CHAPTER 5

INTERFRAME COMPRESSION

5.1 INTRODUCTION

Temporal redundancy is due to the correlation between pixels of temporarily adjacent frames. Lossy video compression techniques such as MPEG depend heavily on effective removal of temporal redundancy to achieve a high compression ratio (Pattanaik et al 2006). Temporal redundancy decreases slowly with time, that is adjacent frames are highly correlated and this correlation reduces as the distance between the frames is increased.

The most popularly used approach to exploit the similarity between the frames is by coding a given frame by first predicting it based on a previously coded frame and then coding the error or the residual in the prediction (Ezhilarasan and Thambidurai 2006). Moreover, the removal of temporal redundancies in a video signal accounts for a significant percentage of the achieved compression. Advanced techniques for coding of the residual signal usually provide little additional compression as compared to the traditional techniques with computational overhead. Rather than improving the residual coding itself, most effective techniques are needed to reduce the residual to be coded (Guy Cote and Lowel Winger 2002).
Normally in a video, consecutive frames typically contain the same imagery, however possibly at different spatial locations because of motion (Borivoje Furht and Oge Marques 2003). Therefore, to improve the predictability, it is important to estimate the motion between the frames and then to form an appropriate prediction that compensates for the motion. The process of estimating the motion between the frames is known as motion estimation and the process of forming a prediction while compensating for the relative motion between the two frames is referred to as motion compensated prediction (HongLiang Li et al 2002). This chapter describes a hybrid block-matching algorithm to estimate the motion vector and also explains how the prediction errors are calculated based on interpolation and extrapolation concepts. The following section explains the motion estimation concept.

5.2 MOTION ESTIMATION

Interframe prediction coding involves removal of temporal redundancy between adjacent video frames. To exploit the temporal redundancy, estimation of motion between the adjacent frames is essential and it contributes significant bit saving by transmitting only the motion vectors. This is the most computationally intensive operation in the coding and transmitting of video signals (Amit and Bawane 2009; Bernardini et al 2008). To estimate the motion vector, there are many approaches like Block-Matching Algorithm (BMA), optical flow based method, and pixel (pel) recursive algorithms. Block based motion estimation is preferred to other algorithms due to a number of reasons. It is relatively straightforward and computationally tractable, it fits well with rectangular video frames and it provides a reasonably effective temporal model for many video sequences (Jinwook Kim and Taegevon Park 2009).
In block matching motion estimation, a single image is subdivided into non-overlapping \( N \times N \) blocks, where \( N \) is usually 16 or 8. Each block in the current frame is compared with blocks of the same size in reference frame in order to find out the best match, which meets an error criteria based on the measurement. The location of the block is defined by co-ordinates \((x,y)\) on the top-left corner of the block. The vector pointing from the current block to the best match block is chosen as Motion Vector (MV). The residual “difference” between current and reference frames is computed by the process of motion compensation, and then coded and transmitted with motion vectors. There are a number of criteria to evaluate the goodness of a match and some of them are as follows (Deepak Turaga and Mohamed Alkanhal 1998)

2. Pixel Difference Classification (PDC).
3. Mean Absolute Difference.
4. Mean Squared Difference.
5. Integral Projection.

Some of these criteria are simple to evaluate, while others are more complicated. Different kinds of algorithms use different criteria for comparison of blocks. One of the first algorithms used for block based motion compensation is the Full Search or the Exhaustive Search (Kilthau et al 2002). In this, each block within a given search window is compared to the current block and the best match is obtained (based on one of the comparison criteria). Although this algorithm is the best one in terms of the quality of the predicted image and simplicity, it is computationally very intensive. With the realization that motion compensation is the most computationally intensive operation in the coding and transmitting of video
streams, people started looking for more efficient algorithms. However, there is a trade-off between the efficiency of the algorithm and the quality of the predicted image. Keeping this trade-off in mind, a lot of algorithms have been developed. These algorithms are called sub-optimal because though they are computationally more efficient than the full search, they do not produce better quality. Several block based fast motion estimation algorithms such as three step search TSS (Li et al 1994), efficient three step search (E3SS) (Xuan Jing and Chau Lap-Pui 2004) 4SS (Po and Ma 1996), block based gradient descent search (BBGDS), DS (Zhu and Ma 2000), cross diamond search (CDS) (Cheung and Po 2002) and HS (Anastasios Hamosfakidis and Yakup Paker 2002) have been proposed in recent years. The efficiency of such algorithms largely depends on the low-resolution coarse search and fine resolution inner search to recognize the slow and fast movements of the object. This chapter describes a hybrid search algorithm based on CDS and E3SS algorithms, which effectively detects the slow and fast movements with less computational time. Sum of Absolute Difference (SAD) and Mean Absolute Difference (MAD) are used as the matching criteria. The performance of the hybrid search scheme is evaluated in terms of PSNR as the fidelity measure, and computational complexity is measured based on the number of search points.

5.3 THE PROPOSED HYBRID ALGORITHM

In order to lessen the computational intricacy involved in CDS and E3SS for different types of video sequences (both slow movements and fast movements), a hybrid algorithm is proposed which involves less computational overhead for any type of video sequence. The pattern for the proposed algorithm is shown in Figure 5.1. It involves the pattern considered for both CDS and E3SS algorithms. Search pattern consists of two diamond patterns and two square patterns.
Figure 5.1 Search Pattern for the Proposed Algorithm

Figure 5.2 shows the detailed flow chart for the implementation of the proposed hybrid algorithm.
Divide the video clipping into frames

Take the reference frame (key frame) and the next (current) frame

Divide the frame into blocks of size 16x16

Select the search range

Check nine points in CSP in the center of search window

- Minimum BDM occurs in center of CSP.
  - Yes: Stop search and Motion vector is (0,0)
  - No: Check corner four points

- Check corner four points

  - If minimum BDM occurs at outer eight points
    - Yes: Perform E3SS and find motion vector
    - No: 2 new points closer to the min BDM of the center half LDSP are checked

  - Minimum BDM occurs at middle wing of CSP
    - Yes: stop search and MV is \((\pm 1,0)\) or \((0, \pm 1)\)
    - No: Perform as DS and find MV

Figure 5.2 Proposed hybrid algorithm flow chart
5.4 PERFORMANCE EVALUATION

The performance of the proposed hybrid scheme is evaluated in terms of number of search points, fidelity measures such as MAD, SAD, PSNR and total processing time. SAD, MAD and PSNR are defined by the Equations (5.1, 5.2, 5.3 and 5.4). Consider a block of pixels of size $M \times N$ in the reference frame, at a displacement of $(i,j)$, where $i$ and $j$ are integers with respect to the candidate block position.

$$SAD(i, j) = \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} |f_k(n_1, n_2) - f_{k-1}(n_1 + i, n_2 + j)|$$  \hspace{1cm} (5.1)

$$MAD(i, j) = \frac{1}{N^2} SAD(i, j) = \frac{1}{N^2} \left( \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} |f_k(n_1, n_2) - f_{k-1}(n_1 + i, n_2 + j)| \right) / N^2$$  \hspace{1cm} (5.2)

Peak-Signal-to-Noise-Ratio (PSNR) characterizes the motion compensated image that is created by using motion vectors and macro blocks from the reference frame and is given by equation

$$PSNR = 10 \log_{10} \left( \frac{(\text{Peak to peak value of original data})^2}{\text{MSE}} \right)$$  \hspace{1cm} (5.3)

$$PSNR_{\text{dB}} = 10 \log_{10} \left( \frac{2^n - 1}{\text{MSE}} \right)$$  \hspace{1cm} (5.4)

The proposed algorithm is tested with standard video sequences and the results are compared with CDS and E3SS algorithms respectively.
Table 5.1 Computations with Cost Function MAD for Carphone

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Frames</th>
<th>CDS</th>
<th></th>
<th>E3SS</th>
<th>Proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Number of Computation (NC)</td>
<td>Comp. Time in secs.</td>
<td>NC</td>
<td>Comp. Time in secs</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>29.98</td>
<td>27.42</td>
<td>23.30</td>
<td>21.94</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>30.00</td>
<td>26.85</td>
<td>23.32</td>
<td>22.58</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>29.99</td>
<td>26.73</td>
<td>23.30</td>
<td>22.26</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>30.12</td>
<td>26.68</td>
<td>23.32</td>
<td>22.76</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>30.10</td>
<td>26.81</td>
<td>23.32</td>
<td>23.07</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>30.07</td>
<td>26.63</td>
<td>23.32</td>
<td>21.81</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>30.09</td>
<td>26.75</td>
<td>23.32</td>
<td>23.84</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>29.90</td>
<td>26.84</td>
<td>23.32</td>
<td>23.73</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>30.01</td>
<td>26.98</td>
<td>23.32</td>
<td>22.81</td>
</tr>
</tbody>
</table>

Table 5.2 Computations with cost function SAD for Carphone

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Frames</th>
<th>CDS</th>
<th></th>
<th>E3SS</th>
<th>Proposed algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Number of Computation (NC)</td>
<td>Comp. Time in secs.</td>
<td>NC</td>
<td>Comp. Time in secs</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>28.86</td>
<td>25.94</td>
<td>23.25</td>
<td>22.38</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>29.27</td>
<td>26.05</td>
<td>23.23</td>
<td>22.29</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>29.24</td>
<td>25.87</td>
<td>23.22</td>
<td>21.95</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>29.67</td>
<td>26.48</td>
<td>23.27</td>
<td>22.19</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>30.05</td>
<td>26.69</td>
<td>23.28</td>
<td>22.19</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>29.92</td>
<td>26.59</td>
<td>23.30</td>
<td>22.28</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>29.74</td>
<td>26.31</td>
<td>23.27</td>
<td>22.76</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>29.50</td>
<td>26.47</td>
<td>23.29</td>
<td>22.64</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>29.48</td>
<td>26.35</td>
<td>23.25</td>
<td>22.64</td>
</tr>
</tbody>
</table>

Tables 5.1 and 5.2 show the computational complexity involved in computing the motion vector based on matching criteria MAD, SAD for the sequence Carphone. E3SS, CDS and the proposed algorithm are compared in the table.
Figure 5.3 shows the computational complexity versus the number of frames.

**Figure 5.3** Computational Complexity using the proposed algorithm for Akiyo sequence

**Figure 5.4** Computational Complexity using the proposed algorithm for Claire
Figure 5.5 Computational Complexity using the proposed algorithm for Carphone

The above Figures 5.3 to 5.5 show the computational complexity involved in the computation of motion vector using the different search algorithms for three different video sequences: Akiyo, Claire and Carphone. Matching criteria used are MAD and SAD. It can be seen from the figures that the proposed technique performs better than CDS and E3SS schemes in terms of the cost functions MAD and SAD, irrespective of the number of frames.
Figure 5.6  Average computational time for different motion estimation techniques (Akiyo)

Figure 5.7  Average computational time for different motion estimation techniques (Claire)
Figure 5.8  Average computational time for different motion estimation techniques (Carphone)

Figures 5.6 to 5.8 show the computational time requirement for the different motion estimation techniques. The proposed algorithm takes less time to estimate the motion vectors.

Figure 5.9  Average search points using first 50 frames
In order to further evaluate the performance of the proposed motion estimation technique, the number of search points has been calculated and Figure 5.9 shows the number of search points required for the computation of motion vectors based on the three algorithms CDS, E3SS and the proposed algorithm. Number of search points is comparatively less in the proposed algorithm than in E3SS and CDS.

5.5 TEMPORAL PREDICTION

Video can be considered as a polynomial of x, y and z. Interpolation and extrapolation automatically predict the polynomial and apply the position co-ordinates (x,y,z) to find the value of the pixel at the required location. The 3D interpolation and extrapolation is a mathematical prediction method, which helps to predict the data at a location in 3D space using a group of values that are already available in the same 3D space. Hence, all the pixel values in a required frame can be predicted using interpolation and extrapolation (Zhu and Belloulata 2004). Normally, adjacent frames in a video are highly correlated. So, some frames may be skipped during transmission and the interpolation and extrapolation concept can be used to predict these frames from the adjacent frames, more accurately with minimum prediction error. But this method becomes very ineffective when the frames are highly uncorrelated and this usually occurs when there is a change of scene or at the beginning of a new video sequence.

Lagrange’s interpolation is the most accurate one (Carnahan et al 1969). It generates a polynomial function that goes through all the available points. The prediction is more accurate if the order of the polynomial is higher. The order depends on the number of points. Few initial frames are extrapolated and using that interpolation is done. While considering both Newton method and Lagrange method with more samples, Lagrange is more
powerful than the other one. (Mathews 1992). After calculating the prediction error, encoding is done with Huffman encoding.

Reducing the number of key frames definitely gives good compression, since key frame is an independent frame and cannot be predicted from the previous frames. In a few long video sequences, adjacent frames are highly correlated hence only one key frame is enough. All other frames are predictable from the key frame and already predicted frames. Reducing key frames increases the number of frames in a package. The abrupt changes definitely need more key frames. And the abrupt changes in a video can be found from the correlation between the adjacent frames. To achieve higher compression, the number of frames within a package should not be fixed; it should be changed depending on the correlation between adjacent frames. So, the variable frame package method can be used instead of fixed frames per package method.

5.5.1 Algorithm steps

1. Convert a video sequence into 3D array
2. Perform RGB to gray conversion
3. Get the user information like number of frames to be used for interpolation, starting frame and the frame to be predicted
4. Drop the alternate frames from the video and insert a frame each with all zeros in these places and this video is considered as the cached video
5. Convert the cached video into frequency domain
6. Apply low pass filtering
7. Convert back to spatial domain and this results in the predicted frame

8. Calculate the prediction error ($P_e$).

5.5.2 Results

The algorithm has been tested with standard video sequences like Claire, Carphone, Mobile and Foreman. In each case, the predicted frame is determined and compared with the respective original frame and the prediction error is calculated.

Figure 5.10 Original 5th frame of flower sequence
Figure 5.11 Interpolated 5th frame of flower sequence

Figure 5.12 Prediction error – flower sequence
Figure 5.13  Prediction error histogram for flower sequence

Figures 5.10 to 5.13 show the original 5th frame, the interpolated 5th frame, the prediction error between the original and interpolated 5th frame and the histogram of the prediction error. Results show that the prediction error is very less.

Figure 5.14 Original 4th frame of mobile sequence
Figure 5.15 Interpolated 4\textsuperscript{th} frame of mobile sequence

Figure 5.16 Prediction error – 4\textsuperscript{th} frame of mobile sequence
Figures 5.14 to 5.17 show the original 4th frame, the interpolated 4th frame, the prediction error between the original and interpolated 4th frame and the histogram of the prediction error when the video input is mobile. Results show that the prediction error is very less.

Figure 5.18 Original 7th frame of carphone sequence
Figure 5.19 Interpolated 7th frame of carphone sequence

Figure 5.20 Prediction error – 7th frame of carphone
Figures 5.18 to 5.21 show the original 7th frame, the interpolated 7th frame, the prediction error between the original and interpolated 7th frame and the histogram of the prediction error. Results show that the prediction error is very less for the Carphone sequence also.

The algorithm has also been tested with four different video sequences downloaded from the Internet and in each case Compression Ratio (CR) has been determined. The results obtained are tabulated in Table 5.3. From the results obtained, it is observed that the compression using interpolation and extrapolation produces better compression ratio.
Table 5.3 Compression ratio obtained for different video sequences

<table>
<thead>
<tr>
<th>15 Frames from (640 x 480) video sequence</th>
<th>Video1</th>
<th>Video2</th>
<th>Video3</th>
<th>Video4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Sequence</td>
<td>13.2 MB</td>
<td>13.2 MB</td>
<td>13.2 MB</td>
<td>13.2 MB</td>
</tr>
<tr>
<td>Raw sequence after variable length coding</td>
<td>3.2 MB (C.R. 4.125)</td>
<td>3.7 MB (C.R. 3.567)</td>
<td>2.9 MB (C.R. 4.551)</td>
<td>2.8 MB (C.R. 4.714)</td>
</tr>
<tr>
<td>Compression using conventional method</td>
<td>2 MB (C.R. 6.6)</td>
<td>2.1 MB (C.R. 6.28)</td>
<td>3.1 MB (C.R. 4.258)</td>
<td>3.8 MB (C.R. 3.474)</td>
</tr>
<tr>
<td>Compression using interpolation and extrapolation</td>
<td>1.8 MB (C.R. 7.333)</td>
<td>2.5 MB (C.R. 5.28)</td>
<td>2.0 MB (C.R. 6.6)</td>
<td>2.8 MB (C.R. 4.714)</td>
</tr>
</tbody>
</table>

To have an understanding of the quality of the reconstructed signal, the original and reconstructed video sequences are shown in Figures 5.22 and 5.23 respectively. Advantages of the proposed concept are, it is a lossless compression technique hence video can be reproduced like the original video. Since interpolation and extrapolation are used, prediction frames are more accurate.
Figure 5.22  A video sequence from Ali Zee’s live concert (song: La isla bonita)
Figure 5.23 Reconstructed video sequence
5.6 CONCLUSION

This chapter discusses the estimation of motion vector using Hybrid search algorithm. The proposed algorithm is compared with existing two algorithms cross diamond search algorithm and effective three step search in terms of number of computations based on MAD and SAD as a cost function. The proposed algorithm involves less number of computations in calculating motion vector. So it is better in reducing computational load. The PSNR is calculated for the three algorithms CDS, E3SS and the proposed algorithm. The results show that the proposed algorithm is in no way degraded in quality in terms of MSE and PSNR.

Following this, interframe compression using interpolation and extrapolation was discussed. Different video sequences like Carphone, Flower and Mobile are considered. Interpolation is done using the proposed concept and the prediction error is calculated in each case. Interpolated frame is found to be the same as the original frame and the prediction error is very less. Less prediction error automatically results in bit saving. The prediction errors are to be properly encoded.

Finally, the performance was evaluated in terms of compression ratio and the proposed algorithm provides higher compression ratio than existing techniques. Using the variable frame package method can further increase the compression ratio, as it reduces the number of key frames required.

Patient video compression as an application for the proposed concepts is illustrated in the following chapter.