Distortion measure analysis for efficient block matching and face region based video coding

Xiong, Bing

2009

Xiong, B. (2009). Distortion measure analysis for efficient block matching and face region based video coding. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/14800>

<https://doi.org/10.32657/10356/14800>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**DISTORTION MEASURE ANALYSIS FOR EFFICIENT
BLOCK MATCHING AND FACE REGION BASED VIDEO
CODING**

XIONG BING

**XIONG BING
SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING
2009**

**DISTORTION MEASURE ANALYSIS FOR EFFICIENT BLOCK
MATCHING AND FACE REGION BASED VIDEO CODING**

2009

DISTORTION MEASURE ANALYSIS FOR EFFICIENT BLOCK MATCHING AND FACE REGION BASED VIDEO CODING

XIONG BING

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

Acknowledgement

Studying for this PhD degree turned out to be the biggest project I have undertaken up to my current life. It took much longer and more exhausting time than I expected, especially during the revision stage when I had to work on it on a part time basis while supporting myself by full time jobs. Even though, I could not have reached this stage without the help and support from many people, to whom I owe a lot of “thank you”.

The greatest appreciation goes to Dr. Zhu Ce, who has helped to greatly improve the quality of the research and the content of the revised thesis as my supervisor for the revision stage. He has discussed with me about the work inside this thesis beyond the midnight quite a few times. With his strict standards and earnest help, the revised thesis is almost a re-written one.

I have to thank the three examiners of my thesis for their valuable and constructive comments and suggestions, which have helped to improve the quality of thesis a lot.

I also want to thank my previous supervisor, Dr. Charayaphan Cheroensak. It is he who introduces me into the computer vision field. Without his encouragement and guidance, I would not have completed the work in this thesis. His strict and assiduous attitude for work made my learning experience with him memorable. He also introduced abundant cutting-edge hardware technologies to me.

I want to say thank you to Yu Wei, Liu Minglei and Zhang Yixuan. We had valuable and inspiring discussions on the project and related material. I also thank them for their help in life and study. I am grateful to my lab mates, flat mates and other friends in Singapore. We have had enjoyable time together and they made my life here much easier.

Special thanks go to Dr. Liu Yang for his sharing of his source code on the skin tone ROI based rate control.

Finally, I have to thank my family for their love, support and encouragement that kept me completing the study.

Table of content

Acknowledgement	I
Table of content	II
List of figures.....	V
List of tables.....	VII
Summary	IX
Chapter 1. Introduction	1
1.1. Background	1
1.2. Motivations.....	3
1.2.1. Block Matching for Video Coding.....	3
1.2.2. Face region based bit allocation for video coding.....	4
1.3. Major Contributions	6
1.4. Organization of the thesis.....	9
Chapter 2. Background and Literature Review.....	11
2.1. Hybrid Video Coding	11
2.2. Motion Estimation.....	13
2.2.1. Definition of motion.....	13
2.2.2. Solutions for motion estimation	15
2.3. Block Matching	17
2.4. Rate Control	21
2.5. Face Detection.....	23
2.6. Summary.....	25
Chapter 3. A New Multiplication-Free Matching Criterion for Block Motion Estimation ..	
.....	26
3.1. Introduction	26
3.2. Deviation Analysis of SAD against MSE	28
3.2.1. SAD and MSE for Block Matching	28
3.2.2. Deviation of SAD against MSE	29
3.3. A New Criterion: Weighted Sum of Absolute Differences.....	31
3.3.1. Definition of Weighted Sum of Absolute Difference.....	31
3.3.2. Weight Determination	33

3.3.3. Optimized Weight for Multiplication Free WSAD.....	34
3.3.4. Geometric Interpretation of Different Measurements.....	36
3.4. Multi-Element WSAD.....	38
3.4.1. Extension of WSAD from Two-element to Multi-element.....	38
3.4.2. Computational Complexity Analysis	40
3.5. Experimental Result and Remarks	41
3.5.1. Block Motion Estimation Results	41
3.5.2. Block Matching Based Motion Estimation for Video Coding.....	43
3.5.3. Vector Quantization	50
3.6. Summary	52
Chapter 4. Transform-Exempt Calculation of Sum of Absolute Hadamard Transformed Differences (SATD)	53
4.1. Introduction	53
4.2. Review of Distortion Metrics in H.264/AVC	55
4.3. Transform-Exempt SATD Calculation	57
4.3.1. TE-SATD Calculation for 2-Point SATD	58
4.3.2. TE-SATD Calculation for 4-Point SATD	61
4.3.3. TE-SATD Calculation for 4×4 block SATD.....	62
4.4. Experimental Results.....	66
4.5. Summary and Discussion	67
Chapter 5. Efficient Block Matching Using Multi-level Intra- and Inter-Subblock Features: Subblock-Based SATD	69
5.1. Introduction.....	69
5.2. Review of SAD Based Measurement.....	71
5.3. SAD Based on Multi-level Intra- and Inter-features.....	73
5.3.1. Analysis of intra-subblock feature based SAD measurement.....	73
5.3.2. Proposed sum and contrast based SAD	75
5.3.3. Multi-Level Sum-Contrast Features.....	78
5.3.4. Fast Feature Calculation Based On Integral Image.....	79
5.3.5. Computation Complexity Analysis	80
5.4. Experimental Results.....	83
5.5. Summary and Discussion	86
Chapter 6. Face Region Based Bit Allocation for Conversational Video Coding.....	88

6.1. Introduction.....	88
6.2. Efficient Motion-based Sub-image Rejection for Face Detection	91
6.2.1. Pyramid structure in face detection: complexity analysis.....	91
6.2.2. Efficient motion detection using block mean difference.....	94
6.2.3. Sub-image rejection using motion features	95
6.3. Face Region Based Bit Allocation in H.264/AVC	97
6.3.1. Face contour extraction	97
6.3.2. Facial geometry based bit allocation.....	100
6.4. Face ROI Based Video Coding Architecture	102
6.5. Experimental Results.....	103
6.6. Summary.....	112
Chapter 7. Conclusion and Discussion	113
7.1. Conclusions	113
7.2. Recommendations for Future Work	115
Appendix..	117
Appendix A.....	117
Appendix B.....	117
Author's publications.....	119
Bibliography	A

List of figures

Figure 2-1 The hybrid video coding structure	12
Figure 2-2 Illustration of 2-dimension projection of 3-dimension motion.....	14
Figure 2-3 A general motion estimation processing system.....	16
Figure 2-4 Illustration of the block based motion estimation.....	16
Figure 3-1 SAD and MSE between two block vectors $\mathbf{b}_r$ and $\mathbf{b}_c$ in two-element case	29
Figure 3-2 Dynamic deviations of SAD, optimal WSAD and MF-WSAD from MSE	36
Figure. 3-3 Geometric comparison of MSE, WSAD and SAD, (a) Relationship of Unit Distance trajectories, (b) ‘Contour map’ of MSE, SAD and WSAD passing through point P.....	37
Figure 3-4 n -element WSAD calculation in an m -level pyramid structure.	39
Figure 3-5 Frame-by-frame motion compensation PSNR comparison. (a) “Dancer”, (b) “Coastguard”	43
Figure 3-6. Comparison of rate-distortion curves using different metrics in H.263: (a) “Foreman.CIF”; (b) “Coastguard. CIF”	45
Figure 3-7 Frame-by-frame rate-distortion comparison using different metrics in H.263 for “Foreman.CIF” sequence: (a) Frame-by-frame reconstructed PSNR results; (b) Number of bits used for coding each frame.	45
Figure 3-8 Frame-by-frame bit comparison based on different metric (including two partial WSAD) in coding “Flower” sequence.....	47
Figure 3-9 Rate-distortion curves using H.264 based on different metrics: (a) “BBC.CIF”, (b) “Foreman. CIF”, with frame rate of 30 fps and QP = 20, 25, 30, 35, 40.	50
Figure 4-1 Geometric comparison of unit distance trajectories of SAD and SATD.	60
Figure 5-1 One intra- and three inter-subblock features in a block.....	75
Figure 5-2 Multi-level structure for <i>sum-contrast</i> features.	79
Figure 5-3 Efficient computation of a feature using the integral image.....	80

Figure 5-4 Rate-distortion performance using SAD and SAD _{SC} at different levels for two sequences.	84
Figure 6-1 Pyramid structure in classification based face detection.	91
Figure 6-2 The number of sub-images N generated in the pyramid structure w.r.t scale factor r	93
Figure 6-3 The number of sub-images N generated in the pyramid structure w.r.t step size s	93
Figure 6-4 Efficient computation of mean using integral image.	95
Figure 6-5 Sub-image rejection procedure based on the three motion features.	96
Figure 6-6 Three motion features for motion-based sub-image rejection.	96
Figure 6-7 Normal line searching strategy.	99
Figure 6-8. Facial geometry priority map for bit allocation.	102
Figure 6-9. Proposed face ROI based coding architecture.	103
Figure 6-10 Comparison of the subimage numbers with and without the proposed 3-stage rejection using motion features for frame #10 in “MissAmerica” sequence.	104
Figure 6-11 Coding frame #30 in Foreman.Cif sequence with 64k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map; (d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.	108
Figure 6-12 Coding frame #80 in Foreman.Cif sequence with 128k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map;(d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.	109
Figure 6-13 Coding frame #45 in Salesman.QCIf sequence with 28k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map;(d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.	110
Figure 6-14 Coding frame #62 in News.Cif sequence with 28k bps. (a) Original frame; (b) ROI detection result; (c) Bit allocation priority map;(d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.	111

List of tables

Table 2-I Face detection approaches	24
Table 3-I. Deviation Comparison of SAD, Optimal WSAD and MF-WSAD from MSE	36
Table 3-II Average PSNR comparison for block motion estimation using different matching criteria	42
Table 3-III Maximum frame PSNR differences using SAD and WSAD against MSE	42
Table 3-IV. Average rate-distortion performance for all P frames in H.263 video coding, benchmarked against WSAD (QP=15).....	46
Table 3-V Comparison of average rate-distortion performance using PWSAD on tested sequences	47
Table 3-VI. Average rate-distortion variation from WSAD in H.264 (QP=30).....	50
Table 3-VII Average Motion Estimation Time for One frame (in seconds)	50
Table 3-VIII. VQ encoding quality (PSNR) comparison of different metrics, benchmarked against WSAD.....	51
Table 4-I. Computational comparison for conventional, fast transform based and TE- SATD based SATD calculations: “ <i>add</i> ”, “ <i>abs</i> ” and “ <i>cmp</i> ” refer to addition/subtraction, absolute and comparison operations, respectively	66
Table 4-II Comparison of SAD, Fast-Transform based SATD and TE-SATD based SATD in H.264/AVC (Intra Frame Only, time in seconds)	67
Table 5-I . Comparison of required operations for motion estimation with different measures (an example of $L=352$, $M=288$, $W=31$ and $N=16$).....	83
Table 5-II Coding performance using SAD_{SC} and SAD (PSNR: dB, bit rate: kbps)	85
Table 5-III Coding performance comparison for three feature-based SADs at different levels (PSNR: dB, bit rate: kbps).....	86
Table 5-IV Computation time comparison for three feature based SADs at different levels (time in seconds).....	86

Table 6-I Sub-image number comparison with and without motion based rejection	104
Table 6-II Coding performance comparison of JM98, Liu's skin tone ROI method and our scheme for CIF sequences	106
Table 6-III Coding performance comparison of JM98, Liu's skin tone ROI method and our scheme for QCIF sequences	106
Table 6-IV Coding performance comparison of JM98, Liu's and our scheme for sequences with two faces	107

Summary

Video coding has been intensively studied for over one decade and a few video compression standards have been well established with wide applications. These coding paradigms or standards have a number of commonalities, the most salient one among which is the hybrid coding structure consisting of diverse techniques to remove both the temporal and the spatial redundancy. One of the major techniques is block matching which has been used in inter-prediction like block motion estimation as well as in intra-prediction such as intra-mode selection in H.264/AVC. Another important topic is bit allocation for rate control to optimize the coding performance. In this thesis, we investigate the block matching technique and face region priority based bit allocation for video coding.

While the block matching based motion estimation has been widely studied, this thesis addresses the efficient block matching from the distortion measurement perspective. Specifically, three aspects on the distortion measurement analysis are considered, namely, a new multiplication-free distortion metric, transform-exempt sum of absolute Hadamard transformed differences (SATD) calculation, and a subblock-based distortion metric for efficient block matching, which are summarized in order as follows.

It is known that the distortion measure function plays a pivotal role in both matching accuracy and computational complexity. To avoid multiplication operations for simpler implementation, sum of absolute difference (SAD) is normally taken as a substitute to approximate the most widely accepted benchmark, mean squared error (MSE). Although it is well known that SAD is an approximate to MSE, it is necessary to know how accurate the approximate is quantitatively, which can provide us more insights and guidance in selecting an appropriate matching criterion function and developing some new ones. However, there is a lack of such quantitative treatment to this fundamental question. In this thesis, we firstly examine the quantitative deviation of SAD from MSE. In order to reduce the deviation, a new matching criterion, namely weighted sum of absolute difference (WSAD), is thereafter proposed, which enhances the matching

accuracy while still maintaining the desirable multiplication-free property. The proposed WSAD is also experimentally validated by applying it in block motion estimation for video coding, showing better rate-distortion performance than SAD.

Secondly, in the latest video coding standard H.264/AVC, the sum of absolute Hadamard transformed differences (SATD) is a new distortion metric adopted as an alternative to the SAD to improve coding efficiency. However, SATD requires more computation load due to the Hadamard transform involved. Inspired by a geometric interpretation for the simplest 2-point SATD calculation, an efficient method to compute the SATD, termed as transform-exempt SATD (TE-SATD), is proposed, which considers the joint effect of the Hadamard transform and the following SAD processing in the SATD calculation, thus enabling the computation of SATD without performing the Hadamard transform. With the same coding performance, the proposed TE-SATD based fast algorithms can save significant computation load.

Then we investigate subblock based efficient metrics for block matching. The conventional SAD distortion measure is based on pixels. Recently, the matching with the intra-subblock feature based SAD measures has been developed to save the huge computational load in block matching based motion estimation. In this thesis, we examine the intra-subblock features based SAD measure, and find that it may yield a detrimental effect on coding efficiency. To overcome the deficiency while still reducing the computation, a set of both intra- and inter-subblock based features are proposed. According to the analysis, the new features based SAD measure can be considered as subblock based SATD, which is computed in a multi-level structure to save computation load in a scalable way.

Most video coders encode every part of the video frame with equal importance. However, as the human visual system (HVS) usually pays the most intensive attention to the region of interest (ROI), a priori knowledge about the content of the video scene could be exploited to improve coding efficiency by coding the ROI more accurately than the rest of the video content. Therefore, we develop an efficient and effective face region based bit allocation for conversational video coding. In order to be able to employ face region based coding, this thesis introduces a robust and efficient motion based face detection method together with a contour extraction method to locate and segment

accurate face ROI in the video content. Based on the extracted accurate face ROI, a facial feature geometry based bit allocation priority model is proposed to improve the ROI reconstruction visual quality for conversational video coding.

Chapter 1.

Introduction

1.1. Background

In the past decade or so, along with the greatly decreasing cost of video acquisition and storage devices, fascinating developments in multimedia representation, processing and communications have taken place, enabling the wide adoption of video technologies such as digital television (DTV), Internet video streaming and digital versatile disk (DVD) video storage. As the video information becomes more pervasive, the demands for multimedia services are rapidly increasing and the expectation of the video quality is becoming higher, which is reciprocally driving the multimedia techniques to continuously evolve.

Among all the techniques involved in video processing, video compression or coding is the key part for the multimedia applications, without which all the digital video storage and transmission would not be practical. The video compression is achieved by removing the three types of redundancy within natural digital video sequences, that is, temporal redundancy, spatial redundancy and statistical redundancy. The property of strong correlations between successive video frames is utilized to predict the current frame from the previously coded frames using motion estimation/compensation to remove the temporal redundancy, while the removal of the spatial redundancy within individual frames is achieved by predicting the current block from the coded blocks together with the transform based coding. Entropy coding then removes the statistical redundancy, reducing the average number of bits necessary to represent the binary video stream.

Various video compression standards (MPEG-1/2/4, H.261/263 and H.264/AVC etc.) have been developed since early 90's and they have been widely used through the years [1]-[5]. Although these standards have specified the decompression procedure, many issues regarding the implementations of compression algorithms are still open for improvements and optimization. Moreover, while the compression ratio has been improved significantly as the development of the video coding techniques (e.g., the latest video coding standard H.264/AVC is able to provide half bit rate savings when compared with MPEG-2 and can provide around 25%-50% compression gain over H.263), these improvements are achieved at the expense of high computing complexity caused by the newly adopted techniques. Therefore, in spite of the advances in video storage and transmission capacity, more efficient and effective video compression is still an important topic for the following years to come, due to the expectation of higher visual quality, especially in video communication system over the rapidly developing wireless network where the bandwidth is still relatively limited and the computation resource is restricted.

Meanwhile, as the video processing technologies have been intensively studied with wide range of diverse techniques employed in the hybrid coding structure commonly used in the well established video coding standards, the boundaries between video processing, graphics and computer vision begin to blur, making more advanced video processing techniques possible with the input from many disciplines. This, therefore, arouses interdisciplinary research by collaboration from different fields. For example, as the research of the computer vision advances, besides the imitation of the simple vision functions by applying low-level image processing techniques, more sophisticated and intelligent techniques to accomplish higher-level visual tasks have attracted great attention during past years, resulting in more advanced techniques to achieve more complicated goals. These various advanced computer vision techniques can be adopted to achieve higher compression ratio and to manage and retrieve the fast-increasing video volume in more meaningful ways. However, due to the short computation time available in the adjacent video frames, requirements of efficient computation are much higher in video processing.

With the digital information under consideration being the natural video content, which offers the motion information in adjacent video frames, it is rational to exploit this

extra temporal information to improve both the efficiency and effectiveness of the algorithms when adopting of computer vision techniques to improve efficiency for video coding and *video understanding* [6].

1.2. Motivations

1.2.1. Block Matching for Video Coding

One of the major techniques used in video coding to achieve high compression ratio is block matching, which has been intensively used in inter-prediction like block motion estimation to remove temporal redundancy as well as in intra-prediction such as intra-mode selection in H.264/AVC to remove spatial redundancy. While the block matching based motion estimation has been widely studied, there are still some fundamental issues to be investigated and improved. This thesis addresses the efficient block matching from the distortion measurement perspective. Specifically, the following issues for the distortion measure motivate further consideration and analysis on it.

Firstly, it is known that the distortion measure function plays a pivotal role in both matching accuracy and computational complexity. Mean squared error (MSE) is the most widely accepted benchmark criterion for matching accuracy. To avoid multiplication (square) operations that yield high computational complexity as well as the difficulty in hardware implementation, sum of absolute difference (SAD) is normally taken as a substitute to approximate MSE. However, although it is well known that SAD is an approximate to MSE, it is necessary to know how accurate the approximate is quantitatively, which can provide us more insights and guidance in selecting an appropriate matching criterion function and developing some new ones. However, there is a lack of such quantitative treatment to the fundamental question.

Secondly, the sum of absolute Hadamard transformed differences (SATD) is an important distortion metric applied in the latest video coding standard H.264/AVC as an alternative to the sum of absolute differences (SAD) to improve the coding efficiency in the subpixel motion estimation and the low-complexity rate-distortion optimization

(RDO). However, SATD requires more computation load due to the involved Hadamard transform and thus more investigations are necessary to accelerate the calculation of SATD while maintaining the coding quality.

Thirdly, in order to speedup the motion estimation in video coding, intra-subblock based features or attributes are proposed to accelerate SAD based block matching which employs conventionally pixel-based SAD metric for distortion measure. However, although it is believed that these subblock feature based SAD can provide satisfactory matching results, the effect on coding efficiency of applying those feature based SAD in motion estimation for video coding yet needs to be examined theoretically.

1.2.2. Face region based bit allocation for video coding

Another important topic in video coding is the bit allocation for rate control to optimize video coding performance. Although the research for video coding has been pursued with impressive results, the subjective reconstructed video quality still suffers from some inherent problems, especially when the video is coded at low bit rate. Most of video coders simply encode every part of the video frame with equal importance without considering the video content. However, the research on human visual system (HVS) suggests that viewer pays more attention to region of interest (ROI) than the rest of the parts in the video scenario, and thus the visual quality in the ROI, such as the human faces in conversational videos, is more crucial in determining the subjective visual quality. Hence, it is rational to improve the compression ratio and subjective visual quality by selecting a small number of ROIs in each frame and to encode them in higher priorities than the rest of the frame. For instance, as the most common scenario of video conference and mobile video communication being the head-and-shoulder type of conversational video content, the important facial features like mouth and eyes should be coded with more bits for a higher subjective visual quality.

Two major issues have to be considered when the concept of ROI based coding priority is applied: the detection of the ROI and the ROI-based priority assignment (or the adjustment of the quantization parameter QP) for the basic coding units in the frame.

While the QP assignment has been well established, the segmentation of priority regions remains an open problem.

Many schemes utilize the visual features including motion to detect the ROI [7]-[9]. In [10], the skin color combined with a direct frame difference based motion detection was introduced to determine face ROI efficiently. However, as the skin color in video frames is sensitive to the lighting condition, it might be unsuitable for certain scenes. Moreover, the color based model cannot provide an accurate face region as other human skin parts such as hands and skin-color-like background may also be mis-included.

In fact, face detection in still image has been intensively studied in the past years with many successful methods reported [6][11][12], among which the learning based algorithms are widely considered as good solutions because of their excellent detection results [13]-[21]. It is therefore reasonable to adopt these methods for robust ROI detection. However, the following issues must be considered when applying those methods in ROI based video coding.

1.2.2.1. Challenge for face detection in video

To solve the unknown face sizes and locations problem in these methods, a pyramid structure is normally used, which partitions the original image and its scaled version into sub-images of fixed size. Then the face is detected if any sub-image is classified as a *face* pattern by a trained classifier. A very large number of sub-images are generated in the pyramid structure, demanding high computation load in classification procedure. Moreover, since the classification can not achieve 100% accuracy, the large number of sub-images will also increase the false alarm rate in face detection.

As the digital information under processing changes from still image to dynamic video sequence, it motivates to apply the motion information to accelerate the computation and improve the accuracy of the algorithm. In order to achieve this goal, efficient motion detection/estimation method and sub-image rejection scheme using motion information are necessary.

1.2.2.2. Face contour information

The output of face detection is usually a rectangular face region containing non-face parts [11]-[21]. Thus, it will cause blocking artifacts and bit waste if this whole rectangular face region is encoded with more bits. More accurate face outline/contour extraction is necessary to output a precise face region, which is especially advantageous for face region based video coding. Furthermore, the properties of the human visual system lead to the claims that edge and contour information are very important to human visual system and are responsible for our perception [22]. Therefore, this thesis uses the face contour to extract the face ROI region.

Moreover, the face contour offers important information for human face analysis. It can be used for 3-Dimension human face reconstruction, as well as in feature representation for human face recognition [23][24]. Algorithms for tracking the face contour have attracted a lot of attention nowadays due to its potential applications [24]-[27]. For video coding applications, accurate face contour provides useful geometrical facial information to enhance the subjective visual quality, and it has the potential to be used for face video object (VO) segmentation in content based video coding.

1.3. Major Contributions

Motivated by the above issues, we have studied the block matching algorithm from its distortion measurement perspective and a new facial geometry based bit allocation priority scheme, with the following contributions achieved.

- a) The first thorough and quantitative analysis of the deviation of SAD measure against MSE, and a proposed measure minimizing the distortion accordingly**

As mentioned above, sum of absolute difference (SAD) is normally taken as a substitute to approximate MSE to avoid multiplication (square) operations for simpler implementation while there is a lack of quantitative treatment on how close the approximation is. In this thesis, we first analyze statistically the

quantitative deviation of SAD from MSE in terms of maximum and average deviations. To minimize the average deviation, a new measure, namely weighted sum of absolute difference (WSAD), is proposed in two-element case firstly, for which an optimal weight is obtained theoretically. The weight is replaced using $\frac{1}{2}$ which is close to the optimal weight, so that the weighting operation can be simplified to a shift operation, and the resulted multiplication free WSAD also shows better matching accuracy with a smaller deviation from MSE than SAD. A geometric interpretation is presented to illustrate how these metrics differ in measuring distortions. The two-element WSAD is further extended to multi-element case in a multi-level pyramid structure. The proposed WSAD is experimentally validated by applying it in block motion estimation for video coding as well as in vector quantization (VQ), showing improvements over SAD in rate-distortion performance (in ITU-T H.263 and H.264 video codec) and in distortion performance (for a given codebook in VQ).

b) A proposed Transform-Exempt Calculation of Sum of Absolute Hadamard Transformed Differences (SATD)

The sum of absolute Hadamard transformed differences (SATD) is an important distortion metric applied in the latest video coding standard H.264/AVC, which is an alternative to the sum of absolute differences (SAD) to improve the coding efficiency in the subpixel motion estimation and the low-complexity rate-distortion optimization (RDO). However, SATD requires more computation load due to the Hadamard transform involved. Most fast algorithms have been introduced to speed up the Hadamard transform, thus accelerating the SATD calculation in turn. However, compared with the SAD computing, 64 more additions are still required in the fast Hadamard transform. Inspired by a geometric interpretation for the simplest 2-point SATD calculation, an efficient way to compute the SATD is found in this thesis, which considers the joint effect of the Hadamard transform and the following SAD processing in the SATD calculation and thus enables the computation of SATD without performing the Hadamard transform. We mathematically extend the 2-point transform-exempt

SATD (TE-SATD) computation scheme to 4-point and 4×4 block SATD calculation. With the same coding performance, the proposed TE-SATD based fast algorithms can save 38% and 17% computation operations for computing 4×4 block SATD when compared with the conventional SATD calculation and the fast Hadamard transform based calculation, respectively. The simulation results on the H.264/AVC reference software validate the efficiency of the proposed scheme.

c) An Efficient Block Matching Using Multi-level Intra- and Inter-Subblock Features: Subblock-Based SATD

Block matching based motion estimation employs conventionally pixel-based sum of absolute differences (SAD) metric for distortion measure. Many fast matching mechanisms have been developed to lower the computational load for block matching, such as those exploiting block feature-based or attribute-based SAD calculations. By examining a subblock-sum based SAD measure where the subblock-sum can be considered to be an intra-subblock feature, we propose both intra- and inter-subblock features based SAD measure for more effective block matching in terms of achieving better coding efficiency. Interestingly, the new feature-based SAD measure can be interpreted as the subblock-based SATD. The new features are constructed in a multi-level structure, which can provide a flexible and scalable means in achieving tradeoff between computation load and coding efficiency. The simulation results using standard video sequences validate the effectiveness and efficiency of the proposed scheme in terms of both coding performance and computation time, compared with other relevant feature based SAD measures.

d) Face region based bit allocation for video coding

In order to improve the visual reconstruction image quality for coding the conversational video, more bits should be allocated to face region according to the human visual system characteristics. In this thesis, we introduce a robust and accurate face region segmentation using efficient motion based face detection and

active contour method. The learning based face detection method is adopted to identify the face region robustly, which is improved by an efficient sub-image motion based rejection method. The accurate contour is further obtained using active contour model, from which a facial geometry based bit allocation priority scheme is proposed. The experiment result applying this face region based coding has demonstrated its advantage on improving the subjective visual quality significantly.

1.4. Organization of the thesis

The rest of this thesis is organized as follows:

Chapter 2 gives brief description of the background knowledge of the related topics and reviews the relative literature. The review will cover the related topics discussed in the thesis, including the hybrid video coding scheme, motion estimation, block matching, rate control and face detection.

Chapter 3 analyzes the deviation of SAD against MSE and proposes a new measure by minimizing the deviation. Firstly, the SAD metric is analyzed by statistically investigating its quantitative deviation from MSE. The result shows that the SAD measure considerably deviates from the MSE metric and deteriorates the matching accuracy accordingly. In order to reduce the deviation, a new criterion function, namely weighted sum of absolute difference (WSAD), is thereafter proposed, where the optimum weight is derived theoretically by minimizing the deviation from MSE and a practical weight is chosen to avoid multiplications in the WSAD. The development of WSAD is firstly considered in the two-element case and a multi-level structure is then introduced for a general multi-element case with a partial WSAD introduced to further save computation. Finally, the WSAD is validated by applying a few applications including video coding for H.263/4 and vector quantization.

Chapter 4 proposes the Transform-Exempt Sum of Absolute Hadamard Transformed Differences (SATD). We examine the joint effect of both steps involved in the 2-point SATD calculation in the first instance, based on which we derive a simpler way to

directly calculate the SATD in the original domain without performing the Hadamard transform. We then derive mathematically the 4-point and 4×4 block fast SATD calculation based on the 2-point TE-SATD scheme, where two methods are presented for 4×4 block fast SATD calculation. The simulation results applying the SATD in the H.264/AVC reference software is then demonstrated to validate the efficiency of the proposed scheme.

Chapter 5 introduces the efficient block matching using multi-level intra- and inter-subblock features. We investigate the performance of applying the intra-subblock features in the block matching motion estimation for video coding based on the analysis, and then a set of new intra- and inter-subblock features are introduced in a multi-level structure to improve block matching performance. Afterwards, the interpretation of the new feature-based SAD and the computational complexity analysis are presented. The experimental results using the proposed motion estimation method in video coding are shown in the end.

Chapter 6 describes the face region based bit allocation for conversational video coding. The number of sub-image generated from the pyramid structure commonly used in face detection methods is thoroughly analyzed firstly. And then an efficient motion based sub-image rejection method is introduced, followed by description on a modified active contour model to extract the accurate face region. Finally, a facial geometry priority model for bit allocation is proposed, offering significantly visual quality improvements when applied in rate control in H.264.

Chapter 7 summarizes and concludes the thesis.

Chapter 2.

Background and Literature Review

In this chapter, we conduct a brief background and literature review on the related topics discussed in the thesis, including hybrid video coding scheme, motion estimation, block matching, rate control and face detection.

2.1. Hybrid Video Coding

Due to wide range of applications, video coding has been intensively studied. Various video compression standards (MPEG-1/2/4, H.261/263 and H.264/AVC etc.) have been established and widely used throughout the years [1]-[5]. Although diverse techniques have been adopted in these international video coding standards, they have a number of commonalities, and the block based hybrid coding structure consisting of predictive coding and transform coding to remove both the temporal and the spatial redundancy is essentially the core among them. The diagram of this hybrid coding structure is shown in Figure 2-1. The reasons that the hybrid structure coding is conducted on a block basis, which divides the input frame into non-overlapping blocks and applies the coding process over each block, are mainly that a) the relative lower computational complexity of block based transform than the whole frame based one makes it more practical, although the transform based on the entire frame can fully exploit the correlation among the pixels; and b) while predicting the whole frame using only one displacement vector does not provide enough granularity to yield a good prediction accuracy, predicting each pixel with individual motion vectors might consume more bits to code the motion vectors than coding the pixels directly. However, the block size for temporal prediction coding (motion estimation/compensation) might not be the same as that for transform coding.

Figure 2-1 The hybrid video coding structure

Transform coding is one of the most popular waveform-based coding schemes for images, and it is the basis for the well known JPEG image coding standard and the video coding standards. In the block based transform coding, a transform matrix containing a set of orthogonal basis functions is employed to transform the original pixel values in a block. Based on the transform matrix, the non-overlapping blocks will be transformed into a set of coefficients which represent the block as linear combination of the basis functions, thus decorrelating the original pixel values in the block and concentrating their energy into a few coefficients. Since the pixel values usually vary smoothly in nature images and video frames, the discrete cosine transform (DCT), with its low frequency coefficients representing the smooth image blocks very well, is the most commonly used transform in image and video coding. The transformed coefficients will then be quantized separately using scalar quantizers and the quantized coefficients will be finally coded using entropy coding.

In predictive coding, motivated by the fact that the adjacent pixels (both temporally and spatially) usually takes similar values, the current pixel is first predicted from the coded adjacent pixels in the same frame or previously coded frames, and the prediction error, instead of the pixel value itself, is thereafter encoded using transform coding. The accuracy of the prediction is thus very critical to the coding performance for predictive

coding. While there are other possible methods, the prediction is usually achieved using a block matching process to select according to certain distortion measure the best prediction block out from a few previously coded blocks either temporally or spatially. Therefore, block matching is one of the major techniques in predictive video coding, which is used in inter-prediction like block motion estimation as well as in intra-prediction such as intra-mode selection in H.264/AVC. In fact, the temporal predictive coding using motion estimation/compensation is the key to the success of the modern video coding standards, which will be further described in the following section.

Generally speaking, in modern video coding schemes, the temporal redundancy in adjacent frames is commonly removed by predictive coding, and the spatial redundancy is removed by transform coding and partially by spatial predictive coding. Another type of redundancy in video signals, the statistical redundancy, is further removed by entropy coding after the temporal and spatial redundancies have been removed.

2.2. Motion Estimation

From the hybrid video coding structure introduced in the preceding section, it can be seen that the motion estimation is a key element in the video coding system. In this section, we further review some basic concepts and some solutions to the motion estimation problem.

2.2.1. Definition of motion

Typically, images are created as the projections of the 3-dimension objects onto a 2-dimension image plane as shown in Figure 2-2. The intensity of the image at each point in the 2-dimension image plane represents the amount of light falling on the surface at that spatial point at a particular time. Let $I(\mathbf{x}, t)$ denote this intensity located at $\mathbf{x}=(x, y)$ in the image at time t . The term motion is described as follows.

The projection of the true 3-D motion of objects on the image plane is referred to as *real motion*. The variation of the intensities $I(\mathbf{x}, t)$ in the image sequence, which is the only perceivable observation in images, is referred as *apparent motion* [28]. To be more

precise, the observed apparent motion in image sequence is referred to as optical flow [29], which is characterized by the displacement vectors $\Delta \mathbf{x} = (d_x, d_y)$ describing the correspondence of the pixels of two images with time interval Δt . The optical flow calculation is a very important topic in computer vision field with extensive applications such as 3-dimension depth reconstruction and obstacle detection [30]-[32]. It should be noted that the term motion commonly referred to in video coding is the *apparent motion*, which may not be the real 2D motion of the objects. This is because the intensity variation can also be caused by the camera movements and the illumination changes besides the object movements. On the other hand, moving objects must bear sufficient texture to generate apparent motion, that is to say, objects with uniform luminance cannot be observed in images.

Figure 2-2 Illustration of 2-dimension projection of 3-dimension motion.

The objective of motion estimation for predictive video coding purpose is different from that for the computer vision applications, which aims at finding the blockwise motion vectors that can yield minimum distortion when predicting current frame using motion compensation (MC). Thus, the fact that the estimated displacement vectors cannot resemble the real motion is not problematic in video coding. Furthermore, in video coding techniques, the motion in the image plane is approximated using a model of in-plane 2D translation for the convenience of algorithms and the ease of their implementations. Therefore, motion estimation (ME), which in video coding field refers to the process of determining 2D translational motion vectors from adjacent frames to

achieve minimum prediction error, is a simplified form of the optical flow calculation. Further information on motion estimation can be found in [33]-[37].

2.2.2. Solutions for motion estimation

A general process for motion estimation is shown in Figure 2-3. A finite consecutive sequence of images of a dynamic scene is fed to the motion estimator. In order to estimate the motion vectors in the image plane, the motion estimator commonly assumes that the *intensity conservation* principle holds, that is, the pixel intensities I of the corresponding points on the objects remains constant even after undergoing a movement of Δx (motion vector) in the image plane during a frame interval Δt :

$$I(\mathbf{x}, t) = I(\mathbf{x} + \Delta \mathbf{x}, t + \Delta t) \quad (2.1)$$

where $I(x, t)$ is the intensity of the pixel at $\mathbf{x} = (x, y)$ for time t , and $\Delta \mathbf{x} = (d_x, d_y)$ is the motion vector. When dx , dy and Δt are small, the right hand side of Equation (2.1) can be expanded to Tailor series as

$$I(\mathbf{x} + \Delta \mathbf{x}, t + \Delta t) = I(\mathbf{x}, t) + \frac{\partial I}{\partial x} d_x + \frac{\partial I}{\partial y} d_y + \frac{\partial I}{\partial t} d_t \quad (2.2)$$

Substituting Equation (2.1) into (2.2) and dividing both sides with d_t , we have

$$\frac{\partial I}{\partial x} \frac{d_x}{d_t} + \frac{\partial I}{\partial y} \frac{d_y}{d_t} + \frac{\partial I}{\partial t} = 0 \quad \text{or} \quad \frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} = 0 \quad (2.3)$$

where $\mathbf{v} = (v_x, v_y)$ is the velocity vector. Equation (2.3) is known as the optical flow equation.

Many techniques have been proposed to solve the motion estimation problem [38] [39]. The difference between the categories mainly lies in the ways they represent the raw images and their unique way of utilizing the optical flow equation in (2.3). Some examples are the gradient based methods [38][40], the frequency based methods [41], the phase based methods [41]-[45], the pixel recursive methods [46]- [49], Bayesian methods [50]. Due to their implementation simplicity and suitability for hardware implementation,

the block based motion estimation are among the most widely used motion estimation methods especially in video coding [51]. Instead of explicitly utilizing the optical flow Equation (2.3), the block based motion estimation methods directly employ the *intensity conservation* Equation (2.1) and assume that the motion for all the pixels in one whole block is uniform.

Figure 2-3 A general motion estimation processing system.

Figure 2-4 Illustration of the block based motion estimation.

The simplest form of block based motion estimation assumes that the block motion is translational in the image plane as illustrated in Figure 2-4. A block $\mathbf{B}$ of sized $N \times N$ at (x, y) in the frame t undertakes a translational movement to a new location $(x+d_x, y+d_y)$ in frame $t + \Delta t$, and the motion vector for this block is then $\mathbf{mv} = (d_x, d_y)$. One of the limitations of this model is that it is only suitable for translational motion. In the cases of zoom, rotational motion and local deformation situations, it might fail to obtain the correct motion vector. This limitation can be overcome by the so called generalized or

deformable block motion matching algorithm, in which the spatial block motion model is generalized in a more complex form. Refer to [53] for more detailed explanation of this algorithm.

Based on the two assumptions (i.e., *intensity conservation* and uniform whole block motion), the block matching algorithm (BMA) is commonly employed to obtain the motion vector in block based motion estimation, which is further discussed in the following.

2.3. Block Matching

From the preceding description on the predictive coding and block based motion estimation, it can be found that block matching, the technique to search and align a predefined or input template in some candidate blocks, is a very important technique for video coding. In fact, block matching algorithm (BMA) is the 2D case of a wider range of algorithms, the so-called *template matching*, which has a wide range of applications in both computer vision and image/video processing fields, such as correspondence point matching in stereo vision, image registration, vector quantization, visual tracking, intra-frame prediction and block motion estimation for video coding [54]-[56].

Block matching problem is generally formulated by quantifying the similarity (or equally the distortion) between the template and the candidate blocks, and the candidate with largest similarity (or smallest distortion) is taken as the best match, which can be expressed as

$$Match = Candidate^{(u',v')} \mid \arg \max_{(u,v) \in \Omega} \{S(Candidate^{(u,v)}, Template)\} \quad (2.4)$$

where S is the matching criterion function or known as similarity function, the superscript (u, v) denotes the coordinate displacement of a candidate block and Ω is its full set within the search range. In the context of block motion estimation, (u', v') infers the motion vector. The similarity function S is often replaced equivalently by a distortion function when it is applied in video coding. From Equation (2.4), the similarity metric (or

the distortion metric) S along with the strategy to search the best match in the set Ω determines the matching accuracy and computational complexity for the block matching.

A number of similarity criterion functions have been developed to achieve different purposes. In video coding, where the target for BMA is to find the best prediction block whose error block can be encoded with minimum bits, the mean square error (MSE) is widely regarded as a good (if not the best) criterion in terms of matching performance. Consider blocks of size $N \times N$ (commonly with $N=16, 8$ or 4), the MSE in the motion estimation context is computed as:

$$MSE = \frac{1}{N^2} \sum_{j=-N/2}^{N/2} \sum_{i=-N/2}^{N/2} \left(I(x+d_x+i, y+d_y+j, t+\Delta t) - I(x+i, y+j, t) \right)^2 \quad (2.5)$$

In a general way, denoting any two blocks under consideration (template or reference block and candidate block) as column-wised vectors of N^2 dimension, $\mathbf{b}_r$ and $\mathbf{b}_c$, the MSE can be rewritten as

$$MSE = \frac{1}{N^2} \|\mathbf{b}_c - \mathbf{b}_r\|_2^2 = \frac{1}{N^2} \|\mathbf{d}\|_2^2 \quad (2.6)$$

with $\mathbf{d} = \mathbf{b}_r - \mathbf{b}_c$ being the difference vector between $\mathbf{b}_r$ and $\mathbf{b}_c$, and $\|\mathbf{a}\|_p$ denotes p-norm of vector $\mathbf{a}$, $\|\mathbf{a}\|_p = (|a_1|^p + |a_2|^p + \dots + |a_n|^p)^{1/p}$. So MSE is proportional to the squared L_2 -norm of the difference vector $\mathbf{d}$ (i.e., the square of Euclidean distance between block vector $\mathbf{b}_c$ and $\mathbf{b}_r$). As shown in Equation (2.4), in block matching the comparison between MSEs is more concerned rather than their exact values, so the average operation in Equation (2.6) can be ignored without affecting the block matching solution, which then becomes the sum of squared error (SSE) used in some literatures.

The mean square error (MSE) is widely regarded as a good (if not the best) criterion in terms of matching performance. Many matching performance indices are developed based on the MSE, such as the peak signal to noise ratio (PSNR) used to measure the objective reconstruction quality in image and video coding and restoration. However, multiplication (or square) operations involved in MSE calculation, which demand comparatively high computation power and complex hardware implementation, restrict its applications especially for real-time processing and low computation power conditions.

Consequently, the sum of absolute difference (SAD) is commonly used as a substitute due to its computational efficiency and ease for hardware implementation by supplanting the multiplication with the absolute operation. In a similar way, SAD is computed as

$$SAD = \sum_{j=-N/2}^{N/2} \sum_{i=-N/2}^{N/2} |I(x+d_x+i, y+d_y+j, t+\Delta t) - I(x+i, y+j, t)| \quad (2.7)$$

or

$$SAD = \|b_c - b_r\|_1 = \frac{1}{N^2} \|d\|_1 \quad (2.8)$$

which is the L_1 -norm distance of the block vectors, or so called Manhattan distance (also known as city block distance).

Another criterion, pixel difference classification (PDC) [51], compares each co-sited pixels of the two blocks and classifies each pixel pair as either matching or not matching. Pixels are matching if the differences between their values are less than some threshold. And the greater the number of matching pixels, the better the match. Therefore, the number of matching pixels will be counted and taken as the similarity for the two blocks. PDC can be viewed as a variant of SAD.

Regarding the searching strategy to find the best match in the search range, the straightforward method is full search (FS), which exhaustively compares the template block with all the candidate blocks. For example, in block based motion estimation for video coding shown in Figure 2-4, FS check possible motion vector in the search window $\Omega = M \times M$ and performs similarity function $M \times M$ times, where M is usually set to 17 for small motion and 33 for large motion. A very large amount of computation is thus demanded in full search strategy. Various fast schemes have been developed to accelerate the algorithm in motion estimation for video coding, such as [57]-[63].

One way to accelerate the BMA is to optimize searching procedure. Many fast searching strategies, which either inspect a subset of the whole checking points in the search range or eliminate impossible candidate blocks with minimal computation, have been proposed in the past ten years. One group of these strategies uses fixed search patterns, such as three-step search, four-step search, hexagon search [58][59]. These methods can greatly accelerate BMA and their simplicity is attractive for hardware

implementation, but they may result in higher matching error compared with the full-search algorithm because the final matched block may not be the global minimum. Another group of fast searching strategies reduce the checking points with run-time determining the searching scheme. The successive elimination algorithm (SEA) and its modified versions remove the impossible candidate blocks using the inequality property between the final SAD value and the efficient sum of norms, and thus decrease a great number of checkpoints quickly to avoid intensive computation [60]-[63]. Also, the partial distortion elimination (PDE) and its modified versions use the partial sum of matching distortion to eliminate impossible candidates with small amount of computation [57].

Instead of optimizing the searching procedure, another way to accelerate the BMA is to optimize the basic matching procedure (the similarity or distortion calculation) itself. Since the basic matching procedure is repeated for all the checking points in the search range, small computation time reduction in this procedure can achieve a large computational gain. For instance, fast algorithms based on pixel sub-sampling are proposed to accelerate the matching procedure for all the blocks by only using a fraction of the pixels for matching procedure [64][65]. Fast method based feature based matching have also been introduced to speeded up the block matching process or to simplify their implementation on hardware [66]-[69]. It should be noted that the fast methods accelerating the matching procedure can also be incorporated into some of the fast searching based methods for further computation saving. It is also worthy noting that although it is believed that the fast matching algorithms usually provide satisfactory matching results, their effects on coding efficiency and quality is not well analyzed in depth.

In this thesis, we will further investigate and improve the efficiency for the block matching algorithm from the distortion measure function perspective by especially considering their effect on the coding efficiency.

2.4. Rate Control

From the hybrid coding structure described above, the residual blocks from the prediction coding stage will be encoded by the transform coding. If the coding parameters, such as the quantization step in the transform coding and the search range for predictive coding, are kept constant for all the blocks and frames, the bits used for each frame will vary drastically because the content of each frame is continuously changing especially in video sequences with high motion and complexity. This is adverse for transmitting the coded video stream through networks due to the constraints of the limited bandwidth and the decoder buffer size. Therefore, rate control, the technique to achieve a constant target bit rate by altering the encoding parameters, is very important for many practical video applications such as video steaming through networks.

The purpose of rate control is to optimize the visual quality of the reconstructed video while achieving a constant target bit rate by altering the coding parameters. The most obvious parameter to control is the quantization parameter QP (or the corresponding quantization step Q_s) since increasing the QP reduces the coding bit rates and vice versa. According to the size of the basic unit for rate control, the QP could be adjusted at macroblock (MB) level, slice level or frame level [70]. However, since the increasing QP will degrade the reconstructed video quality, a mechanism for controlling the QP to balance the coding bit rate and the reconstruction quality. Based on the classic bit allocation work [71]-[73], it has been widely studied in digital video coding and applications. For example, a simple linear rate control model is employed for MPEG-2 TM5 [74]:

$$R = \frac{K}{Q_s} \quad (2.9)$$

where K is a constant, R is the target bits for the basic unit. It is improved in [75][76] for MPEG-4 using a quadratic model to achieve the target bit rate more precisely:

$$R_{texture} = MAD_{actual} \times \left(\frac{K_1}{Q_s^2} + \frac{K_2}{Q_s} \right) \quad (2.10)$$

where $R_{texture}$ is the target texture bits, MAD_{actual} is the actual mean absolute difference of the current basic unit to be coded, and K_1 and K_2 are the second and first order of the R-Q model used in [75][76].

This model is not directly applicable in the latest video coding standards H.264/AVC due to QP-MAD dilemma caused by the adopted rate distortion optimization (RDO) [77]. That is to say, as the QP is used in RDO based motion estimation, the MAD_{actual} used by the rate control to adjust the QP in (2.10) is only available after the QP is used to generate it. Li *et al* have introduced an estimation method for MAD_{actual} using the available information from the previously code frames in [77] and [78] to overcome the dilemma

$$MAD_{actual,pred}[i] = Y_1[i] \times MAD_{actual}[i-1] + Y_2[i] \quad (2.11)$$

where $MAD_{actual,pred}[i]$ denotes the predicted MAD for the basic rate control unit in frame I , $MAD_{actual}[i-1]$ is the actual MAD for the co-sited basic unit in frame $i-1$ and Y_1 and Y_2 are the first order and zeroth order of the prediction model, which are updated after encoding the basic unit. Based on the predicted MAD, the quantization parameter is adjusted by

$$R_{texture}[i] = MAD_{actual,pred}[i] \times \left(\frac{K_1}{Q_s^2} + \frac{K_2}{Q_s} \right) \quad (2.12)$$

However, all these rate control models adjust the QP for bit allocation by treating every part of the video frame equally regardless of its effect on the subjective visual quality on the human visual system. For example, in some video types such as the head-shoulder type of conversational video, the content of the scene is known and thus the region of interest (ROI), such as the mouth and eyes in the conversational video, can be exploited to allocate more bits for coding these regions more accurately to achieve a higher subjectively video quality. In order to achieve ROI based coding, Liu *et al* have introduced a face ROI determination method using the skin color combined with a direct frame difference based motion detection [10]. Since the skin color in video frames is sensitive the lighting condition, it might be unsuitable for certain scenes. Therefore, as the review in the following section shows that face detection problem in still image has

been intensively studied with many successful and robust methods developed, it is rational to adopt more of these methods to achieve more robust ROI determination.

2.5. Face Detection

As robust face detection is essential in face region based video coding, we briefly review the face detection methods here. The general statement of the face detection problem may be described as: given an arbitrary image or image sequence, enable the computer to automatically determine, if any, the locations and extensions of the human faces in the images. The face detection in still images has been intensively studied in the past decade or so. A wide variety of face detection algorithms have been developed, which can be divided into two broad categories according to reference [12]: feature based and image based methods. This classification is distinguished by their ways of utilizing the knowledge about the face and the required *a priori* information of face. The feature-based methods are so called because they explicitly use the feature knowledge of the human face, such as the apparent properties of the skin color [79]-[82], geometry and shape of human faces [83] [84]. Different from the feature based methods, the image base methods use face knowledge implicitly [13][16][20][21][85] . A detailed list of various methods in face detection is shown in Table 2-I. Some general surveys can be found in [11] [12].

Partially due to the fast development of the machine learning theories, the learning based algorithms recently exhibit a great appeal with their excellent results among those methods [13]-[21]. The learning based methods solve the face detection problem by formulating a two-class pattern classification problem, in other words, it judges every image region into ‘face’ or ‘non-face’ pattern. Different pattern classification methods are employed both for the purpose to increase the detection accuracy in certain situations. Despite the different pattern classification methods they employ, in order to solve the unknown face location and size problem, they often share a common pyramid structure to divide the original image and the scaled image into a set of sub-image patterns.

Table 2-I Face detection approaches

Face detection Approaches	1.Feature-based approaches	1.1 Low-level analysis	1.1.1 Edges[90][93]
			1.1.2 Gray-levels[94]
			1.1.3 Color[79][80][81][82][86]
			1.1.4 Motion[99][100][101]
			1.1.5 Texture[91][92]
			1.1.6 Multiple features[97][98]
		1.2 Feature analysis	1.2.1 Feature searching[106][107]
			1.2.2 Constellation analysis[108][109][110]
		1.3 Active shape models	1.3.1 Snakes[23]
			1.3.2 Deformable templates[111][112]
			1.3.3 Point distribution models (PDMs)[113][114]
	2.Image-based approaches	2.1 Linear subspace methods	(Eigen-face)[95][96]
		2.2 Statistical approaches	2.3.2 Naïve Bayes Classifier[115]
			2.3.3 Information-Theoretical Approach[117]
			2.3.4 Hidden Markov Model[116]
		2.4 Learning based methods	2.4.1 Neural networks[13][16][17][18][89]
			2.4.2 Support Vector Machine[85][102][103]
			2.4.3. SNoW[19]
			2.4.4 Adaboost[20][21][104]
			2.4.5 Ensemble[87][88][102][105]

While the detection accuracy is mainly determined by the employed pattern classification method itself, the computation time for learning based face detection lies on two factors. One factor is the speed of the classification process, which is inherent to employed classification method. The other factor, the number of the sub-images generated for the input image, is independent of the classification method selection. It is determined only by the pyramid structure. However, the pyramid structure usually produces a large number of sub-images and thus demands very high computation load for the classification since the classification process has to perform once for every and each of these sub-image patterns. The reduction of the number of sub-images can greatly accelerate the face detection algorithm. In order to overcome this shortcoming, rejection based scheme has been reported to achieve fast detection rate [118]. The idea of the rejection based scheme is to use very simple classifiers to quickly eliminate the simple ‘*non-face*’ sub-image patterns to focus more on those possible ‘*face*’ patterns. It stems from the observation that, typically, only a small portion of an image is possible face regions. A similar method called cascade classifiers follows the same thought in the

fastest face detection system reported in [20] by Viola and Jones. The fast speed is achieved by further accelerating the sub-image verification procedure by their single variable classifiers using the quick Haar-like features.

In this thesis, since the media concerned is image sequences, rather than single static images, the author proposes to further reduce the number of sub-images by taking the advantage of the motion information available. This motion based sub-image reduction method first detects the motion, and then only those sub-images that satisfy some motion features will be input the classification process. This approach can significantly reduce the number of sub-images.

2.6. Summary

In this chapter, some related concepts and solutions for hybrid video coding, motion estimation, block matching, rate control and face detection are described. It can be found that the block matching is a very important technique in video coding. In the following chapters we will analyze the block matching for video coding from the measurement function aspect. In addition, the review of face detection reveals that face detection in still images has various solutions and thus is potential to apply for improving the subjective visual quality of the conversational type of video. This will be further discussed in chapter 6.

Chapter 3.

A New Multiplication-Free Matching Criterion for Block Motion Estimation

3.1. Introduction

As introduced in Chapter 2, block matching, the technique to search and align a predefined or input template in some candidate blocks, has a wide range of applications in both computer vision and image/video processing fields, such as correspondence point matching in stereo vision [54], image registration [55], vector quantization [56], visual tracking and block motion estimation [62]. Block matching problem is generally formulated by quantifying the similarity between the template and the candidate blocks and the candidate with largest similarity is taken as the best match.

$$Match = Candidate^{(u',v')} \mid \arg \max_{(u,v) \in \Omega} \{S(Candidate^{(u,v)}, Template)\} \quad (3.1)$$

where S is the matching criterion function or known as similarity function, the superscript (u, v) denotes the coordinate displacement of a candidate block and Ω is its full set within the search range. In the context of block motion estimation, (u', v') infers a motion vector.

From Equation (3.1), the similarity metric S along with the set Ω determines the matching accuracy and computational complexity for the block matching. A number of criterion functions have been developed, among which mean square error (MSE) is widely regarded as a good (if not the best) criterion in terms of matching performance.

Many matching performance indices are developed based on the MSE, such as the peak signal to noise ratio (PSNR) used to measure the reconstruction quality in image and video coding and restoration. However, multiplication (or square) operations involved in MSE calculation, which demand comparatively high computation power and complex hardware implementation, restrict its applications especially for real-time processing. Contrarily, sum of absolute difference (SAD) is commonly used as a substitute due to its computational efficiency and ease for hardware implementation by supplanting the multiplication with the absolute operation.

Although it is well known that SAD is an approximate to MSE as a matching criterion and thus it was intensively employed in the investigations to achieve fast computation [62]-[63], it is necessary to know to what extent the approximate is quantitatively, which will provide us more insights and guidance in selecting an appropriate matching criterion function and developing some new ones. However, to the best of our knowledge, very few papers have addressed this fundamental question in depth. In this chapter, we firstly analyze the SAD metric by statistically investigating its quantitative deviation from MSE. The result shows that the SAD measure considerably deviates from the MSE metric and deteriorates the matching accuracy accordingly. In order to reduce the deviation, a new criterion function, namely weighted sum of absolute difference (WSAD), is thereafter proposed, where the optimum weight is derived theoretically by minimizing the deviation from MSE. To avoid multiplications in the WSAD, a practical weight is chosen such that the multiplication is replaced by the simplest shift operation. The multiplication-free WSAD with the practical weight is also shown to outperform the SAD with a significantly smaller deviation error. In this way, the new WSAD criterion enhances the matching accuracy while still maintaining the desirable multiplication free property. The development of WSAD is firstly considered in the two-element case. A multi-level structure is then introduced for a general multi-element case and a partial WSAD is discussed to further save computation.

The remainder of this chapter is organized as follows. Section 3.2 briefly reviews SAD and MSE followed by statistically investigating the quantitative deviation of SAD from MSE. Section 3.3 proposes the new WSAD metric and determines theoretically the optimal weight as well as a practical weight realizing multiplication free WSAD in the

two-element case. The deviation analysis for the new WSAD is thoroughly made and a geometric interpretation is presented to different metric for an intuitive justification. Section 3.4 extends WSAD from two-element to multi-element case, where the computational complexity of the multi-element WSAD is analyzed and a partial WSAD is presented to further save computation. And section 3.5 shows the comparative experimental results by applying the proposed WSAD criterion in the block matching based motion estimation for video coding.

3.2. Deviation Analysis of SAD against MSE

3.2.1. SAD and MSE for Block Matching

In block/template matching, a current block and a reference (candidate) block of size $N \times N$ can be represented as two column-wise vectors of N^2 elements $\mathbf{b}_c$ and $\mathbf{b}_r$ respectively. The SAD and MSE measurements between the two vectors are defined as the L_p -norm values of their difference vector $\mathbf{d} = \mathbf{b}_c - \mathbf{b}_r$:

$$E_p = \|\mathbf{b}_c - \mathbf{b}_r\|_p = \|\mathbf{d}\|_p \quad (3.2)$$

with $p=1$ for SAD and $p=2$ for MSE, respectively, and $\|\mathbf{v}\|_p = (|v_1|^p + |v_2|^p + \dots + |v_n|^p)^{1/p}$ denotes the L_p -norm of vector $\mathbf{v}$. In some literatures, the above MSE's squared version E_2^2 (also known as sum of squared error, or SSE) and normalized version E_2^2/N^2 are used to represent MSE. In fact all these representations are equivalent and interchangeable as a criterion for block matching. For ease of notations, we consider the MSE defined in Equation (3.2).

By using L_1 -norm, SAD exempts its calculation from multiplication operations involved in MSE by using absolute operations instead, which provides SAD the merit of low computational cost and ease for hardware implementation leading to its wide adoption in practice. However, it is known that SAD deteriorates the matching performance compared to MSE to some extent, which will be quantitatively analyzed in the following.

3.2.2. Deviation of SAD against MSE

We firstly make the analysis in the two-element case. Denoting the difference vector $\tilde{\mathbf{d}} = \tilde{\mathbf{b}}_c - \tilde{\mathbf{b}}_r = (d_1, d_2)^T$, we have the deviation of SAD from MSE

$$D(\tilde{\mathbf{d}}) = SAD(\tilde{\mathbf{d}}) - MSE(\tilde{\mathbf{d}}) = (|d_1| + |d_2|) - \sqrt{|d_1|^2 + |d_2|^2} \quad (3.3)$$

In polar-coordinates representation, we can rewrite (3.3) as

$$D(\theta) = (|r \cos(\theta)| + |r \sin(\theta)|) - r \quad (3.4)$$

where $MSE = r = \sqrt{|d_1|^2 + |d_2|^2}$, and $\theta \in [0, 2\pi)$ is the angle between $\tilde{\mathbf{d}}$ and x axis, shown in Figure 3-1.

Figure 3-1 SAD and MSE between two block vectors $\mathbf{b}_r$ and $\mathbf{b}_c$ in two-element case

The deviation $D(\theta)$ between MSE (the length of right-angled triangle's hypotenuse, shown as r) and SAD (the sum of the lengths of the two sides of the right-angled triangle, shown as dashed lines d_1 and d_2) is evident in the figure: for a constant MSE value r , SAD varies with the value of θ when θ changes from 0 to 2π and so does the deviation $D(\theta)$ accordingly. The deviation of SAD from MSE can be summarized as follows.

1. The maximum and minimum deviation

Since $|\cos(\theta)| + |\sin(\theta)| \geq 1$, consequently $D(\theta) \geq 0$, which means the SAD measurement is always greater than or equal to MSE when measuring the same

candidate block. This can also be seen from Fig. 1 that for a triangle the sum of two side-lengths ($|d_1|+|d_2|$) is greater than the third side-length (r).

It can be further obtained that $D(\theta)$ reaches its maximum $D_{\max} = (\sqrt{2} - 1)r \approx 0.4142r$ at $\theta = \pi/4, 3\pi/4, 5\pi/4$ or $7\pi/4$ and its minimum $D_{\min} = 0$ at $\theta = 0, \pi/2, \pi$ or $3\pi/2$ (see proof in Appendix A). That is to say, the maximum deviation of SAD from MSE occurs on a smooth difference vector (with $d_1 = d_2$ or $\theta = \pi/4, 5\pi/4$), which corresponds to a reference block with only DC change from current block like a brightness change. This largest positive deviation will decrease the possibility for such a reference block to win the match based on the SAD criterion. On the other hand, SAD becomes equal to MSE for a difference vector with a zero element ($d_1 = 0$ or $d_2 = 0$), which makes the difference vector more likely to win in the SAD-based matching process (note that SAD is always no less than MSE). In view that with same SAD value, a difference vector containing a zero element means the largest difference in the two elements, which leads to a high frequency value in the transform domain. Therefore, from the perspective of coding efficiency, the SAD measure is a worse criterion in that it favors high frequency residual blocks but puts the smooth error blocks at disadvantage.

2. Average deviation

Given $\text{MSE} = r$, the average deviation of SAD from MSE is obtained as:

$$D_{\text{avg}} = \int_0^{2\pi} |D(\theta)| d\theta = \int_0^{2\pi} (|r \cos(\theta)| + |r \sin(\theta)|) - r d\theta = (8 - 2\pi)r \quad (3.5)$$

This formula indicates that the average deviation of SAD from MSE is proportional to the corresponding MSE measurement r . It implies that block matching using SAD may yield inconsistent result with that using MSE metric, especially when the MSE measurement becomes larger.

We now consider the deviation in the multi-element case. Supposes a 3-element difference vector $(d_1, d_2, d_3)^T$, with the SAD and MSE for the first two elements as $\text{SAD}(d_1, d_2)$ and $\text{MSE}(d_1, d_2)$. According to the above two-element deviation analysis, the

smaller MSE can be represented as $\text{MSE}(d_1, d_2) = \text{SAD}(d_1, d_2) - D(d_1, d_2)$, where $D(d_1, d_2) \geq 0$ is the deviation value. Applying the inequality $\sqrt{a^2 + b^2} \leq |a| + |b|$ and the definitions of SAD and MSE in Equation (3.2), we have

$$\begin{aligned} \text{MSE}(d_1, d_2, d_3) &= \sqrt{\text{MSE}^2(d_1, d_2) + d_3^2} \leq \text{MSE}(d_1, d_2) + |d_3| \\ &= \text{SAD}(d_1, d_2) - D(d_1, d_2) + |d_3| = \text{SAD}(d_1, d_2, d_3) - D(d_1, d_2) \end{aligned} \quad (3.6)$$

That is

$$\text{SAD}(d_1, d_2, d_3) - \text{MSE}(d_1, d_2, d_3) \geq D(d_1, d_2) \quad (3.7)$$

Inequality (3.7) indicates that the deviation of SAD from MSE in 3-element case will be greater than or equal to (only when $\text{MSE}(d_1, d_2) = 0$ or $d_3 = 0$) the deviation in the two-element case. Extending this analysis, we can obtain that the deviation of SAD from MSE in the $(n+1)$ -element case vector will be greater than or equal to that in the n -element case. Therefore, the deviation of SAD from MSE in the multi-element case may be accumulated to be larger and larger as the number of elements increases.

3.3. A New Criterion: Weighted Sum of Absolute Differences

3.3.1. Definition of Weighted Sum of Absolute Difference

The analysis of SAD against MSE in the preceding section has revealed the significant deviation of SAD from MSE and the disadvantage of SAD in coding error blocks. In order to overcome or alleviate the disadvantages, a new criterion, namely weighted sum of absolute difference, is considered. The new criterion is on one hand to improve the matching accuracy to be as close to MSE as possible, and on the other hand to maintain the desirable multiplication-free property of SAD. The proposed weighted sum of absolute difference (WSAD) criterion is firstly developed in the two element case and then it will be extended to the multi-element case.

In two-element case, where the difference vector between two block vectors is $\tilde{\mathbf{d}} = (d_1, d_2)^T$, WSAD is defined as:

$$WSAD(\tilde{\mathbf{d}}) = \begin{cases} |d_1| + w|d_2| & \text{if } |d_1| \geq |d_2| \\ w|d_1| + |d_2| & \text{if } |d_2| \geq |d_1| \end{cases} \quad (3.8)$$

or equivalently in polar-coordinate representation where $\tilde{\mathbf{d}} = (r \cos(\theta), r \sin(\theta))^T$,

$$WSAD(r, \theta) = \begin{cases} r|\cos(\theta)| + rw|\sin(\theta)| & \text{if } |\cos(\theta)| \geq |\sin(\theta)| \\ rw|\cos(\theta)| + r|\sin(\theta)| & \text{if } |\sin(\theta)| \geq |\cos(\theta)| \end{cases} \quad (3.9)$$

where $w > 0$ is a weight to be determined in the following subsection. SAD is a special WSAD with $w=1$.

It can be seen from Equation (3.9) that the WSAD is both periodic with period of $\theta = \pi/2$ and symmetric with respect to points at $n\pi/4$ ($n = 0, 1, 2, \dots$). In view of this periodic and symmetric property, we can investigate the deviation of $WSAD(r, \theta)$ from MSE by considering the deviation function $D(\theta)$ only in the range of $\theta \in [0, \pi/4]$, which can be denoted as

$$D(\theta) = r(\cos(\theta) + w\sin(\theta) - 1) \quad \theta \in [0, \pi/4] \quad (3.10)$$

From (3.10), the maximum deviation of WSAD from MSE can be obtained as

$$D_{\max} = r[\cos(\arctg(w)) + w\sin(\arctg(w)) - 1] \quad (3.11)$$

at $\theta = \arctg(w)$ (see proof in Appendix A). The average deviation can be calculated as

$$D_{\text{avg}} = \int_0^{2\pi} |D(\theta)| d\theta = 8r \int_0^{\pi/4} |(\cos(\theta) + w\sin(\theta) - 1)| d\theta \quad (3.12)$$

To make the WSAD approximate MSE statistically to the best, the average deviation D_{avg} shown in (3.12) should be minimized with respect to w , which leads a way to determine the weight w .

3.3.2. Weight Determination

The weight, w , is determined to make WSAD approximate MSE to the closest. That is to say, optimal w minimizes the average deviation of WSAD from MSE D_{avg} .

Theorem 1. WSAD, as defined in Equation (3.8), achieves minimum average deviation from MSE when $w = \sqrt{2} - 1$ with the minimum average deviation being $D_{avg \min} = (16\sqrt{2} - 16 - 2\pi)r$, where r is the MSE value for the difference vector $\tilde{d}$.

Proof: The average deviation of WSAD from is:

$$D_{avg} = 8r \int_0^{\pi/4} |(\cos(\theta) + w \sin(\theta) - 1)| d\theta \quad (3.13)$$

The solution is to find w that minimizes D_{avg} . For $\theta \in [0, \pi/4]$, to remove the absolute sign in Equation (3.13), two cases are considered in the following.

- If $w \geq \sqrt{2} - 1$, it can be proved that $\cos(\theta) + w \sin(\theta) - 1 \geq 0$. Then Equation. (3.13) can be written as

$$D_{avg} = 8r \int_0^{\pi/4} (\cos(\theta) + w \sin(\theta) - 1) d\theta = 8r \left(\frac{1}{\sqrt{2}} - \frac{\pi}{4} + (1 - \frac{1}{\sqrt{2}})w \right) \quad (3.14)$$

which is monotonically increasing with w . Therefore $w = \sqrt{2} - 1$ corresponds to the minimum average deviation $D_{avg \min} = (16\sqrt{2} - 16 - 2\pi)r$.

- If $0 < w \leq \sqrt{2} - 1$, it can be found that $\theta^0 = \arccos(\frac{1-w^2}{1+w^2})$ is the only zero-crossing point of $\cos(\theta) + w \sin(\theta) - 1$. Then

$$\begin{aligned} D_{avg} &= 8r \left\{ \int_0^{\theta^0} (\cos(\theta) + w \sin(\theta) - 1) d\theta - \int_{\theta^0}^{\pi/4} (\cos(\theta) + w \sin(\theta) - 1) d\theta \right\} \\ &= 8r \left\{ -2 \arccos\left(\frac{1-w^2}{1+w^2}\right) + w \left(\frac{1}{\sqrt{2}} + 3 \right) - \frac{1}{\sqrt{2}} + \frac{\pi}{4} \right\} \end{aligned} \quad (3.15)$$

By setting the first-order derivative $D'_{avg} = 0$ to find the extremum of D_{avg} , we have $w^3 + w - \frac{4\sqrt{2}}{3\sqrt{2}+1} = 0$. This is a depressed cubic Equation, for which the real root can be calculated according to Cardano's solution [119], $w \approx 0.7144$. It is beyond the range of $w \leq \sqrt{2} - 1$. Therefore, no stationary points for the function D_{avg} exist in the interval $0 < w \leq \sqrt{2} - 1$. For $0 < w \leq \sqrt{2} - 1$, it can be found $\frac{dD_{avg}}{dw} < 0$, which implies D_{avg} is monotonically decreasing as w increases from 0 to $0 < w \leq \sqrt{2} - 1$. Thus, D_{avg} also reaches the minimum when $w = \sqrt{2} - 1$.

In conclusion, when $w = \sqrt{2} - 1$, WSAD has the minimum average deviation from MSE $D_{avg \min} = (16\sqrt{2} - 16 - 2\pi)r \approx 0.3408r$, which is considered as the optimal WSAD with the optimal weight $w = \sqrt{2} - 1$. It can be seen that WSAD can approximate MSE more closely than SAD whose average deviation from MSE is $(8 - 2\pi)r \approx 1.7168r$ shown in (3.5). Moreover, according to (3.11), the maximum deviation of the optimal WSAD from MSE can be obtained as $0.0824r$, which is also much smaller than that of SAD from MSE $0.4142r$. ■

3.3.3. Optimized Weight for Multiplication Free WSAD

Compared with SAD calculation, the optimal WSAD above requires one multiplication due to the weight in (3.8). The optimal weight $w = \sqrt{2} - 1 \approx 0.414$ can be replaced by $\frac{1}{2}$, which can be realized by a simple 1-bit right shifting operation while offering a WSAD with an average deviation from MSE very close to that of the optimal WSAD as will be explained in the followings. Hence, a practical weight, which is optimized to strike a good tradeoff between matching accuracy and computational complexity, is chosen as $w = 1/2$, and the multiplication free WSAD (MF-WSAD) based on the optimized weight is represented as

$$MF_WSAD(\tilde{\mathbf{d}}) = \begin{cases} |d_1| + |d_2|/2 & \text{if } |d_1| \geq |d_2| \\ |d_1|/2 + |d_2| & \text{if } |d_2| \geq |d_1| \end{cases} \quad (3.16)$$

And the deviation of MF-WSAD from MSE can be summarized as following:

1. Maximum deviation from MSE

From (3.11) the MF-WSAD has the maximum deviation from MSE $D_{\max} = r(\sqrt{3}/2 - 3/4) \approx 0.1r$ at $\theta = \arctg(1/2)$. The minimum deviation is obtained with $D_{\min} = 0$ when $d_1 = 0$ or $d_2 = 0$ (i.e., $\theta = 0, \pi/2, \pi$ or $3\pi/2$).

2. Average deviation from MSE

With $w = 1/2 > \sqrt{2} - 1$, the average deviation of MF-WSAD from MSE can be calculate according to (3.14) as $D_{\text{avg}} = (2\sqrt{2} + 4 - 2\pi)r \approx 0.5452r$.

3. Deviation comparison of SAD, optimal WSAD and MF-WSAD

Table 3-I summarizes the maximum and average deviations of SAD, optimal WSAD and MF-WSAD from MSE respectively. It is evident from the table that the improvement of the optimal WSAD over SAD is remarkable with greatly reduced average and maximum deviation, and that the optimized MF-WSAD approximates the optimal WSAD very closely.

It is also shown in Table 3-I that the maximum deviation of SAD from MSE occurs when $\theta = \pi/4$ ($d_1 = d_2$), which disfavors blocks producing smooth error blocks with DC component only, as discussed in sub-section II-B. In contrast, it can be found that the MF-WSAD reaches the maximum deviation at $\theta = \arctg(1/2)$, that is $d_2 = d_1/2$. These results suggest that, compared with the SAD metric, the MF-WSAD does not only approximate MSE more closely, but also increases the winning chance for those blocks with DC value changes in the matching process. This property is much desirable in motion estimation for video coding as it can increase the smoothness of the error blocks, thus leading to bit saving in encoding the error blocks.

Figure 3-2 dynamically illustrates the deviations of SAD and WSAD from MSE as θ changes from 0 to $\pi/4$, from which it can be seen how the SAD, optimal WSAD and MF-WSAD are deviating from the MSE line with the changing θ . The SAD

curve goes farther away from the MSE line as θ increases from 0 to $\pi/4$, whereas the two WSAD curves ascend until reaching maximum around the middle and then descend closer to the MSE line as θ comes near to $\pi/4$. It can be clearly seen that the optimal WSAD and MF-WSAD constantly have smaller deviation values than SAD at any value of θ .

Table 3-I. Deviation Comparison of SAD, Optimal WSAD and MF-WSAD from MSE

Deviation From MSE	SAD	Optimal WSAD	MF-WSAD
Maximum Deviation D_{max} (Corresponding θ in $[0, \pi/4]$)	$0.4142r$ ($\theta = \pi/4$)	$0.0824r$ ($\theta = \pi/8$)	$0.1180r$ ($\theta = \arctg(1/2)$)
Average Deviation D_{avg}	$1.7168r$	$0.3408r$	$0.5452r$

Figure 3-2 Dynamic deviations of SAD, optimal WSAD and MF-WSAD from MSE

3.3.4. Geometric Interpretation of Different Measurements

This sub-section further geometrically interprets the relationship of the MSE, SAD and WSAD metrics to visually demonstrate their difference and the occurrence of inconsistent matching results when applying them respectively. Firstly, Figure. 3-3 (a) illustrates the geometry relationship among the MSE, SAD and WSAD metrics by drawing their unit distance trajectories comparatively. MSE stands for Euclidean distance which has a circle-shaped unit distance trajectory, while trajectories of unit SAD and optimal WSAD are its inscribed square and regular octagon respectively. The unit distance trajectory of MF-WSAD is also an octagon very close to but inside

the regular octagon of optimal WSAD. Note that a point in the plane corresponds to a difference vector in block matching. It can be seen that any point in the gap area between unit MSE and unit SAD trajectories has an MSE value smaller than one but with an SAD measurement larger than one, which results in the deviation of SAD from MSE. Therefore, for blocking matching employing MSE and SAD metric respectively, points in this area yield inconsistent matching results when compared to the intersected points A, B, C or D, all of which have unit measurement value in both SAD and MSE. The larger the area is, the more points it contains and then the more likely the mismatch may occur. Since the WSAD corresponds to an octagon unit distance trajectory that better approximates a circle, the gap area between MSE and WSAD is much smaller, which greatly reduces the likelihood of mismatching.

Figure. 3-3 Geometric comparison of MSE, WSAD and SAD, (a) Relationship of Unit Distance trajectories, (b) 'Contour map' of MSE, SAD and WSAD passing through point P.

In a more general case, suppose two difference vectors $\tilde{\mathbf{d}}_1$, $\tilde{\mathbf{d}}_2$ are to be compared in the matching process, and the first one $\tilde{\mathbf{d}}_1$ corresponds to the point P in Figure. 3-3 (b). We can draw the trajectories of identical MSE, SAD and WSAD measurement passing through it in a way similar to 'contour map' in topography. Then, using SAD and MSE criteria respectively, inconsistent matching results will occur if the point representing the second difference vector $\tilde{\mathbf{d}}_2$ falls in the gap area between SAD and MSE contours, where

$MSE(\tilde{\mathbf{d}}_1) > MSE(\tilde{\mathbf{d}}_2)$ but $SAD(\tilde{\mathbf{d}}_1) < SAD(\tilde{\mathbf{d}}_2)$ or vice versa. Similarly, with WSAD and MSE matching criteria respectively, mismatch occurs when $\tilde{\mathbf{d}}_2$ is in the gap between WSAD and MSE contours. It can be found that the gap area between MSE and SAD is much larger than that between MSE and WSAD (for ease of comparing, the two gap areas are shown in shadow in Figure. 3-3 (b) by excluding their overlapping areas, among which the gap area for SAD in light shadow is much larger than that for WSAD in dark shadow). Therefore compared with WSAD, SAD is more likely to yield matching results inconsistent with MSE. It can be anticipated that the closer to a circle the identical-distance trajectory of a metric is, the less mismatch there will be between the metric and the MSE.

3.4. Multi-Element WSAD

3.4.1. Extension of WSAD from Two-element to Multi-element

Based on the two-element WSAD in the preceding section, multi-element WSAD is developed to approximate multi-element MSE in this section. For a n -element difference vector $\mathbf{d} = (d_1, d_2, \dots, d_n)^T$, its MSE (L_2 norm) can be decomposed as:

$$\|\mathbf{d}\|_2 = \left(\sum_{i=1}^{n/2} \|\tilde{\mathbf{d}}_i\|_2^2 \right)^{1/2} \quad (3.17)$$

where the two-element vector $\tilde{\mathbf{d}}_i$ ($i=1, \dots, n/2$) is made up of the $(2i-1)$ th and $(2i)$ th elements of $\mathbf{d}$, i.e., $\tilde{\mathbf{d}}_i = (d_{2i-1}, d_{2i})^T$. This decomposition suggests a way to extend WSAD from two-element to multi-element in the following.

For the n -element difference vector $\mathbf{d}$, grouping its neighboring elements two-by-two into a series of two-element vectors $\tilde{\mathbf{d}}_i$ ($i=1, \dots, n/2$), and calculating the two-element WSAD values of these two-element vectors respectively, we can have a new $n/2$ -element vector as $\mathbf{d}_{L1} = (WSAD(\tilde{\mathbf{d}}_1), WSAD(\tilde{\mathbf{d}}_2), \dots, WSAD(\tilde{\mathbf{d}}_{n/2}))^T$. Considering WSAD is a close approximation to MSE in two-element case, that is, $\|\tilde{\mathbf{d}}_i\|_2 \approx WSAD(\tilde{\mathbf{d}}_i)$, we substitute this into (3.17)

$$\|\mathbf{d}\|_2 \approx \left(\sum_{i=1}^{n/2} \|WSAD(\tilde{\mathbf{d}}_i)\|_2^2 \right)^{1/2} = \|\mathbf{d}_{L1}\|_2 \quad (3.18)$$

By iteratively applying this two-element WSAD approximating process m times ($m = \log_2 n$), we obtain an approximation to the n -element MSE as:

$$MSE(\mathbf{d}) = \|\mathbf{d}\|_2 \approx \|\mathbf{d}_{L1}\|_2 \approx \|\mathbf{d}_{L2}\|_2 \approx \dots \approx \|\mathbf{d}_{L(\log_2 n)}\|_2 = d_{L(\log_2 n)} \quad (3.19)$$

where the result of the last two-element WSAD, denoted as $d_{L(\log_2 n)}$, is then taken as the n -element WSAD to approximate MSE for difference vector $\mathbf{d}$. Figure 3-4 illustrates the above iterative calculation process for the multi-element WSAD based on two-element WSAD computing, which is performed from bottom to top. The highest level that contains only one element will be the final n -element WSAD.

Figure 3-4 n -element WSAD calculation in an m -level pyramid structure.

The extension method introduced here is simply based on the fact that the addition is associative. For two difference vectors $\mathbf{d}_1$ and $\mathbf{d}_2$, which are composed of identical elements but in different order (with the same MSE value), however based on the aforementioned extension, their multi-element WSAD measurements may be different because different grouping in the multi-level pyramid structure may lead to different results. Therefore, how to optimize the grouping to make the multi-element WSAD maximally approximate MSE is still open, which will be considered in our next step work. Sorting of the elements before grouping may be a good alternative but the complexity of

sorting is not trivial. There may exist some other methods or variants to generalize the WSAD from 2-dimension to multi-element.

3.4.2. Computational Complexity Analysis

It is known that one SAD calculation with $N \times N$ block size (difference vector of $n = N^2$ element) takes N^2 subtractions, N^2 absolute operations and $N^2 - 1$ additions, the computational complexity of which is represented as

$$C_{SAD} = O(SUB(N^2) + ABS(N^2) + ADD(N^2 - 1)) \quad (3.20)$$

For a general discussion on different implementation platforms, we use $SUB(\bullet)$, $ABS(\bullet)$ and $ADD(\bullet)$ to present the computational complexity for subtraction, and absolute and addition operations respectively.

In the multi-element WSAD calculation, N^2 subtractions and N^2 absolute operations are firstly used to calculate each absolute element of the difference vector $\mathbf{d}$ as that in SAD. Then for a two-element WSAD calculation at each level, one comparison, one shift and one addition are needed. It can be seen from Figure 3-4 that in the proposed m -level pyramid WSAD structure, $N^2/2$ two-element WSAD calculations are required at the root level while level by level the number of two-element WSAD calculations is reduced by half until only 1 such calculation at the last (top) level, totaling $N^2 - 1$ two-element WSAD calculations across all levels. Therefore, totally N^2 subtractions, N^2 absolute operations, $N^2 - 1$ comparisons, $N^2 - 1$ shifts and $N^2 - 1$ additions are needed for one n -element WSAD calculation

$$C_{WSAD} = O(SUB(N^2) + ABS(N^2) + ADD(N^2 - 1) + CMP(N^2 - 1) + SHF(N^2 - 1)) \quad (3.21)$$

where $CMP(\bullet)$ and $SHF(\bullet)$ denote the complexity of comparison and shift operation.

Comparing (3.21) with (3.20), one n -element WSAD calculation needs additionally $N^2 - 1$ comparisons and $N^2 - 1$ shift operations than its counterpart SAD calculation. Furthermore, the analysis extending WSAD from two elements to multi-element also implies that the lower level in the pyramid structure, the more additional comparison and

shift operations are required. Based on this observation, in order to further reduce the computation of WSAD in the multi-level pyramid structure, a partial WSAD (PWSAD) scheme can be developed by applying two-element WSAD only at higher levels while conventional two-element SAD being used in the lower levels. For instance, when SAD is applied at the root level and WSAD is used for the remaining levels, $N^2/2$ two-element WSAD calculations are skipped. Partial WSAD with more levels calculated using SAD can be considered similarly to save more computation. However, as shown in the preceding section SAD cannot approximate MSE as closely as WSAD does, the reduction of additional computation here is achieved at the cost of sacrificing the accuracy in approximating MSE. More levels calculated with SAD will accumulate larger deviation thus deteriorating the matching performance, which will be validated by the experimental results in the next section. The multi-level partial WSAD structure is also shown in Figure 3-4. In the figure, we take the notation of $PSAD_L(m)$ to refer to the PWSAD applying WSAD from the level m upwards.

3.5. Experimental Result and Remarks

The proposed matching criterion can be used in applications involving block/template matching. Here we consider three such applications, that is, block matching based motion estimation, its application in video coding, and vector quantization. For video coding, we tested on both H.263 and H.264 platforms. In H.263 coding, cost function based on residual block only is used for motion estimation with block size of 16×16 or 8×8 . In contrast, in H.264 the cost function based on both the residual block and motion vector is adopted for motion estimation, where the block size can vary from 16×16 to 4×4 among seven possible block sizes. To simplify the notation, in the following we denotes the developed MF-WSAD as WSAD.

3.5.1. Block Motion Estimation Results

Firstly, comparative experiments were conducted for block motion estimation using different matching criteria, where original frames were used as reference frames. Eight QCIF and CIF sequences were tested which contain various motion activities like slow

motion (“Carphone”), medium to fast motions (“Coastguard”, “Dancer”), camera panning (“Stefan”). The block size was 16×16 and full search strategy was applied with the search range of 32×32 .

The average motion compensation PSNR comparisons for different video sequences based on SAD, WSAD and MSE matching criteria are tabulated in Table 3-II. It can be seen that our proposed WSAD produces much closer performance to MSE than the SAD metric consistently for all the tested sequences. The average PSNR improvement of WSAD over SAD is up to 0.23dB for the motion compensated frames of the sequence “Dancer”. Table 3-III shows the maximum frame PSNR differences of SAD and WSAD from MSE for all the tested sequences. Compared with the MSE metric, the SAD leads to a maximum frame PSNR loss of more than 1dB for the sequences “Dancer” and “Foreman”, in sharp contrast to around 0.1dB maximum frame PSNR loss for all the tested sequences by using the WSAD.

Table 3-II Average PSNR comparison for block motion estimation using different matching criteria

Sequence	Carphone	Stefan	Coastguard	Dancer	Goldfish	BBC	Foreman	Akiyo
Format	QCIF	QCIF	CIF	CIF	CIF	CIF	CIF	CIF
SAD	30.180	24.671	30.475	31.162	27.573	24.503	32.073	38.726
MSE	30.349	24.756	30.608	31.437	27.807	24.664	32.220	38.843
WSAD	30.314	24.734	30.593	31.390	27.757	24.616	32.196	38.821
SAD - MSE	-0.169	-0.085	-0.133	-0.275	-0.235	-0.161	-0.147	-0.118
WSAD - MSE	-0.035	-0.022	-0.015	-0.047	-0.050	-0.048	-0.024	-0.022

Table 3-III Maximum frame PSNR differences using SAD and WSAD against MSE

Sequence	Carphone	Stefan	Coastguard	Dancer	Goldfish	BBC	Foreman	Akiyo
SAD - MSE	-0.512	-0.431	-0.580	-1.054	-0.524	-0.429	-1.171	-0.983
WSAD - MSE	-0.104	-0.145	-0.095	-0.148	-0.109	-0.183	-0.117	-0.124

Figure 3-5 illustrates frame-by-frame comparison for the “Dancer” and “Coastguard” sequences respectively, where the first 50 frames are plotted. It can be seen from the Fig. that the SAD matching criterion leads to the lowest PSNR values for the motion compensated frames, while MSE and the proposed WSAD criterion matching measures provide very close results both consistently better than that of SAD. In Figure 3-5 (b) for

the sequence of “Coastguard”, WSAD and MSE measures can outperform the SAD metric by nearly 0.5dB in a few frames around the 36th frame.

(a)

(b)

Figure 3-5 Frame-by-frame motion compensation PSNR comparison. (a) “Dancer”, (b) “Coastguard”

3.5.2. Block Matching Based Motion Estimation for Video Coding

It has been shown from the theoretical analysis in the previous sections that WSAD can always approximate much closer to the MSE than the SAD. When applied in the motion estimation for video coding, it can be predicted that the WSAD can find motion vectors more consistent with the MSE. Moreover, as analyzed in Sub-section II-B and III-C, compared with the conventional SAD metric, the MSE and WSAD tend to favor those blocks with DC value changes in the matching process, which will produce smooth residual block thus saving bits in coding the residual blocks. For example, consider two residual blocks (99, 0, 0, 0) and (25, 25, 25, 25). The SAD values are 99 and 100,

respectively; the MSE values calculated according to Equ. (3.2) are 99 and 50 respectively; and the WSAD values are 99 and 56.25 respectively. It can be seen that the MSE and WSAD will take the second block as the winner residual block which can be coded with fewer bits since it only contains the DC coefficient, whereas the SAD criterion will pick the first one which requires more bits for the following DCT coding due to the high frequency components in it. Therefore better rate-distortion performance can be expected with the WSAD compared with SAD.

Results comparison on H.263 platform

Firstly, comparative experiments were conducted for block motion estimation using different matching criteria in a practical video codec H.263 [121]. Eight QCIF and CIF video sequences were used in the experiments, which contain different types of motion like slow motion (“Carphone”), medium to fast motion (“Coastguard”, “Dancer”), and camera panning (“Stefan”). The coding pattern IPPPPIPPPP (i.e., one Intra frame followed by four P-frames) was used here. The search range was a square region of $[-15, +15]$ and the quantization parameter was set to 15. The coding results using MSE, SAD and WSAD are compared in terms of PSNR of the reconstructed video sequences and the associated bit rate.

Figure 3-6 demonstrates the rate-distortion comparison for coding two sequences using MSE, SAD and WSAD in H.263 respectively. It is clear that MSE is the best and SAD is the worst, while our WSAD performance is very close to the MSE. For example, for “Foreman” sequence in Figure 3-6(a), our WSAD outperforms SAD around 0.5dB higher at a low bit rate (e.g., 6500 bits per frame) when final reconstruction quality depends more on motion compensation results.

Taking a closer look at the performance difference among the SAD, WSAD and MSE metrics, Figure 3-7 compares the detailed rate-distortion performance on a frame-by-frame basis for “Foreman” sequence, where Figure 3-7(a) shows the reconstructed frame-by-frame PSNR results and Figure 3-7(b) plots the number of bits used to code the corresponding frame in the sequence. Since motion estimation is not performed in I frames, only P frames are drawn. From the two figures, for all the frames, not only does

the proposed WSAD offer a higher PSNR performance than SAD, it also achieves lower bit rate in coding these frames. The reason for the lower coding bit rate is that motion vectors from WSAD result in smaller (in the sense of MSE) or smoother residual errors for ease of subsequent DCT coding as discussed before.

Figure 3-6. Comparison of rate-distortion curves using different metrics in H.263: (a) “Foreman.CIF”; (b) “Coastguard. CIF”.

Figure 3-7 Frame-by-frame rate-distortion comparison using different metrics in H.263 for “Foreman.CIF” sequence: (a) Frame-by-frame reconstructed PSNR results; (b) Number of bits used for coding each frame.

Table 3-IV summarizes the average rate-distortion performance for all the tested sequences using all the different metrics. To highlight the difference between the proposed WSAD and the other metrics, the table lists the difference of PSNR (shown as $\Delta PSNR$) in decibels and the difference of bit rate in percentage (Δbit_rate) with respect to the proposed WSAD metric. For example, $\Delta PSNR_{SAD} = PSNR_{SAD} - PSNR_{WSAD}$ is for SAD compared against WSAD in terms of PSNR, and the coding bit rate difference

$$\Delta bit_rate_{SAD} = (bit_rate_{SAD} - bit_rate_{WSAD}) / bit_rate_{WSAD} \times 100\% \quad (3.22)$$

is listed in the table. It can be seen from the table that MSE performs best while SAD is the worst, as predicted. Compared with the SAD metric, the WSAD produces better rate-distortion performance than SAD with smaller bit rate and slightly higher PSNR for the reconstructed video sequences. For example, in “Akiyo” sequence, SAD requires about 15% more bits than WSAD while its PSNR is still slightly poorer than that for WSAD.

Table 3-IV. Average rate-distortion performance for all P frames in H.263 video coding, benchmarked against WSAD (QP=15)

Sequence		Goldfish	Stefan	Foreman	Dancer	Carphone	BBC	Akiyo	Coastguard
Metric	Δ PSNR	-0.027	-0.005	-0.046	-0.028	-0.012	-0.014	-0.029	-0.023
	Δ BitRate	0.0113	0.0059	0.0951	0.0797	0.0741	0.0447	0.1501	0.0517
MSE	Δ PSNR	0.014	0.004	0.01	0.002	0.018	0.006	0.001	0.012
	Δ BitRate	-0.016	-0.0114	-0.0841	-0.1097	-0.0604	-0.036	-0.0396	-0.0714

The multi-level partial WSAD was also evaluated in H.263 motion estimation. Figure 3-8 shows the frame-by-frame comparison of the coding bit rate (number of bits per frame) for “Flower” sequence using PWSAD_L1 and PWSAD_L2 against WSAD, respectively. As expected, the bit rate increases for the two PWSAD measures compared with the WSAD and the PWSAD_L2 increases more. From the figure, we can see that all the WSAD and the two partial WSAD measures outperform SAD by producing bit rate closer to that from MSE. Table 3-V summarizes the average rate-distortion performance for all the tested sequences using PWSAD_L1 and PWSAD_L2. It can be seen that compared with the result of the SAD metric shown in Table 3-IV, the partial WSAD spends smaller number of bits for a slightly better PSNR for the reconstructed video sequences, which means a better rate-distortion performance over SAD.

Figure 3-8 Frame-by-frame bit comparison based on different metric (including two partial WSAD) in coding “Flower” sequence.

Table 3-V Comparison of average rate-distortion performance using PWSAD on tested sequences

Sequence Metric		Goldfish	Stefan	Foreman	Flower	Dancer	Carphone	BBC	Akiyo	Coastguard
PWSAD_L1	PSNR	-0.014	-0.003	-0.019	-0.004	-0.007	-0.008	-0.009	-0.004	-0.018
	bit rate	43.167	10.3	114.633	148.107	151.88	34.632	175.44	5.9	93.167
PWSAD_L2	PSNR	-0.025	-0.005	-0.04	-0.007	-0.027	-0.01	-0.013	-0.013	-0.02
	bit rate	86.833	16.633	223	336.614	267.68	45.684	364.32	11.9	178.567

Results comparison on H.264 platform

We also evaluated the performance of the WSAD in the motion estimation in H.264 [5] to test its effect on smaller-size block matching based on the H.264 reference software JM 8.5 [122]. The same coding pattern IPPPPIPPPP was used as in H.263. The search range was set to $[-16, +16]$. All the block partition sizes recommended for motion compensation in H.264 (16×16 , 16×8 , 8×16 , 8×8 , 8×4 , 4×8 and 4×4) were enabled. The rate-distortion optimization (RDO) option was turned on. Integer-pixel motion estimation based on different metrics was implemented and sub-pixel motion estimation was disabled. The Lagrangian cost function $J = E + \lambda_{motion} \cdot R_{mv}$ is employed in H.264 motion estimation, where the Lagrange multiplier λ_{motion} is used to balance the distortion E for a residual block and bit cost R_{mv} for the corresponding motion vector. The selection of the

Lagrange multiplier was suggested in [125] as $\lambda_{motion} = \sqrt{\lambda_{mode}}$ for E=SAD and $\lambda_{motion} = \lambda_{mode}$ for E=SSE (sum of squared errors), where λ_{mode} is the Lagrange parameter for macro-block prediction mode selection. Note that the MSE defined in (3.2) is the square root of SSE, both of which are equivalent measurements in the context of motion estimation. While the SSE was implemented in our experiment, in the following experimental results we use MSE to represent SSE. In view that WSAD is smaller than SAD for a given motion vector (thus R_{mv} is fixed), in order to maintain a good balance between the two terms in the Lagrangian cost function, we empirically chose to further reduce the Lagrange multiplier from $\sqrt{\lambda_{mode}}$ (for SAD) to $\sqrt{\lambda_{mode}}/4$ for E=WSAD. There may be some other (better) choices of the Lagrange multiplier for the WSAD to maintain the good balance between the two terms in the cost function.

Figure 3-9 demonstrates the rate-distortion comparison for coding two sequences using SAD and WSAD in H.264 respectively (to avoid clutter of overlaid curves, MSE results are not shown in the figure which almost overlap with WSAD curves). For the two sequence “BBC” sequence, WSAD performs better than SAD. Table 3-VI summarizes the R-D performance difference of other metrics benchmarked against the WSAD for all the tested sequences with QP=30 (a typical value in the middle of H.264 QP range 0-51), to show the difference of WSAD from others. It can be seen that the WSAD outperforms SAD in H.264 video coding. For example, compared with the WSAD based coding, the SAD based coding for the “BBC” sequence degrades PSNR 0.16 dB with slight bit rate increase of 0.38%.

It is noticed that the improvement of WSAD over SAD in H.264 is less than that in H.263. The reasons may lie in the following aspects. Firstly, as we have pointed out in Sub-section II-B, the deviation of SAD from MSE will become larger with more elements in the vector. The use of smaller block sizes in H.264 will reduce the deviation caused by SAD, thus making SAD and MSE/WSAD less different. Secondly, as SAD, MSE and WSAD are all functions measuring the residual block only, their effects on the results of motion estimation will be diluted by the motion vector cost (the second term) used in the H.264 motion estimation cost function, especially for a large QP (corresponding to a low bit rate). Thirdly, the dedicated coding techniques for the residual blocks adopted in

H.264, such as the smaller block-size transform, context-adaptive entropy coding and deblocking filtering, may also compensate the poorer rate-distortion performance of SAD-based motion estimation to some extent. However, although the degradation from SAD criteria is reduced in H.264 due to these reasons, the WSAD still offers better R-D performance than SAD.

Moreover, the experimental results in terms of CPU time are tabulated in Table 3-VII. This experiment was conducted on a PC running Windows XP Professional with Intel Core Duo 1.66G CPU and RAM 965M. No special optimization or fast search technique was applied in the implementation. We also disabled the partial distortion based early termination technique for motion estimation in the JM 8.5 codec for a fair time comparison among different metrics. From the table, we noticed that MSE is slightly faster than our WSAD and also close to SAD in our simulation. The reason behind may be as follows. It is known that the Core Duo CPUs have a special multiplier unit for multiplication operations, which means that more computation resources are used for multiplication operations and thus more computational power is consumed. Therefore, it is unfair to compare the computation complexity of MSE with that of other multiplication-free criterion (SAD, WSAD) based on CPU time alone. To make a fair comparison between MSE and a multiplication-free criterion function, more factors such as the used computation resources and the computation power consumption, in addition to the computation time, need to be considered simultaneously. However it is fair to compare the computation time of the other three multiplication-free criteria in our simulation environment. In hardware implementation, these multiplication-free metrics can be constructed with a simple circuitry, which is particularly advantageous in portable handheld devices where a minimum of circuitry and consequent light weight and low power consumption are great advantages.

Figure 3-9 Rate-distortion curves using H.264 based on different metrics: (a) “BBC.CIF”, (b) “Foreman.CIF”, with frame rate of 30 fps and QP = 20, 25, 30, 35, 40.

Table 3-VI. Average rate-distortion variation from WSAD in H.264 (QP=30)

Sequence		Goldfish	Stefan	Foreman	Dancer	Carphone	BBC	Akiyo	Coastguard
Metric	Δ PSNR	-0.13	-0.06	-0.02	-0.05	-0.01	-0.01	-0.16	-0.06
	Δ BitRate	1.29%	0.23%	0.54%	1.53%	1.99%	1.07%	0.38%	2.01%
MSE	Δ PSNR	0.01	0.03	0	0.04	0.09	0.04	0.05	0.03
	Δ BitRate	-0.41%	-0.29%	-0.25%	-0.60%	-0.51%	-0.31%	-0.17%	-0.41%

Table 3-VII Average Motion Estimation Time for One frame (in seconds)

Sequence	Goldfish	Stefan	Foreman	Dancer	Carphone	BBC	Akiyo	Coastguard
Metric	CIF	QCIF	CIF	CIF	QCIF	CIF	CIF	CIF
SAD	7.197	1.905	7.179	7.183	1.880	7.196	7.202	7.191
MSE	8.602	2.247	8.558	8.580	2.214	8.532	8.597	8.573
WSAD	8.741	2.260	9.000	8.961	2.350	8.752	8.767	8.686

3.5.3. Vector Quantization

Besides applying it in video coding, we also evaluated the proposed WSAD in the vector quantization. Vector quantization (VQ) [56] is a well-known technique for data compression with an acceptable fidelity. A VQ codec consists of two parts: encoder and decoder, which are equipped with an identical codebook. Given a VQ codebook $Y = \{y_i^{(k)} \mid i = 1, 2, \dots, N\}$ in the k -dimensional Euclidean space R^k , for any input vector $x^{(k)}$

$(x_0, x_1, \dots, x_{k-1}) \in R^k$, the encoder aims to find a codeword $y_i^{(k)}$ in the codebook which is the closest to $\mathbf{x}^{(k)}$ based on a distortion metric and the corresponding index i is the output. The decoder uses the index i to retrieve the codeword $y_i^{(k)}$ from the same codebook to reconstruct $\mathbf{x}^{(k)}$.

Here we evaluate the different distortion metrics in VQ encoding. Seven 256-grey-level images of size 256×256 (“Lenna”, “Lady”, “Woman”, “House”, “Camera”, “Walter”, “Pattern”) were tested by decomposing them into 4×4 nonoverlapped blocks to form a set of 28,672 16-dimensional vectors. Another data source of 320,000 16-bit samples from two male speakers sampled at 11.025 kHz was also used in our test. Two codebooks of 1024 codewords were obtained using the minimax partial distortion competitive learning method (MMPDCL) [120] for image and speech respectively, with the training data of “Lenna”, “Lady” and “Woman” for the image codebook and speech samples of the first speaker for the speech codebook. The decoding quality is measured using $\text{SNR} = 10 \log_{10}(\sigma_x^2/\text{MSE})$ where σ_x^2 is the variance of the scalar source sample x for speech and $\text{PSNR} = 10 \log_{10}(255^2/\text{MSE})$ for images.

The experimental results are listed in Table 3-VIII. Similarly, to indicate the difference of WSAD from other metrics, only the performance differences by other criteria benchmarked against our WSAD are tabulated. Herein, compared against the WSAD, the SAD yields a considerable coding degradation with the PSNR drop up to 0.483dB (for “Camera”). On the other hand, PSNR performance based on the WSAD is very close to that of MSE with the maximum drop of 0.053dB only (for “Camera”).

Table 3-VIII. VQ encoding quality (PSNR) comparison of different metrics, benchmarked against WSAD

Test Data ΔPSNR	Lenna	Lady	Woman	House	Camera	Walter	Pattern	speech1	speech2
$\Delta\text{PSNR}_{\text{SAD}}$	-0.246	-0.105	-0.171	-0.244	-0.483	-0.176	-0.386	-0.227	-0.217
$\Delta\text{PSNR}_{\text{MSE}}$	0.027	0.008	0.012	0.021	0.053	0.018	0.043	0.024	0.030

Note that in the VQ application, a given codebook size N implies the same bit rate $\log_2 N$ per vector if the codevector indices are represented in fixed length, which is the normal practice in VQ. Therefore, the better PSNR results listed in the table reveal that

our WSAD achieves the distortion improvement at the same bit-rate, which means that the WSAD still produces better rate-distortion performance than SAD in the VQ coding.

3.6. Summary

The matching criterion function has been known to be critical for block matching accuracy as well as computational complexity. In this chapter, we have theoretically examined the deviation of 2-element SAD from MSE. Based on the thorough analysis, an optimized 2-element WSAD has been proposed, which has been shown to approximate MSE more closely with smaller average as well as maximum deviations while still maintaining multiplication-free property. A simple scheme has been presented to extend 2-element WSAD to multi-element WSAD based on a multi-level pyramid structure. Extensive experiments have been conducted to compare the WSAD with the SAD and MSE in video coding and vector quantization applications. The results have substantively validated the superiority of the WSAD over the SAD.

Chapter 4.

Transform-Exempt Calculation of Sum of Absolute Hadamard Transformed Differences (SATD)

4.1. Introduction

Besides the MSE and SAD discussed in the preceding chapter, the latest video coding standard H.264/AVC coder utilizes several advanced coding techniques to attain significantly higher compression ratio than the previous video coding standards, among which the rate-distortion optimization (RDO), the procedure conducted to select the best coding mode out from all possible modes in both the intra- and inter-prediction, is considered to be one of the most important factors that contribute to the success of H.264/AVC in terms of compression ratio and visual quality. Nevertheless, this technique also increases the computational complexity remarkably. In order to select the best coding mode, RDO requires calculation of the rate-distortion cost for all possible modes exhaustively, which involves computation of integer transform, quantization, variable length coding to obtain the bit rate cost through the encoding process and the corresponding inverse operations to obtain reconstructed block distortion through the decoding process. Therefore, to achieve low complexity in H.264/AVC reference software [5][122] uses the sum of absolute differences (SAD) to estimate the rate-distortion cost instead of calculating its exact value through the true encoding/decoding processes, thus lowering the computation burden. However, the coding performance degradation by using SAD is non-negligible in terms of both bit rate and reconstructed

video quality. Hence, another metric, the sum of absolute Hadamard-transformed differences (SATD), has been proposed to be an alternative to SAD to improve the coding performance in the H.264/AVC reference software. Furthermore, SATD is also considered to be applied in sub-pel motion estimation following the full-pel motion estimation based on SAD, for a better estimation in the rate-distortion sense.

SATD is normally calculated in the transformed domain by converting a block difference with Hadamard-transform. More specifically, the SATD computation comprises of two steps, i.e., transforming the block difference and then performing the SAD operation for the transformed elements. Although it usually achieves better coding performance than SAD, the sophisticated SATD requires more computation operations due to the Hadamard-transform involved. For example, the SATD requires 96 more additions/subtractions for transforming a 4×4 block. Therefore, with the two separate steps involved in the SATD calculation, fast algorithms are mainly to reduce the computation by individually speeding up either one step, such as the algorithms applying fast Hadamard transform to accelerate SATD calculation in [123][124]. However, 64 more additions are still required in the fast Hadamard transform [124]. In this chapter, we examine the joint effect of both steps involved in the 2-point SATD calculation in the first instance, based on which we derive a simpler way to directly calculate the SATD in the original domain without performing the Hadamard transform. It is thus termed as transform-exempt SATD (TE-SATD) calculation. We then derive the 4-point and 4×4 block fast SATD calculation based on the 2-point TE-SATD scheme, where two methods are presented for 4×4 block fast SATD calculation. For 4×4 block SATD calculation, both the proposed methods can save 38% and 17% computation operations compared with the conventional SATD calculation and the fast Hadamard transform based SATD method, respectively, while producing exactly the same coding performance. We also test the computation efficiency of the TE-SATD based fast method on the H.264/AVC platform, which uses 4×4 block SATD in the rate-distortion cost estimation.

The remainder of the chapter is organized as follows. Section 4.2 reviews the distortion metrics especially the SATD used in H.264/AVC. In section 4.3, we derive the proposed TE-SATD fast scheme and discuss computation savings comparatively for 2-

point, 4-point and 4×4 block SATD calculation. Comparative simulation results and conclusion are given in Section 4.4 and Section 4.5, respectively.

4.2. Review of Distortion Metrics in H.264/AVC

In the following, we will review different distortion measurements used for coding mode selection in H.264/AVC, where mode selection is crucial to rate-distortion efficiency. Consider a difference block of size $N \times N$, $\mathbf{D}_{N \times N}$, which is obtained by subtracting a reference block from the current block in the context of video coding. Then the sum of squared difference (SSD) for $\mathbf{D}_{N \times N}$ can be calculated as

$$SSD(\mathbf{D}_{N \times N}) = \sum_{i=1}^N \sum_{j=1}^N (d_{ij})^2 \quad (4.1)$$

where d_{ij} denotes the (i, j) -th element of $\mathbf{D}_{N \times N}$. In H.264/AVC, the Lagrangian cost function based on SSD is employed for mode selection as

$$J_{SSD} = SSD + \lambda \cdot R \quad (4.2)$$

where the Lagrange multiplier, λ , is used to trade off the distortion SSD and bit cost R . The SSD is a widely accepted metric to measure the distortion in that it directly determines performance indicators such as the peak signal to noise ratio (PSNR) used to measure the reconstruction quality in image and video coding. However, multiplication (square) operations are involved in the SSD calculation, which demand comparatively high computation power or complex hardware implementation, limiting its affordability especially in resource constrained environment such as handheld portable devices with low computation power. Furthermore, in H.264/AVC rate-distortion optimized (RDO) mode selection, it is found that the computation of SSD is very time-consuming as it is based on the reconstructed reference block, which requires all the coding/decoding operations (such as forward/inverse transformation, quantization/dequantization, variable length coding and so on) to obtain the actual reconstructed block and bit cost.

Alternatively, the H.264/AVC reference software JM [122] opts to use sum of absolute difference (SAD) as a substitute by supplanting the multiplication with the

absolute operation while using the predicted block instead of the reconstructed block for the distortion calculation. The SAD for the difference block can be calculated as

$$SAD(\mathbf{D}_{N \times N}) = \sum_{i=1}^N \sum_{j=1}^N |d_{ij}| \quad (4.3)$$

and the Lagrangian cost function based on SAD becomes

$$J_{SAD} = SAD + \lambda_{SAD} \cdot R_{est} \quad (4.4)$$

where λ_{SAD} is the Lagrange multiplier for SAD metric which was selected empirically [125] to keep a good balance between the two terms in the Lagrangian cost function, and R_{est} is the estimated bit cost which is assigned to be $4K$ when employing J_{SAD} for intra 4×4 mode selection with the penalty K taking 0 for the most probable mode or 1 for other modes.

The SAD metric and the associated cost function could save a lot of computations. However, this will degrade the coding performance in both bit rate and reconstructed video quality not insignificantly [126]. To provide a tradeoff solution between SAD (simple but with relatively poor performance) and SSD (best performance but with high complexity), the JM software suggests an optional sum of absolute transformed differences (SATD) based cost function

$$J_{SATD} = SATD + \lambda_{SATD} \cdot R_{est} \quad (4.5)$$

where λ_{SATD} and R_{est} are the same as those counterparts for SAD metric. The SATD is given as

$$SATD(\mathbf{D}_{N \times N}) = \sum_{i=1}^4 \sum_{j=1}^4 |t_{ij}| \quad (4.6)$$

where t_{ij} are the (i, j) -th element of the following 2-dimensional Hadamard transformed block:

$$\mathbf{T}_{4 \times 4} = T(\mathbf{D}_{4 \times 4}) = \mathbf{H}_{4 \times 4}^T \cdot \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4} \quad (4.7)$$

with $\mathbf{H}_{4 \times 4}$ being a 4×4 Hadamard matrix. For any integer $n > 1$, the Hadamard matrix of $2n \times 2n$ is given as

$$\mathbf{H}_{2n \times 2n} = \mathbf{H}_{2 \times 2} \otimes \mathbf{H}_{n \times n} \quad (4.8)$$

where $\otimes$ denotes the Kronecker product and

$$\mathbf{H}_{2 \times 2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.9)$$

Therefore we can obtain

$$\mathbf{H}_{4 \times 4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad (4.10)$$

It can be seen that the Hadamard matrices are symmetric, i.e., $\mathbf{H}_{2n \times 2n}^T = \mathbf{H}_{2n \times 2n}$.

It is found that the SATD can normally produce better performance than SAD. In the SATD measurement, the Hadamard transform is used for the prediction error block, which closely imitates the integer cosine transform (ICT) applied to prediction error in the coding process of H.264/AVC. Two out of four row vectors (base functions) in the 4×4 Hadamard matrix are equal to those of 4×4 ICT matrix in the H.264/AVC encoder, while the other two row vectors have the similar characteristics as the ICT ones. Therefore, when applying SATD to measure the prediction distortion, the frequency characteristics of the prediction error are nicely emulated, thus providing a better rate-distortion cost estimation for mode selection. However, the Hadamard transform involved in the SATD calculation requires 96 more additions for a 4×4 block transform than the SAD computing, or 64 more additions even when fast Hadamard transform is applied [124]. It is therefore much more desirable if we can further reduce the computational complexity of SATD calculation while maintaining its good coding performance. In the next section, a fast algorithm will be introduced to accelerate the SATD computation by exempting its calculation from Hadamard transform.

4.3. Transform-Exempt SATD Calculation

For notational simplicity, we define an operator $SA(\bullet)$ to sum up the absolute elements of an $M \times N$ matrix $\mathbf{D}_{M \times N}$ as

$$SA(\mathbf{D}_{M \times N}) = \sum_{i=1}^M \sum_{j=1}^N |d_{ij}| \quad (4.11)$$

where d_{ij} denotes the (i, j) -th element of $\mathbf{D}_{M \times N}$. Thus, the SAD defined in Equation (4.3) and the SATD defined in Equation (4.6) can be expressed as

$$SAD(\mathbf{D}_{N \times N}) = SA(\mathbf{D}_{N \times N}) \quad (4.12)$$

$$SATD(\mathbf{D}_{N \times N}) = SA(T(\mathbf{D}_{N \times N})) \quad (4.13)$$

We can see that SAD and SATD are calculated with the same operator but in different domains, i.e., one in the spatial domain and the other in the transform domain. Based on the definition in Equation (4.11), it is straightforward to obtain the following properties of the operator $SA(\bullet)$.

Property 1. For a matrix $\mathbf{D}$ comprising N partitioned sub-matrices $\mathbf{D}_i$, $i=1, 2, \dots, N$, the following holds

$$SA(\mathbf{D}) = \sum_{i=1}^N SA(\mathbf{D}_i) \quad (4.14)$$

Property 2. Denote a vector stacking operator for a matrix $\mathbf{D}_{M \times N}$ as

$\overrightarrow{\mathbf{D}_{M \times N}} = [d_{11} \ d_{12} \ d_{13} \ \dots \ d_{1N} \ d_{21} \ d_{22} \ \dots \ d_{MN}]^T$, we have

$$SA(\overrightarrow{\mathbf{D}_{M \times N}}) = SA(\mathbf{D}_{M \times N}) \quad (4.15)$$

In the following, we will develop the fast algorithms for SATD calculation in steps, which features the exemption of Hadamard transform, therefore known as transform-exempt SATD (TE-SATD) calculation.

4.3.1. TE-SATD Calculation for 2-Point SATD

For a 2-point signal $\mathbf{D}_{2 \times 1} = [d_1 \ d_2]^T$, the conventional SATD calculation firstly converts the signal into the transformed domain using the one-dimensional Hadamard transform, which is computed as

$$\mathbf{T}_{2 \times 1} = T(\mathbf{D}_{2 \times 1}) = \mathbf{H}_{2 \times 2} \cdot \mathbf{D}_{2 \times 1} = [d_1 + d_2 \ d_1 - d_2]^T \quad (4.16)$$

Then the second step is to perform the $SA(\bullet)$ operation over the transformed signal $\mathbf{T}_{2 \times 1}$ to eventually obtain the SATD for $\mathbf{D}_{2 \times 1}$, that is

$$SATD(\mathbf{D}_{2 \times 1}) = SA(\mathbf{T}_{2 \times 1}) = |t_1| + |t_2| \quad (4.17)$$

where $\mathbf{T}_{2 \times 1} = [t_1 \ t_2]^T = [d_1 + d_2 \ d_1 - d_2]^T$ is the transformed signal. It can be seen that the above one-dimensional Hadamard transform in (4.16) requires two addition/subtraction operations (no fast algorithm in this case), and the $SA(\bullet)$ operation in (4.17) needs another two absolute operations and one addition. Therefore, performing the two steps separately, the conventional SATD calculation requires totally two absolute and three addition/subtraction operations even for a 2-point signal. Combining the two steps of processing to explore their joint effect, we can find a new and simplified way of calculating the SATD.

We consider the unit distance trajectories of SAD and SATD shown in Figure 4-1. In the original coordinate system (d_1, d_2) , the trajectory of unit SAD is a rotated square with 45 degrees (shown in the solid line), which means all the points lying on the four sides of the rotated square have the constant SAD value of $|d_1| + |d_2| = 1$. Due to orthogonal property of the normalized Hadamard transform, the effect of Hadamard transform is to rotate the original spatial coordinate by -45 degrees (without considering the scaling effect), which is shown as (d'_1, d'_2) in the figure. Therefore the SATD unit distance trajectory is a corresponding SAD-based trajectory from the perspective of the rotated coordinate system, i.e., the rotated square indicated as the dashed square with respect to (d'_1, d'_2) . However, this trajectory turns out to be an upright square if it is viewed in the original system (d_1, d_2) . Interestingly, this upright square is exactly the trajectory for the function $\max(|d_1|, |d_2|)$ in the original system (d_1, d_2) , which suggests that the SATD for (d_1, d_2) can be directly calculated in the original domain using $\max(|d_1|, |d_2|)$ without performing any transform (note that the scaling weight is not considered). Based on this finding, a transform-exempt 2-point SATD calculation scheme can be achieved as follows.

Figure 4-1 Geometric comparison of unit distance trajectories of SAD and SATD.

Theorem 1. The SATD for $\mathbf{D}_{2 \times 1} = [d_1 \ d_2]^T$ can be calculated as $2 \cdot \max(|d_1|, |d_2|)$.

Proof: Applying (4.16) into (4.13), the SATD for $\mathbf{D}_{1 \times 2}$ is

$$\begin{aligned} SATD(\mathbf{D}_{2 \times 1}) &= SA(T(\mathbf{D}_{2 \times 1})) = SA(\mathbf{H}_{2 \times 2} \cdot \mathbf{D}_{2 \times 1}) \\ &= |d_1 + d_2| + |d_1 - d_2| = 2 \cdot \max(|d_1|, |d_2|) \end{aligned} \quad (4.18)$$

Proof completed. ■

This theorem gives a fast algorithm, transform-exempt SATD or TE-SATD, to compute the SATD for a 2×1 block without performing the Hadamard transform. Compared with the conventional 2-point SATD calculation scheme, the proposed TE-SATD can save a substantial amount of computation as summarized in the following.

- The TE-SATD requires only two absolute operations and one comparison for taking the maximum. The weight of 2 can be implemented by a shift operation and the scaling constant can be actually ignored without affecting the SATD-based comparison result.
- In the conventional SATD calculation, the Hadamard transform needs two addition/subtraction operations, and the $SA(\bullet)$ operation over the transform block demands additional two absolute operations and one addition.

In order to facilitate the comparison of computation complexity, we conducted the following experiment. We performed the absolute, addition, subtraction, and comparison operations for 100 million times respectively on a PC with Intel 1.70G CPU, which took 390, 380, 371, and 371 ms, respectively. Note that in the experiment, the absolute operation was implemented using lookup table (the same way as in H.264/AVC reference software JM 8.5). Therefore, we can simply treat these operations as addition-equivalent operations. From the above analysis for the 2-point SATD computation, we can see that the TE-SATD calculation scheme needs totally 3 addition-equivalent operations versus 5 addition-equivalent operations in the conventional SATD calculation, thus saving $(5-3)/5$ (=40%) computation.

4.3.2. TE-SATD Calculation for 4-Point SATD

The above 2×1 TE-SATD calculation can be extended to 4×1 blocks with the following theorem.

Theorem 2. The SATD for a 4-point signal $\mathbf{D}_{4 \times 1} = [d_1 \ d_2 \ d_3 \ d_4]^T$ can be calculated as

$$SATD(\mathbf{D}_{4 \times 1}) = 2 \cdot [\max(|d_1 + d_3|, |d_2 + d_4|) + \max(|d_1 - d_3|, |d_2 - d_4|)] \quad (4.19)$$

Proof: The one-dimensional Hadamard transform of $\mathbf{D}_{4 \times 1} = [d_1 \ d_2 \ d_3 \ d_4]^T$ is

$T(\mathbf{D}_{4 \times 1}) = \mathbf{H}_{4 \times 4} \cdot \mathbf{D}_{4 \times 1}$. Then the SATD for $\mathbf{D}_{4 \times 1}$ is

$$SATD(\mathbf{D}_{4 \times 1}) = SA(T(\mathbf{D}_{4 \times 1})) = SA(\mathbf{H}_{4 \times 4} \cdot \mathbf{D}_{4 \times 1}) \quad (4.20)$$

Rewriting Equation (4.20) in partitioned matrix format and applying the *Property 1* of the $SA(\bullet)$ operator, we have

$$\begin{aligned} SATD(\mathbf{D}_{4 \times 1}) &= SA(\mathbf{H}_{4 \times 4} \cdot \mathbf{D}_{4 \times 1}) \\ &= SA\left(\begin{bmatrix} \mathbf{H}_{2 \times 2} & \mathbf{H}_{2 \times 2} \\ \mathbf{H}_{2 \times 2} & -\mathbf{H}_{2 \times 2} \end{bmatrix} \begin{bmatrix} \mathbf{D}_{1,2} \\ \mathbf{D}_{3,4} \end{bmatrix}\right) = SA\left(\begin{bmatrix} \mathbf{H}_{2 \times 2}(\mathbf{D}_{1,2} + \mathbf{D}_{3,4}) \\ \mathbf{H}_{2 \times 2}(\mathbf{D}_{1,2} - \mathbf{D}_{3,4}) \end{bmatrix}\right) \\ &= SA(\mathbf{H}_{2 \times 2}(\mathbf{D}_{1,2} + \mathbf{D}_{3,4})) + SA(\mathbf{H}_{2 \times 2}(\mathbf{D}_{1,2} - \mathbf{D}_{3,4})) \end{aligned} \quad (4.21)$$

where $\mathbf{D}_{1,2} = [d_1 \ d_2]^T$ and $\mathbf{D}_{3,4} = [d_3 \ d_4]^T$ are the sub-matrices of $\mathbf{D}_{4 \times 1}$. In the final step of the above equation, the SATD for the 4×1 block has been converted to two individual

2×1 SATD calculations for $(\mathbf{D}_{1,2} + \mathbf{D}_{3,4})$ and $(\mathbf{D}_{1,2} - \mathbf{D}_{3,4})$, respectively. Therefore, applying Theorem 1 to these two 2×1 SATD calculations, we obtain $SATD(\mathbf{D}_{4 \times 1}) = 2 \cdot [\max(|d_1 + d_3|, |d_2 + d_4|) + \max(|d_1 - d_3|, |d_2 - d_4|)]$. ■

The computation complexity for 4×1 block SATD calculation is considered comparatively in the following.

- With the transform-exempt SATD calculation shown in Equ. (4.19), the total number of operations required is 11 (not counting the weighting of 2).
- In contrast, with the conventional SATD computing, 12 additions/subtractions are needed for Hadamard transform and 7 operations (4 absolute operation and 3 additions) for the following $SA(\bullet)$ operator, totaling 19 operations. We can see that 42% computation saving can be obtained with the TE-SATD for 4×1 signals.

For SATD calculation of 4×1 signals, fast Hadamard transform is applicable. It is known that the fast Hadamard transform for N -point block requires $N \log_2 N$ subtraction/addition operations (in the one-dimension case) by using the butterfly structure. Therefore, fast Hadamard transform for 4×1 signals requires 8 operations, plus 7 operations for the $SA(\bullet)$ operator thus totaling 15 operations for the SATD calculation. It can be seen that the proposed TE-SATD can still save $(15-11)/15=27\%$ computation when compared to the fast Hadamard transform based SATD calculation.

4.3.3. TE-SATD Calculation for 4×4 block SATD

In the H.264/AVC reference software, two dimensional 4×4 block Hadamard transform based SATD is employed. The SATD for such a 4×4 block $\mathbf{D}_{4 \times 4}$ is

$$SATD(\mathbf{D}_{4 \times 4}) = SA(T(\mathbf{D}_{4 \times 4})) = SA(\mathbf{H}_{4 \times 4}^T \cdot \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4}) \quad (4.22)$$

Two fast methods to calculate this SATD for 4×4 blocks are developed based on the above TE-SATD for 4×1 and 2×1 blocks respectively.

Method 1: 4×1 TE-SATD based fast 4×4 SATD Calculation

This method is developed based on the TE-SATD calculation for 4×1 blocks introduced in the preceding sub-section. Firstly, we perform a one-dimensional Hadamard transform for $\mathbf{D}_{4 \times 4}$, i.e., computing $\mathbf{P}_{4 \times 4} = \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4}$ or $\mathbf{P}_{4 \times 4} = \mathbf{H}_{4 \times 4}^T \cdot \mathbf{D}_{4 \times 4}$. In the following, we consider $\mathbf{P}_{4 \times 4} = \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4}$ (the same development can be obtained for $\mathbf{P}_{4 \times 4} = \mathbf{H}_{4 \times 4}^T \cdot \mathbf{D}_{4 \times 4}$). Hence, we have

$$\text{SATD}(\mathbf{D}_{4 \times 4}) = \text{SA}(T(\mathbf{D}_{4 \times 4})) = \text{SA}(\mathbf{H}_{4 \times 4}^T \cdot \mathbf{P}_{4 \times 4}) \quad (4.23)$$

Based on Property 1 of the $\text{SA}(\bullet)$ operator and the symmetric property of Hadamard matrix, we obtain

$$\text{SATD}(\mathbf{D}_{4 \times 4}) = \text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_1) + \text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_2) + \text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_3) + \text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_4) \quad (4.24)$$

where $\mathbf{P}_i = [p_{i1} \ p_{i2} \ p_{i3} \ p_{i4}]$, $i=1, 2, 3, 4$, is the i -th row vector in $\mathbf{P}_{4 \times 4}$, with p_{ij} being the i -th row, j -th column element in $\mathbf{P}_{4 \times 4}$. Each $\text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_i)$ can be computed using the 4×1 fast TE-SATD in Theorem 2. Therefore, we can achieve the fast SATD calculation method for two dimensional 4×4 blocks based on the 4×1 TE-SATD, which can save a considerable amount of computation as discussed in the following.

- We firstly study the computation complexity for the conventional 4×4 SATD calculation, in which the 2-dimensional 4×4 Hadamard transform involved requires 96 operations and the following $\text{SA}(\bullet)$ operation over the 4×4 transformed matrix needs additional 31 operations (16 absolute operations and 15 additions). Thus, the conventional SATD calculation for a 4×4 block requires $96+31=127$ operations in total.
- If using fast Hadamard transform, the transform step costs 64 operations [124], thus totaling $64+31=95$ operations for the SATD calculation based on the fast Hadamard transform.
- In the proposed method, the partial Hadamard transform still needs 32 operations when using fast Hadamard transform. Then the 4×1 TE-SATD based calculation of $\text{SA}(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_i)$, $i=1, 2, 3, 4$, in Equ. (4.24) will cost totally

11×4 operations. Finally, three additions are used to sum up the four $SA(\mathbf{H}_{4 \times 4} \cdot \mathbf{P}_i)$. Therefore, the proposed method demands 79 operations in total.

Compared to using conventional Hadamard transform, the proposed Method 1 can save up to 48 operations for 4×4 block SATD calculation or equivalently around 38% computation saving, while compared to using fast Hadamard transform, 16 operations can be saved amounting to about 17% computation saving.

Method 2: 2×1 TE-SATD based fast 4×4 SATD Calculation

An important result is introduced before we develop another fast method for 4×4 block SATD calculation.

Theorem 3. *The SATD for a 4×4 block $\mathbf{D}_{4 \times 4}$ can be computed as $SATD(\mathbf{D}_{4 \times 4}) = SA(\mathbf{H}_{16 \times 16} \overrightarrow{\mathbf{D}_{4 \times 4}})$.*

Proof: Applying Property 2 of the $SA(\bullet)$ operator, we have

$$\begin{aligned} SATD(\mathbf{D}_{4 \times 4}) &= SA(\mathbf{H}_{4 \times 4} \cdot \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4}) \\ &= SA(\overrightarrow{\mathbf{H}_{4 \times 4} \cdot \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4}}) \end{aligned} \quad (4.25)$$

It is known that the vector stacking operator has the following property [127]:

$$\overrightarrow{\mathbf{A}_{N \times N} \cdot \mathbf{B}_{N \times N} \cdot \mathbf{C}_{N \times N}} = (\mathbf{C}_{N \times N}^T \otimes \mathbf{A}_{N \times N}) \cdot \overrightarrow{\mathbf{B}_{N \times N}} \quad (4.26)$$

With the above equation and the symmetric property of Hadamard matrices

$\mathbf{H}_{2n \times 2n}^T = \mathbf{H}_{2n \times 2n}$, we obtain

$$\begin{aligned} SATD &= SA(\overrightarrow{(\mathbf{H}_{4 \times 4} \otimes \mathbf{H}_{4 \times 4}) \mathbf{D}_{4 \times 4}}) \\ &= SA(\mathbf{H}_{16 \times 16} \overrightarrow{\mathbf{D}_{4 \times 4}}) \end{aligned} \quad (4.27)$$

Proof completed. ▀

Based on the theorem, we can achieve fast 4×4 SATD calculation using 2×1 TE-SATD scheme in the following. We firstly arrange $\mathbf{H}_{16 \times 16}$ and $\overrightarrow{\mathbf{D}_{4 \times 4}}$ into partitioned sub-matrices of 2×2 and partitioned sub-matrices (sub-vectors) of 2×1, respectively.

Performing the partitioned sub-matrices multiplication and applying *Property 1* of $SA(\bullet)$ operator, we can get

$$\begin{aligned}
 SATD &= SA \left(\begin{bmatrix} H & H & H & H & H & H & H & H \\ H & -H & H & -H & H & -H & H & -H \\ H & H & -H & -H & H & H & -H & -H \\ H & -H & -H & H & H & -H & -H & H \\ H & H & H & H & -H & -H & -H & -H \\ H & -H & H & -H & -H & H & -H & H \\ H & H & -H & -H & -H & H & -H & H \\ H & -H & H & -H & -H & H & H & -H \end{bmatrix}_{16 \times 16} \begin{bmatrix} D_{11,12} \\ D_{13,14} \\ D_{21,22} \\ D_{23,24} \\ D_{31,32} \\ D_{33,34} \\ D_{41,42} \\ D_{43,44} \end{bmatrix}_{1 \times 16} \right) \\
 &= SA(H(D_{11,12} + D_{13,14} + D_{21,22} + D_{23,24} + D_{31,32} + D_{33,34} + D_{41,42} + D_{43,44})) + \\
 &\quad SA(H(D_{11,12} - D_{13,14} + D_{21,22} - D_{23,24} + D_{31,32} - D_{33,34} + D_{41,42} - D_{43,44})) + \\
 &\quad SA(H(D_{11,12} + D_{13,14} - D_{21,22} - D_{23,24} + D_{31,32} + D_{33,34} - D_{41,42} - D_{43,44})) + \\
 &\quad SA(H(D_{11,12} - D_{13,14} - D_{21,22} + D_{23,24} + D_{31,32} - D_{33,34} - D_{41,42} + D_{43,44})) + \\
 &\quad SA(H(D_{11,12} + D_{13,14} + D_{21,22} + D_{23,24} - D_{31,32} - D_{33,34} - D_{41,42} - D_{43,44})) + \\
 &\quad SA(H(D_{11,12} - D_{13,14} + D_{21,22} - D_{23,24} - D_{31,32} + D_{33,34} - D_{41,42} + D_{43,44})) + \\
 &\quad SA(H(D_{11,12} + D_{13,14} - D_{21,22} - D_{23,24} - D_{31,32} - D_{33,34} + D_{41,42} + D_{43,44})) + \\
 &\quad SA(H(D_{11,12} - D_{13,14} - D_{21,22} + D_{23,24} - D_{31,32} + D_{33,34} + D_{41,42} - D_{43,44}))
 \end{aligned} \tag{4.28}$$

where $H = H_{2 \times 2}$, and $D_{ij,mk} = [d_{ij} \ d_{mk}]^T$ with d_{ij} being the i -th row, j -th column element of $D_{4 \times 4}$. It can be found that each $SA(\bullet)$ in Equ. (4.28) is actually a 2×1 SATD calculation. Therefore, the computing of 4×4 SATD is translated into a sum of eight 2×1 SATD calculations. Then we can apply the fast 2×1 TE-SATD algorithm to accelerate the SATD calculation for 4×4 blocks.

Now we discuss the computational complexity of this method. Firstly we need to perform the processing (addition/subtraction) of the eight partitioned 2×1 sub-vectors of $D_{4 \times 4}$, shown in Equation (4.28). It can be found that the processing of the eight 2×1 sub-vectors is a 8-point sub-vector-based Hadamard transform, where one point refers to a 2×1 sub-vector instead of a scalar. Therefore, we can still apply the same butterfly structure for fast Hadamard transform to obtain sub-vector-based Hadamard transform, which requires totally 48 operations (24×2 , 24 is the operation number for 8-point fast Hadamard transform) considering that one point here is a 2×1 sub-vector containing 2

elements. Then eight 2×1 TE-SATD calculations are needed, each of which requires three operations (one comparison plus two absolute operations). Another seven additions are used to sum up the eight 2×1 TE-SATD results to obtain the final 4×4 SATD value. Thus, this TE-SATD method for 4×4 blocks requires totally 79 ($= 48 + 3 \times 8 + 7$) operations, same as that by the proposed Method 1.

To summarize the computation complexity of the different SATD calculation methods for blocks of 2×1 , 4×1 and 4×4 , Table 4-I tabulates the required operation numbers for SATD computation using the conventional Hadamard transform, fast Hadamard transform and TE-SATD, respectively. It can be seen that the proposed TE-SATD can save noticeable computation operations for these different sizes of block SATD calculation.

Table 4-I. Computational comparison for conventional, fast transform based and TE-SATD based SATD calculations: “*add*”, “*abs*” and “*cmp*” refer to addition/subtraction, absolute and comparison operations, respectively

Method \ Block	2×1	4×1	4×4
Conventional	5 (3 <i>add</i> + 2 <i>abs</i>)	19 (15 <i>add</i> + 4 <i>abs</i>)	127 (111 <i>add</i> + 16 <i>abs</i>)
Fast-Trans.	-	15 (11 <i>add</i> + 4 <i>abs</i>)	95 (79 <i>add</i> + 16 <i>abs</i>)
TE-SATD	3 (2 <i>abs</i> + 1 <i>cmp</i>)	11 (5 <i>add</i> + 4 <i>abs</i> + 2 <i>cmp</i>)	79 (55 <i>add</i> + 16 <i>abs</i> + 8 <i>cmp</i>)

4.4. Experimental Results

Comparative experiments were conducted by using SAD and SATD with fast-transform based and TE-SATD based calculations in H.264 video coding [5][122], where the SATD (or SAD) is based on 4×4 blocks. Considering that the proposed two TE-SATD based methods for 4×4 blocks are equivalent in computation complexity, we only tested the first method in this experiment. The coding results using SAD, conventional SATD (without fast Hadamard transform), fast-transform based SATD and TE-SATD based SATD are compared in terms of average PSNR of the reconstructed video sequences, the associated bit rate and the total computation time for SAD/SATD calculation, in Table 4-II. Eight video sequences (6 CIF and 2 QCIF) were considered and the first 80 frames of each sequence were tested. The coding structure of IPPPPIPPPP (i.e., one Intra frame followed by four P-frames) was used in the experiment. The search range was a square region of $[-8, +8]$ and the quantization parameter (QP) was set to 28 (a

typical value around the middle of the H.264 QP range from 0 to 51). All the block partition sizes recommended in H.264 for motion compensation, including 16×16 , 16×8 , 8×16 , 8×8 , 8×4 , 4×8 and 4×4 , were enabled with the fast motion estimation disabled and the number of reference frames set to 5. The rate-distortion optimization (RDO) option was turned on. The experiments were conducted on a laptop of Intel Pentium Mobile 1.7GHz CPU and 965MB memory running Microsoft Windows XP operating system.

From the table, it can be seen that the coding performance (PSNR and bit rate) for the conventional SATD, the fast-transform based SATD and TE-SATD are exactly the same, which is better than that of SAD. The TE-SATD based calculation is consistently faster than both the conventional SATD and the fast-transform based SATD with less computation time for all the eight testing sequences. Specifically, the average speedup gains (in percentage) of the proposed TE-SATD are 39.8% and 22.0% compared to the conventional SATD and the fast transform based SATD, respectively.

Table 4-II Comparison of SAD, Fast-Transform based SATD and TE-SATD based SATD in H.264/AVC (Intra Frame Only, time in seconds)

Metric Sequence	SAD			Conventional SATD			Fast-Transform Based SATD			TE-SATD		
	Time	PSNR	Bit Rate	Time	PSNR	Bit Rate	Time	PSNR	Bit Rate	Time	PSNR	Bit Rate
Foreman.cif	38.55	35.70	300.24	154.04	36.09	300.59	121.90	36.09	300.59	93.34	36.09	300.59
Coastguard.cif	41.33	32.99	674.83	161.87	33.20	686.01	125.70	33.20	686.01	96.74	33.20	686.01
Dancer.cif	40.40	38.47	334.89	155.29	38.83	331.82	120.78	38.83	331.82	92.84	38.83	331.82
Goldfish.cif	26.76	38.10	392.69	108.35	38.41	388.99	84.22	38.41	388.99	58.40	38.41	388.99
BBC.cif	41.17	33.69	757.82	155.72	33.95	752.05	122.74	33.95	752.05	92.99	33.95	752.05
Akio.cif	14.53	42.00	106.90	61.65	42.26	106.24	45.32	42.26	106.24	37.34	42.26	106.24
Carphone.qcif	5.63	35.35	126.84	21.91	35.70	126.86	16.96	35.70	126.86	13.74	35.70	126.86
Stefan.qcif	8.88	32.60	242.72	37.73	32.76	242.96	28.81	32.76	242.96	24.46	32.76	242.96

4.5. Summary and Discussion

It has been known that SATD is a better distortion metric than SAD, which can normally produce better coding performance but at the cost of higher computational cost. Inspired by the geometric interpretation for the simplest 2-point SATD calculation, we have developed the simplified transform-exempt SATD (TE-SATD) calculation scheme by considering the joint result of the Hadamard transform and the following SAD processing. We have then extended the 2-point TE-SATD scheme to the 4-point and 4×4 block fast SATD calculation, which have exhibited speedup gains over conventional and

fast transform based SATD calculation methods. Note that in H.264/AVC reference software, SATD is also applied in sub-pixel motion estimation when full-pixel motion estimation is completed based on SAD. Therefore, the proposed TE-SATD based algorithms can also be applied therein.

After we have completed our work and writing, we then came across the patent [128] on fast SATD estimation which was publicized very recently. Therefore, we add in the following paragraph for a discussion. Firstly, we would like to highlight that this chapter is our totally independent work without having any knowledge of the patent until we have completed the work and writing. We started the research originally by comparing a few distortion metrics including SAD and SATD, and inquiring some more fundamental reasons why SATD normally (not definitely) outperforms SAD. We then found that the geometric relationship between SAD and STAD in the unit distance trajectory shown in Fig. 1, which clearly shows the 2-point SATD can be obtained as the maximum of two elements in the original domain without performing the transform. Inspired by this, we obtained the simplified 2-point (2×1 block) transform-exempt SATD (TE-SATD) calculation shown in Theorem 1. We then extended to the simplified 4-point (4×1 block) TE-SATD calculation in Theorem 2. With the two theorems, we have derived mathematically two methods to speed up the 4×4 block SATD calculation based on the two TE-SATD schemes respectively. In short, our progressive path is developing a transform-exempt SATD calculation scheme for 2×1 blocks first, then extending to one-dimensional Hadamard transform based 4-point SATD calculation and finally to two-dimensional Hadamard transform based 4×4 block SATD calculation. In the patent [128], the inventor described a fast SATD estimation method that can achieve the same computation saving as ours. The invention is to eliminate a recursion operation (a butterfly step) in the fast Hadamard transform (FHT) butterfly architecture for the SATD calculation. More specifically, the patented method is to perform butterfly steps in the FHT structure except the final step that can be saved by applying the equation $|a+b|+|a-b|=2 \cdot \max(|a|, |b|)$ to every two neighboring units for obtaining the SATD directly. It can be seen that although equivalent results are obtained, our analytical approach and development methods are remarkably different from the patented one.

Chapter 5.

Efficient Block Matching Using Multi-level Intra- and Inter-Subblock Features: Subblock-Based SATD

5.1. Introduction

From Equation (2.4) in Chapter 2, it can be noted that the distortion measure (or similarity measure) consists of two parts, that is, the function itself and the block representation input as argument to it. Therefore, besides optimizing the measure function discussed in previous chapters, the distortion measurement can also be accelerated by introducing efficient block representation for certain distortion measure function. As discussed in Chapter 3, because of its multiplication-free property, sum of absolute difference (SAD) is the commonly used distortion measure. However, the conventional SAD is calculated based on all the pixel intensities inside the block, thus still demanding high computation load especially when the full search strategy, which examines exhaustively all the candidate blocks in the search area, is applied. Therefore, many fast algorithms have been proposed to speed up the motion search process. Many fast methods accelerate the search process by sophisticatedly inspecting a subset of checking points only for a good (if not the best) matching result [58]-[59]. Some schemes speed up the matching procedure by utilizing the bound property of pixel-based SAD to eliminate impossible candidates with (much) simpler calculations, such as the successive elimination algorithm (SEA) and its variants [60]-[63]. Instead of using pixel-based SAD

for the block matching, another way to accelerate the block matching is to simplify the SAD calculation based on a fewer number of block features or attributes rather than all the pixels in the block. Some examples are distortion measurement using sign truncated feature in [66] and matching using pixel sub-sampling in [65]. In this chapter, we study the feature-based block matching.

Recently, Nguyen and Tan introduced a fast matching method in [68] to perform the block matching based on the subblock-sum feature that can be quickly calculated using integral image [21]. In [69], they further improved the method by introducing a new feature based on subblock variances that can also be calculated from the integral image. However, the variance feature is calculated from the integral image of squared pixel intensities thus demanding extra computation load, especially requiring multiplication and division operations. Both the features are obtained inside each individual subblock, thus being termed as intra-subblock features.

In this chapter, we investigate the performance of applying the intra-subblock features in the block matching motion estimation for video coding. Based on the analysis, a set of new simple features, which are calculated from both within individual subblock and between neighboring subblocks (known as inter-subblock features), are proposed. By amalgamating the intra- and inter-subblock information, these features can improve the matching accuracy for block matching motion estimation while achieving computational efficiency as they can also be quickly calculated from the integral image without multiplication and division operations. Furthermore, a multi-level feature representation is introduced in a multi-resolution way for a block, which can provide a flexible and scalable means in a trade-off between saving more computation load and achieving higher matching accuracy. Interestingly, the new feature-based SAD measure can be treated to be the subblock-based sum of Hadmard transformed differences (SATD).

The remainder of this chapter is organized as follows. Section 5.2 briefly reviews the SAD measure for block matching and the recent intra-subblock feature based SAD measures in [68][69]. Section 5.3 firstly makes an analysis of applying the intra-subblock features alone in the SAD calculation for block matching motion estimation, and then introduces a set of new intra- and inter-subblock features in a multi-level structure to

improve block matching performance. Moreover, the interpretation of the new feature-based SAD and the computational complexity analysis are presented. Section 5.4 shows the experimental results using the proposed motion estimation method in video coding and a conclusion is given in Section 5.5.

5.2. Review of SAD Based Measurement

In the following, we will briefly review the definition of the traditional sum of absolute differences (SAD) commonly used in block matching motion estimation and the SAD using the intra-subblock features introduced recently.

Let W represent some attribute of a block. For notational simplicity, a difference function df is used. The difference between two blocks B_1 and B_2 measured with the attribute W can be obtained using the difference function df with respect to W as

$$df(W) = W_{(B_1)} - W_{(B_2)} \quad (5.1)$$

where $W_{(B_i)}$ is the attribute value of the block B_i . The attribute W can be pixel intensity or any other user-defined features.

The following properties of the difference function can be easily verified according to Eqn. (1):

$$\text{Property 1. } df(-W) = -df(W) \quad (5.2)$$

$$\text{Property 2. } df(W_1 + W_2) = df(W_1) + df(W_2) \quad (5.3)$$

$$\text{Property 3. } df(W_1 - W_2) = df(W_1) - df(W_2) \quad (5.4)$$

$$\text{Property 4. } df(a \cdot W) = a \cdot df(W) \quad (5.5)$$

Using the difference function, the traditional sum of absolute differences commonly used as the distortion measure in block matching motion estimation is represented as:

$$SAD = \sum_{0 \leq i, j \leq N-1} |df(I(i, j))| \quad (5.6)$$

where the attribute $I(i, j)$ is the pixel intensity at (i, j) in a block and $N \times N$ is the block size. It can be seen that the traditional SAD involves N^2 difference functions, each of which takes one subtraction, N^2 absolute operations on the difference function results and another $N^2 - 1$ addition to sum them up, and that the amount of computation operations is determined by the number of attributes representing the blocks (N^2 pixel intensities in this case). As a distortion measure, the SAD calculation will be repeated for the all the candidate blocks in the search area, such that motion estimation based on the traditional SAD requires a huge computation load. Therefore, if we can find a fewer number of attributes that can represent the block effectively as well as can be calculated efficiently, the computational complexity for the attribute based SAD calculation will be reduced accordingly.

Recently, Nguyen and Tan proposed a fast block matching using SAD based on such a set of attributes, the subblock sums (BS) in [68] where the method is known as SAD_{BS} . In SAD_{BS} , a block is partitioned into some subblocks arranged in fixed patterns, and the block will then be represented using the sums of all pixel intensities inside those subblocks. For example, the $N \times N$ block is partitioned into two vertical subblocks of identical size $N/2 \times N$, and the two subblock sum features are calculated as

$$\begin{cases} BS_1 = \sum_{i=0}^{N/2-1} \sum_{j=0}^{N-1} I(i, j) \\ BS_2 = \sum_{i=N/2}^{N-1} \sum_{j=0}^{N-1} I(i, j) \end{cases} \quad (5.7)$$

Since they can be quickly computed using the integral image introduced in [20] (which will be described in the next section), the following SAD based on the above two subblock sum features can greatly accelerate the block matching procedure:

$$SAD_{BS} = \sum_{i=1}^2 |df(BS_i)| \quad (5.8)$$

The block can be divided into more subblocks such as the 4-strip, 4-block, 16-strip and other fixed patterns suggested in [68]. Therein, these features are shown to be able to represent the blocks effectively for block matching motion estimation in video coding.

In [69], Nguyen and Tan further proposed a new feature based on the subblock variance, which, together with the subblock sum feature (jointly denoted as SAD_{SV}), improves the matching results at the computational cost. However, both the sums and variances used in SAD_{BS} and SAD_{SV} are calculated using the pixels inside each individual subblock, and they are therefore termed as intra-subblock features. The intra-subblock features, as will be analyzed in the next section, experience a coding inefficiency. Therefore, a new set of intra- and inter-subblock features are proposed to improve the problem by exploiting both the information inside individual subblocks and that between neighboring subblocks.

5.3. SAD Based on Multi-level Intra- and Inter-features

5.3.1. Analysis of intra-subblock feature based SAD measurement

To reveal the deficiency of the SAD using intra-subblock attributes, we consider a block that is represented by two attributes W_1 and W_2 calculated from two subblocks such as the subblock sums in Equation (5.7). Then the SAD for the block is

$$SAD = |df(W_1)| + |df(W_2)| \quad (5.9)$$

Define the *sum* and the *contrast* of the two attributes W_1 and W_2 as (the *contrast* here means the difference of the two attributes):

$$\begin{aligned} s &= W_1 + W_2 \\ c &= W_1 - W_2 \end{aligned} \quad (5.10)$$

Thus, applying the properties of the difference function $df(\cdot)$, the SAD in Equation (5.9) can be expressed as:

$$\begin{aligned} SAD &= |df(W_1)| + |df(W_2)| = |df((s+c)/2)| + |df((s-c)/2)| \\ &= \frac{1}{2}|df(s+c)| + \frac{1}{2}|df(s-c)| \\ &= \frac{1}{2}[|df(s) + df(c)| + |df(s) - df(c)|] \end{aligned} \quad (5.11)$$

where $df(s)$ refers to *sum* difference and $df(c)$ is *contrast* difference. In the above equation, if $|df(s)| \geq |df(c)|$, $df(s) + df(c)$ and $df(s) - df(c)$ have identical signs; otherwise, they have opposite signs. Therefore, we have

$$SAD = |df(W_1)| + |df(W_2)| = \begin{cases} |df(s)| & \text{if } |df(s)| \geq |df(c)| \\ |df(c)| & \text{if } |df(s)| < |df(c)| \end{cases} \quad (5.12)$$

Remark: From Equation (5.12), it can be seen that the intra-subblock feature based SAD only takes into account the larger value of $|df(s)|$ or $|df(c)|$. That is to say, the relatively smaller one of the two difference terms will be discarded, which itself may not be absolutely small, thus leading to a poor matching result. This is especially unfavorable in video coding because of the reason as follows. If the discarded term is a large (at least not small) *contrast* difference, that means the residual (error) block from the candidate block contains high frequency components in transform domain, which is, however, not counted in the SAD measure. It is known that a residual block containing high frequency components requires more bits for encoding. Therefore, the cost of discarding a large *contrast* difference value in the distortion measure for block matching is high which should be avoided as far as possible.

For block matching based on the traditional pixel-based SAD measure, which is also an intra-subblock feature based SAD with each pixel being a subblock, the detrimental effect caused by the above mentioned limitation is normally very small (if not negligible) in view of small *contrast* difference values discarded, as the adjacent pixels in a natural video frame is generally smooth thus the *contrast* differences between them taking small values. However, for the intra-subblock feature based SAD algorithms with larger subblocks, the impact may be significant with possibly large *contrast* difference values discarded, because the magnitudes of features (e.g., the sum of pixels in a subblock) increase substantially and the smooth property may not hold for the neighboring subblocks with large spatial sizes.

5.3.2. Proposed sum and contrast based SAD

From the preceding sub-section, we know that the intra-subblock attribute based SAD may produce unfavorable residual blocks for coding. It is noted that the two terms introduced in the above analysis, the *sum* and the *contrast* of W_1 and W_2 , are themselves two attributes representing the block. The above-mentioned problem can be eliminated if the *contrast* attribute, which is an inter-subblock feature, is also considered in the distortion calculation. Therefore, a couple of features, namely *sum* and *contrast* features, are employed to represent both the intra- and inter-subblock features. For example, as shown in Figure 5-1 (a) and (b) the block partitioned into two vertical sub-blocks will be represented using one intra-subblock feature, *sum* (s), and another inter-subblock feature, *vertical (block) contrast* (vc), both of which are computed from the sums of the two vertical subblocks v_1 and v_2 as

$$\begin{cases} s = v_1 + v_2 \\ vc = v_1 - v_2 \end{cases} \quad (5.13)$$

Figure 5-1 One intra- and three inter-subblock features in a block.

Similarly, as illustrated in Figure 5-1 (c) and (d), we can obtain another two *contrast* features, *horizontal (block) contrast (hc)* and *diagonal contrast (dc)*, which are calculated as

$$\begin{cases} hc = h_1 - h_2 \\ dc = (d_1 + d_4) - (d_2 + d_3) \end{cases} \quad (5.14)$$

where h_1 , h_2 , d_1 , d_2 , d_3 and d_4 refer to the sum of all the pixel intensities in the corresponding subblock in the figure.

These subblocks can be further partitioned into smaller subblocks till reaching subblock size 1×1 . Each partition will generate a quaternion of *sum-contrast* features with 1 sum and 3 contrast features. If totally M quaternion of features are generated for a block, the distortion measured with the *sum-contrast* feature based SAD, denoted as SAD_{SC} , will be calculated as

$$SAD_{SC} = \sum_{i=1}^M (|df(s_i)| + |df(vc_i)| + |df(hc_i)| + |df(dc_i)|) \quad (5.15)$$

It is expected that the inter-subblock attribute will supplement the intra-subblock feature in SAD based matching procedure, thus providing a better matching result for video coding. This may be justified by the following analysis on the relationship between the intra-/inter-subblock feature based SAD (i.e., SAD_{SC}) in Equation (5.15) and the Sum of Hadamard Transformed Differences (SATD) [124].

The SATD based on 4×4 block has been adopted in H.264 reference software [122] for sub-pixel motion estimation and mode selection, which is usually believed to produce a better coding result than SAD [124]. The 4×4 SATD is given as

$$SATD(\mathbf{D}_{N \times N}) = \sum_{i=1}^4 \sum_{j=1}^4 |t_{ij}| \quad (5.16)$$

where t_{ij} is the (i, j) -th element of the following two-dimensional Hadamard transformed block:

$$\mathbf{T}_{4 \times 4} = \mathbf{H}_{4 \times 4}^T \cdot \mathbf{D}_{4 \times 4} \cdot \mathbf{H}_{4 \times 4} \quad (5.17)$$

with $\mathbf{H}_{4 \times 4}$ being the 4×4 Hadamard matrix and $\mathbf{D}_{4 \times 4}$ the difference block. The Hadamard transform closely emulates the DCT transform that is applied to the prediction error in the coding process of H.264. Two out of four base functions of the 4×4 integer DCT matrix in the H.264 encoder are the same as the Hadamard base functions, while the characteristics of the other two are also similar to the DCT ones. Therefore, the frequency characteristics of the prediction error are nicely emulated by the SATD. The measure actually acts as a mixed distortion and rate criterion while being multiplication-free as well.

For a block partitioned in Figure 5-1, we know that the SAD_{SC} based on the *sum-contrast* features is

$$\text{SAD}_{\text{SC}} = |df(s)| + |df(vc)| + |df(hc)| + |df(dc)| \quad (5.18)$$

As shown in Figure 5-1 (d), the block can be represented by the four subblock sums, d_1 , d_2 , d_3 and d_4 . Hence, the block difference based on these four attributes can be arranged into a 2×2 matrix

$$\mathbf{W}_{2 \times 2} = \begin{bmatrix} df(d_1) & df(d_2) \\ df(d_3) & df(d_4) \end{bmatrix} \quad (5.19)$$

Applying a 2×2 Hadamard transform and the properties of $df(\cdot)$, we have

$$\mathbf{T}_{2 \times 2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathbf{W}_{2 \times 2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} df(s) & df(vc) \\ df(hc) & df(dc) \end{bmatrix} \quad (5.20)$$

It can be found that the Hadamard transformed matrix turns out to be composed of the differences based on the four proposed features. In other words, the proposed intra-/inter-subblock feature based SAD_{SC} is actually the 2×2 subblock based SATD for the four subblock sum attributes d_1 , d_2 , d_3 and d_4 . Note that the intra-/inter-subblock feature based SAD_{SC} can be further extended to $N \times N$ subblock based SATD, with more complexity as N increases.

Compared with the subblock variance feature in [69] that is used to represent inter-pixel statistics within a subblock, the proposed inter-subblock features are much more efficient. The fast calculation for the inter-subblock *contrast* values, which will be elaborated in the following based on the same integral frame used to calculate subblock

sum features, is much simpler compared with that of subblock variance feature which requires an extra integral image of squared pixel intensities as well as multiplication and division operations. Moreover, it is possible to construct the intra-/inter-subblock features in a hierarchical structure to yield flexibility in trading off the computational complexity and coding efficiency.

5.3.3. Multi-Level Sum-Contrast Features

We know that the number of *sum-contrast* quaternions used in the SAD_{SC} measure in Equation (5.15) depends on the way how a block is partitioned into subblocks. Partitioning a block into more subblocks will normally achieve better performance as more features will be generated to represent the block more accurately. However, too many subblocks will increase the computational burden. On the other hand, too few subblocks may not provide sufficient spatial granularity for the block thus impairing the matching accuracy. In order to balance the computational load and the matching accuracy, this sub-section provides a multi-level structure to represent the block using *sum-contrast* features in a multi-resolution fashion, which can satisfy different computation resource requirements as a lower level with a larger subblock size demands less computation while a higher level will offer better matching performance with more computations. From our simulation result shown in the next section, it can be seen that using the *sum-contrast* features, even the lowest level can generally produce very good results.

The multi-level feature representation for a block of size $N \times N$ is depicted in the following. The multi-level structure for other block size can be constructed similarly. As shown in Figure 5-1, at the lowest level L_0 , the block is represented using four features, namely *sum*, *vertical contrast*, *horizontal contrast* and *diagonal contrast*. We can build higher levels by further decomposing the block. For example, the block is firstly divided into four subblocks of equal size at Level L_1 , for each of which we can obtain four features of *sum*, *vertical contrast*, *horizontal contrast* and *diagonal contrast* again, totaling the number of features representing the block at this level to 16. For other higher levels, features can be obtained in the same way as illustrated in Figure 5-2. For a 16×16 block, totally 4 levels can be constructed with the size of the subblocks in the highest

level L_4 down to 2×2 . For the Level L_n , $4 \cdot 4^n$ features can be generated. Therefore the SAD_{SC} for L_n is

$$SAD_{SC} = \sum_{i=1}^{4^n} (|df(sum_i)| + |df(vc_i)| + |df(hc_i)| + |df(dc_i)|) \quad (5.21)$$

Figure 5-2 Multi-level structure for *sum-contrast* features.

5.3.4. Fast Feature Calculation Based On Integral Image

As introduced in [69], in order to efficiently compute these subblock based features, an image representation called Integral Image is used. The integral image at location (x, y) is defined as the sum of all pixel intensities (of the input image) above and to the left, including itself, that is

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \quad (5.22)$$

where $II(x, y)$ is the integral image at position (x, y) and $I(x', y')$ is the image intensity at (x', y') . The definition of integral image is demonstrated in Figure 5-3 (a), where the integral image at (x, y) is the sum of all the pixel intensities in the shaded area.

Integral image can be efficiently computed by only a single scan over the original image using the following equations:

$$II(x, y) = II(x-1, y) + s(x, y) \quad (5.23)$$

where $s(x, y)$ is the column pixel intensity sum above (x, y) inclusive:

$$s(x, y) = \sum_{y' \leq y} I(x, y') = s(x, y-1) + I(x, y) \quad (5.24)$$

Figure 5-3 Efficient computation of a feature using the integral image.

Based on the integral image, the proposed features can be efficiently calculated. For example, the *vertical block contrast* feature shown in Figure 5-3 (b) can be quickly computed:

$$vc = (II_e + II_a - II_b - II_d) - (II_f + II_b - II_c - II_e) \quad (5.25)$$

where a , b , c , d , e and f denote the locations. It can be seen that seven additions/subtractions are needed to calculate such a feature. In fact, the computation for this *vertical contrast* feature can further be reduced with the help of sum feature which is normally calculated first. Suppose the sum of the two subblocks in the figure is s , then the *vertical contrast* feature can be calculated using 5 operations (four additions/subtractions and one shift operation) as

$$vc = s - 2(II_f + II_b - II_c - II_e) \quad (5.26)$$

In a similar way we can calculate the *horizontal contrast* and the *diagonal contrast* feature using 5 and 8 operations respectively.

5.3.5. Computation Complexity Analysis

In the following we compare the computation complexity of the proposed SAD_{SC} with the L_0 features against other relevant SAD measurements for one frame motion estimation based on the full search (FS). Note that the proposed feature matching algorithm can also be incorporated into any fast searching methods such as the hexagon search [59]. Similar analysis can be made for different searching strategies.

Consider a video frame of size $L \times M$ which is partitioned into blocks of $N \times N$, and the full search for the best match is carried out in a window containing of $W \times W$ candidates. In the conventional pixel based SAD, to calculate the distortion of one candidate block, N^2 subtractions, $N^2 - 1$ additions and N^2 absolute operations are needed, which makes the computational complexity of the block matching for a single block to be $W^2 \times (3N^2 - 1)$ arithmetic operations. Hence, the computational complexity of BM motion estimation using pixel-based SAD for a frame becomes:

$$C_{ME-SAD} = O(W^2 \times (3N^2 - 1) \times L \times M / N^2) \quad (5.27)$$

Then we examine the computational complexity of SAD_{SC} using the level L_0 features in the proposed method. Since the block matching is performed on four features, the distortion measure requires 11 operations (4 subtraction, 4 absolute operations, and 3 additions). Therefore the computational complexity of the matching in motion estimation for one frame becomes:

$$C_{SAD_{sc}} = O(11 \times W^2 \times L \times M / N^2) \quad (28)$$

However, additional computation is needed to compute the feature values before conducting the feature based matching procedure. Firstly, it has been shown that $2L \times M$ additions are needed to calculate the integral image. Then according to the feature calculation using the integral frame in the preceding sub-section, the four features in the level L_0 for one block demand 21 operations totally (3 for *sum*, 5 for *vertical* and *horizontal contrast* each, and 8 for *diagonal contrast*). Therefore, for frame size of $L \times M$ and block size of $N \times N$, the total number of operations to calculate the feature representation for all blocks in a frame (including the calculation of integral image) is $2L \times M + 21L \times M / N^2$.

In the video coding, decoded frames rather than original frames are used as reference. Therefore, for every frame, the feature calculation has to be carried out twice (one is based on the original current frame to estimate its block motion vectors of which the computation complexity has been discussed in the above paragraphs, while the other is based on the decoded frame to calculate candidate features for motion estimation of the

next frame). For the decoded frame, candidate features have to be calculated on every pixel position, and thus requiring totally $2L \times M + 21(L \times M) = 23 L \times M$ operations. Therefore the feature computation for one frame totally needs:

$$C_{features} = O(25L \times M + 21L \times M / N^2) \quad (5.28)$$

With Equation (28) and (29), the motion estimation computational complexity for one frame using level L_0 feature based SAD_{SC} is:

$$C_{ME-SAD_{sc}} = C_{SAD_{sc}} + C_{features} = O((11W^2 + 21) \times L \times M / N^2 + 25L \times M) \quad (5.29)$$

From Equation (5.27) and (5.29), the computational complexity ratio between the proposed feature based scheme and the pixel based SAD method is

$$C_{ME-SAD_{sc}} / C_{ME-SAD} = (11W^2 + 25N^2 + 21) / (3N^2W^2 - 1) \quad (5.30)$$

For example, for a typical search range of $W=31$ and block size of $N=16$, $C_{ME-SAD_{sc}} / C_{ME-SAD} \approx 2.3\%$, That is to say, the proposed SAD_{SC} using Level L_0 features requires only 2.3% computation operations of that for the pixel based SAD, which shows a considerable speedup gain. On the other hand, the coding rate-distortion performance using SAD_{SC} and SAD are however comparable, which will be demonstrated in the experimental results in the next section. Note that the analysis here does not consider the matching processes on the frame boundary, which requires less search points and thus may slightly change the computational complexity. The computational complexity SAD_{SC} using features in other levels can be analyzed similarly.

In addition, the intra-subblock feature based SAD_{BS} [68] and SAD_{SV} [69] are also simulated for comparison in our experiments in the next section. Therefore, their computational complexities are briefly discussed here. We implement the SAD_{BS} and the SAD_{SV} using the subblock sizes shown in Figure 5-1 to compare with the proposed SAD_{SC} at the level L_0 , that is to say, four subblock sums for SAD_{BS} and four subblock variances for SAD_{SV} that are corresponding to four features in the proposed L_0 SAD_{SC} are used. Hence the computation load for SAD_{BS} and SAD_{SV} is related to the 4-strip or 4-block case in [69]. Based on the analysis therein and the above analysis for SAD_{SC} , the computational complexities for the different SAD schemes with the subblock size at L_0

are compared in Table 5-I. It can be seen that for CIF format ($L=352$, $M=288$) with $W=31$ and $N=16$, compared with the proposed SAD_{SC} , the SAD_{BS} demands less operations (about 31.8% less) while the SAD_{SV} requires substantially more computation (around 51.5% more).

Table 5-I . Comparison of required operations for motion estimation with different measures (an example of $L=352$, $M=288$, $W=31$ and $N=16$)

Measure	Number of Operations
SAD_{SC}	$25L \times M + (11W^2 + 21)L \times M/N^2$ (6,728,832)
SAD_{BS}	$4L \times M + 11W^2 \times L \times M/N^2$ (4,591,620)
SAD_{SV}	$18L \times M + 22W^2 \times L \times M/N^2$ (10,197,000)
SAD	$W^2 \times (3N^2 - 1)L \times M/N^2$ (291,886,452)

5.4. Experimental Results

We conducted experiments by applying different SAD measurements including the proposed SAD_{SC} to the motion estimation in the reference software JM8.5 of the latest video coding standard H.264. Eight video sequences (6 CIF and 2 QCIF) were tested in the experiments, which contain different types of motion such as slow motion, medium to fast motion, and camera panning. The coding structure IPPPPIPPPP (i.e., one Intra frame followed by four P-frames) was used, with a square search range of $[-15, +15]$. The rate-distortion optimization (RDO) option was turned on but only 16×16 block motion estimation was enforced for the testing. The coding results using SAD, SAD_{BS} , SAD_{SV} and SAD_{SC} are compared in terms of rate-distortion-complexity performance, specifically, PSNR of the reconstructed video sequences and the associated bit rate, as well as CPU computation time for motion estimation.

Figure 5-4 plots the rate-distortion comparison for coding “Dancer.CIF” and “GoldFish.CIF” sequences using SAD and the SAD_{SC} from L_0 to L_3 respectively. It can

be seen that while the SAD_{SC} using the features in the level L_0 is slightly worse than the conventional SAD, the SAD_{SC} using features in the higher levels performs very closely to SAD. Although using the features in L_0 may degrade the R-D performance slightly, it is useful in applications with scarce or constraint computation resource such as handheld portable devices because it can save significant computation as shown in the above analysis.

Table 5-II compares the coding performance using SAD and the proposed SAD_{SC} with different levels of features for all the tested video sequences for quantization parameter (QP) of 35 (a typical value around the middle of H.264 QP range 0-51). From the table it can be seen that the SAD_{SC} with L_0 features performance slightly worse than SAD, while the SAD_{SC} using features in L_1 and L_2 can achieve rate-distortion results close to those for SAD.

Figure 5-4 Rate-distortion performance using SAD and SAD_{SC} at different levels for two sequences.

Table 5-II Coding performance using SAD_{SC} and SAD (PSNR: dB, bit rate: kbps)

Measure Sequence		SAD_{sc}			SAD
		L_0	L_1	L_2	
Akio.CIF	PSNR	39.86	39.94	40.01	40.05
	bit_rate	122.99	110.37	106.07	105.55
BBC.CIF	PSNR	29.76	29.79	29.82	29.84
	bit_rate	1275.94	1112.75	1075.75	1066.11
Coastguard.CIF	PSNR	29.54	29.58	29.59	29.59
	bit_rate	637.77	590.95	569.34	561.26
Dancer.CIF	PSNR	35.19	35.14	35.16	35.17
	bit_rate	326.73	295.54	282.43	278.15
Foreman.CIF	PSNR	32.22	32.35	32.41	32.45
	bit_rate	348.79	315.48	306.95	297.53
Goldfish.CIF	PSNR	35.1	35.1	35.11	35.12
	bit_rate	420.1	387.37	372.01	368.83
carphone.QCIF	PSNR	31.73	31.82	31.86	31.88
	bit_rate	147.55	134.94	130.67	129.74
Stefan.QCIF	PSNR	28.17	28.28	28.32	28.33
	bit_rate	393.83	366.07	354.32	350.52

Moreover, the coding performance of the three feature-based schemes, namely, the proposed SAD_{SC} , the SAD_{BS} [68] and the SAD_{SV} [69], is compared with the results shown in Table 5-III. It can be seen that at the same level the proposed SAD_{SC} achieves similar rate-distortion results as the SAD_{SV} , while it produces generally better rate-distortion performance than the SAD_{BS} . Meanwhile, we also conducted the experiment to compare the motion estimation time based on the three SAD measures on a laptop running Windows XP Professional with Intel Pentium Mobile 1.7G CPU and 965M RAM. No special program optimization was made in the implementation. The results are tabulated in Table 5-IV, which clearly show that the proposed SAD_{SC} can achieve slightly slower speed than the SAD_{BS} in average but substantially faster than the SAD_{SC} in the motion estimation. For example, the average motion estimation time (in seconds) for the six CIF sequences with SAD_{SC} are 1.17, 2.88, 10.21 at level L_0 , L_1 and L_2 , respectively, which are close to 1.09, 2.76, 8.35 with the SAD_{BS} , remarkably faster than that using the SAD_{SV} which are 1.56, 4.30 and 14.16 respectively. Therefore, compared with the SAD_{BS} and the SAD_{SV} , our proposed SAD_{SC} shows more promising results in terms of rate-distortion-complexity.

Table 5-III Coding performance comparison for three feature-based SADs at different levels (PSNR: dB, bit rate: kbps)

Measure Sequence		L_0			L_1			L_2		
		SAD_{sc}	SAD_{BS}	SAD_{SV}	SAD_{sc}	SAD_{BS}	SAD_{SV}	SAD_{sc}	SAD_{BS}	SAD_{SV}
Akio.CIF	PSNR	39.86	39.84	39.89	39.94	39.92	39.93	40.01	40.01	39.99
	bit_rate	122.99	126.55	122.28	110.37	111.68	110.26	106.07	107.67	106.33
BBC.CIF	PSNR	29.76	29.72	29.76	29.79	29.79	29.8	29.82	29.81	29.81
	bit_rate	1275.94	1303.39	1259.51	1112.75	1114.79	1111.25	1075.75	1082	1076.94
Coastguard..CIF	PSNR	29.54	29.5	29.55	29.58	29.57	29.58	29.59	29.59	29.59
	bit_rate	637.77	655.89	636.51	590.95	596.5	591.67	569.34	571.04	571.91
Dancer.CIF	PSNR	35.19	35.15	35.18	35.14	35.13	35.15	35.16	35.14	35.15
	bit_rate	326.73	330.82	323.65	295.54	303.47	298.98	282.43	290.96	285.5
Foreman.CIF	PSNR	32.22	32.17	32.23	32.35	32.33	32.34	32.41	32.43	32.41
	bit_rate	348.79	357.18	345.29	315.48	323.42	319.37	306.95	314.45	308.97
Goldfish.CIF	PSNR	35.1	35.09	35.1	35.1	35.09	35.08	35.11	35.1	35.09
	bit_rate	420.1	424.8	418.35	387.37	390.86	386.45	372.01	377.31	373.66
carphone.QCIF	PSNR	31.73	31.69	31.75	31.82	31.8	31.82	31.86	31.85	31.86
	bit_rate	147.55	147.22	146.87	134.94	137.13	135.69	130.67	132.67	131.26
Stefan.QCIF	PSNR	28.17	28.12	28.2	28.28	28.27	28.28	28.32	28.31	28.32
	bit_rate	393.83	402.52	395.19	366.07	364.76	362.95	354.32	354.31	353.78

Table 5-IV Computation time comparison for three feature based SADs at different levels (time in seconds)

Measure Sequence		L_0			L_1			L_2		
		SAD_{sc}	SAD_{BS}	SAD_{SV}	SAD_{sc}	SAD_{BS}	SAD_{SV}	SAD_{sc}	SAD_{BS}	SAD_{SV}
Akio.CIF		1.082	0.96	1.562	2.253	2.402	3.788	7.909	6.083	11.304
BBC.CIF		1.202	1.252	1.754	2.955	2.836	4.674	11.456	9.443	15.975
Coastguard.CIF		1.213	1.014	1.411	3.472	3.035	4.869	11.306	10.632	17.168
Dancer.CIF		1.261	1.211	1.722	2.946	2.624	4.025	10.616	7.615	13.691
Foreman.CIF		1.13	1.162	1.562	2.902	3.014	4.304	10.918	8.9	14.243
Goldfish.CIF		1.16	0.95	1.372	2.776	2.647	4.121	9.049	7.404	12.575
Average(CIF)		1.17	1.09	1.56	2.88	2.76	4.30	10.21	8.35	14.16
Carphone.QCIF		0.221	0.32	0.4	0.971	0.78	1.062	2.644	2.253	3.554
Stefan.QCIF		0.29	0.251	0.3	0.661	0.72	1.06	2.645	2.414	3.735
Average(QCIF)		0.255	0.286	0.35	0.816	0.75	1.061	2.644	2.334	3.645

5.5. Summary and Discussion

Feature based SAD can be used to accelerate block matching process. We have firstly examined the intra-subblock feature based SAD in block matching motion estimation for video coding in terms of coding efficiency. Based on the analysis, a set of simple yet effective intra- and inter-subblock features, which exploit the information within individual subblocks and between neighboring subblocks, have been proposed to improve

the matching accuracy. It has been shown that the SAD based on the new features is equivalent to a 2×2 subblock based SATD. The new features can also be quickly computed from integral images and thus greatly accelerate the block matching based motion estimation while maintaining a good matching accuracy. Furthermore, a multi-level structure has been presented to represent a block using the intra- and inter-subblock features in a multi-resolution sense. The experimental results have demonstrated the efficiency and effectiveness of the proposed intra-/inter-subblock feature based SAD by comparing with other relevant feature based SAD measures.

Chapter 6.

Face Region Based Bit Allocation for Conversational Video Coding

6.1. Introduction

Many video applications, such as video streaming through networks, especially for real-time video communication system with low bandwidth, require the compressed video signal to be transmitted through constant bit streaming channels. However, the changing video content in different frames inherently produces varying coding bits if constant coding parameters are used. Therefore, rate control, the technique to optimize the reconstruction visual quality while achieving a constant target bit rate by altering the encoding parameters, is very important for many practical video applications. It has therefore been intensively studied, and all the established video coding standards have their own rate control strategies [1]-[5]. Most of the video encoders achieve rate control by considering every part of the video frames with equal importance. However, the research on human visual system (HVS) suggests that only a small region of visual angle around the center of gaze is captured at high resolution due to the highly non-uniform distribution of photoreceptors on the human retina [129][130]. That is to say, HVS pays more attention to region of interest (ROI), such as the human faces in the conversational video content. An increasingly popular approach to improve the coding efficiency and the subjective visual quality is to segment a small number of interesting regions in each frame and encode them with different priorities [7]-[9]. In the conversational video communication system, human faces are considered to be the most informative regions.

This chapter will investigate such a coding method for conversational video using face ROI based bit allocation.

Two major issues have to be addressed in applying the ROI based coding, that is, (1) the detection and segmentation of the ROI, and (2) the ROI-based coding priority assignment for the basic coding units in a frame. For the first question, many schemes utilize the visual features including motion to detect the ROI [7]-[9]. However, due to the employment of rate-distortion optimization (RDO) in H.264/AVC, there emerges a dilemma in applying motion information to detect the ROI for the purpose of bit allocation and quantization parameter QP adjustment, since the motion information can only be obtained after the QP is determined. This is a typical chicken-and-egg problem. Therefore, these methods cannot be directly applied for ROI detection in the latest video coding standard H.264/AVC. Liu *et al* introduced an efficient face ROI determination method using the skin color combined with direct frame difference based motion detection [10]. Nevertheless, as the skin color in video frames is sensitive to the lighting condition, the face ROI detection results may be unsatisfactory in some scenes. Moreover, the color based model cannot provide an accurate face region as other human skin parts such as human limbs and skin-color-like background may also be mis-included.

In fact, face detection in still image has been well studied in the past decades. Among the various developed algorithms, the learning/classification based methods [13]-[21] have been widely considered to be one of the best solutions producing good detection results, such as the fast face detection by Viola using Adaboost based on Haar features [21]. While these methods developed for still images may be directly applied to each individual video frame, it is reasonable to further adapt the algorithms for face ROI detection in the context of image sequences to achieve higher computational efficiency and better performance in the face ROI based video coding.

Since the locations and sizes of face regions are unknown, most face detection methods commonly use a pyramid structure to partition a still image into sub-images at different scales, and these sub-images will then be classified into ‘face’ or ‘non-face’ by a trained classifier. The amount of sub-images generated from the pyramid structure is considerable, thus demanding high computation load for the classification process. It is

known that temporal correlation is the most important feature in video, which has been highly exploited for effective video coding. Naturally we consider utilizing temporal motion information to accelerate the face detection process by eliminating impossible sub-images as much as possible before performing the classification process. Nevertheless, it has been mentioned above that there exists a dilemma for the H.264/AVC in using the motion information for ROI based coding. In order to circumvent this dilemma, a simple pixel-wise frame difference algorithm was applied in [10]. While the scheme helps to eliminate the non-motion background, the result from the pixel-wise frame difference is sensitive to image noise. Considering that the integral image has been calculated in Viola's face detection method [21], which is also adopted in our face ROI based video coding framework, we employ a block *mean* difference approach, where the block mean can be efficiently computed from the integral image, to produce more smooth motion detection results for efficient and robust face detection.

Let us consider the second question of ROI-based coding priority assignment. A face region resulting from the Viola's face detection method is of square shape, which actually includes some non-face parts in the ROI. The priority-based encoding of such a squared face ROI will produce blocking artifacts at the boundary of this squared ROI which is very subjectively annoying, while on the other hand some more bit budget is demanded for the coding the squared ROI including non-face parts. Liu *et al* attempted to alleviate the artifact by smoothening the ROI coding priority mask with a mean filter [10]. However, the mean filter does not consider the facial features and may assign some highly informative areas (such as eyes and mouth) with lower coding priorities than other less sensitive areas (like cheek and forehead regions). Since the ROI under consideration is human face, our scheme utilizes the prior knowledge of face geometry to achieve a more meaningful and accurate bit allocation that matches the characteristics of the HVS better.

The remainder of the chapter is organized as follows. Section 6.2 elaborates the complexity of using sub-images for face detection and introduces an efficient motion-based detection method to accelerate the process. In section 6.3, a bit allocation scheme based on the face geometry knowledge is proposed. The complete face based coding

scheme will be summarized in section 6.4. Simulation results are demonstrated in Section 6.4 and conclusion is given in Section 6.6.

6.2. Efficient Motion-based Sub-image Rejection for Face Detection

6.2.1. Pyramid structure in face detection: complexity analysis

The pyramid structure is commonly used in face detection to address the unknown face localization and size problem. The face detection system using the pyramid structure is illustrated in Figure 6-1.

Figure 6-1 Pyramid structure in classification based face detection.

The pyramid scheme scans through the input image at different scales and crops the original or the scaled image into sub-images of the same size as the sample images for training the classifier. Each sub-image (or the features obtained from it) will then be input into the trained classifier to categorize the sub-image as ‘face’ or ‘non-face’. The face is consequently located if any sub-image being classified as ‘face’ sub-image. It can be seen that when a classifier is determined, the computation complexity of this structure heavily depends on the number of sub-images.

Suppose that the size of the input image is $x \times y$, the size of sub-images used for classification is $x_0 \times y_0$, the image scaling factor is r , and the S_x and S_y are the stepping sizes to crop sub-images from the original or scaled images in x - and y -direction

respectively. It can be shown that the total number of the sub-images generated from the input image using the pyramid structure is (see Appendix B for details):

$$N = \frac{xy}{S_x S_y} \frac{r^2 - \min([x_0/x]^2, [y_0/y]^2)}{r^2 - 1} - \frac{(x_0 y + x y_0) (r - \min([x_0/x], [y_0/y]))}{S_x S_y (r - 1)} + \frac{x_0 y_0}{S_x S_y} (1 + \log_r \min([x/x_0], [y/y_0])) \quad (6.1)$$

Typically, x_0 and y_0 are around 20, the values of S_x and S_y are in the range of 1 to 5 pixels and $r=1.2$. Since usually x_0 and y_0 are much smaller than x and y , the number of sub-images obtained in Equation (6.1) can be approximated as

$$N \approx \frac{xy}{S_x S_y} \frac{r^2 - \min([x_0/x]^2, [y_0/y]^2)}{r^2 - 1} \quad (6.2)$$

Figure 6-2 shows the number of sub-images generated in the pyramid structure with different scaling factor r when S_x and S_y are set to 1. The horizontal axis represents the image size x or y (in pixels) for a square size image ($x=y$) for convenience. The vertical axis denotes the number of sub-images generated (in log scale). It can be seen that the number of sub-images is very large. For example, when $r = 1.1$ and the input image size is 256×256 , a large number of sub-images at the magnitude of 10^5 will be generated (the squared point indicated on the curve). The total sub-image number N reduces as the scaling factor increases. However, the decrease rate becomes smaller as r increases. Moreover, based on [21], the face detection accuracy may reduce by as much as 1-2% when r increases.

One can further adjust the parameters to compromise accuracy and processing time. For example, according to (6.1), the total number of the sub-images N is inversely proportional to the stepping sizes, S_x and S_y . Thus, if S_x and S_y are changed from 3 to 5, N will be reduced by a ratio of 9/25. However, the number of sub-images is still very large as shown in Figure 6-3, which plots the relationship of sub-image number N with varying stepping sizes when r is set to 1.2 (stepping size S_x and S_y are set to be identical for convenience of plotting and represented by s in the figure).

Figure 6-2 The number of sub-images N generated in the pyramid structure w.r.t scale factor r

Figure 6-3 The number of sub-images N generated in the pyramid structure w.r.t step size s .

It can be seen from the above analysis that the pyramid structure yields a substantial amount of sub-images, which thus demands very high computation load for the classification process. While processing video sequence, the extra temporal information motivates the employment of this important cue to accelerate the face detection by eliminating as more impossible sub-image as possible. Furthermore, it is known that even a well trained classifier cannot achieve 100% accuracy, especially when the operational environment is different from the initial training conditions, such as the change of luminosity or different facial expressions and poses. Therefore, removing a large number of impossible sub-images before classification will help reduce the face detection false alarm.

6.2.2. Efficient motion detection using block mean difference

The motion information needs to be obtained in order to eliminate some sub-images with the temporal information in the pyramid face detection structure. The dilemma using the motion information for ROI based H.264/AVC coding impedes the employment of the motion estimation result for this goal. The pixel-wise subtraction between two consecutive frames is a simple and widely used technique to detect moving objects in still background, which has been adopted in [10] for face ROI determination. However, the direct subtraction of the pixel intensities at a corresponding position is very sensitive to the pulse noise. In this sub-section, we introduce an efficient motion detection method based on block mean difference.

For a pixel located at (x,y) , the *mean* feature for a block is calculated as:

$$Mean(x, y) = \frac{1}{W^2} \sum_{p=-W/2}^{W/2} \sum_{q=-W/2}^{W/2} i(x+p, y+q) \quad (6.3)$$

where W is the size of the block, and $i(x,y)$ is the image intensity at (x,y) .

It is known that this mean calculation in Equation (6.3) has a low pass filtering characteristic of removing pulse noise in the image. Therefore, subtraction of the *mean* values at a corresponding pixel location is a more robust way to detect moving objects from adjacent frames. The motion detection using the *mean* difference can be expressed as:

$$M(x, y) = \begin{cases} 1 & \text{diff_mean}(x, y) \geq \theta \\ 0 & \text{diff_mean}(x, y) < \theta \end{cases} \quad (6.4)$$

$$\text{diff_mean}(x, y) = \text{mean}(x, y, t+1) - \text{mean}(x, y, t) \quad (6.5)$$

where $M(x,y)$ denotes the binary motion image with ‘1’ indicating pixels undertaking motion and ‘0’ for non-motion pixels, and θ is a user defined threshold. It can be seen from Equation (6.5), the weight $1/W^2$ for the mean calculation in (6.3) can be omitted for motion detection by adjusting the threshold θ accordingly.

With the help of the integral image, which is also used in Viola’s face detection method [21], the block mean feature can be quickly calculated. The integral image has

also been applied to efficiently compute the intra- and inter-subblock features in chapter 5, where it has been described in details (Equation (5.22) and Figure 5-3). For example, the *mean* value for the shadowed block shown in Figure 5-3 (b) can be easily obtained as

$$\text{mean} = a + d - b - c \quad (6.6)$$

where a , b , c , and d are the integral image pixels at the corresponding locations. From Equation (6.6), only three additions/subtractions are needed to calculate one mean feature from the integral image.

Figure 6-4 Efficient computation of mean using integral image.

6.2.3. Sub-image rejection using motion features

In this sub-section, a scheme to quickly reject sub-image patterns based on the motion detection results described above is introduced. The scheme is illustrated in Figure 6-5. For one sub-image, before feeding it into the feature extraction for classification, we calculate three motion features for the sub-image based on the binary motion image. The binary motion image is obtained from the *mean* feature based motion detection result described in previous sub-section. The three motion features are defined as the summation of all pixels in the motion feature masks shown in Figure 6-6, which are referred to as *face region*, *eye region* and *mouth region* motion feature, respectively. Therefore, the three motion features represent the number of the moving pixels in the corresponding regions, respectively. Based on the three motion features, three stages of rejection, each of which corresponds to one of the three motion features, are used to progressively eliminate the ‘non-face’ sub-image region, if a motion feature value at one of the stages is smaller than a threshold. In Figure 6-5, $T1$, $T2$ and $T3$ are the thresholds used for each motion features, which are empirically set to 10% of the area of the corresponding region in our experiments.

Figure 6-5 Sub-image rejection procedure based on the three motion features.

Figure 6-6 Three motion features for motion-based sub-image rejection.

Only the sub-images that pass through the motion based rejection procedure will be fed into the classification process, which will further categorizes them into ‘face’ or ‘non-face’ sub-images. In our framework, Viola’s face detection method [21] is adopted because it offers a good classification result and, more importantly, the integral image is employed to fast calculate the Haar features in the classification. Moreover, the integral image can be shared for fast *mean* calculation to reduce computation load further.

The motion features can also be computed efficiently from the integral image for the binary motion image in a similar way. In view that a large part in a common conversational video frame is still background, most of the sub-images without moving objects could be quickly eliminated, which reduces both the computation complexity and the false alarm for face detection in video. As no previous frame is available for the very

first frame, this motion feature based sub-image rejection scheme is only applied to the second and onward frames. All the sub-images for the first frame will perform the usual classification as that for a still image. Afterwards, the face regions detected from previous frame will be always re-checked by the classifier for a stationary face without going through the motion based rejection procedure. In this way, stationary faces can always be detected.

6.3. Face Region Based Bit Allocation in H.264/AVC

The detected face region in the previous section is of square shape. The HVS properties show that edge and contour information are very important to human visual system and directly related to our perception results [22]. In order to avoid blocking artifact as well as bit waste in the squared ROI coding, this section will discuss an active contour method to extract the accurate face region and then a facial geometry based bit allocation priority model is proposed to achieve better coding efficiency in the face ROI based coding.

6.3.1. Face contour extraction

The active contour model, also termed as *Snake* by the nature of its behavior, is a deformable model, which has been used for contour extraction and image interpretation. It has been used to locate and extract head boundaries and face features for face recognition [131][132]. The original active contour model was introduced by Kass *et al* [133] as a solution to optimal contour that minimizes an energy function incorporating the local strengths of the image features such as image gradient, and their spatial distribution. Consider a contour represented by a set of points, $\mathbf{v}(s) = (x(s), y(s))$, with s in the range of $[0, 1]$ denoting a parameterized variable along the contour arc. The total energy of the active contour model is the integrated energy along the contour, that is:

$$E_{snake} = \int_0^1 (E_{int} + E_{ext}) ds \quad (6.7)$$

where

- E_{int} represents the internal energy term of the *Snake*, measuring the continuity and smoothness of the contour. It is commonly computed as:

$$E_{\text{int}} = \frac{1}{2}(\alpha |\mathbf{v}'(s)|^2 + \beta |\mathbf{v}''(s)|^2) \quad (6.8)$$

where $\mathbf{v}'(s)$ and $\mathbf{v}''(s)$ denotes the first order and second order derivation of the curve with respect to s , and α, β are properly chosen parameters to control the contour's tension and rigidity respectively.

- E_{ext} is the external constraint energy. It can be derived either from object shape information, such as lines and edges of the image, or other user-defined function. The commonly used external energy term is defined in such a way that the contour is attracted to the edges of image I :

$$E_{\text{ext}} = -|\nabla I(x, y)|^2 \quad (6.9)$$

where ∇ is the gradient operator.

The *Snake* algorithms converge to the optimal object contour by minimizing its energy function:

$$u = \arg \min_{c \in C} E_{\text{snake}}(c) \quad (6.10)$$

where u is the optimal contour and C is the whole set for all possible contours c in the image plane.

In the discrete formulation, the contour is represented as a set of control points, $\mathbf{v}_i = (x_i, y_i)$ for $i = 0, \dots, N-1$, with x_i and y_i being the x and y coordinates of the i th control point. The energy function thus becomes

$$E_{\text{snake}} = \sum_{i=1}^N (E_{\text{int}}(\mathbf{v}_i) + E_{\text{ext}}(\mathbf{v}_i)) \quad (6.11)$$

We can use a discrete approximation of the first and second derivatives to compute E_{int}

$$E_{\text{int}}(\mathbf{v}_i) \approx \alpha |\mathbf{v}'| + \beta |\mathbf{v}''| = \alpha |\mathbf{v}_i - \mathbf{v}_{i-1}| + \beta |\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}| \quad (6.12)$$

The internal energy term smoothes the *Snake* contour and tends to converge the contour to one point, while the external energy term E_{ext} offsets the internal energy by attracting the contour to object boundary with salient gradient or user defined attributes, to reach a balance. By defining an external energy function properly, desired *Snake* contours can be extracted for different applications. For instance, the energy based on the

image gradient will position the contour on the object boundary with strongest edges. With changing scenarios, different external energy terms have been developed, such as the Gradient Vector Flow (GVF) for the concave contour in [134] and the Saliency Map (SM) for noisy images in [135]. In order to extract an accurate contour in the region with small gradient amplitudes, such as the chin and neck area, we proposed an effective external energy term, known as Gradient Angular Difference (GAD) in [136] for face contour extraction. The GAD term measures the difference between the orientations of the gradient vectors at neighboring *Snake* points. The image gradient vector orientations bear useful information about the smoothness of the object contour points and they are not affected by the edge gradient amplitude changes. Therefore, by minimizing the GAD term, the resultant contour deforms smoothly along the boundary of face region even in the low contrast areas. Since the focus of the work is on face ROI based video coding, the GAD approach is not detailed here and can be referred in [136].

The *Snake* algorithm is commonly implemented using dynamic programming (DP) [137] to search the optimal contour in the image plane. However, this method converges slowly, and thus takes a long computation time. Since in our scheme the face detection has obtained the square-shaped face region, the face contour can be initialized within the actual face contour. Therefore, a simplified search strategy called “normal line search” is used to speed up the searching process, which is illustrated in Figure 6-7. Specifically, for each control point on the initial contour, the searching for the best face contour is guided along the normal direction of the current contour. Hence, the searching process is significantly simplified compared with the exhaustive search in DP.

Figure 6-7 Normal line searching strategy.

6.3.2. Facial geometry based bit allocation

Bit allocation for rate control plays an important role in video coding, especially for low bandwidth video communication. Two essential issues have to be considered for rate control: a) the assignment of proper target bits to each rate control basic unit, and b) the adjustment of the coding parameters (commonly the quantization parameter QP) in order to achieve the target bits.

Different video coding standards have their own rate control schemes [1]-[5]. In the latest video coding standard H.264/AVC, the rate control is more difficult due to the chicken-and-egg dilemma caused by the rate-distortion optimization (RDO) based motion estimation [77]. The rate control strategy for H.264/AVC standard [77] was further improved in [78]. In the improved scheme [78], which is also adopted in our implementation, the rate control can be implemented at different levels, macroblock (MB) level rate control is considered here.

According to [78], the remaining bit $T[i]$ for the i^{th} MB in the MB level rate control can be obtained by

$$T[i] = \begin{cases} R & ; i=0 \\ T[i-1] - R[i-1] & ; else \end{cases} \quad (6.13)$$

where R is the target bit for current frame and $R[i-1]$ is the actual sum of bits used to encode previous MBs. Based on the remaining bits, the QP value at the MB level is adjusted according to

$$Q_p[i] = \begin{cases} Q_p[i-1] + 1 & ; T[i] < 0 \\ \frac{X_1[i] * \sqrt{MAD_{pred,adapt}[i]}}{T[i] - (N-i)X_2[i]} * \sum_{k=i}^N \sqrt{MAD_{pred,adapt}[k]} & ; T[i] \geq 0 \end{cases} \quad (6.14)$$

where $MAD_{pred,adapt}$ is the adaptive mean absolute difference (MAD) prediction introduced in [78] to achieve a more accurately predicted MAD by adaptively switching between the spatial model and the temporal model, $X_1[i]$ and $X_2[i]$ are the two parameters of their R-Q model that could be updated after the encoding of each basic unit, and N is the number of total basic units. More details on the development of the formulae can be referred in [78].

However, the above rate control model adjusts the QP by treating each part of a video frame equally regardless of its effect on the subjective visual quality. According to HVS characteristics, the face ROI needs to be coded with more bits with higher bit allocation priority for a better subjective visual quality. Therefore, Liu *et al* further developed a face ROI based rate control to address the problem [10]. However, there are some limitations in their scheme. The first one is that the face detection using the skin color may be unsatisfactory for some video scenes as skin color is sensitive to the lighting condition. Secondly, the color based model cannot provide an accurate face region, which may cause bit waste as discussed before. The other deficiency, caused by both the skin color detection and the mean filter over the detected ROI mask used to remove the blocking effects, is the ignorance of the different effects of facial components on human visual system, to which the edge and the contour of featured regions in the ROI are very important [22].

With the prior knowledge of facial feature geometry for a more meaningful and better bit allocation, we propose a facial geometry based priority assignment scheme for MB level rate control. Firstly, a facial geometry priority map is shown in Figure 6-8. All the pixels enclosed by the dashed line (covering eyes, nose and mouth) and the pixels around the face contour (indicated by the dark shadowed regions) are assigned with the highest priority of 3, and the other pixels inside the face area (the light shadowed regions) will be assigned a priority of 2, which is still higher than the lowest priority 1 for the pixels outside the face contour. Note that the numbers in the figure denote the geometrical proportion for the facial regions so that the locations and sizes for the three priority regions can be determined given an extracted face contour. To facilitate an easy implementation, we create a template of size 7×8 according to the proportional values shown in Fig. 8. For an extracted face contour, we first determine a rectangle just enclosing the face contour then scale and fit the pre-designed template into the rectangle to approximately locate the priority regions. In this way, the bit allocation priorities are assigned to all the pixels in the frame, resulting in a priority image $p(i, j)$, from which the MB bit allocation priorities can be calculated as

$$w[k] = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N p(i, j) \quad (6.15)$$

where $w[k]$ is the bit allocation priority for the k^{th} MB of size $N \times N$ in the frame, i, j denote the coordinates within one MB, and $p(i, j)$ is the priority value of pixel (i, j) in the MB.

Figure 6-8. Facial geometry priority map for bit allocation.

With the facial geometry based priority assignment, a QP adjustment method from [10] is used to adjust the QP for a face ROI based rate control at MB level, which computes the QP for each MB with

$$Q_p[i] = \frac{X_I[i]}{T[i] - (N - i)X_2[i]} \sqrt{MAD_{pred,adapt}[i]} * \sum_{k=i}^N \sqrt{\frac{w[k]}{w[i]} MAD_{pred,adapt}[k]} \quad (16)$$

where $T[i]$, $MAD_{pred,adapt}[i]$, $X_I[i]$ and $X_2[i]$ are identical to that in (6.14), while $w[i]$ is the above priority assigned to the i^{th} MB.

6.4. Face ROI Based Video Coding Architecture

The proposed face region based coding architecture for conversational video using the methods described in preceding sections is summarized in Figure 6-9, where the operation steps are shown in block diagrams. It can be seen that the integral image for the input frame are shared by the block mean based motion detection step and the face classification step to reduce the computation load. Different from the conventional coding, one input frame will first perform the processing of face based MB priority assignment, which comprises a few steps to assign proper coding priorities to all the MBs within a

video frame. The proposed scheme can readily fit into any video codec that support MB level rate control, although in our experiments we implement the scheme based on the H.264/AVC codec.

Figure 6-9. Proposed face ROI based coding architecture.

6.5. Experimental Results

In this section, we validate the proposed face ROI based video coding scheme by comparing it with the H.264/AVC codec JM9.8 [138] and the skin tone ROI based method in [10]. The proposed method was implemented based on the source codes of Liu's method [10]. Several standard conversational video sequences were tested in the experiment, including two sequences of CIF format ("Foreman" and "Silent"), two sequences of QCIF format ("Salesman" and "Miss America") and another two CIF sequences containing two faces in one frame ("News" and "Paris"). The first 100 frames from these sequences were tested with the first frame coded as intra-frame (I frame) and the rest frames coded to be inter-frames (P frames). The motion estimation search range was set to 16 and the rate-distortion optimization (RDO) option was turned on.

Firstly, we evaluate the proposed motion based sub-image rejection method. Figure 6-10 compares the number of sub-images with and without the proposed motion based rejection approach for frame #10 in "Miss America" sequence. It can be seen that the

proposed approach can greatly reduce the number of sub-images generated from pyramid structure in face detection. Table 6-I compares the sub-image numbers with and without the motion based rejection for all the tested sequence, where the sub-image numbers are averaged for all the tested 100 frames. It can be seen that the motion based rejection can significantly reduce the sub-image number. It has also been noticed that the rejected sub-image number is dependent on the video content. For example, since the sequence of “Silent.CIF” contains less moving areas in the frames than the “Foreman.CIF” sequence, more sub-images are eliminated using the motion features. Note that we used the source code of Viola’s face detection provided in [139], in which a canny edge based scheme was employed to prune the low-texture sub-images before classification.

Figure 6-10 Comparison of the subimage numbers with and without the proposed 3-stage rejection using motion features for frame #10 in “MissAmerica” sequence.

Table 6-I Sub-image number comparison with and without motion based rejection

Subimage No Sequence	W/O Rejection (Using Canny Pruning)	After 1 st Rejection Stage	After 2 nd Rejection Stage	After 3 rd Rejection Stage
Foreman.CIF	68,276	45,482	41,688	39,059
Silent.CIF	61,959	10,007	8,353	6,116
MissAmerica.QCIF	11,562	3,099	2,365	1,037
Salesman.QCIF	10,649	3,188	2,455	1,547
News.CIF	62,503	17,944	14,511	11,052
Paris.CIF	62,344	15,585	12,041	8,826

The remained sub-images are then fed into the face/non-face classifier to identify face regions in the images. Some result examples are shown in picture (b) of Figure 6-11 to Figure 6-14, which were directly obtained as the outputs from our program. The detected face regions by Viola’s method are shown to be the dashed rectangles. The face contour

extraction results are also shown in the pictures as the solid curves linked by control points in *Snake* algorithm, which present faces very well.

Based on extracted face contours, higher bit allocation priorities will be assigned to the important face features according to the facial geometry priority map. Pictures in (c) in Figure 6-11 to Figure 6-14 demonstrate the assigned priority scale for a frame, where the level of bit assignment priority is denoted by the brightness. The brightest parts refer to the highest priority, which just correspond to the most important facial components like eyes, mouth and face contour, as can be seen from these pictures. The non-ROI (non-face) parts are dimly marked which are assigned with the lowest priority. It can be seen that the face ROI based priority assignment fits closely to the most interested (important) face region and thus matches the HVS characteristic. Therefore, we can expect that the face ROI based bit allocation will further improve the ROI reconstruction quality which will be demonstrated in the following.

We now compare the proposed face ROI based bit allocation against the conventional rate control (non-ROI based) in JM9.8 and Liu's skin tone ROI based method [10]. Table 6-II and Table 6-III tabulate respectively the rate-distortion performance using different rate control methods with some typical bit rates for the testing CIF sequences and QCIF sequences with a single face in each frame, while Table 6-IV lists the results for the CIF sequences containing two faces in each frame. It is known that the overall PSNR, which is calculated from all the MBs in the frames without considering HVS characteristics, cannot represent the subjective visual quality well especially at low bit rates. Therefore, we also compared the PSNR in the face ROI. It can be seen from the tables that for all the sequences our method consistently produces significantly higher ROI PSNR values than the overall PSNR. And the ROI PSNR values by our scheme are also normally higher than those from the Liu's ROI method (up to 1.2 dB improvement). Therefore, although the average overall PSNR from our method slightly drops in most sequences, our method greatly improves the face ROI reconstruction PSNR as well as the subjective visual quality to be shown in the following examples.

Chapter 6: Face Region Based Bit Allocation for Conversational Video Coding

Table 6-II Coding performance comparison of JM98, Liu's skin tone ROI method and our scheme for CIF sequences

Sequence	Method	Bit Rate(kbps)		PSNR-Y(dB)	
		Target	Actual	Overall	ROI
Foreman CIF (30 fps)	JM9.8	28	36.90	25.24	24.81
	Liu's		36.01	25.06	24.84
	Ours		36.33	25.05	25.35
	JM9.8	64	67.69	28.66	28.23
	Liu's		64.14	28.65	28.76
	Ours		64.10	28.61	29.61
	JM9.8	128	131.69	31.43	31.54
	Liu's		128.22	31.69	32.23
	Ours		127.44	30.31	32.08
	JM9.8	256	260.57	33.92	34.24
	Liu's		254.22	34.15	34.83
	Ours		256.78	33.22	35.4
Silent CIF (30 fps)	JM9.8	28	34.35	27.55	26.93
	Liu's		28.14	27.51	27.61
	Ours		28.22	27.16	28.83
	JM9.8	64	69.88	29.84	27.95
	Liu's		64.36	30.95	29.79
	Ours		64.37	30.52	30.56
	JM9.8	128	134.18	32.07	31.15
	Liu's		128.89	32.91	32.96
	Ours		128.65	32.72	33.48
	JM9.8	256	263.88	34.45	34.41
	Liu's		257.28	35.34	36.02
	Ours		258.70	35.31	36.36

Table 6-III Coding performance comparison of JM98, Liu's skin tone ROI method and our scheme for QCIF sequences

Sequence	Method	Bit Rate(kbps)		PSNR-Y(dB)	
		Target	Actual	Overall	ROI
MissAmerica QCIF	JM9.8	28	28.85	37.04	36.24
	Liu's		28.14	37.8	38.70
	Ours		28.18	37.28	38.92
	JM9.8	64	65.76	39.05	38.75
	Liu's		63.97	40.66	40.96
	Ours		64.21	40.26	41.65
	JM9.8	128	129.09	42.06	42.10
	Liu's		128.46	42.61	42.36
	Ours		129.21	42.69	43.00
	JM9.8	256	257.58	43.39	43.94
	Liu's		256.51	44.31	44.6
	Ours		257.43	44.00	45.75
Salesman QCIF	JM9.8	28	30.69	30.44	28.86
	Liu's		27.51	30.88	30.66
	Ours		28.26	30.71	31.27
	JM9.8	64	67.68	33.93	32.28
	Liu's		64.35	33.86	34.43
	Ours		64.29	34.56	35.43
	JM9.8	128	134.17	37.36	36.46
	Liu [11]		128.85	37.71	38.69
	Ours		128.19	38.30	39.14
	JM9.8	256	262.90	40.92	39.99
	Liu's		257.60	42.05	42.19
	Ours		257.09	41.70	43.28

Chapter 6: Face Region Based Bit Allocation for Conversational Video Coding

Table 6-IV Coding performance comparison of JM98, Liu's and our scheme for sequences with two faces

Sequence	Method	Bit Rate(kbps)		PSNR-Y(dB)	
		Target	Actual	Overall	ROI
News CIF	JM9.8	28	35.74	29.11	28.01
	Liu's		28.16	29.02	28.84
	Ours		28.38	28.97	29.03
	JM9.8	64	69.27	31.64	30.57
	Liu's		63.78	31.99	30.45
	Ours		64.10	31.8	32.23
	JM9.8	128	136.68	34.07	33.82
	Liu's		126.98	34.08	33.81
	Ours		127.05	34.38	35.37
	JM9.8	256	269.64	36.98	36.99
	Liu's		255.3	37.33	37.35
	Ours		254.16	37.63	38.91
Paris CIF	JM9.8	28	40.17	20.40	20.35
	Liu's		44.29	20.81	20.90
	Ours		45.36	20.66	21.10
	JM9.8	64	66.62	25.21	25.15
	Liu's		64.36	25.48	25.61
	Ours		64.31	25.31	25.92
	JM9.8	128	133.02	27.50	27.21
	Liu's		128.60	27.33	27.41
	Ours		128.60	27.05	28.10
	JM9.8	256	256.36	29.66	29.60
	Liu's		255.90	30.22	29.97
	Ours		256.88	29.96	30.56

Some examples of the subjective visual quality using our face ROI based coding scheme are illustrated in Figure 6-11 to Figure 6-14, some of which clearly show the significantly subjective quality improvement in the visual attention areas. In those figures, pictures in (a) are the original frames and pictures in (b) show the extracted face ROI results. It can be seen that the face regions based on the extracted contours, shown as the solid red rectangles in the pictures, are more accurate than the results obtained directly from Viola's face detection approach shown in the dashed rectangles. Note that the ROI PSNR values listed in the tables are calculated in the solid red rectangles. The reconstructed frames are shown in (d), (e) and (f), which corresponds to the conventional rate control method in H.264/AVC, Liu's skin tone ROI based method and our method, respectively. It can be seen that while the skin tone based ROI method offers a better visual quality than the H.264/AVC, our method produce even better visual quality in the visual attention area (face region), especially for those shown in Figure 6-10.

Figure 6-11 Coding frame #30 in Foreman.Cif sequence with 64k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map; (d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.

Figure 6-12 Coding frame #80 in Foreman.CIF sequence with 128k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map; (d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.

Figure 6-13 Coding frame #45 in Salesman.QCIf sequence with 28k bps. (a) Original frame; (b) ROI detection result; (c). Bit allocation priority map; (d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.

Figure 6-14 Coding frame #62 in News.Cif sequence with 28k bps. (a) Original frame; (b) ROI detection result; (c) Bit allocation priority map; (d), (e), (f) Reconstructed frame using JM9.8, skin tone ROI and our method, respectively.

6.6. Summary

It is known that one tends to concentrate on the faces when watching a head-and-shoulder types of conversational video sequence. Therefore, in this chapter, human faces in the conversational video scene have been exploited to improve coding efficiency and subjective visual quality by encoding the face ROIs more accurately than the rest of the video content, especially giving more emphasis to the important facial features such as mouth, eyes and face contours that are usually visual attention areas. Bits are allocated according to a proposed facial geometry priorities model for coding the face regions more finely and effectively at the expense of coarser coding of less important background regions.

In order to identify the face regions in the video scene effectively and efficiently, Viola's face detection method has been adopted and improved by an efficient motion-based face sub-image rejection scheme. The motion information can be efficiently computed by using an integral image which is also utilized in obtaining the feature vectors in the face detection approach. More accurate face regions are further extracted using a modified active contour model. This face region based coding method has been implemented based on the H.264/AVC reference software JM9.8. The comparative experimental results have validated the advantage of the proposed scheme.

Chapter 7.

Conclusion and Discussion

7.1. Conclusions

Video coding is the key part for the many multimedia applications. Various video compression standards (MPEG-1/2/4, H.261/263 and H.264/AVC etc.) have been developed. Although these standards have specified the decompression procedure, many issues regarding the implementations of compression algorithms are still open for improvements and optimization, and assorted techniques have been introduced to improve the compression ratio at the expense of the high computing complexity.

As the research of computer vision advances, more advanced techniques achieving more complicated goals are developed. It is rational to adopt these techniques in video coding to achieve high compression ratio with good subjective visual quality in a way that follows the property of human visual system. The motion information available in the natural video content provides additional information to improve both the efficiency and effectiveness of the algorithms when adopting of computer vision techniques.

In this thesis, we studied some schemes to enhance the coding effectiveness and efficiency by both improving the block matching algorithm from its distortion measurement perspective and proposing a new bit allocation method based facial geometry priority for rate control with the following contributions.

Firstly, the quantitative deviation of SAD from MSE in terms of maximum and average deviations was first analyzed theoretically. Based on the analysis, a new measure,

namely weighted sum of absolute difference (WSAD), was proposed to minimize the average deviation. The WSAD was derived in two-element case firstly, for which an optimal weight was obtained theoretically. The weight is approximately adapted to $\frac{1}{2}$ so that the weighting operation can be simplified to a shift operation, and the resulted multiplication free WSAD also shows better matching accuracy with a smaller deviation from MSE than SAD. A geometric interpretation was presented to illustrate how these metrics differ in measuring distortions. The two-element WSAD was then further extended to multi-element case in a multi-level pyramid structure. We applied the WSAD to block motion estimation for video coding as well as in vector quantization (VQ), which validated the improvements of the proposed WSAD over SAD.

Secondly, as the sum of absolute Hadamard transformed differences (SATD) is an important distortion metric applied in the latest video coding standard H.264/AVC to improve the coding efficiency in the subpixel motion estimation and the low-complexity rate-distortion optimization (RDO), we proposed an efficient way to compute the SATD without performing the Hadamard transform by considering its joint effect with the following SAD process in the SATD calculation. This was inspired by our observation of a geometric interpretation for the simplest 2-point SATD. Extended to 4×4 block SATD using two theoretical methods, the proposed transform-exempt SATD (TE-SATD) algorithm can save 38% and 17% computation operations than the conventional SATD calculation and the fast Hadamard transform based calculation respectively while offering the same coding performance. The simulation results on the H.264/AVC reference software validated the efficiency of the proposed scheme.

Thirdly, by examining the adverse effect on the coding efficiency of the SAD based on intra-subblock features, such as the subblock sum based SAD introduced recently to lower the computational load, we proposed both intra- and inter-subblock features based SAD measure for more effective block matching, which was found, interestingly, to be the subblock-based SATD. The new features are constructed in a multi-level structure, which can provide a flexible and scalable means in achieving tradeoff between computation load and coding efficiency. The simulation results using standard video sequences validated the effectiveness and efficiency of the proposed scheme in terms of

both coding performance and computation time, compared with other relevant feature based SAD measures.

Lastly, in order to improve the subjective visual reconstruction quality for coding the conversational video, a facial geometry based bit allocation priority scheme is proposed to allocate more bits for the important facial features according to the human visual system characteristics. A modified active contour model was used to extract the accurate face region after its rough location was identified using a robust learning based face detection. An efficient sub-image rejection method based on efficient motion detection results was introduced to improve the face detection in video. The experimental result applying this face region based coding demonstrated its advantage on improving the subjective visual quality significantly.

7.2. Recommendations for Future Work

Some possible directions of the future research are listed as follows:

- (a). When extending the WSAD from 2-element case to multi-element case, different grouping method may result in different WSAD values. For example, for a difference block vector $\mathbf{d} = (10, 8, 20, 4)$, there may exist grouping methods such as :

There are totally 6 (C_4^2) grouping methods for the four-element vector. The final WSAD result may slightly vary in different grouping methods. In this thesis, we have proposed the multi-level pyramid structure for extending the WSAD to multi-element case which has shown good experiment results.

The quantitative deviation analysis for different methods of extending the 2-element WSAD to the multi-element WSAD turns out to be a challenging

problem, which needs further investigation to find the best possible grouping method, although the improvement on the coding efficiency may be very limited as the proposed WSAD grouping approach has already performed very closely to the MSE.

- (b). In H.264/AVC, the SAD is used in integer pixel motion estimation, and the SATD is applied in fractional-pixel motion estimation. In this thesis, we have correspondingly studied the proposed WSAD to enhance the SAD in integer pixel motion estimation and the TE-SATD to speedup SATD in fractional pixel motion estimation, respectively. However, it is worthy investigating the performance comparison of WSAD and SATD in either integer or fractional pixel motion estimation, theoretically and experimentally. We have conducted some experiments with primary results showing that the proposed WSAD can perform better in the integer motion estimation while the SATD performs better in fractional motion estimation. The theoretical explanation behind has not been achieved, which is still open for further analysis in depth.
- (c). The face contour information obtained in this thesis provides potential segmentation and shape information for the advanced face video object (VO) based video coding and some practical video communication systems. We have developed a real-time face contour tracking system using a low-cost motorized Web-Cam based on the face contour extraction algorithms introduced in chapter 6. We do not include the part in order to keep the thesis focusing on video coding. Coupled with the face region based video coding introduced in this thesis, the Web-Cam based face contour tracking system could be further developed into a more advanced and useful video conferencing system.

Appendix

Appendix A.

For $\theta \in [0, \pi/4]$, find the extremums of the deviation function $D(\theta) = (r \cos(\theta) + w |r \sin(\theta)|) - r$.

Since $\theta \in [0, \pi/4]$, $D(\theta) = r \cos(\theta) + wr \sin(\theta) - r$, we have

$$D'(\theta) = \frac{dD(\theta)}{d\theta} = -r \sin(\theta) + wr \cos(\theta) \quad (8.1)$$

$$D''(\theta) = -r \cos(\theta) - wr \sin(\theta) \quad (8.2)$$

Therefore, for the deviation function, the maximum occurs at $\theta = \arctg(w)$, with maximum of $D_{\max} = r[\cos(\arctg(w)) + w \sin(\arctg(w)) - 1]$.

For SAD, which corresponds to $w=1$, $D(\theta)$ reaches maximum $D_{\max} = (\sqrt{2} - 1)r$ at $\theta = \pi/4$. Also $D(\theta)$ has minimum $D_{\min} = 0$ at $\theta = 0$. Similarly, the maximum and minimum of $D(\theta)$ can be obtained for $\theta \in [\pi/2, 2\pi]$. That is, for SAD, $D(\theta)$ reaches its maximum $D_{\max} = (\sqrt{2} - 1)r$ at $\theta = \pi/4, \pi 3/4, \pi 5/4$ or $\pi 7/4$ and its minimum $D_{\min} = 0$ at $\theta = 0, \pi/2, \pi$ or $\pi 3/2$.

Appendix B.

Suppose that the size of the input image is $x \times y$, the size of sub-images used for classification is $x_0 \times y_0$, the image scaling factor is r , and the S_x and S_y are the stepping sizes to crop sub-images from the original or scaled images in x - and y -direction

respectively. Determine the number of sub-images in the pyramid structure shown in Figure 6-1.

Since the pyramid structure scans the image with steps S_x and S_y in x - and y -direction respectively, the number of sub-image of size $x_0 \times y_0$ for the original image of size $x \times y$ can be obtained using

$$N_0 = \frac{(x-x_0)(y-y_0)}{S_x S_y} = \frac{xy}{S_x S_y} - \frac{xy_0 + x_0 y}{S_x S_y} + \frac{x_0 y_0}{S_x S_y} \quad (8.3)$$

Similarly, for the image scaled by r from the original image, we can calculate its sub-image number as:

$$N_1 = \frac{(\frac{x}{r} - x_0)(\frac{y}{r} - y_0)}{S_x S_y} = \frac{xy}{S_x S_y r^2} - \frac{xy_0 + x_0 y}{S_x S_y r} + \frac{x_0 y_0}{S_x S_y} \quad (8.4)$$

We can further sub-image numbers for all the scaled images, which together form a series as (including the one for the original image)

$$\frac{xy}{S_x S_y} - \frac{xy_0 + x_0 y}{S_x S_y} + \frac{x_0 y_0}{S_x S_y}, \dots, \frac{xy}{S_x S_y (r^2)^n} - \frac{xy_0 + x_0 y}{S_x S_y r^n} + \frac{x_0 y_0}{S_x S_y} \quad (8.5)$$

where integer n denotes the maximum scaled images in pyramid structure. The last scaled image should be at least the size of $x_0 \times y_0$, that is, both $\frac{x}{r^n} \geq x_0$ and $\frac{y}{r^n} \geq y_0$ must hold.

Hence, n is calculated as

$$n = \log_r \min\left(\frac{x}{x_0}, \frac{y}{y_0}\right) \quad (8.6)$$

Therefore, the total sub-image number from the pyramid structure can be calculated as (note that the three parts of all the terms in the series shown in Equation (8.5) can form three geometric progressions)

$$\begin{aligned} N = & \frac{xy}{S_x S_y} \frac{r^2 - \min([x_0/x]^2, [y_0/y]^2)}{r^2 - 1} \\ & - \frac{(x_0 y + xy_0)}{S_x S_y} \frac{(r - \min([x_0/x], [y_0/y]))}{(r-1)} \\ & + \frac{x_0 y_0}{S_x S_y} (1 + \log_r \min([x/x_0], [y/y_0])) \end{aligned} \quad (8.7)$$

Author's publications

Journal Paper

1. Bing Xiong and Ce Zhu, "A New Multiplication-Free Matching Criterion for Block Motion Estimation," *IEEE Trans. on Circuit and System for Video Technology*, accepted, scheduled to appear in Oct 2008.
2. Bing Xiong and Ce Zhu, Charayaphan Charoensak and Wei Yu, "An FPGA Prototyping of Gabor-wavelets Transform for Motion Detection," *International Journal of Computer Sciences and Engineering Systems*, vol. 1, no. 1, pp 65-70, 2007.
3. Bing Xiong and Ce Zhu, "Efficient Block Matching Using Multi-level Intra- and Inter-Subblock Features: Subblock-Based SATD," *IEEE Trans. on Circuit and System on Video Technology*, accepted 2008.

Submitted Journal Paper

1. Ce Zhu, Bing Xiong, "Transform-Exempt Calculation of Sum of Absolute Hadamard Transformed Differences (SATD)," *IEEE Trans. on Circuit and System on Video Technology*, revised 2008.
2. Bing Xiong, Ce Zhu, Charayaphan Charoensak and Yu Wei, "Rapid FPGA Prototyping of Gabor-Wavelet Transform Based Motion Estimation," *Journal of Signal Processing Systems* revised 2008.
3. Bing Xiong and Ce Zhu, "Face Region Based Bit Allocation for Conversational Video," *IEEE Trans. on Circuit and System on Video Technology*, submitted 2008.

Conference Paper

1. Xiong Bing and Charayaphan Charoensak, "Face Contour Tracking in Video," *International conference on image processing (ICIP2004)*, pp. 1021-1024, 2004.
2. Xiong Bing, Yu Wei and Charayaphan Charoensak, "Automatic Focusing Technique for Face Detection and Face Contour Tracking", *International Workshop on Biomedical Circuits and Systems (BioCAS)*, vol. s3.2, pp. 9-12, 2004.
3. Xiong Bing, Yu Wei and Charayaphan Charoensak, "Face Contour Extraction Using Snake," *International Workshop on Biomedical Circuits and Systems*, vol. s3.2, pp. 5-8, 2004.
4. Yu Wei, Xiong Bing and Charayaphan Charoensak, "Multi-Weight Diameter Bisection Method for Ellipse Detection," *International Workshop on Biomedical Circuits and Systems*,

vol. s3.3, pp. 17-20, 2004.

5. Yu Wei, Xiong Bing and Charayaphan Charoensak, "FPGA Implementation of Adaboost Algorithm For Detection of Face Biometrics," *International Workshop on Biomedical Circuits and Systems*, vol. s1.6, pp. 17-20, 2004.
6. Charayaphan Charoensak, Farook Satta, Yu Wei and Xiong Bing, "Design of FPGA Hardware for A Real-Time Blind Source Separation of Fetal ECG Signals," *International Workshop on Biomedical Circuits and Systems*, vol. s1.6, pp. 13-16, 2004.
7. Xiong Bing and Charayaphan Charoensak, "Rapid FPGA Prototyping of Gabor-Wavelet Transform for Applications in Motion Detection," *IEEE 7th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, vol. 3, pp. 1653–1657, 2002.

Bibliography

- [1]. Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s: Video, ISO/IEC 11172-2(MPEG-1 Video), 1993.
- [2]. Information Technology--Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC 13818-2, November 1995.
- [3]. Information Technology-Generic Coding of Audio-Visual Objects Part 2: Visual, ISO/IEC 14496-2(MPEG-4 Video), 1999.
- [4]. Standardization Sector of ITU, "Video Coding for Low Bitrate Communication," ITU-T Rec. H263, 1996.
- [5]. Joint Video Team of ITU-T and ISO/IEC JTC 1, "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, JVT-G050, Mar. 2003.
- [6]. M. H. Yang, N. Ahuja, *Face Detection and Gesture Recognition for Human-Computer Interaction*, Kluwer Academic Publishers, 2001.
- [7]. W. Lai; X. D. Gu, R. H. Wang, W. Y. Ma and H. J. Zhang, "A Content-based Bit Allocation Model for Video Streaming," *International Conference on Multimedia and Expo*, vol. 2, pp. 1315 - 1318, June 2004.
- [8]. L. Itti, "Automatic Foveation for Video Compression Using a Neurobiological Model of Visual Attention," *IEEE Trans. on Image Processing*, vol. 13, no. 10, pp. 1304-1318, 2004.
- [9]. C. W. Tang, "Spatiotemporal Visual Considerations," *IEEE Trans. on Multimedia*, vol. 9, no. 2, pp. 231-238, 2007
- [10]. Y. Liu, Z. G. Li and Y. C. Soh, "Region-of-Interest Based Resource Allocation for Conversational Video Communication of H.264/AVC," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 18, no. 1, pp. 134 - 139, Jan. 2008.

- [11]. M. H. Yang, D. Kriegman and N. Ahuja, "Detecting Face in Image: A survey", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, No. 1, 34-58, January, 2002.
- [12]. E. Hjelm, B. K. Low, "Face Detection: A Survey," *Computer Vision and Image Understanding*, vol. 83, pp. 236-274, 2001.
- [13]. K. K. Sung, T. Poggio, "Example-based Learning for View-based Human Face Detection," A. I. Memo No. 1521, C.B. C. L. Paper No. 112, A.I. laboratory, MIT, 1994.
- [14]. K. K. Sung, T. Poggio, "Example-based Learning for View-based Human Face Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39 -51, 1998.
- [15]. H. A. Rowley, S. Baluja and T. Kanade, "Neural Network-based Face Detection," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 203-208, 1996.
- [16]. H. A. Rowley, S. Baluja, and T. Kanade, "Neural Network-based Face Detection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, 1998.
- [17]. H. A. Rowley, S. Baluja and T. Kanade, "Rotation Invariant Neural Network-based Face Detection," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 38-44, 1999.
- [18]. H. A. Rowley, "Neural Network-based Face Detection," PhD thesis, Carnegie Mellon University, 1999.
- [19]. M. H. Yang, D. Roth and N. Ahuja, "A SNoW-Based Face Detector," *Advances in Neural Information Processing Systems*, vol. 12, pp. 855-861, 2000.
- [20]. P. Viola, M. Jones, "Robust Real-Time Face Detection," *2nd International workshop on statistical and computational theories of vision – modeling, learning, computing, and sampling*, 2001.
- [21]. P. Viola, M. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [22]. Y. Wang, J. Osterman. Y. Q. Zhang, *Video Processing and Communications*, Prentice Hall, 2002.
- [23]. S. R. Gunn, M. S. Nixon, "Global and Local Active Contours for Head Boundary Extraction," *International Journal of Computer Vision*, vol. 30, no. 1, pp. 43-54, 1998.
- [24]. X. Jia, M. S. Nixon, "Extending The Feature Vector for Automatic Face Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 12, pp. 1167-1176, 1995.
- [25]. M. Pardo, E. Sayrol, "Motion Estimation Based Tracking of Active Contours," *Pattern Recognition Letters*, vol. 22, pp. 1447-1456, 2001.

- [26]. S. R. Gunn, M. S. Nixon, "A Dual Active Contour for Head and Boundary Extraction," *IEE Colloquium on Image Processing for Biometric Measurement*, pp. 1/6-4/6, 1994.
- [27]. K. M. Lam, H. Yan, "Locating Head Boundary by Snakes," *International Symposium on Speech, Image Processing and Neural Networking*, pp. 13-16, 1994.
- [28]. K. N. Ngan, T. Meier and D. Chai, *Advanced Video Coding: Principles and Techniques*, Elsevier Science B.V., 1999.
- [29]. J. L. Barron, D. J. Fleet and S.S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43-77, 1994.
- [30]. S. C. Pei; L. G. Liou, "Motion-based Grouping of Optical Flow Fields: the Extrapolation and Subtraction Technique," *IEEE Trans. on Image Processing*, vol. 6, no. 10, pp. 1358-1363, 1997.
- [31]. M. Subbarao, "Interpretation of Image Flow: A Spatio-temporal Approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 3, pp. 266-278, 1989.
- [32]. S. Sull, B. Sridhar, "Runway Obstacle Detection by Controlled Spatiotemporal Image Flow Disparity," *IEEE Trans. on Robotics and Automation*, vol. 15, no. 3, pp. 537-547, 1999.
- [33]. A. M. Tekalp, *Digital Video Processing*, Prentice-Hall, Upper Saddle River, NJ, 1995.
- [34]. A. Singh, *Optic Flow Computation: A Unified Perspective*, IEEE Computer Society Press, 1991.
- [35]. M. I. Sezan, R. L. Lagendijk, *Motion Analysis and Image Sequence Processing*, Kluwer Academic Publishers, 1993.
- [36]. C. Tziritas, C. Labit, *Motion Analysis for Image Sequence Coding*, Elsevier, 1994.
- [37]. B. G. Schunk, "Image Flow Segmentation and Estimation by Constraint Line Clustering," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 10, pp. 1010-1027, 1989.
- [38]. B. K. P. Horn, B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, 1981.
- [39]. S. S. Beauchemin, J. L. Barron, "The Computation of Optical Flow," *ACM Computing Survey*, vol. 27, no. 3, pp. 433-467, 1995.
- [40]. B. K. P. Horn, B. G. Schunck, "Determining Optical Flow," A.I. Memo No. 572, MIT Artificial Intelligence Laboratory, 1981.
- [41]. D. Heeger, "A Model for Extraction of Image Flow," *Proceeding of International Conference on Computer Vision*, pp. 181-190, 1987.
- [42]. D. J. Fleet, A. D. Jepson, "Stability of Phase Information," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 15, no. 12, pp. 1253-1268, 1993.

- [43]. D. J. Fleet, A. D. Jepson, "Computation of Component Image Velocity from Local Phase Information," *International Journal of Computer Vision*, vol. 5, no.1, pp.77-104, 1990.
- [44]. D. J. Fleet, *Measurement of Image Velocity*, Kluwer Academic Publishers, Norwell, 1992.
- [45]. L. Wiskott, "Segmentation from Motion: Combining Gabor- and Mallat-Wavelets to Overcome The Aperture and Correspondence Problems," *Pattern Recognition*, vol. 32, pp. 1751-1766, 1999.
- [46]. D. R. Walker, K. R. Rao, "Improved Pel-recursive Motion Compensation," *IEEE Trans. Comm.*, vol. 32, no. 10, pp. 1128-1134, 1984.
- [47]. J. D. Robbins, A. N. Netravali, "Recursive Motion Compensation: A Review," *Image Sequence Processing and Dynamic Scene Analysis*, pp. 76-103, Springer-Verlag, 1983
- [48]. J. N. Driessen, L. Boroszký and J. Biemond, "Pel-recursive Motion Field Estimation for Image Sequences," *Journal of Visual Communication and Image Representation*, vol. 2, no. 3, pp. 259-280, 1991.
- [49]. V. V. Estrela, *Regularized Pel-recursive Motion Estimation*, Ph. D thesis, Illinois Institute of Technology
- [50]. N. Vasconcelos, A. Lippman, "Empirical Bayesian Motion Segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no.2, pp. 217-221, 2001.
- [51]. H. Gharavi, M. Mills, "Block Matching Motion Estimation Algorithms – New Results," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 37, no. 5, pp. 649-651, 1990.
- [52]. K. Koya, K. Linuma, A. Hirano, Y. Iiyima, and T. Ishiguro, "Motion-compensated Interframe Coding for Video Conferencing," *Proc. NTC 81*, pp. G5.3.1-G5.3.5, New Orleans, LA, 1981.
- [53]. G. Golberg, *Digital Image Warping*, Los Alamitos, IEEE Computer Society Press, 1990.
- [54]. M. E. Izquierdo, "Disparity/Segmentation Analysis: Matching with An Adaptive Window and Depth-Driven Segmentation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 589-607, 1999.
- [55]. R. R. Schulta, M. G. Alford, "Multiframe Integration via The Projective Transformation with Automated Block Matching Feature Point Selection," *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3265-3268, March, 1999.
- [56]. A. Gersho, R. M. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1992.

- [57]. J. N. Kim, S. C. Byun, Y. H. Kim, and B. H. Ahn, "Fast Full Search Motion Estimation Algorithm Using Early Detection of Impossible Candidate Vectors," *IEEE Trans. on Signal Process.*, vol. 50, no. 9, pp. 2355-2365, 2002.
- [58]. L. M. Po, W. C. Ma, "A Novel Four-step Search Algorithm for Fast Block Motion Estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 4, pp. 313–317, Jun. 1996.
- [59]. C. Zhu, X. Lin and L. P. Chau, "Hexagon-Based Search Pattern for Fast Block Motion Estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, Aug. 2002.
- [60]. W. Li, E. Salari, "Successive Elimination Algorithm for Motion Estimation", *IEEE Trans. on Image Processing*, vol. 4, no. 1, Jan. 1995.
- [61]. C. H. Cheung, L. M. Po, "Adjustable Partial Distortion Search Algorithm for Fast Block Motion Estimation," *IEEE Trans. on Circuits System for Video Technology*, vol. 13, no. 1, pp.100-110, Feb. 2003.
- [62]. M. Brünig, W. Niehsen, "Fast Full-Search Block Matching," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 241 - 247, Feb. 2001.
- [63]. C. Zhu, W. S. Qi, W. Ser, "Predictive Fine Granularity Successive Elimination for Fast Optimal Block-Matching Motion Estimation," *IEEE Trans. on Image Processing*, vol. 14, no. 2, pp 213 - 221, Feb. 2005.
- [64]. B.Liu, A. Zaccarin, "New Fast Algorithms for The Estimation of Block Motion Vectors," *IEEE Trans. on Circuits and System for Video Technology*, vol. 3, no.2, pp.148–157, 1993.
- [65]. J.N. Kim, T.S. Choi, "A Fast Full-Search Motion-Estimation Algorithm Using Representative Pixels and Adaptive Matching Scan," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1040 - 1048 , Oct. 2000.
- [66]. J. S. Kim, R. A. Park, "Fast Feature-Based Block Matching Algorithm Using Integral Projections," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 5, pp. 968–971, 1992.
- [67]. X. Lee, Y.Q. Zhang, "A Fast Hierarchical Motion-Compensation Scheme for Video Coding Using Block Feature Matching," *IEEE Trans. on Circuits System Video Technology*, vol. 6 , no. 6, pp 627 – 635, Dec. 1996.
- [68]. V. A. Nguyen, Y. P. Tan, "Fast Block-Based Motion Estimation Using Integral Frames," *IEEE Signal Processing Letters*, vol. 11, no. 9, pp 744-747, Sept. 2004.

- [69]. V. A. Nguyen, Y. P. Tan, "Efficient Block-Matching Motion Estimation Based On Integral Frame Attributes," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp 375 - 385, March 2006.
- [70]. J. L. Mitchell, W. B. Pennebaker, C. E. Fogg and D. J. L. Gall, "*MPEG Video Compression Standard*," Chapman and Hall, 1996.
- [71]. J. Max, "Quantization for Minimum Distortion," *IRE Trans. Information Theory*, vol. 6, March 1960.
- [72]. N. S. Jayant and P. Noll, *Digital Coding of Waveform*, Englewood Cliffs, Prentice Hall, 1984.
- [73]. W. Ding, "Joint Encoder and Channel Rate Control of VBR Video over ATM Networks," *IEEE Trans. on Circuits and System for Video Technology*, vol. 7, pp. 266-278, 1997.
- [74]. "MPEG-2 test model 5," Doc. ISO/IEC JTC1/SC29 WG11/93-400, April 1993.
- [75]. H. J. Lee, T. H. Chiang and Y. Q. Zhang, "Scalable Rate Control for MPEG-4 Video," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, pp. 878-894, 2000.
- [76]. A. Vetro, H. Sun and Y. Wang, "MPEG-4 Rate Control for Multiple Video Objects," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 9, pp 186-199, 1999.
- [77]. Z. G. Li, F. Pan, K. P. Lim, G. Feng, X. Lin and S. Rahardja, "Adaptive Basic Unit Layer Rate Control for JVT," JVT-G012-r1, 7th Meeting, Pattaya II, Thailand, March, 2003.
- [78]. Y. Liu, Z. G. Li and Y. C. Soh, "A Novel Rate Control Scheme for Low Delay Video Communication of H.264/AVC Standards," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, no. 1, pp. 68-78, 2007.
- [79]. S. A. Sirohey, A. Rosendfeld, "Eye Detection in A Face Image Using Linear and Nonlinear Filters," *Pattern Recognition*, vol. 34, pp. 1367-1391, 2001.
- [80]. M. Kapfer, J.B. Pineau, "Detection Of Human Faces in Color Image Sequences with Arbitrary Motions for Very Low Bit-Rate Videophone Coding," *Pattern Recognition Letters*, vol. 18, pp. 1503-1518, 1997.
- [81]. N. Herodotou, K. N. Plataniotis and A. N. Venetsanopoulos, "Automatic Location and Tracking of The Facial Region in Color Video Sequences," *Signal processing: Image communication*, vol. 14, pp. 359-388, 1999.
- [82]. J. Cai, A. Goshtasby, "Detecting Human Faces in Color Images," *Image and Vision Computing*, vol. 18, pp. 63-75, 1999.
- [83]. L. Wiskott, J. M. Fellous, N. Kruger and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775-779, 1997.

- [84]. Y. Wang, B. Yuan, "A Novel Approach for Human Face Detection from Color Images Under Complex Background," *Pattern Recognition*, vol. 34, pp. 1983-1992, 2001.
- [85]. E. Osuna, R. Freud and F. Girosi, "Training Support Vector Machines: an Application to Face Detection," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.
- [86]. C. H. Lee, J. S. Kim and K. H Park, "Automatic Human Face Location in A Complex Background Using Motion and Color Information," *Pattern Recognition*, vol. 29, no. 11, pp. 1877- 1889, 1996.
- [87]. L. Gu, S. Z. Li and H. J. Zhang, "Learning Probabilistic Distribution Model for Multi-View Face Detection," Microsoft research China, 2001.
- [88]. Z. Zhang, M. Li, S. Z. Li and H. J. Zhang, "Multi-View Face Detection with Float boost," *Proceedings. Sixth IEEE Workshop on Applications of Computer Vision*, pp. 184–188, 2002.
- [89]. T. Agui. Y. Kokubo, H. Nagashashi and T. Nagao, "Extraction of Face Recognition from Monochromatic Photographs Using Neural Networks," *Proceeding of The Second International Conference On Automation, Robotics And Computer Vision*, vol. 1, pp. CV-18.8.1-CV-18.8.5, 1992.
- [90]. B. Froba, C. Hublbeck, "Robust Face Detection at Video Frame Rate Based on Edge Orientation Features," *Proc. Of the 5th IEEE International Conference on Automatic Face and Gesture Recognition*, 2002
- [91]. Y. Dai and Y. Nakano, "Face-Texture Model Based on SGLD and Its Application in Face Detection in A Color Scene," *Pattern Recognition*, vol. 29, no. 6, pp. 1007-1017, 1996.
- [92]. M. F. Augusteijin, T. L. Skujca, "Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology," *Proc. of IEEE Conference on Neural Networks*, pp. 392-398, 1993.
- [93]. S. A. Sirohey, "Human Face Segmentation and Identification," Technical report CS-TR-3176, University of Maryland, 1993.
- [94]. H. P. Graf, T. Chen, E. Petajan, and E. Cosatto, "Locating Faces and Facial Parts," *Proceedings of 1st International Workshop on Automatic Face And Gesture Recognition*, pp. 41-46, 1995.
- [95]. M. Kirby, L. Sirovich, "Application of The Karhunen-Loeve Procedure for The Characterization of Human Faces," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, 1990.
- [96]. M. Turk, A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

- [97]. K. Sobottka, I. Pitas, "Face Localization and Feature Extraction Based On Shape and Color Information," *Proceedings of IEEE Conference on Image Processing*, pp. 483-486, 1996.
- [98]. D. Chai, K. N. Ngan, "Locating Facial Region of A Head-Shoulders Color Image," *Proceedings Of The Third International Conference On Automatic Face And Gesture Recognition*, pp. 124-129, 1998.
- [99]. B. K. Low, M. K. Ibrahim, "A Fast and Accurate Algorithm for Facial Feature Segmentation," *Proceedings of IEEE International Conference on Image Processing*, 1997.
- [100]. B. K. Low, "Computer Extraction of Human Faces," PhD thesis, De Montfort University, 1998.
- [101]. I. Crawley, F. Berard, "Multi-Modal Tracking of Faces For Video Communications," *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 640-645, 1997.
- [102]. J. Ng, S. Gong, "Multi-View Face Detection and Pose Estimation Using A Composite Support Vector Machine Across The View Sphere," *Proceedings of International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 14 – 21, 1999.
- [103]. Y. Li, S. Gong, J. Sherrah and H. Liddell, "Support Vector Machine Based Multi-View Face Detection and Recognition," *Image and Vision Computing*, vol. 22, pp. 413-427, 2004.
- [104]. B. Wu; H. Ai, C. Huang and S. Lao, "Fast Rotation Invariant Multi-View Face Detection Based on Real Adaboost," *Proceedings Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 79–84, 2004.
- [105]. H. Je, D. Kim, S. Y. Bang, "Human Face Detection in Digital Video Using SVM Ensemble," *Neural Processing Letters*, vol. 17, pp. 239-252, 2003.
- [106]. L. C. De Silva, K. Aizawa and M. Hatori, "Detection and Tracking of Facial Features By Using A Facial Feature Model and Deformable Circular Template," *IEICE Trans. Information System*, pp. 1195-1207, 1995.
- [107]. S. H. Jeng, H. Y. M. Liao, C. C. Han, M. Y. Chern and Y. T. Liu, "Facial Feature Detection Using Geometrical Face Model: An Efficient Approach," *Pattern Recognition*, vol. 31, 1998.
- [108]. M. C. Burl, T. K. Leung and P. Perona, "Face Localization via Shape Statistics," *International Workshop on Automatic Face And Gesture Recognition*, 1995.

- [109]. M. C. Burl, P. Perona, "Recognition of Planer Object Classes," *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 1996.
- [110]. W. Huang, Q. Sun, C. P. Lam and J. K. Wu, "A Robust Approach to Face And Eyes Detection From Images With Cluttered Background," *Proceedings of International Conference on Pattern Recognition*, 1998.
- [111]. G. Chow, X. Li, "Towards A System for Automatic Facial Feature Detection," *Pattern Recognition*, vol. 26, pp. 1739-1755, 1993.
- [112]. A. L. Yuille, P. W. Hallinan and D. S. Cohen, "Feature Extraction from Faces Using Deformable Templates," *International Journal of Computer Vision*, vol. 8, pp. 99-111, 1992.
- [113]. A. Lanitis, A. Hill, T. Cootes and C. Taylor, "Locating Facial Features Using Genetics Algorithms," *Proceedings of International Conference on Digital Signal Processing*, pp. 520-525, 1995.
- [114]. A. Lanitis, C. Taylor and T. Cootes, "Automatic Interpretation and Coding of Face Images Using Flexible Models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 743-756, 1997.
- [115]. H. Schneiderman and T. Kanade, "Probabilistic Modeling of Local Appearance And Spatial Relationships For Object Recognition," *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 45-51, 1998.
- [116]. A. Rajagopalan, K. Kumar, J. Karlekar, R. Manivasakan, M. Patil, U. Desai, P. Poonacha, and S. Chaudhuri, "Finding Faces in Photographs," *Proceedings of International Conference on Computer Vision*, pp. 640-645, 1998.
- [117]. M. S. Lew, "Information Theoretic View-Based and Modular Face Detection," *Proceedings of International Conference On Automatic Face And Gesture Recognition*, pp. 198-203, 1998.
- [118]. M. Elad, Y. Hel-Or and R. Keshet, "Rejection Based Classifier for Face Detection," *Pattern Recognition Letters*, vol. 23, no. 12, pp. 1459-1471, 2002.
- [119]. R. W. D. Nickalls. "A New Approach to Solving The Cubic: Cardan's Solution Revealed," *The Mathematical Gazette*, vol. 77, pp. 354-359, 1993.
- [120]. C. Zhu, L. M. Po, "Minimax Partial Distortion Competitive Learning for Optimal Codebook Design," *IEEE Trans. on Image Processing*, vol. 7, no. 10, pp. 1400-1409, Oct 1998.
- [121]. "ITU-T Recommendation H.263 Software Implementation," Digital Video Coding Group, Telenor R&D, 1995.

- [122]. Joint Video Team (JVT) Reference Software (Version 8.5), Available: <http://iphome.hhi.de/suehring/tml/>
- [123]. C. P. Fan, J. F. Yang, "Fixed-Pipeline Two Dimensional Hadamard Transform Algorithms," *IEEE Trans. on Signal Processing*, vol. 45, no. 6, pp. 1669-1674, Jun. 1997.
- [124]. C. H. Tseng, H. M. Wang and J. F. Yang, "Enhanced Intra-4×4 Mode Decision for H.264/AVC Coders", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 8, pp 1027-1032, Aug., 2006.
- [125]. T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and Gary J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688–703, Jul. 2003.
- [126]. L. M. Po, K. Guo, "Transform-Domain Fast Sum of The Squared Difference Computation for H.264/AVC Rate-Distortion Optimization," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 17, No. 6, pp. 765 – 773, Jun. 2007.
- [127]. Alexander Graham, "*Kronecker Products and Matrix Calculus with Applications*," Halsted Press, John Wiley and Sons, NY, 1981.
- [128]. Fan-di Jou, "Method for fast SATD estimation," U.S. Patent 2007/019862 A1, Aug. 23 2007.
- [129]. C. M. Privitera and L. W. Stark, "Algorithms For Defining Visual Regions-Of-Interest: Comparison with Eye Fixations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 970–982, 2000.
- [130]. B. A. Wandell, *Foundations of Vision*, Sinauer Associates, Sunderland, MA 01375, 1995.
- [131]. P. C. Yuen, G. C. Feng and J. P. Zhou, "A Contour Detection Method: Initialization and Contour Model," *Pattern Recognition Letters*, vol. 20, pp. 141-148, 1999.
- [132]. M. Pardàs, A. Bonafonte, "Facial Animation Parameters Extraction and Expression Detection Using HMM Signal Processing," *Image Communication*, vol. 17, pp. 675-688, 2002.
- [133]. M. Kass, A. Witkin and D. Terzopoulos, "Snake: Active Contour Models," *International Journal of Computer Vision*, vol. 1, pp. 321-331, 1988.
- [134]. C. Xu, J. L. Prince, "Snakes, Shapes, and Gradient Vector Flow," *IEEE Trans. on Image processing*, vol. 7, no. 3, pp. 359 - 369, 1998.
- [135]. J. Wang, X. Li, "Controlled Accurate Searches with Balloons," *Pattern Recognition*, vol. 36, no. 3, pp. 827-843, 2003.
- [136]. B. Xiong, W. Yu and C. Charoensak, "Face Contour Tracking in Video Using Active Contour Model," *International Conference on Image Processing*, vol. 2, pp. 1021 - 1024, 2004.

- [137]. A. A. Amini, T. E. Weymouth and R. R. Jain, "Using Dynamic Programming for Solving Variational Problems In Vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855-867, 1990.
- [138]. Joint Video Team (JVT) Reference Software (Version 9.8), Available: <http://iphone.hhi.de/suehring/tml/>
- [139]. Intel Open Source Computer Vision Library (Version beta 3.1), Available: <http://www.intel.com/technology/computing/opencv/>