CHAPTER 6

IMAGE SEGMENTATION ALGORITHMS

Contents

- **Introduction**
- **Image segmentation by thresholding**
- **Split/merge and region growing**
- **Relaxation algorithms in region analysis**
- **Connected component labeling**
- **Texture description**

Introduction

- The shape of an object can be described in terms of :
  - its boundary
    - ✓ requires image edge detection and following
  - the region it occupies
    - ✓ requires image segmentation in homogeneous regions
    - ✓ Image regions are expected to have homogeneous characteristics (e.g. intensity, texture)
    - ✓ These characteristics form the feature vectors that are used to discriminate one region from another

Introduction

- An image domain X must be segmented in N different regions $R_1,\ldots,R_N$
- The segmentation rule is a logical predicate of the form $P(R)$
- Image segmentation partitions the set X into the subsets R_i, $i=1,\ldots,N$, having the following properties:

$$X = \bigcup_{i=1}^{N} R_i$$

$$R_i \cap R_j = 0 \text{ for } i \neq j$$

$$P(R_i) = TRUE \text{ for } i = 1,2,\ldots, N$$

$$P(R_i \cup R_j) = FALSE \text{ for } i \neq j$$

Introduction

- The segmentation result is usually a logical predicate of the form $P(R,x,t)$
  - x is a feature vector associated with an image pixel
  - t is a set of parameters (usually thresholds) A simple segmentation rule has the form:

$$P(R): \; f(k,l) < T$$

- In the case of colour images the feature vector x can be the three RGB image components $[f_R(k,l), f_G(k,l), f_B(k,l)\, l)]^T$
- A simple segmentation rule may have the form:

$$P(R,x,t): \; (f_R(k,l) < T_R) \;\&\&\; (f_G(k,l) < T_G) \;\&\&\; (f_B(k,l) < T_B)$$

Introduction

- A region R is called *connected* if :
  - any two pixels (x_A, y_A), (x_B, y_B) belonging to R can be connected by a path (x_A, y_A),..., (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}),...,(x_B, y_B), whose pixels (x_i, y_i) belong to R

and

  - any pixel (x_i, y_i) is adjacent to the previous pixel (x_{i-1}, y_{i-1}) and the next pixel (x_{i+1}, y_{i+1}) in the path

- A pixel (x_k, y_k) is said to be *adjacent* to the pixel (x_l, y_l) if it belongs to its immediate neighbourhood

Introduction

We can define two types of neighbourhoods:

- The *4-neighbourhood* of a pixel *(x,y)* is the set that includes its horizontal and vertical neighbours:

$$N_4((x,y))=\{(x+1,y),(x-1,y),(x,y+1),(x,y-1)\}$$

- The *8-neighbourhood* of *(x,y)* is a superset of the 4-neighbourhood and contains the horizontal, vertical and diagonal neighbours:

$$N_8((x,y))=$$
$$N_4((x,y))\cup\{(x+1,y+1),(x-1,y-1),(x+1,y-1),(x-1,y+1)\}$$

Introduction

- The paths defined by using the 4-neighbourhood consist of horizontal and vertical streaks of length

$$Äx = Äy = 1$$

- The paths using the 8-neighbourhood consist of horizontal and vertical streaks of length 1 and of diagonal streaks having length $\sqrt{2}$

Introduction

- Region segmentation techniques can be grouped in three different classes:
  - *Local techniques* are based on the local properties of the pixels and their neighbourhoods
  - *Global techniques* segment an image on the basis of information obtained globally (e.g. by using the image histogram)
  - *Split*, *merge* and *growing* techniques use both the notions of homogeneity and geometrical proximity

Image segmentation by thresholding

- The simplest image segmentation problem occurs when an image contains
  - an object having homogeneous intensity
  - a background with a different intensity level
- Such an image can be segmented in two regions by simple thresholding:

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

- The choice of threshold T can be based on the image histogram

Image segmentation by thresholding

- If the image contains one object and a background having homogeneous intensity, it usually possesses a bimodal histogram like the one shown below:

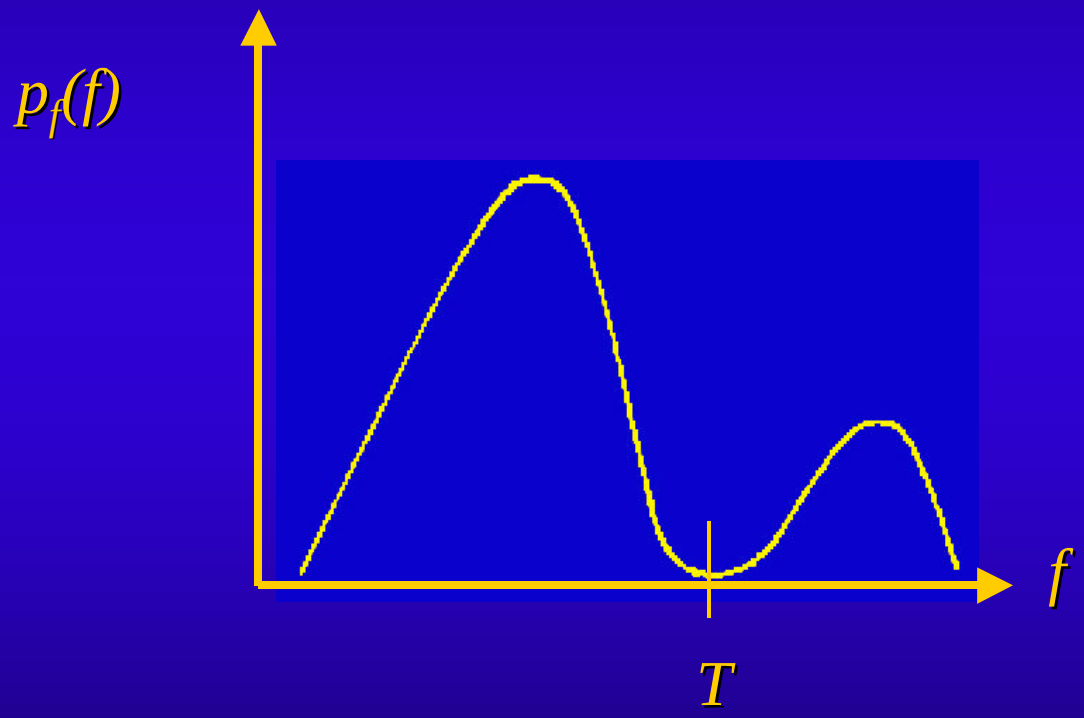


Image segmentation by thresholding

- If the histogram is noisy:
  - The calculation of the local histogram minimum is difficult
  - Histogram smoothing (e.g. by using one-dimensional low-pass filtering) is recommended

- If the intensity of the object or of the background is slowly varying:
  - The image histogram may not contain two clearly distinguished lobes
  - A spatially varying threshold can be applied

Image segmentation by thresholding

- In certain cases the region boundary is desired.
  - If the segmented image *g(x,y)* is available, the boundary obtained by finding the transitions from one region to the other:

$$b(x,y)=\begin{cases}1 & \text{if } \{(g(x,y)\in R_i \text{ and } g(x,y-1)\in R_j, i\neq j)\\ & \text{or } (g(x,y)\in R_i \text{ and } g(x-1,y)\in R_j, i\neq j)\}\\ 0 & \text{otherwise}\end{cases}$$

- The same procedure can be used for ***multiple thresholding***

Image segmentation by thresholding

- Multiple thresholding can be used for segmenting images containing N objects, provided that each object R_i occupies a distinct intensity range which is defined by two thresholds T_{i-1}, T_i
  - The thresholding operation takes the following form:

$$g(x,y) = R_i \text{ if } T_{i-1} \le f(x,y) \le T_i, \quad i = 1,\ldots,N$$

    - The thresholds can be obtained from the image histogram
  - In many cases, the various histogram lobes are not clearly distinguished

Image segmentation by thresholding

- Histogram modification : Perform edge detection and exclude all pixels belonging to edges from the histogram calculation
- Another approach is to define a modified histogram :

$$h(i) = \sum_{k=0}^{N_1-1} \sum_{l=0}^{N_2-1} t(e(k,l)d(f(k,l) - i)$$

e(k,l) is an edge detector output
ä(i) is the delta function

Image segmentation by thresholding

- The thresholding operation for the segmentation in N different regions:

$$g(x,y)=\begin{cases} R_i & \text{if } i[L/N]\le f(k,l)<(i+1)[L/N],\quad i=0,1,\ldots,N-2 \\ R_{N-1} & \text{if } (N-1)[L/N]\le f(k,l)<L \end{cases}$$

  - A monotonically decreasing function t can be chosen for histogram modification:

$$t(e(k,l))=\frac{1}{1+|e(k,l)|}$$

Image segmentation by thresholding

(a) **(b)**

- *(a)* Original image
- *(b)* Image segmentation in four equirange regions

Image segmentation by thresholding

- If the image histogram is concentrated in a small intensity range:
  - Uniform thresholding does not give good results
  - *Non-uniform* thresholding creates much better results in this case
- A non-uniform thresholding technique is based on the histogram equalization technique
  - If *G(f(k,l))* is the transformation function then

$$g(k,l)=\begin{cases} R_i & \text{if } i[L/N]\le G(f(k,l))<(i+1)[L/N] \\ R_{N-1} & \text{if } (N-1)[L/N]\le G(f(k,l))<L \end{cases}$$

Split/merge and region growing algorithms

- Geometrical proximity plays an important role in image segmentation.
- Segmentation algorithms must incorporate both proximity and homogeneity

SPLIT/MERGE AND REGION GROWING ALGORITHMS

- A simple approach to image segmentation is to start from some pixels (*seeds*) representing distinct image regions and to *grow* them, until they cover the entire image

Split/merge and region growing algorithms

- The pixel seeds are chosen in a supervised mode
- At least one seed s_i, $i = 1, \ldots, N$ is chosen per image region R_i
- In order to implement region growing, we need a rule describing a growth mechanism and a rule checking the homogeneity of the regions after each growth step

Split/merge and region growing algorithms

- The growth mechanism is simple: at each stage *(k)* and for each region $R_i^{(k)}$, $i = 1, \ldots, N$, we check if there are unclassified pixels in the 8-neighbourhood of each pixel of the region border

- Before assigning such a pixel x to a region $R_i^{(k)}$, we check if the region homogeneity:

$$P(R_i^{(k)} \cup \{x\}) = TRUE$$

  is still valid

Split/merge and region growing algorithms

- The arithmetic mean m_i and standard deviation $ó_i$ of a class R_i having n pixels:

$$m_i = \frac{1}{n} \sum_{(k,l)\in R_i} f(k,l)$$

$$s_i = \sqrt{\frac{1}{n} \sum_{(k,l)\in R_i} [f(k,l) - m_i]^2}$$

can be used to decide if the merging of two regions R_1, R_2 is allowed

  - If their arithmetic means are close to each other ($|m_1 - m_2| < kó_i$, $i=1,2$) the two regions are merged

Split/merge and region growing algorithms

- Region merging can be incorporated in the growing mechanism
- If no a priori information is available about the image, the image can be scanned in a row-wise manner.
  - If we are currently at the pixel *(k,l)*:
    - ✓ First, we try to merge this pixel with one of its adjacent regions R_i
    - ✓ If this merge fails, or if no adjacent region exists (e.g. for the pixel *(0,0)*), this pixel is assigned to a new region

Split/merge and region growing algorithms

- The merging rule can be based on the region mean and standard deviation described by m_i and $\acute{o}_i$

- If merging $P(R_i \dot{E} (k,l))$ was allowed, the updated mean and standard deviation would be given by:

$$m_i^{'} = \frac{1}{n+1}[f(k,l) + nm_i]$$

$$\acute{o}_i^{'} = \sqrt{\frac{1}{n+1}\left(n\acute{o}_i^{2} + \frac{n}{n+1}[f(k,l) - m_i]^{2} \right)}$$

Split/merge and region growing algorithms

- Merging is allowed if the pixel intensity is close to the region mean value:

$$\left| f(k,l) - m_i \right| \leq T_i(k,l)$$

- If more than one merge are possible, the region with the closest mean value is chosen
- Threshold T_i varies, depepending on the region R_i and the intensity of the pixel $f(k,l)$. It can be chosen this way:

$$T_i(k,l) = \left(1 - \frac{\acute{o}_i'}{m_i'} \right) T$$

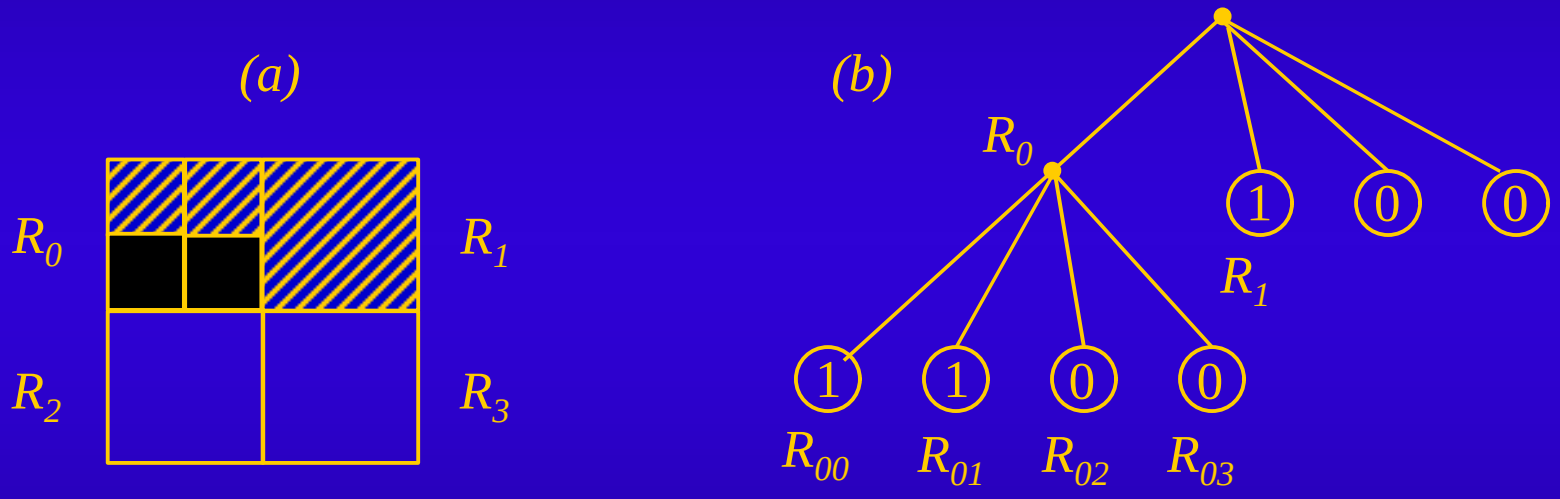
Split/merge and region growing algorithms

- The opposite approach to region merging is region splitting
  - It is a top-down approach and it starts with the assumption that the entire image is homogeneous
  - If this is not true, the image is split into four subimages
  - This splitting procedure is repeated recursively until we split the image into homogeneous regions

Split/merge and region growing algorithms

- If the original image is square $N \times N$, having dimensions that are powers of 2 ($N=2^n$):
  - All regions produced by the splitting algorithm are squares having dimensions $M \times M$, where M is a power of 2 as well ($M=2^m$, $m \leq n$)
  - Since the procedure is recursive, it produces an image representation that can be described by a tree whose nodes have four sons each
  - Such a tree is called a ***quadtree*** and is a very convenient region representation scheme

Split/merge and region growing algorithms



Split/merge and region growing algorithms

Splitting techniques disadvantage.

They create regions that may be adjacent and homogeneous, but not merged → *split and merge* algorithm

- It is an iterative algorithm that includes both splitting and merging at each iteration:
  - ✓ If a region R is inhomogeneous ($P(R)=FALSE$) then is split into four subregions
  - ✓ If two adjacent regions R_i, R_j are homogeneous ($P(R_i \grave{E} R_j) = TRUE$), they are merged
  - ✓ The algorithm stops when no further splitting or merging is possible

Split/merge and region growing algorithms

- The split and merge algorithm produces more compact regions than the pure splitting algorithm
- Its major disadvantage is that it does not produce quadtree region descriptions
- Several modifications of the basic split and merge algorithm have been proposed to solve this problem
  - The most straightforward procedure is to use the splitting algorithm and to postpone merging until no further splitting is possible

Split/merge and region growing algorithms

(a)

(b)

(c)

(d)

- *(a)* Result of the region growing algorithm
- *(b)* Output of the region merge algorithm
- *(c)* Result of the region split algorithm
- *(d)* Output of the split and merge algorithm

Relaxation algorithms in region analysis

- All previous region segmentation methods are deterministic (they assign each image pixel to just one region)
- Such a segmentation is desirable, but not always useful, because they treat ambiguous cases in a rather inflexible way

Relaxation algorithms in region analysis

- It is more useful to produce confidence vectors p_k for each pixel x_k that contain the probabilities $p_k(i)$ that a pixel x_k belongs to a class R_i, $i=1,...,N$:

$$\boldsymbol{p}_k=[p_k(1),\ldots, p_k(N)]^T$$

  - Pixel x_k is assigned to the region R_l having the maximal probability $p_k(l)$
  - Probabilities $p_k(l)$, called *confidence weights*, must satisfy the following relations:

$$0 \le p_k(i) \le 1 \quad , \qquad \sum_{i=1}^{N} p_k(i) = 1$$

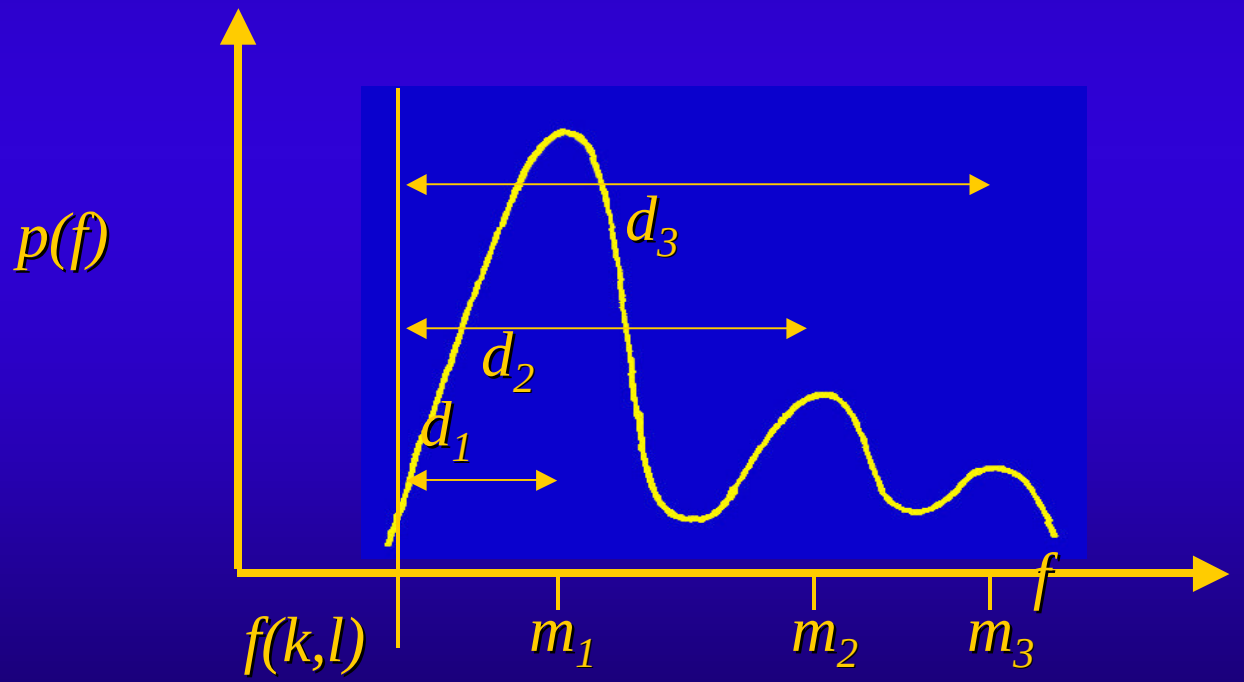
Relaxation algorithms in region analysis

- Let m_i, $i=1,...,N$, be the arithmetic means of the intensity of each region that usually correspond to histogram peaks and $f(x_k) = f(n,l)$ the pixel intensity at location $x_k=(n,l)$. The initial estimate of the confidence weight is given by:

$$p_k^{(0)}(i) = \frac{\dfrac{1}{|f(n,l) - m_i|}}{\displaystyle\sum_{i=1}^{N} \frac{1}{|f(n,l) - m_i|}} \quad , \qquad i = 1,...,N$$

Relaxation algorithms in region analysis

- It is inversely proportional to the distance $d_i = |f(n,l) - m_i|$ of the pixel intensity from the region arithmetic mean m_i:



Relaxation algorithms in region analysis

- In many cases, it is highly probable that two adjacent pixels belong to two specific classes R_i, R_j. Such classes are called *compatible*

- *Incompatible* regions are those that are not expected to be found in adjacent image locations

Relaxation algorithms in region analysis

- The compatibility between two regions R_i, R_j is described in terms of a ***compatibility function r(i,j)***, whose range is $-1 \leq r(i,j) \leq 1$. The value of the compatibility function has the following meaning:

$$r(i,\ j) = \begin{cases} < 0 & \text{Regions } R_i, R_j \text{ are incompatible} \\ = 0 & \text{Regions } R_i, R_j \text{ are independent} \\ > 0 & \text{Regions } R_i, R_j \text{ are compatible} \end{cases}$$

Relaxation algorithms in region analysis

- Compatibility functions are known a priori or can be estimated from an initial image segmentation
  - Incompatible regions tend to compete in adjacent image pixels, whereas compatible regions tend to cooperate
  - Competition and cooperation can be continued in an iterative way until a steady state is reached

    ✓ Each pixel x_k receives confidence contributions from any pixel x_l lying in its neighbourhood

    ✓ The resulting change in confidence weight $p_k(i)$ of the pixel x_k at step (n) is the following:

Relaxation algorithms in region analysis

$$\ddot{A}p_k^{(n)}(i) = \sum_l d_{kl}\left[\sum_{j=1}^{N} r_{kl}(i,j)p_l^{(n)}(j)\right]$$

- The sum of the parameters d_{kl} is chosen to be equal to 1:

$$\sum_l d_{kl} = 1$$

- The updated probabilities for the pixel x_k are given by:

$$p_k^{(n+1)}(i) = \frac{p_k^{(n)}(i)[\,1 + \ddot{A}p_k^{(n)}(i)]}{\sum_{i=1}^{N} p_k^{(n)}(i)[\,1 + \ddot{A}p_k^{(n)}(i)]}$$

Relaxation algorithms in region analysis

- The iterative equations form the *relaxation labelling algorithm for region segmentation*

- The iterations stop when convergence is achieved

- The algorithm is expected to produce relatively large connected homogeneous image regions by removing small spurious noisy regions within larger regions

Connected component labelling

- Digital image segmentation produces either a binary or a multivalued image output *g(k,l)*
  - Each image region is labelled by a region number
  - Each region may consist of several disconnected subregions

- An important task is to identify and label the various connected components of each region
  - **Connected component labelling** assigns a unique number to each blob of 1s. If each blob corresponds to a single object, connected component labelling counts the objects that exist in a binary image

Connected component labelling

- Labelling algorithms can be divided into two large classes:
  - *Local neighbourhood* algorithms (performing iterative local operations)
  - *Divide-and-conquer* algorithms

Connected component labelling

- A local neighbourhood algorithm is described below:

  - ✓ The image is scanned in a row-wise manner until the first pixel at the boundary of a binary image object is hit

  - ✓ A ‘fire’ is set at this pixel that propagates to all pixels belonging to the 8-neighbourhood of the current pixel

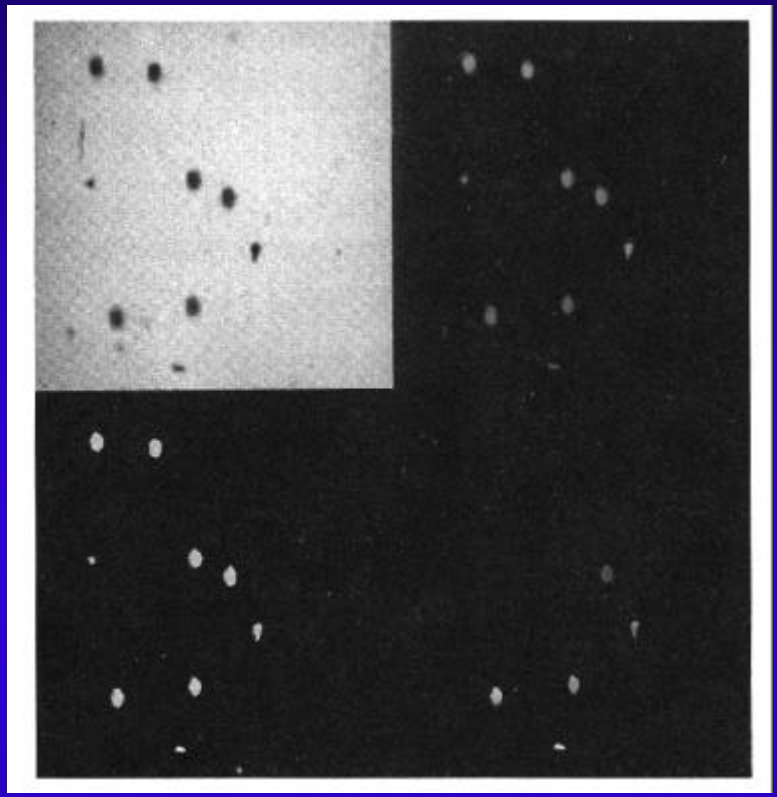
Connected component labelling

- This operation is continued recursively until all pixels of the image object are ‘burnt’ and the fire is extinguished

- After the end of this operation, all pixels belonging to this object have value *0* and cannot be distinguished from the background. A by-product of the algorithm is the area of the object

- This procedure is repeated until all objects in the image are counted

Connected component labelling



- *(a)* Microscope image
- *(b)* Negative image
- *(c)* Thresholded image
- *(d)* Labelled connected regions

Connected component labelling

- Each pixel *f(n, l)* having value 1 is labelled by the concatenation of its *n, l* coordinates
- We scan the labelled image
- We assign to each pixel the minimum of the labels in its 4-connected or 8-connected neighbourhood

- This process is repeated until no more label changes are made

Connected component labelling

- Another local neighbourhood algorithm is the blob colouring algorithm

- It has two passes:
  - In the first pass, colours are assigned to image pixels by using a three-pixel *L*-shaped mask, while colour equivalencies are established and stored, when needed
  - In the second pass, the pixels of each connected region are labelled with a unique colour by using the colour equivalences obtained in the first pass

Connected component labelling

- Another local neighbourhood algorithm is based on the shrinking operation:
  - If a pixel *f(n, l)* has value 1, it retains this value after local shrinking if and only if at least one of its East, South or South-East neighbours has value 1
  - This local operation is described by the following recursive relation:

$$f(n, l) = h[\, h[\, f(n, l-1) + f(n, l) + f(n+1, l) - 1] + h[\, f(n, l) + f(n+1, l-1) - 1]\,]$$

Connected component labelling

- where function **h(t)** is given by:

$$h(t) = \begin{cases} 0 & \text{for} \quad t \leq 0 \\ 1 & \text{for} \quad t > 0 \end{cases}$$

- After repeated scanning of the binary image with the above, each connected component shrinks to the North - West corner of its bounding box, before it vanishes at the next shrinking operation

Connected component labelling

- Divide-and-conquer algorithm for connected component labelling uses the split and merge algorithm :
  - Inhomogeneous regions consisting of 0s and 1s are split recursively until we reach homogeneous regions consisting only of 1s
  - These regions are assigned a unique label. This is the split step
  - Label equivalences can be established by checking the borders of all homogeneous regions
  - Those regions having equivalent labels are merged to a single connected component

Texture description

- The texture of an image, is a measure of image coarseness, smoothness and regularity

- Texture description techniques can be grouped into three large classes:

  - Statistical
    - ✓ Based on region histograms
    - ✓ They measure contrast, granularity, and coarseness

Texture description

- Spectral
  - ✓ Based on the autocorrelation function of a region or on the power distribution in the Fourier transform domain in order to detect texture periodicity

- Structural
  - ✓ They describe the texture by using pattern primitives accompanied by certain placement rules

Texture description

- The simplest texture descriptors are based on the image histogram $p_f(f)$
- Let f_k, $k = 1, ..., N$ be the various image intensity levels
- The first four central moments are given by:

  - ***Mean*** $$\mu = \sum_{k=1}^{N} f_k \, p_f(f_k)$$

  - ***Skewness*** $$\mu_3 = \frac{1}{\sigma^3} \sum_{k=1}^{N} (f_k - m)^3 \, p_f(f_k)$$

Texture description

- ***Variance***

$$s^2 = \sum_{k=1}^{N} (f_k - m)^2 \, p_f(f_k)$$

- ***Kurtosis***

$$\grave{\imath}_4 = \frac{1}{4} \sum_{k=1}^{N} (f_k - m)^4 \, p_f(f_k) - 3$$

- Image entropy is defined in terms of the histogram as well:

$$H = -\sum_{k=1}^{N} p_f(f_k) \ln p_f(f_k)$$

Texture description

- Spatial information can be described by using the histograms of ***grey-level differences***:
  - Let $\mathbf{d} = (d_1, d_2)$ be the displacement vector between two image pixels, and $g(\mathbf{d})$ the grey-level difference at distance $\boldsymbol{d}$:

$$\mathbf{g(d) = | f (k, l) - f (k + d_1, l + d_2) |}$$

  - We denote by $p_g(g, \mathbf{d})$ the histogram of the grey-level differences at the specific distance $\boldsymbol{d}$
  - If an image region has coarse texture, the histogram $p_g(g, \mathbf{d})$ tends to concentrate around $\mathbf{g = 0}$ for small displacements $\boldsymbol{d}$
  - If the region texture is thin, it tends to spread

Texture description

- Several texture measures can be extracted from the histogram of grey-level differences:
  - ***Mean***

$$\grave{\imath}_{\mathrm{d}} = \sum_{k=1}^{N} g_k \, p_g(g_k, \mathrm{d})$$

  - ***Variance***

$$s_d^2 = \sum_{k=1}^{N} (g_k - m_d)^2 \, p_g(g_k, d)$$

  - ***Contrast***

$$c_d = \sum_{k=1}^{N} g_k^2 \, p_g(g_k, d)$$

Texture description

- ***Entropy***

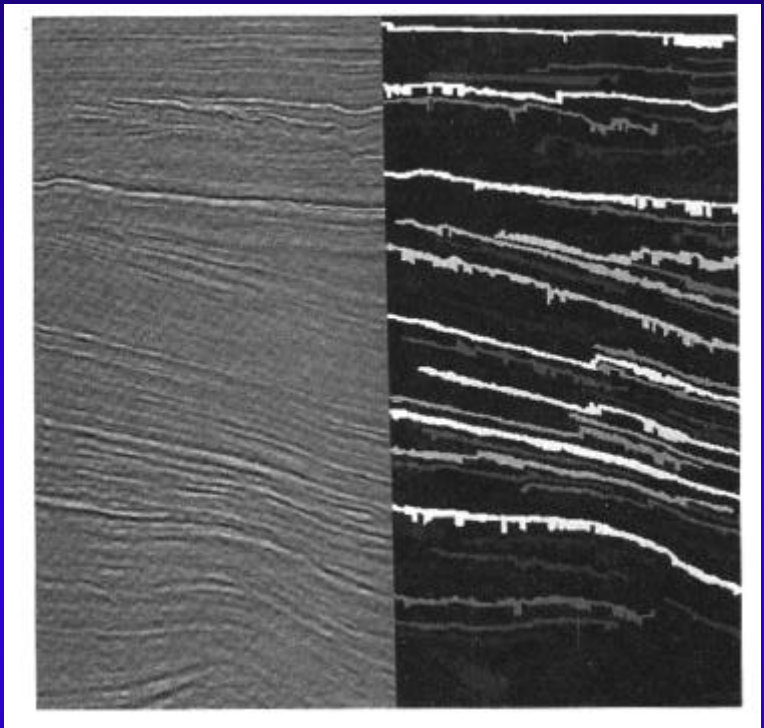
$$H_d = -\sum_{k=1}^{N} p_g(g_k, d) \ln p_g(g_k, d)$$

- Advantages: computational simplicity and capability to give information about the spatial texture organisation

Texture description

- A run length *l* of pixels having equal intensity f in a direction è is an event denoted by *(l, f, è)*
  - The run lengths reveal both directionality and coarseness of image texture
  - Coarse textures tend to produce long grey-level runs
  - Directional texture tends to produce long runs at specific directions *è*

Texture description



- *(a)* Original image
- *(b)* Binary run-length image

Texture description

- If $N(l, f, \grave{e})$ denote the number of events $(l, f, \grave{e})$ in an image having dimensions $N_1 \times N_2$ and N_R is the total number of existing runs :

$$T_R = \sum_{k=1}^{N} \sum_{l=1}^{N_R} N(l, f_k, q)$$

- The ratio $N(l, f, \grave{e}) / T_R$ is the histogram of the grey-level runs at a specific direction è
- The following texture features can be calculated from the grey-level run lengths:

Texture description

- ***Short-run emphasis***

$$A_1 = \frac{1}{T_R} \sum_{k=1}^{N} \sum_{l=1}^{N_R} \frac{1}{k^2} N(l, f_k, q)$$

- ***Long-run emphasis***

$$A_2 = \frac{1}{T_R} \sum_{k=1}^{N} \sum_{l=1}^{N_R} k^2 N(l, f_k, q)$$

- ***Grey-level distribution***

$$A_3 = \frac{1}{T_R} \sum_{k=1}^{N} \left[\sum_{l=1}^{N_R} \frac{1}{k^2} N(l, f_k, q) \right]^2$$

Texture description

- ***Run-length distribution***

$$A_4 = \frac{1}{T_R} \sum_{l=1}^{N_R} \left[\sum_{k=1}^{N} \frac{1}{k^2} N(l, f_k, q) \right]^2$$

- ***Run percentages***

$$A_5 = \frac{1}{N_1 N_2} \sum_{k=1}^{N} \sum_{l=1}^{N_R} N(l, f_k, q)$$

Texture description

- Set of texture descriptors based on the grey-level co-occurrence matrices

- If $p(f_k, f_l, \boldsymbol{d})$ denote the joint probability of two pixels
  - It is estimated from an image by counting the number n_{kl} of occurrences of the pixel values (f_k, f_l) lying at distance $\boldsymbol{d}$ in the image
- If n be the total number of any possible joint pairs the co-occurrence matrix elements $\boldsymbol{c_{kl}}$ are given by:

$$c_{kl} = \hat{p}(f_k, f_l, d) = \frac{n_{kl}}{n}$$

Texture description

- The co-occurrence matrix C_d has dimension $N \times N$, where N is the number of grey levels in the image
- Co-occurrence matrices carry very useful information about spatial texture organisation
  - If the texture is coarse, their mass tends to be concentrated around the main diagonal
  - If the texture is fine, co-occurrence matrix values are much more spread
  - If texture carries strong directional information, the co-occurrence matrices tend to have their mass in the main diagonal

Texture description

- Several texture descriptors have been proposed to characterize the cooccurrence matrix content:

  - ***Maximum probability*** $$p_d = \max_{k,l} c_{kl}$$

  - ***Entropy*** $$H_d = -\sum_{k=1}^{N}\sum_{l=1}^{N} c_{kl} \ln c_{kl}$$

  - ***Element-difference moment of order*** **m**

$$I_d = \sum_{k=1}^{N}\sum_{l=1}^{N} |k-l|^m c_{kl}$$

Texture description

- The spectral characterization of the image texture is based:
  - Either on the autocorrelation function of a two-dimensional image;
  - Or on its power spectrum

Texture description

- Autocorrelation function $R_{ff}(k, l)$ of an image $f(i,j)$:

$$R_{ff}(k,l) = \frac{1}{(2N_1+1)(2N_2+1)} \sum_{i=-N_1}^{N_1} \sum_{j=-N_2}^{N_2} f(i,j)\, f(i+k, j+l)$$

- It can be calculated both for positive and negative lags (k, l). It usually attains a maximum for zero lag *(0,0)* and drops exponentially with *(k,l)* (positive or negative)
- Direct computation of the autocorrelation function is preferred for a small number of lags (*k, l*) by using the above relation

Texture description

- If the calculation of $\mathbf{R_{ff}(k, l)}$ for a large number of lags is needed, the following property of the autocorrelation function can be very useful:

$$R_{ff}(k,l) = IDFT[F(\omega_1,\omega_2)F^{*}(\omega_1,\omega_2)]$$

- Where $F(\omega_1, \omega_2)$ is the discrete time Fourier transform of the image $f(n, m)$
  - Therefore, the autocorrelation function can be
  - calculated by employing the discrete Fourier transform:

$$F(k,l) = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f(n,m)\ \exp\left[-i\left(\frac{2\pi nk}{N}+\frac{2\pi ml}{M}\right)\right]$$

Texture description

- And the inverse DFT:

$$R_{ff}(k,l) =$$

$$\frac{1}{NM}\sum_{u=0}^{N-1}\sum_{u=0}^{M-1} F(u,u)F^{*}(u,u)\exp\left[i\left(\frac{2\eth ku}{N}+\frac{2\eth l\tilde{o}}{M}\right)\right]$$

- An estimate of the power spectrum $P_{ff}(\grave{u}_1, \grave{u}_2)$ of an image, is given by the two-dimensional periodogram:

$$P_{ff}(\grave{u}_1,\grave{u}_2) = \left|F(\grave{u}_1,\grave{u}_2)\right|^2 = F(\grave{u}_1,\grave{u}_2)F^{*}(\grave{u}_1,\grave{u}_2)$$

Texture description

- Premultiplication of the image *f(m, n)* by a two-dimensional window w(m, n) produces a relatively smooth power spectrum estimate

- Let us suppose that we use polar coordinates for power spectrum $P_{ff}(r, \ddot{o})$ description:

$$r = \sqrt{w_1^2 + w_2^2}$$

$$f = \arctan\left(\frac{w_2}{w_1}\right)$$

Texture description

- The average $P_{\ddot{o}}(\ddot{o})$ is a very good descriptor of texture directionality:

$$P_f(f) = \int_0^{r_{\max}} P_{ff}(r, f)\, dr$$

- The above integral can be approximated by a summation within a wedge $\ddot{o}_1 \pounds \ddot{o} < \ddot{o}_2$ in the spectral domain:

$$P_f(f) \approx \sum_{\sqrt{w_1^2 + w_2^2} < r_{\max},\ f_1 \le f < f_2} \left| F(w_1, w_2) \right|^2$$

Texture description

- The radial distribution of the power spectrum can be described by the integral:

$$P_r(r) = \int_0^{2p} P_{ff}(r, f)\, df$$

- This integral can be approximated by the following sum, by splitting the spectral domain into concentric areas:

$$P_r(r) \approx \sum_{r_1^2 \le \sqrt{w_1^2 + w_2^2} < r_2^2} \left| F(w_1, w_2) \right|^2, \qquad r_1 \le r < r_2$$