FOCUS MISMATCH COMPENSATION AND COMPLEXITY REDUCTION TECHNIQUES FOR MULTIVIEW VIDEO CODING

by

PoLin Lai

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

May 2009

Dedication

To my family.

Acknowledgements

I would like to thank my advisor Prof. Antonio Ortega who, through the years, shapes my idea of what research is all about. I would like to thank Prof. C.-C. Jay Kuo and Prof. Ulrich Neumann for being the members in my dissertation committee, and Prof. Alexander A. Sawchuk and Prof. Ramakant Nevatia for being the members in my qualifying exam committee. It is a privilege to have their valuable advices on my work.

I would like to thank Yeping Su, Peng Yin, Purvin Pandit, Dong Tian and Cristina Gomila from Thomson Corporate Research, who provide tremendous support in all these years. I would also like to thank Congxia Dai and Yunfei Zheng for fruitful discussions during my summer interns.

For all the colleagues I met in our Compression Group, it is a great pleasure to work with you. Especially, I would like to thank Jae Hoon Kim for unforgettable years of collaborations and for all the sharing. I would also like to thank Ivy Tseng, May-Chen Kuo, C.K. Chen, Helen Ho from USC, and Julan Hsu from UCLA, for the wonderful friendship that strengthens my will.

Mom, Dad, and my dear brother, thank you for the love, the belief, the encouragement, which make me a person with strong faith in life, to stand, to enjoy, and to embrace.

Table of Contents

List Of Tables

List Of Figures

Abstract

Multiview video systems utilize multiple cameras to simultaneously capture the scene from different viewpoints. They provide video data for new applications such as 3D television and free-viewpoint video. The amount of data in multiview video is very large as compared to monoscopic video. Multiview video coding (MVC) is an emerging research field that focuses on compression of multiview video data. In this dissertation, by exploiting special characteristics of multiview video, we develop techniques that improve MVC efficiency while also taking complexity into account.

First, we analyze focus mismatch exhibited in video content, which is caused by camera focus setting differences. We use geometrical optics to demonstrate how focus settings will affect the captured images. We show that the focus mismatch can be represented in terms of the focus setting parameters (camera-dependency) and the depths of objects (depth-dependency). For 1D parallel camera arrangements in multiview systems, we relate the focus mismatch to the disparity exhibited in frames from different views. The analytical results provide properties that can be exploited to design focus mismatch compensation techniques.

Based on this analysis, we propose a novel adaptive reference filtering (ARF) approach. For MVC inter-view prediction, we exploit the depth-dependency property by

utilizing disparity information to partition frames into depth levels, which are prone to suffer from different types of focus mismatch. For each level, a 2D filter is designed by minimizing the prediction error. Filtered references are then generated for predictive coding. We also extend ARF to monoscopic video where no disparity information is available. Simulation results demonstrate higher coding efficiency as compared to multiple-reference prediction and adaptive interpolation filtering methods.

The third part of this thesis presents complexity reduction techniques for MVC. By analyzing ARF results, we propose i) View-wise ARF adaptation based on RD-cost prediction, and ii) Filter updating based on depth-composition change, to achieve computationally efficient ARF schemes. By exploiting the relationship between motion and disparity, we propose predictive fast search algorithms that can be used when one of the fields is available and we wish to estimate the other field efficiently. Simulation results show that significant complexity reduction can be achieved without significant impact on coding efficiency.

Chapter 1

Introduction

1.1 Multiview Video

In multiview video systems, scenes are captured simultaneously by multiple cameras. These cameras are set to shoot the scenes from different viewpoints. Depending on the application scenario, multiview systems can be built with cameras arranged on a horizontal line with parallel orientation, or on an arc with viewing angles converging to the center of the scene, or even with cameras mounted as a two-dimensional camera array. These systems provide digital video data that is essential for two main application categories: 3D television (3DTV) and free-viewpoint video (FVV) [40].

3DTV aims to provide 3D perception experiences using advanced display technologies which take multiview video as input. A general approach to achieve this goal is to simultaneously render multiple video from different viewing angles, such that when the user is viewing the display device from different physical locations, different stereoscopic views are perceived. Examples of such display systems include the one proposed by Mitsubishi Electric Research Laboratories (MERL) which uses multiple projectors [33],

and a special TV system designed by Philips Electronics [9] which processes multiview video plus depth data to render video for different viewing angles. The main applications for 3DTV are live-event broadcasting, advanced theater, and immersive virtual reality.

As for FVV applications, at the receiver side, users can navigate across different viewpoints by changing the viewing angle to be displayed on a conventional screen. Assume N views are captured by the multiview system. If the requested viewing angle coincides with one of the captured views, the corresponding video can be displayed directly. On the other hand, if the requested viewing angle differs from any of the captured views, a view synthesis process has to be applied, which takes $n \leq N$ adjacent views to render and display that particular viewpoint. Applications of such FVV systems include education, such as archives and medical surgery; entertainment, such as sport broadcasting and gaming systems; and security, such as surveillance and monitoring tasks.

To provide the content for these applications, multiview video has to be captured, stored, and transmitted. It contains very large amounts of data as compared to conventional monoscopic video. If each view is considered independently, the amount of added data will increase with the number of views in multiview systems. Multiview video coding (MVC), which focuses on compression for efficient storage and transmission of multiview video data, has been recognized as a key technology to support any applications that require multiview video. In the next subsection, we will briefly review the recent development of MVC, and introduce terminologies that describe the coding schemes that will be used throughout this dissertation.

1.2 Multiview Video Coding (MVC)

Simulcast is a straightforward approach for multiview video coding, in which each view sequence is encoded independently (see Fig. 1.1). This allows temporal redundancy to be exploited using conventional block-based motion compensated prediction (MCP) techniques. In a multiview video scenario, because different views are capturing the same scene, there exists an additional source of redundancy, namely, inter-view redundancy. Similar to motion compensated prediction, we can use a block matching procedure to find block correspondences between neighboring views and encode the displacement vector and prediction difference (residue), through disparity compensated prediction (DCP). This inter-view redundancy is not exploited in the straightforward simulcast scheme. A MVC structure that exploits both temporal and inter-view redundancy can be constructed as follows: A given frame in view v at time t, can use reconstructed frames within view v as temporal reference(s) for MCP, while using reconstructed frames from other views as inter-view reference(s) for DCP. We will denote such coding method as *joint MCP/DCP*. In the MPEG-2 standard, a multiview profile (MVP) [8] was defined which utilizes a two-layer coding scheme (base layer and enhancement layer). View sequences in the enhancement layer can use either MCP, DCP, or joint MCP/DCP. It has been demonstrated that allowing joint MCP/DCP in MPEG-2 MVP achieves higher coding efficiency as compared to simulcast [8]. This coding scheme can be extended to MPEG-4, with its temporal scalability tool [56]. The coding efficiency is further improved by introducing *weighted average* combined with *block partition* for joint MCP/DCP [11].

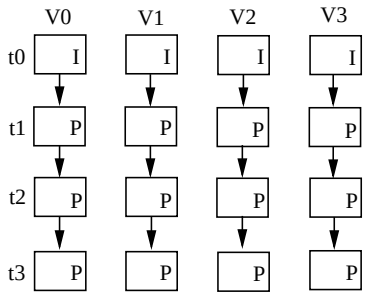

Figure 1.1: Simulcast for multiview video coding

With coding tools such as variable block size MCP, quarter pixel motion estimation, and new content adaptive entropy coding, the H.264/AVC standard [54] established by the joint video team (JVT), achieves superior coding efficiency as compared to the preceding video coding standards. For multiview video coding, Li and He [30] proposed to use the multiple reference prediction tool in H.264/AVC to facilitate temporal and inter-view references, thus supporting joint MCP/DCP. This became a very popular technique which has been used in recent literature to construct various MVC structures based on H.264/AVC [3, 12, 24, 32, 34], including our work in [24], confirming that utilizing joint MCP/DCP leads to higher coding efficiency than simulcast. Furthermore, the study in [13] and [37] show that, for most cases, the joint MCP/DCP method which utilizes inter-view references from the *immediately neighboring views at same timestamp*, can achieve comparable coding efficiency to other schemes that use additional farther away inter-view references. In 2005, recognizing the importance of MVC for future applications, the MPEG committee issued a Call for Proposals on Multi-view Video Coding [18], aiming to establish a standard for MVC. Conforming to recent research trends, all the received MVC proposals were based on the H.264/AVC framework [19]. While differing in the

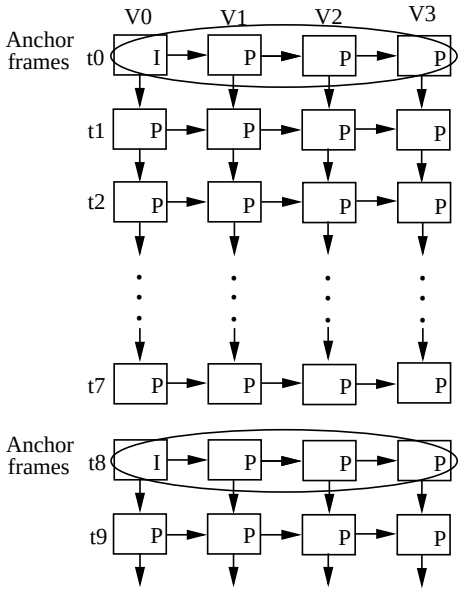

Figure 1.2: An example of MVC structure which utilizes joint MCP/DCP. The arrows indicate prediction direction.

prediction structures, they all feature joint MCP/DCP in order to exploit temporal and inter-view redundancy.

Fig. 1.2 illustrates an MVC coding structure with 32 frames forming one group of pictures (GOP). This GOP contains 4 views in spatial direction and 8 frames in temporal direction. To facilitate random access, within one GOP the first frame of each view is encoded with only inter-view DCP, i.e., no temporal references are needed. We denote these frames *anchor frames*. Similar to I frames in monoscopic video, these anchor frames serve as temporal access points for multiview video. The remaining frames in a GOP, which we will denote as non-anchor frames, can use both temporal and inter-view references to exploit the redundancy in two directions. For example in Fig. 1.2, a block in frame ($V2$,$t1$) can switch between the two corresponding MCP/DCP blocks from frame ($V2$,$t0$) and frame ($V1$,$t1$) respectively. Note that for simplicity, this diagram only

includes frames coded as I-pictures and P-pictures. However the joint MCP/DCP concept can be easily extended to B-pictures with both temporal and inter-view references available for prediction.

As compared to MCP, the main difficulty in DCP is that, due to differences in camera settings and shooting positions/orientations, frames from different views are more likely to exhibit non-translational discrepancy, such as illumination / color mismatch and focus mismatch. To compensate for these mismatches, advanced coding tools such as illumination compensation [31] and our adaptive reference filtering method [23, 25–28] (which will be explained in detail in Chapters 3 and 4) have been developed. Significant coding gain can be achieved when applying them to inter-view prediction in MVC.

1.3 Contributions of This Research

In this research, we investigate the special characteristics exhibited in multiview video data, and develop coding techniques that improve MVC coding efficiency while taking into account the complexity. In particular, coding techniques are designed by exploiting multiview video properties. The main contributions of this thesis can be summarized as the following:

- We consider the problem of focus mismatch. To understand its characteristics in video coding, we analyze focus mismatch based on geometrical optics. We show that focus differences between images can be represented in terms of the focus setting differences and the depths of objects. The focus mismatch kernels are circular symmetric with their shapes varying across different depths. For multiview systems

with a 1D parallel camera arrangement, we further demonstrate that the disparity exhibited in the images can be utilized to reliably identify different types of depth-dependent focus mismatches. Our analytical results provide insight that can be exploited to design coding techniques to compensate for focus mismatch in video content.

- We propose adaptive reference filtering (ARF) approaches to compensate for the depth-dependent focus mismatch. Based on our analysis, we model predictive coding with focus mismatch using point spread functions. Depending on the coding scenario (inter-view or temporal), we estimate different block-wise parameters as features for classification such that an image can be partitioned into regions suffering from different types of focus mismatch. Since focus mismatch is depth-dependent, for MVC inter-view coding, we exploit the block-wise disparity field to partition frames into regions corresponding to different depth levels. As for monoscopic video that undergoes camera focus changes across time, we propose to actually estimate the localized focus changes and partition frames into regions consisting of macroblocks that suffer from a similar type of focus change. In both methods, a 2D Wiener filter is designed for each region (class) to minimize the prediction error. Filtered references are generated for the encoder to perform rate-distortion (RD) optimized coding selection. For video sources containing strong localized focus mismatches, the proposed methods provide higher coding efficiency as compared to other techniques such as multiple-reference prediction and adaptive interpolation filtering.

- The ARF method is extended to inter-view bi-prediction (B-frames) in MVC, in which the two predictors from different views may exhibit different types of focus mismatch. We investigate the interaction between filter estimation and bi-directional search. We show that designing filters only for the averaged bi-predictor could lead to a suboptimal solution when combined with conventional bi-directional search schemes. Instead, we propose a filter design approach that estimates two sets of depth-related filters, each set compensating for the focus mismatch occurring in one of the two references used for bi-directional prediction.

- We analyze the encoding results when using ARF for inter-view prediction. The coding gains demonstrate a strong view dependency, while the estimated filters at different timestamps exhibit strong correlation when the objects' depths remain similar in the corresponding captured scene. These results conform with the analysis based on geometrical optics. We propose i) a rate-distortion prediction method to achieve view-wise ARF adaptation, such that ARF will be applied only to views in which substantial gain can be achieved, and ii) a filter updating mechanism based on depth-composition change, which allows the same set of filters to be used by several consecutive frames until there is significant change in the depth-composition within the scene. These two techniques lead to significantly reduced complexity while preserving coding efficiency.

- We propose fast predictive search algorithms for joint MCP/DCP, which exploit the relationship between motion and disparity fields. After one of the motion/disparity

fields is estimated, our method obtains good candidate vectors to perform the estimation on the other field with very low complexity. We construct a model and analytically demonstrate how mismatches, such as illumination change, will affect the accuracy of the first estimated displacement field (motion or disparity), which consequently will affect the reliability of the candidate vectors obtained with our fast predictive search methods. Analysis and simulations results both indicate that it is more efficient to perform motion estimation first and then apply our fast predictive search to disparity estimation, instead of the other way around.

The rest of this dissertation is organized as follows. First in Chapter 2, we analysis focus mismatch based on geometric optics and derive properties of focus mismatch kernels. Chapter 3 describes in detail the proposed adaptive reference filtering approach to compensate for focus mismatch in video content. Then, in Chapter 4, we study the performance of ARF and introduce methods to reduce ARF complexity while maintaining coding efficiency. In Chapter 5, for joint MCP/DCP, we propose predictive fast search algorithms with new candidate vectors. Finally, conclusions and possible future work are presented in Chapter 6.

Chapter 2

Focus Mismatch in Video Content

2.1 Introduction

Multiview video systems utilize multiple cameras to simultaneously capture scenes from different viewpoints. As compared to conventional monoscopic video, frames from different views are prone to suffer from mismatches other than simple displacement, due to differences in camera settings and/or shooting positions/orientations. These mismatches across different views could be obstacles to achieving high coding efficiency, as conventional block matching in video coding may not be effective to compensate for them.

In this dissertation, we consider one particular type of non-translational mismatch exhibited in video content: focus mismatch, which results in blurriness / sharpness discrepancy among frames from different views. With multiple cameras in MVC systems, focus mismatch is most likely to be caused by heterogeneous cameras settings. For example, camera parameters may be inconsistent, so that the focus settings may be different from view to view. This can cause localized focus mismatch, as objects may not always be in sharp focus across different views. Another source of focus mismatch in multiview

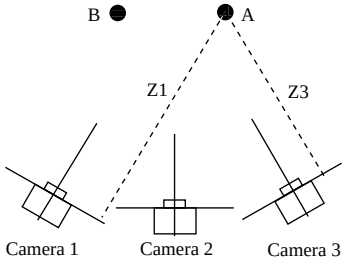

Figure 2.1: Camera arrangement that causes localized focus mismatch

systems could be the camera arrangement. Consider the example in Fig. 2.1, where object A appears at a greater depth (z_1) in View 1 than in View 3 (z_3). Assume all cameras are set with the same perfect-in-focus depth at z_1, then object A may appear in focus in View 1 while it will likely be out of focus (blurred) in View 3. On the other hand, object B will appear sharper in View 3 as compared to View 1. The efficiency of inter-view disparity compensation could deteriorate in the presence of these mismatches.

Among the multiview video test sequences provided in the initial MVC Call for Proposals document [18], the sequence *Race1* exhibits the most clearly perceivable focus discrepancy among frames from different views. It consists of 8 parallel views captured by cameras with a 20cm spacing between each, which we will denote as View 0 $\sim$ View 7. The frames in View 3 are blurred as compared to the frames in View 2; similarly, the frames in View 5 are blurred as compared to the frames in View 6. Fig. 2.2 shows portions of the frames from different views in *Race1*. It can be seen that, besides displacement of the scene, frames from different views also exhibit blurriness mismatch.

In addition to inter-view prediction in MVC, focus mismatch may also occur in monoscopic video. One often observed example of focus change across time happens in dialog

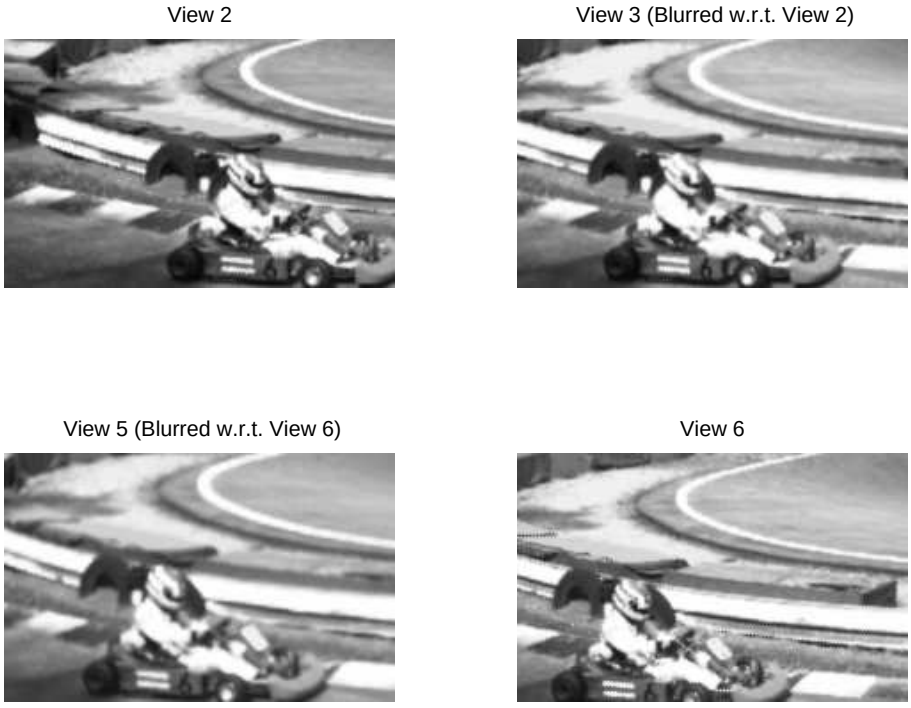

Figure 2.2: Portions of frame 15 from different views of *Race1*

scenes, in which the camera shifts its focus from one character to another one at a different scene depth. The first character becomes blurred (out of focus) while the second gets sharpened (in focus). In other occasions, focus changes are created for transition during scene changes. Fig. 2.3 demonstrates an example of focus change in monoscopic video. From the left image to the right one, we can see that the focused-depth is shifting from the back to the front. In particular, the second and third person (counting from the front) are becoming more "in-focus" while people in the back are getting blurred.

A focus mismatch compensation technique for video coding can be established by first estimating different focus mismatch kernels and then creating filtered versions of the reference frame in order to provide a better match to the current frame. Higher coding efficiency can be achieved by using these filtered references for prediction and transmitting

Figure 2.3: Example of focus change in monoscopic video. (Images from Stanford Computer Graphics Lab., Light Field Photography with a Hand-Held Plenoptic Camera, © Ren Ng)

the filter coefficients as side information, so that filtered reference frames can be generated at the decoder. However, without prior knowledge about focus mismatches, estimating the mismatch kernels can be very complicated as we will have to consider a very large solution space (in terms of the shape and support of the kernels). Thus in order to reliably and efficiently estimate focus mismatch kernels, it is necessary to understand how focus setting differences affect the focus mismatches. Furthermore, as described in the aforementioned example, objects at different depths could undergo different types of focus mismatches. Thus, we also need to investigate how focus mismatch changes at different depths.

In the literature, in order to model how images are captured by a camera, a simplified model is typically used to approximate a camera as an imaging system with a single lens. Geometrical optics is widely utilized to derive some well-known properties of the projected images under a fixed focus setting with the simplified single lens model [29, 36]. In this chapter, we will further extend these previous results to analyze how focus setting *differences* will affect the captured images. We will show that the mismatch exhibited in the images can be represented in terms of the focus setting parameters (camera-dependency)

and the depths of objects (depth-dependency). Then we consider multiple cameras which capture the scene from different viewpoints. For a 1D parallel camera arrangement in a multiview system, we relate the focus mismatch to the disparity exhibited in frames from different views. The analytical results provide a better understanding of the focus mismatch problem and can be exploited to design coding tools aiming to compensate for such mismatch.

The remainder of this chapter is organized as follows: In Section 2.2 we first review the characteristics of images captured with a lens. Then in Section 2.3, we will derive special properties of images under the influence focus setting differences. We will discuss a multiview system with 1D parallel camera arrangement, and also provide a numerical example to illustrate the analytical results. Finally, useful characteristics that can be exploited to design focus compensation tools are summarized in Section 2.4.

2.2 Review: Characteristics of Images Captured with Lens

A camera is typically modeled as an imaging system consisting of a film, a lens with focal length f, and an aperture with diameter a. For digital cameras, the film is made up with an array of image sensors. The plane which contains the film is called the *image plane*, which is parallel to the lens with distance d to it. In Fig. 2.4, we construct a coordinate system with its origin located at the center of the lens and its xy-plane parallel to the image plane. The z-axis, which passes through the lens center, is also called the *optical axis*. Note that the coordinate system in Fig. 2.4 uses left-handed orientation such that points in front of the camera will have positive depth values Z. The two points on the

optical axis with $|z| = f$ are called the *focal points*, which have special properties that will be discussed shortly.

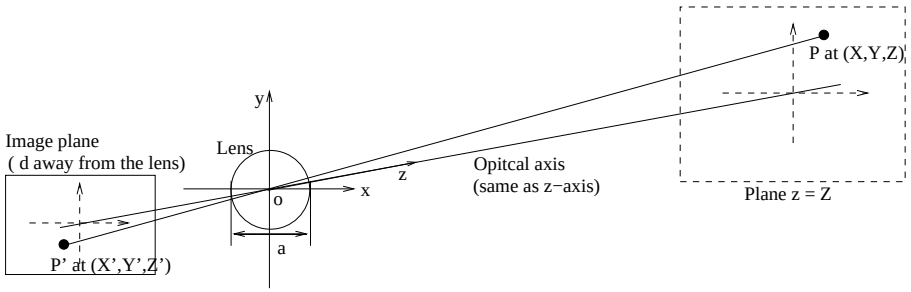

Figure 2.4: The model of a camera equipped with a lens

We analyze the effects of light via geometrical optics, which treats light as rays [29]. In Fig. 2.4, let us consider a point P at location (X, Y, Z) which is visible to the camera. Light rays passing through the lens center will not be refracted. Therefore, the light ray that originates from P, passes through O, will be projected on the image plane at point P' with coordinates (X', Y', X') such that (based on the congruence of triangles):

$$(X', Y', Z') = (-\frac{d}{Z}X, -\frac{d}{Z}Y, -d) \tag{2.1}$$

The minus signs in (2.1) represents the fact that the projected images on the image plane will be reversed and upside down (rotated 180 degrees), as compared to the original appearance in the scene. Similarly, the projections of other visible points produced by light rays passing through O, can also be determined by (2.1). For a given object, its projection on the image plane will become smaller as it moves away from the camera (increasing Z). The principle in (2.1), which describes the light rays passing through lens center, is called the perspective projection. It is widely used in geometric camera

models when the focusing effect of the lens is ignored [15]. However, in this work, it is our goal to analyze the effect of focus introduced by the lens. To take it into account, light passing through other parts of the lens has to be considered as well. According to geometrical optics, light rays parallel to the optical axis on one side of the lens will be refracted to pass through the *focal point* on the other side of the lens (see examples in Fig. 2.5). Furthermore, light rays originating from a point P with depth Z will *converge* to a point $\hat{P}$ on the other side of the lens with distance $\hat{Z}$ that satisfies:

$$\frac{1}{Z} + \frac{1}{\hat{Z}} = \frac{1}{f} \tag{2.2}$$

Fig. 2.5(a) depicts three points P, P_1, P_2 at different depths, the projections P', P_1', P_2' of light rays from these points passing through the lens center, and their converged image points $\hat{P}, \hat{P}_1, \hat{P}_2$. The dashed light paths are determined after finding the converging points based on the light paths depicted with solid lines. As a result, on the image plane with distance d to the lens, a visible point will produce a point projection (perfectly focused) only if it is at a particular depth Z^* that satisfies:

$$\frac{1}{Z^*} + \frac{1}{d} = \frac{1}{f} \;\Rightarrow\; Z^* = \frac{d \cdot f}{d - f} \tag{2.3}$$

When capturing the scene, under a fixed zoom parameter set by f, we can focus on a specific distance Z^* by fine tuning d ($d \geq f$). Operating in a very narrow range, a slight change in d can cause relatively large variation in Z^* (Refer to Fig. 2.5(b)). This can be achieved by using auto-focus (AF) or by manually adjusting the focus ring. For points at distances other than Z^*, their projections on the image plane will be uniform circles

with diameter β, as depicted in Fig. 2.5(a). Using again the congruence of triangles, β can be calculated as:

Depth smaller than Z^* (Fig. 2.5(a) P_1):

$$\begin{aligned} \frac{\beta}{a} &= \frac{\hat{Z}_1 - d}{\hat{Z}_1} \\ \frac{\beta}{a} &= \frac{\left(\frac{Z_1 f}{Z_1 - f} - \frac{Z^* f}{Z^* - f}\right)}{\left(\frac{Z_1 f}{Z_1 - f}\right)} \\ \beta &= \frac{af\,(Z^* - Z_1)}{Z_1\,(Z^* - f)} \end{aligned} \tag{2.4}$$

Depth greater than Z^* (Fig. 2.5(a) P_2):

$$\begin{aligned} \frac{\beta}{a} &= \frac{d - \hat{Z}_2}{\hat{Z}_2} \\ \frac{\beta}{a} &= \frac{\left(\frac{Z^* f}{Z^* - f} - \frac{Z_2 f}{Z_2 - f}\right)}{\left(\frac{Z_2 f}{Z_2 - f}\right)} \\ \beta &= \frac{af\,(Z_2 - Z^*)}{Z_2\,(Z^* - f)} \end{aligned} \tag{2.5}$$

$$\text{Combining (2.4) and (2.5), we can get: } \beta = \frac{af|Z - Z^*|}{Z\,(Z^* - f)} \tag{2.6}$$

It can be seen from (2.3) and (2.6) that the value of β is determined by parameters a, d, f, and the object depth Z. To demonstrate how β varies with different scene depth Z, under the given focus setting, we define:

$$k = \frac{Z}{Z^*} \rightarrow Z = k \cdot Z^* \text{ , i.e. } k \text{ is the depth } \textbf{normalized} \text{ by } Z^* \tag{2.7}$$

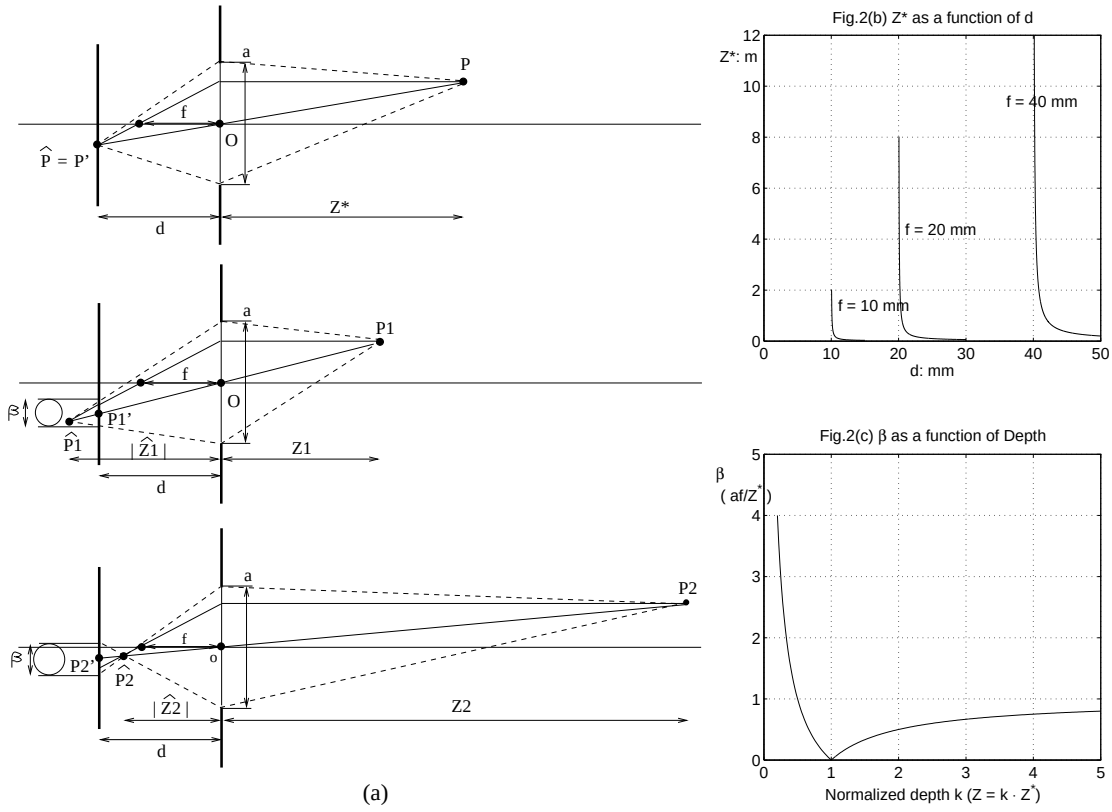

Figure 2.5: (a) Projected images via a lens for points at different depths, (b) Variations of Z^*, (c) Variations of β

Then from (2.6), β can be represented as:

$$\beta = \frac{af|Z - Z^*|}{Z\,(Z^* - f)} = \frac{af|k-1|Z^*}{kZ^*\,(Z^* - f)} = \frac{af|k-1|}{k\,(Z^* - f)} \approx \frac{af}{Z^*} \cdot \frac{|k-1|}{k} \tag{2.8}$$

The approximation in (2.8) is based on the fact that, typically, $f \ll Z^*$ (e.g., f can be less than 100mm while Z^* is of the order of several meters). Fig. 2.5(c) illustrates the variation of β as a function of the normalized depth. As the depth Z deviates from Z^* (thus $k \neq 1$), the diameter β increases, leading to stronger out-of-focus blurriness: Instead of forming a *point image*, on the image plane, the image intensity is *spread over*

a circle with diameter β (area $\pi(\beta/2)^2$). Therefore, for a point with a depth Z, the *point spread function*, PSF, is a circular disk:

$$PSF_Z(x,y) = \begin{cases} 4/(\pi\beta^2), & \text{if } x^2+y^2 \le (\beta/2)^2 \\ 0, & \text{otherwise} \end{cases} \text{, where } \beta = \frac{af|Z-Z^*|}{Z\,(Z^*-f)} \tag{2.9}$$

From Fig. 2.5(a), the center of the point spread function, produced by a visible point P at (X,Y,Z), will be located at coordinate (X',Y') on the image plane, as specified by (2.1). The intensity of the projection resulting from P, denoted as I_P, can then be described as:

$$I_P(x,y) = K_P \cdot PSF_Z(x-X', y-Y'), \tag{2.10}$$

where K_P represents the light intensity produced by P at converging point $\hat{P}$, i.e., the intensity if perfectly focused. On the image plane, this value is spread over a disk as described by (2.10). The total light intensity at (x,y) on the image plane, denoted as $J(x,y)$, is the *superposition of all the projection circles centered at different locations that contribute non-zero values at position* (x,y). In general, projections centered at nearby locations can have different diameters, as β depends on the depth of the visible point that produces the projection. However for typical scenes being captured, points within a small visible region will often have very similar depth Z, so that their corresponding projections can be well approximated by the *same* point spread function (corresponding to a fixed Z). Also, the converged light intensity K_P produced by points with similar depth Z, can be

approximated as a function of only X and Y, which can then be represented as a function of the projected locations (X', Y'): $K_P = K(X, Y, Z) \approx K_Z(X, Y) = K_Z(X', Y')$. The subscript indicates that this representation is valid for points that are at a given Z. The projected image of this region with approximately fixed Z can then be derived as:

$$\begin{aligned} J_Z(x, y) &= \int\int K_Z(X', Y') \cdot PSF_Z(x - X', y - Y')dX'dY' \\ &= K_Z(u, v) * PSF_Z(u, v), \text{ where } (u, v) \text{ are dummy variables} \end{aligned} \tag{2.11}$$

Since $K_Z(X', Y')$ represents the light intensity that would be observed under perfect focus, (2.11) indicates that for a region with depth Z, the effect of focus setting can be modeled as the *perfectly focused image convolved with the point spread function* PSF_Z.

2.3 Focus Mismatch Due to Focus Setting Differences

In Section 2.2, we reviewed imaging characteristics of a single camera equipped with a lens. We now discuss how to model the effect of capturing the same scene under different focus settings. Let us consider two cameras V1 and V2 in a multiview system. Assume they have the same focal length setting f (same zoom) and same aperture setting a [1]. However, their perfect in-focus depths Z^* are not equal, $Z^*_{V1} \neq Z^*_{V2}$. For example, if Z^* is determined by auto-focus, it will depend on the scene contents, which are not exactly the same for different cameras. From (2.6), this will result in differences in how β varies as a function of Z. Follow the derivation in (2.7), we define:

[1] For most cameras currently available in the market, both f and a can be set to a specific value from a finite/discrete values to choose from (e.g., f = 35mm, 70mm... and $a = f/2.7$, $f/5.6$...)

$$k = \frac{Z}{Z^*_{V2}} \quad (Z = k \cdot Z^*_{V2}) \tag{2.12}$$

$$c = \frac{Z^*_{V1}}{Z^*_{V2}} \quad (Z^*_{V1} = cZ^*_{V2}), \tag{2.13}$$

where k is the **normalized depth** using Z^*_{V2} as a reference, and c measures the **degree of focus setting mismatch**. For the two cameras, their respective β_{V1} and β_{V2} can be written as:

$$\beta_{V2} = \frac{af|Z - Z^*_{V2}|}{Z\,(Z^*_{V2} - f)} = \frac{af|k-1|Z^*_{V2}}{kZ^*_{V2}\,(Z^*_{V2} - f)} \approx \frac{af}{Z^*_{V2}} \cdot \frac{|k-1|}{k} = \frac{af}{Z^*_{V2}}|1 - \frac{1}{k}| \tag{2.14}$$

$$\beta_{V1} = \frac{af|Z - Z^*_{V1}|}{Z\,(Z^*_{V1} - f)} = \frac{af|k-c|Z^*_{V2}}{kZ^*_{V2}\,(cZ^*_{V2} - f)} \approx \frac{af}{Z^*_{V2}} \cdot \frac{|k-c|}{ck} = \frac{af}{Z^*_{V2}}|\frac{1}{c} - \frac{1}{k}| \tag{2.15}$$

where the approximations in (2.14) and (2.15) are based on the assumptions as for (2.8). Note that here we also represent β in terms of the reciprocal of the normalized depth, i.e., $\frac{1}{k}$. Fig. 2.6 below demonstrates how the two β's change as functions of depth k and its reciprocal $\frac{1}{k}$, in the case where $c = 0.7$ ($Z^*_{V1} = 0.7Z^*_{V2}$).

From Fig. 2.6, we can see that the two β curves change rapidly when k is small while they vary more slowly as k increases. On the other hand, from (2.14) and (2.15), the value of β is approximately a *linear function* of $\frac{1}{k}$. The intersection of the two β curves, which occurs at $\frac{1}{k} = \frac{1}{2}(1 + \frac{1}{c})$ (see Fig. 2.6), divides the depth range into two regions: $\beta_{V1} > \beta_{V2}$ and $\beta_{V1} < \beta_{V2}$. An object at given depth Z (i.e., at a given k or $\frac{1}{k}$) will appear differently in V1 and V2, under the influence of different β. From (2.9) and (2.11),

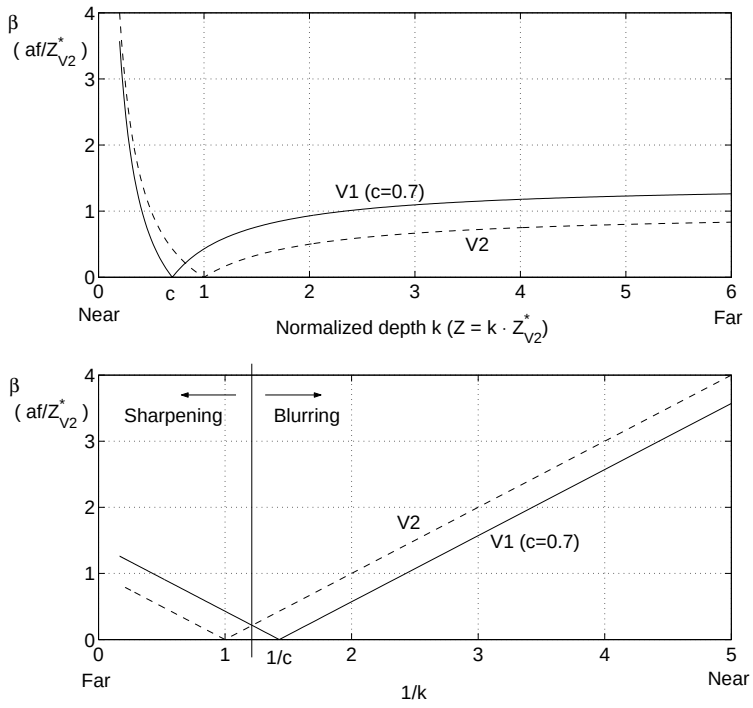

Figure 2.6: β as functions of k and $1/k$ ($Z^*_{V1} = 0.7Z^*_{V2}$, $c = 0.7$)

a larger β indicates that the light is spread over a larger area, resulting in stronger out-of-focus blurriness. If we want to match an image from V1 to the corresponding image from V2 at the same timestamp, objects with depths in the first region require sharpening filters as they are more blurred in V1 than in V2, while objects at depths in the second region need lowpass filtering. To better illustrate how such difference in β will affect the images, we plot the optical transfer function, OTF, which is the the frequency transform of PSF_Z. That is, $Fr\{PSF_Z(x, y)\} = OTF(v_x, v_y)$, where $Fr\{\cdot\}$ denotes the transform from spatial to frequency domain. The following transform pair can be derived [5] by applying the Hankel transform to obtain a frequency domain representation in the polar coordinate system:

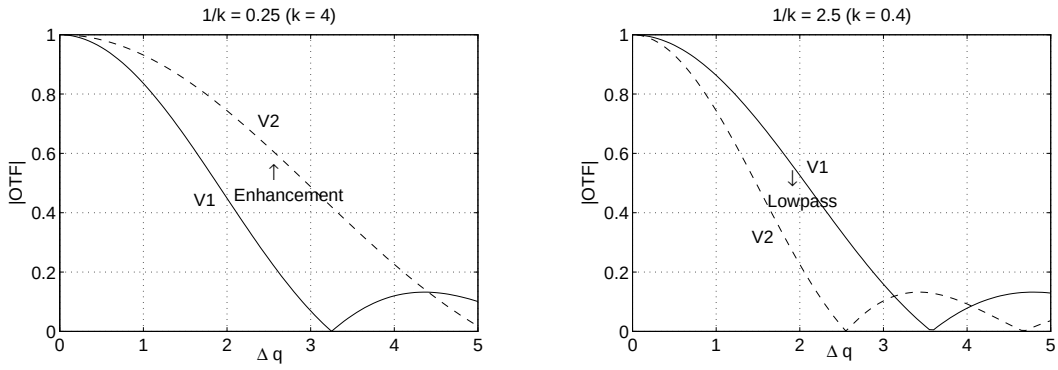

Figure 2.7: The magnitude of OTF at different k ($Z^*_{V1} = 0.7Z^*_{V2}$, i.e. $c = 0.7$)

$$PSF_Z(r) = \begin{cases} 4/(\pi\beta^2), & \text{if } r^2 \leq (\beta/2)^2 \\ 0, & \text{otherwise} \end{cases} \quad \rightarrow \quad OTF_Z(q) = \frac{2J_1(\pi\beta q)}{\pi\beta q} \tag{2.16}$$

In (2.16), $r = \sqrt{(x^2 + y^2)}$, $q = \sqrt{(v_x^2 + v_y^2)}$, with v_x and v_y representing the horizontal and vertical frequencies, and J_1 is the Bessel function of the first kind of order 1.

By plugging (2.14) and (2.15) into (2.16), and defining $\Delta = \frac{\pi a f}{Z^*_{V2}}$, we get:

$$OTF_Z^{V2}(q) = \frac{2J_1(|1 - \frac{1}{k}|\Delta q)}{|1 - \frac{1}{k}|\Delta q} \tag{2.17}$$

$$OTF_Z^{V1}(q) = \frac{2J_1(|\frac{1}{c} - \frac{1}{k}|\Delta q)}{|\frac{1}{c} - \frac{1}{k}|\Delta q} \tag{2.18}$$

Fig. 2.7 illustrates examples of the magnitude of OTF with $\frac{1}{k}$ values from the two regions in Fig. 2.6. It can be seem clearly that in the first region a sharpening filter is required to match OTF^{V1} to OTF^{V2}, while in the second region we need a blurring filter.

For simplicity, let us denote $k' = \frac{1}{k}$. For a given k' (a given depth Z), to match the OTF from V1 to V2, **the frequency response H_Z of the filter required** can be represented as:

$$H_Z(q) = \frac{OTF_Z^{V2}(q)}{OTF_Z^{V1}(q)} = \frac{\beta_{V1}}{\beta_{V2}} \frac{J_1(\pi\beta_{V2}q)}{J_1(\pi\beta_{V1}q)} = \frac{|\frac{1}{c} - k'|}{|1 - k'|} \frac{J_1(|1 - k'|\Delta q)}{J_1(|\frac{1}{c} - k'|\Delta q)} \tag{2.19}$$

From Fig. 2.6, it can be observed that the relationship between the two β curves is symmetric with respect to $k' = \frac{1}{2}(1 + \frac{1}{c})$, if we exchange the roles of β_{V1} and β_{V2} when crossing $\frac{1}{2}(1 + \frac{1}{c})$. This leads to the following interesting property: *the filter responses are reciprocals on the two sides of* $\frac{1}{2}(1 + \frac{1}{c})$. In the following, we will derive such result. Let us define two points k'_+ and k'_- with the same distance κ to $\frac{1}{2}(1 + \frac{1}{c})$, but are on the opposite sides, i.e. $k'_+ = \frac{1}{2}(1 + \frac{1}{c}) + \kappa$ and $k'_- = \frac{1}{2}(1 + \frac{1}{c}) - \kappa$ ($\kappa > 0$):

$$\begin{aligned}
\text{For } k'_+\text{:} \quad \left|\frac{1}{c} - k'_+\right| &= \left|\frac{1}{c} - \frac{1}{2}(1 + \frac{1}{c}) - \kappa\right| = \left|\frac{1}{2c} - \frac{1}{2} - \kappa\right| \\
\left|1 - k'_+\right| &= \left|1 - \frac{1}{2}(1 + \frac{1}{c}) - \kappa\right| = \left|-(\frac{1}{2c} - \frac{1}{2}) - \kappa\right| \\
H_{k'_+}(q) &= \frac{\left|\frac{1}{2c} - \frac{1}{2} - \kappa\right|}{\left|-(\frac{1}{2c} - \frac{1}{2}) - \kappa\right|} \frac{J_1\left(\left|-(\frac{1}{2c} - \frac{1}{2}) - \kappa\right| \Delta q\right)}{J_1\left(\left|\frac{1}{2c} - \frac{1}{2} - \kappa\right| \Delta q\right)} \\
&= \frac{|\alpha - \kappa|}{|-\alpha - \kappa|} \frac{J_1\left(|-\alpha - \kappa| \Delta q\right)}{J_1\left(|\alpha - \kappa| \Delta q\right)} \quad \text{where } \alpha = \frac{1}{2c} - \frac{1}{2}
\end{aligned} \tag{2.20}$$

$$\begin{aligned}
\text{For } k'_-\text{:} \quad \left|\frac{1}{c} - k'_-\right| &= \left|\frac{1}{2c} - \frac{1}{2} + \kappa\right| = |\alpha + \kappa| \\
\left|1 - k'_-\right| &= \left|-(\frac{1}{2c} - \frac{1}{2}) + \kappa\right| = |-\alpha + \kappa| \\
H_{k'_-}(q) &= \frac{|\alpha + \kappa|}{|-\alpha + \kappa|} \frac{J_1\left(|-\alpha + \kappa| \Delta q\right)}{J_1\left(|\alpha + \kappa| \Delta q\right)} \\
&= \frac{1}{H_{k'_+}(q)}
\end{aligned} \tag{2.21}$$

The reciprocal relationship $H_{k'_-} = H^{-1}_{k'_+}$ can readily be seen as $|\alpha+\kappa| = |-\alpha-\kappa|$ and $|-\alpha+\kappa| = |\alpha-\kappa|$. In Fig. 2.8, we show an example of the filter responses at different depths represented in terms of k'. Note that for illustration purposes, for each response, only the portion within the main lobe (see examples in Fig. 2.7) is shown. We can see that for $k' < \frac{1}{2}(1+\frac{1}{c})$ the filters perform sharpening and for $k' > \frac{1}{2}(1+\frac{1}{c})$ the filters are lowpass ones [2]. The filter shape also changes across different depth k'. To further analyze the change in filter responses, with different values of k', we solve for the -$3dB$/+$3dB$ frequencies within the main lobe (as those shown in Fig. 2.8) of the blurring/sharpening filters. Polynomial approximation, $J_1(x) \approx \sum_{m=0}^{n} b_m x^m$, is used to represent $J_1(x)$ from $x = 0$ to its first zero-crossing location (see Fig. 2.7), such that the $\pm 3dB$ frequency can be calculated as ($X = \sqrt{1/2}$ for lowpass filter,$\sqrt{2}$ for sharpening filter):

$$
\begin{aligned}
H_Z(q) = \frac{|\frac{1}{c}-k'|}{|1-k'|}\frac{J_1(|1-k'|\Delta q)}{J_1(|\frac{1}{c}-k'|\Delta q)} &\approx \frac{|\frac{1}{c}-k'|}{|1-k'|}\frac{\sum_{m=0}^{n} b_m \left(|1-k'|\Delta q\right)^m}{\sum_{m=0}^{n} b_m \left(|\frac{1}{c}-k'|\Delta q\right)^m} = X \\
\sum_{m=0}^{n} b_m |1-k'|^{m-1} (\Delta q)^m &= X \sum_{m=0}^{n} b_m \left|\frac{1}{c}-k'\right|^{m-1} (\Delta q)^m \\
\sum_{m=0}^{n} b_m \left(|1-k'|^{m-1} - X\left|\frac{1}{c}-k'\right|^{m-1}\right)(\Delta q)^m &= 0 \qquad (2.22)
\end{aligned}
$$

Coefficients b_m can be determined using approximations proposed in the literature [35], or simply by solving a least-square regression for samples of $J_1(x)$ taken between 0 and the first zero-crossing x. We investigated both methods and found that the lowest order polynomial to achieve very close approximation over the entire range between 0

[2]Note here we demonstrate an example with $c < 1$ ($c = 0.7$). It is straightforward to demonstrate from Fig. 2.6 that for $c > 1$ we will instead have lowpass $H_{k'}$ for $k' < \frac{1}{2}(1+\frac{1}{c})$ and sharpening filters for $k' > \frac{1}{2}(1+\frac{1}{c})$.

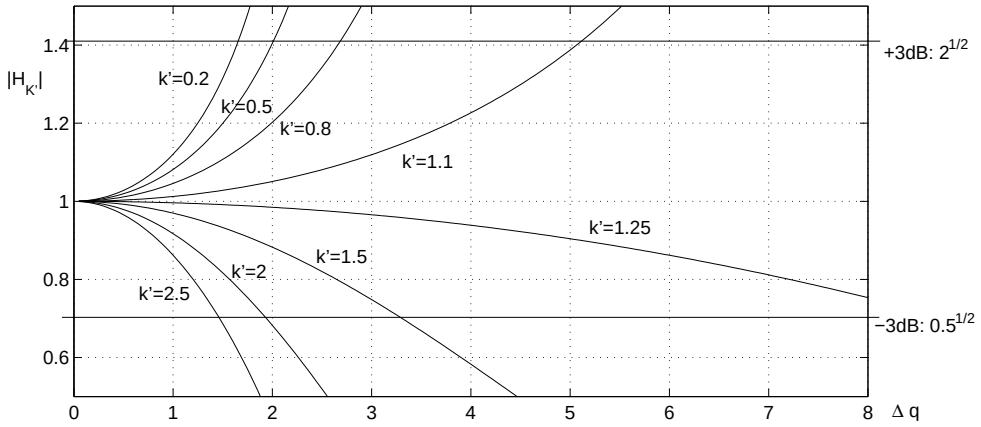

Figure 2.8: Filter responses at different k', with $c = 0.7$, $\frac{1}{2}(1 + \frac{1}{c}) \approx 1.2143$

and the first zero-crossing of $J_1(x)$, has order four ($n = 4$ in (2.22)). However, there is no direct analytical form to represent the roots of a 4th-order polynomial using its coefficients. Thus we illustrate how the $\pm 3dB$ frequency changes with k', calculated using (2.22) with 4th-order approximation, as curves in Fig. 2.9. As shown by the analysis of (2.20) and (2.21), in Fig. 2.9(a), for a given c, the curves on the two sides of $\frac{1}{2}(1 + \frac{1}{c})$ are symmetric, with one being the 3dB frequency and the other being the -3dB frequency. The $\pm 3dB$ frequency changes much more significantly as k' approaches $\frac{1}{2}(1 + \frac{1}{c})$. Subtracting $\frac{1}{2}(1 + \frac{1}{c})$ from k', Fig. 2.9(b) illustrates clearly the symmetry property. It also reveals two important properties of focus mismatches: (i) Larger focus setting difference (c away from 1) results in mismatch kernels with lower $\pm 3dB$ frequencies as compared to smaller focus setting difference (c closer to 1), leading to stronger impact on the images (i.e., stronger focus mismatch between two views). (ii) Smaller focus setting difference results in more rapid change in filter responses as k' changes, as compared to larger focus setting difference. (Note however that, as described in (i), this change corresponds to higher $\pm 3dB$ frequencies.)

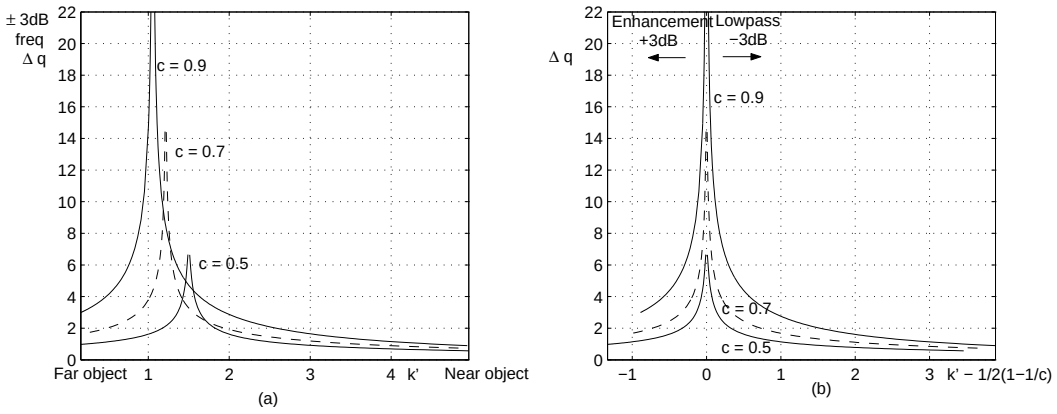

Figure 2.9: Variation of the $\pm 3dB$ frequencies

While we have used multiview as example, the analysis above can be easily applied to monoscopic video where focus changes occur over time. Instead of considering the two curves in Fig. 2.6 as corresponding to two different views, we can simply regard them as the β curves of the same camera at different time t_1 and t_2, while the focus setting is changing (e.g., adjust d to focus on different depths as in the dialog example in Section 2.1). Thus, from t_1 to t_2, objects at $k' < \frac{1}{2}(1 + \frac{1}{c})$ are getting sharpened as their β decreases. As for objects at $k' > \frac{1}{2}(1 + \frac{1}{c})$, they are getting blurred (β increases).

In the following subsection, we will consider a particular case in multiview system in which the cameras are arranged on a one-dimensional horizontal line with parallel shooting orientation. In this scenario, we will demonstrate that the disparity exhibited by images from different views can be exploited as an indication to identify depth-dependent focus mismatch.

2.3.1 Multiview with 1D Parallel Camera Arrangement

One of the most common multiview settings uses a 1D horizontal camera arrangement: Cameras are positioned along a horizontal line with equal spacing b between neighboring cameras, and their optical axes (Fig. 2.4) are parallel. Consider two neighboring cameras V1 and V2, where V1 is located to the left of V2. We once again assume they have the same f and a, and the focus mismatch is caused by a **very slight difference in** d which results in different Z^* and PSF_Z. Since V2 is to the right of V1 at a distance b, as compared to the scene captured by V1, we can regard the scene as shifted by $-b$ along the x-axis for the coordinate system centered at the lens of V2. From (2.1), due to the shift of $-b$, for a visible point with depth Z, the **centers of its projections** on the image planes of $V1$ and $V2$ will be located at P'_{V1} and P'_{V2} respectively as:

$$\begin{aligned} P \text{ at } (X,Y,Z)_{V1}, \text{ the projection } P'_{V1} \text{ at } (X',Y') &= \left(-\frac{d_{V1}}{Z}X, -\frac{d_{V1}}{Z}Y\right) \\ P \text{ at } (X-b,Y,Z)_{V2},\ P'_{V2} \text{ at } \left(-\frac{d_{V2}}{Z}(X-b), -\frac{d_{V2}}{Z}Y\right) &= \left(-\frac{d_{V2}}{Z}X + \delta_Z, -\frac{d_{V2}}{Z}Y\right) \\ \text{where } \delta_Z &= \frac{b \cdot d_{V2}}{Z} \end{aligned} \tag{2.23}$$

If the difference between d_{V1} and d_{V2} is negligible, which is most likely the case, then from (2.23) the **projection centers** of a point with depth Z will appear with a disparity of δ_Z between images from V1 and V2. The disparity δ_Z and depth Z are reciprocals: Objects closer to the cameras (smaller depth) will have a larger disparity; while objects far away (larger depth) will possess a smaller disparity. The relationship between the two

projections in V1 and V2 can be represented as follows, where $K(x, y)$ is the perfectly focused light intensity as in (2.11):

$$K_{Z,V2}(x, y) = K_{Z,V1}(x - \delta_Z, y) \tag{2.24}$$

Since the focal settings of the two cameras are not identical, (hence they focus on different depths $Z^*_{V1} \neq Z^*_{V2}$), they will have different PSF_Z due to the different β. For a visible region with depth Z, from (2.11) and (2.24) the corresponding images will be:

$$\begin{aligned} J_{Z,V1}(x, y) &= K_{Z,V1}(x, y) * PSF_{Z,V1}(x, y) \\ J_{Z,V2}(x, y) &= K_{Z,V1}(x - \delta_Z, y) * PSF_{Z,V2}(x, y) \end{aligned} \tag{2.25}$$

(2.25) means that for a 1D parallel camera arrangement with focus setting difference, the corresponding images in V1 and V2 for a region with depth Z will be displaced with disparity δ_Z, and will be affected by two different PSF_Z. If we can align the two images (e.g., via block matching) so that regions with depth Z match each other after disparity compensation, then the focus mismatch can be analyzed as described in Section 2.3. Furthermore, since the disparity δ_Z is a reciprocal of depth Z, it is proportional to the quantity $\frac{1}{k}$ as defined in (2.8). From (2.14) and (2.15), we can see that the value of β, which determines the PSF/OTF and consequently the filter H_Z, is approximately a *linear function* of δ_Z. Thus, difference in disparity will translate linearly into a difference in β, and therefore we can identify regions with different focus mismatch based on their disparity δ_Z, i.e., regions with similar disparity are likely to suffer from similar focus

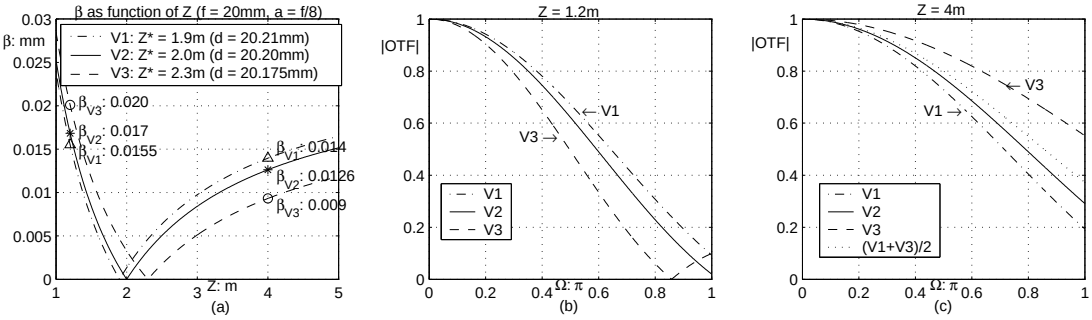

Figure 2.10: An numerical example of focus mismatch in multiview system

mismatch. Since there is no direct measurement of depth available, this property provides us a reliable method to partition an image into depth levels that suffer from different types of focus mismatch.

2.3.2 A Numerical Example

In this subsection, we introduce numerical example of focus mismatch in which specific values can be calculated for the reader to facilitate understanding of the analytical results. Let us consider a multiview system with three cameras V1, V2, and V3. They are set with the same focal length $f = 20mm$ (same zoom), and the same aperture settings $a = f/8$. However, the fine tuning of their perfect in-focus depth Z^* was not done perfectly, with $Z^*_{V1} = 1.9$m, $Z^*_{V2} = 2.0$m, and $Z^*_{V3} = 2.3$m. We can calculate their image plane distances d using (2.3), and the corresponding values are shown in Fig. 2.10(a). Once again we see that a slight change in d (-0.17% from 20.21 to 20.175) results in significant change in Z^* (+21.05% from 1.9 to 2.3).

As in Fig. 2.10(a), the focus setting mismatch results in differences of the β values of the cameras as functions of Z. An object at given depth Z will appear differently in V1, V2 and V3 under the influence of different β parameters. Before plotting the optical

transfer functions (OTF) at different depth, let us describe the frequency range we need to consider. Since for digital cameras, the image intensity is sampled by image sensors, only frequencies up to the Nyquist rate have to be taken into account. For a 1/2" sensor type (H×W = 6.4mm×4.8mm), a resolution of 640×480 pixels leads to a sample-spacing of 0.01mm between pixels. The Nyquist rate is $100/2 = 50$(cycles/mm). In the polar system, $q = \sqrt{50^2 + 50^2} \approx 70.71$. Thus, if we plot OTF_Z as (2.16) with β expressed in the unit of mm, we only need to consider the range up to $q = 70.71$. This value corresponds to $\Omega = \pi$ in the digital domain. Fig. 2.10(b) and (c) show the differences in the corresponding OTF. If we encode V2 using V1 as a reference ($c = Z^*_{V1}/Z^*_{V2} = 1.9/2 = 0.95$, $\frac{1}{2}(1 + \frac{1}{c}) \approx 1.026$), for image portions corresponding to visible regions at $Z = 1.2$m ($k' = 2/1.2 \approx 1.67$), we need to perform lowpass filtering on the data from V1. For visible regions at $Z = 4$m ($k' = 2/4 = 0.5$), the corresponding image portions in V1 need to be slightly sharpened. On the other hand, if we instead use V3 as a reference to encode V2 ($c = Z^*_{V3}/Z^*_{V2} = 2.3/2 = 1.15$, $\frac{1}{2}(1 + \frac{1}{c}) \approx 0.935$), image portions corresponding to depth $Z = 1.2$m need some sharpening in order to match V2, while regions at $Z = 4$m have to undergo a significant amount of lowpass filtering.

Now if the cameras are arranged on a 1D horizontal line with parallel orientation, and the spacing between two cameras is 10cm, then from (2.23) the disparity between V1 and V2 for objects at depth 1.2m and 4m can be calculated as:

$$\begin{aligned} \delta_{1.2\text{m}} = \frac{10}{120} \times 20.20 &\approx 1.68 \text{ (mm), which corresponds to 168 pixels} \\ \delta_{4\text{m}} = \frac{10}{400} \times 20.20 &\approx 0.5 \text{ (mm), which corresponds to 50 pixels} \end{aligned}$$

Disparity between V2 and V3 can be calculated in a similar manner. By performing disparity estimation, we can identify regions within a frame that correspond to different depth levels. Based on the depth-dependency property, each level is anticipated to have different type of focus mismatch kernels such as depicted in Fig. 2.10.

2.4 Summary

In this chapter, we analyzed focus mismatches occur in video content which are caused by focus setting differences. We utilize geometrical optics to derive characteristics of the images captured with a lens, and then demonstrate how images will be affected under different focus settings. The following analytical results provide useful insight for us to design coding techniques to compensate for the focus mismatch:

- For lens-based imaging systems, PSF/OTFs are determined by the blur diameter β, which is a function of depth Z and camera parameters a, f, d.
- β is approximately a *linear function of the reciprocal of depth* ($\frac{1}{k}$).
- In the presence of focus setting differences, the corresponding images will exhibit a *depth-dependent* mismatch. To deal with different types of focus mismatch at different depths, compensation kernels should be designed according to the depth-composition within the scene.
- For a given depth, the mismatch can be modeled using a *convolution kernel* which captures the difference between two point spread functions (or in frequency domain, between two optical transfer functions).

- The focus mismatch kernels can be represented as *blurring/sharpening filters* that are *circular symmetric* in spatial domain.

- For a given pair of views, if Z^*_{V2} and Z^*_{V1} (perfect-in-focus depth) are made available, we can determine c and consequently the value of $\frac{1}{2}(1 + \frac{1}{c})$, which divides the disparity (depth) into two regions that require different filter types: blurring and sharpening.

- In terms of k', on the two sides of $\frac{1}{2}(1 + \frac{1}{c})$, the filter responses are reciprocal.

- For 1D parallel camera arrangement, the disparity δ_Z between frames from different views is also a reciprocal of depth Z (same as the blur diameter β). This property can be exploited in order to identify regions within the image that suffer from different types of mismatch.

- Larger focus setting difference (c away from 1) results in mismatch kernels with lower $\pm 3dB$ frequencies, leading to stronger focus mismatches.

- Across different depth/disparity values, smaller focus setting difference (c closer to 1) results in more rapid change in filter responses as compared to stronger focus setting mismatch.

In the next chapter, we will describe our adaptive reference filtering (ARF) methods to compensate for focus mismatch, which are designed based on the analytical results above.

Chapter 3

Adaptive Reference Filtering (ARF) for Video Coding

3.1 Introduction

In this chapter, we consider the problem of encoding video content that exhibits focus mismatch, such as focus setting differences in inter-view prediction in MVC and focus change over time in monoscopic video. Based on the analysis in Chapter 2, systems for efficient focus mismatch compensation in video coding shall be designed with the following requirements in mind. First, *local compensation* is useful in addressing *depth-dependent* focus mismatches, as different portions of a video frame can undergo different blurriness/sharpness changes with respect to the corresponding areas in frames used as reference. Second, since the characteristics of focus mismatch are determined by camera parameters and depth-composition within the scene, they will change from view to view (in MVC) as well as over time. Thus compensation process should be *adaptive* to the different mismatches exhibited by the video frames. Finally, to optimize overall coding efficiency, the decisions on whether or not to use mismatch compensation, and the selection of mismatch compensation kernels, should be based on *rate-distortion* (R-D) criteria.

To improve motion compensation performance in monoscopic video coding, different approaches have been proposed in which the reference frames are filtered to generate new predictors [6, 46, 51, 52]. The general idea behind these methods is that predictors better matched to the current frame can be created after filtering.

Budagavi proposed blur compensation [6], where a fixed set of blurring (lowpass) filters are used to generate blurred reference frames. For focus changes which lead to circular symmetric mismatch kernels, as we demonstrated in Chapter 2, Budagavi utilizes simple $n \times n$ uniform averaging filters with different sizes n; as for camera panning (which leads to directional mismatch kernel), $1 \times n$ horizontal and $n \times 1$ vertical averaging filters are considered. This technique has two shortcomings for the focus mismatch scenarios we consider. First, the filter selection is made only at the frame-level, i.e., applying different filters to different parts of a frame is not considered. However, as noted in Chapter 2, for the depth-dependent focus mismatches, adaptive local compensation should be exploited. Second, this method relies on a very limited predefined filter set. Sharpening filters (high frequency enhancement), for example, are not included. As a result, the usefulness of the predefined filter set is very constrained as it will only be able to cover particular types of mismatch. Instead, our approach estimates the mismatch kernels between the reference frame and the current frame such that the compensation filters are designed adaptively. In the final motion/disparity search, we allow each block to select the filtered version of predictor that gives the lowest R-D cost to ensure optimized coding efficiency.

In [46, 51, 52], adaptive filtering methods have been proposed for generating subpixel references for motion compensation. Vatis et al. [46], after an initial motion search using the 6-tap interpolation filters defined in H.264/AVC [54], divide blocks in the current

frame into groups exclusively based on the *subpixel positions* of their motion vectors.[1] This partition method is used in order to generate different subpixel interpolation filters for different positions. For each group, an filter is adaptively estimated by minimizing squared prediction error. The subpixels of the reference frame will then be generated using these adapted interpolation filters. In the final motion compensation, the encoder chooses the best match by testing different subpixel positions on the same reference frame. This approach, which we will refer to as adaptive interpolation filtering (AIF), aims to address the aliasing problem and motion estimation error when generating subpixel references. Instead, in our work we design filters by identifying blocks suffering from different types of focus mismatch. Filtered reference frames will first be generated by applying the estimated filters. Then, on each of these filtered reference frames, sub-pixel interpolation (such as in H.264/AVC) will be performed, leading to additional coding gains for disparity compensation.

In this chapter, we propose a novel adaptive reference filtering (ARF) method for encoding video with focus mismatch. Based on the analysis in Section 2.3, we first model predictive coding with focus mismatch using point spread functions and provide a derivation of how the proposed approach is designed. The main contribution is that, to compensate for depth-dependent focus mismatch, we adaptively design multiple filters by estimating the mismatch kernels. In our approach, video frames are first divided into regions that suffer from different types of focus mismatch. For MVC inter-view prediction, we exploit block-wise disparity vectors as feature to roughly classify image blocks into different scene-depth levels. As for monoscopic video with temporal focus

[1] For example, $(1\frac{3}{4}, 23\frac{1}{2})$ and $(45\frac{3}{4}, 6\frac{1}{2})$ will be assigned to the same group.

change, given that depth information is not directly available, we propose an encoding method that estimates localized focus changes. Frames will be then partitioned into regions, each consisting of macroblocks (MB) that suffer from a similar type of focus change (e.g., blurring or sharpening). After frame partitioning, for each region, a 2D filter is calculated to compensate for the focus mismatch by minimizing the prediction residue energy (MMSE filter). These filters can be regarded as *estimators of the focus mismatch kernels*. To provide better coding efficiency, we generate multiple filtered reference frames by applying the obtained filters, and allow each block to be predicted from the reference that provides lowest R-D cost.

We also extend ARF to MVC inter-view bi-directional prediction cases (B-frames), in which predictive coding is performed by using reference frames from two reference lists (denoted as List 0 and List 1). A straightforward extension of ARF to B-frames can be achieved by designing depth-dependent filters that minimize the prediction error between current blocks and the chosen bi-predictors, which will be obtained by averaging two reference blocks, one from each reference list. Note that such an extension would be analogous to that selected for bi-prediction in adaptive interpolation filtering (AIF) [49], in which for a given interpolation position, only one filter is designed and is applied to generate interpolated pixel values for references in both List 0 and List 1. This approach does not separately consider the possibility that different types of mismatch may exist with respect to reference frames in the two lists. Instead, mismatches from the two lists are considered *jointly* in the filter design. Estimating / applying more than one set of filters to different lists is not possible in this framework. For bi-directional prediction, the key observation is that with the above described approach, *joint* filter design is followed

by conventional *independent* search for predictors in each list. Because of this mismatch between filter design and search, the gain with respect to un-filtered bi-prediction is can be reduced. To tackle this problem, we propose a filter design approach that estimates two sets of depth-dependent filters, each set compensating for the focus mismatch affecting one of the two references used for bi-directional prediction. This leads to increased gains as compared to the straightforward filter design for the averaged predictors.

The rest of this chapter is organized as follows: In Section 3.2, based on the analysis in Chapter 2, we first provide a focus mismatch model for predictive video coding. The proposed adaptive reference filtering (ARF) method will then be described in detail in Section 3.3. We discuss two scenarios with focus mismatch: Inter-view coding in MVC, and focus change in monoscopic video. The extension to MVC inter-view B-frames is presented in Section 3.4. Simulation results based on H.264/AVC are summarized in Section 3.5. In Section 3.6, we analyze the complexity of filter estimation in our ARF approach. Finally, conclusions are provided in Section 3.7.

3.2 Filtering Model for Video Coding with Focus Mismatch

Let us denote $I_C(x, y)$ the luminance pixel value of the current frame to be encoded at pixel position (x, y), and let $I_R(x, y)$ denote the corresponding luminance pixel value in the reconstructed reference frame. Let (dv_x, dv_y) denote the displacement vector (i.e., a disparity vector for inter-view prediction or a motion vector in temporal prediction). From the analysis in Chapter 2, in the presence of differences in focus settings, the corresponding images will exhibit *depth-dependent* mismatch which for a given depth,

can be modeled using a *convolution kernel* H_Z which captures the difference between two point spread functions (or in frequency domain, between two optical transfer functions, as described in (2.19)). In this chapter, we model predictive video coding with focus mismatch for pixel (x, y) corresponding to an object at depth Z as:

$$I_C(x, y) = H_Z * I_R(x + dv_x, y + dv_y) \quad \text{where } * \text{ denotes the convolution.} \tag{3.1}$$

For depth-dependent focus mismatch, *multiple* mismatch kernels H_Z might be required in order to model blurriness/sharpness mismatch in different regions within a frame, corresponding to different depths Z, as depicted for example in Fig. 2.8. Based on this model, we propose a coding method in which the *reference frame is first filtered by estimators of the mismatch kernels* H_Z chosen to minimize the prediction error with respect to the current frame. In an ideal scenario, for a region at depth Z that undergoes a certain type of focus change, minimum mean-squared error (MMSE) estimation can be derived by jointly optimizing over both mismatch and displacement:

$$\min_{\psi, dv_x, dv_y} \sum_{x,y} \left(I_C(x, y) - \psi_Z * I_R(x + dv_x, y + dv_y)\right)^2 \tag{3.2}$$

The filter ψ_Z will be an estimator of the mismatch kernel H_Z for a given region with depth Z. However, an encoding system with such joint optimization will require excessive computation. Instead, we adopted a procedure similar to that proposed in adaptive interpolation filtering [46, 51, 52], i.e., such that the displacement is estimated first (e.g. using block-based motion/disparity search), and then the filter coefficients of

ψ_Z are determined. In this approach, the filter will be designed based on the *displacement compensated prediction error* between the reference frame and the current frame. In the next section, we will describe our proposed adaptive reference filtering approach for video coding.

3.3 Adaptive Reference Filtering

To design an adaptive filtering approach for situations in which different regions within a frame may suffer from different types of blurriness/sharpness changes, locally adaptive compensation has to be enabled. In this work, we propose a coding method in which, after performing an initial motion/disparity search to obtain displacement vectors and establish block correspondence, the current image is partitioned into regions suffering from different types of focus mismatch [2]. Each region D_k (k =1, 2, 3...) will then be associated with one adaptive filter ψ_k to be designed in the next step. We call this process *frame partition for adaptive filter design*. The filter ψ_k for each D_k is optimized to minimize the residual energy for all pixels within D_k, i.e.,

$$\min_{\psi_k} \sum_{(x,y)\in D_k} \left(I_C(x,y) - \psi_k * I_R(x+dv_x, y+dv_y)\right)^2, \tag{3.3}$$

This approach will allow multiple filters to be estimated for different parts of a video frame that undergo different blurriness/sharpness changes with respect to the corresponding areas in the reference frames. These filters will be applied to the reference frame to generate filtered references that provide better matches. Then the final motion/disparity

[2]Here we describe our approach in general. Specifics of frame partitions for the multiview and monoscopics cases will be explained in Section 3.3.1.

compensated prediction is performed using both original and filtered frames as references. At this stage each block is allowed to select the reference that provides the lowest R-D cost, regardless of what the initial classification of the block was. Fig. 3.1 provides a flowchart of adaptive reference filtering for video coding. In the following subsections, we describe each step in detail.

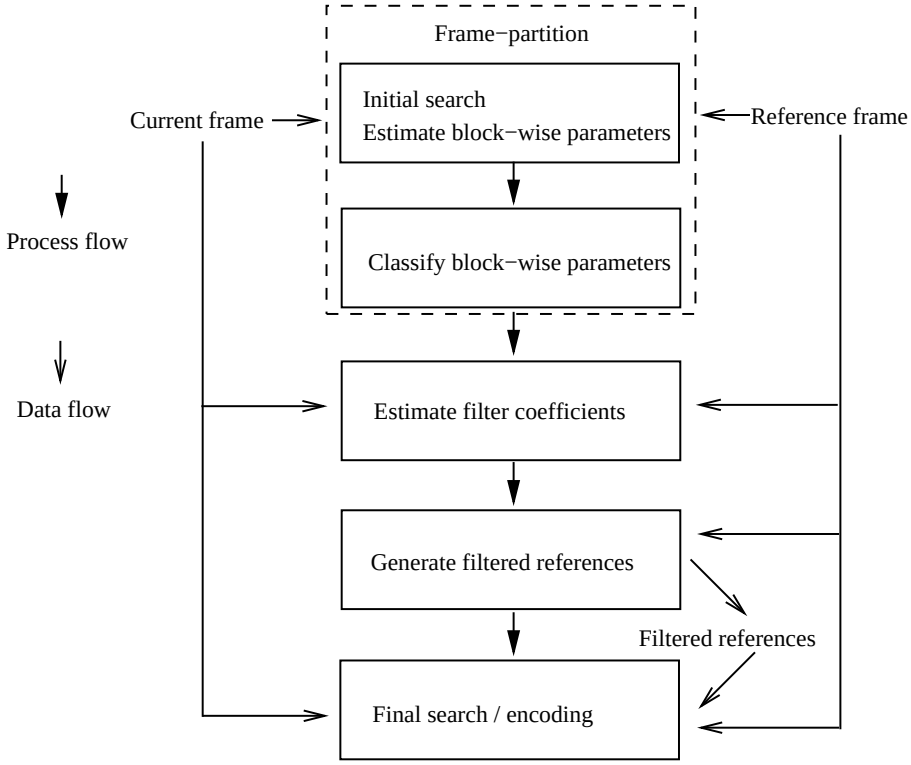

Figure 3.1: Flowchart of Adaptive Reference Filtering for video coding

3.3.1 Frame Partition for Adaptive Filter Design

The first step is to identify different types of blurriness/sharpness changes in different parts of the current frame. An exhaustive approach could be, after having computed the motion/disparity vectors, to assign adaptively to each block in the frame a filter

that minimizes the motion/disparity compensated prediction error. This approach is optimal in the sense that for every block the residual energy is minimized. However, it would significantly increase the bitrate since we would have to transmit filter coefficients for every single block. To limit the amount of side information (filter coefficients) while maintaining the ability to compensate for focus mismatches, we use a procedure to roughly partition a frame into regions, each containing blocks of pixels that suffer from a similar type of focus mismatch. This can be regarded as a region-based approach, in contrast to the global (frame-wise) approach of blur compensation in [6] and to the more local, block based approach we just mentioned. The key issue here is to achieve frame partition such that different types of focus mismatch within a frame can be reliably estimated. In this section, we will propose solutions to two focus mismatch examples: (i) Inter-view prediction in MVC with 1D parallel camera arrangement, and (ii) Monoscopic video when focus setting changes over time.

Inter-view prediction in MVC

As discussed in Section 2.3, under a given camera setting, the type of focus mismatch depends on the depth of the scene. To partition image into regions suffering from different focus mismatch, it is reasonable to consider procedures to identify image regions with different depth levels. When multiple cameras are employed, such as in multiview systems, disparity information has been widely used as an estimation of scene depth [15]. As discussed in Section 2.3.1, for multiview systems with a 1D parallel camera arrangement, the disparity δ_Z between different views is the reciprocal of depth Z: $\delta_Z = \frac{b \cdot d}{Z}$ (again d and b representing the image plane distance and the spacing between two consecutive cameras).

Based on this relationship, numerous approaches have been proposed to estimate scene depth by computing the disparity [1, 16, 20]. While in some existing approaches the goal is to find an accurate/smooth disparity map at the pixel-level, here we simply aim at *separating objects with different scene depths* by modeling their disparities. This is similar to video object segmentation methods in which the motion field is used to identify moving objects [58]. To reduce complexity, compressed domain fast segmentation methods have been proposed that use the block-wise motion vectors, obtained with video coding tools, as input features to classify image blocks [2, 21, 50]. Similarly, we consider procedures to classify blocks into depth levels based on their corresponding disparity vectors. As shown in Chapter 2, the blur diameter β can be closely approximated as a linear function of δ_Z (Fig. 2.6). Thus, **partition a frame into regions of similar δ_Z leads to regions for which a similar β can be applied**, which serves well for our goal of identifying different types of focus mismatch. For multiview systems with cameras arranged in parallel on the same *horizontal line*, classification can be achieved by considering only the x *component* of the disparity vectors. For a 2D camera arrangement as can be found in a camera array, the classification could be extended by taking both x and y components as input features.

We propose to use classification algorithms based on Gaussian mixture models (GMM) to separate blocks into depth-level classes. We adopted expectation-maximization algorithm (EM) based on the GMM [38] to classify the disparity vectors and their corresponding blocks [2, 50, 55]. In this work, an unsupervised EM classification tool developed by Bouman [4] is employed. To automatically estimate the number of Gaussian components in the mixture (thus making the approach unsupervised), the software tool performs an

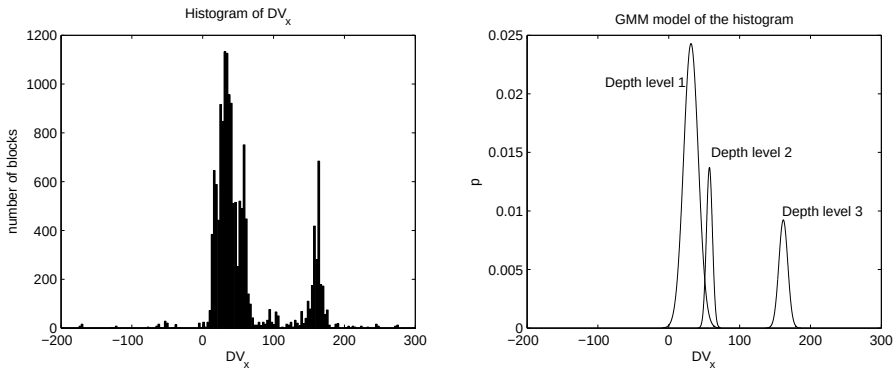

Figure 3.2: Disparity vectors from view 6 to view 7 at the 1st frame in *Ballroom*: Histogram and GMM

order estimation based on the minimum description length (MDL) criterion. The only required parameter to be specified is the maximum number of Gaussian components (K) allowed in the GMM. This tool applies MDL to select the number of Gaussian component (K, $K-1$, ... to 1). We refer to [4, 10, 39] for details about such techniques. Parameters of Gaussian components are estimated using an iterative EM algorithm. Each Gaussian component is used to construct a Gaussian probability density function (pdf) that models one class for classification. Likelihood functions can be calculated based on these Gaussian pdfs. Disparity vectors are classified into different groups by comparing their corresponding likelihood value in each Gaussian component. Blocks are classified accordingly based on the class label of their corresponding disparity vectors. Refining processes can also be considered, such as eliminating a class to which a very small number of blocks has been assigned. In the classification result, each class represents a depth level within the current frame, and blocks classified into a certain level will be associated with one adaptive filter. To illustrate this frame partition based on classification of disparity vectors, we provide a segmentation result in Figs. 3.2 and 3.3.

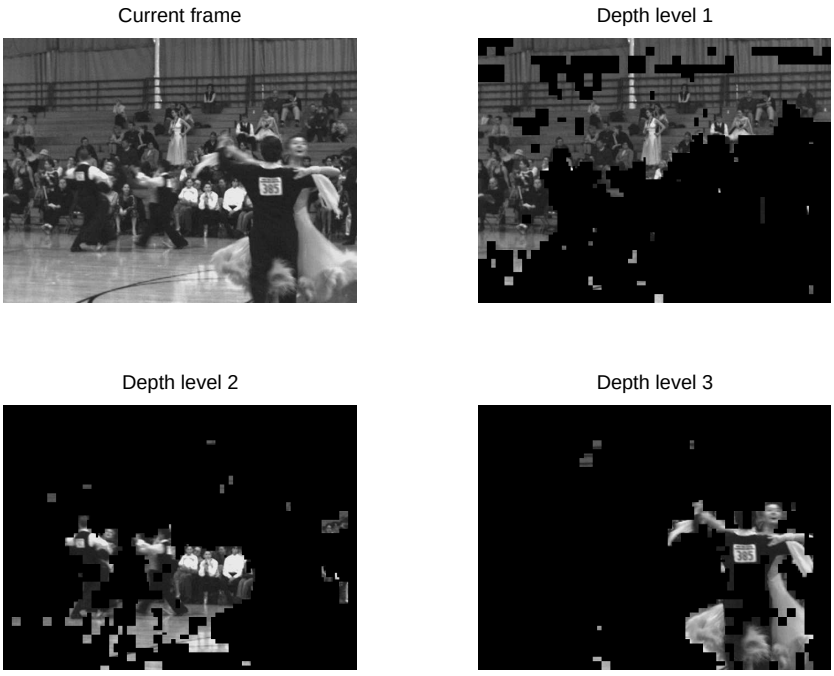

Figure 3.3: The corresponding frame partition result of Fig. 3.2

Fig. 3.2 shows the histogram of the x-component of disparity vectors, obtained from the initial disparity estimation. A corresponding GMM is constructed with a number of components estimated to be 3 (K was set to 4). In Fig. 3.3, the corresponding blocks within each class are shown. It can be observed that after disparity-based classification, depth class 1 corresponds to the far background; class 2 captures two dancing couples and some audience in the mid-range, along with their reflection on the floor; and class 3 includes the couple in the front. Note that intra-coded blocks in the initial disparity estimation are not involved in the filter association process. In this example, the classification tool successfully separates objects with different depths in the current frame. For each depth level, we will then estimate a focus mismatch kernel. (More examples of the proposed disparity-based frame partition are provided in Fig 3.4 and Fig 3.5.)

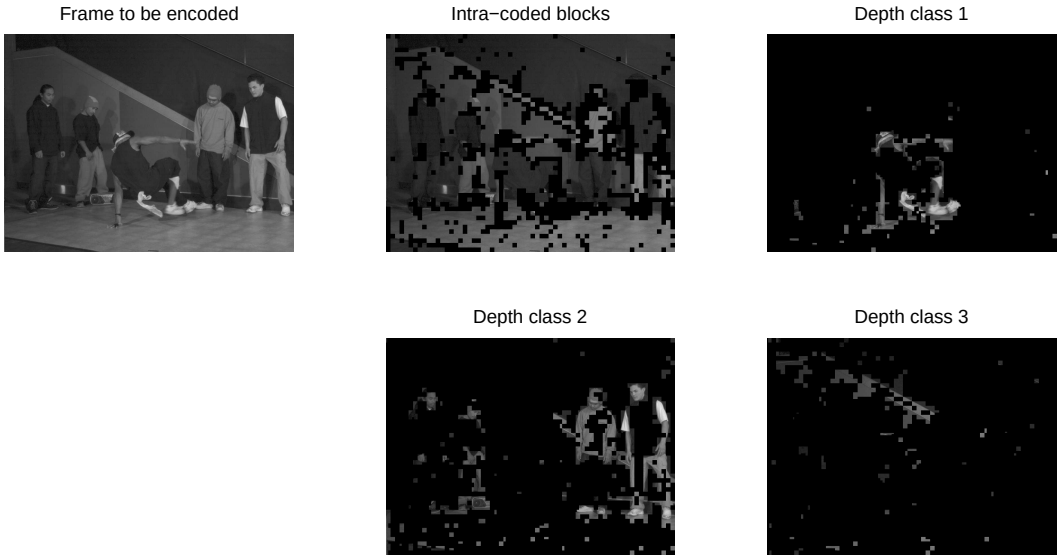

Figure 3.4: Frame partition result: Breakdancer View 1 frame 0

Temporal prediction in monoscopic video

Now we consider compensating focus change in monoscopic video. Again our goal is to identify regions in a video frame that suffer from different types of focus mismatch and design filters which estimate the mismatch kernels. For instance, in the dialog example described in Section 2.1 where the focus is changing from the first characters to the second one, blocks corresponding to the two characters at different depth levels (D_1 and D_2) will be associated with two different filters, ψ_1 and ψ_2, which will produce blurring and sharpening, respectively.

To achieve such classification *without disparity information available* to estimate depth, we considered two possible approaches. First, a set of predefined filters can be chosen to operate on the reference frame. During an initial motion compensation, each block selects the filter that provides the lowest matching error. Blocks with similar filter selections can then be grouped into a class. This approach has a drawback in that it is

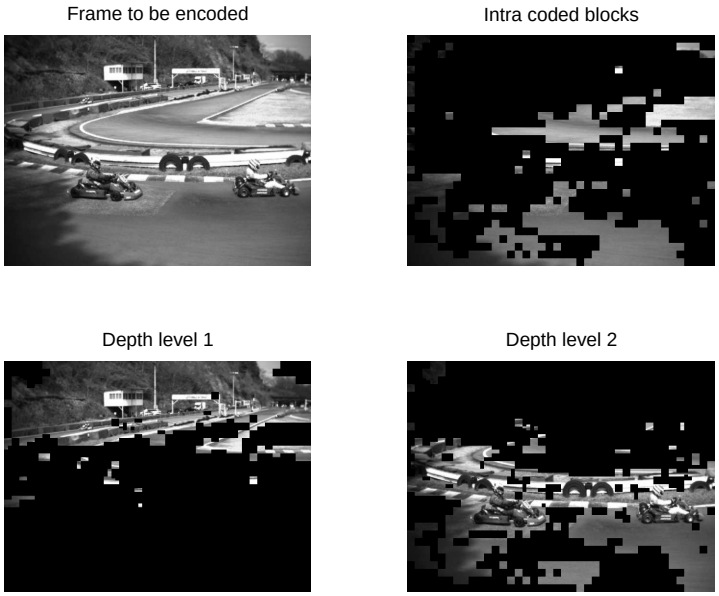

Figure 3.5: Frame partition result: Race1 View 2 frame 30

possible that the predefined filter set is not complete enough to model all types of focus change within the frame. Thus, blocks exhibiting focus changes that are not covered by the predefined filter set, may be grouped with blocks having very different characteristics, leading to suboptimal compensation filters. With only limited knowledge of the focus settings (for example the perfect in focus depth Z^* is not likely to be known), it will be hard to build a satisfactory filter set, unless a very large set of predefined filters is used.

To avoid this problem, we investigate a second approach: During the initial motion compensation, in order to model the localized focus mismatch, a simple filter is estimated for each MB to minimize the prediction residual energy (MMSE filter). The collection of all these MB-wise MMSE filters provides a more comprehensive description of various focus changes present in the current frame. MBs will then be separated into groups

by clustering based on the similarity of their respective filters. This procedure can be summarized as follows:

1. Initial search to obtain displacement vector (dv_x, dv_y).

2. For each MB, calculate MMSE filter f_{mb} such that:

$$\min_{f_{mb}} \sum_{(x,y)\in MB} \left(I_C(x,y) - f_{mb} * I_R(x+dv_x, y+dv_y)\right)^2 \tag{3.4}$$

$$\text{where} \quad f_{mb} = \begin{pmatrix} a & b & a \\ b & c & b \\ a & b & a \end{pmatrix}$$

3. Classify f_{mb} into groups. Filter coefficients are considered as features for the classification algorithm. Each MB belongs to the class to which its corresponding filter was assigned. For each class, one adaptive filter will be estimated in the next stage to compensate for the focus mismatch.

The role of f_{mb} is to capture local focus changes from the reference frame to the current frame. We selected a 3×3 filter with circular symmetry for the following reasons. First, these f_{mb} are estimators of the point spread functions representing different focus changes, which are isotropic as shown in Chapter 2. Second, we are using their coefficient as the input features for classification. Larger filters with more coefficients will result in a much higher dimensional problem, which increases significantly the classification complexity. More importantly, this could also lead to an over-specified classification, which could be sensitive to filter variations, and may not be suited to our goal of identifying *a few rough classes of focus changes* within each frame.

Taking the coefficients as features, we can group the f_{mb} into classes. Such classification can be visualized by plotting each set of f_{mb} coefficients (a, b, c) as a point in 3D space. We have observed that the filter points all lie very close to the $4a + 4b + c = 1$ plane (which we denote as P_f). This is reasonable since the MMSE system is attempting to find a weighted average for pixel values. By performing principal component analysis (PCA), we observe a system with a very insignificant third eigenvalue as compared to the first two (in the order of 10^{-15}), which indicates that the assumption that (a, b, c) belong to a plane is a reasonable one.

To select a classification tool, we performed the following study on f_{mb}: On plane P_f, we shift the filter coefficients away from the MMSE point (a, b, c) by $(\Delta a, \Delta b, \Delta c)$ and record how the MSE changes with different shifts. Statistics are gathered on a frame by frame basis. Fig. 3.6 shows some results for the sequence *fondue-multi* [3] in which the camera focus is changing back and forth among people at different scene depths. We observe that the increase in MSE away from the optimal point has different gradients in different directions. These findings suggest that the classification algorithm should take *directional information* into account, in addition to considering the distance between data points. Simply using the Euclidean distance to cluster the various filters into classes will not be appropriate as this would implicitly assume that the errors generated by changes in the filter coefficients are equal in all directions.

We propose once again to use classification algorithms based on multidimensional GMMs to group the f_{mb} into classes. GMM techniques incorporate covariance matrices such that the directional information can be modeled. Filters f_{mb} are classified into

[3] fondue-multi.wmv by Yi-Ren Ng, Light Field Photography with a Hand-held Plenoptic Camera, Stanford Computer Graphics Lab, http://graphics.stanford.edu/papers/lfcamera/refocus/

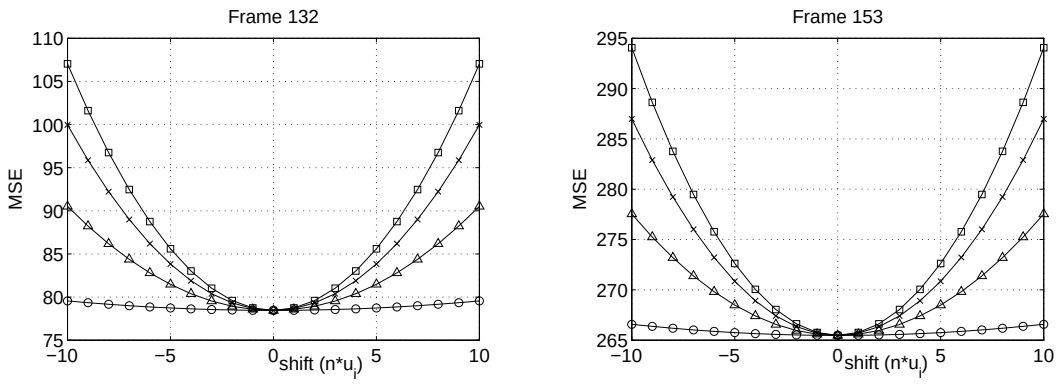

Figure 3.6: Variations of MSE when shifting f_{mb} parameters away from the MMSE solution (3.4) by $(\Delta a, \Delta b, \Delta c) = n\vec{u_i}$. The four curves represent results with different $\vec{u_i}$. $\times$: $\vec{u_1} = (-0.05, 0.025, 0.1)$, $\circ$: $\vec{u_2} = (0.025, -0.05, 0.1)$, $\square$: $\vec{u_3} = (-0.07, 0.0513, 0.0748)$, and $\triangle$: $\vec{u_4} = (0.07, -0.0805, 0.0419)$. Note that (i) the norms of these $\vec{u_i}$ are the same, and (ii) $4\Delta a + 4\Delta b + \Delta c = 0$ such that the shifted filters will still be on P_f.

different groups by comparing their corresponding likelihood value in each Gaussian component. Refining processes can also be considered in the classification based on GMM, such as removing points with too low likelihood from the classes, or eliminating a class to which too few points have been assigned.

An example of frame partition results using the proposed method on the sequence *fondue-multi* is provided in Fig. 3.7. In this example, camera focus is shifting from the front to the back. The first three people are becoming increasingly blurred, while the others are becoming more clear. Based on the MB-wise focus change estimators f_{mb}, the classification tool successfully separates these two groups, as can be observed in classes 1 and 2.

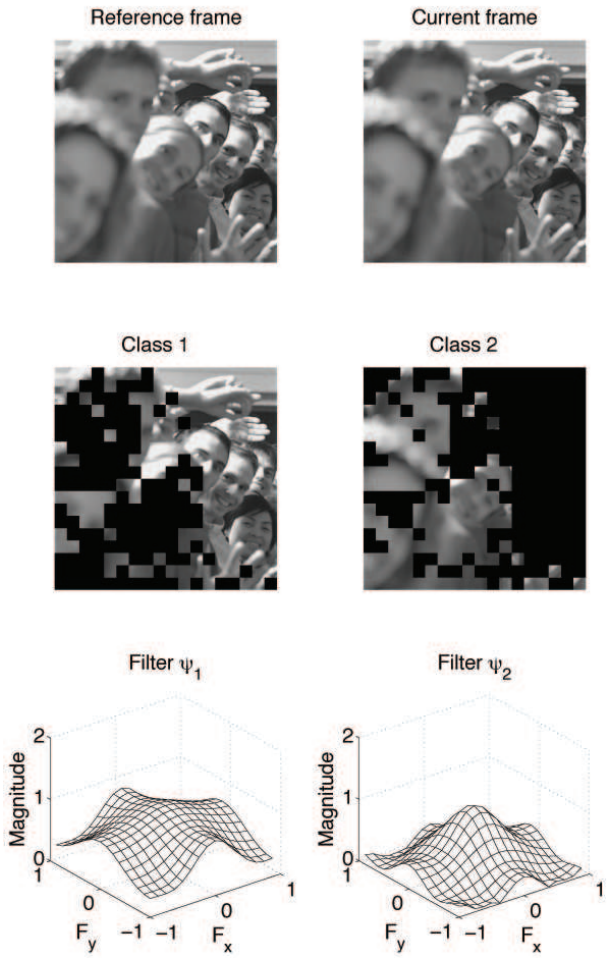

Figure 3.7: An example of frame partition based on focus changes

3.3.2 Filter Design by Estimating Mismatch Kernels

We now discuss how to select a filter for all blocks belonging to a given region/class D_k, which are therefore assumed to have similar focus mismatch. We replace the convolution notation in (3.3) by explicitly expressing the filter operation as

$$\min_{\psi_k} \sum_{(x,y)\in D_k} \left(I_C(x,y) - \sum_{j=-n}^{n} \sum_{i=-m}^{m} \psi_k(i,j) \cdot I_R(x+dv_x+i, y+dv_y+j) \right)^2 \qquad (3.5)$$

The size and shape of 2D filters can be specified by changing m and n. In Chapter 2 we established that the focus mismatch is expected to be *isotropic*. Hence, in this work we utilize square shaped filter kernels with $m = n$. Our analytical results also show that the frequency responses of these filters depend on the camera parameters and object depth, as depicted for example in Fig. 2.8: Some kernels have very smooth responses while others have sharp transitions. It is well known that in the design of FIR filters, smooth responses can be realized with short length filters while for sharp responses we need longer filters. Thus, in order to select a filter size that can reliably estimate the underlying mismatch kernels, we need knowledge about the setting differences in camera parameters and the depth composition of the scene. Furthermore, as described in the numerical example in Section 2.3.2, the dimension and resolution of the camera sensor array also has to be known such that we can convert the filter responses from analog domain to digital domain, and then calculate the filter size in units of pixels. For example, a given blur diameter β will cover more pixels for sensor arrays with smaller distance between pixels. A given analog filter response will lead to a sharper transition in digital domain if the pixel distance is smaller, i.e., higher Nyquist rate. Thus, other things being equal (i.e., same a, f, d, Z, and sensor array dimension), the selected filter length should be proportional to the pixel resolution (inversely proportional to the pixel distance). In practical scenarios, it is very unlikely that we will have full access to all these parameters. Therefore, in this section, we investigate filters with different sizes and constraints. In adaptive interpolation filtering (AIF) approaches, even-length (6×6) filters are proposed in order to interpolate subpixels in between integer pixels. In our proposed ARF approach, we apply adaptive filters directly to the reference frame in order to generate filtered references that are better

matched to the current frame. Odd-length filters centered at the pixel to be filtered are employed in this work.

The filter coefficients $\psi_k(i,j)$ that satisfy (3.5) can be determined by taking derivatives with respect to each coefficient, i.e., $\forall\ \psi_k(I,J)$ where $-m \le I \le m, -n \le J \le n$, we look for a filter such that:

$$\begin{aligned}
\frac{\partial}{\partial \psi_k(I,J)} \sum_{(x,y)\in D_k} \left(I_C(x,y) - \sum_{j=-n}^{n} \sum_{i=-m}^{m} \psi_k(i,j) I_R(x+dv_x+i, y+dv_y+j) \right)^2 &= 0 \\
\sum_{(x,y)\in D_k} \left(I_C(x,y) - \sum_{j=-n}^{n} \sum_{i=-m}^{m} \psi_k(i,j) I_R(x+dv_x+i, y+dv_y+j) \right) I_R(x+dv_x+I, y+dv_y+J) &= 0 \\
\sum_{j=-n}^{n} \sum_{i=-m}^{m} \left(\psi_k(i,j) \sum_{(x,y)\in D_k} I_R(x+dv_x+i, y+dv_y+j) I_R(x+dv_x+I, y+dv_y+J) \right) & \\
= \sum_{(x,y)\in D_k} I_C(x,y) I_R(x+dv_x+I, y+dv_y+J) & \qquad (3.6)
\end{aligned}$$

These Wiener-Hopf equations will lead to optimal linear Wiener filters. By defining $(\tilde{x}, \tilde{y}) = (x+dv_x, y+dv_y)$, and denoting $\tilde{I}_R$ as the disparity shifted pixel value at $I_R(x+dv_x, y+dv_y)$, (3.6) can be further written as:

$$\begin{aligned}
\sum_{j=-n}^{n} \sum_{i=-m}^{m} \psi_k(i,j)\ E\left[I_R(\tilde{x}+i, \tilde{y}+j) I_R(\tilde{x}+I, \tilde{y}+J) \right] &= E\left[I_C(x,y) I_R(\tilde{x}+I, \tilde{y}+J) \right], \\
\text{that is} \quad \sum_{j=-n}^{n} \sum_{i=-m}^{m} \psi_k(i,j)\ Cor_{\tilde{I}_R \tilde{I}_R}(I-i, J-j) &= Cor_{I_C \tilde{I}_R}(I,J), \qquad (3.7)
\end{aligned}$$

where $E[\cdot]$ is the expectation operator and Cor is the correlation function. Both $E[\cdot]$ and Cor operate over all the blocks that are classified into the depth-level D_k. It can be

seen that the linear MMSE Wiener filter is optimized based on the autocorrelation of the disparity shifted pixel value $\tilde{R}$; and the cross-correlation between the current frame and $\tilde{R}$.

In this linear system, the number of equations will be equal to the number of coefficients in ψ_k. Filters with more unknowns can be more efficient to compensate for blurring/sharpening and thus reduce residual energy. However, this comes at the expense of having to transmit more filter coefficients. (For example, a circular symmetric 3×3 filter contains only 3 coefficients, while a full 3×3 matrix has 9 coefficients). In Chapter 2 we have shown that the focus mismatch is *isotropic*. This suggests that we can impose circular symmetry constraint on the filter coefficients. Thus in this section, we consider two examples of 5×5 filters ($m = n = 2$), with a symmetry constraint:

$$\psi_{55cir} = \begin{pmatrix} f & e & d & e & f \\ e & c & b & c & e \\ d & b & a & b & d \\ e & c & b & c & e \\ f & e & d & e & f \end{pmatrix} \tag{3.8}$$

$$\psi_{55hv} = \begin{pmatrix} i & g & e & g & i \\ h & d & b & d & h \\ f & c & a & c & f \\ h & d & b & d & h \\ i & g & e & g & i \end{pmatrix} \tag{3.9}$$

The filter in (3.8) is circular symmetric, with only 6 coefficients to be estimated ($a \sim f$), which we denote as ψ_{55cir}. The filter in (3.9), which we denote as ψ_{55hv}, can be viewed as a compromise between a full matrix and the circular symmetric ψ_{55cir}. It has 9 different coefficients ($a \sim i$). After partitioning the frame using methods described in

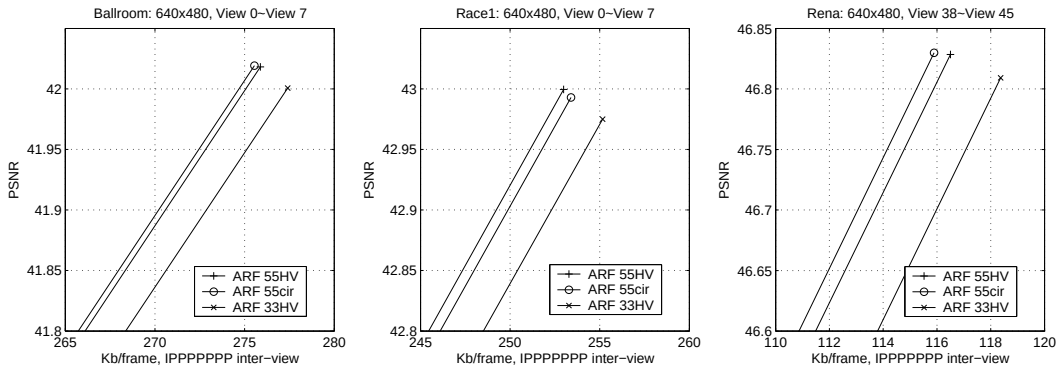

Figure 3.8: Performance of ARF using different filter sizes/constraints (QP22)

Section 3.3.1, for each depth region, a filter in one of the above forms can be obtained by solving (3.6).

Fig. 3.8 provides simulation results of MVC inter-view coding for different timestamps, using ARF with different filter sizes / constraints. (Refer to Section 3.3.3 and Section 3.5 for details on how the simulations were conducted.) We observe that the differences in coding efficiency between using ψ_{55cir} and ψ_{55hv} as filter structures are small, with greater differences at higher bitrates (low QP). Thus in Fig. 3.8 we provide results at QP = 22 for which it is easier to observe the difference in performance. It can be seen that the performance of the two filters ψ_{55cir} and ψ_{55hv} is very similar, e.g., less than 0.025 dB difference in sequences tested. More importantly, in two of the three sequences (*Ballroom* and *Rena*), reducing the number of coefficients by imposing circular symmetric actually provides a slight coding gain. These results indicate that by exploiting the isotropic properties of focus mismatch, we can simplify the filter structure, reducing the side information (coefficients) to be transmitted, while preserving the ability to reliably estimate mismatch kernels. In the simulations in Section 3.5, we will provide ARF coding

results using circular symmetric filters as ψ_{55cir}. (As a reference, in Fig. 3.8 we also include results using 3×3 filters with horizontal/vertical constraints similar to those in (3.9). At QP $= 22$, compared to the difference between ψ_{55cir} and ψ_{55hv}, using such filter results in much larger degradation in coding efficiency, i.e., 0.05 dB degradation for *Ballroom* and *Race1*, and 0.1 dB for *Rena*.)

Figures 3.9, 3.10, and 3.11 provide the frequency responses of the calculated ψ_{55cir}, when we perform inter-view coding between view pairs in *Race1* sequences. In each figure, the three curves correspond to filters estimated for different depth levels of a given anchor frame: the filters, from left to right, correspond to regions ranging from far (small disparity) to near (large disparity). These estimated depth-dependent focus mismatch kernels demonstrate similar behavior to that of the analytical results as depicted for example in Fig. 2.8: For the scenario in which the current frame is blurred as compared to the reference (Figures 3.9 and 3.11), it can be seen that the filters have a low-pass characteristic. When the blurring affecting the frame is stronger (Fig. 3.11), the resulting filters have sharper transitions. In both figures, it can be seen that the responses change gradually from smaller disparity to larger one (see the dot points). On the other hand, when the reference frame is a blurred version of the current frame (Fig. 3.10), the filters emphasize higher frequency ranges so that the reference can be sharpened to create a better match. In this example, the responses rise to a peak at about $\Omega = 0.4\pi$.

As for the focus change example in monoscopic video, the filters' frequency responses are shown in in Fig. 3.7. For parts of the image that are blurred (Class 2), the corresponding filter ψ_2 is a blurring filter (lowpass) with a Gaussian-shaped frequency response. For parts that are getting sharpened (Class 1), the filter ψ_1 emphasizes more the higher

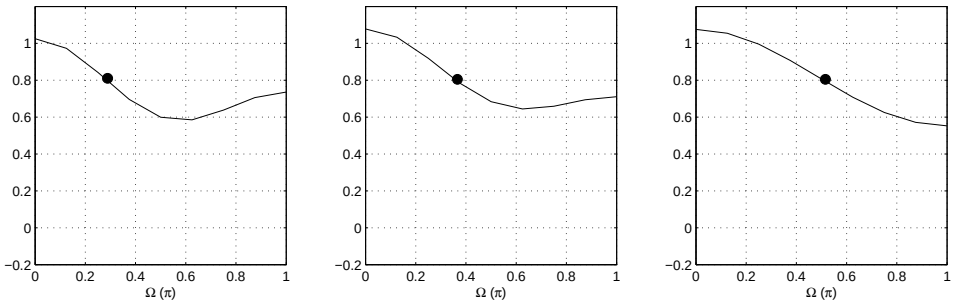

Figure 3.9: Frequency responses of estimated filters when performing inter-view prediction from *Race1* V2 to V3 at Anchor 9. V3 is slightly blurred w.r.t reference V2.

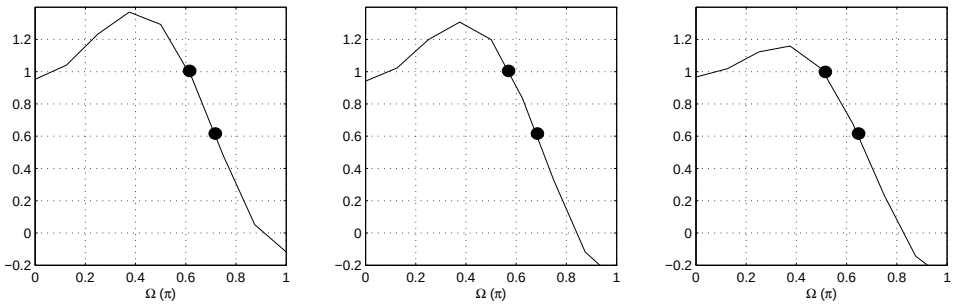

Figure 3.10: Frequency responses of estimated filters when performing inter-view prediction from *Race1* V4 to V3 at Anchor 3. Reference V4 is blurred w.r.t V3.

frequency ranges (non-Gaussian shape as compared to ψ_1) so that the reference can be sharpened to create better match.

3.3.3 Encoding with Filtered References

The optimized filters will be applied to the reference frame in order to provide better matches for predictive video coding. In the reference picture list, the original unfiltered reference as well as multiple filtered references are stored. If subpixel disparity estimation is employed, all these references will be interpolated to generate subpixel values using interpolation filters specified by the codec (e.g., 6-tap interpolation filters in H.264/AVC).

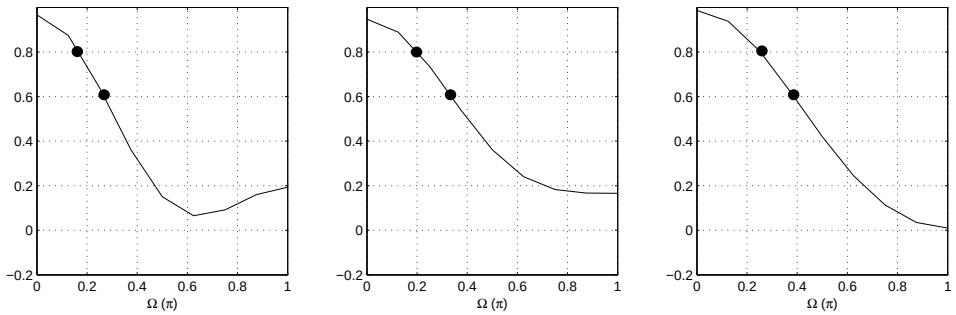

Figure 3.11: Frequency responses of estimated filters when performing inter-view prediction from *Race1* V6 to V5 at Anchor 7. V5 is strongly blurred w.r.t reference V6.

During the final encoding process, original and filtered references can be regarded as inputs for predictive coding with multiple references, such as specified in H.264/AVC [54]. This provides two advantages: Firstly and most importantly, each block can select a block in any filtered or original reference frame, based on R-D optimization.[4] This ensures highest coding efficiency. Secondly, the filter selection of each block can easily be handled by signaling the reference frame index in the bitstream.

To correctly decode the video sequence, the filter coefficients also have to be transmitted. In this work, we directly extend the method proposed in [45, 47], in which the filter coefficients are quantized and encoded as frame level overhead. Using the 5×5 filters with circular symmetric constraints as (3.8), which has 6 coefficients, and assuming there are 4 filters (which is the maximum number of filter allowed in our simulations), there will be a total of 24 coefficients as side information. As comparison, for each frame

[4]Note that after this stage, the filter selection could be regarded as a new "frame partition" D_k, and filters ψ_k could be estimated again based on MBs in different classes. Thus, the estimation of D_k and ψ_k can be carried iteratively until a stopping criterion is met. The complexity involved in such process will be fairly high. In this work we limited ourselves to an algorithm without any iteration.

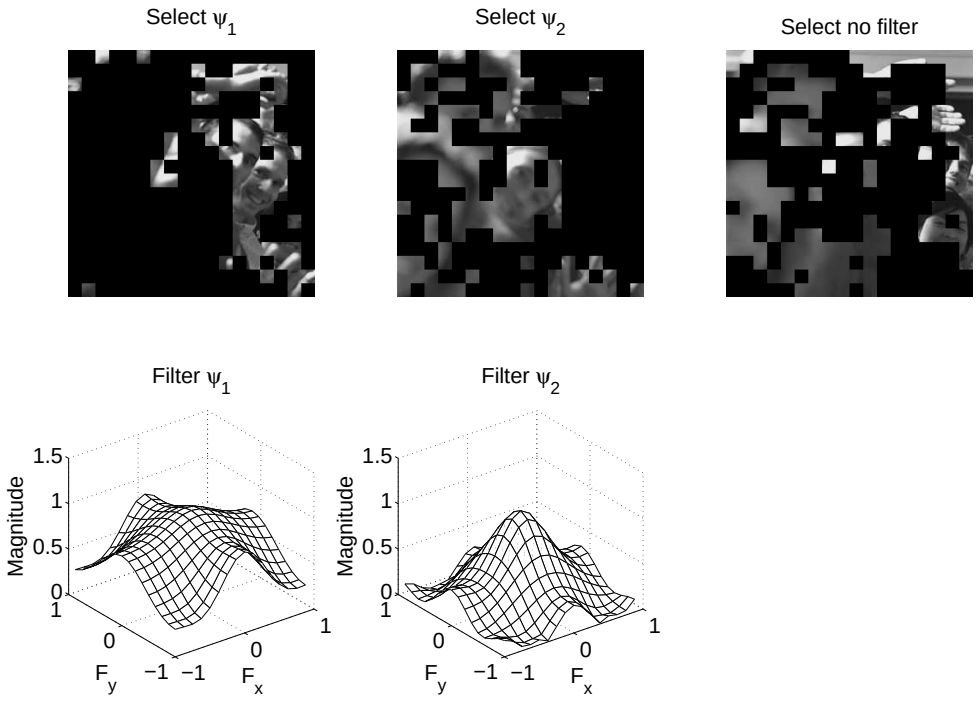

Figure 3.12: Encoding selection with adaptive filtering

AIF [46] requires 54 coefficients to be transmitted in order to specify the interpolation filters. Of course, our method has additional side information as compared to AIF, namely the reference frame index to indicate block-wise filter selection.

In Fig. 3.12 we provide the final filter selection result corresponding to the temporal focus change example in Fig. 3.7. We can see that different filters were selected for the front three people and the others (ψ_1 and ψ_2). The two people in the very back chose the unfiltered reference frame, as they are not being altered much by the focus changes. One interesting point to note is that for smooth regions such as the first three people's foreheads and cheeks, unfiltered reference is also preferred. This is because for these plain regions, changing focus would not have much effect on the pixel values. We observed this same phenomenon for other frames as well.

3.4 ARF for MVC Bi-directional Disparity Compensation with Focus Mismatch

Based on the assumption that regions with similar depth will suffer from similar focus mismatch, in the previous section we proposed an adaptive reference filtering (ARF) approach for MVC inter-view coding, in which a frame is partitioned into different depth levels and depth-adaptive MMSE filters are estimated to compensate for focus mismatch. This method was developed for **inter-view P-frames**, for which a single reference frame is used, taken from one of the neighboring views (IPPP for coding V0∼V3 for example).

In this section, we extend focus mismatch compensation to B-frames, where predictive coding is performed by using reference frames from two reference lists (denoted List 0 and List 1), which consist of previously encoded frames. In MVC inter-view coding, these lists contain frames from different neighboring views (e.g., frames from the left and right views in List 0 and List 1, respectively). As a result, a B-frame to be encoded may suffer from different types of focus mismatch with respect to the reference frames from List 0 and List 1. In what follows, we will first revisit the focus mismatch example in Chapter 2 and then discuss different approaches to design filters. In particular, we emphasize the interaction between filter design and bi-predictive search when filtered references are generated.

3.4.1 Inter-view Bi-directional Prediction with Focus Mismatch

Once again, for a camera equipped with lens, we denote f the focal length, a the aperture diameter, and d the "image plane distance", i.e., the distance between the image plane

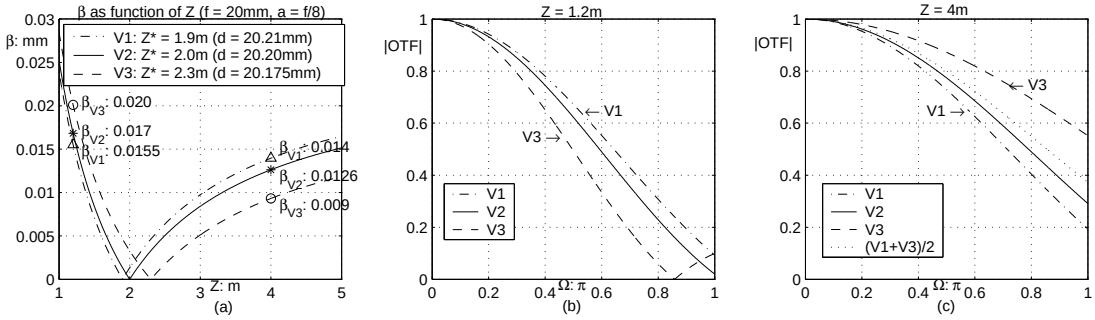

Figure 3.13: An example of focus mismatch in multiview bi-prediction, with $Z^*_{V1} = 1.9$m, $Z^*_{V2} = 2.0$m, and $Z^*_{V3} = 2.3$m. We consider image sensor type 1/2" (H×W = 6.4mm×4.8mm) with a resolution of 640×480 pixels, i.e. the spacing between pixels is 0.01mm (Nyquist rate 100/2 = 50 cycles/mm). In polar system, $q = \sqrt{50^2 + 50^2} \approx 70.71$, which corresponds to $\Omega = \pi$ in (b) and (c).

and the lens. In Chapter 2, we showed that in the presence of difference in focus setting, frames from different views will exhibit a mismatch that is a function of parameters f, a, d and object depth Z. For points at depth Z, the corresponding projections on the image plane will be uniform circles with diameter $\beta = \frac{af(|Z-Z^*|)}{Z(Z^*-f)}$. Z^* is a specific depth at which an object will produce a point projection (perfectly focused) on the image plane: $Z^* = \frac{d \cdot f}{d-f}$.

Now let us revisit the example described in Section 2.3.2, where three cameras V1, V2, and V3 form a multiview system. Assume that the cameras have the same focal length setting f (same zoom), and their aperture settings are also identical: $a = f/8$. However, assume that the fine tuning of their Z^* was not done perfectly ($Z^*_{V1} \neq Z^*_{V2} \neq Z^*_{V3}$), resulting in differences in their β values as functions of Z. Fig. 3.13 shows the same example as in Section 2.3.2 with heterogeneous settings. Figures 3.13(b) and (c) demonstrate the differences in the corresponding optical transform functions (OTF), i.e., the frequency transform of the point spread function (PSF). If we encode V2 with **bi-directional prediction** by putting V1 in List 0 and V3 in List 1 as references, for image

portions correspond to visible regions at $Z = 1.2$m, we need to perform blurring on V1 and sharpening on V3 in order to match V2. On the other hand, for visible regions at $Z = 4$m, the corresponding image portions in V1 need to be slightly sharpened while V3 has to undergo a significant amount of blurring. As for the averaged predictor $\frac{1}{2}(V1 + V3)$ (dotted line) in Fig. 3.13(c), a blurring filter is required to bring down the OTF to that of V2.

If V1, V2, and V3 are arranged on a 1-D horizontal line from left to right with equal spacing b between each other, and assuming their image plane distances d are very similar, we have discussed in Section 2.3.1 that an object at depth Z will result in a disparity $\delta_Z = \frac{b \cdot d}{Z}$ from V1 to V2 and also from V2 to V3. Since the blur diameter β is approximately a linear function of δ_Z (see Section 2.3), we can exploit disparity vectors to identify image portions suffering from different types of focus mismatch. In Section 3.4.2, we will discuss adaptive filtering methods using the three-view example we just discussed.

3.4.2 ARF and Bi-directional Disparity Search

As in Section 3.3, we propose to utilize a two-pass coding scheme with an initial search (the first coding pass) to obtain the block-wise disparity vectors (DVs) and predictors, for disparity-based frame partition and for designing filters. In what follows, we will discuss different filter estimation methods, especially emphasizing how filter estimation interacts with bi-predictive search when filtered references are generated.

3.4.2.1 Filter Design for Averaged Bi-predictor

In B-frames, for a block that chooses to use bi-prediction, its predictor is actually the average of two reference blocks, one from the reference frame in List 0 (I_R^{L0}) and one from the reference frame in List 1 (I_R^{L1}). A straightforward filter design approach, which minimizes the prediction error between current blocks and the averaged predictors would be, for pixels within a given depth level D_i:

$$\min_{\psi_i^{BI}} \sum_{(x,y)\in D_i} \left(I_C(x,y) - \psi_i^{BI} * \frac{1}{2}\left[I_R^{L0}(x+dx_0, y+dy_0) + I_R^{L1}(x+dx_1, y+dy_1) \right] \right)^2 \tag{3.10}$$

In (3.10), (x, y) is the pixel position within a frame, (dx_0, dy_0) and (dx_1, dy_1) are the disparity vectors for I_R^{L0} and I_R^{L1}, respectively, and $*$ denotes convolution. Using the approach as described in Section 3.3.1, we can partition a frame into regions with different depth levels, by classifying the DVs in either direction (L0 or L1), or by taking both directions as two input features for classification. Since for each depth-level D_i the filter is designed for the averaged predictors, it should be applied to both List 0 and List 1, thus filtered references $\psi_i^{BI} * I_R^{L0}$ and $\psi_i^{BI} * I_R^{L1}$ can be generated. In Table 3.1, we summarize this approach as **Method A**.

However, whether the optimal pair of predictors can be found depends on how the bi-predictive search is performed. Searching *jointly* for pairs of vectors from List 0 and List 1 would lead to the optimal solution, but would require high complexity. Typically, simpler search schemes are utilized, such as independent search, which results in degradation of coding efficiency as compared to joint pair-wise search: There is no guarantee that

searching independently for the best matching blocks in $\psi^{BI} * I_R^{L0}$ and $\psi^{BI} * I_R^{L1}$ will lead to an optimal solution to the problem of finding the two blocks in List 0 and List 1 that provide the best prediction after averaging *and* filtering. Clearly, this is also the case for bi-prediction even if no filtering is used [14]. However, in the following, we will demonstrate that the suboptimality is exacerbated when filtering is used.

Consider first the case of **independent search**, where for each block, the encoder *independently* searches for the best predictor from references in List 0 and the best predictor from references in List 1. The bi-predictor is formed by simply averaging the two without performing any additional search. As for the example in Fig. 3.13(c) for depth level at $Z = 4$m, after applying the lowpass filter ψ^{BI} designed for $\frac{1}{2}(I_{V1} + I_{V3})$, the filtered reference $\psi^{BI} * I_{V1}$ will actually have stronger mismatch with respect to V2 as its frequency response is further attenuated as compared to that of V1. During the search within List 0, due to the effect of the lowpass filter ψ^{BI}, the reference $\psi^{BI} * I_{V1}$ may not be preferred over I_{V1}, i.e., it is less likely to be selected. Consequently, the improved predictor $\frac{1}{2}\psi^{BI} * (I_{V1} + I_{V3})$ may not even be tested by the encoder.

As an alternative, in an **iterative search** [14], the search is conducted by, iteratively, fixing the obtained predictor from one side ($I_R^{L0/L1}$) to estimate the best predictor from the the other side ($I_R^{L1/L0}$). This can improve performance as compared to independent search, as some joint estimation is made possible. However the iterative process could still be trapped in a local minimum. For example in Fig. 3.13(c), if the initial selected predictor from List 0 is V1 instead of $\psi^{BI} * I_{V1}$, the resulting predictor after several iterations may still not converge to the optimal predictor $\frac{1}{2}\psi^{BI} * (I_{V1} + I_{V3})$.

One possible approach to resolve such problem above, without performing exhaustive search on pairs of vectors, is to modify bi-predictive search so that within each list, instead of picking only a single "best" predictor, we record the best matched predictors from *each* filtered/non-filtered reference frame $\{I_R^{L0},\ \psi_1^{BI} * I_R^{L0},\ \psi_2^{BI} * I_R^{L0} \dots\}$ and $\{I_R^{L1},\ \psi_1^{BI} * I_R^{L1},\ \psi_2^{BI} * I_R^{L0} \dots\}$. With different combinations of one predictor from each side, multiple averaged predictors can then be evaluated. This would increase the complexity on top of the independent search, and also increase memory requirement as multiple pairs of vectors have to be recorded for mode decision.

In addition to the problems due to the search algorithm, the filter design approach in (3.10) has another drawback that it may not lead to improved predictors within each list. For B-frames, a block is allowed be encoded using predictor from only one of the lists, if the rate-distortion (RD) cost of doing so is lower than using the averaged bi-predictor. However in (3.10) the filters are designed jointly for averaged blocks, so there is no guarantee that after applying them to *individual* frames they will provide good approximations to the original frame. As we discussed in the example of Fig. 3.13(c), the effect of the lowpass ψ^{BI} in fact leads to the filtered reference $\psi^{BI} * I_{V1}$ having stronger mismatch to compensate for I_{V2}. As a result, the filtered references in each list, when used by themselves, may not provide better coding options.

3.4.2.2 Filter Design for Predictors from Each Reference List

To overcome the drawbacks (limited coding choices, integration with bi-predictive search) of the method in (3.10), we consider an alternative filter design approach that estimates

depth-related filters for each reference list. After the first coding pass, assuming a horizontal camera arrangement, we use the same approach as described in Section 3.3.1 to partition the current frame I_C into depth-levels D_1, $D_2 \cdots$ D_K by taking dx_0 and dx_1 as *two features* to classify blocks. In other words, we consider a two-dimensional feature space: Objects closer to the cameras have larger disparities dx_0 *and* dx_1, pointing to opposite directions; while both disparities will be small for far away objects. Instead of minimizing error with respect to the averaged predictor, two sets of filters are estimated, one for List 0 and the other for List 1:

$$\begin{aligned} \Psi^{L0} &= \left\{ \psi_i^{L0} \left| \min_{\psi_i^{L0}} \sum_{(x,y)\in D_i} \left[I_C(x,y) - \psi_i^{L0} * I_R^{L0}(x+dx_0, y+dy_0) \right]^2 \right. \right\} \\ \Psi^{L1} &= \left\{ \psi_i^{L1} \left| \min_{\psi_i^{L1}} \sum_{(x,y)\in D_i} \left[I_C(x,y) - \psi_i^{L1} * I_R^{L1}(x+dx_1, y+dy_1) \right]^2 \right. \right\} \end{aligned} \quad (3.11)$$

This filter design method directly addresses the potentially different types of depth-dependent mismatch exhibited in reference frames from List 0 and List 1 (e.g., as in the example depicted in Fig. 3.13). In (3.11), sets Ψ^{L0} and Ψ^{L1} will both contain K filters. They will be applied to List 0 and List 1 respectively to generate filtered references. In this approach, a given block in I_C will participate in both filter estimations to minimize prediction errors with respect to references in List 0 and List 1.

Note that our approach here is different from a fully independent design, which would involve *performing the complete ARF design twice*: one for L0, one for L1, both with classification and filter estimation as in Section 3.3. In such fully independent approach, which is summarized as **Method B** in Table 3.1, there are also two sets of filters, but a

block in the current frame may belong to two different classes for List 0 and List 1; while in our approach (**Method C** in Table 3.1) there is a single class (*joint classification* with two features, dx_0 and dx_1) for each block.

The proposed two-sets filter design, **Method C**, has the following advantages:

1. Better integration with conventional bi-predictive search schemes: since filters are optimized independently for each list, the search within each list is likely to obtain better matched predictors. Since the two predictors are both focus compensated, they can be used to form the averaged bi-predictor, or serve as the starting point for iterative search.

2. More coding options: Based on (3.11), the filtered references in each list provide better matched predictors that can be used *by themselves, i.e. as P-mode instead of B-mode*, leading to more options (predictor from one of the lists, or from the averaged bi-predictor) for encoder to perform RD optimization.

3. Potential speed up for bi-directional search: In our approach, for a given class k, ψ_k^{L0} and ψ_k^{L1} **are designed for the same depth level within frame** I_C. Thus if we observe that a given block selects a particular filtered reference $\psi_{k'}^{L0} * I_R^{L0}$ after the search within List 0, it is reasonable to constrain the search in List 1 to the reference $\psi_{k'}^{L1} * I_R^{L1}$. A constrained bi-predictive search can be designed based on the search results from one of the lists.

With the advantages above, the joint classification followed by independent filter estimation approach **Method C**, is preferred over **Method A** and **Method B**.

Table 3.1: Filter design methods for bi-directional disparity compensation

	Filter design for averaged bi-predictor	Filter design for predictors from each list	
	Method A: Joint design	**Method B:** Independent design (i.e. perform ARF-P twice)	**Method C:** Joint classification followed by independent filter estimation
Frame partition	One partition D_i by classifying dx_0 and dx_1	Two partitions: D_i^{L0} by classifying dx_0, and D_j^{L1} by classifying dx_1	One partition D_i by classifying dx_0 and dx_1 as two features
Filter estimation	For blocks in D_i, estimate MMSE filter ψ_i^{BI} with respect to the averaged bi-predictor	For blocks in D_i^{L0}, estimate MMSE filter ψ_i^{L0} with respect to the predictor from List 0. For blocks in D_j^{L1}, estimate MMSE filter ψ_j^{L1} with respect to the predictor from List 1	For blocks in D_i, estimate one MMSE filter ψ_i^{L0} with respect to the predictor from List 0, and one MMSE filter ψ_i^{L1} with respect to the predictor from List 1
Filtered references	$\psi_i^{BI} * I_R^{L0}$ and $\psi_i^{BI} * I_R^{L1}$	$\psi_i^{L0} * I_R^{L0}$ and $\psi_j^{L1} * I_R^{L1}$	$\psi_i^{L0} * I_R^{L0}$ and $\psi_i^{L1} * I_R^{L1}$

3.5 Simulation Results

We performed simulations based on H.264/AVC coding standard. To partition a frame, the EM classification tool [4] based on GMM is combined with our encoder. The classifier takes disparity vectors (multiview case) or coefficients of f_{mb} (monoscopic video) as input features. To solve the block-wise MMSE filters f_{mb}, we extended the code from [45]. For both methods, reference frame management functions have been modified to store filtered reference frames. Finally, as described in Section 3.3.3, filter coefficients are encoded as in [45, 47].

The proposed adaptive reference filtering approach (ARF) is compared with AIF [46] and current H.264/AVC. Our proposed ARF utilizes multiple filtered versions from a *single* reference frame. If the EM classification generates K classes, each with a corresponding filter, there will be $N = K + 1$ references in the reference list, including the original unfiltered one. In H.264/AVC, motion compensation with multiple references is a coding tool [54] that also aims to improve coding efficiency by providing better matches. In our simulations, the maximum K allowed is 4. Thus, we also compare our method to H.264/AVC with the number of reference frames set to 5.

Inter-view prediction in MVC: P-frames

In Section 3.3, depending on the information available, we proposed two methods to partition frames into regions suffering from different types of focus mismatch. For MVC inter-view prediction, the depth-dependency characteristics of focus mismatch can be exploited by using disparity information to classify blocks. Here we would like to first verify the efficiency of such design approach. We conduct simulations of ARF based on the classification of disparity vectors, denote as ARF-Z, and also the classification based on the estimated coefficients of MB-wise filters f_{mb}, denote as ARF-f_{mb}. Using H.264/AVC reference software JM10.2 [42], we encode anchor frames only at given timestamps using inter-view coding, i.e., we take a sequence of frames captured at the same time from different cameras and feed this to the encoder as if it were a temporal sequence. The intra period is set equal to the number of views such that the coding structure for anchor frames is IPPP, as depicted in Fig. 1.2. Fig. 3.14 shows the rate-distortion (RD) comparison between the two filter design methods, along with H.264/AVC results as references. The

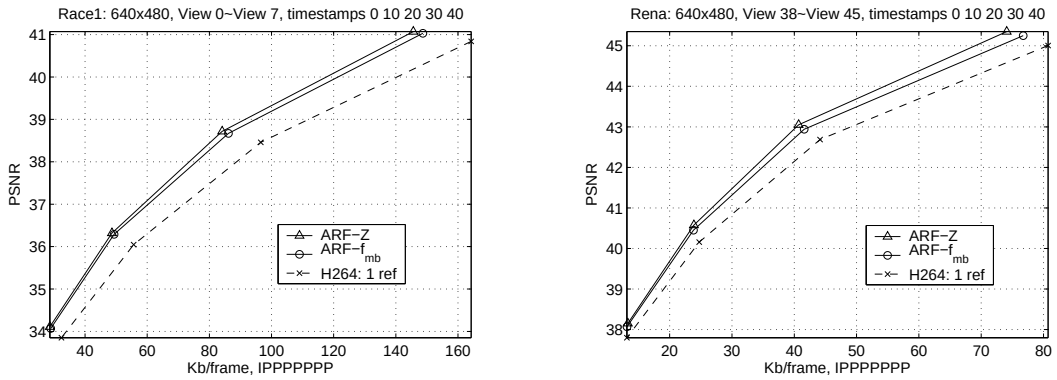

Figure 3.14: Comparison between two ARF methods on inter-view coding

four rate points were obtained at (from left to right) QP = 36, 32, 28, and 24. It can be seen that using disparity information to identify different types of focus mismatch achieves higher coding efficiency as compared to classification based on estimated f_{mb}. The gain is larger at higher bitrates: At QP=24, ARF-Z has 0.15 dB gain over ARF-f_{mb} for *Race1* and 0.25 dB gain for *Rena*. The results justify that the utilization of disparity is a reliable way of estimating different focus mismatch kernels. Furthermore, ARF-Z is less complex than ARF-f_{mb}. In what follows, all our comparisons will be based on ARF-Z.

It can be seen from Fig. 3.15 that for the multiview sequences we tested, ARF-Z provides higher coding efficiency than H.264/AVC, although the gains vary significantly depending on the test sequence. In the *Ballroom* sequence, almost no cross-view mismatch can be observed. Furthermore, frames from different views have been rectified (properly registered) by applying homography matrices [18]. In this situation, the three enhanced predictive coding schemes: multiple reference frame, AIF and ARF all provide very similar coding efficiency, i.e., about 0.3 $\sim$ 0.4 dB gain over H.264/AVC with one reference.

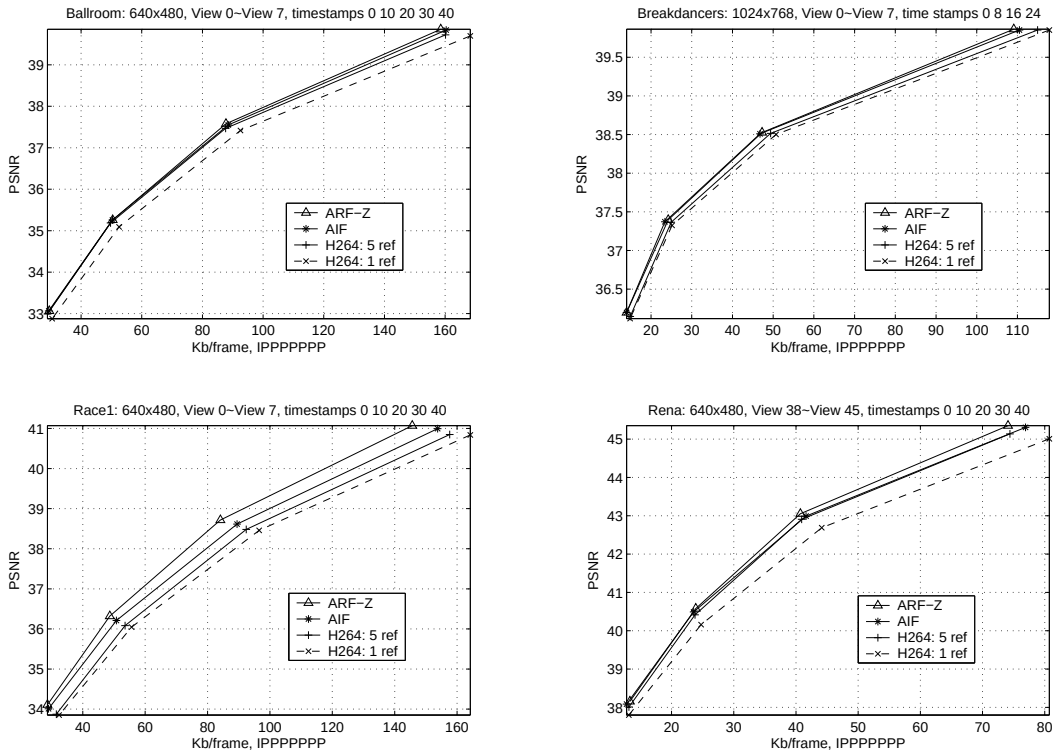

Figure 3.15: Comparison of different techniques applied to inter-view coding

Cameras for the *Breakdancers* sequence are arranged on an arc. View 4 in this sequence is somewhat blurred while all the other views have very a similar subjective visual quality. From the simulation results, we see that adaptive filtering can be used to provide better reference for predictive coding. Both the proposed method and AIF achieve higher coding efficiency than the multiple references method in H.264/AVC. Our ARF method designed based on scene depth, provides a marginal gain over AIF. However, it is worth noting that the achievable gain with these enhanced predictive codings is relatively modest as compared to that achievable with other multiview sequences in Fig. 3.15. It is because the frames in the *Breakdancers* sequence have large homogeneous areas for which intra coding was selected. Fig. 3.4 provides an example of the disparity-based

frame partition and intra-coded areas during the initial disparity estimation. Due to the relatively large number of intra-coded blocks, our proposed method provides a 0.2 dB gain at 100Kb/frame over H.264/AVC with one reference.

Race1 is the sequence that suffers from most severe focus mismatch across views. As a result, the benefit of adaptive filtering is more prominent: AIF provides a 0.3 dB gain over multiple reference H.264/AVC, and the proposed ARF achieves an additional 0.2 $\sim$ 0.3 dB gain over AIF (0.7$\sim$ 0.8 dB over H.264/AVC with one reference). Focus mismatch is more efficiently compensated with our depth dependent adaptive filtering. As for the *Rena* sequence, in the presence of some degree of inter-view discrepancy, our ARF method again provides a 0.15 dB gain over AIF (0.6 dB gain with respect to H.264/AVC with one reference). In this sequence, due to a much closer spacing of the cameras as compared to other test sequences (5cm versus 20cm), the standard multiple reference method achieves coding efficiency similar to that of AIF.

Based on the simulation results, it can be concluded that our proposed method is especially helpful for disparity compensation with focus mismatch. It effectively designs filters as estimators of mismatch kernels to compensate for the possible discrepancies associated with scene depth. For sequences with stronger focus mismatch, it provides greater coding gains over single reference AIF and the multiple reference method without adaptive filtering.

Inter-view prediction in MVC: B-frames

The proposed ARF with support of bi-directional prediction (Section 3.4) is integrated with JMVM 5.0, which is a reference software from JVT dedicated for multiview video

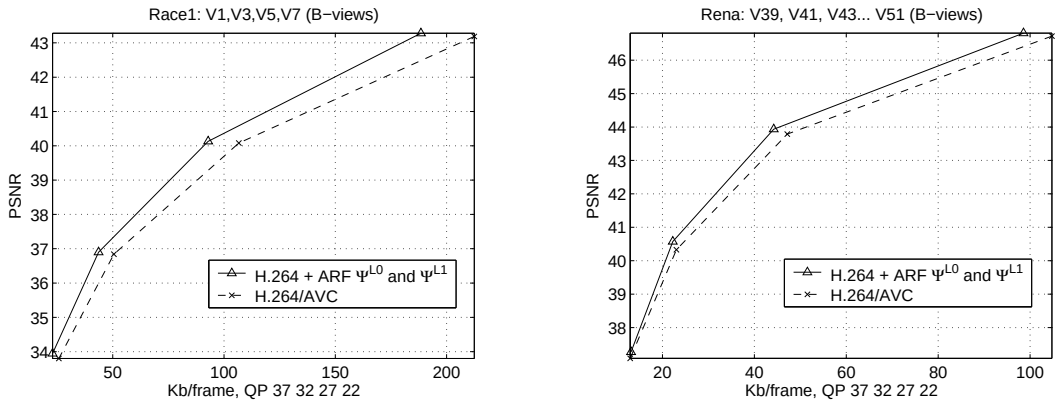

Figure 3.16: Rate-distortion performance of the proposed ARF

coding based on H.264/AVC. We partition a frame into up to three depth-levels and estimate the corresponding filters. Filters of size 5×5 with *circular symmetry* are used (ψ_{55cir} as equation (3.8)). We encode frames only at given timestamps using inter-view coding with IBPBP structure. The interval between two timestamps is 0.5 sec. (e.g. Inter-view coding at every 12th frame for frame rate 25fps, no temporal prediction.)

Without making any modification to the bi-predictive search schemes, we performed simulations based on **Method A** and **Method C** in Table 3.1 using iterative search. (Initial search range ±64, plus 4 iterations with refinement search range ±8.) For the sequences tested, the two-set filter design **Method C** achieves higher coding efficiency than joint filter design **Method A**. Thus in Fig. 3.16, we provide the corresponding RD results of the two-set filter design approach. The four rate points (from low to high) were obtained with QP = 37, 32, 27, and 22. It can be seen that, for views encoded with bi-directional prediction (inter-view B-frames), a 0.6∼0.8 *dB* gain is achieved on *Race1* when applying the proposed two-set ARF design, while the improvement is about 0.3∼0.4 *dB* for *Rena*.

We also tested the fast bi-directional search method proposed for filter design **Method C**, in which the search over multiple filtered references is only performed within one of the lists, and constrained search will then be performed for the other list based on the filter selection in the first list. Since there are two lists, this procedure can obviously be carried out in two different orders: Multiple reference search in List 0 followed by constrained search in List 1, or vice versa. The multiple search should be performed on the list in which blocks are more likely to select the correct filter, i.e., in the list where choosing different filters will results in greater differences in prediction error, such that the filter selection is more reliable to serve as the constraint for the search in the other list. From the analysis in Section 2.3 and Fig. 2.9, across different disparity values (depths), filter responses resulting from smaller focus setting mismatches have greater differences as compared to those resulting from larger focus setting mismatches. However, these filters have higher $\pm 3dB$ frequencies (i.e., smaller setting mismatch, less impact on the images). For natural images, energy is mostly concentrated in the low frequency components. As a result, although there are larger differences in filter responses, they may not corresponds to larger differences when calculating prediction error. On the other hand, larger focus setting mismatch leads to filters that have stronger effect on the images. But at different depths the difference between filter responses is not as significant as in the case of filters resulting from smaller setting mismatch. Thus, without actually knowing the exact setting difference, we conduct simulations for both search orders. The results on different views of *Race1* are shown in Fig. 3.17. The two constrained search schemes lead to about $0.1 \sim 0.2$ dB degradation as compared to multiple reference search in both lists.

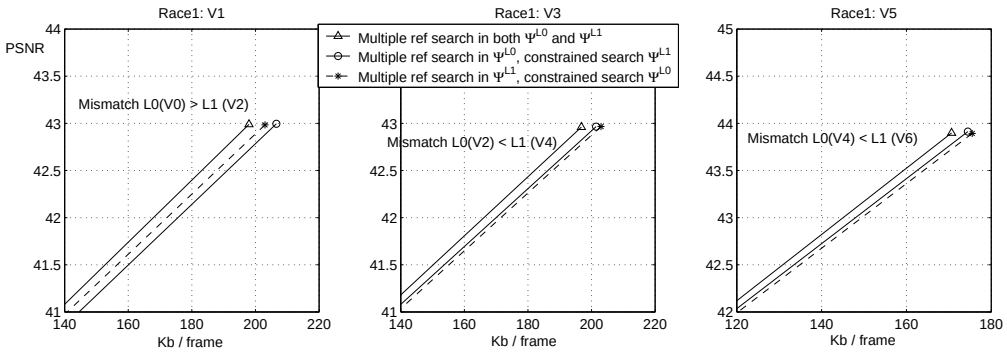

Figure 3.17: Performance of constrained fast search for ARF-B

Temporal prediction in monoscopic video

In Fig. 3.18, we first provide the RD coding results for a monoscopic sequence with strong localized focus changes. From the test sequence *fondue-multi*, we excerpt and encode frames $126 \sim 170$ during which the camera focus is changing back and forth rapidly among people at different scene depths. It can be seen that the proposed adaptive filtering approach ARF-f_{mb} provides about 1 dB gain over H.264/AVC with 1 reference, 0.5 dB gain over H.264/AVC with 5 references, and around $0.2 \sim 0.3$ dB gain as compared to AIF. The results demonstrate that, when video undergoes local focus changes, adaptive filtering can be used to provide better reference for predictive coding. The proposed method and AIF both achieve higher coding efficiency than the multiple reference method in H.264/AVC. Our approach is more effective in modeling the localized focus changes, with multiple versions of filtered references generated for motion compensation.

As a reference, we provide another set of simulations results on the sequence Raven, which exhibits *global motion blur*: At different timestamps, the entire frame may be blurred due to camera movement. It can be seen that for this particular case, the two adaptive filtering approaches, ARF and AIF, achieve almost identical coding performance:

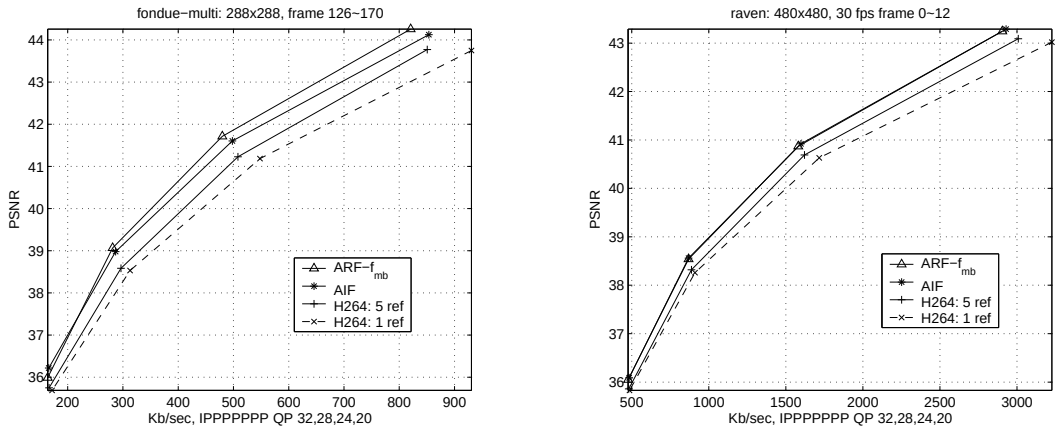

Figure 3.18: Rate-Distortion comparison of different approaches

about 0.3/0.6 dB gain over H.264 with 5/1 references. This result indicates that the proposed ARF method provides more gain than AIF only when the blurring mismatch is localized. In other video sequences extracted from movie trailers containing some degree of focus mismatch, we also achieved $0.4 \sim 0.7$ dB gain over H.264/AVC with 1 reference. The gain is much smaller when applying the proposed method to regular sequences with no focus changes. Thus, in a practical system, it would be desirable to develop tools to detect the existence of focus mismatch (e.g., applying appropriate criteria to residual blocks), so that mismatch compensation is only explored when a potential coding gain can be achieved.

3.6 Complexity Analysis

There are two main factors in our proposed ARF approach that will increase encoding complexity. One is the process of estimating mismatch kernels, which includes initial

motion/disparity search, classification of disparity vectors or coefficients of f_{mb}, calculating filter coefficients, and generating filtered references. This will be discussed in detail in Section 3.6.1. The other factor which introduces additional computation is the motion/disparity search loop over multiple filtered references. Note that filters in AIF are applied to different subpixel positions at a single reference frame. When conducting simulations, we measured the motion/disparity estimation time with a profiling tool provided in JM 10.2 software [42]. When full search is applied, the motion/disparity estimation time in our method (including the initial and final search) is similar to that of H.264/AVC with 5 references, and is about 2.5 times as long as in AIF (note that AIF also involves an initial search). However, unlike the multiple reference method in which references are from different views or different timestamps, thus having different disparity/motion, the references in our system are simply different filtered versions of the *same* frame. Taking this into account, significant complexity reduction could be achieved by reusing motion/disparity information: As we proceed from the unfiltered reference (as in the initial search) to the filtered ones (final search), a much smaller search range could be applied based on previously computed motion/disparity. We performed simulations by changing the final search range from ± 64 to ± 4 using the vectors obtained from the initial search as predictors. The R-D degradation observed is negligible, while the total encoding time is reduced to about $\frac{1}{4}$ [23].

In what follows, we will focus on the complexity associated with the filter design.

3.6.1 Complexity of ARF Filter Design

The ARF filter design process can be decomposed into the following three parts: (a) classification of block-wise parameters (disparity vectors or coefficients of f_{mb}) for frame partition, (b) calculation of the filter coefficients, and (c) generation of filtered references.

In the frame partition process (Section 3.3.1), with the maximum number of Gaussian components set to K, classification is performed based on EM algorithm for $k = K$, K-1, ... 1 [4], to cluster the block-wise features. The k which provides the lowest minimum description length MDL, denoted as k', will be selected to build the final model. The complexity of this unsupervised classification is proportional to the *maximum number* K, and the *number of input elements* to be classified. To speed up this process, one can consider setting a smaller K or performing classification with sub-sampled vectors and classifying the corresponding blocks. Since the value K determines the maximum possible number of depth levels in the classification results, ideally it should be set as sequence-dependent by observing the scene of each multiview sequence. In our ARF approach, we do not assume any prior knowledge about the scene, and let the classification tool decide the number of classes (unsupervised), with a proper upper bound K to start with. We have observed that for *Ballroom*, forcing the classification tool with a reduced K (from 4 to 3 to 2) led to coding degradation that is not negligible. Another possible approach to reduce classification complexity is to use less elements for classification. For example, in MVC inter-view prediction, we can sub-sample the disparity field. The possible degradation due to the sub-sampling method will only be significant on detailed

object boundaries, where more disparity vectors on smaller block size have to be used to differentiate disparity values.

To analyze the complexity of filter calculation, let us denote C the number of distinct filter coefficients in a given filter (e.g., 6 coefficients for ψ_{55cir}), and P_{D_k} the number of pixels in depth class D_k. As shown by (3.6), constructing the Wiener-Hopf equation for one coefficient in one depth class requires $P_{D_k}C+C$ (left side) + P_{D_k} (right side) addition / multiplication operations to calculate the sum of products. Thus, for all filters, each with C coefficients, the total number of operations will be $\sum_k C\,(P_{D_k}C + C + P_{D_k})$. This value is upper bounded by $C\,(PC + C + P) \approx PC(C+1)$, as intra coded blocks will not be assigned to any depth class (i.e. $\sum_k P_{D_k} \leq P$). Solving the linear system for each filter with a set of Wiener-Hopf equations (3.6) requires addition / multiplication operations in the order of C^2 (1st-order linear system with C equations and C unknowns). Thus, designing k' filters will require $k'C^2$ operations. For a typical example with $k' = 4$, $C = 6$ (Section 3.3.2), and $P = 640 \times 480$, the value $k'C^2$ is relatively small as compared to the previous term $PC(C+1)$. As a result, the total number of operations in (b) can be approximated by $PC(C+1)$. For the example filter with $C = 6$, the result is $P \times 42$. Note that this will be similar to the complexity of applying different interpolation filters to generate frame data at all subpixel positions for P pixels, as specified by H.264/AVC (which can be estimated to be $P \times 66$ [48]).

Finally, to generate filtered references, the complexity of the convolution depends on the structure of filters: To generate one filtered pixel, the number of multiplications is equal to the number of distinct filter coefficients, and the number of additions is equal to the number of elements in the filter minus one. For ψ_{cir55} which has 6 coefficients, there

will be 6 multiplications and $5 \times 5 - 1 = 24$ additions to generate one pixel. Thus the total number of operations is $k'P \times 30$, which will be $P \times 120$ when $k' = 4$ or $P \times 90$ when $k' = 3$. Again the complexity of this step is similar to the calculation of sub-pixel values with interpolation filters as in H.264/AVC.

We measure the execution time by performing ARF encoding in three steps: (i) Initial motion/disparity search, (ii) frame partition, filter estimation, and generating filtered references and (iii) final encoding with filtered references. On average, without any complexity reduction method, the processes in (ii) together lead to an increase of about 25% in execution time as compared to a H.264/AVC coding process with one reference, ± 64 search range.

3.7 Conclusions

We have proposed an adaptive filtering approach for encoding video content exhibiting localized focus mismatch in different regions within a frame. Our approach first performs an initial motion/disparity estimation to obtain motion/disparity information and establish block correspondence. For inter-view prediction in MVC, disparity vectors are exploited to identify regions suffering from different types of focus mismatch. Blocks with similar disparity vectors are grouped into classes (scene-depth levels). As for monoscopic video with focus changes, we first capture the local variation of focus changes by estimating MB-wise filters. MBs with similar filters are then grouped together and associated with adaptive filters to be designed. In both cases, EM classification algorithm with GMM basis, which consider directional variations, is applied. It automatically decides the number

of class based on MDL criterion. Based on the classification result, an filter which is an estimator of focus mismatch kernel, is constructed for each class by minimizing prediction error energy. Such filter design approach is adaptive to the focus changes between the current frame and the reference frame. Filtered references are generated by applying the estimated filters. For the sequences we tested, the proposed method provides higher coding efficiency as compared to the current H.264/AVC with multiple reference frames and other adaptive filtering approaches such as AIF. Larger coding gain is achieved for sequences with stronger localized focus mismatch.

We also extend the ARF method to MVC inter-view bi-directional prediction with focus mismatch. We show that the filter design approach for the averaged bi-predictor leads to a suboptimal solution when combined with conventional bi-predictive search schemes. Taking into account the interaction between filter design and the bi-predictive search with filtered references, we proposed a filter estimation method which designs a set depth-related filters for *each* reference list. Simulation results show that for views coded with inter-view bi-directional prediction, the proposed method provides up to 0.8 dB gain over current H.264/AVC in the sequences we tested. We evaluate the efficiency of constrained search within one of the list based on the filter selection in the other list. The degradation as compared to performing multiple search on both lists is about 0.1 dB.

Chapter 4

Computationally Efficient ARF for MVC Inter-view Coding Based on Rate-distortion Prediction and Filter Sharing

4.1 Motivation

The gain in coding efficiency from the ARF approach we just described in Chapter 3 comes at the expense of higher encoding complexity, in particular because this is a *two-pass encoding scheme* (initial disparity estimation, frame partition, filter estimation, and then final encoding with filtered references). Even though the computation during the final disparity estimation can be significantly reduced, without sacrificing coding efficiency, by reusing the disparity information obtained from the initial disparity estimation [23], the other components can still introduce significant complexity (Section 3.6.1).

Without any prior knowledge about the mismatch, the initial search and filter estimation are necessary in order to adaptively design filters. However, for inter-view prediction in MVC, there are certain multiview characteristics that can be exploited to help us apply ARF in a more efficient manner. In Chapter 2, we have demonstrated that focus mismatch in multiview systems is a function of the focus setting difference (view-dependency) and

the object depths (depth-dependency). Assume that during the multiview video capturing process, the cameras being used, the spacing and the relative shooting orientations between cameras, as well as the focus settings (parameters a, f and d), are time invariant (we will refer to this as a "time-invariant multiview setting"). Then the optical transfer function (OTF) of each view will also be time-invariant. The type of focus mismatch will depend on which view pair is being considered, and will also depend on the scene being captured. For a pair of views with larger focus setting mismatch (larger difference in their β curves as functions of disparity/depth), the depth-dependent discrepancy will also be stronger, leading to lower coding efficiency in inter-view prediction and potentially higher coding gain if applying ARF. As for a view pair with no focus setting difference, theoretically the images will only be affected by disparity (as described by (2.25)). The potential benefit of ARF would be limited since there is no depth-dependent mismatch to address.

The multiview test sequences used by the JVT-MVC group [18] were captured under a fixed (time-invariant) multiview setting, i.e., there are no camera adjustments while the video is being captured. It has been reported [17] that these sequences exhibit inter-view mismatches that are strongly view dependent. Correspondingly, in MVC inter-view coding, we have observed that the coding gain achieved by our ARF method, varies significantly among different multiview sequences. Figures 4.1 and 4.2 illustrate coding performance of ARF for different views of *Ballroom* and *Race1*. For the different QP tested, View 1 of *Ballroom* has about 0.5 *dB* gain when ARF is applied. However the coding gain in View 4 is significant only at high bitrates. For views that exhibit strong focus mismatch with respect to the views used for prediction, for example View 4 of *Race1*,

applying ARF provides significant coding gain (greater than 1 dB for tested QP settings). On the other hand, for a view with no perceivable focus mismatch as compared to its reference view, encoding using ARF leads to very limited (or virtually no) improvement in coding efficiency. Considering the additional complexity due to ARF, it is desirable to develop prediction methods to determine whether ARF is beneficial for a given view, such that we can avoid applying ARF to views that would not achieve much coding gain.

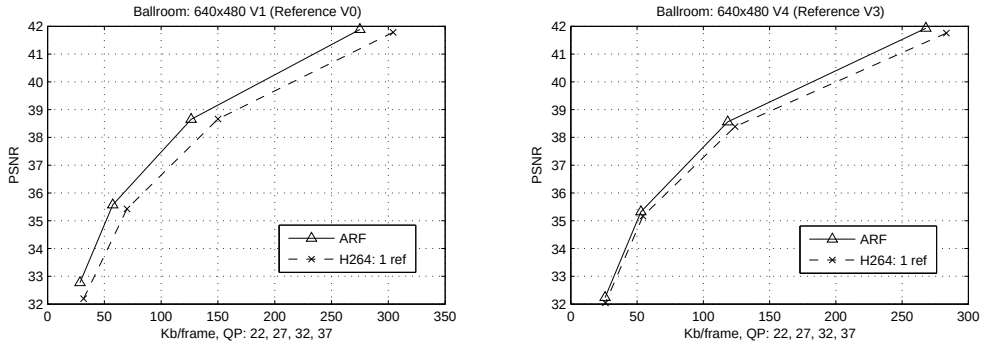

Figure 4.1: ARF performance in different views of *Ballroom*

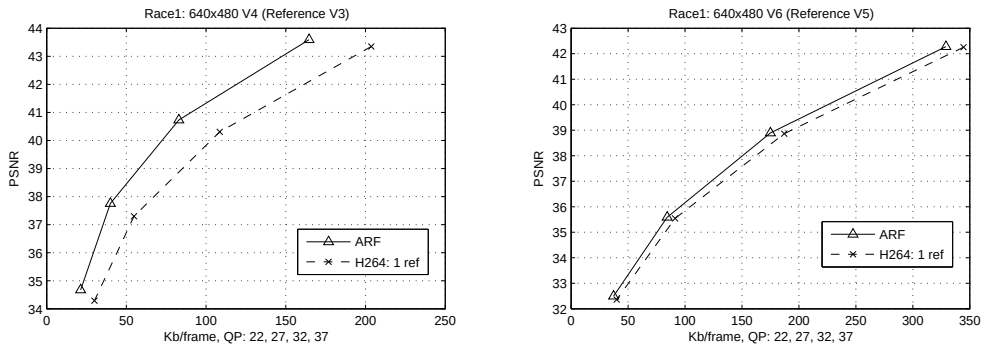

Figure 4.2: ARF performance in different views of *Race1*

Furthermore, since the OTFs are time-invariant, for a given pair of views, the mismatch associated with an object at depth Z will be the same at different times. Thus,

across time, when the captured scene is composed of similar levels of depths, the mismatches present in the images will also be alike, leading to similarity in the ARF filters. It is observed that for a given view, the estimated filters at different timestamps tend to be very similar. Thus, instead of estimating ARF filters at every timestamp, a set of filters can be re-used during a certain time interval until there is significant change in depth-composition within the scene. With time-invariant camera spacing (fixed b in (2.23)), a given depth Z will correspond to the same disparity δ_Z even for images captured at different timestamps. Thus to determine changes in depth-composition, we can compare the distribution of block-wise disparity vectors (DVs) at different timestamps. A more efficient filter estimation/updating scheme can be developed by exploiting this property.

In this chapter, we analyze the performance of ARF in MVC inter-view prediction. Besides the variation in gains for different multiview sequences reported in Section 3.5, we further investigate the variations across different views of a given multivew sequence, and across different times for a given view. We show that the gains in coding efficiency can exhibit significant differences from view to view. Furthermore, the estimated filter coefficients at different timestamps demonstrate strong correlation when the depths of objects in the scene remain similar. By exploiting the properties derived in Section 2.3 and from the performance analysis in Section 4.2, we propose two techniques to design an efficient ARF coding scheme, which maintains coding efficiency with significantly reduced complexity: i) *view-wise ARF adaptation* based on RD-cost prediction, which determines whether ARF is beneficial for a given view, and ii) *filter updating based on depth-composition change*, in which the same set of filters will be used (i.e., no new filters

will be designed) until there is significant change in the depth-composition within the scene. Simulation results in Section 4.3 will show that significant complexity savings are possible (e.g., the complete two-pass ARF encoding process needs to be applied to only 20% ∼35% of the frames) with negligible quality degradation (e.g., around 0.05 dB loss).

4.2 Computationally Efficient ARF for Inter-view Coding

4.2.1 Rate-distortion Analysis and View-wise ARF Adaption

In state of the art video coding techniques, high coding efficiency is achieved by rate-distortion optimization. For each macroblock (MB), the coding mode which provides the lowest rate-distortion cost (RD-cost) will be selected:

$$\text{RD-cost}_{MB} = \min_{mode} \left(D_{mode} + \lambda R_{mode} \right), \tag{4.1}$$

where D_{mode} is the distortion between the original MB and the reconstructed MB using a given mode, R_{mode} is the bitrate required to encode that mode, and λ is the Lagrange multiplier. In video coding schemes such as H.264, λ is often chosen as a monotonic function of QP [43, 53]: larger QP (low bitrate scenario) results in larger λ to put greater penalty on rate. To analyze the performance of ARF, we record the frame-wise R-D cost in the initial disparity estimation (with only unfiltered reference) and in the final encoding (with unfiltered and multiple filtered references). The frame-wise RD-cost is calculated by aggregating MB RD-costs as in (4.1) estimated during MB mode decision. We apply ARF to MVC inter-view coding with IPPPPPPP structure. For multiview data with frame rate 24 fps, inter-view coding is performed only at every 12th

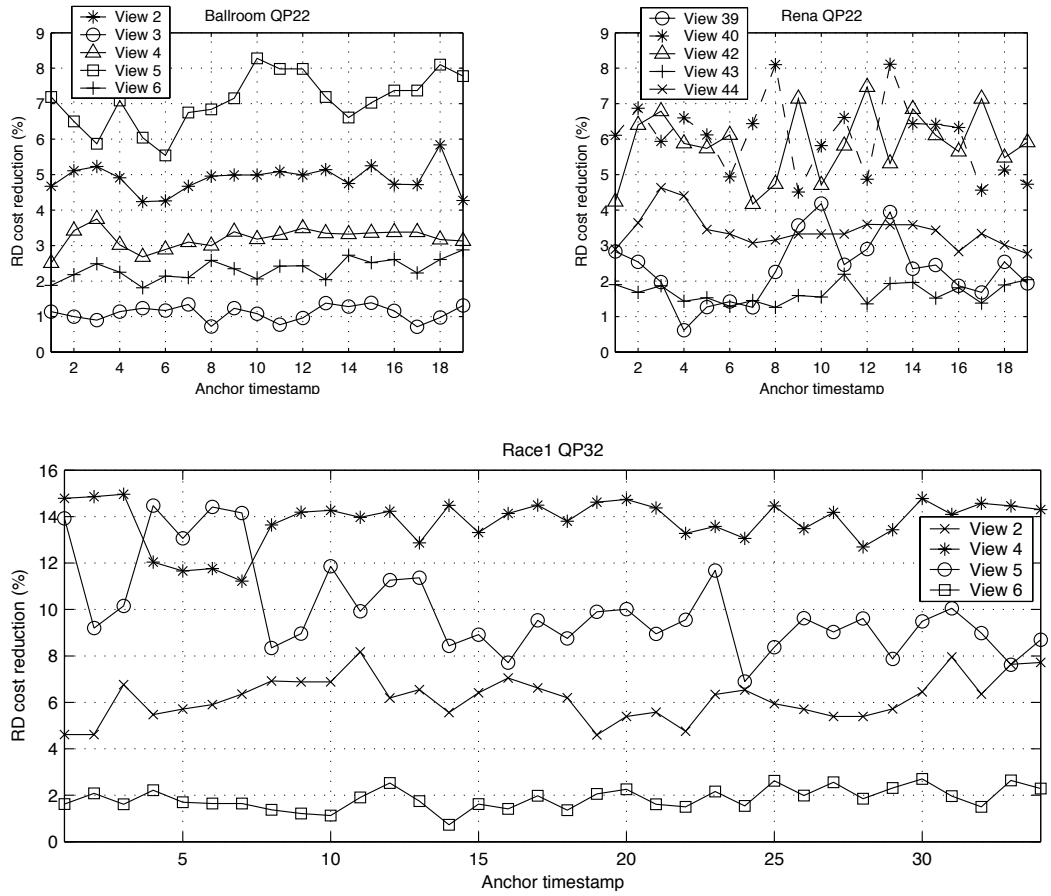

Figure 4.3: Frame-wise RD-cost reduction provided by ARF

frame: 0, 12, 24.....; for data with 30 fps, inter-view coding is performed at every 15th frame: 0, 15, 30..... Thus these timestamps correspond to a half second interval, and we will call them "anchor timestamps" 0, 1, 2... etc. Fig. 4.3 provides a comparison between the frame-wise RD-cost in the initial and final disparity estimation for different views across anchor timestamps.

According to the analytical results in Chapter 2, view pairs with no focus setting differences will not produce depth-dependent mismatches which ARF is designed to compensate for. However, when there exists a focus setting difference, the type and degree of

mismatch will depend on the depths of objects appearing in the scene. This will result in variations in ARF performance. The RD-cost reduction in Fig. 4.3 shows behavior that is consistent with the analytical results. First, the RD-cost reduction achieved by our ARF approach varies significantly from view to view. These variations are consistent with the reported mismatches in multiview test sequences [17]: For views that exhibit strong focus mismatch with respect to the views used for prediction, for example Views 4 and 5 of *Race1*, applying ARF can provide more than 10% reduction in RD-cost. On the other hand, for a view with no perceivable focus mismatch as compared to its reference view, encoding using ARF leads to very limited improvement in coding efficiency. Second, views showing higher gains with ARF (i.e., exhibiting focus mismatch) tend to have larger variations in RD-cost reductions across different timestamps, due to the change in depth-composition. Note that while Fig. 4.3 only depicts results at a given QP for each sequence, the same behavior (variations cross view/time) is observed for all three sequences across QP 22, 27, 32, and 37. However it is worth mentioning that the RD-cost reductions become smaller as QP increases (low-bitrate scenario).

From the analytical results in Section 2.3 and observations above, if for a given view the RD-cost reduction achieved by using ARF is *consistently very small* over multiple anchor timestamps, it is reasonable to consider not applying ARF for future anchor frames. We propose a predictive ARF adaptation method such that, within a period of N anchor timestamps, the encoder evaluates RD-cost reduction provided by ARF in the first T anchor timestamps, and determines whether to apply ARF to the remaining anchor timestamps. In the next period of N anchors, ARF will be tested *again* to determine whether it will be efficient. Let $\mu^V_{(1,T)}$ denote the average RD-cost reduction over anchor

timestamps 1 to T (where $T \geq 1$) in view V when applying ARF, and $\sigma^V_{(1,T)}$ denote the corresponding standard deviation. This ARF adaptation method can be summarized as:

Algorithm 1 ARF adaptation for a given view

```
Divide anchor frames into groups of N anchor frames
for Each group of N anchor frames do
  for 1 ≤ i ≤ T, i.e. the first T frames in each group do
    Apply two-pass ARF
  end for
  Calculate μ^V_(1,T) and σ^V_(1,T)
  if μ^V_(1,T) < κ and σ^V_(1,T) < ε then
    for T + 1 ≤ i ≤ N do
      Conventional encoding (No ARF for the following frames)
    end for
  else
    for T + 1 ≤ i ≤ N do
      Apply two-pass ARF
    end for
  end if
end for
```

We call this approach View-wise ARF adaptation based on RD-cost prediction. In Section 4.3, simulation results will be provided with selected settings of N, T, κ, and ϵ.

4.2.2 Filters Correlation and Filter Updating using Depth Composition

From the analysis in Section 2.3, if multiview setting is time-invariant, the type of mismatch between frames from a given pair of views will depend on the depth-composition within the captured scene. Parts of the scene at depth Z will produce the same type of mismatch at different capturing timestamps, leading to similarity in the corresponding estimated ARF filters. To further investigate how filters vary over time, we perform correlation analysis: For a given view, we concatenate filter coefficients from filters estimated at anchor timestamp t_1 to form a coefficient vector $\mathbf{A}_{t_1}$, and compared it with

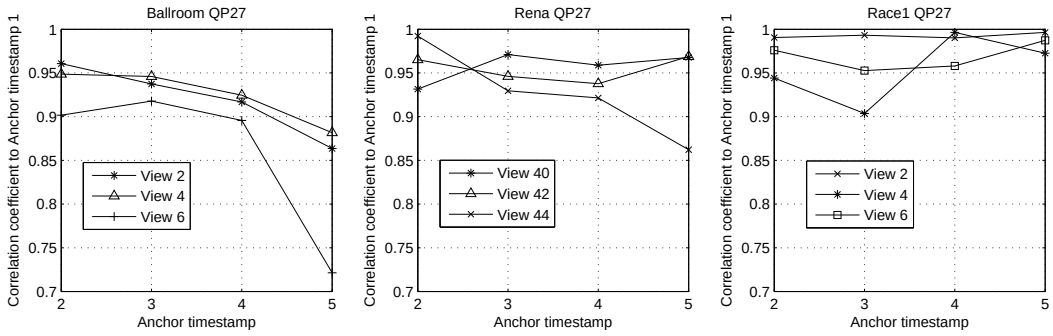

Figure 4.4: Correlation of estimated filter coefficients at different timestamps

the corresponding filter coefficient vector $\mathbf{A}_{t_i}$ at another anchor timestamp t_i. Fig. 4.4 provides results for some of the analyzed sequences.

For *Ballroom*, there are multiple dancing couples moving in the scene. A couple may appear in some frames at depth Z_0, while in the preceding frames there is nothing at this particular depth. Due to such depth-composition difference, the filter correlation has larger variation as compared to the other sequences tested. On the other hand, in *Rena*, the scene composition is consistent across time: A foreground girl and the background remain at same distances to the camera for the entire sequence. The estimated filter coefficients demonstrate very high correlation even over 14 anchor timestamps (about 200 frames). The most interesting case is *Race1*. In this sequence, the *content* in the scene is changing due to the viewing angle shift of the camera-set and the carts driving along the runway. However, frames at different timestamps mostly cover the same range of *depth* (Refer to Fig. 4.7). That is to say, at different timestamps, there is no new "depth level" being introduced (as compared to the *Ballroom* case). As a result, filters still demonstrate very strong similarity. These results are consistent with the depth-dependency property derived in Chapter 2.

When the filters are highly correlated over a certain time interval, e.g., when *depth-composition* within the scene remains similar, it is not necessary to estimate them at every single timestamp. Applying the same set of filters over multiple timestamps will reduce the effort spent on initial disparity search and filter estimation, while the coding efficiency could be preserved. Moreover, when filters are re-used across time, we do not need to transmit filter coefficients. Our analysis suggests that the time intervals during which the filters are re-used and when to re-estimate/update filter coefficients, can be adaptively determined by comparing the *depth-composition* at different timestamps. In ARF, disparity vectors are exploited to partition frames into depth-levels by performing classification based on GMM. To determine whether there has been a change in depth-composition, we compare the GMM classification results at different timestamps. Let $\mu_{V,i}^{GMM}(m)$ denote the mean of Gaussion component m in the DV classification for frame i in view V, and let $P_{V,i}^{GMM}(m)$ be the corresponding percentage of blocks being classified into that class. A Gaussian component is defined as "not being covered", in a reference timestamp r, if its mean is at least D pixels away from any Gaussian mean at timestamp r, $\mu_{V,r}^{GMM}(n)$. If the sum of the $P_{V,i}^{GMM}(m)$ of all these "not covered" components is over a certain percentage P, we determine there is a "change in depth composition" and thus apply the two-pass ARF coding for the current frame to update filters, otherwise the filters estimated at the reference timestamp will be re-used.

$$W_{V,i}(m) = \begin{cases} 1, & \text{if } \forall n \ |\mu_{V,i}^{GMM}(m) - \mu_{V,r}^{GMM}(n)| > D \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$\text{If} \quad \sum_{m} W_{V,i}(m) \cdot P_{V,i}^{GMM}(m) > P \quad \rightarrow \quad \text{Apply two-pass ARF}$$
$$\text{Otherwise} \quad \rightarrow \quad \text{Re-use filters.} \tag{4.3}$$

However, there is an issue preventing us from directly using the above scheme: For the current frame being considered, at the point when making the decision to update filters or not, we actually do not have its disparity information yet. Disparity estimation should be performed only after we decided to apply two-pass ARF, otherwise filter will be re-used and initial estimation is skipped. To overcome this problem, we refer to a view in the earlier coding order for GMM disparity information. For example, when encoding View 2, the DV GMM for frames in View 1 are exploited to determine whether there has been a change in depth-composition, and then to decide whether to update the filters. (Note that in this scheme, we cannot apply filter re-use to the first view being inter-view coded, as there will be no reference disparity.) This method would be most suitable for 1D parallel camera arrangements with equal spacing among cameras, as the disparity values of different views at same timestamp should be very similar in this scenario. We denote such method as *filter updating based on depth-composition change*. Fig. 4.5 shows some filter re-use results when we set $D = 5$ pixels and $P = 15\%$ in (4.2)(4.3). It can be seem that the coding efficiency (reduction in RD-cost) can be well-preserved while the no new filter are estimated. Note that since the RD-cost is calculated by accumulating MB-wise RD-cost, the potential bit saving achieved by not sending filter coefficients is not reflected in Fig. 4.5.

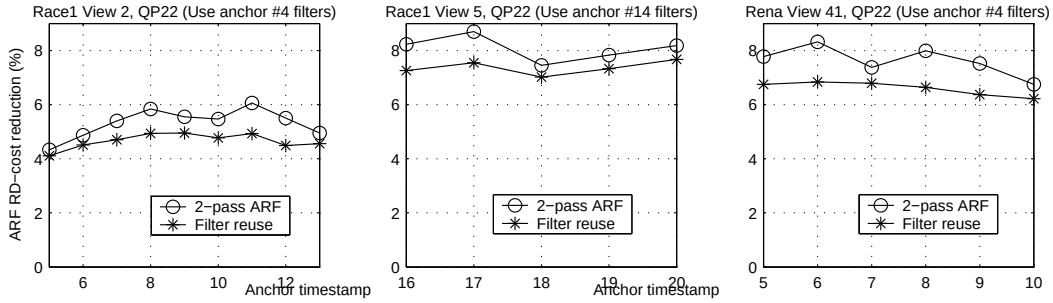

Figure 4.5: Examples of RD performance when filters are re-used

Combining with the ARF adaptation described in Section 4.2.1, an efficient ARF coding scheme is summarized in Algorithm 2. In this new scheme, within a period of N anchor timestamps, after evaluating ARF for the first T anchor timestamps, there are three possible encoding options for the remaining frames. (i) If the encoder decides not to use ARF, the remaining frames will be encoded normally, followed by a GMM classification on the DVs to provide disparity information for the next view. If it instead decides to apply ARF, (4.2) and (4.3) will be used to choose between (ii) simply re-using filters or (iii) performing the two-pass ARF to update filters. Frames encoded by re-using filters do not need to undergo initial disparity search and filter estimation, but simply perform disparity search with filtered references. Classification on DV will then be applied to generate disparity information for the next view.

4.3 Simulation results

We conduct simulations with the proposed efficient ARF techniques. As described in Section 4.2.1, IPPPPPPP inter-view coding is performed at anchor timestamps with half-second time interval between two timestamps. We implemented the ARF coding

Algorithm 2 Efficient ARF coding scheme for MVC inter-view prediction (V total number of views: View $0 \sim$ View $V-1$. Inter-view coding performed on anchor frames for View $1 \sim$ View $V-1$.)

for $1 \leq v \leq V-1$ **do**
  Divide anchor frames into groups of N anchor frames
  for Each group of N anchor frames **do**
    for $1 \leq i \leq T$, i.e. the first T frames in each group **do**
      Apply two-pass ARF: Initial disparity estimation, classification, filter estimation, final encoding
    end for
    Calculate $\mu^V_{(1,T)}$ and $\sigma^V_{(1,T)}$
    if $\mu^V_{(1,T)} < \kappa$ and $\sigma^V_{(1,T)} < \epsilon$ **then**
      for $T+1 \leq i \leq N$ **do**
        Conventional encoding (No ARF for the following frames)
        GMM classification for disparity vectors
      end for
    else
      if v=1, i.e. the first inter-view coded view **then**
        for $T+1 \leq i \leq N$ **do**
          Apply two-pass ARF to update filters
        end for
      else
        Filter reference timestamp $r = T$
        for $T+1 \leq i \leq N$ **do**
          Calculate $W_{V-1,i}(m)$ as in (4.2)
          if $\sum_m W_{V-1,i}(m) \cdot P^{GMM}_{V-1,i}(m) > P$ **then**
            Apply two-pass ARF to update filters
            Filter reference timestamp $r = i$
          else
            Re-use the filters at timestamp r
            Encode with filtered reference
            GMM classification for disparity vectors
          end if
        end for
      end if
    end if
  end for
end for

scheme on top of the H.264/AVC framework using reference software JMVM 5.0. We set $N = 20$ and $T = 4$, i.e., for a 10 second period (20 anchor timestamps), anchor timestamps in the first 2 seconds will be encoded with ARF to evaluate the RD-cost reduction. To set the thresholds κ and ϵ, we observed ARF performance for sequences with virtually no perceivable focus mismatch, thus having very limited improvement, such as *Exit* and *Uli*. The achieved RD-cost reductions for these sequences are in the range of $0\% \sim 2\%$. Thus we set $\kappa = 2\%$ and $\epsilon = 1\%$: The encoder will disable ARF coding if it observes that the average RD-cost reduction over the first 4 anchor frames is less than 2% with a variation less than 1%.

For the remaining frames which require filtering, (4.2) and (4.3) are used to determine whether filters will be re-used or updated (thus performing the entire two-pass ARF). We tested different values of parameters D and P, which resulted in differences in how often filters are updated. For example for views in *Ballroom*, with $D = 5$ pixels, when changing P from 15% to 10% to 5%, over a period of 20 anchor timestamps, the frequency at which the filters are updated will increase from twice to four times to five / six times. Updating filters less frequently leads to larger degradation in coding efficiency as compared to performing two-pass ARF at every anchor timestamp. In Fig. 4.6, we present encoding result with $D = 5$ pixels and P =15%, for which the filters are only updated at most twice[1] over the timestamps tested, as indicated in Table 4.1, 4.2, and 4.3.

It can be seen from the results that the proposed techniques are very efficient in preserving ARF coding gains while the complexity is significantly reduced. After zooming in on the RD curves, we observe a degradation of less than $0.05dB$ as compared to the

[1]Excluding the initial ARF testing period, i.e., the first 4 anchors for each 20-anchor period.

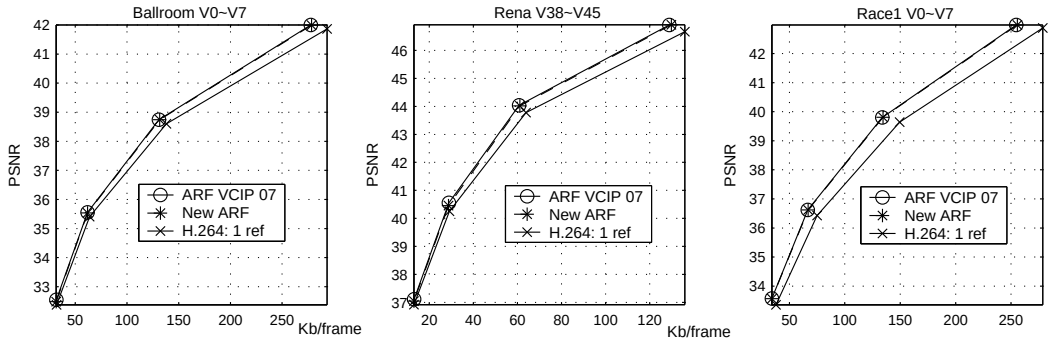

Figure 4.6: Encoding results of the proposed coding scheme (QP = 37, 32, 27, 22)

Table 4.1: Encoding selection of the proposed efficient ARF: *Ballroom* (20 anchors each view. The first four, a1 $\sim$ a4, are encoded with two-pass ARF. If it is determined that the remaining anchors also need filtering, the anchors at which filters are updated are also listed in the table, and the other anchors will be encoded by re-using the filters.)

		Ballroom (20 anchors)						
QP	**Scenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**
22	Filtering for a5 $\sim$ a20? (Update filter at)	Yes (All)	Yes (a5, a10)	No	Yes (a5, a10)	Yes (a5, a10)	Yes (a6, a10)	No
27	Filtering for a5 $\sim$ a20? (Update filter at)	Yes (All)	Yes (a5, a8)	No	Yes (a5, a8)	Yes (a5, a8)	Yes (a5, a8)	No
32	Filtering for a5 $\sim$ a20? (Update filter at)	Yes (All)	Yes (a5, a8)	No	No	Yes (a5, a9)	No	No
37	Filtering for a5 $\sim$ a20? (Update filter at)	Yes (All)	No	No	No	No	No	No

previously proposed ARF results. (Note that the degradation will be even smaller when the filters are updated more frequently than in the example here, e.g. when $P < 15\%$.) Across different QPs, the view-wise ARF adaptation method successfully identifies views for which limited coding gain would be achieved if ARF were applied. With higher QP, ARF is applied to fewer views since the achievable gains tend to be smaller under the low bitrate scenario. For example in Table 4.1 *Ballroom* View 4, ARF is applied only to QP 22 and 27. This behavior matches well with the coding results provided in Fig. 4.1. As

Table 4.2: Encoding selection of the proposed efficient ARF: *Rena* (20 anchors each view. For views that require filtering, since the depth composition remain unchanged, the filters are re-used throughout the remaining anchors.)

		Rena (20 anchors)						
QP	**Scenario**	**V39**	**V40**	**V41**	**V42**	**V43**	**V44**	**V45**
22	Filtering for a5 $\sim$ a20? (Update filter at)	No	Yes	Yes	Yes	No	Yes	No
27	Filtering for a5 $\sim$ a20? (Update filter at)	No	Yes	Yes	Yes	Yes	Yes	No
32	Filtering for a5 $\sim$ a20? (Update filter at)	No	Yes	No	Yes	No	No	No
37	Filtering for a5 $\sim$ a20? (Update filter at)	No	Yes	No	Yes	No	No	No

for views that indeed utilize ARF, most of the frames are encoded using one-pass coding with filtered references constructed using already estimated filters.

4.4 Conclusions

In this chapter, by exploiting the focus mismatch characteristics derived from the theoretical analysis in Section 2.3, we propose techniques to efficiently apply ARF to inter-view prediction in MVC. We analyze the performance of ARF in MVC inter-view prediction. For different views, the gains in coding efficiency demonstrate a strong view-wise variation. For a given view, the estimated filter coefficients at different timestamps exhibit strong correlation when the depth-composition within the scene remain similar. The observed properties conform with the results in Chapter 2. The two techniques introduced in the chapter are i) *view-wise ARF adaptation* based on RD-cost prediction, which determines whether ARF is beneficial for a given view, and ii) *filter updating based on depth-composition change*, in which the same set of filters will be used until there is significant change in the depth-composition within the scene. Simulation results show

Table 4.3: Encoding selection of the proposed efficient ARF: *Race1* (35 anchors each view. a1 $\sim$ a4 and a21 $\sim$ a24 are encoded with two-pass ARF. If it is determined that the remaining anchors need filtering, the anchors at which filters are updated are also listed in the table, and the other anchors will be encoded by re-using the filters.)

		Race1 (35 anchors)						
QP	**Scenario**	**V1**	**V2**	**V3**	**V4**	**V5**	**V6**	**V7**
22	Filtering for a5 $\sim$ a20?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)	(a14)	(a14)	(a14)	(a14)		(a15)
	Filtering for a25 $\sim$ a35?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)						
27	Filtering for a5 $\sim$ a20?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)			(a16)	(a16)		
	Filtering for a25 $\sim$ a35?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)						
32	Filtering for a5 $\sim$ a20?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)			(a16)			
	Filtering for a25 $\sim$ a35?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)						
37	Filtering for a5 $\sim$ a20?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)	(a15)	(a15)	(a15)			
	Filtering for a25 $\sim$ a35?	Yes	Yes	Yes	Yes	Yes	No	Yes
	(Update filter at)	(All)						

significant complexity reduction can be achieved, since the complete two-pass ARF encoding process needs to be applied to only 20% $\sim$35% of the frames, while the coding efficiency is not affected significantly (e.g., around 0.05 dB loss).

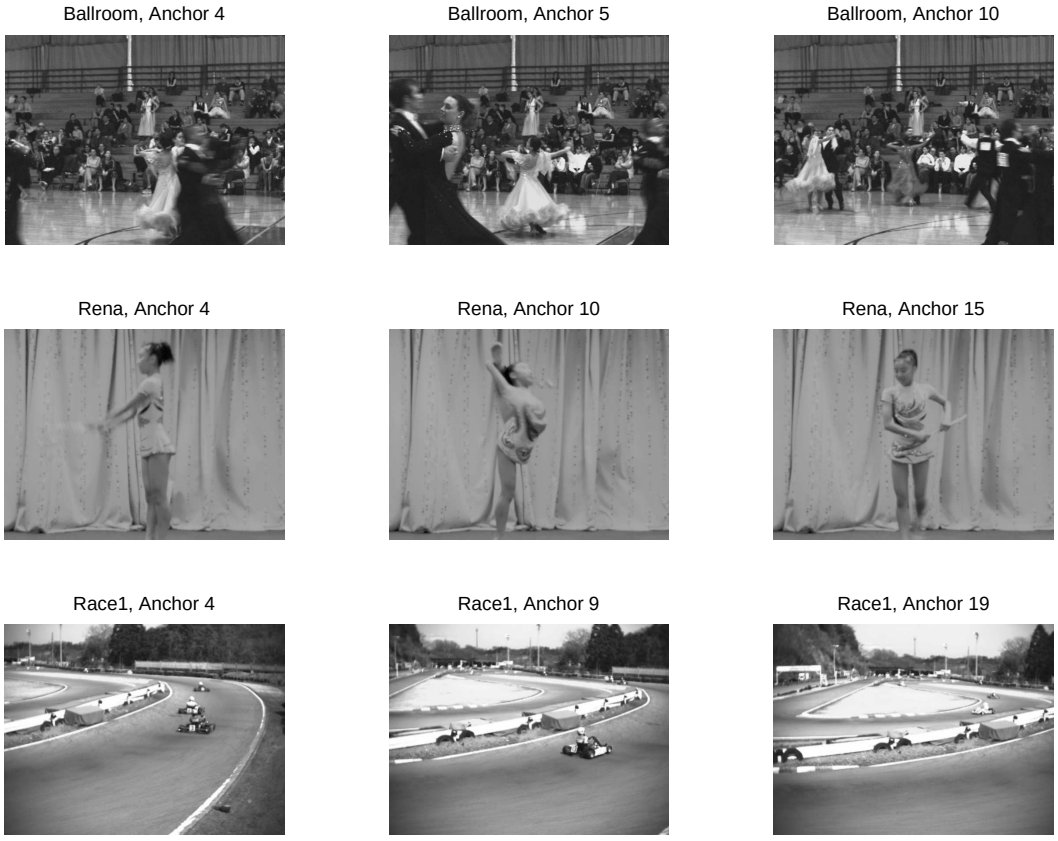

Figure 4.7: Images at different anchor timestamps for the sequences tested

Chapter 5

Predictive Fast Motion/disparity Search for Multiview Video Coding

5.1 Introduction

As described in Chapter 1, temporal and inter-view redundancy can be exploited in MVC by applying block-based motion/disparity compensated prediction (joint MCP/DCP). As before, let us denote $I_C(x, y)$ the luminance pixel value of the current frame to be encoded, with (x, y) representing the pixel position within a frame, and let $I_R(x, y)$ denote the luminance pixel value of the reconstructed reference frame. In block-based motion/disparity compensation, for a given block with size $M \times N$ at position (x_o, y_o) withing I_C, the block matching procedure is performed by evaluating different displacement vectors (dx_i, dy_i), to find the best predictor from the reference I_R that minimizes a cost function. The most commonly used cost function is the sum of absolute differences (SAD). The block matching procedure with SAD as a cost function can be summarized as:

$$\min_{(dx_i, dy_i)} \sum_{y=y_o}^{y_o+M-1} \sum_{x=x_o}^{x_o+N-1} |I_C(x, y) - I_R(x + dx_i, y + dy_i)| \tag{5.1}$$

For MVC structures with joint MCP/DCP, such as the one depicted in Fig. 1.2 and others in [3, 12, 34, 56], higher coding efficiency is achieved by *independently* searching the best predictor within *each* reference (temporal or inter-view), and selecting the one with lower matching cost. While consistent coding gain is obtained, these coding schemes require much higher coding complexity as compared to simulcast: For a frame utilizing joint MCP/DCP, both motion estimation (ME) and disparity estimation (DE) have to be performed, leading to additional search complexity.

To reduce the complexity associated with joint MCP/DCP, fast search methods have been proposed which exploit the relationship between motion and disparity fields. They can be classified into two categories: 1. Joint motion and disparity fields estimation, and 2. *Predictive* fast search algorithms. In the first category, a regularization constraint is utilized to jointly estimate motion and disparity fields. Fig. 5.1 illustrates a possible regularization constraint when a physical point P appears in two views at different time (i.e., P is not occluded).

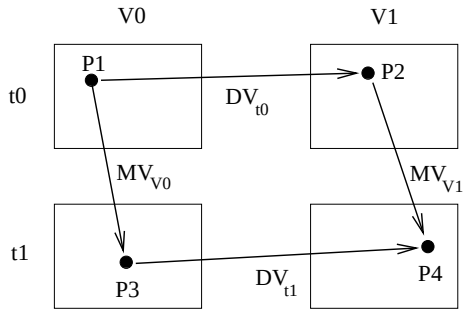

Figure 5.1: The regularization constraint on motion/disparity fields

In Fig. 5.1, $P1, P2, P3, P4$ are the pixels corresponding to the same point P in the scene. MV_{V0} and MV_{V1} are the displacements from $P1$ to $P3$ and from $P2$ to $P4$ (motion

vectors); and DV_{t0} and DV_{t1} are the displacements from $P1$ to $P2$ and from $P3$ to $P4$ (disparity vectors). The depicted regularization constraint can also be represented as:

$$MV_{V0} + DV_{t1} = DV_{t0} + MV_{V1} \tag{5.2}$$

Instead of independently searching for motion and disparity predictors, *joint estimation* of both motion and disparity fields can be performed by imposing such regularization constraints to the block matching procedure [57, 59]. For example in Fig.5.1, after obtaining DV_{t0} and MV_{V0}, the remaining two vectors DV_{t1} and MV_{V1} can be searched jointly by selecting the pair of vectors which provides the minimum matching cost while satisfying (5.2). Although searching complexity can be reduced, these methods have the drawback that, due to the constrained search, obtained predictors might not be as good as those found by independent motion and disparity search. As a result, applying such joint estimation methods causes some noticeable degradation in coding efficiency.

For the second category (predictive fast search), correlation between disparity fields at different timestamps, or correlation between motion fields in different views, are exploited. For example in [11], at certain timestamps, a single global disparity vector between $V0$ and $V1$ is estimated by finding the most dominant disparity value in blocks correspond to static background. To perform ME for a current block in a frame within $V1$, by using this global disparity, the corresponding block in the frame within $V0$ is first obtained. Then the MV of this corresponding block is chosen as search candidate for ME of the current block, and a smaller search range applied. However, a single global disparity vector might not be accurate to provide inter-view correspondence for all blocks within a

frame. In particular, objects closer to the cameras will have larger disparity as compared to the background. As a result, MV candidates obtained with this method could be less reliable for foreground regions.

In this chapter, we propose predictive fast search algorithms (category 2) that can be used when either the motion or the disparity field is available and we wish to estimate the other field efficiently. For a current frame to be encoded, if, say, its disparity field is available after performing DE, and the motion field is available for frames in other views, then instead of using a global disparity vector as in [11], our method uses the estimated block-wise disparity field to locate the corresponding blocks in other views, and exploit their MV as candidates. Likewise, it is also possible to obtain good disparity candidates when the motion field is available for the current frame after ME, and the disparity field is available for frames at other timestamps. Furthermore, we construct a model and analytically demonstrate how mismatches, such as illumination change, will affect the accuracy of the first estimated displacement field (motion or disparity), which consequently will affect the reliability of the candidate vectors obtained with our methods. We also investigate search strategies that can better exploit the characteristics of the candidate vectors obtained with our method. The rest of this chapter is organized as follows. In Section 5.2, we introduce the proposed predictive search algorithms with candidate vectors obtained via local motion/disparity information (in lieu of using a single global vector as in [11]). The analytical model describing the relationship between displacement estimation error and illumination change, is provided in Section 5.3. In Section 5.4, simulation design such as different search strategies, and coding results across different bitrates, are discussed. In this section we also discuss the bit allocation issue [22]

and we demonstrate improved overall performance is achievable when simple bit allocation rules are used. Finally, conclusions are presented in Section 5.5.

5.2 Predictive Fast Search with Candidate Vectors Obtained from Local Motion/Disparity Information

In this section, we consider the problem of complexity reduction in motion/disparity estimation for frames using joint MCP/DCP by proposing search algorithms that, after either the motion field or the disparity field has been estimated, obtain with low complexity a good set of candidate vectors for the other field. From Fig. 5.1, with small temporal and inter-view distances, $MV_{V0} \approx MV_{V1}$ and $DV_{t0} \approx DV_{t1}$. The main novelty in our method is that, we explicitly exploit the fact that MV_{V0} and MV_{V1} can be related via DV_{t1}, while DV_{t0} and DV_{t1} can be related via MV_{V1}. The proposed scheme performs predictive motion search from view to view and predictive disparity search from one timestamp to another time. Details are provided in the next subsections.

5.2.1 Predictive Search I: Disparity then Motion (DtM)

In standard video coding, fast motion search algorithms can be designed based on the assumption that the MV in neighboring blocks are highly correlated. We can use the result of ME in causal neighboring blocks to obtain a set of MV candidates that can be used to initialize a search with reduced search range in the current block [7, 44]. In a MVC scenario, since there are multiple cameras capturing the same scene, it is possible to predict the motion field of one view using the other view's motion as reference [11].

We propose that for a given frame, after its disparity field has been estimated, we will be able to find good search candidates for the motion field with very low complexity. Fig. 5.2 depicts the basic procedure for predicting the motion field from view to view by identifying MV candidates. In Fig. 5.2(a), the vector field represented with solid arrows is estimated first. When encoding a non-anchor frame, DE is performed first. For a given block in a non-anchor frame of current view, Fig. 5.2(b), we track along its disparity vector to its corresponding block in the reference view. Depending on where this block is located, it could be overlapped with at most 4 blocks in the reference view, each having a MV. These can serve as the candidates, denoted MV_1, MV_2, MV_3, MV_4 in Fig. 5.2(b), and provide initialization points for the search. After predicting the motion field of the current view, the same procedure is used to predict the motion field of the next view, using the current view as the reference. We denote this as the Disparity then Motion (DtM) scheme.

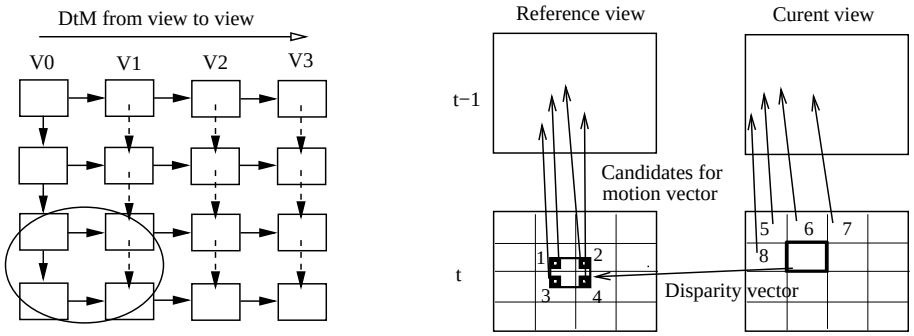

Figure 5.2: (a)Left: The structure of predicting the motion field from the reference view (DtM) (b)Right: Obtaining MV candidates to predict the motion field

To assess the performance of the additional MV candidates obtained using DtM, we define two sets of candidates (refer to Fig.5.2(b)): $A = \{MV_5, MV_6, MV_7, MV_8, \vec{0}\}$, which

contains the motion vectors selected from blocks in a causal neighborhood (as typically used in many fast motion estimation algorithms) and $B = A \cup \{MV_1, MV_2, MV_3, MV_4\}$, which also includes the additional candidates obtained from the neighboring view. For each block in a non-anchor frame, we choose the candidate that provides the minimum SAD from each of the sets. The residual image PSNR values are compared in Table 5.1.

For all three test sequences (*Aqua*, *Ballroom*, and *ST*), it can be seen that higher quality is achieved if the additional candidates provided by the DtM scheme are considered. One of the drawbacks of the fast motion search using neighboring blocks is that the search may only identify a motion vector representing a local minimum when the current block happens to have a MV that is different from those of the neighbors. A typical example is in situations where the current block is located close to the boundary of a moving object. Our DtM approach helps to alleviate this problem by providing a *block correspondence in the other view* to obtain additional MV candidates.

Table 5.1: Compare different sets of MV candidates, PSNR of residue images

Aqua	Set A	Set B	Ballroom	Set A	Set B	ST	Set A	Set B
V1	29.47	30.65	V1	28.45	29.25	V1	30.70	32.74
V0	29.54	30.79	V2	28.29	29.16	V2	31.14	32.93
			V3	27.94	28.84			

5.2.2 Predictive Search II: Motion then Disparity (MtD)

A similar idea can be applied to predict the disparity field after the motion field has been estimated. We can use the disparity at time $t-1$ as the reference to predict the disparity at time t. This approach is illustrated in Fig. 5.3. Again, the field shown

with a solid arrow is estimated first. This time when encoding a non-anchor frame, motion estimation is performed first. For a given block in a non-anchor frame at time t, Fig. 5.3(b), we track along its motion vector to its corresponding block at time $t-1$. This would give us at most 4 different candidates DV_1, DV_2, DV_3, DV_4 to initialize the disparity search. After predicting the disparity field at time t, this disparity field will be used as the reference to predict the disparity field at time $t+1$. We denote this as the Motion then Disparity (MtD) scheme.

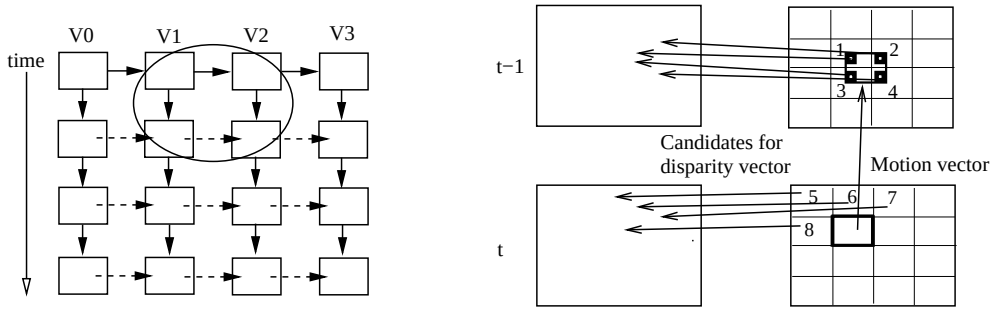

Figure 5.3: (a)Left: The structure of predicting the disparity field from a time instance (MtD) (b)Right: Obtaining DV candidates to predict the disparity field

5.3 Displacement Estimation under Illumination Changes

The accuracy of the first estimated field will affect the reliability of the obtained candidates: If the first field failed to find accurate corresponding blocks, the candidate vectors are less trustworthy as they are from regions that do not correspond to the block we want to encode. There are two primary factors that will affect the accuracy of the estimated block correspondence. Firstly, instead of pixel-based estimation, motion/disparity compensated prediction is performed in a block-based manner. Different block sizes/shapes and the signal characteristics within the block will have influence on the matching process.

Secondly, the accuracy can also be affected by mismatches other than simple displacement, For example, it has been observed that frames from different views in multiview systems can suffer from illumination and focus mismatches [17]. Inter-view disparity compensation could fail to obtain reliable correspondence due to these mismatches.

In [31], a block-based illumination model is proposed by first decomposing a block signal into its mean and a mean-removed signal. In this section, we adopt such decomposition to demonstrate how the accuracy of the estimated displacement will be affected by the illumination mismatch when using SAD as cost function, thus justifying the use of mean-removed block matching utilized in [31]. Moreover, we will derive specific properties for planar regions which relate the estimation error to the signal characteristics such as mean and gradient, and to the block matching size.

Following from (5.1), we denote the current block to be matched as B_C, and the i^{th} candidate reference block with displacement vector (dx_i, dy_i) as B_R^i. That is:

$$\begin{aligned} \text{For } 0 \le x \le N-1,\, 0 \le y \le M-1\text{: } B_C(x,y) &= I_C(x_o + x, y_o + y) \\ B_R^i(x,y) &= I_R(x_o + x + dx_i, y_o + y + dy_i) \end{aligned} \quad (5.3)$$

A block of pixels can be written as the sum of its mean μ and a zero-mean signal $\mathbf{w}$ [31] (which we denote as a *mean-removed structure*), i.e.,

$$B_C(x,y) = \mu_C + \mathbf{w}_C(x,y) \quad \text{and} \quad B_R^i(x,y) = \mu_R^i + \mathbf{w}_R^i(x,y), \quad (5.4)$$

The block matching procedure with SAD cost function can then be represented as:

$$\min_i \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} \left| B_C(x,y) - B_R^i(x,y) \right|$$

$$\min_i \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} \left| (\mu_C - \mu_R^i) + \left(\mathbf{w}_C(x,y) - \mathbf{w}_R^i(x,y) \right) \right| \tag{5.5}$$

The reference block which provides the minimum SAD is denoted as B_R^*. We assume that, when there is no other type of mismatch between I_C and I_R, and no occlusion for B_C in I_R, this reference block B_R^* gives the most accurate correspondence to B_C, i.e. the difference of means is $\Delta\mu_R^i = (\mu_C - \mu_R^*) \approx 0$ and the mean-removed structure difference is $(\mathbf{w}_C - \mathbf{w}_R^*) \approx \mathbf{0}$ (zero matrix).

Now, assume that there are illumination mismatches between the current frame and the reference frame, such that in the neighborhood of the most accurate correspondence B_R^*, the reference blocks are affected by a local illumination change so that:

$$\begin{aligned} \hat{B}_R^i(x,y) &= S \cdot B_R^i(x,y) + C \\ &= (S \cdot \mu_R^i + C) + S \cdot \mathbf{w}_R^i(x,y), \end{aligned} \tag{5.6}$$

where (5.6), S is a multiplicative scale factor and C is an additive offset. Substituting (5.6) into (5.5), we get:

$$\min_i \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} \left| \left(\mu_C - S \cdot \mu_R^i - C\right) + \left(\mathbf{w}_C(x,y) - S \cdot \mathbf{w}_R^i(x,y) \right) \right| \tag{5.7}$$

The block matching with SAD cost function will be affected by S and C. For the special case with only additive illumination offset, i.e., $S = 1$, (5.7) will be simplified as:

$$\min_i \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} \left| \left(\mu_C - \mu_R^i - C\right) + \left(\mathbf{w}_C(x,y) - \mathbf{w}_R^i(x,y)\right) \right| \tag{5.8}$$

In (5.8), a larger $|C|$ will have stronger effect on the estimated displacement, leading to less reliable block correspondence. While the matching for mean is affected by the offset C, matching for the mean-removed structure remains the same. Thus for localized additive illumination offset, we can modify the cost function as a *mean-removed* SAD to preserve block correspondence when performing block matching [31].

In what follows, we discuss an example in the case of homogeneous/smooth regions in order to quantify how illumination parameters S and C affect the accuracy of the estimated displacements.

Example: Homogeneous/Smooth-varying Regions:

For a homogeneous/smooth varying region, we assume that the pixel values fit closely to a planar function such that (1) $B_C(x,y) = ax + by + d$, and (2) The corresponding region in the reference frame, without any illumination change, is a shifted version with displacement vector (Dx, Dy), i.e., $B_R^i(x,y) = a\left[(x - Dx) + dx_i\right] + b\left[(y - Dy) + dy_i\right] + d$. In this scenario, we will have $\mathbf{w}_R^i = \mathbf{w}_C$. By replacing μ_R^i with $(\mu_C - \Delta\mu_R^i)$, the block matching of (5.7) can then be approximated as:

$$\min_i \sum_{y=0}^{M-1}\sum_{x=0}^{N-1} \left|\left((1-S)\mu_C + S\cdot\Delta\mu_R^i - C\right) + \left(\mathbf{w}_C(x,y) - S\cdot\mathbf{w}_C(x,y)\right)\right|$$

$$\min_i \sum_{y=0}^{M-1}\sum_{x=0}^{N-1} \left|(1-S)\left(\mu_C + \mathbf{w}_C(x,y)\right) - C + S\cdot\Delta\mu_R^i\right| \tag{5.9}$$

Given that for $\sum_k |a_k - X|$, the minimum occurs when X is equal to the median of the sequence $\{a_k\}$, the block matching in (5.9) will find the block with mean difference closest to the following value:

$$\Delta\mu_R = -\left(\frac{1-S}{S}\cdot median(\mu_C + \mathbf{w}_C(x,y)) - \frac{C}{S}\right) \tag{5.10}$$

Since the pixel positions x and y are both equally spaced, $median(\mu_C + \mathbf{w}_C(x,y))$ $= median\left\{ax+by+d\big|_{0\le y\le M-1}^{0\le x\le N-1}\right\} = mean\left\{ax+by+d\big|_{0\le y\le M-1}^{0\le x\le N-1}\right\} = \mu_C$. Thus from (5.10), the block matching will find a block with:

$$\Delta\mu_R = -\left(\frac{1-S}{S}\mu_C - \frac{C}{S}\right), \text{for planar regions} \tag{5.11}$$

μ_C and $\Delta\mu_R$ at a given (dx, dy) can be computed as:

$$\mu_C = \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1}(ax+by+d) = a\frac{N-1}{2} + b\frac{M-1}{2} + d \tag{5.12}$$

$$\begin{aligned}\Delta\mu_R|_{(dx,dy)} &= \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1}(ax+by+d) - [a(x-Dx+dx) + b(y-Dy+dy) + d] \\ &= a(Dx-dx) + b(Dy-dy)\end{aligned} \tag{5.13}$$

Using (5.12) and (5.13) in (5.11), we get:

$$a(Dx - dx) + b(Dy - dy) = -\left(\frac{1-S}{S} \cdot (a\frac{N-1}{2} + b\frac{M-1}{2} + d) - \frac{C}{S}\right) \tag{5.14}$$

To illustrate the effect of illumination parameters (S, C), let us consider the case when the region has zero gradient in y direction ($b = 0$), in other words, pixel values linearly change along x-axis only:

$$\begin{aligned} a(Dx - dx) &= -\left(\frac{1-S}{S} \cdot (a\frac{N-1}{2} + d) - \frac{C}{S}\right) \\ dx &= Dx + \left(\frac{1-S}{S} \cdot (\frac{N-1}{2} + \frac{d}{a}) - \frac{C}{aS}\right) \end{aligned} \tag{5.15}$$

We can see that, in the presence of illumination change, the estimated displacement dx will deviate away from the actual displacement Dx, by $\frac{1-S}{S} \cdot (\frac{N-1}{2} + \frac{d}{a})$ - $\frac{C}{aS}$. Thus, besides the obvious fact that larger $|C|$ and S farther away from 1 will result in greater error in the estimated displacement, other interesting properties can be observed from (5.15), which we summarize below:

With only additive illumination change ($S = 1$), $|dx - Dx| = |\frac{C}{a}|$

Planar regions with larger gradient will be more robust to such additive offset, since $|dx - Dx|$ is inversely proportional to a(or b).

With only multiplicative illumination change ($C = 0$), $|dx - Dx| = |\frac{1-S}{S} \cdot (\frac{N-1}{2} + \frac{d}{a})|$

- When a and d are fixed, block matching with larger block size (larger N) is more vulnerable to multiplicative illumination change, leading to larger error in the estimated displacement.

- Under same block size and same gradient (N and a fixed), a block with a larger d, hence a larger mean, will receive stronger influence from multiplicative illumination change, thus will make greater error in the estimated displacement.

In conclusion, stronger illumination mismatch (greater $|C|$ and $|1 - S|$ values) will cause larger error in the estimated displacement, thus providing less reliable block correspondence. For planar regions, more specific properties can be derived which relate the estimation error to the signal characteristics (mean, gradient) and to the block matching size. A larger error in estimated displacement indicates that the selected reference block is farther away from the actual corresponding block B_R^* for B_C. Consequently, the candidate vectors obtained with methods proposed in Section 5.2 will empirically become less trustworthy. In Section 5.4, simulation results will demonstrate that when the first estimated field is affected by illumination change such that the block correspondence is not accurate, performing predictive search for the other field based on the obtained candidates, will lead to larger degradation in coding efficiency.

5.4 Experiments Design and Results

In this section, we design experiments to evaluate the proposed predictive fast search methods which obtain new candidate vectors. For frames utilizing joint MCP/DCP, we consider the following three schemes: 1. Both ME and DE use full search (we denote this

a *dual full search scheme*), 2. DtM, where disparity field is estimated by full search and motion is fast searched with new candidates, and 3. MtD, where motion is full searched and disparity is fast searched with new candidates. The dual full search scheme serves as a baseline of the potential gain as compared to simulcast. As for MtD and DtM, there are multiple candidate vectors that can be used. In Section 5.4.1, we investigate different search patterns utilizing these candidates.

5.4.1 Investigation of Efficient Search Patterns

As described in Section 5.2, the main novelty in the proposed algorithms is that we track along the first estimated field (motion or disparity) to obtain candidate vectors for the other field. The additional candidates provide improved prediction in cases where the motion/disparity vector of the current block is not similar to that of its neighboring blocks.

In most predictive motion search algorithms for monoscopic video, the mean or median of the candidates is used to initialize the search location. This approach relies on the assumption that the motion field tends to be locally smooth. However, the disparity field is not as homogeneous as the motion field and disparity vectors can be seen to exhibit significant variation even across neighboring blocks [30, 56]. To see why this is true, consider that in disparity estimation an area within an object that is closer to the camera will have larger disparity than an area in the same object that is further away from the camera. However motion in the two areas is likely to be the same unless the object is rotating. Thus blocks that belong to the same moving object could have very different disparity even though they have similar motion vectors. This suggests that computing

the mean/median of a set of disparity vector candidates may not provide as good a predictor as applying the same technique to a set of motion vectors candidates. To tackle this problem, we investigate a search strategy, such that multiple searches are performed around each of the candidates, with each of the searches employing a much smaller search range than what would be typically used in combination a single search window centered at the mean of the candidates. Fig. 5.4 illustrates this search pattern. Note that this multiple-search method can be seen as an extension of the enhanced predictive zonal search (EPZS) [44] where we obtain additional candidate vectors via motion/disparity fields.

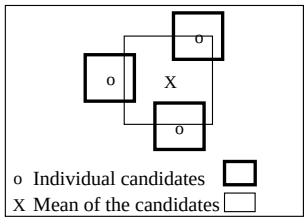

Figure 5.4: The search pattern that uses all candidate vectors

For MtD, there are 9 candidates to be considered: DV_1 to DV_8 and $\vec{0}$. One approach is to order the candidates, based on their SAD, for example, and only perform search around the top priority candidates. Or we can search around all 9 candidates. Since some of the candidates might be the same and their search ranges may overlap, we currently adopt the second approach so that a search is performed for each of the candidates.

To verify the efficiency of the new search pattern, we compared the following two scenarios in the MtD scheme:

- **EachC**: Search with 3×3 windows ($\pm 1 \times \pm 1$) centered at each DV candidate

- **MeanC**: Search with a 9×9 window ($\pm 4 \times \pm 4$) centered at the mean of the DV candidates

Note that the *maximum* number of vectors to search for **EachC** is 81, while **MeanC** always has to check exactly 81 vectors.

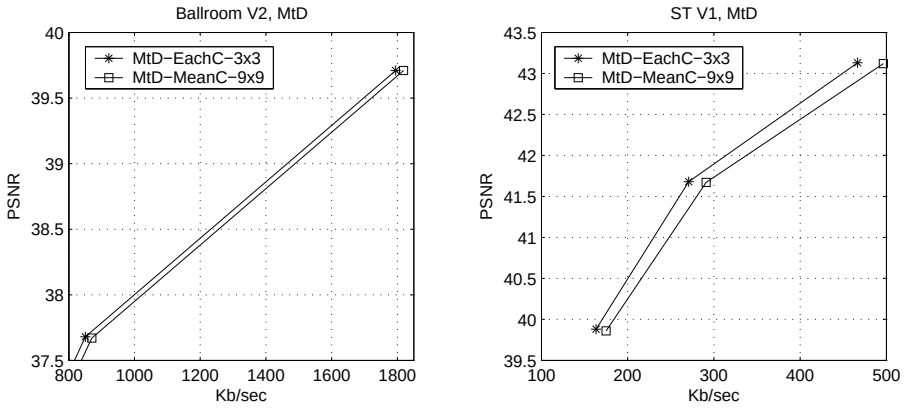

Figure 5.5: Comparison of different search patterns.

Fig. 5.5 provides the simulation results for two different sequences using H.264/AVC-based coding structure described in Fig. 5.2. The search pattern **EachC** achieves higher coding efficiency when we perform the predictive search on disparity estimation. The non-smooth disparity field is better predicted because our search pattern takes all the candidate vectors into account instead of adopting the mean of candidates as a single search center. In the following simulations, we adopt this multiple-search method for both MtD and DtM schemes.

5.4.2 H.264/AVC-based Simulations

Our H.264/AVC-based MVC coding structure is implemented on top of the JM reference software [41]. A immediate benefit of MVC can be observed: the anchor frames are now

encoded using inter-view prediction, which provides a higher coding efficiency as compared to the simulcast case, where they were intra coded. The quality of these encoded anchor frames is crucial because they serve as the temporal references for the later frames. The work in [22] studied the bit allocation issue in dependent video coding scenarios such as those arising in MVC. Their results demonstrate that if more bits are spent on anchor frames, a better overall coding efficiency will be achieved, because all the frames following the anchor can be encoded more efficiently. Here we simply use a smaller quantization step size (the QP parameter in H.264/AVC codec) to encode the anchor frames. For example, if the non-anchor frames are coded with QP = 28, then the anchor frames will be coded with QP = 26. Fig. 5.6 shows the effect of this QP change. Gains from this bit allocation are observed on all the test sequences and under all three coding schemes (dual full search, DtM, MtD). In all the following results provided in this section, we adopt this smaller QP for anchor frames.

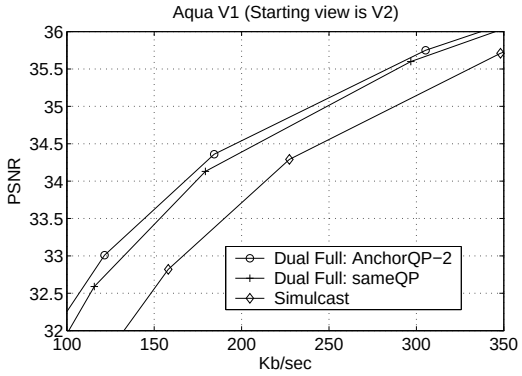

Figure 5.6: The effect of changing QP for the anchor frames

In Figures 5.7, 5.8, and 5.9, we present the rate-distortion curves (R-D curves) from our simulation results. The GOP structure in our simulations has 10 frames in time

direction and 3 to 4 views in spatial direction. The degradation of MtD and DtM, as compared to the dual full search scheme, comes from that one of the field is fast searched with the proposed algorithms. When using a 3×3 range ($\pm 1 \times \pm 1$) around each candidate, the average number of vectors tested for the fast searched field, is about 40 $\sim$ 65 for disparity and 30 $\sim$ 50 for motion. This is a very low cost for the predictive searched field, as compared to full search range of $\pm 32 \times \pm 32 = 4225$ or $\pm 64 \times \pm 64 = 16641$ used in our simulations (see Table 5.2). Note that during the search, here we did not use any early termination techniques such as the one proposed in EPZS [44]. Since in Section 5.4.1 we have demonstrated that the candidates obtained with our methods can be more efficiently exploited with multiple search regions, it is reasonable to consider using them in the context of the EPZS scheme in which its early termination techniques can help further reduce the search complexity.

The MtD approach achieves very good performance for all three test sequences, even with a small search window (3×3) around each candidate. The R-D curves are very close to the curves obtained with the dual full search coding scheme. The performance of DtM predictive search varies among different test sequences. The *Aqua* sequence has the most dense camera setting among our three test sequences: 15 cameras with about 1.8cm spacing. The correlation between motion fields in different views is very high. Fig. 5.7

Table 5.2: Parameters of the sequences and simulation settings for Section 5.4

Sequence	Dimension	Frame Rate	No.Views	GOP in simulation	Full search range
Aqua	320×240	10 fps	15	V2→V1→V0	±32 × ±32
Ballroom	640×480	25 fps	8	V0→V1→V2→V3	±64 × ±64
ST	640×480	15 fps	6	V0→V1→V2	±64 × ±64

shows that the DtM approach provides almost the same coding efficiency as MtD. For the *Ballroom* sequence, the R-D performance of DtM exhibits some degradation (0.1 $\sim$ 0.2 dB) with respect to the dual full search scheme. In our simulations, this is the sequence with the highest gain in coding efficiency (up to 1.5 dB) when comparing MVC with simulcast (Fig. 5.8). DtM preserves much of this gain with a low complexity for predictive motion search. The worst case of DtM appears to be on the ST sequence (Fig. 5.9). As discussed in Section 5.3, the inter-view illumination mismatch reduces the accuracy of the first estimated disparity field. With a 3×3 search range for MV candidates, the DtM's R-D curves lie about halfway between the dual full scheme and simulcast. We provide one more set of simulation results as the search range is increased to 5×5. The corresponding R-D curves move to about 0.1 $\sim$ 0.2 dB below the dual full search scheme.

Once again we see that the key to the performance of the proposed predictive algorithm is that the most reliable estimation should be performed first, so that the fast predictive search on the second field can make use of good candidate vectors. For MVC, since the camera view settings vary among different applications, plus the fact that frames from different views are prone to suffer from illumination and focus mismatches [17], it is likely that computing the motion field first will be in general more efficient, so that MtD should in general be chosen over DtM in MVC.

5.5 Conclusions

Higher coding efficiency can be achieved in MVC by exploiting both temporal and inter-view redundancy. In this chapter we propose novel predictive fast search algorithms to reduce the complexity for MVC. After one of the motion/disparity fields is estimated, the proposed algorithm obtains good candidate vectors to perform the estimation on the other field with very low complexity. A more efficient search pattern employing the candidate vectors is also investigated, and the results conform with the finding in EPZS. The new candidate vectors can provide additional prediction information if the first estimated field is accurate. Since motion estimation generally provides better block correspondence than disparity estimation, MtD generates very consistent coding efficiency among different test sequences, as compared to DtM. Simulation results with H.264/AVC-based MVC structure show that MtD can achieve coding efficiency that is very similar to the dual full search scheme, while the complexity is reduced significantly. The simulations also verify that in general MtD should be chosen over DtM.

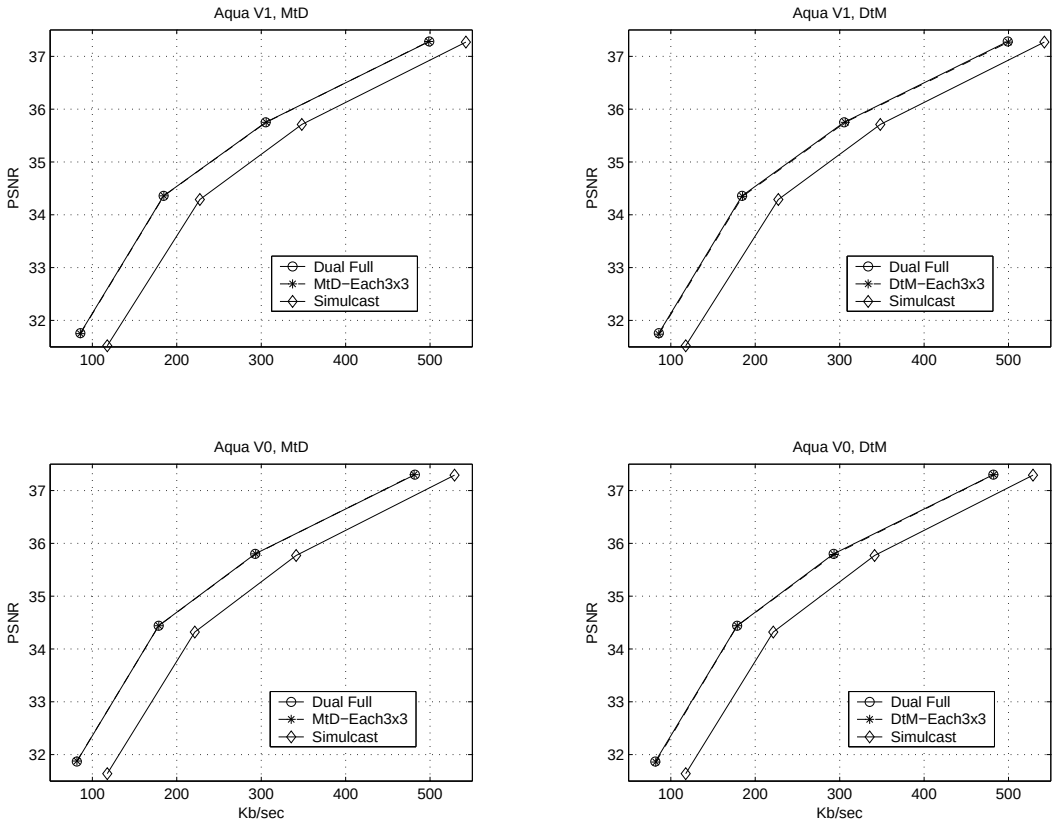

Figure 5.7: Predictive search simulation results for *Aqua* sequence

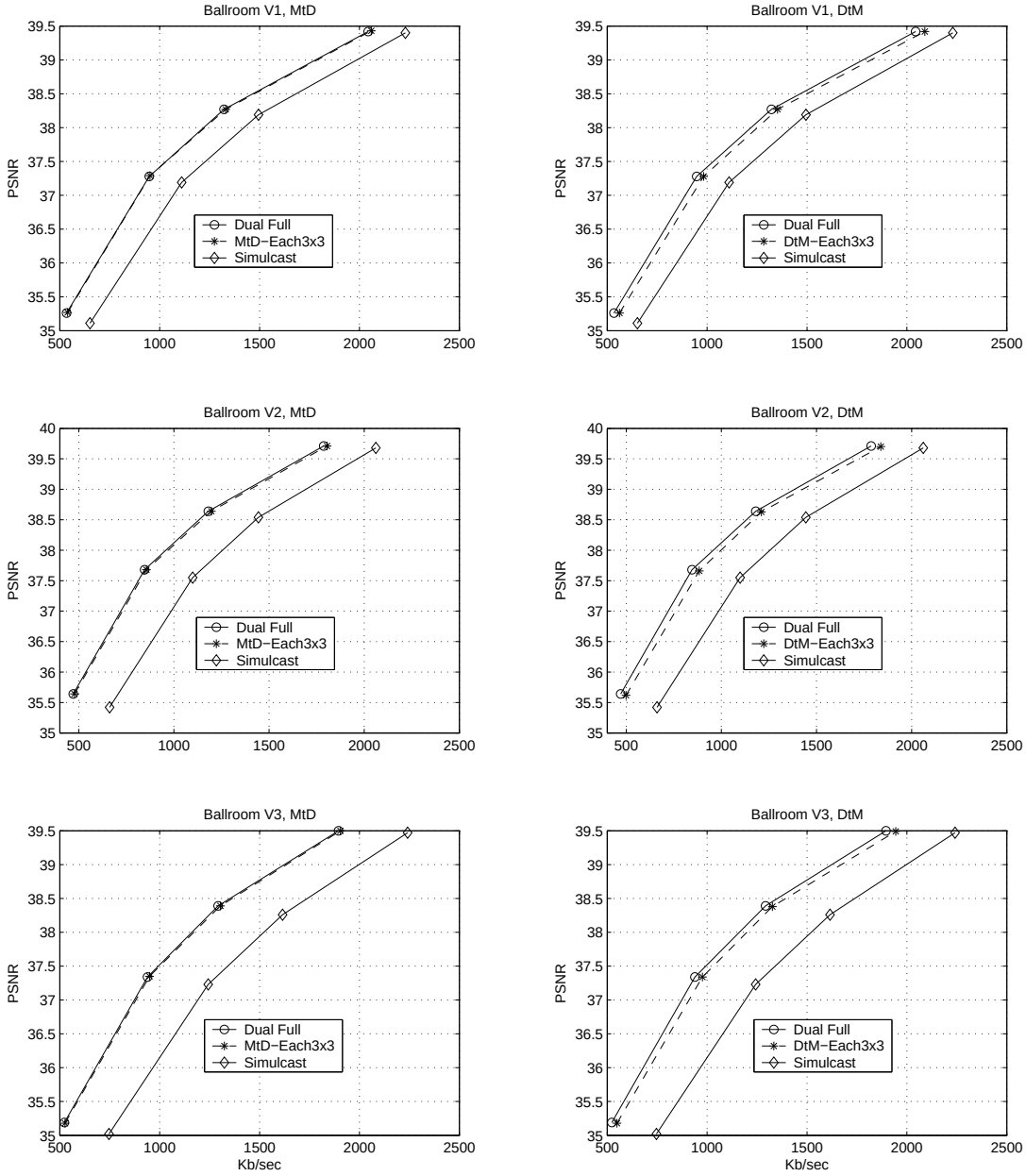

Figure 5.8: Predictive search simulation results for *Ballroom* sequence

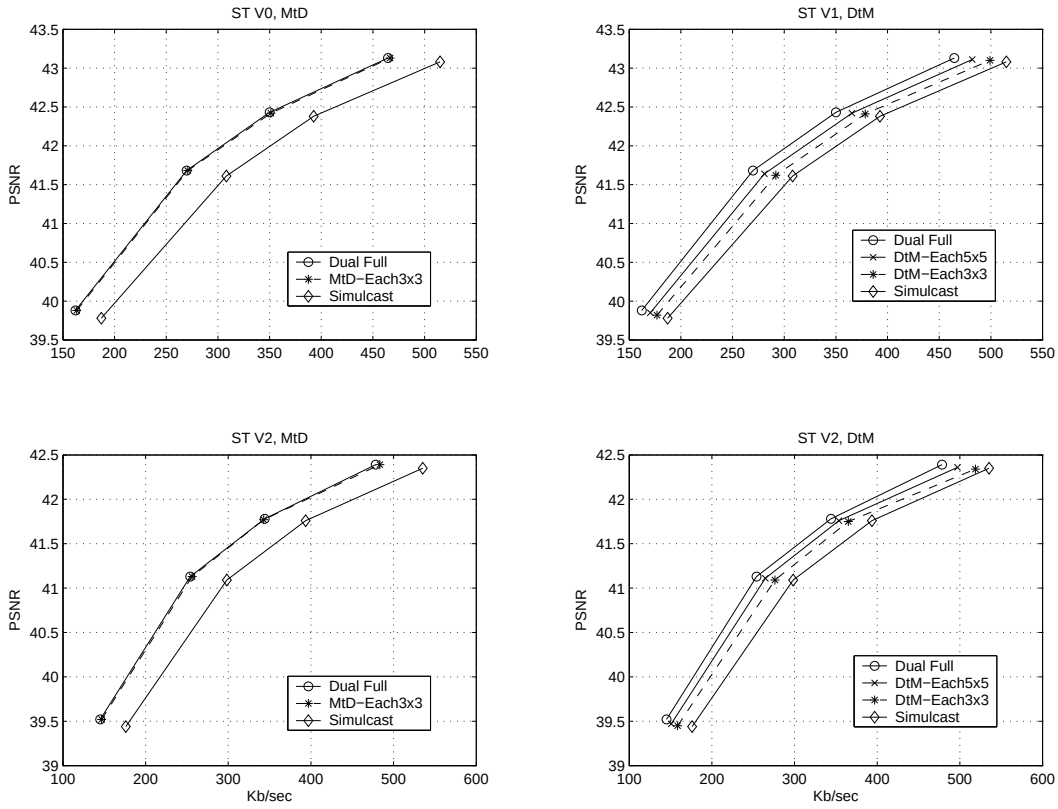

Figure 5.9: Predictive search simulation results for ST sequence

Chapter 6

Conclusions and Future Work

Multiview video compression is a key technology for efficient storage and transmission of multiview video data. In this dissertation, we exploit special characteristics of multiview video to develop techniques that improve MVC efficiency with reduced complexity.

We consider the problem of encoding video content exhibiting focus mismatch due to focus setting differences. We first use geometrical optics to analyze characteristics of images captured under the effect of focus. It is demonstrated that the focus mismatch can be represented in terms of the focus setting parameters (camera-dependency) and the depths of objects (depth-dependency). The focus mismatch kernels are circular symmetric with their shapes varying across different depth. For 1D parallel camera arrangements in multiview systems, we relate the focus mismatch to the disparity exhibited in frames from different views. The analytical results provide useful properties that can be exploited to design focus mismatch compensation techniques.

Based on the analysis, to compensate for depth-dependent focus mismatch, we propose a novel adaptive reference filtering (ARF) approach. We estimate block-wise parameters as features for classification such that an image is first partitioned into regions suffering

from different types of focus mismatch: For inter-view coding in MVC, we exploit the disparity field to partition frames into regions corresponding to different depth levels. As for monoscopic video, with no disparity information, we propose a method to estimate the localized focus changes and partition frames into regions consisting of macroblocks that suffer from a similar type of focus change. After frame-partitioning, for each region, a 2D filter is designed by minimizing prediction error. Filtered references are then generated for encoder to perform rate-distortion optimized coding selection. The ARF approach is also extended to MVC bi-directional inter-view coding, in which we propose filter design method that incorporate well with conventional bi-directional search. Simulation results demonstrate higher coding efficiency as compared to multiple-reference prediction and adaptive interpolation filtering methods.

Finally, complexity reduction techniques for MVC are presented. We analyze the encoding results of ARF on inter-view prediction. It is observed that the coding gains demonstrate a strong view-wise variation, while at different timestamps the estimated filters exhibit strong correlation when the objects' depths remain similar. Based on these findings and the analytical results, we propose i) *view-wise ARF adaptation* based on RD-cost prediction, which determines whether ARF is beneficial for a given view, and ii) *filter updating based on depth-composition change*, in which the same set of filters will be used (i.e., no new filters will be designed) until there is significant change in the depth-composition within the scene. We also propose fast predictive search algorithms, which exploit the relationship between motion and disparity fields, that can be used when one of the fields is available and we wish to estimate the other field efficiently. We construct a model and analytically demonstrate how illumination change will affect the accuracy

of our fast predictive search methods. Simulation results show that when applying these techniques, significant complexity reduction is achieved while the coding efficiency can be well-preserved.

Based on the results in this thesis, there are some interesting extension of the work that can be addressed in future research:

- *Improving ARF based on camera parameters.* In Section 2.4, we listed several properties of focus mismatch based on camera setting parameters and object depth. However, due to the fact that most of these values are not available in the test sequences we have, our ARF approach only exploits qualitative properties such as isotropy and depth/disparity dependency of focus mismatch kernels. It will be interesting to further utilize quantitative properties if camera setting parameters are available. For example in monoscopic video with no disparity information, knowing the setting parameters and depth range of the scene, we can derive focus mismatch kernels and construct a set of representative filters covering these kernels. To identify regions suffering from different type of mismatch, instead of estimating block-wise filters as in Section 3.3.1, we can compare the representative filters to determine which one provides best match and then design MMSE filter for each region.

- *Adaptive filtering for other non-translational mismatches.* The general methodology of ARF, i.e., partition frames into regions based on the properties of the targeted mismatch, and then estimate mismatch kernels, can be extended to other types of mismatch. For example, consider motion blur in sport sequences. Instead of being

isotropic, the direction of motion blur is expected to be highly correlated with motion information. Thus an ARF approach can be developed by separating an image into regions moving in different directions, and design directional filters. Another example is depth-dependent affine transformations in inter-view prediction due to convergent camera arrangement. Once again we can partition a frame into depth levels and for each level estimate affine parameters. Warped reference frames can be generated, after applying obtained affine parameters, to provide better coding efficiency.

Reference

[1] T. Aach and A. Kaup. Disparity-based segmentation of stereoscopic foreground/background image sequences. *IEEE Trans. Communications*, 42, issue.2/3/4, Part.1:673–679, Feb-Apr. 1994.

[2] R. V. Babu, K. R. Ramakrishnan, and S. H. Srinivasan. Video object segmentation: A compressed domain approach. *IEEE Trans. Circuits Systems and Video Technologies (CSVT)*, 14, no.4:462–474, Apr. 2004.

[3] C. Bilen, A. Aksay, and G.B. Akar. A multi-view video codec based on H.264. In *Proc. IEEE 2006 International Conference on Image Processing (ICIP)*, pages 541–544, Atlanta, GA, USA, Oct. 2006.

[4] C. A. Bouman. Cluster: An unsupervised algorithm for modeling Gaussian mixtures. *http://cobweb.ecn.purdue.edu/bouman/software/cluster/*, this version was released in Jul. 2005.

[5] R. N. Bracewell. *The Fourier Transform and Its Applications*. McGRAW-HILL, 3rd edition, 2000.

[6] M. Budagavi. Video compression using blur compensation. In *Proc. 2005 IEEE International Conference on Image Processing (ICIP)*, pages II.882–II.885, Genoa, Italy, Sep. 2005.

[7] J. Chalidabhongse and C.-C.J Kuo. Fast motion vector estimation using multiresolution-spatio-temporal correlations. *IEEE Trans. on Circuits and Systems for Video Technology (CSVT)*, 7, Issue 3:477–488, Jun 1997.

[8] X. Chen and A. Luthra. MPEG-2 multiview profile and its application in 3D TV. In *Proc. SPIE 1997 Multimedia Hardware Architectures*, volume 3021, pages 212–223, 1997.

[9] T. Dekker, S. T. de Zwart, O. H. Willemsen, M. G. H. Hiddink, and W. L. IJzerman. 2D/3D switchable displays. In *Proc. SPIE Liquid Crystal Materials, Devices, and Applications XI*, San Jose, CA, USA, Feb. 2006.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B, 39, no.1:1–38, 1977.

[11] L.-F. Ding, S.-Y. Chien, Y.-W. Huang, Y.-L. Chang, and L.-G. Chen. Stereo video coding system with hybrid coding based on joint prediction scheme. In *Proc. IEEE 2005 International Symposium on Circuits and Systems (ISCAS)*, volume 6, pages 23–26, May 2005.

[12] U. Fecker and A. Kaup. H.264/AVC-compatible coding of dynamic light fields using transposed picture ordering. In *Proc. 13th European Signal Processing Conference (EUSIPCO 2005)*, pages II.882–II.885, Antalya, Turkey, 2005.

[13] U. Fecker and A. Kaup. Statistical analysis of multi-reference block matching for dynamic light field coding. In *Proc. 10th International Fall Workshop Vision, Modeling, and Visualization (VMV 2005)*, pages 445–452, Erlangen, Germany, 2005.

[14] M. Flierl, T. Wiegand, and B. Girod. A locally optimal design algorithm for block-based multi-hypothesis motion-compensated prediction. In *Proc. IEEE Data Compression Conference 1998 (DCC)*, pages 239–248, Mar 1998.

[15] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[16] E. Francois and B. Chupeau. Depth-based segmentation. *IEEE Trans. Circuits and Systems for Video Technology (CSVT)*, 7:237–239, Jun. 1997.

[17] Y.-S. Ho, K.-J. Oh, C. Lee, B. Choi, and J.-H. Park. Observations of multi-view test sequences. *JVT Document W084*, Apr 2007.

[18] ISO/IEC-JTC1/SC29/WG11. Call for proposals on multi-view video coding. *MPEG Document N7327*, Jul 2005.

[19] ISO/IEC-JTC1/SC29/WG11. Submissions received in CfP on multi-view video coding. *MPEG Document M12969*, Jan 2006.

[20] E. Izquierdo. Disparity/segmentation analysis: Matching with an adaptive window and depth-driven segmentation. *IEEE Trans. Circuits and Systems for Video Technology (CSVT)*, 9, no.4:589–607, Jun. 1999.

[21] M. L. Jamrozik and M. H. Hayes. A compressed domain video object segmentation system. In *Proc. IEEE 2002 International Conference on Image Processing (ICIP)*, pages I.113–I.116, Rochester, NY, Sep. 2002.

[22] J.-H. Kim, J. Garcia, and A. Ortega. Dependent bit allocation in multiview coding coding. In *Proc. IEEE 2005 International Conference on Image Processing (ICIP)*, pages II.293–II.296, Genoa, Italy, Sep. 2005.

[23] J.-H. Kim, P. Lai, J. Lopez, A. Ortega, Y. Su, P. Yin, and C. Gomila. New coding tools for illumination and focus mismatch compensation in multi-view video coding. *IEEE Trans. Circuits Systems and Video Technologies (CSVT)*, 17, no. 11:1519–1535, Nov 2007.

[24] P. Lai and A. Ortega. Predictive fast motion/disparity search for multiview video coding. In *Proc. SPIE 2006 Visual Communications and Image Processing (VCIP)*, volume 6077, Jan 2006.

[25] P. Lai, A. Ortega, P. Pandit, P. Yin, and C. Gomila. Adaptive reference filtering for bidirectional disparity compensation with focus mismatches. In *Proc. IEEE 2008 International Conference on Image Processing (ICIP)*, pages 2456–2459, San Diego, CA, USA, Oct 2008.

[26] P. Lai, A. Ortega, P. Pandit, P. Yin, and C. Gomila. Focus mismatches in multiview systems and efficient adaptive reference filtering for multiview video coding. In *Proc. SPIE 2008 Visual Communications and Image Processing (VCIP)*, Jan 2008.

[27] P. Lai, Y. Su, P. Yin, C. Gomila, and A. Ortega. Adaptive filtering for cross-view prediction in multi-view video coding. In *Proc. SPIE 2007 Visual Communications and Image Processing (VCIP)*, Jan 2007.

[28] P. Lai, Y. Su, P. Yin, C. Gomila, and A. Ortega. Adaptive filtering for video coding with focus change. In *Proc. IEEE 2007 ICASSP*, volume I, pages 661–664, Apr 2007.

[29] H.-C. Lee. Review of image-blur models in a photographic system using the principles of optics. *SPIE Optical engineering*, 20, issue. 5:405–421, May 1990.

[30] G. Li and Y. He. A novel multi-view video coding scheme based on H.264. In *Proc. IEEE Joint Conference of the 4th International Conference on Information, Communications and Signal Processing, and the 4th Pacific Rim Conference on Multimedia*, volume 1, pages 493–497, Dec. 2003.

[31] J. Lopez, J-H. Kim, A. Ortega, and G. Ghen. Block-based illumination compensation and search techniques for multiview video coding. In *Proc. 2004 Picture Coding Symposium (PCS)*, San Francisco, CA, USA, Dec. 2004.

[32] E. Martinian, A. Behrens, J. Xin, A. Vetro, and H.-F. Sun. Extensions of H.264/AVC for multiview video compression. In *Proc. IEEE 2006 International Conference on Image Processing (ICIP)*, pages 2981–2984, Atlanta, GA, USA, Oct. 2006.

[33] W. Matusik and H. Pfister. 3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *Proc. ACM 2004 International Conference on Computer Graphics and Interactive Techniques archive (SIGGRAPH)*, pages 814–824, Los Angeles, CA, USA, Aug. 2004.

[34] P. Merkle, K. Muller, A. Smolic, and T. Wiegand. Efficient compression of multi-view video exploiting inter-view dependencies based on H.264/MPEG4-AVC. In *Proc. IEEE 2006 International Conference on Multimedia and Expo (ICME)*, pages 1717–1720, Toronto, Canada, Jul. 2006.

[35] R. P. Millane and J. L. Eads. Polynomial approximations to Bessel functions. *IEEE Trans. Antennas and Propagation*, 51, no.6:1398–1400, Jun 2003.

[36] P. Mouroulis and J. Macdonald. *Geometrical Optics and Optical Design*. Oxford Series in Optical and Imaging Sciences, 1996.

[37] S. Oka, T. Fujii, and M. Tanimoto. Dynamic ray-space coding using inter-view prediction. In *Proc. International Workshop on Advanced Image Technology (IWAIT) 2005*, pages 19–24, Jeju Island, Korea, Jan 2005.

[38] E. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26, no.2, Apr. 1984.

[39] J. Rissanen. A universal prior for integers and estimation by Minimum Description Length. *Institute of Mathematical Statistics Journal: Annals of Statistics*, 11, no.2:417–431, 1983.

[40] A. Smolic, K. Muller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand. 3D video and free viewpoint video - technologies, applications and MPEG standards. In *Proc. IEEE 2006 International Conference on Multimedia and Expo (ICME)*, pages 2161–2164, Toronto, Canada, Jul. 2006.

[41] K. Suhring. Software implementation of H.264: JM Version 9.6. *http://iphome.hhi.de/suehring/tml/index.htm*, this version was released in Jul. 2005.

[42] K. Suhring. Software implementation of H.264: JM Version 10.2. *http://iphome.hhi.de/suehring/tml/index.htm*, this version was released in Jul. 2006.

[43] G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *IEEE Signal Processing Magazine*, 15, Issue 6:74–90, Nov 1998.

[44] A. M. Tourapis, O. C. Au, and M. L. Liou. Highly efficient predictive zonal algorithms for fast block-matching motion estimation. *IEEE Trans. on Circuits and Systems for Video Technology (CSVT)*, 12, Issue 10:934–947, Oct 2002.

[45] Y. Vatis. Software implementation of adaptive interpolation filter. *http://iphome.hhi.de/suehring/tml/download/KTA/*, this software was released in Nov. 2005.

[46] Y. Vatis, B. Edler, D. T. Nguyen, and J. Ostermann. Motion-and aliasing-compensated prediction using a two-dimensional non-separable adaptive wiener interpolation filter. In *Proc. IEEE 2005 International Conference on Image Processing (ICIP)*, pages II.894–II.897, Genoa, Italy, Sep. 2005.

[47] Y. Vatis, B. Edler, I. Wassermann, D. T. Nguyen, and J. Ostermann. Coding of coefficients of two-dimensional non-separable adaptive Wiener interpolation filter. In *Proc. SPIE 2005 Visual Communication and Image Processing (VCIP)*, volume 5960, pages 623–631, San Jose, CA, Jul. 2005.

[48] Y. Vatis and J. Ostermann. Comparison of complexity between two-dimensional non-separabl adaptive interpolation filter and standard Wiener filter. *ITU-T SGI 6/Q.6 Doc. MPEG05/VCEG-AA11*, Apr 2005.

[49] Y. Vatis and J. Ostermann. Prediction of P- and B-frames using a two-dimensional non-separable adaptive Weiner interpolation filter for H.264/AVC. *ISO/IEC-JTC1/SC29/WG11 MPEG Document M13313*, Apr 2006.

[50] Z. Wang, G. Liu, and L. Liu. A fast and accurate video object detection and segmentation method in the compressed domain. In *Proc. IEEE International Conference on Neural Networks and Signal Processing*, pages II.1209–II.1212, Dec. 2003.

[51] T. Wedi. Adaptive interpolation filter for motion compensated prediction. In *Proc. IEEE 2002 International Conference on Image Processing (ICIP)*, pages II.509–II.512, Rochester, NY, Sep. 2002.

[52] T. Wedi. Adaptive interpolation filters and high-resolution displacements for video coding. *IEEE Trans. Circuits Systems and Video Technologies (CSVT)*, 16, no.4:484–491, Apr. 2006.

[53] T. Wiegand and B. Girod. Lagrange multiplier selection in hybrid video coder control. In *Proc. IEEE 2001 International Conference on Image Processing (ICIP)*, pages III.542–545, Thessaloniki, Greece, Oct 2001.

[54] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Systems and Video Technologies (CSVT)*, 13 no.7:560–576, Jul. 2003.

[55] K. Y. Wong and M. E. Spetsakis. Motion segmentation by EM clustering of good features. In *Proc. IEEE 2004 Computer Vision and Pattern Recognition Workshop (CVPR)*, pages 166–173, Jun. 2004.

[56] W. Yang, K. N. Ngan, and J. Cai. MPEG-4 based stereoscopic and multiview video coding. In *Proc. IEEE 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 61–64, Hong Kong, China, Oct. 2004.

[57] W. Yang, K. N. Ngan, J. Lim, and K. Sohn. Joint motion and disparity fields estimation for stereoscopic video sequences. *Elsevier Journal of Signal Processing: Image Communication*, 20, Issue.3:265–276, Mar. 2005.

[58] D. Zhang and G. Lu. Segmentation of moving objects in image sequence: A review. *Springer Journal of Circuits, Systems and Signal Processing*, 20, no.2:143–183, Mar. 2001.

[59] Z. Zhu, G. Jiang, M. Yu, and X. Wu. Fast disparity estimation algorithm for stereoscopic image sequence coding. In *Proc. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (TENCON)*, pages I.285–I.288, Beijing, China, Oct. 2002.