Computational Complexity Reduction for Motion Estimation in Video Compression

動画像圧縮における動き推定の演算量低減に関する研究

February, 2015

Graduate School of Global Information and Telecommunication Studies
Waseda University

Audiovisual Information Processing Research Ⅱ

Chang-Uk JEONG

TABLE OF CONTENTS

TABLE OF CONTENTS · · · · · I
LIST OF FIGURES · · · · · III
LIST OF TABLES · · · · · VI
Chapter 1. INTRODUCTION · · · · · 1
- 1.1. Research Background · · · · · 1
  - 1.1.1. Network Traffic and Digital Video Data · · · · · 1
  - 1.1.2. Video Compression Technologies · · · · · 2
  - 1.1.3. Video Encoder · · · · · 5
- 1.2. Research Objective · · · · · 7
- 1.3. Thesis Organization · · · · · 9
- 1.4. Research Direction and Area Covered by Thesis · · · · · 12

Chapter 2. RESEARCH TRENDS · · · · · 18
- 2.1. Motion Estimation in Video Compression · · · · · 18
- 2.2. Distortion Measurement Criteria · · · · · 20
- 2.3. Overview of Integer-Pixel Motion Estimation · · · · · 23
- 2.4. Overview of Fractional-Pixel Motion Estimation · · · · · 27

Chapter 3. INTEGER-PIXEL MOTION ESTIMATION · · · · · 30
- 3.1. Introduction · · · · · 30
- 3.2. Related Work · · · · · 31
- 3.3. Observation: Motion Vector Distribution · · · · · 33
- 3.4. Proposed Techniques · · · · · 35
  - 3.4.1. Revised Diamond Search Algorithm · · · · · 35
  - 3.4.2. Performance Evaluation of RDS · · · · · 37
  - 3.4.3. Diamond Web-Grid Search Algorithm · · · · · 39
  - 3.4.4. Performance Evaluation of DWS · · · · · 41
  - 3.4.5. Skipping Zero Motion Vector · · · · · 43
  - 3.4.6. Efficient Stationary Block Skip Method for VBSME · · · · · 46
  - 3.4.7. Experimental Results · · · · · 47
- 3.5. Summary · · · · · 49

Chapter 4. VERTICALLY SYMMETRICAL LINEAR MODEL BASED FME · · · · · 50
- 4.1. Introduction · · · · · 50
- 4.2. Previous Quadratic Prediction Based FME Algorithm · · · · · 51
- 4.3. Observation: Real-World Error Surfaces · · · · · 52
- 4.4. Observation: Simplified Error Surface of FME · · · · · 55

4.5. Proposed Linear Model Based FME Algorithm · · · 57
4.6. Experimental Results · · · 60
4.7. Summary · · · 63
Chapter 5. DATA TREND APPROXIMATION BASED INTERPOLATION-FREE FME · · · 64
5.1. Introduction · · · 64
5.2. Parabolic Models to Approximate Matching Errors · · · 65
5.3. Surface Modeling to Approximate Data Trends · · · 70
5.4. Quadratic Bézier Curve · · · 71
5.5. Proposed Method 1 · · · 73
5.6. Proposed Method 2 · · · 75
5.7. Proposed Method 3: Determination of Adjusting Factors · · · 76
5.8. Proposed Method 3: Error Cost Scaling · · · 79
5.9. Proposed Method 3: Modification for H.265/HEVC · · · 83
5.10. Experimental Results Based on H.264/AVC · · · 84
5.11. Experimental Results Based on H.265/HEVC · · · 85
5.12. Summary · · · 87
Chapter 6. ENHANCED 1-D PARABOLIC PREDICTION BASED FME · · · 99
6.1. Introduction · · · 99
6.2. Related Work · · · 100
6.3. Observation · · · 102
6.4. Application of Proposed Technique · · · 109
6.5. Experimental Results · · · 115
6.6. Summary · · · 118
Chapter 7. CONCLUSIONS · · · 119
REFERENCES · · · 121
LIST OF ACADEMIC ACHIEVEMENTS · · · 131

LIST OF FIGURES

Figure 1.1. Global consumer internet traffic. [Source: Cisco VNI, 2012] ············ 1
Figure 1.2. Macroblock partitions in H.264/AVC. ······························ 3
Figure 1.3. Coding tree unit structure in H.265/HEVC. ························· 4
Figure 1.4. Generic video encoder block diagram. ····························· 5
Figure 1.5. H.264/AVC encoder block diagram. [11] ··························· 6
Figure 1.6. H.265/HEVC encoder block diagram. [14] ·························· 6
Figure 1.7. Computational complexity distribution for H.264/AVC encoder. ········ 8
Figure 1.8. Computational complexity distribution for H.265/HEVC encoder. ······· 8
Figure 2.1. The principle of the block matching algorithm. ···················· 18
Figure 2.2. A search path example of the conventional full fractional-pixel search algorithm; the two-step refinement process is hierarchically performed from half-pixel (Step 1) to quarter-pixel (Step 2). ························ 27
Figure 3.1. The Search patterns used in UMHexagonS. (a) Large hexagon pattern, (b) Hexagon search pattern, and (c) Small diamond search pattern. ······· 31
Figure 3.2. The search process of UMHexagonS. ···························· 32
Figure 3.3. Motion vector distributions for (a) “Claire” and (b) “Stefan.” ·········· 33
Figure 3.4. Motion vector probability (%) distribution for three CCIR601s. ········· 34
Figure 3.5. Redefined search patterns from LDSP used in original DS. ············ 35
Figure 3.6. Two examples of search paths of RDS. ························· 36
Figure 3.7. The two new search patterns used in the proposed DWS algorithm: (a) Full diamond search pattern and (b) Dodecagon search pattern. ··········· 39
Figure 3.8. The search process of DWS. ···································· 40
Figure 3.9. (a) A ZMV block found in MV space. (b) The location of ZMV in search range = ±7. ·· 43
Figure 3.10. (a) ZMV ratio comparison. (b) MAE comparison. ···················· 44
Figure 3.11. The existing ZMV skip method. ································ 45
Figure 3.12. The seven variable thresholds used in the proposed ESBSM method and seven segmentations of the macroblock. ························· 46
Figure 3.13. The flowchart of the ME process combined with the proposed ESBSM implemented in the H.264/AVC JM reference software encoder. ······· 47
Figure 4.1. The three test sequences: (a) “Flower,” (b) “Football,” and (c) “Garden.” ·· 52
Figure 4.2. Error surfaces of IME at integer-pixel MV resolution. ················ 53
Figure 4.3. Error surfaces of FME at 1/16-pixel MV resolution. ·················· 54
Figure 4.4. (a) An example of FME error surface. (b) Another angle of (a). ········· 55

Figure 4.5. Average FME error surfaces simplified for 10, 100, and 1000 error surfaces at 1/16-pixel MV resolution. ···· 56
Figure 4.6. The basic concept of the proposed FME algorithm.···· 56
Figure 4.7. The nine integer-pixel search points reused in the proposed linear model based FME algorithm.···· 57
Figure 4.8. Rate-distortion curves for CIF sequence "Football." ···· 62
Figure 4.9. Computational complexity for CIF sequence "Football."···· 62
Figure 5.1. The five main integer-pixel search points (H_1, H_2, C, V_1, V_2) and the four relatively unimportant search points (U). The five main search points form the SDSP. ···· 65
Figure 5.2. Results of a simulation counting the number of known integer-pixel search points with their matching error cost by performing UMHexagonS. (a) Percentages of known search points within a 3×3 range of the local position for the QCIF "Salesman" sequence. (b) Percentages for the CIF "Football" sequence.···· 67
Figure 5.3. Examples of a 1-D parabolic model and a quadratic Bézier curve. (a) The two curves plotted using the three matching errors located at (-1, 5759), (0, 1659), (1, 5759). (b) The curves at (-1, 5759), (0, 1659), (1, 3146). ···· 72
Figure 5.4. Determination of the Bézier curve by controlling p_1'.···· 77
Figure 5.5. Examples of the error cost scaling. The best prediction x-coordinate of the original curve is the same as those of the two distorted curves scaled up and down vertically. ···· 79
Figure 5.6. Relationship between the average IME matching error and QP for the QCIF "Claire" sequence. The local position (X, 0) is represented as X.···· 80
Figure 5.7. Rate-distortion curves for the sequences in H.265/HEVC. ···· 98
Figure 6.1. Two error surfaces for the CCIR601 "Garden" sequence. (a) Error surface of IME at integer-pixel MV resolution. (b) Error surface of FME at 1/16-pixel MV resolution. ···· 100
Figure 6.2. The difference between the real-world FME error surface and the parabolic prediction model at 1/16-pixel MV resolution. ···· 102
Figure 6.3. The number of occurrences by the distance D between the best position of FFPS and that of PPFPS at 1/4-pixel MV resolution. D divided by 4 is represented as the fractional-pixel distance.···· 103
Figure 6.4. The ratios P_1/P_2 and P_2/P_1 by the distance D between the best position of FFPS and that of PPFPS at 1/4-pixel MV resolution.···· 103
Figure 6.5. The line fitting for the interval from 0 to 6 in Figure 6.4.···· 103
Figure 6.6. The frequency of the ratio $R = S_{max}/S_{min}$ for the "Foreman" sequence.···· 106

Figure 6.7. Modification of the distance prediction function. · · · · · · · · · · · · · · · · · · 107
Figure 6.8. Probability of D_p in the central range [0,±1]. · 108
Figure 6.9. PSNR and bitrate measurements by *K*. · 108
Figure 6.10. The illustration of the available way to apply the proposed technique to specific frames. · 109
Figure 6.11. The flowchart of CBFPS implemented in the H.264/AVC JM. · · · · · · · · · 110
Figure 6.12. The seven macroblock types used in H.264/AVC. · · · · · · · · · · · · · · · · · · · 110
Figure 6.13. Quarter-pixel motion vector probability distribution. · · · · · · · · · · · · · · · · 111
Figure 6.14. Error surface of fractional-pixel motion estimation. · · · · · · · · · · · · · · · · · 111
Figure 6.15. Examples of search process of (a) EFFPS, (b) FFPS, (c) SDS, and (d) BBGDS at 1/4-pixel MV resolution. · 112
Figure 6.16. The overall flow of the proposed FME algorithm. · · · · · · · · · · · · · · · · · · 114
Figure 6.17. Rate-distortion curves for the four test input sequences. · · · · · · · · · · · · · 116
Figure 6.18. The average ME time reduction and number of search points. · · · · · · · · 117

LIST OF TABLES

Table 1.1. Feature Comparison between H.264/AVC and H.265/HEVC · · · · · · · · · · · · 4
Table 1.2. Positioning of Each Chapter in the Thesis · 11
Table 1.3. Referenced Integer-Pixel Motion Estimation Algorithms · · · · · · · · · · · · · · 14
Table 1.4. Referenced Fractional-Pixel Motion Estimation Algorithms· · · · · · · · · · · · 15
Table 1.5. Relative Performance Comparison among IME Algorithms · · · · · · · · · · · · 16
Table 1.6. Relative Performance Comparison among FME Algorithms· · · · · · · · · · · · 17
Table 3.1. PSNR (dB) Comparison · 38
Table 3.2. Computational Complexity (SPT) Comparison· 38
Table 3.3. PSNR (dB) Comparison · 42
Table 3.4. Bitrate (bps) Comparison · 42
Table 3.5. Motion Estimation Time (ms) Comparison· 42
Table 3.6. Pseudo Code for Using ZMV Threshold in BMA · 45
Table 3.7. PSNR (dB) Comparison · 48
Table 3.8. Bitrate (bps) Comparison · 48
Table 3.9. Motion Estimation Time (ms) Comparison· 48
Table 4.1. Horizontal, Vertical, and both Groups Using the Nine Integer-Pixel Search Points · 58
Table 4.2. Quantization Operations for FME at 1/8-Pixel MV Resolution· · · · · · · · · · 59
Table 4.3. PSNR (dB) Performance Comparison at 1/4-Pixel MV Resolution · · · · · · · 61
Table 4.4. Bitrate (bps) Performance Comparison at 1/4-Pixel MV Resolution · · · · · 61
Table 4.5. PSNR (dB) Performance Comparison at 1/8-Pixel MV Resolution · · · · · · · 61
Table 4.6. Bitrate (bps) Performance Comparison at 1/8-Pixel MV resolution · · · · · · 61
Table 5.1. Fractional-pixel Motion Vector Matching Probability (%) · · · · · · · · · · · · · · 74
Table 5.2. Average IME Matching Error Costs · 77
Table 5.3. Pseudo-Code for the Third Proposed Method · 77
Table 5.4. Pseudo-Code for the Error Cost Scaling · 81
Table 5.5. Performance Comparison in H.264/AVC (QP=20) · · · · · · · · · · · · · · · · · · · 88
Table 5.6. Performance Comparison in H.264/AVC (QP=24) · · · · · · · · · · · · · · · · · · · 89
Table 5.7. Performance Comparison in H.264/AVC (QP=28) · · · · · · · · · · · · · · · · · · · 90
Table 5.8. Performance Comparison in H.264/AVC (QP=32) · · · · · · · · · · · · · · · · · · · 91
Table 5.9. Average Values with Respect to CBFPS (QPs=20, 24, 28, 32) · · · · · · · · · · · 92
Table 5.10. Performance Comparison in H.265/HEVC · 93
Table 5.11. Bjøntegaard Delta Performance Comparison in H.265/HEVC · · · · · · · · · · 97
Table 6.1. The Preset Correction Coefficients for Test Input Sequences · · · · · · · · · 105

Table 6.2. The Trend Analysis of the Ratio $R = S_{\max}/S_{\min}$ for Test Sequences · · · · · · 106
Table 6.3. The Matching Probability (%) between the Predicted Distance and the Real Distance · 107
Table 6.4. The Quarter-Pixel Motion Vector Matching Probability, Average Search Points, and Speedup Rate· 113

Chapter 1

INTRODUCTION

1.1. Research Background

1.1.1. Network Traffic and Digital Video Data

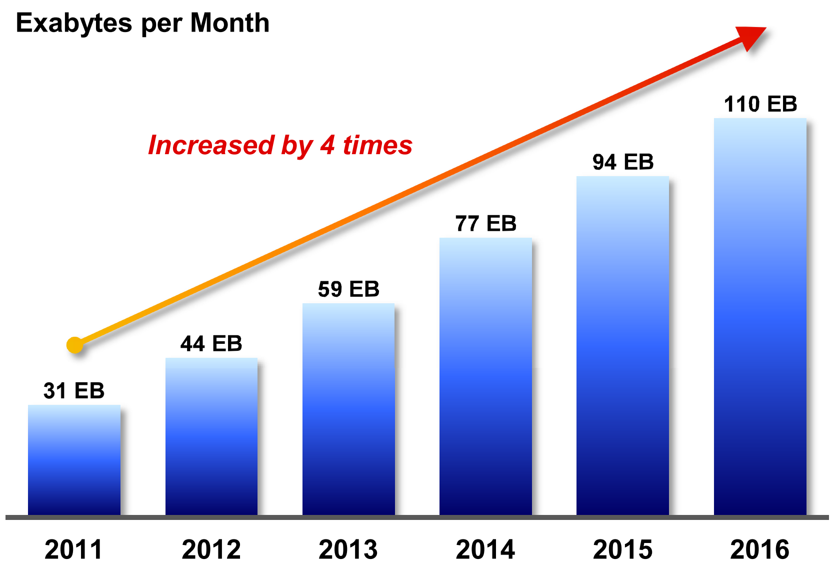


Fig. 1.1. Global consumer internet traffic. [Source: Cisco VNI, 2012]

Mobile network technologies such as the 3G, 4G, and Long Term Evolution (LTE) wireless standards have achieved rapid progress. Nevertheless, the transmission of large amounts of multimedia data increases consumer traffic dramatically on both wireless and wired networks. As shown in Fig. 1.1, global Internet traffic is expected to almost four times from 2011 to 2016, according to the Visual Networking Indexing Forecast from Cisco Systems, Inc. Internet video is now 40 percent of consumer Internet traffic, and will reach about 60 percent by the end of 2015. In addition, the sum of all video types will continue to be approximately 90 percent of global consumer traffic by 2015.

Thanks to the development of digital video compression technology, especially digital broadcasting has become more and more popular, such as standard definition television (SDTV) and high definition television (HDTV). Ultra high definition television (UHDTV) [1], [2], [3] proposed by NHK Science & Technology Research Laboratories is the latest digital video format for next-generation television beyond HDTV. UHDTV includes 4K UHDTV (3840 pixels by 2160 lines, 2160p) and 8K UHDTV (7680 pixels by 4320 lines, 4320p) which support frame rates up to 120 frames per second (fps). 8K UHDTV has 16 times the resolution of 1080p Full HDTV.

1.1.2. Video Compression Technologies

As video compression is the most significant attempt to reduce video data, it is a process by which digital signals are simplified by eliminating redundancy. Video coding technologies keep evolving along with massive growth in online video traffic. Video compression standards, such as MPEG-1 [1], MPEG-2 [5], MPEG-4 [6], H.261 [7], H.263 [8], [9], and H.264/MPEG-4 Advanced Video Coding (H.264/MPEG-4 AVC or H.264/AVC) [10], [11], have been developed through International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) and International Telecommunication Union Telecommunication Standardization Sector (ITU-T).

H.265/High Efficiency Video Coding (H.265/HEVC, HEVC/H.265, or HEVC) has recently been jointly developed by ISO/IEC MPEG and ITU-T VCEG [13], [14], [15]. H.265/HEVC has been designed for the benefit of network service providers and consumers. H.265/HEVC also focuses on improving video coding efficiency for high resolutions beyond HDTV. H.265/HEVC can achieve the bitrate savings more than half with respect to H.264/AVC, particularly over sixty percent for the 4K UHDTV and 1080p Full HDTV sequences [12].

Table 1.1 describes main features between H.264/AVC and H.265/HEVC. H.264/AVC adopted many advanced features such as variable block size motion estimation and motion compensation, multiple motion vectors per macroblock, multiple reference frames, quarter-pixel precision motion estimation and compensation, and context-adaptive binary arithmetic coding (CABAC) for entropy coding. In H.264/AVC, each frame to be encoded is divided into a grid of rectangles, called macroblocks. A macroblock normally consists of 16×16 pixels, and is subdivided into smaller blocks of from 8×8 to 4×4, as shown in Fig. 1.2.

H.265/HEVC applies various new video coding features to obtain much higher coding efficiency than H.264/AVC. The basic processing unit of H.265/HEVC is based on coding tree unit (CTU) structure in place of macroblock units in H.264/AVC, as shown in Fig. 1.3. All previous standards have been adopted the fixed array size of 16×16 luma samples, but H.265/HEVC supports variable size CTUs. Slice segments per picture are divided into a maximum of 64×64 or 128×128 sized largest coding units (LCUs). Each LCU is partitioned into coding units (CUs) varying in size from 64×64 to 8×8, which can be achieved as a recursive quadtree approach. In intra prediction coding, each CU then is split into 2N×2N or N×N, where N is the number of pixels, sized prediction units (PUs), while the PU types in inter prediction include four symmetric partitions (2N×2N, N×N, 2N×N, and N×2N) and four asymmetric partitions (2N×nU, 2N×nD, nL×2N, and nR×2N) by using asymmetric motion partition (AMP) [16] designed for irregular image patterns. The intra picture prediction of H.265/HEVC has 33 angular modes with planar and DC modes, whereas H.264/AVC uses only 9 intra prediction modes.

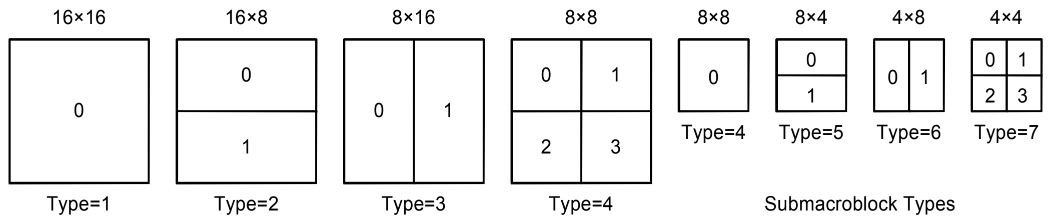


Fig. 1.2. Macroblock partitions in H.264/AVC.

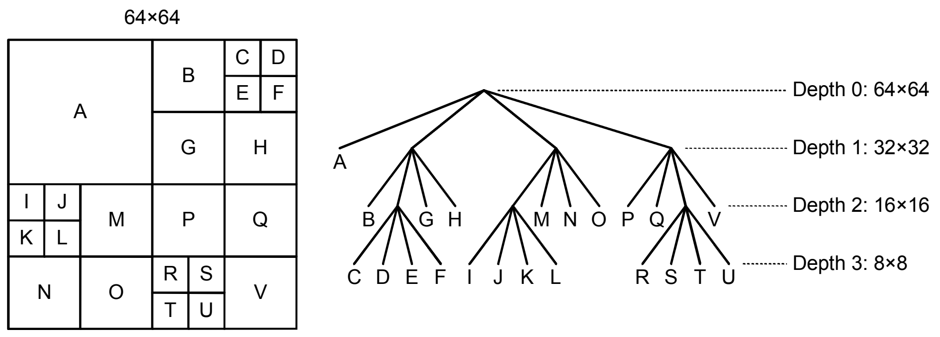


Fig. 1.3. Coding tree unit structure in H.265/HEVC.

Table 1.1. Feature Comparison between H.264/AVC and H.265/HEVC

Feature	**H.264/AVC**	**H.265/HEVC**
Improvement	Half or less bitrate of MPEG-2	Half or less bitrate of H.264/AVC
Basic processing unit	16×16 Macroblock	64×64 Coding tree unit
Block partitions	Variable block size: 16×16 to 4×4	Prediction units: 64×64 to 8×8
Motion vector accuracy	Quarter-pixel	Quarter-pixel
Transform	8×8 and 4×4 DCT approximation	Transform units: 32×32 to 4×4
Entropy coding	CAVLC and CABAC	CABAC (Parallel operations)
Intra prediction	9 modes	35 modes
Inter prediction	Spatial MV prediction (Median) & Direct mode	Advanced MV prediction (Spatial + Temporal) & MERGE mode
Inter prediction interpolation	Luma 6-tap + 2-tap	Luma 7-tap or 8-tap
Built-in deblocking filter	In-loop deblocking filter	In-loop deblocking filter & SAO filter
Maximum resolution and fps	Up to 4096×2304 at 59.94 fps	Up to 8K UHDTV at 120 fps
Encoding runtime	10 times complexity of MPEG-2	10 times complexity of H.264/AVC
Main drawback	High bitrate for UHDTV	High encoding complexity

1.1.3. Video Encoder

Generally, a video encoder can be divided into three units: a temporal redundancy eliminator, a spatial redundancy eliminator, and a statistical redundancy eliminator (entropy encoder), as shown in Fig. 1.4 [17]. The temporal redundancy eliminator estimates and extracts the motion of an object using the close correlation between neighboring video frames, while information related to stationary objects or background is eliminated. As motion estimation (ME) is at the core of the temporal model, it occupies more than half of the total encoding time [18]. Thus, a great deal of research into ME has been conducted in an attempt to decrease the runtime of the ME module.

The H.264/AVC encoder as shown in Fig. 1.5 [11] is twice as efficient as that of MPEG-2. H.264/AVC uses a lower bitrate to get the same quality with MPEG-2. Although the advanced features allow it to encode video data more effectively compared with conventional methods, the increased computational complexity requires a certain level of CPU power to perform real-time video encoding, especially in mobile applications.

The H.265/HEVC encoder as shown in Fig. 1.6 [14] can provide much higher video coding efficiency compared to the H.264/AVC encoder by halving the bitrates while maintaining almost the same subjective quality [12], but this is achieved at the expense of a significant increase in computational complexity [15]. It is known that H.265/HEVC coding requires up to 10 times more processing power compared with H.264/AVC.

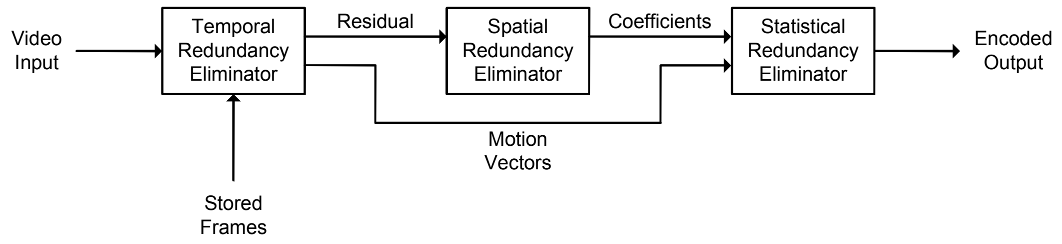

Video Input
Temporal Redundancy Eliminator
Residual
Spatial Redundancy Eliminator
Coefficients
Statistical Redundancy Eliminator
Encoded Output
Motion Vectors
Stored Frames

Fig. 1.4. Generic video encoder block diagram.

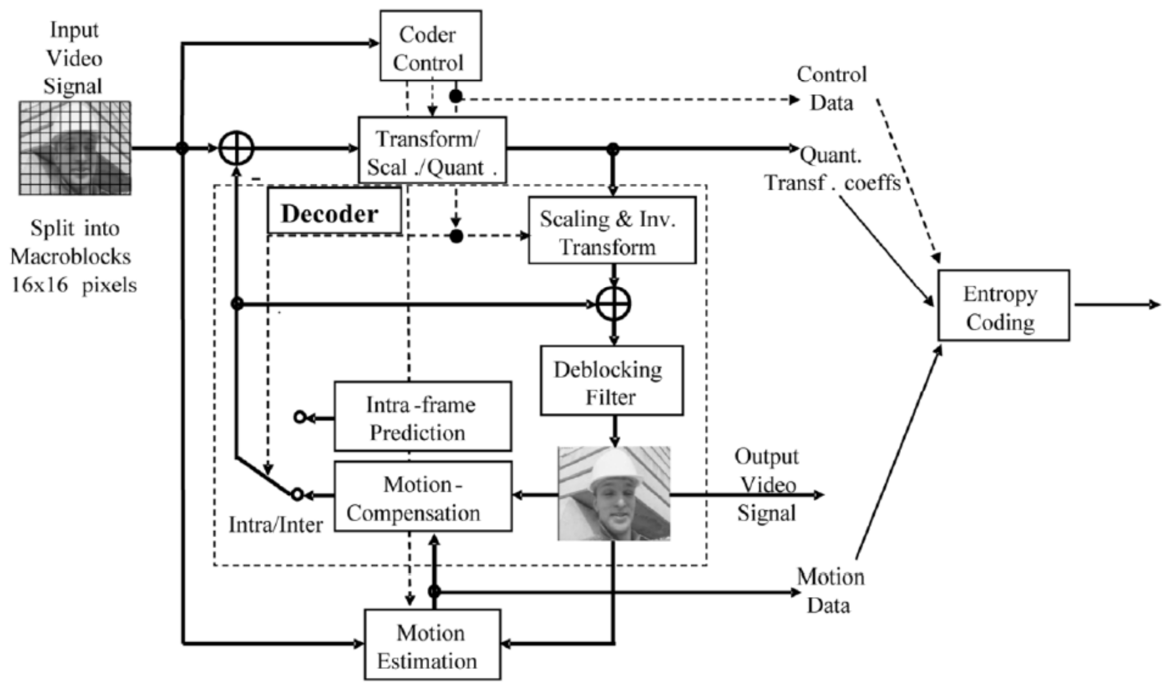


Fig. 1.5. H.264/AVC encoder block diagram. [11]

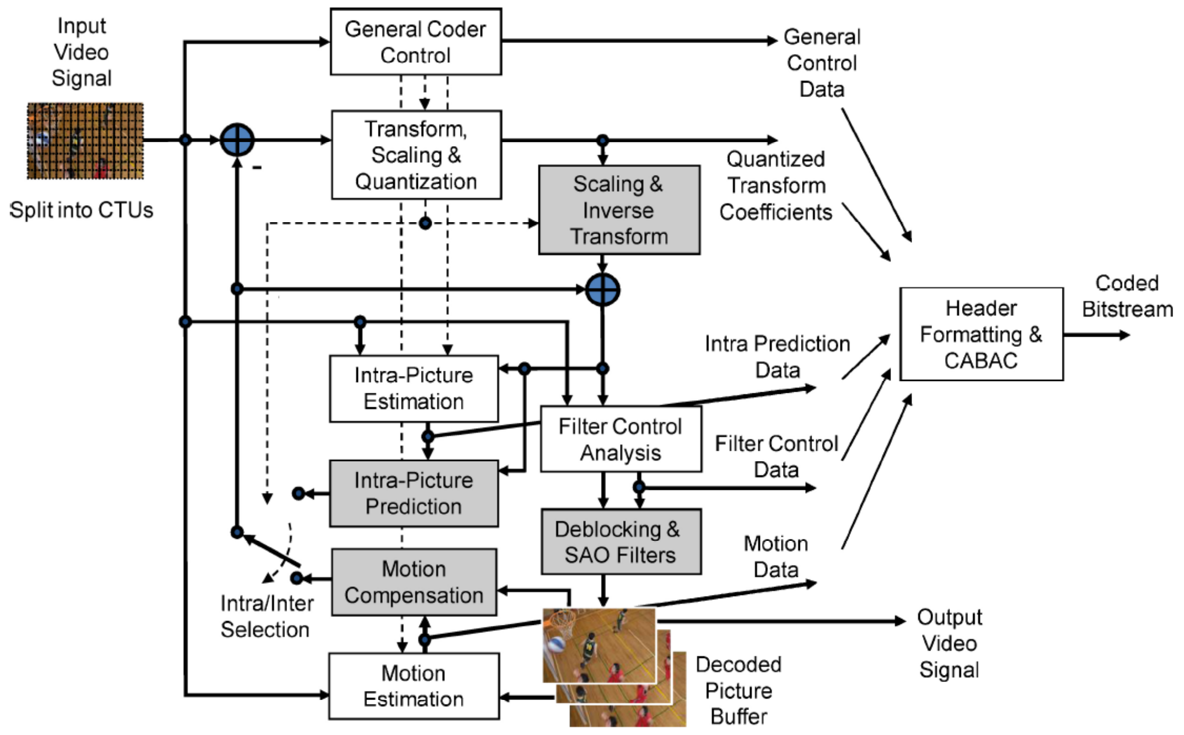


Fig. 1.6. H.265/HEVC encoder block diagram. [14]

1.2. Research Objective

The next emerging UHDTV technologies are being developed rapidly thanks to enormous success and popularity of HDTV and the Internet. However, high quality video streaming services including HDTV and UHDTV have been the major cause current network traffic jam. Video coding standards such as H.264/AVC and H.265/HEVC were released to decrease the massive traffic. Video compression standards have also contributed powerfully to various multimedia applications such as broadcasting, interactive communications, digital storage media, medicine, video on demand (VOD), video surveillance system, and so on. Particularly, H.265/HEVC enables broadcasters to sharply reduce the bandwidth required to deliver UHDTV, whereas it has a much more increased computational complexity in encoding due to many advanced features, compared with previous standards. For this reason, it requires much more powerful hardware with an additional expense.

Fig. 1.7 [17] and Fig. 1.8 [15] illustrate the computational complexity profiles of H.264/AVC and H.265/HEVC encoding, respectively. As shown in Fig. 1.7, “mv_search.c,” which carries out the ME process, accounts for over 60 percent of the total encoding time of H.264/AVC. Similarly, Fig. 1.8 shows that the complexity of “TComRdCost,” “TComInterpolationFilter,” and “TEncSearch,” which are encoding classes related to the ME process, is over 60 percent of the total encoding time of H.265/HEVC. ME has been adopted in most modern video compression standards including H.264/AVC and H.265/HEVC because it is a significant core part to efficiently reduce a large amount of video data by extracting motion vector information between adjacent frames. As mentioned above, the high complexity of ME greatly affects the processing speed of the latest video encoders and it will be a critical factor in real-time hardware or software applications for video encoding. Accordingly, many researches on ME have been conducted for a long time. In this research, the focus is on the development of low complexity ME techniques.

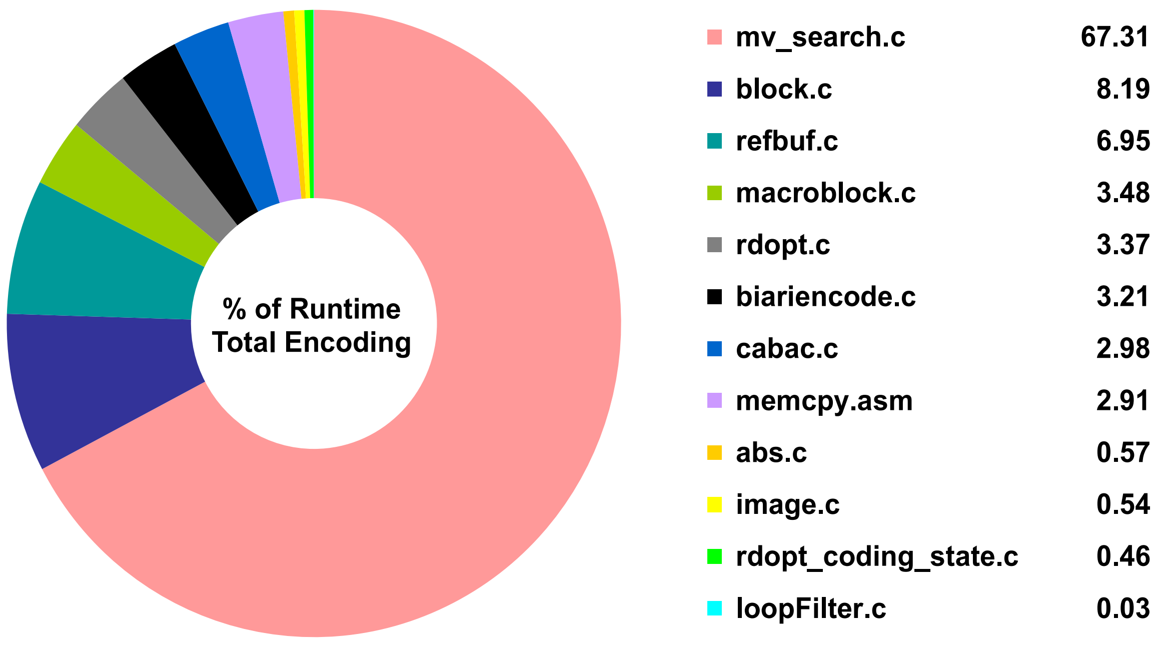


Fig. 1.7. Computational complexity distribution for H.264/AVC encoder.

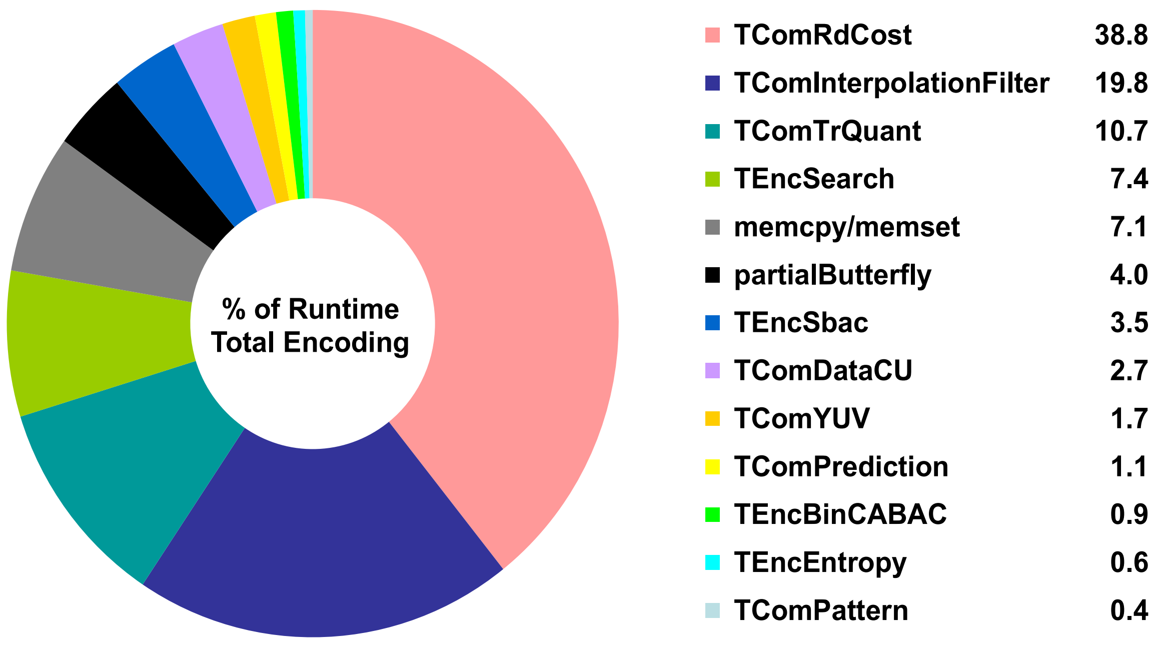


Fig. 1.8. Computational complexity distribution for H.265/HEVC encoder.

1.3. Thesis Organization

The dissertation consists of seven chapters, which is organized in the following sequence: Chapter 1. INTRODUCTION; Chapter 2. RESEARCH TRENDS; Chapter 3. INTEGER-PIXEL MOTION ESTIMATION; Chapter 4. VERTICALLY SYMMETRICAL LINEAR MODEL BASED FME; Chapter 5. DATA TREND APPROXIMATION BASED INTERPOLATION-FREE FME; Chapter 6. ENHANCED 1-D PARABOLIC PREDICTION BASED FME; Chapter 7. CONCLUSIONS.

In Chapter 1, a state-of-the-art video compression standard such as H.264/AVC and H.265/HEVC are introduced. Although the advanced technologies allow it to encode video data more effectively compared with conventional methods, the increased computational complexity requires a certain level of CPU power to perform real-time video encoding, especially in mobile applications. The complexity of the data structure and algorithm of an H.265/HEVC encoder is higher than 10 times an H.264/AVC encoder. The ME runtime of a video encoder is recognized as the greatest portion of overall video encoding time. Accordingly, the ME technique will be very significant attempt for alleviating the computing complexity in H.265/HEVC.

In Chapter 2, various integer-pixel ME (IME) algorithms and fractional-pixel ME (FME) algorithms are reviewed and introduced. Over the past few years, many fast IME algorithms have been developed to replace the conventional full search algorithm with a high level of computational complexity. Block-matching algorithm is the most well-known IME method in video coding. The position of the motion of an object in video sequence can be represented as fractional-pixel as well as integer-pixel. Although FME has a strong impact on peak signal-to-noise ratio (PSNR) of the coded images, the total video encoding time is much more increased because it is performed after the IME process is terminated. The FME process also makes frequent memory access and a certain amount of computation due to interpolation operations, which are needed to produce fractional-pixel search positions.

In Chapter 3, some block-matching based techniques for fast IME are proposed. The existing fast IME algorithms utilize a variety of search strategies to speed up the search process. The conventional high-performance hybrid ME algorithm adopted in the H.264/AVC reference software encoder combines many improved technologies, but there is the potential for speed improvement. Therefore, a modified hybrid ME algorithm is presented to further reduce the computational complexity of the conventional method.

In Chapters 4, 5, and 6, interpolation-free prediction techniques for fast FME are proposed. In both H.264/AVC and H.265/HEVC, the typical full fractional-pixel search (FFPS) at quarter-pixel motion vector (MV) resolution always checks 16 fractional-pixel search positions. The exhaustive search method used in FFPS provides the excellent search quality whereas it causes a serious waste in computational complexity. Thus, low complexity interpolation-free based FME techniques are presented in the thesis. These techniques focus on performing FME without interpolation operations in order to minimize the use of fractional-pixel search points. In Chapters 4 and 5, although the proposed methods overcome the shortcomings of the existing prediction based FME methods, the reconstructed image quality and reduced bitrate ratio are not fully satisfactory. In Chapter 6, therefore, a novel technique using specific correction coefficients is also introduced. The technique can further improve the performance of the existing methods, whereas it requires a small number of fractional-pixel search points in computational complexity.

Finally, Chapter 7 concludes and summarizes the thesis. The future works related to this research field will also be discussed shortly in the final chapter. In addition, the chapters in the thesis can be positioned as described in Table 1.2, of which the abbreviations are listed in Table 1.3 and Table 1.4. The positioning of all the ME algorithms mentioned is represented in Table 1.5 and Table 1.6; they are also explained briefly in Chapter 2.

Table 1.2. Positioning of Each Chapter in the Thesis

Research Field		**Video Coding Standard**	
		H.264/AVC	**H.265/HEVC**
Video Encoder	**Integer-Pixel ME**	**Chapter 3** **3.4.1** Proposed: RDS [78] Competition: DS [60] HEXBS [61] CDS [62]	N/A
Video Encoder	**Integer-Pixel ME**	**Chapter 3** Proposed: EDWS [78] Competition: FFS [66] UMHexagonS [65]	N/A
Video Encoder	**Fractional-Pixel ME**	**Chapter 4** Proposed: VSLFPS [79], [80] Competition: QPFPS (1-D_PM) [69] FFS [66] (IME)	**Chapter 5** Proposed: IFBFPS (METHOD_3) [81], [82] Competition: FFPS [66], [67] PPFPS (1-D_PM) [68], [69], [70], [71]
Video Encoder	**Fractional-Pixel ME**	**Chapter 5** Proposed: IFBFPS (METHOD_1-3) [81], [82] Competition: CBFPS [65] PPFPS (1-D_PM) [68], [69], [70], [71]	
Video Encoder	**Fractional-Pixel ME**	**Chapter 6** Proposed: EPPFPS [83] Competition: FFPS [66], [67] CBFPS [65] PPFPS (1-D_PM) [68], [69], [70], [71]	

1.4. Research Direction and Area Covered by Thesis

All the block-based integer-pixel and fractional-pixel motion estimation (IME and FME) algorithms mentioned in the thesis are listed in Table 1.3–Table 1.6. Each ME algorithm will be briefly explained in Chapter 2. As shown in Table 1.5 and Table 1.6, the MV search speed and search accuracy comparisons among the algorithms are dependent heavily on the experimental results in the referenced papers, and it is difficult to compare precisely their performances because the test conditions, encoding configurations, and implementation environments for each simulation are different from each other. Therefore, the performance evaluation of the ME algorithms may be somewhat subjective. Table 1.5 and Table 1.6 represent simply the unit of search speed as five grades such as "Exhausted," "Very slow," "Slow," "Fast," and "Very fast," whereas that of search accuracy is represented as "Very low," "Low," "High," "Very high," and "Lossless." "lossless" in the Tables means lossless ME approach fully using all the search points required for finding the best matched MV, whereas "lossy" represents lossy ME approach partially using them for speeding up — in other words, when an ME algorithm provides the best quality at all times, it is regarded as "lossless"; otherwise, "lossy."

As shown in Table 1.5, the conventional full search (FS) [66], [67] algorithm can always produce the best matched MVs at the slowest search speed, while the small diamond search (SDS) [63], [65] has poor performance in search accuracy but its processing speed is very fast. SDS, for this reason, is employed complementally to refine the best MV in the final search step of some fast ME algorithms such as the efficient three-step search [63] and the unsymmetrical-cross multi-hexagon-grid search (UMHexagonS) [65]. UMHexagonS, the test zone search (TZS) [67], and the edge based partial distortion search (EPDS) [43] give high performance with relatively low computational cost. On the other hand, in the case of FME as shown in Table 1.6, the exhaustive full fractional-pixel search (EFFPS) [see Fig. 6.15 (a)] provides the best search quality of all the FME algorithms. Nevertheless, since EFFPS has a significantly high computational complexity and does not consider hierarchical search by fractional-pixel resolution change, it has not been adopted in most of the modern video encoders; alternatively, the two-step full fractional-pixel search (FFPS) [66], [67] has replaced EFFPS up to now, in spite of its lossy ME approach. On the contrary, the parabolic prediction based fractional-pixel search (PPFPS) [68], [69], [70], [71] without using additional fractional-pixel search points has an extremely low computational cost, whereas the performance of prediction is insufficiently competitive compared with the other FME algorithms reliant on interpolation.

Chapters 3–6 in the thesis propose some integer-pixel and fractional-pixel ME techniques, respectively. The proposed ME algorithms are developed along with the following three considerations: first, they should be compliant with the recent video compression standard such as H.264/AVC or H.265/HEVC; second, they should gain a competitive advantage over the existing high-performance algorithms; third, they should not limit their practical applications, for instance, consider implementation on mobile devices. The experimental results in Chapter 3 demonstrate the performance among the proposed IME algorithms and the anchor IME algorithms, adopted in the reference software encoder of H.264/AVC, such as the fast full search (FFS) [66] and UMHexagonS. In Chapters 4–6, the simulations compare the performance among the proposed FME techniques, FFPS, and the center-biased fractional-pixel search (CBFPS) [65]; FFPS is used as a reference fractional-pixel ME algorithm in both H.264/AVC and H.265/HEVC, whereas the reference integer-pixel ME algorithms, CBFPS and the test zone search (TZS) [67], are only for H.264/AVC and H.265/HEVC respectively. All the other ME algorithms used for the simulations in the thesis are directly implemented based on the reference software encoders. In the light of the observations above, the thesis will make various attempts to further improve the existing fast integer-pixel and fractional-pixel ME algorithms such as UMHexagonS and PPFPS.

Table 1.3. Referenced Integer-Pixel Motion Estimation Algorithms

Approach	Abbr.	Full Name	Ref. No.
Lossless	FS	Full Search	[66], [67]
	FFS	Fast Full Search	[66]
	PDS	Partial Distortion Search	[36], [37]
	FFSSG	Fast Full Search with Sorting by Gradient	[42]
	FFSSD	Fast Full Search with Sorting by Distortion	[42]
	SEA	Successive Elimination Algorithm	[44]
	MSEA	Multilevel Successive Elimination Algorithm	[46]
	ASO	Adaptive Search Order	[47]
Lossy	NPDS	Normalized Partial Distortion Search	[40]
	APDS	Adjustable Partial Distortion Search	[41]
	EPDS	Edge Based Partial Distortion Search	[43]
	3SS	Three-Step Search	[48]
	N3SS	New Three-Step Search	[57]
	4SS	Four-Step Search	[58]
	BBGDS	Block-Based Gradient Descent Search	[59]
	DS	Diamond Search	[60]
	HEXBS	Hexagon-Based Search	[61]
	CDS	Cross-Diamond Search	[62]
	E3SS	Efficient Three-Step Search	[63]
	CDHS	Cross-Diamond-Hexagonal Search	[64]
	SDS	Small Diamond Search	[63], [65]
	EHS	Extended Hexagon-Based Search	[65]
	UMHexagonS	Unsymmetrical-Cross Multi-Hexagon-Grid Search	[65]
	TZS	Test Zone Search	[67]
	ENS	Eight Neighbor Search	[71]
Proposed Lossy	RDS	Revised Diamond Search	[78]
	EDWS	Diamond Web-grid Search (DWS) Combined with Efficient Stationary Block Skip Method	[78]

Table 1.4. Referenced Fractional-Pixel Motion Estimation Algorithms

Approach	**Abbr.**	**Full Name**	**Ref. No.**
Lossless	EFFPS	Exhaustive Full Fractional-Pixel Search	Fig. 6.15
Lossy	FFPS	Full Fractional-Pixel Search	[66], [67]
	CBFPS	Center-Biased Fractional-Pixel Search	[65]
	PPFPS (1-D_PM)	Parabolic Prediction Based Fractional-Pixel Search (One-Dimensional PPFPS)	[68], [69], [70], [71]
	QPFPS	Quadratic Prediction Based Fractional-Pixel Search (QPFPS not using the modified SDS in its final search step is the same with 1-D_PM.)	[69]
	PDFPS	Prediction Based Directional Fractional-Pixel Search	[72]
	MPFPS	Motion Prediction Fast Fractional-Pixel Search	[73]
	FEFPS	Fast and Efficient Fractional-Pixel Search	[74]
Proposed Lossy	VSLFPS	Vertically Symmetrical Linear Model Based Fractional-Pixel Search	[79], [80]
	IFBFPS (METHOD_1-3)	Interpolation-Free Quadratic Bézier Spline Based Fractional-Pixel Search	[81], [82]
	EPPFPS	Enhanced 1-D Parabolic Prediction Based Fractional-Pixel Search	[83]

Table 1.5. Relative Performance Comparison among IME Algorithms

Performance Eval.		**Integer-Pixel MV Search Speed**					
		Exhausted	Very Slow	Slow	Fast		Very Fast
Integer-Pixel MV Search Accuracy	Lossless	FS	FFS PDS SEA FFSSG FFSSD	MSEA ASO			
	Very High				EPDS UMHexagonS TZS		***EDWS***
	High			NPDS	APDS		
	Low			N3SS 3SS 4SS	BBGDS DS E3SS	***RDS*** EHS HEXBS CDS ENS	CDHS
	Very Low						SDS

Chap. 3

Sect. 3.4.1

The target area in the thesis is indicated as shown in this formatted text.

Table 1.6. Relative Performance Comparison among FME Algorithms

Performance Eval.		**Fractional-Pixel MV Search Speed**				
		Exhausted	Very Slow	Slow	Fast	Very Fast
Fractional-Pixel MV Search Accuracy	Lossless	EFFPS				
	Very High		FFPS			***Chap. 6***
	High			CBFPS	QPFPS PDFPS MPFPS	***EPPFPS***
	Low			FEFPS	***Chap. 5***	***IFBFPS***
	Very Low				***Chap. 4***	***VSLFPS*** PPFPS

The target area in the thesis is indicated as shown in this formatted text.

Chapter 2

RESEARCH TRENDS

2.1. Motion Estimation in Video Compression

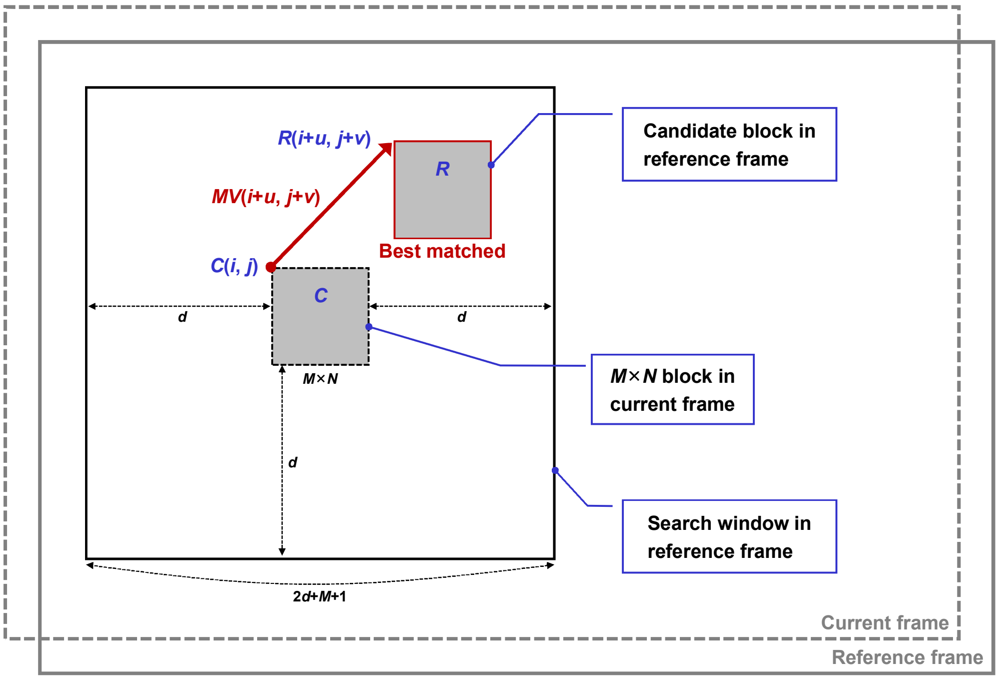

R(i+u, j+v)
R
MV(i+u, j+v)
Best matched
C(i, j)
C
d
d
M×N
d
2d+M+1
Candidate block in reference frame
M×N block in current frame
Search window in reference frame
Current frame
Reference frame

Fig. 2.1. The principle of the block matching algorithm.

Motion estimation (ME) has been used to efficiently eliminate the temporal redundancy in video compression. ME is also developed for different types of applications such as image sequence analysis, video coding, computer vision, and image stabilization. The existing ME techniques can be roughly categorized into time-domain and frequency-domain algorithms. The time-domain algorithms include matching algorithms and gradient-based (recursive) algorithms. The frequency-domain algorithms, meanwhile, include phase correlation algorithms, discrete cosine transform (DCT) matching algorithms, and the wavelet transform matching algorithm [19].

In addition, motion estimation techniques can be classified into pixel recursive approach [20], [21], [22], [23] and block matching approach [24] according to their elementary units, i.e., pixels or blocks. Pixel recursive algorithms try to calculate the displacement of each pixel separately. There are some merits of pixel recursive approach; it can cope with the problem of multiple moving objects and does not have any overhead for transmitting motion information. However, pixel recursive approach recursively employs the luminance change for each pixel in order to seek the motion information of objects, and therefore the decoder requires the expense of high computational complexity, almost the same as the encoder. Pixel recursive approach also has a relative weakness in noise, discontinuities in the motion field, and large displacements [25], [26].

On the other hand, block matching algorithms have been adopted by many reference video encoders due to their low computational cost and robustness to errors. Various block matching based ME algorithms are briefly introduced in section 2.3. In block matching, each block in the current frame is compared with the search area in the reference frame in order to find the best matched block pointed to by motion vector, and then only the motion information is transmitted to perform video data compression, as described in Fig. 2.1. The best motion vector corresponds to the block position with the minimum distortion which can be computed by block distortion measurement (BDM) criteria such as the sum of absolute differences (SAD), the mean absolute error (MAE), the mean squared error (MSE), and the sum of absolute transformed differences (SATD) [17], [19]. SAD and MAE have been commonly used as fast video quality metric for block matching ME due to their simplicity.

2.2. Distortion Measurement Criteria

The sum of absolute differences (SAD) is the most frequently used BDM criterion because of its low complexity. The SAD function used for block matching ME can be computed as follows:

$$SAD(u,v) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |C(i,j) - R(i+u, j+v)| \tag{1}$$

where $M \times N$ denotes the macroblock size and $C(i, j)$ and $R(i+u, j+v)$ represent current and reference area samples respectively. The best motion vector (u, v) is determined by the candidate having the lowest matching error SAD value. It is effectively the simplest metric that calculates every pixel in a block. For low complexity implementation, the constant division by $M \times N$ in the mean absolute error (MAE) criterion shown in (2) is excluded from the SAD function.

$$MAE(u,v) = \frac{1}{M \times N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |C(i,j) - R(i+u, j+v)| = \frac{SAD(u,v)}{M \times N} \tag{2}$$

MAE is a widely used cost function along with SAD. The average distortion value produced by MAE represents the differences between the current block and the reference block but is unconcerned about the relationship between the pixel values in the blocks. In some cases, the MAE function may cause suboptimal results in fast ME algorithms because its performance worsens as the search range becomes wider due to multiple local minima [27], [28]. On the other hand, the sum of squared differences (SSD), also known as the sum of squared errors (SSE), and the mean squared error (MSE) are generally considered to provide better results compared to SAD and MAE respectively, as they can be explained as Euclidean distance between two samples, which is similar to the human visual perception. However, the SSD and MSE functions require much higher computational complexity due to one multiplication per pixel difference, as shown below.

$$SSD(u,v) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(C(i,j) - R(i+u, j+v))^2 \tag{3}$$

$$MSE(u,v) = \frac{1}{M \times N}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(C(i,j) - R(i+u, j+v))^2 = \frac{SSD(u,v)}{M \times N} \tag{4}$$

The sum of absolute transformed differences (SATD) is a well-known block matching metric used in the recent video compression standards such as H.264/AVC and H.265/HEVC, which is used for fractional-pixel refinement and intra prediction mode decision in the recent video encoders. The SATD function works by applying a Hadamard transform (HT) of the differences between the pixels in the current macroblock and the corresponding pixels in the reference block, as described in (5) and (6). Even though SATD produces superior prediction quality from both the perspectives of subjective and objective metrics, it has far more increased computational complexity than that of SAD.

$$SATD(u,v) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}| HT(C(i,j) - R(i+u, j+v)) | \tag{5}$$

$$HT(Diff) = M\left[Diff\right]M^T, \quad M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{6}$$

Rate–distortion optimization (RDO) [29] can be utilized to enhance the quality performance in video encoding. The RDO technique was developed to effectively reduce the amount of video data with minimal quality loss. That is, RDO explains the tradeoff between bitrate and quality. In a generic video encoder the discrete cosine transform (DCT) is performed, followed by the quantization and entropy encoding process. For that, RDO requires much heavier computational burden than those of other video quality metrics, such as SAD, SSD, and SATD. It is used for inter prediction mode selection in H.264/AVC and coding unit (CU) depth decision in H.265/HEVC, which are performed after the final step of the ME process. The rate–distortion cost (RDcost) formula is defined as shown in the following equation.

$$RDcost = Distortion + \lambda_{mode} \times Rates \tag{7}$$

where *Distortion* indicates block matching metric such as SAD, SSD, and SATD, λ_{mode} is the Lagrangian multiplier for adjusting the tradeoff between distortion and bitrate, *mode* represents the available modes in inter coding, and *Rates* denotes the number of bits needed to be encoded. In the RDcost formula, SAD is generally used as the BDM criterion for integer-pixel ME, whereas SATD is applied to the fractional-pixel refinement process.

Peak signal-to-noise ratio (PSNR) is very well-known as a quantitative image quality metric for evaluating the distortion of reconstructed image. Particularly, the luminance PSNR (Y-PSNR) is recognized to be more appropriate for the measurement of visual quality than that of chrominance (Cb and Cr) components [30]. The formula for PSNR, where *n* is the number of bits per sample, can be easily calculated based on the MSE criterion as follows.

$$PSNR = 10 \times \log_{10}\left(\frac{(2^n - 1)^2}{MSE}\right) \tag{8}$$

Bjøntegaard delta matric [31] is widely used to effectively represent the average difference between two rate-distortion curves consisting of pairs of PSNR and bitrate. In the research on the recent video coding standards such as H.264/AVC and H.265/HEVC, Bjøntegaard delta PSNR and bitrate (BD-PSNR and BD-Bitrate) provide objective values for close analysis of simulation results at various quantization parameters (QPs). To evaluate BD-PSNR and BD-Bitrate, the normal rate-distortion curves passing through four data points can be interpolated by applying a third order logarithmic polynomial, as given below.

$$F_{PSNR} = F(x) = c_0 x^3 + c_1 x^2 + c_2 x + c_3, \quad x = \log(BIT) \tag{9}$$

$$F_{Bitrate} = F(y) = c_0 y^3 + c_1 y^2 + c_2 y + c_3, \quad y = \log(SNR) \tag{10}$$

where $F(x)$ and $F(y)$ denote the interpolation curves for BD-PSNR and BD-Bitrate respectively, c_0, …, c_3 are the coefficients for line fitting, and x and y are the logarithms of the output bitrate and PSNR (*BIT* and *SNR*) respectively. The BD-PSNR value over the whole range of bitrates [*bl*, *bu*] is expressed as the average difference ΔI_{PSNR} between the integrals of the two interpolation curves $F_A(x)$ and $F_B(x)$, and in the same way $\Delta I_{Bitrate}$ in BD-Bitrate is computed with the PSNR interval [*sl*, *su*], as shown in (11) and (12) respectively.

$$BD\text{-}PSNR = \Delta I_{PSNR} = \frac{\int_{bl}^{bu} (F_B(x) - F_A(x))dx}{bu - bl} \tag{11}$$

$$BD\text{-}Bitrate = \Delta I_{Bitrate} = \frac{\int_{sl}^{su} (F_B(y) - F_A(y))dy}{su - sl} \tag{12}$$

2.3. Overview of Integer-Pixel Motion Estimation

The conventional full search (FS) [66], [67] algorithm, also known as the exhaustive search algorithm, can achieve the optimal search result and excellent rate-distortion performance because it performs brute-force distortion measurement against every block within the search window in the reference frame. As the FS algorithm also realizes simple data flow and uncomplicated logical operation circuit, it can be easily implemented on hardware devices. Implementation examples of ME architecture based on FS are shown in [32], [33], [34]. However, if the video frame size or the search range in encoding configuration is greatly increased, FS becomes extremely wasteful in terms of computation. The FS process used for ME is known to occupy about 60 to 80% of the total encoding time [35], and the overwhelming computational load makes it often inappropriate for real-time video coding applications. For this reason, in order to replace the exhaustive FS method, various fast integer-pixel ME techniques have been developed over the past few years. Most of the alternative ME algorithms to FS can be grouped into two main techniques, fast exhaustive search and partial ones. Fast exhaustive search approach, which is commonly using all or almost all of the search points within a search range, performs faster than the conventional FS technique, while maintaining the MV search quality; it can also be represented as lossless ME approach. In partial search approach, most of search points are ignored strategically for high speed search. Therefore, the computational cost of partial search methods is normally lower than that of fast exhaustive search approach, but they are with some degradation in search quality.

Fast exhaustive search approach, also known as the fast full search (FFS) [66] technique, can achieve the same block matching performance as FS. Fast exhaustive search methods are attractive enough due to having lower complexity without any degradation in quality. Partial distortion elimination (PDE) [36], [37] is a significant lossless ME algorithm, which was adopted in the reference software encoder of H.264/AVC [38]. In a different way, the H.265/HEVC test model (HM) reference software [67] implements a FFS-like technique; when the fast encoder mode parameter is set to on, only a set of even rows in each block is used for the block distortion computation. The main idea of partial distortion search is explained in [39]. PDE can alleviate the computational cost by omitting unwanted computation during matching error operations when the accumulated partial distortion is larger than the previously measured minimum distortion within the search window. The modified and improved PDE algorithms are introduced in [40], [41], [42], [43], respectively. However, some of the modified PDE techniques [40], [41], [43] cause a little bit quality degradation and bitrate increase, which result in lossy motion estimation, even though they attempt an additional reduction in computational cost.

Successive elimination algorithm (SEA) [44] is also frequently cited as fast exhaustive search approach. SEA is possible to reduce a high number of matching evaluations by applying the sum norm for each block; if the sum norm is inconsistent with a necessary condition derived by Minkowski's inequality [45], the search process for the corresponding block can be skipped. Multilevel SEA (MSEA) [46] can reject many candidates by using the sum norms of subblocks after dividing a block into several subblocks. In [47], the adaptive search order (ASO) is presented as an improvement of MSEA.

Partial search approach attempts a decrease in computation by reducing the number of candidate samples to find the best matched block or performing the early termination or predicting the best motion vector for the current macroblock. Each partial search algorithm is based on various techniques such as unimodal error surface assumption [48], [49], spatio-temporal correlation between the neighboring blocks and frames [50], [51], motion vector probability distribution prediction [57], [58], [60], [62], [64], multi-resolution frame structure [52], [53], [54], and integral projection [55], [56].

In particular, most of the partial search techniques are using general characteristics of real-world video sequences, which primarily include unimodal error surface assumption and motion vector probability distribution. Though the distortion cost increases monotonically as the search point moves away from the optimal one with the global minimum cost, it is not true for all the cases because of the irregular and unsteady motion of objects and background in video sequence. Therefore, the ME search process, such as the three-step search (3SS) [48] using unimodal error surface assumption, could often easily be trapped at a local minimum [57].

Through observational evidences that the motion vector probability distribution for real-world image sequences is center-biased [57], [58], [60], [62], [64], many competitive partial search algorithms have been proposed up to the present, which try to decrease the number of search points with some degradation in block matching accuracy with respect to FS. For example, the new three-step search (N3SS) [57], the four-step search (4SS) [58], the block-based gradient descent search (BBGDS) [59], the diamond search (DS) [60], the hexagon-based search (HEXBS) [61], the cross-diamond search (CDS) [62], the efficient three-step search (E3SS) [63], the cross-diamond-hexagonal search (CDHS) [64], the unsymmetrical-cross multi-hexagon-grid search (UMHexagonS) [65], and the test zone search (TZS) implemented in the reference software such as JSVM [66], JMVC [66], and HM [67] are fast block-based ME algorithms, developed to effectively reduce the computational complexity of the integer-pixel ME (IME) module. Each of the existing IME algorithms has a different search strategy to satisfy both the accuracy of estimation and the search speed.

In this thesis, the proposed fast IME techniques are introduced for further reduction in computational complexity based on unimodal error surface assumption, spatio-temporal correlation, and motion vector probability distribution prediction, which will also archive negligible rate-distortion performance degradation, compared to UMHexagonS [65].

2.4. Overview of Fractional-Pixel Motion Estimation

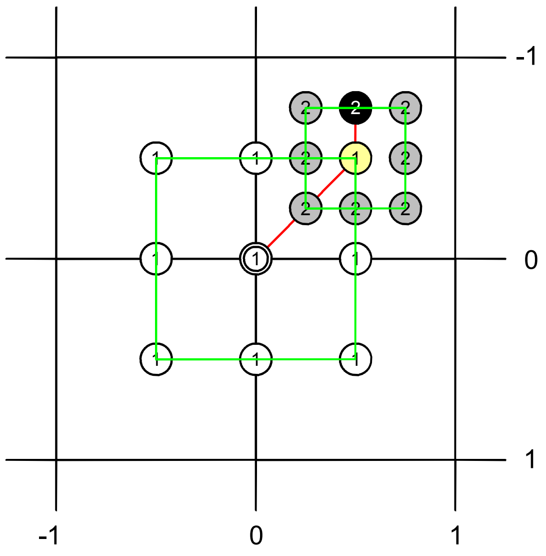


Fig. 2.2. A search path example of the conventional full fractional-pixel search algorithm; the two-step refinement process is hierarchically performed from half-pixel (Step 1) to quarter-pixel (Step 2).

Most motion estimators follow the IME process with a fractional-pixel ME (FME) process. The location of a moving object in a video sequence can be represented at fractional-pixel, as well as integer-pixel precision. FME improves the image quality visibly, but requires higher computational complexity. The runtime of the FME module is over 30% of the total encoding time [18]. The conventional two-step full fractional-pixel search (FFPS) [66], [67], also called the hierarchical fractional-pixel search, is wasteful and inefficient owing to its fixed number of search points, as shown in Fig. 2.2. In addition, an interpolation process (upsampling) must be performed to create the fractional-pixel search area for half-pixel or quarter-pixel search point refinement, which requires a high computational complexity and frequent memory access. To ameliorate these issues, techniques such as the center-biased fractional-pixel search (CBFPS) [65], the fast sub-pixel ME having lower computational complexity [68], the quadratic prediction based fractional-pixel search (QPFPS) [69], the low-complexity algorithm for fractional-pixel motion estimation [70], and the fast ME with interpolation-free sub-sample accuracy [71] have been developed.

CBFPS is a typical interpolation-based fast FME algorithm adopted in the reference encoder software of H.264/AVC. After predicting fractional-pixel MV, The small diamond search pattern (SDSP) is applied to refine the estimated MV. However, if predicted MV is not accurate, the FME search would occasionally be trapped into the local minimum.

Some parabolic prediction based techniques have been proposed in [68], [69], [70], and [71] respectively. In [68], three parabolic models are introduced to approximate the block matching error at half-pixel accuracy. The two dimensional parabolic model in [68] is also used in the low-complexity algorithm for FME [70]. Even though the parabolic models require very low computational complexity for determining the best predicted fractional-pixel MV, they do not provide a detailed solution for quarter-pixel or less-than-one accuracy. In addition, the quality performance of the parabolic prediction based methods requires further improvement for practical applications.

QPFPS is based on the one-dimensional (1-D) mathematical model. In this algorithm, a degenerate quadratic prediction function is used to approximate the matching error value within the fractional-pixel ME search area. The local coordinates of five integer-pixel positions are utilized to compute the model coefficients. To obtain the minimum matching error, the differential operation is performed on the quadratic prediction function. The predicted position is located at the center of SDSP, and then the small diamond search (SDS) algorithm is carried out to refine the best fractional-pixel MV.

The low-complexity algorithm for FME is using a 1-D parabolic decomposition of the two-dimensional (2-D) parabolic model. The existing 2-D parabolic model FME algorithm is not actually suitable in quarter-pixel resolution ME. Thus, the 1-D parabolic decomposition-based method was proposed to fix the shortcoming in quarter-pixel resolution ME. The FME algorithm has lower computational cost and better performance at quarter-pixel MV accuracy, compared to the conventional 2-D parabolic model.

On the other hand, there are many minor improved FME methods. The prediction-based directional fractional-pixel search (PDFPS) [72] combines some techniques such as fractional-pixel MV prediction, early termination, quarter-pixel small diamond search refinement, and directional search. PDFPS produces low computational cost with a slight PSNR loss and bitrate increase. The motion prediction fast fractional-pixel search (MPFPS) [73] tries to reduce computation by using the simplified adaptive search range (SASR) and the mixed small diamond search pattern (MSDSP). The simulation results in [73] show that the computational load is lower than that of PDFPS. The fast and efficient fractional-pixel search (FEFPS) [74] is based on a unimodal error surface assumption. The FME method can further decrease the number of search points by searching only some positions in a particular quadrant, compared to CBFPS.

In this thesis, the proposed fast FME techniques focus on predicting the best fractional-pixel MV without using interpolation operations for reduction in computational complexity. Chapters 4 and 5 present various interpolation-free techniques to overcome some drawbacks to the existing mathematical prediction based FME methods. However, although the proposed techniques can improve further the prediction performance at very low computational cost, the improvement of performance may be not fully satisfactory for practical use. The performance of video encoding techniques can be considered as a tradeoff between rate-distortion and computational cost. In other words, it is also important to adjust the balance between complexity and performance. In Chapter 6, therefore, a novel FME technique is proposed for a reasonable improvement in rate-distortion performance at the expense of low complexity. In the experimental results of each chapter, the performance of the proposed algorithms will be thoroughly evaluated in terms of PSNR and bitrate as well as computational complexity.

Chapter 3
INTEGER-PIXEL MOTION ESTIMATION

3.1. Introduction

Motion estimation (ME) is an effective and representative technique for removing the temporal redundancy in video sequences. Block-based motion estimation has been adopted by most video compression standards including MPEG-2, H.264/AVC, and H.265/HEVC. The block-matching approach is the most popular technique for block-based motion estimation. The full search (FS) [66], [67] algorithm evaluates exhaustively all the reference blocks within a search range in order to determine the best matched block. Even though FS provides outstanding quality performance and simple data flow and circuit control, a prohibitive amount of computation is required if the search range becomes too large. This high computational complexity makes it often not suitable for real-time implementations. Over many years, many fast integer-pixel (IME) algorithms have been proposed to replace FS with a high level of computational complexity. Each fast IME algorithm uses a different search strategy to satisfy both the search quality and search speed. The unsymmetrical-cross multi-hexagon-grid search (UMHexagonS) [65] was proposed as a reference IME algorithm for the reference video encoder of H.264/AVC. UMHexagonS is a highly efficient IME algorithm, but there is the potential for speed improvement. Thus, in this chapter, some fast IME techniques are proposed to further improve the speed performance of UMHexagonS. The proposed techniques have been developed based on close analysis of the statistical data for real-world video sequences as well as the existing techniques used in UMHexagonS. The results of the simulations will demonstrate the performance of the modified fast IME algorithm combined with the proposed techniques.

3.2. Related Work

In this section, a high-performance hybrid integer-pixel ME algorithm, UMHexagonS, is analyzed, and then its modified approach will be proposed for further reduction in computational complexity. UMHexagonS embedded in the H.264/AVC JM (Joint Model) reference software [66], [75] combines many advanced techniques, including early termination, MV prediction, and search patterns of cross, diamond, and hexagon. As shown in Fig. 3.1 (a), a 16 points large hexagon pattern is considered based on the fact that horizontal motion is generally much more frequent than that of vertical motion for real-world video sequences. The two search patterns of Fig. 3.1 (b) and (c) are adopted in the extended hexagon-based search (EHS) [65], which is used as a sub-algorithm in the UMHexagonS algorithm.

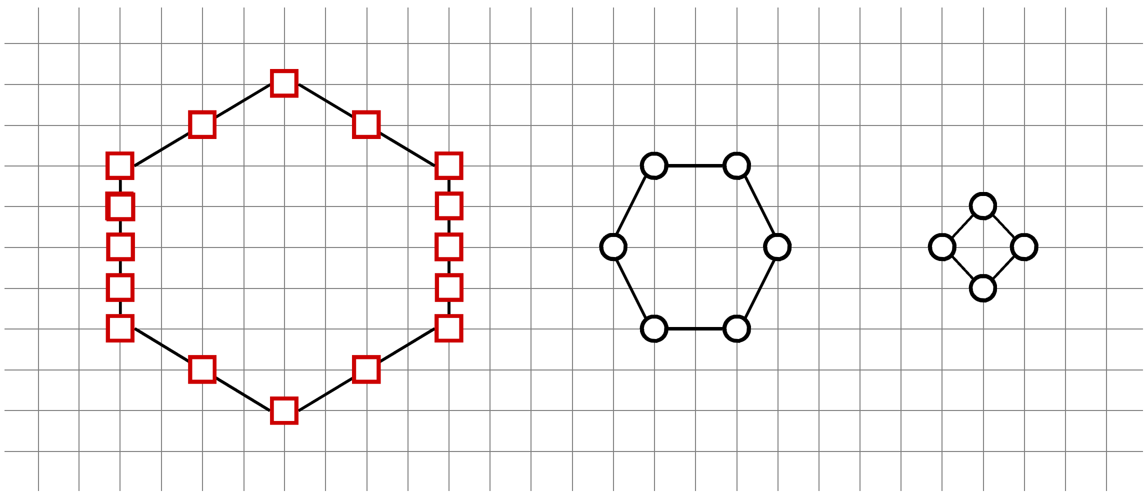


Fig. 3.1. The Search patterns used in UMHexagonS. (a) Large hexagon pattern, (b) Hexagon search pattern, and (c) Small diamond search pattern.

The search process of UMHexagonS is described in Fig. 3.2. The procedure of UMHexagonS is also summarized as follows:

Step_0) Before the IME process, the start search point should be decided at first; then, it performs the early termination (ET) operation [65].

Step_1) First, an unsymmetrical-cross search is made. The spacing between checking points is two and there are twice as many points horizontally than vertically; then, it performs the ET operation.

Step_2) Starting from the best point found in the cross search step, next small rectangular full search is made; then, it performs the ET operation.

Step_3) A multi-hexagon-grid search strategy is taken (from 1 to *SR*/4, *SR*=search range). The 16 points hexagon pattern used in the grid search has more points at the left and right edges; then, it performs the ET operation.

Step_4) In the final search step, EHS (Step_4-1) or small diamond search (Step_4-2) is used to refine the best MV found in the multi-hexagon-grid stage.

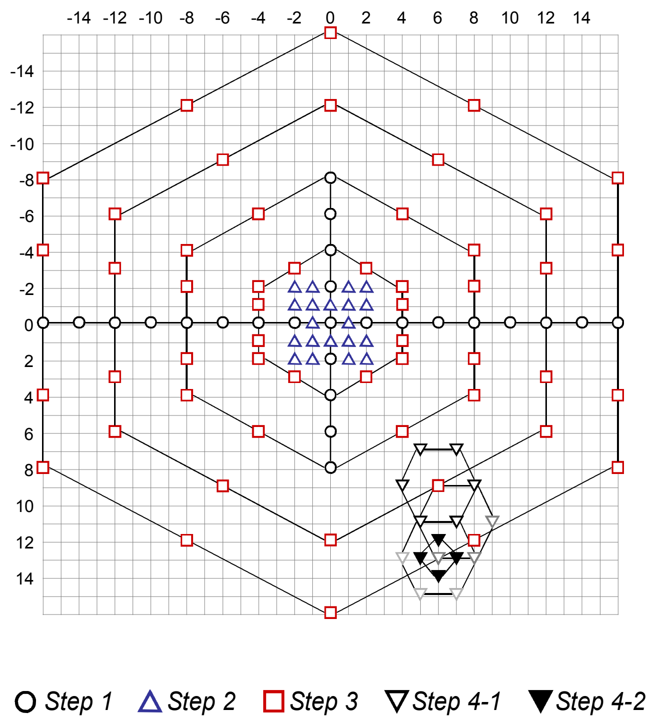
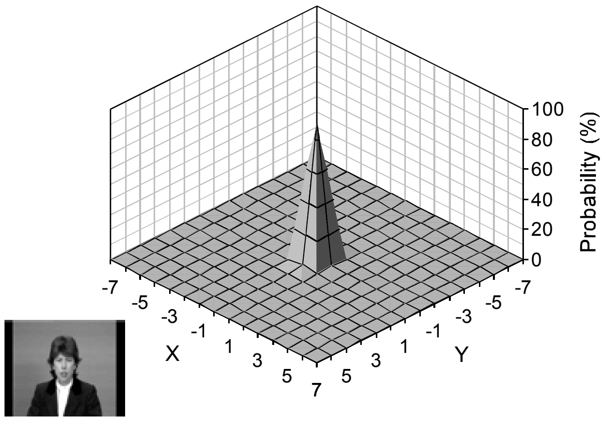
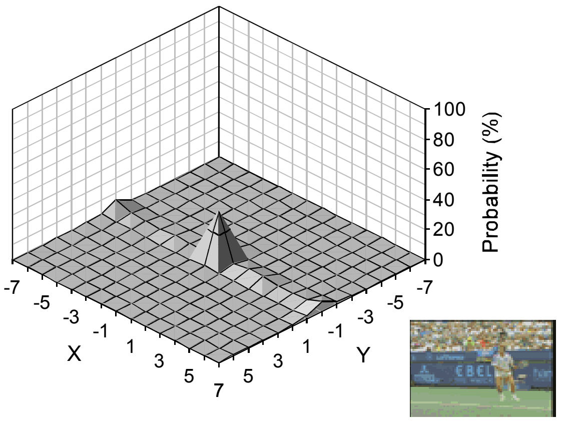


Fig. 3.2. The search process of UMHexagonS.

3.3. Observation: Motion Vector Distribution



(a)



(b)

Fig. 3.3. Motion vector distributions for (a) "Claire" and (b) "Stefan."

The current search pattern in the UMHexagonS process, as explained above, varies with the conditions of each search step. The search patterns used in UMHexagonS can be improved by removing unnecessary search points and reforming into more robust shapes. For this, a statistical analysis on various motions from real-world video sequences is required. As shown in Fig. 3.3 and Fig. 3.4, the motion vector probability (MVP) distributions are resulted from simulations using the full search algorithm with a search range *SR* = ±7. Sequences used for Fig. 3.3 (a) and (b) are CIF Sequence "Claire" (100 frames) and CCIR601 Sequence "Stefan" (200 frames) respectively. Fig. 3.4 is using three CCIR601 Sequences – "Garden," "Stefan," and "Susie" (30 frames). In Fig. 3.3 (a), over 89% of motion vectors are found at the central point (0, 0). Within the central 5×5 area, most of motion vectors are also found. The MVP distribution for the CCIR601 sequence "Stefan" including tough and rough sports scene is shown in Fig. 3.3 (b). Unlike the distribution as illustrated in Fig. 3.3 (a), Fig. 3.3 (b) does not show altogether center-biased MVP distribution but wide distribution.

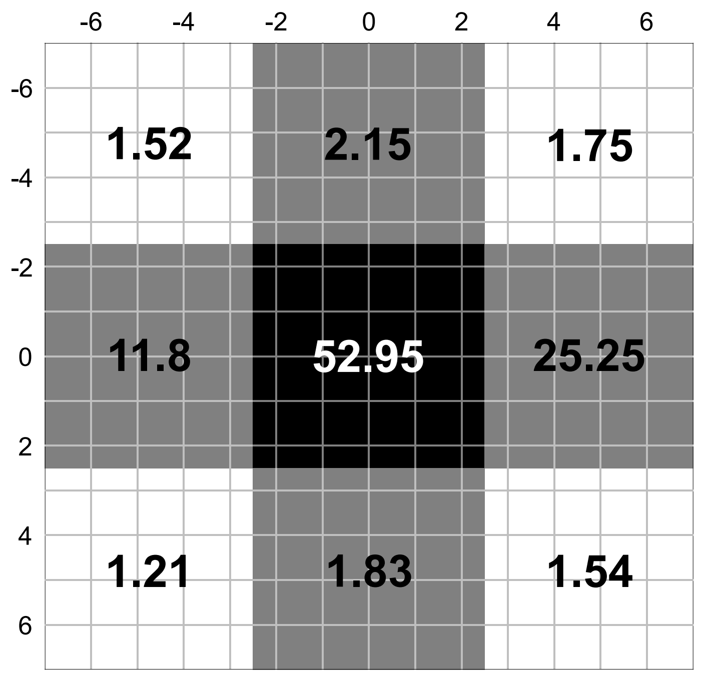
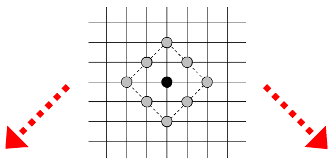


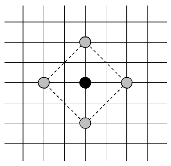
Fig. 3.4. Motion vector probability (%) distribution for three CCIR601s.

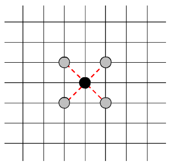
Fig. 3.4 displays the results of MV's accumulated distribution for the three CCIR601 sequences. The MVP distribution diagram can be divided into nine regions to comprehensively analyze the general MVP distribution tendency of video data. In Fig. 3.4, it can be seen that the portion of MV within the 5×5 region in the center of the search range is the highest at 53%. Moreover, the portions of MVP in the horizontal and vertical regions are especially high in the regions other than the central region. The major cause for this phenomenon is closely related to the fact that the camera capturing the images mainly moves in horizontal and vertical directions. Based on this observation, the proposed fast integer-pixel motion estimation techniques are presented in the following section.

3.4. Proposed Techniques

3.4.1. Revised Diamond Search Algorithm


(a) Large diamond-shaped pattern (LDSP) of DS


(b) Redefined LDSP (RLDSP)


(c) Small X-shaped pattern (SXSP)

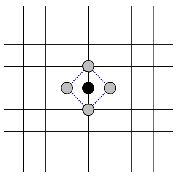

(d) Small diamond-shaped pattern (SDSP)

Fig. 3.5. Redefined search patterns from LDSP used in original DS.

The dense search points in the large diamond-shaped pattern (LDSP) of the original DS algorithm, as shown in Fig. 3.5 (a), are inadequate for searching wide-area movements. On the other hand, in Fig. 3.5 (b), the redefined large diamond-shaped pattern (RLDSP) of the distributed search points is more appropriate for horizontal and vertical searches. Furthermore, the small X-shaped pattern (SXSP) as shown in Fig. 3.5 (c), which has branched out from LDSP, can flexibly cope with local minimum block distortion measure (BDM) points. The small diamond-shaped pattern (SDSP) as shown in Fig. 3.5 (d) is used in the final search step of the proposed revised diamond search (RDS) algorithm. Two Examples of search paths of the proposed RDS are shown in Fig. 3.6. The search process of the RDS algorithm using RLDSP, SXSP, and SDSP can be summarized as follows:

Step_1) The five checking points of RLDSP are tested. If the minimum BDM point is located at the center, go to Step_3; otherwise, go to Step_2.

Step_2) A new RLDSP with the center at the previous minimum distortion point is formed. If the minimum BDM point obtained is located at the center of RLDSP, go to Step_3; otherwise, recursively repeat this step.

Step_3) Switch the search pattern from RLDSP to SXSP. If the minimum BDM point occurs at the center of SXSP, go to Step_4; otherwise, repeat Step_2.

Step_4) One of the five checking points of SDSP is determined as the new minimum BDM point, which is the final MV.

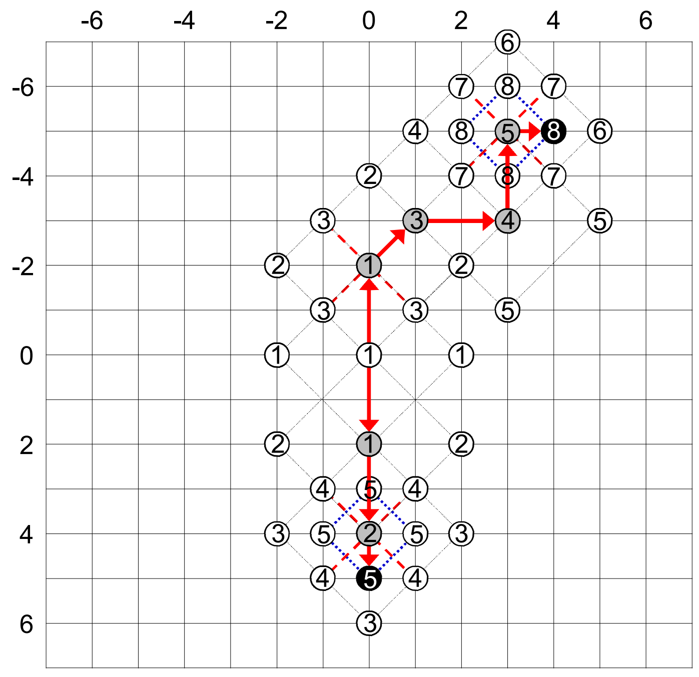


Fig. 3.6. Two examples of search paths of RDS.

3.4.2. Performance Evaluation of RDS

The performance evaluation experiments of the proposed RDS algorithm evaluate the search speed and quality. The search speed can be checked with the average number of search points per block, and the average PSNR (peak signal-to-noise ratio) is used to evaluate the search quality. A total of eight sequences, named "Claire" (CIF 360×288, 100 frames), "Coastguard" (CIF 352×288, 200 frames), "Football" (CIF 360×240, 200 frames), "Salesman" (CIF 360×288, 200 frames), "Football" (CCIR601 720×486, 50 frames), "Garden" (CCIR601 720×486, 90 frames), "Stefan" (CCIR601 720×480, 90 frames), and "Susie" (CCIR601 720×486, 90 frames), are used for the experiment. The sum of absolute differences (SAD) criteria is used for BDM measurement. It is defined as follows:

$$SAD = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} | C_{ij} - R_{ij} | \tag{1}$$

where M and N are the parameters about the macroblock size. C_{ij} and R_{ij} are current and reference area samples respectively. The block size and the search range are set at 16×16 and SR = ±15, respectively. Table 3.1 and Table 3.2 compare the performance of the proposed RDS algorithm with those of BMAs such as DS, HEXBS, and CDS. The results in Table 3.2, where SPT denotes the average number of search points per block, indicate that the proposed RDS always uses less search points than the original DS. However, the proposed RDS provides a higher quality performance than DS in most sequences, as shown in Table 3.1. In particular, the search quality is more remarkable for CCIR601 sequences. In sequences like "Football" and "Stefan" that include intense and abrupt movements, the proposed RDS algorithm outperforms DS in terms of the search speed and quality.

Table 3.1. PSNR (dB) Comparison

BMA	CIF Sequences				Average
	Claire	Coastguard	Football	Salesman	
DS	41.162	29.483	24.241	35.281	32.542
HEXBS	40.837	29.322	24.036	35.238	32.358
CDS	41.147	29.478	24.190	35.377	32.548
RDS	41.139	29.456	24.273	35.271	32.535
BMA	CCIR601 Sequences				Average
	Football	Garden	Stefan	Susie	
DS	24.643	26.252	22.616	34.659	27.043
HEXBS	24.540	26.058	22.999	34.772	27.093
CDS	24.600	26.214	22.558	34.562	26.984
RDS	24.685	26.345	23.227	34.831	27.273

Table 3.2. Computational Complexity (SPT) Comparison

BMA	CIF Sequences				Average
	Claire	Coastguard	Football	Salesman	
DS	13.226	17.692	23.898	13.831	17.162
HEXBS	11.119	13.908	17.462	11.309	13.450
CDS	9.575	18.849	24.453	10.628	15.876
RDS	13.193	16.012	20.700	13.591	15.874
BMA	CCIR601 Sequences				Average
	Football	Garden	Stefan	Susie	
DS	23.051	22.328	20.491	20.463	21.583
HEXBS	16.584	16.561	15.943	15.426	16.128
CDS	23.291	25.549	20.378	21.048	22.566
RDS	19.998	18.973	18.487	18.323	18.945

3.4.3. Diamond Web-Grid Search Algorithm

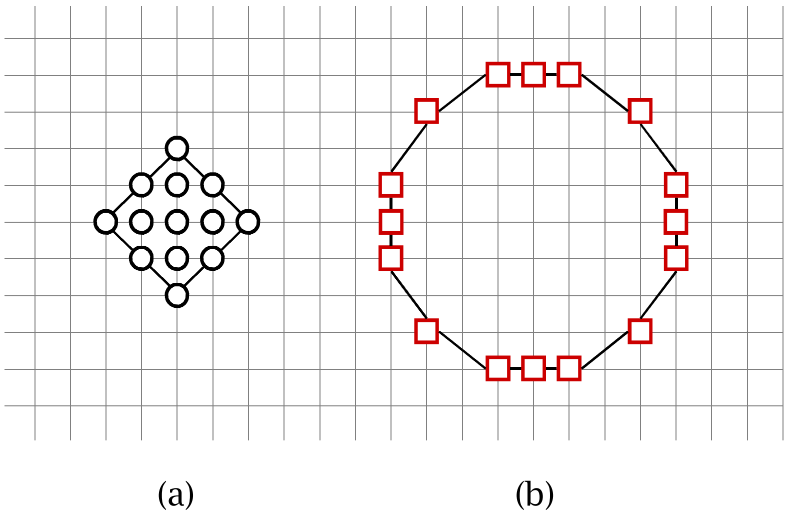


Fig. 3.7. The two new search patterns used in the proposed DWS algorithm: (a) Full diamond search pattern and (b) Dodecagon search pattern.

The RDS algorithm proposed above can be combined with other fast IME algorithms as a sub-algorithm to refine the best estimated MV. In this section, a main-algorithm that can combine with RDS is proposed; RDS will be used as the final MV refinement process of the proposed main-algorithm and try to further improve the performance of it. The proposed main-algorithm, the diamond web-grid search (DWS) algorithm for H.264/AVC motion estimation, was developed based on observations of the MVP distribution and multiple local minima for real-world sequences. The full diamond search pattern and dodecagon search pattern, as shown in Fig. 3.7 (a) and (b), are used as basic search patterns in the proposed DWS algorithm. The full diamond search pattern is very suitable for searching center-biased motions. In contrast to the multi-hexagon-grid used in UMHexagonS, the multi-dodecagon-grid (web-grid) is considering all directions, but is heavier in horizontal and vertical directions than in diagonal directions. In the final search step of the proposed DWS, the proposed RDS can be used to refine the best MV. The search process of the proposed DWS is described in Fig. 3.8. The DWS algorithm can also be summarized in three steps, as follows:

Step_0) Before the integer-pixel motion estimation, the start search point should be decided at first; then, it performs the ET operation.

Step_1) First, a full diamond search and a symmetrical-cross search are made. The spacing between checking points of the symmetrical-cross search pattern is four; then, it performs the ET operation.

Step_2) A web-grid search strategy is taken. The dodecagon pattern is scaled to various sizes (from 1 to *SR*/4); then, it performs the ET operation.

Step_3) In the final search step, an unrestricted center-biased search (Step_3-1: RDS or EHS) or the small diamond search (Step_3-2) is adopted to refine the best MV found in the web-grid stage.

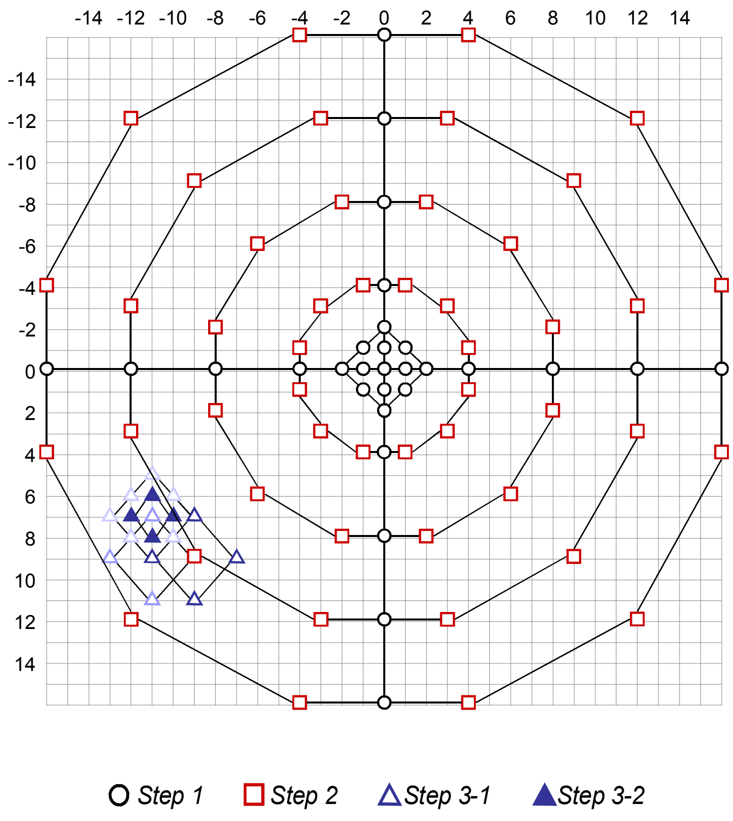


Fig. 3.8. The search process of DWS.

3.4.4. Performance Evaluation of DWS

The experiments are conducted by using four QCIF sequences “Claire,” “Container,” “Grandma,” and “Salesman” and four CIF sequences “Coastguard,” “Mobile,” “News,” and “Stefan.” Three hundred frames are used for each of the eight sequences. We compare the proposed algorithms with fast full search (FFS) and UMHexagonS algorithm using the H.264/AVC JM version 12.4 reference software with search range SR = 16, quantization parameter QP = 40, and rate-distortion optimization RDO = 0, and the baseline profile. The DW+EHS and DW+RDS algorithms, in Step_3-1 of DWS, are using EHS and RDS respectively.

Table 3.3 and Table 3.4 show that the proposed DW+RDS is about 0.011 dB higher compared with that of UMHexagonS in terms of PSNR while the average bitrate increase is 0.024%. From Table 3.5, the motion estimation time reduction is still about 5.3%. On the other hand, DW+EHS can achieve the average bitrate decrease of 0.012 with 0.002 dB PSNR drop on average in comparison with UMHexagonS, whereas the average reduction in ME time is about 5.9%.

Table 3.3. PSNR (dB) Comparison

ME		FFS	UMHexagonS	DW+EHS	DW+RDS
QCIF	Claire	31.477	31.384	31.401	31.451
	Container	27.850	27.841	27.827	27.826
	Grandma	29.683	29.621	29.620	29.654
	Salesman	27.381	27.367	27.357	27.380
CIF	Coastguard	26.675	26.663	26.653	26.647
	Mobile	24.380	24.342	24.344	24.338
	News	29.902	29.864	29.866	29.858
	Stefan	26.372	26.339	26.336	26.354
Δ Average (dB) with respect to UMHexagonS				-0.002	0.011

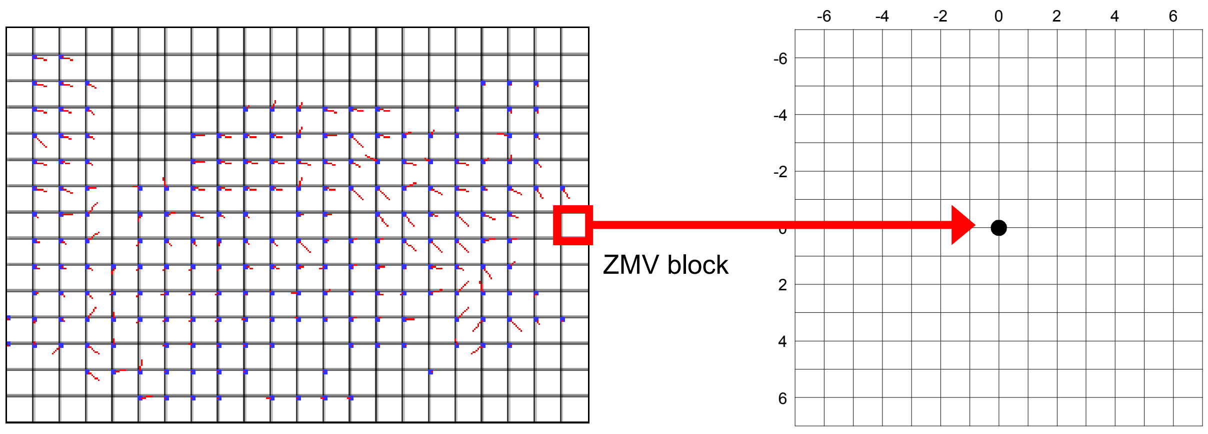
Table 3.4. Bitrate (bps) Comparison

ME		FFS	UMHexagonS	DW+EHS	DW+RDS
QCIF	Claire	7593	7528	7520	7532
	Container	8402	8414	8367	8366
	Grandma	6099	6067	6051	6045
	Salesman	9660	9510	9578	9587
CIF	Coastguard	111963	111654	112114	112002
	Mobile	228985	229682	228922	228806
	News	49051	48988	48814	48884
	Stefan	240217	232526	233430	233794
Δ Average (%) with respect to UMHexagonS				-0.012	0.024

Table 3.5. Motion Estimation Time (ms) Comparison

ME		FFS	UMHexagonS	DW+EHS	DW+RDS
QCIF	Claire	45912	9017	8143	8382
	Container	46505	10792	10371	10321
	Grandma	47674	10085	9489	9471
	Salesman	47576	12073	11237	11593
CIF	Coastguard	196767	55820	52985	52847
	Mobile	189266	60204	57416	56776
	News	186276	39662	37687	38482
	Stefan	189677	57613	54038	53606
Δ Average (%) with respect to UMHexagonS				-5.915	-5.303

3.4.5. Skipping Zero Motion Vector



(a) MV space (b) ZMV = (0, 0)

Fig. 3.9. (a) A ZMV block found in MV space. (b) The location of ZMV in search range = ±7.

A block with a zero motion vector (ZMV) is regarded as a stationary block. ZMVs are usually distributed around the background of a video sequence. In Fig. 3.9, a ZMV is positioned at the center of the search window. Most of BMAs start the search process from the central point of the search window. Attention needs to be paid to the fact that ZMV only uses a single central point, and that the BDM value from the previous frame can be reused. SAD or SATD (sum of absolute transformed differences) are mainly used for measuring block distortion. Since video quality distortion and bitrate are simultaneously considered in H.264/AVC, the rate-distortion optimization (RDO) can also be used, as defined in (2) and (3).

$$RDcost = SAD + \lambda_{mode} \times Rates \tag{2}$$

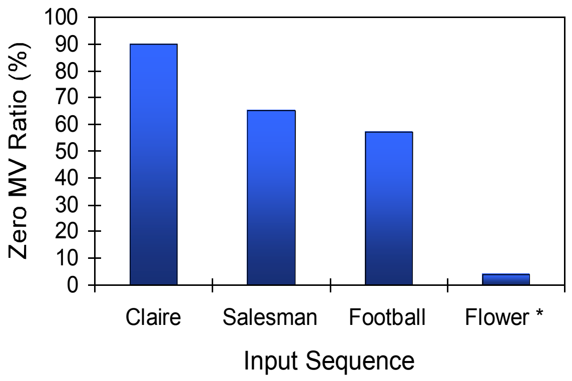
$$\lambda_{mode} = 0.85 \times 2^{(QP-12)/3} \tag{3}$$

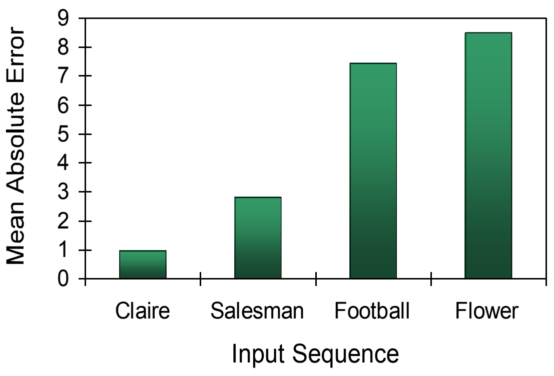
where λ_{mode} is the Lagrangian multiplier and it adjusts the tradeoff between bitrate and video distortion, *QP* is the quantization parameter, and *Rates* represents the number of bits required for coding the difference between the candidate MV and the MV predictor. The sum of squared differences (SSD) can be used instead of SAD for better performance at the cost of an increased computational complexity.

Simulation experiments are conducted using the FS algorithm with the luminance of four CIF sequences (100 frames respectively) to analyze the correlation between ZMV and the mean absolute error (MAE). MAE is calculated as follows:

$$MAE = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} | C_{ij} - R_{ij} | \tag{4}$$

As a result of the simulation, Fig. 3.10 (a) and (b) indicate that the ZMV ratio is inversely proportional to the MAE value. Based on this fact, an existing technique for effectively skipping ZMV [76] is introduced.





“Flower” MV (1, 0) = 42.60%

(a) (b)

Fig. 3.10. (a) ZMV ratio comparison. (b) MAE comparison.

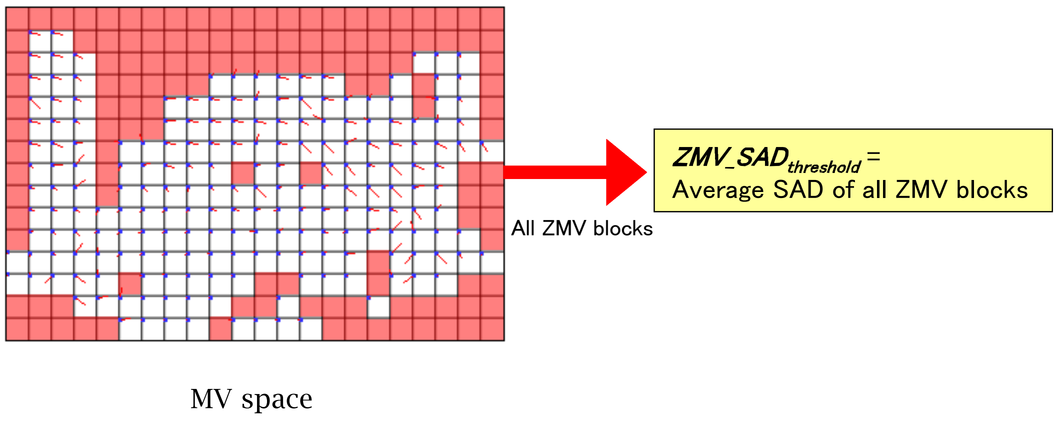


Fig. 3.11. The existing ZMV skip method.

As shown in Fig. 3.11, the existing ZMV skip technique uses the average SAD value of all ZMV blocks as a threshold for the ZMV prediction. The threshold $ZMV\_SAD_{threshold}$ can be calculated as shown in (5).

$$ZMV\_SAD_{threshold} = \frac{1}{N_{ZMV}} \sum \{SAD_{i,j} \mid MV_{i,j} = ZMV\} \tag{5}$$

where N_{ZMV} denotes the total number of ZMV. When the current MV is ZMV, its SAD value is added to a temporary variable, and then the average of the accumulated SAD values, $ZMV\_SAD_{threshold}$, is updated. Table 3.6 expresses the pseudo code for using the threshold in BMA. However, this technique can only be used in the motion estimation unit for MPEG-1, 2 fixed block-size. In other words, it is not appropriate for variable block-size ME (VBSME) of H.264/AVC.

Table 3.6. Pseudo Code for Using ZMV Threshold in BMA

```
$Block\text{-}Matching\_Algorithm(Frame_{reference}, Frame_{current}, i, j) =$
{
    …
    $SAD_{current} \leftarrow SAD(Frame_{reference}, Frame_{current}, Block_{i,j});$
    $if\ SAD_{current} < ZMV\_SAD_{threshold}, return\ ZMV;$
    …
}
```

3.4.6. Efficient Stationary Block Skip Method for VBSME

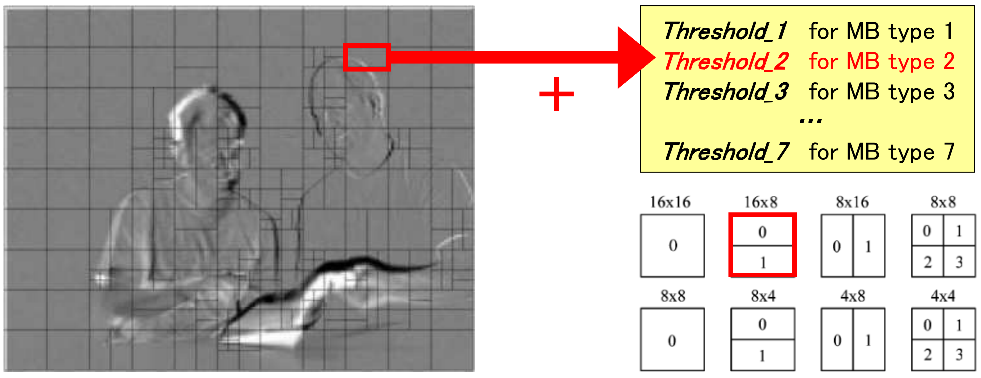

Threshold_1 for MB type 1
Threshold_2 for MB type 2
Threshold_3 for MB type 3
...
Threshold_7 for MB type 7
16x16
16x8
8x16
8x8
8x8
8x4
4x8
4x4
0
1
2
3

Fig. 3.12. The seven variable thresholds used in the proposed ESBSM method and seven segmentations of the macroblock.

In order to implement the ZMV block skip technique in the H.264/AVC JM reference software, a simple solution referred to as the efficient stationary block skip method (ESBSM) is introduced. The proposed solution uses seven different variable threshold variables depending on the type of macroblock (MB) [see Fig. 1.2].

For example, as shown in Fig. 3.12, if a stationary block with ZMV is MB type 2, its minimum SA(T)D (min_mcost) is added to the accumulator variable for MB type 2 which is represented as *ZMVSumMcost[2]* in Fig. 3.13. And then all the seven ZMV thresholds are recalculated right before moving from the current frame to the subsequent frame; the ZMV threshold for MB type 2 is represented as the average value of the ZMV min_mcosts for MB type 2 and the average ZMV min_mcost for MB type 2 can only be used as the ZMV threshold for MB type 2. There are seven variable ZMV thresholds, and each threshold value is determined according to the average ZMV min_mcost of the corresponding MB type. The proposed ESBSM can easily be combined with the existing ME algorithms in H.264/AVC. Fig. 3.13 displays the flow of the ME process combined with the proposed ESBSM implemented in JM. In the experiment results, the proposed DWS algorithm is used as the ME process.

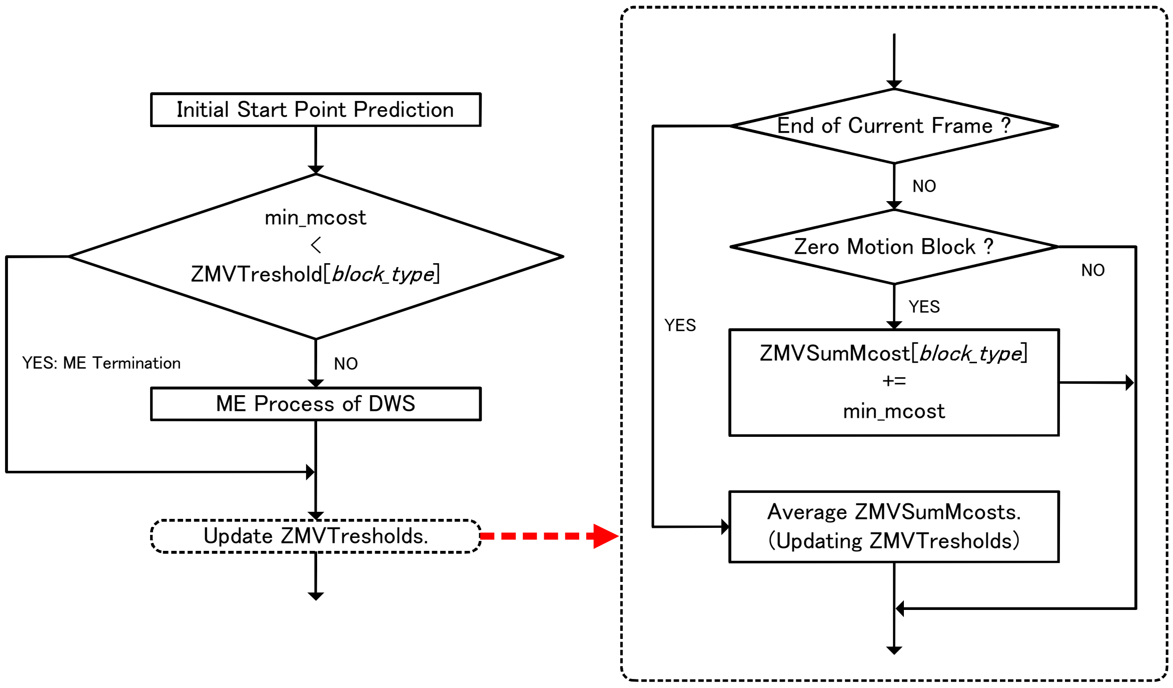

Initial Start Point Prediction
min_mcost < ZMVTreshold[block_type]
YES: ME Termination
NO
ME Process of DWS
Update ZMVTresholds.
End of Current Frame ?
NO
Zero Motion Block ?
YES
NO
YES
ZMVSumMcost[block_type] += min_mcost
Average ZMVSumMcosts. (Updating ZMVTresholds)

Fig. 3.13. The flowchart of the ME process combined with the proposed ESBSM implemented in the H.264/AVC JM reference software encoder.

3.4.7. Experimental Results

The experiments are conducted at JM12.4 with the default settings of *SR* = 16, *QP* = 40, *RDO* = 0, and the baseline profile. Three hundred frames are used for each of the six sequences — QCIF sequences "Claire," "Container," and "Foreman," and CIF sequences "Coastguard," "Mobile," and "Stefan." In addition, 30 frames from three high-resolution HDTV sequences — "Harbour" (720p), "Ship" (2160p), and "Spinningchair" (2160p) are also used in the experiments. The experimental results, as shown in Table 3.7–Table 3.9, indicate that the proposed DW+RDS algorithm combined with ESBSM (EDWS) reduces the computations of UMHexagonS by up to 12%, while maintaining a similar PSNR performance; saving 8.9% ME time and 0.08% bitrate on average.

Table 3.7. PSNR (dB) Comparison

ME		UMHexagonS	EDWS
QCIF	Claire	31.384	31.347
	Container	27.841	27.761
	Foreman	28.137	28.104
CIF	Coastguard	26.663	26.643
	Mobile	24.342	24.332
	Stefan	26.339	26.332
HDTV	Harbour	28.294	28.286
	Ship	34.395	34.372
	Spinningchair	37.166	37.152
Δ Average (dB) with respect to UMHexagonS			-0.026

Table 3.8. Bitrate (bps) Comparison

ME		UMHexagonS	EDWS
QCIF	Claire	7528	7455
	Container	8414	8315
	Foreman	31970	32005
CIF	Coastguard	111654	111346
	Mobile	229682	228682
	Stefan	232526	233541
HDTV	Harbour	1076424	1075376
	Ship	8442496	8557904
	Spinningchair	8048600	8075040
Δ Average (%) with respect to UMHexagonS			-0.080

Table 3.9. Motion Estimation Time (ms) Comparison

ME		UMHexagonS	EDWS
QCIF	Claire	8836	8088
	Container	10744	9626
	Foreman	13035	11744
CIF	Coastguard	56593	49500
	Mobile	61016	54023
	Stefan	57152	51440
HDTV	Harbour	34877	32833
	Ship	299666	280654
	Spinningchair	269397	254437
Δ Average (%) with respect to UMHexagonS			-8.946

3.5. Summary

In this chapter, a fast hybrid motion estimation algorithm was developed to reduce the computational complexity in video encoding. The proposed ME techniques include revised diamond search (RDS) algorithm, full diamond and dodecagon search patterns, and efficient stationary block skip method (ESBSM). The proposed RDS algorithm adopted two new search patterns, the redefined large diamond-shaped pattern (RLDSP) and the small X-shaped pattern (SXSP); RLDSP will be more appropriate to search horizontal and vertical area due to the distributed search points, compared with the large diamond-shaped pattern of the original DS algorithm; SXSP will flexibly cope with local minimum block distortion measure points. As shown in the simulation results, actually, the proposed RDS performs faster and more exact than the original DS for CCIR601 sequences which consist of horizontal, vertical, and vigorous motion contents. The proposed diamond web-grid search (DWS) algorithm consists of the full diamond search pattern and the multi-dodecagon-grid (web-grid); the full diamond search pattern not only is very suitable for searching center-biased motions, but it also contributes to simplifying the search process of UMHexagonS; the multi-dodecagon-grid is considering all directions, but is heavier in horizontal and vertical directions than in diagonal directions; in the final search step of DWS, RDS is used to refine the best MV. The proposed DWS algorithm performs a little faster than UMHexagonS while better search quality is maintained. The proposed ESBSM can omit the ME process and save even more the total encoding time by using the seven variable zero motion vector (ZMV) threshold for variable block-size motion estimation (VBSME) in H.264/AVC. The proposed DWS algorithm combined with ESBSM (EDWS) was simulated in the H.264/AVC JM reference software to substantiate the fact that it can further improve the speed performance of the conventional fast motion estimation algorithm, UMHexagonS.

Chapter 4

VERTICALLY SYMMETRICAL LINEAR MODEL BASED FME

4.1. Introduction

The recent motion estimators for video coding perform an interpolation operation to determine the estimated motion position at fractional-pixel resolution after their integer-pixel ME (IME) process. Even though fractional-pixel ME (FME) has a strong impact on the quality of the reconstructed images, an increase in the total encoding time is necessarily caused because the IME process is followed by the FME process. Since the interpolation filter in the FME process performs a large number of multiplication and addition operations to generate fractional-pixel search points from neighboring integer-pixel samples, it requires a high computational cost as well as frequent memory access. In the H.264/AVC and H.265/HEVC standards, the typical two-step full fractional-pixel search (FFPS) [66], [67] evaluates the eight half-pixel search points followed by the eight quarter-pixel search points. The fixed number of fractional-pixel search points is very wasteful in terms of computational complexity, although it shows a remarkable performance in search quality. The quadratic prediction based fractional-pixel search (QPFPS) [69] tries to decrease the use of interpolation operations by employing a degenerate quadratic prediction model. In this chapter, a vertically symmetrical linear model based FME algorithm including a grouping strategy is proposed to further improve the performance of the degenerate quadratic prediction model. The results of the simulations show the proposed algorithm can achieve a little bit better rate-distortion performance compared with the existing prediction method, while maintaining a similar computational complexity.

4.2. Previous Quadratic Prediction Based FME Algorithm

The quadratic prediction based fractional-pixel search (QPFPS) [69] was proposed to reduce the computational load at quarter-pixel motion vector (MV) resolution. A degenerate quadratic prediction function, which is a simplified version of the two-dimensional parabolic prediction model derived for the matching error approximation, affects the determination of the best prediction position at quarter-pixel MV resolution, which is shown below.

$$P(x,y) = ax^2 + bx + cy^2 + dy + e \tag{1}$$

where x and y indicate fractional-pixel position, a, b, c, d, and e are the five parameters for approximating the matching error cost $P(x, y)$. When the five integer-pixel positions are defined as C = (0,0), H_1 = (-1,0), H_2 = (1,0), V_1 = (0,-1), and V_2 = (0,1) [see Fig. 4.7], the five parameters a, b, c, d, and e can be obtained by solving the simultaneous equation shown below.

$$\begin{cases} P(C) = e \\ P(H_1) = a - b + e \\ P(H_2) = a + b + e \\ P(V_1) = c - d + e \\ P(V_2) = c + d + e \end{cases} \tag{2}$$

As described in (3), the differential operation can be executed on the quadratic prediction function with respect to x and y to assume the minimum error cost $P(x, y)$, which is obtained by being substituted with x_p and y_p.

$$\begin{cases} 2ax_p + b = 0 \\ 2cy_p + d = 0 \end{cases} \tag{3}$$

In the final step, the predicted position is located at the center of the small diamond search pattern (SDSP), and then a modified small diamond search (SDS) algorithm will be carried out to refine the best fractional-pixel search point. In this step case, however, the interpolation process is definitely needed to generate a fractional-pixel search area.

4.3. Observation: Real-World Error Surfaces

The real-world error surfaces of IME and FME are illustrated in Fig. 4.2 and Fig. 4.3. The error surfaces were simulated for the three input sequences: CIF “Flower” and “Football,” CCIR601 “Garden” [see Fig. 4.1] with the search range of ±16. 1/16-pixel MV resolution is used for FME. As shown in Fig. 4.2, the error surfaces of IME are irregular, but those of FME in Fig. 4.3 are undoubtedly unimodal. The fractional-pixel search points within the FME search area are produced by performing the interpolation operations from information about the integer-pixels. Accordingly, the fractional-pixel error cost increases monotonically as the search point moves away from the best point with the minimum error cost. The side of the FME error surfaces is also shaped like parabola, as shown in Fig. 4.3.

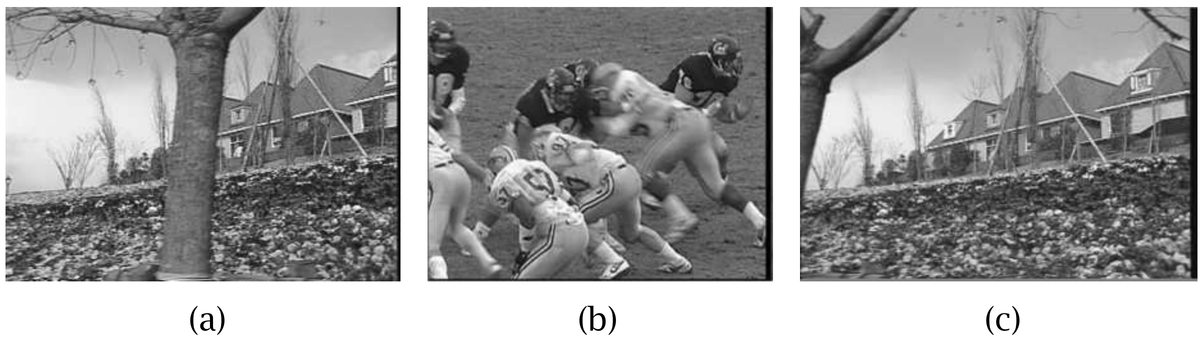
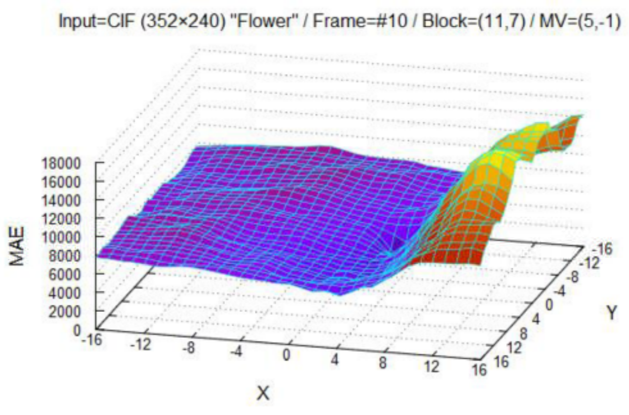
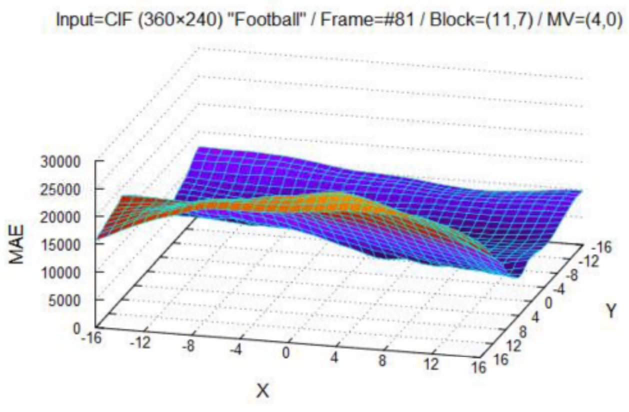


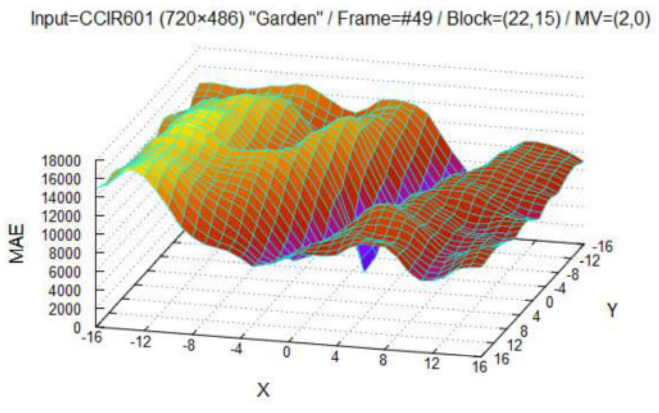
Fig. 4.1. The three test sequences: (a) “Flower,” (b) “Football,” and (c) “Garden.”



(a) “Flower”

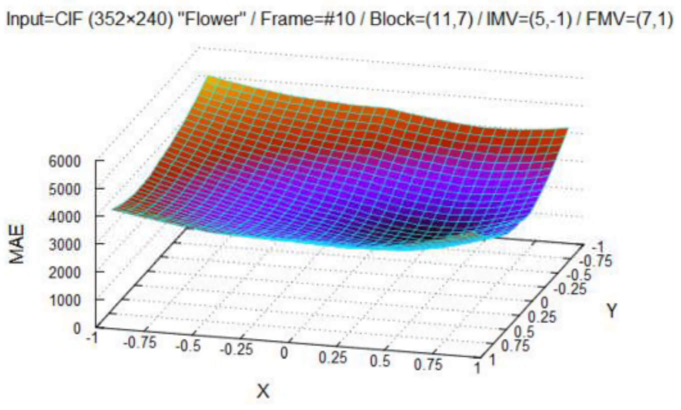


(b) “Football”

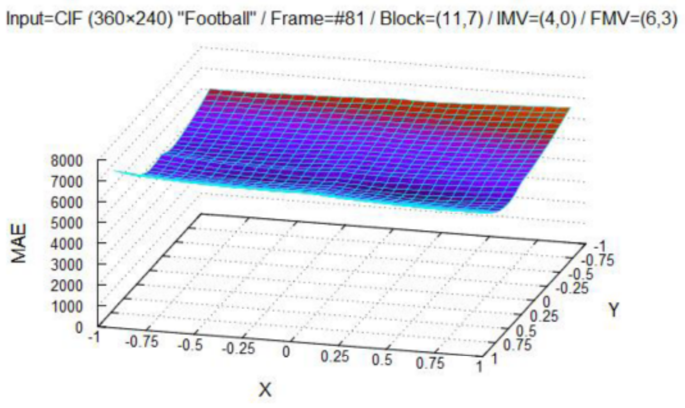


(c) “Garden”

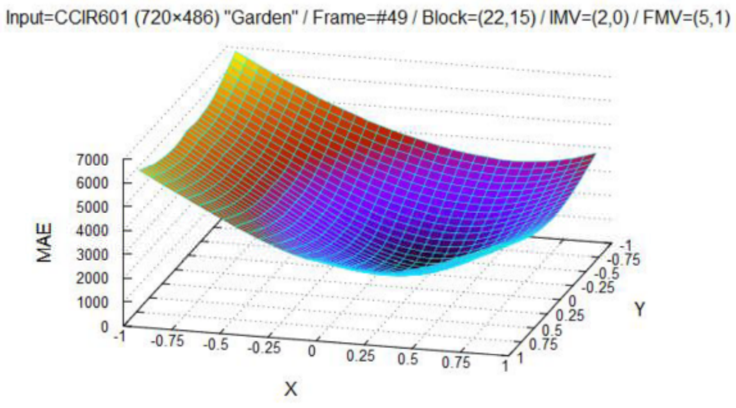
Fig. 4.2. Error surfaces of IME at integer-pixel MV resolution.



(a) "Flower"

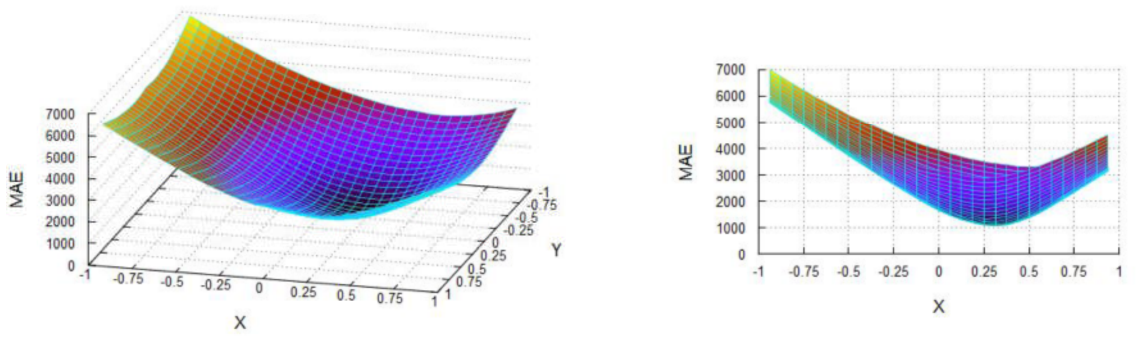


(b) "Football"



(c) "Garden"

Fig. 4.3. Error surfaces of FME at 1/16-pixel MV resolution.



(a) (b)

Fig. 4.4. (a) An example of FME error surface. (b) Another angle of (a).

4.4. Observation: Simplified Error Surface of FME

As already mentioned above, the error surface of FME has been simulated and analyzed directly. Fig. 4.4 (a) shows the FME error surface for CCIR601 sequence “Garden” at 1/16-pixel MV resolution. Unlike that of IME, the FME error surface is clearly unimodal because the sub-pixels were generated by the bilinear interpolation using the existing integer-pixels. Actually, most of FME algorithms are influenced by a kind of interpolation. Therefore, the approach to the FME part should be different from IME. Particularly, we take notice of Fig. 4.4 (b). The shape looks like parabolic definitely. It means we can apply parabolic models including some quadratic functions for FME. The graph also shows that it has vertical symmetry with respect to *x*=5/16.

Furthermore, the vertically symmetrical characteristic of the FME error surface has been observed closely, as illustrated in Fig. 4.5. 1000 FME error surfaces for CCIR601 sequence “Stefan” are gathered sequentially by performing full fractional-pixel search at 1/16-pixel MV resolution and their best MV is (4, 2). The shape average for 10, 100, and 1000 FME error surfaces is computed, respectively, and then only the three minimum MAE values corresponding to the *x*-coordinates -15/16, 1/4, and 15/16 for each error surface are extracted and used to analyze the general characteristics. As a result, Fig. 4.5 shows that the average FME error surfaces simplified represent almost obviously vertical symmetry with respect to x=1/4.

From understanding Fig. 4.4 and Fig. 4.5, a linear model graph can be drawn, as shown in Fig. 4.6. Line $F(x)$ adjoins point (-1, F(-1)) and (0, F(0)). Here, we can introduce symmetry assumption for $F(x)$. Line $G(x)$, which has the negative slope value of $F(x)$, is passing point (1, G(1)). That is, the basic principle of the proposed FME algorithm is to find the location where the two lines intersect.

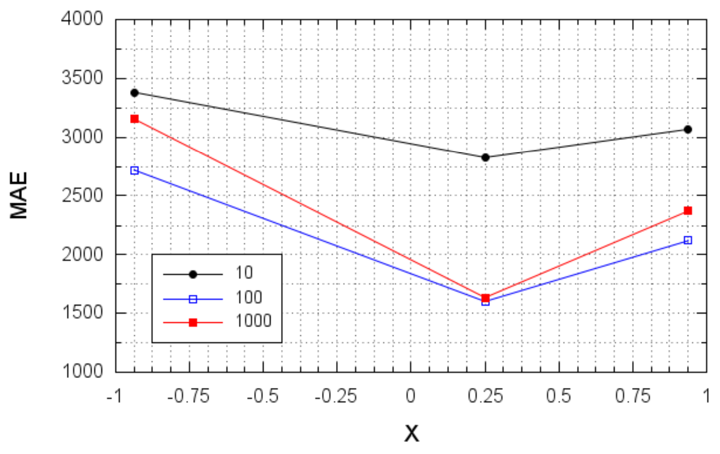


Fig. 4.5. Average FME error surfaces simplified for 10, 100, and 1000 error surfaces at 1/16-pixel MV resolution.

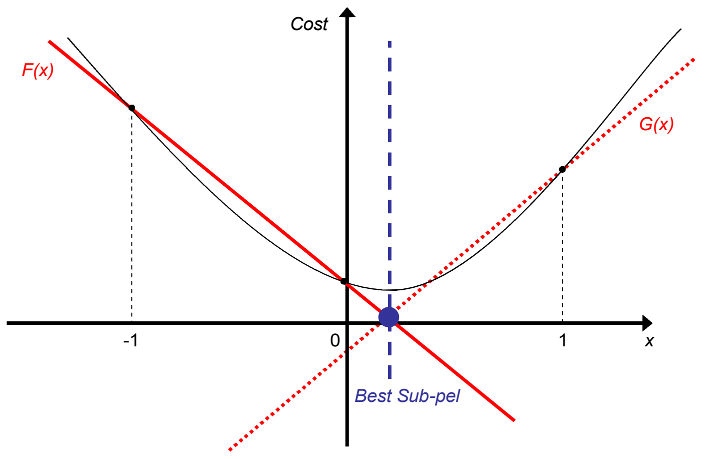


Fig. 4.6. The basic concept of the proposed FME algorithm.

4.5. Proposed Linear Model Based FME Algorithm

The proposed linear model based fractional-pixel motion estimation algorithm reuses the matching error cost of the nine integer-pixel search points as shown in Fig. 4.7. Basically, the linear prediction function applied in the proposed algorithm is described as follows.

$$\begin{cases} F(x) = (-H_1 + C)x + C \\ G(x) = (-H_1 + C)(1 - x) + H_2 \end{cases} \tag{4}$$

In function $F(x)$, for example, $-H_1+C$ is the slope value of the linear equation for a group (H_1, C, H_2). Function $G(x)$ represents a symmetrical linear function with the negative slope value corresponding to $F(x)$. The intersection point x between $F(x)$ and $G(x)$ can be shown below.

$$x = \frac{(H_2 - H_1)}{2(C - H_1)} \tag{5}$$

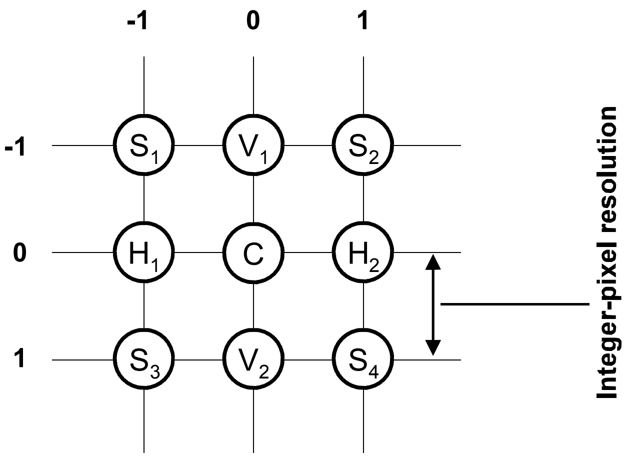


Fig. 4.7. The nine integer-pixel search points reused in the proposed linear model based FME algorithm.

Table 4.1. Horizontal, Vertical, and both Groups Using the Nine Integer-Pixel Search Points

Group	Member	Coordinate
Horizontal	(S_1,V_1,S_2), (S_1,V_1,H_2), (H_1,V_1,S_2), (S_1,C,S_2), (H_1,C,H_2), (H_1,C,S_2), (H_1,C,S_4), (S_1,C,H_2), (S_3,C,H_2), (H_1,V_1,H_2), (H_1,V_2,H_2), (S_3,V_2,S_4), (S_3,V_2,H_2), (H_1,V_2,S_4), (S_3,C,S_4)	x
Vertical	(V_1,C,V_2), (V_1,C,S_3), (V_1,C,S_4), (S_1,H_1,S_3), (S_1,H_1,V_2), (S_2,H_2,S_4), (S_2,H_2,V_2), (V_1,H_1,S_3), (V_1,H_1,V_2), (V_1,H_2,S_4), (V_1,H_2,V_2), (S_1,C,V_2), (S_1,C,S_3), (S_2,C,V_2), (S_2,C,S_4)	y
Horizontal and vertical	(S_1,C,S_4), (S_2,C,S_3)	x or y

In addition to the linear model based prediction, we propose a grouping strategy to enhance the accuracy. The nine integer-pixel search points are grouped according to the close proximity. As listed in Table 4.1, they are divided into three groups, the horizontal, the vertical, and the both horizontal and vertical, of three points. For instance, (S_1, V_1, S_2), (V_1, C, V_2), and (S_1, C, S_4) are the respective members of the above mentioned groups. Each group is used to determine the best x, y, x or y position, respectively. All the groups are calculated to predict the matching error cost of FME, and then the x- and *y*-coordinate of the location which produces the minimum prediction cost are regarded and selected as the best fractional-pixel MV. Before transmitting the best fractional-pixel MV, the quantization step suitable for quarter- or higher-pixel MV resolution is required. We use the quantization process explained in [69] at quarter-pixel MV resolution. As shown in Table 4.2, the best fractional-pixel MV can also be quantized at 1/8-pixel MV resolution.

Table 4.2. Quantization Operations for FME at 1/8-Pixel MV Resolution

Integer-converted x or y	Quantized x or y	Predicted $p = x$ or y
-7	-0.875	$p < -0.8125$
-6	-0.750	$-0.8125 \leq p < -0.6875$
-5	-0.625	$-0.6875 \leq p < -0.5625$
-4	-0.500	$-0.5625 \leq p < -0.4375$
-3	-0.375	$-0.4375 \leq p < -0.3125$
-2	-0.250	$-0.3125 \leq p < -0.1875$
-1	-0.125	$-0.1875 \leq p < -0.0625$
0	0.000	$-0.0625 \leq p \leq 0.0625$
1	0.125	$0.0625 < p \leq 0.1875$
2	0.250	$0.1875 < p \leq 0.3125$
3	0.375	$0.3125 < p \leq 0.4375$
4	0.500	$0.4375 < p \leq 0.5625$
5	0.625	$0.5625 < p \leq 0.6875$
6	0.750	$0.6875 < p \leq 0.8125$
7	0.875	$p > 0.8125$

4.6. Experimental Results

The proposed algorithm has been evaluated based on both the H.264/AVC JM version 12.4 reference software [66], [75] and the H.264/AVC Key Technical Area (KTA) version 2.7 reference software [77]. In JM12.4, the simulation is conducted with the default settings of search range (SR)=16, quantization parameter (QP)=28, rate-distortion optimization (RDO)=off, entropy coding= CAVLC, and baseline profile. In KTA2.7, which includes advanced coding efficiency tools such as adaptive interpolation filters, ME with 1/8-pixel MV resolution, MV competition, adaptive quantization matrix selection, and so on, the simulation is conducted with the default settings of SR=16, QP=40, RDO=on, entropy coding=CABAC, and main profile. After carrying out UMHexagonS for IME, the FME module is applied. Quarter- and 1/8-pixel ME are performed on JM12.4 and KTA2.7, respectively. The four sequences – QCIF sequence “Claire” and CIF sequences “Football,” “Mobile,” and “News” are used for quarter-pixel ME. The other four sequences – QCIF sequence “Salesman” and CIF sequences “Husky” and “Stefan,” and HDTV720p sequence “City” are used for 1/8-pixel ME. They include a variety of motion contents and activities, respectively. 100 frames are encoded for each sequence. The proposed method is directly compared with QPFPS. To assess only the prediction of them without the interpolation process, the modified small diamond search process (MSDSP) used in QPFPS will be skipped.

From Table 4.3 and Table 4.5, the PSNR performance of the proposed algorithm shows a little better improvement compared with the quadratic prediction based method. As shown in Table 4.4 and Table 4.6, the bitrate is also lower than the quadratics’. Particularly, as described in Fig. 4.8, the rate-distortion curve for CIF sequence “Football” shows that the proposed algorithm has a little bit better performance than the existing prediction method. Meanwhile, the computational complexity of the proposed algorithm is similar to that of QPFPS not using MSDSP, as shown in Fig. 4.9.

Table 4.3. PSNR (dB) Performance Comparison at 1/4-Pixel MV Resolution

FME	Sequence			
	Claire (QCIF)	Football (CIF)	Mobile (CIF)	News (CIF)
QPFPS	39.631	36.171	33.792	37.981
Proposed	39.649	36.175	33.794	37.982

Table 4.4. Bitrate (bps) Performance Comparison at 1/4-Pixel MV Resolution

FME	Sequence			
	Claire (QCIF)	Football (CIF)	Mobile (CIF)	News (CIF)
QPFPS	34802	1599065	2062130	232426
Proposed	34627	1579102	2059097	231466

Table 4.5. PSNR (dB) Performance Comparison at 1/8-Pixel MV Resolution

FME	Sequence			
	Salesman (QCIF)	Husky (CIF)	Stefan (CIF)	City (HDTV)
QPFPS	27.455	23.033	26.730	28.244
Proposed	27.487	23.038	26.748	28.247

Table 4.6. Bitrate (bps) Performance Comparison at 1/8-Pixel MV resolution

FME	Sequence			
	Salesman (QCIF)	Husky (CIF)	Stefan (CIF)	City (HDTV)
QPFPS	10915	855072	181999	274032
Proposed	10927	854767	181714	273811

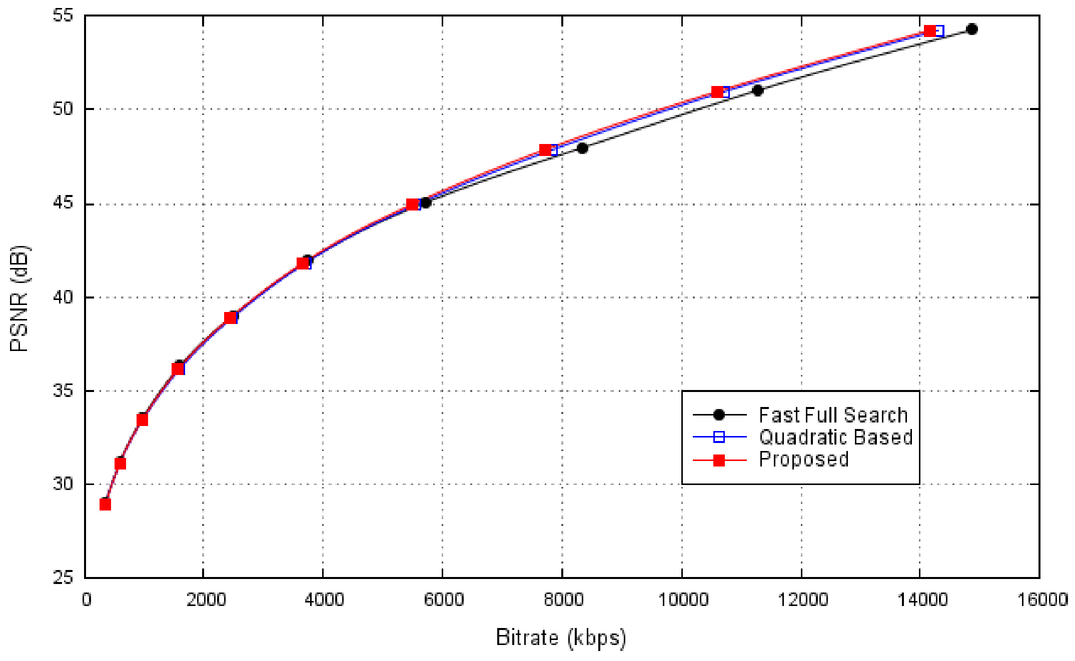


Fig. 4.8. Rate-distortion curves for CIF sequence “Football.”

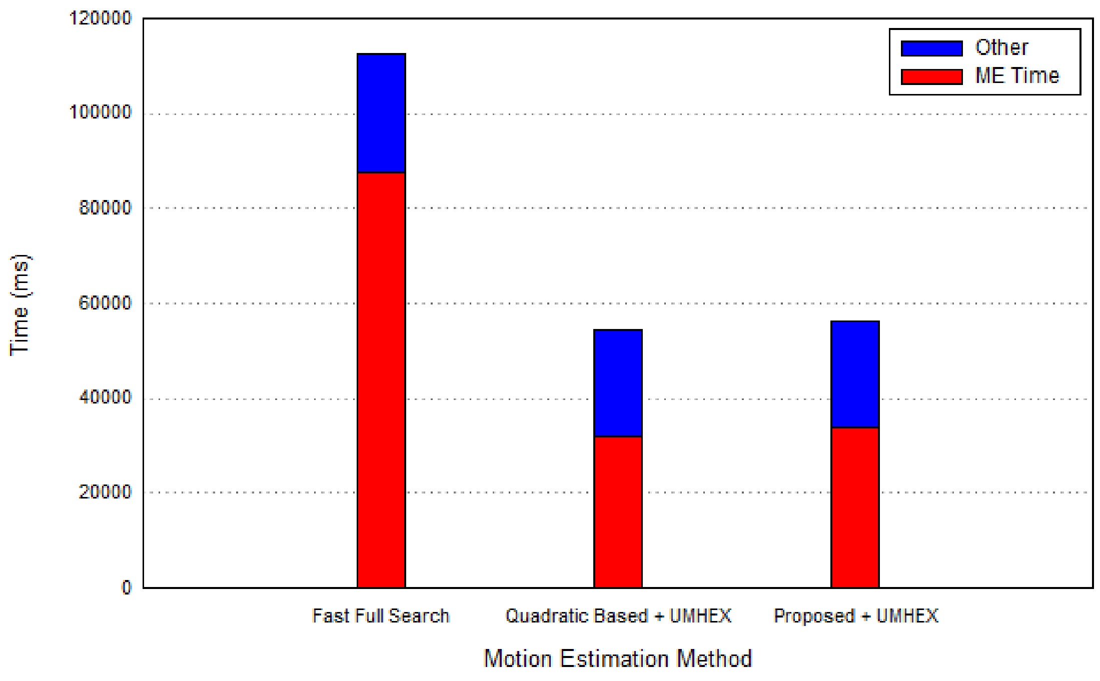


Fig. 4.9. Computational complexity for CIF sequence “Football.”

4.7. Summary

Observing the vertically symmetrical characteristic of the Fractional-pixel ME error surface, symmetry assumption for a simple linear function was introduced. The proposed vertically symmetrical linear model based fractional-pixel motion estimation algorithm can produce a similar performance compared with the existing quadratic prediction based algorithm, but it is simpler in terms of computation. Particularly, the proposed linear model based FME algorithm was designed to implement in the H.264/AVC encoder without using the interpolation process. In addition to the linear model based prediction, a grouping strategy was proposed to further improve its prediction accuracy. In the grouping strategy technique, the nine integer-pixel matching errors are grouped according to the close proximity. The proposed FME algorithm expanded by the grouping strategy technique will have a little bit more increased computational cost because it requires more matching errors of integer-pixel search points. The results of the experiment show that the proposed algorithm can provide a little bit better performance in terms of PSNR compared with a quadratic prediction based algorithm, whereas the average bitrate is more reduced at a negligible increase in computational complexity.

Chapter 5
DATA TREND APPROXIMATION BASED INTERPOLATION-FREE FME

5.1. Introduction

Motion estimation can efficiently eliminate the temporal redundancy to achieve video compression. The computational complexity of a fractional pixel motion estimation (FME) module cannot be negligible, although such modules improve visual quality after the integer-pixel motion estimation process. Most conventional FME methods include an interpolation procedure to form fractional-pixel search points from information about the integer-pixels. The interpolation, however, requires frequent memory access and a certain amount of processing time. The center biased fractional-pixel search (CBFPS) [65] adopted in the H.264/AVC JM reference software [66], [75] is a significant attempt to reduce fractional-pixel search points by using prediction technique and simple search pattern. Various modified versions of CBFPS have also been proposed, but they are still relying on the interpolation process. Thus, some mathematical prediction models [68], [69], [70], [71], have been introduced to approximate the fractional-pixel search area. In this chapter, interpolation-free FME techniques using a data trend approximation and reusing a few integer-pixel matching errors are proposed. The proposed methods were implemented on the reference encoders of H.265/HEVC and H.264/AVC. The experimental results show that the proposed methods produce a similar or better performance than the existing FME methods without the need for any additional search points.

5.2. Parabolic Models to Approximate Matching Errors

Block-based ME evaluates the matching error cost obtained by subtracting the candidate region from the current macroblock in order to find the best matched block within a search range in the reference frame. Equations (1) [68], [71], (2) [68], and (3) [68], [69], have been used to model the matching error *F*(*x*,*y*) at fractional-pixel resolution.

$$F(x,y) = c_1x^2y^2 + c_2x^2y + c_3xy^2 + c_4xy + c_5x^2 + c_6x + c_7y^2 + c_8y + c_9 \tag{1}$$

$$F(x,y) = c_1x^2 + c_2xy + c_3y^2 + c_4x + c_5y + c_6 \tag{2}$$

$$F(x,y) = c_1x^2 + c_2x + c_3y^2 + c_4y + c_5 \tag{3}$$

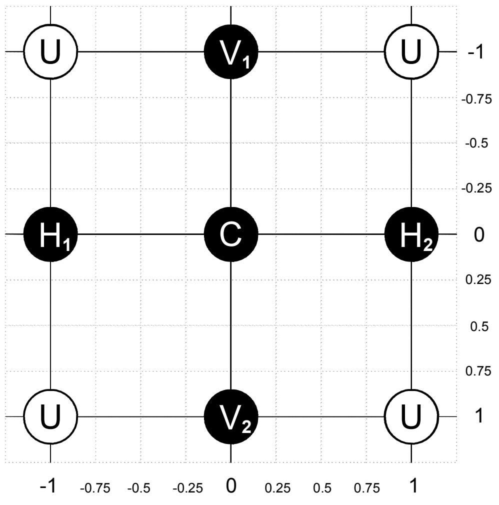
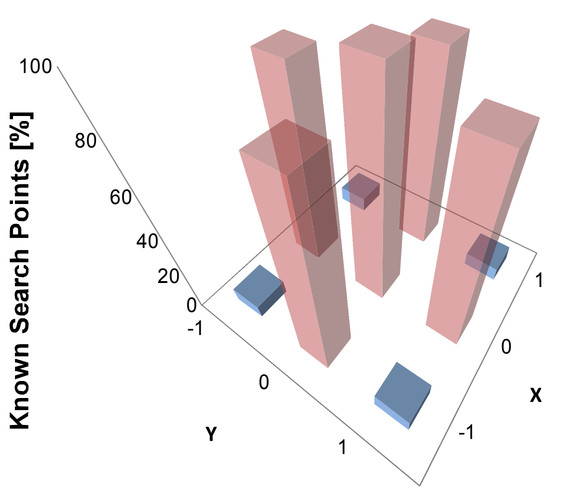


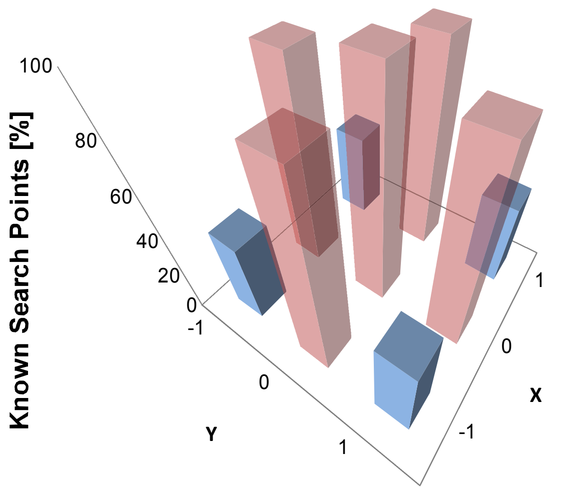
Fig. 5.1. The five main integer-pixel search points (H_1, H_2, C, V_1, V_2) and the four relatively unimportant search points (*U*). The five main search points form the SDSP.

In particular, the parabolic models in (1) and (2) require the matching errors of the nine adjacent search points at integer-pixel resolution, as described in Fig. 5.1, to determine coefficients c_1–c_9 and c_1–c_6, respectively. In other words, if all nine matching error costs are not provided by the IME process, the estimation cannot be guaranteed. To fix this problem, a full search (FS) and an eight neighbor search (ENS) have been used for IME [68], [71]. However, FS is very wasteful in terms of computational complexity, and the rectangular search pattern consisting of eight IME search points used in ENS is inefficient compared with the small diamond search pattern (SDSP) with five IME search points, illustrated in Fig. 5.1. SDSP has been applied to many fast IME algorithms due to its efficiency and simplicity [60], [61], [62], [63], [64], [65], [66], [67]. In addition, many powerful IME algorithms, including UMHexagonS in H.264/AVC, terminate the search process using SDSP in the final step. UMHexagonS occasionally determines the best position by checking only one search point using the early termination technique [65]. The fast IME algorithms using SDSP, therefore, may not calculate the matching errors of the diagonal search points surrounding the best determined position. Fig. 5.2 shows the results of a simulation counting the number of known IME search points with their matching error. In Fig. 5.2, the percentage $K_{x,y}$ of each local position (x,y) can be obtained as follows:

$$K_{x,y}(\%) = \frac{\text{The number of the known search points}}{\text{The total number of the macroblocks used for IME}} \times 100 \qquad (4)$$



(a)



(b)

Fig. 5.2. Results of a simulation counting the number of known integer-pixel search points with their matching error cost by performing UMHexagonS. (a) Percentages of known search points within a 3 × 3 range of the local position for the QCIF "Salesman" sequence. (b) Percentages for the CIF "Football" sequence.

In the simulation, UMHexagonS is used for IME with the QCIF test video sequence "Salesman" (100 frames) and the CIF sequence "Football" (100 frames). These videos include small and large motion objects, respectively. As described in Fig. 5.2, the percentage $K_{0,0}$ of the known integer-pixel search points of the center position (0,0), which represents the local coordinates of the search point corresponding to the best position with the lowest matching error, is always 100%, and the percentages ($K_{-1,0}$, $K_{1,0}$, $K_{0,-1}$, $K_{0,1}$) of the four positions forming the SDSP are also over 90%. In contrast, the four diagonal positions on the edge have low percentages ($K_{-1,-1}$, $K_{1,-1}$, $K_{-1,1}$, $K_{1,1}$) of about 7% and 40% for the "Salesman" and "Football" sequences, respectively. This means that the two models in (1) and (2) have difficulty working with powerful IME algorithms using SDSP. Accordingly, if a fast IME using SDSP was to be followed by the FME process based on (1) or (2), it would cause an increase in the total encoding time and require some modifications to the fast IME module. The models in (1) and (2) are also unstable on the extension to the quarter-pixel or less-than-one FME. This is because some matching errors at the outside half-pixel locations, e.g., (-0.5,-1), must be additionally approximated after calculating the coefficients.

Contrary to the mathematical models discussed above, the parabolic model in (3) can be applied without any difficulties under state-of-the-art IME techniques because it needs only five IME matching errors, as described in Fig. 5.1, to determine the five coefficients c_1–c_5. This model can also be decomposed into two one-dimensional (1-D) parabolic models that approximate the horizontal and vertical matching errors separately, as described in the following equation:

$$F(p) = c_1 p^2 + c_2 p + c_3, \quad (p = x \; or \; y) \tag{5}$$

As has been discussed [69], the minimum matching error cost *F*(*p*) can easily be found by differentiation with respect to *x* and *y*. When *dF*/*dp* = 0, the *x* and *y* coordinates are regarded as the best prediction position (x_b,y_b).

$$F'(p) = 2c_1 p + c_2 = 0, \quad p_b = \frac{-c_2}{2c_1} \tag{6}$$

The 1-D parabolic model uses only the three IME matching error costs, corresponding to (H_1, C, H_2) or (V_1, C, V_2), to compute c_1–c_3, as derived in (7) [69].

$$\begin{aligned} c_1 &= (I_1 + I_2 - 2C)/2, \quad (I_1 = H_1 \; or \; V_1, I_2 = H_2 \; or \; V_2) \\ c_2 &= (-I_1 + I_2)/2 \\ c_3 &= C \end{aligned} \tag{7}$$

The best predicted x_b and y_b coordinates are estimated independently of each other. Here, however, the 1-D parabolic model based prediction has a serious fault, as shown below:

$$C > I_2, \quad if\ p_b = \frac{-c_2}{2c_1} > \frac{1}{2} \tag{8}$$

Equation (8) is an abnormal case, and there is a contradiction because the matching error C at the local location (0, 0) always returns the lowest error cost in the IME. That is, the 1-D parabolic model based prediction alone is not able to find the best prediction position at locations with pixel values greater than 0.5 or less than -0.5. This will have a serious impact on the FME process at quarter-pixel or less-than-one resolution. Moreover, if the matching error H_1 or H_2 is set to zero, denoting an unknown matching error, the 1-D parabolic graph tends to be concave down rather than concave up. As an alternative solution, to enhance the reconstruction PSNR performance, QPFPS [69] adopted an interpolation based refinement procedure in its final search step, although this led to an increase in computational complexity.

5.3. Surface Modeling to Approximate Data Trends

The parabolic models discussed in the previous section can be extended to higher-order polynomial surface models to achieve more accurate prediction. However, higher-order polynomial functions require more computational complexity and IME matching error costs, and often result in unwanted undulations. Thus, different forms of error surface modeling from the above-mentioned parabolic models have been considered. Free-form surface modeling is used to describe the skin of a 3-D geometric element. The surfaces do not have rigid radial dimensions, unlike in parabolic surface modeling. Free-form splines include the following methods: Cardinal, Hermite, Bézier, and non-uniform rational B-spline (NURBS). A Cardinal spline is a sequence of individual curves joined to form a larger curve, and a Hermite spline uses two points and two tangents to model a 2-D curve. Bézier splines, particularly in their quadratic and cubic forms, are widely used to model smooth curves. To model a quadratic Bézier curve, only three control points are required. The latest fast IME algorithms such as UMHexagonS terminate the final search step using SDSP with the five search points shown in Fig. 5.1 as the smallest search pattern. In particular, each of the IME search points (H_1, C, H_2) and (V_1, C, V_2) correspond to the three control points of a quadratic Bézier curve. A quadratic Bézier curve is also a parabolic segment, but it does not pass by all of the control points. Although the curve is not an interpolation between the control points, it can approximate the data trend. Hence, quadratic Bézier curve based FME techniques are introduced. NURBS, which can be defined by degree, weighted control points, knot vector, and evaluation rules, is currently a very popular type of spline. To model a free-form curve with NURBS, the number of control points must be greater than or equal to four. NURBS is a generalization of B-splines and Bézier splines.

5.4. Quadratic Bézier Curve

In the 1-D parabolic model in (5), the three IME search points (H_1, C, H_2) or (V_1, C, V_2) in Fig. 5.1, are used to predict the best fractional-pixel position at the horizontal or vertical location. As discussed in the previous section, quadratic Bézier curves are a natural choice for this problem, because the three IME search points correspond to the three control points of the quadratic Bézier curve. The quadratic Bézier curve algorithm can be explained by (9) and (10). Equation (9) describes a generalization of the Bézier curve.

$$\begin{aligned} &P(t) = \sum_{i=0}^{n} p_i J_{n,i}(t), \quad (0 \le t \le 1) \\ &J_{n,i}(t) = {}_nC_i t^i (1-t)^{n-i} \\ &{}_nC_i = \frac{n!}{i!(n-i)!} \end{aligned} \tag{9}$$

where n denotes the degree of the Bézier curve, p_0, p_1, …, $p_{n\text{-}1}$, p_n are control points, and ${}_nC_i$ is the binomial coefficient. While the parameter t moves from 0 to 1, the function $P(t)$ traces a curve. Let the matching error costs corresponding to the local positions (x_i, 0) and (0, y_i) be X_i and Y_i. Considering the coordinates for 1-D surface modeling, when x_i or y_i = i-1, the IME search points (H_1, C, H_2) and (V_1, C, V_2) can be represented as {(x_0, X_0), (x_1, X_1), (x_2, X_2)} and {(y_0, Y_0), (y_1, Y_1), (y_2, Y_2)}, respectively. At x_i or y_i = m_i and X_i or Y_i = M_i, each of the coordinates m_i and M_i is entered separately as a control point p_i. The quadratic Bézier curve given by the three control points (p_0, p_1, p_2) is described in (10), which forms the core of the proposed techniques.

$$\begin{aligned} P(t) &= \sum_{i=0}^{2} p_i J_{2,i}(t) \\ &= p_0 \times {}_2C_0 t^0 (1-t)^2 + p_1 \times {}_2C_1 t^1 (1-t)^1 + p_2 \times {}_2C_2 t^2 (1-t)^0 \\ &= p_0 (1-t)^2 + p_1 2t(1-t) + p_2 t^2 \end{aligned} \tag{10}$$

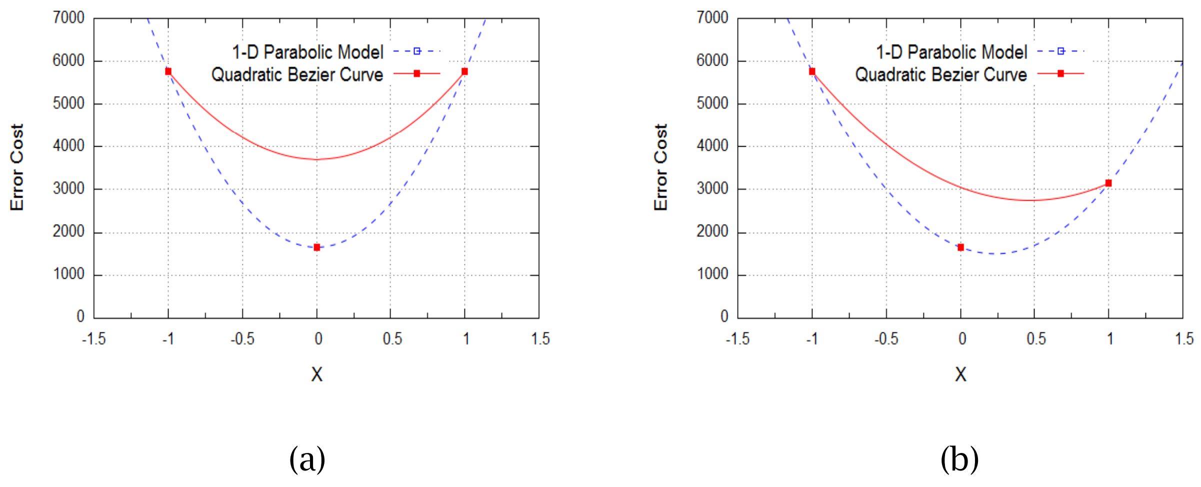


Fig. 5.3. Examples of a 1-D parabolic model and a quadratic Bézier curve. (a) The two curves plotted using the three matching errors located at (-1, 5759), (0, 1659), (1, 5759). (b) The curves at (-1, 5759), (0, 1659), (1, 3146).

Fig. 5.3 shows examples of quadratic Bézier curves and the 1-D parabolic model. As shown, the quadratic Bézier curve does not pass by all three control points, but two points are always passed. The x or y coordinate with the lowest matching error cost will be regarded as the best prediction position x_b or y_b. The best prediction position found by the quadratic Bézier curve, however, tends to be more biased toward x = -1 or 1 than that of the 1-D parabolic model, as illustrated in Fig. 5.3 (b). Thus, a preprocessing algorithm is introduced to correct the one-directional bias.

5.5. Proposed Method 1

In this paper, three FME methods based on quadratic Bézier curves are proposed. The first method differentiates the quadratic Bézier curve in (10) to give:

$$P(t)' = 2t(p_0 - 2p_1 + p_2) - 2(p_0 - p_1) \tag{11}$$

Let $P(t)'$ be zero. When $(p_0, p_1, p_2) = (M_0, M_1, M_2)$, it is possible to obtain the optimum value of t_b that minimizes the matching error cost, as shown below:

$$\begin{aligned} &P(t)' = 2t(p_0 - 2p_1 + p_2) - 2(p_0 - p_1) = 0 \\ &t_b = (p_0 - p_1)/(p_0 - 2p_1 + p_2) \end{aligned} \tag{12}$$

As described in (13), when $(p_0, p_1, p_2) = (m_0, m_1, m_2) = (-1, 0, 1)$, the best fractional-pixel prediction position $P(t_b)$ can be found by substituting the above optimum t_b for t in (10).

$$P(t_b) = p_0(1-t_b)^2 + p_1 2t_b(1-t_b) + p_2 t_b{}^2 = 2t_b - 1 \tag{13}$$

Table 5.1 compares the fractional-pixel motion vector (FMV) found by the 1-D parabolic model based prediction (1-D_PM) and that found by the proposed method (BÉZIER) at quarter-pixel resolution. The matching probabilities given refer to the agreement of the two methods with the best FMV found by FFPS. In the simulation, it is assumed that the FMV matching performance of FFPS is always the best. The QCIF “Salesman” sequence and the CIF “Football” sequence are used as test input images. Both sequences consist of 100 frames. UMHexagonS is used for IME and returns the five neighboring IME matching errors. The best fractional-pixel prediction positions determined by the two algorithms are subjected to quantization operations [69]. Abnormal cases, such as the IME matching error H_1 or H_2 being unknown or zero, are not allowed, and the FME process for the macroblock is skipped in exceptional cases. If the *x* or *y* coordinate of the best FMV found by FFPS is equal to that found by a mathematical model based prediction, it counts the number of matching FMVs in position $|P|$. As shown in Table 5.1 (a) and (b), the 1-D parabolic model based prediction can produce more accurate FMVs than the quadratic Bézier curve based prediction. However, the 1-D parabolic model based prediction can never find the best x_b or y_b located at $|P|>0.5$, unlike the quadratic Bézier curve based prediction. That is, compared with the 1-D parabolic model based prediction, the quadratic Bézier curve approach provides higher robustness to large motions. It should be noted that, in general, macroblocks with larger motions result in higher distortion.

Table 5.1. Fractional-pixel Motion Vector Matching Probability (%)

(a) QCIF "Salesman" Sequence

Method	**Position**	[0,±0.75]	$\|P\| = 0$	$\|P\| = 0.25$	$\|P\| = 0.5$	$\|P\| = 0.75$
1-D_PM	*x*-coord.	44.928	49.969	28.606	14.598	00.000
	y-coord.	39.768	43.321	28.010	15.998	00.000
BÉZIER	*x*-coord.	30.559	33.963	16.497	15.257	30.711
	y-coord.	27.329	30.046	15.285	14.616	27.359

(b) CIF "Football" Sequence

Method	**Position**	[0,±0.75]	$\|P\| = 0$	$\|P\| = 0.25$	$\|P\| = 0.5$	$\|P\| = 0.75$
1-D_PM	*x*-coord.	28.673	50.725	23.179	15.063	00.000
	y-coord.	24.592	39.588	23.874	14.763	00.000
BÉZIER	*x*-coord.	22.465	36.714	13.949	14.273	24.193
	y-coord.	18.278	25.977	13.483	14.364	25.168

5.6. Proposed Method 2

The second method involves predicting p_1' in order to pass close to all three control points. Thus, p_1' should be predicted such that p_1 can exist on the Bézier curve. As p_1 is equal to $P(t = 0.5)$, p_1' can be computed as:

$$
\begin{aligned}
p_1 &= p(\frac{1}{2}) = p_0(1-\frac{1}{2})^2 + 2p_1'\frac{1}{2}(1-\frac{1}{2}) + p_2(\frac{1}{2})^2 \\
p_1' &= \frac{1}{2}(4p_1 - p_0 - p_2)
\end{aligned}
\tag{14}
$$

If p_1 in (12) is replaced by p_1', then the Bézier curve can pass through p_1 as well as p_0 and p_2, as shown below:

$$
t_b = (p_0 - p_1')/(p_0 - 2p_1' + p_2) \tag{15}
$$

Finally, when $(p_0, p_1, p_2) = (M_0, M_1, M_2)$, (16) is used to determine the best fractional-pixel prediction position $P(t_b)$. The result of the prediction is the same as that of the 1-D parabolic model based prediction, although the approach is different. This second proposed method, however, can easily be extended to a third method.

$$
P(t_b) = 2t_b - 1 = (p_0 - p_2)/(2p_0 - 4p_1 + 2p_2) \tag{16}
$$

5.7. Proposed Method 3: Determination of Adjusting Factors

As shown in Table 5.2, each location (x, y) corresponds to the five main IME search points (H_1, H_2, C, V_1, V_2). Table 5.2 (a) shows the average matching error costs at the five IME search points for the QCIF sequence “Claire” (100 frames) and Table 5.2 (b) shows the same information for the CIF sequence “Stefan” (100 frames). In the simulation, the sum of absolute difference (SAD) criterion is used to calculate the matching errors for a given quantization parameter (QP) of 28 and rate-distortion optimized mode (RDO) of 1, based on the H.264/AVC JM version 12.4 reference software [66], [75]. In the case of “Claire,” with reference to Fig. 5.3 (a), the prediction curves are almost symmetric about $x = 0$. For the “Stefan” sequence, on the other hand, the horizontal search points (H_1, C, H_2) form slightly uneven curves, similar to Fig. 5.3 (b). That is, the simulation implies that the FMVs for “Claire” are more center-biased than those for “Stefan.” Actually, since “Claire” comprises stationary and small-motion objects, most of the integer-pixel motion vectors are distributed within the central area. Furthermore, for the quadratic Bézier curve, the following can be assumed:

First, the more similar the matching error H_1 (V_1) is to H_2 (V_2), the closer the best prediction position is to the center. The best prediction position will also be similar to that of the 1-D parabolic model.

Second, the higher or lower H_1 is compared to H_2, the more the best prediction position is biased in one direction. That is, the best prediction position will be located farther away from that of the 1-D parabolic model.

Third, the farther H_1 and H_2 are from C, the closer the best prediction position is to the center. The best prediction position will also be similar to that of the 1-D parabolic model.

Finally, the closer H_1 and H_2 are to C, the more the best prediction position is biased in one direction. That is, the best prediction position will be located farther away from that of the 1-D parabolic model.

Table 5.2. Average IME Matching Error Costs

(a) "Claire"

(x, y)	-1	0	1
-1	–	122.100	–
0	133.955	103.513	133.174
1	–	121.288	–

(b) "Stefan"

(x, y)	-1	0	1
-1	–	311.212	–
0	283.471	236.197	294.131
1	–	308.656	–

Table 5.3. Pseudo-Code for the Third Proposed Method

```
$\boldsymbol{D} = (0.5 \times (4.0 \times p_1 - p_0 - p_2)) - p_1$
$\boldsymbol{AF1}$ = if $(p_0 > p_2)$ then $(p_0 / p_2) - 1.0$, else $(p_2 / p_0) - 1.0$
$\boldsymbol{AF2} = (p_0 + p_2) / (2.0 \times p_1)$
$\boldsymbol{AF3}$ = if $(1.5 > AF2)$ then $AF1 \times 10.0$, else $AF2 - 1.0$
$\boldsymbol{p_1'} = p_1 + (D \times AF3)$
```

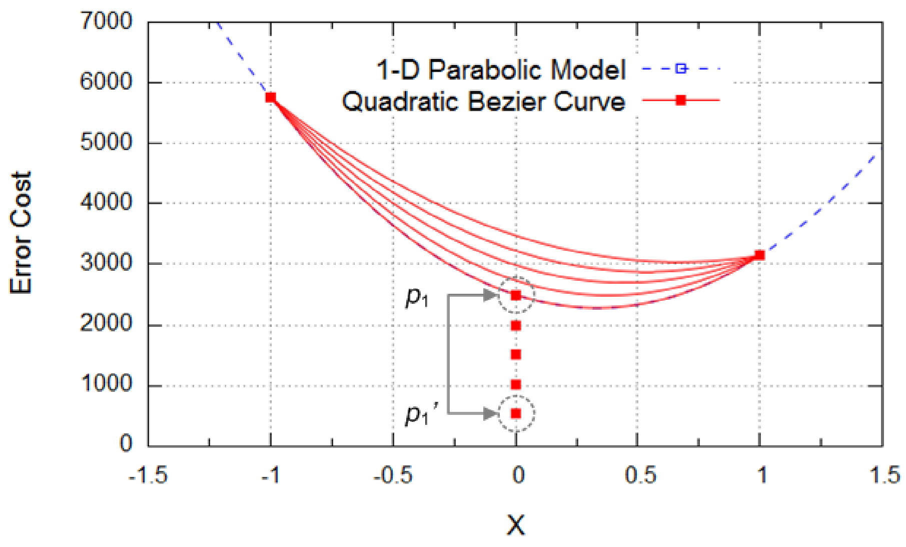


Fig. 5.4. Determination of the Bézier curve by controlling p_1'.

In the second proposed method, it is known that the shape of the quadratic Bézier curve can be determined by controlling the predicted control point p_1', as illustrated in Fig. 5.4. The adjusting factors used in the third proposed method are described by the pseudo-code in Table 5.3. Each adjusting factor is composed according to certain assumptions. Let $(p_0, p_1, p_2) = (M_0, M_1, M_2)$. As shown in Table 5.3, the variable D, which represents the original distance between p_1 and p_1', can be obtained by applying (14), as used in the second method. The adjusting factor $AF1$ is based on the assumption that a higher ratio of p_0 to p_2 will lead to a bigger gap between p_1 and p_1'. The ratio of p_0+p_2 to $2p_1$ gives $AF2$, which represents the relative difference between them. The critical value 1.5 in the second conditional sentence is used to determine the adjusting factor $AF3$. The critical value is experimentally selected to be higher than $AF2 = 1.29$ computed by the matching errors (H_1, C, H_2) in Table 5.2 (b). If $AF2$ is less than the critical value, the relationship between p_0 and p_2 is preferred to that between p_0+p_2 and p_1, in which case $AF3$ is $AF1$ multiplied by 10 determined by many tests. The last line of the pseudo-code shows that the position of p_1' is determined by applying D adjusted by $AF3$. The procedure of the third proposed method for obtaining the best prediction position x_b or y_b can be summarized as follows:

Step_1) The IME process for a prediction block is terminated and returns the IME matching error costs.

Step_2) The adjusting factors for controlling p_1' are determined by the pseudo-code described in Table 5.3.

Step_3) The predicted p_1' is entered in (15), and then the optimum t_b is computed.

Step_4) The best prediction position is found by applying the optimum t_b in (13). The best prediction position is quantized according to a previously reported method [69].

5.8. Proposed Method 3: Error Cost Scaling

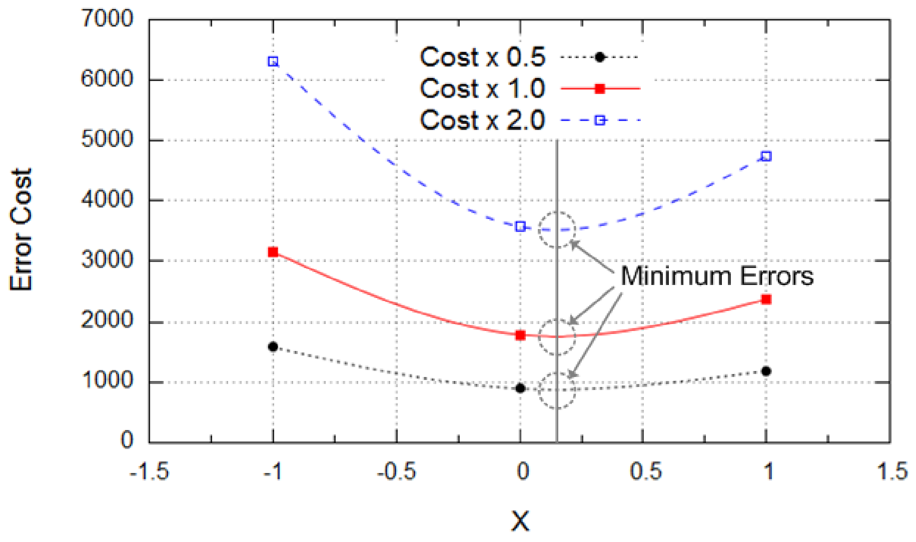


Fig. 5.5. Examples of the error cost scaling. The best prediction *x*-coordinate of the original curve is the same as those of the two distorted curves scaled up and down vertically.

In H.264/AVC JM 12.4, the simulation results may vary slightly with data types, coding style, and compliers. The version 12.4 of JM seems to have some defects. For instance, developers of the reference software can use “float” or “double” as floating-point type but they may get different test results in some cases. If a satisfactory result is not reached, the proposed technique in this section is recommended; the simulations shown below do not use the error cost scaling technique.

As stated above, the third proposed method determines the position of p_1' based on the ratios computed by reusing the IME matching error costs. Here, it should be realized that p_1' is seriously affected by the level of the matching error cost. In particular, the IME matching errors for video images containing large motion objects, such as “Football” and “Stefan,” are very irregular and vigorous. The error cost level can therefore be scaled artificially.

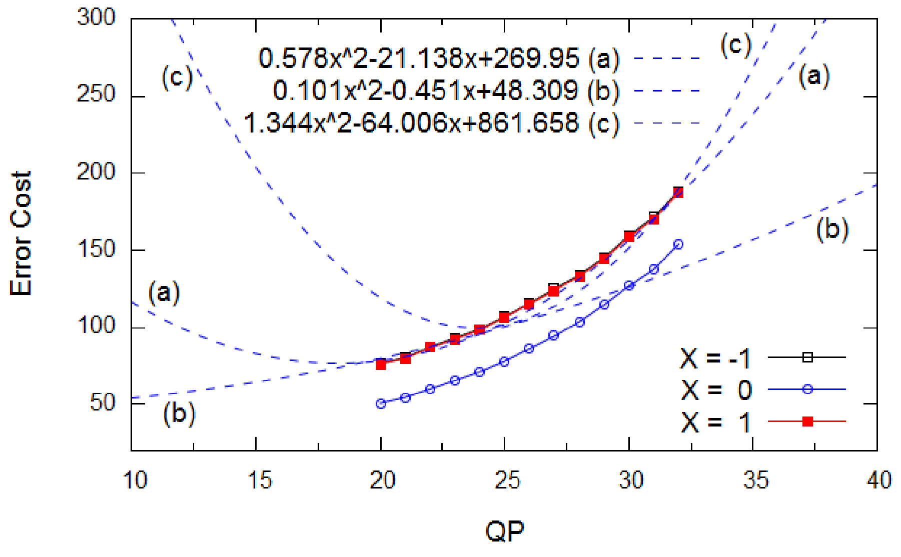


Fig. 5.6. Relationship between the average IME matching error and QP for the QCIF “Claire” sequence. The local position (X, 0) is represented as X.

As shown in Fig. 5.5, there are three curves, including those minimized by 50% and enlarged by 200%, involved in the matching error cost. Due to this scaling, the error cost levels are obviously changed, but the positions on the x-axis are not. Although the IME matching errors are distorted by the scaling, the best prediction position can still be found without any difficulty. That is, the error cost scaling normalizes and distorts the irregular matching error costs, whereas the characteristic shape of the curve is maintained.

In addition, the relationship between the average IME matching error and QP has been considered. Fig. 5.6 represents an extension to the simulation in Table 5.2 (a) and shows variations in matching error at 13 QPs = [20, 32]. As listed in Table 5.3, the adjusting factors were determined based on the average matching errors for “Claire” at QP = 28. However, the average IME matching errors are affected by the QPs, as shown in Fig. 5.6. As a solution to this problem, the error cost scaling process has been designed without modifying the adjusting factors.

Table 5.4. Pseudo-Code for the Error Cost Scaling

```
if ($p_0 < p_2$) then $Max = p_2$, else $Max = p_0$
if ($Max < p_1$) then $Max = p_1$
if (QP $\in$ [20,32]) $CV = 0.578 \times \mathrm{QP}^2 - 21.138 \times \mathrm{QP} + 269.95$
else if (QP $\in$ [0,19]) $CV = 0.101 \times \mathrm{QP}^2 - 0.451 \times \mathrm{QP} + 48.309$
else $CV = 1.344 \times \mathrm{QP}^2 - 64.006 \times \mathrm{QP} + 861.658$
if ($Max > CV$) then
{
$S\_Rate = CV / Max$
$p_0 = p_0 \times S\_Rate$
$p_1 = p_1 \times S\_Rate$
$p_2 = p_2 \times S\_Rate$
}
```

The error cost scaling process can be carried out as described in Table 5.4. First, the highest error cost of the three IME matching errors (p_0, p_1, p_2) = (M_0, M_1, M_2) is obtained. Second, if the maximum error cost is higher than the critical value CV, the maximum error is changed to CV and the scaling rate S_Rate is saved. CV should be adjusted according to QP. With reference to the relationship between the average matching error and QP, as illustrated in Fig. 5.6, the four best CVs (= 78, 97, 130, 186) corresponding to four QPs (= 20, 24, 28, 32) were found. The other CVs can also be approximated using equation fitting. The three coefficients c_1–c_3 of the quadratic function $F(x) = y = c_1x^2+c_2x+c_3$ giving the best fit to a data set can be found by applying the method of least squares, as shown in (17) and (18).

$$E = \sum_{i=1}^{n} \{y_i - (c_1x_i^2 + c_2x_i + c_3)\}^2 \tag{17}$$

$$\begin{bmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} \sum x_i^2 y_i \\ \sum x_i y_i \\ \sum y_i \end{bmatrix}, \ (\sum = \sum_{i=1}^{n}) \tag{18}$$

where n denotes the number of data points (x_i, y_i). In (17), the minimum E can be obtained by differentiating E with respect to each coefficient. The optimum values of c_1–c_3 are found when $\partial E/\partial c_1 = \partial E/\partial c_2 = \partial E/\partial c_3 = 0$, as shown in (18). The three coefficients for the four data points (20, 78), (24, 97), (28, 130), and (32, 186) are computed as follows:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 2155008 & 76544 & 2784 \\ 76544 & 2784 & 104 \\ 2784 & 104 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 379456 \\ 13480 \\ 491 \end{bmatrix} \approx \begin{bmatrix} 0.578 \\ -21.138 \\ 269.95 \end{bmatrix} \tag{19}$$

The dotted line (a) in Fig. 5.6 shows the result of fitting the data points using the above coefficients to determine the best *CV* at QPs = [20, 32], as described in Table 5.4. The above function, however, cannot approximate all the best *CV*s for QPs = [0, 51] due to limitations in the quadratic function. Therefore, two additional quadratic functions, corresponding to dotted lines (b) and (c) in Fig. 5.6, are used for QPs = [0, 19] and QPs = [33, 51], respectively. Each of the two functions is determined by the two sets of data points {(0, 48), (10, 55), (20, 78), (24, 97)} and {(24, 97), (28, 130), (32, 186), (51, 1094)}, which are the best data points found experimentally based on the relationship in Fig. 5.6. At the end of the error cost scaling process, the scaling rate is applied to the other matching error costs. After that, the prediction process of the third proposed method starts. The modified procedure of the third proposed method for obtaining the best prediction position x_b or y_b can be summarized as follows:

Step_1) If the IME process for a macroblock is terminated and returns the IME matching errors, the error cost scaling process in Table 5.4 is performed.

Step_2) The adjusting factors for controlling p_1' are determined by the pseudo-code described in Table 5.3.

Step_3) The predicted p_1' is entered in (15), and then the optimum t_b is computed.

Step_4) The best prediction position is found by applying the optimum t_b in (13). The best prediction position is quantized according to a previously reported method [69].

5.9. Proposed Method 3: Modification for H.265/HEVC

H.265/HEVC is the latest video compression standard suitable for high-resolution video formats such as WQXGA, 4K, and 8K. In place of the 16 × 16 macroblock adopted in H.264/AVC, H.265/HEVC is based on a coding tree unit (CTU) with a maximum size of 64 × 64. The CTU consists of three blocks: a luma coding tree block (CTB), two chroma CTBs, and syntax elements. Since larger CTB sizes generally produce a lower bitrate, large CTB sizes have a strong influence on the coding efficiency with high-resolution video. Each CTB can be split into multiple coding units (CUs). The CU also consists of three blocks: a luma coding block (CB), two chroma CBs, and syntax elements. Each CU is again partitioned into prediction units (PUs) and a quadtree of transform units (TUs). The CU level determines the inter or intra prediction mode, and each CB can be split into prediction blocks (PBs). Larger blocks tend to lead to higher matching error costs. Therefore, the adjusting factor *AF*3 described in Table 5.3 is modified to suit the H.265/HEVC test model (HM) version 12.0 reference software [67], as shown below:

$$\boldsymbol{AF3} = \text{if } (4.0 > AF2) \text{ then } AF1, \text{ else } AF2 - 2.0 \tag{20}$$

H.265/HEVC utilizes 7- or 8-tap interpolation filters for the motion vector refinement process at quarter-pixel resolution, whereas H.264/AVC performs a two-step interpolation: 6-tap interpolation filtering for half-pixel resolution positions followed by a bilinear interpolation for quarter-pixel resolution positions. Both H.265/HEVC and H.264/AVC refine the motion vectors at the quarter-pixel resolution; thus, the quantization process of [69] is also used in the final prediction step of the third proposed method in H.265/HEVC.

5.10. Experimental Results Based on H.264/AVC

The proposed techniques have been implemented in H.264/AVC JM 12.4 on the Windows 7 64-bit OS platform with an Intel i5 CPU@1.80 GHz, and H.265/HEVC HM 12.0 on the Windows 7 64-bit OS platform with an Intel i7 CPU@2.80 GHz. Based on the H.264/AVC JM, the simulation was conducted using the default settings, i.e., search range = 16, QPs = 20, 24, 28, and 32, and RDO = 1 under the baseline profile. As implemented in JM, UMHexagonS is used for fast IME. The performance of the proposed methods (METHOD_1–3) is evaluated by comparison with that of CBFPS and the 1-D parabolic model based prediction (1-D_PM) in terms of PSNR and bitrate. The computational complexity can be compared in terms of the total motion estimation time (MET) and the average number of fractional-pixel search points per block (FSP). CBFPS is chosen as one of the most popular interpolation based FME algorithms. As defined in JM, if a macroblock type is less than or equal to 3, UMHexagonS is performed, followed by FFPS rather than CBFPS for FME. The rule, therefore, is applied to all the FME methods for fair comparison, and the fractional-pixel search points used in FFPS are not counted. In addition, the mathematical model based methods have been implemented in the IME module of JM. However, because the PSNR and bitrate in IME are measured, the computational complexity will only be mentioned briefly. The six sequences used in the test are as follows: QCIF (176×144) "Claire" and "Salesman," CIF (352×288) "Football," "News," "Stefan," and "Table." Each sequence includes different types of motion, which can be classified as small, middling, and large motion. The number of frames to be encoded is 100 at 30 Hz.

As listed in Table 5.5–Table 5.9, the average quality performance of METHOD_3 at each QP is better than that of 1-D_PM. The average PSNR degradation of 0.016 with respect to CBFPS is lower than the 0.025 attained by 1-D_PM. In particular, the average PSNR drop at each QP = 28 and QP = 32 is 0.003, which means that the quality is very close to that of CBFPS. In terms of computational complexity, METHOD_3 has no use for FSPs, whereas CBFPS requires at least five. METHOD_1–3 and 1-D_PM achieve an average MET reduction of 17.198%, 17.824%, 16.894%, and 17.033%, respectively, with respect to CBFPS. The computational load of METHOD_3 can also be reduced from about 11% to 30% compared with CBFPS. When METHOD_3 is directly implemented in the IME module of JM, the average MET reduction is about 46%, whereas the quality is expected to degrade a little compared with that in the FME module with the above simulation conditions. The bitrate comparisons in Table 5.5–Table 5.9 show that METHOD_3 is competitive with 1-D_PM. The bitrate of METHOD_3 shows an average increase of less than 0.081% with respect to 1-D_PM. On the other hand, the average PSNR degradation of 0.022 for METHOD_1 with respect to CBFPS is lower than that of 1-D_PM, but the bitrate increase compared with 1-D_PM averages about 0.2%. The performance of METHOD_2 is equal to that of 1-D_PM with similar complexity.

5.11. Experimental Results Based on H.265/HEVC

Based on the H.265/HEVC HM 12.0, the simulation was carried out under the main profile using the following configuration parameters: group of pictures (GOP) size = 4, intra period = -1, search range = 64, and decoding refresh type = 0. The four QPs used in this test are 22, 27, 32, and 37. As adopted in HM, TZS is selected as a fast IME search algorithm. As mentioned above, the results of the simulation conducted for H.264/AVC JM 12.4 show that the performances of METHOD_1 and METHOD_2 are very similar or equal to that of 1-D_PM, whereas METHOD_3 is better than 1-D_PM for most of the test sequences at each QP. In this simulation based on HM, the performance of METHOD_3 is therefore directly compared with that of 1-D_PM. FFPS implemented in HM is selected as the anchor FME algorithm. The performance comparisons are shown in Table 5.10 and Table 5.11, where Δ PSNR represents the PSNR difference between 1-D_PM and FFPS, or METHOD_3 and FFPS, which means a PSNR degradation with respect to FFPS; Δ Bitrate denotes the bitrate increase in percentage with respect to FFPS; and Δ Encode_T, Δ Inter_T, and Δ FME_T represent the total encoding time reduction, inter prediction time reduction, and FME time reduction in percentage with respect to FFPS, respectively. The Bjøntegaard delta (BD) PSNR (BD-PSNR) and bitrate (BD-Bitrate) [31] are used to evaluate the objective differences between the two rate-distortion curves. Similar to METHOD_3 implemented in JM, METHOD_3 was also implemented without using an interpolation process (upsampling) in the FME module of HM. Since the HM module without upsampling does not accurately measure the PSNR and bitrate, only the complexity reduction is reported; Table 5.10 lists these values in parentheses. The eight sequences used in the test are as follows: WQVGA (416 × 240) “BlowingBubbles” (50 Hz) and “BQSquare” (60 Hz), WVGA (832 × 480) “BQMall” (60 Hz) and “PartyScene” (50 Hz), 720p (1280 × 720) “Johnny” (60 Hz), 1080p (1920 × 1080) “BQTerrace” (60 Hz), WQXGA (2560 × 1600) “SteamLocomotiveTrain” (60 Hz), and “Traffic” (30 Hz). The number of frames to be encoded is 100 for each sequence.

As shown in Table 5.10, the average PSNR and bitrate performances of METHOD_3 are better than those of 1-D_PM. The PSNR degradation for most of the test sequences against four QPs is lower compared with 1-D_PM. The BD-PSNR of METHOD_3 shown in Table 5.11 is also higher than that of 1-D_PM. In particular, the PSNR value for “SteamLocomotiveTrain” is considerably close to that of FFPS, whereas the bitrate increase with respect to FFPS is negligible and lower than that of 1-D_PM. The bitrate for the WQVGA and WVGA sequences shows a much lower increase compared with 1-D_PM. Fig.

5.7 shows the rate-distortion (R-D) curves for the eight sequences with QPs = 22, 27, 32, and 37. The R-D curves show that the R-D performances for the WQVGA and WVGA sequences are better than with 1-D_PM, and the performances for the 720p, 1080p, and WQXGA sequences are close to that of FFPS. In terms of computational complexity, METHOD_3 and 1-D_PM do not require any FSPs, whereas FFPS always uses 16 FSPs. As shown in Table 5.10, the total encoding time of METHOD_3 can be reduced from about 6% to 25%, and the average FME time reduction is 41.465%, with respect to FFPS. When METHOD_3 is implemented without upsampling in the encoder, the encoding time can be dramatically reduced from about 14% to 46% with an FME time reduction of about 99%. METHOD_3 with and without upsampling achieves an average encoding time of 13.427% and 30.625%, respectively. METHOD_3 also has a similar complexity as 1-D_PM.

5.12. Summary

The proposed low-complexity interpolation-free fractional-pixel ME algorithms, METHOD_1–3, were developed to achieve much further reduction in computational complexity compared with the existing interpolation based FME algorithms, with a reasonable rate-distortion degradation. The proposed algorithms are designed based on a quadratic Bézier spline as an alternative to the parabolic prediction models having some drawbacks. The quadratic Bézier spline model based prediction can approximate the best prediction position greater than 0.5 or less than -0.5. Particularly, the proposed METHOD_3 adjusts the predicted position by using a specific threshold. The results of the simulations for various video test sequences demonstrate that the performance of METHOD_3 is superior to the conventional 1-D parabolic model based prediction. In addition, the simplicity of the algorithm will make it suitable for hardware applications. It can be directly implemented in the IME module, and easily extended to 1/8 or 1/16 pixel resolution ME.

Table 5.5. Performance Comparison in H.264/AVC (QP=20)

Sequence	Method	PSNR (dB)	Bitrate (kbps)	MET (ms)	FSP
Claire (QCIF)	CBFPS	45.713	109.838	7091	4.904
	1-D_PM	45.674	116.882	5174	0.000
	METHOD_1	45.676	116.561	5646	0.000
	METHOD_2	45.674	116.882	5340	0.000
	METHOD_3	45.672	116.225	5918	0.000
Salesman (QCIF)	CBFPS	42.217	180.509	7009	5.085
	1-D_PM	42.175	195.919	5679	0.000
	METHOD_1	42.163	195.487	5324	0.000
	METHOD_2	42.175	195.919	5990	0.000
	METHOD_3	42.171	195.960	5434	0.000
Football (CIF)	CBFPS	43.595	4009.248	62397	8.589
	1-D_PM	43.564	4103.251	52649	0.000
	METHOD_1	43.575	4110.252	52252	0.000
	METHOD_2	43.564	4103.251	52831	0.000
	METHOD_3	43.578	4103.340	52029	0.000
News (CIF)	CBFPS	43.651	669.869	31116	5.257
	1-D_PM	43.605	703.241	24109	0.000
	METHOD_1	43.601	703.435	24599	0.000
	METHOD_2	43.605	703.241	24820	0.000
	METHOD_3	43.601	701.443	24353	0.000
Stefan (CIF)	CBFPS	43.299	4244.916	44922	6.871
	1-D_PM	43.264	4325.671	38559	0.000
	METHOD_1	43.259	4330.272	37568	0.000
	METHOD_2	43.264	4325.671	38189	0.000
	METHOD_3	43.265	4324.531	38932	0.000
Table (CIF)	CBFPS	42.699	2918.530	42427	7.370
	1-D_PM	42.640	3045.521	35257	0.000
	METHOD_1	42.647	3047.391	34504	0.000
	METHOD_2	42.640	3045.521	34586	0.000
	METHOD_3	42.645	3040.949	34318	0.000

Table 5.6. Performance Comparison in H.264/AVC (QP=24)

Sequence	Method	PSNR (dB)	Bitrate (kbps)	MET (ms)	FSP
Claire (QCIF)	CBFPS	42.715	61.224	7324	4.795
	1-D_PM	42.732	64.267	5822	0.000
	METHOD_1	42.712	64.154	5936	0.000
	METHOD_2	42.732	64.267	6043	0.000
	METHOD_3	42.733	63.845	5842	0.000
Salesman (QCIF)	CBFPS	38.947	103.531	7708	5.193
	1-D_PM	38.907	113.136	6331	0.000
	METHOD_1	38.915	113.371	6300	0.000
	METHOD_2	38.907	113.136	6635	0.000
	METHOD_3	38.919	113.280	6408	0.000
Football (CIF)	CBFPS	40.464	2637.679	62989	8.137
	1-D_PM	40.464	2711.268	53192	0.000
	METHOD_1	40.456	2714.177	51434	0.000
	METHOD_2	40.464	2711.268	52990	0.000
	METHOD_3	40.469	2710.956	53893	0.000
News (CIF)	CBFPS	41.135	390.722	31177	5.092
	1-D_PM	41.101	410.918	25670	0.000
	METHOD_1	41.097	411.953	25824	0.000
	METHOD_2	41.101	410.918	24434	0.000
	METHOD_3	41.103	410.990	25817	0.000
Stefan (CIF)	CBFPS	39.844	2560.536	45628	6.630
	1-D_PM	39.810	2631.269	39913	0.000
	METHOD_1	39.813	2634.099	38893	0.000
	METHOD_2	39.810	2631.269	37880	0.000
	METHOD_3	39.814	2630.933	38720	0.000
Table (CIF)	CBFPS	39.253	1591.188	45107	7.075
	1-D_PM	39.212	1667.450	35327	0.000
	METHOD_1	39.213	1672.877	36076	0.000
	METHOD_2	39.212	1667.450	36013	0.000
	METHOD_3	39.214	1667.462	36696	0.000

Table 5.7. Performance Comparison in H.264/AVC (QP=28)

Sequence	Method	PSNR (dB)	Bitrate (kbps)	MET (ms)	FSP
Claire (QCIF)	CBFPS	39.716	33.324	8025	4.660
	1-D_PM	39.676	34.181	6292	0.000
	METHOD_1	39.701	34.135	6369	0.000
	METHOD_2	39.676	34.181	6286	0.000
	METHOD_3	39.741	34.351	6656	0.000
Salesman (QCIF)	CBFPS	35.799	59.638	9463	5.286
	1-D_PM	35.763	63.914	7198	0.000
	METHOD_1	35.790	64.222	7647	0.000
	METHOD_2	35.763	63.914	7211	0.000
	METHOD_3	35.789	64.397	6621	0.000
Football (CIF)	CBFPS	37.576	1730.412	62116	7.493
	1-D_PM	37.563	1772.098	53280	0.000
	METHOD_1	37.560	1779.343	53778	0.000
	METHOD_2	37.563	1772.098	54182	0.000
	METHOD_3	37.573	1775.998	54539	0.000
News (CIF)	CBFPS	38.517	230.314	31024	4.920
	1-D_PM	38.500	241.286	26357	0.000
	METHOD_1	38.522	242.534	26760	0.000
	METHOD_2	38.500	241.286	25435	0.000
	METHOD_3	38.510	240.792	27752	0.000
Stefan (CIF)	CBFPS	36.452	1441.358	46835	6.459
	1-D_PM	36.436	1502.765	39502	0.000
	METHOD_1	36.439	1506.394	39958	0.000
	METHOD_2	36.436	1502.765	39168	0.000
	METHOD_3	36.443	1501.999	40728	0.000
Table (CIF)	CBFPS	36.250	861.838	45089	6.591
	1-D_PM	36.225	903.982	38780	0.000
	METHOD_1	36.227	903.686	37849	0.000
	METHOD_2	36.225	903.982	38361	0.000
	METHOD_3	36.235	902.755	38522	0.000

Table 5.8. Performance Comparison in H.264/AVC (QP=32)

Sequence	Method	PSNR (dB)	Bitrate (kbps)	MET (ms)	FSP
Claire (QCIF)	CBFPS	36.753	18.552	8261	4.535
	1-D_PM	36.749	18.607	7019	0.000
	METHOD_1	36.750	18.727	7038	0.000
	METHOD_2	36.749	18.607	6511	0.000
	METHOD_3	36.785	18.895	6900	0.000
Salesman (QCIF)	CBFPS	32.700	33.643	9604	5.294
	1-D_PM	32.655	34.548	9262	0.000
	METHOD_1	32.670	34.735	8663	0.000
	METHOD_2	32.655	34.548	8752	0.000
	METHOD_3	32.657	34.704	8389	0.000
Football (CIF)	CBFPS	34.581	1068.910	66632	6.921
	1-D_PM	34.582	1097.966	54539	0.000
	METHOD_1	34.578	1097.710	55426	0.000
	METHOD_2	34.582	1097.966	55270	0.000
	METHOD_3	34.581	1095.737	54461	0.000
News (CIF)	CBFPS	35.626	135.864	32857	4.826
	1-D_PM	35.627	140.616	29296	0.000
	METHOD_1	35.627	141.439	28093	0.000
	METHOD_2	35.627	140.616	27745	0.000
	METHOD_3	35.638	140.976	28484	0.000
Stefan (CIF)	CBFPS	32.796	671.957	47462	6.372
	1-D_PM	32.787	715.740	41516	0.000
	METHOD_1	32.803	716.966	41070	0.000
	METHOD_2	32.787	715.740	39393	0.000
	METHOD_3	32.799	715.795	40697	0.000
Table (CIF)	CBFPS	33.252	438.204	53105	5.979
	1-D_PM	33.228	455.419	40769	0.000
	METHOD_1	33.234	458.244	41715	0.000
	METHOD_2	33.228	455.419	39176	0.000
	METHOD_3	33.229	456.706	42093	0.000

Table 5.9. Average Values with Respect to CBFPS (QPs=20, 24, 28, 32)

Method	**Δ PSNR (dB)**	**Δ Bitrate Increase (%)**	**Δ MET Reduction (%)**	**FSP**
1-D_PM	-0.025	4.333	17.033	0.000
METHOD_1	-0.022	4.537	17.198	0.000
METHOD_2	-0.025	4.333	17.824	0.000
METHOD_3	-0.016	4.414	16.894	0.000

Table 5.10. Performance Comparison in H.265/HEVC
(a) PSNR and Bitrate with Respect to FFPS

Sequence	QP	Method	Δ PSNR (dB)	Δ Bitrate (%)
BlowingBubbles (WQVGA)	22	1-D_PM	-0.056	4.097
		METHOD_3	-0.051	3.469
	27	1-D_PM	-0.096	3.462
		METHOD_3	-0.084	3.052
	32	1-D_PM	-0.095	2.321
		METHOD_3	-0.092	2.431
	37	1-D_PM	-0.068	1.726
		METHOD_3	-0.080	1.343
BQSquare (WQVGA)	22	1-D_PM	-0.065	4.668
		METHOD_3	-0.060	3.545
	27	1-D_PM	-0.135	5.812
		METHOD_3	-0.115	4.163
	32	1-D_PM	-0.151	5.310
		METHOD_3	-0.124	3.678
	37	1-D_PM	-0.129	3.476
		METHOD_3	-0.119	2.752
BQMall (WVGA)	22	1-D_PM	-0.033	2.973
		METHOD_3	-0.032	2.708
	27	1-D_PM	-0.060	2.976
		METHOD_3	-0.050	2.825
	32	1-D_PM	-0.083	1.853
		METHOD_3	-0.078	2.011
	37	1-D_PM	-0.072	1.106
		METHOD_3	-0.073	1.510
PartyScene (WVGA)	22	1-D_PM	-0.041	3.535
		METHOD_3	-0.034	2.786
	27	1-D_PM	-0.093	3.591
		METHOD_3	-0.072	3.153
	32	1-D_PM	-0.095	2.954
		METHOD_3	-0.080	2.686
	37	1-D_PM	-0.087	1.759
		METHOD_3	-0.085	1.782
Johnny (720p)	22	1-D_PM	-0.030	2.525
		METHOD_3	-0.029	2.553
	27	1-D_PM	-0.050	2.336
		METHOD_3	-0.046	2.530
	32	1-D_PM	-0.058	1.577
		METHOD_3	-0.056	1.488
	37	1-D_PM	-0.022	1.303
		METHOD_3	-0.033	1.456

BQTerrace (1080p)	22	1-D_PM	-0.011	1.767
		METHOD_3	-0.011	1.529
	27	1-D_PM	-0.060	1.541
		METHOD_3	-0.057	1.341
	32	1-D_PM	-0.057	0.815
		METHOD_3	-0.053	0.609
	37	1-D_PM	-0.038	0.666
		METHOD_3	-0.040	0.343
Steam LocomotiveTrain (WQXGA)	22	1-D_PM	-0.011	0.432
		METHOD_3	-0.007	0.598
	27	1-D_PM	-0.029	0.382
		METHOD_3	-0.029	0.292
	32	1-D_PM	-0.022	0.450
		METHOD_3	-0.021	0.237
	37	1-D_PM	-0.013	0.055
		METHOD_3	-0.015	0.380
Traffic (WQXGA)	22	1-D_PM	-0.058	3.015
		METHOD_3	-0.054	2.831
	27	1-D_PM	-0.066	2.571
		METHOD_3	-0.065	2.575
	32	1-D_PM	-0.064	1.720
		METHOD_3	-0.066	1.945
	37	1-D_PM	-0.052	0.761
		METHOD_3	-0.059	1.046
Average		1-D_PM	-0.063	2.298
		METHOD_3	-0.059	2.051

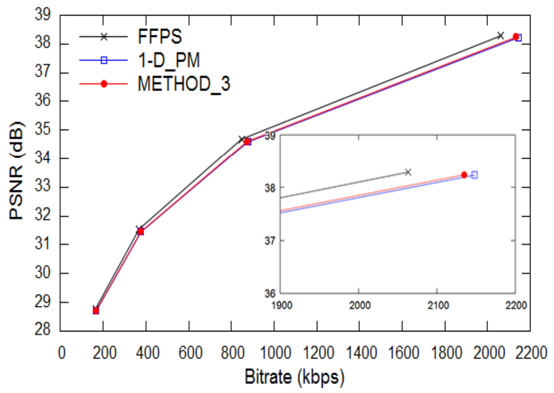
(b) Computational Complexity with Respect to FFPS

Sequence	QP	Method	Δ Encode_T (%)	Δ Inter_T (%)	Δ FME_T (%)	FSP
BlowingBubbles (WQVGA)	22	1-D_PM	06.605	20.893	45.505	0.000
		METHOD_3	06.765 (16.756)	20.278 (45.537)	42.205 (99.451)	0.000
	27	1-D_PM	09.703	20.395	41.924	0.000
		METHOD_3	09.443 (22.857)	18.762 (45.533)	39.981 (99.285)	0.000
	32	1-D_PM	12.381	20.065	40.198	0.000
		METHOD_3	11.627 (28.592)	19.730 (46.861)	42.663 (99.138)	0.000
	37	1-D_PM	13.844	21.181	39.299	0.000
		METHOD_3	12.934 (32.319)	19.909 (49.087)	39.858 (99.049)	0.000
BQSquare (WQVGA)	22	1-D_PM	06.884	25.533	40.787	0.000
		METHOD_3	06.584 (17.505)	21.687 (60.067)	37.350 (99.327)	0.000
	27	1-D_PM	10.263	25.566	39.626	0.000
		METHOD_3	09.702 (24.521)	22.496 (61.176)	39.351 (99.323)	0.000
	32	1-D_PM	14.061	26.712	42.459	0.000
		METHOD_3	13.287 (33.747)	24.523 (60.762)	39.034 (99.129)	0.000
	37	1-D_PM	16.015	26.860	42.440	0.000
		METHOD_3	14.704 (38.832)	24.476 (62.249)	38.732 (99.005)	0.000
BQMall (WVGA)	22	1-D_PM	12.845	22.536	45.810	0.000
		METHOD_3	12.780 (20.808)	22.317 (42.460)	44.808 (99.369)	0.000
	27	1-D_PM	12.151	20.747	43.761	0.000
		METHOD_3	11.163 (21.163)	18.945 (40.792)	42.506 (99.246)	0.000
	32	1-D_PM	16.803	23.920	45.141	0.000
		METHOD_3	14.133 (28.021)	22.610 (44.749)	42.132 (99.335)	0.000
	37	1-D_PM	16.885	24.084	44.590	0.000
		METHOD_3	15.456 (31.359)	21.009 (47.719)	42.455 (99.301)	0.000
PartyScene (WVGA)	22	1-D_PM	06.306	19.727	42.969	0.000
		METHOD_3	06.148 (14.509)	18.025 (43.829)	40.476 (99.150)	0.000
	27	1-D_PM	08.888	20.168	42.282	0.000
		METHOD_3	08.203 (20.633)	18.994 (45.708)	41.422 (99.279)	0.000
	32	1-D_PM	11.329	20.805	41.967	0.000
		METHOD_3	10.749 (26.422)	20.057 (48.203)	41.220 (99.288)	0.000
	37	1-D_PM	13.289	21.779	42.206	0.000
		METHOD_3	12.174 (31.077)	20.179 (50.052)	40.470 (99.249)	0.000
Johnny (720p)	22	1-D_PM	14.021	24.924	40.944	0.000
		METHOD_3	13.301 (34.201)	23.849 (59.671)	40.169 (99.205)	0.000
	27	1-D_PM	17.115	25.626	40.499	0.000
		METHOD_3	16.176 (41.398)	24.738 (61.997)	39.776 (99.289)	0.000
	32	1-D_PM	18.715	26.707	40.819	0.000
		METHOD_3	17.334 (45.036)	24.926 (63.460)	39.834 (99.268)	0.000
	37	1-D_PM	18.926	26.828	40.157	0.000
		METHOD_3	18.176 (46.667)	26.091 (64.623)	40.137 (99.251)	0.000

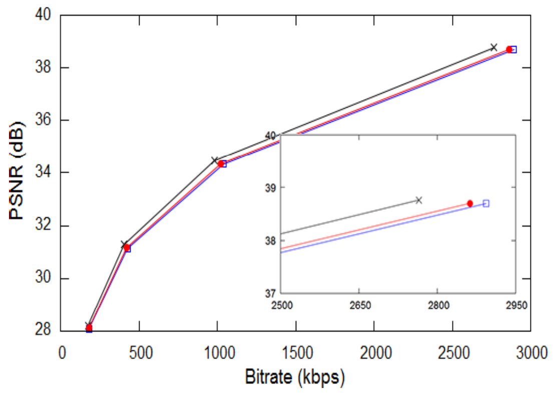
BQTerrace (1080p)	22	1-D_PM	07.428	21.562	42.372	0.000
		METHOD_3	07.282 (18.115)	19.925 (47.629)	41.117 (99.257)	0.000
	27	1-D_PM	11.554	21.636	41.939	0.000
		METHOD_3	10.995 (27.627)	20.980 (50.914)	40.930 (99.259)	0.000
	32	1-D_PM	15.118	22.935	41.049	0.000
		METHOD_3	14.169 (36.615)	21.584 (54.212)	39.726 (99.211)	0.000
	37	1-D_PM	16.384	22.928	40.776	0.000
		METHOD_3	15.568 (39.882)	22.061 (55.330)	40.004 (99.252)	0.000
Steam LocomotiveTrain (WQXGA)	22	1-D_PM	16.058	26.174	47.748	0.000
		METHOD_3	15.937 (26.895)	25.701 (43.724)	47.356 (99.345)	0.000
	27	1-D_PM	22.268	28.674	48.868	0.000
		METHOD_3	21.594 (33.623)	27.634 (50.679)	47.773 (99.380)	0.000
	32	1-D_PM	24.757	30.214	49.250	0.000
		METHOD_3	23.020 (37.768)	28.176 (54.539)	47.700 (99.350)	0.000
	37	1-D_PM	27.138	31.814	49.184	0.000
		METHOD_3	25.568 (41.814)	30.052 (60.733)	48.135 (99.426)	0.000
Traffic (WQXGA)	22	1-D_PM	11.738	24.647	42.238	0.000
		METHOD_3	10.496 (28.169)	22.646 (56.529)	40.334 (99.294)	0.000
	27	1-D_PM	13.944	23.933	40.968	0.000
		METHOD_3	12.992 (33.434)	22.841 (56.353)	40.343 (99.270)	0.000
	32	1-D_PM	15.977	24.365	40.406	0.000
		METHOD_3	15.075 (38.689)	23.223 (57.887)	39.343 (99.282)	0.000
	37	1-D_PM	17.013	24.701	40.708	0.000
		METHOD_3	16.116 (40.947)	23.390 (58.350)	39.531 (99.281)	0.000
Average		1-D_PM	14.263	24.020	42.778	0.000
		METHOD_3	13.427 (30.625)	22.557 (52.857)	41.465 (99.267)	0.000

Table 5.11. Bjøntegaard Delta Performance Comparison in H.265/HEVC

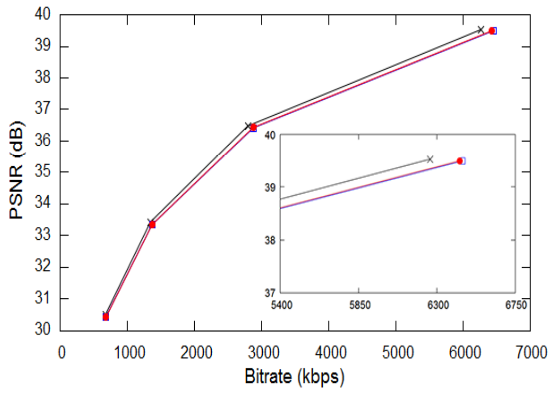
Sequence	**BD-PSNR (dB)**		**BD-Bitrate (%)**	
	1-D_PM	METHOD_3	1-D_PM	METHOD_3
BlowingBubbles	-0.197	-0.183	5.432	4.992
BQSquare	-0.323	-0.252	8.902	6.874
BQMall	-0.160	-0.154	4.030	3.890
PartyScene	-0.218	-0.188	5.314	4.560
Johnny	-0.087	-0.088	3.969	3.959
BQTerrace	-0.075	-0.069	4.285	3.941
SteamLocomoti_	-0.030	-0.030	1.549	1.496
Traffic	-0.126	-0.129	4.213	4.332
Average	-0.152	-0.137	4.712	4.255



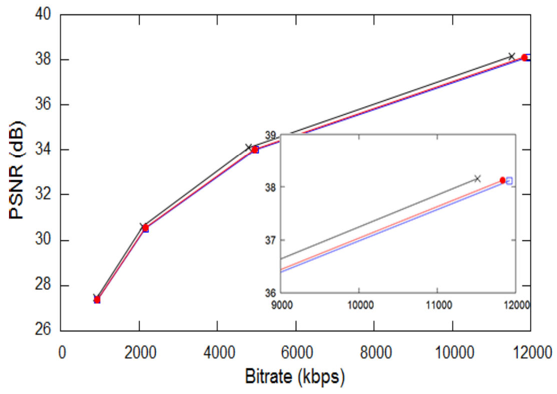
(a) “BlowingBubbles”



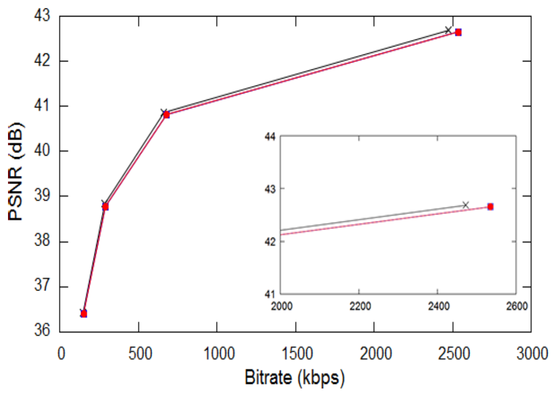
(b) “BQSquare”



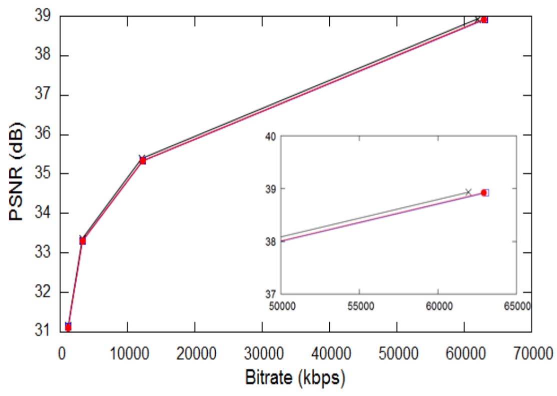
(c) “BQMall”



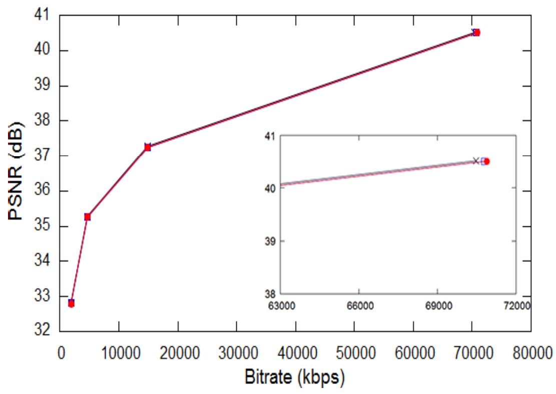
(d) “PartyScene”



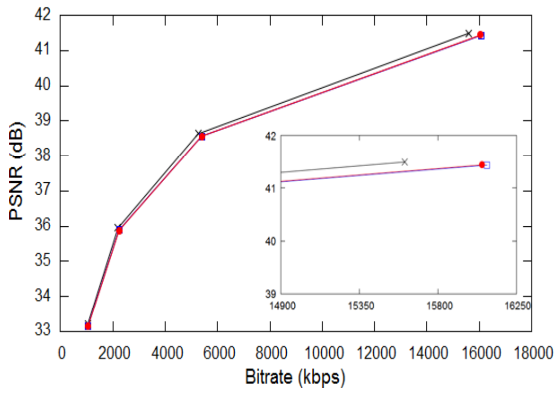
(e) “Johnny”



(f) “BQTerrace”



(g) “SteamLocomotiveTrain”



(h) “Traffic”

Fig. 5.7. Rate-distortion curves for the sequences in H.265/HEVC.

Chapter 6

ENHANCED 1-D PARABOLIC PREDICTION BASED FME

6.1. Introduction

In general, a motion estimator in today's video coding includes fractional-pixel motion estimation (FME) as well as integer pixel motion estimation (IME). FME can provide better quality performance at the cost of higher computational complexity than IME alone. Even though the existing parabolic prediction models [68], [69], [70], [71], have contributed to a significant reduction in computational complexity with minimizing dependence on interpolation filter, the reconstructed image quality is not fully satisfactory for practical applications. In this chapter, thus, the research focuses on developing a high-performance interpolation-free based approach for FME with extremely minimizing the use of the fractional-pixel search points. The proposed modified parabolic prediction based FME technique is based on a linear function employing specific correction coefficients to further improve the performance of the existing parabolic prediction model. In the simulation results, compared with the conventional prediction model based algorithm, the proposed technique can produce more accurate predicted fractional-pixel motion vectors at lower bitrate, whereas it requires a few fractional-pixel search points in terms of computational complexity.

6.2. Related Work

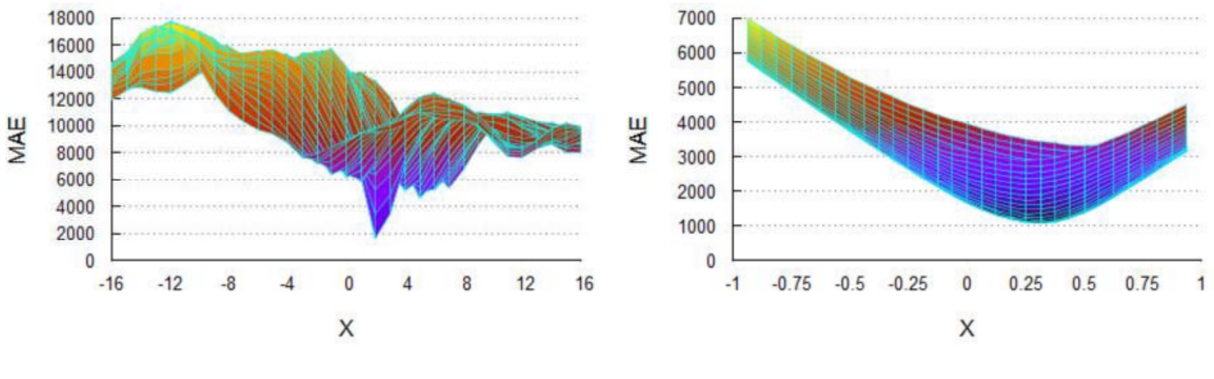


Fig. 6.1. Two error surfaces for the CCIR601 "Garden" sequence. (a) Error surface of IME at integer-pixel MV resolution. (b) Error surface of FME at 1/16-pixel MV resolution.

Two error surfaces are illustrated in Fig. 6.1. They were simulated for the CCIR601 "Garden" sequence with the search range of ±16. 1/16-pixel MV resolution is used for FME. The FME error surface in Fig. 6.1 (b) is undoubtedly unimodal but that of IME in Fig. 6.1 (a) is irregular. The fractional-pixel search points within the FME search area are produced by performing the interpolation process reusing the IME error costs. Accordingly, the FME error cost increases monotonically as the search point moves away from the best point with the minimum error cost. The side of a FME error surface is also shaped like parabola, as shown in Fig. 6.1 (b). To plot FME error surfaces using a parabolic model, a degenerate quadratic parabolic prediction function is introduced in (1), as already mentioned in [69].

$$Q(x, y) = c_4 x^2 + c_3 x + c_2 y^2 + c_1 y + c_0 \tag{1}$$

where x and y denote fractional-pixel position. c_0, c_1, c_2, c_3, and c_4 are the five coefficients of the function Q. If the five integer-pixel search points S_0 = (0,0), S_1 = (-1,0), S_2 = (1,0), S_3 = (0,-1), and S_4 = (0,1) are checked by the IME process, c_0–c_4 can be computed by substituting the coordinates and error costs of the five integer-pixel search points for the variables of (1). As $Q(S_0) = Q(0,0) = c_0$, $Q(S_1) = Q(-1,0) = c_4 - c_3 + c_0$, $Q(S_2) = Q(1,0) = c_4 + c_3 + c_0$, $Q(S_3) = Q(0,-1) = c_2 - c_1 + c_0$, and $Q(S_4) = Q(0,1) = c_2 + c_1 + c_0$, the coefficients c_0–c_4 are calculated as shown below:

$$\begin{aligned} c_0 &= Q(S_0) \\ c_1 &= (-Q(S_3) + Q(S_4)) \div 2 \\ c_2 &= (Q(S_3) + Q(S_4) - 2 \times Q(S_0)) \div 2 \\ c_3 &= (-Q(S_1) + Q(S_2)) \div 2 \\ c_4 &= (Q(S_1) + Q(S_2) - 2 \times Q(S_0)) \div 2 \end{aligned} \tag{2}$$

As shown in (3), the predictive minimum error cost is obtained by partially differentiating Q with respect to x and y, respectively. W ${}_xQ$ ${}_yQ = 0$, the x and y coordinates are regarded as the best prediction position (x_p, y_p). In the final step of the FME process, the best prediction position should be quantized by using the quantization operations [69]. The best quantized prediction position is determined as the best FMV.

$$\begin{aligned} \frac{\partial Q}{\partial x}(x,y) = 2c_4 x + c_3 = 0\,, \quad x_p = -\frac{c_3}{2c_4}\,, \quad (x_p = 0, \quad if\, c_4 = 0) \\ \frac{\partial Q}{\partial y}(x,y) = 2c_2 y + c_1 = 0\,, \quad y_p = -\frac{c_1}{2c_2}\,, \quad (y_p = 0, \quad if\, c_2 = 0) \end{aligned} \tag{3}$$

As explained in this section, the parabolic prediction based fractional-pixel search (PPFPS) has very low computational complexity, whereas the reconstruction quality is unsatisfactory compared with conventional FME methods. The quality performance of QPFPS, which is based on the above parabolic prediction, is more improved by applying an interpolation based FMV refinement process. In the final step of QPFPS, the best predicted position is located at the center of the small diamond search (SDS) pattern, and then a modified SDS algorithm is carried out to refine the best fractional-pixel search point. In this step case, however, the interpolation operations are required to provide a fractional-pixel search area. Accordingly, the total encoding time will be inevitably increased due to the complexity of the interpolation operations and the checking time of fractional-pixel search points. In this paper, therefore, a modified method of PPFPS is proposed to minimize the use of the interpolation operations and to reduce the number of search points; on the other hand, better performance than that of PPFPS will be maintained.

6.3. Observation

Fig. 6.2 shows an example of the difference between the simplified FME error surface of Fig. 6.1 (b) and the quadratic parabolic prediction model. The above-mentioned parabolic prediction model has perfectly bilateral symmetry by an axis but the real-world FME error surface is a little bit uneven. Hence, a solution to narrow the gap is required for enhancing the performance of the parabolic prediction model.

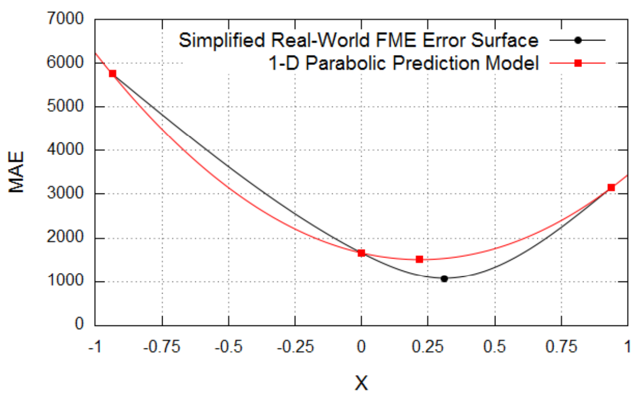


Fig. 6.2. The difference between the real-world FME error surface and the parabolic prediction model at 1/16-pixel MV resolution.

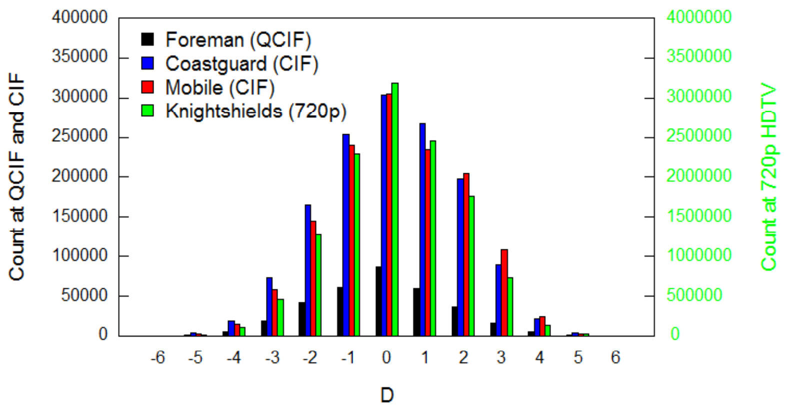


Fig. 6.3. The number of occurrences by the distance *D* between the best position of FFPS and that of PPFPS at 1/4-pixel MV resolution. *D* divided by 4 is represented as the fractional-pixel distance.

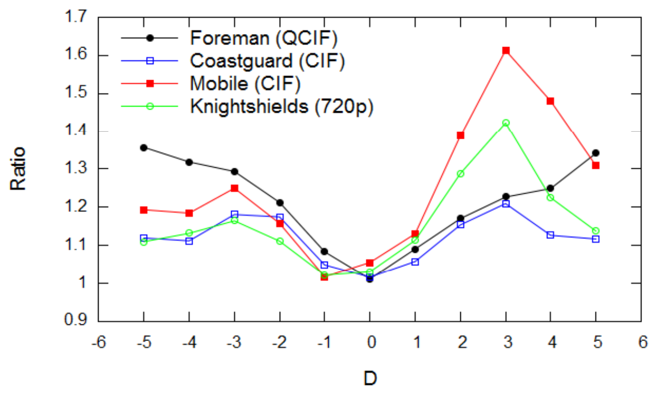


Fig. 6.4. The ratios P_1/P_2 and P_2/P_1 by the distance *D* between the best position of FFPS and that of PPFPS at 1/4-pixel MV resolution.

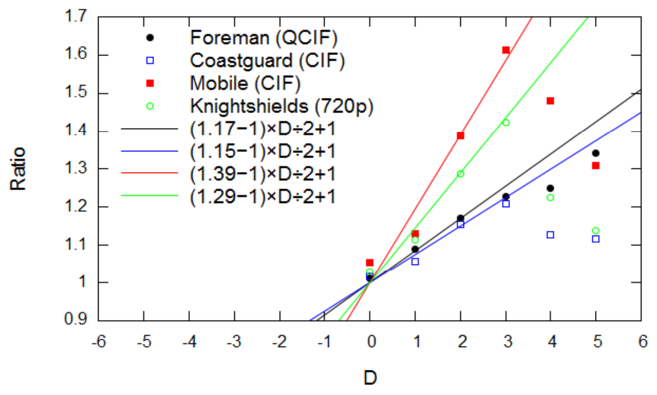


Fig. 6.5. The line fitting for the interval from 0 to 6 in Fig. 6.4.

A simulation based on the H.264/AVC JM version 12.4 reference software [75] using main profile has been tried for QCIF "Foreman," CIF "Coastguard," CIF "Mobile," and 720p HDTV "Knightshields" sequences, w
frames, respectively. The encoder parameters used in JM are configured as quantization parameter (QP) = 24, search range (SR) = 16, rate-distortion optimized mode (RDO) = 1, and symbol mode = 1 (CABAC). UMHexagonS implemented in JM is used for IME. The focus of the simulation is how the average IME error costs will change according to the distance D, which is converted to an integer value by multiplying by four and the maximum range is from -6 (= -3-3) to 6 (= 3-(-3)), between FFPS and PPFPS at quarter-pixel MV resolution. In this simulation, it is assumed that the matching performance of FFPS is the best of all FME algorithms. The three horizontal integer-pixel search points S_0, S_1, and S_2, which also represent the error costs corresponding to the IME positions, are used for an analysis. The average cost of S_i is represented as P_i. In an analysis, the average IME error cost P_1 by the distance D in the interval from -6 to -1 tends to be greater than P_2, whereas in the interval from 1 to 6, P_2 is greater than P_1. When the distance is zero, P_1 and P_2 have almost the same average cost. Fig. 6.3 shows that the number of occurrences of the distance D = zero is the highest, whereas that of the ranges [-6,-3] and [3,6] is very low. As shown in Fig. 6.4, particularly, it is noticeable that the higher the ratio of P_1/P_2 or P_2/P_1 is, the farther the distance tends to be. Based on this observation, the linear function L passing by both two points (0,1) and (2,C) is given as follows:

$$R_p = L(D) = \frac{C-1}{2} \times D + 1 \tag{4}$$

where C indicates the preset correction coefficient (preset corrector, PC), which is the ratio P_2/P_1 corresponding to the distance D = 2, the function L represents the predicted ratio S_1/S_2 or S_2/S_1 = R_p, and the variable D in the function L denotes the distance D. If the preset corrector C is known, the ratio S_1/S_2 or S_2/S_1 by the distance D can be approximated. On the other hand, the predicted distance D_p can be found by using the inverse of the function L. The inverse function of L is shown in (5).

$$D_p = L^{-1}(R) = \frac{2(R-1)}{C-1} \tag{5}$$

where R denotes the ratio $S_{\max}/S_{\min}$ ($S_{\max} = \max\{S_1,S_2\}$, $S_{\min} = \min\{S_1,S_2\}$) for the current macroblock. The predicted distance D_p depends on R and C. The predicted distance D_p must be divided by 4 at quarter-pixel MV resolution, and then D_p added to the best prediction position found by PPFPS is determined as the best corrected prediction position. The preset correctors for each test input sequence are listed in Table 6.1.

Table 6.1. The Preset Correction Coefficients for Test Input Sequences

Location	Horizontal search position (S_0, S_1, S_2)		Vertical search position (S_0, S_3, S_4)	
Distance	-2 ($C = P_1/P_2$)	2 ($C = P_2/P_1$)	-2 ($C = P_3/P_4$)	2 ($C = P_4/P_3$)
Foreman	1.21	1.17	1.02	1.01
Coastguard	1.17	1.15	1.01	1.00
Mobile	1.16	1.39	1.09	1.11
Knightshields	1.11	1.29	1.08	1.06

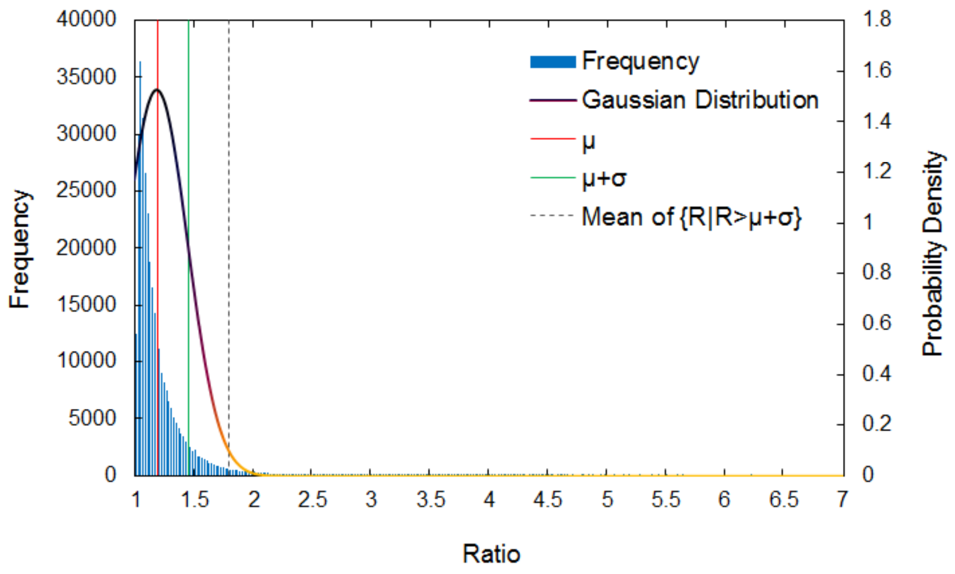


Fig. 6.6. The frequency of the ratio $R = S_{max}/S_{min}$ for the “Foreman” sequence.

Fig. 6.6 shows the frequency of the ratio R for the “Foreman” sequence. As statistical approach, Gaussian distribution has been compared with the frequency graph. In addition, mean and standard deviation of the data have analyzed for validity of the distance prediction function, as listed in Table 6.2. As a result, it is difficult to apply the function itself. That is, data transformation or modification of the function is needed.

Table 6.2. The Trend Analysis of the Ratio $R = S_{max}/S_{min}$ for Test Sequences

Sequence	Search position	Mean (μ)	S.D. (σ)	Minimum	Maximum	Mean of $\{R\|R>\mu+\sigma\}$	Percent of $\{R\|R>\mu+\sigma\}$
Foreman	Horizontal	1.19	0.26	1.00	6.99	1.80	9.65
	Vertical	1.21	0.31	1.00	8.21	1.95	9.50
Coastguard	Horizontal	1.16	0.28	1.00	8.30	1.89	6.50
	Vertical	1.23	0.33	1.00	26.74	1.99	8.37
Mobile	Horizontal	1.28	0.43	1.00	12.56	2.34	9.19
	Vertical	1.25	0.35	1.00	9.51	2.05	9.79
Knightshields	Horizontal	1.11	0.39	1.00	9.89	1.94	6.41
	Vertical	1.11	0.39	1.00	12.50	1.91	6.49

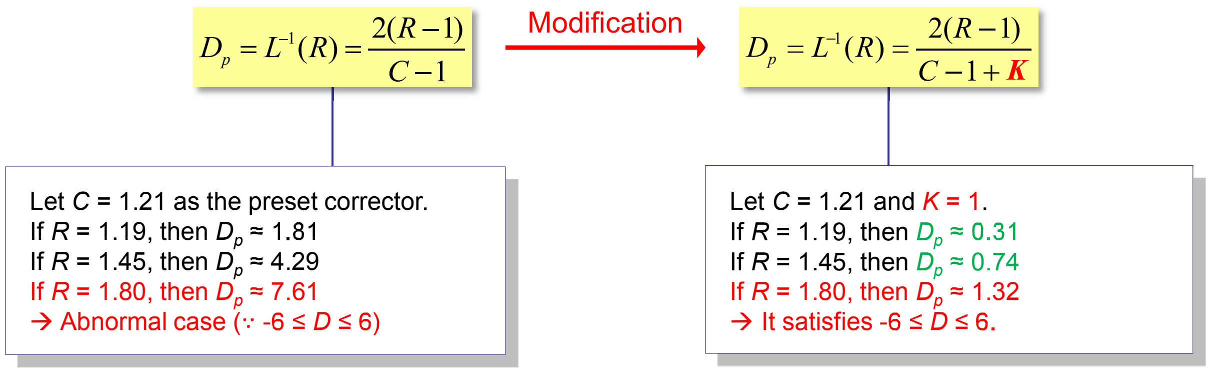

$D_p = L^{-1}(R) = \frac{2(R-1)}{C-1}$
Modification
$D_p = L^{-1}(R) = \frac{2(R-1)}{C-1+K}$
Let C = 1.21 as the preset corrector.
If R = 1.19, then D_p ≈ 1.81
If R = 1.45, then D_p ≈ 4.29
If R = 1.80, then D_p ≈ 7.61
→ Abnormal case (∵ -6 ≤ D ≤ 6)
Let C = 1.21 and K = 1.
If R = 1.19, then D_p ≈ 0.31
If R = 1.45, then D_p ≈ 0.74
If R = 1.80, then D_p ≈ 1.32
→ It satisfies -6 ≤ D ≤ 6.

Fig. 6.7. Modification of the distance prediction function.

The distance prediction function can be modified as shown in Fig. 6.7. The adjusting factor K can be added to the denominator of the linear function. By so doing, the data irregularity can be overcome. The modified distance prediction function will narrow somewhat the gap between the prediction position and the true motion vector. Table 6.3 represents the matching probability (%) between the predicted distance and the real distance.

Table 6.3. The Matching Probability (%) between the Predicted Distance and the Real Distance

Sequence	Search position	*K*								
		-2.0	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0
Foreman	Horizontal	26.45	26.35	26.04	22.21	17.69	25.65	26.47	26.52	26.43
	Vertical	27.43	27.83	27.98	25.03	02.50	24.68	27.25	27.32	27.07
Coastguard	Horizontal	21.92	21.79	21.05	17.04	14.34	21.26	22.05	22.06	22.03
	Vertical	30.21	30.56	30.51	26.78	01.12	26.58	30.47	30.81	30.52
Mobile	Horizontal	22.44	21.83	20.31	13.49	17.60	23.71	24.23	24.12	23.99
	Vertical	26.66	26.49	25.16	19.92	10.03	21.89	24.73	25.57	25.77
Knightshields	Horizontal	25.33	24.97	24.15	20.37	20.88	26.57	26.81	26.67	26.50
	Vertical	30.18	30.41	30.37	28.10	11.02	28.69	30.59	30.69	30.45

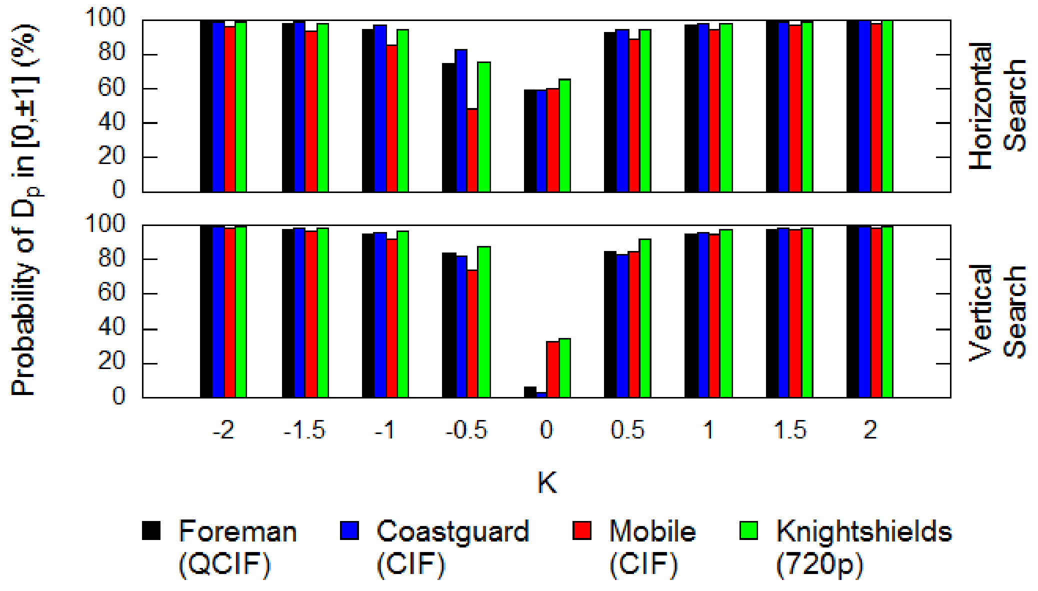


Fig. 6.8. Probability of D_p in the central range [0,±1].

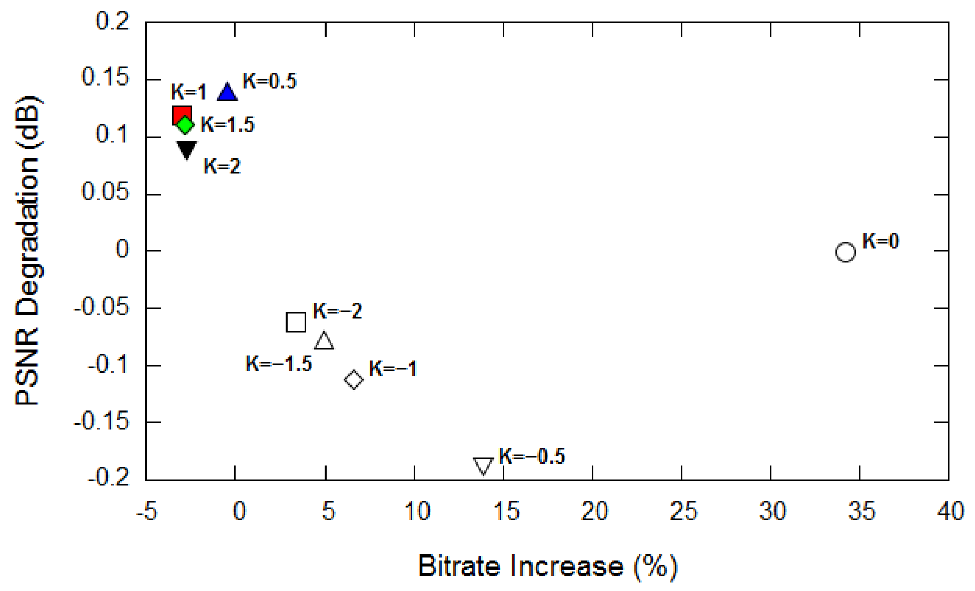


Fig. 6.9. PSNR and bitrate measurements by *K*.

Fig. 6.8 shows the probability of D_p in the central range [0,±1]. The PSNR and bitrate performances by K are represented in Fig. 6.9. The performance changes in the proposed technique by the adjusting factor $K \in [-2,2]$. Based on the simulations and analysis, the best adjusting factor K is set to 1. As shown in Fig. 6.8 and Fig. 6.9, when K is 0, the performance is the worst. Whereas when K equals 1, the bitrate is deceased about 3% with better PSNR.

6.4. Application of Proposed Technique

Two applications using the preset corrector technique are proposed. The first method is based on frame and the second method is based on macroblock type. Fig. 6.10 shows the illustration of the first method. It updates the preset corrector every n frame. Thus, this method must perform the full fractional-pixel search every n frame. In the other frames, the parabolic prediction is carried out, and then the predicted position is corrected by the distance prediction function.

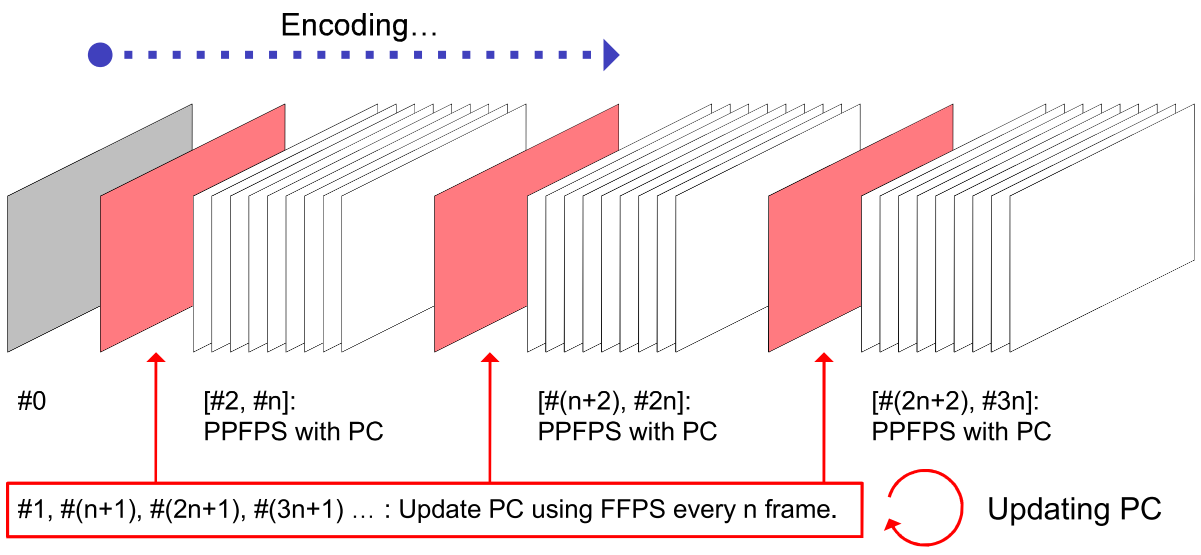


Fig. 6.10. The illustration of the available way to apply the proposed technique to specific frames.

Before explaining the second proposed method, the existing CBFPS algorithm is analyzed. As defined in H.264/AVC JM reference software, CBFPS is only performed when the current block type is submacroblock, as described in Fig. 6.11 and Fig. 6.12. Although this improves the performance, FFPS must be performed in the other block types 1, 2, and 3. It increases computational cost. Therefore, an alternative to FFPS is considered.

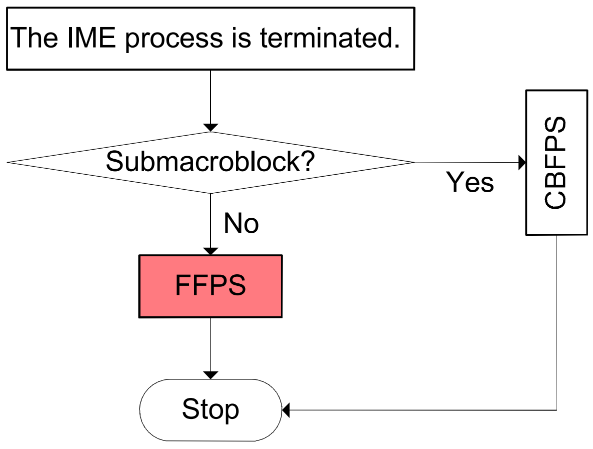


Fig. 6.11. The flowchart of CBFPS implemented in the H.264/AVC JM.

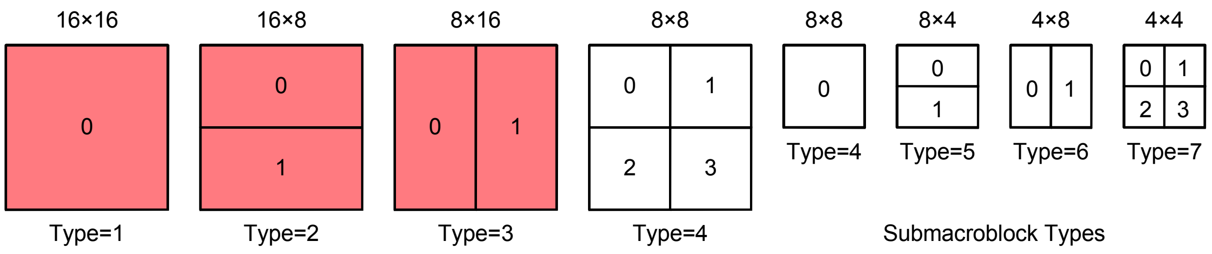


Fig. 6.12. The seven macroblock types used in H.264/AVC.

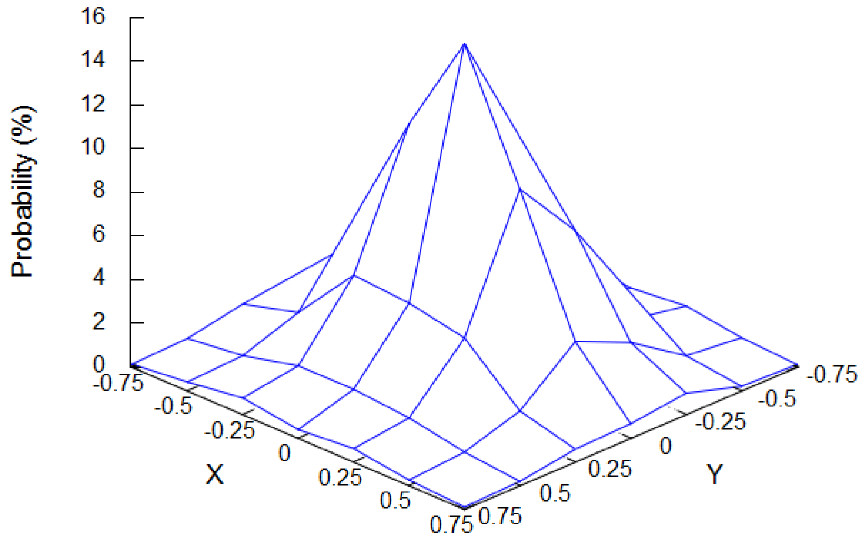


Fig. 6.13. Quarter-pixel motion vector probability distribution.

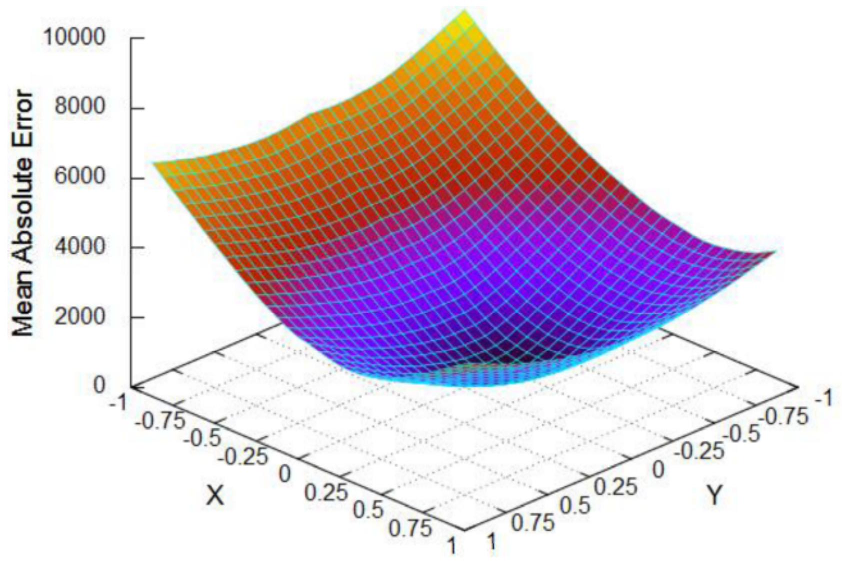
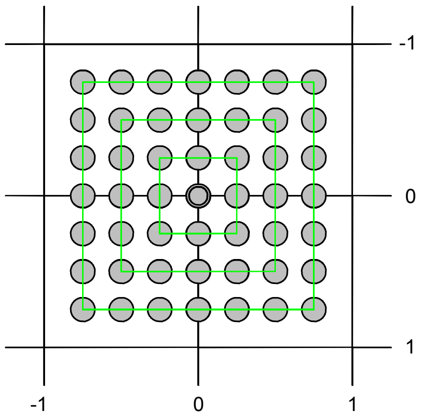


Fig. 6.14. Error surface of fractional-pixel motion estimation.

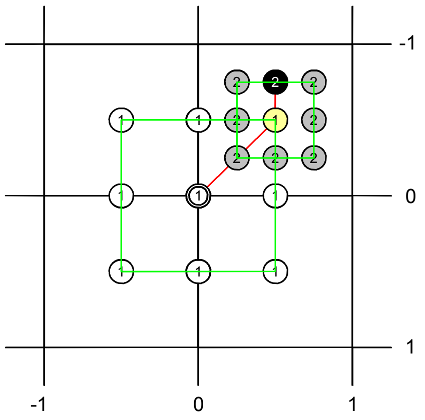
Fig. 6.13 shows the quarter-pixel MV probability distribution; Fig. 6.14 is the FME error surface. Here, the following two statements is worthy of notice.

(a) 66.71% of fractional-pixel MVs are found within the central 3 by 3 area.
(b) The FME error surface is clearly unimodal. There are few local minima.

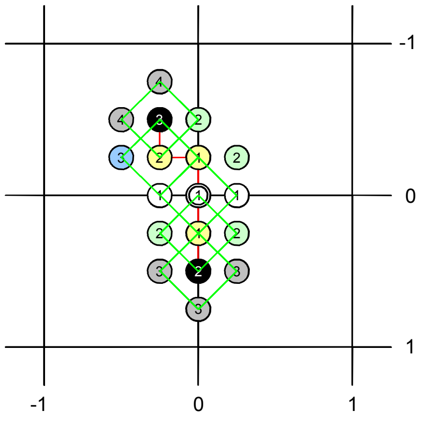
Based on the above observation, the block-based gradient descent search (BBGDS) [59], which is one of the most well-known fast search algorithms for integer-pixel block motion estimation, is introduced as an alternative to FFPS.



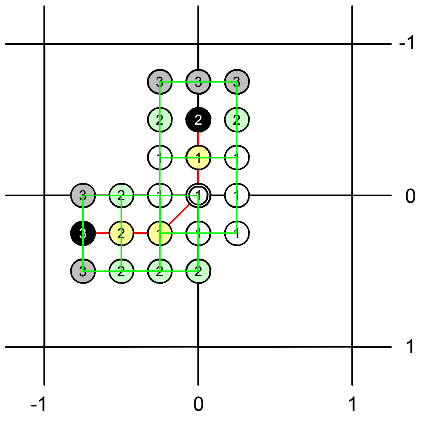
(a) EFFPS



(b) FFPS



(a) SDS



(b) BBGDS

Fig. 6.15. Examples of search process of (a) EFFPS, (b) FFPS, (c) SDS, and (d) BBGDS at 1/4-pixel MV resolution.

Table 6.4. The Quarter-Pixel Motion Vector Matching Probability, Average Search Points, and Speedup Rate

Sequence	Measurement	EFFPS	FFPS	SDS	BBGDS
Foreman	MV match (%)	100.00	86.78	74.57	89.13
	Search points	48.00	16.00	7.37	12.32
	Speedup	1.00	3.00	6.52	3.90
Coastguard	MV match (%)	100.00	93.20	85.64	93.84
	Search points	48.00	16.00	7.84	12.49
	Speedup	1.00	3.00	6.12	3.84
Mobile	MV match (%)	100.00	92.86	80.11	94.32
	Search points	48.00	16.00	7.88	12.98
	Speedup	1.00	3.00	6.09	3.70
Knightshields	MV match (%)	100.00	90.88	82.77	90.36
	Search points	48.00	16.00	7.41	12.03
	Speedup	1.00	3.00	6.48	3.99

To analyze the performance of each fractional-pixel search algorithm, the exhaustive full fractional-pixel search (EFFPS), as shown in Fig. 6.15 (a), is used as the anchor algorithm. EFFPS has not been adopted by the reference software due to its very heavy computational cost. However, the performance is better than FFPS. As shown in Table 6.4, BBGDS, which was originally developed for integer-pixel motion estimation but it is modified in order to be used as the quarter-pixel refinement process in this simulation, can produce more accurate fractional-pixel motion vectors while uses about four less search points. In other words, FFPS can be replaced with BBGDS without risk. On the other hand, the small diamond search (SDS) shows the worst performance results with the highest speedup rate.

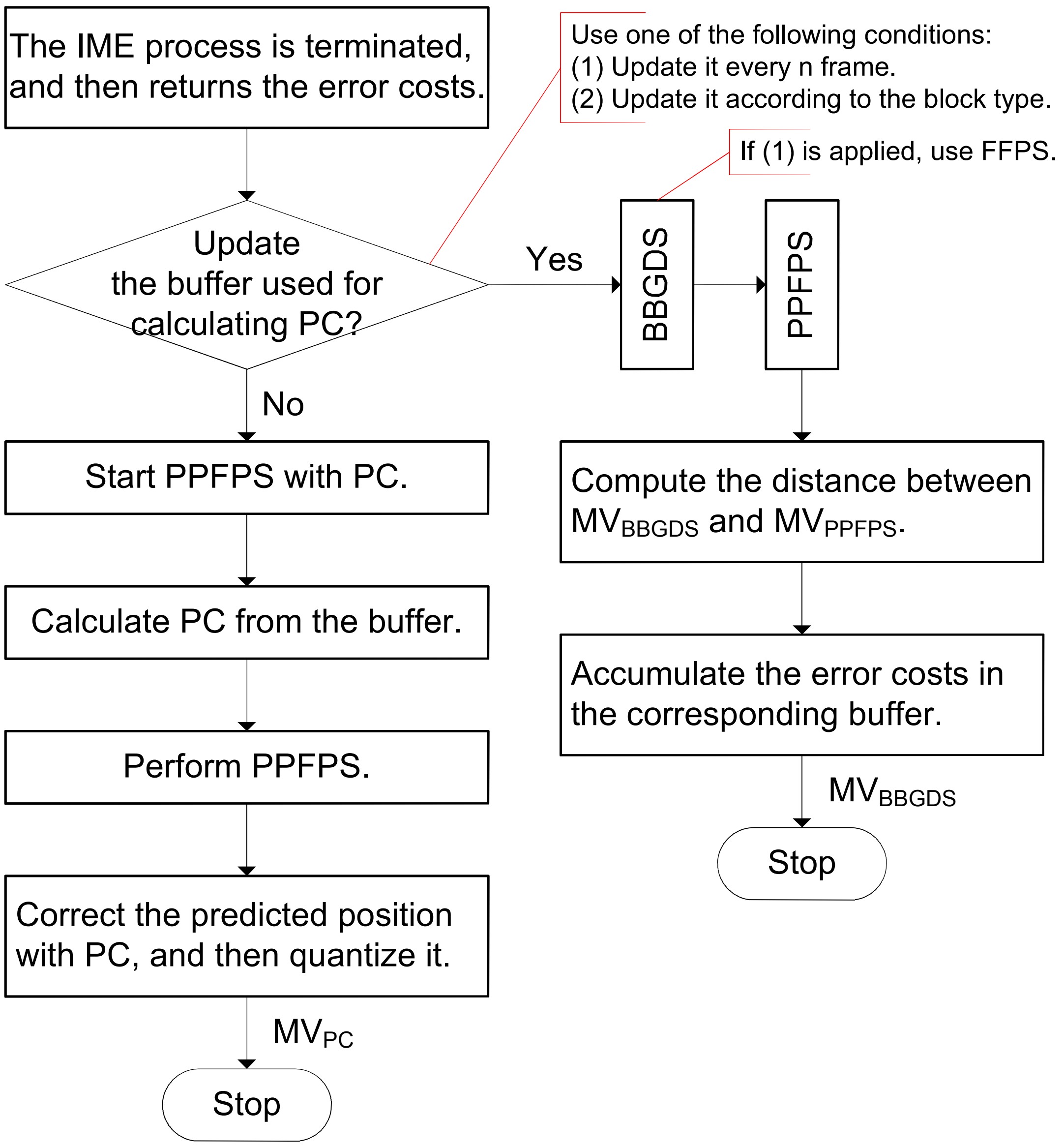
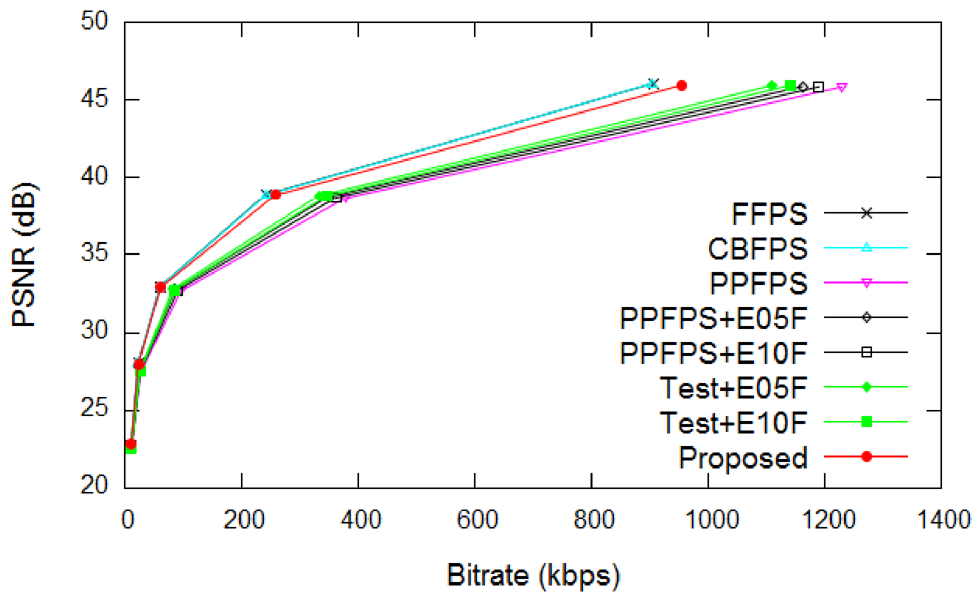

The IME process is terminated, and then returns the error costs.
Use one of the following conditions:
(1) Update it every n frame.
(2) Update it according to the block type.
If (1) is applied, use FFPS.
Update the buffer used for calculating PC?
Yes
BBGDS
PPFPS
No
Start PPFPS with PC.
Calculate PC from the buffer.
Perform PPFPS.
Correct the predicted position with PC, and then quantize it.
MVPC
Stop
Compute the distance between MVBBGDS and MVPPFPS.
Accumulate the error costs in the corresponding buffer.
MVBBGDS
Stop

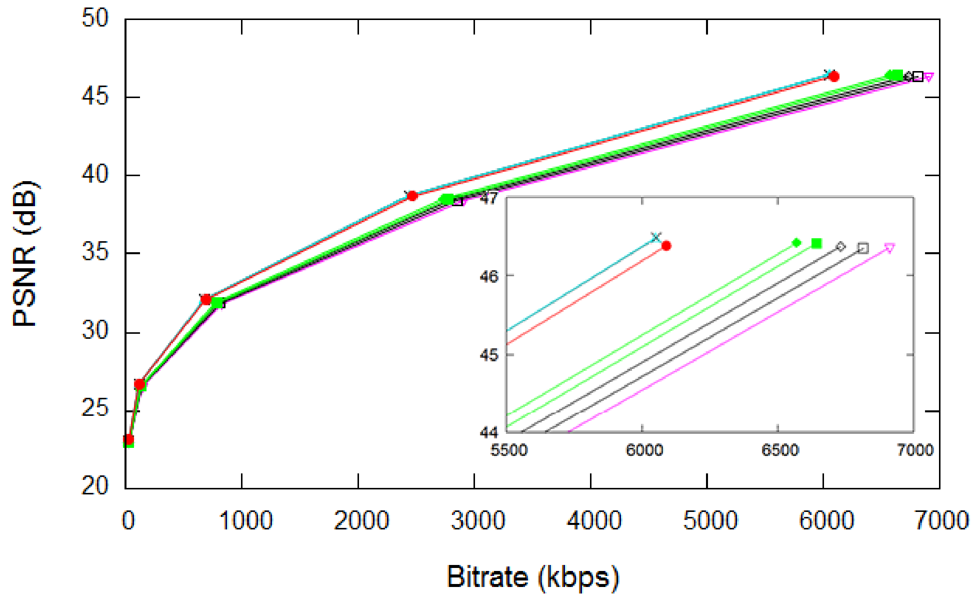
Fig. 6.16. The overall flow of the proposed FME algorithm.

Fig. 6.16 is the flowchart of two proposed methods. The first method uses FFPS and update preset corrector every n frame. The second method uses BBGDS and update preset corrector when block type is 1, 2, or 3. In the other process, they perform parabolic prediction based motion estimation with distance prediction technique.

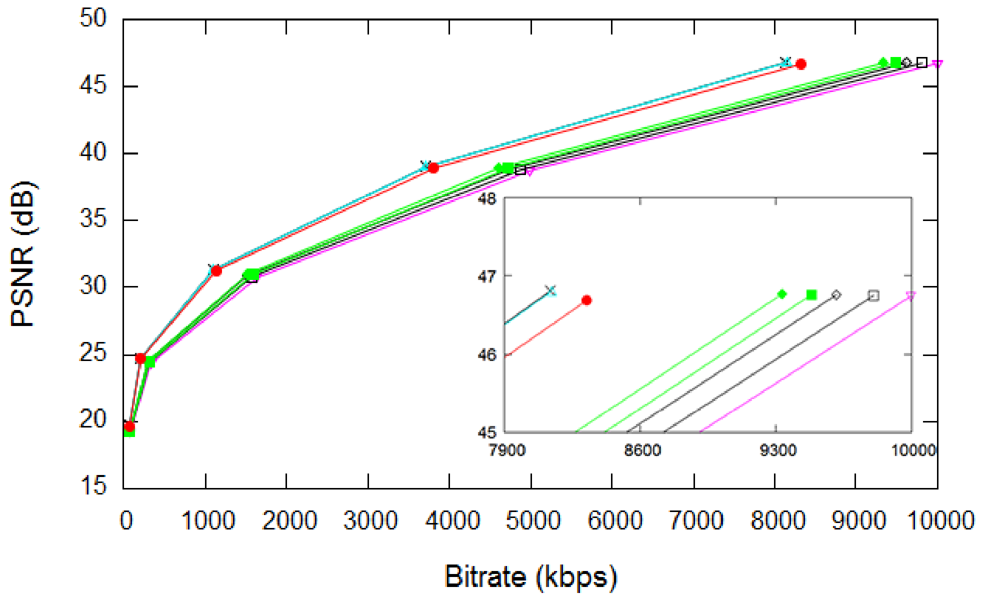
6.5. Experimental Results



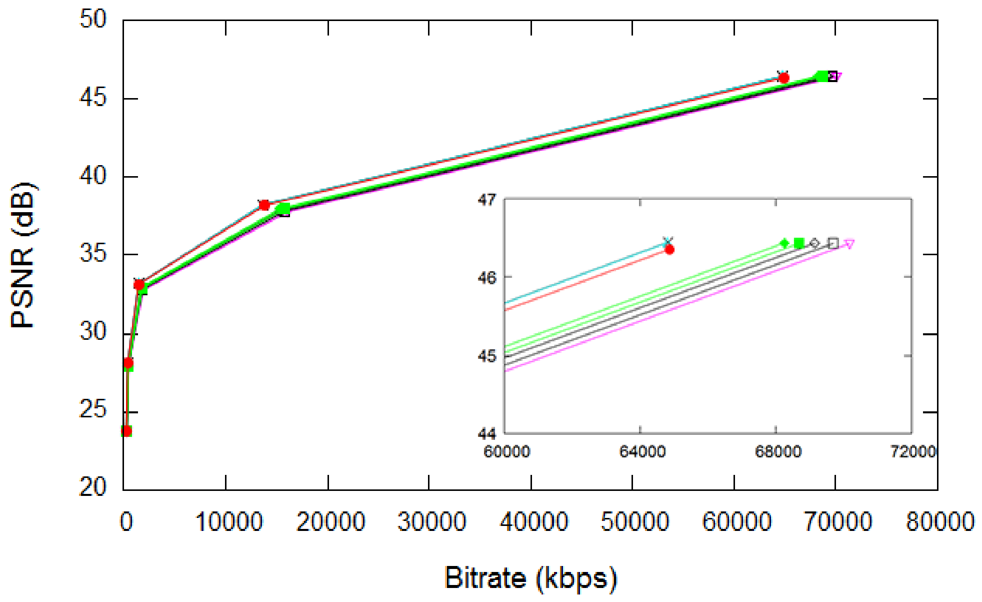
(a) “Foreman” (QCIF)



(b) “Coastguard” (CIF)



(c) “Mobile” (CIF)



(d) “Knightshields” (720p HDTV)

Fig. 6.17. Rate-distortion curves for the four test input sequences.

The proposed technique was implemented in the H.264/AVC JM reference encoder. “PPFPS+E05(10)F” represents the original PPFPS. It performs FFPS instead of PPFPS every 5 (10) frame. “Test+E05(10)F” is the modified PPFPS using the proposed preset corrector technique. It performs FFPS instead of PPFPS every 5 (10) frame. In proposed method, if the current block type < 4, it performs BBGDS. Otherwise, PPFPS with preset corrector is carried out. The adjusting factor *K* is set to 0.9.

Fig. 6.17 illustrates rate-distortion curves for the four popular test input sequences. The green line is the first proposed method based on frame. The led line is the second proposed method based on macroblock type. The performance of the second proposed method is always better compared with original parabolic prediction. It also represents validity of the distance prediction technique using preset corrector. However, the bitrate is increased much more than FFPS and CBFPS. On the other hand, the performance of the red line’s second method shows a significant improvement. The average bitrate increase is 1.92% with respect to FFPS. The PSNR drop is 0.04.

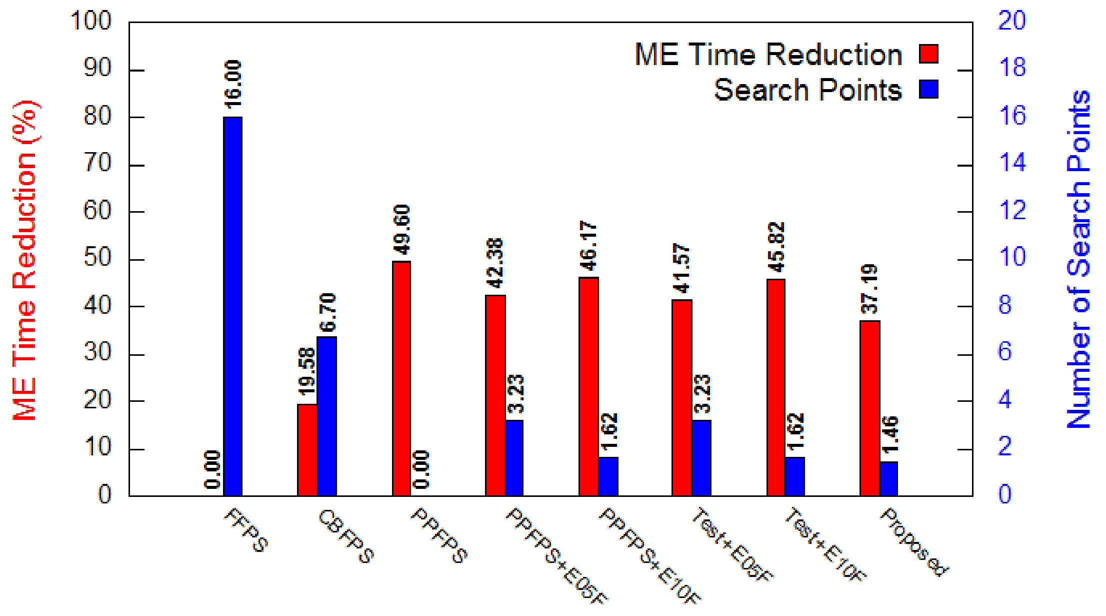


Fig. 6.18. The average ME time reduction and number of search points.

In Fig. 6.18, the graph shows computational complexity comparison. For your reference, the average number of search points is not necessarily proportional to complexity. Although the number of large macroblocks is much less than that of small ones, the larger macroblock size is, the more increased the number of pixels used in matching criteria is. Anyway, the ME time reduction of the second proposed method is about 37.19% with respect to FFPS. With respect to CBFPS, it shows 21.39% reduction. Normal parabolic prediction achieves about 50% motion estimation time reduction compared with FFPS but the rate-distortion performance is the worst.

6.6. Summary

The fractional-pixel ME module of the recent video encoder definitely includes the interpolation process in order to form fractional-pixel search samples from neighboring inter-pixel information, while greatly increasing its computational complexity. The 1-D parabolic prediction model can approximate the FME search area at very low computational cost, whereas its prediction performance is not enough for practical use; the parabolic model has perfectly bilateral symmetry by an axis, but the real-world error surface is not actually symmetrical. In this chapter, therefore, the modified parabolic prediction technique using preset correction coefficients has been proposed to enhance its performance. A linear model used in the proposed technique is presented to narrow the gap between the real-world FME error surface and the 1-D parabolic prediction model. The best prediction position obtained by the 1-D parabolic prediction can be corrected by applying the proposed technique. In the simulation results, the proposed method has a competitive R-D performance and lower computational complexity compared with the existing interpolation based search algorithms. Furthermore, the performance has certain potential of improvement by customizing parameters. Thus, it needs to design more robust algorithm based on in-depth data statistical analysis.

Chapter 7
CONCLUSIONS

The recently released H.265/HEVC standard will achieve a much higher efficient compression performance for high-resolution video formats beyond HDTV, as compared to H.264/AVC. Owing to the significantly increased computational complexity, however, additional costs will be incurred because the real-time encoder is implemented on consumer electronics devices, particularly mobile devices. In this thesis, therefore, the proposed techniques focus on reducing the computational complexity in video encoding.

In particular, the integer-pixel motion estimation module of a video encoder has the highest computational cost. Thus, first of all, low complexity integer-pixel motion estimation techniques were proposed in the thesis. The proposed hybrid motion estimation algorithm is a combination of revised diamond search algorithm, full diamond and dodecagon search patterns, and efficient stationary block skip method. The proposed integer-pixel motion estimation algorithm was successfully implemented in the H.264/AVC reference encoder software. The experimental results show that the proposed algorithm further decreases the computational burden while maintaining negligible rate-distortion performance degradation with respect to the existing reference method; the motion estimation time of proposed algorithm can be reduced by up to 12% compared with the existing reference method.

On the other hand, fractional-pixel motion estimation causes a remarkable increase in the overall encoding time, as the integer-pixel motion estimation is performed, followed by it. Nevertheless, fractional-pixel motion estimation has a strong impact on quality performance, raising about 1 to 3 dB in terms of peak signal-to-noise ratio. For this reason, the complexity optimization of fractional-pixel motion estimation is significantly needed. Apart from integer-pixel motion estimation, fractional-pixel motion estimation requires interpolation operations to generate a fractional-pixel search area. In the thesis, there are several proposed fractional-pixel motion estimation algorithms, the vertically symmetrical linear model based fractional-pixel motion estimation, the interpolation-free fractional-pixel motion estimation based on data trend approximation, and the enhanced 1-D parabolic prediction based fractional-pixel motion estimation. All the proposed fractional-pixel motion estimation algorithms were designed to archive utmost reduction in computational complexity without using upsampling or with using a minimal number of fractional-pixel search points. Though most of the existing fractional-pixel motion estimation methods are using fractional-pixel motion vector refinement for improvement in quality, the proposed ones in this thesis are contributed to substantially reduce the use of block distortion measurement and interpolation operations which are directly involved in computational complexity. In the results of the simulations based on the H.265/HEVC reference software encoder, the interpolation-free fractional-pixel motion estimation based on data trend approximation, the quadratic Bézier spline based method, can achieve about 41% fractional-pixel motion estimation time saving on average — when the interpolation process, required to efficiently determine coding unit (CU) depth and prediction unit (PU) type, is omitted, about 99% fractional-pixel ME time saving — with respect to the conventional full fractional-pixel search algorithm; its computational complexity is almost the same as the existing 1-D parabolic prediction based fractional-pixel search algorithm but it has better rate-distortion performance. Moreover, the proposed prediction technique using preset correction coefficients may provide a clue for what mathematical model prediction-based fractional-pixel motion estimation algorithms are further improved in quality at minimal computational cost. The proposed algorithms can also be combined with each other to further improve performance. This research work will continue far into the future also. I would like to dedicate myself to developing much higher efficient algorithms in this research field.

REFERENCES

[1] *Parameter Values for Ultra-High Definition Television Systems for Production and International Programme Exchange*, ITU-R Rec. BT.2020-1, ITU-R, Jun. 2014.

[2] M. Sugawara, S.-Y. Choi, and D. Wood, "Ultra-high-definition television (Rec. ITU-R BT.2020): A generational leap in the evolution of television [Standards in a nutshell]," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 170–174, May 2014.

[3] M. Sugawara, M. Kanazawa, K. Mitani, H. Shimamoto, T. Yamashita, and F. Okano, "Ultrahigh-definition video system with 4000 scanning lines," *SMPTE Mot. Imag. J.*, vol. 112, no. 10–11, pp. 339–346, Oct. 2003.

[4] *Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s—Part 2: Video*, ISO/IEC JTC1/SC29/WG11, Ref. No. ISO/IEC 11172-2:1993(E), 1993.

[5] *Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, ITU Rec. H.262, ISO/IEC JTC1/SC29/WG11, Ref. No. ISO/IEC 13818-2:1994(E), 1994.

[6] *Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC JTC1/SC29/WG11, Ref. No. ISO/IEC 14496-2:2004(E), Edition 3, 2004.

[7] *Video Codec for Audiovisual Services at p×64 kbit/s*, ITU-T Rec. H.261, ITU-T SG15, Edition 2, 1993.

[8] *Video Coding for Low Bit Rate Communication*, ITU-T Rec. H.263, ITU-T SG16, Edition 3, 2005.

[9] K. R. Rao and J. J. Hwang, *Techniques and Standards for Image, Video and Audio Coding*, Englewood Cliffs, NJ: Prentice Hall, 1996.

[10] T. Wiegand, G. J. Sullivan, and A. Luthra, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)*, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, document JVT-G050r1, 8th Meeting, Geneva, Switzerland, May 2003.

[11] T. W . J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/A *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[12] T. K. Tan, M. Mrak, V. Baroncini, and N. Ramzan, *Report on HEVC Compression Performance Verification Testing*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, document JCTVC-Q1011, 17th Meeting, Valencia, ES, 27 Mar. – 4 Apr. 2014.

[13] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, Y.-K. Wang, and T. Wiegand, *High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS & Last Call)*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, document JCTVC-L1003, 12th Meeting, Geneva, CH, Jan. 2013.

[14] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. W (HEVC) standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[15] F. Bossen, B. Bross, K. Sühring, and D. Flynn, “HEVC complexity and implementation analysis,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec. 2012.

[16] K. McCann, W.-J. Han, I.-K. Kim, J.-H. Min, E. Alshina, A. Alshin, *et al.*, *S ’s Response to the Call for Proposals on Video Compression Technology*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, document JCTVC-A124, 1st Meeting, Dresden, DE, Apr. 2010.

[17] I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley & Sons, West Sussex, England, 2003.

[18] Y.-W. Huang, B.-Y. Hsieh, S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, “Analysis and complexity reduction of multiple reference frames motion estimation in H.264/A *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 507–522, Apr. 2006.

[19] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic Publishers, Boston, Dordrecht, London, 1999.

[20] A. N. Netravali and J. D. Robbins, “Motion compensated television coding: Part I,” *The Bell System Technical Journal*, vol. 58, No. 3, pp. 631–670, Mar. 1979.

[21] D. R. Walker and K. R. RAO, “Improved pel-recursive motion estimation,” *IEEE Trans. Commun.*, vol. 32, no. 10, pp. 1128–1134, Oct. 1984.

[22] V. V. Estrela and N. P. Galatsanos, “Spatially-adaptive regularized pel-recursive motion estimation based on cross-validation,” in *Proc. IEEE Int. Conf. on Image Process.*, vol. 2, pp. 200–203, Oct. 1998.

[23] H. Gharavi and H. Reza-Alikhani, “Pel-recursive motion estimation algorithm,” *IET Electronics Letters*, vol. 37, no. 21, pp. 1285–1286, Oct. 2001.

[24] J. Jain and A. Jain, “Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1799–1808, Dec. 1981.

[25] X. Tian, T. M. Le, and Y. Lian, *Entropy Coders of the H.264/AVC Standard: Algorithms and VLSI Architectures*, Springer Berlin Heidelberg, 2011.

[26] B. Pesquet-Popescu, M. Cagnazzo, and F. Dufaux, “Motion estimation—A video coding viewpoint,” in *Academic Press Library in Signal Processing: Image and Video Compression and Multimedia*, Vol. 5, R. Chellappa and S. Theodoridis, Ed., Elsevier Science, Oxford, England, pp. 27–92, 2014.

[27] S. Wang and H. Chen, “An improve algorithm of motion compensation MPEG video compression,” in *Proc. IEEE Int. Conf. on Veh. Electron.*, vol. 1, pp. 261–264, Sep. 1999.

[28] R. Purwar, N. Prakash, and N. Rajpal, “A matching criterion for motion compensation in the temporal coding of video signal,” *Signal, Image and Video Processing (SIViP)*, vol. 5, no. 2, pp. 133–139, Jun. 2011.

[29] G. J. Sullivan and T. Wiegand, “Rate-distortion optimization for video compression,” *IEEE Signal Process. Mag.*, vol. 15, pp. 74–90, Nov. 1998.

[30] B. Seo, X. Liu, and R. Zimmermann, “HD Video Remote Collaboration Application,” in *The Handbook of MPEG Applications: Standards in Practice*, M. C. Angelides and H. Agius, Ed., John Wiley & Sons, West Sussex, England, pp. 33–58, 2011.

[31] G. Bjøntegaard, *Calculation of Average PSNR Differences between RD Curves*, ITU-T SG16 Q.6, document VCEG-M33, 13th Meeting, Austin, TX, Apr. 2001.

[32] T. Komarek and P. Pirsch, “Array architectures for block matching algorithms,” *Circuits and Systems, IEEE Trans. Circuits Syst.*, vol. 36, no. 10, pp. 1301–1308, Oct. 1989.

[33] C.-H. Hsieh and T.-P. Lin, “VLSI architecture for block-matching motion estimation algorithm,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 2, pp. 169–175, Jun. 1992.

[34] H. Yee and Y. H. Hu, “A novel modular systolic array architecture for full-search block matching motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 5, pp. 407–416, Oct. 1995.

[35] S. M. Akramullah, I. Ahmad, and M.-L. Liou, “Optimization of H.263 video encoding using a single processor computer: performance tradeoffs and benchmarking,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 8, pp. 901–915, Aug. 2001.

[36] *ITU-T Recommendation H.263 Software Implementation*, Digital Video Coding Group, Telenor R&D, 1995.

[37] S. Eckart and C. E. Fogg, “ISO/IEC MPEG-2 software video codec,” in *Proc. SPIE*, vol. 2419, pp. 100–109, Apr. 1995.

[38] A. M. Tourapis, H.-Y. Cheong, and P. Topiwala, *Fast ME in the JM Reference Software*, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, document JVT-P026, 16th Meeting, Poznań, PL, Jul. 2005.

[39] C.-D. Bei and R. M. Gray, “An improvement of the minimum distortion encoding algorithm for vector quantization,” *IEEE Trans. Commun.*, vol. 33, no. 10, pp. 1132–1133, Oct. 1985.

[40] C.-K. Cheung and L.-M. Po, “Normalized partial distortion search algorithm for block motion estimation”, *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, pp. 417–422, Apr. 2000.

[41] C.-K. Cheung and L.-M. Po, “Adjustable partial distortion search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 1, pp. 100–110, Jan. 2003.

[42] B. Montrucchio and D. Quaglia, “New sorting-based lossless motion estimation algorithms and a partial distortion elimination performance analysis,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 2, pp. 210–220, Feb. 2005.

[43] M. G. Sarwer and Q. M. Jonathan Wu, “Efficient two step edge based partial distortion search for fast block motion estimation,” *IEEE Trans. Consumer Electron.*, vol. 55, no. 4, pp. 2154–2162, Nov. 2009.

[44] W. Li and E. Salari, “Successive elimination algorithm for motion estimation,” *IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105–107, Jan. 1995.

[45] M. I. Voĭtsekhovskiĭ, “Minkowski inequality,” in *Encyclopaedia of Mathematics: Heaps and Semi-Heaps—Moments, Method of (in Probability Theory)*, Vol. 3, M. Hazewinkel, Ed., Springer Science+Business Media Dordrecht, pp. 903–904, 1995.

[46] X. Q. Gao, C. J. Duanmu, and C. R. Zou, “A multilevel successive elimination algorithm for block matching motion estimation,” *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 501–504, Mar. 2000.

[47] L.-C. Chang, K.-L. Chung, and T.-C. Yang, “An improved search algorithm for motion estimation using adaptive search order,” *IEEE Signal Process. Lett.*, vol. 8, no. 5, pp. 129–130, May 2001.

[48] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” in *Proc. Nat. Telecommun. Conf.*, pp. G5.3.1–G5.3.5, New Orleans, LA, Nov. 1981.

[49] J.-N. Kim and T.-S. Choi, “A fast three-step search algorithm with minimum checking points using unimodal error surface assumption,” *IEEE Trans. Consumer Electron.*, vol. 44, no. 3, pp. 638–648, Aug. 1998.

[50] A. M. Tourapis, O. C. A . L. Liou, “Predictive motion vector field

adaptive search technique (PMVFAST)—Enhancing block based motion estimation", in *Proc. Visual Communications and Image Processing (VCIP)*, pp. 883–892, Jan. 2001.

[51] X.-Q. Banh and Y.-P. Tan, "Adaptive dual-cross search algorithm for block-matching motion estimation," *IEEE Trans. Consumer Electron.*, vol. 50, no. 2, pp. 766–775, May 2004.

[52] K. M. Uz, M. Vetterli, and D. J. LeGall, "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 86–99, Mar. 1991.

[53] Y.Q. Shi and X. Xia, "A thresholding multiresolution block matching algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 437–440, Apr. 1997.

[54] J. Chalidabhongse and C.-C. J. K   , "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 477–488, Jun. 1997.

[55] J.-S. Kim and R.-H. Park, "A fast feature-based block matching algorithm using integral projections," *IEEE Journal on Selected Areas in Commun.*, vol. 10, no. 5, pp. 968–971, Jun. 1992.

[56] Y.-C. Lin and S.-C. Tai, "Fast full-search block-matching algorithm for motion-compensated video compression," *IEEE Trans. Commun.*, vol. 45, no. 5, pp. 527–531, May 1997.

[57] R. Li, B. Zeng, and M.-L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.

[58] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3,

pp. 313–317, Jun. 1996.

[59] L.-K. Liu and E. Feig, “A block-based gradient descent search algorithm for block motion estimation in video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 419–422, Aug. 1996.

[60] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block matching motion estimation,” *IEEE Trans. Image Process.*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[61] C. Zhu, X. Lin, and L.-P. Chau, “Hexagon-based search pattern for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 349–355, May 2002.

[62] C.-H. Cheung and L.-M. Po, “A novel cross-diamond search algorithm for fast block motion estimation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1168–1177, Dec. 2002.

[63] X. Jing and L.-P. Chau, “An efficient three-step search algorithm for block motion estimation,” *IEEE Trans. Multimedia.*, vol. 6, no. 3, pp. 435–438, Jun. 2004.

[64] C.-H. Cheung and L.-M. Po, “Novel cross-diamond-hexagonal search algorithms for fast block motion estimation,” *IEEE Trans. Multimedia.*, vol. 7, no. 1, pp. 16–22, Feb. 2005.

[65] Z. Chen, J. X , Y. He, and J. Zheng, “Fast integer-pel and fractional-pel motion estimation for H.264/A ,” *J. Vis. Commun. Image R.*, vol. 17, pp. 264–290, Apr. 2006.

[66] *Reference Software for ITU-T H.264 Advanced Video Coding*, ITU-T Rec. H.264.2, ITU-T and ISO/IEC JTC1, Edition 1: Mar. 2005, Edition 2: Jun. 2008, Edition 3: Jun. 2010, Edition 4: Jan. 2012.

[67] F. Bossen, D. Flynn, and K. Sühring, *High Efficiency Video Coding Test Moel 12 (HM 12) Reference Software*, Joint Collaborative Team on Video

Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, document JCTVC-N1010, 14th Meeting, Vienna, AT, Jul. 2013.

[68] J. W. Suh and J. Jeong, “Fast sub-pixel motion estimation techniques having lower computational complexity,” *IEEE Trans. Consumer Electron.*, vol. 50, pp. 968–973, Aug. 2004.

[69] J.-F. Chang and J.-J. Leou, “A quadratic prediction based fractional-pixel motion estimation algorithm for H.264,” *J. Vis. Commun. Image R.*, vol. 17, pp. 1074–1089, Oct. 2006.

[70] M. Sayed, W. Badawy, and G. Jullien, “Low-complexity algorithm for fractional-pixel motion estimation,” in *Proc. IEEE Int. Conf. on Image Process.*, pp. 1565–1568, Nov. 2009.

[71] S. Dikbas, T. Arici, and Y. Altunbasak, “Fast motion estimation with interpolation-free sub-sample accuracy,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 7, pp. 1047–1051, Jul. 2010.

[72] L. Yang, K. Yu, L. Jiang, and L. Shipeng, “Prediction-based directional fractional pixel motion estimation for H.264 video coding,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Process.*, vol. 2, pp. II/901–II/904, Mar. 2005.

[73] J. S. Kim, K. W. Lee, and M. H. Sunwoo, “Novel fractional pixel motion estimation algorithm using motion prediction and fast search pattern,” in *Proc. IEEE Int. Conf. on Multimedia and Expo*, pp. 821–824, Jun. 2008.

[74] H. Nisar and T.-S. Choi, “Fast and efficient fractional pixel motion estimation for H.264/AVC video coding,” in *Proc. IEEE Int. Conf. on Image Process.*, pp. 1561–1564, Nov. 2009.

[75] A. M. Tourapis, A. Leontaris, K. Sühring, and G. J. Sullivan, *H.264/MPEG-4 AVC Reference Software Manual*, ISO/IEC JTC1/SC29/WG11 and ITU-T

SG16 Q.6, document JVT-X072, 24th Meeting, Geneva, CH, Jun. 2007.

[76] S. Hiratsuka, S. Goto, T. Baba, and T. Ikenaga, "A locally adaptive subsampling algorithm for software based motion estimation," in *Proc. IEEE Int. Symp. on Circuits Syst.*, vol. 3, pp. 2891–2894, May 2005.

[77] K. Sühring, *et al.*, *H.264/AVC Key Technical Area (KTA) Reference Software*, [Online]. Available: http://iphome.hhi.de/suehring/tml/

[78] C.-U. Jeong, J.-K. Choi, T. Ikenaga, and S. Goto, "A diamond web-grid search algorithm combined with efficient stationary block skip method for H.264/AVC motion estimation," *Journal of Korean Society for Internet Information*, vol. 11, no. 2, pp. 49–60, Apr. 2010.

[79] C.-U. Jeong and H. Watanabe, "A linear model-based sub-pixel motion estimation method for H.264/AVC standard," *Workshop on Picture Coding Symposium*, pp. 11–12, Dec. 2010.

[80] C.-U. Jeong and H. Watanabe, "A vertically symmetrical linear model-based fractional-pixel motion estimation algorithm for H.264/AVC encoder," *The Special Interest Group of Information Processing Society of Japan (IPSJ SIG) Technical Reports*, vol. 2010-AVM71, no. 20, pp. 1–5, Dec. 2010.

[81] C.-U. Jeong and H. Watanabe, "Interpolation-free fractional pixel motion estimation based on data trend approximation," in *Proc. Int. Conf. on Signals and Electronic Systems (ICSES)*, pp. 1–6, Sep. 2012.

[82] C.-U. Jeong and H. Watanabe, "Interpolation-free fractional pixel motion estimation based on data trend approximation," *GITS/GITI Research Bulletin 2013-2014*, pp. 52–62, Oct. 2014.

[83] C.-U. Jeong and H. Watanabe, "A modified parabolic prediction based fractional pixel motion estimation using preset corrector," in *Proc. Asia-Pacific Conf. on Commun. (APCC)*, pp. 526–527, Oct. 2012.

LIST OF ACADEMIC ACHIEVEMENTS

Journals

[1] C.-U. Jeong and H. Watanabe, "Interpolation-free fractional pixel motion estimation based on data trend approximation," *GITS/GITI Research Bulletin 2013-2014*, pp. 52–62, Oct. 2014.

[2] C.-U. Jeong, J.-K. Choi, T. Ikenaga, and S. Goto, "A diamond web-grid search algorithm combined with efficient stationary block skip method for H.264/AVC motion estimation," *Journal of Korean Society for Internet Information*, vol. 11, no. 2, pp. 49–60, Apr. 2010.

International Conferences

[1] C.-U. Jeong and H. Watanabe, "A modified parabolic prediction based fractional pixel motion estimation using preset corrector," in *Proc. Asia-Pacific Conf. on Commun. (APCC)*, pp. 526–527, Oct. 2012.

[2] C.-U. Jeong and H. Watanabe, "Interpolation-free fractional pixel motion estimation based on data trend approximation," in *Proc. Int. Conf. on Signals and Electronic Systems (ICSES)*, pp. 1–6, Sep. 2012.

International Workshops

[1] C.-U. Jeong and H. Watanabe, "A linear model-based sub-pixel motion estimation method for H.264/AVC standard," *Workshop on Picture Coding Symposium*, pp. 11–12, Dec. 2010.

Domestic Technical Reports

[1] C.-U. Jeong and H. Watanabe, “Fractional-pixel position prediction techniques based on Bézier curve for H.264/AVC encoder,” *The Special Interest Group of Information Processing Society of Japan (IPSJ SIG) Technical Reports*, vol. 2011-AVM73, no. 8, pp. 1–6, Jul. 2011.

[2] C.-U. Jeong and H. Watanabe, “A vertically symmetrical linear model-based fractional-pixel motion estimation algorithm for H.264/AVC encoder,” *The Special Interest Group of Information Processing Society of Japan (IPSJ SIG) Technical Reports*, vol. 2010-AVM71, no. 20, pp. 1–5, Dec. 2010.