INF 5520 – Digital Image Analysis

TEXTURE

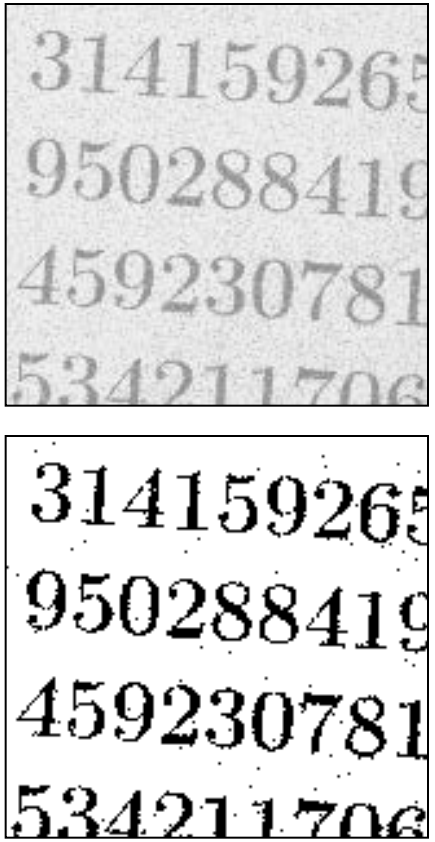
Fritz Albregtsen 28.08.2018

Plan for today

- G&W, “DIP”, Ch.3.3.3 (pages 827-839)
- Why texture, and what is it?
- Statistical descriptors
  - First order
    - Mean, variance, ..., moments, ...
  - Second order
    - Gray level co-occurrence matrices
  - 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.
- Objects are seldom alike, even if they are of same class.
- Often several classes.
- Lighting may vary over image.
- Reflection, color etc. may vary.

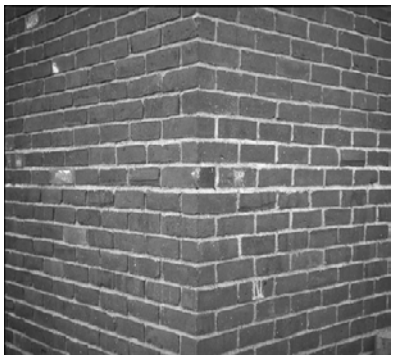
- We perceive and utilize
  - intensity,
  - color,
  - and **texture**.

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

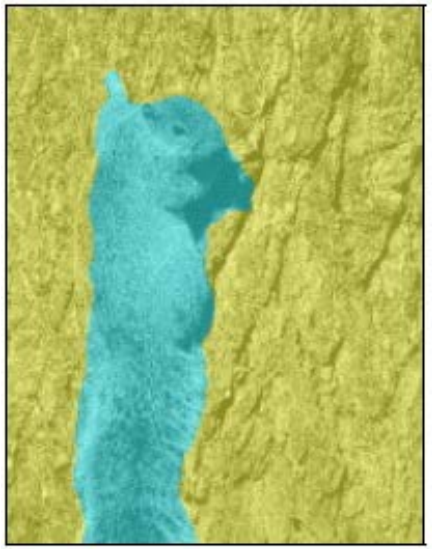
What is texture?

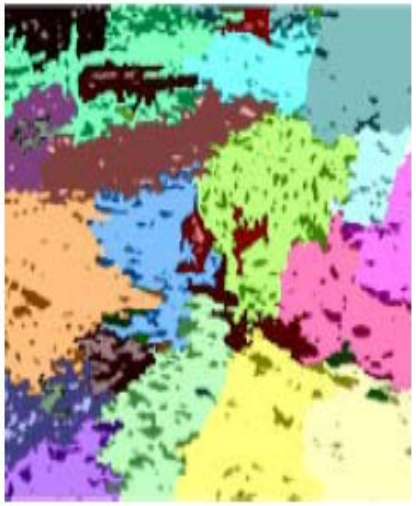
- Intuitively obvious,
  but no precise definition exists
  - ”fine, coarse, grained, smooth” etc
- Texture consists of texture primitives, texels,
  - a contiguous set of pixels with some tonal and/or regional property
- Texture can be characterized by
  - intensity (color) properties of texels
  - 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?





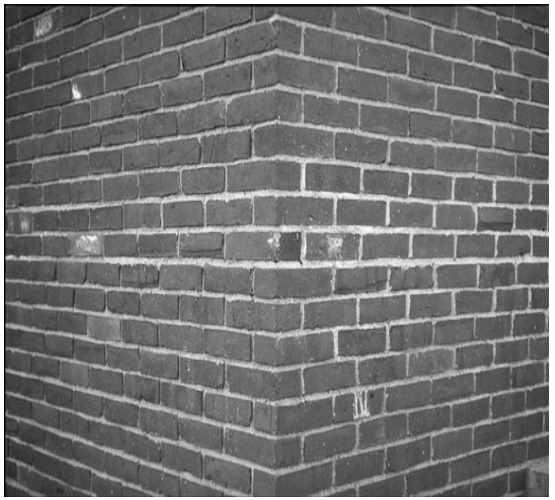
What is a texel?

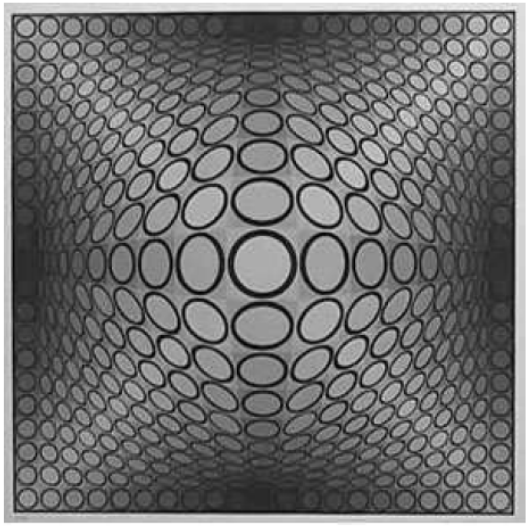
Texel = texture element, the fundamental unit of texture space.
Can be defined in a strict geometrical sense, or statistically.

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

Texture analysis applications

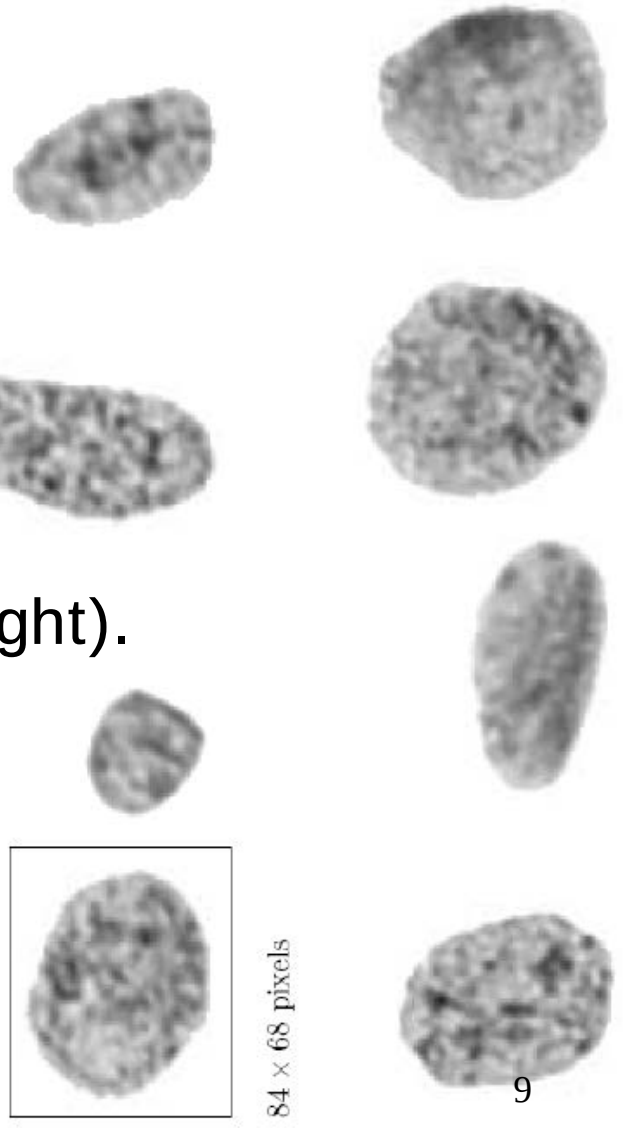
- Segment an image into regions with the same texture, i.e. as a complement to graylevel or color
- Recognize or classify objects in images based on their texture
- Find textural edges in an image, i.e., where the texture changes
- ”shape from texture”
- object detection, compression, synthesis
- Industrial 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**

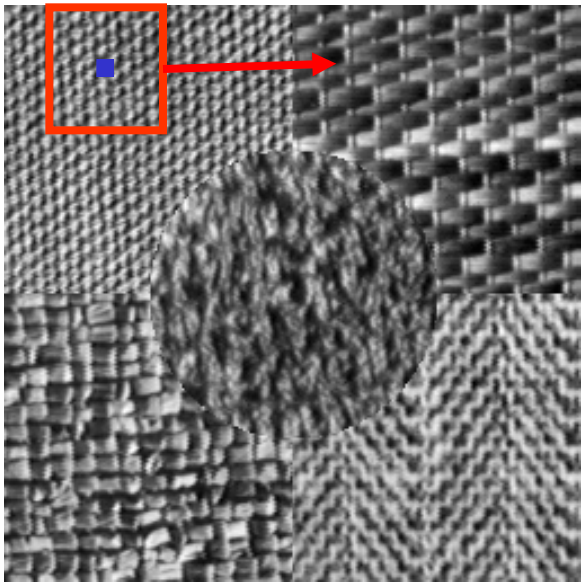


Features

- 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 the local (sliding) window
    - Features that describes 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!

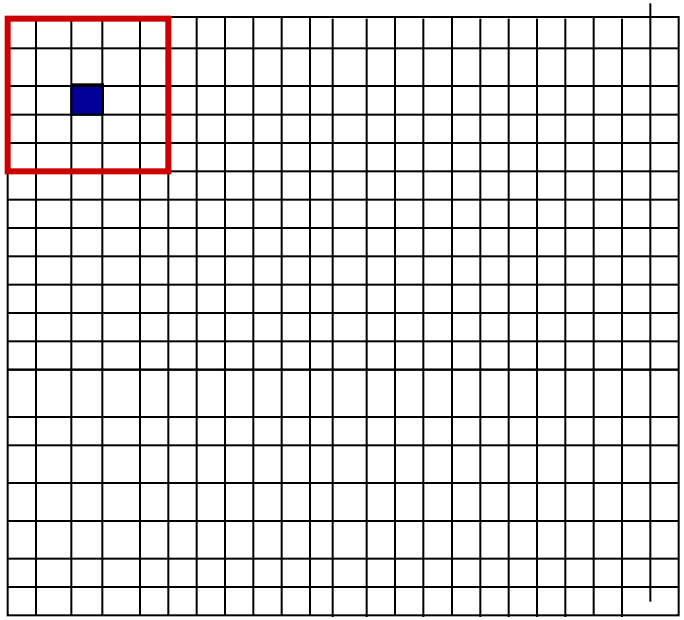
Computing texture images

- 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

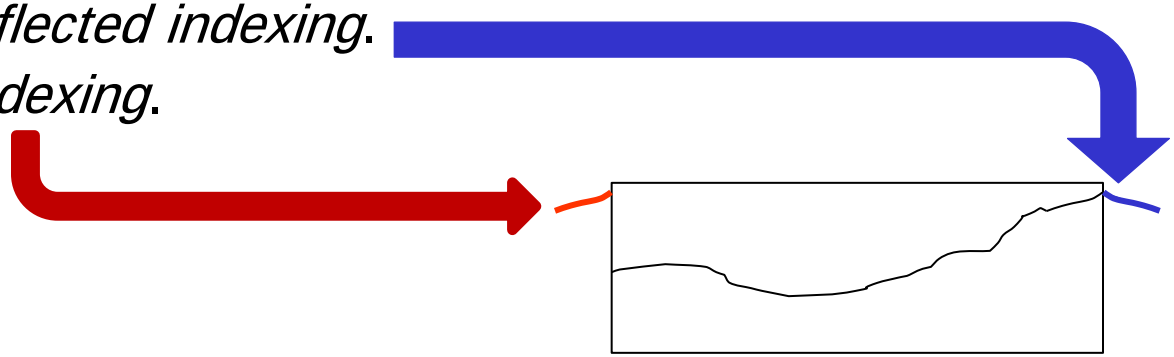
- Pixels close to the image border can be handled in the same manner as for filtering/convolution (reflected indexing, circular indexing)



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



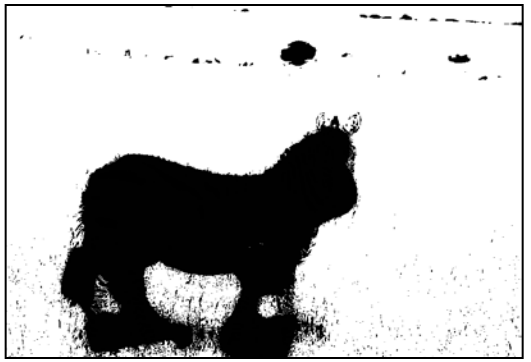
Texture feature image example

Input image.

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

New homogeneity image.

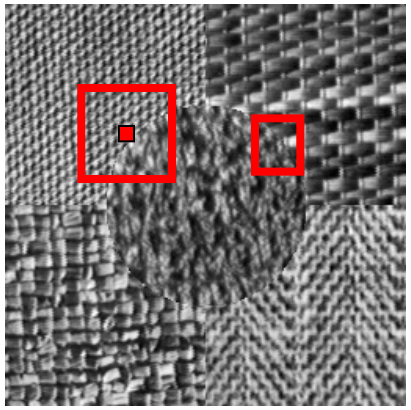
Try to get an image where pixels belonging to the same texture type get similar values.



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



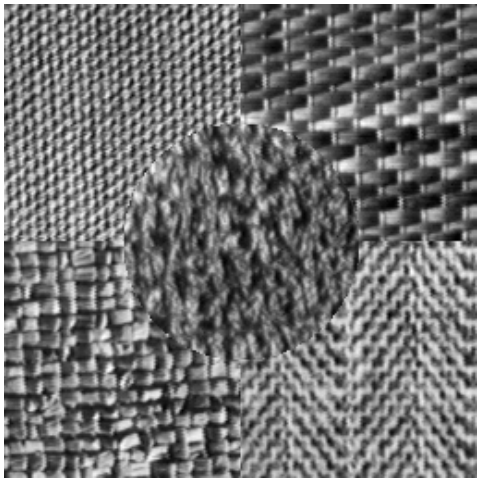
Uncertainty relation

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

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

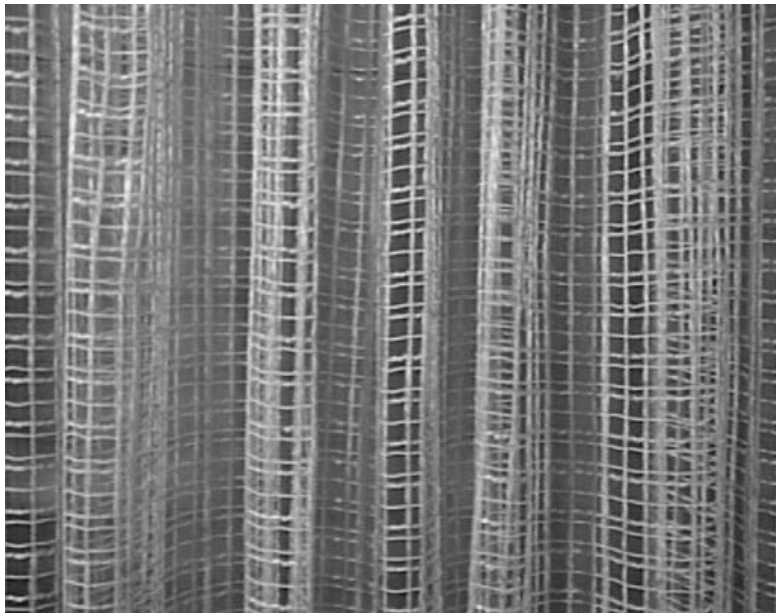
- Related to Heisenberg's uncertainty relation in physics:

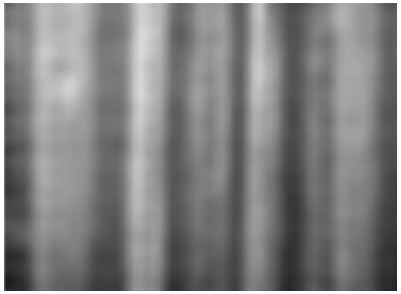
$$\Delta x \Delta p \approx h$$

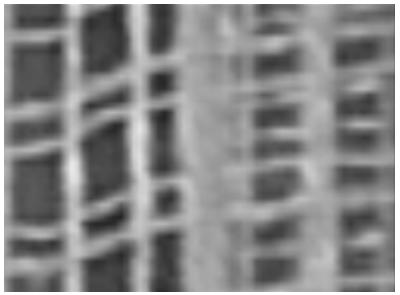


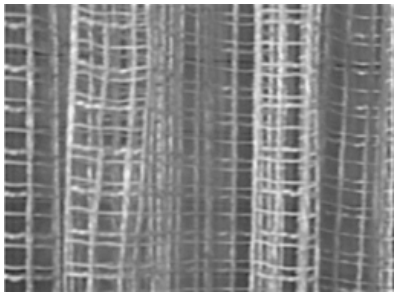
Texture description is scale dependent

- 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.

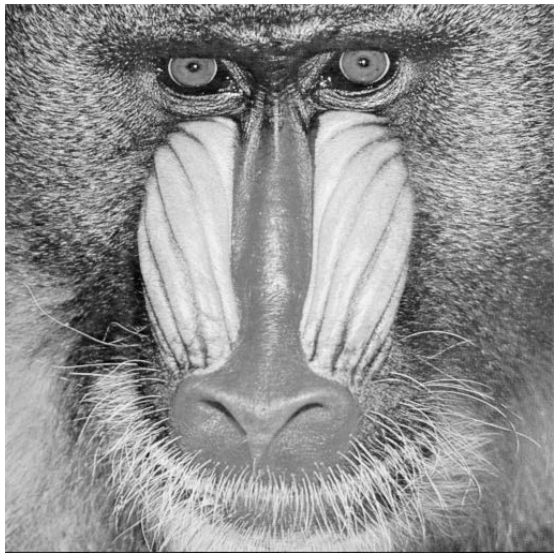








Example of scale dependence



Original image

Variance feature computed in window of size 3x3

Variance feature computed in window of size 15x15

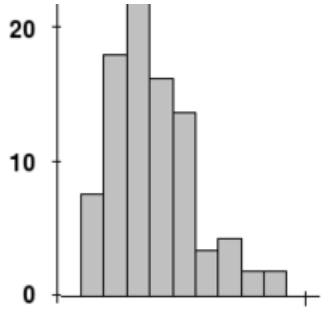
Statistical texture description

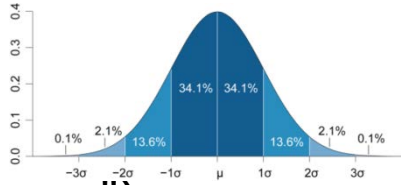
- Describe texture in a region by a vector of statistics (feature vector)
  - First order statistics from graylevel intensity histogram ***p(i)***
    - Mean, variance, 3. and 4. order moment, entropy, energy
  - Second order statistics,
    describing relation between pixel pairs
    - How does the 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 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
  (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

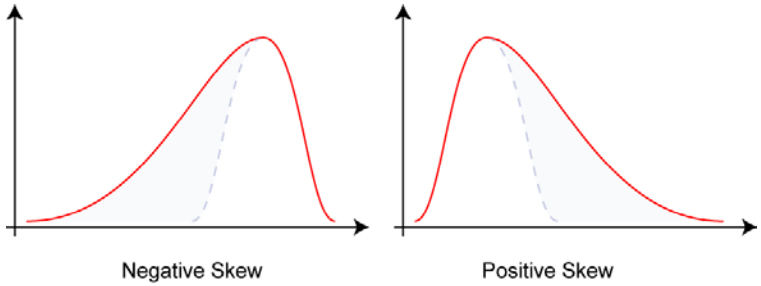
Using variance estimates

- "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



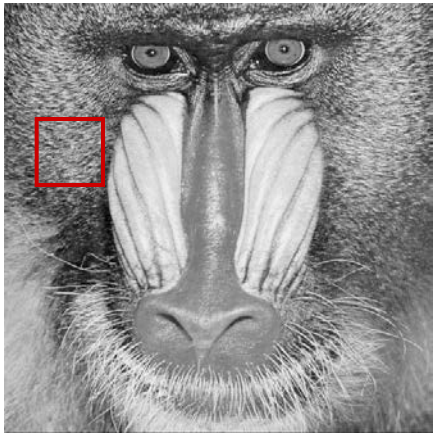
$$\gamma_3 = \frac{1}{\sigma^3}\sum_{i=0}^{G-1}(i-\mu)^3 P(i) = \frac{m_3}{\sigma^3}$$

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

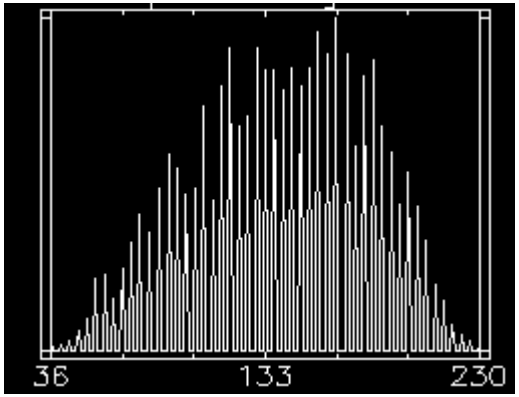
- **Negative skew**: The left tail is longer; the mass of the distribution is concentrated on the right of the figure.

- **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).

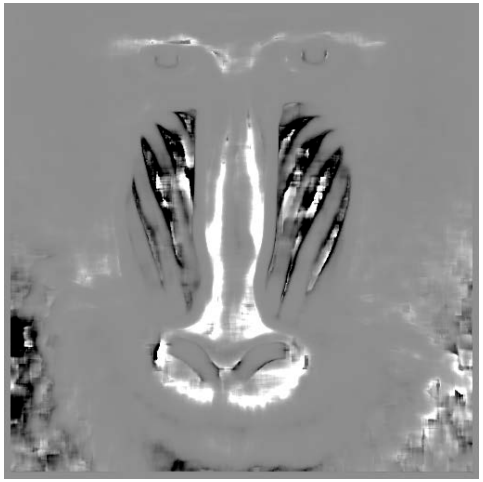
Skewness example



Region 1



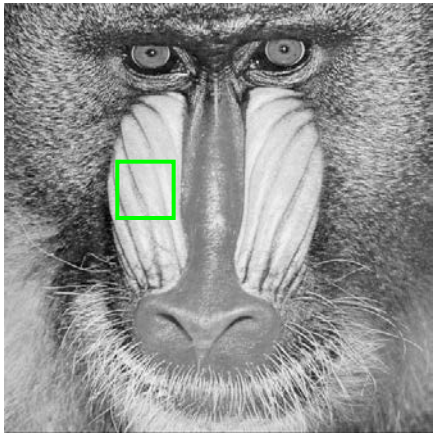
Histogram - Region 1



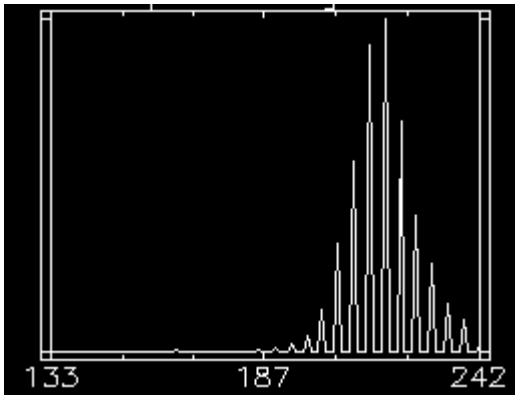
Skewness feature
Computed in 15x15 window

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.

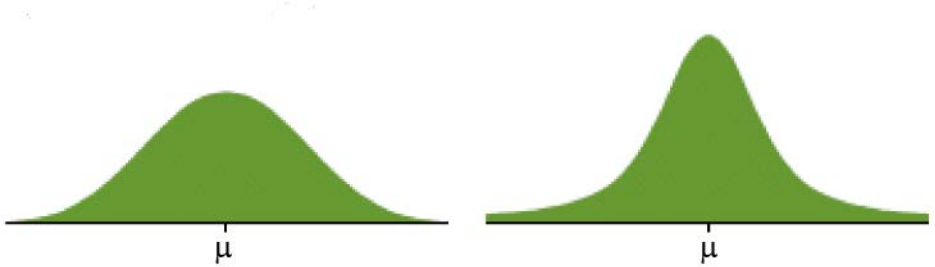


Region 2



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?)

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

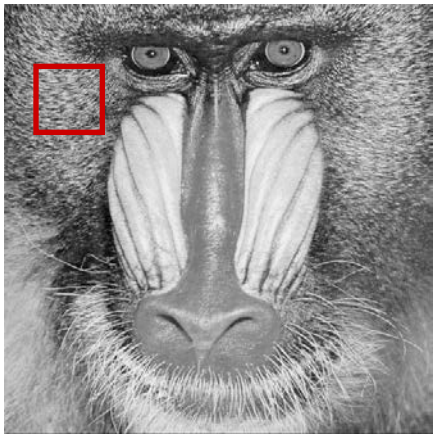
- If all pixel values are the same, H = 0.
- If all pixel values are equally probable:
  - There are G = 2^b gray levels, each having a probability p(i) = 1/G = $1/2^b$, so:

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

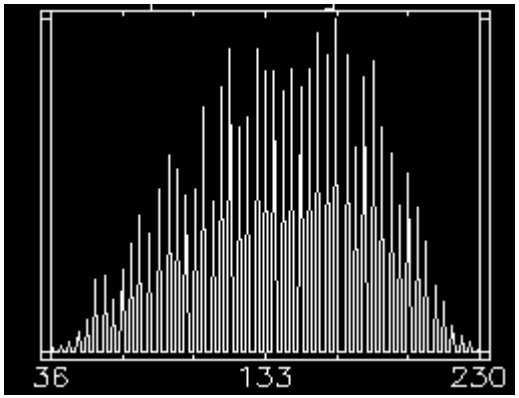
- We see that $0 \leq H \leq b$ (b = number of bits per pixel)

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

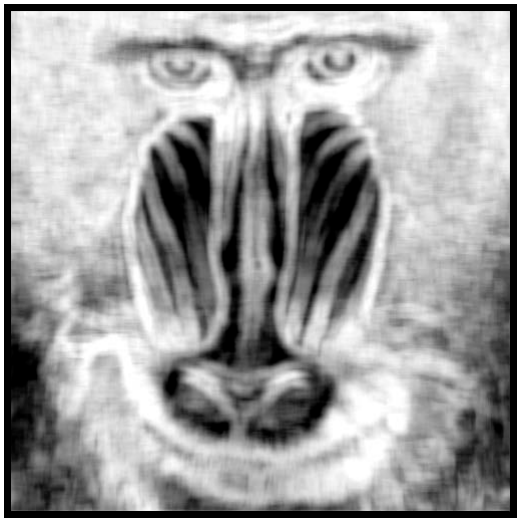
Entropy example

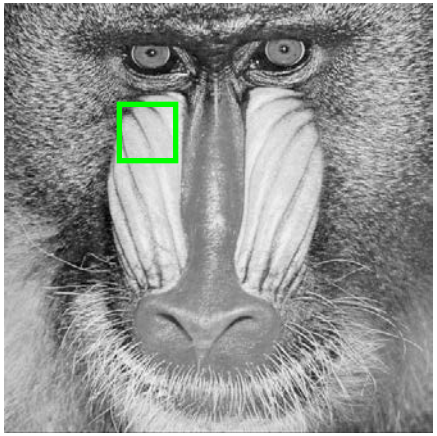


Region 1

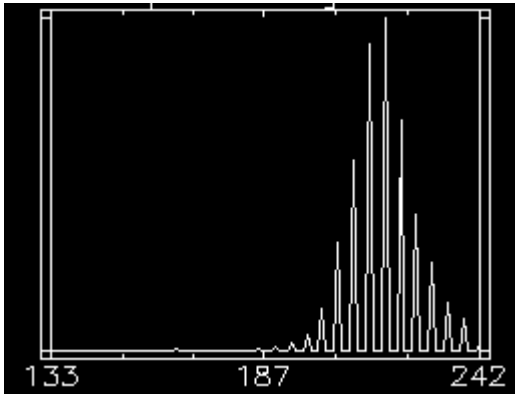


Histogram - Region 1





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)×0+1×1=1

- **Thus, 1/G ≤ E ≤ 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/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

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

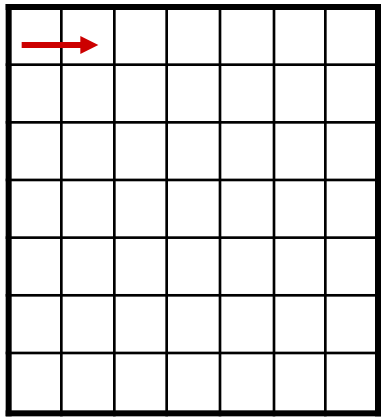
and

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

- 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
- 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

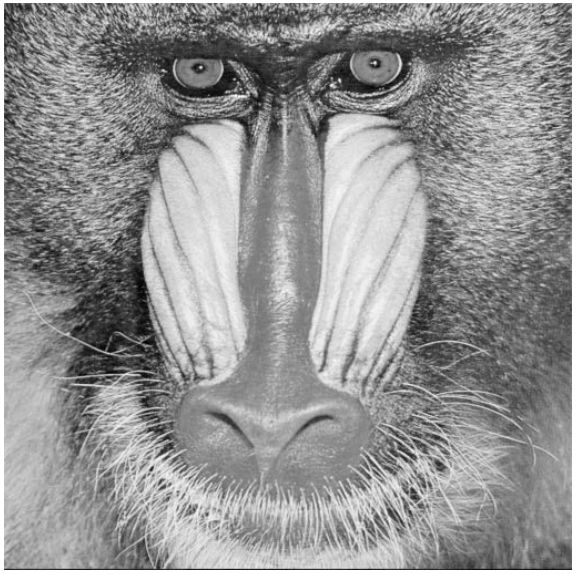
	j=0	1	2	3
i= 0	1/20	2/20	1/20	0
1	0	1/20	3/20	0
2	0	0	3/20	5/20
3	0	0	2/20	2/20

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

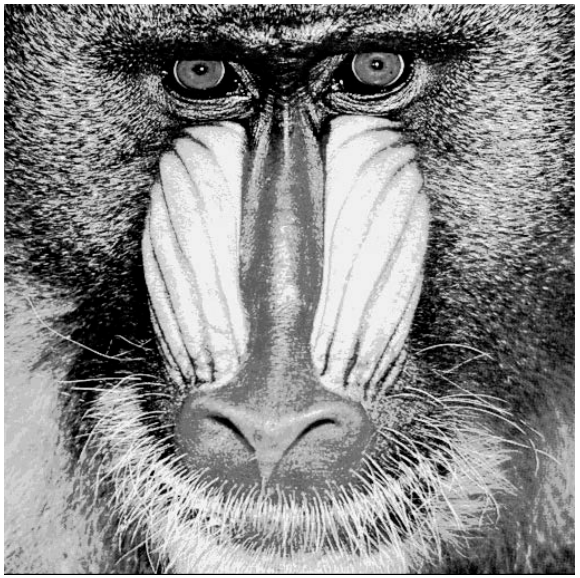
GLCM – practical issues

- 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:
  - *16* graylevels is usually sufficient
  - window should be *31 x 31 – 51* x *51* pixels.

Preprocessing examples

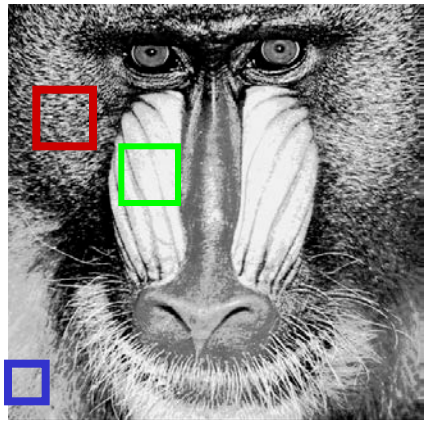


Original image



Histogram-equalized
and requantized to
16 gray levels

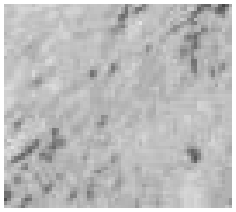
GLCM matrices for subregions



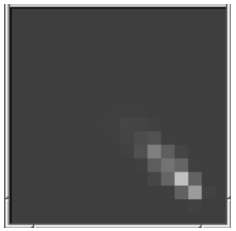
Subregions of image

Nose

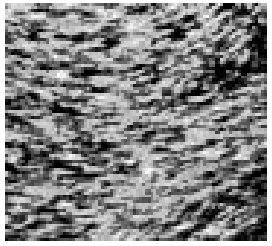
Fur2



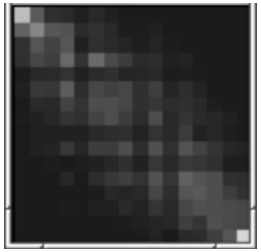
GLCM matrices
d=1, θ=0



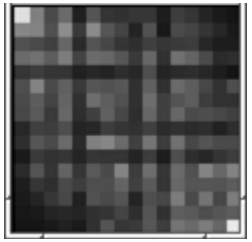
Fur1



GLCM matrix
d=1, θ=0

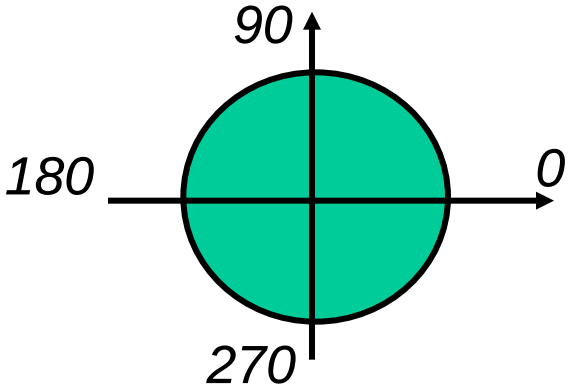


GLCM matrix
d=1, θ=90



GLCM

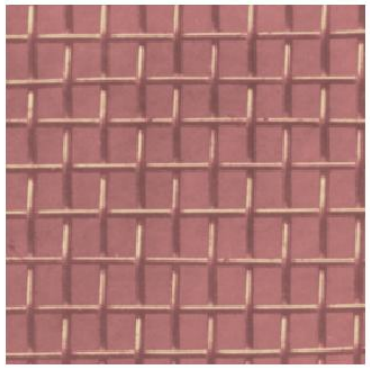
- Usually a good idea to reduce the number of (d,θ) variations evaluated
- Simple pairwise relations:
  - $P(d,0^0) = P^t(d,180^0)$
  - $P(d,45^0) = P^t(d,225^0)$
  - $P(d,90^0) = P^t(d,270^0)$
  - $P(d,135^0) = P^t(d,315^0)$

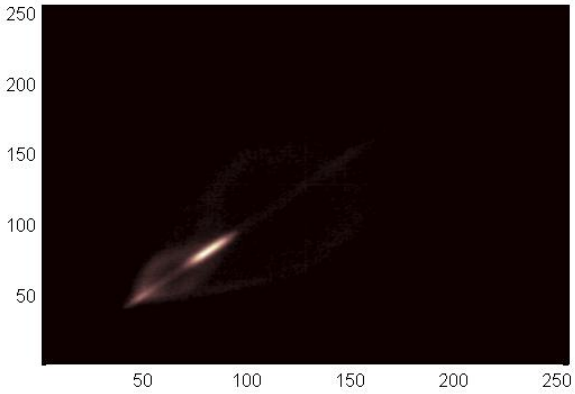


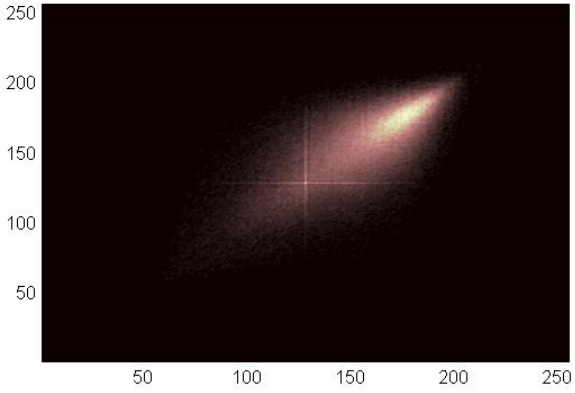
  - *Symmetric GLCM:*
    - *Count "forwards" + "backwards"*
- Isotropic cooccurrence matrix by averaging

  $P(\theta)$, $\theta \in \{0^o, 45^o, 90^o, 135^o\}$
  - *Beware of differences in effective window size!*
- An isotropic texture 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).

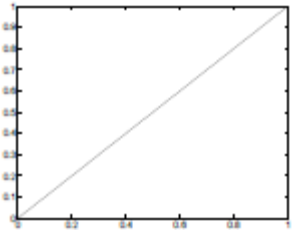
Classical GLCM features

- 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

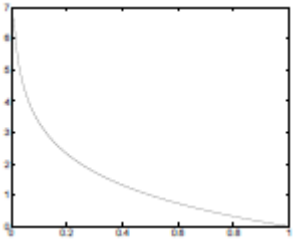
- Angular Second Moment, $ASM = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} \{P(i,j)\}^2$

  - 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, $E = -\sum_{i=0}^{G-1} \sum_{j=0}^{G-1} P(i,j) \times log(P(i,j))$

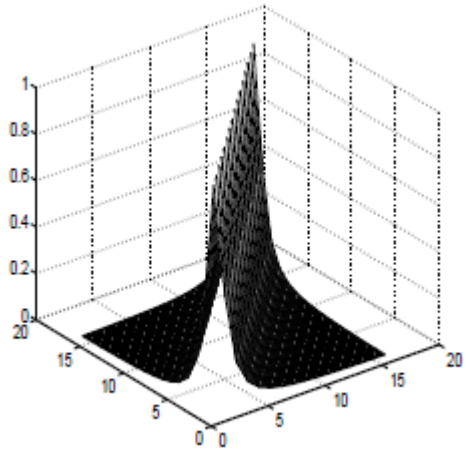
  - Inhomogeneous scenes have high entropy, while a homogeneous scene has a low entropy.
  - *Maximum Entropy is reached when all 2. order probabilities are equal.*
  - *Note that 0 ≤ P(I,j) ≤ 1.*



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

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

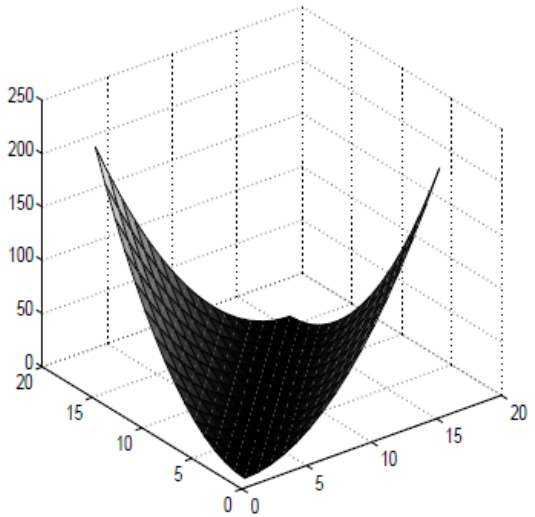
  - IDM is influenced by the homogeneity of the image.
  - Because of the weighting factor, IDM will get small contributions from inhomogeneous areas ($i \neq j$).
  - The result is a low IDM value for inhomogeneous images, and a relatively higher value for homogeneous images.

Position-based GLCM Features-2

- Inertia, also called Contrast:

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

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



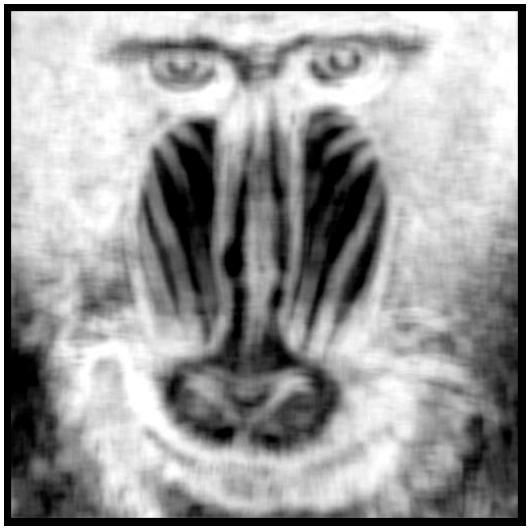
- The Inertia weighting function is zero along the diagonal (i = j), and increases towards $(G-1)^2$ away from the diagonal.

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

GLCM feature image examples, w= 15

GLCM contrast

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

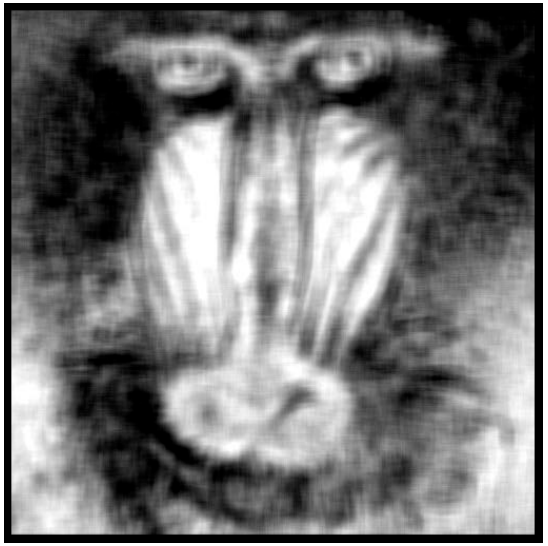


GLCM entropy

GLCM entropy is negatively correlated with ASM

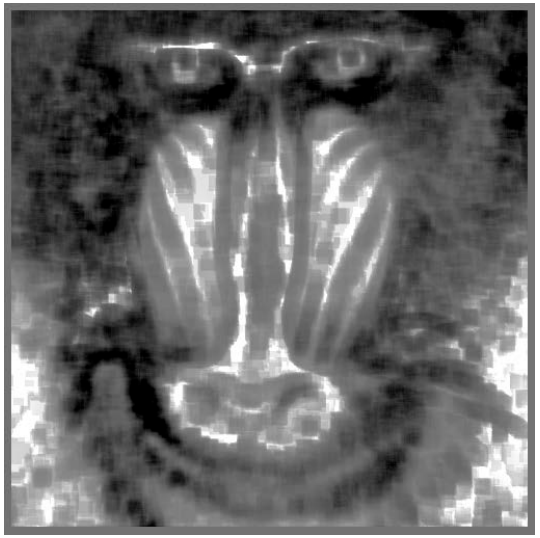
GLCM variance

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 search and selection of reduced dimensionality feature sets.
- It should also motivate us to generate 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.

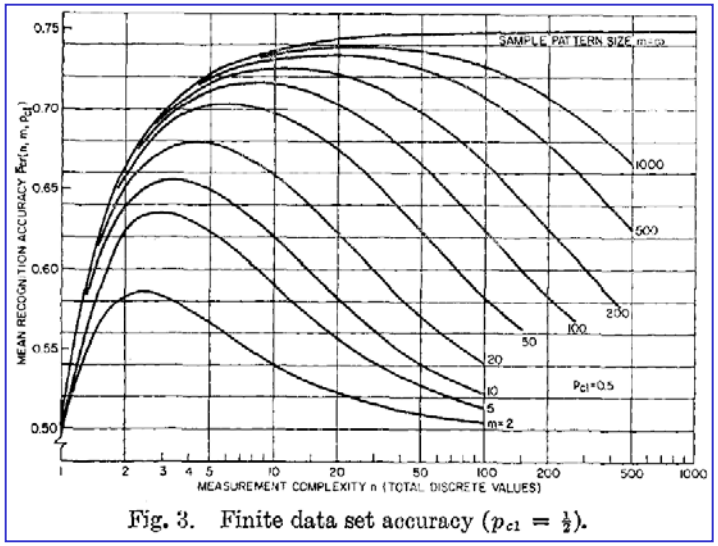


Fig. 3. Finite data set accuracy ($p_{c1} = \frac{1}{2}$).

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

  « Inertia / Contrast»: $(i-j)^2$

Low-D adaptive weighting functions

- Instead of a large number of *predefined, non-adaptive features*, we could find a few weighting functions adapted to the images.
- This would extract information from the parts of the GLCM matrix that actually contain information about texture differences between e.g., two different clinical classes (i.e., good and bad prognosis).
- Given a number of images of each of two classes, compute
  - the average $P(i,j|\omega)$
  - the variance $\sigma^2(i,j1\omega)$ of P
  - for each position (i,j) of the GLCM for each class, ω.

Low-D adaptive weighting functions

- Then for each position (i,j), compute
  - the class difference matrix value

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

  - and the *Mahalanobis* class distance matrix

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

- *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).

Sum and difference histograms

- The sum histogram *S* is simply the histogram of the sums of pixels *d*x 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 *d*x 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

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

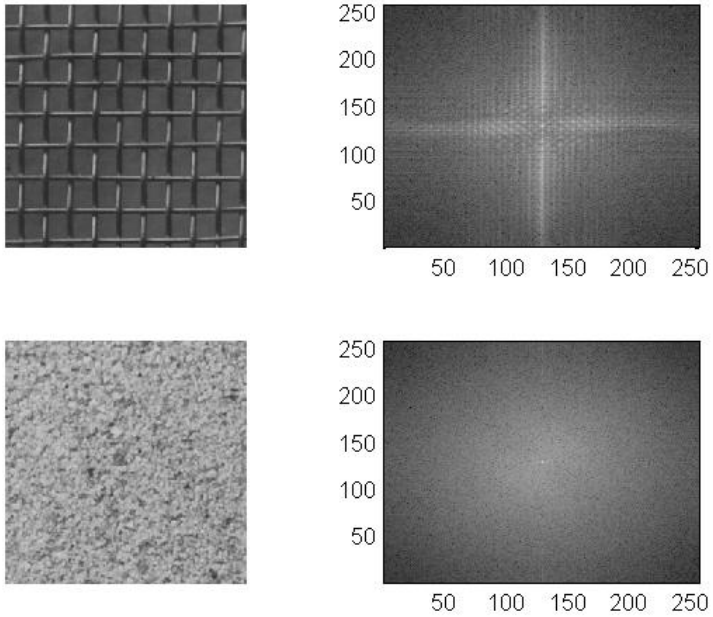
- Some of the features mentioned earlier is derived from the histograms
  - Sum Average, $AVE = \sum_{i=0}^{2G-2} P_s$
  - Sum Entropy, $SEN = -\sum_{i=0}^{2G-2} P_s(i) \log (P_s(i))$
  - Difference Entropy, $DEN = -\sum_{i=0}^{G-1} P_d(i) \log (P_d(i))$
  - Inverse Difference moment, $IDM = \sum_{i=0}^{G-1}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

Fourier analysis

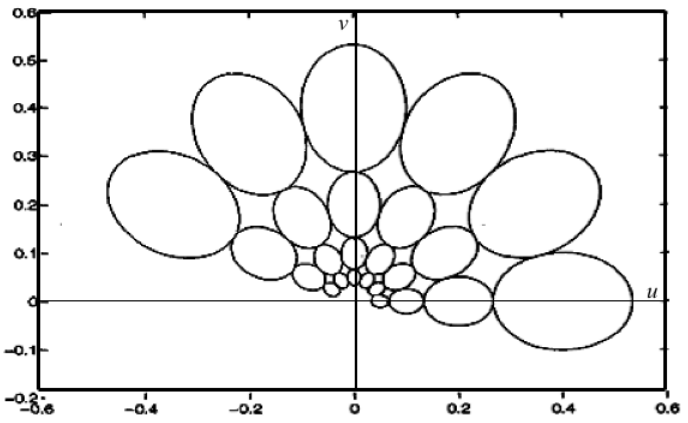
- The Fourier spectra give direction and frequency for periodic or near periodic 2D patterns
- Local FFT in windows
- Texture with a dominating direction will have peaks in the spectra along a line orthogonal to the texture orientation
- High frequency = fine texture = peaks in the spectra far from the 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**