Stellingen

Behorende bij het proefschrift:

Edge-Based Image Representation and Coding

Peter van Beek

4 December 1995

I

In de belangrijkste literatuur over randdetectie in digitale beelden wordt veel aandacht besteedt aan de keuze van het gebruikte filter. Ten onrechte wordt er weinig tot geen aandacht geschonken aan het gebruikte mechanisme om randen te detecteren in de filterresponsie, terwijl dit laatste evenzeer van belang is.

II

De in dit proefschrift geïntroduceerde multischaal optimalisatiemethode bezit een hogere graad van eenvoud en flexibiliteit dan multiresolutie methoden uit de literatuur, omdat een hiërarchische representatie van de *data* niet vereist is.

(Dit proefschrift, Hoofdstuk 3)

III

Interframe coderen van curven en contouren in digitale beelden levert geen winst op ten opzichte van intraframe coderen, tenzij distortie van de curven wordt toegestaan.

(Dit proefschrift, Hoofdstuk 5)

IV

Onderzoek binnen de beeldcodering bij lage bit rates kan geen vooruitgang boeken, zolang er geen beeldkwaliteitscriteria omtrent vervormingen van geometrische structuren in beelden beschikbaar zijn.

V

Bij discussies over kunstmatige intelligentie wordt vaak het "Chinese kamer argument" van J. Searle aangehaald, waarmee zou worden aangetoond dat computers per definitie geen werkelijk begrip kunnen hebben van wat ze doen (zelfs niet als ze intelligent gedrag lijken te vertonen). Dit Chinese kamer argument is echter inconsistent, misleidend en onjuist.

(Zie J. R. Searle, "Is the brain's mind a computer program?" Scientific American, Januari 1990, vol. 262, no. 1)

VI

Wetenschappers zijn uitermate gefascineerd door het onbekende. Toch zit in hun werkwijze de aanname opgesloten dat het onbekende niet wezenlijk verschilt van het bekende.

VII

Vanouds vertrouwt de mens meer op de dingen die hij ziet dan op de dingen waarvan hij slechts hoort. Bij toenemend gebruik van visuele communicatie systemen en virtual reality systemen zal dit vertrouwen langzaam maar zeker afnemen.

VIII

Vroeger was het vaak zo dat nationalistisch(e) en etnisch(e) onverdraagzaamheid en geweld wortelden in misplaatste ideeën over *superioriteit* van de eigen groep. Tegenwoordig wordt vaker gewezen op vermeende *achterstanden* van de eigen groep op maatschappelijk en economisch terrein om dergelijk gedrag uit te lokken.

IX

De massale toestroom van het grote publiek op het Internet gedurende het afgelopen jaar vertoont veel gelijkenissen met de trek van migranten naar het ("wilde") westen van het Amerikaanse continent, ongeveer honderdvijftig jaar geleden.

X

Veel mensen geloven niet in "Windows everywhere," maar zijn er wel bang van.

("I don't believe in ghosts, but I am afraid of them." - Byron)

Edge-Based Image Representation and Coding

P. J. L. van Beek

Edge-Based Image Representation and Coding

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. ir. K. F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie,
door het College van Dekanen aangewezen,
op maandag 4 december 1995 te 10:30 uur
door

Petrus Johannes Leonardus VAN BEEK

elektrotechnisch ingenieur,
geboren te Amsterdam.

Dit proefschrift is goedgekeurd door de promotor: Prof. dr. ir. E. Backer.

Toegevoegd promotor: Dr. ir. J. C. A. van der Lubbe.

Samenstelling promotiecommissie:

Rector Magnificus (voorzitter)
Prof. dr. ir. E. Backer, Technische Universiteit Delft (promotor)
Dr. ir. J. C. A. van der Lubbe, Technische Universiteit Delft (toegevoegd promotor)
Prof. dr. ir. J. Biemond, Technische Universiteit Delft
Prof. dr. B. Sankur, Bogaziçi Universiteit, Istanbul, Turkije
Dr. P. W. Verbeek, Technische Universiteit Delft
Prof. dr. R. Prasad, Technische Universiteit Delft
Prof. dr. W. M. Oppedijk van Veen, Technische Universiteit Delft

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Beek, Petrus Johannes Leonardus van

Edge-based image representation and coding /
Petrus Johannes Leonardus van Beek. - [S.l. : s.n.]. - Ill.
Thesis Technische Universiteit Delft. - With ref. - With summary in Dutch.
ISBN 90-9008901-2
Subject headings: image coding / edge analysis / surface interpolation.

Table of Contents

Summary

With the steadily increasing use of visual information, the ability to efficiently store and transmit large amounts of image data becomes more important. A wide variety of digital image and video coding methods have been proposed and some methods have already been subject to standardization, thus facilitating rapid dissemination of compression capabilities in practice. Spurred by the expected further growth in the storage, manipulation and exchange of multi-media information, research trends in image coding are directed at new methods.

This thesis discusses the efficient representation and coding of digital images and video by parametrically modeling relevant image intensity surface variations. Specifically, we describe intensity edge analysis, intensity surface reconstruction, edge curve matching and the coding of edge model-parameters. Our main application is in very low bit rate image coding.

An introduction to the image coding problem and to the edge-based approach is provided in Chapter 1. The main question in image coding is: Can we obtain high compression ratios and still reproduce the original image with sufficient intelligibility and quality? Many natural images can be viewed as consisting mainly of smoothly varying intensity surfaces, interleaved occasionally by sharp intensity changes. These sharp intensity changes correspond to features in the imaged scene, such as object boundaries, shadow boundaries, or surface reflectance changes. Many image coding methods do not take these characteristic properties of images into account explicitly. In this thesis, image representation and image coding are based on the analysis, modeling and manipulation of zero-order intensity changes called *edges*, therefore, the approach is called *edge-based.* A clear break with earlier coding methods is made by explicitly taking the geometry of the edge structure in images into account.

Intensity edges are important in the process of understanding an image or a scene. This hypothesis is supported by an enormous amount of studies on edges in the computer vision field and by studies on vision in animals and humans. From intermediate-level image representations which carry information about intensity edges alone, intelligible images can be

reconstructed. These observations suggest that useful results may be obtained in high-compression image coding when an edge-based approach is employed.

In Chapter 2, an edge analysis method is proposed and an edge-based image representation is defined. An idealized parametric model is used to describe the intensity variation across an edge, fitting the model to the intensity signal at all points where an edge is present. We restrict ourselves to the use of a *Gaussian smoothed intensity edge* model, parameterized by *contrast, width* and *center intensity value*. It is shown that derivative-of-Gaussian filters can be used to detect edges and recover their model-parameters at the same time. Error analyses and experimental results are included, for the purpose of studying the effects of some typical model deviations. The *scale* of the analyzing Gaussian filters influences the accuracy and precision of the parameter estimates. In general, a good scale for these filters is given by the value of the edge width. However, in well-acquisitioned imagery, the filter scale can be kept small.

Further, the edge-based image representation itself is described. It consists of *edge primitives*, each describing: (a) the loci of edge points lying along a connected *edge curve* in the image plane (*geometric* data) and (b) the variations of the image intensity function within a small band along such a curve, in terms of edge contrast, width and center intensity value parameters (*photometric* data). Finally, we describe how the edge points along such a curve can be reconstructed from the edge primitives.

Although the representation by edge primitives does not allow the exact recovery of the original input image, an approximation can be computed which often is numerically and perceptually similar to the original. This is discussed in Chapter 3. The edge primitives describe only the image samples along edge curves. The full intensity surface must be *reconstructed* or *interpolated* from this sparse data. Within the framework of *regularization theory*, this interpolation problem can be cast into the minimization of a *smoothness functional*. The optimal solution can be found using standard iterative algorithms, e.g. a *relaxation* algorithm. The constraints given by the edge primitives in our representation are iteratively *propagated* across the entire image domain by this algorithm. However, the standard optimization algorithm is too slow.

A new optimization algorithm is proposed, which allows more global and thus faster propagation of information across the image grid. The new algorithm can be derived by introducing *multi-scale smoothness functionals* in the regularization. A convergence analysis and experimental results show that the *multi-scale* relaxation algorithm converges much faster than the standard algorithm.

Chapter 4 discusses further compression of the data stored in the edge primitives, and the application to still image coding at low bit rates. The edge locations (geometric data) are coded in a lossless manner by a variant of chain coding. A sequence of chain codes is modeled by a second-order Markov chain and compressed using Huffman variable length encoding. The values in a sequence of model-parameters of the same type (e.g. the contrast) in edge points along an edge curve (photometric data) are treated as the samples of a one-dimensional *parameter signal*. A simple, lossy coder is designed on the basis of the statistics of the

parameter signals. This coder consists of a downsampling stage and a DPCM stage. Optimal Uniform Threshold Quantizers are used to quantize the prediction error. Huffman variable length encoding is used to further compress this data.

Experiments on a few images show that the final quality in terms of Peak Signal-to-Noise Ratio lies between 23 and 33 dB at rates between 0.05 and 0.41 bit per pixel (compression ratios between 20 and 170). The primary visual artifacts in the decoded images are a loss of texture, "halo" effects and jagged edges. The shape and location of each feature is preserved well. Baseline JPEG outperforms the edge-based coder at bit rates higher than 0.20 bit per pixel, while the edge-based coder outperforms JPEG at rates below 0.20 bit per pixel.

Spatially adaptive coding can be realized by the use of *a priori knowledge* during the edge detection process. *Knowledge-based image coding* of head-and-shoulders images can lead to better visual quality of important parts of these images.

In Chapter 5, the image compression method is extended to the case of image *sequences*. To exploit the temporal correlation in image sequences, one must establish the correspondence between edge primitives from different images in the sequence. To this end, a frame-to-frame matching algorithm is designed, based on a *similarity measure*. The matching algorithm seeks, for each edge primitive in the current frame, a primitive in a previous frame with maximum overall similarity. *Geometric* similarity is measured by computing the minimum mean-squared distance between edge curves. *Photometric* similarity is measured by cross-correlating the parameter signals. About 75% of the edge points in an image can be matched by this algorithm.

A sequence of chain codes of a matched curve can be coded *interframe*, using the sequence of chain codes of the matching curve as a prediction and transmitting only a sequence of (Huffman coded) prediction error chain codes. A displacement vector is transmitted as well, to code the location of the curve starting point. In our experiments, *no* coding gain can be obtained by using interframe edge location coding instead of intraframe coding, even if the prediction errors in the chain codes are coded *lossy*. Interframe coding of the parameter signals (photometric data) is not discussed.

For various head-and-shoulders image sequences, the final quality in terms of PSNR lies between 27 and 33 dB at rates between 0.09 and 0.26 bit per pixel (compression ratios per frame between 30 and 90). The visual quality of the decoded image sequences is rather disappointing in our experiments, due to severe temporal flickering between different brightness levels of regions all over the image. The difference in image quality between lossless and lossy coding of interframe prediction errors in the chain codes is quite small on average.

Chapter 6 contains a closing discussion of our overall approach and results. Concerning application in practice, one should note that the threshold used in edge detection was fixed manually in our experiments, but it can be of great influence on the output image quality and compression ratio in practice. Concerning real-time implementation, one should note that the main computational burden is created by the iterative nature of the intensity surface reconstruction algorithm. Our coding scheme can be compared to some other new approaches

in low bit rate image coding, such as segmentation-based or region-based coding and object-based coding. One may conclude that the lack of significant advances in terms of rate-distortion performance of all these methods compared to block-based methods is mainly due to the difficulty of the shape or location coding problem. Although practical application of the edge-based method in image coding still seems far away, we feel that there is room for further improvement of our representation and coding algorithms.

Chapter 1
Introduction

This thesis is on the compact representation of digital images by means of edge information. This image representation and compression method has been developed primarily for low bit rate image coding applications, such as the videophone. This introductory chapter discusses the background of the image coding problem, proceeds with issues in image representation and modeling, and then discusses our approach to compact image representation. Finally, an outline of the thesis is given.

1.1 Background

With the steadily increasing use of visual information, the ability to efficiently store and transmit large amounts of image data becomes more important. Therefore, *image coding* techniques are of concern to image processing applications such as broadcast television, remote sensing, facsimile, medical imaging, video telephony, video conferencing, etc. Image coding generally aims at reducing as much as possible the amount of data necessary to represent images, while keeping the *intelligibility* and *quality* of the visually reproduced image sufficiently high [77]. Here, intelligibility refers to recognizability or interpretability by human beings, while quality in the narrow sense refers to the degree of perfection in reproducing the original color, texture etc. The levels of image quality and intelligibility required depend on the application and vary widely. In such cases as medical images, nearly exact reproduction of the original image data is paramount. In other cases, such as when deaf people use the videophone to communicate sign language or lip movements, the intelligibility of detailed movements is more important and one might sacrifice a significant degree of overall quality. For other applications, such as retrieval operations in image databases, the possibility of a *variable* reconstruction quality could be advantageous. The *complexity*

requirements of a coding system play a significant role in the trade-off between compression, quality and intelligibility.

A wide variety of digital image (sequence) coding methods have been proposed, see for instance [58][60][86]. Recently, coding methods suitable for a range of applications have been subject to international standardization (e.g. [28][55][56][93]), facilitating rapid dissemination of efficient transmission or storage capabilities. However, new applications have already been foreseen (e.g. [5]), for which image coding methods formerly proposed are not really suitable. Examples are the following.

- Videophone services via PSTN or via wireless networks; electronic newspapers, remote monitoring. Some of these applications have to operate at very low bit rates (e.g. in the range of 8-64 kilobits per second), but still require acceptable image quality.
- Interactive multimedia databases. Applications such as these may require flexible handling of the image data, e.g., image retrieval by *content* rather than by identifiers [13][96]. Preferably, the representation in which the image is stored should allow for fast content analysis.

For these and other reasons, research trends in image coding are directed at new approaches in order to find even more efficient descriptions of the image data (see [40] for an overview).

1.2 Image representation and modeling

1.2.1 General

A simplified diagram of the typical environment for image coding systems is given in Figure 1.1. This thesis is mainly concerned with the *encoder* and *decoder* stages. Channel (de-)coding and (de-)modulation are assumed to be part of the channel. Sampling and A/D conversion are part of the image acquisition, D/A conversion is part of the display device. Image acquisition and display are not considered in this thesis. The channel is not considered either and is assumed errorless.

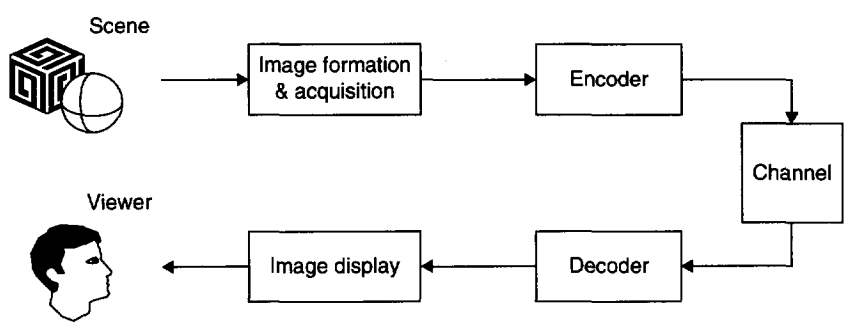


Figure 1.1 Image coding environment.

The question that is central to the coding process concerns the nature of the *representation* of the images and the *image model* employed. Classified by the degree of modeling of the image, the image formation process, or the original scene involved, the various coding methods proposed range from pure *waveform* coders to pure *model-based* coders. Other classes of methods, such as *transform* coders and *multi-resolution* coders, range somewhere between these two.

In the class of *waveform coding* methods, the image intensities are coded directly, often exploiting statistical relationships between individual pixels. These methods rely mainly on *stochastic* image models, such as random fields. Results from information theory played an important role in the development of these, historically early, methods. *Transform* coders, *subband coders* and *multi-resolution* coders treat groups of pixels collectively. These methods rely mainly on linear systems theory, Fourier transform theory, and sampling theory. A tendency can be noted towards modeling *three-dimensional* object properties, besides the more readily available *two-dimensional* image properties. In the class of *model-based* coders, an image is often described in terms of parametric models, related to the shape and motion of the objects which are supposedly present in the scene, and to scene illumination and camera projection. Only the model-*parameters* are stored or transmitted. For these methods, techniques from the computer vision field are highly useful, such as motion and depth estimation, segmentation and even pattern recognition. Computer graphics techniques can play a role as well, in resynthesizing an image from the model-parameters. In this thesis, we do not use three-dimensional object- or scene-models.

On a complexity scale, coding methods range again from waveform coders to model-based coders. Most waveform coders are computationally relatively simple. Model-based coding methods, however, are computationally much more demanding, because extra effort has to be put into both analyzing the image at the transmitter to estimate the model-parameters and synthesizing an image at the receiver from the coded model-parameters. Also, waveform coders tend to exploit *local* relationships in images, while model-based coders exploit more *global* relationships. Again, transform coders and multi-resolution coders range somewhere between these two extremes.

Waveform coders apply to many different classes of signals, because they attempt to reconstruct the original waveform exactly, without detailed knowledge specific to any particular class. Model-based coders, however, cannot be applied to all classes of signals, because they often rely on a priori knowledge about the scene content. It is through the use of this a priori knowledge that very high compression can be achieved. Furthermore, they do not attempt to reproduce the original waveform exactly, but try to retain the features of the scene being imaged that are essential to its intelligibility.

Because reconstructed images are eventually viewed by a human, their fidelity is of crucial importance. Preferably, the fidelity measure used should reflect the way the human visual system judges these images. Some coding techniques have been based on models developed in the context of perception research. For instance, the division of an image into different frequency bands in subband coding can be seen to correspond closely to models of the spatial sensitivity properties of the visual system. Some perceptual effects relevant in image

communication systems are discussed in [43]. Discussions of visual perception in the context of image processing can be found in [59] and [77]. For a more fundamental treatment, see [20] and [31]. Expressing subjective image quality as a well-defined numerical criterion still poses a problem, despite the many attempts that have been made. The complexity of the perception process and its variability among people still constitute an obstacle to the formulation of complete computational models. In this thesis, we do not consider visual perception models explicitly.

1.2.2 Approaches in high compression systems

As mentioned, many image coding methods are based on statistical models of the image signal. Very often the image is considered as a sample from a discrete stationary random field. Stationarity in the wide sense means homogeneity (in space and time) of the ensemble mean and covariance. In reality, however, the assumption of stationarity does not hold over the entire image domain or over fixed size patches (e.g. square blocks). Many natural images consist of different *regions* corresponding to surfaces of different objects in the scene, or to surfaces with different reflectance properties, or to surfaces under different illumination conditions. At the boundaries of these regions lie sharp intensity changes or *edges*. Therefore, homogeneously textured regions of arbitrary shape on the one hand and boundary edges on the other hand are quite characteristic features of many images.

Most waveform coding methods do not take these image characteristics into account. This leads to a loss in coding efficiency, or, alternatively, to visually disturbing artifacts in the reconstructed image, especially when these methods are applied at very low bit rates. Examples of these artifacts are *blocking effects* in DCT coding or *edge blurring* in DPCM. High compression systems should either be adaptive to the local characteristics of an image [60], or should be designed on the basis of a nonstationary model.

Some coding methods have been proposed, which take these local characteristics into account *implicitly*. For instance, a *two-component* approach has been proposed ([46][54][128]), in which smooth or low frequency components of an image are separated from the detailed or high frequency components. A *multi-scale* or *multi-resolution* representation emerges if the above procedure is iterated. This representation reflects the notion that the distinction between smooth variations and fine detail variations depends on the scale or resolution at which an image is being analyzed at some instant. Multi-scale and multi-resolution approaches have received widespread attention in the last decade or so [106]. Recently, *wavelet theory* has been studied in the context of multi-scale representations and the analysis of nonstationary signals [105], as well as in image compression [8].

A clear break with earlier coding methods is made by *explicitly* taking into account the geometry of the structure in images, such as regions or boundary edges. The analysis of this structure encompasses some kind of detection or segmentation stage preceding the actual coding stage, and this can be done in essentially two ways.

a. By concentrating on slowly varying parts of the image intensity field. The image domain is partitioned into sub-domains of arbitrary shape containing homogeneous intensities, using some kind of regional segmentation step (e.g. region growing). Since these sub-domains

form closed regions in the image, this can be called a *region-based* approach. An example of an image analysis method based on this kind of piece-wise homogeneous image model can be found in [35]. Coding methods based on this approach are called *contour/texture coding* or *segmentation-based coding* and were proposed in [17][42][69][70][75]. Following the segmentation, the region boundary and the region intensities are compressed separately. Recently, similar techniques have been applied in image sequence coding [85][109].

b. By concentrating on the rapidly changing parts of the image intensity field. Derivative filters and maxima or zero-crossing detectors are used to find such rapidly changing variations. These rapid variations correspond to features like edges, ridges, valleys, peaks, holes, etc. in the intensity surface. Therefore, this can be called a *feature-based* approach. Coding methods inspired by the feature-based model have been explored relatively sparsely, but see e.g. [4][26][37][44][70]. The feature extraction step is followed by a step in which intensity profiles along the features are analyzed and coded. The original image is often reconstructed from the feature profiles along with some additional low frequency information.

In this thesis, we follow the feature-based approach and, in particular, we improve and extend the method proposed by Carlsson in [26]. One reason for studying the feature-based approach is that the region-based approach has been studied more widely in recent years and its potentials for application in high compression systems have become increasingly clear. It appears that region-based coding methods do not perform significantly better than a standard block-wise DCT method using objective criteria such as the Signal-to-Noise Ratio (e.g. [17][104]), although most authors note an improvement in subjective quality. Feature-based approaches have not been studied to this extent, and it is one of the objectives of this thesis to explore them.

Another reason for applying the feature-based approach in image coding is the following. It has been noted by many authors that features such as intensity edges (i.e., sharp changes in the image intensity surface) describe important image structure and that they carry information about the structure of the original scene. As such, intensity edges form clues to recovering the structure of objects in the three-dimensional scene [7][52][76][83][84] as well as their motion [32][51]. An entire body of computer vision literature has been devoted to edge analysis. The above claim has been supported by observations of natural vision in animals and humans in neurophysiological and psychovisual research [31][76][99][101]. For instance, the eye appears to fixate more often on features with high information content, especially on contours that change direction rapidly [88]. Moreover, the human visual system can interpret terse sketches or line drawings with remarkable ease, which suggests that the intelligibility of a given image depends strongly on only a few perceptually important features. For instance, a few lines can convey object boundaries and, to some extent, textures as well. In a particular application such as lip-reading by deaf people via videophones, faithful reproduction of the shape and motion of parts of the face may be more important than the exact color of these parts. Studying representations in which more emphasis is given to *geometric* information (shape and motion) than to *photometric* information (texture or shading) appears worthwhile

in the context of high compression communication systems [26][92]. Here, intelligibility appears to be a more suitable requirement than quality in the narrow sense. Given the strict bit rate constraints, an approach that is primarily based on representing the edge features in image data may lead to useful results. Because the number of these perceptually relevant edge features can be kept small, and because the intensity variations associated with these features are spatially localized and can be described using only a few parameters, this approach promises a compact representation.

1.2.3 An edge-based image representation

An image can be viewed as consisting mainly of smoothly varying intensity surfaces interleaved occasionally by sharp intensity changes. These sharp intensity changes can be associated with different types of features, such as edges, ridges, valleys, peaks or holes. Many of these features are defined with respect to curves in the image plane, across which the image intensity function exhibits a one-dimensional behavior. In computer vision literature, these are collectively referred to as *edges*, e.g., *step edges* or *rooftop edges*. Step edges are associated with zero-order discontinuities in the intensity surface (discontinuities of the intensity function itself); rooftop edges are associated with first-order discontinuities (discontinuities of the first-order derivative of the intensity function). These types of features often correspond to surface reflectance changes, illumination changes and surface depth or orientation discontinuities in the scene. Other types of features, such as corners and junctions, are two-dimensional in nature and are often defined with respect to single points in the image plane. Those kinds of features are sometimes related to the topology of objects in the scene, e.g., they can signal occlusions. Still other kinds of features can be defined, each in some way connected to physical properties of the original scene.

Among the various features found in images, step edges appear to be the most abundant and important. Traditionally, the emphasis in computer vision literature has been on the analysis of step edges. In this thesis, we restrict ourselves to the description of *edge points* (one-dimensional case) and *edge curves* (two-dimensional case) with zero-order discontinuities in the image intensity function. Therefore, the approach is called *edge-based.*

A representation describing the intensity variations of edges can be considered a compact representation of the entire image intensity surface. It can convey perceptually relevant information about the image and about the imaged scene. Although such a representation might not be a complete description of the original image at first sight, the hope is that natural image data forms a sufficiently restricted class of intensity surfaces that a stable reconstruction might be found.

There is a long history of representations based on edges in computer vision literature. The interest in these representations started with Marr (e.g. [83]), who conjectured that a representation based on intensity changes, detected on several different scales, would be complete. However, Marr did not show how to reconstruct the original image given the edge-based representation. A brief survey of other studies that followed Marr's is given by Hummel et al. [53], who also present mathematical results and a stable reconstruction algorithm. More recently, the interest in edge-based representations has been revived because

of results by Mallat et al. [81][82] in the context of wavelet theory. These methods all employ a multi-scale or multi-resolution analysis, recognizing the fact that edges and, in general, intensity variations can occur over a range of scales. The scale of an intensity variation is determined by its *spatial extent* and its *amplitude*. A disadvantage of the multi-scale approach is that the analysis provides a large amount of data, probably containing redundant information. So, the representation is not compact at all, unless some of the data is discarded or combined with other pieces of data.

Of course, especially in image coding, one seeks a compact description of the input images. In this thesis, we follow an approach to the analysis of edge features that is fully model-based. We attempt to describe the sharp intensity variations in an image by adopting a simple parametric edge model and by straightforwardly fitting the model to these intensity variations. This fitting procedure is performed at all points in the signal where an edge is believed to be present. The location of these points (*geometric data*) combined with the model-parameters describing the intensity variations (*photometric data*) together form an image representation. The edge model includes the most important properties that characterize edge features, such as their location, contrast, orientation and width. The width property of edge transitions has not been taken into account in most region- or feature-based compression methods explored to date. It allows a better representation of intensity variations. It also allows us to relate an edge to a scale on which it manifests itself.

In the approach followed here, the intensity surface *between* edge features is not represented explicitly. The edge features in the representation describe the intensity surface only locally, namely along curves in the image plane corresponding to, e.g., object contours in the scene. From this sparse intensity data, an approximation to the complete image intensity surface across its entire domain must be recovered by applying some kind of interpolation. Since there cannot lie any sharp intensity transition between edges, the interpolation should result in a smooth surface. This interpolation can be based on a surface model. Although the representation by edge features alone does not allow exact recovery of the original input image, it is shown in this thesis that with the help of the surface model a uniquely determined image can be computed which is *perceptually close* to the original. Smooth surface interpolation has been thoroughly studied in computer vision literature, e.g. [48][112][115].

The above discussion of our approach indicates that preference is given to sharp intensity variations as image primitives, as opposed to texture elements. Textural variations are small, randomly appearing changes in the intensity field, associated with local variations of the reflectance, roughness of surfaces, very small objects and noise. In literature, computational models of texture and procedures to estimate their parameters have not been established yet to an extent that they can be incorporated in a more general model-based coding environment. Textural variations are not treated in this thesis. Of course, we recognize the fact that the edge-based approach will inevitably lead to a loss of quality in the decoded images due to a loss of texture. However, the primary concern in very high compression systems should be intelligibility, not quality in the narrow sense. This can be achieved by focussing on the sharp variations in images. This approach is not expected to lead to satisfactory results when applied to classes of images, in which textural variations play an important role in the intelligibility of

the scene being imaged. However, it is expected that this situation occurs fairly rarely. If necessary, the textural variations in an image can be treated as a kind of residual, which can be encoded separately from the main structural component. This residual should be highly compressible by a waveform coder.

1.3 Scope and outline of the thesis

1.3.1 Scope

We employ the edge-based approach to image representation and very low bit rate coding. We confine ourselves to luminance or grey-level images. Color information is usually transformed so that luminance and chrominance components can be separated. The luminance component is generally of more importance than the chrominance components, both to the intelligibility and the quality of an image. The method comprises the use of only two-dimensional models, i.e., three-dimensional properties of the scene being imaged are not modeled explicitly.

An edge model is used to describe intensity variations at every point in an image where an edge is believed to be present. Edge detection and analysis is set in the well-known framework of filtering with scalable derivative operators. Although the dependance of the method on the analysis scale is recognized, multi-scale integration of edge information is not discussed here.

From the intensity data along edges, a full intensity surface can be recovered using prior expectations about the nature of the images. Within the class of iterative approaches, often encountered in literature, we propose a new interpolation method, leading to faster convergence.

We apply this method to the compression of both still images and image sequences. Because the edge model-parameters still contain significant redundancy, further compression of this data can be achieved. Here, we use conventional methods from data compression and data reduction. Optimal bit allocation is outside the scope of this thesis.

In image sequence compression, the inherent temporal relationships between images in a sequence are utilized so as to reduce the amount of information to be transmitted or stored. This involves analysis of the correspondence of features originating from different images and interframe coding of edge feature data.

Coding of any residual textural component is not considered in this thesis. However, we do consider how the performance of an edge-based coder might be improved if knowledge about the content of the scene is used to adapt the local quality of the representation.

1.3.2 Outline

From the above, it follows that the main problems that the stated model brings about are: (*a*) *edge feature extraction*; (*b*) *intensity surface reconstruction*; (*c*) *image sequence analysis*; (*d*) *parameter coding*.

The main parts that make up a complete system for image compression based on our approach are depicted schematically in Figure 1.2, where the main processes at the transmitting side are

depicted at the top, while the main processes at the receiving side are depicted at the bottom. This simplified diagram is also used to give the outline of this thesis.

The first part of this thesis is depicted in the diagram on the left. It is concerned with the basic image representation, in which image analysis (edge detection and edge model-parameter estimation) and image synthesis (intensity surface reconstruction based on a surface model) play the most important role. The second part of this thesis is depicted in the diagram on the right. It is concerned with the model's application to image compression (using the statistics of model-parameters).

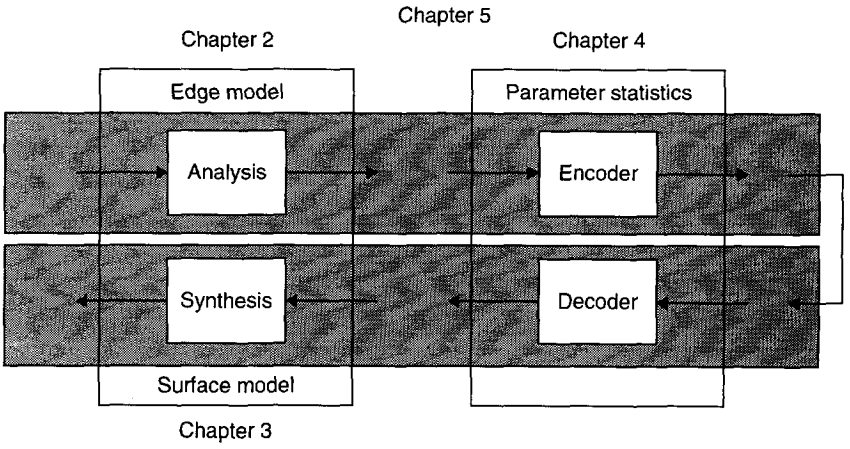


Figure 1.2 Edge-based image coding scheme and thesis outline.

The outline of this thesis then becomes as follows. The next chapter, **Chapter 2**, discusses the edge feature detection and parameter estimation problem. After the edge model is introduced, three different estimation methods are discussed. Following an error analysis of these methods, some experiments are reported. The one-dimensional formulation is then extended to the two-dimensional case. The edge-based image representation is defined, which describes the values of the intensities of points along edges.

In **Chapter 3**, the intensity surface reconstruction problem is discussed. The surface reconstruction problem is formulated in the context of regularization theory and leads to an iterative optimization algorithm. The introduction of a new, multi-scale, formulation of the problem is shown to lead to faster convergence of the iterative algorithm. Several experiments are reported, comparing the performance of the new algorithm to other, existing, algorithms.

The following two chapters address the efficient coding of the extracted information. Firstly, in **Chapter 4**, the still image case is outlined. The edge information consists of the geometry (or shape) of the loci of the edge features extracted and the photometric parameters (intensity data) associated with these features. Compression methods are designed for each component of the data, based on their statistics. The attainable compression ratio and the image quality are

illustrated with some experiments. Also, knowledge-based image compression in the context of the videophone application is discussed.

Secondly, in **Chapter 5**, image sequence coding is discussed. An edge feature matching algorithm is proposed, which provides the coder with estimates of the motion of these edge features. These motion parameters are then used to code the edge feature location data predictively. Also, a new lossy predictive edge location coding method is described. Some experiments are reported, comparing the performance of an interframe coder to an intraframe coder.

We finish this thesis with a general discussion in **Chapter 6**. In this chapter, we discuss our contributions and compare our approach to others in literature. We discuss implementation aspects and, finally, give some recommendations for further research.

Appendix A contains a convergence analysis useful to the discussion in Chapter 3. **Appendix B** provides some background information to Chapter 4 on the design of the edge primitive coder. Following the **References** chapter, a chapter with **Lists of Symbols, Definitions and Abbreviations** is included for convenience.

Chapter 2
Edge Analysis

In this chapter, we study a simple model-based method for analyzing edges in one- and two-dimensional signals and we describe how an edge-based image representation can be formed on the basis of the analysis results.

2.1 Introduction

This section elaborates on the nature of our representation, formulates the edge analysis problem and outlines the rest of the chapter.

2.1.1 Problem formulation

As mentioned in Chapter 1, *intensity edges* describe important image structure and form clues to recovering the structure of the original scene, in both human vision and computer vision. In this chapter, we discuss the modeling and analysis of such edges, with the intention of capturing the relevant information in images.

Edges occur either in *points* in one-dimensional signals (1-D) or along *curves* in two-dimensional (2-D) intensity surfaces or images. If one would try to characterize an image by its edge content, a description of such image structure should contain:

- the loci of edge points along relevant curves in the image,
- the variations of the image surface function within a small band along the edge curves.

Therefore, the first goal in the analysis stage is to detect the edge curves and to describe the variations of the image function across them. Of course, the edge curves cannot be extracted from an image directly: the curves themselves can have complicated shapes and the image field may vary along the curve in a complex way as well. The approach taken here is a very

common one: edge points are detected point-wise (more or less separately) and each detected point is then connected with neighboring edge points based on some definition of connectivity. In most images, this tends to form connected curves. Furthermore, at each point along an edge curve, the curve is assumed to be locally straight, such that the image intensity pattern at that point is locally one-dimensional and varies only in a direction perpendicular to the curve's local tangent direction. This assumption holds for most of the edge points in an image. These assumptions are implicit in many edge detection schemes, but usually they do not hold for other types of patterns, such as corners or junctions, the geometry of which has an inherent two-dimensional nature. Now, the edge detection problem and the problem of describing the image variation across edges can be treated as a one-dimensional problem.

Our approach to describing intensity variations across edge features is to adopt a simple model and straightforwardly fit the model to the signal at hand. One assumes the phenomenon underlying a feature is some kind of idealized function which can be described efficiently with a parameterized model. Here, we follow this approach to describe images and explore its potentials for image compression.

Only a few authors have used this kind of description for the purpose of efficient image coding [4][26][82]. None of them treated the problem of accurately describing or modeling the image intensity variation along edge curves in depth and none of them reported extensive experimentation.

2.1.2 Outline

A large part of this chapter is confined to the one-dimensional problem. In Section 2.2, an idealized one-dimensional edge model is introduced. In Section 2.3 we discuss the general approach to edge detection that we have adopted. Next, in Section 2.4, three methods to analyze and describe edges in one-dimensional signals are proposed, based on the edge models introduced before. We basically show that filtering of the signal can be used to detect an edge and simultaneously obtain measurements from which model-parameters can be estimated. An error analysis of these methods follows in Section 2.5 and some experiments are reported in Section 2.6. The one-dimensional edge and analysis model is extended to the two-dimensional case in Section 2.7. In Section 2.8 we explain how an edge-based image representation can be formed and how an image can be partially reconstructed from this data. We finish this chapter with conclusions.

2.2 Edge model

In this section, some ideal edge models are described. Let us first try to give an informal definition of edges. *Edges* in two-dimensional images are characterized by sharp intensity changes in one direction, together with little or no changes in the perpendicular direction. Thus, edges have locally a one-dimensional structure. Edges correspond to zero-order *discontinuities* in the characteristics of surfaces in the original scene. Edges have often been characterized by simple models such as *step edges*, *ramp edges* and *smoothed step edges*.

A model for an isolated ideal step edge in a 1-D function $f(x)$ at $x = 0$ is specified by

$$f(x;b,c) = cU(x) + b \ , \tag{2.1}$$

in which $U(x)$ is the unit step function, c is the *contrast* of the edge and b is the function value at the *base* of the edge. Infinitely sharp edges such as step edges, however, do not exist in practical signals or images, due to the band-limiting characteristics of the (optical) acquisition system. A linear model for an edge $s(x)$ in a signal, corresponding to a discontinuity in the scene function $f(x)$, distorted by a point spread function (psf) $p(x)$, is:

$$s(x) \equiv s(x;b,c,w) = f(x;b,c) * p(x;w) \ . \tag{2.2}$$

The psf $p(x; w)$ is a smoothing function and depends on some kind of resolution or smoothing parameter $w > 0$, related to the image formation process. If there is some kind of discontinuity in $f(x)$, then there will appear some kind of edge in $s(x)$. When, for instance, $p(x; w)$ is a rectangular pulse, a step edge in the original function $f(x)$ appears as a ramp edge in the signal $s(x)$. Specifically, a *Gaussian* function $g(x;\sigma)$, with σ the smoothing parameter, is used here as the psf:

$$g(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) . \tag{2.3}$$

The Gaussian is often used as an approximation to the psf of an optical system. Also, the use of the Gaussian is convenient mathematically.

A step edge as given in Eq. (2.1) convolved according to Eq. (2.2) by a Gaussian psf (so $p(x; w) = g(x; \sigma)$ with $\sigma = w$), appears as a scaled error function in the signal $s(x)$:

$$s(x;b,c,w) = \frac{c}{2}\left(1 + \mathrm{erf}\left(\frac{x}{w\sqrt{2}}\right)\right) + b \ , \tag{2.4}$$

with erf(.) the error function and w the parameter controlling the *width* of the edge. This edge model will further be referred to as the *Gaussian smoothed step edge* or simply *smooth edge*.

Thus, the model-parameters of a one-dimensional smooth edge are:

- contrast across the edge c;
- intensity at the base of the edge b or, alternatively, intensity at the edge center $m=b+c/2$;
- spatial width of the edge w.

These parameters are depicted schematically in Figure 2.1 for a Gaussian smoothed step edge. It is obvious that the contrast and center intensity parameters are more important in characterizing an edge than is its width. The width parameter w is bounded below by the blurring of the optical system. We allow for a different w parameter for each edge point, because optical blurring is potentially spatially variant and because an edge can have an "intrinsic" smoothness caused by unsharp discontinuities in the scene function f. Thus, the spatial width w of an edge is considered to be the result of *both* the intrinsic edge smoothness and the smoothness introduced by the image formation process.

For edge points in a two-dimensional signal, the orientation of the line tangential to the edge curve in the image plane is relevant as well, as is discussed later.

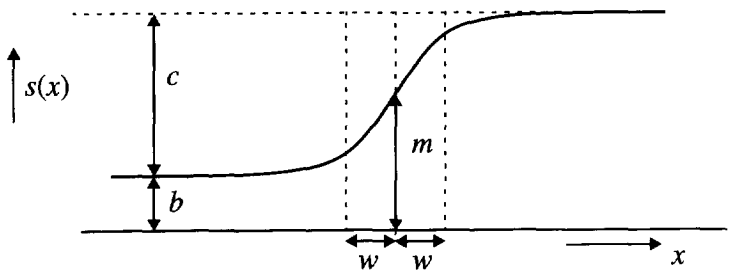


Figure 2.1 Gaussian smoothed step edge model in one dimension with center intensity parameter m, contrast parameter c, width parameter w and base intensity parameter b.

In practice, stochastic degradations are present in the signal as well. This can be included in our linear model as an additive noise term $n(x)$:

$$i(x) = s(x) + n(x) = f(x) * p(x) + n(x) \ . \tag{2.5}$$

Intensity changes can thus be approximated using a model parameterized by the above attributes. In practice, this model is actually valid only in a small area close to the center of the intensity change.

2.3 Edge detection method

A very large body of literature is devoted to the detection of edges in images, e.g. [18][25][49][50][74][83][119][121]. A common method is to use derivative operators, which respond to sharp variations in the signal. Usually a smoothing operator is applied as well to filter out some noise, which is inherently present in real signals. The necessity for smoothing can be formally justified on the grounds that differentiation needs to be *regularized*, due to the sensitivity to noise [119]. Regularization can be achieved by convolution with a smoothing kernel.

2.3.1 Derivative and smoothing operators

The most common derivative operators are the first and second derivative operators, since edges corresponding to zero-order discontinuities lead to maxima in the first derivative of the signal and to zero-crossings in the second derivative. Common operators in the two-dimensional case are the *gradient*, the *second derivative in the gradient direction* and the *laplacian*. We primarily use first (directional) derivative operators and the gradient. First derivative operators are less sensitive to noise than are second derivative operators.

Many types of smoothing filters can be used (e.g. box-operators, smoothing splines, Gaussians), each optimal in a certain sense. In practice, very often a Gaussian filter is used, because it is optimal or near-optimal in many senses.

- A multi-dimensional Gaussian is separable, which greatly reduces the computational complexity of the multi-dimensional convolution or filtering stage at the heart of a computer implementation.
- The real-valued Gaussian function minimizes a joint space-frequency uncertainty criterion, i.e., it is localized well both in the spatial and frequency domain [119]. Hence, Gaussians are the best trade-off between support-limited and band-limited filters.
- It can be shown that the Gaussian is the unique filter kernel for which no spurious detail is generated if the amount of smoothing is increased, which implies a nice smoothing behavior (see [67] and [130] for related results).
- The derivative of a Gaussian is near-optimal with respect to various edge detection performance criteria (detection, localization and uniqueness), formulated mathematically by Canny in [25]. Canny showed that the first derivative of a Gaussian is a good approximation to the optimal detector, in the case of a one-dimensional step edge hampered by additive white Gaussian noise. Note that the validity of Canny's localization criterion has been subject to discussion, see [22], [68], [113] and [114].

2.3.2 Canny edge detection

The edge detection scheme used here is the same as that of Canny [25]. In one-dimensional signals, this corresponds to filtering with a Gaussian smoothing kernel and differentiation. These two steps are combined into filtering with the derivative of a Gaussian function:

$$g'(x;\sigma) = \frac{-x}{\sigma^3\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right), \tag{2.6}$$

with σ the smoothing parameter. The computer implementation uses a sampled version of Eq. (2.6) truncated to an interval $-\Delta \le x \le \Delta$. The effect of the truncation is negligible when choosing $\Delta \ge 5\sigma$. The effect of the discretization is studied later. After the signal is filtered with the derivative of a Gaussian, edges are detected by marking local maxima in the magnitude of the filter output. This is the edge detection and localization method that is applied throughout this thesis.

The smoothing step used in edge detection introduces an extra parameter, namely the size of the smoothing filter or the size of the neighborhood taken into account. This parameter is useful to control the *scale* of the filter, i.e., the level on which an image is analyzed. When one uses a Gaussian smoothing kernel, the smoothing parameter σ is used as the scale parameter. The notion of analysis scale can be justified by the fact that intensity changes in images occur on several scales. When one uses operators of a large size, small changes are effectively filtered out. However, when one uses operators of a small size, the processing is sensitive even to small scale variations. With the scale parameter, a family of images can be derived (each smoothed to a certain extent) often called a *scale space* (see [67] and [126]). Analysis of (the

behavior) of signals across the scales (from coarse to fine or vice versa) is called a *multi-scale* analysis, (e.g. multi-scale edge detection [83]). The concept of scale has turned out to be very useful in image processing and we will show that it is relevant to our problem as well.

2.4 Parameter estimation methods

Computation of the properties of edges given a signal containing those edges, is basically a model fitting or parameter estimation problem. Here, we use a simple parameterized model and obtain observables by convolution of the signal with kernels identical to those used in edge detection. Such model-based estimation of the properties of edges has received relatively minor attention in the image analysis literature. While gradients of images have of course been considered extensively in edge detection, few authors have investigated the recovery of the contrast and width of an edge from the gradient value. One approach is to analyze edges on different scales using filters of different sizes, e.g. [29], [47], [87], [89], [124] and [125]. The width of the edge is either found by searching through many different scales or by (over-) constraining the unknown parameter values given a few values of the gradient evaluated at different scales. Most of these authors report no error analysis or experimentation, or only experiments with very wide and noiseless synthetic edges. Another approach is to use filters of different derivative order, e.g. [65]. Some methods, e.g. proposed in [29] and [74], apply only to the one-dimensional case. In another context, the width or smoothness of edges in images has been considered in methods to estimate scene depth from single images, e.g. [71] or [94]. This is because the amount of smoothing in an optical acquisition system actually depends on the distance of the point being imaged to the camera (as well as on its band-limiting properties). The simple method to characterize an edge used by Carlsson [26] does not take into account the width of an edge. Also, this method can easily lead to a loss of edge contrast.

We propose three methods to estimate both the edge contrast (c) and width (w). These simple methods follow straightforwardly from the previous equations, based on the use of a Gaussian smoothed step edge model and Gaussian derivative filters. The first method (called "*multi-scale estimation*") uses multiple measurements of smoothed gradient values at the edge center obtained on different scales. The second method (called "*multi-derivative estimation*") uses measurements of different smoothed derivatives at the edge center obtained on one scale. The third method (called "*multi-point estimation*") uses measurements of smoothed derivatives at different points near the edge center obtained on one scale. As few measurements as possible are used. Estimation of the edge center intensity value (m) is not discussed, since it can be obtained from the signal directly, once the edge has been localized.

2.4.1 Multi-scale estimation

Consider the 1-D Gaussian smoothed step edge $s(x)$ given by Eq. (2.4). As mentioned, this signal is filtered during edge detection with the derivative of a Gaussian, given by Eq. (2.6). Ignoring the noise, the model output $d(x; c, w, \sigma)$ of the filtering stage is (using [45]):

$$d(x;c, w, \sigma) = s(x;b, c, w) * g'(x;\sigma) = \frac{c}{\sqrt{2\pi(w^2+\sigma^2)}} \exp\left(\frac{-x^2}{2(w^2+\sigma^2)}\right). \quad (2.7)$$

The shape of the model output is the same as a Gaussian function with scale parameter $\sigma' = \sqrt{w^2 + \sigma^2}$. The peak value of this output lies at the edge center, in this case at $x = 0$, and depends only on the edge contrast c, edge width w and edge detection filter scale σ. The edge is detected and localized by marking this peak in the filter output. The filter scale parameter σ is known. Therefore, in theory, the two unknown edge parameters c and w can be obtained exactly using only the measurements of two peak values of two different responses, obtained by filtering on two scales σ_1 and σ_2 and subsequent peak-detection in both filter outputs. From the following two measurements:

$$\begin{aligned} d_1 \equiv d(0;c, w, \sigma_1) &= \frac{c}{\sqrt{2\pi\,(w^2 + \sigma_1^2)}} \\ d_2 \equiv d(0;c, w, \sigma_2) &= \frac{c}{\sqrt{2\pi\,(w^2 + \sigma_2^2)}} \end{aligned} \tag{2.8}$$

with $\sigma_2 > \sigma_1 > 0$, one can obtain:

$$w^2 = \frac{\sigma_2^2 d_2^2 - \sigma_1^2 d_1^2}{d_1^2 - d_2^2} \tag{2.9}$$

and

$$c^2 = (d_1 d_2)^2 \, \frac{2\pi\,(\sigma_2^2 - \sigma_1^2)}{d_1^2 - d_2^2} \, . \tag{2.10}$$

The latter equations are equivalent to those found in [47].

2.4.2 Multi-derivative estimation

An alternative to filtering the edge signal with the derivative of a Gaussian on several scales is to filter with further *derivatives* all on *one scale*. In particular, one can use the *first* and *third* derivative of a Gaussian. Again, the output of filtering the Gaussian smoothed step edge with the *first* derivative of a Gaussian is used to detect and localize the edge. Filtering the Gaussian smoothed step edge with the *third* derivative of a Gaussian yields the second derivative of the response given by Eq. (2.7):

$$d''(x;c, w, \sigma) = s(x;b, c, w) * g'''(x;\sigma) = \frac{c\,(x^2 - w^2 - \sigma^2)}{\sqrt{2\pi}\,(w^2 + \sigma^2)^{5/2}} \exp\left(\frac{-x^2}{2\,(w^2 + \sigma^2)}\right) . \tag{2.11}$$

If we now define d_2 as the value of this response at $x = 0$ and define d_1 as before,

$$\begin{aligned} d_1 \equiv d(0;c, w, \sigma) &= \frac{c}{\sqrt{2\pi\,(w^2 + \sigma^2)}} \\ d_2 \equiv d''(0;c, w, \sigma) &= \frac{-c}{\sqrt{2\pi}\,(w^2 + \sigma^2)^{3/2}} \end{aligned} \tag{2.12}$$

then we obtain:

$$w^2 = -\frac{d_1}{d_2} - \sigma^2 \tag{2.13}$$

and

$$c = d_1 \sqrt{2\pi (w^2 + \sigma^2)} = d_1 \sqrt{-2\pi \frac{d_1}{d_2}} . \tag{2.14}$$

Thus, c and w can be computed using the responses of the edge signal to first and third derivatives of a Gaussian both on the same scale σ.

2.4.3 Multi-point estimation

Another possibility to compute estimates of c and w is to sample the response of the edge to the first derivative of a Gaussian filter *at multiple points*, i.e., one uses the output of the filtering *on one scale* at a few points near the edge center. Again, the peak in this output is used to detect and localize the edge. Note that the position of this peak is constrained to grid point locations in practice (where one works with sampled signals instead of continuous ones); hence, it does not always coincide with the *true* edge center. In the multi-point method, any deviation of the position of the peak value in the filter output with respect to the position of the true edge center can be computed, as is shown below.

If the actual position of an edge is at $x = x_0$ instead of $x = 0$, then the actual model response is:

$$d(x;c, w, \sigma) = s(x - x_0;b, c, w) * g'(x;\sigma) = \frac{c}{\sqrt{2\pi (w^2 + \sigma^2)}} \exp\left(\frac{-(x - x_0)^2}{2 (w^2 + \sigma^2)} \right) , \tag{2.15}$$

instead of Eq. (2.7). Assume the *detected peak* lies at $x = 0$ (coinciding with a grid point); hence, one has $|x_0| < 0.5$. Using the following three measurements of this response:

$$\begin{aligned} d_1 \equiv d(0;c, w, \sigma) &= \frac{c}{\sqrt{2\pi (w^2 + \sigma^2)}} \exp\left(-\frac{x_0^2}{2 (w^2 + \sigma^2)} \right) \\ d_2 \equiv d(a;c, w, \sigma) &= \frac{c}{\sqrt{2\pi (w^2 + \sigma^2)}} \exp\left(-\frac{(a - x_0)^2}{2 (w^2 + \sigma^2)} \right) \\ d_3 \equiv d(-a;c, w, \sigma) &= \frac{c}{\sqrt{2\pi (w^2 + \sigma^2)}} \exp\left(-\frac{(-a - x_0)^2}{2 (w^2 + \sigma^2)} \right) \end{aligned} \tag{2.16}$$

one can obtain:

$$w^2 = \frac{a^2}{\ln(\frac{d_1 d_1}{d_2 d_3})} - \sigma^2 , \tag{2.17}$$

$$x_0^2 = \frac{\ln(d_2/d_3)}{2a}(w^2+\sigma^2) = \frac{a\ln(d_2/d_3)}{2\ln(\frac{d_1 d_1}{d_2 d_3})} \tag{2.18}$$

and

$$c = d_1\sqrt{2\pi(w^2+\sigma^2)}\ \exp\left(\frac{x_0^2}{2(w^2+\sigma^2)}\right) = d_1\sqrt{\frac{2\pi a^2}{\ln(\frac{d_1 d_1}{d_2 d_3})}}\left(\frac{d_2}{d_3}\right)^{\frac{1}{4a}}. \tag{2.19}$$

The distance a can, in principle, be chosen freely, but $a = 1$ is a practical choice in sampled 1-D signals. Thus, c and w can be estimated using the output of filtering of a model edge with the first derivative of a Gaussian on one scale. Also, subpixel localization is enabled by using three points of this output near the peak instead of two.

2.4.4 Discussion

So far, only the 1-D case has been discussed. In the 2-D situation, the problem is converted back into the 1-D problem by analyzing the 2-D signal in the direction of the local gradient, as will be explained in Section 2.7. This means one needs to use (smoothed) directional derivative responses in the direction of the gradient, which must be formed using partial derivatives in the x- and the y-directions. In the case of the first derivative in the direction of the gradient, these are the first partial derivatives in the x- and y-directions which together form the gradient itself: these must always be evaluated. However, in the case of the third derivative in the direction of the gradient, as many as *nine* different partial derivatives have to be evaluated - a considerable computational burden. This is a significant disadvantage of the multi-derivative method, if it is to be applied in a 2-D space.

So far, only the case of a single, isolated edge has been discussed. In real signals, *many* edges may be present at various unknown locations. In that case, it is less trivial to decide which detected edges on a particular scale correspond to which edges on another scale. This *correspondence* between detected edge points on two different scales must be established in the multi-scale method to relate two points to one and the same underlying edge and to be able to compute its parameters. This problem becomes even more pressing in 2-D signals, because the edge curves which are formed in 2-D images can *deform* as well as dislocate on coarse scales. Therefore, one will need to *track* edge points or edge curves from one scale to the next coarser (or finer) scale in order to make the correspondence. A few approaches to this problem are known in the literature which may be applied here [12][100], at the expense of extra computational costs.

Thus, the multi-point method is the least computationally demanding method of the three methods, particularly with practical 2-D signals. Also, it is the only method which can compensate for the effects that occur when the actual center of an edge lies somewhere between two grid points.

The three methods to estimate the contrast and width of edges in 1-D signals proposed above do not in any way consider deviations from the model response, neither systematic nor stochastic. Moreover, these methods are based on a *continuous-time* model, whereas in the practice of digital signal processing, one deals with *discrete-time* (or discrete-space) signals, i.e., signals defined on a set of discrete grid points only. Here, we assume Eq. (2.8), Eq. (2.12) and Eq. (2.16) to be valid as a first-order approximation. We assume the component in the filtered signals corresponding to the noise response is small, because of the noise-suppressing qualities of Gaussian smoothing. By the error analysis and experiments reported in the following sections, we hope to be able to adjudicate on the matter of *when* this approximation is accurate enough and *when* it breaks down. Specifically, the results may be used to guide the choice of an appropriate filter scale σ, which is the most important free parameter in all methods. In general, the filter scale determines the amount of noise smoothing and, at the same time, the size of the filter. Using a large filter size means powerful noise smoothing, but it also means that any features in the signal present near the edge being analyzed clearly influence the estimate of the parameters of that edge.

2.5 Error analysis

As mentioned, the above theoretical model is an approximation to the situation encountered in real signals. In practice, errors will occur caused by, e.g.,

1. the effects of using a discrete-time approximation to the continuous-time model,
2. the effects of noise being superposed onto the model signal,
3. the effects of other, nearby, edges not taken into account in the model.

These effects are examined theoretically in this section and some experimentation is reported in the next section. In this section, we specifically study the effects of these errors on:

- the fidelity of the peak value in the filter response,
- the location of the peak.

Of course, the effects of the mentioned errors on:

- the accuracy of the computed values of the contrast and width,
- the fidelity of the reconstructed edge,

are even more important. The accuracy in the computed contrast and width estimates and the fidelity of an edge, reconstructed with these estimates, depend not only on the errors in the filter peak value. They depend also, of course, on the sensitivity of the subsequent computations to these errors. It can be seen from Eq. (2.9), Eq. (2.13) and Eq. (2.17) that computation of w depends on the *ratios* between different responses only, while computation of c depends on absolute values of responses also, as can be seen from Eq. (2.10), Eq. (2.14) and Eq. (2.19). However, these effects are hard to analyze theoretically, due to the severe nonlinear nature of these equations. Therefore, the errors in the computed contrast and width estimates and the fidelity of the final reconstructed edge are studied experimentally in Section 2.6.

2.5.1 The influence of discretization

In practice, the continuous-time convolution integrals implicit in Eq. (2.7), Eq. (2.11) and Eq. (2.15) are approximated by a discrete summation of samples of the input signal multiplied by filter samples. This approximation is exact when the Fourier transforms of the signal and filter are both band-limited in the sense of the sampling theorem and when the sampling frequency is high enough [91]. The Fourier transform of our model edge is (using [45]):

$$\frac{c}{j\omega_x}\ \exp\left(-\frac{w^2\omega_x^2}{2}\right)+2\pi\left(b+\frac{c}{2}\right)\delta(\omega_x)\ ,$$

with ω_x the spatial frequency and $\delta(.)$ the Dirac delta function. The Fourier transforms of the first and third derivative of a Gaussian are:

$$j\omega_x\ \exp\left(-\frac{\sigma^2\omega_x^2}{2}\right)\quad\text{and}\quad(j\omega_x)^3\ \exp\left(-\frac{\sigma^2\omega_x^2}{2}\right).$$

Unfortunately, none of these Fourier transforms is band-limited in a strict sense. The accuracy of the approximation depends on how fast the magnitude of these Fourier transforms fall off at high frequencies. The magnitudes of the above transforms fall off faster for increasing values of w, respectively σ. Therefore, the wider an edge (high w) and the smoother the filter (high σ), the more aliasing of high frequency components is suppressed and the more accurate is the discrete filter response. The filter scale parameter can be controlled; the edge width can not be controlled. Information on the origin and digitization of the test-images used in this thesis is not available. However, in practice, one may assume that the images are properly band-limited and sampled by the acquisition equipment (such as camera's). In any case, the continuous model is an approximation to the actual situation, and the values of w and σ influence the fidelity of this approximation. More about the sampling requirements for signals when using these filters can be found in [122].

Another problem is that the true center of an edge hardly ever coincides with a point of the discrete grid. This leads to errors in the responses $d(x;c, w, \sigma)$ and $d''(x;c, w, \sigma)$ taken at points on the grid. Therefore, in the first (multi-scale) and second (multi-derivative) parameter estimation methods, one may have to use an edge detection method with subpixel localization accuracy and use some form of interpolation of the response between grid points to get a better estimate of the peak value. In the case of the third (multi-point) method, any shift of the location of the true edge center with respect to the discrete grid is computed and compensated for. Therefore, in the absence of any other deviations from the model, the error in the response due to a slightly shifted edge is not significant with the latter method.

2.5.2 The influence of noise

In this subsection, the effects of the presence of white Gaussian noise (superposed onto the edge model signal) on the filtering stage are discussed. Canny's signal-to-noise ratio criterion and localization criterion (cf. [25]) can be used to assess the performance of a filter in revealing an edge $s(x)$, which is present in a signal $i(x)$ but hampered by additive white

Gaussian noise $n(x)$ (see Eq. (2.5)). Assume the true edge is centered at $x = 0$. The signal-to-noise ratio criterion, denoted by Ω, rates how well the peak value in the response can be resolved in the presence of noise [25]:

$$\Omega = \frac{|d(0)|}{\sqrt{Var\{n(x) * h(x)\}}} \tag{2.20}$$

where $d(x) = s(x) * h(x)$ and $h(x)$ is the edge detection filter used. The localization criterion, denoted by Λ, rates the variance in the displacement x_d of the local maximum in the amplitude of the filter response with respect to $x = 0$ due to the noise [25]:

$$\Lambda = \frac{1}{\sqrt{E\{x_d^2\}}} \approx \frac{|d''(0)|}{\sqrt{Var\{n(x) * h'(x)\}}} \tag{2.21}$$

where $d''(x) = s(x) * h''(x)$. Note that Canny's model of the edge location error seems valid only under small and moderate noise conditions [68].

For the Gaussian smoothed step edge and the Gaussian *first* derivative filter, these yield (using [45] or Mathematica [127]):

$$\Omega_1 = \frac{c}{\sigma_n}\sqrt{\frac{2\sigma^3}{(w^2+\sigma^2)\sqrt{\pi}}} \quad \text{and} \quad \Lambda_1 = \frac{c}{\sigma_n}\sqrt{\frac{4\sigma^5}{3(w^2+\sigma^2)^3\sqrt{\pi}}}, \tag{2.22}$$

with σ_n the standard deviation of the noise and σ the filter scale parameter. These are plotted as a function of σ in Figure 2.2, for an edge with $w = 1.0$ and $c = 100.0$ plus noise with $\sigma_n = 1.0$. The signal-to-noise ratio Ω_1 rises monotonically with σ, while the localization Λ_1 reaches a maximum at $\sigma = w\sqrt{5}$ and then decays slowly to zero. Therefore, if the edge width is nonzero, then the optimum scale, as far as localization is concerned, will be nonzero as well. It appears from these graphs that one should choose the filter scale σ to a large value in order to average out the noise and reach higher signal-to-noise ratios, but not too large because then the localization performance would suffer. These two criteria are of concern for each of the three methods we are studying, because detecting and localizing an edge in the first place is performed identically in all three methods. Edge localization performance in the presence of noise has also been studied in [61].

In the multi-derivative method, the fidelity of the response of the edge to a *third* derivative of a Gaussian filter is relevant as well. Although not used in the detection of edges, this response is used in the parameter estimation. A signal-to-noise ratio criterion similar to Eq. (2.20), but using the Gaussian third derivative filter and its response, yields (with Mathematica [127])

$$\Omega_2 = \frac{c}{\sigma_n}\sqrt{\frac{8\sigma^5}{15(w^2+\sigma^2)^3\sqrt{\pi}}}. \tag{2.23}$$

It can be shown that $\Omega_2 \leq \Omega_1$ for all σ, so the multi-derivative method seems to be more noise sensitive due to the use of a third derivative filter.

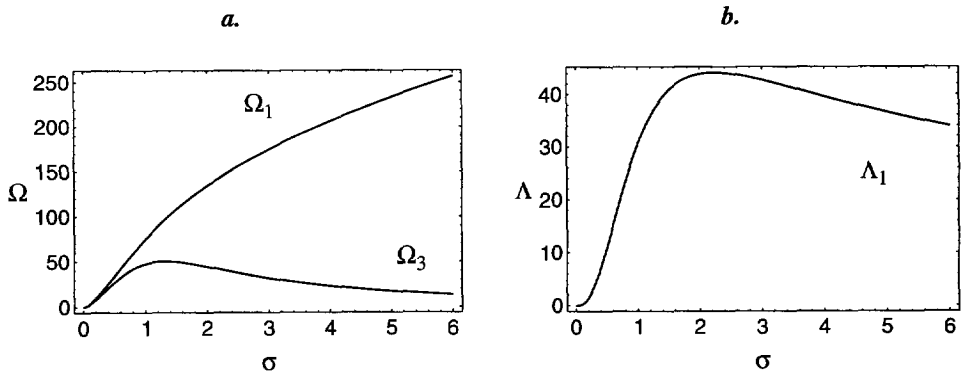


Figure 2.2 (a) Signal-to-noise ratio criterion Ω_1 for the Gaussian smoothed step edge model and Gaussian first derivative filter, compared to a difference signal-to-noise ratio criterion Ω_3 with $a = 2.0$; (b) Localization criterion Λ_1.

For the multi-point method proposed before, one can define another signal-to-noise ratio criterion, which rates how well the different values of the same response, used in the estimation, can be discerned from one another under noisy circumstances. The difference between the values of the response sampled at $x = 0$ and at $x = a$ now carries this information, although this quantity is not explicitly used in the parameter estimation. We propose (assuming no shift of the edge center):

$$\Omega = \frac{|d(0) - d(a)|}{\sqrt{Var\{n(x) * h(x)\}}}, \tag{2.24}$$

which yields for the model edge and Gaussian first derivative filter:

$$\Omega_3 = \Omega_1 \left| 1 - \exp\left(-\frac{a^2}{2(w^2 + \sigma^2)} \right) \right|. \tag{2.25}$$

It can be shown that, for a smooth edge, $\Omega_3 \leq \Omega_1$ for all σ, so the multi-point method seems more noise sensitive than the multi-scale method. It is obvious that this "difference signal-to-noise ratio" decreases with decreasing a. Hence, the distance a between the first and second response value has a strong influence on the performance under noisy conditions - this parameter acts as a kind of "internal" or "hidden" scale. It might be advantageous to let a increase with increasing σ. The difference ratio for $a = 2.0$ is compared with the original ratio in Figure 2.2(a), for an edge with $w = 1.0$, $c = 100.0$, $\sigma_n = 1.0$. It can be seen that Ω_3 initially follows Ω_1 but declines when σ grows larger.

2.5.3 The influence of nearby edges

Thus far, only isolated edges have been discussed. Of course, in real signals (such as images) one finds several edges in several places. Naturally, the filter response of a particular edge is

affected by the presence of another edge, especially when the scale of the filter is large in comparison to the distance between these two edges. Obviously, the peak value of this response can change because of the influence of a nearby edge. Also, the peak location can shift. Further, "phantom" edges can appear, which are due to spurious extrema in the filter response. Lastly, edges can disappear when analyzed on large scales (i.e. a peak in the filter response is no longer present). This kind of behavior has been described in depth in [79].

Let's consider two, relatively nearby, model edges. One at $x = x_1 = 0$ with contrast c_1 and width w_1 and another at $x = x_2$ with contrast c_2 and width w_2. The width of these edges is assumed to be small with respect to the distance between them. Two types of interaction between these edges can occur. Suppose, without loss of generality, that $c_1 > 0$; then if c_2 has the same sign, the edges together form a *staircase* edge; if c_2 has the opposite sign, the edges together form a *pulse* edge. Both are illustrated in Figure 2.3. Each causes a distinct type of behavior of the location estimate and filter response of each constituent edge.

Figure 2.3 (a) Pulse edge; (b) Staircase edge.

First, we analyze the location error for one edge due to interaction with the other edge. The total filter response $d(x; c_1, c_2, w_1, w_2, \sigma)$ of the two edges to the Gaussian derivative filter - ignoring the noise response - is

$$d(x) = \frac{c_1}{\sqrt{2\pi\,(w_1^2+\sigma^2)}}\exp\left(\frac{-x^2}{2\,(w_1^2+\sigma^2)}\right)+\frac{c_2}{\sqrt{2\pi\,(w_2^2+\sigma^2)}}\exp\left(\frac{-(x-x_2)^2}{2\,(w_2^2+\sigma^2)}\right). \quad (2.26)$$

The location of each edge is given by the peaks in this response. However, the extrema given by $d'(\hat{x}) = 0$ cannot be found analytically. Here, an edge location estimate is denoted by $\hat{x}$. One can show that for the pulse edge, the peaks in the filtered signal move further apart as the filter scale σ increases, while for the staircase edge, the peaks move closer together as σ increases and one of them eventually disappears. In any case, the location error increases as the filter scale increases.

The question now arises as to what the *upper bound* on σ is, in order to keep the location error below a pre-specified limit. To answer this, we first have to suppose that $w_1 = w_2 = w$ and use an auxiliary variable τ, with $\tau^2 = w^2 + \sigma^2$. One can now solve $d'(\hat{x}) = 0$ for τ. This yields:

$$\tau^2 = \frac{x_2^2 - 2\hat{x}x_2}{2\,\ln\left(-\frac{c_2}{c_1}\left(1-\frac{x_2}{\hat{x}}\right)\right)}$$

with $\hat{x}$ the location estimate of a peak. Now let's assume that $x_2 > 0$ (the second edge is at the right of the first edge) and that we have a pulse edge with $c_2 = -c_1$. We concentrate on the left edge, which is actually located at $x = x_1 = 0$ but its peak shifts to the left when $\tau > 0$, therefore $\hat{x}_1 < 0$. Define this peak shift (location error) as $x_d \equiv |\hat{x}_1 - x_1| = |\hat{x}_1| = -\hat{x}_1$, so $x_d > 0$. It follows that if one wants to keep the shift below some value, $x_d < D$, then one should have

$$\tau^2 < \frac{x_2^2 + 2Dx_2}{2 \ln\left(1 + \frac{x_2}{D}\right)} . \tag{2.27}$$

Given the values w and x_2, this formula gives an upper bound on the filter scale σ for an allowable shift. A useful value is $D = 0.5$ (corresponds to a location error below pixel-accuracy). For example, for a pulse edge with $x_1 = 0$ and $x_2 = 4$, one should have $\tau^2 < 4.55$ approximately. This means that if the flanks of the pulse edge are quite sharp, say $w = 0.5$, then one should keep σ below approximately 2.1 in order to keep the location error x_d below 0.5.

Besides the location error, there will also be an error in the value of the filter response being taken at or near the estimated location of the edge center. Here, we seek to specify an upper bound on σ to keep this error below a pre-specified limit, for each of the cases relevant in the proposed methods. These are: the error in the peak value of the response of the edge to a first derivative of a Gaussian filter (for all three methods), the error in the peak value of the response of the edge to a third derivative of a Gaussian filter (multi-derivative method) and the error in the values of the response of the edge to a first derivative of a Gaussian at a certain distance left and right of the peak (multi-point method).

Let's consider again the same two nearby model edges with the same width w, but possibly different contrast c_1 and c_2, lying at $x = x_1 = 0$ and at $x = x_2$. We assume the location error is zero. Then, one can derive straightforwardly from Eq. (2.26) that in order to keep the relative error in the response value at x of the first edge to a *first* derivative of a Gaussian filter, $d(x)$, below 5%, one should have

$$\tau^2 < \frac{x_2 (x_2 - 2x)}{-2 \ln\left(0.05 \cdot \left|\frac{c_1}{c_2}\right|\right)} . \tag{2.28}$$

The corresponding upper bound on τ has been illustrated in Figure 2.4(a) as a function of x_2, for the case when $|c_1| = |c_2|$ and $x = 0$, i.e., presumably near the peak value of the response. In this case, τ $(=\sqrt{w^2 + \sigma^2})$ should lie below approximately 2/5 x_2. As an example, if $x_2 = 5.0$ (the distance of the second to the first edge is 5.0), then $\tau < 2.0$ or $w^2 + \sigma^2 = \tau^2 < 4.0$. Further, if $w = 1.0$, then one should keep σ below 1.7. This upper bound seems quite strict.

Also shown in the same figure is the upper bound on τ to keep the relative error in the response at $x = 0$ of the first edge to a *third* derivative filter below 5%. This upper bound cannot be evaluated analytically, but was found numerically for values of x_2 between 0.5 and 20.0. This curve shows that the upper bound on τ is even tighter in this case than in the case of the first

derivative filter. This may be explained by the fact that the third derivative filter is a little wider than the first derivative filter, hence its response is hampered sooner by nearby features.

In Figure 2.4(b), the upper bound on τ for the first derivative case with $|c_1| = |c_2|$ and $x = 0$ is compared to upper bounds when $x = -1$ and $x = 1$. These are given by Eq. (2.28). The relative error in the values of the response *left* and *right* of the response peak is kept below 5% if the latter upper bounds are not violated. This case is of interest in the multi-point method. The response left of $x = 0$ is less sensitive to the edge at x_2, but in practice one does not know a priori how edges are positioned with respect to each other.

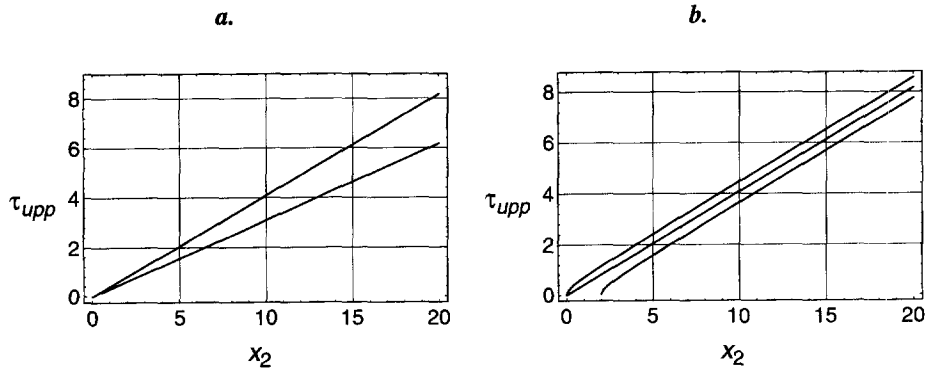


Figure 2.4 (a) Upper bounds on τ for the response at $x = 0$ to a first derivative filter (top curve) and a third derivative filter (bottom curve); (b) Upper bounds on τ for the response at $x = -1$ (top curve), $x = 0$ (middle curve) and $x = 1$ (bottom curve) to a first derivative filter.

2.5.4 Discussion

In the above analysis, we have studied the effect of some model-deviations on the location of the filter response peak and on the fidelity of this peak value. The requirements on the filter scale σ (which is the most relevant free parameter in each method) for minimizing these effects are conflicting: *large* scales should be used to be able to use a discrete approximation to the continuous model; scales *proportional to the edge width* should be used to optimize both signal-to-noise ratio and localization criteria in the presence of noise; scales *small* with respect to the distance to other edges should be used to minimize their influence. These conflicting requirements pose a trade-off between *resolution* and *noise resistance*, a well-known trade-off in edge detection and other problems in image processing. A scale which is approximately *equal* to the edge width may provide an optimal trade-off between these requirements. Thus, in general, the best results are likely to be obtained if the scale of the analyzing filter matches the width of an edge being analyzed.

2.6 Experimental results

To corroborate the analysis from the previous section and to further quantify the influence of the model-errors, extensive simulations were carried out with synthetic edge profiles. The

influence of discretization, noise and a nearby edge was studied for all three parameter estimation methods. In all these experiments, sampled versions of continuous 1-D edge models (with the edge center at $x = 0$) and of Gaussian first derivative filters $g'(x;\sigma)$ and third derivative filters $g'''(x;\sigma)$ were formed, with various parameter values - most importantly w and σ. The value of b (or m) is arbitrary, since it has no influence on the response. The value of c is not completely arbitrary, but it turned out to have very little influence on the relative errors made during computation of the response, hence it was kept constant most of the time. The length of the edge signal was chosen such that border effects never played a role (about 100 samples). The length of the 1-D filter was chosen such that the necessary truncation of the tails of the filter had negligible influence ($\Delta = 8\sigma$). To obtain the response, the synthesized edges were convolved discretely with a filter, i.e. using summation. The edge center was localized by searching for the maximum value in the output (of the response to a *first* derivative filter). The computed estimates of the edge center location, edge contrast and edge width are denoted by $\hat{x}_0$, $\hat{c}$ and $\hat{w}$ respectively.

The simulations showed that all three parameter computation methods perform approximately equally as far as accuracy is concerned. Hence, for reasons of brevity, the results of experiments using the third - *multi-point* - method are reported here only.

2.6.1 The influence of discretization

In this experiment, we study the differences between the continuous-time model and the discrete-time implementation.

Firstly, the relative errors in the filter response values were computed, with w and σ set to various values. The simulations showed that the relative errors in $d(\hat{x}_0;c, w, \sigma)$ (the peak value of the response of a smooth edge to a *first* derivative of a Gaussian filter), in $d''(\hat{x}_0;c, w, \sigma)$ (the value of the response of an edge to a Gaussian *third* derivative filter at the detected peak) and in $d(\hat{x}_0 \pm 1.0;c, w, \sigma)$ (the value at a distance 1.0 from the peak of the response of an edge to a Gaussian first derivative filter) all behave quite similarly. As expected, these relative errors decrease quickly; they fall below 5% when $w \geq 0.6$ and $\sigma \geq 1.0$.

Naturally, errors in the response values lead to errors in the estimates of the edge contrast $\hat{c}$ and edge width $\hat{w}$. Two illustrative plots of the relative errors in these estimates, computed using the multi-point method, are shown in Figure 2.5. The relative error in $\hat{w}$ is high when the actual value of w is very small, i.e., when the smooth edge is degenerated to a narrow ramp edge. The relative error in $\hat{c}$ is high only when too small filter scales σ are used, i.e., when the filter is 'undersampled'. The relative error in the contrast is generally much smaller than the error in the width, for all three methods.

A synthetic edge can be reconstructed once estimates of its contrast and width have been computed. Then, one can compare the original edge $s(x; b, c, w)$ to the reconstructed edge $s(\hat{x}_0; b, \hat{c}, \hat{w})$ on the basis of, e.g., the Normalized Mean Square Error (NMSE) criterion. The simulations showed that the NMSE follows the behavior of the relative error in $\hat{c}$, because the fidelity of the reconstructed edge depends much more on the accuracy of $\hat{c}$ than on the accuracy of $\hat{w}$. It falls below 0.5% when the filter scale $\sigma \geq 1.0$.

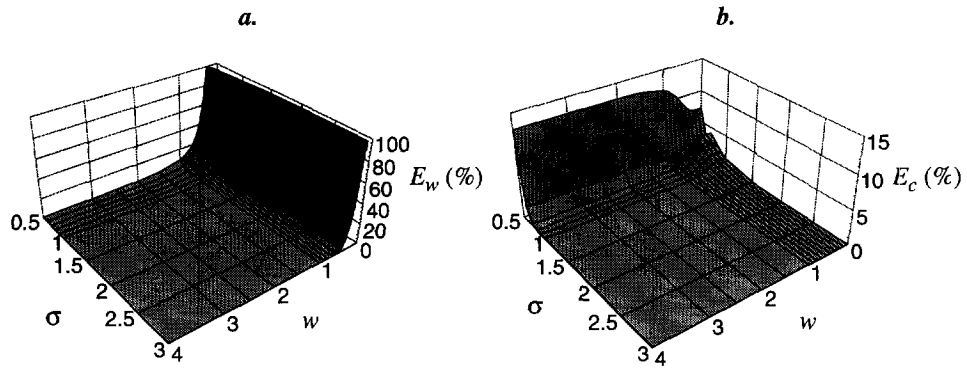


Figure 2.5 Relative error in $\hat{w}$ (a) and relative error in $\hat{c}$ (b) for edges with $c = 10.0$ and with various w using the multi-point method (with $a = 1$) on various scales σ.

2.6.2 The influence of noise

The following experiment was performed to get a better quantitative idea of errors caused by noise. Samples drawn from a zero mean Gaussian noise distribution (with variance between 1.0 and 64.0) were added to the sampled edge profiles (preceding filtering and edge analysis). Each experiment was repeated ten times with the same input parameters, but each with a different noise realization. Then, the mean value and standard deviation of the estimates of each parameter were computed over the ten samples thus obtained, indicating the bias and spread in the estimates. Under noisy conditions, parameter estimates are influenced by both localization errors - detecting the wrong or a dislocated peak - and errors in the peak values.

Localization

Edge localization precedes parameter estimation and is identical for all three methods. The standard deviation of location estimates ($sd(\hat{x}_0)$) of edges hampered by noise with power 4.0 is given in Table 2.1 for various combinations of edge widths and filter scales. Note that the *subpixel* location estimate is not taken into account here. In general, the location standard deviation declines from left to right, i.e., for a particular edge width w, the location error is high if the filter scale σ is too low (cf. Eq. (2.22) and Figure 2.2b). These, and other, simulations show that the location error is quite small if $\sigma \geq w$, even under moderately strong noise conditions ($c = 50.0$ and $\sigma_n^2 = 64.0$).

Parameter estimation

Now, some results on the estimation of contrast and width parameters using the multi-point method are presented. In Table 2.2, the means and standard deviations in width estimates (respectively $m(\hat{w})$ and $sd(\hat{w})$) over ten simulations are given. In Table 2.3, the means and standard deviations in contrast estimates (respectively $m(\hat{c})$ and $sd(\hat{c})$) are given. These experiments were performed with $a = 1.0$ and $\sigma_n^2 = 4.0$. These tables show that the accuracy

of this method is quite satisfactory under low noise conditions, except for very broad edges that are analyzed by filters with small σ. To avoid bias in the parameter estimates, one should choose σ approximately equal to or somewhat higher than w. As expected, using larger filter scales results in better noise smoothing, thus smaller spreads in the estimates (specifically, in the estimates of the contrast; the data concerning the width seem inconclusive concerning the precision).

Obviously, the performance decreases with increasing power of the noise. On the whole though, it holds that at least the contrast parameter can be computed sufficiently accurately for a broad range of differently sized edges, provided the filter scales are large enough, even in the presence of moderate amounts of noise. From other experimental results, not shown here, one can see that the multi-scale and multi-derivative method perform similarly to the multi-point method.

Table 2.1 Location standard deviation ($sd(\hat{x}_0)$) for noisy edges with $c = 50.0$, $b = 100.0$, $\sigma_n^2 = 4.0$, various w and σ.

	σ				
w	0.7	1.0	2.0	3.0	4.0
1.0	0.0	0.0	0.0	0.0	0.0
1.5	0.57	0.0	0.0	0.0	0.0
2.0	0.52	0.57	0.0	0.0	0.0
2.5	0.74	0.99	0.32	0.32	0.0
3.0	1.17	0.67	0.47	0.0	0.0
4.0	1.93	1.81	0.97	0.32	0.0

Table 2.2 Computing w with multi-point method (means and standard deviations), for noisy edges with $c = 50.0$, $b = 100.0$, $\sigma_n^2 = 4.0$, $a = 1.0$, various w and σ.

σ	1.0		2.0		3.0		4.0		6.0	
w	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$
0.3	0.45	0.15	0.34	0.31	0.26	0.31	0.42	0.29	0.52	0.55
1.0	0.99	0.17	1.01	0.13	0.96	0.14	0.95	0.35	0.88	0.57
1.5	1.59	0.35	1.46	0.22	1.53	0.18	1.47	0.11	1.50	0.45
2.0	1.88	0.52	2.00	0.11	2.02	0.24	2.11	0.23	1.96	0.25
3.0	1.80	0.85	2.78	0.38	3.01	0.38	3.04	0.18	2.98	0.28
4.0	2.05	1.56	3.15	0.51	4.01	0.47	3.83	0.30	4.08	0.26

Table 2.3 Computing c with multi-point method (means and standard deviations), for noisy edges with $c = 50.0$, $b = 100.0$, $\sigma_n^2 = 4.0$, $a = 1.0$, various w and σ.

σ	1.0		2.0		3.0		4.0		6.0	
w	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$
0.3	50.82	1.75	49.87	1.25	50.0	1.22	50.21	1.0	50.31	0.71
1.0	50.08	3.17	50.08	1.61	50.25	1.11	49.64	0.94	49.81	0.73
1.5	51.61	5.66	50.03	1.75	49.93	1.77	50.25	0.67	49.87	0.62
2.0	48.34	7.76	49.98	2.20	50.07	1.39	50.36	1.04	49.68	0.75
3.0	36.05	11.42	48.47	4.26	50.68	2.04	49.82	0.98	49.72	0.75
4.0	30.78	17.06	42.65	5.50	50.48	2.99	49.18	1.43	50.09	0.72

2.6.3 The influence of nearby edges

In the following experiment, we study the accuracy of computed contrast and width values in the presence of another, nearby, edge. Sampled versions of a pulse edge model were formed, consisting of one rising edge at $x = x_1 = 0$ and one falling edge at $x = x_2$, with various values of x_2. After filtering with a Gaussian derivative filter, the left peak was detected to be the center of the rising edge. The location error due to the other edge is not discussed here; the simulations behaved as the error analysis from the previous section predicted.

Values of contrast and width of a pulse edge with $x_2 = 8.0$ and various w, computed using the multi-point method using various σ, are shown in Table 2.4. As expected, the estimates $\hat{w}$ and $\hat{c}$ are accurate when both σ and w are small. For increasing values of σ and w, the accuracy in $\hat{c}$ and especially in $\hat{w}$ deteriorates. Other simulations confirm that the bounds on the filter scales gradually become less restricted for larger values of x_2. Conversely, the accuracy in $\hat{c}$ and $\hat{w}$ deteriorates quite fast for smaller values of x_2. Other simulations also show that the multi-scale and multi-derivative method perform similarly to the multi-point method.

Table 2.4 Computing w and c with multi-point method, for a pulse edge with various $w = w_1 = w_2$; $c = c_1 = c_2 = 50.0$, $b = 100.0$, $x_1 = 0.0$ and $x_2 = 8.0$, $a = 1.0$.

σ	1.0		1.5		2.0		2.5		3.0	
w	$\hat{w}$	$\hat{c}$	$\hat{w}$	$\hat{c}$	$\hat{w}$	$\hat{c}$	$\hat{w}$	$\hat{c}$	$\hat{w}$	$\hat{c}$
0.3	0.49	50.70	0.44	50.14	0.38	49.79	0.00	48.99	0.00	48.02
0.7	0.71	50.15	0.70	50.00	0.65	49.60	0.00	47.73	0.00	47.02
1.0	1.00	50.02	1.00	49.96	0.93	49.27	0.46	46.67	0.00	45.50
1.5	1.50	49.96	1.48	49.59	1.35	48.03	0.92	44.71	0.00	42.12
2.0	1.97	49.27	1.89	48.03	1.69	45.52	1.23	41.84	1.15	41.32

2.6.4 Discussion

The experimental results provide further quantitative insight into the accuracy of the discussed parameter estimation methods. On the whole, the results are as expected from the theoretical error analyses of Section 2.5. However, the predicted differences in performance between the three methods turned out to be insignificant (under the restricted conditions simulated here).

Estimation of the contrast and width of a model edge using either of the three methods proposed can be performed sufficiently accurately, providing the filter scales meet certain requirements.

- For all three methods, the discrete approximation to the continuous model holds when the filter scale is large enough: $\sigma \geq 1.0$ is sufficient in general.
- For all three methods, the sensitivity to the presence of white Gaussian noise in the signal depends on the size of the filter with respect to the width of the edge. The location error is kept small if $\sigma \geq w$ holds approximately. Estimation of the contrast and width parameters can be performed sufficiently accurately even under moderately strong noise conditions ($c = 50.0$, $\sigma_n^2 = 25.0$), provided $\sigma \geq w$ holds approximately. The performance can roughly be predicted by Eq. (2.22), Eq. (2.23) and Eq. (2.25).
- For all three methods, the sensitivity to the presence of a nearby edge depends on the size of the filter with respect to the distance to the nearby edge. The location error increases as the filter scale σ increases. The upper bound on the filter scale, so that the location error is kept below a pre-specified limit, is predicted correctly by Eq. (2.27). The value of $\sqrt{w^2 + \sigma^2}$ should at least lie below approximately 2/5th of the distance between two edges with equal contrast in order to be able to compute meaningful contrast and width values, as predicted by Eq. (2.28).

In general imagery, the optimal filter scale σ is difficult to choose a priori. Choosing $\sigma \approx w$ is probably the best compromise, but w is unknown and may vary.

In well-acquisitioned natural imagery (in sharp focus, properly sampled and low noise), the majority of edge points has width w between approximately 0.5 and 1.5. It can be seen from the previous tables that the effect of noise (with low to moderate power) on the estimation accuracy of these edges is quite small, for a wide range of filter scale values σ. At the same time, the effect of a nearby edge can be seen to be quite strong for increasing values of σ. In real imagery, it is quite common to find edges lying near each other. Therefore, one may consider it more important to keep the filter scale σ low, in order to avoid the detrimental effect of nearby edges, than to insist on larger values of the filter scale, in order to suppress the effect of noise. Thus, setting $\sigma = 1.0$ seems a reasonable choice in those cases.

For all three methods, the contrast parameter can be computed with greater accuracy and precision than the width parameter, even in the presence of noise or if another edge is present nearby. This is very convenient in our application, since the fidelity of a reconstructed edge (e.g. in terms of the NMSE) largely follows the fidelity of the estimated contrast $\hat{c}$.

2.7 Edge analysis in 2-D

In this section, we extend the 1-D analysis from the previous sections to the 2-D case. First, we define a 2-D edge, and show how its 2-D filter response is related to the 1-D response discussed earlier. Then, the implementation of the edge detection step and the parameter computation step on a 2-D discrete grid is discussed. This section finishes by reporting some experiments with 2-D edge curves.

All the steps taken that have to do with edge detection, are similar to parts of the edge detection scheme as proposed by Canny. To compute edge parameters, we use the multi-point method. The experimental results showed that there is no significant difference between the three estimation methods as far as accuracy is concerned. At the same time, the multi-point method is the most favorable method if computational complexity and reliability in the 2-D case are taken into account (see Section 2.4.4). As in the 1-D case, computation of the contrast and width can be done entirely "on the fly" during edge detection, with little extra effort. Therefore, only the multi-point method is discussed in the 2-D case and this method is the method of choice in practice, i.e. with real images.

2.7.1 2-D edge model and 2-D edge analysis method

A definition of a 2-D edge can be given with the aid of the 1-D edge (see Eq. (2.4)) as follows:

$$s_{2D}(x, y; b, c, w, \theta) = s_{1D}(x\cos\theta + y\sin\theta; b, c, w) ,$$

assuming the locus of the edge center is a straight line going through the origin at an angle θ with respect to the y-axis. The *edge direction* is perpendicular to that line and passes the origin at an angle θ with respect to the x-axis. The edge direction θ is an extra model-parameter when compared to the 1-D case. Now, the 2-D model edge $s_{2D}(x,y)$ is given by:

$$s_{2D}(x, y) \equiv s_{2D}(x, y; b, c, w, \theta) = b + \frac{c}{2}\left(1 + \mathrm{erf}\left(\frac{x\cos\theta + y\sin\theta}{w\sqrt{2}}\right)\right) . \qquad (2.29)$$

To detect an edge in two dimensions, one uses the *gradient* of the Gaussian smoothed signal. Thus, a 2-D signal is filtered twice: once with the directional derivative in the x-direction of a 2-D Gaussian $g_{2D}(x,y;\sigma)$ and once with the directional derivative in the y-direction of a 2-D Gaussian. The 2-D Gaussian is given by:

$$g_{2D}(x, y;\sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) .$$

The two filtering steps of a 2-D model edge by Gaussian derivatives are as follows:

$$d_x(x, y; c, w, \sigma, \theta) = s_{2D}(x, y; b, c, w, \theta) * \frac{\partial}{\partial x}(g_{2D}(x, y;\sigma)) ,$$

$$d_y(x, y; c, w, \sigma, \theta) = s_{2D}(x, y; b, c, w, \theta) * \frac{\partial}{\partial y}(g_{2D}(x, y;\sigma)) .$$

Both (2-D) filtering steps can be performed separably. Using the fact that $s_{2D}(x,y)$ (as given by Eq. (2.29)) can also be written as the convolution of a 2-D step function and a 2-D Gaussian, and by using [45], it can be shown that:

$$\begin{aligned} d_x(x, y;c, w, \sigma, \theta) &= \cos\theta \cdot d_{1D}(x\cos\theta + y\sin\theta;c, w, \sigma) \\ d_y(x, y;c, w, \sigma, \theta) &= \sin\theta \cdot d_{1D}(x\cos\theta + y\sin\theta;c, w, \sigma) \end{aligned} \tag{2.30}$$

where $d_{1D}(x; c, w, \sigma)$ is the 1-D response given by Eq. (2.7).

These filter responses together form the *Gaussian smoothed gradient* of the edge model signal, as follows:

$$\mathbf{d}_{2D}(x, y;c, w, \sigma, \theta) = d_x(x, y;c, w, \sigma, \theta)\,\mathbf{i}_x + d_y(x, y;c, w, \sigma, \theta)\,\mathbf{i}_y \tag{2.31}$$

where $\mathbf{i}_x$ is the unit vector in the x-direction and $\mathbf{i}_y$ is the unit vector in the y-direction.

It follows straightforwardly from Eq. (2.30) and Eq. (2.31) that the magnitude of the Gaussian smoothed gradient of the 2-D model can be mapped to the magnitude of the 1-D model response by:

$$\left\|\mathbf{d}_{2D}(x, y;c, w, \sigma, \theta)\right\| = \left|d_{1D}(x\cos\theta + y\sin\theta;c, w, \sigma)\right| . \tag{2.32}$$

Furthermore, the direction of the smoothed gradient is equal to the direction of the edge:

$$\operatorname{atan}\left(\frac{d_y(x, y;c, w, \sigma, \theta)}{d_x(x, y;c, w, \sigma, \theta)}\right) = \theta . \tag{2.33}$$

A candidate edge point is detected if the magnitude of the smoothed gradient $\mathbf{d}_{2D}(x, y; c, w, \sigma, \theta)$ reaches a local maximum in the direction θ, at that point. A peak in the first derivative in the gradient direction is identical to a zero-crossing in the second derivative in the gradient direction.

Now, assuming the locus of the edge center is a straight line, one can use the smoothed gradient $\mathbf{d}_{2D}(x,y; c, w, \sigma, \theta)$ to estimate the unknown edge model-parameters c, w and x_e using the multi-point model, exactly as one would do in 1-D. Eq. (2.32) can be used to convert the *magnitude* of this 2-D filter response to the *magnitude* of the 1-D response. The *sign* of the edge contrast in 1-D is replaced by the edge *direction* in 2-D.

2.7.2 Implementation

In practice, of course, one works on a discrete grid. The geometry for the situation on a discrete grid is displayed in Figure 2.6, which shows a piece of the sampling grid containing the 3x3 pixel neighborhood around the point $(x,y) = (0,0)$. Here, we have dropped the dependency on c, w, σ or θ in the notation of the various responses. The magnitude of $\mathbf{d}_{2D}(x,y)$ at each of these grid points can be computed with the known filter response values $d_x(x,y)$ and $d_y(x,y)$. Of course, each of these responses is actually approximated by discrete filtering by summation. In the filtering in 2-D, we generally use sampled Gaussian filters truncated at

$\Delta = 5\sigma$. To check for a local maximum at the point (0,0), the magnitude of $\mathbf{d}_{2D}(x,y)$ is approximated by linear interpolation in two points along the edge direction, using four grid points. In the case shown in Figure 2.6, one has:

$$\left\|\mathbf{d}_{2D}(1,q)\right\| = q \cdot \left\|\mathbf{d}_{2D}(1,1)\right\| + (1-q) \cdot \left\|\mathbf{d}_{2D}(1,0)\right\| ,$$

$$\left\|\mathbf{d}_{2D}(-1,-q)\right\| = q \cdot \left\|\mathbf{d}_{2D}(-1,-1)\right\| + (1-q) \cdot \left\|\mathbf{d}_{2D}(-1,0)\right\| ,$$

with

$$q = \frac{|d_y(0,0)|}{|d_x(0,0)|} . \tag{2.34}$$

Note that the edge direction θ determines which four points in the 3x3 neighborhood are used to interpolate between, and how q is computed (q is the inverse of Eq. (2.34) when $|d_y(0,0)| > |d_x(0,0)|$).

A candidate edge pixel is present at (0,0) if

$$\left(\left\|\mathbf{d}_{2D}(1,q)\right\| < \left\|\mathbf{d}_{2D}(0,0)\right\|\right) \wedge \left(\left\|\mathbf{d}_{2D}(-1,-q)\right\| < \left\|\mathbf{d}_{2D}(0,0)\right\|\right) .$$

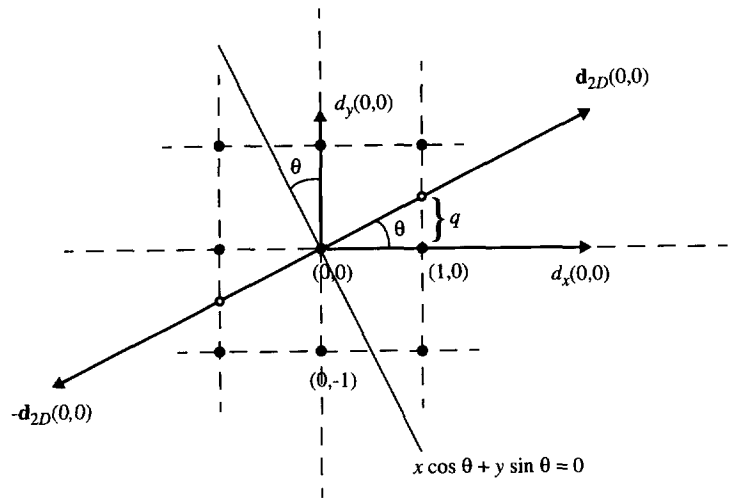


Figure 2.6 Edge detection in 2-D; $\mathbf{d}_{2D}(x,y)$ is the smoothed gradient of the image, formed by smoothed directional derivatives $d_x(x,y)$ and $d_y(x,y)$; the gradient is perpendicular to the straight line $x\cos\theta + y\sin\theta = 0$, which delineates the locus of the edge center.

Now, the problem is converted into a 1-D problem again and edge parameter computation with the multi-point method can be applied by defining d_1, d_2 and d_3 as follows:

$$
\begin{aligned}
d_1 &\equiv \left\|\mathbf{d}_{2D}(0,0)\right\| = \left|d_{1D}(0)\right| \\
d_2 &\equiv \left\|\mathbf{d}_{2D}(1,q)\right\| = \left|d_{1D}(1\cos\theta + q\sin\theta)\right| = \left|d_{1D}(\sqrt{1+q^2})\right| \\
d_3 &\equiv \left\|\mathbf{d}_{2D}(-1,-q)\right\| = \left|d_{1D}(-1\cos\theta - q\sin\theta)\right| = \left|d_{1D}(-\sqrt{1+q^2})\right|
\end{aligned}
\tag{2.35}
$$

where the right sides can be found with Eq. (2.32) and Eq. (2.34). This corresponds to the 1-D situation given by Eq. (2.16) with $a = \sqrt{1+q^2}$. The unknown contrast, width and location shift can be computed with Eq. (2.17) - Eq. (2.19). We have used only the magnitude of the response in Eq. (2.35), the edge direction θ is given by Eq. (2.33).

The intensity value at the edge center m, which is related to the base value and contrast value by $m = b + c/2$, is obtained from the original image, smoothed with a 2-D Gaussian filter $g_{2D}(x,y;\sigma)$, where the scale parameter σ has the same value as used above. Gaussian filtering leaves the edge center intensity value unchanged but suppresses any noise and lowers the gradient magnitude value at the edge center, which makes the extraction of m more robust. The actual edge center does not have to lie exactly at a grid point location; it can lie somewhere *between* grid points. Therefore, the intensity at the edge center is computed by bilinear interpolation between the four surrounding sampling points in the smoothed intensity image. The edge center associated with a particular pixel lies somewhere along the gradient at that pixel, with an offset of x_0 from the central pixel location. This value is computed in the multi-point method.

In most images, edge pixels detected in a local manner as outlined above tend to form connected *curves*. These curves are not necessarily closed; connected curves which are closed are called *contours*. We use 8-connectivity as a criterion for connectedness.

As proposed by Canny, we use a hysteresis thresholding step to select strong edge points, which makes use of the connectivity of edge points. In hysteresis thresholding, one uses one *high* threshold T_h and a second *low* threshold T_l. All candidate edge points (gradient maxima) with a gradient magnitude above T_h are categorized as definite edge points immediately. Further, a candidate edge point with a gradient magnitude *below* T_h but *above* T_l is categorized as an edge point too, *if* it is somehow 8-connected to an edge point with gradient magnitude above T_h (i.e. directly connected or via a connected path of such edge points). This kind of thresholding avoids streaking of edge contours and has been reported to work quite well on natural images. We vary only the high threshold as an instrument for edge selection in our experiments and invariably set $T_l = 0.5\ T_h$. Further, the value of T_h is determined by specifying a value for t_s, the *fraction* of pixels with gradient magnitude lower than T_h. We also apply a thinning step to the thresholded edge points, such that the edge curves are all one pixel thick [73]. Finally, we use a threshold ξ to indicate the minimum length of an edge curve.

2.7.3 Experiments

The analysis above holds only if the edge curve is locally straight at the point being analyzed. In reality, edge curves can display features which are inherently two-dimensional and violate the straightness assumption, such as nonzero curvature and sharp corners. Edge localization in the presence of curved edges and corners has been studied extensively by [15] and [122] for

the *second derivative in the gradient direction* operator (SDGD) and the *laplace* operator. The location of a zero-crossing of the second derivative in the gradient direction of a 2-D edge with nonzero curvature is shifted with respect to the true edge location. Here, we concentrate on the effect of nonzero curvature on the estimated values of the contrast and width.

In this experiment, we have defined a circular object with radius R centered around (0,0). The border of this object forms a curve with constant curvature $1/R$; the edge points on this curve have contrast c and width w everywhere. Thus, the object edge s_{2D} is given by:

$$s_{2D}(x, y; b, c, w, R) = b + \frac{c}{2}\left(1 + \mathrm{erf}\left(\frac{R - \sqrt{(x^2 + y^2)}}{w\sqrt{2}}\right)\right). \tag{2.36}$$

This model is only valid if w is small with respect to R. Using the above-described method, the contrast and width along the circular edge curve were estimated for different values of R and on different scales σ. Due to the discretization, these estimates, $\hat{c}$ and $\hat{w}$, do not have exactly the same value in every point on the curve. Therefore, both the mean value and the standard deviation of these estimates were computed. Note, however, that the errors in these estimates are of a *systematic* nature, rather than of a *stochastic* nature. The error depends on R, w, σ and on the local edge direction.

The results for an edge with $c = 50.0$, $b = 100.0$ and $w = 2.0$ are shown in Table 2.5 and Table 2.6. The circumference of the edge curve and therefore the number of grid points forming the curve in a discrete signal also varies with R and with σ, hence the range of this number for each radius is mentioned in the tables. The radius (R) decreases from left to right, therefore the curvature of the circle ($1/R$) increases from left to right. As expected, the performance decreases for increasing curvatures and increasing filter scales. Specifically, the *average error* in the estimates increases when R becomes very small, e.g. for $R = 5.0$. Further, the *spread* in the estimates increases for increasing σ. When σ is increased above the value 4.0, the model is no longer valid and the values of $\hat{w}$ become meaningless. Note that for $\sigma = 1.0$ the estimates remain accurate even on a circle of very small radius (at least, until $R = 5.0$). In practice, even sharper curvatures and corners may occur.

Table 2.5 Results estimating c in edge points of a circular curve in 2-D using the multi-point method; c = 50.0, b = 100.0, w = 2.0. The number of points on the edge curve is n.

$R=$	80.0		40.0		20.0		10.0		5.0	
$n\in$	[452-453]		[221-225]		[108-113]		[52-57]		[25-37]	
σ	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$	$m(\hat{c})$	$sd(\hat{c})$
1.0	49.19	0.76	49.19	0.69	49.53	0.65	49.12	0.98	50.05	0.76
2.0	49.84	0.91	49.70	0.62	49.98	0.86	49.95	0.84	45.98	0.87
3.0	49.73	1.17	49.65	0.74	49.71	1.05	49.41	1.22	36.54	0.66
4.0	49.37	1.66	49.33	1.25	49.67	2.09	47.32	1.42	-	-

Table 2.6 Results estimating w in edge points of a circular curve in 2-D using the multi-point method; c = 50.0, b = 100.0, w = 2.0. The number of points on the edge curve is n.

$R=$	80.0		40.0		20.0		10.0		5.0	
$n\in$	[452-453]		[221-225]		[108-113]		[52-57]		[25-37]	
σ	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$	$m(\hat{w})$	$sd(\hat{w})$
1.0	1.95	0.06	1.95	0.05	1.97	0.05	1.97	0.07	2.05	0.05
2.0	1.98	0.08	1.97	0.06	2.01	0.08	2.06	0.07	1.98	0.08
3.0	1.95	0.15	1.95	0.11	1.98	0.15	2.13	0.18	1.21	0.18
4.0	1.83	0.37	1.86	0.29	2.00	0.44	2.02	0.32	-	-

2.8 Edge-based image representation

This section outlines how our edge-based image representation is built up using the results from the edge detection and analysis stage. It also explains how to reconstruct a partial intensity surface on the basis of this representation. Finally, some examples are included which illustrate some of the main steps taken by the analysis and reconstruction algorithms.

2.8.1 Format definition

The edge-based image representation is built up on the basis of connected *curves*, defined by the set of detected edge points and by the use of 8-connectivity. These connected curves can be either closed or open, in the latter case the curves end in *end points*. Also, curves can join or split in *branch points*. Points on closed curves and points on open curves between endpoints or branch points are called *link points*. Each of these link points is connected to exactly two neighboring edge points, through *links*. Each link is coded by a *Freeman chain code* [41][59]. There are eight different chain codes for eight different directions, as shown in Figure 2.7.

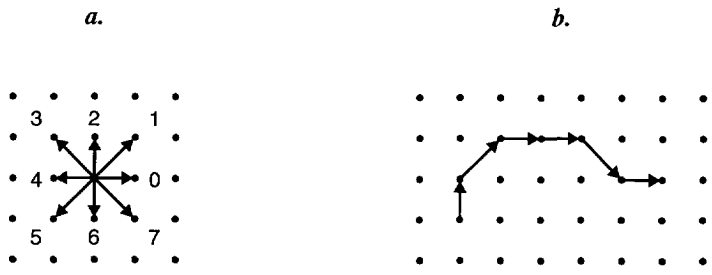


Figure 2.7 (a) Chain coding: the eight different links defined for eight directions, coded 0 to 7; (b) An example of a piece of a curve; the sequence of chain codes is 210070.

A *chain* or *sequence* of links can now be stored in a one-dimensional array-type data structure. Including the coordinates of its starting point and its length, the sequence of links forms a geometric description of an *edge curve*. The location of an edge point is coded by the combination of the edge curve starting point and the chain of links. Although a subpixel edge location shift is computed for each edge point by the multi-point method, this parameter is not stored in our representation for reasons of compactness. Thus, edge point locations are represented to pixel accuracy only.

For each edge point along a curve, the contrast, the width, the center intensity and the local orientation is computed. Arrays with edge model-parameter values of each edge point can be added to the description of each edge curve. An edge curve supplemented by photometric edge parameter data is called an *edge primitive*.

One can further economize on the representation by leaving out the local orientation data. The local orientation of an edge point is already represented implicitly by the shape of the edge curve at that point, because the edge orientation is always perpendicular to the tangent of the curve at that point. Instead, we now have to give the edge contrast a *sign* again because of the two-fold ambiguity that remains in the edge orientation once the tangent is given. This sign is determined by both the direction of the tangent along the curve and the local edge direction given by the smoothed gradient (e.g. using a right-hand rule). Of course, this tangent is only known approximately from the discrete chain of links stored in our representation, therefore, the local orientation estimated from this data is always inexact. The above-mentioned sign is assigned to the c parameter.

The complete image representation consists of the set of edge primitives. An edge primitive or edge curve is denoted by C_i, where i lies between 1 and M, the number of such edge curves. The n-th edge point on edge curve i is denoted by $E_{i,n}$, with $0 \le n \le N_i - 1$. N_i is the number of edge points on curve i. The sequence of links for edge curve i is denoted by $l_i(n)$, with $1 \le n \le N_i - 1$ for an open curve. In the same fashion, sequences with parameter values along edge curve i are denoted by $m_i(n)$, $c_i(n)$ and $w_i(n)$, with $0 \le n \le N_i - 1$, corresponding to the center intensity, contrast and width respectively. The set of edge primitives can be stored in a linked-list data structure. This data structure can be processed very efficiently, e.g., smoothing of edge parameters along edge curves or edge sharpening can be performed quite easily.

2.8.2 Edge intensities reconstruction

When one wants to reconstruct an image on the basis of our edge-based image representation, one must start by reconstructing the intensities of the pixels in the image making up the edges. With the edge information extracted from an image as described in Section 2.7.2 and stored in linear data structures as described in Section 2.8.1, one has a description of the variations in the image intensity surface along edge curves. This description of the intensity variations holds only in a narrow strip along each edge curve. Thus, this allows only for a partial reconstruction of the intensity surface. Reconstruction of the *complete* surface is described in Chapter 3 of this thesis.

The edge intensities reconstruction scheme is straightforward and is illustrated in Figure 2.8. Each pixel on the grid is considered in turn and given an intensity value if necessary. For each

pixel (x,y) in the image, one first determines the location (x_e,y_e) of the nearest edge pixel, by using a *Vector Distance Transform* [21][33]. The Vector Distance Transform can compute very efficiently, for each non-edge pixel on a grid, the vector to the nearest edge pixel. Here, this vector is given by $(x_e\text{-}x, y_e\text{-}y)^T$ and is pointed to edge pixel $E_{i,j}$ on curve C_i. Having determined the nearest edge pixel, the computed model-parameters $c_i(j)$, $w_i(j)$, $m_i(j)$ associated with this edge pixel can be retrieved from the data structure immediately. Then it is determined whether the distance to the edge pixel (given by the length of the vector) is within a certain maximum, i.e., whether it lies within a strip along the edge curve. This maximum depends on the local width of the edge at (x_e,y_e), because the strip needs to be wider for edge curves made up of edges with larger w. Thus, the test is as follows:

$$\left\| (x_e - x, y_e - y)^T \right\| \leq f_w \cdot \hat{w}$$

with f_w a constant factor and $\hat{w} = w_i(j)$, the width parameter value of $E_{i,j}$. The constant f_w can be chosen such that the intensity variation perpendicular to the curve (i.e. as given by the erf-shaped model) is reconstructed almost completely, i.e., until the error falls below a small number. We have used $f_w = 2.0$ in all our experiments, resulting in an error in the contrast of at most 2.5%.

The local tangent direction in (x_e,y_e) is approximated using the ingoing and outgoing (chain coded) links on the curve, $l_i(j)$ and $l_i(j+1)$, as shown in Figure 2.8. With this tangent direction and with local sign information, the cosine and sine of the edge direction θ can be computed. Finally, an intensity value is inserted at the pixel (x,y) with the appropriate model value

$$s_{1D}((x - x_e)\cos\hat{\theta} + (y - y_e)\sin\hat{\theta}; \hat{b}, \hat{c}, \hat{w}).$$

Here, $\hat{b} = m_i(j) - \frac{1}{2}c_i(j)$, $\hat{c} = c_i(j)$ and $\hat{w} = w_i(j)$.

The algorithm, including the Vector Distance Transform, does not need any search steps and is therefore fast (constant per pixel time).

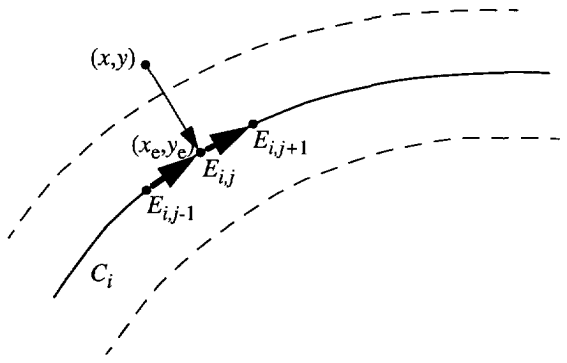


Figure 2.8 Edge intensities reconstruction in the point (x,y). The nearest edge point $E_{i,j}$ at (x_e,y_e) is used to extract information about the local intensity surface.

2.8.3 Examples

Some of the main steps taken by the algorithm that extracts edge information from a 2-D image and the algorithm that reconstructs the intensities of edges are illustrated in Figure 2.9, Figure 2.10 and Figure 2.11.

Figure 2.9(a) shows a synthetic 50x50 grey-level image, picturing a circular object with a radius of 15 pixels. The edge contrast is 75.0, the edge width is 1.0 and the background value is 100.0. The image in Figure 2.9(b) shows the edges detected in the image after filtering with Gaussian derivatives ($\sigma = 1.0$), nonmaximum suppression, hysteresis thresholding and thinning. The values of c, w and m in each edge point, computed using the multi-point method, are not shown. The image in Figure 2.9(c) shows a partial reconstruction of the intensity surface (based on the extracted edge parameters) with $f_w = 2.0$. The Mean Square Error (MSE) of the reconstructed pixel intensities is 13.5. Lastly, the image in Figure 2.9(d) shows the difference image between (a) and (c), where the difference has been set to zero at pixels where no intensity value was reconstructed. Also, this error image has been multiplied by 8 and an offset of 128 has been added afterwards, for visibility. Both white and black points signal large errors, while grey points signal small errors. One notices in (d) that the error is higher in pixels at edges with a local orientation at oblique angles compared to pixels at edges with a local horizontal or vertical orientation. Also, the edges in (c) look somewhat 'jagged'. These effects are probably caused by the fact that the edge pixels are reconstructed using only pixel accuracy for the location of the edge curve (i.e. location parameters are 'quantized' to coincide with grid points only), while the actual edge curve, of course, can lie *between* pixels.

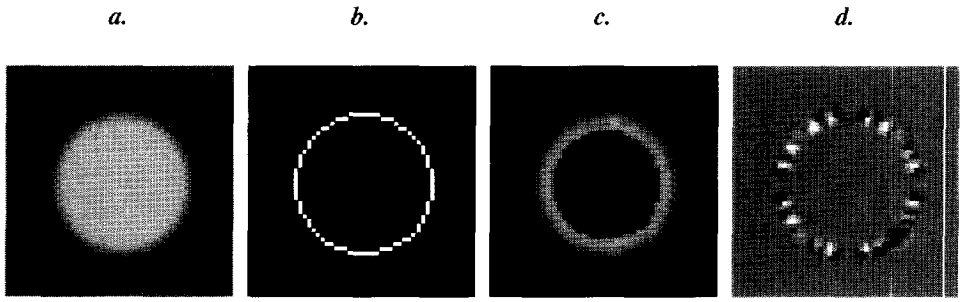


Figure 2.9 (a) Original image; (b) Edges detected after filtering, nonmaximum suppression, hysteresis thresholding and thinning; (c) Edge profiles reconstructed; (d) Error image - with the intensities multiplied by 8 and an offset of 128 added.

Figure 2.10(a) shows a 49x49 real world grey-level image, taken from a larger image. The image in (a) shows part of a light object against a uniform background and displays several types of edge features: three straight edge curves at the lower middle, of which the right two

are close together; two curved edge curves at the upper middle and left; a small elliptical edge curve in the middle; a convex and a concave corner at the right and left. The image in Figure 2.10(b) shows the edges detected in the image after filtering with Gaussian derivatives ($\sigma = 1.0$), non maximum suppression, hysteresis thresholding and thinning. The values of c, w and m in each edge point, computed using the multi-point method, are not shown. The image in Figure 2.10(c) shows a partial reconstruction of the intensity surface (based on the extracted edge parameters) with $f_w = 2.0$. The MSE of the reconstructed pixels is 111.0. Lastly, the image in Figure 2.10(d) shows the scaled difference image between (a) and (c). Again, the error has been set to zero at pixels where no intensity value was reconstructed and the error image has been multiplied by 8 and an offset of 128 has been added afterwards. One can notice in (c) that the width of the strip around edge curves, within which image pixels are reconstructed, varies, because the estimate of w varies. At part of the upper edge curve, the estimated width is probably too large. One can see in (d) that the most significant errors seem to occur exactly at the edge centers. Again, this can be explained by the fact that edge point locations are preserved to pixel accuracy only.

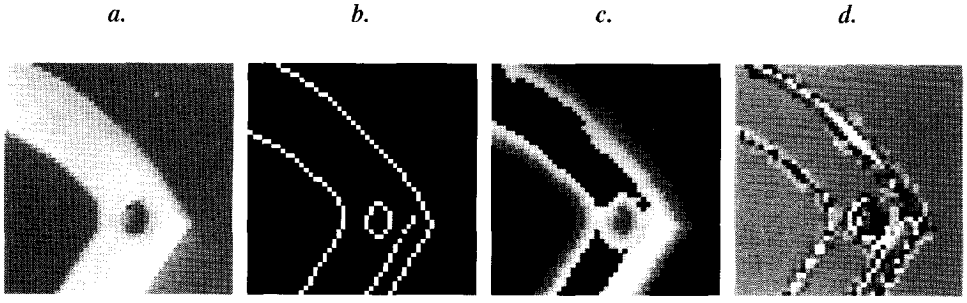


Figure 2.10 (a) Original image; (b) Edges detected after filtering, nonmaximum suppression, hysteresis thresholding and thinning; (c) Edge profiles reconstructed; (d) Error image - with the intensities multiplied by 8 and an offset of 128 added.

Figure 2.11(a) shows the original 256x256 "Scarf" grey-level image. This image contains various kinds of edge features. The image in Figure 2.11(b) shows the edges detected in the image after hysteresis thresholding and thinning. The image in Figure 2.11(c) shows a partial reconstruction of the intensity surface, based on information of the values of c, w and m in each edge point, with $f_w = 2.0$. The MSE of the reconstructed pixels is 49.0. Lastly, the image in Figure 2.11(d) shows the scaled difference image between (a) and (c). Again, the error has been set to zero at pixels where no intensity value was reconstructed and the error image has been multiplied by 8 and an offset of 128 has been added afterwards. In the original image, one can see a clear difference in width or smoothness between the left and right outlines of the face. The reconstruction in (c) clearly shows that our algorithm has recognized this difference.

Further, the reconstruction can be seen to contain most of the relevant image intensity variations. The error image in (d) shows that spots with significant errors occur below the right eye, at the left eyebrow, below the tip of the nose and somewhere in the scarf at the left.

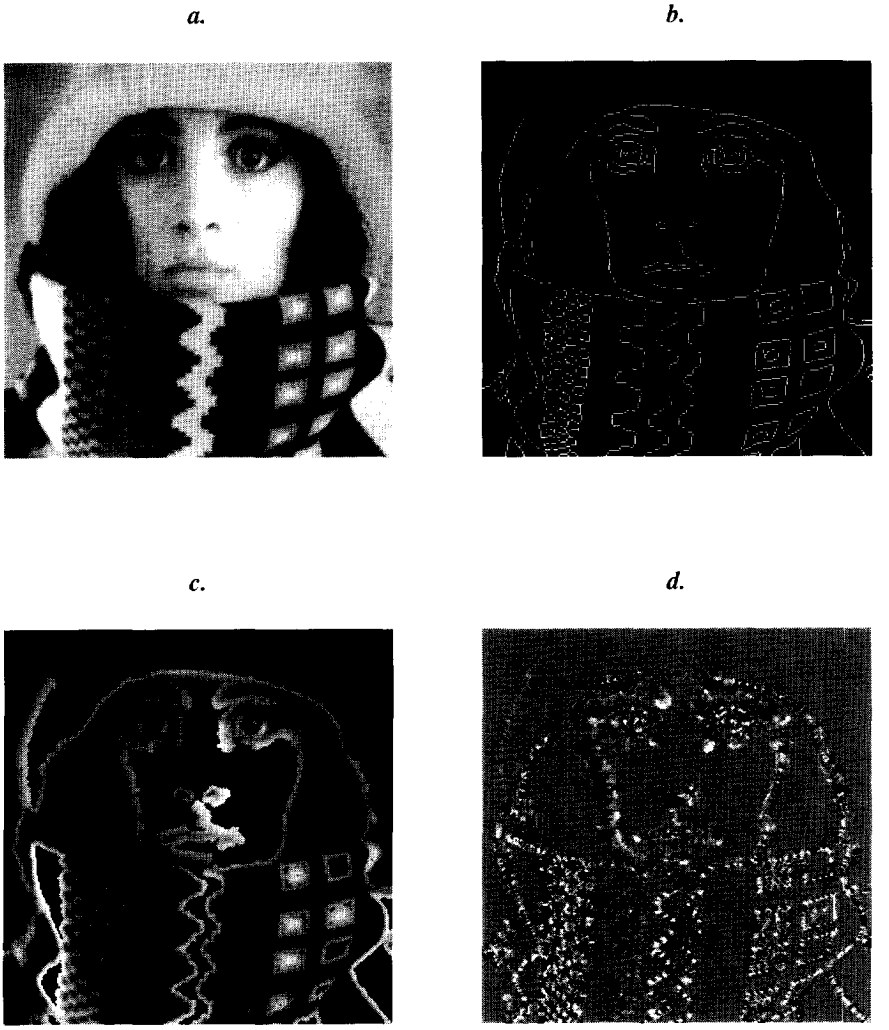


Figure 2.11 (a) Original Scarf image; (b) Edges detected after filtering, nonmaximum suppression, hysteresis thresholding and thinning; (c) Edge profiles reconstructed; (d) Error image - with the intensities multiplied by 8 and an offset of 128 added.

2.9 Conclusions

In this chapter, we have described how to extract a compact description of the structure of an image. Our approach is based on the assumption that the majority of relevant intensity variations in an image is formed by the edges. Using a simple one-dimensional model of an edge allows one to formulate the edge description problem as a parameter estimation problem. The observables in this problem are samples of filtered versions of the original signal. The unknown parameters are the edge contrast and edge width. The filter kernels used are similar or identical to the ones used in edge detection, namely derivatives of Gaussians. Three different methods using different filter responses have been proposed: multi-scale estimation, multi-derivative estimation and multi-point estimation. We have shown how this approach can be extended to the two-dimensional case.

This approach has the following advantages:

- Using an edge model parameterized by only a few parameters allows for a compact representation.
- The edge model-parameters can, in our case, be recovered quite easily from filtered versions of the signal just after the particular feature has been detected. Therefore, the computational complexity of the method is quite low. Once the necessary convolutions have been carried out, only simple, local, calculations are involved.
- The influence of noise is suppressed by the inherent Gaussian smoothing applied in the method. In the multi-point method, which was chosen to be used in 2-D images because of its minimum computational complexity, no derivatives of an order higher than one are used, thereby keeping the noise sensitivity of the scheme to a minimum.

Error analyses and experimentation have been carried out to study when the approach holds and to obtain quantitative information about the effects of model-deviations. The three estimation methods behave similarly with respect to the influence of sampling, noise and nearby edges.

The concept of *scale* (specifically, the analyzing filter scale) has turned out to be important in edge detection and model-parameter computation. It appears that there is only *one* optimal scale for each area in an image that allows robust recovery of the underlying local structure. Setting the (global) filter scale too low in a noisy environment will have a detrimental effect on the location, contrast and width estimate of an edge. Setting the filter scale too high will distort the estimates of location, contrast and width when several edges are positioned close together and when edge curves in 2-D have nonzero curvature. Thus, one must find a scale that is a good compromise between these two conflicting requirements: one has to "focus" on the underlying structure.

Having the filter scale approximately equal to the edge width seems a good compromise in general imagery, but the edge widths are unknown and may vary. Therefore, in images with relevant structure on different scales, one may have to apply a multi-scale analysis to solve the trade-off between resolution and noise resistance. E.g., one could *track* edge curves through the scale space in order to find a good scale for each edge curve.

However, in well-acquisitioned natural imagery (in sharp focus, properly sampled and low noise) most edges are sharp, i.e., have small width. In this case, the problem with noise is much less severe and much less common than the problem with multiple, nearby, edges and with curved edges and corners in 2-D. It would then suffice to use only a single filter scale, fixed at a small value such as 1.0.

Finally, we have described how to reconstruct part of the intensity surface of an image on the basis of the edge-based representation extracted before. Some examples have illustrated the fidelity of these partial reconstructions.

Further research can be directed toward generalizing the approach, e.g., by deriving similar equations (as in Section 2.4) for other 1-D and 2-D feature models. Also, in the 2-D case, the use of more than three points in the estimation and subsequent *over*-constraining the parameters should be considered. The edge intensities reconstruction algorithm can be improved by interpolating the edge curve locus between grid locations.

Chapter 3
Intensity Surface Reconstruction

In Chapter 2, we described how to extract a compact description of the structure of an image and how to reconstruct a part of the intensity surface on the basis of this description. This chapter is concerned with reconstructing the *full* intensity surface (i.e. across the entire image domain) on the basis of a partial reconstruction. The global reconstruction problem is basically a problem of finding an intensity surface which interpolates a set of irregularly placed constraint values and which at the same time agrees with certain prior expectations one has about images. In the first section, the problem is formulated more precisely and an outline of the rest of the chapter is given.

3.1 Introduction

3.1.1 Problem formulation

In Chapter 2, we have discussed the analysis of the edge content of an image and the forming of an image representation on the basis of that analysis. This representation consists of *edge primitives*, which describe the location and shape of *edge curves* in the image, and the intensity variations locally along these curves. These intensity variations are modeled pointwise using a simple edge model. The parameters in this model are the contrast, center intensity and the width of the intensity profile in an edge point; the parameters of all edge points on all edge curves are contained in the representation. The data in this representation in itself describes the intensity surface only in part of the image's spatial domain, namely only locally near each edge curve. Therefore, the remaining parts of the intensity surface must be *reconstructed* in a stable and efficient manner. The reconstructed image should, of course, be approximately equal to the original. This is the problem addressed in this chapter. Obviously, the reconstruction should consist of some sort of smooth interpolation between the already

known pixels of the intensity surface (pixels which are constrained or fixed by the data in our representation). However, straightforward interpolation methods do not apply because the known pixels are not distributed uniformly across the spatial domain. Another problem is caused by the fact that the given edge curves do not necessarily form closed contours, due to the nature of the edge extraction process.

A few varieties of edge-based representations - similar to the representation proposed in this thesis - can be found in the literature. Besides studying reconstruction algorithms, some authors also study the theoretical question of *completeness* of a representation, i.e., whether a representation *uniquely* defines the input image from which it was extracted. Completeness has been conjectured by Marr in the case of the zero crossings of the Laplacian of the signal in a Gaussian scale space (see [53] for mathematical results) and by Mallat in the framework of wavelet theory (the wavelet maxima representation, see [81] and [82]). In practice, a close approximation to the original image may be obtained stably from these kinds of representations if they are rich enough, e.g. [26][53][82].

As mentioned, the edge parameters in our representation do not describe the original intensity surface, denoted by $i(x, y)$, uniquely. Furthermore, these parameters are, in our case, subject to an encoding process, which consists of both reversible (lossless) and irreversible (lossy) operations (consider again Figure 1.2, picturing the role of the reconstruction or synthesis process in the entire coding scheme). So the partial reconstruction along edges, denoted by $g(x, y)$, is made on the basis of a distorted version of the data in the representation. A little more precisely, the problem can be defined as follows, using an abstract operator $H\{.\}$ to denote the mapping from i to g:

$$g = H\{i\} \ . \tag{3.1}$$

The problem is to find a solution i, which obeys Eq. (3.1), given the data in g. In our case, g does not *uniquely* describe i (i.e. $H\{.\}$ is onto but not injective), therefore *many* solutions consistent with Eq. (3.1) exist. Furthermore, the data in g can contain errors (or noise) as mentioned above. This is a nonlinear inverse problem.

Mathematical analysis of our representation is severely complicated by the fact that the mapping $H\{.\}$ includes many serious nonlinearities (thresholding, quantization). We therefore do not attempt to perform such an analysis, but we restrict ourselves to the formulation of a reconstruction algorithm and its evaluation in terms of the quality of the solutions it produces, its efficiency and its stability. The numerical and the perceptual similarity of the reconstructed image to the original are an indication of the quality of our representation and reconstruction scheme. Efficiency in terms of algorithm speed is another important criterion in this chapter.

3.1.2 Outline

Since a unique solution to the problem in Eq. (3.1) is not immediately available, the problem is said to be *ill-posed* [14][118]. A powerful approach to handle ill-posed problems that is able to include certain *a priori knowledge* to restrict the class of admissible solutions is *regularization theory* [118]. The regularization approach is a very general one and has been

used in many related image processing problems, such as image restoration (see [16]) and computer vision (see [14], [97] and [117]).

In Section 3.2, the regularization technique is introduced and its application to the problem at hand is described. The approach leads to a large system of linear equations, the solution of which is computed by an iterative minimization algorithm. In this algorithm, the constraints given by the edge primitives in our representation are iteratively *propagated* across the entire image domain. In Section 3.3, we extend the regularization method and propose a new algorithm, which is aimed at providing a speed-up of the iterative process. The idea behind the algorithm is to allow faster propagation of the constraint information in the edge primitives by more global interactions. The algorithm is theoretically underpinned by the introduction of *multi-scale smoothness models* in the regularization. A convergence analysis is included in Appendix A. Experiments comparing the new method with other methods are reported in Section 3.4. We finish this chapter with conclusions in Section 3.5.

3.2 Regularized image intensity reconstruction

3.2.1 Regularization

A common form of regularization for finding a solution to the problem in Eq. (3.1), given the data g and $H\{.\}$, is to construct the functional $E(.)$

$$E(u) = P(u, g) + \lambda S(u) , \qquad \lambda > 0 ,$$

consisting of a functional $P(u, g)$, which measures the *discrepancy* between possible solutions $u \in U$ (U is the space of admissible solutions) and the data g, and a *stabilizing functional* $S(u)$, which measures the *smoothness* or *stability* of the solution u. The optimal solution $\hat{u}$ is obtained by minimizing $E(u)$:

$$\hat{u} = \arg \min_{u \in U} E(u) . \tag{3.2}$$

The *regularization parameter* λ is used to control the amount of regularization applied and to reach a balance between the tightness of the fit between solution and data (as measured by P) and the smoothness of the solution (as measured by S).

$P(u, g)$, also called the *data compatibility constraint*, is usually based on a norm on the difference between the data and the result of applying $H\{.\}$ to the proposed solution:

$$P(u, g) = \| H\{u\} - g \|^2 .$$

However, the resulting problem cannot be solved in our case since $H\{.\}$ contains many noncontinuous and nonlinear operations, leading to an unfeasible solution. This problem can be circumvented by replacing the complex operator $H\{.\}$ by a *sampling* operator H, which simply extracts from u those samples coinciding with points where g is known (remember that

$g(x, y)$ represents a partial reconstruction of the image, in which the reconstructed points are set to a certain value while the rest are set to zero). Thus,

$$P(u, g) = \|Hu - g\|^2 = \sum_j \alpha_j \left(u(x_j, y_j) - g(x_j, y_j)\right)^2 \tag{3.3}$$

where the constants α_j can be used to weight the data points according to our belief in their *relative* fidelity. The notation on the right side of this equation reflects the fact that a finite set of data points (x_j, y_j) is available with data values $g(x_j, y_j)$.

The stabilizing functional $S(u)$ usually contains integrals of a linear combination of the first p-order derivatives of the candidate solution u. The general stabilizers for univariate regularization on a function $u(x)$ as proposed by Tikhonov [118] were generalized further to multivariate stabilizers on functions $u(\mathbf{x})$ by Terzopoulos [117]. These stabilizers can be viewed as generating multivariate *generalized splines*, smooth surfaces that interpolate or approximate any given data. In the two-dimensional domain, the stabilizing functionals are defined on $u(x, y)$ and can be written as

$$S(u) = \sum_m \iint_{\Re^2} q_m \left(\sum_{j=0}^{m} \binom{m}{j} \left(\frac{\partial^m u}{\partial x^j \partial y^{m-j}} \right)^2 \right) dxdy \ .$$

These functionals are invariant under translation and rotation (see also [23]).

If only $q_1 = 1$ (and the rest are 0) then the stabilizing functional becomes

$$S(u) = \iint_{\Re^2} \left(\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right) dxdy \ , \tag{3.4}$$

which is often called the *membrane* functional, referring to a physical analogue. Solutions to Eq. (3.4) are surfaces which are themselves continuous, but their first partial derivatives need not be continuous.

If only $q_2 = 1$ then the stabilizing functional becomes

$$S(u) = \iint_{\Re^2} \left(\left(\frac{\partial^2 u}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 u}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 u}{\partial y^2} \right)^2 \right) dxdy \tag{3.5}$$

which is often called the *thin plate* functional, referring to another physical analogue. Solutions to Eq. (3.5) are surfaces which are themselves continuous and have continuous first partial derivatives. Eq. (3.5) has become a popular functional for the problem of recovering a scene surface from sparse depth data (see [48]), because it is the natural two-dimensional extension to the cubic spline. Image interpolation experiments with various functionals are reported in [63].

Stabilizing functionals are designed to correspond to our prior expectations of the physical world, which implies having a *prior model* of the image intensity surface, that is used to guide

the reconstruction algorithm. Because the stabilizing functionals in Eq. (3.4) and Eq. (3.5) measure the amount of *variation* in a surface, they act as a kind of *smoothness constraints* on the solution surface. These smoothness constraints lead to the smoothly interpolated surfaces desired.

We follow Carlsson [26] in his use of Eq. (3.4) by using stabilizing functionals on image intensity surfaces based on *first* partial derivatives only. Justification for the use of this stabilizing functional in our problem is the following. Firstly, note that the relevant edge points of the original image are represented in g and that the solution surface u is kept close to g in these edge points (data compatibility constraint). Between these edge points, the use of Eq. (3.4) leads to an interpolated intensity surface which is least likely to contain sharp variations in terms of its gradient magnitude - which is, in fact, the quantity used in the edge analysis process to detect interesting features to be included in our representation (albeit a smoothed version of the gradient). Thus, the use of this particular stabilizing functional is justified in our case because it minimizes the gradient magnitude in points where one knows this quantity should indeed be small. Regularization provides a rigorous mathematical basis for the use of the smoothness constraint following from this notion. Further, the functional in Eq. (3.4) is computationally simple, as shown in the following subsection.

3.2.2 Discretization

Solving Eq. (3.2), with $P(u, g)$ given by Eq. (3.3) and $S(u)$ given by Eq. (3.4), is a variational problem which can be handled in many different ways. One approach is to firstly use the calculus of variations (see [108] and [115]) leading to a partial differential equation giving a necessary condition for the function $u(x, y)$ to minimize the integral $E(u)$.

Another approach, the one used here, is to start by replacing $E(u)$ by a discrete equivalent directly. To this end, u must be discretized, e.g., by using finite differencing to approximate the derivatives in $S(u)$. The resulting functionals $P(u, g)$ and $S(u)$ are given by:

$$P(u,g) = \sum_m \sum_n \alpha(m,n) \; (u(m,n) - g(m,n))^2 \tag{3.6}$$

and

$$S(u) = \sum_m \sum_n (u(m,n) - u(m-1,n))^2 + (u(m,n) - u(m,n-1))^2 \tag{3.7}$$

where $(m, n) \in \aleph^2$ are points on the discrete grid and $\alpha(m, n) = \alpha_j$ in points (x_j, y_j) where a data point is available and $\alpha(m, n) = 0$ otherwise. In matrix/vector notation, these can be written concisely as follows. Firstly, Eq. (3.6) is replaced by

$$P(\mathbf{u}, \mathbf{g}) = (\mathbf{u} - \mathbf{g})^T \mathbf{P} (\mathbf{u} - \mathbf{g}) \; . \tag{3.8}$$

Here, $\mathbf{u}$ and $\mathbf{g}$ are lexicographically ordered vectors containing values of the candidate solution and data point values respectively, and $\mathbf{P}$ is a diagonal matrix with entries $\alpha(m,n)$ along the main diagonal. Similarly, Eq. (3.7) is replaced by

$$S(\mathbf{u}) = \mathbf{u}^T\mathbf{S}\mathbf{u} \tag{3.9}$$

where **S** is an extremely sparse matrix called the *stiffness matrix*. It contains the value 4 along the main diagonal and the value -1 along four other diagonals (*tridiagonal with fringes*). Using Eq. (3.8) and Eq. (3.9), one can write the combined functional in discrete form as follows:

$$E(\mathbf{u}) = P(\mathbf{u}, \mathbf{g}) + \lambda S(\mathbf{u}) = \mathbf{u}^T(\mathbf{P} + \lambda\mathbf{S})\,\mathbf{u} - 2\mathbf{u}^T\mathbf{P}\mathbf{g} + \mathbf{g}^T\mathbf{P}\mathbf{g} .$$

Since $E(\mathbf{u})$ is a nonnegative quadratic form in **u**, a necessary and sufficient condition on **u** at the minimum of $E(\mathbf{u})$ is obtained by differentiating it with respect to **u** and equating to zero. This leads to the following system of linear equations:

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{3.10}$$

with $\mathbf{A} = \mathbf{P} + \lambda\,\mathbf{S}$ and $\mathbf{b} = \mathbf{P}\,\mathbf{g}$. Writing out one row of this system in the ordinary notation, and rearranging, gives more insight:

$$\begin{aligned} &\alpha(m, n)\ \ (u(m, n) - g(m, n)) + \\ &\lambda\,(4u(m, n) - u(m-1, n) - u(m, n-1) - u(m+1, n) - u(m, n+1)) = 0 . \end{aligned} \tag{3.11}$$

When λ is very small, the first term in the equation must be made small as well, so the data compatibility constraint dominates. Conversely, for $\lambda \to \infty$, putting the second term (in the form of a discrete Laplacian operator) to zero becomes more important, so the smoothness constraint dominates. Note that this trade-off is only effective in data points where $\alpha(m,n) > 0$; in other points, the reconstruction is guided by the smoothness constraint only.

From the above expression, one may note that the stabilizing operators used in regularization are related to certain noncausal stochastic image models, discussed by Jain [59]. This was also noted in [16]. Both are strongly related to partial differential equations of the elliptic type.

One may also note from Eq. (3.6) and Eq. (3.7) that, firstly, the stabilizing functional can be thought of as a prior image model and, in Bayesian terms, as being related to a *prior* probability density function on u. Secondly, the data discrepancy functional P can be related to an *a posteriori* probability density function on g, given u. Thus, one can interpret the regularization procedure as Bayesian estimation: the solution provided by regularization theory is, in certain cases, the maximum a posteriori (MAP) estimate of a random field u given the data g (see [117]; see [18] and [111] for arguments on the choice between the deterministic/mechanical approach and the probabilistic approach in computer vision).

3.2.3 Iterative solution

The solution to Eq. (3.10) must be found by an appropriate numerical method for solving large linear systems. Direct methods are not feasible due to the excessive amount of storage needed. *Iterative* methods do not need excessive storage and can make use of the sparseness of **A**. We start out by applying a relaxation method [98]. Relaxation methods are local in nature and readily parallellizable. The method of *Successive Overrelaxation* (SOR) works as follows.

Start out with an initial approximation $u^0(m, n) = g(m, n)$. Then, at each iteration i, the new solution u^{i+1} is computed for all (m, n), starting at $(m, n) = (0, 0)$ and scanning the grid in a row-by-row fashion, as follows. Firstly, a *residual term* $r(m, n)$ is computed,

$$r^{i+1}(m, n) = \alpha(m, n)\Big(u^i(m, n) - g(m, n)\Big) + \lambda\Big(4u^i(m, n) - u^{i+1}(m-1, n) - u^{i+1}(m, n-1) - u^i(m+1, n) - u^i(m, n+1)\Big) \tag{3.12}$$

where updated values of u^{i+1} to the left and above are used immediately when they become available. Then, this term is subtracted from the current solution to form the new solution:

$$u^{i+1}(m, n) = u^i(m, n) - \omega \frac{r^{i+1}(m, n)}{(\alpha(m, n) + 4\lambda)} \tag{3.13}$$

where ω is the *overrelaxation parameter.* When $\omega = 1.0$, the current solution $u(m, n)$ is corrected at each iteration by the scaled residual; when $1.0 < \omega < 2.0$, $u(m, n)$ is *overcorrected* to speed up convergence. Each iteration is performed entirely "in place", i.e., computation of $r(m, n)$ requires only samples of $u(m, n)$ from the previous iteration or of previously processed samples from the current iteration (note the indices in Eq. (3.12)). Again, the value of λ (with respect to the $\alpha(m,n)$) can be seen to control whether the update in a particular point of the current solution lies in the direction of "more smoothness on u" or in the direction of "better fit between u and g". If the fidelity of each of the data point values is equal, it is easiest to set $\alpha(m,n) = 1$ in all points where a data value is available and $\alpha(m,n) = 0$ in all other points. This is what we have done in all our experiments. In such a case, only λ has to be used to control the regularization. If the solution has to fit the data constraints exactly, one might set λ to a very small number compared to the values of $\alpha(m,n)$. A simpler technique to implement these "hard constraints" is to keep the data point values fixed by zeroing out (in each iteration) components of the residual vector at points where $\alpha(m,n) \neq 0$. In this "hard-constraints" algorithm, the value of λ in effect approaches the value 0.0 in the limit.

For $\omega = 1.0$, the SOR method is equivalent to the *Gauss-Seidel* method. An optimal value for ω exists, $1.0 < \omega^* < 2.0$, which, however, cannot be found easily for non-standard problems. The SOR method can be seen to be very similar to the *steepest descent* method, because the residual $r(m, n)$ points in the direction of the negative gradient of $E(u)$. It is also worth mentioning the *conjugate gradient descent* method (see e.g. [98]) which we have used for comparison purposes. This method is also a descent method except that subsequent step directions are conjugate with respect to each other.

An example of the iterative optimization algorithm is given in Figure 3.1. Here, the same grey-level image as that in Figure 2.9, with a light circular object against a darker background, was used to extract the edge parameters. In Figure 3.1(a) the reconstructed edges are shown, which are the input data to the image intensity surface reconstruction algorithm, the SOR method with $\omega = 1.0$. Reconstructed intensity surfaces are shown in (b), (c), (d) and (e), after 20, 40, 80 and 160 iterations respectively. The data in (a) are used as hard constraints, i.e., the reconstructed grey-level surfaces in (b) through (e) all *interpolate* the data in (a). The example

shows the iterative "propagation" or "diffusion" of the constraints in the input data across the entire image, by optimizing the smoothness in each point (except the data points). It can be seen that the resulting image (the image in (e) is close to convergence) is a visually agreeable approximation of the original image (shown in Figure 2.9(a)).

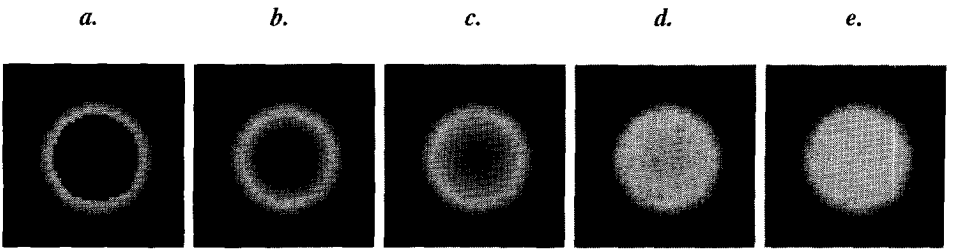


Figure 3.1 Iterative image intensity surface reconstruction on a 50x50 grid. (a) Input data of 50x50 pixels; (b) Result after 20 iterations; (c) Result after 40 iterations; (d) Result after 80 iterations; (e) Result after 160 iterations.

Note that a more intelligent initialization of the iterative process can be made than just setting all the non-constrained grid points to zero. A better initialization can help reduce the total number of iterations necessary to reach the solution. A good choice would be to set the non-constrained grid points to values derived from edge parameters of the nearest edge point, in exactly the same way as the values of constrained pixels (edge pixels) are set (see Section 2.8.2). This way, the entire image would be initialized with grey-level values derived from the edge primitives, as opposed to initializing large parts of the image with zero values, as in Figure 3.1(a). Of course, still only the pixel values within a small band along edge curves would be *constrained* in the global intensity surface reconstruction, other pixel values would *not* be constrained.

3.2.4 Convergence considerations

The above-described SOR method and similar methods (such as Gauss-Seidel or conjugate gradient) consist of local computations such as Eq. (3.12). The global solution evolves by iterative propagation of the constraints given by the data points in g. Convergence of these methods - measured in the number of iterations necessary - is often very slow. Consider for example an image grid of size NxN. Each iteration of such a method costs $O(N^2)$ operations. The SOR method needs approximately N iterations to reach three-figure accuracy for a standard problem, but only if ω is set optimally [98]. The Gauss-Seidel method (keeping ω fixed at 1.0) needs about N^2 iterations to accomplish the same accuracy. The conjugate gradient descent method converges completely within N^2 iterations, in the absence of roundoff errors. The number of iterations required to reach an acceptable solution may be much smaller in our case than in the case of standard problems, because sometimes edge curves more or less enclose regions with a diameter much smaller than N. However, the number of iterations may still be prohibitive when working with reasonably sized images ($N > 200$).

Various methods to speed up the convergence of these algorithms have been proposed in the numerical analysis literature. Many of these approaches are based on the notion that the propagation of information across the solution grid is speeded up if one allows more global interactions between nodes on the grid, instead of the purely local interactions used in the standard approach. Therefore, many methods employ a hierarchical representation of the data. For example, *multi-grid methods* [98] treat the problem at different levels of resolution - low-resolution levels corresponding to *coarsely* sampled grids. At the coarse levels, information is propagated much faster at the cost of lower spatial resolution. Going back down to finer levels regains the fidelity of the solution. Multi-grid methods are claimed to need only $O(N^2)$ *operations* (in total) to converge, for certain model boundary value problems. Multi-grid relaxation methods were applied successfully to a number of computer vision problems (optical flow estimation, shape from shading) by Terzopoulos [115][116]. However, multi-grid methods are rather difficult to implement in general and need to be tailored to the type of problem at hand. In our case, application of the multi-grid method is complicated further by the type of constraint data which must be handled. The given data are in the form of intensity values along edge curves, which can be irregularly placed in the image and cannot be straightforwardly subsampled, because smoothing *across* discontinuities (such as edges) is not allowed.

More recently, another method has been introduced, which is easier to implement both on serial and parallel machines. This method uses hierarchical bases in the solution space instead of the usual nodal basis, where elements of the hierarchical basis set have larger support than the nodal basis elements. Hierarchical bases have been used for surface interpolation in computer vision by Szeliski [112], who used simple triangle functions as basis functions, and by Pentland [95], who used wavelet bases. This method is explained here briefly, since we have used Szeliski's combination of hierarchical bases and conjugate gradient minimization for comparison purposes.

Suppose one has represented a 1-D continuous function u on the interval $[0, 4]$ discretely by a vector of nodal basis variables $\mathbf{u} = (u_0\ u_1\ u_2\ u_3\ u_4)^T$ in a finite element space, where the basis functions are simple triangular functions, as illustrated in Figure 3.2(a). One could use a set of *hierarchical* triangular basis functions to represent the same function as illustrated in Figure 3.2(b), which shows the three hierarchical levels for this example ($l = 1, 2, 3$). These basis functions have more global support than the basis functions in Figure 3.2(a), so that interactions between variables at the coarse (high) levels have a much wider spatial impact. The variables in the hierarchical basis domain are represented by $\mathbf{v} = (v_0\ v_1\ v_2\ v_3\ v_4)^T$.

One needs to change basis from the nodal basis to the hierarchical basis to transform the function $\mathbf{u}$ into $\mathbf{v}$ or vice versa. This change of basis can be represented by a simple linear matrix transform $\mathbf{u} = \mathbf{T}\mathbf{v}$, in which each column of $\mathbf{T}$ gives the values of each hierarchical basis function at the nodal basis positions. E.g., one can see from Figure 3.2 that $u_2 = 0.5v_0 + v_2 + 0.5v_4$ (the height of each basis function is 1.0). The transform $\mathbf{T}$ can be computed very efficiently in a recursive manner in $O(N^2)$ operations.

A conjugate gradient descent algorithm working on $\mathbf{v}$ instead of $\mathbf{u}$ converges much faster. The hierarchical bases conjugate gradient method needs in the order of $\log(N^2)$ iterations to converge completely, instead of N^2. For further details of the algorithm, we refer to [112].

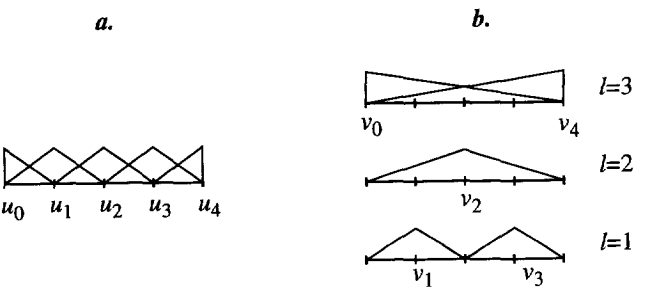


Figure 3.2 (a) Usual nodal finite element basis functions; (b) Hierarchical finite element basis functions split up into three levels (l = 1, 2, 3), each related to the size of the basis function support. (Adapted from Szeliski [112].)

Very recently, yet another approach to fast regularization has been proposed in [80], which is based on a new class of *multi-scale* stochastic prior models defined on a *quad-tree*. The algorithm resulting from the new problem definition makes use of the quad-tree to produce very efficiently a least-squares estimate of the solution. This non-iterative algorithm requires only $O(N^2)$ *operations* to reach the solution.

3.3 Multi-scale regularization

In this section, we propose a new method for solving Eq. (3.1), within the framework of regularization theory, which leads to an iterative algorithm similar to the one discussed previously, but with tremendously faster convergence. Yet each iteration has about the same complexity as that in the standard algorithm. We start by introducing a new type of stabilizing functional.

3.3.1 Multi-scale stabilizing functionals

Again, the idea underlying this approach is to allow more global interaction between points of the grid on which the solution is being computed, compared to the strictly local interaction allowed in the standard approach. These local interactions followed from applying finite differencing to u to approximate the derivatives in $S(u)$ as given by Eq. (3.4). The standard functional includes derivative measurements of the surface only *on the finest possible scale*. A new formulation can be justified by the notion that smoothness persists on *several* scales, coarse as well as fine. We propose to extend the surface prior model given by Eq. (3.4) to a class of *multi-scale* prior models by defining *multi-scale stabilizing functionals* as follows.

Let $h_k \equiv h(\mathbf{x}; \sigma_k)$ be a smoothing kernel, parameterized by σ_k, which controls the amount of smoothing applied. Now define a stabilizing functional on $u(x, y)$ on scale σ_k as follows:

$$S_k(u) = \int\int_{\Re^2} \left(\left(\frac{\partial h_k}{\partial x} * u \right)^2 + \left(\frac{\partial h_k}{\partial y} * u \right)^2 \right) dxdy \ , \tag{3.14}$$

where we use "smoothed derivative" operators $\frac{\partial h_k}{\partial x}$ and $\frac{\partial h_k}{\partial y}$ instead of the usual derivatives $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$. These smoothed derivative operators measure the variation in the surface u at a more *global* scale than the usual point derivative. Alternatively, because differentiation and convolution are commutative, this can be seen as taking derivative measurements of the solution u smoothed by h_k up to the scale σ_k:

$$S_k(u) = \int\int_{\Re^2} \left(\left(\frac{\partial}{\partial x}(h_k * u) \right)^2 + \left(\frac{\partial}{\partial y}(h_k * u) \right)^2 \right) dxdy \ . \tag{3.15}$$

Now, we can form a sum of the stabilizing functionals S_k, which corresponds to a multi-scale prior model of the image intensity surface:

$$S^*(u) = \sum_{k=1}^{K} \beta_k \ S_k(u) \tag{3.16}$$

where $\beta_k > 0$, k=1...K, are a set of weights. One can show that $S^*(u)$ is a valid stabilizing functional in the sense of Tikhonov [118]. Therefore, the same mathematical machinery as before can be brought forward to minimize the corresponding regularizing functional $E^*(u) = P(u, g) + \lambda S^*(u)$. The data compatibility constraint $P(u, g)$ is defined exactly as before.

Different choices of the smoothing kernels h_k lead to different optimal surfaces that minimize the multi-scale functional of Eq. (3.16). One can, for instance, use the two-dimensional Gaussian as a smoothing kernel. In that case, the resulting formulation of the reconstruction algorithm would fully reflect the edge analysis process in the sense that it minimizes the Gaussian smoothed gradient magnitude at points other than the edge points, which is exactly the quantity used in the edge analysis process to detect interesting features (see also the remarks at the end of Section 3.2.1).

Our primary interest here is computational efficiency and this dictates our choice of h_k. We show that, for a particular choice of h_k, an efficient and simple algorithm follows. Specifically, we use a **one-dimensional** box function

$$h(t;\sigma_k) = \frac{1}{\sigma_k}(U(t) - U(t - \sigma_k))$$

to define **one-dimensional** smoothers $h(x; \sigma_k)$ in the x-direction and, similarly, $h(y; \sigma_k)$ in the y-direction. Assuming $u(x,y)$ is continuous, substitution of $h(x; \sigma_k)$ for the first h_k and $h(y; \sigma_k)$ for the second h_k in Eq. (3.14) or Eq. (3.15) and manipulation leads to simply:

$$S_k(u) = \frac{1}{\sigma_k^2} \int\int_{\Re^2} \left((u(x,y) - u(x-\sigma_k, y))^2 + (u(x,y) - u(x, y-\sigma_k))^2 \right) dxdy \ .$$

This expression can be transferred into the discrete domain by simply sampling $u(x, y)$ at equidistant points $(x, y) = (m \cdot \Delta, n \cdot \Delta)$, with Δ the grid spacing, and restricting the scale parameter σ_k to multiples of the grid spacing, so $\sigma_k = o_k \cdot \Delta$, with $o_k \in \aleph$. This leads to:

$$S_k(u) = \frac{1}{o_k^2} \sum_m \sum_n \left(u(m,n) - u(m-o_k, n) \right)^2 + \left(u(m,n) - u(m, n-o_k) \right)^2 \ . \quad (3.17)$$

This is a straightforward generalization of Eq. (3.7): instead of differencing adjacent points only, also more distant points are differenced. Note, however, that we have not used the concept of finite differencing explicitly anywhere in the new formulation. We also note an analogy to the finite element approach, where a suitable function space is devised by a set of simple interpolating basis functions, leading to simple equations. Here, the smoothing kernel was devised in such a way that the partial derivatives in the original stabilizing functional vanished[a]. Note also, that in Eq. (3.17) we still use *all* the available points on every scale, no "subsampling" on coarse scales takes place as in multi-grid techniques.

The resulting functional $E^*(u) = P(u, g) + \lambda S^*(u)$ is still a quadratic form and can be minimized in much the same way as the previous functional $E(u)$. It is important to realize that minimizing $E^*(u)$ leads to an optimal solution *other* than the solution obtained by minimizing $E(u)$. While the minimizers of $E(u)$ have desirable properties in terms of its visual quality, it remains to be shown whether this is also true for the minimizers of $E^*(u)$. The latter solution may, or may not, be close to the former solution. In the next section, we report on experiments which show that, in fact, the multi-scale algorithm converges to a solution which is always close to the solution found by the original single-scale algorithm. This is a desirable property in cases where one insists on finding the surface which minimizes the standard functional $E(u)$. In such a case, the algorithm proposed here can be used to initialize an algorithm minimizing the standard functional, thereby obtaining a speed-up. However, a *mathematical* analysis of the properties of the surfaces which minimize $E^*(u)$ has not been performed.

3.3.2 Discrete formulation and iterative solution

The weights $\beta_k \in \Re$ in Eq. (3.16) and scale parameters $o_k \in \aleph$ in Eq. (3.17) can, in principle, be chosen freely to mold the behavior of the stabilizing functional $S^*(u)$. Interchanging the order of summation, the stabilizing functional $S^*(u)$ could also be written as a smoothness

a. This also means that if partial derivatives of orders higher than one were used in the stabilizing functional (e.g. with a thin-plate functional), another smoothing kernel should be used in order for the equations to be simplified. In general, for a p-th partial derivative, piecewise polynomials up to order p-1 can be used.

constraint on a surface smoothed by *the summation of kernels* h_k. The summation of kernels would form one composite kernel, which could be "shaped" using the β_k and o_k parameters.

Again, we opt for simplicity and efficiency. The scale parameter is set according to $o_k = k$, k=1...K. We set $\beta_k = o_k^2$, which results in a further simplification of the equations. In effect, this means that stabilizing functionals $S_k(u)$ of higher (coarser) scales are weighted more heavily than stabilizing functionals of lower (finer) scales.

The resulting multi-scale stabilizing functional is

$$S^*(u) = \sum_k \sum_m \sum_n (u(m,n) - u(m-k,n))^2 + (u(m,n) - u(m,n-k))^2 .$$

In matrix/vector notation, this can be written as follows:

$$S^*(\mathbf{u}) = \sum_k \mathbf{u}^T \mathbf{S}_k \mathbf{u} = \mathbf{u}^T \mathbf{S}^* \mathbf{u}$$

where $\mathbf{S}^*$ is the sum of scale-k stiffness matrices $\mathbf{S}_k$. The data compatibility functional $P(u, g)$ is the same as before. Minimizing the resulting functional $E^*(u) = P(u, g) + \lambda S^*(u)$ leads to the system of equations $\mathbf{A}^* \mathbf{u} = \mathbf{b}$, with $\mathbf{A}^* = \mathbf{P} + \lambda \mathbf{S}^*$. Note that $\mathbf{S}^*$ is not a sparse matrix, and neither is $\mathbf{A}^*$. Writing out one row of this system again gives more insight (cf. Eq. (3.11)):

$$\sum_{k=1}^{K} (\alpha(m,n) (u(m,n) - g(m,n)) + \lambda (4u(m,n) - u(m-k,n) - u(m,n-k) - u(m+k,n) - u(m,n+k))) = 0 \tag{3.18}$$

where we have rearranged the equation, such that all terms come under the summation, and we have implicitly scaled the regularization parameter λ by a factor K.

Again, we use Successive Overrelaxation (SOR) to solve $\mathbf{A}^* \mathbf{u} = \mathbf{b}$. However, we apply the iteration scheme on a *scale-by-scale* basis, i.e., each full iteration is split up into a sequence of simpler iteration steps, each working on a certain scale k. At each such "simple iteration", the residual at scale k is given by (cf. Eq. (3.12)):

$$r_k^{i+1}(m,n) = \alpha(m,n) \left(u^i(m,n) - g(m,n) \right) + \lambda \left(4u^i(m,n) - u^{i+1}(m-k,n) - u^{i+1}(m,n-k) - u^i(m+k,n) - u^i(m,n+k) \right) . \tag{3.19}$$

The residual is subtracted from the current solution in the usual way (cf. Eq. (3.13)):

$$u^{i+1}(m,n) = u^i(m,n) - \omega \frac{r_k^{i+1}(m,n)}{(\alpha(m,n) + 4\lambda)} . \tag{3.20}$$

Note that Eq. (3.19) and Eq. (3.20) are computed for all points (m,n) on the grid, for one particular scale k. Hence, this is called a scale-k iteration. After such a simple iteration is

completed, one can go to another scale k' and perform a similar iteration step. Now Eq. (3.19) is a very simple generalization of Eq. (3.12): instead of using the four adjacent pixels to measure the smoothness of u, pixels at distance k to the north, south, east and west are used. Again, the resulting algorithm runs entirely "in place", i.e., no extra storage space is needed. The operation is very similar to the computations in a filter process. Some of the corresponding "optimization filter masks" are depicted in Figure 3.3 for different scales k. At the upper left, the usual mask is drawn, which is a discrete approximation to a Laplacian. At the right and bottom, two masks are drawn which are used in the multi-scale optimization algorithm, at higher scales. It is clear that with these masks, interactions can take place on a more global scale. At the same time, each scale-k iteration takes the same amount of time as an iteration of the ordinary, single-scale, algorithm (when both are implemented on a serial computer). Hence, intensity information can propagate faster across the grid, speeding up the convergence of the algorithm. A theoretical analysis of the convergence properties of the scale-k stiffness matrices $\mathbf{S}_k$ and a comparison of different algorithms is given in Appendix A. The analysis in Appendix A further substantiates the convergence speed-up which can be obtained using the algorithm described here.

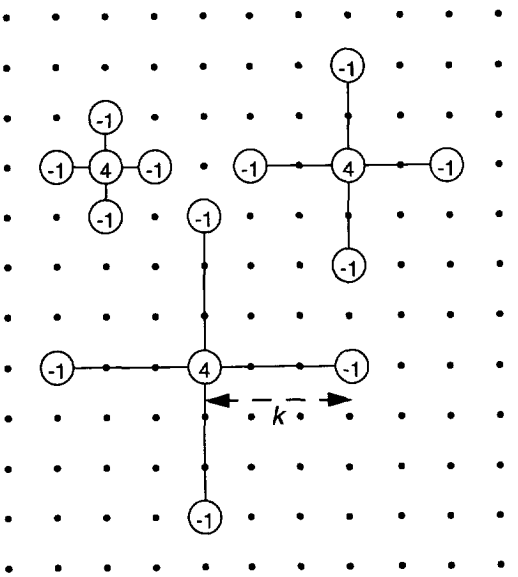


Figure 3.3 Multi-scale optimization filter masks of different scale k on a discrete grid.

If one were to use a different kind of smoothing kernel h_k, different kinds of "filter masks" would be applied during the minimization algorithm. If, for instance, one would use a Gaussian function, then the well-known Laplacian-of-a-Gaussian filter mask would result

instead of the approximations to Laplacians as shown in Figure 3.3, which are unusual in image processing (except the one at the upper left). Because *more* points on the grid would then be taken into account at each iteration, the resulting algorithm might need *fewer* iterations to reach a solution, at the cost of an increase in the computational complexity *per* iteration[b]. However, these effects remain to be investigated.

It remains to set up a scheme by which all the scales $k \in \{1...K\}$ are "visited". A scheme which has empirically been shown to be useful is the following. First, let k increase from 1 until K with steps of 1, i.e. simply visit all the scales once, from the finest to the coarsest. Then, let k decrease from K until 1, i.e. simply visit all the scales again, in reverse order. We call this process, going once upward through all the scales and once downward, a *cycle*. It turns out that, while the propagation speed of data values across the grid grows when k increases, smoothness at the fine levels is lost because the values of distant points is used to update each point on the grid. Subsequently, this smoothness can be regained when we let k decrease from K back to 1. After a complete cycle has been finished, a smooth result appears again. In the meantime, each cycle, consisting of $2 \cdot K$ iterations, brings the process considerably further (closer to convergence) than do $2 \cdot K$ "ordinary" single-scale iterations, which run in exactly the same time on a serial computer. This claim is substantiated by the convergence analysis in Appendix A. Note that, although each scale-k iteration can be performed separately, the algorithm should not be halted until a full cycle has been completed and all scales have been visited.

The resulting algorithm, which we refer to as *Multi-Scale SOR* (MS-SOR), is an extremely simple extension compared to the ordinary SOR algorithm and is quite simple to implement. At the same time, it provides great speed-ups of the convergence, as the experiments in Section 3.4 show. Before going on to the experiment section, we describe an important refinement of the basic algorithm in the following subsection.

3.3.3 Discontinuities and adaptive scale control

The multi-scale regularizing operator corresponds to a model of the image intensity surface. The multi-scale stiffness matrix $\mathbf{S}^*$ and the optimization filter masks corresponding to this matrix are directly related to the "model behavior" of the intensity surface. However, this model behavior - smoothness on several scales - does not persist across discontinuities in the image such as edges. Smoothness, as measured by the multi-scale optimization masks shown in Figure 3.3, should only be enforced *between* the discontinuities given by the edge curves. Therefore, such optimization masks should *not* be allowed to cross any discontinuities, i.e., the "branches" on a mask between the center point (with coefficient 4) and one of the four other points (with coefficient -1) are not allowed to intersect an edge curve.

This means that at each point on the grid (m,n), one must have an estimate of the distance from that point to the closest edge point. Then, this distance, denoted by $\rho_{m,n}$, can be used to "clip" the scale k locally, such that it never exceeds this value. Thus, the maximum scale which is

b. Very fast (recursive) implementations of the Laplacian filter have been developed for edge detection (e.g. [34]), which may also be useful here. These are based on various smoothing kernels.

visited during the optimization algorithm is no longer a constant (K) but depends on the location:

$$1 \leq k_{m,n} \leq \rho_{m,n} \leq K.$$

In our case, this is not very difficult to accomplish, since the discontinuities in the surface are exactly known: they are given by the constraints in $g(m, n)$, which represents the edge content of the image. An example is given in Figure 3.4, which shows a strip of pixels forming a sharp edge profile (with intensity jumps from roughly the value 50 to the value 100) along some edge curve and an optimization mask at location (m,n). The branches of this mask, with length $k_{m,n}$ are not allowed to cross the edge curve, approximately at distance $\rho_{m,n}$.

An estimate of the distances to the edge profile pixels can be made very efficiently by the Distance Transform. A city block Distance Transform [21] suffices. One can precompute this Distance Transform and store it for use during the optimization process. Extra storage space of the size of one image is then necessary. The running time of a Distance Transform is approximately equal to the time necessary to complete one or two iterations of the optimization algorithm. Once estimates of these distances are available, the adaptive scale control can be performed at almost no extra cost.

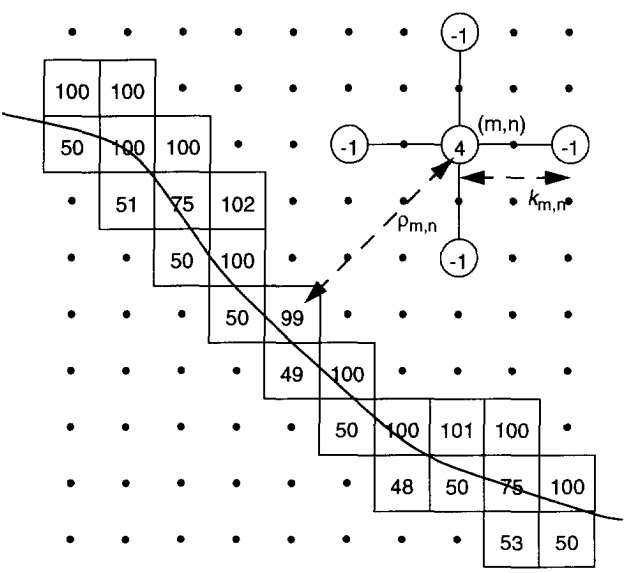


Figure 3.4 A discontinuity and the maximum scale of an optimization mask at position (m,n). The scale should be smaller than the distance to the closest edge point.

We remark here that a similar modification to the hierarchical bases conjugate gradient descent algorithm (discussed briefly in Section 3.2.4) has been made. This algorithm uses hierarchical basis functions, some of which have global extent. However, none of the basis functions may cross discontinuities such as edges, because smoothness constraints only hold *between* edges and may not be propagated *across* them. Therefore, we have implemented an *adaptive* transform **T** to change bases, in which recursive construction of higher levels of basis functions is cut off *locally* before these functions can cross an edge. This builds an adaptive hierarchical representation. Again, the Distance Transform is used to estimate the distance to the nearest edge.

3.4 Experimental results

In this section, the performance of the new multi-scale optimization algorithm on different data sets is evaluated.

3.4.1 Experimental setup

The main objective is to study the convergence of the algorithm in terms of the number of iterations it takes before the error in the solution becomes sufficiently small. To this end, the error in the solution at the i-th iteration is computed, denoted by e^i. The error is usually defined as the root-mean-squared (RMS) difference between the optimal solution $\hat{u}$ and the solution at the i-th iteration u^i:

$$e^i = \sqrt{\frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left(u^i(m,n) - \hat{u}(m,n) \right)^2}$$

for an M by N grid. The optimal solution is defined as the solution minimizing the single-scale regularization functional $E(u)$, i.e. the solution to Eq. (3.10). It can be obtained by running the single-scale SOR algorithm (SOR) or the conjugate gradient algorithm a sufficient number of iterations. We stress again that the solution to the multi-scale functional $E^*(u)$ does not have to be identical to the solution of $E(u)$. Therefore, in the following, the differences between these solutions in terms of the remaining RMS error and in terms of the visual appearance of these solutions is discussed as well. Of course, the same parameter settings (specifically, λ) as the ones used to obtain the optimal solution were used during the actual experiment.

The performance of the MS-SOR algorithm can be compared to the SOR algorithm by varying the maximum scale (K) used: for $K = 1$, MS-SOR is equivalent to SOR; for $K \geq 1$, MS-SOR is different. We also compare the MS-SOR algorithm to the hierarchical bases conjugate gradient algorithm (HBCG) from Szeliski [112]. When comparing algorithms, one must also remember that $2 \cdot K$ basic iterations of the MS-SOR algorithm together form *one* cycle, in which all the scales are visited twice (once going up and once going down). *One* basic iteration of the MS-SOR algorithm takes almost the same amount of computation as *one* ordinary iteration of the SOR algorithm, on a serial computer. However, *one* iteration of the hierarchical basis conjugate gradient algorithm takes on the order of a *few* iterations of the SOR algorithm.

Further, we study the effect of the regularization parameter λ and the sensitivity of the optimal solution with respect to this parameter. In most experiments, the overrelaxation parameter ω was set to 1.0, in which case Successive Overrelaxation actually becomes the Gauss-Seidel method.

The data sets used are:

- synthetic examples,
- real-world single images,
- real-world image sequences.

3.4.2 Synthetic examples

In this subsection, some synthetically generated surfaces are used to illustrate the behavior of the MS-SOR algorithm. Note that each graph in this subsection used to illustrate a synthetic surface actually shows a subsampled version of the real surface, in perspective view, for interpretability.

Constant surfaces

Figure 3.5(a) shows a (subset of a) set of surface constraints on a 113x113 grid, which are consistent with the values of two surfaces of constant height, bordering on each other in the middle. The heights of these surfaces are 100 and 150. The two surfaces can be considered to form a perfect, straight, step edge together. Data values are only available at the middle, describing the edge transition, and at the boundaries. Figure 3.5(b) shows the optimal solution $\hat{u}$, which was obtained using SOR for 5000 iterations. Note that the solution forms a good reconstruction in the sense that it is, in this case, identical to a pair of constant height surfaces. It clearly shows the surfaces with constant value. Figure 3.5(c) shows, as an illustration, an intermediate solution for the SOR algorithm after 100 iterations. The algorithm is clearly a long way from convergence.

In this experiment, the data values were used as hard constraints, i.e. the solution is fixed at the constraint points (equivalent to $\alpha(m,n) >> \lambda$). The overrelaxation parameter ω is set at 1.0. The optimal solution was obtained using the same parameter settings. The results with the MS-SOR algorithm (with adaptive scale control) for $K \in \{1, 2, 4, 8, 16\}$ are shown in Figure 3.5(d).

This figure clearly shows the dramatic speed-ups attainable when using the multi-scale algorithm. The algorithm converges to the optimal solution much faster, when K is allowed to increase from 1 to 16. For $K > 16$ (not shown here) the behavior is approximately equal to $K = 16$, in this case. As an illustration, for $K = 1$ (the single-scale algorithm) the solution after 100 iterations (shown in Figure 3.5(c)) has a RMS error value of $e^{100} = 77.71$. By contrast, for $K = 16$, the algorithm has already converged completely after 128 iterations (= four cycles of $2K$ iterations) and has reached an acceptable error level after just 32 iterations (= one cycle) of $e^{32} = 0.33$! The figure also shows that, in this case, the multi-scale algorithm converges to the same optimal solution as the single-scale algorithm.

The undulating effect e^i has in the figure shows the effect of the scheme we use to cycle through all the scales. This is most clear for $K = 4$ or 8. When going through one cycle, the scales are visited one by one, first from 1 through K and then back again. Each undulation of e^i in Figure 3.5(d) clearly corresponds to one cycle. At the start of a cycle, k is low and e^i decreases only slowly; then, as k goes up, e^i starts decreasing faster and faster until k reaches K; then, as k goes down again, e^i starts decreasing slower and slower until k is back down at 1. Then, after one cycle, the process repeats itself. Clearly, iterations at the highest scales pay off in the sense that global convergence is fastest. Iterations on lower scales, on the other hand, are necessary to optimize smoothness locally.

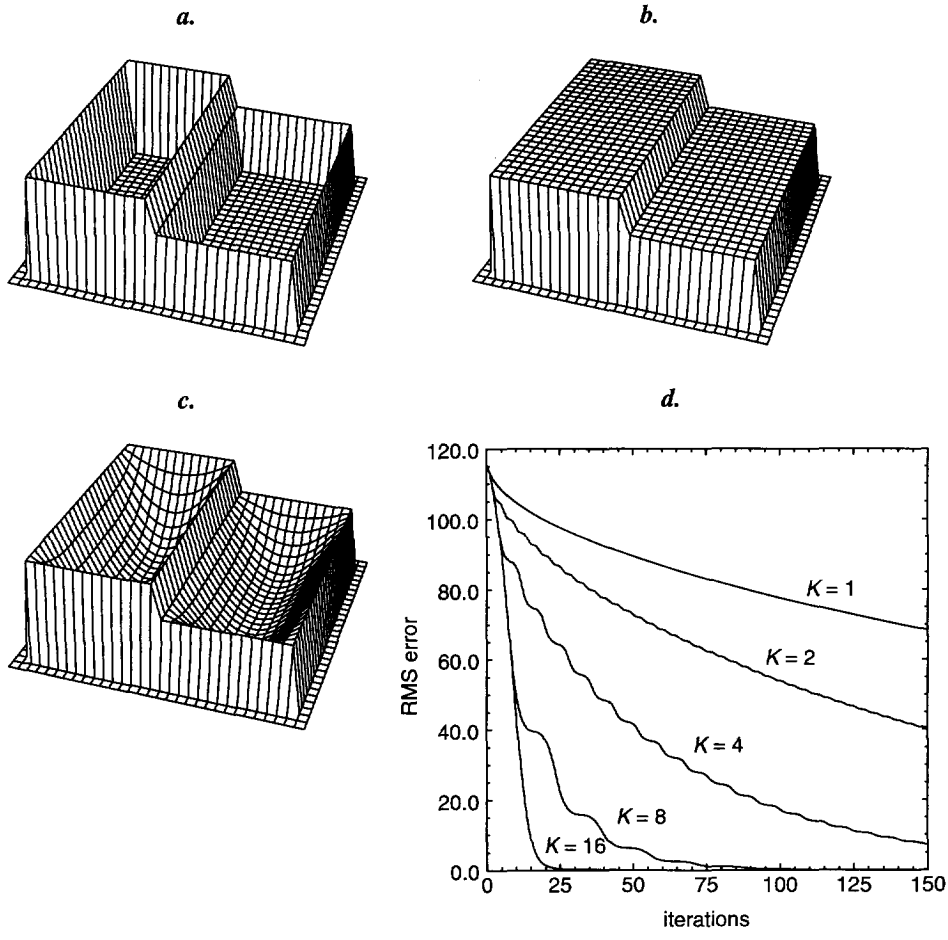


Figure 3.5 (a) Synthetically generated data set consistent with samples from two adjacent constant surfaces; (b) Optimal solution minimizing E(u); (c) Intermediate solution of the SOR algorithm at 100 iterations; (d) Results with MS-SOR: RMS difference between the solution at iteration i and the optimal solution as a function of i, for K between 1 and 16.

Linear surfaces

Figure 3.6(a) shows a (subset of a) set of surface constraints on a 113x113 grid, which are consistent with the values of two surfaces with linear variation, bordering on each other in the middle. This could be seen as a rooftop feature. Data values are only available in the middle and at the boundaries. These values range between 23 (at the base of the rooftop) and 205 (at the top). Figure 3.6(b) shows the optimal solution $\hat{u}$ which was obtained by running MS-SOR (4 cycles, $K = 50$) as an initialization and then SOR for 1000 iterations. It clearly shows the linear rooftop feature.

The performance of the MS-SOR algorithm for $K \in \{1, 2, 4, 8, 16\}$ on this data is almost the same as with the previous example, in terms of the RMS error value e^i. The MS-SOR algorithm shows similar speed-ups, but does not converge exactly to the same solution as SOR. (when $K = 16$, the remaining RMS error is approximately 0.63).

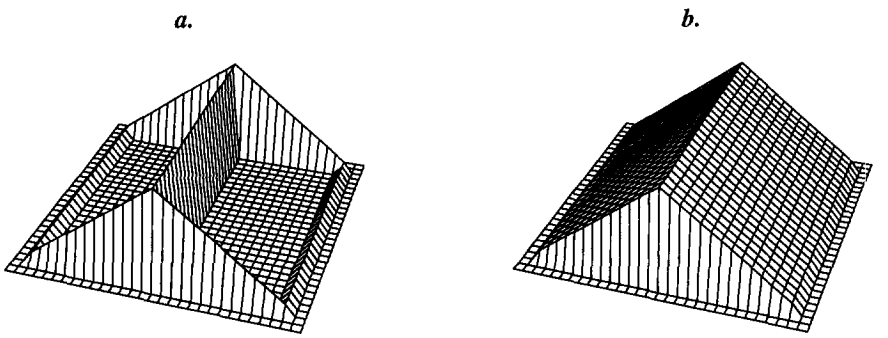


Figure 3.6 (a) Synthetically generated data set consistent with samples from two adjacent linear surfaces; (b) Optimal solution minimizing $E(u)$.

Parabolic surface

Figure 3.7(a) shows a (subset of a) set of surface constraints on a 111x111 grid, which are consistent with the values of a surface with parabolic variation. Data values are only available at the boundaries. These values range between 43 (at the base of the surface) and 147 (at the top). Figure 3.7(b) shows the optimal solution $\hat{u}$ for the functional $E(u)$ which was obtained by running MS-SOR (20 cycles, $K = 75$) as an initialization and then SOR for 5000 iterations. One notes that the optimal surface is not consistent with a parabolic surface everywhere. This is caused by the fact that the algorithm tries to set the Laplacian of the solution function to zero everywhere, while a parabolic surface certainly has nonzero Laplacian. Constant and linear surfaces have zero Laplacian everywhere, therefore they can be recovered well. Quadratic surfaces can not be recovered as well.

In this experiment, the data values were used as hard constraints. Again, $\omega = 1.0$. The results with the MS-SOR algorithm (with adaptive scale control) for $K \in \{1, 2, 4, 8, 16\}$ are shown in Figure 3.7(c). The RMS error value e^i behaves in almost the same way as it does with the

constant and linear surfaces but it decreases more slowly. Again, MS-SOR provides a substantial speed-up compared to ordinary SOR. Note that MS-SOR, again, does not converge exactly to the same solution as SOR. It is very close: the remaining RMS error for $K = 16$ is approximately 0.74. Once the error has reached that level, it continues to oscillate (due to the cycle scheme) between the values 0.74 and 0.90 (approximately). For increasing K, the difference between the solution as found by MS-SOR and the optimal solution as found by SOR increases slightly.

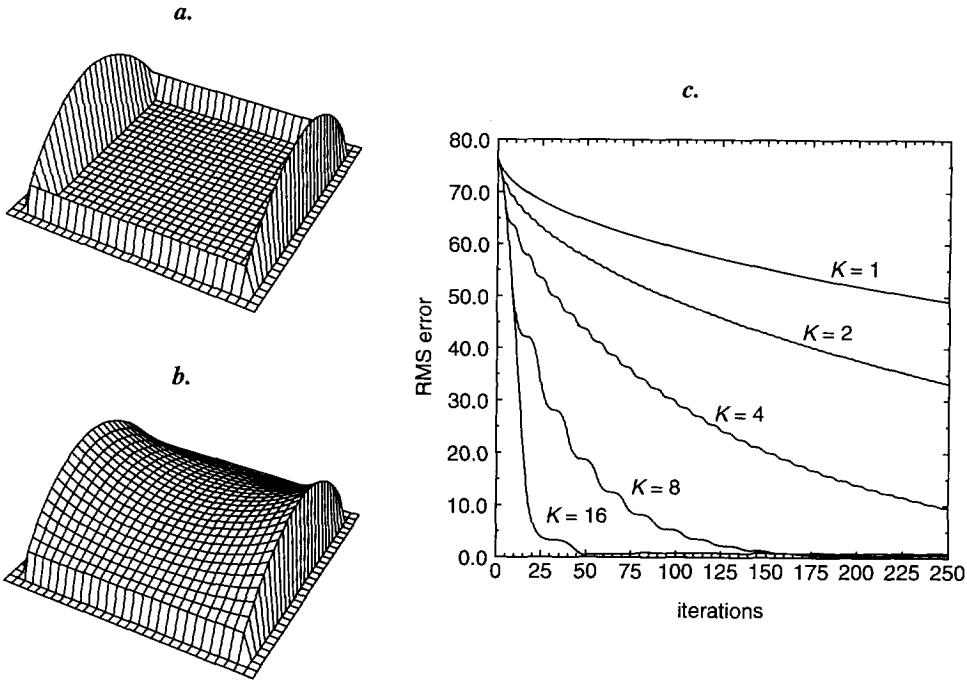


Figure 3.7 (a) Synthetically generated data set consistent with samples from a parabolic surface; (b) Optimal solution minimizing E(u); (c) Results with MS-SOR: RMS difference between the solution at iteration i and the optimal solution as a function of i, for K between 1 and 16.

Sampled sinusoid

The following experiment was set up to study the effect of varying λ (i.e., varying the strictness of the constraints given by the data values) and K (the maximum scale). Figure 3.8(a) shows a (subset of a) sparse random sampling of a two-dimensional sinusoid function $f(m, n) = 255 \cdot \sin(\pi m/110) \cdot \sin(\pi n/110)$ on a 111x111 grid (the number of samples is 5% of the original).

In this experiment, we set the value of λ to 10.0, 2.0, 1.0, 0.2, 0.1 respectively. We also used the input data as hard constraints, which corresponds to $\lambda \rightarrow 0.0$. For each value of λ, K was set to 1, 2, 4 and 8, respectively. Again, $\omega = 1.0$. As an example, Figure 3.8(b) shows the resulting surface for $K = 2$ under the hard constraints condition. Because each value of λ defines a **different** functional $E(u)$ and a **different** optimal solution, the results were compared (in the RMS sense) to the **original** input $f(m, n)$ instead of any optimal solution, as in the previous experiments. The behavior of the resulting RMS error measure is illustrated in Figure 3.8(c) for $K = 2$ and for different λ. This graph shows that the algorithm converges faster for smaller values of λ, i.e. the smaller the allowable data error, the faster the convergence.

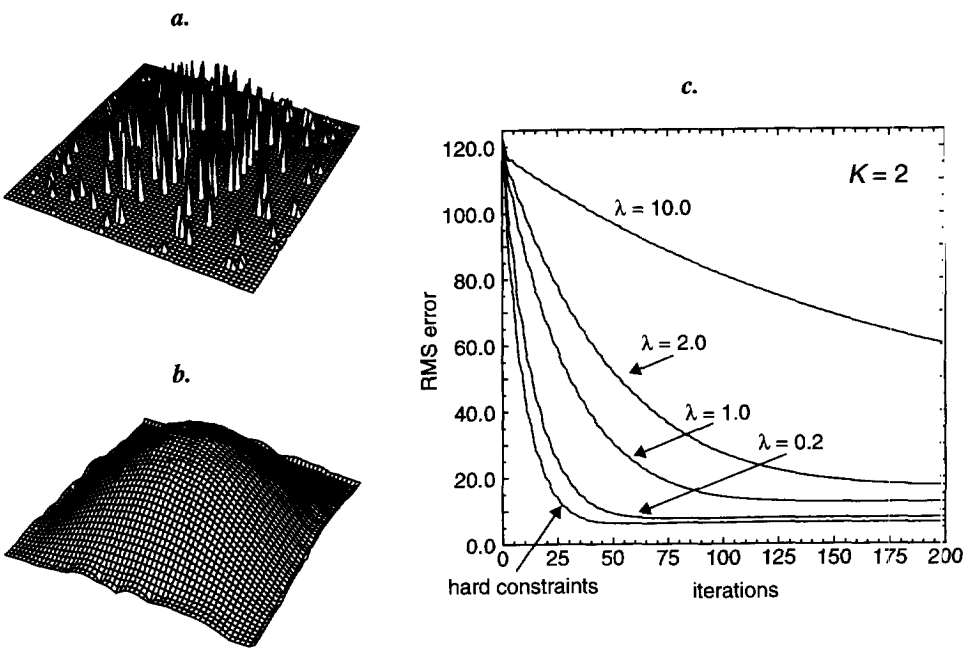


Figure 3.8 (a) A 5% random sampling of a two-dimensional sinusoid; (b) Solution obtained with MS-SOR using the data in (a) as hard constraints and with $K = 2$; (c) Results with MS-SOR: RMS difference between the solution at iteration i and the original sinusoid as a function of i, for $K = 2$ and different values of λ.

The values of the RMS of each *final* solution (after the algorithm has converged) are shown in Figure 3.9, for all values of K and λ. As expected, the RMS error at convergence is lower for smaller values of λ and increases for increasing λ. This graph also shows that the sensitivity of the final RMS error to the maximum scale K is much smaller than the sensitivity to the λ parameter. Furthermore, the differences in RMS value due to different values of K appear to decrease as λ decreases.

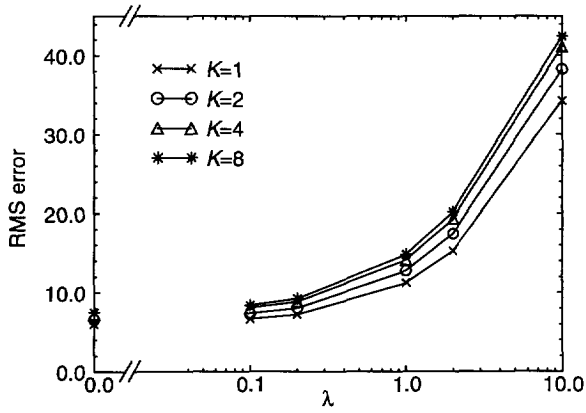


Figure 3.9 Final RMS values of the solutions for the sampled sinusoid for different values of λ and different values of K. The RMS values at the left ($\lambda \rightarrow 0.0$) were obtained using the input data as hard constraints.

3.4.3 Real-world single images

In this section, we discuss an experiment with the well-known real-world grey-level image "Lena", shown in Figure 3.10(a). Pixel values in grey-level images range between 0 and 255. Figure 3.10(b) shows a set of pixel constraints lying along the edge curves of the original. First, a Canny-type edge detector was applied to the original (as described in the previous chapter, $\sigma = 1.0$, $t_s = 0.8$). Then, every pixel of the original image lying 4-adjacent to an edge pixel was kept at its original grey-level and all the other pixels were set to zero. The result, in (b), can be considered a very basic representation of the relevant grey-level variations in the original image, since a sample was taken of the grey-level surface on each side of an edge. Note that this representation is very similar, but not identical to the edge-based representation proposed in this thesis. Here, only the base value and contrast of each edge are represented by the two samples on each side of an edge (although these values contain errors in the case of broad edges), the width of an edge is not preserved. This experiment serves only as an example to show the performance of the intensity surface reconstruction algorithm. The number of samples in (b) is approximately 16% of the original amount of pixels.

The image in Figure 3.10(c) shows the optimal solution (minimizing $E(u)$) using the data in (b) as hard constraints. The optimal solution was obtained using the hierarchical bases conjugate gradient descent algorithm (HBCG) of Szeliski: 200 iterations with maximum level equal to 6. The maximum level is denoted by L_{HBCG}. This image was obtained in approximately 71 seconds on a Sun SPARCstation 10. The value of the PSNR of this image with respect to the original in (a) is 25.09 dB. The image in Figure 3.10(d) was obtained using the multi-scale relaxation algorithm (MS-SOR), in approximately the same amount of computing time as with (c), 70 seconds. Here, 18 cycles with $K = 16$ were performed, corresponding to a total of 576 iterations. The PSNR of (d) with respect to the original in (a) is 24.92 dB. Visually, the images in (c) and (d) are indistinguishable. This example shows that,

with grey-level images, the visual quality of the solutions found by the MS-SOR algorithm is equal to that obtained by algorithms which minimize the standard regularization functional.

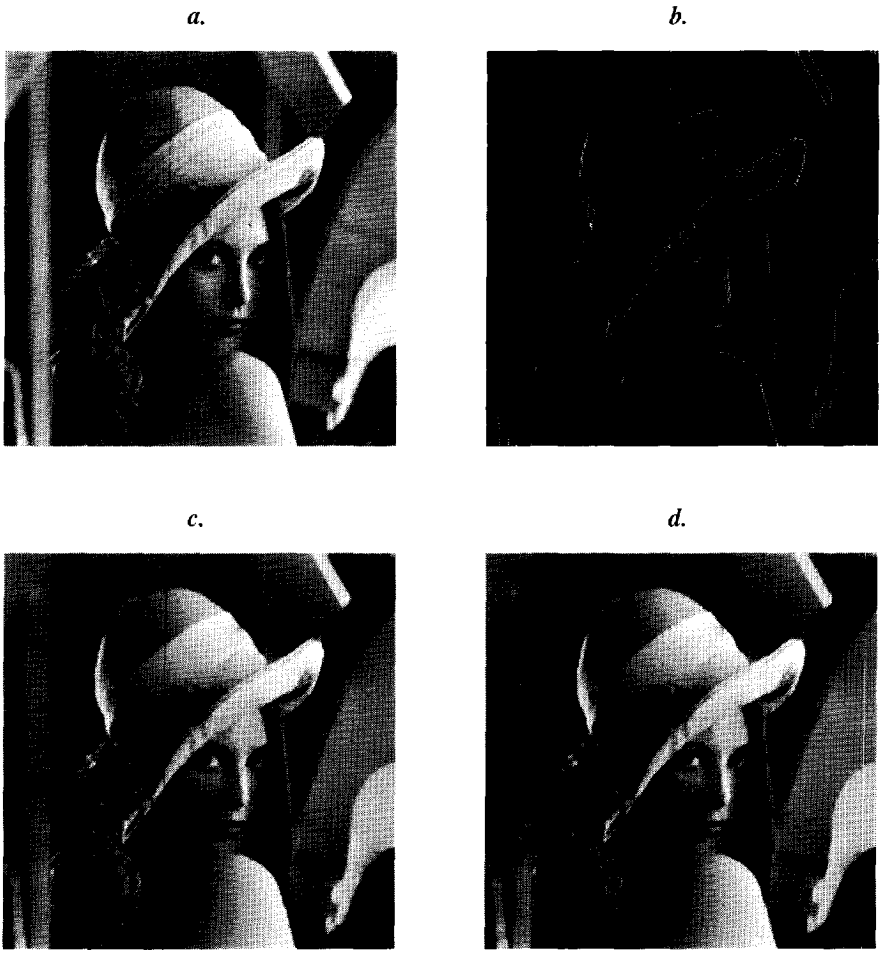


Figure 3.10 (a) Original "Lena" image (256x256); (b) Sampled 16% of the pixels in (a), the sampled pixels lie along edges; (c) Reconstruction obtained from (b) using 200 iterations of HBCG to minimize the single-scale smoothness functional; (d) Reconstruction obtained from (b) using 576 iterations of MS-SOR to minimize the multi-scale functional. The images in (c) and (d) take an equal amount of time to compute and are visually indistinguishable.

The visual quality of Figure 3.10(c) and (d) can be appreciated subjectively in the sense that most edges have been retained with high fidelity and the images appear to convey the contents of the original scene within a tolerable error level. The visual quality of other images, interpolated from a sparse sampling in a similar manner, is comparable to the example shown here. Of course, the quality depends on the edge detection threshold.

The RMS values of the error during minimization are shown in Figure 3.11. Results using the MS-SOR algorithm are shown in Figure 3.11(a) and results using the HBCG algorithm are shown in Figure 3.11(b). For MS-SOR, we used $\omega = 1.0$ and $K \in \{1, 2, 4, 8, 16\}$. For HBCG, the maximum level $L_{\text{HBCG}} \in \{1, 2, 4, 6\}$.

Again, convergence of MS-SOR improves for increasing K. Also, as asserted by Szeliski, the performance of the HBCG algorithm increases for increasing L_{HBCG}. Comparing the two, it is clear that HBCG needs fewer iterations to converge to the optimal solution than MS-SOR: MS-SOR with $K = 16$ needs about 250 iterations before the RMS comes within a range of 0.1 from its minimum value, while HBCG with $L_{\text{HBCG}} = 6$ needs about 60 iterations. However, one basic iteration of MS-SOR is much **simpler** and takes **fewer** computational steps than one iteration of HBCG. Analysis of the program code shows that MS-SOR requires 15 multiplication or addition operations per pixel per iteration, of which 7 are floating point operations; HBCG needs 47 multiplication or addition operations per pixel per iteration, of which 27 are flops. Indeed, a short benchmarking experiment with the Lena image showed that the same number of iterations of HBCG (with $L_{\text{HBCG}} = 4$) takes about three to five times the running time of MS-SOR ($K = 16$). Also, the results with MS-SOR can be further improved if the optimal overrelaxation factor ω could be found. An experiment with $\omega = 1.25$ was performed, in which the algorithm needed about 200 iterations to converge. One may conclude from the above that the MS-SOR algorithm is at least competitive with the HBCG algorithm as far as computational efficiency is concerned, while its simplicity is far greater.

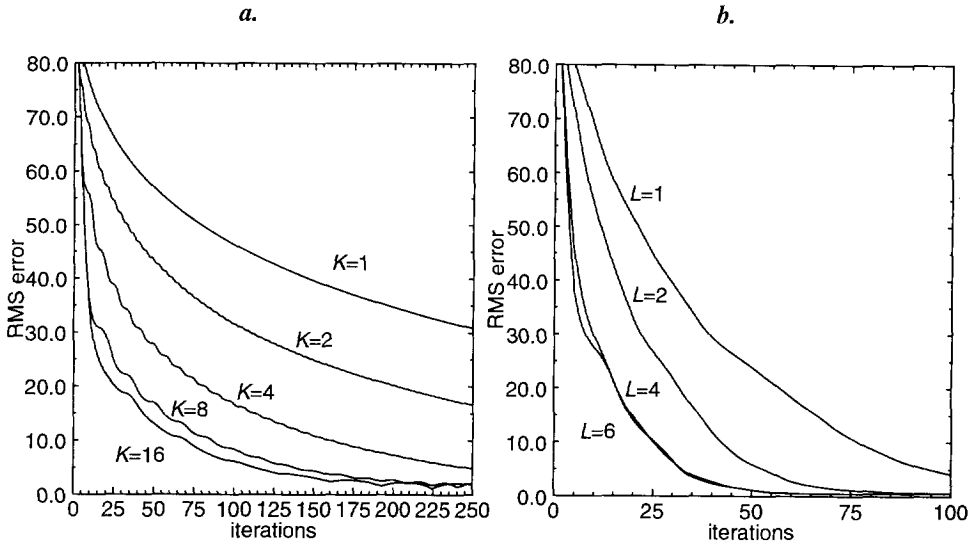


Figure 3.11 RMS difference between the solution at iteration i and the optimal solution as a function of i, for the Lena grey-level image: (a) Results with MS-SOR, for different values of K; (b) Results with HBCG, for different values of the maximum level, L_{HBCG}.

3.4.4 Real-world image sequences

SOR-type algorithms are very good at quickly annihilating the high frequency components of the error in a solution, while the low frequency components take very long to die out (this can explain the slow convergence of these methods). When one needs to reconstruct subsequent images from an image *sequence*, a computational gain can be obtained by making use of the correlation between these images. This correlation is very high if the motion in the image sequence is moderate. In this case, the low frequency components of an image being reconstructed can easily be 'picked up' from previous images in a sequence, since the difference information between subsequent images in a sequence is largely of a *high frequency* nature. Thus, a SOR-type algorithm would need fewer iterations per image (or *frame*) if it is part of an image sequence.

The next experiment can illustrate this. Here, we applied our method to the first 9 frames of the well-known "Claire" image sequence (each image is of size 352x288). For illustration, a frame of this sequence can be found in Chapter 4. In each grey-level frame of the sequence, edges were detected using a Canny-type edge detector ($\sigma = 1.0$, $t_s = 0.85$). Then, every pixel lying 4-adjacent to an edge pixel was kept and all the other pixels were set to zero, thus forming a very basic representation of the sharp variations in the original image as explained in the previous subsection. Then, the first frame of this sequence was reconstructed completely, on the basis of its edge samples, i.e., by iterating an optimization algorithm until convergence. Our experiment starts from there and concerns the reconstruction of frames two to nine only. Now, each time a new frame is to be reconstructed, the result of the previous frame is used as a first initialization of the iteration process as follows. For each new frame, the new edge samples are taken into account by filling in the corresponding pixels (i.e. the known pixels lying along edges) and keeping them fixed during the iteration process, but the remaining pixels *are not initialized to zero*, rather they take on the value they had in the *previous* frame. These remaining pixels are, of course, free in the sense that they can be set to other values by the iterative reconstruction algorithm. But now, the solution is found faster because the initial image is close to the solution already, except for mainly high frequency errors.

The value of the RMS error during the subsequent iteration processes of frames two to nine is shown in Figure 3.12. Here, the RMS difference between the solution at a particular iteration and the *original* frame was used. Four cycles of the MS-SOR algorithm with $K = 10$ were performed on each frame, i.e. a total of 80 iterations on each frame. The edge information in each incoming frame is used as hard constraints and $\omega = 1.0$.

Each time a new frame comes in, the RMS error peaks, because of the "innovations" due to, e.g., the motion in the scene. As can be seen, the MS-SOR algorithm can annihilate this error quite fast, to a tolerable RMS value. Of course, care must be taken that the solution does not "drift" away from convergence in time, i.e., if too few iterations are performed on each image frame, each subsequent reconstructed image will be further from convergence instead of reaching convergence (or at least a tolerable error level) for each frame.

Of course, more sophisticated methods can be used to take information from previous frames into account. One might think of using a motion compensator or a Kalman filter to produce a better prediction of each next frame in a sequence. Similar approaches have been studied in [30] in the context of multi-frame regularization for motion estimation.

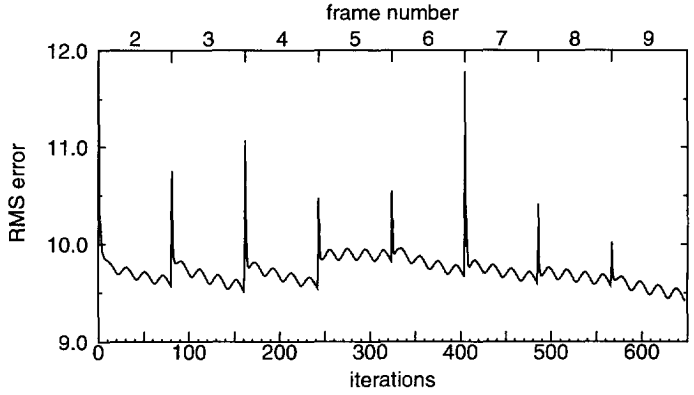


Figure 3.12 Results with MS-SOR on the Claire sequence, frames 2 to 9. Each frame to be reconstructed is initialized with information from the previous frame. The RMS difference between the solution at a certain iteration and the corresponding original frame is shown.

3.5 Conclusions

Within the framework of regularization theory, prior expectations about the nature of images can be invoked to obtain an interpolated intensity surface on the basis of a small set of intensity value constraints positioned along the edges in the image. This prior expectation is cast into the form of a smoothness functional which must be minimized, yielding an intensity surface with the least amount of (first-order) variation between constraints. This minimization can be performed using standard iterative algorithms, like successive overrelaxation (SOR) or conjugate gradient descent. While the quality of the reconstructed images is satisfactory, the number of iterations necessary to reach an acceptable level of quality can be quite large when standard algorithms are used.

In this chapter, we have proposed a new optimization algorithm based on simple iterations similar to the SOR type, which allows more global and thus faster propagation of information across the image grid. The new algorithm can be derived by introducing *multi-scale smoothness* functionals as a means to measure the amount of variation on multiple scales in the intensity surface. Experiments show that a very simple implementation of the multi-scale SOR algorithm converges much faster than standard SOR and that it is at least competitive with a hierarchical version of the conjugate gradient descent algorithm (Szeliski's HBCG).

The advantages of our multi-scale optimization algorithm (based on the membrane functional and SOR-type iterations) can be summarized as follows.

- It converges to a solution close to the one found by standard (single-scale) algorithms. Also, it can be used to provide a good initialization if one insists on finding the intensity surface that minimizes the standard smoothness functional.
- It gives visually pleasing reconstruction results, when applied to synthetic surfaces as well as to real-world grey-level images.
- It is relatively fast, i.e., it needs far fewer iterations to converge than the standard SOR algorithm. It is at least competitive with other "fast" algorithms (like hierarchical bases conjugate gradient descent).
- On a serial computer, the additional amount of computation necessary in each iteration (compared to one iteration of standard SOR) is almost negligible. Also, it is very easy to implement. Further, it is somewhat more suitable for a parallel implementation than any conjugate gradient algorithm, because the latter performs some global computations (inner products across the entire grid). With MS-SOR, communication between grid points is necessary if they lie within a restricted distance of each other.
- As it turns out, the algorithm is very flexible in the sense that any individual pixel can be given a particular maximum scale up to which smoothing must take place. Thus, it allows adaptive scale control, which is very important if discontinuities in the intensity surface play a part in the problem.

By the use of hierarchical (e.g. HBCG) or multi-scale (e.g. MS-SOR) algorithms, significant speed-ups of iterative image reconstruction algorithms can be obtained. More computational gains are expected if the image is embedded in a sequence, i.e., in a dynamic situation. Further computational gains seem possible by optimizing the algorithm: a significant number of options in the final implementation have not been investigated here, such as a different weighting of functionals at the different scales (β_k), a different sampling of the scales (σ_k) or a different schedule to visit all the scales. Furthermore, finding optimal values of the relaxation parameter ω will speed-up convergence even more.

The multi-scale approach to surface reconstruction may be extended using other stabilizing functionals (e.g. the thin-plate). In the case of image sequences, a temporal smoothness measure may be included in the stabilizing functional. This would increase the temporal consistency between intensity surfaces in the sequence.

Chapter 4
Still Image Coding

In the previous chapters, we introduced an edge-based image representation. It was shown how the parameters in this representation can be extracted from one-dimensional signals and two-dimensional images and how these signals can be reconstructed from the representation. The following chapters deal with the efficient *encoding* of the extracted parameters. Our prime intention in these chapters is to show how the edge-based image representation could be used to develop a high compression coding system. Furthermore, these chapters provide insight into the compression ratio and image quality that can be obtained by such a system. The question is whether one is able to code images at very low bit rates while maintaining a higher level of intelligibility and quality than would be maintained by waveform coders. This chapter is on the coding of *still images*. The next chapter is on the coding of *image sequences*.

4.1 Introduction

4.1.1 Problem formulation

We refer again to Figure 1.2, which illustrates the general coding scheme used here. The edge model-parameter extraction process and the resulting data structure containing these parameters is described in Chapter 2. Within the coding context, interpolation of a full intensity surface as discussed in Chapter 3, takes place *after* coding, transmission or storage, and decoding of the edge model-parameters. Specifically, the edge primitives contain:

a. the location of edge points, chained together into edge curves,

b. contrast (c), width (w) and center intensity value (m) model-parameters in each edge point.

The values of a parameter (e.g. the contrast) in each edge point along an edge curve are simply concatenated and subsequently treated as if they were the samples of a one-dimensional

waveform, to be called a *parameter signal* (e.g. the contrast signal) or *parameter sequence*. If there are M edge curves, there will be M such parameter signals of each type. For each edge curve, one has the *contrast signal* $c(n)$, *width signal* $w(n)$ and *center intensity signal* $m(n)$. Besides these parameter signals, the sequence of chain-coded links $l(n)$ codes the location of edge points on an edge curve.

The previous edge analysis stage can be considered to be the message extraction part of the coding scheme. The main parameters in that stage which influence the content of our representation are: the scale of the analysis filters (σ), the threshold used to select edge points (t_s), the minimum number of links in an edge curve (ξ). In this and the following chapter, we apply the edge analysis process on a single scale and use $\sigma = 1.0$ throughout the design of the coder and the experiments. The images used in this thesis contain little noise and are quite sharp. This allows us to use the minimum scale on which the edge analysis algorithm is expected to work properly in sampled images, thus avoiding as much cross talk between nearby edges as possible. The edge selection threshold is crucial in determining the relative number of edge points in the image and therefore has a strong effect on the final quality of the decoded and reconstructed images, and on the number of bits necessary to code the image. The relative number of edge points depends strongly on the kind of input image but usually lies around 5-10%. Thus, the edge information extracted can already be considered a compact representation, albeit that the compression ratio is not very high.

Some form of data reduction has already taken place during the edge analysis stage. The parameter encoding stage of the scheme is intended to further compress the amount of information in the edge primitives and may employ any technique from the full scale of data compression and data reduction techniques already known ([58][60]). Since the value of an edge parameter is expected to vary only slowly along each edge curve, the parameter signals are expected to contain a significant amount of redundancy. An example of each of these signals is given in Figure 4.1. This example illustrates that, although some sharp transitions do occur in these signals, they also contain large stationary parts in which the variation is quite small. This observation suggests that high correlation exists between neighboring samples of each parameter signal. Besides compressing the edge parameter signals, also the edge point location or shape information must be encoded efficiently.

4.1.2 Outline

In Section 4.2 we investigate further which coding method is suitable for the type of edge parameter signals illustrated in Figure 4.1. A lossy coding scheme is designed on the basis of the statistics of these parameter signals. Next, in Section 4.3 we describe how the edge point location information can be coded. We have chosen to code this information without loss. Section 4.4 reports experiments with the designed edge-based coder for still images. Besides reporting the objective performance, we pay some attention to the kind of visible artifacts introduced by this type of coding scheme. We also compare our scheme to the JPEG compression standard.

In the following, we do not address the problem of *bit allocation*. The objective in bit allocation is to adaptively distribute the total number of available bits among different

components in the representation of a signal (e.g. by assigning quantizers, each with a different rate and distortion) such that the overall distortion in the decoded signal is minimal. Each parameter signal corresponding to each edge curve can be considered such a component in terms of the bit allocation problem. Our coding scheme is suboptimal in the sense that it applies a fixed method on each edge curve and does not consider the relative effect of the distortion of one type of parameter signal (e.g. the contrast signal) on the overall distortion, with respect to the effects of the distortion of other types of parameter signals (e.g. the width signal or center intensity signal).

In Section 4.5, we discuss adaptive coding and argue that for certain scenes the perceptual quality of decoded images can be improved by adaptively assigning more bits to certain areas in the image than to other areas. This can, e.g., be realized by the use of a priori knowledge during the edge selection process and by developing more refined edge selection criteria. This chapter is closed with conclusions.

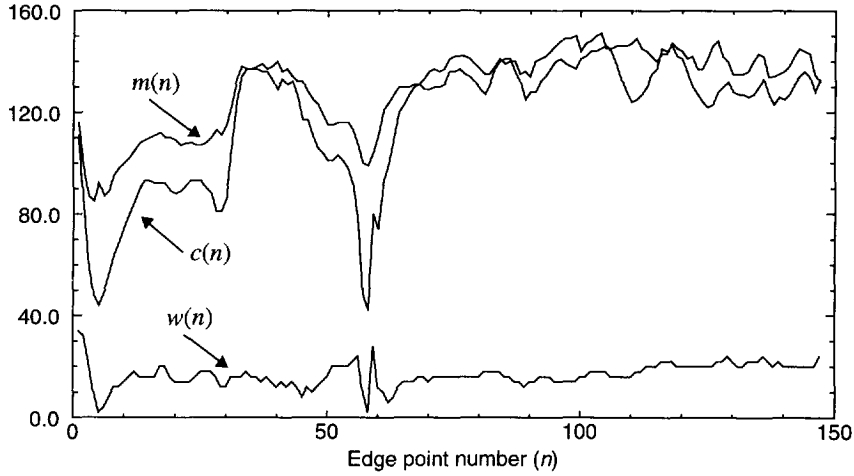


Figure 4.1 Examples of parameter signals taken from the edge-based representation of the "Scarf" image. These are taken from one of the longer edge curves (147 edge points). The center intensity signal is denoted by m(n), the contrast signal by c(n), the width signal (multiplied by 20 for visibility) by w(n).

4.2 Edge parameter coder design

4.2.1 Description

Although the ideas for transmitting image-derived values along edges have been around for a long time, only a few types of methods for coding these values have been reported. Simple uniform quantization has been used ([46], [69], [37]) or linear prediction followed by quantization ([82]). Quantization is usually followed by some form of entropy coding, e.g., Huffman coding. Some have used approximation by polynomials ([70], [26]). The use of the

Discrete Cosine Transform or subband or wavelet decompositions has not been reported, although popular in image coding in general.

In principle, each edge primitive can be coded separately. Here, we have used the simple scheme illustrated in Figure 4.2. A combination of downsampling and *Differential Pulse Code Modulation* (DPCM) is used to exploit the high auto-correlation between neighboring parameter samples, as suggested in the previous section. Note that Figure 4.1 also suggests significant cross-correlation might exist between the contrast and center intensity signal. This cross-correlation between different types of signals is not exploited in our coding scheme. In the next subsection, the downsampling process is described. Then, in Section 4.2.3, we further investigate the auto-correlation of each signal and the gain which can be achieved by DPCM quantization over direct quantization. The auto-correlation and DPCM gain depend on the number of steps in the downsampling applied previously. In Section 4.2.4, the design of the quantizer in the DPCM loop is discussed and the Variable Length Code (VLC), which is applied subsequently.

We use in our scheme an optional low-pass pre-filter preceding the parameter encoding, to ensure that the extracted parameter signals are band-limited. We also apply a low-pass post-filter after the DPCM signal reconstruction in the decoder, in order to smooth out some of the granular noise introduced by the (coarse) quantization.

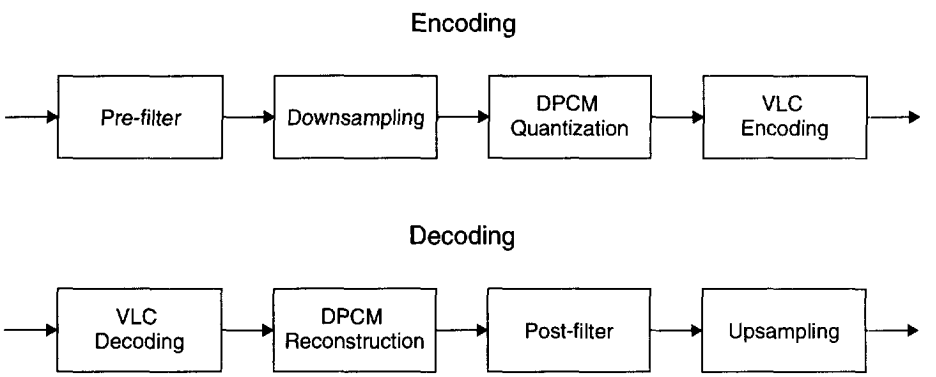


Figure 4.2 Block diagrams of encoding and decoding stages of edge parameter signals.

4.2.2 Downsampling

We call *downsampling* the process of repeated low-pass filtering and decimation of a discrete input signal, denoted by $f(n)$ with length N. Here, the low-pass filter is chosen such that it reduces the bandwidth of the input by a factor of two and the decimation step likewise reduces the number of samples in the signal by approximately a factor of two. Repeated low-pass filtering and decimation of the input signal generates a sequence of signals, increasingly reduced in length, denoted by $f_l(n)$. The number of filter/decimation steps used in the downsampling is denoted by L, so $0 \le l \le L$ and $f_0(n) = f(n)$.

We use Burt's widely used symmetric Gaussian-like filter in the downsampling, which is given by $g(0) = 0.4$, $g(1) = g(-1) = 0.25$ and $g(-2) = g(2) = 0.05$ (see [24]). The low-pass filtering with the filter $g(n)$ and the decimation can be described simultaneously by

$$f_l(n) = \sum_{i=-2}^{2} g(i)\ f_{l-1}(2n - i)\ ,$$

with $0 \le n \le N_l - 1$, $N_l = \left\lceil \frac{1}{2} \cdot N_{l-1} \right\rceil$, $N_0 = N$.

In our case, only the (L times decimated) signal $f_L(n)$ is quantized and transmitted, producing a distorted version $\hat{f}_L(n)$.

We call *upsampling* the process of repeated expansion and interpolation filtering of a downsampled signal $\hat{f}_l(n)$. The expansion step consists of inserting zeros between every sample of the input, such that the number of samples increases by a factor of two. Interpolation filtering consists of applying a low-pass filter to the intermediate result, thereby interpolating the newly inserted samples.

We use Burt's Gaussian-like filter in the upsampling as well. The expansion and interpolation filtering can be described simultaneously by

$$\hat{f}_l(n) = 2 \sum_{i=-2}^{2} g(i)\ \hat{f}_{l+1}\left(\frac{n-i}{2}\right)$$

with $0 \le n \le N_l - 1$ and only terms for which $\frac{n-i}{2}$ is integer are included. The reconstructed parameter signal is $\hat{f}(n) = \hat{f}_0(n)$.

During the filtering, we use an odd symmetric extension at the start and end of the signal to perform the filtering without introducing artificial transitions.

4.2.3 Auto-correlation structure and DPCM

Each downsampled parameter signal $f_L(n)$ is coded using DPCM (see e.g. [60]). DPCM is a form of causal predictive waveform encoding, in which a prediction of the current sample is formed on the basis of *previously reconstructed* samples. The *prediction error sequence*, $e(n)$, is actually quantized and transmitted. The quantized prediction error is denoted by $\hat{e}(n)$. The *auto-correlation* properties of the input signal are crucial for the design of an optimal linear predictor and for minimizing the variance of the prediction error sequence. This variance is bounded below by the variance of the prediction error sequence *without quantization* (i.e. when the prediction is done on the basis of past *input* samples), denoted by $\varepsilon(n)$. The latter variance, σ_ε^2, is minimized by solving the well-known *Yule-Walker equations*, which involves computation of the auto-correlation of the input signal. The higher the auto-correlation, the lower σ_ε^2, and the higher the *coding gain* G_p of DPCM over PCM (Pulse Code Modulation or direct quantization), defined as

$$G_p = \sigma_f^2 / \sigma_\varepsilon^2$$

where σ_f^2 is the variance of the input signal. This gain also depends on the order of the predictor.

In Table 4.1, values found for the auto-correlation coefficients of parameter signals from the "Scarf" image are illustrated. For the definition of auto-correlation used here, see Appendix B.1. No downsampling was applied to the input parameter signal in this case. Similar correlation values can be found for parameter signals extracted from other images as well. As can be seen in the table, the correlation between neighboring samples is very high for the contrast (c) and center intensity (m) parameter signals, while it is somewhat lower for the width (w) signal.

Table 4.1 Mean, variance and auto-correlation coefficients (up to the fourth) of parameter signals from the Scarf image. Edges are extracted with $\sigma = 1.0$, $t_s = 0.85$, $\xi = 4$, yielding 145 edge curves with 3714 edge points. Parameters c and w are pre-filtered, no downsampling takes place.

Type	μ_f	σ_f^2	$\rho_f(1)$	$\rho_f(2)$	$\rho_f(3)$	$\rho_f(4)$
m	129.07	622.50	0.966	0.931	0.895	0.867
c	71.03	942.83	0.968	0.925	0.885	0.858
w	0.97	0.24	0.909	0.783	0.680	0.616

Table 4.2 shows the value of the prediction coding gain G_p which can maximally be achieved by the optimal Yule-Walker predictor design, using one, two, three or four predictor coefficients. The coding gain is quite high. Again, it is higher for both the contrast and center intensity signals than for the width signal. This table also shows that using more than two prediction coefficients does not improve the coding gain significantly. We will only use second-order predictors in the following.

Table 4.2 Coding gains G_p for parameter signals from the Scarf image, for linear predictors up to the fourth order. The same settings apply as in Table 4.1.

	predictor order			
Type	1	2	3	4
m	15.02	15.04	15.05	15.19
c	15.99	16.63	16.67	17.29
w	5.77	6.16	6.22	6.32

The above results suggest that significant coding gains might still be achieved, even if the parameter signals are downsampled *before* DPCM is applied. This is confirmed by the data in Table 4.3 and Table 4.4, which show auto-correlation coefficients and coding gains for parameter signals extracted from the "Scarf" image, after *one* downsampling step (i.e. by a factor of two) and after *two* downsampling steps (i.e. by a factor of four) respectively. Coding

gains are computed for a second-order predictor. It can be seen that the auto-correlation coefficients are still quite high and so are the theoretical coding gains.

Therefore, it seems advantageous to apply downsampling to the parameter signals, because the gains of downsampling, in terms of bits spent, are not reduced quickly by a loss of efficiency of the DPCM stage. The combination of downsampling and DPCM is a simple yet efficient one. The prediction coefficients of the DPCM stage are transmitted to the receiver as side information.

Table 4.3 Auto-correlation coefficients (up to the fourth) and coding gains using a second-order predictor for parameter signals from the Scarf image, after one downsampling step (L=1). Further settings apply as in Table 4.1.

Type	ρ(1)	ρ(2)	ρ(3)	ρ(4)	G_p
m	0.943	0.890	0.836	0.776	8.98
c	0.939	0.889	0.852	0.825	8.53
w	0.848	0.699	0.614	0.585	3.58

Table 4.4 Auto-correlation coefficients (up to the fourth) and coding gains using a second-order predictor for parameter signals from the Scarf image, after two downsampling steps (L=2). Further settings apply as in Table 4.1.

Type	ρ(1)	ρ(2)	ρ(3)	ρ(4)	G_p
m	0.910	0.801	0.700	0.652	5.98
c	0.947	0.888	0.857	0.887	9.73
w	0.814	0.681	0.633	0.619	2.98

4.2.4 Prediction error quantization

Before the quantization of the parameter samples of each edge curve takes place, the overall mean of the samples of each type is subtracted from each edge curve of the corresponding type and transmitted separately, quantized into 6 bits. Also, instead of coding *signed* contrast samples, only the *magnitude* of each contrast sample is transmitted. Instead, *one* sign is computed for all samples of the contrast signal of an edge curve, on the basis of the majority of the signs of contrast samples per curve. Thus, only *one* sign-bit per edge curve is transmitted.

The (scalar) quantizer inside the DPCM loop is designed on the basis of the statistics of the prediction error sequence $\varepsilon(n)$, for all n except $n = 0$ (the first sample is coded using fixed length coding taking 4 bits). To this end, we have investigated these statistics experimentally and have fitted the resulting frequency distributions to some model distributions.

In Figure 4.3(a), the frequency distribution of DPCM prediction error values for the contrast parameter signal (after one downsampling step, second-order predictor), taken over a set of fourteen different images, is shown as an illustration. The shape of the frequency distribution is quite peaked. The prediction error frequency distribution for the other types of signals are similar (and so are the distributions for parameter signals after two downsampling steps applied or not downsampled at all). As a model distribution we use the class of *Generalized Gaussian* probability density functions (pdf) $p(x; \mu, \sigma, \alpha)$, with shape parameter α (see Appendix B.2.1). The Generalized Gaussian probability density function (pdf) with $\alpha = 0.75$ is given as an illustration in Figure 4.3(b). When overlaid, the measured frequency distribution and the Generalized Gaussian distribution fit quite well, visually.

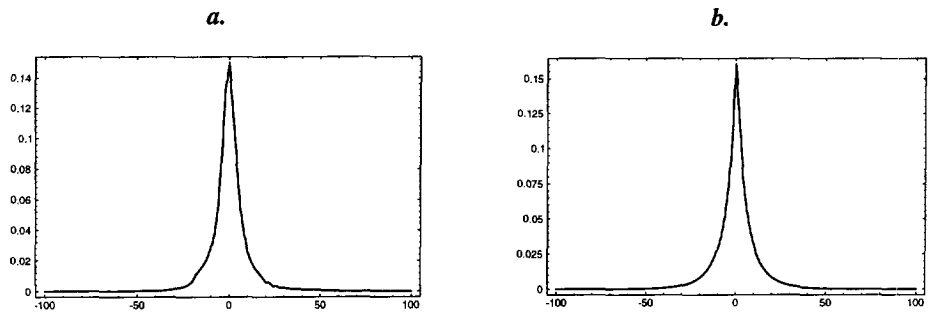


Figure 4.3 (a) Frequency distribution of DPCM prediction error values for contrast parameter signals downsampled with L=1, $\mu_\varepsilon = 0.295$, $\sigma_\varepsilon = 11.021$ (the prediction error values were binned into a histogram of size 101 by grouping them into ranges of 2.0); (b) Generalized Gaussian pdf, with the same μ and σ as in (a) and $\alpha = 0.75$ (the continuous pdf was binned into a histogram of size 101 by numerical integration within ranges of 2.0, thus calculating the expected frequency of events in each bin).

To determine objectively the shape of the model distribution which matches the given distribution best (among a set of available candidates), we use the chi-square test statistic [98]. The chi-square statistic for binned distributions is used in the following form:

$$\chi^2 = \sum_i \frac{(h_s(i) - h(i))^2}{h(i)}$$

where the sum is over all the bins, $h_s(i)$ is the frequency of events observed in the i-th bin and $h(i)$ is the frequency of events expected in the i-th bin according to the known probability density function. The chi-square test statistic results when fitting the frequency distribution of the prediction error of the contrast parameter signal in Figure 4.3(a) to Generalized Gaussian distributions with $\alpha = 0.50$, 0.75, 1.00 and 2.00 is shown in Table 4.5 as an example. In this case, the distribution with $\alpha = 0.75$ is the best match objectively as well as visually.

The complete results for the contrast, center intensity and width signals after zero, one or two downsampling steps are given in Appendix B.2.2. These results show that the value of the

chi-square test statistic is always relatively low for $\alpha = 0.75$. This value appears to give the overall most favorable fit between the Generalized Gaussian distribution and the distribution of the prediction error sequence of the parameter signals, among the set of candidate values. Note that the value $\alpha = 0.75$ is not always the optimal one. However, for the sake of simplicity, we use the value 0.75 in all cases in our coding scheme.

Table 4.5 Values of the chi-square test statistic when matching the distribution of the prediction error of the contrast signal (downsampled once, second-order predictor) with a Generalized Gaussian distribution with shape parameter α.

α	χ^2
0.50	0.10
0.75	0.03
1.00	0.11
2.00	$1.20\ 10^{10}$

Quantizers can now be designed on the basis of the given distribution. Here, *Uniform Threshold Quantizers* (UTQs) are employed [38]. The rate-distortion performance of UTQs is nearly equivalent to that of optimal *Entropy Constrained Quantizers* [38][60]. The latter quantizers are designed to minimize the distortion while keeping the output entropy fixed (as opposed to the well-known Max-Lloyd quantizers which keep the number of representation levels fixed). The decision levels of UTQs are uniformly spaced, but the representation levels generally are not. An example of such a quantizer, optimized for a Generalized Gaussian distribution with $\mu = 0$, $\sigma = 1.0$ and $\alpha = 0.75$, is given in Appendix B.2.3. In practice, the quantizer representation and decision levels are scaled to the variance of the input, in our case to σ_ε^2 (which is transmitted to the receiver as side information).

The quantizer output is compressed further by using *variable length encoding* (VLC) [19]. Here, *Huffman coding* is used because it is both simple and efficient. Code words are assigned on the basis of the probability of each quantizer representation level. The expected rate of the quantizer is then given by the average length of the code words.

4.3 Edge location coder design

4.3.1 Description

This section discusses the coding of the edge point locations, also referred to in literature as the problem of *contour coding*. In this thesis, the location data is coded without loss. Lossy techniques such as approximation by spline curves or circular arcs are not considered in this thesis, nor do we restrict the edge curves in their geometry.

A family of techniques which have been found efficient for coding this type of data (similar to line drawings) is known as *chain coding* [41][59]. In chain coding, the *links* between

successive eight-connected edge points on a curve are assigned a code. As explained in Chapter 2, there are eight different chain codes (numbered 0...7), each coding a different direction on the grid (east, north-east, north, north-west, west, south-west, south and south-east).

Chain coding has sometimes been used in combination with Huffman coding to make use of the non-uniform probability of each chain code, or of combinations of successive chain codes [26][37][46]. We use an approach similar to the one proposed recently by Lu and Dunham [78], because it is both simple to implement and very efficient. More details are presented in the next subsection.

With chain coding, one still has to code the *starting points* of edge curves, but also *branch points* and *end points*. This is discussed in Section 4.3.3.

4.3.2 Chain coder

Denote a sequence of chain-coded links in an edge curve by $l(n)$. Because of the smoothness of many edge curves, subsequent links are statistically dependent and may be modeled by a k-th order Markov chain. The values of the k previous chain-coded links together form the *state* of the Markov chain. In each state (i.e., depending on the k previous links), certain links are more likely to occur next than are others. For example, consider a piece of a chain-coded curve given by '0 1 2 1 2 2 3 3 2' and let k=2. At the point when the first '2' is about to be processed, the state of the Markov chain is '0 1'. After the '2' has been processed, the state becomes '1 2'. Subsequently, the state becomes '2 1', '1 2', '2 2', '2 3', etc., each time a chain code has been processed. Being in the state '1 2', it is more likely that the next chain code is, e.g., a '2' than a '4' (the latter would mean a sharp turn). Thus, depending on the previous chain codes, there is a non-uniform probability distribution of the next chain code.

Huffman's coding technique can be employed to exploit these non-uniformities by encoding high-probability links with short code words. To this end, Huffman code word tables have to be designed for each state in the Markov model. This can be done on the basis of empirically determined (conditional) frequency distributions of the chain-coded links.

We use only zero-, first- and second-order Markov models. Frequency distributions for these models have been determined experimentally using a set of training images, and are shown in tables in Appendix B.3, along with the lengths of the assigned Huffman code words. A summary of the resulting entropies, calculated on the basis of the frequency distributions, and expected rates, calculated on the basis of the assigned code words, is given in Table 4.6.

In practice, the first link in a sequence is coded using the Huffman code for the zero-order Markov model, the second link is coded using the appropriate Huffman codes for the first-order model, and the remaining links are coded using the appropriate Huffman codes for the second-order model. Then, a termination code must be sent to signal the end of the sequence. We have implemented the termination code by adding a link to the sequence, which codes a reverse link with respect to the last link, since this cannot normally occur in practice. This termination code has been taken into account in the Huffman code word assignment procedure described above, to avoid the termination code being assigned very long code

words. In most cases this code costs only three bits. By use of the termination code, the *length* of each edge curve does not have to be transmitted.

Table 4.6 Summary of entropy and expected rate of chain-coded links for Markov models of different order.

Model order	Entropy	Expected Rate
0	2.771	2.799
1	1.627	1.764
2	1.487	1.629

4.3.3 Special points

Many edge curves are connected with each other via branch points. Therefore, the starting point of an edge curve does not have to be coded for **all** edge curves, but only for a number of curves equal to the number of connected components in the edge image. The other starting points correspond to the branch points, indicated by some extra bits as explained below. We have not implemented a coder for the starting point locations, but we assume they can be coded with 12 bits on average using, e.g., run-length coding techniques.

After the edge location decoder has encountered a sequence termination code, it now has to know whether the sequence ends in a branch point or in an end point, in order to know if it has to proceed at that point with another edge curve or start at some other point. This can be encoded by one bit. When this bit signals an endpoint, the decoder knows it does not have to return at that point thereafter. When this bit signals a branch point, the decoder knows firstly that it must expect the chain codes for a new edge curve and secondly, it has to return at that point after the next branch point or end point is encountered. It does not yet know **how many times** it has to return at that point, but, fortunately, there are only two possibilities (because the maximum number of edge curves starting or ending in a branch point is four): return once or twice. This can be encoded by another bit.

4.4 Experimental results

4.4.1 Method

In this section, we show the results of some experiments with the designed edge-based image coder. These results can be used to evaluate the quality of the entire coding scheme, i.e., including the edge analysis, edge parameter coding, edge profile reconstruction and full intensity surface reconstruction. We refer to the entire coding scheme as the *Edge Primitive Coder* (EPC).

In the next subsection, some illustrative results of the edge location coder are presented. Then, results of the entire edge primitive coder are presented. In these experiments, we have used three different grey-level images: "Scarf", "House" and frame three of the "Claire" sequence.

These images were not in the set used to design the Huffman code words of the edge point location coder. The quality of the coder can firstly be evaluated objectively, i.e. in terms of the rates and distortions reported. Also, some of the decoded images have been included to enable subjective judgement by the reader. In the last subsection, a comparison of the edge primitive coder and JPEG is made.

The following symbols occur in one or more of the tables with results in the next subsections.

Concerning edge analysis and edge location coding:

- t_s: The edge detection threshold;
- N_E: The number of edge points detected;
- N_C: The number of edge curves;
- N_L: The number of links;
- B_L: The number of bits to code the links;
- N_S: The number of starting points;
- B_O: The number of bits to code the starting points, branch and end points;
- B_E: The total number of bits to code the edge location data ($=B_L+B_O$);
- R_E: The rate of the edge location coder in bits per pixel (bpp).

Concerning edge parameter signal coding:

- type: The type of parameter signal;
- L: The number of downsampling steps (thus reducing the number of samples by 2^L);
- Q: The number of levels of the quantizer;
- N_P: The number of samples in a (downsampled) parameter signal;
- B_P: The number of bits to code a parameter signal.

Concerning the overall coding performance:

- B_T: The total number of bits to code *all* the data;
- R: The final rate in bits per pixel (bpp);
- C: The compression ratio;
- D: The distortion in the reconstructed edge pixels with respect to the original pixels.

Concerning image intensity reconstruction:

- λ: The regularization parameter;
- K: The maximum scale in each cycle;
- J: The number of cycles applied;
- SNR: The signal-to-noise ratio in dBs;
- PSNR: The peak-to-peak signal-to-noise ratio in dBs.

In the edge analysis stage, the filter scale is $\sigma = 1.0$ and the minimum curve length is $\xi = 4$ in all cases. In the coding stage, the number of prediction coefficients is two in all cases. In the edge profile reconstruction, $f_w = 2.0$ in all cases. In the image intensity reconstruction, the

MS-SOR algorithm with $\omega = 1.0$ is used in all cases. The intensity surface reconstruction algorithm, as discussed in Section 3.3, is applied to an initial approximation of the entire image intensity surface, obtained using the technique described in Section 2.8.2. Here **all** the pixels are given an initial value derived from the (de)coded edge parameters of the nearest edge pixel, but **only** pixels lying in a small band along edge curves are constrained in the reconstruction algorithm, as in Chapter 3.

4.4.2 Results of edge location coding

We have applied the edge location coder to three different test images: Scarf, House and Claire (frame three of the sequence). For each image, the edge selection threshold is set manually to such a value, so that at least some of the features which are believed to be most relevant in the image are detected.

The results are summarized in Table 4.7. The table shows that the chain coder itself spends about 1.4 - 1.9 bits on each link, depending on the overall smoothness of the edge curves. The House image, containing many long, relatively straight, curves has a relatively low coding cost per link. The Scarf and Claire images, on the other hand, contain many, relatively short, curves and are most costly. Including the information for special points, the actual number of bits spent per link is about 1.7 - 2.2 bits. The total number of bits spent for coding the edge locations is about 4000 - 8000, depending on the relative number of edges present in the image. The Scarf image costs the most bits to code in total, because it contains many more links to be transmitted than the other images. This is, of course, due to the value of the detection threshold used in the experiments: the threshold needs to be set relatively low with the Scarf image, in order to preserve relevant features in the face, which have, in this case, low contrast (especially the mouth).

Table 4.7 Results edge location coding for each of the images. For explanation of the symbols in the left column, see Section 4.4.1.

	Scarf	House	Claire
t_s	0.85	0.90	0.94
N_E	3714	2503	1982
N_C	145	59	74
N_L	3653	2469	1926
B_L	6885	3669	3342
B_L/N_L (bit per link)	1.885	1.486	1.735
N_S	85	36	56
B_O	1310	550	820
B_E	8195	4219	4162
B_E/N_L (bit per link)	2.243	1.709	2.161
R_E (bit per pixel)	0.125	0.064	0.041

4.4.3 Results of image coding

In the experiments reported in this section, we try to evaluate the quality of the decoded images using the edge primitive coder given an approximate number of bits (or a rate). As mentioned in Section 4.1, we do not address the problem of bit allocation in this thesis. Instead, for each of the images used, we illustrate the performance of our coding scheme when using three different rate settings: one (relatively) high, one medium and one low. While the rate-distortion results thus obtained are suboptimal (because no bit allocation is applied), they do give a general idea of the quality which can minimally be achieved given a number of bits used to code an image. Again, we have used the three grey-level images Scarf (256x256 pixels), House (256x256 pixels) and Claire (352x288 pixels). These images are shown in respectively Figure 4.4(a), Figure 4.5(a) and Figure 4.6(a).

We accomplish the above by varying the following three parameters: the edge selection threshold (t_s), the number of downsampling steps applied to each parameter signal (L), and the number of representation levels of the DPCM quantizer used (Q). The performance characteristics of each quantizer used are given in Appendix B.2.3. The edge selection threshold is varied little and in such a way that a number of relevant features in the images are preserved.

Objective performance

The data in Table 4.8, Table 4.10 and Table 4.12 summarize the coding results for each of the images. The suffix 'I' is used to denote the settings resulting in the highest bit rate, 'II' to denoted the medium bit rate and 'III' to denote the lowest bit rate. Each of these mentioned tables is accompanied by another table, which summarizes the relevant parameters and the results of the intensity surface reconstruction stage. These tables, which also give the final quality ratings (in terms of signal-to-noise ratios) of the decoded images, are numbered Table 4.9, Table 4.11 and Table 4.13.

From these tables, we can make the following observations.

- Similar remarks as those made in Section 4.4.2 can be made about the edge location coder. The total number of bits necessary to code the edge locations (B_E) is about 3500 - 8000. Both the number of curves (N_C) and the number of links (N_L) are important. Compare, e.g., the results for Scarf I, House I and Claire I: Scarf has about the same number of edge curves as Claire, but the average number of links per edge curve is lower for Claire (20) than for Scarf (25). Therefore, the number of bits used for edge location coding is lower for Claire than for Scarf. The House image contains fewer curves than the Claire image, but the average number of links per curve is much higher (40), therefore the number of links N_L is about equal for both images. At the same time, many more starting points have to be coded for Claire than for House, therefore the total number of bits (B_E) is still higher for the Claire image.
- The number of bits used per link by the location coder (B_E/N_L) is about 2.0.
- An approximately equal number of bits B_P is used to code each of the parameter signals (m, c, and w) in cases where the same number of downsampling steps is applied. The rates at

which the parameters signals are coded (B_P/N_P) lie around 1.5 bits per sample (bps) for the high rate codings and medium rate codings, conforming to the quantizers used. However, for the low rate codings, this rate lies around 2.5 bps. This is due to the heavy downsampling applied in these cases ($L = 4$), which leads to difficulties in the estimation of the auto-correlation (used to set up the predictor and quantizer).

- The total number of bits spent to code the image B_T lies between 15000-25000 bits for the high rate codings, between 8000-15000 bits for the medium rate codings and between 5000-10000 for the low rate codings. This corresponds to about 7 bits per edge point (B_T/N_E bpe), 4 bpe and 3 bpe, respectively.
- Note that the number of bits spent on location coding B_E is relatively high with respect to the total number of bits B_T: about 30% for the high rate codings (with suffix I), about 50% for the medium rate codings (suffix II) and about 70% for the low rate codings (suffix III).
- The final rate R lies between 0.2-0.4 bits per pixel (bpp) for the high rate codings, between 0.08-0.24 bpp for the medium rate codings and between 0.05-0.18 bpp for the low rate codings. This corresponds to a compression ratio C of about 20 to 40 for the high rate codings, 30 to 100 for the medium rate codings and 50 to 150 for the low rate codings.
- The final quality in terms of PSNR is about 30 dB for the high rate codings, about 28 dB for the medium rate codings and about 24 dB for the low rate codings.
- Note that the difference in the PSNR between the high rate codings (suffix I) and the medium rate codings (suffix II) is quite small, while the rate in the medium rate codings is significantly lower than the rate in the high rate codings. While the rate drops further in the low rate codings (suffix III) with respect to the medium rate codings, the quality now drops significantly as well, due to the heavy downsampling. This may signify that the number of steps in the downsampling should preferably be kept low. From a rate-distortion performance point of view, the edge selection threshold may then a better instrument to influence the rate.

Table 4.8 Results edge parameter coding for the Scarf image. The first row gives the value of the edge detection threshold t_s. Then follow: the data on the edge location coder; the data on the edge parameter signal coder; the final data on the coded edge curve representation. For explanation of the symbols in the left column, see Section 4.4.1.

	Scarf I			Scarf II			Scarf III		
t_s	0.85			0.85			0.85		
N_E	3714			3714			3714		
N_C	145			145			145		
N_L	3653			3653			3653		
B_E	8195			8195			8195		
type	*m*	*c*	*w*	*m*	*c*	*w*	*m*	*c*	*w*
L	0	0	0	1	1	2	4	4	4
Q	25	25	25	15	15	15	13	13	13
N_P	3784	3784	3784	1927	1927	1001	398	398	398
B_P	5841	6111	6540	2777	3003	1721	905	1534	980
B_T	26687			15696			11614		
R (bpp)	0.407			0.240			0.177		
C	19.65			33.40			45.14		
D (mse)	62.47			95.09			286.14		

Table 4.9 Results intensity surface reconstruction for the Scarf image. For explanation of the symbols in the left column, see Section 4.4.1.

	Scarf I	Scarf II	Scarf III
λ	0.10	0.10	0.10
K	35	35	35
J	2	2	2
SNR (dB)	14.00	13.01	9.76
PSNR (dB)	28.24	27.25	24.00

Table 4.10 Results edge parameter coding for the House image. The first row gives the value of the edge detection threshold t_S. Then follow: the data on the edge location coder; the data on the edge parameter signal coder; the final data on the coded edge curve representation. For explanation of the symbols in the left column, see Section 4.4.1.

	House I			House II			House III		
t_S	0.88			0.90			0.90		
N_E	2697			2503			2503		
N_C	65			59			59		
N_L	2656			2469			2469		
B_E	4611			4219			4219		
type	*m*	*c*	*w*	*m*	*c*	*w*	*m*	*c*	*w*
L	0	0	0	1	1	2	4	4	4
Q	25	25	25	15	15	15	13	13	13
N_P	2718	2718	2718	1282	1282	658	217	217	217
B_P	4301	4124	4385	1857	1887	991	459	541	468
B_T	17421			8954			5687		
R (bpp)	0.266			0.137			0.087		
C	30.10			58.55			92.19		
D (mse)	185.68			247.69			609.22		

Table 4.11 Results intensity surface reconstruction for the House image. For explanation of the symbols in the left column, see Section 4.4.1.

	House I	House II	House III
λ	0.10	0.10	0.10
K	50	50	50
J	2	2	2
SNR (dB)	17.04	16.03	11.02
PSNR (dB)	29.22	28.22	23.21

Table 4.12 Results edge parameter coding for the Claire image. The first row gives the value of the edge detection threshold t_s. Then follow: the data on the edge location coder; the data on the edge parameter signal coder; the final data on the coded edge curve representation. For explanation of the symbols in the left column, see Section 4.4.1.

	Claire I			Claire II			Claire III		
t_s	0.90			0.94			0.95		
N_E	2965			1982			1689		
N_C	140			74			55		
N_L	2866			1926			1645		
B_E	6576			4162			3410		
type	*m*	*c*	*w*	*m*	*c*	*w*	*m*	*c*	*w*
L	0	0	0	1	1	2	4	4	4
Q	25	25	25	15	15	15	13	13	13
N_P	3005	3005	3005	1019	1019	530	162	162	162
B_P	4244	4623	4817	1448	1593	898	398	648	399
B_T	20260			8101			4855		
R (bpp)	0.200			0.080			0.048		
C	40.03			100.11			167.05		
D (mse)	86.89			191.45			751.24		

Table 4.13 Results intensity surface reconstruction for the Claire image. For explanation of the symbols in the left column, see Section 4.4.1.

	Claire I	Claire II	Claire III
λ	0.10	0.10	0.10
K	75	75	75
J	2	2	2
SNR (dB)	17.44	14.42	9.05
PSNR (dB)	32.35	29.33	23.96

Subjective performance

In Figure 4.4, some images of Scarf are shown, illustrating the coding performance visually. Figure 4.4(a) shows the original Scarf image; (b) shows the edge image of Scarf in exp. II; (c) shows the coded image in exp. I; (d) shows the coded image in exp. II; (e) shows the coded image in exp. III; (f) shows the error image in exp. II, multiplied by 4 and a value of 128 added for visibility. In Figure 4.5, the corresponding images of House are shown. Figure 4.5(a) shows the original House image; (b) shows the edge image of House in exp. II; (c) shows the coded image in exp. I; (d) shows the coded image in exp. II; (e) shows the coded image in exp. III; (f) shows the error image in exp. II, multiplied by 4 and a value of 128 added for visibility. In Figure 4.6, the corresponding images of Claire are shown. Figure 4.6(a) shows the original Claire image; (b) shows the edge image of Claire in exp. II; (c) shows the coded image in exp. I; (d) shows the coded image in exp. II; (e) shows the coded image in exp. III; (f) shows the error image in exp. II, multiplied by 4 and a value of 128 added for visibility.

The edge images are included to give an impression of the relative amount of edge points detected and which features of the original image are represented by the edge primitives. The error images are included to give an impression of the spatial distribution of the errors made by the coding scheme and the relative magnitudes of these errors.

The decoded images shown in Figure 4.4, Figure 4.5 and Figure 4.6 illustrate the subjective quality of the decoded images using the edge-based coding scheme. Certain types of artifacts in these images are visible immediately; others can be noted when comparing the images with their originals. We now list the most important artifacts that can be noted.[a]

- A loss of *texture*. This is, of course, inherent to the edge-based representation. With the Scarf image, the loss of fine grain texture does not seem very important. For the House image, however, the absence of the texture (such as the bricks and roofing-tiles) gives the image an unnatural look. Also, some texture *discontinuities*, which contain some important structural information about the scene, are missing. The Claire image, which contains almost no texture except in the hair, is not hampered much by this type of artifact.
- The apparent absence of certain features (such as the nose in the Scarf image) and the impression of unnatural flatness or dullness of the intensity in certain large regions (such as the right cheek in the Scarf image) render the images somewhat artificial. This effect may be attributed to the fact that small intensity gradients are not detected during edge analysis. A lot of images contain large regions with small gradients only. No intensity information associated with these regions is transmitted; the intensities are interpolated at the receiver.
- Edges along contours have a jagged appearance (see e.g. the right contour of the face in the Scarf image). This is due to the choice of our edge model and the way edge profiles are reconstructed: the edge center always coincides with a pixel position while the actual edge center does not have to lie exactly on pixel positions.

a. Subjective evaluation was performed by viewing the images on a high-definition monitor (Sony Trinitron), at a viewing distance of approximately 1.5 m, under indirect dim lighting conditions.

- Small stripes, or dashes perpendicular to edges, are most noticeable in the Scarf image - along the left contour of the face. These artifacts are often combined with smeared dark or light spots, which might be called "smudges" or small "halos", most noticeable in the Claire image - in the facial area. This may be the effect of the coarse quantization of the parameter signals, particularly the contrast and center intensity values.

Other artifacts which may be noted are the following.

- In some cases, sharp corners are not retained very well. The shape of the corner gets rounded and the intensity level seems distorted.
- Some edges and small spots seem blurred, i.e. not as crisp as in the original.

A positive aspect of the image quality is that the positions and the shapes of most features in the images, such as objects, *are* well retained, thus the intelligibility stays at an acceptable level. Artifacts which are typical for other schemes at low bit rates, such as severe loss of spatial resolution, or annoying horizontal and vertical discontinuities (blocking effect) are not present.

It can be noted that the visual difference between coded images is not very large in the case of Scarf and House, while this visual difference is much larger in the case of Claire.

Discussion of results

Some remarks can be made regarding the performance of the edge-based coding scheme. The edge-based coding scheme is particularly good at coding images in which the relevant features are spatially localized along highly contrasted edge curves. Especially images with a uniform background and sparsely distributed features in the foreground can be coded well, since the scheme does not spend any bits on large regions with uniform intensity. E.g., for the Claire image, the compression ratio lies between 40 and 170 at reasonable subjective qualities, while the PSNR lies between approximately 24 dB and 33 dB. Also, when the edges in an image lie along very long and straight curves, the number of bits spent on edge location coding decreases. Because the latter number is relatively high with respect to the total number of bits spent, the scheme reaches high compression. E.g., for the House image, the compression ratio lies between 30 and 90, while the PSNR lies between approximately 23 dB and 29 dB. For images containing irrelevant high contrast features, and in which the relevant features have low contrast, the edge detection threshold must necessarily be set low. This leads to low compression ratios. E.g., in the Scarf image, many bits are spent on numerous small features in the girl's scarf, while the features in the face are actually more relevant. For this image, the compression ratio lies between 20 and 45, while the PSNR lies between 24 dB and 28 dB.

The loss of texture inherent to the edge-based coding scheme is quite disturbing in certain types of images, but not in others. Especially images with texture discontinuities have an artificial appearance when coded. Reconstructed edges look jagged in most images. A halo effect in some points along edges is present in the decoded images. The spatial shapes and positions of features are retained quite well.

The most relevant experimental results (in terms of bit rate and signal-to-noise ratio) are summarized in Table 4.14. These results are also depicted graphically in Figure 4.7.

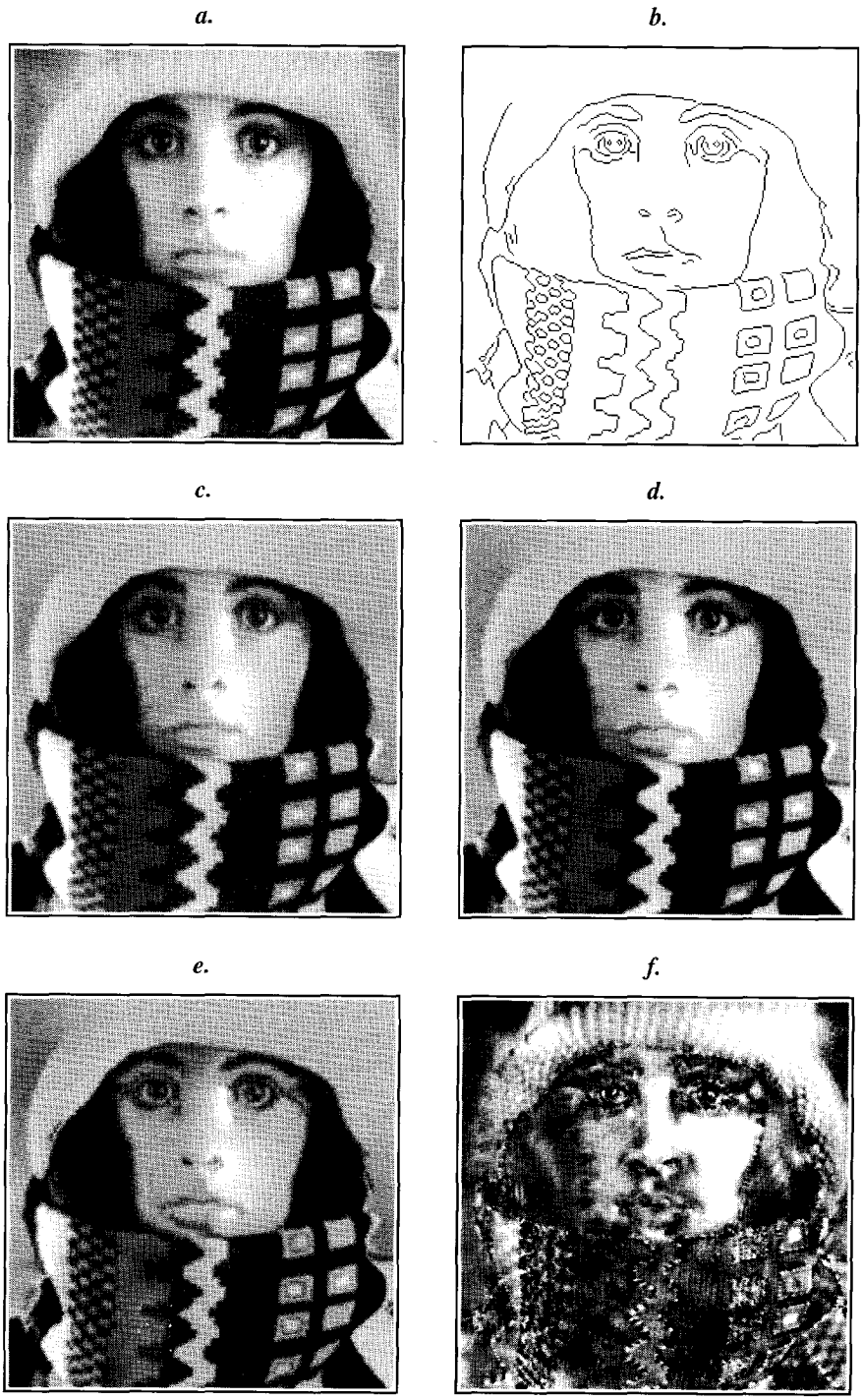


Figure 4.4 (a) - (f) Image coding results for "Scarf" (for description see page 91).

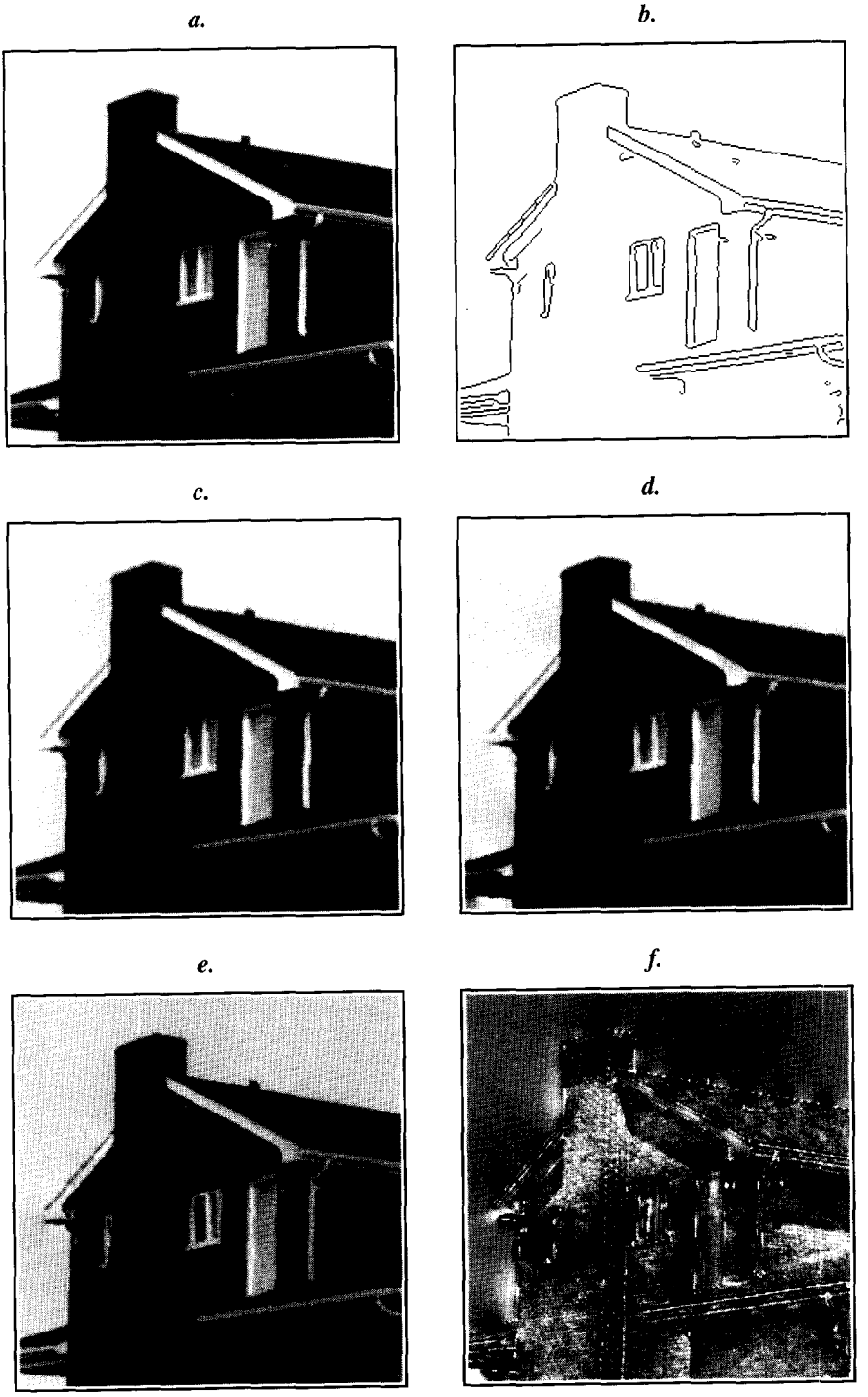


Figure 4.5 (a) - (f) Image coding results for "House" (for description see page 91).

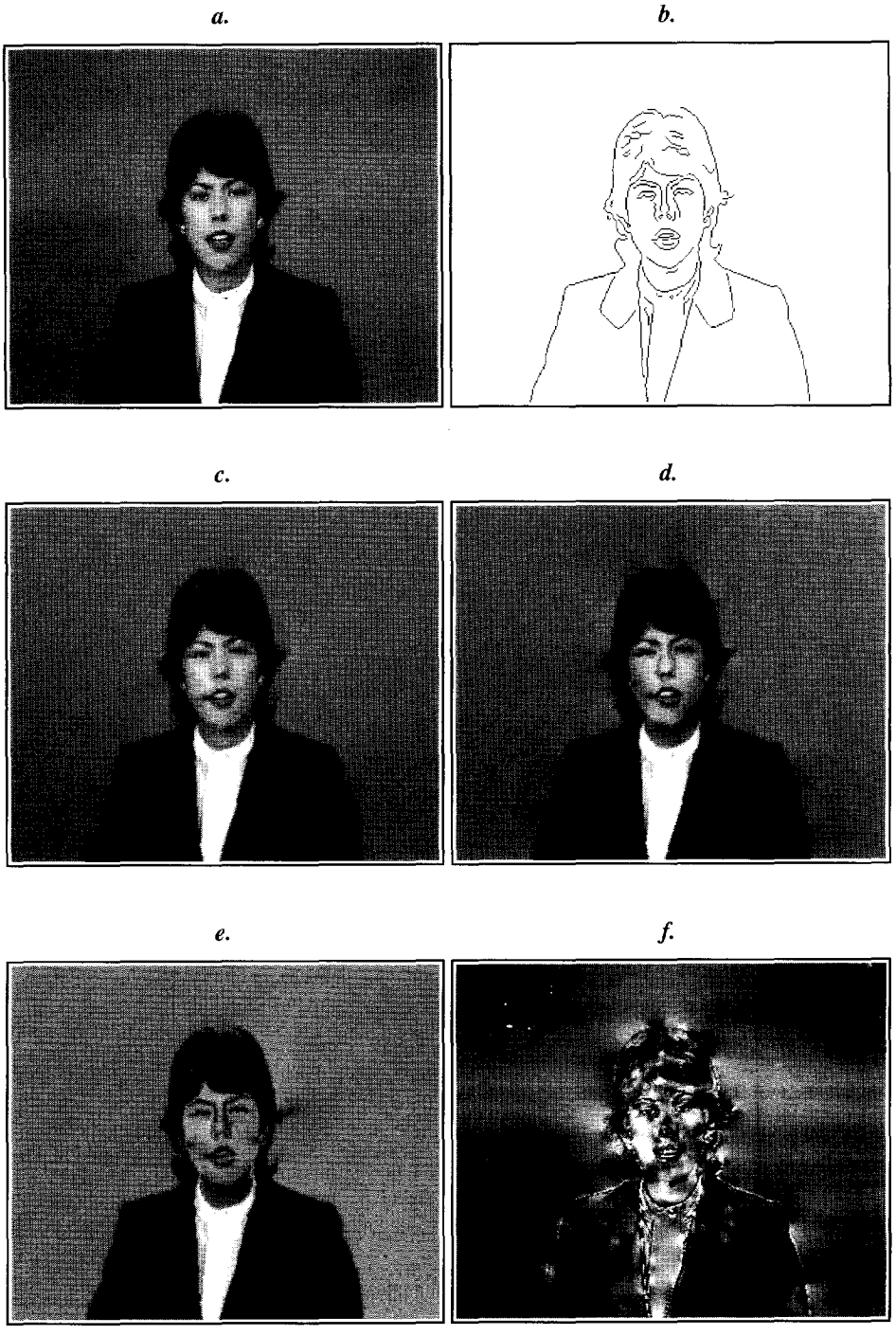


Figure 4.6 (a) - (f) Image coding results for "Claire" (for description see page 91).

Table 4.14 Summary of objective coding performance of the Edge Primitive Coder for each of the quality settings I, II and III and for each of the images.

	Scarf		House		Claire	
	R (bpp)	PSNR (dB)	R (bpp)	PSNR (dB)	R (bpp)	PSNR (dB)
I	0.407	28.24	0.266	29.22	0.200	32.35
II	0.240	27.25	0.137	28.22	0.080	29.33
III	0.177	24.00	0.087	23.21	0.048	23.96

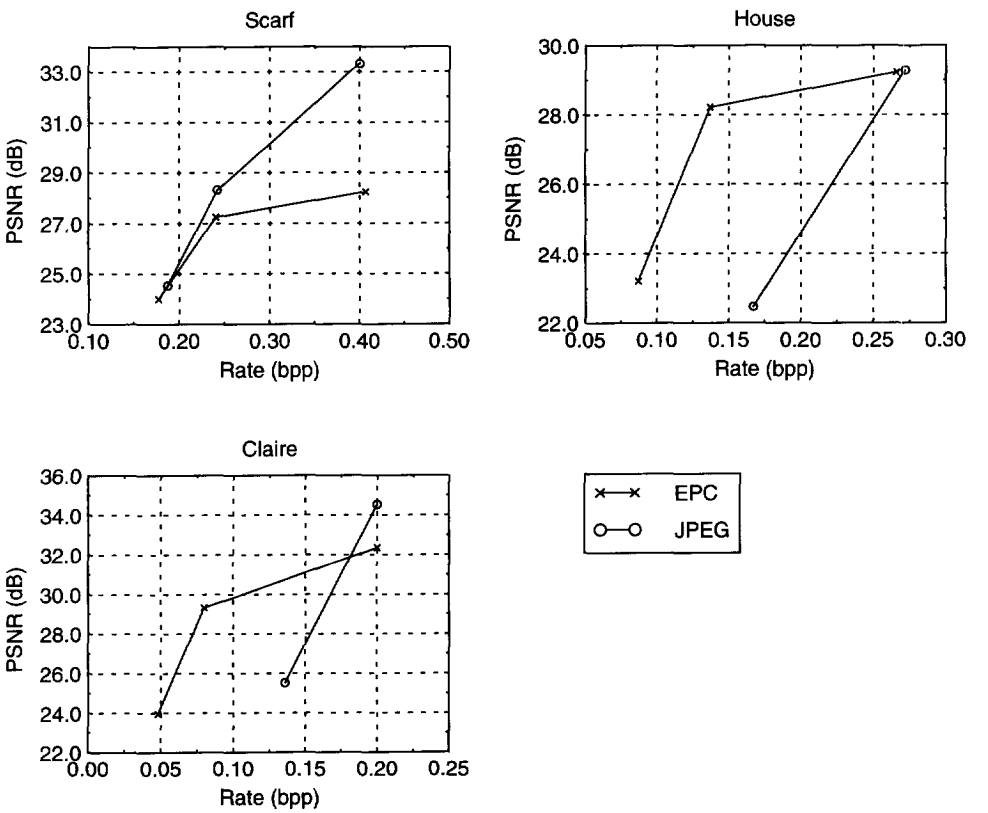


Figure 4.7 Coding performance of the Edge Primitive Coder (EPC) and the baseline JPEG coder for the Scarf, House and Claire images, in terms of PSNR versus bit rate.

4.4.4 Comparison with JPEG

To show the usefulness of the edge-based coding scheme at very low bit rates, in this subsection we make a comparison with the JPEG still image compression standard [123]. To this end, we have JPEG compressed[b] each of the images used (Scarf, House and Claire) to a target bit rate comparable to one of the bit rates produced in the experiments with the edge primitive coder (EPC). This was achieved by setting the JPEG quality factor to an appropriate value. With Scarf, the bit rates for cases I, II and III were used as targets, while for House and Claire only the bit rates for cases I and II could be reproduced with JPEG (lower bit rates could not be achieved). First we compare the performance of the edge primitive coder and JPEG objectively, by their rates and distortions; then we show some of the JPEG coded images to allow subjective evaluation.

The objective coding performance of JPEG, in terms of rate and peak signal-to-noise ratio, is summarized in Table 4.15. This should be compared to the performance of the EPC scheme, summarized in Table 4.14. These results are also depicted graphically in Figure 4.7, together with the results of the edge primitive coder.

Table 4.15 Objective coding performance of JPEG for three different quality settings. Q_{JPEG} is JPEG's quality factor.

Scarf			House			Claire		
Q_{JPEG}	R (bpp)	PSNR (dB)	Q_{JPEG}	R (bpp)	PSNR (dB)	Q_{JPEG}	R (bpp)	PSNR (dB)
17	0.400	33.33	10	0.272	29.28	12	0.200	34.54
6	0.242	28.32	2	0.167	22.47	2	0.136	25.52
3	0.187	24.54	-	-	-	-	-	-

The following conclusions can be drawn from these results. At (relatively) high rates, JPEG outperforms the EPC scheme: for the Scarf image, by as much as 5 dB (PSNR); for the House image, by less than 1 dB; for the Claire image, by approximately 2 dB. At medium rates, JPEG still outperforms EPC for the Scarf image in terms of the PSNR (by approximately 1 dB), but it is inferior to the EPC in the case of the House and Claire images. In the latter two cases, the *quality* difference is about 4 or 5 dB, while the *rate* of JPEG is (in both cases) actually higher than the rate at which the EPC codes these images, because JPEG is already at its minimal quality in these cases ($Q_{JPEG} = 2$). These rates can be considered to be very low in the case of JPEG, while this is not the case for the EPC. Coding the Scarf image at a very low rate (using $Q_{JPEG} = 3$), JPEG can produce a decoded image with approximately the same quality at the same rate as the EPC scheme, tuned to a low rate (case III). The EPC scheme, at quality

b. We have used an implementation of the "baseline" public domain algorithm from the Independent JPEG Group, with standard luminance quantization table for the DCT coefficients and Huffman encoding of the quantized coefficients.

setting III, is able to code the House and Claire images at approximately *half* the *minimum* bit rate of JPEG, producing decoded images with comparable quality!

To illustrate the visual differences between the quality of JPEG and EPC at *medium* bit rates, we have included in Figure 4.8 the images of Scarf and House coded with JPEG at bit rates 0.242 bpp and 0.167 bpp respectively. Blocking is visible in the Scarf image coded with JPEG, and the overall quality might be considered a little worse than the quality of the Scarf image coded with EPC at the same rate (cf. Figure 4.4 (d)). For the House and Claire images, the blocking effect is quite severe, and the quality is distinctively less than that of the EPC coded images at medium bit rate. Compare, for instance, Figure 4.8(b) to Figure 4.5(d). This can be explained by the fact that JPEG spends a lot of bits on overhead and on transmitting low frequency information for blocks in the background. The background contains no features of interest. The edge primitive coder spends the available bits much more spatially localized, at features of interest, i.e. rapidly changing intensity patterns.

One may conclude from the objective as well as visual results that, JPEG outperforms EPC for bit rates higher than 0.20 bpp, while EPC outperforms JPEG for bit rates below 0.20 bpp.

Figure 4.8 (a) Scarf coded with JPEG (with $Q = 6$), the rate is 0.242 bpp and the PSNR is 28.32 dB, cf. Figure 4.4(d); (b) House coded with JPEG (with $Q = 2$), the rate is 0.167 bpp and the PSNR is 22.47 dB, cf. Figure 4.5(d).

4.5 Adaptive coding for very low bit rates

4.5.1 Background

The edge-based image representation consists of a collection of edge primitives C_i, as explained in Chapter 2. These edge curves are obtained by selection of high-contrast edge

points using a single, global, threshold. Only the selected edge primitives are coded and transmitted to the receiver. Better results in image coding may be achieved if a more sophisticated, adaptive, selection mechanism would be used. Because the edge primitives are spatially localized and separated, one can, in principle, select them at will and, eventually, use the selected primitives to reconstruct a certain image. In order to make this selection in a meaningful way, one must have a notion of the relevancy of specific edge primitives and be able to make this notion concrete in the form of some objective criterion.

A notion of perceptual relevance could be the visibility of a local feature for a human observer. The criterion for relevance might then be, e.g., that the average contrast along an edge curve must exceed some minimum value, or that the length of the edge curve must exceed some minimum value. Another criterion might be the saliency of the edge primitive against its local background, since small features stand out in otherwise smooth areas, but tend to be masked in textured areas.

Another kind of notion of relevance is related to the *semantic* content of an image. Certain features are directly related to particular objects which can be highly important in some application contexts. Thus, in certain situations people tend to pay more attention to certain parts of an image being viewed than to other parts. An example is lip-reading by deaf people communicating through the videophone. A typical image may, in this case, contain an indoor background in front of which the torso, arms, shoulders and head of the speaker are visible, but the viewer very likely spends most of his time observing a small region on the display that corresponds to an area near the speaker's mouth. Faithful reproduction of the face, and particularly the mouth area, improves the intelligibility of the auditory speech (see e.g. [107] and [110]). Therefore, in image compression, one may want to reproduce certain regions or features in the image with higher quality than other regions, e.g., by spending more bits on the relevant regions. This kind of spatial adaptivity makes a better perceptual quality of decoded images possible at very low bit rates. It is the topic of this section.

Adapting to the local semantic content of an image can be realized in the edge primitive coder in two ways: either by (a) adaptively selecting edge points during the edge extraction stage, or by (b) varying the number of bits spent on each edge curve during the parameter coding stage. With (a), one could, e.g., vary the edge detection threshold adaptively, such that more edge points are detected in regions of semantic importance, while fewer edge points are detected in other regions. With (b), one could, e.g., tune the downsampling or quantization stages applied to parameter signals lying in specific areas. The first possibility seems more attractive, since it does not require any additional overhead to be transmitted as to the coding quality used for each parameter signal (e.g. kind of quantizer used), as would be the case for the second possibility. Once extracted, the parameter signals can all be coded the same way if desired.

Of course, to automatically identify semantically relevant features in an image, one needs more powerful image analysis tools than discussed in Chapter 2. A thorough discussion of this problem is outside the scope of this thesis; we mention here only that some advances have recently been made in this area. A complete scheme for localizing the facial features in image sequences of head-and-shoulders scenes is proposed in [10] and [102]. Head-and-shoulders

scenes are typical for the videophone, an important application of very low bit rate coding schemes.

The localization scheme is successful at extracting the speaker's face, eyes and mouth in some videophone test sequences, such as "Miss America" and "Claire". Throughout the object localization scheme, one makes use of a priori knowledge about the imaged scene in question (hence the term *knowledge-based* image coding). By matching measured properties of extracted candidate image primitives (such as regions) to their expected values, one can find a particular object searched for. By making use of relations between objects and of regions-of-interest, one can improve the performance of the scheme. For a description of the algorithms and experimental results, we refer to [102]. More about the general object localization problem and a review of facial feature localization algorithms proposed in the past can be found in [103].

4.5.2 Knowledge-based image coding

In this subsection, we illustrate the idea of *knowledge-based coding* by showing how the edge primitive coder can benefit from knowledge about the location of the face region in a facial image. In principle, the edge detection process can spatially adapt the threshold it uses on the basis of this knowledge. However, since this example is for illustration purposes only, we have performed the edge selection process *manually*. For comparison with the results in Section 4.4, the "Scarf" image is used. We proceeded as follows.

Initially, edges were (automatically) detected at a very low (i.e. sensitive) threshold ($t_s = 0.60$), creating an abundance of edge curves (415). Then, certain edge curves, which were judged to be perceptually irrelevant, were manually deleted. Especially edge curves outside the face and short edge curves were deemed perceptually irrelevant. Finally, 116 edge curves were retained, with a total of 3768 edge points. The total number of edge points was fixed such that it approximately equals the number of edge points in the case where the edge detection process is performed completely automatically, with $t_s = 0.85$, as in Section 4.4. In Figure 4.9 (a), the edge image for Scarf with manually selected edge curves is shown. This should be compared to Figure 4.4 (b). Many relevant features in the face are now included, which were not present before. The same result might be achieved by using a variable threshold in the edge detection stage. Following the edge selection, the edge curves with their parameter signals are coded such that the output bit rate is approximately equal to the "Scarf II" case as in the previous section, cf. Table 4.8. The results are presented in Table 4.16 and Table 4.17 and are as follows.

The number of bits spent on edge location coding (B_E) is comparable to the normal case (cf. Table 4.8). The edge parameter signals are all coded identically, by downsampling once and quantizing with $Q = 25$. The total number of bits spent, B_T, and the rate, R, are comparable to case II in Table 4.8, the values are only slightly higher. The compression ratio is slightly lower.

The output (peak-to-peak) signal-to-noise ratio after intensity surface reconstruction is also comparable to case II (cf. Table 4.9). Objectively, the performance of both coders seems the same.

We emphasize again that, because all parameter signals are coded the same way, no overhead needs to be transmitted on account of the adaptive processing: the edge selection stage is used as the only tool for varying the number of bits spent in different regions in the image.

Table 4.16 Results knowledge-based image coding for the Scarf image. For explanation of the symbols in the left column, see Section 4.4.1.

Parameter	Scarf		
N_C	116		
N_L	3681		
B_E	8044		
type	*m*	*c*	*w*
L	1	1	1
Q	25	25	25
N_P	1922	1922	1922
B_P	2913	3244	3440
B_T	17641		
R (bpp)	0.269		
C	29.72		
D (mse)	67.95		

Table 4.17 Results intensity surface reconstruction for the Scarf image. For explanation of the symbols in the left column, see Section 4.4.1.

Parameter	Scarf
λ	0.20
K	35
J	2
SNR (dB)	12.33
PSNR (dB)	27.17

For a visual comparison, the reconstructed image is shown in Figure 4.9(b). This should be compared to Figure 4.4(d). The facial region of the image now has a much more natural look, because some features are better represented (especially the nose and mouth) and because the intensity levels are better preserved. Also, the annoying "halo" effect near some of the edges is much more suppressed. However, some artificial discontinuities are introduced in the face and a higher distortion (disappearance of certain features) in the scarf becomes apparent. Which image is considered better in the end, depends on the application.

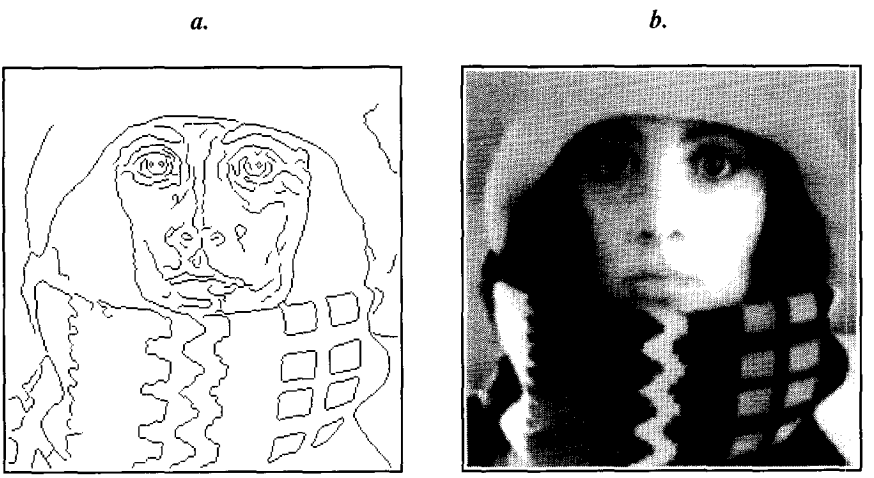


Figure 4.9 (a) Edge image of Scarf with manually selected edge curves; (b) Reconstructed image on the basis of coded edge parameter signals corresponding to the edge curves in (a). The edge image should be compared to Figure 4.4(b). The reconstructed image should be compared to Figure 4.4(d).

4.6 Conclusions

The application of the edge-based image representation to the coding of still images at low bit rates has been discussed in this chapter. The representation consists of: (a) the location of the edge points; (b) the edge model-parameter values in the edge points (center, contrast and width).

Edge location coding

The links between eight-connected edge points on an edge curve are coded in a lossless manner by a variant of chain coding. Huffman variable length encoding is used to compress sequences of such links. Experiments show that the edge location coder spends approximately 2 bits per link and between 4000 and 8000 bits in total (per image) to code the location of edge points (including side information for the starting and end points of curves).

Edge parameter coding

A simple, lossy, coder has been designed, consisting of a downsampling stage and a DPCM stage. Experiments show that the auto-correlation within each parameter signal is high, even when downsampling them by a factor of two or four. On average, the first correlation coefficient is typically above 0.9 for the contrast and center intensity signals, and above 0.8 for the width signals. Consequently, the DPCM coding gain is high. The pdf of the DPCM prediction error samples can be modeled well by a Generalized Gaussian pdf with shape parameter 0.75. Uniform Threshold Quantizers are used to quantize the prediction error samples. Huffman variable length encoding is used to further compress the data.

The parameter signal coder spends approximately 5 bits per edge point if no downsampling is applied, approximately 2 bits per edge point if the parameter signals are downsampled by a factor of 2 and less than 1 bit per edge point if the signals are downsampled by a factor of 16. The corresponding total numbers of bits are: approximately 15000, 5000 and 2000 bits.

Overall results

- The relative number of bits spent on coding the edge point locations is high with respect to the total number of bits spent: between 30% and 70% at different bit rates.
- The final bit rate depends strongly on the image content, but high compression ratios can be obtained. The bit rate ranged between 0.05 and 0.41 bit per pixel (compression ratios between 170 and 20) at quality levels in terms of the PSNR between 23 and 33 dB. See also Table 4.14 and Figure 4.7.
- The primary visual artifacts arising in the decoded images are: a loss of texture, a halo effect and jagged edges. The shape and location of each feature is preserved well.
- When comparing the edge-based coder to the baseline JPEG coder, one may conclude that JPEG outperforms the edge-based coder at bit rates higher than 0.20 bpp, while the edge-based coder outperforms JPEG at bit rates lower than 0.20 bpp. See also Table 4.14, Table 4.15 and Figure 4.7.
- The edge-based coder is especially good at coding images in which the primary features are spatially localized in long, straight lines and images which contain little texture.

Adaptive coding

Depending on the viewing context, certain objects in a scene should be represented in an image more faithfully than others. This can be accomplished quite easily in the edge-based coder if the edge detection stage can be adapted appropriately.

Further research

Using *arithmetic coding* instead of Huffman coding in both the edge location coder and edge parameter coder may improve the coding performance. Development of a bit allocation algorithm will lead to a better distribution of available bits between components in the representation and thereby to a better image quality.

Chapter 5
Image Sequence Coding

This chapter is on the coding of image sequences (*video*), using the edge-based image representation introduced in Chapter 2 and Chapter 3. The methods described in this chapter extend the algorithms described in Chapter 4, which was on *still* image coding.

5.1 Introduction

5.1.1 Problem formulation

Coding techniques for image sequences exploit the temporal correlations between successive images (*frames*) in the sequence, besides exploiting the spatial correlations within each image, as is already being done in still image coding. Employing both techniques often leads to higher compression ratios. The usual approach is to use information about the previous frame(s) to predict the current frame. Normally, the prediction error can be coded much more efficiently than the original signal. Very often, some form of motion estimation is used to obtain an accurate prediction, e.g. block-matching. (For an overview of motion estimation techniques, see [1] and [36].) The motion information, called *motion vectors*, must be transmitted as side information. This approach has resulted in the development of video coding standards like H.261 [28] and MPEG [55], in which the temporal processing (in casu, motion-compensated prediction) precedes the spatial processing (in casu, decomposition based on the discrete cosine transform).

The edge-based scheme uses a decomposition of an image into edge primitives, performed by the edge analysis process. These edge primitives are inherently defined in the original spatial domain and cannot be defined identically in the prediction error domain. Hence, in our case, the spatial processing precedes any temporal processing. The edge analysis is applied to each original frame in a sequence. Then, in the edge-based coding scheme, there are two types of

information to be transmitted: (a) edge point *locations*, or, equivalently, curve *shape* information; (b) edge *parameter* information, related to the *grey levels* along edge curves in the original image.

For both sources, temporally predictive or *interframe* coding techniques can be developed. This means that for edge points in the current frame, corresponding edge points in the previous frame(s) must first be found, which can then be employed to compute predictions. This is commonly called the *correspondence problem*. Since it is very inefficient to transmit motion information for each edge *point* and probably too complex to find corresponding edge *points*, we chose to define the problem at the edge *curve* level. We try to find corresponding edge *curves*. This is the *edge primitive matching* problem. The problem is naturally set on an *intermediate* level, therefore no *low-level* techniques (directly on the raw image data) are applied and neither are any *high-level* techniques (object-based). Application of intermediate-level matching techniques for motion estimation in image sequence coding is quite rare, due to the complexity of general types of scenes and the computational effort usually involved.

The performance of the edge primitive matching stage is of paramount importance to the overall performance of the image sequence coding scheme. Only after corresponding edge curves are found, can predictive coding take place. In this thesis, we restrict ourselves to the predictive coding of the edge point *location* source. Predictive coding of one-dimensional signals such as the parameter signals along the curves in the edge-based representation is conceptually and computationally very similar to the motion-compensated coding of two-dimensional images as applied in many video coding systems which have been proposed in literature (e.g. [36]). Interframe coding of edge point *locations* (as well as of region *contours*) is a relatively novel problem and has been investigated only scarcely in the past (e.g. [27]). Thus, predictive coding of photometric parameter signals is not studied here.

The general strategy proposed for interframe coding of edge primitives is quite simple. For each edge primitive in the current frame, one tries to find a corresponding edge primitive from the previous frame using a matching algorithm. When the matching stage is able to find a corresponding edge primitive, then the current edge curve is coded predictively (**inter**frame mode). Otherwise, the edge curve is coded directly (**intra**frame mode).

The main questions addressed in this chapter are:

- Can edge curves from different images from an image sequence be successfully matched, in order to compute their motion and apply predictive coding?
- Does (temporally) predictive coding of the shape of edge curves lead to a higher compression ratio than direct coding?

5.1.2 Outline

The edge primitive matching problem is discussed in Section 5.2. In the matching, both geometric and photometric properties of the edge primitives are taken into account. An example of the performance of the proposed matching algorithm is included. Interframe edge point location coding is discussed in Section 5.3. Novel techniques for lossless as well as lossy predictive chain coding are proposed. Experimental results on real image sequences are

presented in Section 5.4 to illustrate and evaluate the proposed techniques. The chapter ends with conclusions.

5.2 Edge primitive matching

5.2.1 Description

Feature matching is a problem which has received widespread attention in the computer vision field [39]. Matching algorithms are applied very often in stereo- and motion-analysis problems. In matching, it is important that the essential attributes of the entities being matched are taken into account. The matching algorithm proposed here is based on a *similarity* measure $\Phi(.,.)$, which determines how well two arbitrary edge primitives match to each other. One finds, for each edge primitive from one image, the corresponding edge primitive from another image by seeking a primitive with maximum similarity to the first primitive.

Edge primitive or edge curve u from frame i is denoted by C_u^i. The geometric shape of its curve is represented by a starting point (xs_u^i, ys_u^i) and a sequence of links $l_u^i(n)$. If N_u^i is the number of edge points on the curve, then $1 \le n \le N_u^i - 1$ for an open curve. The photometric properties of the edge primitive are its parameter signals $c_u^i(n)$, $w_u^i(n)$ and $m_u^i(n)$, with $0 \le n \le N_u^i - 1$. Similarly, another edge primitive or edge curve v from frame j, $j<i$, is denoted by C_v^j.

We define $\Phi(.,.)$ such that $0.0 \le \Phi(C_u^i, C_v^j) \le 1.0$, where $\Phi(.,.) = 1.0$ means a perfect match and $\Phi(.,.) = 0.0$ means a total mismatch. The matching strategy is quite simple and is as follows. For each edge curve C_u^i in the current frame i, find the curve $\tilde{C}_v^j$ in a previous frame j, with maximum $\Phi(C_u^i, C_v^j)$ among all curves C_v^j. If the matching is successful, C_u^i is called the *matched curve* and $\tilde{C}_v^j$ is called the *matching curve*. Note that we do not force uniqueness of the match or apply any relational constraints.

We only try to match curves with $N_u > 8$. This is because very short curves may produce unreliable matches and, from a coding point of view, it is inefficient to code very short curves in interframe mode, since the overhead becomes too high. Further, we require that the starting points of candidate matching curves lie within a certain range: $-8 < xs_u^i - xs_v^j \le 8$ as well as $-8 < ys_u^i - ys_v^j \le 8$. This is because these differences, which represent the displacement vectors of the starting point, must be coded with a limited number of bits. This constraint can be justified by assuming that the motion of objects between successive frames is limited.

Our similarity measure takes into account both *shape* (geometric) and *grey-level* (photometric) properties of the edge primitives. Shape similarity, grey-level similarity, and their combination are discussed in the following subsections.

5.2.2 Shape similarity

Literature on shape matching and shape analysis problems is quite abundant. A theoretical framework for geometrical shape analysis and shape similarity measures, providing many references to earlier work, is formulated by van Otterloo [90]. These measures are based on a variety of related parametric representations, and are invariant to curve position, orientation

and scale. Two curve matching schemes, related to the former work, are proposed by Arkin et al. [6] and Kamgar-Parsi et al. [62]. These papers are concerned with exactly computing a metric **dis**similarity function between two polygonal curves with low combinatorial complexity algorithms. Here, we adopt the formalism developed by van Otterloo and use a dissimilarity function equivalent to the ones used by Kamgar-Parsi et al. [62] and Koch and Kashyap [66]. In the following, we omit the superscript denoting the frame number, where possible.

Curve representation

A continuous curve in the 2-D plane can be parametrically represented by its *position vector function*

$$\mathbf{z}(t) = x(t)\ \mathbf{i}_x + y(t)\ \mathbf{i}_y \tag{5.1}$$

where $\mathbf{i}_x$ is the unit vector in the x-direction and $\mathbf{i}_y$ is the unit vector in the y-direction and $0 \le t \le L$, L being the *arc-length* or *perimeter* of the curve. Points $(x(t), y(t))$ on the curve are ordered according to the arc-length parameter t. Suppose one wants to measure the (dis)similarity between two curves, C_u and C_v, with position functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ and arc-lengths L_u and L_v, respectively, as illustrated in Figure 5.1. If C_u and C_v both correspond to the same three-dimensional (3-D) curve in the original scene, undergoing a *rigid* motion and viewed at two time instances, the relationship between $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ can be defined through a transformation, which may include 3-D structure parameters, 3-D motion parameters and 3-D-to-2-D projection parameters. Here, we use a model allowing only translation, rotation and scaling in the 2-D plane. Also, we minimize the amount of scaling necessary, assuming that curves in subsequent frames have approximately the same scale (considering the type of image sequences handled here).

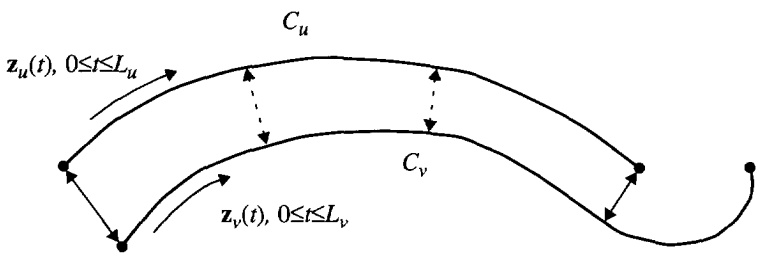


Figure 5.1 Matching two curves C_u and C_v with position vector functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ in the 2-D plane. In this case, $L_u < L_v$, so that actually $\mathbf{z}_u(t)$ is matched only with part of $\mathbf{z}_v(t)$.

Here, the continuous position functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ are formed from the edge curves by considering the chain-coded links, connecting edge points on the discrete image grid, to be small *straight line segments*. These segments have length 1.0 (horizontal or vertical link) or $\sqrt{2.0}$ (diagonal link). Thus, C_u and C_v can be considered to be *polygonal* curves. Also, $\mathbf{z}_u(0)$

simply coincides with one of the end points of C_u and the arc-length L_u is simply the sum of the lengths of the individual links along the curve (and likewise for $\mathbf{z}_v$).

Our dissimilarity functional is defined only on curves of equal arc-length. However, corresponding curves taken from different frames seldom have exactly equal length. To obtain two curves of equal length, first, an equal number of links is taken from both curves. That is, one simply uses only $\min\{N_u - 1, N_v - 1\}$ links (for open curves) from both curves to form the position functions. Subsequently, one applies a scaling to the second position function $\mathbf{z}_v(t)$ to eliminate any remaining length difference. The resulting arc-length is denoted by L. This is illustrated in Figure 5.1 by the fact that C_u is shorter than C_v and C_u is aligned with the beginning sub-curve of C_v. We show later how the matching procedure can be made invariant with respect to the particular choice of sub-sequence taken from the sequence of links.

Dissimilarity measure

First, we define a **dis**similarity functional as the *minimum* mean-squared distance between corresponding points along two curves, over all possible rotations and translations. Corresponding points are points at the same arc-length, i.e., equally far from the beginning of each curve. The dissimilarity functional $Q(.,.)$ is defined as:

$$Q(\mathbf{z}_u, \mathbf{z}_v) = \min_{\varphi, \Delta x, \Delta y} \frac{1}{L} \int_0^L \left\| \mathbf{z}_u(t) - (\mathbf{R}\mathbf{z}_v(t) + \mathbf{T}) \right\|^2 \, \mathrm{d}t \,, \tag{5.2}$$

with $\mathbf{R}$ a rotation matrix, rotating $\mathbf{z}_v$ by φ around the origin, and $\mathbf{T}$ a translation vector, translating $\mathbf{z}_v$ over Δx in the x-direction and Δy in the y-direction:

$$\mathbf{R} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \,, \qquad \mathbf{T} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \,.$$

The higher the value of $Q(.,.)$, the more *dissimilar* are the two curves C_u and C_v. In earlier research [9], we found this dissimilarity measure to be relatively insensitive to noisy distortions of the curve shapes.

To realize the efficient computation of $Q(.,.)$, first, the straight line segments of both C_u and C_v must be broken up further into N corresponding straight line segments, such that corresponding line segments have equal length p_n and

$$\sum_{n=1}^{N} p_n = L \,.$$

An example of this decomposition of two polygonal curves with respect to each other is given in Figure 5.2(a). The corresponding position functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ now consist of N components (each corresponding to a straight line segment):

$$\mathbf{z}_{u,n}(t) = x_{u,n}(t)\,\mathbf{i}_x + y_{u,n}(t)\,\mathbf{i}_y \quad , \quad 1 \le n \le N,$$
$$\mathbf{z}_{v,n}(t) = x_{v,n}(t)\,\mathbf{i}_x + y_{v,n}(t)\,\mathbf{i}_y \quad , \quad 1 \le n \le N.$$

The functions $x_n(t)$ and $y_n(t)$ of a straight line segment (omitting for a moment the subscript denoting the curve number) can be expressed in terms of the end points of such a segment and its length, e.g.,

$$x_n(t) = \frac{x_n - x_{n-1}}{p_n} \cdot t + x_{n-1}, \qquad y_n(t) = \frac{y_n - y_{n-1}}{p_n} \cdot t + y_{n-1}, \tag{5.3}$$

with $0 \le t \le p_n$ for each line segment, as shown in Figure 5.2(b). Alternatively, these functions can be expressed in terms of the center point, angle with the x-axis and length of a segment:

$$x_n(t) = a_n + t\cos\theta_n, \qquad y_n(t) = b_n + t\sin\theta_n, \tag{5.4}$$

with $-p_n/2 \le t \le p_n/2$ for each line segment, again shown in Figure 5.2(b).

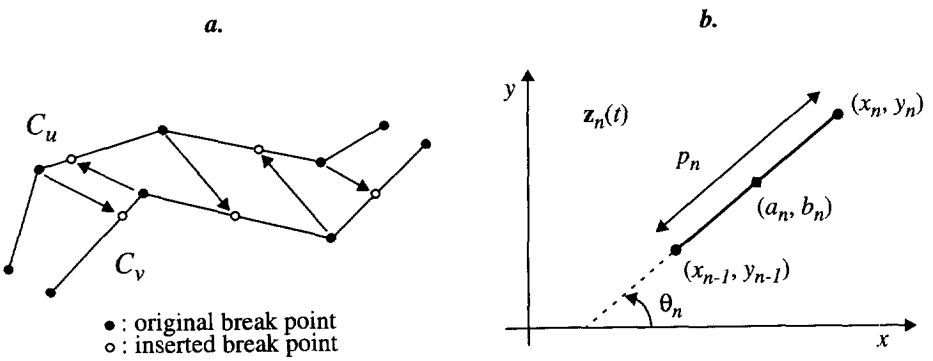


Figure 5.2 (a) Decomposition of C_u (originally consisting of four segments) and C_v (originally consisting of three segments) with respect to each other, by inserting dummy break points such that both curves contain an equal number of line segments (now six), and corresponding line segments have equal length; (b) A straight line segment and its parameters.

Substituting either Eq. (5.3) or Eq. (5.4) in Eq. (5.2) leads, after integration over t (along each line segment), to a discrete sum for $Q(.,.)$ of N terms. The minimum value of $Q(.,.)$ can now be derived *analytically* by equating the partial derivatives with respect to φ, Δx and Δy to zero. Using Eq. (5.3), this leads to a result equivalent to the one derived by Koch and Kashyap (see [66]). Using Eq. (5.4), this leads to the result derived by Kamgar-Parsi et al. (see [62]). For the resulting optimal translation and rotation parameters, φ^*, Δx^* and Δy^*, we refer to these papers and do not repeat them here. They are expressed in terms of *moments* and *cross-moments* of the line segment parameters. We use the formulae given by Kamgar-Parsi

et al., leading to the *unique* minimum of $Q(.,.)$. The optimal translation and rotation parameters and the minimum of $Q(.,.)$ can be calculated in time complexity $O(N)$, i.e. in time proportional to the number of links in the curves.

Invariance to starting and end points

Thus, we now have defined a dissimilarity measure $Q(.,.)$, which is both mathematically and computationally attractive. Note that $Q(.,.)$, although invariant to rotations and translations, still depends on the choice of starting and end points of the sub-sequence of links taken from the curves, and to the order in which these links are traversed. The order of traversal of the curve links is a priori unknown and may change over subsequent frames due to slightly changing orientations. The order of traversal as well as the number of links on the curve may be different for two given curves, even when these curves correspond to the same physical object in the scene. Therefore, the measure has to be made invariant with respect to the choice of the starting points of the curves. In general situations, the solution might be to match the curve C_u to all possible sub-curves of C_v with the same number of links as in C_u. This could be done by "sliding" C_u along C_v, and reversing C_u as well, computing $Q(.,.)$ for each "shift", and remembering the overall minimal match. However, preliminary experiments showed that this sliding procedure produced optimal matches with *nonzero* shift only in a small percentage of all the matches found [120]. Also, including the possibility of a nonzero shift slightly increased the bit rate necessary to code the curve shapes, using the final interframe shape coding scheme. Still, it seems at least necessary to account for a possibly reversed order of traversal of the links on the curves and for the possibly different number of links of the curves. Hence, we only allow for four different ways to form the position vector functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$ from the links of the curves:

a. the starting point of C_u is aligned with the starting point of C_v and both curves are traversed in their original order;

b. the starting point of C_u is aligned with the end point of C_v and the second curve is traversed in reverse order;

c. the end point of C_u is aligned with the end point of C_v and both curves are traversed in reverse order;

d. the end point of C_u is aligned with the starting point of C_v and the first curve is traversed in reverse order.

The problem can be understood visually by observation of Figure 5.1. Note that the original digital curves C_v and C_u could have started in either end point on those curves. So, to match the corresponding pieces of the curves as drawn in the graph, one may have to reverse the parameterization of one or of both the curves **before** forming the position functions $\mathbf{z}_u(t)$ and $\mathbf{z}_v(t)$.

Final shape similarity

From the curve **dis**similarity measure $Q(.,.)$, which is not bounded in principle, we want to calculate a curve *shape similarity measure* $\Phi_s(.,.)$, with $0.0 \le \Phi_s(.,.) \le 1.0$. We have used

$$\Phi_s(\mathbf{z}_u, \mathbf{z}_v) = \frac{1.0}{1.0 + \dfrac{Q(\mathbf{z}_u, \mathbf{z}_v)}{Q_0}} \tag{5.5}$$

to map $Q(.,.)$ into $\Phi_s(.,.)$. Currently, the normalizing factor $Q_0 = 5.0$. The choice of normalizing factor is arbitrary as far as the ordering of the similarity values $\Phi_s(.,.)$ is concerned, but it was chosen empirically in such a way that the resulting values can be somewhat appreciated intuitively. Values of $\Phi_s(.,.)$ close to 1.0 signal high similarity, whereas values close to 0.0 mean low similarity.

One may assume that the rotation of curves taken from two subsequent frames is limited and that the translation is limited as well. Therefore, we only accept a candidate match between two curves C_u and C_v, if the translation and rotation parameters computed during calculation of $Q(.,.)$ are bounded. First, we recompute these parameters such that the effect of the translation and rotation is separated and that the value of φ^* falls between $-\pi$ and π. Then, we apply the following restrictions:

$$|\varphi^*| \leq \frac{\pi}{8}, \qquad |\Delta x^*| \leq 16, \qquad |\Delta y^*| \leq 16.$$

Otherwise, $\Phi_s(C_u, C_v)$ is set to 0.0.

Finally, since we want to penalize matching two curves with very different numbers of links, we also compute a *length similarity measure* $\Phi_l(.,.)$, defined as follows (for open curves):

$$\Phi_l(N_u, N_v) = \left(\frac{\min\{N_u - 1, N_v - 1\}}{\max\{N_u - 1, N_v - 1\}} \right)^{1/2} \tag{5.6}$$

Both the shape similarity and length similarity measures are used in the definition of the overall similarity measure, as is explained in Section 5.2.4.

5.2.3 Grey-level similarity

The grey-level or photometric similarity between two edge primitives C_u and C_v can be determined using the parameter signals associated with these primitives, such as the contrast, width and center intensity signals along the curves. We simply use a cross-correlation measure to estimate this similarity. The photometric similarity $\Phi_p(.,.)$ between two parameter signals f_u (of length N_u) and f_v (of length N_v) is defined as follows:

$$\Phi_p(f_u, f_v) = \frac{\sum_{n=0}^{N-1} f_u(n) \cdot f_v(n)}{\max\left\{ \sum_{n=0}^{N-1} (f_u(n))^2, \sum_{n=0}^{N-1} (f_v(n))^2 \right\}}, \tag{5.7}$$

with $N = \min\{N_u, N_v\}$. The time complexity of this similarity computation is $O(N)$.

Again, four different alignments of curves C_u and C_v are used, corresponding to the alignments discussed previously in Section 5.2.2. In the current matching algorithm, only the edge contrast (c) and center (m) signals are used, where we have assumed that the edge width (w) signal is less relevant. That is, we expect that the latter does not discriminate between different curves as well as the former types of signals.

5.2.4 Overall similarity

The overall similarity measure $\Phi(C_u, C_v)$ between two edge primitives C_u and C_v is formed by multiplication of the shape and photometric similarity measures, defined by Eq. (5.2), Eq. (5.5), Eq. (5.6) and Eq. (5.7). Thus,

$$\Phi(C_u, C_v) = \Phi_s(\mathbf{z}_u, \mathbf{z}_v) \cdot \Phi_l(N_u, N_v) \cdot \Phi_p(m_u, m_v) \cdot \Phi_p(c_u, c_v) \ . \tag{5.8}$$

It holds that $0.0 \leq \Phi(C_u, C_v) \leq 1.0$. Values of $\Phi(.,.)$ close to 1.0 signify high similarity between edge primitives, whereas values close to 0.0 signify low similarity.

To eliminate bad matches, the similarity $\Phi(C_u^i, \tilde{C}_v^j)$ between an edge primitive and its matching counterpart is set to 0.0 if it lies below $\Phi_{\min}$. Currently, we use the value $\Phi_{\min} = 0.20$, found empirically.

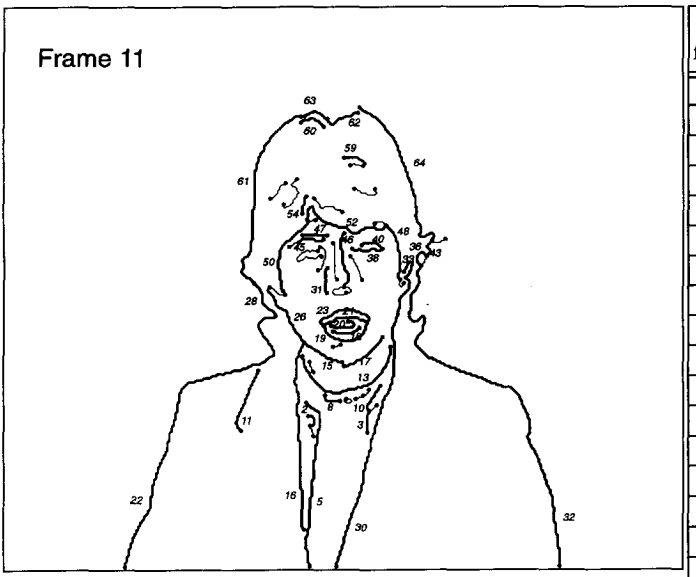
5.2.5 Example

As an illustration of the performance of the edge primitive matching, we show the results of the algorithm applied to the edge primitives in two frames of the "Claire" sequence in Figure 5.3. The matching algorithm was applied in the context of the interframe edge primitive coder, as further explained in the following sections. In Figure 5.3(a), the original edge curves of frame eleven are displayed. In (b), the edge curves of frame nine are displayed. The edge primitives of frame eleven, C_u^{11}, are matched to the coded and decoded edge primitives of frame nine, C_v^9. The edge curve shapes in frame nine were coded lossless, while the edge parameter signals were coded lossy (as in Chapter 4). Curves which were matched are drawn in thicker lines than curves which could not be matched. Matched curves, in (a), and matching curves, in (b), are numbered. The correspondence between the numbered curves can be looked up in the table in Figure 5.3(c), which also specifies the similarity values.

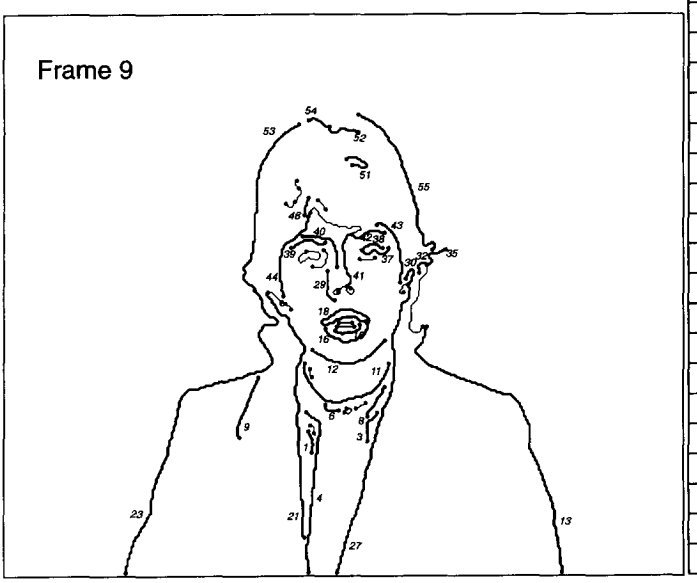
In this example, 39 out of 64 curves in frame eleven were matched (about 61%). Frame nine contained 55 curves. The number of matched edge points in the matched curves is approximately 1400 from a total of approximately 1700 (about 82%). In most instances of the 25 curves which were not matched, the number of points on the curves was considered too low and matching was simply not attempted. The other reason for not finding a match was that the similarity with other curves in the vicinity was too low (remember that $\Phi_{\min} = 0.20$).

Of the 39 matched curves from frame eleven, only 3 can be considered mismatches upon inspection: curve 2 (matched to 1, because the corresponding curve was too short), 20 (matched to 15) and 60 (matched to 54, because no corresponding curve was available). Therefore, the percentage of curves matched *successfully* is about 56 in this case.

a.



b.



c.

curve frame 11	curve frame 9	similarity
2	1	0.391
3	3	0.742
5	4	0.976
8	6	0.913
10	8	0.812
11	9	0.801
13	11	0.914
15	12	0.648
16	21	0.342
17	12	0.643
18	15	0.842
19	16	0.833
20	15	0.558
21	18	0.580
22	23	0.887
23	18	0.683
26	21	0.462
28	23	0.293
30	27	0.963
31	29	0.758
32	13	0.824
33	30	0.796
36	32	0.602
38	37	0.715
40	38	0.880
43	35	0.645
45	39	0.801
46	41	0.817
47	40	0.587
48	43	0.837
50	44	0.794
52	42	0.402
54	48	0.759
59	51	0.677
60	54	0.481
61	53	0.898
62	52	0.671
63	54	0.728
64	55	0.801

Figure 5.3 Matching result for Claire sequence: (a) edge curves from frame 11, matched curves are numbered and drawn with thick lines; (b) edge curves from frame 9, matching curves are numbered and drawn with thick lines; (c) table showing edge curve correspondence and similarity values. A frame of the Claire sequence can be found in Chapter 4 for comparison.

5.3 Edge point location coding

5.3.1 Description

Once the edge primitive matching is done, coding of the edge point location information (stored in the sequences of chain-coded links) can be done. Each edge curve is coded separately[a]. The decision as to whether **intra**frame coding is applied on a curve or **inter**frame coding, depends entirely on the results of the matching stage. When, for a curve in the current frame, a match has been found with a curve in a previous frame, the current curve is coded in **inter**frame mode. Otherwise, it is coded in **intra**frame mode. For each curve, **one** bit must be transmitted prior to any other information, signifying whether that curve is coded in intraframe or interframe mode. In the following subsections, we describe intraframe coding and propose novel methods for both lossless and lossy interframe coding of sequences of chain-coded links. A summary of the information transmitted in both the intraframe mode and interframe mode is given in Table 5.1 and Table 5.2.

5.3.2 Intraframe coding

Coding of the edge location information of a curve in **intra**frame mode is very similar to the method described in Chapter 4. Curves in intraframe mode are coded lossless. First, for each edge curve, the starting point coordinates are transmitted. Here, we assume these can be coded using **twelve** bits. The sequence of chain-coded links are treated as a second-order Markov chain (except for the first link, which is treated as a zero-order chain, and the second link, which is treated as a first-order chain). Nine-symbol Huffman codes are designed, based on experimentally measured frequency distributions of the eight chain codes and an end-of-chain symbol[b]. The links are coded using the Huffman code words.

5.3.3 Lossless interframe coding

Curves in the current frame (denoted by C_u^i) for which a match was found, can be coded by using information from the matching curve from a previous frame (denoted by $\tilde{C}_v^j$, $j<i$) as a prediction for the information which must be transmitted for the current curve. In general, this would include the optimal coordinate transform (in our case, a rigid rotation and translation) taking the matching curve to the prediction of the current curve, and some kind of location prediction error information. However, preliminary experiments showed that including *both* the rotation and translation in this transform, does not necessarily lead to a better prediction than including *only* a translation in the transform [120]. This being the case and taking into account that the rotation information can pose a significant overhead, we do not use the rotation information here. Instead, we only transmit translation information in the form of the

a. For simplicity, we do not use the efficient coding of special starting or end points (like branch points) as proposed in Chapter 4. Thus, a starting point is transmitted for *every* edge curve.

b. We use a *separate* end-of-chain symbol to signal the termination of a chain, as opposed to the technique used in Chapter 4. This is because no restrictions can be formulated on interframe coded link sequences, as will become apparent in the next subsection.

differences of the x- and y-coordinates of the starting points of curve C_u^i with $\tilde{C}_v^j$. These displacement vectors are:

$$vx_u^i = xs_u^i - \tilde{xs}_v^j \qquad \text{and} \qquad vy_u^i = ys_u^i - \tilde{ys}_v^j \ .$$

Since curves are only taken into account during the matching when these quantities lie between -7 and 8, the displacement vectors can be coded using **four** bits each.

To signal to the decoder *which* curve from the previous frame must be used in the prediction, the *label* or number of that curve (v) must be transmitted. We assume **six** bits are enough to code this information.

Also, to signal to the decoder how the two curves must be aligned, **two** bits are needed (there are four possibilities, see Section 5.2.2).

Finally, prediction error information needs to be transmitted. The sequence of chain-coded links of $\tilde{C}_v^j$ can be used as a prediction of the sequence of chain-coded links of C_u^i. The former sequence is denoted by $l_u^i(n)$, the latter by $\tilde{l}_v^j(n)$. In general, the lengths of these sequences are not equal. Therefore, only $\min\{N_u^i - 1, N_v^j - 1\}$ links of $\tilde{C}_v^j$ can actually be used as predictions for the same number of links of C_u^i. A prediction error is simply formed - link by link - by subtracting (modulo 8) the prediction links from the original links:

$$\Delta l_u^i(n) = \left(l_u^i(n) - \tilde{l}_v^j(n) + 8 \right) \bmod 8, \qquad 1 \le n \le \min\{N_u^i - 1, N_v^j - 1\} \ .$$

The sequence of prediction error links is terminated by an end-of-chain code.

This technique is called *interframe* or *differential* chain coding. The notion underlying this technique is that the sequence of prediction error links $\Delta l_u^i(n)$ contains many zeros and hopefully has a lower entropy than the original link sequence (due to the high shape similarity between C_u^i and $\tilde{C}_v^j$). The prediction error sequence is treated as a second-order Markov chain as well. A *separate* code book with Huffman codes is designed for the prediction error links, based on experimentally measured frequency distributions.

When $N_v^j < N_u^i$, some chain-coded links remain to be transmitted. These are simply coded in an intraframe fashion. Thus, links $l_u^i(n)$ with $N_v^j \le n \le N_u^i - 1$ are coded using the intraframe Huffman code book. These are terminated using an end-of-chain code as well.

The decoder can reconstruct a coded curve from the current frame by the following steps:

a. Check the classification bit to see whether the curve is coded intra- or interframe;

b. If the curve is coded interframe, get the matching curve from a previous frame (stored in a curve memory) using the curve label v;

c. The starting point of the curve can either be found directly (intraframe curve) or (interframe curve) by adding the displacement vectors vx_u^i and vy_u^i to the coordinates of the starting points of the matching curve $\tilde{xs}_v^j$ and $\tilde{ys}_v^j$;

d. For an intraframe curve, the sequence of links can now be retrieved from the Huffman code words that follow, otherwise, the next four steps must be performed;

e. The alignment bits specify which part of the link sequence of the matching curve must be used and whether it must be reversed or not;

f. Decode the prediction error links and add them (modulo eight) to the links from the matching curve: $l_u^i(n) = \left(\tilde{l}_v^j(n) + \Delta l_u^i(n)\right) \bmod 8$, until a termination code is received;

g. Decode any remaining intraframe links received until the next termination code and concatenate them with the previously reconstructed link sequence;

h. The alignment bits specify whether the reconstructed link sequence must be reversed or not as a final step.

5.3.4 Lossy interframe coding

Optionally, the prediction error link sequence can be coded lossy, so as to decrease its entropy further. A simple scheme is to zero out certain parts of the prediction error sequence. The zero symbol is already the most frequently encountered symbol in the prediction error sequences; increasing this frequency lowers the entropy and assigns the shortest Huffman code word available to more links. Below we describe an algorithm to achieve this, but first we state some general restrictions which we applied to the lossy interframe scheme.

- The length of the link sequence must be preserved. Otherwise, the algorithm would become very complex (especially the correspondence between parameter signal samples and edge points on the curve).
- The starting and end point of the current curve C_u^i must be preserved. We use this restriction, because starting and end points can be considered important in preserving the general topology of the shapes formed by the set of curves (e.g. if they are branch points).
- It must be possible to control the distortion of the decoded curve. Here, a maximum displacement η of any point in any direction is applied. Denoting the coordinates of the n-th point on curve u by $(x_{u,n}^i, y_{u,n}^i)$ and the corresponding point on the decoded curve by $(\hat{x}_{u,n}^i, \hat{y}_{u,n}^i)$, then:

$$\left|x_{u,n}^i - \hat{x}_{u,n}^i\right| \le \eta \quad \wedge \quad \left|y_{u,n}^i - \hat{y}_{u,n}^i\right| \le \eta \quad , \quad 0 \le n \le N_u^i$$

- Points on C_u^i which already have the same position as the corresponding point on the displaced version of $\tilde{C}_v^j$, i.e. $\left(x_{u,n}^i = \tilde{x}_{v,n}^j + vx_u^i\right) \wedge \left(y_{u,n}^i = \tilde{y}_{v,n}^j + vy_u^i\right)$, are preserved.

The following algorithm zeros out certain parts of the prediction error link sequences, by working on the original sequences and distorting $l_u^i(n)$ such that it becomes closer to $\tilde{l}_v^j(n)$. This algorithm is applied in interframe chain coding *before* the subtraction stage. It obeys the above restrictions. Note that the starting points of C_u^i and the displaced version of $\tilde{C}_v^j$ always coincide, since the displacement is defined as such. See also Figure 5.4 for illustration.

Repeat the following steps (until the end of one of the input sequences is reached):

a. Find two points p and q on C_u^i, which are also intersected - after the displacement - by two corresponding points on $\tilde{C}_v^j$, i.e.

$$\left(x_{u,p}^i = \tilde{x}_{v,p}^j + vx_u^i\right) \wedge \left(y_{u,p}^i = \tilde{y}_{v,p}^j + vy_u^i\right) \wedge$$

$$\left(x_{u,q}^i = \tilde{x}_{v,q}^j + vx_u^i\right) \wedge \left(y_{u,q}^i = \tilde{y}_{v,q}^j + vy_u^i\right) .$$

b. Check if all points r, between p and q, on C_u^i lie within a distance η of the corresponding points on the displaced version of $\tilde{C}_v^j$, i.e.

$$\left(\left|x_{u,r}^i - \left(\tilde{x}_{v,r}^j + vx_u^i\right)\right| \le \eta\right) \wedge \left(\left|y_{u,r}^i - \left(\tilde{y}_{v,r}^j + vy_u^i\right)\right| \le \eta\right), \qquad p < r < q .$$

c. If the above condition is met, the links of the prediction curve $\tilde{C}_v^j$ between the p-th and q-th point are copied to the corresponding links of the current curve C_u^i, i.e.

$$l_u^i(r) = \tilde{l}_v^j(r) \qquad , \qquad p < r \le q .$$

Effectively, the differential chain codes between the p-th and q-th point are set to zero, thereby introducing some distortion in the decoded curve.

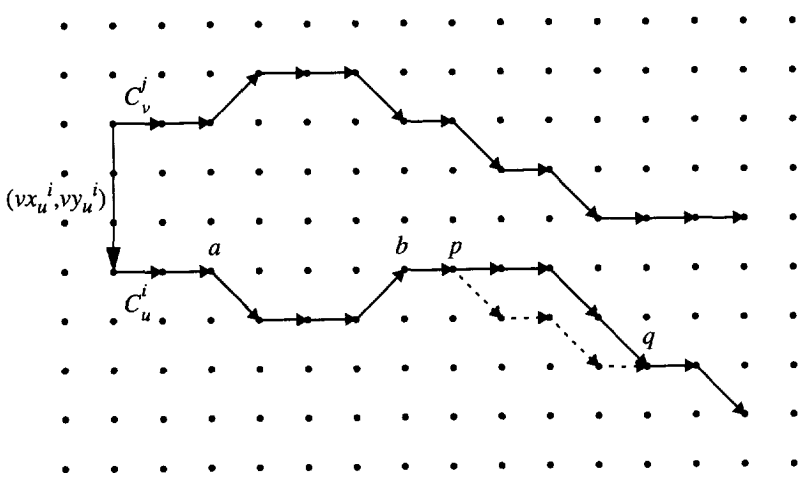


Figure 5.4 Lossy differential chain coding. The prediction curve is at the top and the current curve at the bottom, both drawn in solid lines. When $\eta = 0$, the decoded curve follows the current curve exactly. When $\eta = 1$, the decoded curve follows the current curve except for the part between the p-th and q-th point, where it follows the curve drawn in dashed lines.

An example of this procedure is given in Figure 5.4. The original link sequence $l_u^i(n)$ is: 0070010007707. The original link sequence $\tilde{l}_v^j(n)$ is: 0010070707000. The differential link sequence $\Delta l_u^i(n)$ with $\eta = 0$ (i.e. no distortion allowed) is: 0060020100707. The differential link sequence $\Delta l_u^i(n)$ with $\eta = 1$ is: 0060020000007. The part of the sequence between the p-th and the q-th point obeys the above restrictions and the prediction error is set to zero. Effectively, two more zeros appear in the differential sequence than before. The part of the sequence between the a-th and b-th point is not affected, because the deviation of the constituent points would be too large. The last link is not affected either, even though the deviation of the last point would be only one unit, because we do not allow the first or last point to be displaced.

*Table 5.1 Summary of information to be transmitted for each **intra**frame curve. Each link sequence includes the end-of-chain symbol. The '*' symbol means the number of bits for that item is variable and unknown a priori.*

Type	Class	(xs_u, ys_u)	$l_u(n)$
# bits	1	12	*

*Table 5.2 Summary of information to be transmitted for each **inter**frame curve.*

Type	Class	Label v	(vx_u, vy_u)	Align mode	$\Delta l_u(n)$	$l_u(n)$
# bits	1	6	8	2	*	*

5.4 Experimental results

In this section, we describe experiments with image sequence coding and their results. The goals of the experiments are to gain quantitative insight into the operation of an interframe edge primitive coder and to identify important design issues.

5.4.1 Method

The image sequence coding scheme used in the experiments, in which both the edge primitive matching and interframe location coder are embedded, is depicted in Figure 5.5. The overall structure of the scheme is very similar to that of other video coding schemes, except that, here, the spatial processing precedes the temporal processing. The corresponding scheme for the decoder is depicted in Figure 5.6. The symbols are as explained in the previous sections, the superscript denoting the frame number has been omitted for clarity reasons.

The interframe location coding loop is clearly visible in both graphs. The Location Coding (respectively Location Decoding) block is considered to include the subtractive (respectively additive) unit that is familiar from other video coding schemes. All parameter signals (center intensity value m, contrast c and width w) are coded **intra**frame, therefore, a full loop is not present in the graph. The primitive memory at the encoder side contains the edge location and

edge parameter information for all edge primitives from the previous frame. The primitive memory at the decoder side need only contain the edge location information from the previous frame.

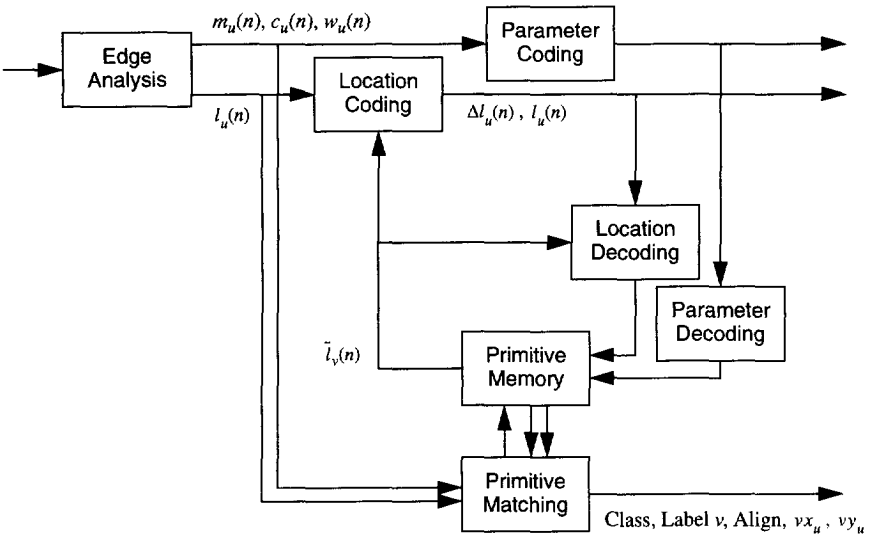


Figure 5.5 Experimental edge-based image sequence coding scheme: encoder.

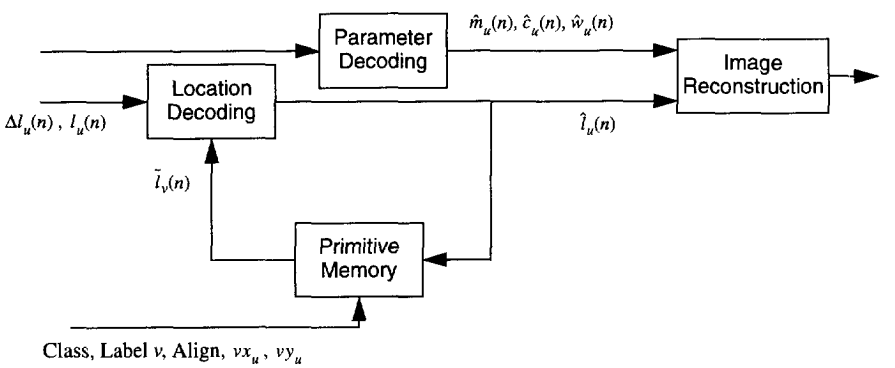


Figure 5.6 Experimental edge-based image sequence coding scheme: decoder.

We performed four experiments to compare four different modes of the image sequence coding scheme: experiments I through IV. Only the edge location coding is different in each experiment.

I. All edge primitive data is coded intraframe. No distortion is allowed on the edge point location edge information, i.e. the link chains. This experiment is used to determine the "baseline" performance, with which interframe coding can be compared. The matching algorithm is not used in experiment I.

II. Interframe location coding is applied, as described in Section 5.3.3, i.e., in the lossless mode. No distortion is allowed on the edge locations.

III. Interframe location coding is applied, as described in Section 5.3.4, i.e., in the lossy mode. We set $\eta = 1$, thus allowing some distortion on the edge locations.

IV. Interframe location coding is applied, as described in Section 5.3.4, i.e., in the lossy mode. We set $\eta = 2$, thus allowing more distortion on the edge locations.

All other settings remained the same throughout all the experiments, including those in the edge analysis stage, the edge primitive matching algorithm, the edge parameter coding algorithm and the intensity surface reconstruction algorithm. In the primitive extraction stage, the filter scale $\sigma = 1.0$ and the minimum curve length $\xi = 4$. The edge selection threshold is set differently for each image sequence. The parameters in the edge primitive matching algorithm are as mentioned in Section 5.2. Minimum similarity $\Phi_{min} = 0.20$; the difference between starting points of candidate matching curves must be greater than -8 and less than or equal to 8. In the edge parameter coding stage, each parameter signal is always subsampled once (by a factor of two); the DPCM stage uses a second-order predictor and a Uniform Threshold Quantizer of twenty-nine levels. In the intensity reconstruction stage, two cycles of MS-SOR iterations are performed in all cases, with maximum scale $K = 25$ (a total of 100 iterations) and $\lambda = 0.1$ (relatively tight constraints). In all the experiments, the similarity between subsequent frames in a sequence is exploited during the intensity surface reconstruction stage. This is done by using parts of the reconstructed previous frame to initialize the reconstruction of the current frame, as explained in Chapter 3. Other parameters are set as in Chapter 4.

We have used three different image sequences to test the intra/interframe coder: the "Claire" sequence, the "Miss America" sequence and the "Carphone" sequence. A frame of the "Claire" sequence is shown in Chapter 4. Frames of the "Miss America" sequence and the "Carphone" sequence are shown in Figure 5.7. All sequences originally have a frame frequency of 30 frames per second, but we only code every other frame, thus reducing the frame frequency to 15 frames per second. The "Carphone" sequence contains global motion (as opposed to the other sequences); no further special consideration is given to this fact.

First, the performance of the edge primitive matching algorithm is discussed. Then, we discuss the efficiency of the (interframe) chain coder, and, finally, the performance of the overall coding scheme.

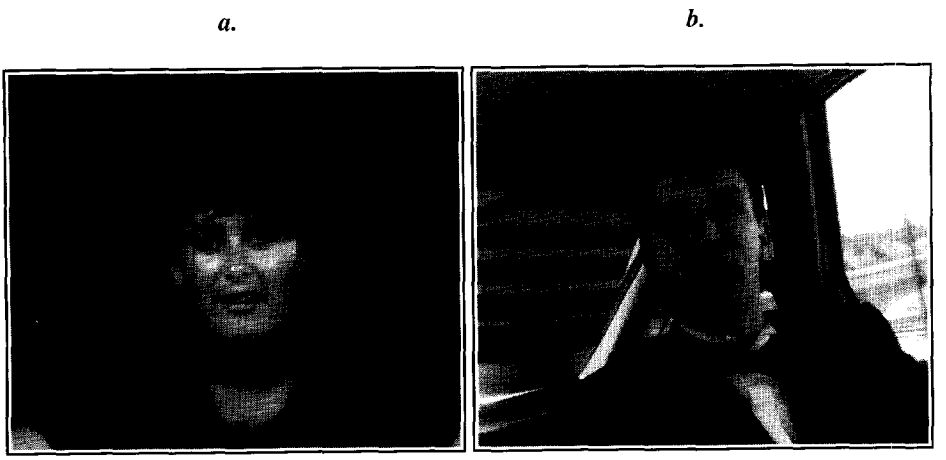


Figure 5.7 (a) Frame 1 of the Miss America image sequence (352x288 pixels); (b) Frame 21 of the Carphone image sequence (352x288 pixels).

5.4.2 Matching performance

Here, we show some results of the matching algorithm as embedded in the coder. The number of curves coded in interframe mode depends on the number of curves for which a match can be found in the previous frame. Note that the edge primitives from the previous frame are coded; hence, the edge location information from the previous frame may be distorted in experiments III and IV, but it cannot be distorted in experiment II. The parameter signals are distorted for certain in each experiment. The matching algorithm is not used in experiment I.

We only show the results of the Claire sequence, which are representative of the results with the other sequences, as far as the relative number of matched curves is concerned.Thirty-five frames of the Claire sequence were used (frame numbers 1 to 69, skipping all even frames). The edge selection threshold was fixed at $t_s = 0.95$.

The graph in Figure 5.8(a) shows, for each frame in the Claire sequence: (*i*) the total number of edge *curves* detected, which is the same in each experiment; (*ii*) the number of edge curves matched during experiment II; (*iii*) the same for experiment III; (*iv*) the same for experiment IV. The number of edge curves is denoted by N_C. About 60% of all curves C_u^i in each frame is matched to a curve $\tilde{C}_v^{i-2}$ from a previous frame.

The graph in Figure 5.8(b) shows, for each frame in the Claire sequence: (*i*) the total number of edge *links* present, which is the same in each experiment; (*ii*) the number of links matched during experiment II; (*iii*) the same for experiment III; (*iv*) the same for experiment IV. The number of links is denoted by N_L. When the lengths of matching edge curves differed, only the corresponding links were counted. About 75% of all links in each frame are matched to a link from a previous frame. This number is higher than the previous number because the average length of matched curves is greater than the average length of non-matched curves (short curves are rejected during the matching).

One may conclude that the matching algorithm performs stably under different conditions; the similarity measures used appear relatively insensitive to small noisy distortions of the information stored in the edge primitives.

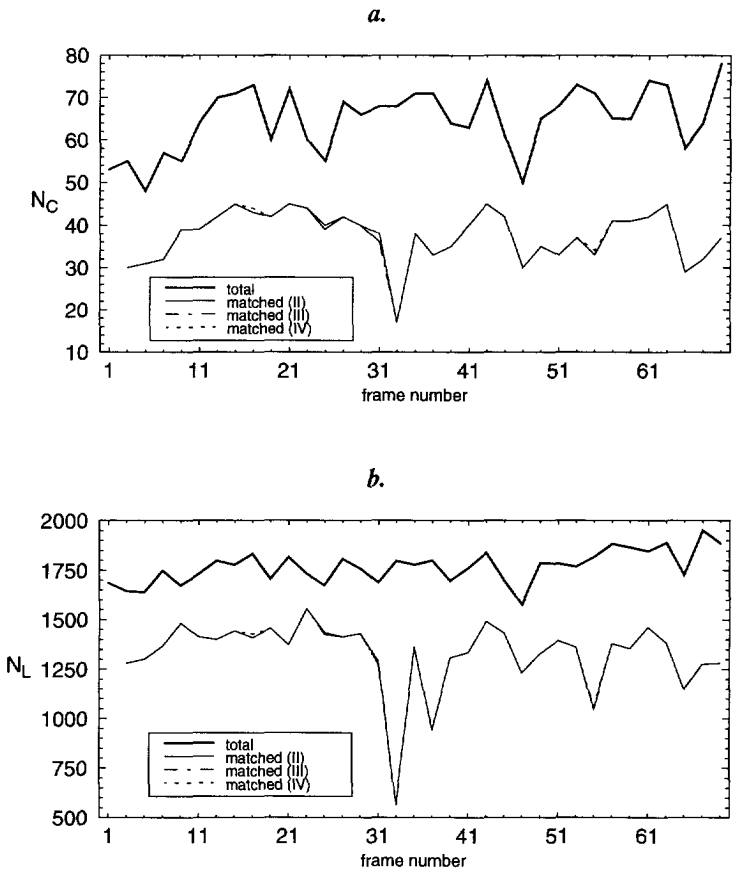


Figure 5.8 Matching results for the Claire sequence: (a) number of curves and number of matched curves in experiments II, III and IV; (b) number of links and number of matched links in experiments II, III and IV.

5.4.3 Intraframe versus interframe coding: chain code entropies

In each experiment, the Claire and Miss America sequences are used to collect frequency distributions of the chain codes, which are used to design the Huffman codes. Then, the Huffman codes are considered to be known at both the transmitter and receiver and, as such, used in the coding experiments. Thus, the same Huffman codes are applied in the coding of the Claire and Miss America sequence (which can be considered "training" sequences) as in

the coding of the Carphone sequence (which can be considered a "test" sequence). The training data consisted of thirty-five frames of the Claire sequence (frame numbers 1 to 69, skipping all even frames) and seventy-five frames of the Miss America sequence (frame numbers 1 to 149, skipping all even frames).

The designed Huffman codes and their estimated entropy and expected rate are of crucial importance to the performance of the entire coding scheme. One hopes to find lower entropies for the prediction error links (or interframe links) than for the normal links (or intraframe links), such that a saving in the total number of bits can be achieved.

First, we show the estimated entropies and expected rates of the Huffman coded links for experiment I (all curves coded intraframe) in Table 5.3. As expected, these results are very similar to the ones obtained in Chapter 4. The expected rate of the Huffman codes designed for a second-order Markov model - which are expected to be the most frequently used codes - is about 1.7 bits per link.

Table 5.3 Experiment I: Entropy and expected rate of chain-coded links for different Markov models.

Model order	Entropy	Expected Rate
0	2.914	2.945
1	1.660	1.793
2	1.526	1.656

Next, we show the estimated entropies and expected rates of the Huffman coded links for experiment II (matched curves coded lossless interframe). Table 5.4 shows the results for normally coded links (**intra** links) and for prediction-error links (**inter** links). The entropies and rates for **intra** coded links unfortunately have become higher, compared to the results in experiment I. While the entropy and rate of zero-order **inter** coded links are considerably lower than of their **intra** coded counterparts in Table 5.3, the entropy and rate of the second-order **inter** coded links is only slightly lower than of their **intra** coded counterparts in Table 5.3. The rate of second-order **inter** links is approximately 1.6 bits per link. The difference, approximately 0.1 bit per link on average, is disappointingly low.

*Table 5.4 Experiment II ($\eta = 0$): Entropy and expected rate of **intra** coded links and **inter** coded links for different Markov models.*

	Intra links		**Inter** links	
Model order	Entropy	Expected Rate	Entropy	Expected Rate
0	3.091	3.119	1.574	1.672
1	1.947	2.074	1.454	1.596
2	1.897	2.024	1.408	1.574

Next, we show the estimated entropies and expected rates of the Huffman coded links for experiment III (matched curves coded lossy interframe, $\eta = 1$). Table 5.5 shows the results for normally coded links (**intra** links) and for prediction-error links (**inter** links). The results for the **intra** links are almost identical to those of experiment II and are not discussed further. The *entropy* of the **inter** coded links now drops considerably further, compared to the entropies in Table 5.3, but the *expected rate* still drops only slightly. The differences in expected rate between lossless and lossy chain coding is also rather small (cf. Table 5.4).

*Table 5.5 Experiment III ($\eta = 1$): Entropy and expected rate of **intra** coded links and **inter** coded links for different Markov models.*

	Intra links		**Inter** links	
Model order	Entropy	Expected Rate	Entropy	Expected Rate
0	3.091	3.119	1.212	1.457
1	1.948	2.074	1.093	1.416
2	1.898	2.025	1.057	1.410

Finally, we show the estimated entropies and expected rates of the Huffman coded links for experiment IV (matched curves coded lossy interframe, $\eta = 2$). Table 5.6 shows the results for **intra** links and for **inter** links. Again, the results for the **intra** links are almost identical to those of experiment II and are not discussed further. Both the *entropy* of the **inter** coded links as well as the *expected rate* drop only slightly, compared to Table 5.5 and Table 5.4. The expected rate of second-order prediction-error links is approximately 1.4 bits per link. The differences in expected rate between lossless and lossy interframe chain coding are still rather small. The gain with respect to intraframe coding is a little better, but still small.

*Table 5.6 Experiment IV ($\eta = 2$): Entropy and expected rate of **intra** coded links and **inter** coded links for different Markov models.*

	Intra links		**Inter** links	
Model order	Entropy	Expected Rate	Entropy	Expected Rate
0	3.091	3.119	1.154	1.429
1	1.950	2.075	1.032	1.389
2	1.900	2.025	0.995	1.384

One may conclude that the gain of **inter**frame edge curve shape coding over **intra**frame coding is expected to be low. While the zero-order statistics of prediction-error links show considerable improvement (a lower *entropy*) over normal links, the second-order statistics are quite comparable. An extra gain can be achieved by allowing some distortion on the curves shapes. However, the Huffman coding technique is not able to design efficient code books at the lowest entropies estimated; therefore, the *expected rates* show only small improvements.

5.4.4 Intraframe versus interframe coding: overall test results

Firstly, the overall coding scheme is evaluated in objective terms; some remarks about the subjective quality of the decoded sequences are made later.

Objective evaluation

The test data consisted of thirty-five frames of the Claire sequence (frame numbers 1 to 69, skipping all even frames), seventy-five frames of the Miss America sequence (frame numbers 1 to 149, skipping all even frames) and seventy-five frames of the Carphone sequence (frame numbers 1 to 149, skipping all even frames).

The results for the Claire sequence are summarized in Table 5.7. The edge selection threshold $t_s = 0.95$. The results for the Miss America sequence are summarized in Table 5.8. The edge selection threshold $t_s = 0.95$. The results for the Carphone sequence are summarized in Table 5.9. The edge selection threshold $t_s = 0.875$. These tables show numbers ***averaged*** over all frames in each sequence. We compare the results of experiments II, III and IV to the results of experiment I, in order to evaluate the differences in performance between interframe coding with fully intraframe coding. The symbols used for the output parameters in the tables are as follows.

- N_C: The number of edge curves;
- N_L: The number of links;
- B_L: The number of bits to code the links;
- R_L: B_L/N_L, the average number of bits per link;
- B_O: The number of bits to code the overhead (starting points for **intra** curves; labels, alignment mode and displacement vectors for **inter** curves);
- B_C: The number of bits to code the intra/inter classification of all curves;
- B_E: The total number of bits to code the edge location data ($\approx B_L+B_O+B_C$);
- N_P: The number of samples in a (downsampled) edge parameter signal;
- B_P: The number of bits to code *all* three parameter signals;
- B_T: The total number of bits to code *all* the data;
- R: The final rate in bit per pixel;
- C: The final compression ratio;
- PSNR: The final peak-to-peak signal-to-noise ratio in dBs of the output image.

Firstly, from the first two rows of each table, it can be seen that about 60% of the edge curves in each frame were matched in experiments II, III and IV throughout all three sequences, and about 75% of all the links. Note that frames from the Miss America sequence contain more curves, on average, than frames from the Claire sequence and that, in turn, frames from the Carphone sequence contain more curves than frames from the Miss America sequence. The number of bits to code the links B_L is in keeping with this observation. However, the link rate R_L is (especially in experiment I) lower for the Carphone sequence compared to both the Claire and Miss America sequence, even though the Huffman codes have been designed on the latter two sequences. This can be explained by the fact that the Carphone sequence contains more straight curves, which can be coded efficiently.

Inspecting the link rates R_L for intra and inter curves in experiments II, III and IV, one can conclude that these are as expected from the results reported in Section 5.4.3. Intra classified links in experiments II, III and IV have a much higher average rate than in experiment I. Inter classified links in experiments II, III and IV have somewhat lower rate than the intra coded links from experiment I, but the difference is only small. Most links are coded using Huffman tables for a second-order Markov chain, and for these, the difference in expected rate was low. Fortunately, the difference in average rate grows somewhat bigger when allowing some distortion on the curves (experiments III and IV). One can note, that in experiment II (in each sequence) the difference is so small that the total number of bits to code the links B_L is *higher* than that in experiment I. In experiments III and IV, this number is *lower* than that in experiment I, as desired.

*Table 5.7 Results experiments I to IV for intra-/interframe edge-based coding of the Claire sequence, **averaged** per frame. For explanation of the symbols in the left column, see page 126. The first five rows give separate results for each type of curve (intra or inter) in each experiment.*

	I	II		III		IV	
type	*intra*	*intra*	*inter*	*intra*	*inter*	*intra*	*inter*
N_C	64.9	27.7	37.6	27.8	37.5	27.7	37.5
N_L	1767.8	442.7	1327.5	443.4	1326.8	442.5	1327.7
B_L	3044.1	1045.2	2125.0	1046.6	1871.1	1044.6	1837.2
R_L (bpl)	1.721	2.460	1.606	2.458	1.414	2.460	1.388
B_O	779.0	332.5	600.9	333.5	599.5	332.8	600.5
B_C	-	65.3		65.3		65.3	
B_E	3823.0	4152.5		3906.8		3872.2	
B_E/N_L (bpl)	2.161	2.347		2.208		2.188	
N_P	1832.1	1832.1		1832.1		1832.1	
B_P	5063.7	5063.7		5063.7		5063.7	
B_T	8886.8	9216.2		8970.6		8936.0	
R (bpp)	0.088	0.091		0.088		0.088	
C	91.57	88.32		90.75		91.09	
PSNR (dB)	30.40	30.40		30.04		29.78	

The fifth row in each table shows the overhead B_O which is relatively higher for inter curves (16 bits per curve) compared to intra curves (12 bits per curve). The extra overhead in interframe coding experiments II, III and IV to code the curve classification B_C (1 bit per curve) must be added as well. It then turns out that, even in experiments III and IV, the total number of bits necessary to code the edge location data B_E is *higher* in all the interframe

coding experiments than in the fully intraframe coding experiment. The average number of bits per link (including the overhead) B_E/N_L still is approximately 2.0-2.5 bits per link. This result is, of course, rather disappointing.

The number of samples in each parameter signal N_P and the number of bits to code all three signals B_P is equal in experiments I, II, III and IV. The total number of bits to code all the data B_T is in keeping with the previous result: **inter**frame coding takes, on average, *more* bits to code a frame than does **intra**frame coding - between 9000 bits and 26000 bits. Converting into a transmission rate, this corresponds to (at 15 frames per second) approximately 135 kbit per second to 390 kbit per second.

*Table 5.8 Results experiments I to IV for intra-/interframe edge-based coding of the Miss America sequence, **averaged** per frame. For explanation of the symbols in the left column, see page 126.*

	I	II		III		IV	
type	*intra*	*intra*	*inter*	*intra*	*inter*	*intra*	*inter*
N_C	102.4	43.1	59.3	43.1	59.2	43.1	59.3
N_L	2069.8	565.5	1503.3	566.4	1502.4	565.5	1503.3
B_L	3792.2	1413.1	2501.0	1415.5	2242.9	1413.6	2199.6
R_L (bpl)	1.832	2.517	1.664	2.518	1.494	2.519	1.464
B_O	1229.0	512.1	948.1	517.5	947.7	517.0	948.3
B_C	-	102.4		102.4		102.4	
B_E	5021.2	5480.7		5228.3		5183.9	
B_E/N_L (bpl)	2.425	2.647		2.525		2.504	
N_P	2170.6	2170.6		2170.6		2170.6	
B_P	6304.6	6304.6		6304.6		6304.6	
B_T	11325.8	11785.3		11533.0		11488.5	
R (bpp)	0.112	0.116		0.114		0.113	
C	71.70	68.91		70.42		70.69	
PSNR (dB)	32.70	32.70		32.57		32.44	

The final rate R in bits per pixel lies between 0.088 for the Claire sequence and 0.259 for the Carphone sequence. Compression ratios per frame lie between 30 and 90. The quality of the reconstructed image lies approximately between 27 dB for the Carphone sequence and 33 dB for the Miss America sequence. The quality of the output image is equal in experiments I and II because the edge location data is coded lossless in both cases. The effect of the distortion allowed on the curve shapes in experiments III and IV on the final quality is quite small in terms of the PSNR - approximately 0.4 dB on average.

*Table 5.9 Results experiments I to IV for intra-/interframe edge-based coding of the Carphone sequence, **averaged** per frame. For explanation of the symbols in the left column, see page 126.*

	I	II		III		IV	
type	*intra*	*intra*	*inter*	*intra*	*inter*	*intra*	*inter*
N_C	179.3	70.9	107.9	70.8	107.9	70.8	107.9
N_L	5404.4	1211.1	4184.3	1206.5	4188.9	1206.5	4188.9
B_L	8626.7	2694.4	6212.8	2690.7	5503.6	2692.0	5438.5
R_L (bpl)	1.597	2.252	1.487	2.257	1.316	2.258	1.300
B_O	2152.0	850.2	1725.6	849.9	1726.1	850.1	1725.8
B_C	-	178.7		178.7		178.7	
B_E	10778.7	11679.4		10976.2		10913.3	
B_E/N_L (bpl)	1.995	2.162		2.032		2.020	
N_P	5582.4	5582.4		5582.4		5582.4	
B_P	14578.0	14578.0		14578.0		14578.0	
B_T	25356.7	26257.4		25554.2		25491.3	
R (bpp)	0.250	0.259		0.252		0.251	
C	32.02	30.92		31.78		31.85	
PSNR (dB)	27.82	27.82		27.46		27.31	

The graphs in Figure 5.9 to Figure 5.11 show some of the most important output parameters for *every frame* in each sequence. In each graph, the results of experiments I to IV are displayed jointly. The number of edge location bits B_E is shown in Figure 5.9 and the total number of bits to code all the data B_T in Figure 5.10. Both graphs show that the best results (i.e. the lowest number of bits) are always obtained by **intra**frame coding (experiment I). The worst result is obtained by **lossless inter**frame coding (experiment II with $\eta = 0$). **Lossy inter**frame coding (experiment III with $\eta = 1$ and experiment IV with $\eta = 2$) ranges somewhere between the two. The graphs also show that these numbers can vary a few thousands of bits along the sequence.

The peak-to-peak signal-to-noise ratio PSNR is shown in Figure 5.11. The best results (highest PSNR) are obtained in experiments I and II, full **intra**frame coding and **lossless inter**frame coding. The results in experiments III and IV (**lossy inter**frame coding) follow next, as expected. The PSNR varies about 3 dB along the sequences, except for one frame of Miss America. This frame contained two curves with different contrast sign, which became connected into one curve, for which only *one* contrast sign may be transmitted. This led to a low PSNR value.

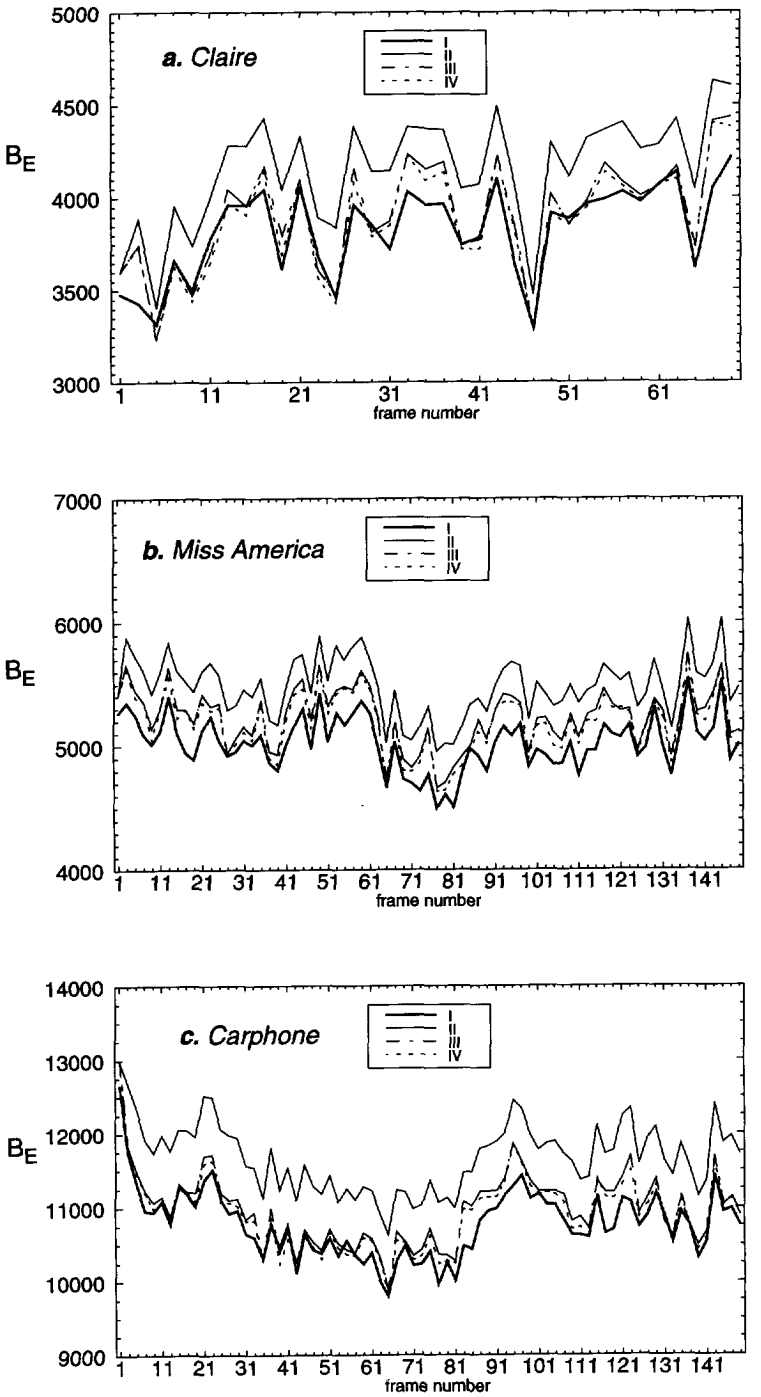


Figure 5.9 Coding results: number of edge location bits for each frame of the (a) Claire sequence; (b) Miss America sequence; (c) Carphone sequence.

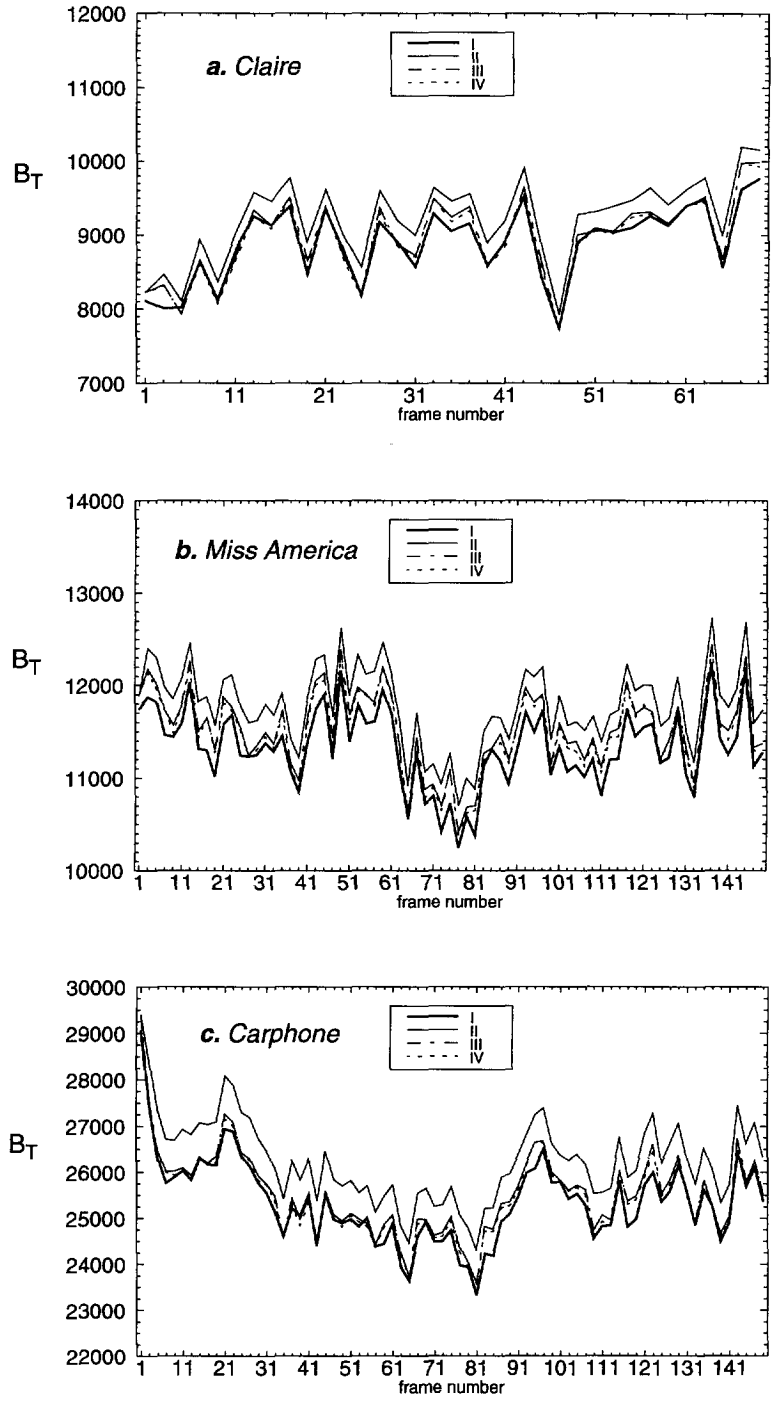


Figure 5.10 Coding results: total number of bits for each frame of the (a) Claire sequence; (b) Miss America sequence; (c) Carphone sequence.

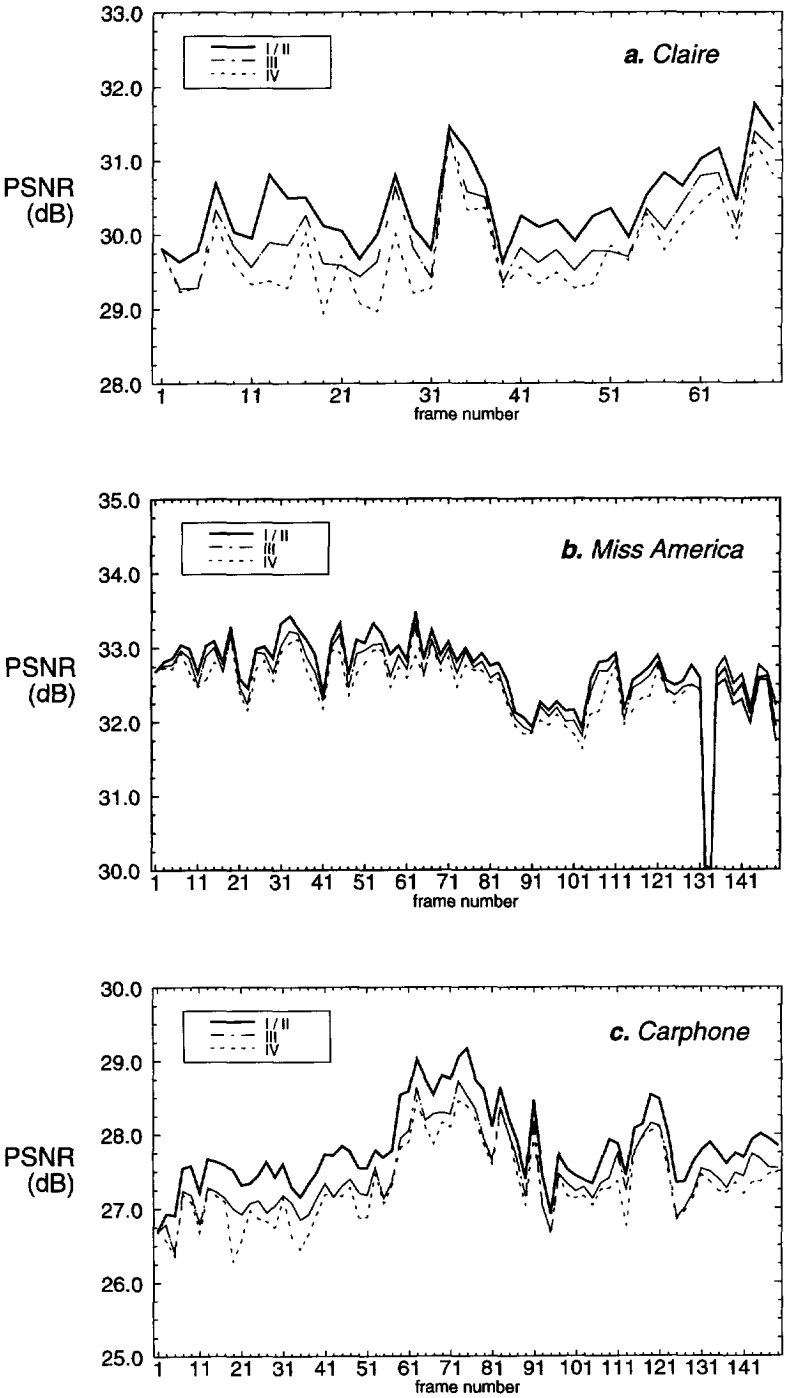


Figure 5.11 Coding results: Peak Signal-to-Noise Ratio for each frame of the (a) Claire sequence; (b) Miss America sequence; (c) Carphone sequence.

Subjective evaluation

When viewing the decoded image sequences on a monitor, an impression of the subjective quality can be obtained. Here, we restrict ourselves to describing the overall impression and to listing some of the artifacts that can be noted.

The overall impression of the quality was not very positive. Artifacts which can be noted in the still image condition have already been mentioned in Chapter 4. Some of these artifacts, although not annoying when viewed as a still image, become very bothersome in the video condition. Severe flickering of regions all over the image between brighter and less bright states occurs, giving the entire image a very unstable impression. In the same fashion, "hazes" of darker and lighter patches seem to blow over the image. More specifically, one can note the following artifacts.

- Significant flickering of large-sized regions in the images and "hazes" of medium-sized regions blowing over the image;
- Severe edge business and visibility of small-sized light patches moving along edges; mosquito effects (flickering of tiny dark regions near contours);
- "Halos" and "auras" (corresponding to iso-intensity borders) become even more visible than in the still image case, probably due to their apparent motion;
- Appearing and disappearing of features such as edges.

The flickering effect and the unstable impression are most likely caused by the following.

- Edge extraction is performed "intraframe"; therefore, certain features may be detected in one frame but not in the other, because their gradient magnitude slightly fluctuates around the edge selection threshold. Also, certain curves may form closed regions in one frame, but not in the other.
- Intensity surface reconstruction is performed "intraframe"; surfaces with minimum variation in one frame may undergo severe changes in the next frame, due to the moving and changing edge content (and also due to disappearing edges). *Temporal* smoothness is not guaranteed during the surface reconstruction.

Thus, in the author's opinion, the visual quality of the decoded video sequence is severely decreased with respect to the quality of the constituent still frames, even though the objective quality (measured by the PSNR) may be the same.

The visual differences between **lossless** and **lossy** interframe coding were small in the experiments performed, as illustrated below. In Figure 5.12, we show in (a) the original edges detected in frame 21 of the Carphone image sequence and in (c) the decoded image using the edges in (a) (taken from experiment I). In Figure 5.12(b), the lossy coded edges (with $\eta = 2$) are shown, and in (d) the decoded image using the edges in (b) (taken from experiment IV). The differences between (a) and (b) are only slight, and so are the differences between (c) and (d). Some artifacts can be seen in the edge curves in the rear window of the car: small deviations of a few pixels, but also two long curves lying closer to each other. When viewing the sequence, sometimes a curve can be seen to hold still while its surroundings move slowly;

then, after some time, the whole edge curve moves suddenly to correct its position. Also, the mouth of the person can be seen to be distorted. Although the impact of lossy edge location coding on the overall quality does not seem strong in this case, it is difficult to judge the effect properly because the other distortions are so dominant.

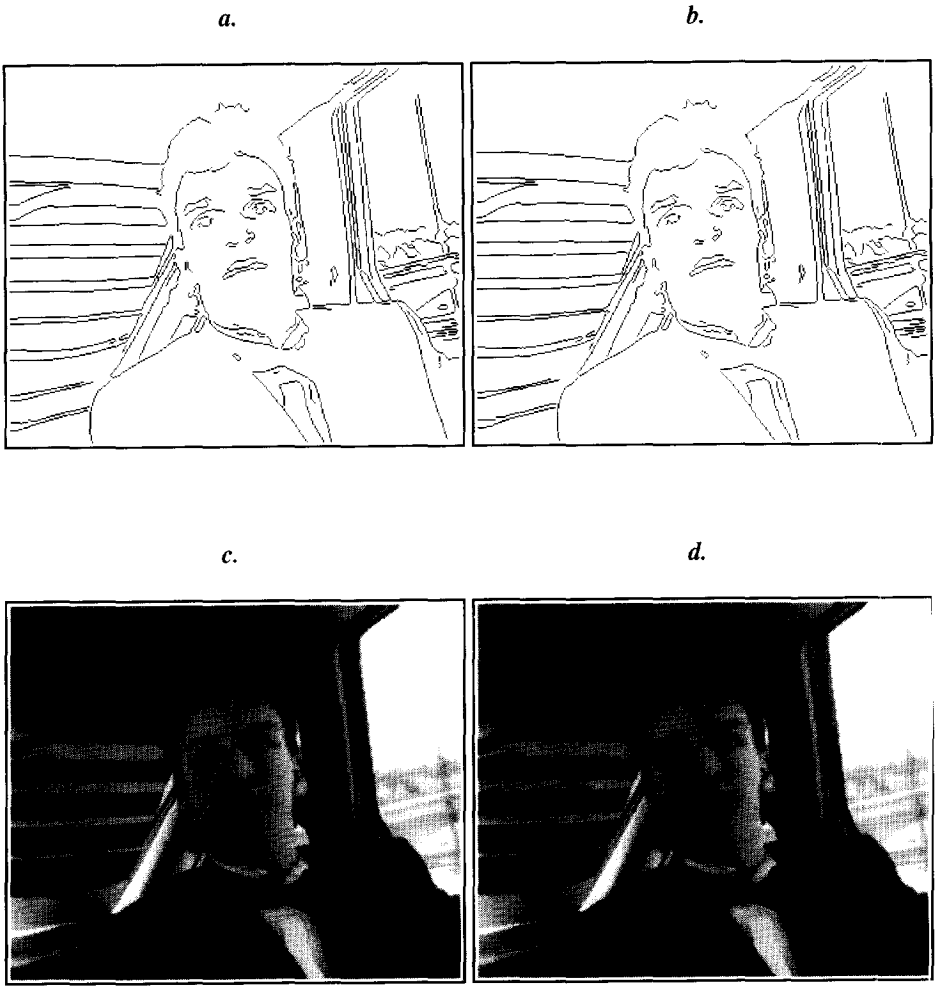


Figure 5.12 Image sequence coding results: (a) edges detected in frame 21 of the Carphone sequence; (b) lossy coded edges of frame 21 ($\eta = 2$); (c) reconstructed image based on (a) (experiments I and II); (d) reconstructed image based on (b) (experiment IV). The corresponding original frame of the Carphone sequence is shown in Figure 5.7(b).

5.5 Conclusions

In image sequences, the temporal correlation between edge primitives is exploited in the design of an interframe coder. Interframe edge primitive matching and interframe edge location coding have been discussed in this chapter.

Edge primitive matching

A matching algorithm has been designed based on a similarity measure which takes into account both *shape* and *grey-level* properties of the edge primitives. Shape differences are measured by a *minimum mean-squared distance* functional. Grey-level similarity is measured by cross-correlating the parameter signals. The matching algorithm seeks, for each edge curve in the current frame, a curve in a previous frame with maximum overall similarity. The matching algorithm is computationally quite efficient.

In the experiments, about 60% of all curves in each frame are matched to some curve from a previous frame. At the same time, about 75% of all links in each frame are matched to a link from a previous frame. A matching rate of 75% for the links is a reasonable result considering the relatively low complexity of the matching algorithm applied. Also, the performance was stable for the image sequences tested and insensitive to small distortion of edge information (due to coding).

Interframe edge location coding

The (chain-coded) link sequences of matched curves can be coded interframe, using the link sequence of the matching curve as a prediction, and transmitting only a sequence of prediction error links. A displacement vector is transmitted to code the location of the curve starting point. Optionally, the prediction error links can be coded lossy.

Experiments showed that the gain of **inter**frame edge curve shape coding over **intra**frame coding is low. While the zero-order statistics of prediction error links show considerable improvement (a lower entropy) over normal links, the improvement of the second-order statistics is quite small. An extra gain can be achieved by allowing some distortion on the shapes of the curves. However, the Huffman coding technique is not able to design efficient code books at the lowest entropies estimated; therefore, the average rates of the Huffman codes show only small improvements. The improvement is too small compared to the extra overhead that must be transmitted for interframe coded curves, so that a gain *cannot* be achieved with respect to intraframe coding. Thus, in our experiments, **inter**frame coding takes, on average, *more* bits to code a frame than does **intra**frame coding.

The final rate R in bits per pixel lies between 0.088 for the Claire sequence and 0.259 for the Carphone sequence. Compression ratios per frame lie between 30 and 90. The quality of the reconstructed images lies between 27 dB for the Carphone sequence and 33 dB for the Miss America sequence. The effect on the final quality of the distortion allowed on the curve shapes in lossy interframe coding is quite small in terms of the PSNR - about 0.4 dB on average.

The visual quality of the decoded images in our experiments is rather bad in our opinion. Severe flickering of regions all over the image between brighter and darker states occurs,

giving the entire image a very unstable impression. In the same fashion, annoying "hazes" of darker and lighter patches seem to blow over the image. Also, severe edge business can be noted.

Further research

The results of the interframe edge location experiments are, of course, rather disappointing. One may try run-length coding of the zeros in interframe chain code sequences to increase the efficiency. Also, more complex coding strategies could be envisioned. For instance change detection may be used, such that edge primitives which have not changed do not have to be transmitted. The matching algorithm can still be useful when attempting to code the grey-level information in the *edge parameter signals* predictively.

The use of a temporal smoothness measure in addition to the spatial smoothness measure in the intensity surface reconstruction should be studied. This might alleviate the temporal flickering problem.

Chapter 6
Discussion

In this chapter, we discuss the significance of the results of our overall approach to image representation and coding. We further compare the merits and drawbacks of our scheme to those of some other new approaches in image coding. We comment on some implementation aspects and, finally, give some recommendations for further research.

6.1 Contributions

In this thesis, we have described an approach to the efficient representation of intensity images by way of their edges. Intensity edges are assumed to represent a significant amount of the intensity surface variations in many images.

An edge model is used to describe sharp transitions in the intensity surface lying along curves in the image plane. Simple methods to estimate the contrast, width and center value of these edge transitions were discussed in Chapter 2. Choosing an appropriate scale on which an image is analyzed, turned out to be important to the robustness of these simple methods.

The information in the representation constrains the intensity surface only near the edge curves. To obtain a reconstruction of the full intensity surface, one assumes the surface between the edge curves to be as smooth as possible. Then, a unique surface can be found by iteratively optimizing a mathematically formalized smoothness measure. A fast optimization algorithm was found and a new theoretical framework was proposed in Chapter 3, explaining the speed-up obtained.

Similar representations have been proposed by Cumani, Grattoni and Guiducci in [32] and [47] and by Neumann, Ottenberg and Stiehl in [87] and [89]. However, none of these provided a thorough validation of their approach by theoretical or experimental study. A similar

approach to step edge modeling has been proposed by Alter-Gartenberg et al. in [3], who carefully studied the accuracy of the estimate of contrast they obtained.

The approach has been applied to image coding. The data in the representation (edge point location information and edge model parameter signals) can be compressed further by using conventional coding techniques (chain coding, downsampling and predictive quantization). At very high compression ratios and for certain types of still images, a better quality of the decoded images can be obtained by an edge-based coder than by a block-based coding scheme like JPEG (see Chapter 4). For example, images of simple scenes (such as head-and-shoulders type scenes with uniform background, encountered in the videophone application) can be highly compressed. An edge-based coder is particularly suited for images in which the primary features are spatially localized in long straight lines and images which contain little texture. While the distortions are severe at such a high compression ratio, they appear less disturbing to a human viewer than those produced by JPEG.

A similar approach to image coding was discussed by Carlsson in [26]. However, no numerical results on the fidelity of the decoded images was included in that paper.

The temporal correlation in image sequences can be exploited by matching edge curves from subsequent images and applying predictive coding. We have proposed new methods for lossless as well as lossy interframe chain coding. Although a fully interframe image sequence coder has not been studied here, it seems that little gain can be achieved with respect to the still image case. We have not been able to achieve higher compression ratios in coding the edge point location information by taking interframe correlations between curves into account (see Chapter 5). The edge point location information forms a large part of the total amount of information to be transmitted, as discussed in Chapter 4.

It follows from the above that use of an edge-based coder seems advantageous in a limited number of applications. One application where it may be useful is lip-reading by hearing-impaired people via the videophone. One may assume that, in images communicated for the purpose of lip-reading, the *shapes* and *positions* of features in the human face (such as the mouth) are more important than the exact intensity level in a certain area in the face. At low bit-rates, the edge-based scheme preserves these important properties much better than do block-based schemes. An early attempt to communicate two-level "cartoon" images (showing perceptually relevant features of the human face and hands) at very low bit rates was reported by Pearson in [92]. Another example where the edge-based scheme may be useful is that of databases of (still) images of industrial objects or other man-made artefacts. Such objects often have smooth surfaces and smooth outlines. Pictures of such objects contain little texture, and the edge curves that are present have smooth shapes. Such pictures can be handled well by the edge-based compression scheme. Further, one may want to retrieve images from an image database using queries by *content* rather than by *identifiers* or *text annotations* [96]. A database of images compressed by the edge-based scheme allows direct and general purpose queries using, e.g., a sketch or line drawing of an object. Time-consuming low-level image analysis operations do not have to applied during a search, because these have already been performed during compression. At the same time, exhaustive pre-annotation of relevant image events is unnecessary, because these are still present in the compressed image data.

The methods we have studied in Chapter 2 and Chapter 3 are applicable in other image processing fields than image coding. The simple methods we have proposed to estimate edge properties in images, such as their contrast and width, can be applied to a problem in computer vision. As the point-spread-function of an optical acquisition system actually depends on both the focus of the camera and the distance of the point being imaged to the camera (object points which are out of focus are blurred), the smoothness or width of edges in images has been considered in methods to estimate scene depth from single images, e.g. [71] or [94], called *depth from defocus*. This form of blur parameter identification could also be of interest in certain image restoration problems [16], where the point-spread-function of the distortion can be modeled by a Gaussian function. Some calibration is necessary in both cases to account for the internal camera parameters.

The new multi-scale method we have proposed so solve the surface interpolation problem faster can be applied in many problems in image processing as well as outside this field. Among the problems in image processing are: *motion estimation*, *disparity estimation*, *recovering lightness*, *shape from shading* (see [14] and [116]). More generally, any algorithm based on the a relaxation method to solve elliptic second-order partial differential equations on large-sized grids can be speeded up by this technique. Also, the method provides a powerful low-pass filtering technique, as discussed in Appendix A.

6.2 Implementation aspects

Here, we comment on the applicability of the proposed coding method in practice, and its computational complexity.

In principle, all the crucial components of an edge-based coding system for the compression of (still) images have been outlined in the previous chapters. However, a number of parameters play an important role in the overall design, which have not been studied here in depth. In the image analysis stage, important parameters are the filter scale (σ) and the edge selection threshold (t_s).

During the coding experiments, the filter scale was kept fixed at 1.0. This works adequately in images which are sharp, properly sampled and contain little noise. Nevertheless, keeping the scale fixed at any value is not a viable option for particular images which contain features of a very different extent. In the latter case, a multi-scale analysis might be necessary and scale space matching must be applied to establish the correspondence between edge curves from different scales (e.g., see [12] and [100]). Some progress has been made by us in this area: a flexible yet simple matching algorithm has been implemented which can both efficiently and robustly track edge curves through the scale space [11]. It makes efficient use of both the data structure that holds the edge curves and of the image data structure.

In all our experiments, the edge selection threshold was fixed manually, such that the significant features in the scene were preserved in the decoded image. The edge selection threshold might conveniently be used to vary the output image quality and compression ratio. As the threshold is increased, more features with higher contrast disappear from the coded image and the compression ratio increases. As the threshold is decreased, more features with

lower contrast appear in the coded image and the compression ratio decreases. It is then the user's responsibility to adapt the threshold, such that the decoded image still conveys the scene features which he or she regards as being relevant.

The main parameters to be set in the edge parameter signal coding stage itself are the number of downsampling steps (L) and the number of quantization levels (Q) or the quality of the quantizer. The effect of these parameters on the quality of the decoded images was addressed in Chapter 4, but not extensively. It was found that the overall coder could compete with JPEG when these parameters were pushed into the low quality area.

The main parameters to be set in the intensity surface reconstruction stage are the regularization parameter (λ) and the maximum scale to be visited (K). In most of our experiments, we kept λ close to zero, implying tight data constraints. One may let λ increase for increasingly coarse quantizers used in the coding, i.e., depending on Q. The reason is that coarser quantization means that the variance or uncertainty in the data increases. The maximum scale K can easily be related to the maximum distance on the grid found by the Distance Transform, which is applied when setting up the multi-scale reconstruction algorithm.

Bit allocation and bit rate control, which are part of many coding systems, have not been discussed in this thesis. With the still image coder proposed in Chapter 4, it is difficult to compute input parameters such that an optimal quality is attained for a given target compression ratio and for a given image. In addition, the experimental image sequence coder used in Chapter 5 produces a variable bit rate and a variable quality. This restricts the use of our coding system to applications where a fixed bit rate is not required.

A system for coding *still* images operates under a relatively mild time constraint; a system for coding image *sequences* must deal with a very strict time constraint. The time necessary to process an image can be considered to depend on the number of pixels in the image and the average number of operations necessary for each pixel. With real-time image sequence processing, the number of frames that must be processed per second is a factor as well. As far as computational complexity is concerned, it is clear that the main difficulty with our scheme lies in the intensity surface reconstruction stage.

Real-time *en*coding seems possible. The edge analysis stage consists mainly of two filtering stages to obtain partial derivatives. In addition, various local computations must be performed. In the order of 500 operations per pixel must be performed in the edge analysis stage. The edge primitive coding algorithms can work on one-dimensional data structures and require only a fraction of the processing time needed by other stages. This is because the number of points that must be processed in this stage is only a fraction (5-10%) of the number of pixels in the original image.

Real-time *de*coding seems much more difficult. The intensity surface reconstruction algorithm is employed on the decoding side of the system and is quite computation intensive. Starting from a good initialization, the MS-SOR algorithm still needs in the order of 50-100 iterations to converge to an acceptable solution. Each iteration, about 15 operations per pixel are performed, of which 7 are floating-point operations. In total, this forms between 750 and 1500

operations per pixel. In low bit rate coding, the size of an image typically ranges from 128x128 pixels to 512x512 pixels, while the frame frequency lies between 10 and 30 frames per second. This means that between 1.3×10^{-7} and 6.1×10^{-6} seconds are available to process each pixel, if all pixels are processed sequentially.

While still being far away from real-time operation, it is expected that this goal can be attained in the future because of the steady increase in processing power of (VLSI) hardware and because of the inherently parallel nature of filtering and relaxation algorithms, suggesting parallel implementations.

6.3 A wider perspective on very low bit rate coding

Recently, a growing interest in very low bit rate coding of image sequences can be observed (e.g., see [64]). Standardization efforts are already under way [5], which give further stimulus to research activities.

Early endeavors in the field of high compression image coding are reported by Kunt and co-workers in [69] and [70]. These papers already contained the main ideas used in various approaches presented later. The most important difference between these new methods and standard block-based methods (JPEG, MPEG) is that images are often partitioned into regions of arbitrary shape (corresponding to objects in the scene) instead of fixed-sized square blocks.

- These methods are referred to by a variety of names, such as *second-generation* coding, *segmentation-based* coding [17], *contour/texture* coding [75] or *region-based* coding [42]. In *pyramid coding*, the segmentation is restricted to quad-tree partitionings. These methods generally use 2-D shape models and are mainly used in still image coding. Another recent 2-D technique is *fractal coding* [57], in which self-similarity in the image data is exploited.
- A promising approach for moving images is called *object-based* coding, proposed by Musmann and co-workers [85]. Here, objects are defined by their motion, shape and color. The motion and shape of objects can be parameterized using general 2-D or 3-D models. Camera and illumination models can also be taken into account. The primary aim is to achieve more accurate motion-compensated predictions than can be achieved with block-based schemes.
- Combination of 3-D object modeling and a priori knowledge leads to an approach called *knowledge-based* coding or *model-based* coding [2][40]. In these methods, the presence of specific objects in the scene is presupposed, such as a human face. A facial model is initially fitted to the image and its motion is subsequently tracked through the sequence [102]. Detailed facial expressions can be compactly described by predefined model deformation actions. The general analysis problem is highly complex and unsolved as yet, although advances have been made [103].

Leaving model-based techniques out of consideration, very few of the approaches mentioned above have shown significant improvements in *objective* rate-distortion characteristics over conventional coding techniques, despite the efforts made. In our opinion, compelling evidence that these approaches lead to much more efficient coding schemes than standard DCT coding

or hybrid motion-compensation/DCT coding has not been reported yet. Most studies report comparable performance, for example, see [104] for a comparison of several region-based transform coding techniques to standard DCT coding, or [8] for a comparison of a region-based wavelet transform coder to standard wavelet transform coding. Often, an improvement in *subjective* (perceptual) image quality is claimed - the blocking effect (to which the human visual system is very sensitive) is usually avoided by region/object-based techniques. Nevertheless, the range of applicability of DCT-based coding is unrivalled.

In this respect, the edge-based coding method we have proposed here is still in line with previous efforts, although the approach is entirely different from most others. The edge-based coder performs better than JPEG at very low bit rates on certain types of images, but not on others. Instead of the blocking artifact, other types of distortions can be noted. Also, the applicability of our technique seems restricted as yet.

An important reason for the general lack of advances is that, although region-based and object-based techniques allow a more efficient description of the intensity surface inside regions or on moving objects, this has to be balanced against the extra amount of information to be transmitted that describes region contours or object shapes. Virtually no advances have been made in this shape coding problem during the last decade. In addition, in the opinion of the author, many people seem to make unrealistic assumptions about the number of bits necessary to code the edge location or curve shapes. A possible alternative is to code the shapes of curves in a lossy manner instead of exactly, e.g., using spline approximations. However, it is not fully clear yet how human observers perceive the distortions caused by approximative shape coding.

There seems little doubt that, currently, block-based coding schemes like MPEG-1 and MPEG-2 form the method of choice for new audio-visual services and practical applications, ranging from video telephony through home entertainment quality video up to HDTV. However, the use of region-based, object-based or edge-based schemes may become more preferable in the near future, when processing the image data on a semantic level becomes more important (i.e. in terms of its color, shape, texture and motion attributes).

An advantage of region-based or object-based methods compared to the edge-based method is that *decompression* is easier. A patch of the intensity surface inside a region is described *explicitly* in the region-based approach (e.g. by transform coefficients) and decompression is straightforward. In the edge-based approach, parts of the intensity surface are described *implicitly* by the smoothness constraint and decompression involves iterative interactions between different points.

An advantage of the edge-based approach over region-based and object-based approaches is that processing of the *compressed* image data is easier. In the edge-based approach, geometric as well as photometric information is stored in one-dimensional data structures, which allows for more efficient processing than with two-dimensional data structures. Certain operations, specifically contrast enhancement, edge sharpening and edge blurring, can be performed directly on the edge primitives.

6.4 Further research

From the previous sections, it is clear that practical application of an edge-based approach in image coding is still far away. Nonetheless, there is much room for improvement on our representation and algorithms. Possibilities for future research can roughly be divided into studies of the basic edge-based representation (as discussed in Chapter 2 and Chapter 3) on the one hand and studies of the application of the representation in image coding (as discussed in Chapter 4 and Chapter 5) on the other hand.

Edge parameter estimation can be improved using a multi-scale analysis, instead of a single-scale analysis, as has already been discussed. The use of more points in the estimation, to improve the robustness, might be reconsidered. In the general case (coding applications aside), it would also enrich the edge-based representation if the subpixel locations of edge points were included. More accurate localization of the edge curve would probably allow for a much better reconstruction of local edge intensities.

In the case of image sequences, the perceptual quality of the reconstructed images may be improved by the inclusion of a temporal smoothness measure in the global intensity surface reconstruction. This would increase the temporal consistency in the sequence and may alleviate some of the flickering problems reported in Chapter 5.

The question may be raised as to whether the reconstructed image quality might be improved by enforcing edge curves to be closed. In particular, if the representation were extended to include chrominance features (instead of just luminance values), this approach might be advantageous.

The parameter coding stage might be improved by the use of other compression techniques. The edge-based coder might also be extended by coding the residual error after the intensity reconstruction using a more conventional coding technique, like cosine or wavelet transform coding. Since most structural information is removed from the original image, the residual error image can probably be coded more efficiently by this type of technique than can the original. This approach has sometimes been called *two-component* coding. This technique would make the coding method more suitable for compression at higher bit rates and greatly improve the applicability.

Finally, theoretical work remains to be done in the mathematical analysis of edge-based representations. Uniqueness and stability properties of representations based on zero-crossings of Laplacian-of-Gaussian filtered images have been analyzed by Hummel and Moniot in [53]; a representation based on wavelet transform maxima has been proposed by Mallat and Zhong in [82]. Both are multi-scale representations. However, mathematical analysis is complicated due to the nonlinearities in the analysis stages. Even though completeness of these representations cannot be proven in general, numerical studies show that very good approximations to the original image can be recovered. Thus, local measurements on image intensity surfaces by partial derivative operators seem to constrain the global surface such that these measurements may be used as a (perceptually) complete representation of the original surface. Other algorithms in image processing based on these representations may benefit from the apparent richness of the information captured.

Appendix A
Convergence of Multi-Scale Relaxation

This appendix contains a theoretical analysis concerning the optimization algorithms of Chapter 3. This analysis supports our choice of optimization method and further explains the speed-ups obtained.

We study the special structure of the scale-k stiffness matrices $\mathbf{S}_k$ (for different k), and their use in iterative optimization algorithms of the relaxation type. Specifically, we analyze the convergence properties of iterative methods associated with these matrices. We compare three different algorithms, the first following the standard approach and the other two being variants of the multi-scale approach.

The rates of convergence of both the Gauss-Seidel and the Successive Overrelaxation method are higher than the rate of convergence of the simpler *Jacobi method* [98]. The Jacobi method can be studied easier. Therefore, we initially follow the analysis of Young in [129] of the Jacobi method for a model problem on a 2-D grid of size M+1 by N+1. Simplifying, we do not consider the effect of the data constraints but concentrate on the parts of the equations which have to do with improving the smoothness of the solution in the interior of large regions without constraints.

In this case, the scale-k iterations in the Jacobi method are (cf. Eq. (3.19) and Eq. (3.20)):

$$u^{i+1}(m,n) = \frac{1}{4}\Big(u^i(m+k,n) + u^i(m-k,n) + u^i(m,n+k) + u^i(m,n-k) \Big) . \qquad \text{(A.1)}$$

Alternatively, this can be written in matrix/vector notation as $\mathbf{u}^{i+1} = \mathbf{B}\mathbf{u}^i$, where the *iteration matrix* $\mathbf{B}$ has entries 1/4 on four diagonals (not on the main diagonal). $\mathbf{B}$ is related to the matrix $\mathbf{S}_k$. The eigenvalues of $\mathbf{B}$ determine the rate of convergence of the method [98]. Eigenvectors $v(m, n)$ and eigenvalues γ of the Jacobi iteration matrix $\mathbf{B}$ must satisfy:

$$\gamma \cdot v(m,n) = \frac{1}{4}\left(v(m+k,n) + v(m-k,n) + v(m,n+k) + v(m,n-k)\right) \qquad (A.2)$$

for $1 \le m \le M-1$, $1 \le n \le N-1$, and $v(m,n) = 0$ on the boundaries, cf. Young [129], Chapter 4, Eq. 6.10 and 6.11.

It can be shown that the functions

$$v_{p,q}(m,n) = \sin\frac{p\pi m}{M} \cdot \sin\frac{q\pi n}{N} \qquad (A.3)$$

satisfy the above equations, and that the corresponding eigenvalues are given by

$$\gamma_k(p,q) = \frac{1}{2} \cdot \left(\cos\frac{pk\pi}{M} + \cos\frac{qk\pi}{N}\right), \qquad (A.4)$$

for $1 \le p \le M-1$, $1 \le q \le N-1$.

Note that Eq. (A.1) can also be interpreted as a low-pass filtering operation. Further, the eigenvalues given by Eq. (A.4) together form the coefficients of the *Sine Transform* of the corresponding filter. The Sine Transform is related to the Fourier Transform and Eq. (A.4) can be interpreted as a *frequency spectrum* of the filter, for frequency components p, q. The eigenfunctions $v_{p,q}(m,n)$ are the basis functions of the Sine Transform. So, each iteration in Eq. (A.1) corresponds to a multiplication in the frequency domain with the spectrum of Eq. (A.4).

Standard algorithm

The above equations with $k = 1$ give the standard relaxation algorithm. It is known that standard relaxation algorithms generally require many iterations to eliminate the 'global' low-frequency components of the error in the solution, while the 'local' high-frequency components are eliminated very quickly. The error in the solution obeys Eq. (A.1) in the interior of the grid. The effect of repeated multiplication of the error spectrum with the filter spectrum given by Eq. (A.4) with $k = 1$ is illustrated in Figure A.1 for the one-dimensional case, with $M = 25$, for 8 iterations.

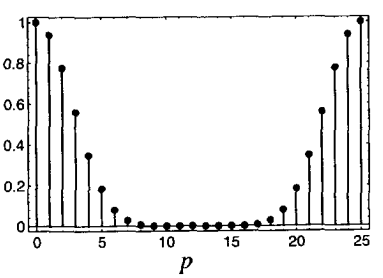


Figure A.1 The attenuation of the error spectrum by 8 iterations of the standard Jacobi algorithm in a one-dimensional situation with M = 25, for frequency components p.

Indeed, it can be seen from Figure A.1 that high-frequency components are effectively filtered out; low-frequency components are not suppressed as well.

Multi-scale approach - variant I

This algorithm follows directly from the multi-scale equations from Chapter 3. In this method, the right sides of Eq. (A.1) are summed up for $k = 1...K$, divided by K, and the resulting value is assigned to $u^{i+1}(m, n)$. This takes the same amount of computation as K iterations of the standard algorithm. A different iteration matrix **B** can now be defined, related to the matrix $\mathbf{S}^*$, which is the *sum* of scale-k stiffness matrices $\mathbf{S}_k$, for $k = 1...K$. The spectrum of the corresponding filter is given by:

$$\frac{1}{K} \cdot \sum_{k=1}^{K} \gamma_k(p, q) \ . \tag{A.5}$$

The attenuation effect on the error spectrum of one such iteration is shown in Figure A.2, for the one-dimensional case, with $M = 25$ and $K = 8$. One can see that this algorithm suppresses the low-frequency components of the error spectrum better than the standard algorithm.

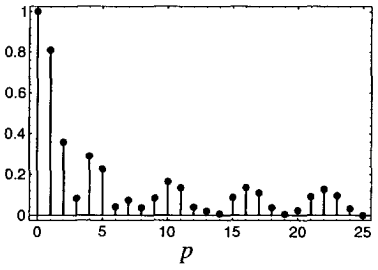


Figure A.2 The attenuation of the error spectrum by one iteration of variant I of the multi-scale approach in a one-dimensional situation with $M = 25$ and $K = 8$, for frequency components p.

Multi-scale approach - variant II

This algorithm is the one actually implemented in Chapter 3. Here, iterations Eq. (A.1) are performed sequentially, for $k = 1...K$. A sequence of such scale-k iterations is called a *cycle*. Again, this takes the same amount of computation as K iterations of the standard algorithm. The product of the spectra of the corresponding filters is given by:

$$\prod_{k=1}^{K} \gamma_k(p, q) \tag{A.6}$$

The attenuation effect on the error spectrum of these iterations is shown in Figure A.3 , for the one-dimensional case, with $M = 25$ and $K = 8$. One can see that this algorithm is much better

than the previous two in suppressing the low-frequency components of the error spectrum. The effect becomes even stronger for increasing K. Frequency components over a wide range of the spectrum are heavily attenuated by this method. It appears that all the cosines of different frequencies cancel each other out!

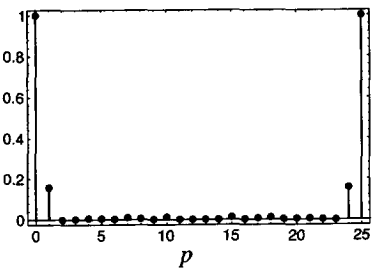


Figure A.3 The attenuation of the error spectrum by one cycle of variant II of the multi-scale approach in a one-dimensional situation with $M = 25$ and $K = 8$, for frequency components p.

The spectral radius

The eigenvalue of **B** with largest modulus is called the *spectral radius* ρ and determines the *asymptotic rate of convergence* of an iterative method. The eigenvalue with largest modulus is the eigenvalue with $p = q = 1$:

$$\rho_k = \max_{p,q} |\gamma_k(p, q)| = |\gamma_k(1, 1)| = \frac{1}{2} \cdot \left| \cos\frac{k\pi}{M} + \cos\frac{k\pi}{N} \right| . \tag{A.7}$$

The smaller the spectral radius, the higher is the rate of convergence.

For $k = 1$, this gives the spectral radius of the standard Jacobi method (cf. Young [129], Chapter 4, Eq. 6.16). However, it is obvious that the spectral radius of Eq. (A.7) decreases significantly for $k > 1$! Therefore, the convergence properties of scale-k iterations with $k > 1$ are much better than iterations with $k = 1$.

Similarly, the spectral radius of the iteration matrix associated with the first variant of the multi-scale approach is given by

$$\kappa = \frac{1}{K} \cdot \left| \sum_{k=1}^{K} \gamma_k(1, 1) \right| . \tag{A.8}$$

One can now compare the effect of one iteration of this variant with K iterations of the standard Jacobi method. The spectral radius of the first is given by Eq. (A.8). The effect of K iterations of the standard method corresponds to $(\rho_1)^K$, i.e., the spectral radius in Eq. (A.7) with $k = 1$, taken to the power K. One can observe that

$$\kappa \leq (\rho_1)^K$$

for large values of M and N and $1 < K < M/2$ and $1 < K < N/2$. Therefore, variant I of the multi-scale approach has a higher rate of convergence than the standard method.

Variant II of the multi-scale approach is even more efficient. There, one applies cycles consisting of K scale-k iterations, for $k = 1...K$. The effective spectral radius is given by:

$$\prod_{k=1}^{K} \rho_k \quad , \tag{A.9}$$

i.e., the product of the spectral radii ρ_k for $k = 1...K$. It can easily be shown that

$$\prod_{k=1}^{K} \rho_k \leq \kappa$$

and

$$\prod_{k=1}^{K} \rho_k \leq (\rho_1)^K$$

for typical values of N, M and K. Thus variant II of the multi-scale approach has the highest rate of convergence. This is the method implemented in Chapter 3, except that in Chapter 3, Gauss-Seidel or SOR iterations are used instead of Jacobi iterations.

The improvement of the implemented multi-scale variant over the former two methods can be quite significant for large values of K, as shown in the illustration in Figure A.4. It shows the three spectral radii discussed, for a grid with $M = N = 25$, for $K = 1...15$. Clearly, the convergence properties of the last method (curve *c*) are much better than either of the two former methods (curve *a* and *b*) for large K. All three methods take an equal amount of computation time on a serial computer.

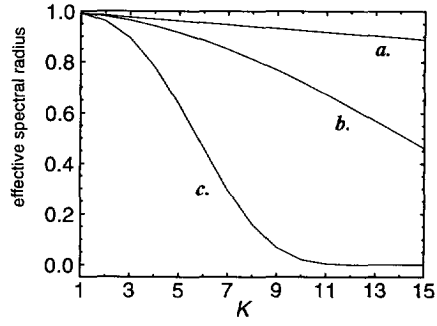


Figure A.4 (a) Spectral radius to the power K of the standard Jacobi iteration with $M = N = 25$, K between 1 and 15; (b) Spectral radius for the sum of scale-k matrices (scales up to K) with $M = N = 25$, K between 1 and 15; (c) Product of spectral radii of scale-k iterations (scales up to K) with $M = N = 25$, K between 1 and 15.

Appendix B
Edge Primitive Coder Design

This appendix contains additional information concerning the design of the edge primitive coder of Chapter 4.

B.1 Parameter signal auto-correlation

Suppose $f(n)$ is an input parameter signal, with $0 \le n \le N-1$. The mean value μ_f of f is estimated by

$$\mu_f = \frac{1}{N} \sum_{n=0}^{N-1} f(n) \ . \tag{B.1}$$

The variance σ_f^2 and auto-correlation coefficients $\rho_f(k)$, k=0, 1, 2, ... , are estimated by

$$r_f(k) = \frac{1}{N} \sum_{n=0}^{N-1} (f(n) - \mu_f)\ (f(n+k) - \mu_f) \ , \tag{B.2}$$

$$\sigma_f^2 = r_f(0), \qquad \rho_f(1) = \frac{r_f(1)}{r_f(0)}, \qquad \rho_f(2) = \frac{r_f(2)}{r_f(0)}, \qquad \text{etc.} \tag{B.3}$$

Odd symmetric extension is used beyond the end of the signal to obtain all the terms in the sum of Eq. (B.2), and weighting of the last terms is applied. That is, when $n + k > N - 1$ in a particular term, then the weight

$$\frac{N-n}{k+1}$$

is multiplied with this term. The auto-correlation coefficients in the tables in Chapter 4 give the *average* auto-correlation over all parameter signals of the same type in *one* image.

B.2 Prediction error quantization

B.2.1 Generalized Gaussian distribution

The Generalized Gaussian probability density function $p(x; \mu, \sigma, \alpha)$, with mean μ, deviation σ and shape parameter α is given by

$$p(x;\mu, \sigma, \alpha) = c_1 \exp\left(-\left|c_2\left(\frac{x-\mu}{\sigma}\right)\right|^{\alpha}\right) \tag{B.4}$$

with

$$c_1 = \frac{c_2 \alpha}{2\sigma\Gamma(\frac{1}{\alpha})} \quad \text{and} \quad c_2 = \sqrt{\frac{\Gamma(\frac{3}{\alpha})}{\Gamma(\frac{1}{\alpha})}} \tag{B.5}$$

and $\Gamma(.)$ is the Gamma function [38]. The shape parameter α determines the shape of the pdf: if α tends to 0, the shape of the pdf becomes very peaked, approaching a delta function; when α grows bigger, the shape of the pdf becomes flatter; special cases arise for $\alpha = 1.0$ and $\alpha = 2.0$, when the pdf equals the Laplacian and Gaussian distribution respectively.

B.2.2 Chi-square test statistics

This subsection contains all the results of Chi-square testing of empirical prediction error data from a test-image set. The image set consisted of fourteen different images, of which seven were from a videophone type scene, and seven were various others. The number of prediction coefficients throughout the experiments was two. The results in Table B.1, Table B.2 and Table B.3 are for downsampling zero, one and two times. The data is matched with Generalized Gaussian distributions with various shape parameters.

Table B.1 Chi-square test statistic results when matching the distribution of the prediction error of parameter signals to the Generalized Gaussian distribution for various α. The parameter signals are not downsampled. The mean and standard deviation of the prediction error are given for completeness.

Type	μ	σ	α			
			0.50	0.75	1.00	2.00
m	0.04	6.34	0.17	0.06	0.10	$1.55\ 10^{17}$
c	-0.14	6.73	0.09	0.03	1.71	$6.26\ 10^{32}$
w	0.23	2.12	0.11	0.04	10.60	$2.17\ 10^{49}$

Table B.2 Chi-square test statistic results when matching the distribution of the prediction error of parameter signals to the Generalized Gaussian distribution for various α. The parameter signals are downsampled once. The mean and standard deviation of the prediction error are given for completeness.

Type	μ	σ	α			
			0.50	0.75	1.00	2.00
m	0.14	6.53	0.44	0.74	72.04	$1.01\ 10^{42}$
c	0.30	11.02	0.10	0.03	0.11	$1.20\ 10^{10}$
w	0.34	2.57	0.25	0.08	0.13	$9.25\ 10^{22}$

Table B.3 Chi-square test statistic results when matching the distribution of the prediction error of parameter signals to the Generalized Gaussian distribution for various α. The parameter signals are downsampled twice. The mean and standard deviation of the prediction error are given for completeness.

Type	μ	σ	α			
			0.50	0.75	1.00	2.00
m	0.47	11.71	0.30	0.11	0.11	$1.39\ 10^{7}$
c	1.04	28.94	0.13	0.26	0.45	1.06
w	0.52	2.70	0.39	0.15	0.09	$17.03\ 10^{3}$

B.2.3 Uniform Threshold Quantizers

The relevant performance characteristics of the quantizers (Uniform Threshold Quantizers, see [38]) used in our experiments are given in Table B.4. All the quantizers used were designed for a source with Generalized Gaussian pdf with mean 0.0, variance 1.0 and shape parameter 0.75. The VLC code words associated with each representation level are assigned according to Huffman's method. As an example, the quantizer with 15 representation levels is specified completely in Table B.5.

Table B.4 Performance characteristics of Uniform Threshold Quantizers for a source with Generalized Gaussian pdf with shape parameter 0.75 and variance 1.0. The entropy and expected rate using Huffman coding are given in bits per sample; the expected distortion is given in the mean square error sense.

# Repr. Levels	Entropy (bps)	Rate (bps)	Distortion (mse)
13	0.630	1.171	0.350
15	0.752	1.219	0.298
25	1.399	1.559	0.129

Table B.5 Uniform Threshold Quantizer with 15 representation levels designed for a source with Generalized Gaussian pdf with shape parameter 0.75 and variance 1.0 and corresponding Huffman code words.

Repr. Level	Decision Level	Probability	Code word
-18.57036781	-	0.00000005	111111111111
-15.44915104	-17.19225311	0.00000030	11111111110
-12.78677559	-14.54729176	0.00000244	1111111110
-10.11973953	-11.90232944	0.00002216	11111110
-7.44496250	-9.25736713	0.00023444	111110
-4.75510120	-6.61240530	0.00312438	1110
-2.02248645	-3.96744299	0.06467993	10
-0.00000000	-1.32248104	0.86387289	0
2.02248645	1.32248104	0.06467993	110
4.75510120	3.96744299	0.00312438	11110
7.44496250	6.61240530	0.00023444	1111110
10.11973953	9.25736713	0.00002216	11111110
12.78677559	11.90232944	0.00000244	11111111110
15.44915104	14.54729176	0.00000030	1111111111110
18.57036781	17.19225311	0.00000005	1111111111110

B.3 Edge location coding

This section of the appendix contains tables with empirical frequency distributions of Freeman chain codes using a zero-order, first-order and second-order Markov model for the source. These are Table B.6, Table B.8 and Table B.10. Also, the entropy (in bits per symbol) corresponding to each table is mentioned.

On the basis of each frequency distribution, code words can be assigned by the Huffman coding method. Here, Table B.7, Table B.9 and Table B.11 give only the *lengths* of the code words resulting from Huffman's assignment procedure. Also, the expected rate (in bits per symbol) on the basis of this code word assignment is mentioned.

The training image set consisted of eleven different images, of which six were of a videophone type scene, one contained some industrial parts, one was of a building scene, one contained a flowerbed, one contained a woman with hat and one shows a microscope recording of gold particles in glass.

Table B.6 Frequency distribution for zero-order Markov model. In the zero-order model, the values of previous chain codes are not taken into account. Therefore, there is only one state (denoted by 'x' here). The Markov model returns to the same state each time a chain code has been processed. The entropy is 2.771 bps. Note that the frequency distribution is not uniform.

State	Chain code							
	0	1	2	3	4	5	6	7
x	0.117	0.044	0.062	0.051	0.187	0.151	0.266	0.122

Table B.7 Huffman code word lengths for zero-order Markov model. In the zero-order model, the values of previous chain codes are not taken into account. Therefore, there is only one state (denoted by 'x' here). The Markov model returns to the same state each time a chain code has been processed. The expected rate is 2.799 bps. Code words of different length are assigned to the different symbols. E.g. the chain code '6' (with frequency 0.266) is assigned a code word of length 2, while the chain code '3' (with frequency 0.051) is assigned a code word of length 5.

State	Chain code							
	0	1	2	3	4	5	6	7
x	3	5	4	5	2	3	2	3

Table B.8 Frequency distribution for first-order Markov model. In the first-order model, the value of the previously processed chain code determines the state. Therefore, there are eight different states. A transition to the next state occurs after a chain code is processed. The entropy is 1.627 bps. Note that entries along the main diagonal have the highest frequencies. E.g. in the state '0', the frequency of another '0' chain code occurring in the data is highest, since most curves contain smooth parts.

State	Chain code							
	0	1	2	3	4	5	6	7
0	0.071	0.015	0.000	0.000	0.006	0.000	0.000	0.021
1	0.010	0.010	0.011	0.000	0.000	0.003	0.000	0.002
2	0.000	0.009	0.033	0.012	0.000	0.000	0.001	0.000
3	0.000	0.000	0.009	0.017	0.018	0.000	0.000	0.002
4	0.009	0.000	0.000	0.014	0.119	0.048	0.000	0.000
5	0.000	0.010	0.000	0.001	0.047	0.048	0.046	0.003
6	0.000	0.000	0.010	0.000	0.000	0.047	0.176	0.044
7	0.027	0.001	0.000	0.007	0.000	0.001	0.044	0.045

Table B.9 Huffman code word lengths for first-order Markov model. In the first-order model, the value of the previously processed chain code determines the state. Therefore, there are eight different states. A transition to the next state occurs after a chain code is processed. The expected rate is 1.764 bps. Note that short code words are assigned to entries along the main diagonal. E.g. because the frequency of a '0' chain code in the state '0' is highest, it is assigned a code word of length 1.

State	Chain code							
	0	1	2	3	4	5	6	7
0	1	3	7	7	4	5	6	2
1	2	2	2	5	6	3	6	4
2	7	3	1	2	6	5	4	7
3	7	5	3	2	1	6	7	4
4	4	7	5	3	1	2	6	7
5	6	3	6	5	2	2	2	4
6	7	7	4	6	5	2	1	3
7	3	6	7	4	7	5	2	1

Table B.10 Partial frequency distribution for second-order Markov model. In the second-order model, the values of the two previously processed chain codes determine the state. Therefore, there are sixty-four different states, of which only twenty-four states are shown here. The entries corresponding to most other states show a very low frequency of occurrence. The entropy is 1.487 bps.

State	Chain code							
	0	1	2	3	4	5	6	7
0 0	0.052	0.009	0.000	0.000	0.004	0.000	0.000	0.011
0 1	0.007	0.004	0.002	0.000	0.000	0.002	0.000	0.001
0 7	0.013	0.000	0.000	0.002	0.000	0.000	0.002	0.005
1 0	0.005	0.005	0.000	0.000	0.001	0.000	0.000	0.000
1 1	0.002	0.004	0.003	0.000	0.000	0.000	0.000	0.001
1 2	0.000	0.004	0.006	0.001	0.000	0.000	0.000	0.000
2 1	0.000	0.002	0.006	0.000	0.000	0.000	0.000	0.000
2 2	0.000	0.004	0.023	0.006	0.000	0.000	0.001	0.000
2 3	0.000	0.000	0.006	0.004	0.002	0.000	0.000	0.000
3 2	0.000	0.000	0.004	0.005	0.000	0.000	0.000	0.000
3 3	0.000	0.000	0.002	0.009	0.006	0.000	0.000	0.001
3 4	0.001	0.000	0.000	0.005	0.011	0.003	0.000	0.000
4 3	0.000	0.000	0.001	0.003	0.009	0.000	0.000	0.001
4 4	0.006	0.000	0.000	0.008	0.082	0.029	0.000	0.000
4 5	0.000	0.004	0.000	0.000	0.031	0.011	0.003	0.001
5 4	0.003	0.000	0.000	0.002	0.029	0.016	0.000	0.000
5 5	0.000	0.004	0.000	0.000	0.010	0.024	0.012	0.001
5 6	0.000	0.000	0.002	0.000	0.000	0.021	0.022	0.003
6 5	0.000	0.003	0.000	0.000	0.005	0.012	0.028	0.001
6 6	0.000	0.000	0.006	0.000	0.000	0.022	0.133	0.024
6 7	0.002	0.000	0.000	0.003	0.000	0.001	0.028	0.012
7 0	0.015	0.002	0.000	0.000	0.002	0.000	0.000	0.009
7 6	0.000	0.000	0.003	0.000	0.000	0.002	0.024	0.017
7 7	0.009	0.001	0.000	0.003	0.000	0.001	0.012	0.024

Table B.11 Partial table of Huffman code word lengths for second-order Markov model. In the second-order model, the values of the two previously processed chain codes determine the state. Therefore, there are sixty-four different states, of which only twenty-four states are shown here, corresponding to the states shown in the previous table. The expected rate is 1.629 bps.

State	Chain code							
	0	1	2	3	4	5	6	7
0 0	1	3	7	7	4	5	6	2
0 1	1	2	3	6	7	4	7	5
0 7	1	5	7	3	7	6	4	2
1 0	1	2	7	7	3	5	6	4
1 1	3	1	2	6	7	5	7	4
1 2	7	2	1	3	6	5	4	7
2 1	4	2	1	6	7	5	7	3
2 2	7	3	1	2	6	5	4	7
2 3	7	5	1	2	3	6	7	4
3 2	7	3	2	1	6	5	4	7
3 3	7	6	3	1	2	5	7	4
3 4	4	7	5	2	1	3	6	7
4 3	7	5	4	2	1	6	7	3
4 4	4	7	5	3	1	2	6	7
4 5	7	3	7	6	1	2	4	5
5 4	3	7	5	4	1	2	6	7
5 5	7	4	7	6	3	1	2	5
5 6	7	7	4	6	5	2	1	3
6 5	7	4	7	6	3	2	1	5
6 6	7	7	4	6	5	3	1	2
6 7	4	6	7	3	7	5	1	2
7 0	1	4	7	7	3	5	6	2
7 6	7	7	3	6	5	4	1	2
7 7	3	5	7	4	7	6	2	1

References

[1] J. K. Aggarwal and N. Nandhakumar, August 1988, "On the computation of motion from sequences of images - A review," *Proceedings IEEE*, vol. 76, no. 8, pp. 917-936

[2] K. Aizawa, T. Saito, and H. Harashima, 1989, "Model-based analysis-synthesis image coding (MBASIC) system for a person's face," *Signal Processing: Image Communication*, vol. 1, no. 2, pp. 139-152

[3] R. Alter-Gartenberg, F. O. Huck and R. Narayanswamy, May 1990, "Image recovery from edge primitives," *J. Opt. Soc. of Am. A*, vol. 7, no. 5, pp. 898-911

[4] R. Alter-Gartenberg and R. Narayanswamy, 1990, "Image coding by edge primitives," *Visual Communications and Image Processing '90*, Proc. SPIE 1360, ed. M. Kunt, pp. 1608-1619

[5] D. Anastassiou, September 1994, "Current status of the MPEG-4 standardization effort," *Visual Communications and Image Processing '94*, Proc. SPIE 2308, ed. A. K. Katsaggelos, pp. 16-24

[6] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, March 1991, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 209-215

[7] D. H. Ballard and C. M. Brown, 1982, *Computer Vision*, Prentice-Hall, Englewood Cliffs

[8] H. J. Barnard, 1994, *Image and Video Coding Using a Wavelet Decomposition*, Ph.D. Thesis, Delft University of Technology, Delft

[9] P. J. L. van Beek, B. Sankur, and J. C. A. van der Lubbe, April 1992, "Contour extraction and matching for image sequence coding," *4th International Conference*

on Image Processing and its Applications, IEE, Maastricht, the Netherlands, pp. 109-114

[10] P. J. L. van Beek, M. J. T. Reinders, B. Sankur, and J. C. A. van der Lubbe, November 1992, "Semantic segmentation of videophone image sequences," *Visual Communications and Image Processing '92*, Proc. SPIE 1818, ed. P. Maragos, pp. 1182-1193

[11] P. J. L. van Beek, J. J. Gerbrands, J. C. A. van der Lubbe, and B. Sankur, September 1993, "Edge-based multi-scale image representation," *Proc. of the eighth workshop on Image and Multi-Dimensional Signal Processing*, IEEE Signal Processing Society, Cannes, France, pp. 152-153

[12] F. Bergholm, November 1987, "Edge focussing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 6, pp. 726-741

[13] P. B. Berra, F. Golshani, R. Mehrotra, and O. R. Liu Sheng, August 1993, "Guest Editors' Introduction: Multimedia Information Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 4, pp. 545-549

[14] M. Bertero, T. A. Poggio, and V. Torre, August 1988, "Ill-posed problems in early vision," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 869-889

[15] V. Berzins, 1984, "Accuracy of Laplacian edge detectors," *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 195-210

[16] J. Biemond, R. L. Lagendijk, and R. M. Mersereau, May 1990, "Iterative methods for image deblurring," *Proceedings of the IEEE*, vol. 78, no. 5, pp. 856-883

[17] M. J. Biggar, O. J. Morris and A. G. Constantinides, April 1988, "Segmented-image coding: performance comparison with the discrete cosine transform," *IEE Proceedings Pt. F*, vol. 135, no. 2, pp. 1221-132

[18] A. Blake and A. Zisserman, 1987, *Visual Reconstruction*, M.I.T. Press, Cambridge

[19] D. E. Boekee and J. C. A. van der Lubbe, 1988, *Informatietheorie* (in Dutch), Delftse Uitgevers Maatschappij, Delft

[20] K. R. Boff, L. Kaufman and J. P. Thomas (eds.), 1986, *Handbook of Perception and Human Performance, vol. I: Sensory Processes and Perception*, John Wiley and Sons, New York

[21] G. Borgefors, 1984, "Distance transformations in arbitrary dimensions," *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 321-345

[22] K. L. Boyer and S. Sarkar, January 1994, "On the localization performance measure and optimal edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 106-108

[23] M. Brady and B. K. P. Horn, April 1983, "Rotationally symmetric operators for surface interpolation," *Computer Vision, Graphics and Image Processing*, vol. 22, no. 1, pp. 70-94

[24] P. J. Burt and E. H. Adelson, April 1983, "The Laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. 31, no. 4, pp. 532-540

[25] J. Canny, November 1986, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698

[26] S. Carlsson, July 1988, "Sketch based coding of gray level images," *Signal Processing*, vol. 15, no. 1, pp. 57-83

[27] G. Caronna and G. Zarone, December 1988, "Some experiments in interframe contour coding," *Alta Frequenza*, vol. LVII, no. 10, pp. 603-609

[28] CCITT, August 1990, "Draft Revision of Recommendation H.261: Video Codec for Audiovisual Services at px64 kbit/s," *Image Communication*, vol. 2, no. 2, pp. 221-239

[29] J. S. Chen and G. Medioni, February 1989, "Detection, localization and estimation of edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 191-198

[30] T. M. Chin, M. R. Luettgen, W. C. Karl, and A. S. Willsky, 1993, "An estimation theoretic perspective on image processing and the calculation of optical flow," in: *Motion Analysis and Image Sequence Processing*, eds. M. Ibrahim Sezan and R. L. Lagendijk, Kluwer Academic Publishers, pp. 23-51

[31] T. N. Cornsweet, 1970, *Visual Perception*, Academic Press, New York

[32] A. Cumani, A. Guiducci and P. Grattoni, 1991, "Image description of dynamic scenes," *Pattern Recognition*, vol. 24, no. 7, pp. 661-673

[33] P. E. Danielsson, November 1980, "Euclidean Distance Mapping," *Computer Graphics and Image Processing*, vol. 14, no. 3, pp. 227-248

[34] R. Deriche, January 1990, "Fast algorithms for low-level vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 78-87

[35] H. Derin and W. S. Cole, 1986, "Segmentation of textured images using Gibbs Random Fields," *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 72-98

[36] J. N. Driessen, 1992, *Motion Estimation For Digital Video*, Ph.D. Thesis, Delft University of Technology, Delft

[37] S. L. Eddins, April 1993, "Three-component image compression with frequency-weighted texture coding and edge modeling," *Proceedings 1993 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, USA, vol. V, pp. 369-372

[38] N. Farvardin and J. W. Modestino, May 1984, "Optimum quantizer performance for a class of non-gaussian memoryless sources," *IEEE Transactions on Information Theory*, vol. 30, no. 3, pp. 485-497

[39] M. A. Fischler and O. Firschein (eds), 1987, *Readings in Computer Vision (Issues, Problems, Principles, and Paradigms)*, Morgan Kaufmann Publishers, Los Altos, California

[40] R. Forchheimer and T. Kronander, December 1989, "Image coding - from waveforms to animation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 2008-2023

[41] H. Freeman, 1970, "Boundary encoding and processing," in: *Picture Processing and Psychopictorics*, eds. B. S. Lipkin and A. Rosenfeld, Academic Press, New York, pp. 241-266

[42] M. Gilge, T. Engelhardt, and R. Mehlan, 1989, "Coding of arbitrarily shaped image segments based on a generalized orthogonal transform," *Signal Processing: Image Communication*, vol. 1, no. 1, pp. 153-180

[43] B. Girod, 1992, "Psychovisual aspects of image communication," *Signal Processing*, vol. 28, no. 3, pp. 239-251

[44] G. Giunta, T. R. Reed and M. Kunt, 1991, "Image sequence coding using oriented edges," *Signal Processing: Image Communication*, vol. 2, no. 4, pp. 429-439

[45] I. S. Gradshteyn and I. M. Ryzhik, 1965, *Table of Integrals, Series and Products*, Academic Press, New York

[46] D. N. Graham, March 1967, "Image transmission by two-dimensional contour coding," *Proceedings of the IEEE*, vol. 55, no. 3, pp. 336-346

[47] P. Grattoni and A. Guiducci, 1990, "Contour coding for image description," *Pattern Recognition Letters*, vol. 11, pp. 95-105

[48] W. E. L. Grimson, April 1983, "An implementation of a computational theory of visual surface interpolation," *Computer Vision, Graphics and Image Processing*, vol. 22, pp. 39-69

[49] R. M. Haralick, January 1984, "Digital step edges from zero crossing of second directional derivatives," *IEEE Transactions On Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 58-68

[50] E. C. Hildreth, April 1983, "The detection of intensity changes by computer and biological vision systems," *Computer Vision, Graphics and Image Processing*, vol. 22, no. 1, pp. 1-27

[51] E. C. Hildreth, 1986, "Computing the velocity field along contours," in: *Motion: Representation and Perception*, eds. N. I. Badler and J. K. Tsotsos, Elsevier Science Publishing, pp. 121-127

[52] B. K. P. Horn, 1986, *Robot Vision*, The MIT Press, Cambridge, Massachusetts

[53] R. Hummel and R. Moniot, December 1989, "Reconstructions from zero crossings in scale space," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 2111-2130

[54] B. R. Hunt, February 1980, "Nonstationary statistical image models (and their application to image data compression)," *Computer Graphics and Image Processing*, vol. 12, pp. 173-186

[55] ISO/IEC 11172 (MPEG-1) Draft International Standard, October 1992, *"Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5 Mbit/s"*

[56] ISO/IEC 13818 (MPEG-2) Committee Draft, November 1993, *"Generic Coding of Moving Pictures and Associated Audio"*

[57] A. E. Jacquin, October 1993, "Fractal image coding: a review," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1451-1465

[58] A. K. Jain, March 1981, "Image data compression: a review," *Proceedings of the IEEE*, vol. 69, no. 3, pp. 349-389

[59] A. K. Jain, 1989, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs

[60] N. S. Jayant and P. Noll, 1984, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Englewood Cliffs

[61] R. Kakarala and A. O. Hero, July 1992, "On achievable accuracy in edge localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 777-781

[62] B. Kamgar-Parsi, A. Margalit and A. Rosenfeld, March 1991, "Matching general polygonal arcs," *CVGIP: Image understanding*, vol. 53, no. 2, pp. 227-234

[63] N. B. Karayiannis and A. N. Venetsanopoulos, 1991, "Image interpolation based on variational principles," *Signal Processing*, vol. 25, pp. 259-288

[64] A. K. Katsaggelos (ed.), September 1994, *Visual Communications and Image Processing '94*, Proc. SPIE 2308

[65] V. Kayargadde and J.-B. Martens, November 1994, "Estimation of edge parameters and image blur using polynomial transforms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 442-461

[66] M. W. Koch and R. L. Kashyap, November 1989, "Matching polygon fragments," *Pattern Recognition Letters*, vol. 10, no. 5, pp. 297-308

[67] J. J. Koenderink, 1984, "The structure of images," *Biological Cybernetics*, vol. 50, pp. 363-370

[68] J. Koplowitz and V. Greco, December 1994, "On the edge location error for local maximum and zero-crossing edge detectors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 12, pp. 1207-1212

[69] M. Kunt, A. Ikonomopoulos and M. Kocher, April 1985, "Second-generation image coding techniques," *Proceedings IEEE*, vol. 73, no. 4, pp. 549-574

[70] M. Kunt, M. Bénard, and R. Leonardi, November 1987, "Recent results in high-compression image coding," *IEEE Transactions on Circuits and Systems*, vol. 34, no. 11, pp. 1306-1336

[71] S.-H. Lai, C.-W. Fu, and S. Chang, April 1992, "A generalized depth estimation algorithm with a single image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 4, pp. 405-411

[72] E. H. Land and J. J. McCann, 1971, "Lightness and retinex theory," *J. Opt. Soc. Am.*, vol. 61, no. 1, pp. 1-11

[73] M. S. Landy and Y. Cohen, June 1985, "Vectorgraph coding: efficient coding of line drawings," *Computer Vision, Graphics and Image Processing*, vol. 30, no. 3, pp. 331-344

[74] D. Lee, June 1989, "Edge detection, classification and measurement," *Proc. of IEEE Comp. Soc. Conference on Computer Vision and Pattern Recognition CVPR '89*, San Diego, California, pp. 2-10

[75] F.-C. Leou and Y.-C. Chen, 1991, "A contour-based image coding technique with its texture information reconstructed by polyline representation," *Signal Processing*, vol. 25, pp. 81-89

[76] M. D. Levine, 1985, *Vision in Man and Machine*, McGraw-Hill, New York

[77] J. S. Lim, 1990, *Two-Dimensional Signal and Image Processing*, Prentice-Hall International, London

[78] C.-C. Lu and J. G. Dunham, October 1991, "Highly efficient coding schemes for contour lines based on chain code representations," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1511-1514

[79] Y. Lu and R. Jain, April 1989, "Behavior of edges in scale space," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 337-356

[80] M. R. Luettgen, W. C. Karl, and A. S. Willsky, January 1994, "Efficient multiscale regularization with applications to the computation of optical flow," *IEEE Transactions on Image Processing*, vol. 3, no. 1, pp. 41-63

[81] S. Mallat and S. Zhong, May 1989, "Wavelet maxima representation," *Wavelets and Applications, Proceedings of the International Conference*, ed. Y. Meyer, Marseille, pp. 207-284

[82] S. Mallat and S. Zhong, July 1992, "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 710-732

[83] D. Marr and E. Hildreth, 1980, "Theory of edge detection," *Proceedings Royal Society of London B*, vol. 207, pp. 187-217

[84] D. Marr and H. K. Nishihara, 1978, "Visual information processing: Artificial Intelligence and the sensorium of sight," *Technology Preview*, vol. 81, no. 1, pp. 2-23

[85] H. G. Musmann, M. Hötter, and J. Ostermann, 1989, "Object-oriented analysis-synthesis coding of moving images," *Signal Processing: Image Communication*, vol. 1, no. 2, pp. 117-138

[86] A. N. Netravali and B. G. Haskell, 1988, *Digital Pictures - Representation and Compression*, Plenum Press, New York

[87] H. Neumann, K. Ottenberg, and H. S. Stiehl, 1992, "Finding and describing local structure in discrete two-dimensional computed tomograms," *11th IAPR International Conference on Pattern Recognition*, the Hague, the Netherlands, vol. 3, pp. 408-412

[88] D. Noton and L. Stark, June 1971, "Eye movements and visual perception," *Scientific American*, vol. 224, no. 6, pp. 34-43

[89] K. Ottenberg, H. Neumann, and H. S. Stiehl, August 1992, "Quantitative description and reconstruction of intensity functions using scale-space and multi-resolution processing," *Signal Processing VI: Theories and Applications*, Proc. of the EUSIPCO '92, Brussels, vol. III, Elsevier, pp. 1425-1428

[90] P. J. van Otterloo, 1991, *A Contour-Oriented Approach to Shape Analysis*, Prentice Hall

[91] A. Papoulis, 1980, *Circuits and Systems (a Modern Approach)*, Holt, Rinehart and Winston

[92] D. E. Pearson and J. A. Robinson, April 1985, "Visual communication at very low data rates," *Proceedings of the IEEE*, vol. 73, no. 4, pp. 795-812

[93] W. B. Pennebaker and J. L. Mitchell, 1983, *JPEG - Still Image Data Compression Standard*, Van Nostrand Reinhold, New York

[94] A. P. Pentland, July 1987, "A new sense for depth of field," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp. 523-531

[95] A. P. Pentland, 1992, "Surface interpolation using wavelets," in: *Computer Vision - ECCV '92, Proceedings Second European Conference on Computer Vision*, Italy, ed. G. Sandini, Springer Verlag, Berlin, pp. 615-619

[96] A. Pentland, R. W. Picard and S. Sclaroff, February 1994, "Photobook: Tools for Content-Based Manipulation of Image Databases," *Storage and Retrieval for Image and Video Databases II*, Proc. SPIE 2185, eds. W. Niblack and R. C. Jain, pp. 34-37

[97] T. Poggio, V. Torre and C. Koch, September 1985, "Computational vision and regularization theory," *Nature*, vol. 317, no. 26, pp. 314-319

[98] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1992, *Numerical Recipes in C: The Art of Scientific Computing*, Second Edition, Cambridge University Press

[99] V. S. Ramachandran, August 1988, "Perceiving shape from shading," *Scientific American*, pp. 58-65

[100] S. V. Raman, S. Sarkar, and K. L. Boyer, June 1991, "Tissue boundary refinement in magnetic resonance images using contour-based scale space matching," *IEEE Transactions on Medical Imaging*, vol. 10, no. 2, pp. 109-121

[101] F. Ratliff, June 1972, "Contour and contrast," *Scientific American*, vol. 226, no. 6, pp. 91-101

[102] M. J. T. Reinders, P. J. L. van Beek, B. Sankur and J. C. A. van der Lubbe, March 1995, "Facial feature localization and adaptation of a generic face model for model-based coding," *Signal Processing: Image Communication*, vol. 7, no. 1, pp. 57-74

[103] M. J. T. Reinders, 1995, *Model Adaptation for Image Coding*, Ph.D. Thesis, Delft University of Technology, Delft

[104] K. J. Rijkse, E. Jensen, I. Lagendijk, and P. J. L. van Beek, October 1994, "Coding of arbitrarily shaped image segments," *Workshop on Image Analysis and Synthesis for Image Coding*, Heinrich-Hertz-Institut für Nachrichtentechnik, Berlin

[105] O. Rioul and M. Vetterli, October 1991, "Wavelets and signal processing," *IEEE Signal Processing Magazine*, pp. 14-38

[106] A. Rosenfeld (ed.), 1984, *Multiresolution Image Processing and Analysis*, Springer Series in Information Sciences, vol. 12, Springer-Verlag, New-York / Berlin

[107] P. M. T. Smeele, 1994, *Perceiving Speech: Integrating Auditory and Visual Speech*, Ph.D. Thesis, Delft University of Technology, Delft

[108] I. S. Sokolnikoff and R. M. Redheffer, 1958, *Mathematics of Physics and Modern Engineering*, McGraw-Hill Book Company, New York

[109] M. Soryani and R. J. Clarke, 1992, "Segmented coding of digital image sequences," *IEE Proceedings Pt. I*, vol. 139, no. 2, pp. 212-218

[110] Q. Summerfield, A. MacLeod, M. McGrath, and M. Brooke, 1989, "Lips, teeth, and the benefits of lipreading," in: *Handbook of Research on Face Processing*, ed. H.D. Ellis, Elsevier Science Publishers B.V. (North-Holland), pp. 223-233

[111] R. Szeliski, 1989, *Bayesian Modeling of Uncertainty in Low-Level Vision*, Kluwer Academic Publishers, Boston, Massachusetts

[112] R. Szeliski, June 1990, "Fast surface interpolation using hierarchical basis functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 6, pp. 513-528

[113] H. D. Tagare and R. J. P. deFigueiredo, December 1990, "On the localization performance measure and optimal edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1186-1190

[114] H. D. Tagare and R. J. P. deFigueiredo, January 1994, "Reply to 'On the localization performance measure and optimal edge detection'," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 1, pp. 108-110

[115] D. Terzopoulos, 1984, "Multilevel reconstruction of visual surfaces: variational principles and finite-element representations," in: *Multiresolution Image Processing and Analysis*, ed. A. Rosenfeld, Springer-Verlag, pp. 237-310

[116] D. Terzopoulos, March 1986, "Image analysis using multigrid relaxation methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 129-139

[117] D. Terzopoulos, July 1986, "Regularization of inverse visual problems involving discontinuities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 413-424

[118] A. N. Tikhonov and V. Y. Arsenin, 1977, *Solutions of Ill-Posed Problems*, V. H. Winston & Sons, Washington, D.C.

[119] V. Torre and T. Poggio, March 1986, "On edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 147-162

[120] R. M. J. A. Verbarendse, April 1994, *Interframe Contour Analysis and Contour Coding*, Master's Thesis (in Dutch), Delft University of Technology, Delft

[121] P. W. Verbeek, H. A. Vrooman, and L. J. Van Vliet, October 1988, "Low-level image processing by max-min filters," *Signal Processing*, vol. 15, no. 3, pp. 249-258

[122] L. J. van Vliet, 1993, *Grey-Scale Measurements in Multi-Dimensional Digitized Images*, Ph.D. Thesis, Delft University of Technology, Delft University Press, Delft

[123] G. K. Wallace, April 1991, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30-44

[124] W. L. G. van Warmerdam and V. R. Algazi, September 1989, "Describing 1-D intensity transitions with Gaussian derivatives at the resolutions matching the transition widths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 9, pp. 973-977

[125] D. J. Williams and M. Shah, July 1993, "Edge characterization using normalized edge detector," *CVGIP: Graphical models and image processing*, vol. 55, no. 4, pp. 311-318

[126] A. P. Witkin, August 1983, "Scale-space filtering," *Proceedings 8th Int. Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 1019-1022

[127] S. Wolfram, 1991, *Mathematica: A System for Doing Mathematics by Computer*, Second Edition, Addison-Wesley Publishing Company, Redwood City

[128] J. K. Yan and D. J. Sakrison, November 1977, "Encoding of images based on a two-component source model," *IEEE Transactions on Communications*, vol. 25, no. 11, pp. 1315-1322

[129] D. M. Young, 1971, *Iterative Solution of Large Linear Systems*, Academic Press, New York

[130] A. Yuille and T. Poggio, January 1986, "Scaling theorems for zero crossings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 15-25

List of Symbols and Abbreviations

General Notation

x	a scalar variable
$\|x\|$	scalar magnitude of x
$\mathbf{x}$	a vector variable
$\mathbf{x}^T$	transpose of $\mathbf{x}$
$\|\mathbf{x}\|$	vector length / magnitude of $\mathbf{x}$
$\mathbf{X}$	a matrix
$*$	convolution operator in both the continuous and the discrete case
$f(x)$	a continuous univariate function of x
$f(n)$	a discrete univariate function of n
f'	derivative of $f(x)$
$f(x, y)$	a continuous bivariate function of x and y
$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}$ or f_x, f_y	partial derivatives of a function $f(x,y)$ with respect to x or y
$f(m, n)$	a discrete bivariate function of m and n
$\hat{f}$	estimate or approximation of f
j	imaginary unit
μ_f, σ_f	expected value and standard deviation of a random signal f
$\rho_f(k)$	correlation coefficient of order k of a random signal f
$\aleph$	natural numbers
$\Re$	rational numbers

Symbols

α_j or $\alpha(m,n)$	fidelity of j-th data point value in (m,n) in regularization
b, $\hat{b}$	base value of an edge, and its estimate
c, $\hat{c}$	contrast value of an edge, and its estimate
$c_i(n)$	sequence of contrast values along curve i
C_i	edge curve i
C_u^p	edge curve u in frame p
$\tilde{C}_v^q$	matching edge curve v from frame q
$d(x;c, w, \sigma)$	filter response of 1-D Gaussian smoothed step edge model to a Gaussian derivative filter
$\mathbf{d}_{2D}(x, y;c, w, \sigma, \theta)$	gradient response of 2-D Gaussian smoothed step edge model to Gaussian partial derivative filters
$E_{i,j}$	the j-th edge point on edge curve i
$E(u)$	error functional in regularization
f_w	constant factor in edge intensity profile reconstruction
$\Phi(.,.)$	similarity functional in edge curve matching
$g(x;\sigma)$	1-D Gaussian function or filter
$g_{2D}(x, y;\sigma)$	2-D Gaussian function or filter
g, $g(x_j,y_j)$, $g(m,n)$ or $\mathbf{g}$	given data in the regularized interpolation problem
$i(x, y)$	image intensity function
k, K	reconstruction scale level, and its maximum
l, L	subsampling level, and its maximum
$l_u(n)$	sequence of (chain-coded) links describing the shape of curve u
$\Delta l_u(n)$	sequence of prediction error links of curve u
λ	regularization parameter
m, $\hat{m}$	center value of an edge, and its estimate
$m_i(n)$	sequence of center values along curve i
N_u	number of edge points on curve u
ω	overrelaxation parameter in SOR
ω_x, ω_y	frequency domain variables
$P(u, g)$	data compatibility functional in regularization
$\mathbf{P}$	data compatibility matrix in regularization

Q	number of levels in a quantizer for parameter signal coding
$Q(\mathbf{z}_u, \mathbf{z}_v)$	minimum mean-squared distance between $\mathbf{z}_u$ and $\mathbf{z}_v$
$s\,(x;b, c, w)$	1-D Gaussian smoothed step edge model
$s_{2D}\,(x, y;b, c, w, \theta)$	2-D Gaussian smoothed step edge model
$S(u)$, $S_k(u)$	ordinary and scale-k stabilizing functional in regularization
$\mathbf{S}$, $\mathbf{S}_k$	ordinary and scale-k stiffness matrix in regularization
σ	(filter) scale parameter
t	arc-length parameter of the position function of a curve
t_s	edge detection threshold
θ	edge orientation in 2-D
u, $u(x,y)$, $u(m,n)$ or $\mathbf{u}$	solution surfaces in regularized interpolation problem
$u^i(m,n)$	solution surface in regularized interpolation problem at iteration i
$U\,(x)$	unit step function
vx_u, vy_u	displacement vector for curve u
w, $\hat{w}$	width (or smoothness) value of an edge, and its estimate
$w_i(n)$	sequence of width values along curve i
x, y	spatial domain variables
x_0, x_1, x_2	edge locations
$\hat{x}_0$, $\hat{x}_1$, $\hat{x}_2$	estimates of edge locations
x_d	edge location error
$x(t)$, $y(t)$	coordinates of points along a curve
xs_u, ys_u	coordinates of the starting point of curve u
ξ	minimum edge curve length
$\mathbf{z}_u$ or $\mathbf{z}_u\,(t)$	position vector function of curve u
$\mathbf{z}_{u,n}\,(t)$	n-th straight line segment of $\mathbf{z}_u$

Definitions

The *Signal-to-Noise Ratio* (SNR), of an approximation $\hat{f}$ (e.g. a decoded version) to f, is defined as follows:

$$\mathrm{SNR} = 10.0 \times {}^{10}\log \frac{\mathrm{Var}\{f\}}{\mathrm{Var}\{f-\hat{f}\}} .$$

The *Peak Signal-to-Noise Ratio* (PSNR) is defined as follows, with $f_{\max} = 255$ in grey-level images:

$$\mathrm{PSNR} = 10.0 \times {}^{10}\log \frac{f_{\max}^2}{\mathrm{Var}\{f-\hat{f}\}} .$$

The *Normalized Mean-Square-Error* (NMSE) is defined as follows:

$$\mathrm{NMSE} = \frac{\mathrm{Var}\{f-\hat{f}\}}{\mathrm{Var}\{f\}} \times 100\% .$$

Abbreviations

DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation (predictive quantization)
EPC	Edge Primitive Coder
HBCG	Hierarchical Bases Conjugate Gradient
H.261	CCITT Recommendation for Video Codec
JPEG	Joint Photographic Experts Group
MPEG	Moving Pictures Experts Group
MS-SOR	Multi-Scale Successive Over-Relaxation
PCM	Pulse Code Modulation (direct quantization)
pdf	probability density function
psf	point spread function
PSTN	Public Service Telephone Network
SOR	Successive Over-Relaxation
UTQ	Uniform Threshold Quantizer
VLC	Variable Length Coding

Representatie en Codering van Beelden op Basis van Intensiteitsranden

Samenvatting

Vanwege de gestage toename in het gebruik van visuele informatie, worden technieken om beelden efficiënt op te slaan en te versturen steeds belangrijker. Een grote verscheidenheid aan beeld- en video-codeermethoden is in het verleden voorgesteld en enkele methoden zijn reeds gestandaardiseerd, hetgeen de toepassing van beeldcompressie in de praktijk op grote schaal mogelijk maakt. Aangespoord door de verwachte groei in de opslag, manipulatie en uitwisseling van multi-media informatie, is het onderzoek in de beeldcodering nieuwe wegen ingeslagen.

Dit proefschrift bespreekt een methode om digitale beelden en video efficiënt te representeren en coderen, door relevante variaties in het beeldintensiteitsoppervlak, ook wel *randen* genoemd, parametrisch te modelleren. De volgende onderwerpen worden met name besproken: analyse van intensiteitsranden, reconstructie van het intensiteitsoppervlak, matching van randen en het coderen van de parameters van het randmodel.

Hoofdstuk 1 bevat een inleiding in de beeldcodering en in de aanpak met behulp van randmodellen. Het belangrijkste vraagstuk in de beeldcodering is: Hoe sterk kunnen beelden gecomprimeerd worden, als de interpreteerbaarheid en de kwaliteit van de gedecodeerde beelden voldoende hoog moeten blijven? Men kan natuurlijke beelden vaak beschrijven als vloeiend verlopende intensiteitsoppervlakken, nu en dan onderbroken door scherpe veranderingen in de intensiteit. Deze scherpe overgangen in de intensiteit van het beeld corresponderen met belangrijke kenmerken van de scene die afgebeeld wordt, zoals het optreden van objectranden, schaduwranden of veranderingen in de reflectiviteit van oppervlakken. Veel codeermethoden maken geen expliciet gebruik van deze karakteristieke eigenschappen van beelden. In dit proefschrift zijn representatie en codering van beelden

gefundeerd op de analyse, modellering en manipulatie van scherpe overgangen in de intensiteit, de *intensiteitsranden*. Er wordt, in vergelijking met vroegere codeermethoden, een duidelijk nieuwe weg ingeslagen door de geometrie van de randen in beelden expliciet te beschrijven.

Intensiteitsranden zijn belangrijk bij de interpretatie van beelden of scènes. Deze hypothese wordt ondersteund door de enorme hoeveelheid onderzoek die binnen de computer vision aan intensiteitsranden is besteed en door studies van visuele perceptie in dieren en mensen. Vanuit een beeldrepresentatie die alleen informatie over intensiteitsranden bevat kunnen interpreteerbare beelden worden gereconstrueerd. Wellicht kunnen dus goede resultaten worden behaald op het gebied van de beeldcodering bij hoge compressie door een benadering te kiezen die is gebaseerd op het gebruik van intensiteitsranden.

In Hoofdstuk 2 wordt een methode besproken om intensiteitsranden te analyseren en wordt een beeldrepresentatie op basis van randen gedefinieerd. Een geïdealiseerd parametrisch model wordt gebruikt om de variatie in de intensiteit van een rand te beschrijven, waarbij het model op het intensiteitssignaal wordt gepast, op ieder punt waar een rand ligt. We beperken ons tot een model van een zgn. *Gaussisch geëffende intensiteitsrand*, dat beschreven wordt door *contrast*, *breedte* en *centrum-waarde* parameters. Er wordt aangetoond dat Gaussische-afgeleide filters gebruikt kunnen worden om deze randen te detecteren en tegelijkertijd de modelparameters te berekenen. De effecten van enkele veel voorkomende afwijkingen van het geïdealiseerde model worden bestudeerd aan de hand van fouten-analyses en experimentele resultaten. De *schaal* van de Gaussische analyse-filters beïnvloedt de nauwkeurigheid van de berekende modelparameters. In het algemeen is de breedte van de rand een goede indicatie voor de in te stellen schaal van de Gaussische filters, die gebruikt worden bij het analyseren van randen. Echter, bij een juiste acquisitie van de beelden kan de schaal van de filters beter laag worden gehouden.

We beschrijven vervolgens hoe de beeldrepresentatie geconstrueerd wordt, uitgaande van de geëxtraheerde informatie over de intensiteitsranden. De representatie wordt opgebouwd uit zgn. *rand-primitieven*, die ieder het verloop beschrijven van: (a) de plaats van randpunten langs een randcurve in het beeldvlak (*geometrische* data) en (b) het intensiteitsoppervlak binnen een smalle band langs een dergelijke curve, in termen van de contrast, breedte en centrum-waarde modelparameters (*fotometrische* data). Ten slotte wordt beschreven hoe de randpunten langs een dergelijke curve gereconstrueerd kunnen worden op basis van de rand-primitieven.

Hoewel het originele beeld niet exact terug verkregen kan worden vanuit de representatie, bestaande uit rand-primitieven, kan vaak wel een beeld worden berekend dat het origineel numeriek en visueel benadert. Dit wordt besproken in Hoofdstuk 3. De rand-primitieven in de representatie schrijven alleen de waarden van pixels langs randcurven voor. Het gehele intensiteitsoppervlak moet worden gereconstrueerd of *geïnterpoleerd* vanuit deze schaars verdeelde data. Met behulp van de *regularisatie-theorie* kan dit interpolatie probleem vertaald worden naar het minimaliseren van een *gladheids-functionaal*. De optimale oplossing kan worden gevonden door middel van standaard iteratieve algoritmen, zoals *relaxatie*-algoritmen.

De informatie die in de rand-primitieven zit opgeslagen wordt door deze algoritmen iteratief gepropageerd over het gehele beelddomein. Deze standaard algoritmen werken echter te traag.

Er wordt een nieuw optimalisatie algoritme voorgesteld, waarmee globalere en dus snellere propagatie van informatie over het beelddomein mogelijk is. Dit nieuwe algoritme kan worden afgeleid door *multi-schaal gladheids-functionalen* in de regularisatie te introduceren. Een analyse van de convergentie-eigenschappen en experimentele resultaten tonen aan dat het *multi-schaal* relaxatie-algoritme veel sneller convergeert dan het standaard algoritme.

In Hoofdstuk 4 wordt verdere compressie en reductie van de data in de rand-primitieven besproken en de toepassing hiervan binnen de codering van stilstaande beelden bij lage bit rates. De posities van randpixels (geometrische data) worden verliesvrij gecodeerd met behulp van een variant van chain-codering. Een sequentie van chain codes wordt gemodelleerd door een tweede-orde Markov ketting en gecomprimeerd met behulp van variabele lengte codering volgens de methode van Huffman. De waardes in een sequentie van modelparameters van één type (bijvoorbeeld het contrast) van de randpixels langs een randcurve (fotometrische data) worden opgevat als de bemonsteringen van een één-dimensionaal *parameter signaal*. Een eenvoudige, niet-verliesvrije, codeertechniek is ontworpen op basis van de statistiek van de parameter signalen. Deze techniek bestaat uit een stap waarin het signaal wordt gefilterd en onderbemonsterd, en een stap waarin DPCM toegepast wordt. Optimale *Uniform Threshold Kwantisatoren* worden gebruikt om de prediktiefout te kwantiseren. Huffman variabele lengte codering wordt toegepast om deze data verder te comprimeren.

Bij experimenten op enkele beelden blijkt de kwaliteit van de gedecodeerde beelden (in termen van Peak Signal-to-Noise Ratio) tussen de 23 en 33 dB te liggen bij een bit rate tussen de 0.05 en 0.41 bit per pixel (compressie ratio tussen 20 en 170). De belangrijkste visuele artefacten in de gedecodeerde beelden zijn: verlies van textuur, "halo-effecten" en gekartelde randen. De vormen en posities van ieder object worden goed weergegeven. Standaard JPEG presteert beter dan de codeermethode gebaseerd op randen bij bit rates boven de 0.20 bit per pixel, terwijl de laatste methode beter presteert dan JPEG bij bit rates lager dan 0.20 bit per pixel.

Spatiëel adaptief coderen kan worden gerealiseerd door gebruik te maken van *a priori kennis* tijdens de detectie van randen. *Kennis-gestuurde beeldcodering* van hoofd-en-schouder beelden kan een verbeterde kwaliteit van belangrijke delen van deze beelden opleveren.

In Hoofdstuk 5 wordt de beeldcompressie methode uitgebreid naar het geval van *beeldsequenties*. Om gebruik te kunnen maken van de temporele correlatie in beeldsequenties moet men eerst de verbanden tussen rand-primitieven uit verschillende beelden in de sequentie vaststellen. Daartoe is een frame-naar-frame matching algoritme ontworpen, gebaseerd op een *gelijkheids-maat*. Dit algoritme zoekt, voor elke rand-primitieve in het huidige frame, een primitieve in een voorgaande beeld, die maximale gelijkheid vertoond met de huidige primitieve. Geometrische gelijkheid wordt gemeten door de minimale gemiddelde kwadratische afstand tussen de twee curven te berekenen. Fotometrische gelijkheid wordt gemeten door de kruiscorrelatie tussen parameter signalen te berekenen. Het algoritme is in staat om ongeveer 75% van alle randpixels in een beeld te matchen.

Een sequentie van chain codes van een curve kan *interframe* worden gecodeerd, door de sequentie van chain codes van de bijbehorende curve (gevonden door het matching algoritme) als prediktie te gebruiken. Alleen de prediktiefouten in de chain codes hoeven dan nog (Huffman gecodeerd) verstuurd te worden. Tevens wordt een bewegingsvector verzonden die de lokatie van het beginpunt van de curve codeert. In de experimenten die zijn uitgevoerd kon echter *geen* codeerwinst worden behaald door gebruik te maken van interframe coderen van de posities van randen i.p.v. intraframe coderen, zelfs niet als de prediktiefouten in de chain codes niet-verliesvrij worden gecodeerd. Interframe coderen van de parameter signalen (fotometrische data) wordt niet besproken.

Voor een aantal hoofd-en-schouder beelden lag de uiteindelijke beeldkwaliteit (in termen van de PSNR) tussen de 27 en 33 dB, bij bit rates tussen de 0.09 en 0.26 bit per pixel (compressie ratio's per frame tussen de 30 en 90). De visuele kwaliteit van de gedecodeerde beeldsequenties bij de experimenten is nogal teleurstellend, vanwege ernstige flikkeringen van gebieden van lichte naar donkere helderheidsniveaus in het hele beeld. Het verschil in beeldkwaliteit tussen verliesvrij en niet-verliesvrij coderen van de interframe prediktiefouten in de chain codes is gemiddeld klein.

Hoofdstuk 6 bevat een afsluitende discussie over de gevolgde aanpak en de resultaten. Wat betreft de toepassing van deze methode in de praktijk moet men opmerken dat de drempel die gebruikt wordt in de rand-detectie bij onze experimenten met de hand is ingesteld, maar een grote invloed kan hebben op de kwaliteit van de gedecodeerde beelden en de behaalde compressie ratio. Wat betreft de benodigde rekentijd moet men opmerken dat het iteratieve karakter van het algoritme om intensiteitsoppervlakken te reconstrueren het grootste probleem oplevert. De voorgestelde codeermethode kan worden vergeleken met enkele andere nieuwe methoden binnen de lage bit rate codering, zoals segmentatie-gebaseerd of gebieds-gebaseerd coderen. Men kan dan concluderen dat, voor al deze methoden, de complexiteit van het coderen van de posities van randen het voornaamste struikelblok is dat significante vooruitgang ten opzichte van blok-gebaseerde methoden in de weg staat. Hoewel praktische toepassing in de beeldcodering van de methode die in dit proefschrift beschreven is nog ver weg lijkt, lijkt er nog genoeg ruimte te zijn om de gebruikte beeldrepresentatie en de codeeralgoritmen verder te verbeteren.

Acknowledgments

I would like to take the opportunity to thank all the people who in some way contributed to the completion of this thesis.

Firstly, I owe my family all thanks for their support during all my years of study and for giving me the freedom to pursue my aspirations. Very special thanks go to Jeanette Mee Lan for her tremendous caring, support and encouragement during the past four years.

I am highly indebted to my office mates Herjan Barnard and Marcel Reinders. I have greatly enjoyed our pleasant and productive cooperation. Without their readiness to help me or to answer any and all of my questions, this thesis could not have been completed. Further, I have appreciated our many refreshing discussions on topics outside our scientific work.

Thanks go to all my fellow research assistants at the Information Theory Group and in the Videophone Project. Exchanging ideas and sharing experiences with them has been helpful.

To Dick Boekee and Jan van der Lubbe I am grateful for giving me the opportunity to do the research. I would like to acknowledge the contribution of Bülent Sankur from the Bogaziçi University to this thesis. He was an enlivening supervisor during his stay in Delft, and he has shown a continuing interest in my work afterwards. I thank my promotor Eric Backer, Jan Gerbrands and Jan van der Lubbe for their supervision and for bringing the research to a good end. I thank Piet Verbeek for his enthusiasm about my work.

Ms. J. B. Zaat-Jones corrected the English text.

Curriculum Vitae

Petrus Johannes Leonardus (Peter) van Beek was born in Amsterdam, the Netherlands, on July 18, 1967. From 1979 to 1985, he received his secondary education (VWO) at the Fons Vitae Lyceum in Amsterdam. He obtained an M.Sc. (ir.) degree in Electrical Engineering (Computer Science variant) from the Delft University of Technology, the Netherlands, in December 1990.

From December 1990 to April 1995, he worked as a research assistant (Assistent in Opleiding) towards a Ph.D. degree at the Information Theory Group of the Electrical Engineering Department of the Delft University of Technology. The research described in this thesis was carried out in this period, within the context of the interdisciplinary Videophone Project. Within this project, eight researchers from four different departments (Electrical Engineering, Applied Physics, Industrial Design Engineering and Technical Social Sciences) studied technical as well as non-technical issues concerning visual telecommunication services, in particular the videophone.