國 立 交 通 大 學

資訊科學與工程研究所

博 士 論 文

高效能視訊壓縮之先進動態補償預估方法

Advanced Motion-Compensated Prediction (MCP) for High-Efficiency Video Coding

研 究 生：陳漪紋

指導教授：彭文孝 教授

李素瑛 教授

中 華 民 國 一百 年 九 月

高效能視訊壓縮之先進動態補償預估方法

Advanced Motion-Compensated Prediction (MCP) for High-Efficiency Video Coding

研 究 生：陳渏紋　　　Student：Yi-Wen Chen

指導教授：彭文孝　　　Advisor：Wen-Hsiao Peng

李素瑛　　　Suh-Yin Lee

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

博 士 論 文

A Dissertation

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

October 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年十月

誌　謝

首先，最感謝的是指導教授彭文孝老師與李素瑛老師。博士班期間，彭老師與李老師在研究上的諄諄善誘與耐心教誨，才讓我得以成就此篇博士論文。李老師在生活態度、待人處世，以及各方面應對進退上給我的教導，都讓我終身受益無窮。感謝杭學鳴教授在計畫書口試、校內口試以及校外口試時提供寶貴的建議與鼓勵。感謝所有口試委員：蔡文錦教授、陳宏銘教授、張寶基教授、陳美娟教授與郭天穎教授在口試過程中不吝提供多年的珍貴研究經驗，充實了本論文的深度與廣度，使本論文更趨完善。諸位口試委員都是我在學術研究上最佳學習典範。

最後要謝謝家人給予我的關懷與支持讓我在博士求學過程中無後顧之憂。一路走來，家人總是給予我溫暖的關懷讓我能夠勇往直前的動力。有了他們的辛苦與支持，才有今日的我。感恩家人與其它親友對我的祝福與勉勵。

要感謝的人很多，在此向所有曾經幫助關心過我的人，致上最真切的謝意。

僅以此論文，獻給關心與幫助過我的大家。

高效能視訊壓縮之先進動態補償預估方法

研究生: 陳渏紋　　　　　　指導教授：　彭文孝　教授
李素瑛　教授

國立交通大學資訊工程學系

摘要

動態補償預估方法(Motion-Compensated Prediction，MCP)能移除視訊訊號在時間軸上的重複性，因此是許多視訊壓縮標準中常見的壓縮技術。雖然動態補償預估方法已經被提出並且研究超過 20 年，本論文仍將從理論、應用與實作等不同面向來重新探討動態補償預估方。

首先，我們以新的觀點重新解讀動作補償預估機制的運作，我們將動態補償預估方法視為兩個步驟；第一個步驟為運動向量取樣，第二個步驟則為利用取樣所得之運動向量作像素預估值的估算。我們同時提出理論的分析來支持我們提出之新觀點並用以驗證現存常見之不同動態補償預估方法例如區塊動態補償(Block Motion Compensation，BMC)、SKIP 預估方法與樣板比對預估方法(Template Matching Prediction)等等。實驗結果也證明提出之架構能準確分析各種不同的動態補償預估方法。

承續上述觀點，我們提出了參數化交疊區塊動作補償(Parametric Overlapped Block Motion Compensation，POBMC)的技術來加強 MCP 的效率。傳統的區塊動作補償(OBMC)是用來解決區塊動作補償(BMC)所具有之動作不確定性(Motion Uncertainty)的問題，藉由考慮鄰近區塊動作估測(Block Motion Estimation，BME)的結果，來做亮度值的估測。OBMC 已被證實能夠提供較 BMC 為佳的編碼效率。然而在 H.264/AVC 採用了可變區塊大小動作補償(Variable Block Size Motion Compensation，VBSMC)的技術下，OBMC

與 VSBMC 的結合使得 OBMC 使用的權重計算與儲存變成了一大挑戰。我們透過亮度與動作自相關係數的理論模型，以及將 BME 產生的運動向量近似為區塊中心點動作向量的假設，提出了 POBMC 技術。此技術根據每個像素點各自所有的鄰近動作向量以及此像素點到各動作向量對應的取樣點(區塊中心點)距離，來分配最佳的權重以達到最佳的 MCP 效能。

最後，我們利用提出之參數化交疊區塊動作補償架構來結合樣版比對預估以及方塊動量補償預估。由於樣板比對所產生的運動向量是不需耗費位元傳送，因此所以提出之雙向預估模式只需要傳送一個方塊運動向量即可達到利用兩個運動向量作雙向預估之效果。延續所提出的運動向量取樣架構，當樣板比對所找出的運動向量被近似為樣板重心點的運動向量後，此結合預估可以藉由找出最佳方塊運動向量的取樣點來達到最佳的結合預估效率。此外由於樣板比對預估有運算複雜度的問題，所提出的特殊雙向預估架構更可彈性地利用任何解碼端可推導出之運動向量來取代樣版比對運動向量以達到降低複雜度的目的。實驗結果最終也證明所提出之雙向預估模式可以有效增進現行視訊壓縮效能。

在本論文中，我們首先將 MCP 的結構視為運動向量取樣及預估亮度場(Intensity Field)重建兩個部份。從此觀點出發，我們接著提出參數化交疊區塊動作補償的技術來加強 MCP 的效率。藉由提出的參數化交疊區塊動作補償架構，我們更進一步發展出一套特殊的雙向預估方法(Bi-Prediction)結合樣板比對(Template Matching)之運動向量與傳統之方塊運動向量來增加預估的效率。我們相信，延續本論文所提之 MCP 分析架構將有利於未來更多動態向量預估方法相關技術的改進以增進視訊壓縮之效率。

Advanced Motion-Compensated Prediction (MCP) for High-Efficiency Video Coding

Student: Yi-Wen Chen

Advisor: Prof. Wen-Hsiao Peng
Prof. Suh-Yin Lee

Department of Computer Science,

National Chiao Tung University

Abstract

Motion-Compensated Prediction (MCP) has been the most popular approach, in the block-based hybrid video coding framework, for removing temporal redundancy. This dissertation attempts to reexamine its design from a theoretical perspective, with an aim to expose undisclosed details that crucially determine its performance and to seek further improvements.

Firstly, we introduce an analytical interpretation of MCP by viewing its process as consisting of motion sampling followed by the reconstruction of a temporal predictor. In this context, block-based motion estimation acts as a motion sampler taking samples at block centers while block-based motion compensation (BMC) interpolates between motion samples using the nearest-neighbor rule to reconstruct the motion field. Such an interpretation clearly reveals the essence of various MCP schemes. We have shown that the distinction between BMC, SKIP prediction and template matching prediction (TMP) lies in the choice of motion sampling structure, and likewise, that the celebrated control grid interpolation (CGI) and overlapped block motion compensation (OBMC) outperforms BMC, because of using more sophisticated motion interpolation algorithms.

This new interpretation of MCP also helps us to conceptualize the combination of

OBMC with variable block-size motion partitioning, which was done heuristically in the H.263 standard. We cast this problem as forming a linear estimate of a pixel's intensity from motion samples taken on an irregular grid. To circumvent the difficulties arising from the least-squares solution, we express the optimal OBMC weights in closed form based on parametric signal assumptions. The computation of this parametric OBMC (POBMC) solution requires only the geometric relations between the prediction pixel and its nearby block centers, offering a generic framework capable of reconstructing a temporal predictor from any irregularly sampled motion vectors.

The last part of this dissertation proposes a novel bi-prediction scheme combining BMC and TMP, the design of which is another highlight of the motion sampling and reconstruction concept. This scheme attains bi-prediction performance with only one set of motion parameters. Specifically, we transform the problem of finding an optimized block motion vector based on the contribution from the template motion vector into that of searching for its optimal sampling location. The result is a particular type of geometry motion partitioning. This notion is further extended to enable a low-complexity, template-matching-free implementation.

The techniques above have been evaluated in several core experiments of the JCV-VC committee, showing very promising results. This demonstrates that when looking deeper into the underlying principles, it is possible to make further improvements to existing designs or bring completely new ideas. We believe the other components of the hybrid-based video coding framework can also be improved with the same philosophy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview of Dissertation

The advances in video production technology and the consumer demand have led to the ever-increasing demands for further video coding standard towards higher resolution (4Kx2K resolution and above) and particularlily better video quality. After the success of existing H.264/AVC video coding standard, ITU-T Video Coding Experts Group (VCEG) targeted a new generation of video compression technology that has substantially higher compression capability than the H.264/AVC standard. Thus, ITU VCEG and MPEG worked together again and formed the so-called Joint Collaborative Team on Video Coding (JCT-VC) in January 2010. A joint Call-for-Proposal (CfP) for High Efficiency Video Coding (HEVC) was issued[2] to collect promsing coding tools as a good starting point to develop next-generation video coding design.

In most of the modern video coding standards such as MPEG1, MPEG2, MPEG4, H.263, H.264 and HEVC, a hybrid block-based motion compensated DCT-like transform coding architecture is still utilized. Motion-compensated prediction (MCP) is the key to the success of the modern video coding standards, as it removes the temporal redundancy in video signals and reduces the size of bitstreams significantly. Although MCP has been studied for over twenty years, we

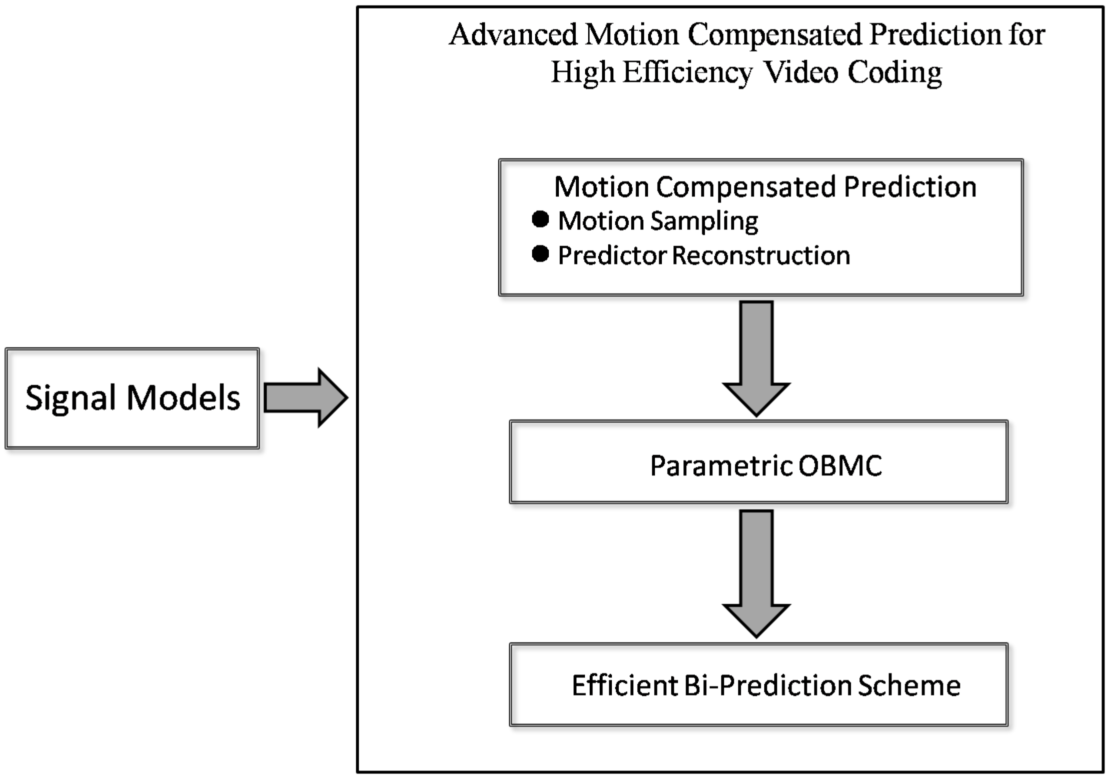

Advanced Motion Compensated Prediction for High Efficiency Video Coding
Signal Models
Motion Compensated Prediction
● Motion Sampling
● Predictor Reconstruction
Parametric OBMC
Efficient Bi-Prediction Scheme

Figure 1.1: The relationship between the proposed works and the signal models.

believe a deeper understanding of the principles behind the designs would bring a fundamental breakthrough in improving coding efficiency. In this dissertation, we therefore focus on improving MCP effeciency to provide better coding performance within a reasonable computation complexity overhead.

As shown in Fig. 1.1, to gain more insights into MCP, we first view MCP as a two stage process; it takes motion samples at block center and then generats predictor by the sampled motion vectors. Under this point of view and based on the presumed signal models for intensity and motion vectors, we then propose a parametric window design to tackle the problem of adapting overlapped block motion compensation (OBMC) windows for use with VBSMC. Lastly, we also demonstrate how template and block motion estimates can jointly be applied in a parametric overlapped block motion compensation (OBMC) framework to form an efficient bi-predictioin scheme to further improve temporal prediction. The

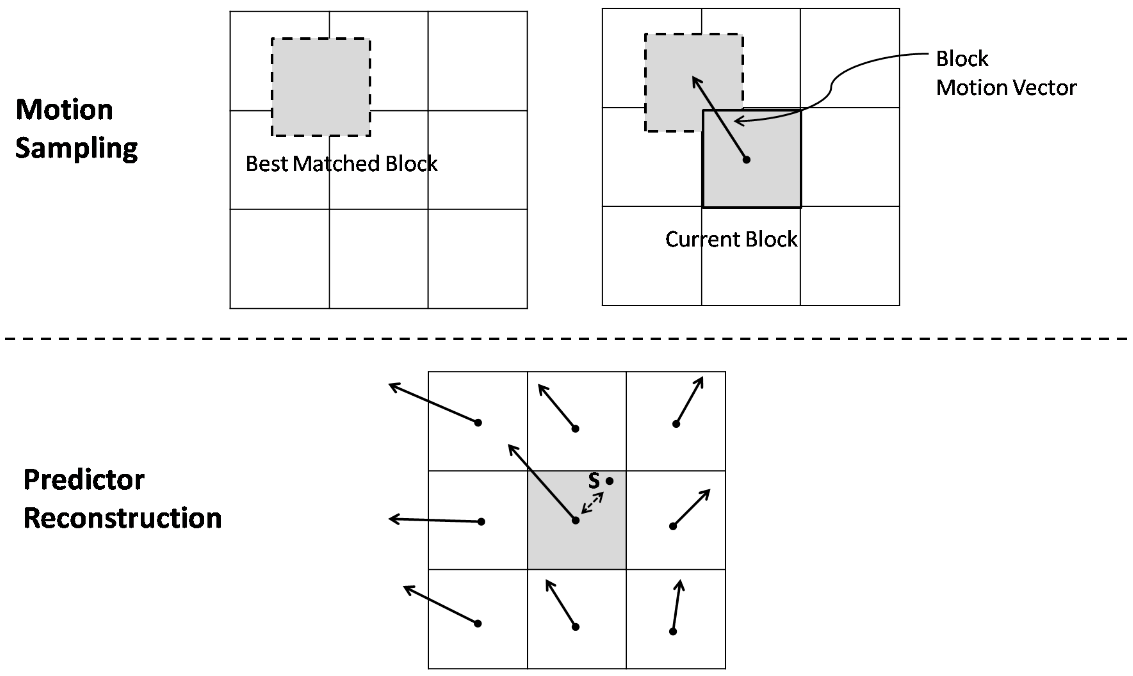


Figure 1.2: The operatioins of block motion compensation (BMC).

following summarizes our major contributions for developing MCP coding tools and some complexity reduced approaches.

1.2 Motion-Compensated Prediction (MCP): An Analytical Perspective

An insightful perspective on MCP is to view its process as consisting of sparse motion sampling followed by the reconstruction of temporal predictors. In this context, as illustrated in Fig. 1.1, block-based motion estimation acts as a motion sampler taking samples at block centers while BMC interpolates, using the nearest-neighbor rule, between motion samples to construct the motion field. This interpretation facilitates a better understanding of various MCP schemes from a unied framework. For example, if we take such a view, VBSMC is merely an enhancement of BMC in motion sampling as shown in Fig. 1.1.

The models are then applied to the analysis of prediction efficiency of various MCP schemes. To justify our theoretical analysis, we also show that template

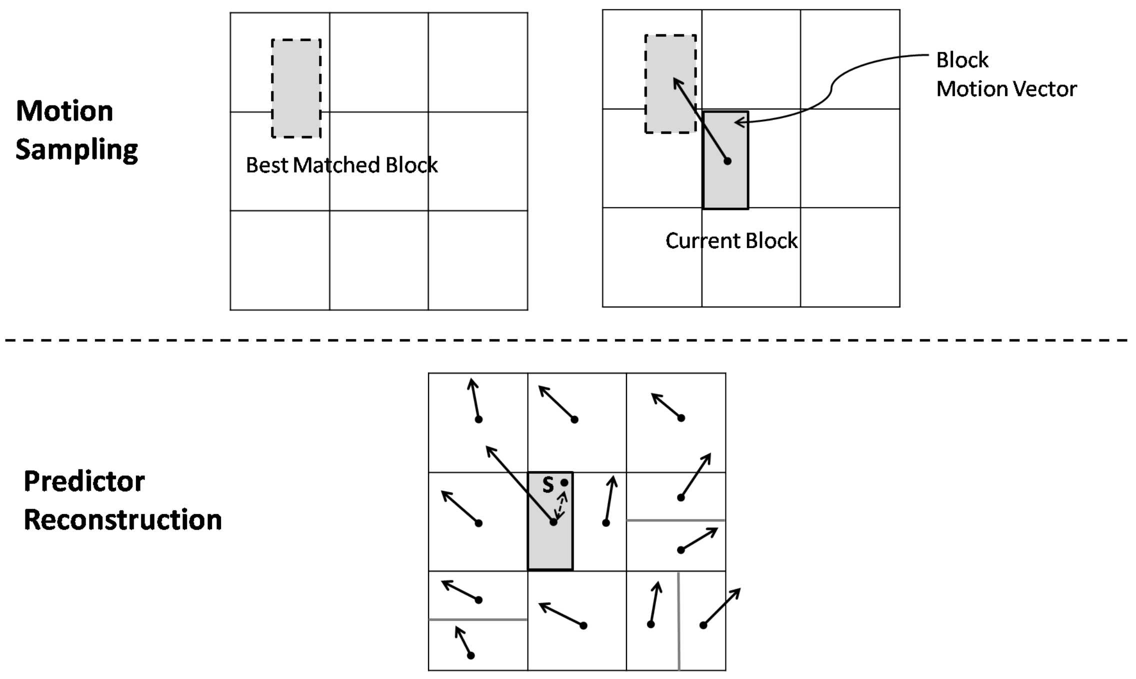


Figure 1.3: The operatioins of variable block size motion compensation (VBSMC).

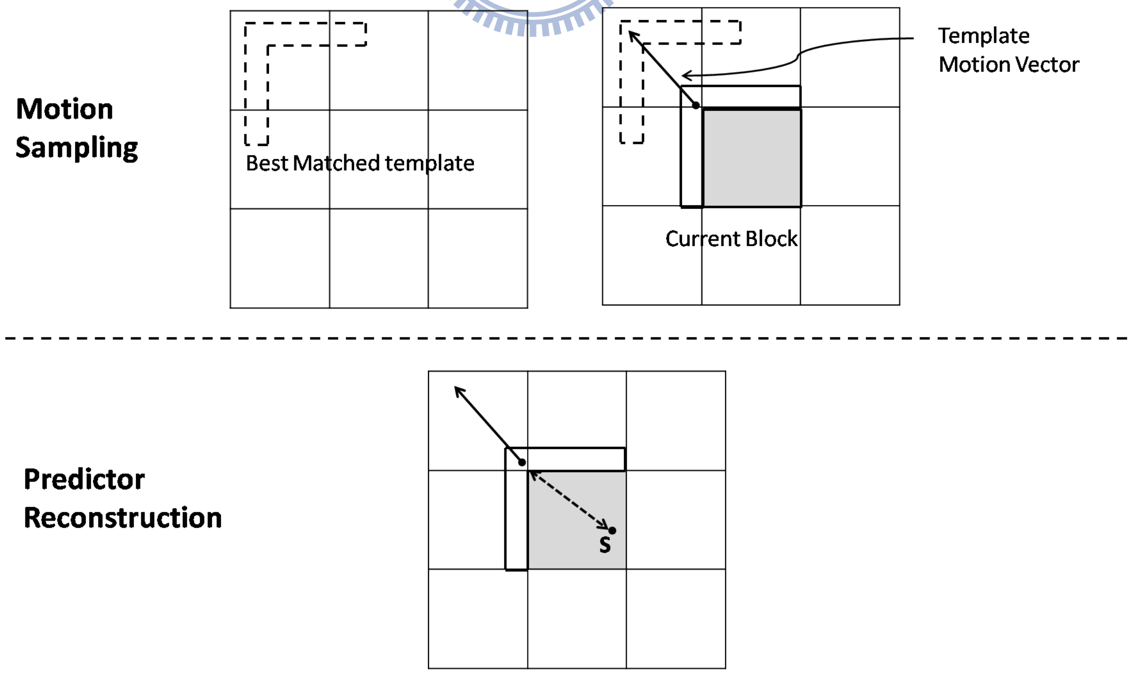


Figure 1.4: The operatioins of template matching prediction (TMP).

matching prediction (TMP), which estimates motion for a current block by using its surrending pixels (cf. Fig. 1.1), consistently outperforms SKIP prediction, but hardly competes with block motion compensation (BMC) unless both the motion and intensity fields are less random or have high spatial correlation.

To facilitate the analysis of various MVP schemes, we adopt the signal models, which assumes that the autocorrelation function of the intensity and motion fields follows some quadritic and exponential forms. Given these assumptions, we then examine the prediction error for BMC, CGI, OBMC, TMP and Skip predictioin. It is interesting that the mean-square error (MSE) of OBMC exhibits the same form as that of CGI, suggesting that OBMC and CGI have identical prediction efficiency and they outperform the other MCP scehemes in terms of prediction efficiency. Nevertheless, OBMC is generally preferable to CGI. The reasons are twofold. First, the true motion for every pixel is not easily accessible, which makes it difficult to estimate the weighting coefficients for CGI. Second, OBMC can not only alleviate motion uncertainty, but it also serves to attenuate quantization noises in reference pictures. These arguments also explain why OBMC normally outperforms CGI in practice.

With the above observations, we focus on improving OBMC. However, window design for OBMC becomes difficul when variable block size motion compensation (VBSMC) are incorporated. In an effort to adapt OBMC for use with VBSMC, we approach the problem using parametric solutions as detailed in the next subsection.

1.3 Parametric OBMC

This work adapts overlapped block motion compensation (OBMC) to suit variable block-size motion partitioning. The motion vectors (MVs) for various partitions are formalized as motion samples taken with an irregular grid. With this viewpoint, determining OBMC weights to associate with these samples becomes an under-determined problem since a distinct solution has to be sought for each prediction pixel.

We tackle this problem by expressing the optimal weights in closed-form based on parametric signal assumptions. The computation of this solution requires only the geometrical relationship between the prediction pixel and its nearby block centers, leading to a generic framework allowing for reconstructing temporal predictors from any irregularly sampled MVs. A modified implementation is also proposed to address MV location uncertainty and to reduce computational complexity.

Extensive experiments have been conducted using the KTA software. Experimental results demonstrate that our scheme performs better than similar previous works, and provides about 5% BD-Rate savings compared to H.264/AVC anchor. When compared to the recently proposed Quadtree-based Adaptive Loop Filter (QALF) [12]and Enhanced Adaptive Interpolation Filter (EAIF)[29], POBMC also shows a comparable gain. Furthermore, the combination of it with either filter gives a combined effect that is almost the sum of their separate improvements.

Along with other promising coding tools in KTA2.4, the proposed POBMC [8] was submitted, for subjective viewing tests, in response to the HEVC Call for Proposals issued jointly by MPEG and VCEG in April, 2010 [2]. It was ranked 12 overall and 10 with low delay configurations among 27 proposals, in terms of the average mean opinion score [3]. After the CfP competition in the 1st JCT-VC meeting, TMuC (Testing Model under Consideration) is constructed mainly from the best performer's codebase and the other top-performing HEVC proposals.

1.4 Bi-Prediction Combining TMP and BMC

An efficient bi-predictioin scheme combining TMP with BMC using POBMC is proposed. Template matching prediction (TMP), which estimates the motion for a target block by using its surrounding pixels, has been observed to perform efficiently in inter-frame coding. In this work, we expos how template and block motion estimates can jointly be applied in a parametric overlapped block motion compensation (OBMC) framework to further improve temporal prediction. When integrated in HM3.0, the reference software of HEVC, the combined technique is observed to achieve Y-BD-rate savings of 2% BD-rate reductioin. The notion is further extended to allow the template MV to be replaced with one of those MVs for neighboring prediction units, enabling a low-complexity and template-matching-free implementation. Experiments show that this reduced-complexity approach can provide competitive coding gain with lower computation complexity and memory access bandwidth.

Currently, the JCT-VC committeehas finished the HEVC working draft 4 and HEVC test model (HM) [23]. With its promising results and compatibility with existing tool features, the proposed new bi-prediction scheme which combines the implicitly inferred motion and block motion with POBMC is being evaluated in several formal core experiments in JCT-VC meeting[10][16][7][6].

1.5 Organization and Contribution

For more details of each part of the proposed advanced MCP schemes for High-Efficiency Video Coding, the rest of this dissertation is organized as follows:

- Chapter 2 introduces a new viewpoint by viewing MCP as a motion sampler taking motion sampling followed by a reconstruction of prediction signal.[26].

  - We have analyzed, both theoretically and empirically, the prediction efficiency of BMC, CGI, OBMC, TMP and SKIP prediction.
  - We have shown that although TMP hardly competes with BMC, it is shown to outperform SKIP prediction, which explains why the bit rate can be significantly reduced when TMP is efficiently combined with SKIP prediction.
  - We have shown that OBMC and CGI outperform other MCP schemes.

- Chapter 3 details the algorithm of Parametric Overlapped Block Motion Compensation (POBMC) [9].
  - Our scheme requires only the geometry relation to compute the weight vector for OBMC.
  - Compared to EAIF and QALF, our scheme shows a comparable gain. Furthermore, the combination of it with either filter gives a combined effect that is almost the sum of their separate improvements.
  - Our scheme is a suboptimal yet computationally efficient implementation, which need not solve the Wiener-Hopf equation and thus requires no matrix inverse computation.
  - By integrating POBMC into AVC/H.264 reference software KTA, our codec [8], submitted for subjective test in response to the Call for Proposals for Video Compression Technology issued jointly by ITU-T VCEG and ISO/IEC MPEG, ranks 12 overall (and 10 with Low Delay Settings) among 27 proposals.

- Chapter 4 introduces a bi-prediction scheme with only single motion overhead as for unidirectional prediction.

  - It combines motion vectors found by template and block matchings with OBMC
  - The concept of adaptive motion merging is incorporated to enable a template-matching-free implementation.
  - The proposed bi-prediction scheme is being evaluated using HEVC reference software and provides top one coding efficiency in the core experiment 1 of 6th JCTVC meeting.

- Lastly, Chapter 5 summarizes our works and illustrates the research activities in the future.

Chapter 2

Motion-Compensated Prediction: An Analytical Perspective

2.1 Introduction

Motion compensated prediction (MCP) is an algorithmic technique employed in the encoding/decoding of video data for removing temporal redundancy. In hybrid video coding schemes such as MPEG and H.264/AVC standards, pictures are predicted from previous or bidirectionally from previous and future pictures by a block-based motion compensation (BMC) scheme. It uses one single motion vector (MV) (two MVs for bipredictioin schemes) as an estimate of the true motion field for a block of pixels, in order to trade off the accuracy of motion representation for less overhead.

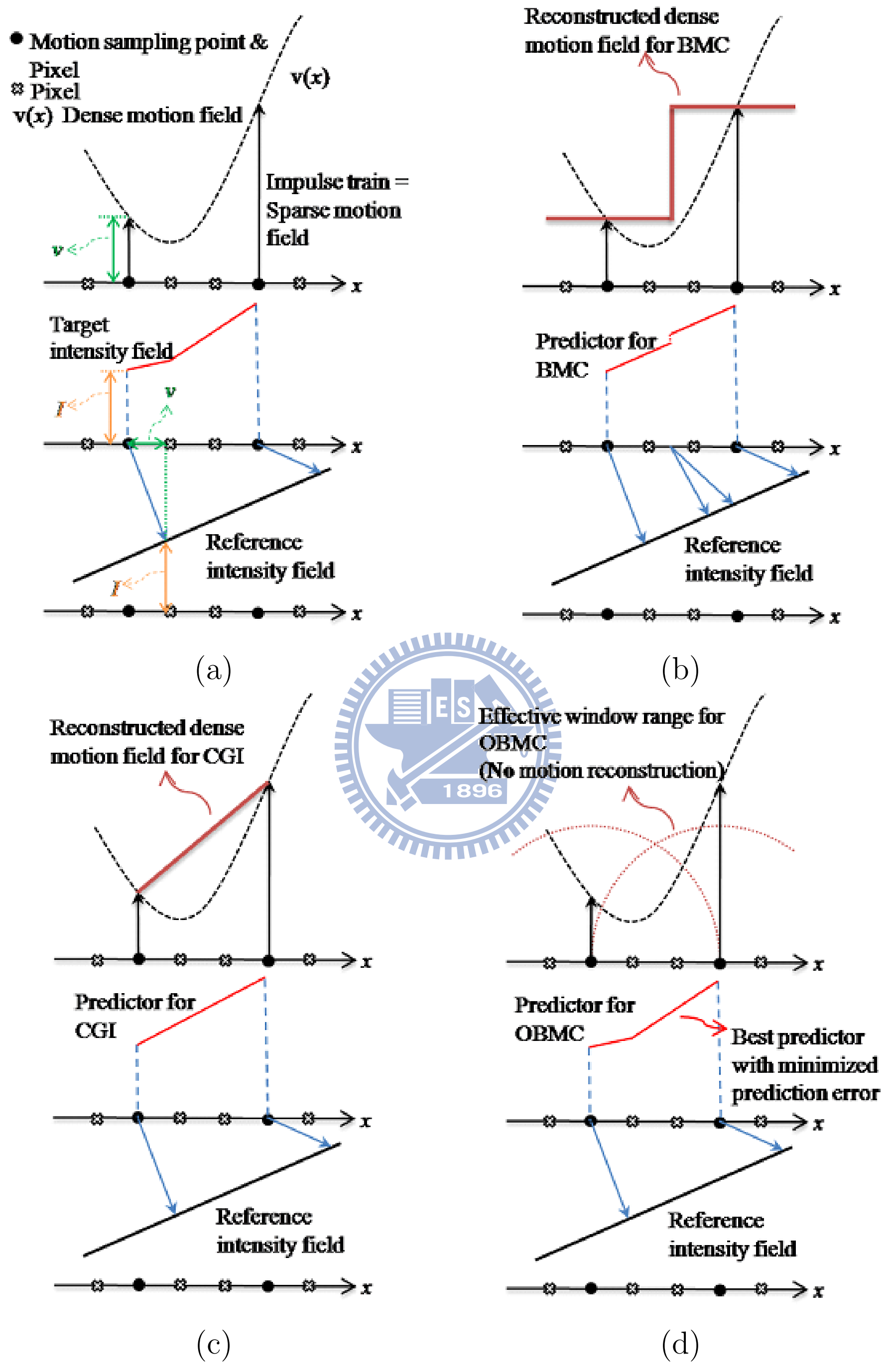
An insightful perspective on MCP is to view its process as consisting of sparse motion sampling followed by the reconstruction of temporal predictors. In this context, block-based motion estimation acts as a motion sampler taking samples at block centers while BMC interpolates, using the nearest-neighbor rule, between motion samples to construct the motion field. This interpretation facilitates a better understanding of various MCP schemes from a unified framework. For example, if we take such a view, VBSMC is merely an enhancement of BMC in motion sampling. The various MB partitionings are assimilated to different sampling structures, and choosing a specific block partitioning can be thought of as determining a local sampling pattern. By a similar reasoning, the difference between BMC and CGI is easily seen to be a different choice of motion interpolator. Somewhat less intuitive is OBMC, which does not directly reconstruct the motion field. Nevertheless, it was shown in [25] that an optimal OBMC window is also an optimal motion interpolation function, with which CGI can achieve the same mean-square prediction error as OBMC. This result furnishes another view of OBMC from the standpoint of motion interpolation. As an illustration, Fig. 2.1 contrasts graphically these techniques for the 1-D case.

With these ideas in mind, in the following sections, we shall first show that when chosen to minimize the mean-square block matching error, the MV is shown to approximate the true motion of the block center based on the motion and intensity fields of video signals. We then apply the statistical motion distribution model to the analysis of prediction efficiency of various MCP schemes such as Template Matching Prediction (TMP), BMC and SKIP prediction. The analytical results are justified by empirical experiments with typical image sequences.

2.2 Motion and Intensity Models

In this section, we review two statistical models used to characterize the motion and intensity fields of video signals. These models will serve as the basis for analyzing the motion compensation error of motion compensated prediction (MCP) in this dissertation.

To analyze the distribution of motion-compensated residuals, Tao *etal.* [24] assumes that the *autocorrelation* function of the intensity and motion fields can be



● Motion sampling point & Pixel
⊗ Pixel
v(x) Dense motion field
v(x)
Impulse train = Sparse motion field
v
x
Target intensity field
I
v
Reference intensity field
I
(a)
Reconstructed dense motion field for BMC
Predictor for BMC
Reference intensity field
(b)
Reconstructed dense motion field for CGI
Predictor for CGI
Reference intensity field
(c)
Effective window range for OBMC (No motion reconstruction)
Predictor for OBMC
Best predictor with minimized prediction error
Reference intensity field
(d)

Figure 2.1: Various MCP schemes in the 1-D case: (a) MCP based on the true motion field, (b) BMC, (c) CGI and (d) OBMC.

approximated with a quadratic function and an exponential function, respectively:

$$\begin{aligned} E[I_k(\mathbf{s}_1)I_k(\mathbf{s}_2)] &= \sigma_I^2\left(1-\tfrac{||\mathbf{s}_1-\mathbf{s}_2||_2^2}{K}\right) \\ E[v_x(\mathbf{s}_1)v_x(\mathbf{s}_2)] &= E[v_y(\mathbf{s}_1)v_y(\mathbf{s}_2)] = \sigma_m^2\rho_m^{||\mathbf{s}_1-\mathbf{s}_2||_1}, \end{aligned} \tag{2.1}$$

where $I_k(\mathbf{s})$ represents the intensity value of pixel $\mathbf{s} = (x(\mathbf{s}), y(\mathbf{s}))^T$ of frame k; $\mathbf{v}(\mathbf{s}) = (v_x(\mathbf{s}), v_y(\mathbf{s}))^T$ denotes its motion vector; and $\{\sigma_I^2, K\}$ and $\{\sigma_m^2, \rho_m\}$ are their respective variance and correlation coefficient. Likewise, in [30] Zheng *et al.* introduces a motion distribution model assuming that the difference between motion at two pixels obeys the normal distribution:

$$v_x(\mathbf{s}_1) - v_x(\mathbf{s}_2) \text{ or } v_y(\mathbf{s}_1) - v_y(\mathbf{s}_2) \sim N(0, \alpha\,\|\mathbf{s}_1 - \mathbf{s}_2\|_2^2), \tag{2.2}$$

where α is a constant indicating the degree of motion variation in the horizontal or vertical direction.

Given these models, they both show that the block-based motion estimate tends to be the motion of the block center $\mathbf{s}_c$, with the mean-square prediction error for pixel $\mathbf{s}$, $d(\mathbf{s}; \mathbf{v}(\mathbf{s}_c)) \equiv I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_c))$, given respectively by

$$E[d^2(\mathbf{s}; \mathbf{v}(\mathbf{s}_c))] = \frac{8\sigma_I^2\sigma_m^2}{K}\left(1 - \rho_m^{||\mathbf{s}-\mathbf{s}_c||_1}\right) \tag{2.3}$$

and

$$E[d^2(\mathbf{s}; \mathbf{v}(\mathbf{s}_c))] = \epsilon||\mathbf{s} - \mathbf{s}_c||_2^2, \tag{2.4}$$

where ϵ is a factor related to the randomness of the motion and intensity fields (the randomness increases with increasing ϵ). According to these equations, the prediction error is larger for boundary pixels, which agrees with the general observation.

2.3 Analysis of Various MCP Schemes

Given these statistical models, we next examine the prediction error for various MCP schemes including BMC, CGI[21], OBMC [19], TMP[13] and SKIP modes.

Assume at first the sampling structure is a square lattice. Such is the case when an image is divided into equally spaced square blocks for motion estimation.

2.3.1 Error Variance Distribution of BMC, CGI and OBMC

The prediction error of pixel $\mathbf{s}, \mathbf{s} \in \mathcal{B}$ for BMC, CGI and OBMC can be expressed respectively as

$$\begin{aligned} d^{BMC}(\mathbf{s}) &= I_k(\mathbf{s}) - I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_0)) \\ d^{CGI}(\mathbf{s}) &= I_k(\mathbf{s}) - I_{k-1}\left(\mathbf{s} + \sum_{i=0}^{3} w_i^{(c)}(\mathbf{s})\mathbf{v}(\mathbf{s}_i)\right) \\ d^{OBMC}(\mathbf{s}) &= I_k(\mathbf{s}) - \sum_{i=0}^{3} w_i^{(o)}(\mathbf{s}) I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i)) \end{aligned}$$

where $\{w_i^{(c)}(\mathbf{s})\}$ are chosen such that $\sum w_i^{(c)}(\mathbf{s})\mathbf{v}(\mathbf{s}_i)$ forms a vector LMMSE estimate of $\mathbf{v}(\mathbf{s})$ subject to the unit gain constraint[1]. By a similar approach, the weighting coefficients $\{w_i^{(o)}(\mathbf{s})\}$ and $\{w_i^{(ig)}(\mathbf{s})\}$ are derived to linearly estimate $I_k(\mathbf{s})$ based on the data sets $\{I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}_i))\}$ and $\{I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{t}_i))\}$, respectively. Particularly, in computing $\{w_i^{(ig)}(\mathbf{s})\}$ the motion vectors at $\mathbf{t}_i, i = 1, 2, 3$ are taken to be known, while during actual motion compensation they are interpolated from those of nearby block centers (with the results denoted by $\widetilde{\mathbf{v}}(\mathbf{t}_i)$).

The mean-square prediction error (MSE) for the four MCP schemes can be evaluated by using (2.1), although the algebra is a bit tedious. We shall thus use CGI as an example to indicate the main idea without going into formal details. To start off, the vector LMMSE estimator for $\mathbf{v}(\mathbf{s})$ is firstly found by combining the scalar estimator for each of its components. As such, $w_i^{(c)}(\mathbf{s})$ is a matrix-valued function (of dimension 2x2). However, a great simplification can be made since

[1]We consider this Wiener filter rather than bilinear filter [21] since our interest is in determining the theoretic limit of CGI.

(a) the horizontal and vertical motion fields are independent of each other and (b) they share an identical signal model as hinted in (2.1). The former makes the matrix become diagonal while the latter further equalizes the diagonal elements. Together the two conditions reduce $w_i^{(c)}(\mathbf{s})$ to a scalar, with its value given by the ith element of

$$\mathbf{w}^{(c)}(\mathbf{s}) = \mathbf{R}^{-1}\left[\mathbf{P} - \mathbf{U}\left(\frac{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{P} - 1}{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{U}}\right)\right], \tag{2.5}$$

where $\mathbf{U}$ is a unit vector and $\mathbf{R}_{ij} = E[v_x(\mathbf{s}_i)v_x(\mathbf{s}_j)]$ and $\mathbf{P}_j = E[v_x(\mathbf{s})v_x(\mathbf{s}_j)]$ for $0 \le i, j \le 3$.

To complete the evaluation of $E[\left\|d^{CGI}(\mathbf{s})\right\|^2]$, we still need to know $E\left[I_k^2(\mathbf{s})\right]$, $E[I_{k-1}^2(\mathbf{s} + \sum w_i^{(c)}(\mathbf{s})\mathbf{v}(\mathbf{s}_i))]$, and $E[I_k(\mathbf{s})I_{k-1}(\mathbf{s} + \sum w_i^{(c)}(\mathbf{s})\mathbf{v}(\mathbf{s}_i))]$. The first two terms, according to (2.1), are simply σ_I^2, while the last one can be computed by substituting (2.1) and (2.5) into (2.6).

$$\begin{aligned} &E\left[I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s}))I_{k-1}\left(\mathbf{s} + \sum_{i=0}^{3} w_i^{(c)}(\mathbf{s})\mathbf{v}(\mathbf{s}_i)\right)\right] \\ &= \sigma_I^2 E\left[1 - 2K^{-1}\left(\sum_{i=0}^{3} w_i^{(c)}(\mathbf{s})(v_x(\mathbf{s}) - v_x(\mathbf{s}_i))\right)^2\right] \end{aligned} \tag{2.6}$$

where we have used the fact that $\sum w_i^{(c)}(\mathbf{s}) = 1$. A straightforward computation then gives

$$\begin{aligned} E\left[\left\|d^{CGI}(\mathbf{s})\right\|^2\right] = \frac{f}{2}\sum_{0\le i,j\le 3} w_i^{(c)}(\mathbf{s})w_j^{(c)}(\mathbf{s})\,\Big(1 - \rho_m^{\|\mathbf{s}-\mathbf{s}_i\|_1} \\ -\rho_m^{\|\mathbf{s}-\mathbf{s}_j\|_1} + \rho_m^{\|\mathbf{s}_i-\mathbf{s}_j\|_1}\Big), \end{aligned} \tag{2.7}$$

with the scaling factor $f = 8\sigma_I^2\sigma_m^2 K^{-1}$. Following similar derivations to those for CGI, we can calculate the MSE for the other schemes as

$$E\left[\left\|d^{BMC}(\mathbf{s})\right\|^2\right] = f\left(1 - \rho_m^{\|\mathbf{s}-\mathbf{s}_0\|_1}\right)$$

$$E\left[\left\|d^{OBMC}(\mathbf{s})\right\|^2\right] = E\left[\left\|d^{CGI}(\mathbf{s})\right\|^2\right]_{\mathbf{w}^{(c)}(\mathbf{s})=\mathbf{w}^{(o)}(\mathbf{s})}$$

where $I_\Delta(i) = I_k(\mathbf{s}) - I_{k-1}(\mathbf{s}+\tilde{\mathbf{v}}(\mathbf{t}_i))$ and $E[I_\Delta(i)I_\Delta(j)]$ can be expanded and evaluated term by term through a calculation similar to (2.6).

It is interesting that the MSE of OBMC exhibits the same form as that of CGI, with $\mathbf{w}^{(o)}(\mathbf{s})$ substituting for $\mathbf{w}^{(c)}(\mathbf{s})$. Somewhat surprisingly, $\mathbf{w}^{(o)}(\mathbf{s})$ is found to be equal to $\mathbf{w}^{(c)}(\mathbf{s})$, suggesting that OBMC and CGI have identical prediction efficiency and that the OBMC filter $\mathbf{w}^{(o)}(\mathbf{s})$ is also a good motion interpolator. Nevertheless, OBMC is generally preferable to CGI. The reasons are twofold. First, the true motion for every pixel is not easily accessible, which makes it difficult to estimate $\mathbf{w}^{(c)}(\mathbf{s})$ for CGI. Second, OBMC can not only alleviate motion uncertainty, but it also serves to attenuate quantization noises in reference pictures. These arguments also explain why OBMC normally outperforms CGI in practice.

2.3.2 Error Variance Distribution of TMP

Template Matching Prediction (TMP) is a decoder-side motion derivation scheme. As shown in Fig. 2.3, TMP finds the predictor for a target block $\mathcal{B}$ by minimizing the predictor erro over the pixels in its immediate inverse-L-shaped neighborhood $\mathcal{B}$.To gain some insights into TMP, Fig. 4.1 plots the mean-square prediction error surface with a TMP MV for motion compensation of the target block. It is seen that this MV tends to minimize the prediction error in the upper left quarter, a result that is intuitively agreeable since it approximates the true motion associated with the template centroid. Although it has been observed that TMPcan provide coding gain[13], there is almost no satisfactory theoretical basis that clearly interprets the theoretical aspects of TMP thoroughly. In the following sections, we will first analyze the prediction efficiency of TMP followed and then the comparisons of TMP, BMC and Skip prediction. This section provides a theoretical analysis to

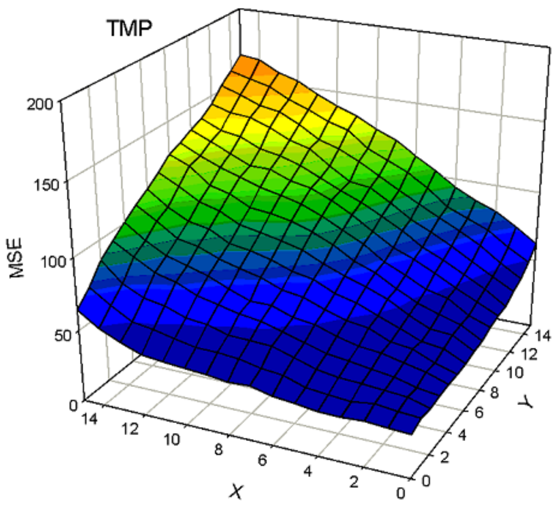


Figure 2.2: Mean-square prediction error surface of TMP using the sequence "Football".

expose the factors that determine the prediction efficiency of TMP. The analysis is carried out based on the statistical models introduced in previous section

We begin by examining the distribution of TMP error variance. To do so requires modeling the template motion estimate. Proceeding as the approach described in [30], we can obtain, with the results that

$$\mathbf{s}_t = \arg\min_{\mathbf{t}} \sum_{\mathbf{s}\in\mathcal{T}} E[d^2(\mathbf{s};\mathbf{v}(\mathbf{t}))] = \left(\frac{\sum_{\mathbf{s}\in\mathcal{T}} x(\mathbf{s})}{|\mathcal{T}|}, \frac{\sum_{\mathbf{s}\in\mathcal{T}} y(\mathbf{s})}{|\mathcal{T}|} \right)^T . \tag{2.8}$$

Thus, the motion estimate found by minimizing the template matching error is likely to be the motion associated with the *centroid* of the template, a result that is intuitively agreeable and is a direct extension of that for (rectangular) block matching.

As shown in Fig. 2.3, the centroid of the template $\mathbf{s}_t$ is obviously not at the block center when the template is straddled on the top and to the left of the target block $\mathcal{B}$. Thus we can expect TMP to yield higher prediction error than BMC for block $\mathcal{B}$. A little computation using $\mathbf{s}_t$ in place of $\mathbf{s}_c$ in (2.7) and (3.5) further shows that the error is lower in the upper left quarter and higher in the lower

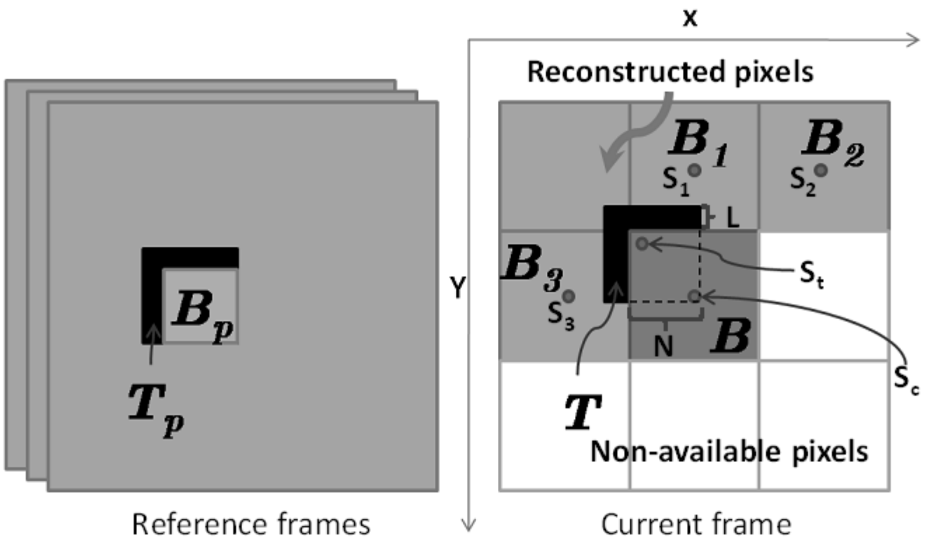


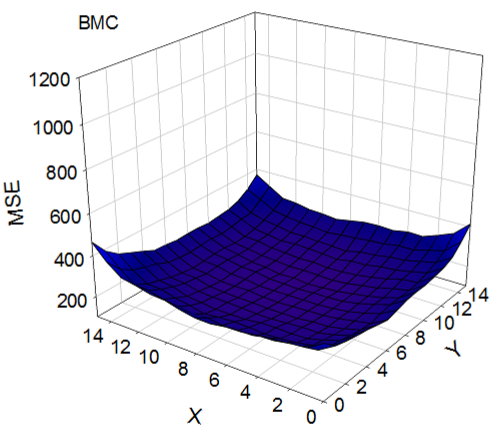
Figure 2.3: Template Matching Prediction.

right quarter. This result is well supported by the empirical data displayed in Fig. 2.4, Fig. 2.5 and Fig. 2.6, where the actual error surface and the ones predicted by the two models are compared. For clarity we have rotated the error surfaces counterclockwise by 135°. From the figure, we also observe that Zheng's model seems to perform better in estimating error variances.

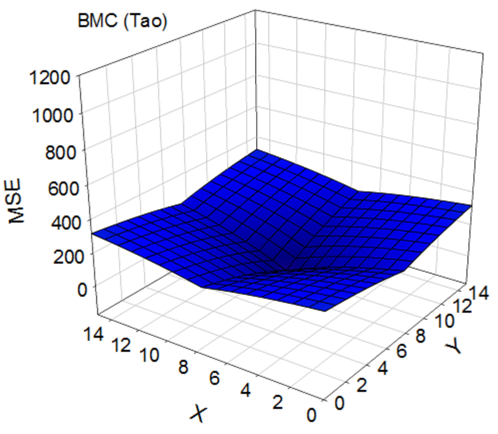
In summary, although TMP does not require extra motion information, its prediction efficiency is generally much worse than that of BMC in the mean-square error (MSE) sense. An exception is when both the intensity and motion fields are less random or have high spatial correlation, that is, with Tao's model, σ_I^2, σ_m^2 are smaller or ρ_m, K tend to be larger and with Zheng's model, ϵ is small. It is then natural to question how it can achieve a bit-rate saving of 10%. The answer becomes clear when its performance is compared with that of SKIP prediction.

2.3.3 Error Variance Distribution of SKIP Prediction

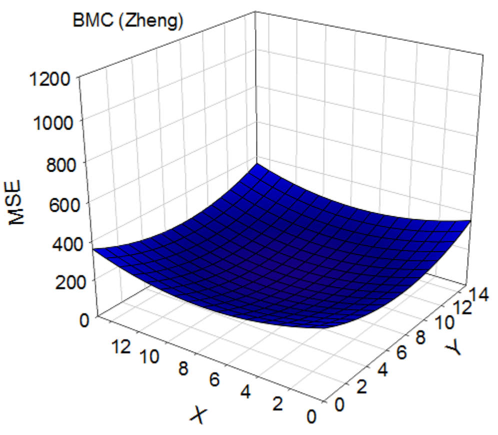
We shall now derive formulae that will enable us to estimate the error variance for SKIP prediction. Recall that if a block is coded in SKIP mode, its motion vector



(a)

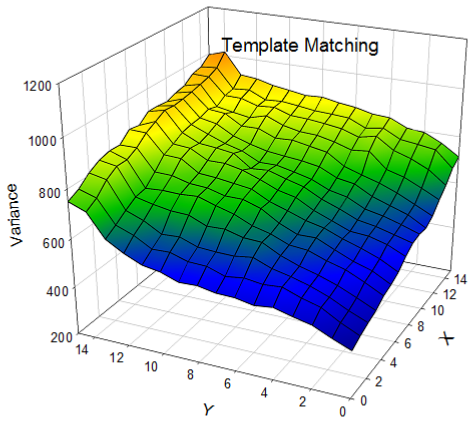


(b)

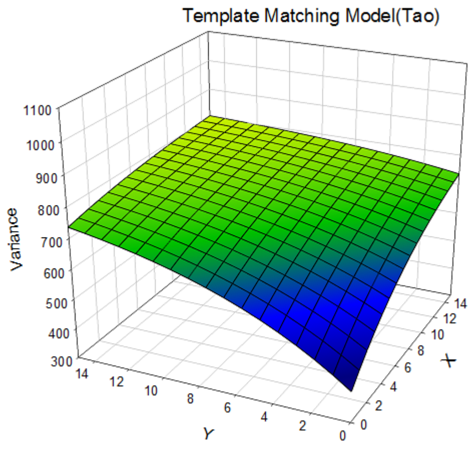


(c)

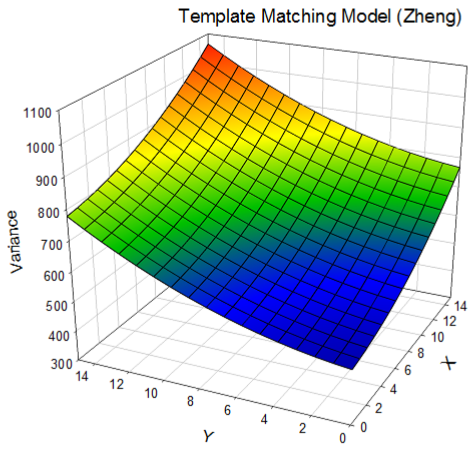
Figure 2.4: Mean-square prediction error surfaces of block $\mathcal{B}$ produced with BMC by (a) empirical results, (b) Tao and (c) Zheng's model, respectively. The sequence is Football and the block size used for motion compensation is 16x16.



(a)

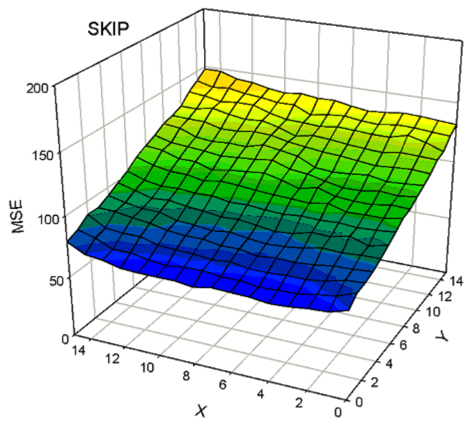


(b)

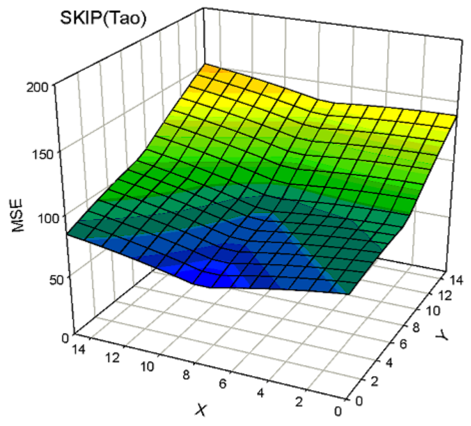


(c)

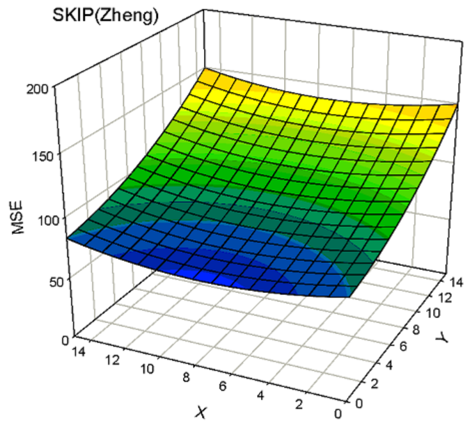
Figure 2.5: Mean-square prediction error surfaces of block $\mathcal{B}$ produced with TMP by (a) empirical results, (b) Tao and (c) Zheng's model, respectively. The sequence is Football and the block size used for motion compensation is 16x16.



(a)



(b)



(c)

Figure 2.6: Mean-square prediction error surfaces of block $\mathcal{B}$ produced with SKIP by (a) empirical results, (b) Tao and (c) Zheng's model, respectively. The sequence is Football and the block size used for motion compensation is 16x16.

is determined by the median of those in its neighborhood. Using the example shown in Fig. 2.3, the inferred vector $\widehat{\mathbf{v}}$ for block $\mathcal{B}$ is

$$\begin{aligned}\widehat{v}_x &= \text{Median}\{v_x(\mathbf{s}_1), v_x(\mathbf{s}_2), v_x(\mathbf{s}_3)\} \\ \widehat{v}_y &= \text{Median}\{v_y(\mathbf{s}_1), v_y(\mathbf{s}_2), v_y(\mathbf{s}_3)\}\end{aligned} \tag{2.9}$$

where $(v_x(\mathbf{s}_i), v_y(\mathbf{s}_i))^T, i = 1, 2, 3$ are the motion vectors associated with blocks $\mathcal{B}_i$ and are approximated by the motion of their centers. The corresponding mean-square prediction error for pixel $\mathbf{s}, \mathbf{s} \in \mathcal{B}$ then becomes

$$\begin{aligned} E\left[d^2(\mathbf{s};\widehat{\mathbf{v}})\right] &= E\left[(I_k(\mathbf{s}) - I_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}))^2\right] \\ &= E\left[(I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s})) - I_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}))^2\right]. \end{aligned} \tag{2.10}$$

Computing the expectation in (2.10), which involves *order statistics,* is in general a difficult task. To circumvent the difficulties, we take a simpler approach by assuming that $\widehat{\mathbf{v}}(i,j) \equiv (\widehat{v}_x, \widehat{v}_y) = (v_x(\mathbf{s}_i), v_y(\mathbf{s}_j)), i, j = 1, 2, 3$, with each ordered pair being equally likely. Hence, we can replace (2.10) with

$$\begin{aligned} &E\left[d^2(\mathbf{s};\widehat{\mathbf{v}})\right] \\ &= \tfrac{1}{9}\sum_{i=1}^{3}\sum_{j=1}^{3} E\left[(I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s})) - I_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(i,j)))^2\right], \end{aligned} \tag{2.11}$$

which can readily be evaluated by incorporating Tao's model. A straightforward calculation then gives

$$E\left[d^2(\mathbf{s};\widehat{\mathbf{v}})\right] = \frac{8\sigma_I^2\sigma_m^2}{3K}\sum_{i=1}^{3}\left(1 - \rho_m^{\|\mathbf{s}-\mathbf{s}_i\|_1}\right). \tag{2.12}$$

Similarly, repeating the procedure in [30], we obtain the result for Zheng's model as

$$E\left[d^2(\mathbf{s};\widehat{\mathbf{v}})\right] \approx \frac{\epsilon}{3}\sum_{i=1}^{3}\|\mathbf{s} - \mathbf{s}_i\|_2^2, \tag{2.13}$$

where the approximation is due to the use of Taylor's expansion in computing the prediction error $I_{k-1}(\mathbf{s} + \mathbf{v}(\mathbf{s})) - I_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(i,j))$.

Table 2.1: Comparison of Mean-Square Prediction Error

	Football QP22			Foreman QP22			Football QP38			Foreman QP38		
Schemes	Emp.	T.	Z.	Emp.	T.	Z.	Emp.	T.	Z.	Emp.	T.	Z.
BMC(8)	112	109	113	19	17	19	141	134	141	43	40	43
TMP L2(8)	372	302	342	41	29	31	398	307	360	70	48	64
TMP L4(8)	382	346	369	39	33	34	405	351	385	70	55	66
BMC(16)	238	232	238	28	27	28	256	246	256	59	55	59
TMP L2(16)	590	530	609	54	48	34	600	516	597	85	67	66
TMP L4(16)	588	555	620	55	50	37	596	539	607	86	70	69
SKIP(16)	913	916	887	129	136	140	913	914	885	329	340	339

It is interesting to know that both (2.12) and (2.13) are merely a weighted sum of the mean-square prediction errors, i.e. $\sum_{i=1}^{3} (E[d^2(\mathbf{s}; \mathbf{v}(\mathbf{s}_i))]/3)$, when $\mathbf{v}(\mathbf{s}_i)$,$i = 1, 2, 3$ are separately utilized for motion compensation of pixel $\mathbf{s}$. In fact, this is a direct consequence of our assumption made about $\widehat{\mathbf{v}}$. Its validity is justified by the empirical data given in Fig. 2.4, 2.5 and 2.6, where it is seen that the error surfaces predicted by (2.12) and (2.13) resemble closely the actual one. Also, as expected, with the help of $\mathbf{v}(\mathbf{s}_2)$ SKIP prediction tends to minimize the error at the upper part of the block, especially at the upper right quarter.

2.3.4 Comparison of BMC, TMP and SKIP Prediction

Table 2.1 compares the MSE of residual signals for different schemes. The empirical values and those predicted by the models are illustrated. For experiments, we use CIF Football and Foreman sequences, each being 50-frame long. The search range for block or template matching is ±32 pixels, with quarter-pel accuracy. To simulate quantization effects, the reference frame and the template region (of size 2 or 4) are coded by H.264/AVC. In addition, the model parameters $\sigma_I^2\sigma_m^2/K, \rho_m$ and ϵ are estimated by a least-square fit to empirical data.

From the table, several observations can be made: (a) the models are consistent with experimental results (at least qualitatively); (b) with explicit motion information, BMC yields a minimum MSE among all the schemes; (c) TMP consistently outperforms SKIP prediction regardless of the template or target block size; and (d) the MSE of TMP increases as the template or target block size is increased. The third explains why the bit rate can be significantly reduced when TMP is applied to SKIP macroblocks as an alternative prediction source [13]. The last is due to the fact that the template centroid deviates more from the center of the target block. Remarkably, these results are true in an average sense, meaning that a hybrid of TMP and BMC may outperform either one alone, as reported in [13].

2.4 Summary

We have re-examed the predictioin efficiency of motion compensated prediction (MCP) and interpreted it as a motion sampler followed by the reconstruction of prediction signal. We also show that, in a statistical sense, block matching based motion estimation will result in motion vectors that are most likely to be the motion vectors sampled at block centers. With the help of motion and intensity models, the comparison of BMC, CGI, OBMC, TMP and SKIP prediction are also demonstrated both theoretically and empirically. Although TMP hardly competes with BMC, TMP is shown to outperform SKIP prediction, which explains why the bit rate can be significantly reduced when TMP is efficiently combined with SKIP prediction. Based on this theoretical framework, in this dissertation, we then apply some of these results to design a parametric solution for OBMC to suit for irregular motion sampling structures.

Chapter 3

Parametric Overlapped Block Motion Compensation

3.1 Introduction

As discussed in chapter 2, various algorithms have been proposed to improve BMC. The most straightforward technique is variable block-size motion compensation(VBSMC), which increases motion sampling density in areas with complex motion to compensate for the inefficiency of BMC. By contrast, Control-Grid Interpolation (CGI) [21] and Overlapped Block Motion Compensation (OBMC) [19] use more sophisticated algorithms to reconstruct the motion field without additional samples. The former improves motion interpolation by employing a triangular filter function, while the latter directly gives a linear estimate of each pixel's intensity based on predictors derived from the current and nearby block MVs. Both are able to alleviate blocking artifacts effectively, but in practice, OBMC is preferred to CGI since the averaging of predictors also helps to reduce quantization noise [25]. To reduce and equalize prediction error within blocks, two approaches have been proposed: overlapped block motion compensation (OBMC) [3] and variable blocksize motion compensation (VBSMC) [4][5]. OBMC improves the motion compensation accuracy for every pixel by considering nearby motion estimates as different plausible hypotheses for its true motion. VBSMC, on the other hand, extends BMC naturally to allow the use of subblocks of varying size in motion compensation. While OBMC requires no extra side information, VBSMC must additionally signal the choice of block size and motion vector. Each method has some merits and faults, and this dissertation seeks to form an optimized hybrid of the two techniques.

Motivated by the preceding observations, we are led to seek an optimized hybrid of VBSMC and OBMC, aiming to trade better prediction for fewer MVs while retaining the flexibility to adapt motion sampling structure according to variations in image statistics. However, determining OBMC weights to associate with MVs on an irregular grid poses a challenging problem. This is because the variable block-size partitioning yields spatially varying geometric relations between a prediction pixel and its nearby block centers. In this case, solving for the weights with the least-squares method would become an under-determined problem since a distinct solution has to be sought for each possible context. Clearly, there may be more parameters to be estimated than there are data points.

This problem is not new. A similar situation occurred in the development of H.263 [1]. At that time, it was resolved by treating larger blocks as a collection of smaller blocks with the same MV in each smaller block as in the larger aggregate block and by applying a fixed window function to all MVs. In an attempt to extend the notion to H.264/AVC, Wang *et al.* [27] additionally proposed to weight more heavily those MVs from smaller aggregate blocks, which they believed can more reliably represent the motion of neighboring blocks, although no justification was given. Both methods suffer from the same problem that inner pixels in larger blocks are not properly compensated. Essentially, the MVs utilized for OBMC of those pixels are replicated from the same (aggregate) block MVs, producing a net effect like BMC. A third method that has recently been proposed is irregular-grid OBMC [11], which circumvents this deficiency by an adaptive window support that scales with local motion sampling density. It, however, remains unclear how to choose a proper scaling factor for each MV.

This dissertation departs from heuristic methods to approach the problem from a theoretical perspective. We formalize the notion of motion-compensated prediction (MCP) as a two-stage process consisting of sparse motion sampling followed by the reconstruction of temporal predictors. Within such a framework, OBMC in its generalized form is seen to find a LMMSE estimate for every pixel's intensity based on motion-compensated signals derived from MVs sampled at nearby block centers. This viewpoint allows us to derive a parametric solution, termed POBMC, for determining the optimal weights in closed form. In doing so, the signal models in [30] are adopted to describe the probabilistic structures of the underlying intensity and motion fields. One important result of our POBMC is that its parameters include only the ℓ^2 distances between the locations of the prediction pixel and the MVs involved–i.e., their geometric relations are all that are needed to determine the weights. This leads to a generic method of reconstructing temporal predictors from any sparsely and irregularly sampled motion data.

Although our approach has some parallels with the other parametric solution [24], the unique features that distinguish this work from it include

1. Our focus is to adapt OBMC to suit variable block-size motion partitioning, while [24] concentrates on adjusting OBMC windows, based on the use of fixed block-size partitioning, in response to variations in sequence statistics;

2. We adopt an alternative signal model [30], which not only better represents the reality but also gives a result that is considerably more intuitive and

tractable;

3. We address the uncertainty associated with a block MV's location by introducing a compensation term to reflect its dispersion around the block center;

4. We propose a suboptimal yet computationally efficient implementation, which need not solve the Wiener-Hopf equation and thus eliminates the need to compute matrix inverse.

In addition, we implement the proposed scheme with KTA 2.4r1 [20] and provide a performance comparison with the recently proposed Enhanced Adaptive Interpolation Filter (EAIF) [29] and Quadtree-based Adaptive Loop Filter (QALF) [12] together with an analysis on how they interact with each other.

In the common test conditions, our POBMC delivers better rate-distortion (R-D) performance than both the H.263 OBMC [1] and the parametric solution [24]. Relative to an H.264/AVC anchor with extended macroblock (MB) size, it achieves 3.1% (0.7-13.6%) BD-rate reductions, compared to 4.6% (0.5-10.1%) and 7.2% (1.3-18.0%) with the single use of EAIF and of QALF, respectively. Although POBMC has the least gain among these filters, it can be combined efficiently with either of the other two filters. The result is an improvement that is almost the sum of their separate effects. In particular, the combination of POBMC and QALF performs very close to or better than that of EAIF and QALF, even in cases where the single use of EAIF outperforms that of POBMC.

The rest of this dissertation is organized as follows: Section II revisits the notion of motion-compensated prediction from a perspective based on motion sampling and reconstruction. Section III presents in detail the derivation of our parametric solutions. Section IV examines their properties by contrasting theoretical predictions with empirical data. Section V evaluates the compression performance of POBMC from various aspects and provides a runtime analysis. Section VI concludes this dissertation with a summary of our observations and a list of future works. Finally, the implementation details of POBMC is elaborated in Appendix.

3.2 Parametric Overlapped Block Motion Compensation (POBMC)

3.2.1 Review of OBMC

This section briefly reviews the basics of OBMC, to aid the understanding of our POBMC. In words, OBMC is to find a LMMSE estimate of a pixel's intensity value $I_k(\mathbf{s})$ based on motion-compensated signals $\{I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))\}_{i=1}^{L}$ derived from its nearby block MVs $\{\mathbf{v}(\mathbf{s}_i)\}_{i=1}^{L}$. From an estimation-theoretic perspective, these MVs are plausible hypotheses for its true motion, and to maximize coding efficiency, their weights $\mathbf{w} = [w_1, w_2, ..., w_L]^T$ are chosen to minimize the mean-square prediction error subject to the unit-gain constraint [19]:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \xi(\mathbf{w}) \text{ s.t. } \sum_{i=1}^{L} w_i = 1, \tag{3.1}$$

where

$$\xi(\mathbf{w}) = E\left\{\left(I_k(\mathbf{s}) - \sum_{i=1}^{L} w_i I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))\right)^2\right\}.$$

Applying the Lagrangian method to (3.1) then gives

$$\mathbf{w}^* = \mathbf{R}^{-1}\left[\mathbf{P} - \mathbf{U}\left(\frac{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{P} - 1}{\mathbf{U}^T\mathbf{R}^{-1}\mathbf{U}}\right)\right], \tag{3.2}$$

where $[\mathbf{R}]_{ij} = E[I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))]$ and $[\mathbf{P}]_j = E[I_k(\mathbf{s})I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))]$ stand, respectively, for auto- and cross-correlation matrices, and $\mathbf{U}$ is a column vector with all elements equal to one [19]. Given that the underlying intensity and motion fields are stationary and that motion samples are taken on a square lattice (such is the case when an image is divided into a group of square blocks for motion search), the optimal weights $\mathbf{w}^*$ for pixel $\mathbf{s}$ depend solely on its relative position within a block. They are often obtained using the least-squares method due to lack of knowledge of the probabilistic models of real data.

The concept of OBMC can be generalized to the case where motion sampling structure is irregular. The challenge, however, becomes how to compute for each pixel its optimal weights to associate with nearby MVs, given that both auto- and cross-correlation functions are spatially varying. The least-squares solution, although feasible in theory, is impractical because the storage of weighting coefficients optimized for different contexts demands huge memory requirements. To tackle this problem, we resort to a parametric solution.

3.2.2 Signal Models

POBMC aims to give a closed-form formula for the optimal weights. To do so, it usually needs to assume signal models for the intensity and motion fields. The choice of the models often involves a trade-off between accuracy, simplicity and tractability, and can sometimes be quite subtle. For instance, Tao *et al.* [24] model the auto-correlation functions of the intensity and motion fields using quadratic and exponential functions, respectively. These models are so chosen that $\mathbf{R}$ and $\mathbf{P}$ can be expressed in closed form. In general, different models have their merits and faults, and what model best represents reality is normally justified by empirical simulations.

In this dissertation we aim to give a direct estimate of the optimal weights $\mathbf{w}^*$. This is accomplished by adopting the motion model proposed in [30], which assumes that the difference between the true motion of any two pixels, e.g., $\mathbf{s}_1$ and $\mathbf{s}_2$, has a normal distribution of the form

$$v_x(\mathbf{s}_1) - v_x(\mathbf{s}_2) \text{ or } v_y(\mathbf{s}_1) - v_y(\mathbf{s}_2) \sim \mathcal{N}(0, \alpha r^2(\mathbf{s}_1, \mathbf{s}_2)), \tag{3.3}$$

where α is a positive number indicating the degree of motion randomness in the horizontal or vertical direction [1], and $r(\mathbf{s}_1, \mathbf{s}_2)$ is the ℓ^2 distance (measured in the unit of pixel) between $\mathbf{s}_1$ and $\mathbf{s}_2$. Caution, however, must be exercised when using (3.3) because it is an incomplete specification. The variance $\alpha r^2(\mathbf{s}_1, mathbfs_2)$ must be bounded from above for the model to be proper. To see this, let us assume the motion field is stationary and symmetric. It then follows from (3.3) that

$$E\{v_x(\mathbf{s}_1)v_x(\mathbf{s}_2)\} = E\{v_y(\mathbf{s}_1)v_y(\mathbf{s}_2)\} = \sigma_m^2 + \mu_m^2 - \frac{\alpha r^2(\mathbf{s}_1, \mathbf{s}_2)}{2}, \tag{3.4}$$

where μ_m and σ_m^2 are the mean and the variance of the motion field, respectively. Using the Cauchy-Schwarz inequality, we have $4\sigma_m^2 \geq \alpha r^2(\mathbf{s}_1, \mathbf{s}_2) \geq 0$. The lower bound is obvious, but the upper bound deserves more attention. According to (3.4), it implies that the MVs of two far-away pixels are negatively correlated. A tighter bound that agrees more with the general observation is $2\sigma_m^2$, which will make them become uncorrelated. We can equivalently define a clipper function for $r(\mathbf{s}_1, \mathbf{s}_2)$ to have the property $\widehat{r}(\mathbf{s}_1, \mathbf{s}_2) = Clip(0, r(\mathbf{s}_1, \mathbf{s}_2), \tau)$, where the clipping threshold $\tau = \sqrt{2\sigma_m^2/\alpha}$. Hereafter we shall omit the tedious repetition of this constraint by using $\widehat{r}(\mathbf{s}_1, \mathbf{s}_2)$ in place of $r(\mathbf{s}_1, \mathbf{s}_2)$.

3.2.3 Optimal Weights in Parametric Form

With the signal model in (3.3), we next proceed to determine the optimal weights $\mathbf{w}^*$ using *calculus*. To begin with, we rewrite, by noting that $\sum_{i=1}^{L} w_i = 1$, the

[1]The smaller α value suggests the motion field has higher spatial correlation.

mean-square prediction error $\xi(\mathbf{w})$ in Eq. (3.1) as

$$\xi(\mathbf{w}) = E\left\{\left(\sum_{i=1}^{L} w_i d(\mathbf{s};\mathbf{v}(\mathbf{s}_i))\right)^2\right\}, \tag{3.5}$$

where $d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)) = I_k(\mathbf{s}) - I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$ denotes the residual signal when $I_k(\mathbf{s})$ is predicted from the motion-compensated signal $I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$ using the MV $\mathbf{v}(\mathbf{s}_i)$ for block i. (3.5) can be written more compactly in matrix notation as

$$\xi(\mathbf{w}) = \mathbf{w}^T E\{\mathbf{d}\mathbf{d}^T\}\mathbf{w} = \mathbf{w}^T\mathbf{D}\mathbf{w}, \tag{3.6}$$

where $\mathbf{d} = [d(\mathbf{s};\mathbf{v}(\mathbf{s}_1)), d(\mathbf{s};\mathbf{v}(\mathbf{s}_2)), ..., d(\mathbf{s};\mathbf{v}(\mathbf{s}_L))]^T$.

To continue, we borrow a result in [30], which shows that if (3.3) is valid, then $E\{d^2(\mathbf{s};\mathbf{v}(\mathbf{s}_i)\}$ has a closed-form formula given by

$$\begin{aligned} E\{d^2(\mathbf{s};\mathbf{v}(\mathbf{s}_i)\} &= E\{(I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s})) - I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i)))^2\} \\ &= \epsilon\widehat{r}^2(\mathbf{s},\mathbf{s}_i), \end{aligned} \tag{3.7}$$

where ϵ is a constant indicating the joint randomness of the motion and intensity fields; $I_k(\mathbf{s}) = I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}))$ with $\mathbf{v}(\mathbf{s})$ denoting the true motion of pixel $\mathbf{s}$; and the block MV $\mathbf{v}(\mathbf{s}_i)$ is approximated as the motion associated with the block center $\mathbf{s}_i$. What remain to be determined in $\mathbf{D}$ are those off-diagonal entries, i.e., $E\{d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)d(\mathbf{s};\mathbf{v}(\mathbf{s}_j)\}, i \neq j$; in fact, their derivations are merely an application of (3.7). With a little bit of algebra [2], we obtain

$$\begin{aligned} &E\{d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)d(\mathbf{s};\mathbf{v}(\mathbf{s}_j)\} \\ =&E\{(I_k(\mathbf{s})-I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i)))(I_k(\mathbf{s})-I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j)))\} \\ =&\frac{1}{2}E\{(I_k(\mathbf{s})-I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))^2\} \\ +&\frac{1}{2}E\{(I_k(\mathbf{s})-I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))^2\} \\ -&\frac{1}{2}E\{(I_k(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))-I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))^2\} \\ =&\frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_i)+\widehat{r}^2(\mathbf{s},\mathbf{s}_j)-\widehat{r}^2(\mathbf{s}_i,\mathbf{s}_j)\right) \end{aligned} \tag{3.8}$$

[2] $(a-b)(a-c) = \frac{1}{2}(a-b)^2 + \frac{1}{2}(a-c)^2 - \frac{1}{2}(b-c)^2$

The astute reader may feel a sense of misgiving about the approximation $E\{(I_k(\mathbf{s}+\mathbf{v}(\mathbf{s}_i)) - I_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_j))^2\} \approx \epsilon\widehat{r}^2(\mathbf{s}_i, \mathbf{s}_j)$, as it does not seem to be a direct extension of (3.7). The subtle difference is the replacement of $\mathbf{v}(\mathbf{s})$ with $\mathbf{v}(\mathbf{s}_i)$. However, assuming that $\mathbf{v}(\mathbf{s}_i)$ represents the true motion of the block center $\mathbf{s}_i$, its proof can be carried out in the same manner as for (3.7). Another testament to its mathematical correctness is that (3.8) includes (3.7) as a special case where $\mathbf{s}_i = \mathbf{s}_j$.

Returning to (3.6), we are now ready to find the optimal weights. Since $\xi(\mathbf{w})$ is to be minimized subject to $\sum_{i=1}^{L} w_i = 1$, the solution space has only a dimension of $L-1$. To simplify the computation, we define a reduced-dimension weight vector $\widetilde{\mathbf{w}} = [\widetilde{w}_1, \widetilde{w}_2, ..., \widetilde{w}_{L-1}]^T$, the elements of which are free variables and are related to the weight vector $\mathbf{w}$ by

$$\mathbf{w} = \mathbf{e} - \mathbf{M}\widetilde{\mathbf{w}}, \tag{3.9}$$

where

$$\mathbf{e} = [0, 0, ..., 1]^T_{L\times 1} \text{ and}$$

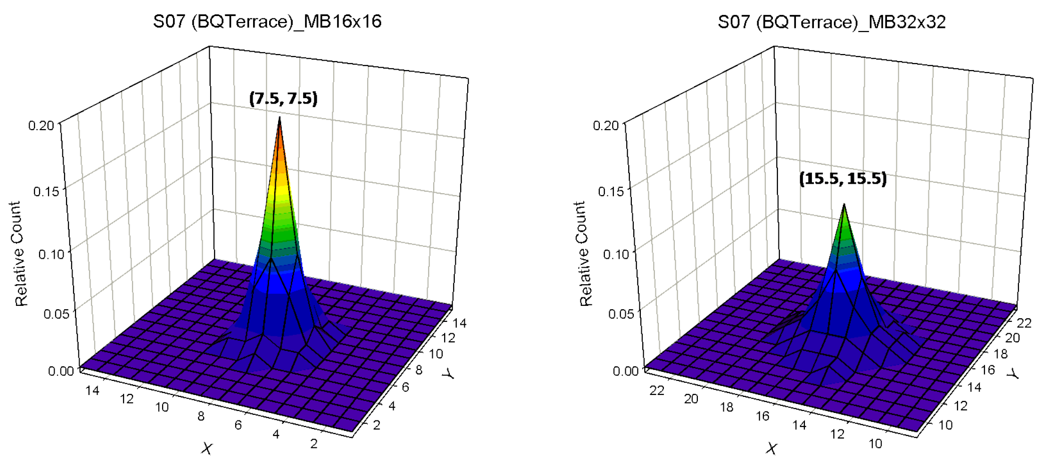
$$\mathbf{M} = \begin{bmatrix} -\mathbf{I} \\ \mathbf{U}^T \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -1 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}_{L\times(L-1)}.$$

When spelled out, (3.9) simply states that $w_i = \widetilde{w}_i$, $1 \le i \le L-1$ and $w_L = 1 - \sum_{i=1}^{L-1} \widetilde{w}_i$. Substituting (3.9) into (3.6), setting the gradient of ξ with respect to $\widetilde{\mathbf{w}}$ to $\mathbf{0}$, and solving the resulting system equations then yields

$$\widetilde{\mathbf{w}}^* = (\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{M})^{-1}\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{e}. \tag{3.10}$$

The result of $\widetilde{\mathbf{w}}^*$ immediately gives that of $\mathbf{w}^*$ by (3.9):

$$\mathbf{w}^* = \left(\mathbf{I} - \mathbf{M}(\mathbf{M}^\mathbf{T}\mathbf{D}\mathbf{M})^{-1}\mathbf{M}^\mathbf{T}\mathbf{D}\right)\mathbf{e} \tag{3.11}$$



(a) (b)

Figure 3.1: The distribution of a block MV's location when the block size used for motion search is varied: (a) 16x16 and (b) 32x32. The MV location is approximated by the centroid position of the first ten pixels, in a block, having relatively smaller prediction error.

Inspection of (3.11) reveals that the optimal weights depend solely on the distances between the prediction pixel $\mathbf{s}$ and the block centers involved $\{\mathbf{s}_i\}_{i=1}^{L}$. The ϵ term is absent in the final result. This remarkable property allows MVs sampled on a possibly irregular grid to be incorporated for OBMC, providing a reconstruction method applicable to any sampling structures.

3.2.4 Optimal Weights in a Special Case

An interesting special case occurs by considering $\mathbf{D}$ as a diagonal matrix. In this case, the prediction errors $\{d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)\}_{i=1}^{L}$ are uncorrelated with each other, i.e., $E\{d(\mathbf{s};\mathbf{v}(\mathbf{s}_i)d(\mathbf{s};\mathbf{v}(\mathbf{s}_j)\} = 0, \forall i \neq j$, and $\mathbf{w}^*$ becomes

$$\mathbf{w}^* = \left(\sum_{i=1}^{L} \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_i)}\right)^{-1} \left[\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)}, \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)}, \dots, \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_L)}\right]^T. \tag{3.12}$$

The proof of this result requires some work but involves only straightforward computations. (3.12) is a great simplification of (3.11): the optimal weights w_i^* are simply the normalized inverses of the corresponding squared distances between $\mathbf{s}$ and $\mathbf{s}_i$. It has the interpretation that prior to normalization, the contribution of each MV $\mathbf{v}(\mathbf{s}_i)$ to estimating its nearby pixel intensities is a function of pixel $\mathbf{s}$ that decays quadratically with $\widehat{r}(\mathbf{s}, \mathbf{s}_i)$. If we take such a view, other functions can be substituted for $1/\widehat{r}^2(\mathbf{s}, \mathbf{s}_i)$. For example, it may be just as well to adopt the raised cosine or bilinear function of various supports, or to change the power of $1/\widehat{r}(\mathbf{s}, \mathbf{s}_i)$. As an afterthought, each of these functions may correspond to making some specific assumptions about the motion and intensity fields. Due to its simplicity, (3.12) will be included in the following sections as an alternative to (3.11).

3.2.5 MV Location Uncertainty

In the preceding derivation, we have always assumed that a block MV represents the true motion of the block center. However, this is an approximation; in fact, it may correspond to the motion of any pixel around the center. To see this, consider a small group of pixel locations in a block where prediction errors are relatively smaller. We think of the block MV as the motion connected to their centroid. Although not precise, this expedient provides a rough estimate of the MV location without having to acquire the true motion field. Fig. 3.1 presents two plots showing the centroid distributions when the block size used for motion search is varied. Two observations are immediate: (a) the means of both distributions are close to the block center, which justifies the widely accepted approximation, and (b) the variance is non-zero and increases with the increasing block size, which suggests that the locations of $\mathbf{s}_i, \mathbf{s}_j$ in (3.7) and (3.8) should be modeled probabilistically.

We now generalize both equations to consider their random effects. To conform with our previous notation, we denote by $\widetilde{\mathbf{s}}_i = \mathbf{s}_i + \mathbf{n}_i$ (respectively, $\widetilde{\mathbf{s}}_j = \mathbf{s}_j + \mathbf{n}_j$) their true locations, which are characterized by an independent, additive noise vector $\mathbf{n}_i$(respectively, $\mathbf{n}_j$) with mean zero and covariance matrix

$$\mathbf{K}_{\mathbf{n}_i\mathbf{n}_i} = \begin{bmatrix} \delta_i^{(x)} & \rho_i\sqrt{\delta_i^{(x)}\delta_i^{(y)}} \\ \rho_i\sqrt{\delta_i^{(x)}\delta_i^{(y)}} & \delta_i^{(y)} \end{bmatrix}.$$

Substituting $\widetilde{\mathbf{s}}_i$ for $\mathbf{s}_i$ in (3.7) and applying the law of iterated expectations, we get

$$\begin{aligned} & E\{d^2(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i)\} \\ =& E\left\{E\{d^2(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i))|\widetilde{\mathbf{s}}_i\}\right\} = \epsilon E\left\{\widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_i)\right\} \\ \simeq& \epsilon E\left\{(\mathbf{s}^{(x)}-\mathbf{s}_i^{(x)} - \mathbf{n}_i^{(x)})^2 + (\mathbf{s}^{(y)}-\mathbf{s}_i^{(y)} - \mathbf{n}_i^{(y)})^2\right\} \\ \simeq& \epsilon\widehat{r}^2(\mathbf{s},\mathbf{s}_i) + \epsilon(\delta_i^{(x)} + \delta_i^{(y)}), \end{aligned} \tag{3.13}$$

where the superscripts x, y indicate the two components of a point or a vector. In (3.13), the locations of pixel $\mathbf{s}$ and the block center $\mathbf{s}_i$ are treated as known variables because we know exactly what MVs will be utilized for the motion compensation of pixel $\mathbf{s}$. As such, they are deterministic quantities and the expectation in the penultimate approximation is taken with respect to $\mathbf{n}_i$ only. In the course, we have tacitly ignored the clipping effect on $r(\mathbf{s},\widetilde{\mathbf{s}}_i)$,which however is crucial for our signal models to be proper (Section 3.2.2). A way out of this difficulty is to assume that $\mathbf{s}_i$ is close enough to $\mathbf{s}$ so that the result in (3.13) is a good approximation. This assumption can be justified to some extent since in practical implementation of our schemes, we use only those neighboring MVs that are closer to a pixel for its motion compensation. From (3.13), the consequence of MV location uncertainty is an increase in the mean-square prediction error. Of particular interest is that the penalty depends only on the variances of $\mathbf{n}_i$ (or equivalently, the variances of $\widetilde{\mathbf{s}}_i$) regardless of its distribution.

A similar calculation leads us to

$$
\begin{aligned}
&E\{d(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_i)d(\mathbf{s};\mathbf{v}(\widetilde{\mathbf{s}}_j)\}\\
=&\frac{1}{2}\epsilon E\left\{\left(\widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_i)+\widehat{r}^2(\mathbf{s},\widetilde{\mathbf{s}}_j)-\widehat{r}^2(\widetilde{\mathbf{s}}_i,\widetilde{\mathbf{s}}_j)\right)\right\}\\
\simeq&\frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_i)+\delta_i^{(x)}+\delta_i^{(y)}\right)+\frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_j)+\delta_j^{(x)}+\delta_j^{(y)}\right)\\
-&\frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s}_i,\mathbf{s}_j)+\delta_i^{(x)}+\delta_i^{(y)}+\delta_j^{(x)}+\delta_j^{(y)}\right)\\
=&\frac{1}{2}\epsilon\left(\widehat{r}^2(\mathbf{s},\mathbf{s}_i)+\widehat{r}^2(\mathbf{s},\mathbf{s}_j)-\widehat{r}^2(\mathbf{s}_i,\mathbf{s}_j)\right),
\end{aligned}
$$

where $\mathbf{n}_i$ and $\mathbf{n}_j$ are assumed to be independent. As shown, the variance terms in (3.14) cancel each other out, leading to the same result as in (3.8). Simply substituting (3.13) into the matrix $\mathbf{D}$ in (3.11) gives the *modified optimal weights* with consideration of MV location uncertainty. These results also apply to the case where $\mathbf{D}$ is a diagonal matrix.

In concluding this section, we want to point out that the proposed scheme has two parameters to be determined: the clipping threshold τ and the degree of MV location uncertainty $\delta = \delta_i^{(x)} + \delta_i^{(y)}$. The latter actually denotes a set of parameters, one for each distinct block size. As will be discussed later, they can be determined by off-line training.

3.3 Analysis of Window Functions

While (3.11) characterizes the contributions of a set of MVs to estimating the intensity of a pixel, an equivalent yet more insightful perspective is to see the window function of each MV, which specifies its weights used to estimate pixel intensities in a neighborhood [19]. In this section, we shall gain further insights into the proposed solutions from this viewpoint. To ease comprehension, we first consider the simpler case of fixed block-size motion partitioning, followed by the more sophisticated one involving variable block-size partitioning.

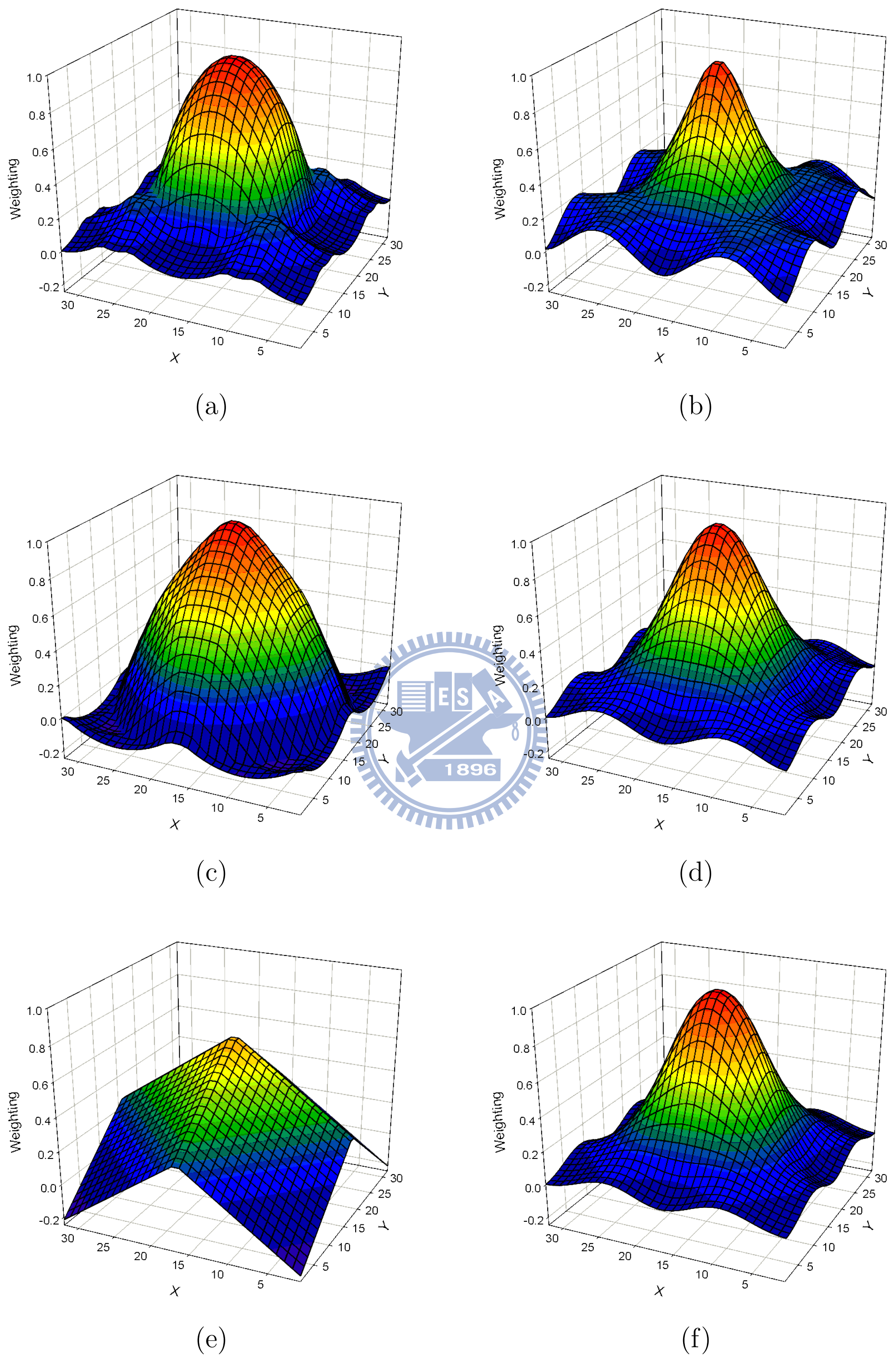


Figure 3.2: The effect of the clipping threshold value on the shape of the proposed parametric windows with (a)(c)(e) non-diagonal and (b)(d)(f) diagonal D matrices. From top to bottom, the clipping threshold values are 10, 15, 35, respectively.

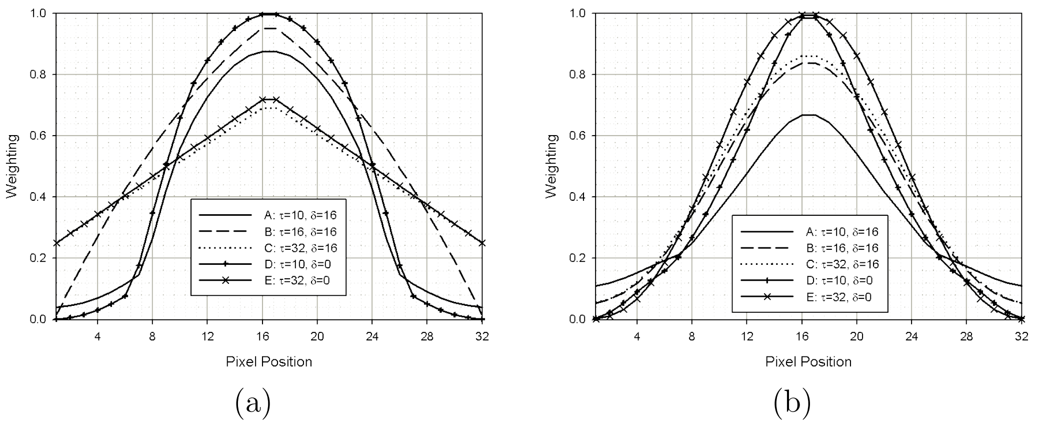


Figure 3.3: Window functions along the slide of Y=16 based on a (a) non-diagonal or (b) diagonal D matrix.

3.3.1 Theoretical Window Functions

Fig. 3.2(a), (b) and (c) plot the window functions for various clipping threshold values τ's. Their counterparts in Fig. 3.2(d), (e) and (f) show the results when the off-diagonal entries of **D** are set zero. In the former case, we observe that the window shape inflates with the increasing τ, and eventually converges to a bilinear function. This trend of inflation continues in the latter case although the change in the window shape is not that radical, especially when the value of τ becomes high enough. These phenomena can be explained by noting that a higher clipping threshold implies a stronger correlation between the motion of different pixels (smaller α) or a larger motion variance (larger σ_m^2). Under these circumstances, it is intuitive to expect that the influence of a block MV will extend to more pixels.

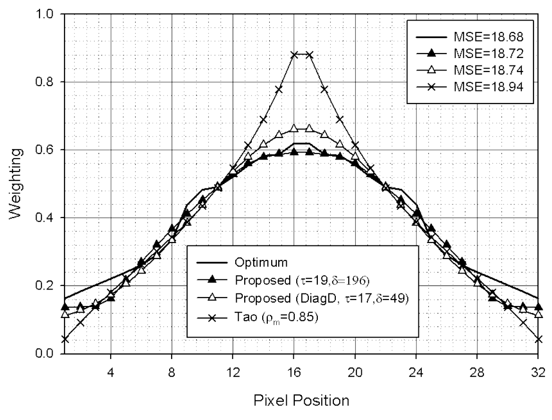
To gain a better appreciation of how the window shape evolves, Fig. 3.3 further displays the cross sections of these windows along the slide $Y = 16$. There are several points to be noted here. First, the weights around the block center ($X = 16$) are seen to be smaller than 1. This result is a manifestation of MV location uncertainty. As expected, their values tend to approach 1 if we have $\delta = \delta_i^{(x)} + \delta_i^{(y)} = 0$ (cf. (3.13)). Some other interesting observations follow from comparing the window values at $X = 16.5$ (current block center) and at $X = 0.5$ or 32.5 (neighboring block centers). The windows with a diagonal $\mathbf{D}$ resemble normal functions in shape, and exhibit an upward trend in magnitude near the block center (respectively, a downward trend at the neighboring block centers) as the value of τ increases. In the general case, however, the behavior is more intricate: the peak value escalates first and then declines. But, both cases have one thing in common–their windows converge to a function dependent only on δ when τ is large enough.

3.3.2 Comparison with Empirical Window Functions

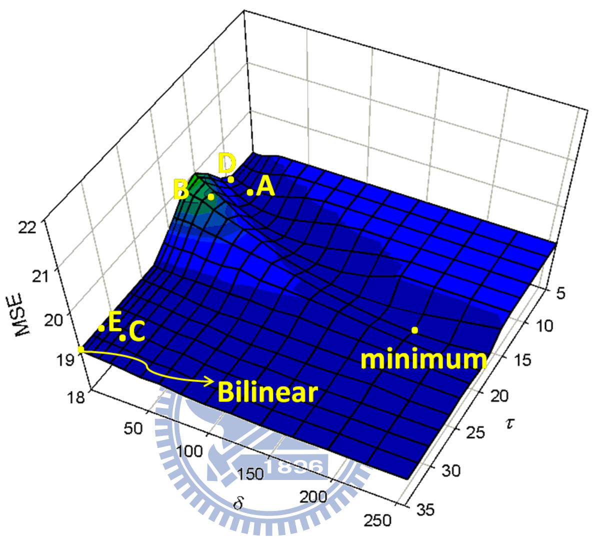
The different results above lead us to wonder which model is more reasonable and how much the penalty is for keeping only the diagonal entries of $\mathbf{D}$. In this section, we provide empirical justifications by contrasting the parametric windows with those obtained by the least-squares method. Results of [24] are also included for comparison. Particularly, to demonstrate the best achievable performance of our parametric schemes, both the values of τ and δ are searched exhaustively based on minimizing the mean-square prediction error, and so is the parameter ρ_m in [24][3].

From the results presented in Fig. 3.4(a) and Fig. 3.5(a), we see that the proposed windows with a non-diagonal $\mathbf{D}$ match closely the least-squares ones. The other windows, although showing similar magnitudes at block boundaries ($X = 8.5$ or 24.5), have much higher weights near the block center ($X = 16.5$). Despite their distinct appearances, the penalties in MSE are somehow surprisingly not as high as expected. We find that there are actually several window functions

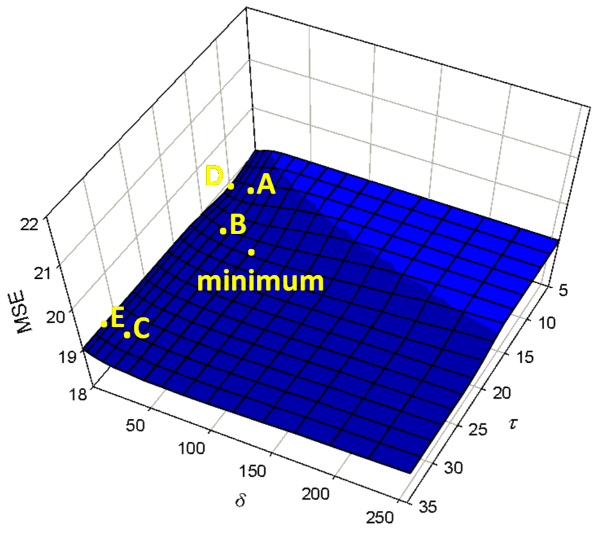
[3]The parametric solution in [24] originally has four parameters to be determined. But, a little neat algebra shows that the resulting window is dependent on only the correlation coefficient of the motion field, ρ_m.



(a)

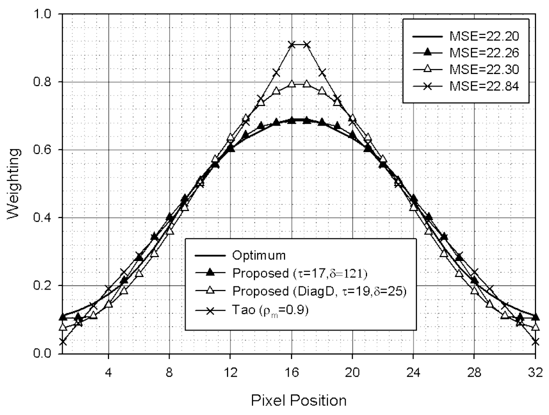


(b)

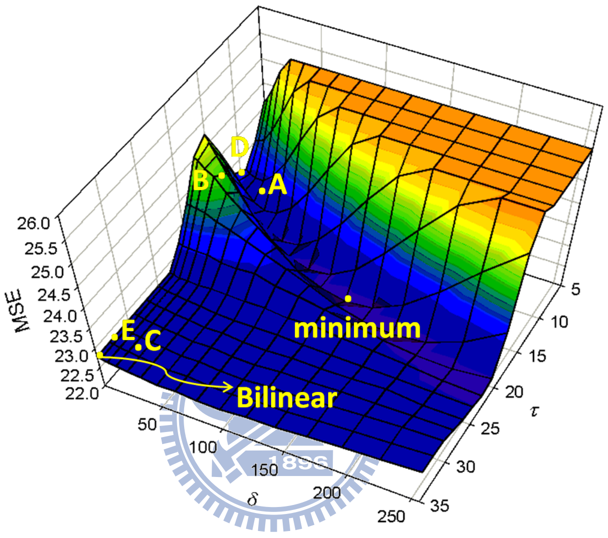


(c)

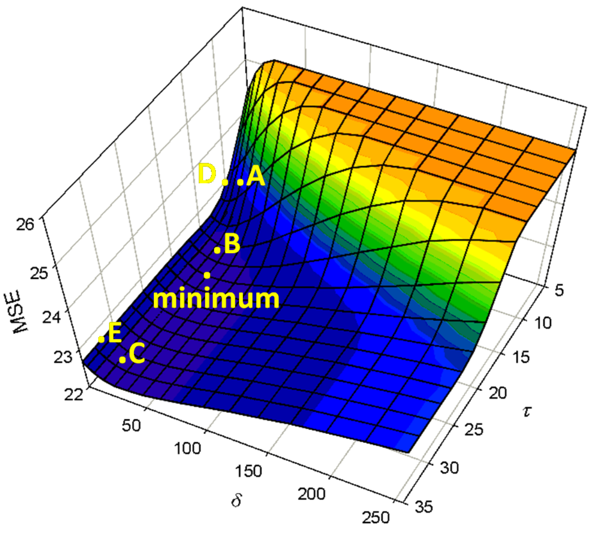
Figure 3.4: Comparisons of window functions and their MSE surfaces using testing sequence "S04": (a) parametric windows versus optimal least-squares windows; the MSE surfaces of the proposed parametric solution with a (b) non-diagonal or (c) diagonal D matrix.



(a)



(b)



(c)

Figure 3.5: Comparisons of window functions and their MSE surfaces using testing sequence "S03": (a) parametric windows versus optimal least-squares windows; the MSE surfaces of the proposed parametric solution with a (b) non-diagonal or (c) diagonal D matrix.

(of different shapes) performing almost equally well. To see this, Fig. 3.4(b) and Fig. 3.5(b) plot the resulting MSEs as functions of τ and δ for the case of non-diagonal $\mathbf{D}$. The best settings, which yield similar MSEs as achieved by the optimal windows, are labelled "minimum". From the figure, these settings roughly have $\tau = 15 \sim 20$ and $\delta = 100 \sim 200$; however, many other choices deliver similar performance. For example, all the settings in Fig. 3.3, except setting B, are among the best performing ones even though their window shapes differ noticeably. In particular, it seems beneficial (in terms of MSE) to have a high clipping threshold. This may be attributed to the incorporation of an R-D based motion search criterion, which imposes a motion smoothness constraint and thus increases the motion correlation. Notice that the MSE stops decreasing when τ exceeds a certain value because, as mentioned previously, the window converges to a function dependent on δ. From Fig. 3.4(b) and Fig. 3.5(b), one may doubt the necessity of using parametric windows, since the bilinear function shown in Fig. 3.2(c) seems to provide sufficiently good performance. While this is indeed the case for fixed block-size motion partitioning, it is not true for the variable block-size case. As will be seen in the next section, difficulties arise when a fixed window is to be applied to MVs sampled on an irregular grid. In addition, it is worth noting that this bilinear function differs from the usual one [19] in that its peak value at the block center is much smaller than 1, which has to do with the MV location uncertainty. A side experiment shows that the latter yields larger MSEs due to the ignorance of this uncertainty. Finally, some sequences are seen to be more sensitive to parameter selection than others. Fig. 3.4(c) and Fig. 3.5(c) further show the results for the case of diagonal $\mathbf{D}$. Comparing with Fig. 3.4(b) and Fig. 3.5(b), there is an obvious distinction in the region with $\tau \geq 15$, where

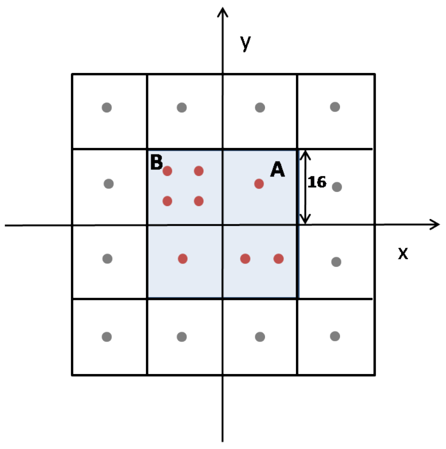


Figure 3.6: An irregular motion sampling grid due to the use of variable block-size motion partitioning.

the MSE first decreases slightly along the δ-axis and then increases rapidly (in some cases) with the increasing δ. The "minimum" points thus have $\delta = 10 \sim 50$. Analogous to the general case, there are also more than one window function showing similar performance to the least-squares solutions; examples are settings B and C in Fig. 3.3. Interestingly, what are improper settings in the general case now become proper ones. This is because the meanings of both parameters change when we twist $\mathbf{D}$. It is for the same reason that the estimation of their values becomes difficult. Fortunately, their optimal values are found empirically to be less sequence-dependent, even if not truly independent, and can be obtained off-line. According to the MSE results in Fig. 3.4(a) and Fig. 3.5(a), ignoring the off-diagonal entries of $\mathbf{D}$ does not seem to have a significant impact on prediction performance, if δ and τ are chosen properly. Because of its simplicity and fairly good performance, we shall hereafter adopt (3.12) as our parametric solution.

3.3.3 Window Functions on Irregular Sampling Grids

The discussion so far has been restricted to window functions for regular motion sampling structures. We now turn our attention to irregular ones, which could

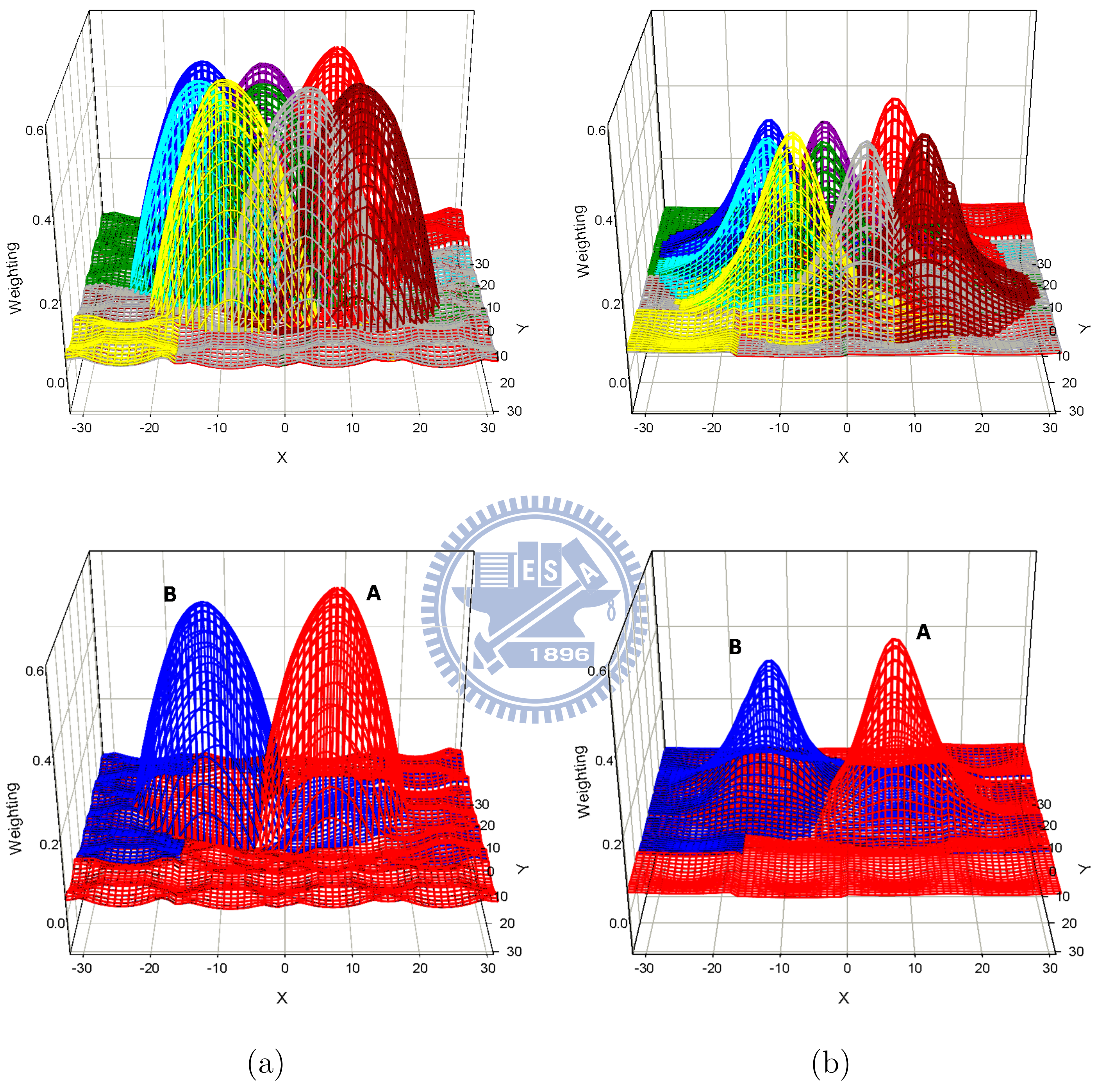
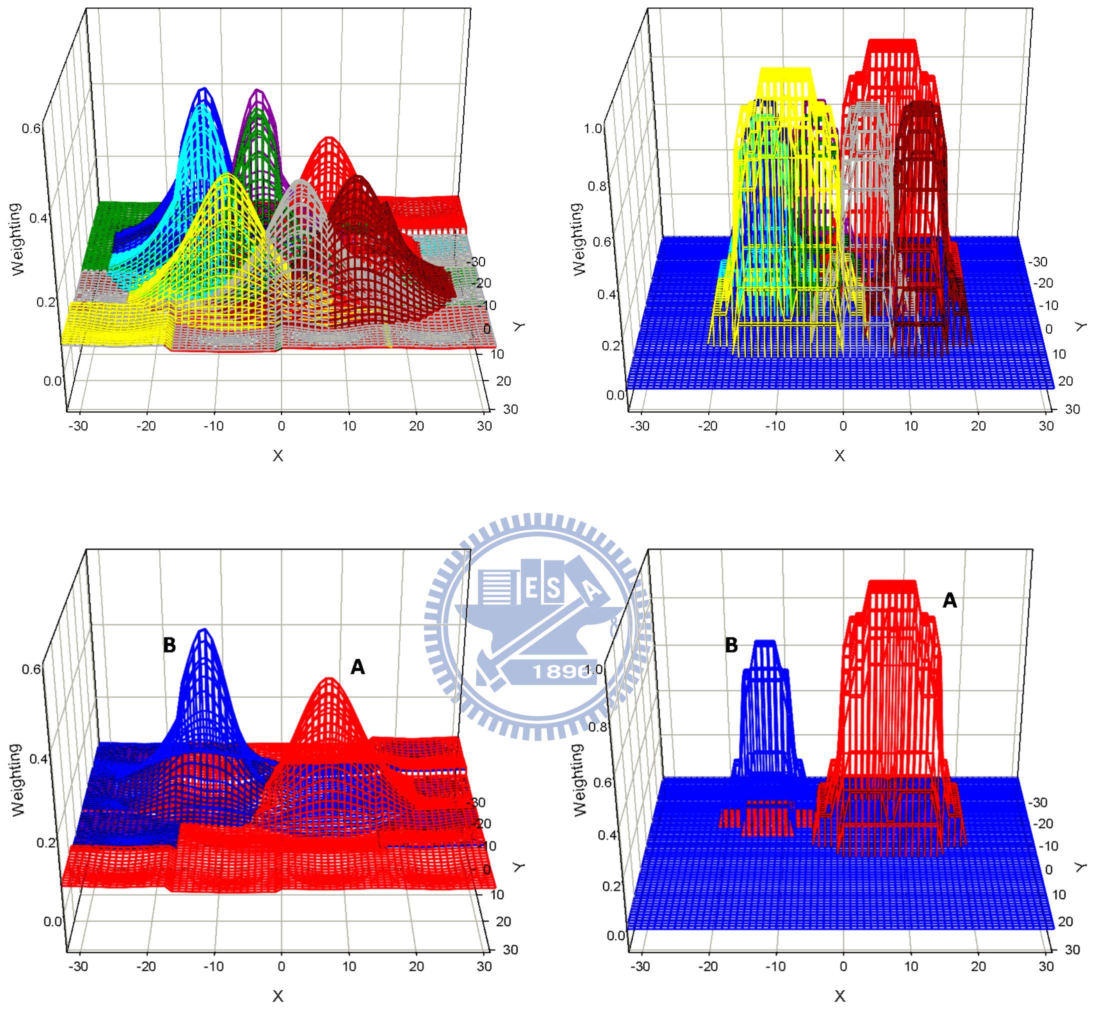


Figure 3.7: Window functions overlaid on the irregular motion sampling grid shown in Fig. 3.6: the proposed parametric windows with (a) a non-diagonal D matrix (Clip=17, δ=121), (b) a diagonal D matrix (Clip=19, δ=25).



(a) (b)

Figure 3.8: Window functions overlaid on the irregular motion sampling grid shown in Fig. 3.6: the proposed parametric windows with (a) a diagonal D matrix plus a MB-adaptive adjustment of δ (Clip=19, δ=16 for 8x8 MVs and δ=36 otherwise), and (b) the H.263 windows.

result from an application of variable block-size motion partitioning. One example of such a structure is given in Fig. 3.6.

Fig. 3.7(a) and (b) plot the window functions associated with those MVs in the shaded area for the cases of non-diagonal and diagonal **D**, respectively. Some of them are displayed separately for close scrutiny. We see that these functions may not be symmetric, and their shapes depend highly on the local sampling pattern. An interesting observation is that windows in areas with a higher motion sampling density (i.e., in MBs with more partitions) are less concentrated; this implies a stronger averaging of the relevant MVs. In theory, it seems reasonable as these MVs, spatially closer to each other, are assumed to be highly correlated, but in practice, this may not always be the case. Sometimes a MB is segmented into smaller partitions because it spans across multiple objects with different motion. We thus propose to adjust the value of δ in a MB-adaptive manner. Recall that it indicates the dispersion of a MV's location around the block center; generally, MVs for smaller blocks should have a smaller δ value. After such adjustment, the results, as illustrated in Fig. 3.8(a), are more centralized windows in high density areas, which help to mitigate the over-blurring of block boundaries. In particular, the tuning of δ need not be fine-granular; an adaptive selection between two distinct levels is sufficient (Section 3.4).

Finally, the results based on the window function in H.263 are shown in Fig. 3.8(b). Recall that in H.263, a fixed window function is used for OBMC. To apply the same window to all MVs, the issue of variable block-size motion partitioning is resolved by treating larger blocks as a collection of smaller blocks with the same MV in each smaller block as in the larger aggregate block. Hence, in computing the effective weighting factor of a MV at some specific pixel, we add up the contributions of all its replicas. From the figure, the window functions for larger blocks have a value equal or close to 1 around the block centers, which implies their inner pixels are not properly compensated by OBMC. This result is a direct consequence of the MV replication mechanism. As will be seen next, this approach leads to poor compression performance; in fact, applying any fixed window with the same approach will suffer from similar problem.

3.4 Experimental Results

The proposed POBMC is analyzed by a series of tests: the first test compares its compression performance with that of similar previous works, including the OBMC in H.263 [1] and the parametric solution in [24]; the second test contrasts POBMC with the two in-loop filters, EAIF [29] and QALF [12], in KTA [20]; and the third test studies how these in-loop filters interact with each other in a complete codec. Finally, we conclude this section with a software runtime analysis, to offer a rough indication of its complexity characteristics. All OBMC schemes are implemented in KTA 2.4r1 [20], with details given in Appendix. All tests, unless otherwise stated, use the configurations shown in Table 3.1. The results are obtained by encoding the first 100 frames of standard JCT-VC test sequences [2]. Those for POBMC are produced using (3.12) with $\tau = 32$.

3.4.1 R-D Performance Analyses

3.4.1.1 Comparison of OBMC Algorithms

Table 3.2 compares the compression performance of different OBMC schemes by showing their BD-rate savings [4] relative to an anchor encoding, which conforms to H.264/AVC High Profile with inclusion of extended MB size (up to 32x32). In particular, they all involve an adaptive switching between BMC and one or more

Table 3.1: Encoder Configurations

Configuration	Setting		
GOP Structure	IPPP...	IBBB...	Hierarchical B
Intra Period	0	0	24
QPP & QPB	QPI+1	QPI+1	QPI+1/2/3/4
QPI	22, 27, 32, 37		
CABAC	On		
Reference Frames	4 (P), 2 (B_L0, B_L1)		
8x8 Transform	On		
De-blocking	On		
RDO	On		
Motion Vector Range	± 128		
Motion Search	3 (EPZS)		
Sub Pixel MC	On		
Adaptive Rounding	Off		
Motion Comp. Block Sizes	8x8 to 32x32		

OBMC options. The subscript H-I indicates the use of only one OBMC option. For example, POBMC$_{\text{H-I}}$ denotes the hybrid of BMC and POBMC with $\delta = 16$, and POBMC$_{\text{H-II}}$ includes one more option with $\delta = 0$. To signal the choice of MCP schemes, a flag is sent for each non-skip super MB (a MB of size 32x32). If a super MB is split into partitions smaller than or equal to 16x16, the flag will be transmitted at the 16x16 block level. For POBMC$_{\text{H-II}}$, one additional bit is used to signal the δ value.

It is observed from Table 3.2 that Bilinear$_{\text{H-I}}$ performs similarly to 263$_{\text{H-I}}$. Essentially, they both use a fixed window function and thus suffer from the same problem of having to use the MV replication mechanism to address variable block-size partitioning. As expected, adapting window functions on the fly is beneficial, and the benefits of our parametric solutions are more obvious. In the IPPP case, POBMC$_{\text{H-I}}$ has an average BD-rate saving of 2.9%, with a minimum of 0.9% and a maximum of 9.6%. Adding one more OBMC option with $\delta = 0$ (POBMC$_{\text{H-II}}$) further improves performance slightly, giving, on average, 0.1-0.6% extra gains. These results provide only a lower bound on what is achievable, because the creation of

POBMC predictors currently uses motion parameters[4] optimized for BMC. It is for the same reason that the gains of POBMC (and similarly, of the other OBMC schemes) over the anchor decrease radically when the IBBB or Hierarchical B structure is in use. Note that both bi-prediction and POBMC are multi-hypothesis MCP techniques, and we are comparing a well-optimized bi-prediction with a sub-optimal POBMC. Another cause of this performance decline is the increased use of the SKIP and Direct modes, especially in the Hierarchical B sequence. In those modes, POBMC is not functioning. Even so, $\text{POBMC}_{\text{H-II}}$ still yields 0.9-5.8% and 0.7-3.9% BD-rate savings in the IBBB and Hierarchical B sequences, respectively.

The obvious inefficiency of POBMC can be improved by optimizing motion parameters. The rightmost column ($\text{POBMC}_{\text{H-II}}$+MD) of Table 3.2 presents the results when partition choices (and only partition choices) are additionally adapted for POBMC at a higher computational cost (Appendix). As can be seen, the performance improves further in the IPPP case, adding up to an average saving of 5.2%, whereas the improvements are much smaller in the remaining cases. To see why, Fig. 3.9 contrasts the mode distributions before and after this optimization, where the percentage of each mode indicates its average spatial coverage in a frame, i.e., the number of pixels coded by that particular mode. It is observed that despite the use of suboptimal motion parameters, POBMC is still more efficient than single-hypothesis MCP. It thus tends to favor the use of larger partitions, thereby reducing the overhead for signaling motion parameters. However, the impact of this suboptimality can be such that the superiority of POBMC over bi-prediction becomes modest. This explains why POBMC is enabled less frequently in the IBBB or Hierarchical B sequence and why the increase in the use of larger

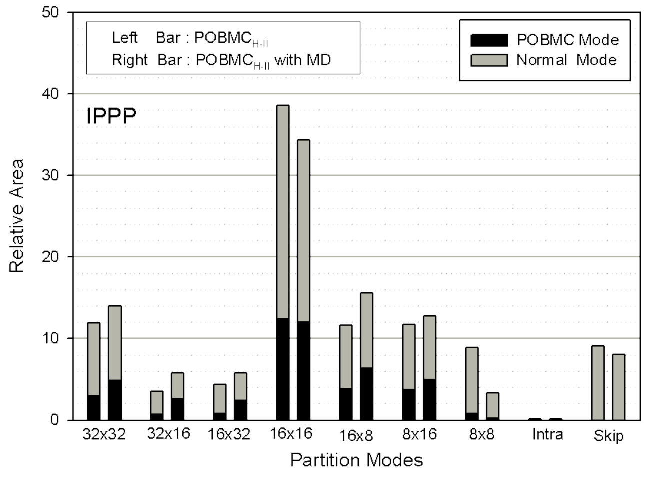
[4]Here, the motion parameters include the partition choice, the prediction type (forward-, backward- or bi-prediction) for each partition, as well as the corresponding MV(s).

partitions is not as significant as in the former case. Nevertheless, it is expected to perform better if the MVs and prediction directions are further optimized.

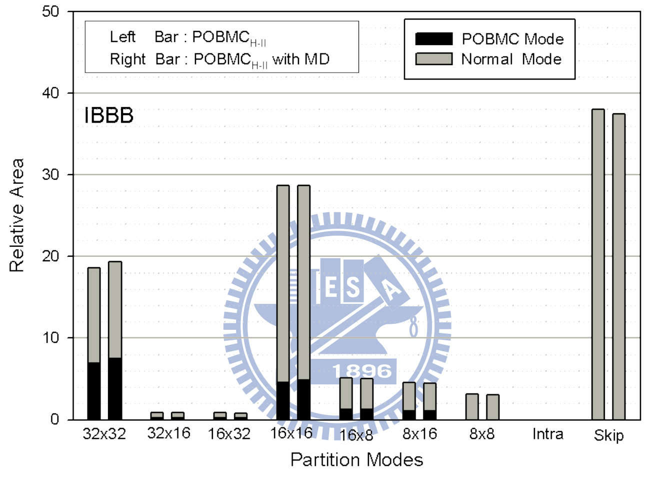
3.4.1.2 Comparison of In-Loop Filters and the Combined Effects

Having compared the compression performance of various OBMC schemes, we now proceed to show how POBMC performs relative to EAIF and QALF. We begin by contrasting the functions, implementations, and optimization criteria of these filters in Table 3.3. As can be seen, they all form a predictor (at every pixel position) from a weighted average of the reference samples, using the Wiener or least-squares criterion. However, what samples are involved depends on their main functions. For example, to address motion uncertainty, POBMC takes in those pointed to by the current and neighboring block MVs, $I'_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$, some of which may be at sub-pixel positions. To generate sub-pixel samples, EAIF interpolates between adjacent integer samples, with a filter support lying mainly around $\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)$, where $\widehat{\mathbf{v}}(\mathbf{s}_c)$ is the current MV in integer precision. The small deviation $\mathbf{d}_i$ (in units of integer samples) from $\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)$ partly helps to mitigate the problem of motion uncertainty. In complete analogy with EAIF, QALF also linearly combines nearby integer samples, but for a purpose of smoothing out quantization noise present in the reference frame. The filtered image $P_{k-1}(\mathbf{s})$, when used for MCP, yields a form similar to that of EAIF: $P_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)) = \sum_{i\in\mathcal{I}_3} w_i I'_{k-1}(\mathbf{s}+\widehat{\mathbf{v}}(\mathbf{s}_c)+\mathbf{d}_i)$. While averaging the reference samples, these filters all reduce noise to some extent. Their functions overlap, but each has its own emphasis. As a result, they possess very different filter characteristics: POBMC has a pixel-adaptive filter function, whereas that of EAIF and that of QALF are sub-pixel-dependent and fixed[5], respectively.

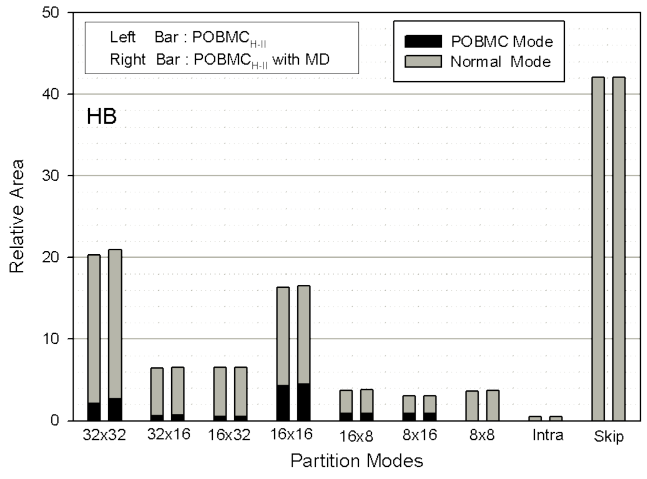
[5]Note that some advanced QALF algorithms [14][18] feature a pixel-adaptive filter.



(a) IPPP structure



(b) IBBB structure



(c) HB structure

Figure 3.9: Mode distribution comparison of POBMC$_{H\text{-}II}$ and POBMC$_{H\text{-}II}$+MD using testing sequence "BQSquare".

Table 3.2: BD-rate Saving Comparison of Various OBMC Schemes

Category		Non-parametric					
Scheme		$263_{\text{H-I}}$			$\text{Bilinear}^{*}_{\text{H-I}}$		
GOP		IPP	IBB	HB	IPP	IBB	HB
HD	S03	0.2	0.8	0.4	0.4	0.9	0.4
	S04	0.3	0.3	0.3	0.3	0.3	0.4
	S05	0.4	0.7	0.2	0.5	0.7	0.3
	S06	2.0	0.7	0.3	2.1	0.8	0.2
	S07	2.9	1.1	0.9	3.1	1.0	0.9
	Avg.	**1.2**	**0.7**	**0.4**	**1.3**	**0.7**	**0.4**
WVGA	S08	-0.2	-0.1	-0.2	-0.1	-0.2	0.2
	S09	0.3	0.4	0.3	0.4	0.3	0.3
	S10	0.0	0.2	-0.1	0.2	0.3	0.2
	S11	0.2	0.6	0.3	0.3	0.7	0.3
	Avg.	**0.1**	**0.3**	**0.1**	**0.2**	**0.3**	**0.3**
QWVGA	S12	0.1	0.3	0.2	0.1	0.4	0.2
	S13	0.8	0.8	0.4	0.9	0.7	0.4
	S14	-0.1	-0.1	-0.2	0.1	0.2	-0.1
	S15	0.1	0.2	0.2	0.1	-0.1	0.2
	Avg.	**0.2**	**0.3**	**0.2**	**0.3**	**0.3**	**0.2**
720p	S16	1.0	0.3	0.6	1.1	0.2	0.7
	S17	1.6	1.8	0.8	1.5	2.0	0.7
	S18	1.0	0.5	0.5	1.2	0.4	0.6
	Avg.	**1.2**	**0.8**	**0.6**	**1.3**	**0.8**	**0.7**
Overall		**1.1**	**0.6**	**0.4**	**1.2**	**0.6**	**0.4**

Category		Parametric											
Scheme		$\text{Tao}_{\text{H-I}}$			$\text{POBMC}_{\text{H-I}}$			$\text{POBMC}_{\text{H-II}}$			$\text{POBMC}_{\text{H-II}}\text{+MD}$		
GOP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	1.0	1.4	0.6	5.0	2.3	1.2	5.1	2.7	1.7	7.2	2.8	1.8
	S04	0.7	0.5	0.5	0.9	0.9	1.2	1.4	1.1	1.9	2.6	1.4	2.0
	S05	1.7	1.4	0.5	3.3	2.3	1.0	3.6	2.6	1.6	5.9	2.8	1.6
	S06	2.3	1.2	0.4	3.5	2.1	1.1	3.6	2.5	1.6	5.7	2.9	1.6
	S07	7.8	1.9	1.4	9.6	3.1	3.2	10.2	3.7	3.9	13.6	4.0	4.2
	Avg.	**2.7**	**1.3**	**0.7**	**4.3**	**2.1**	**1.5**	**4.8**	**2.5**	**2.1**	**7.0**	**2.8**	**2.2**
WVGA	S08	1.0	0.4	0.4	1.8	1.0	0.7	1.9	1.5	1.1	3.4	1.9	1.4
	S09	1.4	0.8	0.5	2.6	1.3	1.1	2.9	1.6	1.3	4.6	1.8	1.4
	S10	1.3	0.4	0.4	1.4	0.8	0.7	1.8	0.9	0.9	3.5	0.9	1.0
	S11	1.5	0.8	0.4	2.0	1.4	1.0	2.4	1.6	1.4	4.0	1.6	1.4
	Avg.	**1.3**	**0.6**	**0.4**	**2.0**	**1.1**	**0.9**	**2.3**	**1.4**	**1.2**	**3.9**	**1.6**	**1.3**
QWVGA	S12	1.5	0.5	0.4	2.2	1.1	0.6	2.4	1.6	0.8	4.0	1.8	0.9
	S13	2.1	1.4	0.6	3.3	2.3	1.2	3.8	2.5	1.2	5.9	3.0	1.2
	S14	1.0	0.4	0.3	1.4	0.9	0.7	1.7	1.3	0.9	3.0	1.5	1.1
	S15	1.1	0.5	0.4	1.1	0.9	0.6	1.4	1.3	0.7	2.7	1.7	0.7
	Avg.	**1.4**	**0.7**	**0.4**	**2.0**	**1.3**	**0.8**	**2.3**	**1.7**	**0.9**	**3.9**	**2.0**	**1.0**
720p	S16	1.4	0.5	0.8	2.1	1.0	1.5	2.4	1.5	2.4	4.1	1.8	2.5
	S17	2.2	3.3	1.2	4.1	5.5	2.2	4.7	5.8	2.6	7.7	6.2	2.6
	S18	1.5	0.8	0.8	2.9	1.4	1.5	3.5	1.7	1.8	5.3	1.8	2.0
	Avg.	**1.7**	**1.5**	**0.9**	**3.0**	**2.7**	**1.7**	**3.5**	**3.0**	**2.3**	**5.7**	**3.3**	**2.4**
Overall		**1.9**	**1.0**	**0.6**	**2.9**	**1.8**	**1.2**	**3.3**	**2.1**	**1.6**	**5.2**	**2.4**	**1.7**

*: the bilinear window function shown in Fig. 3.2(c) is used.

Positive values mean bitrate savings, whereas negative values mean bitrate increments.

Table 3.4 compares the BD-rate savings of POBMC$_{\text{H-II}}$+MD, EAIF and QALF. In the IPPP case, POBMC and EAIF offer a nearly identical average BD-rate saving ($\sim$5.3%), although the results in individual test sequences are highly variable. The gain of QALF, by contrast, is about 2% higher. In particular, they all perform better in high-resolution sequences. It is worth noting that the performance impact due to the use of bi-prediction on EAIF or QALF is minimal to moderate (see the results for the IBBB and Hierarchical B structures). The impact is mostly negative for EAIF, but is positive in quite a few sequences for QALF. Overall, both exhibit a performance decline in these prediction sequences; however, the performance drop relative to the IPPP setting is comparatively much smaller. The reasons are twofold: first, these tools have less overlap with bi-prediction than POBMC, in terms of functionalities, and second, the SKIP and Direct modes can still benefit from enabling them.

Although POBMC has the least gain among these filters, it can be combined more efficiently with either of the other two filters. Table 3.5 presents the BD-rate savings of all possible joint applications of these filters. An interesting observation is that the combination of POBMC and QALF performs very close to or better than that of EAIF and QALF, even in the IBBB or Hierarchical B case where the single use of EAIF outperforms that of POBMC. This leads us to suspect that a considerable part of the gain from EAIF is actually due to an attenuation in quantization noise. In this regard, EAIF shares similarities with QALF. This conjecture is supported by another observation that EAIF is still advantageous to high-resolution sequences, in which the signal aliasing is supposed to be less severe. The results in the rightmost column of Table 3.5 also indicates that the benefits of additionally enabling EAIF on top of POBMC+QALF are quite limited, except

Table 3.3: Comparison of In-Loop Prediction Filters

Filter	Function	Predictor Formulation	Filter Optimization
POBMC	Motion Uncertainty	$P_k(\mathbf{s}) = \sum_{i\in\mathcal{I}_1} w_i(\mathbf{s})I'_{k-1}(\mathbf{s}+\mathbf{v}(\mathbf{s}_i))$	$E\left\{(I_k(\mathbf{s}) - P_k(\mathbf{s}))^2\right\}$
EAIF	Aliasing	$P_k(\mathbf{s}) = \sum_{i\in\mathcal{I}_2} w_i(\mathbf{v}(\mathbf{s}_c))I'_{k-1}(\mathbf{s}+\hat{\mathbf{v}}(\mathbf{s}_c)+\mathbf{d}_i)$	$\sum_{\mathbf{s}}(I_k(\mathbf{s}) - P_k(\mathbf{s}))^2$
QALF	Quantization Noise	$P_{k-1}(\mathbf{s}) = \sum_{i\in\mathcal{I}_3} w_i I'_{k-1}(\mathbf{s}+\mathbf{d}_i)$	$\sum_{\mathbf{s}}(I_{k-1}(\mathbf{s}) - P_{k-1}(\mathbf{s}))^2$

I'_{k-1} : reference frame with coding noise

Table 3.4: BD-rate Saving Comparison of Various In-loop Filters

Scheme		POBMC$_{\text{H-II}}$+MD			EAIF			QALF		
GOP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	7.2	2.8	1.8	9.8	5.6	2.9	12.8	9.3	9.1
	S04	2.6	1.4	2.0	2.6	2.0	2.2	2.8	3.2	4.4
	S05	5.9	2.8	1.6	3.2	2.1	2.8	5.8	4.4	5.7
	S06	5.7	2.9	1.6	7.2	4.3	3.5	8.9	6.3	6.0
	S07	13.6	4.0	4.2	9.3	8.7	7.4	10.3	15.0	11.7
	Avg.	**7.0**	**2.8**	**2.2**	**6.4**	**4.5**	**3.8**	**8.1**	**7.6**	**7.4**
WVGA	S08	3.4	1.9	1.4	6.2	8.1	3.9	10.7	11.1	7.9
	S09	4.6	1.8	1.4	4.0	3.4	2.3	4.7	3.8	3.2
	S10	3.5	0.9	1.0	4.5	2.6	3.3	5.2	7.9	9.1
	S11	4.0	1.6	1.4	1.9	1.5	1.1	2.2	2.5	2.4
	Avg.	**3.9**	**1.6**	**1.3**	**4.2**	**3.9**	**2.6**	**5.7**	**6.3**	**5.7**
QWVGA	S12	4.0	1.8	0.9	3.7	2.2	2.4	5.3	3.2	3.5
	S13	5.9	3.0	1.2	8.8	8.0	7.5	9.1	15.0	18.0
	S14	3.0	1.5	1.1	2.0	1.7	1.4	2.8	3.6	4.1
	S15	2.7	1.7	0.7	0.5	0.5	0.5	1.8	1.3	1.9
	Avg.	**3.9**	**2.0**	**1.0**	**3.8**	**3.1**	**3.0**	**4.7**	**5.8**	**6.9**
720p	S16	4.1	1.8	2.5	7.1	4.7	6.3	12.4	8.1	8.2
	S17	7.7	6.2	2.6	8.4	10.1	7.6	11.7	13.0	10.7
	S18	5.3	1.8	2.0	8.4	4.5	6.6	10.5	6.6	7.6
	Avg.	**5.7**	**3.3**	**2.4**	**8.0**	**6.4**	**6.9**	**11.5**	**9.2**	**8.8**
Overall		**5.2**	**2.4**	**1.7**	**5.5**	**4.4**	**3.9**	**7.3**	**7.1**	**7.1**

in few low-resolution sequences, such as S10 and S13, where the aliasing may still be significant.

3.4.2 Complexity Analyses

3.4.2.1 Encoding and Decoding Times

This section compares the software runtimes of the algorithms discussed above to provide a rough indication of their complexity characteristics. For the runtime measurements, a single machine equipped with Intel Core i7-860 CPU (2926MHz), 8 GB RAM, and SATA-2 hard drive is used to run encoding or decoding in a single thread. The execution time is measured on Windows Vista 64-bit SP1 using NTimer. YUV writing is enabled during encoding. When interpreting the

Table 3.5: BD-rate Saving Comparison of Various Combinations of In-loop Filters

Scheme		POBMC+EAIF			POBMC+QALF			EAIF+QALF			All Enabled		
GOP		IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB	IPP	IBB	HB
HD	S03	11.3	7.1	4.4	13.9	10.8	10.4	13.9	10.2	10.0	15.1	11.8	11.0
	S04	4.0	2.9	3.4	4.4	4.1	5.5	4.0	3.5	5.2	5.2	4.6	6.7
	S05	8.2	4.1	4.0	10.8	6.1	6.9	6.6	4.5	6.2	11.1	5.4	7.4
	S06	10.5	6.0	4.8	12.0	8.0	6.7	10.9	7.8	7.5	13.6	9.3	8.3
	S07	18.9	11.2	10.2	22.4	17.6	14.0	13.8	16.1	13.8	24.0	17.6	14.7
	Avg.	**10.6**	**6.3**	**5.4**	**12.7**	**9.3**	**8.7**	**9.8**	**8.4**	**8.5**	**13.8**	**9.7**	**9.6**
WVGA	S08	10.6	8.9	4.9	15.0	11.2	8.5	13.2	13.5	8.9	16.6	15.2	9.8
	S09	7.5	4.3	3.0	8.1	5.1	4.2	7.1	5.2	4.9	9.9	6.2	6.0
	S10	8.8	2.7	4.0	11.5	8.5	9.8	7.6	8.2	10.1	14.0	9.2	10.6
	S11	5.0	2.2	1.9	5.8	3.4	3.2	3.4	3.3	3.1	6.8	3.5	3.9
	Avg.	**8.0**	**4.5**	**3.5**	**10.1**	**7.1**	**6.4**	**7.8**	**7.6**	**6.8**	**11.8**	**8.5**	**7.6**
QWVGA	S12	6.0	3.1	2.9	8.2	4.6	4.0	6.1	3.8	4.5	8.3	4.9	4.9
	S13	15.9	9.9	8.3	20.0	17.4	18.7	13.8	15.9	19.6	24.6	17.7	20.3
	S14	5.3	2.3	2.2	7.1	4.3	4.8	3.7	3.8	4.4	7.8	4.6	4.9
	S15	3.2	1.4	0.7	4.0	2.4	2.3	2.1	1.6	2.4	4.3	2.7	2.8
	Avg.	**7.6**	**4.2**	**3.5**	**9.8**	**7.2**	**7.5**	**6.4**	**6.3**	**7.7**	**11.3**	**7.5**	**8.3**
720p	S16	9.6	5.3	7.8	14.5	8.9	9.7	13.5	8.5	8.8	14.9	9.1	10.6
	S17	14.9	14.7	9.6	18.6	17.7	10.5	12.8	13.7	12.3	18.8	18.1	13.9
	S18	11.1	5.8	8.1	14.0	7.2	8.8	12.0	7.7	10.0	14.8	9.4	10.9
	Avg.	**11.9**	**8.6**	**8.5**	**15.7**	**11.3**	**9.7**	**12.8**	**10.0**	**10.4**	**16.2**	**12.2**	**11.8**
Overall		**9.4**	**5.7**	**5.0**	**11.9**	**8.6**	**8.0**	**9.0**	**8.0**	**8.2**	**13.1**	**9.3**	**9.2**

Table 3.6: Runtime Comparison of Various In-Loop Filters

Scheme		$263_{\text{H-I}}$	$\text{Tao}_{\text{H-I}}$	$\text{POBMC}_{\text{H-I}}$	$\text{POBMC}_{\text{H-II}}$	$\text{POBMC}_{\text{H-II}}$+MD	EAIF	QALF
Enc	IPP	1.47	1.61	1.63	1.71	2.01	2.22	1.40
	IBB	1.37	1.50	1.57	1.62	1.97	2.00	1.45
	HB	1.32	1.42	1.48	1.60	2.00	2.12	1.25
Dec	IPP	1.40	1.48	1.64	1.68	1.73	1.41	1.53
	IBB	1.35	1.45	1.51	1.57	1.66	1.25	1.56
	HB	1.48	1.54	1.62	1.64	1.68	1.46	1.43

results, one should be aware that the software runtime can be more related to the implementation quality of an algorithm than to its intrinsic complexity.

The upper half of Table 3.6[6] presents the encoding times of these algorithms in the form of average time ratios[7] relative to the anchor encoding. As can be seen from the left most column, adding OBMC option increases the encoding time by 30-50%. This is attributed to the extra computation required for mode decision and pixel-adaptive weighting. Moreover, computing weighting coefficients by parametric solutions (e.g., $\text{Tao}_{\text{H-I}}$ and $\text{POBMC}_{\text{H-I}}$) involves floating-point arithmetic, thereby taking 10-20% longer encoding time, and as expected, the more flexible $\text{POBMC}_{\text{H-II}}$ increases the encoding time further (˜10%). When $\text{POBMC}_{\text{H-II}}$+MD is in use, the encoding time almost doubles due to multi-pass R-D based mode decision. Similar results can also be seen in EAIF. By contrast, QALF introduces only a moderate time increase (20-45%), the reason probably being that determining restoration filter and segmentation map may still be computationally less demanding than performing mode decision, which requires actually coding all the MBs using all possible options. We finally note that the impacts of these algorithms on encoding time appear consistent regardless of prediction structures.

While there is a wide variation between the encoding times of these algorithms, the difference between their decoding times is less significant. Still, performing OBMC leads to slow decoding times: relative to the anchor decoding (which decodes bit-streams produced by the anchor encoding), a 40-70% increase of decoding time is observed (the lower half of Table 3.6). The main reasons include the increased memory accesses needed to fetch multiple predictors and the extra operations required for weighting them in a pixel-adaptive manner. Another cause

[6]$\text{Bilinear}_{\text{H-I}}$ is of comparable complexity to $263_{\text{H-I}}$ and gives similar runtime results.

[7]The average of the encoding time ratios of some specific algorithm and the anchor over all test sequences and QP settings.

Table 3.7: BD-rate and Runtime Comparisons of POBMC$_{\text{H-II}}$+MD and Its Simplified Version

Scheme	POBMC$_{\text{H-II}}$+MD			Simplified		
GOP	IPP	IBB	HB	IPP	IBB	HB
HD	7.0	2.8	2.2	5.9	2.4	2.3
WVGA	3.9	1.6	1.3	3.2	1.5	1.1
QWVGA	3.9	2.0	1.0	3.3	1.9	0.9
720p	5.7	3.3	2.4	4.5	3.2	2.3
Overall	**5.2**	**2.4**	**1.7**	**4.3**	**2.2**	**1.6**
			Time Ratios			
Enc	2.01	1.97	2.00	1.46	1.40	1.33
Dec	1.73	1.66	1.68	1.34	1.32	1.30

of this increase is an implementation expedient, which requires parsing the bitstream twice (Appendix). Again, the parametric solutions incur a higher penalty than the non-parametric ones, and ours appear to be more complex. The latter, however, is not because our schemes are computationally more demanding, but because the OBMC mode is selected more often when they are in use. Recall that these algorithms allow the encoder to switch adaptively between BMC and OBMC. We also find that when using non-parametric windows, OBMC exhibits similar decoding complexity characteristics to EAIF or QALF. Essentially, they all perform filtering of pixel intensities based on pre-computed filters.

3.4.2.2 Simplification

In the current implementation of POBMC (and the other OBMC schemes), generating a prediction value for a pixel may require MVs associated with the neighboring blocks to its right or below. Such a non-causal access of MVs requires motion parameters for the entire picture to be first generated and buffered, which induces a large delay in both the encoding and the decoding processes. This, however, can be avoided by utilizing only those MVs in the causal neighborhood. Some performance loss is expected, as the pixels in the bottom-right quarter of a prediction block may not benefit much from OBMC. Table 3.7 compares the BD-rate savings and the runtimes of POBMC$_{\text{H-II}}$+MD and its simplified version based on this notion. Specifically, the simplifications include a causal MV access and the use of only one δ option ($\delta = 16$). From the results, we see that the performance of the simplified POBMC is only slightly worse than that of POBMC$_{\text{H-II}}$+MD. However, both the encoder and the decoder run much faster. This is mainly because on the encoder side, the determination of motion compensation methods and partition choices can now be done in one single pass, and on the decoder side, there is no need to first extract all motion parameters. To further reduce the complexity of POBMC, additional constraints can be placed on the number of MVs involved and on their ranges. Our preliminary study shows that using only four neighboring MVs, truncated to be in the range of current MV +/- four pixels, can achieve 10-15% runtime reductions without introducing noticeable performance degradation. In addition, the techniques discussed in [22] can be applied to POBMC. Lastly, we want to point out that although (3.11) inevitably requires floating-point arithmetic, (3.12) can have a fixed-point implementation. This can be seen by considering the prediction of a pixel based on three MVs. In this case, the OBMC weight associated with one of these MVs is computed as

$$\begin{aligned} w_1^* &= \frac{\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)}}{\frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)} + \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)} + \frac{1}{\widehat{r}^2(\mathbf{s},\mathbf{s}_3)}} \\ &= \frac{\widehat{r}^2(\mathbf{s},\mathbf{s}_2)\widehat{r}^2(\mathbf{s},\mathbf{s}_3)}{\widehat{r}^2(\mathbf{s},\mathbf{s}_1)\widehat{r}^2(\mathbf{s},\mathbf{s}_2) + \widehat{r}^2(\mathbf{s},\mathbf{s}_2)\widehat{r}^2(\mathbf{s},\mathbf{s}_3) + \widehat{r}^2(\mathbf{s},\mathbf{s}_3)\widehat{r}^2(\mathbf{s},\mathbf{s}_1)}. \end{aligned}$$

Obviously, with proper scaling and rounding, the computations of the numerator, the denominator, and the quotient can all be done in fixed-point arithmetic, even though it still takes quite some work to obtain the result. Compared with (3.2), which is adopted by the parametric solution in [24], both (3.11) and (3.12) require less computation. While (3.12) need not perform matrix inverse, (3.11) only has to invert a smaller matrix of dimension $(L-1) \times (L-1)$, compared to $L \times L$ in

(3.2).

3.5 Conclusion

In this dissertation, we introduced a parametric solution for OBMC (termed POBMC) and studied its properties from both theoretical and empirical aspects. In the course, we found it convenient to approximate block MVs as motion samples taken at block centers; this expedient helped us to conceptualize the combination of OBMC with variable block-size motion partitioning. Because in practice it is far from adequate to describe the location of a block MV as a deterministic variable, we amended our solution to reveal its random nature. The novelty of our scheme was shown to require only the geometric relations between the prediction pixel and its nearby block centers for computing the window function. It thus lends itself to reconstructing temporal predictors from any sparsely and irregularly sampled motion data. The superiority of our scheme over similar previous works was confirmed by extensive experiments. Moreover, its performance was shown to be comparable to EAIF and QALF. Above all, it can work with either (or all) of them to yield an improvement that is nearly the sum of their separate effects.

Along with the coding tools in KTA2.4r1, part of this work [8] was submitted, for subjective viewing tests, in response to the HEVC Call for Proposals issued jointly by MPEG and VCEG in April, 2010 [2]. It was ranked 12 overall and 10 in low delay configurations among 27 proposals, in terms of the average mean opinion score [3]. The notion of POBMC has recently been extended to develop a reduced-overhead bi-prediction scheme [15][17]. Owing to its promising results, the technique is currently being evaluated in a core experiment of JCT-VC.

Chapter 4

Bi-Prediction Combining TMP and BMC

4.1 Introduction

Using the data accessible to the decoder for motion inference has recently emerged as a promising technique for next generation video coding. Template matching prediction (TMP) is a typical example of utilizing the decoder-side information for motion inference. It estimates the motion vector (MV) for a target block on the decoder side by minimizing the matching error over the reconstructed pixels in its immediate inverse-L-shaped neighborhood (usually termed the template). Motion merging [28] and OBMC [19] also follow the same rationale. They both view the received motion data as a source of information about the motion field and forms a better prediction of a pixels intensity based on its own and/or nearby block MVs.

For the above reasons, we are led to develop a biprediction scheme, which incurs only the overhead for unidirectional prediction. The idea is to combine MVs resulting from the template and block matchings with OBMC. Of particular interest in this combination is that the template MV is inferred on the decoder side; it thus has only to signal one block MV while attaining biprediction performance.

The choice of a proper window function is critical in these applications. We approach this problem through the parametric OBMC framework in Chapter 3. The resulting window function is shown to resemble a particular type of geometry motion partitioning as shown in Fig. 4.4(c) with its MVs located on the diagonal running from the upper left to the lower right. Particularly, asymmetric-like partitioning as shown in Fig. 4.4(a),(b) arises when the template region is of rectangular shape and is located to the left or on the top of a target block.

To gain more insights into this biprediction scheme, Fig. 4.1 plots the mean-square prediction error surface with the single use of TMP MV for motion compensation of the target block. It is seen that this MV tends to minimize the prediction error in the upper left quarter, a result that is intuitively agreeable since it approximates the true motion associated with the template centroid. This observation implies that the block MV should be managed to contribute more to minimizing the error in the remaining part, especially in the bottom right region. We thus also propose in this chapter a new search criterion for the block MV to achieve this objective.

The proposed bipredictioin scheme can be further generalized to allow a template-matching-free implementation, which replaces the template MV with a decoded MV specified by a mechanism similar to motion merging. The approach is to replace the template MV with one of those for prediction units (PUs) in a causal neighborhood of the current PU. In particular, the selection of MV is made adaptive by adopting the same signaling mechanism as for motion merging [28]. Depending on the direction in which the MV (and other motion parameters) is derived, a separate window function is applied for OBMC. Remarkably, this scheme can produce, with less overhead, an effect similar to combining asymmetric/geometry motion partitions [14][18] and PU-based motion merging [28].

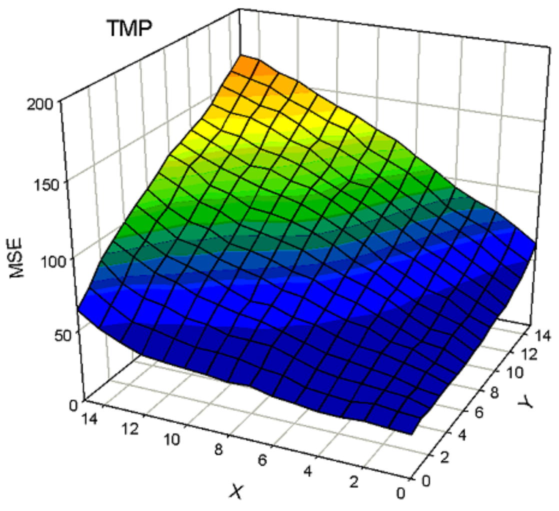


Figure 4.1: Mean-square prediction error surface of TMP using the sequence "Football".

In the following sections, we will first detail the concept of the joint application of TMP and BMC. A low complixity and template-matching-free implementation is then introduced. We will also show the experiments conducted using the HEVC reference software and the common test configurations to evaluate three variants of this scheme. The best of them achieves an average BD-rate saving of 2.2%, with a minimum of 0.2% and a maximum of 4.7%. It is observed that the average decoding time increases by 10% while the encoding time doubles.

4.2 Combining Template and Block Motion Compensation

In this section, we present the proposed bi-prediction scheme and the design of its window function.

4.2.1 Concept of Operation

Fig. 4.2 depicts its concept of operation. Like the conventional bi-prediction, it predicts a target block based on two predictors. These predictors however are weighted in a pixel-adaptive manner using POBMC, with one of them derived from a MV $\mathbf{v}_t$ found by template matching [13] and the other from the usual motion compensation. Since $\mathbf{v}_t$ can be inferred on the decoder side, this scheme has only to signal motion parameters for one block MV (denoted as $\mathbf{v}_b$). In this application, $\mathbf{v}_t$ cannot be specified discretionarily. It is thus important to find a $\mathbf{v}_b$ that, when applied jointly with $\mathbf{v}_t$, minimizes the prediction error over the target block $\mathcal{B}$. The question arises naturally what is the most appropriate choice for $\mathbf{v}_b$ and the combining weights which, along with the given $\mathbf{v}_b$, would result in minimal prediction residual? To answer the question, the overall prediction error of the proposed bi-prediction is first formulated as follows:

$$\arg \min_{\{\mathbf{v}_b,i\},\{w(\widetilde{\mathbf{s}})\}} \sum_i \sum_{\mathbf{s}\in\mathcal{B}} \Big(I(\mathbf{s}) - w_t(\widetilde{\mathbf{s}})\, \tilde{I}(\mathbf{s}+\mathbf{v}_{t,i}) - w_b(\widetilde{\mathbf{s}})\, \tilde{I}(\mathbf{s}+\mathbf{v}_{b,i}) \Big)^2 \qquad (4.1)$$

where $\widetilde{\mathbf{s}}$ denotes the pixel position within a block (i.e., relative to the block origin), $w_b(\widetilde{\mathbf{s}})$ denotes its window function, $w_t(\widetilde{\mathbf{s}}) = 1 - w_b(\widetilde{\mathbf{s}})$, and $\tilde{I}$ is the reference picture. Clearly, $w_b(\widetilde{\mathbf{s}})$ has a decisive effect on the value of $\mathbf{v}_{b,i}$ and thus on the prediction performance of this scheme.

4.2.2 Optimized Prediction

It is intuitive to use an iterative procedure for solving this problem as described by 4.1. Firstly, given an initial estimate of $\mathbf{v}_{b,i}$ and $w_b(\widetilde{\mathbf{s}})$, the motion vectors are refined conditioned on the given block motion field $\mathbf{v}_{b,i}$ and window function $w_b(\widetilde{\mathbf{s}})$. Conversely, the window function is refined conditioned on the given block motion field until the vector field and window function converge. Although the iterative procedure can ideally find the optimal solution, the process is time consuming and less instructive.

A more instructive way of solving this problem is to consider the notion of motion sampling and reconstruction through the wiener-based approach. To observe

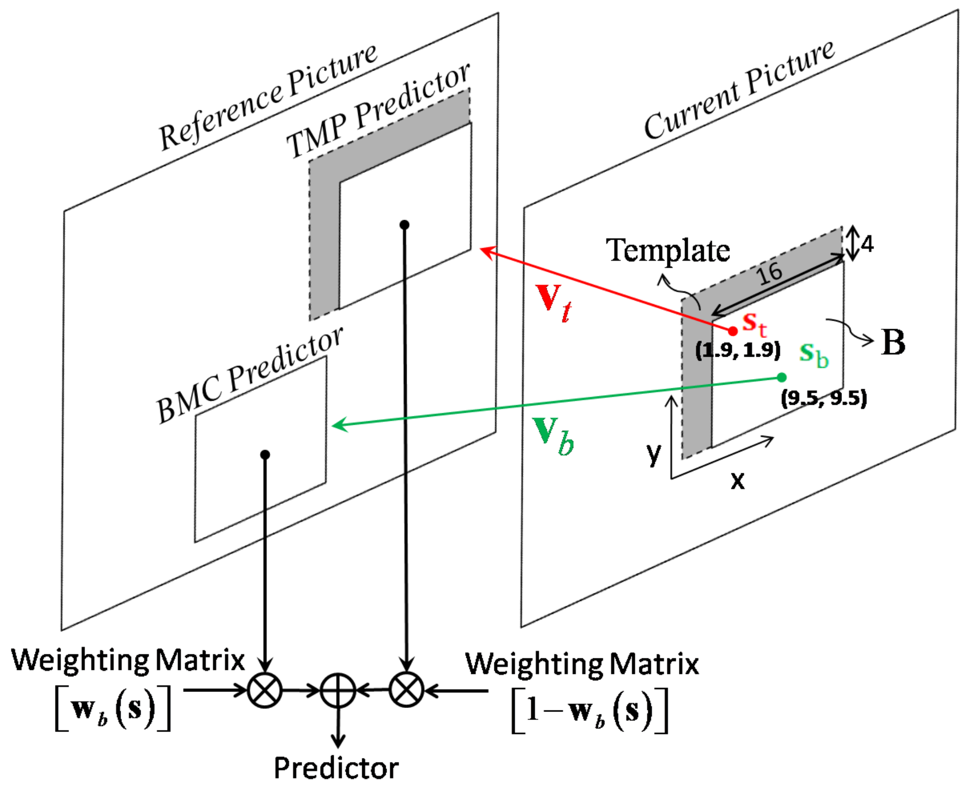


Figure 4.2: Joint application of TMP and BMC.

the statistical characteristic of the proposed bi-prediction scehme, the problem is reformulated as follow:

$$\sum_{\mathbf{s}\in\mathcal{B}} E\left\{\left(I(\mathbf{s}) - w_t(\tilde{\mathbf{s}})\,\tilde{I}(\mathbf{s}+\mathbf{v}(\mathbf{s}_t)) - w_b(\tilde{\mathbf{s}})\,\tilde{I}(\mathbf{s}+\mathbf{v}(\mathbf{s}_b))\right)^2\right\} \qquad (4.2)$$

Instead of using an iterative procedure to determine $w_b(\mathbf{s})$ (and thus $w_t(\mathbf{s})$), as is the case with the OBMC [19], we approach this problem by resorting to the parametric framework in Chapter 3. To proceed, $\mathbf{v}_t$ is approximated as the pixel true motion $\mathbf{v}(\mathbf{s}_t)$ at the template centroid $\mathbf{s}_t$, a justification of which can be found in [26]. However, we avoid making the same approximation for $\mathbf{v}_b$ because its search criterion is no longer to minimize the sum of squared prediction errors[1] (cf. (4.4)). It is replaced instead by the true motion $\mathbf{v}(\mathbf{s}_b)$ of some unknown pixel $\mathbf{s}_b$ in $\mathcal{B}$. We now cast the problem of determining $w_b(\mathbf{s})$ as the search for an $\mathbf{s}_b$

[1]A block MV approximates the pixel true motion at the block center only if its search criterion is to minimize the sum of squared prediction errors [30][24].

in $\mathcal{B}$ that minimizes the sum of mean-square prediction errors (SMSE) over $\mathcal{B}$ as indicated by 4.2.

Each term in the summation of (4.2) is simply the mean-square prediction error, produced by OBMC based on $\mathbf{v}(\mathbf{s}_t)$ and $\mathbf{v}(\mathbf{s}_b)$, at some pixel $\mathbf{s}$ and can be modeled with (3.5). For a given $\mathbf{s}_b$, (4.2) is minimized when each operand reaches its minimum, that is, according to (3.12), when $w_b(\mathbf{s}) = w_b^*(\mathbf{s}) = r^2(\mathbf{s};\mathbf{s}_t)/(r^2(\mathbf{s};\mathbf{s}_t) + r^2(\mathbf{s};\mathbf{s}_b))$. Noting this, we have:

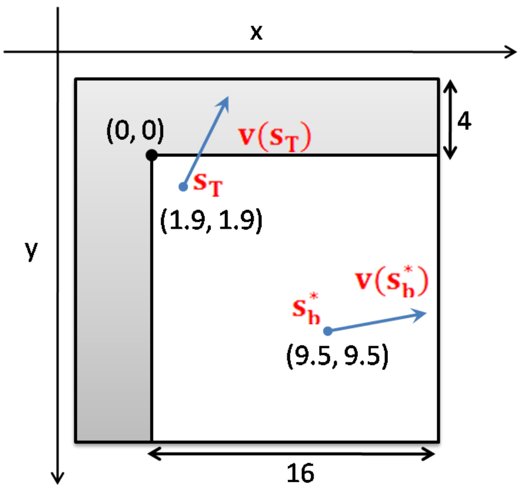
$$\mathbf{s}_b^* = \arg\min_{\mathbf{s}_b \in \mathcal{B}} \sum_{\mathbf{s}\in\mathcal{B}} \epsilon(w_t^*(\mathbf{s}))^2 r^2(\mathbf{s};\mathbf{s}_t) + (w_b^*(\mathbf{s}))^2 r^2(\mathbf{s};\mathbf{s}_b)). \tag{4.3}$$

Due to its non-linear nature, $\mathbf{s}_b^*$ has to be found by numerical simulations, i.e., we have to compute SMSE for every admissible location of $\mathbf{s}_b$. Fortunately, the computation is tedious but not difficult, and can be made offline. Once it has been solved, the $w_b^*(\mathbf{s})$ is immediate by (3.12). Applying this window function to (4.4), we get an optimized block matching criterion, with which a $\mathbf{v}_b^*$ approximating the pixel true motion at $\mathbf{s}_b^*$ can be found.

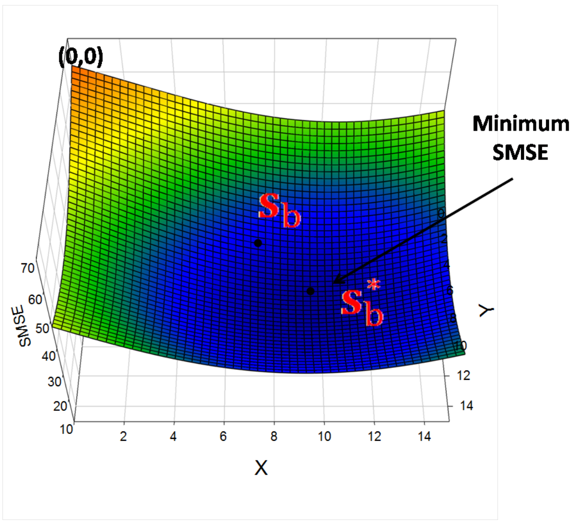
To get a sense of where $\mathbf{s}_b^*$ should be located, Fig. 4.3 (b) plots the SMSE as a function of its location. As can be observed, the SMSE becomes smaller when $\mathbf{s}_b$ sits in the bottom right quarter; a further precise calculation shows that its optimal location occurs at point (9.5,9.5) for a 16×16 target block. This is of no surprise because $\mathbf{v}_t$, located at the template centroid (1.9,1.9), has a higher correlation with the motion field in the upper left quarter and thus contributes more to minimizing the errors there. Intuitively, $\mathbf{s}_b$ should be so placed that the errors in the remaining part of $\mathcal{B}$ can be minimized.

4.2.3 Window Functions

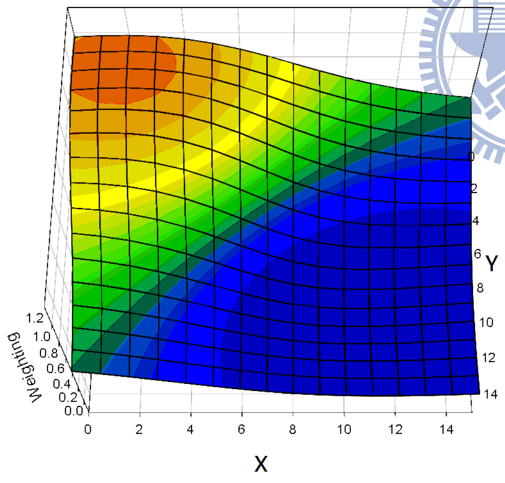
Fig. 4.3 (c) and (d) plot the window functions, $1 - w_b^*(\mathbf{s})$ and $w_b^*(\mathbf{s})$, for $\mathbf{v}_t$ and $\mathbf{v}_b$, respectively. As can be seen, their waveforms suggest a special type of geometry



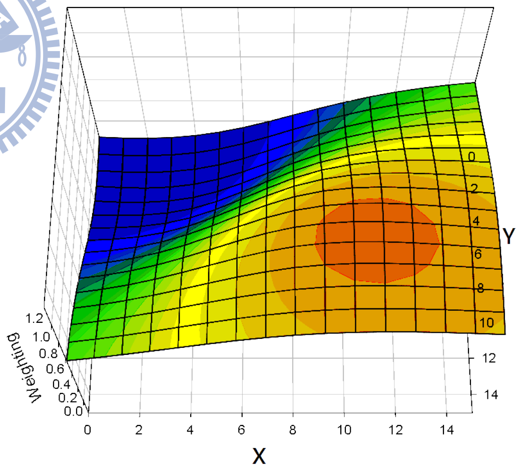
(a) InvL



(b) SMSE Surface



(a) $w_t^* (\mathbf{s})$



(b) $w_b^* (\mathbf{s})$

Figure 4.3: SMSE surface as a function of $\mathbf{S}_b$'s location, and the optimal window functions associated with $\mathbf{v}_t$ and $\mathbf{v}_b$, respectively.

motion partition [14] with two MVs located on the diagonal running from the upper left to the lower right. Following the same line of derivation, we can obtain window functions for other template designs. Some results are given in Fig. 4.4. In particular, asymmetric-like motion partitions [14][18] result when the template region locates directly to the left or above a target block (See Fig. 4.4). Two conceptual differences however are to be noted. First, unlike explicit geometry or asymmetric partitions, these implicit "soft" partitions incur less motion overhead (only one MV is to be signaled). Second, there is a strong interdependency between the transmitted and inferred MVs due to OBMC (cf. (4.4)).

4.2.4 Optimized Block Matching Criterion

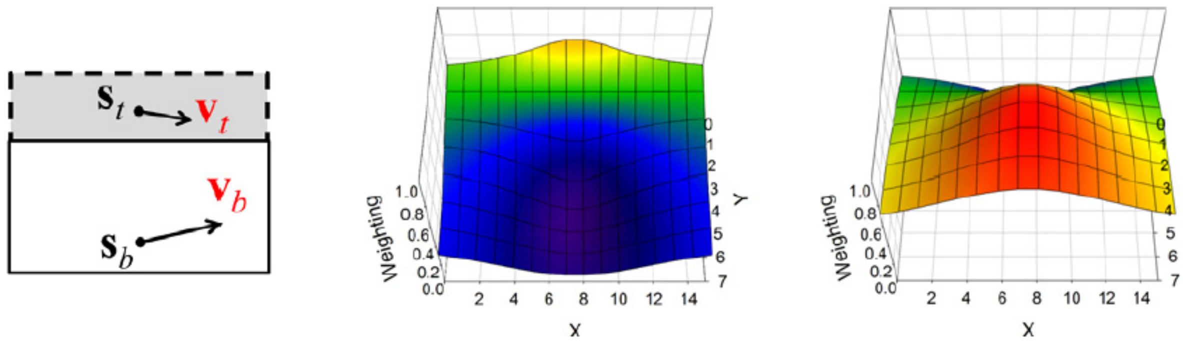
After the optimal window function is determined, in practical implementation, the optimal blcok motion vector for the target block is found by minimizing the modified search criterion:

$$\mathbf{v}_b^* = \arg\min_{\mathbf{v}_b} \sum_{\mathbf{s}\in\mathcal{B}} \left(I\left(\mathbf{s}\right) - w_t^*\left(\mathbf{s}\right) \tilde{I}\left(\mathbf{s}+\mathbf{v}_t\right) - w_b^*\left(\mathbf{s}\right) \tilde{I}\left(\mathbf{s}+\mathbf{v}_b\right) \right)^2, \tag{4.4}$$

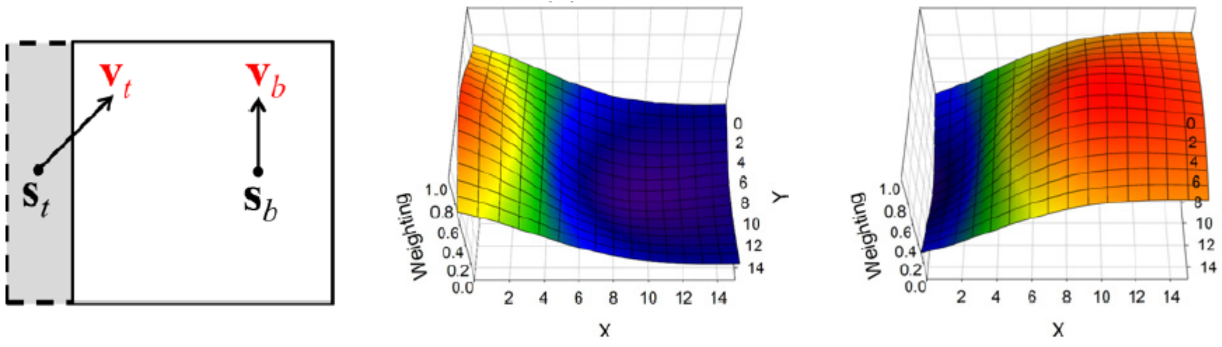
where $w_b^*\left(\mathbf{s}\right)$ and $w_t^*\left(\mathbf{s}\right)$ represent the optimal window function as shown in Fig. 4.3.

4.3 Extension with Motion Merging

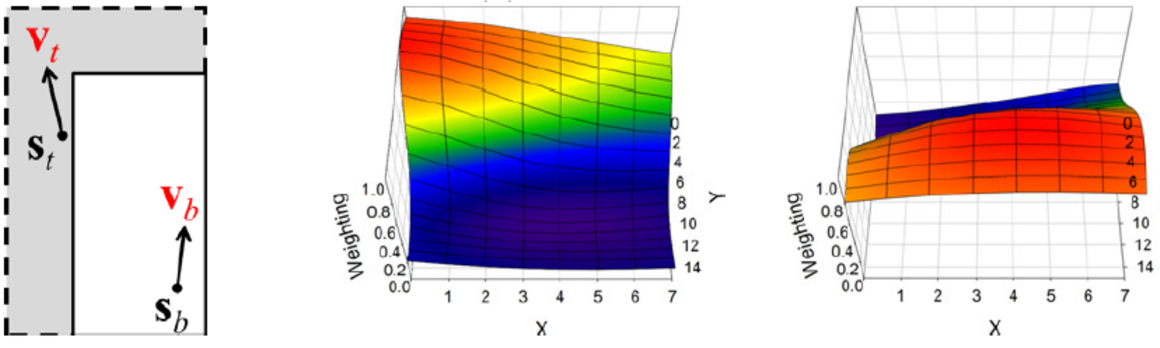
Recognizing that performing template matching not only increases the decoding complexity but also complicates the pipeline design of the decoder, we additionally propose a low-complexity and TMP-free implementation. This is accomplished by replacing the template MV $\mathbf{v}_t$ by one of those decoded MVs from neighboring partitions (cf. Fig. 4.5). In this way, the need to perform TMP is waived at the cost of extra bits. To minimize the overhead, we adopt the same signaling mechanism as for motion merging [28]. When enabled, it sends additional flags



(a) Rect-T



(b) Rect-L



(c) InvL

Figure 4.4: Window functions for typical template designs.

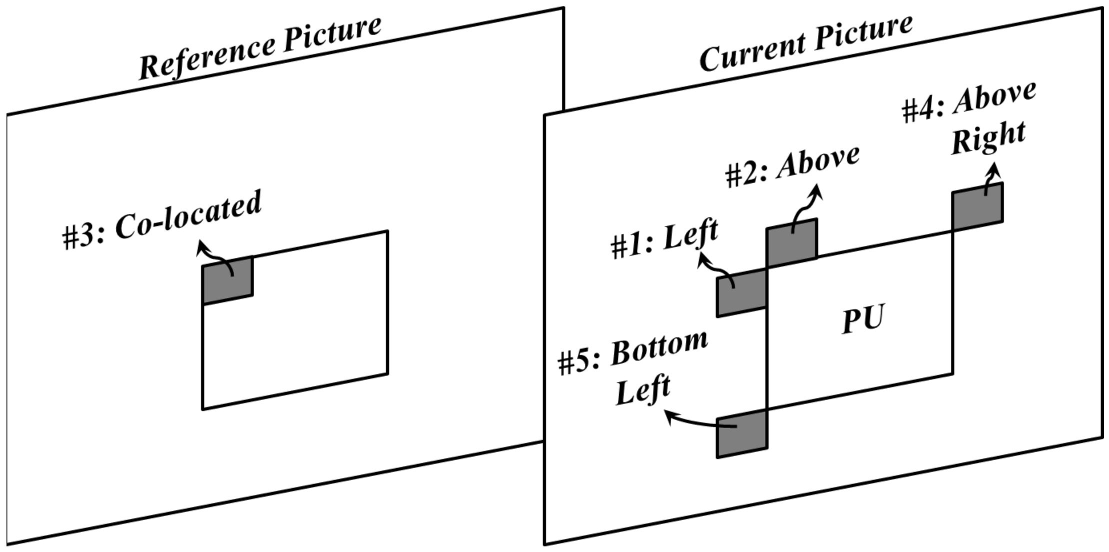
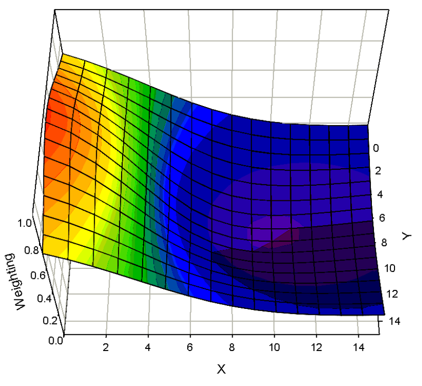
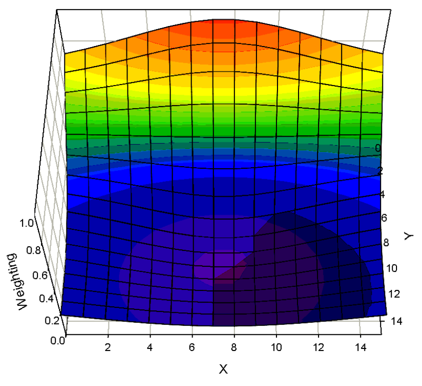


Figure 4.5: Adaptive motion merging and the approximation of $\mathbf{v}_t$.

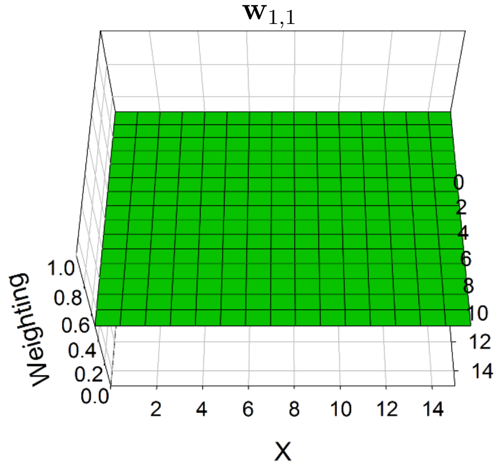
to indicate which merge candidate is selected from a predefined candidate set containing the motion parameters of the spatially neighboring partition, to the left, above, above-right, bottom-left the current one or of the co-located partition. The motion parameters of the selected candidate is then reused as $\mathbf{v}_t$. Depending on the inference direction, a separate window function is applied for OBMC. For example, in Fig. 4.5, if $\mathbf{v}_t$ is deduced from left, above, co-located, above-right or bottom-left merge candidate, the window function $\mathbf{w}_{1,1}$, $\mathbf{w}_{1,2}$, $\mathbf{w}_{1,3}$,$\mathbf{w}_{1,4}$, $\mathbf{w}_{1,5}$ (cf. Fig. 4.6) is selected, respectively. In other cases where left candidate and above candidate have the same motion parameter, the weighting function $\mathbf{w}_{1,6}$ as shown in Fig. 4.6 is used instead. Essentially, we treat the MVs deduced from each merge candidate as if they were the pixel true motion associated with the corresponding template centroids. Because these assumptions may not always hold true, we let the encoder to switch adaptively between this proposed mode and the usual inter mode.



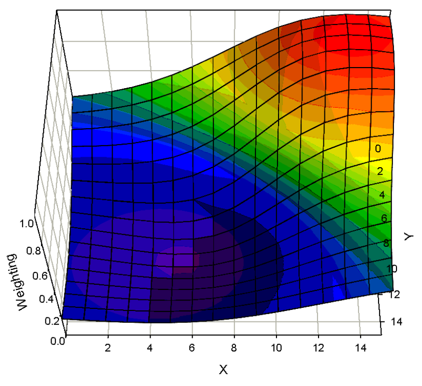
$\mathbf{w}_{1,1}$



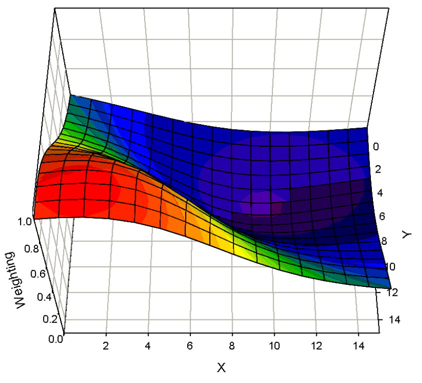
$\mathbf{w}_{2,1}$



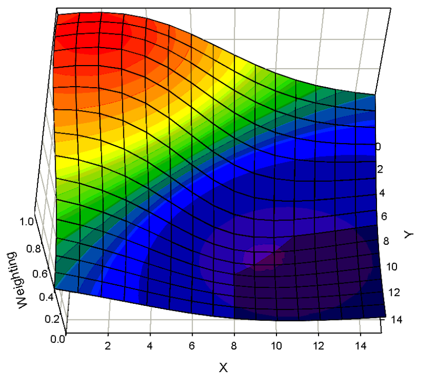
$\mathbf{w}_{3,1}$



$\mathbf{w}_{4,1}$



$\mathbf{w}_{5,1}$



$\mathbf{w}_{6,1}$

Figure 4.6: Window functions for different merge candidates.

4.4 Simplification with Motion Merging

To reduce the buffer size for the weighting coefficients and also to reduce the computational complexity of calculating the values of predicted pixels, the simplified window functions are proposed. In this simplified biprediction scheme with motion merging, a window function $\mathbf{w}_{m,n}$ can take values from either the set (0.125, 0.5, 0.875) or the set (0.125, 0.5, 0.75). The distribution of its weighting coefficients forms a partitioning of a PU into three non-overlapping regions, each corresponds to the application of some specific coefficient. As an illustration, Fig. 4.7 presents the various partitionings that can arise from the use of the window functions shown in Fig. 4.8 and Fig. 4.9. As can be seen, the coefficients 0.5 occupy a region that may distribute vertically, horizontally, or diagonally (at an angle of 45 or 135 degree).

To resize a window function according to the target PU size, the start position a and the width b of this region (see Fig. 4.7 for their definitions) are stored and can be referred as a look-up table when performing the proposed biprediction mode. To further improve the prediction efficiency, as can be seen in Fig. 4.8 and Fig. 4.9, two different sets of window functions are utilized for a 2Nx2N PU. In this simplified scheme, each 2Nx2N PU, when coded in the proposed mode, can select between two sets of window functions, denoted by $\mathbf{w}_{m,n}$ with m=1, , 6 and n=2,3 to adapt for fine turning the weighting coefficients. To indicate which set is in use, i.e., the value of n, one additional flag is sent for each 2Nx2N PU. For a chosen set of window functions, the parameter m is determined similarly by letting the encoder to switch adaptively between this proposed mode and the usual inter mode.

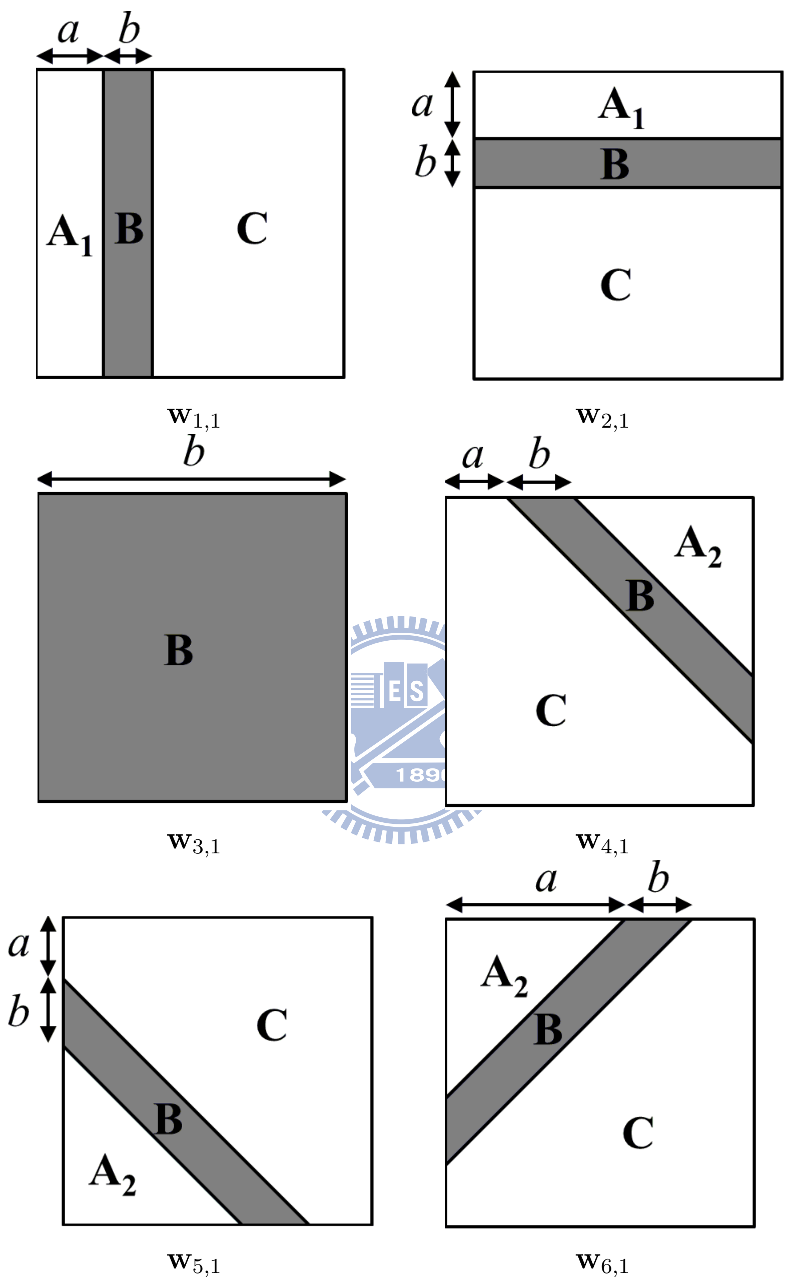
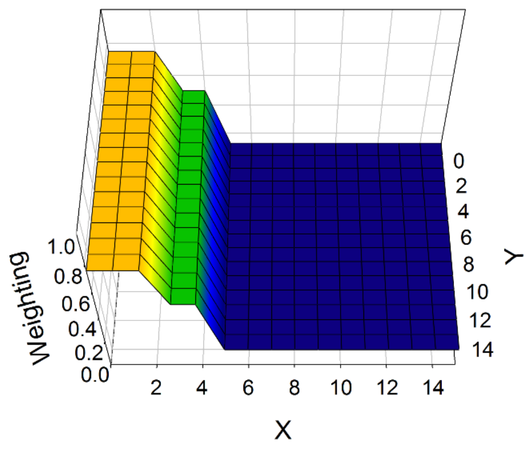
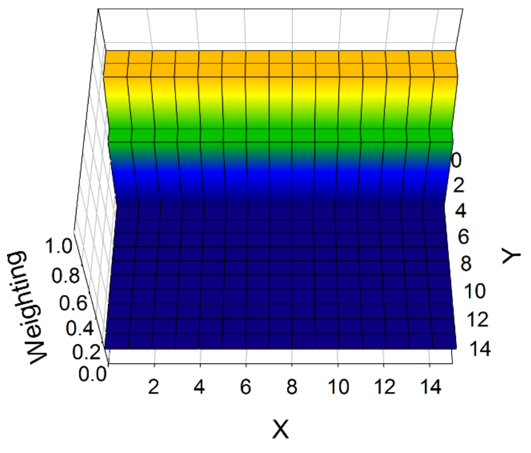


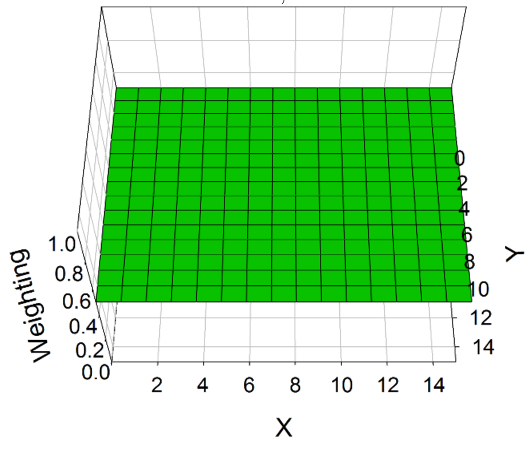
Figure 4.7: Partitioning of a 2Nx2N PU due to the application of the proposed window functions. In region A1, A2, B and C, $\mathbf{w}_{m,n}$(i, j)=0.75, 0.875, 0.5 and 0.125,respectively.



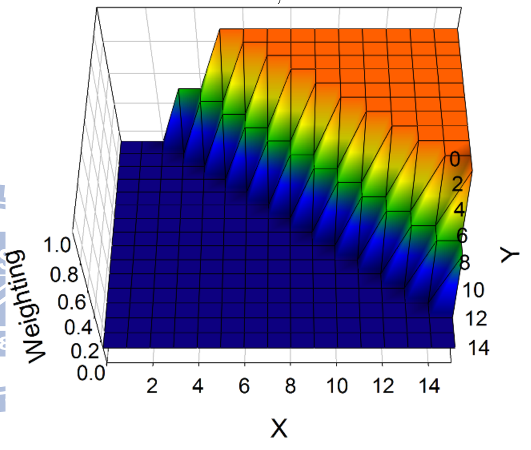
$\mathbf{w}_{1,2}$



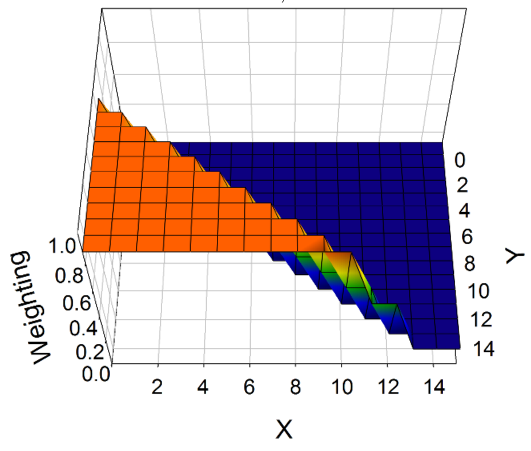
$\mathbf{w}_{2,2}$



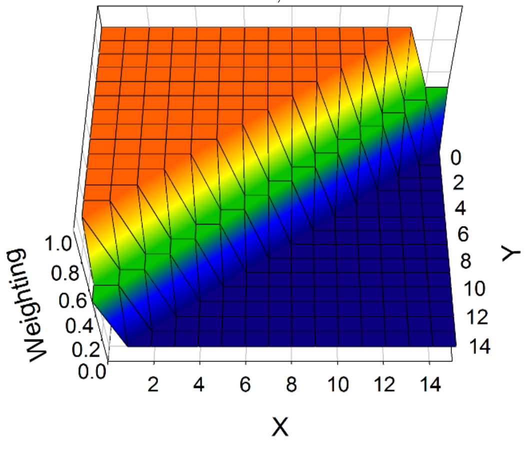
$\mathbf{w}_{3,2}$



$\mathbf{w}_{4,2}$



$\mathbf{w}_{5,2}$



$\mathbf{w}_{6,2}$

Figure 4.8: Simplified window functions set 1 for different merge candidates.

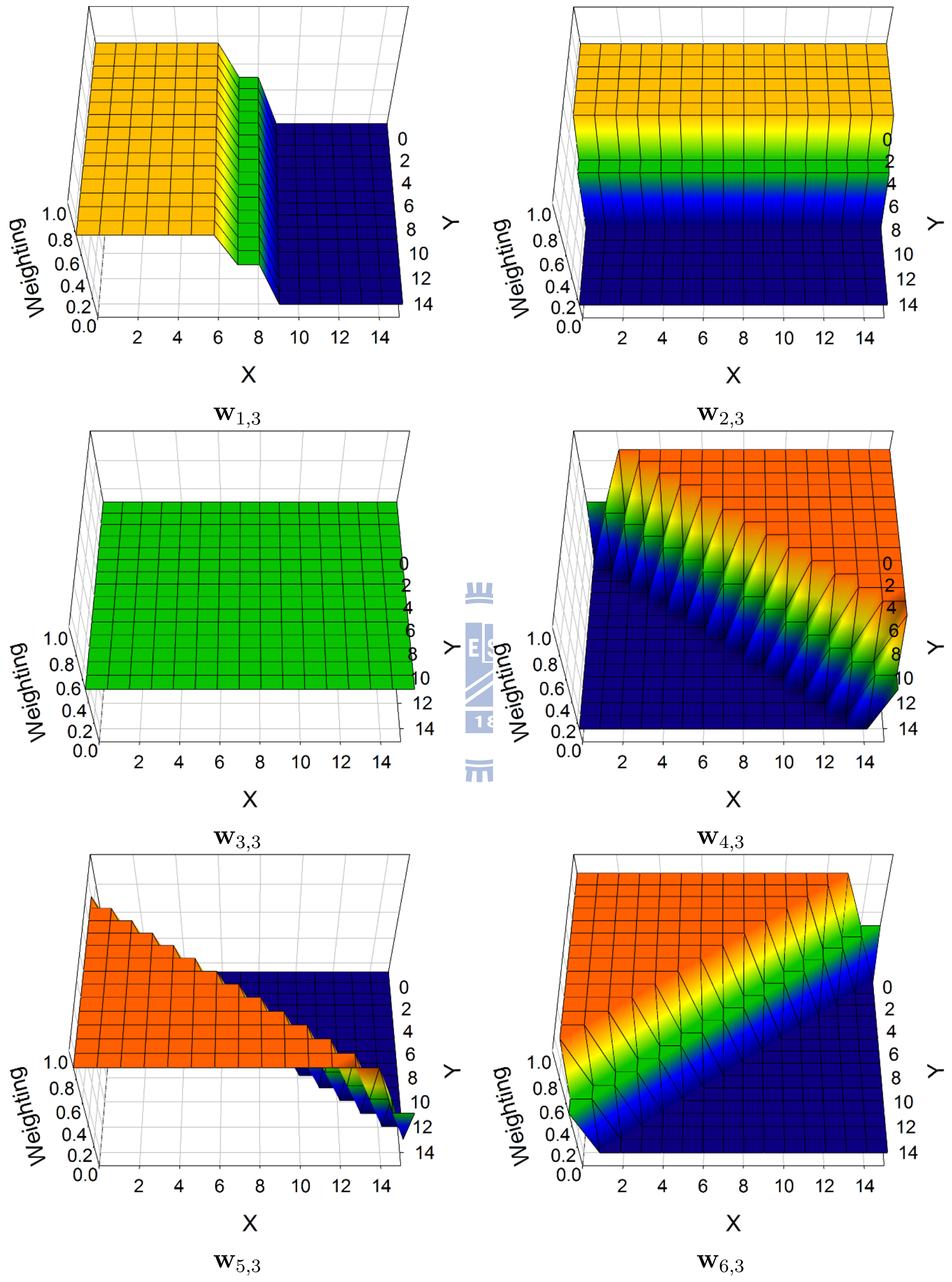


Figure 4.9: Simplified window functions set 2 for different merge candidates.

4.5 Experiments

Based on the proposed scheme (referred hereafter to as TB-mode), we develop three algorithms featuring different performance and complexity trade-offs. Extensive experiments are carried out using the HEVC reference software (HM3.0) and the common test conditions [5] to compare their BD-rate savings relative to anchor encodings with TB-mode disabled. Rough estimates of their complexity characteristics are made by showing the encoding and decoding time ratios relative to anchor settings. All three algorithms use parametric window functions. The following summarizes these algorithms:

- **Algorithm #1** applies TB-mode to 2N×2N prediction units (PUs)[2]. The $\mathbf{v}_t$ is found by performing shape-adaptive template matching in a search range of MVp±8 pixels. For each 2N×2N PU, one flag is sent to switch adaptively between TB-mode and the usual inter mode. When the former is chosen, it codes two extra bits (at most) to specify the template shape (InvL, Rect-T or Rect-L). A separate set of window functions are designed for each distinct 2N×2N PU size.

- **Algorithm #2** TB-mode with motion merging by signaling $\mathbf{v}_t$ with motion merging mechanism (See Section 4.3).

- **Algorithm #3** simplifies Algo. #2 using two sets of simplified window functions. (See Section 4.4).

Table 4.1 presents the average BD-rate savings of these algorithms in different test classes and configurations. As can be seen, Algo. #1 has an average BD-rate

[2]CU, the basic compression unit as the MB in AVC, can have various sizes but is restricted to be a square shape. PUs are the various MB partitions having a square or rectangular shape with several sizes.

(Y) saving of 1.9% over all test cases, with a minimum of 1.2% and a maximum of 3.4%. Due to the extra computations needed for template matching, it doubles the decoding time while increasing the encoding time by about 70%.

Algo. #2 gives similar coding performance at a faster decoding speed than Algo. #1. Averagely, it perform an average BD-rate (Y) saving of 1.7% over all test cases, with a minimum of 0.9% and a maximum of 3.2%. with an average decoding time only 4% slightly longer than anchors'. In tests with High Efficiency configurations, its decoder can run even a bit faster. As expected, without having to perform template matching, its decoding complexity drops significantly. But. somehow surprisingly, the $\mathbf{v}_t$ signaled by motion merging seems to outperform that inferred by template matching. The reason may be twofold. First, the motion merging signaling mechanism incurs less overhead: while it needs only one extra bit to signal $\mathbf{v}_t$, Algo. #1 requires, on average, more than one bit to indicate the template shape. Second, template matching may result in poor $\mathbf{v}_t$ due to coding noise. Recall that its search criterion is to minimize the error over the reconstructed pixels. Nevertheless, we believe Algo. #1 has plenty of room for further improvement.

Algo. #3 is the one with lowest computational complexity with an average BD-rate saving of 1.6% and a maximum saving up to 2.9%. This is attributed to the simplified window functions. But because this scheme utilizes PU-adaptive window selection from two sets of window functions, the encoder has to perform, for each PU, one extra motion search to evaluate TB-mode, which accounts for the slightly increased encoding time. But as for the decoder, the complexity is relative low due to a simple calculatioin of the predictor values.

Table 4.1: BD-rate savings and processing time ratios

Random Access	**High Efficiency**			**Low Complexity**		
Algo.	#1	#2	#3	#1	#2	#3
Class B	-1.2	-0.9	-1.0	-1.4	-1.0	-0.9
Class C	-2.0	-1.5	-1.6	-1.7	-1.2	-1.3
Class D	-1.9	-1.6	-1.8	-1.6	-1.2	-1.4
All	-1.7	-1.4	-1.2	-1.0	-1.1	-1.2
Enc. Time [%]	176	172	184	175	172	183
Dec. Time [%]	172	168	112	194	176	113
Low Delay	**High Efficiency**			**Low Complexity**		
Algo.	#1	#2	#3	#1	#2	#3
Class B	-1.9	-1.5	-1.3	-2.4	-2.0	-2.0
Class C	-2.3	-1.8	-1.7	-2.4	-1.9	-2.0
Class D	-2.1	-2.0	-1.9	-2.2	-1.9	-2.0
Class E	-3.3	-2.9	-1.8	-3.4	-3.2	-2.9
All	-2.4	-2.1	-1.7	-2.6	-2.3	-2.2
Enc. Time [%]	157	150	162	155	149	159
Dec. Time [%]	220	155	109	269	155	110

4.6 Conclusion

Summarizing, in this chapter, we propose a bi-prediction scheme, which combines MVs found by template and block matchings with an optimized OBMC window function. Since the template MV is inferred on the decoder side, it has only to signal one block MV. This notion is further generalized by incorporating adaptive motion merging to allow a template-matching-free implementation. Three algorithms featuring different performance and complexity trade-offs are presented; they all show moderate coding gains. The best of them produces an effect similar to performing partial motion merging for geometry/asymmetric motion partitions.

Chapter 5

Conslusion

In this dissertation, we have proposed an analytical perspective of MCP and developed a generic scheme to reconstruct predictor from any irregular motion sampling structure. Proceeding in the generic reconstruction scheme, a bi-prediction combining TMP and BMC is introduced to further improve prediction efficiency. Most of techniques proposed in this dissertation are evaluated and cross-checked in CEs of the JCV-VC committee, showing very promising results. In the followings, we summarize the works in this dissertation and show how the proposed schemes can be further improved.

5.1 Motion-Compensated Prediction: An Analytical Perspective

We found it convenient and insightful to interpret MCP as a motion sampling and reconstruction process. Such an interpretation reveals many important aspects of MCP which would otherwise be difficult to see. Using this notion, we provide a theoretical support for various MCP schemes from the sampling perspective. It is shown that the motion estimate found by template matching tends to be the motion associated with the template centroid and that TMP consistently outperforms SKIP prediction, but hardly competes with block motion compensation

(BMC) unless both the motion and intensity fields are less random or have high spatial correlation.

5.2 Parametric OBMC

We have proposed a parametric window design that enables OBMC to make use of variable blocksize motion estimates. In the common test conditions, our POBMC delivers better rate-distortion performance than the previous approaches such as H.263 OBMC. Relative to an H.264/AVC anchor with extended macroblock (MB) size, it achieves 3.1% (0.7-13.6%) BD-rate reductions. With its promising results and compatibility with existing tool features, POBMC is being evaluated in several formal core experiments in HEVC standardization activities at the time of writing.

5.3 Bi-PredictionUsing Template and Block MVs

A framework describing the joint prediction of explicitly transmitted and implicitly inferred MVs is proposed with an example showing the joint prediction of template matching prediction (TMP) and block based motion compensation (BMC). By considering the contribution of decoder-side inferred MVs to the prediction of the target block, the optimum sampling locations of explicit MVs are derived and then the join prediction of implicit and explicit MVs can be optimized using corresponding weighting coefficients derived by POBMC. Moreover, the MVs associated with each optimum sampling location are estimated by a modified search criterion to optimize its prediction quality. Relative to the anchor (HEVC reference software, HM3.0), the proposed bi-prediction achieves an average BD-rate saving of 1.6%, with a minimum of 0.9% and a maximum of 2.9%.

5.4 Future Work

It is a general feeling that video coding efficiency has been pushed to the limit; however, when looking deeper into the underlying principles behind the designs, there seems to be room for further improvements. We shall apply the methodology introduced in this dissertation to expose more undisclosed details of MCP to further improve video coding performance.

We plan to extend the works of this dissertation in several directions: (1) to study the quantization effect of POBMC coefficients, (2) to go beyond conventional motion partitioning to develop more efficient sampling patterns for POBMC, and finally, (3) to combine POBMC with advanced inter prediction techniques in the state-of-the-art video codec usch as HEVC, (4) to find the best solution of the proposed bi-prediction scheme with the optimal trade-off coding between coding efficiency and complexity such as encoding/decoding time and memory access bandwidth.

Bibliography

[1] "Video coding for low bitrate communication," *ITU-T, Recommendation H.263*, Apr. 1995.

[2] "Joint call for proporsals on video compression technology," *ISO/IEC JTC1/SC29/WG11, Doc. N11113*, Jan. 2010.

[3] V. Baroncini, J.-R. Ohm, and G. J. Sullivan, "Report of subjective test results of responses to the joint call for proposals (cfp) on video coding technology for high efficiency video coding (hevc)," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-A204*, April 2010.

[4] G. Bjontegaard, "Improvements of the bd-psnr model," *ITU-T SG16, Doc. VCEG-AI11*, Jul. 2008.

[5] F. Bossen, "Common test conditions and software reference configurations," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-C500*, Oct. 2010.

[6] C.-C. Chen, Y.-Y. Chen, C.-L. Lee, W.-H. Peng, and H.-M. Hang, "Ce2: Report of obmc with motion merging," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-F049*, July 2011.

[7] C.-C. Chen, C.-L. Lee, W.-H. Peng, and H.-M. Hang, "Ce1: Report of dmvd-based bi-prediction," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-E154*, Mar. 2011.

[8] Y.-W. Chen, T.-W. Wang, C.-L. Lee, C.-H. Wu, W.-H. Peng, and et. al., "Description of video coding technology proposal by nctu," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-A123*, Apr. 2010.

[9] Y. W. Chen, T. W. Wang, Y. C. Tseng, W. H. Peng, and S. Y. Lee, "A parametric window design for obmc with variable block size motion estimates," *Proc. IEEE Int. Workshop Multimedia Signal Processing*, 2009.

[10] Y.-W. Chen, C.-H. Wu, C.-L. Lee, T.-W. Wang, and W.-H. Peng, "Mb mode with joint application of template and block motion compensations," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-B072*, July 2010.

[11] B.-D. Choi, J.-W. Han, and S.-J. Ko, "Irregular-grid-overlapped block motion compensation and its practical application," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 8, pp. 1221–1226, 2009.

[12] T. Chujoh, N. Wada, and G. Yasuda, "Quadtree-based adaptive loop filter," *ITU-T SG16, Doc. JVT-C181*, Jan. 2009.

[13] S. Kamp, M. Evertz, and M. Wien, "Decoder side motion vector derivation for inter frame video coding," *Proc. Int. Conf. Image Processing*, 2008.

[14] M. Karczewicz and et al., "Video coding technology proposal by qualcomm inc." *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-A121*, Apr. 2010.

[15] C.-L. Lee, C.-C. Chen, Y.-W. Chen, W.-H. Peng, and et. al., "Bi-prediction combining template and block motion compensations," submitted to IEEE Int. Conf. Image Processing 2011, http://mapl.nctu.edu.tw/ewchen/ICIP2011PPLUS.pdf.

[16] C.-L. Lee, C.-C. Chen, Y.-W. Chen, M.-H. Wu, C.-H. Wu, W.-H. Peng, and H.-M. Hang, "Bi-prediction combining template and block motion compensations," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-D175*, Jan. 2011.

[17] C.-L. Lee, C.-C. Chen, W.-H. Peng, and et. al., "Bi-prediction combining template and block motion compensations," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-D175*, january 2011.

[18] K. McCann and et al., "Samsung's response to the call for proposals on video compression technology," *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-A124*, April 2010.

[19] M. T. Orchard and G. J. Sullivan, “Overlapped block motion compensation: An estimation-theoretic approach,” *IEEE Trans. on Image Processing*, vol. 3, no. 5, pp. 693–699, Sep. 1994.

[20] K. Software, http://iphome.hhi.de/suehring/tml/download/KTA/.

[21] G. J. Sullivan and R. L. Baker, “Motion compensation for video compression using control grid interpolation,” *Proc. ICASSP*, vol. 4, pp. 2713–2716, Apr. 1991.

[22] G. J. Sullivan and M. T. Orchard, “Methods of reduced-complexity overlapped block motion compensation,” *Proc. IEEE Int. Conf. Image Processing*, vol. 2, pp. 957–961, Nov. 1994.

[23] T.-K. Tan, G. J. Sullivan, and J.-R. Ohm, “Summary of hevc working draft 1 and hevc test model (hm),” *ITU-T SG16 WP3 Q.6 and ISO/IEC JTC1/SC29/WG11, JCTVC-C405*, Oct. 2010.

[24] B. Tao and M. T. Orchard, “A parametric solution for optimal overlapped block motion compensation,” *IEEE Trans. on Image Processing*, vol. 10, no. 3, pp. 341–350, Mar. 2001.

[25] Y.-C. Tseng, C.-H. Wu, Y.-W. Chen, T.-W. Wang, and W.-H. Peng, “On the analysis and design of motion sampling structure for advanced motion-compensated prediction,” *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 949–952, Sep. 2010.

[26] T.-W. Wang, Y.-W. Chen, and W.-H. Peng, “Analysis of template matching prediction and its application to parametric overlapped block motion compensation,” *Proc. IEEE Int. Symp. on Circuits and Syst.*, pp. 1563–1566, Jun. 2010.

[27] Z. Wang, W. Wang, Y. Lu, H. Cui, and K. Tang, “Coding mode adapted overlapped block motion compensation in h.264,,” *Proc. IMACS Multiconference on Computational Engineering in Systems Applications,*, vol. 1, pp. 1665–1668, Oct. 2006.

[28] M. Winken and et al., “Description of video coding technology proposal by fraunhofer hhi,” *ISO/IEC JTC1/SC29/WG11, MPEG2010/ JVTVC-A116*, Apr. 2010.

[29] Y. Ye and M.Karczewicz, "Enhanced adaptive interpolation filter," *ITU-T SG16, Doc. T05-SG16-C-0464*, Apr. 2008.

[30] W. Zheng, Y. Shishikui, M. Naemura, Y. Kanatsugu, and S. Itoh, "Analysis of space-dependent characteristics of motion-compensated frame differences based on a statistical motion distribution model," *IEEE Trans. on Image Processing*, vol. 11, no. 4, pp. 377–386, Apr. 2002.