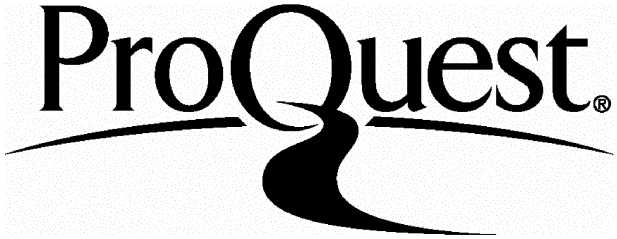
SPACE AND TIME EFFICIENT DATA STRUCTURES IN TEXTURE FEATURE EXTRACTION

Andrew E. Svolos

University College London

A Thesis Submitted to the University of London for the Degree of Doctor of Philosophy

1998



Abstract

Texture feature extraction is a fundamental stage in texture image analysis. Therefore, the reduction of its computational time and storage requirements is an important objective.

The Spatial Grey Level Dependence Method (SGLDM) is one of the most important statistical texture analysis methods, especially in medical image processing. Co-occurrence matrices are employed for its implementation. However, they are inefficient in terms of computational time and memory space requirements, due to their dependency on the number of grey levels in the entire image (grey level range). Since texture is usually measured in a small image region, a large amount of memory space is wasted while the computational time of the texture feature extraction operations is unnecessarily raised. Their inefficiency puts up barriers to the wider utilisation of the SGLDM in a real application environment, such as a clinical environment.

In this thesis, three novel approaches which are based on dynamic binary trees to organise the textural information extracted from an image region by the SGLDM, are presented and evaluated both theoretically and through their application to the analysis of natural textures and medical images of various modalities. Their novelty is based on their ability to eliminate the redundant information stored in the co-occurrence matrix, due to their dependency only on the number of distinct grey levels in the analysed local region. These are shown to provide efficiency in both storage requirements and computational time compared to the co-occurrence matrix, especially in the analysis of images employing their full dynamic range. Two experiments involving the classification of clinical and non-clinical image data are performed, which clearly show the benefit of analysing images using SGLDM, without reducing their grey level range.

Acknowledgements

I am indebted to my first supervisor, Prof. Andrew Todd-Pokropek, for his continuous support, guidance, and understanding. I am also very grateful to my second supervisor, Dr. Alfred Linney, for his support, encouragement, and especially his patience.

My special thanks to Dr. Jing Deng, Prof. Andrew Todd-Pokropek, Tryphon Lambrou, Chloe Hutton, and Dr. Joao C. Campos for providing me with the medical data and helping me with the region selection. The assistance from the prematurely deceased Dr. John Gardener in the ultrasonic analysis was extremely valuable. Especially, I would like to thank my colleague and friend, Tryphon Lambrou, for helping me with the classification experiments, Dr. Joanne Mathias for her help and advice, Chloe Hutton for her support throughout my first steps in this research, and Patricia Goodwin for her advice, encouragement, friendship, and of course, for bringing her beautiful plants, making our working place more colourful.

The research work in this thesis would not have been possible without the generous funding from The Charitable Establishment of Bakalas Bros., on the basis of the decision **No 1080844/2464/B0011/16-12-1994** of the Hellenic Secretary of State for Finance.

I dedicate this work to my beloved father Ilias, my beloved mother Eleni, and my beloved sister Evanthia. Without your support it would have been impossible to carry on. I thank you from the bottom of my heart.

"Either make the tree good, and its fruit good; or make the tree bad, and its fruit bad; for the tree is known by its fruit."

Matthew: 12.33 (RSV)

To the everlasting Light that gives light to my life.

To the Mother of the universe that lives on the top of the high mountain.

Contents

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Relative percentage reduction in memory space for the plain co-occurrence trees

Relative percentage reduction in memory space for the enhanced co-occurrence trees (static rule)

Relative percentage reduction in memory space for the enhanced co-occurrence trees (move to front rule & counter rule)

Relative percentage reduction in memory space for the self-adjusting co-occurrence trees

The analysed image types

Computational time results for the fur images

Computational time results for the water images

Computational time results for the weave images

Computational time results for the normal mammograms

Computational time results for the ultrasound heart images

Computational time results for the ultrasound kidney images

Computational time results for the ultrasound liver images

Computational time results for the ultrasound ovaries images

Computational time results for the ultrasound spleen images

The analysed image types cont.

Computational time results for the MR bone images

Computational time results for the 12 bit MR brain images

Computational time results for the 8 bit MR brain images

Computational time results for the 12 bit MR muscle images

Computational time results for the 15 bit MR muscle images

The analysed image types cont.

Computational time results for the CT liver images (32 × 32 region size)

Computational time results for the CT liver images (16 × 16 region size)

Computational time results for the CT brain images (16 × 16 region size)

Computational time results for the CT brain images (32×32 region size)

Computational time results for the CT lung images

The effect of dynamic range reduction on the textural features cont.

List of Tables

Chapter 7

Chapter 9

Back Pocket

PUBLISHED PAPERS

1. A. E. Svolos, C. A. Hutton, and A. Todd-Pokropek. Co-occurrence trees: A dynamic solution for texture feature extraction. In *Proc. IEEE EMBS 18th Int. Conf. on Engineering in Medicine and Biology*, pages 1142–1144, Amsterdam, The Netherlands, 1996.

2. A. E. Svolos and A. Todd-Pokropek. An evaluation of a number of techniques for decreasing the computational complexity of texture feature extraction through an application to ultrasonic image analysis. In *Proc. IEEE EMBS 19th Int. Conf. on Engineering in Medicine and Biology*, pages 601–604, Chicago, IL, 1997.

3. A. E. Svolos and A. Todd-Pokropek. Self-adjusting binary search trees: An investigation of their space and time efficiency in texture analysis of magnetic resonance images using the spatial grey level dependence method. In *Proc. SPIE Medical Imaging Int. Conf.: Image Processing*, pages 220–231, San Diego, CA, 1998.

4. A. E. Svolos and A. Todd-Pokropek. Time and space results of dynamic texture feature extraction in MR and CT image analysis. *IEEE Trans. on Information Technology in Biomedicine*, ITB-2(2):48–54, 1998.

Chapter 1

Introduction

1.1 Texture – An essential image feature

Texture is a fundamental feature that can be used in the analysis of images in several ways, e.g. in the classification of medical images into normal and abnormal tissue, in the segmentation of scenes into distinct objects and regions, and in the estimation of the three-dimensional orientation of a surface. Texture has long been recognised as a key to human perception. Therefore, it has been extensively employed in computer vision tasks. Its importance has been proved for many types of images ranging from multi-spectral remote sensed aerial and satellite data to biomedical images. In particular, in medical image analysis texture has been successfully employed for tissue characterisation, i.e. the determination of tissue type (normal or pathological) and further classification of tissue pathology (see [97]), in many cases. Texture analysis has been shown to increase the level of diagnostic information extracted from various image types of most medical imaging modalities and to quantitatively characterise differences in tissue imperceptible by human observers.

There is no formal mathematical definition that includes all members of the very diverse texture family. This is due to the wide variability of texture. According to [63], [65], texture is a property that characterises almost all surfaces, e.g. the weave of a fabric and the pattern of crops in a field. It contains important information about the way the surfaces are structurally arranged. The authors in

the above references distinguish between tone and texture in grey level images. Tone is equivalent to grey level assigned to each pixel in an image.

Texture can be considered to be decomposed into two constituents, namely the tonal primitives and the placement rules. A tonal primitive is a maximally connected set of pixels (a region) having a given tonal property, such as a range of grey levels. This region can be described in terms of its area and shape. Tonal primitives can be geometrical objects, such as line segments or closed polygons. On the other hand, the primitive placement rules describe the spatial dependence among the primitives in the texture. Therefore, an image texture can be described by the number and types of its primitives as well as the spatial organisation of its primitives determined by the placement rules. This organisation may be random or determined by the spatial dependence of one primitive on a neighbouring primitive (second order), or, in general, on $n-1$ neighbouring primitives (nth order). Another way to define texture is by considering it as a global pattern resulting from the repetition, either deterministically or randomly, of local sub-patterns.

One important aspect of texture description is its scale dependence, according to [144]. That means that texture primitives can be defined in more than one hierarchical level. At different scales different textures may arise, coarser or finer. Coarseness is one of the properties that qualitatively describe texture. Other such properties are smoothness, granulation, randomness, etc. According to [61], the two major properties of a texture are its coarseness and directionality. Each of these properties translates into some property of the tonal primitives and the spatial dependence among them. For example, a fine texture results if the tonal primitives are small and there is a large variation between neighbouring primitives. On the other hand, if the tonal primitives are larger (they consist of a larger number of pixels) a coarse texture arises. In any case, the fine/coarse texture property depends on scale.

Textures can also be classified into two categories according to their strength, i.e. weak and strong textures. No clear spatial relationship can be found among the texture primitives in a weak texture. Therefore, weak textures can be adequately described by the frequency of occurrence of texture primitive types in some region. Strong textures are characterised by non-random spatial relation-

ships and are usually sufficiently described by means of the frequency of co-occurrence of pairs of texture primitives in some spatial relationship. Therefore, both primitives and primitive placement rules are necessary for the characterisation of such textures.

The main problem in texture analysis is that real textures are extremely difficult to define mathematically and therefore, get analysed by machines, although it is a quite easy task for human beings to recognise and describe them in empirical terms. It is this lack of definition that makes automatic description or recognition of textures a very complex and as yet an unsolved problem.

1.2 Texture analysis

According to [61], two major texture analysis methods exist: statistical and syntactic or structural. Statistical methods employ scalar measurements (features) computed from the image data, which characterise the analysed texture. In this context, texture is described by a set of statistics extracted from a large ensemble of local picture properties. First order statistics, such as the grey level mean and the grey level variance, have been employed in the classification of a limited set of textures. Second order statistics, such as those computed in the SGLDM, have been widely used in texture analysis. Higher order statistics can also be measured. The features extracted by the statistical methods usually measure textural characteristics, such as coarseness, contrast, and directionality.

Syntactic or structural methods describe texture by means of tonal primitive descriptions and primitive placement rules. Pure structural methods are based on the assumption that texture is made up of primitives, which appear in near regular repetitive spatial arrangements. In this case, texture analysis requires isolation of the texture primitives and description of the primitives and the primitive placement rules. Besides these methods, other texture analysis methods, which are referred to as structural-statistical in [63], belong to this category. They are structural in the sense that primitives must be explicitly defined. They are statistical in that the spatial dependence between primitives, or lack of it, is characterised using probabilities.

A primitive is characterised by a list of attributes, such as the average grey level, the area of the primitive region, and its shape. Primitive spatial relationships are usually described by means of a grammar which represents the rules for building a texture from a set of primitives. Within this context, there is a one to one correspondence between textures and formal languages. Symbols representing the primitives comprise the alphabet. Transformation rules that represent the spatial relationships between primitives, apply to the symbols of the alphabet to generate the texture. A tree grammar is commonly used to represent these placement rules. This texture description method is best for regularly structured textures. Real textures, though, are most often irregular, in the sense that structural errors, distortions, or even structural variations, frequently happen in them. Thus, to make syntactic description of real textures feasible, variable rules must be incorporated into the description grammars and non-deterministic or stochastic grammars must be employed (see [55]).

Several ways have been devised for the extraction of primitive descriptions and placement rules from a given texture. In [78], [107], the traditional Fourier spectrum of an image was employed for this purpose. In [161], a texture primitive was assumed to be a region of homogeneous grey level. The problem of extracting primitives from a given texture was therefore equivalent to the problem of segmenting the image into homogeneous regions. The placement rules were computed from the relative positions of the centroids of the extracted primitives. In [171], the structural texture description algorithm first computed the edge map of the given texture and afterwards, it extracted the texture primitives grouped by type, employing this map. Once the primitives were isolated, it employed the relative positions of the centroids of nearest-neighbour primitives to produce a set of placement rules.

1.3 Texture classification and segmentation

Automatic texture classification is an important problem in image processing. In this problem, an image region is known to contain data from one of a finite number of texture classes. The goal is to assign the image region (sample) to the correct

class (sample labelling). For example, in medical image processing the classes may be normal and abnormal breast tissue and the goal, the determination of whether a given digital X-ray breast image (mammogram) belongs to the first or the second class. There are three general methods for solving the above problem:

1. statistical classification,

2. syntactic or structural classification,

3. neural network classification.

Syntactic classification is based on some sort of description of the structure of the texture in each image region (see Section **1**.2). According to the actual type of structural description employed, a different classification approach is followed. In [171], a symbolic representation was constructed from each texture sample and a decision tree utilised this information to classify the samples. In [161], the properties of the primitives and primitive placement rules of each texture region were computed and compared with the corresponding properties of models describing texture classes. Each texture sample was assigned to the class the model of which was closest to the properties of the sample. In [55], the unknown texture was represented by a string of symbols and a method known as parsing (see [133]) was used to determine the grammar that could produce this string and consequently its texture class. Syntactic description is suitable to the analysis and classification of textures exhibiting a more regular structure. For this reason, syntactic texture description methods have seen limited use in real texture analysis applications, especially in medical image processing. Therefore, we will not present them in more detail.

Neural network classification is based on artificial neural networks, which are relatively new computing systems whose architecture is made of a large number of interconnected simple processing elements called nodes. The input of each node is the weighted sum of the output of all the nodes to which it is connected. Its output value is, in general, a non-linear function of this sum. This function is referred to as the activation function of the nodes.

Artificial neural networks emulate the biological computational mechanism. Biological systems perform computations via interconnections of biological cells

called neurons. The basis of these computations is a small number of serial steps, each of which is massively parallel employing a vast number of simple processing cells which are locally connected. They allow the real time processing of complex information. Neural networks are, in a sense, a non-algorithmic, black box approach. This black box is trained in order to learn the correct response (e.g. correct classification) for each of the training samples. From this point, it is an attractive approach, since it minimises the required amount of *a priori* knowledge of the properties of the data and detailed knowledge of the internal system operation. After training, it is hoped that the internal structure has been self-organised to enable the correct response for new, yet similar, data, based on the "experience" gained from the training set.

One of the most popular artificial neural networks is the back propagation neural network. Its training consists in finding a mapping between the set of input values and the set of output values. This mapping is accomplished by adjusting the weights using a learning algorithm, such as the generalised delta rule (see [133]). After the weights are adjusted on the training set, their values become fixed and the neural network can be employed for the classification of unknown samples.

A number of attempts have been made for the utilisation of this new emerging tool in texture analysis and classification. In [77], a neural network texture classification was proposed based on the Gabor filters (see Section **2**.7). In [90], a back propagation neural network was employed for texture classification based on wavelet packets (see Section **2**.7). Augusteijn and Clemens ([5]) used a neural network architecture for the classification of 9 natural textures from the Brodatz album ([17]). In [6], a neural network classifier was employed for ground cover identification from satellite images. Similarly, in [95] a neural network architecture was used for the classification of satellite images into various cloud types. Neural networks have been used in medical image analysis, as well. In [109], a neural network was used for the segmentation of digital chest radiographs (see Section **3**.2). In [82], a number of neural networks were evaluated in the classification of ultrasound liver images (see Section **3**.3). In [45], [129], a neural network classifier was employed for the classification of mammograms (see Section **3**.2).

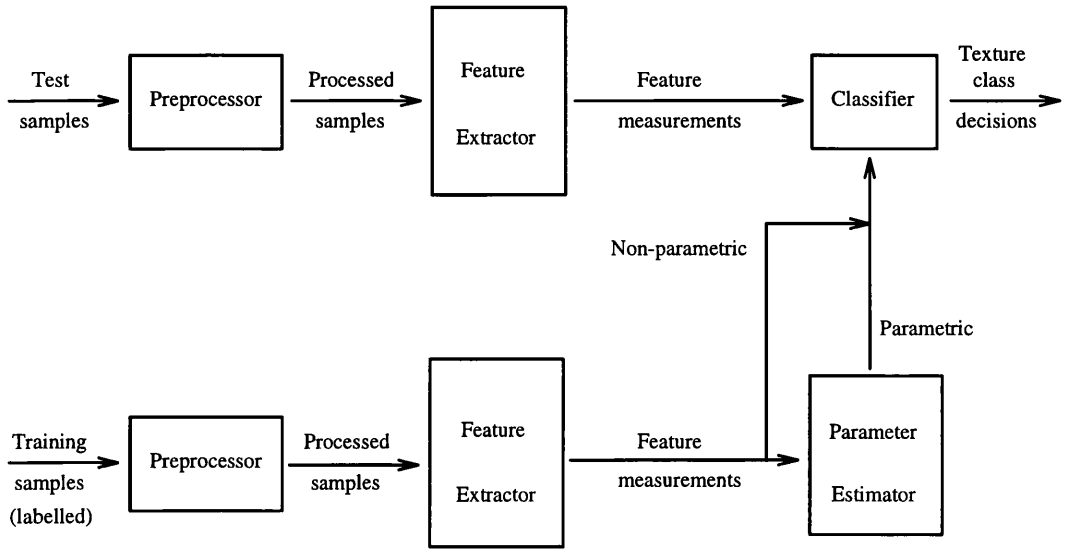


Figure 1.1: A typical statistical texture classification system.

The above studies showed that the neural network approach is a potential immediate competitor against statistical classification, but further evaluation of its performance needs to be carried out, especially in medical image analysis.

Statistical classification is based on characteristic measurements (features) extracted from the analysed textural region. Usually, these scalar measurements are organised in a vector form (feature vector), which characterises the analysed texture. A block diagram outlining a typical statistical texture classification system, modified from [170], is shown in Figure 1.1. Actually, the illustrated system is a supervised classification system. This is the most common system in texture classification.

There are two broad categories of classification strategies: supervised and unsupervised. In supervised classification, the texture classes are known *a priori* and labelled samples for each class are available (training set). Labelled samples are those whose texture class is known in advance. This set of samples is necessary to adapt the classifier to each of the known texture classes, before the set of the test samples is fed into the classifier. The texture class of the test samples is unknown and the classifier is employed to determine the class of each one of them. In the unsupervised classification there are no labelled samples. That is, the training set consists of samples whose class is unknown. Given this set, the classifier tries to find natural partitions of the sample data. As one would expect,

the problem of unsupervised classification is much more difficult.

The classifiers are further categorised into parametric and non-parametric. In parametric classification, the parameters of the class-conditional sample density functions must first be determined from the training data, given the form of these functions (e.g. Gaussian or uniform). In practice, we may not be able to determine a specific form. In this case, non-parametric classification must be employed, where the training samples are directly used for the classification. The performance of the classifiers is usually determined by the percentage of samples correctly classified (classification accuracy).

One of the most interesting supervised non-parametric classifiers is the k-nearest neighbour classifier. This is a simple classifier, which is widely used, especially when the underlying class-conditional probability distributions are unknown. It classifies a given test sample by assigning it the label that most frequently encountered among the k nearest to it training samples (see [48]). The Euclidean metric ([133]) is commonly used for the computation of the distance between the test sample and each training sample. The *leave-one-out* method ([56]) is most often used for the estimation of the classification accuracy of this classifier for a given classification problem. According to this method, one sample from the set of labelled samples is excluded, the classifier is trained on the remaining samples (training set), and subsequently, the excluded sample is tested using the classifier (test sample). This operation is repeated for each sample in the set of labelled samples. Then, the number of correctly classified samples is counted in order to estimate the classification accuracy. According to [29], [56], [65], the *leave-one-out* method is widely used, when the number of available samples for each class is relatively small.

In Figure **1.1**, the feature extraction stage extracts the scalar measurements from the texture samples based on a texture analysis method, such as the Spatial Grey Level Dependence Method (SGLDM). These measurements, or else textural features, are considered as a condensed representation of the analysed texture, ideally containing all important information for its characterisation. The computational time as well as the memory requirements of this stage can be very high and most often, much higher than the corresponding requirements of the other

constituent parts of the classification system, significantly increasing the time and memory space of the whole texture classification process. Reducing the dynamic range of the analysed textures is one way to keep these requirements tolerable. However, as we will see in the next chapters, this reduction may lead to significant information loss which can be vital, especially in medical image analysis. Therefore, efficient algorithms, in terms of computational time and memory space, must be devised for the feature extraction stage. This is also justified by the fact that this stage is an essential part, not only of a texture classification system, but also of a texture segmentation system and in general, of any texture analysis system.

Developing time and space efficient algorithms will help not only to speed up the whole process, but also to minimise grey level reduction, possibly allowing for more useful information to flow into the feature extractor, more descriptive texture features to be computed and consequently, better classification accuracy to be achieved. In this thesis, a number of algorithms for the feature extraction stage, which show efficiency in both time and space, are proposed and evaluated for one of the best and most widely employed texture analysis methods, namely the Spatial Grey Level Dependence Method (SGLDM).

Finally, we will say a few words about the texture segmentation problem. This problem involves the accurate partitioning of an image into differently textured regions (textures). Alternatively, texture segmentation can be viewed as the problem of accurately delineating the borders between different textures in an image. Two types of methodologies are widely employed to solve the general image segmentation problem ([26]). The first one is the region growing methodology, which starts from small regions with certain properties and expands them as long as their properties are not violated. The other one is the edge detection, which searches for parts of the image where a transition occurs from one region to another region having different properties. The interested reader should refer to one of the many surveys of texture segmentation techniques ([61], [74], [126], [173]).

In particular, texture segmentation is a problem of great importance in medical image analysis. Much work has been carried out in this field. In [89], texture was employed for the segmentation of digital dental radiographs into bone and

teeth (see Section **3**.2). In [109], digital chest radiographs were segmented into various regions, such as the heart and the lungs, based on texture features extracted from local neighbourhoods around each pixel (see Section **3**.2). Fortin et al. ([53]) attempted to segment Magnetic Resonance (MR) cardiac images into the ventricular walls and the ventricular blood pools (see Section **3**.4). In [99], MR brain images were segmented into white matter, grey matter, and CSF (CereboSpinal Fluid) (see Section **3**.4). The same segmentation (white and grey matter only) was attempted in [19], analysing Computed Tomography (CT) images of the brain (see Section **3**.5). Finally, in [22] electron micrograph images of cells were segmented into the various parts of the nucleus, employing a number of texture analysis methods (see Section **3**.6).

1.4 Equal probability quantising

Texture samples usually pass through a pre-processing stage (the first stage in the texture classification pipeline in Figure **1**.1) before feature extraction. This stage performs functions such as normalisation, enhancement, or noise cleaning on the image. However, the pre-processor in texture analysis serves two main purposes. First, it normalises the data to eliminate undesirable effects, such as variations in illumination. Second, it requantises the data in a known fashion.

The normalisation usually consists in modifying the grey levels of the original image data, so that the resulting image histogram is uniformly shaped (histogram equalisation). In this way, all first order statistical information, such as sample mean and sample variance, is lost. The justification for performing this step is that first order statistics can be directly affected by factors such as variations in illumination and thus, histogram equalisation (normalisation) serves to remove these undesirable effects. Also, research in visual perception indicated that texture is characterised by higher order spatial statistics ([57], [80], [81]).

Additionally, it might be necessary to requantise the data in order to make them suitable for processing by the feature extraction stage. Frequently, a grey level reduction is performed for the purpose of reducing the computational cost of this stage. This reduction is necessary in most cases when using the SGLDM,

due to the large time and space requirements of the co-occurrence matrix. Grey level reduction can also be justified when the initial image contains noise which may adversely affect texture analysis, or redundant information (redundancy here refers to any information not necessary to achieve a certain level of texture discrimination).

In the experiments performed in this thesis, equal probability quantisation was employed for normalising the histograms of the analysed textures. The basic idea behind the "equal probability quantizing" algorithm is the step by step approximation of the cumulative probability function of a uniform distribution, based on the estimated cumulative probability function of the original texture. A detailed description and analysis of this algorithm can be found in [65].

Equal probability quantising has been used for the normalisation of various texture types by many researchers ([61]). In [101], it was employed for pre-processing natural textures, prior to texture analysis. Shanmugan et al. ([136]) employed equal probability quantising for normalising radar images illustrating various geological formations. Similarly, in [58] and [176] it was employed for flattening the histogram of satellite images showing geological terrain types. In [64], equal probability quantising was employed for the normalisation of photomicrograph images of various sandstone types.

This algorithm has also been used for the normalisation of medical images. According to [32], differences in exposure time and processing conditions of radiographic films and subsequently, differences in scanner settings used in creating digital images may significantly affect the contrast of digital images. The differences in contrast may directly affect the texture measurements for texture analysis algorithms such as the SGLDM. However, medical diagnosis should be independent of differences induced by the above factors. The authors claimed that equal probability quantising is the best technique for pre-processing radiographic images in order to normalise image contrast, especially when SGLDM is employed for texture analysis. They also claimed that equal probability quantising is a near optimal way to reduce the grey level range in an image, because in this way the pre-processed image retains as much of the original texture information as possible, much more than with other grey level reduction methods, such as linear

reduction. However, as far as we know, there is no study which clearly shows that this algorithm is the best pre-processing technique for all medical image types. Additionally, the normalisation (histogram equalisation) of medical images prior to texture analysis may not be appropriate in all cases.

We will present two examples that show the effectiveness of the "equal probability quantizing" algorithm. In the first example, a natural texture image illustrating asphalt was normalised. The image had a size of 512×512 and its initial dynamic range was 8 bits (256 grey levels). From this image, a region of size 64×64 was extracted and normalised in 256, 64, 32, and 16 grey levels. In Figures **1**.2 (a)–(e), the unequalised image region and the equalised image regions for the above dynamic ranges are illustrated. In addition, in Figures **1**.3 (a)–(e) the corresponding histograms are shown. These figures clearly show the equalisation effect of the "equal probability quantizing" algorithm, which becomes stronger as the number of grey levels becomes smaller, since the uniform distribution is better approximated. Finally, in Figure **1**.4 a 3-D illustration of the co-occurrence matrix (2-D histogram) for $(0, 1)$ displacement vector (see Section **4**.1) is shown before normalisation and after normalisation for the initial dynamic range. From these figures, it is clear that the grey level pairs are more spread out in the normalised region, as a result of the uniform distribution approximation in first order statistics (1-D histogram).

In the second example, normalisation of an ultrasound image illustrating kidney parenchyma was performed. The image had a size of 187×170 and a dynamic range of 8 bits. An image region of size 30×30 was extracted and normalised in 256, 64, 32, and 16 grey levels. Figures **1**.5 (a)–(e) illustrate the unequalised and equalised image region of the ultrasound kidney image. Figures **1**.6 (a)–(e) illustrate the corresponding histograms. Finally, Figures **1**.7 (a) and (b) show the 3-D illustration of the co-occurrence matrix of the unequalised and the equalised region, respectively, for $(0, 1)$ displacement vector and the initial dynamic range. We note that the "equal probability quantizing" algorithm had similar effects as in the case of the asphalt image.

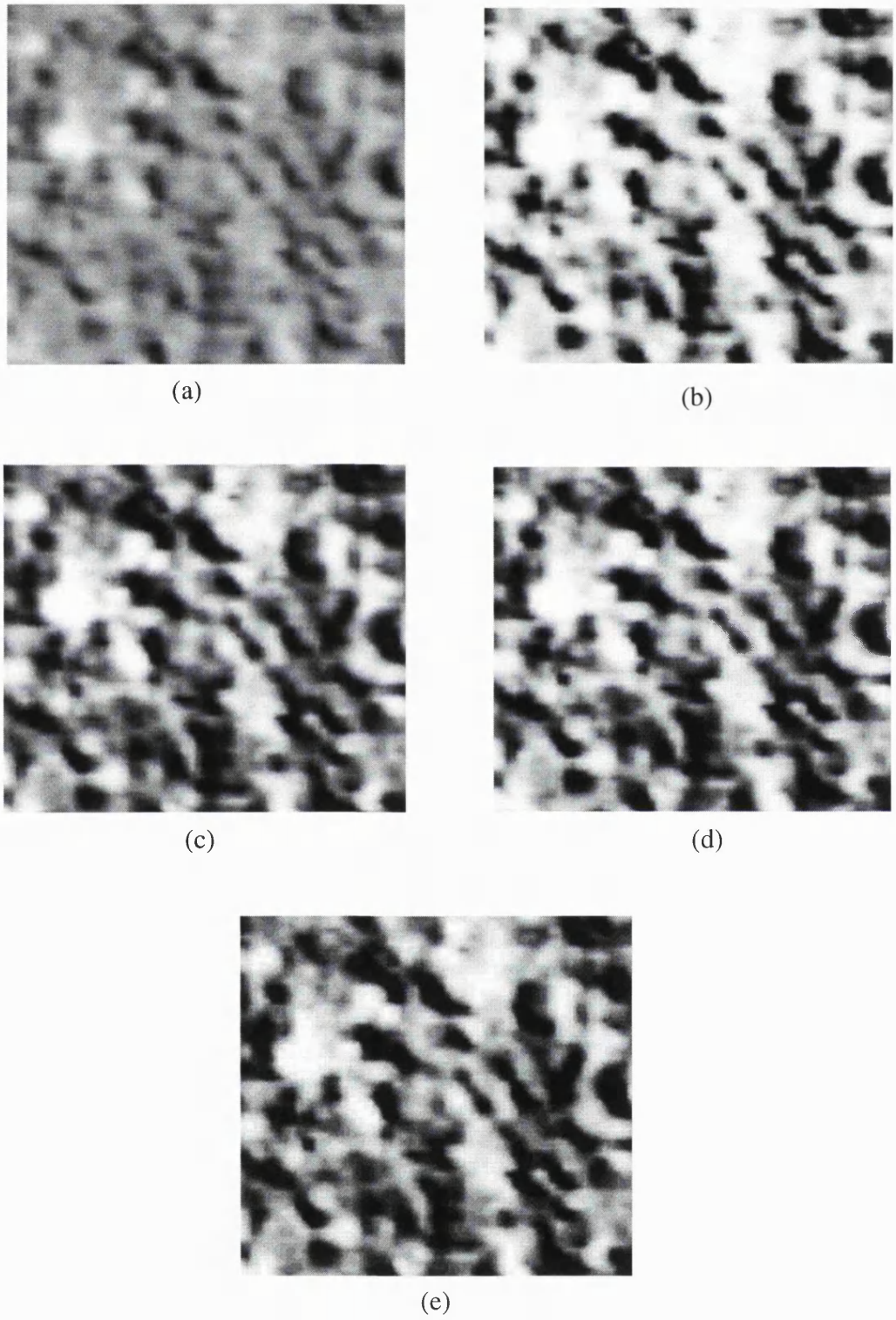
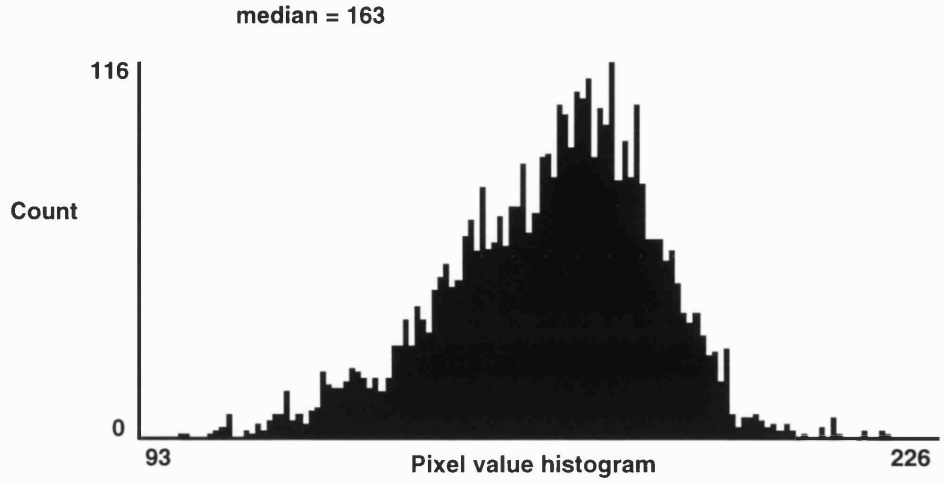
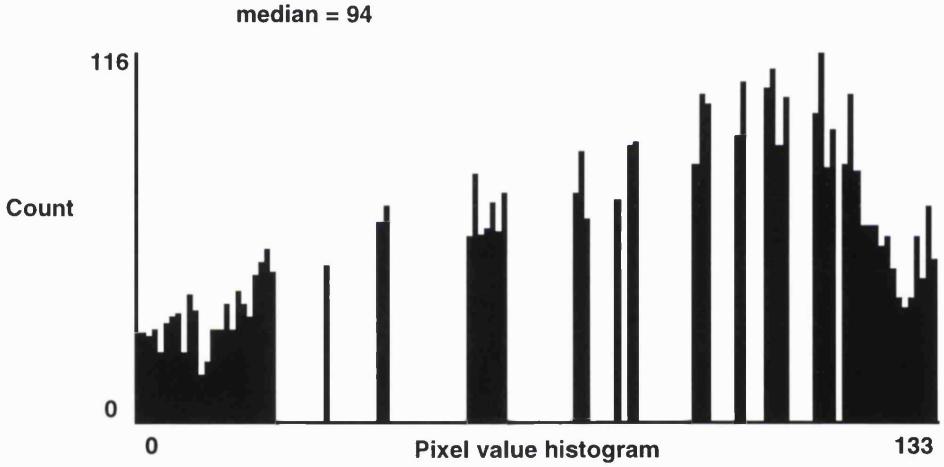


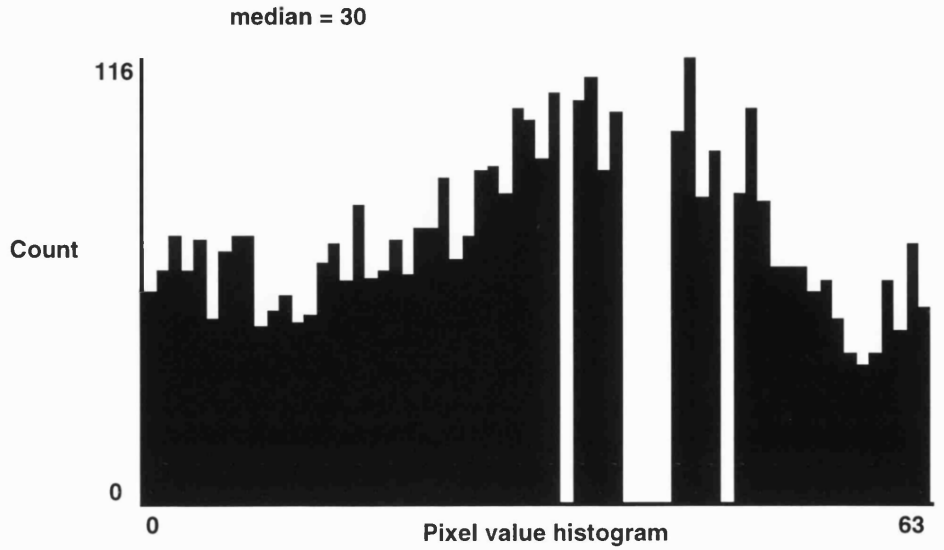
Figure **1**.2: Illustration of an asphalt image region (a) unequalised and equalised in (b) 256, (c) 64, (d) 32, and (e) 16 grey levels using the "equal probability quantizing" algorithm.


median = 163
116
Count
0
93
Pixel value histogram
226

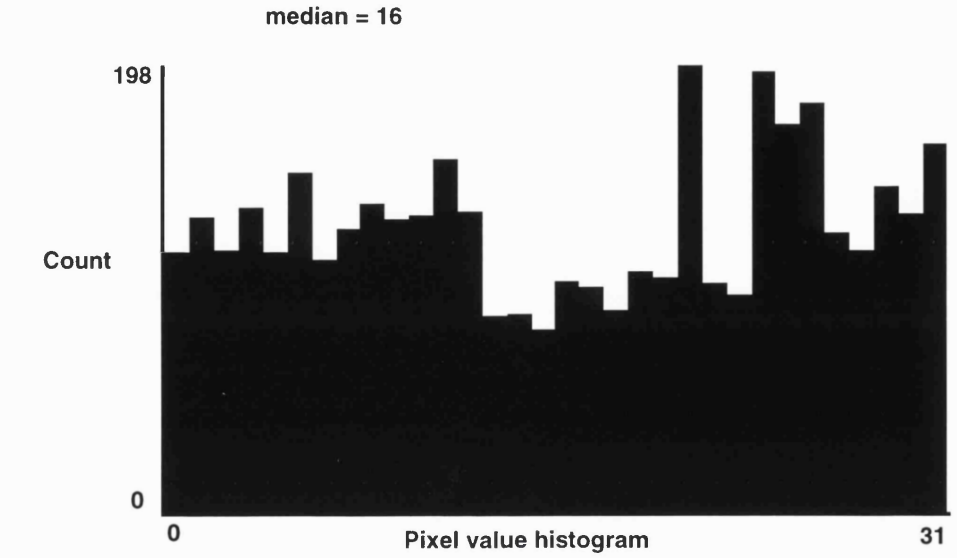
(a)


median = 94
116
Count
0
0
Pixel value histogram
133

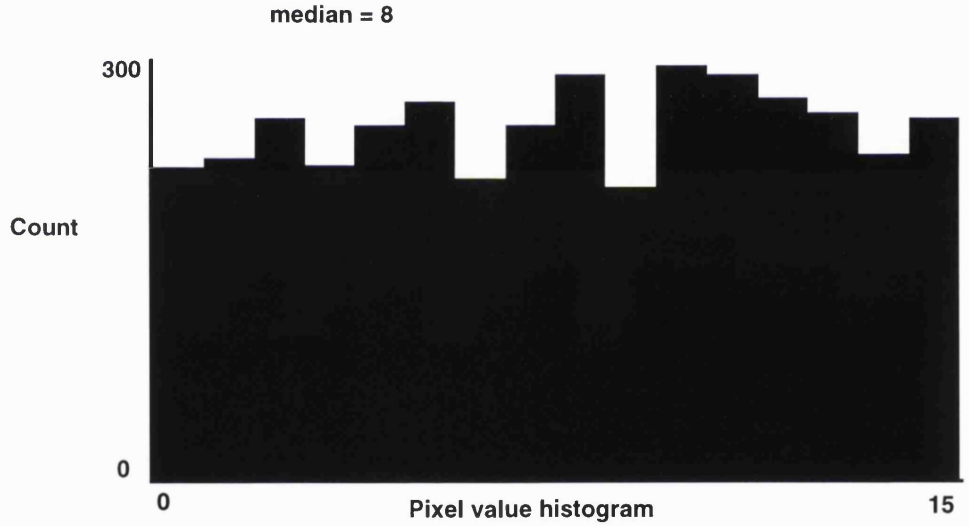
(b)


median = 30
116
Count
0
0
Pixel value histogram
63

(c)

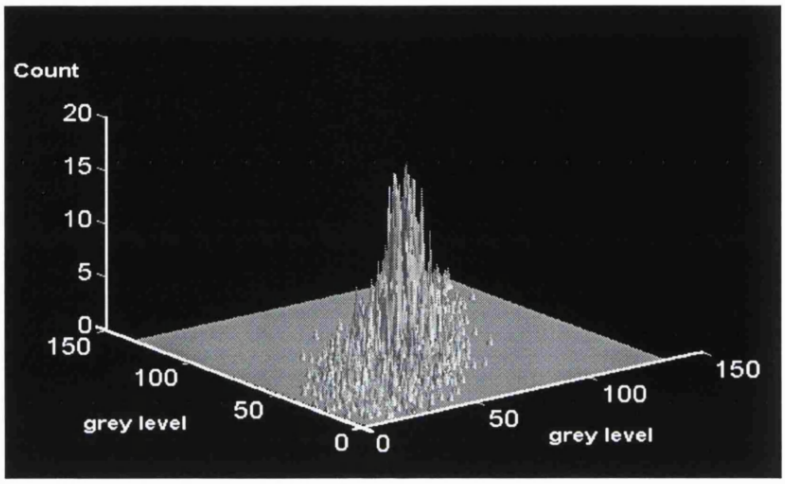


(d)

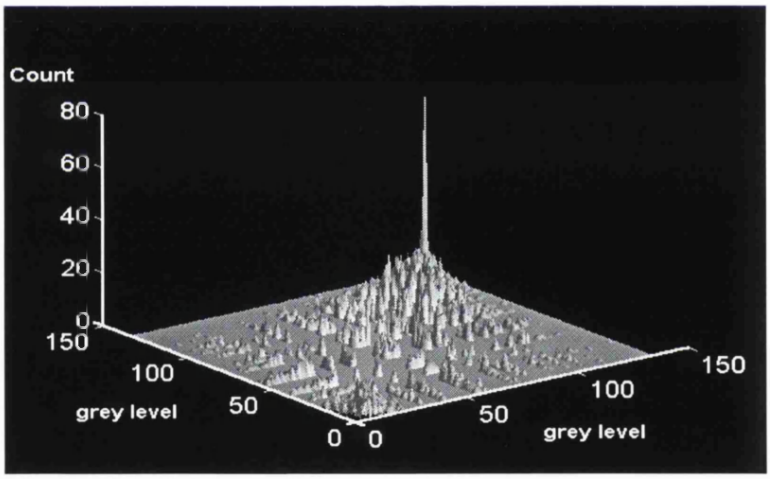


(e)

Figure **1**.3: The corresponding histograms of the images illustrated in Figure **1**.2.



(a)



(b)

Figure **1**.4: 3-D illustration of the co-occurrence matrix of the images in Figures **1**.2 (a) and (b) for (0,1) displacement vector.

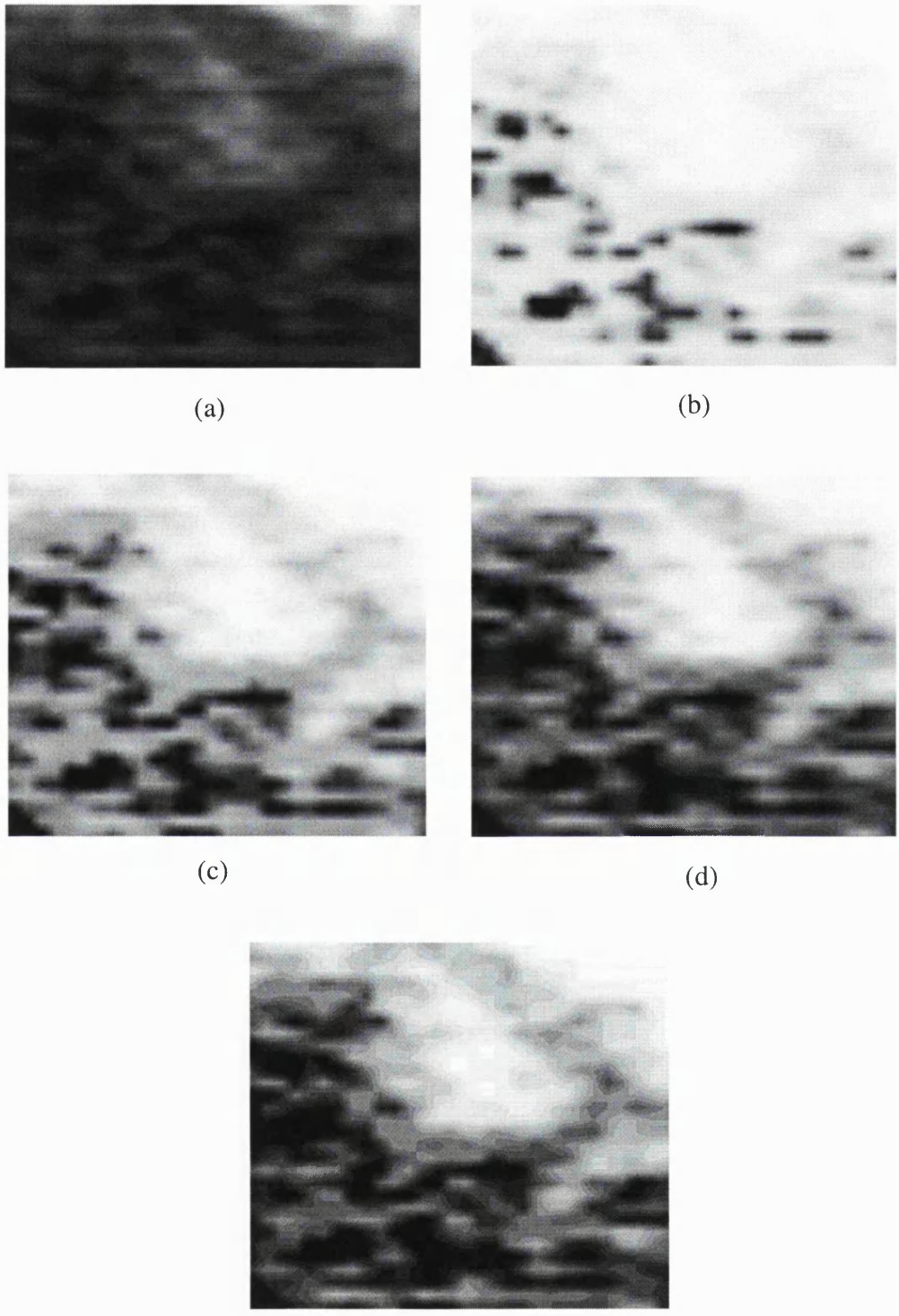
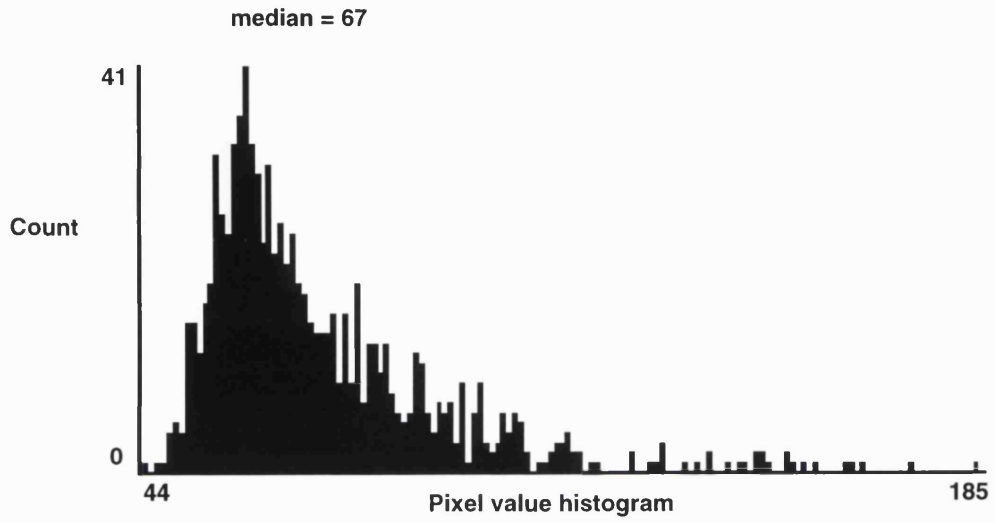
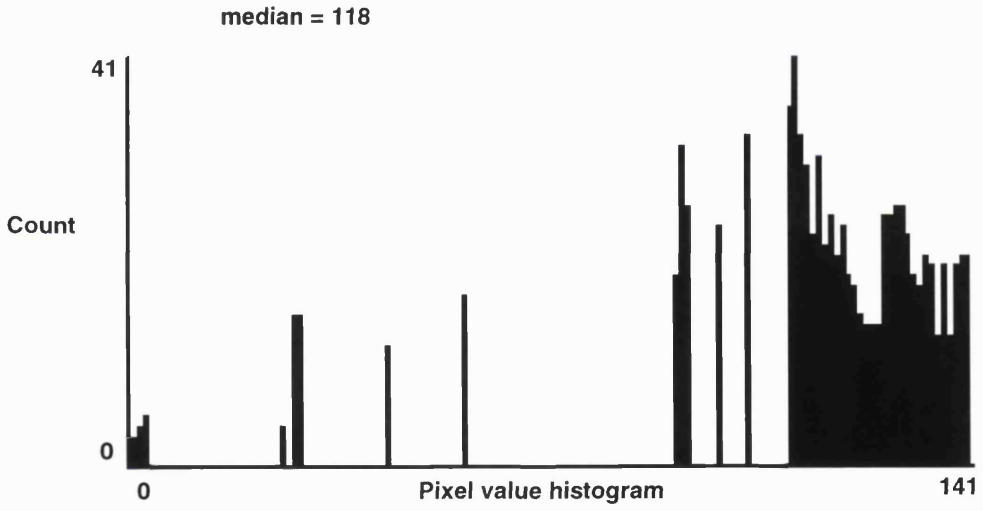


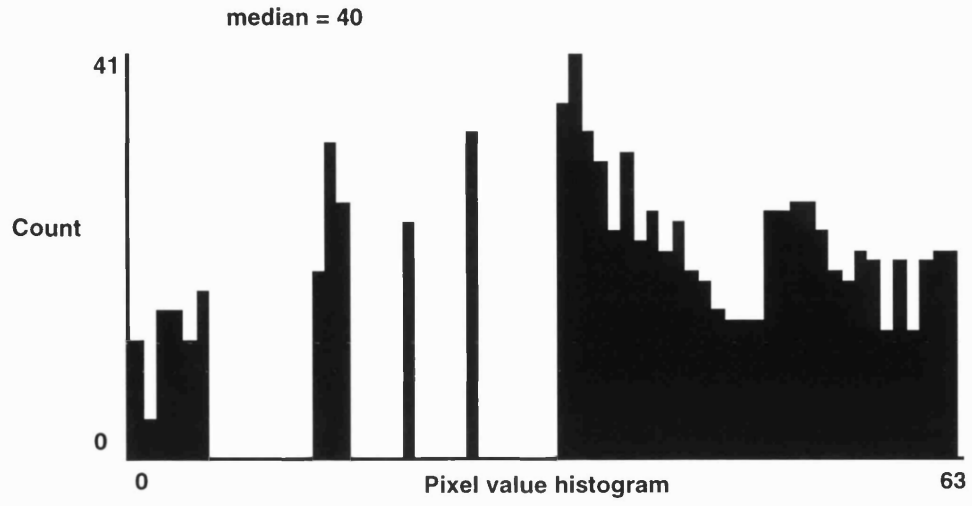
Figure 1.5: Illustration of an ultrasound kidney image region (a) unequalised and equalised in (b) 256, (c) 64, (d) 32, and (e) 16 grey levels using the "equal probability quantizing" algorithm.


median = 67
41
Count
0
44
Pixel value histogram
185

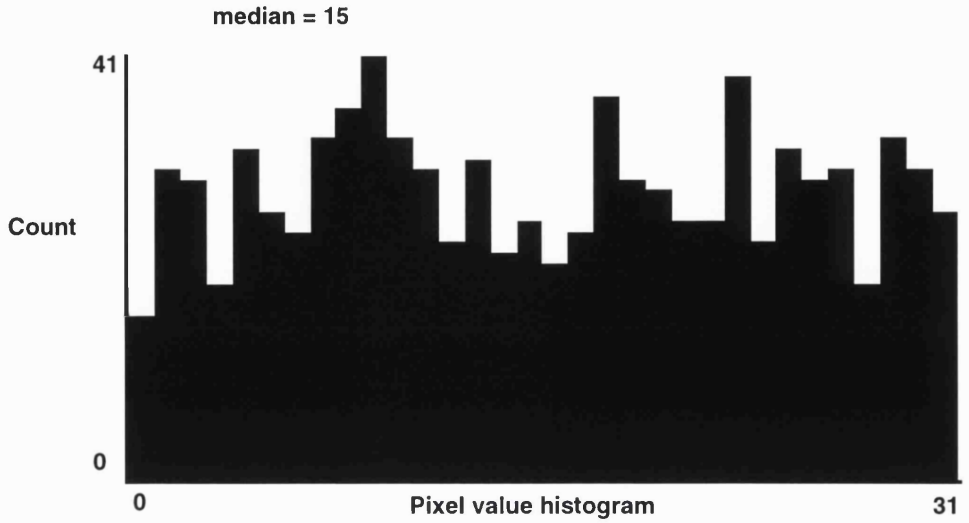
(a)


median = 118
41
Count
0
0
Pixel value histogram
141

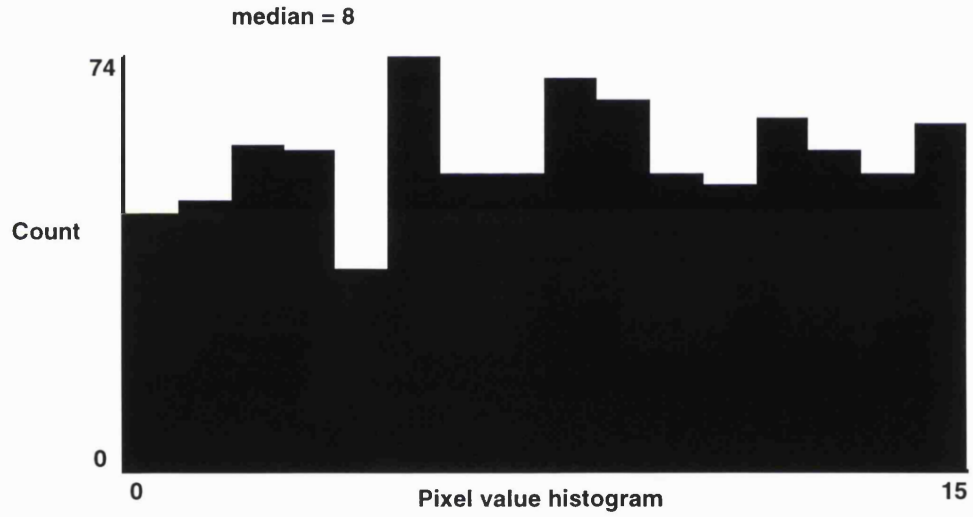
(b)


median = 40
41
Count
0
0
Pixel value histogram
63

(c)

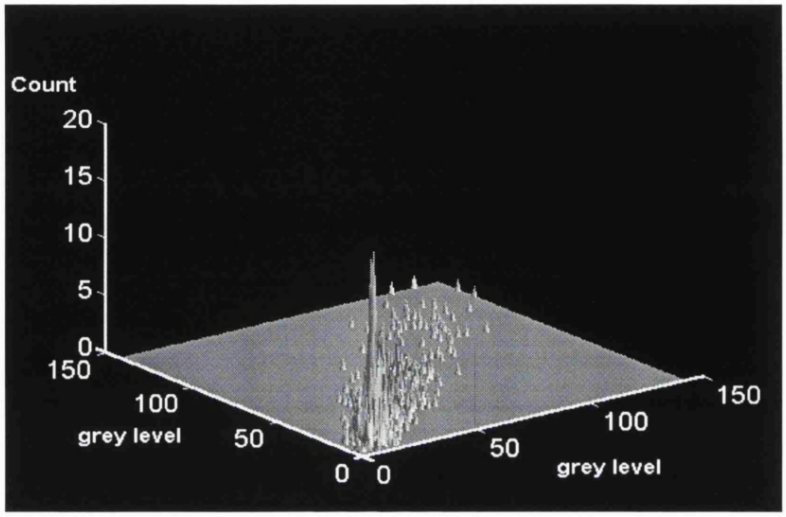


(d)

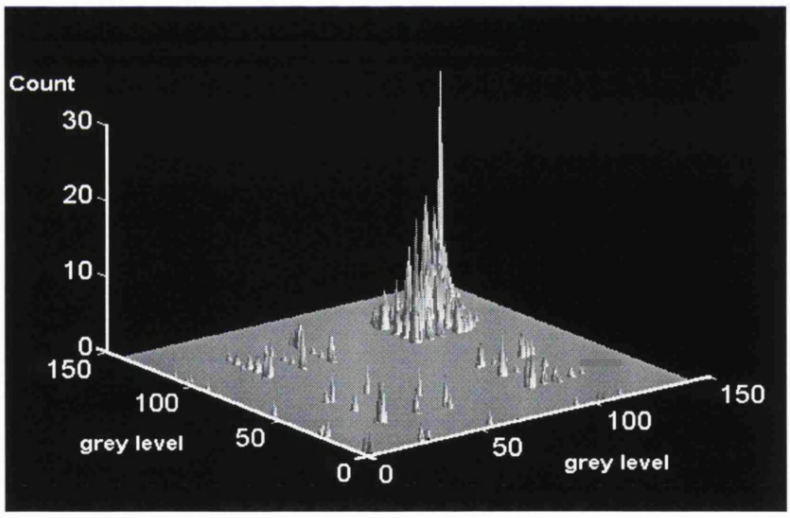


(e)

Figure 1.6: The corresponding histograms of the images illustrated in Figure 1.5.



(a)



(b)

Figure **1**.7: 3-D illustration of the co-occurrence matrix of the images in Figures **1**.5 (a) and (b) for (0,1) displacement vector.

1.5 Contributions and general organisation of the thesis

In this thesis, three novel approaches which are based on dynamic binary trees to organise the textural information extracted from an image region by the SGLDM, are presented and evaluated both theoretically and through their application to the analysis of natural textures and medical images of various modalities. Their novelty is based on their ability to eliminate the redundant information stored in the co-occurrence matrix in the form of zero entries, due to their dependency only on the number of distinct grey levels in the analysed local region. The zero estimated co-occurrence probabilities contribute nothing to the computation of the textural features. The proposed approaches are shown to provide efficiency in both storage requirements and computational time compared to the co-occurrence matrix, especially in the analysis of images employing their full dynamic range. Two experiments involving the classification of clinical and non-clinical image data are performed, which clearly show the benefit of analysing images using SGLDM, without reducing their grey level range.

The rest of the thesis is organised as follows. In Chapter **2**, a review of the most significant statistical texture analysis methods is presented. In Chapter **3**, the most widely employed statistical methods in medical image analysis are surveyed. The Spatial Grey Level Dependence Method (SGLDM) is presented in Chapter **4** along with the drawbacks of the co-occurrence matrix approach to implementing SGLDM. The dynamic data structures which form the core of the proposed approaches, are presented in Chapter **5**, whereas the proposed approaches (co-occurrence trees) along with a theoretical evaluation of their efficiency, are presented in Chapters **6**, **7**, and **8**. Experimental results from the analysis of real image data (natural and medical) are presented in Chapter **9**, which corroborate the theoretical results of Chapters **6**, **7**, and **8**. The experimental results are discussed in Chapter **10**. Finally, the conclusions of this study are presented in Chapter **11**. The future work is outlined in this chapter as well.

Chapter 2

Texture analysis methods

In the last 25 years, a large number of texture analysis methods have been proposed in numerous papers. A detailed presentation of all of these algorithms is beyond the scope of this thesis. The interested reader should refer to one of the many excellent surveys ([38], [61], [63], [108], [126], [173]). In this chapter, we review the most referred in texture analysis literature and the most recently devised texture characterisation methods, with emphasis on those being employed in medical image analysis. The Spatial Grey Level Dependence Method (SGLDM) is presented in a separate chapter (Chapter **4**).

2.1 Grey level difference method

The grey level difference method has been widely used in texture analysis applications, especially in remote sensing ([47], [94], [95], [174]). This method is based on statistics computed from a grey level difference histogram ([38], [176]). A grey level difference histogram gives the frequency of occurrence of pairs of pixels separated by a displacement vector d and whose grey levels differ by a fixed amount k. It is defined as:

$$H(k;d) = \#\{((x_1,y_1),(x_2,y_2)) : |I(x_1,y_1) - I(x_2,y_2)| = k \wedge x_2 = x_1 + dx \ \wedge \ y_2 = y_1 + dy\} \quad (\mathbf{2}.1)$$

where $d = (dx, dy)$ is the displacement vector, $I(x,y)$ is the grey level of the image at pixel (x,y) and symbol # denotes the cardinality of a set. The shape of

this histogram can be intuitively related to texture structure. If the displacement vector d is small, high frequency values for large differences (large k) indicate a fine texture, whereas high frequency values for small differences (small k) indicate a coarse texture.

Usually, the normalised grey level difference histogram is computed, that is, $H(k;d)$ is divided by the total number of pairs of pixels separated by the displacement vector d. One statistic that has been used as a texture feature is the mean statistic defined as:

$$MEAN = \sum_k k \cdot H^{(norm)}(k;d) \qquad (\mathbf{2}.2)$$

(see [38]). Other features that can be extracted from the normalised grey level difference histogram include contrast, angular second moment, entropy ([176]) and inverse difference moment ([33]).

2.2 Grey level run length method

A grey level run length primitive (or simply grey level run) is a maximal collinear connected set of pixels having the same grey level. Grey level runs are characterised by the grey level of the run, the length of the run, and the direction of the run. Their intuitive relation to the structure of a texture is that a coarse texture is expected to give relatively long runs more often, whereas a fine texture should primarily contain short runs.

The grey level run length method is based on grey level runs. It was introduced by Galloway ([58]). Galloway estimated the joint probability $p(i,j;\vartheta)$ of runs of grey level i and length j from a textured region for various directions ϑ. From this estimated joint probability she extracted five texture features, namely short run emphasis, long run emphasis, grey level non-uniformity, run length non-uniformity, and image fraction in runs. This texture analysis method has been employed especially in remote sensed data analysis ([1], [9]).

According to [33], the grey level run length method presents two drawbacks:

- it is very sensitive to noise

- it does not capture the very important information contained in the second order probabilities of the form $p(i,j)$, $i \neq j$, where i, j are grey levels of pixels at a specific interpixel distance and orientation.

2.3 Texture description using stochastic models

Texture description methods based on stochastic models consider textures as realisations of stochastic processes. These models assume that a pixel has some kind of dependence on its neighbourhood. This can be a linear dependence (e.g. autoregressive models) or a conditional probability (e.g. Markov random field model). The parameters of the model are estimated from the analysed region and used as texture descriptors. A number of stochastic models have been used in the past for texture description.

In [21], an image region $I(x,y)$ is decomposed into a series of grey level planes defined as:

$$f_g(x,y) = \begin{cases} 1\,, & \text{if } I(x,y) = g \\ 0\,, & \text{if } I(x,y) \neq g \end{cases} \tag{2.3}$$

for $g = 0, 1, \ldots, N_g - 1$, where N_g is the number of grey levels in the analysed texture. The texture is characterised by the probability distribution of the pixel counts X_g, where $X_g = \sum_x \sum_y f_g(x,y)$. Actually, X_g is the number of pixels that belong to grey level plane g. A parametric stochastic model is used for describing the probability distribution of X_g, $g = 0, 1, \ldots, N_g - 1$, for the given texture. If the pixels in each grey level plane are assumed to be distributed randomly, the pixel counts can be described by a Poisson distribution. In some cases, however, this assumption is not valid due to local clustering of points.

One class of models that are capable of describing clustered data includes mixtures of Poisson distributions. For a mixture of k different Poisson densities the model is described by:

$$P(X_g) = \sum_{j=1}^{k} \pi_g^{(j)} \cdot P_{\lambda_g^{(j)}}(X_g) \tag{2.4}$$

where $\pi_g^{(j)}$, $j = 1, \ldots, k$ are the relative frequencies of the Poisson densities $P_{\lambda_g^{(j)}}(x)$ with parameters $\lambda_g^{(j)}$, i.e.

$$P_{\lambda_g^{(j)}}(x) = \frac{(\lambda_g^{(j)})^x \cdot e^{-\lambda_g^{(j)}}}{x!}, \quad j = 1, \ldots, k \tag{2.5}$$

The authors claim that mixture models of 2 and 3 Poisson densities give very satisfactory results in texture classification.

Markov random fields (MRF) have been widely used as texture models. In this context, a Markov random field considers the grey level of a pixel to be dependent on the grey levels of pixels in some neighbourhood. Actually, this is always the case in reality unless the image is simply random noise. Let L be a lattice of size $N \times N$ and $I(x, y)$ denote the grey level at pixel (x, y) on lattice L. We can relabel $I(x, y)$ to be $I(i)$, $i = 1, 2, \ldots, M$ where $M = N^2$. Point j is said to be a neighbour of point i if the conditional probability

$$P(I(i)|I(1), I(2), \ldots, I(i-1), I(i+1), \ldots, I(M))$$

depends on $I(j)$. The above relation does not imply that the neighbours of a point are necessarily close in terms of distance, although this is the usual case. A colouring $\mathbf{X}$ of lattice L with N_g levels is a function from the points of L to the set $\{0, 1, \ldots, N_g - 1\}$. According to [36], a Markov random field is described by a joint probability density on the set of all possible colourings $\mathbf{X}$ of the lattice L which satisfies the following conditions:

- positivity: $P(\boldsymbol{X}) > 0$, for all $\mathbf{X}$,
- Markovianity: $P\{X(i)|X(j)$ for all $j \in \{1, \ldots, M\}$, $j \neq i\} =$
  $P\{X(i)|X(j)$ for all j that belong to a neighbourhood of $i\}$,
- homogeneity: the above conditional probability depends only on the configuration of the chosen neighbourhood and is translation invariant.

In [36], the so-called autobinomial model is utilised (see also [14]). In this model, the conditional probability $P\{X(i) = k|X(j)$ for all j in neighbourhood$\}$ where $k \in \{0, 1, \ldots, N_g - 1\}$, is binomial with parameter $\vartheta(T) = \frac{e^T}{1+e^T}$ and number of tries $N_g - 1$. The value of T depends on the order of the model. The order of

the Markov random field in turn depends on the neighbourhood (see [36]). From the above, the conditional probability $P\{X(i) = k|$ neighbours$\}$ can be written as:

$$P\{X(i) = k|T\} = \begin{pmatrix} N_g - 1 \\ k \end{pmatrix} \cdot \left\{ \frac{e^T}{1+e^T} \right\}^k \cdot \left\{ \frac{1}{1+e^T} \right\}^{N_g-k-1} \quad (2.6)$$

Cross et al. ([36]) claimed that texture properties, such as directionality and coarseness, can all be reflected in the parameter values of the above model. The Markov random field model is more appropriate for microtextures. Textures showing regularity and inhomogeneous textures do not seem to fit very well. Also, large image regions are required to get good parameter estimates.

In [27], a texture description method was proposed which employs the correlation coefficients between pixel pairs $(I(x,y), I(x+d_x, y+d_y))$, where (d_x, d_y) represents the displacement vector. It holds that $0 \leq \max(d_x, d_y) \leq d$, where d is a small number. More specifically, two pixels far away from each other are assumed to be uncorrelated. The texture is modelled by a two-dimensional second order discrete (homogeneous) Gaussian random field. Therefore, the second order probabilities $p(k,l;d_x,d_y) = Prob\{I(x,y) = k, I(x+d_x, y+d_y) = l\}$ are characterised by the mean vector $\mathbf{m}$ and covariance matrix $\mathbf{V}$. Since the underlying random field is stationary (homogeneous), $\boldsymbol{m} = (m,m)$, where m is the expected value of the grey level of a pixel (average image brightness). The covariance matrix $\mathbf{V}$ is defined as:

$$\mathbf{V} = \begin{bmatrix} \sigma^2 & \rho_{d_x d_y} \cdot \sigma^2 \\ \rho_{d_x d_y} \cdot \sigma^2 & \sigma^2 \end{bmatrix} = \sigma^2 \cdot \begin{bmatrix} 1 & \rho_{d_x d_y} \\ \rho_{d_x d_y} & 1 \end{bmatrix} \quad (2.7)$$

where σ^2 is the variance of grey level and $\rho_{d_x d_y}$ the correlation coefficient for the displacement vector (d_x, d_y). These coefficients describe the spatial dependencies of pairs of pixels (second order statistics) and can be used as texture features.

Another way to describe texture is by using autoregressive models. In these models, the grey level $I(x,y)$ of a pixel (x,y) is expressed as a linear combination of the grey levels of pixels in a neighbourhood of (x,y). In [83], a univariate autoregressive model was employed for texture modelling. The image being modelled was assumed to be homogeneous. Consider an infinite periodic image

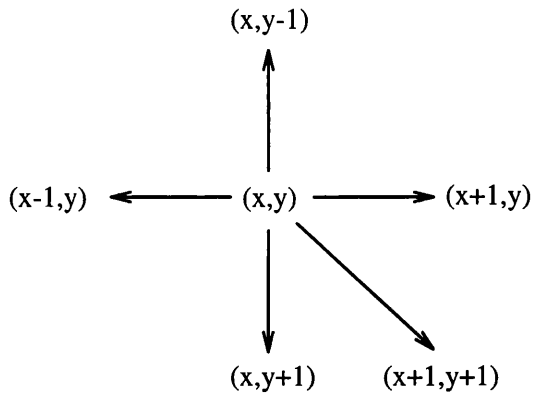


Figure 2.1: A neighbourhood configuration.

$I(x,y)$, $-\infty < x,y < \infty$. This image is obtained by repeating the corresponding finite image an infinite number of times, using an appropriate repetition formula. Then, $I(x,y)$ is assumed to be generated from the random field described by the autoregressive equation:

$$I(x,y) + \sum_{k=1}^{n} \vartheta_k \cdot I(x_k, y_k) = \sqrt{\beta} \cdot u(x,y) \qquad (2.8)$$

where $\{(x_k, y_k)\}_{k=1\ldots n}$ defines the effective neighbourhood of influence for pixel (x,y). For example, a possible neighbourhood is $\{(x-1,y), (x,y-1), (x+1,y), (x,y+1), (x+1,y+1)\}$ (see Figure 2.1). $\{u(x,y)\}_{-\infty<x,y<\infty}$ is a sequence of Gaussian independent identically distributed (i.i.d.) random variables with zero mean and unit variance. The texture is described by the estimated parameters (θ, β), where $\theta = (\vartheta_1, \ldots, \vartheta_n)^T$.

2.4 Texture analysis based on local linear transforms

In [92], a set of texture energy transforms was presented, which characterise texture based on measurements made in local neighbourhoods of pixels. The transforms require one-dimensional convolutions and simple non-linear operations. The original image is first filtered with a set of small convolution masks, typically 5×5, with integer coefficients. One-dimensional convolution masks (vector masks) are

first defined, which correspond to features such as edge, spot, and wave, e.g.:

$$E_3 \;:\; -1 \;\; 0 \;\; 1$$

$$S_3 \;:\; -1 \;\; 2 \;\; -1$$

$$W_5 \;:\; -1 \;\; 2 \;\; 0 \;\; -2 \;\; 1$$

The index of the above vectors indicates their size.

The 1×3 vectors form a basis for the larger vector sets. Each 1×5 vector can be generated by convolving two 1×3 vectors. A set of 1×7 vectors can be generated by convolving 1×3 and 1×5 vectors, or by twice convolving 1×3 vectors. Similarly, by convolving a vertical vector mask with a horizontal vector mask, matrix (two-dimensional) masks are formed, e.g.

$$S_3E_3 \;:\; \begin{matrix} 1 & 0 & -1 \\ -2 & 0 & 2 \\ 1 & 0 & -1 \end{matrix}$$

The names indicate the vector masks used for their generation. In a similar manner, we can get 5×5 and 7×7 masks.

The most important property of the above two-dimensional masks is that they are separable, so only one-dimensional convolution operations with the image are required, which are computationally cheap. After the image is convolved with the appropriate mask, it is processed with a non-linear local texture energy filter, such as a moving-window average of the absolute filtered image values. Texture is characterised by the texture energy values at each pixel location resulting from the application of several convolution masks.

2.5 Texture analysis based on fractals

Fractals provide a mathematical framework to study the irregular complex shapes found in nature. According to Mandelbrot ([105]), one important criterion for a bounded set A in Euclidean n-space to be a fractal set is its self-similarity. A is said to be self-similar when it is the union of $N(r)$ distinct (non-overlapping)

copies of itself, each of which has been scaled down by a ratio r in all co-ordinates. The fractal or similarity dimension D of A is given by the relation: $1 = N(r) \cdot r^D$, or equivalently $D = \frac{\log N(r)}{\log \frac{1}{r}}$. The natural fractal surfaces, such as the textural surfaces, do not in general possess this deterministic self-similarity. Instead, they exhibit statistical self-similarity, that is, they are composed of $N(r)$ distinct subsets each of which is scaled down by a ratio r and is identical in all statistical respects to the original. The fractal dimension of these surfaces is also given by the above relations.

Pentland ([122]) presented evidence that most natural surfaces are spatially isotropic fractals and that intensity images of these surfaces are also fractals. Actually, this work provided the foundation for the use of features derived from fractal models (e.g. fractal dimension) in image analysis. More importantly, Pentland introduced fractal based texture analysis by showing that there exists a correlation between texture coarseness and the fractal dimension of a texture. He proposed fractal functions as texture models and the fractal dimension as the texture feature. The fractal dimension has been shown to correlate with the intuitive roughness of a texture. A small value of fractal dimension D indicates a smooth texture, whereas a large value of D indicates a rough texture. There are various methods for estimating the fractal dimension from a textured region (see [85], [131]).

Other fractal parameters have also been employed for texture characterisation. In [102], the H parameter was used. H was related directly to D, the fractal dimension, by $D = 2 - H$. Four values of H in each region of a textured image were estimated, one for each principal direction (horizontal, vertical, and two diagonals). Then, a composite average H was computed by averaging the H values from all regions of the analysed image. This composite H was used to characterise the texture. In [121], a texture description method was proposed based on the rate $S(\epsilon)$ of the decrease in the area of the grey level surface of a texture with increasing scale ϵ. The magnitude of this fractal signature (rate of decrease) relates to the amount of detail that is lost when ϵ increases. High values of $S(\epsilon)$ reflect strong grey level variations at distance ϵ. Thus, the fractal signature $S(\epsilon)$ gives important information about the roughness of the grey level surface.

Another important fractal parameter, which has been used for texture description is lacunarity. Given a fractal set A, let $p(m)$ be the probability that there are m points within a box of size L centred about an arbitrary point of A. It holds that $\sum_{m=1}^{K} p(m) = 1$, where K is the number of possible points within the box. The lacunarity can be defined as follows:

$$C(L) = \frac{M_2 - M^2}{M^2} \tag{2.9}$$

where $M = \sum_{m=1}^{K} m \cdot p(m)$ and $M_2 = \sum_{m=1}^{K} m^2 \cdot p(m)$. Lacunarity is small when the texture is dense (fine) and large when the texture is coarse. Actually, this parameter was introduced to describe fractal surfaces having the same fractal dimension but different appearances (see [105]). According to [85], the fractal dimension alone is not sufficient for the description of natural textures. The authors claimed that both lacunarity and fractal dimension need to be employed for this purpose.

2.6 Texture analysis methods based on mathematical morphology

Another way to characterise texture is by applying morphological operations on the analysed image (see [135]). In [175], an attempt to describe texture using grey level mathematical morphology was presented. In this paper, texture was characterised by a number of signatures derived by applying morphological operations on the image using linear structuring elements of different lengths and orientations. Morphological operations included dilation, erosion, opening, and closing (for definitions see [66], [147]). Corresponding matrices M were formed for each operation, where the $M(i, j)$ entry was equal to the sum of the grey levels of the pixels in the image generated by operating on the initial image using the structuring element of length i and orientation j. Texture signatures were the entire matrices, the sums of rows of the matrices over all angles, which measured rotation invariant features, and the sums of columns over all lengths, which measured orientation features of all spatial sizes.

A special approach in texture analysis using mathematical morphology is based on granulometric size distributions. Here, an image is treated as a collection of grains. By sieving the grains using sieves of increasing mesh size the residual area of the image is being reduced as larger grains fall through the sieve. The distribution of the size of the residual area can be employed for texture characterisation (see also [135]).

Dougherty et al. ([46]) followed this approach. The binary morphological operation opening was employed. Given a binary image I and a structuring element H, the opening of I by H is defined by:

$$I \circ H = \bigcup_{\{x|H_x \subset I\}} H_x \tag{2.10}$$

where H_x is the translation of H by x. In other words, $y \in I \circ H$ if and only if there exists some translation H_x of H such that $y \in H_x \subset I$. Key to the construction of granulometries is the following property of the opening operation: if $F \circ H = F$ (in this case F is called H-open), then for any binary image I, $I \circ F \subset I \circ H$. An immediate consequence is that if $H_1, H_2, H_3, \ldots$ is an increasing sequence of structuring elements for which H_{k+1} is H_k-open, then the filtered (opened) images form a decreasing sequence $I \circ H_1 \supset I \circ H_2 \supset I \circ H_3 \supset \ldots$. The image sequence $\{I \circ H_k\}$ is called a granulometry.

Counting the number of pixels remaining in each succeeding opening, we get a decreasing function $\Psi(k)$ such that for some K, $\Psi(k) = 0$, for $k \geq K$. Textural information can be derived by studying the size distribution $\Psi(k)$. In practice, H_1 consists of a single point, so that $\Psi(1)$ gives the total number of pixels with value 1 in binary image I (by convention, in binary images the pixels belonging to the objects get value 1, whereas the pixels belonging to the background get value 0). Since $\Psi(k)$ is a decreasing function, the normalisation $\Phi(k) = 1 - \frac{\Psi(k)}{\Psi(1)}$ is a probability distribution function. Then, the discrete derivative $d\Phi(k)$ is a discrete probability density function. In morphological granulometry nomenclature it is called the pattern spectrum of image I. Moments of the pattern spectrum, such as the mean and standard deviation, can be employed as texture features. A natural extension of this method is the grey level morphological granulometry. It involves the grey level opening of the grey

level image using grey level structuring elements of various shapes (e.g. spheres).

2.7 Texture analysis based on multichannel filtering

Multichannel filtering based texture analysis has been an area of research for several years, mainly due to the support that has received from neurophysiological experiments. It is based on decomposing a textured image into feature images using a bank of filters. The "channels" corresponding to different filters are assumed to capture some specific local characteristics of the input texture such as directionality, edgeness, etc. A general filter bank is often too large because it is designed to capture general texture properties. An input image with a given texture can be usually characterised by employing only a small subset of filters. This gives rise to an important problem with these texture analysis methods, that is, minimising the number of filters while keeping texture characterisation power at an acceptable level (filter selection problem).

The most widely used multichannel/multiresolution texture analysis methods are the Gabor transform and the wavelet transform. The term multiresolution comes from their ability to characterise texture in several scales. Through this multiscale texture characterisation, it has been claimed that they lead to better texture discrimination than methods which primarily depend on a single scale, such as the SGLDM and the Markov random field model.

These transforms are also called spatial/spatial-frequency methods, because they are based on image representations that capture the frequency content in localised regions in the spatial domain. Spatial/spatial-frequency representations are capable of preserving both local and global information. Thus, they overcome the shortcomings of the traditional Fourier-based techniques which are due to lack of locality. Such methods are able to achieve good resolution in both the spatial and frequency domains and are consistent with recent theories on human vision. Spatially large linear filters have a small spatial frequency extent. Such filters can supply precise information about the periodicities, which can distinguish the

textures in much the same way as with the Fourier power spectrum, but may be insensitive to local texture features. On the other hand, spatially small linear filters have a large spatial frequency extent. Such filters can provide more precise information about local features and therefore, provide the means of distinguishing texture elements. However, they are less sensitive to the frequencies that characterise the distribution of these elements.

Gabor filters can be intermediate to the above bandwidth extremes and allow simultaneous measurement of texture elements and their distributions, giving optimal joint sensitivity to local and global properties of texture. The Gabor decomposition of an image has been extensively used for texture characterisation (see [5], [6], [16], [49], [73], [76], [123]). As we briefly mentioned above, there is an *uncertainty relation* associated with 2D spatial linear filters, which limits the resolution simultaneously attainable in space and frequency domain. Decreasing the effective spatial area of the filter has the inevitable result of increasing its effective area in the frequency domain, thereby decreasing its spatial frequency and orientation selectivity. In [37], it was shown that the 2D Gabor filters have optimal joint resolution in that they minimise the product of effective areas occupied in the 2D space and frequency domains, achieving the theoretical lower limit $\frac{1}{16\pi^2}$ for the joint entropy (uncertainty) of 2D spatial position, orientation, and frequency.

In [164], a 2D Gabor filter was realised as a sinusoidal plane wave of some frequency and orientation within a two-dimensional Gaussian envelope. The 2D Gabor filters used in this paper were discrete realisations of the function:

$$G(x, y; \omega, \vartheta, \rho, x_0, y_0) = e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \cdot \sin[\omega(x\cos\vartheta - y\sin\vartheta) + \rho] \qquad (\mathbf{2}.11)$$

where x_0, y_0 specified the centre of the 2D Gaussian envelope. With unit aspect ratio the standard deviation σ was the same along both the horizontal and vertical axes. For the sinusoidal plane wave, ω was the frequency, ϑ the angle of orientation and ρ the phase of the plane wave. A pair of quadrature filters was used for each frequency and orientation, where phase ρ was 0 and $\frac{\pi}{2}$ for the two filters of the pair, respectively.

For each region in the analysed image, the correlation of each filter pair with

the pixels in the region was carried out:

$$R_1(x_0, y_0; \omega, \vartheta) = \sum_x \sum_y G(x, y; \omega, \vartheta, \rho_1, x_0, y_0) \cdot I(x, y) \quad (2.12)$$

$$R_2(x_0, y_0; \omega, \vartheta) = \sum_x \sum_y G(x, y; \omega, \vartheta, \rho_2, x_0, y_0) \cdot I(x, y) \quad (2.13)$$

where x_0, y_0 was the centre of the analysed region, R_1, R_2 were the two response functions of each filter pair, $I(x, y)$ was the analysed image and ρ_1, ρ_2 the two phases of each quadrature filter pair ($\rho_1 = 0$ and $\rho_2 = \frac{\pi}{2}$). Each image region was characterised by either the Pythagorean sum of the above responses:

$$A(x_0, y_0; \omega, \vartheta) = [R_1^2(x_0, y_0; \omega, \vartheta) + R_2^2(x_0, y_0; \omega, \vartheta)]^{\frac{1}{2}} \quad (2.14)$$

or the sum of the absolute values:

$$B(x_0, y_0; \omega, \vartheta) = |R_1(x_0, y_0; \omega, \vartheta)| + |R_2(x_0, y_0; \omega, \vartheta)| \quad (2.15)$$

The mean of the measures performed in the regions of a textured image was proposed as a texture feature. Alternatively, the variance of the above measures could be used for texture characterisation.

As we have already mentioned, the largest problem with Gabor filters is the filter selection problem, since a general filter bank is often too large. Research efforts try to find methods to select a minimal set of filters among the general filter bank, that best characterise a given texture. In [77], a neural network approach was introduced for this purpose. The network was trained to find a minimal set of specific filters which can characterise the given textured image. Actually, the filter selection task was formulated as an optimisation problem of finding both the minimum number of the filters and their coefficients. Another disadvantage of Gabor filters is that they are computationally quite intensive. In [159], it was mentioned that the Gabor decomposition of images imposes excessive storage requirements and is computationally highly demanding. Also, if small changes in texture frequency and orientation have to be captured, the number of filters required is very large. In addition, the outputs of Gabor filter banks are not mutually orthogonal, which may result in a significant correlation between texture

features. Finally, these transformations are usually not reversible which limits their applicability for texture synthesis.

According to [167], most of the above problems can be avoided by applying the wavelet transform for the analysis and characterisation of a signal at different scales. The wavelet representation of an image can actually be used to characterise texture (see [104]). Based on the wavelet transform, the image can be decomposed into a set of independent, spatially oriented frequency channels, as shown below.

Let $I(x, y)$ be the image being decomposed and $A^d_{2^{j+1}}I$, $-J \leq j < 0$ is the discrete approximation of I at resolution 2^{j+1}, where $A^d_1 I$ is the discrete approximation of image I at the original resolution. Then, $A^d_{2^{j+1}}I$ can be decomposed into $A^d_{2^j}I, D^1_{2^j}I, D^2_{2^j}I$, and $D^3_{2^j}I$ using the wavelet decomposition. $A^d_{2^j}I$ is the discrete approximation of I at the next coarser resolution 2^j and $D^1_{2^j}I, D^2_{2^j}I, D^3_{2^j}I$ are the detail images which correspond to three spatial orientations (horizontal edges, vertical edges, diagonal edges) and represent the difference of information between 2^{j+1} and 2^j resolutions. Actually, $A^d_{2^j}I$ is derived by low-pass filtering the $A^d_{2^{j+1}}I$, while $D^1_{2^j}I, D^2_{2^j}I$, and $D^3_{2^j}I$ are derived by band-pass filtering the $A^d_{2^{j+1}}I$ in a certain orientation for each one. $D^1_{2^j}I$ gives the vertical high frequencies, $D^2_{2^j}I$ the horizontal high frequencies, and $D^3_{2^j}I$ the high frequencies in both directions. For any $J > 0$, $A^d_1 I$ is completely represented by the $3J + 1$ discrete images: $\left\{A^d_{2^{-J}}I, \{D^1_{2^j}I\}_{-J\leq j<0}, \{D^2_{2^j}I\}_{-J\leq j<0}, \{D^3_{2^j}I\}_{-J\leq j<0}\right\}$. The above set is called a wavelet representation of image I. The image $A^d_{2^{-J}}I$ is the coarse discrete approximation at resolution 2^{-J} and the $D^k_{2^j}I$ $(-J \leq j < 0,\ k = 1, 2, 3)$ images give the detail signals for the three different spatial orientations at different resolutions. However, a wavelet representation having more than three orientation tunings is feasible.

The wavelet transform has many useful properties, such as the multiresolution representation and orthogonality. It also gives fast algorithms compared to other multiresolution methods, such as the Gabor transform. For the above, it is considered an attractive tool for texture characterisation.

Another texture analysis approach, based on the wavelet packet transform, was presented in [24], [90]. According to the authors, a lot of textures contain significant information in the middle frequency region of their spectrum. Thus,

the pyramid-structured wavelet transform (standard wavelet transform) is not appropriate for characterising such textures, since it decomposes a signal only in the low frequency region of its spectrum. In this manner, frequency channels with significant energy may not be detected for texture discrimination. Through wavelet packets, a signal is decomposed not only in the low frequency region but in any of the four frequency regions of its spectrum mentioned in the previous paragraph, if sufficient energy is detected. A tree-structured decomposition is thus performed.

2.8 The generalised co-occurrence matrix

A generalisation of the grey level co-occurrence matrix is the Generalised Co-occurrence Matrix (GCM) ([39], [40], [158]), which models image texture as a two-dimensional spatial arrangement of local image features. In the co-occurrence matrix approach (SGLDM), the image feature is the pixel, the attributes of the feature are the position and grey level of the pixel and the spatial relation is the displacement vector defined on the position attributes of pairs of pixels (see Section 4.1). In the GCM method, the notion of image feature is extended to a set of more complex features including edge points, edge segments, uniform regions, etc. Let this set of features be denoted by $Y = \{y_i | i \in [0, n_y)\}$. The second generalisation allows an image feature to possess a set of attributes. Attribute a of feature y can take on value $V_a(y) \in \{k | k \in [0, n_a)\}$. For example, the orientation attribute of the edge point feature may be quantised to 8 different integer values in the range $[0, 8)$, denoting angular increments of 45 degrees. The third generalisation involves an extension of the spatial relation notion. It is convenient to define a Boolean function of pairs of features (specifically, of their attributes which determine their spatial position) $R(y_i, y_j)$ called a spatial predicate. An arbitrary second order spatial relation between pairs of features may then be specified using an appropriate spatial predicate.

The generalised co-occurrence matrix of an attribute a over a set of image features y, pairs of which satisfy a spatial relation R, is a square matrix

$C(y, a, R) = [c_{ij}(y, a, R)]$ of dimension n_a, whose entries $c_{ij}(y, a, R)$ are defined as:

$$c_{ij}(y, a, R) = \frac{\#\{(y_k, y_l)|y_k, y_l \in Y \wedge R(y_k, y_l) = TRUE \wedge V_a(y_k) = i \wedge V_a(y_l) = j\}}{N} \tag{2.16}$$

N is a normalisation factor defined as:

$$N = \#\{(y_k, y_l)|y_k, y_l \in Y \wedge R(y_k, y_l) = TRUE\} =$$

$$\sum_{i=0}^{n_a-1} \sum_{j=0}^{n_a-1} c_{ij} \tag{2.17}$$

The division by this factor is required so that the matrix can approximate a discrete joint probability density of co-occurring attributes under the given spatial relationship.

2.9 Fourier transform based texture analysis

Fourier transform has been extensively used in texture analysis, especially in characterising the coarseness and directionality of a given texture. The power spectrum is actually used. The radial distribution of values in the power spectrum is sensitive to texture coarseness. If texture is coarse, high values of the power spectrum concentrate near the origin, while in fine texture the values are more spread out. Also, the angular distribution of values in the power spectrum is sensitive to the directionality of the texture. A texture with many edges or lines in a given direction ϑ has a power spectrum with high values concentrated around the perpendicular direction $\vartheta + \frac{\pi}{2}$, while in a non-directional texture the power spectrum should also be non-directional.

The features used for texture characterisation are derived from sampling the power spectrum using various sampling geometries (see [96]). The power spectrum is defined by the relation $\Phi_I(u, v) = |F_I(u, v)|^2$, where $F_I(u, v)$ is the Fourier transform of image I. These features are:

- annular-ring sample signatures, which give a measure of the texture coarseness and are defined by:

$$a_i = \int_0^{2\pi} \int_{\rho_i}^{\rho_i+\Delta\rho} \Phi_I(\rho,\vartheta) \cdot \rho \, d\rho d\vartheta, \quad i = 1, 2, \ldots, m_a \tag{2.18}$$

where $\rho = (u^2+v^2)^{\frac{1}{2}}$, $\vartheta = \tan^{-1}\left(\frac{v}{u}\right)$ (polar co-ordinates) and m_a the number of annular rings

- wedge sample signatures, which contain directionality information and are defined by:

$$\omega_i = \int_{\rho_{min}}^{\rho_{max}} \int_{\vartheta_i}^{\vartheta_i+\Delta\vartheta} \Phi_I(\rho,\vartheta) \cdot \rho \, d\vartheta d\rho, \quad i = 1, 2, \ldots, m_\omega \tag{2.19}$$

where m_ω is the number of wedges

- slit sample signatures, which is used less frequently and can be described by:

$$s_i(\vartheta) = \int_{-v_{max}}^{v_{max}} \int_{u_i}^{u_i+\Delta u} \Phi_I(u,v) du dv, \quad i = 1, 2, \ldots, m_s \tag{2.20}$$

where it is assumed that the power spectrum is first rotated to a desired direction (angle ϑ) and m_s is the number of slits.

Fourier spectrum features have been extensively used in the past for texture characterisation ([1], [5], [6], [9], [33], [51], [93], [176]). Fourier transform based methods usually perform well on textures showing strong periodicities. Their performance significantly deteriorates, though, when the periodicities weaken. Also, from a computational point of view Fourier transform is very expensive. It is worthwhile to note here that the textural content of the phase spectrum is low (see [52]).

In [100], a texture analysis method based on the power spectrum and phase spectrum of a homogeneous texture image was proposed. 28 texture features were presented, which measure the shape of the spatial frequency spectrum and include the location, size, and orientation of peaks as well as the entropy of the normalised power spectrum. Let $P(u,v)$ be the power spectrum obtained from a homogeneous texture region. Then, the normalised power spectrum $p(u,v)$ is defined as:

$$p(u,v) = \frac{P(u,v)}{\sum_{u,v\neq 0} P(u,v)} \tag{2.21}$$

The normalised spectrum has the characteristics of a probability distribution. According to [100], the major advantage of texture features computed in the spatial frequency domain is that they seem to be less sensitive to typical noise processes than spatial domain features. This happens because such noise processes tend to dramatically alter local spatial variation of grey level intensity, while having relatively uniform spatial frequency characteristics. Again, the phase spectrum was shown to be relatively unimportant for texture analysis.

2.10 Various statistical texture analysis methods

2.10.1 Texture analysis based on texture spectrum

A new texture analysis method based on the notion of the texture spectrum was proposed in [68]. It has been employed in remote sensed data analysis ([60]). Texture is characterised by a set of basic small units, termed texture units, which are defined in local neighbourhoods of pixels.

More specifically, for each neighbourhood 3×3 a set TU is defined which contains 8 elements, i.e. $TU = \{E_1, E_2, \ldots, E_8\}$, where E_i corresponds to the ith 8-neighbour of the central pixel (x, y) in the 3×3 neighbourhood (assuming a certain ordering of its 8-neighbours). E_i is defined by:

$$E_i = \begin{cases} 0\,, & \text{if } v_i < v_0 \\ 1\,, & \text{if } v_i = v_0 \\ 2\,, & \text{if } v_i > v_0 \end{cases} \tag{2.22}$$

where v_0 is the grey level of pixel (x, y) and v_i is the grey level of its ith 8-neighbour. The above set is the texture unit for pixel (x, y).

The frequency distribution of texture units defines the texture spectrum which reveals the global textural properties. Different textures have correspondingly different spectra when comparing certain characteristics, such as number, position,

width, and height of their principal peaks. Several texture features can be defined over the texture spectrum (see [68]). They are categorised according to whether they measure the macro-texture or the micro-texture of the analysed image.

2.10.2 Textural edgeness

Texture can also be conceived in terms of edgeness in an image region. A gradient image can be computed from the analysed region applying a gradient operator, such as the quick Roberts gradient operator at each pixel (the sum of the absolute value of the differences between diagonally opposite neighbouring pixels) (see [127]). A texture measure can be the average value of the gradient in the analysed region.

Sutton and Hall ([150]) extended this idea by making the gradient a function of the distance between the pixels. For every distance d and image region I defined over neighbourhood N they computed:

$$g(d) = \sum_{(x,y)\in N} \{|I(x,y) - I(x+d,y)| + |I(x,y) - I(x-d,y)| +$$

$$|I(x,y) - I(x,y+d)| + |I(x,y) - I(x,y-d)|\} \qquad (\mathbf{2}.23)$$

Sutton and Hall applied this textural measure in a pulmonary disease identification experiment and obtained very good results in discriminating between normal and abnormal lungs. Textural edgeness has also been employed for the characterisation of remote sensed data ([1], [9], [94]).

2.10.3 The autocorrelation function as a texture descriptor

As we have already mentioned in Section **1**.1, one texture property relates to the spatial size of the tonal primitives. Tonal primitives of large size indicate a coarse texture, whereas tonal primitives of small size indicate a fine texture. The autocorrelation function gives information about the size of the tonal primitives

and therefore, the coarseness of a given texture. It is defined as follows (see [63]):

$$\rho(x,y) = \frac{1}{(L_x-|x|)\cdot(L_y-|y|)} \cdot \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(u,v)\cdot I(u+x,v+y)\,dudv}{K}, \quad |x| < L_x \text{ and } |y| < L_y \tag{2.24}$$

$$K = \frac{1}{L_x \cdot L_y} \cdot \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I^2(u,v)dudv \tag{2.25}$$

where (x,y) is an interpixel distance, L_x, L_y define the size of the analysed region and $I(u,v)$ is the grey level at pixel position (u,v), $0 \leq u \leq L_x$ and $0 \leq v \leq L_y$. We assume that outside the analysed region $I(u,v) = 0$. If the tonal primitives in this region are relatively large the autocorrelation function will decrease slowly with distance. If, on the other hand, they are small it will decrease quickly with distance. The autocorrelation function has been used in the analysis of remote sensed data ([1]).

2.10.4 Texture analysis using the statistical feature matrix

In [177], a new texture analysis method was proposed based on the statistical feature matrix. Let $\delta = (x,y)$ be an interpixel distance vector. A δ statistical feature is defined as a second order statistical feature of an image for interpixel distance δ. In [177], δ contrast, δ covariance, and δ similarity measures were used as δ statistical features and were defined as:

- δ contrast: $CON(\delta) \equiv E\{[I(u,v) - I(u+x,v+y)]^2\}$
- δ covariance: $COV(\delta) \equiv E\{[I(u,v) - n]\cdot[I(u+x,v+y) - n]\}$
- δ dissimilarity: $DSS(\delta) \equiv E\{|I(u,v) - I(u+x,v+y)|\}$

where $E\{\cdot\}$ denotes the expectation operation, $I(u,v)$ is the image, and n the average grey level of image I. The statistical feature matrix is a matrix having dimensions $(L_r+1) \times (2L_c+1)$, whose (i,j) element ($i = 0,1,\ldots,L_r$, $j = 0,1,\ldots,2L_c$) is the δ statistical feature of the image for $\delta = (j - L_c, i)$ and L_r, L_c are constants which determine the maximum interpixel spacing distance.

Three statistical feature matrices were defined from the above statistical features, namely the contrast matrix, the covariance matrix, and the dissimilarity

matrix. The whole matrices were used as features for texture characterisation. The main disadvantage of this method is that it needs to scan the image data each time a different matrix (corresponding to different statistical feature) is computed.

2.11 Comparison studies of the various texture analysis methods

In [33], a comparison of four texture analysis methods, namely the SGLDM (co-occurrence matrix method), the grey level run length method, the grey level difference method, and the power spectral method, was performed. The comparison method used was a theoretical evaluation (in [33], it was called the theoretical evaluation methodology) of the degree in which a certain texture method captures the textural information contained in an image. In other words, this methodology measures the amount of textural information contained in the data structures of the four aforementioned methods, that is, the co-occurrence matrices, the grey level run length matrices, the grey level difference density functions, and the power spectra. Thus, the comparison carried out in the above paper was not dependent on the set of features used with these methods. Rather, the relative loss of important textural information that happens in going from the raw image data to the data structure of each method was evaluated and compared.

In this analysis, textures were first represented as random fields $X(n, m)$, $0 \leq n < \infty$, $0 \leq m < \infty$ generated using Markov chains. Then, the results were generalised using translation stationary random fields of which the Markov random field was a special case. Pairs of different (visually distinct) textures were considered in the evaluation process. The results showed that the SGLDM is better than the other three methods in all cases (Markov random field and translation stationary random fields in general) but one. The authors could not generalise this result for the grey level run length method in the case of the translation stationary random fields. However, due to the drawbacks of the grey level run length method (see Section **2**.2), the authors claimed that the superiority of the SGLDM is certain in all cases.

In [176], an evaluation of the aforementioned texture analysis methods was performed. The evaluation measured the ability of these methods to discriminate terrain types in aerial and satellite images. The metric of comparison was the percentage of the overall correct classification. The conclusions drawn from the above study were the following. The texture features based on the SGLDM performed better in classifying terrain types than the ones based on the power spectral method. Also, they did about equally well with those from the grey level difference method in the performed classifications. This conclusion was different from the one drawn in the theoretical comparison of Conners and Harlow ([33]), which found the SGLDM to be better than the grey level difference method. Conners and Harlow claimed that this difference was reasonable since, as they proved in [33], the features based on the co-occurrence matrices, which were used by Weszka et al. ([176]) for their classification, namely energy, entropy, correlation, local homogeneity, and inertia (see Section **4**.1 for definition), do not capture all the important textural information contained in the grey level co-occurrence matrices.

In [117], a comparison between four texture analysis methods, based on their performance in recognising classes of visual texture, was carried out. The methods compared were the SGLDM, the discrete Markov random field model, the Gabor multichannel filters, and the fractal method. Four classes of textured images were used in the classification process; two types of synthetic images (fractal surfaces and Gaussian Markov random field images) and two types of natural images (leather samples and painted surfaces). According to the classification results in this paper, the SGLDM was ranked first, with the fractal method being close. The Markov random field model and the Gabor filters did not perform very well in this classification where small regions were analysed.

In [74], a comparative study of several texture analysis methods was carried out, with particular emphasis on their applicability to image segmentation. Simple bipartite synthetic texture images combining different stochastic textures separated by a stochastic boundary, were used for feature extraction and segmentation. Several methods were tested, such as the SGLDM, texture analysis employing the fractal dimension, Laws' texture energy measures (see Section **2**.4), etc.

The above study showed that the SGLDM was among the methods that gave the best overall segmentation results.

Chapter 3

Texture analysis in medical imaging

3.1 Introduction

The fundamental objective of any diagnostic imaging investigation is tissue characterisation. This means that images are acquired with the aim of determining whether the tissue in the chosen region of investigation possesses normal or pathological characteristics. Furthermore, based on the appearance of the relevant tissue in the image an attempt is made to classify the pathology (type of abnormality) as precisely as possible. This process, which is typically performed by an expert radiologist, requires a complex assessment of the various image features that characterise this appearance and their comparison with the radiologist's empirical database. Various biomedical imaging modalities are currently widely used in hospitals and clinics for diagnosis, disease treatment, surgical planning, etc., such as Computed Tomography (CT), Electron Micrograph (EM), ultrasound, and Nuclear Magnetic Resonance (NMR).

Making a diagnosis from a medical image is a process which involves both the perception of "signs" and decision making based on what is perceived. Hence, the process is subject to both the failings of the human perceptual system as well as misjudgements in the decision making process induced by fatigue and boredom. Human factors such as intelligence, experience, mood, distraction, fatigue, and

visual acuity modify the effectiveness with which radiologists can perceive and diagnose information patterns in medical images. More importantly, the human eye-brain complex is only able to appreciate a limited level of complexity in an image. According to [80], [81], textures that have the same second-order statistics cannot be visually discriminated.

Given the inherently subjective nature of many of the judgements associated with the diagnostic process, there exists a significant range of diagnostic problems for which the accuracy of this process is not acceptable. Furthermore, clinicians are increasingly looking for greater specificity in the pathological characterisation of tissues from their appearance in images, so that they be able to offer more effective treatment regimens to their patients. Also, with the high expense of some medical imaging methods it is very important that the maximum possible diagnostic information be extracted from the acquired images. Consequently, it is not surprising that investigators try to find methods to aid the clinicians in making diagnoses.

In reality, there is an increasing research effort towards the use of objective computer-based image analysis to automatically or semi-automatically extract diagnostically significant image features, which can be used to distinguish normal from pathological tissue and to further characterise the state of the pathological tissue. Research has shown that the ability of computers to make quantitative measurements over medical images leads to a greater reliability in diagnosis, due to reduction of the errors and variability found in qualitative measurements made by the radiologists.

Intensity, morphology, and texture are considered as important image features for the assessment of tissue appearance. First order statistical features, such as the mean and variance of pixel intensity, can be calculated quickly but extract mainly the information of the overall grey level distribution and are susceptible to variation of parameters, such as the brightness of the image. Image texture has been shown to be a particularly sensitive feature for the assessment of pathology. Texture analysis was able to increase the level of information extracted from the image and to quantitate differences in appearance imperceptible by human observers in many cases.

According to [97], texture analysis is a method of extracting the maximum available diagnostic information from medical images. Lerski claimed that it is of potential increasing value, as digital images and image processing systems become more widely available. According to [140], texture analysis is of great importance in the diagnostic process for many diseases. In [34], it was mentioned that the extraction of features which characterise the texture of the lungs and the bones is a very important process in chest analysis. Chest exams make up about a third of the total number of examinations performed in a typical radiology department, since physicians commonly employ such exams to determine the health status of a patient and especially the condition of the heart, the lungs and the bones, such as the ribs and the spine.

Medical applications of computer-based texture analysis date back to the early 1970s, when it became possible to digitise X-rays for computer processing. The explosion in the use of digital techniques has led to the possibility of utilising texture analysis for most medical imaging modalities. Typical work has been carried out on digital X-rays, ultrasound, CT, and MRI (Magnetic Resonance Imaging). Syntactic texture analysis methods are not as highly developed in the case of medical applications as the statistical methods. Statistical texture analysis methods can extract textural information from the spatial distribution of the grey levels, which has been shown to be of great importance in automatic tissue characterisation and diagnosis. According to [22], statistical texture analysis is of great value in biomedical image analysis. One of the most significant statistical texture analysis methods, which has found wide application in this area, is the SGLDM. According to [34], it is a superior general purpose algorithm for texture analysis.

3.2 Digital X-rays images

Texture analysis has been extensively employed in the processing of digital X-ray images by computers for detecting pathological conditions. In the following, we mention some of the most significant applications in this area.

The examination of chest radiographs (X-ray images) for the detection of

various diseases (especially lung diseases) is one of the most difficult problems in diagnostic radiology. The necessity for mass diagnostic screening of such radiographs has significantly increased radiologist's workload and has made the interpretation of chest X-ray images a considerable problem for them. Therefore, the automated computer analysis of these images has received increasing attention.

One of the most important lung diseases is Coal Worker's Pneumoconiosis (CWP). This disease was endemic among miners. Its diagnosis is dependent on a history of exposure to coal dust and the presence of certain distinctive features on the chest radiographs. Examination of this type of radiographs is not only the most reliable method for diagnosing the disease, but also the only way of assessing progression. The profusion and character of opacities observable on a chest radiograph indicate the severity of this disease. Profusion is the most significant parameter and is related to the number of opacities per unit area. Its accurate categorisation is an important medical problem. Texture analysis has been employed in a number of studies for the automatic classification of profusion.

In [150], measures based on textural edgeness were employed for the classification of chest X-ray images into normal and abnormal lung tissue including the case of pneumoconiosis. A high classification accuracy was reported. In [88], it was claimed that a high percentage of all radiographs examined are diagnosed as not showing radiographic evidence of pneumoconiosis. Therefore, a cost effective and fast automatic film reader to consistently screen out all definite normals, is more than necessary. According to the authors, lung regions with pneumoconiosis and normal lung regions contain distinct textures. The SGLDM and the power spectrum method were employed for texture characterisation. Classification results comparable to experienced radiologists were reported for both methods. A high classification accuracy in the detection of CWP was also reported by Stark and Lee ([146]) using the power spectrum texture analysis method. In [43], seven different texture analysis methods were used for the characterisation of textured regions in digital chest X-ray images. The goal of this study was the classification of pulmonary opacities into the classes defined by the International Labour Office (ILO) for the classification of radiographs of CWP. Among the methods used was

the SGLDM. Again, high classification accuracy was reported. Katsuragawa et al. ([84]) showed that measures obtained from the power spectrum of lung texture in radiographs correspond closely with the ILO classification categories for pneumoconiosis.

Another application of texture analysis is in the segmentation of digital X-ray images, which is a significant medical problem. In [109], an attempt was carried out to segment digital chest radiographs in several anatomical regions, e.g. the heart and the lungs. Segmentation in such radiographs is usually accomplished by a process of pixel classification, where the image is segmented into regions by assigning individual pixels to specific region classes. In this paper, a large set of locally measured candidate features was formed. Among them, there was a number of texture features which were mainly used for the segmentation of the lung regions of the chest radiographs. These regions are characterised by more grey level variation, due to the radiographic complexity of the lung vasculature. The texture features used in this study included measures from the SGLDM, Laws' texture energy measures, and the fractal dimension. Six of the eight features in the best feature set came from the SGLDM, proving for one more time the superiority of this method in medical image analysis.

An important application of texture characterisation methods is in the automatic analysis of bone X-ray images for the detection and monitoring of various bone diseases, the most prevalent of which is osteoporosis. In [102], fractals were used to model the texture present in radiographs of the human calcaneus (heel) during the course of immobilisation from a fracture and subsequent recovery. Fractal parameter H was actually employed for this purpose (see Section **2**.5). Following a period of immobilisation, some of the fine trabecular structure is dissolved away while the coarse structure is left behind. This is typical of the process of osteoporosis in which calcium is dissolved from the bone structure. In the above study, it was shown that parameter H was able to follow closely the course of osteoporosis. A lower value of this parameter indicated a greater degree of this disease. Therefore, it seems to be appropriate for its quantification.

In [128], it was claimed that the fractal dimension is a suitable and sensitive descriptor of interdental bone structure in digital radiographs and provides

an important means of monitoring the progress of osteoporosis. The same results found in [169], where the fractal dimension was used for the quantitative description of the alterations of the trabecular pattern in wrist radiographs of osteoporotic patients. Fortin et al. ([53]) considered ranging femora according to the degree of osteoporosis, a problem which radiologists find difficult to solve. The authors turned this problem into a texture analysis problem. As bone is lost, thin trabecular structures become reduced in number, length, and thickness, weakening the femoral neck as a whole and increasing the likelihood of fracture. Fractal parameter H was found to be appropriate for the quantification of the bone structure of the human femur from radiographs. Similarly, Samarabandu et al. ([130]) showed that fractal dimension is a robust texture measure for detecting different bone structures, irrespective of scale.

In [143], features derived from grey level morphological granulometry and Laws' texture energy measures were employed to characterise the texture of the trabeculae in the neck of the femur. They claimed that the former method (morphological granulometry) was slightly better in grading the degree of osteoporosis in the radiographs of the femoral neck. Southard and Southard ([145]) investigated the feasibility of using dental radiographs of maxillary alveolar bone as a possible screening tool for osteoporosis. In particular, they determined which textural features undergo changes most closely corresponding to bone mineral loss. They tested features from the SGLDM, the grey level difference method, the Fourier power spectrum method, Laws' texture energy measures, and the fractal dimension. They found that Laws' texture energy measures and the fractal dimension are the most promising features for the early detection of bone loss from dental radiographs. In [142], texture analysis of radiographs of the human femur was employed for their classification into classes based on the Singh Index. This index is used by the radiologists to describe the degree of trabecular bone loss from the femur in osteoporosis. Laws' texture energy measures and a novel texture analysis method were employed in this classification. The former method gave very encouraging results.

Certain periodontal diseases are also characterised by bone loss. In [89], dental radiographs were segmented using fractal texture analysis for the detection and

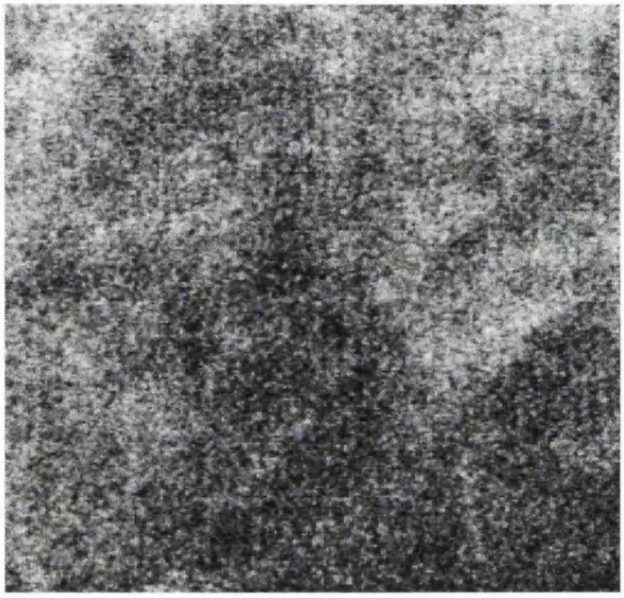
monitoring of such periodontal diseases. In particular, the segmentation was achieved through the identification of each pixel in the radiograph as being either bone, teeth, or a boundary between bone and teeth. The local fractal dimension was computed from a local region centred at each pixel and employed for its identification. The results obtained in this study indicated the potential utility of fractal analysis in the segmentation of dental radiographs.

Hodgkin's disease is another example in the automatic diagnosis of which texture analysis can play an important role ([34]). This disease is a malignancy of the lymphatic system. The problem in managing a patient with Hodgkin's disease is to sufficiently resolve the uncertainty about tumour extent, without harming the patient with invasive diagnostic procedures, such as laparotomy. Texture analysis of lymphographs (X-ray images of lymph nodes) was shown to be useful in the characterisation of the irregularity of the lymph nodes. The authors claimed that it can potentially help physicians to avoid laparotomy for the diagnosis of this disease.

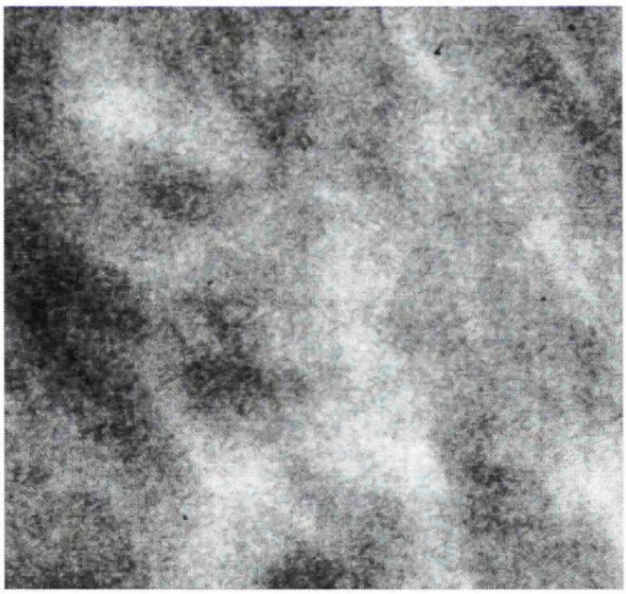
Breast cancer is the most prevalent type of cancer among women. Mammography (breast X-ray imaging) is at present the most effective method for the detection of early breast cancer. One normal and one abnormal mammogram are shown in Figures **3**.1 (a) and (b). However, it is well known that a considerable number of lesions which are visible on mammograms, are missed by the radiologists. A computer-aided diagnosis system can provide a consistent second opinion to a radiologist, which may reduce false-negative diagnoses. Furthermore, it can be useful in differentiating malignant from benign lesions, thereby decreasing the number of benign cases that are sent for biopsy. In addition, the necessity for breast screening programs has resulted in a large volume of mammograms requiring interpretation and a variety of computer systems have been proposed to help radiologists deal with them more efficiently.

A number of papers have presented applications of texture analysis methods in mammography. In [103], the SGLDM and the grey level difference method were employed for the characterisation of the texture of the breast parenchyma in mammograms. In [20], the fractal dimension was used for the same purpose. The author's goal was to provide a quantitative scale of risk for breast cancer

by analysing the mammographic parenchymal patterns. The study showed that the fractal dimension can be employed to distinguish between patterns associated with a high risk of breast cancer and those associated with a low risk. In [111], two texture analysis methods, namely morphometric granulometry and Laws' texture energy measures, were tested for the discrimination between glandular and fatty regions in mammograms. The identification of glandular tissue is essential both in assessing asymmetry between left and right breasts, which can be used as a cue for automated cancer detection, and in estimating the radiation risk associated with mammographic screening. Dhawan et al. ([45]) employed textural features from the SGLDM and the grey level difference method to characterise the global texture as well as features from the wavelet transform to characterise the local texture of the microcalcification area of interest in mammograms. The above two texture analysis methods (SGLDM and the grey level difference method) were again employed in [129] for the classification of mammograms into mass and normal breast tissue.



(a)



(b)

Figure **3**.1: Examples of mammograms (a) normal mammogram, (b) abnormal mammogram.

3.3 Ultrasonic images

The use of ultrasonography as an imaging modality has become widely spread because of its ability to visualise main organs with no deleterious effects. The basic idea of ultrasonic imaging is to send a fine beam of ultrasonic waves through the human tissue and then receive the characteristic echo reflections from the internal body structures to form the ultrasound image (see Figures **3**.2 (a) and (b)). The different grey levels of this image represent the acoustic properties of the various body structures such as attenuation of acoustic waves, speed of sound and acoustic impedance. All these factors contribute to the shape and intensity of the returned waves according to the underlying tissue properties and hence, this fact is the basis for the use of ultrasonography as an imaging technique. The main limitation for ultrasound is its inherent inability to visualise air-containing or bonny structures. This limitation does not apply for most of the abdominal body structures, as they are composed of soft tissues and blood vessels.

Although the processes by which ultrasound energy interacts with tissue to produce images are not completely understood, it is known that the images of tissues are made up of two components. The first component is due to specular reflections of ultrasonic energy from interfaces within the tissue. The second component is due to interference among the backscattered reflections from the many scattering elements representing the internal structure of organs and is visualised as a small scale, high contrast speckle pattern (texture) in the ultrasonic image. This pattern is due to the tissue volume being scanned, the transducer generating the ultrasonic pulse, the tissue lying between the transducer and scattering volume, and to a lesser extent, the signal processing applied to the data before display.

The ultrasonic properties of normal tissue are different from those of abnormal tissue due to disruption of the normal tissue architecture and this difference is reflected in the grey level image as different texture. However, visual criteria for diagnosing the various diseases, e.g. diffuse liver diseases, from ultrasonic images are in general confusing and highly subjective. They depend on the ability of the sonographer to observe certain characteristics from the texture in the image

and to compare them with those developed for various pathologies in order to determine the type of the disease. Moreover, some diseases are highly similar in their diagnostic criteria, which tend to confuse the sonographers even more. Therefore, physicians may resort to invasive methods which pose various risks.

To solve some of these problems in diagnosis, researchers have developed more quantitative criteria using computer-based systems. Several studies have suggested that the use of quantitative tissue characterisation could significantly increase the usefulness of ultrasound in the diagnosis of various diseases. One important feature for this purpose is texture.

Texture has been extensively employed in the analysis of ultrasonic liver images. In [125], three texture characterisation methods were used in the classification of liver tissue into normal, diffuse parenchymal (hepatitis, cirrhosis, and fatty infiltration), and malignant. Diffuse liver diseases are those in which at least one complete lobe of the liver is affected. The methods used were the grey level run length method, the SGLDM, and the power spectrum method. The study showed that the use of textural features results in a better overall classification accuracy than the one achieved through subjective analysis. According to [25], the intensity surface of an ultrasound liver image can be modelled as a fractal surface and therefore, it can be characterised by the fractal dimension. In this study, a high accuracy was reported in the classification of liver images into normal and abnormal tissue using this textural feature. Garra et al. ([59]) showed that the use of textural parameters can increase the accuracy in the classification of liver images into normal and diffuse liver disease (cirrhotic and fatty). In [177], the statistical feature matrix approach (see Section **2**.10.4) was used for the classification of ultrasonic liver images into normal liver, hepatitis, and cirrhosis. The authors claimed that the dissimilarity statistical feature matrix achieves better classification in less computational time than previous techniques.

In [178], the classification of ultrasonic liver images into three tissue types, namely normal liver tissue, hepatoma (hepatocellular carcinoma), and cirrhosis was investigated using various texture analysis methods. They found the SGLDM to be superior to the other examined conventional methods (Fourier power spectrum, grey level difference method, and Laws' texture energy measures). Sun et al.

([149]) employed four texture analysis methods to analyse liver parenchyma in ultrasonic liver images, namely the SGLDM, the statistical feature matrix method, the texture spectrum method (see Section **2**.10.1), and the fractal method. The authors reported that the SGLDM gave the best results. Finally, Kadah et al. ([82]) investigated several textural methods in the classification of ultrasound liver images into normal, fatty, and cirrhotic. These were the grey level run length method, the SGLDM, and the edge co-occurrence matrix (see Section **2**.8). They found the SGLDM to be the best method.

Texture has also been shown to be an important diagnostic feature for detecting abnormalities in the myocardium from echocardiograms. In [23], quantitative texture analysis was employed to the diagnosis of two cardiomyopathies; hypertrophic cardiomyopathy, which is a genetically inherited disorder characterised by altered myocardial microstructure and amyloid infiltration, which causes myocardial fibre necrosis. Two texture analysis methods were employed in this study, namely the grey level run length method and the grey level difference method. In [31], the same texture analysis methods were employed in order to test the hypothesis that echocardiographic image texture varies with cardiac contraction in normal human subjects. The authors claimed that this change in the image texture can be used in distinguishing normal from abnormal myocardium.

Texture has also been used in the analysis of other ultrasound image types. In [87], features from textural edgeness (see Section **2**.10.2), the grey level run length method, and the SGLDM were employed in the differentiation of ultrasound images of prostatic carcinoma from those illustrating benign prostatic hypertrophy. A high classification accuracy was reported. In [11], an attempt was made to classify ultrasonic images of prostatic tissue into normal, benign prostatic hypertrophy, and prostatic cancer. These three texture types closely resemble each other and therefore, it is very difficult to be discriminated by visual examination. In this study, features from the autocorrelation function, the grey level run length method, and the SGLDM were employed in the classification. The authors reported satisfactory results for the SGLDM. The same results were also reported in [12].

Sjögren's syndrome is an autoimmune disease of the parotid glands. In [4],

textural features from the Fourier power spectrum of ultrasound images of the parotid glands gave good results in distinguishing normal images from those with Sjögren's syndrome. Zuna ([182]) employed a number of textural features for the classification of ultrasonic thyroid images into normal, thyroidal carcinoma, and thyroidal cystic lesions. He reported that the features from the SGLDM and textural edgeness had the most discriminating power. Basset et al. ([10]) analysed ultrasonic images of the muscles of the thigh using the SGLDM, in order to measure the degree of muscle tiredness during a physical exercise. The study involved three categories of volunteers, i.e. high level sportsmen, mid level sportsmen, and non-sportive persons. In [114], the SGLDM was employed to characterise the tissue of *in vivo* human placentae from ultrasound images. The purpose was to detect any changes in placental appearance, which could be attributed to a change in placental structure, due to some external factor such as smoking. The results in this study showed the potential clinical usefulness of the SGLDM in tissue characterisation of human placentae from ultrasound images.

Texture characteristics in ultrasound images (speckle) are markedly influenced by the ultrasound imaging system settings. Texture analysis can thus be significantly improved, when the ultrasound signal is pre-processed to eliminate the tissue-independent factors. However, in [134] it was shown that the correlation feature of the SGLDM (see Section **4**.1) was stable with respect to these factors in the analysis of a number of liver and synthetic ultrasound images.

(a)

(b)

Figure 3.2: Examples of ultrasound images (a) liver, (b) spleen.

3.4 Magnetic resonance imaging

Magnetic Resonance Imaging (MRI) has received considerable attention in the medical field because it is a non-invasive, high resolution, and non-hazardous imaging modality. MR images are reconstructed from RadioFrequency (RF) signals generated by the body as a result of the interaction of RF magnetic fields and certain nuclei, when the body is in the presence of a strong static magnetic field. Hydrogen nuclei (protons), abundantly found in the human body as water and lipids, are most frequently used as probes in MRI. MRI can be used to produce cross-sectional tomographic images in transverse, coronal, sagittal, and oblique planes with excellent soft tissue differentiation (see Figures **3**.3 (a) and (b)). It has the potential to create high resolution images that may be extremely useful in the diagnosis and monitoring of a large number of pathological and developmental disorders.

However, recent studies showed that before such diagnoses are made, it is essential to quantitatively describe and characterise properties, such as texture, in MR images. Quantitative MRI is expected to offer a unique potential for *in vivo* tissue characterisation. In [41], it was claimed that the use of texture analysis in quantitative MR imaging is expected to improve non-invasive characterisation for many tissue types, such as brain, liver, and skeletal muscle.

The human brain has especially been a primary area of interest in clinical MR. The MR images of the brain show very good contrast between grey and white matter, CerebroSpinal Fluid (CSF), and other cerebral structures. Significant changes in the contrast between grey and white matter have been noticed in pathological conditions, such as in the presence of tumour, oedematous tissue, etc. The relative grey and white matter composition and interfaces are indicators of the process of myelination in developing children. It has also been hypothesised that there are differences in the size and convolutedness of the surface in the temporal lobe of the brain in dyslexia patients, which may potentially be seen in MR images.

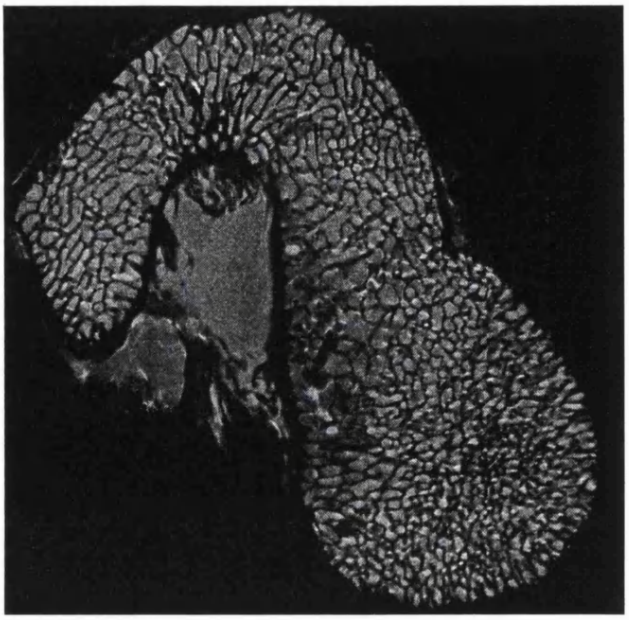
In [98], an attempt was made to characterise the texture present in MR images of the human brain. The second order textural features employed in this study

were derived from the SGLDM. The study showed that this method is able to capture the microtexture, which is essential for the discrimination between tumourous and oedematous tissue in MR brain images. The same conclusion was reached in [132]. In this study, features from several texture analysis methods were tested. Among them, there were textural features from the SGLDM, textural edgeness, and the grey level run length method. The best feature combination for the discrimination between the two abnormalities (tumour and oedema) included features from the first two methods. In [99], the Gaussian Markov random field model (see Section **2**.3) was tested in the segmentation of white and grey matter in MR brain images giving good results. According to the author, many other research groups were investigating this particular model in the automatic tissue classification and segmentation of MR brain images.

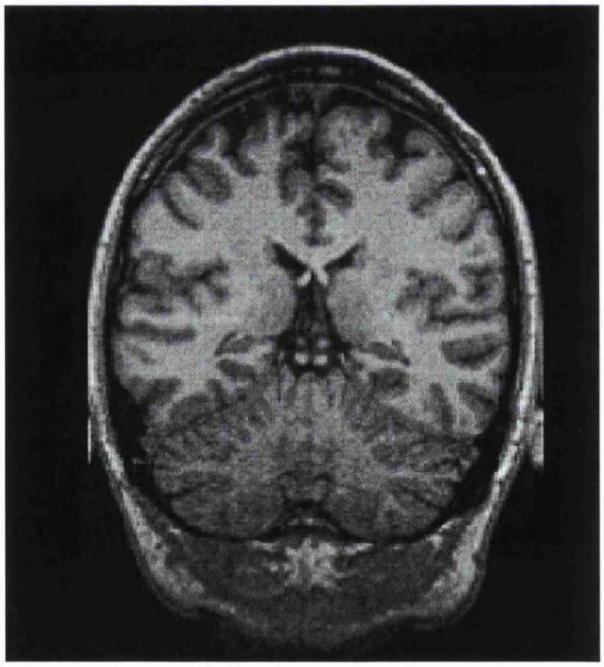
Another important application of texture analysis in MRI is the characterisation of the trabecular bone structure. In [28], the texture in high resolution MR images of the trabecular bone of the wrist was analysed by applying grey level morphological granulometry using flat-top structuring elements of various sizes (see Section **2**.6). Moments such as mean, variance, and skewness, were derived from the local pattern spectrum at each pixel and employed in the segmentation of the MR images into normal and osteoporotic trabecular bone. Very promising results were reported indicating the potential usefulness of the method in detecting the early stages of the disease. Similar results were reported in [72] for the classification of MR *in vivo* images of the wrist using the same method (morphological granulometry). In addition, the grey level run length method was applied to the analysis of MR *ex vivo* images of vertebrae. It gave promising results in assessing the size of marrow spaces and the trabeculae thickness from vertebrae images. These parameters comprise very important information for the physicians because they reflect bone quality.

Fortin et al. ([53]) employed the fractal parameter H (see Section **2**.5) in the segmentation of cardiac MR images. The goal of this study was to extract the boundary between the ventricle walls and the blood pool surrounded by the ventricle wall. A proper segmentation is very important, because it allows the automatic determination of several important diagnostic parameters, such as the

instantaneous muscle mass and blood volume. However, this task is complicated by the presence of the papillary muscles extending into the blood pool from the ventricle wall. The authors claimed that texture analysis yields information sufficient to minimise such complications.



(a)



(b)

Figure **3**.3: Examples of MR images (a) femur image, (b) brain image.

3.5 X-ray computed tomography

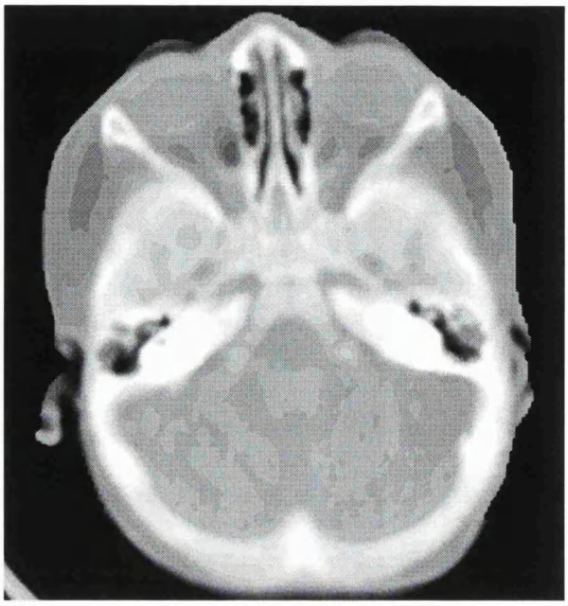
The objective of CT scanning is to take a large number of one dimensional views of a two-dimensional transverse axial slice (usually transverse scanning is employed) from many different directions and reconstruct the structure within the slice (see Figures **3**.4 (a) and (b)). X-rays are used for the production of the one-dimensional projections. The good resolution of CT images can capture sufficient information for the detection of changes in normal tissue architecture and enables the accurate segmentation of various structures in the body in many cases.

According to [112], texture is an essential feature in the analysis of CT images. It reflects, in a large degree, the changes in functional characteristics of various organs at the onset of disease. The authors investigated the use of texture for the detection of abnormalities in CT liver images, which are beyond human appreciation and also difficult to determine by other classical methods of image processing. In this study, the SGLDM, the grey level run length method, and the grey level difference method were employed. Three classes were used in the classification of the CT liver images: normal liver images, malignant liver images with the malignancy clearly visible, and malignant liver images with the malignancy not visible. The results showed that texture analysis of CT images is successful in identifying the onset of disease in liver tissue, which cannot be seen even by trained human observers. Especially, features from the SGLDM and the grey level run length method succeeded in discriminating the class with the non-visible malignancy and thus, in detecting the onset of liver malignancy. In [30], textural features from the Fourier power spectrum method were employed in the classification of CT liver images into normal and cirrhotic liver. This preliminary study gave very encouraging results.

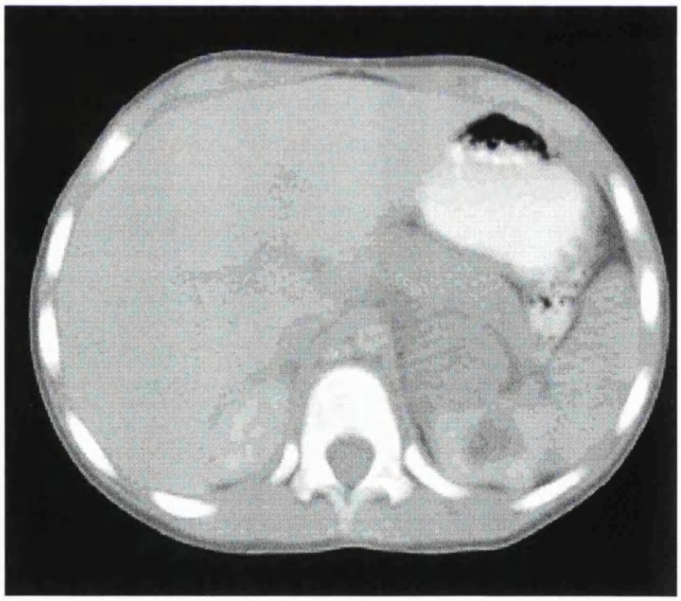
In [19], CT images of white/grey matter were analysed using first and higher order statistical features. Among them, there were textural features derived from the SGLDM. They measured local texture properties and were used for the separation of white from grey matter. This separation is vitally important for the evaluation of cerebral blood flow, neurological development of the neonatal brain, and for studies of hydrocephalus and atrophy. Also, Straughan et al. ([148])

employed the SGLDM in the classification of CT lung images into normal, benign, and malignant tissue. A very high classification accuracy was reported. In addition, in [139] fractal parameter H was employed in the classification of CT lung images into non-fibrotic lung disease (e.g. pneumonia and alveolar hemorrhage) and fibrotic lung disease (diffuse alveolar damage). The authors reported encouraging results in this preliminary study.

Finally, quantitative computed tomography has been employed to assess bone density, which is used to quantify the degree of osteoporosis and to monitor the effectiveness of treatment regimens. Durand and Rüegsegger ([50]) employed the run length method to distinguish between normal and osteoporotic patients, analysing CT images of the tibia bone. The authors claimed that this method is able to detect the trabeculae and capture significant information about them, such as their size and orientation and therefore, it results in better discrimination than other methods, such as the Fourier method.



(a)



(b)

Figure 3.4: Examples of CT images (a) brain image, (b) abdomen image.

3.6 Other biomedical applications

An important use of texture characterisation methods is in cell image analysis. The analysis of the Electron Micrograph (EM) image of a cell by a cytopathologist generally involves qualitative analysis of the individual cell and more specifically, of its nuclear structure. It can yield information about the cell, such as whether the cell is malignant or benign. In the examination, the cytopathologist looks for features such as arrangement, size, and distribution of the chromatin particles. However, it is difficult for a human observer to identify the boundary between different nuclear regions, because the grey levels usually change gradually. Using computer analysis techniques, such as texture analysis, EM images can be characterised in a quantitative manner and then segmented automatically without human intervention.

In [22], texture analysis was applied to the partitioning of the nuclei in EM images of cells. Two approaches were adopted, namely the fractal method and the SGLDM. According to the author, the performance of the latter method was the best in characterising the texture in the EM images, which is much more difficult to characterise than natural textures. In [180], statistical texture analysis was employed for the description of the chromatin structure in cell nuclei of neoplastic liver cells of the rat. The goal was to differentiate them from normal liver cells. Also, in [172] textural features from the Fourier power spectrum of the cell images were derived for the characterisation of the chromatin structure. Peet and Sahota ([120]) employed textural features in the classification of the nuclei of cells of two insect populations.

Thiran and Macq ([160]) employed texture analysis for the characterisation of the internal structure of the nuclei of healthy and malignant cells from the lungs and the digestive tract. The method used was based on mathematical morphology and succeeded in capturing the coarse granularity and irregular clumping of the chromatin in the nuclei of cancerous cells. Terzopoulos and Zucker ([158]) described an automated system for detecting Osteogenesis Imperfecta (OI), an inheritable disorder of human connective tissue. Researchers discovered that OI cells cultured *in vitro* exhibit a morphological defect when compared with

normal cells. The shape of normal cells allows them to appear in regular patterns, whereas the OI cells have irregular shapes presenting a rough appearance. In this study, textural features extracted from the SGLDM and the edge co-occurrence matrix method (see Section **2**.8) were employed in the discrimination between images of normal and OI cell cultures. According to the authors, the classification accuracy of the proposed system outperforms that achieved by the specialists.

In [70], a digital image analysis procedure was described for the detection of Senile Plaques (SP) and measurement of SP size, shape, and total fractional area in digital micrographs of silver-stained tissue sections. Senile Plaques are extracellular lesions that appear in large numbers in the human brain, in the Alzheimer's disease. This disease is a chronic degenerative disease of the central nervous system, producing progressive loss of memory, cognitive function, and ultimately death. Senile Plaques, which can be stained by silver compounds, are detected in digital micrographs and if observed in sufficient densities, are diagnostic for this disease. In the proposed automatic analysis, textural features, among other properties, were used to distinguish the SP objects from the non-SP objects which are also stained by the silver compounds (e.g. blood vessels, cell nuclei, and bundles of axonal fibres). The authors employed the SGLDM for this purpose.

Finally, another important application of texture analysis is in the characterisation of nuclear medicine images. According to [18], texture analysis can contribute significantly to the detection of various pathological diseases from nuclear medicine images of the liver, thyroid, lungs, and kidneys. In this study, textural features from the SGLDM were employed.

Chapter 4

Spatial Grey Level Dependence Method

4.1 Description

The Spatial Grey Level Dependence Method (SGLDM) is a texture analysis method which characterises texture in an image region by means of features derived from the spatial distribution of pairs of grey levels (second-order distribution) having certain interpixel distances (separations) and orientations. It is based on the assumption that texture information is contained in the overall spatial relationship which the grey levels in an image region have to one another.

According to [67], this method is widely used in statistical texture analysis. Conners et al. ([35]) claimed that there is no known visually distinct texture which cannot be discriminated by the SGLDM. The experiments of Weszka et al. ([176]) showed that this method extracts much more textural information from an image region than the autocorrelation function. According to [167], there has been no study that clearly indicates the superiority of multichannel filtering approaches over more traditional ones, such as the SGLDM. Basset et al. ([11]) claimed that the SGLDM is a powerful method that is widely used in texture analysis.

In [5], co-occurrence matrices showed comparable performance with features derived from the Fourier power spectrum in the classification of natural textures.

In [54], the SGLDM was employed for the analysis of digital image data from a mountainous area of Canada. In [64], [65], the SGLDM was employed in the analysis and classification of photomicrographs of reservoir rocks. The analysis and characterisation of the pore structure of these rock types is important to geologists and petroleum engineers. Hepplewhite and Stonham ([69]) claimed that the SGLDM gave very good results in the classification of defect and non-defect areas of a disk surface for quality control in magnetic disk production.

Texture has also been shown to be a very important feature in remote sensing applications involving analysis, classification, and segmentation of images acquired using various satellites and airborne platforms. In [9], it was shown that the SGLDM has better discrimination power than the grey level run length method and the Fourier power spectrum method, in the classification of remote sensed data. Similarly, Augusteijn et al. ([6]) found the SGLDM to be superior to other texture analysis methods for the classification of various ground types in satellite images. Lee et al. ([95]) claimed that the SGLDM is capable of discriminating various cloud types in images acquired by satellites. Gong et al. ([60]) claimed that this method can largely improve land-use classification accuracy in such images. According to [42], remote sensed image texture has a large variety of appearances and the SGLDM has the ability to characterise such diverse textures. In [106], it was shown that the accuracy in land-cover classification of satellite images is considerably improved when texture features are included in the feature set. The SGLDM was employed in this study. Haralick et al. ([65]) reported a high accuracy by employing the SGLDM in the classification of panchromatic aerial photographs and satellite images containing various land-cover categories.

Specifically, a number of studies have investigated the use of textural features in the analysis of Synthetic Aperture Radar (SAR) images. A SAR image provides information about the surface of a target by measuring and mapping the reflected energy. The spatial variability in the scattering properties of the ground areas that constitute a target, gives rise to a specific texture in the SAR image. Holmes et al. ([71]) employed the SGLDM for real time analysis and identification of different types of sea ice in SAR images. Data gathered from the analysis of these image types are invaluable for the assessment and monitoring of climatic changes in the

polar regions. In [136], the same texture analysis method was employed in the classification of large scale geological formations from SAR images. According to the authors, the texture features separated the different classes very well. In [116], SAR images were analysed for automatic crop identification. The SGLDM was employed for texture analysis. According to the author, there is strong supportive evidence that this method is representative of the best texture algorithms. In this study, it gave satisfactory crop separability, which was not improved when tonal features were added to the feature set employed for classification. In [124], an evaluation of the role of the texture features in SAR image classification of agricultural areas was performed. Features from the SGLDM were employed in this preliminary study and it was shown that they increased the classification accuracy compared to that achieved by using only tonal features. Finally, Barber and LeDrew ([8]) employed the SGLDM in the discrimination of sea ice classes from SAR images. A satisfactory accuracy was reported.

In the following, we present the Spatial Grey Level Dependence Method in a formal mathematical way. Suppose that $N_x, N_y \subset \mathcal{N}$, where $\mathcal{N}$ is the set of all positive integer numbers including 0, $N_x = \{0, 1, 2, \ldots, n_x - 1\}$, and $N_y = \{0, 1, 2, \ldots, n_y - 1\}$ ($n_x = |N_x|$, $n_y = |N_y|$). Also, let $G \subset \mathcal{N}$, where $G = \{0, 1, 2, \ldots, n_g - 1\}$, ($n_g = |G|$). Then, a digital image I is a function which maps a pair $(i, j) \in N_x \times N_y$ onto set G, i.e. $I : N_x \times N_y \rightarrow G$. N_x is called the x spatial domain of I, N_y is called the y spatial domain of I and the Cartesian product $N_x \times N_y$ is called the spatial domain of I. G is the set of all grey level values of I. A digital image I can be considered as a realisation of a discrete-space discrete-state stochastic (or random) process $\mathcal{I}$ (see [119]). For the determination of the properties (statistical properties) of $\mathcal{I}$, knowledge of the distribution functions (nth order distributions)

$$F_n(g_1, g_2, \ldots, g_n; i_1, i_2, \ldots, i_n) = Prob\{\mathcal{I}(i_1) \leq g_1, \mathcal{I}(i_2) \leq g_2, \ldots, \mathcal{I}(i_n) \leq g_n\} \tag{4.1}$$

is required for every $i_k \in \{0, 1, \ldots, n_x n_y - 1\}$, where $1 \leq k \leq n$ and for every n, where $1 \leq n \leq n_x n_y$. In the above equation $\mathcal{I}$ is considered a 1-D process.

Texture depends on the spatial (statistical) distribution of the grey levels in an image ([65]). Therefore, the above nth order distribution functions F_n completely characterise the texture in the given image. However, in practical situations of texture characterisation a small n is adequate. Experiments on human visual perception indicated that second-order probabilities (spatial distribution of pairs of grey levels) play an important role in human texture discrimination ([80], [81]). Gagalowicz ([57]) suggested that the human eye cannot discriminate two texture fields, which have locally the same second-order statistics. If we consider $n \in \{1, 2\}$, then texture is characterised by the distribution functions up to the 2nd order and the equation (4.1) becomes:

$$F_1(g_1; i_1) = Prob\{\mathcal{I}(i_1) \leq g_1\} \tag{4.2}$$

$$F_2(g_1, g_2; i_1, i_2) = Prob\{\mathcal{I}(i_1) \leq g_1, \mathcal{I}(i_2) \leq g_2\} \tag{4.3}$$

In practice, only F_2 is considered since F_1 can be derived from F_2. Also, $F_2 \equiv F$ is simplified even more by considering stochastic processes which are strict-sense stationary, that is, their statistical properties are invariant to a shift of the origin (homogeneous processes). Conners et al. ([33]) assumed that texture is generated by a translation stationary (strict-sense stationary) stochastic process in their theoretical evaluation. That in turn means that

$$F(g_1, g_2; i_1, i_2) = F(g_1, g_2; i_1 + c, i_2 + c) \tag{4.4}$$

for any integer c for which $0 \leq i_1 + c, i_2 + c < n_x n_y$.

If we alternatively characterise the stochastic process by means of its probability density function f, where $f(g_1, g_2; i_1, i_2) = Prob\{\mathcal{I}(i_1) = g_1, \mathcal{I}(i_2) = g_2\}$, we get that

$$f(g_1, g_2; i_1, i_2) = f(g_1, g_2; i_1 + c, i_2 + c) \tag{4.5}$$

for any c defined as above. From the strict-sense stationarity assumption of the stochastic process it follows that

$$f_1(g_1; i_1) = Prob\{\mathcal{I}(i_1) = g_1\} = f_1(g_1) \tag{4.6}$$

independent of i_1 and

$$f_2(g_1, g_2; i_1, i_2) = f(g_1, g_2; i_1, i_2) = f(g_1, g_2; \tau) \tag{4.7}$$

where $\tau = i_2 - i_1$ and $|\tau| < n_x n_y$, f_1 is the probability density function of F_1, and f_2 is the corresponding function of F_2. Equation (4.6) means that the stochastic process has the same first-order statistics independent of the pixel position. Equation (4.7) means that the second-order statistics depend only on the distance of two pixels in 1-D space and not on their exact positions.

In 2-D space, distance τ translates into a vector $\overline{d} = (d_x, d_y)$ (displacement vector), where d_x, d_y are integers which correspond to separation along the x- and y- axis, respectively. Equivalently, displacement vector $\overline{d}$ can be expressed in polar co-ordinates as (d, ϑ), where $d = \pm(d_x^2 + d_y^2)^{\frac{1}{2}}$ is the interpixel distance and $\vartheta = \tan^{-1}\left(\frac{d_x}{d_y}\right)$ is the orientation relative to the horizontal axis in the clockwise direction. In this thesis, we assume that the origin is on the top left corner, the x-axis represents the rows and the y-axis represents the columns of the 2-D discrete space. Note that d and ϑ are not always integers. Therefore, in practical implementations the form $\overline{d} = (d_x, d_y)$ is usually preferred. The probability density function given by (4.7) can be rewritten as

$$f(g_1, g_2; \tau) = f(g_1, g_2; d, \vartheta) \tag{4.8}$$

The SGLDM estimates the above second-order probability density functions by relative frequencies measured in the image:

$$f(g_1, g_2; d, \vartheta) \simeq p(g_1, g_2; d, \vartheta) =$$

$$\frac{\#\{((k,l),(m,n)) \in (N_x \times N_y)^2 : I(k,l) = g_1, I(m,n) = g_2, \mathcal{D}((k,l),(m,n)) = d, \mathcal{O}((k,l),(m,n)) = \vartheta\}}{N(d, \vartheta)} \tag{4.9}$$

where $\mathcal{D}()$ and $\mathcal{O}()$ are functions that give the distance and orientation of a pair of pixels, respectively and $N(d, \vartheta)$ is the number of pixel pairs in I, having distance d and orientation ϑ. The estimation given by (4.9), stems from the ergodicity assumption for the underlying texture generating stochastic process ([119]). In practice, a small number of distances d and orientations ϑ are used to characterise a given texture. Often $|d| \leq 2$ and $\vartheta \in \{0°, 45°, 90°, 135°\}$. Also, no distinction

may be drawn between positive and negative distances. In this case d is always positive and $p(g_1, g_2; d, \vartheta) = p(g_2, g_1; d, \vartheta)$. That is, if we store the estimated probabilities for a specific distance and orientation in matrix form, the resulting matrix will be symmetric. However, some valuable information may be lost in the symmetric approach.

One important aspect of the SGLDM is the choice of the interpixel distance or distances that best capture the texture content of an image. According to [158], the usefulness of texture representations is proportional to the amount of structure that they capture. Thus, emphasis must be given to the spatial relations employed in the SGLDM. The above choice is application dependent. For example, in [114] it is suggested that in ultrasonic image analysis the interpixel distance must be set at a point where further increase lets one texture feature like contrast (see below) have an almost constant value. The author claims that for distances smaller than this point texture features are influenced by the properties of the ultrasound imaging system more than the properties of the tissue being scanned. In [65], a general way of finding an upper bound on the distance is suggested. This upper bound can be the point where the autocorrelation function of an image becomes too small. In [181], another general method for determining the best interpixel distance is proposed. The estimated second-order probabilities for a specific spatial relationship are considered to form a contingency table. If the texture is not random (i.e. there is some sort of structure) but the estimated probabilities for this spatial relationship do not capture the underlying structure, the rows and columns of this table are independent. The χ^2 (chi square) significance test is employed as a measure of independence of the rows and columns. If the statistical hypothesis that they are independent cannot be rejected, the estimated probabilities for the specific interpixel distance do not capture the underlying structure well.

As far as the orientation is concerned, an averaging of the texture features over the four basic directions $(0°, 45°, 90°, 135°)$ is usually performed, in order to get rotation invariant features regarding these directions. This property is very desirable in many cases in texture analysis and in image processing in general. Also, the size of the image region to be analysed is of great importance. If the

region is too small it may not have enough information to adequately characterise the given texture. If on the other hand it is too large, it may contain more than one texture. There are no general rules as to which region size is more appropriate, since this is application dependent.

After the second-order probabilities are estimated, a number of texture features proposed in [65] can be derived to characterise a given texture (the definition of the parameters used in the formulas below follow them). These are:

1. Angular Second Moment:

$$f_1 = \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} \{p(i,j)\}^2 \tag{4.10}$$

2. Contrast:

$$f_2 = \sum_{n=0}^{n_g-1} n^2 \cdot \sum_{\substack{i=0 \\ |i-j|=n}}^{n_g-1} \sum_{j=0}^{n_g-1} p(i,j) \tag{4.11}$$

3. Correlation:

$$f_3 = \frac{\sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} (ij) \cdot p(i,j) - \mu_x \cdot \mu_y}{\sigma_x \sigma_y} \tag{4.12}$$

4. Sum of Squares (Variance):

$$f_4 = \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} (i - \mu_x)^2 \cdot p(i,j) \tag{4.13}$$

5. Inverse Difference Moment:

$$f_5 = \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} \frac{1}{1 + (i-j)^2} \cdot p(i,j) \tag{4.14}$$

6. Sum Average:

$$f_6 = \sum_{i=0}^{2(n_g-1)} i \cdot p_{x+y}(i) \tag{4.15}$$

7. Sum Entropy:

$$f_7 = -\sum_{i=0}^{2(n_g-1)} p_{x+y}(i) \cdot \log\{p_{x+y}(i)\} \tag{4.16}$$

8. Sum Variance:

$$f_8 = \sum_{i=0}^{2(n_g-1)} (i - f_6)^2 \cdot p_{x+y}(i) \tag{4.17}$$

9. Entropy:

$$f_9 = -\sum_{i=0}^{n_g-1}\sum_{j=0}^{n_g-1} p(i,j)\cdot\log\{p(i,j)\} \tag{4.18}$$

10. Difference Variance:

$$f_{10} = \sum_{i=0}^{n_g-1}(i-\mu_{x-y})^2\cdot p_{x-y}(i) \tag{4.19}$$

11. Difference Entropy:

$$f_{11} = -\sum_{i=0}^{n_g-1} p_{x-y}(i)\cdot\log\{p_{x-y}(i)\} \tag{4.20}$$

12. First Information Measure of Correlation:

$$f_{12} = \frac{H_{xy}-H_{xy}^1}{\max\{H_x,H_y\}} \tag{4.21}$$

13. Second Information Measure of Correlation:

$$f_{13} = \left(1-\exp\{-2.0\cdot(H_{xy}^2-H_{xy})\}\right)^{\frac{1}{2}} \tag{4.22}$$

14. Maximal Correlation Coefficient:

$$f_{14} = \sqrt{\lambda_2} \tag{4.23}$$

where $p(i,j)$ is the second-order probability for grey level pair (i,j) and for a specific distance d and orientation ϑ (estimated by (4.9)),

$$p_x(i) = \sum_{j=0}^{n_g-1} p(i,j) \tag{4.24}$$

$$p_y(j) = \sum_{i=0}^{n_g-1} p(i,j) \tag{4.25}$$

$$p_{x+y}(k) = \sum_{\substack{i=0\\ i+j=k}}^{n_g-1}\sum_{j=0}^{n_g-1} p(i,j),\ k=0,\ldots,2(n_g-1) \tag{4.26}$$

$$p_{x-y}(k) = \sum_{\substack{i=0\\ |i-j|=k}}^{n_g-1}\sum_{j=0}^{n_g-1} p(i,j),\ k=0,\ldots,n_g-1 \tag{4.27}$$

$$\mu_x = \sum_{i=0}^{n_g-1} i\cdot p_x(i) \tag{4.28}$$

$$\mu_y = \sum_{j=0}^{n_g-1} j \cdot p_y(j) \tag{4.29}$$

$$\sigma_x = \left\{ \sum_{i=0}^{n_g-1} (i-\mu_x)^2 \cdot p_x(i) \right\}^{\frac{1}{2}} \tag{4.30}$$

$$\sigma_y = \left\{ \sum_{j=0}^{n_g-1} (j-\mu_y)^2 \cdot p_y(j) \right\}^{\frac{1}{2}} \tag{4.31}$$

$$\mu_{x-y} = \sum_{k=0}^{n_g-1} k \cdot p_{x-y}(k) \tag{4.32}$$

$$H_{xy} = -\sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} p(i,j) \cdot \log\{p(i,j)\} \tag{4.33}$$

$$H_{xy}^1 = -\sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} p(i,j) \cdot \log\{p_x(i) \cdot p_y(j)\} \tag{4.34}$$

$$H_x = -\sum_{i=0}^{n_g-1} p_x(i) \cdot \log\{p_x(i)\} \tag{4.35}$$

$$H_y = -\sum_{j=0}^{n_g-1} p_y(j) \cdot \log\{p_y(j)\} \tag{4.36}$$

$$H_{xy}^2 = -\sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} p_x(i) \cdot p_y(j) \cdot \log\{p_x(i) \cdot p_y(j)\} \tag{4.37}$$

λ_2 is the second largest eigenvalue of matrix Q, where $Q(i,j) = \sum_{k=0}^{n_g-1} \frac{p(i,k) \cdot p(j,k)}{p_x(i) \cdot p_y(k)}$ and the log function is base 10. Angular second moment is also called energy. Contrast is called inertia as well. Inverse difference moment is alternatively called local homogeneity.

According to [65], it is difficult to identify which specific textural characteristic each of the above texture features represents. Angular second moment reaches values close to its maximum value 1 when either the image region is homogeneous or presents a periodic structure. It is smallest when the estimated second-order probabilities are all as equal as possible. It is large when some probabilities are high while others are low. If we consider that the estimated probabilities form a

matrix (co-occurrence matrix), then contrast is essentially the moment of inertia of the matrix around its main diagonal. It measures the degree of spread of the matrix values. It is sensitive to large local grey level differences. It also characterises the spatial frequencies of a textured region, i.e. a low contrast texture indicates low spatial frequencies. Correlation is considered to be a measure of grey level linear dependencies in a textured region and therefore, it characterises the macrotexture in an image. Entropy gets the largest value when the estimated probabilities are equal. When they are very unequal its value is small. We can say that entropy measures the disorder in a textured region. If a region contains completely random values of grey levels (white noise), then this region has maximum entropy. For a uniform texture entropy takes small values. Both angular second moment and entropy measure texture uniformity although entropy is inversely correlated to energy. Inverse difference moment takes larger values for smaller grey level differences in pairs of pixels. Therefore, it is more sensitive to low contrast textures. It measures the similarity of grey levels in the pixel pairs. Variance is a measure of heterogeneity. It increases when the grey levels differ largely from their mean.

From all the texture features defined above only the first 13 are of interest, since the last one (maximal correlation coefficient) is very time consuming and is usually not implemented. In practice, a subset of them is used for texture description, mainly due to the high computational cost of the co-occurrence matrices employed in the SGLDM. However, a number of researchers have used the estimated second-order probabilities from a textured region directly as texture features ([165], [170]). In this way, they claimed that one saves the time needed for computing global measures over these probabilities. Other researchers proposed additional features for the SGLDM. They claimed that these features along with the features in [65], capture more of the textural information contained in the estimated second-order probabilities.

In [35], two new texture features are defined:

- Cluster Shade:

$$f_1 = \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} (i + j - \mu_x - \mu_y)^3 \cdot p(i,j) \tag{4.38}$$

- Cluster Prominence

$$f_2 = \sum_{i=0}^{n_g-1} \sum_{j=0}^{n_g-1} (i + j - \mu_x - \mu_y)^4 \cdot p(i,j) \tag{4.39}$$

(for the definition of the parameters used in the above formulas see page 96). The above two features are believed to measure the uniformity and proximity textural characteristic (the perceptual characteristic corresponding to clusters or lines formed by proximate points of uniform brightness).

In [67], three new texture features based on the SGLDM are proposed. The texture features are:

1. Diagonal Moment:

$$D = \frac{\sum_{i=0}^{n_g-1} i \cdot p(i,i)}{\sum_{i=0}^{n_g-1} p(i,i)} \tag{4.40}$$

2. High Level Moment:

$$H = \frac{\sum_{j=D}^{n_g-1} (j - D) \cdot p(D,j)}{\sum_{j=D}^{n_g-1} p(D,j)} \tag{4.41}$$

3. Low Level Moment:

$$L = \frac{\sum_{j=0}^{D} (D - j) \cdot p(j,D)}{\sum_{j=0}^{D} p(j,D)} \tag{4.42}$$

The above texture features need only a small subset of the estimated second-order probabilities for their computation and consist of simple arithmetic operations. Therefore, they are computationally cheap. The authors claimed that using only these features, a reasonable accuracy is achieved and the computational time is largely reduced.

In our research work, we employ all the first thirteen texture features defined in [65]. These features are functions of the probabilities $p(i,j)$, $p_x(i)$ (given by (4.24)), $p_y(j)$ (given by (4.25)), $p_{x+y}(k)$ (given by (4.26)), and $p_{x-y}(k)$ (given by

(4.27)). So, in order to efficiently implement the SGLDM, we need to find a fast way to estimate the above probabilities. We also need to find an efficient method to store the estimated second-order probabilities, since the co-occurrence matrix, as we will see in Section 4.2, is inappropriate for this purpose. This method must permit the storage of the probabilities in a digital computer memory (main memory), without losing the information contained in the co-occurrence matrices.

4.2 Disadvantages of the co-occurrence matrix approach

Although the SGLDM is one of the best general texture analysis methods and has been widely used for texture characterisation, the employment of the co-occurrence matrices for storing the estimated second-order probabilities has a lot of disadvantages. These disadvantages limit its applicability and prevent the extraction of all the texture information from a given image that can be captured by the SGLDM. A lot of authors have referred to those disadvantages which are the memory requirements of the co-occurrence matrix approach and the computational time for calculating textural features from these matrices. Many of the researchers who used this method, were forced to reduce the grey level range of the image, so that they could handle the large requirements of the co-occurrence matrices. Doing so, however, they lost significant texture information in many cases, which could have led to better texture discrimination. Others employed a limited set of texture features to reduce the computational time complexity. Still others completely abandoned this method due to these requirements and shifted towards other texture analysis methods which, though, were shown not to be equally powerful.

In [3], it was recognised that the SGLDM is quite effective for the discrimination of different textures and that it has been applied to the problem of classification and segmentation of artificial and natural scenes with success. However, the authors said that the high computational cost of the co-occurrence matrix is a significant drawback. Baraldi and Parmiggiani ([7]) claimed that it may be

reasonable to use the grey level difference method (see Section **2**.1) instead of the SGLDM, in order to benefit from a better compromise between texture assessment accuracy, memory storage requirements, and computational cost. In [67], it was mentioned that one way to reduce the computational time of the SGLDM is to use a smaller number of grey levels to code the digitised images. But then, a large amount of information may be lost and the discrimination rate may get much worse. The authors claimed that analysing images with a large dynamic range using this method results in unacceptable computational times. For this reason, the SGLDM is inappropriate for such images.

In [170], the authors suggested a reduction in the number of grey levels N_g of the analysed images, when the SGLDM is employed. The reason is the high computational cost using the co-occurrence matrix, which is proportional to N_g^2. However, according to the authors, this leads to degraded classifier performance. A similar reduction was performed in [158] in a biomedical application (see Section **3**.6). The authors had to reduce the grey level range from 64 grey levels to 16 grey levels, in order to get reasonable computational time. Wu and Chen ([177]) mentioned that the large amount of time required by the co-occurrence matrix approach represents a severe drawback that limits its applicability. In [173], it was claimed that a reduction in the number of grey levels via histogram equalisation techniques is usually a necessary pre-processing step for computational efficiency, when using the SGLDM.

According to [178], low computational complexity and high classification accuracy are two of the most important considerations for the classification of ultrasonic liver images. The SGLDM is one of the worst in terms of speed, since its computational complexity depends on the size of the co-occurrence matrix, or else, the number of grey levels in the analysed image. Thus, one way to reduce this complexity is to use a smaller number of grey levels to code the image. However, the authors implied that this leads to loss of information and reduced classification accuracy. Also, another major problem is the storage requirements of the co-occurrence matrix. For a digitised image with 2^n grey level resolution, the size of the co-occurrence matrix is 4^n. Unser ([166]) claimed that, although the second-order probabilities of the SGLDM capture most of the visually useful

textural information, the requirement for large memory makes the above method computationally inefficient. This is due to the use of the co-occurrence matrices, which depend on the second power of the total number of grey levels.

In [108], it was mentioned that images are typically quantised into a small number of grey levels, when analysed using the SGLDM. The reason is computational efficiency. Hepplewhite and Stonham ([69]) claimed that the co-occurrence matrix approach suffers from cost of memory space, since its size is dependent upon the number of grey levels in the image. Grey level reduction is a solution to the above problem but suffers from information loss. In [44], the problem of the memory space requirements of the co-occurrence matrix was reported. An effort to minimising these requirements was attempted by first quantising an image into a smaller number of grey levels and then retaining the entries of the co-occurrence matrices that gave the largest co-occurrence frequencies for each interpixel distance and orientation. These entries were further combined to reduce storage requirements even more. A significant reduction in memory was achieved. However, a significant information loss happened, too. A grey level reduction was performed in [62] as well, in order that the dimensions of the co-occurrence matrix become manageable. In [114], it was stated that the size of the co-occurrence matrix is proportional to N_g^2, where N_g is the number of grey levels in the analysed image. In order to save on computer memory requirements the author had to reduce the number of grey levels from 256 to 32, prior to ultrasonic image analysis.

Lerski et al. ([98]) claimed that although the SGLDM is one of the best methods for the texture analysis of MR images, it is usually necessary to limit the image grey level values for saving memory and computational time. They claimed that if an 8-bit image is analysed, that implies a 256×256 co-occurrence matrix which is very sparse because of the many zero entries. This is actually the reason why the co-occurrence matrix creates so many problems. Zero entries contribute nothing to the computation of the textural features. Therefore, they are actually redundant information. In [54], it was referred that the co-occurrence matrix may have many zero entries. Zucker and Terzopoulos ([181]) claimed that in the analysis of real texture pictures one or more allowable grey levels may never occur.

This causes entire rows and (or) columns in co-occurrence matrices to have zero entries.

Wu et al. ([179]) excluded the SGLDM from their forest inventory study of an area in southern China using satellite images, because of its computational complexity. According to Jensen ([79]), the computational burden imposed by the SGLDM is a significant problem in the utilisation of this method in the analysis of satellite images. Nüesch ([116]) claimed that the computational cost of analysing a single 1024×1024 SAR image using the SGLDM was tremendous. Desachy ([42]) reported that the SGLDM suffers from heavy cost in memory space. Reduction in the grey level range of satellite images is mandatory for analysis. According to Augusteijn et al. ([6]), the calculation of co-occurrence measures from images consisting of 256 grey levels is extremely slow. The computational time can be reduced through a reduction in the number of grey levels. However, the effect of this reduction on classification performance was not investigated in this study. Welch et al. ([174]) claimed that the principal drawback of the SGLDM is its large memory and computational time requirements. The same fact reported in [95].

Miller and Astley ([111]) analysed mammograms having a 6 bit dynamic range. They claimed that higher grey level resolution would reveal more subtle textural details and lead to better discrimination of the various tissues in breast parenchyma. According to [22], the CPU time needed by the SGLDM is its largest drawback in its utilisation in the analysis of EM images of cells. Barber and LeDrew ([8]) had to reduce the dynamic range of SAR images from 8 bits (256 grey levels) to only 4 bits (16 grey levels), in order to get an acceptable computational time using the SGLDM. Similarly, Pultz and Brown ([124]) had to reduce the dynamic range of SAR images from 8 bits to 6 bits for computational efficiency in their analysis using the same method.

The conclusion from all the above references is that, in order to deal with the significant drawbacks of the co-occurrence matrix, we need to find a way to eliminate the zero entries. These entries not only contribute nothing to the texture discrimination process, but also create all these problems reported in the previous paragraphs for the SGLDM, in terms of computational time and memory space requirements. The only efficient way to perform this elimination is to replace

the co-occurrence matrix with a number of dynamic data structures, such as the binary trees, which have the ability to completely eliminate redundancy.

Chapter 5

Dynamic Data Structures

5.1 Introduction

Data structures constitute the core of every algorithm. Finding efficient data structures in memory space and computational time is one of the principal aims in computer science research. The reason is that the efficiency of the algorithms is strongly dependent on the data structures being employed.

Before we begin our discussion about dynamic data structures, we need to introduce the notation that is used throughout the remainder of this thesis, for the expression of the time and space complexities of algorithms. The notation is the following:

- $f(N) = O(g(N))$ is used to denote the fact that there exist constants c and N_0, such that $f(N) \leq c \cdot g(N)$, for all $N \geq N_0$.

- $f(N) = \Omega(g(N))$ is used to denote the fact that there exist constants c and N_0, such that $f(N) \geq c \cdot g(N)$, for all $N \geq N_0$.

- $f(N) = \Theta(g(N))$ is used to denote the fact that there exist constants c_1, c_2, and N_0, such that $c_1 \cdot g(N) \leq f(N) \leq c_2 \cdot g(N)$, for all $N \geq N_0$.

For example, suppose that the equation $f(N) = 10 \cdot N$ gives the memory requirements of an algorithm when its input has size N. Then, the space complexity of this algorithm is $O(N)$. If we assume that an algorithm needs at least $50 \cdot \log N$

time units[1] to process its input data, then we say that $\Omega(\log N)$ is a lower bound for the time complexity of this algorithm. In the rest of the thesis, all time complexities were estimated using the uniform cost assumption. That means that every arithmetic and pointer operation (access to a memory location) takes $O(1)$ (constant) time.

Three types of complexity analysis are common in the estimation of the performance of algorithms and data structures, namely worst case analysis, average case analysis, and amortised analysis. In a worst case analysis we derive worst case bounds. That is, we estimate the time or memory space of the input data in the worst case. This is the most frequent type of analysis. In the average case analysis, we assume a known probability distribution for the input data and compute the expected time or space complexity under this probability distribution assumption. The amortised analysis ([156]) is actually an averaging over time. Therefore, it is used for estimating time complexities. More precisely, we can define amortisation as averaging the computational time needed for processing the elements of a worst case data sequence over the sequence. In a worst case analysis the estimated time complexity might be too pessimistic. On the other hand, an average case analysis depends on a probability distribution assumption which may not reflect the real distribution of the data sequence. The author in [156] claimed that the amortisation analysis can give both realistic and robust time complexities.

One of the most important problems in computer science is the *dictionary problem.* The problem refers, as it is widely known, to the efficient organisation of a set of elements, e.g. the grey levels in an image, so as to best support a number of primitive operations on the above set. Let $S \subseteq U$ be any subset of the universe U (the set of all elements). We assume that an information $inf(x)$ is associated with every element $x \in S$. The classical dictionary problem asks for a data structure which supports the following three primitive operations:

[1] all logarithms encountered in time and space complexities are base 2

$$\text{Access}(x): \quad \begin{cases} (\text{true}, inf(x)), & \text{if } x \in S \\ \text{false}, & \text{otherwise} \end{cases}$$

$\text{Insert}(x, f)$: replace S by $S \cup \{x\}$ and associate information f with x $(inf(x) = f)$

$\text{Delete}(x)$: replace S by $S - \{x\}$

Note that the operations insert and delete are destructive, that is, the old version of set S is destroyed by these operations. This is the dynamic dictionary problem. In the static dictionary problem only the access operation is supported.

Since the mid-fifties many sophisticated solutions were developed for the dynamic dictionary problem. They can be classified into two main categories, namely representation-based data structures and comparison-based data structures. In the representation-based structures the representation of the elements of U is used as a string over some alphabet, in order to access information. A typical representative of these structures is hashing. In the comparison-based data structures only comparisons between the elements of U are used for this purpose. In this category, two types of data structures can be distinguished, namely explicit and implicit. An implicit data structure for a set S, $|S| = N$ (the symbol $|\cdot|$ denotes cardinality of a set), uses a single array of size N to store the elements of S. The co-occurrence matrix is of this type. The explicit data structures use additional memory for the storage of explicit pointers between elements. These pointers are actually employed for building the relevant data structure.

5.2 Explicit data structures or search trees

Explicit data structures include the various types of trees. We can define a tree as an acyclic connected graph $T = (V, E)$, where $V = V_1 \cup V_2 \cup V_3$ is the set of nodes of the tree and E is the set of edges between the nodes of the tree. In the case of trees it holds that $|V_1| = 1$. If $|V| = 1$ then $V_1 = V_2$ and $V_3 = \emptyset$, otherwise the ordered set (V_1, V_2, V_3) constitutes a partition of V. V_1 contains a special node

called the root of the tree. V_2 consists of the nodes of the tree which are called leaves of the tree. The nodes of V_3 are called the internal nodes of the tree. The tree is a hierarchical structure where the root is located at level 0 and the leaves are located at l levels $n-l+1,\ldots,n$. All the other nodes (internal nodes) can be distributed over all levels except level 0 and level n. The depth of a tree T ($d(T)$) or height of T ($h(T)$) is the length of the largest path from the root to leaves and is equal to n. Also, we define the depth of a node v ($d(v)$), $v \in T$ (in the following we consider $T \equiv V$; the set of the edges E does not play an important role here and we will ignore it), as the length of the unique path from the root to this node. The path is unique because T is an acyclic graph. The depth of a node also gives the level at which this particular node is located. The height of a node $v \in T$ is defined as $h(v) = d(T) - d(v)$. From the above definitions, it is obvious that for the depth of the root r of a tree T it holds that $d(r) = 0$ and for the corresponding height it holds that $h(r) = d(T)$ (see Figure **5**.1). From now on, we will use the symbols defined in this figure to represent internal nodes and leaves, unless specified otherwise.

A tree T can also be defined by the following recursive definition:

- a tree T may contain only one node which is simultaneously the root and leaf of T,
- if v is a node and $T_1,\ldots,T_k$ are trees then $T = (v, T_1,\ldots,T_k)$ is a tree, where v is the root of T directly connected with the roots of these trees.

From this definition, it is clear that a tree T can be considered as a relation $\mathcal{R} : \mathcal{V} \rightarrow \mathcal{T}^k$, where $\mathcal{V}$ is the universe of nodes and $\mathcal{T}$ is the universe of graphs defined by our first definition. k is called degree of the root and $T_1,\ldots,T_k$ subtrees of the tree T. A set of trees is called a forest of trees. The notion of degree is also defined for the rest of the nodes of the tree as follows: If a node v, which is not the root, has degree d in graph T, then its degree $dg(v)$ in tree T is $dg(v) = d-1$ (see Figure **5**.1). These $d-1$ links lead to $d-1$ nodes $v_1,\ldots,v_{d-1}$ which form an ordered set $(v_1,\ldots,v_{d-1})$ and are called the children of v. Each v_i, $i = 1,\ldots,d-1$ can be considered root of the tree T_{v_i} containing all nodes of T that can be accessed from v through v_i. The ordered set $(v, T_{v_1},\ldots,T_{v_{d-1}})$ is called the subtree T_v of

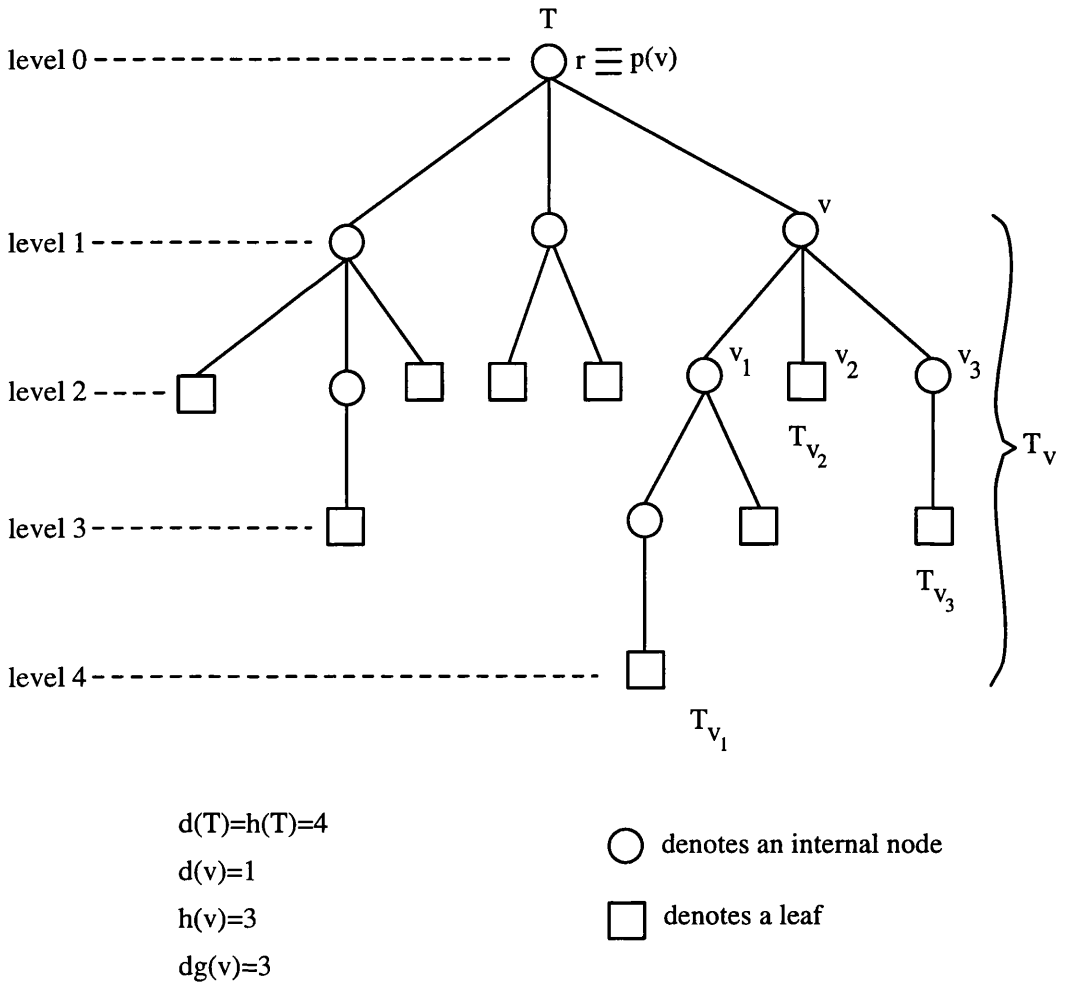


Figure 5.1: Example of the structure of a general tree T.

tree T (see Figure 5.1). Also, we call the trees of the forest $\{T_{v_1}, T_{v_2}, \ldots, T_{v_{d-1}}\}$ subtrees of node v. Leaves are the nodes which have no children. The edge of v that remains (recall that the degree of v in graph T is d) is the edge that actually connects node v with the tree T. This edge leads to node $p(v)$ which is called the parent of v (see Figure 5.1). Root has no parent. A path from v towards the root r of T connects v with nodes $p(v), p^2(v), \ldots, p^m(v), \ldots, p^{d(v)}(v) \equiv r$, where $p^2(v)$ is the grandparent of v and generally $p^m(v)$ is the mth ancestor of v. Root has no ancestors.

A binary tree is a tree T where each node $v \in T$ has degree at most 2, that is, $0 \leq dg(v) \leq 2$ (see Figure 5.2). A full binary tree is a tree T where for each $v \in T$, $dg(v) \in \{0, 2\}$. Therefore, in a full binary tree all nodes have two children, except from the leaves which have no children (see Figure 5.3).

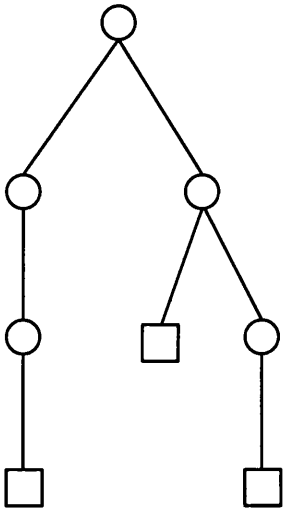


Figure 5.2: Example of a binary tree.

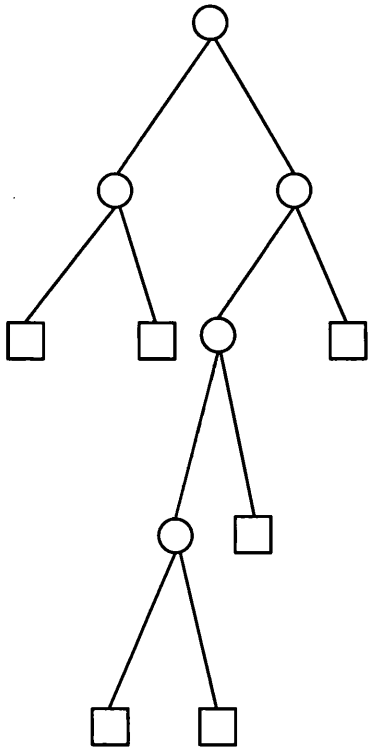


Figure 5.3: Example of a full binary tree.

There are three ways for traversing a tree, one of which can be applied only to binary trees. Let T be a tree, r its root, and $T_{v_1}, T_{v_2}, \ldots, T_{v_{dg(r)}}$ the subtrees of the root.

- Preorder:

  1. visit root r of the currently accessed tree T
  2. apply step 1. to the subtrees $T_{v_1}, T_{v_2}, \ldots, T_{v_{dg(r)}}$ of r in this order

- Postorder:

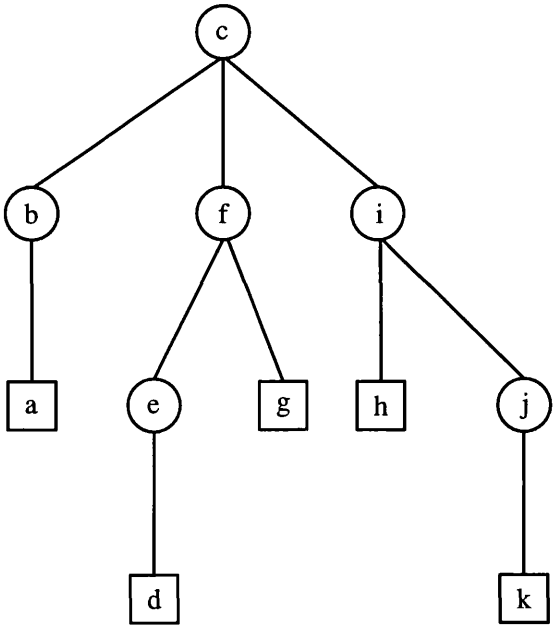
  1. apply step 1. to the subtrees $T_{v_1}, T_{v_2}, \ldots, T_{v_{dg(r)}}$ of the root r of the currently accessed tree T, in this order
  2. visit root r of the currently accessed tree T

- Symmetrical or lexicographic order (defined only for binary trees):

  1. apply step 1. to subtree T_{v_1} (left subtree) of the root r of the currently accessed tree T
  2. visit root r
  3. apply step 1. to subtree T_{v_2} (right subtree) of the root r of the currently accessed tree T

Figures **5**.4 and **5**.5 show examples of the above traversals.

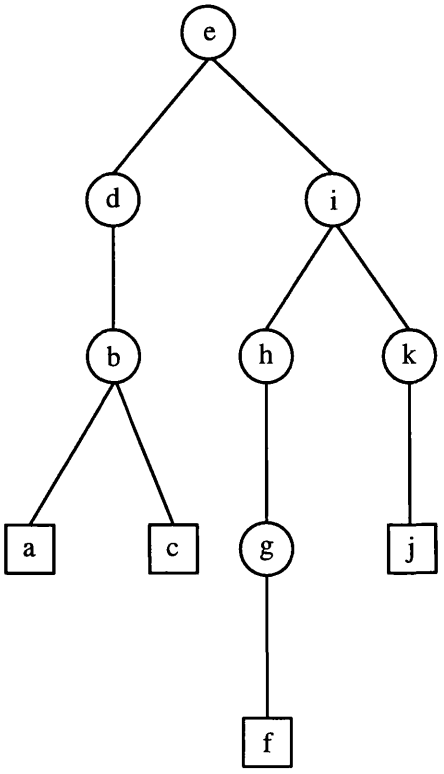
A complete full binary tree of depth d is a full binary tree which has the maximum possible number of nodes (see Figure **5**.6). We can prove the following lemma. The proof is based on induction, which is a standard tool for proving lemmas in data structure theory.



Preorder traversal sequence: c, b, a, f, e, d, g, i, h, j, k

Postorder traversal sequence: a, b, d, e, g, f, h, k, j, i, c

Figure 5.4: Example of a preorder and postorder traversal.



Symmetrical traversal sequence: a, b, c, d, e, f, g, h, i, j, k

Figure 5.5: Example of a symmetrical traversal.

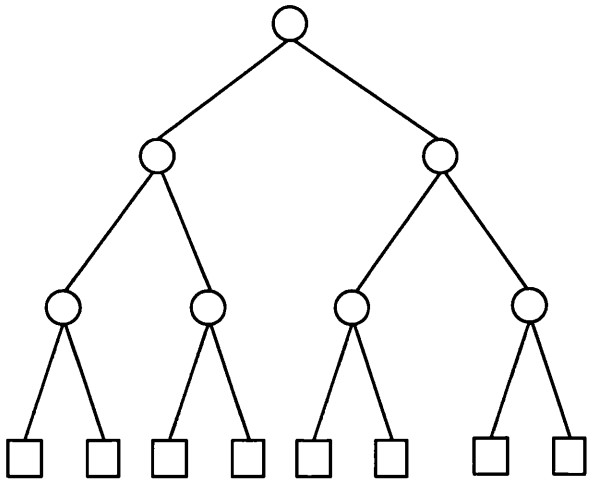


Figure 5.6: A complete full binary tree of depth 3.

Lemma 5.2.1 *A complete full binary tree of depth d has exactly $2^{d+1} - 1$ nodes. The number of the leaves of the tree is exactly 2^d.*

Proof:

We can easily prove the above lemma by induction on the depth of the tree. For $d = 0$, tree T has only one node which is root and leaf at the same time. Let $N(d)$ denote the number of nodes of a complete full binary tree T with depth d and $L(d)$, the number of leaves of this tree. Then $N(0) = L(0) = 1$. We assume that $N(d) = 2^{d+1} - 1$ and $L(d) = 2^d$.

A complete full binary tree T of depth $d + 1$ is constructed by substituting every leaf of tree T of depth d with a node having two leaves for children (see Figure 5.7). Thus

$$N(d+1) = 2^{d+1} - 1 - 2^d + 3 \cdot 2^d = 4 \cdot 2^d - 1 = 2^{d+2} - 1 \qquad (5.1)$$

and

$$L(d+1) = 2 \cdot 2^d = 2^{d+1} \qquad (5.2)$$

□

From the above lemma, it results that a full binary tree of depth d has at most $2^{d+1} - 1$ nodes and at most 2^d leaves. We will now prove the following useful lemma.

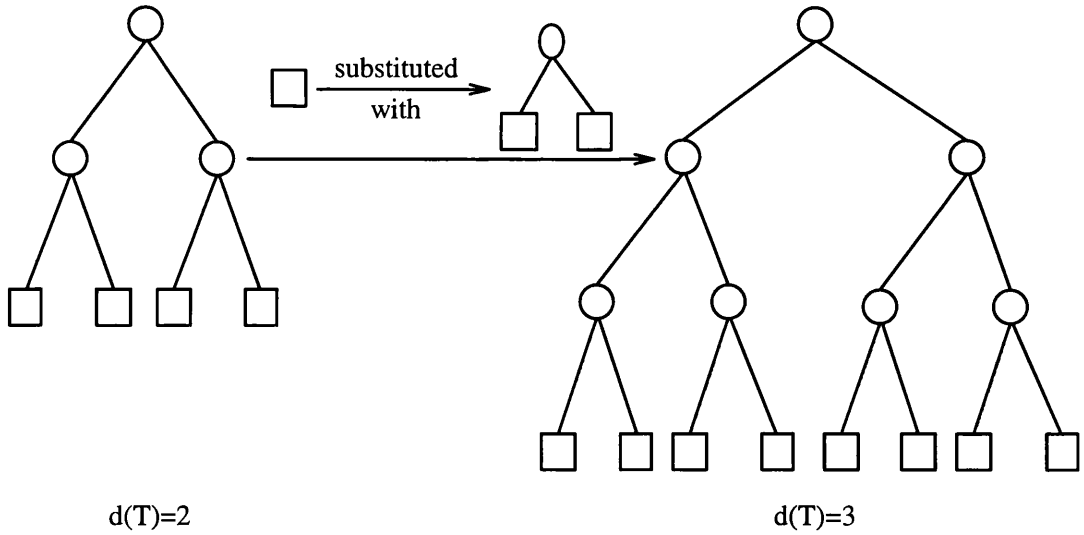


Figure 5.7: Construction of a complete full binary tree of depth 3 from a complete full binary tree of depth 2.

Lemma 5.2.2 *A full binary tree T with N leaves has depth d that satisfies* $\lceil \log N \rceil \leq d \leq N - 1$.

Proof:

To prove that the above relation holds, we assume that there is a full binary tree T having N leaves and depth d, where $d < \log N$ ($\log N \leq \lceil \log N \rceil$). As we have already said, this tree has at most 2^d leaves. Therefore, it holds that $N \leq 2^d$. This is, though, a contradiction since

$$d < \log N \Rightarrow 2^d < 2^{\log N} \Rightarrow 2^d < N \tag{5.3}$$

Therefore, for N leaves we need a full binary tree whose depth cannot be less than $\lceil \log N \rceil$.

The other extreme (maximum depth) happens when there is one leaf at each level, except from the root level where there is only one node (the root) and the last level which must have at least two leaves, for a full binary tree of depth $d \geq 1$ (see Figure 5.8). Actually, in this case, as it is clear from the above figure, the full binary tree degenerates to a list-like structure having depth $d = N - 1$. This is a most undesirable outcome in terms of the time efficiency of the primitive operations in the dictionary problem. This is also the case where a full binary tree of depth d has the least number of nodes which is $2 \cdot d + 1$ and the least number of leaves which is $d + 1$.

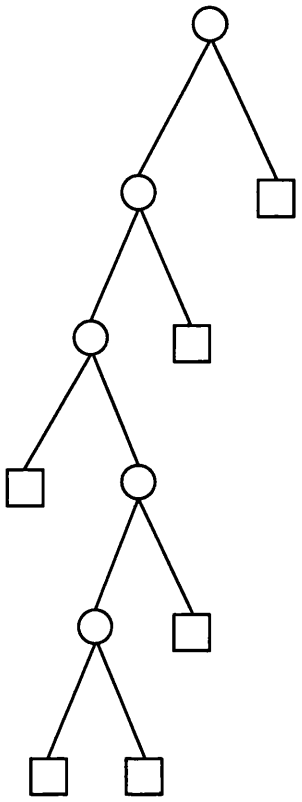


Figure 5.8: A worst case of a full binary tree having N=6 leaves.

We can now show that there is a full binary tree T with depth $\lceil \log N \rceil$ that has N leaves, in order to prove the inequality $d \geq \lceil \log N \rceil$. From what we said above, it follows that such a tree (with depth $\lceil \log N \rceil$) must have at least $\lceil \log N \rceil + 1$ and at most $2^{\lceil \log N \rceil}$ leaves. Starting from a tree that has $\lceil \log N \rceil + 1$ leaves and substituting one leaf at a time with a tree having only two leaves, except from the leaves of the last level, we can create a full binary tree of depth $\lceil \log N \rceil$ having N leaves (see Figure 5.9). □

From what we have said, it is clear that if N is the number of nodes of a full binary tree of depth d, then $2 \cdot d + 1 \leq N \leq 2^{d+1} - 1$. This is though only a necessary condition. That is, if we are given an arbitrary number N, it is not always possible to find a full binary tree of some depth d which has N nodes in it. Actually, N must be an odd number. This is because in a full binary tree, each node (leaf) can give birth to two other nodes during the construction process. So, nodes are created in pairs except from the root. Therefore, it is clear that the number of nodes of a full binary tree is odd.

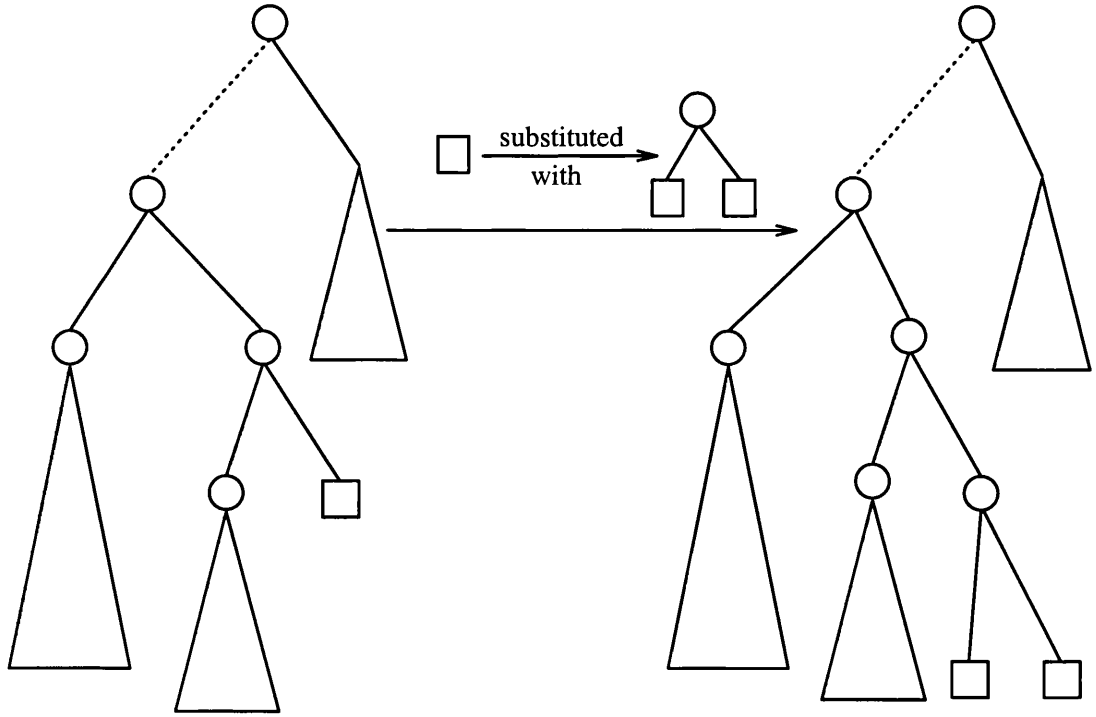


Figure 5.9: Increasing the number of leaves without changing the depth of the tree.

Suppose that we have an odd number N. Following arguments similar to the proof of Lemma 5.2.2, we can show that the full binary tree having N nodes, has depth $d \geq \lceil \log(N+1) \rceil - 1$. In the worst case, $d = \frac{N-1}{2}$ (note that this is an integer since N is odd), since a list-like structured full binary tree (worst case full binary tree) of depth d has $2 \cdot d + 1$ nodes. Therefore, a full binary tree of N nodes (N is always odd) has depth d where $\lceil \log(N+1) \rceil - 1 \leq d \leq \frac{N-1}{2}$.

In order to include the case where N is even, we expand the definition of the full binary tree. We define a full binary tree to be a binary tree T, where each node has either two children (internal node) or no children (leaf), except from the internal nodes having leaves as children, which may have one leaf as their only child. We call such a tree a full binary tree in the wide sense (see Figure 5.10). We call a full binary tree defined by the former definition a full binary tree in the strict sense. Actually, if $\mathcal{T}_{(sfb)}$ is the set of all trees $\mathcal{T}$ that are full binary trees in the strict sense, $\mathcal{T}_{(wfb)}$ is the set of all full binary trees in the wide sense, $\mathcal{T}_b$ is the set of all binary trees and finally, $\mathcal{T}$ is the set of all trees, it holds that $\mathcal{T} \supset \mathcal{T}_b \supset \mathcal{T}_{(wfb)} \supset \mathcal{T}_{(sfb)}$ (see Figure 5.11).

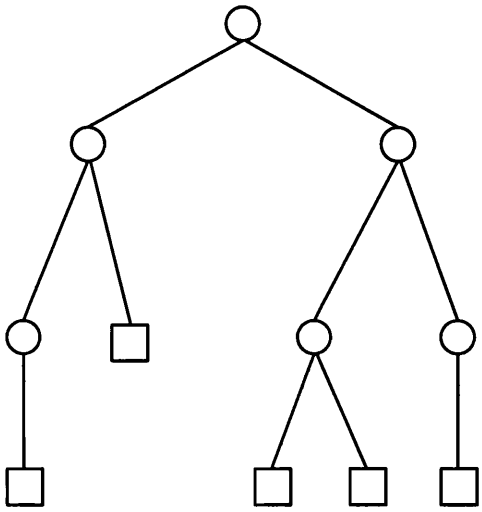


Figure 5.10: Example of a full binary tree in the wide sense.

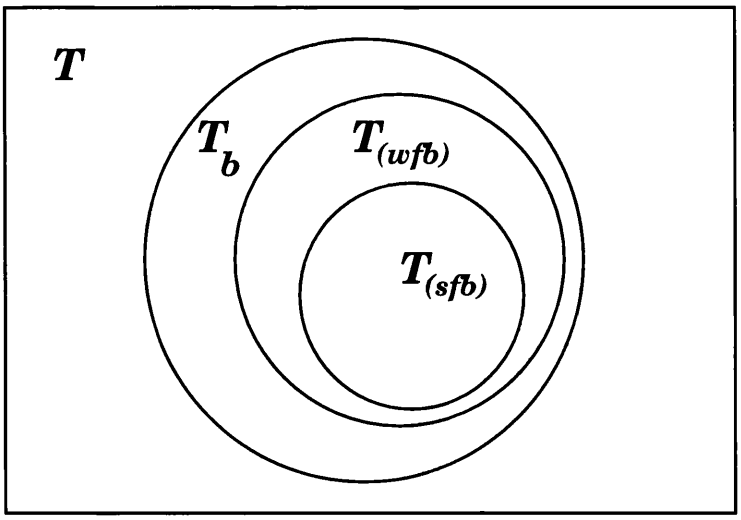


Figure 5.11: Illustration of the containment relation of the sets of the various types of binary trees.

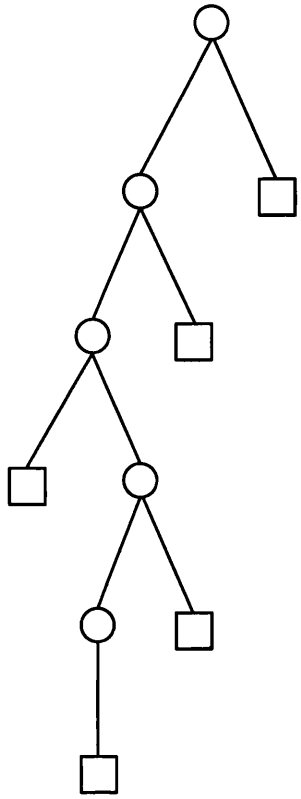


Figure 5.12: A worst case full binary tree in the wide sense of depth 5.

A full binary tree[2] can be constructed from a full binary tree in the strict sense by removing $k \geq 0$ leaves from nodes having two leaves as children. A consequence of this fact is that a full binary tree of depth d has at most $2^{d+1} - 1$ nodes, equal to the number of nodes of a complete full binary tree (a complete full binary tree is always in the strict sense). Also, the full binary tree of depth d with the least number of nodes is a list-like structured tree where each level has two nodes, except from the first (root level) and the last level, which they both have one node (see Figure 5.12). It is thus obvious that in this case $N = 2 \cdot d$, so for a full binary tree of depth d it holds that $2 \cdot d \leq N \leq 2^{d+1} - 1$. Now, given a full binary tree with N nodes it holds that $\lceil \log(N+1) \rceil - 1 \leq d \leq \lfloor \frac{N}{2} \rfloor$. This can be proved following similar arguments as in the proof of Lemma 5.2.2.

A search tree for a set $S = \{x_1 < \ldots < x_N\}$ is a full binary tree T which stores the elements of S in its nodes. Each node of the search tree contains an element from S or an auxiliary element. All these elements are stored in the

[2]in the following, when we refer to full binary trees we will mean full binary trees in the wide sense, unless stated otherwise

following order. Let v be an internal node or the root of the search tree T. Let $val(v)$ denote the element that is stored in node v. Let also v_1, v_2 be its left and right child, respectively. Then it is always valid that $val(v_1) \leq val(v) < val(v_2)$. We discriminate between two versions of search trees, namely the leaf-oriented version and the node-oriented version. In the leaf-oriented version, the elements of S are stored in the leaves of the search tree T. All other nodes contain auxiliary information, which is used to access the elements of S in the leaves. Therefore, tree T has $N = |S|$ leaves. Actually, the leaf-oriented version of a search tree is a full binary tree in the strict sense.

In the node-oriented version, set S is stored in the entire tree, so a search tree in this case has N nodes. It is obvious that such a search tree must be a full binary tree in the wide sense (recall that in a strict sense full binary tree the number of nodes must be odd). Also, the values of node v and its two children (if it has two children) now satisfy the inequality $val(v_1) < val(v) < val(v_2)$, where v_1, v_2 are its left and right child, respectively. If v has only one child v_1 then it holds that $val(v_1) < val(v)$. Therefore, in the node-oriented version, in contrast to the leaf-oriented version, the content of a node is always strictly greater than the content of its left child (if it has a child, i.e. it is not a leaf). The node-oriented version needs less memory space but leads to more complex algorithms than the leaf-oriented version. Since, as we have already said, storage is crucial in the SGLDM, we adopted the node-oriented version.

In the following, we define the basic operations for the node-oriented version of search trees. These operations are similarly defined in the case of the leaf-oriented version, with slight modifications. From now on, when we say search tree we will mean a node-oriented search tree. The pseudocode for the three primitive operations access, insert, and delete is the following:

Access(x):

$v \leftarrow root$;
WHILE ($val(v) \neq x$ **AND** v is not a leaf) **DO**
  IF ($val(v) > x$)
    $v \leftarrow lson(v)$;
  ELSE
    $v \leftarrow rson(v)$;
  ENDIF
ENDWHILE
IF ($val(v) \neq x$)
  RETURN (**FALSE**,v);
ELSE
  RETURN (**TRUE**,v);
ENDIF

Insert(x, f):

$found \leftarrow Access(x)$;
IF ($found.1$ =**TRUE**)
  $update(found.2, f)$;
ELSE
  $add(x, found.2, f)$;
ENDIF

Delete(x):

$found \leftarrow Access(x)$;
IF ($found.1$ =**TRUE**)
  $update(found.2, f)$;
    or
  $remove(found.2)$;
ELSE
  no operation;
ENDIF

As we have said, the function $val(\cdot)$ returns the element stored in the relevant node. Functions $lson(\cdot)$ and $rson(\cdot)$ return the left and the right child of the

relevant node, respectively. Variable *found* is actually a structure having two fields. The first field stores the outcome of the access operation. The second field contains the last node this operation reached. This is the requested node, only if the access operation found the element it was looking for in the tree. Subroutine *update*$(\cdot)$ performs what its name implies, i.e. it updates the information content of the relevant node. Finally, subroutines *add*$(\cdot)$ and *remove*$(\cdot)$ result in the addition of a new node in the search tree and the removal of a node from the search tree, respectively.

From the above pseudocode, we note that the access operation plays a central role in a search tree. The time complexities of the insert and delete operations depend mainly on the time complexity of the access operation. The access operation in turn needs time proportional to the depth of the search tree. If d is the depth of the tree then, in the worst case, that is, when access needs to reach the leaves of the tree that are farthest from the root, $O(d)$ time is needed to complete the operation. For a search tree of N nodes we showed that d can be as much as $\lfloor \frac{N}{2} \rfloor$. Therefore, the time complexity for the access operation can be as much as $O(N)$. This is actually the time needed when the data structure is a linear list and we perform a linear search. That is not a surprise, since in the extreme case where $d = \lfloor \frac{N}{2} \rfloor$, the search tree degenerates to a list-like structure. We will show that the average case is much better. The average length of the path that the access operation needs to traverse in the worst case, i.e. from the root to a leaf, is given by the following theorem:

Theorem 5.2.1 *The average path length of a random full binary tree (created by random insertions), in the worst case, is* $O(\log N)$*, where* N *is the number of elements stored in the tree.*

Proof:

Say that $d(\lambda)$ is the average path length for a search tree T with λ leaves. It is clear that $d(1) = 0$. We also have that:

$$d(\lambda) = \frac{1}{\lambda} \sum_{i=1}^{\lambda} d_i \tag{5.4}$$

where d_i is the length of the path from the root to leaf i in tree T and $\frac{1}{\lambda}$ is the probability that we access leaf i.

If we add a new leaf to tree T, a new tree T' results having $\lambda+1$ leaves. Three cases exist:

1. In the first case, the new leaf becomes the only child of a leaf in T which is then transformed into an internal node in T' (see Figure **5**.13). In this case, the length of an existing path (a path from the root to a leaf in T) increases by one in T'.

2. In the second case, the new leaf becomes the second child of an internal node in T (see Figure **5**.14). In this case, a new path from root to the new leaf is created in T', whose length is equal to the length of the worst case of the path that the access operation followed, in the first place.

3. In the final case, the access operation of the insertion reaches a leaf, which is the only child of an internal node (see Figure **5**.15). If we simply add the new leaf at this point, we will have the case where an internal node has only one child which happens to be another internal node. This situation violates the definition of the full binary tree and is handled by making the new leaf the second child of the father of the leaf which the access operation reached in T, in the first place. It is clear that this case is equivalent to the second one.

We now assume that the above three cases have equal probability to occur (random tree). We also assume that each path in T has the same likelihood $\frac{1}{\lambda}$ to accept the new leaf in each case.

- In the first case we have that:

$$d^{(1)}(\lambda+1) = \frac{1}{\lambda}\cdot\left\{\frac{1}{\lambda}\cdot(d_1+1+d_2+\cdots+d_\lambda)+\right.$$

$$\left.\frac{1}{\lambda}\cdot(d_1+d_2+1+\cdots+d_\lambda)+\cdots+\frac{1}{\lambda}\cdot(d_1+d_2+\cdots+d_\lambda+1)\right\}=$$

$$\frac{1}{\lambda}\cdot\frac{1}{\lambda}\left\{\lambda\cdot\sum_{i=1}^{\lambda}d_i+\lambda\right\}=$$

$$\frac{1}{\lambda}\cdot\sum_{i=1}^{\lambda}d_i+\frac{1}{\lambda}=d(\lambda)+\frac{1}{\lambda} \qquad (\mathbf{5}.5)$$

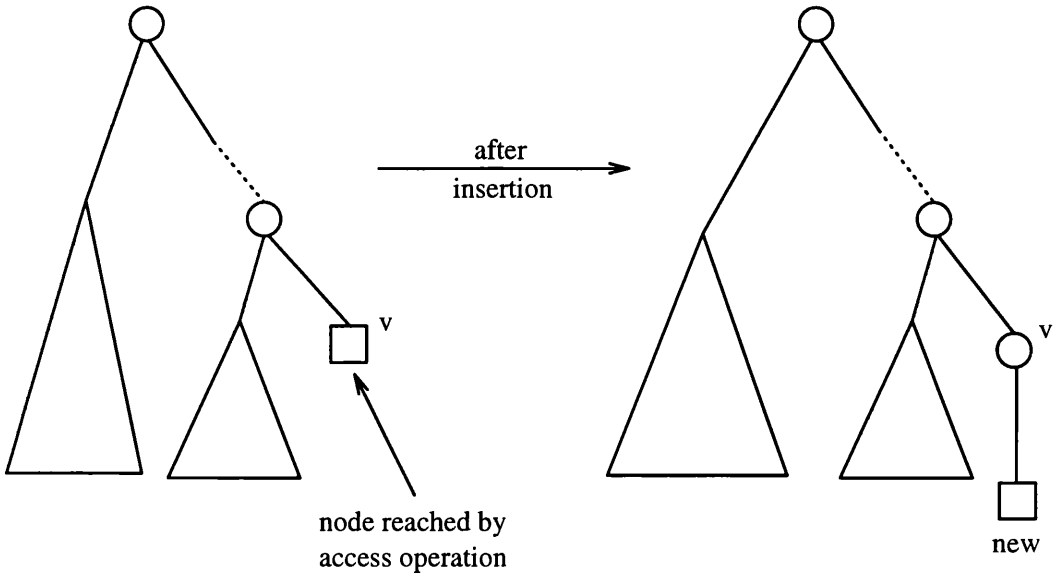


Figure 5.13: Insertion of a new node (Case 1).

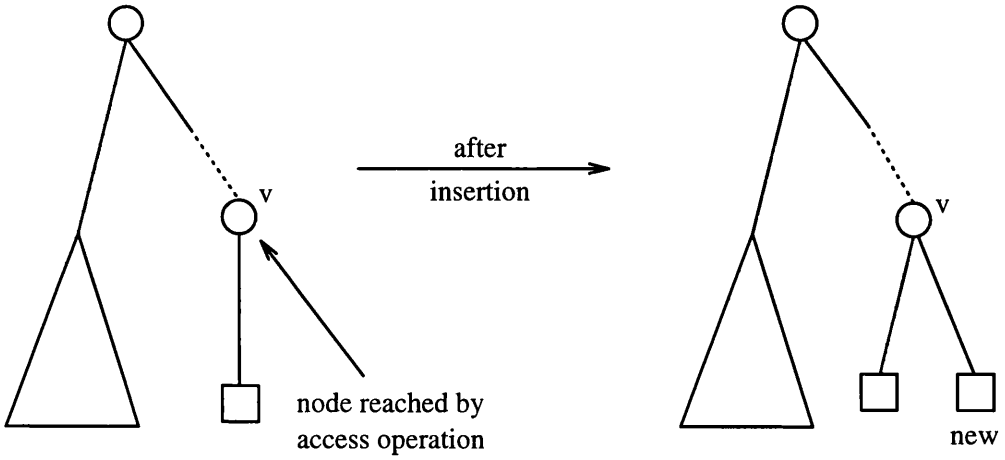


Figure 5.14: Insertion of a new node (Case 2).

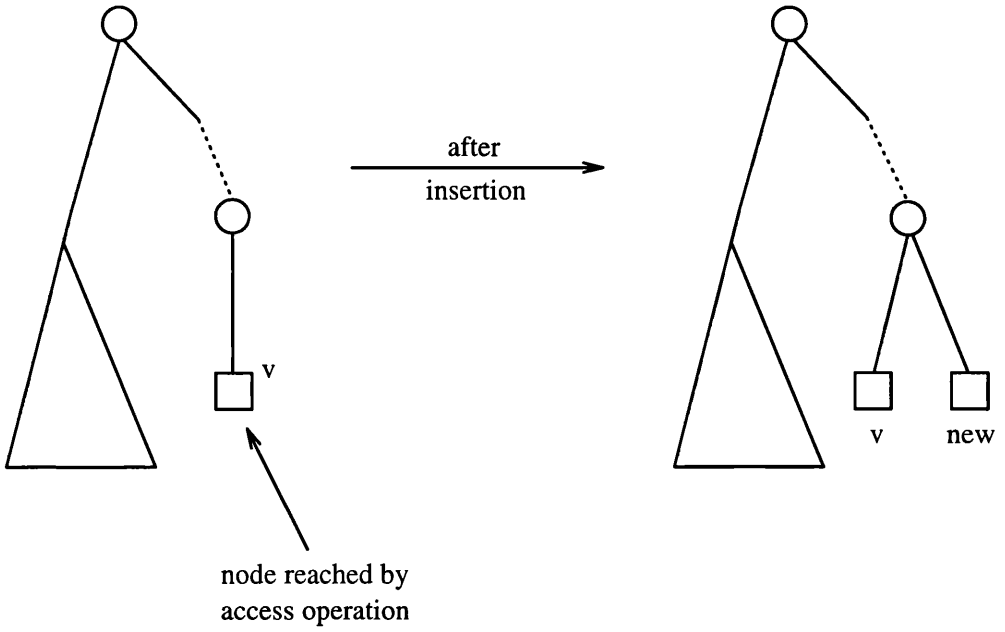


Figure 5.15: Insertion of a new node (Case 3).

- In the second and third case we have that:

$$d^{(2)}(\lambda+1) = d^{(3)}(\lambda+1) =$$

$$\frac{1}{\lambda} \cdot \left\{ \frac{1}{\lambda+1} \cdot (2d_1 + d_2 + \cdots + d_\lambda) + \frac{1}{\lambda+1} \cdot (d_1 + 2d_2 + \cdots + d_\lambda) + \cdots + \right.$$

$$\left. \frac{1}{\lambda+1} \cdot (d_1 + d_2 + \cdots + 2d_\lambda) \right\} =$$

$$\frac{1}{\lambda} \cdot \frac{1}{\lambda+1} \cdot \left\{ \lambda \cdot \sum_{i=1}^{\lambda} d_i + \sum_{i=1}^{\lambda} d_i \right\} =$$

$$\frac{1}{\lambda} \cdot \sum_{i=1}^{\lambda} d_i = d(\lambda) \qquad (5.6)$$

The latter result is reasonable, since in these two cases the paths do not increase in length but in number by one. So, the average path from the root to a leaf is not expected to change from tree T with λ leaves to tree T' with $\lambda + 1$ leaves.

Since we assumed that the above three cases have the same probability to occur, it holds that:

$$d(\lambda+1) = \frac{1}{3}\cdot\left(\frac{1}{\lambda}+d(\lambda)\right)+\frac{2}{3}\cdot d(\lambda) \Leftrightarrow$$

$$d(\lambda+1) = \frac{1}{3}\cdot\frac{1}{\lambda}+d(\lambda) \tag{5.7}$$

Since $d(1)=0$, this recursive relation has solution

$$d(\lambda)=\frac{1}{3}\cdot\sum_{x=1}^{\lambda-1}\frac{1}{x}=O\left(\int_1^{\lambda-1}\frac{1}{x}dx\right)=O(\log(\lambda-1))=O(\log\lambda) \tag{5.8}$$

Since $\lambda \leq N$, it follows that $d(\lambda)=O(\log N)$. □

We saw above that depth is a crucial parameter for the performance of the search trees. It determines the time complexity of the primitive operations access, insert, and delete. According to the above theorem, the average depth in a full binary tree is logarithmic on the number N of elements stored in the tree and therefore, much better than the worst depth which is linear. This led to the development of balanced trees, where the depth is guaranteed to be $O(\log N)$. Using these trees, we avoid the worst case where a search tree degenerates to a list-like structure.

A search tree T for set S, with $|S|=N$, is called balanced if it holds that $h(T)=c_1\cdot\log N+c_2$, where c_1, c_2 are constants and $h(T)$ is the height of tree T. Balanced trees are defined according to a balancing criterion, which ensures the $O(\log N)$ depth property, so that they be able to perform the primitive operations access, insert, and delete in $O(\log N)$ time, where N is the number of nodes in the tree. Based on the balancing criterion, we discriminate between two categories of balanced trees:

- height-balanced trees: In this category, the balancing criterion involves the height of the subtrees of a tree T (the subtrees of its nodes). Let v be the parent of two children and T_l, T_r, its left and right subtree, respectively. Then, the balancing criterion imposes some constraints on $h(T_l)$ and $h(T_r)$, the heights of T_l and T_r. For example $|h(T_r)-h(T_l)|\leq c$, where c is a

constant, is such a constraint. The balancing criterion may be expressed in several forms and so various height-balanced trees result, e.g. AVL-tree, B-tree, HB-tree, Red-Black tree, (a,b)-tree, half-balanced tree and BB-tree (see [110], [163]).

- weight-balanced trees: In this category, the balancing criterion involves the weight of the subtrees of a tree T. The weight of a tree T, denoted as $|T|$, is the number of nodes in T. Let again v be the parent of two children in T. The balancing criterion imposes some constraints upon the weights of T_v and the subtrees of v, T_l and T_r (left and right subtree, respectively). For example, $a \leq \frac{|T_l|}{|T_v|} \leq 1-a$, where a is a constant, $|T_l|$ the weight of tree T_l and $|T_v|$ the weight of tree T_v. Examples of weight-balanced trees are the BB[a]-tree and the internal-path tree (see [110], [163]).

From the three primitive operations that are defined on a search tree T, namely access, insert, and delete, only the last two (insert and delete) change the structure of T (destructive operations). In the balanced trees, in order to avoid the degeneration to list-like structures and thus preserving the $O(\log N)$ depth property, we may need to perform some operations after an insertion or deletion. These operations are called rebalancing operations and are actually performed when the balancing criterion is violated. In order to detect this ill condition, each node of a balanced tree has to store some additional information called balance information. This information can be a function of its height, the height of its parent (if it exists) and the height of its children in the height-balanced trees, or a function of its weight, the weight of its parent (if it exists) and the weight of its children in the weight-balanced trees. It is used for the implementation of the balancing criterion.

After an insertion or deletion, the balance information may change at the node which is the parent of the leaf that was inserted in an insert operation or deleted in a delete operation, to reflect the new local structure of the balanced tree. This may in turn cause a local violation of the balancing criterion. Then, rebalancing operations are performed in order to validate the balance information, according to the balancing criterion. They are categorised into:

- balance changes, which involve changes of the balance information at a node or at a node and its parent and characterised as cheap operations (their time cost is small),

- restructuring operations, which cause a change in the structure of the subtree T_v of a node v which is in the local neighbourhood of the node that caused the violation of the balancing criterion (e.g. v can be its grandparent). In this case, we say that a restructuring operation is applied to node v, or else, tree T_v is being restructured. These operations involve some pointer and balance information changes in the above local neighbourhood and characterised as expensive rebalancing operations (they cost much more than the balance changes). One of the design criteria of balanced trees is the avoidance of these operations as much as possible.

One rebalancing operation may not be sufficient to restore the balancing criterion over the entire balanced tree. This is because a rebalancing operation may cause a violation of the balancing criterion in another node. In this case, a rebalancing operation may need to propagate higher up in the path from the root to the leaf that was inserted or deleted and so as many as $O(\log N)$ such operations may be needed, in order for the balance information to be valid over the entire search tree.

In our research, one of the search trees that was chosen to store the textural information extracted from an image region by the SGLDM, was a height-balanced tree called the **BB-tree** (Balanced Binary tree), due to its unique properties. In the following section we describe this useful tree.

5.3 Balanced Binary tree

A BB-tree ([155]), is a balanced full binary tree, where each node v stores an integer, denoted $rank(v)$, which satisfies the following constraints:

1. if v is a node with parent $p(v)$, then $rank(v) \leq rank(p(v)) \leq rank(v) + 1$,
2. if v is a node with grandparent $p^2(v)$, then $rank(v) < rank(p^2(v))$,
3. if v is a leaf then $rank(v) = 0$ and $rank(p(v)) = 1$, provided that v has a parent (i.e. it is not the root of the tree).

Actually, $rank(v)$ is the balance information stored in v, while the above constraints comprise the balancing criterion.

An alternative way to define BB-tree is by considering that each node v stores a bit, denoted $bit(v)$, which satisfies the following constraints:

1. if v is a leaf, then $bit(v) = 1$,
2. if v is the root, then $bit(v) = 1$,
3. if for a node v, $bit(v) = 0$, then $p(v)$ exists and $bit(p(v)) = 1$
4. for any node v, the number of nodes u on a path from v to a leaf for which $bit(u) = 1$ is the same for any path from v to a leaf

In reality, $rank(v)$ and $bit(v)$ of a node v which is not the root, are related according to the relation $bit(v) = rank(p(v)) - rank(v)$. If we colour black a node v having $bit(v) = 1$ and red a node v having $bit(v) = 0$, the above constraints can be expressed in the following way:

1. the root of a BB-tree and its leaves are black,
2. no two successive red nodes can exist on the same path,
3. the number of black nodes on any path from node v to a leaf is the same

According to constraints 2 and 3, the shortest possible path in a BB-tree consists of only black nodes, while the longest possible path consists of black nodes alternated with red nodes. So, the ratio of the length of the longest path to

the length of the shortest path in a BB-tree is at most 2, which clearly shows the balance property of BB-trees according to their height. Obviously, this relation holds for each subtree T_v of a node v of a BB-tree. All the above definitions and notations for the BB-tree are equivalent and they will be used interchangeably.

The following rebalancing operations need to be performed for a BB-tree to remain balanced:

- promotions/demotions
- single/double rotations

Promotions/demotions are actually balance changes. Promotions are applied only in an insert operation, while demotions are applied in a delete operation. Single/double rotations are restructuring operations. They are applied in both insert and delete operations.

Promotion of a node v is equivalent to increasing $rank(v)$ by one. This operation is applied in one case when two successive nodes on the same path (a node v and its parent $p(v)$) become red, a situation which violates one of the constraints mentioned above. Node $p^2(v)$ (the grandparent of v) is promoted in this case. The result of this operation can be equivalently seen as a colour exchange (called colour flip) between $p(v)$ which is red and $p^2(v)$ which is black. Note that $p^2(v)$ cannot be red since then we would have three consecutive red nodes (v, $p(v)$, and $p^2(v)$). This is an impossible state, since immediately after two successive red nodes arise on a path, promotion comes into operation to resolve this invalid situation. Promotion may result in a new violation that must be sorted out.

Equivalently, demotion of a node u is the decrease of $rank(u)$ by one. This operation is applied in some cases where the value $bit(v)$ of a node v becomes $bit(v) = 2$ ($rank(p(v)) - rank(v) = 2$), which is a forbidden value. The result of this operation is that v becomes black ($bit(v) = 1$) and its parent $p(v)$ gets $rank(p(v)) = rank(p(v)) - 1$ (demotion). After demotion, it might happen that $bit(p(v)) = 2$ (new violation), so demotion may need to propagate higher up.

Single and double rotations change the structure of a BB-tree in order to keep it balanced. Actually, a rotation is a local transformation, which changes the depths of specific nodes in order for the ratio of the length of two paths from a

node v down to the leaves to be at most 2 and thus for balance property to be preserved (see above). A double rotation is equivalent to two single rotations.

In this research work, we will only refer to the insert operation in detail, since we are only interested in the extraction of textural information from an image region and its subsequent storage in the proposed structure. The access operation is the same for all search trees and also, as we have already mentioned, is a non-destructive operation. Consequently, it causes no change in the balance state of a tree.

An insertion of a value x in a BB-tree is accomplished by adding a leaf which stores x to the appropriate place in the tree, which has been indicated by the access operation (see Section 5.2). The following cases exist:

- the new leaf becomes the right child of an internal node (see Figure 5.16 (a))[3]. This case happens when the access operation ends up at an internal node v, which means that $x > val(v)$. This is the simplest case and no further action should be taken.

- the access operation ends up at a leaf w whose parent v has one child (see Figure 5.16 (b)). In this case it holds that $x < val(v)$. The resulting structure, which is illustrated in the above figure, is invalid according to the definition of the full binary tree (node v must have two children or one child that is a leaf). It is thus transformed to the structure shown in Figure 5.16 (c). The following changes in the values stored in the above nodes need to be performed. Node v should get $val(v) = max(x, val(w))$ and node n should get $val(n) = min(x, val(w))$. Also, node w should get $val(w) = val(v)$ (the old value of node v). No rebalancing operation needs to be performed in this case.

- the access operation ends up at a leaf w, which is the left child of a node v that has two children (see Figure 5.16 (d)). This case happens when $x < val(v)$. The following actions have to be performed. Node w becomes an internal node and is coloured red. If $x > val(w)$ then leaf n (the new

[3]coloured circles denote red nodes

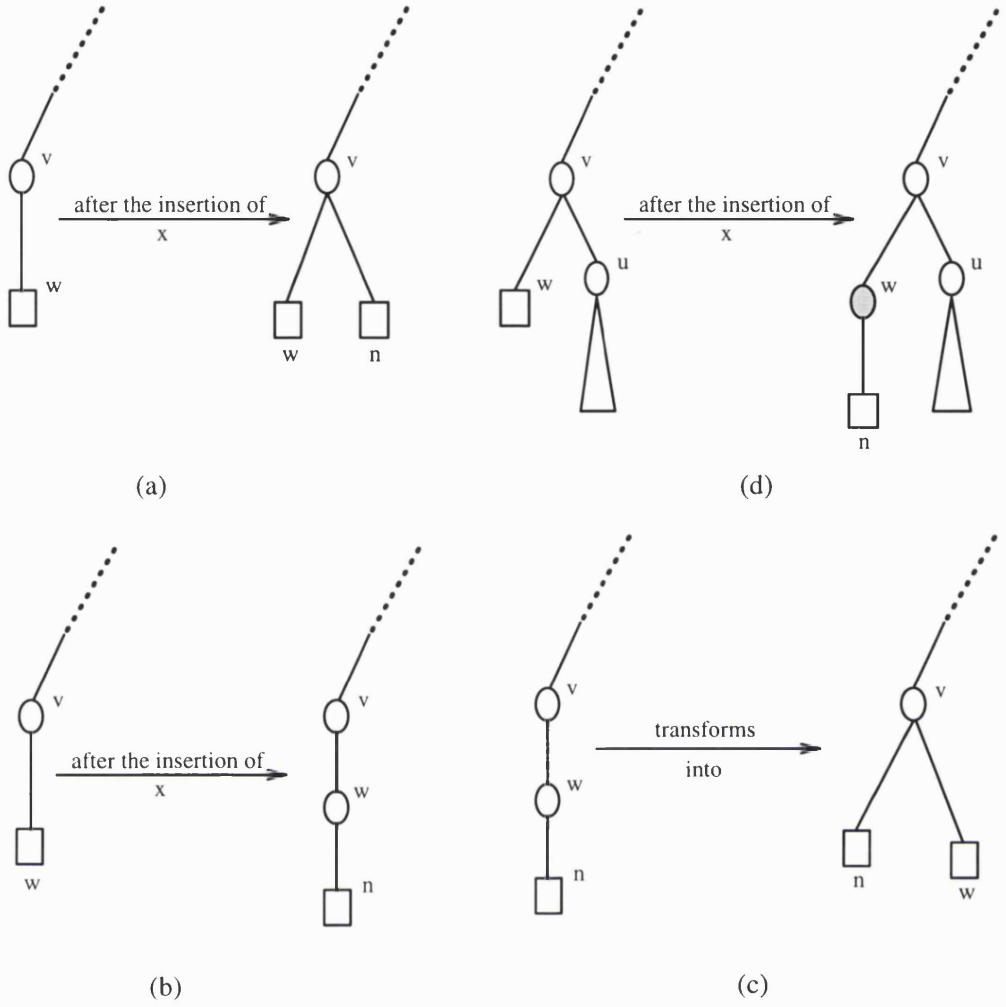


Figure 5.16: Illustration of the various cases of insertion in BB-tree.

leaf) gets $val(n) = val(w)$ and w gets $val(w) = x$, otherwise no other change is made on the new leaf or node w. If now v is black the insert operation concludes at this point. If, on the other hand, is red the balancing criterion is violated and therefore, a rebalancing operation must be performed. Which one of the three operations (promotion, single rotation, double rotation) will be performed depends on the colour of the sibling of v and whether v is left child or right child. We thus have the following three subcases:

- the sibling of v is red (see Figure 5.17 (a)). In this case a promotion is performed, which makes v and y nodes black and z node red. z node becoming red may in turn violate the balancing criterion and so one more rebalancing operation may come into action. This process can

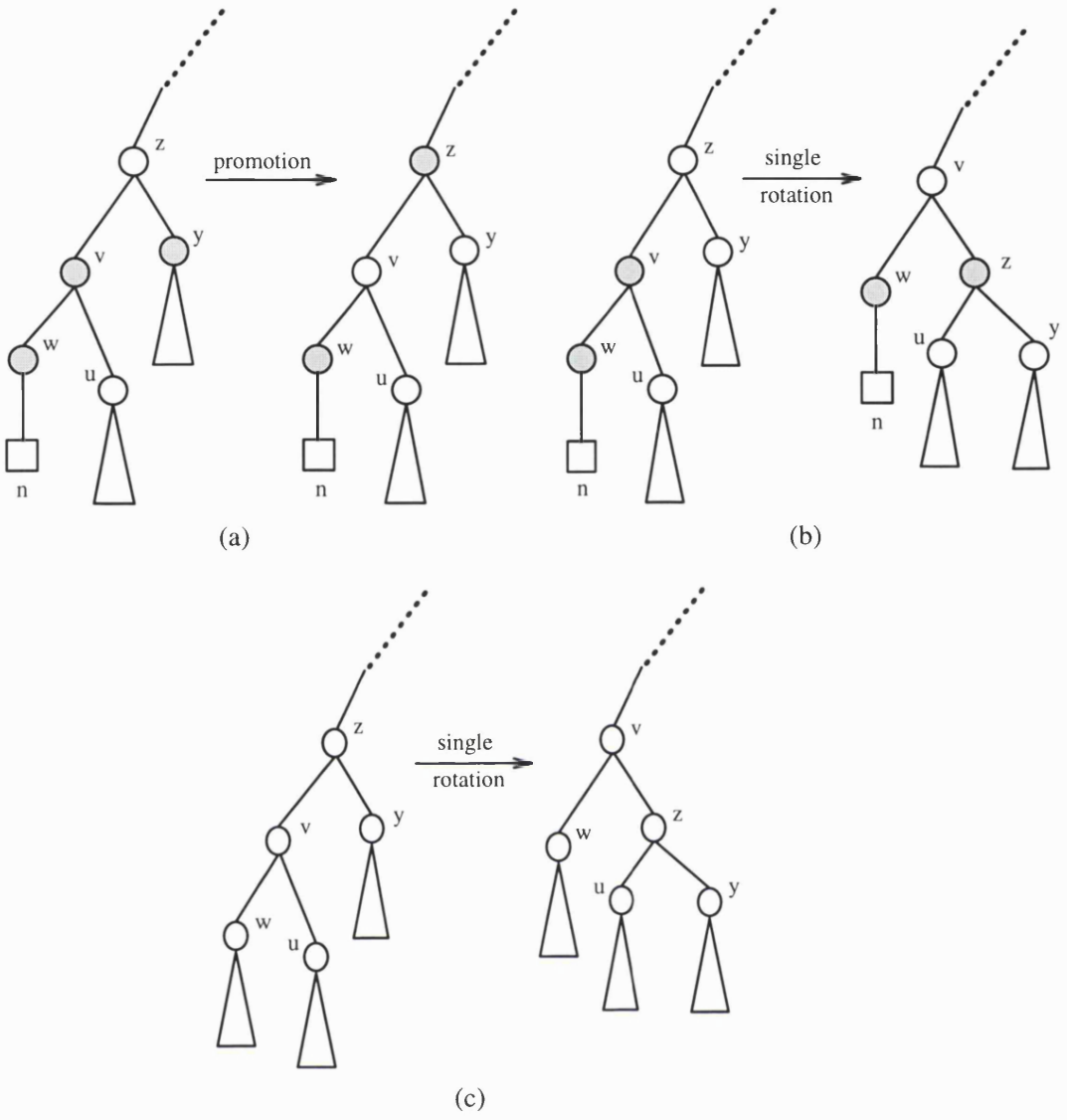


Figure 5.17: Illustration of the various cases of insertion in BB-tree cont.

continue higher up towards the root.

- the sibling of v is black and v is the left child of z (see Figure 5.17 (b)). In this case, a single rotation is performed, as illustrated in the above figure, in which case node v becomes black and node z becomes red. No other rebalancing operation is needed after a single rotation is performed. In general, a single rotation is performed as illustrated in Figure 5.17 (c) (the symmetrical case is omitted).

- the sibling of v is black and v is the right child of z (see Figure 5.18 (a)). In this case, a double rotation is performed, as illustrated in the above

figure, in which case node w becomes black and node z becomes red. Actually, in this case, subtree T_u is empty and u is a leaf, otherwise we would have a violation of the definition of the full binary tree. To see this note that before the insertion of leaf n node v had one child, i.e. a leaf. In order for the balancing criterion to hold, its other child, node u, should be either a leaf or an internal node having as children leaves (remember that the ratio of the length of the longest path from a node v to the leaves to the length of the shortest path from v to the leaves can be at most 2 for the tree to be balanced). If u is a leaf then we conclude at this point. Suppose now that u is not a leaf. Then it can't be black. Let's start from the point where node u was a leaf. After the addition of its first child (the first leaf below it), this node was coloured red (see the cases above). In order for node u to become black one rebalancing operation must be performed. However, it is clear from Figure **5**.17 (a) that a promotion cannot change its colour. Also, single/double rotations cannot happen and turn it into a black node because then, as it is clear from the right parts of Figures **5**.17 (b) and **5**.18 (a), node u should be a father of at least one internal node. Therefore, node u must be red. But in this case node v could not be red because that would violate the balancing criterion (no two consecutive red nodes can exist in the same path). Therefore, if node u is red, i.e. if u is an internal node and not a leaf, node v must be black which is a contradiction to our initial assumption. The conclusion is that node u should be a leaf and therefore, no violation of the definition of the full binary tree happens in the right part of Figure **5**.18 (a).

The values stored in nodes v and u must also be exchanged, i.e. $val(v) = val(u)$ and $val(u) = val(v)$ (the old $val(v)$). If node w had been a parent of two children, as is true for node y in the general case of double rotation shown in Figure **5**.18 (b) (the symmetrical case is omitted), this exchange wouldn't have happened. Also, similarly to the case of a single rotation, no other rebalancing operation is needed. Therefore, only promotion propagates higher up until, either

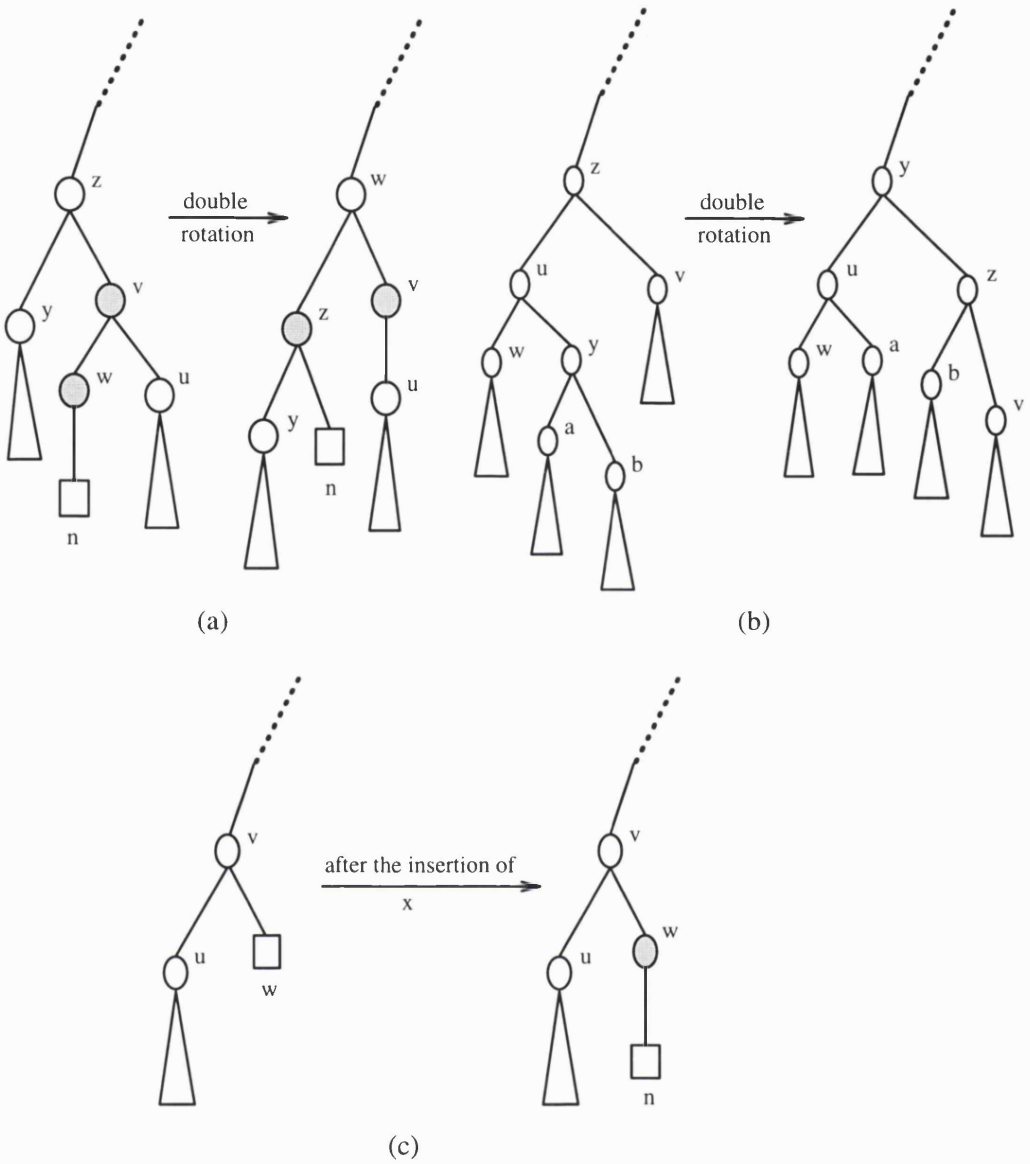


Figure 5.18: Illustration of the various cases of insertion in BB-tree cont.

the balancing criterion is not violated or a single/double rotation is performed.

- the access operation reaches a leaf w, which is the right child of a node v having two children (see Figure 5.18 (c)). This is actually a symmetrical case of the previous one and is handled similarly.

Before we present the basic properties of the BB-tree, we should speak about the time cost of the balance changes (promotions/demotions) and the restructuring

operations (single/double rotations). Since each rebalancing operation involves a local change in at most 6 nodes (the case of a double rotation after a deletion) and each change takes constant time, the time cost of a rebalancing operation is $O(1)$. This of course does not mean that restructuring operations are as cheap as the balance changes. One balance change needs to make changes in the colour of 3 nodes. On the other hand, one restructuring operation needs to make changes in the colour and pointer values of up to 6 nodes. Therefore, restructuring operations are much more expensive than the simple balance changes.

The BB-tree has the following properties ([155], [163]):

1. after an insertion/deletion of an element, $O(\log N)$ balance changes need to be performed at most, but only $O(1)$ restructuring operations.

2. after a sequence of m insertions/deletions, the number of rebalancing operations is $O(m)$. That means that the amortised cost of rebalancing after each insertion/deletion is $O(1)$ (see at the beginning of Section **5**.1).

3. only one bit is necessary for the storage of the balance information at each node v ($bit(v)$).

4. The probability that a sequence of rebalancing operations will propagate up to node v with $h(v) = t$ after an insertion/deletion, is $O(\frac{1}{c^t})$, where c is constant, i.e. the probability decreases exponentially with the height.

5. Top-down rebalancing is possible.

Properties 1 and 2 show that, in a BB-tree, only $O(1)$ expensive restructuring operations need to be performed after an insertion/deletion. Property 3 is very important for the storage requirements of the proposed structure. Property 4 relates to the fact that rebalancing operations happen near the leaves of the tree most of the time. The higher is a node the smaller the probability that a sequence of rebalancing operations will reach this height and therefore, the smaller the total rebalancing cost.

Finally, property 5 states that a rebalancing from the root down to the leaves can instead be performed. This rebalancing is actually carried out in the direction

which the access operation follows towards the leaves. The disadvantage of this method is that it needs $O(\log N)$ restructuring operations, much more than in the bottom-up rebalancing (from the leaves up to the root). Top-down rebalancing also needs more complicated algorithms. However, it is very useful in a parallel processing environment where more than one concurrent processes try to update a BB-tree (insertion/deletion). Top-down rebalancing leads to simpler protocols and also allows for more concurrency in the tree (more processes can act in parallel on the tree).

5.4 The weighted dictionary problem

A generalisation of the dictionary problem is the weighted dictionary problem. In our previous discussion, we implied that the elements stored in a data structure (list or tree) are accessed in random order, that is, they have a uniform access distribution. In this problem, we take into consideration the fact that the access distribution of the elements may not always be uniform.

More precisely, let $S = \{x_1, x_2, \ldots, x_N\}$ be the set of elements stored in a data structure (list or tree). Each element $x_i \in S$ is assigned a weight w_i, which is a positive number that represents the probability p_i that element x_i will be accessed in a sequence of access operations. Let $W = \sum_{i=1}^{N} w_i$, then $p_i = \frac{w_i}{W}$. The basic goal of the weighted dictionary problem is to organise the stored elements, so that elements with high access probability be nearer the entry point of an access operation (the head of a list or the root of a tree) than elements with low access probability. In this way, the cost of an access operation and consequently, the cost of an insert and delete operation becomes much less. Therefore, by taking advantage of the access distribution of the elements, we can perform the primitive operations of the dictionary problem (access, insert, and delete) much more efficiently. We can show this for a simple list structure.

Suppose that the access probability p_i of the element $x_i \in S$ is given by $p_i = \frac{1}{i \cdot H_N}$ (Zipf's Law) (see [15], [86]), where $H_N = \sum_{i=1}^{N} \frac{1}{i}$ is the Nth Harmonic number. It is obvious that it holds $1 > p_1 > p_2 > \ldots > p_N > 0$. If we store the elements x_i in a list structure in random order, ignoring their access probabilities,

the expected cost $E(c)$ of an access operation is

$$E(c) = \sum_{i=1}^{N} p_i \cdot \left\{ \sum_{j=1}^{N} c \cdot j \cdot \frac{1}{N} \right\} =$$

$$\sum_{i=1}^{N} p_i \cdot \frac{c}{N} \frac{N(N+1)}{2} =$$

$$c\frac{N+1}{2} = O(N) \tag{5.9}$$

In deriving the above linear expected cost, we assumed that the access operation is performed by comparing the elements of the list, one by one, with the requested element, starting from the head, until we find it. Each comparison costs c time units. We also used the relation $\sum_{i=1}^{N} p_i = 1$.

If we now store the elements according to their access probability, i.e. x_i is stored in the ith node of the list, then the above expected cost becomes

$$E(c) = \sum_{i=1}^{N} c \cdot i \cdot p_i = c \cdot \sum_{i=1}^{N} i \frac{1}{i \cdot H_N} = c\frac{N}{H_N} \tag{5.10}$$

According to [115], it holds that $H_N = \ln N + \Theta(1)$. Therefore, relation (5.10) becomes $E(c) = O(\frac{N}{\ln N})$. It is thus obvious that by making use of the access probabilities of the elements in S, we managed to reduce the expected cost considerably. Actually, the ordering of the elements in a list according to their access probabilities, is optimal in terms of the expected cost of the access operation. However, the access probabilities must be known *a priori*. This is almost impossible in a real application. Thus, we have to resort to other solutions which are near optimal.

A class of such solutions are called permutation rules and are used to reorder (reorganise) the list when an element is accessed in it, so that the list structure will eventually adapt to the access distribution, speeding up subsequent accesses to the list. A permutation rule is defined by $\{\pi_i : 1 \leq i \leq N\}$, i.e. a set of permutations over N elements. When the kth element in the list is accessed, permutation π_k is used to reorder the list. Two of the most important near optimal permutation rules are the *move to front* rule and the *transposition* rule.

In the move to front rule, after the kth element is accessed in the list it is moved to the head of the list. All elements which it passes over move down one

position. For example, suppose

$$L_t : x_{i_1^t} \rightarrow x_{i_2^t} \rightarrow \ldots x_{i_k^t} \rightarrow \ldots x_{i_N^t}$$

be the structure of the list before accessing element $x_{i_k^t}$ and $L_{t'}$ be the corresponding structure after this access. Then

$$L_{t'} : x_{i_k^t} \rightarrow x_{i_1^t} \rightarrow x_{i_2^t} \rightarrow \ldots x_{i_N^t}$$

In the transposition rule, the accessed element is transposed with the one above it. Suppose that L_t (defined above) is the structure of the list before accessing its kth element. Then, the corresponding $L_{t'}$ now becomes

$$L_{t'} : x_{i_1^t} \rightarrow x_{i_2^t} \rightarrow \ldots x_{i_k^t} \rightarrow x_{i_{k-1}^t} \rightarrow \ldots x_{i_N^t}$$

The goal of both permutation rules is to move the most frequently accessed elements nearer the head of the list. In this way, subsequent accesses to these elements become cheaper. However, as it was shown in [15], the move to front rule approaches much more rapidly its asymptotic cost (least cost). For every permutation rule, it takes some time before the structure of the list adapts to the access distribution and the rule reveals its best performance, provided that the access probabilities do not change with time (see Figure **5**.19; the graph is taken from [15]). The intuitive reasoning for the rapid convergence of the move to front rule is the following. In the initial random ordering of the elements in a list, many elements with high access probability may be far down in the list. The move to front rule swiftly relocates these elements near the head of the list. In contrast, the transposition rule allows elements to rise only one position per access, and therefore, the cost decreases slowly. In a real application, fast convergence is an appealing property and thus, the move to front rule is better suited to most applications.

Another class of rules, which have been used to change the structure of the list after an access operation, so that it adapts to the access distribution, consists of rules that use extra space to store the estimated access probabilities. These rules are called counter rules (see [91]). Here, the node at the ith position in the list stores not only the ith element, but also a counter which corresponds to the

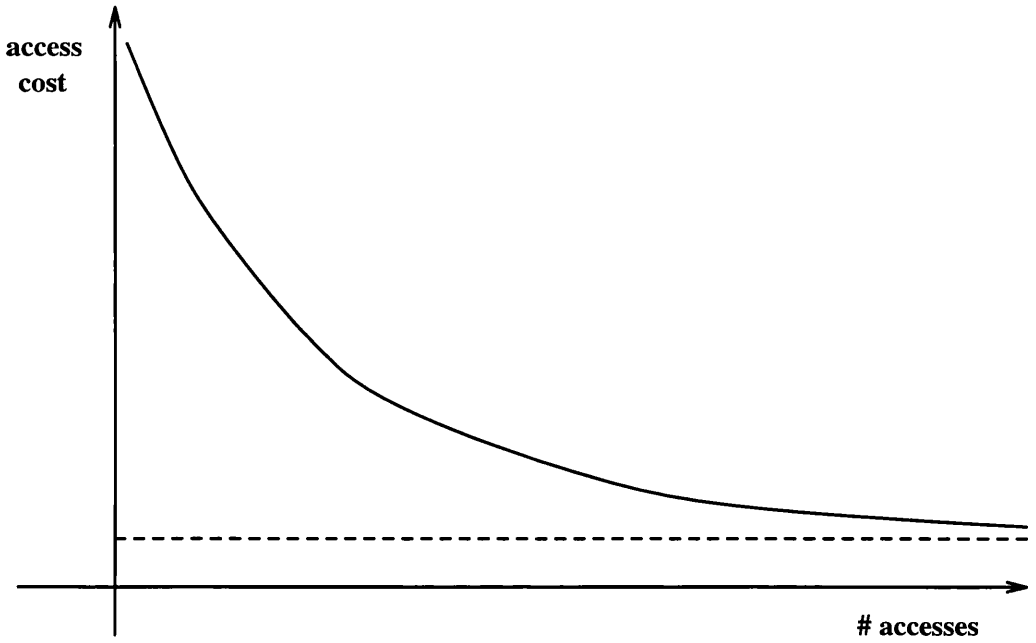


Figure 5.19: The cost curve of a permutation rule and its asymptote.

estimated access probability of this element. All these rules are based on the law of large numbers. In reality, each counter is a function of the number of times k the corresponding element was requested within the access sequence in the past. Each time an element is accessed, the corresponding counter is increased by one. If p is the access probability of this element, then, according to the law of large numbers, for a sufficiently long sequence of n accesses it holds that

$$\forall \epsilon > 0 \quad Prob\left\{\left|\frac{k}{n} - p\right| \leq \epsilon\right\} \longrightarrow 1, \quad n \longrightarrow \infty \tag{5.11}$$

Therefore, the ratio $\frac{k}{n}$ approximates the corresponding access probability p and as the number of accesses n increases, the approximation error becomes arbitrarily small. The application of the counter rules theoretically gives better performance than the permutation rules, but has the disadvantage of the extra space needed to store the counters which is $O(N)$, where N is the length of the list.

The most representative rule of this kind is the *frequency count* rule, which keeps the elements of the list sorted according to their counters. In this rule, each counter is equal to the number of times the corresponding element was requested. The reader should bear in mind that the optimal ordering of the elements in a list is according to their access probabilities in decreasing order, starting from the head of the list. The frequency count rule is based on this

fact to give asymptotically optimal performance (see [15]). Because of the law of large numbers (see (5.11)), if $p_1 > p_2$ the probability that element x_1 has been requested more times than element x_2, and therefore is ahead of x_2 in the list, approaches 1 as the length of the access sequence increases. Thus, the frequency count rule guarantees that the cost for accessing x_1 (more frequently accessed element) will be less than that for accessing x_2 (less frequently accessed element).

Real texture is not a completely random process, i.e. the pixels are spatially correlated. In other words, the probability of co-occurrence of two grey levels at a certain interpixel distance and orientation in a given real texture is not independent and the same for all grey level pairs. Therefore, the estimation of the second-order probabilities in the SGLDM is an example of the weighted dictionary problem. By taking advantage of the different access probabilities of the grey level pairs, which are equal to the corresponding co-occurrence probabilities, we may improve the computational time for the feature extraction stage in this texture analysis method. Therefore, an enhanced version of the co-occurrence trees is proposed in Chapter 7, which employs additional memory space to store the most frequently occurring grey level pairs in the analysed texture in a separate structure, in order that subsequent accesses to them are cheaper. However, the space complexity of this new approach has to be computed and the net advantage of using it, considering both time and space, needs to be investigated.

5.5 Self-adjusting binary search trees

The weighted dictionary problem can be expanded to binary search trees, as well. The goal is to put the elements, which have the largest access probabilities, nearer the root of the tree, so as to reduce their access time. In this way, we will reduce the total access time to the binary tree.

Let $S = \{x_1, x_2, \ldots, x_N\}$ be the set of elements stored in the binary tree, where $x_1 < x_2 < \ldots < x_N$. Similarly to Section 5.4, each element x_i is assigned a positive integer w_i, which represents the access probability p_i for this element. Let $W = \sum_{i=1}^{N} w_i$. Then, $p_i = \frac{w_i}{W}$. A measure of the average access time in this case is given by $\sum_i w_i \cdot \frac{d_i}{W}$, where d_i is the depth of the element x_i in the search tree.

Of interest is the quantity $\sum_i w_i \cdot d_i$, which is called the total weighted depth and which should be kept as small as possible. Unterauer ([168]) claimed that the aim should be the minimisation of the quantity $\sum_i p_i \cdot d_i$, which is called the weighted path length. It is obvious that this expression is equivalent to the total weighted depth.

According to a theorem in [13], in any binary search tree the total weighted depth satisfies the relation $\sum_i w_i \cdot d_i \geq W \cdot \sum_i p_i \cdot \log\left(\frac{1}{p_i}\right)$. Based on this inequality, Bent et al. ([13]) concluded that any binary search tree having the property $d_i = O\left(\log\left(\frac{W}{w_i}\right)\right)$, $\forall i$, has minimum average access time to within a constant factor. They called the quantity $O\left(\log\left(\frac{W}{w_i}\right)\right)$ the ideal access time for element x_i. If the weights w_i are the same for all elements x_i then $d_i = O(\log N)$, where N is the cardinality of the set S. Actually, this is the depth of the Balanced Binary tree, so BB-tree has the ideal access time property for elements having a uniform access distribution.

Given a set of elements S and their corresponding weights (access frequencies), we can always find a static binary search tree for this set which has the minimum total weighted depth. This binary tree is called the optimal search tree for set S. For example, suppose that $S = \{x_1, x_2, x_3, x_4, x_5\}$, where $x_1 < x_2 < x_3 < x_4 < x_5$. Also, suppose that for their corresponding weights w_i, $i = 1, \ldots, 5$, it holds that $w_1 < w_5 < w_3 < w_4 < w_2$. Then, the optimal search tree for the above set of elements is shown in Figure **5.20** and the minimum total weighted depth for set S is given by relation:

$$\sum_{i=1}^{5} w_i \cdot d_i = w_1 + w_4 + 2w_3 + 2w_5 = w_1 + 2(w_3 + w_5) + w_4 \qquad (\mathbf{5}.12)$$

The average access time for the optimal search tree is not known in closed algebraic form but in [2] it was shown that

$$H - \log H - \log e + 1 \leq t_{opt} \leq H + 1 \qquad (\mathbf{5}.13)$$

for $H \geq 1$, where t_{opt} is the average access time and H is the entropy of the access distribution of the elements in S, i.e. $H = -\sum_{i=1}^{N} p_i \cdot \log p_i$.

There are two problems associated with the optimal search trees. The first problem is the requirement to know the access distribution of the stored elements,

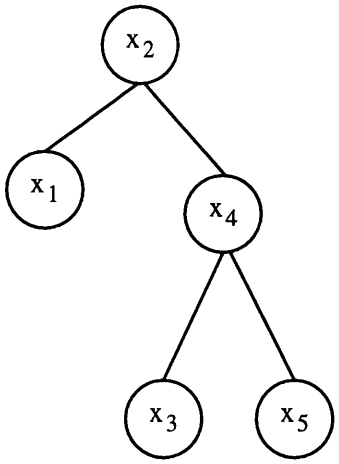


Figure 5.20: An example of an optimal search tree.

a-priori. The second problem is the time needed for the construction of this static tree, which is $O(N^2)$ (see [86]). However, most of the real problems such as the texture analysis of an image, are dynamic in nature. Even if the underlying access distribution model were known, the time for building the optimal tree would be prohibitive.

Allen and Munro ([2]) claimed that, in order that the binary search trees be used in the weighted dictionary problem, a way must be devised to dynamically modify a tree, so that the average access time will be as small as possible. They proposed a restructuring operation called *move to root* which attempts to raise the most frequently accessed elements higher up, nearer the root of the tree. They claimed that this operation gives a near optimal solution, that is, the average access time is within a constant factor of that of an optimal static binary search tree. The binary search trees, which use such restructuring operations to dynamically adapt their structure to the distribution of the accessed elements, in order to minimise the average search time, are called **self-adjusting** or **self-organising** binary search trees. A binary tree, which has been recently proposed for the weighted dictionary problem and has the above property, is the **splay** tree.

5.5.1 Splay trees

A splay tree is a self-adjusting binary search tree, which employs a restructuring operation called *splaying* after every access (see [141]). This restructuring

operation consists of a sequence of rotations along the path from the accessed node to the root, which move the accessed element up to the root of the tree. Actually, performing this restructuring operation at a node v of the splay tree, or else splaying the tree at a node v, comprises three cases:

1. **zig case**: If $p(v)$ (the parent of v) is the root of the tree, the splaying algorithm rotates the edge joining v with $p(v)$ and concludes. It is equivalent to one single rotation (see Figure **5**.21).

2. **zig-zig case**: If $p(v)$ is not the root and v and $p(v)$ are both left or both right children, the algorithm first rotates the edge joining $p(v)$ with $p^2(v)$ (the grandparent of v) and then it rotates the edge joining v with $p(v)$. It is equivalent to two single rotations (see Figure **5**.22).

3. **zig-zag case**: If $p(v)$ is not the root of the splay tree and v is a left child while $p(v)$ is a right child or vice versa, the algorithm first rotates the edge joining v with $p(v)$ and then it rotates the edge joining v with the new $p(v)$. It is equivalent to one double rotation (see Figure **5**.23).

The symmetrical variants of the cases presented above are omitted.

The move to root restructuring operation proposed by Allen and Munro ([2]), rotates the edge joining v to its parent $p(v)$ and repeats this step, until v becomes the root of the tree. Therefore, it is a sequence of single rotations, such as the one shown in Figure **5**.21. The restructuring operation presented in this section, splaying, is similar to the move to root operation in that it does rotations bottom up, along the access path and moves the accessed element up to the root. However, it differs in that it does the rotations in pairs, in an order that depends on the structure of the access path. Based on this structure, it either performs two single rotations or it performs one double rotation each time. Therefore, splaying not only moves the node storing the requested element up to the root, but roughly halves the depth of every node along the access path. This halving effect is a very strong property and makes splaying more efficient in computational time than the move to root operation.

We will now present the implementation of the two primitive operations that are of interest in texture analysis, i.e. the access and insert operation.

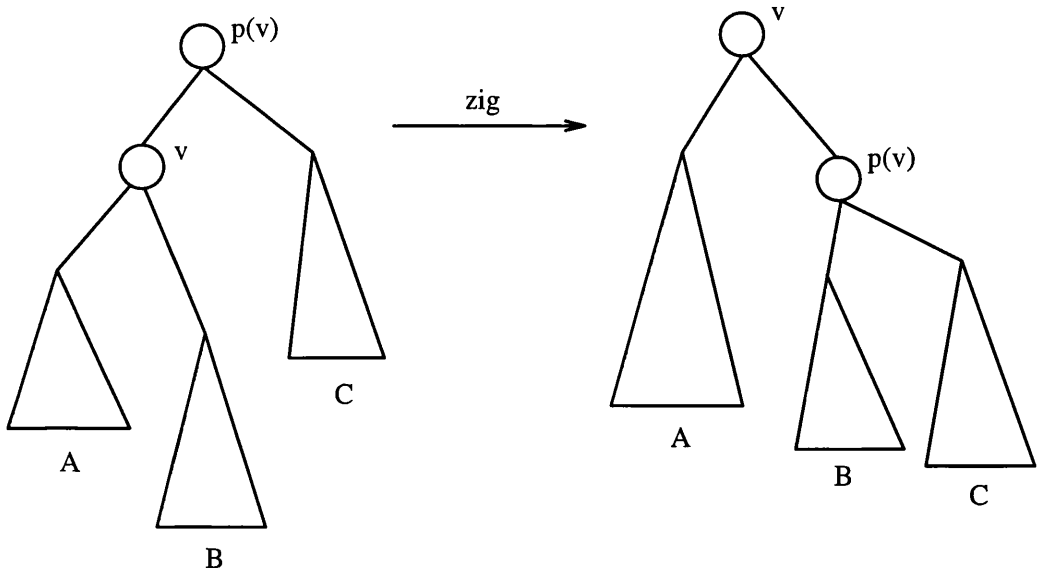


Figure 5.21: The zig case in splaying.

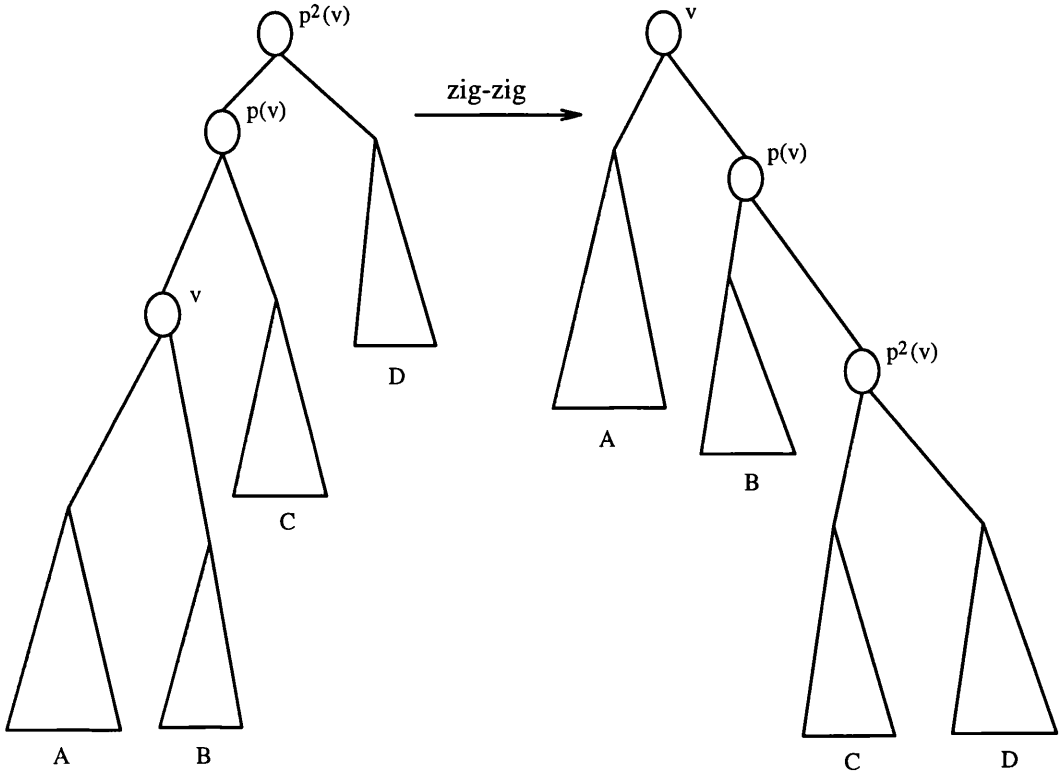


Figure 5.22: The zig-zig case in splaying.

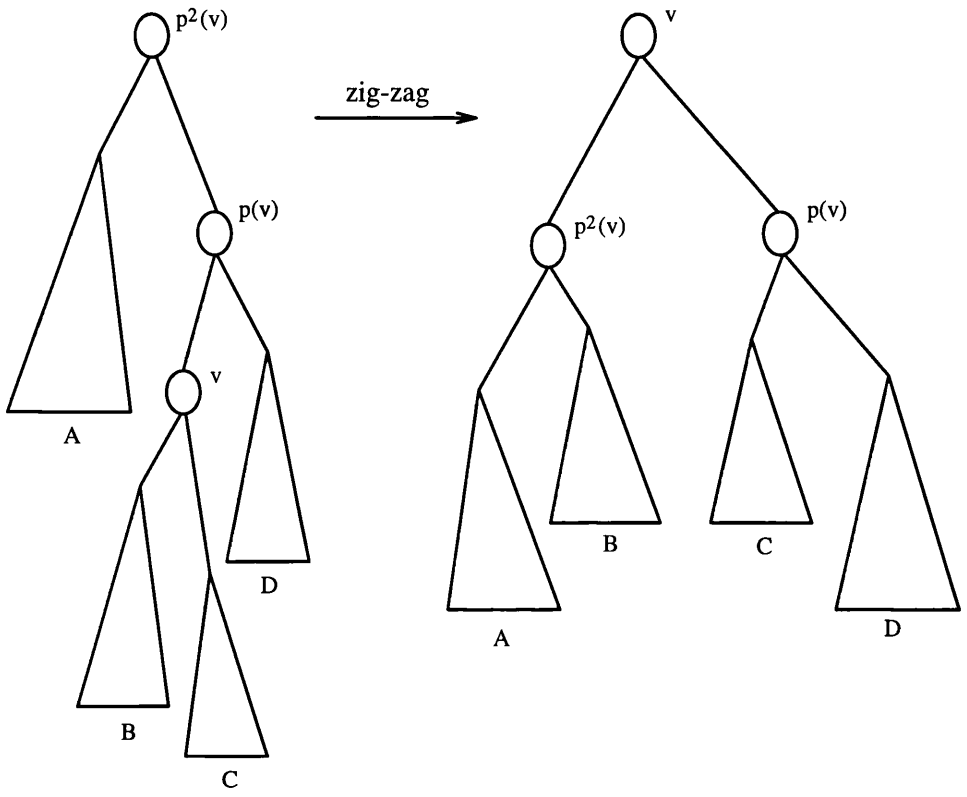


Figure 5.23: The zig-zag case in splaying.

Access(x): $v \leftarrow root$;
WHILE ($val(v) \neq x$ **AND** v is not a leaf) **DO**
IF ($val(v) > x$)
$v \leftarrow lson(v)$;
ELSE
$v \leftarrow rson(v)$;
ENDIF
ENDWHILE
IF ($val(v) \neq x$)
RETURN (**FALSE**,v);
ELSE
splay(v);
RETURN (**TRUE**,v);
ENDIF

The above pseudocode is similar to the one presented in Section **5.2**. The only difference is the splaying restructuring operation which is performed after a successful search by calling function $splay(\cdot)$.

We will now show an example of the access operation in a splay tree. Suppose that we have the splay tree shown in Figure **5.24**. We want to perform the $access(60)$ operation. The path followed by the access algorithm is shown in Figure **5.25**. After the successful access of element 60, the splaying restructuring operation is performed at the node containing this element. The structure of the splay tree after the execution of the above splaying operation, is shown in Figure **5.26**. Note that the root node now contains the element 60.

Similarly, the insert primitive operation is implemented as in Section **5.2**:

$Insert(x, f)$: $found \leftarrow Access(x)$;
IF ($found.1$ =**TRUE**)
$update(found.2, f)$;
$splay(found.2)$;
ELSE
$splay(add(x, found.2, f))$;
ENDIF

Here, the $add(\cdot)$ subroutine returns the new node that stores element x. The only difference between the above pseudocode and the one presented in Section **5.2** is that after the completion of the insert operation, splaying at the updated or inserted node is performed.

It is difficult to estimate the time complexity of the access operation in splay trees, especially for a non-uniform access distribution. The interested reader should refer to [141]. When all elements have the same access probability (uniform access distribution), the amortised time complexity of the access operation is $O(\log N)$ for a splay tree having N nodes. As we said before, the amortised time is the time average over a worst case sequence of access operations. The above complexity shows that splay trees are theoretically as efficient as the balanced trees, such as the BB-tree, when amortised running time is the performance measure. Also, on any sufficiently long sequence of access operations splay trees are as efficient, to within a constant factor, as optimal static search trees (see [157]).

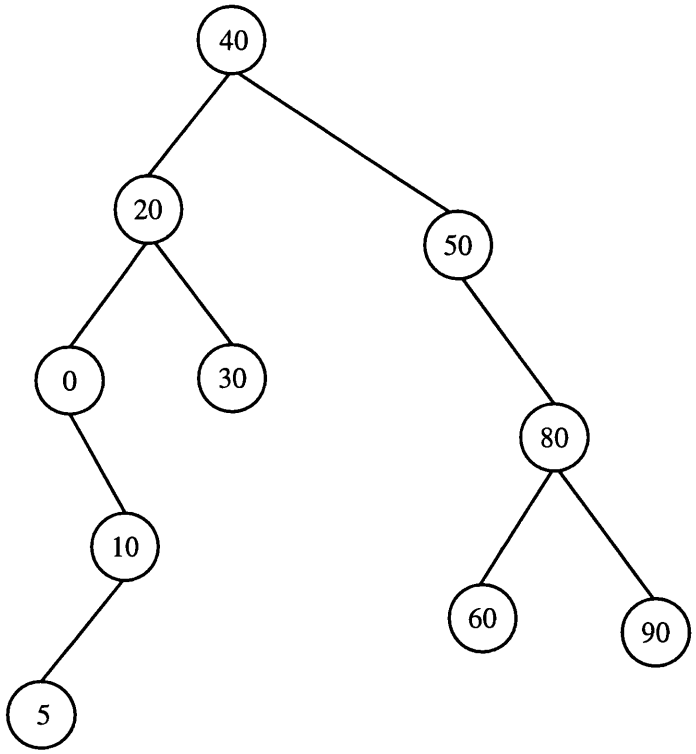


Figure 5.24: An example of a splay tree.

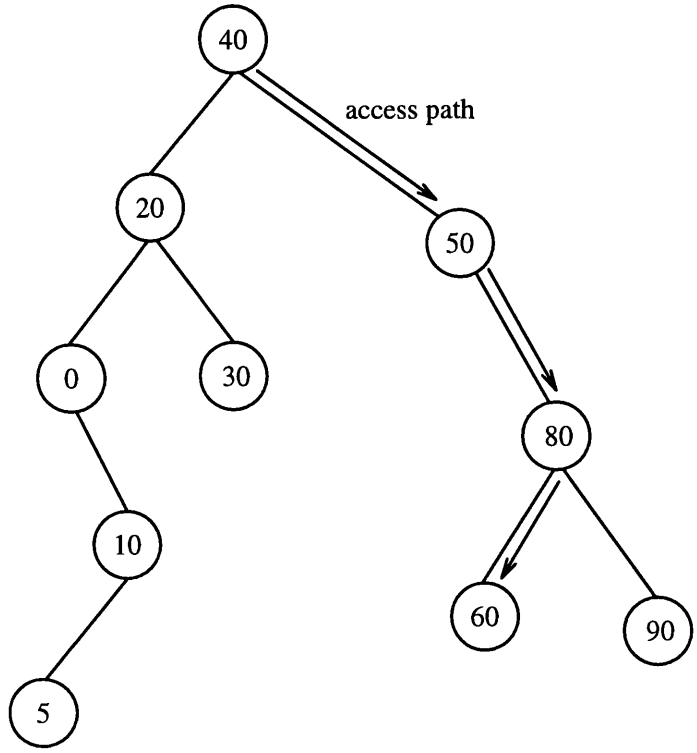


Figure 5.25: The route for accessing element 60 in the illustrated splay tree.

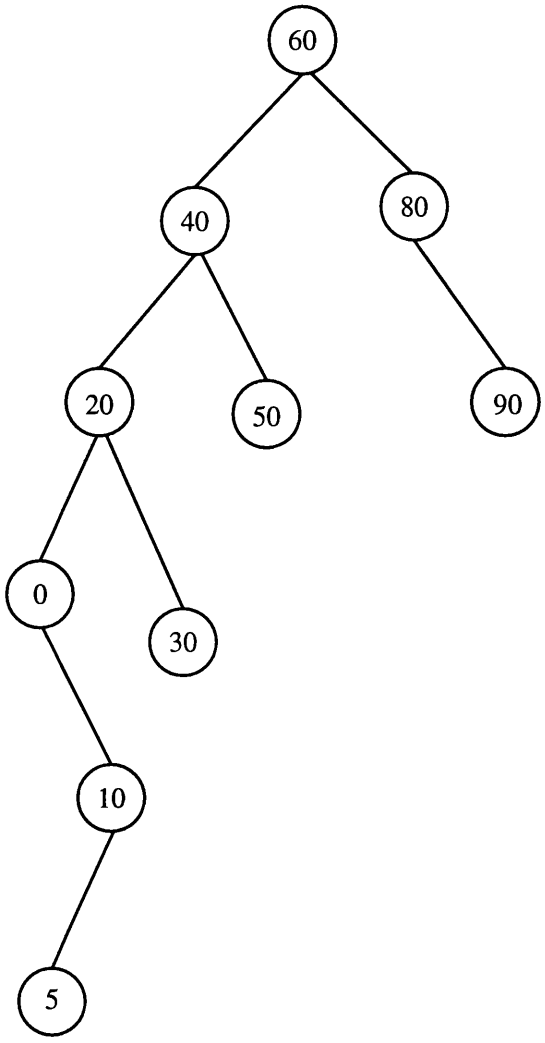


Figure 5.26: The structure of the tree after splaying at node 60.

There are several advantages and disadvantages using the splay tree instead of the Balanced Binary tree. The advantages are the following:

- Balanced Binary trees (BB-trees) are not as efficient as possible, if the access distribution of the elements is non-uniform. In contrast, splay trees dynamically adapt to the access distribution and therefore, they are theoretically more efficient.

- BB-trees need extra space for the storage of balance information which is $O(N)$, where N is the number of stored elements. In contrast, splay trees do not need any balance information.

- Splay trees employ much simpler restructuring operations. Consequently, their access and update (insert and delete) algorithms are significantly simpler and easier to implement.

Two are the most important disadvantages of splay trees:

- They require structural changes (rotations) at every successful access, not only during the update operations which is the case for the BB-trees. Also, these changes continue higher up, along the access path and stop only when the root is reached.

- Although the amortised time complexity of an access operation in a splay tree is $O(\log N)$, the individual accesses within the sequence can be very expensive. Their cost may be unacceptable for certain real-time applications.

Chapter 6

Plain co-occurrence trees

6.1 Description

The texture feature extraction stage of a statistical texture analysis system is the one that transforms the texture information of an image into a feature vector of scalar quantities. We have already mentioned that this stage is one of the most (if not the most) computationally demanding stages of the whole process (see Section **1**.3). Speeding up this stage will make texture classification, texture segmentation, and generally, texture analysis to run faster.

Time and space are critical factors in most practical applications and determine the viability of a particular algorithm. That is, a certain algorithm may give the best solution to a certain problem, but if the above factors get unacceptably high values, the algorithm will only be of theoretical value. Estimating and reducing the time and space complexity of algorithms is a central problem in computer science and engineering.

In this thesis, a novel dynamic scheme for organising the textural information extracted by the SGLDM is proposed. This scheme, which is called the **co-occurrence trees** ([151]), is able to overcome the memory space and time inefficiency of the co-occurrence matrix mentioned in Section **4**.2. This is because its size depends only on the number of distinct grey level pairs found in the analysed image region. In fact, this is accomplished without reducing the textural information captured by the estimated second-order probabilities of the

SGLDM in any way. Actually, the efficiency of the proposed scheme is based on the elimination of the large sparsity of the co-occurrence matrix. In most cases, a large number of the entries of this matrix are zero which contribute nothing to the texture feature extraction process. The consequence is that a large amount of memory space is allocated for redundant information. In addition, a large portion of the time for the computation of the texture features is spent for processing these unnecessary zero entries.

More specifically, the proposed scheme solves the aforementioned problems in the SGLDM, one of the best texture analysis methods, by substituting the inefficient co-occurrence matrix with a dynamic set of appropriate dynamic data structures. In this chapter, we describe our first approach which is called the **plain co-occurrence trees**. The basic element of the dynamic set in this approach is a modified version of the Balanced Binary tree (BB-tree) (see Section **5**.3). In particular, the node-oriented BB-tree is used in order to maximise the efficiency of the proposed approach in terms of memory space (see Section **5**.2). Actually, this dynamic set is a forest of BB-trees. There are five main reasons why this particular dynamic data structure was chosen:

- **time efficiency:** The basic operations of the dictionary problem (access, insert, delete) cost $O(\log N)$ time, where N is the cardinality of the elements stored in the tree. This logarithmic upper bound is optimal for binary trees and is a consequence of the fact that the depth of the tree is guaranteed to be $O(\log N)$ in all cases (balanced tree), under random insertions and deletions ([110]). In addition, the BB-tree has the important property that the rebalancing cost, i.e. the cost of the operation for keeping the tree balanced, in the amortised case is only $O(1)$ (see Section **5**.3).

- **space efficiency:** BB-tree is the only tree that needs only one bit per node to store the information necessary for rebalancing (balance information). In the proposed approach, this bit can be embedded in the information content (the information stored in each node) in many cases, especially for images with a dynamic range higher than 8 bits. So, no additional space is needed.

- **simplicity:** It is one of the simplest trees in terms of structure and in terms of rebalancing rules and operations (dictionary and rebalancing) defined on it.

- **ease of implementation:** It is one of its strongest points regarding its applicability.

- **direct representation:** A direct correspondence between the co-occurrence matrix and the co-occurrence trees exists. There is no loss of intuition as to what textural information is stored in the trees.

According to [155], [162], [163], BB-tree is the best choice for practical applications requiring a balanced search tree in main memory.

Let n_g be the number of grey levels in the analysed region. Then, the co-occurrence trees consist of at most n_g BB-trees. The actual number of BB-trees depends on the spatial relationship (d, ϑ) the grey level pairs need to satisfy in a specific image region. In other words, it depends on the number of distinct grey levels $i \in \{0, \ldots, n_g - 1\}$ in the region, for which there exists grey level j so that the pair (i, j) satisfies the spatial relationship (d, ϑ). It is obvious that the maximum number of grey levels i is n_g. For each such grey level i there exists a BB-tree which we denote T_i. T_i stores the distinct grey levels j in the analysed region, for which pairs (i, j) satisfy the spatial relationship (d, ϑ). For each such j there is a node in tree T_i. We will show the above by an example.

Suppose that we have the following 8×8 image having 8 grey levels:

0	1	0	1	0	2	0	3
3	1	4	2	0	2	0	6
5	7	5	3	0	0	0	1
2	5	7	3	0	1	3	3
2	0	2	3	4	5	0	3
0	0	0	2	0	0	3	0
7	0	0	2	0	1	0	1
2	1	0	0	2	0	1	0

In Section **4**.1, we mentioned that an alternative way to express the spatial relationship is using the displacement vector form $\overline{d} = (dx, dy)$, where dx, dy are integers which correspond to separation along the x- and y- axis, respectively. Suppose that in our example the spatial relationship that the pixel pairs need to satisfy, expressed in this displacement vector form, is $(0, 1)$ (one pixel in the horizontal direction; see Section **4**.1). Also, suppose that we make the distinction between 0° and 180° direction (distinction between positive and negative distances; see also Section **4**.1), that is, only the 0° direction is considered. Then, the creation process of the co-occurrence trees is shown in Figures **6**.1 and **6**.2 [1]. The corresponding co-occurrence matrix is shown below:

	0	1	2	3	4	5	6	7
0	*	*	*	*	0	0	*	0
1	*	0	0	*	*	0	0	0
2	*	*	0	*	0	*	0	0
3	*	*	0	*	*	0	0	0
4	0	0	*	0	0	*	0	0
5	*	0	0	*	0	0	0	*
6	0	0	0	0	0	0	0	0
7	*	0	0	*	0	*	0	0

In the above table, symbol '*' denotes a non-zero entry. The (i, j) entry (row i, column j) gives the number of times grey level i is immediately on the left side of grey level j in the above image.

In the creation process shown in Figures **6**.1 and **6**.2, only the grey level pairs that change the structure of the trees (that cause the creation of additional nodes) appear. The other pixel pairs that satisfy the specific spatial relationship $(0, 1)$ increase a counter in the relevant (already existing) node. For example, the second $(0, 1)$ grey level pair causes an increase of the counter stored in the node corresponding to grey level 1 in BB-tree T_0. So, each node stores not only the

[1] $\emptyset$ denotes a null tree, squares denote leaves, circles denote internal nodes, and coloured circles denote red nodes

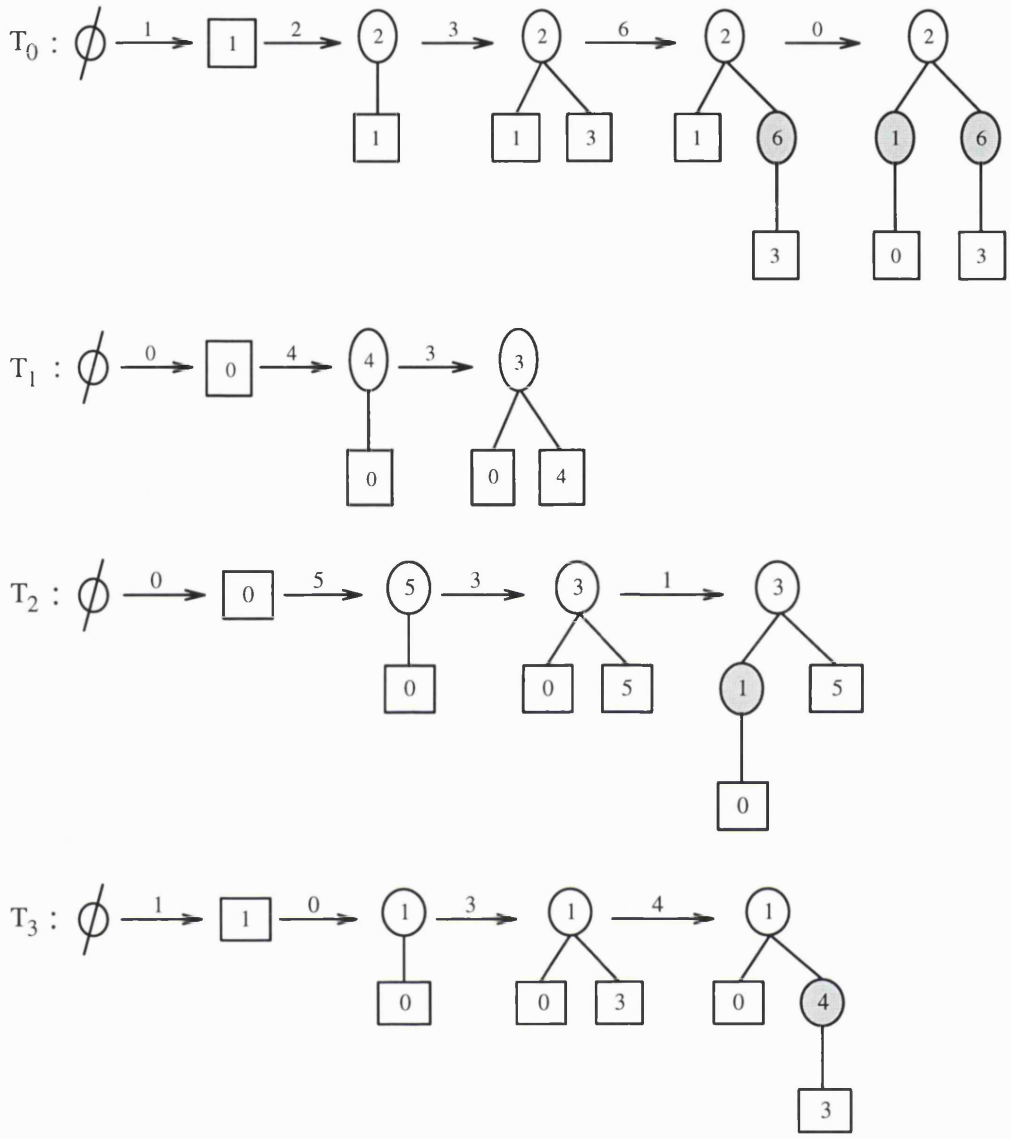


Figure **6**.1: An example of the creation of the co-occurrence trees.

corresponding grey level but also a counter. After the conclusion of the creation process of the co-occurrence trees, these counters are equal to the unnormalised co-occurrence frequencies of the grey level pairs in the analysed region that satisfy the specific spatial relationship. Therefore, they contain the same information as the co-occurrence matrix, but without the redundant zero entries.

Note that, in the above example row 6 of the co-occurrence matrix contains nothing but zero values. The corresponding tree T_6 in the co-occurrence trees is null, i.e. it does not exist. In contrast, co-occurrence matrix stores a whole row of zero entries for this grey level. This is a very common situation in real

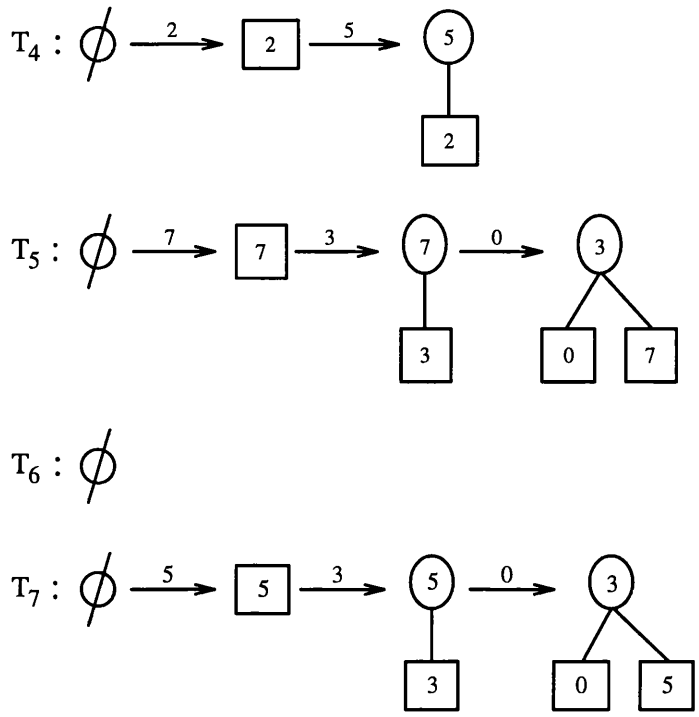


Figure **6**.2: An example of the creation of the co-occurrence trees cont.

applications. Also, from the above example the correspondence between the rows of the co-occurrence matrix and the trees in the proposed approach should be clear.

We have mentioned that it is important to efficiently implement the elementary operations which comprise the core of the computation of the 13 texture features (see Section **4**.1). One of these operations is $p_y(j) = \sum_i p(i,j)$. In order to compute $p_y(j)$ in the co-occurrence trees, we need to access each non empty tree T_i and search for the node containing grey level j. However, accessing a tree takes logarithmic time in the worst case. In fact, we can improve this time for the above operation by using a set of list data structures, in addition to BB-trees. This way, we can get a constant ($O(1)$) access time for each $p(i,j)$ in the above operation. This set of list structures is built upon the trees of the proposed approach as follows. Let (i,j) be one grey level pair that satisfies the given spatial relationship (d,ϑ) in the analysed region. Then, there is a list L_j which contains a node that corresponds to grey level i. This node is actually the node containing grey level j in tree T_i. For our previous example, lists L_0 and L_3 are shown in Figure **6**.3.

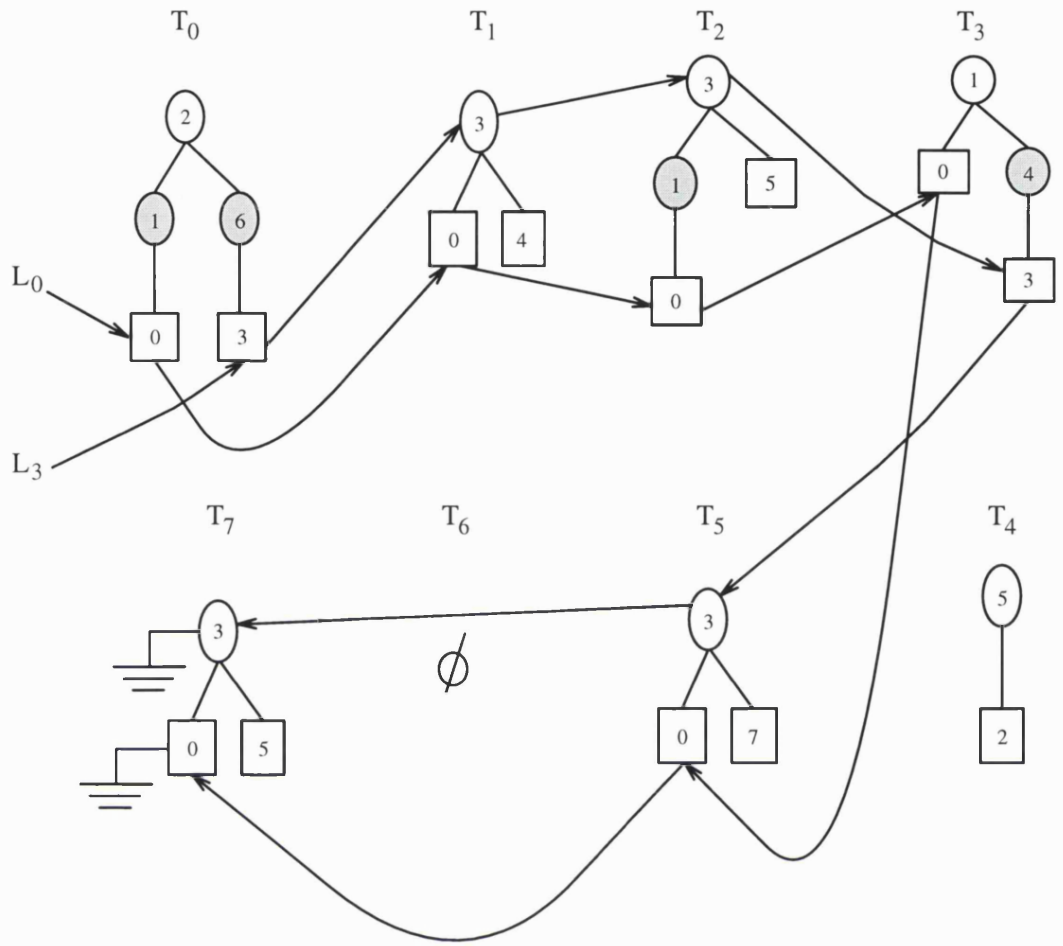


Figure **6**.3: An example of the list structures in the co-occurrence trees.

Each node of the co-occurrence trees belongs to one and only one list. That is, the nodes that contain grey level j belong only to list L_j. We can say that the above set of lists partitions the set of nodes in the BB trees into disjoint subsets, according to the grey level stored in them. For the implementation of this set of lists one additional field is stored in each node, which contains a pointer to the next node in the list to which it belongs.

Finally, two additional fields exist in each node, which are actually pointers to its left and right child. If one or both children do not exist the null pointer is stored in the relevant field. These pointers are used for building the BB-trees. The balance information in each node, as we have already mentioned, is just one bit in the case of the BB-trees, which corresponds to the colour of the node (red or black). This bit can be embedded in the field that stores the grey level of the node, in many practical applications. Especially, for images with a dynamic

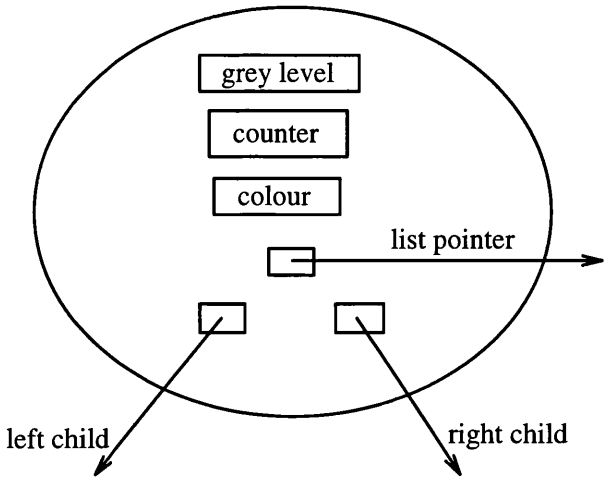


Figure 6.4: Structure of a BB-tree node.

range higher than 8 bits, it is very unlikely that grey levels greater than $2^{15} - 1$ be found in the image. Since two bytes are needed to store the grey levels in this case, there is an unused bit which can be employed to store the balance information. Otherwise, an additional byte is needed for each node (e.g. images with 8-bit dynamic range). The structure of the node in the BB-trees is shown in Figure 6.4.

Another elementary operation is $p_x(i) = \sum_j p(i,j)$. In order to implement this operation, tree T_i that corresponds to grey level i needs to be accessed. Actually, each node of T_i (more specifically its counter) contributes to the computation of the above sum. Thus, an efficient way of traversing the nodes of tree T_i needs to be employed.

We saw in Section 5.2 that there are three methods for traversing a binary tree, namely preorder traversal, postorder traversal, and symmetrical or lexicographic traversal. One way to implement any of these traversals is by using recursion (a function calls itself recursively until a condition is fulfilled). This is not, though, the best way to do that, because of the large computational time and space requirements to save and restore the environment (e.g. local variables) of the calling function. The most efficient way to implement a traversal is to use a stack data structure, where the nodes of the tree (actually pointers to the nodes) are stored as they are traversed.

From the aforementioned traversing methods, the symmetrical traversal is the most appropriate, since it has been specifically designed for the binary trees. As we saw in Section 5.2, this method first visits the left subtree of the root, then the root, and finally, the right subtree. This process is, in turn, performed in each subtree. A pseudocode for the symmetrical traversal employing a stack data structure, is shown below:

```
x ← root;
IF (x =NULL)
    EXIT;
ENDIF
push(x);
WHILE (stack_pointer > 0) DO
    WHILE (lson(x) ≠NULL) DO
        push(lson(x));
        x ← lson(x);
    ENDWHILE
    pop(x); eject(x);
    WHILE (x ≠NULL AND rson(x) =NULL) DO
        pop(x);
        IF (x ≠NULL)
            eject(x);
        ENDIF
    ENDWHILE
    IF (x ≠NULL)
        push(rson(x));
        x ← rson(x);
    ENDIF
ENDWHILE
```

In the above pseudocode, $push(\cdot)$ and $pop(\cdot)$ functions are the standard functions for manipulating a stack. Function $lson(\cdot)$ returns a pointer to the left child of its argument or null, if it does not exist. Function $rson(\cdot)$ does the same for the

right child. Function $eject(\cdot)$ returns the node that is currently traversed. The order in which the nodes of a tree are ejected is the symmetrical or lexicographic order (see Figure **5**.5).

The other two elementary operations, namely $P_{x+y}(k)$ defined by (**4**.26) and $P_{x-y}(k)$ defined by (**4**.27), are computed in a straightforward way. For example, in order to compute $p_{x+y}(k)$, we need to search for grey level $k-i$ in tree T_i, if T_i exists, for each grey level i. Each search takes logarithmic time. Actually, only the trees T_i for which $0 \leq i \leq n_g - 1$ and $k - n_g + 1 \leq i \leq k$ hold at the same time, need to be searched, where n_g is the number of grey levels in the analysed region (see proof of Lemma **6**.2.4). Elementary operation $p_{x-y}(k)$ is more complex, due to the condition $|i-j| = k$ that must be satisfied. In this case, we need to search for grey level $i+k$ in BB-trees T_i, where $0 \leq i \leq n_g - k - 1$ and(or) for $i-k$ in BB-trees T_i, where $k \leq i \leq n_g - 1$ (see proof of Lemma **6**.2.5). Again, only the BB-trees T_i that are not null, are searched. Each search takes logarithmic time here, as well.

In order to access the BB-trees or the set of the lists defined above, a way must be devised to organise the pointers to the roots of the trees and the heads of the lists. Two ways are proposed and evaluated in this work, which correspond to two different methods of implementing the proposed approach (two versions of the co-occurrence trees):

- **semi-dynamic** version: Here, the pointers to the roots of the BB-trees and the heads of the lists are stored in two separate static arrays; one for the roots and another one for the lists. This means that pointers are stored even in the case where the corresponding BB-trees or lists are empty. If N_g is the number of grey levels in the entire analysed image, the static arrays need memory space proportional to N_g. The semi-dynamic version is shown in Figure **6**.5.

- **full-dynamic** version: In this case, the pointers to the roots of the BB-trees and the heads of the lists are organised into two separate BB-trees. Only the pointers to non empty trees and lists are stored. The full-dynamic version is shown in Figure **6**.6.

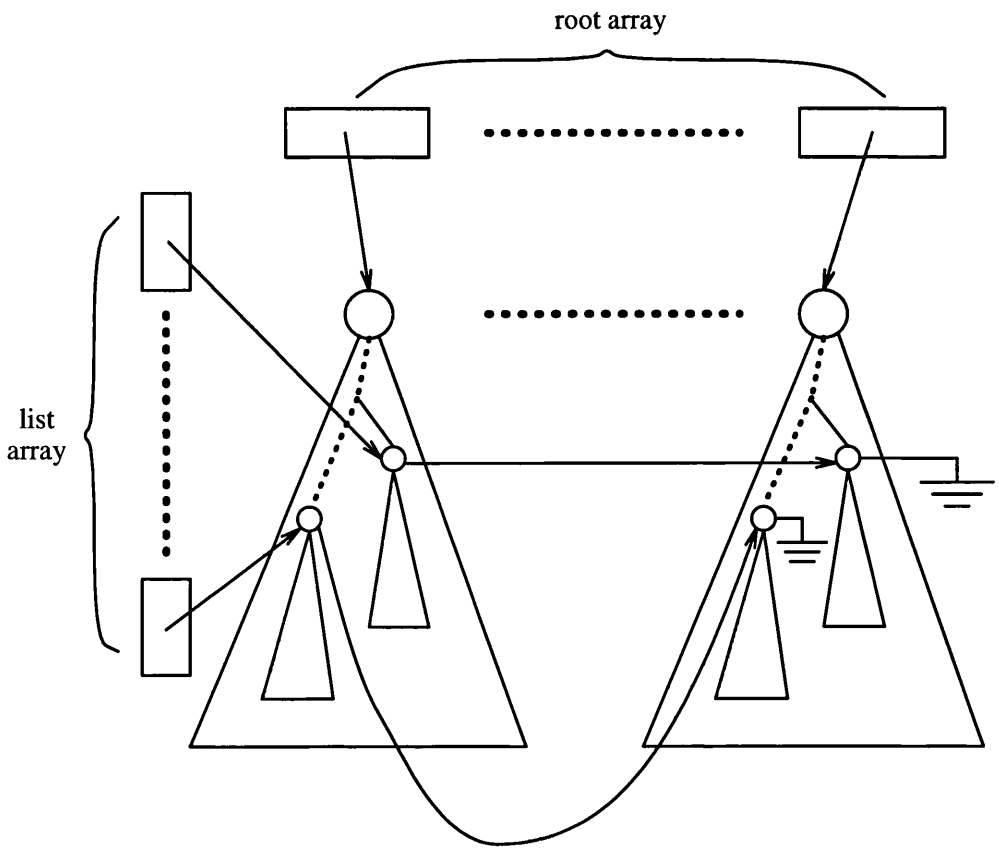


Figure 6.5: Structure of the semi-dynamic version.

Accessing grey level pair (i, j) is performed in the following way. First, the algorithm gets the pointer to the root of the BB-tree T_i directly from the root array, in the case of the semi-dynamic version. In the case of the full-dynamic version, it has to search the root tree to find the node that corresponds to grey level i. This node contains the pointer to the root of tree T_i (for each non empty BB-tree in the full-dynamic version, there is a node in the root tree which stores its corresponding grey level as well as a pointer to its root). After the algorithm reaches the root, it searches for grey level j in BB-tree T_i, in the way presented in Section 5.2.

No single version of the co-occurrence trees is more appropriate for all cases of the analysed images. Both of them have advantages and disadvantages, as will be shown in the results of Chapter 9. Actually, there are trade-offs in using one or the other version in texture analysis. These trade-offs are presented in detail in Chapter 10.

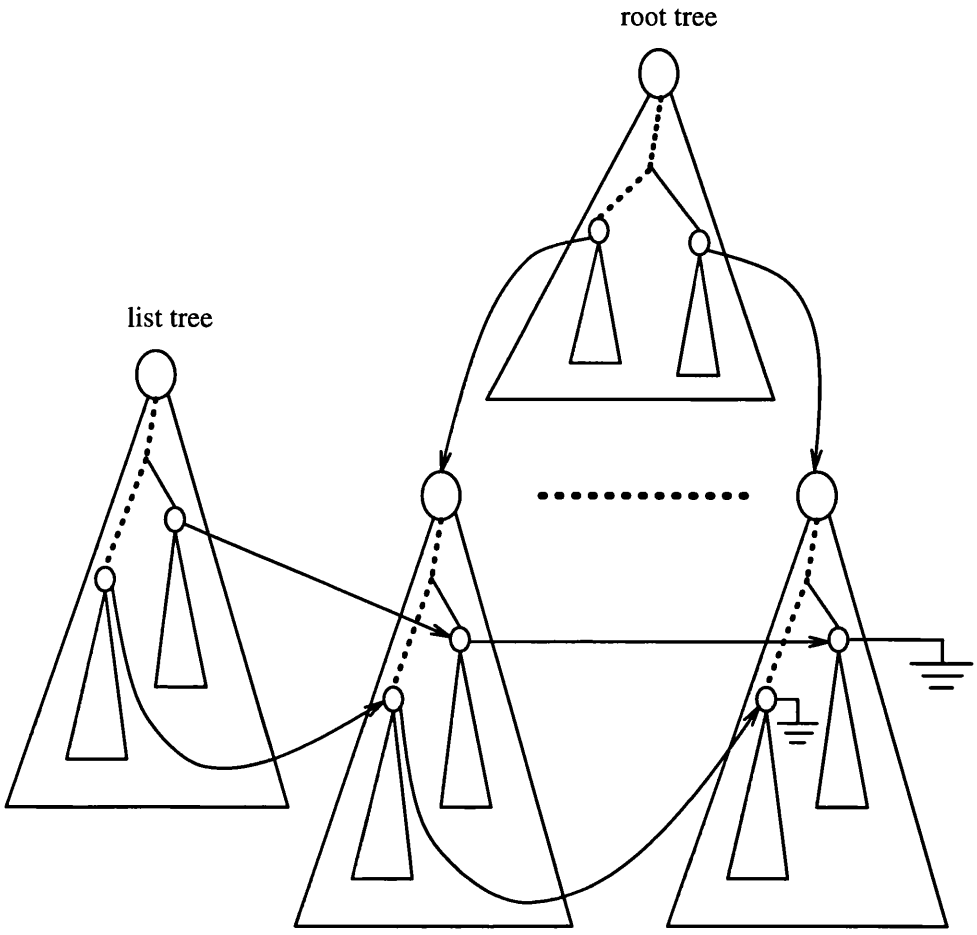


Figure **6**.6: Structure of the full-dynamic version.

In general, semi-dynamic version allows faster access to the co-occurrence trees, since the time needed to access the roots of the trees and the heads of the lists is constant ($O(1)$). This is a consequence of the use of static arrays to store the relevant pointers. The full-dynamic version uses BB-trees to store these pointers and so, logarithmic time is needed to access them. On the other hand, using static arrays for storing the pointers increases the memory space requirements, because the space allocated to them is proportional to the number of grey levels in the entire image. A large number of null pointers are expected to be stored in these arrays, especially for images with a large dynamic range. In our previous example (see Figures **6**.1 and **6**.2), a null pointer needs to be stored for BB-tree T_6, even though this tree does not exist. In the full-dynamic version, null pointers are not stored. Therefore, no redundant information is contained in the structures that store the pointers to the roots of the BB-trees and the heads

of the lists, in this case. These reduced memory requirements are crucial for analysing images with a large dynamic range where, as we will see in Section **9**.4, the system often runs out of memory using the semi-dynamic version.

6.2 Complexity analysis

We first present some lemmas about the memory space complexity of the plain co-occurrence trees and the time complexity of the elementary operations used in the computation of the 13 texture features proposed in [65]. Then, we present a theorem which gives the least improvement (i.e. in the worst case) in time and space of the proposed approach compared with the co-occurrence matrix. As we said in Section **5**.1, there are two general ways to estimate the space and time complexity of an algorithm, i.e. worst case analysis and average case analysis. Here, a worst case analysis is performed which is the most frequent type of analysis, allowing the derivation of lower bounds in the efficiency of the algorithm. Let N_g be the number of grey levels (grey level range) in an image and n_g the number of grey levels (local grey level range) in the analysed region.

Lemma 6.2.1 *The space complexity of the proposed approach is* $O(n_g^2 + N_g)$ *for the semi-dynamic version and* $O(n_g^2)$ *for the full-dynamic version, in the worst case.*

Proof:

Suppose that the forest of the co-occurrence trees is composed of $T_1, T_2, \ldots, T_n$ trees, where tree T_i $(1 \leq i \leq n)$ is the BB-tree that corresponds to some grey level k, $0 \leq k \leq n_g - 1$. That is, in the above notation i is an index in the tree sequence and not the actual grey level that corresponds to tree T_i, which is grey level k. It is clear from our discussion in the previous section that $1 \leq n \leq n_g$. If $n = 1$, then the first grey level of the pixel pair is the same for all pairs in the analysed region that satisfy the given spatial relationship (d, ϑ) (there is only one tree in the forest). In the worst case, there is a spatial relationship where all n_g grey levels appear as first members of the pixel pairs. In this case there are n_g BB-trees.

Let $N(T_i)$ be a function that gives the number of nodes in tree T_i. This number is actually the number of distinct grey levels, which are second members of the pixel pairs in the analysed region having as first member the grey level that corresponds to tree T_i and satisfying the given spatial relationship. Therefore, it holds that $0 \leq N(T_i) \leq n_g$. If $N(T_i) = 0$ then for the given spatial relationship, T_i is the null tree. In the worst case, there is a spatial relationship for which $N(T_i) = n_g$ (tree T_i can have up to n_g nodes). So, the total number of nodes of the co-occurrence trees can be as large as $\sum_{i=1}^{n_g} n_g = n_g^2$. Since the space complexity for each node is $O(1)$ (constant), we get that the space complexity for the forest of trees in the proposed approach is $O(n_g^2)$.

In the semi-dynamic version, the structure for accessing the roots of the BB-trees and the structure for accessing the heads of the lists need $O(N_g)$ space, since their memory requirements depend on the number of grey levels in the entire image, as we said earlier in this chapter. In the full-dynamic version, the above structures are BB-trees and they need $O(n_g)$ space in the worst case, since there are at most n_g trees (at most n_g first members in the pixel pairs) and at most n_g lists (at most n_g second members in the pixel pairs).

We mentioned in the previous section that the elementary operation $p_x(i)$ needs a stack for its implementation. The size of the stack can be as large as the longest path in a BB-tree. This is clear from the algorithm for the symmetrical traversal presented earlier. Since the depth of a BB-tree T_i is $O(\log N(T_i))$ (from now on T_i corresponds to grey level i) and $N(T_i) = n_g$ in the worst case, we get $O(\log n_g)$ as the space complexity for the stack.

The total space for the proposed approach is

$$O(n_g^2) + O(N_g) + O(\log n_g) = O(n_g^2 + N_g + \log n_g) = O(n_g^2 + N_g) \qquad (\mathbf{6}.1)$$

for the semi-dynamic version and

$$O(n_g^2) + O(n_g) + O(\log n_g) = O(n_g^2 + n_g + \log n_g) = O(n_g^2) \qquad (\mathbf{6}.2)$$

for the full-dynamic version, in the worst case. □

Lemma 6.2.2 *The computational time complexity of the $p_x(i)$ elementary operation is $O(n_g)$, in the worst case.*

Proof:

As we have already mentioned, in order to compute $p_x(i)$ a symmetrical traversal of BB-tree T_i needs to be performed. Since $N(T_i) = n_g$ (number of grey levels in the analysed region) in the worst case, n_g nodes need to be visited in T_i. Each visit at a node costs $O(1)$ time. This time includes the push and pop operations in the stack (for each node we have to perform one push and one pop operation), the access of the counter stored in the node and its contribution to the computation of $p_x(i)$, and finally, the access of the children of the node through the pointers stored in it. Thus, the total time for traversing tree T_i is $n_g \cdot O(1) = O(n_g)$, in the worst case.

In the semi-dynamic version, finding the root of tree T_i costs $O(1)$ time, since the roots of the trees are stored in an array. Therefore, the total time in this case is:

$$O(n_g) + O(1) = O(n_g) \tag{6.3}$$

In the full-dynamic version, we need to access the root tree in order to get the root of tree T_i. Since an access operation in the BB-tree T_i costs $O(\log N(T_i))$, in the worst case the time is $O(\log n_g)$. So, the total time in this case is:

$$O(n_g) + O(\log n_g) = O(n_g + \log n_g) = O(n_g) \tag{6.4}$$

Therefore, in both versions of the plain co-occurrence trees the total time is $O(n_g)$ for the $p_x(i)$ operation, in the worst case. □

Lemma 6.2.3 *The computational time complexity of the $p_y(j)$ elementary operation is $O(n_g)$, in the worst case.*

Proof:

For the implementation of the $p_y(j)$ operation the algorithm needs to access all nodes in list L_j. This list contains all nodes in the co-occurrence trees which correspond to pixel pairs whose second member is grey level j (see the previous section).

If $N(L_j)$ denotes the length of the list, it is clear that $0 \leq N(L_j) \leq n_g$. If $N(L_j) = 0$ then there is no pixel pair in the analysed region for which the second member is grey level j, for the given spatial relationship. In the worst case, list L_j passes through n_g trees in the forest, that is, there are n_g distinct pixel pairs having grey level j as their second member in the analysed region. The cost for accessing a node in list L_j is $O(1)$ (constant). It includes the access of the counter of the node and its contribution to the computation of $p_y(j)$ and the access of the next node in the list through the pointer stored in it. Thus, the total time for traversing list L_j is $O(1) \cdot n_g = O(n_g)$, in the worst case.

In the semi-dynamic version, an entry in the array that stores the heads of the lists needs to be accessed. This costs $O(1)$ time. Therefore, the total time in this case is:

$$O(n_g) + O(1) = O(n_g) \tag{6.5}$$

In the full-dynamic version, the list tree needs to be accessed in order to find the head of list L_j. Since the list tree is a BB-tree the cost for this operation is $O(\log n_g)$, since there are n_g lists and thus n_g heads of lists in the worst case. The total time is:

$$O(n_g) + O(\log n_g) = O(n_g + \log n_g) = O(n_g) \tag{6.6}$$

Therefore, in the worst case $p_y(j)$ elementary operation needs $O(n_g)$ time for both versions of the plain co-occurrence trees. □

Lemma 6.2.4 *The computational time complexity of the $p_{x+y}(k)$ elementary operation is $O((k+1)\log n_g)$ if $0 \leq k \leq n_g - 2$ and $O((2n_g - k - 1)\log n_g)$ if $n_g - 1 \leq k \leq 2(n_g - 1)$, in the worst case.*

Proof:

We have seen in Section 4.1 that $p_{x+y}(k)$ elementary operation is defined by:

$$P_{x+y}(k) = \sum_{\substack{i=0 \\ i+j=k}}^{n_g-1} \sum_{j=0}^{n_g-1} P(i,j)\,, \quad k = 0, 1, \ldots, 2(n_g - 1) \tag{6.7}$$

This can be rewritten as:

$$P_{x+y}(k) = \sum_{\substack{i=0 \\ k-n_g+1 \leq i \leq k}}^{n_g-1} P(i, k-i) \tag{6.8}$$

From the above formula, it is clear that the algorithm needs to access grey level $k-i$ in tree T_i, for all i where $0 \leq i \leq n_g - 1$ and $k - n_g + 1 \leq i \leq k$. As we have already seen in the previous proofs, this access costs $O(\log n_g)$ in the worst case. Also, the access of the counter of the corresponding node and its subsequent contribution to the computation of $p_{x+y}(k)$ costs $O(1)$ time. Therefore, the total access time for each tree T_i is $O(1) + O(\log n_g) = O(\log n_g)$, in the worst case.
There are two cases for parameter k:

- if $0 \leq k \leq n_g - 2$, then:

$$\left.\begin{array}{c} 0 \leq i \leq n_g - 1 \\ k - n_g + 1 \leq i \leq k \end{array}\right\} \Rightarrow 0 \leq i \leq k \qquad (6.9)$$

  In this case $p_{x+y}(k) = \sum_{i=0}^{k} p(i, k-i)$ and therefore, $k+1$ trees T_i need to be accessed. As we have seen in proof of Lemma **6**.2.2, in the semi-dynamic version, finding the root of tree T_i costs $O(1)$ time. In the full-dynamic version, the above time is $O(\log n_g)$, in the worst case. Therefore, the total worst time is

$$(k+1) \cdot \{O(\log n_g) + O(1)\} = O((k+1) \cdot \log n_g) \qquad (6.10)$$

  for the semi-dynamic version and

$$(k+1) \cdot \{O(\log n_g) + O(\log n_g)\} = O((k+1) \cdot \log n_g) \qquad (6.11)$$

  for the full-dynamic version.

- if $n_g - 1 \leq k \leq 2(n_g - 1)$, then:

$$\left.\begin{array}{c} 0 \leq i \leq n_g - 1 \\ k - n_g + 1 \leq i \leq k \end{array}\right\} \Rightarrow k - n_g + 1 \leq i \leq n_g - 1 \qquad (6.12)$$

  In this case $p_{x+y}(k) = \sum_{i=k-n_g+1}^{n_g-1} p(i, k-i)$. The number of trees T_i that need to be accessed in this case is $n_g - 1 - k + n_g - 1 + 1 = 2n_g - k - 1$.

Similarly to the previous case, the total time in the worst case is

$$(2n_g - k - 1) \cdot \{O(\log n_g) + O(1)\} = O((2n_g - k - 1) \cdot \log n_g) \quad (6.13)$$

for the semi-dynamic version and

$$(2n_g - k - 1) \cdot \{O(\log n_g) + O(\log n_g)\} = O((2n_g - k - 1) \cdot \log n_g) \quad (6.14)$$

for the full-dynamic version.

□

Lemma 6.2.5 *The computational time complexity of the $p_{x-y}(k)$ elementary operation is $O((n_g - k)\log n_g)$, in the worst case.*

Proof:

The $p_{x-y}(k)$ elementary operation is defined by:

$$P_{x-y}(k) = \sum_{\substack{i=0 \\ |i-j|=k}}^{n_g-1} \sum_{j=0}^{n_g-1} P(i,j)\,, \quad k = 0, 1, \ldots, n_g - 1 \quad (6.15)$$

This formula can be rewritten as:

$$p_{x-y}(k) = \begin{cases} \displaystyle\sum_{\substack{i=0 \\ k \le i \le n_g+k-1}}^{n_g-1} P(i, i-k), & j < i \\ \\ \displaystyle\sum_{\substack{i=0 \\ -k \le i \le n_g-k-1}}^{n_g-1} P(i, i+k), & j \ge i \end{cases} \quad (6.16)$$

Therefore, we need to search for grey level $i - k$ if it exists (i.e. if $i - k \geq 0$) and also for grey level $i + k$ if it exists (i.e. if $i + k \leq n_g - 1$) in tree T_i. So, we have two cases:

- if $j < i$, then:

$$\left.\begin{array}{c} 0 \le i \le n_g - 1 \\ k \le i \le n_g + k - 1 \\ 0 \le k \le n_g - 1 \end{array}\right\} \Rightarrow \quad k \le i \le n_g - 1 \qquad (\mathbf{6}.17)$$

- if $j \ge i$, then:

$$\left.\begin{array}{c} 0 \le i \le n_g - 1 \\ -k \le i \le n_g - k - 1 \\ 0 \le k \le n_g - 1 \end{array}\right\} \Rightarrow \quad 0 \le i \le n_g - k - 1 \qquad (\mathbf{6}.18)$$

From the above relations, it is obvious that there are three case for i. If for a specific i relation (**6**.17) holds but relation (**6**.18) does not, we need to search only for grey level $i - k$ in tree T_i. If relation (**6**.17) does not hold but relation (**6**.18) holds, then we need to search only for grey level $i + k$ in T_i. If, on the other hand, both relation (**6**.17) and (**6**.18) hold for a specific i, we need to search for both grey level $i - k$ and grey level $i + k$ in tree T_i. There are $n_g - 1 - k + 1 = n_g - k$ trees T_i in which we need to search for grey level $i - k$. Similarly, there are $n_g - k - 1 + 1 = n_g - k$ trees T_i in which we need to search for grey level $i + k$. In the worst case, we need to search $2(n_g - k)$ trees T_i.

We saw in the previous proofs that each access to BB-tree T_i costs $O(\log n_g)$, in the worst case. We also know that finding the root of tree T_i costs $O(1)$ time in the semi-dynamic version and $O(\log n_g)$ worst time in the full-dynamic version. Therefore, the total worst time for the semi-dynamic version is:

$$2 \cdot (n_g - k)\,\{O(\log n_g) + O(1)\} = O(2(n_g - k)\log n_g) = O((n_g - k)\log n_g) \quad (\mathbf{6}.19)$$

The corresponding time for the full-dynamic version is:

$$2(n_g - k)\,\{O(\log n_g) + O(\log n_g)\} = O(4(n_g - k)\log n_g) = O((n_g - k)\log n_g) \qquad (\mathbf{6}.20)$$

□

Theorem 6.2.1 *The reduction in space complexity using the plain co-occurrence trees instead of the co-occurrence matrix, is at least* $\Omega(\frac{N_g^2}{n_g^2+N_g})$ *times for the semi-dynamic version and at least* $\Omega((\frac{N_g}{n_g})^2)$ *times for the full-dynamic version. The reduction in time complexity for the elementary operations* $p_x(i)$ *and* $p_y(j)$ *using the plain co-occurrence trees instead of the co-occurrence matrix, is at least* $\Omega(\frac{N_g}{n_g})$ *times. The corresponding reduction for the texture features using the elementary operations* $p_{x+y}(k)$, $p_{x-y}(k)$, *is at least* $\Omega(\frac{N_g^2}{n_g^2 \log n_g})$ *times.*

Proof:

According to Lemma **6**.2.1, the space complexity of the plain co-occurrence trees is $O(n_g^2+N_g)$ for the semi-dynamic version and $O(n_g^2)$ for the full-dynamic version, in the worst case. The co-occurrence matrix needs $O(N_g^2)$ space in all cases since, as we have already said, its size depends on the number of grey levels in the entire image and each entry of the matrix needs $O(1)$ (constant) space. Therefore, the reduction is at least $\Omega(\frac{N_g^2}{n_g^2+N_g})$ times for the semi-dynamic version and at least $\Omega((\frac{N_g}{n_g})^2)$ times for the full dynamic version.

According to Lemmas **6**.2.2 and **6**.2.3, the time complexity for the $p_x(i)$ and $p_y(j)$ elementary operations is $O(n_g)$, in the worst case. In the co-occurrence matrix approach, in order to compute $p_x(i)$ and $p_y(j)$, we need to access the entries of row i and column j, respectively. Obviously, this access takes $O(N_g)$ time in all cases, since the access time for each entry of the matrix is $O(1)$ (constant) and the size of the matrix is N_g. Therefore, a speedup of at least $\Omega(\frac{N_g}{n_g})$ is achieved using the proposed approach instead of the co-occurrence matrix.

According to Lemma **6**.2.4, the time complexity for the $p_{x+y}(k)$ elementary operation is $O((k+1)\log n_g)$, if $0 \leq k \leq n_g - 2$ and $O((2n_g - k - 1)\log n_g)$, if $n_g - 1 \leq k \leq 2(n_g - 1)$, in the worst case. This elementary operation has to be computed for all k, where $0 \leq k \leq 2(n_g - 1)$, for the texture features in which it is used. Therefore, the total time needed for the computation of these texture features using the plain co-occurrence trees is:

$$\sum_{k=0}^{n_g-2} O((k+1)\log n_g) + \sum_{k=n_g-1}^{2(n_g-1)} O((2n_g-k-1)\log n_g) =$$

$$O(\log n_g \cdot \left\{ \sum_{k=0}^{n_g-2} (k+1) + \sum_{k=n_g-1}^{2(n_g-1)} (2n_g-k-1) \right\}) =$$

$$O(n_g^2 \log n_g) \qquad (6.21)$$

In the co-occurrence matrix approach it holds that $0 \leq k \leq 2(N_g-1)$. Using similar arguments as in the proof of Lemma **6**.2.4, we can prove that the cost of the $p_{x+y}(k)$ elementary operation in this approach is $(k+1)\cdot O(1)$, if $0 \leq k \leq N_g-2$ and $(2N_g-k-1)\cdot O(1)$, if $N_g-1 \leq k \leq 2(N_g-1)$, in all cases. In the co-occurrence matrix, the logarithmic factor $O(\log n_g)$ is replaced by the constant factor $O(1)$, since the access to an entry of the co-occurrence matrix takes $O(1)$ time. So, in this approach the total time for the texture features that need to compute $p_{x+y}(k)$ is:

$$\sum_{k=0}^{N_g-2} (k+1)\cdot O(1) + \sum_{k=N_g-1}^{2(N_g-1)} (2N_g-k-1)\cdot O(1) =$$

$$O(1)\cdot \left\{ \sum_{k=0}^{N_g-2} (k+1) + \sum_{k=N_g-1}^{2(N_g-1)} (2N_g-k-1) \right\} =$$

$$O(N_g^2) \qquad (6.22)$$

in all cases. Therefore, the speed-up is at least $\Omega(\frac{N_g^2}{n_g^2 \log n_g})$.

According to Lemma **6**.2.5, the time complexity for the $p_{x-y}(k)$ elementary operation is $O((n_g-k)\log n_g)$, in the worst case. For all texture features in which it is used, this operation needs to be computed for all k, $0 \leq k \leq n_g-1$. Therefore, the total time for these texture features using the plain co-occurrence trees approach is:

$$\sum_{k=0}^{n_g-1} O((n_g-k)\log n_g) = O\left\{ \log n_g \cdot \left(\sum_{k=0}^{n_g-1} (n_g-k) \right) \right\} =$$

$$O(n_g^2 \log n_g) \qquad (6.23)$$

Similarly as above, the co-occurrence matrix needs $(N_g - k) \cdot O(1)$ time for the $p_{x-y}(k)$, where $0 \leq k \leq N_g - 1$. Therefore, in this approach the time for the texture features that use this elementary operation is:

$$\sum_{k=0}^{N_g-1} (N_g - k) \cdot O(1) = O(\sum_{k=0}^{N_g-1} (N_g - k)) =$$

$$O(N_g^2) \tag{6.24}$$

in all cases. Thus, the speed-up is at least $\Omega(\frac{N_g^2}{n_g^2 \log n_g})$ in this case, as well. □

The theoretical results in the above theorem indicate that the reduction in space complexity achieved by the plain co-occurrence trees is significant. Full-dynamic version is the most efficient approach in memory space, since it does not depend on N_g at all. Therefore, it seems to be the most appropriate for applications where memory requirements are large, e.g. analysis of images with a dynamic range higher than 8 bits.

Texture features, such as the correlation, need to compute $p_x(i)$ for all grey levels i and(or) $p_y(j)$ for all grey levels j. In these cases, following arguments similar to the proof of the previous theorem, we can show that the speed-up is at least $\Omega(\frac{N_g^2}{n_g^2})$. Thus, a significant reduction in time complexity is achieved, as well. In the cases of the $p_{x+y}(k)$ and $p_{x-y}(k)$ elementary operations, the speed-up achieved by using the proposed approach may not be significant, except for the case where $n_g \ll N_g$. However, this case is not unusual, especially when high dynamic range images are analysed.

All the theoretical results presented in the above theorem, are lower bounds on the performance of the proposed approach. In the average case, the performance of the plain co-occurrence trees in space and computational time is expected to be much better. This will become more clear in Chapter **9**, where we present results from real data analysis. In contrast, the co-occurrence matrix approach shows the same behaviour in all cases. That is, its space and time complexity always depend on N_g (the number of grey levels in the entire image).

Chapter 7

Enhanced co-occurrence trees

7.1 Description

We mentioned in Section 5.4 that by taking advantage of the fact that the probability distribution of the occurrence of grey level pairs in a given real texture is not uniform, we may be able to improve the time performance of the co-occurrence trees. The key idea is that if the grey level pairs with the largest access probabilities, which are identical to their co-occurrence probabilities, are stored in lists of small length (much smaller than the depth of the BB-trees), the time cost of subsequent access operations can be reduced. This is because in this way, the frequently accessed grey level pairs can be directly found within the small lists, thereby avoiding searching the BB-trees which might be much costlier. To implement this idea, each BB-tree T_i of the plain co-occurrence trees (i is its corresponding grey level) is enhanced by the addition of a list P_i, which is called the **probability list** for grey level i. The addition of the probability lists to the plain co-occurrence trees leads to the second dynamic approach proposed in this thesis, which is called the **enhanced co-occurrence trees** ([152]).

In reality, the elements of the probability list P_i correspond to those grey level pairs (i, j) which are the most likely to be accessed in the near future according to a rule. This rule selects the appropriate pairs and updates the list structure accordingly. Actually, the element of the list P_i that corresponds to the pair (i, j), contains a pointer to a node in tree T_i which corresponds to the

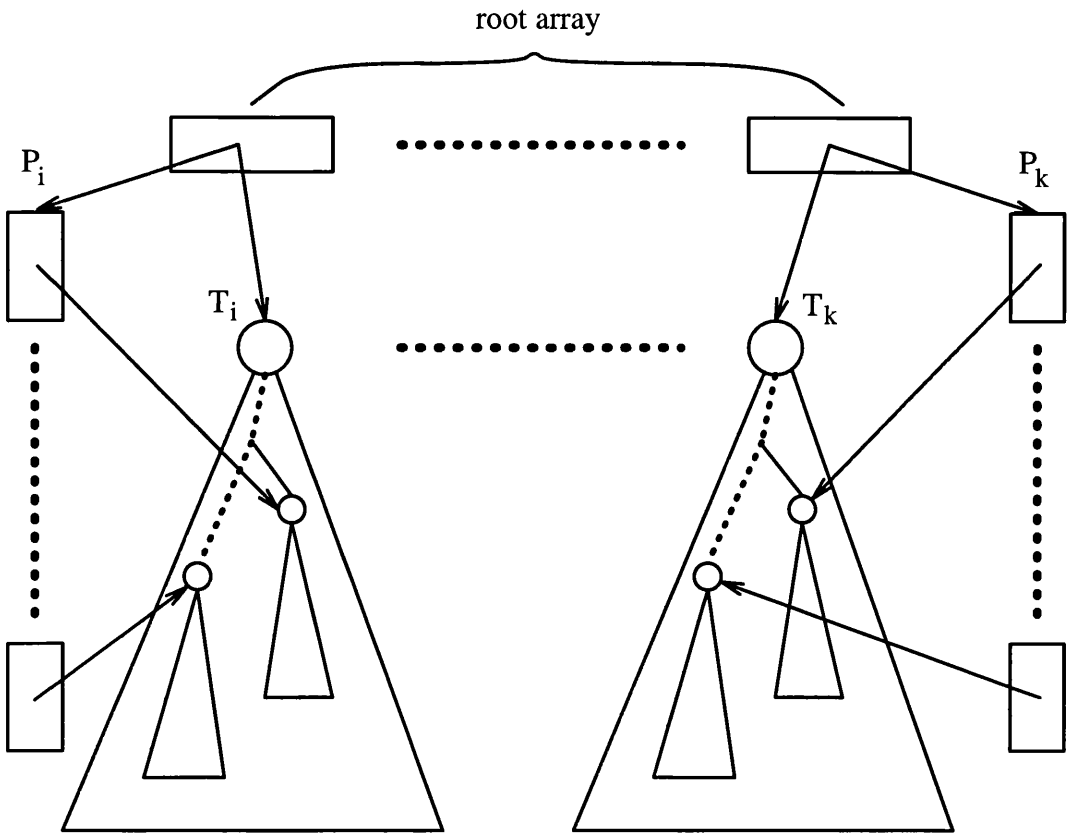


Figure **7**.1: Semi-dynamic version with probability lists.

same grey level pair. It also contains the grey level j. Figures **7**.1 and **7**.2 show the two versions of the enhanced co-occurrence trees, namely the **semi-dynamic** version and the **full-dynamic** version, along with the corresponding probability lists (recall that this approach is actually the plain co-occurrence trees with the addition of a number of lists). The set of lists L_j (see Section **6**.1) is not shown in the above figures for clarity.

The update rule takes action whenever it is necessary to change the elements of list P_i, so that this list contains pointers to the nodes in tree T_i that correspond to the most frequently accessed grey level pairs (i, j). More specifically, after each access in one of the BB-trees, the content and(or) the structure of its probability list is updated. The update rule changes the content of a list by assigning new values to its elements. It can also change its structure by relocating the elements inside the list (permutation). In order to clarify all the above, we need to present how the access operation is performed in the enhanced co-occurrence trees.

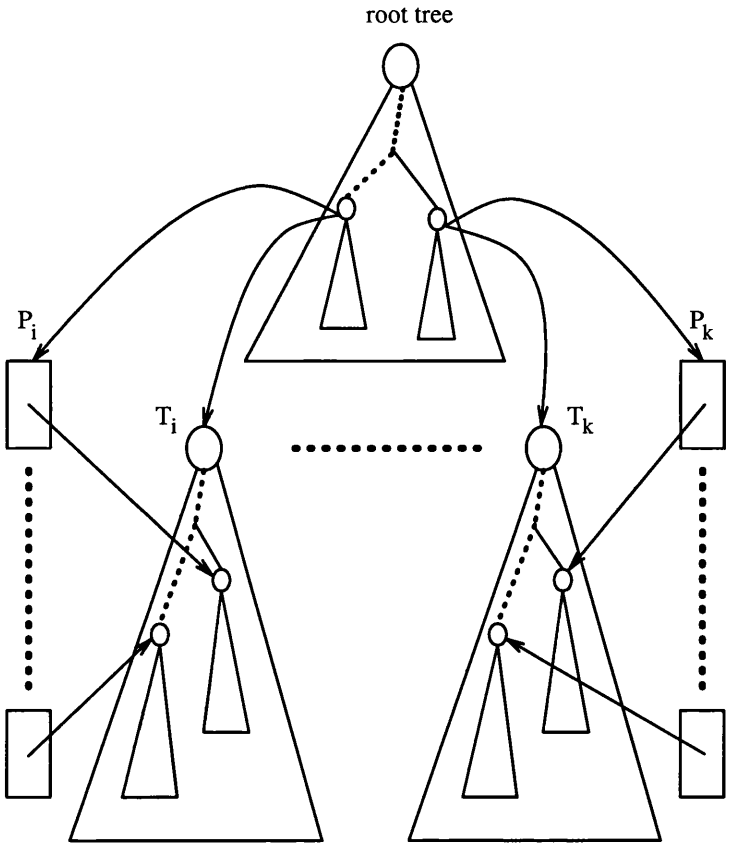


Figure 7.2: Full-dynamic version with probability lists.

Suppose that the grey level pair (i, j) needs to be accessed. First, the algorithm searches in the probability list P_i for the grey level j. The pointer to the head of list P_i is found in the element of the root array that corresponds to grey level i, in the case of the semi-dynamic version. In the case of the full-dynamic version, the pointer is stored in the node of the root tree that corresponds to grey level i. Therefore, in this case the algorithm first needs to search the root tree for grey level i.

The search in the probability list P_i for grey level j is linear, that is, the algorithm has to compare it with every element of the list starting from the head, until it finds it there or it exhausts all elements in this list. If the access operation finds the requested grey level j in the list, it simply follows the relevant pointer to the node of the tree T_i that corresponds to grey level pair (i, j). Otherwise, it searches for the grey level j in the BB-tree T_i, in the usual way (see Section **5**.2) and then, it updates the list P_i accordingly. The pseudocode for the access operation for grey level pair (i, j) is the following:

```
Access(i, j):   v ← head(i);
                WHILE (val(v) ≠ j AND v ≠ tail(i)) DO
                    v ← next(v);
                ENDWHILE
                IF (val(v) ≠ j)
                    access(T_i, j);
                    update(i);
                ELSE
                    RETURN (TRUE, pointer(v));
                ENDIF
```

The auxiliary function $head(i)$ returns back the head (the first node) of the list P_i while function $tail(i)$ returns the tail (the last node) of P_i. Here, the auxiliary function $val(v)$ returns the corresponding grey level of node v. Also, the auxiliary function $next(v)$ gives the next node in the probability list P_i. In addition, the function $access(T_i, j)$ searches for grey level j in BB-tree T_i and the subroutine $update(i)$ updates the list P_i. Finally, the auxiliary function $pointer(v)$ returns a pointer to the relevant node in BB-tree T_i (the node that contains grey level j). This pointer is stored in node v (element v) of list P_i.

From the above discussion, it is clear that the main problem is the determination of the K grey levels j that are put in probability list P_i in each update. Parameter K is actually the length of the probability lists (all probability lists have the same length) and as we will show in the next section, is one of the main factors that affect the time performance of the enhanced co-occurrence trees. Three update rules are investigated in this thesis (see also [152]):

a) the static rule, where the K grey levels are determined *a-priori*, according to a given co-occurrence distribution,

b) the move to front rule, where the grey level j that has been most recently accessed in tree T_i and its $K-1$ closest neighbours in T_i comprise the new P_i,

c) the counter rule, which is similar to the move to front rule, but now the grey levels in the new P_i are sorted in descending order, according to their estimated co-occurrence frequencies.

7.1.1 Static rule

In the static rule, the K grey levels with the largest co-occurrence probabilities are determined *a-priori*, using the Gaussian distribution texture model described in [27]. According to this model, one texture can be considered as a homogeneous (i.e. stationary) random field, described by a second-order Gaussian probability density function. Due to the homogeneity (stationarity) of the random field, it holds that

$$f(g_1, g_2; i_1, i_2) = Prob\{I(i_1) = g_1, I(i_2) = g_2\} = f(g_1, g_2; d, \vartheta) \equiv f(g_1, g_2) \tag{7.1}$$

where d is the interpixel distance and ϑ is the orientation of the grey level pair (g_1, g_2) in the analysed texture (see Section 4.1). Now, according to the Gaussian distribution the above probability density function is given by the relation:

$$f(g_1, g_2) = \frac{1}{2\pi\sqrt{|\mathbf{V}|}} \cdot e^{-\frac{1}{2}\cdot\{([g_1\ g_2]-\mathbf{m})\cdot\mathbf{V}^{-1}\cdot([g_1\ g_2]-\mathbf{m})^T\}} \tag{7.2}$$

where $\mathbf{m}$ is the 1×2 mean vector and $\mathbf{V}$ is the 2×2 covariance matrix of the random variables g_1, g_2.

Parameters $\mathbf{m}$ and $\mathbf{V}$ completely specify the Gaussian probability density function. If $\overline{d} = (d_x, d_y)$ is the corresponding displacement vector (spatial relationship equivalent to (d, ϑ) form; see Section 4.1), then the mean vector $\mathbf{m}$ can be estimated by the following equation:

$$\hat{\mathbf{m}} = [\hat{m}_1\ \hat{m}_2] = \frac{1}{(n_x - d_x)\cdot(n_y - d_y)} \cdot \sum_{i=0}^{n_x-d_x-1} \sum_{j=0}^{n_y-d_y-1} [I(i,j)\ I(i+d_x, j+d_y)] \tag{7.3}$$

where n_x is the x dimension (number of rows) of the analysed texture, n_y is the y dimension (number of columns) of the analysed texture, and $I(i,j)$ is the picture brightness at pixel location (i,j).

In order to estimate the covariance matrix $\mathbf{V}$, we first compute the following quantities:

$$\left[\hat{\sigma_1^2}\ \hat{\sigma_2^2}\right] = \frac{1}{(n_x - d_x) \cdot (n_y - d_y)} \cdot$$

$$\sum_{i=0}^{n_x - d_x - 1} \sum_{j=0}^{n_y - d_y - 1} \left[(I(i,j) - \hat{m_1})^2\ (I(i + d_x, j + d_y) - \hat{m_2})^2 \right] \qquad (7.4)$$

and

$$\hat{\sigma_3} = \frac{1}{(n_x - d_x) \cdot (n_y - d_y)} \cdot$$

$$\sum_{i=0}^{n_x - d_x - 1} \sum_{j=0}^{n_y - d_y - 1} (I(i,j) - \hat{m_1}) \cdot (I(i + d_x, j + d_y) - \hat{m_2}) \qquad (7.5)$$

Then, the estimation $\hat{\mathbf{V}}$ of the covariance matrix $\mathbf{V}$ is defined as:

$$\hat{\mathbf{V}} = \begin{bmatrix} \hat{\sigma_1^2} & \hat{\sigma_3} \\ \hat{\sigma_3} & \hat{\sigma_2^2} \end{bmatrix} \qquad (7.6)$$

It holds that $E\{\hat{\mathbf{m}}\} = \mathbf{m}$ and $E\{\hat{\mathbf{V}}\} = \mathbf{V}$, where $E\{\cdot\}$ is the expectation operator (see [27]). Therefore, $\hat{\mathbf{m}}$ and $\hat{\mathbf{V}}$ are unbiased estimators of the corresponding quantities.

After the parameters $\mathbf{m}$ and $\mathbf{V}$ are estimated, the co-occurrence probabilities of the grey level pairs can in turn be estimated from the Gaussian model defined by (7.2). The grey levels j with the K largest co-occurrence probabilities are computed for each grey level i, $0 \leq i < N_g$, where N_g is the number of grey levels in the analysed image. Then, these grey levels j are stored in the probability list P_i, in decreasing order, according to their estimated co-occurrence probabilities. The content and structure as well as the number N_g of the probability lists remain unchanged throughout the analysis of the given image, for the specific spatial relationship (d, ϑ). These hold for both versions of the enhanced co-occurrence trees.

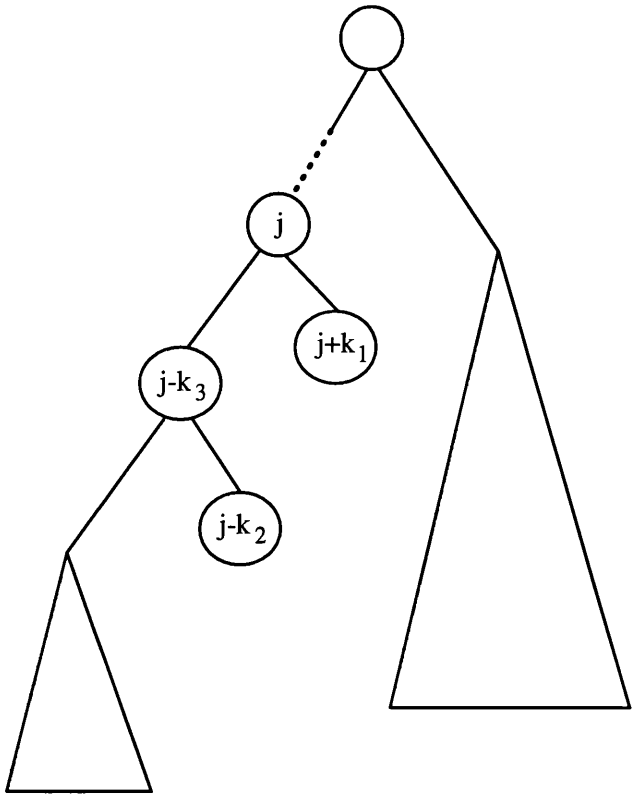


Figure 7.3: An example of a BB-tree.

7.1.2 Move to front rule

In this case, we follow a variation of the move to front rule that was presented in Section 5.4. When grey level j is accessed in BB-tree T_i, grey level j and its $K-1$ closest neighbours in T_i comprise the new content of probability list P_i (K is the probability list length). The goal of this rule is to keep the most frequently accessed grey levels in the analysed textural region within the probability lists, so that the texture analysis algorithm can rapidly access them.

In contrast with the static rule, the move to front rule is dynamic, modifying the number and contents of the probability lists, so that they can follow the co-occurrence distribution in the analysed local region. No *a-priori* assumption about this distribution needs to be made. We will show how this rule works with the following example.

Suppose that $K = 4$ and the algorithm needs to access grey level j in BB-tree T_i (T_i is shown in Figure 7.3). According to the property of the Balanced Binary tree, it holds that $j - k_3 < j - k_2 < j < j + k_1$, where k_1, k_2, $k_3 > 0$

(recall that in this thesis we use the node-oriented version of the BB-tree; see also Section 5.2). After the algorithm accesses grey level j in T_i, it changes the content of the probability list P_i so that this list now contains pointers to the grey levels $j - k_3$, $j - k_2$, j, and $j + k_1$ in T_i.

7.1.3 Counter rule

The counter rule is similar to the move to front rule. The only difference is that the grey levels in the probability lists are sorted in descending order, according to their access frequencies. Normally, extra storage is needed for these counters which might be significant. However, in our case the nodes of the BB-trees already contain these access frequencies, which are actually partial estimates of the co-occurrence probabilities used for the computation of the texture features. Therefore, no additional memory space is needed for the implementation of the counter rule.

Suppose that the function $count(j)$ gives the access frequency associated with grey level j in BB-tree T_i (i.e. how many times the grey level pair (i, j) has been encountered in the analysed region so far, satisfying the given spatial relationship). Without loss of generality, we assume that

$$count(j) > count(j + k_1) > count(j - k_2) > count(j - k_3) \qquad (7.7)$$

in our previous example. Then, probability list P_i will contain the above grey levels in this order, i.e. j, $j + k_1$, $j - k_2$, $j - k_3$.

It is clear that the grey levels have to be sorted first, before they are put in the probability lists. Although, a fast algorithm such as quicksort (see [86]), can be used to perform the sorting, the necessary computational time is not negligible. Therefore, the efficiency of this rule has to be investigated through the analysis of real data.

7.2 Complexity analysis

Two main factors affect the time complexity of the enhanced co-occurrence trees. These are the length of the probability lists (parameter K) and clearly, the rule used to update the content and(or) structure of the lists. In particular, from our discussion in the previous section it appears that the length of the probability lists should be kept as small as possible. This is due to the linear search that has to be performed when the algorithm accesses a list. Unfortunately, we cannot apply any other, more sophisticated search method, such as binary search or interpolation search (see [86], [110]), because the elements of the lists are not ordered according to their grey levels. Therefore, the time complexity of this search is linear on the length of the list K.

On the other hand, the probability that a grey level will be found in a list and thus, a search in the relevant BB-tree won't be necessary, depends not only on the update rule being applied but also on the length K. The larger the length K of the lists, the more probable is to find the requested grey level in these lists. We can show the relationship between these factors and the time performance of the proposed approach through the following worst case analysis.

Suppose that the update rule chooses the grey levels that will be put in the probability list P_i according to a fixed but unknown distribution. If the length of the lists P_i is K, then the update rule chooses K grey levels during each update of P_i. Suppose that the probability that it chooses grey level j in one trial is p and that this grey level has the smallest probability of being selected among all candidate grey levels. Then, the probability that it chooses grey level j in the lth trial, where $1 \leq l \leq K$, follows the geometric distribution, i.e. this probability is $p(1-p)^{l-1}$ (we assume that length K is smaller than the number of grey levels in the analysed image; otherwise the update rule can choose all grey levels and $p = 1$). Now, the probability that grey level j will be put in list P_i during an update is given by:

$$\sum_{l=1}^{K} p \cdot (1-p)^{l-1} = p \cdot \sum_{l=1}^{K} (1-p)^{l-1} = p \cdot \sum_{l=0}^{K-1} (1-p)^{l} =$$

$$p \cdot \frac{(1-p)^K - 1}{-p} = 1 - (1-p)^K \qquad (7.8)$$

We now assume that the cost for visiting a node in a probability list is the same as the cost for visiting a node in a BB-tree (let this cost be c, where $c > 0$). In the worst case, the next grey level pair that needs to be accessed after an update operation is (i, j). Therefore, based on (7.8) the worst case time is:

$$t(K) = [1 - (1-p)^K] \cdot c \cdot K + (1-p)^K \cdot [c \cdot K + 2 \cdot c \cdot \log N] \qquad (7.9)$$

The first term in the above equation corresponds to the worst case when the grey level j is actually found in probability list P_i (note that the worst case is to find grey level j in the Kth location in list P_i). The second term corresponds to the worst case when the grey level j is not an element of list P_i. In this case, after the algorithm searches all K locations of P_i it continues the search in BB-tree T_i. The worst situation in this case is when grey level j is stored in a leaf of T_i having depth $2 \log N$ [1], where N is the number of elements in T_i (see Section 5.3) and it is assumed that $N > 1$. Note that the above estimated time for the access operation is a function of K, the length of the probability list.

After performing some arithmetic operations, (7.9) becomes

$$t(K) = c \cdot K + 2 \cdot c \cdot (1-p)^K \cdot \log N \qquad (7.10)$$

For $K = 0$, the above time becomes $t(0) = 2 \cdot c \cdot \log N$. This is actually the worst case time for the plain co-occurrence trees, i.e. without the probability lists. Let us compute the first derivative of $t(K)$, $\dot{t}(K)$. We have that

$$\dot{t}(K) = c + 2 \cdot c \cdot \log N \cdot \ln(1-p) \cdot (1-p)^K \qquad (7.11)$$

[1] the reader should recall that the logarithm is base 2

We first find the points where $\dot{t}(K) = 0$. We have that

$$
\begin{aligned}
& c + 2 \cdot c \cdot \log N \cdot \ln(1-p) \cdot (1-p)^K = 0 \Leftrightarrow \\
& (1-p)^K = \frac{-1}{2 \cdot \log N \cdot \ln(1-p)} \Rightarrow \\
& K \cdot \log(1-p) = \log \frac{1}{2 \cdot \log N \cdot \ln \frac{1}{1-p}} \Rightarrow \\
& K = \frac{-\log 2 - \log\log N - \log\ln \frac{1}{1-p}}{\log(1-p)} \Rightarrow \\
& K = \frac{1 + \log\log N + \log\log \frac{1}{1-p} - \log\log e}{\log \frac{1}{1-p}} \Rightarrow \\
& K \equiv K_{opt} = \frac{0.47 + \log\log N + \log\log \frac{1}{1-p}}{\log \frac{1}{1-p}} \qquad (7.12)
\end{aligned}
$$

Since $0 < p < 1$ and $N > 1$, the logarithms in (7.9), (7.10), (7.11), and (7.12) are defined.

Computing now the second derivative of $t(K)$, $\ddot{t}(K)$ we have that

$$\ddot{t}(K) = 2 \cdot c \cdot \log N \cdot \ln^2(1-p) \cdot (1-p)^K \qquad (7.13)$$

Note that $\ddot{t}(K) > 0$, $\forall K \geq 0$. Also, for $K \longrightarrow \infty$ we have that

$$\lim_{K \to \infty} t(K) = \infty \qquad (7.14)$$

since $1 - p < 1$ and therefore, $\lim_{K \to \infty}(1-p)^K = 0$. Figure 7.4 illustrates $t(K)$ for $K \geq 0$.

In our analysis $K_{opt} < 0$ has no meaning. If this inequality holds, then $t(K) \geq 2 \cdot c \cdot \log N$, $\forall K \geq 0$. Therefore, there is no advantage of using probability lists regarding the worst case time (see previous page). In order for $K_{opt} \geq 0$ to be true, it must hold that $0.47 + \log\log N + \log\log \frac{1}{1-p} \geq 0$, since $\log \frac{1}{1-p} > 0$ (see (7.12)).

The conclusion from the above analysis is that there is a value for the length K of the probability lists ($K = K_{opt}$), where the worst case time is minimum (optimal length). This length depends on N (the number of elements stored in BB-tree T_i) and the update rule which is represented by the worst case probability p. It is obvious that as the length K of the probability lists increases, $t(K)$ decreases until it reaches the minimum. After this point $t(K)$ increases to infinity.

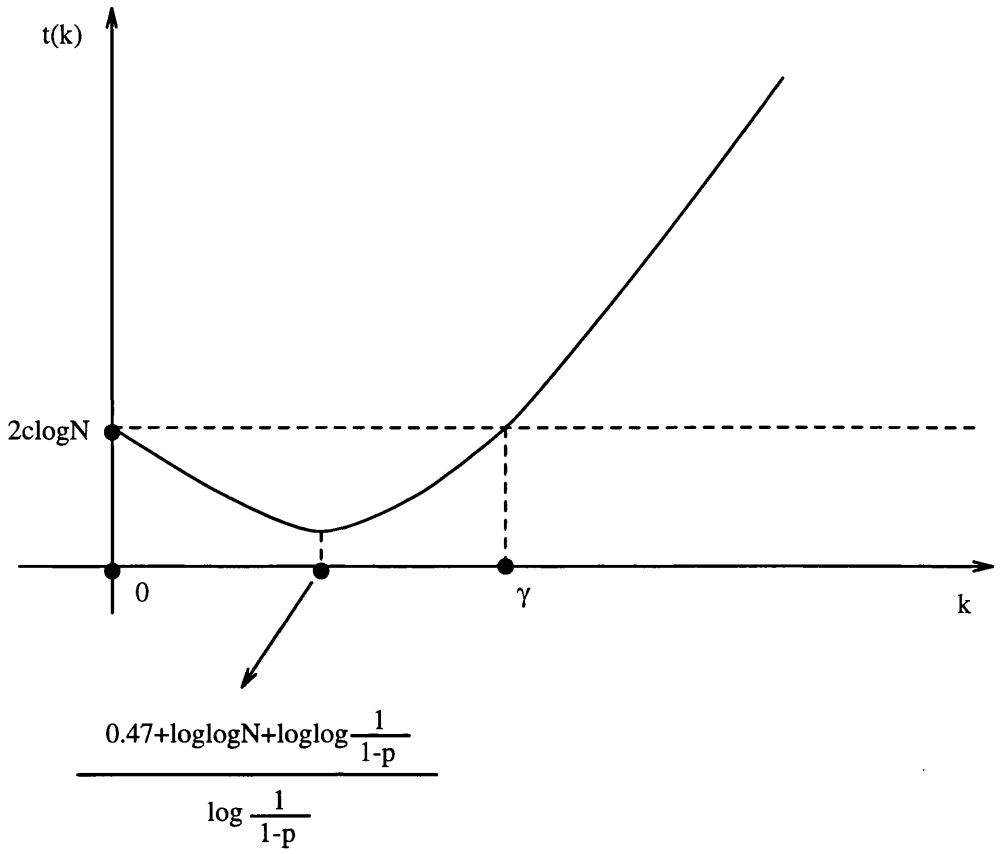


Figure 7.4: Plot of the worst case time cost of the access operation in the enhanced co-occurrence trees against the probability list length.

Therefore, there is a point ($K = \gamma$) after which $t(K)$ becomes larger than the corresponding time in the plain co-occurrence trees and so, there is no advantage of using probability lists. The experimental results from the application of the enhanced co-occurrence trees to the analysis of real textures in Section 9.4 confirm the existence of an optimal length for the probability lists. At this point, we need to emphasise that the above analysis is general enough, since no assumption is made about the probability distribution which the update rule employs, in order to choose the grey levels that puts in the probability lists during an update operation.

In Tables 7.1 (a)–(g), we show the optimal length K_{opt}, the worst time of the enhanced co-occurrence trees for this length (t_1), and the worst time of the plain co-occurrence trees (t_2), for various values of p and N. Cost c is assumed to be equal to 1. In the cases where no result is shown, the corresponding (N, p) pairs give K_{opt} a negative value. Therefore, time t_1 has no meaning.

p	K_{opt}	t_1	t_2
0.01	–	–	8
0.05	–	–	8
0.1	–	–	8
0.2	3	7.10	8
0.3	3	5,74	8

(a) $N = 16$.

p	K_{opt}	t_1	t_2
0.01	–	–	10
0.05	–	–	10
0.1	1	10	10
0.2	4	8.10	10
0.3	4	6.40	10

(b) $N = 32$.

p	K_{opt}	t_1	t_2
0.01	–	–	12
0.05	–	–	12
0.1	2	11.72	12
0.2	4	8,92	12
0.3	4	6.88	12

(c) $N = 64$.

p	K_{opt}	t_1	t_2
0.01	–	–	14
0.05	–	–	14
0.1	4	13.19	14
0.2	5	9.59	14
0.3	5	7.35	14

(d) $N = 128$.

p	K_{opt}	t_1	t_2
0.01	–	–	16
0.05	–	–	16
0.1	5	14.45	16
0.2	6	10.19	16
0.3	5	7.69	16

(e) $N = 256$.

p	K_{opt}	t_1	t_2
0.01	–	–	18
0.05	–	–	18
0.1	6	15.75	18
0.2	6	10.72	18
0.3	5	8.03	18

(f) $N = 512$.

p	K_{opt}	t_1	t_2
0.01	–	–	20
0.05	1	20	20
0.1	7	16.57	20
0.2	7	11.19	20
0.3	6	8.35	20

(g) $N = 1024$.

Table 7.1: Optimal probability list length (K_{opt}) and worst time results from the simulation of the enhanced co-occurrence trees (t_1) and the plain co-occurrence trees (t_2) based on the model of page 181.

From the results of the above tables, it is clear that for very low probabilities p ($p < 0.1$), the enhanced co-occurrence trees give no advantage at all (recall that when $K_{opt} < 0$, the enhanced co-occurrence trees approach is no better than the plain co-occurrence trees, for all $K \geq 0$). For larger p, the worst time for an access operation in the enhanced approach becomes less than the corresponding time in the plain approach. In particular, for $p = 0.3$ the worst access time in the enhanced co-occurrence trees becomes approximately half of the corresponding time in the plain co-occurrence trees. Since probability p depends on the update rule being applied, it becomes clear that this rule plays a central role in the time performance of the enhanced approach. In order that this approach becomes more efficient than the plain co-occurrence trees, the worst case probability p should not be too low (not less than 0.1 according to the above analysis).

From the above results, it is also true that as the number of elements N stored in the BB-tree increases, the enhanced approach becomes better. This is because the cost for accessing the BB-tree, which is $2 \cdot c \cdot \log N$ in the worst case, increases. Therefore, the use of the probability lists becomes more efficient. Moreover, the optimal length K_{opt} of the probability lists is not large in any case. In the above analysis, even for $N = 1024$ the length K_{opt} is not more than 7. Therefore, the space complexity of the enhanced version is not expected to be much worse than in the plain co-occurrence trees.

In the following, similarly to Section **6**.2, we present some lemmas about the memory space and time complexity of the enhanced co-occurrence trees. Again, we perform a worst case analysis. We also present a theorem that shows the least improvement of this approach compared with the co-occurrence matrix. Let N_g be the number of grey levels (grey level range) in an image and n_g the number of grey levels (local grey level range) in the analysed region. Let also K be the length of the probability lists.

Lemma 7.2.1 *The worst case space complexity of the enhanced co-occurrence trees employing the static rule is* $O(n_g^2+(K+1)\cdot N_g)$ *for the semi-dynamic version and* $O(n_g^2 + K \cdot N_g)$ *for the full-dynamic version. The corresponding complexity of the enhanced co-occurrence trees employing the move to front or the counter rule is* $O(n_g^2 + N_g)$ *for the semi-dynamic version and* $O(n_g^2)$ *for the full-dynamic version.*

Proof:

The proof is similar to the proof of Lemma **6**.2.1. The only difference stems from the additional memory space required by the probability lists. In the static rule, N_g probability lists need to be used in the worst case, for the analysis of the given texture. Each list requires $O(K)$ memory space, since each node in the probability lists needs $O(1)$ space. Therefore, the total additional memory space is $O(K \cdot N_g)$. According to Lemma **6**.2.1, the total space needed by the plain co-occurrence trees is $O(n_g^2 + N_g)$ for the semi-dynamic version and $O(n_g^2)$ for the full-dynamic version. Thus, the total memory space of the enhanced co-occurrence trees employing the static rule is

$$O(n_g^2 + N_g) + O(K \cdot N_g) = O(n_g^2 + (K+1) \cdot N_g) \qquad (7.15)$$

for the semi-dynamic version and

$$O(n_g^2) + O(K \cdot N_g) = O(n_g^2 + K \cdot N_g) \qquad (7.16)$$

for the full-dynamic version.

In the cases of the move to front and counter rule, there are only n_g probability lists in the worst case (as many as the grey levels in the analysed local region). Similarly, each list requires $O(K)$ space. Therefore, the additional memory space is $O(K \cdot n_g)$. Based again on Lemma **6**.2.1, the total space for the enhanced co-occurrence trees employing the move to front or the counter rule is

$$O(n_g^2 + N_g) + O(K \cdot n_g) = O(n_g^2 + N_g + K \cdot n_g) = O(n_g^2 + N_g) \qquad (7.17)$$

for the semi-dynamic version and

$$O(n_g^2) + O(K \cdot n_g) = O(n_g^2 + K \cdot n_g) = O(n_g^2) \qquad (7.18)$$

for the full-dynamic version. □

In the analysis of the time behaviour of the co-occurrence trees with probability lists, we will follow the general model presented at the beginning of this section. According to this model, the worst case access time in the enhanced co-occurrence trees is given by the equation (**7**.10). The algorithms for the $p_x(i)$ and $p_y(j)$ elementary operations are the same as the corresponding ones in the plain co-occurrence trees. The reason is that these operations do not have to search for a particular grey level in the BB-trees. The $p_x(i)$ elementary operation needs to access all nodes in tree T_i, while the $p_y(j)$ operation needs to access all nodes in list L_j (see Section **6**.1). Therefore, these operations do not have to use the probability lists and thus, their time complexity is given by Lemmas **6**.2.2 and **6**.2.3 in the case of the enhanced co-occurrence trees, as well.

However, the time complexity of the other two elementary operations, namely $p_{x+y}(\lambda)$ and $p_{x-y}(\lambda)$, differ from the time complexity of their equivalents in the plain co-occurrence trees. This is because during the execution of these operations, the algorithm needs to search for specific grey levels in the BB-trees. Therefore, the use of the probability lists may speed up their computation. This becomes more clear in the following lemmas.

Lemma 7.2.2 *The worst case time complexity of the $p_{x+y}(\lambda)$ elementary operation is $O\left\{(\lambda+1)\cdot[K+(1-p)^K\log n_g]\right\}$ for the semi-dynamic version and $O\left\{(\lambda+1)\cdot(K+\log n_g)\right\}$ for the full-dynamic version, if $0 \leq \lambda \leq n_g - 2$. In the case where $n_g - 1 \leq \lambda \leq 2(n_g-1)$, the corresponding worst case time complexities become $O\left\{(2n_g-\lambda-1)\cdot[K+(1-p)^K\log n_g]\right\}$ for the semi-dynamic version and $O\left\{(2n_g-\lambda-1)\cdot(K+\log n_g)\right\}$ for the full-dynamic version.*

Proof:

The proof is similar to the proof of Lemma **6**.2.4. The only difference is that an access operation now costs

$$O\{t(K)\} = O\left\{c\cdot K + 2\cdot c\cdot(1-p)^K\cdot\log n_g\right\} = O\left\{K+(1-p)^K\cdot\log n_g\right\} \tag{7.19}$$

instead of $O(\log n_g)$, in the worst case. According to the proof of Lemma **6**.2.4, there are two cases for parameter λ:

- $0 \leq \lambda \leq n_g - 2$

  In this case, the total time is

$$(\lambda + 1) \cdot \left\{O(1) + O(K + (1-p)^K \cdot \log n_g)\right\} = O\left\{(\lambda + 1) \cdot \left[K + (1-p)^K \cdot \log n_g\right]\right\} \quad (7.20)$$

  for the semi-dynamic version and

$$(\lambda + 1) \cdot \left\{O(\log n_g) + O(K + (1-p)^K \cdot \log n_g)\right\} = O\left\{(\lambda + 1) \cdot (K + \log n_g)\right\} \quad (7.21)$$

  for the full-dynamic version, since $p < 1$ and therefore, $(1-p)^K < 1$.

- $n_g - 1 \leq \lambda \leq 2(n_g - 1)$

  In this case, the computational time complexity for the $p_{x+y}(\lambda)$ operation becomes

$$(2n_g - \lambda - 1) \cdot \left\{O(1) + O(K + (1-p)^K \cdot \log n_g)\right\} = O\left\{(2n_g - \lambda - 1) \cdot [K + (1-p)^K \cdot \log n_g]\right\} \quad (7.22)$$

  for the semi-dynamic version and

$$(2n_g - \lambda - 1) \cdot \left\{O(\log n_g) + O(K + (1-p)^K \cdot \log n_g)\right\} = O\left\{(2n_g - \lambda - 1) \cdot (K + \log n_g)\right\} \quad (7.23)$$

  for the full-dynamic version.

□

Note that the worst case time complexity of the full-dynamic version does not depend on the worst case probability p. This is because the worst time for accessing the root tree, which is $O(\log n_g)$, is always greater than the worst time for accessing a BB-tree, which is $O\left((1-p)^K \cdot \log n_g\right)$ in the enhanced co-occurrence trees.

Lemma 7.2.3 *The worst case time complexity of the $p_{x-y}(\lambda)$ elementary operation is $O\left\{(n_g - \lambda) \cdot \left[K + (1-p)^K \cdot \log n_g\right]\right\}$ for the semi-dynamic version and $O\left\{(n_g - \lambda) \cdot (K + \log n_g)\right\}$ for the full-dynamic version.*

Proof:

Similarly to the proof of Lemma **6**.2.5, in the worst case the algorithm needs to search $2(n_g - \lambda)$ BB-trees. Each search in a BB-tree costs $O(K + (1-p)^K \log n_g)$, according to the adopted model (see (**7**.19)). Based on the proof of Lemma **6**.2.5, the worst case time complexity of the semi-dynamic version is given by:

$$2(n_g - \lambda) \cdot \left\{O(1) + O(K + (1-p)^K \cdot \log n_g)\right\} = O\left\{(n_g - \lambda) \cdot \left[K + (1-p)^K \cdot \log n_g\right]\right\} \quad (\mathbf{7}.24)$$

and the corresponding time complexity of the full-dynamic version is given by:

$$2(n_g - \lambda) \cdot \left\{O(\log n_g) + O(K + (1-p)^K \cdot \log n_g)\right\} = O(\{(n_g - \lambda) \cdot (K + \log n_g)\} \quad (\mathbf{7}.25)$$

Note that the only difference between the above relations and the corresponding ones in Lemma **6**.2.5 comes from the replacement of the access time $O(\log n_g)$ with the access time $O(K + (1-p)^K \log n_g)$ in the enhanced co-occurrence trees. Note also that the full-dynamic version does not depend on the parameter p, here as well. □

We can now proceed to the following theorem, which gives the reduction in memory space and time complexity of the enhanced co-occurrence trees compared with the co-occurrence matrix.

Theorem 7.2.1 *The reduction in memory space complexity using the enhanced co-occurrence trees instead of the co-occurrence matrix, is at least* $\Omega\left(\frac{N_g^2}{n_g^2+(K+1)N_g}\right)$ *times for the semi-dynamic version and at least* $\Omega\left(\frac{N_g^2}{n_g^2+K\cdot N_g}\right)$ *times for the full-dynamic version, when the static rule is employed. In the cases of the move to front and counter rule, the corresponding reduction is at least* $\Omega\left(\frac{N_g^2}{n_g^2+N_g}\right)$ *times for the semi-dynamic version and at least* $\Omega\left(\left(\frac{N_g}{n_g}\right)^2\right)$ *times for the full-dynamic version. The reduction in time complexity for the texture features using the elementary operations* $p_{x+y}(\lambda)$ *and* $p_{x-y}(\lambda)$*, is at least* $\Omega\left(\frac{N_g^2}{n_g^2\cdot[K+(1-p)^K\cdot\log n_g]}\right)$ *for the semi-dynamic version and at least* $\Omega\left(\frac{N_g^2}{n_g^2\cdot(K+\log n_g)}\right)$ *times for the full-dynamic version.*

Proof:

The proof is similar to the proof of Theorem **6**.2.1. According to Lemma **7**.2.1, the worst case space complexity of the enhanced co-occurrence trees is proven to be $O(n_g^2+(K+1)\cdot N_g)$ for the semi-dynamic version and $O(n_g^2+K\cdot N_g)$ for the full-dynamic version, when the static rule is employed. In the cases of the move to front and counter rule, the corresponding complexity is proven to be $O(n_g^2+N_g)$ for the semi-dynamic version and $O(n_g^2)$ for the full-dynamic version. Since the co-occurrence matrix needs $O(N_g^2)$ space in all cases (see proof of Theorem **6**.2.1), the reduction in memory space is at least $\Omega\left(\frac{N_g^2}{n_g^2+(K+1)N_g}\right)$ times for the semi-dynamic version and at least $\Omega\left(\frac{N_g^2}{n_g^2+K\cdot N_g}\right)$ times for the full-dynamic version, when the static rule is employed. The corresponding reduction is at least $\Omega\left(\frac{N_g^2}{n_g^2+N_g}\right)$ times for the semi-dynamic version and at least $\Omega\left(\left(\frac{N_g}{n_g}\right)^2\right)$ times for the full-dynamic version, when the move to front or the counter rule is employed.

Based on Lemma **7**.2.2, the worst case time complexity for the $p_{x+y}(\lambda)$ elementary operation in the semi-dynamic version is given by (**7**.20) if $0\leq\lambda\leq n_g-2$ and (**7**.22) if $n_g-1\leq\lambda\leq 2(n_g-1)$. Again similarly to the proof of Theorem **6**.2.1, the total worst time needed for the texture features using $p_{x+y}(\lambda)$ operation is

$$\sum_{\lambda=0}^{n_g-2} O\left\{(\lambda+1)\cdot\left[K+(1-p)^K\cdot\log n_g\right]\right\}+$$
$$\sum_{\lambda=n_g-1}^{2(n_g-1)} O\left\{(2n_g-\lambda-1)\cdot\left[K+(1-p)^K\cdot\log n_g\right]\right\}=$$
$$O\left\{n_g^2\cdot\left[K+(1-p)^K\cdot\log n_g\right]\right\} \qquad (7.26)$$

in the semi-dynamic version. The corresponding time for the co-occurrence matrix is $O(N_g^2)$ in all cases (see proof of Theorem **6**.2.1). Therefore, the speedup is at least $\Omega\left(\frac{N_g^2}{n_g^2\cdot[K+(1-p)^K\cdot\log n_g]}\right)$.

The worst case time complexity for the $p_{x+y}(\lambda)$ elementary operation in the full-dynamic version is given by (**7**.21), in the case where $0\leq\lambda\leq n_g-2$ and (**7**.23), in the case where $n_g-1\leq\lambda\leq 2(n_g-1)$. Again, the total worst time is:

$$\sum_{\lambda=0}^{n_g-2} O\left\{(\lambda+1)\cdot(K+\log n_g)\right\}+$$
$$\sum_{\lambda=n_g-1}^{2(n_g-1)} O\left\{(2n_g-\lambda-1)\cdot(K+\log n_g)\right\}=$$
$$O\left\{n_g^2\cdot(K+\log n_g)\right\} \qquad (7.27)$$

Therefore, the speedup for the texture features that use the $p_{x+y}(\lambda)$ operation in the case of the full-dynamic version, is at least $\Omega\left(\frac{N_g^2}{n_g^2\cdot(K+\log n_g)}\right)$.

According to Lemma **7**.2.3, the worst case time complexity for the $p_{x-y}(\lambda)$ elementary operation is given by (**7**.24) for the semi-dynamic version. Similarly to the proof of Theorem **6**.2.1, the total worst time for the texture features that use $p_{x-y}(\lambda)$ operation is:

$$\sum_{\lambda=0}^{n_g-1} O\left\{(n_g-\lambda)\cdot\left[K+(1-p)^K\cdot\log n_g\right]\right\}=O\left\{n_g^2\cdot\left[K+(1-p)^K\cdot\log n_g\right]\right\} \qquad (7.28)$$

According to the proof of the same theorem, the co-occurrence matrix needs $O(N_g^2)$ time for these texture features, in all cases. Therefore, the speedup that is achieved is at least $\Omega\left(\frac{N_g^2}{n_g^2\cdot[K+(1-p)^K\cdot\log n_g]}\right)$.

The worst case time complexity for the $p_{x-y}(\lambda)$ operation in the full-dynamic version is given by (**7**.25), according to Lemma **7**.2.3. Similarly to the proof of Theorem **6**.2.1, the total worst time for the texture features that use this elementary operation in this version of the enhanced co-occurrence trees is

$$\sum_{\lambda=0}^{n_g-1} O\left\{(n_g-\lambda)\cdot(K+\log n_g)\right\} = O\left\{n_g^2\cdot(K+\log n_g)\right\} \qquad (7.29)$$

Therefore, the speedup is at least $\Omega\left(\frac{N_g^2}{n_g^2\cdot(K+\log n_g)}\right)$. □

From the above theorem and Theorem **6**.2.1, we note that the space complexities of the enhanced co-occurrence trees are increased by the factor $K\cdot N_g$, compared with the corresponding complexities of the plain co-occurrence trees, when the static rule is employed. Although K is kept small, $K\cdot N_g$ can be significant, especially for images with a large dynamic range. We also note that the space complexities in the cases of the move to front and counter rule remain the same. This is because the part of the memory space that is attributed to the probability lists in these cases is $K\cdot n_g$, which is very small compared to the n_g^2 term, i.e. the memory space required by the rest of the co-occurrence trees. Therefore, n_g^2 is the dominant factor in the space complexity of these rules.

As far as the $p_{x+y}(\lambda)$ and $p_{x-y}(\lambda)$ elementary operations are concerned, the worst case speedup in the semi-dynamic version is greater than the corresponding one in the plain co-occurrence trees. This is because the denominator $n_g^2\cdot\log n_g$ in the lower bound expression of the plain co-occurrence trees is replaced by the $n_g^2\cdot\left[K+(1-p)^K\cdot\log n_g\right]$ term in the enhanced co-occurrence trees. Since K is small (usually less than 5) and $(1-p)^K<1$, this term is in many cases less than the term $n_g^2\cdot\log n_g$ that appears in the time complexity of the plain co-occurrence trees. Therefore, the least reduction in computational time using the semi-dynamic version of the enhanced co-occurrence trees instead of the co-occurrence matrix, is greater than the corresponding reduction using the semi-dynamic version of the plain co-occurrence trees.

In the case of the full-dynamic version, there is a decrease in the worst case speedup achieved. The denominator $n_g^2\cdot\log n_g$ in the lower bound expression in the plain co-occurrence trees is substituted with the term $n_g^2\cdot(K+\log n_g)$ in the

case of the enhanced co-occurrence trees, which is larger. As we have already said, this happens because in the full-dynamic version, in order to access a BB-tree we first need to access the root tree. This access operation takes $O(\log n_g)$ time which is much greater than the term $O\left((1-p)^K \cdot \log n_g\right)$, the time to access a BB-tree in the adopted model. Therefore, in the full-dynamic version, plain co-occurrence trees seem to be better in terms of the computational time complexity.

Finally, two points need to be stressed out. First, the complexities presented in this section represent worst case memory space and computational time results. In reality, the performance of the enhanced co-occurrence trees is expected to be better. This will become more clear from the results of the application of this method to real data. Second, not all update rules exhibit the same time performance, even if the worst case probability p is the same. For example, the counter rule needs to sort the grey levels stored in the probability lists in decreasing order, according to their access frequencies. This structural change takes $O(K^2)$ time in the worst case and $O(K \log K)$ time in the average case, using the quicksort algorithm which is one of the best sorting algorithms (see [86]). This time has to be taken into consideration in the estimation of the total computational time of the texture feature extraction process, when using the counter rule.

Chapter 8

Self-adjusting co-occurrence trees

8.1 Description

We saw in Section 5.5 that the weighted dictionary problem can be expanded to binary search trees. In this case, the aim is to put the elements with the largest access probabilities nearer the root of a tree (the entry point of an access operation), in order to reduce their access time in subsequent accesses. We also saw that the self-adjusting binary trees possess this valuable property, i.e. they can restructure themselves in order that the most frequently accessed elements be placed nearer their roots. In particular, the self-adjusting binary trees have the ability to dynamically adapt their structure to a non-uniform access distribution, in order to improve their time efficiency. Since the co-occurrence distribution of the grey levels in most real textures satisfies this condition, it is worth investigating the efficiency of these trees, in terms of memory space and computational time. The reader should recall at this point, that the co-occurrence distribution of the grey levels and the access distribution of the grey levels in the co-occurrence trees are identical. Substituting the BB-trees in the plain co-occurrence trees with self-adjusting binary trees, we get the **self-adjusting co-occurrence trees** ([153]). This is actually the third proposed approach in this thesis for the SGLDM.

The aforementioned adaptation of the self-adjusting binary trees is accomplished by performing appropriate restructuring operations. In this thesis, we will investigate four different restructuring operations. Two of them have already

been presented in Section **5**.5. These are the move to root restructuring operation and the splaying operation (splay trees). The implementation of the splaying operation that was presented in the above section, is based on the *bottom-up* method, i.e. the algorithm first accesses the node storing the requested grey level and then it splays at this node. An alternative way to perform splaying is according to the *top-down* method. This method performs splaying on the way down the tree during the access. Thus, it avoids traversing the access path for a second time, as in the bottom-up method. In the following, we will show how this method works.

Suppose that we want to access grey level x in a splay tree. During the access and concurrent splaying operation, the tree is broken into three pieces, i.e. a left tree, a middle tree, and a right tree. The left and right trees contain all grey levels in the original tree so far known to be less than x and greater than x, respectively. The middle tree is the subtree of the original tree rooted at the currently accessed node on the access path. Initially, the middle tree is the original tree and the left and right trees are empty. The top-down splaying algorithm searches down from the root for the requested element x, two nodes at a time, breaking links along the access path and adding each subtree detached to the bottom right of the left tree or the bottom left of the right tree. If the algorithm takes two steps down in the same direction, though, it first performs one rotation before breaking a link. Actually, this splaying operation consists of repeatedly applying the appropriate case from the three cases described below.

Suppose that the search has reached node u.

- **zig-case:** it is applied when node v contains the requested grey level x (see Figure **8**.1),
- **zig-zig case:** the algorithm applies one rotation to the $u - v$ link, before it breaks the $v - y$ link in the middle tree (see Figure **8**.2),
- **zig-zag case:** the algorithm first breaks the $u - v$ link and then it breaks the $v - y$ link in the middle tree (see Figure **8**.3).

The symmetrical variants of the above cases are omitted.

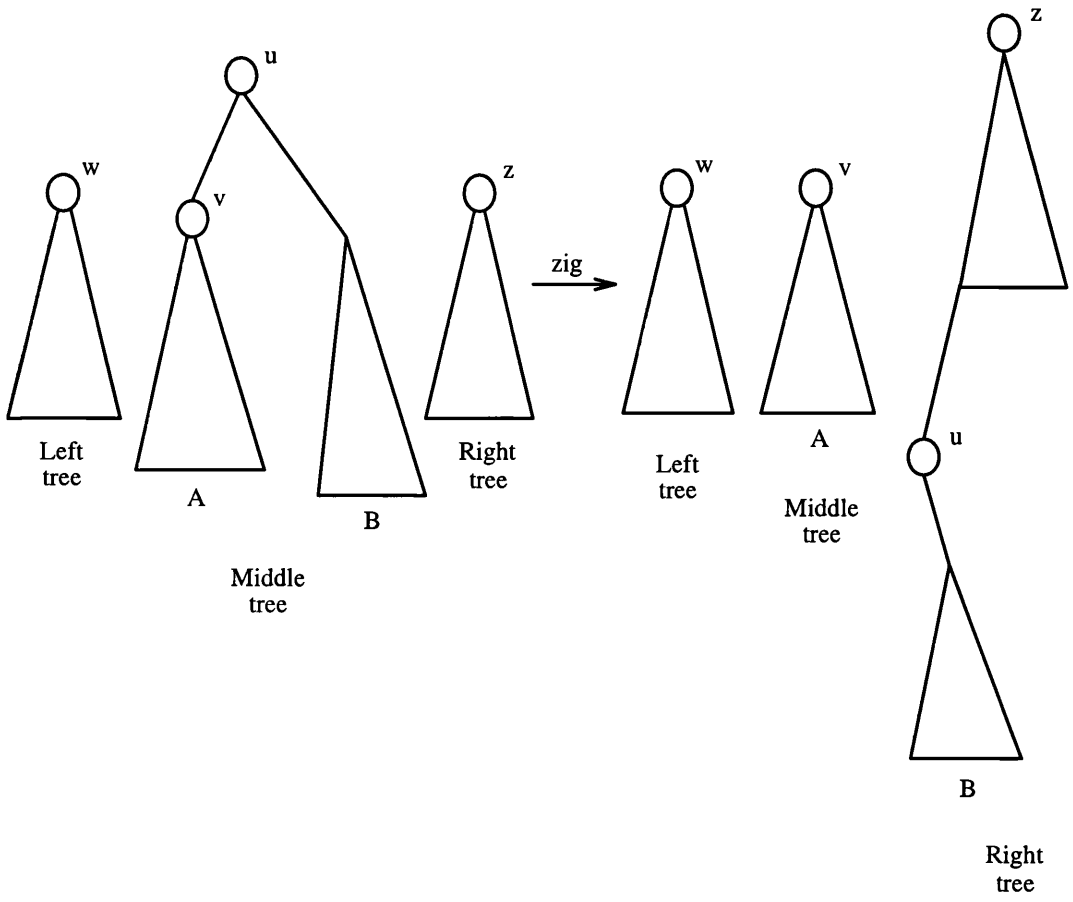


Figure 8.1: The zig case in top-down splaying.

After none of the aforementioned cases applies, the algorithm reassembles the three remaining trees, as shown in Figure 8.4[1]. In this figure, node u contains the requested grey level x. The resulting tree is the new splay tree after restructuring.

One significant disadvantage of splaying is the large amount of restructuring it needs to perform (see Section 5.5.1). One way to reduce this amount and at the same time preserve some of the properties of splay trees is the *semi-splaying* method. The idea behind this method is to modify the restructuring operation, so that it rotates only some of the edges along an access path, thus moving the node storing the requested grey level only partway towards the root. Semi-splaying has many variants, depending on the condition that must be fulfilled in order that the restructuring algorithm stops.

[1]Figures 8.1, 8.2, 8.3, and 8.4 were modified from [141]

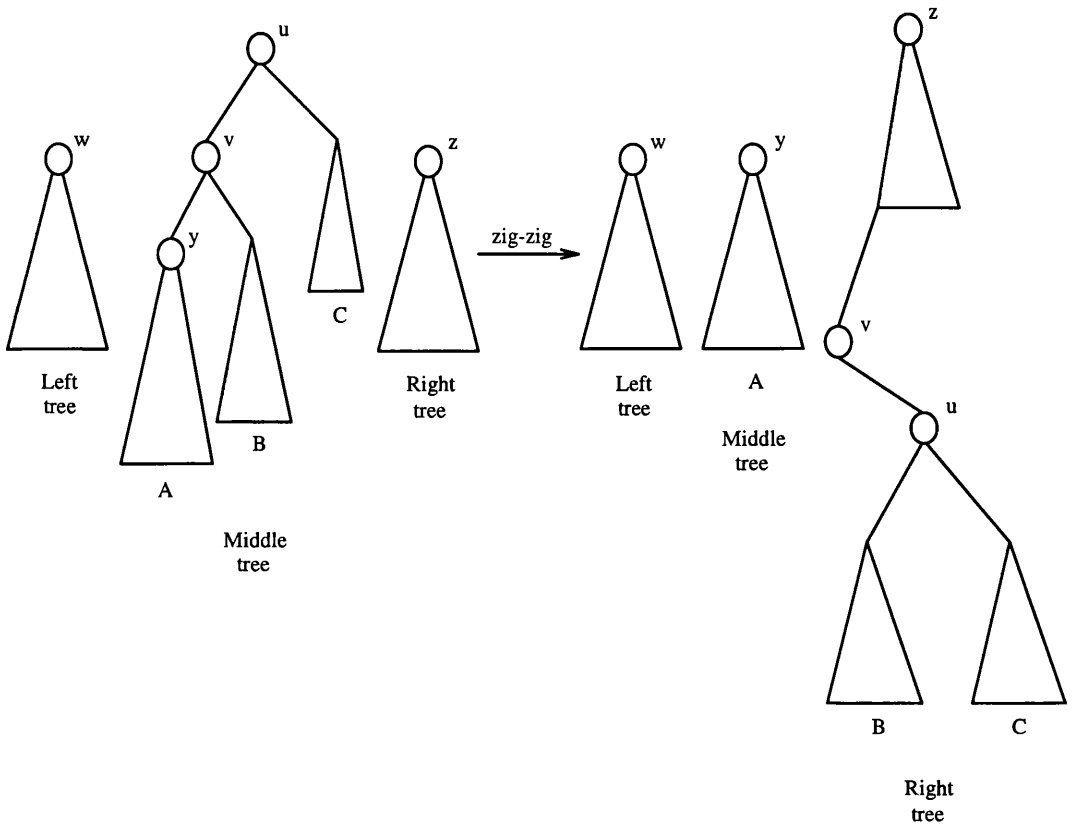


Figure 8.2: The zig-zig case in top-down splaying.

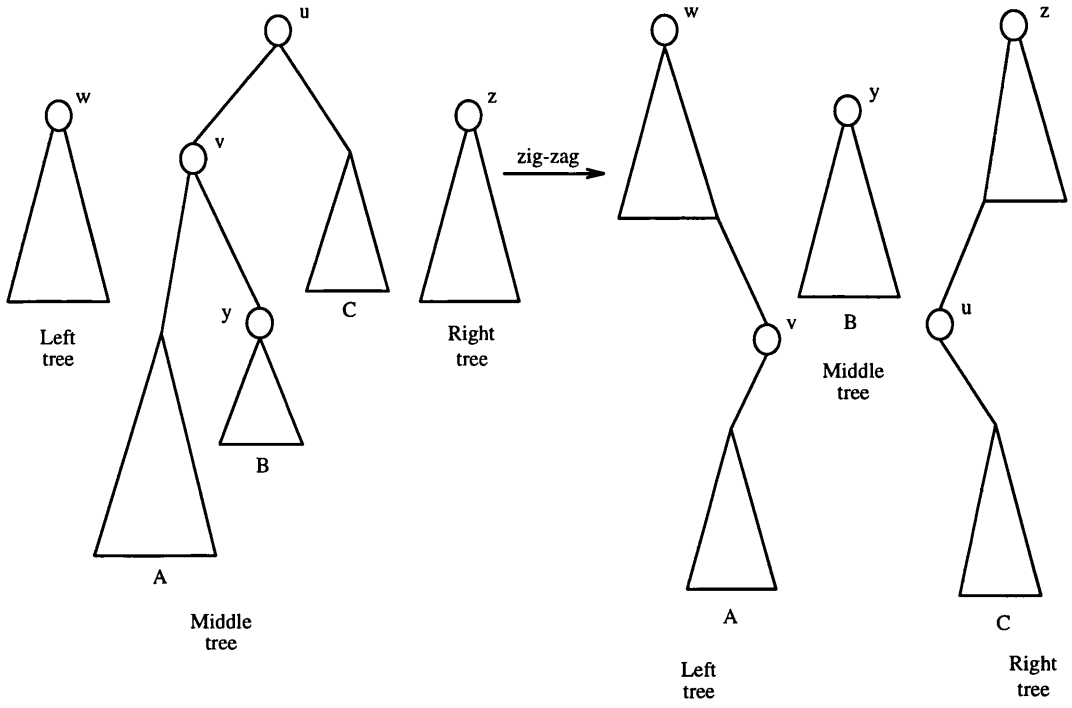


Figure 8.3: The zig-zag case in top-down splaying.

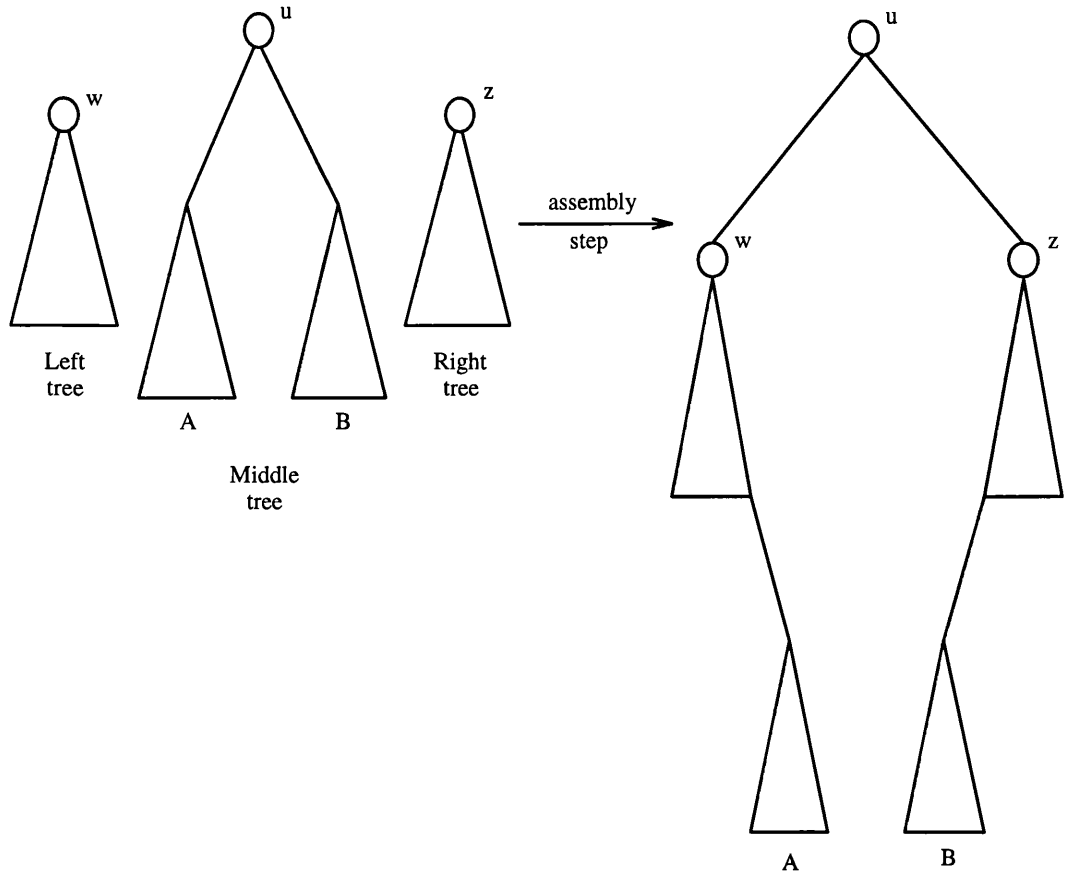


Figure 8.4: The final step in top-down splaying.

In this thesis, we propose a novel semi-splaying rule which is specifically tailored to texture analysis, since it takes advantage of the textural information stored in the co-occurrence trees. In particular, the algorithm takes into consideration the access frequency of each grey level on its access path in a tree which, as we have already said, is actually the co-occurrence frequency of the corresponding grey level pair (the first member of the pair is the grey level that corresponds to the tree, while the second member is the accessed grey level). The proposed semi-splaying operation proceeds as follows.

After the node that stores the desired grey level is accessed, the splaying operation commences and continues until the above node becomes the root of the tree or the counter (co-occurrence frequency) of the node currently accessed is greater or equal to the counter of this node. The intuition behind this restructuring rule is that on the way up towards the root, if a node is met that has

greater co-occurrence frequency than the node that is pushed upwards (the node that contains the requested grey level), then that node must be higher. This is because it is accessed more frequently and therefore, it must be nearer the root. The time efficiency of the proposed semi-splaying method will be evaluated through its application to the analysis of real data and compared with the other three restructuring methods presented here.

The two primitive operations access and insert are implemented in the way described in Section **5**.5.1, in the general case of the self-adjusting binary trees, as well. The only difference is that function $splay(\cdot)$ is replaced by a more general function $restructure(\cdot)$. This function performs move to root, bottom-up splaying, top-down splaying, or semi-splaying, according to the restructuring operation applied to the self-adjusting trees that are employed in texture analysis.

Finally, similarly to the plain and enhanced co-occurrence trees, there are two versions of the self-adjusting co-occurrence trees, namely the **semi-dynamic** version and the **full-dynamic** version. Figures **6**.5 and **6**.6 show the corresponding structure of these two versions of this approach, as well. In the semi-dynamic version, the structures that store the roots of the self-adjusting binary trees and the heads of the lists are static arrays. In the full-dynamic version, the above structures are themselves self-adjusting binary trees. Accessing grey level pair (i, j) is performed in the same way as in Section **6**.1. The only difference is that now BB-trees are replaced by self-adjusting binary trees. The space and time efficiency of the two versions of the self-adjusting co-occurrence trees are evaluated separately in the next section.

8.2 Complexity analysis

The worst case space complexity of the self-adjusting co-occurrence trees is the same as the corresponding complexity of the plain co-occurrence trees. This happens because the only difference between them in terms of memory space, is the lack of the balance information from the self-adjusting co-occurrence trees. That does not affect the space complexity, only the relevant constant. Suppose that the actual memory space occupied by a BB-tree is $k \cdot n_g$ bytes in the worst

case, where n_g is the number of grey levels in the analysed region. If the bit that represents the balance information in each node of a BB-tree (the colour of the node; see Section **5**.3) is embedded in the information content of each node, which is not unusual for images having a dynamic range higher than 8 bits, no gain in memory space is achieved using the self-adjusting binary trees instead. If the above bit occupies a byte of memory space in each node of the BB-trees, a decrease of n_g bytes is achieved by employing the self-adjusting binary trees, since the memory space of these trees is $(k-1) \cdot n_g$ bytes in this case. However, the space complexity is the same ($O(n_g)$) for both kinds of trees. Therefore, Lemma **6**.2.1 gives the space complexity of the two versions of the self-adjusting co-occurrence trees (semi-dynamic and full-dynamic), as well.

As far as the computational time complexity is concerned, it has been shown that the amortised time bound for an access operation in a splay tree (a self-adjusting binary tree employing bottom-up splaying, top-down splaying, or semi-splaying) is $O(\log N)$ (see [141]), where N is the number of elements stored in the tree. The only difference between the various kinds of splay trees is in the actual constants of the computational time. At this point, the reader should recall that $O(k \cdot f(n)) = O(f(n))$, where k is a constant. Lemmas **6**.2.2, **6**.2.3, **6**.2.4, and **6**.2.5 give the time complexities in the case of the self-adjusting co-occurrence trees using splay trees, as well. Similarly, Theorem **6**.2.1 also gives the lower bounds on the improvement in space and time using this approach instead of the co-occurrence matrix. The only difference from the plain co-occurrence trees is that the complexities in the above lemmas and theorem are valid for the amortised case in the self-adjusting co-occurrence trees, i.e. when the time average over a worst case sequence of operations is considered. The proofs for the space and time complexities of this approach are exactly the same as the ones presented for the lemmas and theorem in Section **6**.2.

A self-adjusting binary tree employing the move to root restructuring operation, exhibits an amortised access time of $O(N)$, where again N is the number of elements stored in the tree (see [2]). The time complexities of the $p_x(i)$ and $p_y(j)$ elementary operations in the corresponding self-adjusting co-occurrence trees, are also given by Lemmas **6**.2.2 and **6**.2.3, respectively. In the case of the

semi-dynamic version, the proofs are the same as the proofs of these lemmas. In the case of the full-dynamic version, the amortised time complexity for accessing the root tree or the list tree is $O(n_g)$. Following similar arguments as in the proofs of the above lemmas, the total time of the $p_x(i)$ and $p_y(j)$ elementary operations for this version of the self-adjusting co-occurrence trees is proven to be $O(n_g) + O(n_g) = O(2n_g) = O(n_g)$, in the amortised case. The time complexities of the $p_{x+y}(k)$ and $p_{x-y}(k)$ elementary operations are given by the following lemmas:

Lemma 8.2.1 *The computational time complexity of the $p_{x+y}(k)$ elementary operation is $O((k+1) \cdot n_g)$ if $0 \leq k \leq n_g - 2$ and $O((2n_g - k - 1) \cdot n_g)$ if $n_g - 1 \leq k \leq 2(n_g - 1)$, in the amortised case, for the self-adjusting co-occurrence trees using the move to root restructuring operation.*

Proof:

The proof is similar to the proof of Lemma **6**.2.4. The only difference is that the access time for the self-adjusting tree is now $O(n_g)$ instead of $O(\log n_g)$. Replacing the corresponding term in (**6**.10) and (**6**.13), we get that in the case of the semi-dynamic version, the total amortised time is given by

$$(k+1) \cdot \{O(n_g) + O(1)\} = O((k+1) \cdot n_g) \tag{8.1}$$

if $0 \leq k \leq n_g - 2$ and

$$(2n_g - k - 1) \cdot \{O(n_g) + O(1)\} = O((2n_g - k - 1) \cdot n_g) \tag{8.2}$$

if $n_g - 1 \leq k \leq 2(n_g - 1)$.

Doing the same in (**6**.11) and (**6**.14), we get that in the case of the full-dynamic version, the corresponding time is given by

$$(k+1) \cdot \{O(n_g) + O(n_g)\} = O((k+1) \cdot n_g) \tag{8.3}$$

if $0 \leq k \leq n_g - 2$ and

$$(2n_g - k - 1) \cdot \{O(n_g) + O(n_g)\} = O((2n_g - k - 1) \cdot n_g) \tag{8.4}$$

if $n_g - 1 \leq k \leq 2(n_g - 1)$. □

Lemma 8.2.2 *The computational time complexity of the $p_{x-y}(k)$ elementary operation is $O((n_g-k)\cdot n_g)$, in the amortised case, for the self-adjusting co-occurrence trees using the move to root restructuring operation.*

Proof:

The proof is again similar to the proof of Lemma **6**.2.5. We only need to replace the $O(\log n_g)$ access time with the $O(n_g)$ time in (**6**.19) and (**6**.20), in order to get the amortised time complexity of $p_{x-y}(k)$ for the two versions of the self-adjusting co-occurrence trees using the move to root restructuring operation. The corresponding amortised time for the semi-dynamic version is therefore given by

$$2(n_g - k) \cdot \{O(n_g) + O(1)\} = O((n_g - k) \cdot n_g) \tag{8.5}$$

Similarly, the amortised time for the full-dynamic version is given by

$$2(n_g - k) \cdot \{O(n_g) + O(n_g)\} = O(4 \cdot (n_g - k) \cdot n_g) = O((n_g - k) \cdot n_g) \tag{8.6}$$

□

The least reduction in space complexity and the least reduction in the amortised time complexity for the $p_x(i)$ and $p_y(j)$ elementary operations, using the self-adjusting co-occurrence trees with the move to root restructuring operation instead of the co-occurrence matrix, are also given by Theorem **6**.2.1. The proof is the same as the proof of this theorem. In the following theorem, we give lower bounds on the corresponding reduction in the time complexity of the $p_{x+y}(k)$ and $p_{x-y}(k)$ elementary operations.

Theorem 8.2.1 *The reduction in time complexity for the texture features that use the elementary operations $p_{x+y}(k)$ and $p_{x-y}(k)$, employing the self-adjusting co-occurrence trees with the move to root restructuring operation instead of the co-occurrence matrix, is at least $\Omega\left(\frac{N_g^2}{n_g^3}\right)$ times.*

Proof:

Once again, the proof is similar to the corresponding part of the proof of Theorem **6**.2.1. Replacing the $O(\log n_g)$ access time with the $O(n_g)$ time in (**6**.21),

we get that the total amortised time for the texture features that use the $p_{x+y}(k)$ elementary operation is given by

$$\sum_{k=0}^{n_g-2} O((k+1)\cdot n_g) + \sum_{k=n_g-1}^{2(n_g-1)} O((2n_g-k-1)\cdot n_g) = O(n_g^3) \qquad (8.7)$$

for both versions of the self-adjusting co-occurrence trees using the move to root restructuring operation.

Doing the same in (**6**.23), we get that the corresponding time for the texture features that use the $p_{x-y}(k)$ elementary operation is given by

$$\sum_{k=0}^{n_g-1} O((n_g-k)\cdot n_g) = O(n_g^3) \qquad (8.8)$$

again for both the semi-dynamic and the full-dynamic version. Since the corresponding total time for the co-occurrence matrix is $O(N_g^2)$ in all cases (see proof of Theorem **6**.2.1), the least reduction is $\Omega\left(\frac{N_g^2}{n_g^3}\right)$ for both elementary operations in the self-adjusting co-occurrence trees. □

From the above theoretical results, we can conclude that the self-adjusting co-occurrence trees using the move to root restructuring operation are the least efficient among the presented kinds of the self-adjusting co-occurrence trees, in terms of computational time. Still, they are as efficient as the splay co-occurrence trees in terms of space complexity.

The memory space required by the self-adjusting co-occurrence trees is actually their strong point. This approach is indeed the most efficient among all approaches compared in this thesis, in terms of the actual memory required. It remains to see its efficiency compared not only with the co-occurrence matrix, but also with the rest of the co-occurrence trees, when both memory space and computational time are taken into consideration. Its application to real data will shed some light on this question in Section **9**.4.

Chapter 9

Results

9.1 Memory space results

In the first part of this chapter, we present results regarding the memory space requirements of the approaches presented for the implementation of the SGLDM. The results are given in the form of the percentage of the relative reduction in memory space of one approach A with respect to another approach B. That is, if S_A represents the space requirements of approach A in bytes and S_B represents the corresponding requirements of approach B with which approach A is compared, then $\frac{S_B - S_A}{S_B} \cdot 100$ % is the percentage of the relative reduction in memory space of A with respect to B. This implicit form of the memory space requirements of the various approaches presented in this thesis will allow us to draw immediate conclusions about the efficiency of one approach compared to another, in terms of memory space. The sign of the above quantity is important because it indicates whether approach A needs less space than approach B or vice versa. If the sign is negative, then approach B needs less space than approach A. If it is positive then the opposite holds.

The memory space results are shown for the worst case of the proposed approaches, i.e. the different kinds of the co-occurrence trees. Let N_g be the number of grey levels in the analysed image. Let also n_g be the number of grey levels in the analysed local region. In the worst case, there exists at least one displacement vector for which n_g^2 grey level pairs in the local region satisfy the spatial

relationship this vector represents. At this point, we need to stress for one more time the fact that for the co-occurrence matrix approach there is only one case. Its memory space requirements depend on N_g^2 for a given image and not on the number of grey level pairs that satisfy the given spatial relationship in the analysed local region. The results shown here give lower bounds on the improvement in memory space using the proposed approaches in the SGLDM instead of the co-occurrence matrix. Results are presented for three values of N_g, namely $N_g = 64$, $N_g = 256$, and $N_g = 2048$. These values are representative of the dynamic ranges encountered, especially in medical images. The memory space results for other values of the N_g parameter lead to similar conclusions. The n_g parameter ranges from 2 (the least number of grey levels in a textured region) up to $\frac{N_g}{2}$. This range is likely to include all values of this parameter found in most cases in practice.

Figures **9**.1–**9**.3 show the percentage of the relative reduction in memory space for the plain co-occurrence trees. More specifically, the figures show the percentage reduction of the semi-dynamic version relative to the co-occurrence matrix, of the full-dynamic version relative to the co-occurrence matrix, and of the full-dynamic version relative to the semi-dynamic version, respectively.

Figures **9**.4–**9**.6 show the corresponding results for the static rule of the enhanced co-occurrence trees. Figures **9**.7–**9**.9 show the corresponding results for the move to front and counter rule of this approach. The memory space requirements of these two rules are the same, since they only differ in the way they store the grey levels in the probability lists, that is, without order in the case of the move to front rule or in descending order according to their access frequencies, in the case of the counter rule (see Sections **7**.1.2 and **7**.1.3). The results from the enhanced co-occurrence trees are presented for five different values of the length of the probability lists (parameter K; see Section **7**.2), i.e. $K = 1, 2, 3, 4, 5$. In the following, we will see that this set includes the values for which this approach exhibits the best behaviour in terms of computational time. Actually, it is the recommended set of values for the length of the probability lists.

Figures **9**.10–**9**.12 present the relative percentage reduction in memory space for the self-adjusting co-occurrence trees and for the same cases as in the previous two approaches, i.e. semi-dynamic version vs. co-occurrence matrix, full-dynamic

version vs. co-occurrence matrix, and full-dynamic version vs. semi-dynamic version, respectively. Finally, Figure **9**.13 shows the relative percentage reduction in memory space using the self-adjusting co-occurrence trees instead of the plain co-occurrence trees for the semi-dynamic version and the full-dynamic version, respectively. In the case where the number of grey levels in the image is 2048 ($N_g = 2048$), the improvement in memory space achieved using the semi-dynamic version of the self-adjusting co-occurrence trees instead of the semi-dynamic version of the plain co-occurrence trees is very close to +0%, although this does not appear very well in the corresponding graph (see Figure **9**.13 (a)).

Finally, there is no point to compare the enhanced co-occurrence trees with the plain co-occurrence trees, in terms of memory, to see if there is any improvement. This is because, as we saw in Section **7**.1, the enhanced co-occurrence trees approach is actually the plain co-occurrence trees with the addition of the probability lists. Therefore, this approach always needs more space than the plain co-occurrence trees and thus, there will be no improvement in using the enhanced co-occurrence trees instead, in terms of memory. Let c denote the memory space required by a single node of the probability lists in bytes (all nodes need the same space). Based on the proof of Lemma **7**.2.1, in the case of the static rule the additional memory space is $c \cdot K \cdot N_g$ bytes, whereas in the case of the move to front and counter rule the corresponding space is $c \cdot K \cdot n_g$ bytes. Although parameters c and K are usually small, the total additional memory could be large. Thus, when the analysed images have a large number of grey levels and the system employed for texture analysis has a limited memory, this approach should not be used.

RELATIVE PERCENTAGE REDUCTION IN MEMORY SPACE FOR THE PLAIN CO-OCCURRENCE TREES

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

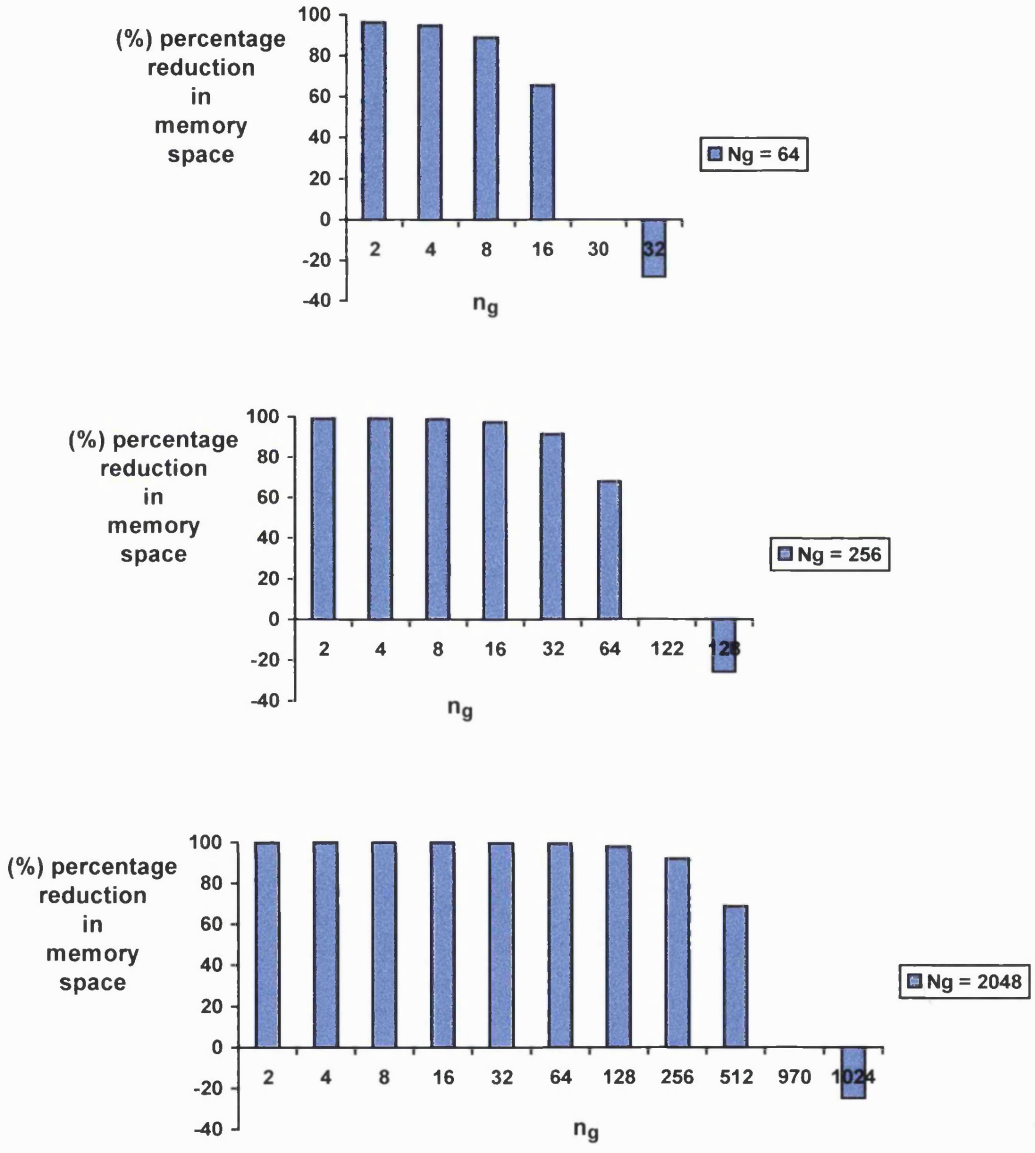


Figure 9.1: Relative percentage reduction in memory space for the plain co-occurrence trees; semi-dynamic version vs. co-occurrence matrix.

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

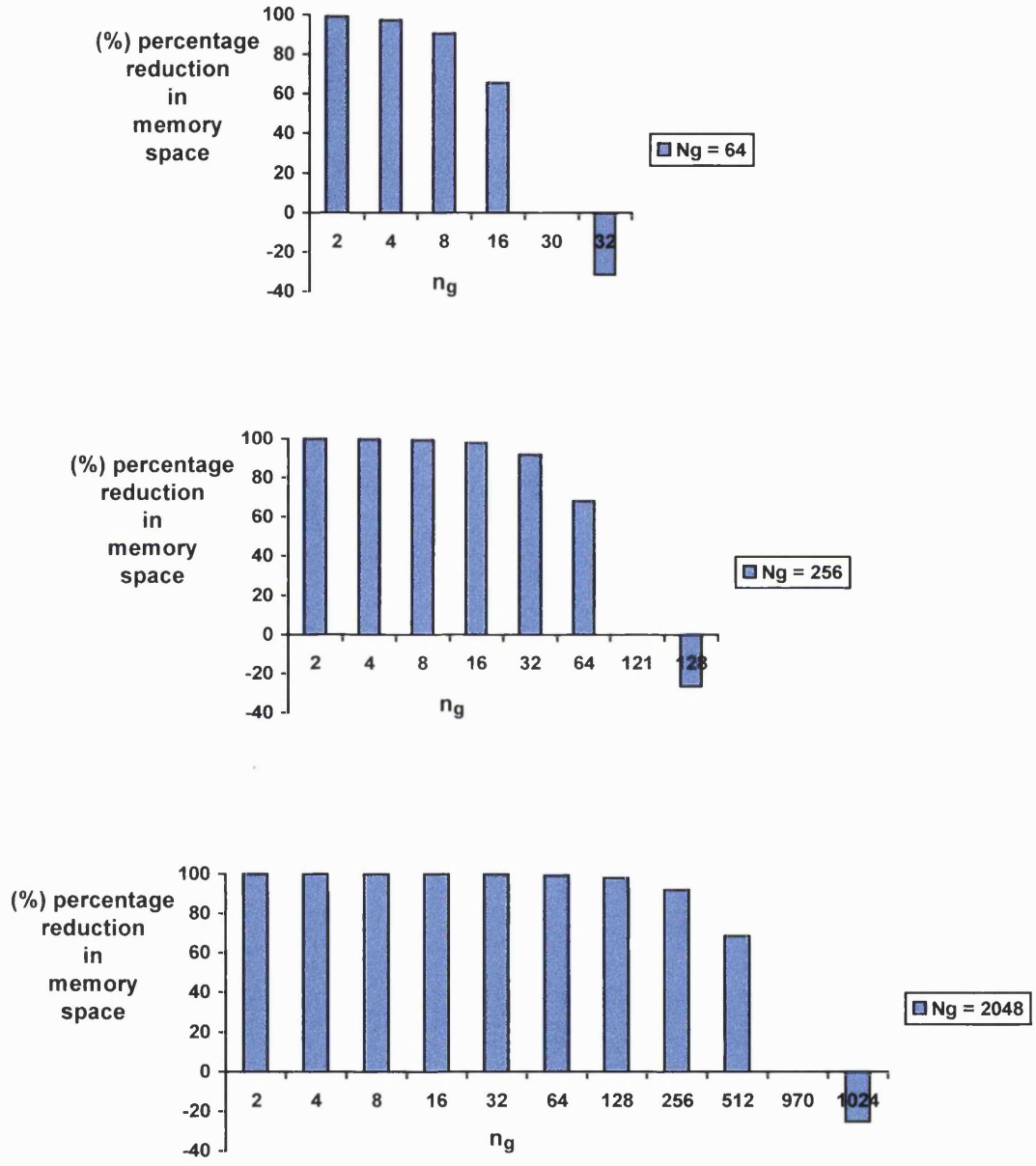


Figure 9.2: Relative percentage reduction in memory space for the plain co-occurrence trees; full-dynamic version vs. co-occurrence matrix.

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

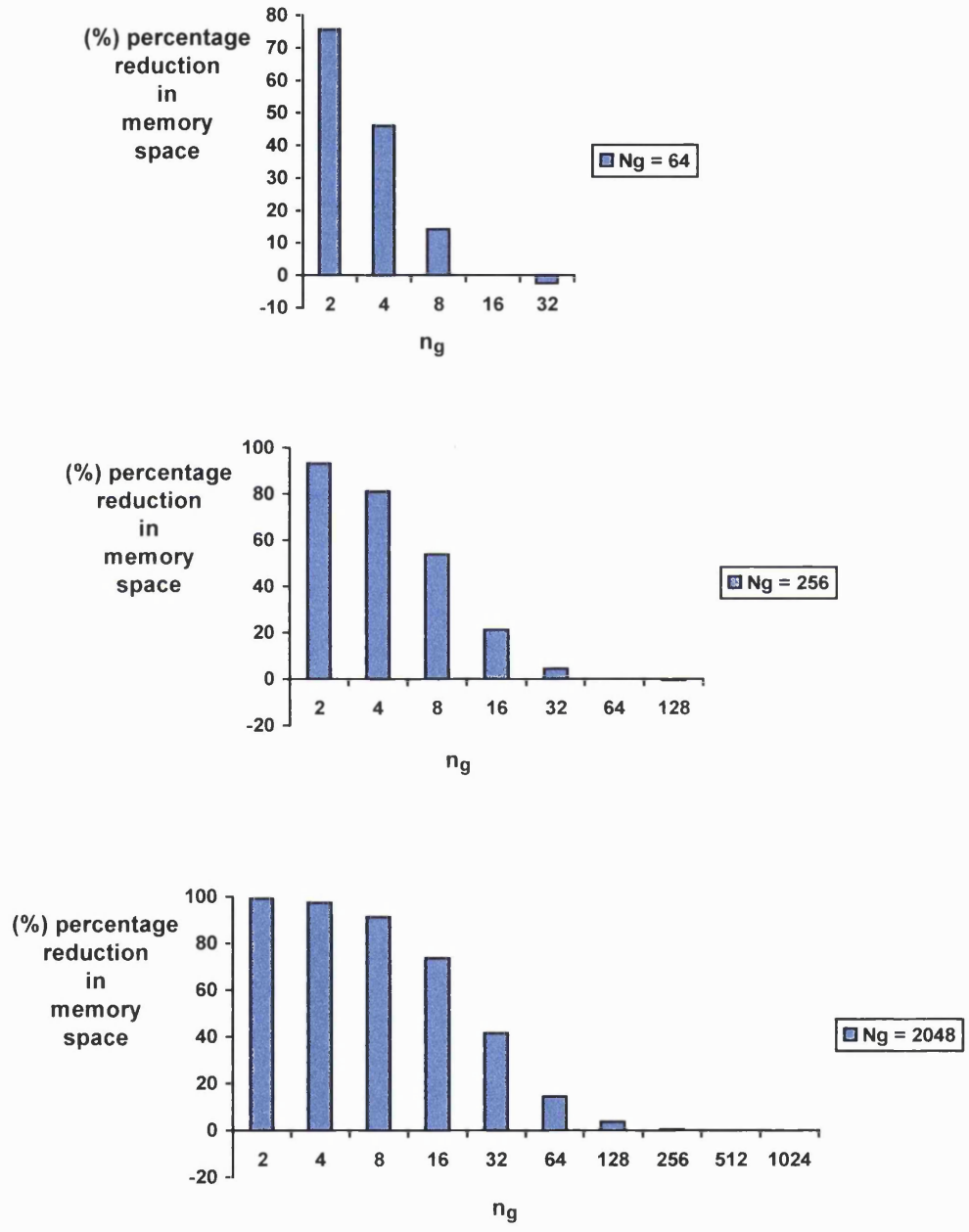
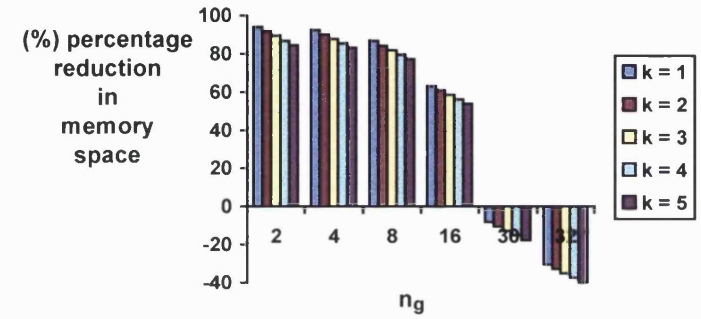


Figure 9.3: Relative percentage reduction in memory space for the plain co-occurrence trees; full-dynamic version vs. semi-dynamic version.

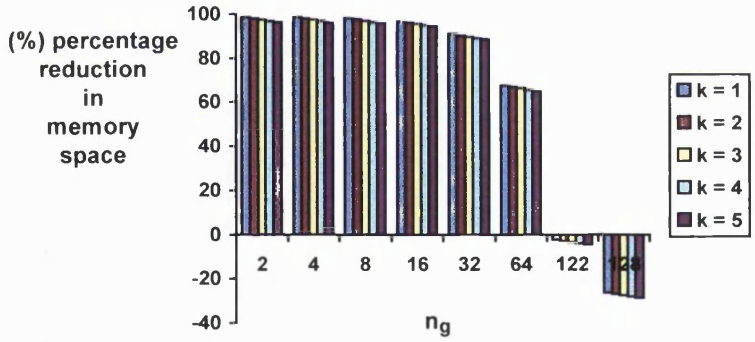
RELATIVE PERCENTAGE REDUCTION IN MEMORY SPACE FOR THE ENHANCED CO-OCCURRENCE TREES (STATIC RULE)

ng: number of grey levels in the analysed region.

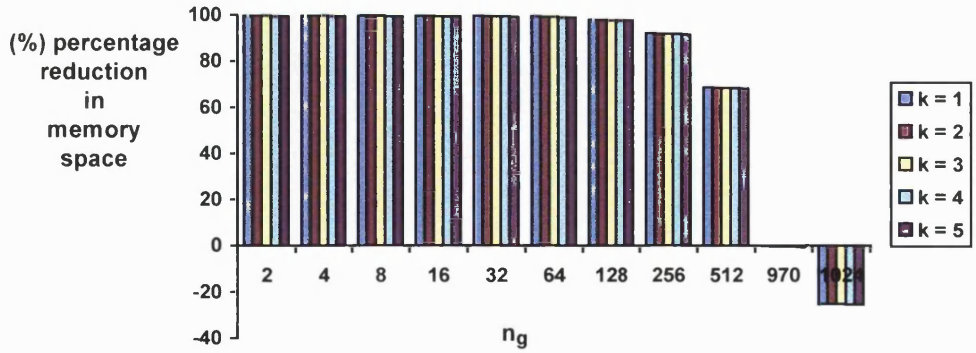
k: length of the probability lists.



(a)



(b)

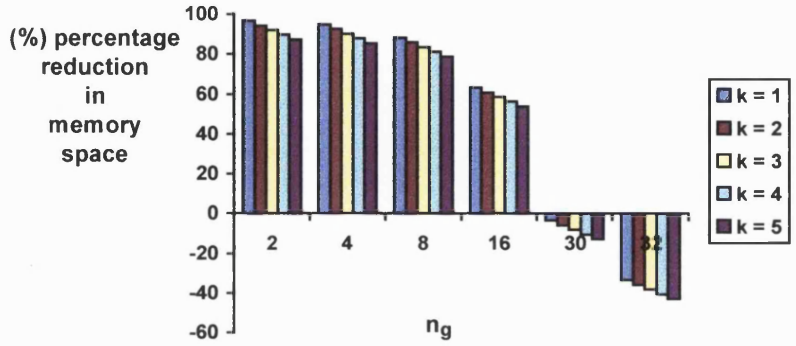


(c)

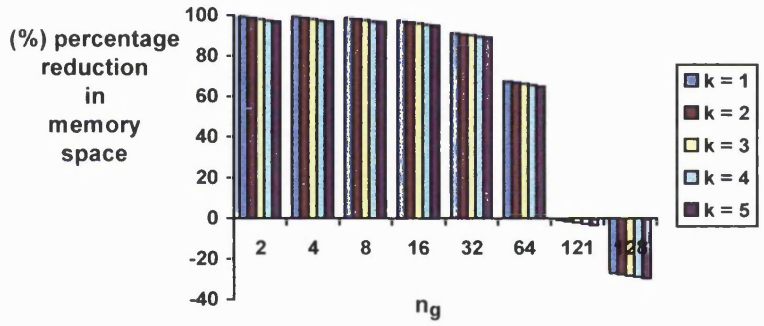
Figure 9.4: Relative percentage reduction in memory space for the enhanced co-occurrence trees (static rule); semi-dynamic version vs. co-occurrence matrix (a) Ng=64, (b) Ng=256, (c) Ng=2048.

ng: number of grey levels in the analysed region.

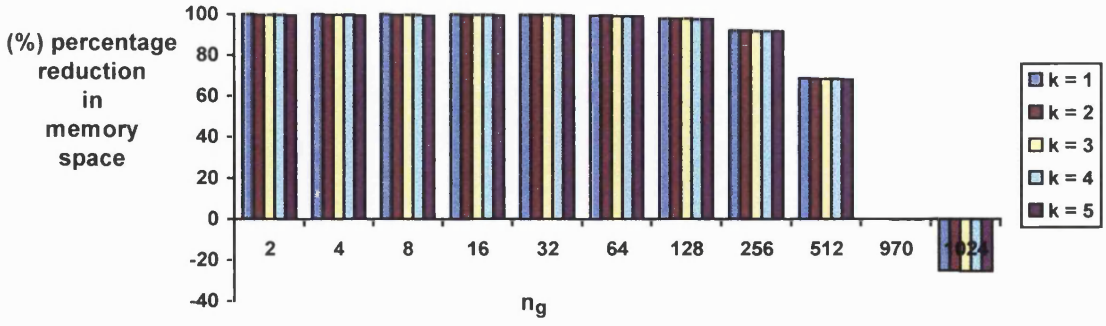
k: length of the probability lists.



(a)



(b)

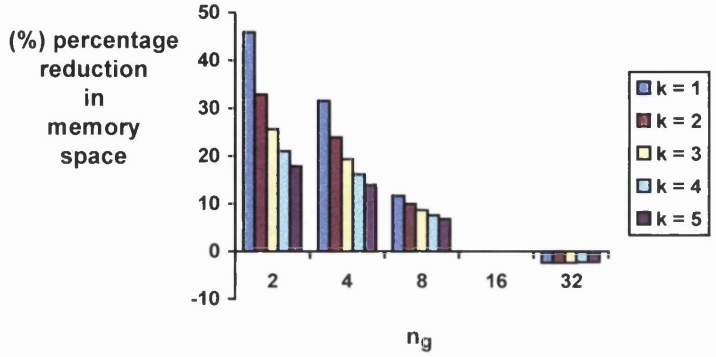


(c)

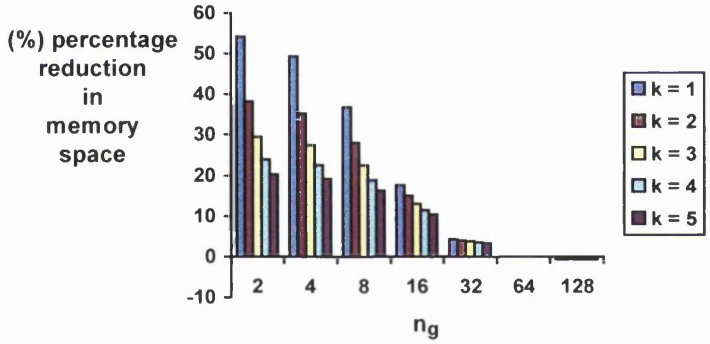
Figure 9.5: Relative percentage reduction in memory space for the enhanced co-occurrence trees (static rule); full-dynamic version vs. co-occurrence matrix (a) Ng=64, (b) Ng=256, (c) Ng=2048.

ng: number of grey levels in the analysed region.

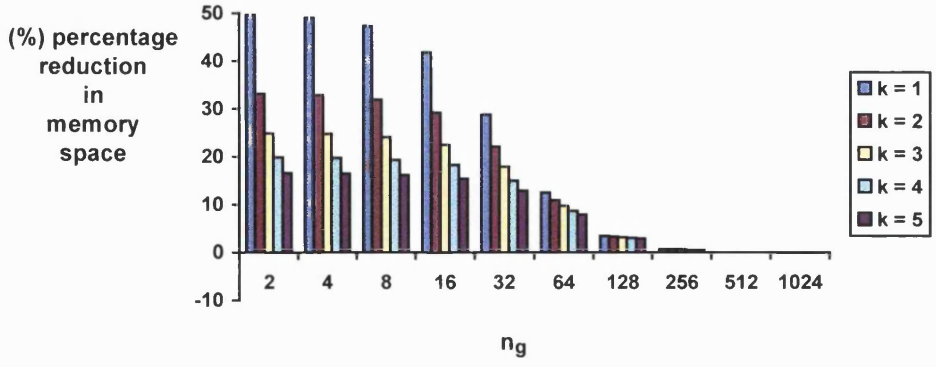
k: length of the probability lists.



(a)



(b)



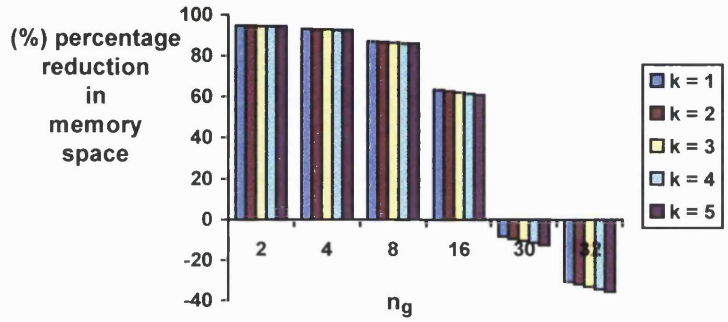
(c)

Figure 9.6: Relative percentage reduction in memory space for the enhanced co-occurrence trees (static rule); full-dynamic version vs. semi-dynamic version (a) Ng=64, (b) Ng=256, (c) Ng=2048.

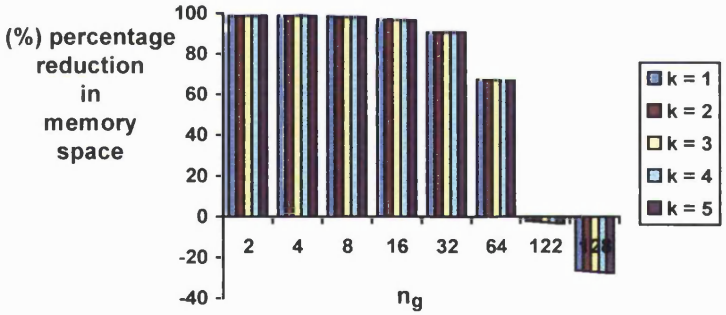
RELATIVE PERCENTAGE REDUCTION IN MEMORY SPACE FOR THE ENHANCED CO-OCCURRENCE TREES (MOVE TO FRONT RULE & COUNTER RULE).

ng: number of grey levels in the analysed region.

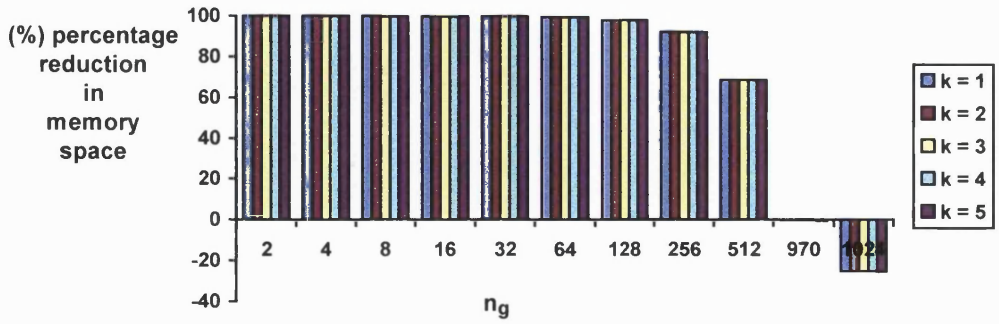
k: length of the probability lists.



(a)



(b)

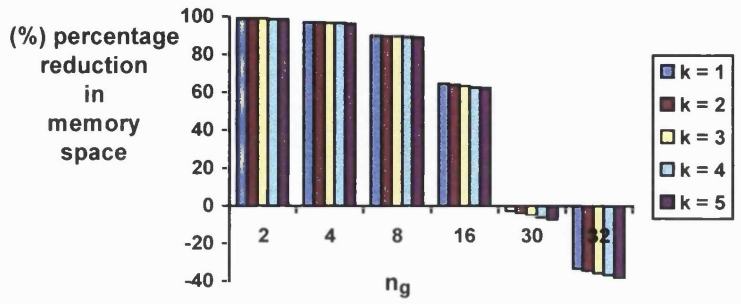


(c)

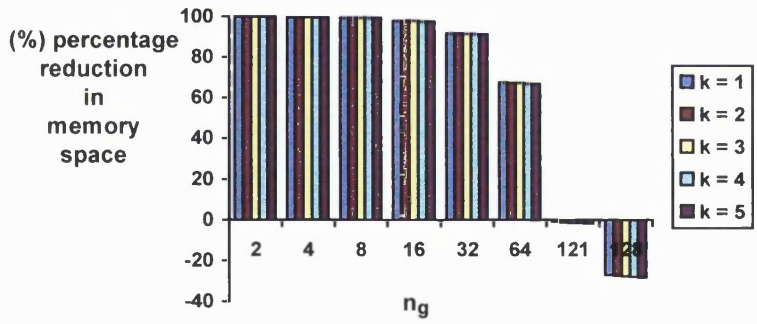
Figure 9.7: Relative percentage reduction in memory space for the enhanced co-occurrence trees (move to front and counter rule); semi-dynamic version vs. co-occurrence matrix (a) Ng=64, (b) Ng=256, (c) Ng=2048.

ng: number of grey levels in the analysed region.

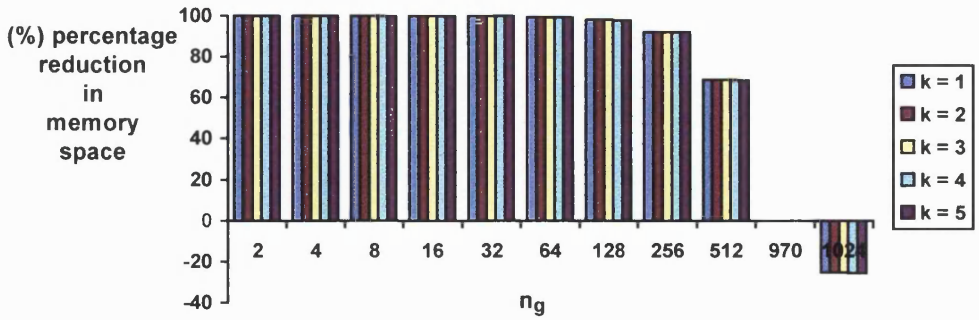
k: length of the probability lists.



(a)



(b)

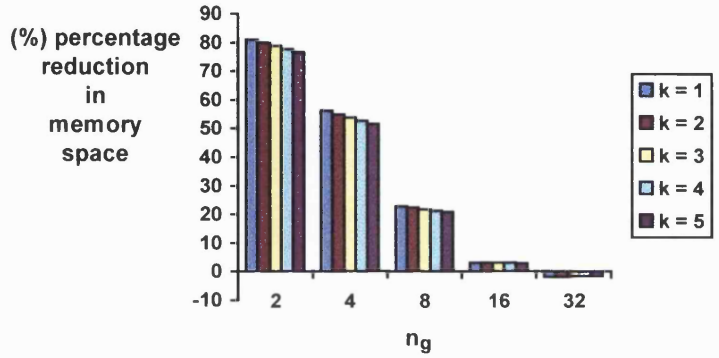


(c)

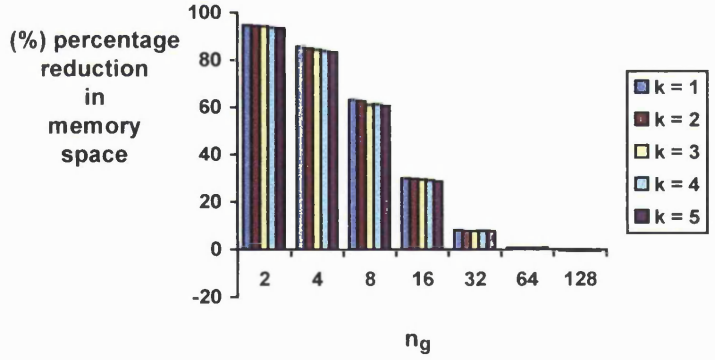
Figure 9.8: Relative percentage reduction in memory space for the enhanced co-occurrence trees (move to front and counter rule); full-dynamic version vs. co-occurrence matrix (a) Ng=64, (b) Ng=256, (c) Ng=2048.

ng: number of grey levels in the analysed region.

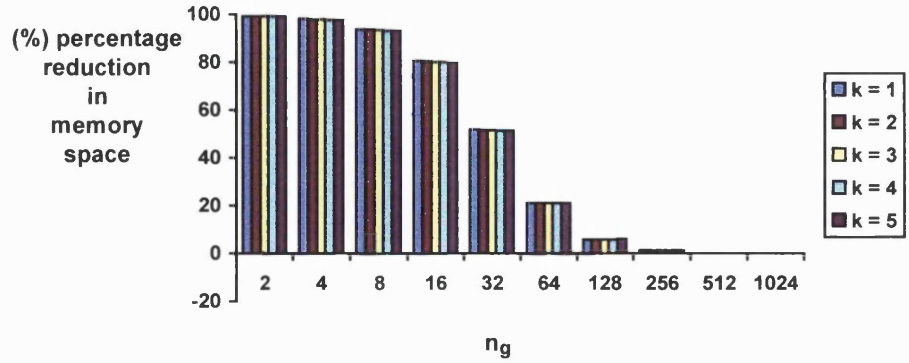
k: length of the probability lists.



(a)



(b)



(c)

Figure 9.9: Relative percentage reduction in memory space for the enhanced co-occurrence trees (move to front and counter rule); full-dynamic version vs. semi-dynamic version (a) Ng=64, (b) Ng=256, (c) Ng=2048.

RELATIVE PERCENTAGE REDUCTION IN MEMORY SPACE FOR THE SELF-ADJUSTING CO-OCCURRENCE TREES

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

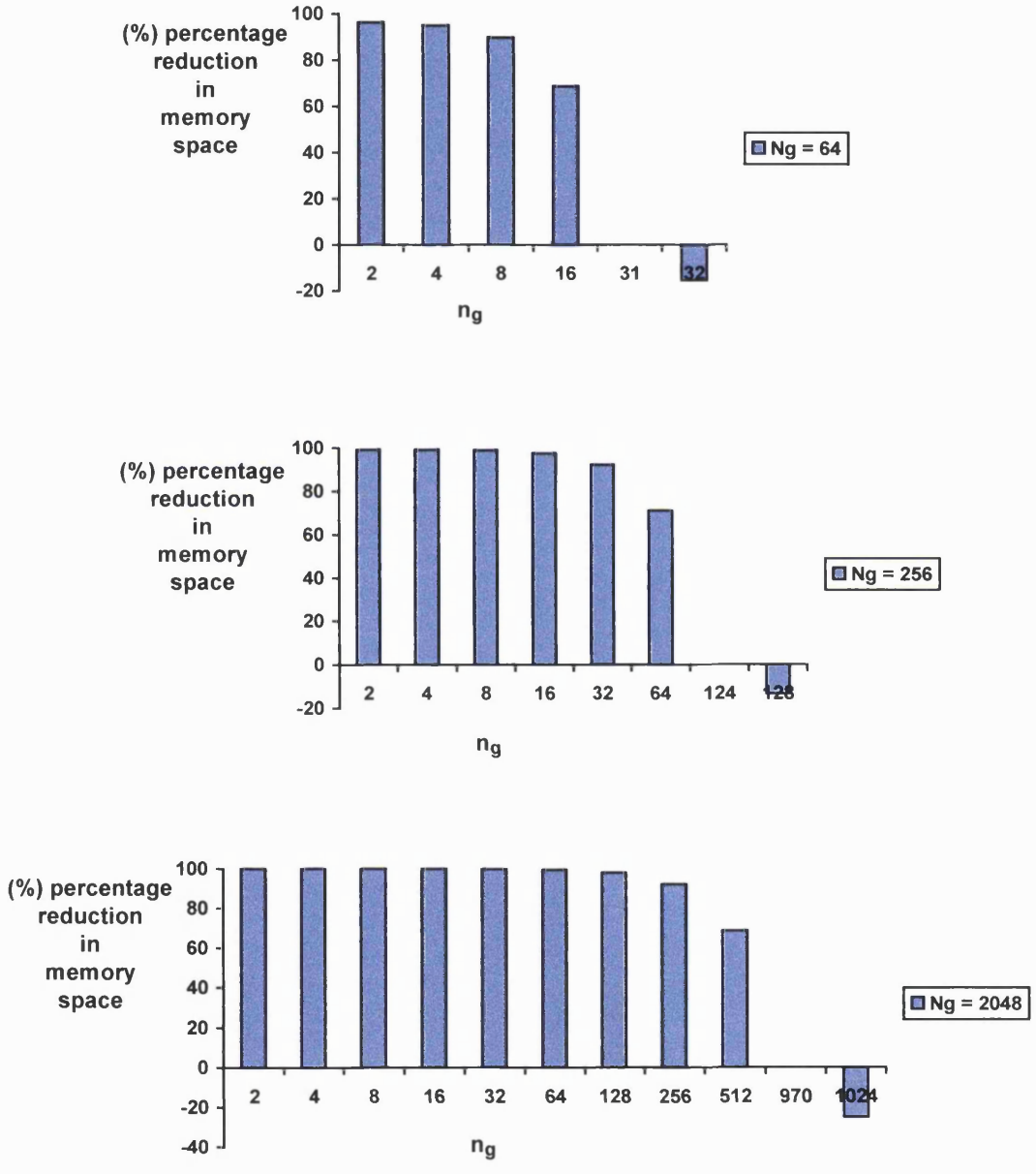


Figure 9.10: Relative percentage reduction in memory space for the self-adjusting co-occurrence trees; semi-dynamic version vs. co-occurrence matrix.

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

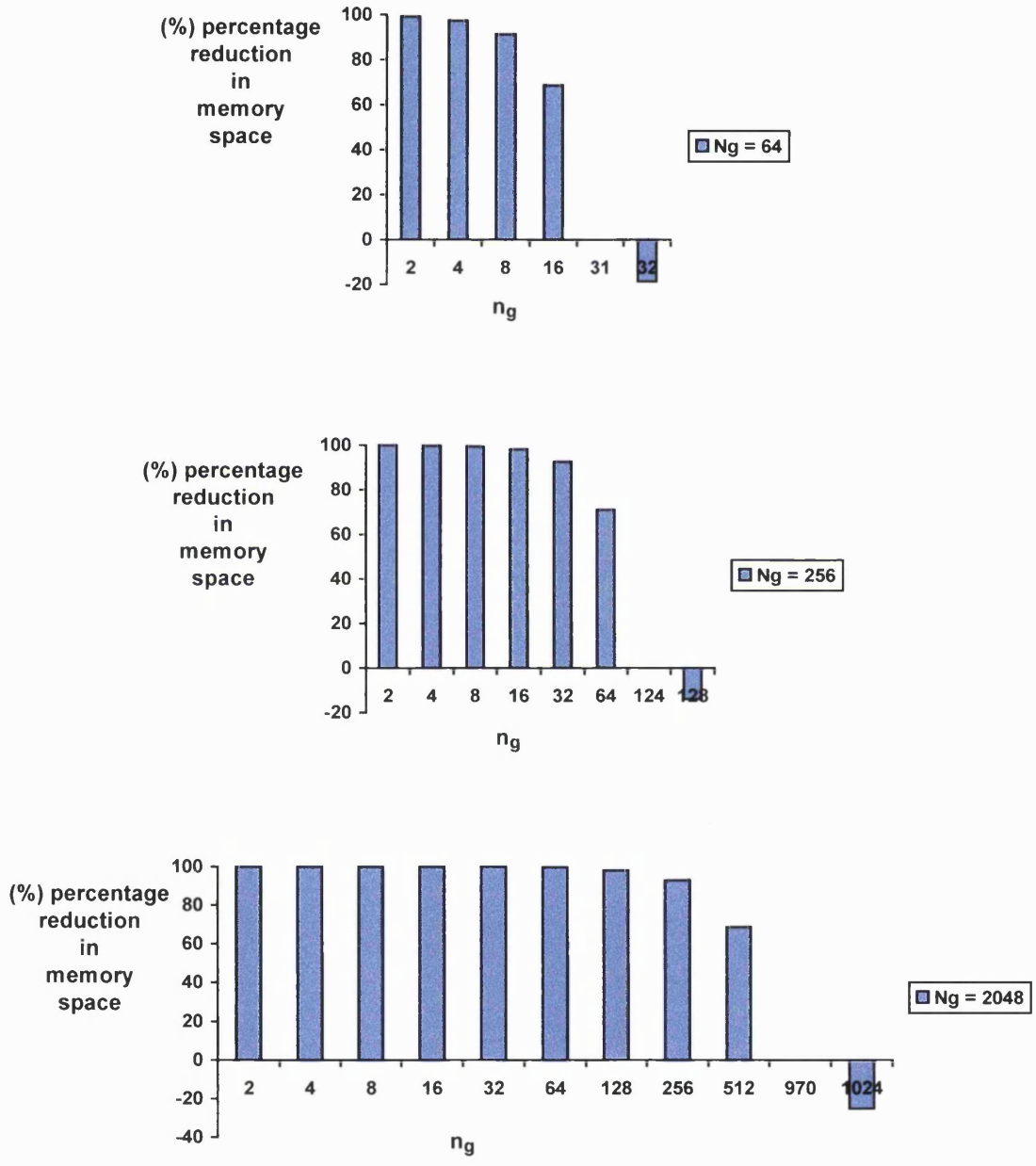


Figure 9.11: Relative percentage reduction in memory space for the self-adjusting co-occurrence trees; full-dynamic version vs. co-occurrence matrix.

Ng: number of grey levels in the image.

ng: number of grey levels in the analysed region.

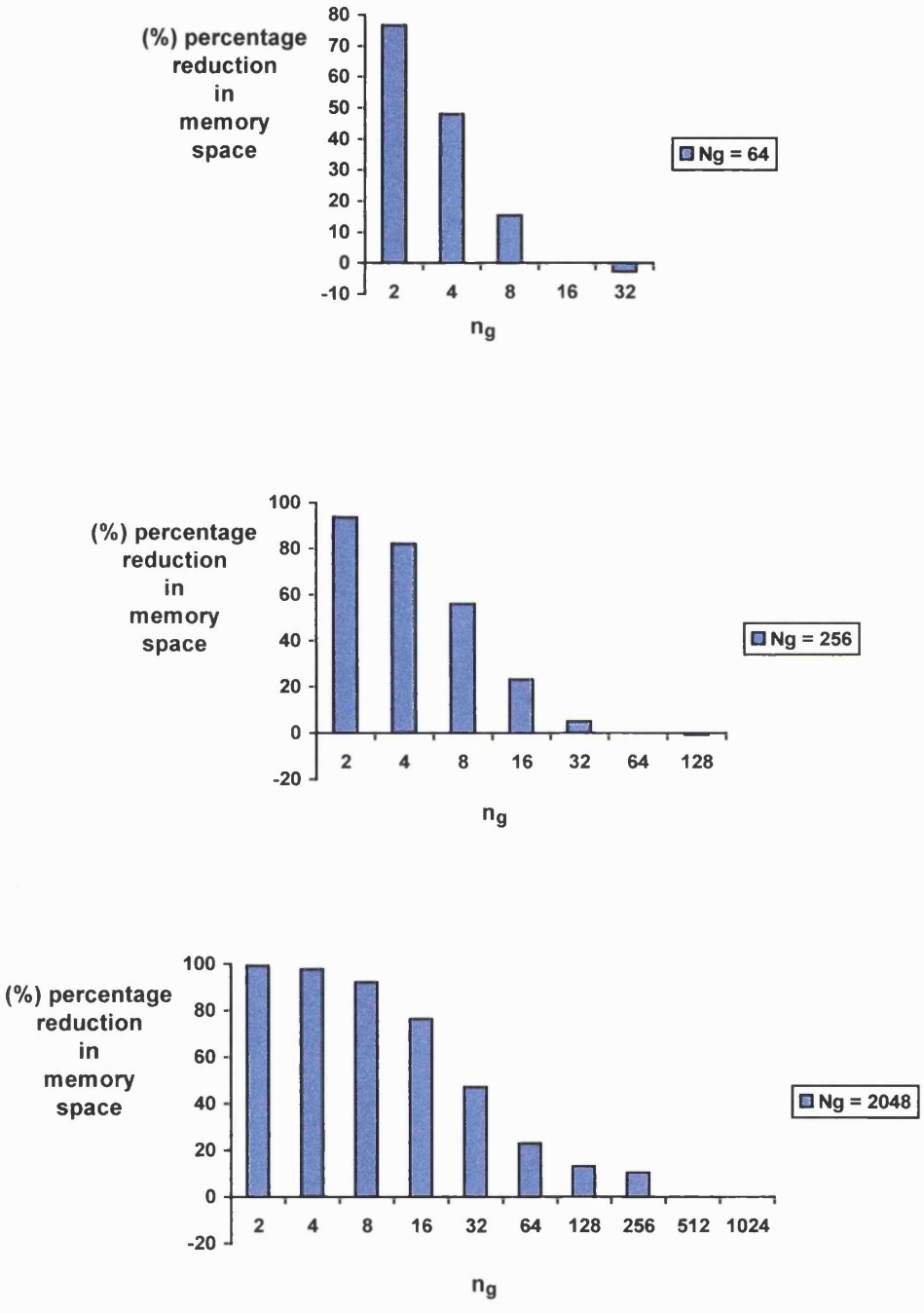
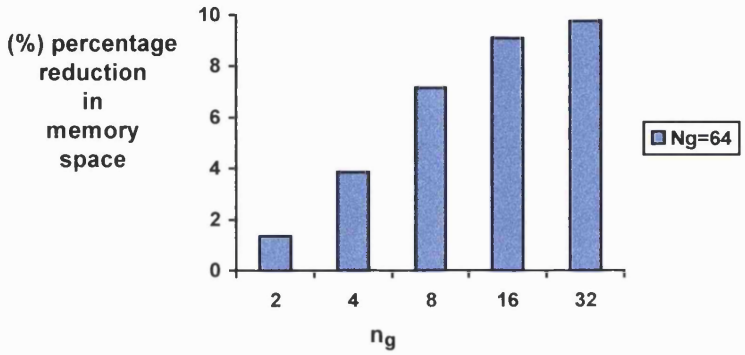
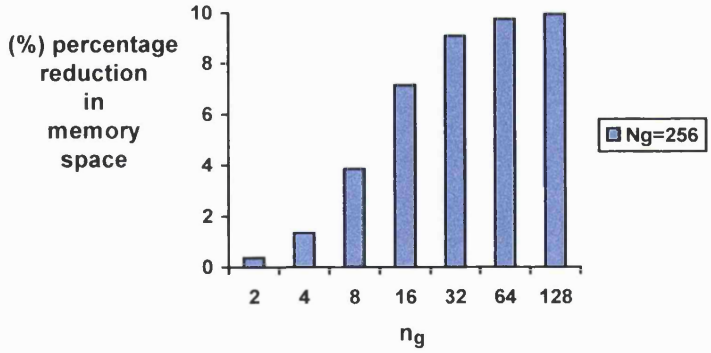


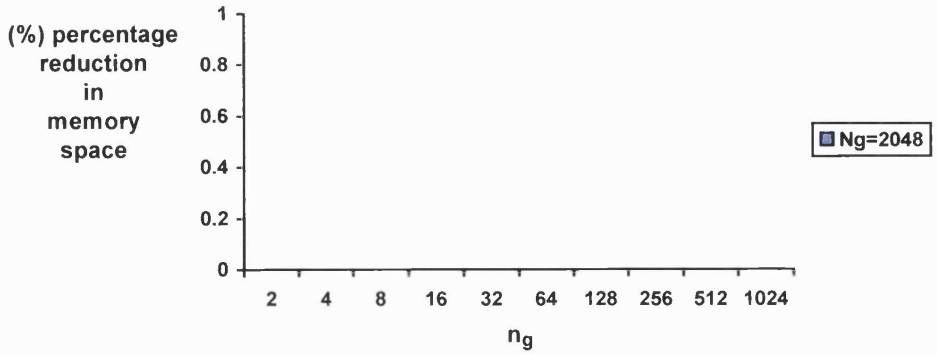
Figure 9.12: Relative percentage reduction in memory space for the self-adjusting co-occurrence trees; full-dynamic version vs. semi-dynamic version.

Ng: number of grey levels in the image.

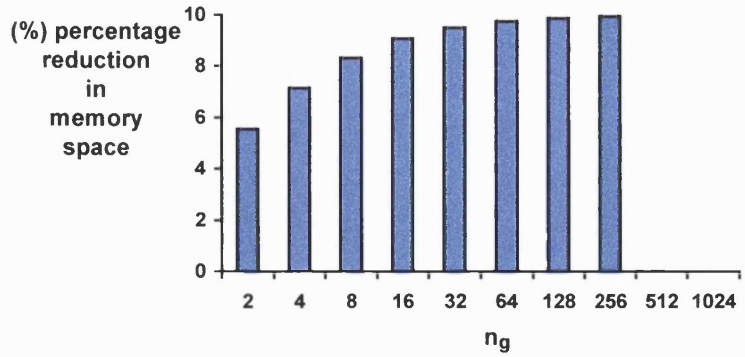
ng: number of grey levels in the analysed region.







(a)



(b)

Figure 9.13: Relative percentage reduction in memory space using the self-adjusting co-occurrence trees instead of the plain co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

9.2 The environment for the development and execution of the algorithms presented for the SGLDM

In this section, we present in greater detail the environment that was used for the development and execution of the texture analysis algorithms. Some of the programs were initially developed on a PC having an Intel 80486 microprocessor that was running at the speed of 66 MHz. Also, a large part of the code was written on a UNIX Sun SPARC workstation, in order us to take advantage of the much more flexible debugging tools of the UNIX environment. That was necessary especially for the implementation of the BB-tree (the basic building block of the plain and enhanced co-occurrence trees) as well as the full-dynamic version of the proposed approaches, which needed the development and implementation of sophisticated algorithms and programming techniques.

All programs were written in the C++ language. One of the main features of this language is portability. Thus, the code was easily transferred to an Intel Pentium Dell P100t personal computer. This was the computing platform on which all algorithms were executed and the computational time results were estimated. This computer is the fastest personal computer in my group (Quantitative Medical Imaging group of the Medical Physics Department, UCL) and it was one of the fastest available personal computers at the time the experiments were performed. It runs at the speed of 100 MHz and has a main memory of 40 MB. The choice of a personal computer as the running platform for the developed algorithms was motivated by the fact that such a computer is much cheaper and therefore, more widely available than a UNIX workstation, such as a Sun SPARCstation. Therefore, the time performance of the compared approaches on a personal computer is of great importance.

9.3 Classification of natural and medical images

Before we present the computational time results from the analysis of real image data by employing the approaches presented for the SGLDM, we need to show the benefit of analysing real images in their initial dynamic range, that is, without reducing their grey level resolution during pre-processing. We will present results in the form of the classification accuracy derived from the statistical classification of real textures. Two applications were examined in this study.

The first application involved the classification of natural textures in five categories, namely asphalt, grass, fur, water, and weave (see Figures **9**.14–**9**.18). The images of three of the above categories were acquired from the corresponding photographs of the well known Brodatz album ([17]) using a digital scanner (fur - D93, water - D38, and weave - D17). The fur photograph was taken from the hide of a calf while the weave photograph was taken from a herringbone weave. The images from the rest of the categories (asphalt and grass) were taken from a public database. Each initial image had a size of 512×512 and a dynamic range of 8 bits (256 grey levels).

The second application involved the classification of mammograms (see Section **3**.2). Each image had a size of 256×256 and a dynamic range of 6 bits (64 grey levels). The data set consisted of 64 mammograms which were taken from the database of the Medical Physics Dept., UCL. There were two classes, namely normal and abnormal breast tissue (see Figures **9**.19 and **9**.20). 33 of the analysed mammograms were abnormal while 31 were normal. The abnormal mammograms included the cases of well defined masses and calcifications.

All initial images were pre-processed using the "equal-probability quantizing" algorithm (see Section **1**.4). In that section, we explained why it was necessary to normalise the images prior to texture analysis. Other algorithms which are simpler, such as histogram equalisation ([75]), could have been used for this purpose. There are two reasons why we chose this particular algorithm. First, it has been specifically used in the normalisation of images for texture analysis. Second, this algorithm provides the ability to reduce the number of grey levels in the initial image while normalising it, prior to texture analysis. Therefore, it was

suitable for our applications.

In both experiments, i.e. classification of natural textures and digital X-ray breast images, each image in the initial data set was pre-processed employing this algorithm. The natural texture images were equalised in 256, 64, 32, and 16 grey levels. Thus, 4 data sets were derived. From each image in each data set 64 non-overlapping subimages of size 64×64 were extracted. Each data set, therefore, consisted of 320 labelled samples (64 samples from each of the five aforementioned categories of natural textures). Each sample was labelled according to the category to which the initial image belonged. The mammograms were equalised in 64, 32, and 16 grey levels. Therefore, 3 data sets were derived. All images contained breast parenchyma only. Therefore, we didn't extract any region; instead the whole images were used for texture analysis.

In both applications, the k-nearest neighbour classifier was employed (see Section **1**.3). The classification accuracy was estimated in the form of the percentage of samples that were correctly classified. That is, if N was the total number of samples in one application and M was the total number of samples that were correctly classified according to their class label, then the accuracy was estimated as $\frac{M}{N} \cdot 100\%$.

In order to estimate this accuracy we used the *leave-one-out* method (see Section **1**.3). This method has been employed in the classification of natural textures ([29], [39], [40], [100], [117], [177]), mammograms ([20]), and digital bone radiographs ([142]). The Euclidean distance was employed to find the k nearest samples to the test sample examined each time. Let $\boldsymbol{f_i}$ be the feature vector of sample x_i and $\boldsymbol{f_j}$ the corresponding vector of sample x_j. Let also the feature vectors have a dimension $d \times 1$. Then, the Euclidean distance between them is defined as

$$d(\boldsymbol{f_i}, \boldsymbol{f_j}) \triangleq \|\boldsymbol{f_i} - \boldsymbol{f_j}\| = \sqrt{(\boldsymbol{f_i} - \boldsymbol{f_j})^T \cdot (\boldsymbol{f_i} - \boldsymbol{f_j})} =$$

$$+\sqrt{\sum_{l=1}^{d} (f_i[l] - f_j[l])^2} \qquad (\mathbf{9}.1)$$

where $f[l]$ is the l-th component of vector $\boldsymbol{f}$.

The above similarity measure has been widely used in pattern classification tasks, mostly due to its simplicity. However, when it is used, some kind of normalisation of the computed features must be performed in order to prevent those with large numerical values from dominating the distance computation ([29], [40], [100], [117], [177]). The normalisation followed here was the standardisation of the features by their sample mean and standard deviation ([29], [100], [177]). It was performed using the following equation:

$$f_i'[l] = \frac{f_i[l] - \mu_l}{\sigma_l},\ l = 1, 2, \ldots, d,\ i = 1, \ldots, N \tag{9.2}$$

where $\mu_l = \frac{1}{N} \cdot \sum_{i=1}^{N} f_i[l]$, $\sigma_l = \left[\frac{1}{N} \cdot \sum_{i=1}^{N} (f_i[l] - \mu_l)^2 \right]^{\frac{1}{2}}$, and N is the total number of samples.

The value of the parameter k in both applications was 3, i.e. a 3-nearest neighbour classifier was actually employed. This classifier has been widely used in the classification of natural textures ([29], [100], [113], [177]). The 3-nearest neighbour classifier is common among the k-nearest neighbour classifiers. One of its advantages is the fact that it can resolve ties when there are two classes. However, in our first application (classification of natural textures) it is clear that ties could happen, since there existed five different categories and the three nearest samples to the test sample might belong to three different categories. Therefore, if the majority rule in the selection of a class for the test sample could not be applied to its three nearest neighbours, the class of the training sample that was nearest to the test sample became its class. The k-nearest neighbour classifier has also been used in the classification of mammograms ([45]), ultrasound liver images ([82]), and digital X-ray chest images ([146]). In addition, in [98] the k-nearest neighbour classifier was considered to be one of the most promising methods for the classification of MR images. In our second application (mammogram classification), since there were only two classes (normal and abnormal breast tissue), $k = 3$ was sufficient to resolve any ties.

All thirteen texture features proposed in ([65]) were employed in both applications, namely angular second moment, contrast, correlation, sum of squares (variance), inverse difference moment, sum average, sum entropy, sum variance, entropy, difference variance, difference entropy, first information measure of cor-

relation, and second information measure of correlation (see Section **4**.1). The semi-dynamic version of the plain co-occurrence trees was used for the computation of the above textural features from the samples. The displacement vectors employed in this study were: $(0,1)$, $(1,1)$, $(1,0)$, $(1,-1)$, corresponding to one pixel distance in the 0°, 45°, 90°, and 135° orientation, respectively. The direction of the displacement vectors was assumed to be unimportant (see Section **4**.1). The values of each textural feature from each sample were averaged over the above four displacement vectors giving a new rotation invariant feature for the above orientations.

Each distinct combination of two and three features was selected for the formation of the feature vector and evaluated. We didn't choose more than three features each time because of the computational burden and the very satisfactory accuracy achieved with combinations of two and three textural features. In total, $\begin{pmatrix} 13 \\ 2 \end{pmatrix} + \begin{pmatrix} 13 \\ 3 \end{pmatrix} = 364$ feature combinations were evaluated in both applications. For each such combination the classification accuracy was estimated using the *leave-one-out* method, as we have already mentioned. The best classification accuracy was reported along with the feature combinations that achieved this result. In Table **9**.1 (a), the best classification accuracy achieved for the natural textures for different grey level resolutions is illustrated. Three textural features dominated the pairs and triplets which comprised the best feature combinations for the four data sets of the first application. These were the sum of squares (variance), the difference variance, and the sum variance. In Table **9**.1 (b), the best accuracy from the classification of the mammograms is shown. Here, the dominant textural features in the best combinations were the difference variance, the sum entropy, and the sum average.

In order to better understand why the reduction in classification accuracy shown in Tables **9**.1 (a) and (b) actually happened, as the grey level resolution of the initial images was being reduced, we will present examples that show the variation of the values of the textural features with the number of grey levels. This variation can give us an idea of the degree to which the initial texture changed, that is, how much of the initial textural detail was lost, as the

dynamic range	classification accuracy
256	99.06 %
64	95.00 %
32	94.69 %
16	94.38 %

(a) Natural textures.

dynamic range	classification accuracy
64	92.19 %
32	82.81 %
16	79.69 %

(b) Mammograms.

Table **9**.1: Classification accuracy results from the performed experiments.

grey level resolution became smaller.

The percentage of the relative change in the average value of each textural feature (averaged over the four displacement vectors mentioned in the previous paragraph) with respect to its average value in the initial image (initial grey level resolution) was computed for each of the reduced grey level resolutions employed in the classification. A positive percentage represents a reduction in the corresponding value while a negative percentage represents an increase. This percentage is illustrated for the aforementioned textural features (see the previous paragraph), namely the sum of squares, the sum average, the sum entropy, the sum variance, and the difference variance.

In Figures **9**.26 (a)–(e), the corresponding percentages from one asphalt, one grass, one fur, one water, and one weave image are shown for the three reduced dynamic ranges used in the classification (64, 32, and 16 grey levels). In these illustrations, "sos" stands for the sum of squares feature, "sa" for the sum average, "se" for the sum entropy, "sv" for the sum variance, and "dv" for the difference

variance. The percentages were estimated from the 64×64 regions shown in Figures **9**.14–**9**.18. Similar changes resulted from the analysis of other regions in the above images as well as other images of the same category (e.g. other fur images). Finally, in Figure **9**.27 the corresponding percentages from the analysis of one normal and one abnormal mammogram are shown for 32 and 16 grey levels (the reduced dynamic ranges used in the classification). The analysed images are illustrated in Figures **9**.19 and **9**.20. Again, similar results were derived from the analysis of other images belonging to the same category.

Figure **9**.14: Asphalt image.

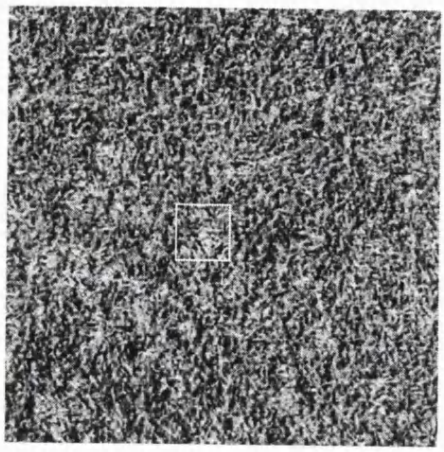


Figure **9**.15: Grass image.

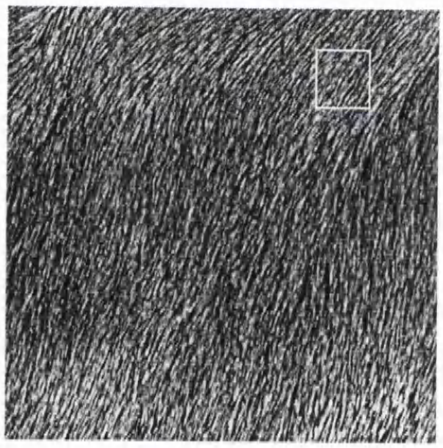


Figure **9**.16: Fur image.

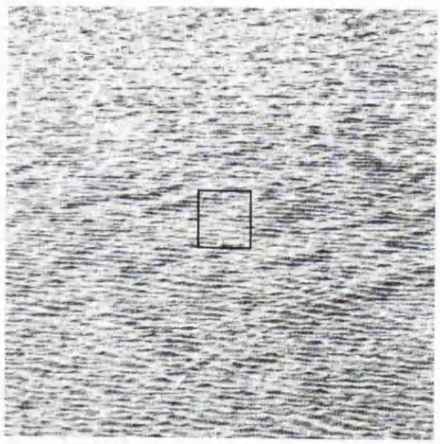


Figure **9**.17: Water image.

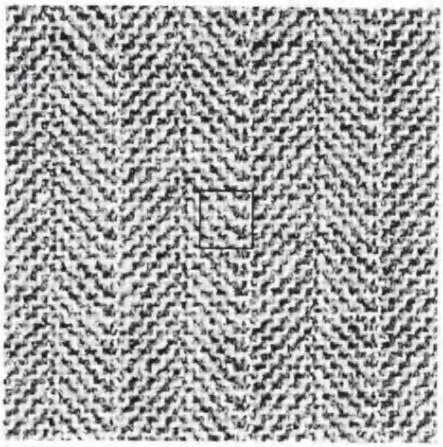


Figure **9**.18: Weave image.

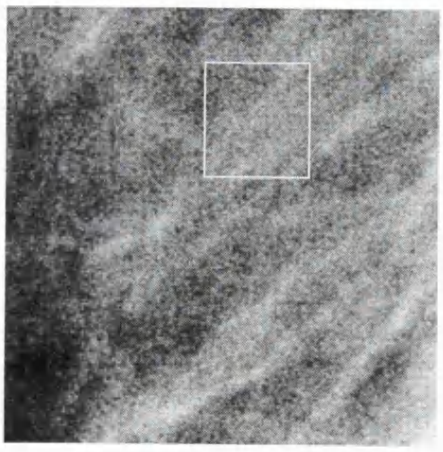


Figure **9**.19: Normal mammogram.

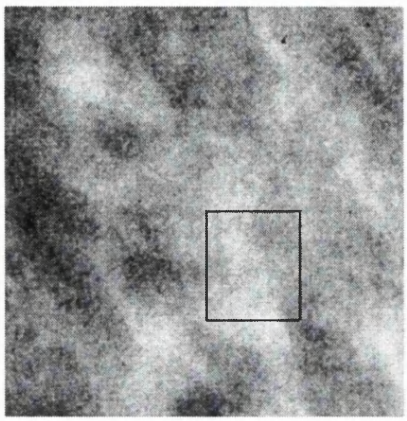


Figure **9**.20: Abnormal mammogram.

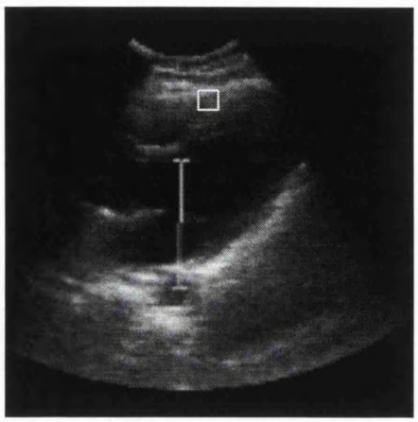


Figure **9**.21: Ultrasound heart image.

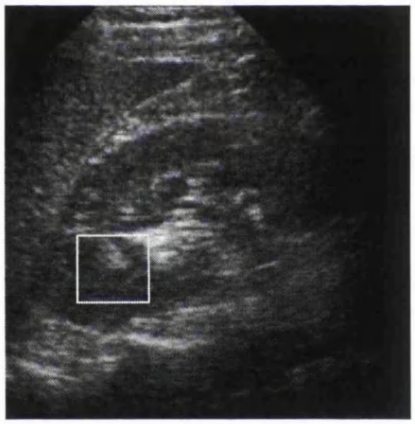


Figure **9**.22: Ultrasound kidney image.

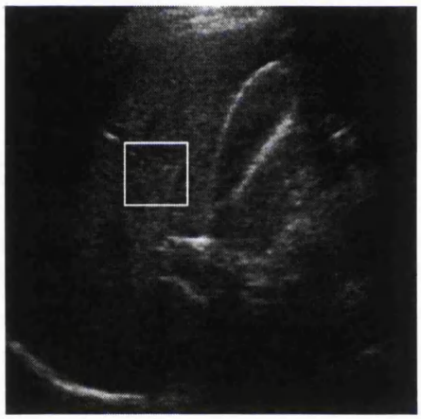


Figure **9**.23: Ultrasound liver image.

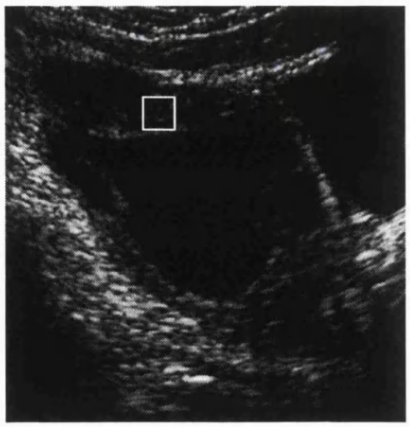


Figure **9**.24: Ultrasound ovaries image.

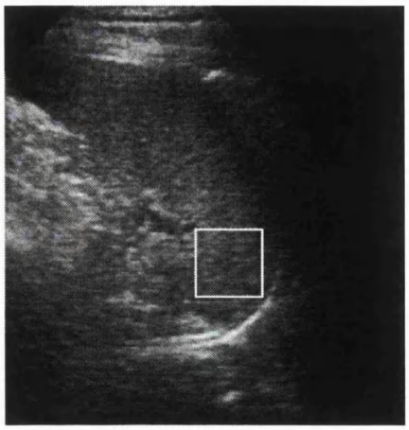
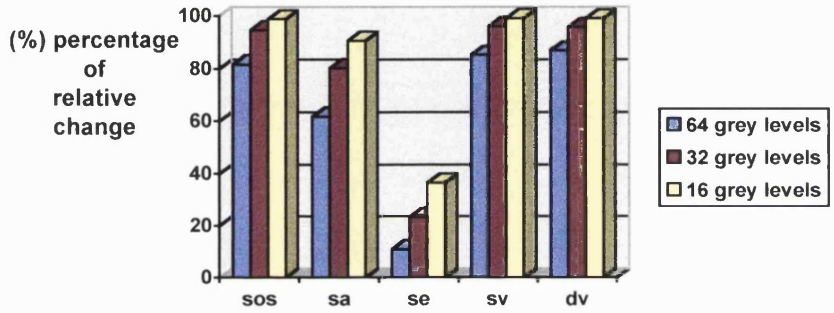


Figure **9**.25: Ultrasound spleen image.

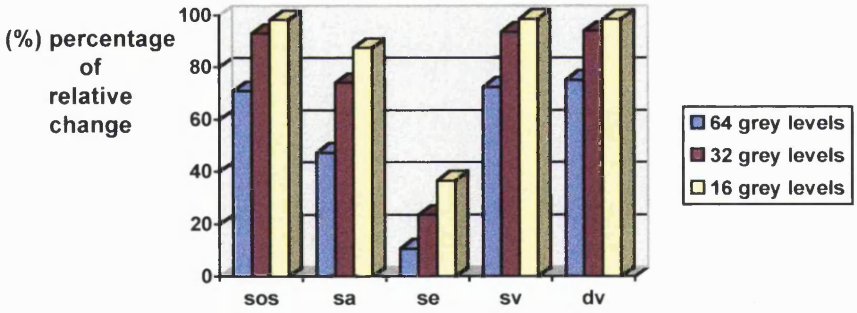
sos: sum of squares. **sa**: sum average.

se: sum entropy. **sv**: sum variance.

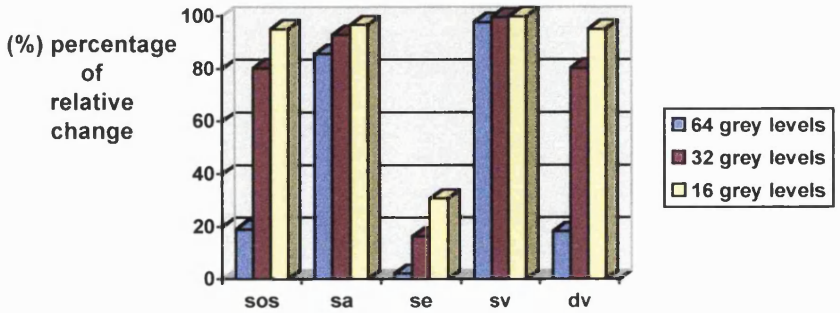
dv: difference variance.



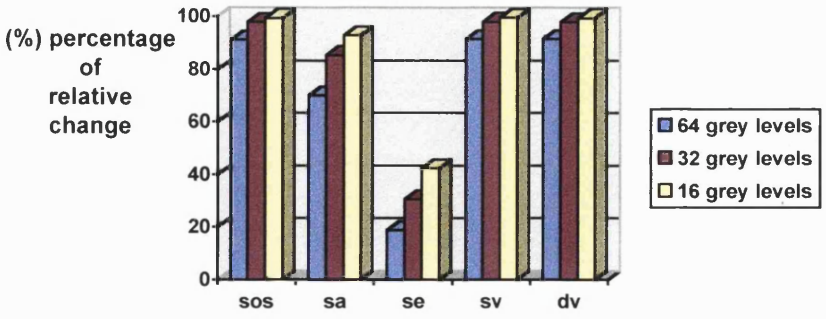
(a)



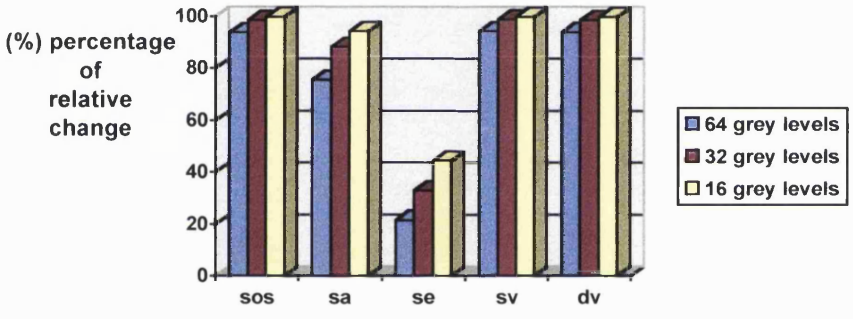
(b)



(c)



(d)



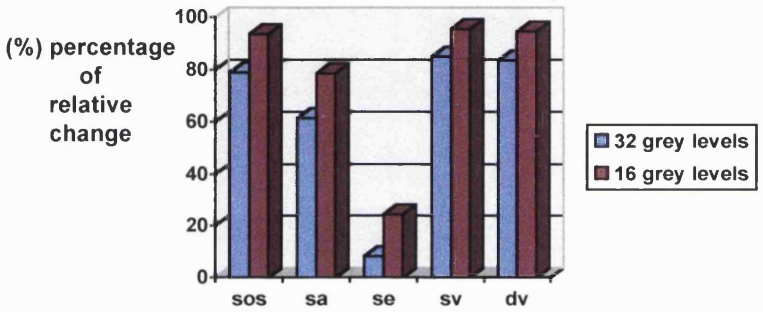
(e)

Figure 9.26: Examples of the percentage of the relative change in the values of the textural features of natural textures for various reduced grey level resolutions (a) asphalt image, (b) grass image, (c) fur image, (d) water image, (e) weave image.

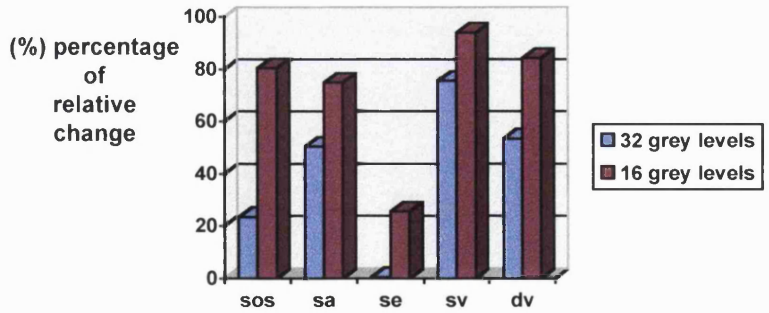
sos: sum of squares. **sa**: sum average.

se: sum entropy. **sv**: sum variance.

dv: difference variance.



(a)



(b)

Figure 9.27: Examples of the percentage of the relative change in the values of the textural features of mammograms for various reduced grey level resolutions (a) normal mammogram, (b) abnormal mammogram.

9.4 Computational time results

In the following, we present computational time results from the analysis of real image data for each of the compared approaches. The purpose is twofold. First, we would like to show that the approaches proposed in this thesis are more efficient than the co-occurrence matrix, for most of the dynamic ranges encountered in practice. At the same time, we will show that the proposed approaches make feasible the analysis of images using the SGLDM in their initial dynamic range. Preliminary results from the plain co-occurrence trees were presented in [154]. The computational time refers to the time required by the feature extraction stage only (see Section **1**.3). It was measured from the point where the texture feature extraction algorithm started building the appropriate data structure (co-occurrence matrix or co-occurrence trees), until all textural features were computed.

All analysed images were first pre-processed using the "equal-probability quantizing" algorithm (see Section **9**.3). However, they retained their initial grey level resolution (no reduction in the number of grey levels was performed). Five image categories were employed in texture analysis. The first category consisted of images containing natural textures. The source of these images was mentioned in Section **9**.3. The other four categories consisted of medical images of various modalities, namely digital X-ray, ultrasound, CT, and MR. All images belonging to these categories were taken from the database of the Medical Physics Department, UCL. In some cases, a subimage of the initial image was first extracted in order to avoid including artifacts in the pre-processing and subsequent analysis of the images.

All analysed region parameters (number, size, and location on the image) were chosen under the guidance of an experienced radiologist and with the aim to cover as much of the clinically useful texture in the image, such as liver parenchyma and trabecular bone tissue, as possible. Structures that were of no interest and also could affect the values of the textural features, such as large vessels and bones, were avoided being included in the selected regions as much as possible. Moreover, the number of the selected regions is typical of a short-scale clinical examination.

That is, the total time for analysing all regions extracted from a particular image type (e.g. MR brain image type) in our study is considered to approximate the total time[1] for analysing a medium number of medical images of the same type (same modality, dynamic range, body tissue) in a clinical application, where a few regions (regions of interest) are usually chosen in each image for classification purposes. In addition, the above time is considered to be indicative of the total time needed to segment a number of medical images (again, the time refers to the feature extraction stage). In an image segmentation application, a large number of regions are analysed in each image in order to segment the various areas of homogeneous tissue.

8 displacement vectors were employed for the texture analysis of the selected regions in all five categories, which are among the most widely used in the SGLDM. They corresponded to the four basic orientations, i.e. 0°, 45°, 90°, and 135°. The interpixel distance was chosen to be 1 and 2 pixels. Therefore, the 8 displacement vectors were: $(0,1)$, $(1,0)$, $(1,1)$, $(1,-1)$, $(0,2)$, $(2,0)$, $(2,2)$, and $(2,-2)$. The direction of the displacement vectors was assumed to be unimportant (see Section **4**.1). All thirteen texture features were computed (see Section **4**.1), except from a few cases where the computation of some features needed an excessive amount of time. These were the cases of the data sets containing images of 15 bit dynamic range (the MR human femur images and the MR images of sagittal sections of the thigh; see Section **9**.4.4). The sum entropy, the sum variance, the difference entropy, and the second information measure of correlation did not participate in the computation process for the above image types.

The estimated computational time of the feature extraction stage is presented in various forms. First, the total time for the analysis of all regions in each data set, employing the above 8 displacement vectors, was estimated for each of the approaches presented in this thesis, using the time measurement capabilities of the personal computer. The time resolution was the $\frac{1}{1000}$ of a second (one millisecond). Then, an averaging over time was performed and the average time per region and displacement vector was calculated. This is similar to the amortisation technique

[1]the total time of the feature extraction stage

for estimating the running time of algorithms (see [156]). Both the total time and the estimated average time per region comprise reliable and robust time measures on which comparisons can be based. In a previous paragraph, we discussed the benefits from the employment of the total time measure. The average time per region in the different forms presented is an appropriate tool for comparing the various approaches in the SGLDM and can lead to useful conclusions, as we will see in the next chapter (Chapter **10**).

The total time and the average time per region are shown only for the co-occurrence matrix and the basic approach in this thesis, that is, the semi-dynamic and full-dynamic version of the plain co-occurrence trees. In the relevant figures, "mtx" stands for the co-occurrence matrix, "sd" for the semi-dynamic version, and "fd" for the full-dynamic version. All times were measured in milliseconds. Also, a zero height column indicates a "not able to compute" situation. That means that the corresponding approach was not able to produce results because it led the system to run out of memory or get stuck for an excessive amount of time.

For all the other approaches, namely the semi-dynamic and full-dynamic version of the enhanced co-occurrence trees and the semi-dynamic and full-dynamic version of the self-adjusting co-occurrence trees, the average time per region is shown in the form of the relative percentage reduction with respect to the corresponding time of the corresponding version of the plain co-occurrence trees. That is, if t_A is the average time per region for a specific version of the plain co-occurrence trees and analysed data set and t_B is the corresponding time for the same version of another type of co-occurrence trees, e.g. the enhanced co-occurrence trees, the relative percentage reduction is computed as: $\frac{t_A - t_B}{t_A} \cdot 100\%$. A positive percentage corresponds to a decrease in the average time per region relative to the plain co-occurrence trees, while a negative percentage corresponds to the opposite.

In particular, for the enhanced co-occurrence trees the above percentage was estimated for the first 10 values of the parameter K (length of the probability lists), i.e. $1 \leq K \leq 10$. Results from the three rules presented in this thesis, namely the static rule, the move to front rule, and the counter rule are shown

separately. In addition, in the case of the self-adjusting co-occurrence trees results from the four techniques presented in this thesis, namely move to root (symbolised by "mtr"), top-down splaying (symbolised by "tdsplay"), splaying (symbolised by "splay"), and semi-splaying (symbolised by "smsplay") are shown separately in the relevant figures.

9.4.1 Time results from the analysis of natural textures

As we said, the first category of analysed images consisted of those images containing natural textures. Five data sets belonged to this category, namely asphalt, grass, fur, water, and weave (see also Section **9**.3). As we have already mentioned, all images in this category were 512×512, having a dynamic range of 8 bits (256 grey levels). A 128×128 subimage of each initial image was extracted for pre-processing and subsequent texture analysis.

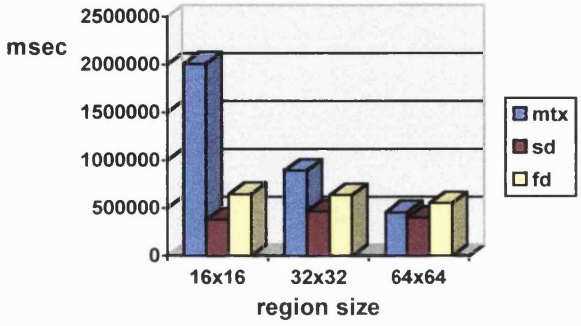
Nonoverlapping regions of dimensions 16×16, 32×32, and 64×64, covering the whole area of each subimage, were extracted and analysed, that is, 64 image regions of dimensions 16×16, 16 regions of dimensions 32×32, and 4 regions of dimensions 64×64. In total, each data set of the first category consisted of 320 regions of dimensions 16×16, 80 regions of dimensions 32×32, and 20 regions of dimensions 64×64. Results are shown for each region size, separately.

Figures **9**.28–**9**.32 show the computational time results for the asphalt data set. Figures **9**.33–**9**.37 show the time results for the grass data set. Figures **9**.38–**9**.42 show the corresponding results for the fur data set. Figures **9**.43–**9**.47 illustrate the time results for the water data set. Finally, Figures **9**.48–**9**.52 illustrate the corresponding time results for the weave data set.

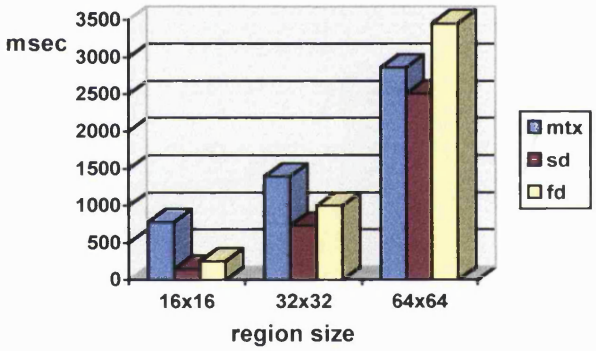
COMPUTATIONAL TIME RESULTS FOR THE ASPHALT IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



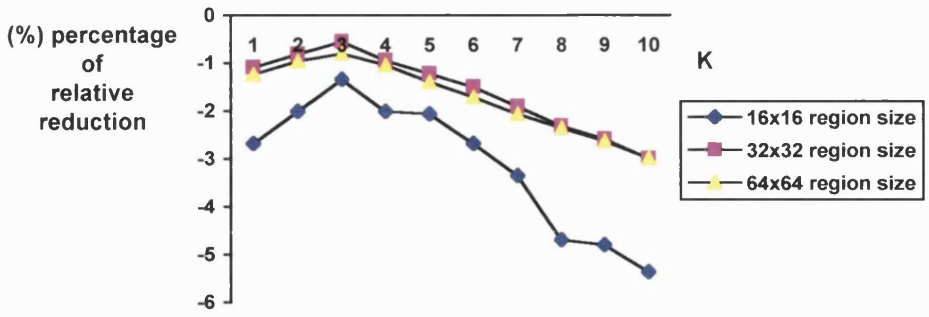
(a)



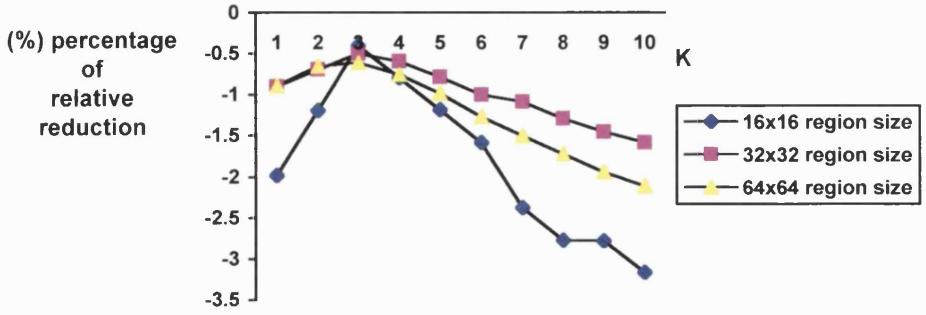
(b)

Figure 9.28: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.

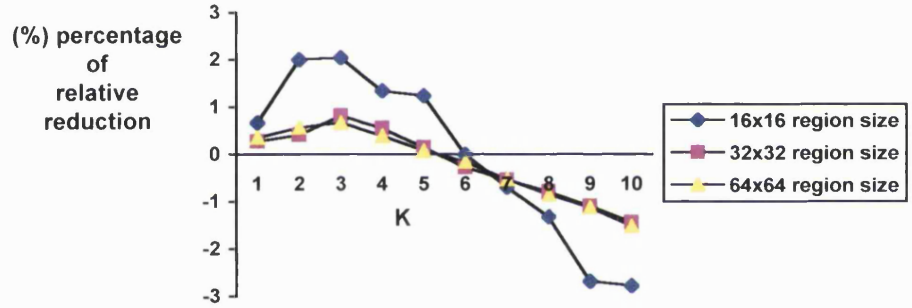


(a)

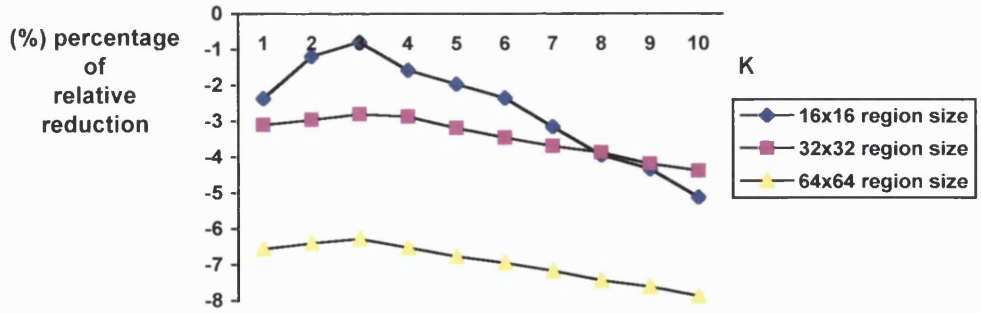


(b)

Figure 9.29: Enhanced co-occurrence trees - static rule (a) semi-dynamic version, (b) full-dynamic version.

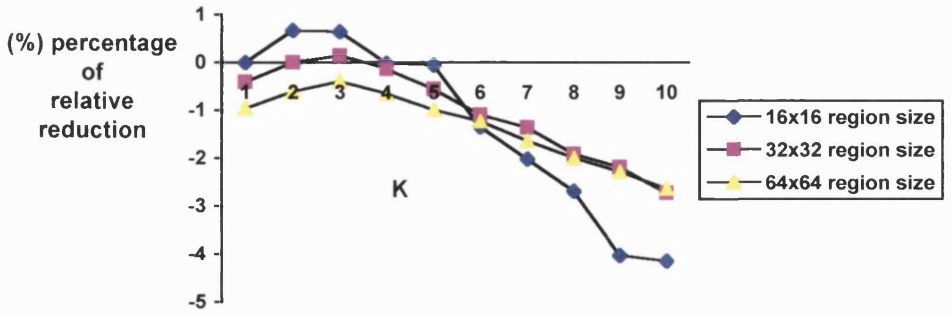


(a)

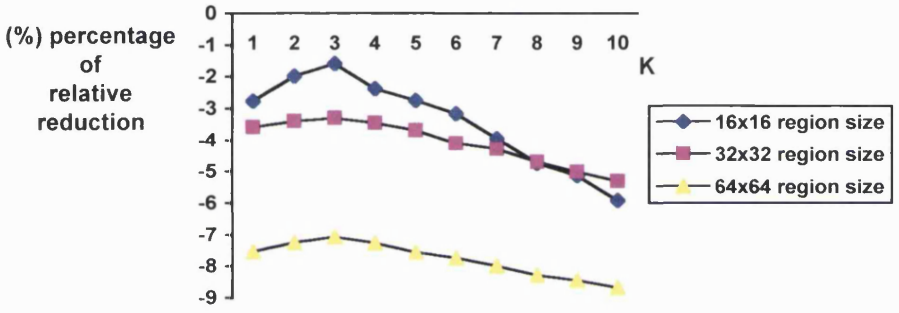


(b)

Figure 9.30: Enhanced co-occurrence trees - move to front rule (a) semi-dynamic version, (b) full-dynamic version.



(a)

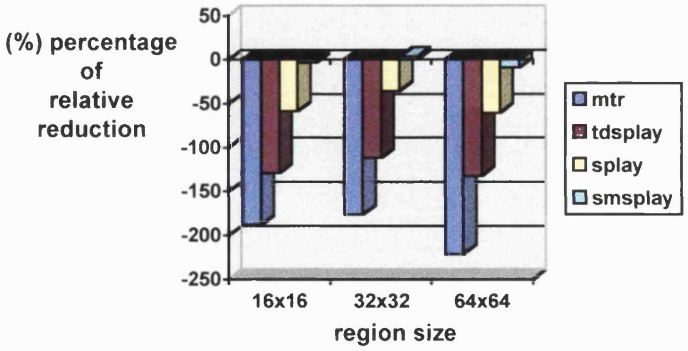


(b)

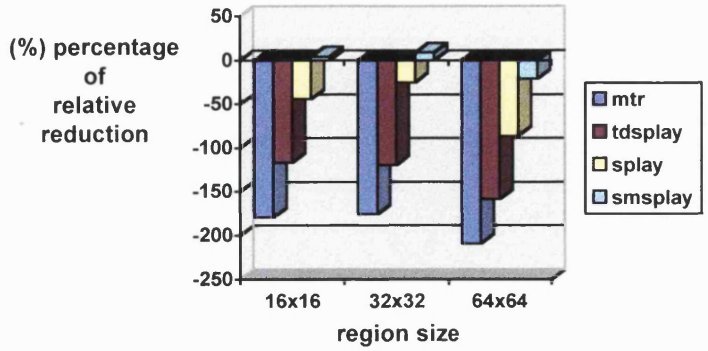
Figure 9.31: Enhanced co-occurrence trees - counter rule
(a) semi-dynamic version, (b) full-dynamic version.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.



(a)



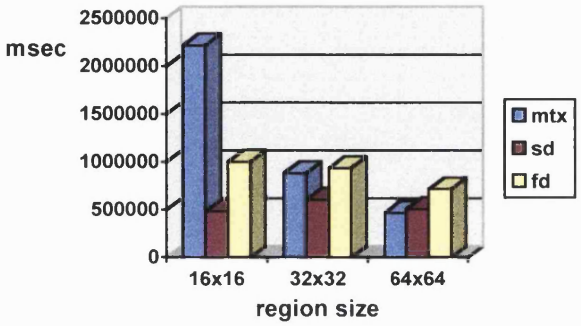
(b)

Figure 9.32: Self-adjusting co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

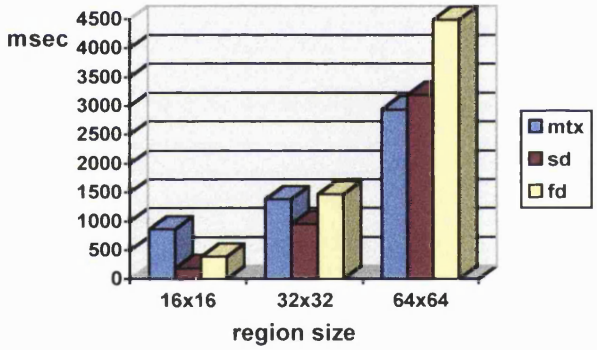
COMPUTATIONAL TIME RESULTS FOR THE GRASS IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



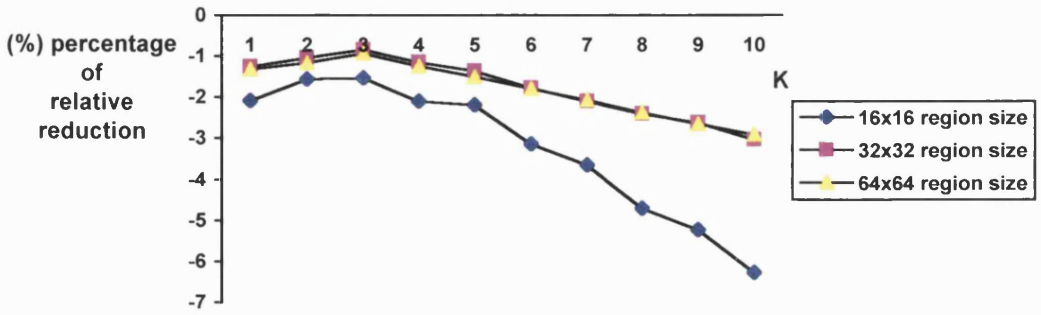
(a)



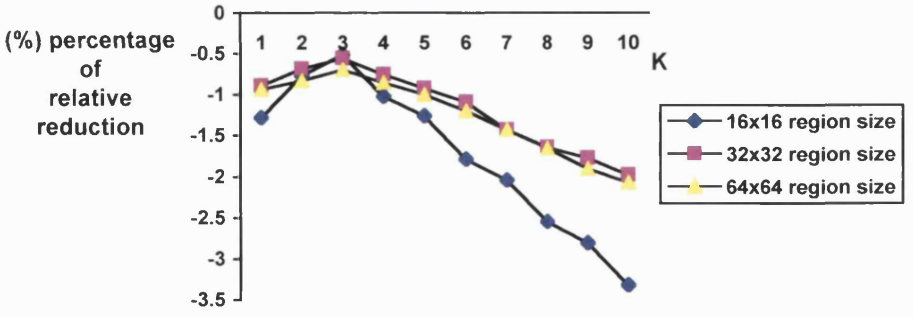
(b)

Figure 9.33: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.

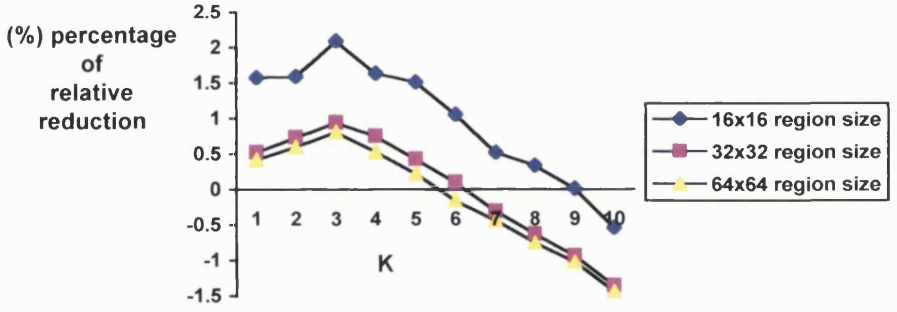


(a)

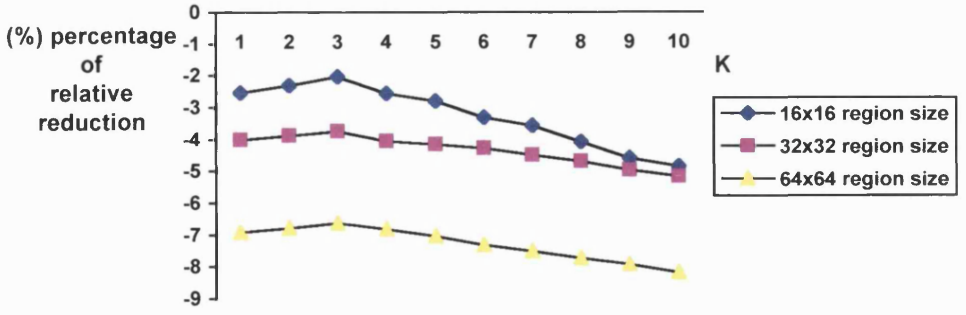


(b)

Figure 9.34: Enhanced co-occurrence trees - static rule (a) semi-dynamic version, (b) full-dynamic version.

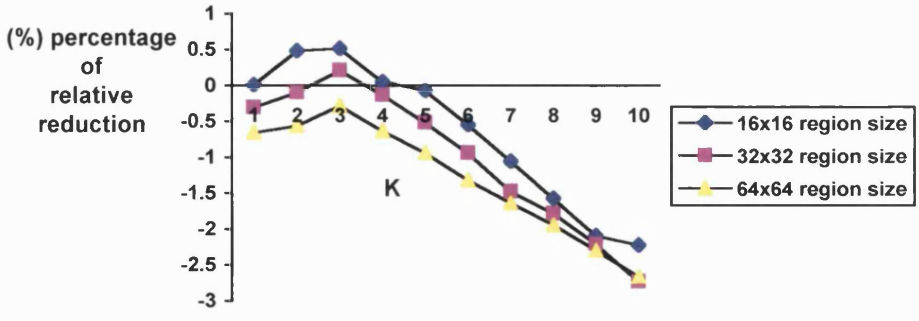


(a)

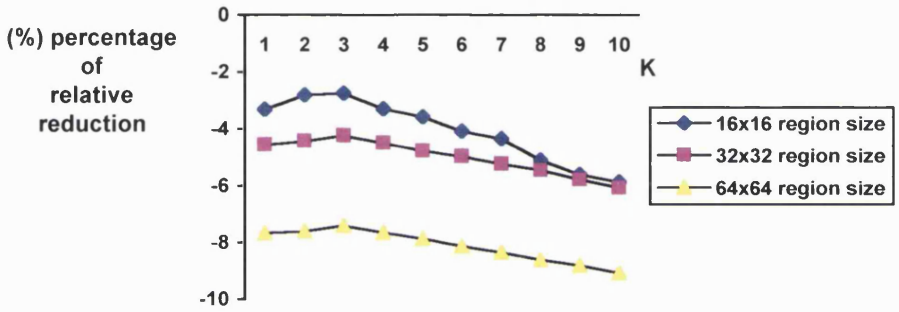


(b)

Figure 9.35: Enhanced co-occurrence trees - move to front rule (a) semi-dynamic version, (b) full-dynamic version.



(a)

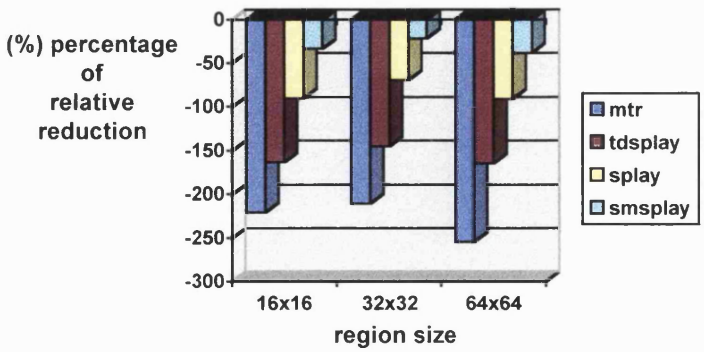


(b)

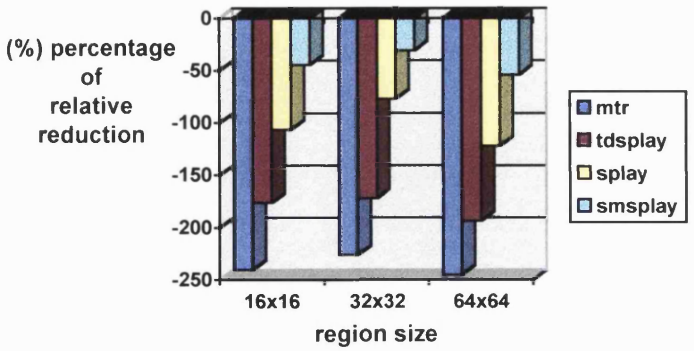
Figure 9.36: Enhanced co-occurrence trees - counter rule
(a) semi-dynamic version, (b) full-dynamic version.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.



(a)



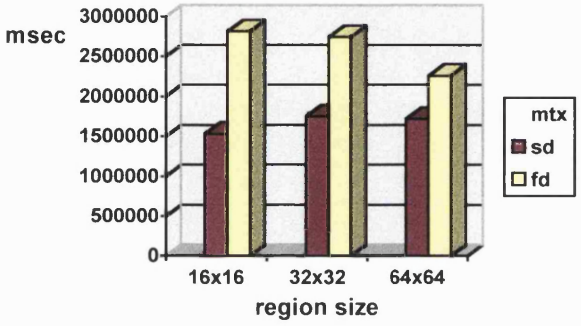
(b)

Figure 9.37: Self-adjusting co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

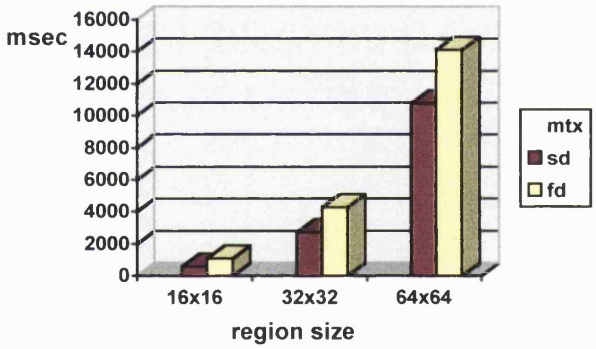
COMPUTATIONAL TIME RESULTS FOR THE FUR IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



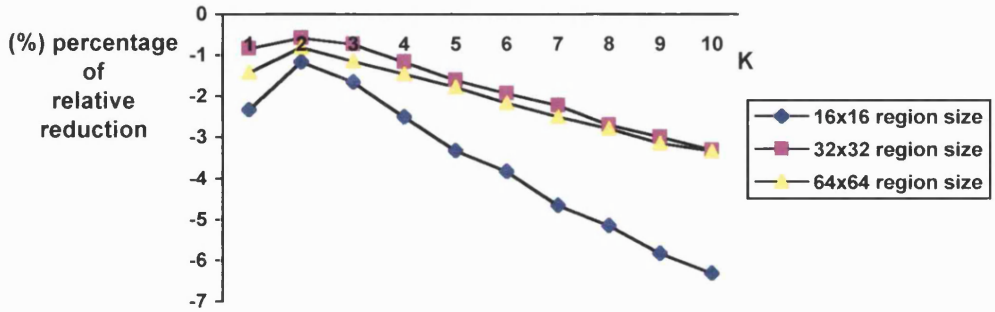
(a)



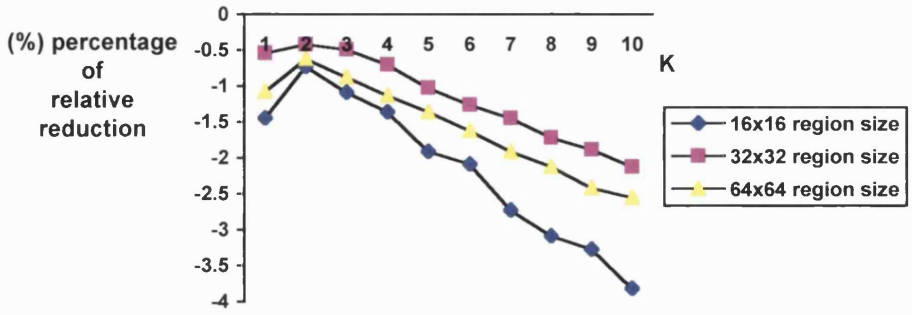
(b)

Figure 9.38: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.

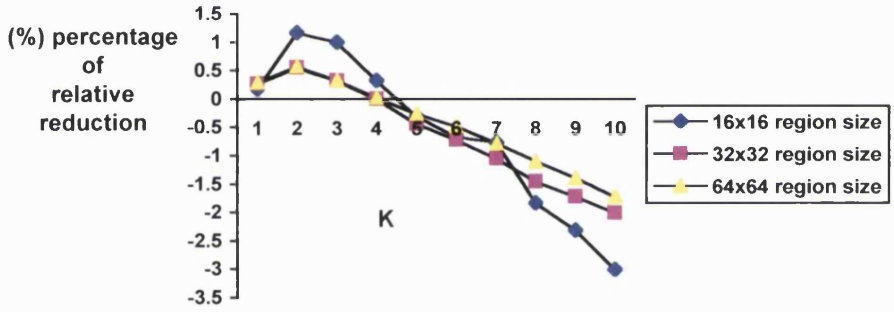


(a)

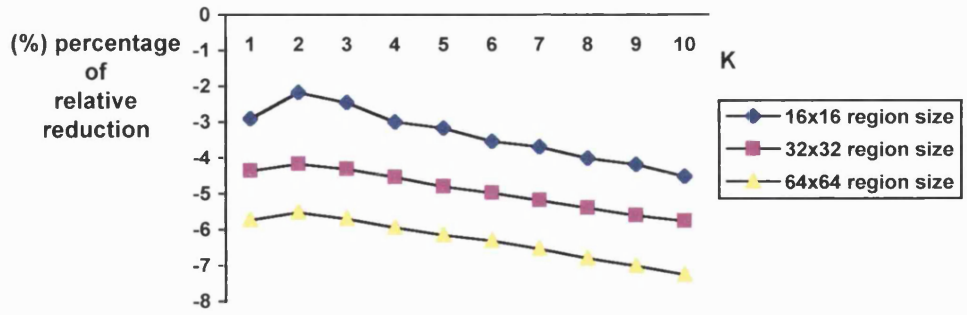


(b)

Figure **9**.39: Enhanced co-occurrence trees - static rule (a) semi-dynamic version, (b) full-dynamic version.

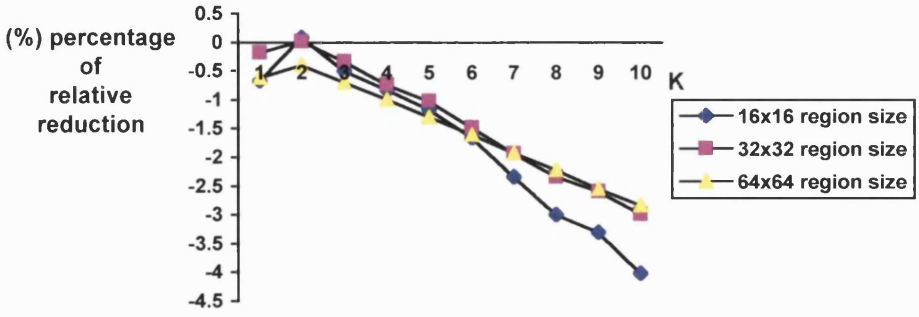


(a)

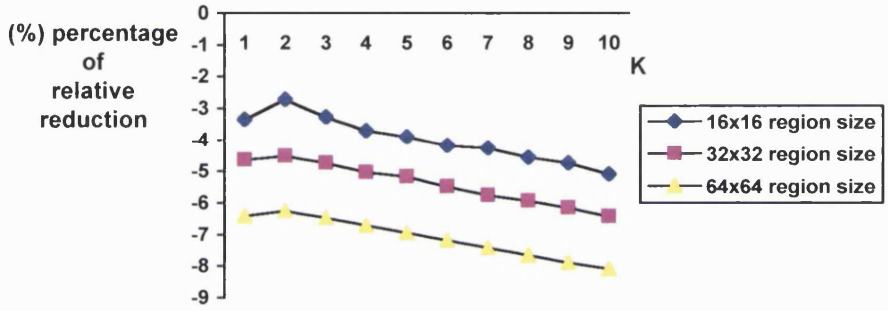


(b)

Figure **9**.40: Enhanced co-occurrence trees - move to front rule (a) semi-dynamic version, (b) full-dynamic version.



(a)

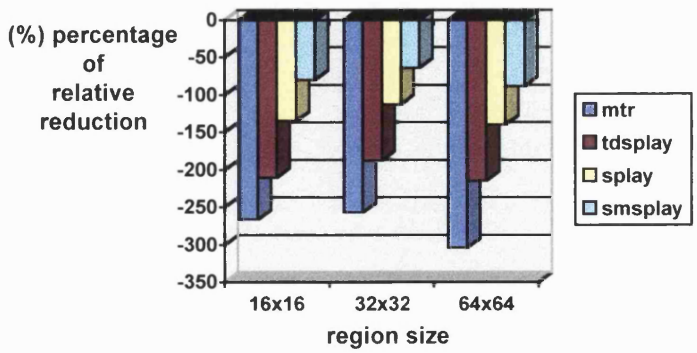


(b)

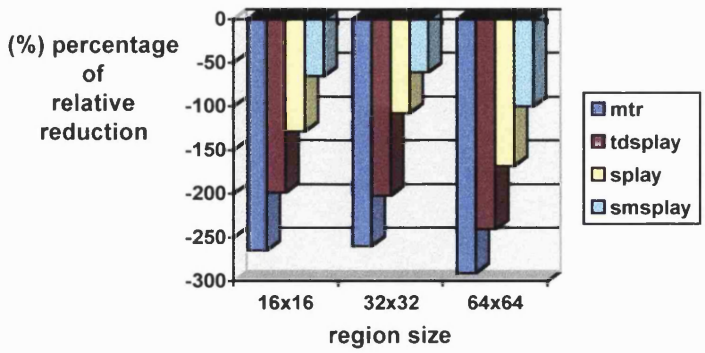
Figure 9.41: Enhanced co-occurrence trees - counter rule
(a) semi-dynamic version, (b) full-dynamic version.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.



(a)



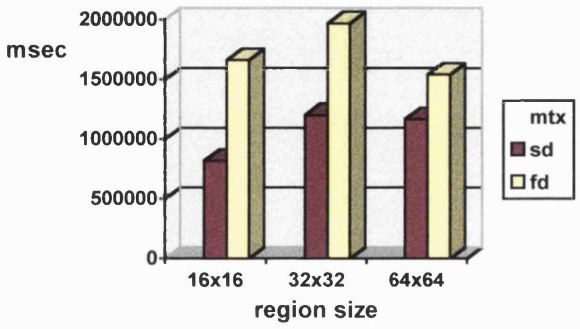
(b)

Figure 9.42: Self-adjusting co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

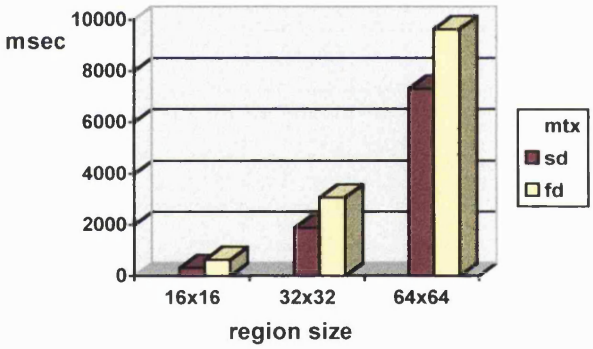
COMPUTATIONAL TIME RESULTS FOR THE WATER IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



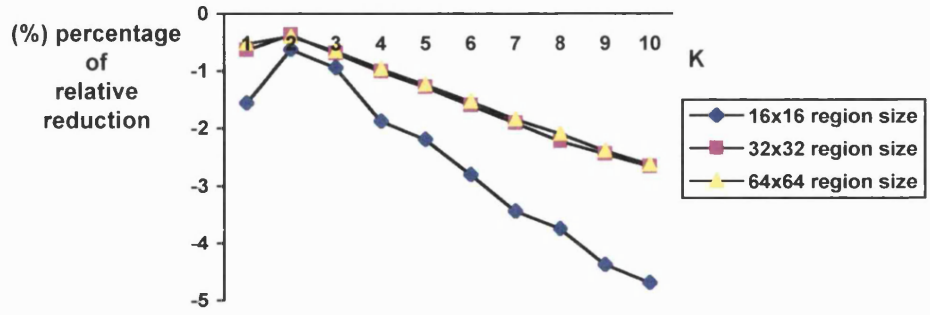
(a)



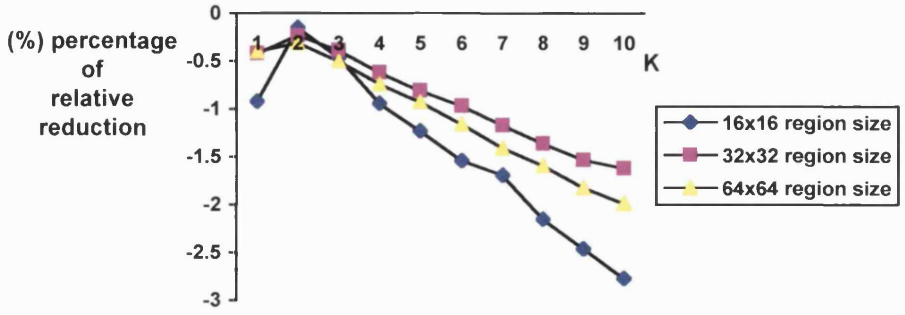
(b)

Figure 9.43: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.

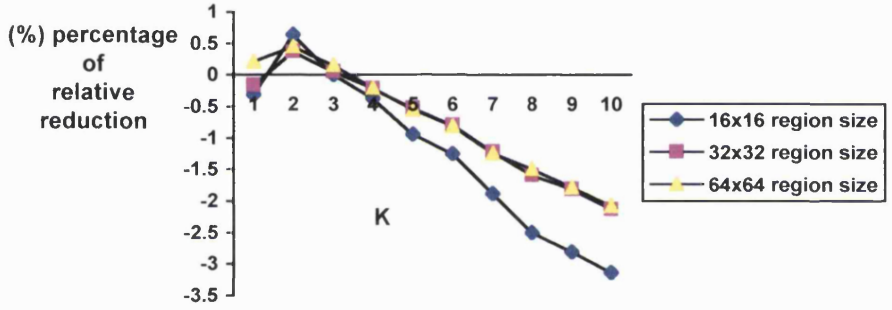


(a)

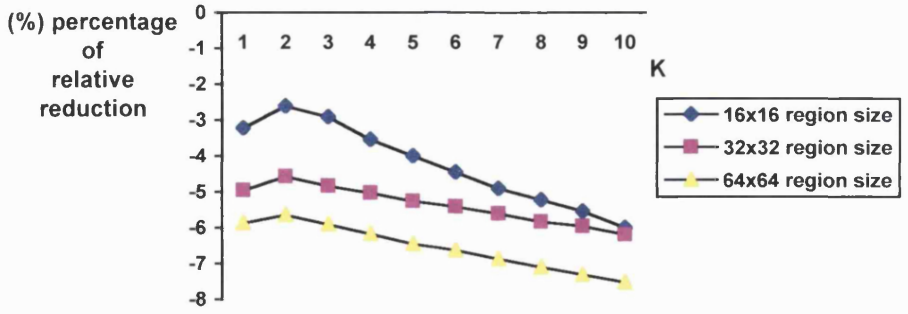


(b)

Figure 9.44: Enhanced co-occurrence trees - static rule (a) semi-dynamic version, (b) full-dynamic version.

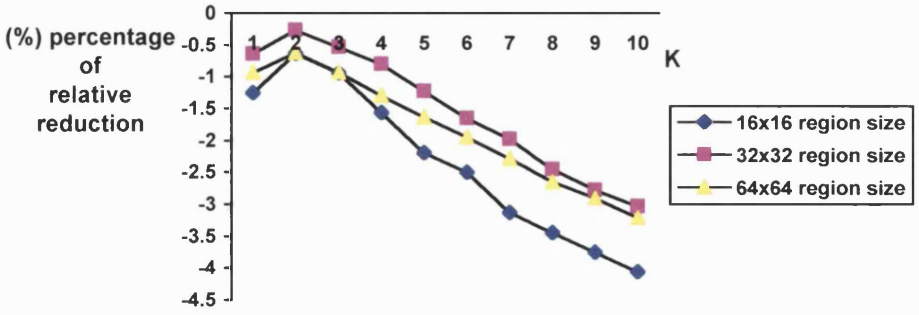


(a)

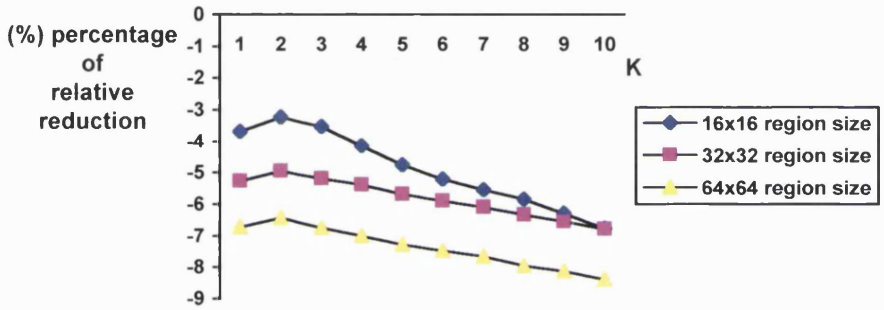


(b)

Figure 9.45: Enhanced co-occurrence trees - move to front rule (a) semi-dynamic version, (b) full-dynamic version.



(a)

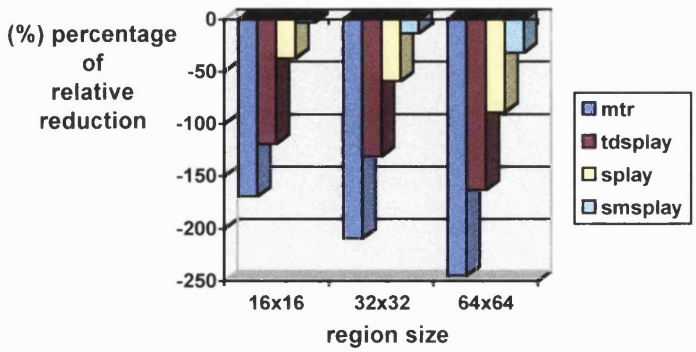


(b)

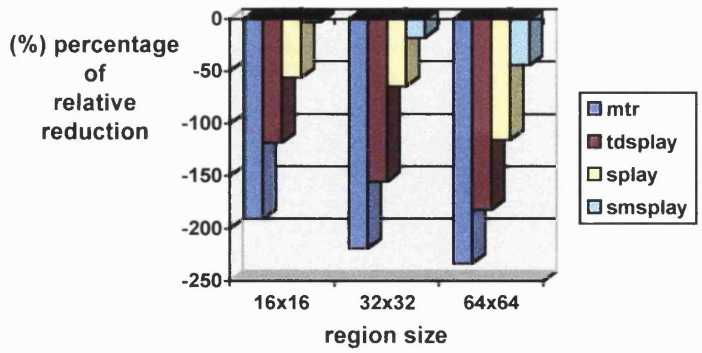
Figure 9.46: Enhanced co-occurrence trees - counter rule
(a) semi-dynamic version, (b) full-dynamic version.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.



(a)



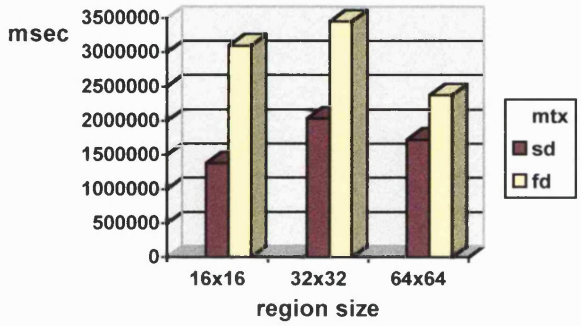
(b)

Figure 9.47: Self-adjusting co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

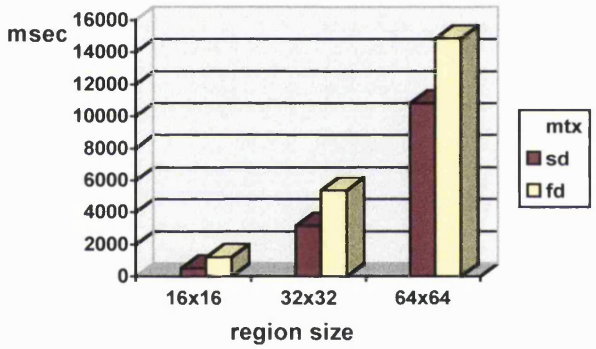
COMPUTATIONAL TIME RESULTS FOR THE WEAVE IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



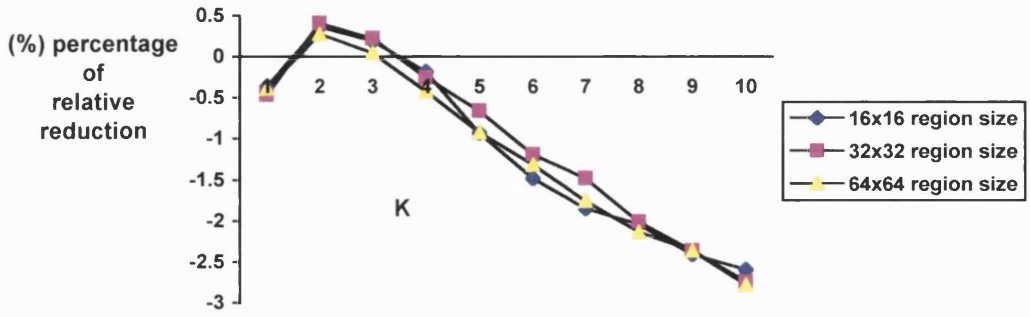
(a)



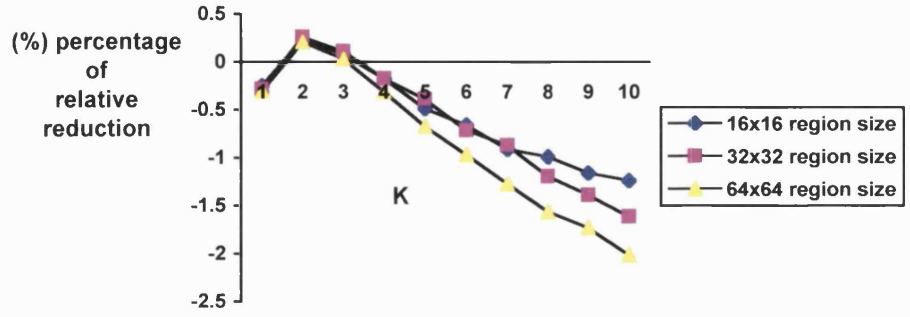
(b)

Figure 9.48: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.

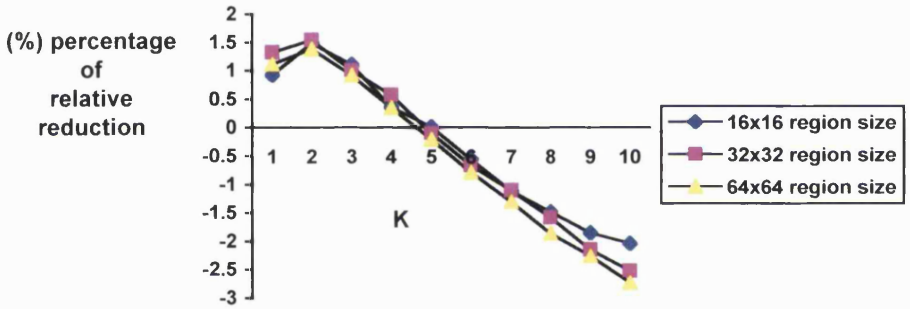


(a)

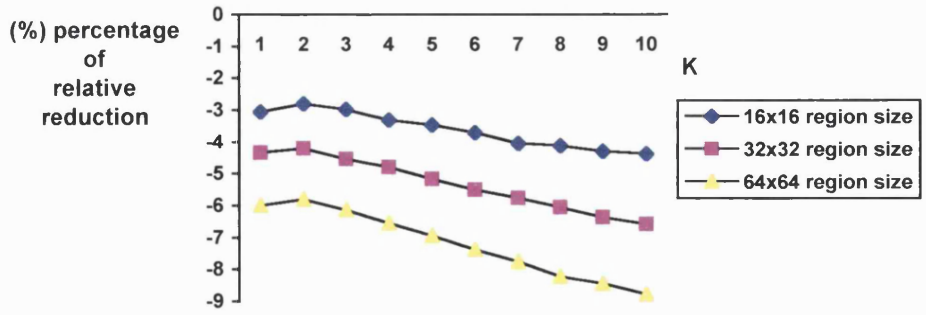


(b)

Figure 9.49: Enhanced co-occurrence trees - static rule (a) semi-dynamic version, (b) full-dynamic version.

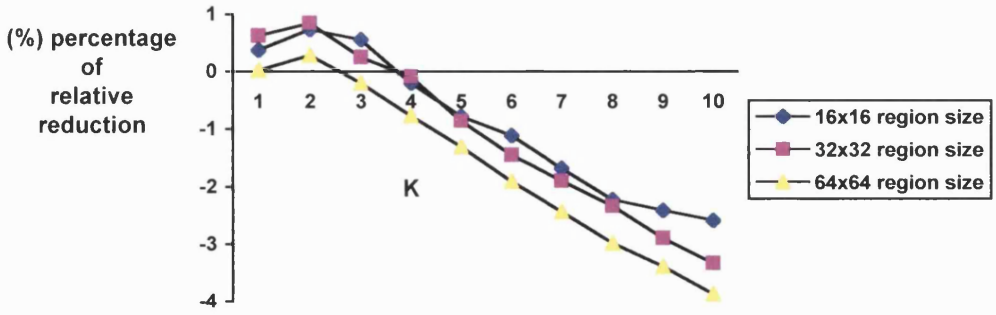


(a)

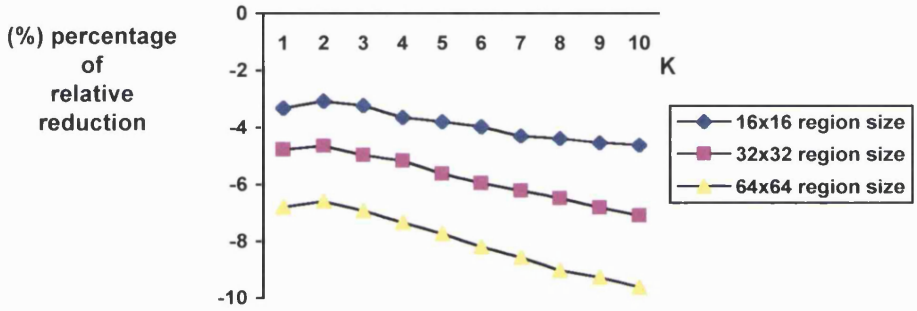


(b)

Figure 9.50: Enhanced co-occurrence trees - move to front rule (a) semi-dynamic version, (b) full-dynamic version.



(a)

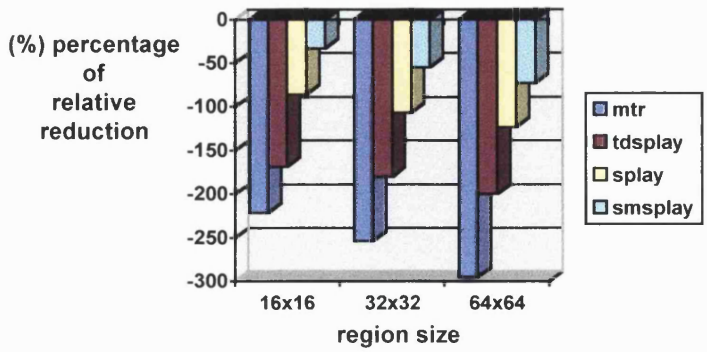


(b)

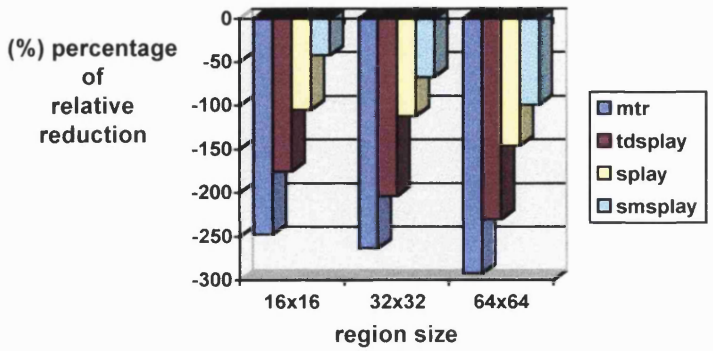
Figure 9.51: Enhanced co-occurrence trees - counter rule
(a) semi-dynamic version, (b) full-dynamic version.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.



(a)



(b)

Figure 9.52: Self-adjusting co-occurrence trees (a) semi-dynamic version, (b) full-dynamic version.

9.4.2 Time results from the analysis of normal mammograms

The second category consisted of digital X-ray images of the breast (mammograms; see Section **9**.3). All images illustrated normal breast tissue (see Figure **9**.19). Images of abnormal breast tissue gave similar computational time results and they were not considered in this study. All analysed images were 256 × 256 and they had a dynamic range of 6 bits (64 grey levels).

Nonoverlapping textural regions of size 64 × 64 were extracted from each mammogram. The total number of analysed regions was 480. Figures **9**.53–**9**.55 illustrate the computational time results for the normal mammogram data set.

9.4.3 Time results from the analysis of ultrasound images

The third category consisted of ultrasound images (see Section **3**.3). Five data sets were formed; the first data set contained ultrasound heart images, the second data set contained ultrasound kidney images, the third data set contained ultrasound liver images, the fourth data set contained ultrasound images of the ovaries and finally, the fifth data set contained ultrasound spleen images. All images had a dynamic range of 8 bits.

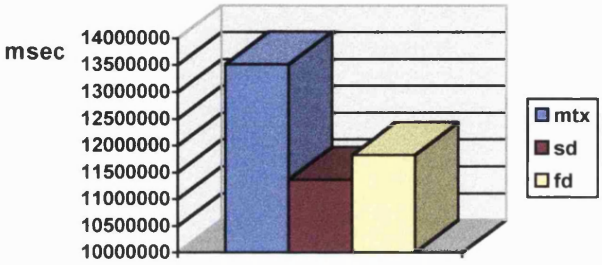
The ultrasound heart images had dimensions 256 × 256. The images were first normalised. 100 overlapping regions of size 6 × 6 were then extracted from these images illustrating the myocardium (see Figure **9**.21). Figures **9**.56–**9**.58 illustrate the computational time results for this image type. The ultrasound kidney images had dimensions 256 × 256, too. After normalisation, 100 overlapping regions of size 30 × 30 were extracted from these images illustrating kidney parenchyma (see Figure **9**.22). Figures **9**.59–**9**.61 show the time results for this image type. Similarly, the ultrasound liver images had dimensions 256 × 256. After normalisation, 100 overlapping regions of size 30 × 30 were extracted illustrating liver parenchyma (see Figure **9**.23). Figures **9**.62–**9**.64 illustrate the computational time results for this image type. The dimensions of the ultrasound images of the ovaries were 512 × 512. 40 overlapping regions of size 25 × 25 were extracted from all normalised images illustrating the ovarian tissue (see Figure **9**.24). Figures

9.65–**9**.67 show the time results for the ultrasound ovarian image type. Finally, the ultrasound images of the spleen had dimensions 256×256. After normalisation, 100 overlapping regions of size 30×30 were extracted from the images illustrating spleen parenchyma (see Figure **9**.25). Figures **9**.68–**9**.70 show the corresponding time results.

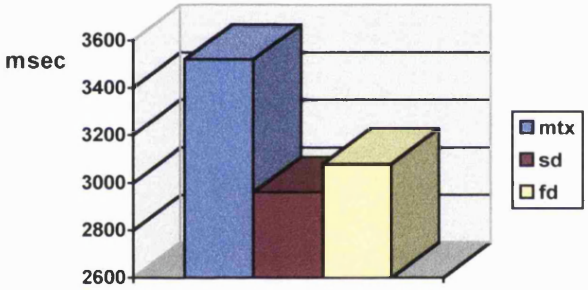
All analysed ultrasound image types are of great clinical interest. We saw in Section **3**.3 the importance of the texture of myocardium images in echocardiography. We also saw the diagnostic importance of the ultrasound images of liver parenchyma. All the other image types are also of great clinical interest to the radiologists.

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



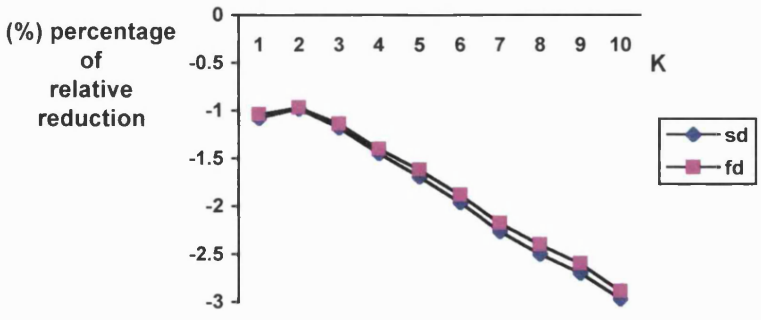
(a)



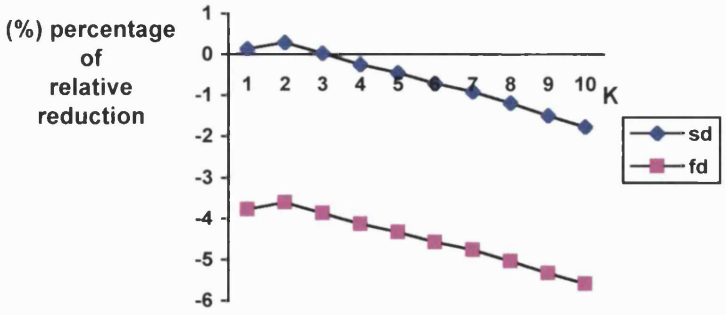
(b)

Figure 9.53: Plain co-occurrence trees (a) total time, (b) average time per region.

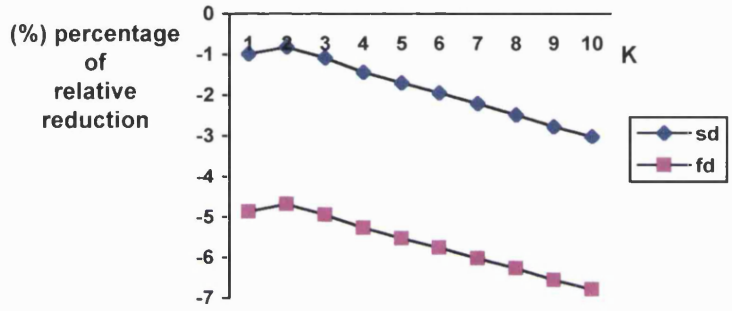
K: probability list length.



(a)



(b)



(c)

Figure 9.54: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

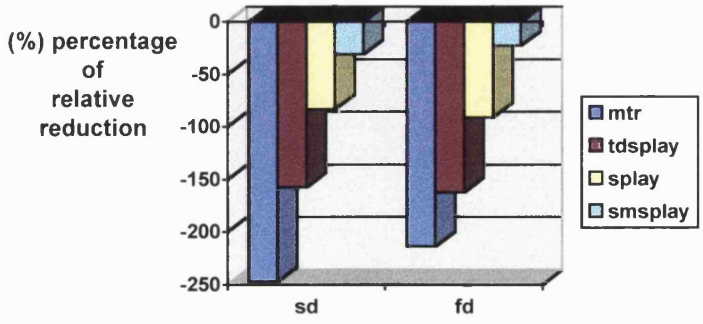
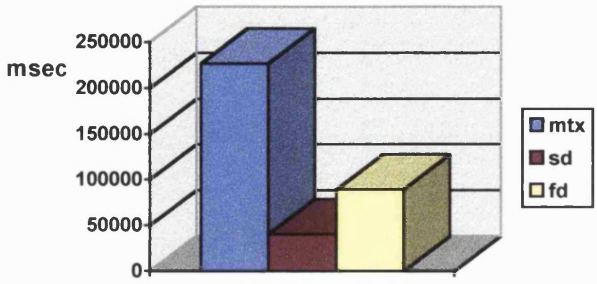


Figure 9.55: Self-adjusting co-occurrence trees.

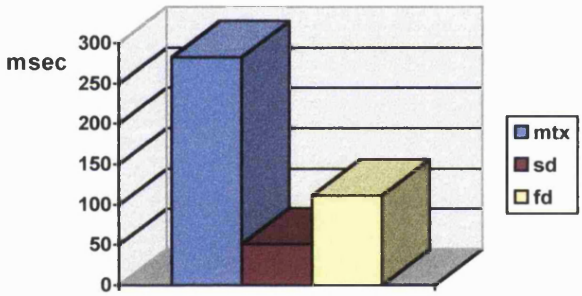
COMPUTATIONAL TIME RESULTS FOR THE ULTRASOUND HEART IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



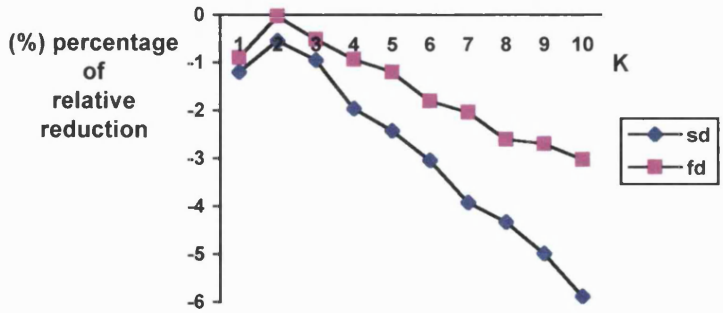
(a)



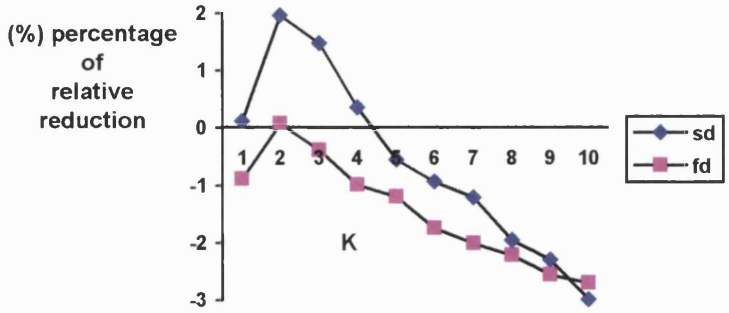
(b)

Figure 9.56: Plain co-occurrence trees (a) total time, (b) average time per region.

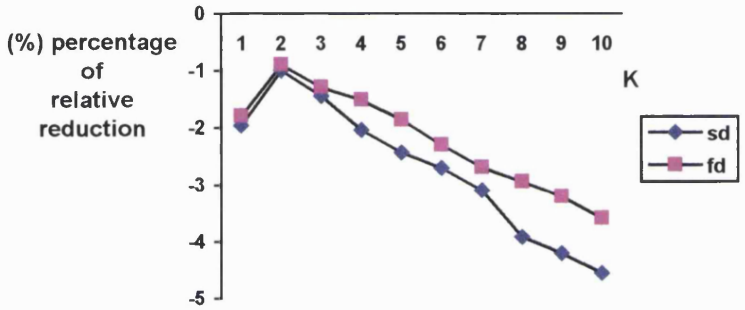
K: probability list length.



(a)



(b)



(c)

Figure **9**.57: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

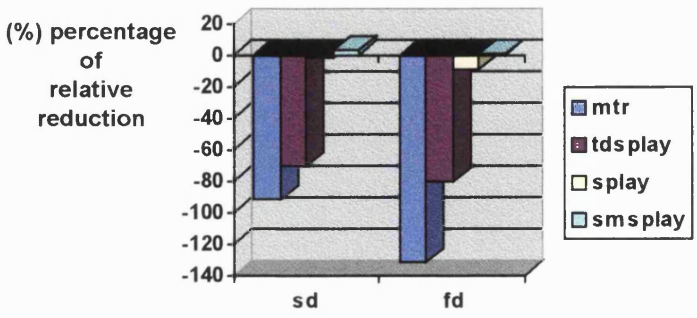
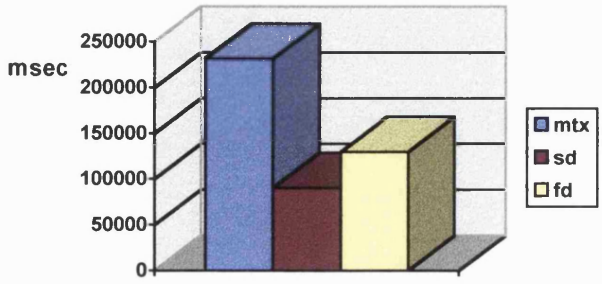


Figure **9**.58: Self-adjusting co-occurrence trees.

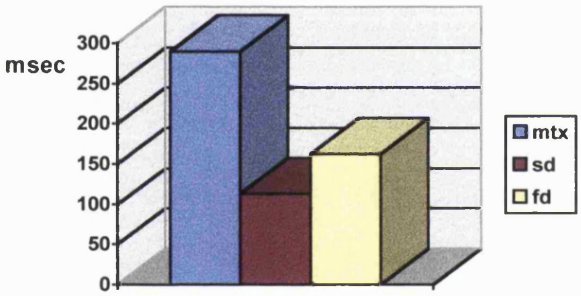
COMPUTATIONAL TIME RESULTS FOR THE ULTRASOUND KIDNEY IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



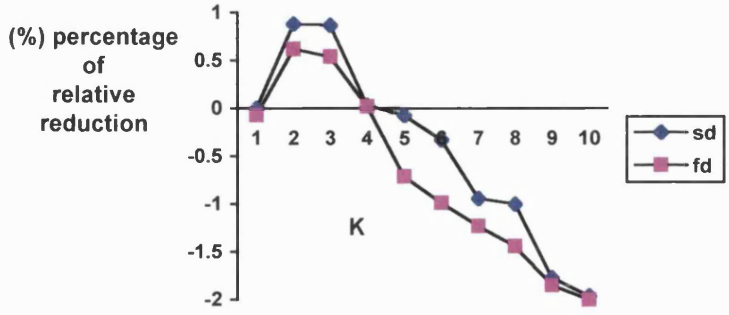
(a)



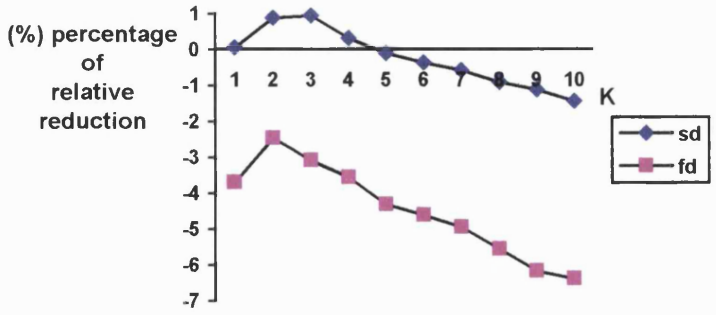
(b)

Figure 9.59: Plain co-occurrence trees (a) total time, (b) average time per region.

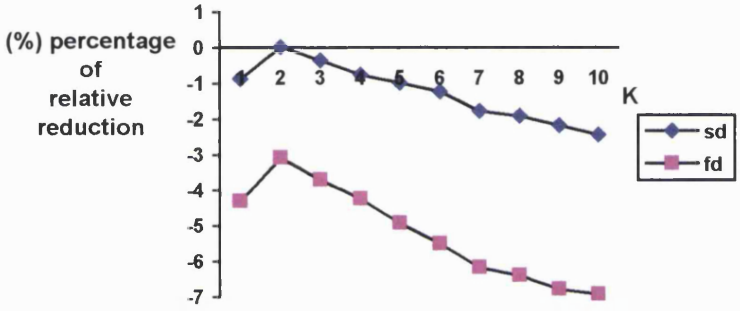
K: probability list length.



(a)



(b)



(c)

Figure **9**.60: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

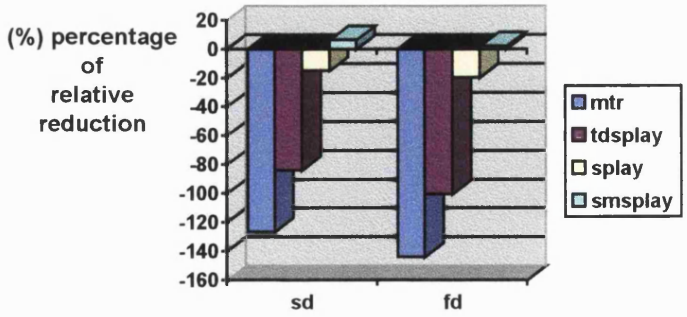
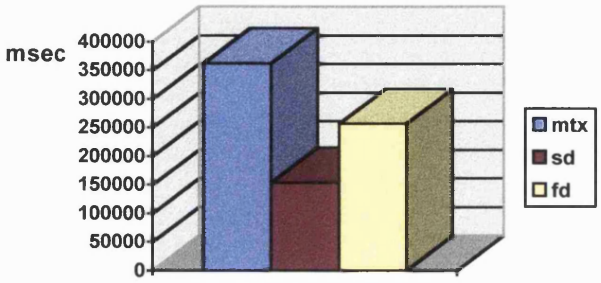


Figure **9**.61: Self-adjusting co-occurrence trees.

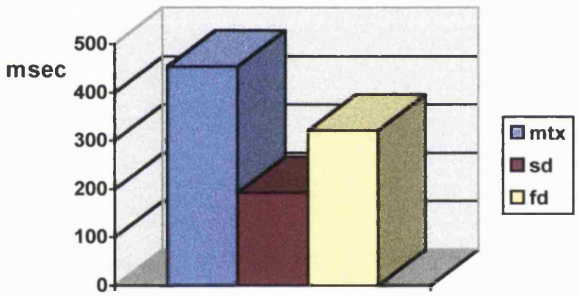
COMPUTATIONAL TIME RESULTS FOR THE ULTRASOUND LIVER IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



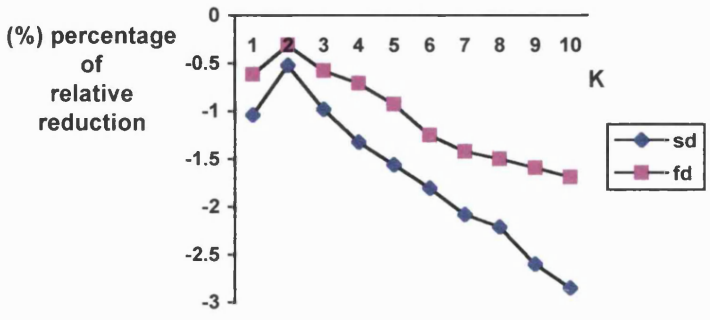
(a)



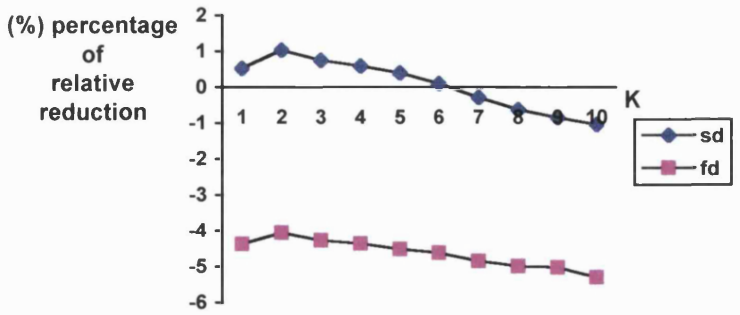
(b)

Figure 9.62: Plain co-occurrence trees (a) total time, (b) average time per region.

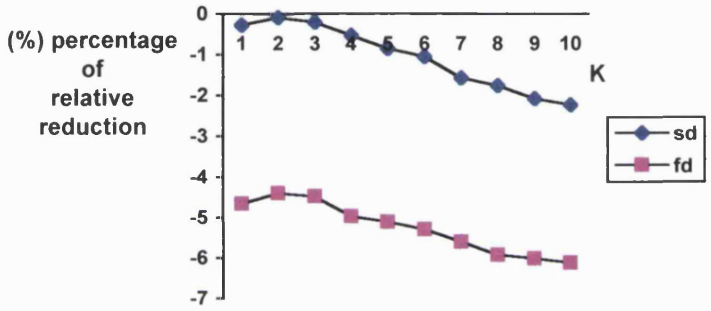
K: probability list length.



(a)



(b)



(c)

Figure 9.63: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

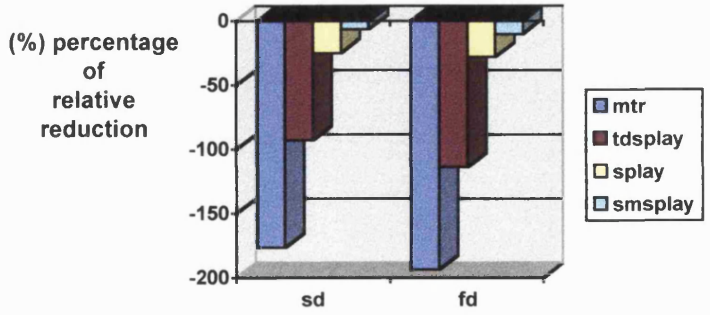
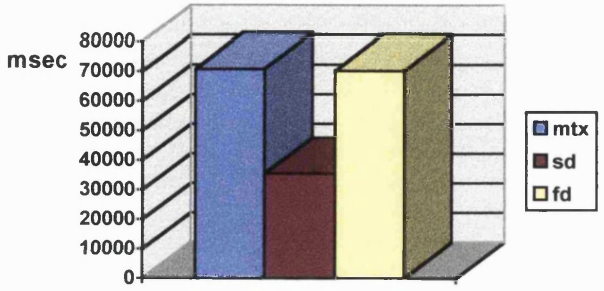


Figure 9.64: Self-adjusting co-occurrence trees.

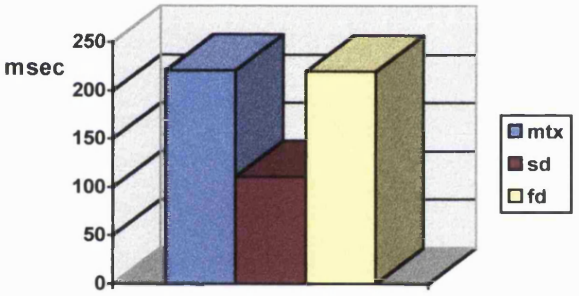
COMPUTATIONAL TIME RESULTS FOR THE ULTRASOUND OVARIES IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



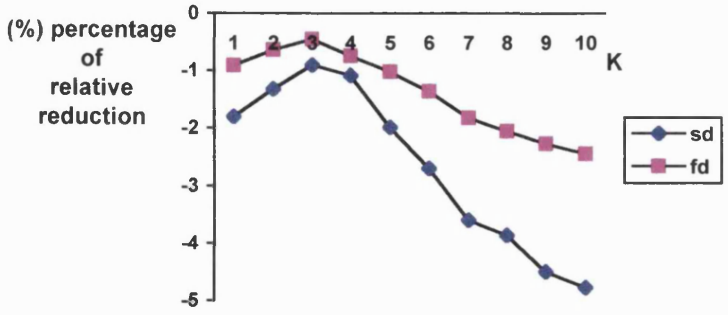
(a)



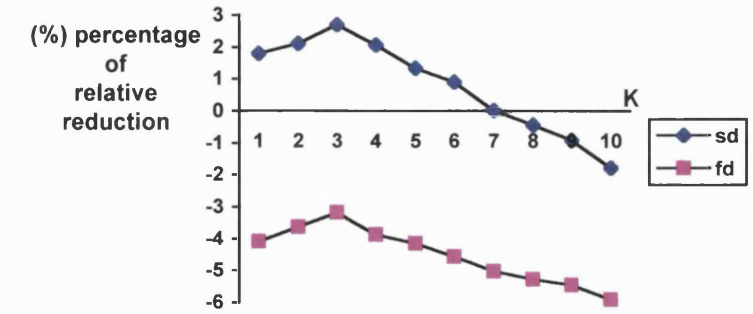
(b)

Figure 9.65: Plain co-occurrence trees (a) total time, (b) average time per region.

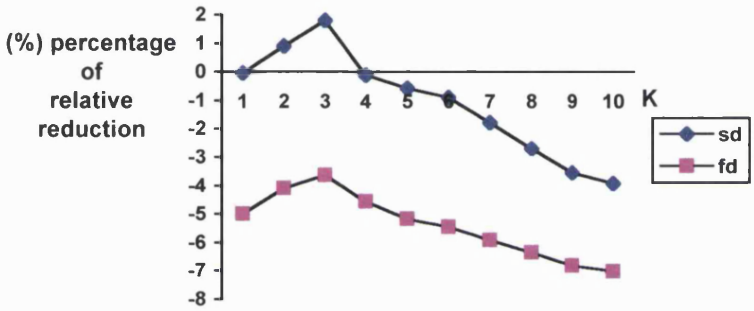
K: probability list length.



(a)



(b)



(c)

Figure 9.66: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

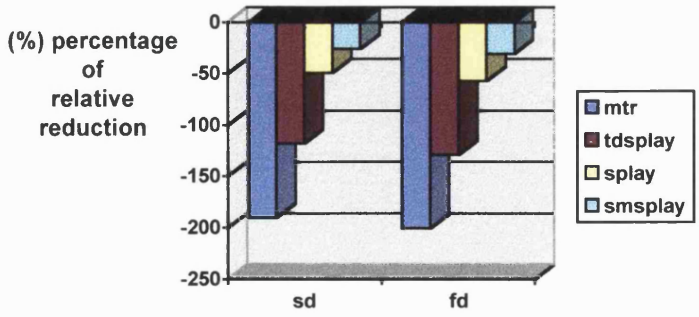
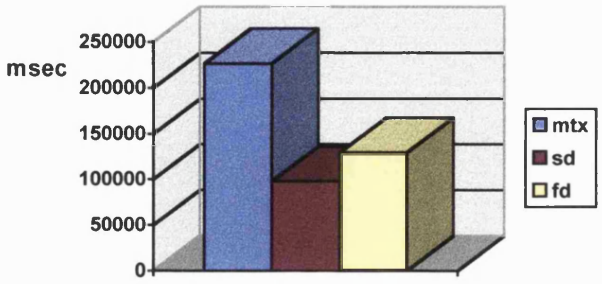


Figure 9.67: Self-adjusting co-occurrence trees.

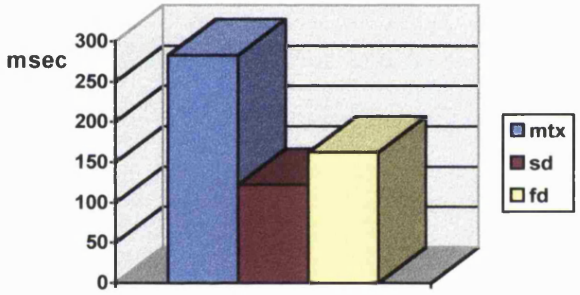
COMPUTATIONAL TIME RESULTS FOR THE ULTRASOUND SPLEEN IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



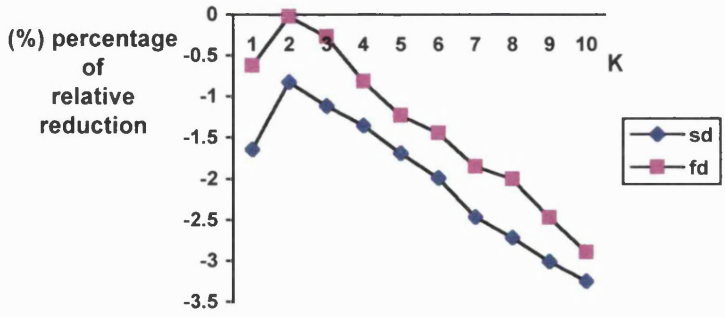
(a)



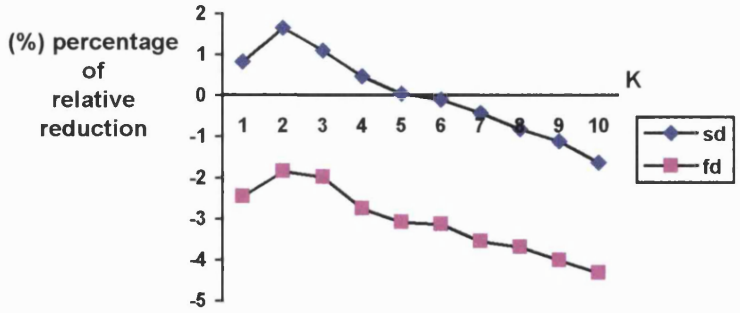
(b)

Figure **9**.68: Plain co-occurrence trees (a) total time, (b) average time per region.

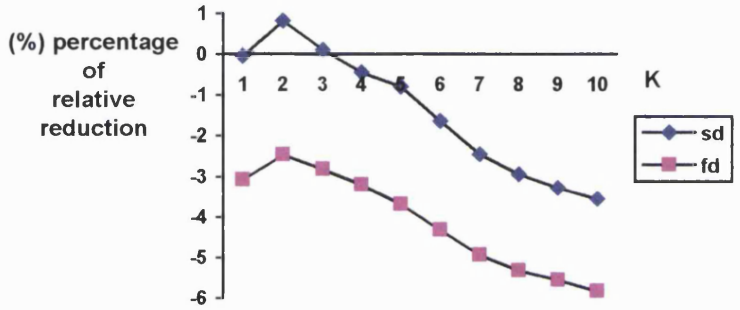
K: probability list length.



(a)



(b)



(c)

Figure **9**.69: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

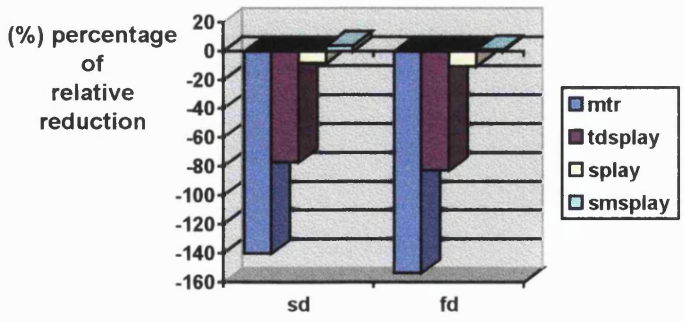


Figure **9**.70: Self-adjusting co-occurrence trees.

9.4.4 Time results from the analysis of MR images

The fourth category consisted of MR images (see Section **3**.4). This category included five data sets. The first data set contained images of the human femur. The images had dimensions 512 × 512 and a dynamic range of 15 bits. After normalisation, 90 overlapping regions of size 32 × 32 were extracted illustrating the trabecular bone tissue (see Figure **9**.71). Figures **9**.76–**9**.78 illustrate the computational time results for this image type. Note that only the full-dynamic version of the various kinds of the co-occurrence trees was able to compute the textural features. Also, the static rule of the enhanced co-occurrence trees failed to produce any results for both the semi-dynamic and the full-dynamic version.

The second data set contained images of the human brain. The dimensions of the images were 256 × 256 and their dynamic range was 12 bits. 60 overlapping regions of size 10 × 10 were extracted after normalisation (see Figure **9**.72). The selected regions illustrated only white matter of the brain. Figures **9**.79–**9**.81 show the computational time results for this data set. The third data set contained images of the human brain, as well. The dimensions of the images were again 256 × 256 but the dynamic range was 8 bits. 60 overlapping regions of size 8 × 8 were extracted after normalisation. Half of them illustrated grey matter and the rest illustrated white matter of the human brain (see Figure **9**.73). Figures **9**.82–**9**.84 illustrate the corresponding computational time results.

The fourth data set contained images of human skeletal muscle. These were actually cross-sections of the thigh (see Figure **9**.74). The dimensions of the images were 256 × 256 and the dynamic range was 12 bits. 200 overlapping regions of size 10 × 10 were extracted after the images were normalised. Figures **9**.85–**9**.87 show the time results for this data set. Note that for the enhanced co-occurrence trees only the full-dynamic version was able to compute the textural features. Also, the static rule failed to give results for both the semi-dynamic and the full-dynamic version. Finally, the fifth data set also contained images of human skeletal muscle, but this time the images illustrated sagittal sections of the thigh (see Figure **9**.75). The dimensions of the images were 256 × 256 and the dynamic range was 15 bits. After the images were normalised, 200 overlapping

regions of size 16×16 were extracted. Figures **9**.88–**9**.90 show the time results for this data set. Similarly to the MR human femur data set, only the full-dynamic version of the co-occurrence trees was able to produce any results. In addition, the static rule was not able to compute the textural features for both the semi-dynamic and the full-dynamic version in this case, as well.

As we have already said (Section **3**.4), the texture contained in MR images has been shown to be of potential clinical importance. Texture analysis of the trabecular bone tissue images can reveal useful information about the metabolic state of bone tissue. In addition, texture analysis of MR brain images can give valuable information for the identification of various abnormalities, such as brain tumours. Also, the texture in the MR skeletal muscle images is of potential value.

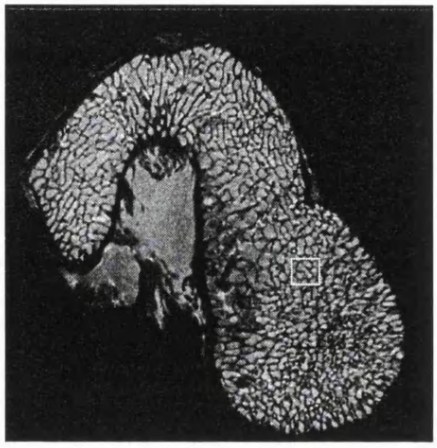


Figure **9**.71: MR femur image.

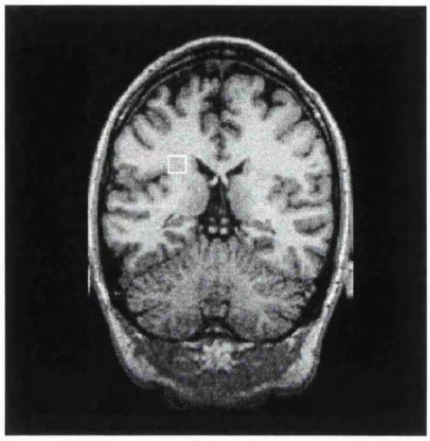


Figure. **9**.72: MR brain image (12 bit).

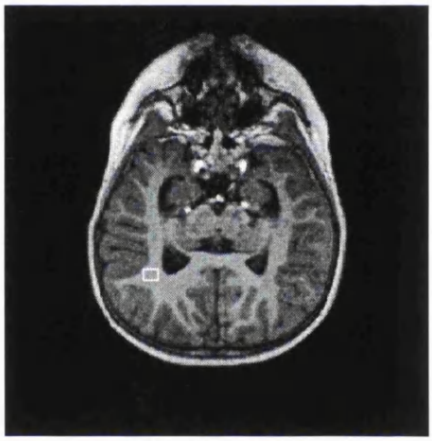


Figure **9**.73: MR brain image (8 bit).

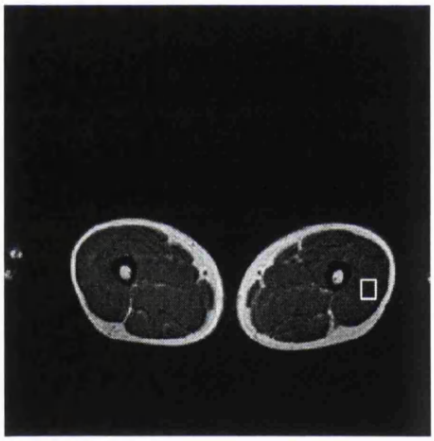
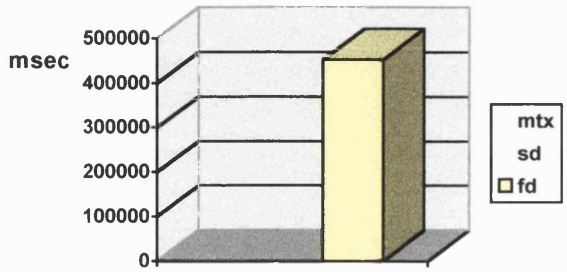


Figure **9**.74: MR muscle image (12 bit).

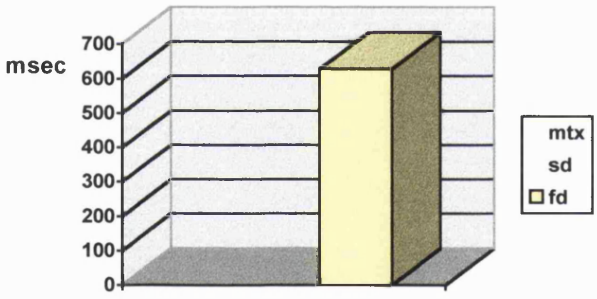
Figure **9**.75: MR muscle image (15 bit).

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



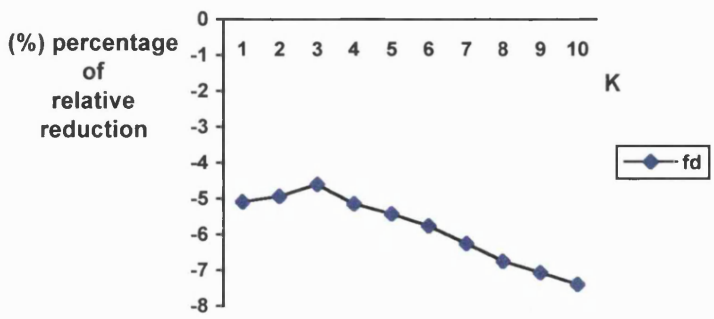
(a)



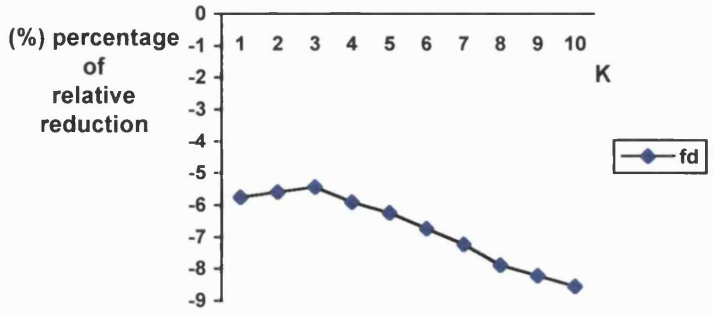
(b)

Figure 9.76: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.77: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

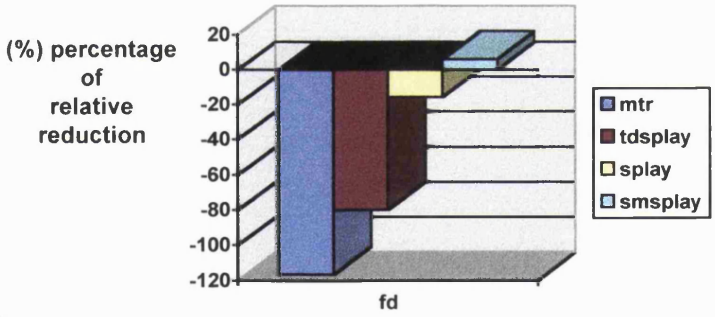
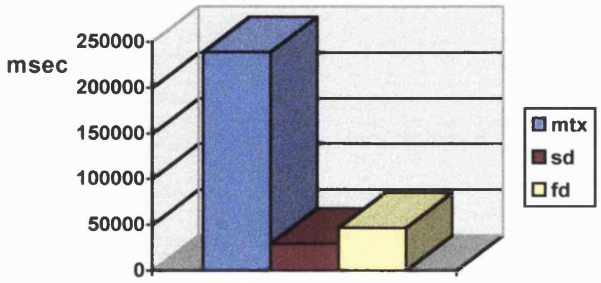


Figure 9.78: Self-adjusting co-occurrence trees (only the full-dynamic version).

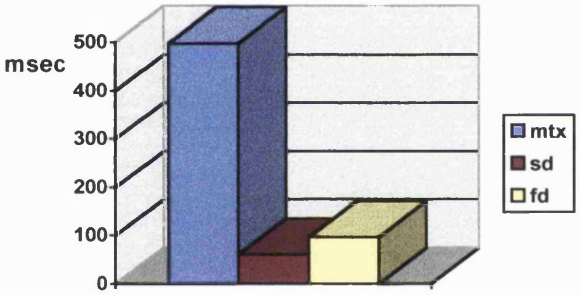
COMPUTATIONAL TIME RESULTS FOR THE 12 BIT MR BRAIN IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



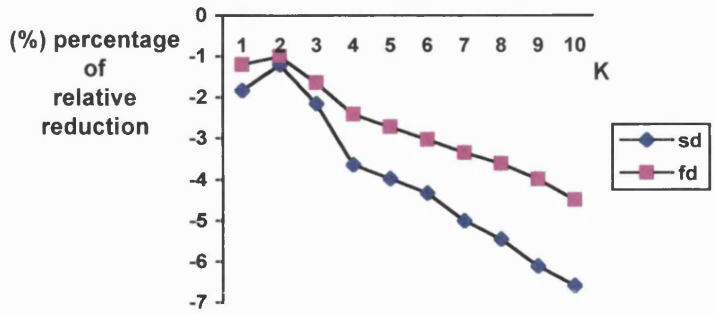
(a)



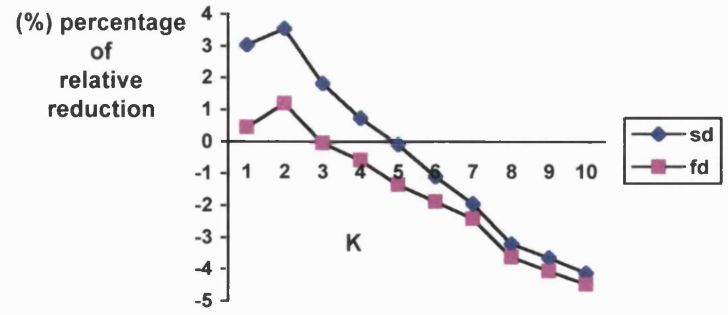
(b)

Figure 9.79: Plain co-occurrence trees (a) total time, (b) average time per region.

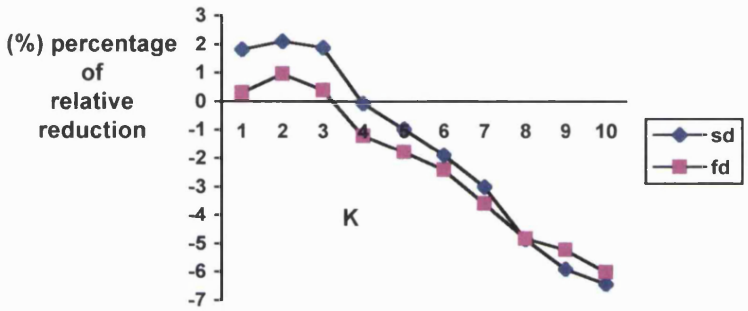
K: probability list length.



(a)



(b)



(c)

Figure 9.80: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

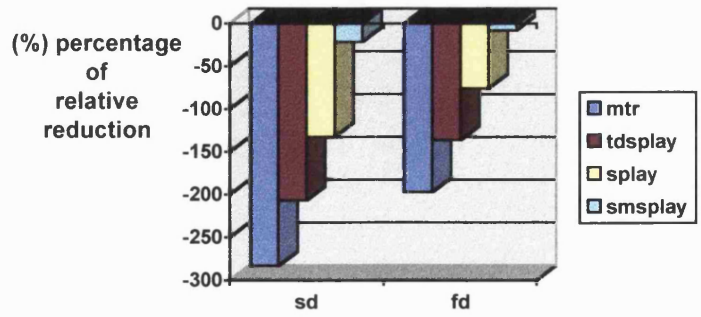
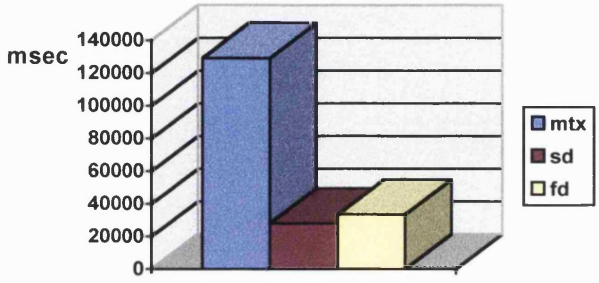


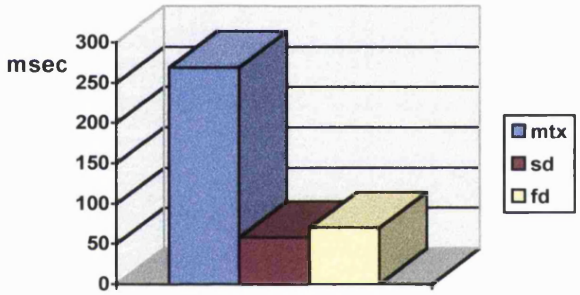
Figure 9.81: Self-adjusting co-occurrence trees.

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



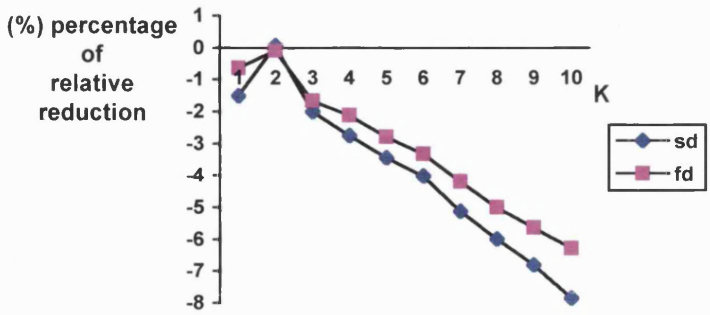
(a)



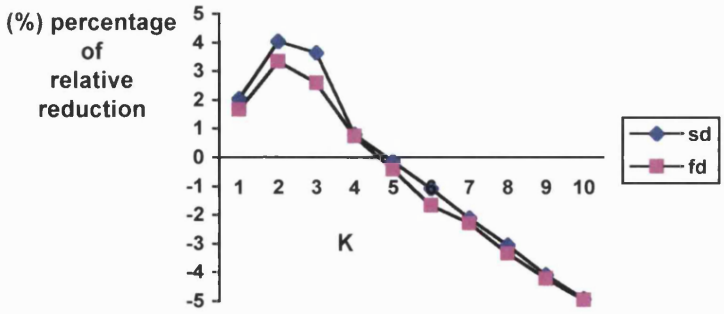
(b)

Figure 9.82: Plain co-occurrence trees (a) total time, (b) average time per region.

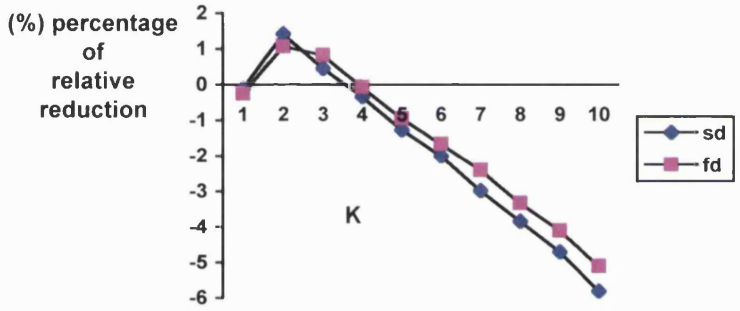
K: probability list length.



(a)



(b)



(c)

Figure 9.83: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

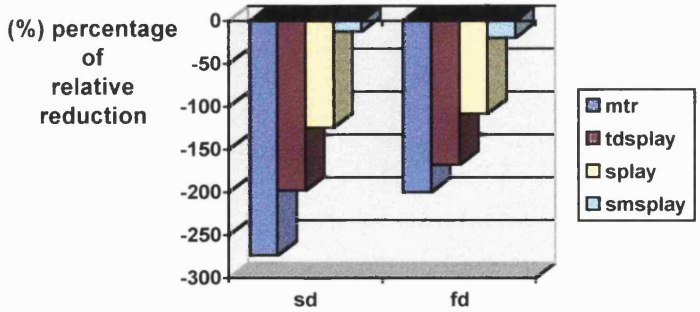
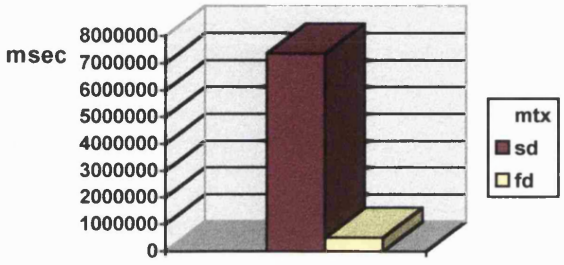


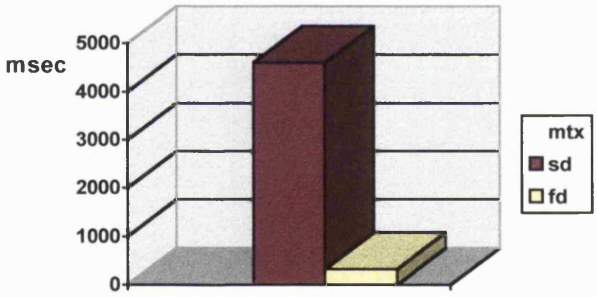
Figure 9.84: Self-adjusting co-occurrence trees.

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



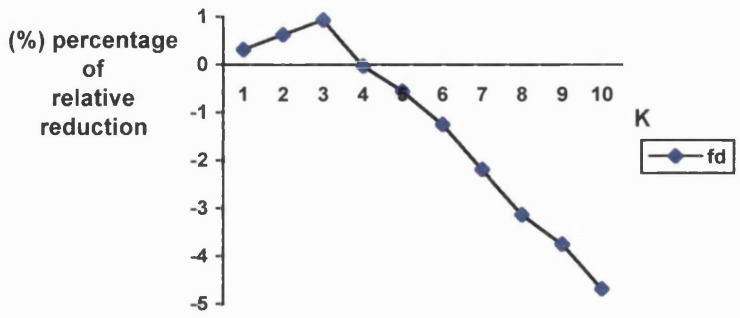
(a)



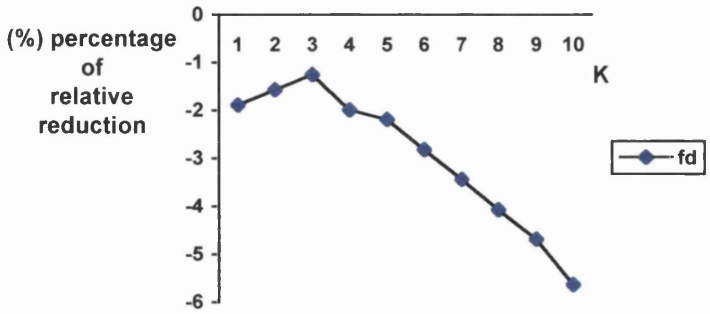
(b)

Figure 9.85: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.86: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

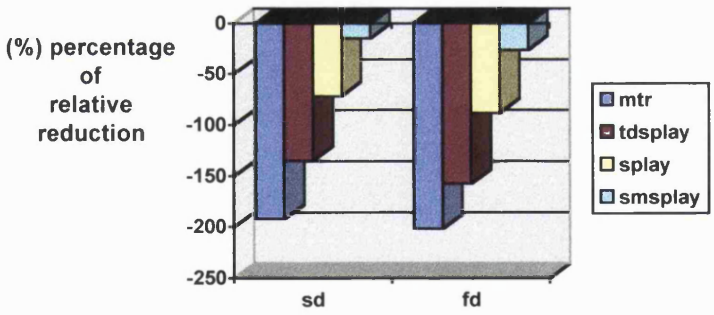
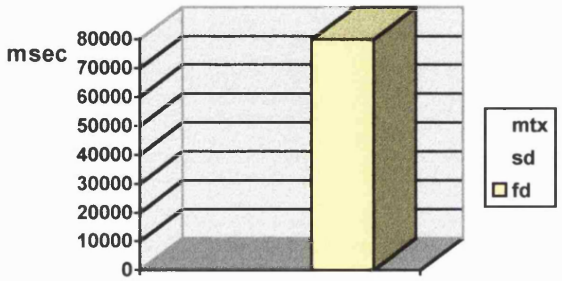


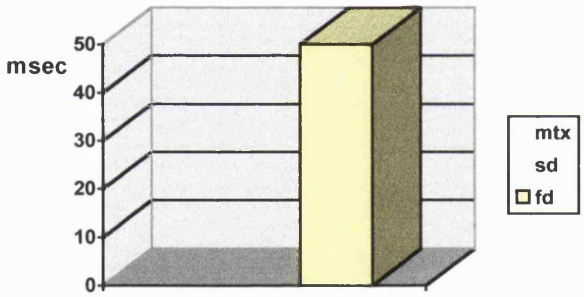
Figure 9.87: Self-adjusting co-occurrence trees.

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



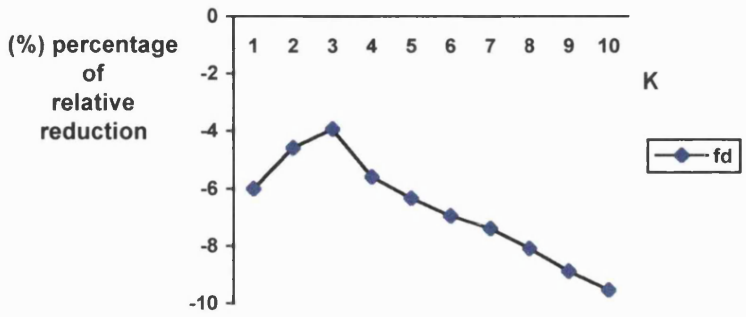
(a)



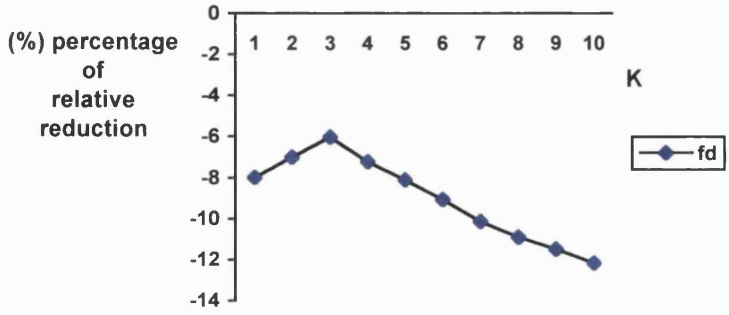
(b)

Figure 9.88: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.89: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

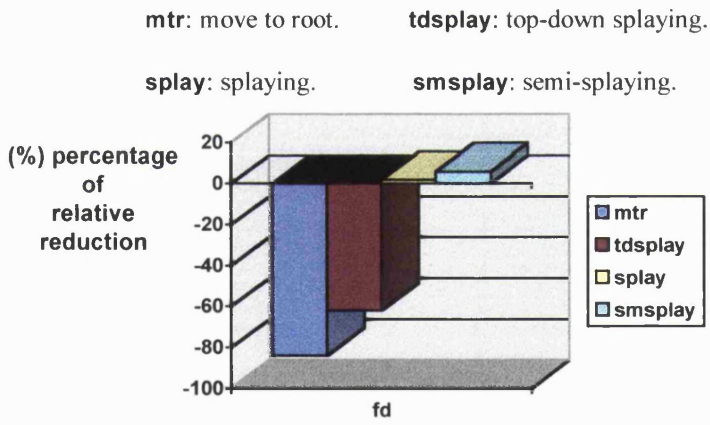


Figure 9.90: Self-adjusting co-occurrence trees (only the full-dynamic version).

9.4.5 Time results for the analysis of CT images

Finally, the last category consisted of CT images. It included five data sets, as well. The first data set contained images of the human liver. The images had dimensions 512×512 and a dynamic range of 12 bits. After normalisation, 60 overlapping regions of size 32×32 were selected for texture analysis (see Figure **9**.91). Figures **9**.96–**9**.98 show the computational time results for this data set. Note that only the full-dynamic version was able to compute the textural features in the case of the enhanced co-occurrence trees. Also, the static rule was not able to produce any results for both the semi-dynamic and the full-dynamic version. The second data set contained images of the human liver, as well. These images had also dimensions 512×512 and a 12 bit dynamic range. However, the size of the selected regions was now 16×16. Again, 60 overlapping regions were extracted for texture analysis after normalisation (see Figure **9**.92). Figures **9**.99–**9**.101 show the corresponding time results for this data set. The same observations can be made here, as in the case of the previous CT liver data set. All regions in this data set as well as the previous one illustrated liver parenchyma.

The third data set contained images of the human brain. The dimensions of the images were 206×206 and their dynamic range was 12 bits. 60 overlapping regions of size 16×16 were extracted after normalisation (see Figure **9**.93). They illustrated white/grey matter. Figure **9**.102–**9**.104 show the computational time results for this data set. Again, only the full-dynamic version was able to give results in the case of the enhanced co-occurrence trees. Moreover, the static rule failed to give any results for both versions. The fourth data set contained images of the human brain, as well. The dimensions of the images were 230×202 and the dynamic range was 12 bits. Again, 60 overlapping regions were extracted after normalisation which illustrated white/grey matter (see Figure **9**.94). The size of the regions was 32×32. Figures **9**.105–**9**.107 show the corresponding time results for this data set. The same observations can be made here, as in the case of the previous MR brain data set.

Finally, the fifth data set contained images of the human lungs. The dimensions of the images were 383 × 479 and their dynamic range was 8 bits. After normalisation, 60 overlapping regions of size 32 × 32 were extracted which illustrated the lung tissue (see Figure **9**.95). Figures **9**.108–**9**.110 illustrate the computational time results for the CT lung data set.

In Section **3**.5, we referred to the medical significance of the CT human abdomen images. Of particular interest is liver parenchyma. Texture analysis has been used with success in differentiating normal from abnormal liver tissue and in specifying the type of abnormality, such as cirrhosis. Also, the analysis of the brain tissue in CT head images is of vital importance to the assessment of a number of physiological parameters and pathological diseases. Another field of study investigates the usefulness of the texture analysis of CT lung images in the diagnosis of various lung diseases.

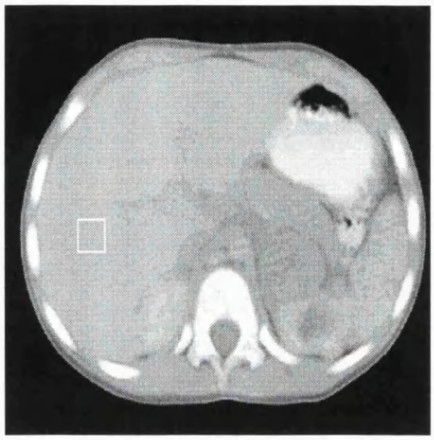


Figure **9**.91: CT liver image (32x32).

Figure **9**.92: CT liver image (16x16).

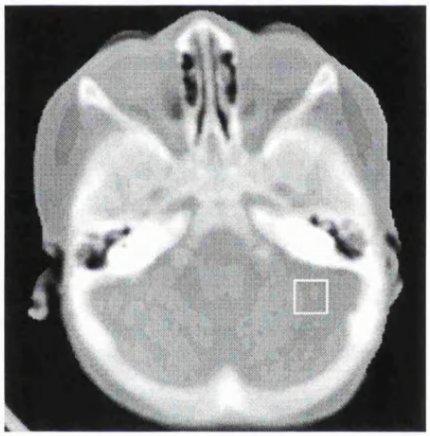


Figure **9**.93: CT brain image (16x16).

Figure **9**.94: CT brain image (32x32).

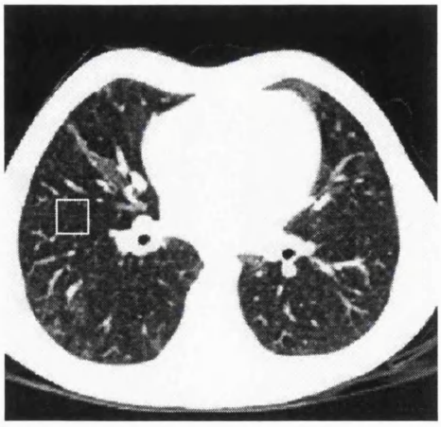
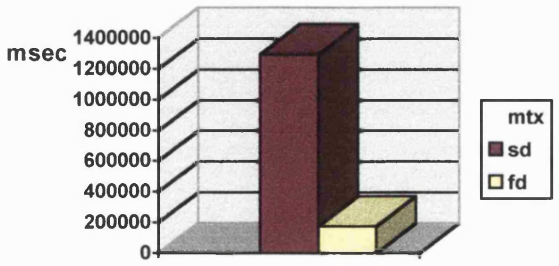


Figure **9**.95: CT human lung image.

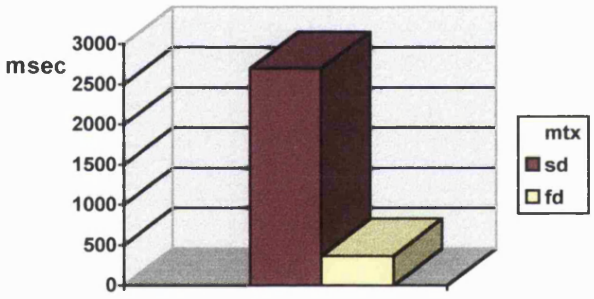
COMPUTATIONAL TIME RESULTS FOR THE CT LIVER IMAGES (32x32 REGION SIZE)

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



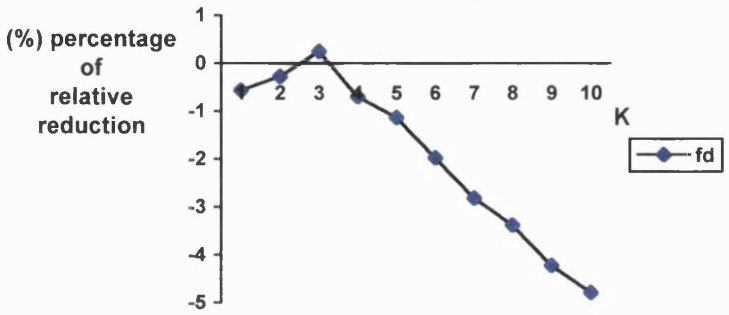
(a)



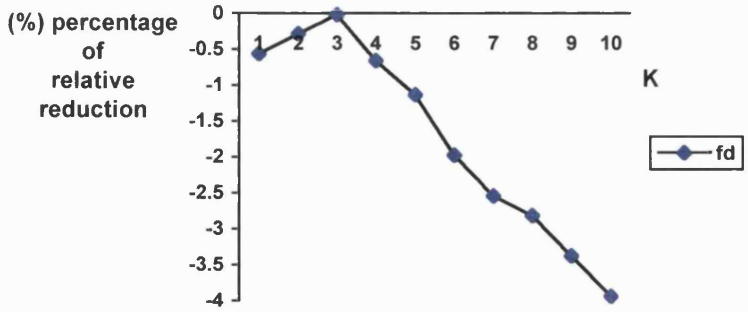
(b)

Figure 9.96: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.97: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

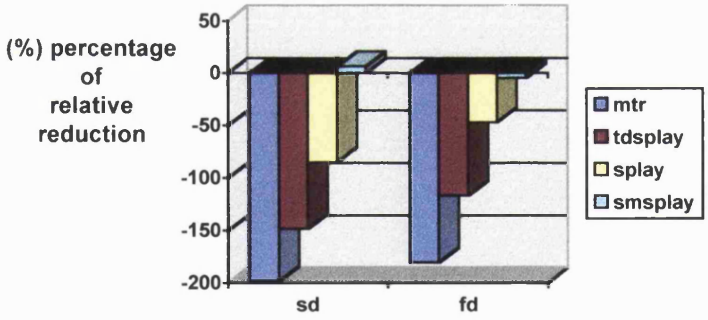
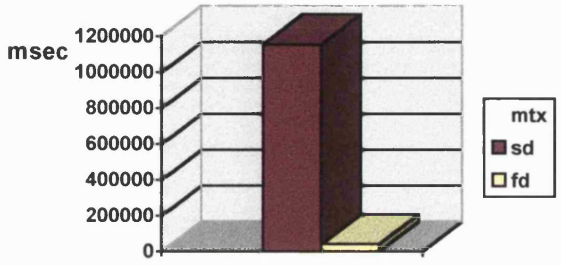


Figure 9.98: Self-adjusting co-occurrence trees.

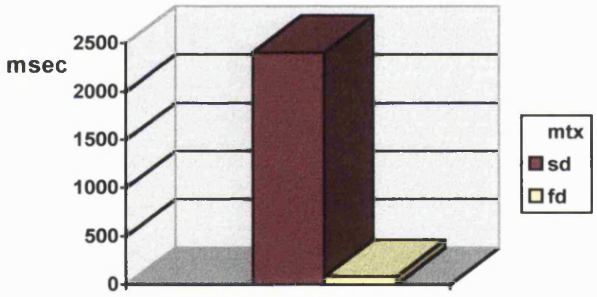
COMPUTATIONAL TIME RESULTS FOR THE CT LIVER IMAGES (16x16 REGION SIZE)

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



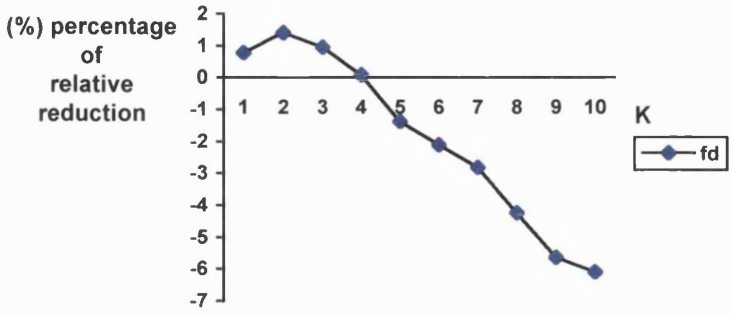
(a)



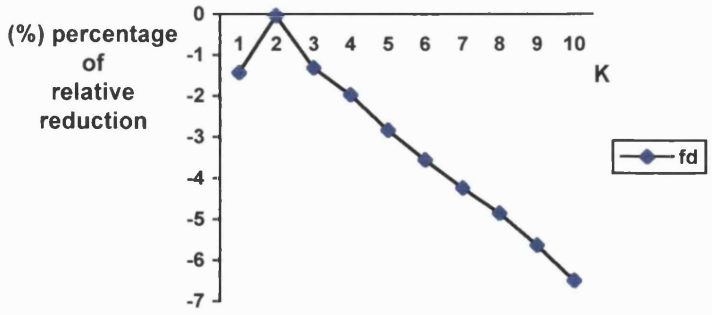
(b)

Figure 9.99: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.100: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

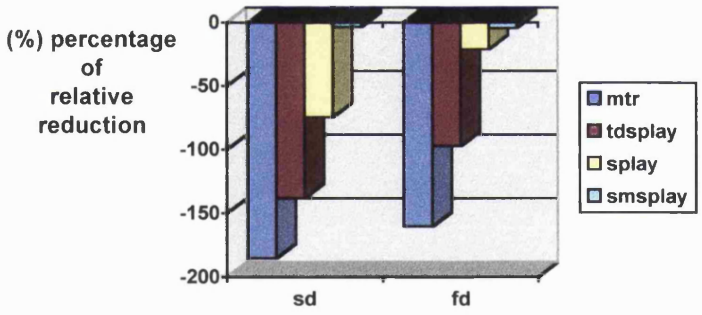
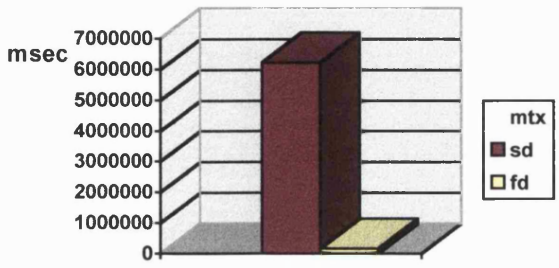


Figure 9.101: Self-adjusting co-occurrence trees.

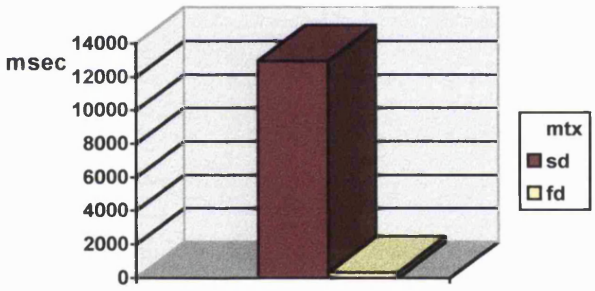
COMPUTATIONAL TIME RESULTS FOR THE CT BRAIN IMAGES (16x16 REGION SIZE)

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



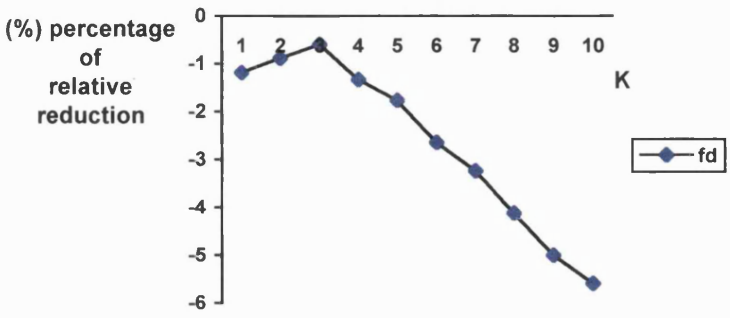
(a)



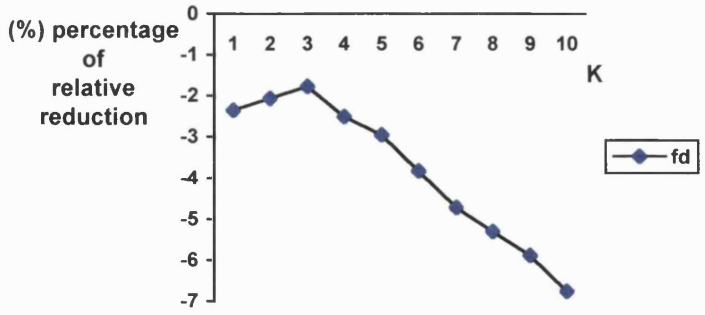
(b)

Figure 9.102: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.103: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

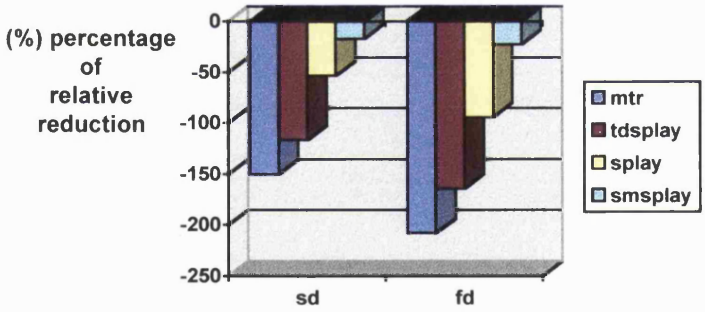
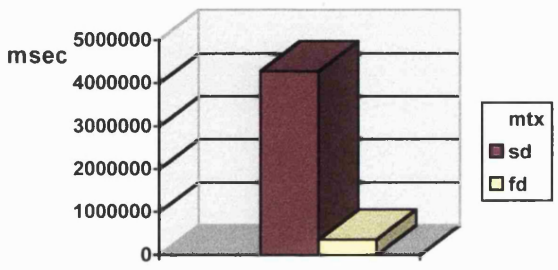


Figure 9.104: Self-adjusting co-occurrence trees.

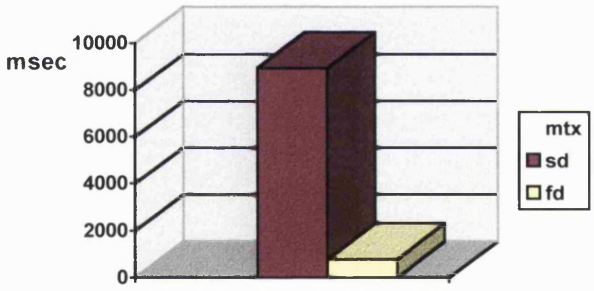
COMPUTATIONAL TIME RESULTS FOR THE CT BRAIN IMAGES (32x32 REGION SIZE)

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



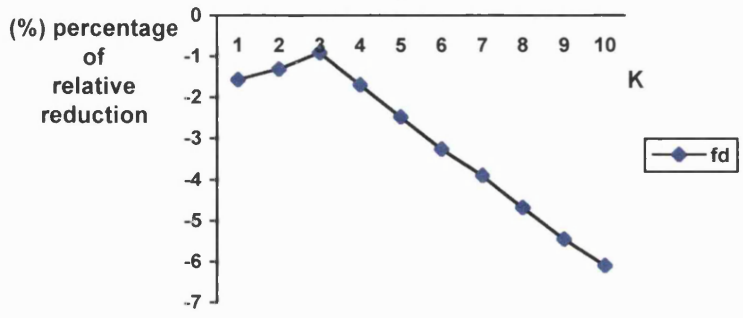
(a)



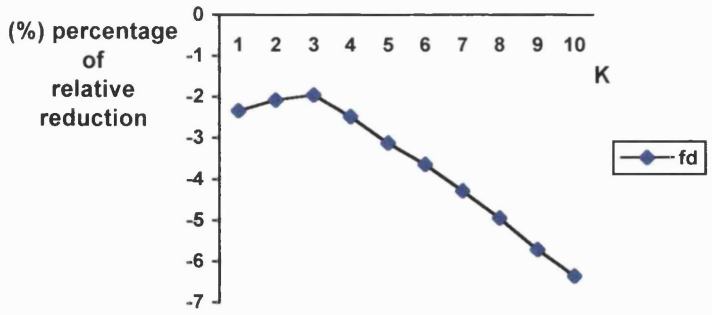
(b)

Figure 9.105: Plain co-occurrence trees (a) total time, (b) average time per region.

K: probability list length.



(a)



(b)

Figure 9.106: Enhanced co-occurrence trees (a) move to front rule, (b) counter rule (only the full-dynamic version; static rule was unable to give results).

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

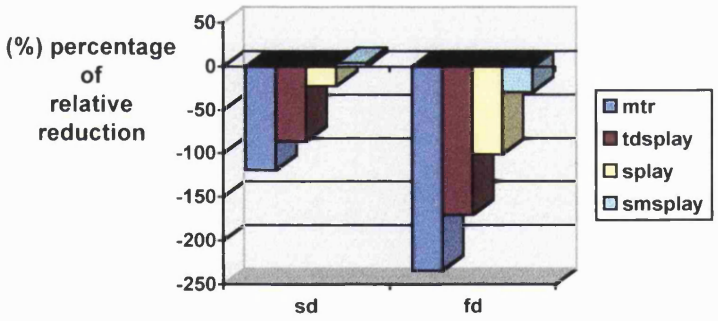
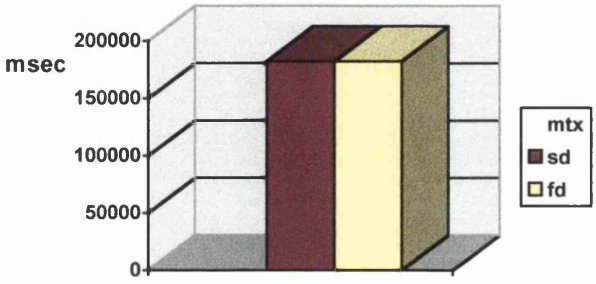


Figure 9.107: Self-adjusting co-occurrence trees.

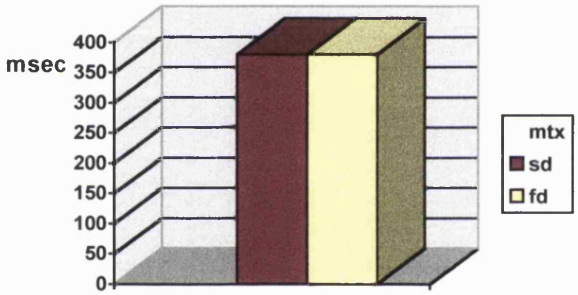
COMPUTATIONAL TIME RESULTS FOR THE CT LUNG IMAGES

mtx: co-occurrence matrix. **sd**: semi-dynamic version.

fd: full-dynamic version.



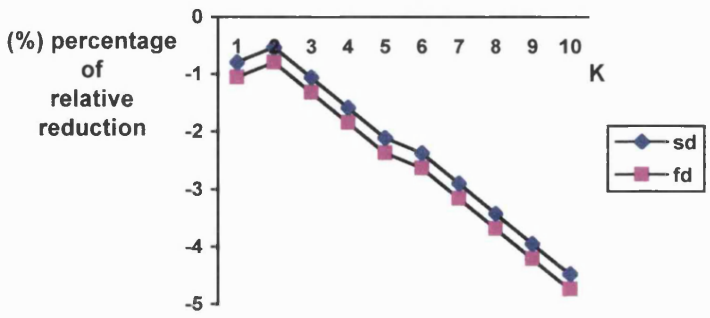
(a)



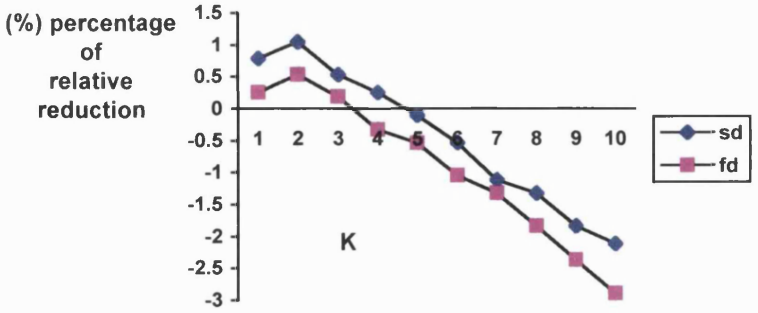
(b)

Figure 9.108: Plain co-occurrence trees (a) total time, (b) average time per region.

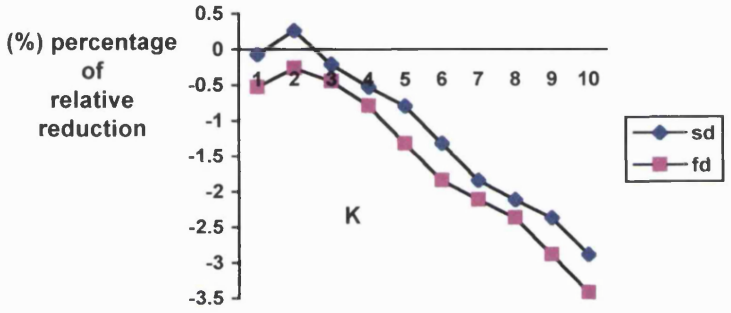
K: probability list length.



(a)



(b)



(c)

Figure 9.109: Enhanced co-occurrence trees (a) static rule, (b) move to front rule, (c) counter rule.

mtr: move to root. **tdsplay**: top-down splaying.

splay: splaying. **smsplay**: semi-splaying.

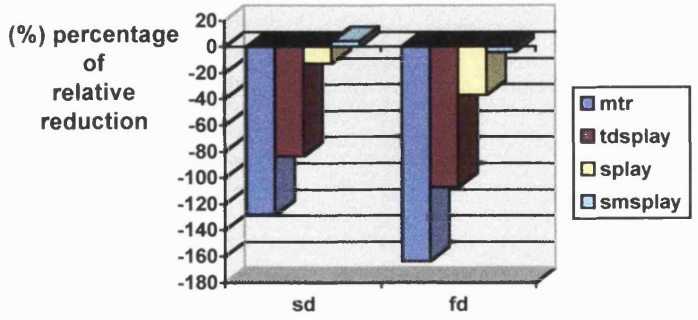


Figure 9.110: Self-adjusting co-occurrence trees.

9.4.6 Sparsity of the co-occurrence matrix and the effect of dynamic range reduction on the textural features of medical images

In this subsection, we give examples of the sparsity of the co-occurrence matrix computed from the analysed image types. This sparsity is in actual fact the reason for the large computational time inefficiency of this approach that was illustrated in the relevant figures. In order to show this sparsity, we selected one region from each image of each data set. For the first category (natural images) the size of the regions was 64×64. The location of the selected regions is illustrated in Figures **9**.14–**9**.18. For the other categories (medical images), the selected regions were among the regions extracted for the estimation of the computational time for each data set. The locations of these regions for each image type are illustrated in Figures **9**.19, **9**.21–**9**.25, **9**.71–**9**.75, and **9**.91–**9**.95. For each of the selected regions we computed the co-occurrence matrix for the $(0, 1)$ displacement vector. Again, the direction of the displacement vector was assumed to be unimportant (see Section **4**.1). Then, the average percentage of zero entries in the computed co-occurrence matrices was estimated for each data set. These percentages are shown in Tables **9**.2 (a)–(d).

Finally, we present examples that show the effect of the reduction in the grey level range of the analysed medical images on the textural features. Our purpose is to indicate that the exploitation of the full dynamic range in the analysis of medical images of various modalities using the SGLDM, may be advantageous. Actually, we repeated the same experiment we performed for the natural images and the mammograms, in order to get an idea of how much of the initial textural detail may be lost, as we reduce the number of grey levels in the ultrasound, the MR, and the CT images.

image type	% zero entries
asphalt	92.04
grass	88.11
fur	89.28
water	91.19
weave	90.12

(a) Natural textures.

image type	% zero entries
mammogram	90.33
heart	92.01
kidney	80.00
liver	88.93
ovaries	81.22
spleen	84.90

(b) Normal mammogram and ultrasounds.

image type	% zero entries
MR bone	$\simeq 100$
MR brain (12-bit)	99.15
MR brain (8-bit)	99.65
MR muscle (12-bit)	$\simeq 100$
MR muscle (15-bit)	$\simeq 100$

(c) MR images.

image type	% zero entries
CT liver (32×32 region)	99.99
CT liver (16×16 region)	99.99
CT brain (16×16 region)	$\simeq 100$
CT brain (32×32 region)	99.99
CT lung	99.71

(d) CT images.

Table **9.2**: Average sparsity of the co-occurrence matrices of the analysed data sets.

The textural features that gave the best feature combinations in the classification of the natural images and the mammograms were employed in this experiment, as well. The variation of the values of the textural features sum of squares, sum average, sum entropy, sum variance, and difference variance was again computed in the form of the percentage of the relative change of the average value of each of the above features (averaged over the four displacement vectors mentioned in Section **9**.3), with respect to the corresponding average value in the initial image, as the number of grey levels was being reduced. The full-dynamic version of the plain co-occurrence trees was employed for the computation of the textural features.

Two images from two data sets of each of the three categories were analysed, namely one ultrasound kidney and one ultrasound spleen image from the ultrasound image category, one MR brain image (8 bit dynamic range) and one MR muscle image (cross-section of the thigh) from the MR image category and finally, one CT liver image (16×16 analysed region) and one CT lung image from the CT image category. The analysed regions are shown in Figures **9**.22, **9**.25, **9**.73, **9**.74, **9**.92, and **9**.95. Similar results were derived from the analysis of other regions in the above images, as well as other images of the same type (e.g. ultrasound kidney images). The relative percentage change was computed for 64, 32, and 16 grey levels in the case of the ultrasound images and the CT lung and MR brain images. All these images initially had a dynamic range of 256 grey levels. In the case of the CT liver and MR muscle image which had an initial dynamic range of 12 bits, the corresponding percentage was computed for 256, 64, 32, and 16 grey levels. Figures **9**.111 (a)–(f) illustrate the relevant percentages. Again, "sos" stands for the sum of squares textural feature, "sa" stands for the sum average, "se" stands for the sum entropy, "sv" stands for the sum variance, and finally, "dv" stands for the difference variance.

Finally, we repeated the above experiment for the MR bone image type. This is a special case because, as we said earlier in this section, it was not possible to compute all thirteen textural features for this image type, since its dynamic range was very high (15 bits). In Figure **9**.112, we illustrate the variation of the values of the textural features contrast (symbolised by "con"), sum of squares (symbolised

by "sos"), sum average (symbolised by "sa"), difference variance (symbolised by "dv"), and first information measure of correlation (symbolised by "fimc") for 256, 64, 32, and 16 grey levels, in the same form as above. The MR bone image as well as the analysed region are shown in Figure **9**.71. The region size was 32×32. Again, similar results were derived from the analysis of other regions extracted from the same image type.

9.4.7 Details of the implementation of the co-occurrence matrix

We conclude this chapter by describing some important points of the implementation of the co-occurrence matrix in the performed computational time experiments. Every effort was made to increase the efficiency of this matrix with respect to the texture feature computation time without increasing the required memory space. In particular, we made the following modifications to the brute force co-occurrence matrix:

1. If G_{min} and G_{max} were the smallest and largest grey levels found in the analysed image, the memory space allocated to the co-occurrence matrix was proportional to their difference, i.e. $N_g = G_{max} - G_{min} + 1$.

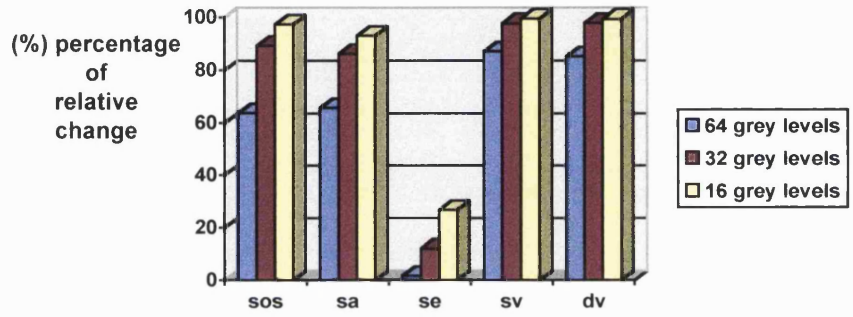
2. If no distinction was necessary between positive and negative distances (see Section **4**.1), i.e. if the co-occurrence matrix was symmetric, then only one triangle (lower left) was actually stored.

3. The zero entries of the co-occurrence matrix, that is, the estimated second order probabilities that were equal to zero, did not take part in the computation of the textural features.

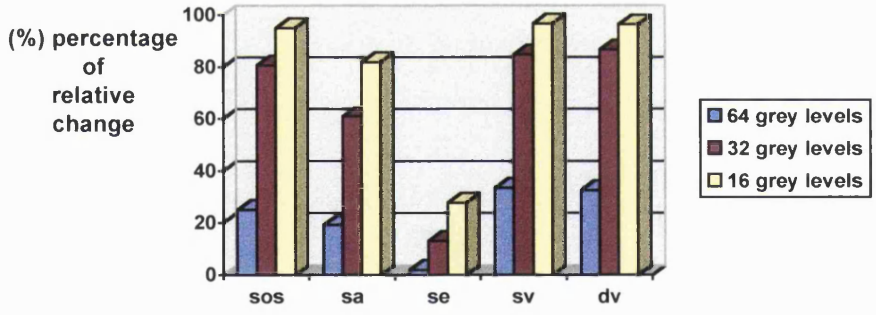
sos: sum of squares. **sa**: sum average.

se: sum entropy. **sv**: sum variance.

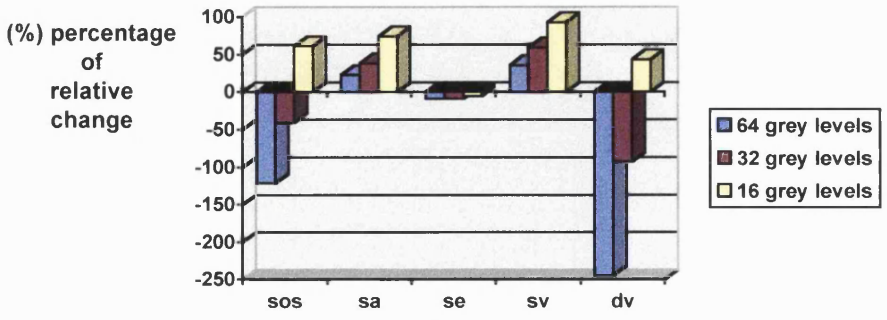
dv: difference variance.



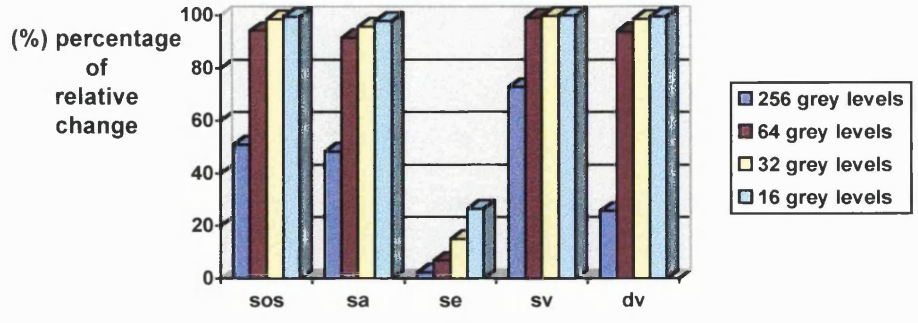
(a)



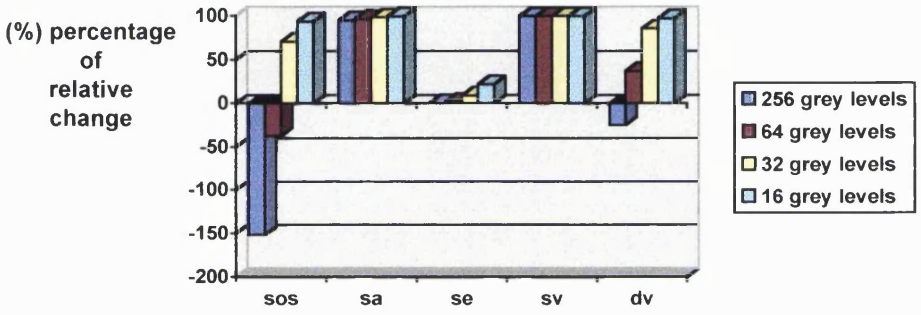
(b)



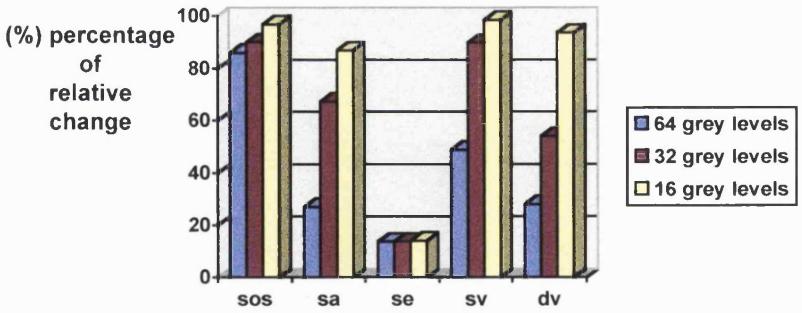
(c)



(d)



(e)



(f)

Figure 9.111: Examples of the percentage of the relative change in the values of the textural features of medical images for various reduced grey level resolutions (a) ultrasound kidney, (b) ultrasound spleen, (c) MR brain, (d) MR muscle, (e) CT liver, (f) CT lung.

con: contrast. **sos**: sum of squares.

sa: sum average. **dv**: difference variance.

fimc: first information measure of correlation.

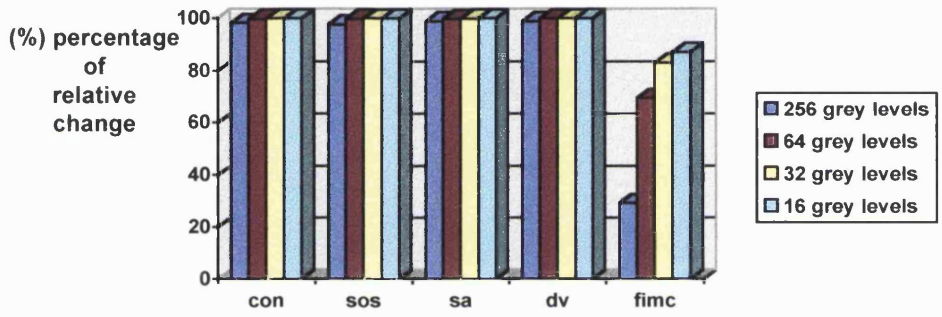


Figure 9.112: Example of the percentage of the relative change in the values of the textural features of the MR bone image type for various reduced grey level resolutions.

Chapter 10

Discussion

10.1 Comparison of the various approaches for the SGLDM in terms of memory space

The memory space requirements of the approaches presented in this thesis for the SGLDM, namely the co-occurrence matrix, the plain co-occurrence trees, the enhanced co-occurrence trees, and the self-adjusting co-occurrence trees were shown in Section **9**.1. The illustrated results allow us to readily compare the semi-dynamic version with the co-occurrence matrix (see Figures **9**.1, **9**.4, **9**.7, **9**.10), the full-dynamic version with the co-occurrence matrix (see Figures **9**.2, **9**.5, **9**.8, **9**.11), and finally, the full-dynamic version with the semi-dynamic version of the proposed approaches (see Figures **9**.3, **9**.6, **9**.9, **9**.12). The memory space results were computed for three dynamic ranges, i.e. 6 bits, 8 bits, and 12 bits. These dynamic ranges cover the majority of the cases encountered in practice, especially in medical image analysis.

The presented memory space requirements are for the worst case (see Section **9**.1). This is, though, a theoretical case. In practice, the number of distinct grey level pairs that satisfy a given displacement vector is much less than n_g^2 (n_g is the number of grey levels in the analysed local region). Therefore, a significant reduction in almost all practical cases, is expected by using the proposed approaches and especially their full-dynamic version. This statement is reinforced by the sparsity of the co-occurrence matrix presented in the form of the average

percentage of zero entries in Tables **9**.2 (a)–(d). The sparsity is due to the dependency of the memory space requirements of the co-occurrence matrix on N_g^2 in all cases (N_g is the number of grey levels in the entire analysed image). The number of grey level pairs, though, that satisfy a given spatial relationship (displacement vector) in an analysed local region, is a very small fraction of the N_g^2 quantity, in practical cases. For the analysed natural textures, the average percentage of zero entries was more than 88%. Similar results were derived from the analysed medical images. For the mammograms, the above percentage was more than 90% while for the ultrasound images, it was more than 80%. Finally, for the MR and CT images this percentage approached 100%.

From the presented worst case memory reduction with respect to the co-occurrence matrix (semi-dynamic and full-dynamic version of the proposed approaches versus co-occurrence matrix), it is clear that the proposed approaches are more efficient in all cases when $n_g \simeq \frac{N_g}{2}$ or less. This corresponds to a sparsity of the co-occurrence matrix of about 77% or more. This lower limit clearly includes all image types analysed in this study (see Tables **9**.2 (a)–(d)). As far as we know, it also includes the majority of the cases encountered in practice.

Figures **9**.3, **9**.6, **9**.9, **9**.12 compare the semi-dynamic version with the full-dynamic version of the proposed approaches. As we said in Section **6**.1, their basic difference is that the full-dynamic version employs trees (dynamic structures), even for the storage of the roots of the trees (the trees that store the estimated second order probabilities) and the heads of the lists. Therefore, this version is expected to be the most efficient in terms of memory space requirements, since it completely eliminates redundancy (zero entries). However, it needs some space for the implementation of these two additional trees (e.g. storage of the relevant pointers), which increases with the number n_g of grey levels in the analysed local region. From the illustrated results, it is clear that improvement is generally achieved when $n_g \simeq \frac{N_g}{4}$ or less. We note also that as N_g increases the full-dynamic version becomes better. In the case of the 12 bit images ($N_g = 2048$) the full-dynamic version is not worse than the semi-dynamic version, even in the case where $n_g = \frac{N_g}{2}$ ($n_g = 1024$), although the semi-dynamic version exhibits a better performance in this case for the 6 bit and 8 bit images. This will become

more clear in Section **10**.3, where we will see that for images with large dynamic ranges only the full-dynamic version could compute the textural features. This was due to the very large number of zero entries the co-occurrence matrix and the semi-dynamic version of the proposed approaches had to store.

Looking at the memory space results of the enhanced co-occurrence trees (Figures **9**.4–**9**.9), it is clear that as parameter K increases the space efficiency of this approach decreases. This is sensible since this parameter is actually the length of the probability lists. More space is needed for the creation of longer lists. Comparing now this approach with the plain co-occurrence trees, the enhanced co-occurrence trees need more space. Since this approach is, in reality, the plain co-occurrence trees with the addition of a number of lists proportional to the number of grey levels in the analysed local region, the memory space needed by the enhanced co-occurrence trees is always more than the space needed by the corresponding plain co-occurrence trees.

Finally, Figures **9**.13 (a) and (b) compare the self-adjusting co-occurrence trees with the plain co-occurrence trees in terms of memory space. As we can clearly see from the graphs, the self-adjusting co-occurrence trees show some improvement. In particular, as the number of grey levels n_g in the analysed region increases, the improvement in memory space using the self-adjusting co-occurrence trees instead of the plain co-occurrence trees increases too, until it reaches its maximum value which is about 10%. From the above graphs, it is also clear that there is no improvement for the semi-dynamic version when $N_g = 2048$ and for the full-dynamic version when $n_g \geq 512$. In order to explain the above, we need to recall that the basic difference between the plain co-occurrence trees and the self-adjusting co-occurrence trees is that in the latter approach there is no need to store any balance information in the nodes of the trees (see Section **5**.5.1). However, for images with a large number of grey levels (512 or larger) the bit used as balance information in each node of the plain co-occurrence trees is actually embedded in the information content of the node (see Section **6**.1). In these cases, no additional space is needed for the balance information and therefore, no improvement results from the employment of the self-adjusting co-occurrence trees instead of the plain co-occurrence trees.

Before we conclude this section, we need to point out that to the best of our knowledge, there is no previous effort to reducing or eliminating the sparsity of the co-occurrence matrix, a part from reducing the dynamic range of the analysed images. Section 4.2 contains an extensive list of references that refer to the sparsity problem of the co-occurrence matrix along with the disadvantages that stem from it as well as the only solution found so far, i.e. the reduction of the grey level range of the analysed textures.

10.2 The advantage of analysing textures in their initial dynamic range

We mentioned in Section 1.4 that in some cases, grey level reduction may be necessary for reasons other than computational time and space efficiency, e.g. noise reduction. However, there is fairly strong indication that the net result of reducing the dynamic range of images, when the SGLDM is employed in texture analysis, is significant information loss in many cases. We saw in Section 4.2 that many researchers have reported this problem.

In this study, in order to show in practice that the reduction in the number of grey levels may lead to a loss of textural information, we performed two classification experiments which have been described in detail in Section 9.3. From the results shown in Tables 9.1 (a) and (b), it is obvious that as the grey level resolution became smaller, the classification accuracy (best classification accuracy achieved) dropped both for the mammograms and the natural images. Especially, in the classification of mammograms into normal and abnormal breast tissue, the accuracy dropped from about 92.2% in the case of the initial 6 bit dynamic range images down to about 79.7%, when 16 grey level images were employed. Notice also that the reduction in the number of grey levels from 64 to 32 led to a classification accuracy reduction of about 10%. In the case of the natural images, the reduction in classification accuracy was smoother. Still, it dropped from 99.06% when using 256 grey levels (the initial dynamic range) to 94.69%, when reducing the number of grey levels to 16. Also, the largest amount of reduction (about

4%) happened when we decreased the number of grey levels from 256 to 64. We therefore note that in both classification experiments, the rate of reduction in classification accuracy was initially large and then it became much smaller, pointing out that for the analysed cases, the initial dynamic range textures contain the most useful information for texture discrimination.

Figures **9**.26 and **9**.27 can give us an idea why this profile in the reduction of the classification accuracy was actually derived, as we decreased the dynamic range of the analysed textures. We can clearly see that the corresponding change in the values of most of the dominant textural features in the classification process was initially large. As we have said in Section **9**.3, these features formed the feature combinations that gave the best classification accuracy in both experiments. From the illustrated graphs, it is also obvious that as we further decreased the number of grey levels, the change in the values of most features was much smaller than the initial change. These graphs, therefore, can adequately explain the observed, initially large and subsequently smaller, reduction in the classification accuracy in the performed experiments.

Finally, similar changes in the values of the textural features mentioned above were derived for images of different modalities. Figures **9**.111 (a)–(f) illustrate the corresponding results from the analysis of two ultrasound, two MR, and two CT images in different dynamic ranges (see Section **9**.4.6). From these graphs, it is clear that reducing the number of grey levels in the analysed images a definite change in the values of the textural features happened for all image types. In the special case of the MR bone image (15 bit image), we note from Figure **9**.112 that a very large change happened in the values of the illustrated textural features, as we reduced the dynamic range to 8 bits. However, further reduction in grey level resolution caused only a small change for most of these features. This is an indication that for this image type, images having a higher than 8 bit dynamic range contain a large amount of textural information, which is lost when the dynamic range is reduced. All the above figures, therefore, indicate that, similarly to the case of the natural textures and mammograms, the classification accuracy will drop as the grey level resolution of the images involved in a classification will become smaller, for all aforementioned image types.

10.3 Comparison of the presented approaches in SGLDM in terms of computational time

In the last part of this chapter, we will discuss the computational time results from the analysis of natural textures (e.g. asphalt, grass, and water) as well as medical images from most imaging modalities (digital X-ray, ultrasound, MR, and CT), employing their full dynamic range. The relevant results are illustrated in two forms which were described in Section **9**.4. Computational time results in the form of the total time and the average time per region are shown for each analysed image type, only for the co-occurrence matrix and the two versions of the plain co-occurrence trees (see Section **9**.4). For the rest of the compared approaches (enhanced and self-adjusting co-occurrence trees), the form of the percentage reduction in the average time per region relative to the corresponding time in the plain co-occurrence trees was preferred. Since it is possible to directly compare the plain co-occurrence trees, which is the basic approach in this thesis, with the co-occurrence matrix, we are more interested to know the improvement of the other proposed approaches over the plain co-occurrence trees. Of course, it is always feasible to indirectly compare them to the co-occurrence matrix through their comparison with the plain co-occurrence trees.

Before we discuss the computational time results presented in Section **9**.4, in greater detail, we need to emphasise the fact that the aim of the performed experiments was not to show that the time results derived from the analysed image types follow precisely the theoretical lower bounds proved in the corresponding theorems of Sections **6**.2, **7**.2, and **8**.2. The theoretical analysis presented there proves that by applying the proposed approaches, a decrease in computational time should be achieved compared to the co-occurrence matrix, and this decrease is independent of any particular computing system employed for texture analysis. It also gives insights on how the proposed data structures work internally, which help the reader understand why the above reduction in computational time actually happened, which is discussed in the following sections.

10.3.1 Plain co-occurrence trees

In Figures **9**.28, **9**.33, **9**.38, **9**.43, and **9**.48, the total time and the average time per region are illustrated for the asphalt, grass, fur, water, and weave image types. As we said in Section **9**.4, 3 different sizes were employed for the analysed regions, namely 16×16, 32×32, and 64×64, which covered the whole analysed image of size 128×128. Although the initial dimensions of each image were 512×512, we preferred to extract and analyse a subimage of dimensions 128×128, because the computational time that was needed for the analysis of a 512×512 image using the co-occurrence matrix, was unacceptably large, especially for a 16×16 region. The above region sizes have been widely used in the analysis of natural textures. Moreover, by varying the size of the analysed regions we can draw useful conclusions about the way the region size affects the time performance of the compared approaches. The detailed computational time results are presented in the appendix.

When we compare the co-occurrence matrix with the semi-dynamic version of the plain co-occurrence trees based on the computational time results, it is clear that the latter approach is much better. The co-occurrence matrix was not able to produce any results in three of the five analysed natural texture types. Only in one case did it give better time than the semi-dynamic version. That was for the grass image type and the 64×64 region size. However, the difference was very small (about 0.67 min in total time).

Comparing now the co-occurrence matrix with the full-dynamic version, we note that the latter approach exhibited better time behaviour except from two cases. For the asphalt and grass image types and for the 64×64 region size, the co-occurrence matrix achieved the best time between these two approaches. Finally, comparing the semi-dynamic version with the full-dynamic version we note that the first approach showed a better time behaviour in all analysed cases.

The worst time behaviour of the co-occurrence matrix was due to its large sparsity. A lot of its entries were zero increasing the computational time and memory space requirements, without actually contributing to the computation of the textural features. Due to this large redundancy, in the majority of the

analysed natural textures this approach couldn't produce any results at all.

One would expect that the full-dynamic version would give the best time results, since it completely eliminates the redundant zero entries. The fact is, however, that maintaining a BB-tree has overhead (rebalancing cost) and also the access and insert operations cost logarithmic time (see Section **6**.1). The reader should recall at this point that the only difference between the semi-dynamic and the full-dynamic version is that the latter employs two additional BB-trees, instead of static arrays, for the storage of the root of the trees and the heads of the lists. Obviously, the zero entries in the root and list array of the semi-dynamic version were not so many as to increase the computational time more than the overhead for maintaining the root and list tree of the full-dynamic version. This overhead, along with the logarithmic access and insert cost in the above additional trees, were responsible for the inferior performance of the full-dynamic version in the analysis of the natural textures.

Finally, we note from the presented results that as the size of the analysed regions increased, the time difference between the co-occurrence matrix approach and the plain co-occurrence trees decreased. The main reason is that the number of redundant zero entries becomes smaller as the region size increases (the co-occurrence matrix becomes less sparse). Therefore, the co-occurrence matrix approach becomes more efficient. Also, the number of the distinct grey level pairs (i, j) that satisfy a given displacement vector increases as well. This factor is very important for the performance of the proposed approaches, because it determines the depth of each BB-tree T_i and consequently, the access and insert cost (see Section **6**.1) as well as the rebalancing cost (see Section **5**.3). The larger the number of insertions in a BB-tree the larger the probability that rebalancing of the tree will be needed. The above explain why the computational time of the co-occurrence matrix became smaller than the corresponding time of the semi-dynamic version in one case, where the region size was 64×64.

In Figures **9**.53, **9**.56, **9**.59, **9**.62, **9**.65, and **9**.68, the total time and the average time per region are illustrated for the normal mammograms and the ultrasound image types (heart, kidney, liver, ovaries, and spleen). The detailed computational time results are presented in the appendix. The following observations can

be made regarding the analysis of the above image types. First, the co-occurrence matrix exhibited the worst behaviour. In comparison with the semi-dynamic version the time difference was large, as it is also clear from the illustrated results. For example, the difference in total time for the analysis of the normal mammograms was about 36 min. The corresponding time difference for the analysis of the ultrasound heart images was about 3.11 min. The full-dynamic version scored in between the co-occurrence matrix and the semi-dynamic version, in all analysed cases. In all but one case it was closer to the semi-dynamic version in terms of time performance. Only in the case of the ovaries image type was the full-dynamic version closer to the co-occurrence matrix.

In Figures **9**.76, **9**.79, **9**.82, **9**.85, and **9**.88, the total time and the average time per region are shown for the MR image types. Again, the detailed computational time results are shown in the appendix. From these results, we note that the **co-occurrence matrix**, in the majority of the cases, was **not able** to produce any results. That happened for the **MR femur** and the **MR skeletal muscle** images (both **12 bit** and **15 bit** image type). It only gave results for the two brain image types. The **semi-dynamic** version of the plain co-occurrence trees was **not able** to produce any results for two of the analysed MR image types, namely the **MR femur** image type and the **MR skeletal muscle** image type (**15 bit** dynamic range). The **full-dynamic** version was the only approach that gave results for all analysed MR image types.

From the presented results, it is clear that the full-dynamic version presented the best overall time performance in the analysis of the MR image types. The semi-dynamic version was better in two cases, namely the two brain image types (12 bit and 8 bit dynamic range). However, the full-dynamic version was close (see Figures **9**.79 and **9**.82). The co-occurrence matrix was the worst of all approaches. It was not able to produce any results in the case of three image types, while for the rest, it had a large time difference from the plain co-occurrence trees. From the semi-dynamic version, it had about 3.49 min difference in total time for the brain image type (12 bit dynamic range) and about 1.69 min difference in total time for the brain image type (8 bit dynamic range). The reason for this worst behaviour was the high dynamic range of the analysed images. Two of the image

types had a 12 bit dynamic range while another two had a 15 bit dynamic range. Therefore, the sparsity of the co-occurrence matrix was very high. Indeed, it approached 100% (see Table **9**.2 (c)). For the same reason, the root and list array of the semi-dynamic version had a lot of redundant zero entries. The results from the analysis of the skeletal muscle image type (12 bit dynamic range) are very characteristic of the inefficiency of the semi-dynamic version in such cases. The time difference between this version and the full-dynamic version was about 1.9 h in total time.

Finally, in Figures **9**.96, **9**.99, **9**.102, **9**.105, and **9**.108 the total time and average time per region are shown for the CT image types. The detailed computational time results are shown in the appendix. For these image types, the **co-occurrence matrix** was **not able** to produce any results. The same observations can be made here as in the case of the MR images. The dependence of the co-occurrence matrix on the number of grey levels in the entire analysed image resulted in its inability to produce any results at all. Similarly, the high dynamic range (12 bit) in four of the analysed CT image types resulted in a very large number of zero entries in the root and list array of the semi-dynamic version. This fact explains its large time inefficiency compared to the full-dynamic version. Only for the lung image type, the dynamic range of which was 8 bits, the two versions (semi-dynamic and full-dynamic) exhibited the same behaviour in terms of the computational time. Therefore, we can say that the full-dynamic version gave the best overall time performance for the CT images, too.

10.3.2 Enhanced co-occurrence trees

The percentages of the improvement in the average time per region for the semi-dynamic and the full-dynamic version of the enhanced co-occurrence trees relative to the corresponding versions of the plain co-occurrence trees, were presented in Sections **9**.4.1, **9**.4.2, **9**.4.3, **9**.4.4, and **9**.4.5. The following observations can be made.

For the **static** rule, we note that only in two cases, namely the weave image type and the ultrasound kidney image type, did it exhibit some improvement

over the plain co-occurrence trees (see also the appendix). Its performance was also **very close** to the performance of the plain co-occurrence trees, in the case of the **ultrasound heart** image type, the **ultrasound spleen** image type, and the **MR brain** image type (**8 bit** dynamic range). This rule was **not able** to give any results at all, in the case of the **MR femur** image type, the two **MR skeletal muscle** image types (**12 bit** and **15 bit** dynamic range), the **CT liver** image types ($\mathbf{32 \times 32}$ and $\mathbf{16 \times 16}$ region size), and the **CT brain** image types ($\mathbf{16 \times 16}$ and $\mathbf{32 \times 32}$ region size).

The above results indicate that the Gaussian model, which was employed in this study for the static rule (see Section **7**.1.1), did not give a good description of the co-occurrence distribution for the majority of the analysed images. Only in two image types this model was able to predict successfully the grey level pairs with the largest co-occurrence probabilities (weave and ultrasound kidney image type). Also, the static rule failed to give any results for most of the high dynamic range image types. This was due to its requirement to load in advance in main memory the estimated co-occurrence probabilities for all grey levels in the analysed image.

In the case of the **move to front** rule, we can clearly see from the illustrated results and the detailed time results presented in the appendix that there is an improvement using the **semi-dynamic** version, for the majority of the analysed image types. However, the semi-dynamic version was **not able** to produce any results in the case of the **MR femur** image type, the two **MR skeletal muscle** image types (**12 bit** and **15 bit** dynamic range), the **CT liver** image types ($\mathbf{32 \times 32}$ and $\mathbf{16 \times 16}$ region size), and the **CT brain** image types ($\mathbf{16 \times 16}$ and $\mathbf{32 \times 32}$ region size). Only the **full-dynamic** version could compute the textural features in the above cases, but there was **no significant improvement** in time compared to the corresponding version of the plain co-occurrence trees, except from one case. For the **MR skeletal muscle** image type (**12 bit** dynamic range), the full-dynamic version gave an improvement of 0.94% for $\mathbf{K = 3}$. It corresponds to a reduction of about 4.5 sec in total time. The reason for the above behaviour of the semi-dynamic version was for one more time the static nature of the root and list array, which translates into their dependence on the number of grey levels

in the whole analysed image.

In summary, the computational time results show that, although there was an improvement in time over the plain co-occurrence trees by employing the move to front rule, this was not very large for all analysed cases. Only for the asphalt, grass, ultrasound heart, ultrasound ovaries, and MR brain image types (12 bit and 8 bit dynamic range) did the move to front rule give an improvement of 2% or more. In these cases, the probability lists offered some speedup in the computation of the textural features in the SGLDM up to about 4%.

Finally, regarding the **counter** rule we can see from the illustrated percentages that there were a number of cases where an improvement actually happened using the **semi-dynamic** version (see also the detailed time results in the appendix). Again, the semi-dynamic version **failed** to produce any results in the case of the high dynamic range image types, namely the **MR femur** image type, the two **MR skeletal muscle** image types (**12 bit** and **15 bit** dynamic range), the **CT liver** image types ($\mathbf{32 \times 32}$ and $\mathbf{16 \times 16}$ region size), and the **CT brain** image types ($\mathbf{16 \times 16}$ and $\mathbf{32 \times 32}$ region size). Only the **full-dynamic** version was able to produce results in the above cases, but there was **no improvement** compared to the corresponding version of the plain co-occurrence trees.

Compared to the move to front rule, the counter rule was inferior regarding the computational time, in all analysed cases. The immediate conclusion is that sorting the grey levels stored in the probability lists in descending order, according to their access frequencies (see Section **7**.1.3) does not offer any improvement over the move to front rule. The time needed for this sorting dominates the total time for the computation of the textural features and makes any advantage stemming from it vanish.

In general, the use of the probability lists in the enhanced co-occurrence trees yielded some improvement over the plain co-occurrence trees. The move to front rule for updating the content of these lists was the superior rule, which agrees with the theoretical results found in the literature (see Section **5**.4). We also note that the best performance was exhibited for two values of the parameter K ($K = 2, 3$). The existence of an optimal range for the length of the probability lists, in terms of the time needed for the computation of the textural features, was

predicted theoretically in Section 7.2. It is clear that the experimental time results from the performed analysis of natural textures as well as medical images of various modalities, verify these theoretical findings. Finally, we observe from the illustrated graphs that as the length of the probability lists increases past the optimal length, the time behaviour of the enhanced co-occurrence trees becomes much worse. This is because the time needed to perform the linear search in these lists in order to find the requested grey levels, dominates the total computational time.

10.3.3 Self-adjusting co-occurrence trees

We conclude this section by discussing the computational time results from the self-adjusting co-occurrence trees. The corresponding improvements over the plain co-occurrence trees are shown in Sections 9.4.1, 9.4.2, 9.4.3, 9.4.4, and 9.4.5, separately for each of the four employed techniques (move to root, top-down splaying, splaying, and semi-splaying), for both the semi-dynamic and the full-dynamic version.

From the illustrated graphs, it is clear that the only technique that showed some improvement was the semi-splaying technique. The move to root technique gave the worst time results as expected (see Section 8.2). The time for a single access operation in the trees increased due to the fact that the binary trees were not balanced. The consequence was that the total computational time increased substantially. Similar behaviour was exhibited by the top-down splaying technique. The overhead from performing splaying at the same time that the trees were traversed from the root down to the leaves during an access operation, increased the computational time so that the performance of the self-adjusting co-occurrence trees was much worse than the performance of the plain co-occurrence trees, in the case of the top-down splaying.

The situation was improved by employing the **bottom-up splaying** (normal splaying). Here, the overhead for performing the splaying operation was not so much as in the case of the top-down splaying. Still, the computational time performance was inferior to the corresponding performance of the plain co-occurrence trees. Only in one analysed case this technique exhibited better behaviour. For

the **MR skeletal muscle** image type (**15 bit** dynamic range), the **full-dynamic** version of the self-adjusting co-occurrence trees employing the splaying technique showed an improvement of 2% compared to the corresponding version of the plain co-occurrence trees. This percentage corresponds to a reduction of about 1.6 sec in total time. The **semi-splaying** technique was the best of all compared techniques in the self-adjusting co-occurrence trees approach. Actually, it showed an improvement over the plain co-occurrence trees for the cases shown in the appendix.

As we said in Section 8.1, the goal in the case of the self-adjusting co-occurrence trees is to take advantage of their ability to adapt their structure to the co-occurrence distribution in the analysed texture, in order to speed up the computation of the textural features. However, as it is clear from the time performance of the first three techniques, the overhead from moving the accessed elements all the way up to the root of the trees dominates the computational time and makes this approach inferior to the plain co-occurrence trees when these techniques are employed.

The success of the semi-splaying technique is based on trading off worse adaptation to the access distribution against the decrease in the overhead. In other words, this technique moves the accessed grey levels only partway towards the root of the trees. Thus, the amount of required restructuring is reduced. However, it may not hold any more that the most frequently accessed elements are found nearer the root of the trees (the entry point of an access operation). As we have already said, semi-splaying has many variants. The proposed variant in this thesis takes into account the texture information stored in the nodes of the trees in the form of the estimated second order probabilities, in order to minimise the consequences of the above worse adaptation (see Section 8.1). The presented time results show that this variant indeed works well in practice.

Finally, similarly to the plain and enhanced co-occurrence trees, the **semi-dynamic** version of the self-adjusting co-occurrence trees **failed** to compute the textural features in some analysed cases, namely the **MR femur** image type and the **MR skeletal muscle** image type (**15 bit** dynamic range). The reason was again the semi-dynamic nature of this version, i.e. the employment of static

arrays instead of dynamic trees for the storage of the roots of the self-adjusting trees and the heads of the lists.

Chapter 11

Conclusions & Future work

11.1 Conclusions

The main aim of this thesis was to find efficient ways to overcome the disadvantages in terms of memory space and computational time, that result from the utilisation of the co-occurrence matrix in the analysis of images employing the spatial grey level dependence method and to evaluate them both theoretically and through their application to the analysis of natural textures and medical images of various modalities. Our objective was also to make the SGLDM applicable to the analysis of a wider range of image types, especially medical image types that cannot be analysed using the co-occurrence matrix and which may contain important information that may lead to better texture discrimination.

The approaches proposed in this thesis for organising the textural information extracted from an image region by the SGLDM in the form of the estimated second order probabilities, were shown to be superior to the co-occurrence matrix in overall performance. In almost all analysed cases, which included most of the dynamic ranges encountered in practice, especially in medical image analysis, the co-occurrence trees showed better performance in terms of both memory space and computational time. The reason for this superior performance is their ability to eliminate the large number of zero entries of the co-occurrence matrix, which are redundant, contributing nothing to the computation of the textural features in the SGLDM. In this way, co-occurrence trees allow the same textural features

to be computed much faster. This is very important since the computational time has been reported to be a critical factor for many real clinical applications.

By eliminating the redundancy of the co-occurrence matrix, co-occurrence trees also allow the full dynamic range of the images to be exploited in texture analysis, classification, and segmentation, using the SGLDM texture analysis method, which is considered to be one of the best. With the employment of the co-occurrence trees, there is no longer a need to reduce the number of grey levels in the analysed image. This pre-processing step is necessary in the majority of the cases, when the co-occurrence matrix is employed, but may lead to a significant loss of textural information. This information loss may be very important, especially in medical image analysis, because it may reflect subtle differences in tissue appearance, vital for automatic discrimination. We saw in two classification experiments that analysing natural textures and mammograms in their initial dynamic range led to improved classification accuracy. We have fairly strong indications that the same holds for other image types as well, especially for the high dynamic range medical images. It is therefore expected that the use of the co-occurrence trees instead of the co-occurrence matrix in the SGLDM, will greatly improve classification accuracy and image segmentation results in a very large number of cases.

Comparing the different proposed kinds of the co-occurrence trees, we concluded that the enhanced co-occurrence trees and especially the move to front rule, showed the best overall time performance, since this approach improved the computational time relative to the plain co-occurrence trees, for the majority of the analysed cases. However, its memory space requirements were the worst, because of the employment of the probability lists. Due to these lists, the memory space needed by the enhanced co-occurrence trees is always more than the corresponding space of the plain co-occurrence trees. The additional memory space required by the probability lists depends on the number of grey levels in the analysed local region as well as their length (parameter K).

Parameter K is a very important factor which, as we saw, affects the overall performance of the enhanced co-occurrence trees. Longer lists increase the probability of finding the requested grey level in them and thus, they decrease the

probability of accessing the BB-trees, which is more expensive in terms of time. However, they also lead to increased access time for the requested grey levels that are not stored in them, due to the linear search. We saw that there is a range of values of the list length, for which the enhanced co-occurrence trees approach exhibits the best behaviour (optimal range). In the performed analysis of real textures, the optimal values of K were found to be 2 and 3. Parameter K should be kept small in all practical applications.

Finally, the self-adjusting co-occurrence trees employing the semi-splaying technique, yielded an improvement over the plain co-occurrence trees in only 9 out of the 21 analysed image types. However, the memory space requirements of this approach were the least, although close to the corresponding requirements of the plain co-occurrence trees.

Overall, if memory demands are not a serious problem for the system used for texture analysis, the enhanced co-occurrence trees employing the move to front rule appear to be the best approach for most analysed image types, in terms of computational time. Otherwise, the plain co-occurrence trees should be preferred, since their time performance is very close to that of the enhanced co-occurrence trees, but their memory requirements are much less. If, however, the maximum efficiency in terms of memory utilisation is required, the self-adjusting co-occurrence trees employing the semi-splaying technique should be generally used.

Finally, comparing the two versions of the co-occurrence trees, we found that the semi-dynamic version is much faster in the analysis of images having a dynamic range of 8 bits or less. Therefore, this version is recommended for the analysis of these image types. On the other hand, the full-dynamic version showed a better overall performance in the analysis of higher dynamic range images ($>$ 8 bits dynamic range), where the semi-dynamic version was either much slower, or led the system to run out of memory. The full-dynamic version appears to be the ideal solution for the analysis of these images. Since a lot of medical images belong to this category, the importance of this particular version of the co-occurrence trees in medical image analysis should be clear. The above conclusions can be summarised as follows:

- The approaches proposed in this thesis were shown to be superior to the co-occurrence matrix in both memory space and computational time.

- The proposed approaches enable the analysis of images employing their full dynamic range.

- The classification accuracy improves when the full-dynamic range is employed in the classification of natural textures and mammograms.

- In the case of the enhanced co-occurrence trees approach, the move to front rule showed the best time performance.

- The length of the probability lists in the enhanced co-occurrence trees (parameter K) was shown to be a very important factor which affects the performance of this approach. Both the theoretical and experimental results indicated that there exists a range of values of the list length, for which the enhanced co-occurrence trees exhibit the best behaviour (optimal range).

- Semi-splaying was found to be the superior technique in the case of the self-adjusting co-occurrence trees approach.

- Choosing the most appropriate approach and version for a specific application depends on the dynamic range of the analysed images as well as the available memory in the computing system employed for texture analysis.

11.2 Future work

In the following, we will present some ideas for further extending the research work presented in this thesis. As we said earlier, one of the big advantages in employing the co-occurrence trees instead of the co-occurrence matrix in the SGLDM, is their ability to analyse textures in their initial dynamic range. This potentially leads to the application of this powerful texture analysis method to a much broader range of image types, especially in medical image analysis where high dynamic range images are often encountered. We saw that there is fairly strong indication, that the employment of the entire grey level range of an image type may lead

to better texture discrimination. In particular, we saw that in both performed classification experiments (natural textures and mammograms), the classification accuracy achieved by employing the initial dynamic range of the images was improved. Especially, in the case of the mammograms the improvement was significant. Automated medical diagnosis is particularly expected to get the most benefit from any improvement that may arise from the utilisation of the full dynamic range of the analysed images.

Julesz conjectured in [80] that the human visual system cannot discriminate between two textures having the same second-order statistics. The SGLDM is one of the texture analysis methods based on the significance of second-order statistics for texture discrimination. The fact that two textures have identical second-order statistics does not imply that their higher-order statistics (third or fourth) are identical. On the contrary, two textures can have different third or fourth-order statistics, but the same first and second-order statistics. In this case, textures may not be discriminated visually, but that doesn't mean that they are the same.

Several researchers have investigated higher-order statistics in texture analysis ([118], [137], [138]). One of the main reasons that there is not much research work in this area, is the fact that these statistics need a very large amount of memory space. In [118], it was implied that the wider employment of higher-order co-occurrence matrices is prevented by their large size. This is again caused by the large number of redundant zero entries in these matrices. As the order of co-occurrence increases, the number of zero entries increases substantially. The consequence is that a very large amount of memory space is wasted and the computational time for the texture features is prohibitive. In [118], third and fourth-order co-occurrence matrices were used in the classification of natural textures. The authors reported a higher classification accuracy by employing higher-order co-occurrences. However, in order to handle their large memory requirements, they had to reduce the number of grey levels in the analysed images. They used 14 grey levels in the classification employing third-order statistics and only 7 grey levels, when they employed fourth-order statistics. The grey level range of the original images was 256. Reducing, though, the number of grey levels so much

led to a significant information loss. The classification accuracy might be much better if they could exploit as much of the original grey level range as possible.

The co-occurrence trees can be used to overcome the memory space problems, through the design and implementation of their higher-order counterparts. Especially, the full-dynamic version, which can completely eliminate the redundant zero entries, can be employed to store the estimated higher-order probabilities from an analysed region with maximum efficiency. Higher-order statistics are expected to have a great value, especially in medical image analysis. The reason is that different textures in medical images cannot always be discriminated visually. Therefore, we might have to rely on higher than second-order statistics, in order to distinguish between normal and abnormal tissue or specify the type of abnormality at early stages, since differences in tissue appearance might be very subtle. Second-order statistics cannot always capture these differences. The employment of higher-order co-occurrence trees may help the scientific community to have a closer look at these subtle differences and achieve even better classification rates and segmentation results, in many more cases.

Finally, in Section **5**.3 we mentioned that one of the properties of the BB-trees (the basic building blocks of the plain and enhanced co-occurrence trees) is the possibility of top-down rebalancing. We mentioned that this kind of rebalancing from the root down to the leaves is very useful in a parallel processing environment. This actually means that it is possible to design and implement a parallel version of the co-occurrence trees. That may greatly reduce the computational time needed to perform the texture analysis task by sharing the workload among a number of processors. If, for example, we assume that the processing system has n_g processing elements (n_g is the number of grey levels in the local analysed region), which is not unreasonable given the scale of current massively parallel processors, the time complexity of the $p_x(i)$ and $p_y(j)$ elementary operations can be reduced to $O(1)$ (see Lemmas **6**.2.2 and **6**.2.3). Top-down rebalancing may be the key to the parallel implementation of the co-occurrence trees, since it allows more concurrency in accessing a BB-tree (more processes can act in parallel on a tree).

Appendix A

Detailed time results

A.1 Plain co-occurrence trees

A.1.1 Natural textures

The **co-occurrence matrix** gave the following time results:

- For the **asphalt** image type, it gave:
  - about 33.47 min in total time and 0.78 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 14.89 min in total time and 1.40 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 7.63 min in total time and 2.86 sec in the average time per region for the **64** $\times$ **64** region size.
- For the **grass** image type, it gave:
  - about 36.99 min in total time and 0.87 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 14.77 min in total time and 1.38 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 7.84 min in total time and 2.94 sec in the average time per region for the **64** $\times$ **64** region size.

- For the **fur** image type, it gave **no results**.
- For the **water** image type, it gave **no results**.
- For the **weave** image type, it gave **no results**.

The **semi-dynamic** version of the plain co-occurrence trees gave the following time results:

- For the **asphalt** image type, it gave:
  - about 6.36 min in total time and 0.15 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 7.82 min in total time and 0.73 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 6.69 min in total time and 2.51 sec in the average time per region for the **64** $\times$ **64** region size.
- For the **grass** image type, it gave:
  - about 8.15 min in total time and 0.19 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 10.18 min in total time and 0.95 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 8.51 min in total time and 3.19 sec in the average time per region for the **64** $\times$ **64** region size.
- For the **fur** image type, it gave:
  - about 25.61 min in total time and 0.6 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 29.23 min in total time and 2.74 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 28.69 min in total time and 10.76 sec in the average time per region for the **64** $\times$ **64** region size.

- For the **water** image type, it gave:
  - about 13.67 min in total time and 0.32 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 20.06 min in total time and 1.88 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 19.49 min in total time and 7.31 sec in the average time per region for the **64** $\times$ **64** region size.
- For the **weave** image type, it gave:
  - about 23.03 min in total time and 0.54 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 33.92 min in total time and 3.18 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 28.80 min in total time and 10.80 sec in the average time per region for the **64** $\times$ **64** region size.

The **full-dynamic version** of the plain co-occurrence trees gave the following time results:

- For the **asphalt** image type, it gave:
  - about 10.81 min in total time and 0.25 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 10.68 min in total time and 1 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 9.21 min in total time and 3.45 sec in the average time per region for the **64** $\times$ **64** region size.
- For the **grass** image type, it gave:
  - about 16.73 min in total time and 0.39 sec in the average time per region for the **16** $\times$ **16** region size,

- about 15.65 min in total time and 1.47 sec in the average time per region for the **32** $\times$ **32** region size,
- about 11.95 min in total time and 4.48 sec in the average time per region for the **64** $\times$ **64** region size.

- For the **fur** image type, it gave:
  - about 46.94 min in total time and 1.1 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 45.76 min in total time and 4.29 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 37.65 min in total time and 14.12 sec in the average time per region for the **64** $\times$ **64** region size.

- For the **water** image type, it gave:
  - about 27.71 min in total time and 0.65 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 32.85 min in total time and 3.08 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 25.68 min in total time and 9.63 sec in the average time per region for the **64** $\times$ **64** region size.

- For the **weave** image type, it gave:
  - about 51.64 min in total time and 1.21 sec in the average time per region for the **16** $\times$ **16** region size,
  - about 57.50 min in total time and 5.39 sec in the average time per region for the **32** $\times$ **32** region size,
  - about 39.57 min in total time and 14.84 sec in the average time per region for the **64** $\times$ **64** region size.

A.1.2 Normal mammograms and ultrasound image types

The **co-occurrence matrix** gave the following time results:

- about 225.31 min in total time and 3.52 sec in the average time per region for the **normal mammograms**,
- about 3.78 min in total time and 0.28 sec in the average time per region for the **ultrasound heart** image type,
- about 3.86 min in total time and 0.29 sec in the average time per region for the **ultrasound kidney** image type,
- about 6.05 min in total time and 0.45 sec in the average time per region for the **ultrasound liver** image type,
- about 1.18 min in total time and 0.22 sec in the average time per region for the **ultrasound ovaries** image type,
- about 3.77 min in total time and 0.28 sec in the average time per region for the **ultrasound spleen** image type.

The **semi-dynamic** version of the plain co-occurrence trees gave the following time results:

- about 189.42 min in total time and 2.96 sec in the average time per region for the **normal mammograms**,
- about 40.48 sec in total time and 51 msec in the average time per region for the **ultrasound heart** image type,
- about 1.51 min in total time and 0.11 sec in the average time per region for the **ultrasound kidney** image type,
- about 2.56 min in total time and 0.19 sec in the average time per region for the **ultrasound liver** image type,
- about 35.46 sec in total time and 0.11 sec in the average time per region for the **ultrasound ovaries** image type,

- about 1.63 min in total time and 0.12 sec in the average time per region for the **ultrasound spleen** image type.

The **full-dynamic** version of the plain co-occurrence trees gave the following time results:

- about 197.13 min in total time and 3.08 sec in the average time per region for the **normal mammograms**,
- about 1.48 min in total time and 0.11 sec in the average time per region for the **ultrasound heart** image type,
- about 2.16 min in total time and 0.16 sec in the average time per region for the **ultrasound kidney** image type,
- about 4.28 min in total time and 0.32 sec in the average time per region for the **ultrasound liver** image type,
- about 1.17 min in total time and 0.22 sec in the average time per region for the **ultrasound ovaries** image type,
- about 2.16 min in total time and 0.16 sec in the average time per region for the **ultrasound spleen** image type.

A.1.3 MR images

The **co-occurrence matrix** gave the following time results:

- For the **MR brain** images having a **12 bit** dynamic range, the co-occurrence matrix gave about 3.98 min in total time and 0.5 sec in the average time per region.
- For the **MR brain** images having an **8 bit** dynamic range, it gave about 2.16 min in total time and 0.27 sec in the average time per region.
- For the **MR femur** images, it gave **no results**.
- For the **MR skeletal muscle** images having a **12 bit** dynamic range, it gave **no results**.

- For the **MR skeletal muscle** images having a **15 bit** dynamic range, it gave **no results**.

The **semi-dynamic** version of the plain co-occurrence trees gave the following time results:

- about 29.38 sec in total time and 61 msec in the average time per region for the **MR brain** image type (**12 bit** dynamic range),
- about 27.94 sec in total time and 58 msec in the average time per region for the **MR brain** image type (**8 bit** dynamic range),
- about 122.67 min in total time and 4.6 sec in the average time per region for the **MR skeletal muscle** image type (**12 bit** dynamic range),
- **no results** for the **MR femur** image type,
- **no results** for the **MR skeletal muscle** images having a **15 bit** dynamic range.

The **full-dynamic version** of the plain co-occurrence trees gave the following time results:

- about 7.54 min in total time and 0.63 sec in the average time per region for the **MR femur** image type,
- about 46.66 sec in total time and 97 msec in the average time per region for the **MR brain** image type (**12 bit** dynamic range),
- about 33.55 sec in total time and 70 msec in the average time per region for the **MR brain** image type (**8 bit** dynamic range),
- about 8.53 min in total time and 0.32 sec in the average time per region for the **MR skeletal muscle** image type (**12 bit** dynamic range),
- about 1.33 min in total time and 50 msec in the average time per region for the **MR skeletal muscle** image type (**15 bit** dynamic range).

A.1.4 CT images

The **co-occurrence matrix** gave the following time results:

- For the **CT liver** image type (**32** $\times$ **32** region size), it gave **no results**.
- For the **CT liver** image type (**16** $\times$ **16** region size), it gave **no results**.
- For the **CT brain** image type (**16** $\times$ **16** region size), it gave **no results**.
- For the **CT brain** image type (**32** $\times$ **32** region size), it gave **no results**.
- For the **CT lung** image type, it gave **no results**.

The **semi-dynamic** version of the plain co-occurrence trees gave the following time results:

- about 21.59 min in total time and 2.7 sec in the average time per region for the **CT liver** image type (**32** $\times$ **32** region size),
- about 19.26 min in total time and 2.41 sec in the average time per region for the **CT liver** image type (**16** $\times$ **16** region size),
- about 103.68 min in total time and 12.96 sec in the average time per region for the **CT brain** image type (**16** $\times$ **16** region size),
- about 71.52 min in total time and 8.94 sec in the average time per region for the **CT brain** image type (**32** $\times$ **32** region size),
- about 3.04 min in total time and 0.38 sec in the average time per region for the **CT lung** image type.

The **full-dynamic version** of the plain co-occurrence trees gave the following time results:

- about 2.93 min in total time and 0.37 sec in the average time per region for the **CT liver** image type (**32** $\times$ **32** region size),
- about 39.89 sec in total time and 83 msec in the average time per region for the **CT liver** image type (**16** $\times$ **16** region size),

- about 2.72 min in total time and 0.34 sec in the average time per region for the **CT brain** image type (**16** $\times$ **16** region size),

- about 6.16 min in total time and 0.77 sec in the average time per region for the **CT brain** image type (**32** $\times$ **32** region size),

- about 3.04 min in total time and 0.38 sec in the average time per region for the **CT lung** image type.

A.2 Enhanced co-occurrence trees

The **static** rule showed improvement over the plain co-occurrence trees in the following cases:

- In the case of the **weave** image type, the largest improvement was 0.41% and was achieved by the **semi-dynamic** version for $\mathbf{K = 2}$ (K is the probability list length) and for the **32** $\times$ **32** region size. This corresponds to a reduction of about 8 sec in total time.

- In the case of the **ultrasound kidney** image type, the largest reduction in computational time was 0.88% and was achieved by the **semi-dynamic** version for $\mathbf{K = 2}$. This corresponds to a reduction of about 800 msec in total time.

The **semi-dynamic** version of the **move to front** rule showed improvement over the plain co-occurrence trees in the following cases:

- For the **asphalt** image type, the largest improvement was 2.04% and was achieved for $\mathbf{K = 3}$ and **16** $\times$ **16** region size. It corresponds to a reduction of about 7.68 sec in total time.

- For the **grass** image type, it was 2.09% and was also achieved for $\mathbf{K = 3}$ and **16** $\times$ **16** region size. It corresponds to a reduction of about 10.24 sec in total time.

- For the **fur** image type, it was 1.16% and was achieved for $\mathbf{K = 2}$ and $\mathbf{16 \times 16}$ region size. It corresponds to a reduction of about 17.92 sec in total time.

- For the **water** image type, it was 0.63% and was achieved for $\mathbf{K = 2}$ and $\mathbf{16 \times 16}$ region size. It corresponds to a reduction of about 5.12 sec in total time.

- For the **weave** image type, it was 1.54% and was achieved for $\mathbf{K = 2}$ and $\mathbf{32 \times 32}$ region size. It corresponds to a reduction of about 31.36 sec in total time.

- For the **normal mammograms**, the largest improvement was 0.30% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 34.56 sec in total time.

- For the **ultrasound heart** image type, the largest improvement was 1.96% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 0.8 sec in total time.

- For the **ultrasound kidney** image type, it was 0.94% and was achieved for $\mathbf{K = 3}$. It corresponds to a reduction of about 0.8 sec in total time.

- For the **ultrasound liver** image type, it was 1.04% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 1.6 sec in total time.

- For the **ultrasound ovaries** image type, it was 2.7% and was achieved for $\mathbf{K = 3}$. It corresponds to a reduction of about 1 sec in total time.

- For the **ultrasound spleen** image type, it was 1.64% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 1.6 sec in total time.

- For the **MR brain** image type (**12 bit** dynamic range), the largest improvement was 3.54% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 3.84 sec in total time.

- For the **MR brain** image type (**8 bit** dynamic range), it was 4.04% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 4.8 sec in total time.

- For the **CT lung** image type, the largest improvement was 1.05% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 1.92 sec in total time.

The **semi-dynamic** version of the **counter** rule showed improvement over the plain co-occurrence trees in the following cases:

- For the **asphalt** image type, the largest improvement was 0.67% and was achieved for $\mathbf{K = 2}$ and $\mathbf{16 \times 16}$ region size. It corresponds to a reduction of about 2.56 sec in total time.

- For the **grass** image type, it was 0.52% and was achieved for $\mathbf{K = 3}$ and $\mathbf{16 \times 16}$ region size. It corresponds to a reduction of about 2.56 sec in total time.

- For the **weave** image type, it was 0.85% and was achieved for $\mathbf{K = 2}$ and $\mathbf{32 \times 32}$ region size. It corresponds to a reduction of about 17.28 sec in total time.

- For the **ultrasound ovaries** image type, the largest improvement was 1.80% and was achieved for $\mathbf{K = 3}$. It corresponds to a reduction of about 0.6 sec in total time.

- For the **ultrasound spleen** image type, it was 0.82% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 0.8 sec in total time.

- For the **MR brain** image type (**12 bit** dynamic range), it was 2.09% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 3.36 sec in total time.

- For the **MR brain** image type (**8 bit** dynamic range), it was 1.42% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 4.32 sec in total time.

- For the **CT lung** image type, the largest improvement was 0.26% and was achieved for $\mathbf{K = 2}$. It corresponds to a reduction of about 0.5 sec in total time.

A.3 Self-adjusting co-occurrence trees

The **semi-splaying** technique showed improvement over the plain co-occurrence trees in the following cases:

- For the **asphalt** image type, the largest improvement was 8.78% and was achieved by the **full-dynamic** version for the $\mathbf{32 \times 32}$ region size. It corresponds to a reduction of about 56 sec in total time.

- For the **ultrasound heart** image type, the largest improvement was 2.76% and was achieved by the **semi-dynamic** version. It corresponds to a reduction of about 1.1 sec in total time.

- For the **ultrasound kidney** image type, it was 6.27% and was achieved by the **semi-dynamic** version. It corresponds to a reduction of about 5.7 sec in total time.

- For the **ultrasound spleen** image type, the largest improvement was 3.84% and was also achieved by the **semi-dynamic** version. It corresponds to a reduction of about 3.8 sec in total time.

- For the **MR femur** image type, the largest improvement employing the semi-splaying technique was 5.91% and was achieved by the **full-dynamic** version. It corresponds to a reduction of about 39.6 sec in total time.

- For the **MR skeletal muscle** image type (**15 bit** dynamic range), the largest improvement was 6% and was achieved by the **full-dynamic** version. It corresponds to a reduction of about 4.8 sec in total time.

- For the **CT liver** image type ($\mathbf{32 \times 32}$ region size), it was 6.72% and was achieved by the **semi-dynamic** version. It corresponds to a reduction of about 2.16 min in total time.

- For the **CT brain** image type (**32** $\times$ **32** region size), it was 4.15% and was achieved by the **semi-dynamic** version. It corresponds to a reduction of about 2.97 min in total time.

- For the **CT lung** image type, the largest improvement was 4.47% and was achieved by the **semi-dynamic** version. It corresponds to a reduction of about 8.16 sec in total time.

Bibliography

[1] L. Alexander and H. Kritikos. An investigation of the relation between radar images and visible texture. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 795–799, Maiami Beach, FL, 1980.

[2] B. Allen and I. Munro. Self-organizing binary search trees. *Journal of the ACM*, 25:526–535, 1978.

[3] F. Argenti, L. Alparone, and G. Benelli. Fast algorithms for texture analysis using co-occurrence matrices. *IEE Proc. Pt. F*, 137:443–448, 1990.

[4] Y. Ariji, M. Ohki, and K. Eguchi, et al. Texture analysis of sonographic features of the parotid gland in Sjögren's syndrome. *American Journal of Roentgenology*, 166:935–941, 1996.

[5] M. F. Augusteijn and L. E. Clemens. A performance evaluation of texture measures for image classification and segmentation using the cascade-correlation architecture. In *Proc. IEEE Int. Conf. on Neural Networks*, pages 4300–4305, Orlando, FL, 1994.

[6] M. F. Augusteijn, L. E. Clemens, and K. A. Shaw. Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier. *IEEE Trans. on Geoscience and Remote Sensing*, GE-33:616–625, 1995.

[7] A. Baraldi and F. Parmiggiani. An investigation of the textural characteristics associated with gray level co-occurence matrix statistical parameters. *IEEE Trans. on Geoscience and Remote Sensing*, GE-33:293–304, 1995.

[8] D. G. Barber and E. F. LeDrew. SAR sea ice discrimination using texture statistics: a multivariate approach. *Photogrammetric Engineering and Remote Sensing*, 57:385–395, 1991.

[9] B. Bargel. Classification of remote sensed data by texture and shape features in different spectral channels. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 2–4, Maiami Beach, FL, 1980.

[10] O. Basset, V. Ramiaramanana, and P. Chirossel, et al. Characterisation of muscle tissue during an effort by texture analysis of ultrasonic images. In *Proc. IEEE Ultrasonics Symp.*, pages 1455–1458, Cannes, France, 1994.

[11] O. Basset, Z. Sun, and G. Gimenez. Tissue characterisation by texture analysis of ultrasound images. In *Proc. SPIE 15th Int. Conf. on Applications of Digital Image Processing*, pages 547–558, San Diego, CA, 1992.

[12] O. Basset, Z. Sun, and J. L. Mestas, et al. Texture analysis of ultrasonic images of the prostate by means of co-occurrence matrices. *Ultrasonic Imaging*, 15:218–237, 1993.

[13] S. W. Bent, D. D. Sleator, and R. E. Tarjan. Biased search trees. *SIAM Journal on Computing*, 14:545–568, 1985.

[14] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B*, 36:192–236, 1974.

[15] J. R. Bitner. Heuristics that dynamically organize data structures. *SIAM Journal on Computing*, 8:82–110, 1979.

[16] A. C. Bovik, M. Clark, and W. S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-12:55–73, 1990.

[17] P. Brodatz. *Textures: a Photographic Album for Artists and Designers.* Dover Publications, 1966.

[18] P. T. Cahill and R. J. R. Knowles. Texture analysis of uniform fields: a prerequisite to tissue signature identification analysis of nuclear medicine

images. In *Proc. Int. Conf. on Pattern Recognition and Image Processing*, pages 127–130, Dallas, TX, 1981.

[19] P. T. Cahill, R. J. R. Knowles, and B. Kneeland, et al. Segmentation of white/gray matter by a texture based clustering algorithm. In *Proc. IEEE 6th Int. Conf. on Pattern Recognition*, pages 633–636, Munich, Germany, 1982.

[20] C. B. Caldwell, S. J. Stapleton, and D. W. Holdsworth, et al. Characterization of mammographic parenchymal pattern by fractal dimension. *Physics in Medicine and Biology*, 35:235–247, 1990.

[21] D. R. Carmichael, S. J. Clarke, and L. M. Linnett. Spatial models for texture classification. In *Colloquium on Texture Classification: Theory and Applications*, pages 9/1–9/5. Computing and Control Division, IEE, 1994.

[22] K. L. Chan. Quantitative characterization of electron micrograph image using fractal feature. *IEEE Trans. on Biomedical Engineering*, BME-42:1033–1037, 1995.

[23] K. Chandrasekaran, P. E. Aylward, and S. R. Fleagle, et al. Feasibility of identifying amyloid and hypertrophic cardiomyopathy with the use of computerized quantitative texture analysis of clinical echocardiographic data. *Journal of the American College of Cardiology*, 13:832–840, 1989.

[24] T. Chang and C. C. J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Trans. on Image Processing*, IM-2:429–441, 1993.

[25] C. C. Chen, J. S. Daponte, and M. D. Fox. Fractal feature analysis and classification in medical imaging. *IEEE Trans. on Medical Imaging*, MI-8:133–142, 1989.

[26] P. C. Chen and T. Pavlidis. Image segmentation as an estimation problem. In *Image Modeling*, pages 9–28. Academic Press, New York, 1981.

[27] P. C. Chen and T. Pavlidis. Segmentation by texture using correlation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5:64–69, 1983.

[28] Y. Chen, E. R. Dougherty, and S. M. Totterman, et al. Classification of trabecular structure in magnetic resonance images based on morphological granulometries. *Magnetic Resonance Imaging*, 29:358–370, 1993.

[29] Y. Q. Chen, M. S. Nixon, and D. W. Thomas. Statistical geometrical features for texture classification. *Pattern Recognition*, 28:537–552, 1995.

[30] A. J. Coleman, K. A. Tonge, and S. C. Rankin. The power spectral density as a texture measure in computed tomographic scans of the liver. *British Journal of Radiology*, 55:601–603, 1982.

[31] S. M. Collins, D. J. Skorton, and N. V. Prasad, et al. Quantitative echocardiographic image texture: normal contraction-related variability. *IEEE Trans. on Medical Imaging*, MI-4:185–192, 1985.

[32] R. W. Conners and C. A. Harlow. Equal probability quantizing and texture analysis of radiographic images. *Computer Graphics and Image Processing*, 8:447–463, 1978.

[33] R. W. Conners and C. A. Harlow. A theoretical comparison of texture algorithms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-2:204–222, 1980.

[34] R. W. Conners, C. A. Harlow, and S. J. Dwyer, III. Radiographic image analysis: past and present. In *Proc. IEEE 6th Int. Conf. on Pattern Recognition*, pages 1152–1169, Munich, Germany, 1982.

[35] R. W. Conners, M. M. Trivedi, and C. A. Harlow. Segmentation of a high resolution urban scene using texture operators. *Computer Vision, Graphics and Image Processing*, 25:273–310, 1984.

[36] G. R. Cross and A. K. Jain. Markov random field texture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5:25–39, 1983.

[37] J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two dimensional visual cortical filters. *Journal of the Optical Society of America A*, 2:1160–1169, 1985.

[38] L. S. Davis. Image texture analysis techniques-A survey. In J. C. Simon and R. M. Haralick, editors, *Digital Image Processing*, pages 189–201. D. Reidel Publishing Company, 1981.

[39] L. S. Davis, M. Clearman, and J. K. Aggarwal. An empirical evaluation of generalized co-occurrence matrices. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-3:214–221, 1981.

[40] L. S. Davis, S. A. Johns, and J. K. Aggarwal. Texture analysis using generalized co-occurence matrices. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-1:251–259, 1979.

[41] J. D. De Certaines, O. Henriksen, and A. Spisni, et al. In vivo measurements of proton relaxation times in human brain, liver, and skeletal muscle: a multicenter MRI study. *Magnetic Resonance Imaging*, 11:841–850, 1993.

[42] J. Desachy. Texture features in remote sensing imagery. In J. C. Simon and R. M. Haralick, editors, *Digital Image Processing*, pages 203–210. D. Reidel Publishing Company, 1981.

[43] J. F. Desaga, J. Dengler, and T. Wolf, et al. Digitisation of films and texture analysis for digital classification of pulmonary opacities. *Rontgenblatter*, 41:147–151, 1988.

[44] E. S. Deutsch and N. J. Belknap. Texture descriptors using neighbourhood information. Technical Report 184, Computer Science Center, College Park, University of Maryland, MD, 1972.

[45] A. P. Dhawan, Y. Chitre, and C. K. Bonasso, et al. Analysis of mammographic microcalcifications using gray-level image structure features. *IEEE Trans. on Medical Imaging*, MI-15:246–259, 1996.

[46] E. R. Dougherty, J. T. Newell, and J. B. Pelz. Morphological texture-based maximum-likelihood pixel classification based on local granulometric moments. *Pattern Recognition*, 25:1181–1198, 1992.

[47] P. Dreyer. Classification of land cover using optimized neural nets on SPOT data. *Photogrammetric Engineering and Remote Sensing*, 59:617–621, 1993.

[48] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[49] D. Dunn and W. E. Higgins. Optimal Gabor filters for texture segmentation. *IEEE Trans. on Image Processing*, IM-4:947–964, 1995.

[50] E. P. Durand and P. Rüegsegger. Cancellous bone structure: analysis of high-resolution CT images with the run-length method. *Journal of Computer Assisted Tomography*, 15:133–139, 1991.

[51] C. R. Dyer and A. Rosenfeld. Fourier texture features: suppression of aperture effects. *IEEE Trans. on System, Man, and Cybernetics*, SMC-6:703–705, 1976.

[52] J. Eklundh. On the use of Fourier phase features for texture discrimination. *Computer Graphics and Image Processing*, 9:199–201, 1979.

[53] C. Fortin, R. Kumaresan, and W. Ohley, et al. Fractal dimension in the analysis of medical images. *IEEE Engineering in Medicine and Biology Mag.*, 11:65–71, 1992.

[54] S. E. Franklin and D. R. Peddle. Texture analysis of digital image data using spatial cooccurrence. *Computers and Giosciences*, 13:293–311, 1987.

[55] K. S. Fu. Syntactic image modeling using stochastic tree grammars. In *Image Modeling*, pages 153–169. Academic Press, New York, 1981.

[56] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, Inc., 1990.

[57] A. Gagalowicz. Visual discrimination of stochastic texture fields based upon their second order statistics. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 786–788, Maiami Beach, FL, 1980.

[58] M. Galloway. Texture analysis using gray level run lengths. *Computer Graphics and Image Processing*, 4:172–199, 1974.

[59] B. S. Garra, M. F. Insana, and T. H. Shawker, et al. Quantitative ultrasonic detection and classification of diffuse liver disease. Comparison with human observer performance. *Investigative Radiology*, 24:196–203, 1989.

[60] P. Gong, D. J. Marceau, and P. J. Howarth. A comparison of spatial feature extraction algorithms for land-use classification with SPOT HRV data. *Remote Sensing of Environment*, 40:137–151, 1992.

[61] L. V. Gool, P. Dewaele, and A. Oosterlinck. Survey. Texture Analysis Anno 1983. *Computer Vision, Graphics, and Image Processing*, 29:336–357, 1985.

[62] C. C. Gotlieb and H. E. Kreyszig. Texture descriptors based on co-occurrence matrices. *Computer Vision, Graphics and Image Processing*, 51:70–86, 1990.

[63] R. M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, PROC-67:786–804, 1979.

[64] R. M. Haralick and K. Shanmugam. Computer classification of reservoir sandstones. *IEEE Trans. on Geoscience and Electronics*, GE-11:171–177, 1973.

[65] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. on System, Man, and Cybernetics*, SMC-3:610–621, 1973.

[66] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9:532–550, 1987.

[67] D. C. He, L. Wang, and J. Guibert. Texture feature extraction. *Pattern Recognition Letters*, 6:269–273, 1987.

[68] D. C. He and Li Wang. Texture unit, texture spectrum and texture analysis. *IEEE Trans. on Geoscience and Remote Sensing*, GE-28:509–512, 1990.

[69] L. Hepplewhite and T. J. Stonham. Magnetic disk inspection using texture recognition. In *Colloquium on Texture Classification: Theory and Applications*, pages 8/1–8/4. Computing and Control Division, IEE, 1994.

[70] L. S. Hibbard and D. W. McKeel, Jr. Multiscale detection and analysis of the Senile Plaques of Alzheimer's disease. *IEEE Trans. on Biomedical Engineering*, BME-42:1218–1225, 1995.

[71] Q. A. Holmes, D. R. Nuesch, and R. A. Shuchman. Textural analysis and real-time classification of sea-ice types using digital SAR data. *IEEE Trans. on Geoscience and Remote Sensing*, GE-22:113–120, 1984.

[72] J. Bezy-Wendling, A. Bruno, and P. Reuse. MRI Texture analysis applied to trabecular bone. An experimental study. In *Proc. 1st Int. Conf. on Computer Vision, Virtual Reality, and Robotics in Medicine*, pages 439–443, 1995.

[73] J. Bigün and J. M. H. du Buf. N-folded symmetries by complex moments in Gabor space and their application to unsupervised texture segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-16:80–87, 1994.

[74] J. M. H. du Buf, M. Kardan, and M. Spann. Texture feature performance for image segmentation. *Pattern Recognition*, 23:291–309, 1990.

[75] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[76] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24:1167–1186, 1991.

[77] A. K. Jain and K. Karu. Learning texture discrimination masks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-18:195–205, 1996.

[78] S. N. Jayaramamurthy. Texture discrimination using digital deconvolution filters. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 1184–1186, Miami Beach, FL, 1980.

[79] J. R. Jensen. Spectral and textural features to classify elusive land cover at the urban fringe. *Professional Geographer*, 31:400–409, 1979.

[80] B. Julesz. Visual pattern discrimination. *IRE Trans. on Information Theory*, 8:84–92, 1962.

[81] B. Julesz, E. N. Gilbert, and L. A. Shepp, et al. Inability of humans to discriminate between visual textures that agree in second order statistics-Revisited. *Perception*, 2:391–405, 1973.

[82] Y. M. Kadah, A. A. Farag, and J. M. Zurada, et al. Classification algorithms for quantitative tissue characterization of diffuse liver disease from ultrasound images. *IEEE Trans. on Medical Imaging*, MI-15:466–478, 1996.

[83] R. L. Kashyap. Univariate and multivariate random field models for images. *Computer Graphics and Image Processing*, 12:257–270, 1980.

[84] S. Katsuragawa, K. Doi, and H. MacMahon, et al. Quantitative computer-aided analysis of lung texture in chest radiographs. *Radiographics*, 10:257–269, 1990.

[85] J. M. Keller, S. Chen, and R. M. Crownover. Texture description and segmentation through fractal geometry. *Computer Vision, Graphics and Image Processing*, 45:150–166, 1989.

[86] D. E. Knuth. *The Art of Computer Programming vol. 3: Sorting and Searching.* Addison-Wesley Publ. Company, 1973.

[87] C. Kratzik, E. Schuster, and A. Hainz, et al. Texture analysis - A new method of differentiating prostatic carcinoma from prostatic hypertrophy. *Urological Research*, 16:395–397, 1988.

[88] R. P. Kruger, W. B. Thompson, and A. F. Turner. Computer diagnosis of pneumoconiosis. *IEEE Trans. on System, Man, and Cybernetics*, SMC-4:40–49, 1974.

[89] W. S. Kuklinski, K. Chandra, and U. E. Ruttirmann, et al. Application of fractal texture analysis to segmentation of dental radiographs. In *Proc. SPIE 3rd Medical Imaging Int. Conf.: Image Processing*, pages 111–117, Newport Beach, CA, 1989.

[90] A. Laine and J. Fan. Texture classification by wavelet packet signatures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-15:1186–1191, 1993.

[91] K. Lam, M. K. Siu, and C. T. Yu. A generalized counter scheme. *Theoretical Computer Science*, 16:271–278, 1981.

[92] K. I. Laws. Texture energy measures. In *Proc. Image Understanding Workshop*, pages 47–51, 1979.

[93] F. C. Leachtenauer. Optical power spectrum analysis: scale and resolution effects. *Photogrammetric Engineering and Remote Sensing*, 43:1117–1125, 1977.

[94] B. G. Lee, R. T. Chin, and D. W. Martin. Automated rain-rate classification of satellite images using statistical pattern recognition. *IEEE Trans. on Geoscience and Remote Sensing*, GE-23:315–324, 1985.

[95] J. Lee, R. C. Weger, and S. K. Sengupta, et al. A neural network approach to cloud classification. *IEEE Trans. on Geoscience and Remote Sensing*, GE-28:846–855, 1990.

[96] G. G. Lendaris and G. L. Stanley. Diffraction-pattern sampling for automatic pattern recognition. *Proceedings of the IEEE*, PROC-58:198–216, 1970.

[97] R. A. Lerski. Texture analysis of medical images. In *Proc. Int. Symp. on Physics of Medical Imaging and Advances in Computer Applications*, pages 64–68, 1993.

[98] R. A. Lerski, K. Straughan, and L. R. Schad, et al. MR Image texture analysis - an approach to tissue characterization. *Magnetic Resonance Imaging*, 11:873–887, 1993.

[99] Z. Liang. Tissue classification and segmentation of MR images. *IEEE Engineering in Medicine and Biology Mag.*, 12:81–85, 1993.

[100] S. S. Liu and M. E. Jernigan. Texture analysis and discrimination in additive noise. *Computer Vision, Graphics and Image Processing*, 49:52–67, 1990.

[101] D. Lu, J. T. Tou, and T. Gu. A simplified procedure for statistical feature extraction in texture processing. In *Proc. IEEE Int. Conf. on Pattern Recognition and Image Processing*, pages 589–592, Dallas, TX, 1981.

[102] T. Lundahl, W. J. Ohley, and S. M. Kay, et al. Fractional Brownian motion: a maximum likelihood estimator and its application to image texture. *IEEE Trans. on Medical Imaging*, MI-5:152–161, 1986.

[103] I. E. Magnin, F. Cluzeau, and C. L. Odet. Mammographic texture analysis: an evaluation of risk for developing breast cancer. *Optical Engineering*, 25:780–784, 1986.

[104] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-11:674–693, 1989.

[105] B. B. Mandelbrot. *The Fractal Geometry of Nature.* San Fransisco, CA: Freeman, 1982.

[106] D. J. Marceau, P. J. Howarth, and J. M. M. Dubois, et al. Evaluation of the grey level co-occurence matrix method for land-cover classification using SPOT-imagery. *IEEE Trans. on Geoscience and Remote Sensing*, GE-28:513–518, 1980.

[107] T. Matsuyama, S. Miura, and M. Nagao. A structural analysis of natural textures by Fourier transformation. In *Proc. IEEE 6th Int. Conf. on Pattern Recognition*, pages 289–292, Munich, Germany, 1982.

[108] R. W. McColl and G. R. Martin. Texture analysis and synthesis. Technical Report 136, Dept. of Computer Science, University of Warwick, 1989.

[109] M. F. McNitt-Gray, H. K. Huang, and J. W. Sayre. Feature selection in the pattern classification problem of digital chest radiograph segmentation. *IEEE Trans. on Medical Imaging*, MI-14:537–547, 1995.

[110] K. Mehlhorn and A. Tsakalidis. Data structures. In *Handbook of Theoretical Computer Science*, pages 301–341. Elsevier Science Publishers, co-published by MIT-Press, 1990.

[111] P. Miller and S. Astley. Classification of breast tissue by texture analysis. *Image and Vision Computing*, 10:258–265, 1992.

[112] A. H. Mir, M. Hanmandlu, and S. N. Tandon. Texture analysis of CT images. *IEEE Engineering in Medicine and Biology Mag.*, 14:781–786, 1995.

[113] O. R. Mitchell, C. R. Myers, and W. Boyne. A max-min measure for image texture analysis. *IEEE Trans. on Computers*, C-26:408–414, 1977.

[114] D. T. Morris. An evaluation of the use of texture measurements for the tissue characterization of ultrasonic images of in-vivo human placenta. *Ultrasound in Medicine and Biology*, 14:387–395, 1988.

[115] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[116] D. R. Nüesch. Classification of SAR imagery from an agricultural region using digital texture analysis. In *Proc. ISP Symp.*, pages 231–240, Toulouse, France, 1982.

[117] P. P. Ohanian and R. C. Dubes. Performance evaluation for four classes of textural features. *Pattern Recognition*, 25:819–833, 1992.

[118] E. Oja and K. Valkealahti. Compressing higher-order co-occurrences for texture analysis using the Self-Organizing Map. In *Proc. IEEE Int. Conf. on Neural Networks*, pages 1160–1164, Perth, Australia, 1995.

[119] A. Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, 2nd edition, 1984.

[120] F. G. Peet and T. S. Sahota. Surface curvature as a measure of image texture. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-7:734–738, 1985.

[121] S. Peleg, J. Naor, and R. Hartley, et al. Multiple resolution texture analysis and classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6:518–523, 1984.

[122] A. P. Pentland. Fractal-based description of natural scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6:661–674, 1984.

[123] M. Porat and Y. Y. Zeevi. The generalized Gabor scheme of image representation in biological and machine vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-10:452–467, 1988.

[124] T. J. Pultz and R. J. Brown. SAR image classification of agricultural targets using first- and second-order statistics. *Canadian Journal of Remote Sensing*, 13:85–91, 1987.

[125] U. Raeth, D. Schlaps, and B. Limberg, et al. Diagnostic accuracy of computerized B-scan texture analysis and conventional ultrasonography in diffuse parenchymal and malignant liver disease. *Journal of Clinical Ultrasound*, 13:87–99, 1985.

[126] T. R. Reed and J. M. H. du Buf. A review of recent texture segmentation and feature extraction techniques. *CVGIP-Image Understanding*, 57:359–372, 1993.

[127] A. Rosenfeld and N. Thurston. Edge and curve detection for visual scene analysis. *IEEE Trans. on Computers*, C-20:562–569, 1971.

[128] U. E. Ruttimann and J. A. Ship. The use of fractal geometry to quantitate bone structure from radiographs. *Journal of Dental Research*, 69:287, 1990. Abstract no 1431.

[129] B. Sahiner, H. P. Chan, and N. Petrick, et al. Classification of mass and normal breast tissue: a convolution neural network classifier with spatial domain and texture images. *IEEE Trans. on Medical Imaging*, MI-15:598–610, 1996.

[130] J. Samarabandu, R. Acharya, and E. Hausmann, et al. Analysis of bone X-rays using morphological fractals. *IEEE Trans. on Medical Imaging*, MI-12:466–470, 1993.

[131] N. Sarkar and B. B. Chaudhuri. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Trans. on System, Man, and Cybernetics*, SMC-24:115–120, 1994.

[132] L. R. Schad, S. Blüml, and I. Zuna. MR tissue characterization of intracranial tumors by means of texture analysis. *Magnetic Resonance Imaging*, 11:889–896, 1993.

[133] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural, and Neural Approaches.* John Wiley and Sons, Inc., 1992.

[134] D. Schlaps, I. Zuna, and M. Walz, et al. Ultrasonic tissue characterization by texture analysis: elimination of tissue-independent factors. In *Proc. SPIE Int. Symp. on Pattern Recognition and Acoustical Imaging*, pages 128–134, Newport Beach, CA, 1987.

[135] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, New York, 1983.

[136] K. S. Shanmugan, V. Narayanan, and V. S. Frost, et al. Textural features for radar image analysis. *IEEE Trans. on Geoscience and Remote Sensing*, GE-19:153–156, 1981.

[137] H. C. Shen and C. Y. C. Bie. Feature frequency matrices as texture image representation. *Pattern Recognition Letters*, 13:195–205, 1992.

[138] H. C. Shen, C. Y. C. Bie, and D. K. Y. Chiu. A textured-based distance measure for classification. *Pattern Recognition*, 26:1429–1437, 1993.

[139] K. Shimizu, T. Johkoh, and J. Ikezoe, et al. Fractal analysis for classification of ground-glass opacity on high-resolution CT: an in vitro study. *Journal of Computer Assisted Tomography*, 21:955–961, 1997.

[140] J. Sklansky. Medical radiographic image processing and pattern recognition. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 374–382, Miami Beach, FL, 1980.

[141] D. D. Sleator and R. E. Tarjan. Self-adjusting binary seach trees. *Journal of the ACM*, 32:652–686, 1985.

[142] P. P. Smyth, J. E. Adams, and R. W. Whitehouse, et al. Application of computer texture analysis to the Singh Index. *British Journal of Radiology*, 70:242–247, 1997.

[143] P. P. Smyth, C. J. Taylor, and J. Adams. Texture analysis using local property maps. In *Proc. 6th British Machine Vision Conf.*, pages 47–56, Birmingham, England, 1995.

[144] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision.* Chapman and Hall Computing, 1993.

[145] T. E. Southard and K. A. Southard. Detection of simulated osteoporosis in maxillae using radiographic texture analysis. *IEEE Trans. on Biomedical Engineering*, BME-43:123–132, 1996.

[146] H. Stark and D. Lee. An optical digital approach to the pattern recognition of Coal Worker's Pneumoconiosis. *IEEE Trans. on System, Man, and Cybernetics*, SMC-6:788–793, 1976.

[147] S. R. Sternberg. Grayscale morphology. *Computer Vision, Graphics and Image Processing*, 35:333–355, 1986.

[148] K. Straughan, R. A. Lerski, and D. H. Carr. Use of texture analysis in medical imaging. *British Journal of Radiology*, 61:786, 1988.

[149] Y. N. Sun, M. H. Horng, and X. Z. Lin, et al. Ultrasonic image analysis for liver diagnosis: a noninvasive alternative to determine liver disease. *IEEE Engineering in Medicine and Biology Mag.*, 15:93–101, 1996.

[150] R. M. Sutton and E. L. Hall. Texture measures for automatic classification of pulmonary disease. *IEEE Trans. on Computers*, C-21:667–676, 1972.

[151] A. E. Svolos, C. A. Hutton, and A. Todd-Pokropek. Co-occurrence trees: A dynamic solution for texture feature extraction. In *Proc. IEEE EMBS 18th Int. Conf. on Engineering in Medicine and Biology*, pages 1142–1144, Amsterdam, The Netherlands, 1996.

[152] A. E. Svolos and A. Todd-Pokropek. An evaluation of a number of techniques for decreasing the computational complexity of texture feature extraction through an application to ultrasonic image analysis. In *Proc. IEEE EMBS 19th Int. Conf. on Engineering in Medicine and Biology*, pages 601–604, Chicago, IL, 1997.

[153] A. E. Svolos and A. Todd-Pokropek. Self-adjusting binary search trees: An investigation of their space and time efficiency in texture analysis of magnetic resonance images using the spatial grey level dependence method. In *Proc. SPIE Medical Imaging Int. Conf.: Image Processing*, pages 220–231, San Diego, CA, 1998.

[154] A. E. Svolos and A. Todd-Pokropek. Time and space results of dynamic texture feature extraction in MR and CT image analysis. *IEEE Trans. on Information Technology in Biomedicine*, ITB-2(2):48–54, 1998.

[155] R. E. Tarjan. Updating a balanced search tree in $O(1)$ rotations. *Information Processing Letters*, 16:253–257, 1983.

[156] R. E. Tarjan. Amortized computational complexity. *SIAM Journal on Algebraic and Discrete Methods*, 2:306–318, 1985.

[157] R. E. Tarjan. Sequential access in splay trees takes linear time. *Combinatorica*, 5:367–378, 1985.

[158] D. Terzopoulos and S. W. Zucker. Texture discrimination using intensity and edge co-occurences. In *Proc. IEEE 5th Int. Conf. on Pattern Recognition*, pages 565–569, Maiami Beach, FL, 1980.

[159] A. Teuner, O. Pichler, and B. J. Hosticka. Unsupervised texture segmentation of images using tuned matched Gabor filters. *IEEE Trans. on Image Processing*, IM-4:863–870, 1995.

[160] J. P. Thiran and B. Macq. Morphological feature extraction for the classification of digital images of cancerous tissues. *IEEE Trans. on Biomedical Engineering*, BME-43:1011–1019, 1996.

[161] F. Tomita, Y. Shirai, and S. Tsuji. Description of textures by a structural analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-4:183–191, 1982.

[162] A. Tsakalidis. Private communication.

[163] A. Tsakalidis. *Data Structures.* University of Patras Publications, 1989.

[164] M. R. Turner. Texture discrimination by Gabor functions. *Biological Cybernetics*, 55:71–82, 1986.

[165] M. Unser. A fast texture classifier based on cross entropy minimisation. In *Proc. 2nd European Signal Processing Conf.*, pages 261–264, Erlangen, Germany, 1983.

[166] M. Unser. Local linear transforms for texture measurements. *Signal Processing*, 11:61–79, 1986.

[167] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Trans. on Image Processing*, IM-4:1549–1560, 1995.

[168] K. Unterauer. Dynamic weighted binary search trees. *Acta Informatica*, 11:341–362, 1979.

[169] P. F. Van Der Stelt and W. G. M. Geraets. Use of the fractal dimension to describe the trabecular pattern in osteoporosis. *Journal of Dental Research*, 69:287, 1990. Abstract no 1430.

[170] A. L. Vickers and J. W. Modestino. A maximum likelihood approach to texture classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-4:61–68, 1982.

[171] F. M. Vilnrotter, R. Nevatia, and K. E. Price. Structural analysis of natural textures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-8:76–89, 1986.

[172] Z. Wang and W. Abmayr. Fourier transformation for chromatin texture and shape in cell image analysis. In *Proc. IEEE 6th Int. Conf. on Pattern Recognition*, pages 1212–1212, Munich, Germany, 1982.

[173] H. Wechsler. Texture analysis - a survey. *Signal Processing*, 2:271–282, 1980.

[174] R. M. Welch, K. S. Kuo, and S. K. Sengupta. Cloud and surface textural features in polar regions. *IEEE Trans. on Geoscience and Remote Sensing*, GE-28:520–528, 1990.

[175] M. Werman and S. Peleg. Min-Max operators in texture analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-7:730–733, 1985.

[176] J. Weszka, C. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Trans. on System, Man, and Cybernetics*, SMC-6:269–285, 1976.

[177] C. M. Wu and Y. C. Chen. Statistical feature matrix for texture analysis. *CVGIP: Graphical models and Image Processing*, 54:407–419, 1992.

[178] C. M. Wu, Y. C. Chen, and K. S. Hsieh. Texture features for classification of ultrasonic liver images. *IEEE Trans. on Medical Imaging*, MI-11:141–152, 1992.

[179] J. K. Wu, Q. F. Zheng, and W. T. Wang. A forest inventory using LANDSAT imagery in the Mao-shan area of china. *Int. Journal of Remote Sensing*, 6:1783–1795, 1985.

[180] G. Zinser, D. Komitowski, and J. Bille. Evaluation of the chromatin structure of cell nuclei by 3-D light microscopic image processing. In *Proc. IEEE 6th Int. Conf. on Pattern Recognition*, pages 1173–1175, Munich, Germany, 1982.

[181] S. W. Zucker and D. Terzopoulos. Finding structure in co-occurrence matrices for texture analysis. In *Image Modeling*, pages 423–445. Academic Press, New York, 1981.

[182] I. Zuna. Computerized ultrasonic tissue characterization: methods and clinical use. In *Proc. Int. Symp. on Computer Assisted Radiology*, pages 155–163, Berlin, Germany, 1987.

Time and Space Results of Dynamic Texture Feature Extraction in MR and CT Image Analysis

Andreas E. Svolos, *Member, IEEE*, and Andrew Todd-Pokropek, *Member, IEEE*

Abstract— **Texture feature extraction is a fundamental part of texture image analysis. Therefore, the reduction of its computational time and storage requirements should be an aim of continuous research.**

The Spatial Grey Level Dependence Method (SGLDM) is one of the most important statistical texture description methods, especially in medical image analysis. Co-occurrence matrices are employed for the implementation of this method; however, they are inefficient in terms of computational time and memory space, due to their dependency on the number of gray levels (gray-level range) in the entire image. Since texture is usually measured in a small image region, a large amount of memory is wasted while the computational time of the texture feature extraction operations is unnecessarily raised. Their inefficiency puts up barriers to the wider utilization of SGLDM in a real application environment, such as a clinical environment.

In this paper, the memory space and time efficiency of a dynamic approach to texture feature extraction in SGLDM is investigated through a pilot application in the analysis of magnetic resonance (MR) and computed tomography (CT) images.

Index Terms— **Co-occurrence matrix, medical image analysis, Spatial Grey-Level Dependence Method (SGLDM), texture analysis.**

I. Introduction

TEXTURE is a fundamental feature for image analysis, classification, and segmentation. Its usefulness has been proved for many types of images, ranging from multispectral remote-sensed aerial and satellite data to biomedical images. In medical image analysis, texture has been successfully used for tissue characterization, i.e., the determination of tissue type (normal or pathological) and further classification of tissue pathology [1].

Texture analysis has been shown to increase the level of diagnostic information extracted from images of most medical imaging modalities and quantitatively characterize differences in appearances inaccessible to human observers. According to [2], texture analysis is of great importance in the diagnostic process for many diseases. In [3], it is mentioned that the extraction of features characterizing the texture of the lungs and the bones is a very important process in chest analysis. Medical applications of computer-based texture analysis date back to the early 1970's, when it became possible to digitize x-rays for computer processing. The explosion in the use of digital techniques has led to the possibility of utilizing texture analysis for most medical imaging modalities. Typical work has been carried out on digital x-rays [4]–[9], ultrasound images [10]–[16], magnetic resonance (MR) images [17], [18], and x-ray computer tomography (CT) images [19], [20].

There is no formal mathematical definition texture. This is due to its wide variability. According to [21], [22], texture is a property that characterizes almost all surfaces, e.g., the grain of wood, weave of a fabric, pattern of crops in a field. Two major categories of texture analysis methods exist: statistical and syntactic or structural. Statistical methods employ features computed from the spatial distribution of the gray levels for texture characterization. These features usually measure textural characteristics like coarseness, contrast, and directionality. On the other hand, syntactic or structural methods describe texture by means of primitive descriptions and placement rules. The primitives are geometrical objects, such as line segments, open polygons, and closed polygons. The primitive placement rules describe the spatial relationships of the primitives. Syntactic texture analysis methods are not as highly developed, in the case of medical applications, as the statistical methods. The textural information extracted by the statistical methods has been shown to be invaluable for automatic tissue characterization and diagnosis. According to [23], statistical methods have a great value in biomedical image analysis.

One of the most significant statistical texture description methods is the Spatial Grey Level Dependence Method (SGLDM). Conners *et al.* [3] claim that it is a superior method for biomedical texture analysis. SGLDM [22], [24] characterizes the texture in an image region by means of features derived from the spatial distribution of pairs of gray levels (second-order distribution) having certain interpixel distances (separations) and orientations. It is based on the assumption that texture information is contained in the overall spatial relationship that the gray levels have to one another. However, the co-occurrence matrix [22], which is used for storing the estimated second-order probabilities, is inefficient in terms of memory as well as the computational time needed for the calculation of textural features. Many researchers who employed this method were forced to reduce the gray-level range of the image to handle its large requirements. Doing so, however, significant texture information was admitted to be lost, which could have led to a better texture discrimination. In [25], gray-level range reduction resulted in a degraded classifier performance. Wu *et al.* [11] claimed that low computational complexity and high classification accuracy are two of the most important considerations for the classification of ultrasonic liver images. According to them, SGLDM is one of the worst methods in terms of speed, since its computational complexity depends on the size of the co-occurrence matrix,

Manuscript received December 17, 1997; revised April 23, 1998.

The authors are with the Image Processing Laboratory, Department of Medical Physics, University College London, WC1E 6JA London, U.K. (e-mail: rmapasv@medphys.ucl.ac.uk; svolos@diogenis.ceid.upatras.gr).

Publisher Item Identifier S 1089-7771(98)05718-5.

which in turn depends on the number of gray levels in the entire analyzed image. Morris [16] had to reduce the number of gray levels from 256 to 32, prior to ultrasonic image analysis, employing SGLDM, to save on computer memory requirements. Lerski *et al.* [17] claimed that, although SGLDM is one of the best methods for the texture analysis of MR images, it is necessary to limit the gray-level range to save memory and computational time.

The significant drawbacks of the co-occurrence matrix are due to its dependency on the number of gray levels in the entire image. This results in whole rows and/or columns of the matrix containing zero entries, which contribute nothing to the texture feature computation process. Recently, a dynamic approach to texture feature extraction was proposed, which is called the co-occurrence trees [26]. It contains the same textural information as the co-occurrence matrix, but it completely eliminates redundancy. In this paper, we investigate the efficiency of this method in terms of memory space and computational time, through its application to the analysis of MR and CT images. The rest of this paper is organized as follows. In Section II, we describe the dynamic approach. In Section III, we present memory space and computational time results from the analysis of real data, using both the co-occurrence matrix and the co-occurrence trees. In Section IV, we discuss the results of Section III and investigate the usefulness of the dynamic approach in a clinical environment. We conclude in Section V.

II. Co-Occurrence Trees

As we have already mentioned, SGLDM is based on the computation of spatial second-order statistics from local regions of an image (regions of interest). The first step in this process involves the estimation of the second-order probabilities $p(i, j; d, \vartheta)$ of having a pair of gray levels (i, j) at an interpixel distance d and orientation ϑ, for each (i, j) in the analyzed region [22]. Probability $p(i, j; d, \vartheta)$ is approximated by the normalized co-occurrence frequency $P_{(i,j)}$, which is the number of occurrences of pair (i, j) divided by the total number of pairs that satisfy the (d, ϑ) spatial relation. Texture feature extraction concludes with the computation of a number of features from these probabilities.

Instead of storing $p(i, j; d, \vartheta)$ in a matrix form, it is possible to create a dynamic data structure (e.g., a tree) for each gray level i and put a node in it that corresponds to its paired gray level j. In this way, a set of dynamic data structures is employed for the storage of the estimated second-order probabilities. The co-occurrence trees [26] comprise such a set of dynamic data structures, called balanced binary trees (BB-trees). A BB-tree [27] is a full binary search tree. An example is illustrated in Fig. 1. It has been characterized as the best balanced tree for main memory applications [28]. This is due to the following properties.

- Access, insert, and delete operations cost $O(\log n)$ time,[1] where n is the cardinality of the set of elements stored in the tree.

[1] All logarithms are base 2.

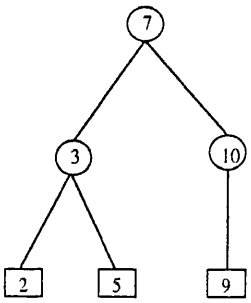


Fig. 1. Balanced binary tree for the gray-level sequence {2, 3, 7, 5, 9, 10}.

- Amortized rebalancing cost (the cost of the operations necessary to keep the tree balanced) is only $O(1)$ [27].
- Only one bit is needed for the balance information in each node of the tree; actually, in the co-occurrence trees approach, this bit can be embedded in the information content of each node, so practically no space is needed for this purpose.

Let n_g denote the number of gray levels in the analyzed region of interest. The co-occurrence trees contain at most n_g BB-trees, one tree for each gray level. Let T_k be the BB-tree that corresponds to gray level k. Then, for each pair (k, l) of gray levels that satisfies the (d, ϑ) spatial relation in the analyzed region, there is a node in T_k that stores the relevant information. This information consists of the gray level l, co-occurrence frequency $P_{(k,l)}$, and two pointers to its children, which are used for building the corresponding BB-tree T_k. In addition, the node contains a pointer to a node in some other BB-tree, which also contains gray level l. These pointers form lists of nodes, one list for each such gray level l, which partition the co-occurrence trees into disjoint subsets of nodes, according to the gray level stored in them. The lists are used for speeding some texture feature extraction operations. An example of a co-occurrence trees formation and its equivalence to the co-occurrence matrix is illustrated in Fig. 2 for the $(1, 0°)$ spatial relation. The pointers in the co-occurrence trees structure form the relevant lists L_0, L_1, and L_2. Note that only the gray level pairs that satisfy $(1, 0°)$ relation in the sample image region are stored in the co-occurrence trees, in contrast with the co-occurrence matrix. The heads of the lists as well as the pointers to the roots of the BB-trees are stored in two separate data structures (two additional BB-trees).

III. Results

In this section, we first present computational time results from the analysis of real clinical image data for each one of the compared approaches, namely, the co-occurrence matrix and the co-occurrence trees. All algorithms were run on a Dell P100t PC. This choice was motivated by the fact that a PC is much cheaper and, therefore, more widely available than a Unix workstation, such as a Sun SPARC station. Therefore, the time performance of the above approaches on a PC is of great importance. The PC employed in this study had a main memory of 40 MB and one hard disk unit of 800 MB. Its CPU was an Intel Pentium microprocessor running at 100 MHz. All programs were written in the C++ programming language and were compiled and run under Windows 3.11 (16-bit)

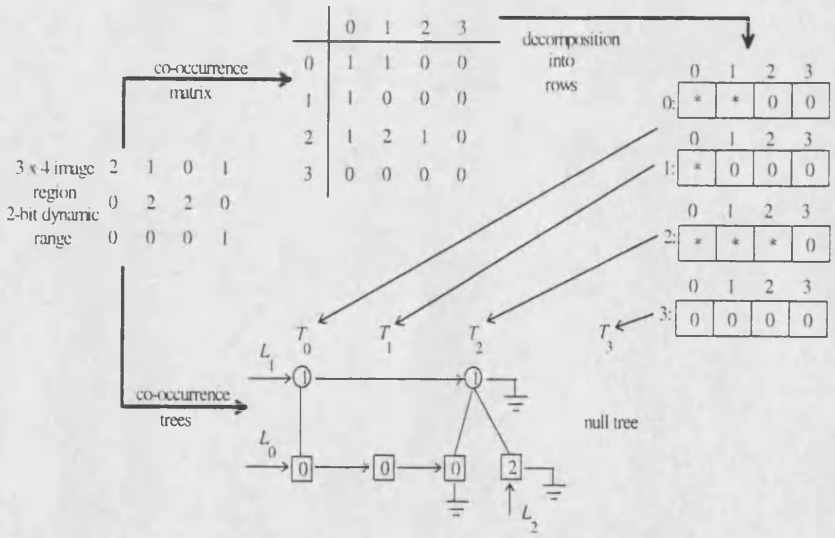


Fig. 2. Example of a co-occurrence matrix, co-occurrence trees, and their relationship ("*" indicates a nonzero entry).

(a) (b) (c)

Fig. 3. Images from the MR data set: (a) MR femur image, (b) MR brain image (8-bit dynamic range), and (c) MR brain image (12-bit dynamic range).

operating system. All image data were first stored on the hard disk. Then, for each analyzed region, the appropriate data were loaded from the hard disk into the main memory. The time for this operation is the only constituent time of the texture feature extraction process that depends on the size of the analyzed region. This time is the same for both the co-occurrence matrix and the co-occurrence trees approach and, therefore, was ignored in this study. The computational time was being measured from the point where all necessary data were in main memory until all texture features were computed. No virtual memory paging scheme was required in this computation.

Two data sets were employed for the evaluation of the time performance of the compared approaches. The first data set consisted of MR images, while the second consisted of CT x-ray images. Each one of the two data sets included images of three of the most commonly analyzed texture types for tissue characterization. Ten images of each type were used for texture analysis. All images were taken from the database of the Department of Medical Physics, University College London, U.K. All analyzed region parameters (size and location on the image) were chosen under the guidance of an experienced radiologist, aiming to cover as much of the clinically interesting texture in the analyzed image as possible. No reduction in their gray-level resolution was performed. To the best of our knowledge, this is the first reported texture analysis of MR and CT images at their initial dynamic range using SGLDM.

In the MR data set, the first image type consisted of high-resolution images of the human femur [see Fig. 3(a)]. The images had a spatial resolution of 512 × 512 and a dynamic range of 15 bits. Thirty regions that covered most of the useful texture in each image were selected for analysis. Both the width and height of each region was set at 32 pixels. The second image type consisted of MR images of the human brain [see Fig. 3(b)]. Their spatial resolution was 256 × 256, and their dynamic range was 8 bits. Twenty regions of size 8 × 8 were selected in each analyzed image. Of medical interest in this image type is the white and gray matter of the brain tissue. Ten regions were selected from the white matter, and the other ten regions were selected from the gray matter. The third image type also contained MR images of the human brain [see Fig. 3(c)]. However, the images belonging to this type had a gray-level resolution of 12 bits. Their spatial resolution was 256 × 256. Twenty regions of size 10 × 10 were selected from the white matter in each analyzed image.

(a) (b) (c)

Fig. 4. Images from the CT data set: (a) CT liver image (32 × 32 region size), (b) CT liver image (16 × 16 region size), and (c) CT brain image.

In the CT data set, the first two image types illustrate the human abdomen [see Fig. 4(a) and (b)]. All images belonging to these types had the same spatial resolution (512 × 512) and dynamic range (12 bits). One of the areas of medical interest in a CT human abdomen image is liver parenchyma. Twenty regions from the liver tissue were selected in each image. The size of the regions for the images of the first type was 32 × 32. The corresponding region size for the second image type was 16 × 16. The last image type consisted of CT images of the human brain [see Fig. 4(c)]. The images had a spatial resolution of 512 × 512 and a dynamic range of 12 bits. Twenty regions of size 16 × 16 were selected for texture analysis.

All images were preprocessed before analysis using histogram equalization [29]. This normalization is necessary before feature extraction in order that the features be invariant under monotonic gray-level transformations induced by variations in image acquisition conditions, such as lighting. According to [14], normalizing the data improves the overall diagnostic accuracy in medical image analysis and classification. Other normalization techniques have been employed in texture analysis, such as the "equal-probability quantizing" algorithm [22]. To the best of our knowledge, there has been no study proving one particular method to be the optimal form of normalization for texture analysis. The histogram equalization algorithm presented in [29] was chosen here because it is much simpler and faster than other existing techniques.

Eight distinct pairs (d, ϑ) were employed, which are among the most widely used in medical image analysis. They correspond to an interpixel distance of one and two pixels and the four basic orientations 0, 45, 90, and 135°. The direction was considered to be unimportant, (see [22]) and therefore, the orientations 180, 225, 270, and 315° were not employed in this analysis. From each analyzed region, the first 13 textural features proposed in [22] were computed. The only exception was the case of the first image type of the MR data set (MR images of the human femur). Four of the above textural features were excluded from the computations, namely, sum entropy, sum variance, difference entropy, and second information measure of correlation. The reason was that the computation of these features takes a very long time for images of such a large dynamic range (15 bits). Since these features have been rarely used in the texture analysis process, this exclusion is not important. In this case, the comparison between the co-occurrence matrix and the proposed approach can be based on the time results from the computation of the rest of the texture features. The average total time for the analysis of all selected regions in each image, using all the above (d, ϑ) pairs, was computed for each image type. In addition, an averaging over time was performed and the average time per region was estimated, again for each image type. Both the total time and the estimated average time per region are reliable and robust time measurements on which a comparison between the two approaches can be based. The corresponding computational time results are shown in Figs. 5 and 6. In these graphs, "cmtx" stands for the co-occurrence matrix approach and "ctree" stands for the co-occurrence trees. In the case of a zero height column, the corresponding approach failed to produce any results because it led the system to run out of memory.

Finally, results about the memory requirements of the compared approaches are presented for each analyzed image type in the form of the sparsity of the co-occurrence matrix (see Figs. 7 and 8). Since the proposed approach eliminates redundancy (zero entries), this sparsity reveals the reduction in memory space using the co-occurrence trees (see also Section II). Only part of each co-occurrence matrix is actually shown in the three-dimensional (3-D) illustrations. The rest of it contains only zero entries. Each of the illustrated matrices was computed from one region of one image of the corresponding image type, employing the $(1, 0°)$ spatial relation (see Figs. 3 and 4). The co-occurrence matrices from other regions and other images of the same type are similar. Note that the 3-D illustration of the relevant co-occurrence matrix is not shown in Fig. 7(a); only the percentage of zero entries. This is because the co-occurrence matrices of the MR femur image type have a huge size and, therefore, cannot be displayed.

IV. Discussion

From Figs. 7 and 8, it is clear that the co-occurrence matrix is very sparse. In all cases, the percentage of zero entries

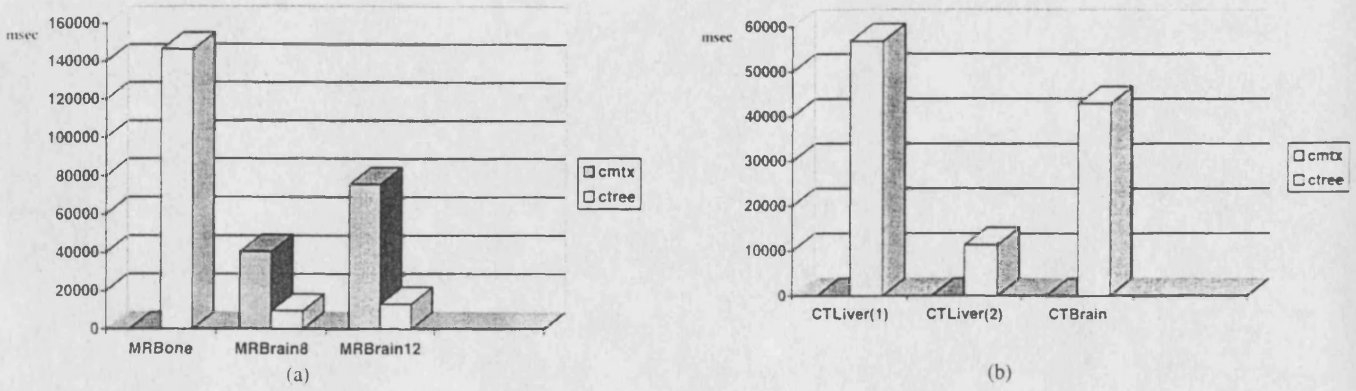


Fig. 5. Average total time: (a) MR data set and (b) CT data set.

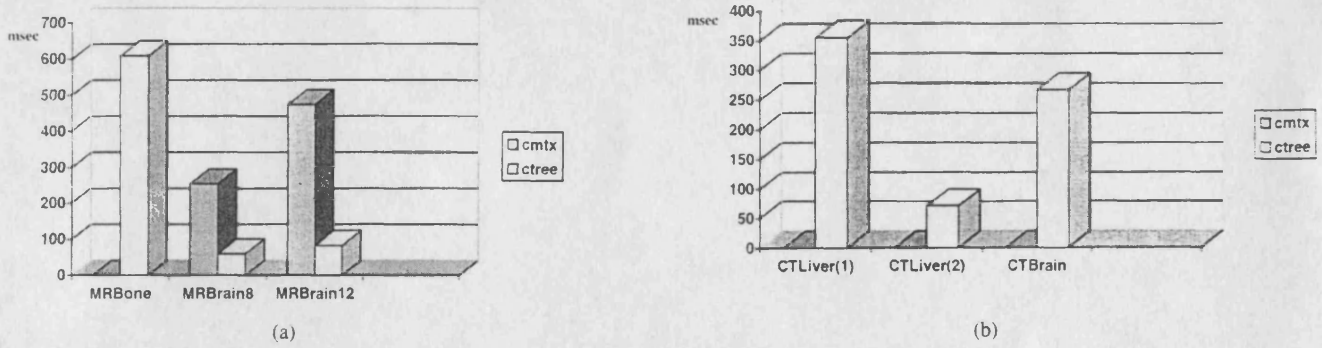


Fig. 6. Average time per region: (a) MR data set and (b) CT data set.

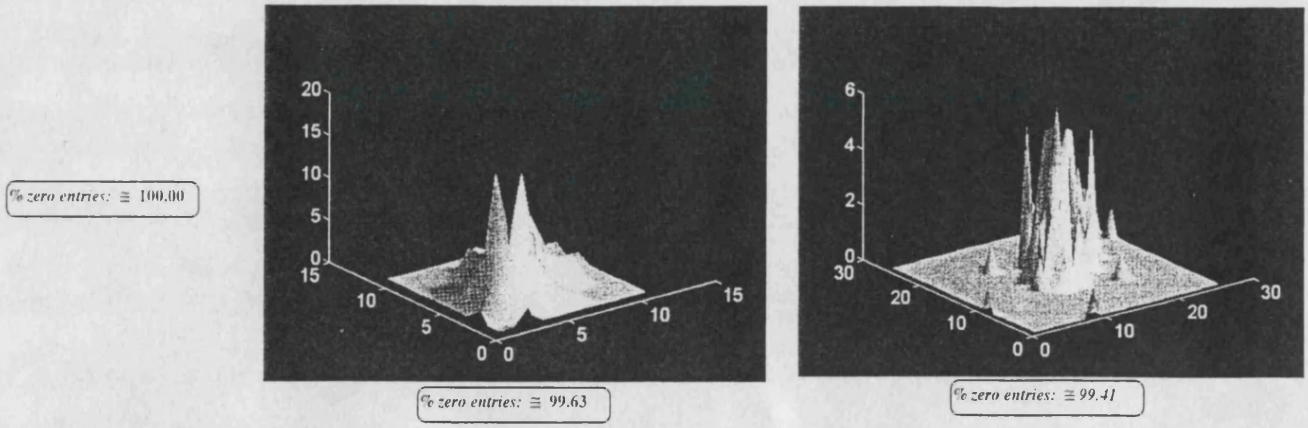
is more than 99%. In particular, for the CT images and the MR images of the human femur, its sparsity approaches 100%. Therefore, its inefficiency in terms of memory space is obvious. A large reduction in the number of gray levels in the analyzed image (usually down to 32 or less) is necessary in order that the co-occurrence matrix can be employed. This preprocessing step is performed in most cases when SGLDM is used in the texture analysis of medical images. However, reduction in the number of gray levels leads to an unavoidable loss of textural information, which has been shown to be significant, especially in the analysis of medical images (see Section I).

The zero entries in the co-occurrence matrices contribute nothing to the computation of the textural features. Because of their very large sparsity, the texture analysis algorithm spends almost all of its time processing zero entries and therefore, the computational time is significantly raised. In the following, we will first discuss the time results from the analysis of the images belonging to the MR data set. The analyzed image types are of great clinical importance. The MR images of the human femur clearly show the trabecular bone tissue [see Fig. 3(a)]. Texture analysis reveals information about its metabolic state. This information is used in the assessment of the existence and degree of metabolic bone diseases, such as osteoporosis. Texture analysis of the MR brain images gives valuable information for brain tissue characterization, that is, whether tissue is normal or abnormal and, in the latter case, identification of the type of abnormality (e.g., brain tumor and brain edema). Also, the segmentation of white and gray matter in images of the brain is a problem of considerable medical significance.

Figs. 5(a) and 6(a) show the computational time results for the MR data set. For the images of the human femur, the co-occurrence matrix led the system to run out of memory. This was due to its enormous memory requirements. Only the co-occurrence trees allowed the computation of the textural features in this case. The average total time was about 2.4 min, whereas the average time per region was about 608 ms. For the second MR image type (MR images of the human brain—8-bit dynamic range), the co-occurrence matrix gave worse time results than the co-occurrence trees. The average total time was about 41 s, and the average time per region was about 255 ms. The corresponding times for the co-occurrence trees were about 10 s in average total time and 60 ms in average time per region. Almost the same results were given by the third MR image type (MR images of the human brain—12-bit dynamic range). The co-occurrence matrix again gave the worst results. The average total time was about 1.25 min, and the average time per region was 474 ms. For the co-occurrence trees approach, the average total time was about 13 s and the average time per region was about 82 ms.

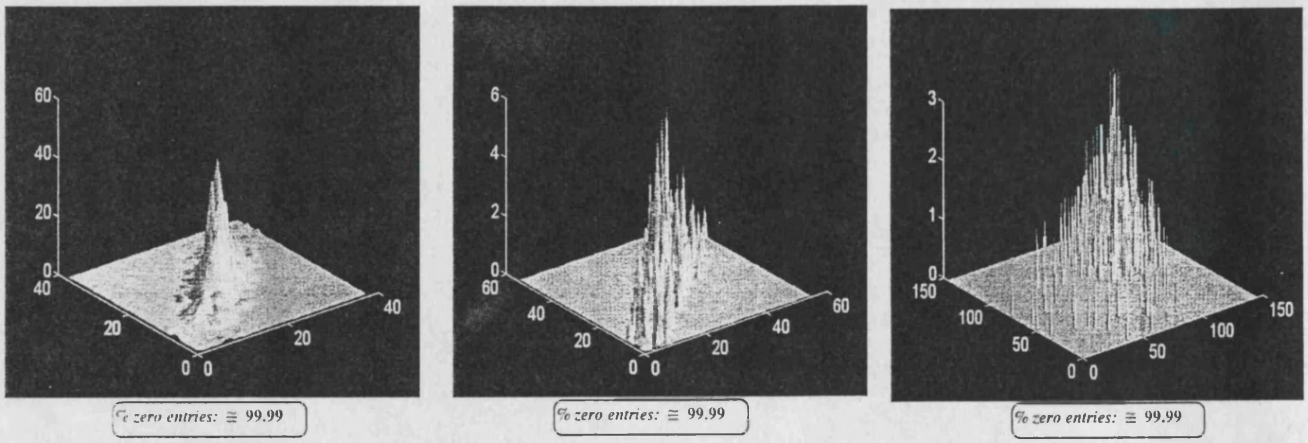
The following observations must be made about the MR data set. First, the percentage of the redundant zero entries in the co-occurrence matrix is very high (>99%) (see Fig. 7). This inefficiency is the main reason for either completely abandoning SGLDM or reducing the gray-level range in the image, losing valuable information though. This is also the reason for the worst time behavior of this approach.

From Fig. 8, it is clear that the percentage of the redundant zero entries approaches 100% (99.99%) for all images in the CT data set. Therefore, the space inefficiency of the co-



(a) (b) (c)

Fig. 7. Three-dimensional illustration and percentage of zero entries of typical co-occurrence matrices from the MR data set: (a) MR femur image (3-D illustration is not available), (b) MR brain image (8-bit dynamic range), and (c) MR brain image (12-bit dynamic range).



(a) (b) (c)

Fig. 8. Three-dimensional illustration and percentage of zero entries of typical co-occurrence matrices from the CT data set: (a) CT liver image (32 × 32 region size), (b) CT liver image (16 × 16 region size), and (c) CT brain image.

occurrence matrix is obvious in this case as well. Figs. 5(b) and 6(b) show the computational time results for this data set. The enormous memory requirements of the co-occurrence matrix led the system to run out of memory for all three CT image types. Only the co-occurrence trees allowed the computation of the textural features. For the first image type (CT liver; 32 × 32 region size), this method gave about 57 s in average total time and 355 ms in the average time per region. For the second image type (CT liver; 16 × 16 region size), it gave about 11 s in average total time and about 71 ms in the average time per region. For the third image type (CT brain), it gave about 43 s in average total time and about 268 ms in the average time per region.

Finally, some preliminary tests have been performed under the Windows NT (32-bit) operating system. All programs were run on a Pentium-S microprocessor at the speed of 133 MHz. The system had 32 MB of main memory. Both the co-occurrence matrix and the co-occurrence trees were tested on selected images from the MR and CT data set. The co-occurrence matrix was able to produce results for the three image types belonging to the CT data set. However, its time performance was much inferior to the corresponding performance of the co-occurrence trees. No significant differences were noticed for all other analyzed cases.

V. Conclusions

Overall, the results from the analyzed data sets clearly indicate that the co-occurrence trees method is much superior to the co-occurrence matrix. In particular, in medical image analysis, the co-occurrence matrix often leads the system to run out of memory (especially for medical images with dynamic ranges $\geq$12 bits). The very high percentage of zero entries indicates that most of the co-occurrence matrix stores redundant information, which not only contributes nothing to the textural feature computation process, but it also dramatically increases its memory requirements. The conclusion is that the co-occurrence matrix is inappropriate, especially for the analysis of medical images of a large dynamic range, which seem to contain texture information of the greatest diagnostic value. However, the superiority of the proposed approach, in terms of memory space and computational time, should be investigated for other imaging modalities as well. Preliminary results in the texture analysis of ultrasonic images and mammograms seem very promising.

REFERENCES

[1] R. A. Lerski, "Texture analysis of medical images," in *Proc. Int. Symp. Phys. Med. Imaging Adv. Comput. Applicat.*, 1993, pp. 64–68.

[2] J. Sklansky, "Medical radiographic image processing and pattern recognition," in *Proc. 5th Int. Conf. Pattern Recognit.*, 1980, pp. 374–382.

[3] R. W. Conners, C. A. Harlow, and S. J. Dwyer, "Radiographic image analysis: Past and present," in *Proc. 6th Int. Conf. Pattern Recognit.*, 1982, pp. 1152–1169.

[4] J. F. Desaga, J. Dengler, and T. Wolf, "Digitization of films and texture analysis for digital classification of pulmonary opacities," *Rontgenblatter*, vol. 41, pp. 147–151, 1988.

[5] M. F. McNitt-Gray, H. K. Huang, and J. W. Sayre, "Feature selection in the pattern classification problem of digital chest radiograph segmentation," *IEEE Trans. Med. Imag.*, vol. 14, pp. 537–547, Sept. 1995.

[6] C. B. Caldwell, S. J. Stapleton, and D. W. Holdsworth, "Characterization of mammographic parenchymal pattern by fractal dimension," *Phys. Med. Biol.*, vol. 35, pp. 235–247, 1990.

[7] P. Miller and S. Astley, "Classification of breast tissue by texture analysis," *Image Vision Comput.*, vol. 10, pp. 258–265, 1992.

[8] T. Lundahl, W. J. Ohley, and S. M. Kay, "Fractional Brownian motion: A maximum likelihood estimator and its application to image texture," *IEEE Trans. Med. Imag.*, vol. MI-5, pp. 152–161, Jan. 1986.

[9] J. Samarabandu, R. Acharya, and E. Hausmann, "Analysis of bone X-rays using morphological fractals," *IEEE Trans. Med. Imag.*, vol. MI-12, pp. 466–470, 1993.

[10] K. Chandrasekran, P. E. Aylward, and S. R. Fleagle, "Feasibility of identifying amyloid and hypertrophic cardiomyopathy with the use of computerized quantitative texture analysis of clinical echocardiographic data," *J. Amer. Col. Cardiol.*, vol. 13, pp. 832–840, 1989.

[11] C. M. Wu, Y. C. Chen, and K. S. Hsieh, "Texture features for classification of ultrasonic liver images," *IEEE Trans. Med. Imag.*, vol. 11, pp. 141–152, Jan. 1992.

[12] C. C. Chen, J. S. Daponte, and M. D. Fox, "Fractal feature analysis and classification in medical imaging," *IEEE Trans. Med. Imag.*, vol. 8, pp. 133–142, Jan. 1989.

[13] Y. N. Sun, M. H. Horng, and X. Z. Lin, "Ultrasonic image analysis for liver diagnosis: A noninvasive alternative to determine liver disease," *IEEE Eng. Med. Biol. Mag.*, vol. 15, pp. 93–101, 1996.

[14] Y. M. Kadah, A. A. Farag, and J. M. Zurada, "Classification algorithms for quantitative tissue characterization of diffuse liver disease from ultrasound images," *IEEE Trans. Med. Imag.*, vol. 15, pp. 466–478, Sept. 1996.

[15] S. M. Collins, D. J. Skorton, and N. V. Prasad, "Quantitative echocardiographic image texture: Normal contraction-related variability," *IEEE Trans. Med. Imag.*, vol. MI-4, pp. 185–192, Jan. 1985.

[16] D. T. Morris, "An evaluation of the use of texture measurements for the tissue characterization of ultrasonic images of *in-vivo* human placenta," *Ultrasound Med. Biol.*, vol. 14, pp. 387–395, 1988.

[17] R. A. Lerski, K. Straughan, and L. R. Schad, "MR Image texture analysis—An approach to tissue characterization," *Magn. Resolut. Imag.*, vol. 11, pp. 873–887, 1993.

[18] J. B. Wendling, A. Bruno, and P. Reuse, "MRI texture analysis applied to trabecular bone. An experimental study," in *Proc. 1st Int. Conf. CVRMed.*, 1995, pp. 439–443.

[19] P. T. Cahill, R. J. Knowles, and B. Kneeland, "Segmentation of white/gray matter by a texture based clustering algorithm," in *Proc. 6th Int. Conf. Pattern Recognit.*, 1982, pp. 633–636.

[20] A. H. Mir, M. Hanmandlu, and S. N. Tandon, "Texture analysis of CT images," *IEEE Eng. Med. Biol. Mag.*, vol. 14, pp. 781–786, 1995.

[21] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, May 1979.

[22] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-3, pp. 610–621, June 1973.

[23] K. L. Chan, "Quantitative characterization of electron micrograph image using fractal feature," *IEEE Trans. Biomed. Eng.*, vol. 42, pp. 1033–1037, Oct. 1995.

[24] R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, pp. 204–222, Jan. 1980.

[25] D. Terzopoulos and S. W. Zucker, "Texture discrimination using intensity and edge co-occurrences," in *Proc. IEEE 5th Int. Conf. Pattern Recognit.*, 1980, pp. 565–569.

[26] A. E. Svolos, C. A. Hutton, and A. Todd-Pokropek, "Co-occurrence trees: A dynamic solution for texture feature extraction," in *Proc. IEEE EMBS 18th Int. Conf. Eng. Med. Biol.*, 1996, pp. 1142–1144.

[27] K. Mehlhorn and A. Tsakalidis, "Data structures," in *Handbook of Theoretical Computer Science*. Amsterdam, The Netherlands: Elsevier, 1990, pp. 301–341.

[28] R. E. Tarjan, "Updating a balanced search tree in O(1) rotations," *Inf. Proc. Lett.*, vol. 16, pp. 253–257, 1983.

[29] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

Andreas E. Svolos (M'98) was born in Athens, Greece, on January 29, 1971. He received the Ptychion (five-year first degree) in computer engineering from the University of Patras, Patras, Greece, in 1993. He is currently pursuing the Ph.D. degree from the University College London, U.K., where he is writing his dissertation on space and time-efficient data structures in texture analysis.

He was a Research Associate in the fields of parallel signal and image processing at the Computer Technology Institute, Athens, Greece, from 1993 to 1995. His research interests are in the areas of data structure theory, image processing, and parallel and distributed processing.

Mr. Svolos is a member of ACM, SCS, and SPIE.

Andrew Todd-Pokropek (M'82) received the B.A. degree in physics from the University of Oxford, Oxford, U.K., and received a postgraduate degree from the University of London, London, U.K.

He is currently Professor of medical physics, a joint post created at the Department of Medical Physics, University College London, and the Department of Physics and Radiology, Institute of Child Health, and is he also Director of the INSERM Unit of Research U494 "Quantitative Medical Imaging," Paris, France. He started in the then rapid development of nuclear medicine instrumentation, was involved in the development of one of the first computer systems for use in medical imaging, and participated in the design of the first SPECT system in Europe. He has become increasingly involved with medical image processing and data handling, first in nuclear medicine and later in radiology in general. His areas of research extend from finding solutions for inverse problems to defining standards for medical image transfer to developing expert systems for use in medicine. He has been widely published and invited to lecture by many different national and international organizations and has been awarded a number of honors.

the level of diagnostic information extracted from images of most medical imaging modalities and to quantitatively characterise differences in appearances inaccessible to human observers. Texture analysis is of great importance in the diagnostic process for many diseases.[4] The extraction of features which characterise the texture of the lungs and the bones is a very important process in chest analysis.[5] Medical applications of computer-based texture analysis date back to the early 1970s, when it became possible to digitise X-rays for computer processing. The explosion in the use of digital techniques has led to the possibility of utilising texture analysis for most medical imaging modalities. Typical work has been carried out on digital X-rays,[6, 7, 8] ultrasound images,[9, 10, 11] magnetic resonance (MR) images[12, 13] and X-ray CT images.[14, 15]

Magnetic Resonance imaging has received considerable attention in the medical field because it is an invasive, high resolution and nonhazardous imaging modality. It can be used to produce cross-sectional tomographic images in transverse, coronal, sagittal and oblique planes with excellent soft tissue differentiation. These images are routinely used to diagnose widely varying pathological conditions. The human brain has especially been a primary area of interest in clinical MR. The MR images of the brain show very good contrast between grey and white matter, cerebrospinal fluid (CSF) and other cerebral structures. Significant changes in the contrast between grey and white matter are seen in pathological conditions, such as in the presence of tumour, oedematous tissue, etc. MR imaging has the potential to produce high resolution images that may be extremely useful in the diagnosis and monitoring of pathological and developmental disorders. However, before such diagnoses can be made it is essential to quantitatively describe and characterise properties like texture in MR images. MR images of the human femur, wrist and vertebrae can give invaluable information about the state of the bone tissue. The assessment of the existence and degree of metabolic bone diseases, like osteoporosis, is of great clinical importance. This can be done by characterising the trabecular bone structure. Texture analysis of MR images can give quantitative information regarding this structure. Especially, the analysis of MR bone images of a very large dynamic range (≥ 12 bits) can be very useful in characterising the trabecular structure, as accurately as possible.

Syntactic texture analysis methods are not as highly developed in the case of medical images analysis as the statistical methods. Statistical texture analysis methods can extract textural information from the spatial distribution of the grey levels which is invaluable in automatic tissue characterisation and diagnosis. It has been claimed that statistical texture analysis is of great value in biomedical image analysis.[16] One of the most significant statistical texture description methods, which has found wide application in this area, is the Spatial Grey Level Dependence Method (SGLDM). It has been claimed that it is a superior general purpose method for texture analysis.[5] SGLDM [2, 17] characterises the texture in an image region by means of features derived from the spatial distribution of pairs of grey levels (second-order distribution), having certain interpixel distances (separations) and orientations. It is based on the assumption that texture information is contained in the overall spatial relationship which the grey levels in an image region have to one another.

However, the co-occurrence matrix,[2] which is used for storing the estimated second order probabilities in SGLDM, is inefficient in terms of memory requirements and computational time for the calculation of textural features. Many of the researchers used this method were forced to reduce the grey level range of the image so that they could handle the large requirements of the co-occurrence matrix. Doing so, however, they lost significant texture information which would have led to better texture discrimination. Low computational complexity and high classification accuracy are two of the most important considerations for the classification of medical images.[9] SGLDM is one of the worst methods in terms of speed, since its computational time complexity depends on the size of the co-occurrence matrix, or else, the number of grey levels in the entire image. The inefficiency of the co-occurrence matrix is due to this dependency. This fact most often causes entire rows and (or) columns of the matrix to contain zero entries which contribute nothing to the texture feature computation process.

In this paper, we investigate the memory space and computational time efficiency of a number of dynamic data structures called self-adjusting binary search trees, in replacing the co-occurrence matrix. The rest of the paper is organised as follows: In Section 2, we describe the dynamic approach. In Section 3, we present space and time results from the analysis of MR images of the human brain and the human femur, using both approaches, i.e., the co-occurrence matrix and self-adjusting binary search trees. In Section 4, we discuss the results of Section 3. We conclude in Section 5 discussing the usefulness of the proposed approach in a clinical application environment.

Self-adjusting binary search trees: An investigation of their space and time efficiency in texture analysis of magnetic resonance images using the spatial grey level dependence method

Andreas E. Svolos and Andrew T. Pokropek

Department of Medical Physics, UCL, London, WC1E 6JA, UK

ABSTRACT

Texture feature extraction is a fundamental stage in texture analysis. Therefore, the reduction of its computational time and memory requirements should be an aim of continuous research. The Spatial Grey Level Dependence Method (SGLDM) is one of the most significant statistical texture description methods, especially in medical image analysis. However, the co-occurrence matrix is inefficient in terms of time and memory requirements. This is due to its dependency on the number of grey levels in the entire image. Its inefficiency puts up barriers to the wider utilisation of the SGLDM in a real application environment.

This paper investigates the space and time efficiency of self-adjusting binary search trees, in replacing the co-occurrence matrix. These dynamic data structures store only the significant textural information extracted from an image region by the SGLDM. Furthermore, they have the ability to restructure themselves in order to adapt to the co-occurrence distribution of the grey levels in the analysed region. This results in a better time performance for texture feature extraction.

The proposed approach is applied to a number of magnetic resonance images of the human brain and the human femur. A comparison with the co-occurrence matrix, in terms of space and computational time, is performed.

Keywords: texture analysis, co-occurrence matrix, spatial grey level dependence method, self-adjusting binary search trees, medical image analysis

1. INTRODUCTION

Texture is a fundamental feature for image analysis, classification and segmentation. Its usefulness has been proved for many types of images, ranging from multispectral remote sensed aerial and satellite data to biomedical images. There is no formal mathematical definition of what texture is. This is due to its wide variability. Texture is a property that characterises almost all surfaces,[1,2] e.g., the grain of wood, the weave of a fabric. Two major categories of texture analysis exist: statistical and syntactic or structural. Statistical methods employ statistics computed from the image data which characterise the distribution of the grey levels in the analysed image region. Features extracted by statistical methods usually measure textural characteristics like coarseness, contrast and directionality. Syntactic or structural methods describe texture by means of primitive descriptions and primitive placement rules. Primitives can be geometrical objects like line segments, open polygons or closed polygons. Primitive placement rules describe the spatial relationships of primitives.

In medical image analysis, texture has been successfully used for tissue characterisation, i.e., the determination of tissue type (normal or pathological) and further classification of tissue pathology.[3] Texture analysis has been shown to increase

Further author information -
A.E.S.(correspondence): Email: rmapasv@medphys.ucl.ac.uk; Telephone: ++44 171 209 6462; Fax: ++44 171 209 6269
A.T.P.: Email: atoddpok@medphys.ucl.ac.uk

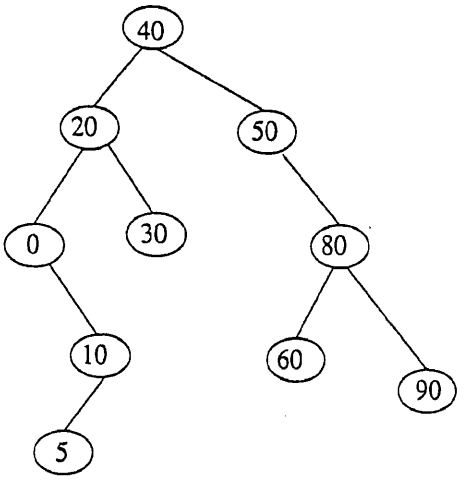


Fig. 2: An example of a self-adjusting tree.

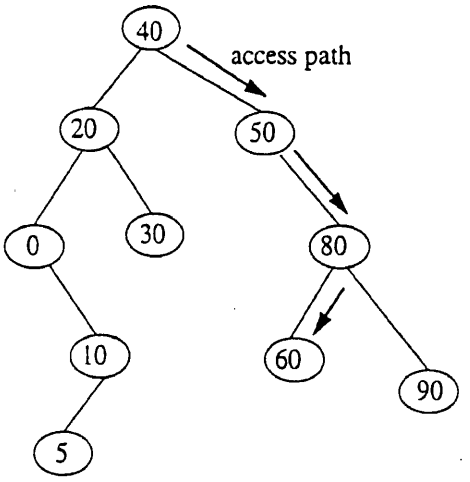


Fig. 3: The path for accessing element 60.

We would like to access element 60 in the illustrated tree. The path followed by the access algorithm is shown in Fig. 3. After the successful access of element 60, the splaying restructuring operation is performed at the node containing this element. The structure of the tree after splaying is shown in Fig. 4. Note that the root node now contains element 60.

In texture analysis, only the primitive operations of the search trees access (search) and insert are of interest. In Fig. 5 and 6, we present the pseudocode for these operations in the case of the self-adjusting binary search trees.

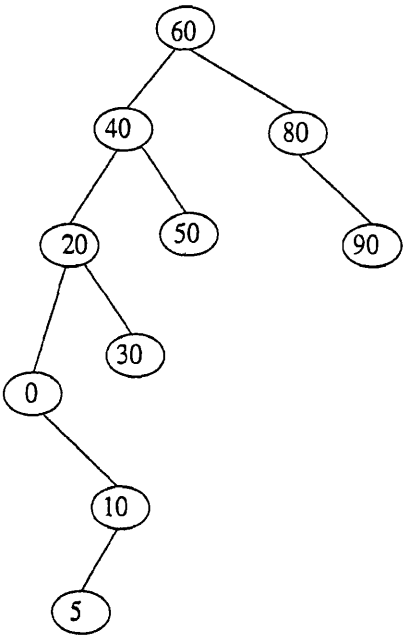


Fig. 4: The self-adjusting tree after splaying at the node containing element 60.

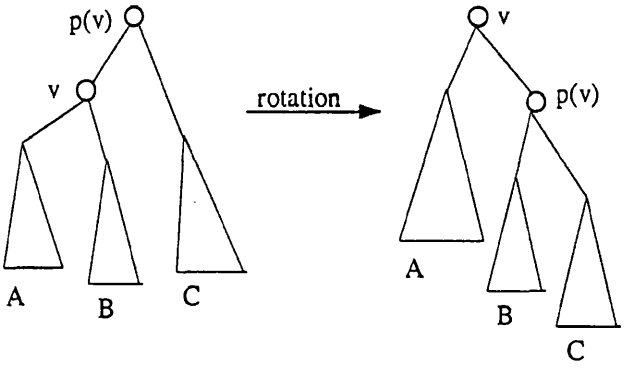


Fig. 1: Illustration of the single rotation operation.

2. SELF-ADJUSTING CO-OCCURRENCE TREES

SGLDM, as we have already mentioned in the previous section, is based on the computation of second order statistics from the spatial distribution of grey level pairs in a local region of the image (region of interest). First, it estimates the second order probabilities $p(i,j;d,\vartheta)$, which are the probabilities of having grey level pair (i,j) at an interpixel distance d and orientation ϑ, for each (i,j) in the analysed region.[2] Instead of storing the above estimated probabilities in a matrix form (co-occurrence matrix), we can create a dynamic data structure (e.g., a binary search tree [18]) for each grey level i and put a node in it that corresponds to its paired grey level j. In this way, we construct a set of dynamic data structures for the storage of the estimated second order probabilities.

A binary search tree that is appropriate for the above purpose is the self-adjusting binary search tree. This tree is a search tree which means that it stores a set of elements S, one element in each node, and the primitive operations search, insert and delete can be defined on it.[18] It is a binary tree because each node has at most two children. The unique property of the self-adjusting tree is its ability to dynamically adapt to the access distribution (access frequency) of the elements in set S. This adaptation is performed by a sequence of restructuring operations which push the most frequently accessed elements towards the root. After a sufficiently long sequence of accesses, these elements are as close to the root as possible. Since the algorithm for accessing an element in a search tree starts from its root and proceeds towards the leaves, until it finds the given element or it reaches the bottom of the tree (its leaves), it is obvious that the closer to the root the most frequently accessed elements are the more time efficient the access operation is.

The restructuring operation mentioned above can be defined in several different ways. In this paper, we investigate the efficiency of a restructuring operation called splaying.[19] This operation consists of a sequence of rotations along the path from the accessed node (the node that contains the requested element) to the root, which move the accessed element up to the root of the tree. A rotation is a primitive operation which changes the structure of the tree locally (see Fig. 1). In this figure, v is a node of the tree and $p(v)$ is its parent, i.e., the next node on the path from node v to root. The symmetric variant of the illustrated case is omitted. Actually, splaying restructuring operation consists of three types of primitive operations, namely zig, zig-zig, and zig-zag.[19] Each of these primitive operations consists of one or two rotations. Splaying not only moves the node storing the requested element up to the root, but also it roughly halves the depth of every node along the access path. This halving effect is a very strong property because it reduces the worst case access time in the tree. This property makes splaying one of the most efficient restructuring operations in terms of computational time. We will now show an example of the splaying restructuring operation. Suppose that we store set $S = \{0,5,10,20,30,40,50,60,80,90\}$ in a self-adjusting binary search tree. The corresponding tree is shown in Fig. 2.

b) the full-dynamic version, where an additional self-adjusting binary tree is employed for the storage of the pointers to the roots.

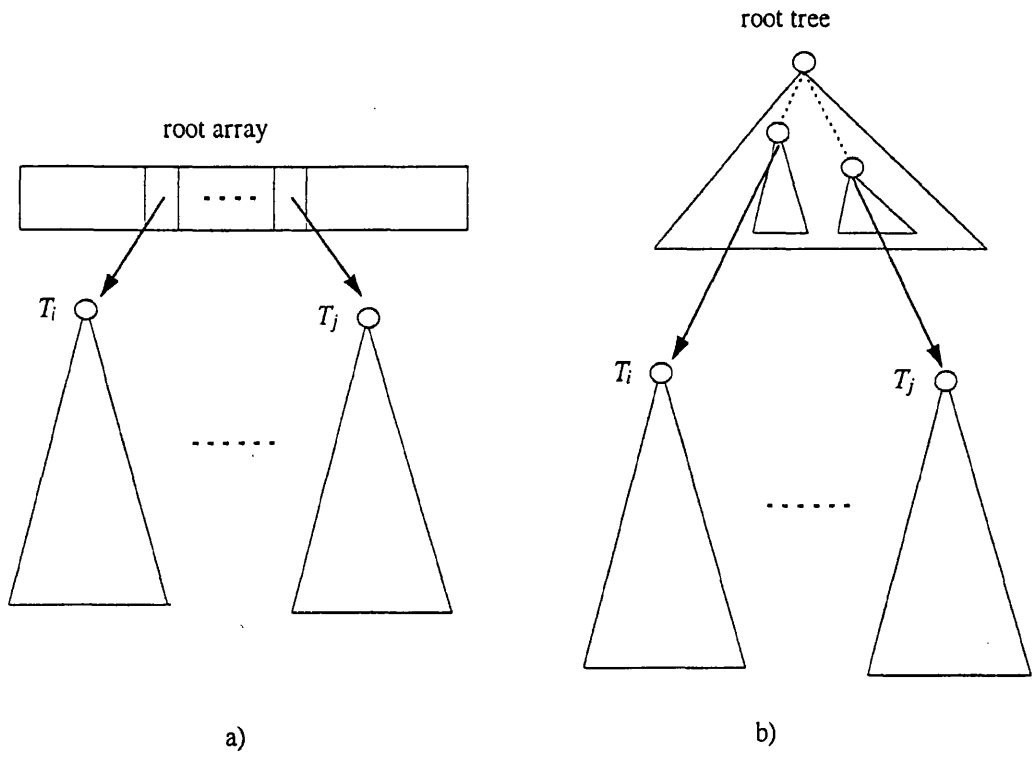


Fig. 7: Self-adjusting co-occurrence trees; a) semi-dynamic version, b) full-dynamic version.

3. RESULTS

In this section, we first present the memory space requirements of the co-occurrence matrix and we compare them to the corresponding requirements of the self-adjusting co-occurrence trees, for various values of N_g (the number of grey levels in the entire image). Fig 8 a), b) and c) show the memory space in bytes of the co-occurrence matrix, the semi-dynamic version and the full-dynamic version, respectively. Again, n_g is the number of grey levels in the analysed region of interest. The illustrated memory requirements of the self-adjusting co-occurrence trees are in the worst case, that is, when n_g^2 grey level pairs satisfy the given spatial relationship (d,ϑ) in the analysed image region. Fig. 9 a) and b) show the percentage reduction in memory space of the semi-dynamic version and the full-dynamic version respectively, relative to the memory space of the co-occurrence matrix. Negative values mean that for the specific (N_g,n_g) pair, co-occurrence matrix is more efficient in terms of space.

In the following, we present computational time results from the analysis of real clinical image data for each one of the compared approaches, namely the co-occurrence matrix, the semi-dynamic version of the self-adjusting co-occurrence trees and the corresponding full-dynamic version. Three data sets of MR images were employed for texture analysis using SGLDM. Each data set consisted of 10 images. The first data set illustrated the human femur (see Fig. 10 a)). The analysed images had a 15 bit dynamic range and their spatial resolution was 512 x 512. 30 regions from each image were selected for texture analysis. The size of each region was 32 pixels. The second data set illustrated the human brain (see Fig. 10 b)). All images had an 8 bit dynamic range and their spatial resolution was 256 x 256. In this data, 20 regions of

```
Access(x) :  v ← root;
             WHILE (val(v) ≠ x AND v is not a leaf) DO
                 IF (val(v) > x)
                     v ← lson(v);
                 ELSE
                     v ← rson(v);
                 ENDIF
             ENDWHILE
             IF (val(v) ≠ x)
                 RETURN (false, v);
             ELSE
                 splay(v)
                 RETURN (true, v);
             ENDIF
```

Fig, 5: Pseudocode for the access primitive operation.

```
Insert (x, inf) :  found ← Access*(x);
                   IF (found.1 = TRUE)
                       update(inf, found.2);
                       splay(found.2);
                   ELSE
                       splay(add(x, found.2, inf));
                   ENDIF
```

Fig. 6: Pseudocode for the insert primitive operation.

In the above pseudocodes, x is the element we would like to access in the tree, *root* is the root of the tree and v is a variable that stores the currently accessed node. Variable *found* is a structure variable having two fields. The first field gives the information whether the requested element was found in the tree or not and the second field gives the last accessed node in the tree. Of course, if the requested element was found, this field contains the node that stores this element. Variable *inf* contains the information that we would like to store in the accessed node, or the new node if element x was not found in the tree. This information, in our case, is the new co-occurrence frequency of the accessed grey level pair (see below). Function *val*(·) returns back the element that is stored in the currently accessed node. Functions *lson*(·) and *rson*(·) return the left child and the right child of the currently accessed node, respectively. The reader should recall that the self-adjusting tree we are presenting in this paper is a binary tree. Subroutine *splay*(·) performs the actual splaying operation at the accessed node.In the body of the *Insert*(·) routine, the *splay*(·) subroutine is explicitly called so that *Access*(·) routine does not have to do that. *Access**(·) routine is actually the routine illustrated in Fig. 5 but without the spalying step. Finally, the auxiliary functions *update*(·) and *add*(·) perform what their name implies, i.e., *update*(·) changes the content of the currently accessed node while *add*(·) stores the given information in the new inserted node.

Self-adjusting co-occurrence trees comprise a forest (set) of self-adjusting binary search trees which can replace the co-occurrence matrix in SGLDM. Let n_g denote the number of grey levels in the analysed region of interest. The above set contains at most n_g self-adjusting trees, one tree for each grey level. Let T_i be the tree for grey level i. Then, for each pair (i,j) of grey levels that satisfies the (d,ϑ) spatial dependence relation in the analysed region, there is a node in T_i that stores the relevant information. This information consists of the grey level j, the co-occurrence frequency $P_{(i,j)}$ (number of occurrences of pair (i,j) in the analysed region, according to (d,ϑ) spatial relation) and two pointers (one to the left child and the other to the right child of the node, if they exist). These pointers are used for building the corresponding tree that contains the above node.

The pointers to the roots of the self-adjusting trees of the proposed approach are stored in a separate data structure. Two versions of the self-adjusting co-occurrence trees exist, depending on whether this structure is static or dynamic (see Fig. 7 a) and b):

a) the semi-dynamic version, where the above structure is a static array,

All algorithms ran on a Pentium Dell P100t personal computer.The texture analysis of each selected region included the computation of the complete set of textural features (13 in number),[2] with the exception of the human femur data set. We excluded the computation of some texture features from the analysis of this data set, namely, sum entropy, sum variance, difference entropy and second information of correlation (the 13th textural feature).[2] The reason was that these features were very time consuming for images of such a large dynamic range (15 bits). This exclusion is not important since the above texture features are rarely used in texture analysis. The average total time for the analysis of the regions in each image was computed for each data set. The corresponding time for each of the compared approaches is illustrated in Fig. 11 a). In addition, an averaging over time was performed and the average time per region was estimated, again for each data set. The corresponding results are shown in Fig. 11 b). Whenever there is a column of zero height in the above illustrations it means that the corresponding approach led the system to run out of memory. Also, "mtx" denotes the co-occurrence matrix approach, "sd" denotes the semi-dynamic version of the proposed approach and "fd" the full-dynamic version of this approach. Both the average total time and the estimated average time per region comprise reliable and robust time measurements on which a comparison between the co-occurrence matrix and the two versions of the self-adjusting co-occurrence trees, can be based.

	n_g									
N_g	2	4	8	16	32	64	128	256	512	1024
8	46.88	-37.50	-375.00	-	-	-	-	-	-	-
16	80.47	59.38	-25.00	-362.50	-	-	-	-	-	-
32	91.99	86.72	65.63	-18.75	-356.25	-	-	-	-	-
64	96.44	95.12	89.84	68.75	-15.63	-353.13	-	-	-	-
128	98.33	98.00	96.68	91.41	70.31	-14.06	-351.56	-	-	-
256	99.19	99.11	98.78	97.46	92.19	71.09	-13.28	-350.78	-	-
512	99.60	99.58	99.50	99.17	97.85	92.58	71.48	-12.89	-400.39	-
1024	99.80	99.80	99.78	99.69	99.37	98.05	92.77	71.68	-25.20	-400.20

a)

	n_g									
N_g	2	4	8	16	32	64	128	256	512	1024
8	46.88	-62.50	-450.00	-	-	-	-	-	-	-
16	86.72	59.38	-37.50	-400.00	-	-	-	-	-	-
32	96.68	89.84	65.63	-25.00	-375.00	-	-	-	-	-
64	99.17	97.46	91.41	68.75	-18.75	-362.50	-	-	-	-
128	99.79	99.37	97.85	92.19	70.31	-15.63	-356.25	-	-	-
256	99.95	99.84	99.46	98.05	92.58	71.09	-14.06	-353.13	-	-
512	99.99	99.96	99.87	99.51	98.14	92.77	71.48	-13.28	-401.56	-
1024	100.00	99.99	99.97	99.88	99.54	98.19	92.87	71.68	-25.39	-400.78

b)

Fig. 9: Relative percentage reduction in memory space; a) semi-dynamic version, b) full-dynamic version.

4. DISCUSSION

Of particular interest are Fig. 9 a) and b) which show the percentage reduction in memory space requirements of the proposed approach compared to the co-occurrence matrix. Results from the semi-dynamic and the full-dynamic version are illustrated separately. From the above figures, it is clear that as the number of grey levels in the image (N_g) increases, the percentage reduction in memory requirements increases dramatically, due to the large number of zero entries in the co-occurrence matrix. In fact, for $N_g \geq 128$ reduction is more than 90 %, in most cases. Only when the number of grey levels

size 8 x 8 were selected from each image for further analysis. The third data set again illustrated the human brain (see Fig. 10 c)), but now the dynamic range was 12 bits. The spatial resolution was again 256 x 256. 20 regions of size 10 x 10 were selected from each image for analysis. All the above images came from the database of the department of Medical Physics, UCL. All analysed region parameters (size and location on the image) have been chosen under the guidance of an experienced radiologist and with the aim to cover as much of the clinically interesting texture in the analysed image, as possible.

N_g	Space in bytes
8	128
16	512
32	2048
64	8192
128	32768
256	131072
512	524288
1024	2097152

a)

n_g	Space in bytes
2	68
4	208
8	704
16	2560
32	9728
64	37888
128	149504
256	593920
512	2629632
1024	10502144

c)

	n_g									
N_g	2	4	8	16	32	64	128	256	512	1024
8	68	176	608	-	-	-	-	-	-	-
16	100	208	640	2368	-	-	-	-	-	-
32	164	272	704	2432	9344	-	-	-	-	-
64	292	400	832	2560	9472	37120	-	-	-	-
128	548	656	1088	2816	9728	37376	147968	-	-	-
256	1060	1168	1600	3328	10240	37888	148480	590848	-	-
512	2084	2192	2624	4352	11264	38912	149504	591872	2623488	-
1024	4132	4240	4672	6400	13312	40960	151552	593920	2625536	10489856

b)

Fig. 8: Memory space requirements; a) co-occurrence matrix, b) semi-dynamic version, c) full-dynamic version.

An equivalent expression for the spatial relationship is the displacement vector form $d=(d_x,d_y)$, where d_x is the distance of the two grey levels in each pair in rows and dy is their distance in columns. Actually, it is the Cartesian equivalent of the (d,ϑ) polar co-ordinates. The reason that this form was chosen is that it can be processed much more easily by computers. 8 displacement vectors were employed for the texture analysis of each selected region, i.e., (0,1), (1,0), (1,1), (1,-1), (0,2), (2,0), (2,2), and (2,-2). These vectors are the most commonly employed in the analysis of medical images using SGLDM.

Prior to analysis, all images were pre-processed using histogram equalisation.[20] This normalisation is necessary since a textured image should be independent of the first order statistics of the grey levels in it. These statistics are sensitive to variations in image acquisition conditions, such as lighting. Normalising the data set has been shown to improve the overall diagnostic accuracy in medical image analysis and classification.[10]

difference in speed is small. This difference is due to the faster access to the structure that keeps the roots of the self-adjusting trees, in the case of the semi-dynamic version. Accessing an element in a tree is slower than accessing the same element in a static array. Because the dynamic range of the analysed images was small, the number of zero entries should have been much smaller. Therefore, there was no processing of much redundant information in the root array of the semi-dynamic version and thus, there was no advantage of replacing it with a dynamic data structure like a tree. On the contrary, the larger access time to the root tree was the dominant factor in the time performance of the full-dynamic version, forcing it to run slower.

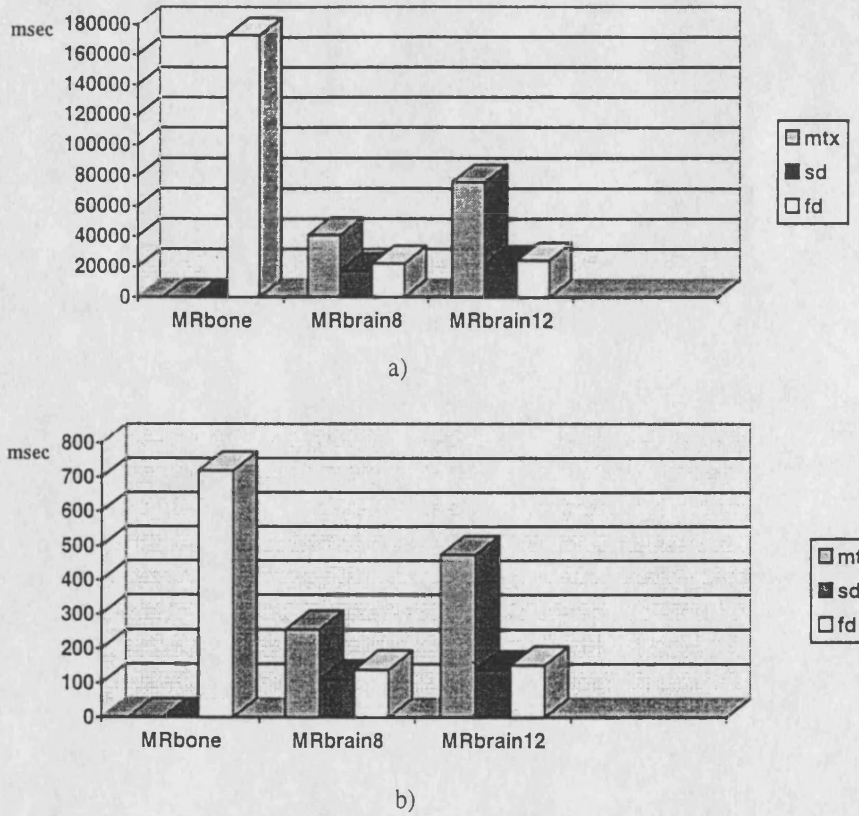
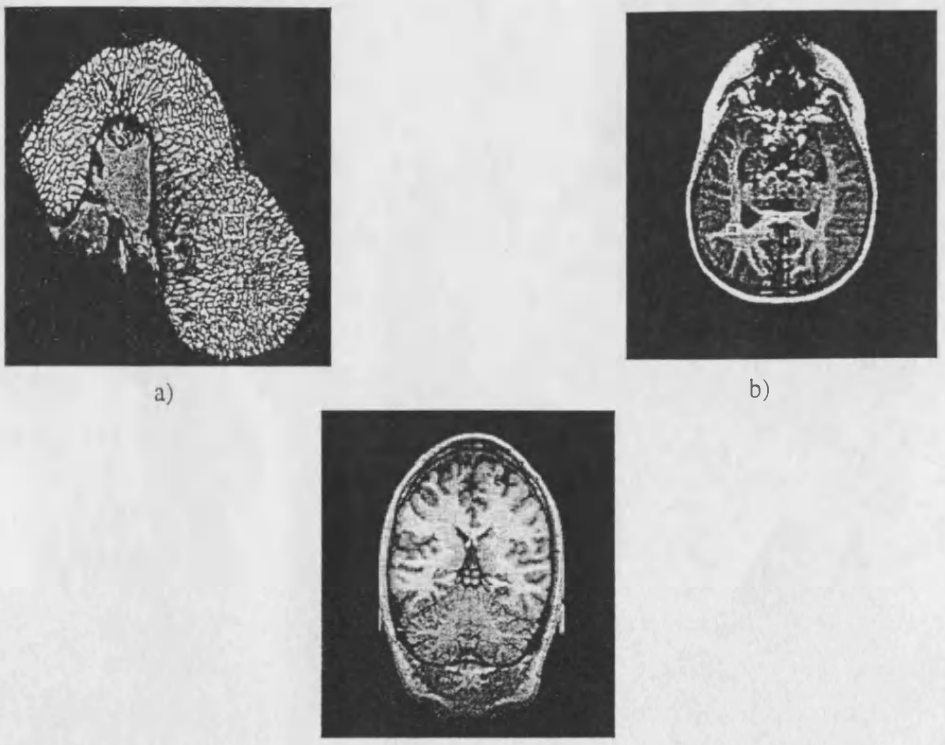


Fig. 11: Time results from the analysed data sets; a) average total time, b) average time per region.

The above become more clear in the case of the 12 bit MR brain data set. In this case, the co-occurrence matrix again exhibited the worst time behaviour, scoring about 76 sec in average total time and about 474 msec in the average time per region. Comparing these results with the corresponding ones in the previous case we can clearly see that its behaviour became worse. This fact should not surprise us since the dynamic range is now larger and therefore, the number of the redundant zero entries in the co-occurrence matrix should be larger, too. Semi-dynamic version gave the best results, about 22 sec in average total time and about 138 msec in average time per region. The full-dynamic version gave about 24 sec in average total time and about 152 msec in average time per region. Of particular interest here is the time difference between the two versions of the proposed approach. It is now about 2 sec in average total time and about 14 msec in average time per region. Comparing these results with the corresponding ones in the previous case we can see that the time difference became less, although the actual computational times increased. The reason is that in this case, the root array of the semi-dynamic version contained more redundant information. Therefore, the advantage of using a tree in its place, as in the case of the full-dynamic version, becomes stronger.

in the analysed region (n_g) is half the number of grey levels in the entire image or more, the co-occurrence matrix shows better behaviour in terms of space. Also, even when $N_g < 128$, in most cases, the reduction in memory space achieved by using the proposed approach is significant.



a) b)

c)

Fig. 10: Images of the analysed data sets. a) MR image of the human femur, b) MR image of the human brain (8 bit dynamic range), c) MR image of the human brain (16 bit dynamic range).

Fig. 11 a) and b) illustrate the average total time and the average time per region for each of the analysed data sets. We will begin our discussion with the results from the first data set. Since this set consists of images having a very large dynamic range (15 bits), the space inefficiency of the co-occurrence matrix should be very large, too. Only a few entries should be nonzero relative to the total number of entries. Therefore, the texture analysis algorithm should spend almost all its time to process zero entries that contribute nothing to the calculation of the textural features. In reality, though, the co-occurrence matrix approach led the system to run out of memory, due to its enormous memory requirements for these images. In such cases we have two options. We either completely abandon SGLDM or we reduce the grey level range in the images, losing though valuable diagnostic information. Even the semi-dynamic version of the proposed approach had the same result. This is because a part of this version is static (the root array; see Fig. 7 a)) and therefore its memory space requirements depend on the number N_g of the grey levels in the entire image. The only method that allowed the computation of the textural features was the full-dynamic version. The average total time was about 2.9 min whereas the average time per region was 720 msec.

In the case of the data set that contains the MR brain images of 8 bit dynamic range, the co-occurrence matrix again gave the worst time performance. The average total time was about 41 sec whereas the average time per region was 255 msec. The semi-dynamic version gave about 18 sec in average total time and 110 msec in the average time per region. The full-dynamic version scored in between, giving about 22 sec in average total time and about 139 msec in the average time per region. The time difference between the two versions of the self-adjusting co-occurrence trees was 4.6 sec in average total time and about 29 msec in the average time per region. Since the dynamic range of the analysed images in this data set is now much smaller (only 8 bits), the semi-dynamic version was faster than the full-dynamic version. However, the

15. A. H. Mir, M. Hanmandlu, and S. N. Tandon, "Texture analysis of CT images", *IEEE Eng. Med. Biol. Mag.* **14**, pp. 781-786, 1995.

16. K. L. Chan, "Quantitative characterization of electron micrograph image using fractal feature", *IEEE Trans. Biomed. Eng.* **BME-42**, pp. 1033-1037, 1995.

17. R. W. Conners and C. A. Harlow, "A theoretical comparison of texture algorithms", *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-2**, pp. 204-222, 1980.

18. K. Mehlhorn and Tsakalidis, "Data structures", *Handbook of Theoretical Computer Science*, pp. 301-341, Elsevier Science Publ., 1990.

19. D. D. Sleator and R. E. Tarjan, "Self-adjusting binary search trees", *J. ACM* **32**, pp. 652-686, 1985.

20. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, 1989.

5. CONCLUSIONS

Overall, the results from the three analysed data sets presented in the previous section clearly indicate that the self-adjusting co-occurrence trees approach is much superior to the co-occurrence matrix. In particular, we saw that for medical images having a very large dynamic range, the co-occurrence matrix led the system to run out of memory. Abandoning SGLDM is not the best solution since it has been proved to be one of the best texture analysis methods. On the other hand, reducing the number of grey levels leads to loss of textural information which might be significant for diagnostic purposes (e.g., assessing the stage of a disease, like osteoporosis, from the MR image as accurately as possible). Even in the case of MR images with small dynamic range (8 bits) the proposed approach was far better.

Comparing the semi-dynamic version with the full-dynamic version, the latter is more appropriate for MR images with a large dynamic range, because the percentage of the redundant zero entries is very high. We saw that in the analysis of the first data set only this version was able to produce results. Semi-dynamic version was the fastest approach in the case of the other two data sets. Especially for images with 8 bit dynamic range or less, the semi-dynamic version should be preferred.

6. REFERENCES

1. R. M. Haralick, "Statistical and structural approaches to texture", *Proc. IEEE* **67**, pp. 786-804, 1979.

2. R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification", *IEEE Trans. Sys. Man Cybern.* **SMC-3**, pp. 610-621, 1973.

3. R. A. Lerski, "Texture analysis of medical images", *Proc. Int. Symp. on Physics of Medical Imaging and Advances in Computer Applications*, pp. 64-68, 1993.

4. J. Sklansky, "Medical radiographic image processing and pattern recognition", *Proc. 5th Int. Conf. on Pattern Recognition*, pp. 374-382, 1980.

5. R. W. Conners, C. A. Harlow, and S. J. Dwyer, "Radiographic image analysis: past and present", *Proc. 6th Int. Conf. on Pattern Recognition*, pp. 1152-1169, 1982.

6. E. L. Hall, W. O. Crawford, and F. E. Roberts, "Computer classification of pneumoconiosis from radiographs of coal workers", *IEEE Trans. Biomed. Eng.* **BME-22**, pp. 518-527, 1975.

7. M. F. McNitt-Gray, H. K. Huang, and J. W. Sayre, "Feature selection in the pattern classification problem of digital chest radiograph segmentation", *IEEE Trans. Med. Imag.* **MI-14**, pp. 537-547, 1995.

8. J. Samarabandu, R. Acharya, E. Hausmann, and K. Allen, "Analysis of bone X-rays using morphological fractals", *IEEE Trans. Med. Imag.* **MI-12**, pp. 466-470, 1993.

9. C. M. Wu, Y. C. Chen, and K. S. Hsieh, "Texture features for classification of ultrasonic liver images", *IEEE Trans. Med. Imag.* **MI-11**, pp. 141-152, 1992.

10. Y. M. Kadah, A. A. Farag, et al., "Classification algorithms for quantitative tissue characterization of diffuse liver disease from ultrasound images", *IEEE Trans. Med. Imag.* **MI-15**, pp. 466-478, 1996.

11. U. Raeth, D. Schlaps, et al., "Diagnostic accuracy of computerized B-scan texture analysis and conventional ultrasonography in diffuse parenchymal and malignant liver disease", *J. Clin. Ultrasound* **13**, pp. 87-99, 1985.

12. R. A. Lerski, K. Straughan, et al., "MR Image texture analysis - an approach to tissue characterization", *Magn. Res. Imag.* **11**, pp. 873-887, 1993.

13. J. Bezy-Wendling, A. Bruno, and P. Reuse, "MRI Texture analysis applied to trabecular bone. An experimental study", *Proc. 1st Int. Conf. CVRMed*, pp. 439-443, 1995.

14. P. T. Cahill, R. J. Knowles, B. Kneeland, and B. C. Lee, "Segmentation of white/gray matter by a texture based clustering algorithm", *Proc. 6th Int. Conf. on Pattern Recognition*, pp. 633-636, 1982.

frequencies respectively, where $p_1 \geq p_2 \geq \ldots \geq p_k > 0$ and $\sum p_i = 1$. In addition, let $p_i = 1/i \cdot H_k$, where $H_k = \sum_{i=1}^{k} 1/i$ (Zipf's Law) ([16]). If the elements are stored according to the uniform distribution assumption (i.e. in random order, ignoring their access frequencies) the expected cost $E(c)$ of an access operation is $E(c) = \frac{1}{k}\sum_{i=1}^{k} i = \frac{k+1}{2}$. On the contrary, if the elements are stored according to order $(n_1, n_2, \ldots, n_k)$, i.e. n_i is stored in the i_{th} node of the list, the above cost becomes $E(c) = \frac{k}{H_k} \cong \frac{k}{\ln k}$ ([16]), which is much lower. This variant of the *dictionary problem*, which takes into account the access distribution of the stored elements is called the *weighted dictionary problem* ([17]). Since real texture is not a completely random process (noise), i.e. the pixels are spatially correlated, *weighted dictionary problem* is expected to show better performance. In this paper we investigate this possibility through a number of methods proposed in the following section.

II. THE PROPOSED SCHEME

An approach is presented which enhances the co-occurrence trees by adding a list of grey levels p_list(i) for each grey level i in the analysed textured region (see Fig. 1). The length of every list is K which is a parameter of our approach. The idea is to keep in each p_list(i) those grey levels j which have the K largest estimated probabilities $p(i, j | d, \theta)$. For each of these grey levels there is a pointer in p_list(i) to the corresponding node in tree T_i of the co-occurrence trees, if such a node exists. Whenever we need to access the (i, j) grey level pair in the co-occurrence trees, we first check if j is in p_list(i). If it is and there exists a pointer to tree T_i we simply follow this pointer. Otherwise, we proceed the same as in the co-occurrence trees approach. The main problem here is the determination of the K grey levels j for each i. Three rules are investigated in this paper:

a) the static rule, where the K grey levels are determined *a-priori*, according to a given co-occurrence distribution,

b) the move to front rule, where the grey level j that has been most recently accessed in tree T_i and its $K-1$ closest neighbours in T_i comprise the new p_list(i),

c) the counter rule, which is similar to the move to front rule but now the grey levels j in the new p_list(i) are sorted in descending order, according to their access frequencies.

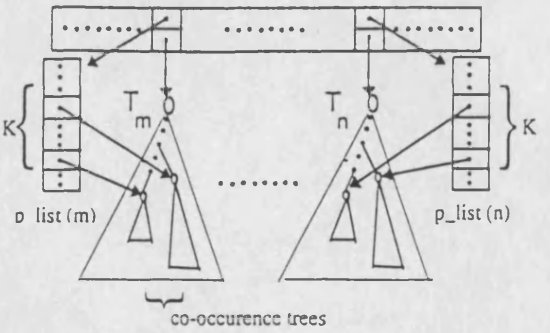


Figure 1: The proposed structure

In the first rule, the K grey levels with the largest co-occurrence probabilities are determined using the Gaussian distributed texture model described in ([18]). Note that the estimated co-occurrence probabilities are equal to the normalised access frequencies of the grey level pairs in the analysed textured region. The move to front and counter rule are very well known methods in the *weighted dictionary problem* ([16]). Another parameter, ρ, is associated with these two methods that determines how often the rules take action. The larger the parameter ρ, the more unlikely is an update in one of the lists in the next access operation. In the limiting case ($\rho = \infty$), both methods are identical to the co-occurrence trees approach.

III. MATERIALS AND METHODS

In this section we report an application of the proposed methods in ultrasonic image analysis. First, we acquired 10 ultrasonic images of 512×512 size from the human kidneys using various ultrasound scanners in order to show the robustness of the approach (see Fig. 2). The grey level resolution was 8 bits per pixel, giving a maximum dynamic range of 256 grey levels. We stored the images on the hard disk of a Pentium Dell P100t personal computer. We then clipped them down to 450×320 size, keeping only the useful information for further processing. After this step, we normalised them using the equal-probability quantizing algorithm in ([10]). This step was necessary to remove variations in illumination and also keep only the second order statistics of the images, i.e. the useful information in texture analysis. We selected 10 overlapping regions from each of the 10 images, getting a total of 100 textured regions for further analysis. The size of the selected regions was 30×30 or else 900 pixels. According to ([9]), the number of pixels in the region of interest must be at least 800 pixels to obtain reliable statistics. We applied all

AN EVALUATION OF A NUMBER OF TECHNIQUES FOR DECREASING THE COMPUTATIONAL COMPLEXITY OF TEXTURE FEATURE EXTRACTION THROUGH AN APPLICATION TO ULTRASONIC IMAGE ANALYSIS

Andrew E. Svolos and Andrew-Todd Pokropek, Member, IEEE
Image Processing Laboratory, Department of Medical Physics and Bioengineering, UCL,
London WC1E 6JA, UK. email: rmapasv@medphys.ucl.ac.uk

Abstract— **Texture feature extraction has been proved to be a fundamental process in medical image analysis. Therefore, the reduction of its computational time and storage requirements should be an aim of continuous research.**

This paper investigates a number of techniques in the direction of the above goal. They are all based on the space efficient co-occurence trees in the spatial grey level dependence method (SGLDM). The techniques are applied to a number of ultrasonic images, giving lower bound results on their time performance. A compàrison with the co-occurence matrix approach is performed. Finally, their usefulness in a real clinical application is discussed.

Index Terms— **Co-occurence trees, Co-occurence matrix, BB-tree, ultrasonic image analysis.**

I. INTRODUCTION

Texture is a fundamental feature for image analysis. Its usefulness has been proved for many types of images, ranging from multispectral remote sensed aerial and satellite data to biomedical images. In medical image analysis, texture has been successfully used for tissue characterisation, i.e., the determination of tissue type (normal or pathological) and further classification of tissue pathology ([1]). Texture analysis has been shown to increase the level of diagnostic information extracted from images of most medical imaging modalities and to quantitate differences in appearances inaccessible to human observers. Its great potential for clinical diagnosis has been presented in numerous papers in digital x-ray ([[2], [3]), x-ray CT ([4], [5]), MR ([6]) and ultrasonic ([7]-[9]) image analysis.

Two basic methods for texture description exist: syntactic and statistical. Syntactic methods describe texture by means of primitive descriptions and primitive placement rules. Statistical methods employ features extracted from the image that measure coarseness, contrast, directionality and other textural characteristics. Syntactic methods have seen very limited use in medical applications and also are computationally demanding ([6]). Among the statistical methods, one of the most heavily employed is the spatial grey level dependence method (SGLDM) ([10], [11]). SGLDM is based on the estimation of the second order joint conditional probability density functions $f(i,j|d,\theta)$ of an image. $f(i,j|d,\theta)$ give the probability of going from grey level i to grey level j, given that the intersample spacing is d pixels and the direction is θ degrees. A number of textural features are usually computed from the estimated probabilities $p(i,j|d,\theta)$ and used for texture description. Experiments on human texture perception indicated that second order probabilities play an important role in human texture discrimination ([12]). In medical image analysis, SGLDM has been employed in all aforementioned imaging modalities, i.e. in digital x-rays ([2], [3]), x-ray CT ([4], [5]), MRI ([6]) and ultrasound ([7]-[9]), fact that proves its great value.

Nevertheless, almost all of the above papers have referred to the disadvantage of using the co-occurrence matrix approach ([10]) in SGLDM, which is its inefficiency in both memory space and computational time. This is due to its size dependency on the gray level range of the entire image. A number of attempts have been made in the past to eliminate this weakness ([13], [14]). Many of the proposed solutions suffer from information loss ([13]) while others ([14]) employ a limited set of textural features to achieve a reasonable computational time, reducing though texture discrimination power. In a recent paper ([15]), a dynamic approach to organising the textural information extracted by the SGLDM was proposed, which depends only on the number of grey levels in the local image region being analysed. The authors reported a substantial reduction in both time and space. Their basic data structure was a modified version of the Binary Balanced Tree (**BB-tree**), which is well known for its efficiency in solving the *dictionary problem*. BB-tree, though, stores its elements as if they were uniformly distributed. Therefore, the access operation takes $O(\log n)$ time (n is the cardinality of the set of elements stored in the tree) for all elements, no matter how often each one of them is accessed. If the elements with larger access frequencies were nearer the root of the tree the expected cost of an access operation (which is the dominant operation in building a BB-tree) would be much better.

We can show this for a simple list structure. Suppose that $S = \{n_1, n_2, \ldots, n_k\}$ and $P = \{p_1, p_2, \ldots, p_k\}$ are the sets of the elements stored in a list and their normalised access

REFERENCES

[1] A. Lerski, "Texture analysis of medical images," in *Proc. of the Int. Symp. on Physics of Medical Imaging and Advances in Computer Applications*, pp. 64-68, Feb. 1993.

[2] R. P. Kruger, W. B. Thompson, and A. F. Turner, "Computer diagnosis of pneumoconiosis," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-4, pp. 40-49, 1974.

[3] M. F. McNitt-Gray, H. K. Huang, and J. W. Sayre, "Feature selection in the pattern classification problem of digital chest radiograph segmentation," *IEEE Trans. on Med. Imag.*, vol. MI-14, pp. 537-547, Sept. 1995.

[4] P. T. Cahill, R. J. R. Knowles, B. Kneeland, and B. C. P. Lee, "Segmentation of white/gray matter by a texture based clustering algorithm," in *Proc. of the IEEE Comp. Soc. Conf. on Patt. Recogn. and Im. Proc.*, pp. 127-130, Aug. 1981.

[5] A. H. Mir, M. Hanmandlu, and S. N. Tandon, "Texture analysis of CT images," *IEEE Eng. in Med. and Biol. Mag.*, vol. 14, pp. 781-786, Nov./Dec. 1995.

[6] R. A. Lerski, K. Straughan, L. R. Schad, et al., "MR image texture analysis-an approach to tissue characterization," *Magn. Res. Imag.*, vol. 11, pp. 873-887, 1993.

[7] U. Raeth, D. Schlaps, B. Limberg, et al., "Diagnostic accuracy of computerized B-scan texture analysis and conventional ultrasonography in diffuse parenchymal and malignant liver disease," *J. Clin. Ultrasound*, vol. 13, pp. 87-99, Feb. 1985.

[8] Y. N. Sun, M. H. Horng, X. Z. Lin, and J. Y. Wang, "Ultrasonic image analysis for liver diagnosis: a noninvasive alternative to determine liver disease," *IEEE Eng. in Med. and Biol. Mag.*, vol. 15, pp. 93-101, 1996.

[9] Y. M. Kadah, A. A. Farag, J. M. Zurada, et al., "Classification algorithms for quantitative tissue characterization of diffuse liver disease from ultrasound images," *IEEE Trans. on Med. Imag.*, vol. MI-15, pp. 466-478, Aug. 1996.

[10] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610-621, Nov. 1973.

[11] R. W. Conners, and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-2, pp. 204-222, May 1980.

[12] B. Julesz, E. N. Gilbert, L. A. Shepp, et al., "Inability of humans to discriminate between visual textures that agree in second-order statistics-Revisited," *Perception*, vol. 2, pp. 391-405, 1973.

[13] J. Desachy, *Texture Features in Remote Sensing Imagery*. D. Reidel Publishing, 1981.

[14] H. D. Chen, L. Wang, and J. Guibert, "Texture feature extraction," *Patt. Recogn. Lett.*, vol. 6, pp. 269-273, 1987.

[15] A. E. Svolos, C. A. Hutton, and A. T. Pokropek, "Co-occurence trees: a dynamic approach for texture feature extraction," in *Proc. of the IEEE EMBS 18th Int. Conf. on Eng. in Med. and Biol.*, 1996.

[16] J. R. Bitner, "Heuristics that dynamically organize data structures," *SIAM J. of Comput.*, vol. 8, pp. 82-110, Feb. 1979.

[17] K. Mehlhorn and A. Tsakalidis, "Data Structures," in *Handbook of Theoretical Computer Science*. North Holland Edition, 1990.

[18] P. C. Chen and T. Pavlidis, "Segmentation by texture using correlation," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. PAMI-5, pp. 64-69, Jan. 1983.

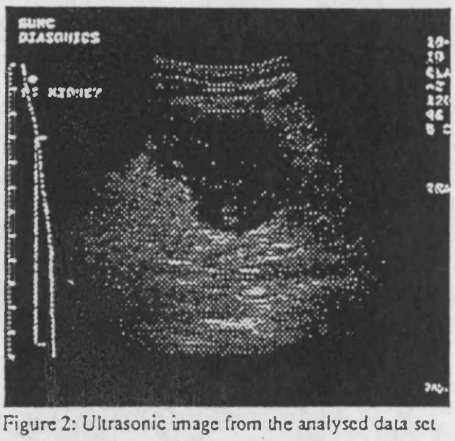
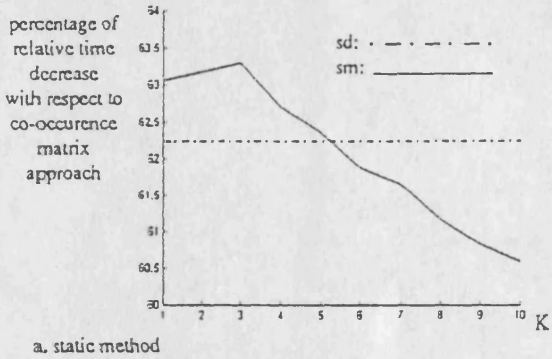
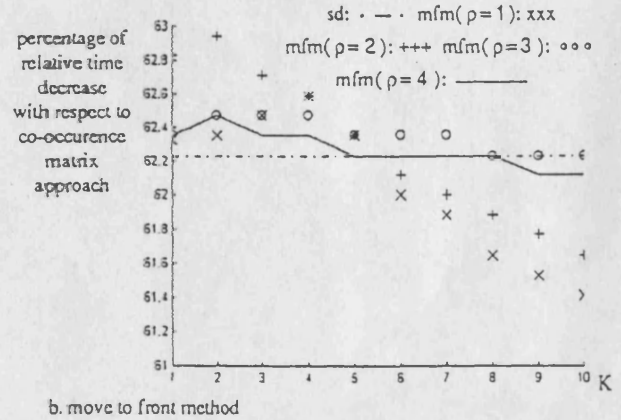


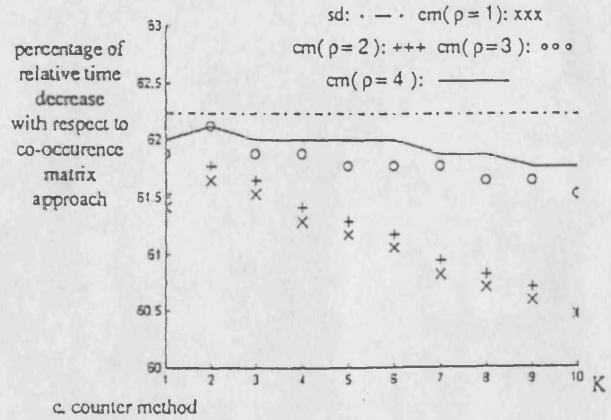
Figure 2: Ultrasonic image from the analysed data set



a. static method



b. move to front method



c. counter method

Figure 3: Results from the analysed images

the proposed methods to each one of the 100 selected regions for an intersample spacing distance $d=1$ pixel, direction $\vartheta=0^{\circ}$ and parameter $K \in \{1,2,\ldots,10\}$. The semi-dynamic version of the co-occurrence trees was used for all three methods. In the static method, for each i grey levels j with the K largest co-occurrence probabilities estimated over all analysed regions were found and put in p_list(i), in descending order with respect to their probabilities. In the move to front and counter method, parameter $\rho \in \{1,2,3,4\}$. Since the difference among the time results of the analysed regions for each method and parameter setting was negligible in the millisecond time scale used, we employed the time average as the appropriate robust statistic for our comparison purposes. Finally, for each method and parameter setting we calculated the relative decrease in computational time with respect to the co-occurrence matrix approach.

IV. RESULTS

In Fig. 3, the results from the analysis discussed in the previous section are shown. In all three plots, the percentage relative computational time decrease of the plain semi-dynamic method with respect to the co-occurrence matrix approach is depicted as well. Fig. 3a shows the time performance of the static method, while Fig. 3b and Fig. 3c show the performance of the move to front and counter rule respectively, for various values of parameter ρ.

V. DISCUSSION

As expected, all three methods present a substantial improvement compared with the co-occurrence matrix approach since they are based on the co-occurrence trees. The percentage reduction of the semi-dynamic version is also shown to be very close to the percentage reductions of the proposed methods. At this point, we need to stress the fact that the present study involves the worst case time behaviour of these methods. This is why we chose to analyse ultrasound images which, as it is known, have a small dynamic range (256 grey levels). For the same reason we selected regions in each image of the smallest possible size (30×30) containing enough information for analysis. As the region size increases, the improvement in time complexity is expected to become larger, especially for images with larger dynamic ranges, e.g. 16 bit MR and CT images. Therefore, one should regard the presented results as lower bounds on the time efficiency of the proposed methods in medical image analysis.

The static method scored the lowest computational times among the examined methods, which is not a surprise if we consider that the most probable grey levels were determined *a-priori*. This is also an indication that the texture model proposed in ([18]) may be appropriate for ultrasonic image analysis. The move to front method is ranked second, which is in agreement with the theoretical results in the literature ([16]). The counter method gives the worst results. This is because the time needed to sort the grey levels in p_list(i) dominates the total computational time. For images with larger dynamic ranges and image regions with larger sizes the time results of this method should become much better. One very important thing to mention here is that this method needs no extra space for the access frequencies of the grey levels since they are equal to the estimated co-occurrence probabilities. All three methods present a maximum of performance for some value of the parameter K. It appears that as the size of p_list(i) increases computational time first decreases, reaches a minimum and then increases to infinity. The results also indicate that the optimal size of the lists should be small, usually 2 or 3. Above that point the search time within the lists starts dominating the total time. The computational time also increases as parameter ρ becomes larger. The results clearly show that the performance of the move to front and counter method tends to that of the plain semi-dynamic approach. This is reasonable since as ρ increases, the list updates are less often. From the results it is also obvious that there is a value for the parameter ρ ($\rho=2$) where the move to front method reaches a global maximum of performance. For $\rho=1$, list updates are too often and they dominate the total computational time. For $\rho>2$ the method tends to become identical to the plain semi-dynamic approach. In the counter method, as ρ increases the time decreases reaching that of the semi-dynamic approach in the limiting case.

Finally, we need to mention that the best improvement of the proposed methods compared with the plain semi-dynamic method is 2.80% for the images analysed. This corresponds to a reduction of about 150 msec in the computational time for the analysis of one image region. However, as we said at the beginning of this discussion, this is only a lower bound for the proposed methods. Also, in a real application where hundreds or thousands of image regions need to be analysed, the decrease of total computational time will be substantial and the utilisation of a "time sharing" image processing system will be much more efficient.

VI. CONCLUSIONS

The proposed methods for texture feature extraction yield a significant reduction in computational time in comparison with the co-occurrence matrix approach, without sacrificing the efficient space complexity of the co-occurence trees. Results from the analysis of ultrasonic images show an improvement compared with the plain semi-dynamic co-occurrence trees method, which will be significant in a real clinical application. In addition, their performance is expected to increase greatly when applied to MR and CT images.

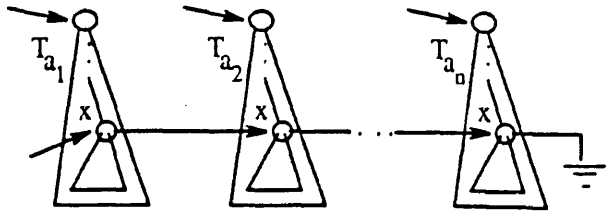


Figure 1: Co-occurence trees structure.

In the second section, the proposed scheme is described. It uses, as its main data structure, a modified version of the *Binary Balanced Tree* (*BB-tree*) ([10]) and particularly its node-oriented version. A BB-tree is a full binary tree and it is characterized as the best balanced tree for the *dictionary problem* in main memory due to its following properties: a) *access*, *insert* and *delete* operations cost $O(\log n)$ [2], b) the *amortized* rebalancing cost is only $O(1)$ ([11]) and c) only one bit is necessary for the balance information in each node of the tree; actually, in our case this bit can be embedded in the information content of each node, so practically no space is needed. In the third section, an updating algorithm for the proposed scheme is presented which minimises the updating cost using an auxilliary list data structure. In the fourth section, our approach and the co-occurence matrix are compared and the superiority of the proposed scheme is proved.

II. THE PROPOSED SCHEME

The approach presented in this paper is composed of a set of BB-trees (BB-tree forest) which are modified to suit the requirements of texture feature extraction. If $L_{\mathcal{R}}$ denotes the number of gray levels in the local neighbourhood $\mathcal{R}$ of a given pixel, the above set contains *at most* $L_{\mathcal{R}}$ BB-trees, one tree for each gray level. If T_a is the BB-tree for the a gray level then for each pair (a, x) of gray levels that satisfies the (ρ, θ) spatial dependence criterion ([1]) in $\mathcal{R}$, there is a node in T_a (x-node) that stores the relevant information. This information consists of the x gray level, the relative frequency P_{ax} (number of occurences of (a, x) in $\mathcal{R}$) and a pointer to a x node in some other tree T_b. These pointers link all x-nodes forming a list (x-list) for each x (see Fig. 1). This list is used for the efficient implementation of the $P_y(j) = \sum_i P(i, j)$ primitive operation. Also, each node contains two pointers (one to its left child and the other to its right child) used for building the relevant BB-tree. Note that the proposed scheme stores only the significant information in $\mathcal{R}$ (zero entries are not stored in contrast to the co-occurence matrix). Thus, not only are the memory requirements reduced but also the speed of the texture feature extraction operations is increased.

[2] n is the cardinality of the set of elements stored in the tree

The heads of the x-lists, as well as the pointers to the roots of the BB-trees, are stored in two separate data structures. Two versions of the co-occurence trees are proposed: a) the *semi-dynamic* version, where the above structures are arrays, allows faster access ($O(1)$ time) but inherits all the disadvantages of the static data structures mentioned earlier, b) the *full-dynamic* version uses two additional BB-trees but needs logarithmic access time.

III. UPDATE ALGORITHM

Let r be the number of rows and c the number of columns of the window that defines the local region $\mathcal{R}$ about a given pixel. The proposed updating structure is composed of *at most* $2(c-1)$ lists, $c-1$ lists (l_{eq}-lists) for the i-nodes of the T_j trees, where $P_{ij} = 1$ and $c-1$ additional lists (l_{gr}-lists) for the nodes with $P_{ij} > 1$. Each node in a list points to a node in some tree. Let $((x_1, y_1), (x_2, y_2))$ be a pair of pixels in $\mathcal{R}$ that satisfies the (ρ, θ) spatial relationship, and (L_1, L_2) the corresponding gray level pair. If $P_{L_1L_2} = 1$, there is a node in l_{eq}-list that corresponds to y_1 column ($l^{y_1}_{eq}$-list) which points to the L_2-node in T_{L_1} tree. If $P_{L_1L_2} > 1$ then the above node is in $l^{y_1}_{gr}$-list.

Suppose that the window moves one pixel to the right[3]. Let y_{old} denote the column which is no longer within the window and y_{new} the new rightmost column within it. Then, the pseudo-code for the update algorithm is shown in Fig. 2.

IV. RESULTS-COMPARISONS

One of the most attractive features of dynamic schemes is that, in contrast to static structures, they exhibit an average case behaviour in addition to the worst case. In this paper only results from the worst case analysis of the proposed scheme are being presented.

Let L denote the number of gray levels in the entire image. The space complexity of the co-occurence matrix is then $O(L^2)$ whereas for the proposed scheme is $O(L^2_{\mathcal{R}})$. So, there is a reduction of $O((\frac{L}{L_{\mathcal{R}}})^2)$. Since in most cases $L_{\mathcal{R}}$ is much smaller than L, the reduction is significant.

The computational time complexity of the $P_x(i)$ and $P_y(j)$ primitive operations is $O(L_{\mathcal{R}})$ for the proposed scheme, and $O(L)$ for the co-occurence matrix. So, there is a speed-up of $O(\frac{L}{L_{\mathcal{R}}})$ which is significant, too. Similarly, the complexity of the $P_{x+y}(k)$ and $P_{x-y}(k)$ primitive operations is $O(L_{\mathcal{R}} \log L_{\mathcal{R}})$ for the proposed scheme, and $O(L)$ for the co-occurence matrix. In most practical cases, $L_{\mathcal{R}} \log L_{\mathcal{R}} < L$, so an increase of $O(\frac{L}{L_{\mathcal{R}} \log L_{\mathcal{R}}})$ in speed is being achieved.

[3] one similar structure may be used for the updating in the vertical direction

CO-OCCURENCE TREES: A DYNAMIC SOLUTION FOR TEXTURE FEATURE EXTRACTION

Andrew E. Svolos, Chloe A. Hutton, and Andrew-Todd Pokropek, Member, IEEE

Department of Medical Physics and Bioengineering, University College London, London WC1E 6JA, UK. email: rmapasv@medphys.ucl.ac.uk

Abstract— **Texture feature extraction has shown valuable results in medical image analysis, segmentation and classification. Thus, efficient methods for storing the textural information and computing the textural features of an image must be devised.**

Co-occurence matrices are powerful for discriminating different textures but are not efficient in terms of memory requirements. Due to their dependency on the number of gray levels in the entire image a large amount of space is being wasted while time complexity for the feature extraction operations is raised.

This paper presents a novel, dynamic approach to organizing the textural information which provides efficiency in both storage requirements and computational time, being dependent only on the number of gray levels in the examined local region. Furthermore, an updating algorithm which achieves a near half reduction in time is presented and evaluated. Finally, results are presented and discussed which clearly indicate the superiority of the proposed approach.

Index Terms— **Texture feature extraction, Co-occurence matrix, BB-tree.**

I. INTRODUCTION

Texture is one of the fundamental properties for image analysis, segmentation and classification. Two basic methods for texture description exist: syntactic and statistical. Syntactic methods describe texture by means of primitive descriptions and primitive placement rules whereas statistical methods employ features extracted from the image which measure coarseness, contrast, directionality and other textural characteristics.

Co-occurence matrices ([1]) are very powerful for statistical texture description. According to [2], all known visually distinct texture pairs can be discriminated by using the above matrices. In medical imaging, their performance has been shown to be one of the best, especially in Ultrasonic, MR and CT image analysis ([3], [4]). Co-occurence matrices are based on second order statistics (the spatial relationships of pairs of gray levels). Texture is described by extracting a number of textural features from them. The texture feature extraction method introduced in [1] has been proven to be one of the best in overall performance ([5]). However, they exhibit some significant disadvantages because of their size dependency on the number of gray levels in the entire image. Since texture is usually measured in a local neighbourhood about a given pixel, a large number of entries are equal to zero, contributing nothing to the texture description of a local region. Also, the computational time of the texture feature extraction operations includes the time for processing those entries. The above problems become more serious when the examined images are composed of a large number of gray levels. Additionally, their static nature makes them inefficient, demanding a continuous recompilation of an application should the number of gray levels vary. These problems have been addressed in the past ([4], [6]-[9]). Many of the proposed solutions suffer from information loss ([7]) while others ([8]) employ a limited set of texture features to achieve a reasonable computational time, reducing though texture discrimination power. Still others ([9]) completely abandon the co-occurence matrices and use other methods for texture description, though not equally powerful.

In this paper, a novel scheme for organizing the texture information in an image is proposed. This scheme, which is called the *co-occurence trees*, solves in an efficient way the memory problem mentioned above, its size being dependent on the number of gray levels in the local neighbourhood of a given pixel. It contains the same basic information as the co-occurence matrix, but without the redundant entries. Also, the computational time for the texture feature extraction operations now depends only on the significant (non-zero) entries. The efficiency of the proposed algorithm has been proved for the following primitive operations[1]: $P_x(i) = \sum_{j=1}^{L_w} P_{ij}$, $P_y(j) = \sum_{i=1}^{L_w} P_{ij}$, $P_{x+y}(k) = \sum_{\substack{i=1 \\ i+j=k}}^{L_w} \sum_{j=1}^{L_w} P_{ij}$, $k = 2, 3, \ldots, 2 \cdot L_w$, and $P_{x-y}(k) = \sum_{\substack{i=1 \\ |i-j|=k}}^{L_w} \sum_{j=1}^{L_w} P_{ij}$, $k = 0, 1, \ldots, L_w - 1$. They represent the core of the calculation of the 28 textural features proposed in [1].

[1] the detailed proofs are beyond the scope of this paper

```
/* remove the nodes from the l_gr^y_old-list and update the corresponding relative frequencies(n is the node in the tree structure and n.p the corresponding relative frequency) */
  for each x ∈ l_gr^y_old do
    n = pointed_to(x); n.p − −; free(x);
/* new (L1, L2) gray level pairs are inserted as follows */
  for each (L1, L2) do
    if not exists(T_L1) do
      T_L1 = create_tree(L1);
/* if l_eq^y_old is not empty, remove one node from it otherwise allocate a node from free memory; update the node to be an L2-node and insert it in T_L1 */
      if exists(l_eq^y_old) do
        n = insert_tree(update(remove_list(l_eq^y_old), L2), T_L1);
      else
        n = insert_tree(update(alloc(), L2), T_L1);
/* a new node that points to n is inserted in the l_eq^y_new-list */
      insert_list(n, l_eq^y_new);
    else if n in T_L1 and n.gray_level = L2 do
      n.p + +; insert_list(n, l_gr^y_new);
    else
/* same as in the case where T_L1 doesn't exist */
      if exists(l_eq^y_old) do
        n = insert_tree(update(remove_list(l_eq^y_old), L2), T_L1);
      else
        n = insert_tree(update(alloc(), L2), T_L1);
      insert_list(n, l_eq^y_new);
/* free all remaining nodes in l_eq^y_old-list and the nodes that are pointed to by them in the tree structure */
  if exists(l_eq^y_old) do
    for each x ∈ l_eq^y_old do
      n = pointed_to(x); free(n); free(x);
```

Figure 2: Pseudo-code for the update algorithm.

The updating algorithm proposed in this paper needs $r(\log L_{\mathcal{R}} + 1) \cdot O(1)$ time. Without using it the update costs $2r \log L_{\mathcal{R}} \cdot O(1)$. Thus, a decrease by a factor of $0.5 + \frac{1}{2 \log L_{\mathcal{R}}}$ is achieved. There is, though, an increase in space complexity by $O(r(c-1))$ but it is small compared to the near half reduction in computational time. Fig. 3 shows the percentage of reduction in memory requirements for various L in the worst case[4], using our scheme and particularly the *full-dynamic* version instead of the co-occurence matrix. This version gives slightly better results than the *semi-dynamic* version, especially for $L \leq 64$. Note that a significant reduction is achieved for $L_{\mathcal{R}}$ slightly less than $\frac{L}{2}$, when $L \leq 128$ and for $L_{\mathcal{R}}$ slightly more than $\frac{L}{3}$, when $256 \leq L \leq 1024$. Since the above cases are more than representative for practical $L_{\mathcal{R}}$ values in the worst case, the superiority of the proposed scheme in saving memory is obvious.

V. CONCLUSIONS

The proposed dynamic scheme for texture feature extraction is much superior to the co-occurence matrix in both space and time complexity, greatly reducing the storage requirements and the time needed for the primitive texture feature extraction operations.

[4] when $\mathcal{R}$ contains $L_{\mathcal{R}}^2$ distinct gray level pairs

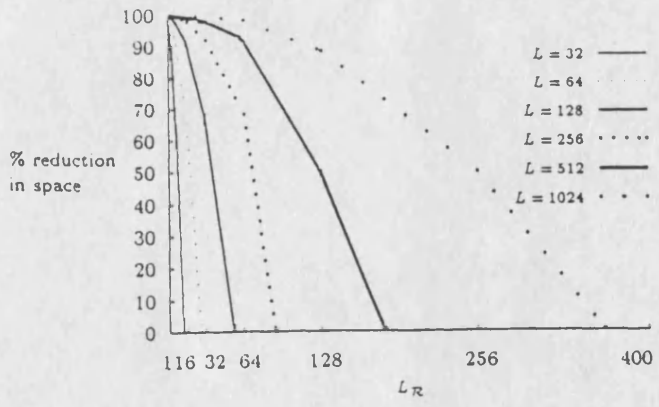


Figure 3: Percentage of reduction in space using the proposed scheme (*full-dynamic* version) instead of the co-occurence matrix.

A near half reduction in update computational time is being achieved by the proposed update algorithm.

REFERENCES

[1] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 610–621, Nov. 1973.

[2] R. W. Conners, M. M. Trivedi, and C. A. Harlow, "Segmentation of a high resolution urban scene using texture operators," *CVGIP*, vol. 25, pp. 273–310, Mar. 1984.

[3] A. H. Mir, M. Hanmandlu, and S. N. Tandon, "Texture Analysis of CT Images," *IEEE Engineering in Medicine and Biology*, vol. 14, pp. 781–786, Nov./Dec. 1995.

[4] Chung-Ming Wu, Yung-Chang Chen, and Kai-Sheng Hsieh, "Texture features for classification of ultrasonic liver images," *IEEE Trans. on Med. Imag.*, vol. 11, pp. 141–152, June 1992.

[5] J. M. H. du Buf, M. Kardan, and M. Spann, "Texture feature performance for image segmentation," *Pattern Recognition*, vol. 23, pp. 291–309, 1990.

[6] F. Argenti, L. Alparone, and G. Benelli, "Fast algorithms for texture analysis using co-occurence matrices," *IEE Proceedings*, vol. 137, Pt. F, pp. 443–448, Dec. 1990.

[7] J. Desachy, *Texture Features in Remote Sensing Imagery*. D. Reidel Publishing, 1981.

[8] HE Dong-Chen, Li Wang, and Jean Guibert, "Texture feature extraction," *Pat. Rec. Let.*, vol. 6, pp. 269–273, 1987.

[9] D. Patel and T. J. Stonham, "Texture image classification and segmentation using RANK-order clustering," in *Proceedings of 11th ICPR*, vol. 3, 1992.

[10] R. E. Tarjan, "Updating a Balanced Search Tree in O(1) Rotations," *Inf. Proc. Let.*, vol. 16, pp. 253–257, June 1983.

[11] K. Mehlhorn and A. Tsakalidis, "Data Structures," in *Handbook of Theoretical Computer Science*. North Holland Edition, 1990.