To

my parents
Abstract

In this thesis, most commonly encountered video compression techniques and international coding standards are studied. The study leads to the idea of a reconfigurable codec which can adapt itself to the specific requirements of diverse applications so as to achieve improved performance.

Firstly, we propose a multiple layer affine motion compensated codec which acts as a basic building block of the reconfigurable multiple tool video codec. A detailed investigation of the properties of the proposed codec is carried out. The experimental results reveal that the gain in coding efficiency from improved motion prediction and segmentation is proportional to the spatial complexity of the sequence being encoded.

Secondly, a framework for the reconfigurable multiple tool video codec is developed and its key parts are discussed in detail. Two important concepts virtual codec and virtual tool are introduced. A prototype of the proposed reconfigurable multiple tool video codec is implemented. The codec structure and the constituent tools of the codec included in the prototype are extensively tested and evaluated to prove the concept. The results confirm that different applications require different codec configurations to achieve optimum performance.

Thirdly, a knowledge based tool selection system for the reconfigurable codec is proposed and developed. Human knowledge as well as sequence properties are taken into account in the tool selection procedure. It is shown that the proposed tool selection mechanism gives promising results.

Finally, concluding remarks are offered and future research directions are suggested.
Acknowledgement

I wish to thank my supervisor Professor Josef Kittler for his guidance, help, and supervision of my research work during the past four years. He gave me inspiration and allowed me plenty of freedom to achieve my research goals. His kind criticism and patient proof reading has led to the current shape of the thesis.

I would like to thank Dr. Miroslaw Bober for his kind help and for providing me with the Robust Hough Transform (RHT) based motion estimation algorithm. Much of the early research work involved a close collaboration with him, in particular in exploiting the ability of the RHT motion estimation algorithm in video coding.

I also want to thank my colleagues in the CVSSP group for discussion and help. In particular, thanks to my office-mates Dr. William Christmas and Dr. Adrian Hilton for their help in English; to Dr. Radek Marik for introducing me to the world of C++ programming; to Dr. Nasser Khalili for interesting discussions.

I would like to thank my wife Xiaolin and my daughter Shiran for their support and understanding. They have given me the courage, energy and time to carry out my research.

My tuition fees were paid by a CVCP's ORS award and a University studentship. My subsistence was partially covered by the University studentship and the European Commission through the ACTS SCALAR project. All financial support is gratefully acknowledged.

The equipment and research funding were provided by the following projects:
European Union ACTS-SCALAR and BPSRC Grant GRK/42776
Some of the software used in my research was provided by my colleagues. Thanks are due to Dr. Miroslaw Bober for the Robust Hough Transform based motion estimation and segmentation algorithm, to Dr. Ming Xue for his Wavelet algorithm, and to Mr. Leszek Cieplinski for the ZTVQ and ZTW software.
Contents

1 Introduction
  1.1 The Problem of Digital Obesity ......................................................... 1
  1.2 Possible Solutions ........................................................................... 2
  1.3 Coding Standards ........................................................................... 3
    1.3.1 MPEG-1 ................................................................................... 3
    1.3.2 MPEG-2 ................................................................................... 4
    1.3.3 MPEG-3 ................................................................................... 4
    1.3.4 H.261 ...................................................................................... 5
    1.3.5 H.263 ...................................................................................... 5
    1.3.6 MPEG-4 ................................................................................... 6
    1.3.7 Discussion ................................................................................ 7
  1.4 Thesis Overview .............................................................................. 9
  1.5 Scientific Contributions .................................................................. 11
  1.6 Conclusion .................................................................................... 12

2 Compression Tools for Waveform Based Compression .................... 13
  2.1 Spatial Compression Tools .............................................................. 13
    2.1.1 Transform Coding .................................................................... 13
      2.1.1.1 Discrete Cosine Transform (DCT) ..................................... 14
      2.1.1.2 Wavelet and Sub-band Coding ............................................ 14
    2.1.2 Vector Quantisation .................................................................. 15
    2.1.3 Fractal Compression .................................................................. 19
  2.2 Temporal Compression Tools .......................................................... 20
    2.2.1 Linear Prediction ...................................................................... 20
    2.2.2 Motion Compensated Prediction ............................................... 21
      2.2.2.1 Motion Models .................................................................. 21
      2.2.2.2 Estimation of Motion Parameters ...................................... 23
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.2.3 Motion Compensation</td>
<td>24</td>
</tr>
<tr>
<td>2.3 Lossless Compression Tools</td>
<td>27</td>
</tr>
<tr>
<td>2.3.1 Huffman Coding</td>
<td>28</td>
</tr>
<tr>
<td>2.3.2 Arithmetic Coding</td>
<td>28</td>
</tr>
<tr>
<td>2.4 Discussion</td>
<td>29</td>
</tr>
<tr>
<td>2.5 Conclusions</td>
<td>34</td>
</tr>
<tr>
<td>3 Multiple Layer Affine Motion Compensated Codec</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>35</td>
</tr>
<tr>
<td>3.2 Motion Model and Coding Efficiency</td>
<td>36</td>
</tr>
<tr>
<td>3.3 Multiple Layer Codec Structure</td>
<td>37</td>
</tr>
<tr>
<td>3.3.1 Global Motion Estimation and Compensation</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2 Background/Foreground Segmentation</td>
<td>40</td>
</tr>
<tr>
<td>3.3.3 Background Memory</td>
<td>42</td>
</tr>
<tr>
<td>3.3.4 Extended Quadtree Decomposition</td>
<td>44</td>
</tr>
<tr>
<td>3.3.5 Motion Boundary Coding</td>
<td>45</td>
</tr>
<tr>
<td>3.3.5.1 Straight Line Approximation</td>
<td>48</td>
</tr>
<tr>
<td>3.3.5.2 Fixed Pattern Partition</td>
<td>49</td>
</tr>
<tr>
<td>3.3.6 Rate Control</td>
<td>50</td>
</tr>
<tr>
<td>3.3.6.1 Intra frame bit generation model</td>
<td>52</td>
</tr>
<tr>
<td>3.3.6.2 Intra frame buffer regulation</td>
<td>55</td>
</tr>
<tr>
<td>3.3.6.3 MACC rate control algorithm</td>
<td>56</td>
</tr>
<tr>
<td>3.3.7 Deblur Intra Coding</td>
<td>61</td>
</tr>
<tr>
<td>3.4 Experimental Results</td>
<td>62</td>
</tr>
<tr>
<td>3.5 Conclusion</td>
<td>67</td>
</tr>
<tr>
<td>4 Reconfigurable Multiple Tool Video Codec</td>
<td>69</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>69</td>
</tr>
<tr>
<td>4.2 RMT Codec: A System Description</td>
<td>71</td>
</tr>
<tr>
<td>4.2.1 General structure</td>
<td>71</td>
</tr>
<tr>
<td>4.2.2 Bitstream syntax</td>
<td>73</td>
</tr>
<tr>
<td>4.2.2.1 Picture layer</td>
<td>73</td>
</tr>
<tr>
<td>4.2.2.2 RMT header layer</td>
<td>73</td>
</tr>
<tr>
<td>4.2.2.3 Codec layer</td>
<td>76</td>
</tr>
<tr>
<td>4.3 Communication Unit</td>
<td>76</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 User Interface</td>
<td>76</td>
</tr>
<tr>
<td>4.3.2 Image I/O Interface</td>
<td>76</td>
</tr>
<tr>
<td>4.3.3 Data I/O Interface</td>
<td>76</td>
</tr>
<tr>
<td>4.4 Tool Configuration Unit</td>
<td>77</td>
</tr>
<tr>
<td>4.4.1 Problem Statement</td>
<td>77</td>
</tr>
<tr>
<td>4.4.2 Optimal tool selection</td>
<td>78</td>
</tr>
<tr>
<td>4.4.3 Human Assistance and Heuristics</td>
<td>81</td>
</tr>
<tr>
<td>4.5 Tool Archiving Unit</td>
<td>82</td>
</tr>
<tr>
<td>4.6 Summary</td>
<td>83</td>
</tr>
<tr>
<td>5 The Design of a Prototype RMT Codec</td>
<td>84</td>
</tr>
<tr>
<td>5.1 System Description</td>
<td>84</td>
</tr>
<tr>
<td>5.2 Predefined Codec Structures</td>
<td>84</td>
</tr>
<tr>
<td>5.2.1 H.261</td>
<td>85</td>
</tr>
<tr>
<td>5.2.1.1 Application Range</td>
<td>85</td>
</tr>
<tr>
<td>5.2.1.2 Codec Structure</td>
<td>86</td>
</tr>
<tr>
<td>5.2.1.3 Tool configuration</td>
<td>88</td>
</tr>
<tr>
<td>5.2.2 H.263</td>
<td>89</td>
</tr>
<tr>
<td>5.2.2.1 Application Range</td>
<td>89</td>
</tr>
<tr>
<td>5.2.2.2 Codec Structure</td>
<td>90</td>
</tr>
<tr>
<td>5.2.2.3 Tool configuration</td>
<td>93</td>
</tr>
<tr>
<td>5.2.3 Multiple-layer Affine Compensated Codec (MACC)</td>
<td>93</td>
</tr>
<tr>
<td>5.2.3.1 Application Range</td>
<td>94</td>
</tr>
<tr>
<td>5.2.3.2 Codec Structure</td>
<td>94</td>
</tr>
<tr>
<td>5.2.3.3 Tool configuration</td>
<td>94</td>
</tr>
<tr>
<td>5.3 RMT Codec Structures</td>
<td>95</td>
</tr>
<tr>
<td>5.3.1 Application Range</td>
<td>95</td>
</tr>
<tr>
<td>5.3.2 Codec Structure</td>
<td>96</td>
</tr>
<tr>
<td>5.3.3 Tool configuration</td>
<td>100</td>
</tr>
<tr>
<td>5.4 Tool Development</td>
<td>100</td>
</tr>
<tr>
<td>5.4.1 Spatial Compression Tools</td>
<td>101</td>
</tr>
<tr>
<td>5.4.1.1 DCT</td>
<td>101</td>
</tr>
<tr>
<td>5.4.1.2 Zero Tree Vector Quantisation</td>
<td>101</td>
</tr>
<tr>
<td>5.4.1.3 Wavelet/Subband</td>
<td>101</td>
</tr>
<tr>
<td>5.4.2 Motion Estimation Tools</td>
<td>102</td>
</tr>
</tbody>
</table>
CONTENTS

5.4.2.1 Block Matching Algorithm .......................................................... 102
5.4.2.2 Robust Hough Transform ............................................................. 102
5.4.2.3 Multiresolution Block Matching Algorithm .................................. 102
5.4.3 Motion Compensation Tool Box ...................................................... 105
5.4.3.1 Block based motion compensation ............................................. 105
5.4.3.2 Overlapped block based motion compensation ........................ 105
5.4.4 Lossless Compression Tool Box ...................................................... 105
5.4.4.1 Huffman Coding ............................................................................. 105
5.4.4.2 Arithmetic Coding .......................................................................... 105
5.4.5 Rate Control Tools ................................................................................ 106
5.4.5.1 TMN-5 Rate Control (TMN5) .................................................... 106
5.4.5.2 MPEG-4 Anchor Rate Control (OFFLINE) ............................. 107
5.4.5.3 MACC Rate Control (MRCTL) ................................................ 107
5.5 Experimental Results ................................................................................ 107
5.5.1 Pre-defined codec structure ............................................................. 108
5.5.2 Baseline codec .................................................................................. 109
5.5.2.1 Spatial compression tools ............................................................. 109
5.5.2.2 Motion estimation and motion compensation tools ................. 109
5.5.2.3 Variable length coding tools ...................................................... 111
5.6 Discussion ............................................................................................... 111

6 Knowledge Based Image Sequence Compression ........................................ 114
6.1 Introduction .......................................................................................... 114
6.2 The structure of the knowledge base .................................................... 115
6.2.1 Coding performance database ....................................................... 116
6.2.1.1 Sequence classification .............................................................. 116
6.2.1.2 Performance data and tool configuration data records ............. 117
6.2.2 General knowledge database .......................................................... 118
6.3 Construction of the coding performance database ............................ 119
6.3.1 Detection of texture details ............................................................. 119
6.3.2 Description of motion activity ....................................................... 124
6.3.3 Detection of the presence of a global motion ................................. 126
6.4 Construction of general knowledge database ..................................... 129
6.5 Knowledge assisted tool selection ....................................................... 131
6.6 Experimental results ............................................................................ 132
List of Figures

1.1 Positioning of luminance and chrominance samples .................................................. 8
1.2 Elements of Macroblock ........................................................................................ 8
1.3 Hybrid Codec Structure ........................................................................................ 9

2.1 Warping Prediction ......................................................................................................27
2.2 Rate-Distortion curves of spatial compression tools for "Barb" ..........................31
2.3 Rate-Distortion curves of spatial compression tools for "Boat" ...........................32
2.4 Rate-Distortion curves of spatial compression tools for "Lena" ............................32
2.5 Rate-distortion characteristics for sequence "Container Ship" .........................34

3.1 Quadratic and Robust Redescending Kernels .................................................. 39
3.2 Foreground/Background segmentation ...................................................................41
3.3 Heuristic threshold map ........................................................................................42
3.4 Comparison of heuristic thresholding ....................................................................42
3.5 Object motion and the background memory .........................................................43
3.6 Extended Quadtree Decomposition .........................................................................45
3.7 The relation between the block $B_e$ and its neighbouring blocks ......................47
3.8 Examples of pseudo-Huffman code ......................................................................... 48
3.9 Line approximation of motion boundary .................................................................. 49
3.10 Segmentation around the head of Salesman ..................................................... 50
3.11 Basic Partition Patterns .........................................................................................50
3.12 Derived Partition Patterns ......................................................................................51
3.13 Average results for tested sequences (a) PSNR (b) Bit rate ............................63
3.14 Comparative results for the "Akiyo" sequence (a) PSNR (b) Bit rate ...............64

4.1 Basic Structure of the RMT Codec ........................................................................ 72
4.2 Syntax diagram for RMT codec ...............................................................................73

5.1 Source coder of H.261 ......................................................................................... 86
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>GOB arrangement of H.261</td>
<td>87</td>
</tr>
<tr>
<td>5.3</td>
<td>Bit stream syntax of RMT H.261 codec</td>
<td>88</td>
</tr>
<tr>
<td>5.4</td>
<td>Source coder of H.263</td>
<td>90</td>
</tr>
<tr>
<td>5.5</td>
<td>GOB arrangement of H.263 in a CIF picture</td>
<td>92</td>
</tr>
<tr>
<td>5.6</td>
<td>Bitstream Syntax of Baseline codec</td>
<td>97</td>
</tr>
<tr>
<td>5.7</td>
<td>Structure of picture layer</td>
<td>98</td>
</tr>
<tr>
<td>5.8</td>
<td>Structure of GOB layer</td>
<td>99</td>
</tr>
<tr>
<td>5.9</td>
<td>Multi-resolution pyramid</td>
<td>103</td>
</tr>
<tr>
<td>5.10</td>
<td>Test results of RMT H.263 and TMN-2.0 H.263</td>
<td>108</td>
</tr>
<tr>
<td>5.11</td>
<td>PSNR comparison of different motion algorithms</td>
<td>110</td>
</tr>
<tr>
<td>5.12</td>
<td>Bit rate comparison of different motion algorithms</td>
<td>111</td>
</tr>
<tr>
<td>5.13</td>
<td>Computational time comparison of different motion algorithms</td>
<td>112</td>
</tr>
<tr>
<td>6.1</td>
<td>Simple knowledge based image sequence codec</td>
<td>115</td>
</tr>
<tr>
<td>6.2</td>
<td>Knowledge based decision making engine</td>
<td>116</td>
</tr>
<tr>
<td>6.3</td>
<td>Statistical Features and Bit Rate</td>
<td>120</td>
</tr>
<tr>
<td>6.4</td>
<td>Texture Detail Measurement and Bit Rate</td>
<td>123</td>
</tr>
<tr>
<td>6.5</td>
<td>Questionnaire form</td>
<td>132</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Digital colour TV broadcasting signal ................................................................. 1
1.2 Bit rate of digital image sequences ................................................................. 2

2.1 Weighting values for prediction with motion vectors of the current luminance block .................................................................................................. 25
2.2 Weighting values for prediction with motion vectors of the luminance blocks at the top or bottom of the current luminance block ........................................... 26
2.3 Weighting values for prediction with motion vectors of the luminance blocks to the left or right of the current luminance block ........................................... 26
2.4 Coding results of BMA and RHT based motion estimation .................................. 33
2.5 Bit rate comparison of Huffman and arithmetic coding ................................. 34

3.1 Number of zero motion blocks .............................................................................. 46
3.2 Variable Length Coding Table ............................................................................. 51
3.3 Prediction constant for CIF image sequences ..................................................... 53
3.4 Prediction constant for QCIF image sequences ................................................... 54
3.5 Average PSNR and bit rate for tested sequence .................................................. 65
3.6 Total number of DCT-P bits and SPCB ............................................................... 66
3.7 Total number of bits spent on motion related parameters and gains from motion prediction ......................................................................................... 67

4.1 VLC table for encoding tool index ..................................................................... 75

5.1 Implemented Tools in RMT Prototype ............................................................... 85
5.2 Number of pixels per line and number of lines for each of the H.263 picture formats ................................................................................................. 89
5.3 Relationship between base level GOBs and enhanced level GOBs ................... 99
5.4 Search number for ±16 search range ................................................................. 104
5.5 Intra Frame Coding Results ............................................................................... 109
LIST OF TABLES

5.6 Bitrate comparison of different VLC tools ........................................................... 112
6.1 Texture Detail Measurements and Bit Rate .......................................................... 121
6.2 NZQAC_{16} and Bit Rate at quantisation step 16 .............................................. 122
6.3 Motion activity measurement and bits required to encode motion related
    information .................................................................................................................. 125
6.4 Texture detail of CIF images ................................................................................ 133
6.5 Motion activities of CIF images ......................................................................... 133
6.6 Sequence classification ......................................................................................... 134
6.7 Global Motion Detection ...................................................................................... 134
6.8 Sequence classification by questionnaire answers .............................................. 135
6.9 INTRA frame coding results using different spatial compression tools .......... 136
6.10 Sequence coding results using different INTRA frame spatial compression
tools ............................................................................................................................... 137
6.11 Sequence coding results using different motion estimation algorithms .......... 138
6.12 Coding results using tool selection ..................................................................... 139
Chapter 1

Introduction

1.1 The Problem of Digital Obesity

With the fast development of telecommunication and entertainment industry in multimedia environment, many new services, such as video-telephone, video-mobile phone, video compact disc (CD), Internet Television (TV), video on demand, etc., require efficient visual signal transmission and storage in digital form. Simply digitising traditional video signal such as analogue TV or using directly the output of digital cameras can only meet the requirements of a very small spectrum of applications as the excessive amount of data created by digital video would quickly saturate the media resources. Here are two examples:

Example 1: Digitisation of standard colour TV broadcasting signal. Suppose that the bandwidth of TV signal is 6.5 megahertz (MHz), using 8 bit analogue to digital (A/D) converter, the minimum bit rate will be 104 megabits-per-second (Mbps) as shown in Table 1.1.

Example 2: Digital image sequences at 25 frame/second using 4:2:0 colour space. The bit rate for the quarter common intermediate format (QCIF), common intermediate format (CIF) and (CCIR) 601 size image sequences are shown in Table 1.2.

As we can see, the bit rates of digitised video signals range from 7.6 to over 120 Mbps. On the other hand, the available channel capacity is about 28.8 kbps in today's Public

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>6.5 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Rate</td>
<td>13 MHz</td>
</tr>
<tr>
<td>A/D Precision</td>
<td>8 bits</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>104 Mbps</td>
</tr>
</tbody>
</table>

Table 1.1: Digital colour TV broadcasting signal
1.2 Possible Solutions

Switching Telephone Network (PSTN) using a modem. Even in an Integrated Service Digital Network (ISDN), the channel capacity for a single telephone is 64 kbps. In other words, even at its lower bound, one needs 264 PSTN or 119 ISDN telephone channels to transmit a single digitised TV video signal. It is obvious that such a high bit rate signal is far beyond any practical use of the channel resources. The solution is to use compression techniques to reduce the bit rate. According to a report in Business Week, Feb. 14, 1994, "The biggest obstacle to the vaunted multimedia revolution is digital obesity. That's the bloat that occurs when pictures, sound and video are converted from their natural analogue form into computer language for manipulation or transmission. ... Compression, a rapidly developing branch of mathematics, is putting digital on a diet. ... Its popularity is rooted in economics: compression lowers the cost of storage and transmission by packing data into a smaller space. Many new electronic products and services simply couldn't exist without it."

In the following section, we will give an overview of the possible solutions to the digital obesity problem for video signals (1.2). A brief review of current international standards for video coding is given in section 1.3. Advantages and disadvantages as well as application ranges of the current video coding standards are also discussed. Finally, an overview of this thesis is given in section 1.4.

### 1.2 Possible Solutions

Video compression technology, one of the most active research areas, provides possible solutions to the digital obesity problem for video signals. Past research from psychophysics tells us that the bandwidth of the human visual system is not greater than 100 bits per second (bps). A statistical analysis of video signal also indicates that there is a very strong correlation in both spatial domain and temporal domain. This means that very large redundancy exists in the video signal. Theoretically, such a signal can be compressed to a very high degree without a significant loss of its quality.

<table>
<thead>
<tr>
<th>Image size</th>
<th>QCIF</th>
<th>CIF</th>
<th>CCIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Luminance Dimension</td>
<td>176x144</td>
<td>352x288</td>
<td>704x576</td>
</tr>
<tr>
<td>Chrominance Dimension</td>
<td>88x72</td>
<td>176x144</td>
<td>352x288</td>
</tr>
<tr>
<td>Components Depth</td>
<td>8 bits</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>Bit Rate</td>
<td>7.603Mbps</td>
<td>30.413 Mbps</td>
<td>121.651 Mbps</td>
</tr>
</tbody>
</table>

Table 1.2: Bit rate of digital image sequences
similarity to achieve compression. As the waveform of the signal is preserved in the compression process, this kind of approaches are called *waveform based compression*. In Chapter 2, we will examine some waveform based compression techniques.

Another possible solution is to build a generation model for the signal and use a set of parameters (which usually require a fewer bits to transmit) to regenerate the signal at the receiver side. This approach is called *model based compression* or *analysis-synthesis compression*. It does not necessarily have to generate exactly the same or very close waveform to the original signal, but the final receiver (human) should be able to interpret the signal correctly without difficulty.

The third one is a mixture of waveform and model-based compression methods. It takes advantage of the high compression ratio offered by model based methods and retains the natural feel of waveform based methods.

In image sequence compression, model based methods are still in a very preliminary stage of development and do not appear usable in any real-time communications. The current efforts still focus on the waveform based approaches. Several international standards have been set up for waveform based approaches to facilitate the information exchange between different hardware/software vendors. In the next section, we will briefly review the current video coding standards and the upcoming MPEG-4.

### 1.3 Coding Standards

The current video coding standards are drafted by the International Standardisation Organisation (ISO) Motion Picture Expert Group (MPEG) and International Telecommunication Union (ITU) (H.261 and H.263). These standards strictly define the bit-stream structure being used by encoder and decoder so that the compressed information between the hardware/software products of different vendors can be interchanged. The standards also define the compression techniques and the codec structure. However, the implementation details are left to developers and manufacturers.

#### 1.3.1 MPEG-1

MPEG-1 was optimised for CD-ROM or applications at about 1.5 Mbps. Video was strictly non-interlaced (i.e. progressive). It starts with a relatively low resolution video sequence (possibly decimated from the original) of about 352 by 240 pixels by 30 frames/s (US standard, different numbers for Europe), but includes the original high quality audio. The images are in colour, but converted to the YUV space, and the two chrominance
channels (U and V) are decimated further to 176 by 120 pixels.

The basic scheme of MPEG-1 is to predict motion from frame to frame in the temporal domain, and then to use discrete cosine transform (DCT) to reduce the redundancy in the spatial domain. The DCT is performed on 8x8 blocks, and the motion prediction is based on 16x16 blocks of pixels of the luminance (Y) channel. The DCT coefficients (of either the actual data, or the prediction errors) are quantised, and as a result many of the coefficients will then end up being zero. The quantisation can change for every macroblock (a macroblock is 16x16 pixels of Y and the corresponding blocks of 8x8 pixels in both U and V). The results of the process, which includes the DCT coefficients, the motion vectors, and the quantisation parameters, is Huffman coded using fixed tables. The DCT coefficients have a special Huffman table that is two-dimensional in which one code specifies a run-length of zeros and the non-zero value that ended the run. Also, the motion vectors and the DC components of DCT are coded by Differential Pulse Code Modulation (DPCM).

1.3.2 MPEG-2

Conceptually MPEG-2 is similar to MPEG-1, but includes extensions to cover a wider range of applications. The primary application targeted during the MPEG-2 definition process was the all-digital transmission of broadcast TV quality video at coded bit rates between 4 and 9 Mbps. However, the MPEG-2 syntax has been found to be efficient for other applications such as those at higher bit rates and sample rates (e.g. High definition television (HDTV)). The most significant enhancement over MPEG-1 is the addition of syntax for efficient coding of interlaced video.

1.3.3 MPEG-3

MPEG-3 was targeted for HDTV applications with sampling dimensions up to 1920 x 1080 x 30 Hz and coded bit rates between 20 and 40 Mbps. It was later discovered that with some (compatible) fine tuning, MPEG-2 and MPEG-1 syntax worked very well for HDTV video. The key was to maintain an optimal balance between the sample rate and coded bit rate.

HDTV is now part of the MPEG-2 High-1440 Level and High Level toolkit. Therefore, the effort on MPEG-3 is now terminated.
1.3: Coding Standards

1.3.4 H.261

H.261 targets video-conference applications at the rate of P×64 kilobits per second (kbps) which is a multiple of the basic Pulse Code Modulation (PCM) digital telephone channel (known as B channel). It uses a macroblock structure to combine the luminance signal and chrominance signal into a single colour space as in MPEG. A fixed quantisation table is shared by all macroblocks for DCT coefficients.

H.261 is intended for teleconferencing applications where motion is naturally more limited. Motion vectors are restricted to a range of +/- 15 pixels. The accuracy is reduced since H.261 motion vectors are restricted to integer-pixel accuracy. Other syntactic differences as compared with MPEG video include: no bidirectional interpolated picture type (B pictures), a different quantisation method.

1.3.5 H.263

H.263 is targeted for low bit rate video communication. The basic configuration reflects the H.261 video codec with enhanced features for more efficient compression of video data. The codec can operate on five standardised picture formats: sub-QCIF (128 x 96), QCIF (176 x 144), CIF (352 x 288), 4CIF (704 x 576) and 16CIF (1408 x 1152).

As in H.261, a hybrid of inter-picture prediction to utilise the temporal redundancy and transform coding of the remaining signal to reduce the spatial redundancy is adopted. Motion is compensated to half-pixel accuracy as opposed to H.261 where full-pixel accuracy and a loop-filter are used. The symbols to be transmitted are encoded using variable length coding. Apart from the basic configuration, four negotiable coding options are included further to improve the coding efficiency. They are:

- **Unrestricted Motion Vector (UMV) mode** In this mode motion vectors are allowed to point outside the picture. The edge pixels are used as prediction for the "not existing" pixels. With this mode a significant gain is achieved if there is movement along the edge of the pictures, especially for the smaller picture formats. Additionally, this mode includes an extension of the motion vector range so that larger motion vectors can be used. This is especially useful in the case of camera movement.

- **Syntax-based Arithmetic Coding mode** Arithmetic coding is used instead of Huffman coding to encode symbols. The Peak Signal to Noise Ratio (PSNR) and reconstructed frames will be the same, but fewer bits will be produced (about 5 to 15% reduction).
• **Advanced Prediction mode** This option means that overlapped block motion compensation (OBMC) is used for the Predicted pictures (P-pictures). Four 8x8 vectors instead of one 16x16 vector are used for some of the macroblocks in the picture, and motion vectors are allowed to point outside the picture as in the UMV mode above. The encoder has to decide which type of vectors to use. Four vectors use more bits, but give better prediction. The use of this mode generally gives a considerable improvement, especially subjectively because OBMC results in reduced blocking artifacts.

• **PB-frame mode** A PB-frame consists of two pictures being coded as one unit. The name PB comes from the name of picture types in MPEG where there are P-picture and Bidirectional-interpolated-pictures (B-picture). Thus a PB-frame consists of one P-picture which is predicted from the last decoded P-picture and one B-picture which is predicted from both the last decoded P-picture and the P-picture currently being decoded. This last picture is called a B-picture, because parts of it may be bidirectionally predicted from the past and future P-pictures. For relatively simple sequences, the frame rate can be doubled with this mode without increasing the bitrate substantially. For sequences with a lot of motion, PB-frames do not work as well as B-pictures in MPEG. This is because there are no separate bidirectional vectors in H.263. The forward vectors for the P-picture are scaled and added to a small delta-vector. The advantage over MPEG is much less overhead for the B-picture part, which is really useful for the low bitrates and relatively simple sequences most often generated by video-phones.

H.263 is now used as the anchor algorithm in MPEG-4 development for very low bitrate video coding. It can deliver a reasonably good image quality at 10kbps for video-conference type sequences.

### 1.3.6 MPEG-4

MPEG-4 aims to establish universal, efficient coding of different forms of audio-visual data, called audio-visual objects. Conceptually, it is quite different from the previous MPEG standards. In fact, MPEG-4 intends to provide a universal codec which not only covers all previous coding standards but also provides a much wider application range. The audio-visual objects can be of natural or synthetic origin. To achieve the goal, two basic elements are defined:
1.3: Coding Standards

- A set of coding tools for audio-visual objects capable of providing support to different functionalities such as object based interactivity and scalability, and error robustness, in addition to efficient compression.

- A syntactic description of coded audio-visual objects, providing a formal method for describing the coded representation of these objects and the methods used to encode them.

The coding tools will be defined in such a way that users can put several standard MPEG-4 tools together to satisfy specific user requirements. The syntactic description will be used to convey to a decoder the choice of tools made by the encoder. It can also be used to describe new algorithms which is an organised collection of tools and download their configuration to the decoding processor for execution.

The bitrate range of MPEG-4 video covers: below 64 kbps (low), 64-384 kbps (intermediate) and 384 kbps - 1.8 Mbps (high). As far as real time communication is concerned, the MPEG-4 video is about to support QSIF/QCIF video at below 24 kbps with quality equivalent to, or better than that achievable with H.263 (same bit rate with all options enabled). This work will require the development of fundamentally new algorithmic techniques. When completed, the MPEG-4 standard will enable a whole spectrum of new applications, including interactive mobile multimedia communication, video-phone, mobile audio-visual communication, multimedia electronic mail, remote sensing, electronic newspapers, interactive multimedia databases, multimedia video-text, games, interactive computer imagery, sign language captioning.

Since the primary target for these applications is bit rate of up to 64 kbps at good quality, it is anticipated that new coding techniques allowing higher compression than traditional techniques may be necessary. Morphology, fractals, vector quantisation, wavelet, variable block size coding, model based coding are all in the offering. MPEG-4 will adopt an open policy towards any new tools and algorithms (Please note that the initial version of MPEG-4 video still uses DCT as its core compression technique with the ability to use wavelet for still image compression).

1.3.7 Discussion

From the above discussion, one can see that different standard coding algorithms are designed for different applications. They all use block based image partition, DCT spatial compression tool and hybrid prediction codec structure. Two important concepts in the coding standard are the definition of macroblock and the hybrid prediction structure. The
former enables efficient coding of colour image sequences and the latter enables efficient reduction of temporal redundancy. The pictures in standard codecs are coded as luminance

![Luminance sample](image)

**Figure 1.1:** Positioning of luminance and chrominance samples

and two colour difference components ($Y$, $C_b$ and $C_r$). The chrominance components are decimated by a factor of two. The position of luminance and chrominance samples are shown in Figure 1.1. A macroblock relates to a $16 \times 16$ pixels area of $Y$ component and the spatially corresponding $8 \times 8$ pixels area of the two chrominance components ($C_b$ and $C_r$). Further, a macroblock consists of four luminance blocks and two spatially corresponding chrominance blocks as shown in Figure 1.2. Each luminance or chrominance block relates to an area of $8 \times 8$ pixels. The temporal redundancy is removed by motion compensated

![Macroblock Elements](image)

**Figure 1.2:** Elements of Macroblock

prediction and the encoding of prediction error. A typical hybrid codec (H.263) is shown in Figure 1.3.

The current state of the art video compression standard for low bit rate communications is H.263. However, as a single track coding standard, one cannot expect it to
fulfill the broad requirement in today's multimedia communications. On the other hand, MPEG-4 as an open, application dependent coding standard, provides more space for researchers to develop new compression methods to plug into the MPEG-4 system.

### 1.4 Thesis Overview

In this thesis, we will mainly focus on very low bit rate video coding techniques for visual communication in multimedia environments. This requirement restricts us to develop a low delay, low complexity codec with sufficient flexibility.

In Chapter 2, widely used compression tools and their performance are reviewed. We examine:

- block based motion compensation, triangle warp motion compensation and overlapped motion compensation for motion compensated prediction tools;
• Huffman coding and arithmetic coding for lossless compression tools;
• DCT, wavelet/subband, vector quantisation and fractal compression for spatial compression tools;
• fix-block size and variable block codec for structural tools.

In Chapter 3, an instance of hybrid structure video codec (MACC) is introduced and its performance is studied. We show that improved motion estimation and the use of complex motion model give improved image quality for all tested sequences. But a bit rate reduction relatively to H.263 is achieved only in those sequences containing large and complex motion.

In Chapter 4, a reconfigurable video codec is proposed, motivated by the observation that different sequences require different coding techniques to achieve optimal coding results. It is also beneficial to have a codec which can be operated in different modes in order to facilitate communication with systems employing different coding standards.

In Chapter 5, the design of a prototype reconfigurable multiple tool (RMT) codec is presented. The focus of this chapter is concentrated on the practical side of the RMT codec. The compatibility with current coding standards such as H.261 and H.263 is taken into account in the design. The tools developed in our research and available in public domain are integrated and tested. The experimental results prove that the RMT codec is capable of delivering better performance for various sequences with the right combination of coding tools.

In Chapter 6, a knowledge base concerning the capability of the various coding tools available in the codec and their performance as a function of image sequence statistics is built and applied to control the reconfigurable video codec. Two knowledge models are proposed. The first one is the knowledge injection model, in which human knowledge compiled by means of a questionnaire is incorporated into the codec. The second one is a self learning model, in which the performance, the tool combination and the statistical properties of each sequence are archived and analysed. The experimental results show that the proposed knowledge model can be used to select a good tool combination for the RMT codec to achieve an above average performance.

In the last chapter, we draw some concluding remarks on the thesis and directions for future research are proposed.
1.5 Scientific Contributions

In the thesis, the following scientific contributions have been made:

- The study of widely used compression techniques revealed that some compression techniques outperform others for certain types of applications and vice versa.
- A multiple layer codec structure has been proposed.
- A probabilistic background/foreground segmentation method with heuristic thresholding which gives satisfactory results for separating the foreground objects from their background has been developed.
- A background memory reference updating scheme used for improving the coding efficiency of uncovered background regions has been devised.
- An extended quadtree decomposition procedure supporting the use of background motion compensation, dual motion inner block segmentation and background memory reference updating has been proposed.
- A pseudo-Huffman code which improves the coding efficiency of adjacent block positions has been developed.
- Two methods for inner block motion boundary coding (straight line approximation and fixed pattern partition) have been devised and investigated.
- A bit rate control method capable of regulating the first INTRA frame coding by using Non-Zero Quantised Alternative-current Coefficients (NZQAC) measurements has been conceived and implemented.
- A framework for a reconfigurable multiple tool video codec has been proposed and designed. The concept of virtual codec and virtual tool is the major innovative contribution behind the development of the reconfigurable multiple tool codec.
- An expandable variable length code has been developed for encoding the tool update data field. The code combines efficiency with flexibility. The expandability of the code guarantees the maximum flexibility of the system.
- A prototype codec realising the idea of reconfigurable multiple tool codec has been implemented to demonstrate its practical potential.
- A knowledge based tool selection system has been studied and developed.
• A questionnaire based knowledge injection mechanism has been developed with a graphical interface to the reconfigurable codec prototype.

1.6 Conclusion

In this chapter, we first explored the digital obesity and possible solutions to reduce such obesity in video communications. We then briefly reviewed existing and upcoming international standards. We have argued that although different standards target different applications, the basic compression techniques used by various standards remain largely unchanged. Finally, we gave an overview of the thesis and a list of scientific contributions presented in the thesis.
Chapter 2

Compression Tools for Waveform Based Compression

The commonly used tools for waveform based video compression can be classified into 3 categories:

- Spatial compression tools: to reduce the spatial redundancy in the original image or prediction error image.
- Temporal compression tools: to reduce the temporal redundancy between video frames.
- Lossless compression tools: to reduce the symbol redundancy when converting symbols previously generated by spatial and/or temporal compression tools into bit streams;

In the following sections, some commonly used techniques and their performance are reviewed.

2.1 Spatial Compression Tools

2.1.1 Transform Coding

One of the most powerful tools to reduce the spatial redundancy is to use a linear transform to change the distribution of the signal so that most of the signal energy is concentrated in a few components. The transform technique alone cannot provide any compression to the input signal. The compression is achieved by discarding or coarsely quantising the less important components. The most widely used transform techniques in video compression are Discrete Cosine Transform and Wavelet Transform.
2.1.1.1 Discrete Cosine Transform (DCT)

For natural images, discrete cosine transform [1] is known to provide a good approximation to the Karhunen - Loeve Transform (KLT) which is optimal in terms of minimum signal representation error. Many image compression methods, including the JPEG, MPEG, and H.261 standards, are based on the discrete cosine transform. DCT used in the standard methods (JPEG, MPEG and H.261) is a separable two-dimensional discrete cosine transform of size 8 by 8. The transform and the inverse transform are defined by the following equations:

\[
F(u, v) = \frac{1}{4} C(u) C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} f(i, j) \cos [(2i + 1)u \pi /16 ] \cos [(2j + 1)v \pi /16 ]
\]

\[
f(i, j) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u) C(v) F(u, v) \cos [(2i + 1)u \pi /16 ] \cos [(2j + 1)v \pi /16 ]
\]

(2.1)

where \( i, j \) are pixel coordinates in the spatial domain;
\( u, v \) are coordinates in the transform domain.

\( C(u) = 1/\sqrt{2}, \) for \( u = 0 \), otherwise \( 1; \)
\( C(v) = 1/\sqrt{2}, \) for \( v = 0 \), otherwise \( 1. \)

DCT can be calculated from discrete Fourier transform[89]. This makes it possible to use an FFT like technique to calculate DCT efficiently. Various fast DCT algorithms have been proposed by Chen et al.[20], Vetterli[91], Loeffler et al.[55], Kamangar and Rao[44], Cho et al.[21], etc. Hung and Meng had compared various fast DCT algorithms in [35]. A more detailed discussion about DCT can be found in [72][13].

2.1.1.2 Wavelet and Sub-band Coding

Wavelets are a mathematical tool for decomposing functions hierarchically. A given signal, \( f(x) \), can be represented as a weighted sum of simple building blocks, called basis functions:

\[
f(x) = \sum_{i} c_{i} \Psi_{i}(x)
\]

(2.2)

where \( \Psi_{i}(x) \) are basis functions and \( c_{i} \) are coefficients, or weights. Since the basis functions are known, the coefficients contain all the information about the signal. Choosing sinusoids as the basis functions yields a Fourier representation in which the coefficients reveal the frequency domain behaviour of the signal. DCT is another example of such a representation.

If we constrain all the basis functions in \( \{ \Psi_{i} \} \) to be scaled and translated versions of the same prototype function \( \Psi \) which satisfies two conditions:
it must be oscillatory (waves),

- its amplitude quickly decays to zero in both the positive and negative directions,

we will have a set of wavelet basis functions. $\psi$ is called mother wavelet. If the scaling is accomplished by multiplying $x$ by some scale factor which is chosen to be a power of 2, yielding $\psi(2^n x)$ where $n$ is an integer, the cascaded octave bandpass filter structure will be obtained. Because $\psi$ has a finite support, it will need to be translated along the time axis in order to cover the entire signal. This translation is accomplished by considering all the integer shifts of $\psi$,

$$\psi(2^n x - k), k \in \mathbb{Z}$$  \hspace{1cm} (2.3)

This will give us a wavelet decomposition of the signal,

$$f(x) = \sum_{n \text{finite}} \sum_{k \text{finite}} c_{nk} \psi_{nk}(x)$$  \hspace{1cm} (2.4)

where

$$\psi_{nk}(x) = 2^{n/2} \psi(2^n x - k)$$  \hspace{1cm} (2.5)

(the multiplication by $2^{n/2}$ is needed to make the bases orthonormal). The coefficients $c_{nk}$ are computed by the wavelet transform, which is the inner product of the signal $f(x)$ with the basis function $\psi_{nk}(x)$.

The use of wavelet theory in image processing can be traced back to the pyramid decomposition[17] of images which operates on discrete image data. In [58], Mallat related wavelet theory to the pyramid decomposition. Vaidyanathan related a multiresolution transform to multirate filtering banks in [88]. The connection between wavelet and subband coding[93] was exposed by Vetterli et al.[90][74]. For video coding, there are two kinds of approaches. The first one uses wavelet decomposition as a spatial domain compressor and motion compensated prediction to reduced the temporal redundancy[8][65]. The other one uses a 3-D transform technique to avoid the difficulty of motion estimation[45][24].

### 2.1.2 Vector Quantisation

A quantiser, denoted by $Q$, maps its input value $x \in I$ onto a finite set of output values $y \in O$

$$x \xrightarrow{Q} y, x \in I, y \in O = \{y_j\}, O \subset I$$  \hspace{1cm} (2.6)
where $I$ is called the input space; $O$ is a subset of $I$, called the output space. To transmit the quantised value, $y$ is mapped on to a symbol $s \in \{s_j\}$ through the codec $C$

$$y \xrightarrow{C} s, y \in O, s \in \{s_j\}$$

(2.7)

where $\{s_j\}$ is a set of symbols which can be optimised for transmission (through Huffman or arithmetic coding). Note that mapping $C$ is invertible, i.e. the equation 2.6 and 2.7 can be rewritten as

$$x \xrightarrow{E} s, x \in I, s \in \{s_j\}$$

(2.8)

$$s \xrightarrow{D} y, s \in \{s_j\}, y \in \{y_j\}$$

(2.9)

where $E$ is called encoder, $D$ is called decoder. If we extend the above equations to a multi-dimensional space, then

$$\vec{x} \xrightarrow{E} s, \vec{x} \in I, s \in \{s_j\}$$

(2.10)

$$s \xrightarrow{D} \vec{y}, s \in \{s_j\}, \vec{y} \in \{\vec{y}_j\}$$

(2.11)

where $\vec{x} = \{x_0, x_1, ..., x_N\}, \vec{y} = \{y_0, y_1, ..., y_N\}$ are vectors in an $N$ dimensional space. Equation 2.10 defines the process of vector quantisation (VQ) which maps input vector $\vec{x}$ on to a symbol $s$ by finding the minimum distance codeword in a codebook which is defined by 2.11.

The advantage of using vector quantisation is that the dependence between individual values can be taken into account to obtain a compact representation of the input signal. A typical example is a colour map: a colour picture can be represented by a 2D array of triplets (RGB values). In most pictures those triplets do not cover the whole RGB space but tend to concentrate in certain areas. For example, the picture of a forest will typically have a lot of green. One can select a relatively small subset (typically 256 elements) of representative colours, i.e. RGB triplets, and then approximate each triplet by the representative of that small set. In the case of 256 colours one can use 1 byte instead of 3 for each pixel.

For a scalar quantiser, if the probability density function (pdf) $p_s(x)$ of the input is known, an optimal quantiser which minimises the mean-square error can be designed by the Lloyd method[54]

$$t_i = \frac{l_i + l_{i+1}}{2}, i = 1, ..., M - 1,$$

(2.12)
and

\[ l_i = \frac{\int_{t_{i-1}}^{t_i} xp_s(x)dx}{\int_{t_{i-1}}^{t_i} p_s(x)dx}, \quad i = 1, \ldots, M. \]  

(2.13)

where \( \{t_1, \ldots, t_{M-1}\} \) are decision levels, \( \{l_1, \ldots, l_M\} \) are quantisation levels. Equation 2.12 and 2.13 can be iteratively solved for quantisation levels and decision levels for any probability density function \( p_s(x) \).

Linde, Buzo and Gray extended Lloyd's method to design a vector quantiser [53]. Their method, known as the LBG algorithm, designs an optimal vector quantiser for a training sequence of unknown distribution using the mean-square error distortion measure. The training is achieved by the following steps:

Step 1
- Set iteration counter \( m \leftarrow 0 \); distortion threshold \( \epsilon > 0 \)
- Choose a set of initial codebook vectors \( \vec{l}_i(m), i = 0, \ldots, L - 1 \) for a given training sequence \( \vec{t}_n, n = 1, \ldots, N \).

Step 2
- Given codebook \( \vec{l}_i(m), i = 0, \ldots, L - 1 \), find the minimum distortion partition \( C_i(m), i = 0, \ldots, L - 1 \) of the training sequence: \( \vec{t}_n \in C_i(m) \), if \( d(\vec{t}_n, \vec{l}_i(m)) \leq d(\vec{t}_n, \vec{l}_j(m)) \) for \( j = 0, \ldots, L - 1, j \neq i \).
- Calculate the average distortion \( D(m) \) for the given codebook \( \vec{l}_i(m) \) and partition \( C_i(m), i = 0, \ldots, L - 1 \)

\[ D(m) = \frac{1}{N} \sum_{i=1}^{L} \frac{M_i}{N} D_i(M) \]

where \( M_i \) is the number of training vectors in \( C_i(m) \), \( D_i(m) \) is the average distortion in cell \( C_i(m) \) given by

\[ D_i(m) = \frac{1}{M_i} \sum_{\vec{t}_{ij} \in C_i(m)} d(\vec{t}_{ij}, \vec{l}_i(m)) \]

Step 3
- if the stop criterion

\[ \left| \frac{D(m - 1) - D(m)}{D(m - 1)} \right| < \epsilon \]
is satisfied, stop the iteration with $\bar{u}_i(m), i = 0, ..., L - 1$ as the final codebook; otherwise continue.

Step 4

• Set $m \leftarrow m + 1$;

• Find the optimal codebook for partition $C_i(m), i = 0, ..., L - 1$ by computing the centroids of the training vectors in each partition $C_i(m), i = 1, ..., L - 1$ as the new codebook vector $\bar{u}_i(m)$.

$$\bar{u}_i(m) = \frac{1}{M_i} \sum_{j \in C_i(m)} \bar{t}_j,$$

where $M_i$ is the number of training vectors in $C_i(m)$.

• Goto step 2.

Vector quantisation has gained its popularity in image coding community since it was introduced in 1980. It was used both in still image compression and image sequence compression. The early works include mean/shape VQ proposed by Baker and Gray[6], classified VQ proposed by Ramamurthi et al.[70][71], codebook replenishment VQ proposed by Goldberg and Sun[84][28], finite state VQ proposed by Aravind et al.[5] and Baker et al.[7], etc. Because VQ is a general form quantiser, it can be used in the transform domain as well. The vector quantisation had been used with DCT[79], sub-band coding[92], wavelet transform[27], etc.

Because the efficiency of the VQ increases when the dimensionality of the vectors increases, the problem of high dimensional vector quantisation was studied by several researchers. The major difficulty of using high dimensional VQ is that the computational and storage requirement increase exponentially with the product of the number of dimensions and bit rate. Residual VQ(RVQ) or multi-stage VQ proposed by Juang and Gray [43] appears to be the only solution to the memory problem. However, the original RVQ proposed by Juang can only be used in two stages due to its codebook design method. Kossentini et al. proposed an improved codebook design method which uses multiple search and global optimisation to generate an RVQ codebook[49][48]. Their results in very low bit rate image coding using a variable rate codebook and non-uniform codebook size are comparable with other approaches.
2.1.3 Fractal Compression

The rediscovery of fractal geometry is usually traced back to the IBM mathematician, Benoit B. Mandelbrot and his seminal book *The Fractal Geometry of Nature*. The book put forth a powerful thesis: traditional geometry with its straight lines and smooth surfaces does not resemble the geometry of trees and clouds and mountains. Fractal geometry, with its convoluted coastlines and detail ad infinitum, does.

The use of fractals in image compression was first proposed by Barnsley. The mathematical theories of *Iterated Function System (IFS)* and *Recurrent Iterated Function Systems*, along with the important *College Theorem*, constitute the foundations of fractal image compression. However, these theories alone do not provide any constructive procedure for the “encoding” of a grey tone image in an automatic way. Going from a given image to an Iterated Function System (IFS) that can generate the original (or at least closely resemble it), known as the inverse problem, remains unsolved (although, with the so called *Graduate Student Algorithm* it might be possible to find an IFS for a given image with the assistance of a man).

Jacquin was the first person to propose a *Fractal Block Coding system* that enables automated encoding of a natural grey tone image. The algorithm is based on a general theory of iterated contractive transformation in metric spaces of images introduced in [10][9][11]. Since then, a number of extensions and improvements to Jacquin's algorithm have been proposed by various researchers[38][62][61][66][85]. These extensions address the following issues:

- influence of the type of image partition, pool of block transformations, and optimisation of the parameters defining these transformations.
- reduction of the computational complexity of the encoding process.
- relationship between fractal block coding and other block based image coding technique such as DCT, wavelet transform and vector quantisation, etc.
- applicability of the theory to the three-dimensional case for fractal block coding of video.

Fractal image compression appears to be a promising new technology. It gives a new way to capture and exploit the redundancy in images, which are not used by more traditional image coding techniques. Fractal compression has many similarities to vector quantisation - both are lossy, take large amount of computation in encoder and very low complexity in decoder. However, there are notable difference between the two:
(a). In VQ the range blocks and domain blocks are of the same size; in an IFS the domain blocks are always larger.

(b). In VQ the domain blocks are copied directly; in an IFS each domain block undergoes a luminance scaling and offset.

(c). In VQ the codebook is stored apart from the image being coded; in an IFS the codebook is not explicitly stored. It is comprised of portions of the attractor as it emerges during iteration. For that reason it is called a "virtual codebook." It does not exist independently of the affine transformations that define an IFS.

(d). In VQ the codebook is shared among many images; in an IFS the virtual codebook is specific to each image.

The compression ratios of fractal coding typically range from 4:1 to 100:1. For colour images, higher compression ratios can be obtained as in other image compression methods. As compared with standard methods, for the same compression ratio (especially in very low bit rate coding), the fractal compression presents a more acceptable image quality. It is also on the hot list for the MPEG-4 coding standard.

2.2 Temporal Compression Tools

In an image sequence, most of the redundancy comes from the temporal domain. To reduce such a redundancy, one can treat the time axis as the third dimension and use the above spatial compression tools to compress the sequence. This is so called 3-D coding or volume based compression [2, 97]. However, to take the full advantage of 3-D coding, several frames are required to form the third dimension and this inevitably increases the delay of the codec. Therefore, this approach is not suitable for use in an interactive communication environment.

A more popular approach is to adopt a hybrid structure codec which uses a prediction technique to reduce the temporal redundancy. In the following discussion, we will only deal with prediction based compression tools in temporal domain.

2.2.1 Linear Prediction

A widely used technique in image coding (including still image and image sequence) is the differential pulse-code modulation (DPCM). The simplest DPCM system is the one with only a unit delay in the predictor block which is used in standard algorithms. The output of such a system can be expressed by:
2.2: Temporal Compression Tools

\[ e = s(n) - s'(n - 1) \] (2.14)

where \( s(n) \) is the input signal, \( s'(n - 1) \) is quantised input signal with unit delay and \( e \) is the differential signal to be quantised. Because \( s(n) \) and \( s'(n - 1) \) are highly correlated for image signals, the variance of \( e \) is far less than that of \( s \). That means fewer bits are needed to code \( e \) than to code \( s \) if the error expectation is the same.

2.2.2 Motion Compensated Prediction

For image sequence, very strong correlations exist between adjacent frames. Such strong correlations can be used to reduce the redundancy in the image sequences. Motion compensation has become a very important technique in video coding. It allows one to improve the accuracy of prediction in the temporal domain so that more efficient compression can be achieved while maintaining good picture quality.

One of the most difficult tasks in motion compensated prediction is to extract motion parameters from the projection of a 3-D moving object on the 2-D image plane. The difficulty is two fold. Firstly, for a given image there is no prior knowledge of the motion pattern, i.e., motion model unknown. Secondly, for a given motion model, to obtain the model parameter involves a large amount of computation and the results may be affected by the presence of noise, multiple motions in the region of interest, etc. In the following discussion, we will briefly review the motion model used in image sequence coding and some of the commonly used parameter estimation methods.

2.2.2.1 Motion Models

For a given region, suppose that only a single rigid moving object exists and the motion vector field can be described by a continuous function \( v_x(x, y) \) for the \( x \) component and \( v_y(x, y) \) for the \( y \) component. Using the Taylor expansion, they can be expressed as

\[
v_x(x, y) = v_x(x_0, y_0) + \frac{1}{1!}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]v_x(x_0, y_0) + \frac{1}{2!}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]^2v_x(x_0, y_0) + ... \]

\[
v_y(x, y) = v_y(x_0, y_0) + \frac{1}{1!}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]v_y(x_0, y_0) + \frac{1}{2!}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]^2v_y(x_0, y_0) + ... \]
If $(x - x_0)$ and $(y - y_0)$ are close to zero (this means the region is small enough) or the motion field changes slowly (small derivatives), then all the derivative terms can be neglected and the equation becomes

$$v_x(x, y) = v_x(x_0, y_0) = t_x$$
$$v_y(x, y) = v_y(x_0, y_0) = t_y$$

$(t_x, t_y)$ is the displacement vector for translational motion. Equation 2.16 is the widely used translational motion model. It can be seen that this motion model only works on a small region with a smooth motion field.

If the object moves with a constant acceleration, none of the first order derivatives is zero but the higher order derivatives can be omitted, and the equation 2.15 becomes

$$v_x(x, y) = v_x(x_0, y_0) + \frac{1}{11}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]v_x(x_0, y_0)$$
$$= [v_x(x_0, y_0) - x_0 \frac{\partial v_x(x_0, y_0)}{\partial x} - y_0 \frac{\partial v_x(x_0, y_0)}{\partial y}] + x \frac{\partial v_x(x_0, y_0)}{\partial x} + y \frac{\partial v_x(x_0, y_0)}{\partial y}$$
$$= a_0 + a_1 x + a_2 y$$

$$v_y(x, y) = v_y(x_0, y_0) + \frac{1}{11}[(x - x_0) \frac{\partial}{\partial x} + (y - y_0) \frac{\partial}{\partial y}]v_y(x_0, y_0)$$
$$= [v_y(x_0, y_0) - x_0 \frac{\partial v_y(x_0, y_0)}{\partial x} - y_0 \frac{\partial v_y(x_0, y_0)}{\partial y}] + x \frac{\partial v_y(x_0, y_0)}{\partial x} + y \frac{\partial v_y(x_0, y_0)}{\partial y}$$
$$= b_0 + b_1 x + b_2 y$$

This is so called affine motion model. Thus an affine motion model can cope with any motion with constant acceleration, which is typical of the motion encountered in real scene sequences.

A more sophisticate motion model can be obtained by retaining up to $n_{th}$ order derivatives to describe the motion field. This is so called polynomial motion model which can be expressed as

$$v_x(x, y) = \sum_{i,k=0}^{n} a_{i,k} x^i y^k$$
$$v_y(x, y) = \sum_{i,k=0}^{n} b_{i,k} x^i y^k$$

Because of the computational complexity, only the use of a second order polynomial motion model has been reported in video coding [19].
2.2.2.2 Estimation of Motion Parameters

In 1979, Netravali and Robbins [3] published a first recursive motion estimation algorithm to improve the estimation accuracy and to increase the measuring range of the displacement. In recursive motion estimation algorithms, the current local estimate of the motion vector field $\hat{V}_i$ is used to produce a new improved estimate $\hat{V}_{i+1}$ according to

$$\hat{V}_{i+1} = \hat{V}_i + U_i$$

(2.19)

where $U_i$ is the so-called update term at iteration $i$. The iteration can be computed either for a single pixel at consecutive pixels along a scanning line, from line to line, or from frame to frame. Accordingly, these techniques are called pel-recursive estimation algorithms with horizontal, vertical, or temporal recursion. Given $\hat{V}_i = [\hat{d}_{x_i}, \hat{d}_{y_i}]^T$, a function of the displaced frame difference DFD

$$DFD(x, y, \hat{V}_i) = s_k(x, y) - s_{k-1}(x - \hat{d}_{x_i}, y - \hat{d}_{y_i})$$

(2.20)

can be used as a criterion for calculating the update $\hat{V}_{i+1}$. In [3], the authors propose to minimise the square value of the DFD recursively using the gradient method as

$$\hat{V}_{i+1} = \hat{V}_i - \frac{1}{2} \epsilon \nabla_{\hat{V}_i} [DFD(x, y, \hat{V}_i)]^2$$

(2.21)

where $\nabla_{\hat{V}_i}$ is the gradient operator with respect to $\hat{V}_i$ and $\epsilon$ is a positive constant. In this method, the convergence speed and accuracy depends on the value of $\epsilon$. A large $\epsilon$ will yield a quick but less accurate convergence whereas a small $\epsilon$ leads to a more accurate but slow convergence. Apart from $\epsilon$, the initial guess $\hat{V}_0$ also affects the convergence. To overcome the speed and accuracy dilemma, improvements over the basic pel-recursive algorithm have been suggested in [30, 18] to speed up the convergence while maintaining a good accuracy.

The pel-recursive algorithm is based on the assumption that DFD is a single peak continuous function of displacement vector $\hat{V}$. This may not be the case in real scene sequences and the algorithms may not converge to the correct correlation peak or even may not converge at all. In the early 80's, a non-recursive estimation algorithm called Block Matching Algorithm was proposed [42, 86]. The basic idea of block matching is to shift the position of the block in a search area so that a predefined error criterion can be minimised. The most frequently used error criterion to measure the merit of displacement by vector $(i, j)$ are the mean square error MSE

$$MSE(i, j) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} [s_k(m, n) - s_{k-1}(m + i, n + j)]^2$$

(2.22)
and mean absolute frame difference MAD

\[ \text{MAD}(i, j) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} |s_k(m, n) - s_{k-1}(m + i, n + j)| \] (2.23)

From experiments it has been found that the matching criterion has no significant influence on the search [83]. Therefore, MAD criterion is more widely used since it has a lower computational load and is more robust to noise.

The computational load of the block matching is proportional to the area \( MN \) and the square of the possible search range. Several methods to reduce the computational complexity have been proposed [42, 86, 95, 83, 23, 47] but inevitably at the price of a sub-optimal performance. Until now, it is still the most frequently used method in video coding.

The use of block matching implies that all pixels within the block undergo coherent motion. When multiple motion exists, the estimated motion vectors will be biased. When the block size increases, the chances of having multiple moving objects inside the block also increase. Using a small block size may overcome this problem but the possibility of converging to a false peak caused by the turbulence of the luminance is increased. To overcome this problem, statistical based estimation algorithms have been proposed [14, 15, 31]. In this kind of algorithms, the motion parameters are firstly estimated for small regions (even single pixels) inside the region of interest. Then the histogram of the estimated motion vectors is calculated and the peak is chosen to be the estimated motion vector for the region. The pixels which do not move coherently with the majority of the pixels are marked as outliers.

**2.2.2.3 Motion Compensation**

The basic motion compensation scheme compensates the same region as that used for estimating the motion parameters. It is simple and efficient. However, the major disadvantage of this scheme is that it produces blocking artifacts caused by different motion vectors among adjacent blocks. Two improvements over the basic motion compensation scheme is to use overlapped block motion prediction [37] and triangular mesh based warping prediction [56].

**Overlapped motion compensation for luminance** Each pixel in an 8*8 luminance prediction block is a weighted sum of three prediction values, divided by the total weight sum of 8 (with rounding). In order to obtain the three prediction values, three motion
2.2: Temporal Compression Tools

Vectors are used: the motion vector of the current luminance block, and two out of the four "remote" vectors:

- the motion vector of the block at the left or right side of the current luminance block;
- the motion vector of the block above or below the current luminance block.

Remote motion vectors from other group of blocks (GOB) are used in the same way as remote motion vectors inside the current GOB.

For each pixel, the remote motion vectors of the blocks at the two nearest block borders are used. This means that for the upper half of the block the motion vector corresponding to the block above the current block is used, while for the lower half of the block the motion vector corresponding to the block below the current block is used (see Table 2.2). Similarly, for the left half of the block the motion vector corresponding to the block at the left side of the current block is used, while for the right half of the block the motion vector corresponding to the block at the right side of the current block is used (see Table 2.3).

If one of the surrounding blocks was not coded or was coded in the INTRA mode, the corresponding remote motion vector is set to zero. However, in the Prediction and Bidirectional-interpolation (PB) frame mode a candidate motion vector predictor is not set to zero if the corresponding macroblock was coded in the INTRA mode. If the current block is at the border of the picture and therefore a surrounding block is not present, the corresponding remote motion vector is replaced by the current motion vector. In addition, if the current block is at the bottom of the macroblock (for block number 3 or 4, see Figure 1.2), the remote motion vector corresponding to an 8*8 luminance block in the macroblock below the current macroblock is replaced by the motion vector of the current block.

The weighting values for the prediction are given in Table 2.1, Table 2.2 and Table 2.3.

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2.1: Weighting values for prediction with motion vectors of the current luminance block
2.2: Temporal Compression Tools

Table 2.2: Weighting values for prediction with motion vectors of the luminance blocks at the top or bottom of the current luminance block

Table 2.3: Weighting values for prediction with motion vectors of the luminance blocks to the left or right of the current luminance block

Triangular mesh based warping prediction Warping prediction is a kind of geometric transform for motion compensated interframe prediction (MC). Input images are divided into triangular patches, and each patch is predicted with a warping transform. The prediction errors are encoded via an orthogonal transform. Because no blocking artifacts appear in the prediction image, high compression performance can be achieved without serious degradation of subjective video quality.

In contrast with traditional block based prediction, warping prediction removes blocking artifacts [1,2]. Figure 2.1 illustrates how warping prediction works. Grid points are located in the current picture, and motion vectors are encoded only on the grid points. We refer to the area enclosed by the lines connecting the grid points as a patch. MC prediction for each pixel in a patch is implemented with an affine transform defined as

\[
\begin{pmatrix}
x' \\
y'
\end{pmatrix} = \begin{pmatrix}
a_1 & a_2 \\
a_4 & a_5
\end{pmatrix} \begin{pmatrix}
x \\
y
\end{pmatrix} + \begin{pmatrix}
a_3 \\
a_6
\end{pmatrix}
\]  

(2.24)
Here, \((x', y')\) denotes a pixel position in a current picture, and \((x'y')\) is the MC predicted position of the pixel in a reference picture. The values of affine parameters \(a_1\) to \(a_6\) are found from the three vertices of a patch in the current picture \((x_i, y_i)\) and the reference picture \((x'_i, y'_i)\) by solving the equation 2.25:

\[
\begin{pmatrix}
  x'_1 & x'_2 & x'_3 \\
  y'_1 & y'_2 & y'_3 \\
  1 & 1 & 1
\end{pmatrix} = \begin{pmatrix}
  a_1 & a_2 & a_3 \\
  a_4 & a_5 & a_6 \\
  1 & 1 & 1
\end{pmatrix} \begin{pmatrix}
  x_1 & x_2 & x_3 \\
  y_1 & y_2 & y_3 \\
  1 & 1 & 1
\end{pmatrix} \tag{2.25}
\]

The parameters are obtained with a fractional precision. Because the MC prediction image is produced continuously over the patch borders by the interpolation, blocking artifacts do not appear in the image. Moreover, complex motion such as rotation and zooming can be represented by the prediction.

![Warping Prediction](image)

**Figure 2.1: Warping Prediction**

### 2.3 Lossless Compression Tools

Entropy was originally introduced as a measure of the degree of disorder of a molecular system in thermodynamics. In information theory, it is the measure of information which
was first used by Shannon [80]. The amount of information $I$ transferred in coding a symbol of probability $p$ is given by:

$$I = \log_2 \left( \frac{1}{p} \right)$$

The entropy $H$ of a message is defined as the average information per symbol. It can be expressed by the sum of information contributed by each symbol, $s$, weighted by the probability $p(s)$ of that symbol:

$$H = \sum_s p(s) \log_2 \left( \frac{1}{p(s)} \right)$$

Because $\log_2$ is the base-2 logarithm, the information $I$ is expressed in bits. Therefore, unless the probabilities of symbols are integer powers of 1/2, the ideal code length will not be an integer. To construct an efficient coding system in which the average code length approaches the entropy, various techniques were proposed. Among them, two commonly used methods in image compression are Huffman coding and arithmetic coding.

### 2.3.1 Huffman Coding

Huffman coding [33] is a statistical data compression technique which gives a reduction in the average code length used to represent the symbols of an alphabet. The Huffman code is an example of a code which is optimal in the case where all symbol probabilities are integral powers of 1/2. A Huffman code can be built in the following manner:

- Rank all symbols in the order of probability of occurrence.

- Successively combine the two symbols of the lowest probability to form a new composite symbol; eventually a binary tree where each node is the probability of all nodes beneath it can be built.

- Trace a path to each leaf, noting the direction at each node.

For a given frequency distribution, there are many possible Huffman codes, but the total compressed length will be the same. It is possible to define a canonical Huffman tree, that is, pick one of these alternative trees. Such a canonical tree can then be represented very compactly, by transmitting only the bit length of each code.

### 2.3.2 Arithmetic Coding

The Huffman coder is optimal when and only when the symbol probabilities are integer powers of 1/2, which is usually not the case. The technique of arithmetic coding
2.4: Discussion

[75][77][76][51] does not have this restriction: It achieves the same effect as treating the message as one single unit, and thus attains the theoretical entropy bound to compression efficiency for any source.

Arithmetic coding works by representing a symbol by an interval of real numbers which usually corresponds to the probability of the symbol appearance between 0 and 1. To code the symbol, the arithmetic coder creates a code stream that is a binary fraction pointing to the interval for the symbol being coded. To code additional symbols, the interval created by previous symbol(s) is further sub-divided. As the message becomes longer, the interval needed to represent it becomes smaller and smaller, and the number of bits needed to specify that interval increases. Successive symbols in the message reduce this interval in accordance with the probability of that symbol. The more likely symbols reduce the range by less, and thus add fewer bits to the message. If the message is long enough, the efficiency of the coder approaches 100%[41]

The key technique of using arithmetic coding is to build a suitable model for the source. The simplest model is a fixed one, for example a table of standard letter frequencies for English text which we can then use to get letter probabilities. The arithmetic coder used in JPEG/JBIG is called QM-Coder. It is a descendent of the Q-coder[68] with significant improvements in the interval subdivision [26] and probability estimation.

2.4 Discussion

DCT and related coding methods are still among the most popular algorithms for image and video compression because of their compaction property and relative ease of implementation. However, when they are used in a very low bit rate codec, the block artifacts due to coarse quantisation of the coefficients become visible and deteriorate the image quality.

Wavelet and sub-band based approaches do not have such a drawback when used for very low bit rate compression. The wavelet transforms have a speed comparable to DCT’s, and usually achieve better image quality when used at the same low bit rate. A problem (and advantage) of wavelet transforms is that the most appropriate choice of mother wavelet is largely dependent on the source data. This property of the wavelet transform makes it impossible to develop a general transform that would be suitable for all images. With a carefully designed mother wavelet for the signal being coded, wavelet transforms are capable of achieving a high compression ratio with good reproduction quality.

Unlike transform based compression technique (DCT, wavelet) which changes the en-
ergy distribution of the signal in the transform domain so that the signal can be represented by a few coefficients, VQ is a highly efficient quantisation method that can be used alone as a compression tool or combined with other compression techniques to enhance their strength. The major obstacle of using VQ is that it requires a large amount of computational power and memory storage. Recent developments in multi-stage VQ[48] and pruned tree search [69] have made it possible for VQ to achieve high compression ratios with good reproduction quality.

Fractal compression can be regarded as a special VQ with a virtual codebook. It does not explicitly have a codebook like in VQ. Instead, in IFS the “codebook” is comprised of portions of the attractor as it emerges during iteration. The codebook in VQ is usually shared by all images but the virtual codebook in IFS is specific to each image. Therefore, fractal compression is in fact a more refined version of VQ. For this reason, it is expected to achieve a better compression than VQ. Some experiments show that the fractal compression is capable of producing better image quality than JPEG when compression ratio is greater than 40:1.

In Figure 2.2 to Figure 2.4 a set of rate-distortion curves are given for a DCT based algorithm (JPEG, IJP), Wavelet based algorithm (SARB) and fractal based algorithm (TRNA, TRNB) for grey level images “Barb”, “Boat” and “Lena” from: “ftp://links.uwaterloo.ca/pub/BragZone/GreySet2/Barb/..Results”.

The algorithms under tests are:

- JPEG baseline JPEG, Images Incorporated, version 3.1
- IJP improved JPEG, Independent JPEG Group, version 5b
- SARB treecode/treedecd Wavelet, Said & Pearlman, version 7.01
- TRNA Fractal based, Fisher, 3-level quadtree, version 0.03
- TRNB Fractal based, Fisher, 4-level quadtree, version 0.03

From Figure 2.2 to Figure 2.4, it can be seen that:

(a). Tree encode/decode wavelet compression gives better results at all compression ratios.

(b). Fractal based compression performs worse than other compression methods at low compression ratio. But it has a slower declining rate than other compression methods. Therefore, it outperforms the DCT based compression algorithm at high com-
pression ratio and has the potential to outperform wavelet based compression methods at very high compression ratio.

(c). DCT and wavelet based compression methods show the same trend in rate distortion. However, it would appear that the wavelet based compression has a more stable performance than the DCT based compression at high compression ratios.

(d). Implementation details may affect the performance of an algorithm dramatically.

![Compression of Barb](image.png)

Figure 2.2: Rate-Distortion curves of spatial compression tools for "Barb"

In the temporal domain, motion compensated prediction plays an important role in hybrid structure codecs. Three issues have to be addressed: choice of motion model, estimation of motion parameters and motion compensation techniques.

Traditionally, only translational motion model, BMA based estimation and a simple block based compensation are used. From the discussion, it is clear that the translational motion model should only be used in small regions where the motion field is smooth. BMA should be used on blocks with single coherent motion and a simple block based compensation should be used in a smooth motion field. As these conditions cannot be met in real scene sequences normally, it would be beneficial to use a more complex motion model, and more sophisticated estimation and compensation techniques.
2.4: Discussion

Figure 2.3: Rate-Distortion curves of spatial compression tools for “Boat”

Figure 2.4: Rate-Distortion curves of spatial compression tools for “Lena”
The affine motion model and second order polynomial motion model are the most frequently used complex motion models. The former can deal with motion fields with constant accelerations such as scaling, rotation, shearing and translation. Apart from those motion patterns, the latter can cover an even broader range of motion like planar surface motion under perspective projection [67]. Statistical methods such as the Robust Hough Transform (RHT) can correctly estimate the predominant motion within a region of interest. It is also possible to estimate multiple motions using the same scheme. They can be used with other motion estimation methods to improve the estimate when large regions are involved. Table 2.4 gives the results of an experimental comparison of three different types of sequences (Akiyo, Hall and Weather) for BMA and RHT based motion estimation algorithms.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>PSNR (dB)</th>
<th>Bit Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BMA</td>
<td>RHT</td>
</tr>
<tr>
<td>Akiyo</td>
<td>37.83</td>
<td>37.73</td>
</tr>
<tr>
<td>Hall</td>
<td>37.88</td>
<td>36.07</td>
</tr>
<tr>
<td>Weather</td>
<td>33.37</td>
<td>33.94</td>
</tr>
</tbody>
</table>

Table 2.4: Coding results of BMA and RHT based motion estimation

The overlapped block motion compensation and triangle warping motion compensation take into account adjacent motions. The block artifacts caused by motion compensation are therefore greatly reduced. They also give a better prediction when only translational parameters are estimated. From the literature, it appears that triangle warping gives an even better performance[57]. Figure 2.5 gives comparative results for sequence “Container Ship”. The VM4 corresponds to an overlapped-block motion compensation method whereas P6 corresponds to the triangle warping method, where “no filter” indicates that the deblocking filtering is turned off.

In the lossless compression, there is no doubt that arithmetic coding outperforms Huffman coding as in the latter the symbol probabilities can only be the power of 2. By simply replacing the Huffman coding with arithmetic coding, one can expect 3 to 10% bit rate reduction. Table 2.5 gives comparative coding results for these two approaches for sequences “Coastguard” and “Container” in CIF format.
2.5 Conclusions

From the above discussion, it can be concluded that for spatial compression, DCT based algorithms are still the most suitable tools to apply in low compression ratio applications. When higher compression ratios are required, wavelet and sub-band, VQ, fractal and their combinations may provide better performance than DCT. So far as temporal compression tools are concerned, complex motion model, statistical based parameter estimation, the use of overlapped block as well as triangle warping motion compensation techniques should be included in the short list of measures to improve the performance of existing standards. For the lossless compression tools, arithmetic coding should be chosen whenever possible.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Huffman</th>
<th>Arithmetic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coastguard</td>
<td>505.8kbps</td>
<td>474.1kbps</td>
</tr>
<tr>
<td>Container Ship</td>
<td>155.3kbps</td>
<td>142.5kbps</td>
</tr>
</tbody>
</table>

Table 2.5: Bit rate comparison of Huffman and arithmetic coding
Chapter 3

Multiple Layer Affine Motion Compensated Codec

3.1 Introduction

In a low bit rate video codec, block based approaches still play an important role because of their simplicity and relatively easy implementation in real time. Variable block size coding [29, 46, 96] which adaptively changes the block size according to different motion patterns and the use of the shape of moving objects could deliver a better performance in terms of PSNR and bit rate than that achievable with a fixed block size codec. However, the artifacts present in the decoded sequence especially around motion boundaries, are more apparent and annoying in a variable block size codec because of the use of larger blocks. To improve the coding quality in such regions, various motion segmentation methods have been proposed [50, 60, 102]. For example, the algorithm presented by Lai et al. [50] is based on the assumption that regions moving independently have different grey-levels, and can be segmented by grey-level thresholding. Such an approach is very efficient since motion boundary can be recovered from the reconstructed frame on the receiver side and there is no bit-rate overhead for the coding of the boundary. But it would fail in highly textured regions or in low contrast regions. Moccagatta et al. [60] proposed a method, where the motion segmentation was accomplished by an exhaustive search for the motion boundary and object motions with the objective to minimise the prediction error energy (equivalent to MSE). However, their approach is computationally demanding and does not appear to be amenable to real-time implementation. Moreover, only translational motion model was used in the above approaches which could result in large prediction errors when a complex motion is involved in a large block or region. This may eventually force the block or region to be split into smaller blocks or regions even if they undergo a coherent motion.
3.2: Motion Model and Coding Efficiency

Consequently, this leads to a decline in coding efficiency.

The other important issue in very low bit rate video coding is how to deal with the global motion caused by a moving camera. Such motion will usually involve a large background area and affine or perspective projection transformation.

The desire to improve the coding efficiency and the quality of a variable block size codec lead naturally to the use of more complex motion models. Various studies investigating the benefits of using more complex motion models in motion compensated prediction have been reported [16, 63, 31]. However, the problem of reliable estimation of complex (eg. affine) motion parameters from an image sequence is a very difficult task. The main reason is that objects moving independently may bias the estimation of the global motion. Another complication is due to the fact that estimation of a complex motion requires a search in a high dimensional space which is computationally expensive. Here, we employ a statistical motion estimation algorithm which utilises a Hough Transform based technique for parallel motion estimation and segmentation [102] to estimate the affine motion parameters. To improve the efficiency of the motion compensation stage, we propose a video codec using a layered structure, inner block segmentation and affine motion compensated prediction to get high quality reconstruction of image sequences at low bit rates. The codec takes advantage of the simplicity of the block based algorithms while maintaining a better coding efficiency of segmentation based algorithms. Experimental results confirm the improved performance when compared to existing techniques and standards.

3.2 Motion Model and Coding Efficiency

From the discussions in the previous chapter, it follows that the translational motion model is valid only when the region involved is small or the motion field is very smooth. This condition cannot be met in the majority of real scene sequences. Therefore, a spatial compression for the motion compensated error is required. This simple motion model + error coding scheme works well in low bit rate coding for video conference type sequences. However, when very low bit rate coding is required for other types of sequences, such a scheme cannot give satisfactory results because the coding of large regions with coherent motion is inefficient. To improve the efficiency of motion compensated prediction, a larger region size and more complex motion model such as an affine motion model and a polynomial motion model have to be used. Here, we propose to use the affine motion model as a candidate for complex motion compensated prediction.

Note that the six parameter affine motion model needs more bits to encode motion
parameters than the two parameter translational motion model. This overhead may sometimes offset the benefit of a better motion prediction. In [31], it has been reported that the affine motion model does not perform particularly well in local motion prediction involving slow moving scenes.

Generally speaking, a complex motion model is more suitable for use on large regions, whereas a simple translational motion model is sufficient for small regions. Based on these observations, we propose an approach which uses different motion models on different size regions. This approach makes a compromise between motion compensation and spatial compression and thus a more efficient codec can be expected.

3.3 Multiple Layer Codec Structure

As we pointed out in the previous section, the optimal performance of a video codec may be achieved if the motion model can match the motion field in the scene. To balance the bit overhead in encoding motion parameters and motion compensated error image, we propose a multiple layer codec structure which uses a complex motion model for large regions while maintaining a low bit overhead for motion parameters on small regions.

In our early approach [102, 101], a variable block size structure and straight line segmentation were used to partition the image frame into regions. The translational motion model was used in these approaches. Recently, we have extended our earlier approach into a multiple layer affine motion compensated codec [100, 98] in which the affine motion prediction is applied to large regions to get better motion prediction and the translational motion prediction is applied to small regions to reduce the bit overhead. The proposed codec consists of 4 layers:

(a). Frame level: 6-parameter affine motion estimation and compensation is applied. This gives accurate and bit efficient compensation for global motion in scenes where camera is not stationary. A more accurate motion parameter quantiser is also used here.

(b). Superblock level: The superblock consists of $2 \times 2$ macroblocks. A simplified 4 parameter affine model which is able to cope with translational and rotational motion is applied for coding at the group of blocks level. We have found that for $32 \times 32$ regions, this simplified model gives satisfactory results.

(c). Macroblock level: only translational model is used. Because the region involved is small, the translational model can reduce the bit rate and deliver a reasonable
performance.

(d). Basic block level: when a motion boundary is present inside a macroblock, translational motion parameters are estimated for each region (based on the luminance component). The motion parameters for chrominance blocks are estimated from luminance parameters.

For the last three layers, an inner block segmentation may be applied if a motion boundary happens to be inside the block. For a given sequence, the first frame is encoded using a spatial compression technique (DCT+entropy coding). The consecutive frames are compressed using temporal and spatial compression techniques. Firstly, global motion compensation and foreground/background segmentation are applied at frame level. The background region is stored in a background memory. Affine motion model is used at this level. For foreground regions, an extended quadtree decomposition is used to encode the remaining three layers. This scheme differs from traditional quadtree decomposition in two respects:

(a). Different motion models are used at different block sizes.

(b). A motion based inner block segmentation is used when a dual motion is detected within a block.

The codec also has an improved rate control mechanism which can change the quantisation step on both spatial compressor and temporal compressor. It also allows variable resolution in both spatial and temporal domains. Finally, a distributed deblur intra coding and motion compensation counter (section 3.3.7) are used to maintain the sharpness of the image.

In the following subsections, we will discuss some implementation details of the codec.

3.3.1 Global Motion Estimation and Compensation

Motion estimation and motion compensation as one of the vital parts in a hybrid structure codec are the key to efficiently reducing temporal redundancy. The motion estimation is normally based on the previous reconstructed frame and the current processed (original) frame so that the quantisation errors of motion parameters can be compensated. Such a scheme also introduces some gray-level noise (corresponding to the reconstruction errors) into the motion estimation procedure. Such noise can sometimes seriously bias the estimation of true displacement vectors. Moreover, often more than one moving object exist within a region of interest. Such coexistence of multiple moving objects inside the
region may cause biased estimation of the true displacement vector if not handled properly. Biased estimation of motion may have a little impact when only small regions are involved. However, when large regions or the whole frame are involved, any bias in the motion estimate may result in poor performance of the codec as excessive number of bits may be needed to encode the motion compensation error. Therefore, a better motion estimator is required which should not only be robust in the presence of noise, but also be capable of coping with multiple motions within a region. Commonly used estimation techniques such as block matching, which minimise mean-square-error (MSE) or sum-of-absolute-difference (SAD) between motion compensated blocks, give biased estimates for blocks with multiple motion or for those degraded by the presence of significant noise. To obtain an unbiased estimate of the motion parameters, a statistical estimation approach conceptually based on the Hough Transform and Robust Statistics[14] is used here. Rather than minimising the MSE error, we maximise the support measure $H$ defined by a robust redescending kernel $\rho$:

$$H(\tilde{d}) = \sum_{x,y \in \text{Region}} \rho(I_0(x + dx, y + dy) - I_1(x, y))$$  \hspace{1cm} (3.1)$$

where $I_0$ and $I_1$ are the intensity in the reference and consecutive frames respectively and $\tilde{d} = (dx, dy)$ is the motion parameter vector to be estimated. In our experiments, Tukey redescending kernel was used. Figure 3.1 shows the shapes of: (a) standard quadratic kernel used in block matching and (b) Tukey redescending kernel.

The main idea behind this approach is that during iterative estimation of the motion parameters of one moving object, outliers (e.g. pixels belonging to other moving objects or
for which noise is significant) are ignored. Consequently we not only obtain a more accurate motion estimate, but also perform motion segmentation in parallel with estimation.

### 3.3.2 Background/Foreground Segmentation

Segmentation into Foreground/Background regions can be achieved using an algorithm as that one would use for identifying motion/stationary regions in sequences with stationary background in [101]. For sequences with a stationary background, a simple change detection based segmentation may work well. For sequences with moving background, an extended change detection which involves the computation of the global motion compensated frame difference is needed to take the background motion into account. For a well motion compensated frame difference, a Gaussian distribution will provide a good approximation to the residual pixel value distribution. In order to segment foreground/background regions, we assume that the grey level of a background pixel $x_{\text{grey}}$ which is well motion compensated falls into the interval $[-\delta, \delta]$ with a probability $P(\delta)$:

$$P(\delta) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\delta}^{\delta} e^{-\frac{x^2}{2\sigma^2}} dx$$

$$= \frac{2}{\sigma\sqrt{2\pi}} \int_{0}^{\delta} e^{-\frac{x^2}{2\sigma^2}} dx$$  \hspace{1cm} (3.2)

Therefore, we can choose a threshold $T_d = \delta$ so that $P(T_d)$ is greater than or equal to a certain value. To calculate $P(T_d)$, we first estimate the probability distribution $p(x_i)$ of grey level values $x_i$ by its histogram $H(x_i)$:

$$p(x_i) = \frac{1}{M} H(x_i)$$  \hspace{1cm} (3.3)

where $M = \text{rows} \times \text{columns}$ is the size of the image. Then $P(T_d)$ can be calculated by accumulating $p(x_i)$:

$$P(T_d) = \sum_{i=0}^{T_d} p(x_i)$$  \hspace{1cm} (3.4)

We can find $T_d$ that satisfies the specified constraint on $P(T_d)$ and use it as the threshold for change detection. After the change detection, we obtain a binary image on which morphological open-close operations are performed to eliminate isolated small segments. Finally, a contour link is applied to pixels marked as foreground pixels to form the final foreground/background segmentation. Figure 3.2 illustrates this process by showing respectively the original image, the frame difference image between frame 0 and frame 4, the binary image after change detection, and the foreground/background segmentation superimposed on the original image.
For video conference sequences, a heuristic threshold map is used further to improve the segmentation result (Figure 3.3). When such a map is used, the probability for zone 0 is used as a basis to calculate the threshold for other zones.

\[
P_0 = P(T_{d0}) \\
P_i = 1 - (1 - P_{i-1})/2, i = 1, 2, 3
\]

(3.5)

The corresponding threshold \(T_{di}\) can then be calculated using Equation 3.4 by substituting \(P(T_d)\) with \(P(T_{di}) = P_i\). For comparison, results estimated using the heuristic zone threshold and uniform threshold are shown in Figure 3.4. It can be seen that the use of the heuristic zone threshold significantly reduces the noise on top of the difference image while maintaining good sensitivity to changes caused by the object movement.
3.3: Multiple Layer Codec Structure

Figure 3.3: Heuristic threshold map

(a) Without heuristic zone thinhresholding  (b) With heuristic zone thresholding

Figure 3.4: Comparison of heuristic thresholding

3.3.3 Background Memory

The background memory is used to improve the coding efficiency on sequences with stationary background. The use of the facility is controlled by the output of the global motion estimator. If the global motion estimation indicates that the sequence is of stationary background, the background memory is turned on. It can also be controlled by external flags.

When the background memory is turned on, the zero motion search will use the background memory as reference instead of previous frame. The advantage of using background
memory as a reference frame is that previously registered background can be used when a previously occluded background region reemerges in consequence of the motion of foreground objects. Therefore, no intra coding will be necessary for such kind of reappearing regions. Figure 3.5 shows how background memory can help encoding occluded background regions. In Figure 3.5 (c), the reappearing region can be directly copied from background memory instead of using spatial coding so that a better coding efficiency can be achieved. If the object moves back and forth in a limited distance (such as a speaker's head), the background memory can save a considerable number of bits that would otherwise be needed to encode the repeatedly reappearing background regions.

The background memory must be the same at both transmitter and receiver side of the codec. Here, we use the reconstructed image from the bit stream to create such a
background memory. The update procedure consists of three steps:

(a). When a first intra frame or a scene change frame appears, the whole background memory is flushed with that frame. A confidence matrix in which each element corresponds to an 8×8 area in the background memory is created and initialised to 1.

(b). For successive frames, if the motion vector for an 8×8 region is zero, the corresponding confidence element is increased by 1. The background for that region will not be updated.

(c). If the motion vector for a certain 8×8 region is non-zero, the following rules will be used to update the background memory:

- If the neighbouring blocks all have non-zero motion vectors, the block is in the centre of a moving object. No background memory update is performed. The corresponding confidence element is set to zero.
- If the block appears to be an edge block identified by a discontinuous motion field, the direction of the motion vector is used to identify whether there is an occlusion. If the block moves towards the regions with zero motion, no occlusion is present. In this case, the background memory will not be updated but the confidence element will be set to zero. If the block moves away from the region with zero motion, it means an occluded region is present.
- Occluded regions are firstly compared with the background memory to see whether the content has been stored in the background memory. If an occluded region is not in the background memory, the region has to be spatially encoded and the background memory is updated. The corresponding confidence element will be set to 1. Otherwise, the confidence element will be increased by 1.

In the motion estimation procedure, the zero motion will only be checked against background memory with a non-zero confidence element. For blocks with the corresponding zero confidence elements, the zero motion will be checked against the previous frame. In this way, a small motion in a large uniform region may be dealt with as a zero motion. Therefore, a better coding efficiency can be expected.

3.3.4 Extended Quadtree Decomposition

From the description of the multiple layer codec, it can be seen that the last three layers are in fact a variable block size codec structure with quadtree decomposition. To cope with
multiple motion within a block and to take advantage of foreground/background segmentation, the standard quadtree decomposition is modified to accommodate new elements introduced. Although more than two moving objects could be considered, in the following discussion we confine ourselves to at most two different motions per block. After global motion compensation and foreground/background segmentation, the image is divided into equal size blocks with a maximum block size of $K \times K$ pixels. Then each of the blocks is processed by the decomposition procedure shown in Figure 3.6.

Figure 3.6: Extended Quadtree Decomposition

Please note that in the decomposition procedure, motion estimation only applies to
blocks in the foreground region. Thus the computational load is reduced as the motion estimation procedure is one of the most time consuming parts of the algorithm. It can also be noted that the foreground/background segmentation does not need to be very accurate because any wrongly positioned block will create a relatively large global motion compensation error and has to be processed in the foreground processing branch.

The blocks falling into the background region take about 40% to 90% of the total number of blocks (Table 3.1). For these blocks, a run-length coding technique is applied to efficiently encode the block status. For the foreground blocks, an inter-block prediction and differential coding technique is used to encode the motion parameters. If a block under consideration, say \( B_c \), has exactly the same velocity as one of its neighbouring blocks, there is no need to transmit the velocity. It is sufficient to indicate which of the neighbours moves with the same motion. We process blocks from left to right, and from top to bottom, so only neighbouring blocks located above and to the left are considered. The main difficulty is that blocks may have different sizes. To overcome this problem, all velocities are mapped onto the finest grid, called the basic grid, corresponding to the image divided into blocks of the minimal size (basic blocks). Each of the basic blocks is indexed, relatively to the block under consideration. The total number of basic neighbouring blocks is:

\[
N_{B_c} = 2 \frac{L_c}{L_{\text{min}}} \quad (3.6)
\]

where \( L_{\text{min}} \) is the side length of the basic block in pixels, while \( L_c \) denotes the size of the block under consideration \( B_c \). The procedure selects the basic neighbouring block with an index \( k \in [1, N_{B_c}] \) that minimises the error criterion \( \epsilon \) for the block \( B_c \):

\[
\epsilon(B_c, B_{Bk}) \leq \epsilon(B_c, B_{Bj}); \forall k, j \in [1, N_{B_c}] \quad (3.7)
\]

Figure 3.7 shows possible examples of relations between block \( B_c \) and its neighbouring blocks.
To encode the index of basic blocks, we use a pseudo-Huffman code which is based on the fact that the decomposition structure of the blocks at the top and to the left of the current block is known on the receiver side. We use the following rules to create the pseudo-Huffman code:

- In a pair of blocks, the top one is assigned 0 and the bottom one is assigned 1; the left one is assigned 1 and the right one is assigned 0.
- If two basic blocks belong to the same level decomposition, they should have the same code.

Figure 3.8 gives two examples of pseudo-Huffman code with a given neighbouring block decomposition. In Figure 3.8 (a), the pseudo-Huffman codes for the basic blocks are:

- 1, 1, 1, 1 (left)
- 011, 010, 00, 00 (top)

In Figure 3.8 (b), the pseudo-Huffman codes for the basic blocks are:

- 11, 10 (left)
- 0, 0 (top)

It can be seen that if the size of the block at the top and to the left of the block being processed is equal or greater than the size of the block being processed, only one bit is needed to encode the neighbouring index. In the worst case, the scheme uses the same number of bits as the fixed length coding.

In the event that the above procedure fails to find the same motion velocity from the
neighbouring blocks, a differential coding is used to reduce the redundancy. The reference velocity takes the median of the velocity of the first basic block on the left, the velocity of the first basic block at the top and the velocity of the first basic block at the top-right corner. The velocity difference is encoded using variable length coding.

3.3.5 Motion Boundary Coding

In the above procedure, when dual motion mode is detected, one needs to encode motion boundaries within the block. There are several ways to encode the object boundaries. The most commonly used one is the chain-code. It is able to describe accurately the shape of any area. However, the bit rate required for chain code is high. For the four-link chain-code, under certain constraint, the theoretical lower bound is

$$\log_2(1 + \sqrt{2}) = 1.27$$

per node[25] (after arithmetic coding). To increase the coding efficiency of boundary coding, some approximation has to be made. Here, we propose two coding schemes which are able to encode boundaries efficiently while maintaining a low overhead of bit rate.

3.3.5.1 Straight Line Approximation

The straight line approximation was first proposed in [102, 101].

In order to have a compact description of the boundary, we assume that it can be approximated by a straight line. This assumption works well in practice, and its main
advantage is that the algorithm has a degree of freedom and can split large blocks into small ones, to improve line fitting. Figure 3.9 shows the representation of the motion boundary within a block. A straight line is used to describe the motion boundary within a block. Two parameters are needed to encode the line representing an object boundary: the intersection point $X_0$ and the line direction $\theta$ (e.g., the angle between the line and the base of the block). Calculation shows that it is more economical (in terms of bit rate) to encode a boundary than to split the block and to code the motion parameters of each sub-block. Furthermore, such an approach also gives a better approximation of the real motion boundary, thus a better reconstruction can be obtained. As an example, Figure 3.10 shows a region around the head of the "salesman" (a) and correspondent motion segmentation (b). Note that this approach works equally well for motion boundaries that do not coincide with intensity edges, as opposed to methods based on gray-level segmentation, however this is at the expense of extra bits used for the coding of the boundary line.

Such an approximation requires 5 bits to encode the angle of the line and 2 bits to indicate which side of the block is intersected, and 3-5 bits (depending on the size of the block) to encode the intersection point. For a $32 \times 32$ block, the straight line approximation requires a total of 12 bits (which corresponds to a 0.0117 bits/pixel overhead) to encode the line. For $16 \times 16$ blocks, the overhead increases to 0.043 bits/pixel. Such overhead is comparable to the average bit rate for some head and shoulder type sequences (e.g. akiyo in CIF format, 0.0572bits/pixel). Therefore, a more compact presentation of the object boundary is needed.

### 3.3.5.2 Fixed Pattern Partition

In experiments, we have found that the overhead for spatially encoding a well motion-compensated block is quite small. This allows us to use an even coarser representation of
the motion boundary. In [99], a fix-pattern segmentation method to encode the motion boundary within the block was proposed.

Firstly, a set of basic patterns is created which can partition a block into halves. The basic set of patterns consists of four partition maps as shown in Figure 3.11. From these four basic patterns, we can derive another 8 secondary patterns as shown in Figure 3.12. Because of the symmetry, we can assume that the probability of the four basic partition patterns is the same and the probability of the eight derived partition patterns is also the same. Therefore, a simple variable length coding table (Table 3.2) can be obtained. When a motion boundary is found within a block, one of the partition patterns is selected as the best-matching pattern. Compared to line segmentation, the average code length used to encode the motion boundary is reduced from 10-12 bits to 3.6 bits per block.

3.3.6 Rate Control

One of the most commonly used rate control methods is to use the buffer content as the criterion. This approach is exemplified by the TMN-5 rate control [87]. The control of the
3.3: Multiple Layer Codec Structure

![Diagram of derived partition patterns]

Figure 3.12: Derived Partition Patterns

<table>
<thead>
<tr>
<th>Partition Pattern</th>
<th>VLC Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic (a)</td>
<td>100</td>
</tr>
<tr>
<td>Basic (b)</td>
<td>101</td>
</tr>
<tr>
<td>Basic (c)</td>
<td>110</td>
</tr>
<tr>
<td>Basic (d)</td>
<td>111</td>
</tr>
<tr>
<td>Derived (a)</td>
<td>0111</td>
</tr>
<tr>
<td>Derived (b)</td>
<td>0110</td>
</tr>
<tr>
<td>Derived (c)</td>
<td>0101</td>
</tr>
<tr>
<td>Derived (d)</td>
<td>0100</td>
</tr>
<tr>
<td>Derived (e)</td>
<td>0011</td>
</tr>
<tr>
<td>Derived (f)</td>
<td>0010</td>
</tr>
<tr>
<td>Derived (g)</td>
<td>0001</td>
</tr>
<tr>
<td>Derived (h)</td>
<td>0000</td>
</tr>
</tbody>
</table>

Table 3.2: Variable Length Coding Table

bit generating components which normally involves changing both the quantisation step on the quantiser and the temporal sub-sampling rate, so that a target bit rate can be achieved. This method does not take into account the statistical properties of the image frame and therefore it cannot achieve optimal rate control results while maintaining perceptually good image quality.

Here we propose a new rate control scheme which uses both the buffer content and spatial complexity to obtain a bit creation model so that a more accurate rate control can be achieved. The proposed rate control mechanism controls the following parameters in the bit generating procedure:

- Temporal resolution
- Spatial resolution
3.3: Multiple Layer Codec Structure

- Quantiser for spatial compression tools
to achieve the desired bit rate. There are two stages of rate control - a global rate control and a local rate control. In the global rate control, all three components mentioned above can be controlled by using predefined preference and/or negotiable preference. In the local rate control, only the quantiser associated with the spatial compression tools can be controlled. The preference is given to the global rate control for two reasons:

(a). Human visual system is more sensitive to spatial discrepancy than to temporal discrepancy.

(b). Extra bit overhead is needed to encode a local adjustment of the spatial compression quantiser.

3.3.6.1 Intra frame bit generation model

When DCT based intra frame coding scheme is used, the average number of bits created for each intra block is

\[ B = B_{dc} + \frac{C \times NZQAC_{QP} \times QP}{Q} \]  

(3.9)

where \( B_{dc} \) is the average number of bits used for the DC component, \( NZQAC_{QP} \) is the average number of non-zero quantised AC coefficients obtained with quantisation step \( QP \), \( Q \) is the quantisation step to be used and \( C \) is the average number of bits used for each AC coefficient to encode the block. Through Equation 3.9, the total number of bits required for the whole frame is

\[ B_{\text{intra}} = B \times \frac{\text{rows} \times \text{columns} \times 6}{16 \times 16} \]  

(3.10)

where \( \text{rows} \) and \( \text{columns} \) denote the horizontal and vertical size of the image frame, 16x16 is the size of a macroblock. In Equation 3.10, the only variable is \( B \). Therefore, \( B \) is the key to intra frame bit rate prediction. In Equation 3.9, the first item of the equation contributes to the bits required for the DC components. The second item represents the bits required for coding the AC components. Under H.261/H.263 syntax, \( B_{dc} \) equals 8 for intra block coding. \( QP \) and \( Q \) are known parameters. \( NZQAC_{QP} \) can be calculated from the input image. The only unknown is the average code length to encode the non-zero AC coefficient \( C \) which needs to be decided by experiment. Table 3.3 gives the experimental results for various CIF sequences and Table 3.4 gives the experimental results for various QCIF sequences.

From Table 3.3 and Table 3.4, it can be seen that:
For the same image, the average code length $C$ is rather consistent when $NZQAC$ and the bits spend for intra frame coding are calculated using the same quantisation step.

For different images, the average code length $C$ is rather close. For all $C$ calculated using the same quantisation step for intra bits and $NZQAC$, the minimum is 6.08 and the maximum is 8.82.

The average value of $C$ is slightly smaller when $NZQAC$ is calculated using quantisation step 8 as compared to $NZQAC$ calculated using quantisation step 16.

The standard deviation of $C$ is also smaller when calculated using $QP = 8$.

Based on the above observations, we can assume that:

- The average value of $C$ is proportional to quantisation step $QP$.
- The deviation of $C$ is also proportional to quantisation step $QP$.

As the difference of average value of $C$, denoted as $\bar{C}$, and the deviation of $C$, denoted as $\sigma_c$, using different quantisation steps $QP$ is relatively small, we can use a linear approximation.
Table 3.4: Prediction constant for QCIF image sequences

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Bits required QP=8</th>
<th>NZQAC*QP QP=8</th>
<th>C (NZQAC8) QP=8</th>
<th>C (NZQAC16) QP=8</th>
<th>C (NZQAC8) QP=16</th>
<th>C (NZQAC16) QP=16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>22008</td>
<td>12880</td>
<td>32.82</td>
<td>31.19</td>
<td>7.08</td>
<td>6.67</td>
</tr>
<tr>
<td>Bream</td>
<td>22712</td>
<td>13072</td>
<td>37.07</td>
<td>33.10</td>
<td>6.52</td>
<td>6.04</td>
</tr>
<tr>
<td>Carphone</td>
<td>26256</td>
<td>14920</td>
<td>39.57</td>
<td>38.14</td>
<td>7.31</td>
<td>6.92</td>
</tr>
<tr>
<td>Children</td>
<td>48080</td>
<td>24960</td>
<td>87.74</td>
<td>86.95</td>
<td>6.65</td>
<td>6.20</td>
</tr>
<tr>
<td>Claire</td>
<td>18776</td>
<td>12000</td>
<td>23.34</td>
<td>23.36</td>
<td>8.09</td>
<td>8.36</td>
</tr>
<tr>
<td>Coastguard</td>
<td>26336</td>
<td>13576</td>
<td>41.19</td>
<td>30.59</td>
<td>7.06</td>
<td>5.77</td>
</tr>
<tr>
<td>Container</td>
<td>30040</td>
<td>17280</td>
<td>48.61</td>
<td>48.08</td>
<td>7.01</td>
<td>6.94</td>
</tr>
<tr>
<td>Foreman</td>
<td>28088</td>
<td>15664</td>
<td>44.80</td>
<td>42.18</td>
<td>7.02</td>
<td>6.56</td>
</tr>
<tr>
<td>Grandma</td>
<td>19688</td>
<td>10824</td>
<td>30.78</td>
<td>23.35</td>
<td>6.54</td>
<td>5.31</td>
</tr>
<tr>
<td>Hall</td>
<td>27832</td>
<td>16552</td>
<td>42.66</td>
<td>44.47</td>
<td>7.29</td>
<td>7.45</td>
</tr>
<tr>
<td>Mad</td>
<td>18056</td>
<td>10592</td>
<td>25.14</td>
<td>20.84</td>
<td>7.13</td>
<td>6.26</td>
</tr>
<tr>
<td>Missa</td>
<td>12512</td>
<td>8424</td>
<td>14.81</td>
<td>12.44</td>
<td>7.06</td>
<td>6.68</td>
</tr>
<tr>
<td>News</td>
<td>33104</td>
<td>18664</td>
<td>54.37</td>
<td>55.91</td>
<td>7.02</td>
<td>6.89</td>
</tr>
<tr>
<td>Salesman</td>
<td>28040</td>
<td>14400</td>
<td>48.44</td>
<td>37.38</td>
<td>6.47</td>
<td>5.36</td>
</tr>
<tr>
<td>Silent</td>
<td>25808</td>
<td>13744</td>
<td>42.55</td>
<td>35.17</td>
<td>6.66</td>
<td>5.69</td>
</tr>
<tr>
<td>Suzie</td>
<td>16896</td>
<td>9904</td>
<td>23.97</td>
<td>19.63</td>
<td>6.82</td>
<td>5.79</td>
</tr>
<tr>
<td>Td</td>
<td>14080</td>
<td>8712</td>
<td>17.15</td>
<td>13.01</td>
<td>7.33</td>
<td>6.22</td>
</tr>
<tr>
<td>Trevor</td>
<td>29264</td>
<td>15592</td>
<td>50.58</td>
<td>44.41</td>
<td>6.53</td>
<td>5.77</td>
</tr>
<tr>
<td>Weather</td>
<td>76232</td>
<td>42104</td>
<td>136.43</td>
<td>151.83</td>
<td>7.06</td>
<td>7.37</td>
</tr>
</tbody>
</table>

| Average   |                    |               | 6.98           | 6.43           | 7.83           | 7.15           |
| σc        |                    |               | 0.378          | 0.761          | 0.584          | 0.546          |

To calculate \( \bar{c} \) and \( \sigma \) for a given \( QP \), using the data in Table 3.3 we can derive the following equations for CIF image sequences:

\[
\bar{c} = 6.84 + 0.026875 \times QP \tag{3.11}
\]

\[
\sigma_c = 0.289 + 0.025 \times QP \tag{3.12}
\]

Similarly, we can derive the relationships for QCIF image sequences from Table 3.4:

\[
\bar{c} = 6.81 + 0.02125 \times QP \tag{3.13}
\]

\[
\sigma_c = 0.21 + 0.021 \times QP \tag{3.14}
\]
In practical applications, the rate control algorithm normally requires that after the first frame coding, the buffer contents should be about constant. This is so called initial buffer regulation problem. It is normally achieved by two means:

(a). Optimal selection of quantisation step $Q$.

(b). Adjusting the frame rate to drain the excessive bits.

As we pointed out in introduction, the selection of initial quantisation step $Q$ is purely heuristic. 16 is chosen in TMN-5 and 20 is recommended in VM-5. Those numbers do not take into account any of the application requirements and therefore, the coding delay and image quality cannot be guaranteed.

With the knowledge of average code length $C$ and $NZQAC$, we can predict and control the bits generated for the first frame. Using the above bit generation model, the optimal quantisation step can be estimated by the following iterative calculation procedure:

(a). Calculate the initial quantisation step $Q_0$ by

$$Q_0 = \text{Int} \left( \frac{C_{QP} \times NZQAC_{QP} \times QP}{\frac{B_{budget} \times 256}{\text{rows} \times \text{columns} \times 8} - 8} \right)$$

(b). Set $i=1$;

(c). Calculate $C_{Q_{i-1}}$ using either Equation (3.11) or Equation (3.13) according to image size.

(d). Estimate the $i$th iteration quantisation step $Q_i$ using $C_{Q_{i-1}}$

$$Q_i = \text{Int} \left( \frac{C_{Q_{i-1}} \times NZQAC_{QP} \times QP}{\frac{B_{budget} \times 256}{\text{rows} \times \text{columns} \times 8} - 8} \right)$$

(e). Restrict the range of $Q_i$ to $[1, 31]$

$$Q_i = \text{MIN}(31, Q_i)$$

$$Q_i = \text{MAX}(1, Q_i)$$

(f). If $Q_i = Q_{i-1}$ then stop else goto step b.

This procedure normally converges in just 1 or 2 iterations. Therefore, it is a very practical way to get the optimal quantisation step for buffer regulation.
3.3.6.3 MACC rate control algorithm

The rate control strategy is as follows:

(a). Define control parameters

- Target Bit Rate: $R$
- Buffer size: $B_s$
- Buffer content pointer: $B_p$
- Buffer full position: $B_f$
- Buffer empty position: $B_e$
- Frame rate of input sequence: $F_{Ri}$
- Buffer draining rate per frame: $B_{pp}$
- Target frame rate: $F_{R\text{target}}$
- Buffer draining rate under the target frame rate: $B_{ft}$
- Frame increment number: $N$
- Target number of bits for the next frame: $T$

(b). Initialisation of the rate controller.

- $B_s = R/2$ (corresponding to 0.5 sec. delay for communication applications) or $B_s = 100R$ (for other type applications);
- $B_p = 0$
- $B_f = 0.8B_s$
- $B_e = MIN(0.2B_s, 0.1R)$
- $B_{pp} = R/F_{Ri}$
- $B_{ft} = R/F_{R\text{target}}$
- $N = \frac{F_{Ri}}{F_{R\text{target}}}$
- The bits budget for the first intra frame sets the buffer contents to the middle of the buffer after the buffer draining or half of the required bit rate, whichever is smaller:
  
  \[ R_{first} = MIN(0.5B_s + B_{ft}, R/2) \]

(c). Selecting QP for the first frame:
• Use equations 3.15, 3.16 and 3.17 to calculate the optimal quantisation step $QP_{first}$ for the first frame.

• Estimate minimum and maximum bits generated by using $QP_{first}$. Using equation 3.11 and 3.12 for CIF image or equation 3.13 and 3.14 for QCIF image, calculate $C$ and $\sigma_c$ for $QP_{first}$. We use $C - 2\sigma_c$ for $C_{min}$ and $C + 2\sigma_c$ for $C_{max}$. Using equation 3.9 and 3.10 we can obtain $B_{1min}$ and $B_{1max}$ as follows:

  • while ($B_{1min} - B_f < B_s$) {
    if ($QP_{first} > 1$) {
      $QP - 1$;
      recalculate $B_{1min}$;
    }
    else
      stuffing bits;
  }

  • while ($B_{1max} - B_f > B_f$) {
    if ($QP_{first} < 31$) {
      $QP + 1$;
      recalculate $B_{1max}$;
    }
    else
      use lower spatial resolution;
  }

• if lower spatial resolution has to be used, recalculate $NZQAC$ and $C$ for the given resolution and repeat the above procedure until a satisfactory buffer content is reached. To avoid a buffer overflow, the following condition must be met at all time:

$$B_{1max} - B_{pp} \leq B_s$$

(d). Select the control parameters for the first inter frame following the first intra frame.

The same spatial resolution as that of the first intra frame is used. After encoding the first frame using the selected $QP_{first}$ and the spatial resolution, the total number of bits generated is $B_{first}$.

• Target bits and temporal resolution calculation:
3.3: Multiple Layer Codec Structure

- Fixed frame rate:

\[ B_p = B_{first} - B_{ft} \]
\[ T = \text{MAX}(B_{ft}, 0.4R - B_p) \]

- Variable frame rate:

\[ B_p = B_{first} - B_{ft} \]
\[ \text{if}(B_p \geq B_f) N = N + (B_p - B_f)/B_{pp}; \]
\[ \text{if}(B_p \leq B_e) N = N - (B_e - B_p)/B_{pp}; \]
\[ N = \text{MAX}(1, N) \]
\[ N = \text{MIN}(8, N) \]
\[ B_p = B_{first} - N \times B_{pp}; \]
\[ FR_{target} = FR_i/N; \]
\[ B_{ft} = R/FR_{target}; \]
\[ T = \text{MAX}(B_{ft}, 0.4R - B_p); \]

- Initial quantisation level:

\[ QP_0 = QP_{intra} \]

- Start encoding using \( QP_0 \). To reduce the overhead of sending side information, a local adjustment is carried out for every 9th macroblock:

\[ QP_{imb} = QP_0[1 + 12(B_{imb} - \frac{imb_{MBT}}{R})] \]
\[ QP_{imb} = \text{MAX}(1, QP_{imb}); \]
\[ QP_{imb} = \text{MIN}(31, QP_{imb}); \]
\[ \Delta QP = QP_{imb} - QP_{imb-9}; \]
\[ \Delta QP = \text{MAX}(-2, \Delta QP); \]
\[ \Delta QP = \text{MIN}(2, \Delta QP); \]
\[ QP_{imb} = QP_{imb-9} + \Delta QP; \]

where \( B_{imb} \) is the number of bits generated until current macroblock.

(e). Control strategy for the remaining inter frames. The number of bits generated by the previous inter frame is \( B_n \), where \( n \) is the total number of inter frames coded.
• Target bits and temporal resolution calculation for the current frame:
  - Fixed frame rate:
    \[ B_p = B_p - B_f \]
    \[ T = \text{MAX}(B_f, 0.4R - B_p) \]
  - Variable frame rate:
    \[ B_p = B_p - B_f \]
    \[ \text{if}(B_p \geq B_f)N = \frac{(B_p - B_f)}{B_p} + N; \]
    \[ \text{if}(B_p \leq B_e)N = N - \frac{(B_e - B_p)}{B_p}; \]
    \[ N = \text{MAX}(1, N) \]
    \[ N = \text{MIN}(8, N) \]
    \[ B_p = B_p - N \times B_{pp}; \]
    \[ \text{FR}_{\text{target}} = \frac{\text{FR}_i}{N}; \]
    \[ B_{ft} = \frac{R}{\text{FR}_{\text{target}}}; \]
    \[ T = \text{MAX}(B_{ft}, 0.4R - B_p); \]

• Calculate the average quantisation step in the previous inter frame:
  \[ QP_n = \frac{1}{MB} \sum_{i=1}^{MB} QP_{imb} \]

• Predict bits for the current frame using the previously encoded inter frames:
  \[ B_{\text{pre}+1} = \frac{1}{nQP_n} \sum_{i=1}^{n} QP_i \times B_i \quad (3.18) \]
where \( QP_i \) is the average quantisation step for previous \( i \)th frame, \( B_i \) is the number of bits spent on the previous \( i \)th frame, \( n \) is the number of inter frames encoded previously. To facilitate the computation, the above equation can be rewritten as:

\[ B_{\text{pre}+1} = \frac{1}{nQP_n} \left( \sum_{i=1}^{n-1} QP_i \times B_i + QP_n \times B_n \right) \]
\[ = \frac{1}{nQP_n} (B_{\text{pre}} \times (n - 1)QP_{n-1} - 1) + \frac{1}{n} B_n \]
\[ = B_{\text{pre}} \frac{QP_{n-1}}{QP_n} \left( \frac{1}{n} - \frac{1}{n} \right) + \frac{1}{n} B_n \quad (3.19) \]
therefore, $B_{pre_{n+1}}$ can be calculated iteratively from previous estimation of $B_{pre}$. However, it is known that the correlation between adjacent frames is much stronger than frames far apart. In equation 3.19, when $n$ is very large, the impact of the last inter frame will become very small. To overcome this problem, we restrict the maximum value $n$ to a predefined number $n_t$ (normally in the range 5 to 20). The equation 3.19 can then be rewritten as:

$$B_{pre_{n+1}} = \begin{cases} 
B_{pre} \frac{Q_{P_n}^{T_1}}{Q_{P_n}^{T_n}} (1 - \frac{1}{n}) + \frac{1}{n} B_n, & n \leq n_t \\
B_{pre} \frac{Q_{P_n}^{T_2}}{Q_{P_n}^{T_n}} (1 - \frac{1}{n_t}) + \frac{1}{n_t} B_n, & n > n_t 
\end{cases}$$  \hspace{1cm} (3.20)

- Calculate initial quantisation step

$$Q_{P_0} = Q_{P_n}[1 + \frac{(B_{pre_{n+1}} - T)}{2T}]$$

- Decide spatial resolution:
  - if $B_p < 0.3B_g$ and $Q_{P_0} < 16$ and the spatial resolution is not the highest resolution, the spatial resolution is enhanced by one level. The bits predicted by equation 3.20 should be rescaled by the following equation:

$$B_{pre_{n+1}} := 3B_{pre_{n+1}}$$

Recalculate $Q_{P_0}$ using the new $B_{pre_{n+1}}$.

- if $B_p > 0.7B_g$ and $Q_{P_0} > 24$ and the spatial resolution is not the lowest resolution, the spatial resolution should be reduced by one level. The bits predicted by equation 3.20 should be rescaled by the following equation:

$$B_{pre_{n+1}} := \frac{1}{3}B_{pre_{n+1}}$$

Recalculate $Q_{P_0}$ using the new $B_{pre_{n+1}}$.

- Start encoding using $Q_{P_0}$. If local adjustment is enabled, then for every 9th macroblock do an adjustment:

$$Q_{P_{imb}} = Q_{P_n}[1 + \frac{(B_{pre_{n+1}} - T)}{2T}] + 12 \frac{(B_{imb} - \frac{imb}{MBT})}{R}$$

$$Q_{P_{imb}} = MAX(1, Q_{P_{imb}});$$

$$Q_{P_{imb}} = MIN(31, Q_{P_{imb}});$$

$$\Delta Q_P = Q_{P_{imb}} - Q_{P_{imb-9}};$$

$$\Delta Q_P = MAX(-2, \Delta Q_P);$$

$$\Delta Q_P = MIN(2, \Delta Q_P);$$

$$Q_{P_{imb}} = Q_{P_{imb-9}} + \Delta Q_P;$$
where $B_{\text{imb}}$ is the number of bits generated until the current macroblock.

As compared to other rate control strategies such as TMN-5, the proposed method has the following advantage:

- The proposed method does not require the initial quantisation step to be selected by external means which is crucial for a successful rate control algorithm.
- It uses both spatial and temporal rescaling to help stabilising the buffer content whereas TMN-5 only uses temporal rescaling.
- More accurate control is achieved by using the bits prediction model.
- It supports both online control and off-line control by initialising the buffer size differently.
- It uses a similar macroblock level control as TMN-5, but with different target bits and predicted bits estimation method.

### 3.3.7 Deblur Intra Coding

Because of the low-pass filtering effect of sub-pixel accuracy motion compensation, the sharpness of the image is lost in the process of repetitive application of motion compensation. To remedy this problem, intra coding has to be applied after certain number of inter codings. There are three different approaches to apply the intra coding.

- Apply intra coding to the whole frame at regular intervals. This approach not only restores the sharpness of the image but also recovers any block which has been erroneously motion compensated due to transmission errors or for any other reason. Its drawback is that the bit-rate overhead is high which may contribute to the buffer overflow. Consequently a larger buffer is required to accommodate the bursts in the bit stream.

- Apply intra coding to the regions which have been motion compensated at regular intervals. For sequences with stationary background, this approach significantly reduces the bit overhead in sending intra coded regions. However, for sequences with non-stationary background, there is no difference between this approach and the previous one.

- Only regions that have been motion compensated a certain number of times will be intra coded. Any intra coding during the coding progress will reset the counter so
that unnecessary intra coding is avoided. A motion compensation counter is required for each smallest block in this scheme. Whenever the value of the counter is greater than a given threshold, an intra coding is forced for that block. Because usually only a small percentage of blocks is motion compensated repeatedly many times, the overall overhead for extra intra coding is reduced. The only disadvantage is a potential loss of robustness to transmission errors and therefore a channel coding may be required to increase error resilience.

In our approach, we opt for the third approach which is more suitable for very low bit rate coding. It also has the advantage that a more evenly distributed bit stream will be created as compared to the other two schemes.

3.4 Experimental Results

The effectiveness of affine motion compensation was tested using a set of MPEG-4 CIF test sequences. Figure 3.13 (a) and Figure 3.13 (b) show the average PSNR and average bit rate (7.5 Hz frame rate) using the proposed algorithm, the algorithm presented in [101] (VSBM), and H.261 [36, 34] respectively.

VSBM presented in [101] has almost the same structure as the algorithm presented in this thesis except that it uses translational motion model on all block layers and there is no global motion estimation and segmentation stage to stabilised the background motion. Results of H.261 were obtained by $p \times 64$ implementation as presented in [34].

From Figure 3.13 (b), we notice that for sequences with global motion and large local motion such as “bream”, “coastguard” and “foreman” the proposed method gains most in bit rate due to improved motion compensated prediction and multiple layer structure. For sequences with stationary background but exhibiting large and complex motion such as “children”, “mother and daughter”, “news”, “silent” and “weather”, the proposed algorithm also gains in bit rate from the use of the affine motion model. However, for sequences with stationary background and small motion like “akiyo”, “container” and “hall”, the bit rate is slightly higher than VSBM which uses only translational motion model. This is caused by a small bit overhead introduced by the additional motion parameters used by the affine motion model. Overall, the bit rate (on average) is reduced by 40% as compared to H.261 and by 9.2% as compared to VSBM. The average overall PSNR of the proposed algorithm is slightly lower than that of VSBM and H.261 (0.026 dB and 0.057 dB respectively). An interesting phenomenon is that for sequences with stationary background and small motion, the proposed algorithm gives higher PSNR than VSBM.
and H.261. This may imply that the codec still benefits from the use of complex motion model by improved motion prediction and thus achieves a higher PSNR. (The bit rate is slightly higher than for VSBM, but still lower than for H.261).

Figure 3.14 shows a comparison of the PSNR and bit rate for the proposed algorithm and H.261 using the "akiyo" sequence without the bit-rate control. The PSNR in Figure 3.14 (a) shows that the proposed algorithm has a less turbulent PSNR than that of the
3.4: Experimental Results

Figure 3.14: Comparative results for the "Akiyo" sequence (a) PSNR (b) Bit rate

H.261 which has two peaks when the intra frame coding is applied to an entire frame (with the rate control, there will be two valleys around this point). Because of the use of MC counter and distributed de-blurring intra mode, the proposed algorithm has a more stable bit rate than H.261 which also has two peaks around the de-blurring intra point (Figure 3.14 (b)).

In order to evaluate the benefits of the motion segmentation we compare the algorithm
### 3.4: Experimental Results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>PSNR (dB)</th>
<th>Bit Rate (kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>proposed</td>
<td>No Seg.</td>
</tr>
<tr>
<td>Akiyo</td>
<td>36.05</td>
<td>35.91</td>
</tr>
<tr>
<td>Bream</td>
<td>34.45</td>
<td>34.03</td>
</tr>
<tr>
<td>Carphone</td>
<td>34.05</td>
<td>33.93</td>
</tr>
<tr>
<td>Children</td>
<td>32.22</td>
<td>32.04</td>
</tr>
<tr>
<td>Coastguard</td>
<td>32.03</td>
<td>31.89</td>
</tr>
<tr>
<td>Container</td>
<td>33.98</td>
<td>33.84</td>
</tr>
<tr>
<td>Foreman</td>
<td>33.46</td>
<td>32.75</td>
</tr>
<tr>
<td>Hall</td>
<td>35.09</td>
<td>34.86</td>
</tr>
<tr>
<td>Mad</td>
<td>35.52</td>
<td>35.31</td>
</tr>
<tr>
<td>News</td>
<td>34.14</td>
<td>33.97</td>
</tr>
<tr>
<td>Salesman</td>
<td>33.16</td>
<td>33.05</td>
</tr>
<tr>
<td>Silent</td>
<td>33.64</td>
<td>33.52</td>
</tr>
<tr>
<td>Suzie</td>
<td>35.54</td>
<td>35.23</td>
</tr>
<tr>
<td>Td</td>
<td>35.02</td>
<td>34.76</td>
</tr>
<tr>
<td>Weather</td>
<td>32.35</td>
<td>32.18</td>
</tr>
<tr>
<td>Average</td>
<td>34.05</td>
<td>33.82</td>
</tr>
</tbody>
</table>

Table 3.5: Average PSNR and bit rate for tested sequence

with its variant in which motion segmentation is inhibited (one motion mode per block). MPEG-4 QCIF test sequences were used and the rate control was disabled. In Table 3.5 we give the average PSNR and bit rates for all 15 tested sequences. It can be seen that the proposed algorithm gives higher PSNR on all 15 tested sequences and lower bit rate on 11 sequences. On average, the proposed algorithm gives a 0.2 dB higher PSNR and 2.2 kbps lower bit rate.

To measure the effectiveness of the motion compensation (MC), the total number of bits spent on encoding motion compensated errors (DCT-P Bits) and the total number of blocks to be spatially coded (SPCB) were calculated. It is known that for a well motion compensated frame, the number of bits spent on MC errors is small. If multiple motions exist within a block, one normally needs more bits to encode the MC errors or sometimes the block has to be spatially encoded. The comparison of results is shown in Table 3.6. From Table 3.6, it can be seen that the proposed algorithm always gives better motion compensation in terms of DCT-P bits and SPCB. On the other hand, a better prediction does not necessarily mean a lower bit rate. The bit rates for the “earphone”, “mad”, “silent” and “suzie” sequences are higher than the bit rates achieved by a variant of the algorithm with motion segmentation disabled. Further investigation showed that the increased overhead to encode the partition parameters (the line) and the extra motion
3.4: Experimental Results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>DCT-P Bits</th>
<th>SPCB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>proposed</td>
<td>No Seg.</td>
</tr>
<tr>
<td>Akiyo</td>
<td>98692</td>
<td>124512</td>
</tr>
<tr>
<td>Bream</td>
<td>869627</td>
<td>1114346</td>
</tr>
<tr>
<td>Carphone</td>
<td>551890</td>
<td>667634</td>
</tr>
<tr>
<td>Children</td>
<td>1242046</td>
<td>1479153</td>
</tr>
<tr>
<td>Coastguard</td>
<td>902410</td>
<td>1085300</td>
</tr>
<tr>
<td>Container</td>
<td>221026</td>
<td>247931</td>
</tr>
<tr>
<td>Foreman</td>
<td>560295</td>
<td>823161</td>
</tr>
<tr>
<td>Hall</td>
<td>217075</td>
<td>278053</td>
</tr>
<tr>
<td>Mad</td>
<td>123681</td>
<td>170353</td>
</tr>
<tr>
<td>News</td>
<td>341390</td>
<td>441591</td>
</tr>
<tr>
<td>Salesman</td>
<td>249507</td>
<td>333490</td>
</tr>
<tr>
<td>Silent</td>
<td>269663</td>
<td>372868</td>
</tr>
<tr>
<td>Suzie</td>
<td>110855</td>
<td>154934</td>
</tr>
<tr>
<td>Td</td>
<td>659397</td>
<td>840949</td>
</tr>
<tr>
<td>Weather</td>
<td>442232</td>
<td>583644</td>
</tr>
</tbody>
</table>

Table 3.6: Total number of DCT-P bits and SPCB

parameters can sometimes eliminate the benefits from better motion prediction. Table 3.7 gives the comparison of the total number of bits spent on motion related parameters (MC bits, including partition parameters, motion model, and motion parameters) and the gain from improved motion compensation (MC Gain). The gain is calculated by subtracting the total number of bits (MC + DCT.P) used by the proposed technique from the number of bits used by the version without motion segmentation.

\[
MC_{Gain} = (MC + DCT.P)_{proposed} - (MC + DCT.P)_{NoSeg}.
\]  

From Table 3.5 and Table 3.7, it can be seen that for those sequences with a higher bit rate the gain from motion is negative. However, the gain does not take into account that improved MC reduces the number of bits used for intra block coding. In the sequence “Td”, the negative gain is compensated by the reduced number of intra coded blocks. The overall bit rate for this sequence is still improved. We also notice that the proposed algorithm performs very well with hybrid natural and synthetic sequences (class E) such as “Bream”, “Children” and “Weather”. Most of the sequences in this group involve highly-textured background, with dominant high spatial frequency contents. Such textured regions are very sensitive to any failure in motion compensation.
3.5 Conclusion

In this chapter, we presented an algorithm which combines robust motion segmentation and motion prediction with a variable size block coding. The algorithm introduces a new way to solve the motion boundary segmentation problem and exploits an inter-block prediction and pseudo-Huffman coding scheme which can effectively reduce the bit-rate without sacrificing the image quality.

The experimental results are encouraging: under the same PSNR constraint, the proposed algorithm can achieve a lower bit rate than the standard variable block size codec in which only translational motion model is used. The algorithm can also achieve significantly lower bit rate than H.261. The gain from better motion prediction and motion segmentation is also evaluated. The results suggest that for the majority of the sequences, improved motion prediction and segmentation means improved image quality (higher PSNR) and improved coding efficiency (lower bit rate). However, the gain in coding efficiency could be negative for some sequences because of the overhead to encode extra motion parameters and partition parameters. The experiment reveals that the gain in coding efficiency from improved motion prediction and segmentation is proportional to the spatial complexity of the sequence being encoded. Consequently, a sequence classifier could be useful to ensure

<table>
<thead>
<tr>
<th>Sequence</th>
<th>MC Bits proposed</th>
<th>MC Bits No Seg.</th>
<th>MC Gain (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>29902</td>
<td>14072</td>
<td>9990</td>
</tr>
<tr>
<td>Bream</td>
<td>234465</td>
<td>80720</td>
<td>9074</td>
</tr>
<tr>
<td>Carphone</td>
<td>285639</td>
<td>91328</td>
<td>-58567</td>
</tr>
<tr>
<td>Children</td>
<td>153358</td>
<td>44191</td>
<td>127940</td>
</tr>
<tr>
<td>Coastguard</td>
<td>258549</td>
<td>94634</td>
<td>28975</td>
</tr>
<tr>
<td>Container</td>
<td>59240</td>
<td>35141</td>
<td>2806</td>
</tr>
<tr>
<td>Foreman</td>
<td>377800</td>
<td>118335</td>
<td>3401</td>
</tr>
<tr>
<td>Hall</td>
<td>54037</td>
<td>22885</td>
<td>29826</td>
</tr>
<tr>
<td>Mad</td>
<td>104571</td>
<td>42602</td>
<td>-15297</td>
</tr>
<tr>
<td>News</td>
<td>105491</td>
<td>34843</td>
<td>29553</td>
</tr>
<tr>
<td>Salesman</td>
<td>88811</td>
<td>31323</td>
<td>26405</td>
</tr>
<tr>
<td>Silent</td>
<td>168212</td>
<td>47267</td>
<td>-17740</td>
</tr>
<tr>
<td>Suzie</td>
<td>106819</td>
<td>40903</td>
<td>-21837</td>
</tr>
<tr>
<td>Td</td>
<td>256881</td>
<td>70837</td>
<td>-4492</td>
</tr>
<tr>
<td>Weather</td>
<td>66537</td>
<td>20717</td>
<td>95592</td>
</tr>
</tbody>
</table>

Table 3.7: Total number of bits spent on motion related parameters and gains from motion prediction
the best result for all sequences.
Chapter 4

Reconfigurable Multiple Tool Video Codec

4.1 Introduction

In the current multimedia environment, multiple coding standards exist in parallel as the result of various application requirements. It is known that to date there is no universal coding technique which is able to cover the entire range of applications. It is also known that there are coding techniques which are optimised for certain type of applications. Communication in a multimedia environment requires the codec to cover a much wider range of applications than those currently available. Even for the same application range, some approaches may be more suitable than others for certain image sequences. The experimental results in the previous chapter and our research \[100, 99\] also indicate that the proposed codec structure (MACC) may sometimes give a worse performance than a simpler codec structure for a certain type of sequences. Therefore, a codec, which can be configured according to specific application requirements and to be reconfigured to accommodate changes, can be expected to give a better performance.

One of the possible solutions to this problem is to put several codecs together and to apply the most suitable one for the application at hand. One only needs to develop an interface to various standard codecs. The advantage of this approach is that many mature codecs developed according to the international standards can be used and their performance can well be predicted as a function of different applications. There are two disadvantages of this loose integration of multiple codecs. Firstly, the solution cannot meet the requirement to operate in a time varying multimedia environment such as Internet. Switching between different codecs requires an extra overhead such as head information, intra frame coding etc. Consequently, it may result in a lower coding efficiency. Secondly,
the use of multiple codecs would increase the system complexity unnecessarily. From our earlier discussion, we have concluded that many parts of various coding standards are common such as Huffman coding, motion estimation and motion compensation, DCT etc. Reusing of those common components will reduce the complexity and cost of the codec. Such an idea leads to another possible solution.

As we know, the major difference among different coding standards lies in the bitstream syntax and coding control strategy. The basic compression techniques are almost the same. If we put those common compression techniques for public use and implement a different codec structure which complies with certain coding standards, we can have a codec which can be configured to multiple standards by using different codec structures and bitstream syntax. Such a codec can enjoy the advantages of the first approach but with a reduced complexity, achieved by reusing common compression components. Furthermore, new codecs may be created by combining different compression techniques. The close integration of multiple codec structures will mean that no intra frame coding is needed when changing from one codec structure to another. Thus a better coding efficiency than that afforded by the first approach can be achieved.

Taking the idea of the second approach further, the compression tools and the codec structures can be generalised to a few tool boxes so that any compression tool and codec structure may be combined together to form a real codec. Based on this idea, we propose a reconfigurable multiple tool (RMT) video codec which uses generalised compression tools and codec structure to satisfy the broad requirements set by multimedia communications. The codec also has an open structure to ensure an easy addition of new compression techniques. The bitstream syntax of the codec guarantees backward compatibility with the current coding standards. The bitstream created by standard codecs can either be encapsulated in the RMT bitstream or sent out as a stand alone codec so that it can facilitate communication with terminals other than RMT. With some modification, the syntax of RMT can be fully compatible with the top level of the upcoming MPEG-4[64] video standard.

In this chapter, a framework for reconfigurable multiple tool codec (RMT) is proposed. The bitstream syntax, system integration and compatibility with existing coding standards are discussed in section 4.2. In section 4.3, the communication unit of the RMT codec is described. The tool configuration unit of the RMT codec is discussed in section 4.4 where a theoretical tool selection method based on constrained optimisation theory is also presented to show that the optimal performance of the codec can be achieved. The possibility of using human assistance and heuristics in the tool selection procedure is also
discussed and a query based system is proposed. Finally, the tool archiving unit of the RMT codec is described in 4.5 and a brief summary is given in 4.6.

4.2 RMT Codec: A System Description

We have already mentioned in the introduction that no coding technique alone can serve the entire range of applications but each technique is more suitable to one particular task. We also pointed out that a simple combination of multiple codecs not only wastes hardware and software resources but also results in lower coding efficiency in a time varying coding environment. The RMT codec, which consists of a combination of multiple compression tools and codec structures, is a very promising solution to serve very wide application requirements.

4.2.1 General structure

In general, an RMT codec should satisfy a few criteria so that it can remain competitive and compatible with the corresponding stand alone codecs:

(a). It should be able to provide the same functionalities as loosely integrated multiple codecs. It should provide the same bitstream syntax as integrated standards and have an equal or better performance.

(b). It should allow new compression techniques to be easily added to the codec's tool boxes.

(c). It should allow new codec structures to be easily added to the codec's structure library.

(d). The configuration of a selected tool set should be easily changed and reconfigured during the coding procedure (on the fly reconfiguration) without reinitialising the codec.

(e). The tool set and codec structure should be embedded in the bitstream so that a decoder can reconstruct the image directly from bitstream without any a priori knowledge of the encoder.

To satisfy the above criteria, the proposed RMT codec consists of three major parts: a communication unit, a tool configuration unit, and a tool archiving unit. The kernel of the codec is a reconfigurable virtual RMT codec which can be initialised using a codec structure and tools from the tool archiving unit. The selection and initialisation of
the codec structure and coding tools are carried out in the tool configuration unit, in which
the application requirements are translated to the choice of a specific codec structure and
tool set so that the application requirements can be satisfied. The selected codec structure
and tool set are then initialised to form a working codec. The image and data input/output
are performed in the communication unit which also provides a user interface to the codec.
A simple illustration of the operation of the RMT codec is shown in Figure 4.1.
4.2.2 Bitstream syntax

The bitstream syntax of RMT should both satisfy the coding standard involved and the RMT requirements. Therefore, the bitstream syntax of RMT consists of two layers: a discardable RMT head layer and a codec layer. The RMT head layer can be removed (in duplex communication with standard based terminals) or filtered out (in data storage) when the RMT codec communicates with standard based codecs. The syntax diagram of the RMT codec is illustrated in Figure 4.2 where the picture layer and the RMT header layer are shown.

![Syntax diagram for RMT codec](image)

**Figure 4.2: Syntax diagram for RMT codec**

4.2.2.1 Picture layer

As shown in Figure 4.2, the picture layer consists of a discardable RMT header layer and a codec layer. When a predefined codec structure is chosen, the picture layer can either start from the RMT header layer or the picture layer of the codec. Whether or not to use the RMT header in a predefined codec structure is normally determined by external means.

4.2.2.2 RMT header layer

The RMT header itself is byte aligned. It consists of a 32-bit starting code (RSC), a 16-bit flag field (RTF) and a byte aligned variable length tool update data field (TUD).

**RSC RMT starting code (32 bits)** RSC is a 32-bit word. The value of RSC is 0000 0000 0000 1000 0000 1000 0000 1000.
RTF RMT Tool Flag field (16 bits) A 16-bit word RTF immediately following RSC is used to indicate the presence of tool update words. From the most significant bit (Bit 0) to the least significant bit (Bit 15), the definition of each bit is shown as follows:

- Bit0 PCS (Predefined codec structure) Set to 1 for the use of a predefined codec structure, 0 for the use of the RMT codec structure;
- Bit1 CS Codec structure update;
- Bit2 SPCI Spatial compression tool for intra coding update;
- Bit3 SPCP Spatial compression tool for inter coding update;
- Bit4 MC Motion compensation tool update;
- Bit5 VLC Variable length coding update;
- Bit6 QM Quantisation method update;
- Bit7 RCTL Rate control tool update;
- Bit8 ARF Artifacts removal filtering tool update;
- Bit9-Bit11 Reserved. Always 000;
- Bit12 - Bit15 User defined coding tool update. Up to 4 user defined tool boxes can be added to the RMT tool shelf.

The above Bit2 - Bit8 and Bit12 - Bit15 are set to 1 when a tool update is required otherwise they will be set to 0. To avoid simulation of the starting code in the codec layer, an exclusive/or operation with word 0101 0101 0101 0101 is applied to RTF before transmission. At the receiver side, the same exclusive/or operation is applied again to restore the value of RTF.

TUD Tool Update Data field (variable length, minimum length 0 bits) As discussed above, a value 1 in the RTF flag indicates a requirement for a tool update of the corresponding tool box. A variable length word is used to indicate which tool is used in that tool box for current picture coding. Table 4.1 gives the definition of the variable length code used in the tool update data field. In theory, the number of tools in a tool box is not limited in such a coding system. In reality, it is rarely required that more than 20 tools be used in a single tool box in the RMT codec and thus 8 bits are sufficient. After a
<table>
<thead>
<tr>
<th>Tool index</th>
<th>Number of bits</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1100</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1101</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1110</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>1111 0000</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1111 0001</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1111 0010</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>1111 0011</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>1111 0100</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>1111 0101</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>1111 0110</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>1111 0111</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>1111 1000</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>1111 1001</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>1111 1010</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>1111 1011</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>1111 1100</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>1111 1101</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>1111 1110</td>
</tr>
<tr>
<td>21</td>
<td>16</td>
<td>1111 1110 0000 0000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>275</td>
<td>16</td>
<td>1111 1111 1111 1110</td>
</tr>
<tr>
<td>276</td>
<td>32</td>
<td>1111 1111 1111 1111 0000 0000 0000 0000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>65800</td>
<td>32</td>
<td>1111 1111 1111 1111 1111 1111 1111 1111 1111 1110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: VLC table for encoding tool index

careful selection and optimisation, the number of tools in a tool box can be reduced to 6, or even 3 where only 2 or 4 bits are needed for each tool box. To avoid simulation of the starting code in the codec layer and to ensure a byte alignment for the tool update date field, a 2-bit RSTUF stuffing word 10 is used. Whenever 12 successive zeros are detected in the tool update data field, 2 stuffing words will be inserted to avoid a starting code simulation. On the receiver side, the first 1010 after 12 successive zeros will be skipped. At the end of the tool update data field, 0 to 3 stuffing words may be inserted to obtain a byte alignment. In this way, the RMT header can be safely added to the bitstream of any predefined codec structure and will be filtered out when communicating with a standard...
4.3: Communication Unit

The communication unit is the interface between the RMT codec and the outside world. It consists of three interfaces: user interface, image I/O interface, and data I/O interface.

4.3.1 User Interface

The application requirements and the codec performance are interchanged through the user interface. The interface also provides a means for the user to intervene in the coding process and to override some of the codec configurations. The user should be able explicitly to configure the codec according to previous experience.

The user interface also provides access to the codec performance output for analysis and comparison. One of the special features in the RMT codec is that the performance data and codec configuration can be exported to a database (e.g. Microsoft Access) or spreadsheet (e.g. Excel) for archiving and a further analysis.

4.3.2 Image I/O Interface

The image I/O interface provides a means to input the image signal to the encoder and to output image signals from the decoder. It currently supports all five 4:2:0 YUV source formats (Sub-QCIF, QCIF, CIF, CIF\times4, CIF\times16) used in the H.263 standard.

4.3.3 Data I/O Interface

The bitstream from the data channel is made accessible in the data I/O interface. Functionalities provided by the interface include data input/output, a synchronisation check and codec header recognition.
4.4 Tool Configuration Unit

The performance of the RMT codec depends on the correct selection of the tool set. For a specific application, an experienced researcher may be able to select a good set of tools and a codec structure to get a good codec performance using his/her previous experience. However, such a role may not be expected from an ordinary user. An automated tool selection procedure or human aided semi-automatic tool selection procedure should be an integrated part of the RMT codec.

4.4.1 Problem Statement

To satisfy the requirements of various applications, the RMT codec tool selector should meet a few criteria.

(a). It should be able to identify the communication requirements set by the terminal it communicates with and to select a corresponding codec structure.

(b). The application requirements for the given input sequence should be met if at all possible. In the event that not all the requirements (bit rate, picture quality and coding delay etc.) can be met simultaneously, a user defined preference should be respected to achieve a compromise. The selected tool should be able to give the best performance to meet the application requirements for a given input sequence.

The first criterion can easily be met by identifying the PSC code in the incoming bit stream or by detecting external signals. The second criterion leads to a conditional optimisation problem. It can be stated as follows:

For a given application requirement with a bit rate constraint $B_a$, picture distortion constraint $D_a$, coding delay constraint $T_a$ and image sequence $S$, a codec structure $CS$ and a tool set $TL_{CS}$ should be chosen so that

\[ B(CS(TL_{CS})) \leq B_a \]  \hspace{1cm} (4.1)
\[ D(CS(TL_{CS})) \leq D_a \]  \hspace{1cm} (4.2)
\[ T(CS(TL_{CS})) \leq T_a \]  \hspace{1cm} (4.3)

For a given sequence, there may be more than one $CS(TL_{CS})$ that can satisfy the above conditions. The optimal codec structure and tool set will depend on the application type.
For real time communication, the optimisation problem would be formulated as:

minimise

\[ \text{MIN}(D(CS(T_{LCS})) ) \]  \hspace{1cm} (4.4)

subject to

\[ C_1 : R(CS(T_{LCS})) \leq R_o \]
\[ C_2 : T(CS(T_{LCS})) \leq T_a \]

For data storage applications, coding delay \( T_a \) is not important and no constant bit rate is required. Therefore, one would choose an acceptable level of distortion and optimise \( B \):

\[ \text{MIN}(R(CS(T_{LCS})) ) \]  \hspace{1cm} (4.5)

subject to

\[ C : D(CS(T_{LCS})) \leq D_o \]

The user may define other factor(s) to optimise, but the above two are the most commonly used application types in the RMT tool selection procedure.

4.4.2 Optimal tool selection

The above optimisation problem has been well studied in mathematics under the name optimisation of constrained functions. Consider a general form of the minimisation problem:

\[ \text{MIN}(f(x)) \]

subject to

\[ g_i(x) \leq 0, i = 1, 2, ..., m \]

where

\[ x = (x_1, x_2, ..., x_n)^T \]

is a vector in \( n \) dimensional space. There are two mathematical solutions to the above problem. The basic idea is to modify the objective function to convert the constrained optimisation problem to an unconstrained problem.
Lagrangian multiplier method [4] One way to convert the constrained optimisation problem to an unconstrained one is to use Lagrange multipliers. Let $f, g_i, 1 \leq i \leq m$, have continuous partial derivatives in an open set containing $x^*$. If $x^*$ minimises $f(x)$ subject to the constraints $g_i(x) \leq 0, 1 \leq i \leq m$, which satisfy the linear independence constraint qualification condition, then there exists nonnegative Lagrange multipliers $\lambda_1, \ldots, \lambda_m$ such that

$$f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0$$

$$\sum_{i=1}^{m} \lambda_i g_i(x^*) = 0$$

$$\lambda_i \geq 0, 1 \leq i \leq m$$

Defining the Lagrangian function by

$$\Phi(x, \lambda) = f(x) + \langle \lambda, g(x) \rangle$$

where

$$\lambda = (\lambda_1, \ldots, \lambda_m)^T$$

and

$$g(x) = [g_1(x), g_2(x), \ldots, g_m(x)]^T$$

the Kuhn-Tucker theorem can be stated concisely as

$$\nabla_x \Phi(x^*, \lambda) = 0$$

$$\langle \lambda, \nabla_\lambda \Phi(x^*, \lambda) \rangle \geq 0$$

where $\nabla_x$ denotes the gradient with respect to $x$ and $\nabla_\lambda$ is the gradient with respect to $\lambda$.

As pointed out in [12], the Lagrangian function $\Phi$ may not be search-able by purely numerical methods to locate all local minima. The conditions to use Lagrangian multipliers require that $g$ is strictly convex and $f$ is strictly convex or concave for minimisation and maximisation, respectively. Those conditions are not always possible to verify in real applications.

Penalty functions method Another way to convert constrained optimisation problems into unconstrained problems is to modify the objective functions with some functions of
constrained equations. One such form of modified function is

\[ F(x, k) = f(x) + \sum_{i=1}^{m} k_i g_i(x)^2 \]

where \( k_i \) are large positive-valued constants if \( f \) is to be minimised and large negative quantities for those \( f \) which are to be maximised. The problem should now be treated as an unconstrained one, and as \( k_i \) are increased, \( g_i \) are forced to zero at the optimum. The choice of \( k_i \) is critical and it should be progressively increased during the course of the search.

Back to tool selection problems stated above, they lead to typical constrained minimisation problems. The use of the Lagrangian multiplier method in optimal coding of image sequences has been reported in [73]. The optimisation objective is set to minimise the bit rate \( R \) while keeping distortion \( D \) constant, i.e.,

\[ \Phi(\lambda, x) = R(x) + \lambda D(x) \]

where \( x \in S \), \( S \) is the set of possible tool combinations. For a given \( \lambda \), the minimisation of the cost function \( \Phi \) results in a solution \( x^*(\lambda) \), an associated rate \( R^*(\lambda) \) and a distortion \( D^*(\lambda) \). It has been demonstrated [81] that, for positive \( \lambda \), the pairs \( (R^*(\lambda), D^*(\lambda)) \) trace out the optimal rate distortion curve. In [73], the optimisation is carried out on \( \lambda \), i.e., the goal of the optimisation is to find a \( \lambda_c \) and the corresponding minimised cost function \( \Phi(\lambda_c) \) so that \( D^*(\lambda_c) = D_0 \). Such an optimisation procedure can guarantee that the optimal solution will be found if it exists. However, it cannot guarantee a solution as it might not exist. Besides, it is very time consuming as for each given \( \lambda \) the whole set of \( x \) has to be searched to find the minimum cost function \( \Phi(\lambda) \). The process may be accelerated by pruning unrealistic tool combinations from \( S \) and using a convex search to find \( \lambda_c \).

The penalty function method has the same computational load as the Lagrangian multiplier method when it is applied to the tool selection problem in video coding. The acceleration of the algorithm will be the key to successful tool optimisation. One of the possible solutions is to prune the set of candidate tool combinations into a subset which only contains the most likely tool combinations for a given situation. In the next subsection, we will discuss other possible approaches which exploit human assistance and heuristics.
4.4.3 Human Assistance and Heuristics

As we discussed above, the optimal tool selection may be achieved in theory but is almost impossible in reality. Considering the computational requirement, only a small set of tool combinations can be explored in the optimisation. Fortunately, in video coding, many tool combinations can be safely pruned out. One example is that global motion compensation tool will be useless for sequences with stationary background. If we can detect all such conditions, the tool combination set may be pruned to a tool set of very small size. However, many seemingly simple problems for humans are in fact very difficult for machines to solve. One example is that a human can easily tell what is moving in the scene but a huge amount of effort is needed for a machine to reach the same conclusion. Therefore, with the knowledge and assistance of an expert, some tasks may be greatly simplified.

Here, we propose a questionnaire based human assistance system for the RMT codec tool selector. When the codec is switched on, a few questions will be asked by the codec in the form of a multiple choice question and a numerical confidence level ranging from 0 to 10, in which 0 is the least confident (not know at all) and 10 is the most confident (certain). For answers with confidence lower than 5, its value may be overridden by the codec.

The following is a sample set of questions that may be asked by the codec:

- Presence of camera motion: Yes or no; if "Yes"
  - Camera motion pattern: surveillance, other type

- Subject type: human, car, boat, other moving objects;

- Scene composition: head and shoulder, general

- Background type: natural, synthetic

- Background texture: low, medium, high

Depending on a previous response, further questions may be asked. Some of the tools will be chosen according to the answers to the questionnaire. For example, if we get a set of answers like the following:

- Presence of camera motion: no;

- Subject type: human;
4.5 Tool Archiving Unit

All codecs and tools come from the tool archiving unit which can be divided into two areas: a codec structure library and a tool shelf. In the codec structure library, two categories of codec structures are archived: predefined codec structures and RMT codec structures. The former guarantees that the RMT codec is fully compatible with known coding standards and algorithms. The latter provides a more flexible environment to configure the codec for a specific application so that a better performance can be achieved. A universal codec structure called virtual codec structure is provided in the codec structure library as an abstract codec for use with the codec initialiser. The compression techniques used in a given codec structure are from the tool shelf in which various tool boxes are provided to carry out certain compression tasks. In each tool box, an abstract prototype tool called virtual tool is provided to list all possible operations and functionalities for the tool box. The virtual tool can be substituted by a real tool from the tool box in the configuration and reconfiguration procedure. The real tool may or may not provide all the operations and functionalities listed in the virtual tool. From the above discussion, we can see that the codec structure library is in fact a special tool box with codec structures as tools. Because the codec structure and the compression tool are working at different levels, we have separated the codec structure from the tool box shelf and put it as a stand alone entry. The hierarchy of the tool archiving unit is indicated below

- Codec structure library
  - Virtual codec structures

- Scene composition: head and shoulder;
- Background type: natural;
- Background texture: low;

we can expect that the sequence will be easy to encode and no global motion tool should be used. As the scene is a head and shoulder type, the region of interest enhancement tool and face tracking tool may be used to enhance the subject quality, etc. On the other hand, if the background is synthetic, a graphic-oriented coding method may be used to encode the background instead of a transform or vector quantisation based spatial compression technique.
4.6 Summary

A systematic framework for a reconfigurable multiple tool codec is presented in this chapter. The various parts of the codec have been discussed in detail. A mathematical tool selector is developed using constrained optimisation methodology. The codec structure not only provides an efficient means for multimedia visual communications. It also provides a friendly environment for researchers and developers. The extendibility of the structure of the codec also gives the user some confidence in the ability to exploit new tools and techniques that continuously become available in this fast evolving technology area. Finally, we propose a possible implementation scheme for the RMT codec. A specific implementation of a prototype of the RMT codec will be studied in the next chapter.
Chapter 5

The Design of A Prototype RMT Codec

5.1 System Description

In this chapter, we design a prototype RMT codec to put the ideas in the previous chapter into reality. Two major concerns in the design of a prototype are compatibility and reconfigurability. In order to achieve reconfigurability, the RMT codec is divided into two layers: codec structure and coding tools. In the codec structure layer, two popular international standards (H.261 and H.263) as well as the MACC codec were implemented as predefined codecs. To make the full use of the available coding tools, a fully reconfigurable codec structure was developed (Baseline codec). In the coding tool layer, apart from wrapping tools extracted from H.261/H.263 and MACC, some non-standard tools which may give better performance under certain conditions such as wavelet and vector quantisation were developed. A simple tool selector and sequence classifier based on statistical measures derived from the image sequence are also developed. A detailed list of the coding tools used in the prototype codec can be found in Table 5.1.

In the following sections, we focus on various codec structures and coding tools and their roles in the RMT prototype codec.

5.2 Predefined Codec Structures

The backward compatibility requires that the codec be capable of communicating with terminals using current and upcoming international coding standards. Such functionality is implemented through predefined codec structures in which a predefined combination of tools is used to provide the required functionalities. The reconfigurability in a predefined codec structure is limited to the specification of the coding standards used. As RMT
Table 5.1: Implemented Tools in RMT Prototype

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Implemented Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPCI</td>
<td>DCT, Wavelet, ZTW, ZTVQ</td>
</tr>
<tr>
<td>SPCP</td>
<td>DCT, Wavelet, ZTW</td>
</tr>
<tr>
<td>VLC</td>
<td>Avlc, Hvlc2d, Hvlc3d</td>
</tr>
<tr>
<td>ME</td>
<td>RHT, BMA, MBMA</td>
</tr>
<tr>
<td>GME</td>
<td>RHT</td>
</tr>
<tr>
<td>MC</td>
<td>Basic, Adv</td>
</tr>
<tr>
<td>Quant</td>
<td>H26P, MPEG</td>
</tr>
<tr>
<td>Rate Control</td>
<td>MRCTL, TMN-5, TMN Offline</td>
</tr>
<tr>
<td>ARF</td>
<td>VM5</td>
</tr>
<tr>
<td>Codec Structure</td>
<td>H.261, H.263, MACC, Baseline, HSOB</td>
</tr>
</tbody>
</table>

is intended to support visual communication, currently the predefined algorithms only include MACC, H.261 and H.263.

5.2.1 H.261

ITU-T Recommendation H.261 is an international standard aimed to provide a "video codec for audiovisual services at p×64 kbits/second". It has been implemented by many vendors and has been widely used in video-conference applications. The ability to communicate with H.261 terminals is an essential requirement for any video codec.

5.2.1.1 Application Range

Bit Rate  As defined in the Recommendation, the bit rate range for video transmission is between 40kbps and 2Mbps.

Source Format  The source coder operates on non-interlaced pictures occurring 30001/1001 (approximately 29.97) times per second. The operation is based on common intermediate format (CIF) pictures in which the luminance component has 352 pixels per line, 288 lines per picture. The two chrominance components have 176 pixels per line, 144 lines per picture each. The source coder can also operate on Quarter-CIF (QCIF) format pictures in which the number of pixels per line and the number of lines per picture are half of those in the CIF format.
5.2.1.2 Codec Structure

The diagram of the H.261 source coder is shown in Figure 5.1. It uses a macroblock based picture partition with motion compensated prediction and 2-D spatial prediction.

**Figure 5.1: Source coder of H.261**

**Motion compensation** Motion compensation is macroblock based. It uses a single motion vector for each macroblock. Both horizontal and vertical components of the motion vector have integer values not exceeding ±15. The vector is used for all four luminance blocks in the macroblock. The motion vector for both colour difference blocks is derived by halving the component values of the macroblock vector and truncating the magnitude parts towards zero to yield integer components.

**Loop Filter** The prediction process may be modified by a two-dimensional spatial filter (FIL) which operates on pixels within a predicted 8 by 8 block. The filter is separable into
one-dimensional horizontal and vertical functions. Both are non-recursive with coefficients of 1/4, 1/2, 1/4 except at block edges where one of the taps would fall outside the block. In such cases the 1-D filter is changed to have coefficients 0, 1, 0. Full arithmetic precision is retained with rounding to 8 bit integer values at the 2-D filter output. Values whose fractional part is one half are rounded up. The filter is switched on/off for all six blocks in a macroblock according to the macroblock type.

**Transformer**  A separable two-dimensional discrete cosine transform of size 8×8 is used. A detailed discussion of such transforms can be found in section 2.1.1.1.

**Group of blocks (GOB)** A GOB defined in H.261 relates to 176 pixels by 48 lines of $Y$ and the spatially corresponding 88 pixels and 24 lines of each $C_b$ and $C_r$. The arrangement of GOBs in a picture is shown in Figure 5.2.

![GOB arrangement of H.261](image)

**Bitstream Syntax** The bit stream of an H.261 coder under the RMT codec is shown in Figure 5.3. The bit stream consists of a disposable RMT head and the H.261 compatible bitstream. The full description of the H.261 bitstream syntax can be found in ITU-T Recommendation H.261 [36].

**Starting Code** The starting codes used in H.261 include Picture Starting Code (PSC), Group of Block Starting Code (GBSC) and Group Number (GN). They are defined below:
5.2: Predefined Codec Structures

- **Picture starting code (PSC)** (20 bits), 0000 0000 0000 0001 0000.
- **Group of blocks starting code (GBSC)** (16 bits), 0000 0000 0000 0001.
- **Group number (GN)** (4 bits). Four bits represent a Group number in Figure 5.2. Group numbers 13, 14 and 15 are reserved for future use. Group number 0 is used in the PSC.

Neither PSC nor GBSC+GN should be simulated to ensure correct decoding of the bit-stream.

### 5.2.1.3 Tool configuration

The following is the list of tools configured for H.261 codec (see Appendix B for acronyms):

- PCS: enabled
- CS: H.261
- SPCH: DCT
- SPCP: DCT
- MC: Basic
- VLC: Huffman
- QM: H.261/H.263
- RCTL: TMN-5, TMN offline
- ARF: None
- ME: BMA, MBMA, RHT (Integer accuracy)

The reconfigurable tools in H.261 codec only include motion estimation tools and rate control tools as they are not defined in the Recommendation. The selection of RCTL
and ME tools is made through external means. At the system level, the RMT head can be disabled so that the bitstream fully complies with the specification of the H.261 Recommendation.

5.2.2 H.263

To accommodate the requirements for low bit rate video coding, ITU-T study group 15 drafted Recommendation H.263. The basic configuration of the video source coder is based on the ITU-T Recommendation H.261. The source coder in H.263 also uses a hybrid of inter-picture prediction to utilise the temporal redundancy and transform coding of the prediction error to reduce spatial redundancy. To improve the coding efficiency, it uses half-pixel accuracy motion compensated prediction and four negotiable options to further improve the performance. The four negotiable options are: unrestricted motion vectors, arithmetic coding, advanced prediction mode and PB frames.

5.2.2.1 Application Range

**Bit Rate**  No bit rate constraints on the video bit rate are given in the Recommendation. The constraints will be given by the terminals or the network.

**Source Format**  The source coder operates on non-interlaced pictures occurring 30 000/1001 (approximately 29.97) times per second. Five standardised picture formats can be used with the source coder: Sub-QCIF, QCIF, CIF, 4CIF and 16CIF. For each of these picture formats, the luminance sampling structure is \( dx \) pixels per line, \( dy \) lines per picture in an orthogonal arrangement. Sampling of each of the two colour difference components is at \( dx/2 \) pixels per line, \( dy/2 \) lines per picture, orthogonal. The values of \( dx, dy, dx/2 \) and \( dy/2 \) are given in Table 5.2 for each of the picture formats.

<table>
<thead>
<tr>
<th>Picture Format</th>
<th>( dx )</th>
<th>( dy )</th>
<th>( dx/2 )</th>
<th>( dy/2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub-QCIF</td>
<td>128</td>
<td>96</td>
<td>64</td>
<td>48</td>
</tr>
<tr>
<td>QCIF</td>
<td>176</td>
<td>144</td>
<td>88</td>
<td>72</td>
</tr>
<tr>
<td>CIF</td>
<td>352</td>
<td>288</td>
<td>176</td>
<td>144</td>
</tr>
<tr>
<td>4CIF</td>
<td>704</td>
<td>576</td>
<td>352</td>
<td>288</td>
</tr>
<tr>
<td>16CIF</td>
<td>1408</td>
<td>1152</td>
<td>704</td>
<td>576</td>
</tr>
</tbody>
</table>

Table 5.2: Number of pixels per line and number of lines for each of the H.263 picture formats
5.2.2.2 Codec Structure

The diagram of the H.263 source coder is illustrated in Figure 5.4. Comparing Figure 5.1 and Figure 5.4, one can see that the two codecs are very similar except that no loop filter is present in H.263 source coder.

![Diagram of H.263 source coder](image)

**Figure 5.4: Source coder of H.263**

**Motion Compensation** Motion compensated prediction in H.263 is also macroblock based. The decoder will accept one vector per macroblock or if the Advanced Prediction mode of H.263 is used one or four vectors per macroblock. If the PB-frame mode is used, an additional delta vector can be transmitted per macroblock for adaptation of the forward motion vector for prediction of the B-macroblock.

Both horizontal and vertical components of the motion vectors have integer or half integer values. In the default prediction mode, these values are restricted to the range [-16, 15.5] (this is also valid for the forward and backward motion vector components for
B-pictures). In the Unrestricted Motion Vector mode however, the maximum range for vector components is \([-31.5,31.5]\), with the restriction that only values that are within the range of \([-16,15.5]\) around the predictor for each motion vector component can be reached if the predictor is in the range \([-15.5,16]\). If the predictor is outside \([-15.5,16]\), all values within the range \([-31.5,31.5]\) with the same sign as the predictor, plus the zero value, can be reached.

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the referenced picture which are spatially to the right or below the pixels being predicted. Motion vectors are restricted such that all pixels referenced by them are within the coded picture area, except when the Unrestricted Motion Vector mode and/or the Advanced Prediction mode is used.

**Transformer** The same separable two-dimensional discrete cosine transform of size 8\(\times\)8 as the one used in H.261 is used.

**Group of blocks (GOB)** Each picture is divided into groups of blocks (GOBs). A group of blocks (GOB) comprises \(k\times16\) lines, depending on the picture format (\(k = 1\) for sub-QCIF, QCIF and CIF; \(k = 2\) for 4CIF; \(k = 4\) for 16CIF). The number of GOBs per picture is 6 for sub-QCIF, 9 for QCIF, and 18 for CIF, 4CIF and 16CIF. The GOB numbering is done by means of a vertical scan of the GOBs, starting with the upper GOB (number 0) and ending with the lower GOB. An example of the arrangement of GOBs in a picture for the CIF picture format is given in Figure 5.5. Data for each GOB consists of a GOB header (may be empty) followed by data for macroblocks. Data for GOBs is transmitted per GOB in the order of an increasing GOB number. A GOB comprises one macroblock row for sub-QCIF, QCIF and CIF, two macroblock rows for 4CIF and four macroblock rows for 16CIF.

**Bitstream Syntax** The bitstream syntax of the RMT H.263 codec is very similar to that of RMT H.261 (Figure 5.3) except that a H.263 compatible bitstream is following the disposable RMT head. The full description of the H.263 bitstream syntax can be found in ITU-T Recommendation H.263 [37].

**Starting Code** The starting codes used in H.263 include Picture Starting Code (PSC), Group of Block Starting Code (GBSC), Group Number (GN) and End Of Sequence (EOS) as defined below:
5.2: Predefined Codec Structures

- **Picture starting code (PSC)** (22 bits+0-7 stuffing bits), 0000 0000 0000 0000 1 00000. All PSCs must be byte aligned.

- **Group of blocks starting code (GBSC)** (17 bits), 0000 0000 0000 0000 1. GBSC may be byte aligned by inserting a GSTUFF code of 0 to 8 zeros before GBSC.

- **Group number (GN)** (5 bits). A fixed length codeword of 5 bits. The bits are the binary representation of the number of the Group of Blocks. For the GOB with number 0, the GOB header is empty; group number 0 is used in the PSC. Group number 31 is used in the EOS and the values from 18 to 30 are reserved for future use by the ITU.

- **End Of Sequence (EOS)** (22 bits+0-7 stuffing bits), 0000 0000 0000 0000 1 11111. It is up to the encoder to insert this codeword or not. EOS may be byte aligned. This can be achieved by inserting ESTUF before the start code such that the first bit of the start code is the first (most significant) bit of a byte.

From the above discussion, it can be seen that although the diagram of H.261 source coder and H.263 source coder are very similar, the implementation details of the two codecs are quite different. The H.263 uses a different GOB structure, half pixel accuracy motion compensation and four negotiable options to achieve a higher coding efficiency and a wider
application range. The H.263 can be used from a very low bit rate video coding to HDTV applications. The application range of H.261 should be covered by H.263. Therefore, unless the terminal RMT codec is communicating with a H.261 terminal, H.263 codec should be used to achieve a higher performance.

5.2.2.3 Tool configuration

The following is a list of the tools configured for H.263 codec:

- PCS: enabled
- CS: H.263
- SPCI: DCT
- SCPD: DCT
- MC: Basic, Overlapped block
- VLC: Huffman, Arithmetic
- QM: H.261/H.263
- RCTL: TMN-5, TMN offline
- ARF: None
- ME: BMA, MBMA, RHT (Half pixel accuracy)

The reconfigurable tools in the H.263 codec include motion compensation tools, variable length coding tools, motion estimation tools and rate control tools. The selection of MC and VLC tools depends on the use of negotiable options, namely Advanced Prediction mode and Syntax-based Arithmetic Coding mode. Along with other negotiable options, they will be registered in the Picture Type information word in the picture header of H.263. This arrangement is to ensure that the RMT header can be safely discarded without affecting decoder initialisation. The selection of the ME and RCTL tools is achieved by external means.

5.2.3 Multiple-layer Affine Compensated Codec (MACC)

The MACC algorithm was discussed in detail in Chapter 3. Here we only highlight the most important parts relevant to its use in the RMT environment.
5.2.3.1 Application Range

Bit Rate The bit rate constraints in the MACC codec will be given by the terminals or the network.

Source Format MACC takes the same input source format as H.263 which has been discussed above.

5.2.3.2 Codec Structure

The codec structure of the MACC codec can be found in 3.3.

Motion Compensation MACC uses unconstrained motion vectors and Global motion compensation using affine motion models. A variable sub-pixel accuracy motion compensation is used. This is the most significant difference as compared to the H.261/H.263 codec.

Transformer The same separable two-dimensional discrete cosine transform of size $8 \times 8$ as the one used in H.261/H.263 is used.

Group of blocks (GOB) No Group of blocks level is defined in MACC. Instead, MACC uses a multiple layer structure.

Bitstream Syntax The full description of bitstream syntax can be found in 3.3.8.

Starting Code The starting code used in MACC only includes 24 bits Picture Starting Code (PSC) which has a value of 0000 0000 1111 0000 0000 1000. The PSC should be byte aligned by inserting 0 to 7 zeros before PSC.

5.2.3.3 Tool configuration

The following is a list of tools configured for MACC codec:

- PCS: enabled
- CS: MACC
- SPCI: DCT
- SPCP: DCT
5.3 RMT Codec Structures

The key to the easy construction and reconfiguration of RMT lies in the use of virtual tools and virtual codec. As we have mentioned above, each tool group has a virtual tool which lists all possible operations provided by the actual tools in that group. This feature enables us to construct a codec without consideration of the actual tools used so that the codec can easily be reconfigured by substituting virtual tools with the actual tools. We define such a codec as a virtual codec in which a combination of virtual tools is structurally put together so that certain functionalities can be fulfilled. Normally, a virtual codec should provide a codec structure, coding control mechanism and a statistical report. The virtual codec also determines the possible application range and those actual coding tools that are usable.

In the prototype, we have developed a virtual codec called Baseline codec which is based on H.263 source coder with extensions to use a wider range of compression tools provided in the RMT tool archiving unit. The Baseline codec supports the full range of tools provided in the RMT prototype tool shelf. Its structure also supports the use of Global motion compensation, 2 level spatial scalability and region of interest based multiresolution spatial compression. It has a similar application range as the H.263 with a possibly better performance on low bit rate applications with camera movement.

5.3.1 Application Range

The bit rate constraint of the Baseline codec is given by the terminals or the network. The source coder operates on non-interlaced video signal at frame rate between 25 to 30
Hz. In addition to the five standard image formats as specified in H.263, the source coder also supports rectangular pictures of arbitrary size up to 2048×2048 pixels.

### 5.3.2 Codec Structure

It uses the same codec structure as H.263 source coder. However, it differs from H.263 in the following aspects:

- Motion compensation
- Spatial compression
- Rate control
- Artifacts removal filtering
- Quantisation method for DCT coefficients
- Bitstream syntax and starting codes

**Motion Compensation** Quarter-pixel accuracy and integer-pixel accuracy may be used in quantising the motion vectors. Global motion compensation may be used (unrestricted motion vector mode must be enabled) with affine or perspective motion model.

**Spatial compression tools** Non-DCT, non-block based spatial compression tools may be used to replace the DCT. In the Baseline codec, SPCI and SPCP can use different tools to achieve the best possible performance.

**Rate Control** MACC rate control may be used to enable both temporal and spatial rescalability. A rectangular **Region Of Interest (ROI)** may be defined in the source coder so that different spatial resolutions may be used in different regions of the picture.

**Artifacts removal filtering** VM-5 ARF tool may be used to remove the block artifacts caused by coarse quantisation at low bit rate.

**Quantisation Method** Apart from H.261/H.263 quantisation method, MPEG quantisation method may be used.

**Bitstream Syntax and Starting Codes** A diagram of the bitstream syntax of the Baseline codec is shown in Figure 5.6. The abbreviations and semantics are explained in the following paragraphs.
Picture Layer  The data for each picture consists of a picture header followed by data of Group of Blocks and Spatial compression of prediction error, eventually followed by an end-of-sequence code and stuffing bits. The structure is shown in Figure 5.7. ROIP is only present if indicated by ROI. GMV is only present if indicated by GMC. TR, PQUANT, CPM, PLCI, TRB, DBQUANT etc. have the same meaning as defined in H.263 and will not be discussed here. EOS may not be present, while STUF may be present only if EOS is present.

RMT Head | TR | PTYP | PQUANT | CPM | PLCI
TRB | BDQUANT | ROI | ROIP | GMC | GMCV | GOB Layer | SPC | EOS

Figure 5.7: Structure of picture layer

- **RMT Head (Variable length).** The RMT head has been defined in the previous chapter (4.2.2.2).

- **PTYPE (11bits)** The first 8 bits of PTYPE is the same as PTYPE defined in H.263. The 9th bit is used to indicate a different spatial resolution where 1 indicates that the enhanced level (original image size) is used and 0 indicates that the base level (a quarter of original image size) is used. Bits 10 and 11 are used to indicate picture coding type: 00 - Intra, 01 - Inter, 10 - B frame, 11 - PB frame.

- **ROI (1 bit)** Indicates the status of ROI. 0 disabled, 1 enabled. ROI should be macroblock aligned.

- **ROIP (32 bits)** When ROI is enabled, the following four 8 bit words are used to state the position of ROI. The up-left and bottom-right vertices of the ROI rectangle are used. The coordinates of the up-left vertex are divided by 16 before transmission. The coordinates of the bottom-right is increased by 1 then divided by 16 before transmission.

- **GMC (2 bit)** Global motion compensation status. 2 bits are used to indicate motion model used in global motion compensation. 00 - no GMC; 01 - GMC using 2 parameter translational model; 10 - GMC using 6 parameter affine model; 11 - GMC using 8 parameter perspective motion model. When GMC is not 00, the following field will be GMV.

- **GMV (variable length).** According to GMC, 0 to 4 trajectory points are used to encode the motion parameters.

- **GOBs of base or enhanced level**

- **SCE** Spatial compression of MC error image for non-block based spatial compression techniques (Wavelet/subband).

- **EOS (24 bits)** A 24 bit word of value of 0001 0000 0000 0000 1111 1111 may be used to indicate the end of sequence.
**Group of Block Layer** The Baseline codec uses a similar group of blocks structure as the H.263. The GOB layer consists of a GOB head and data for macroblocks. A GOB may not be transmitted if all MBs are not coded (COD=1). Figure 5.8 shows the structure of the GOB layer. GSTUFF, GN and GQUANT in Baseline codec has the same definition as those in H.263 and will not be discussed here.

- GBSC (18 bits) the value is 0001 0000 0000 0000 11.
- GN (5 bits) to indicate the GOB number.
- EL (1 bit) to indicate whether GOB belongs to the base layer (EL=0) or the enhanced layer (EL=1). The EL bits will only be valid when ROI is enabled. When ROI is disabled, EL will always be set to 1.

<table>
<thead>
<tr>
<th>GBSC</th>
<th>GN</th>
<th>EL</th>
<th>GQUANT</th>
<th>MOB Layer</th>
</tr>
</thead>
</table>

Figure 5.8: Structure of GOB layer

The enhanced level GOBs only exist when ROI is enabled. The number of MOBs in the enhanced level GOBs varies with the horizontal size of ROI. The composition of the enhanced level GOBs takes k MOB lines where k=1 for sub-QCIF, QCIF, CIF format and image size up to 512×512, k=2 for 4CIF format and image size between 512×512 and 1024×1024, k=4 for 16CIF format pictures and image size between 1024×1024 and 2048×2048. The type of enhanced level GOBs has a simple relationship to spatially corresponding base level GOBs as shown in Table 5.3

<table>
<thead>
<tr>
<th>Base layer</th>
<th>Enhanced layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-GOB</td>
<td>P-GOB</td>
</tr>
<tr>
<td>P-GOB</td>
<td>B-GOB</td>
</tr>
<tr>
<td>B-GOB</td>
<td>prohibited</td>
</tr>
</tbody>
</table>

Table 5.3: Relationship between base level GOBs and enhanced level GOBs

**Macroblock Layer** The MOB layer consists of a MOB head, a MV data field and a spatial compression data field. The MOB head consists of COD, CBPC, CBPY, and DQUANT. The MV data field consists of a MV, MV2 to MV3 if INTER_AV mode is
selected, or $DMV$ if bidirectional prediction is selected. The SPC data field may or may not be present depending on the coding mode and the compression tool used.

### 5.3.3 Tool configuration

The following is a list of tools configured for the Baseline codec:

- PCS: disabled
- CS: Baseline
- SPCI: DCT, Wavelet, ZTVQ, ZTW
- SPCP: DCT, Wavelet, ZTVQ, ZTW
- MC: Basic, Overlapped block
- VLC: Huffman, Arithmetic
- QM: H.261/H.263, MPEG
- RCTL: TMN-5, TMN offline, MACC
- ARF: None, VM-5
- ME: BMA, MBMA, RHT

All reconfigurable tools are selected by external means.

### 5.4 Tool Development

The success of the RMT codec largely depends on the availability of a set of powerful compression tools and an efficient use of those tools. The backward compatibility requires that the compression tools used in constructing standard coding algorithms should be included. As the first step, the majority of tools included in the prototype codec are based on our own and public domain software. They have been wrapped with a suitable interface for reconfigurability.

According to the tools' functionality, the tool set can be classified as follows:

- Spatial compression tools
- Motion estimation tools
- Motion compensation tools
5.4: Tool Development

- Lossless compression tools
- Rate control tools
- Codec structure and algorithmic tools

In each tool set, one virtual tool and several real tools are included. In the codec structure tool, virtual tools from other sets are used to construct the virtual codec. The actual algorithm and tools used to encode a sequence is chosen either through external means or through the tool selection mechanism. In the decoder, the virtual decoder is configured by the information contained in the bit stream (visual object). Therefore, once the decoder has the correct tools, it can decode the bit stream by initialising the corresponding tool set.

5.4.1 Spatial Compression Tools

Spatial compression tools provide means to reduce the signal redundancy in the spatial domain. They can be applied both to the original signal and the motion compensated prediction error signal. The tools used in RMT include both block based as well as non-block based approaches. The following tools are currently integrated in the RMT spatial compression tool set:

5.4.1.1 DCT

The DCT used in the RMT is the same as the one used in the H.261/H.263/MPEG video compression standards. It is the primary spatial compression tool used both in constructing predefined standardised coding algorithms and the RMT codec.

5.4.1.2 Zero Tree Vector Quantisation

Zero Tree VQ used in the RMT codec is developed by Cieplinski [22]. It uses a multi-resolution codebook to achieve a high compression ratio. The experimental results show that ZTVQ performs best with highly textured images. It is used as a non-block based spatial compression tool in RMT.

5.4.1.3 Wavelet/Subband

The Wavelet/Subband tool uses a non-block based approach so that fewer block artifacts can be expected when used at very low bit rates. Two versions of the wavelet compression tools are integrated into RMT spatial compression tool set. One is the traditional approach
using Huffman coding developed by Xue[94]; the other is the zero tree based approach originally developed by Said and Pearlman[78] and modified by Cieplinski[52] to use with the RMT codec.

5.4.2 Motion Estimation Tools

In the virtual ME tool, complex motion models are supported up to second order polynomial models. The actual tools may not support all motion models but all of them should support the translational motion model up to 1/4 sub-pixel accuracy. The current ME tools include:

5.4.2.1 Block Matching Algorithm

Block matching algorithm (BMA) is one of the most commonly used ME method in video coding. The one available in the RMT codec is based on the Telenor tmm-1.6. It only supports the translational motion model.

5.4.2.2 Robust Hough Transform

Robust Hough Transform (RHT) based motion estimation algorithm is a statistic based method. The method developed by Bober [15] had also been used in MACC and our earlier approach. RHT supports translational, simplified affine (rotation + translation and scaling) and the full affine motion model.

5.4.2.3 Multiresolution Block Matching Algorithm

Until now, block matching has been the most commonly used motion estimation method in video coding. However, exhaustive search block matching algorithm (ESBMA) has a high computational load. The number of calculations required is proportional to the size of the block and the square of the search range. Several acceleration methods for BMA were proposed in the past such as a logarithm search and a three point search. In these methods, it is assumed that the error norm will increase when the search direction is away from the true motion vector and the error norm will decrease when the search direction is towards the true motion vector. Such an assumption will not be applicable in textured regions. The high frequency components will create false peaks and valleys and consequently the search may end up in a local minimum. To overcome such problems, a low-pass pre-filtering is normally required before such search approaches can be applied but even this solution can not guarantee that the global minimum will be found. For this reason, the exhaustive search is still widely used. Here we propose a multi-resolution search block
matching algorithm (MBMA) which can significantly reduce both the computational load as compared to exhaustive search and false matches caused by high frequency components. The method has a sub-pixel accuracy of 1/4 of pixels.

![Multi-resolution pyramid](image)

Figure 5.9: Multi-resolution pyramid

The first step in MBMA is to build a five level multi-resolution pyramid of the input reference image as shown in Figure 5.9. The lowest resolution is 1/4 of the original reference image and the highest resolution is 4 times the original reference image. When a lower resolution image is built, we take the average of 4 neighbourhood pixels to form the new pixel:

\[ L(i, j) = \frac{O(2i, 2j) + O(2i + 1, 2j) + O(2i, 2j + 1) + O(2i + 1, 2j + 1)}{4} \]

where \( L(i, j) \) is a pixel in the lower resolution image and \( O(i, j) \) is a pixel in the original resolution image. For a higher resolution image, 4 new pixels are created as follows:

\[
H(2i, 2j) = O(i, j) \\
H(2i + 1, 2j) = \frac{O(i, j) + O(i + 1, j)}{2} \\
H(2i, 2j + 1) = \frac{O(i, j) + O(i, j + 1)}{2} \\
H(2i + 1, 2j + 1) = \frac{O(i, j) + O(i + 1, j) + O(i, j + 1) + O(i + 1, j + 1)}{4}
\]

where \( H(i, j) \) is a pixel in the higher resolution image and \( O(i, j) \) is a pixel in the original resolution image. In this way, we first build images of half and twice the resol-
ution. Repeating the above procedure on the 0.5 and 2 times resolution images, we can build 0.25 and 4 times resolution images as well. Once the multi-resolution pyramid has been constructed, we can apply normal ESBMA on various resolutions to find motion parameters.

In normal ESBMA, a spiral search and partial error comparison is used to accelerate the search. It normally starts from (0,0) and the search range is increased gradually. In such a scheme, if two peaks happen to have the same value, the one close to (0,0) will be picked up. More importantly, if partial error exceeds the minimum recorded error, the error accumulation will be terminated and a new point will be searched so that unnecessary computation is reduced. In MBMA, the normal ESBMA is executed at the lowest resolution level. The estimated vector (x1,y1) is used as the starting point for ESBMA in the next higher resolution level. A small search range is used in the higher resolution level so that the computational load is reduced and a more steady motion field can be expected. Because of the low pass filtering effect in creating the lower resolution image and the use of ESBMA in the lowest resolution level, the false match caused by high frequency components is greatly reduced. As an estimate obtained at a lower resolution is a very good approximation in a higher resolution level, a small search area around the point estimated at the lower resolution level should give the optimal result. If we suppose that the estimation in lower resolution has an error of ±1 pixel, a 2×2 search area around the estimated vector at the higher resolution is sufficient. We can repeat such a small range ESBMA search through each resolution level until the maximum resolution level is reached. The gain in computational load is apparent. Table 5.4 gives a comparison of the computational load between ESBMA and MBMA.

<table>
<thead>
<tr>
<th>ESTIMATION ACCURACY</th>
<th>ESBMA</th>
<th>MBMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer pixel</td>
<td>256</td>
<td>24</td>
</tr>
<tr>
<td>half pixel</td>
<td>1024</td>
<td>28</td>
</tr>
<tr>
<td>quarter pixel</td>
<td>4096</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 5.4: Search number for ±16 search range

It can be seen that with increased sub-pixel accuracy, the total number of search increases exponentially for ESBMA while the number of search increases linearly for MBMA. Moreover, the total number of search steps is greatly reduced.
5.4.3 Motion Compensation Tool Box

As motion-compensated frame difference has to be calculated by the motion estimation tools, the motion compensation tool is embedded in the virtual motion estimation tool. The built-in MC tool within the virtual ME tool is a block-based motion compensation method which supports all motion models and various block sizes supported by the virtual ME tool. This tool can be substituted by independent MC tools which may not support all motion models:

5.4.3.1 Block based motion compensation

This is the basic motion compensation mode used in the motion estimation procedure to calculate transformed frame difference. It is supported by all codec structures.

5.4.3.2 Overlapped block based motion compensation

This is the one used in H.263 option F. The current implementation only supports 8x8 blocks and the translational motion model so that it is fully compatible with H.263.

5.4.4 Lossless Compression Tool Box

As is well known, arithmetic coding is always better than or equal to the Huffman coding, it is normally recommended to use arithmetic coding. However, Huffman coding is also needed when a predefined algorithm is used.

5.4.4.1 Huffman Coding

The fixed table Huffman coding adopted is a modified variant of the one used in Hung’s p64 codec[34]. Two different variable length coding tables are exploited to accommodate the requirements set by H.261 and H.263 coding standards.

5.4.4.2 Arithmetic Coding

The syntax-based arithmetic coding (SAC) improves the coding efficiency for lossless symbol coding as compared to Huffman coding. However, the fixed coding table used by SAC is based on the statistics of a large quantity of image data which may not be the optimum one for a specific sequence. To overcome this problem, one can create a specific table for the sequence being encoded and transmit the table to the receiver. The drawback of this method is that the overhead of transmitting the coding table may exceed the gain from efficient coding. Here, we propose a way of getting the coding table from coded symbols at the receiver side so that no extra overhead is required.
The codec is initialised using the SAC coding table. The symbols of the first frame are encoded using the standard SAC codec. Both on the transmitter and receiver side, the frequencies of encoded symbols are calculated. As the calculation is carried out from the output bit stream, this method is called Feedback Arithmetic Coding (FAC). When the next frame is being encoded (indicated by the presence of frame header), the following update calculation is carried out both on the transmitter and receiver side:

$$FQ_{new} = W_{old}FQ_{old} + (1 - W_{old})FQ_{fb}$$  \hspace{1cm} (5.1)$$

where $FQ_{new}$ is the new frequency to be used in the next frame arithmetic coding; $W_{old}$ is a weighting factor applied to the old frequency in the range of 0.7 to 0.9; $FQ_{old}$ is the old frequency used in SAC table and $FQ_{fb}$ is the frequency calculated from the bit stream. The value of the weighting factor influences the speed of convergence to its optimum performance and the stability of its performance. Larger values of $W_{old}$ give a more stable performance but slow convergence to the optimum performance whereas a smaller value just the opposite. If $W_{old}$ is set at 0.8, the influence of the original frequency decreases to 10% after 10 frames and the influence of the original frequency table will become 1% after 20 frames. In practical coding, this procedure only takes a few seconds to converge.

5.4.5 Rate Control Tools

The rate control tools used in the prototype codec include TMN5 rate control algorithm, the Telenor MPEG-4 Anchor rate control algorithm and MACC rate control algorithm.

5.4.5.1 TMN-5 Rate Control (TMN5)

TMN-5 rate control algorithm uses a buffer regulation mechanism to simulate a codec with a limited buffer and coding delay. It uses the following parameters to regulate the quantisation step of the DCT coefficients quantiser to achieve buffer regulation:

- Target bit rate
- Average quantisation step in the previous frame
- The number of bits spent for the previous picture
- The target number of bits for the current picture
- The frame rate
For TMN5 rate control, it is assumed that the process of encoding is temporarily stopped when the buffer is nearly full. This means that a buffer overflow and forced switching to fixed blocks will not occur. However, this also means that no minimum frame rate and delay can be guaranteed. A detailed description of the algorithm can be found in Appendix A.1.

5.4.5.2 MPEG-4 Anchor Rate Control (OFFLINE)

The OFFLINE rate control in the tmn-2.0 codec is used to create MPEG-4 Anchor bit stream. This rate control does not skip any extra pictures after the first frame, and it uses a fixed frame rate. The purpose of the rate control algorithm is to achieve the target bitrate as a mean bitrate for the whole sequence. In other words, it is a rate control method optimised for offline compression. However, the target rate may not be achieved under one or more of the following conditions:

(a) too high frame rate

(b) too low start value for the quantisation parameter

(c) the rate control is started too late

(d) the sequence encoded is too short

As the successful application of the OFFLINE rate control depends on the manual adjustment of several parameters, it is only useful for demonstration purposes. A detailed description of the OFFLINE rate control algorithm is given in Appendix A.2.

5.4.5.3 MACC Rate Control (MRCTL)

The MRCTL has been discussed in detail in Chapter 3. As in the other two schemes, the local adjustment of the parameters of the rate control scheme is also linked to the DCT spatial compression tool. However, as the global adjustment mechanism of MRCTL only relates to temporal and spatial resolution, it can be used with any other spatial compression tools as well. Because of the use of spatial scalability, the minimum frame rate can be guaranteed at the price of sacrificing the spatial resolution.

5.5 Experimental Results

The RMT prototype codec has been tested using both CIF and QCIF format sequences for the predefined codecs and the RMT codec. Various tool configurations for the RMT codec have been tested.
5.5: Experimental Results

5.5.1 Pre-defined codec structure

We first tested the predefined codec structure. Figure 5.10 gives comparative results for the H.263 implementation. The compressed file size and average PSNR are used as criteria. The comparison is between RMT H.263 and Telenor tmn-2.0 [87]. The sending of RMT header was disabled during this test. The test results show that for the predefined algorithm, the RMT codec produces similar results to those obtained by tmn-2.0, with small variations in performance when alternative motion estimation tools are used.

![Figure 5.10: Test results of RMT H.263 and TMN-2.0 H.263](image)

(a) Compressed file size  
(b) PSNR (Y)

(c) PSNR (U)  
(d) PSNR (V)
5.5.2 Baseline codec

The experiments with the baseline codec are focused on the performance comparison of different tool combinations. Among them, the spatial compression tool, motion estimation tool and variable length coding tool are tested.

5.5.2.1 Spatial compression tools

With all other parameters kept unchanged, we use different spatial compression tools to encode sequences (For still image, a comparison can be found in 2.4). Table 5.5 gives the results of the comparative study of intra frame coding for 16 CIF sequences.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>PSNR (dB)</th>
<th>Bitrate (bit/pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DCT</td>
<td>DCT-A</td>
</tr>
<tr>
<td>akiyo</td>
<td>31.80</td>
<td>32.47</td>
</tr>
<tr>
<td>bream</td>
<td>30.40</td>
<td>30.76</td>
</tr>
<tr>
<td>bus</td>
<td>26.36</td>
<td>26.64</td>
</tr>
<tr>
<td>calendar</td>
<td>23.73</td>
<td>23.91</td>
</tr>
<tr>
<td>children</td>
<td>28.20</td>
<td>28.81</td>
</tr>
<tr>
<td>coastguard</td>
<td>27.46</td>
<td>27.75</td>
</tr>
<tr>
<td>container</td>
<td>28.59</td>
<td>28.83</td>
</tr>
<tr>
<td>firel</td>
<td>32.72</td>
<td>33.42</td>
</tr>
<tr>
<td>foreman</td>
<td>29.60</td>
<td>30.26</td>
</tr>
<tr>
<td>hall</td>
<td>30.12</td>
<td>30.58</td>
</tr>
<tr>
<td>mad</td>
<td>31.99</td>
<td>32.63</td>
</tr>
<tr>
<td>news</td>
<td>29.47</td>
<td>29.92</td>
</tr>
<tr>
<td>ratna</td>
<td>29.48</td>
<td>29.86</td>
</tr>
<tr>
<td>silent</td>
<td>28.89</td>
<td>29.43</td>
</tr>
<tr>
<td>td</td>
<td>32.97</td>
<td>33.47</td>
</tr>
<tr>
<td>weather</td>
<td>26.67</td>
<td>27.02</td>
</tr>
<tr>
<td>Average</td>
<td>29.28</td>
<td>29.73</td>
</tr>
</tbody>
</table>

Table 5.5: Intra Frame Coding Results

5.5.2.2 Motion estimation and motion compensation tools

Three different motion estimation algorithms (block matching algorithm (BMA), Multiresolution block matching algorithm (MBMA) and robust Hough Transform based motion estimation algorithm (MEZRHT)) have been tested using 14 CIF sequences and 16 QCIF sequences. The tests were carried out using the Baseline codec. The results are
shown in Figure 5.11 and Figure 5.12. The global motion compensation was disabled for all sequences. The test results show that the Robust Hough Transform based algorithm gives the highest PSNR on average while the multiresolution block matching gives the lowest PSNR on average. However, the difference is very small and does not affect the visual quality of the reconstructed image. From the coding efficiency point of view, the MBMA performs the best while the robust Hough Transform based algorithm performs the worst. The computational load of the different algorithms is also compared. The

![PSNR comparison](image)

Figure 5.11: PSNR comparison of different motion algorithms

results are obtained using a Sun Sparc 20. In Figure 5.13, the required CPU time is given. It can be seen that the multiresolution block matching has the lowest average computing time whereas the block matching has the highest one. We also note that the time requirement for the robust Hough Transform based algorithm does not change significantly among different sequences.
5.5.2.3 Variable length coding tools

The variable length coding tools were tested using 14 CIF sequences. The test results are given in Table 5.6. Three coding tools are tested. The results show that arithmetic coding (AVLC) gives better results on all tested sequences compared to the two Huffman coding methods. Between two Huffman coding methods, the level-run-last 3-D coding method (HVLC3D) which is used both in the H.263 and MPEG-4 verification model (VM) outperforms the level-run plus end-of-block (EOB) signal 2-D method (HVLC2D) which is used in the H.261 coding standard on average. However, in 5 out 14 cases, the HVLC3D performs worse than HVLC2D. This indicates that the HVLC3D method still needs some refinement.

5.6 Discussion

In this chapter, we designed a prototype RMT codec according to the ideas proposed in the previous chapter. The design incorporates the tools developed in our earlier research.
Computing time comparison

Figure 5.13: Computational time comparison of different motion algorithms

Table 5.6: Bitrate comparison of different VLC tools
and some public domain software. The two key techniques for the successful integration of various tools are the virtual codecs and virtual tools. We also developed a coding system which has the ability to expand its capacity without a limit while maintaining high efficiency for the most frequently accessed tools.

Through the extensive tests carried out in the previous section, a set of optimal tool combinations for the tested sequences are derived. The comparison with the tmn-2.0 H.263 reveals that the prototype (baseline) codec is capable of delivering a higher performance on all types of sequences in terms of PSNR and bit rate.
Chapter 6

Knowledge Based Image Sequence Compression

6.1 Introduction

In the previous chapter, we studied a reconfigurable multiple tool video codec. The experimental results demonstrated that a proper combination of tools can achieve better performance than that of a fixed tool codec. However, the optimal tool combination is sequence and application dependent. An optimisation procedure known as Lagrangian Multiplier Method (see Chapter 4) may be used to obtain the optimal tool combination. Unfortunately, this method requires a very large amount of computing power even with a moderate tool set size. It also requires an extra delay to ensure that a selected tool set performs consistently for the whole sequence. Furthermore, unless the whole sequence is used in the optimisation procedure, the optimal performance can not be guaranteed.

As it is not feasible to find the best (optimal) tool set combination for a given application, we would like to find a “good” solution to the problem at a reasonable cost. Let us consider a human researcher who is doing research in image sequence compression. He will carry out many experiments on various coding tools. Through experiments, he will accumulate some experience and will know that for a given application some tool combinations will perform well and some combinations will perform poorly. When he is given a new sequence within a certain application requirement, he may be able to choose a “good” tool set to fulfill the job if he has met a similar situation before. He may sometimes make a bad decision, but he will learn from the failure and will not repeat the same mistake if he remembers what was wrong with that decision. Through sufficient experiments with success and failure, the researcher may eventually become a very experienced decision maker about the tool selection. He will choose a good tool combination for almost any
given application requirements and any sequence. This procedure is known as learning and it relies to a great extent on a priori knowledge. In this chapter, we will study the possibility of incorporating a learning capability (machine knowledge) and the ability to exploit human experiences (human knowledge) into an RMT codec so that it can choose a good (may not be the best) tool set for a given application. As a priori knowledge is being used in the codec, the approach is referred to as knowledge based image sequence coding (KISC).

The knowledge based image sequence coding is a simple combination of the RMT codec and a knowledge based decision making system as shown in Figure 6.1. The knowledge based decision making mechanism consists of a knowledge base, a learning procedure (knowledge base update mechanism), a decision making engine and a tool library as shown in Figure 6.2.

![Figure 6.1: Simple knowledge based image sequence codec](image)

The discussion will start with the structure of the knowledge base in section 6.2. The extraction and archiving of machine knowledge is studied in section 6.3. The injection of human knowledge into the knowledge base is discussed in section 6.4. Then, a knowledge assisted tool selection scheme is proposed in section 6.5. Finally, experimental results and a brief summary will be presented in section 6.6 and 6.7 respectively.

### 6.2 The structure of the knowledge base

One of the key elements of the KISC codec is the knowledge base in which past coding performance and associated tool selection and sequence classification are archived so that the
information can be used later to choose a better tool combination for a given application. The basic structure of the knowledge base consists of two parts: a coding performance database and a general knowledge database. In the coding performance database, the information entries consist of three parts: sequence classification, coding performance data, and tool configuration data. In the general knowledge database, the information entries consist of two parts: scene component and scene description.

6.2.1 Coding performance database

6.2.1.1 Sequence classification

We use texture detail and motion activity information as the key for sequence indexation. Both foreground and background areas are used. The information is described by fuzzy descriptors so that human knowledge can easily be integrated. The descriptors are listed
The structure of the knowledge base

below:

• Texture detail: low, medium, high
• Motion activity: low, medium, high
• Presence of global motion: yes, no

Therefore, the total number of descriptor combinations is 18. For every key, more than one performance data and tool configuration records are allowed.

6.2.1.2 Performance data and tool configuration data records

One of the difficulties in comparing different coding tool performances is the fact that a different coding tool may give a different PSNR and bitrate for the same subject. Such difference may sometimes appear in the form of improvement of one measurement and deterioration of another. In such circumstances, it is hard to judge which tool is superior to its counterparts. Here, we propose to use a unified measurement Unit Bitrate PSNR, noted as UBP, which is defined as:

\[ UBP = \frac{PSNR}{Bitrate} \] (6.1)

We have to point out that UBP is not a universal measurement of the coding performance. It is only valid for comparing different coding tools while the discrepancy of bitrate and PSNR is relatively small. Therefore, it is only a complementary measurement to the standard PSNR and bitrate.

The data entry in the coding performance database consists of the following fields:

• Average PSNR (Y)
• Average PSNR (U)
• Average PSNR (V)
• Average bitrate
• Unit Bitrate PSNR(UBP)
• Frame rate
• Deviation of PSNR
• Deviation of bitrate
6.2: The structure of the knowledge base

• Tool configuration data

To facilitate the data analysis, the performance data are sorted according to the bit rate used to encode the sequence.

6.2.2 General knowledge database

The purpose of the general knowledge database is to archive known knowledge about scene components which include foreground objects and background. The description of the scene component uses a multiple-entry-multiple-layer form as illustrated below:

• Active motion objects
  – Life form
    * Human
      • close up
      • head and shoulder
      • half body
      • full length
      • from distance
    * Other Life Form
      • Close Up
      • Normal
      • Long Shot
  – Motorised vehicle
    * Close Up
    * Normal
    * Long Shot

• Passive motion objects
  – Rigid objects
    * Close Up
    * Normal
    * Long Shot
  – Non-rigid objects
For each object description, two associated data fields (texture details and motion status) are attached. Because of the nature of the knowledge, an even greater ambiguity about the data is allowed. For example, one can use low to high to describe the texture details and/or motion activities of an object. Apart from texture details and motion status, motion restriction information like the maximum movement extent, the acceleration rate etc. may also be used so that the proper parameters in the tool selection can be set up.

Knowledge injection: Before or during the process of encoding, known knowledge can be injected into the codec such as: the subject is human, camera has a fixed support with a panning head and a zoom lens, the focus of the lens can be changed from 24mm to 70mm etc. With such knowledge, the codec can be configured optimally according to its environment and the application requirement.

6.3 Construction of the coding performance database

Once we have decided the structure of the knowledge base, the next step is to create the required database and collect data. For a machine knowledge database, the coding performance data can easily be obtained at the end of each coding session. However, the sequence classification needs to be manipulated so that the data can be entered into the right place. Such classification can be assisted by a human operator as discussed in the next section. As it was mentioned in the previous section, the foreground and background texture details and motion activities are used for classifying different sequences. A sequence may be classified either manually or automatically. We also noted that fuzzy descriptors are used to denote the texture details and motion activities. Here, we propose a method for an automatic sequence classification.

6.3.1 Detection of texture details

It is known that the texture detail is quantifiable by many statistical features such as standard deviation, variance etc. Many researchers have used those features to measure the texture detail of the image and use this information in their coding practice[82, 32]. However, our research showed that the correlation between simple statistical features and the texture detail was relatively weak. A typical result of relating texture detail to variance is shown in Figure 6.3 which plots the standard deviation and variance vs. bit rate for
15 cif images. It should be noted that the deviation curve has been rescaled for display purpose.

Here we propose a novel approach to characterise image properties based on frequency domain features of the image. It is known that the energy of alternating current component (ACE) of an image is closely related to its texture detail. The higher the AC energy, the harder it is to compress the image. We also know that the distribution of AC components is related to the compressibility of the image. Here, we use $ACE$ and the number of non-zero quantised AC components (NZQAC) which was introduced in Chapter 3 as the measurements representing the texture detail of an image.

To calculate $ACE$ and NZQAC, the image is firstly partitioned into $8 \times 8$ blocks and the DCT transform is applied to the partitioned blocks. The $ACE$ is defined as the average squared AC coefficients of the entire picture:

$$ACE = \frac{1}{K} \frac{1}{63} \sum_{k=1}^{K} \sum_{i=1}^{63} coef f_k[i] \cdot coef f_k[i]$$

(6.2)

where $coef f_k[i]$ is the $i$th AC component of the DCT transform of the $k$th block($coef f_k[0]$
is the DC component), K is the total number of 8×8 blocks. The $NZQAC_q$ is defined as the number of non-zero AC coefficients after quantisation using quantisation step q (q=1,2,...,31):

$$NZQAC_q = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=1}^{63} NZ(q, coeff_k[i])$$ (6.3)

where $NZ(q, coeff_k[i])$ is a threshold function defined as

$$NZ(q, coeff_k[i]) = \begin{cases} 
1, & \text{abs(coeff}_k[i]) >= 2 * q \\
0, & \text{otherwise}
\end{cases}$$ (6.4)

Table 6.1 gives $ACE$, $NZQAC_8$ and $NZQAC_{16}$ and the corresponding bit rate calculated using DCT with quantisation step $q = 8$ for 14 CIF format images. From the definition, one can see that $NZQAC_q$ represents the average code number required to encode AC components within a block and $ACE$ is the average AC energy per pixel. Figure 6.4 visualises the results shown in Table 6.1. The curves of $NZQAC_8$ and $NZQAC_{16}$ have been rescaled for display purposes. From Figure 6.4 one can see that

- Both $ACE$ and $NZQAC_q$ are proportional to the bits required to compress the image;

- $NZQAC_8$ gives almost a linear relationship with the required bit rate to compress the image at the same quantisation step;
• *ACE* and *NZQAC* appear to be complimentary to each other.

![Table 6.2: *NZQAC* and Bit Rate at quantisation step 16](image)

Further experiments show that *NZQAC* also has a very close linear relationship with bit rate calculated using quantisation step *q* = 16 (see Table 6.2). Therefore, we can assume that *NZQAC* has a linear relationship with the bit rate requirements of DCT compressed image at the same quantisation step *q*. Furthermore, we discover that the value of *NZQAC* for a given image is inversely proportional to the quantisation step *q*.

Here, we propose to use *NZQAC* as the premier spatial complexity measurement and *ACE* as the secondary texture detail measurement.

The texture detail (SC) of the image is defined as a fuzzy set \{low, medium, high\}. The relationships between the texture detail, SC, the measurement of spatial complexity, \(MSC = NZQAC_q\), and *ACE* is defined through a fuzzy membership function as shown in the following equations.

Fuzzy membership function for MSC:

\[
FMF(MSC)_{low} = \begin{cases} 
1, & MSC \leq 15 \\
2 - \frac{MSC}{15}, & 15 < MSC < 30 \\
0, & MSC \geq 30
\end{cases} \quad (6.5)
\]
Fuzzy membership function for ACE:

\[
FMF(ACE)_{low} = \begin{cases} 
1, & ACE \leq 100 \\
2 - \frac{ACE}{100}, & 100 < ACE < 200 \\
0, & ACE \geq 200 
\end{cases}
\]

(6.8)

\[
FMF(ACE)_{medium} = \begin{cases} 
ACE \left(1 + \frac{100}{ACE} \right), & ACE \leq 100 \\
1, & 100 < ACE < 200 \\
3 - \frac{ACE}{200}, & 200 \leq ACE < 400 \\
0, & ACE \geq 400 
\end{cases}
\]

(6.9)

\[
FMF(ACE)_{high} = \begin{cases} 
ACE, & ACE \leq 400 \\
2 - \frac{ACE}{200}, & 400 < ACE < 600 \\
1, & ACE \geq 600 
\end{cases}
\]

(6.10)
To calculate the texture detail for a given sequence, we use MSC and its associated fuzzy membership functions ($FMF(MSC)_{low}$, $FMF(MSC)_{medium}$, $FMF(MSC)_{high}$) as the primary measurement. If one of the membership functions has a value greater than 0.75, the corresponding texture detail content is evaluated. If none of the membership functions has a value greater than 0.75, the secondary measurement ACE and its associated fuzzy membership functions ($FMF(ACE)_{low}$, $FMF(ACE)_{medium}$, $FMF(ACE)_{high}$) are used. A fuzzy OR operation is performed to obtain the corresponding membership function, i.e., the final set of complexity can be expressed as

$$FMF_L = \begin{cases} FMF(MSC)_L, & \text{if } FMF(MSC)_L \geq 0.75 \\ FMF(MSC)_L \lor FMF(ACE)_L, & \text{otherwise} \end{cases}$$ (6.11)

$$FMF_M = \begin{cases} FMF(MSC)_M, & \text{if } FMF(MSC)_M \geq 0.75 \\ FMF(MSC)_M \lor FMF(ACE)_M, & \text{otherwise} \end{cases}$$ (6.12)

$$FMF_H = \begin{cases} FMF(MSC)_H, & \text{if } FMF(MSC)_H \geq 0.75 \\ FMF(MSC)_H \lor FMF(ACE)_H, & \text{otherwise} \end{cases}$$ (6.13)

The largest membership function $FMF$ and the corresponding texture detail will be chosen as the representative of the texture detail for the given image.

### 6.3.2 Description of motion activity

The motion activity of a sequence is measured according to the complexity of the motion and the difficulty to utilise the motion compensated prediction efficiently. The following features are taken into account:

- Severity of both global and local motion
- Smoothness of the motion field

The motion activity is rated as a fuzzy set \{high, medium and low\}. In order to quantify the complexity, we first measure the following features:

- Average absolute displacement (AAD);
- Deviations of displacement vectors (DDV).

The motion activity measurement and bits required to encode the motion vectors and motion compensation errors for 14 MPEG CIF test sequences are shown in Table 6.3. It can be noted that the number of bits required to encode motion vector ($B_{MV}$) correlates...
6.3: Construction of the coding performance database

<table>
<thead>
<tr>
<th>Sequence</th>
<th>AAD</th>
<th>DDV</th>
<th>$B_{MV}$</th>
<th>$B_{err}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>0.058</td>
<td>0.056</td>
<td>1626</td>
<td>242</td>
</tr>
<tr>
<td>Bream</td>
<td>3.69</td>
<td>22.51</td>
<td>2023</td>
<td>1707</td>
</tr>
<tr>
<td>Bus</td>
<td>13.69</td>
<td>255.56</td>
<td>2445</td>
<td>12911</td>
</tr>
<tr>
<td>Calendar</td>
<td>1.48</td>
<td>6.62</td>
<td>1624</td>
<td>16805</td>
</tr>
<tr>
<td>Children</td>
<td>0.83</td>
<td>8.76</td>
<td>1685</td>
<td>6798</td>
</tr>
<tr>
<td>Coastguard</td>
<td>2.77</td>
<td>15.96</td>
<td>2101</td>
<td>6440</td>
</tr>
<tr>
<td>Container</td>
<td>0.64</td>
<td>4.98</td>
<td>1516</td>
<td>958</td>
</tr>
<tr>
<td>Foreman</td>
<td>6.63</td>
<td>128.15</td>
<td>2818</td>
<td>2398</td>
</tr>
<tr>
<td>Hall</td>
<td>1.33</td>
<td>17.34</td>
<td>1863</td>
<td>238</td>
</tr>
<tr>
<td>Mad</td>
<td>3.05</td>
<td>57.79</td>
<td>2217</td>
<td>94</td>
</tr>
<tr>
<td>News</td>
<td>0.40</td>
<td>4.93</td>
<td>1434</td>
<td>1141</td>
</tr>
<tr>
<td>Silent</td>
<td>0.15</td>
<td>0.35</td>
<td>1563</td>
<td>109</td>
</tr>
<tr>
<td>Td</td>
<td>3.68</td>
<td>85.81</td>
<td>2894</td>
<td>2731</td>
</tr>
<tr>
<td>Weather</td>
<td>0.082</td>
<td>0.088</td>
<td>694</td>
<td>358</td>
</tr>
</tbody>
</table>

Table 6.3: Motion activity measurement and bits required to encode motion related information

with the values of $AAD$ and $DDV$. However, there is no linear relationship between $B_{MV}$ and $AAD$ as well as $DDV$. Furthermore, the number of bits required to encode motion compensated error ($B_{err}$) not only depends on the motion activity but it depends on the texture detail of the image as well. Here, we use two sets of fuzzy membership functions to roughly define the motion activity using $AAD$ and $DDV$.

Fuzzy membership functions for AAD:

\[
FMF_{low} = \begin{cases} 
1, & AAD \leq 1.5 \\
2 - \frac{AAD}{1.5}, & 1.5 < AAD < 3.0 \\
0, & AAD \geq 3.0 
\end{cases} \tag{6.14}
\]

\[
FMF_{medium} = \begin{cases} 
0, & AAD \leq 1.5 \\
\frac{AAD}{1.5} - 1, & 1.5 \leq AAD < 3.0 \\
1, & 3.0 \leq AAD < 6.0 \\
4 - \frac{AAD}{2}, & 6.0 \leq AAD < 8.0 \\
0, & AAD \geq 8.0 
\end{cases} \tag{6.15}
\]

\[
FMF_{high} = \begin{cases} 
0, & AAD \leq 6.0 \\
\frac{AAD}{6} - 1, & 6.0 \leq AAD < 12.0 \\
1, & AAD \geq 12.0 
\end{cases} \tag{6.16}
\]

Fuzzy membership functions for DDV:

\[
FMF_{low} = \begin{cases} 
1, & DDV \leq 2.0 \\
2 - \frac{DDV}{2}, & 2.0 < DDV < 4.0 \\
0, & DDV \geq 4.0 
\end{cases} \tag{6.17}
\]
6.3: Construction of the coding performance database

\[
F_{MF_{medium}} = \begin{cases} 
0, & DDV \leq 2.0 \\
\frac{DDV}{2} - 1, & 2.0 < DDV < 4.0 \\
1, & 4.0 \leq DDV < 8.0 \\
2 - \frac{DDV}{8}, & 8.0 \leq DDV < 16.0 \\
0, & DDV \geq 16.0 
\end{cases} \tag{6.18}
\]

\[
F_{MF_{high}} = \begin{cases} 
0, & DDV \leq 8.0 \\
\frac{DDV}{22.0} - \frac{1}{2}, & 8.0 \leq DDV < 32.0 \\
1, & DDV \geq 32.0 
\end{cases} \tag{6.19}
\]

The motion activity measurement TCM is defined as a fuzzy logical function of fuzzy sets \(F_{AAD}\) and \(F_{DDV}\)

\[TCM = F_{AAD} \ast F_{DDV} \tag{6.20}\]

where \(\ast\) is a fuzzy AND operation. Using Minimax rule in fuzzy logic, Equation 6.20 can be rewritten as

\[TCM = \min(F_{AAD}, F_{DDV}) \tag{6.21}\]

6.3.3 Detection of the presence of a global motion

The presence of a global motion caused by a camera movement is detected by utilising a global motion estimator. The global motion estimator, which is also used in the coding procedure, is based on block matching and Taylor expansion equations.

Here, we use six parameter affine motion model for global motion estimation:

\[
\begin{align*}
\mu(x,y) &= a_0 + a_1 x + a_2 y \\
\nu(x,y) &= a_3 + a_4 x + a_5 y
\end{align*} \tag{6.22}
\]

The motion estimation is based on a block matching algorithm (BMA) using the sum of absolute differences (SAD) as the distortion measurement. A set of sparsely sampled points is used to calculate the best matching displacement \((p,q)\). The distortion measure is calculated at the displaced samples and their surrounding 3x3 neighbours. In our experiment, 81 points are selected for CIF image sequences and the samples are evenly distributed. Using the Taylor expansion, the local energy function at a block position \((i,j)\) can be expressed by

\[
\begin{align*}
E_{i,j}(u,v) &= E_{i,j}(p_{i,j}, q_{i,j}) + (u - p_{i,j}) \frac{\partial E_{i,j}}{\partial u} + (v - q_{i,j}) \frac{\partial E_{i,j}}{\partial v} \\
&+ \frac{1}{2} (u - p_{i,j})^2 \frac{\partial^2 E_{i,j}}{\partial u^2} + \frac{1}{2} (v - q_{i,j})^2 \frac{\partial^2 E_{i,j}}{\partial v^2} + (u - p_{i,j})(v - q_{i,j}) \frac{\partial^2 E_{i,j}}{\partial u \partial v}
\end{align*} \tag{6.23}
\]
Let $S$ be a vector of the $3 \times 3$ SAD values surrounding the best matched displacement $(p, q)$, i.e.,

$$
S = (e(p-1,q-1), e(p+1,q), e(p-1,q+1), e(p,q+1), e(p+1,q), e(p+1,q+1))
$$

(6.24)

where $e()$ denotes SAD value. Using a differential operator, the partial derivatives can be expressed as:

$$
\frac{\partial E(p,q)}{\partial u} = \frac{1}{9}((-1,2,-1,2,2,-1,2,-1,2)S)
$$

$$
\frac{\partial E(p,q)}{\partial v} = \frac{1}{9}((-1,0,1,-1,0,1,-1,0,1)S)
$$

$$
\frac{\partial^2 E(p,q)}{\partial u^2} = \frac{1}{9}((-1,-1,-1,0,0,1,1,1)S)
$$

$$
\frac{\partial^2 E(p,q)}{\partial v^2} = \frac{1}{9}(1,2,1,1,-2,1,1,-2,1)S
$$

(6.25)

To facilitate a later discussion, we rewrite Equation 6.22 in a matrix form as:

$$
\begin{pmatrix}
u \\
v
\end{pmatrix} = P(x_{ij})a
$$

(6.26)

where

$$
P(x_{ij}) = \begin{pmatrix}
1 & x_{ij} & y_{ij} & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & x_{ij} & y_{ij}
\end{pmatrix}
$$

$$
a = (a_0, a_1, a_2, a_3, a_4, a_5)^t
$$

If we substitute $(u,v)$ in Equation 6.23 by Equation 6.26, the affine motion parameters can be obtained by minimising

$$
\sum_{i,j} E_{i,j}(P(x_{ij})a)
$$

(6.27)

where $x_{ij}$ denotes the centre position of the block $(i,j)$. This leads to the Euler equation

$$
\frac{\partial \sum_{i,j} E_{i,j}(P(x_{ij})a)}{\partial a} = 0
$$

(6.28)

which is equivalent to

$$
\sum_{i,j} \frac{\partial E_{i,j}(P(x_{ij})a)}{\partial a} = 0
$$

(6.29)

From (6.23), (6.26) and (6.29), the partial derivative for $a_0$ is

$$
\frac{\partial E_{i,j}(P(x_{ij})a)}{\partial a_0} = \frac{\partial E_{i,j}}{\partial u} + (u-p_{i,j}) \frac{\partial^2 E_{i,j}}{\partial u^2} + (v-q_{i,j}) \frac{\partial^2 E_{i,j}}{\partial u \partial v}
$$

$$
= u \frac{\partial^2 E_{i,j}}{\partial u^2} + v \frac{\partial^2 E_{i,j}}{\partial u \partial v} - (\frac{\partial E_{i,j}}{\partial u} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial u^2} + q_{i,j} \frac{\partial^2 E_{i,j}}{\partial u \partial v})
$$

(6.30)

$$
= 0
$$
Similarly, the partial derivatives for $a_1$ to $a_5$ can be written as

$$x_{ij}(u - \frac{\partial^2 E_{i,j}}{\partial^2 u} + v \frac{\partial^2 E_{i,j}}{\partial u \partial v}) - x_{ij}(-\frac{\partial E_{i,j}}{\partial u} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial^2 u} + q_{i,j} \frac{\partial E_{i,j}}{\partial u \partial v}) = 0 \quad (6.31)$$

$$y_{ij}(u - \frac{\partial^2 E_{i,j}}{\partial^2 u} + v \frac{\partial^2 E_{i,j}}{\partial u \partial v}) - y_{ij}(-\frac{\partial E_{i,j}}{\partial u} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial^2 u} + q_{i,j} \frac{\partial E_{i,j}}{\partial u \partial v}) = 0 \quad (6.32)$$

$$u \frac{\partial^2 E_{i,j}}{\partial u \partial v} - \frac{\partial^2 E_{i,j}}{\partial^2 v} - (-\frac{\partial E_{i,j}}{\partial v} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial u \partial v} + q_{i,j} \frac{\partial E_{i,j}}{\partial u \partial v}) = 0 \quad (6.33)$$

$$x_{ij}(u - \frac{\partial^2 E_{i,j}}{\partial^2 u} + v \frac{\partial^2 E_{i,j}}{\partial u \partial v}) - x_{ij}(-\frac{\partial E_{i,j}}{\partial v} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial u \partial v} + q_{i,j} \frac{\partial E_{i,j}}{\partial u \partial v}) = 0 \quad (6.34)$$

$$y_{ij}(u - \frac{\partial^2 E_{i,j}}{\partial^2 u} + v \frac{\partial^2 E_{i,j}}{\partial u \partial v}) - y_{ij}(-\frac{\partial E_{i,j}}{\partial v} + p_{i,j} \frac{\partial^2 E_{i,j}}{\partial u \partial v} + q_{i,j} \frac{\partial E_{i,j}}{\partial u \partial v}) = 0 \quad (6.35)$$

and therefore

$$A a - b = 0 \quad (6.36)$$

where

$$A = \sum_{i,j} \left\{ \frac{\partial^2 E_{i,j}}{\partial^2 u} U(x_{ij}) \frac{\partial^2 E_{i,j}}{\partial^2 v} U(x_{ij}) \right\}$$

$$U(x_{ij}) = \left\{ \frac{1}{x_{ij}} \frac{x_{ij}}{x_{ij}^2} \frac{y_{ij}}{y_{ij}^2} \right\}$$

and

$$b = \sum_{i,j} P(x_{ij})^2 \left( -\frac{\partial E_{i,j}}{\partial u} + \frac{\partial^2 E_{i,j}}{\partial^2 u} \frac{p_{i,j}}{x_{ij}^2} + \frac{\partial^2 E_{i,j}}{\partial u \partial v} \frac{q_{i,j}}{x_{ij} y_{ij}} \right)$$

By solving Equation 6.36, we obtain the affine coefficients as

$$a = A^{-1} b \quad (6.37)$$

To ensure the robustness of global motion estimation, we use a robust regression procedure to eliminate the local motion influence. The procedure consists of the following steps:

(a). Sparsely sample the image to obtain evenly distributed M points as the centres for block matching. Mark all M points as inliers and set the iteration counter to N.
(b). Estimate global motion parameters using only inliers;

(c). Calculate the motion compensation error energy for each point using Equation (6.23) and (6.26).

(d). Calculate the average $E_{average}$ and the standard deviation $Dev$ of the error energy for all inliers;

(e). Calculate the upper threshold and lower threshold using

$$T_{upper} = E_{average} + C_{upper}Dev$$

$$T_{lower} = E_{average} + C_{lower}Dev$$

where $C_{upper}$ and $C_{lower}$ are constants. Each constant corresponds to a threshold defining an area under the tail of a standard Normal distribution. In our implementation, $C_{upper} = 1.96$ and $C_{lower} = 0.36$. The values of $C_{upper}$ and $C_{lower}$ correspond to probabilities of 97.5% and 64% respectively.

(f). If an inlier's error energy is greater than the upper threshold, it is marked as an outlier.

(g). If an outlier's error energy is less than the lower threshold, it is marked as an inlier.

(h). Decrease the iteration counter by one; if the iteration counter is greater than zero, then goto step 2; otherwise return the estimated parameters.

The robust regression procedure significantly improves the accuracy of the global motion estimation when local motion is present. However, the computational load is much higher due to the iteration procedure.

### 6.4 Construction of general knowledge database

One of the features of the KISC is its ability to use human knowledge and experience for tool selection and parameter setting. Such a feature is achieved through the use of a general knowledge database in which human knowledge is archived. The knowledge is expressed in a coding related manner. Entries in the database consist of an object description and associated features such as texture detail, motion activity, movement restriction and etc.

The experimental database only consists of a few entries which are listed below:

- Active motion objects
6.4: Construction of general knowledge database

- Life form
  - Human
    - Close up: Texture detail - low to medium; Motion activity - low to high; Motion restriction - non-rigid motion.
    - Head and shoulder: Texture detail - low to high; Motion activity - low to high; Motion restriction - non-rigid motion.
    - Half body: Texture detail - low to high; Motion activity - low to high; Motion restriction - non-rigid motion.
    - Full length: Texture detail - low to high; Motion activity - low to medium; Motion restriction - non-rigid motion.
    - Long shot: Texture detail - low to high; Motion activity - low to medium; Motion restriction - can be treated as rigid motion object.
  - Other Life Form
    - Close up: Texture detail - low to high; Motion activity - low to high; Motion restriction - non-rigid motion.
    - Normal: Texture detail - low to high; Motion activity - low to medium; Motion restriction - non-rigid motion.
    - Long shot: Texture detail - low to high; Motion activity - low to medium; Motion restriction - can be treated as rigid motion object.
- Motorised vehicle
  - Close up: Texture detail - low to medium; Motion activity - low or high; Motion restriction - rigid motion.
  - Normal: Texture detail - low to medium; Motion activity - low to high; Motion restriction - rigid motion.
  - Long shot: Texture detail - low to high; Motion activity - low to medium; Motion restriction - rigid motion object.

- Passive motion objects
  - Rigid objects: Texture detail - low to high; Motion activity - low to high; Motion restriction - motion continuity, smooth acceleration.
  - None rigid objects: Texture detail - low to high; Motion activity - low to high; Motion restriction - local motion continuity may exist.
It can be seen that the knowledge is expressed in a quite ambiguous way. It may only be used in conjunction with the coding performance database. However, the general knowledge can be extended to include more specific entries with less ambiguity. It may even put more complex information such as shape mask for human body, motion restrictions imposed on parts of the human body. The potential use of such a database is endless.

6.5 Knowledge assisted tool selection

As we have pointed out in the introduction, it is not feasible to use an optimisation procedure in tool selection due to the computational load and permissible coding delay. We also noted that it is possible to choose a good tool combination so that the codec can perform well for a given sequence. The experience for a human researcher in tool selection is accumulated through a large number of experiments with success and failure. In this section, we propose a knowledge assisted tool selection mechanism which can interact with the user in a tool selection process. The knowledge assisted tool selection procedure consists of three steps: information gathering, database searching, and decision making.

As the first step, the information about the sequence being coded is retrieved through an interactive procedure. Firstly, the user is requested to supply all the possible known information to the codec through a questionnaire. The questionnaire consists of a multiple choice sheet and the user is only required to select one answer for every question. The purpose of the questionnaire is to supply some simple facts about the sequence such as lighting conditions, the scene composition, the camera movement etc. to the sequence classifier so that a better classification may be achieved. A typical questionnaire form is given in figure 6.5. The answers in the questionnaire are fed into the general knowledge base search engine so that statistical properties of the objects and the scene can be extracted. Second, data quantifying the texture detail, the motion activity and the presence of a global motion is computed. The data is compared with the current content of the general knowledge database and any disagreement is highlighted so that the user can decide whether to accept the classification made by the codec. Finally, the application requirement is entered by the user so that all the necessary information is gathered.

Using rules based on the past performance, the system will recommend a tool set combination for the user. The user can then accept or modify the choice made by the system. After the user has made any changes to the system recommendation, the tool combination and options are passed to the codec to start the coding procedure. The user can opt to bypass the above knowledge computation procedure to start the codec directly.
6.6: Experimental results

Various parts of the KISC codec have been tested using CIF image sequences. In this section, we will demonstrate the performance of the sequence classifier and the tool selector based on the coding results obtained by using the tool selection process.

6.6.1 Sequence Classification

Firstly, the texture classifier is tested using CIF format sequences. The test results are given in Table 6.4. In Table 6.4, \( TX \) represents the texture classification using the adopted measurements; \( FMF \) represents the fuzzy membership function value associated with the classification.

The motion activity of the above sequences is also tested. The test results are shown in Table 6.5.

Combining the results given in Table 6.4 and Table 6.5, we can obtain the sequence classification. A comparison with the original MPEG sequence class is given in Table 6.6.

Table 6.6 shows that 9 out of 13 sequences are classified correctly. For all incorrectly classified sequences, one level higher classes are assigned. However, a comparison based on the compressed file size clearly shows that the file sizes for the Container and Hall sequences are significantly higher than other class A sequences and close to some of the
class B sequences. On average, the file size of the Foreman sequence is much higher than that of the News and Silent sequences. This indicates that our classifier estimates the sequence complexity quite well. On the other hand, all hybrid natural and synthetic sequences (class E) are classified as class B sequences where their compression complexity is close to class C sequences. One of the possible reasons is that the classifier only looks
6.6: Experimental results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Texture</th>
<th>Motion Activity</th>
<th>Classification</th>
<th>MPEG Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>low</td>
<td>low</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>Bream</td>
<td>medium</td>
<td>medium</td>
<td>B</td>
<td>E</td>
</tr>
<tr>
<td>Bus</td>
<td>high</td>
<td>high</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Calendar</td>
<td>high</td>
<td>medium</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>Children</td>
<td>medium</td>
<td>low</td>
<td>B</td>
<td>E</td>
</tr>
<tr>
<td>Coastguard</td>
<td>medium</td>
<td>medium</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Container</td>
<td>medium</td>
<td>medium</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>Foreman</td>
<td>medium</td>
<td>high</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>Hall</td>
<td>medium</td>
<td>low</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>Mad</td>
<td>low</td>
<td>high</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>News</td>
<td>medium</td>
<td>medium</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Silent</td>
<td>medium</td>
<td>low</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Weather</td>
<td>high</td>
<td>low</td>
<td>B</td>
<td>E</td>
</tr>
</tbody>
</table>

Table 6.6: Sequence classification

into the first two frames for motion activities which may be misleading in this particular type of sequences.

The performance results for the global motion detector are given in Table 6.7. The

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Detected</th>
<th>Global Motion</th>
<th>Real Global Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Bream</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Bus</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Calendar</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Children</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Coastguard</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Container</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Foreman</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hall</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Mad</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>News</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Silent</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Weather</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.7: Global Motion Detection

detector has given an erroneous indication for sequence “Calendar”. The investigation shows that the scene consists of a large inconsistent local motion which leads to misclassification. For the rest of the sequences, the detector gives the correct classification of
the global motion status.

The user classification extracted from the questionnaire is given in Table 6.8. The human knowledge based classification can only be used as a guide to accurate machine classification (Currently, any mismatch will be highlighted and the user can choose which way to correct).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Texture</th>
<th>Motion Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akiyo</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>Bream</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>Bus</td>
<td>low to medium</td>
<td>low to high</td>
</tr>
<tr>
<td>Calendar</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>Children</td>
<td>low to high</td>
<td>low to medium</td>
</tr>
<tr>
<td>Coastguard</td>
<td>medium</td>
<td>medium</td>
</tr>
<tr>
<td>Container</td>
<td>low to high</td>
<td>low to medium</td>
</tr>
<tr>
<td>Foreman</td>
<td>low to medium</td>
<td>low to high</td>
</tr>
<tr>
<td>Hall</td>
<td>low to high</td>
<td>low to medium</td>
</tr>
<tr>
<td>Mad</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>News</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>Silent</td>
<td>low to high</td>
<td>low to high</td>
</tr>
<tr>
<td>Weather</td>
<td>low to high</td>
<td>low to medium</td>
</tr>
</tbody>
</table>

Table 6.8: Sequence classification by questionnaire answers

6.6.2 Tool Selection

6.6.2.1 Coding results comparison using different tools

From the test results in Chapter 5, some of the tools in the tool box can easily be identified as the preferred tools. For example, arithmetic coding delivers the best performance for all tested sequences. In this particular case, the PSNRs are identical and only bit rates are different. However, for other tools, both PSNR and bit rates are changing. Here, we use the unit bitrate PSNR (UBP) as the measurement. In Table 6.9, the INTRA frame coding results using different spatial compression tools are given using UBP (the data were converted using the data in Table 5.5). The results show that for low texture sequences with the value of UBP over 100, ZTVQ gives the best results. For medium and high texture sequences, ZTW performs better with a few exceptions. For the sequence “container”, ZTVQ is marginally better than ZTW. For “bus” and “children”, both have UBP values in the range between 60 and 65. DCT with deblocking filtering gives the best performance.
### Table 6.9: INTRA frame coding results using different spatial compression tools

<table>
<thead>
<tr>
<th>Sequence</th>
<th>DCT</th>
<th>DCT-A</th>
<th>ZTVQ</th>
<th>ZTW</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>108.5</td>
<td>110.7</td>
<td>127.7</td>
<td>124.2</td>
</tr>
<tr>
<td>bream</td>
<td>95.0</td>
<td>96.2</td>
<td>110.8</td>
<td>112.2</td>
</tr>
<tr>
<td>bus</td>
<td>60.2</td>
<td>60.9</td>
<td>50.8</td>
<td>59.93</td>
</tr>
<tr>
<td>calendar</td>
<td>36.5</td>
<td>36.8</td>
<td>30.28</td>
<td>38.9</td>
</tr>
<tr>
<td>children</td>
<td>63.7</td>
<td>65.1</td>
<td>54.8</td>
<td>63.2</td>
</tr>
<tr>
<td>coastguard</td>
<td>81.2</td>
<td>82.0</td>
<td>63.5</td>
<td>86.8</td>
</tr>
<tr>
<td>container</td>
<td>73.0</td>
<td>73.6</td>
<td>77.4</td>
<td>77.4</td>
</tr>
<tr>
<td>foreman</td>
<td>93.9</td>
<td>96.1</td>
<td>98.1</td>
<td>105.4</td>
</tr>
<tr>
<td>hall</td>
<td>80.7</td>
<td>81.9</td>
<td>87.9</td>
<td>88.5</td>
</tr>
<tr>
<td>mad</td>
<td>123.0</td>
<td>125.5</td>
<td>151.2</td>
<td>131.1</td>
</tr>
<tr>
<td>news</td>
<td>77.2</td>
<td>78.4</td>
<td>73.7</td>
<td>82.1</td>
</tr>
<tr>
<td>silent</td>
<td>93.6</td>
<td>96.9</td>
<td>81.0</td>
<td>100.5</td>
</tr>
<tr>
<td>td</td>
<td>136.6</td>
<td>138.6</td>
<td>211.0</td>
<td>142.8</td>
</tr>
<tr>
<td>weather</td>
<td>36.8</td>
<td>37.3</td>
<td>39.6</td>
<td>39.8</td>
</tr>
<tr>
<td>Average</td>
<td>82.85</td>
<td>84.29</td>
<td>89.84</td>
<td>89.49</td>
</tr>
</tbody>
</table>

The UBP is also used to measure the performance of motion estimation algorithms. Table 6.11 presents the results of a comparison of different motion estimation techniques. The results show that multi-resolution search blocking matching (MBMA) gives a better performance in 12 out of 14 sequences whereas block matching (BMA) performs better on the “bus” and “coastguard” sequences. Both block matching methods give a better performance than the robust Hough transform (MEZRHT) based method on average. The performance margin between the two block matching methods is not significant. However, on some sequences, MEZRHT does outperform BMA but the overall performance of MEZRHT is disappointing. Considering the low computational load of MBMA (see Figure 5.13), we can conclude that MBMA is the best of the tested motion estimation methods.

#### 6.6.2.2 Knowledge based tool selection

From the above experimental results, there is no need for motion estimation tools to use the tool selection procedure as MBMA gives the best performance for majority of sequences in terms UBP. Similar rules can be applied to variable length coding tools in which arithmetic coding is the preferred tool. Here, we only test the selection of spatial coding tools using the rule based (knowledge based) method. Using the results in Table 6.9, the rules used for determining spatial coding tools for INTRA frame only coding are
Table 6.10: Sequence coding results using different INTRA frame spatial compression tools

<table>
<thead>
<tr>
<th>Sequence</th>
<th>UBP-DCTA</th>
<th>UBP-ZTVQ</th>
<th>UBP-ZTW</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>860.51</td>
<td>847.31</td>
<td>875.38</td>
</tr>
<tr>
<td>bream</td>
<td>152.17</td>
<td>152.28</td>
<td>152.40</td>
</tr>
<tr>
<td>bus150</td>
<td>62.66</td>
<td>62.85</td>
<td>62.81</td>
</tr>
<tr>
<td>calendar</td>
<td>39.57</td>
<td>39.66</td>
<td>39.57</td>
</tr>
<tr>
<td>children</td>
<td>111.85</td>
<td>111.66</td>
<td>111.90</td>
</tr>
<tr>
<td>coastguard</td>
<td>85.77</td>
<td>85.76</td>
<td>85.89</td>
</tr>
<tr>
<td>container</td>
<td>310.77</td>
<td>306.66</td>
<td>309.73</td>
</tr>
<tr>
<td>foreman</td>
<td>150.65</td>
<td>150.63</td>
<td>151.07</td>
</tr>
<tr>
<td>hall</td>
<td>391.08</td>
<td>384.60</td>
<td>391.41</td>
</tr>
<tr>
<td>mad</td>
<td>647.93</td>
<td>643.39</td>
<td>652.12</td>
</tr>
<tr>
<td>news</td>
<td>320.83</td>
<td>317.71</td>
<td>318.42</td>
</tr>
<tr>
<td>silent</td>
<td>328.03</td>
<td>327.22</td>
<td>328.67</td>
</tr>
<tr>
<td>td</td>
<td>210.17</td>
<td>216.19</td>
<td>212.45</td>
</tr>
<tr>
<td>weather</td>
<td>261.33</td>
<td>261.61</td>
<td>257.88</td>
</tr>
<tr>
<td>Average</td>
<td>280.95</td>
<td>279.11</td>
<td>282.12</td>
</tr>
</tbody>
</table>

Table 6.10: Sequence coding results using different INTRA frame spatial compression tools

as follows:

- If texture is low, ZTVQ is used;
- If texture is medium or high, ZTW is used.
- If block based method is required, DCT-A is used regardless of the picture texture complexity.

When the whole sequence is involved, however, the rule should be different because of the interaction between the INTRA frame coding and motion compensated INTER frame coding. Using Table 6.10, the following rules are derived for the sequence coding:

- If texture is high, ZTVQ is used;
- If texture is low or medium, ZTW is used.
- If block based method is required, DCT-A is used regardless of the picture texture complexity.

The coding results using the tool selection based on the above rules are given in Table 6.12. The results show that the use of the knowledge based tool selection procedure can produce above average results for individual tools.
### 6.7 Summary

In this chapter, we discussed a knowledge based image sequence codec which consists of a reconfigurable multiple tool (RMT) codec and a knowledge assisted tool selection system. The tool selection system should be able to use statistical data of the sequence and the associated tool performance to give a near optimal tool configuration. The key element of the system is the sequence classifier. A fuzzy logic reasoning system and a statistical feature based texture and motion activity classifiers have been developed. We also developed a global motion detector based on a global motion estimation algorithm which uses block matching and the Taylor expansion to derive affine motion parameters. The performance of the classifier on our test sequences proved to be satisfactory. The ability to archive the coding performance data and associated tool configuration of the codec is also a valuable and powerful means for future research and development.

In order to use the data in the coding performance database, we propose to use a unified bit rate and PSNR measurement called unit bitrate PSNR (UBT). The use of UBT successfully solved difficulty of evaluating different coding tools giving different PSNR and bit rates. Using the available resources, we have created a sample knowledge assisted tool selection system. The experimental results prove that the idea is realistic and the performance of the codec using the knowledge assisted tool selection is above average.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>UBP-BMA</th>
<th>UBP-MBMA</th>
<th>UBP-MEZRHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>596.1</td>
<td>606.1</td>
<td>591.9</td>
</tr>
<tr>
<td>bream</td>
<td>108.8</td>
<td>112.0</td>
<td>86.7</td>
</tr>
<tr>
<td>bus150</td>
<td>35.3</td>
<td>35.2</td>
<td>29.0</td>
</tr>
<tr>
<td>calendar</td>
<td>33.5</td>
<td>33.7</td>
<td>31.9</td>
</tr>
<tr>
<td>children</td>
<td>85.5</td>
<td>86.1</td>
<td>81.2</td>
</tr>
<tr>
<td>coastguard</td>
<td>67.9</td>
<td>67.6</td>
<td>50.4</td>
</tr>
<tr>
<td>container</td>
<td>232.0</td>
<td>251.8</td>
<td>242.1</td>
</tr>
<tr>
<td>foreman</td>
<td>112.6</td>
<td>114.0</td>
<td>92.3</td>
</tr>
<tr>
<td>hall</td>
<td>261.8</td>
<td>286.4</td>
<td>274.1</td>
</tr>
<tr>
<td>mad</td>
<td>373.6</td>
<td>418.8</td>
<td>377.9</td>
</tr>
<tr>
<td>news</td>
<td>231.9</td>
<td>234.9</td>
<td>221.4</td>
</tr>
<tr>
<td>silent</td>
<td>237.9</td>
<td>242.6</td>
<td>231.8</td>
</tr>
<tr>
<td>td</td>
<td>170.9</td>
<td>177.1</td>
<td>168.5</td>
</tr>
<tr>
<td>weather</td>
<td>190.3</td>
<td>191.5</td>
<td>180.3</td>
</tr>
<tr>
<td>Average</td>
<td>108.7</td>
<td>110.2</td>
<td>95.8</td>
</tr>
</tbody>
</table>

Table 6.11: Sequence coding results using different motion estimation algorithms
<table>
<thead>
<tr>
<th>Sequence</th>
<th>UBP-SEL-INTRA</th>
<th>UBP-SEL-INTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>akiyo</td>
<td>127.7</td>
<td>875.38</td>
</tr>
<tr>
<td>bream</td>
<td>112.2</td>
<td>152.40</td>
</tr>
<tr>
<td>bus150</td>
<td>59.93</td>
<td>62.85</td>
</tr>
<tr>
<td>calendar</td>
<td>38.9</td>
<td>39.66</td>
</tr>
<tr>
<td>children</td>
<td>63.2</td>
<td>111.9</td>
</tr>
<tr>
<td>coastguard</td>
<td>86.8</td>
<td>85.89</td>
</tr>
<tr>
<td>container</td>
<td>77.4</td>
<td>309.73</td>
</tr>
<tr>
<td>foreman</td>
<td>105.4</td>
<td>151.07</td>
</tr>
<tr>
<td>hall</td>
<td>88.5</td>
<td>391.41</td>
</tr>
<tr>
<td>mad</td>
<td>151.2</td>
<td>652.12</td>
</tr>
<tr>
<td>news</td>
<td>82.1</td>
<td>318.42</td>
</tr>
<tr>
<td>silent</td>
<td>100.5</td>
<td>328.67</td>
</tr>
<tr>
<td>td</td>
<td>142.8</td>
<td>212.45</td>
</tr>
<tr>
<td>weather</td>
<td>39.8</td>
<td>261.61</td>
</tr>
<tr>
<td>Average</td>
<td>91.17</td>
<td>282.40</td>
</tr>
</tbody>
</table>

Table 6.12: Coding results using tool selection

However, due to the limited number of tools available for experiments, the strength of the knowledge assisted tool selection can not be fully exploited. With the development of video coding techniques, we believe that more and more new coding tools and codec structures will be made available to suit the growing demand for better and more efficient visual communications. This will provide sufficient room for KISC to improve itself.
Chapter 7

Conclusions

In this thesis, we firstly reviewed the recent rapid development of digital visual telecommunications and the associated data explosion problem. Possible solutions based either on video coding standards or on non-standard data compression techniques were then discussed. The advantages and disadvantages as well as the application potential of the current video coding standards were evaluated.

From the study of the coding standards, a handful of common techniques used by all coding standards were extracted. We concluded that the two most important concepts used by all coding standards are the macroblock structure and the hybrid prediction codec structure. A numerical evaluation of various waveform based compression tools revealed that different compression tools perform best in different applications. The observation prompted the idea of a reconfigurable codec which can adapt itself to specific application requirements.

First of all, a multiple layer affine motion compensated codec was proposed and adopted as a basic building block of the reconfigurable multiple tool video codec. A detailed investigation of the properties of the multiple layer affine motion compensated codec was carried out. We then developed a framework for the reconfigurable multiple tool video codec. The framework was realised in terms of a prototype into which the various coding tools developed in our earlier research were integrated. The test results showed that it outperformed one of the H.263 implementations (tmn-2.0) available in the public domain.

To solve the problem of optimising the codec configuration for any given sequence, we proposed to use a knowledge based approach. Human knowledge inferred by means of a questionnaire was injected into the codec. The coding performance of the various coding tools was archived and analysed so that a set of rules guiding the tool selection could be derived. The experimental results showed that the use of a knowledge-assisted tool selection procedure can achieve an above average performance as compared with
individual fixed set tools.

7.1 Contributions

In the thesis, the following contributions to knowledge based image sequence coding have been made:

- By reviewing the current video coding standards it has been established that the differences between them are very small. The analysis has shown that it should be possible to develop a codec to cover the whole range of applications by a close integration of the elements of the coding standards.

- The study of widely used compression techniques revealed that some compression techniques outperform others for certain types of applications and vice versa.

- A multiple layer codec structure has been proposed. It is capable of using an inner block segmentation technique to achieve a better coding efficiency for motion vector coding.

- A probabilistic background/foreground segmentation method with heuristic thresholding which gives satisfactory results for separating the foreground objects from their background has been developed.

- A background memory reference updating scheme used for improving the coding efficiency of uncovered background regions has been devised.

- An extended quadtree decomposition procedure supporting the use of background motion compensation, dual motion inner block segmentation and background memory reference updating has been proposed.

- A pseudo-Huffman code which improves the coding efficiency of adjacent block positions has been developed.

- Two methods for inner block motion boundary coding (straight line approximation and fixed pattern partition) have been devised and investigated.

- A bit rate control method capable of regulating the first INTRA frame coding by using NZQAC measurements has been conceived and implemented. We discovered that the average code length used for the INTRA DCT coefficients coding did not change much among different sequences and it varied with the quantisation step. A
set of equations encapsulating the relationship between the average code length and quantisation step has been derived from experimental results.

- A framework for a reconfigurable multiple tool video codec has been proposed and designed. The bit stream syntax and the codec structure make the proposed codec capable of using various codec structures and coding tools while maintaining the compatibility with the existing coding standards.

- The concept of virtual codec and virtual tool are the major innovative contributions behind the development of the reconfigurable multiple tool codec. The use of a virtual codec makes it possible to reconfigure the codec structure on the fly whereas the use of virtual tools makes it possible to switch between coding tools during the coding process.

- An expandable variable length code is developed for encoding the tool update data field. The code combines efficiency with flexibility. The expandability of the code guarantees the maximum flexibility of the system.

- A prototype codec realising the idea of reconfigurable multiple tool codec has been implemented to demonstrate its practical potential. Several codec structures have been integrated into the prototype codec. The experimental results showed that the prototype codec is as efficient as a dedicated codec when only standard DCT based techniques are used. With the use of some specialised coding tools on certain types of sequences, the prototype codec outperformed the standard codec (H.263).

- A knowledge based tool selection system has been studied and developed. During its development, a sequence texture classifier, a motion classifier, and a global motion detector has been designed. The rule based tool selection system that uses the output of the above classifier is illustrated to give above average performance.

- A questionnaire based knowledge injection mechanism has been developed with a graphical interface to the reconfigurable codec prototype.

### 7.2 Future Work

The proposed reconfigurable video codec framework and the development of the prototype codec provide a good basis for a further development. Here we give a list of possible extensions to the current work.
7.2: Future Work

7.2.1 Development of new coding tools

Due to the limited time and human resources, the tool set of the current codec is far from complete. Many new ideas and algorithms could be incorporated as new coding tools or codec structures. The following is a possible list of the candidates:

- Fractal compression
- New motion estimation algorithms
- New motion compensation methods
- Codec structure to support the H.263+ coding standard
- Codec structure to support segmentation based coding
- Codec structure to support 3-D motion compensation

7.2.2 Tool selection and knowledge injection

Further experiments are need to establish a more complete and robust knowledge database for tool selection. For the knowledge injection, a more complex model could be developed to take full advantage of the knowledge concerning moving objects and their surrounding environment.

7.2.3 Potential web applications

The fundamental structure of the reconfigurable codec makes it a good alternative for video retrieving through world wide web. A Java based decoder would be ideal to serve this purpose.

7.2.4 Standardisation effort

The flexibility and expandability of the reconfigurable codec make it an ideal candidate for the next generation video compression standards. The ability to use multiple compression tools and multiple codec structures will enable more sophisticated compression techniques to be used in codecs which may have to meet contradictory requirements.
Appendix A

TMN Rate Control Algorithms

Two types of rate control algorithms are utilised in the TMN H.263 codec. The first one is used for real-time adjustment and online rate control. It is suitable for use in real time communication. The second one is used for offline rate control and is suitable for use in file compression. The details of both rate control algorithms are discussed in the following.

A.1 TMN-5 Rate Control (TMN5)

For realistic simulations with a limited buffer and coding delay, a buffer regulation is needed. Mechanisms for regulating the output bitrate are:

- Changing the step size at the macroblock level.
- Buffer regulation:
  - The first intra picture is coded with quantiser parameter $QP = 16$. After the first picture the buffer content is set to:
    \[
    \frac{R}{f_{\text{target}}} + 3 \frac{R}{FR}
    \]  
    \[\text{(A.1)}\]
    and
    \[B_{i-1} = \bar{B} \]  
    \[\text{(A.2)}\]
  - For the following pictures the quantiser parameter is updated at the beginning of each new macroblock line. The formula for calculating the new quantiser parameter is:
    \[
    QP_{\text{new}} = Q\bar{P}_{i-1}[1 + \frac{\Delta_{1B}}{2\bar{B}} + 12\frac{\Delta_{2B}}{R}],
    \]
    \[
    \Delta_{1B} = B_{i-1} - \bar{B},
    \]
    \[
    \Delta_{2B} = B_{\text{imb}} - \frac{mB}{MB}\bar{B}
    \]  
    \[\text{(A.3)}\]
where:

- \( Q_{P_{i-1}} \) is the mean quantiser parameter for the previous picture,
- \( B_{i-1} \) denotes the number of bits spent for the previous picture,
- \( \bar{B} \) is the target number of bits per picture,
- \( mb \) indicates the present macroblock number,
- \( MB \) is the number of macroblocks in a picture,
- \( B_{imb} \) is the cumulative number of bits spent,
- \( R \) is the bitrate,
- \( FR \) is the frame rate of the source material (typically 25 or 30 Hz),
- \( ft_{\text{target}} \) is the target frame rate and
- \( f_0 \) is a parameter relating \( ft_{\text{target}} \) and \( Q_{P_{i-1}} \) (Default value = 10).

The first two terms of this formula are constant for all macroblocks within a picture. The third term adjusts the quantiser parameter during the coding of the picture. The frame rate \( ft_{\text{target}} \) and new \( \bar{B} \) are calculated at the start of each frame:

\[
\bar{B} = \frac{R}{ft_{\text{target}}}
\]

The calculated \( Q_{P_{\text{new}}} \) must be adjusted so that the difference does not exceed the definition range. The buffer content is updated after each complete frame in the following way:

\[
Buffer_{\text{content}} = \frac{R}{ft_{\text{target}}} + 3 \frac{R}{FR}
\]  

(A.4)

For each of the remaining frames:

\[
Buffer_{\text{content}} = Buffer_{\text{content}} + B_{\text{total}}
\]

while (\( Buffer_{\text{content}} > 3 \frac{R}{FR} \)) {

\[
Buffer_{\text{content}} = Buffer_{\text{content}} - \frac{R}{FR};
\]

\( frame_{\text{incr}} + 1; \)

}

The variable \( frame_{\text{incr}} \) indicates the number of frames to be skipped.

For TMN5 rate control, it is assumed that the process of encoding is temporarily stopped when the buffer is nearly full. This means that buffer overflow will not occur. However, this also means that no minimum frame rate and delay can be guaranteed.
A.2 MPEG-4 Anchor Rate Control (Offline)

The Offline rate control in tmn-2.0 codec is used to create MPEG-4 Anchor bit stream. This rate control does not skip any extra pictures after the first frame, and it uses a fixed frame rate. It is possible to start the rate control after a certain percentage of the sequence has been encoded with a fixed quantisation parameter. Its purpose is to achieve the target bitrate as a mean bitrate for the whole sequence. In other words, it is a rate control method optimised for offline compression.

The offline control starts with the following known parameters:

- The total number of seconds of video signal $S_{ec}$
- The number of frames left to be encoded $Frame_{left}$
- The target bitrate $B$
- Quantisation step used for previous frame $QP$
- The total number of bits used $Buf$
- The number of bits used for previous frame $bits_{pre}$

With the knowledge of the above parameters, the offline rate control can be stated as:

(a). Set $NewQP = QP$

(b). Calculate the bits left for rest of the sequence

$$Buf_{rest} = Sec \times B - Buf$$

(c). If $Frame_{left} = 0$ then stop; else the bits for each frame left is calculated

$$Bits_{restofpicture} = Buf_{rest}/Frame_{left}$$

(d). The possible adjustment of quantisation step is calculated using

$$\Delta QP = MAX(1, QP \times 0.1)$$

(e). The adjustment decision is based on the bits used for the previous frame:

$$IF \ (bits_{pre} > Bits_{restofpicture} \times 1.15) \ NewQP = \ MIN(31, QP + \Delta QP) \ IF$$

$$IF \ (bits_{pre} < Bits_{restofpicture} / 1.15) \ NewQP = \ MAX(1, QP - \Delta QP)$$
If one uses the offline rate control, one will risk not achieving the target rate under one or more of the following conditions:

(a). too high frame rate  
(b). too low start value for the quantisation parameter  
(c). the rate control is started too late  
(d). the sequence encoded is too short

As the successful application of Offline rate control depends on the manual adjustment of several parameters, it is only useful for demonstration purpose.
Appendix B

Glossary

A
ACE Alternative current energy
ARF Artifact removal filtering

B
BMA Block matching algorithm

C
CCIR The International Radiocommunication Consultative Committee (Now known as ITU-R)
CD Compact disc
CIF Common intermediate format, it has 352 columns and 288 lines per picture and a frame rate of 30 000/1001 frames/second.
CS Codec structure

D
DCT Discrete cosine transform
DFD Displaced frame difference
DPCM Differential pulse-code modulation
APPENDIX B. GLOSSARY

E
EOB End of block
EOS End of sequence
ESBMA Exhaustive search block matching algorithm

G
GOB Group of blocks
GBSC GOB starting code
GN Group number

H
HDTV High definition television

I
IFS Iterated Function System
ISDN Integrated service digital network
ISO International Standardisation Organisation
ITU International Telecommunication Union

J
JPEG Joint Picture Expert Group

K
KISC Knowledge based image sequence coding
KLT Karhunen - Loeve Transform

M
MACC Multiple layer affine motion compensated codec
MAD Mean absolute frame difference
MBMA Multiresolution search block matching algorithm
MC Motion compensation
ME Motion estimation
MOB Macro block
MPEG Moving Picture Expert Group
MSE Mean square error

NZQAC Non-zero quantised alternative current coefficient

PCS Predefined codec structure
PSC Picture starting code
PSNR Peak signal to noise ratio
PSTN Public switching telephone network

QCIF Quarter common intermediate format
QSIF Quarter standard interchange format
QM Quantisation method

RCIT Rate control
RHT Robust Hough Transform
RMT Reconfigurable multiple tools
RSC RMT starting code
RTF RMT tool flag field
RVQ Residual vector quantization

SAC Syntax based arithmetic coding
SAD Sum of absolute difference
SIF Standard interchange format. It is a format for exchanging video images of 240 lines with 352 pixels each for NTSC, and 288 lines by 352 pixels for PAL and SECAM. At the
nominal field rates of 60 and 50 fields/s, the two formats have the same data rate.

SPCI Spatial compression for INTRA frame coding
SPCP Spatial compression for INTER frame coding

T

TMN Test model near term

U

UBP Unit bitrate PSNR

V

VLC Variable length coding
VQ Vector quantization
VSBM Variable size block matching

Z

ZTVQ Zero-tree vector quantization
ZTW Zero-tree wavelet coding
Bibliography


