Region-based Video Compression

Ratna Rambaruth

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Centre for Vision, Speech and Signal Processing,
School of Electronic Engineering, Information Technology and Mathematics
University of Surrey
Guildford, Surrey GU2 5XH

September 1999

© Ratna Rambaruth 1999
Abstract

First generation image coding standards are now well-established and coders based on these standards are commercially available. However, for emerging applications, good quality at even lower bitrates is required. Ways of exploiting higher level visual information are currently being explored by the research community in order to achieve high compression. Unfortunately very high level approaches are bound to be restrictive as they are highly dependent on the accuracy of lower-level vision operations. Region-based coding only relies on mid-level image processing and thus is viewed as a promising strategy. In this work, substantial advances to the field of region-based video compression are made by considering the complete scheme. Thus, improvements to the failure regions coding and the motion compensation components have been devised. The failure region coding component was improved by predicting the texture inside the failure region from the neighbourhood of the region. A significant gain over widely used techniques such as the SA-DCT was obtained. The accuracy of the motion compensation component was increased by keeping an accurate internal representation for each region both at the encoder and the decoder side. The proposed region-based coding system is also evaluated against other systems, including the MPEG4 codec which has been recently approved by the MPEG community.
Acknowledgements

I would like first like to convey my most sincere thanks to both my supervisors Joseph Kittler and Bill Christmas who have been very supportive and encouraging while allowing the necessary freedom for exploring ideas. I am also grateful to several members of the CVSSP who have helped me with technical aspects throughout the duration of my project.

I would also like to thank the sponsors for this project, namely the CVCP, the European Commission and the CVSSP group.

Finally I would like thank my parents for having always motivated in all my endeavours.
## Contents

1 Introduction ........................................... 3
   1.1 Existing Coding Techniques ......................... 4
      1.1.1 Transform-based techniques ....................... 5
      1.1.2 Model-Based Techniques ............................. 5
      1.1.3 Coding Standards ..................................... 6
   1.2 Region-based video compression .................... 11
   1.3 Achievements ............................................. 11
   1.4 Thesis Overview ........................................... 12
   1.5 Conclusions ............................................... 14

2 Region-Based Video Coding .......................... 15
   2.1 Existing Region-Based Strategies .................... 17
      2.1.1 Segmentation of the Displaced Frame Differences .... 17
      2.1.2 Volume-Based Coding .................................. 18
      2.1.3 Partitioning of each Frame ............................ 18
      2.1.4 Segmentation of the Decoded Intra Image ............ 19
      2.1.5 MPEG4 video coding algorithm ........................ 19
   2.2 Coding of Intra Frames ................................. 20
   2.3 Spatial Segmentation ..................................... 21
      2.3.1 Thresholding ............................................ 21
      2.3.2 Edge Detection .......................................... 22
      2.3.3 Region Growing and Merging .......................... 22
      2.3.4 Morphological Segmentation .......................... 22
## CONTENTS

2.3.5 Texture-Based Segmentation ................................................... 23
2.3.6 Visually Significant Regions .................................................... 23

2.4 Contour Coding ................................................................. 24
2.4.1 Contour Simplification ....................................................... 24
2.4.2 Chain Codes ..................................................................... 25
2.4.3 Polygonal Approximation .................................................. 26
2.4.4 Centroid-Based ............................................................... 27
2.4.5 Morphological Skeleton .................................................... 28

2.5 Texture Coding ................................................................. 28
2.5.1 Reference Texture Coding ................................................ 28
2.5.2 Shape Adaptive DCT ......................................................... 29
2.5.3 Block Padding Techniques ................................................ 29
2.5.4 Polynomial Approximation ................................................. 29
2.5.5 Orthogonal Basis Functions .............................................. 30
2.5.6 Gaussian-Markov Random Field Models ............................ 30
2.5.7 Morphological interpolation .............................................. 30

2.6 Region-Based Motion Estimation ............................................. 31
2.6.1 Techniques ...................................................................... 31
2.6.2 Choice of the Motion Model ............................................. 31
2.6.3 Global Motion Estimation ............................................... 32

2.7 Motion Field Segmentation .................................................. 32

2.8 Prediction Error Coding ....................................................... 32

2.9 Conclusions .......................................................................... 33

3 System Description ................................................................ 35
3.1 Initialisation of Regions .......................................................... 36
3.2 Motion Estimation ................................................................. 38
3.3 Region-Based Motion Prediction ........................................... 39
3.4 Motion compensation of the mask of an arbitrarily-shaped region 40
3.5 Motion segmentation ............................................................ 41
3.6 Building up the Knowledge of the Scene and Objects ............... 42
3.7 Failure Regions .................................................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7.1 Detection of Failure Regions</td>
<td>45</td>
</tr>
<tr>
<td>3.7.2 Encoding of Failure Regions</td>
<td>45</td>
</tr>
<tr>
<td>3.7.3 Updating of Region Description</td>
<td>46</td>
</tr>
<tr>
<td>3.8 Encoding of Prediction Error</td>
<td>46</td>
</tr>
<tr>
<td>3.9 Bitstream Structure</td>
<td>47</td>
</tr>
<tr>
<td>3.10 Conclusions</td>
<td>47</td>
</tr>
<tr>
<td>4 Efficient Interframe Coding for a Region-based Scheme</td>
<td>49</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>49</td>
</tr>
<tr>
<td>4.2 Texture coding of failure regions</td>
<td>50</td>
</tr>
<tr>
<td>4.3 Results</td>
<td>53</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>57</td>
</tr>
<tr>
<td>5 Region Representation for a Region-based Coding Scheme</td>
<td>59</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>59</td>
</tr>
<tr>
<td>5.2 Smoothing due to Successive Interpolations</td>
<td>61</td>
</tr>
<tr>
<td>5.2.1 Explanation of smoothing by theoretical analysis</td>
<td>61</td>
</tr>
<tr>
<td>5.2.2 Illustration of smoothing effect</td>
<td>65</td>
</tr>
<tr>
<td>5.3 Motion Compensation of Regions from a more Accurate Internal</td>
<td>67</td>
</tr>
<tr>
<td>Representation</td>
<td>67</td>
</tr>
<tr>
<td>5.4 Results</td>
<td>68</td>
</tr>
<tr>
<td>5.4.1 Translational motion</td>
<td>68</td>
</tr>
<tr>
<td>5.4.2 Affine motion</td>
<td>73</td>
</tr>
<tr>
<td>5.4.3 Floating point representation within a complete region-based coding system</td>
<td>74</td>
</tr>
<tr>
<td>5.4.4 Effect of the floating point representation on a region based codec</td>
<td>75</td>
</tr>
<tr>
<td>5.4.5 Comparison with a block based codec</td>
<td>76</td>
</tr>
<tr>
<td>5.5 Conclusions</td>
<td>76</td>
</tr>
<tr>
<td>6 Tuning and Evaluation of the Region-based Coding System</td>
<td>85</td>
</tr>
<tr>
<td>6.1 Evaluation of the RBC system</td>
<td>85</td>
</tr>
<tr>
<td>6.1.1 Manual segmentation</td>
<td>86</td>
</tr>
</tbody>
</table>
6.1.2 Spatial segmentation on the decoded intra image ...... 87
6.1.3 Motion segmentation ........................................... 88
6.1.4 Motion-based Merging ........................................... 90
6.1.5 Performance of the complete system ....................... 91
6.2 Comparison with region-based coding schemes .......... 91
6.3 Comparison of the optimised region-based system with a block-based system ............................................. 92
6.4 Comparison of the region-based system with MPEG4 .... 93
   6.4.1 Externally generated alphamap ............................. 94
   6.4.2 Objects from automatic segmentation ..................... 95
6.5 Conclusions and future work ................................. 99

7 Conclusions ................................................................. 101
   7.1 Summary ............................................................. 101
   7.2 Future Work ......................................................... 102
List of Figures

1.1 Example sequence of compression modes ................. 9

2.1 Motion representation for large region compared to that of enclosed blocks ........................................ 16
2.2 Motion continuity is preserved for objects but not for blocks .......... 16
2.3 (a) 8 directional vectors; (b) an example of 8-connected chain codes to describe a region; (c) an example of 4-connected chain codes to describe the same region .................. 25
2.4 Contour approximation with arbitrary control points .......... 26
2.5 Centroid based representation including example of missing segment ...................................................... 27

3.1 Region-based codec ........................................... 37
3.2 Region adjacency graph ...................................... 38
3.3 Failure regions which cannot be predicted by motion compensation 43
3.4 Structure of layers in bitstream ................................ 47

4.1 Illustration of failure regions for frame rate of 7.5Hz (left) and 1.875Hz (right) ........................................ 51
4.2 Picture area used for prediction for the Mean Fill method .......... 53
4.3 Decoded frame 48 using SA-DCT (top) and Dynamic Mean Fill (bottom) ........................................ 55
4.4 Detail for decoded texture ..................................... 56

5.1 Filtering effect on a block of samples due to successive applications of a linear filter .......................... 64

vii
5.2 Smoothing effect of successive bilinear interpolation operations . 66
5.3 Manipulation of internal data structure for regions .............. 67
5.4 Improvement in bitrate due to float representation (Container sequence) ............................................................... 69
5.5 Improvement in bitrate due to float representation (Calendar125 sequence) ............................................................... 70
5.6 Successive reconstructions derived from the internal floating point representation ........................................................ 78
5.7 Frame 14 of Calendar125 with different prediction schemes; left - motion compensated (MC) prediction, middle - MC prediction error, right - final reconstruction. ............................................................... 79
5.8 Improvement in bitrate and quality due to float representation (Lennart sequence; Quantization = 16) ............................................................... 80
5.9 Bitrate comparison for H.263 test sequences ................. 81
5.10 PSNR comparison for H.263 test sequences ................. 81
5.11 Improvement in bitrate due to floating point representation (Calendar125 sequence) ............................................................... 82
5.12 Improvement in bitrate due to floating point representation (Container sequence) ............................................................... 82
5.13 Comparison of motion compensated error with and without floating point representation for frame number 8 of the Calendar125 sequence ............................................................... 83
5.14 Comparison of motion compensated error using the region-based coding method with floating point representation for frame number 36 of the Calendar125 sequence ............................................................... 84
6.1 Manual segmentation of the first frame of the Container sequence 86
6.2 Quality of spatial segmentation with varying intra Quantization step for the Silent sequence ............................................................... 89
6.3 Refinement of segmentation using motion information .................. 90
6.4 Comparison between H.263 codec and region-based codec for H.263 test sequences ............................................................... 93
LIST OF FIGURES

6.5 Simple decomposition for a video compression system for natural video input ................................................. 94
6.6 Experimental setup for comparison of video object symbol coding ................................................................. 95
List of Tables

1.1 Table showing the data rate for CIF sequences and the data rate to be achieved for real-time transmission through PSTN and playback from storage media ...................................................... 4

4.1 Number of bits required to code uncovered regions in Calendar125 55
4.2 Number of bits required to code uncovered regions in Container 56

6.1 Bit spending in the block-based H.263 coding scheme ............. 87
6.2 Bit spending in a region-based coding scheme with an ideal segmentation ................................................................................................... 88
6.3 Variation in bitrate for Calendar125 sequence using different merge factors ................................................................. 90
6.4 Optimum coding parameters for H.263 sequences ................. 92
6.5 Coding efficiency comparison between MPEG4 and RBC ........ 95
6.6 Symbol coding efficiency comparison between MPEG4 and RBC for Container sequence ................................................................. 97
6.7 Symbol coding efficiency comparison between MPEG4 and RBC for Calendar125 sequence ................................................................. 97
6.8 Decomposition of bits produced by MPEG4 coding for the Container sequence ................................................................. 98
6.9 Decomposition of bits produced by RBC for Container sequence 98
6.10 Comparison between optimised RBC and MPEG4 ............... 99
Chapter 1

Introduction

The importance of communication is recognised by all and has created increasing demand for communication channels, which are unfortunately a limited resource. Thus tremendous work for bandwidth reduction of transmitted data was motivated. This particularly applies to audiovisual signals which are nowadays considered vital for effective communication. The need for compression is specifically acute because of the broadband nature of source video signal. This thesis focuses on video compression because of its growing importance.

Among the transmission applications for which extensive compression is essential are video teleconferencing and video telephony. Other major applications for very low bitrate video coding include electronic newspaper, remote sensing, video-surveillance, tele-medicine and communication for deaf people. The widespread availability of personal workstations enable multimedia applications such as multimedia email, interactive multimedia database, multimedia videotex, interactive computer imagery, video game and multimedia annotation.

The need for compression can be easily justified. If we adopt the CIF (common intermediate format) standard, a digital picture format for low bitrate video of resolution 352x288, it means that there are 101376 pixels in each frame. Each raw colour pixel represents 12 bits of data. Each video frame, uncompressed, is about 150 KB in size. 30 frames per second translates to about 37 million bits of data per second. In one hour, one would accumulate over 16 GB of data (more than hard drive). It also exceeds the transfer bandwidth
### Table 1.1: Table showing the data rate for CIF sequences and the data rate to be achieved for real-time transmission through PSTN and playback from storage media

<table>
<thead>
<tr>
<th>Data rate</th>
<th>Transmission bandwidth</th>
<th>Storage bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>37Mb/s</td>
<td>16Kb/s</td>
<td>300Kb/s</td>
</tr>
</tbody>
</table>

of most computer system peripherals. Double spin CD-ROM drives transfer about 300 kB/s. ISA bus computers can transfer about 2.5 MB through the bus. Thus the video file would be impossible to access in real time. Transmission of videophone scenes through the available public switched telephone network (PSTN) would also be impossible. As it is known, the available network PSTN is mainly used for the transmission of speech. When trying to transmit such a colour video signal via the PSTN under the assumption that the channel capacity is 16 Kbps, and 8 Kbps are used for video and voice, respectively, the compression ratio must be as high as 4666 : 1 ([54]). Please see table 1 for a summary.

It has been seen above that not only are there several applications requiring the transmission of digital video data but also that the amount of data generated even for a simple application is very large. It will be seen in the following section how this problem can be tackled by several possible video compression methods. Afterwards the type of video compression technique which is the focus of this thesis, i.e. region-based video compression, will be briefly described. In section 1.3, the contributions to the region-based video compression research are listed, after which an overview of the thesis is provided.

## 1.1 Existing Coding Techniques

The possible solutions for achieving video compression are varied. There are certain basic requirements which should be satisfied by any low-bit-rate coder [58]: it must operate in real time with limited delay; it must operate at the
capacity of the communication channel, which may vary with time; and the quality of reconstructed images or sequences must be acceptable to humans. Transform-based techniques and model-based techniques are briefly considered in the following two subsections. The final subsection reviews current video coding standards.

1.1.1 Transform-based techniques

The coding scheme which has received the most attention within the research community is the hybrid MC-DCT technique. For very low bitrate applications, this scheme has proved unsatisfactory due to the appearance of severe blocking artifacts resulting from the use of blocks in the motion estimation. Moreover, waveform based techniques are inflexible in terms of content-based interactivity and other added functionality.

Purely waveform-based techniques, in which the time axis is taken to be the third dimension, have been investigated, e.g. 3D DCT and 3D subband. However due to the fact that temporal redundancy is not adequately exploited, results are not satisfactory.

1.1.2 Model-Based Techniques

This term refers to a coding approach which employs a semantic model to represent the contents of an image sequence. It consists of two steps: analysis and synthesis. The goal of the analysis is to find for each component of the scene a model which will be appropriate for the description of the transformations which it undergoes during a video clip. Each frame is then synthesised based on the estimated motion description and the knowledge of the previous frame and model parameters.

Model-based schemes can be divided into two types: semantic-based coding which uses explicit object models, e.g. a human head, to analyse and synthesise images, and region-oriented coding in which the objects are unknown. For a restricted scene, e.g. a typical head-and-shoulder scene, the first scheme can give a higher coding efficiency. Because an object model is not necessary for
the second scheme, it can be applied to a more general class of scenes. Regions typically in current schemes are modelled as arbitrarily shaped planar patches which undergo translational, quasi-affine, affine or perspective transformations. This technique is the subject of this thesis and will be discussed in greater detail in the next section.

1.1.3 Coding Standards

Standardization of compressed video formats facilitates the manipulation, storage and transmission over computer networks and broadcast channels as a clear syntax for the representation of the encoded bitstream is defined. To date, standards are based on first generation coding techniques. In this section the ITU and the ISO standards are briefly overviewed. A few proprietary standards which have been popular are QuickTime (Apple), AVI (Microsoft) and RealVideo (RealNetworks) but are not discussed here.

**International Telecommunication Union (ITU) - H.261**

Intended applications are videoconferencing and videophone services over the integrated services data network (ISDN) at p x 64 kbps, p = 1,...,30. A maximum delay of 150 ms is specified so as to make bidirectional video communication seem natural. The input image format - Common Intermediate Format (CIF) - and the video multiplex data structures are fixed within the standard so that various decoders can interpret the bitstream without any ambiguity.

The video compression algorithm can operate in the intra mode, which is based on a block-by-block DCT coding (i.e. its reconstruction does not depend on any previous frames), or in the inter mode for which the algorithm can be summarised as follows:

(a). Estimate a motion (displacement) vector for each macroblock (16 pixels by 16 lines). Block matching on 16 x 16 luminance blocks is generally used although motion compensation (MC) is not compulsory in the standard.

(b). Select a compression mode for each macroblock (MB) from “Inter”, “In-
1.1: Existing Coding Techniques

ter+MC" and "Inter+MC+PIL" which denote interframe with zero motion vector, motion compensated interframe and motion-compensated with loop-filtering respectively. If loop filtering is chosen, the prediction error block is filtered before being transformed.

(c). Process each macroblock to generate a header followed by a data bitstream that is consistent with the compression mode chosen.

The resulting prediction error, called the displaced frame difference (DFD), is then coded using the 2D waveform-based DCT.

Other significant features include the quantization scheme and the mechanism for rate control. For more details on this standard please refer to Tekalp [85].

ITU - H.263

H.263 was devised for low bit rate video applications. The baseline configuration is similar to that of H.261 described above in which spatial and temporal redundancies in the video input are reduced by the DCT spatial transform and motion compensation respectively. The enhanced feature for increased efficiency is half-pixel instead of full-pixel motion compensation. The transmitted symbols are variable length coded. On top of the improved baseline configuration, options are available to improve the coding efficiency:

- Unrestricted Motion Vector mode - In this mode larger motion vectors than those allowed in H.261(+/− 15 pixels) can be used. Motion vectors are also allowed to point outside the picture.

- Syntax-based Arithmetic Coding mode - Arithmetic coding instead of Huffman coding is used to encode symbols.

- Advanced Prediction mode - Four 8 × 8 vectors instead of one 16 × 16 vector can be used per macroblock. This means that the encoder has to decide which type of vectors to use and code the decision in the header of the macroblock.
• PB-frame mode - A PB-frame consists of two pictures being coded as one unit. Each of the above options can result in varying improvements in bitrate depending on the type of video input.

ITU - H.26L

The Video Coding study group within the ITU have started work intended to promote the exploration of video coding algorithms that address areas which their previous video coding standards did not adequately address [94]. Such areas include enhanced visual quality at bitrates below 28.8 kbps, enhanced error robustness (in particular addressing the needs of mobile networks), lower complexity, lower end-to-end delay, enhanced temporal and/or spatial resolution, extra functionality. Furthermore application areas other than the traditional real-time audio/visual conversational services upon which the previous algorithmic work was anchored are intended to be supported. These include applications with the following attributes:

• Capture/Encode/Store/Forward/Decode/Display (CESFDD) - such as video on demand.

• Capture/Encode/Store/Forward/Store/Decode/Display (CESFSDD) - such as video mail.

• Capture/Encode/Forward/Store/Decode/Display (CEFSDD)

International Standardisation Organisation (ISO) - MPEG1

MPEG-1 standard was developed for storage of video and audio as an enabling technology for interactive multimedia systems. The target bitrates are about 1.5 Mbps. The MPEG-1 algorithm is similar to H.261 except for some additional features and compression modes. Random access, fast-forward/reverse and low decoding delay (1 s) are needed for interactivity in multimedia services. The provision for two types of interframe compression modes allows for greater flexibility in algorithms for encoding. An example of modes for a sequence of
frames are shown in Figure 1.1. These are forward prediction (P-pictures) and bidirectional prediction (B-pictures). P-pictures are always encoded relative to the preceding I- and P-pictures whereas B-pictures can be forward, backward or bidirectional relative to other I- or P-pictures. Several benefits can be derived from this configuration [85].

ISO - MPEG2

The MPEG-2 algorithm was upgraded with respect to MPEG-1 so as to cope with interlaced and higher-definition inputs. A scalable bitstream was also made possible. Better quantization and coding options were introduced. The primary application targeted during MPEG2 definition process was the all-digital transmission of broadcast TV quality video at coded bitrates between 4 and 9 Mbps. However, the MPEG2 syntax was found to be efficient for other applications such as those at higher bitrates and sample rates (e.g. High definition television (HDTV)).

ISO - MPEG4

In this section, the concept and goals of the MPEG4 standard are briefly introduced. More details about the algorithm will be provided in the next chapter as they will be relevant to part of the experimentation described in this thesis.

MPEG4 aims at applying higher-level computer vision techniques which are likely to yield high compression while catering for content-related functionalities. The latter aim provides standardised ways to represent units of aural or
visual content (called *media objects*), to describe the composition of a scene and for users to interact with the audiovisual scene at the decoder end. Media objects in coded form can be represented independent of their surroundings or background. Examples of media objects are still images, video objects (e.g., talking head) and audio objects (e.g., the speech of that person). The scene composition language (BIFS) is very similar to that used in the Computer Graphics world, i.e., the Virtual Reality Modelling Language (VRML). The types of functionality available at the user end are:

- placing media objects anywhere in a given coordinate system;
- applying transforms to change the geometrical or acoustical appearance of a media object;
- grouping primitive media objects in order to form compound media objects;
- applying streamed data to media objects in order to modify their attributes (e.g., animation parameters driving a synthetic face);
- changing the user's viewing and listening points anywhere in the scene.

As seen above the standard specifies the bitstream syntax for describing media objects and compositing them into a scene. The production of the media objects is left to the developers of the application at the encoder end. This approach is well suited to applications in which the source material is synthetically generated which means that the objects are created individually anyway. However, traditional rectangular frame-based video sequences have to be analysed so as to extract meaningful objects. Research in this latter area is still ongoing and will dictate the success of MPEG4 to some extent at least for applications requiring traditional video input.

Version 1 of the MPEG4 standard was released last year (1998). Version 2 will be ready this year and will include body animation and coding of 3D meshes as well as the functionality provided by version 1.
1.2 Region-based video compression

Region-based video coding is attractive both for achieving high compression as well as for increased content-based functionality. Compared to first-generation coding techniques, it is hoped that the more natural partitioning of the image will allow a better representation of the motion of individual regions, thereby reducing the prediction error - especially at boundaries between objects moving in different directions. This can help in reducing blocking effects which become visible in sequences coded at a very low bitrate by a block-based technique. With respect to semantic-based coding, region-based coding for the purpose of compressing general scenes is less complex in that the exact modelling of each component in the scene is not crucial. For example, complex objects such as deformable objects and 3D objects which move freely in a 3D coordinate system can be approximated with a combination of simple 2D patches.

The major components of a region-based coding scheme are: the intraframe coding, the partitioning of the image, the shape description, the inner texture representation, the region-based motion estimation and compensation and the encoding of the prediction error. Improvements can be devised either at the system level or at the component level. A considerable amount of work has been done within the MPEG-4 community in each of these areas. However even for research at the component level, the evaluation has to be done at the system level as the behaviour of the components are heavily interdependent. Thus during this work, a considerable amount of focus has been on the design of the complete codec which dictates the right perspective for lower level tool design. At the same time, the possible limitations of each tool have to be considered during the codec design.

1.3 Achievements

The first major achievement of this work is that a complete region-based codec has been designed and built. This was necessary before any of the components could be studied such that the effect on the system as a whole could be mea-
sured. Although many tools from available software could be reused, others
(e.g. segmentation, region-based motion compensation) had to be developed
from scratch. The design allows alternative tools to be easily plugged in and
the developed tools to be reused within other codecs.

Secondly, three novel ways of coding failure regions were devised. The im­
provement was possible due to close observation of properties of failure regions
which occur during the region-based coding of natural video sequences. The
methods provide increasingly accurate approximations of the failure region and
use information from the neighbourhood of the failure region which is already
available at the decoder side. Comparison with an established technique (SA-
DCT) is provided.

Thirdly, a dramatic improvement in coding efficiency was achieved through
a better internal representation for regions. This is because inaccuracies gather
due to repeated motion compensation which includes a filtering step whenever
non-integer motion is present. The technique thus works for certain types of
regions while causing no degradation for others.

Extensive evaluation of the devised codec was also performed, first by obser­
vation of the various video signal outputs of the system and then by comparing
with other coding schemes such as H.263 and MPEG4. This was useful for the
proper tuning of the coding parameters and also for pinpointing aspects of the
system which need further exploration. The inadequacy of the MPEG4 coding
technique to allow video objects to interact with one another was found to be
costly in coding terms.

1.4 Thesis Overview

Chapter 2 focuses on region-based video coding and starts by discussing the
rationale for investigating such a scheme. The different types of systems which
have been said to be region-based previously in the literature, including the
newly-developed MPEG4 standard, and the various components which can be
useful in implementing a complete system are thereafter described.

The coding strategy which is being investigated in detail is presented in
Chapter 3. A flow diagram will be used to point out the features of the system and the tools which form part of the baseline codec. Alternative components in the tool set will also be described. The proposed improvements to the codec will be compared to this baseline codec in subsequent chapters of the thesis.

Advances made for coding of uncovered regions are described in chapter 4, which starts by demonstrating the properties of uncovered regions which occur during region-based analysis of natural image sequences. Hence it will be seen that the established texture coding techniques used for coding arbitrarily shaped regions are not the most appropriate for the purpose of coding failure regions. The new methods which have been devised are then described and supporting results are provided.

The importance of an accurate representation for regions which are repeatedly motion compensated is shown in chapter 5. A new internal representation for inner texture as well as the contour of planar regions is therefore devised which is not dependent on a grid-like representation of images. This is kept on both sides of the codec. The savings of bits achieved for specific regions are then presented.

The effect of the improved representation described in chapter 5 on the overall performance for test sequences is shown in chapter 5.4.3. A comparison with a block-based codec is made.

The whole region-based coding system is tuned and evaluated in chapter 6. The strengths as well as the shortcomings of the developed codec are highlighted. The individual components are considered first after which the whole system is compared with other region-based codecs including MPEG4 and with H.263.

Finally the conclusions of the work are summarised and the achievements underlined in chapter 7. Ideas for future work derived from the observations presented in the previous chapters are provided.
1.5 Conclusions

In this chapter, the importance of achieving efficient digital video compression was explored in the light of the variety of applications which can become very popular within the consumer community at large. The existing solutions, including standards, were reviewed and the justification for the choice of region-based coding system as the one with the best mileage for improvement was argued. Finally, a list of scientific contributions in the field of video compression was given.
Experimental results show that a block-based decomposition of images requires many bits for the coding of the prediction error in order to produce good quality reconstructions. This is because the representation does not fit the content of moving images properly. A more natural partitioning of images would be one in which each region would demonstrate a coherent motion. Obviously this cannot be achieved from the first frame only as it contains no motion information. Let us assume for the time being that a meaningful segmentation can be recovered from a few frames. The following savings - compared to a block-based technique - in number of bits for motion vectors may be foreseen:

- more efficient motion compensation of large regions

  Traditional block matching algorithms segment the image into 16x16 blocks. Motion vectors for each block can be differentially encoded. However this only partially exploits the correlation within the vector field, i.e. in only one dimension. If an object spans several blocks vertically, the same information will be repeated in the bitstream (figure 2.1). This can be avoided when a region-based strategy is used.

- better motion-compensated prediction around motion borders

  If a region (block) contains multiple moving objects, block-based motion compensation gives poor results as a global smoothness constraint is imposed on the block. Consequently, more bits have to be used to encode the
Figure 2.1: Motion representation for large region compared to that of enclosed blocks

Figure 2.2: Motion continuity is preserved for objects but not for blocks

motion compensated error. In a region-based strategy, motion boundaries generally coincide with the region boundaries and a better prediction can be achieved.

• better interpolation and prediction of motion vectors.

It is common to perform prediction or interpolation of motion vectors for successive frames, i.e. motion vectors are differentially encoded (for prediction). This is very useful under the assumption that motion within the block is roughly constant over time, which indeed often holds. Referring to Figure 2.2, one can see that it is much more appropriate, and possibly beneficial, to perform motion prediction and interpolation in a region-based coding scheme as the arbitrary segments correspond to some semantic region in the image [42].

• more accurate description of regions of interest

If it is possible to define a portion of the image which is very significant
for a viewer (for e.g. a face in a videotelephone sequence) by some automatic means (see Section 2.3.6), bitrate reduction can also be achieved by allocating a high priority to that region. It can then be updated more frequently than the rest of the image. Such regions can be defined much more precisely in a scheme with arbitrary shape compared to one with blocky shapes.

Similarly, accurate descriptions of object boundaries are desirable if adaptive motion models are to be used.

In addition to coding efficiency, a region-based scheme provides other benefits such as object-based scalability, content-based editing and direct editing on the bitstream.

It has been shown in this introduction that certain benefits may be reaped when using a semantic partitioning of an image. In the following section several region-based coding schemes are presented and the rest of the chapter describes the required tools in a region-based codec.

2.1 Existing Region-Based Strategies

The term region-based coding has been used to refer to various schemes. These are described below. However not all of them utilise the object description to the full. Another criterion to be borne in mind when considering a system design is the efficiency with which it reduces the amount of shape information to be transmitted.

2.1.1 Segmentation of the Displaced Frame Differences

Regions of high energy in the motion compensated error image are isolated by some segmentation method and separately coded. The justification for the use of this technique is that the prediction error image tends to be composed of isolated regions of high energy, especially along motion boundaries. Ebrahimi et al. [25] use morphological segmentation, contour coding for the transmission of the region shape information and quantization of the interior pixels. The
latter feature, it is claimed, relies on the fact that the spatial sensitivity of
the human visual system is generally lower in the moving parts of the images.
Christopoulos et al. [20] however prefer using a simplified contour and texture
approximation by a linear combination of weakly separable bases.

As this scheme segments regions corresponding to failure areas rather than
to objects in the scene, it is not attempting to exploit the same features of
moving images depicted in the introduction. Furthermore, the shape of the
selected regions have to be sent from scratch at least for the first interframe.
Subsequent alterations of the boundaries may not be economically feasible.

2.1.2 Volume-Based Coding

The time-varying image sequence is treated as a single 3D data volume [78, 47],
the voxels of which are grouped into several 3D regions. To accomplish the 3D
segmentation, Willemin et al. [88] use a 3D split-and-merge algorithm. The
criterion for the splitting and merging processes is the accuracy with which an
approximated 3D polynomial can represent the volume. The grey-level varia­
tion can then be coded using the coefficients of the approximating polynomial
as features. The enclosing surface of each region is coded using a pyramidal
structure in the 3D space.

In addition to the fact that temporal redundancy is spatially encoded, sev­
eral frames need to be accumulated before the start of the processing. This is
undesirable for a communication system which relies on a low coding delay.

2.1.3 Partitioning of each Frame

The central tool to this scheme [75] is the segmentation which solves the region
correspondence problem and allows a region to be tracked in successive frames.
The segmentation step is thus similar to that in the volume based codecs above.
The difference in this scheme is that temporal redundancy is reduced by motion-
based prediction rather than by spatial methods. The motion estimation stage
is greatly simplified as it only needs to determine which motion can be used to
reconstruct the current region from the previous one.
2.1: Existing Region-Based Strategies

This scheme can potentially perform well in all the cases mentioned in the introduction of this chapter, provided that the segmentation step is reliable enough.

2.1.4 Segmentation of the Decoded Intra Image

The intra image is coded using a standard image coding technique [92, 93]. It is then reconstructed in exactly the same way both on the decoder side and on the encoder side. Spatial segmentation is carried out on both sides so as to maintain a matching region description both at the encoder and decoder. In this way, expensive contour and texture information need not be sent. Motion estimation and prediction are carried out for every region as for every block in a standard method (Section 1.1.3). This technique will be further elaborated on in Chapter 3 as it is the basis of the proposed strategy.

2.1.5 MPEG4 video coding algorithm

The MPEG4 coding algorithm supports content-based functionality as it allows the separate coding of data from each semantic component of the scene [16, 48]. However, from the point of view of compression the region-based representation is not fully exploited as the block-based processing is still used.

The information related to the shape, motion and texture for each video object (VO) is coded into a separate layer in order to separate decoding of the VO’s. Each layer (VOL) is coded in the same way in the basic algorithm. Actually certain layers can be coded as a face, body or mesh-based object but this is not considered here. Similar to previous MPEG baseline coders, the MPEG4 algorithm processes successive VOL’s in a block-based fashion. After coding the VO shape information, the input VO is partitioned into non-overlapping macroblocks. Each macroblock contains blocks of data from both luminance and co-sited chrominance bands - four luminance blocks and two chrominance blocks, each with size $8 \times 8$. The previously coded frame of the VO ($N - 1$) is kept in a frame store in both encoder and decoder. Motion compensation is performed on a block or macroblock basis - only one motion
vector is estimated between VOP frame \( N \) and VOP frame \( N - 1 \) for a particular block or macroblock to be encoded. The motion compensated error is calculated by subtracting each pel in the motion-shifted block or macroblock in VOP frame \( N - 1 \) from its counterpart in frame \( N \). An \( 8 \times 8 \) DCT is then applied to each of the blocks followed by quantization of the DCT coefficients with subsequent run-length coding and entropy coding (VLC).

### 2.2 Coding of Intra Frames

Visual distortion is inevitable when images are coded at high compression rates. The aim is to keep the image as acceptable to humans as possible while achieving high compression. Techniques for still image coding can be categorised into transform coding and second generation coding [21].

Each transform method is composed of three steps: transformation of the original image, quantization of the transformed coefficients and entropy coding of the quantized coefficients. The purpose of transform coding is to decorrelate the picture content into a few compacted coefficients and to encode these coefficients rather than the original pixels of the image. Examples of transforms are the discrete cosine transform (DCT) the wavelet transform and the Harr transform. However, at high compression (30:1-40:1), these methods cause artifacts such as blocking and ringing, to which the visual system is very sensitive.

Second generation techniques attempt to decompose images into some visual primitives. One method is to extract spatially homogeneous regions from the image and represent them with their contour and inner texture. This coincides with the physiological model of vision, which is known to be sensitive to edges present in an image. The three main steps involved in this technique are segmentation, contour coding and texture coding.

Fractal coding is based on the iterated functions systems theory which is closely related to fractal geometry. The main idea consists in expressing each segment of an image by a transformation from another part of higher resolution. Particularly, fractal-based coding can achieve high compression for natural images containing structures such as grass, trees, etc. However a generic
fractal-based coder is yet to be developed.

2.3 Spatial Segmentation

Segmentation is the process that subdivides an image into its constituent parts based on some criterion. In a coding framework, ideally, each part should correspond to an object in the scene as seen from the introduction of this chapter. It is, however, often the case that there is not enough information within the image to achieve this. Instead, a segmentation with redundant regions (more than one region belonging to the same object) or vice versa (a region spanning more than one object) may be obtained. The coding scheme has to be able to exploit a sub-optimal initial segmentation as much as possible, for example by incorporating motion information when it becomes available (Section 3.6).

2.3.1 Thresholding

Pixels are classified according to their luminance value [40]. The thresholds can be determined using a variety of statistical techniques, for example:

(a). The histogram of the image is examined for locating peaks and valleys. If it is multimodal, then the valleys can be used for selecting thresholds.

(b). Select the threshold \( t \) so that a predetermined fraction of the total number of samples is below \( t \).

(c). Adaptively threshold by examining local neighbourhood histograms.

(d). Selectively threshold by examining histograms only of those points that satisfy a chosen criterion. For example, in low-contrast images, the histogram of those pixels whose Laplacian magnitude is above a prescribed value will exhibit clearer bimodal features than that of the original image.

(e). If a probabilistic model of the different segmentation classes is known, determine the threshold to minimise the probability of error.
However thresholding tends to be useful only when the amplitude features sufficiently characterise the object (for example in the segmentation of printed documents).

### 2.3.2 Edge Detection

Edge detection is a popular image processing tool which is well documented [71]. In an edge-based approach, the boundaries of objects are used to partition an image. Points that lie on the boundary of an object must be marked. Such points, called 'edge points' can often be detected by analysing the neighbourhood of the point. By definition, the regions on either side of an edge point (i.e. the object and the background) have dissimilar characteristics. Thus, in edge detection, the emphasis is on detecting dissimilarities in the neighbourhood of points. Most edge detectors use only intensity characteristics; however, more sophisticated characteristics which can be derived from intensity values such as texture and motion may also be used. An edge linking step then has to be performed to obtain disconnected regions. This set of techniques is not very appropriate to image coding for the following reasons: selection of optimum threshold is difficult, broken contours are common and they are susceptible to noise. However they are still used.

### 2.3.3 Region Growing and Merging

Regions are formed by merging pixels which satisfy a certain homogeneity criterion. For example, this can be the similarity to the grey level, location in the colour space, uniformity of texture of a measure of how well a polynomial surface could fit the region. A very large number of regions tend to be produced. Postprocessing involves merging regions with similar criterion levels and eliminating small regions.

### 2.3.4 Morphological Segmentation

The algorithm consists of four steps: simplification, marker extraction, decision-making and modelling [78, 74, 77]. The step which is the most distinct from
other segmentation techniques is the pre-processing in which morphological area filters are used, i.e. small regions are removed without affecting the contours of the preserved regions. The function of the marker extraction is to identify several homogeneous regions according to a flatness criterion. This can be done by a region growing technique. In the final step, uncertain areas (marked zero) are assigned to existing clusters. Please see [24] for further details.

### 2.3.5 Texture-Based Segmentation

Humans observe a textured region as being homogeneous, although the intensity across the region may be non-uniform. If a segmentation is merely based on intensity measurements, it will produce results which do not match with the perception of the scene. Several features which can describe texture and therefore can be used as homogeneity criterion for any of the above segmentation techniques have been used in computer vision [36]. Features which have been successfully applied to image segmentation for coding include mean deviation about each pixel [26] and measurement in the fractal dimension [41, 39].

### 2.3.6 Visually Significant Regions

The purpose of this tool is to assign to each region obtained from any of the above segmentation methods a score which is proportional to its relative importance. Delmot [22] uses a multicriterion classification with the global score of a region being a weighted sum of the scores obtained for the different criteria. The weights have to be adapted depending on the scene to be analysed. The image border criterion favours the central areas of the image; the hit-score criterion aims at eliminating regions without pixels in a user-defined area; the face texture criterion rejects all regions whose chrominance components do not match with a set of face texture samples; the motion criterion gives importance to moving regions; and, finally, the continuity criterion prevents drastic change in region scores from frame to frame. Though these are simple measurements, it is claimed, interesting regions can be determined and coded with higher priority.
2.4 Contour Coding

The cost for contour coding is usually high in any segmentation based technique (about 1.4 bits per pixel, even with sophisticated methods). On the other hand, contours contain the most important visual information for humans. Thus transmission of contours should be bypassed as far as possible. This is a system issue which can be addressed efficiently as described in Section 2.1.4. However contour coding cannot be completely avoided, e.g. in the cases shown in Section 3.6, when a new contour is identified by the motion analysis. The techniques below then have to be used. Other techniques which have been investigated by the MPEG-4 community are the baseline-based method [52, 51], the Modified Modified Reed (MMR) [91] (which is the standard method for the G4 facsimile) and the context-based arithmetic coder (CAE) [66]. The latter reference also contains a comparison of the above 3 techniques. Contour coding techniques which are optimal in the rate distortion sense have also been investigated [84, 57, 83, 44]. The issue of interframe contour coding has also been addressed [17].

Most contour coding techniques employ a simplification step (section 2.4.1) prior to coding so as to remove local details which are irrelevant to humans but costly to encode. In fact these details might only be due to noise. Thereafter different bitrate reduction techniques may be applied depending on the chosen representation.

2.4.1 Contour Simplification

Gu [33] uses majority filtering for the purpose of contour simplification and compares his approach to Fourier descriptors [68], curve filtering by estimation of abscissa and geometrical curves and morphological filtering. He argues that traditional morphological filters such as opening and closing are not relevant for the contour simplification task because the results are dependent on the label assignment in the partition image instead of relying on contour information. A cross structuring element of size one was used. This can remove spark labels very effectively. An average bitrate reduction of 23 percent for a simplified contour with respect to a contour obtained directly from morphological
Figure 2.3: (a) 8 directional vectors; (b) an example of 8-connected chain codes to describe a region; (c) an example of 4-connected chain codes to describe the same region.

2.4.2 Chain Codes

The simplest method of boundary representation is the four - or eight-connected chain code as in Figure 2.3. From a given starting point (S), a digitised contour can be traced by a sequence of moves in one of four or eight initial directions and thereafter three or seven subsequent onward directions, using 2 or 3 bits per link respectively. After the start, only relative changes in direction need to be coded (differential coding). Only three symbols are then needed for the four-connected code.

As boundaries in natural images are not purely random, runs of similar directions are to be expected and may be efficiently coded using variable word length coding [19]. Pattern recognition techniques can be used to identify frequently occurring chains. Further reductions can be obtained through vector quantization. In [56], the first- and second-order Markov models were used to describe the source structure and the Huffman or arithmetic coding were used as variable length coding.
2.4.3 Polygonal Approximation

For polygonal representation, the components are: the control points for the polygon which intersect with the shape to be coded and the description of the contour between each pair of control points. A variety of different algorithms for obtaining the representation exist. For example,

- the initial control points may or may not be fixed,
- splines or lines may be used for the approximation between control points,
- an error signal between the approximated shape and the real one may or may not be included.

An example of an approximation procedure for a contour segment is described by Gu [33] and Pikaz [70]. In the former, it begins from two end points in the contour segment (see Figure 2.4 for an illustration of the procedure using straight lines). An interpolated curve given by the adopted model (line or spline) passing through these two points is established. The maximum geometrical distance between this curve and the original contour segment is calculated. If this distance is greater than a given threshold, a new control point is added. This step is repeated until the desired threshold is reached.

A refinement on this is to detect the dominant points of the contour in a prior step [55] followed by approximation with line segments. Another variation is to find the equilateral polygon approximation which allows a one-dimensional...
Figure 2.5: Centroid based representation including example of missing segment description of the contour [73]. Xu [90] uses several inscribed polygons to describe the shape.

Spline or straight line approximations, can rarely produce an accurate description of a contour but are still useful for coding purposes as small details are removed.

2.4.4 Centroid-Based

Region shapes can also be described from their centroid and the distances between the centroid and the points where radial lines intersect with the boundary (figure 2.5). The centroid is the average of all boundary pixels of the region. Special cases in which the centroid lies outside the region may arise. A rule for describing a new centroid is needed. Another special case is when the radial line intersects with more than one boundary pixel. If either intersection point is taken, the technique will fail in describing a portion of the boundary, which then has to be represented otherwise [18].

The sampling rate is controlled by the angle between the radial lines and can be varied depending on size and object complexity. To attain more compression the distance values are input to the DCT and then the coefficients of the DCT are variable length coded. The coordinates of the centroid are separately coded.
2.4.5 Morphological Skeleton

Another shape-oriented representation is the "morphological skeleton" [9]. It is built by taking the centres of all maximally inscribed discs. A disk is maximally inscribed if no larger disk could be inscribed in the region. The skeleton representation consists of two parts: the centre positions of the maximally inscribed discs and the corresponding sizes. This method cannot compete with the above in terms of coding efficiency [33].

2.5 Texture Coding

The majority of texture coding techniques presented have been explored in the context of still image coding. In segmentation-based video coding, texture coding is needed for failure regions which cannot be predicted using motion compensation, in which case the following techniques are still directly applicable. For the purpose of coding of motion-compensated error, the various methods have to be assessed differently as these regions have different properties compared to those of a natural image.

The simplest way of representing the texture within a region is by its mean value [35]. Although the result is often acceptable, it looks unnatural - similar to a cartoon picture. The following techniques have been developed to create a more natural picture.

2.5.1 Reference Texture Coding

In this method, transformation from already sent texture is performed whenever possible. The coding efficiency depends on the number of bits required to describe which block the present one is being referred from. Only translation is used in [76]. The block is referable if the minimum mean squared error with respect to all possible blocks is below a certain threshold. This method is not very successful for smooth regions.
2.5: Texture Coding

2.5.2 Shape Adaptive DCT

This rather simplistic technique has received attention in the MPEG-4 community as it seems to provide good results. All pixels of a block that should be coded are shifted to the top and vertical 1D DCTs of different lengths are applied. The resulting coefficients are shifted to the left and horizontal DCT's are applied. A zigzag scan, omitting those positions that do not belong to the object spectrum, converts the 2D arrangement into a 1D one. SA-DCT is superior to block padding techniques according to Park et al. [67].

2.5.3 Block Padding Techniques

This is a group of techniques in which segments of the block outside the current region are replaced with a value which minimises the DCT coefficients. Zero-stuffing, mean-stuffing and mirror image extensions have been attempted. It is however claimed [8, 46, 2] that high frequencies are introduced by the latter. Instead, padding with zeros in the frequency domain is proposed in order to achieve smooth interpolation. Chen et al. [15] try to determine the optimal extrapolation of the region to a circumscribing square such that computing the 2-D DCT of the square produces a minimum number of high-amplitude DCT coefficients. Moon et al. have devised the boundary block merging technique [59], in which the active pixels of two different boundary blocks are merged if none of the pixels overlap.

2.5.4 Polynomial Approximation

This is a coding technique which proved to achieve high compression ratio (100:1) while maintaining a high fidelity in the semantic content of the image [49, 6]. If a linear combination of smooth low-order two-dimensional polynomial functions are used, the method does not cope well with areas of the image containing high spatial frequencies, like edges, texture and small details. In addition, computational loads are somewhat large. For this reason, Baseri and Modestino [5] use the 2D spline on the lowest frequency subband of the image only.
Leou and Chen [53] use a polyline representation which simply consists in scanning all the pixels within the region inwards in a spiral fashion. It is claimed that the correlation of these pixels is high and therefore long lines can be used to faithfully represent the signal.

2.5.5 Orthogonal Basis Functions

The texture inside the arbitrarily shaped segment is approximated by a weighted sum of basis functions, e.g. polynomials [31]. A set of base functions which are orthogonal on the segment are needed. These can be obtained using orthogonalisation schemes. The encoder then proceeds as a traditional transform coder. The decoder applies the same orthogonalisation procedure to construct the same orthogonal basis that was used at the encoder, thus avoiding the transmission of basis vectors. Weakly separable bases have been proposed [69, 12, 79, 82] for speed of computation of the orthogonal bases.

2.5.6 Gaussian-Markov Random Field Models

GMRF model parameters can be used as texture features and therefore a reconstruction method for textured regions [50]. The latter used polynomial expansions for the remaining uniform regions. It was reported that the synthesised image looked like a puzzle image with visually detectable degradations mainly near the boundaries. A nonlinear smoothing filter that has low-pass characteristics depending on the gradient magnitude perpendicular to the boundary was used as a postprocessing stage.

2.5.7 Morphological interpolation

The interpolative coding techniques are based on the coding and transmission of a raw sketch from which the remaining pixel values may be reconstructed (Casas et al. [11]) by geodesic dilation. At a rather modest compression ratio (30:1), the quality of the reconstruction of a natural image does not look satisfactory.
2.6 Region-Based Motion Estimation

The function of the motion estimation module depends on where it fits in the coding scheme. Given the chosen scheme (Chapter 3), techniques which satisfy the following formulation only can be considered: the motion estimation should assign to each region in the segmentation map a transformation which will allow the reconstruction of a certain portion of the current frame (see also Section 3.3).

2.6.1 Techniques

Traditional motion estimation techniques are optical flow equation based methods, block motion model based methods, Hough-transform based methods, pel-recursive methods, Bayesian methods and feature based methods [85]. Most of these can be generalised to take arbitrarily-shaped regions as input.

In the Hough transform based motion, the search for the motion vector is carried out in the parameter space. Each pixel supports motion vectors which compensate it satisfactorily. In order to eliminate the contribution of outliers a robust kernel can be applied to the support value. Pixels which are outside the region simply do not take part in the voting process.

One technique of motion estimation which is only applicable to a region-based scheme was proposed by Carlsson and Reillo [10]. It involves the estimation of the motion of curved contour segments and the interpolation of the displacement field between contours. However, they assume that the contours move rigidly in the image plane, which immediately restricts the motion model. Other work on contour tracking can be found in [34, 29].

2.6.2 Choice of the Motion Model

The more complex the motion model, the more precise the motion description of the segment which means that prediction errors would be reduced. However the coding cost is increased as more bits are needed for the extra parameters. Furthermore, the computational load increases with the complexity of the model.
Nicolas et al. [65] include a motion model adaptation step in the motion estimation module, which means that for each region the optimum motion model can be automatically determined.

### 2.6.3 Global Motion Estimation

The explicit splitting of the background from the foreground is well suited to a region-based coder. In Moscheni [62], a first estimate of camera motion is obtained either by a clustering technique performed on the local motion field or by the tracking of the previous mask. The global motion estimation is then carried out on the background region.

### 2.7 Motion Field Segmentation

Classical segmentation used on grey-level images may be applied to motion segmentation. Thresholding of the motion field in order to separate the background from the foreground was tried by Fan et al. [28]. Chae et al. [13] used a region-growing segmentation where the similarity measure between motion vectors of two regions was used as the homogeneity criterion. Murray and Buxton [63] used a statistical method, i.e. a Bayesian approach in which the Markov random field models the prior expectation of the optical flow field. A quadtree segmentation with regression on the prediction error frame was attempted by Baker and Maeder [4]. But much more attractive are the techniques which allow simultaneous motion estimation and segmentation [86, 14].

### 2.8 Prediction Error Coding

In unchanged areas of an image, the prediction error is completely flat. For motion compensated parts, there is usually a little disturbance due to the quantization of the motion parameters. However the energy in failure areas is large. Thus prediction error coding is done in two steps. For example in H.263, if the energy within a macroblock exceeds a certain threshold, it is encoded from scratch and the resulting error image is coded in a similar way to an intra im-
age. This is good for large failure areas resulting from motion failure, i.e. when the motion is too complex for the motion estimation module to determine.

However, because other failure areas tend to be long and thin (along motion boundaries) rather than in big blocks used in the H.263 standard, segmentation of these areas was proposed (section 2.1.1). Thus only the actual failure area has to be coded from scratch. The gain should compensate for the added overhead of sending the contour of the failure region.

At this stage it is appropriate to consider how efficient prediction error coding is relative to the coding of motion vectors in order to determine the optimum motion model to be used. The more complex the motion model the fewer the motion failure regions expected but also the higher the number of bits for coding each motion vector. Some work has been done on finding the right balance between motion model complexity and prediction error coding [61].

### 2.9 Conclusions

The techniques which are relevant to region-based video compression have been reviewed. From the literature survey, it was evident that most of the components, i.e. the spatial segmentation, motion estimation and motion segmentation, cannot be expected to behave ideally. Indeed for the spatial segmentation component, the ideal output cannot even be specified.
Chapter 3

System Description

It was seen from the previous chapter that not only are there different ways to implement each component of a region-based coding system but also different ways to configure the system as well. Ideally, to achieve improvements in codec design, several tools should be available. The system structure should also be easily reconfigurable so as to allow extensive experimentation within a complete system and with a large set of input data whenever a new component has to be evaluated. We achieved this to some degree in spite of the relatively small scale of the work. In this chapter, the system structure and all the default components which were chosen for this study are described in detail. There was a strong focus on producing a completely automatic baseline system against which potential improvements to the system and to its constituent tools could be tested. The improvements which were devised and tested in this way include better tools for failure region encoding and motion compensation which are described in detail in chapters 4, 5 and 5.4.3. Justification for the design decisions made is provided to some extent in this chapter. For a few components the choices are further supported experimentally (chapter 6).

The baseline codec is depicted in figure 3.1 using a data flow diagram in which the boxes represent operations and the arrows represent data flow. The encoder part of the system transforms the video input into compact symbols while the decoder reconstructs the video data from the symbols it receives via the communication channel. The codec can operate either in the Intra or
the Inter mode as indicated by the switches both in the encoder and decoder diagrams. In the Inter mode, the encoder includes a decoding operation as the reconstruction of the previous frame is used to code the current one. Thus the segmentation operation on the decoded Intra frame is performed in exactly the same way both at the encoder and decoder ends and all the data available at the decoder is also available at the encoder (excluding cases of channel error).

The latter feature of the region-based coder allows the exploitation of the object representation in terms of the motion while minimising the amount of shape information to be sent. This part of the scheme for compression is similar to the one described in Section 2.1.4. The original segmentation is subsequently updated using the motion information. The majority of interframe information is contained in the motion vectors and possibly in the prediction error. Cases in which region information needs to be transmitted are described in detail in Sections 3.6 and 3.7. The chapter is concluded with a summary of the information contributing to the bitstream.

3.1 Initialisation of Regions

The initial partitioning of the frame may be obtained by any of the spatial segmentation methods described in section 2.3, apart from the edge-based approaches which would be sensitive to any blockiness in the decoded intra image. For the context of video compression, desirable properties of the segmentation algorithm are good performance for a general class of sequences and possibility to vary the number of regions produced. A simple region growing algorithm followed by morphological processing was implemented.

To perform region growing, a particular homogeneity criterion must first be defined. Starting from a pixel (seed), its neighbouring pixels are examined to select those sharing the same property. The procedure stops when all the connected pixels sharing the property are labelled, producing a single region. Another region is grown next to the previous one beginning with the first pixel that does not belong to the first region. The property used was colour homogeneity (threshold $T_1$). The postprocessing step involves removing small
3.1: Initialisation of Regions

Figure 3.1: Region-based codec
regions (threshold T2) by merging them with one of their neighbours without affecting the contour of the larger regions. This was achieved using a region adjacency graph which is a data structure used to represent the connectivity of regions (Figure 3.2). The nodes of the graph correspond to regions from the segmentation map. The edges correspond to contour segments which separate adjacent regions. The contour is stored as a list of edgels, where an edgel is a term describing the boundary between two pixels. The small regions are eliminated by deleting the node from the graph as well as all the edges to the node. The segmentation map built from the resulting region adjacency graph then does not contain any small regions. The number of regions can easily be varied using thresholds T1 and T2.

Experimentation on the ability of this technique to cope with a decoded intra image as input will be presented in chapter 6.

3.2 Motion Estimation

We now describe the motion estimation technique which we use in order to obtain accurate motion vectors. In our setup, we assume that the segments on which motion analysis needs to be performed have been previously extracted through spatial segmentation or spatio-temporal segmentation, such that they correspond to regions whose motion can be modelled by an affine (or translational) transformation. The affine transformation is given by
3.3: Region-Based Motion Prediction

We use a motion estimation algorithm based on the Hough Transform [7]. Individual pixels vote for members in the parameter space which can predict their position in the next frame well. A gradient-based search strategy is used to find the parameter with the highest score. The algorithm is made robust to real-life occurrences like transparency effects, occlusions, camera noise, specular reflections and shadows by the use of a robust kernel. Other features of the algorithm include a multiresolution search strategy which reduces computational complexity and facilitates the detection of large motion and compensation for a mean level shift (i.e. a uniform change of illumination).

The algorithm is adapted for arbitrarily shaped regions by using a mask at the stage of the calculation of the support function. The whole region is estimated in one step, i.e. it is not subdivided into blocks, because we are only considering regions which exhibit a coherent motion for our current experiment.

Once we obtain motion vectors at the encoder side, we have to spend bits on transmitting them at high precision. This implies some overhead compared to systems in which the motion vectors are only sent at half-pel accuracy (e.g. H.263) or less. However this overhead is minimal with respect to the possible gains for certain regions, for example large, textured regions, as will be demonstrated by the results.

3.3 Region-Based Motion Prediction

Given a region $R$ in the previous frame ($n-1$) and the current frame ($n$), the motion parameters for $R$ may be determined using the Robust Hough Motion Estimation algorithm (Section 2.6).

To reconstruct the image, the motion parameters are used to determine the position and appearance of each region in the following frame. Uncovered regions, regions of overlap, motion failure regions and new objects are so far

\[
\tilde{x}'_i = \left(\begin{array}{c} a_1 & a_2 \\ a_3 & a_4 \end{array}\right) \tilde{x}_i + \left(\begin{array}{c} a_5 \\ a_6 \end{array}\right)
\]
left blank in the reconstruction. These are dealt with as described in section 3.7.

Additional operations at this stage include the updating of the segmentation map which will be used in the coding of the following frame. The new segmentation map is constructed using the same contour information as for the prediction error image and the labels are filled in. So far the regions of overlap are left blank.

### 3.4 Motion compensation of the mask of an arbitrarily-shaped region

The motion prediction of the mask of an arbitrary shape which undergoes complex motion is not as straightforward as the prediction of a contour whose motion is restricted to integer translation. In the latter case only one step is needed:

- transform all the pixel positions inside the mask to obtain the new pixel positions using equation 5.2.

For a complex motion like zooming out, there can be holes in the new mask for the region because there are more pixels in the object in the current mask.

Therefore the following procedure is used to ensure the correct prediction of masks and thus object shapes even after several frames:

- keep a data structure for a contour which is not related to a grid structure with each region description; this means that a floating point representation of the contour is needed.

- transform the position of each element of this data structure using the motion vector as in equation 5.2 and store the new contour as a reference for the next frame.

- convert the floating point contour to a contour on the image grid by rounding each element.
• if necessary, use linking routine to form a complete contour on the image grid; the linking routine required should be very simple because the gaps in the contour are not expected to be longer than 1 or 2 pixels.

• retrieve all the pixels enclosed by the contour using a simple region growing routine.

Hence, provided that the motion of the region has been well modelled, its exact shape can be retrieved and no additional bits are required in the Inter mode for the sake of contours.

3.5 Motion segmentation

As seen earlier, motion segmentation is required in several situations, e.g. where the spatial segmentation fails to identify any object contours, where new objects appear in the scene or where the motion model is inadequate. Although motion segmentation does not produce contours as precise as spatial segmentation, in the above mentioned cases motion segmentation is the only option.

Several methods for motion segmentation were reviewed in chapter 2. The thresholding method was chosen for its ease of implementation and also, with the appropriate postprocessing, the resulting regions have the same properties as those obtained from the spatial segmentation.

The method proceeds as follows. During the motion estimation process, the use of the robust kernel ensures that the dominant motion vector is returned. Thus in the prediction error image for the region after compensation with the dominant motion vector, the areas of high error denote the outliers. If the proportion of outliers exceeds a threshold (T3), the motion of the region is estimated again but this time only at the outlier locations. These steps are recursively repeated until the number of outliers is below the threshold. At this point the segmentation map contains many small regions, as outliers do not tend to form large connected regions. The same postprocessing techniques as for the spatial segmentation are applied - except that a different value for the threshold is used (T4) - and the regions in the final segmentation are estimated again.
with the kernel switched off. The results of this process will be demonstrated in chapter 6.

Yokoyama et al [92] do not perform any motion segmentation. Instead, if the prediction error for a region is high, that region is again spatially segmented.

### 3.6 Building up the Knowledge of the Scene and Objects

Based on the results from motion analysis, some knowledge can be gathered about the scene content. Two distinct cases can be isolated:

(a). if adjacent regions experience sufficiently similar motion, they are likely to belong to the same real world object, in which case they can be merged. The criterion used for merging two regions with motion parameter $A$ and $B$ is given next. Let the difference between $A$ and $B$ be described as

$$D = \begin{pmatrix} D_v & 0 \\ D_t & 0 \end{pmatrix}$$

where $D_t$ are the translational and $D_v$ the other parameters of the affine transform respectively.

$$merge\_criterion = \begin{cases} 1 & \text{if } \forall d \in D_t, d < 0.1 \times merge\_factor \\ \& \forall d \in D_v, d < 0.01 \times merge\_factor \\ 0 & \text{otherwise} \end{cases}$$

where $merge\_factor$ is a parameter input to the coding system. This criterion is simple compared to other criteria for motion homogeneity which have been described in the literature but is still expected to improve the region description. For example, Morier et al [60] used a combination of criteria related to the quality of the prediction error image. Gu [32] based the criterion prediction error image modified by a morphological operation.

(b). if within one region in the current segmentation map there is evidence of
3.7 Failure Regions

Motion compensated error images consist of areas of high energy (resulting from complete failure of motion estimation) coupled with small errors (due to the inaccuracy of the motion prediction). Thus it could be more efficient to code the prediction error first region by region and then as an entire frame. Yokoyama et al [92] do not exploit the properties of the motion compensated

---

**Figure 3.3:** Failure regions which cannot be predicted by motion compensation.

(a) Uncovered background  (b) New object
(c) Uncovered region  (d) Overlapping regions

For describing a split in the bitstream (Figure 3.4(d)), the following components are required: the region to be split, the labels of the new regions on the lower level and the contours separating the new regions. For a merge (Figure 3.4(e)), the label of the higher level node and the labels of all the regions to be merged are required. Thus changes in the region description are encoded compactly.

3.7 Failure Regions

Motion compensated error images consist of areas of high energy (resulting from complete failure of motion estimation) coupled with small errors (due to the inaccuracy of the motion prediction). Thus it could be more efficient to code the prediction error first region by region and then as an entire frame. Yokoyama et al [92] do not exploit the properties of the motion compensated
error image to the full. They merely divide it into blocks and code the ones of high energy at higher priority.

Failure regions cannot be reconstructed using motion parameters. They can arise through different situations. For example, the motion module could fail to provide a description of the motion of a certain region if the motion is too complex. This is a case of a motion failure region. Figure 3.3 illustrates other failure situations. The shaded area in Figure 3.3(a) represents an uncovered portion of the background as the region moves. In Figure 3.3(b), a new object is entering the scene and obviously, it cannot be predicted from regions existing in previous frames. Although this case is a fairly common occurrence, it has been ignored by Yokoyama et al [92] as they restricted their test set to simple videophone sequences. The case of an uncovered object (3.3(c)) cannot be distinguished from the uncovered background case in the present scheme and therefore is encoded in exactly the same way. Yokoyama et al. [92] perform a spatial segmentation of every uncovered portion and send the contour and texture information of new uncovered regions. If regions overlap as in figure 3.3(d), it is impossible to determine initially whether to fill the shaded region with the texture of region 2 or of region 3 as it is equally likely that either is in the foreground. Thus the region of overlap is a failure region.

An attempt has been made to describe most of the common failure cases; however, the list is not exhaustive; several other combinations can be thought of. The examples given are mainly intended to provide an insight into the issue of prediction error coding.

The remainder of this section describes how failure regions are detected and encoded. Initially, only case(b) is distinguished from the rest for the purpose of encoding as it is the only one in which contour information needs to be sent. It is detected by a different mechanism, i.e. through the motion segmentation module, which means that it can be easily classified.
3.7: Failure Regions

3.7.1 Detection of Failure Regions

Detecting portions of uncovered background, regions of overlap and motion failure regions is straightforward. After constructing the prediction by motion compensation, holes will be left in these regions. The error image, which will also be available at the decoder, can be scanned for connected components which correspond to failure regions, thus defining their contour already.

The motion estimation module is expected to provide the information about any new moving object. When supplied with a region to estimate, it can detect whether there is multiple motion within that region and output a segmentation of the region.

3.7.2 Encoding of Failure Regions

As seen in the previous section, contour information for uncovered background, regions of overlap and motion failure regions need not be sent. The remaining texture information is sent in the same order as the regions are encountered. In order to keep the segmentation information complete over the image, these regions have to be assigned a label. For uncovered segments, this will be the label of the background while for regions of overlap, the label of the foremost region. As no distinction is made between these two type of regions at the encoding stage, the label needs to be explicitly sent. If a failure region is the result of complex motion, a new label has to be assigned as its corresponding region in the previous frame is not known. The method for assigning labels to holes in the prediction error image is yet to be explored.

For new regions detected by motion segmentation, the newly assigned label as well as the contour and texture information are transmitted. These are potentially very expensive as contour data will need to be sent for every frame as the object enters the scene.

One could at a later stage envisage different actions for each of the above situations if they can be reliably classified. For example, if it is known that a failure region is a case of uncovered background, its texture could be interpolated from known background. For the overlapping region case, the transmission
of relative depth of objects could resolve the prediction problem.

### 3.7.3 Updating of Region Description

The predicted segmentation map so far contains all the regions which have been successfully motion compensated. Pixels in uncovered regions have not been assigned any label yet although the texture for these regions is already available on the prediction image. This means that the segmentation of the uncovered regions can be carried out in both encoder and decoder just like the initial segmentation of the intra frame. In fact the same segmentation algorithm can be used except for one additional step in the beginning of the process. Because the regions which are already on the segmentation map have to be retained, the region growing procedure starts by a seed in each region and gathers the statistics of the region without altering its shape at first. The region-growing then proceeds as usual on the pixels surrounding the existing region. The remaining failure pixels if any are thereafter segmented using the same region growing technique. Afterwards a morphological filtering step is applied just like in the segmentation process on the intra frame so that the predicted segmentation retains the properties of the segmentation on the intra image. As for the intra segmentation, no contour information needs to be sent.

### 3.8 Encoding of Prediction Error

As mentioned in Section 3.7, the prediction error can be sent for failure regions or for the whole frame. After all the operations of motion compensation and failure region description, there can still be significant errors compared to the true image because of quantization of motion parameters and other inaccuracies. The resulting prediction error can be encoded by any of the standard image sequence coding methods.
(a) Sequence layer

| Intraframe | Interframe1 | ... | InterframeN | Contour Upgrade |

(b) Interframe layer

| region1 | ... | regionN | Prediction error | Failure regions | Splits | Merges |

c) Region layer

| label | motion parameters | motion model |

d) Failure Region layer

| label | Contour | Texture |

e) Split layer

| split label | new_label1 | Contour | new_label2 |

(f) Merge layer

| merge label | old_label1 | old_label2 |

Figure 3.4: Structure of layers in bitstream

3.9 Bitstream Structure

In this section, the various units of information contributing to the bit count are highlighted. The basic model consisting of layers is shown in Figure 3.4. Not all layers are shown. Each block in a layer corresponds to a data packet which may itself consist of another layer. The topmost layer is the sequence layer which contains all the information about the sequence and the lowest layers only contain information which can be directly converted to bits.

3.10 Conclusions

The baseline coding system described is very simple and does not include supplementary features such as rate control, temporal interpolation and scalability. As will be seen in subsequent chapters, this is sufficient for experimenting with
the algorithmic aspects which aim at exploiting the potentially better motion
description available in a region-based codec. The amount of contour informa-
tion to be coded by the proposed codec is limited by performing the spatial
segmentation on the decoded intra image on both sides of the codec. Temporal
information is thereafter utilised to update the original object description
throughout the sequence. The differences with a similar region-based coding
system [92] have been highlighted at various stages in this chapter.
Chapter 4

Efficient Interframe Coding for a Region-based Scheme

4.1 Introduction

In this chapter, we focus on the optimisation of the coding of uncovered regions. We find that the widely used tools for coding the texture within arbitrarily shaped regions of the intra image are not appropriate for failure regions. We propose a new method for doing so which consists in applying a predictive coding step to the newly visible portions based on already transmitted spatial data [72].

In region-based coding the behaviour of individual segments is analysed at the transmitter and the reconstruction is carried out at the receiver using the coded information. However little work has been presented on the interaction amongst objects as they undergo motion, i.e. overlapping and uncovered regions. Although this information might only make up a small percentage of the final bitrate it can become quite important to optimise its compression in very low bitrate environments.

For this purpose, texture coding techniques traditionally used in the context of region-based coding of still images have been explored (see section 2.5). The simple method of representing a region solely by its mean value was found unacceptable because of the resulting cartoon-like appearance [35]. Improvements
could be achieved by using low-order polynomials to approximate the content of the region more closely [49]. To represent more complex texture, Gaussian-Markov Random Field model parameters were used as features by Kwon and Chellappa [50]. DCT-based texture coding techniques such as Block-Padding [8] and the Shape-Adaptive DCT (SA-DCT) [45] have been thoroughly investigated within MPEG-4 experiments. In the former one, all pixels outside the region of interest are replaced by a value, which can be determined in a variety of ways (e.g. zero or mean of active pixels), prior to standard DCT encoding. In the SA-DCT, pixels are rearranged so that a smaller region than the original block needs to be transformed. The SA-DCT will be described in more detail in section 4.2. It was concluded in a recent study that the SA-DCT is superior to block-padding techniques [67]. However only the type of segments which arise in intra frame coding were considered.

It is now attempted to assess the applicability of one of the above techniques (i.e. the SA-DCT) to the coding of uncovered regions and propose new ones which are better suited. The improvement stems from the use of existing spatial information surrounding the failure region to predict its content. This prediction could simply consist in a mean value calculated from the neighbourhood. Two refinements to this basic technique are identified. The first is to utilise temporal clues to determine which pixels are likely to belong to the same object as the uncovered region. Secondly, if the object being uncovered is textured, a local operator for the prediction is more convenient.

In the next section we present the various techniques under investigation in detail. The simulation results for these methods are shown in section 4.3 after a brief description of the framework within which the texture coding component was tested. The relative merits of each method are also discussed. Section 4.4 contains concluding remarks.

### 4.2 Texture coding of failure regions

In order to devise a good way of sending interframe texture information, we first need to consider its characteristic properties so as to eliminate any redundancy
Figure 4.1: Illustration of failure regions for frame rate of 7.5Hz (left) and 1.875Hz (right)

in the data. An example of the shapes and sizes of uncovered regions occurring in a natural image sequence after the motion compensation step is shown fig 4.1. The regions are shaded in white and are shown for two different frame rates. In both cases the regions are thin and elongated. The content of these regions has to be intra-coded as it was never before projected onto the image plane.

A very popular technique for this purpose is the SA-DCT [45], in which an arbitrary segment is transformed such that the resulting number of coefficients is exactly the same as the number of pixels within the segment. This is achieved by first shifting all the active pixels to the top of a block and applying a 1D DCT to each of the active columns. The resulting coefficients are then shifted to the left of the block and the 1D DCT is again applied, this time to each of the active rows. This method performs well when blocks contain a large number of active pixels. Unfortunately this is not the case for natural sequences; failure regions tend to be very thin and lie adjacent to the boundaries of moving objects (see Fig 4.1). This means that there is not much intra-region correlation to be exploited. In addition, the regions have to be split into 8 x 8 blocks, which reduces the correlation even further.

Keeping in mind that most uncovered regions are likely to belong to the background and are each significantly thinner than the standard block frequently used for transform coding, we can say that some benefit could be de-
CHAPTER 4. EFFICIENT INTERFRAME CODING FOR A REGION-BASED SCHEME

Derived from assembling these regions together so that they form a small number of blocks. One simple way of achieving this is to fill blocks (which for the current purpose can be considered as packets) in a raster fashion with pixels from each region in turn (Raster Packing). The latter are also scanned in a raster fashion.

An alternative method is to predict a value for a region based on pixels in its close proximity. Fig 4.2(a) demonstrates the 8-connected neighbourhood (outlined with a bold line) for the failure region (in dark grey) caused by the motion of the object which is in the foreground. The region is filled with the average value of the intensities in the neighbouring area (Mean Fill), and any variations within the region can be added on by the prediction error coding stage. Note that bits are not spent explicitly by these types of texture coding component. Instead the quality of the texture prediction influences the bitrate produced by the prediction error coding component.

A better way of predicting the uncovered texture content is to take into account our knowledge of the interaction of the adjacent patches. For instance, for any given connected failure region, a few heuristics can be used to determine which object it was uncovered by. The texture of that object is unlikely to be correlated with that of the failure region. Thus the values of that region should not be used in the prediction of the failure region (see fig 4.2(b)). This technique will now be referred to as the Advanced Mean Fill. The clues we have used are magnitude of displacement and connectivity with adjacent regions. In cases where the desired prediction is not obtained, slightly more bits will be spent by the prediction error coding component. Wang and Adelson [87] have described more methods for decomposing image sequences into image layers which allow satisfactory reasoning about which object is projected onto the image plane.

The two above methods using one mean value can be expected to perform well when an uncovered region is uniform. However if it contains significant variations a better way is needed for approximating the region. A better approximation for the failure region texture can be obtained by calculating the prediction separately for every pixel of the region based on a small window
Figure 4.2: Picture area used for prediction for the Mean Fill method centred around the pixel (Dynamic Mean Fill). Figure 4.2(c) shows the neighbourhood used for the prediction of an example pixel. Because of the restricted area used for the prediction, the likely gain using this technique is highly dependent on the type of the image texture.

4.3 Results

The overall encoding framework within which the above-mentioned texture coding algorithms were tested was described in the previous chapter. It is important to consider the whole scheme, as the different components are inter-dependent
and so cannot be reliably judged in isolation.

In order to calculate the effective number of bits spent on transmitting revealed regions we have to consider both the texture coding branch \(^4\) and the prediction error branch \(^2\) (see figure 3.1), because it cannot be guaranteed that all the revealed information has been transmitted via the texture coding branch:

\[
\text{Total texture bits} = \text{Bits}^4 + \text{Bits}^2
\]

The first set of experiments were performed on the background area of the CIF resolution Calendar125 sequence; i.e. only the portion of the background which is revealed as a result of the camera panning to the left is coded as a failure region. Results were obtained on the first 64 frames at different frame rates - which means that a varying number of frames were skipped - so as to simulate a varying degree of motion and hence varying sizes of failure region. A manual segmentation was used in order to construct object descriptions as we wish to focus on dealing with failure regions in an ideal object based situation. The performance of all texture coding methods seen is likely to degrade in a completely real environment. The total number of bits spent on coding uncovered texture by using the techniques described in section 4.2 for different frame rates are shown in table 4.1. The PSNR and the visual quality in the failure areas were similar for all the techniques as is illustrated by fig 4.3. The results for the Mean Fill and the Advanced Mean Fill are the same because the region from which the failure region mean value is predicted coincide in this case. The prediction techniques perform well, especially considering the large amount of information in the background of this sequence.

Similar observations can be made for the Container sequence (see table 4.2). The primary observation is that both the Dynamic Mean Fill and the Advanced Mean Fill techniques work best even when the failure regions are rather large. Secondly, the SA-DCT always yields the worst results as most of the blocks end up having too few active pixels. Furthermore it is ignoring valuable information present in the form of neighbouring pixel intensities. However it is worth noting that the relative improvement in performance of the prediction techniques
4.3: Results

Figure 4.3: Decoded frame 48 using SA-DCT (top) and Dynamic Mean Fill (bottom)

<table>
<thead>
<tr>
<th>Skip</th>
<th>SA-DCT</th>
<th>Raster Packing</th>
<th>Mean Fill</th>
<th>Advanced Mean Fill</th>
<th>Dynamic Mean Fill</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>23123</td>
<td>20519</td>
<td>10068</td>
<td>10068</td>
<td>7377</td>
</tr>
<tr>
<td>8</td>
<td>18914</td>
<td>19334</td>
<td>11886</td>
<td>11886</td>
<td>9655</td>
</tr>
<tr>
<td>12</td>
<td>15880</td>
<td>17546</td>
<td>11356</td>
<td>11356</td>
<td>10013</td>
</tr>
<tr>
<td>16</td>
<td>16914</td>
<td>18347</td>
<td>12629</td>
<td>12629</td>
<td>11225</td>
</tr>
</tbody>
</table>

Table 4.1: Number of bits required to code uncovered regions in Calendar125
CHAPTER 4. EFFICIENT INTERFRAME CODING FOR A REGION-BASED SCHEME

<table>
<thead>
<tr>
<th>Skip</th>
<th>SA-DCT</th>
<th>Raster Packing</th>
<th>Mean Fill</th>
<th>Advanced Mean Fill</th>
<th>Dynamic Mean Fill</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>24302</td>
<td>13398</td>
<td>15101</td>
<td>10840</td>
<td>8830</td>
</tr>
<tr>
<td>8</td>
<td>15486</td>
<td>10831</td>
<td>12240</td>
<td>9498</td>
<td>7402</td>
</tr>
<tr>
<td>12</td>
<td>9858</td>
<td>5700</td>
<td>4858</td>
<td>1854</td>
<td>769</td>
</tr>
<tr>
<td>16</td>
<td>9917</td>
<td>6983</td>
<td>7933</td>
<td>5320</td>
<td>5096</td>
</tr>
</tbody>
</table>

Table 4.2: Number of bits required to code uncovered regions in Container

Figure 4.4: Detail for decoded texture

compared to the SA-DCT is smaller for larger skips. This is because the SA-DCT gets more efficient when there are enough active pixels within each block. The raster packing method does better than the SA-DCT for the Container sequence as it uses fewer blocks. An example of the quality of prediction which is possible using the Dynamic Mean Fill method is shown in figure 4.4.

One shortcoming of a region-based coding scheme is that the failure regions always coincide with parts of the image which are not of high quality due to
camera errors, for example blurring at sharp edges. This causes persistent
noisiness in portions of the image which are gradually uncovered (see fig 4.3).
Francois et al also mention this problem [30]. They attempt to solve it by a
filtering operation during the quantization stage of the coding process.

4.4 Conclusions

We have introduced new methods for texture coding which are more efficient
than the SA-DCT for arbitrarily shaped uncovered regions as we exploit the
known characteristics of such regions like their shape and their relationship
with neighbouring regions. The improved performance is due to the fact that
the number of bits spent on coding DC coefficients is greatly reduced either
by using prediction (Mean Fill, Advanced Mean Fill and Dynamic Mean Fill
methods) or by reducing the number of blocks which need to be coded (Raster
Packing method).
Chapter 5

Region Representation for a Region-based Coding Scheme

5.1 Introduction

As mentioned previously, the superior performance of second-generation schemes with respect to standard block-based ones remains to be established as various components of the former still need to be optimised. One of these components, motion compensation of arbitrarily shaped objects, is the subject of this chapter.

It is important to consider the actual algorithmic setup before concentrating on the design and applicability of lower level tools. Several video coding schemes with some region-oriented elements have been proposed, employing different segmentation approaches (spatial [78] or motion-based [64] or spatio-temporal [89]) or different kinds of object memory (templates [27] or layered [87]). For the purposes of this chapter we opt to only distinguish between those in which motion compensation is handled differently [30, 80]. The first applies forward motion compensation, i.e a region in frame “n” is transformed to the corresponding one in frame n + 1. In the latter, blocks or portions of blocks which form part of regions in frame n + 1 are mapped onto blocks in the previous frame, n. This is known as backward motion compensation and is similar to motion compensation in traditional block-based coding schemes.
During the process of reconstruction of individual frames of a video sequence, portions of the image moving by a non-integer amount have to be interpolated whether forward or backward motion compensation is used. In fact, this operation might have to be performed repeatedly on the same portion of image. The interpolation is usually carried out in coding schemes using a bilinear operator. Within the Image Processing community, several reconstruction kernels have been explored for applications such as enlarging of digital pictures [23]. Analysis in the frequency domain showed that all the interpolation kernels (e.g. linear, quadratic, Catmull-Rom cubic) have some low-pass filtering property. Repeated application of such a filter can degrade picture quality to a significant extent and cause an increase in bitrate for maintaining picture quality.

This is why we propose a representation which prevents repeated loss of high frequency information over a few frames. This is achieved by avoiding the cascading of interpolation operations on what is essentially the same portion of an image. Instead an internal representation of each region of the original image is kept, from which a grid representation can be derived whenever required. The internal representation itself does not get filtered. We outline the operation on this data structure later on for forward motion compensation. Backward motion compensation can also be performed on this internal data structure, but the sequence of operations would be slightly different.

The rest of the chapter is organised as follows: in the next section, we describe how an accurate motion estimate can be obtained for the type of region which we are focusing on. These are regions with significant high frequency content which undergo smoothing due to repeated compensation as will be shown in section 5.2. In section 5.3, a novel region representation is presented which avoids smoothing and thus wastage of bits. The manipulation of the data structure at different stages is also described in detail. We then provide comparative results in section 5.4 for regions with different types of motion. In the last section, the significance of using an internal floating point representation for all the regions within a natural image sequence will be assessed. We conclude
with a summary and several ideas for further investigation (section 5.5).

## 5.2 Smoothing due to Successive Interpolations

In this section we show that, as frames get reconstructed by a region-based codec, the effects of smoothing do build up for certain moving objects such that at each frame some information has to be sent to add the high frequency data back onto the objects.

### 5.2.1 Explanation of smoothing by theoretical analysis

Figure 3.1 shows the block diagram of a type of region-based codec. The encoder contains a decoder as it always needs to keep track of the reconstruction achieved at the decoder end. The collection of region descriptions maintained at both ends represents the data gathered from previous frame(s). Initially, the region description is obtained by some type of segmentation (spatial or spatio-temporal). In this particular codec the segmentation is performed in the decoded intra frame (which can be coded by a transform-based technique) on both sides of the system [92]. Alternatively, the segmentation could be performed only at the encoder which would however imply the transmission of costly contour information.

Operations applied by the encoder on the region data include motion estimation and compensation. The encoder is also responsible for keeping the region descriptions as close to real world objects as possible by analysing the input video content. Any updates on the internal data representations are communicated to the decoder. At the decoder side motion compensation is performed on this data (using the motion vectors sent through the communication channel) so as to obtain the prediction for the succeeding frame.

The conventional way of storing the region descriptions is to keep the reconstruction of the previous frame along with masks that define each region currently being used. Operations can then be applied to all the pixels within a given region by referring to the mask.
If we denote the \( i^{th} \) pixel's position and value by \( \bar{x}_i \) and \( \bar{p}_i \) respectively, then a region \( R \) can be defined as a set of its pixels:

\[
R = \{ \bar{x}_i, \bar{p}_i | i = 1, \ldots, M; \bar{x}_i \in \mathbb{Z}^2 \},
\]

(5.1)

where \( M \) is the number of pixels within the region. To perform forward motion compensation, each new pixel position \( (\bar{x}_i') \) is found by applying the motion vector to the old pixel position while leaving the intensity and chromaticity values unchanged:

\[
\bar{x}_i' = \bar{x}_i + \left( \begin{array}{cc} a_1 & a_2 \\ a_4 & a_5 \end{array} \right) (\bar{x}_i - \bar{x}_c) + \left( \begin{array}{c} a_3 \\ a_6 \end{array} \right),
\]

(5.2)

where \( \bar{x}_c \) is the centre of the region \( R \). The compensated region is:

\[
R' = \{ \bar{x}_i', \bar{p}_i' | i = 1, \ldots, M; \bar{x}_i' \in \mathbb{R}^2 \}
\]

(5.3)

At this stage the transformed points of the region (i.e. \( \bar{x}_i' \)) may not coincide with the grid structure of an image, for example if the object translates by a non-integer number (in case a translational motion model is used) or if it undergoes a complex motion like a zoom and/or rotation.

But the values at the grid positions of the image have to be reconstructed for two distinct purposes, i.e.

- for viewing, and
- for keeping the region representation for the next frame.

Thus spatial interpolation is needed to retrieve the intensity value of the required pixels from the known intensities of the surrounding transformed points. The final region as created for the image grid is thus,

\[
R'' = \{ (\bar{x}_j', \bar{p}_j'') | j = 1, \ldots, M''; \bar{x}_j' \in \mathbb{Z}^2 \},
\]

(5.4)

where \( \bar{p}_j'' = f(N(\bar{x}_j')) \), and \( f \) is the interpolation function and \( N \) is the neighbourhood over which the interpolation is carried out. The number of pixels now in the region, \( M'' \), can be different from the original number of pixels.

Bilinear interpolation is widely used in the coding community as a kind of spatial interpolation. It can provide satisfactory results for the viewer although
some smoothing occurs. The neighbourhood and interpolation functions for bilinear interpolation are defined next.

The neighbourhood \( N(x_i) \) is a subset of \( R' \) such that its elements are the four corners of the smallest quadrilateral enclosing the point \( x_i \). To avoid ambiguities where there are more than one smallest enclosing quadrilateral, we find it by first drawing a coordinate system centred on \( x_i \) with the \( x- \) and \( y- \) axes aligned with the row and column of the image grid respectively. The corners of the required quadrilateral are then the closest point in each quadrant to \( x_i \) of the coordinate system defined above.

The bilinear interpolation function is defined as follows. Consider a neighbourhood \( N = \{(x_{i'}, y_{i'})\} \) which is the quadrilateral enclosing grid point \( x_i \) after the transformation of each element of the region \( R \). The value of \( p_i \) of point \( x_i \) is extracted using the bilinear function.

\[
\tilde{p}_i = \frac{l_b - l_i}{l_b - l_t} \left[ \frac{k_{br} - k_{i}}{k_{br} - k_{bd}} \cdot \tilde{p}_{i'} + \frac{k_{bd} - k_{i}}{k_{bd} - k_{bd}} \cdot \tilde{p}_{i''} \right] + \frac{l_t - l_i}{l_b - l_t} \left[ \frac{k_{i} - k_{i'}}{k_{i} - k_{bd}} \cdot \tilde{p}_{i'} + \frac{k_{bd} - k_{i'}}{k_{bd} - k_{bd}} \cdot \tilde{p}_{i''} \right],
\]

where \( l_t \) and \( l_b \) are the locations where the \( y- \) axis intersects the upper and lower sides of the enclosing quadrilateral.

Owing to the averaging operations in the bilinear function, some smoothing occurs when the image values are extracted. The filtering effect is more or less pronounced depending on the distance between grid points and available compensated points, and on the amount of texture present in the region. The next step in the coding procedure, i.e. prediction error coding, can restore the region to an acceptable quality level. The region description after adding the prediction error is

\[
R'''' = \{(x_i, \tilde{p}_i''')|i = 1, \ldots, M\}
\]

This is the region which will be used as the reference region when reconstructing the following frame in the Inter mode. Note that the original values \( (p_i) \) of the region \( R \) have been discarded. The image can be judged to be satisfactory for viewing purposes. However it is not necessarily adequate as a reference for
Figure 5.1: Filtering effect on a block of samples due to successive applications of a linear filter

reconstructing the next frame especially if a high quantization factor is used for compressing the prediction error. The filtering effect of the interpolation becomes more serious in successive frames because the region is repeatedly interpolated and thus gets smoother. Francois et al have also come across this problem [30] with a similar coding scheme. They use biquadratic interpolation but this can only help to a certain extent, as will be experimentally shown later. The increased filtering effect implies that more bits have to be spent by the prediction error coding stage so as to restore the region to an acceptable level of detail.
5.2.2 Illustration of smoothing effect

We first illustrate how a signal gets affected by repeated averaging operations by means of a simple example. Consider the step shown in figure 5.1. We want to obtain the sample values at odd positions to simulate an interpolation operation. The resulting signal is shown in figure 5.1. The original samples have been discarded as this is what happens in a video compression system involving sub-pixel motion compensation. The filtering incurred in this step is inevitable, whichever type of interpolation filter is used (i.e. quadratic, cubic, spline, etc.). Now, we want to obtain the sample values at the even positions again. This means that additional filtering occurs such that the original edge in the signal is gradually smoothed to an unacceptable level.

Figure 5.2 illustrates an example of the degradation which occurs to a region through repeated interpolation, where 5.2(a) is the initial object description created at the decoder and 5.2(b), 5.2(c) show the object after 2 and 4 interpolations respectively. The motion of the object is about 0.5 pixel per frame to the right for this example. Objects moving at a speed closer to integer values or objects with less high frequency information would undergo a smaller amount of smoothing. The horizontal smoothing near the chimney area of the ship can be easily noticed. No vertical smoothing occurs because there is no motion in the vertical direction and therefore no vertical averaging is needed during the interpolation step. The colour components of the moving object are also affected by the interpolating filter but probably to a lesser extent because they contain fewer high frequencies than the luminance component.

In a coding system, the object description does not reach the state of distortion demonstrated in figure 5.2 because, for each inter-coded frame, a difference image is sent which can rectify errors introduced by the motion prediction step. This prediction error image is however not exact at the decoder side due to the quantization step during the transmission process. This means that the object description does not get restored to its original accuracy. Thus there is still an element of build-up in the accumulation of errors if the quantization step is large. In fact, there is a balance to be struck in setting the value of the quan-
Figure 5.2: Smoothing effect of successive bilinear interpolation operations

tization step: a small quantization step causes a high immediate bitrate but maintains the object description whereas a large quantization step causes accumulation of errors in the object description and thereby an increase in bitrate
5.3 Motion Compensation of Regions from a more Accurate Internal Representation

Sizeable savings in bitrate at a given quality can be achieved if a region representation is used which does not allow information previously available to the decoder to be thrown away by successive interpolations. Such a representation is one which keeps individual points of a region together with their associated intensity values independent of a grid structure, i.e.

$$R^m = \{(\vec{x}_i, \vec{y}_i) | i = 1...M; \vec{x}_i \in \mathbb{R}^2\} \quad (5.6)$$

We call this a floating point representation because the pixel positions are allowed to take floating point values. An image can be built from this representation for viewing purposes, after which the interpolated regions are never used. The interpolated regions are not kept as a reference for the next frame. Instead the floating point representation is kept for the latter purpose. This means that the smoothing effect does not accumulate and thus the content of the region can remain as accurate as in the first predicted frame.

It now remains to be seen how this region representation can be updated if the content of the region does not stay the same as in the first frame. This
happens, for example, if there are changes in the illumination of the object. Furthermore, if the motion is not accurately estimated or if the motion model used is not adequate for the region, the region representation does not tally with the content of the input video. These differences are calculated by the encoder and transmitted using a transform-based technique. The transmitted error values are available at the grid points, whereas the error values at the compensated points are needed to update the region representation. An interpolation from the neighbouring error values is thus needed for each point belonging to the region. The resulting error value is then added to the available value. Some smoothing of the image does occur but the interpolation is only done once on each error image and does not cause much accumulation of error.

Additional cost in computational complexity is incurred in the proposed scheme as two interpolation operations have to be carried out during each motion compensation step instead of just one. Furthermore, the interpolation from a non-integer grid can be more complex than standard interpolation. Memory requirements are also slightly higher for the new technique as an internal representation has to be kept for each object. However the gain in bitrate (or quality) can easily outweigh the above two considerations, especially in forthcoming transmission applications (for example, the videophone) where the channel capacity is fixed through physical limitations.

5.4 Results

We present the results for translational motion and for more complex affine motion separately as different factors such as the density and “orientation” of the floating points come into play. This will therefore affect the way in which the performance of the proposed technique is assessed.

5.4.1 Translational motion

To demonstrate our simulations we have chosen the Container and the Calendar125 sequences. In both sequences, we have considered only the moving
Figure 5.4: Improvement in bitrate due to float representation (Container sequence)
Quantization step = 8

Quantization step = 16

Quantization step = 32

Frame number →

- floating point representation
- Bilinear Interpolation
- Biquadratic Interpolation

Figure 5.5: Improvement in bitrate due to float representation (Calendar125 sequence)
object with some high frequency content, i.e. a portion of the ship in the Container sequence and the calendar in the Calendar sequence; the background has been ignored. Alternate frames were dropped, as the motion of the object happens to be in the range of 0.3 - 0.7 pixels every 2 frames and this is the range we want to experiment with. As discussed in section 5.2, this is when we expect the smoothing effect due to bilinear interpolation to be most noticeable. A synthetic segmentation was used to construct the initial region description as we wish to focus on the maintaining of the region over time. Different quantization step sizes (8, 16 and 32) were used to code the motion compensated error images as this parameter influences the behaviour of the system significantly as shown in section 5.2.2. The quantization step for the intra image was 16. A motion vector quantization step size of 0.1 was used because accurate motion measurement is also needed in order for the novel motion compensation algorithm to be most effective. Fortunately, the estimation of the motion of regions containing a large amount of texture can be done accurately. Thus good motion measurements can be exploited whenever they can be obtained. The quantized motion vectors were coded losslessly.

In order to measure the quality of reconstruction obtained by different techniques, the final number of bits spent on coding the residual motion compensated error over the area of the selected moving regions was considered. The interpolation techniques tried for the grid representation of regions were the bilinear and biquadratic ones. Pixel values were obtained from the floating-point representation of regions by fitting a plane to neighbouring available compensated points. The four nearest neighbours were selected and a plane was fitted to these using a least mean square method. Error values at compensated points from grid points were calculated likewise.

The results from the above three techniques are plotted on figure 5.4 and figure 5.5. It can be observed that the proposed technique substantially outperforms both types of interpolation on a standard grid structure for both examples. In addition the quality of the reconstructed objects is also higher as is shown objectively by the PSNR measurement on the luminance component.
only over the area of the moving object (see figures 5.4 and 5.5). It is also interesting to note that biquadratic interpolation actually provides limited benefit with respect to bilinear interpolation for both sequences. The gain might not be as significant in different situations. However, the performance of the new technique is never expected to be worse than traditional techniques unless the motion is such that the object can be seen at a much higher resolution in the succeeding frame than in the reference frame.

One can see where the benefit described above stems from by looking at figure 5.6, which is showing successive reconstructions for a moving object similarly to figure 5.2. In the present case, most of the detail of the object gets faithfully rendered which means that very few bits have to be spent by the prediction error component on this region.

Other trends in our data can also be explained from a theoretical viewpoint:

- Looking at the results for quantization step of 8 for the Container-sequence, the bitrate is high at first because the Intra frame was coded with a quantization step of 16 and therefore the Inter frame coding can add some details missed by the Intra coding step. Our technique successfully manages to add enough detail to keep the bitrate low and quality constant after a few frames. The other techniques reach an equilibrium level at a higher bitrate because in each motion prediction step, the object is smoothed.

- For a quantization step of 16, the bitrate rises for bilinear and biquadratic interpolation as the smoothing effect quickly builds up before reaching a sub-optimal equilibrium, while our technique can again maintain the bitrate low.

- At a high quantization step of 32, no equilibrium is reached in the time window depicted because the content of the transmitted prediction error image is not sufficient to compensate for the smoothing effect. The PSNR falls rapidly for all three techniques but the new technique always results in the best quality measurement.
Figure 5.7 shows the quality of prediction at frame number 14 which is possible using the float representation (fig. 5.7(a)) compared to the bilinear interpolation from the previous grid representation (fig. 5.7(b)). The predicted image is shown in the leftmost part of each figure. The corresponding error image (middle) obtained using the float representation contains fewer variations. The final reconstruction (right) is visually more pleasant because the high frequency edges have been retained.

5.4.2 Affine motion

Preliminary experiments were also carried out to measure the performance of the proposed motion compensation tool where there is more complex motion, for example any combination of translation, rotation, shear or zoom. The most interesting case is zooming in, i.e. when the object gets larger with respect to the image plane because it moves closer or because of camera zoom. Thus more detail of the interior of the object gets revealed with time and the object occupies more pixels. However only the original number of points are kept by the internal region description. Affine motion compensation is different from translational motion compensation in another respect: the displacement of compensated points from integer grid positions is variable in the former case. It therefore has to be seen experimentally whether improvement in performance can still be achieved as the lower bound on the performance cannot be theoretically predicted.

We have used a simple sequence in which the camera zooms in on a textured, planar surface. The bitrate and PSNR comparisons are shown in figure 5.8. Similar trends as for translational motion can be observed, i.e. there is simultaneous decrease in bitrate and increase in measured quality for a fixed quantization step during the prediction error coding stage. Therefore, our proposed technique can be valuable even in circumstances of more complex object motion than translation.
5.4.3 Floating point representation within a complete region-based coding system

As seen from the previous sections, there is a definite advantage in using a precise internal representation for certain regions, namely those which contain a significant amount of detail and undergo sub-pixel motion, as predicted from theory. The frequency of occurrence of above-mentioned regions will be one determining factor for the magnitude of the overall improvement when coding a whole image sequence.

In this section, the significance of using an internal floating point representation for all the regions within a natural image sequence will be assessed. It will also be possible to provide a comparison with the performance achieved by a block-based system.

The basic experimental setup will be outlined in the next section and specific results and observations will be presented in section 5.4.3.

Experimental Setup

For the region based coding scheme, the experimental setup used is the same as described in chapter 3. The floating point representation is used instead of the pixel-based representation for the region texture. This means that all the region operations (i.e. merging, splitting and merging with the failure pixels) had to be slightly modified to work on the new representation.

The H.263 video compression algorithm, without any of the advanced options, was used for block-based coding. The motion estimation component, i.e. the RHT-based motion estimation, was the same for both systems.

Results

Experiments were carried out on several image sequences from the H.263 test set. A summary of the total bitrates and average PSNR achieved for each of these sequences is shown in figures 5.9 and 5.10. The use of the floating point representation is found to produce a lower bitrate and a slightly higher PSNR compared to the standard bilinear interpolation for the majority of the
sequences. The exceptions were the BREAM, BUS150 and FOREMAN sequences, for which the block-based codec is still superior mainly because of the imperfect segmentation in the region based codec as will be seen further in section 5.4.5. First, a closer observation of the effect of the floating point representation on a region based codec is carried out in the following section.

5.4.4 Effect of the floating point representation on a region based codec

A considerable improvement in the quality of motion compensation for the Calendar125 image sequence in particular was observed (see figure 5.11). The motion compensated error images for the intensity channel for frame 8 with and without using the floating point representation are shown in figures 5.13(a) and 5.13(b) respectively. As expected, large errors occur where the region description does not model the scene adequately, i.e. over the ball and the train and also where a region straddles the background and the calendar in the upper portion of the image. However, in figure 5.13(b), smaller, high frequency errors appear even in regions for which the motion has been correctly estimated. These are due to the low-pass filtering effect of the interpolation step during motion compensation and are drastically reduced when the floating point representation is used (figure 5.13(a)). Similar effects occur in the chrominance channels as well, and especially so for this sequence which contains sharp edges in the chrominance channels.

Figures 5.13(c) and 5.13(d) demonstrate the relative number of bits spent in coding the corresponding motion compensated error image (the intensity of each block is proportional to the number of bits required to code the block). It can also be seen that in portions of the image where a good motion estimate has been obtained, much fewer bits are needed in the case where the floating point representation has been used while where a good motion estimate is not available, the number of bits needed is similar in both cases.
5.4.5 Comparison with a block based codec

In the previous chapter, no comparison with a block-based video coding scheme could be provided because the required object (e.g. the calendar) could not be tracked as in a region-based coding scheme. This comparison is now possible because the bitrate due to the MC error of the whole frame is available for both schemes (see figure 5.11). It must be noted however that this overall figure is not the sole indication of the merit of either scheme. The region based scheme could be improved, for example, by finding an initial region description which is closer to an optimum one. It was still found that useful observations could be made from such an experiment. In figure 5.14, it can be seen that the background and calendar parts of the scene are compensated better within the region-based coding scheme while the block-based model is more suitable for certain other parts. This means that although the overall bitrate for the region-based scheme is higher at present, there is scope for improvement if a better scene description can be constructed. Similar results were obtained for the Container sequence, for which the various bitrates are shown in figure 5.12.

5.5 Conclusions

We have demonstrated the smoothing effect for certain regions due to successive interpolation operations inherent in region-based codecs. The technique which we propose in order to bypass this problem was shown to be very effective in terms of reduction in bitrate for better image quality. This is because accurate information sent to the decoder for the intra frame is kept through a novel region representation. The way to update these regions, should their content vary, was also defined.

It has also been shown experimentally that the use of the floating point representation of regions is advantageous for most of the natural image sequences in the H.263 test set. In certain cases the coding efficiency of the region-based coding has been brought up to that of the block-based technique while the former also provides the added advantage of enabling content-based functionality.
There is much scope for investigation related to the subject of this chapter. For example, it would be interesting to see whether different interpolation methods such as bilinear interpolation or quadratic fitting in conjunction with the new technique could improve the reconstruction of moving regions. Another way in which the new region representation can benefit a region-based codec would be in the temporal interpolation step, where one attempts to construct a prediction for dropped frames. Because an accurate object description is available for both the frames before and after the one which is to be interpolated, a very good reconstruction will be obtained. Furthermore, the region representation could also be applied to the scheme of template-based video coding [27], in which templates of regions occurring at some point within the video sequence are kept in a background memory. In this way, the problem of selecting between the internal region representation and the previous frame mentioned in section 5.4 would also be solved.
Figure 5.6: Successive reconstructions derived from the internal floating point representation
(a) Prediction from internal float representation

(b) Bilinear interpolation from a grid with quantization 16

Figure 5.7: Frame 14 of Calendar125 with different prediction schemes; left - motion compensated (MC) prediction, middle - MC prediction error, right - final reconstruction.
Figure 5.8: Improvement in bitrate and quality due to float representation 
(Lennart sequence; Quantization = 16)
Figure 5.9: Bitrate comparison for H.263 test sequences

Figure 5.10: PSNR comparison for H.263 test sequences
CHAPTER 5. REGION REPRESENTATION FOR A REGION-BASED CODING SCHEME

Figure 5.11: Improvement in bitrate due to floating point representation (Calendar125 sequence)

Figure 5.12: Improvement in bitrate due to floating point representation (Container sequence)
Figure 5.13: Comparison of motion compensated error with and without floating point representation for frame number 8 of the Calendar125 sequence.
Figure 5.14: Comparison of motion compensated error using the region-based coding method with floating point representation for frame number 36 of the Calendar125 sequence
In chapter 3, it was seen that the behaviour of several components of the region-based coding system relies on the setting of the parameters. In this chapter, this behaviour will be studied with a view to selecting the optimum set of parameters to be used for achieving large compression ratios. The effect of the tuning of individual components on the whole system will be first studied in the following section while the tuning of the complete system will be discussed in section 6.1.5. The implemented coding system will be evaluated with respect to other region-based coding systems in section 6.2. Quantitative comparisons with a block-based system and with MPEG4 will be presented in sections 6.3 and 6.4 respectively.

6.1 Evaluation of the RBC system

In this section, a few of the components of the region-based coding system are evaluated. In chapter 3, these components were described and design decisions were made based on their expected behaviour. Section 6.1.1 first confirms that a good segmentation is imperative for achieving a low bitrate. An example of the quality of the segmentation on the decoded intra frame for different quantization values used in the intra coding step is presented in section 6.1.2.
The motion segmentation and motion-based merging components are evaluated in section 6.1.3 and 6.1.4 respectively. Finally, the overall bitrate achieved by the region-based coding system is presented (section 6.1.5).

6.1.1 Manual segmentation

One of the barriers in assessing the potential of a region-based coding system is the initial segmentation. A segmentation which represents the objects in the scene well is needed in order to achieve good compression. An example of the compression which can be achieved if an ideal segmentation is available is therefore presented next.

The experiment was carried out on the Container sequence and the manual segmentation on the first frame is shown in figure 6.1. The main object is subdivided so that a simple motion model (translational) can be used to represent its motion adequately and because we wish to compare it with the block based system which is tuned to work with the translational model as well. The results are shown in the following tables (6.1 and 6.2). For the block-based system, the number of bits spent on coding the motion vectors is very high. For the region-based system, there is some overhead in sending the region label and region splitting and merging information, but this is minimal. The overall performance of the two systems are similar in terms of PSNR and bit-rate.
### 6.1: Evaluation of the RBC system

<table>
<thead>
<tr>
<th>Frame no.</th>
<th>Prediction error (PE) bits</th>
<th>Motion vector (MV) bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1100</td>
<td>2297</td>
</tr>
<tr>
<td>8</td>
<td>1629</td>
<td>1997</td>
</tr>
<tr>
<td>12</td>
<td>2006</td>
<td>1831</td>
</tr>
<tr>
<td>16</td>
<td>1973</td>
<td>1912</td>
</tr>
<tr>
<td>20</td>
<td>1927</td>
<td>1931</td>
</tr>
<tr>
<td>24</td>
<td>2006</td>
<td>1839</td>
</tr>
<tr>
<td>28</td>
<td>2027</td>
<td>1966</td>
</tr>
<tr>
<td>32</td>
<td>1932</td>
<td>1745</td>
</tr>
<tr>
<td>36</td>
<td>2087</td>
<td>1855</td>
</tr>
<tr>
<td>40</td>
<td>2116</td>
<td>1610</td>
</tr>
<tr>
<td>44</td>
<td>2051</td>
<td>1801</td>
</tr>
<tr>
<td>48</td>
<td>2187</td>
<td>1845</td>
</tr>
<tr>
<td>Total</td>
<td>23041</td>
<td>22957</td>
</tr>
</tbody>
</table>

Average $PSNR_y = 31.0571 dB$

Average $PSNR_u = 38.4552 dB$

Average $PSNR_v = 38.3632 dB$

Average Bit Rate = 0.0911678 bits/pixel

Total Number of Bits = 120149

Average Bit Rate = 73.5606 Kbits/s

Table 6.1: Bit spending in the block-based H.263 coding scheme

#### 6.1.2 Spatial segmentation on the decoded intra image

This is the initial segmentation carried out on the first intra-coded frame. It is the first set of regions which is inserted into the region database and thus affects the entire compression/decompression process even though modifications can subsequently be made to the database through region splitting and merging.

The quality of the transmitted intra frame determines that of the segmentation performed on it to a large extent. For instance if the intra coding introduces artifacts, such as spurious edges, blurring or ringing, the segmentation process will definitely be affected as it generally relies heavily on the intensity/chromaticity differences between neighbouring pixels in order to identify boundaries between objects.

The intra coding component used was DCT-based which is widely known to cause blocking artifacts at high compression ratios due to the coarse quantization of the transform coefficients. The larger the quantization step, the higher
Table 6.2: Bit spending in a region-based coding scheme with an ideal segmentation

is the compression ratio but also the lower the quality of the reconstructed intra image and subsequent images which are predicted from the first one. It is thus desirable to keep the size of the quantization step within a range such that it produces a good quality intra image and does not impair motion prediction drastically for subsequent frames. The intra quantization step also influences the quality of the initial segmentation. It would be useful to determine whether a good segmentation can be obtained throughout the usable range of the quantization step. It can be seen from figure 6.2 that a good segmentation is obtained even if the quantization is very high and produces a blocky decoded intra image.

6.1.3 Motion segmentation

The instances where the motion segmentation component is utilised are when the spatial segmentation is faulty or when a new region enters the scene after
Figure 6.2: Quality of spatial segmentation with varying intra Quantization step for the *Silent* sequence

the intra coding. In order to test this component, the system was operated by starting with an extremely coarse segmentation, which means that certain regions straddle boundaries of the objects in the scene. It can be seen that although the indicated region does get segmented, the exact motion boundary is not found.
CHAPTER 6. TUNING AND EVALUATION OF THE REGION-BASED CODING SYSTEM

Figure 6.3: Refinement of segmentation using motion information

Table 6.3: Variation in bitrate for Calendar125 sequence using different merge factors

<table>
<thead>
<tr>
<th>Merge factor</th>
<th>Minimum region size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300</td>
</tr>
<tr>
<td>0</td>
<td>558456</td>
</tr>
<tr>
<td>1.5</td>
<td>536689</td>
</tr>
</tbody>
</table>

6.1.4 Motion-based Merging

This component is very useful in situations where an object gets oversegmented by the initial decomposition step. For example, the background of a scene can contain complex spatial detail but usually undergoes a simple motion if the camera is moved, in which case a single motion vector is required. The processing of the Calendar125 sequence illustrates this (see table 6.3). The bitrate is significantly lowered by increasing the merge factor. The biggest improvement is for the case where the initial region size is small.
6.1.5 Performance of the complete system

It has been seen in the previous sections that when parameters of the system are varied independently, the effect on the performance is visible. The task of finding the optimal parameter set is much more complex, however, because the different parts of the system are closely inter-related. Unfortunately this problem does not lend itself well to numerical optimisation in practice, first because of the high dimensionality owing to the large number of parameters combined with the high time complexity of the developed system, and secondly the optimisation would need to be performed on a significant portion of the sequence, thereby excluding low-latency applications. Observations from such an experiment can still be useful for the purposes of setting parameters in situations where some \textit{a priori} information is available about the type of the image sequence to be coded. Thus the system can be tuned to several applications depending on the class of image sequence expected by each.

Figure 6.4 shows the bitrates which can be achieved for the sequences in the H.263 test set by the region-based coder. The parameters for this experiment were found by a simple grid-based search (see table 6.4). The following parameters were defined in chapter 3: spatial segmentation threshold (T1) and minimum region size (T2) in section 3.1, split factor (T3) and motion segmentation threshold (T4) in section 3.5 and finally merge factor in section 3.6. This search was not exhaustive because the useful range and granularity of the parameters is not known. Although it might be possible to find better sets of parameters for each of the sequences, other tools which help in finding better object representations might be more effective for the purpose of lowering the bitrate.

6.2 Comparison with region-based coding schemes

A quantitative comparison with other region-based coding techniques (for example, the ones described in section 2.1) was not possible because the whole systems could not be implemented. However it can be said that no region-based
coding system which performs better than H.263 for a general class of sequences has been reported in the literature. Yokoyama et al have produced comparative results only with respect to H.261 which has already been proven to be much less efficient than H.263 ([3]).

6.3 Comparison of the optimised region-based system with a block-based system

The standard H.263 coder was used as the block-based coder against which the comparison was made for the purpose of this experiment. Because the region-based coding system can be configured in a variety of ways, it is difficult to

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Minimum region size</th>
<th>Spatial Segmentation threshold</th>
<th>Split factor</th>
<th>Motion segmentation threshold</th>
<th>Merge factor</th>
<th>Bitrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>500</td>
<td>30</td>
<td>0.7</td>
<td>8</td>
<td>0.1</td>
<td>88064</td>
</tr>
<tr>
<td>bream</td>
<td>400</td>
<td>25</td>
<td>0.7</td>
<td>8</td>
<td>2</td>
<td>401352</td>
</tr>
<tr>
<td>bus150</td>
<td>400</td>
<td>35</td>
<td>0.7</td>
<td>8</td>
<td>0.1</td>
<td>785389</td>
</tr>
<tr>
<td>calendar125</td>
<td>500</td>
<td>35</td>
<td>0.8</td>
<td>8</td>
<td>2</td>
<td>527328</td>
</tr>
<tr>
<td>children</td>
<td>700</td>
<td>35</td>
<td>0.7</td>
<td>8</td>
<td>2</td>
<td>425045</td>
</tr>
<tr>
<td>coastguard</td>
<td>500</td>
<td>35</td>
<td>0.7</td>
<td>10</td>
<td>2</td>
<td>310947</td>
</tr>
<tr>
<td>container</td>
<td>400</td>
<td>25</td>
<td>0.7</td>
<td>8</td>
<td>2</td>
<td>126859</td>
</tr>
<tr>
<td>foreman</td>
<td>500</td>
<td>35</td>
<td>0.7</td>
<td>10</td>
<td>2</td>
<td>253741</td>
</tr>
<tr>
<td>hall</td>
<td>500</td>
<td>35</td>
<td>0.7</td>
<td>10</td>
<td>2</td>
<td>99763</td>
</tr>
<tr>
<td>mad</td>
<td>600</td>
<td>35</td>
<td>0.7</td>
<td>10</td>
<td>2</td>
<td>71476</td>
</tr>
<tr>
<td>news</td>
<td>400</td>
<td>20</td>
<td>0.7</td>
<td>8</td>
<td>0.1</td>
<td>151187</td>
</tr>
<tr>
<td>silent</td>
<td>700</td>
<td>35</td>
<td>0.7</td>
<td>8</td>
<td>2</td>
<td>114288</td>
</tr>
<tr>
<td>td</td>
<td>700</td>
<td>35</td>
<td>0.7</td>
<td>8</td>
<td>2</td>
<td>144297</td>
</tr>
<tr>
<td>weather</td>
<td>600</td>
<td>25</td>
<td>0.8</td>
<td>10</td>
<td>0.9</td>
<td>254735</td>
</tr>
</tbody>
</table>

Table 6.4: Optimum coding parameters for H.263 sequences
make a meaningful comparison. Indeed, the optimal parameter set can vary from sequence to sequence. As described in section 6.1.5, the parameters were defined for each sequence individually. Ideally, the optimal set of parameters should be calculated from a training set and then the system should be evaluated using an independent test set for an unbiased comparison. This was however not possible owing to the small number of samples in the H.263 test set. The results of the comparison are shown in figure 6.4. Compared to the H.263 block-based codec, the developed codec has a better performance only for the Coastguard sequence.

6.4 Comparison of the region-based system with MPEG4

The comparison of the region-based coding system with MPEG4 was made for two types of situation: first where the alphamaps for each frame of the sequence is available (section 6.4.1) and secondly where only the natural video input is available (section 6.4.2). Applications requiring video compression generally
6.4.1 Externally generated alphamap

As was seen from chapter 2, the MPEG4 video compression system is significantly different in structure from the implemented region-based coding system. The most important difference is that the behaviour of MPEG4 is optimised for a specific type of video input (Video Objects), although it can also accept standard rectangular shaped input as a special case. Video Object information comprises of the shape, location and interior texture of the object for every single frame. This information can be readily available for synthetically generated video material. For natural video material however, not only is a segmentation for each frame needed but also the correspondence information between regions in successive frames. Thus for the processing of natural video, MPEG4 can be viewed as the symbol coding, back end of a complete compression system (see figure 6.5). The front end of the system is the analysis component which is responsible for decomposing the natural video input into Video Objects. The developed region-based compression system can also be viewed in a similar manner except that it encompasses both the analysis and the symbol coding stages.
Figure 6.6: Experimental setup for comparison of video object symbol coding

It can therefore be said that MPEG4 and the implemented region-based coding system address different functionality and are therefore difficult to compare. One advantage of MPEG4, however, is that it falls back to MPEG2 mode if the shape information of objects within the scene is not available. If used in this mode, all the comments in the previous section about the comparison between a block-based system and a region-based system apply, i.e. even though the region based system is currently less efficient, it has more potential for improvement.

Table 6.5 shows that RBC is more than twice as efficient as MPEG4 in terms of compression. However the MPEG4 bitstream also contains shape information which can allow content-based functionality at the decoder end. RBC decomposes the scene into regions well enough to provide a low bitrate but this decomposition does not accurately match the scene content.

### 6.4.2 Objects from automatic segmentation

In order to compare the symbol coding capability of the two systems, the same video object input has to be used for both. This was achieved by first coding the video sequence with RBC and then using the resulting video object description...
as input to the MPEG4 symbol coding component (see figure 6.6).

The RBC can be configured to yield different coding efficiency for any particular sequence. As seen from section 6.1, certain parameters affect the efficiency a good deal. This means that the performance of the RBC with respect to MPEG4 will vary depending on the configuration adopted for the video analysis component. In the following section, the cases where RBC is expected to perform better than MPEG4 will be demonstrated. Afterwards, the results on data from the H.263 test set will be summarised.

Detailed comparison

If any of the following conditions apply, RBC is highly likely to result in a lower bitrate than MPEG4:

- The motion model for each of the regions in the scene is adequate. This means that each of the regions in the scene can be correctly predicted using the chosen motion model. Since only one motion vector is required per region as opposed to one motion vector per macroblock in MPEG4, large savings can be made on bits required for motion representation. The translational motion model was used for the experiments in this section for RBC. Another potential advantage of using one motion vector per region is that higher motion models could be better exploited.

- There is a sufficiently large number of regions in the scene. This is obvious from table 6.6 and the first three lines of table 6.7. It was not possible to experiment with very large numbers though as only 32 objects are allowed in MPEG-4. In RBC, the penalty for having many regions, i.e. expensive contour coding, is avoided by performing the initial spatial segmentation on both sides of the codec. The spatial segmentation of newly-uncovered regions is also done on both sides of the codec.

- There is a high degree of activity in the scene such that splits and merges are needed to maintain a good region description. This is mainly because the description of a split or merge is not directly accommodated by the
6.4: Comparison of the region-based system with MPEG4

<table>
<thead>
<tr>
<th>No of regions</th>
<th>MPEG4</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>120792</td>
<td>143883</td>
</tr>
<tr>
<td>10</td>
<td>132960</td>
<td>136841</td>
</tr>
<tr>
<td>12</td>
<td>150760</td>
<td>135965</td>
</tr>
<tr>
<td>16</td>
<td>126120</td>
<td>96214</td>
</tr>
</tbody>
</table>

Table 6.6: Symbol coding efficiency comparison between MPEG4 and RBC for Container sequence

<table>
<thead>
<tr>
<th>Split factor</th>
<th>Merge factor</th>
<th>Last frame</th>
<th>Minimum region size</th>
<th>No of objects</th>
<th>MPEG4</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>36</td>
<td>600</td>
<td>18</td>
<td>484688</td>
<td>499166</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>36</td>
<td>500</td>
<td>22</td>
<td>506272</td>
<td>493330</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>36</td>
<td>300</td>
<td>30</td>
<td>545696</td>
<td>492307</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>60</td>
<td>300</td>
<td>30</td>
<td>816512</td>
<td>767360</td>
</tr>
<tr>
<td>0.1</td>
<td>1.0</td>
<td>36</td>
<td>400</td>
<td>27</td>
<td>586288</td>
<td>474217</td>
</tr>
<tr>
<td>0.7</td>
<td>0.9</td>
<td>20</td>
<td>500</td>
<td>28</td>
<td>408376</td>
<td>333026</td>
</tr>
</tbody>
</table>

Table 6.7: Symbol coding efficiency comparison between MPEG4 and RBC for Calendar125 sequence

current MPEG4 syntax as the MPEG4 model excludes interaction among individual video objects. However split and merge information produced by the video analysis component can be indirectly fed into MPEG4 by defining new objects whenever a split or merge occurs. This means that the inner texture and the contour on the new objects have to be coded again in the intra mode which immediately causes a substantial rise in number of bits. This can be seen from the last two rows of table 6.7.

The decomposition of bits used for coding shows that that Shape and Motion bits are higher in MPEG4 (table 6.8) than in RBC (table 6.9), but Texture bits are much lower. This is probably because the translational motion model is not
CHAPTER 6. TUNING AND EVALUATION OF THE
REGION-BASED CODING SYSTEM

<table>
<thead>
<tr>
<th>Frame number</th>
<th>Shape bits</th>
<th>Motion bits</th>
<th>Texture bits</th>
<th>Syntax bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5650</td>
<td>0</td>
<td>42639</td>
<td>34688</td>
</tr>
<tr>
<td>4</td>
<td>1647</td>
<td>1505</td>
<td>1236</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>1093</td>
<td>1464</td>
<td>1551</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>937</td>
<td>1811</td>
<td>2169</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>1091</td>
<td>1479</td>
<td>2035</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>1171</td>
<td>1673</td>
<td>2993</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>1342</td>
<td>1492</td>
<td>1895</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>1119</td>
<td>1231</td>
<td>1987</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.8: Decomposition of bits produced by MPEG4 coding for the *Container* sequence

<table>
<thead>
<tr>
<th>Frame number</th>
<th>Shape bits</th>
<th>Motion bits</th>
<th>Texture bits</th>
<th>Syntax bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>54479</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>220</td>
<td>4574</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td>240</td>
<td>6359</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>220</td>
<td>6407</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>220</td>
<td>6793</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>240</td>
<td>7307</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>16</td>
<td>200</td>
<td>6915</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>16</td>
<td>220</td>
<td>6804</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6.9: Decomposition of bits produced by RBC for *Container* sequence

adequate for representing the motion of the regions in the region-based coding technique.

Global comparison

The experiment was repeated on each of the sequences from the H.263 test set. The configuration which delivered the best result in the optimisation experiment
was used for each sequence. This was done so as to be able to exploit the strengths of the region-based coding scheme. It can be seen from table 6.10 that MPEG4 does not cope well. This is because the region description has to be changed through splitting and merging for the best performance within a region-based coding scheme.

### Table 6.10: Comparison between optimised RBC and MPEG4

<table>
<thead>
<tr>
<th>Sequence</th>
<th>MPEG4</th>
<th>RBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>118377</td>
<td>85970</td>
</tr>
<tr>
<td>bream</td>
<td>444096</td>
<td>464134</td>
</tr>
<tr>
<td>children</td>
<td>607512</td>
<td>525391</td>
</tr>
<tr>
<td>coastguard</td>
<td>362080</td>
<td>316368</td>
</tr>
<tr>
<td>container</td>
<td>277472</td>
<td>122484</td>
</tr>
<tr>
<td>foreman</td>
<td>439752</td>
<td>264263</td>
</tr>
<tr>
<td>hall</td>
<td>220744</td>
<td>100227</td>
</tr>
<tr>
<td>mad</td>
<td>109320</td>
<td>73422</td>
</tr>
<tr>
<td>news</td>
<td>187776</td>
<td>147379</td>
</tr>
<tr>
<td>silent</td>
<td>199688</td>
<td>111191</td>
</tr>
<tr>
<td>td</td>
<td>164128</td>
<td>149054</td>
</tr>
</tbody>
</table>

6.5 Conclusions and future work

In this chapter, it was first seen that the varying of parameters for components has to be done globally because each parameter affects different branches of the system. For example, allowing many regions to be split can lead to better motion prediction but also to a high cost in contour coding. In section 6.2, it was seen that the proposed region-based coding system exploits the full potential of having arbitrarily-shaped regions better than the one described by Yokoyama et al [92]. Indeed, the coding performance can get close to a block-based system (section 6.3) but unfortunately a separate optimisation of parameters for each
individual sequence was needed. This can still be useful for applications which expect a given class of video input, in which case the codec can be tuned for that particular class. From the above section, it was seen that there were three types of situation which are better handled by the RBC scheme than by MPEG4. However the comparison with MPEG4 should be viewed in light of the fact that the goals of MPEG4 are different.

The future work which builds upon the observations made from this chapter are that more experimentation is needed for determining the optimal parameter set for coding a given sequence or a given class of sequences. Furthermore more complex models than the planar patch model are needed in order to improve the prediction in a large variety of situation which occur in natural video sequences. The coding performance of the region-based system approached that of the block-based one in this work and the inclusion of higher level models is expected to contribute further towards achieving this goal while providing added functionality. Region-based systems with a different architecture from the one developed can be built so as to quantify exactly the ability of the system to exploit region-based models.
Chapter 7

Conclusions

7.1 Summary

As introduced in the first chapter of this thesis, the purpose of this work was to identify improvements to the components of a region-based codec with the added condition that the observations and tests should be carried out at the system level. However, the scope was limited to a basic system due to the small scale of the project. The rationale for this effort on region-based coding systems was also discussed.

The state-of-the-art for region-based video compression including its constituent components was reviewed in chapter 2. One important conclusion from the literature survey was that in order to obtain the best performance from a region-based codec, the superior motion representation aspect of arbitrarily-shaped regions should be exploited while minimising the amount of contour information to be sent. Based on these observations, a baseline region-based codec was designed and implemented. This system was described in detail in chapter 3.

Experiments with the baseline system allowed the observation of the type of failure regions which occur in natural image sequences. The shape of failure regions is generally different from that of regions corresponding to real objects. Thus, it was seen that applying the same inner texture coding technique for both types of regions was not appropriate. In chapter 4, novel ways of coding
failure regions were described and tested.

It also became apparent that the successive motion compensation of a region entailed degradation in the quality of the region (chapter 5). This was due to the interpolation step required for non-integer motion vectors. A simple solution was found to this problem, i.e. a floating point representation of the region was kept, so that there was no need for an approximation onto an integer grid after each non-integer motion compensation operation. This means that the region was kept at its original precision. A comparison of the bitrate with respect to a block-based system was also provided.

In chapter 6, the main components of the region-based coding system were first evaluated. Then a comparison was made with other video coding systems, i.e. other region-based coding systems, a block-based system and the MPEG4 standard. The results when compared to those of the block-based system were promising as it was demonstrated that if the regions represent real objects, the performance of the region-based codec was very good. During this project a lot of effort was dedicated towards extracting an accurate region description from a natural image sequence, but still more work is needed in this area. With regards to the comparison with MPEG4, the results of the proposed region-based codec was much better in certain cases, but the set of functionalities required was also an important factor in the comparison.

7.2 Future Work

Several improvements can be built upon the baseline video codec. For example, the toolset can be expanded to include more of the tools described in chapter 2. Thus for each function a selection of tools would be available to choose from. Finding the optimum tool configuration, as well as the associated optimum control parameters, for each given class of input automatically can be an interesting subject for research.

In addition, the baseline codec can be improved in terms of functionality (such as quality/temporal/resolution/content-based scalability, rate control, temporal interpolation and content-based manipulation) so as to investigate the
benefits of the suggested improvements in a fully functional video compression system. For example, the novel motion compensation technique is expected to be useful for temporal interpolation.

Work remains to be done in order achieve automatically an object description which adequately matches the real-life situation. This was achieved to some extent by the proposed region-based codec which modelled the scene as a set of planar patches. Higher motion models need to be investigated thoroughly. Ideally better object models than planar patches are needed so as to allow the representation of complex motion such as non-rigid motion, body motion, 3D motion, motion of objects containing intricate structures (e.g. trees). The tasks involved in attaining this object description will be the selection of objects along with the appropriate model with which to represent its movement throughout the duration of the sequence. Through the use of these complex models, hopefully a lower bitrate can be achieved than block-based video coding as well as powerful content-based functionality.
Bibliography


