5-1-1998

Localized temporal decorrelation for video compression

Dinesh Nadarajah

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Localized Temporal Decorrelation for Video Compression

Dinesh Nadarajah

April 14, 1998
Title of Thesis:  Localized Temporal Decorrelation for Video Compression

I, Dinesh Nadarajah, hereby grant permission to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature: ______________________

Date : 4/14/98
Acknowledgements

I would like to thank Prof. R. Unnikrishnan without whose help and understanding this degree would not have been possible. I would also like to thank Ajit Bopadikar for his advise and support when I need them most; not to mention those long walks. I would also like to acknowledge some of my friends who have helped me in various ways. Mohan Dhandapani, Arati Naik, and Vasuki Seshadri for helping me with various corrections. Mandar Chati for both proof reading my thesis and helping me learn LaTeX.
Abstract

Many of the current video compression algorithms perform analysis and coding operations in a block-wise manner. Most of them use a motion compensated DCT algorithm as the basis. Many other codecs, mostly academic and in their infancy and known as Second Generation techniques, utilize region and contour based and model based techniques. Unfortunately, these second-generation methods have not been successful in gaining widespread acceptance in both the standards and the consumer world. Many of them require specialized computationally intensive software and/or hardware. Due to these shortcomings, current block based methods have been fine-tuned to get better performance at even very low bit rates (sub 64 kbps).

Block based motion estimation is the principal mechanism used to compensate for motion between frames in an image sequence. Although current algorithms are fast and quite effective, they fail in compensating for uncovered background areas in a frame. Solutions such as hierarchical motion estimation schemes do not work very well since there is no reference in past, and in some cases, future frames for an uncovered background resulting in the block being transmitted as an intra frame (which requires the most bandwidth among all type of blocks). This thesis introduces an intermediate stage, which compensates for these isolated uncovered areas. The intermediate stage uses a localized decorrelation technique to reduce frame to frame temporal redundancies. The algorithm can be easily incorporated into existing systems to achieve an even better performance and can be easily extended as a scalable video coding architecture. Experimental results show that the algorithm, used in conjunction with motion estimation, is quite effective in reducing temporal redundancies.
3.2.1 Example 1: Simple Illustration ........................................... 24

3.2.2 Example 2: Hierarchical Extension ................................. 24

3.3 Summary .............................................................................. 26

4 Experimental Results and Analysis ........................................ 27

4.1 Introduction ........................................................................ 27

4.2 Extended Boundary Search ................................................. 28

4.3 Localized Temporal Decorrelation ..................................... 31

5 Software Implementation ....................................................... 51

5.1 Implementation Issues and Considerations .......................... 51

5.2 Matlab ............................................................................... 52

6 Conclusions and Future Research .......................................... 53

A A Review of Digital Video Compression ................................. 54

A.1 Digital Video Representation ............................................. 54

A.2 Redundancies and Irrelevancies ......................................... 55

A.3 Video Compression Systems ............................................. 56

A.4 Current Standards in Digital Video Compression .................... 60

B Motion Estimation ................................................................. 63
B.1 Temporal Redundancy Reduction ........................................... 63
B.2 Motion in Video ................................................................. 63
B.3 Motion Estimation Techniques ............................................. 64
B.4 Block Based Motion Estimation ............................................. 65
   B.4.1 Block Matching Criteria .................................................. 66
   B.4.2 Search Strategy .............................................................. 67
   B.4.3 Block Size ................................................................. 71
B.5 Example of Block Based Motion Estimation and Compensation ... 71
B.6 Problems with Block Based Motion Estimation ......................... 77

C Quantization and Human Visual System Considerations ................ 78

D Subband Image Coding - A Brief Outline ................................. 81

E Zero Tree Coding Algorithm ................................................. 86

F Matlab Code ........................................................................... 90
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Video as an image sequence</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Delta modulation based decorrelation</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Frame 1 of Claire image sequence</td>
<td>6</td>
</tr>
<tr>
<td>2.4</td>
<td>Frame 2 of Claire image sequence</td>
<td>6</td>
</tr>
<tr>
<td>2.5</td>
<td>Error between frame 1 (fig. 2.3) and frame 2 (fig. 2.4)</td>
<td>7</td>
</tr>
<tr>
<td>2.6</td>
<td>Frame to frame relationship when using motion estimation</td>
<td>7</td>
</tr>
<tr>
<td>2.7</td>
<td>Simplified block diagram of compression system using motion estimation and compensation</td>
<td>8</td>
</tr>
<tr>
<td>2.8</td>
<td>Prediction error image after motion estimation and compensation</td>
<td>9</td>
</tr>
<tr>
<td>2.9</td>
<td>8-by-8-by-8 video cube for XYZ compression</td>
<td>12</td>
</tr>
<tr>
<td>2.10</td>
<td>3D-Subband breakdown of video signal [Vet]</td>
<td>13</td>
</tr>
<tr>
<td>2.11</td>
<td>Frame grouping for temporal subband filtering</td>
<td>14</td>
</tr>
<tr>
<td>2.12</td>
<td>Critical subband decomposition and ME/MC along temporal axis L : Low frequency frames, D : High frequency (detail) frames</td>
<td>14</td>
</tr>
</tbody>
</table>
3.1 Decision making process involved in current popular block based video codecs ........................................ 18

3.2 Proposed intermediate stage ................................................................. 21

3.3 Simple illustration of uncovered background ................................. 24

3.4 Hierarchical temporal decorrelation .................................................. 25

3.5 Pyramidal representation of decorrelation coefficients ...................... 25

4.1 Flower Garden frame zero padded for extended boundary motion estimation and compensation ............................... 28

4.2 Performance comparison of boundary extended motion different motion estimation algorithms for Flower Garden image sequence .................. 29

4.3 Timing performance of motion estimation algorithms for Flower Garden image sequence ................................. 30

4.4 Typical coefficient profile for one level decorrelation (same block size as motion estimation) for Flower Garden image sequence .................. 32

4.5 Comparison of different block sized localized decorrelation techniques on Flower Garden image sequence ........................................ 33

4.6 (a) Block categories for Flower Garden sequence with motion estimation, (b) Block categories for Flower Garden sequence with motion estimation and localized decorrelation ........................................ 35

4.7 Frame 1 of Susie image sequence .......................................................... 36

4.8 (a) Block categories for Susie sequence with motion estimation only, (b) Block categories for Susie sequence with motion estimation and localized decorrelation .............................. 37
(a) Original Flower Garden image sequence frame number 30

(b) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation only

(c) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation and 16x16 single level block localized decorrelation

(d) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation and 4x4 single level block localized decorrelation

(e) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD)

(f) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD)

(g) PSNR curves for 30 frame Flower Garden test sequence

(a) Original Susie image sequence frame number 30

(b) Susie sequence frame number 30 reconstructed with motion estimation and compensation only

(c) Susie sequence frame number 30 reconstructed with motion estimation and compensation and 16x16 single level block localized decorrelation

(d) Susie sequence frame number 30 reconstructed with motion estimation and compensation and 4x4 single level block localized decorrelation
4.10 (e) Susie sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD) ................. 45

4.10 (f) Susie sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD) ................. 45

4.10 (g) PSNR curves for 30 frame Susie test sequence ................. 46

4.11 (a) Original Football image sequence frame number 30 ............... 47

4.11 (b) Football sequence frame number 30 reconstructed with motion estimation and compensation only ......................... 47

4.11 (c) Football sequence frame number 30 reconstructed with motion estimation and compensation and 16x16 single level block localized decorrelation ......................... 48

4.11 (d) Football sequence frame number 30 reconstructed with motion estimation and compensation and 4x4 single level block localized decorrelation 48

4.11 (e) Football sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD) ......................... 49

4.11 (f) Football sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD) ......................... 49

4.11 (g) PSNR curves for 30 frame Football test sequence .................. 50

A.1 Video as an image sequence .................................................. 55

A.2 Closed loop video compression system ...................................... 56
A.3 GIF compressed lena image - file size = 72498 bytes ............... 58
A.4 JPEG compressed (75% quality) image - file size = 11482 bytes ... 58
A.5 Simplified block diagram of a motion compensated closed loop DCT
                   based video coder ........................................ 62
B.1 Simple motion illustrations ........................................ 64
B.2 Translatory motion vector illustration ............................. 64
B.3 Full search method ................................................... 67
B.4 Three step search points ............................................ 68
B.5 Cross search algorithm ............................................... 69
B.6 Square search algorithm ............................................. 70
B.7 Frame 1 of flower garden image sequence ............................ 72
B.8 Frame 2 of flower garden image sequence ............................ 72
B.9 Motion vectors generated by blocked based full search method ... 73
B.10 Prediction error image obtained using full search method ....... 74
B.11 Motion vectors generated by blocked based cross search method ... 75
B.12 Prediction error image obtained using cross search motion estimation 76
C.1 Uniform (linear) quantizer input/output function ................. 79
D.1 Analysis and synthesis filter banks ................................ 82
D.2 A two level hierarchical signal decomposition ..................... 82

viii
D.3 A two level hierarchical image decomposition filter bank system . . . . 83

D.4 Three level subband decomposition of Claire image . . . . . . . . . . 84

E.1 Parent child relationship used in Zero Tree Algorithm . . . . . . . . 87

E.2 Subband priority scanning order . . . . . . . . . . . . . . . . . . . . . . 88

E.3 Significance mapping algorithm flowchart . . . . . . . . . . . . . . . 88

E.4 Encoding algorithm flow chart . . . . . . . . . . . . . . . . . . . . . . 89
Chapter 1

Introduction

In recent years the demand for multimedia information has grown at a rapid pace. With the ever increasing popularity of the Internet and the *World Wide Web* (WWW) and visually oriented interactive applications, the use and manipulation of multimedia content has been pushed to the forefront. Multimedia information contain audio-visual (audio and video) and data signals. Examples of multimedia information are CD-ROM based computer/video games and interactive television terminals. Because of the interactive nature of multimedia information, a large portion of multimedia content is occupied by the digital video signal component.

Video is the most attractive component in multimedia signals. Since it relays information that directly appeals to the human eye, many applications are being extended with video for the changing consumer market. Examples of such cases are video telephony and video electronic mail (video-email). Streaming of live audio and video over packet switched networks has also become an important area of research. Vigorous activities involving application and communications technology development and standardization are also being undertaken to facilitate rapid deployment of video based communications.

Digital video signals require a large amount of bandwidth and storage space for transmission and archiving. Due to the limitations of channel capacities and storage technologies, video signals must be compressed. Without efficient and effective compression techniques, video signals cannot be employed for any practical applica-
tions. Applications such as low bit rate video transmission over standard telephone lines, content based video library search facilities, and high quality custom configurable video and audio services cannot be implemented without compression due to the demand in bandwidth and processing power.

The storage and transmission requirements imposed by multimedia signals has resulted in intense research in compression and storage technologies and efficient transcoding (dynamic rate conversions for bandwidth change) and transmission rate control systems. Since much of the multimedia bandwidth is occupied by video, most of the research effort has been focused on compressing video signals.

Most video compression systems (Appendix A) achieve compression by first reducing the temporal correlation and then decorrelating the spatial redundancies. Hence to achieve good overall performance, an effective temporal redundancy reduction technique must be incorporated. The temporal decorrelation operation is the most critical component in any video compression system.

Many of the temporal decorrelation techniques currently in use are either compute intensive or complicated or do not perform very well for all types of video sequences (image dependent methods). One of the biggest problems with current block based temporal redundancy reduction techniques is that they are not efficient in compensating for isolated uncovered background regions. This thesis proposes a technique to extend current block based algorithms so as to overcome this short coming. The proposed method is low in complexity and thus would not be at the computational expense of existing implementations. The method can be implemented as a hierarchical algorithm for scalable video architectures.

Chapter 2 reviews some of the currently existing and proposed temporal decorrelation techniques and highlights the pros and cons of each method. Chapter 3 details the proposed scheme and outlines it’s advantages over the systems described in Chapter 2. Experimental results are presented in Chapter 3 and conclusions and future system improvements are discussed in Chapter 4. Many topics that are indirectly related to this thesis are attached as appendices to this report and referenced frequently.
Chapter 2

A Review of Temporal Decorrelation Techniques

2.1 Introduction

As illustrated in figure 2.1, digital video can be generated by displaying successive still images (frames) in time (refer to Appendix A). Since digital video is a representation of continuous events in time, individual frames are highly correlated with adjacent frames. Many objects that appear in one frame also appear in previous or subsequent frames after undergoing some form of a motion transformation. Rotation, translation, and zoom are a few examples of this transformation. Hence decorrelation techniques can be used to remove temporal redundancies (unnecessary information).

There are two popular methods of accomplishing temporal decorrelation. Spatial domain based predictive decorrelation methods are currently the most commonly used techniques and have been incorporated into international video coding standards such as the Motion Pictures Experts Group (MPEG) and the International Telecommunications Union (ITU). Block based motion estimation and compensation algorithms (Appendix B) fall under this category. Transform based methods attempt to decorrelate the temporal pixel values by applying transformations like the Discrete Cosine Transform (DCT) [Ahm] or Subband Transforms [Vet] along the temporal axis also.
This chapter reviews some of the existing and proposed temporal decorrelation methods and highlights the advantages and disadvantages of each of them.

### 2.2 Delta Modulation

The easiest way to decorrelate frames in time is to use the previous frame as a prediction for the current frame. This is similar to a *Differential Pulse Code Modulation* (DPCM) ([Ger]) system with a prediction filter of

\[
H(z) = z^{-1}.
\]  

(2.1)

In this method the difference between the current frame and the previously coded frame, also known as the *Displace Frame Difference* (DFD), is appropriately quantized (Appendix C) and compressed for transmission and/or storage. The closed loop delta modulation system is shown below in figure 2.2.

Just as in one dimensional (1D) delta modulation scheme, for this method of temporal decorrelation to work efficiently, the temporal sampling rate (frame rate) must be high. That is, there must be very little motion between adjacent frames in time. If there is a lot of motion between frames, then the prediction would be poor.
and hence more bits would be necessary to encode the prediction error. Figures 2.3 and 2.4 show two temporally adjacent original frames. The direct error between these two frames are shown in figure 2.5. The variance of this error frame is 40.4.

2.3 Decorrelation with Motion Estimation

Motion estimation techniques can be used to perform temporal decorrelation between frames in an image sequence. Unlike the Delta Modulation scheme, motion estimation and compensation techniques generate a prediction frame based on the motion information between two frames. The temporal relationship between frames in a typical frame to frame motion estimation based system (like MPEG and the 'H' series algorithms from ITU) is given in figure 2.6. A simplified block diagram of a compression system using motion estimation and compensation for temporal decorrelation is shown in figure 2.7.

Block based motion estimation techniques are by far the most popular method of motion estimation. Both the MPEG and the ITU video compression standards employ some form of block based algorithm for temporal decorrelation [Rao],[Tek],[Has]. Block based schemes attempt to describe the motion of a block of region from one frame to another based on a disparity criterion such as the Mean Squared Error (MSE). The motion information (motion vectors) obtained as a result of motion prediction must also be transmitted for the decoder to be able to perform motion compensation. This is similar to transmitting the filter coefficients in a DPCM based...
Figure 2.3: Frame 1 of Claire image sequence

Figure 2.4: Frame 2 of Claire image sequence
Figure 2.5: Error between frame 1 (fig. 2.3) and frame 2 (fig. 2.4)

Figure 2.6: Frame to frame relationship when using motion estimation
Figure 2.7: *Simplified block diagram of compression system using motion estimation and compensation*

compression scheme. Appendix B contains detailed reviews and examples of various block based motion estimation principles and algorithms.

Figure 2.8 illustrates the prediction error obtained after block based motion estimation was used to compensate for the motion between the two frames in figures 2.3 and 2.4. The full search method along with a block size of 16 pixels was used and the search area was limited to a 15-by-15 square region. The MSE threshold criterion was set to zero. The variance of the prediction error was calculated to be 2.85 (compared to 6.36 for the simple error frame in figure 2.5). It can be easily seen that motion estimation performs very well in decorrelating temporal motion and thus would aid in achieving a better compression performance.

One of the major disadvantages of the block-based schemes is the computational resource necessary to achieve good temporal decorrelation. The above compensation example (figure 2.8) required 303,551,240 *floating point operations* (flops) to achieve the given result. Many fast algorithms have been proposed but they still are compute intensive. Hence, motion estimation and compensation consume a large percentage of the computational resources of a video compression system. However they have a lower computational load when compared with image segmented object based motion estimation and model based techniques.
Figure 2.8: Prediction error image after motion estimation and compensation
Despite these drawbacks, the success and popularity of the block-based methods can be attributed to its simplicity and ease of hardware implementation. Since the algorithm is based on the reduction of a global distortion criterion, it performs very well with most types of image sequences, natural or synthetic (computer generated).

### 2.4 Three Dimensional Decorrelation Techniques

Three dimensional (3D) transform based techniques attempt to achieve spatial and temporal decorrelation using a single transformation. The transformation is usually, but not necessarily, applied to the temporal axis first followed by the spatial axis. Two of the popular 3D transforms are extensions of popular 2D versions of the DCT and subband transforms. Both these transforms have been extensively studied and applied to still image coding (Appendices D and E).

#### 2.4.1 3D-DCT Video Coding

Since the DCT is a separable transform [Rao], the one dimensional forward DCT transform

\[
X^{C_2}(m) = C(m) \sum_{n=0}^{N-1} x(n) \cos \left( \frac{m(2n+1)}{2N} \right), m = 0, 1, \ldots, N - 1 \tag{2.2}
\]

can be applied in a successive manner to each of the three axes to obtain the 3D-DCT transformation. The three dimensional, N point, forward transform is given by ([Wes])

\[
F(u, v, w) = C(u) \cdot C(v) \cdot C(w) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \sum_{z=0}^{N-1} f(x, y, z) \cdot \cos \left( \frac{(2x + 1)u\pi}{16} \right) \cdot \cos \left( \frac{(2y + 1)v\pi}{16} \right) \cdot \cos \left( \frac{(2z + 1)w\pi}{16} \right) \tag{2.3}
\]
where,

- $x, y, z$ are index pixels in pixel space,
- $f(x, y, z)$ is the value of a pixel in pixel space,
- $u, v, w$ are index pixels in the DCT space,
- $F(u, v, w)$ is the transformed pixel value in DCT space, and $C(i) = \frac{1}{\sqrt{2}}$ for $i = 0$ and $C(i) = 1$ for $i > 0$.

The inverse N point transform (3D-IDCT) is given by

$$
\tilde{f}(x, y, z) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \sum_{w=0}^{N-1} C(u) \cdot C(v) \cdot C(w) \cdot F(u, v, w) \\
\cdot \cos \left( \frac{(2x + 1)u\pi}{16} \right) \cdot \cos \left( \frac{(2y + 1)v\pi}{16} \right) \cdot \cos \left( \frac{(2z + 1)w\pi}{16} \right). \tag{2.4}
$$

The above mentioned 3D-DCT based transform has been used in a video compression algorithm similar to the JPEG still image coding method. The algorithm is called the XYZ Compression Algorithm [Wes]. In this scheme, the video stream is divided into groups of 8 frames. Each frame in the group is divided into 8-by-8 blocks forming an 8-by-8-by-8 video cube as illustrated in figure 2.9.

Each cube is then independently transformed using the 3D-DCT and then quantized and coded. A thorough treatment of this method and related algorithmic and architectural implementation issues are also addressed in [Wes].

Since the forward 3D-DCT and the 3D-IDCT are identical, the XYZ compressor and decompressor are similar. That is the encoder and decoder is symmetric. This is one of the major advantages of this scheme. Very good quality video at high compression ratios (of the order of 1:75) has been reported with this algorithm. The biggest drawback with this method is that eight frames have to be stored in memory both at the encoder and the decoder and all eight frames must be decoded to view
any one of them. This is undesirable since it limits the speed of real time spatial domain search and indexing capabilities. Because of the 8-by-8-by-8 video cube based coding nature of the algorithm, functionality such as real time frame rate conversion (temporal scaling), frame skipping, and trick modes (fast forward, slow motion, etc.) are limited.

2.4.2 3D-Subband Coding (3D-SBC)

Like the DCT, 2D-SBC techniques used in still image compression (Appendix E) have been extended as 3D-SBC algorithms to code video signals [Gle1, Gha]. Multi-resolution representations are obtained for the video signal in the temporal as well as the spatial axes. Figure 2.10 illustrates this procedure where the temporal filtering is done first followed by the spatial filtering - although this could be done in any order.

Although subband transform based methods have been successfully used in 1 dimensional and still image coding and spatial decorrelation in video ([Cla], [DeV], [Oht], [Rao], [Sai], [Sha]), it has not been widely used to decorrelate data along the temporal axis. This is mainly because temporal subband decomposition using long filter banks introduces artificial artifacts in the high pass bands and smear motion.
information in the low pass bands [Vet]. The artificial artifacts introduced in the analysis phase, appear as visually unpleasant distortions when decompressed. If 3D subband techniques are to be employed in video coding, then only short filters (such as the Haar filter) must be used for temporal decorrelation. Also long temporal filters give rise to additional frames due to linear convolution.

2.4.3 3D-SBC with Motion Estimation

Shortcomings of the pure 3D subband techniques have lead researchers to combine such techniques with motion estimation discussed earlier [Kim1]. First the input frame sequence is divided into groups of frames. The length of the temporal subband filter determines the number of frames in each group (if circular convolution is to be used for filtering). Figure 2.11 illustrates the frame grouping for a four tap filter.

The frames in each group are treated as samples in a one-dimensional signal and are subjected to critical subband decomposition. This would result in all but one frame containing the low frequency information (figure 2.12).
Figure 2.11: Frame grouping for temporal subband filtering

Figure 2.12: Critical subband decomposition and ME/MC along temporal axis L: Low frequency frames, D: High frequency (detail) frames
The first low frequency information frame will be coded as a still image (*intra frame*) while subsequent low frequency frames will be motion estimated using the previous one (figure 2.12). Essentially, the low frequency information frame will substitute the actual frames in the closed loop compression system illustrated in figure 2.7.

The main advantage of this method is that it reduces the number of motion estimations per second in an image sequence and hence reduces the computational load significantly. If a four tap filter is used as the temporal decorrelation filter, then for a 32 frames per second image sequence only seven motion estimations have to be performed compared to 31 with the simple direct frame to frame motion estimation method discussed earlier. In low bit rate applications such as video telephony, this is a significant reduction in computational load.

Since subband decomposition smears motion information, using the low frequency band for motion estimation purposes could result in poor motion field predictions. The loss of motion information is severe when long filters are used since the number of frames averaged over time is increased. Another disadvantage with this method, as mentioned earlier, is that long filters introduce artifacts into the high frequency band frames. Because of this, only short filters must be used for temporal decorrelation. This reduces benefits of using subband based decorrelation.

2.5 Summary

This chapter highlighted the importance of motion estimation and compensation methods in temporal decorrelation. Since motion estimation algorithms are compute intensive, dual decorrelation techniques and the like must be resorted to in order to reduce the number of motion estimation operations. Three dimensional subband and DCT based systems are either too complex or introduce unwanted artifacts. Subband decomposition with motion estimation has been proposed as an alternative to frame to frame motion compensation. But motion estimation with a averaged frame result in poor predictions and do not lend very well to frame content search techniques. The following chapter introduces a localized temporal decorrelation technique as an exten-
sion to current block based methods. The proposed system is both computationally simple and can be easily incorporated into existing video coding architectures.
Chapter 3

Localized Temporal Decorrelation in Spatial Domain

As discussed in chapter 2, good temporal decorrelation is an essential component in any video compression system. Since spatial decorrelation is performed on the temporal decorrelation residuals, the first phase will affect the overall compression system performance. By overall system performance, we mean not only the quality of the video but also the computational load, compression ratio, scalability, and other practical factors.

Current popular video compression systems use block based methods to perform temporal decorrelation. In many standardized and popular video codecs, discussed in Appendix A, after block based temporal decorrelation, DCT is used to perform spatial decorrelation. Usually motion estimation is performed using either a 16-by-16 block or an 8-by-8 block. The resulting prediction error is spatially decorrelated using the DCT on an 8-by-8-block basis. The decision-making process involved in this is illustrated by the flow chart in figure 3.1.

Despite their success, block based motion estimation and compensation techniques are too compute intensive. This would not affect high bandwidth applications where very accurate motion compensation is not necessary, but at low bit rates, it could introduce undesirable blocking artifacts and coding delays. On the other hand, motion estimation cannot be completely eliminated because good motion compensation is
Figure 3.1: Decision making process involved in current popular block based video codecs
critical for system compression performance and there are no replacement methods that produce comparable performance and functionality.

Although block based motion estimation algorithms provide good temporal decorrelation, they do not perform well in predicting motion along frame edges. Also motion compensation of hidden surfaces (such as a hidden background uncovering) is poor. To overcome the first problem, boundary extended search mechanisms have been suggested and used successfully (this has been incorporated into the new ITU-H.263 very low bit rate video-coding standard). Boundary extended search systems search for a matching block beyond the boundary of a frame.

Hidden surfaces are regions found in one frame but are not present in the previous frame. To overcome this problem, bi-directional motion estimation algorithms and hierarchical block wise algorithms have been utilized. In bi-directional motion estimation algorithms, a frame is predicted using the past and the future frames. In hierarchical block based methods, block based motion estimation algorithms are repeatedly applied using smaller block sizes. Both these remedies suffer from the fact that increased numbers of operations and system (hardware) requirements are needed for more sophisticated motion estimation algorithms. In the case of bi-directional motion compensation systems, a frame in the future, say \( k + 5 \), must be transmitted before \( k + 1, k + 2, \) etc. are transmitted.

A problem with hierarchical motion estimation scheme is that after the first level of estimation, only the hidden backgrounds remain mostly uncompensated. In many cases it is not possible to find a suitable match for it in another frame and hence repeated motion estimation will only contribute to system complexity and coding delays.

A common problem with the classical motion compensated DCT algorithm is that the DCT component is very compute intensive and does not lend very well to hierarchical coding. Hierarchical spatial coding is preferred in low bit rate systems where equal priority must be given to all parts of an image frame. i.e. some amount of information is transmitted for a coarse approximation of the entire frame and then, where necessary, the finer details are transmitted.
Due to the above-mentioned limitations in current coding systems, other means must be resorted to for better temporal decorrelation performance. The following section develops a temporal decorrelation system that employs localized hierarchical redundancy reduction along with motion compensation for superior performance. It will be shown that this system offers superior decorrelation performance at lower computational cost and can be easily incorporated into existing systems.

### 3.1 Proposed System

In this section we develop a simple extension to the current popular block based video codecs. As noted earlier, block based motion estimation schemes are the most popular method of temporal decorrelation. But repeated or hierarchical application of motion compensation only increases coder complexity. Once one level of motion compensation has been completed, the remaining temporal prediction error is mainly due to uncovered background or edge motion.

The proposed method inserts an intermediate stage into the coding algorithm shown in figure 3.1 (see figure 3.2). The proposed method attempts to reduce the mean squared error between an original image block and a motion compensation based predicted block using a decorrelation coefficient. In the following derivations, bold letters are used to represent matrices. If $B_o$ and $B_p$ are the original and predicted image blocks respectively, then the prediction error matrix is given by

$$E_p = B_o - B_p.$$  \hspace{1cm} (3.1)

The aim of the proposed method is to scale the motion based predicted block $B_p$ by a factor $n$ such that the error between $B_o$ and $n \cdot B_p$ is further reduced. Hence the new error is given by

$$E_p = B_o - n \cdot B_p.$$  \hspace{1cm} (3.2)
Figure 3.2: Proposed intermediate stage
The norm-squared value of this error is given by

\[ e_p^2 = ||E_p||^2 = < E_p, E_p > \]  \hspace{1cm} (3.3)

which when expanded results in

\[ e_p^2 = < B_o - n \cdot B_p, B_o - n \cdot B_p > \\
= < B_o, B_p > - 2 \cdot n \cdot < B_o, B_p > + n^2 \cdot < B_p, B_p > \]  \hspace{1cm} (3.4)

Since the aim is to reduce the squared error as much as possible, the first derivative of \( e_p^2 \) with respect to \( n \) must be set to zero in order to find the theoretical minimizing value of \( n \). i.e.

\[ \frac{d}{dn} (e_p^2) = 0. \]  \hspace{1cm} (3.5)

Therefore,

\[ \frac{d}{dn} (e_p^2) = -2 \cdot < B_o, B_p > + 2 \cdot n \cdot < B_p, B_p > = 0. \]  \hspace{1cm} (3.6)

Solving the above equation produces

\[ n = \frac{< B_o, B_p >}{< B_p, B_p >}. \]  \hspace{1cm} (3.7)

The above equation can be rewritten as

\[ n = \frac{< B_o, B_p >}{||B_p||^2}. \]  \hspace{1cm} (3.8)
where $\|B_p\|$ is the L2 norm of $B_p$. Since $n$ is a scalar value, the summation equation for generating $n$ is given by

$$
n = \frac{\sum_{r} \sum_{c} B_o(r,c) \cdot B_p(r,c)}{\sum_{r} \sum_{c} B_p(r,c) \cdot B_p(r,c)} \tag{3.9}
$$

where $r$ and $c$ are the row and column counters respectively.

Therefore the original region, $B_o$, can be reconstructed from the predicted region $B_p$ as $B_o = n \cdot B_p$ where $n$ is given by equation 3.9.

It can be seen from the above equations for $n$ that the denominator is always positive, i.e. $\|B_p\|^2 \geq 0$. The denominator, for a general case, can take any value between positive infinity and negative infinity. In the case of an image, where all luminance (image information) and chrominance (color information) values are real and positive, the inner product of $B_o$ and $B_p$ is greater than zero. i.e.

$$
0 \leq <B_o, B_p> \leq \infty. \tag{3.10}
$$

Therefore, when dealing with images, $n$ is always greater than or equal to zero. The average values of both the numerator and the denominator are in the range of 0 to $255^2$ (assuming 8 bit gray scale level representation). For computational purposes, an infinitesimal value of the order of $10^{-16}$ is added to the denominator to prevent division by zero.

The value of $n$ can also be used as a measure of similarity between the original and predicted blocks. If the two blocks are similar then $n$ would be equal to 1.
3.2 Example by Illustration

3.2.1 Example 1: Simple Illustration

Consider the following scenario. The original block in frame 1 in figure 3.3 is white (scale value of 255) and the corresponding block in frame 2 is gray (scale value of 128). This is a good example of a region uncovered by foreground movement. No amount of motion estimation and compensation will minimize the error (because the color gray is not found anywhere in the previous frame). Such an effort will only increase the coder complexity and introduce unwanted coding delays. On the other hand, localized temporal decorrelation would require just one parameter value of approximately 1.99 would reduce the error to zero perfect reconstruction.

3.2.2 Example 2: Hierarchical Extension

The localized temporal decorrelation can be extended to reduce temporal redundancy hierarchically. This is illustrated in figure 3.4. At level one, $16 - by - 16$ blocks are decorrelated. This results in a single scalar value. The resulting prediction can be further divided into $8 - by - 8$ blocks and new decorrelation scalar values can be
Figure 3.4: *Hierarchical temporal decorrelation*

Figure 3.5: *Pyramidal representation of decorrelation coefficients*

computed. The second process results in four more coefficients.

The process illustrated in figure 3.4 can be repeated to generate finer approximations. The scalar values can be used to construct a pyramid of scalar redundancy reduction coefficients (figure 3.5) for each block. Such a representation is advantageous for low bandwidth applications where a coarse approximation is made to reconstruct the block/image and then a finer approximation is made, given the computational and transmission time (this is also known as spatial scalability). Such mechanisms can be used in a block-wise manner also. i.e. repeated redundancy reduction could be applied to those blocks with a high degree of error so as to obtain a finer error approximation whereas only a single or no approximation might be necessary for some blocks.
3.3 Summary

This chapter developed a localized decorrelation method of removing temporal redundancy. The advantage of this method is low computational complexity and a scalable architecture. Also since the decorrelation is done in the spatial domain the method can be easily implemented in software for real-time applications and can be incorporated (as illustrated in figure 3.2) into existing transform based coding systems as an additional step. Chapter 4 will present some experimental results highlighting some of the advantages and performance gains obtained using the localized temporal decorrelation algorithm.
Chapter 4

Experimental Results and Analysis

4.1 Introduction

This chapter illustrates the improved temporal decorrelation performance obtained using the localized temporal decorrelation technique. The chapter first illustrates the improved performance obtained by extending the region of search along the boundaries for motion estimation and compensation and continues to provide results of various other experiments which illustrate the performance gain obtained with the proposed algorithm.

In all the experiments the Mean Squared Error (MSE) or the Peak Signal to Noise Ratio (PSNR) has been used as a measure of performance. Since the MSE and the PSNR are inversely proportional, one measure would suffice and the two are used interchangeably. The expressions for the MSE and the PSNR are given below.

\[
MSE = \frac{1}{N \cdot M} \sum_{n=1}^{N} \sum_{m=1}^{M} (B_o - B_p)^2
\]  

(4.1)

\[
PSNR = 20 \cdot \log \left( \frac{255^2}{MSE} \right)
\]  

(4.2)
where $B_o$ and $B_p$ are the N-by-M original and predicted image blocks respectively. The PSNR equation is based on the assumption that the maximum index value in the image is 255 and is measured in decibels (dB).

### 4.2 Extended Boundary Search

This section illustrates the motion compensation performance obtained by extending the search region around the boundaries of the reference image or frame. By extended boundaries we mean the mechanism of enlargement of the frame dimensions. All experiments conducted for this thesis extend frames by padding the boundaries with a fixed number of zeros. Figure 4.1 shows a zero padded Flower Garden frame.

Figure 4.2 illustrates the superior performance of the extended region search method. The figure illustrates the MSE values obtained with different motion es-
Figure 4.2: Performance comparison of boundary extended motion different motion estimation algorithms for Flower Garden image sequence

timation schemes. The full search method, the cross search method and the square search method are used with no boundary extensions. It can be easily seen that by simply extending the boundary produces results similar to (and at times better than) those obtained with the full search method. It should be noted that the extended boundary search was done with the square search criterion. Even with this logarithmic method, the error performance is comparable to the full search method.

At this point the choice of algorithm might be brought into question. If a fast algorithm such as the square search method can generate results comparable to the full search method with an extended boundary, then would not a full search method with the extended boundary generate even better results? The answer is yes. The full search algorithm can be used with the extended boundary criteria, but the computational load would be significantly higher.
Figure 4.3: *Timing performance of motion estimation algorithms for Flower Garden image sequence*

Figure 4.3 illustrates the time taken in seconds for motion compensation by each of the methods used in the earlier experiment. The full search method is not shown here because it takes an average of 6 minutes to perform motion compensation between two adjacent frames. Inclusion of this data would totally obscure the timing results of the other methods - furthermore it is not important for a frame by frame comparison due to the wide margin of difference (6 minutes compared to a few seconds). Hence the choice of algorithm would depend on the application (real time or off-line) and the speed of implementation and architecture (hardware based, multi-threaded, etc.) available. For example, the full search method might be used to distribute video on a DVD-ROM or a CD-ROM or in delayed broadcasting applications such as tape, edit, store, and broadcast systems. Real time systems might have to resort to one of the fast algorithms due to coding delay constraints.

It can be seen from the above results that for a small increase in estimation time,
the extended square search method generates superior performance compared to the other fast motion compensation methods. All localized decorrelation experiments are based on this technique.

4.3 Localized Temporal Decorrelation

One of the principal motivations for the localized decorrelation method is that repeated motion estimation is useless in uncovered background cases and that transform based decorrelation technique are too compute intensive.

When one frame is motion compensated with another, in the ideal, we would like each block to match the one in the past frame. i.e. the decorrelation coefficient would be one - at least in most cases. This is clearly illustrated in figure 4.4. Motion estimation was done with 16-by-16 blocks and the same block size was used for decorrelation (by one level).

It can be seen from figure 4.4 that motion estimation has been quite successful in predicting the temporal motion. Most of the blocks have a decorrelation value of approximately 1. The spikes seen on the graphs indicate blocks with bad motion compensation or uncovered background areas. Better decorrelation performance can be obtained by using smaller blocks for motion estimation, which translates to more computational load and transmission data.

Instead of performing repeated hierarchical motion compensation with smaller block sizes, we can use the localized decorrelation process developed in chapter 3 to reduce some of the redundancies further. Figure 4.5 compares the performance of one level localized decorrelation with different block sizes. All motion estimation and compensation has been performed with 16-by-16 blocks. As discussed before, the 16-by-16 block decorrelation does not gain much of a performance margin, but the smaller blocks do illustrate a significant reduction in redundancies. It can be easily concluded that the localized decorrelation mechanism works very well in compensating for uncovered background regions and reducing overall temporal redundancy.
Figure 4.4: Typical coefficient profile for one level decorrelation (same block size as motion estimation) for Flower Garden image sequence
Figure 4.5: Comparison of different block sized localized decorrelation techniques on Flower Garden image sequence
The real advantage of the localized decorrelation system is that it can help reduce the number of intra-blocks encoded in the classical Motion Compensated DCT methods, thus reducing the total number of bits and enabling higher compression ratios. For example, if a decision boundary is used to determine if a block is inter or intra (such as the flow diagram illustrated in chapter 3), then the localized decorrelation algorithm can be used to convert some of the intra-blocks to intermediate ones which require the transmission of only a motion vector and a few decorrelation coefficients and may be a small number of DCT coefficients (inter blocks).

The bar chart in figure 4.6 (a) shows the distribution of different block categories for each frame in a sequence. The sequence used is the Flower Garden sequence. If the mean squared error between the reference block and the original block is less than a pre-determined value (say about 0.1% of the maximum mean squared error) (category 1), then nothing needs to be transmitted. If the above condition fails then motion estimation is performed. If the prediction is good, i.e. the error is now less than a preset limit, only the motion vector has to be transmitted (category 2). Now if the error is greater than a second limit then the block would have to be transmitted as an intra-block (category 4) else as an inter block (category 3).

Now if the proposed localized decorrelation is used in the above algorithm, it can be seen from figure 4.6 (b) that some of the blocks that need to be transmitted as inter or intra blocks require the transmission of only a few coefficients. In this figure 4-by-4 blocks were used to decorrelate blocks that did not meet the decision criterion. It should be noted that the decorrelation technique does not only convert on inter blocks to coefficients but also converts some intra blocks into inter blocks.

As always is the case, many of the results obtained in video coding algorithms are image/content dependent. Since Flower Garden sequence is generated by camera pan motion and given the large number of detail content in the images, this sequence would require a large number of additional information. i.e. most of the blocks are either inter or intra blocks. This can be compared with a simple head and shoulder "Susie" video sequence, frame 1 of which is illustrated in figure 4.7. Figures 4.8 (a) and 4.8 (b) illustrate block distribution results obtained with the Susie image sequence. The effectiveness of the algorithms is again highlighted by the bar charts. Since the Susie image sequence has relatively low motion, the motion estimation obtained with
Figure 4.6: (a) Block categories for Flower Garden sequence with motion estimation, (b) Block categories for Flower Garden sequence with motion estimation and localized decorrelation
the block-based method is very good.

The above experiments were conducted with the assumption of an original reference frame for each future frame. But this is not the case in most instances. Often the reconstructed frames are used as reference images for future frames. The following experimental results illustrate that even under such conditions, localized temporal decorrelation works quite well.

Figures 4.11, 4.12 and 4.13 illustrate comparative results obtained using the "Flower Garden", "Susie", and the "Football" test image sequences. In each of the figure sets, the last of the original frame, i.e. frame number 30 is illustrated in subfigure (a). Subfigure (b) illustrates the 30th frame obtained using only 16-by-16 block motion estimation and compensation. As discussed earlier, it can be clearly seen from case (b) that motion compensation would be unsuccessful in areas where hidden regions are exposed.

Subfigures (c) and (d) show resulting images from motion compensation and decorrelation. (c) is obtained after a 16-by-16 block-wise decorrelation and (d) after a 4-by-4 block-wise decorrelation. As expected, finer resolution decorrelation results in improved image quality. In both these cases, no residual error was added to the reconstructed frames.
Figure 4.8: (a) Block categories for Susie sequence with motion estimation only, (b) Block categories for Susie sequence with motion estimation and localized decorrelation
Subfigures (e) and (f) are generated by adding quantized version of the residual error frame. Case (e) was reconstructed with 1 bit quantization and (f) with 2 bit quantization. The improvement in quality can be easily seen from these images. The perceptual (subjective) results are verified by the PSNR curves (objective results) for these image sequences (subfigures (f)).

The effectiveness of a decorrelation technique is best measured by comparing the compression performance of the algorithm. Based on an image independent, non-optimized quantization, and a fixed symbol bit representation, the compression ratios for each of the above cases were determined. The results are tabulated below. In computing the compression ratio, 12 bits were assigned to each motion vector and 5 bits to each decorrelation coefficient. It should be noted that these compression figures do not contain entropy coding of various parameters such as motion vectors, quantized data, and decorrelation coefficients. If such techniques are incorporated, then an additional compression factor of 5 to 15 may be obtained. Motion fields and decorrelation coefficients are highly correlated and hence can be very efficiently compressed. Using DCT to compress the residual error frame would further improve the compressing efficiency. The human visual system parameters can also be incorporated to remove irrelevant information.

<table>
<thead>
<tr>
<th>Case No.</th>
<th>Total Bits</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Original sequence</td>
<td>18,432,000</td>
<td>1.00</td>
</tr>
<tr>
<td>(b) Sequence after ME/MC only</td>
<td>704,400</td>
<td>26.17</td>
</tr>
<tr>
<td>(c) After 16x16 decorrelation</td>
<td>749,400</td>
<td>24.60</td>
</tr>
<tr>
<td>(d) After 4x4 decorrelation</td>
<td>1,424,400</td>
<td>12.94</td>
</tr>
<tr>
<td>(e) With 1 bit error frame</td>
<td>3,728,400</td>
<td>4.94</td>
</tr>
<tr>
<td>(f) With 2 bit error frame</td>
<td>6,032,400</td>
<td>3.06</td>
</tr>
</tbody>
</table>

It can be easily seen from the experimental results from this chapter that the localized decorrelation technique can be used very efficiently in compression video information. The technique is low in complexity and computational load and can be easily incorporated into existing popular video codecs. The algorithm can be fine-tuned to achieve high performance in real time for very low bit rate operation.
Figure 4.9: (a) Original Flower Garden image sequence frame number 30

Figure 4.9: (b) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation only
Figure 4.9: (c) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation and $16 \times 16$ single level block localized decorrelation

Figure 4.9: (d) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation and $4 \times 4$ single level block localized decorrelation
Figure 4.9: (e) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD)

Figure 4.9: (f) Flower Garden sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD)
Figure 4.9: (g) PSNR curves for 30 frame Flower Garden test sequence
Figure 4.10: (a) Original Susie image sequence frame number 30

Figure 4.10: (b) Susie sequence frame number 30 reconstructed with motion estimation and compensation only
Figure 4.10: (c) Susie sequence frame number 30 reconstructed with motion estimation and compensation and 16x16 single level block localized decorrelation

Figure 4.10: (d) Susie sequence frame number 30 reconstructed with motion estimation and compensation and 4x4 single level block localized decorrelation
Figure 4.10: (e) Susie sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD)

Figure 4.10: (f) Susie sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD)
Figure 4.10: (g) PSNR curves for 30 frame Susie test sequence
Figure 4.11: (a) *Original* Football image sequence frame number 30

Figure 4.11: (b) *Football sequence frame number 30 reconstructed with motion estimation and compensation only*
Figure 4.11: (c) Football sequence frame number 30 reconstructed with motion estimation and compensation and 16x16 single level block localized decorrelation

Figure 4.11: (d) Football sequence frame number 30 reconstructed with motion estimation and compensation and 4x4 single level block localized decorrelation
Figure 4.11: (e) Football sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 1 bit quantization of error frame (DFD)

Figure 4.11: (f) Football sequence frame number 30 reconstructed with motion estimation and compensation, 4x4 single level block localized decorrelation and 2 bit quantization of error frame (DFD)
Figure 4.11: (g) PSNR curves for 30 frame Football test sequence
Chapter 5

Software Implementation

5.1 Implementation Issues and Considerations

A logarithmic search algorithm was used in the experimentation for motion compensation due to time limitations. As noted earlier, the full search technique takes an average of 6 minutes to perform motion estimation and compensation between two adjacent frames.

In the decorrelation process, as mentioned earlier, an infinitesimal value of the order of $10^{-16}$ was added to the denominator in the decorrelation equation to prevent division by zero.

The prediction residual error was quantized using a Lloyd-Max quantizer designed on the assumption that the error frames have a Laplacian probability density function (pdf).

The reconstructed frames were stored as GIF images and a GIF animation program was used to create the AVI movie files.
5.2 Matlab

All simulations to support the theory were performed using MatLab (Matrix Laboratory). Matlab is a matrix data type based scientific computing environment. The algorithms developed in this thesis uses the matrix based computing philosophy to the fullest. Since images can be treated as matrices, analysis and development with Matlab is much easier than using a programming language such as C. Matlab has built-in operators to manipulate matrices that makes it easy to use and enables rapid prototyping.

One of the major disadvantages of Matlab is its speed. Since programs are interpreted, it is much slower than native code. Matlab provides mechanisms to incorporate native code by means of dynamic loading of modules. But this process is platform dependent and is a tedious process.

Matlab also provides a wide range of functions for image and signal processing. The Image Processing Toolbox supports many different image formats and image display functions. The command prompt based interactive environment enables the user to experiment with different functions, both built in and user defined, and study the results immediately.

Some of the Matlab code used in the simulations is given in Appendix F.
Chapter 6

Conclusions and Future Research

As illustrated with the above experiments, the localized temporal decorrelation scheme can be used as an effective intermediate stage in current block based video compression algorithms. The algorithm can also be extended to other video coding systems such as model based or region methods where small hidden surfaces can be compensated with very few coefficients.

The one disadvantage of the method is that the effectiveness of the algorithm depends, like all other video codecs, on the video content and also the block size used for decorrelation. Smaller the block size better the decorrelation but more number of coefficients have to be transmitted. Hence this is a good intermediate stage but cannot replace the DCT based inter and intra block stages completely.

This research and report has concentrated on the effectiveness of the localized decorrelation system with respect to temporal motion compensation. Experimental results have been used to highlight the performance improvements obtained with this system. Future research must be concentrated on implementation of the algorithm on a real time basis. The concept can also be extended to include wavelet based energy localization mechanisms and three-dimensional extensions of this algorithm (space and time localization).
Appendix A

A Review of Digital Video Compression

A.1 Digital Video Representation

Video is a three dimensional signal. It is a function of space \((x, y)\) and time, \(t\). The two spatial variables correspond to the still images displayed over time in a video sequence hence video is also referred to as a time varying image sequence or simply an image sequence (figure A.1).

Each image in the sequence is known as a \textit{frame} and the number of frames displayed per second is called the \textit{frame rate} of the image sequence. Different video display standards use different frame rates. The \textit{National Televisions Standards Committee} (NTSC) has adopted a frame rate of 30 \textit{frames per second} (fps) for home television broadcast and display systems. Cinema theaters display movies at 24 fps. [Tek], [Cla], and [Bha] discuss various video sampling, scanning, and display formats.

Most video compression systems take advantage of the fact that there is little motion between adjacent frames in an image sequence and that there is a high degree of correlation between adjacent pixels in each of the frames. Redundancy and irrelevancy reduction techniques are used to decorrelate these in order to compress image sequences.
A.2 Redundancies and Irrelevancies

Most video compression systems utilize both redundant and irrelevant information to achieve compression. Redundant (extra) information are those that can be predicted using prior or statistical knowledge of the signal. For example, pixels in an image can be predicted by using the statistics of neighboring pixel values.

Video signals also contain the information that is beyond the human perceptual threshold. Such information is regarded as irrelevant (unwanted) information and need not be considered for compression. Very high frequency spatial signals in an image are considered irrelevant since the human eye would not be able to distinguish them.

Compression systems that use only the redundant information are called lossless compressors since the information can be reconstructed exactly without error. But in the case of video compression, since the end user is the Human Visual System (HVS), irrelevant information can also be utilized to achieve high compression ratios. Systems that use irrelevant information are called lossy compressors since from a statistical perspective some of the signal data has been discarded; but they are considered perceptually lossless since the eye cannot detect the loss in information.
A.3 Video Compression Systems

Video compression systems, including the Motion Pictures Experts Group (MPEG) and the International Telecommunications Union (ITU) standards, use a closed loop algorithm (figure A.2) to compress image sequences. The closed loop structure, which is similar to the Differential Pulse Code Modulation (DPCM) technique used in signal and image compression, is employed to prevent the accumulation of error from frame to frame.

As illustrated in figure A.2, most video compression systems compress the image sequence in two steps. First temporal decorrelation techniques are used to remove redundancies along the time axis. This can also be considered as a temporal prediction operation. The prediction error frame, also referred to as the Displace Frame Difference (DFD), is then decorrelated along the spatial coordinates.

Since the spatial decorrelator operates on the temporal prediction error, very good temporal decorrelation is necessary for efficient video compression. This reiterates the earlier mentioned statement that temporal decorrelation is the most critical component in any image sequence compression system. Currently, the most popular method of temporal decorrelation is motion estimation and compensation techniques. Motion estimation methods attempt to describe the motion of an object or region from one frame to another. The MPEG and ITU standards employ motion estimation and compensation techniques to decorrelate the frames in time. [Bha], [Cla], [Duf], [Rao],
and [Tek] provide detailed discussions on motion estimation and compensation algorithms.

As shown in figure A.2, spatial decorrelation methods are used on the temporal prediction error image to further remove redundant and irrelevant information. Spatial decorrelators take advantage of the fact that most of the pixels in an image are statistically correlated. This is the principle used in still image compression.

The Discrete Cosine Transform (DCT) introduced in [Ahm] is currently the most popular method for spatial decorrelation. The Joint Photographic Experts Group (JPEG) algorithm used for still image compression uses a two dimensional version of the DCT (2D-DCT). The algorithm dissects the image into 8-by-8 blocks. Each block is subjected to a two dimensional Discrete Cosine Transform (DCT). There are four different types of DCT [Rao]. Of these DCT-II is the most widely used transform. The one dimensional N point transform of the DCT-II is given below in equation A.1. For a two dimensional signal, like an image, the 1-D transform is first applied to the rows and then the columns of the image or visa versa.

\[
X_{C^2}(m) = C(m) \sum_{n=0}^{N-1} x(n) \cos \left( \frac{m(2n+1)}{2N} \right), m = 0, 1, \ldots, N - 1
\]  

(A.1)

The DCT converts the image blocks from the spatial domain to the frequency domain. A suitable quantizer is applied to the transformed blocks. This quantization process is the only lossy component in the compression algorithm. The quantized DCT coefficients are entropy coded for storage or transmission.

Since the JPEG compression algorithm is lossy, it generates a smaller image file when compared with a GIF image file. But the smaller file size comes at a price. JPEG compression systems rely on the availability of irrelevant information in the image. i.e. information that cannot be perceived by the Human Visual System (HVS). Figure A.3 shows a decompressed GIF image of lena (a standard test image). As one can see there are no visual artifacts. This is because, as mentioned earlier, GIF is a lossless compression system. On the other hand one can clearly see the blocking artifacts generated in the decompressed JPEG image (figure A.4).
Figure A.3: GIF compressed lena image  file size = 72498 bytes

Figure A.4: JPEG compressed (75% quality) image  file size = 11482 bytes
[Bha], [Cla], [Tek] contain detailed discussions on image compression techniques and standards. [Mur] contains a detailed description of all popular image compression formats and is an excellent reference for all possible standard computer image coding algorithms. The book also comes with pre-written software algorithms.

The DCT algorithm used in the JPEG standard is also used in video coding. MPEG and the ITU image sequence coding standards use the 2D-DCT for spatial decorrelation of the DFD frame. The image is broken down into blocks of size 8-by-8 pixels and the 2D-DCT is applied to each of the blocks. The 2D-DCT converts the block pixel values to a two-dimensional frequency representation. The transformed coefficients are appropriately quantized (Appendix C) and entropy coded for transmission and/or storage.

Due to the block-wise break down of the image, DCT based image and video compression systems suffer from high distortion effects at high compression ratios (low bit rates). Also the DCT transformation is computationally intensive. [Rao] is an excellent reference on the subject of DCT and DCT based compression systems and coding techniques. Lapped transforms have been proposed as an alternative to DCT in [Mal] but have not gained widespread use. Lapped transforms can be used in an overlapped manner to achieve improved quality images at equal compression ratios.

In recent years, subband image coding techniques have become widely popular. In this technique, filters are used to decompose an image into different resolution bands (Appendix E). Each band is quantized and appropriately coded.

Although not adopted by any standards body, subband coding systems offer many advantages over DCT based techniques. Subband coding is computationally faster and less complicated and since the decomposition is performed over the entire image, no blocking artifacts are introduced into the decoded image. A review of subband based image coding along with some references is presented in appendix E.
A.4 Current Standards in Digital Video Compression

There are currently many different video compression algorithms in use. Each one developed for specific applications by different industrial, academic and scientific organizations. For example, Motion JPEG treats each frame in the image sequence as an individual image and compresses them using the standard JPEG compression algorithm. This technique is mostly used for studio editing.

The H.261 standard developed by the ITU standards body primarily for the purposes of transmitting video over low bit rate channels such as the Integrated Services Digital Network (ISDN). The standard was developed to transmit pictures at rates ranging from 64 kbps to 1920 kbps. The ITU has now proposed the H.263 standard for video transmission at rate below 64 kbps. The H.263, which will replace the H.261 standard in some of the application areas, was primarily designed for video-telephony.

MPEG has also developed many different standards for video compression for various applications. MPEG I was developed to support data transfer rates of up to 1.5 Mbps. This format is ideal for storing multimedia information on CD-ROMs which have similar data transfer rates. MPEG I does not support interlaced video. Interlaced video must be converted to non-interlaced format before being coded.

MPEG 2 was defined to support data rates of up to 5 Mbps for NTSC and PAL quality or 10 Mbps studio quality video. It can also support data rates of 80 Mbps to 100 Mbps for HDTV and DVD applications. MPEG 2 is backward compatible with MPEG I. Flexibility of input format, random access capability, and bit stream scalability among many others are also accommodated in the MPEG 2 definition.

MPEG 4, like H.263, targets very low bit rate applications. The standard, to be introduced towards the turn of this century, is to incorporates many second generation video coding techniques that are catered for specific types of image sequences such as head and shoulder video. Second generation coding techniques are content dependent since they take advantage of various features present in the image sequence. [Aiz] has implemented a technique based on image modelling. This technique is useful in
transmitting human interation scenes. The algorithm models the scene and updates a similar model at the decoder to represent motion. Morphological filtering ([Ser]) techniques have also been used for achieving reasonable low bit rate performance.

Fractal based video compression has also been proposed as an alternative to block based DCT technique to achieve very high compression [Bar, Fis, Pen]. Fractal based techniques attempt to describe a region in an image with a transformed reference region in the same or another image. Fractals work well on natural images and fractal based systems are highly asymmetric. i.e. the encoding process consumes much more time than the decoding process. All the above techniques are classified as second generation techniques and are primarily used for low bit rate applications [Ebr, deF].

Detailed information concerning the ITU and MPEG standards are given in [Cha], [Strl], [Bha], [Rao], and [Tek]. The current status of the MPEG standards and the information on the working groups and proposals can also be obtained through the Internet at http://www.mpeg.org.

The computer industry also uses many different formats for multimedia applications. The Quicktime format from Apple Computers Inc., the AVI format from Microsoft and Indeo from Intel are some of the popular ones. The MPEG format is also used to store multi-media information on CD-ROMs and exchange video files over the Internet.

Figure A.5 illustrates a typical motion compensated DCT based video compression system as used by the ITU standards and also by MPEG 1 and 2.

As pointed out earlier, the temporal decorrelation block is the most critical component in any image sequence compression system. Optimizing the temporal decorrelator performance and coupling it with a good spatial decorrelator would improve the overall video compression system efficiency, both from a computational and a performance perspective.
Figure A.5: Simplified block diagram of a motion compensated closed loop DCT based video coder
Appendix B

Motion Estimation

B.1 Temporal Redundancy Reduction

Adjacent video sequence frames are highly correlated in the temporal axis. Many of the objects and regions displayed in one frame also appear in subsequent frames. Motion estimation and compensation techniques take advantage of this correlatedness for compression. Video motion and motion estimation principles and techniques are discussed in this appendix.

B.2 Motion in Video

There are many different types of motion associated with image sequences. These can be categorized into object based motion and camera motion. Translatory movement of various objects in a video constitute to object motion. Pan and zoom related motion are due to camera movement. Some of these motion types are illustrated in the figure B.1.
Figure B.1: Simple motion illustrations

Figure B.2: Translatory motion vector illustration

B.3 Motion Estimation Techniques

The principle behind motion estimation and compensation is quite simple. The objective is to find a motion vector that best describes the movement of an object or region from one frame to another. As an example consider the two frames in figure B.2.

Figure B.2 illustrates a simple translatory motion of the 'L' shaped region in time. The shaded region has move from coordinates \((x, y)\) in frame \(k\) to \((a, b)\) in frame \(k+1\). Therefore the motion vector, \(V\), describing the motion of the region is given by \(V = (a - x, b - y)\). Hence to one needs to move the shaded region by the vector \(V\) to reconstruct frame \(k + 1\). Therefore only the \(k^{th}\) frame and the motion vector, \(V\), has to be transmitted or stored.
Motion estimation techniques can be classified into four main groups [Duf].

1. Gradient techniques,
2. Pel-recursive techniques,
3. Block matching techniques, and
4. Frequency domain techniques.

Of the above listed, only pel-recursive and block matching techniques were specifically developed for coding purposes. Gradient techniques were developed primarily for image sequence motion analysis applications. Frequency domain techniques are not widely used in the field of video coding. Block matching techniques, based on the minimization of a disparity criteria, has become the most popular method for motion estimation and compensation. Many of the current video coding standards have embraced the block based method.

B.4 Block Based Motion Estimation

Block based motion estimations techniques are by far the most popular method for motion estimation and compensation. Block based methods are low in complexity and can be very easily implemented on hardware. All the current existing standards use block based methods to perform temporal redundancy and irrelevancy reduction.

In block matching techniques, the objective is to find the best matching block from one frame to another. Each frame is dissected into blocks and the block are used to construct a future frame. Block matching is performed in the spatial domain. Different types of block matching algorithms exist and they differ in:

1. the matching criteria,
2. search strategy, and
3. the block size.

Each of these criterion are discussed in detail below.

B.4.1 Block Matching Criteria

When searching for a matching block, some disparity criteria is minimized or maximized to obtain the best matching block. Some of the common criteria used in current coders are the Mean Squared Error (MSE), Mean Absolute Difference (MAD), and the Maximum Pel Count (MPC).

In MSE, the squared average of the error between the predicted and the original block is minimized. The MSE is given by

\[ MSE(d_1, d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1,n_2) \in B} \left[ B_k(n_1,n_2) - B_{k-1}(n_1+d_1,n_2+d_2) \right]^2 \quad (B.1) \]

\( B_k \) is the original block in the current \( (k^{th}) \) frame and \( B_{k-1} \) is the predicted block in the \( (k - 1)^{th} \) frame. The motion vectors are given by \( d_1 \) and \( d_2 \). The row and column search parameters is given by \( n_1 \) and \( n_2 \) and the block size is \( N_1 \)-by-\( N_2 \).

In the MAD criteria,

\[ MSE(d_1, d_2) = \frac{1}{N_1 \times N_2} \sum_{(n_1,n_2) \in B} | B_k(n_1,n_2) - B_{k-1}(n_1+d_1,n_2+d_2) | \quad (B.2) \]

is minimized. The MAD value is preferred over the MSE criteria due to the complexity involved in VLSI implementation of the square operator.

For the maximum pel count criteria, each pixel in a predicted block is classified as either a matching or non-matching pixel. A threshold value is used to classify the
pixels in this manner. If a pixel is within the threshold value of the original pixel, then it is classified as matching, else is not. The block with the most number of matching pixels is chosen as the best matching block.

Mean squared error method provides the best error minimization but the MAD and MPC criteria are easier to implement in hardware and hence faster. The MSE is mostly used in systems that do not require real time encoding and therefore are mostly used in storage and retrieval systems.

### B.4.2 Search Strategy

There are many search strategies in use currently. Three of the most widely used ones are the full or the brute force search method, the three step method and the cross search method. These are discussed in detail below.

#### Full Search Technique

In full block search method, all points in a selected region are searched for a given block such that either the MSE or the MAD criterion is met. The search limits vary
for different applications and the required accuracy. In figure B.3, the best match for
the dotted block would be obtained in the shaded region (search area).

Since the full search method has to search all coordinates of a region, it is mostly
used in off-line compression systems which do not require real time motion estimation.
Storage and retrieval systems are an example of such a case.

**Three - Step Search**

This method is a fast search technique. It is widely popular for hardware implementa-
tions which are used in real time applications. This method uses the gradient descent
approach to satisfy the disparity criteria.

In figure B.4, the '0' marks the location corresponding to the reference coordinates
of the block in the \( k^{th} \) frame. The circled number indicates the minimum error point
for a given step. The algorithm is described below [Tek][Bha].
**Cross Search Algorithm**

The cross search method (figure B.5) is similar to the 3 step search, except the search is done in a 'X' or a '+' search pattern. The search is stopped if the best search is at the center or at the boundary of the search window. The algorithm is implemented as follows.

---

**Figure B.5: Cross search algorithm**

<table>
<thead>
<tr>
<th>Three Step Search Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> Search the nine points ('0' and the eight '1's) and evaluate either the MSE or the MAD value. If minimum is at '0' then there is no motion.</td>
</tr>
<tr>
<td><strong>Step 2</strong> Now search the eight points around the circled '1' (the eight '2's) to find the minimum point. If '1' is the minimum, then stop.</td>
</tr>
<tr>
<td><strong>Step 3</strong> Search the eight '3's around the circled '2' and take minimum point. This gives the position of the best matching block.</td>
</tr>
</tbody>
</table>
Figure B.6: Square search algorithm

<table>
<thead>
<tr>
<th>Cross Search Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
</tr>
</tbody>
</table>

**Square Search Algorithm**

The square search algorithm is identical to the cross search algorithm except that all eight points around the central coordinate (figure B.6) are searched for the best match - compared to 4 points in the cross search method.

In all the above search methods, threshold values are incorporated into the algorithm so as to compensate for poor minimization. For example if the error between the predicted block and the original block is above a threshold, then the original block must be transmitted as it is. This is known as intra coding. Increased search area results in better motion estimation but proportionally increases the computational load. The full search method generates the best estimate but is also the most compu-
tationally intensive algorithm. The techniques discussed above can also be used as the basis for other algorithms such as the overlapped block motion estimation algorithm and the adaptive block size method.

B.4.3 Block Size

The block size used depends on the application, required speed and resolution, and accuracy. Smaller the block size more accurate the temporal prediction but greater the computational load and bandwidth requirement. Most major codecs use either an 8-by-8 or a 16-by-16 block size. Algorithms with adaptive block sizes have been proposed and implemented but have not been embraced by standards and application for various implementation problems.

B.5 Example of Block Based Motion Estimation and Compensation

This section illustrates two of the block matching algorithms, namely the full search method and the cross search method, used in motion estimation and compensation. Figure B.7 and B.8 show two adjacent frames from the flower garden sequence.

The motion between frame 1 and frame 2 were estimated and compensated using the full search and the cross search techniques discussed above. A 16-by-16 block size was used with the threshold set to zero. A search area of 31 pixels was utilized. Motion estimation and compensation was done at pel accuracy level.

Figures B.9 and B.10 show the results from the full search method. Figure B.9 illustrates the resulting motion vector and the prediction error image is given in figure B.10. The average mean squared prediction error is 253.1311 compared to 1741 for the simple difference image (frame2 frame1). The algorithm took approximately 150 seconds to complete.

Figure B.11 shows the motion vectors obtained using the cross search method.
Figure B.7: Frame 1 of flower garden image sequence

Figure B.8: Frame 2 of flower garden image sequence
Figure B.9: Motion vectors generated by blocked based full search method
Figure B.10: Prediction error image obtained using full search method
The parameters for the algorithm were the same used for the full search method. The motion estimation was completed within 3.5 seconds. The average mean squared error was 294.55.

As can be seen from the above experimental results, although full search algorithm was almost 50 times slower than the cross search algorithm, it generated a better prediction image and thus would require fewer bits to encode the prediction error image. The above techniques can be combined with overlapped block motion estimation and hierarchical block motion estimation methods to generate fast accurate prediction images.
Figure B.12: *Prediction error image obtained using cross search motion estimation*
B.6 Problems with Block Based Motion Estimation

Block matching algorithms suffer from many drawbacks. Among the major ones are blocking artifacts, poor motion compensated prediction along edges, and unreliable motion fields. Several techniques have been proposed to overcome these problems. [Con] and [Ill] propose multi-resolution based motion estimation. Adaptive block size based motion estimation is proposed in [Acc] and overlapped block motion estimation is used in [Auy] and [Oht]. As with all motion estimation and compensation techniques, the above mentioned alternatives are computation intensive and increase the coding delay.

Despite the above mentioned problems, block search methods are the most popular motion estimation techniques in use today. This is mainly due to ease of hardware and software implementations, comparatively robust performance with all types of images (synthetic and natural) and very low computational load compared to object based segmented motion estimation or the likes. All current standards employ block based motion estimation and compensation.
Appendix C

Quantization and Human Visual System Considerations

Quantization is the process by which a signal is confined to a predefined set of output values. A good example of a quantizer is an analog to digital converter (ADC). Analog signals have an infinite set of output values, i.e. the value can have an infinite number of resolutions. When this signal is sampled and converted to the discrete time domain using an 8 bit ADC, the output signal is confined to \(2^8 = 256\) distinct discrete values.

The principle behind quantization is simple. Any signal value which lies in between a given range is assigned a specific value. Therefore if an analog signal is quantified to 16 discrete levels, then one needs to transmit only a 4 bit code to indicate which of the 16 levels a given sample belong to. Thereby the signal is compressed.

Quantization is a lossy process, i.e. it is a non-reversible function. Once a signal has been quantized, the original signal cannot be recovered. When a digital image, which is already a quantized analog image, is quantized, it results in further loss of information and resolution. The level of quantization determines the quality of the output.

Uniform and pdf optimized quantizers are two of the most popular quantization methods employed in digital image compression. Uniform quantizers, as the name
implies, quantizes the input signal in a uniform (linear) manner. The operation of the uniform quantizer is illustrated in a stepwise manner.

1. If a signal is to be uniformly quantized to \(N\) bits, then the signal input range is first divided into \(2^N\) distinct decision boundaries, say \(d_1, d_2,\) and so on.

2. If an input value falls in between \(d_i\) and \(d_{i+1}\), then the value is quantized to the mean value of the range. i.e. quantized value equals \((d_i + d_{i+1})/2\).

The input-output relationship for a uniform quantizer is given below in figure C.1.

Probability density function (PDF) optimized quantizers rely on the statistics of the signal source to quantize the signal. The quantizer, also known as the Loyd-Max quantizer, allocates the decision boundaries based on the probability density distribution of the signal. The determination of the values of the decision boundaries (values of \(d_i\)) is a recursive process. [Max] and [Woo] contain detailed discussions on the development of the pdf optimized quantizers.

The uniform quantizer is the most popular one due to the lower computational complexity and ease of hardware implementation. This quantizer is suitable if the input signal has a uniform pdf. In many instances, the input signal has a Laplacian
like pdf. This is common in prediction error type of signals and also in high frequency subbands. The pdf optimized quantizer generates much superior results with non-uniformly distributed pdfs. The drawback of the pdf optimized quantizer is that since it is statistically dependent, the mean and the standard deviation of the input signal must be computed and transmitted as overhead information. Without this information reconstruction of the signal would be erroneous.

In vector quantization, a collection of symbols is treated as a single source symbol. For example, in an image two pixels are grouped together to form a new symbol. The two pixels can be used to form a vector. Now instead of quantizing each pixels separately, one can quantize the vector. Quantizing in multi-dimension, provides a means to achieve better quality and high compression ratios [Xue]. The problem with vector quantization methods is that it requires the training of a code book for quantization and the transmittal of that code book for decoding. [Gre] is an excellent reference on the subject of quantization and contains detailed discussions on scalar and vector quantization techniques, algorithms and applications.

The end user of the output of an image processing system is the human eye. Hence the characteristics of the human eye and the human perceptual system play an integral part in any image or video compression system. Studies have shown that the eye's frequency response is poor at high frequencies [Gle]. This result has been incorporated into compression system so as to maximize the visual quality at a given compression ratio. It is the HVS characteristics that permit the usage of irrelevant information in compression.

The HVS model has been incorporated into many quantization and image compression schemes. [Gra] has incorporated the HVS model into vector quantization schemes for image compression.
Appendix D

Subband Image Coding - A Brief Outline

In subband coding techniques, a signal is decomposed into a multitude of resolutions and then each resolution is encoded appropriately. A signal can be decomposed into its multi-resolution components by first filtering and down sampling (decimation). The filter is used to prevent anti-aliasing effects. Because of the decimation at the output of each filter, the total number of samples is preserved. The decomposed signal can be reconstructed by up-sampling and interpolating. Figure D.1 illustrates a two band analysis and synthesis filter bank system.

In the system illustrated above, the signal is passed through a low pass filter \(h_L(n)\) and a high pass filter \(h_H(n)\), each with a cutoff frequency at \(\pi/2\). The interpolation and decimation is done by a factor of two This method is referred to as a Quadrature Mirror Filter (QMF) [Joh]. For a given low pass filter with an impulse response \(h_L(n)\) and a transfer function of \(H_L(z)\), for perfect reconstruction, the other three filters are related by the following set of equations.

\[
\begin{align*}
H_H(z) &= H_L(-z) \\
G_L(z) &= H_L(z) \\
G_H(z) &= -H_H(z) = -H_L(-z)
\end{align*}
\]
[Vai] is an excellent reference on multi-rate filter bank theory and its applications.

In many applications, including signal coding, only the low frequency band output is further decomposed into its subbands. To accomplish this, the low frequency component is fed back into the analysis filter banks and the process is repeated until the required number of resolutions is obtained (figure D.2). At each stage the output of the filter banks is decimated by a factor of two.

The subband decomposition process illustrated in figures D.1 and D.2 can be easily extended to 2-D signals like images. For example each row of an image can be considered as a 1-D signal and passed through the filter banks to generate a low frequency and a high frequency image. The decimation is done column wise. Now both the low and the high frequency images are passed through the filter bank
structure once more but this time the filtering is done column wise and the decimation row wise. The filter bank structure illustrated in figure D.3 can be used to decompose an image into four equal sized sub-bands.

As discussed earlier, for multiple resolutions, the low frequency image can be fed back into the QMF banks for further decomposition. The reconstruction is carried out in a similar but reverse process. Figure D.4 shows an image after undergoing a three level subband decomposition. It is evident that most of the signal energy is contained in the low frequency resolution of the image. The high frequency bands, also sometimes referred to as detail bands, contain information regarding image edge details and isolated artifacts.

Different types of coding schemes have been proposed to encode the subbands. [Gon] uses a uniform quantizer with variable zero bin width to quantize the subbands and then encode each subband using Arithmetic coding. [Sha] proposes an embedded algorithm to compress the image. This algorithm relates the high frequency bands to the low frequency bands thus creating a tree like structure. Therefore only the significant coefficients of the tree are encoded. By using the [Sha] coder one can obtain an
Figure D.4: Three level subband decomposition of Claire image
embedded bit stream for the image. [Sai] and [Mar] have propose conceptually similar algorithms to implement the coder. These adhere to the [Sha] coder in principle but implement the algorithm differently. A similar algorithm was used to code the images during simulation testing of the proposed technique.

Since, in subband coding, the entire image is decomposed, this does not introduce any blocking artifacts such as those found in DCT based methods. This is a major advantage of the subband coding system. As with any method this too has disadvantages. Circular convolution filtering introduces 'ringing' effects around image edges. This can be overcome using circular symmetric extensions.

Good introductions to the subject of wavelet and subband theory is contained in [Mul, Rag, Rio, and Str]. [Vai] is an excellent reference on the subject of subband filter theory and design. [Vet] is a good reference on the subject of subband filtering applications in signal processing. Subband based image and video coding is also discussed in this text. Fast algorithmic implementations of wavelets and subband decomposition is discussed in [Rio1]. Human visual system based wavelet/subband coding and quantization is discussed in [Kim, Rou]
Appendix E

Zero Tree Coding Algorithm

Subband based embedded tree coding algorithms have been used successfully to compress digital images and video. In this section we present a Zero Tree Coding (similar to the ones discussed in [Sha], [Sai] and [Mar]) algorithm. The algorithm is implemented on an image that has undergone subband decomposition. The key to the success of the algorithm is the assumed parent-child relationship among the coefficients. The parent-child relationship used in the algorithm is similar to the one used in [Sha]. This relationship is illustrated in figure E.1. The figure illustrates an 8-by-8 image decomposed into subbands.

The bold arrows from the lowest subband show the parent child relationship between the lowest band and the neighboring three subbands. For all the other coefficients there is a one to four relationship. i.e. each coefficient in a subband has four children. All subband coefficients except the ones in the highest frequency bands have children and the coefficients in the lowest subband have no parent coefficients.

The subbands are scanned in a priority order as shown in figure E.2. The lowest bands have the highest priority and the highest bands have the least priority. The scanning path is given by the arrow traced from band to band.

The first step in a zero tree coding algorithm is to establish a significant coefficients map. i.e. a map of all the significant subband coefficients. To generate the significance map, one has to scan the coefficients in the reverse priority order depicted in figure
E.1. Parent child relationship used in Zero Tree Algorithm

E.2. During the scanning process all the coefficients in all the bands are categorized into one of four groups. They are significant coefficient, isolated zero value, zero tree root or don’t care value. The classification algorithm flow chart is given in figure E.3.

Once the significance map has been generated, the coefficients can be coded for transmission or storage. Principle behind the encoding procedure is very straightforward. Only those coefficients that have a significant or an isolated zero parent are classified and transmitted. Since this classification had been done in the significance mapping stage, all coefficients except those classified as Don’t care are encoded and transmitted. For an isolated zero a '00' bit is transmitted while for a significant coefficient a '1' followed by the coefficient quantized index in transmitted. A zero tree root (ZTC) index is preceded by a '01' binary code. Figure F4 outlines the encoder algorithm flowchart.
Figure E.2: *Subband priority scanning order*

**Figure E.3: Significance mapping algorithm flowchart**
Figure E.4: Encoding algorithm flow chart
Appendix F

Matlab Code

function [X]=pgmread(filename)
%PGMREAD Read a PGM (Portable Gray Map) file from disk. Only binary
% encoded PGM images ((P5)are supported.
% [X]=PGMREAD('filename') reads the file 'filename' and returns
% the indexed image X. If no extension is given for the filename,
% the extension '.pgm' is assumed.
% See also: PGMWRITE, BPMREAD, GIFREAD, HDFREAD, PCXREAD,
% TIFFREAD, XWDREAD.
% Marcelo Neira Eid 12/13/96
% mne@puc.cl
% Last revision: Mon Dec 9 15:24:45 PST 1996

if (nargin~=1)
    error('Requires a filename as an argument.');
end;
if (isstr(filename)~=1)
    error('Requires a string filename as an argument.');
end;
if (isempty(findstr(filename,'')))==1)
    filename=[filename,'.pgm'];
end;

fid=fopen(filename,'rb');
if (fid==-1)
    error(['Error opening ','filename,' for input.']);
end;
aux=fgetl(fid);
if (strcmp(aux,'P5')==0)
fclose(fid)
error(['/filename,' 'is not a valid PGM binary encoded image']);
end;

% Below the comments are stripped
comments=1;
while(comments)
aux=fgets(fid);
if (aux(1)~='#')
comments=0;
end;
end;

% Get the dimensions
[width height]=strtok(aux);
width=str2num(width);
height=str2num(height);

% This strip the number of grays information. Since we know they are 255
% a priori, there isn't need to capture this information
aux=fgets(fid);
X=fread(fid);
fclose(fid);
X=reshape(X,width,height);

% The image is transposed after the read. Also Matlab pixels
% start from 1 so we transpose the image and add 1 to it.
X=X';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [state]=pgmwrite(X,filename)
% PGMWRITE Write a PGM (Portable Gray Map) file from disk. Only binary
% encoded PGM images ((P5) are supported.
% PGMWRITE(X,'filename') writes a PGM file containing the
% indexed image X to a disk file called 'filename'. If no file
% extension is given with the filename, the extension '.pgm'
% is assumed.
% See also: PGMREAD, BMPWRITE, GIFWRITE, HDFWRITE, PCXWRITE, TIFFWRITE,
% XWDWRITE.
% Marcelo Neira Eid 12/13/96
% mne@puc.cl
% Last revision: Mon Dec 9 15:32:46 PST 1996
if (nargin~=2)
    error('Requires two arguments.');
end;

if (isstr(filename)~=1)
    error('Requires a string filename as the second argument.');
end;

if (isempty(findstr(filename,'.'))===1)
    filename=[filename,'.pgm'];
end;

fid=fopen(filename,'wb');
if (fid==-1)
    error(['Error opening ',filename,' for output.']);
end;

[width height]=size(X');
fprintf(fid,'P5
');
fprintf(fid,'# CREATOR: pgmwrite.m Version 1.0\n');
v=version;
fprintf(fid,'# m-file for Matlab %s\n',v);
fprintf(fid,'# By Marcelo Neira Eid <mne@puc.cl>\n');
fprintf(fid,'# %d \%d \%d \%d
',width,height,255);
AUX=X';
AUX=(AUX-min(min(AUX)))/(max(max(AUX))-min(min(AUX)))*255;
fwrite(fid,AUX);
fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FULLME Full search method based motion estimation
% MV = FULLME(Frame1,Frame2,Threshold,Blk,P)
% Performs motion estimation between Frame1 and Frame2 using a block based method. The block size is determined by Blk. Threshold determines a threshold value for error tolerance. P determines the block search area for each block. The function returns a N by 4 matrix containing the coordinates of matching blocks. The first column of MV contains the row coordinate of the first block in Frame1 and the second column the column coordinate. Columns 3 and 4 are for the corresponding matching blocks in Frame2.

% Updated : May 4, 1997

function MV = fullme(Frame1,Frame2,Threshold,Blk,P)
[noRow,noCol] = size(Frame1);

if (rem(noRow,Blk) ~= 0 | rem(noCol,Blk) ~= 0)
    error('Block size is not multiple of Frame size');
else
    MV = [];

    for r = 1:Blk:noRow
        for c = 1:Blk:noCol

            %## extract blocks from frame 1 and frame 2
            B2 = Frame2((r:r+Blk-1),(c:c+Blk-1));
            B1 = Frame1((r:r+Blk-1),(c:c+Blk-1));

            %## calculate error
            error = sum(sum((B2-B1).^2)) / (Blk * Blk);

            if (error >= Threshold)
                [R1,C1] = fullmv(Frame1,B2,r,c,P);
                MV = [MV;R1;C1;r;c];
            else
                MV = [MV;r;c;r;c];
            end
        end
    end

return;

########################################################################

% FULLMV Motion estimation using full search method
% [R1,C1] = FULLMV(Frame1,Block2,R2,C2,P)
%
% Function uses the full search method to find the best
% match for the matrix "Block2" in the matrix "Frame1".
% R2 and C2 are the location of Block2 in the parent
% matrix. The parameter "P" determines the search area.
%
% Updated : May 4, 1997

function [R1,C1] = fullmv(Frame1,Block2,R2,C2,P)

[nRowsF1,nColsF1] = size(Frame1);
[Blk,Blk] = size(Block2);
R1 = R2;
C1 = C2;

% calculate initial error
Block1 = Frame1((R1:R1+Blk-1),(C1:C1+Blk-1));
Error = (sum(sum((Block2 - Block1).^2))) / (Blk * Blk);

% initialize minimum error
MinError = Error;

for r = R2 - P:R2 + P
    for c = C2 - P:C2 + P
        if(r > 0 & c > 0 & r <= nRowsFl - Blk + 1 & c <= nColsFl - Blk + 1)
            Block1 = Frame1((r:r+Blk-1),(c:c+Blk-1));
            Error = (sum(sum((Block2 - Block1).^2))) / (Blk * Blk);
            if (Error < MinError)
                R1 = r;
                C1 = c;
                MinError = Error;
            end
        end
    end
end

return;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PSNR Compute peak signal to noise ratio
% 
% P = PSNR(F1,F2)
% 
% computes the PSNR value between F1 and F2
function p = psnr(f1,f2)

% normalize values of F1 and F1 between 0 and 255
f1 = (f1 >= 0).*f1;
f1 = round((f1 > 255).*255 + (f1 <= 255).*f1);

f2 = (f2 >= 0).*f2;
f2 = round((f2 > 255).*255 + (f2 <= 255).*f2);

maxval = 255;
p = 10 * log10(maxval^2 / mse(f1 - f2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LTDF Localized temporal decorrelation function
% 
% NEWF2 = LTDF(F2,NF2,BLK)
% 
% Computes the localized decorrelated matrix NEWF2 with F2 and NF2
% with the block size spacidied by BLK
function newf2 = ltdf(f2,nf2,blk)

[row,col] = size(f2);   % get row and column number
newf2 = zeros(row, col);  % initialize output matrix

for r = 1:blk:row
    for c = 1:blk:col
        b2 = f2(r:r+blk-1,c:c+blk-1);  % get sub-block from f2
        nb2 = nf2(r:r+blk-1,c:c+blk-1);  % get sub-block from nf2

        den = sum(sum(nb2 .* nb2));  % compute denominator
        if (den == 0)
            den = eps;
        end

        coeff = (sum(sum(nb2 .* b2))) / den;  % compute coefficient

        newf2(r:r+blk-1,c:c+blk-1) = nb2 * coeff;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PHASE5 Main program used to analyze localized decorrelation
% technique with block size of 4-by-4. Motion estimation
% was done with 16-by-16 blocks and the residual error is
% quantized to 1 bit and added to the prediction frame. The
% output frames are stored as .GIF files.
pause off
%==================================================================
% definition of all program parameters
%==================================================================

% definition of image file path
pth = 'd:\work\images\football\fb';
outpth = 'd:\phase5\football\fb';

% number of rows in image
imgrow = 240;

% number of columns in image
imgcol = 352;

% number of pixels for extension of image
imgext = 16;

% number of images
noImages = 30;

% decorrelation block size
decblk = 4;
% quantization bits
Qbit = 1;

% motion estimation parameters
%----------------------------------------
blk = 16;       % block size
P = 15;         % block search area

% save coefficients
PSNRVal=[];
map = gray(256);

%---------------------------------------------------------------

% read first frame and extend image boundaries
%---------------------------------------------------------------
nf2 = pgmread([pth int2str(1) '.pgm']);

disp('Writing frame 1 GIF file');
gifwrite(nf2+1,map,[outpth int2str(1) '.gif']);
f1 = zeros(imgrow + imgext * 2,imgcol + imgext * 2);
imagesc(f1);colormap(gray);pause

% parameter for statistical profile of blocks
blockProfile = [];
errorProfile = [];

% main loop
for n = 2:noImages

  % read in new reference frame
  f1(17:16+imgrow,17:16+imgcol) = nf2;

disp(['Processing image: ',num2str(n),'.pgm']);

  % read the next frame and initialize prediction frame
  f2 = pgmread([pth int2str(n) '.pgm']);
  nf2 = zeros(size(f2));

  % initialize frame by frame block monitor
  frameBlkProf = zeros(1,5);

  % step through the image in a block-wise manner
  for r = 1:blk:imgrow
    for c = 1:blk:imgcol

      % read the blocks from frames 1 & 2
      block1 = f1(r+imgext:r+imgext+blk-1,c+imgext:c+imgext+blk-1);
      block2 = f2(r:r+blk-1,c:c+blk-1);

  end
end
% initialize predicted block 2
nBlk2 = [];

% calculate initial difference energy
err = mse(block2 - block1);

% perform motion estimation
[R1,C1] = squaremv(f1,block2,r+imgext,c+imgext,P);

% get prediction block
nBlk2 = f1(R1+blk-1,C1:blk-1);

% reconstruct frame 2
nf2(r:r+blk-1,c:c+blk-1) = nBlk2;

end
end

% localized temporal decorrelation
disp('Localized Temporal Decorrelation');
nf2 = ltdf(f2,nf2,decblk);

% get prediction error
err = f2 - nf2;

% quantize error
disp('Quantizing error');
err = lapquant(err,Qbit);

% add prediction error
nf2 = nf2 + err;

% fix coefficients
nf2 = (nf2 <= 255).*nf2 + (nf2 > 255)*255;
nf2 = (nf2 >= 0).*nf2;

% round off new frame 2
nf2 = round(nf2);

% display nf2
imagesc(nf2);
pause

% calculate PSNR
PSNRVal = [PSNRVal psnr(f2,nf2)];

% write new frame 2 GIF file
gifwrite(nf2+1,map,[outpth int2str(n) '.gif']);

end
Bibliography


100


