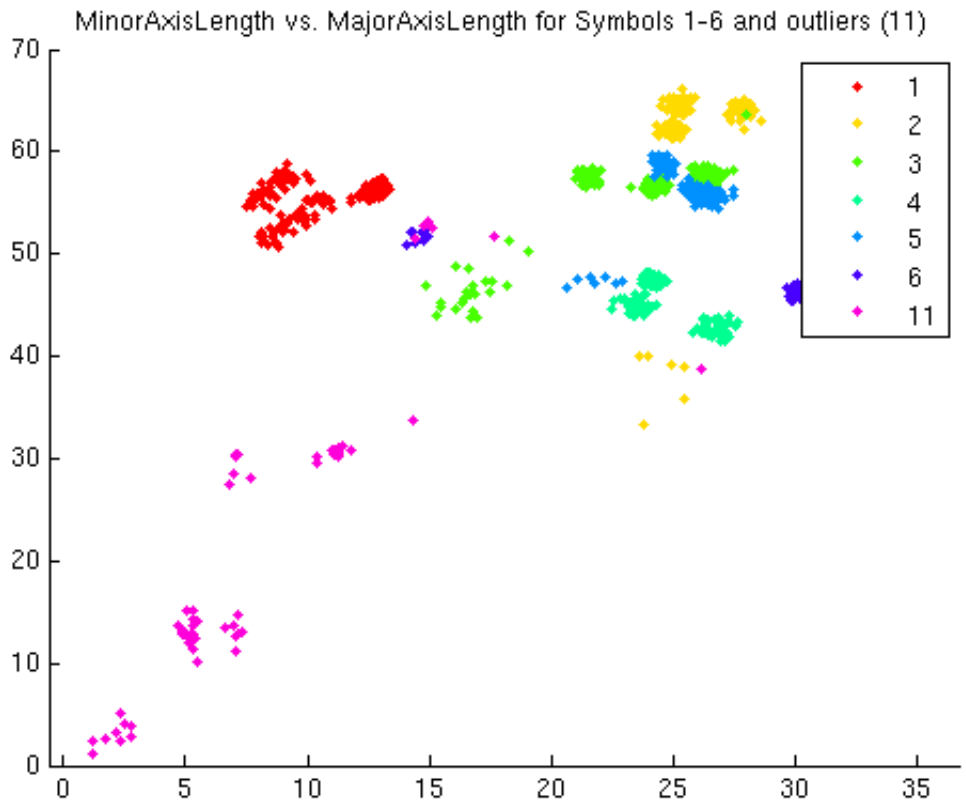
Supervised learning is normally used for image classification and the coming lectures are about it.

Scatter plots - reminder

- A 2D scatter plot is a plot of feature values for two different features. Each object's feature values are plotted in the position given by the features values, and with a class label telling its object class.
- A scatter plott visualize the space spanned by 2 or more features: called the **feature space**
- Matlab: gscatter(feature1, feature2, labelvector)
- Classification is done based on *more than two features*, but this is difficult to visualize.
- Features with good class separation show clusters for each class, but different clusters should ideally be separated.



Discriminating between classes in a scatter plot



This Photo by Unknown Author is licensed under CC BY-SA-NC

Concepts in classification

- In the following lectures we will cover these topics related to classification:
  - Training set
  - Validation set
  - Test set
  - Classifier accuracy/confusion matrices.
  - Computing the probability that an object belongs to a class.
  - Bayes rule
  - Discriminant functions/Decision boundaries
  - Normal distribution, mean vector and covariance matrices
  - kNN classification
  - Support vector machines
  - Unsupervised classification/clustering
  - Dimensionality reduction

Training data set

- A set of known objects from all classes for the application.
- Typically a high number of samples from each class.
  - Approximately equal number of samples from each class if possible
- Used to **determine the parameters of the given classification algorithm**.
  - Example 1: fit a Gaussian density to the data
  - Example 2: find a Support Vector Model that fits the data
  - Example 3: find the weights in a neural net

Validation data set

- A set of known objects from all classes for the application.
- Typically a sufficiently high number of samples from each class.
- Used to **optimize hyperparameters**, e.g.
  - Compare feature selection algorithms
  - Compare classifiers
  - Compare network architectures

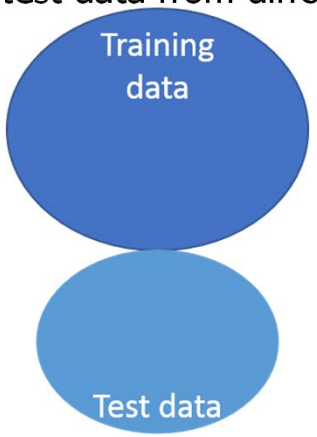
Test data set

- A set of known objects from all classes for the application.
- Typically a high number of samples from each class.
- Used ONCE to **estimate the accuracy of a trained model**.
- Should never optimize any parameters on the test data set.

Challenge: domain shift

- Training and test data from same distribution

Training and test data from different distributions

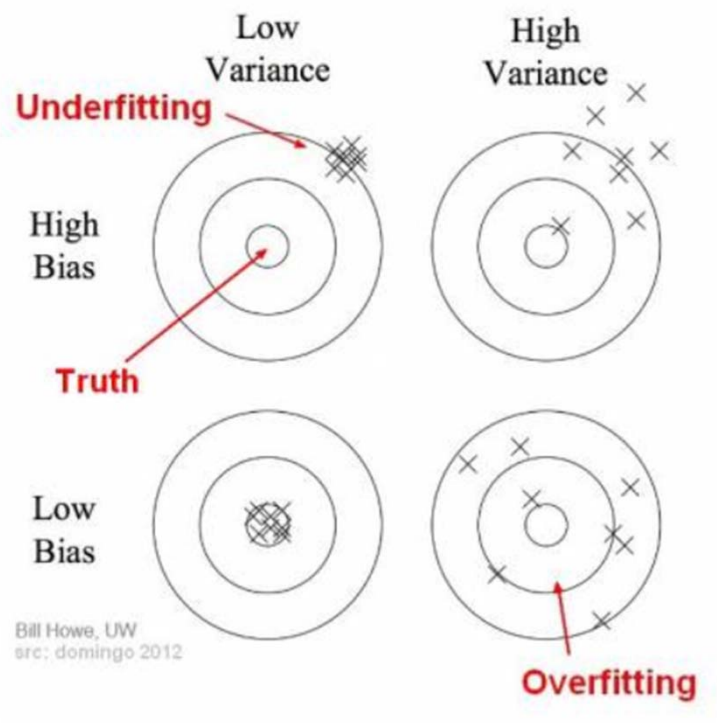


Generalization describes how well a classifiier is expected to perform on new data.

- Ohen ghoor!

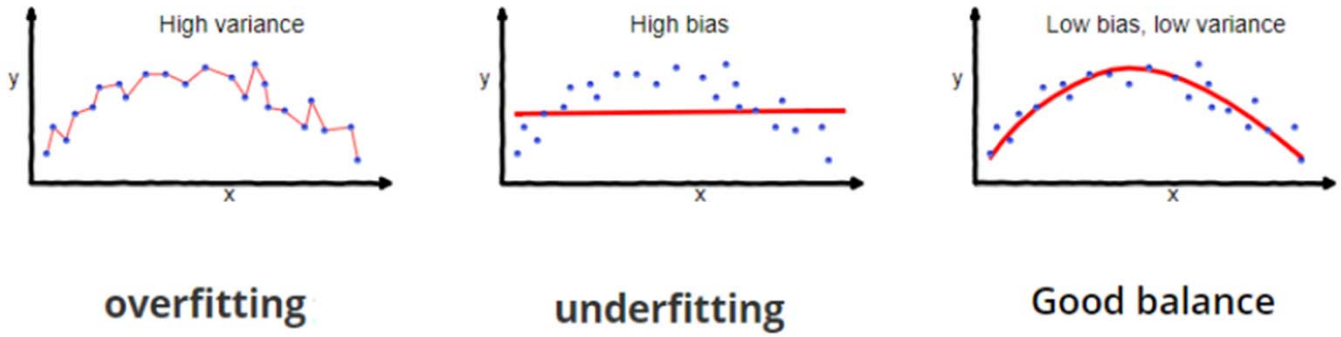
Bias-Variance tradeoff

- Given a model we fit to the data, e.g Y=f(X)+e
  Y: true values, X: measurements, e: error
  Estimate of Y: $\widehat{f(x)}$
- Bias: the difference between the average prediction and the correct value $E\left(\widehat{f(x)}\right) - f(x)$
- Variance: The variance of the predictions $\widehat{f(x)}$

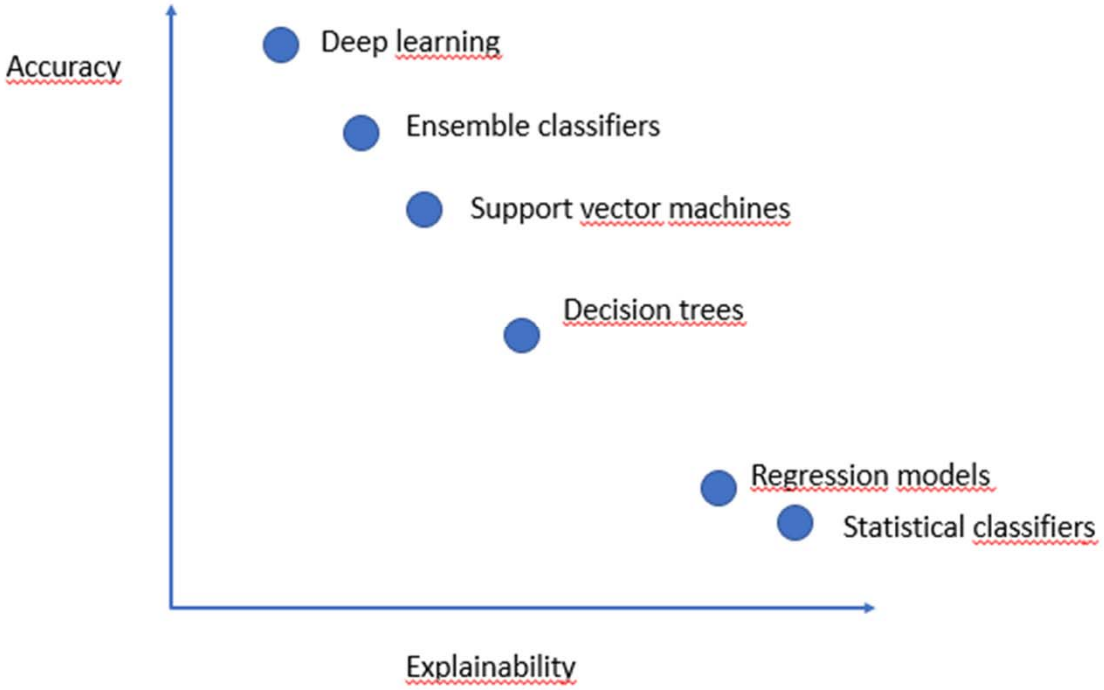


Underfitting and overfitting

- Underfitting: the model is not able to capture the underlying pattern of data
  - High bias, low variance
  - In our setting: high error rate on training data and validation data
- Overfitting: the model captures the noise with the underlying pattern of data
  - Low bias, high variance
  - In our setting: low error rate on training data, high error on validation data



Explaining the result



Always use the simplest model that can perform well on your validation data!

Explaining the decisions

- Where we are currently using deep learning:

- This is a cat

- Goal:

- This is a cat (p=0.87)
- It has fur, whiskers, claws.
- Most characteristic feature:

QUESTIONS
ANSWERS

Thesholding as a binary classification problem

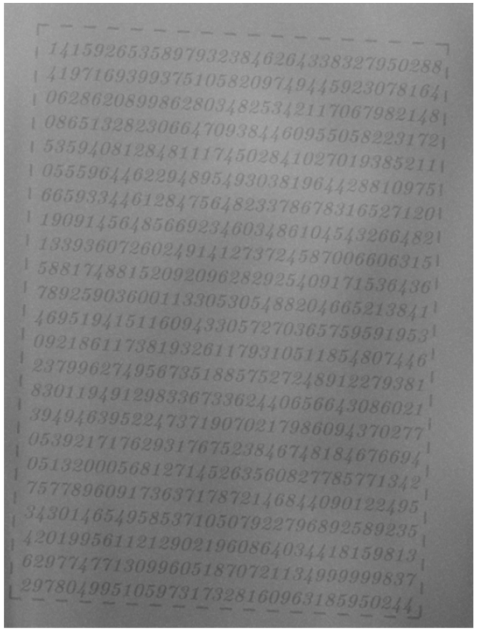
Gray level of foreground and background pixels modelled as probability density functions

From INF2310: Thresholding (Gonzalez and Woods 10.3)

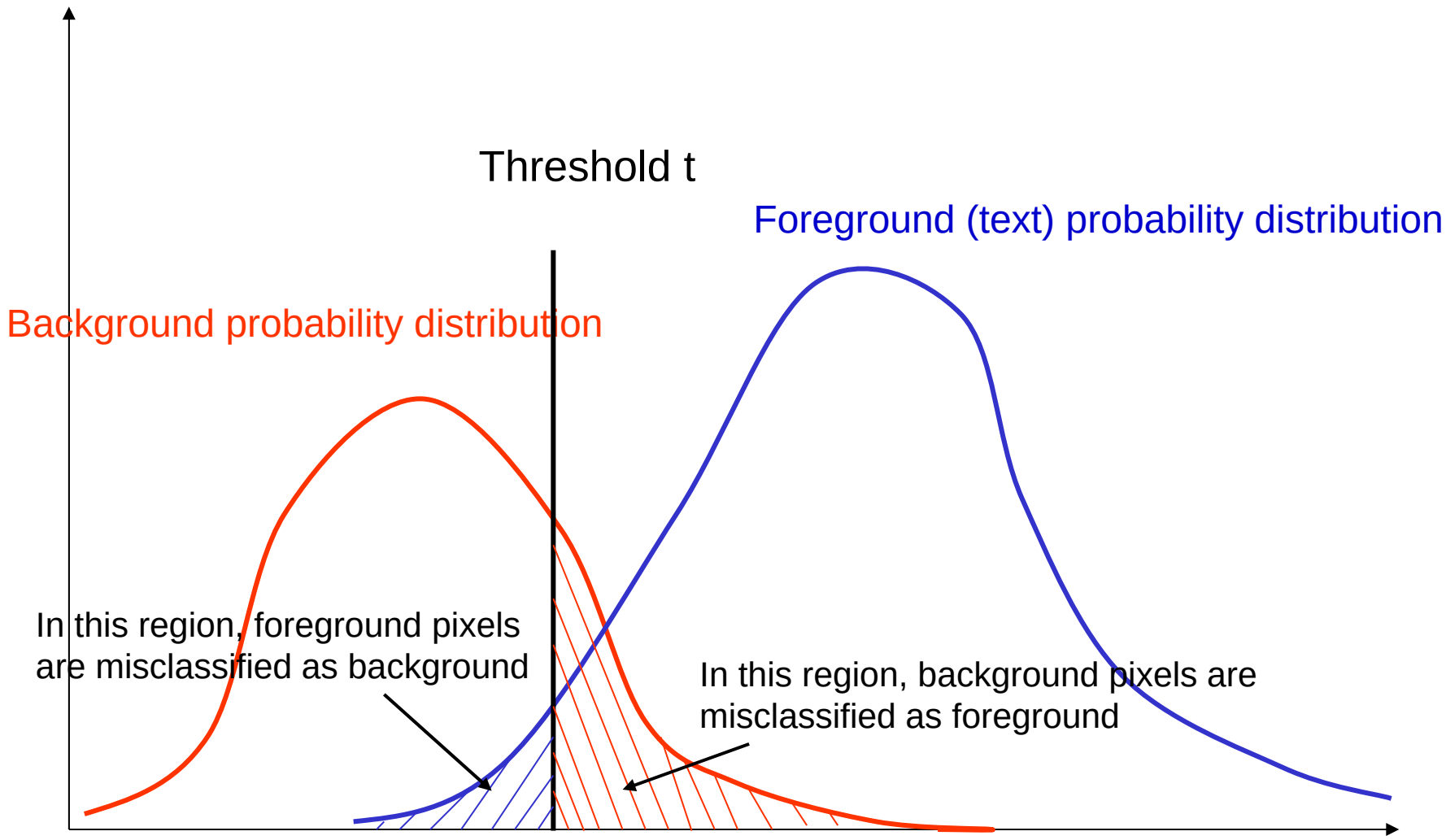
- Basic thresholding assigns all pixels in the image f(x,y) to one of 2 classes: foreground or background

$$g(x,y) = \begin{cases} 0 \text{ if } & f(x,y) \le T \\ 1 \text{ if } & f(x,y) > T \end{cases}$$

- g(x,y) is the thresholded image
- This can be seen as a 2-class classification problem based on **a single feature, the gray level.**
- The 2 classes are background and foreground, and the threshold T defines the border between them.



Classification error for thresholding

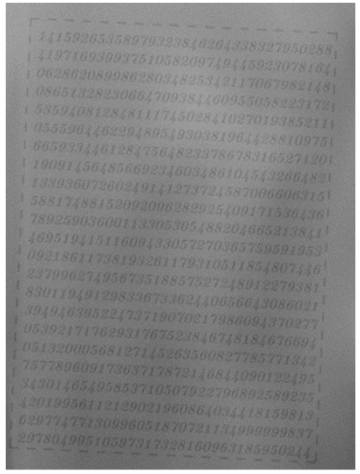


Histograms

- h(i)=number of pixels with gray level i
- $\sum_{i=0}^{NxM} h(i) = NxM\ for\ a\ NxM\ image$
- Normalized hisogram p(i)=h(i)/(NxM)
- For a normalized histogram
- $\sum_{i=0}^{NxM} p(i) = 1$

Classification error for thresholding

- We assume that b(z) is the normalized histogram for background *b(z)* and *f(z)* is the normalized histogram for foreground.
- The histograms are estimates of the probability distribution of the gray levels in the image.
- Let *F* and *B* be the prior probabilities for background and foreground(*B+F=1*)
- The normalized histogram for the image is then given by



$$p(z) = B \cdot b(z) + F \cdot f(z)$$

- The probability for misclassification given a treshold t is:

$$E_B(t) = \int_{-\infty}^{t} f(z)dz$$

$$E_F(t) = \int_{t}^{\infty} b(z)dz$$

Find T that minimizes the error

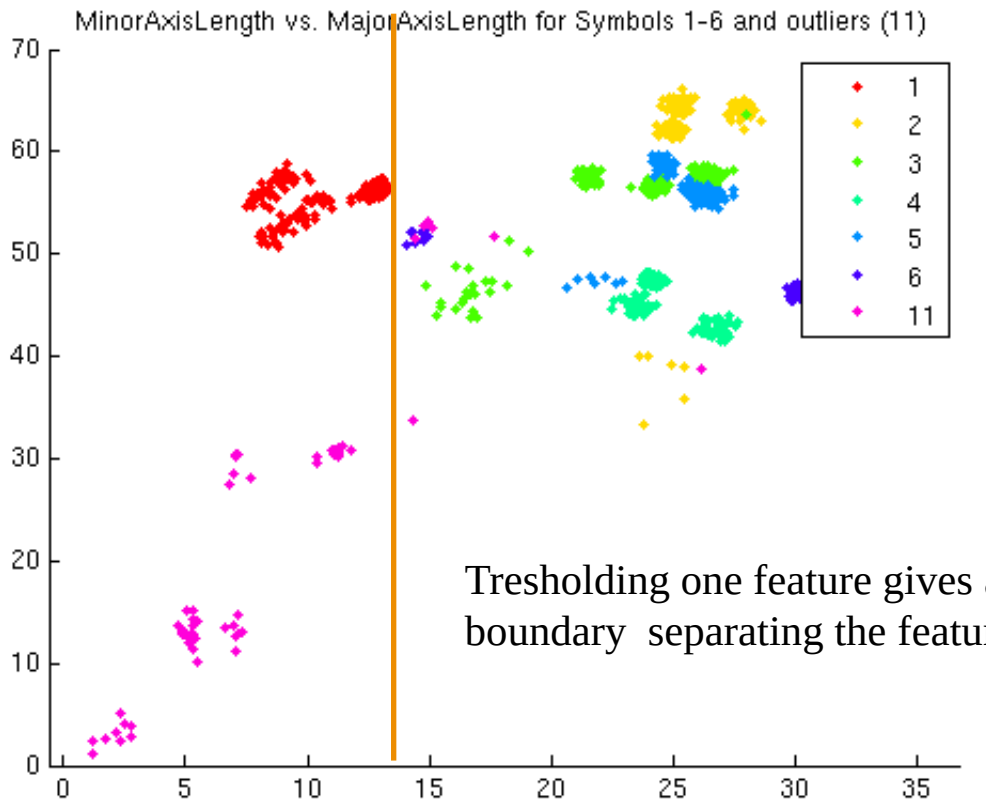
$$E(t) = F\int_{-\infty}^{t} f(z)dz + B\int_{t}^{\infty} b(z)dz$$

$$\frac{dE(t)}{dt} = 0 \Rightarrow F \cdot f(T) = B \cdot b(T)$$

Minimum error is achieved by setting T equal to the point where the probabilities for foreground and background are equal.

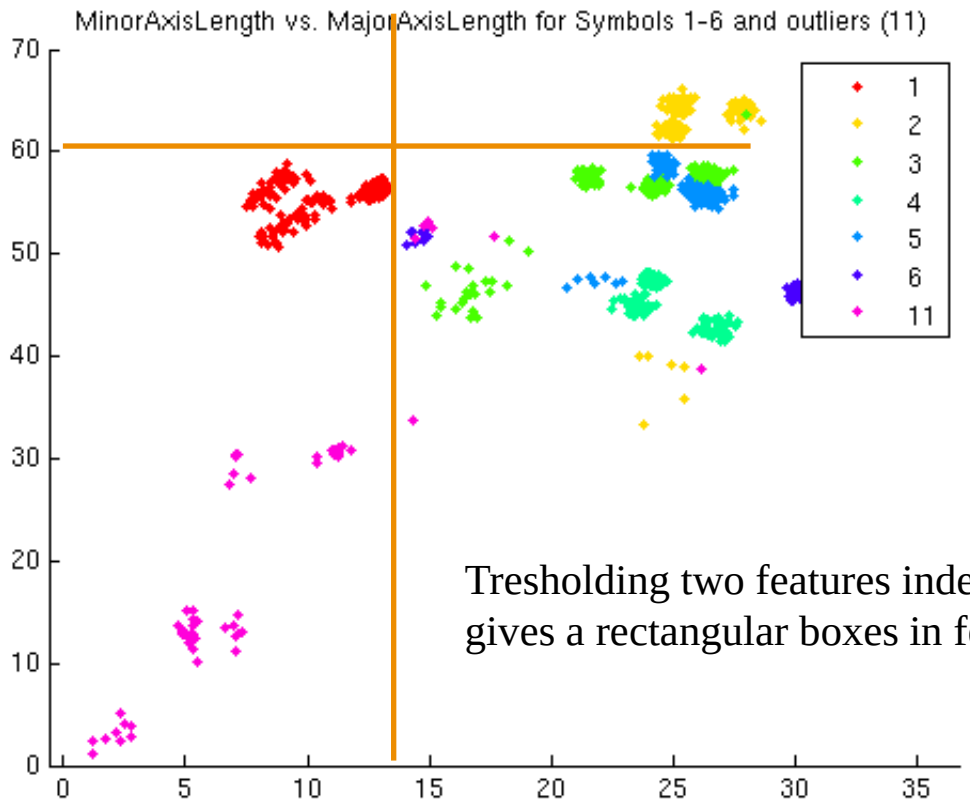
The locations in feature space where the probabilities are equal is called decision boundary in classification.
The goal of classification is to find the boundaries.

Partitioning the feature space using thresholding – 1 feature and 1 threshold



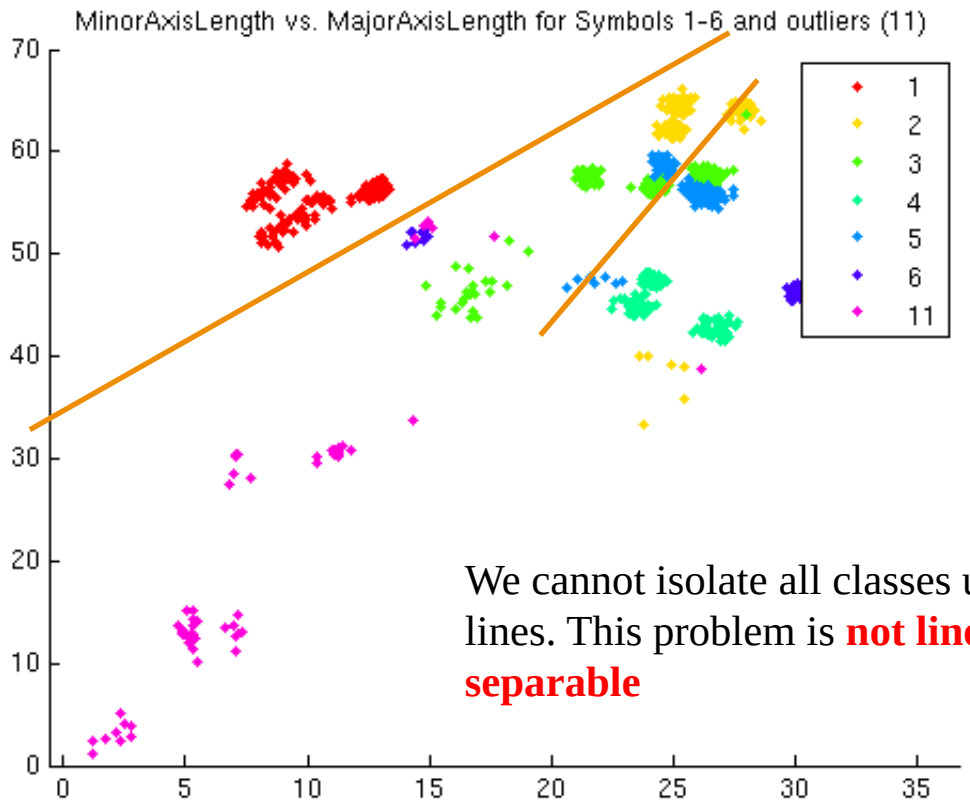
Tresholding one feature gives a linear boundary separating the feature space

Partitioning the feature space using thresholding – 2 features and 2 thresholds



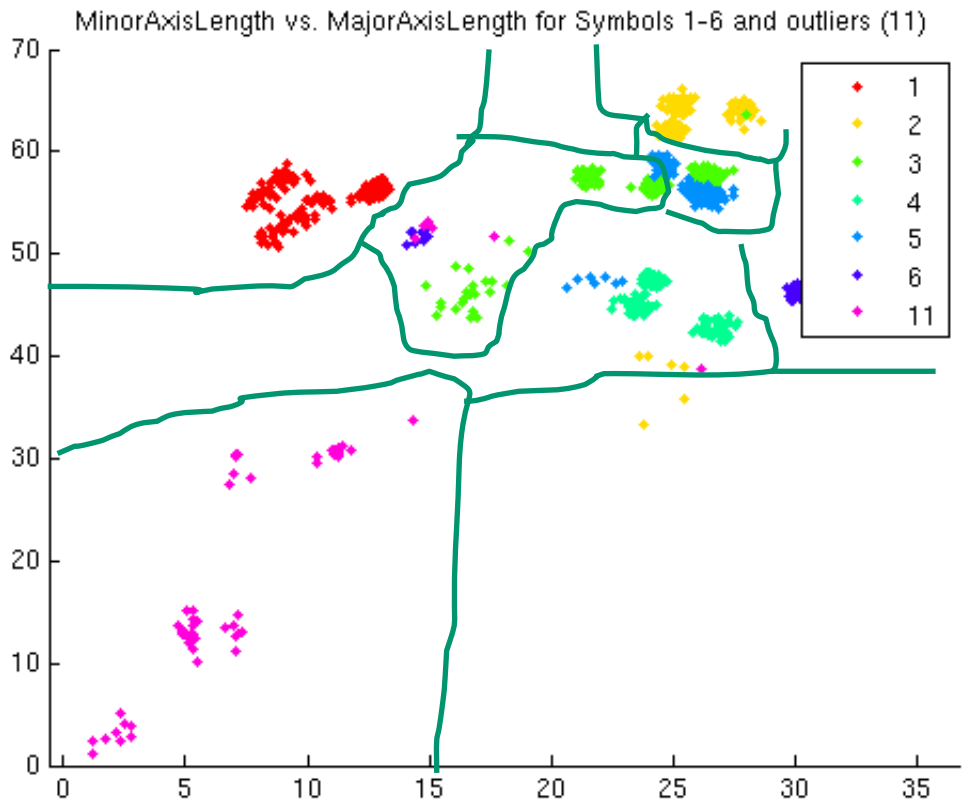
Tresholding two features independently gives a rectangular boxes in feature space

Can we find a line with better separation?



We cannot isolate all classes using straight lines. This problem is **not linearly separable**

The goal of classification: partitioning the feature space with smooth boundaries



QUESTIONS
ANSWERS

Types of classifiers

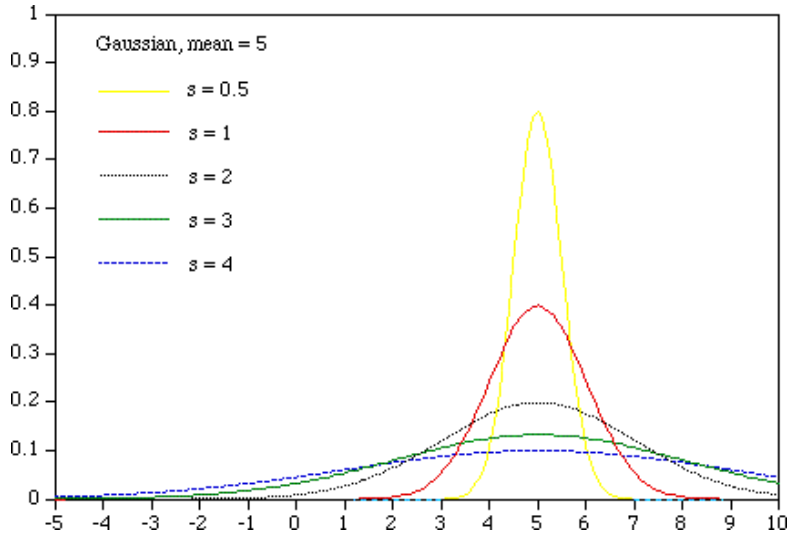
- Parametric probability distribution:
  - Assume a type of probability distribution, fit the parameters of the distribution to the data.
- Non-parametric probability distribution:
  - Estimate the distribution locally using e.g. Parzen windows
  - K Nearest Neighbor classifier
- Minimize loss function e.g. separation between classes
  - Support Vector Machines: maximize the distance between the closest points in two classes
  - Multilayer feed-forward network: minimize the cross entropy between the true labels and the estimated labels

Distributions, standard deviation and variance

- A univariate (one feature) Gaussian distribution (normal distribution) is specified given the mean value μ and the variance σ^2:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Variance σ^2, standard deviation σ



Two Gaussian distributions for thresholding a single feature

- Assume that *b(z)* and *f(z)* are Gaussian distributions, then

$$p(z) = \frac{B}{\sqrt{2\pi\sigma_B^{\ 2}}} e^{-\frac{(x-\mu_B)^2}{2\sigma_B^{\ 2}}} + \frac{F}{\sqrt{2\pi\sigma_F^{\ 2}}} e^{-\frac{(x-\mu_F)^2}{2\sigma_F^{\ 2}}}$$

- μ_B and μ_F are the mean values for background and foreground.
- σ_B^2 and σ_F^2 are the variance for background and foreground.

The 2-class classification problem summarized

- Given two Gaussian distributions *b(z)* and *f(z)*.
- The classes have prior probabilities F and B.
- Every pixel should be assigned to the <u>class that minimizes the classification error.</u>
- The classification error is minimized at the point where F f(z) = B b(z).

- What we will do now is to generalize to K classes and D features.

The goal of classification

- We estimate the decision boundaries (equivalent to the threshold for multivariate data) based on training data.
- Classification performance is always estimated on a separate ”test” data set.
  - We try to measure the generalization performance.
- The classifier should perform well when classifying new samples
  - Have lowest possible classification error.
- We often face a tradeoff between classification error on the training set and generalization ability when determining the complexity of the decision boundary.

Bayesian decision theory

- A fundamental statistical approach to pattern classification.
- Named after Thomas Bayes (1702-1761), an english priest and matematician.
- It combines prior knowledge about the problem with a probability distribution function.
- The most central concept (for us) is Bayes decision rule.

Bayes rule for a classification problem

- Suppose we have J, j=1,…J classes.
- ω_j is the class label for a pixel,
  x is the observed gray level/feature vector.
  - We can classify a single pixel, an object, or an entire image
- We can use Bayes rule to find an expression for the class with the highest probability:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)}$$

$$\text{posterior probability} = \frac{\textit{likelihood} \times \text{prior probability}}{\text{normalizing factor}}$$

- For thresholding, $P(\omega_j)$ is the prior probability for background or foreground. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).
- Small p means a probability distribution
- Capital P means a probability (scalar value between 0 and 1)

Bayes rule explained

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)}$$

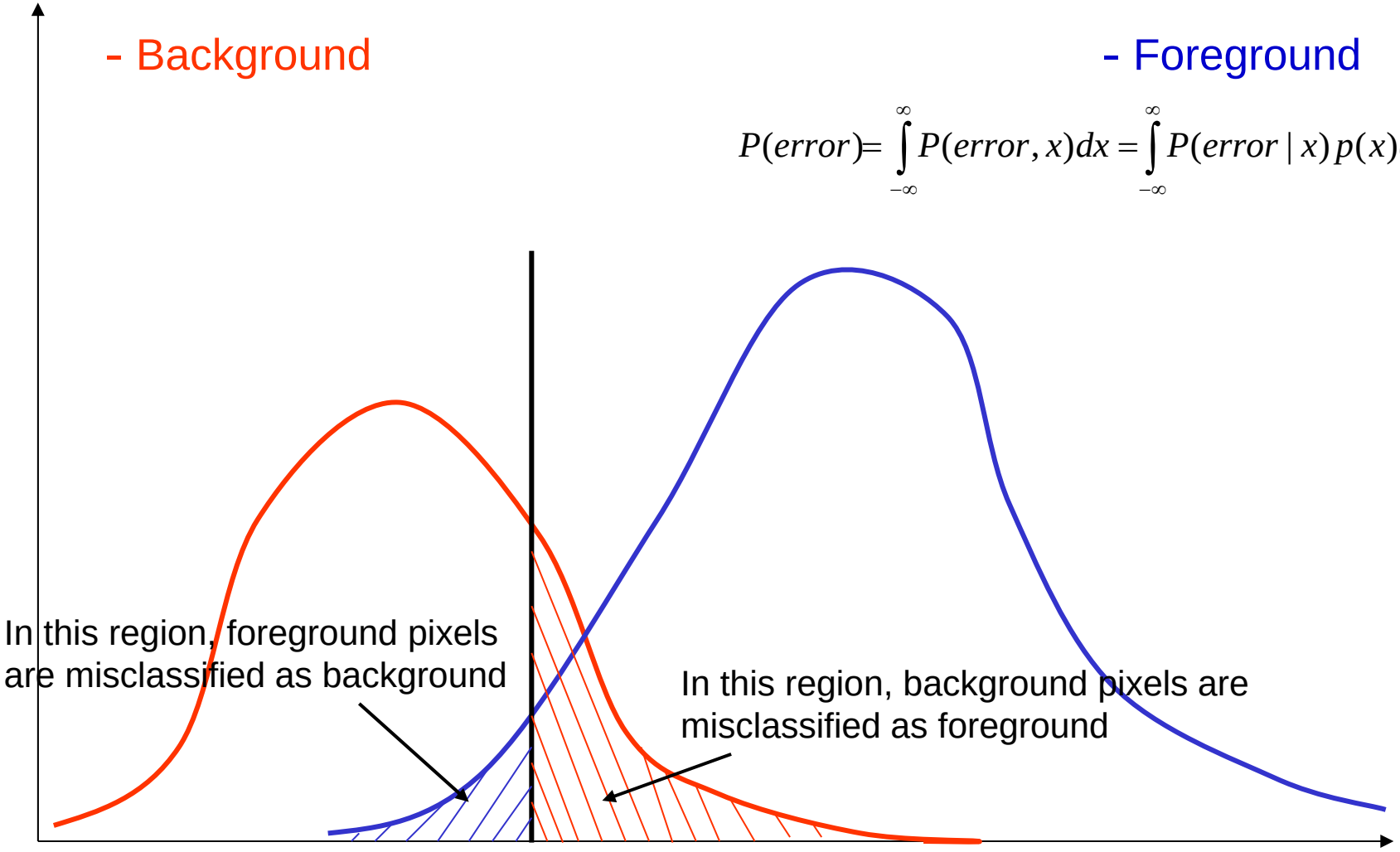
- $p(x|\omega_j)$ is the probability density function that models the likelihood for observing gray level x if the pixel belongs to class ω_j.
  - Typically we assume a type of distribution, e.g. Gaussian, and the mean and covariance of that distribution is fitted to some data that we know belong to that class. This fitting is called classifier training.
- $P(\omega_j|x)$ is the posterior probability that the pixel actually belongs to class ω_j. We will soon se that the the classifier that achieves the minimum error is a classifier that assigns each pixel to the class ω_j that has the highest posterior probability.
- p(x) is just a scaling factor that assures that the probabilities sum to 1.

Probability of error

- If we have 2 classes, we make an error either if we decide ω_1 if the true class is ω_2 , and if we decide ω_2 if the true class is ω_1.
- If $P(\omega_1|x) > P(\omega_2|x)$ we have more belief that x belongs to ω_1, and we decide ω_1.
- The probability of error is then:

$$P(error \mid x) = \begin{cases} P(\omega_1 \mid x) \text{ if we decide } \omega_2 \\ P(\omega_2 \mid x) \text{ if we decide } \omega_1 \end{cases}$$

Back to classification error for thresholding



$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$

Minimizing the error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)dx = \int_{-\infty}^{\infty} P(error \mid x)\, p(x)dx$$

- When we derived the optimal threshold (INF 2310), we showed that the minimum error was achieved for placing the threshold (or *decision boundary* as we will call it now) at the point where

$$P(\omega_1|x) = P(\omega_2|x)$$

- This is still valid.

Bayes classification with J classes and D features

- How do we generalize:
  - To more the one feature at a time
  - To J classes
  - To consider loss functions (that some errors are more costly than others)

Bayes rule with J classes and d features

- If we measure d features, ***x*** will be a d-dimensional feature vector.
- Let $\{\omega_1,\ldots,\omega_J\}$ be a set of J classes.
- The posterior probability for class j is now computed as

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j)P(\omega_j)}{p(\mathbf{x})}$$

$$p(x) = \sum_{j=1}^{c} p(\mathbf{x} \mid \omega_j)P(\omega_j)$$

- Still, we assign a pixel with feature vector x to the class that has the highest posterior probability:

$$\text{Decide } \omega_1 \text{ if } \mathrm{P}(\omega_1 \mid \mathbf{x}) \geq P(\omega_\mathrm{j} \mid \mathbf{x}), \text{ for all } j \neq i$$

Feature space and decision regions

- The minimum error criterion (or a different classifier criterion) partitions feature space into disjoint region R_i.
- The **decision surface** separates the regions in multidimensional space.
- For minimum error, the equation for the decision surface is

$$P(\omega_j \mid \mathbf{x}) = P(\omega_k \mid \mathbf{x})$$

Exactly where the threshold was set in minimum error thresholding!

Discriminant functions

- The decision rule

$$\text{Decide } \omega_1 \text{ if } P(\omega_1 \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x}), \text{ for all } j \neq i$$

can be written as assign **x** to ω_1 if

$$g_i(\mathbf{x}) > g_j(\mathbf{x})$$

- The classifier computes J discriminant functions $g_i(\mathbf{x})$ and selects the class corresponding to the largest value of the discriminant function.
- Since classification consists of choosing the class that has the largest value, a scaling of the discriminant function $g_i(\mathbf{x})$ by $f(g_i(\mathbf{x}))$ will not effect the decision if f is a monotonically increasing function.
- This can lead to simplifications as we will soon see.

Equivalent discriminant functions

- The following choices of discriminant functions give equivalent decisions:

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}$$

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i)$$

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$$

- The effect of the decision rules is to divide the feature space into c decision regions R_1,.....R_c.
- If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$, then $\mathbf{x}$ is in region R_i.
- The regions are separated by decision boundaries, surfaces in features space where the discriminant functions for two classes are equal

The Gaussian density - univariate case (a single feature)

- To use a classifier we need to select a probability density function $p(x|\omega_i)$.
- The most commonly used probability density is the normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma} \right)^2 \right]$$

with expected value (or mean) $\mu = \mathrm{E}[\mathrm{x}] = \int_{-\infty}^{\infty} x p(x) dx$

and variance $\sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) dx$

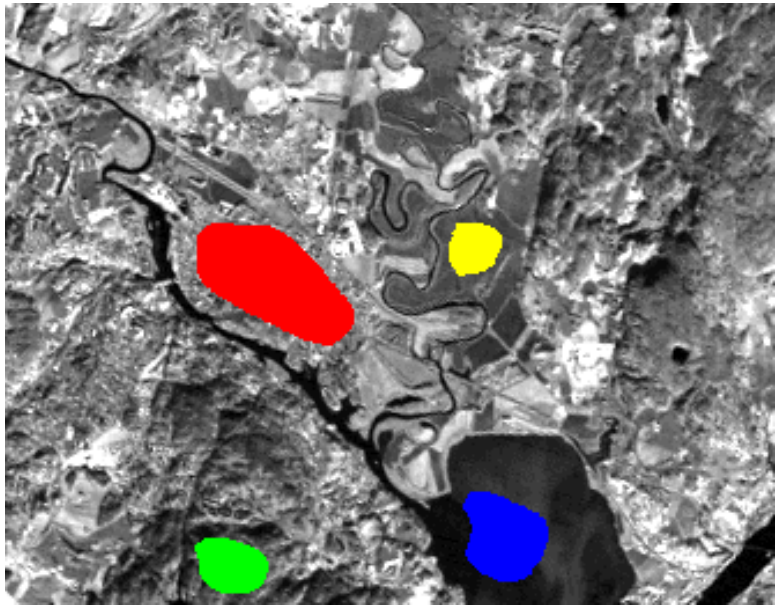
QUESTIONS
ANSWERS

Training data

- Collect image data with labels for each object type
  - Is the object a pixel, region, or image ? This varies.
- Each object category is assigned a number, called the class label.
- Example: Class 1: ‘dog’, class 2: ‘cat’, class 3: ‘bird’
- Integer (categorical) labels are used for implementation.

Example: image and training masks for pixel-based classification

The masks contain labels for the training data.
If label=1, then the pixel belongs to class 1 (red), and so on.
If a pixel is not part of the training data, it will have label 0.
A pixel belonging to class k will have value k in the mask image.
The mask is often visualized in pseudo-colors on top of the input image, where each class is assign a color.
We should have a similar mask for the test data.



How do we make training masks?

- Use a program that allows us to draw different shapes on top of an image, and save the result as a mask image
  - Matlab: e.g. drawpolygon:
    - pedestrian1 = imread('bike_001.png');
    - imshow(pedestrian1);
    - pedestrian1_ped=drawpolygon();
    - ped1 = createMask(pedestrian1_ped);
    - imwrite(ped1,"bike001_background.png");
- Depending on the task, we either draw:
  - Regions within each class (avoiding boundaries) for general classification
  - A larger Region-of-interest for region-based tasks like traffic sign segmentation
  - A detailed boundary for segmentation tasks

Training a univariate Gaussian classifier

- To be able to compute the value of the discriminant function, we need to have an estimate of μ_j and σ_j^2 for each class j.
- Assume that we know the true class labels for some pixels and that this is given in a mask image. The mask has N_k pixels for each class.
- Training the classifier then consists of computing μ_j and σ_j^2 for all pixels with class label j in the mask file.
- They are computed from training data as:
- For all pixels x_i with label k in the training mask, compute

$$\mu_k = \frac{1}{N_k} \sum_{i=i}^{N_k} x_i$$

$$\sigma_k^2 = \frac{1}{N_k} \sum_{i=i}^{N_k} (x_i - \mu_k)^2$$

Training a pixel-based classifier using masks

```
for i=1:N
  for j=i:M
    if mask(i,j)>==K
      increment nof. Samples in class K
      store the feature vector f(i,j) in a vector of training samples from class K
   end
end
end

For class k=1:K
    compute mean(k) and sigma(k)
```

$$\mu_k = \frac{1}{N_k}\sum_{i=i}^{N_k} x_i$$

$$\sigma_k^2 = \frac{1}{N_k}\sum_{i=i}^{N_k} (x_i - \mu_k)^2$$

How do to classification with a univariate Gaussian (1 feature)

- Decide on values for the prior probabilities, $P(\omega_j)$. If we have no prior information, assume that all classes are equally probable and $P(\omega_j)=1/J$.
- Estimate μ_j and σ_j^2 based on training data based on the formulae on the previous slide. (Training)
- For each pixel in a new image:

  For class j=1,....J, compute the discriminant function

$$P(\omega_j \mid x) = p(x \mid \omega_j)P(\omega_j) = \frac{1}{\sqrt{2\pi}\sigma_j}\exp\left[-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2\right]P(\omega_j)$$

  Assign pixel x to the class C with the highest value of $P(\omega_j|x)$ by setting label_image(x,y)= C

The result after classification is an image with class labels corresponding to the most probable class for each pixel.

We compute the classification error rate from an independent test mask.

Minimum distance classification

- As we will see next week, if we assume that all features are uncorrelated and have unit variance, the Gaussian model reduces to
  - Assign the feature vector to the class with the closest mean value measured by Euclidean distance.
  - Compute the distance to all class means and choose the class with the closest mean.
  - Euclidean distance for a feature vector of dimension F:
  - $\sqrt{\sum_{f=1}^{F}\left(x_f - \mu_{f,K}\right)^2}$

QUESTIONS
ANSWERS

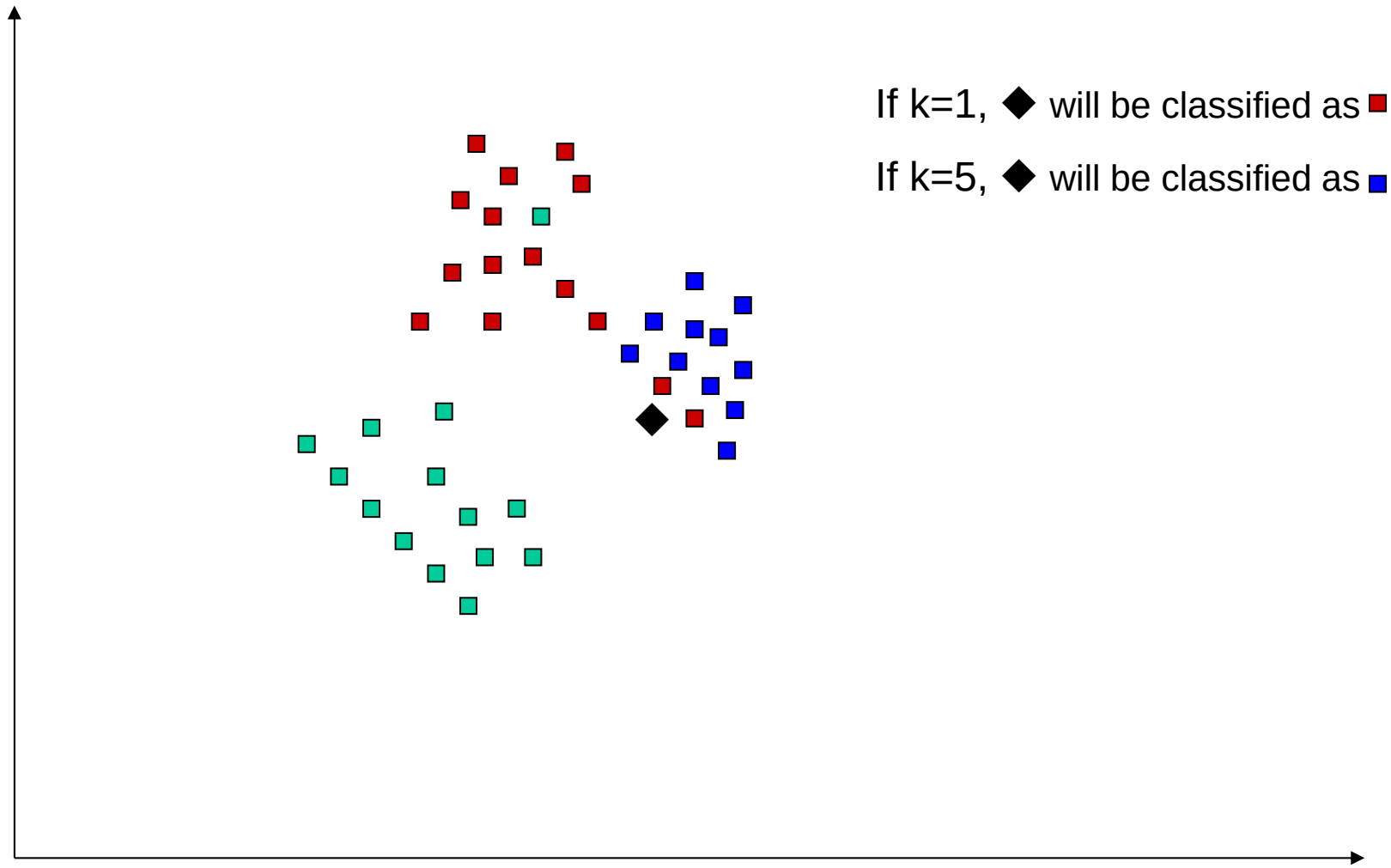
An alternative classifier - kNN

- Does not assume any particular probability density

k-Nearest-Neighbor classification

- A very simple classifier.
- Classification of a new sample x_i is done as follows:
  - Out of N training vectors, identify the k nearest neighbors (measured by Euclidean distance) in the training set, irrespectively of the class label.
  - Out of these k samples, identify the number of vectors k_i that belong to class ω_i , i:1,2,….M (if we have M classes)
  - Assign x_i to the class ω_i with the maximum number of k_i samples.
- k should be odd, and must be selected a priori.

kNN-example



If k=1, ◆ will be classified as ■

If k=5, ◆ will be classified as ■

About kNN-classification

- If *k=1* (1NN-classification), each sample is assigned to the same class as the closest sample in the training data set.
- If the number of training samples is very high, this can be a good rule.
- If k->∞, this is theoretically a very good classifier.
- This classifier involves no ”training time”, but the time needed to classify one pattern x_i will depend on the number of training samples, as the distance to all points in the training set must be computed.
- ”Practical” values for *k*: *3<=k<=9*
- *Classification performance should **always** be computed on the test data set.*
- *Find the best value of k on a validation data set.*

Estimating classification error

- A simple measure of classification accuracy can be to count the percentage of correctly classified pixels overall (averaged for all classes), or per. class. If a pixel has true class label k, it is correctly classified if ω_j=k.

- Estimate the classification error by classifying all pixels in the test set and count the percentage of wrongly classified pixels.

Validating classifier performance

- Classification performance is evaluated on a different set of samples with known class - the test set.
- The training set and the test set must be independent!
- Normally, the set of ground truth pixels (with known class) is partionioned into a sets for training, validation and test.
- If the classifier has hyperparameters, we use a separate validation set to find the best value of them.
  More on this in a later lecture.

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

True class labels

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

Confusion matrix - cont.

Alternatives:

- Report nof. correctly classified pixels for each class.
- Report the percentage of correctly classified pixels for each class.
- Report the percentage of correctly classified pixels in total.
  - Why is this not a good measure if the number of test pixels from each class varies between classes?

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

Using more than one feature

- The power of the computer lies in deciding based on more than 1 feature at a time
- A simple trick to do this is to assume that features i and j and independent, then p(i,j|c)=p(i|c)p(j|c)
- The joint decision based on D independent features is then:

$$P(\omega_j \mid x_1, x_2, \ldots x_D) = \frac{p(x_1 \mid \omega_j)\, p(x_2 \mid \omega_j) \cdots p(x_D \mid \omega_j) P(\omega_j)}{p(x_1, x_2, \ldots x_D)}$$

Upcoming lectures

- Multivariate Gaussian
- Classifier evaluation
- Support vector machine classification
- Feature selection/feature transforms
- Unsupervised classification

Thanks! Any more questions?

IN 5520
14.10.20

Multivariate classification

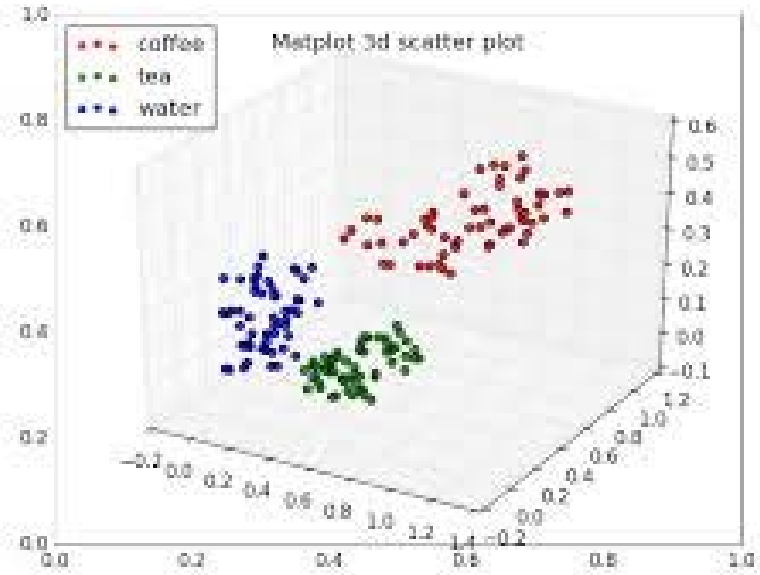
Anne Solberg (anne@ifi.uio.no)

Based on Chapter 2.1-2.4 in Pattern Recognition, Theodoridis and Koutroumbas

Mandatory 2

- Available before next group session
- Implement your own classification algorithm

The goal of supervised classification



Find a partition of multivariate feature space that we believe will work well when classifying new data
A simple model is easier to interpret/explain

Classification in multivariate feature space

- Goal:
  - Learn by concept
  - Learn by mathematics
  - Learn by geometry
  - Learn by implementation

Today's focus

- Basics of probability theory
- Bayes rule
- From a l-dimensional feature vector $\mathbf{x}=[x_1,....x_s]^T$
- The multivariate Gaussian density
- Discriminant functions for the Gaussian density
- If time: a classification example

Bayes rule for a classification problem

- Suppose we have J, j=1,...J classes. ω is the class label for a pixel, and $\boldsymbol{x}$ is the observed feature vector).
- We can use Bayes rule to find an expression for the **class with the highest probability**:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}$$

$$\text{posterior probability} = \frac{\textit{likelihood} \times \text{prior probability}}{\text{normalizing factor}}$$

- $P(\omega_j)$ is the prior probability for class ω_j. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).

Probability theory

- Let x be a discrete random variable that can assume any of a finite number of M different values (*in our case M classes*).

- The probability that x belongs to class m is

  $p_m = \Pr(x=m)$, m=1,…M

- A probability distribution must sum to 1 and $\sum_{m=1}^{M} p_m = 1$ probabilities must be positive so $p_m \geq 0$ and

Expected values - definition

- The expected value or mean of a random variable x is:

$$E[x] = \mu = \sum_x xP(x)$$

- The variance or second order moment σ^2 is:

$$E[x^2] = \sum_x x^2 P(x)$$

$$Var[x] = \sigma^2 = E[(x-u)^2] = \sum_x (x-u)^2 P(x)$$

Pairs of random variables - definitions

- Let *x* and *y* be two random variables.
- The joint probability of observing a **pair** of values (x=i,y=j) is p_{ij}.
- Alternatively we can define a joint probability distribution function P(x,y) for which

$$P(x,y) \geq 0, \quad \sum_x \sum_y P(x,y) = 1$$

- The marginal distributions for x and y (if we want to eliminate one of them) is:

$$P_x(x) = \sum_y P(x,y)$$

$$P_y(y) = \sum_x P(x,y)$$

Expected values of two variables

- Expected values of two variables:

$$E(f(x,y)) = \sum_x \sum_y f(x,y)P(x,y)$$

$$\mu_x = E(x) = \sum_x \sum_y xP(x,y)$$

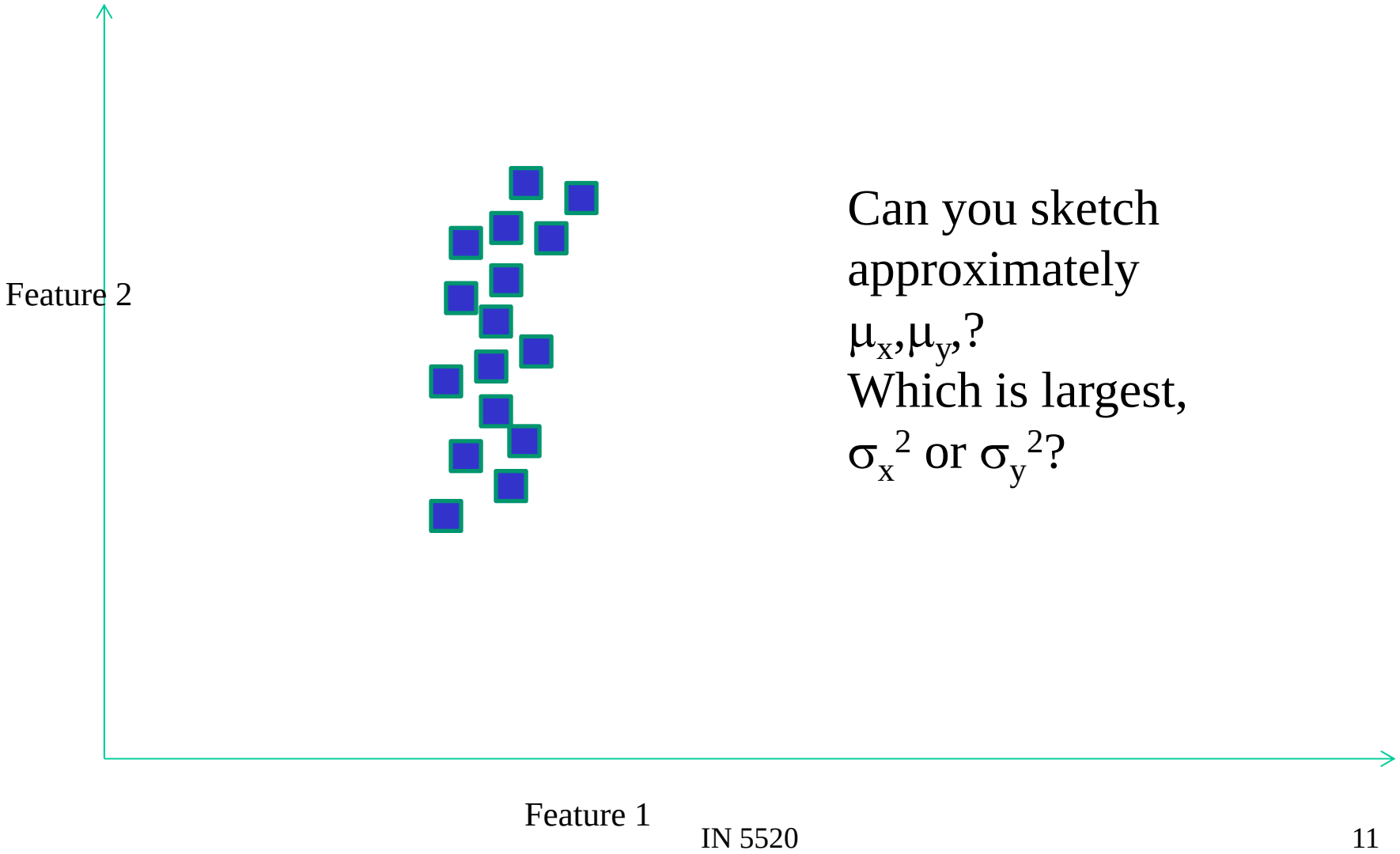
$$\mu_y = E(y) = \sum_x \sum_y yP(x,y)$$

$$\sigma_x^2 = E\left[(x-\mu_x)^2\right] = \sum_x \sum_y (x-\mu_x)^2 P(x,y)$$ Variance of feature x

$$\sigma_y^2 = E\left[(y-\mu_y)^2\right] = \sum_x \sum_y (y-\mu_y)^2 P(x,y)$$ Variance of featureyx

$$\sigma_{xy} = E\left[(x-\mu_x)(y-\mu_y)\right] = \sum_x \sum_y (x-\mu_x)(y-\mu_y)P(x,y)$$ Covariance of feature x and y

Where (in this course) have you seen similar formulas?



Can you sketch approximately μ_x, μ_y,?
Which is largest, σ_x^2 or σ_y^2?

Statistical independence - definitions

- Variables *x* and *y* are statistical independent if and only if

$$P(x, y) = P_x(x)P_y(y)$$

- In words: two variables are indepentent if the occurrence of one does not affect the other.
- If two variables are not independent, they are dependent.
- If two variables are independent, they are also uncorrelated.
- For more than two variables: all pairs must be independent.
- Two variables are uncorrelated if

$$\sigma_{xy} = 0$$

- If Cov[X ,Y] = E[X Y] – E[X]E[Y] =0, we must have E[X Y] = E[X]E[Y]

- If two variables are uncorrelated, they *can* still be dependent.

Conditional probability

- If two variables are statistically dependent, knowing the value of one of them lets us get a better estimate of the value of the other one. We need to consider their covariance.
- The conditional probability of *x* given *y* is defined:

$$\Pr[x = i \mid y = j] = \frac{\Pr[x = i, y = j]}{\Pr[y = j]}$$

and for distributions :

$$\mathrm{P}(\mathrm{x} \mid \mathrm{y}) = \frac{\mathrm{P}(\mathrm{x}, \mathrm{y})}{\mathrm{P}(\mathrm{y})}$$

- Example: Draw two cards from a deck. Drawing a king in the first draw has probability 4/52. Drawing a king in the second draw (given that the first draw gave a king) is 3/51.

QUESTIONS
ANSWERS

The conditional density p(**x**| ω_s)

- Any probability density function can be used to model p(**x**| ω_s)
- A common model is the multivariate Gaussian density.
- The multivariate Gaussian density with l features:

$$p(\mathbf{x} \mid \omega_s) = \frac{1}{(2\pi)^{l/2} \left|\boldsymbol{\Sigma}_s\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_s)^t \boldsymbol{\Sigma}_s^{-1}(\mathbf{x}-\boldsymbol{\mu}_s)\right]$$

- If we have l features, μ_s is a vector of length l and and Σ_s a $l \times l$ matrix (depends on class *s*)

$$\boldsymbol{\mu}_S = \begin{bmatrix} \mu_{1s} \\ \mu_{2s} \\ \\ \\ \mu_{ls} \end{bmatrix} \qquad \boldsymbol{\Sigma}_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1l} \\ \sigma_{21} & \sigma_{22} & . & . & . \\ \sigma_{31} & \sigma_{11} & . & . & . \\ . & . & . & . & . \\ \sigma_{l1} & \sigma_{l2} & . & \sigma_{ll-1} & \sigma_{ll} \end{bmatrix}$$

Symmetric l×l matrix
σ_{ii} is the variance of feature i
σ_{ij} is the covariance between feature i and feature j
Symmetric because $\sigma_{ij} = \sigma_{ji}$

- $|\Sigma_s|$ is the determinant of the matrix Σ_s, and Σ_s^{-1} is the inverse

Inspecting p(**x**|ω_s)

$$\boxed{p(\mathbf{x} \mid \omega_s)} = \boxed{\frac{1}{(2\pi)^{l/2}\left|\boldsymbol{\Sigma}_s\right|^{1/2}}} \exp\left[-\frac{1}{2}\boxed{(\mathbf{x}-\boldsymbol{\mu}_s)^t}\boxed{\boldsymbol{\Sigma}_s^{-1}}\boxed{(\mathbf{x}-\boldsymbol{\mu}_s)}\right]$$

Scalar

$1 \times l$ vector transposed

$l \times 1$ vector transposed

Scalar probability

$l \times l$ matrix

Inverse of covariance matrix

Hint: Set up an example with $x=[x_1, x_2]^T$ and check dimensions

The mean vectors μ_s for each class

- The mean vector for class s is defined as the expected value of **x**:

$$\boldsymbol{\mu}_s = E[\mathbf{x}] = \begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_l) \end{bmatrix} = \begin{bmatrix} \mu_1^{\,s} \\ \mu_2^{\,s} \\ . \\ . \\ \mu_l^{\,s} \end{bmatrix}$$

class s

feature number l

- with l features, the mean vector μ will be of size $1 \times l$

Link to moments

- From lecture on moments:

$$m_{10} = \sum_x \sum_y x f(x,y) = \bar{x} m_{00} \quad \Rightarrow \quad \bar{x} = \frac{m_{10}}{m_{00}}$$

$$m_{01} = \sum_x \sum_y y f(x,y) = \bar{y} m_{00} \quad \Rightarrow \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

  - m_{00} was the number of pixels in the object

- If **f**=[x,y] is a sample from distribution p(x,y), the mean is defined as

$$\mu_x = \sum_x \sum_y x p(x,y)$$

$$\mu_y = \sum_x \sum_y y p(x,y)$$

Remark – what is maximum likelihood estimation

- The true value of μ and Σ is unknown.
- A distribution has some unknown parameters
- Maximum likelihood estimation:
  - These parameters are assumed unknown, but deterministic (not random), meaning that they have a single true, uknown value (and no uncertainty)
  - Estimate by finding the value that maximize the likelihood given the set of observed samples
- Bayesian estimation, on the other hand, assumes that these parameters are random variables from some distribution.
  - A set of samples gives us the maximum aposteriori value of the parameters.

Maximum likelihood estimation

- We assume the the feature vector **x** is distributed according to $p(\mathbf{x}|\omega_k)$ if it belongs to class ω_k.
- In this case we assume $p(\mathbf{x}|\omega_k)$ is a Gaussian distribution with unknown parameters θ (μ_k and Σ_k for the Gaussian distribution).
- Let $X=[\mathbf{x}_1,\ldots\ldots\mathbf{x}_M]$ be M random samples drawn from $p(\mathbf{x}|\omega_k)$.
- If all samples are independent,
- $P(X;\,\theta_k) = \prod_{m=1}^{M} p(x_m;\,\theta_k)$
- The Maximum likelihood method estimates θ_k as the value that maximize the likelihood function:
- $\widehat{\theta_k} = \underset{\theta_k}{\operatorname{argmax}} \prod_{m=1}^{M} p(x_m\,;\,\theta_k)$
- This is equivalent to maximizing the logarithm of this, called the log-likelihood

Estimating the mean vectors μ_s

- If we have M_s training samples that we know belong to class s, we can estimate the mean vector as (Maximum likelihood estimates given the observed samples):

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} \mathbf{x}_m,$$

where the sum is over all training samples belonging to class s

For a derivation of this, see e.g.:
https://towardsdatascience.com/maximum-likelihood-estimation-explained-normal-distribution-6207b322e47f

QUESTIONS
ANSWERS

The covariance matrix Σ_s for each class

- The covariance for class s is defined as the expected value of $(\mathbf{x}-\mu)(\mathbf{x}-\mu)^t$:

$$\mathbf{\Sigma}_s = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1l} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{l} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{l1} & \sigma_{l2} & . & . & \sigma_{ll} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & \sigma_{1l} \\ \sigma_{21} & \sigma_2^2 & . & . & \sigma_{2l} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{l1} & \sigma_{l2} & . & . & \sigma_l^2 \end{bmatrix}$$

- with l features, the covariance matrix Σ_s will be of size $l x l$.

$\hat{f}$

More on the covariance matrix Σ_s

- The covariance matrix Σ_s will always be symmetric and positive semidefinite.
- If all components of x have non-zero variance, Σ_s will be positive definite.
- σ_{ij} is the covariance between features *i* and *j*.
- If features x_i and x_j are uncorrelated, $\sigma_{ij} = 0$.

- In the general case, Σ_s will have $l(l+1)/2$ different values.

Estimating the covariance matrix Σ_s for each class

- If we have M_s training samples that we know belong to class s, we can estimate the covariance matrix Σ_s . (The estimate of a random variable f is denoted $\hat{f}$)

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} (\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)^t$$

where the sum is over all training samples belonging to class s

- The Maximum likelihood estimate of each term σ_{ij} is computed as:

$$\sigma_{ij,s}{}^2 = \frac{1}{M_s} \sum_{m=1}^{M_s} (x_{m,i} - \hat{\mu}_{i,s})(x_{m,j} - \hat{\mu}_{j.s})^t$$

for the covariance between feature i and j for class s

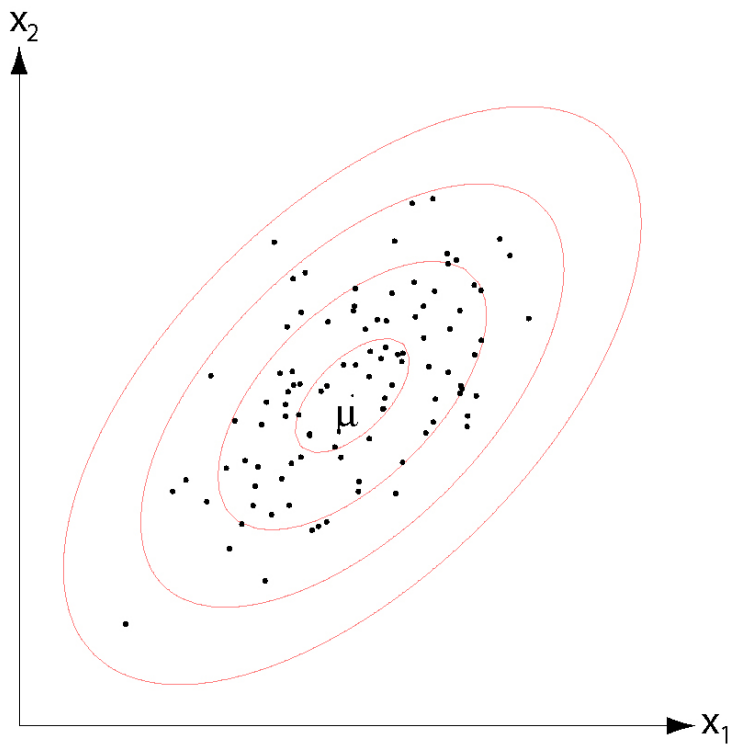
The covariance matrix and ellipses

- In 2D, the Gaussian model can be thought of as approximating the classes in 2D feature space with ellipses.
- The mean vector $\mu=[\mu_1, \mu_2]$ defines the the center point of the ellipses.
- σ_{12}, the covariance between the features defines the orientation of the ellipse.
- σ_{11} and σ_{22} defines the width of the ellipse.

$$\Sigma_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

- The ellipse defines points where the probability density is equal
  - Equal in the sense that the distance to the mean as computed by the Mahalanobis distance is equal.
  - The Mahalanobis distance between a point x and the class center μ is:

$$r^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$



The main axes of the ellipse is determined by the eigenvectors of Σ.
The eigenvalues of Σ gives their length.

- Let us consider two features with mean 0, feature 1 has variance σ_1^2 , feature 2 variance, σ_2^2 and feature 1 and 2 has covariance 0.
- The curve of points with equal probability is given as

$$\mathbf{x}^T \Sigma^{-1} \mathbf{x} = [x_1, x_2] \begin{bmatrix} 1/\sigma_1^2 & 0 \\ 0 & 1/\sigma_2^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = C \quad \text{or}$$

$$\frac{\mathrm{x}_1^2}{\sigma_1^2} + \frac{\mathrm{x}_2^2}{\sigma_2^2} = C$$

for some constant C

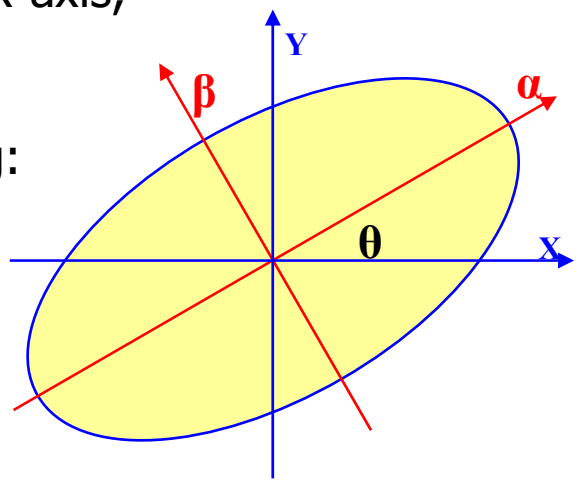
From lecture on moments: Object orientation

- Orientation is defined as the angle, relative to the X-axis, of an axis through the centre of mass that gives the lowest moment of inertia.
- Orientation θ relative to X-axis found by minimizing:

$$I(\theta) = \sum_{\alpha}\sum_{\beta} \beta^2 \; f(\alpha, \beta)$$

where the rotated coordinates are given by

$$\alpha = x\cos\theta + y\sin\theta, \qquad \beta = -x\sin\theta + y\cos\theta$$



- We found that object orientation was given by:

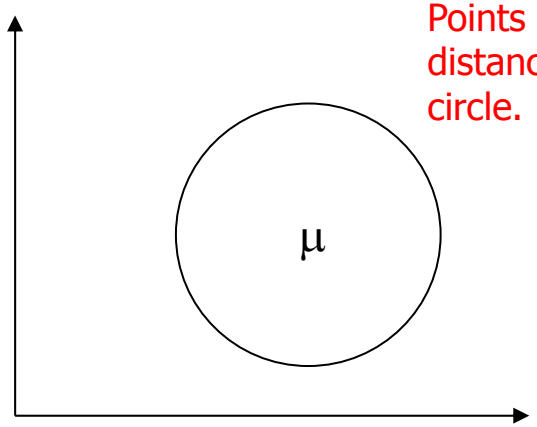
$$\underline{\underline{\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mu_{11}}{(\mu_{20} - \mu_{02})}\right]}}, \qquad where \;\; \theta \in \left[0, \pi/2\right] if \;\; \mu_{11} > 0, \;\; \theta \in \left[\pi/2, \pi\right] if \;\; \mu_{11} < 0$$

Can we use this to find the orientation of the covariance matrix?

Euclidean distance vs. Mahalanobis distance

- Euclidean distance between point x and class center μ:

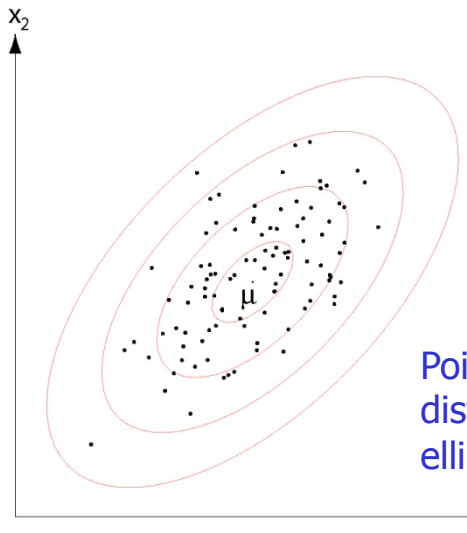
$$(x-\mu)^T(x-\mu)=\|x-\mu\|^2$$



Points with equal distance to μ lie on a circle.

- Mahalanobis distance between x and μ:

$$r^2=(x-\mu)^T\Sigma^{-1}(x-\mu)$$



Points with equal distance to μ lie on an ellipse.

Back to the Gaussian:

- We now have all the terms to compute

$$p(\mathbf{x} \mid \omega_s) = \frac{1}{(2\pi)^{l/2} \left|\boldsymbol{\Sigma}_s\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_s)^t \boldsymbol{\Sigma}_s^{-1} (\mathbf{x} - \boldsymbol{\mu}_s)\right]$$

Training a multivariate Gaussian classifier

- Training the classifier then consists of computing μ_s and Σ_s for all pixels with class label s in the mask file.
- For all pixels x_i with label s in the training mask, compute

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} \mathbf{x}_m ,$$

where the sum is over all training samples belonging to class s

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} \left(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s\right)\left(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s\right)^t$$

where the sum is over all training samples belonging to class s

How do to classification with a multiivariate Gaussian

- Decide on values for the prior probabilities, $P(\omega_j)$. If we have no prior information, assume that all classes are equally probable and $P(\omega_j)=1/J$. l is the number of features.
- Estimate μ_j and σ_j^2 based on training data based on the formulae on the previous slide. (Training)
- For each pixel in a new image:

  For class j=1,….J, compute the discriminant function

$$P(\omega_j|x) = p(\mathbf{x}|\omega_j)P(\omega_j) = \frac{1}{(2\pi)^{l/2}|\mathbf{\Sigma}_j|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^t \mathbf{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right] P(\omega_j)$$

  Assign pixel x to the class C with the highest value of $P(\omega_j|x)$ by setting label_image(x,y)= C

The result after classification is an image with class labels corresponding to the most probable class for each pixel.

QUESTIONS
ANSWERS

How a Gaussian classifier partions feature space

Discriminant functions for the normal density

- When finding the class with the highest probability, these functions are equivalent:

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}$$

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i)$$

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$$

- Let us now look at $g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$
- With a multivariate Gaussian we get:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \Sigma_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) - \frac{l}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

- Let us look at this expression for some special cases:

Case 1: $\Sigma_j=\sigma^2 I$

- $\mathbf{\Sigma}_j^{-1}=\mathbf{I}/\sigma^2$
- $|\mathbf{\Sigma}_j|= \sigma^{2n}$

- The discriminant functions can be expressed as:

$$g_i(\mathbf{x}) = -\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma^2} + \ln P(\omega_i)$$

$$\text{where } \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = (\mathbf{x}-\boldsymbol{\mu}_i)^t(\mathbf{x}-\boldsymbol{\mu}_i)$$

- Thus we model the probabilities as n-dimensional *spheres* because points that have equal discriminant function will lie on a circle around the mean μ_i.

Case 1: $\Sigma_j=\sigma^2 I$ – simplifying the expression

- The discriminant functions simplifies to **linear** functions using such a shape on the probability distributions

$$g_j(\mathbf{x}) = -\frac{1}{2(\sigma^2 I)}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{l}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\sigma^2 I\right| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\mathbf{x} - 2\boldsymbol{\mu}_j^T\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j) - \frac{l}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\sigma^2 I\right| + \ln P(\omega_j)$$

Common for all classes, no need to compute these terms

Since ***$x^T x$* <u>is common for all classes</u>**, an equivalent $g_j(\boldsymbol{x})$ is a linear function of ***x:*** .

$$\frac{1}{(\sigma^2)}\boldsymbol{\mu}_j^T\mathbf{x} - \frac{1}{2(\sigma^2)}\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j + \ln P(\omega_j)$$

Case 1: $\Sigma_j=\sigma^2 I$

- Now we get an equivalent formulation of the discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } w_{i0} = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0) = 0$$

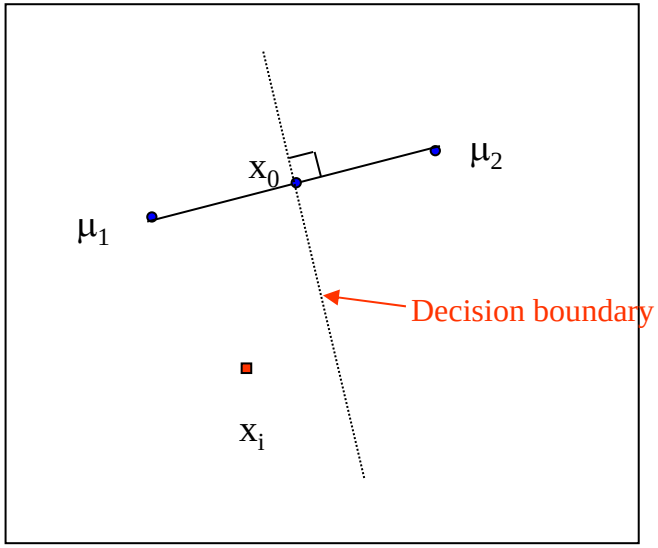
$$\text{where } \mathbf{w} = \boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}$$

$$\text{and } x_0 = \frac{1}{2}\left(\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right) - \frac{\sigma^2}{\left\|\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}\left(\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right)$$

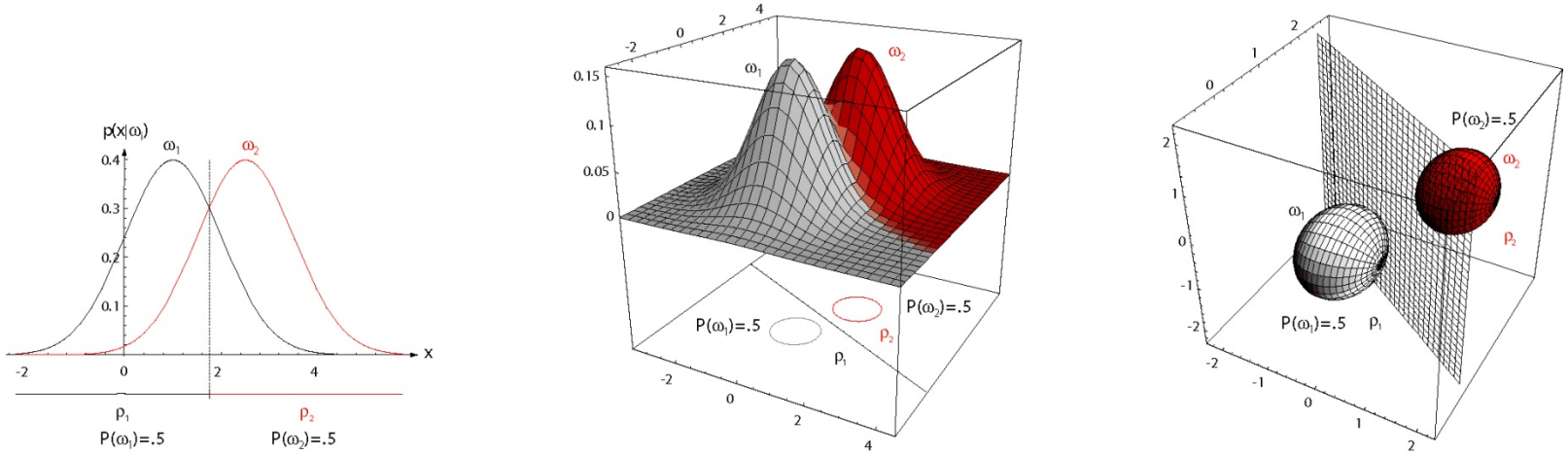
- $\mathbf{w}=\mu_i-\mu_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.
- If $P(\omega_i)=P(\omega_j)$ the hyperplane will be located halfway between the mean values.
- Proving this involves some algebra, see the proof at https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

Case 1: $\Sigma_j=\sigma^2 I$ – Decision boundary

- The discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $x_0=(\mu_1 - \mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



With l features, $\Sigma_j=\sigma^2 I$



- The distributions are spherical in *l* dimensions.
- The decision boundary is a generalized hyperplane of *l-1* dimensions
- The decision boundary is perpendicular to the line separating the two mean values
- This kind of a classifier is called a linear classifier, or a linear discriminant function
  - Because the decision function is a linear function of ***x***.
- If $P(\omega_i)= P(\omega_j)$, the decision boundary will be half-way between μ_i and μ_j

Minimum distance classification

- If all classes have equal diagonal covariance matrix and equal prior probabilities, x_0 will be the point halfway between the mean vectors.

- Classification will consist of assigning feature vector x to the same class as the closest mean measured by Euclidean distance $||x-\mu_i||$.

- A classifier based on the Euclidean distance is called a **minimum distance classifier**.

Case 2: Common covariance, $\Sigma_j = \Sigma$

- If we assume that all classes have the same shape of data clusters, an intuitive model is to assume that their probability distributions have the same shape
- By this assumption we can use all the data to estimate the covariance matrix
- This estimate is common for all classes, and this means that also in this case the discriminant functions become linear functions

$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

Common for all classes, no need to compute

Since $\boldsymbol{x}^T\boldsymbol{x}$ is common for all classes, $g_j(\boldsymbol{x})$ again reduces to a linear function of $\boldsymbol{x}$.

Case 2: Common covariance, $\Sigma_j = \Sigma$

- An equivalent formulation of the discriminant functions is

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + wi_0$$
$$\text{where } \mathbf{w}_i = \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i$$
$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- The decision boundaries are again hyperplanes.
- Because $\mathbf{w}_i = \mathbf{\Sigma}^{-1}(\mu_i - \mu_j)$ is not in the direction of $(\mu_i - \mu_j)$, the hyperplane will not be orthogonal to the line between the means.

Case 2

- Do an eigenvector decomposition of Σ

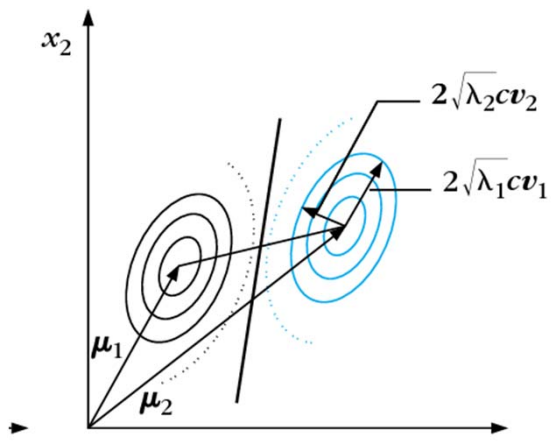
$$\text{Eigenvalues}: \lambda_1, \ldots \lambda_l$$
$$\text{Eigenvectors}: \Phi = [v_1, \ldots, v_l]$$

- Project the data onto the eigenvectors by setting $\mathbf{x}' = \Phi^T \mathbf{x}$
- It can be shown that the contours with equal probability in the transformed space is:

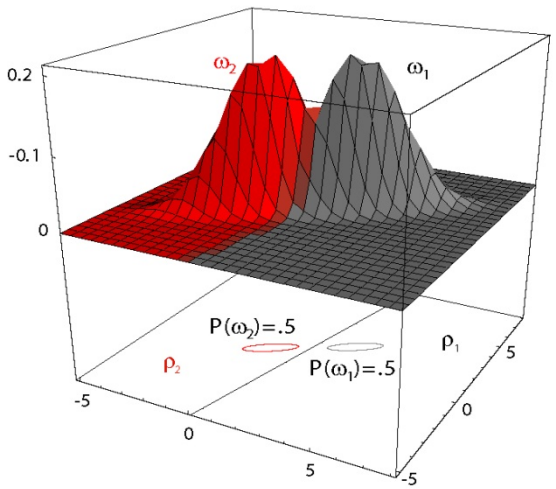
$$\frac{(x_1' - \mu_{i1})}{\lambda_1} + \ldots + \frac{(x_l' - \mu_{il})}{\lambda_l} = C^2$$

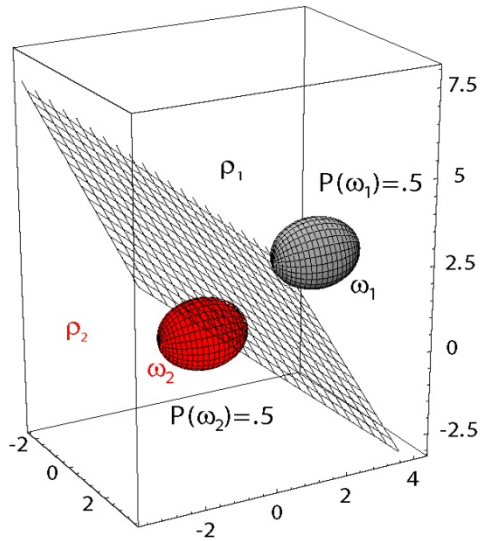
- The center of mass of the ellipses are a μ_i, the principal axes align with the eigenvalues and have length

$$2\sqrt{\lambda_k} C$$



Case 2:, $\Sigma_j = \Sigma$





- The classes can be described by hyperellipsoides in l dimensions.
- All hyperellipsoids have the same orientation.
- The decision boundary will again be a hyperplane.
- Because $\boldsymbol{w} = \Sigma^{-1}(\mu_i - \mu_j)$ is generally not in the direction of $\mu_i - \mu_j$, the hyperplane will not be perpendicular to the line between the means.
- Consider a point x_0 on the line $\mu_i - \mu_j$, defined by the prior probabilities:
  - If $P(\omega_i) = P(\omega_j)$, x_0 will be half way between the means.
  - The separating hyperplane will *intersect* the line at x_0

Case 3:, Σ_j=arbitrary

- When all classes are modeled as having different *shapes,* the discriminant functions cannot be simplified
- This means that the discriminant functions will be *quadratic* functions

- Decision boundaries will be hyperquadrics and assume any of the general forms:
  - hyperplanes, pairs of hyperplanes, hyperspheres, hyperellisoides, hyperparaboloids, hyperhyperboloids...

Case 3:, Σ_j=arbitrary

- The discriminant functions will be quadratic:

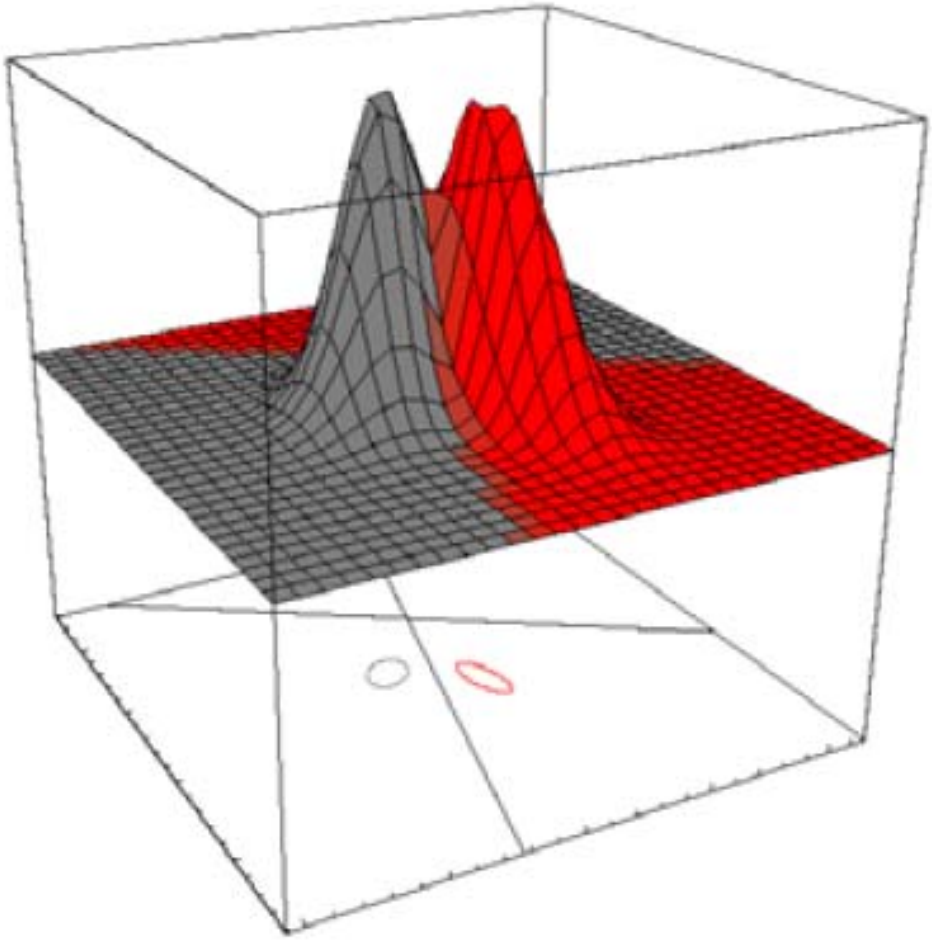
$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + wi_0$$

$$\text{where } \mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \qquad \mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

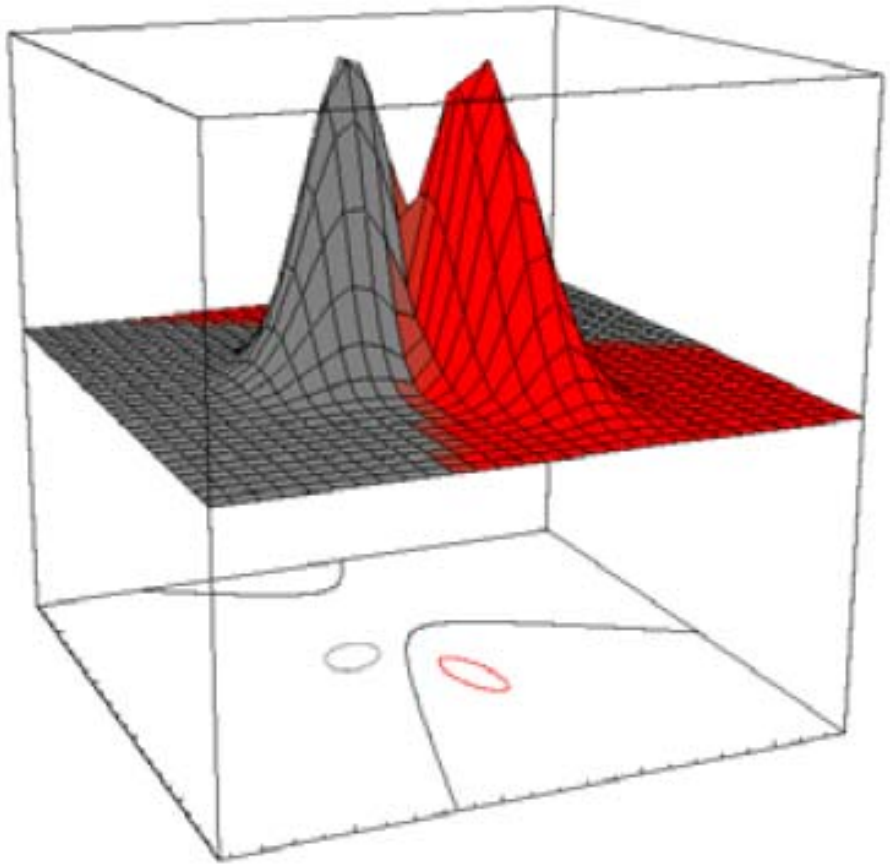
$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln\left|\boldsymbol{\Sigma}_i\right| + \ln P(\omega_i)$$

- The decision surfaces are hyperquadrics and can assume any of the general forms:
  - hyperplanes
  - hypershperes
  - pairs of hyperplanes
  - hyperellisoids,
  - Hyperparaboloids,..
- The next slides show examples of this.
- In this general case we cannot intuitively draw the decision boundaries just by looking at the mean and covariance.

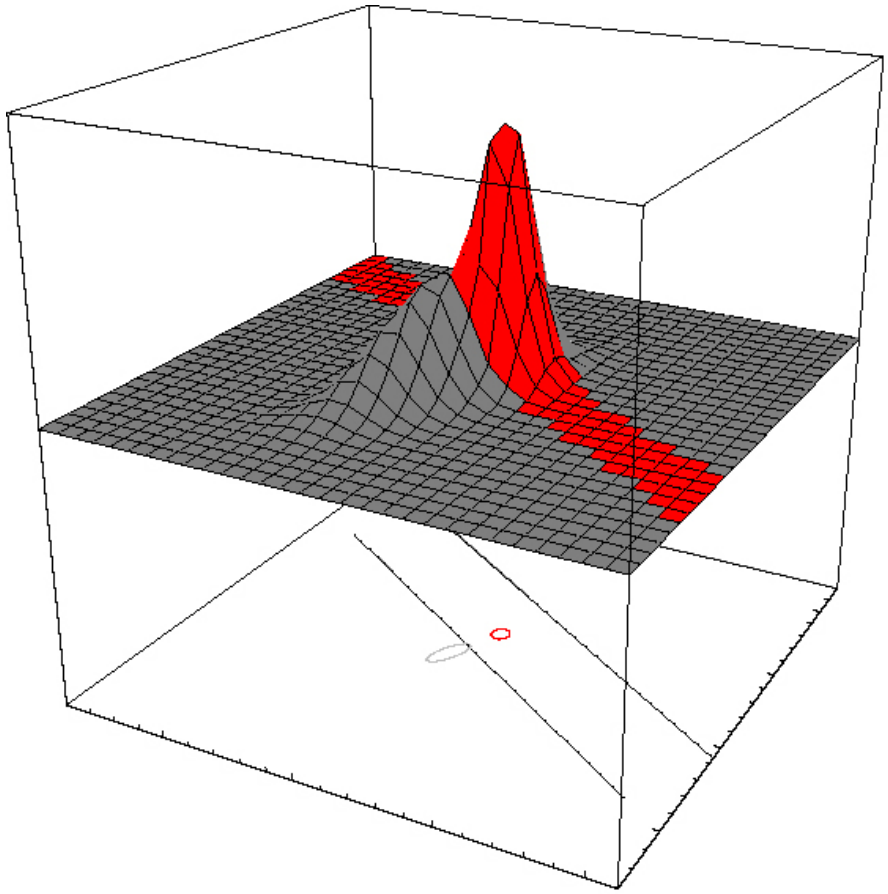
The full model, Σ_j=arbitrary - example



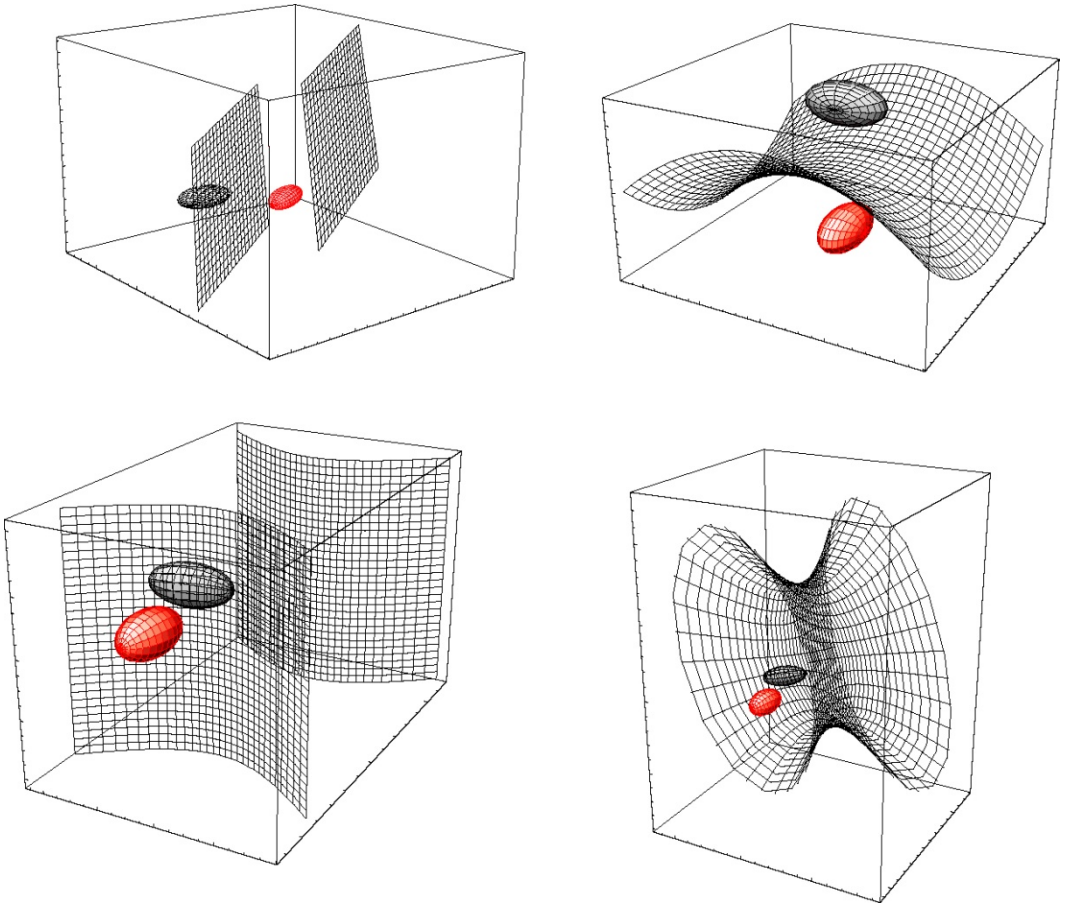
The full model, Σ_j=arbitrary - example



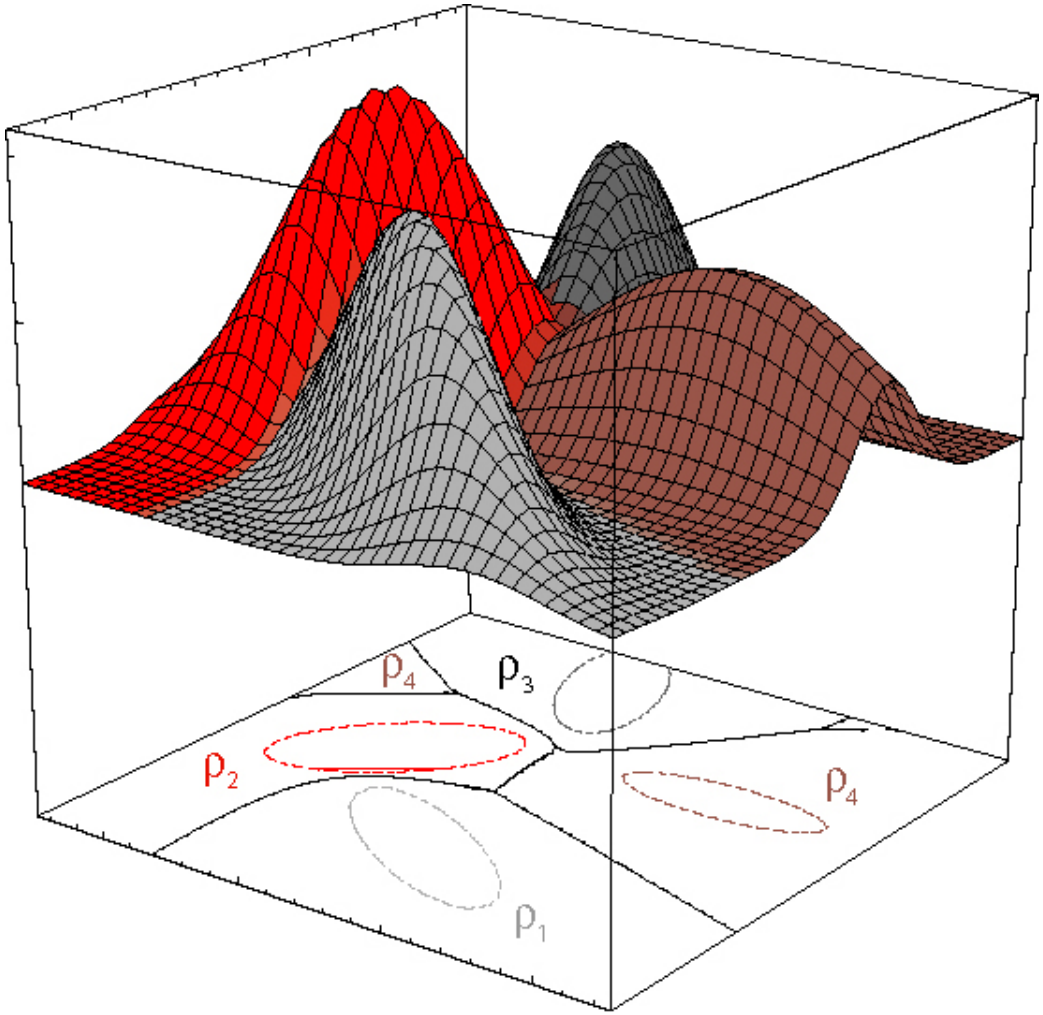
The full model, Σ_j=arbitrary - example



The full model, Σj=arbitrary - example



A multiclass example



QUESTIONS
ANSWERS

Is the Gaussian classifier the only choice?

- The Gaussian classifier gives linear or quadratic discriminant function.
- Other classifiers can give arbitrary complex decision surfaces (often piecewise-linear)
  - Mixtures of Gaussians
  - Other probability density functions (t-distribution, exponential distributions).
  - Softmax-classifier
  - Neural networks
  - Support vector machines
  - Ensembles of simple classifiers
    ADAboost
    Random forest/decision trees
  - kNN (k-Nearest-Neighbor) classification
  - Logistic classification

A classification example

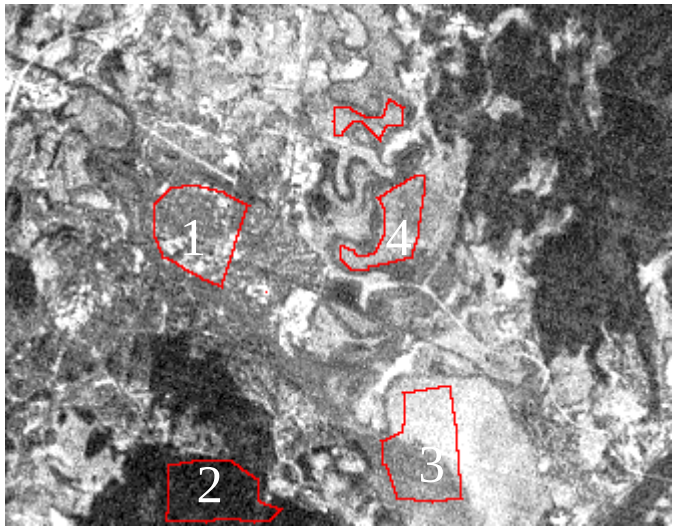
Landsat image with 6 spectral bands
The 6 bands will be the features
Training areas and test areas shown
in mask

Upper part: RGB-false color image created from bands 4,5 and 6 with training and test regions overlaid.

Lower part: image of training regions only

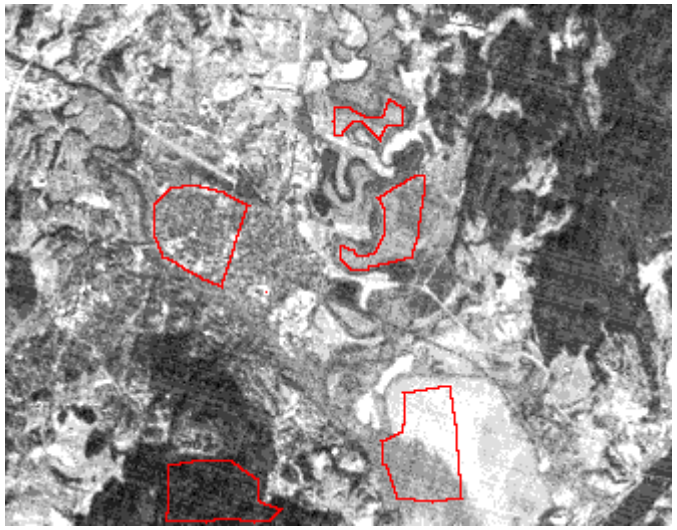
Visual inspection of feature 1

Class 2 (forest) seems to be well separated,
Maybe also class 1 (urban)



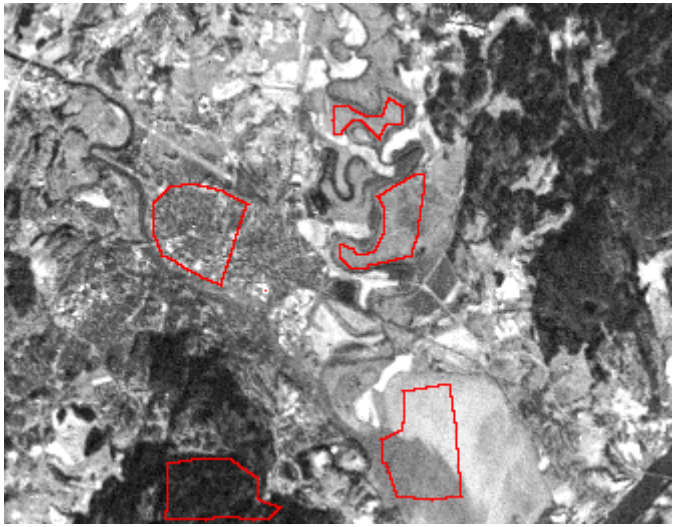
Visual inspection of feature 2

Class 2 (forest) seems to be well separated



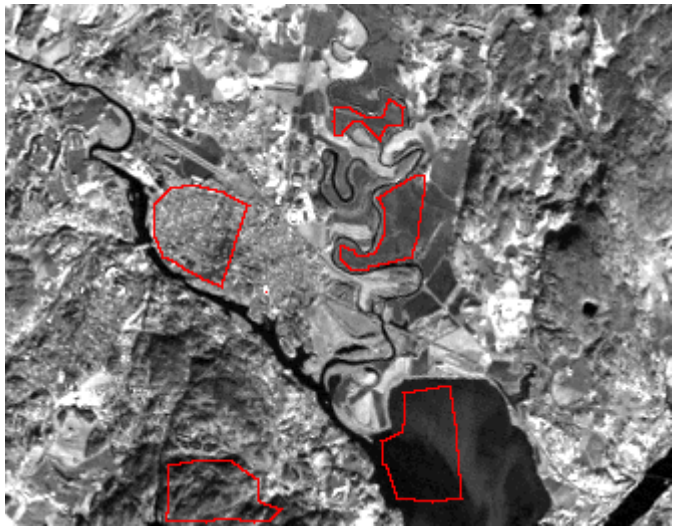
Visual inspection of feature 3

Class 2 (forest) seems to be well separated,
Class 1 (urban) seems to be well separated



Visual inspection of feature 4

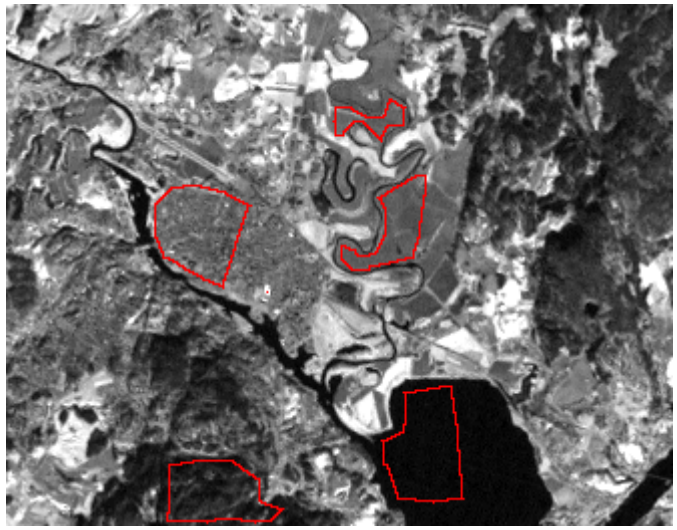
Class 1 (water) seems to be well separated,
Maybe also class 4 (agricultural)



Visual inspection of feature 5

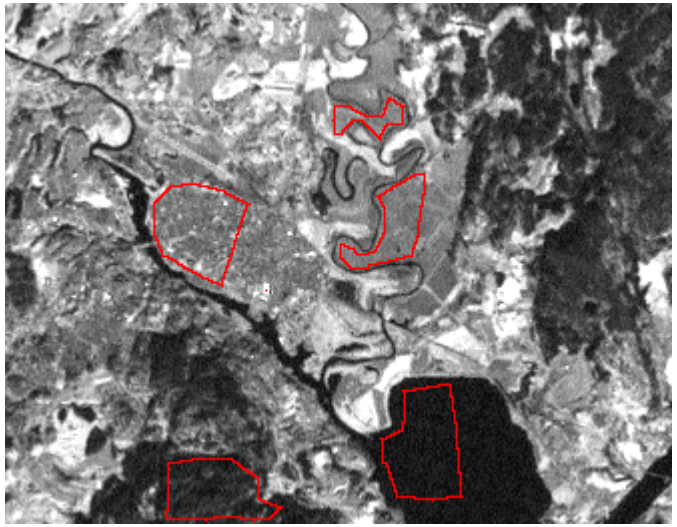
Water and forest appears similar - but the variance might be different

Urban and agricultural appears similar – but the variance might be different

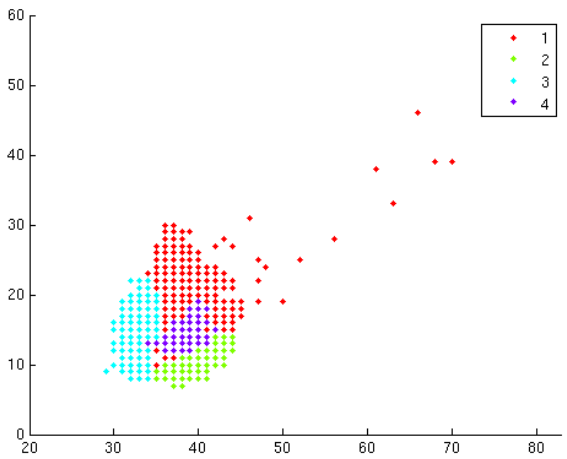


Visual inspection of feature 6

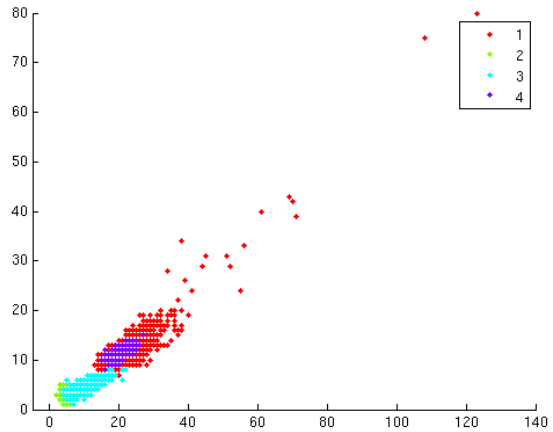
Seems similar to feature 5,
but with better contrast



Selected scatter plots (gscatter)

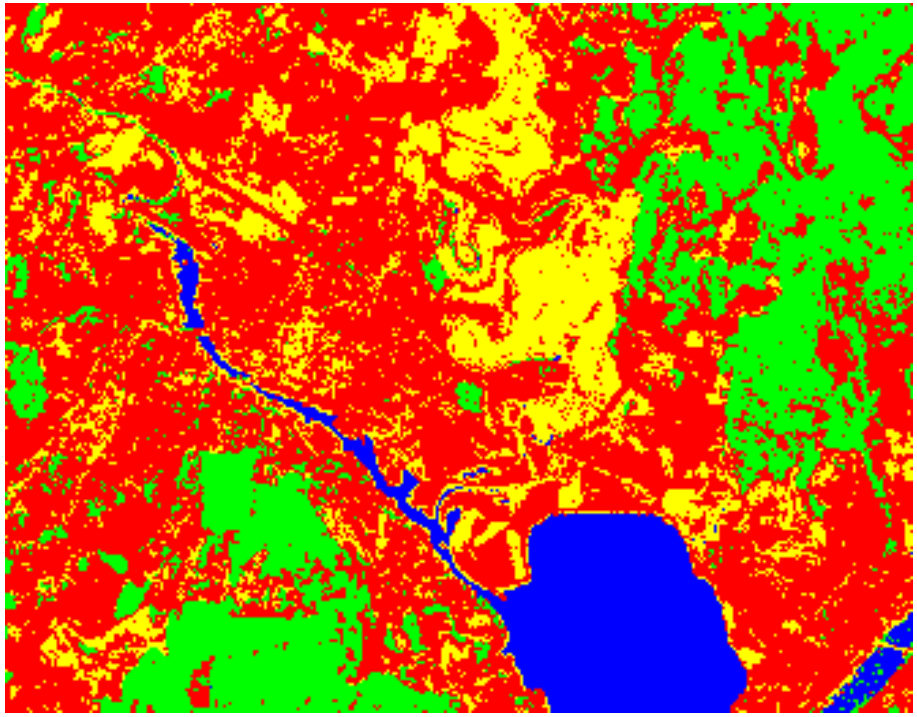


Scatterplot between feature 1 and 4



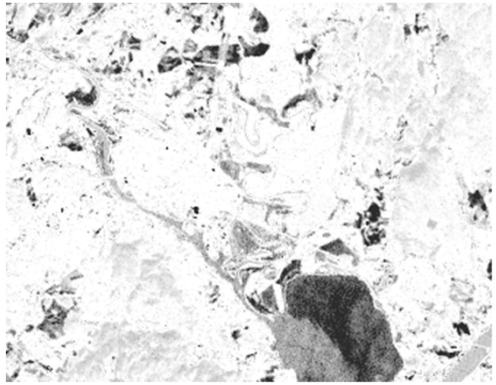
Scatterplot between feature 5 and 6

Classified images

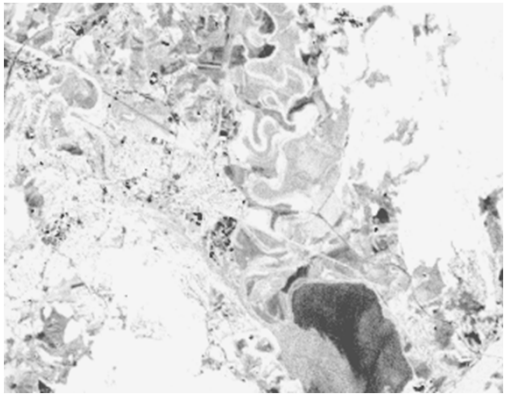


The entire image classified to the most probable class
A color table is used to display the different classes.

Display the posterior probabilities as images



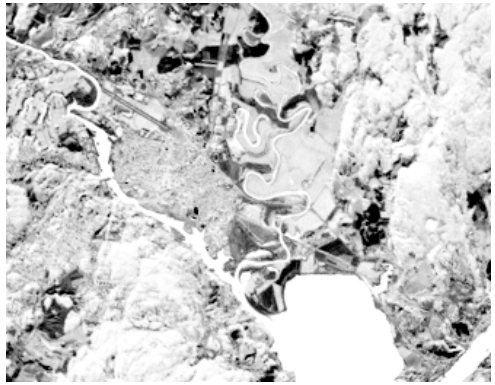
Posterior probability for class urban



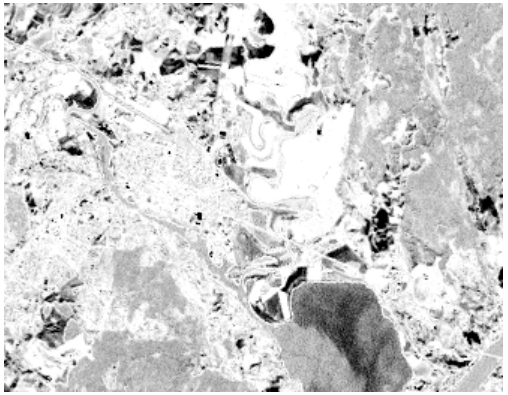
Posterior probability for class forest

Dark values:
Probabilities close to 0

Bright values:
Probabilities close to 1



Posterior probability for class water



Posterior probability for class agricultural

Confusion matrix for the training set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1340	2	0	310
Class 2	43	1253	0	2
Class 3	0	0	1738	0
Class 4	131	3	0	1266

Accuracy per class: Averaged over all classes: 91.7%

Class1: 81%

Class2: 96%

Class3: 100%

Class4: 90%

Confusion matrix for the test set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1474	3	1	251
Class 2	513	2311	0	0
Class 3	14	0	1953	0
Class 4	213	2	0	1390

Accuracy per class:
Class1: 85%
Class2: 81%
Class3: 98%
Class4: 86%

Averaged over all classes: 87.5%

Learning goals from this lecture

- Be able to **use and implement** Bayes rule with a l-dimensional Gaussian distribution.
- Know how μ_s and Σ_s are estimated.
- Understand the 2-dimensional case where a covariance matrix is illustrated as an ellipse.
- Be able to simplify the general discriminant function for 3 cases.
- Be able to compute the discriminant function e.g. for case 1.
- Have a geometric interpretation of classification with 2 features.
- Be able to solve theoretical exercises on classification.

QUESTIONS
ANSWERS

IN 5520
28.10.20
Pratical guidelines for classification
Evaluation
Feature selection
Principal component transform

Anne Solberg (anne@ifi.uio.no)

Literature

- Practical guidelines of classification – lecture foils
- Chap 5.6-5.7 Class separability and feature selection (see undervisningsmateriale/lecturenotes)
- Chap 6.1-6.2 PCA (see undervisningsmateriale/lecturenotes)
  Recommended reading about PCA, text from Chapter 17 Principal Component Analysis (see undervisningsmateriale/lecturenotes)

Approaching a classification problem

- Collect and label data
- Get to know the data: exploratory data analysis
- Choose features
- Consider preprocessing/normalization
- Choose classifier
  - Estimate classifier parameters on training data

Is the accuracy in the range I want?
NOT?
Analyze errors to understand why not
Improve algorithm, repeat

Approaching a classification problem

- Collect and label data
- Get to know the data: exploratory data analysis
- Choose features
- Consider preprocessing/normalization
- Choose classifier
  - Estimate classifier parameters on training data
- Estimate hyperparameters on validation data
  - Alternative: cross-validation on the training data set
- Compute the accuracy on test data

Model fit and bias,variance

- f(x) unknown function
- Training samples $x_1,\ldots.x_n$
- Measurements $y_i = f(x_i) + \varepsilon_i$ (ε_i :noise with zero mean and variance σ_ε^2)
- Estimate $\widehat{f(x)}$
- Want to minimize the error $E[(y-\widehat{f(x)})^2]$
- Bias : Difference between f(x) and $\widehat{f(x)}$ - how well does the model fit
- Variance of the estimate: $E[(\hat{f}(x) - E[\hat{f}(x)])^2]$

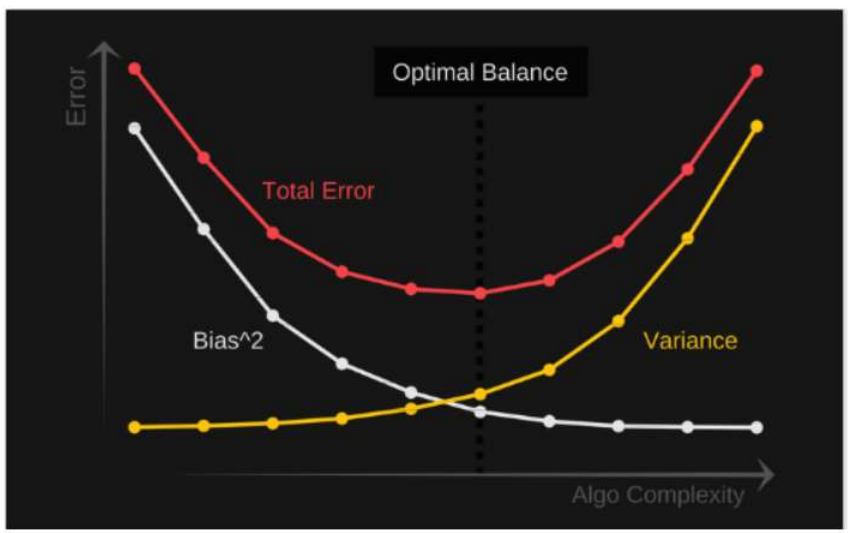
Balancing the total error

- Total squared error $Err(X) = E\left[\left(X - \widehat{f(x)}\right)^2\right]$

  The Err(x) can be further decomposed as

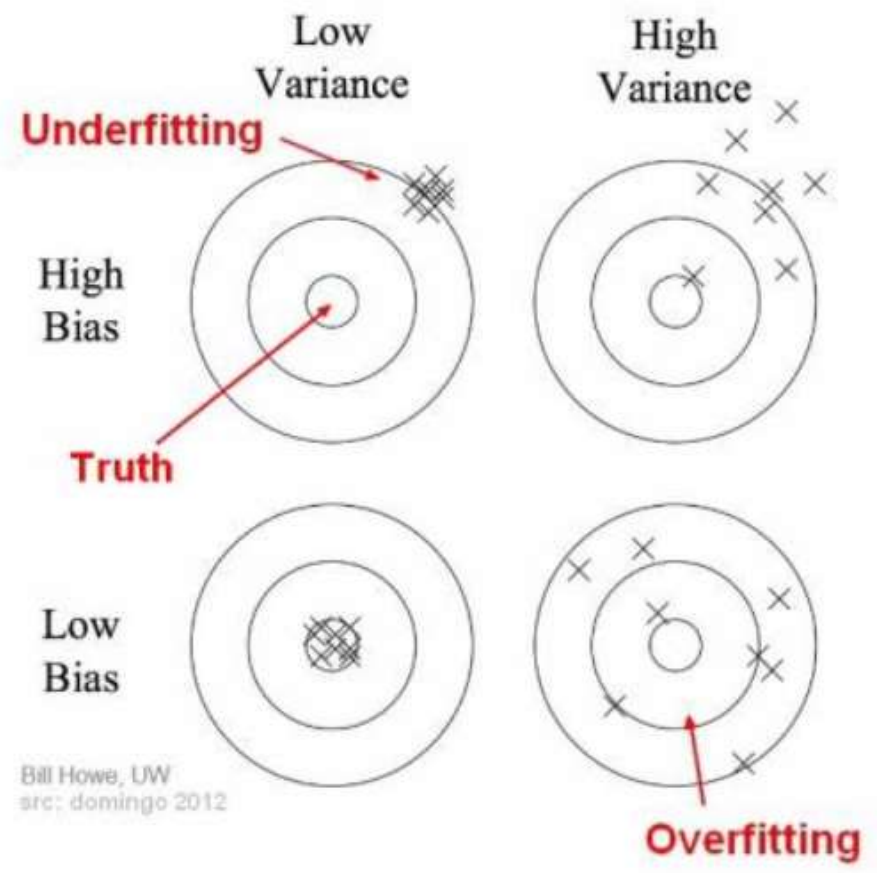
$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$



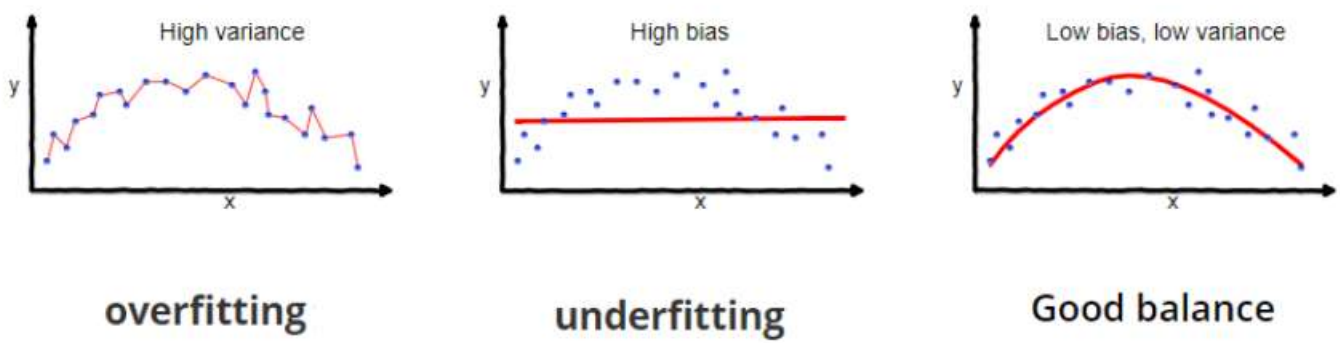
Balance the complexity of the model by the training error and validation error

Bias-Variance tradeoff



Underfitting and overfitting

- Underfitting: the model is not able to capture the underlying pattern of data
  - High bias, low variance
  - In our setting: high error rate on training data and validation data
- Overfitting: the model captures the noise with the underlying pattern of data
  - Low bias, high variance
  - In our setting: low error rate on training data, high error on validation data



Measures of classification accuary

- Average error rate
- Confusion matrices
- True/false positive/negatives
- Precision/recall and sensitivity/specificity
- ROC-curves

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

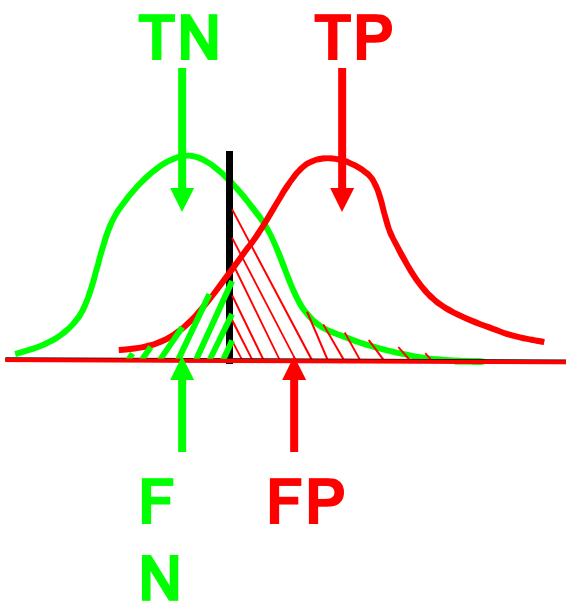
True class labels

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

True / False positives / negatives

- **True positive (TP):**
  Patient has cancer
  and test result is positive.
- True negative (TN):
  A healthy patient
  and a negative test result.
- **False positive (FP):**
  Healthy patient that gets a positive test result.
- **False negative (FN):**
  Cancer patient that gets a negative test result.

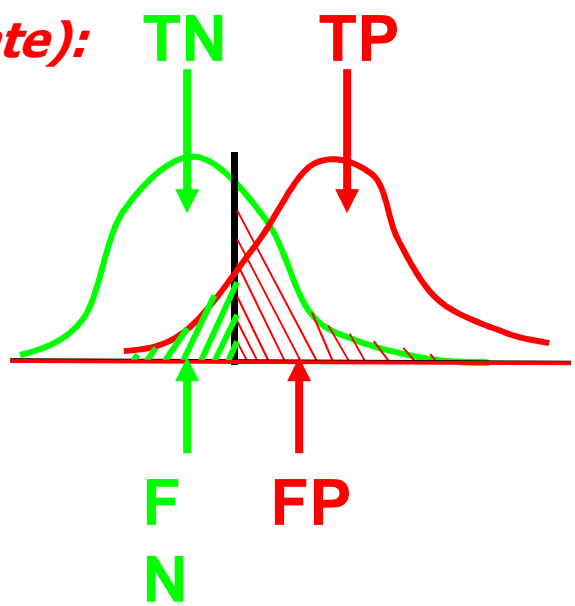
E.g., testing for cancer
No cancer | Cancer



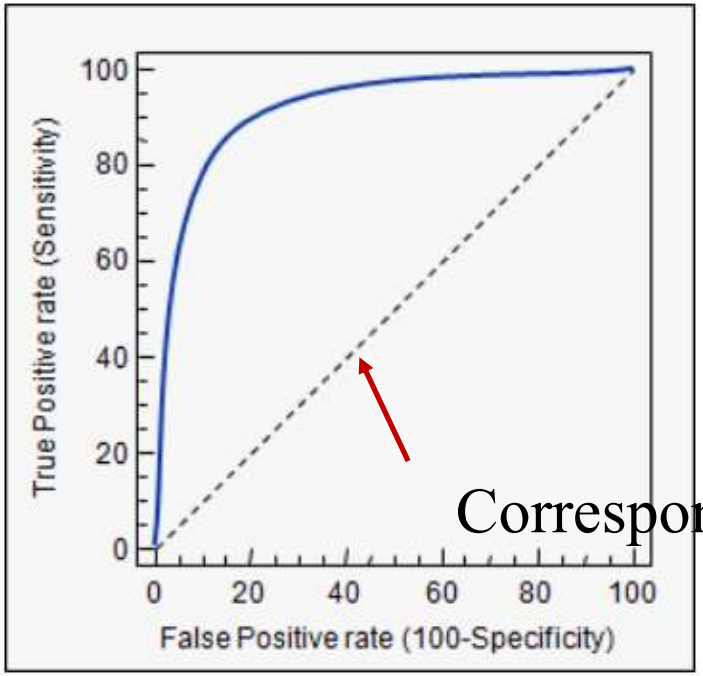
- *Good to have: **TP** & **TN***
- *Bad to have: **FP** (but this will probably be detected)*
- *Worst to have: **FN** **(may go un-detected)***

Sensitivity and specificity, F1-score

- ***Precision:***
  - **Precision = TP/(TP+FP)**
  - The probability that patient is really sick given that he is classified sick.
- ***Sensitivity (also called recall/true positive rate):***
  Sensitivity/TPR = TP/(TP+FN)
  - The probability that the test is positive given that the patient is sick.
  - Higher sensitivity means that fewer desease cases go undetected.
- ***Specificity: (also called true negative rate)***
  Specificity = TN/(TN+FP)
  - The probability that a test is negative given that the patient is not sick.
  - Higher specificity means that fewer healthy patients are labeled as sick.
- **Combined score**
  - F1= 2(precision*recall)/(precision+recall)



Receiver Operating Characteristic(ROC)-curve



Used for binary classification problems to set the detection threshold.

Good performance: the curve should be close to the upper left corner

Corresponds to random guess

Outliers and doubt

- In a classification problem, we might want to identify outliers and doubt samples

- We might want an ideal classifier to report
  - 'this sample is from class l' (usual case)
  - 'this sample is not from any of the classes' (outlier)
  - 'this sample is too hard for me' (doubt/reject)

- The two last cases should lead to a rejection of the sample!

Outliers

- Heuristically defined as "… samples which did not come from the assumed population of samples"
- The outliers can result from some breakdown in preprocessing.
- Outliers can also come from pixels from other classes than the classes in the training data set.
  - Example: K tree species classes, but a few road pixels divide the forest regions.
- One way to deal with outliers is to model them as a separate class, e.g., a gaussian with very large variance, and estimate prior probability from the training data
- Another approach is to decide on some threshold on the aposteriori probability– and if a sample falls below this threshold for all classes, then declare it an outlier.
- Related to normalization: a max/min normalization will be sensitive to outliers

Doubt samples

- Doubt samples are samples for which the class with the highest probability is not significantly more probable than some of the other classes (e.g. two classes have essentially equal probability).

- Doubt pixels typically occurr on the border between two classes (”mixels”)
  - Close to the decision boundary the probabilities will be almost equal.

- Classification software can allow the user to specify thresholds for doubt.

The ”curse” of dimensionality

– In practice, the curse means that, for a given sample size, there is a maximum number of features one can add before the classifier starts to degrade.

- For a finite training sample size, the correct classification rate initially increases when adding new features, attains a maximum and then begins to decrease.

- For a high dimensionality, we will need lots of training data to get the best performance.

- => ≈10 samples / feature / class.

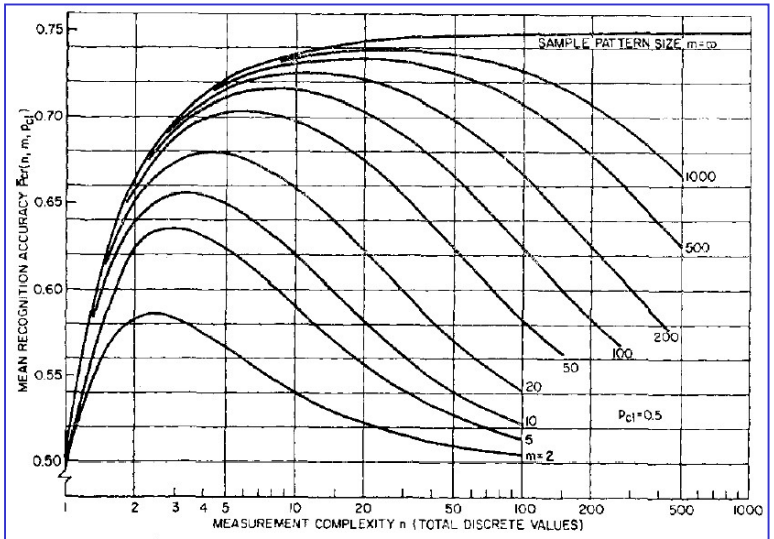


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

Correct classification rate as function of feature dimensionality, for different amounts of training data. Equal prior probabilities of the two classes is assumed.

Use few, but good features

- To avoid the ”curse of dimensionality” we must take care in finding a set of relatively few features.
- A good feature has high within-class homogeneity, and should ideally have large between-class separation.
- In practise, one feature is not enough to separate all classes, but a good feature should:
  - separate some of the classes well
  - Isolate one class from the others.
- If two features look very similar (or have high correlation), they are often redundant and we should use only one of them.
- Class separation can be studied by:
  - Visual inspection of the feature image overlaid the training mask
  - Scatter plots
- Evaluating features as done by training can be difficult to do automatically, so manual interaction is normally required.

How do we beat the ”curse of dimensionality”?

- Generate few, but informative features
  - Careful feature design given the application
- Get as much labelled data as possible!
- Try a simple classifier first
  - Do the features work? Do we need additional features?
  - Iterate between feature extraction and classification
- Reducing the dimensionality
  - Feature selection – select a subset of the original features
  - Feature transforms – compute a new subset of features based on a linear combination of all features
    - Example: Principal component transform
      - Unsupervised, finds the combination that maximizes the variance in the data.
- When you are confident that the features are good, consider a more advanced classifier.

Linear feature transforms

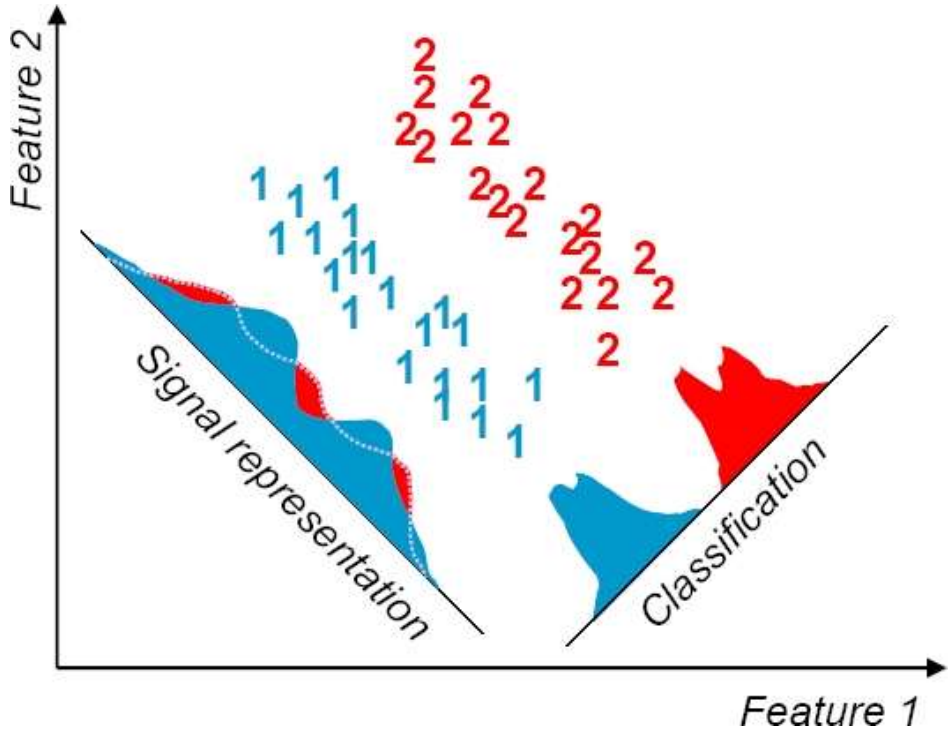
- Feature extraction can be stated as
  - Given a feature space $x_i \in \mathbb{R}_n$ find an optimal mapping $y = f(x) : \mathbb{R}_n \rightarrow \mathbb{R}_m$ with $m < n$.
  - An optimal mapping in classification :the transformed feature vector y yield the same classification rate as x.
- The optimal mapping may be a non-linear function
  - Difficult to generate/optimize non-linear transforms
  - Feature extraction is therefore usually limited to linear transforms $y = A^T x$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{11} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}$$

Signal representation vs classification

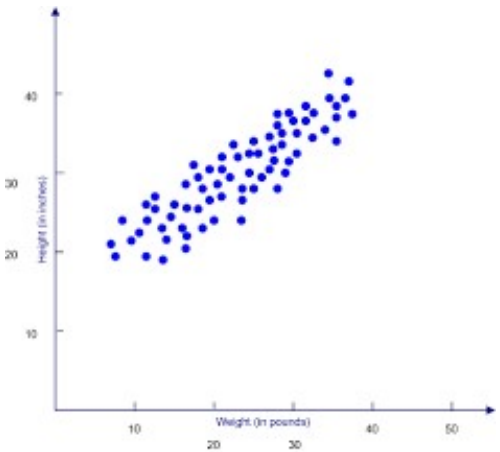
- Principal components analysis (PCA)
  - - signal representation, unsupervised
  - Minimize the mean square representation error (unsupervised)
- Linear discriminant analysis (LDA)
  - -classification, supervised
  - Maximize the distance between the classes (supervised)

    Nice theory, but rarely used as the number of selected features must be less than the number of classes
  - Not covered in this course



Idea behind (Principal Component Transform)

- Find a projection $\mathbf{y}=A^T\mathbf{x}$ of the feature vector $\mathbf{x}$
- Three interpretations of PCA:
  - Find the projection that maximize the variance along the selected projection
  - Minimize the reconstruction error (squared distance between original and transformed data)
  - Find a transform that gives uncorrelated features



Definitions: Correlation matrix vs. covariance matrix

- Σ_x is the covariance matrix of x

$$\Sigma_x = E\left[(x-\mu)(x-\mu)^T\right]$$

- R_x is the correlation matrix of x

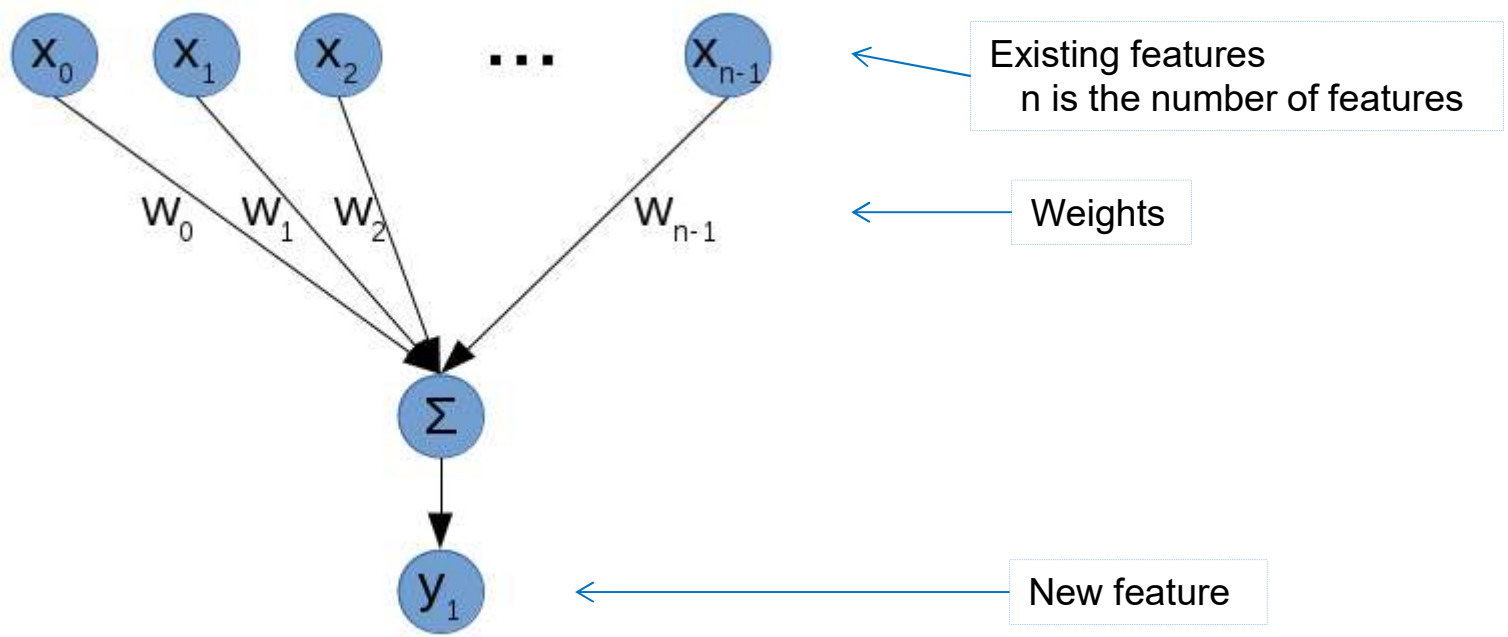
$$R_x = E\left[(x)(x)^T\right]$$

- $R_x = \Sigma_x$ if $\mu_x = 0$.

Principal component or Karhunen-Loeve transform - motivation

- Features are often correlated, which might lead to redundancies.
- We now derive a transform which yields **uncorrelated** features.
- We seek a linear transform $y=A^T x$, and the y_is should be uncorrelated.
- The y_is are uncorrelated if $E[y(i)y(j)^T]=0$, $i \neq j$.
- If we can express the information in x using uncorrelated features, we might need **fewer** coefficients.

Linear feature transforms I/II



From the original features to **one** new feature

Linear feature transforms II/II

- Multiple output features by applying **different weights for each one**:

$$y_1 = \sum_{i=0}^{n-1} w_{i1} x_i, \quad y_2 = \sum_{i=0}^{n-1} w_{i2} x_i, \quad \ldots \quad y_m = \sum_{i=0}^{n-1} w_{im} x_i$$

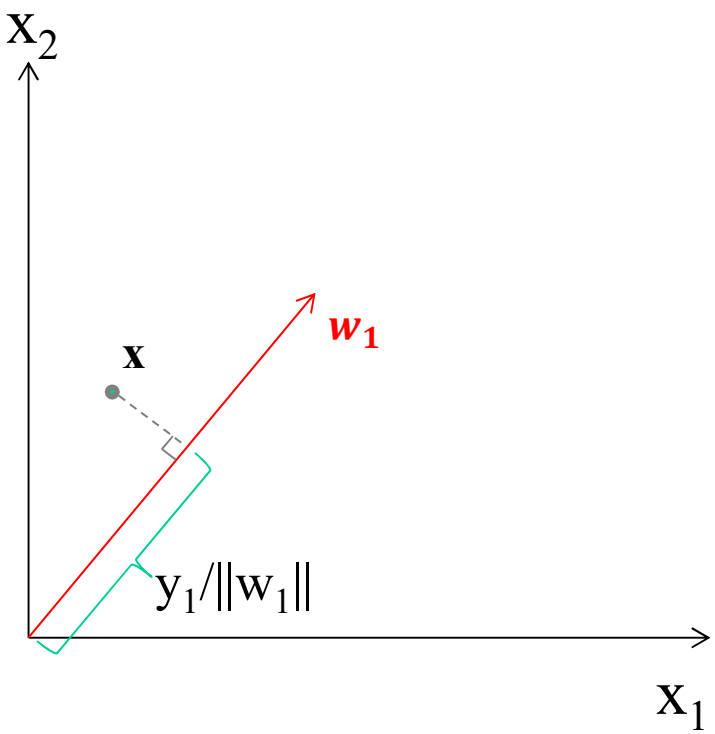
- In matrix notation $\mathbf{y} = \mathbf{A}^T\mathbf{x}$, $\mathbf{A}=[\mathbf{w}_1\ \mathbf{w}_2 \ldots \mathbf{w}_m]$

- If **y** has fewer elements than **x**, we get a feature reduction

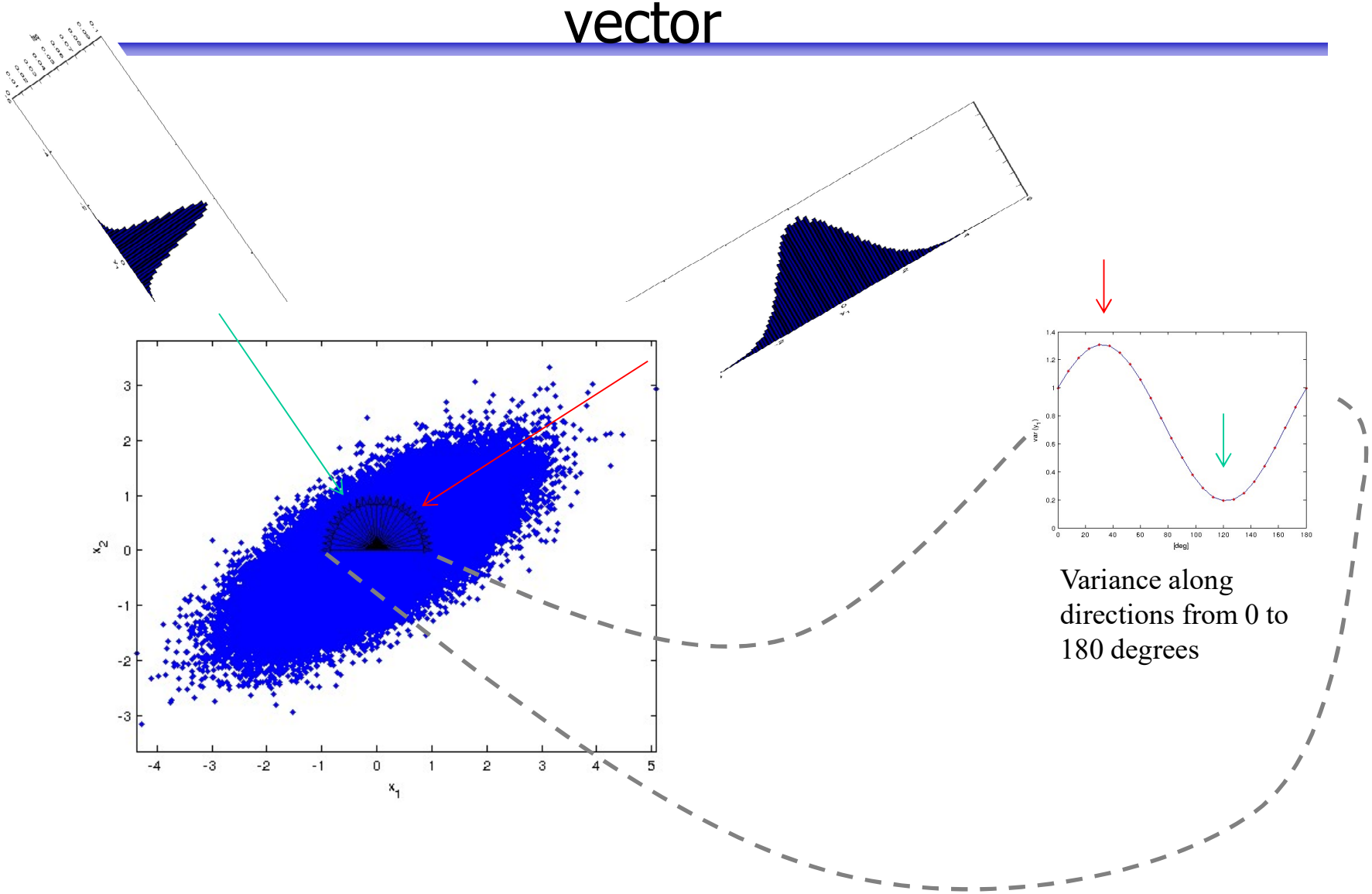
The weights | Visualization and intuition

$$y_1 = \sum_{i=0}^{n-1} w_{i1} x_i = \boldsymbol{w}_1{}^T \boldsymbol{x}$$

A linear transform is a shift of basis vectors
by projecting the points on the the basis vectors



Variance of y_1 as a function of direction of basis vector



Variance along directions from 0 to 180 degrees

Variance of y_1 cont.

- Assume mean of **x** is subtracted, so **x** has zero mean.
- The mean of the projections will also be zero:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{x_i}\cdot\vec{w})\vec{w} = \left(\left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)\cdot\vec{w}\right)\vec{w}$$

$$\begin{aligned}\sigma_{y_1}^2 &= \frac{1}{N}\sum_i y_i^2 \\ &= \frac{1}{N}\sum_i(\mathbf{w}^T\mathbf{x}_i)^2 = \frac{1}{N}\sum_i \mathbf{w}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{w} = \mathbf{w}^T\underbrace{\left(\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i^T\right)}\mathbf{w} \\ &= \mathbf{w}^T\mathbf{R}\mathbf{w}\end{aligned}$$

Called σ^2_w on some slides

The sample covariance matrix / scatter matrix; **R**

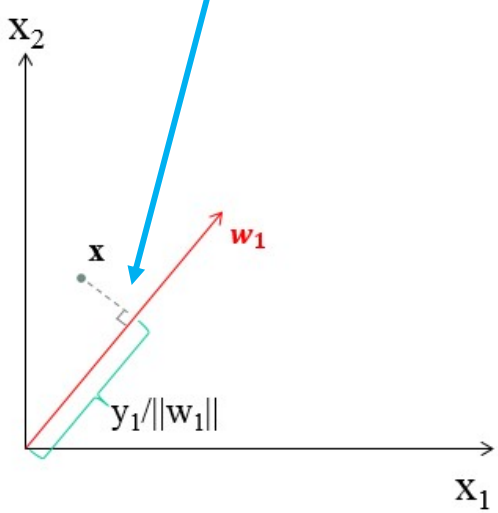
Variance and projection residuals

Single sample

Projection onto **w**, assuming |**w**|=1

$$
\begin{aligned}
\|\vec{x_i} - \underbrace{(\vec{w}\cdot\vec{x_i})}_{\text{«}y_i\text{»}}\vec{w}\|^2 &= (\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w})\cdot(\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}) \\
&= \vec{x_i}\cdot\vec{x_i} - \vec{x_i}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} \\
&\quad -(\vec{w}\cdot\vec{x_i})\vec{w}\cdot\vec{x_i} + (\vec{w}\cdot\vec{x_i})\vec{w}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} \\
&= \|\vec{x_i}\|^2 - 2(\vec{w}\cdot\vec{x_i})^2 + (\vec{w}\cdot\vec{x_i})^2\vec{w}\cdot\vec{w} \\
&= \vec{x_i}\cdot\vec{x_i} - \underbrace{(\vec{w}\cdot\vec{x_i})^2}_{\text{«}y_i^2\text{»}}
\end{aligned}
$$

w·w=1



Residuals – sum over all samples

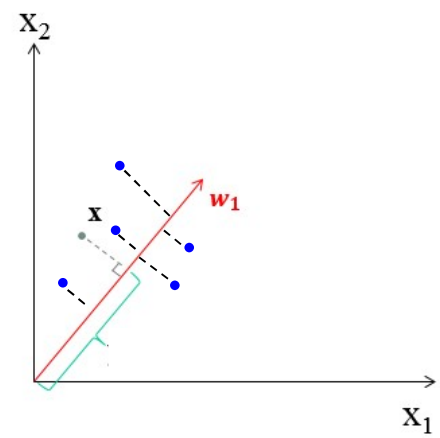
$$MSE(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}||\vec{x_i}||^2 - (\vec{w}\cdot\vec{x_i})^2$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n}||\vec{x_i}||^2 - \boxed{\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2}\right)$$

Sum all *n* samples (not dimensions)

Does not depend on w

Minimizing MSE is equivalent to maximizing this term



Maximizing the variance

- The mean of a square is always equal to the square of the mean plus the variance:

$$\boxed{\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2} = \left(\frac{1}{n}\sum_{i=1}^{n}\vec{x_i}\cdot\vec{w}\right)^2 + \mathrm{Var}\left[\vec{w}\cdot\vec{x_i}\right]$$

The mean of $(\vec{w}\cdot\vec{x_i})^2$

- But the projected data has zero mean, so this is equivalent to maximizing the variance.

Minimizing the squared errror is equivalent to maximizing the variance in the projected component

Criterion function

- Goal: Find transform minimizing representation error
- We start with a single weight-vector, **w**, giving us a single feature, y_1
- Let $J(\mathbf{w}) = \mathbf{w}^T\mathbf{R}\mathbf{w} = \sigma_w^2$
- Now, let's find $\max_{\boldsymbol{w}} J(\boldsymbol{w})$ $s.t. \left|\left|\boldsymbol{w}\right|\right| = 1$

As we learned on the previous slide, maximizing this is equivalent to minimizing representation error

Transform this problem into a unconstrained problem with a Lagrange multiplier (we skip details here)

Maximizing variance of y_1

$$\mathscr{L}(\mathbf{w},\lambda) \equiv \sigma_{\mathbf{w}}^2 - \lambda(\mathbf{w}^T\mathbf{w} - 1)$$

$$\frac{\partial L}{\partial \lambda} = -\mathbf{w}^T\mathbf{w} + 1$$

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathrm{R}\mathbf{w} - 2\lambda\mathbf{w}$$

Lagrangian function for maximizing σ^2_w with the constraint $\mathbf{w}^T\mathbf{w}=1$

$\Downarrow$ Equating zero

$$\mathbf{w}^T\mathbf{w} = 1$$

$$\mathrm{R}\mathbf{w} = \lambda\mathbf{w}$$

Unfamiliar with Lagrangian multipliers? See http://biostat.mc.vanderbilt.edu/wiki/pub/Main/CourseBios362/LagrangeMultipliers-Bishop-PatternRecognitionMachineLearning.pdf

The maximizing **w** is an eigenvector of R!

And $\sigma^2_w=\lambda$!

$\mathbf{w}_2$, $\mathbf{w}_3$, .. I/III

- $\mathbf{w}_1$ should be the eigenvector of $\mathbf{R_x}$ corresponding to the largest eigenvalue
- Ok, I've got the $\mathbf{w}_1$ giving me the transform (linear weights) that maximizes the variance / minimizes the representation error ..
- .. Now I want another one that again maximizes the variance / minimizes the representation error, but the new feature should be uncorrelated with my previous one ..
- .. Which $\mathbf{w}_2$ would give me this?

Eigendecomposition of covariance matrices

Real-valued, symmetric, «n-dimensional» covariance matrix

$$\mathbf{R} = \lambda_1 \mathbf{a}_1 \mathbf{a}_1' + \lambda_2 \mathbf{a}_2 \mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n \mathbf{a}_n'$$

Eigenvalue (let's say largest)

Eigenvector corresponding to λ_1

Smallest eigenvalue

$\mathbf{a}^T_i \mathbf{a}_j = 0$ for $i \neq j$

Remember: λ_i=variance of $\mathbf{x}^T \mathbf{a}_i$

Remark: $a' = a^T$

$\mathbf{w}_2$, $\mathbf{w}_3$, .. II/III

- What does uncorrelated mean? Zero covariance.
- Covariance of y_1 and y_2:

$$\frac{1}{N}\sum_{i}^{N} y_1(i)' y_2(i) = \frac{1}{N}\sum_{i}^{N} \mathbf{w}_1' \mathbf{x}(i)\mathbf{x}(i)' \mathbf{w}_2 = \mathbf{w}_1' \mathbf{R} \mathbf{w}_2$$

- We already have that $\mathbf{w}_1 = \mathbf{a}_1$
- From last slide, requiring $\mathbf{w}_1' \mathbf{R} \mathbf{w}_2 = \mathbf{a}_1' \mathbf{R} \mathbf{w}_2 = 0$ means requiring $\mathbf{w}_2' \mathbf{a}_1 = 0$

$\mathbf{w}_2$, $\mathbf{w}_3$, .. III/III

- We want $\max_{\mathbf{w}} \mathbf{w}'\mathbf{R}\mathbf{w}$, s.t. $|\mathbf{w}|=1$ *and* $\mathbf{w}'\mathbf{a}_1=0$

- We can simply remove $\lambda_1\mathbf{a}_1\mathbf{a}_1'$ from $\mathbf{R}$, creating $\mathbf{R}_{next} = \mathbf{R} - \lambda_1\mathbf{a}_1\mathbf{a}_1'$, and again find $\max_{\mathbf{w}} \mathbf{w}'\mathbf{R}_{next}\mathbf{w}$ s.t. $|\mathbf{w}|=1$

- Studying the decomposition of $\mathbf{R}$ (a few slides back), we see that the solution is the eigenvector corresponding to the second largest eigenvalue

- Similarly, the $\mathbf{w}_3$, $\mathbf{w}_4$ etc. are given by the following eigenvectors sorted according to their eigenvalues

$\mathbf{w}_2$, $\mathbf{w}_3$, .. III+/III

$\max_{\mathbf{w}}\ \mathbf{w}'\mathbf{R}\mathbf{w}$, s.t. $|\mathbf{w}|=1$

$$\mathbf{R} = \lambda_1 \mathbf{a}_1\mathbf{a}_1' + \lambda_2 \mathbf{a}_2\mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n\mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_1$$

$$\mathbf{R} = \cancel{\lambda_1 \mathbf{a}_1\mathbf{a}_1'} + \lambda_2 \mathbf{a}_2\mathbf{a}_2' + \ldots + \lambda_n \mathbf{a}_n\mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_2$$

$$\mathbf{R} = \cancel{\lambda_1 \mathbf{a}_1\mathbf{a}_1'} + \cancel{\lambda_2 \mathbf{a}_2\mathbf{a}_2'} + \ldots + \lambda_n \mathbf{a}_n\mathbf{a}_n' \quad \rightarrow \quad \mathbf{w}=\mathbf{a}_3$$

… etc.

Eigenvectors sorted by their corresponding eigenvalues

Principal component transform (PCA)

- Place the *m* «principle» eigenvectors (the ones with the largest eigenvalues) along the columns of A
  - They are given as the eigenvectors of the covariance matrix R

- Then the transform $\mathbf{y} = \mathbf{A}^T\mathbf{x}$ gives you the *m* first principle components

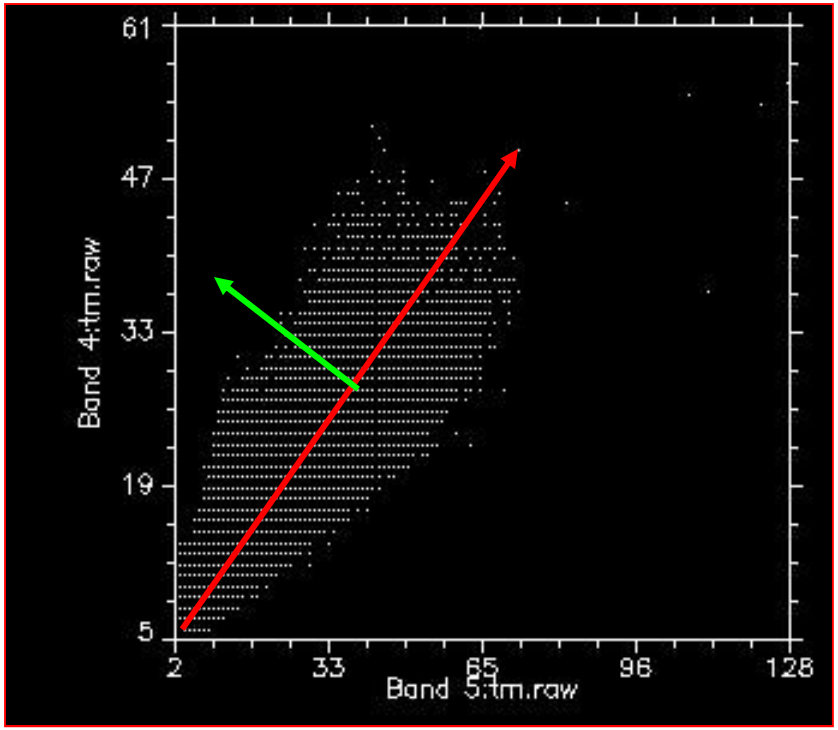
- The *m*-dimensional **y**
  - have uncorrelated elements
  - retains as much variance as possible
  - gives the best (in the mean-square sense) description of the original data (through the «image»/projection/reconstruction **Ay**)

Note: The eigenvectors themselves can often give interesting information

PCA is also known as Karhunen-Loeve transform

Geometrical interpretation of principal components

- The eigenvector corresponding to the largest eigenvalue is the direction in n-dimensional space with highest variance.
- The next principal component is orthogonal to the first, and along the direction with the second largest variance.

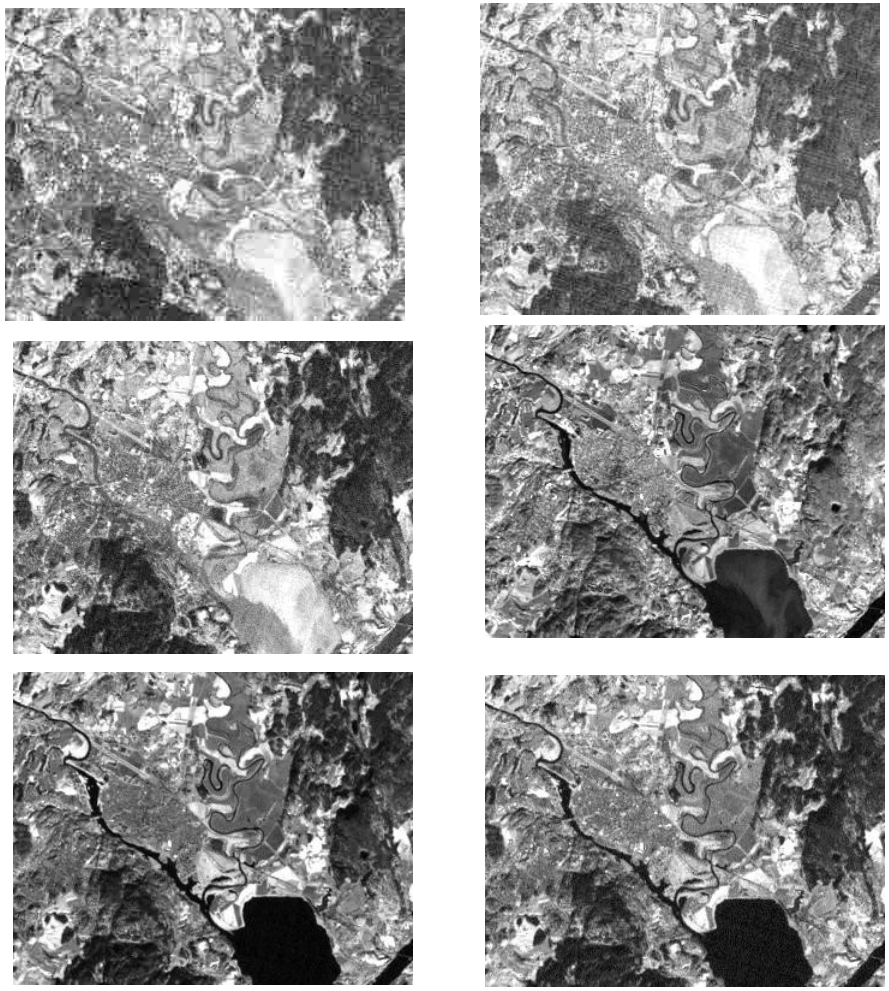


Note that the direction with the highest variance is NOT related to separability between classes.

PCA and multiband images

- We can compute the principal component transform for an image with *n* bands
- Let **X** be an *N*x*n* matrix having a row for each image sample
- Sample covariance matrix (after mean subtracted): $\boldsymbol{R} = \frac{\mathbf{1}}{\boldsymbol{N}} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$
- Place the (sorted) eigenvectors along the columns of **A**
- **Y**=**XA** will then contain the image samples, however most of the variance is in the «bands» with the lowest index (corresponding to the largest eigenvalues), and the new features are uncorrelated

PCA example – original image

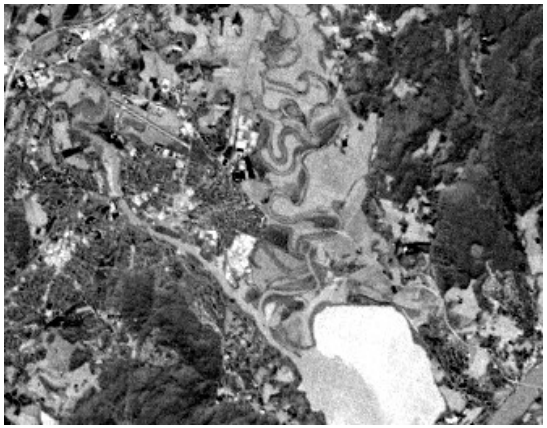


- Satellite image from Kjeller
- 6 spectral bands with different wavelengths

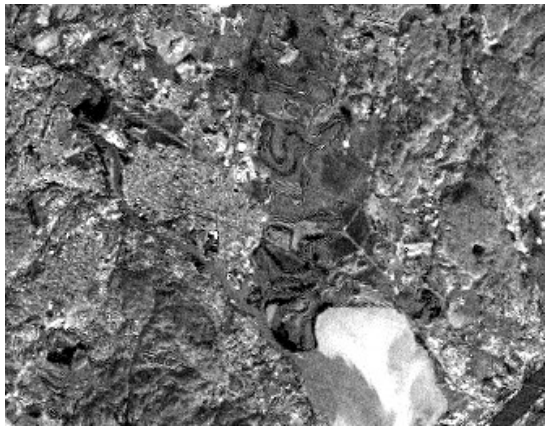
1	Blue	0.45-0.52	Max. penetration of water
2	Green	0.52-0.60	Vegetation and chlorophyll
3	Red	0.63-0.69	Vegetation type
4	Near-IR	0.76-0.90	Biomass
5	Mid-IR	1.55-1.75	Moisture/water content in vegetation/soil
7	Mid-IR	2.08-2.35	Minerals

Example cont: Principal component images

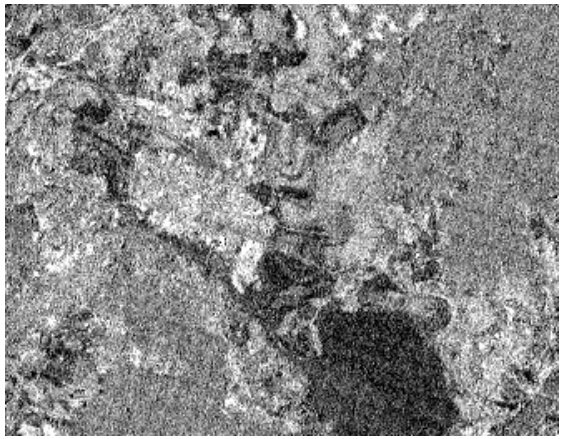
Principal component 1



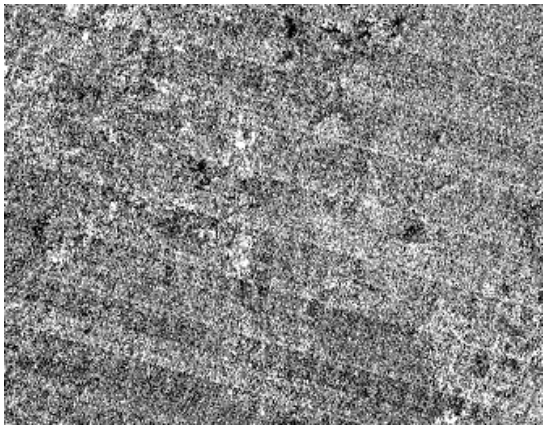
Principal component 2



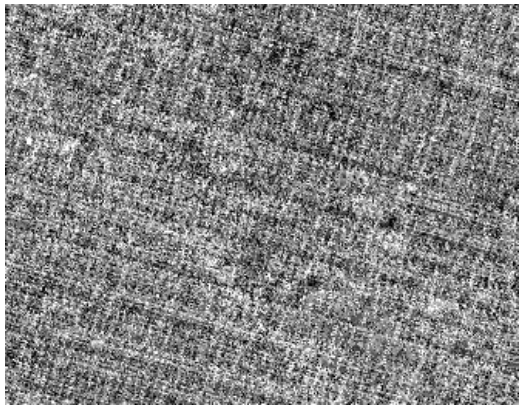
Principal component 3



Principal component 4



Principal component 5

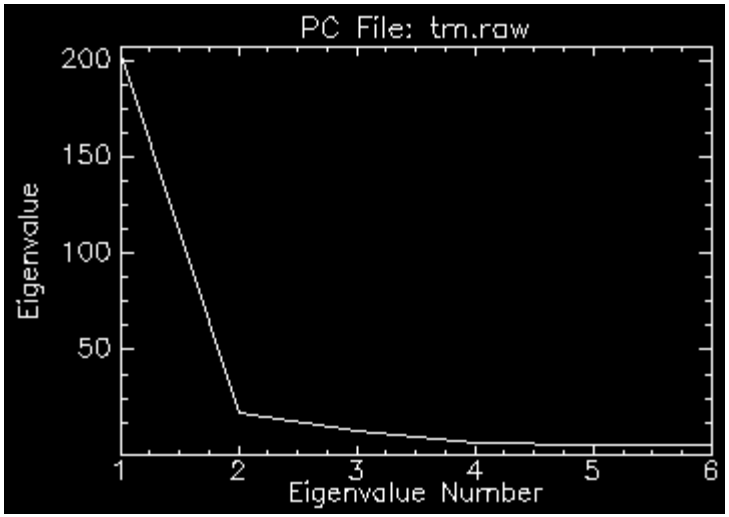


Principal component 6

Example cont: Inspecting the eigenvalues

The mean-square representation error we get with m of the N PCA-components is given as

$$E\left[\|x-\hat{x}\|^2\right]=\sum_{i=1}^{N-1}\lambda_i-\sum_{i=1}^{m}\lambda_i=\sum_{i=m}^{N-1}\lambda_i$$

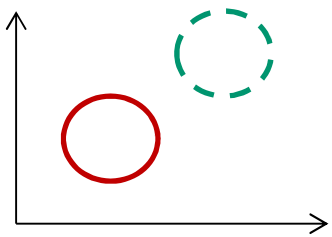


Plotting λs will give indications on how many features are needed for representation (explaining the variance)

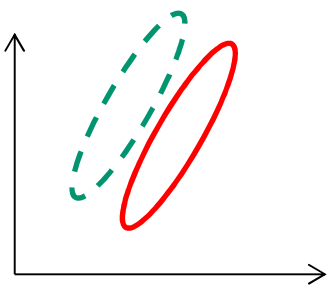
PCA and classification

- Reduce overfitting by detecting directions/components without any/very little variance

- Sometimes high variation means useful features for classification:



- .. and sometimes not:



Exhaustive feature subset selection

- If – for some reason – you know that you will use d out of D available features, an exhaustive search will involve a number of combinations to test:

$$n = \frac{D!}{(D-d)!\,d!}$$

d!=1*2*..*d

- If we want to perform an exhaustive search through D features for the optimal subset of the d ≤ m "best features", the number of combinations to test is

$$n = \sum_{d=1}^{m} \frac{D!}{(D-d)!\,d!}$$

- Impractical even for a moderate number of features!

d ≤ 5, D = 100 => n = 79.374.995

Suboptimal feature selection

- Select the best single features based on some quality criteria, e.g., estimated correct classification rate.
  - A combination of the best single features will often imply correlated features and will therefore be suboptimal .
- "Sequential forward selection" implies that when a feature is selected or removed, this decision is final.
- "Stepwise forward-backward selection" overcomes this.
  - A special case of the "add - a, remove - r algorithm".
- Improved into "floating search" by making the number of forward and backward search steps data dependent.
  - "Adaptive floating search"
  - "Oscillating search".

Distance measures used in feature selection

- In feature selection, each feature combination must be ranked based on a criterion function.
- Criteria functions can either be distances between classes, or the classification accuracy on a validation test set.
- If the criterion is based on e.g. the mean values/covariance matrices for the training data, distance computation is fast.
- Better performance at the cost of higher computation time is found when the classification accuracy on a validation data set (different from training and testing) is used as criterion for ranking features.
  - This will be slower as classification of the validattion data needs to be done for every combination of features.

Distance measures between classes

- How do we compute the distance between two classes:
  - Distance between the closest two points?
  - Maximum distance between two points?
  - Distance between the class means?
  - Average distance between points in the two classes?
  - Which distance measure?
    - Euclidean distance or Mahalanobis distance?
- Distance between K classes:
  - How do we generalize to more than two classes?
  - Average distance between the classes?
  - Smallest distance between a pair of classes?

Class separability measures

- How do we get an indication of the separability between two classes?
  - Euclidean distance between class means $|\mu_r - \mu_s|$
  - Bhattacharyya distance
    - Can be defined for different distributions
    - For Gaussian data, it is

$$B = \frac{1}{8}(\mu_r - \mu_s)^T \left(\frac{\Sigma_r + \Sigma_s}{2}\right)^{-1} (\mu_r - \mu_s) + \frac{1}{2}\ln \frac{\left|\frac{1}{2}(\Sigma_r + \Sigma_s)\right|}{\sqrt{|\Sigma_r||\Sigma_s|}}$$

  - Mahalanobis distance between two classes:

$$\Delta = (\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2)$$

$$\Sigma = N_1 \Sigma_1 + N_2 \Sigma_2$$

Examples of feature selection - Method 1 - Individual feature selection

- Each feature is treated individually (no correlation/covariance between features is consideren)
- Select a criteria, e.g. a distance measure
- Rank the feature according to the value of the criteria C(k)
- Select the set of features with the best individual criteria value
- Multiclass situations:
  - Average class separability or
  - C(k) = min distance(i,j) - worst case ← Often used

- Advantage with individual selection: computation time
- Disadvantage: no correlation is utilized.

Method 2 - Sequential backward selection

- Select l features out of d
- Example: 4 features x_1,x_2,x_3,x_4
- Choose a criterion C and compute it for the vector $[x_1,x_2,x_3,x_4]^T$
- Eliminate one feature at a time by computing $[x_1,x_2,x_3]^T$, $[x_1,x_2,x_4]_T$, $[x_1,x_3,x_4]^T$ and $[x_2,x_3,x_4]^T$

- Select the best combination, say $[x_1,x_2,x_3]^T$.

- From the selected 3-dimensional feature vector eliminate one more feature, and evaluate the criterion for $[x_1,x_2]^T$, $[x_1,x_3]_T$, $[x_2,x_3]^T$ and select the one with the best value.
- Number of combinations searched:

  $1+1/2((d+1)d-l(l+1))$

Method 3: Sequential forward selection

- Compute the criterion value for each feature. Select the feature with the best value, say x_1.
- Form all possible combinations of features x1 (the winner at the previous step) and a new feature, e.g. $[x_1,x_2]^T$, $[x_1,x_3]^T$, $[x_1,x_4]^T$, etc. Compute the criterion and select the best one, say $[x_1,x_3]^T$.
- Continue with adding a new feature.
- Number of combinations searched: ld-l(l-1)/2.
  - Backwards selection is faster if l is closer to d than to 1.

Learning goals for this lecture

- Understand how different measures of classification accuracy work:
  - Confusion matrix
  - Sensitivity/specifity/TP/TN/FP/FN/ROC
  - Average classification accuracy
- Be familiar with the curse of dimensionality and the importance of selecting few, but good features
- Know simple forward and backward feature selection.
- Know how PCA works
  - Max variance <-> min projection error
  - Eigenvectors of sample cov.mat. / scatter matrix

IN 5520 – Support Vector Machine Classifiers (SVM)

Anne Solberg (anne@ifi.uio.no)

21.10.20

- Linear classifiers with maximum margin for two-class problems
- The kernel trick – from linear to a high-dimensional generalization
- Generation from 2 to M classes
- Practical issues/Parameter estimation

Mandatory 2 out now

- Deadline November 9
- Any questions??
- Remark: png-file converions will be updated

Curriculum

- Lecture foils are most important!
- The lectures are based on selected sections from Pattern Recognition, Third Edition, by Theodoridis and Koutroumbas:
  - 3.1-3.2, 3.7 (but 3.7.3 is a SVM-variant that we will skip)
  - 4.17
  - These sections use optimization theory described in Appendic C. We only include enough mathematics to state the optimization problem, and you are not required to understand how this optimization is solved.
  - Another useful note: Andrew Ng's note http://cs229.stanford.edu/notes/cs229-notes3.pdf

From last week: Case 1: $\Sigma_j=\sigma^2 I$

- An equivalent formulation of the discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t \mathbf{x} + w_{i0}$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } w_{i0} = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0) = 0$$

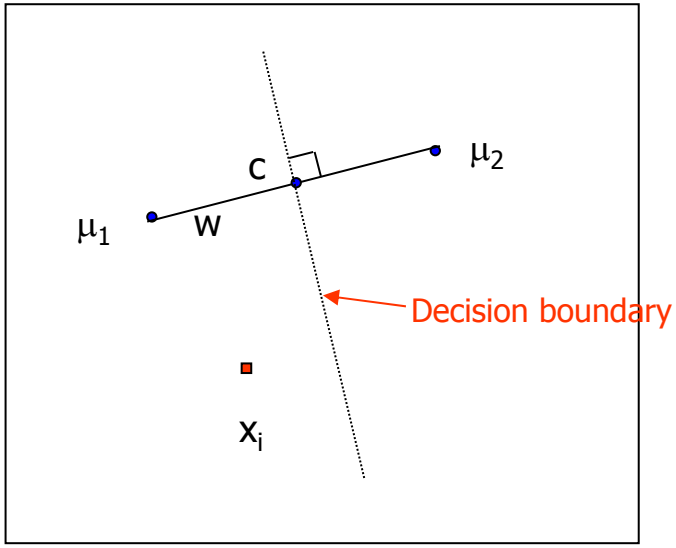
$$\text{where } \mathbf{w} = \boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}$$

$$\text{and } x_0 = \frac{1}{2}\left(\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right) - \frac{\sigma^2}{\left\|\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}\left(\boldsymbol{\mu}_\mathrm{i} - \boldsymbol{\mu}_\mathrm{j}\right)$$

- $\mathbf{w}=\mu_i-\mu_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.

Gaussian model with $\Sigma=\sigma I$
Linear decision boundary

- We found that the discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $c=(\mu_1 - \mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



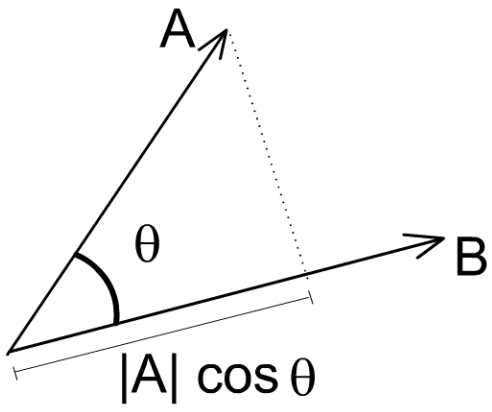
Reminder: Linear algebra basics: Inner product between two vectors.

- The inner product (or dot product) between two vectors (of length N) a and b or is given by

$$\langle a,b\rangle = \sum_{i=1}^{N} a_i b_i = a^T b$$

- The angle between two vectors A and B is defined as:

$$\cos\theta = \frac{\langle A,B\rangle}{\|A\|\|B\|}$$



- If the inner product of two vectors is zero, they are normal to each other.
- If two vectors are similar (small angle), cos θ will be close to 1 : the inner product measures similarity

Introduction to Support Vector Machine classifiers

- To understand Support Vector Machine (SVM) classifiers, we need to study the linear classification problem in detail.
- We will need to see how classification using this (the linear case on the previous slide) can be computed using inner products.
- We start with two linearly separable classes.
- Extension to two non-separable classes.
- Extension to M classes.
- The last step will be to use kernels to separate classes that cannot be separated in the input space.

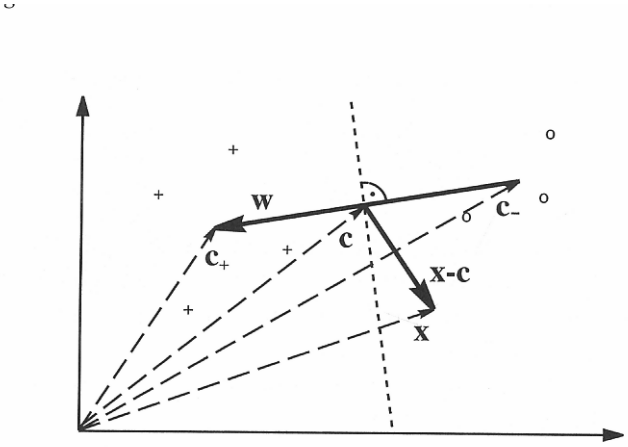
A new view at linear Gaussian classification using inner products

- We have two classes (+ and -) represented by the class means c_+ and c_-.
- Let the feature vector be x_i, and let y_i be the class of feature vector i.
- If we have m_+ samples from class + and m_- samples from class -, the class means are given by

$$c_+ = \frac{1}{m_+} \sum_{\{i|y_i=+)} x_i$$

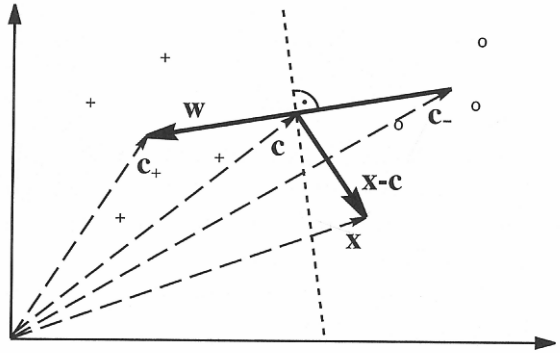
$$c_- = \frac{1}{m_-} \sum_{\{i|y_i=-)} x_i$$

- A new pattern x should be classified to the class with the closest mean.



- Half way between the means lies the point $c=(c_+ + c_-)/2$.

  If a point x is on the decision boundary hyperplane, then $w^T x=0$.



We can compute the class of a new sample x by checking whether the vector x-c (connecting x to c) encloses an angle smaller than $\pi/2$ (in terms of absolute value) with the vector $w=c_+ - c_-$.

This angle is given by the inner product between w and x-c: $w^T(x-c)$

- The angle between two vectors is computed by the inner product, which changes sign as the angle passes through $\pi/2$.

Support Vector Machines
Two linear separable classes

- Let x_i, i=1,...N be all the l-dimensional feature vectors in a training set with N samples.
- These belong to one of two classes, ω_1 and ω_2.
- We assume that the classes are **linearly separable**.
- This means that a hyperplane

  $g(x)=w^T x+w_0=0$

  correctly classifies all these training samples.
- $w=[w_1,...w_l]$ is called a weight vector, and w_0 is the threshold.

Introduction to linear SVM

- Discriminant function:
  $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$

  Weights/orientation

  Threshold/bias

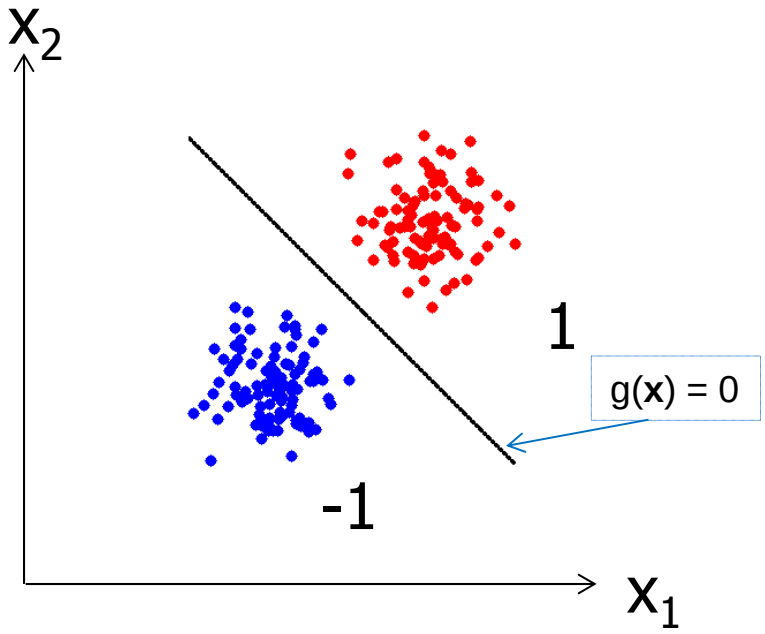
- Two-class problem,
  $y_i \in \{-1,1\}$

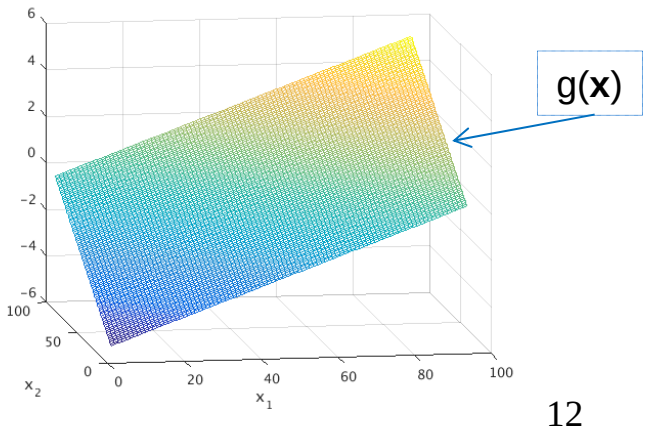
  Class indicator for pattern *i*

- $y_i = \begin{cases} -1, & g(\mathbf{x}_i) < 0 \\ 1, & g(\mathbf{x}_i) > 0 \end{cases}$

  Class prediction

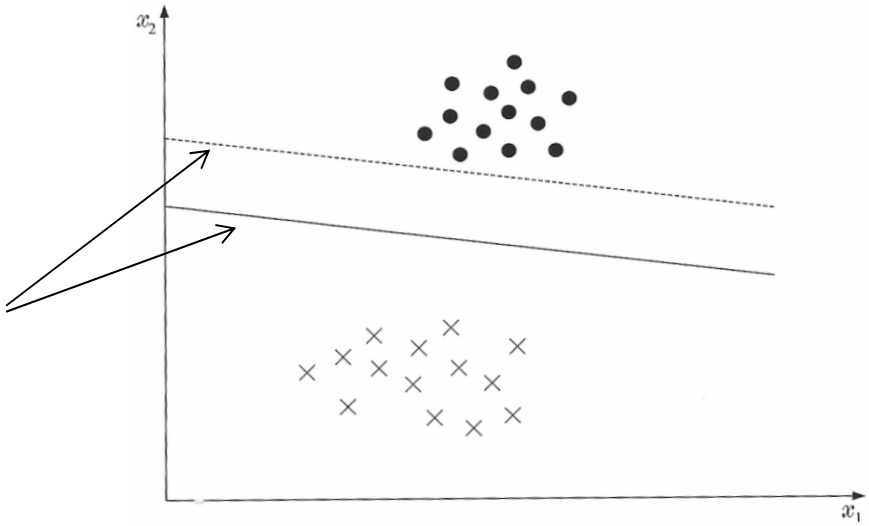
  Input pattern





Separable case: Many candidates

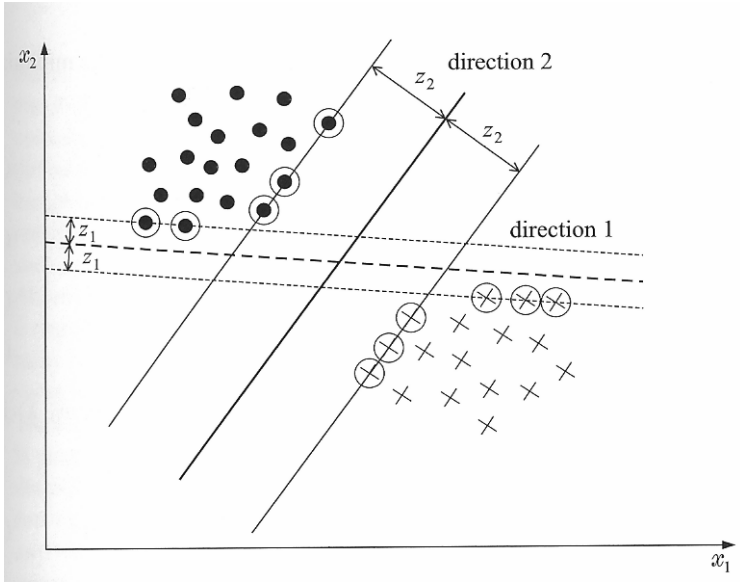
- Obviously we want the decision boundary to separate the classes ..
- .. however, there can be many such hyperplanes.
- Which of these two candidates would you prefer? Why?



Hyperplanes and margins

- If both classes are equally probable, the **distance from the hyperplane to the closest points** in both classes should be equal. This is called the margin.

- SVM finds the decision boundary that maximize the distance from the closest points in both classes to the boundary.



Goal: Find **w** and w_0 maximizing the margin!

How would you write a program finding this? Not easy unless we state the objective function cleverly!

Distance to the decision boundary,z

- Since w/||w|| is a unit vector in the direction w,

  B=x-z*w/||w||

- Because B lies on the decision boundary,

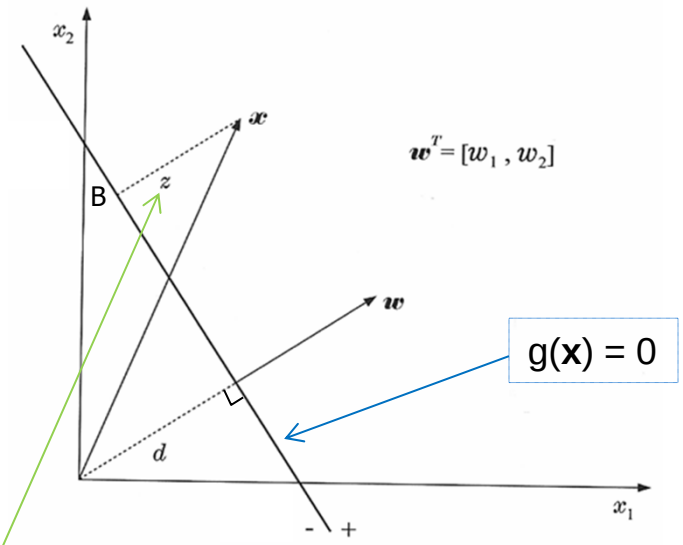
  $w^T B + w_0 = 0$

$$w^T\left(x - z\frac{w}{\|w\|}\right) + w_0 = 0$$

Solve this for z :

$$z = \frac{w^T x + w_0}{\|w\|} = \left(\frac{w}{\|w\|}\right)^T x + \frac{w_0}{\|w\|}$$

z is called the **margin** of the classifier



g(**x**) = 0

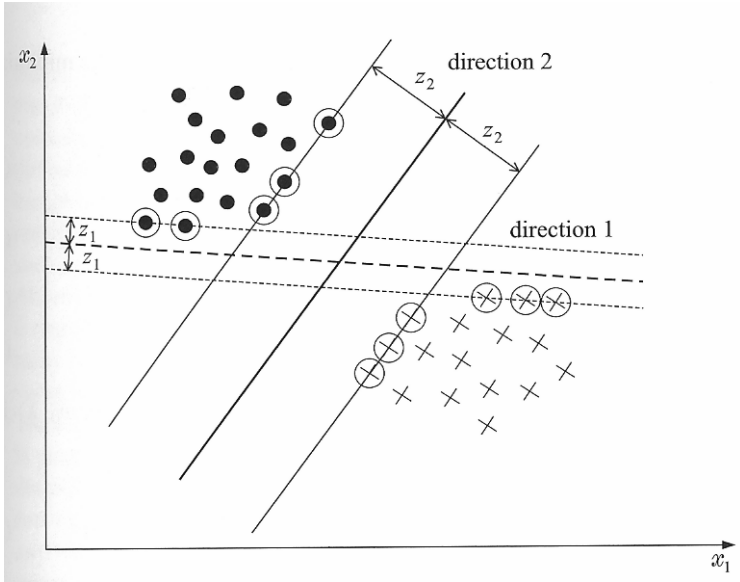
Distance from **x** to the decision boundary

Hyperplanes and margins

- The margin for «direction 1» is $2z_1$, and for «direction 2» it is $2z_2$.
- $g(x) = \mathbf{w}^T\mathbf{x}+w_0$

- The distance from a point to the separating hyperplane is

- $z = \frac{w^T x + w_0}{\|w\|} = \frac{g(w)}{\|w\|}$



Goal: Find **w** and w_0 maximizing the margin!

How would you write a program finding this? Not easy unless we state the objective function cleverly!

Rescaling g(x)

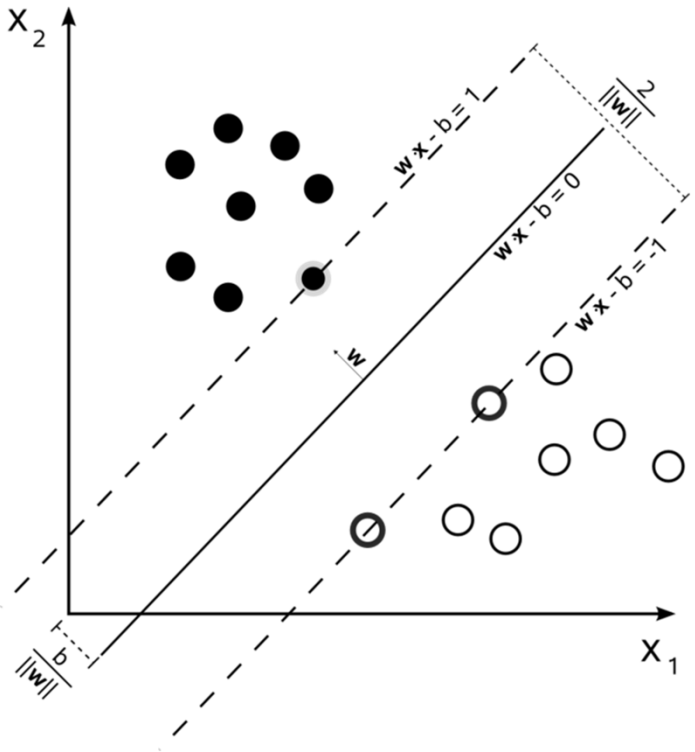
Does not change the margin

- We can scale g(**x**) such that g(**x**) will be equal to 1 or -1 at the closest points in the two classes. This is equivalent to:

1. Have a margin of $\frac{1}{\|w\|}+\frac{1}{\|w\|}=\frac{2}{\|w\|}$

2. Require that

$$g(x)=w^T x+w_0 \geq 1, \qquad \forall x \in \omega_1$$
$$g(x)=w^T x+w_0 \leq -1, \qquad \forall x \in \omega_2$$



Remember our goal: Find **w** and w_0 yielding the maximum margin
NOTE: in this figure, b is used instead of w0

Maximum-margin objective function

- The hyperplane with maximum margin can be found by solving the optimization problem (w.r.t. **w** and w_0):

The ½ factor is for later convenience

$$\begin{aligned} &\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2 \\ &\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N \end{aligned}$$

Note: We assume here fully class-**separable** data!

- Checkpoint: Do you understand the formulation?
- How is this criterion related to maximizing the margin?

Note! We are somewhat done -- Matlab (or similar software) can solve this now.

But we seek more insight!

QUESTIONS
ANSWERS

The optimization problem with margins

$$\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$$

$$\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N$$

- This is a **quadratic optimization** task with a set of **linear inequality contraints**.
- It can be shown that the solution has the form:

$$\mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i x_i \quad \text{where} \quad \sum_{i=1}^{N} \lambda_i y_i = 0$$

- The λ_i's are called Lagrange multipliers.
- The λ_i's can be either 0 or positive.
- We see that the solution w is a <u>linear combination of $N_s \leq N$ feature vectors associated with a $\lambda_i > 0$.</u>

Solving the optimization problem

- The optimization problem $\text{minimize} \quad J(w) = \frac{1}{2}\|w\|^2$

  $\text{subject to} \quad y_i(w^T x_i + w_0) \geq 1, \quad i = 1,2,\ldots N$

  has a dual representation with equality constraints:

$$\max_{\lambda} \boxed{\left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i{}^T x_j \right)} \longleftarrow \mathrm{L}$$

$$\text{subject to} \quad \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}}\, y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

- This is easier to solve and can be reformulated as:

$$\text{maximize} \quad L(w, w_0, \lambda)$$

$$\text{subject to} \quad \mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i x_i$$

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

- L is called the Lagrange multipliers
- They can be either 0 or positive.
- The solution w can be expressed as a linear compination of Ns<N feature vectors with $\lambda_i>0$
- These are known as the support vectors of the problem

$$\text{maximize} \quad L(w, w_0, \lambda)$$

$$\text{subject to} \quad \mathrm{w} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i x_i$$

$$\sum_{\mathrm{i}=1}^{\mathrm{N}} \lambda_{\mathrm{i}} y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

Solving the optimization problem

This is easier to solve and can be reformulated as:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \boxed{x_i^T x_j}\right)$$

$$\text{subject to} \quad \sum_{i=1}^{N}\lambda_i y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \forall i$$

- Note that the training samples x_i and x_j occurr as inner products of pairs of feature vectors. **The solution does not depend on the dimensionality of the feature vector, only on the scalar inner product.**
- The computational complexity can be expected to depend on the number of pixels in the training data set, N.
- In this setting we just accept that the solution can be found in optimization theory.
- Matlab and python libraries are used.

Support vectors

- The feature vectors $\mathbf{x}_i$ with a corresponding $\lambda_i>0$ are called the support vectors for the problem.

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i$$

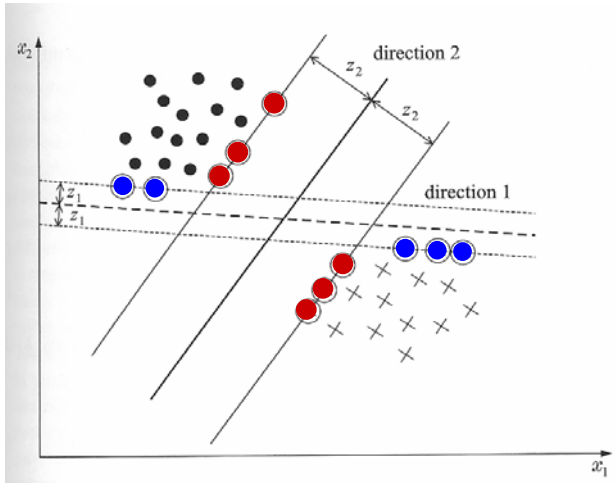
- The classifier defined by this hyperplane is called a Support Vector Machine.

- Depending on y_i (+1 or -1), the support vectors will thus lie on either of the two hyperplanes

  $\mathbf{w}^T\mathbf{x}+w_0=\pm 1$

- The support vectors are the points in the training set that are closest to the decision hyperplane.

- The optimization has a unique solution, only one hyperplane satisfies the conditions.



The support vectors for hyperplane 1 are the blue circles.

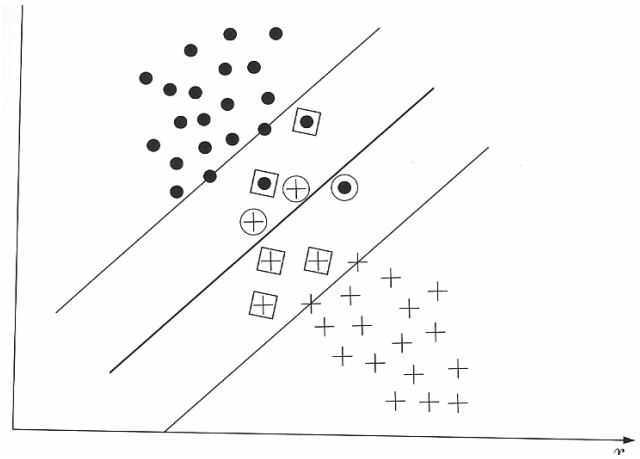
The support vectors for hyperplane 2 are the red circles.

QUESTIONS
ANSWERS

The nonseparable case

- If the two classes are nonseparable, a hyperplane satisfying the conditions $w^T x - w_0 = \pm 1$ cannot be found.
- The feature vectors in the training set are now either:

1. Vectors that fall outside the band and are correctly classified.
2. Vectors that are inside the band and are correctly classified. They satisfy $0 \leq y_i(w^T x + w_0) < 1$ □
3. Vectors that are misclassified – expressed as $y_i(w^T x + w_0) < 0$ ○



□ Correctly classified

○ Erroneously classified

- The three cases can be treated under a single type of contraints if we introduce slack variables ξ_i:

$$y_i[w^T x + w_0] \geq 1 - \xi_i$$

  - The first category (outside, correct classified) have $\xi_i=0$
  - The second category (inside, correct classified) have $0 \leq \xi_i \leq 1$
  - The third category (inside, misclassified) have $\xi_i > 1$

- The optimization goal is now *to keep the margin as large as possible and the number of points with $\xi_i > 0$ as small as possible.*

Cost function – nonseparable case

- The cost function to minimize is now

$$J(w, w_0, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

$$\text{where} \qquad \mathrm{I}(\xi_i) = \begin{cases} 1 & \xi_\mathrm{i} > 0 \\ 0 & \xi_\mathrm{i} = 0 \end{cases}$$

and ξ is the vector of parameters ξ_i.

- C is a parameter that controls how much misclassified training samples is weighted.
- We skip the mathematics and present the alternative dual formulation:

$$\max_{\lambda}\left(\sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j x_i^T x_j\right)$$

$$\text{subject to} \quad \sum_{\mathrm{i}=1}^{\mathrm{N}}\lambda_\mathrm{i} y_i = 0 \quad \text{and} \quad 0 \le \lambda_i \le C \quad \forall i$$

- All points between the two hyperplanes (ξ_i>0) can be shown to have λ_i=C.

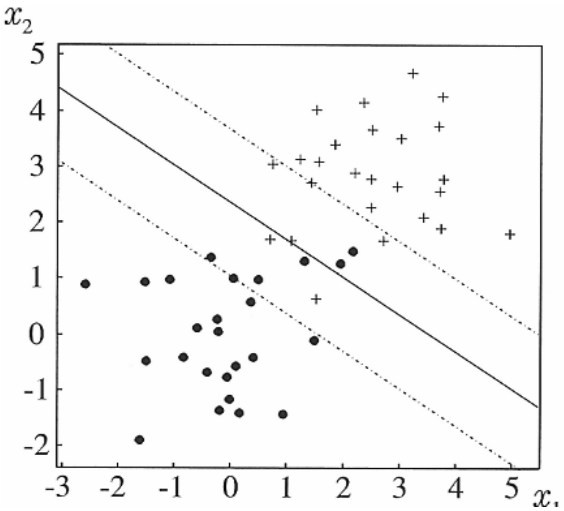
Nonseparable vs. separable case

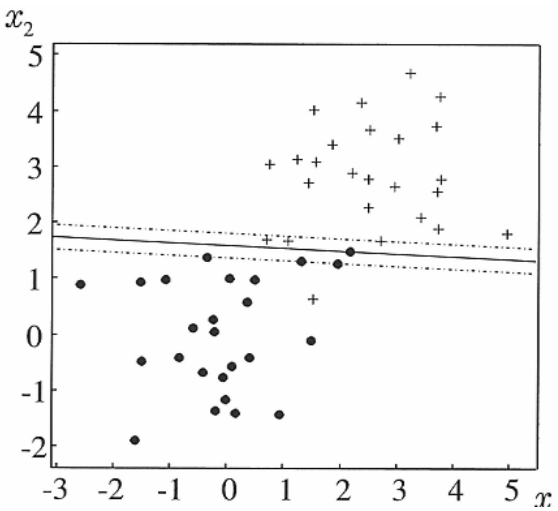
- Note that the slack variables ξ_i does not enter the dual problem explicitly.
- The only difference between the linear separable and the non-separable case is that the Lagrange-multipliers are bounded by C.
- Training a SVM classifier consists of solving the optimization problem.
  - The problem is quite complex since it grows with the number of training pixels.
  - It is computationally heavy.

An example – the effect of C

- C is the misclassification cost.

C=0.2





C=100

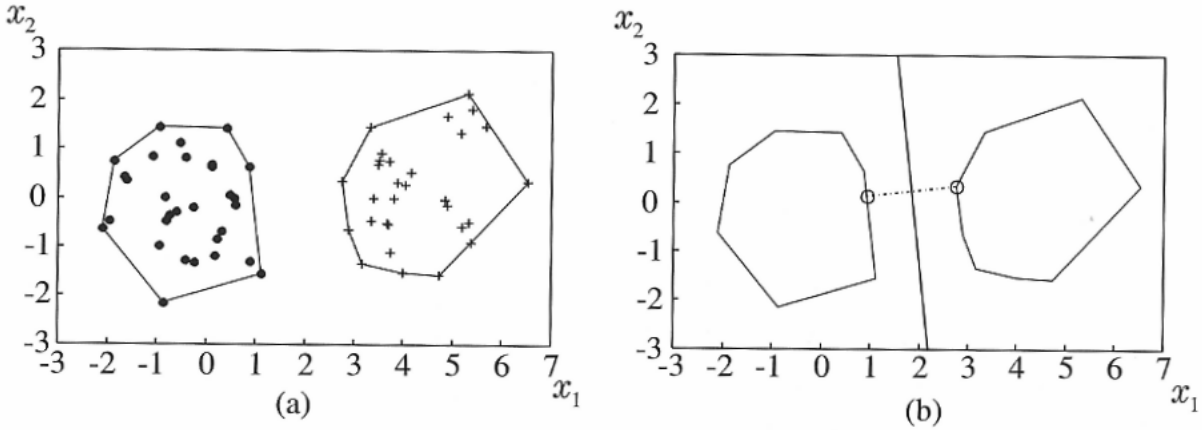
- Selecting too high C will give a classifier that fits the training data perfect, but fails on different data set.

- The value of C should be selected on the validation set.

QUESTIONS
ANSWERS

SVM: A geometric view

- SVMs can be related to the convex hull of the different classes. Consider a class that contains training samples $X=\{x_1,\ldots x_N\}$.
- The convex hull of the set of points in X is given by all convex combinations of the N elements in X.
  - A region *R* is convex if and only if for any two points x_1, x_2 in *R*, the whole line segment between x_1 and x_2 is inside the *R*.
  - The convex hull of a region is the smalles convex region H which satisfies the conditions $R \subseteq H$.



- The convex hull for a class is the smallest convex set that contains all the points in the class (X).
- Searching for the hyperplane with the highest margin is equivalent to <u>searching for the two nearest points in the two convex sets.</u>
  - This can be proved, but we just take the result as an aid to get a better visual interpretation of the SVM hyperplane.

Reduced convex hull

- To get a useable interpretation for nonseparable classes, we need the reduced convex hull.
- The convex set can be expressed as:

$$conv\{X\} = \{y : \; y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X, \; \sum_{i=1}^{N} \lambda_i = 1, \;\; 0 \le \lambda_i \le 1 \;\}$$

- The **reduced convex hull** is :

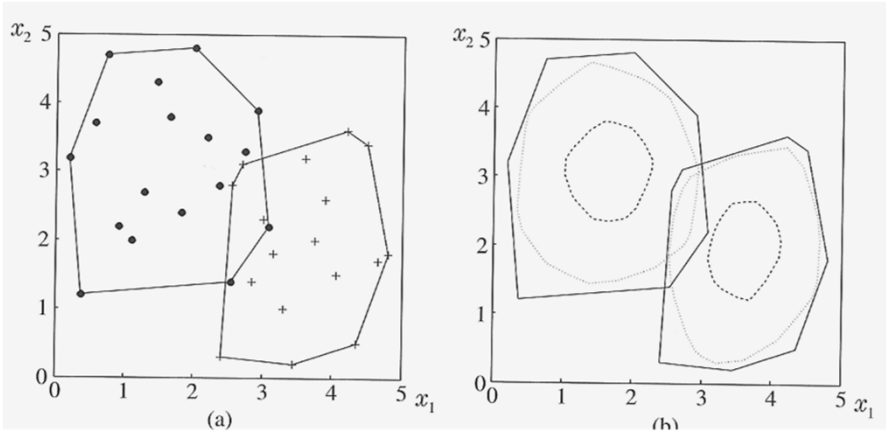
$$R\{X, \mu\} = \{y : \; y = \sum_{i=1}^{N} \lambda_i x_i : x_i \in X, \; \sum_{i=1}^{N} \lambda_i = 1, \;\; 0 \le \lambda_i \le \mu \;\}$$

Here we add a restriction that λ_i must also be smaller than μ

- μ is a scalar between 0 and 1. μ = 1 gives the regular convex hull.

Reduced convex hull - example

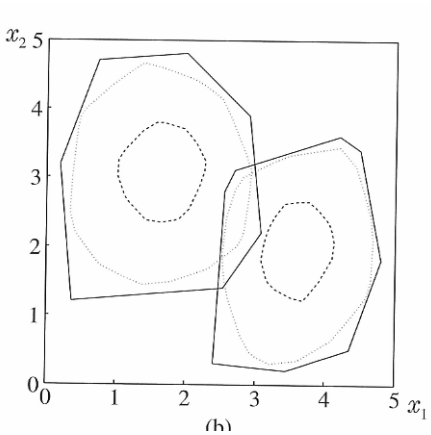
μ=1
Regular
convex
hulls

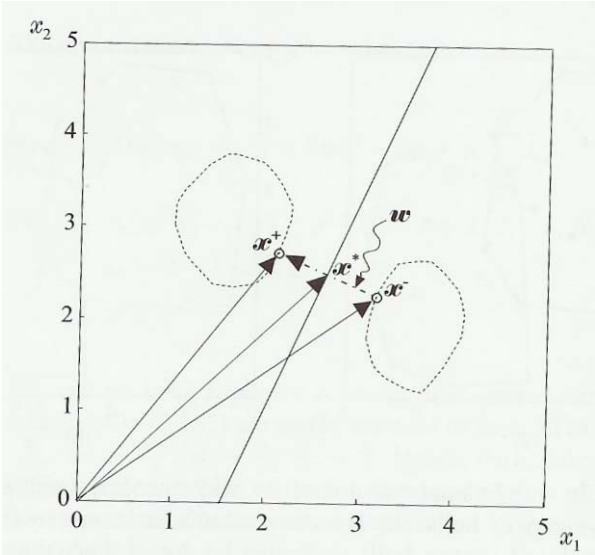


.....: μ=0.4
----: μ=0.1
Reduced
convex
hulls

- Data set with overlapping classes.
- For small enough values of μ, we can make the two reduced convex hulls non-overlapping.
- A very rough explanation of the non-separable SVM problem is that a value of μ that gives non-intersecting reduced convex hulls must be found.
- Given a value of μ that gives non-intersecting reduced convex hulls, the best hyperplane will bisect the line between the closest points in these two reduced convex hulls.

Relating μ and C





- Given a value of μ that gives non-intersecting reduced convex hulls, find the hyperplane by finding the closest two points in the two sets.
- Several values of μ can give nonintersecting reduced hulls.
- μ is related to C, the cost of misclassifying training regions (see page 101).
- A high C will give regions that just barely give nonintersecting regions.
- The most robust considering a validation data set is probably a smaller value of C (and μ).

Checkpoint

- What does this criterion mean:

$$J(w, w_0, \xi) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} I(\xi_i)$$

where $\quad \mathrm{I}(\xi_i) = \begin{cases} 1 & \xi_\mathrm{i} > 0 \\ 0 & \xi_\mathrm{i} = 0 \end{cases}$

and ξ is the vector of parameters ξ_i.

- Which points are the support vectors in the linear case?

QUESTIONS
ANSWERS

SVMs: The nonlinear case intro.

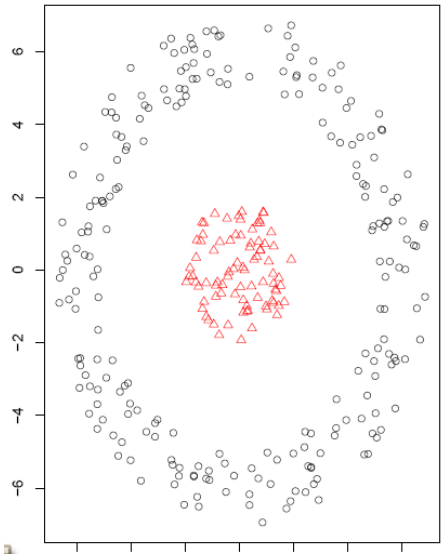
- The training samples are l-dimensional vectors; we have until now tried to find a linear separation in this l-dimensional feature space
- This seems quite limiting
- What if we increase the dimensionality (map our samples to a higher dimensional space) before applying our SVM?
- Perhaps we can find a better linear decision boundary in that space? Even if the feature vectors are not linearly separable in the input space, they might be (close to) separable in a higher dimensional space

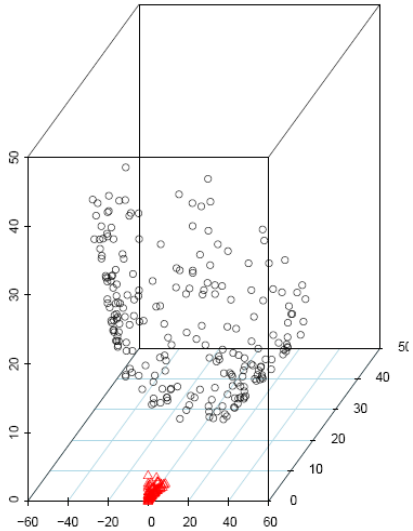
An examle: from 2D to 3D

- Let **x** be a 2D vector **x**=[x_1,x_2].
- In the toy example on the right, the two classes can <u>not</u> be linearly separated in the original 2D space.
- Consider now the transformation

$$y = \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}$$

- Now, the transformed points in this 3D space <u>can</u> be separated by a plane.
- The separating plane in 3D maps out an ellipse in the original 2D space





Cf. next slide, note that $y_i^T y_j = (x_i^T x_j)^2$.

Introducing the kernel trick

- A kernel can take the feature vectors into a higher dimensional space where they are linearly separable.
- Consider a simple example with two points:
  - $\mathbf{x} = (x_1,x_2,x_2)^T$ and $\mathbf{y}=(y_1,y_2,y_3)^T$
- If we want to add interactions between the points, we could transform the data into 9-dimensional space:
  - $\phi(\mathbf{x}) = (x_1^2,x_1x_2,x_1x_3,x_2x_1,x_2^2,x_2x_3,x_3x_1,x_3x_2,x_3^2)^T$
  - $\phi(\mathbf{y}) = (y_1^2,y_1y_2,y_1y_3,y_2y_1,y_2^2,y_2y_3,y_3y_1,y_3y_2,y_3^2)^T$
- The inner product in 9-dimensional space then gives us the interactions:
- $\varphi(x)^T\varphi(y) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$
- If we use a kernel function, we can calculate this i 3-dimensional space using kernels
- $k(x,y) = (x^T y)^2 = \sum_{i,j=1}^{3} x_i x_j y_i y_j$

SVMs and kernels

- Note that in both the optimization problem and the evaluation function, g(**x**), the samples come into play as inner products only

$$\max_{\lambda} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j \right)$$
$$\text{s.t.} \quad 0 \le \lambda_i \le C \quad \forall i$$
$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

$$g(x) = w^T x + w_0$$
$$= \sum_{i=1}^{N_s} \lambda_i y_i x^T x_i + w_0$$

Called «kernel»

- If we have a function evaluating inner products, K($\mathbf{x}_i$,$\mathbf{x}_j$), we can ignore the samples themselves
- Let's say we have K($\mathbf{x}_i$,$\mathbf{x}_j$) evaluating inner products in a higher dimensional space:

  -> no need to do the mapping of our samples explicitly!

Useful kernels for classification

- Polynomial kernels

$$K(x,z) = \left(x^T z - 1\right)^q, \quad q > 0$$

- Inner product measures similarity between x and z.
- If x has dimension d, the kernel takes the data into dxd-dimensions

Useful kernels for classification

- Radial basis function kernels (very commonly used!)

The inner product is related to the similarity of the two samples.

$$K(x,z) = \exp\left(-\frac{\|x-z\|^2}{\sigma^2}\right)$$

Note the we need to set the σ parameter

The «support» of each point is controlled by σ.

- This measures the similarity between x and z, scaled between 0 and 1
- σ determines the size of the similarity regions
- If we rewrite the exponential term, we see that this expresses the inner product in an infinite number of dimensions

$$\exp\left(-\frac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2\right) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right)$$

$$= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2}\|\mathbf{x}'\|^2\right) \frac{x'^{n_1}_1 \cdots x'^{n_k}_k}{\sqrt{n_1! \cdots n_k!}}$$

Useful kernels for classification

- Hyperbolic tangent kernels (often with β=2 and γ=1)

$$K(x,z) = \tanh\left(\beta x^T z + \gamma\right)$$

The kernel formulation of the objective function

- Given the appropriate kernel (e.g. «radial» with width σ) and the cost of misclassification C, the optimization task is:

$$\max_{\lambda}\left(\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j K(x_i, x_j)\right)$$

$$\text{subject to} \quad 0 \le \lambda_i \le C, \quad i = 1,....N$$

$$\sum_i \lambda_i y_i = 0$$

- The resulting classifier is:

$$\text{assign x to class } \omega_1 \text{ if } g(x) = \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + w_0 > 0 \text{ and to class } \omega_2 \text{ otherwise}$$

Example of nonlinear decision boundary

- This illustrates how the nonlinear SVM might look in the original feature space
- RBF kernel used

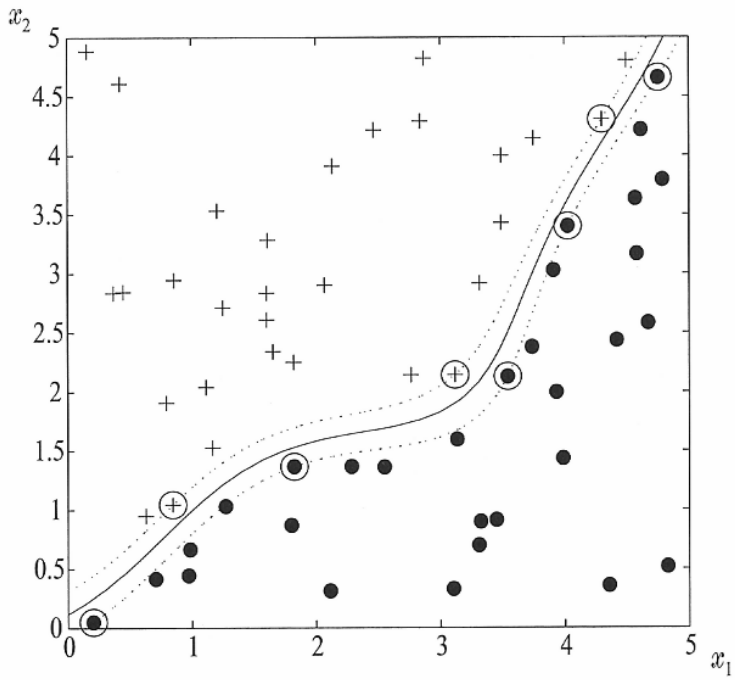


Figure 4.23 in
PR by Teodoridis et.al.

From 2 to M classes

- All we have discussed up until now involves only separating 2 classes. How do we extend the methods to M classes?
- Two common approaches:
  - One-against-all
    - For each class m, find the hyperplane that best disciminates this class from all other classes. Then classify a sample to the class having the highest output. (To use this, we need the VALUE of the inner product and not just the sign.)
  - **Compare all sets of pairwise classifiers**
    - **Find a hyperplane for each pair of classes. This gives M(M-1)/2 pairwise classifiers. For a given sample, use a voting scheme for selecting the most-winning class.**

How to use a SVM classifier

- Find a library with all the necessary SVM-functions ☺
  - Matlab: for example fitcsvm
  - Python: scikit-learn
- Read the introductory guides.
- Often a radial basis function kernel is a good starting point.
- Scale the data to the range [-1,1] (features with large values will not dominate).
- **Find the optimal values of C and σ by performing a grid search on selected values and using a validation data set.**
- Retrain the classifier using the best value from the grid search.
- Test using a separate test set.

How to do a grid search

- Grid search: try pairs of (C,σ). Compute the accuracy on the validation data set.
- Select the pair that gets the best classification performance on the validation data set.
- Use the following values of C and σ:
  - $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$
  - $\sigma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$

Summary / Learning goals

- Understand enough of SVM classifiers to be able to use it for a classification application.
  - Understand the basic linear separable problem and what the meaning of the solution with the largest margin is.
    - Geometrically and by using formulas
  - Understand how SVMs work in the non-separable case using a cost for misclassification.
  - Accept the kernel trick: that the original feature vectors can be transformed into a higher dimensional space, and that linear SVM is applied in this space without explicitly doing the feature transform
  - Know briefly how to extend from 2 to M classes.
  - Know which parameters (C, etc.) the user must specify and how to perform a grid search for these.
  - Be able to find a SVM library and use it correctly

QUESTIONS
ANSWERS

INF 5520 – Digital Image Analysis

MORPHOLOGICAL IMAGE PROCESSING

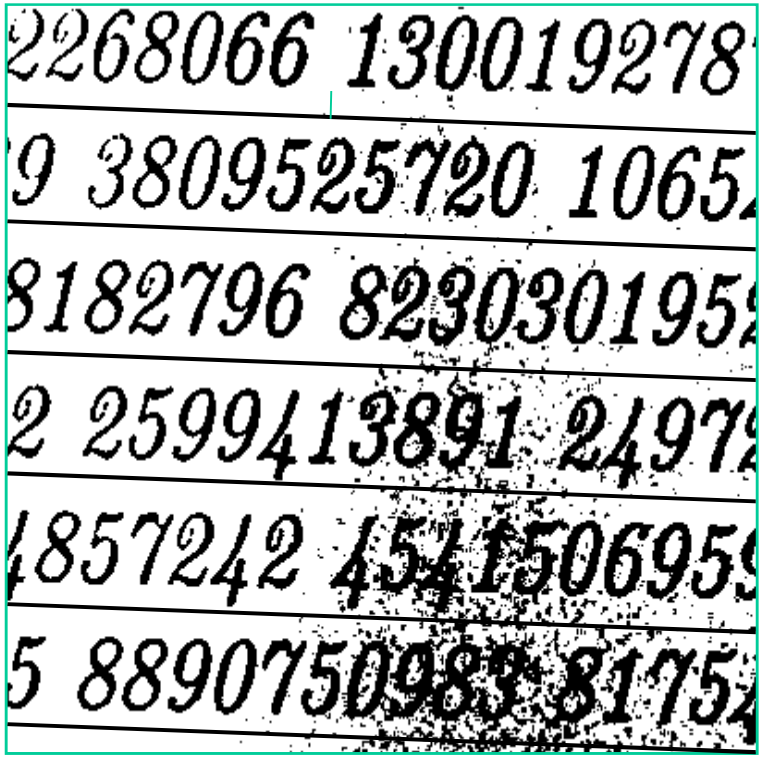
Fritz Albregtsen 04.11.2020

Today

- Gonzalez and Woods, Chapter 9, except sections on Skeletons (9.5.7 in 3Ed), Pruning (9.5.8 in 3Ed), Reconstruction (9.5.9 in 3Ed, 9.6 in 4Ed), and Gray scale reconstruction (9.6.4 in 3Ed, pp 688 in 4Ed).
- Repetition:
  - binary erosion, dilatation, opening, closing
  - (alternative: relaxed classification)
- Binary region processing: «Hit or miss» transformation, connected components, convex hull, thinning/thickening.
- Grey-level morphology:
  - erosion, dilation, opening, closing,
  - smoothing, gradient, top-hat, bottom-hat, granulometry.

Example

- Text segmentation and recognition.
- Binary morphological operations are useful after segmentation in order to improve segmentation of the objects.



Some symbols have been fragmented.

Some symbols are connected with background noise.

Symbols can be connected with neighboring symbols.

Need to remove lines or frames.

Simple set theory – read yourself
2.6.4 in 3. Ed., pp 91 in 4.Ed.

- Let A be a set in $\mathbb{Z}^2$ (integers in 2D) .
  If the point a=(a_1,a_2) is an element in A we denote: $a \in A$
- If a is not an element in A we denote: $a \notin A$
- An empty set is denoted $\varnothing$.
- If all elements in A are also part of B,
  A is called a subset of B and denoted: $A \subseteq B$
- The union of two sets A and B consists of
  all elements in either A or B, and is denoted: $A \cup B$
- The intersection (="snitt") of A and B consists of all elements
  that are part of both A and B and is denoted: $A \cap B$
- The complement of a set A is the set of elements not in A:

$$A^C = \{w \mid w \notin A\}$$

- The difference of two sets A and B is:

$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^C$$

- In addition: Set reflection and translation (see Section 9.1).

Set theory on binary images

- The complement of a binary image

$$g(x,y)=\begin{cases}1 & \text{if } f(x,y)=0\\ 0 & \text{if } f(x,y)=1\end{cases}$$

- The intersection of two binary images f and g is

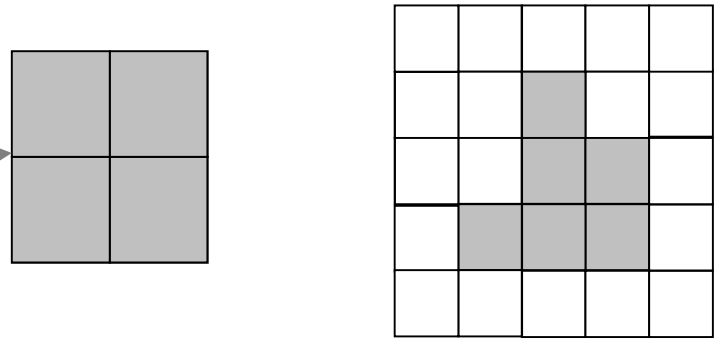
$$h=f\cap g=h(x,y)=\begin{cases}1 & \text{if } f(x,y)=1 \quad \text{and} \quad g(x,y)=1\\ 0 & \text{otherwise}\end{cases}$$

- The union of two binary images f and g is

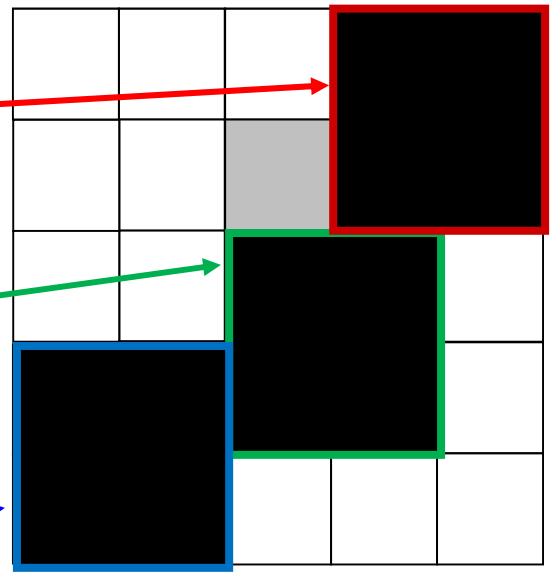
$$h=f\cup g=h(x,y)=\begin{cases}1 & \text{if } f(x,y)=1 \quad \text{or} \quad g(x,y)=1\\ 0 & \text{otherwise}\end{cases}$$

Repetition - Hit vs. Fit

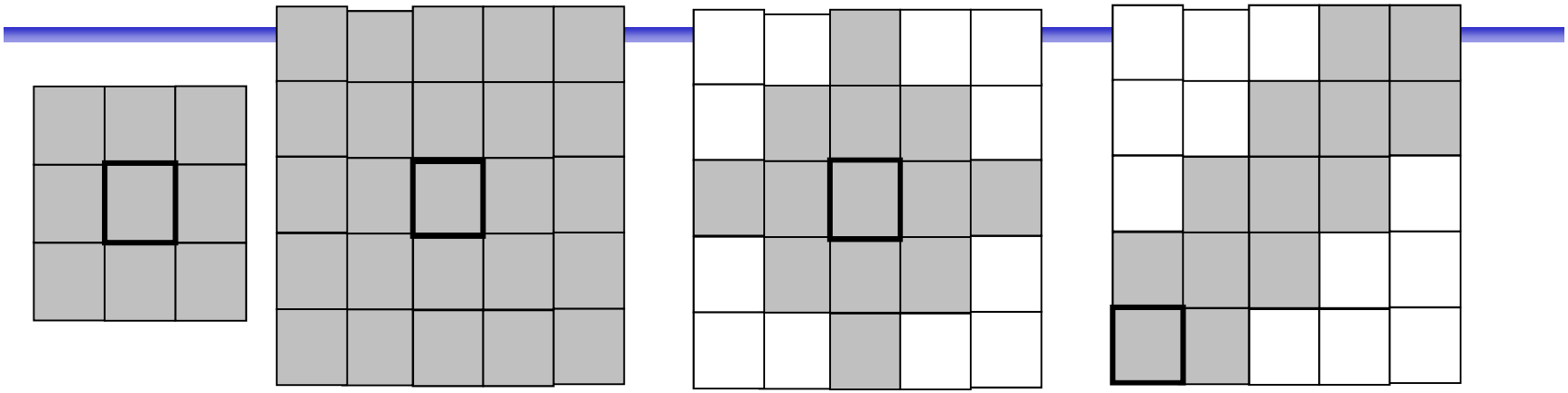
- A structuring element for a binary image is a small matrix of pixels
- Let the structuring element overlay the binary image containing an object at different pixel positions.



- The following cases arise:
  - Positions where the element does not overlap with the object.
  - Positions where the whole element fits inside the object
    - the element fits the object
  - Positions where the element partly overlaps the object
    - the element hits the object



Repetition – structuring elements



- structuring elements can have different sizes and shapes

- A structuring element has an origin/center
  - The origin is a pixel position
  - The origin *can* be outside the element.
  - The origin is often marked on the structuring element using □
  - Otherwise, we assume the center pixel is the origin.
- The structuring element can be *flat* or *non-flat* (have different values)
  - We will here work with a flat structuring element

Repetition
Erosion of a <u>binary</u> image

- To compute the erosion of pixel (x,y) in image f with the structuring element S: place the structuring elements such that its origo is at (x,y). Compute

$$g(x,y)=\begin{cases}1 & \text{if S fits f}\\ 0 & \text{otherwise}\end{cases}$$

- Erosion of the image f with structuring element S is denoted

  ε (f|S) = f ⊖ S

- Erosion of a set A with the stucturing element B is defined as the position of all pixels x in A such that B is included in A when origo of B is at x.

$$A\,\theta\,B=\{x|B_x\subseteq A\}$$

0	1	0	0	0	0	0	0	1	1	0
1	1	1	0	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	0	1	1	1	0
0	1	1	1	1	0	0	0	1	1	1
0	0	1	1	0	0	0	0	0	1	0

eroded by

1	1	1
1	1	1
1	1	1

gives

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

0	1	0
1	1	1
0	1	0

gives

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	1	1	0	0	0
0	0	0	1	0	1	1	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0
0	0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

Edge detection using erosion

- Erosion removes pixels on the border of an object.
- We can find the border by subtracting an eroded image from the original image: $g = f - (f \ominus s)$
- The structuring element decides if the detected edge pixels will be 4-neighbors or 8-neighbors

0	0	1	1	1	1	0	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
1	1	1	1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	0	0	0	0	1	1	1	0	0	0

Eroded by

0	1	0
1	1	1
0	1	0

gives

0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	1	1	0	0
0	0	1	1	0	1	1	1	1	0	0
0	1	1	0	0	0	1	1	1	1	0
0	0	1	1	0	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0
0	0	1	0	0	0	1	1	1	0	0
0	0	1	0	0	0	1	1	1	0	0
0	0	1	0	0	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

difference => edge pixels

0	0	1	1	1	1	0	1	1	1	0
0	1	0	0	0	0	1	0	0	1	0
0	1	0	0	1	0	0	0	0	1	0
1	0	0	1	0	1	0	0	0	0	1
0	1	0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0
0	1	1	1	1	0	0	0	1	1	0
0	0	0	0	0	1	1	1	0	0	0

0	0	1	1	1	1	0	1	1	1	0
0	1	1	0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0	0	1	0
1	1	0	1	0	1	0	0	0	1	1
0	1	0	1	1	1	0	0	0	1	1
0	1	0	0	0	0	0	0	0	1	0
0	1	1	1	1	1	0	1	1	1	0
0	0	0	0	0	1	1	1	0	0	0

8-edgels – 4-edgels

0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Edge detection

$$f - (f \theta S)$$

Example use: find border pixels in a region

Dilation of a binary image

- Place S such that origo lies in pixel (x,y) and use the rule

$$g(x,y) = \begin{cases} 1 & \text{if S hits f} \\ 0 & \text{otherwise} \end{cases}$$

- The image f dilated by the structuring element S is denoted:

$$f \oplus S$$

- Dilation of a set A with a structuring element B is defined as the position of all pixels x such that B overlaps with at least one pixel in A when the origin is placed at x.

$$A \oplus B = \{x \mid B_x \cap A \neq \emptyset\}$$

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	1	1	0	0	0
0	0	0	1	0	1	1	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0
0	0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

Dilated by

1	1	1
1	$\boxed{1}$	1
1	1	1

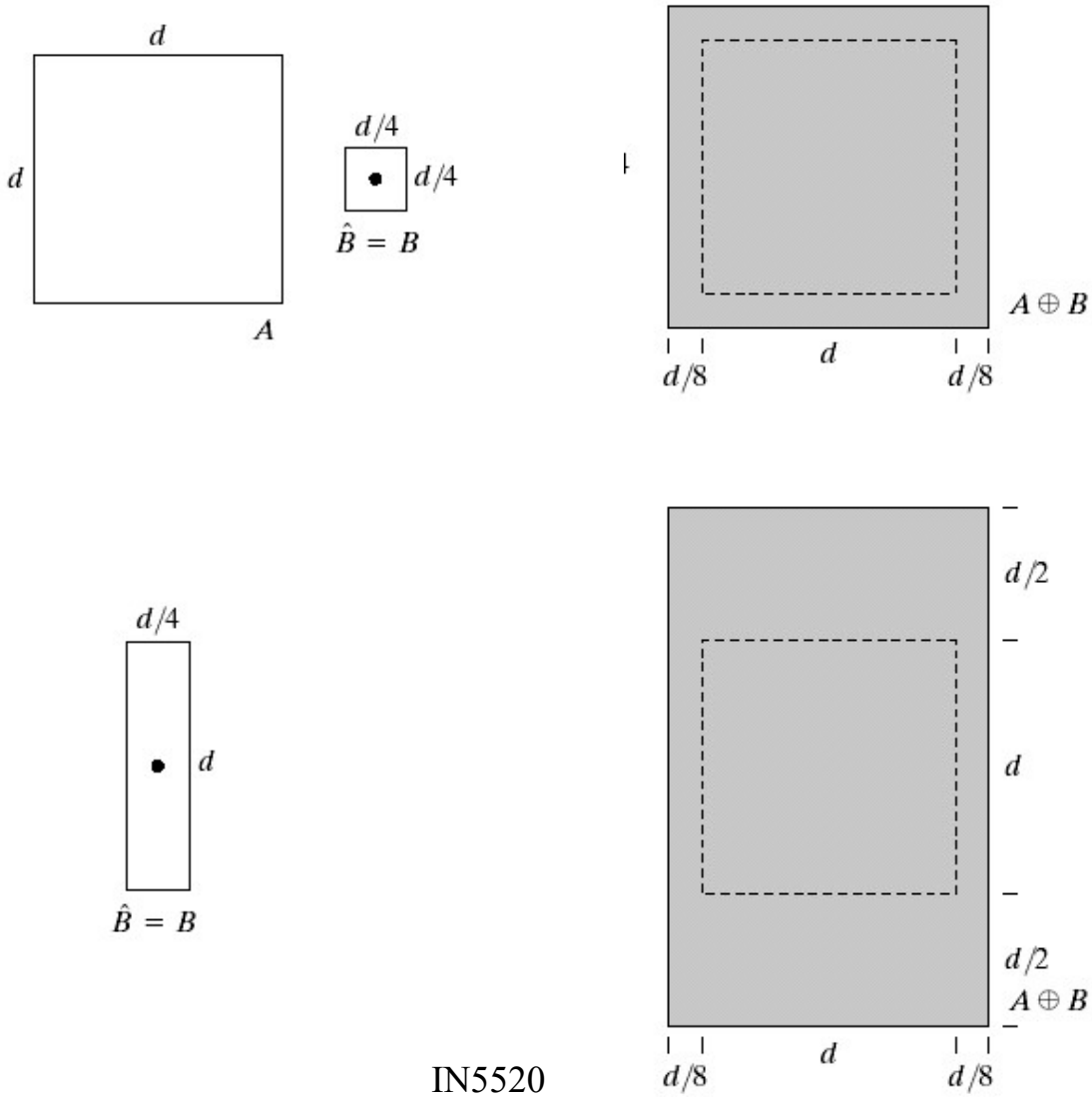
0	1	0
1	$\boxed{1}$	1
0	1	0

gives

1	1	1	0	0	0	1	1	1	1	0
1	1	1	1	0	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	0	0	0	1	1	1

0	1	0	0	0	0	0	1	1	0	0
1	1	1	0	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	0	1	1	1	0
0	1	1	1	1	0	0	0	1	1	1
0	0	1	1	0	0	0	0	0	1	0

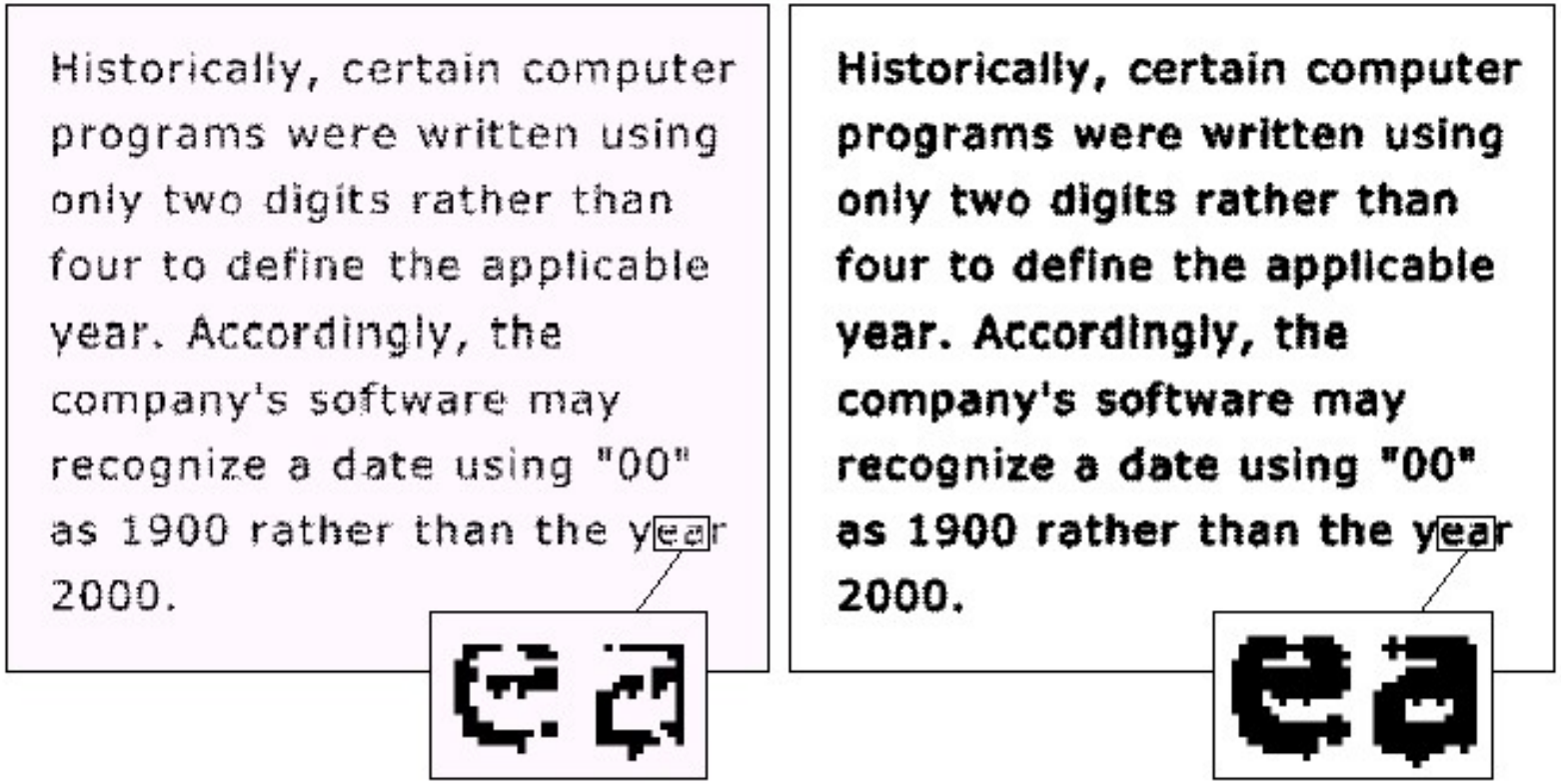
Dilation



Effects of dilation

- Expand the object borders
  - Both inside and outside borders of the object
- Dilation fills holes in the object
- Dilation smooths out the object contour
- Depends on the structuring element
- Bigger structuring element gives greater effect

Example of use of dilation – fill gaps



0	1	0
1	1	1
0	1	0

Combined operation I: Opening

- Erosion of an image removes all structures that the structuring element cannot fit inside, and shrinks all other structures.

- Dilating the result of the erosion with the same structuring element, the structures that survived the erosion (were shrunken, not deleted) will be restored.

- This is called morphological opening:

$$f \circ S = \left(f \, \theta \, \mathrm{S} \right) \oplus S$$

- The name tells that the operation can create an opening between two structures that are connected only by a thin bridge, without shrinking the structures (as erosion would do).

Opening

- The operation can create an opening between two structures that are connected only in a thin bridge, without just shrinking the structures (as erosion alone would do).

$$f \circ S = (f \,\theta\, \mathrm{S}) \oplus S$$

original

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	1	0
0	1	1	0	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	1	0
0	1	1	1	1	1	0	1	1	1	0
0	1	1	1	1	0	0	0	1	1	0
0	1	1	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

erosion

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

dilated erosion

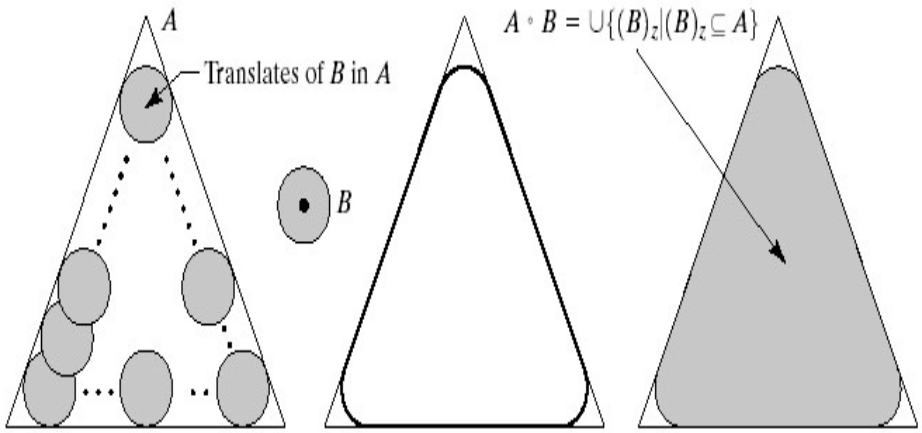
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	0	1	1	1	0
0	1	1	1	0	0	0	1	1	1	0
0	1	1	1	0	0	0	1	1	1	0
0	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

difference

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

Visualizing opening

- Imagine that the structuring element traverses the edge of the object.
  - First on the inside of the object. The object shrinks.
  - Then the structuring element traverses the outside of the resulting object from the previous passage.
  - The object grows, but small branches removed in the last step will not be restored.



Combined operation II: Closing

- A dilation of an object grows the object and can fill gaps.
- If we erode the result with the rotated structuring element, the objects will keep their structure and form, but small holes filled by dilation will not appear.
- Objects merged by the dilation will not be separated again.
- Closing is defined as $$f \bullet S = \left(f \oplus \hat{S}\right) \theta \, \hat{S}$$
- This operation can close gaps between two structures without growing the size of the structures like dilation would.

Closing

- This operation can close gaps between two structures without growing the size of the structures like dilation would.

$$f \bullet S = \left(f \oplus \hat{S}\right) \theta \, \hat{S}$$

original

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	0
0	1	1	0	0	1	1	1	1	1	0
0	1	1	1	0	0	1	1	1	1	0
0	1	1	1	1	0	0	1	1	1	0
0	1	1	1	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

0	1	0
1	1	1
0	1	0

dilation

0	1	0	0	0	0	1	1	1	1	0
1	1	1	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0	0	1	0

eroded dilation = closing

0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	0
0	1	1	0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	0	0	1	1	0
0	1	1	1	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0

Difference from original

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

- Very different result from a simple dilation (e.g., slide 14)

Alternative: Relaxation – the motivation

- If an image contains dark/light objects on a light/dark background, the objects may be extracted by thresholding.
- If distributions overlap, there will be errors in the segmentation.
- A relaxation process can be used to reduce these errors;
  - First classify all the pixels probabilistically,
  - then adjust the probabilities for each pixel,
    based on its neighbors‘ probabilities,
    with light reinforcing light and dark reinforcing dark.

Q: Intuitive practical applications of this (e.g., in remote sensing)?

- When this adjustment process is iterated, the dark probabilities become very high for pixels that belong to dark regions, and vice versa, so that thresholding becomes (more) trivial.

Relaxation – a possible method

- Several relaxation methods exist, e.g., Rosenfeld et al. (1976):

  $$p_{ij}^{(r+1)} = p_{ij}^{(r)} (1+ q_{ij}^{(r)}) / sum_{j=1}^{m} [p_{ij}^{(r)} (1 + q_{ij}^{(r)})]$$

  where $q_{ij}^{(r)}$ is the average, over all neighbors (h) of the ith pixel, of the $sum_{k=1}^{m} [c(i,j;h,k)] \, p_{hk}^{(r)}$ over all m classes.
- Here $p_{ij}^{(r)}$ is the estimate, at the r*th* iteration, that the i*th* pixel $(1 \le i \le n)$, belongs to the j*th* class, $(1 \le j \le m)$; in thresholding $m = 2$.
- The "compatibility coefficient" $c(i, j; h, k)$ gives the compatibility of the pair of events *(pixel i in class j; pixel h in class k).*
- E.g., set $c = 0$ for non-neighboring pairs of pixels, while for neighboring pixel pairs, c is the mutual information:

  c = log [prob(object i in class j, object h in class k]/
  [prob (object i in class j) prob(object h in class k].

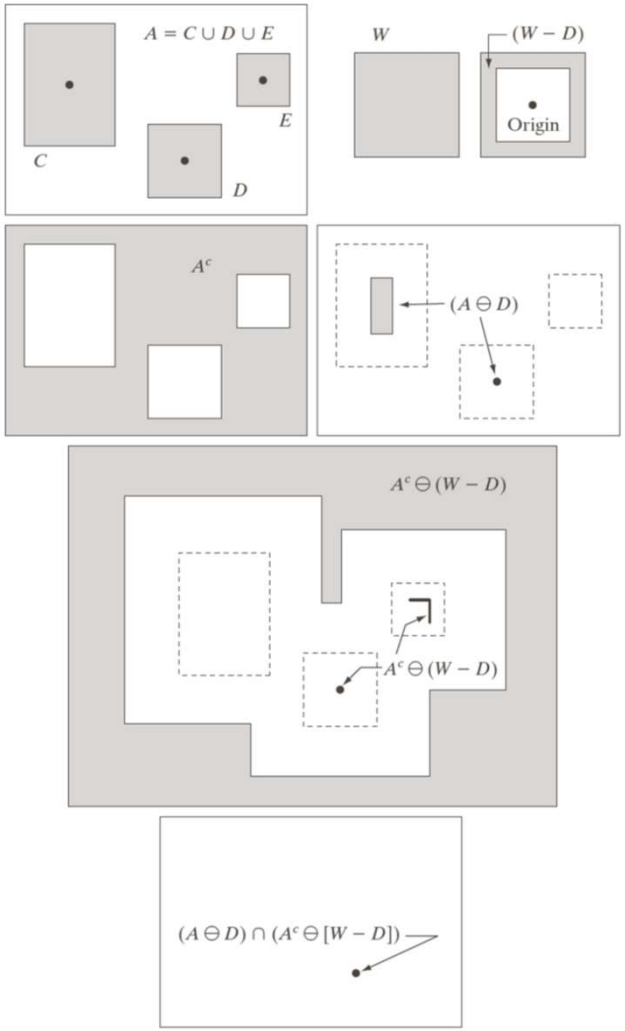
  Q: How would you obtain c?

“Hit or miss”- transformation

- Transformation used to detect a given pattern in the image – “template matching”.
- Objective: Find location of shape D in set A.
- D can fit inside many objects, so we need to look at the local background W-D.
- First, compute the erosion of A by D, AθD (all pixels where D can fit inside A)
- To fit also the background: Compute A^C, the complement of A. The set of locations where D exactly fits is the intersection of AθD and the erosion of A^C by W-D, A^C θ(W-D).
- Hit-or-miss (HMT) is expressed as A ⊛ D:

$$(A\theta D)\cap\left[A_C\theta(W-D)\right]$$

Main use: Detection of a given pattern or removal of single pixels

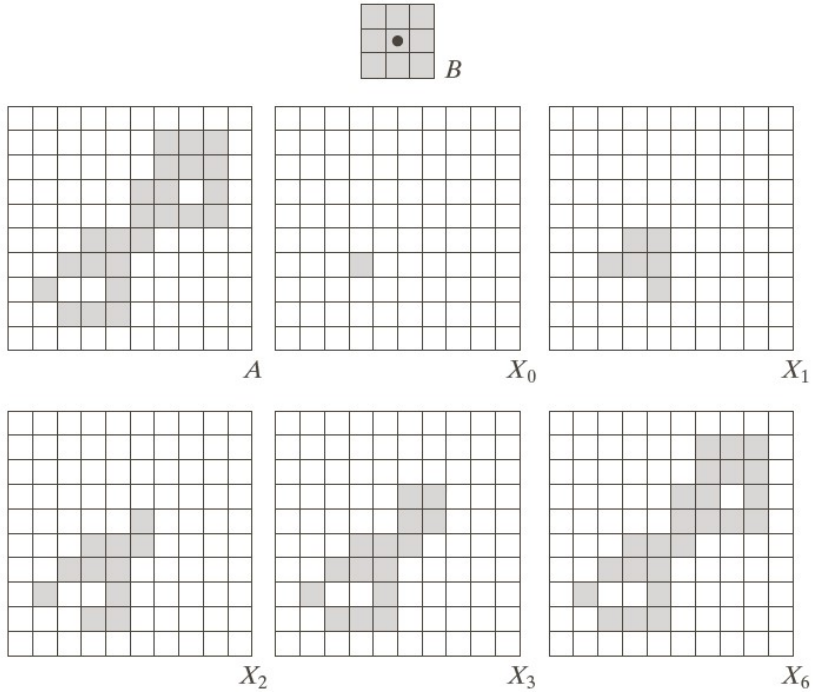


Extracting connected components

- Detects a connected object Y, in image A, given a point p in Y
  - Start with X_0, a point in Y
  - Dilate X_0 with either a square or plus
  - Let X_1 be only those pixels in the dilation that are part of the original region.
  - Continue dilating X_1 to give X_k until $X_k = X_{k-1}$

$$X_k = X_{k-1} \oplus B \cap A$$

$$X_0 = p, \; k=1,2,3,$$



Computing convex hull using morphology

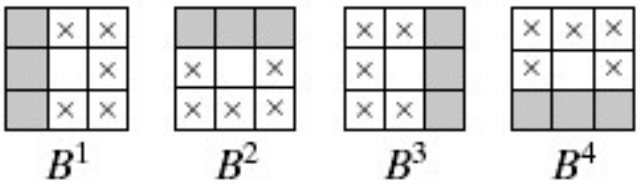
- The convex hull C of a set of points A may be estimated using the Hit-or-Miss transformation
- Consider the four structuring elements B^1-B^4.
- Apply hit-or-miss with A using B^1 iteratively until no more changes occur. Let D^1 be the result.
- Then do the same with B^2,..,B^4 to compute D^2…D^4 in the same manner.
- Then compute the convex hull by the union of all the D^is.

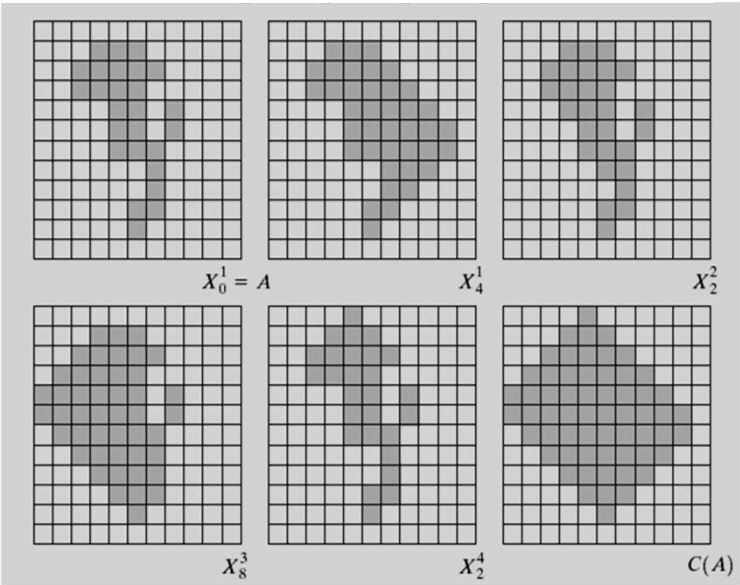
$$X_k^i = \left(X_{k-1}^i \circledast B^i\right) \cup A;\ i = 1, 2, 3, 4;$$
$$k = 1, 2, 3, \ldots;\ X_0^i = A;\ \text{and}$$
$$D^i = X_{\text{conv}}^i.$$

Then C (A) = $\bigcup D_i$

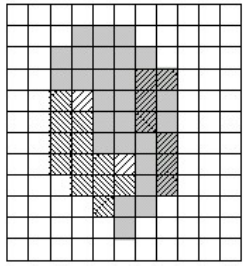
- Gives too big area to guarantee convexity
- Can be corrected by taking the intersection to the maximum dimension in x and y direction

C_res = C (A) ∩ ROI(A)



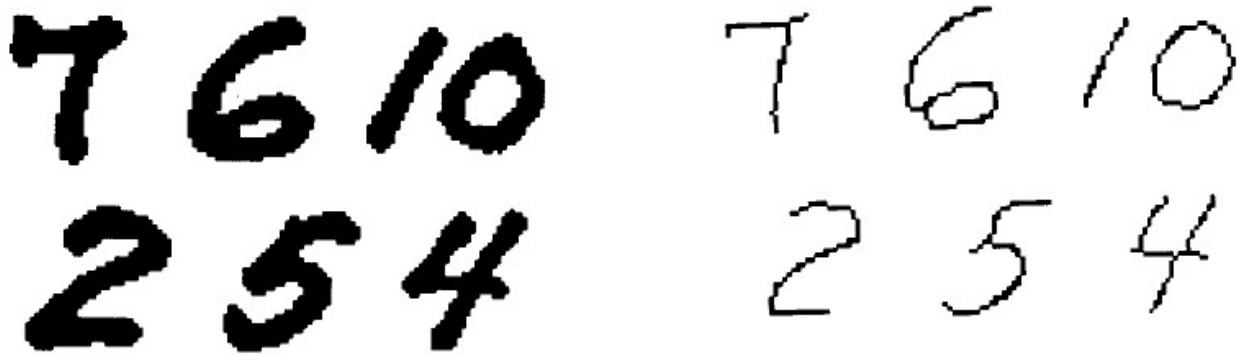


C_res



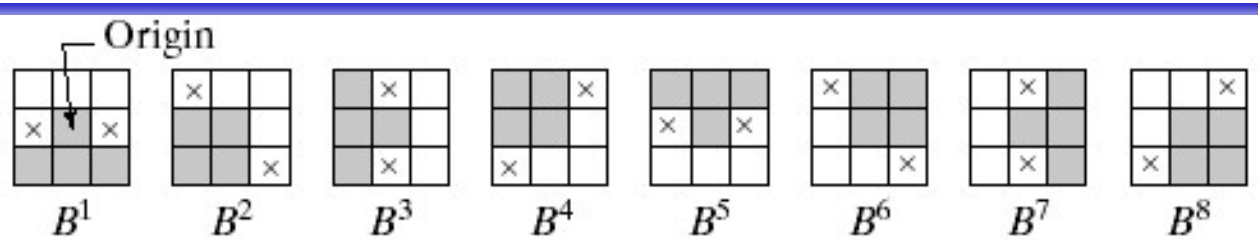
x in B means don't care

Region thinning and skeletons

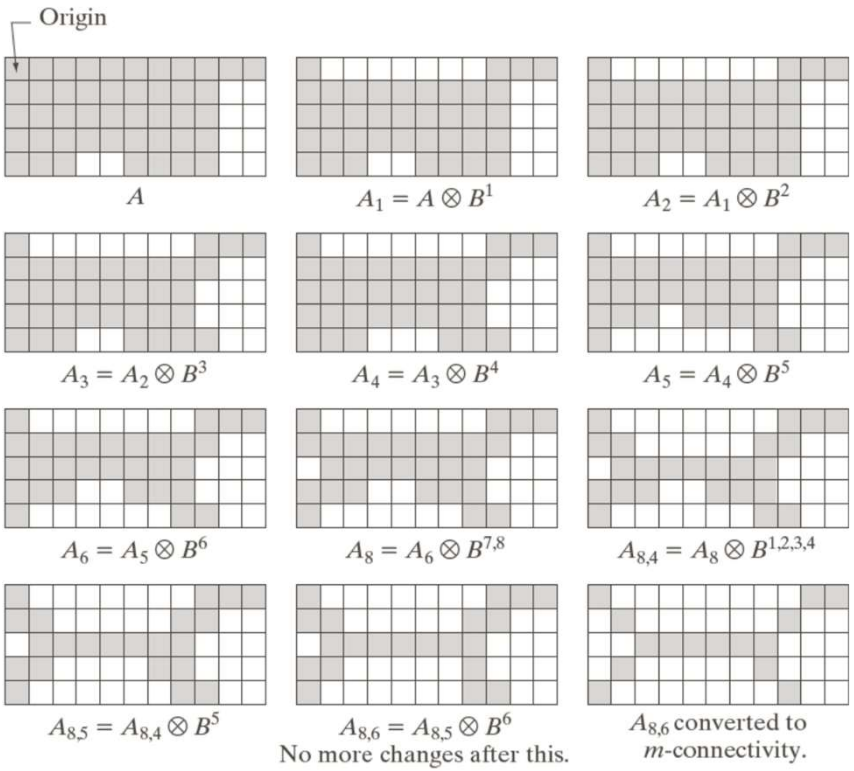


- Let the region object be described using an intrinsic coordinate system, where every pixel is described by its distance from the nearest boundary pixel.
- The skeleton is defined as the set of pixels whose distance from the nearest boundary is locally maximum.
- Many different methods for computing the skeleton exist.
- Shape features can later be extracted from the skeleton.
- Skeletonization may imply loss of information.
- *Thinning* is a possible procedure to compute the skeleton.

Thinning



- Thinning of set A with structuring element B (a set of n structuring elements, here n=8)
- First, thin A by one pass of B^1, then thin the result by one pass of B^2, and so on, until A is thinned with one pass of B^n.
- Then repeat the entire process until no further changes occur.



- $A \otimes B = A - (A \circledast B)$
- $A \otimes \{B\} =$
  $((\ldots((A \otimes B^1) \otimes B^2)\ldots) \otimes B^n)$

Thickening

- Thickening is the dual operator of thinning.

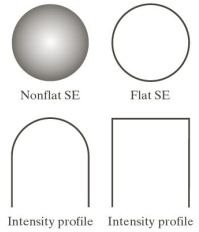
- It can be computed as a separate operation, but thickening the object is normally computed by thinning the background and then complementing the result:

  from $C=A^C$, thin C, then form C^C.

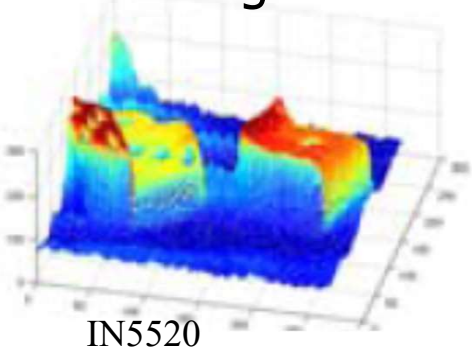
  (Example: Fig 9.22 in G&W)

Gray level morphology

- We apply a simplified definition of morphological operations on gray level images
  - Grey-level erosion, dilation, opening, closing
- Image f(x,y)
- Structuring element b(x,y)
  - May be nonflat or flat



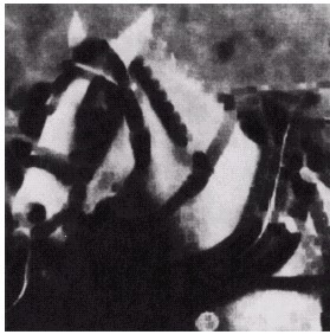
- Assume symmetric, flat structuring element, origo at center (this is sufficient for normal use).
- Erosion and dilation then correspond to local minimum and maximum over the area defined by the structuring element



Gray level erosion /dilation

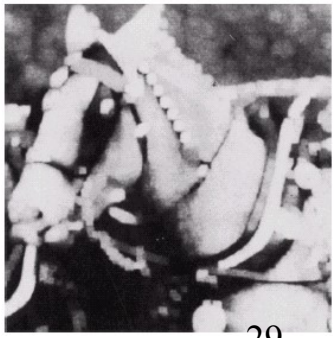
- **Erosion**:
  - Place the structuring element with origo at pixel (x,y)
  - Chose the local **minimum** gray level in the region defined by the structuring element B
  - Assign this value to the output pixel (x,y)
  - Results in darker images and removal of light details:

$$[f \,\theta\, b](x,y) = \min_{(s,t)\in B}\{f(x+s, y+t)\}$$

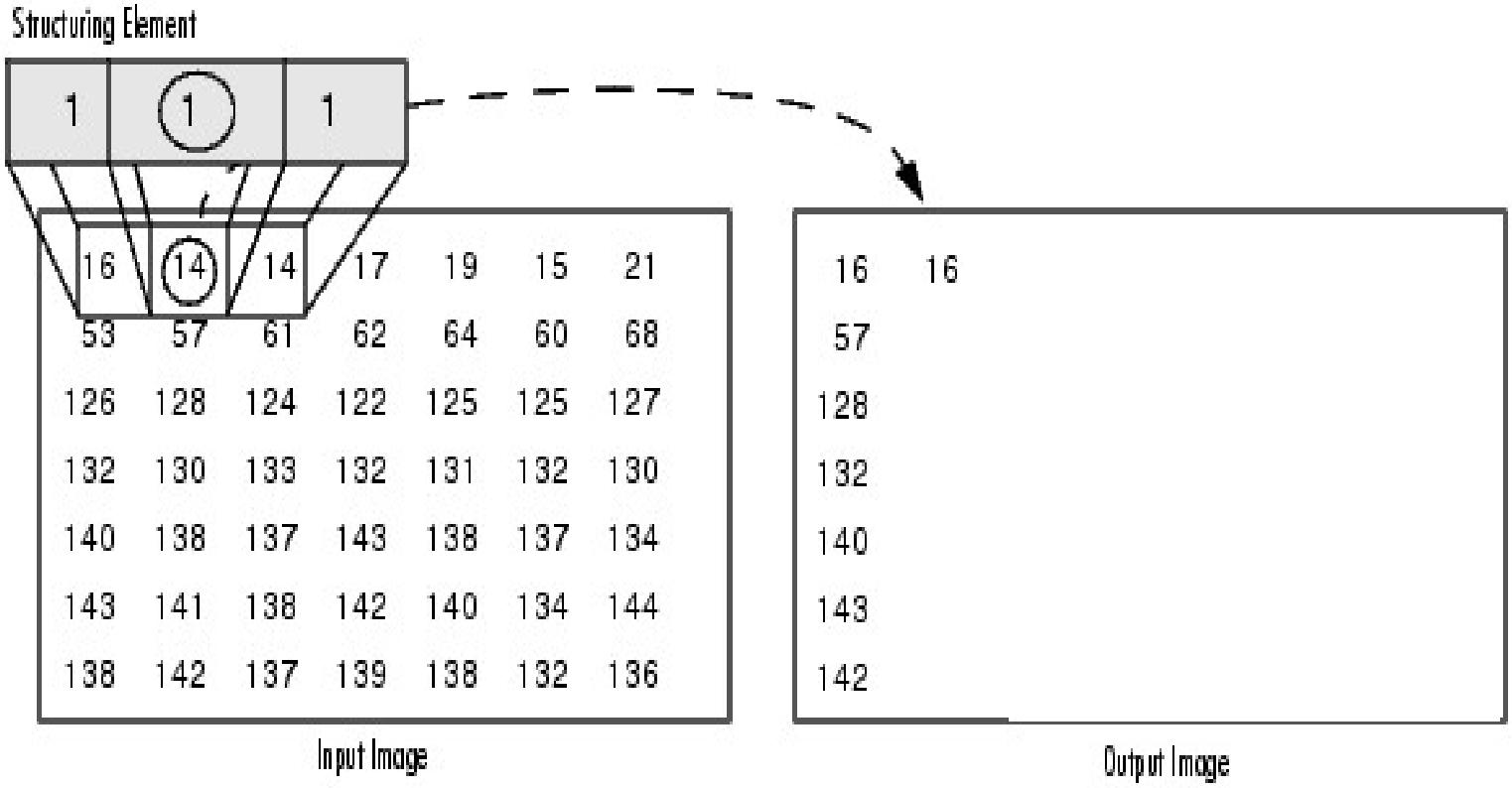


- **Dilation**:
  - Chose the local **maximum** gray level in the region defined by the (reflected) structuring element B
  - Let pixel (x,y) in the outimage have this value.
  - Gives brighter images where dark details are removed

$$[f \oplus b](x,y) = \max_{(s,t)\in B}\{f(x-s, y-t)\}$$



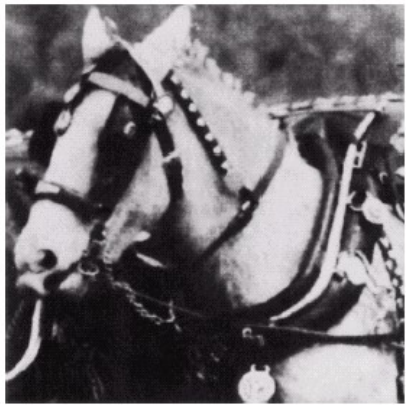
Gray level morphology- some details



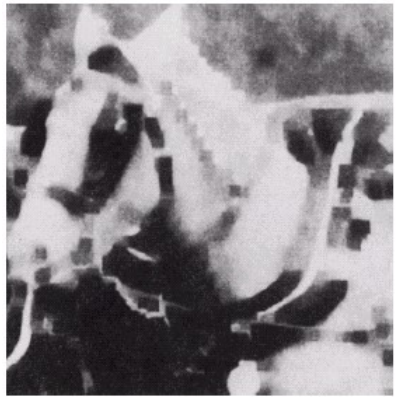
Morphological Dilation of a Grayscale Image

Gray level opening and closing

- Corresponding definition as for binary opening and closing
- Result in a filter effect on the intensity
- Opening: Bright details are smoothed
- Closing: Dark details are smoothed



$$f \circ S = (f \,\theta\, \mathrm{S}) \oplus S$$
$$= \max(\min(f))$$



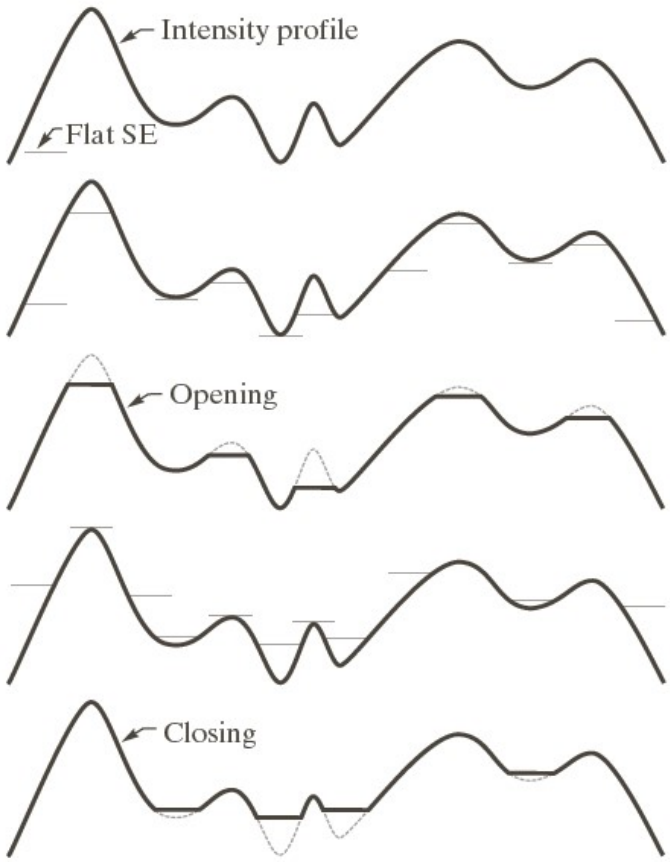
$$f \bullet S = (f \oplus \mathrm{S})\, \theta\, S$$
$$= \min(\max(f))$$

Interpretation of grey-level opening and closing

- Intensity values are interpreted as height curves over the (x,y)-plane.

- Opening of f by b:
  Push the structuring element up from below towards the curve f.
  The value assigned is the highest level b can reach.
  smooths bright values down.

- Closing:
  Push the structuring element down from above towards the curve f.
  smooths dark values upwards



Morphological filtering

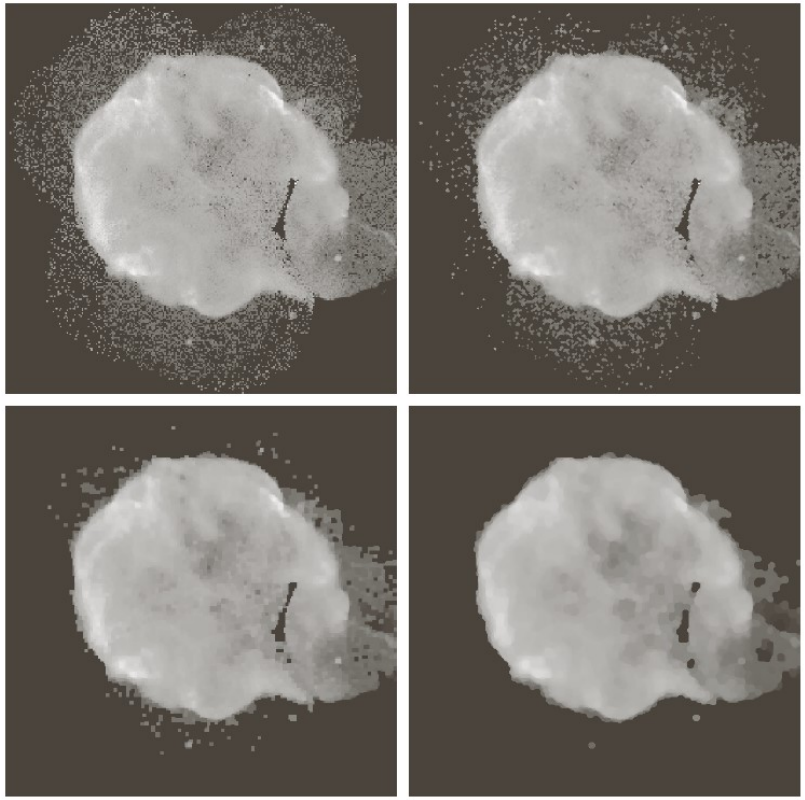
- Grey-level opening and closing with flat structure elements can be used to filter out noise.
- This is particularly useful for e.g. dark or bright noise.
- Remark: bright or dark is relative to surroundings, eg. local extremas are filtered.
- To remove bright noise, do first opening, then closing.

  $$\min(\max(\max(\min(f(x,y)))))$$

  This can be repeated with different SE's.
- The size of the structuring element should reflect the size of the noise objects that we wish to remove.
- NB! "2D Min and Max are separable!

Example – morphological filtering

Original



Opening and closing by disk of size 1

Opening and closing by disk of size 5

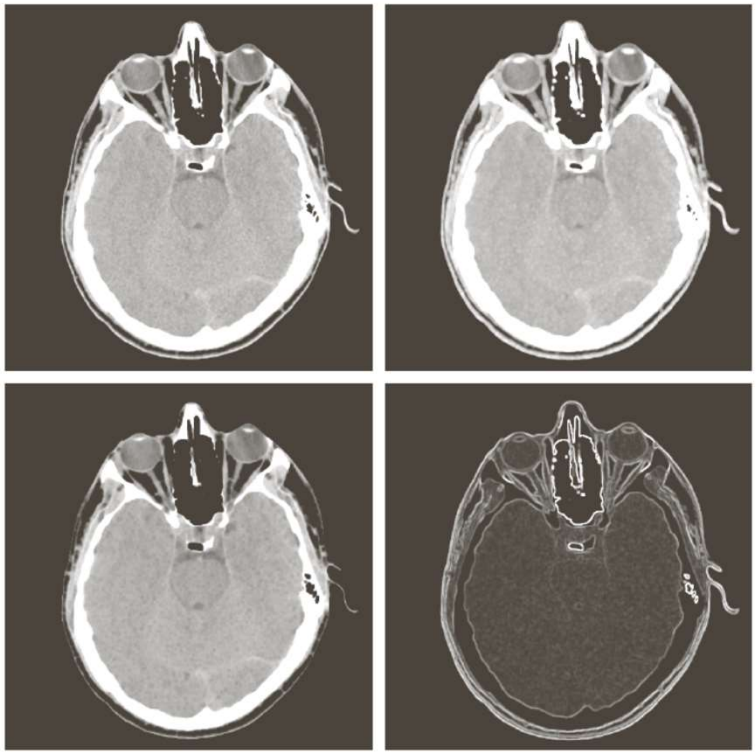
Opening and closing by disk of size 3

Morphological gradient

- Gray level dilation will (under some conditions) give an image with equal or brighter values – as it is a local max-operator.
- Erosion will under the same conditions produce an image with equal or lower values - as it is a local min-operator.
- This can be used for edge detection
- Morphological gradient = $(f \oplus S) - (f \ominus S)$

  => gradient = dilation minus erosion

Morphological gradient



Dilation

Gradient
= dilation-erosion

Erosion

Top-hat transformation

- Purpose: detect (or remove) structures of a certain size.
- Top-hat: detects light objects on a darker background
  - also called white top-hat.
- Top-hat (image minus its opening):

$$f - (f \circ b)$$

- Bottom-hat: detects dark objects on a brighter background
  - also called black top-hat.
- Bottom-hat (closing minus image):

$$(f \bullet b) - f$$

- Very useful when correcting for uneven illumination or objects on a varying background ☺ (see slide 32).

Example – top-hat

Original, un-even background

Global thresholding using Otsu's method.

Objects in the lower right corner disappear. Misclassification of some background in upper left corner.

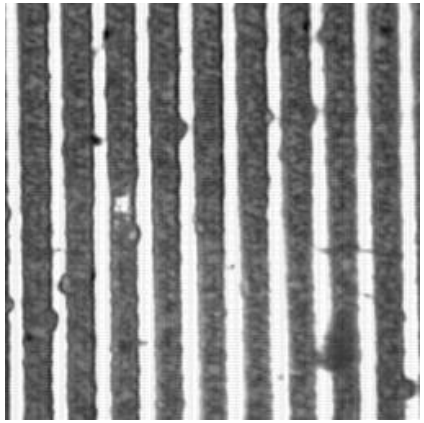
Opening with a 41x41 structuring element removes objects and gives an estimate of the background

An alternative to opening would be a moving 41x41 median filtering

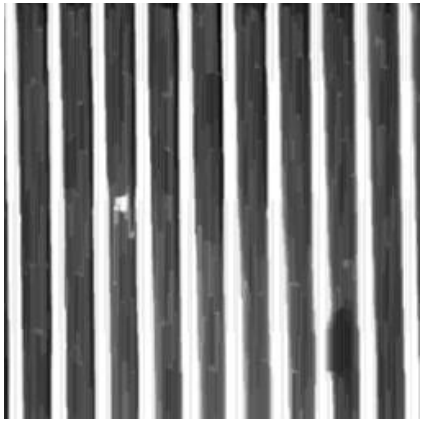
Top-hat transform (original – opening)

Top-hat thresholded with global Otsu threshold

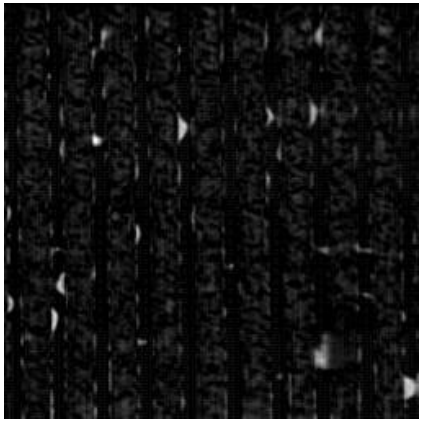
Fault detection using 'Bottom-hat'



Original

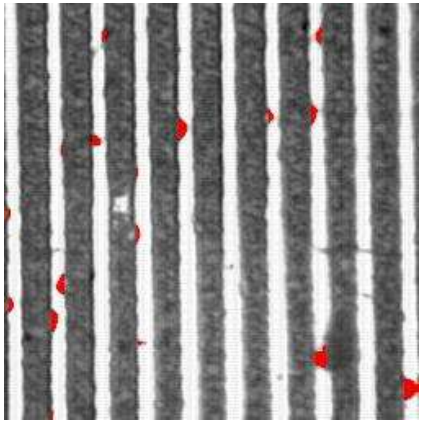


Closing' with vertical structuring element



Bottom-Hat= Original – closing

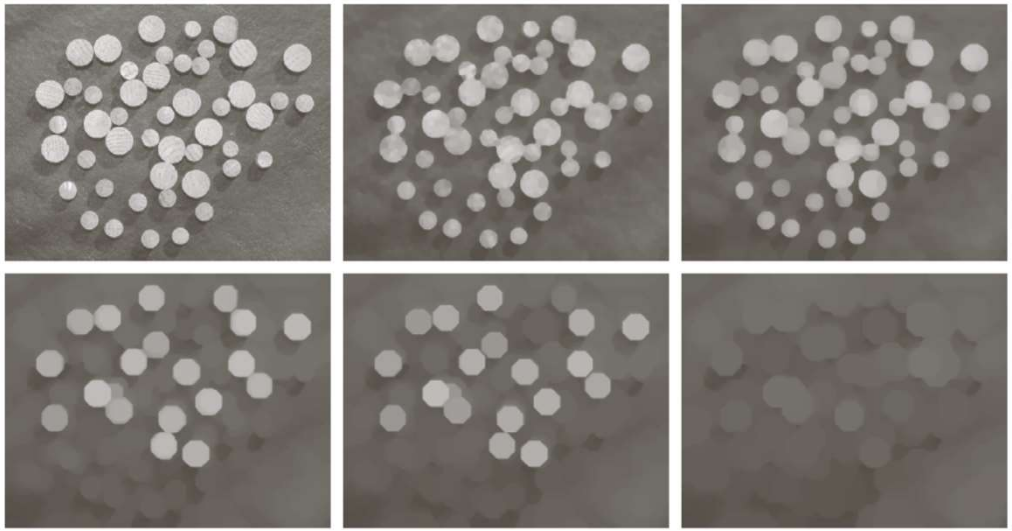
Threholded previous image



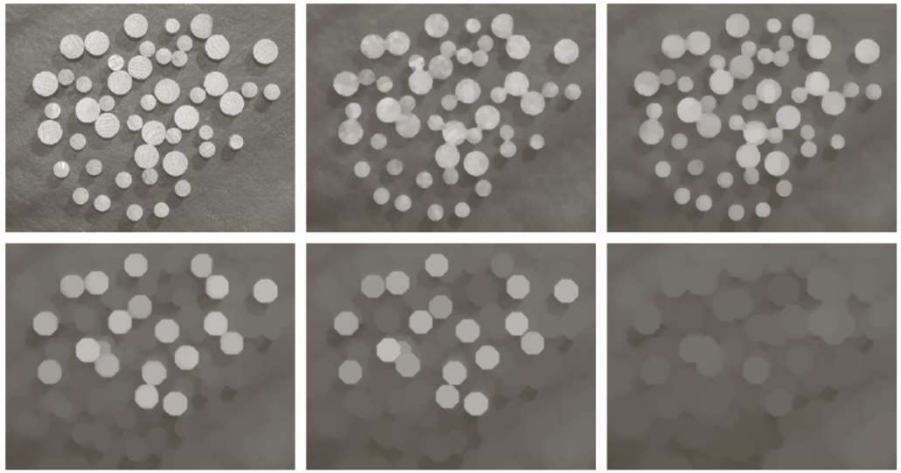
Error detection

Example application: granulometry

- Granulometry: determine the size distribution of particles in an image.
- Assumption: objects with regular shape on a background.
- Principle: perform a series of openings
  with increasing radius r of structuring element
- Compute the sum of all pixel values after the opening.
- Compute the difference in this sum between radius r and r-1,
  and plot this as a function of radius.



Example - granulometry



a b c
d e f

FIGURE 9.41 (a) 531 × 675 image of wood dowels. (b) Smoothed image. (c)–(f) Openings of (b) with disks of radii equal to 10, 20, 25, and 30 pixels, respectively. (Original image courtesy of Dr. Steve Eddins, The MathWorks, Inc.)

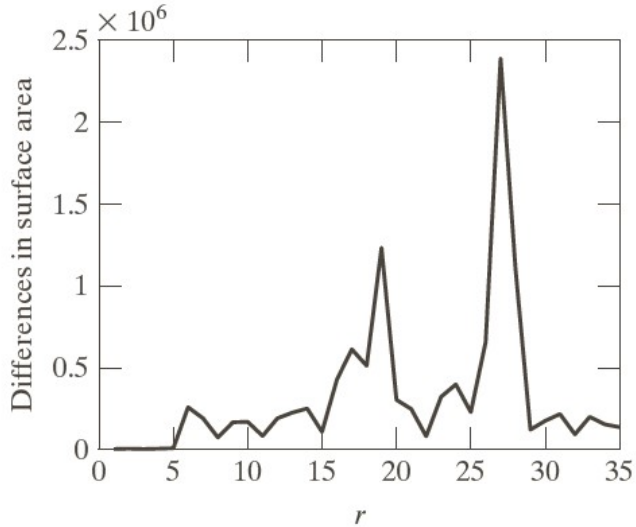


FIGURE 9.42 Differences in surface area as a function of SE disk radius, r. The two peaks are indicative of two dominant particle sizes in the image.

Learning goals - morphology

- Understand in detail binary morphological operations and selected applications:
  - Basic operators (erosion, dilation, opening, closing)
    - Understand the mathematical definition, perform them ”by hand” on new objects
  - Applications of morphology:
    - edge detection, connected components, convex hull etc.
    - Verify the examples in the book
- Grey-level morphology:
  - Understand how grey-level erosion and dilation (and opening and closing) works.
  - Understand the effect these operations have on images.
  - Understand top-hat, bottom-hat and what they are used for.