UNIVERSITY OF CALIFORNIA, SAN DIEGO

Comparison of Depth Image-Based Rendering and Image Domain Warping in 3D Video Coding

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Can Bal

Committee in charge:

Professor Truong Q. Nguyen, Chair
Professor Serge J. Belongie
Professor Pamela C. Cosman
Professor Sujit Dey
Professor Donald I. MacLeod

2014

The dissertation of Can Bal is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2014

DEDICATION

To my mother, father and grandmother

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge Prof. Truong Nguyen for his support as my advisor and the chair of my committee. His leadership and guidance during my studies have proved to be invaluable.

I would also like to thank Dr. Yan Ye, Dr. Aljoscha Smolic and Dr. Peng Yin for their mentorship and support. They have contributed to my education and career in an immeasurable way.

In addition, I want to thank all my friends for making San Diego feel like home and being there for me through hard times.

Finally, I would like to express my immense gratitude to my mother, father and grandmother. It is their patience and love that made all this possible.

Chapter 2, in part, has been submitted for publication of the material as it may appear in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen, Truong Q., IEEE, 2015 [1]. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in 3D Research. Tran, Lam C.; Bal, Can; Pal, Christopher J.; Nguyen, Truong Q., Springer, 2012 [2]. The dissertation author was among the primary investigators and authors of this paper.

Chapter 4, in full, is a reprint of the material as it appears in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen, Truong Q., IEEE, 2014 [3]. It is an extension of the material appearing in the Proceedings of the IEEE International Conference on Image Processing 2013. Bal, Can; Nguyen, Truong Q., IEEE, 2013 [4]. The dissertation author was the primary investigator and author of both of these papers.

Chapter 5, in full, has been submitted for publication of the material as it may appear in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen, Truong Q., IEEE, 2015 [1]. The dissertation author was the primary investigator and author of this paper.

VITA

2014	Ph.D. in Electrical Engineering (Signal and Image Processing), University of California, San Diego, CA
2010-2014	Gradate Student Researcher, University of California, San Diego, CA
2013	Intern, InterDigital Communications, San Diego, CA
2012	Intern, Disney Research Zurich, Zurich, Switzerland
2012	Intern, InterDigital Communications, San Diego, CA
2011	Intern, Dolby Laboratories, Burbank, CA
2009	M.S. in Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey
2007-2009	Research Assistant, Bilkent University, Ankara, Turkey
2007-2009	Teaching Assistant, Bilkent University, Ankara, Turkey
2007	B.S. in Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey

PUBLICATIONS

Can Bal and T. Q. Nguyen, "A Novel 3D Video Codec Based on 3D-AVC and Image Domain Warping," *IEEE Trans. Circuits Syst. Video Technol.*, submitted, unpublished.

Can Bal and T. Q. Nguyen, "Multiview Video Plus Depth Coding With Depth-Based Prediction Mode," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 995–1005, Jun. 2014.

Can Bal and T. Q. Nguyen, "Depth-based Prediction Mode for 3D Video Coding," in *IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2013, pp. 2187–2191.

N. Stefanoski, **Can Bal**, M. Lang, O. Wang, and A. Smolic, "Depth Estimation and Depth Enhancement by Diffusion of Depth Features," in *IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2013, pp. 1247–1251.

A. Jain, **Can Bal**, T. Q. Nguyen, "Tally: A Web-based Subjective Testing Tool", in *Int. Workshop on Quality of Multimedia Experience (QoMEX)*, Jul. 2013, pp. 128,129.

L. Tran, **Can Bal**, C. J. Pal, T. Q. Nguyen, "On consistent inter-view synthesis for autostereoscopic displays", *3D Research*, vol. 3, no. 1, pp. 1–10, Jan. 2012.

A. Jain, **Can Bal**, A. Robinson, D. MacLeod, T. Q. Nguyen, "Temporal Aspects of Binocular Suppression in 3D Video", in *Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM)*, 2012.

R. Khoshabeh, **Can Bal**, A. Jain, L. Tran, S. Chan, T. Nguyen, "Next-generation 3D: From Depth Estimation to the Display", *IEEE COMSOC MMTC E-Letter*, vol. 6, no. 8, pp. 32–36, Aug. 2011.

Can Bal, A. K. Jain, and T. Q. Nguyen, "Detection and removal of binocular luster in compressed 3D images," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2011, pp. 1345–1348.

PATENTS

Y. Ye, G. W. McClellan, Y. He, X. Xiu, Y. He, J. Dong, **Can Bal**, E. Ryu, "Codec architecture for multiple layer video coding", US Patent App. 13/937,645, Jan. 9, 2014.

T. Nguyen, L. Tran, **Can Bal**, and R. Khoshabeh, "Multi View Synthesis Method and Display Devices with Spatial and Inter-View Consistency," WO 2013/062944, May 3, 2013.

ABSTRACT OF THE DISSERTATION

Comparison of Depth Image-Based Rendering and Image Domain Warping in 3D Video Coding

by

Can Bal

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California, San Diego, 2014

Professor Truong Q. Nguyen, Chair

3D became successful in the movie theaters but failed to become mainstream for home use. The inconvenience of wearing glasses is arguably the reason and researchers have been investigating solutions for glasses-free 3D displays. Today, the most promising solution is the autostereoscopic display, which require many views of the same scene to be displayed simultaneously for a comfortable viewing experience. However, coding 3D video (3DV) with too many views is impractical with current networks and the 3DV coding standard, H.264/MVC (MVC), as the necessary bitrate is linearly proportional to the number of coded views. Instead, only a sparse set of anchor views can be compressed with some supplementary data and the remaining can be synthesized at the decoder.

In this dissertation, we compare two very popular view synthesis methods, Depth Image-Based Rendering (DIBR) and Image Domain Warping (IDW), in terms of their coding efficiency and complexity. First, we establish a common formulation that allows us to compare DIBR and IDW and their associated 3DV representations mathematically.

Then we provide the details of a fast DIBR-based view synthesis method and its implementation on GPU. We show that it can synthesize views with good objective quality and can provide inter-view consistency with almost constant time complexity in terms of the number of synthesized views.

Moreover, we present a new coding tool, "Depth-based Prediction Mode" (DBPM), and incorporate it into the coding loop of MVC. Using DBPM, we realize a novel MVD codec and we show that view synthesis can also be used for better prediction of the anchor views. DBPM uses the supplementary depth data and DIBR to achieve up to 9.2%, 9.9% and 6.7% bitrate savings over MVC for coding MVD data, depth maps and multiview videos, respectively.

Finally, we establish a codec framework based on the next generation candidate 3DV coding standard (3D-AVC), which has prediction tools similar to the DBPM already incorporated, and show that both DIBR and IDW can be used in this framework without any syntax changes to the standard. Using this framework we show that IDW achieves better coding performance than DIBR with average bitrate savings of 12.8% for anchor views and 1.5% for the synthesized views with significantly lower computational complexity. Finally, we provide an analysis on the effect of camera noise on measuring the quality of synthesized views with DIBR and IDW and show that camera noise produce a bias towards better measurements for DIBR. Recalculating the bitrate savings on sequences without camera noise shows that IDW can actually achieve average bitrate savings of 8.8% in the synthesized views instead of 1.5%.

Chapter 1

Introduction

In the past few years, 3D video technologies has been gaining interest with a high pace both in the movie industry and in consumer electronics. Nowadays almost all the blockbuster movies are released with a 3D option, and stereoscopic 3DTVs from all major brands are readily available on the market. With the interest in 3D, some broadcasters even launched 3D channels.

3D has proven to be very successful in the movie industry, but even with all of the high quality content, it did not achieve a similar success for home use. TV manufacturers and broadcasters seem to be less interested in 3D than they were a few years earlier. Some of previously mentioned 3D broadcast channels are even announced to shut down due to low ratings.

This is arguably because of the current 3D display technologies. Current mainstream 3D displays use polarization technology and require glasses for the 3D perception. This inconvenience is rather minuscule for occasional use at the movie theaters but it prevented 3DTVs to become mainstream. Additionally 3D glasses cause headaches and nausea for some people.

To overcome this obstacle, the research community and companies have been investigating solutions for glasses-free 3D displays. Currently, autostereoscopic displays are the most likely option to make it to the mass market. These displays use a regular LCD panel with a "lenticular sheet", which distributes the light emitted from different pixels in different directions. They display multiple views of the same scene simultaneously, where each of them is only visible from a

certain viewpoint. This allows the viewers to see a different stereo pair of views depending on their position, which allows them to see 3D without glasses. In order to deliver a comfortable viewing experience, autostereoscopic displays have to accommodate a large number of views. Some of them display up to 28 views [5] and this number is likely to grow as the resolution of TV panels increases.

The 3D video (3DV) representation supported by autostereoscopic displays is called *Multiview Video* (MVV). MVV representation has some inherent problems. First, capturing MVV content requires a synchronized camera rig with one camera for each viewpoint. Such setups are costly to build and maintain, impractical to calibrate, synchronize and operate, and usually immobile. Additionally, the number of views and the amount of depth that each 3D display can accommodate varies. Since MVV consists of fixed number of viewpoints at fixed locations, supporting a variety of displays is almost impossible with MVV.

The current standard 3DV compression method is the *Multiview Video Coding* (MVC) extension of H.264/AVC (AVC) [6]. Although MVC achieves significantly better compression in comparison to coding each view separately, MVC still produces bitrates linearly proportional to the number of views in MVV data [7]. Hence, using MVC and broadcasting MVV data to be displayed on autostereoscopic displays is unrealistic given the bandwidth of current networks.

To address the problems associated with MVV and MVC, the Moving Picture Experts Group (MPEG) issued a Call for Proposals (CfP) to develop a new 3DV data format and an associated compression method, which will enable synthesizing an arbitrary number of virtual views [8]. This will allow coding only a sparse set of anchor views and synthesizing the remaining views of MVV as virtual views at the display end. The caveat with this approach is that the view synthesis operation has to be both very fast and be able to synthesize good quality views in order to provide a good viewing experience on autostereoscopic displays.

Currently, the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V), a collaboration of International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and MPEG, is working on solutions based on the existing video coding standards: High Efficiency Video

Coding (HEVC) [9] and AVC.

JCT-3V is advancing the standardization process in two main categories based on the underlying codec architecture [10]: HEVC-based and AVC-based. The HEVC-based track includes MV-HEVC and 3D-HEVC. MV-HEVC will be the HEVC-based counterpart of MVC for MVV coding, whereas 3D-HEVC is more flexible and only requires the base view to be decodable by an HEVC decoder. In comparison, AVC-based track includes MVC+D and 3D-AVC. MVC+D (Depth) will be the standard for coding *Multiview Video + Depth* (MVD) data, which consists of MVV data and associated depth values for each of the texture pixels. As its name suggests, the requirement for MVC+D is the MVV data to be decodable by an MVC decoder. Finally, 3D-AVC is the AVC-based counterpart of 3D-HEVC and the only requirement is that the base view is decodable by an AVC decoder.

MVD is currently the most popular 3DV representation and is widely used in research. It is also the main representation in the standardization process. There are also alternative 3DV representations proposed to JCT-3V, such as Global View and Depth (GVD) [11] and *MVV + Warps* (MVW) [12], that are also being considered for the next generation 3DV codecs.

Each 3DV representation has an associated rendering method for synthesizing virtual views. MVD uses the well-known *Depth Image-Based Rendering* (DIBR) [13]; and since GVD is also a depth-based representation, it uses a modified version of DIBR as well [14]. On the other hand, the MVW representation is based on *warps* and requires a special rendering method called *Image Domain Warping* (IDW) [15, 16].

In this dissertation we compare two very popular rendering methods, DIBR and IDW in the context of 3DV coding. In Chapter 2, we first establish a common formulation for DIBR and IDW, and their associated view synthesis algorithms. This chapter lays the groundwork for the rest of the dissertation. Then, in Chapter 3, we provide the details of an advanced and fast DIBR-based view synthesis method and its implementation. This chapter serves as a proof of concept for the feasibility of fast view synthesis with good synthesis quality. Next, in Chapter 4 we present a new coding tool called "Depth-based Prediction Mode" (DBPM), and

show that it is possible to achieve coding gains over MVC using this new prediction mode. Our findings in this chapter support the need for a new 3DV representation and show that addition of some supplementary data in the 3DV representation can enhance the compression efficiency even when only the anchor views are considered. Finally, in Chapter 5 we establish a codec framework based on 3D-AVC, which has prediction tools similar to the DBPM already incorporated, and compare DIBR and IDW in terms of coding efficiency and computational complexity.

Chapter 2

Framework for Comparison of View Synthesis Algorithms

2.1 Introduction

View synthesis refers to a family of algorithms that generate an arbitrary number of virtual views at the display end from a number of anchor views and their associated supplementary data (e.g. depth data, camera parameters etc.). These algorithms may consist of processing blocks such as rendering, linear and non-linear filtering, blending and hole filling. Not all algorithms will contain all of these blocks, but the rendering operation that maps pixels of an anchor view onto the pixels of a virtual view is the core of all of them.

In this chapter, we build a unified formulation for two popular rendering methods in 3DV coding, DIBR and IDW and their associated view synthesis algorithms.

2.2 Depth and Disparity Relation

A multiview camera setup with two anchor cameras (gray) and a virtual camera (white) is depicted in Fig. 2.1. An anchor and a virtual camera form a stereo pair. Let us define these cameras as camera A and V with optical centers

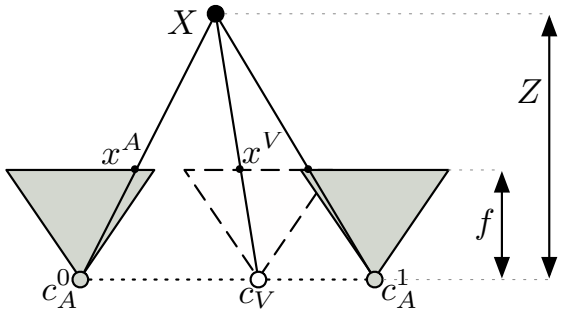


Figure 2.1: Rectified multiview camera setup with two anchor views (gray) and a virtual view (white).

c_A and c_V, and corresponding image planes I_A, I_V, respectively. Each pixel on I_A and I_V are projections of points in 3D space and are related to each other.

For the rectified camera setup, pixels of I_A and I_V are related to each other by a horizontal shift. The amount of shift $d \in \mathbb{R}$ is called *disparity*, and is inversely proportional to the depth of the corresponding point in 3D space (Z). Given the focal length (f) and the baseline between cameras ($c = c_V - c_A$), d can be calculated from Z as $d = (f \cdot c)/Z$.

Let $\mathbf{x} = (x, y) \in \mathbb{R}^2$ indicate a pixel on an image plane. The disparity between pixel $\mathbf{x}$ on I_A and the corresponding pixel on I_V is denoted as $d_{A \to V}(\mathbf{x}) \in \mathbb{R}$. Similarly, $d_{A \leftarrow V}$ is the disparity between pixels of I_V and I_A. In our subscript notation, the first and second labels indicate the real and virtual cameras, respectively, and the direction of the arrow indicates the direction of mapping.

Let X be a point in 3D space and $I_A(x^A, y)$ and $I_V(x^V, y)$ be the projections of X on image planes I_A and I_V. From the definition, $I_A(x^A, y) = I_V(x^V, y)$, where x^A and x^V are related to each other as:

$$d_{A \to V}(x^A, y) = x^A - x^V = -d_{A \leftarrow V}(x^V, y) \tag{2.1}$$

2.3 Depth Map

In order to represent the depth values in a format compatible with standard video coding methods, they are quantized and stored as n-bit monochromatic images called *depth maps*. The quantization is such that the depth values

$Z \in [Z_{\min}, Z_{\max}]$, where $Z_{\min}$ and $Z_{\max}$ are the minimum and maximum depth values that exist in a scene, of nearby objects have smaller quantization steps and greater magnitudes than the depths of farther objects.

Let D_A denote the depth map associated with I_A. Since depth and disparity are related to each other, it is possible to relate the values of D_A directly to the disparities $d_{A\to V}$ as:

$$D_A(\mathbf{x}) = \left\lfloor \frac{d_{A\to V}(\mathbf{x}) - o}{s} + 0.5 \right\rfloor \tag{2.2}$$

$$d_{A\to V}(\mathbf{x}) = s \cdot D_A(\mathbf{x}) + o \tag{2.3}$$

where

$$s = \frac{f \cdot c}{2^n}\left(\frac{1}{Z_{\text{near}}} - \frac{1}{Z_{\text{far}}}\right),\ o = \frac{f \cdot c}{Z_{\text{far}}} \tag{2.4}$$

2.4 Depth Image-Based Rendering

DIBR is the most popular rendering method in 3DV coding research. It takes a reference image I_A and an associated depth map D_A as inputs, and uses the depth information to map pixels of the reference image to points in 3D space. Then, it projects these points onto the virtual image plane $\widehat{I}_V$ to finalize the rendering operation, producing geometrically correct renders. Since some of the pixels on the virtual image plane do not have correspondences in the reference image (e.g. revealed background regions that are covered by foreground objects in the reference view), DIBR cannot fill the entire virtual image, which results in holes. These holes are then filled by other components of the view synthesis algorithm.

The first step for DIBR is to calculate the disparity map $d_{A\to V}$ from a given depth map D_A using Eq. (2.3). $d_{A\to V}$ maps the pixels of I_A to pixels on image plane $\widehat{I}_V$ as:

$$\widehat{I}_{A\to V}(x,y) = I_A(x^A, y) = I_A(x + d_{A\to V}(x^A, y), y) \tag{2.5}$$

$$\widehat{D}_{A\to V}(x,y) = D_A(x^A, y) = D_A(x + d_{A\to V}(x^A, y), y) \tag{2.6}$$

Eq. (2.5) can be used to render $\widehat{I}_{A\to V}$ directly, an operation we call "forward mapping". However, forward mapping yields a noisy render which contains errors

and missing pixels (appearing as cracks) due to depth quantization [2]. Instead, a virtual depth map corresponding to image plane $\widehat{I}_{A\to V}$ can first be rendered using Eq. (2.6) and the artifacts can be treated with a post-processing filter, such as median filtering [17] or by using "placement matrices" [2]. Due to their piece-wise smooth nature and monochromaticity, treating depth maps usually yields better results than treating images.

It is possible for two distinct pixels $\mathbf{x}^A$ and $\acute{\mathbf{x}}^A$ on D_A to map to the same pixel position on $\widehat{D}_{A\to V}$. In this case, the pixel closer to the camera is given preference:

$$\widehat{D}_{A\to V}(x,y) = \begin{cases} D_A(x + d_{A\to V}(\mathbf{x}^A), y), & \text{if } D_A(\mathbf{x}^A) > D_A(\acute{\mathbf{x}}^A). \\ D_A(x + d_{A\to V}(\acute{\mathbf{x}}^A), y), & \text{otherwise.} \end{cases} \tag{2.7}$$

Explicit tracking of these cases can actually be avoided by simply using Algorithm 1 instead.

Algorithm 1 Forward mapping algorithm.

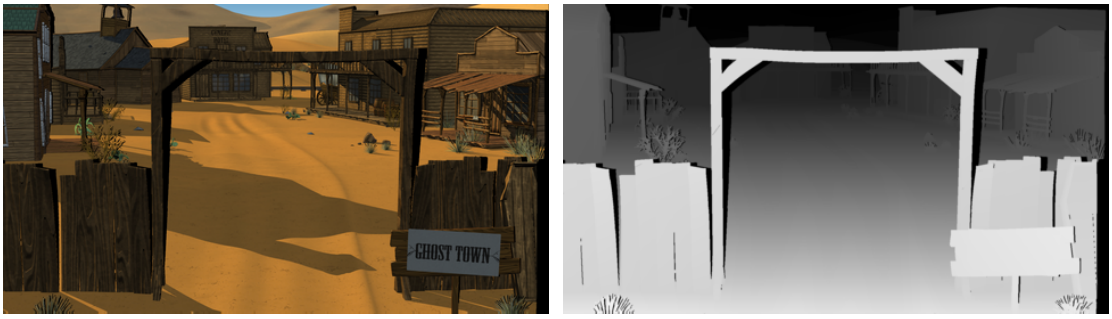
```
for D ∈ {0, ..., 2^{n-1}} do                ▷ scan through all depth map values
    d = s · D + o   ▷ calculate disparity value, s and o are defined in Eq. (2.2)
    for (x, y) where D_A == D do            ▷ find pixels where D_A is equal to D
        D̂_{A→V}(x − d, y) = D ▷ set the corresponding pixel value of D̂_{A→V} to D
    end for
end for
```

From $\widehat{D}_{A\to V}$ and Eq. (2.1) & (2.3), a disparity map $\widehat{d}_{A\leftarrow V}$ can be calculated that relates the pixels of I_A to pixels on $\widehat{I}_V$ as:

$$\widehat{I}_{A\to V}(x,y) = I_A(x - \widehat{d}_{A\leftarrow V}(x,y), y) \tag{2.8}$$

Once $\widehat{D}_{A\to V}$ is calculated, rendering $\widehat{I}_{A\to V}$ simply becomes a look-up operation, which we call "reverse mapping". This operation is important as all the rendering methods discussed in this chapter can ultimately be posed as a reverse mapping operation, where the look-up table differs depending on the rendering method.

A sample rendered image generated by DIBR and the associated disparity map used for the reverse mapping operation are shown in Fig. 2.2.



(a) $\widehat{I}_{A\to V}$ (b) $\widehat{d}_{A\leftarrow V}$

Figure 2.2: (a) Sample rendered image using DIBR and (b) the corresponding disparity map for the reverse mapping operation for *GTFly* sequence frame #150.

2.5 Reverse Depth Image-Based Rendering

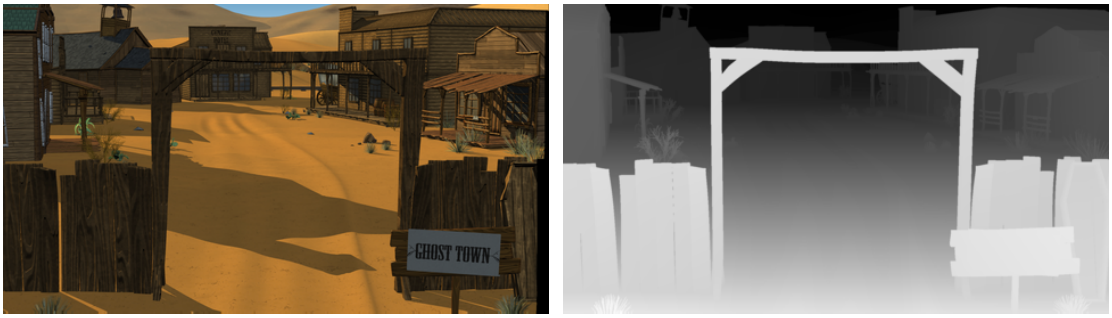
It is possible to have the depth map corresponding to the virtual view already available to the rendering method. This can happen when there are multiple anchor views and the rendering method is mapping pixels of one anchor view to another. In such a scenario it is possible to define a variant of DIBR and we call it "Reverse Depth Image-Based Rendering" (R-DIBR).

R-DIBR takes the depth map corresponding to the virtual view, D_V, as the input and maps the pixels of the virtual view to points in 3D space. Each of these points are then projected onto the reference image plane to find the correspondences between the reference and virtual image planes. Finally the corresponding pixels from the reference image are copied over to the virtual image plane to finalize the render. Assuming depth maps are error-free, R-DIBR is identical to DIBR for the non-occlusion regions. However R-DIBR cannot identify occlusion regions and fills them with incorrect pixels from the reference image.

For R-DIBR the first step is to calculate the disparity map $d_{V\to A}$ from the depth map D_V using Eq. (2.3). Once $d_{V\to A}$ is calculated, rendering operation simply becomes a reverse mapping operation defined as:

$$\widehat{I}_{A\to V}(x,y) = I_A(x - d_{V\to A}(x,y), y) \tag{2.9}$$

R-DIBR is used when the virtual rendered image serves as a reference in



(a) $\widehat{I}_{A\to V}$ (b) $d_{V\to A}$

Figure 2.3: (a) Sample rendered image using R-DIBR and (b) the corresponding disparity map for the reverse mapping operation for *GTFly* sequence frame #150.

a 3DV codec as it allows each pixel to be rendered independently. This feature allows the decoder to render only a region of interest on the virtual image plane thus reduces the decoder complexity significantly [18].

A sample rendered image generated by R-DIBR and the associated disparity map used for the reverse mapping operation are shown in Fig. 2.3.

2.6 Warps and Disparity

IDW, developed by Disney Research Zurich, is among the view synthesis algorithms proposed to JCT-3V [12]. Unlike DIBR, IDW does not rely on depth maps but uses warps to map pixels of the reference image onto a virtual image plane.

A warp operates over a quad grid defined on an image plane. A quad grid consists of quads Q defined by four quad vertices ($\mathbf{v} \in \mathbb{R}^2$). Note that these quad vertices are not necessarily aligned with the integer pixel grid.

A warp operator $\Psi_{A\to V}$ is a mapping from vertices of quads Q^A defined on the reference image plane I_A to the vertices of warped quads $\widetilde{Q}^V$ defined on the virtual image plane $\widetilde{I}_V$. In order to differentiate between different quads, subscripts $k \in [0,\ldots,K-2]$ and $l \in [0,\ldots,L-2]$ are introduced where K and L are the number of quad vertices in the horizontal and vertical directions. With this notation, $Q_{k,l}$ indicates the quad with vertices $\{\mathbf{v}_{k,l}, \mathbf{v}_{k+1,l}, \mathbf{v}_{k,l+1}, \mathbf{v}_{k+1,l+1}\}$ and is

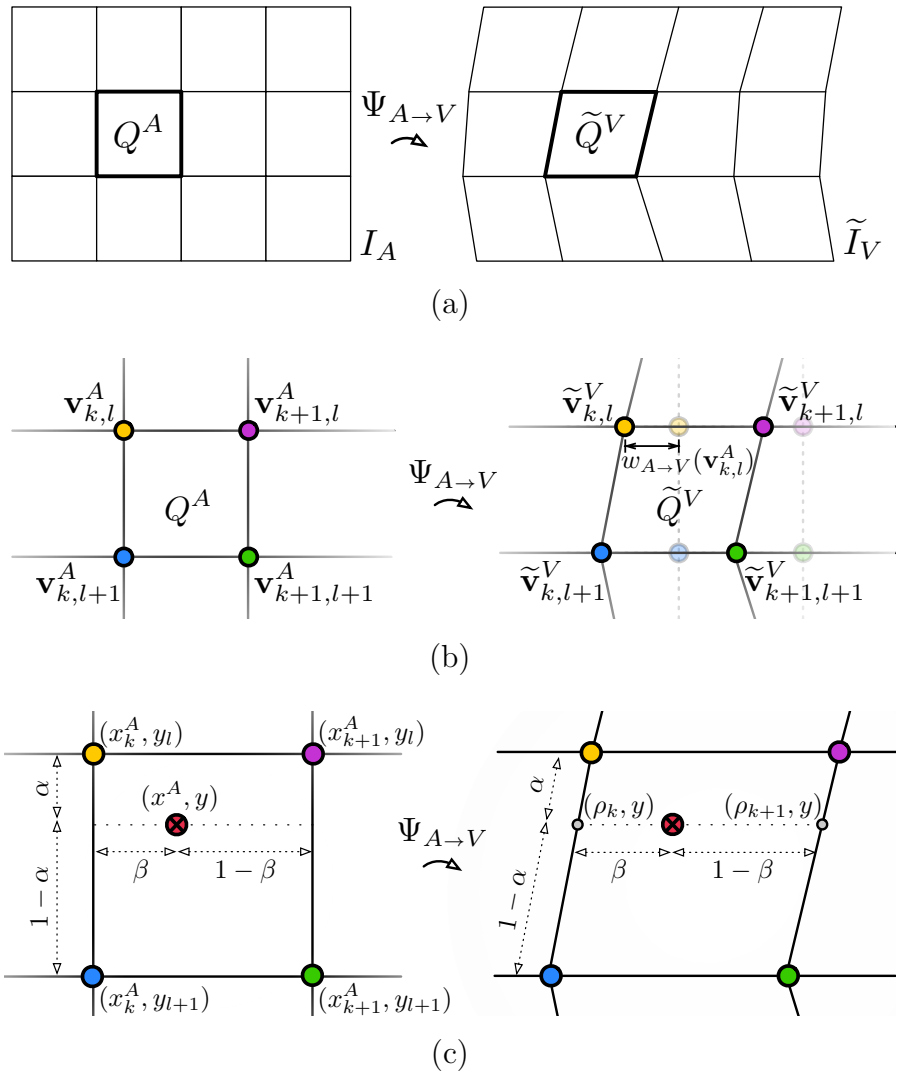


Figure 2.4: Illustration of Image Domain Warping for (a) an image (b) a quad (c) a pixel in a quad.

illustrated in Fig. 2.4.

For the rectified camera setup, the warp operator shifts the quad vertices only horizontally. Therefore, the warp operator $\Psi_{A\to V}$ is defined by the amount of horizontal disparities $w_{A\to V} \in \mathbb{R}$ between the corresponding quad vertices of I_A and $\widetilde{I}_V$. This is analogous to the disparities in DIBR, hence we use a similar notation. Let $\mathbf{v}^A_{k,l}$ be a quad vertex located at (x^A_k, y_l), and $\widetilde{\mathbf{v}}^V_{k,l}$ be the corresponding warped

quad vertex at $(\widetilde{x}_k^V, y_l)$. Then the warp from $\mathbf{v}_{k,l}^A$ to $\widetilde{\mathbf{v}}_{k,l}^V$ is defined as:

$$\begin{aligned}\Psi_{A\to V}(\mathbf{v}_{k,l}^A) &= \Psi_{A\to V}(x_k^A, y_l) = (\widetilde{x}_k^V, y_l) \\ &= (x_k^A - w_{A\to V}(x_k^A, y_l), y_l)\end{aligned} \tag{2.10}$$

$\Psi_{A\to V}$ defines a one-to-one correspondence between the quad vertices on I_A and $\widetilde{I}_V$; hence, an equivalent warp operator $\Psi_{A\leftarrow V}$ can be defined as a mapping in the opposite direction. Similarly for this operator, since $w_{A\to V}$ is the disparity between two vertices, the disparity in the opposite direction, $w_{A\leftarrow V}$, can be calculated as:

$$\widetilde{w}_{A\leftarrow V}(\widetilde{x}_k^V, y_l) = \widetilde{x}_k^V - x_k^A = -w_{A\to V}(x_k^A, y_l) \tag{2.11}$$

2.7 Image Domain Warping

IDW uses the warp operator $\Psi_{A\to V}$ to warp the pixels within each quad Q^A onto the pixels of the corresponding quad $\widetilde{Q}^V$, generating the rendered virtual image $\widetilde{I}_{A\to V}$ from the reference image I_A. This produces virtual images without holes (except for when the corresponding pixels are out of the reference image boundaries). This inevitably introduces geometrical distortions in the virtual image; hence, warps have to be carefully generated in order to introduce the geometrical distortions only around non-salient regions [15, 16].

From Eq. (2.11), a pixel $\widetilde{I}_{A\to V}(x, y)$ where $(x, y) \in \widetilde{Q}_{k,l}^V$ can be computed from the corresponding pixel (x^A, y) in $Q_{k,l}^A$ by the reverse mapping operation as:

$$\widetilde{I}_{A\to V}(x, y) = I_A(x^A, y) = I_A(x - \widetilde{w}_{A\leftarrow V}(x, y), y) \tag{2.12}$$

However, $w_{A\to V}$, and thus $\widetilde{w}_{A\leftarrow V}$ from Eq. (2.11), is only defined at the quad vertices. The disparity value $\widetilde{w}_{A\leftarrow V}(x, y)$ for an arbitrary pixel must be interpolated from the disparities at the vertices of $\widetilde{Q}_{k,l}^V$. Different interpolation methods can be used in this step. For example, Krähenbühl et al. [19] propose using EWA splatting. In our experiments, we found that simple bilinear interpolation suffices, yielding visually comparable results at a much lower computational cost to EWA splatting.

As illustrated in Fig. 2.4, for the rectified camera setup, bilinear interpolation can be handled in two steps. The first step is to calculate the locations of the

intermediate pixels (ρ_k, y) and (ρ_{k+1}, y), which lie on the line segments connecting left and right quad vertices respectively. Let $\alpha = (y - y_l)/(y_{l+1} - y_l)$. Then ρ_k is calculated as:

$$\rho_k = x_k^A - (1-\alpha) \cdot w_{A \to V}(x_k^A, y_l) - \alpha \cdot w_{A \to V}(x_k^A, y_{l+1})$$

Secondly, with $\beta = (x - \rho_k)/(\rho_{k+1} - \rho_k)$, the interpolated values of $\widetilde{w}_{A \leftarrow V}$ are simply:

$$\widetilde{w}_{A \leftarrow V}(x, y) = x - (1-\beta) \cdot x_k^A - \beta \cdot x_{k+1}^A$$

The interpolated $\widetilde{w}_{A \leftarrow V}$ is analogous to a full-resolution disparity map relating the virtual image plane to the reference. Therefore, the reverse mapping operation defined in Eq. (2.12) is actually equivalent to Eq. (2.8). With that, we can define a unified reverse mapping operation that describes the final rendering step for all the rendering methods discussed in this chapter as:

$$\breve{I}_{A \to V}(x, y) = I_A(x - \breve{d}_{A \leftarrow V}(x, y), y) \tag{2.13}$$

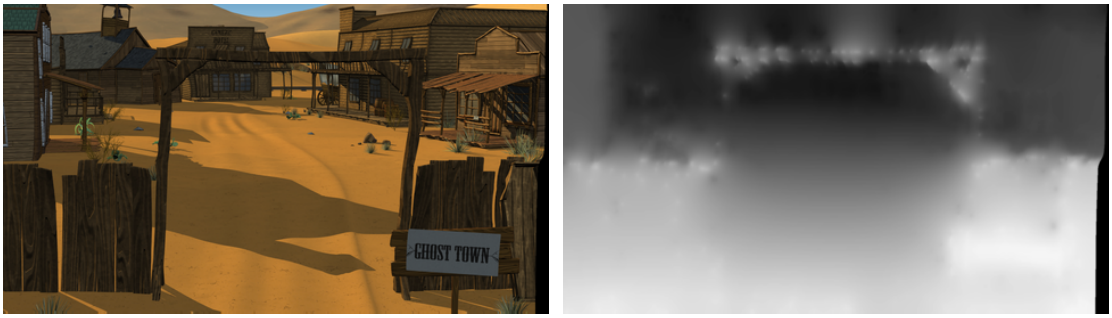
where $\breve{d}_{A \leftarrow V}$ is the interpolated $\widetilde{w}_{A \leftarrow V}$ for IDW, the rendered $\widehat{d}_{A \leftarrow V}$ for DIBR and the input disparity map $d_{V \to A}$ for R-DIBR. Please note that the symbol $\breve{}$ refers to a generic rendering operation in the rest of the paper, which is replaced by $\widetilde{}$ for IDW and $\widehat{}$ for DIBR.

A sample rendered image generated by IDW and the associated disparity map used for the reverse mapping operation are shown in Fig. 2.5.

2.8 Warp Map

For IDW, warp vertices $\mathbf{v}_{k,l}^A$ are positioned on a regular quad grid. Such a grid, illustrated on the image plane I_A in Fig. 2.4, consists of quads with fixed width and height and is determined by the number of vertices in the warp and the corresponding image resolution. For instance, for an image with $M \times N$ pixels and a warp with $K \times L$ vertices, the quad width and height are $M/(K-1)$ and $N/(L-1)$, respectively, and the vertices of the regular quad are defined as:

$$\mathbf{v}_{k,l}^A = (k \cdot \tfrac{M}{K-1}, l \cdot \tfrac{N}{L-1})$$



(a) $\widetilde{I}_{A\to V}$ (b) $\widetilde{w}_{A\leftarrow V}$

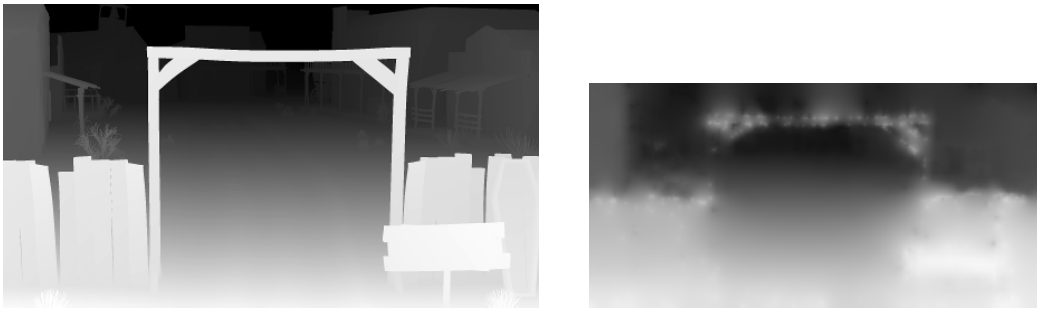
Figure 2.5: (a) Sample rendered image using IDW and (b) the corresponding disparity map for the reverse mapping operation for *GTFly* sequence frame #150.

For this choice of $\mathbf{v}^A_{k,l}$, the warp operator $\Psi_{A\to V}(\mathbf{v}^A_{k,l})$ is uniquely defined by only its vertex disparities $w_{A\to V}(x^A_k, y_l)$. This allows quantizing and storing warps as monochromatic images that can be compressed efficiently using standard video codecs. Stefanoski et al. [20] propose using linear quantization to quantize the vertex disparities and store them as an n-bit *warp map* (W) as:

$$W_A(\mathbf{v}) = \left\lfloor \frac{w_{A\to V}(\mathbf{v}) - o}{s} + 0.5 \right\rfloor \tag{2.14}$$

In our implementation, we used the same quantization scheme but chose the scale s and offset o parameters according to Eq. (2.2). This consistency with depth map quantization allowed us to use the depth-based coding tools and syntax of existing 3DV codecs without change. An example warp map with this choice of s and o is depicted in Fig. 2.6 along with a sample depth map corresponding to the same frame in the video.

The number of quad vertices are chosen such that a warp can be calculated robustly from a sparse set of constraints while providing enough granularity to avoid salient distortions around object boundaries. With this trade-off, warp maps typically have smaller resolutions than depth maps. For example, the warp maps calculated from the warp data provided by Stefanoski and Smolic [12] have resolutions of 401×201. In addition, as discussed earlier, warp maps do not enforce exact geometrical constraints; hence, they typically contain lower frequency content than the corresponding depth maps. These features, as shown in Fig. 2.6,



(a) (b)

Figure 2.6: Sample (a) 25% reduced resolution depth map (b) warp map corresponding to *GTFly* sequence frame #150 (sizes in proportion).

make warp maps ideal for block-based video coding as they can be compressed very efficiently.

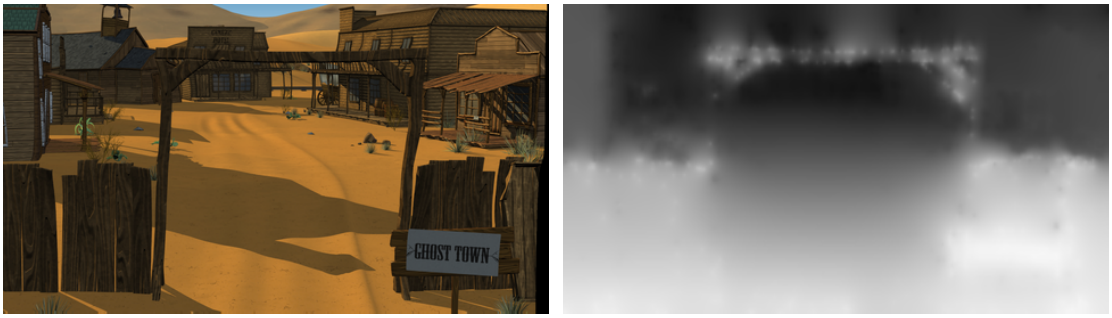
2.9 Reverse Image Domain Warping

Similar to R-DIBR, it is also possible to define a variant of IDW, which defines the regular quad grid on the virtual image plane $\widetilde{I}_V$ and we call this "Reverse Image Domain Warping" (R-IDW).

For R-IDW, the input warp operator $\Psi_{V\to A}$ warps the regular quads Q^V on the virtual image onto the warped quads $\widetilde{Q}^A$ on the reference image. This is equivalent to having a warp $\Psi_{V\leftarrow A}$ in the opposite direction operating on the already warped vertices $\widetilde{\mathbf{v}}^A = \Psi_{V\to A}(\mathbf{v}^V)$ of the quad grid on I_A. Hence, for rendering the virtual view $\widetilde{I}_V$, R-IDW is the same as IDW except the pixels within a regular quad Q^V are interpolated from the pixels falling into the warped quad $\widetilde{Q}^A$.

As the quads Q^V already have fixed dimensions, for R-IDW, the interpolated pixel disparities $\widetilde{w}_{V\to A}$ can be simply calculated by directly applying a standard image interpolation technique on the vertex disparities $w_{V\to A}$. Once $\widetilde{w}_{V\to A}$ is calculated, the virtual image is rendered using Eq. (2.13) with the reverse mapping operator $\breve{d}_{A\leftarrow V} = \widetilde{w}_{V\to A}$.

Defining the regular quad grid on the virtual image plane provides an im-



(a) $\widetilde{I}_{A\to V}$ (b) $\widetilde{w}_{V\to A}$

Figure 2.7: (a) Sample rendered image using R-IDW and (b) the corresponding disparity map for the reverse mapping operation for *GTFly* sequence frame #150.

portant advantage to R-IDW. For IDW, just to determine which warped quad a certain pixel belongs to, initially the warped quad vertices and quad boundaries have to be calculated on $\widetilde{I}_V$. On the other hand, for R-IDW this process is very simple due to the regularity of the quads, which makes R-IDW less computationally complex than IDW and more suitable for block-based prediction in 3DV compression.

A sample rendered image generated by R-IDW and the associated disparity map used for the reverse mapping operation are shown in Fig. 2.7.

Please note that R-IDW is only defined for the completeness of this chapter, but it has not been utilized in any of the projects discussed in this dissertation.

2.10 Blending and Stitching

In 3DV coding, the virtual views are typically synthesized from a stereo pair of anchor views. In this setup, rendering is repeated for each of the anchor views and these individual renders are then combined into the final render. This *blending* operation can mitigate potential artifacts or fill in holes introduced by the individual renders.

Let the optical centers for the left and the right anchor cameras be c_A^0 and c_A^1, respectively. Since we assume a rectified camera setup, the optical center of the virtual camera c_V is located in between c_A^0 and c_A^1. Without loss of generality,

let us assume $c_A^0 = 0$ and $c_A^1 = 1$, and $\theta \in (0, 1)$ be the baseline between c_A^0 and c_V. Then the final rendered image $\breve{I}_\theta$ corresponding to $c_V = \theta$ is calculated as:

$$\breve{I}_\theta(\mathbf{x}) = (1 - \gamma(\mathbf{x})) \cdot \breve{I}_{0\to\theta}(\mathbf{x}) + \gamma(\mathbf{x}) \cdot \breve{I}_{1\to\theta}(\mathbf{x}) \tag{2.15}$$

where γ is a blending operator that defines how individual renders are combined together. For DIBR γ is:

$$\widehat{\gamma}(\mathbf{x}) = \begin{cases} 0, & \text{if } \widehat{D}_{0\to\theta}(\mathbf{x}) > (\widehat{D}_{1\to\theta}(\mathbf{x}) + \tau) \text{ or } \widehat{I}_{1\to\theta}(\mathbf{x}) \text{ is a hole.} \\ 1, & \text{if } \widehat{D}_{1\to\theta}(\mathbf{x}) > (\widehat{D}_{0\to\theta}(\mathbf{x}) + \tau) \text{ or } \widehat{I}_{0\to\theta}(\mathbf{x}) \text{ is a hole.} \\ \theta, & \text{if } |\widehat{D}_{0\to\theta}(\mathbf{x}) - \widehat{D}_{1\to\theta}(\mathbf{x})| < \tau. \end{cases} \tag{2.16}$$

where $\tau > 0$ is a preset threshold. After blending, the image rendered with DIBR might still have holes left. A hole-filling operation is applied to these remaining holes to finalize the synthesis.

On the other hand, for IDW γ is much simpler and depends only on the virtual camera location θ:

$$\widetilde{\gamma}(\mathbf{x}) = \begin{cases} 0, & \text{if } \theta \leq 0.5 \text{ or } \widetilde{I}_{1\to\theta}(\mathbf{x}) \text{ is missing.} \\ 1, & \text{if } \theta > 0.5 \text{ or } \widetilde{I}_{0\to\theta}(\mathbf{x}) \text{ is missing.} \end{cases} \tag{2.17}$$

Eq. (2.17) simply states that the anchor view closer to the virtual view is used to render $\breve{I}_\theta$ completely, except for the missing pixels which are taken from the other anchor view. This is because the warps used for IDW introduce geometrical distortions around non-salient regions, and if a complex blending operation such as Eq. (2.16) is used, the distortions from each render clash and become salient in the final render. In order to differentiate the choices of γ, we will refer to the blending operation for IDW as "stitching" in the rest of the paper. Finally, unlike DIBR, after stitching IDW does not produce any holes, therefore does not require any additional steps to obtain the final synthesized view.

2.11 Visual Quality

DIBR and IDW generate very different virtual images, making objective quality metrics potentially unreliable for comparing the visual quality of the synthesized views. In order to address this issue, a number of subjective studies have

been conducted comparing DIBR and IDW with compressed data [21–23]. Some of these studies [21, 22] report that a significant majority of the subjects found no quality difference between DIBR and IDW, and the rest of the subjects presented no significant bias toward any of the rendering methods. Although these studies involved a rather small number of subjects, Stefanoski et al. [23] also present results for an extensive subjective study conducted by MPEG. Those results show that the IDW-based coding solution achieves comparable visual quality with the competing depth-based solutions and is actually among the top performers. Based on these findings, we conclude DIBR and IDW produce visually equivalent synthesized views, thus assume that objective quality metrics such as PSNR should provide reliable measurements for their direct comparison.

2.12 Conclusion

In this chapter we established a common formulation for DIBR and IDW, and showed that both rendering methods can be represented as a "reverse mapping" operation. The reverse mapping operation is defined by a per-pixel disparity map, which allows a direct comparison of DIBR and IDW. We also presented a "warp map" representation and a quantization scheme consistent with the depth maps, which allows warps to be coded using existing video codecs and depth-based coding tools.

To begin with, DIBR produces geometrically correct renders and naturally marks the occlusion regions in the renders, which allows a more advanced and geometrically correct blending operation.

In comparison, IDW does not generate occlusion regions. This eliminates the need of a hole filling operation, but inevitably introduces geometrical errors in the rendered images. Therefore, warps have to be carefully generated so that these geometrical errors are introduced only around non-salient regions.

Additionally, when the disparity maps for IDW and DIBR corresponding to the reverse mapping operations are compared, it can be observed that IDW generates renders that have lower frequency depth content. This is only partially due to

the resolution of the warps as reduced resolution depth maps still yield disparity maps with significantly higher frequency content. Warps inherently possess this characteristic and this allows warp maps to be compressed very efficiently using existing block-based video codecs.

Acknowledgments

This chapter, in part, has been submitted for publication of the material as it may appear in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen, Truong Q., IEEE, 2015 [1]. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Fast View Synthesis using CUDA and Depth Image-Based Rendering

3.1 Introduction

To provide an enjoyable depth perception on autostereoscopic displays, the baseline between each displayed view needs to be adjusted according to the specifications of the display, the distance of the user to the screen, and the degree of depth perception desired to be perceived by the user. Given all these constraints and the number of views required by the display, the necessary baseline between the anchor cameras typically has to be quite large.

A large baseline introduces the occlusion problem, where some of the regions that would be visible in the synthesized views are occluded in all of the captured views. This is where most DIBR-based view synthesis methods differentiate from each other. So far a significant number of different approaches have been proposed in the literature. There exist methods that range from very fast but simplistic to very advanced but slow.

Some of the view synthesis methods use a circular camera setup so that the occluded areas in one anchor view is almost always uncovered in another.

This type of camera setup allows a layered approach where a background layer is extracted from the anchor views and then used for filling in the holes in the virtual views [24, 25]. This approach is especially useful when the virtual cameras are to be positioned arbitrarily in the 3D space, as in the case of free viewpoint TV (FTV).

On the other hand, capture systems with a circular camera setup are not always preferable depending on the application. For example, for autostereoscopic displays, the cameras corresponding to the displayed views have to be positioned along a line or a wide arc. This setup is called the rectified camera setup. However when the application requires rectified viewpoints, capturing anchor views with a circular camera setup brings unnecessary complexity to the depth estimation and view synthesis processes.

For rectified camera setups, the occlusion regions are filled by hole filling algorithms that aim to generate realistic looking texture data. Some of these algorithms process the occlusion regions pixel-by-pixel while others use patch-based inpainting. The pixel-based algorithms tend to use interpolation or simple inpainting methods but they typically suffer from blur [26, 27]. Some of the pixel-based algorithms employ more advanced techniques and optimization steps to overcome this problem [17, 28] but patch-based algorithms are usually superior [29, 30]. Finally, some algorithms take it a step further and can also provide spatial, temporal [31] and inter-view consistency [29] for the generated texture data.

As view synthesis methods are intended to be used for 3DV playback on autostereoscopic displays, it is crucial that they are capable of synthesizing many views in real-time. Typically view synthesis methods that employ more complex hole filling algorithms fail to provide fast enough synthesis and simpler methods are not able to create visually pleasing synthesized views for a good user experience.

In the recent years, CPU clock speeds have been saturated and chip manufacturers started to focus on increasing the number of cores on each chip rather than trying to make each core faster. This raised the interest in parallel computing. More recently, *Graphics Processing Unit* (GPU) processors have enabled real-time implementation of many algorithms by housing hundreds of cores on a

single GPU. This advancement brought excitement about using GPUs for scientific calculations and GPU manufacturers started to invest in developing specific programming languages to make GPUs more accessible for scientific computing.

In order to take advantage of the multiple cores on a GPU, an algorithm needs to be highly parallelizable. Luckily many image processing tasks are well suited for this as each pixel or group of pixels can be processed separately. Most processes in the view synthesis methods are also parallelizable; therefore some researchers proposed taking advantage of the processing power of the current GPUs to significantly speed-up synthesis process [32, 33]. These existing methods can achieve very fast view synthesis, but they are rather limited to very simple approaches.

In order to provide fast and also high quality view synthesis, in this chapter, we propose an advanced DIBR-based view synthesis method and analyze its performance for a CUDA implementation. CUDA is the programming language Nvidia chips support and is the most commonly used general purpose GPU programming language today.

3.2 View Synthesis using CUDA

The proposed view synthesis method combines the "coordinate alignment" approach introduced by Tran et al. [29] with the background layer idea mentioned earlier in order to provide inter-view consistency and fast hole filling with a patch-based depth-assisted inpainting method. A high level block diagram of the proposed method is provided in Fig. 3.1.

3.2.1 Rendering Foreground Views

Placement Matrix

As described in Section 2.4, the first step for DIBR-based view synthesis is calculating the reverse mapping operator $\widehat{d}_{A\leftarrow V}$. Due to the quantization of depth map values (Eq. (2.2)) $\widehat{d}_{A\leftarrow V}$ might contain errors and missing pixels appearing as cracks. In the proposed method, before the reverse mapping operation, we

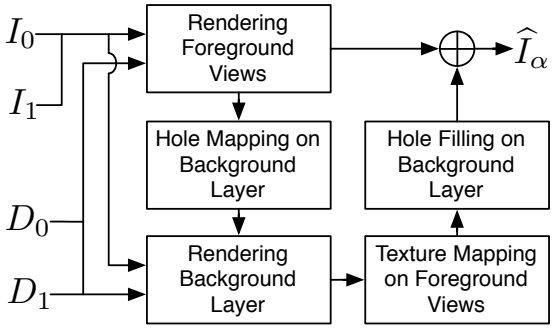


Figure 3.1: Block diagram of the proposed view synthesis method.

(a) (b)

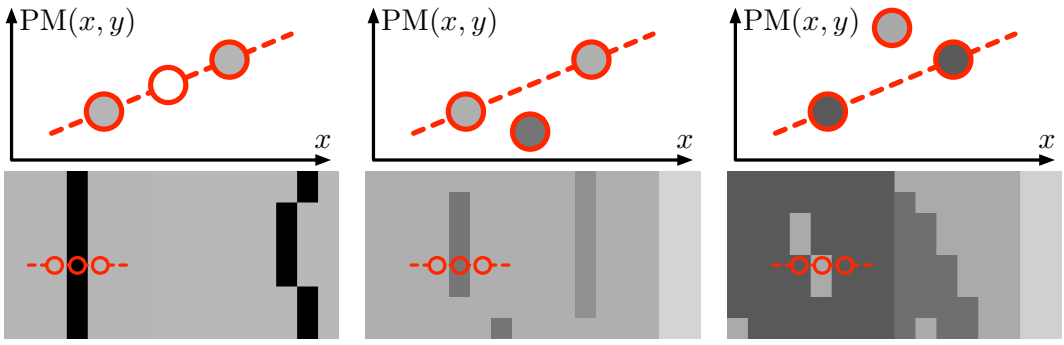
Figure 3.2: An example of a placement matrix (a) with cracks due to quantized disparity map and (b) after refinement procedure.

first form a "placement matrix" to treat these artifacts. Placement matrices are introduced by Tran et al. [28] for a more robust rendering process for the virtual view $\widehat{I}_{A\to V}$. The corresponding placement matrix for $\widehat{I}_{A\to V}$ is defined as:

$$\mathrm{PM}_{A\to V}(x,y) = x - \widehat{d}_{A\leftarrow V}(x,y) \tag{3.1}$$

The placement matrix $\mathrm{PM}_{A\to V}(x,y) = \acute{x}$ determines the pixel $I_A(\acute{x},y)$ that should be mapped to the pixel $\widehat{I}_{A\to V}(x,y)$ after the reverse mapping operation. An example of a placement matrix with the mentioned cracks is depicted in Fig. 3.2(a). For this specific example, the cracks are most salient on the face of the sculpture and on the rings located on the right-hand side.

Due to the construction of placement matrices and as neighboring pixels



(a) No value (b) Low value (c) High value

Figure 3.3: Three possible cases that cause cracks on placement matrices.

are expected to have similar depth values, on each row (fixed y) we expect to observe non-decreasing PM values for increasing x, with rapid jumps around object boundaries. The discussed cracks contradict to this and are easy to detect on the placement matrices. They appear in three different forms and these are depicted in Fig. 3.3. In this figure, on the bottom half there are 16×8 pixel sections from the placement matrices. The red lines on these sections depict the rows and three neighboring PM values where the cracks appear. On the top half the corresponding values of the PM that lead to a crack is illustrated. The first case happens when there is a missing PM value in between two known values. The second case happens when a background object fills in a crack although that value should correspond to a foreground object, and the last case happens when the opposite happens. In each case, we simply replace these middle values with the average of the neighboring two values. Fig. 3.2(b) shows the placement matrix after this refinement step.

After the placement matrices are formed and refined, the pixels of I_A are mapped to the image plane $\widehat{I}_V$ as:

$$\widehat{I}_{A\to V}(x,y) = I_A(\mathrm{PM}_{A\to V}(x,y),y)$$
$$\widehat{D}_{A\to V}(x,y) = D_A(\mathrm{PM}_{A\to V}(x,y),y)$$

(a) (b)

Figure 3.4: The blended render (a) with ghost artifact (b) after refinement.

Blending

For the stereo anchor view case, the individual renders from each anchor view are blended together by using Eq. (2.15) and Eq. (2.16). In order to simplify the blending operation for a faster implementation, τ is chosen to be 0 in Eq. (2.16). This operation corresponds to the following: If there is only one candidate pixel for a certain location, we simply use that. Otherwise we choose the one that is closer to the camera. We also keep track of the locations that have two candidate pixels for later use.

Refinement

Usually depth values are unreliable at the object boundaries, and after the blending operation this may cause "ghost artifacts". This problem is depicted in Fig. 3.4(a), and it is most salient in the background region around the ring on the upper right hand corner.

In order to repair the ghost artifacts, we apply a refinement step after the blending operation. For the locations where we had two candidate pixels, we apply the following refinement operation and decide on the final value of the blended

image:

$$\widehat{I}_{\theta}(\mathbf{x}) = \widehat{I}_{A^{*}\rightarrow V}(\mathbf{x}) \text{ where } A^{*} = \underset{A\in\{0,1\}}{\arg\min} \sum_{m=-1}^{1} \sum_{\substack{n=-1 \\ n\neq m=0}}^{1} |\widehat{I}_{\theta}(x-m, y-n) - \widehat{I}_{A\rightarrow\theta}(x,y)|$$

This operation essentially checks the likelihood of the candidate values of a pixel with respect to its 8-point neighborhood. Then it chooses the one that is more likely among the two candidates. This refinement step needs to get executed multiple times before all the ghost images are removed. In our experiments we found 5 or fewer iterations to be sufficient. Fig. 3.4(b) shows the the blended image after the refinement step is applied.

Hole Dilation

As mentioned before, depth values along the object boundaries are unreliable. This also causes problems around hole boundaries. As it can be seen in Fig. 3.4(a), the boundaries of the holes contain textures from other objects. This becomes a problem during hole filling if not treated prior to it. In order to remove these pixels, we simply apply a morphological dilation operation to the holes, specifically using a disk with radius of 3 pixels in our method.

3.2.2 Rendering Background Layer

Once the foreground views are rendered, generally DIBR-based methods apply the hole filling algorithm directly on the rendered foreground views [17, 28]. This is not optimal since holes are most likely to belong to the background, but the background pixels are partially covered by the foreground objects after the rendering process. Since some of the important background pixels might be covered in the rendered foreground views, hole filling algorithm cannot utilize all the important information available in the captured views.

In our method, we account for this problem by synthesizing a background layer as well that combines all the pixels belonging to background region onto a single layer. This is similar to the previously mentioned background layer

Figure 3.5: An example of a synthesized background layer.

idea [24, 25]. In order to do that, the pixel mapping steps described for foreground views are repeated, except the pixels that are farther away from the camera are mapped over the closer ones. Mathematically this requires the Eq. (2.7) and Eq. (2.16) to be changed to Eq. (3.2) and Eq. (3.3), respectively. An example of the background layer is depicted in Fig. 3.5.

$$\widehat{D}_{A\to V}(\mathbf{x}) = \begin{cases} D_A(x + d_{A\to V}(\mathbf{x}^A)), & \text{if } D_A(\mathbf{x}^A) < D_A(\acute{\mathbf{x}}^A). \\ D_A(x + d_{A\to V}(\acute{\mathbf{x}}^A)), & \text{otherwise.} \end{cases} \tag{3.2}$$

$$\widehat{\gamma}(\mathbf{x}) = \begin{cases} 0, & \text{if } \widehat{D}_{0\to\theta}(\mathbf{x}) < \widehat{D}_{1\to\theta}(\mathbf{x}) \text{ or } \widehat{I}_{1\to\theta}(\mathbf{x}) \text{ is a hole.} \\ 1, & \text{if } \widehat{D}_{1\to\theta}(\mathbf{x}) < \widehat{D}_{0\to\theta}(\mathbf{x}) \text{ or } \widehat{I}_{0\to\theta}(\mathbf{x}) \text{ is a hole.} \end{cases} \tag{3.3}$$

3.2.3 Hole Mapping on Background Layer

In section 3.2.1 rendering an initial foreground view is discussed. Before proceeding onto the hole mapping step of the proposed method, first multiple views corresponding to different virtual camera positions are rendered.

All the rendered initial foreground views have a number of holes, but some of these holes actually correspond to the same occlusion regions in other views (i.e. same uncovered region in the background). In order to keep the synthesized texture consistent for different rendered views, these holes need to be filled together. Tran et al. [29] handles this problem by explicitly aligning the coordinates

of the corresponding holes in different views. This operation requires a lot of computation since every hole in each view needs to be compared to all the holes in the other views.

Instead, we propose a simpler approach where we first decide on a depth level for each of the holes in each rendered foreground view. This determines where these holes would be positioned on the background layer. Once we know these positions, we locate the holes on the background layer and mark the corresponding pixels as holes. This step essentially overlays the corresponding holes from each foreground view on the background layer, which are then filled as a single hole in the hole filling step.

For deciding on the depth level of the holes, we check for the depth values on the boundary of the hole and simply select the minimum value encountered. This choice of depth level forces the holes to be filled in with the neighboring background texture.

3.2.4 Hole Filling

After locating all the holes on the background layer, we apply the hole filling algorithm on the background layer. This allows us to utilize all of the background information available from the anchor views in order to fill in the missing regions.

In the proposed method, we use the very popular patch-based inpainting algorithm called "Exemplar based Inpainting" [34] for the hole filling step. Briefly, this technique fills in the target region (Ω), from a source region (Φ) in an image by copying over an appropriate patch from the source region. Fig. 3.6 shows an illustration of this setting. The patches are copied onto the target region one by one, and centered on a pixel on the contour of the fill region, or the so-called "fill front" ($\delta\Omega$), in each iteration of the algorithm.

In our implementation, we directly applied the method proposed by Criminisi et al. [34], except we did not use $\Phi = I - \Omega$ (I signifies the whole image). Instead we calculate Φ as follows: Let D_{BG} be the depth map of the background layer and D_{Ω} be the minimum depth map value along the boundary of the hole. Then in order to incorporate the depth information within the inpainting process,

Figure 3.6: The source region, fill region and fill front setting used in inpainting.

(a) (b)

Figure 3.7: Background layer (a) with mapped holes from foreground views (b) after holes are filled.

we choose $\Phi = |D_{\mathrm{BG}} - D_{\Omega}| < \zeta \cdot |s|^{-1}$, where s is defined in Eq. (2.4) and ζ is a constant threshold and is taken to be 3 in our experiments.

The background layer after the hole filling step is shown in Fig. 3.7(b).

3.2.5 Texture Mapping on Foreground Views

Once the holes are filled on the background layer, then the newly synthesized textures are simply mapped back onto the previously rendered foreground views. In Fig. 3.8, the final synthesized views for camera positions $\theta \in \{0.25, 0.5, 0.75\}$ are shown.

(a)

(b)

(c)

Figure 3.8: Synthesized (left) and corresponding captured (right) views at position (a) $\alpha = 0.25$ (b) $\alpha = 0.5$ (c) $\alpha = 0.75$.

3.2.6 Generating Additional Views

After the view synthesis process is completed, with the proposed method any number of additional views can be synthesized with low computational complexity as only a subset of the previously described steps need to be executed.

As it has already been calculated, the background layer is utilized directly while synthesizing new views, which allows the synthesis process to skip the cumbersome hole filling operation. This also ensures that the holes of the new views are filled with texture that is consistent with the previously synthesized views.

With the proposed method, an additional view is synthesized simply as follows: First the new foreground view is rendered with holes. Then each hole is assigned a depth level, which determines its position on the background layer. Once the hole is positioned on the background layer, it is simply filled by copying the corresponding pixels from the background layer.

3.3 Experimental Results

3.3.1 View Synthesis Quality

The proposed method is tested on the well known Middlebury dataset [35], and the quality of the synthesized views is measured using PSNR. There are other view synthesis methods that use the same experimental setup as ours [17, 28, 29], therefore we use their reported results as reference. In our experiments, for each scene in the dataset, we synthesized three views positioned at $\theta = \{0.25, 0.5, 0.75\}$, but some of the reference view synthesis methods [28, 29] only report results for position $\theta = 0.5$. Therefore for direct comparison with the competing methods, we also only report the results for position $\theta = 0.5$ in this section.

According to the results in Table 3.1, the proposed method is not the top performer. However it can synthesize views with competitive object quality to the best performing methods. In average the quality of the synthesized views with the top performing method [29] is only 0.47 dB better in PSNR than the proposed method.

Table 3.1: View synthesis quality comparison of the proposed method.

	Jain et al. [17]	Tran et al. [28]	Tran et al. [29]	Proposed
Image Name	PSNR (dB)	PSNR (dB)	PSNR (dB)	PSNR (dB)
Art	31.63	32.66	**32.82**	32.09
Books	30.23	30.92	**31.06**	30.15
Cloth1	35.00	**35.99**	**35.99**	35.68
Dolls	31.56	33.05	**33.22**	31.95
Laundry	32.05	32.13	**32.16**	31.63
Moebius	33.35	34.30	**34.58**	33.92
Monopoly	30.14	32.19	32.27	**32.65**
Plastic	37.79	37.77	34.10	**38.09**
Reindeer	33.40	33.70	**37.90**	33.40
Wood1	36.34	37.47	**37.50**	37.36
Average	33.15	34.02	**34.16**	33.69

3.3.2 Execution Speed

A fast view synthesis method is crucial for supporting autostereoscopic displays with many views. Hence, we also provide the details of the execution time for our method. We implemented the proposed method in CUDA and ran the experiments on an Nvidia 580GTX graphics card with 512 CUDA cores and 1536MB memory. For the images listed in Table 3.1, the execution times for each main block of the proposed method are shown in Fig. 3.9.

As it can be seen from Fig. 3.9, the bottleneck of our method is the hole filling algorithm. That is because it can not be parallelized as it fills the holes sequentially, only one patch at a time. On the other hand, other parts of the proposed method can be parallelized to take advantage of the processing power of the GPU. Thus once the background layer is synthesized, an additional view can be generated in almost real-time as the proposed algorithm can skip the hole filling step.

The view synthesis methods that we compare the proposed method against also report their execution speeds. Our implementation is the fastest among all of them; however this is not a fair comparison as they are all implemented in different programming languages. Instead we compare the complexity of each method with

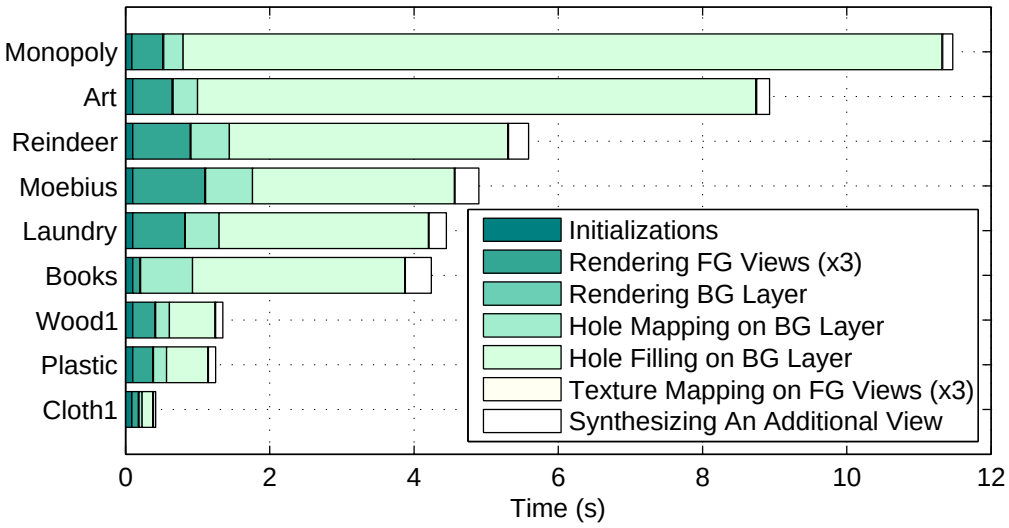


Figure 3.9: The execution time of the main blocks of the proposed view synthesis method.

respect to the number of synthesized views.

The complexity of the method proposed by Jain et al. [17] is almost linear with the number of synthesized views. On the other hand, due to the coordinate alignment step, the method proposed by Tran et al. [29] requires even more time, which is a polynomial of the number of synthesized views. In comparison, the proposed method first uses a small number of views (3 in our experiments) to generate the background layer, and then each additional view can be synthesized using the same background layer. Therefore the complexity of the proposed method can be assumed to be almost constant with the number of synthesized views since the time required by synthesizing a new view is negligible in comparison to the time required by the hole filling step. This makes the proposed method ideal to be used with autostereoscopic displays that require many viewpoints.

To demonstrate the power of the proposed method, we also implemented an interface that allows the user to adjust the point of view and the depth content of a 3D image interactively in real-time. This interface was only for demo purposes and the details of it is omitted in this chapter.

3.4 Conclusion

In this chapter, we proposed a fast view synthesis method that synthesizes views with good objective quality and also provides inter-view consistency. Among the compared methods, the proposed method produces PSNR values only 0.47dB lower on average than the top performer but has almost constant time complexity in terms of the synthesized views. In comparison, the better performing two methods have linear and polynomial time complexity.

Due to its efficiency and speed, the proposed method is useful for generating content for autostereoscopic displays that require many viewpoints. It can also be used to support different displays with different specifications as it allows synthesizing any number of views very efficiently and can provide an interactive experience to the user.

Acknowledgments

This chapter, in full, is a reprint of the material as it appears in 3D Research. Tran, Lam C.; Bal, Can; Pal, Christopher J.; Nguyen, Truong Q., Springer, 2012 [2]. The dissertation author was among the primary investigators and authors of this paper.

Chapter 4

Multiview Video plus Depth Coding with Depth-based Prediction Mode

4.1 Introduction

Similar to H.264/AVC (or simply AVC), the current 3DV compression standard H.264/MVC (or simply MVC) uses block-based prediction modes and exploits the statistical correlation between views as well as the temporal redundancies within a view. In addition to AVC intra and motion-compensated prediction (MCP) modes, MVC introduces a new *disparity-compensated prediction* (DCP) mode. DCP requires only high-level syntax changes to the AVC standard, making the implementation an easy task given AVC implementations (both in software and hardware) are readily available. DCP operates similarly to MCP, with the only difference that DCP references previously decoded frames of other views whereas MCP references the frames within the same view.

Although the addition of DCP can provide significant bitrate savings over separate coding of views with AVC, as discussed in Chapter 1, bitrates generated by MVC are linearly proportional to the number of encoded views [7]. Moreover, the multiview video (MVV) representation restricts the number and location of

the views to the captured data. To address these limitations the multiview video plus depth (MVD) representation has been proposed.

The addition of depth data to the 3DV representation created new challenges for compression. This triggered interest in the research community to develop new coding tools tailored to the characteristics of the new representation.

First, depth maps consist of piece-wise smooth regions with sharp edges around the object boundaries. Existing lossy compression techniques proved to be competent for coding large textureless regions within depth maps, yet they tend to flatten the depth edges after the transform and quantization operations. This causes unnatural geometrical distortions to the synthesized views. In order to account for this, new coding tools that yield sharper edges along the depth discontinuities have been proposed [36–41]. In addition to these, methods that increase the coding efficiency for smooth depth regions are also developed. Some researchers proposed using depth specific prediction modes [42], and others showed that reduced resolution depth maps can provide coding gains while maintaining good synthesis quality [43]. JCT-3V mandates to use reduced resolution depth maps in the common test conditions for the 3DV standardization process [44].

Second, in MVD representation, texture videos and depth maps contain significant statistical correlations and it is possible to exploit these for better compression. For instance, there are coding methods that recognize structural similarity, and recycle block partitioning information from coded texture videos for coding the depth maps [37, 45]. Additionally, there are methods that take advantage of the shared motion between the texture and depth data [38, 46–49] in order to improve coding efficiency.

While coding MVD data, the encoder has to make certain optimizations to provide the best quality for the given bitrate constraints. Initially, the encoder needs to decide on the allocation of the total bitrate between the texture video and the depth maps. Various solutions that address this problem have been proposed in the literature [50–54]. Once the bitrate budget for the depth maps is determined, the depth encoder uses rate-distortion optimization (RDO) to decide between the available coding tools to provide the best possible quality. However, since depth

maps are never seen by the end users, but are used for virtual view synthesis, measuring the depth map quality directly using video quality metrics such as PSNR can be misleading. Instead, researchers proposed new metrics and models that measure the geometrical distortions introduced in the synthesized views to assist the RDO process [55–59].

Finally, there are coding methods that take advantage of the multiview geometry between different views of 3DV in order to increase the overall coding efficiency. The depth-based 3DV representation enables synthesis of virtual views, which can also be utilized as a means of prediction within the codec. The typical view synthesis consists of the following steps: Initially pixels of the reference views get mapped and are blended together on a common virtual image plane. Depending on the scene geometry, some of these pixels on the virtual image plane may not have correspondences in the reference views (i.e. revealed background regions that are covered by foreground objects in the reference views). These regions are then filled by hole filling algorithms. Additionally, synthesis algorithms may employ linear and/or non-linear filtering operations to treat potential synthesis artifacts.

Among these methods, Martinian et al. propose to insert synthesized virtual frames in the reference picture buffer (RPB) and show that this can increase coding efficiency even with a simple extension to MVC [60]. This approach requires using the existing macroblock level signaling, and transmitting disparity vectors and reference picture IDs to the decoder for the synthesized references. However, if depth maps are reliable signaling the disparity vectors is redundant as the pixels of the synthesized reference are expected to be aligned already with the frame being predicted. Thus, others extend this approach by adding new DIRECT/SKIP modes to the codec that are specific to the synthesized reference [61–63]. These methods report that these modes get chosen the majority of the time [61] allowing the encoder to signal "view synthesis prediction" (VSP) very efficiently.

Apart from others, in order to reduce the complexity of the decoding process for VSP, Jäger and Feldmann propose to use only the base view information for synthesizing the virtual frames and skip the computationally expensive steps of full view synthesis, such as the blending and hole filling operations [63]. They also

implement an alternative decision mechanism instead of the typical Lagrangian RDO process to decide on using their proposed SKIP mode.

Contrarily, Bosse et al. propose a depth-aware encoder control method to utilize view synthesis for bitrate savings without changing the codec syntax [64]. Their proposed method identifies reliable regions from the synthesized virtual frames and encodes them with lower fidelity by omitting the coding of redundant residual information. These regions are then detected at the decoder and enhanced in a post-processing step using the synthesized virtual frames.

All the mentioned methods so far rely on *forward* mapping of anchor view texture pixels to the virtual image plane, using the depth map belonging to the anchor view. Alternatively, Rusanovskyy et al. propose to use *backwards* view synthesis prediction (B-VSP), which uses the depth map belonging to the view to be coded to find the correspondences between the reference views and that particular view [65]. The advantage of B-VSP is that it does not require an entire virtual view to be synthesized for decoding. Instead, it allows individual blocks to be decoded separately only from the information contained in the collocated block of its associated depth map. This reduces the complexity of the decoder over forward VSP methods. On the other hand, it requires the depth maps for all the anchor views to be contained in the bitstream, and each corresponding depth map to be decoded prior to decoding the anchor texture videos (except for the base view).

VSP is shown to be effective for coding texture videos, but it can also be utilized for depth map coding. As an example, Morvan et al. propose a simple extension to the H.264/AVC codec for depth map coding, and report bitrate savings by using a synthesized reference for prediction [66]. They insert a synthesized depth frame to the RPB and position it such that the SKIP mode uses this frame as the reference instead. They test for different synthesis algorithms with varying complexities and show a trade-off between computational complexity and coding efficiency.

In this chapter we propose a "depth-based prediction mode" (DBPM) and a MVC-based MVD codec that uses the base view texture and depth information

Table 4.1: DBPM syntax vs. MVC inter prediction modes

MVC Inter Prediction Modes	DBPM
`mb_type`	`mb_type`
`mb_pred(mb_type)` {	*skipped*
`ref_idx` s	*reference is the base view*
`mvd` s }	*motion info not needed*
`coded_block_pattern` (`cbp`)	`coded_block_pattern`
if (`cbp`) {	if (`cbp`) {
`residual(0,15)` }	`residual(0,15)` }

for enhancing the coding efficiency of multiview videos as well as depth maps. The DBPM works similar to VSP DIRECT mode, but it requires only minor changes to MVC. Its simple syntax allows using a synthesized reference with a small signaling cost.

With DBPM, we utilize the multiview geometry for enhanced prediction of both the texture videos and their associated depth maps. In this chapter, first we provide a rate-distortion analysis of the proposed codec and compare it with MVC in the contexts of MVD data, depth map and multiview video coding. Second, we report the results of a subjective study comparing the perceptual quality of stereo videos coded with DBPM support and with MVC. Then we analyze the usage statistics of DBPM in the proposed codec and show that a typical Lagrangian rate-distortion optimization is effective to successfully choose between DBPM and other prediction modes available to the proposed encoder. Additionally, we provide a complexity analysis of the proposed codec, and finally discuss the effects of different encoder control parameters on the decoder side virtual view synthesis quality.

4.2 Depth-based Prediction Mode

MVC exploits interview correlations to increase the video coding efficiency through DCP mode, which uses estimated disparity vectors to relate macroblocks from one view to another. Estimating disparity vectors is similar to estimating a block level depth map; however it fails to capture the true multiview geometry in the scene. Attempts to exploit multiview geometry for compression show that VSP

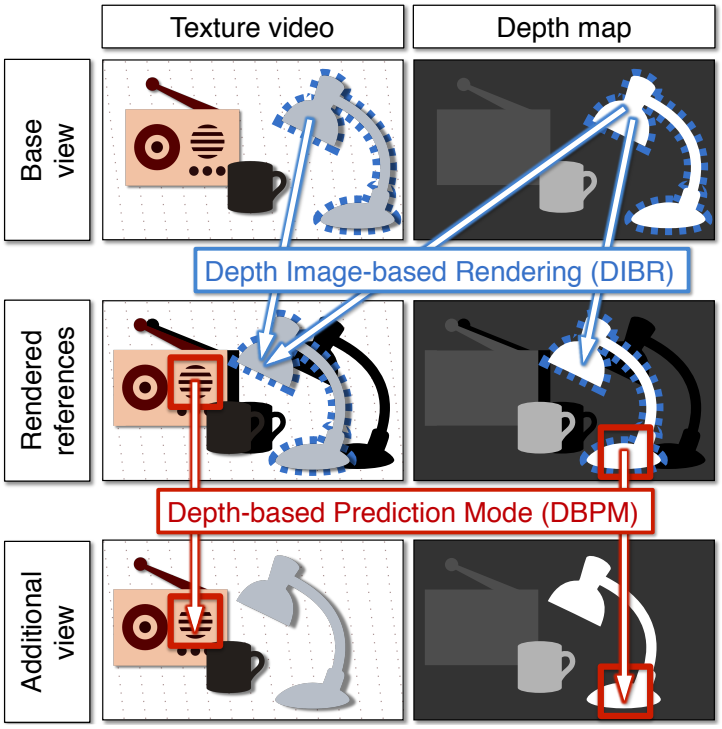


Figure 4.1: Illustration of the Depth-based Prediction Mode.

can further improve video coding efficiency by utilizing pixel level depth maps. As discussed in Section 4.1, some of these coding methods require picture level syntax changes to existing codecs, while others propose additional macroblock changes for efficient signaling of VSP modes.

In contrast the "depth-based prediction mode" (DBPM) allows using a synthesized reference picture for prediction without any high level syntax changes to the MVC, and requires only simple macroblock level syntax changes to the standard. DBPM can be used concurrently with existing prediction modes of MVC without introducing a significant overhead due to the changes in the syntax [4].

The novelty of DBPM is its simple syntax design, and its simplicity is based on two valid assumptions. First, a synthesized reference picture is most effective for a particular frame of a particular view and does not need to be stored for longer term use. Hence, with DBPM the synthesized reference frame is stored in a temporary buffer instead of the RPB. This allows avoiding picture level syntax changes to the codec. Second, a pixel level synthesis operation is expected to

align the pixels of the reference frame with the frame to be predicted. With this assumption, similar to VSP DIRECT/SKIP modes, DBPM does not signal any motion information to the decoder.

In addition to these assumptions, DBPM restricts how the synthesized reference is generated and limits the synthesis operation to use the base view information only. This restriction has certain benefits. Initially, combined with the first assumption, it allows skipping to signal the reference picture IDs (or reference views in this case) to the decoder, making the DBPM syntax even simpler. In addition, with a single reference view, the codec requires to repeat the *Depth Image-Based Rendering* (DIBR) [13] only once and can skip the blending operation for generating the synthesized reference. This reduces the computational cost of the DBPM. Furthermore, since hole filling is a costly and an error prone operation, it is also skipped for generating the DBPM references. This reduces the computational cost required by DBPM even further. Since only a single DIBR operation is conducted for generating the DBPM references, to differentiate with virtual view synthesis, they will be referred as *rendered* references in the rest of the chapter.

DBPM is essentially signaled only through a new `mb_type`. Once it is signaled to the decoder, the decoder simply refers to the rendered reference and copies the collocated macroblock for prediction. Since depth maps are typically estimated and contain local errors, the rendered references may suffer from local geometric distortions and fail to serve as strong reference for certain regions. If that is the case, similar to VSP DIRECT mode, DBPM supports signaling additional residual data. The comparison of the DBPM syntax with existing MVC inter prediction modes is provided in Table 4.1. Additionally, an illustration of DBPM is provided in red in Fig. 4.1 along with the DIBR operation, which is depicted in blue.

In this chapter we propose an MVC-based MVD codec with DBPM-support. In the proposed codec we utilize DBPM both for the prediction of the texture videos and the depth maps. The decoding process is very similar for both, with the only difference that depth maps are already self sufficient for generating the rendered reference. A block diagram of the proposed codec is provided in Fig. 4.2.

Due to errors in the depth map or simply because of the occlusion regions

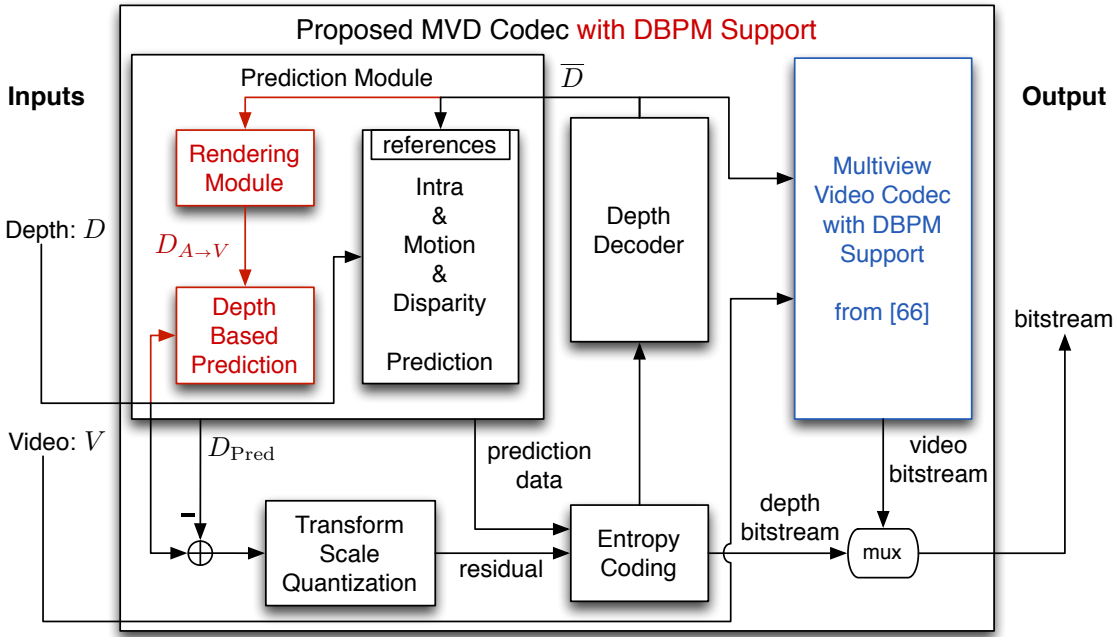


Figure 4.2: Block diagram of the proposed MVD codec with DBPM support.

in the rendered references, DBPM cannot always provide good prediction quality. To maintain video quality, a DBPM-enabled codec needs a decision mechanism to detect when this happens and rely on other prediction modes. In order to achieve that, the proposed encoder uses a typical Lagrangian rate-distortion optimization (RDO) mechanism to choose the best performing prediction mode between DBPM and existing MVC prediction modes. This decision mechanism proves to be effective and chooses DBPM only when the depth map is reliable. This will be further addressed in Section 4.3.3.

The proposed DBPM-enabled MVD codec is compatible with the 3D-AVC standardization track. It first encodes the base view video and its associated depth map using AVC. Then, while encoding additional views, along with DCP, DBPM gets enabled and the mode with the least Lagrangian cost is selected.

4.3 Experimental Results

In this section, we report results on the performance of the proposed codec. We use both objective metrics and subjective tests to assess its performance. First,

Table 4.2: Test conditions

Codec	Based on JMVC 8.5
Interview coding structure	2 views: P-I — 3views: P-I-P
Texture QPs	26, 31, 36, 41
Depth QPs	26, 31, 36, 41
Depth resolution	Tested for both 50% and 25% reduction in each direction
GOP structure	IPPP...
Anchor period	24 frames
Symbol mode	CABAC

we use Bjontegaard Delta Rate (BD-Rate) [67] and rate-distortion (RD) curves to measure the coding efficiency. Second, we provide results of a subjective study that evaluates the perceptual quality of stereo videos coded with DBPM support. To further understand its prediction quality, we report the percentage of the macroblocks the proposed encoder chooses to predict using DBPM. Then we compare the complexity of the proposed codec with MVC; and finally we analyze the effects of different encoder parameters on the decoder side virtual view synthesis quality.

We use the AVC-based experiments described in 3DV common test conditions (CTC) [44] for analysis. Our test conditions have minor differences with the CTC and are listed in Table 4.2. Instead of using fixed texture and depth quantization parameter (QP) combinations, we test the proposed codec with various combinations of texture and depth QPs. Additionally, CTC mandates only using depth maps with 50% reduced resolution in each direction. Instead we also test for 25% resolution reduction. For the sake of simplicity, in the rest of the chapter we will refer to 50% and 25% reduced resolution depth map inputs as "Depth-0.5" and "Depth-0.25", respectively.

4.3.1 Rate-Distortion Analysis

In this section, we compare the coding performance of the proposed codec with MVC using BD-Rate and RD curves. Note that BD-Rate calculates the average difference between two RD curves and its negative values signify percent bitrate savings against an anchor coding method.

We analyze the rate-distortion performance of DBPM in three different contexts. First, we provide results for the proposed MVD codec. The proposed codec encodes the texture videos and the depth maps both with DBPM support. In comparison we use MVC to encode texture and depth channels disjointly. Second, to isolate the contribution of the DBPM support for depth maps, we analyze its prediction performance in the context of depth map coding. Third, we report results for coding multiview videos as DBPM is effective also for coding texture videos without depth maps. Since DBPM requires a depth map, in this comparison the bitstream generated by our codec includes the base view depth map, whereas MVC bitstream does not.

MVD Coding

The BD-Rate results for coding MVD data using the proposed codec vs. MVC are reported in Tables 4.3 and 4.4. These results show that the proposed codec can achieve up to 9.2% bitrate savings with DBPM support and expectedly, the gains vary depending on the depth map quality. For example, *GTFly* and *UndoDancer* are among the sequences with largest gain since they are computer generated sequences with ground truth depth maps. In comparison, the depth maps of the *Newspaper* sequence are noisy, and consist of both temporal inconsistencies and spatial errors. This leads to inefficient coding of the depth maps and geometric distortions in the rendered references for DBPM. Thus, *Newspaper* is among the sequences that benefited the least from DBPM support. In addition, comparing the BD-Rates for different depth map resolutions, we see that both Depth-0.5 and Depth-0.25 yield comparable gains to each other. This is because the reduction in depth map resolution and consequently the reduction of depth bitrate compensates for the loss of effectiveness of DBPM. This shows that a depth bitrate constraint can be satisfied for different combinations of depth map resolution and QP, and the optimal choice depends on the desired quality of virtual view synthesis. This problem is addressed in Section 4.3.5.

Another important observation is that bitrate savings from DBPM add up as the number of anchor views increase from 2 to 3. This is expected due

Table 4.3: BD-Rate (%) for coding 2 view MVD data - measured against MVC.

	Depth-0.5				Depth-0.25			
Depth QP	26	31	36	41	26	31	36	41
Balloons	-1.58	-1.93	-2.06	-2.19	-1.90	-2.08	-2.11	-2.11
Kendo	-0.84	-1.09	-1.14	-1.13	-1.01	-1.12	-1.13	-1.03
Newspaper	-0.07	-0.12	-0.10	-0.11	-0.01	-0.12	-0.04	-0.12
UndoDancer	-2.90	-2.14	-1.42	-0.93	-2.62	-2.11	-1.48	-0.82
GTFly	-6.12	-6.06	-5.70	-4.99	-5.80	-5.67	-5.30	-4.80
PoznanHall2	-4.03	-4.21	-3.78	-3.59	-4.46	-4.52	-4.11	-3.75
PoznanStreet	-1.88	-1.97	-1.87	-1.56	-1.94	-1.97	-1.80	-1.58

Table 4.4: BD-Rate (%) for coding 3 view MVD data - measured against MVC.

	Depth-0.5				Depth-0.25			
Depth QP	26	31	36	41	26	31	36	41
Balloons	-2.57	-3.16	-3.33	-3.53	-3.20	-3.55	-3.51	-3.50
Kendo	-2.03	-2.53	-2.84	-2.88	-2.61	-2.90	-2.99	-2.80
Newspaper	-2.55	-3.17	-3.52	-3.64	-3.32	-3.80	-3.78	-3.67
UndoDancer	-7.06	-5.52	-3.93	-2.94	-6.57	-5.42	-3.97	-2.62
GTFly	-9.19	-9.03	-8.51	-7.50	-8.82	-8.56	-7.97	-7.16
PoznanHall2	-5.26	-5.35	-4.95	-4.49	-5.82	-5.78	-5.26	-4.68
PoznanStreet	-3.81	-4.04	-3.75	-3.09	-4.13	-4.05	-3.64	-3.08

to the P-I and P-I-P interview coding structures used for our experiments. Yet, it is not possible to generalize this for more anchor views. DBPM can provide good prediction as long as there is enough correlation between the base and the additional views, and this might not hold true depending on the sequence, position of the cameras of anchor views and the interview coding structure.

BD-Rate allows measuring bitrate savings in a concise manner, yet it fails to associate these savings with the absolute number of bits saved. Hence, in addition to the BD-Rate results, we also provide the RD curves for the *GTFly* sequence (3 views), which yields the most gain for the proposed codec. Fig. 4.3 shows that the proposed codec can deliver the same quality of MVD data with up to 900 kbps less bitrate than MVC.

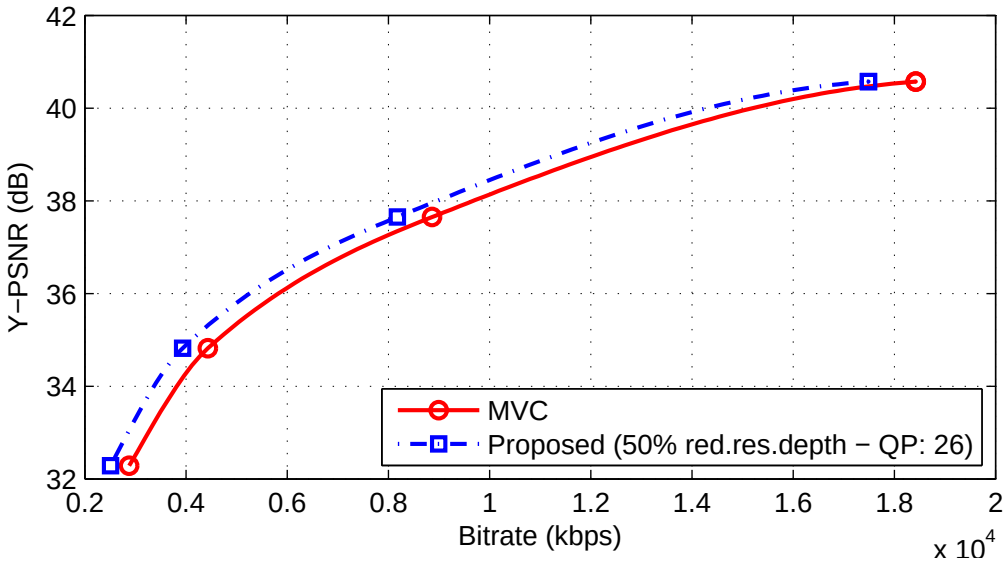


Figure 4.3: RD curves for *GTFly*, 3 views, coding MVD data.

Depth Map Coding

In the proposed codec DBPM is utilized not only for the prediction of the texture videos but also for the depth maps. To better understand the contribution of the DBPM support for the depth maps to the bitrate savings reported in Section 4.3.1, we also provide BD-Rate results for depth map coding in Table 4.5. Since depth maps consist of piece-wise smooth regions, DBPM faces a stronger competition against existing MVC prediction modes. Looking at the results in Table 4.5, DBPM proves to be successful with up to 9.9% bitrate saving when the depth maps are accurate. On the other hand, for depth maps with limited accuracy, the encoder chooses DBPM infrequently and the DBPM-enabled codec starts to yield slightly worse performance than MVC. These bitrate losses are limited to around or less than 1%, which are due to the syntax overhead introduced by DBPM.

Multiview Video Coding

DBPM is also effective for coding multiview video. The BD-Rate results analyzing the DBPM performance for multiview video coding are provided in Tables 4.6 and 4.7. These results show that even with the overhead spent on the base

Table 4.5: BD-Rate (%) for depth maps - measured against MVC.

	Depth-0.5		Depth-0.25	
	2 views	3 views	2 views	3 views
Balloons	0.32	0.51	0.41	0.74
Kendo	0.08	0.26	0.29	0.36
Newspaper	0.75	1.24	0.61	0.79
UndoDancer	**-2.98**	**-4.37**	**-2.46**	**-3.00**
GTFly	**-7.56**	**-9.93**	**-5.88**	**-8.00**
PoznanHall2	1.03	0.22	0.19	**-1.33**
PoznanStreet	**-1.69**	**-2.59**	**-0.99**	**-2.05**

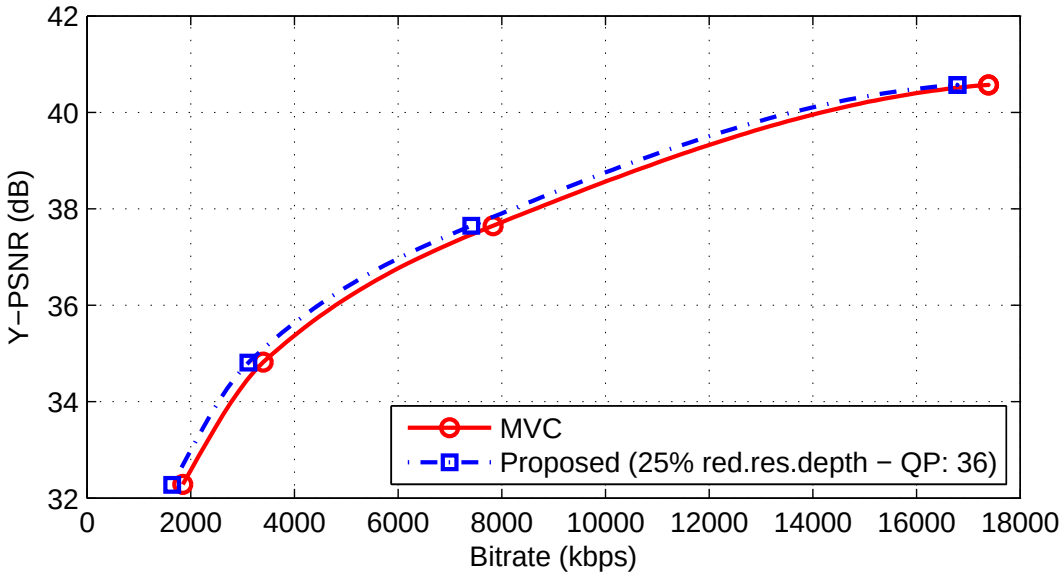


Figure 4.4: RD curves for *GTFly*, 3 views, coding multiview video (the proposed codec also encodes the base view depth map).

view depth map, the DBPM support can provide up to 6.7% bitrate savings over MVC, which corresponds to about 500 kbps. This bitrate savings can be seen in Fig. 4.4. On the other hand, there exist cases where the proposed codec cannot provide any savings and even yields significant bitrate loss in comparison to MVC. Nonetheless, this happens when an unnecessarily large bitrate is allocated to the depth map and gains achieved by DBPM cannot account for this depth overhead. However it is important to notice that for almost all sequences and all the tested number of anchor views, there is at least one combination of depth map QP and

Table 4.6: BD-Rate (%) for coding 2 view multiview video - measured against MVC.

	Depth-0.5				Depth-0.25			
Depth QP	26	31	36	41	26	31	36	41
Balloons	23.98	11.30	4.80	1.66	8.62	3.08	0.60	**-0.49**
Kendo	32.53	18.30	9.41	4.86	14.90	7.63	3.47	1.57
Newspaper	37.06	17.76	8.31	4.24	14.71	6.96	3.32	1.75
UndoDancer	2.31	1.12	0.65	0.36	**-0.19**	**-0.59**	**-0.55**	**-0.24**
GTFly	7.69	2.17	**-0.89**	**-2.24**	**-0.31**	**-2.49**	**-3.48**	**-3.73**
PoznanHall2	14.06	5.80	1.21	**-0.67**	3.14	**-0.31**	**-1.86**	**-2.33**
PoznanStreet	14.22	6.40	2.46	0.95	4.57	1.40	**-0.03**	**-0.49**

Table 4.7: BD-Rate (%) for coding 3 view multiview video - measured against MVC.

	Depth-0.5				Depth-0.25			
Depth QP	26	31	36	41	26	31	36	41
Balloons	13.69	5.12	0.91	**-1.14**	3.28	**-0.41**	**-1.87**	**-2.51**
Kendo	18.83	9.36	3.57	0.79	7.05	2.32	**-0.22**	**-1.21**
Newspaper	20.24	7.49	1.40	**-1.12**	5.41	0.32	**-1.86**	**-2.59**
UndoDancer	**-3.77**	**-3.47**	**-2.64**	**-2.14**	**-5.04**	**-4.46**	**-3.38**	**-2.24**
GTFly	0.22	**-3.36**	**-5.12**	**-5.58**	**-5.16**	**-6.41**	**-6.71**	**-6.40**
PoznanHall2	6.33	1.01	**-1.85**	**-2.66**	**-0.95**	**-3.04**	**-3.80**	**-3.75**
PoznanStreet	6.80	1.50	**-0.85**	**-1.39**	0.06	**-1.86**	**-2.46**	**-2.32**

resolution that yields gain over MVC. Although not addressed in this work, this shows that the encoder can consider this while allocating bitrate between texture videos and the depth map, and optimize for the depth map resolution and QP to spend bits on the depth map only when it is beneficial.

4.3.2 Subjective Tests

Although PSNR provides an easy way of measuring the quality of coded pictures, it is arguable that it is not always correlated with perceptual quality. For example, a pixel shift on a picture could yield low PSNR values although it would not be perceived as a degradation. On the contrary, a temporal flicker in a video sequence would be highly disturbing although PSNR values of each individual

Table 4.8: Test sequences that are used for the subjective tests.

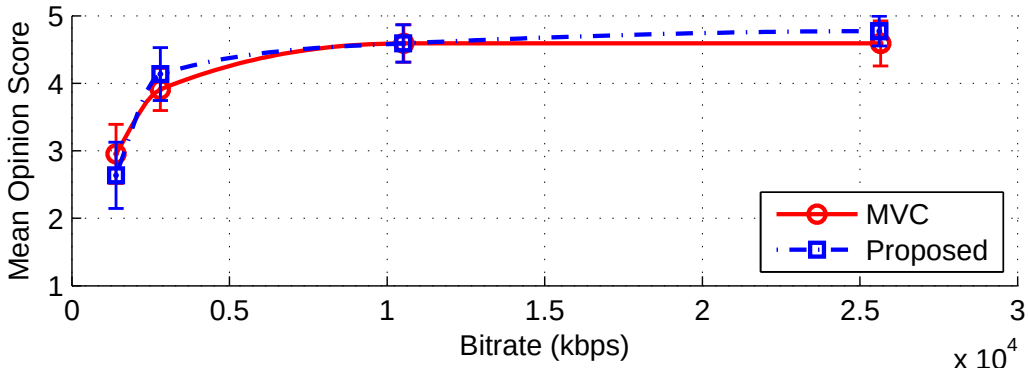
Sequence	Depth resolution	Depth QP
UndoDancer	Depth-0.25	31
PoznanHall2	Depth-0.25	41
PoznanStreet	Depth-0.25	41

frame could be high. Hence, we provide results of a subjective study that compare the performance of the proposed codec with MVC for stereo video coding.

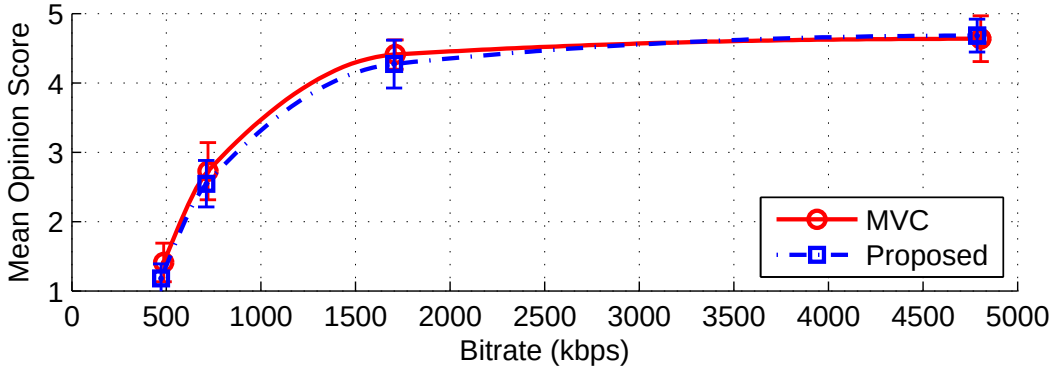
This study is conducted using the ITU-R recommended standard testing method, Double Stimulus Impairment Scale (DSIS) [68]. Stereo videos are displayed on a 47" LG 47LW6500 stereoscopic 3D TV and the experiments are conducted using "Tally", a subjective testing software introduced by Jain et al. [69]. 22 subjects participated in the experiments with at most 3 subjects in each session. 14 of them were male and 8 were female, with ages ranging from 21 to 53 with an average age of 29. Each participant is screened for visual acuity, color blindness and stereo acuity.

The study consisted of 24 test cases, (3 sequences, 4 bitrate levels and 2 tested codecs). The sequences chosen for the tests are listed in Table 4.8. For MVC only the texture videos are encoded, whereas for the proposed codec an additional depth map is also encoded, only to be used by DBPM. While using the proposed codec, the depth map QP and resolution for each sequence are chosen based on the gains reported in Table 4.6.

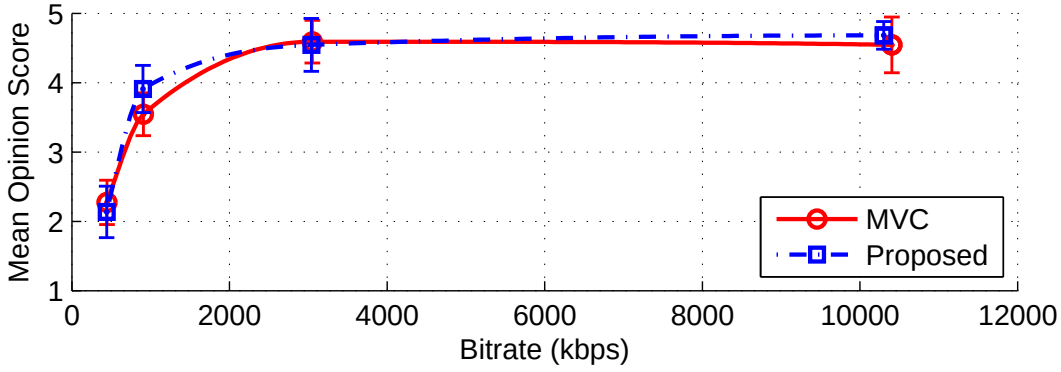
The RD curves based on the Mean Opinion Scores (MOS) of the subjects are depicted in Fig. 4.5. These plots show that for all sequences, the perceptual quality of the proposed codec is very close to MVC. Due to the confidence levels, it is not possible to derive strict conclusions on which codec performs better from the reported MOS scores. Yet, it is possible to confirm that the proposed codec does not generate any unexpected visual artifacts (e.g. temporal flicker) that PSNR measurements reported in Section 4.3.1 could have overlooked.



(a)



(b)



(c)

Figure 4.5: Subjective test results for stereo video coding, for sequences (a) *UndoDancer* (b) *PoznanHall2* (c) *PoznanStreet*.

4.3.3 DBPM Usage

The strength of DBPM is its low signaling cost, but it should be preferred over other MVC prediction modes when it can also provide good prediction quality.

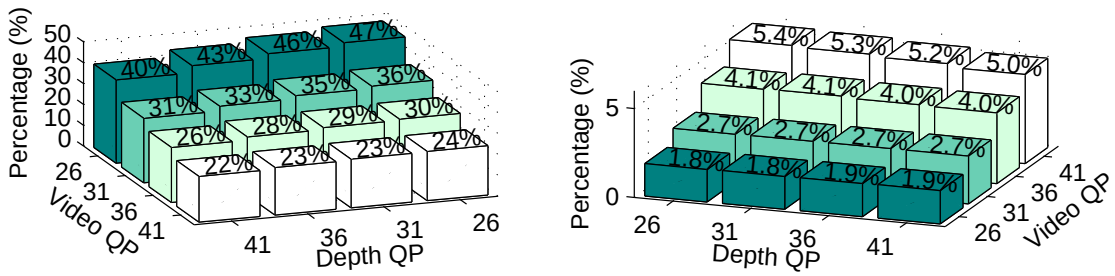


Figure 4.6: Percentage of the DBPM macroblocks for Depth-0.5, 3 views and sequences (a) *GTFly* (b) *Newspaper*.

In order to achieve this, the proposed codec uses the Lagrangian RDO module of MVC to decide between DBPM and other available modes. Here, we report the percentage of macroblocks that are predicted with DBPM and show that this framework can indeed adapt to different conditions affecting the prediction quality of DBPM. The DBPM usage for sequences *GTFly* and *Newspaper* are provided in Fig. 4.6. These sequences, as discussed in Section 4.3.1, reflect the two extreme cases for depth map quality and are the most informative.

For *GTFly*, results show that the proposed encoder chooses DBPM for up to 47% of the macroblocks. This shows that DBPM can be very effective for prediction when the depth map quality is high. Expectedly, DBPM usage decreases with increasing depth map QP but this drop in usage is minimal. On the other hand, increasing video QP has a more pronounced affect on DBPM usage, lowering it down to 22%. This is caused by the decreasing quality of the base view video with increasing video QP, which consequently reduces the quality of the rendered reference for DBPM.

In comparison, when the depth maps have low quality as in the case of *Newspaper*, encoder infrequently prefers DBPM, with DBPM usage ranging only from 2% to 5%. The trend for DBPM usage with varying depth QP for *Newspaper* is similar to the trend for *GTFly*; but the effect of video QP on DBPM usage is the opposite. For *Newspaper* DBPM usage increases with video QP, and it is because DBPM starts to operate as a low cost alternative to existing MVC prediction modes for lower bitrates.

Table 4.9: Test platform.

CPU	Intel i7-980X 6-core 3.33GHz
Memory	12GB DDR3-1066
Operating system	Ubuntu Linux 10.04 LTS

4.3.4 Complexity Analysis

In this section we compare the complexity of the proposed codec with MVC. We estimate the complexity based on the run-time of the encoding and decoding processes. The experiments are conducted on a desktop computer with specifications listed in Table 4.9. The average percent increase in complexity for different test sequences and depth resolutions are listed in Table 4.10. According to the results, the encoder complexity is only slightly more than MVC, with around 4% increase. In comparison, the increase in decoder complexity is more prominent and is about twice the complexity of MVC; yet we believe that this is still within acceptable boundaries and real-time implementations could be possible.

We designed DBPM such that the rendering module in the codec is simple and introduces only a limited complexity increase over MVC. In the proposed codec, we limited the rendering operation to DIBR, which allowed us to skip other costly view synthesis operations that would contribute to the complexity increase much more than the DIBR [2]. However, the majority of the complexity increase in the proposed codec is still due to the rendering operation. For a fair comparison with the MVC software, in our experiments, we did not use any hardware acceleration for the DIBR operation; yet a GPU implementation of the DIBR can render an entire picture in the order of milliseconds. Hence, a hardware acceleration could considerably lower the run-time of the proposed codec [2]. In the future we foresee that codecs will employ hardware implementations of the DIBR operation as real-time decoder side virtual view synthesis is desired for the next generation 3DV codecs.

Table 4.10: Average computational complexity of the proposed codec in comparison to MVC.

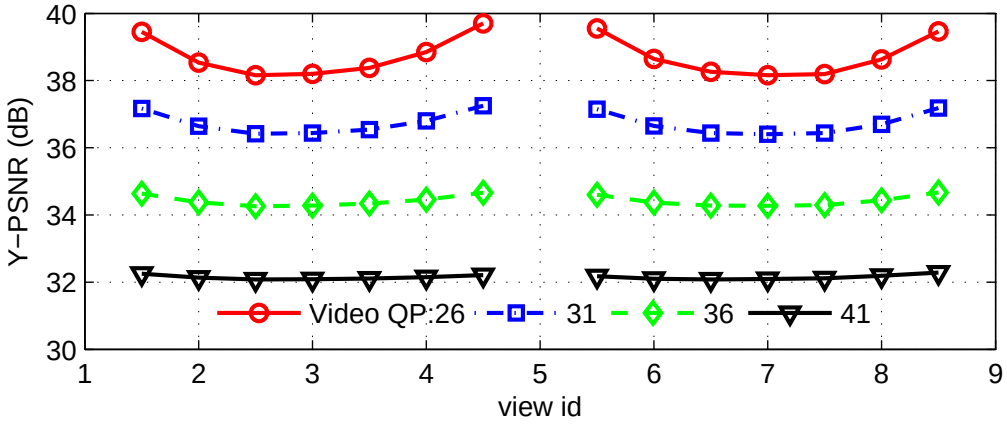
	Depth-0.5		Depth-0.25	
	Encoder	Decoder	Encoder	Decoder
Balloons	105	189	101	190
Kendo	105	175	104	183
Newspaper	97	222	95	216
UndoDancer	104	200	104	187
GTFly	106	211	106	210
PoznanHall2	101	276	101	256
PoznanStreet	110	232	110	223
Average	104%	215%	103%	209%

4.3.5 Virtual View Synthesis

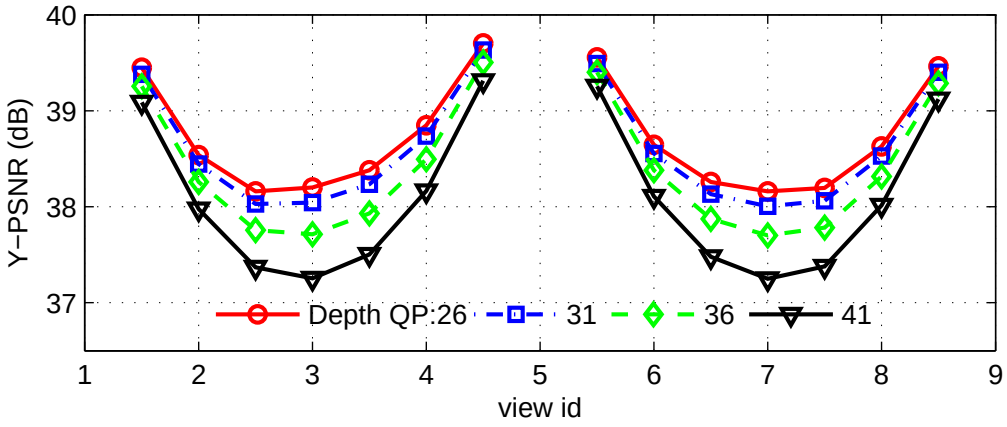
As discussed earlier, one of the biggest advantages of having a depth-based 3D video representation is the support for the virtual view synthesis. So far we investigated the effects of encoder parameters on the quality of the coded anchor views. On the other hand, encoder control parameters also affect the quality of the virtual view synthesis and it is important to understand the effect of each parameter.

Virtual view synthesis, as also mandated in the 3DV common test conditions [44], typically refers to synthesizing virtual views in between pairs of neighboring anchor views, or so called view interpolation. In order to analyze the performance of the proposed codec in this context, 7 virtual views are interpolated in between each neighboring pair of anchor views. To serve as a reference, the synthesis operation is conducted first using uncompressed texture videos and their associated full resolution depth maps. Then the same operations are repeated for the compressed data to explore effects of different video QPs, depth QPs and resolutions on the quality of interpolated views. For these experiments we used the software provided in the 3DV CTC [44].

The change in view synthesis quality for *GTFly* is depicted in Figure 4.7 for varying video and depth QPs. Results for other sequences follow similar trends and therefore are omitted here. The results show that the view synthesis quality



(a)

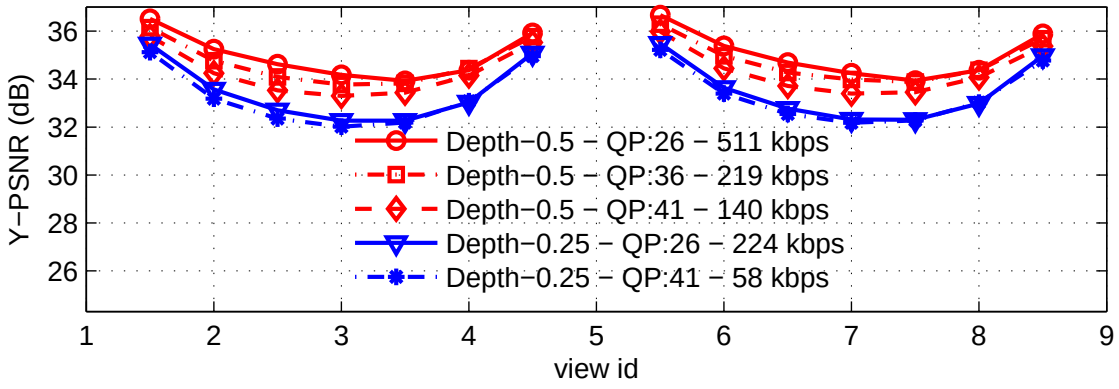


(b)

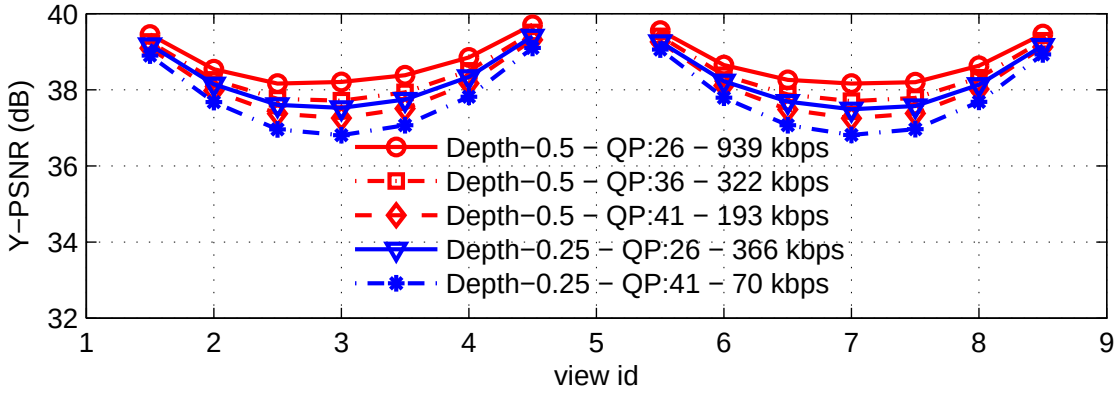
Figure 4.7: View interpolation quality for *GTFly*, Depth-0.5, (a) fixed depth QP (26) and varying video QP (b) fixed video QP (26) and varying depth QP.

decreases both with increasing video QP and depth QP. This is not surprising as increasing video QP degrades the quality of the reference views, and increasing depth QP introduces geometrical error to the synthesis. Yet, one interesting observation is that the quality is more robust against high QPs of depth maps than texture videos. For high quality depth maps (low QP), the PSNR values for the interpolated views level around the PSNR values of the anchor views, and they decrease slightly more as depth QP increases.

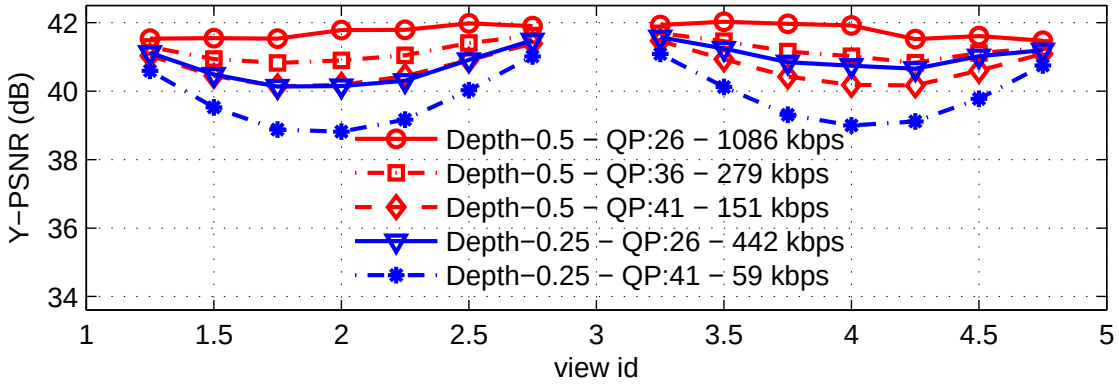
Depth map resolution provides an additional control over the bitrate spent on the depth, and has a more pronounced effect on the view synthesis quality than depth map QP. The view synthesis quality for various combinations of depth map



(a)



(b)



(c)

Figure 4.8: View interpolation quality for fixed video QP (26), varying depth QP and resolution for sequences (a) *UndoDancer* (b) *GTFly* (c) *Balloons*.

resolution and QP are plotted in Figure 4.8. Since similar depth map bitrates can be achieved with different combinations of depth map resolution and QP, the

bitrates associated with each plot are also provided (total for 3 views).

Looking at the results in Tables 4.6 and 4.7, for coding multiview video, we see that low resolution depth maps typically yield better coding performance for the proposed codec. However, as shown in Tables 4.3 and 4.4, the coding gains achieved by different combinations of depth map QP and resolution vary only slightly while coding an MVD sequence. Hence, for coding MVD data, the choice of depth resolution and QP should be mostly based on the virtual view synthesis quality. For the tested depth map resolutions, the higher resolution but higher QP depth maps achieve better synthesis quality than their lower resolution alternative at a similar bitrate. This holds true for all the sequences in Figure 4.8 and is most noticeable for the *UndoDancer* sequence.

All in all, the optimal choice for the depth map resolution is application dependent. If depth information is only to be used for DBPM but not for virtual view synthesis, a lower resolution depth map can be a better candidate. Contrarily, when view interpolation is desired, QP seems to be a better option for controlling the depth map bitrate instead of the depth map resolution.

4.4 Conclusion

In this chapter, we proposed a multiview video plus depth (MVD) codec that utilizes the depth-based macroblock prediction mode (DBPM) to increase coding efficiency. The proposed codec uses DBPM not only for coding texture videos but also for the depth maps.

Addition of DBPM to the codec requires only minor macroblock level changes to the MVC syntax, and this design allows DBPM to be signaled very efficiently. DBPM provides a reliable method of prediction when depth information is accurate, and it gets chosen by the encoder for up to 47% of the macroblocks. Contrarily, when depth information is unreliable, encoder mostly relies on other prediction modes, yet for some macroblocks DBPM still serves as a low cost option.

In comparison to MVC, for the AVC-based experiments described in 3DV common test conditions, the proposed codec can provide up to 9.2%, 9.9% and

6.7% bitrate savings for coding MVD data, depth map and multiview video coding respectively.

Moreover, we reported results for a subjective study comparing the perceptual quality of stereo videos coded with DBPM support and with MVC. Based on our findings in this study, we showed that the proposed codec does not introduce any unexpected visual artifacts and attain similar perceptual quality as MVC.

We also provided a complexity analysis of the proposed codec and compared it with the complexity of MVC. According to the results, the encoder complexity of the proposed codec is comparable to MVC, and the decoder complexity is about two times of the complexity of MVC. The complexity increase is due to the rendering module in the codec and we argue that the rendering module can be implemented in hardware, which would decrease decoder complexity significantly.

Finally, we analyzed the effects of different encoder parameters of the proposed codec on virtual view synthesis quality. For these experiments, we synthesized intermediate views between neighboring pairs of coded anchor views and measured the quality degradation with respect to varying texture video QP, depth map QP and resolution. The results show that video QP has the most influence. It controls the quality of the anchor views, which essentially determines the upper bound for the synthesis quality. Depth map QP and resolution also contribute towards the synthesis quality and they both provide means for controlling the bitrate allocated to the depth maps. For the tested conditions, we showed that there is a trade-off between the depth map resolution and QP, with the optimal combination depending on the application. According to our findings, typically low resolution depth maps yield better coding gains for anchor views of multiview video; however for similar bitrates, higher resolution but higher QP depth maps generally yield better view synthesis performance.

Acknowledgments

This chapter, in full, is a reprint of the material as it appears in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen,

Truong Q., IEEE, 2014 [3]. It is an extension of the material appearing in the Proceedings of the IEEE International Conference on Image Processing 2013. Bal, Can; Nguyen, Truong Q., IEEE, 2013 [4]. The dissertation author was the primary investigator and author of both of these papers.

Chapter 5

A Novel 3D Video Codec Based on 3D-AVC and Image Domain Warping

5.1 Introduction

3D-AVC is the upcoming AVC-based 3DV coding standard for MVD data. It aims to efficiently compress the depth maps while utilizing them to increase the coding efficiency of the texture videos. Currently, some of the new coding tools adopted in 3D-AVC are the *In-loop Block-based View Synthesis Prediction* (VSP), *Depth-based Motion Vector Prediction* (DMVP), *Adaptive Luminance Compensation* (ALC), *Nonlinear Depth Representation* (NDR) and *View Synthesis Distortion* (VSD). Details of these tools can be found in the current 3D-AVC Test Model Description document [70].

Additionally, 3D-AVC and its syntax are designed to support reduced resolution depth maps, which have been shown to provide significant coding gains and can be rescaled with minimal error for view synthesis [43, 71]. This feature allows the resolution of the depth maps to be independent of the resolution of their associated videos. However, researchers generally adopt dyadic subsampling for the depth maps to simplify the upsampling process.

The block diagram of the coding process for 3D-AVC is shown in Fig. 5.1(a). The dashed blocks in the diagram are only present for the reduced resolution depth map coding. The *Post Dilation Filter* is proposed to JCT-3V by Seok et al. [72] and has been shown to be very effective in treating the synthesis artifacts introduced by depth map rescaling. It is adopted into the reference codec software (3DV-ATM) and mandated to be enabled in the Common Test Conditions (CTC) [73].

In this work, instead of using the typical MVD representation, we propose a novel warp-based 3DV codec that uses the MVW representation. We realize the proposed codec using 3D-AVC without any changes to its syntax. To the best of our knowledge, this codec is the first in literature that compresses MVW data using 3D-AVC. Using the proposed codec, we also establish a framework that allows us to compare warp and depth-based 3DV coding in terms of performance and complexity.

5.2 Warp Map Coding using 3D-AVC

For warp map coding, 3D-AVC can be used without any changes to the syntax. The support for arbitrary depth map resolution already exists and this allows warps to be incorporated within the bitstream by simply replacing depth map inputs with warp maps. Especially if s and o in Eq. (2.14) are selected carefully, specifically according to Eq. (2.2), all of the new coding tools present in 3D-AVC continue to function effectively even with warp map inputs.

In our experiments, we use the latest 3D-AVC reference codec software 3DV-ATM (v13.1) [74]. 3DV-ATM is not programmed to fully support arbitrary depth resolutions as some of functions in the software work only for dyadic sub-sampling of the depth maps. We slightly modified the codec to add this support but we did not make any modifications to the 3D-AVC syntax. With that, we realized the proposed warp-based codec using 3D-AVC as depicted in Fig. 5.1(b). The two main differences of the proposed codec with its depth-based counterpart is that it does not require the warp maps to be upsampled at the decoder, and DIBR operations in the view synthesis are replaced by IDW.

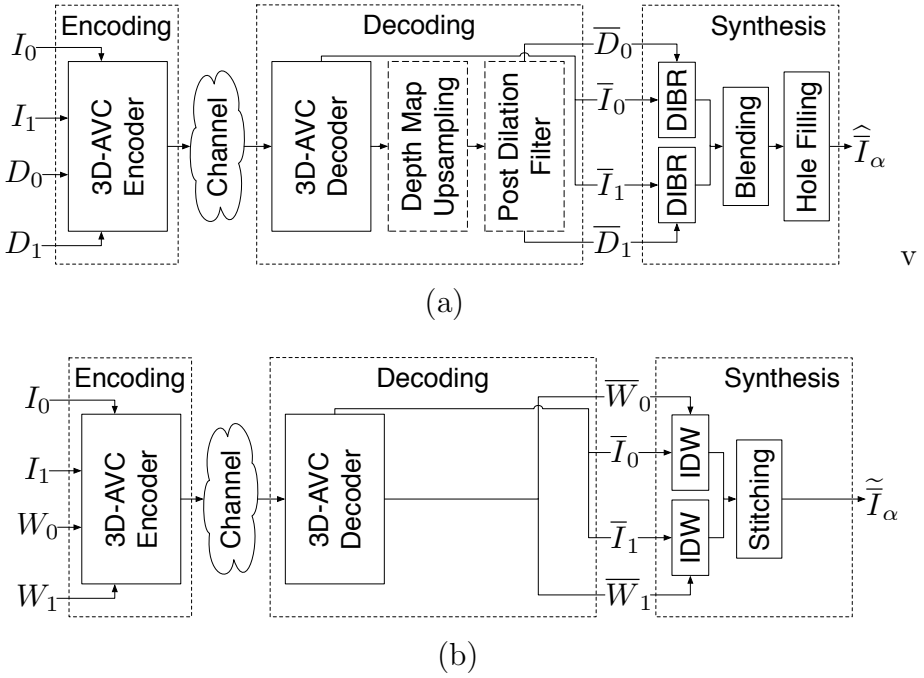


Figure 5.1: Block diagram of the 3D-AVC codec for (a) depth map (b) warp map inputs.

As mandated by CTC, in our experiments we used the rendering software from 3D-HTM (v9.3) [73] for view synthesis. This rendering software implements all of the necessary depth-based view synthesis components such as DIBR, blending, and hole filling. It is written in C++ and does not use any GPU hardware acceleration. We implemented IDW using bilinear upsampling as discussed in Section 2.7, and incorporated it into the 3D-HTM rendering software. Our additions to the software are also written in C++ and do not use any special optimized libraries or GPU hardware acceleration either.

5.3 Experimental Results

In this section, we analyze the coding efficiency and computational complexity of the proposed codec and compare it with the depth-based 3D-AVC. In our experiments, we follow the 3D-AVC test conditions mandated by CTC [73] unless otherwise stated. We only consider the stereo input case as the warp data provided by Stefanoski and Smolic [12] contains only 2 anchor views.

Due to the resemblance of warp maps to reduced resolution depth maps, we compare the proposed codec with 3D-AVC using depth maps with full resolution, and 50% and 25% reduced resolution in each direction. For the sake of simplicity, in the rest of the paper we will refer to depth-based coding methods with full resolution, and 50% and 25% reduced resolution depth map inputs as "Depth-1.0", "Depth-0.5" and "Depth-0.25", respectively. In comparison, we will refer to the proposed warp-based codec as "Warp". As there are no preset depth/warp QP values for Depth-0.25 and Warp in CTC, for these methods we simply use the same set of QPs mandated for Depth-0.5.

5.3.1 Rate-Distortion Performance

To measure the rate-distortion performance, we rely on *Bjontegaard Delta Rate* (BD-Rate) [67] and rate-distortion (RD) curves. Note that BD-Rate is measured against an anchor coding method, and its negative values signify average bitrate savings in percentage achieved by another coding method for the same objective quality video.

In our analysis we use Depth-1.0 as the anchor and the associated BD-Rate results are provided in Table 5.1 and 5.2. According to these results the proposed codec (Warp) can provide 12.8% and 1.5% bitrate savings on average for the anchor and synthesized views, respectively. This shows that warp maps can be compressed very efficiently using 3D-AVC. However, Warp is not the best performing method among the competition, especially for the synthesized views. In comparison to Warp, Depth-0.25 provides a slightly better average bitrate savings for anchor views with a difference of only 0.2%, while Depth-0.5 provides better synthesis quality at the expense of a significant bitrate loss in anchor views.

An important observation here is that Warp is performing poorly for the *Poznan* sequences, and these sequences are the only ones that adversely affect its average BD-Rate for the synthesized views. The reason behind this is discussed in Section 5.3.2.

Table 5.1: BD-Rate (%) for anchor views for CTC QPs - measured against Depth-1.0.

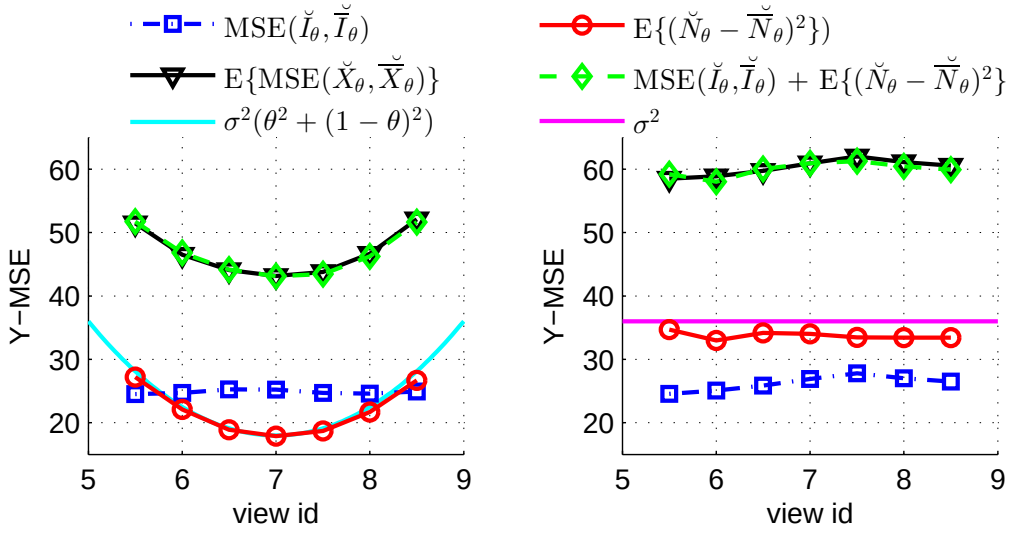
	Depth-0.5	Depth-0.25	Warp (no VSD)	**Warp**
Balloons	-6.78	-12.74	-10.75	-12.52
Kendo	-7.79	-16.48	-14.14	-16.73
Newspaper	-6.02	-10.41	-10.16	-11.40
UndoDancer	-9.90	-13.22	-12.37	-12.49
GTFly	-10.11	-14.13	-12.97	-13.36
PoznanHall2	-8.34	-13.48	-12.73	-13.24
PoznanStreet	-6.79	-10.81	-9.59	-9.91
Average	-7.96	**-13.04**	-11.81	-12.81
Average w/o Poznan seq.	-8.12	**-13.40**	-12.08	-13.30

Table 5.2: BD-Rate (%) for synthesized views for CTC QPs - measured against Depth-1.0.

	Depth-0.5	Depth-0.25	Warp (no VSD)	**Warp**
Balloons	-2.67	-4.83	-3.41	-4.40
Kendo	-4.53	-10.44	-4.50	-5.81
Newspaper	0.60	5.37	-13.66	-14.46
UndoDancer	0.40	8.25	-9.83	-9.60
GTFly	-9.76	-10.49	-10.87	-9.72
PoznanHall2	-5.27	-5.87	15.37	14.66
PoznanStreet	-5.82	-7.63	18.10	18.56
Average	**-3.87**	-3.66	-1.26	-1.54
Average w/o Poznan seq.	-3.19	-2.43	-8.46	**-8.80**

5.3.2 Effect of Camera Noise on the PSNR Calculation of the Synthesized Views

Due to the camera sensor, lighting conditions, etc., camera noise might be present in the captured videos. The noise level in the anchor views affects the PSNR results measured on the synthesized views. To model such a camera noise N on the anchor views, we use zero-mean ($\mu = 0$) additive white Gaussian noise (AWGN) with variance σ^2.



(a) (b)

Figure 5.2: MSE of the (a) DIBR (Depth-1.0) (b) IDW (Warp) generated synthesized views for the noise added *GTFly* sequence compressed with texture QP 36.

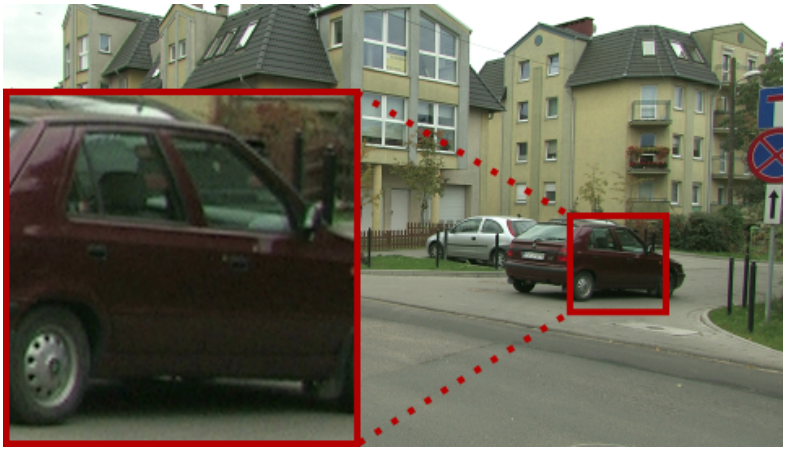
Let $\breve{X}_\theta$ be the noisy observation of a synthesized image $\breve{I}_\theta$. From Eq. (2.15), $\breve{X}_\theta$ can be written as:

$$\breve{X}_\theta = \breve{I}_\theta + \breve{N}_\theta \tag{5.1}$$

$$= \breve{I}_\theta + (1-\gamma)\cdot\breve{N}_{0\to\theta} + \gamma\cdot\breve{N}_{1\to\theta} \tag{5.2}$$

Since the rendering operation for both DIBR and IDW is based on pixel shifting and the pixels of N are i.i.d., we can assume that $\breve{N}_{0\to\theta}(\mathbf{x})$ and $\breve{N}_{1\to\theta}(\mathbf{x})$ are equivalent to the noise on the left (N_0) and right (N_1) reference images. With that, $\breve{N}_\theta$ becomes a zero-mean Gaussian with variance $\breve{\sigma}^2 = (\gamma^2+(1-\gamma)^2)\cdot\sigma^2 \leq \sigma^2$.

3D-AVC is a block-based lossy coding algorithm and relies on quantization matrices for compression. Quantization matrices operate on the DCT coefficients such that high-frequency coefficients are quantized more heavily than low-frequency coefficients. This quantization step can be modeled as a low-pass filtering operation, and the compressed noise $\overline{N}$ can be modeled as low-pass filtered colored Gaussian noise. Let $H(e^{j\omega})$ be the frequency response of this low-pass filter. Then $(N-\overline{N})$ can also be modeled as colored Gaussian noise generated by



(a)

(b)

Figure 5.3: Demonstration of camera noise on frame #140 of the (a) *PoznanStreet* (b) *PoznanHall2* sequence.

the high-pass filter $G(e^{j\omega}) = 1 - H(e^{j\omega})$. Since $\mu = 0$ and quantization matrices do not introduce any offset to the DC coefficient, then $\mathrm{E}\{N - \overline{N}\} = 0$. Additionally, the variance of the high-pass filtered noise $\mathrm{E}\{(N - \overline{N})^2\}$ can be calculated as [75, eq. (C.67), Appendix C]:

$$\mathrm{E}\{(N - \overline{N})^2\} = \acute{\sigma}^2 = \sigma^2 \cdot \frac{1}{2\pi} \int_{\pi}^{\pi} |G(e^{j\omega})|^2 d\omega \tag{5.3}$$

In order not to introduce any bias, quantization matrices are designed not to amplify the input. Therefore we can also assume that $\acute{\sigma}^2 \leq \sigma^2$.

Following a similar analysis to the mean and variance calculation of $\breve{N}_\theta$,

$(\breve{N}_\theta - \breve{\overline{N}}_\theta)$ can be shown to be a zero-mean Gaussian noise with variance:

$$\mathrm{E}\{(\breve{N}_\theta - \breve{\overline{N}}_\theta)^2\} = \breve{\sigma}^2 = (\gamma^2 + (1-\gamma)^2) \cdot \acute{\sigma}^2 \tag{5.4}$$

In Section 5.3, we calculate PSNR or equivalently the mean squared error (MSE) on a $M \times N$ compressed noisy synthesized image, $\breve{\overline{X}}_\theta$ as:

$$\mathrm{MSE}(\breve{X}_\theta, \breve{\overline{X}}_\theta) = \frac{1}{MN} \sum (\breve{X}_\theta - \breve{\overline{X}}_\theta)^2 \tag{5.5}$$

Using Eq. (5.1), we can rewrite Eq. (5.5) as:

$$\mathrm{MSE}(\breve{X}_\theta, \breve{\overline{X}}_\theta) = \frac{1}{MN} \sum ((\breve{I}_\theta - \breve{\overline{I}}_\theta) + (\breve{N}_\theta - \breve{\overline{N}}_\theta))^2$$

and since the noise is uncorrelated with the synthesized images, its expected value becomes:

$$\begin{aligned} \mathrm{E}\{\mathrm{MSE}(\breve{X}_\theta, \breve{\overline{X}}_\theta)\} = \mathrm{MSE}(\breve{I}_\theta, \breve{\overline{I}}_\theta) + \mathrm{E}\{(\breve{N}_\theta - \breve{\overline{N}}_\theta)^2\} + \\ \frac{2}{MN} \sum (\breve{I}_\theta - \breve{\overline{I}}_\theta) \cdot \mathrm{E}\{\breve{N}_\theta - \breve{\overline{N}}_\theta\} \end{aligned} \tag{5.6}$$

Using $\mathrm{E}\{\breve{N}_\theta - \breve{\overline{N}}_\theta\} = 0$ and Eq. (5.4), Eq. (5.6) is simplified to:

$$\begin{aligned} \mathrm{E}\{\mathrm{MSE}(\breve{X}_\theta, \breve{\overline{X}}_\theta)\} &= \mathrm{MSE}(\breve{I}_\theta, \breve{\overline{I}}_\theta) + \mathrm{E}\{(\breve{N}_\theta - \breve{\overline{N}}_\theta)^2\} \\ &= \mathrm{MSE}(\breve{I}_\theta, \breve{\overline{I}}_\theta) + (\gamma^2 + (1-\gamma)^2) \cdot \acute{\sigma}^2 \end{aligned} \tag{5.7}$$

For DIBR, the blending operator γ is given in Eq. (2.16). We argue that most of the pixels in rendered images $\widehat{I}_{0\to\theta}$ and $\widehat{I}_{1\to\theta}$ are non-hole pixels. Moreover, the overlapping non-hole pixels of $\widehat{I}_{0\to\theta}$ and $\widehat{I}_{1\to\theta}$ typically have similar corresponding depth values such that $|\widehat{D}_{0\to\theta}(\mathbf{x}) - \widehat{D}_{1\to\theta}(\mathbf{x})| < \tau$. This is clearly not valid for occlusion regions, yet the percentage of the occluded pixels are generally low enough that we can assume otherwise for this analysis. In comparison, from Eq. (2.17), γ is always either 0 or 1 for IDW. With these observations we can update Eq. (5.7) as:

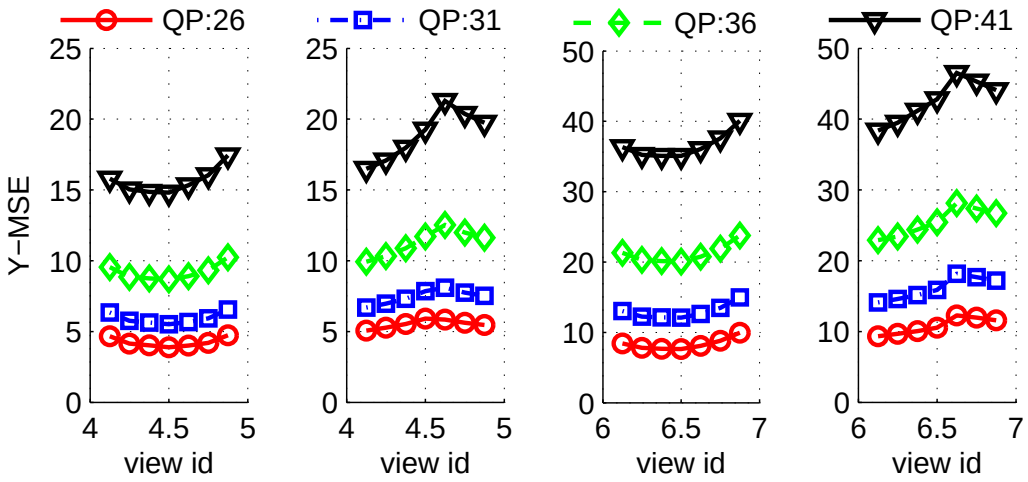
$$\mathrm{E}\{\mathrm{MSE}(\breve{X}_\theta, \breve{\overline{X}}_\theta)\} = \begin{cases} \mathrm{MSE}(\breve{I}_\theta, \breve{\overline{I}}_\theta) + (\theta^2 + (1-\theta)^2) \cdot \acute{\sigma}^2 & \text{if DIBR.} \\ \mathrm{MSE}(\breve{I}_\theta, \breve{\overline{I}}_\theta) + \acute{\sigma}^2 & \text{if IDW.} \end{cases} \tag{5.8}$$

In order to verify our analysis, we conducted experiments on the *GTFly* sequence. *GTFly* is a computer generated sequence and is known to contain no camera noise. We generated independent N with $\sigma = 6$ for both of the anchor views. We chose this σ to visually match the noise levels in the *Poznan* sequences. We also created a new sequence from the same realization of the noise and compressed the noise and the noisy *GTFly* sequence using a fixed texture QP of 36. The associated depth/warp QPs are selected according to the CTC. We repeated the experiment for 10 realizations of the noise and took the average to approximate the $\mathrm{E}\{\cdot\}$. The results are plotted in Fig. 5.2 and are consistent with our analysis.

Both from Fig. 5.2 and Eq. (5.8), due to the blending operation, it can be seen that DIBR yields a smaller MSE than IDW for the synthesized views in the presence of camera noise. Please note that this does not necessarily correlate with the true visual quality of the synthesized views.

Clearly, camera noise biases BD-Rate calculations in favor of DIBR relative to IDW for the synthesized views. Among our test sequences, we found that the *Poznan* sequences possess enough camera noise that this effect becomes significant. The camera noise present in these sequences is depicted in Fig. 5.3. To confirm this bias, we also provide the MSE of the synthesized views for these sequences in Fig. 5.4. The trends of MSE in Fig. 5.4 are consistent with the plots provided in Fig. 5.2, which supports our claim.

Since *Poznan* sequences skew the results toward the advantage of DIBR, we recalculated the BD-Rates without these sequences and listed these results in Table 5.1 and 5.2 as well. With the new averages, Warp becomes the top performer for synthesized views with average bitrate savings of 8.8%. For anchor views, it still yields slightly less bitrate savings than Depth-0.25 but the difference drops to only 0.1%. Similar results can also be seen from the RD curves corresponding to the noise-free *GTFly* sequence in Fig. 5.5.



(a) DIBR (b) IDW (c) DIBR (d) IDW

Figure 5.4: MSE of the DIBR (Depth-1.0) and IDW (Warp) generated synthesized views for the (a) (b) *PoznanHall2* (c) (d) *PoznanStreet* sequence.

5.3.3 Performance of the Depth-based Coding Tools of 3D-AVC for Warp Maps

As mentioned in Section 5.2, 3D-AVC uses depth maps to increase the coding efficiency of the texture videos. Since the proposed codec does not introduce any syntax changes to 3D-AVC, the depth-based coding tools treat the input warp maps as depth maps during the encoding process. Since warps are not calculated to provide accurate depth information, they cannot provide the same efficiency as the depth maps for texture video coding.

The results reported in Table 5.1 and 5.2 are calculated using the total bitrate of the texture videos and depth/warp maps. However, in order to understand the loss in coding efficiency for the texture videos, we also calculated the BD-Rate results for the anchor views using texture bitrates only. As it can be seen from these results, which are provided in Table 5.3, the depth-based coding tools of 3D-AVC perform slightly worse for warp map inputs, accounting for about 1.7% bitrate loss. However, this loss is more than recouped by the efficiency of warp map coding.

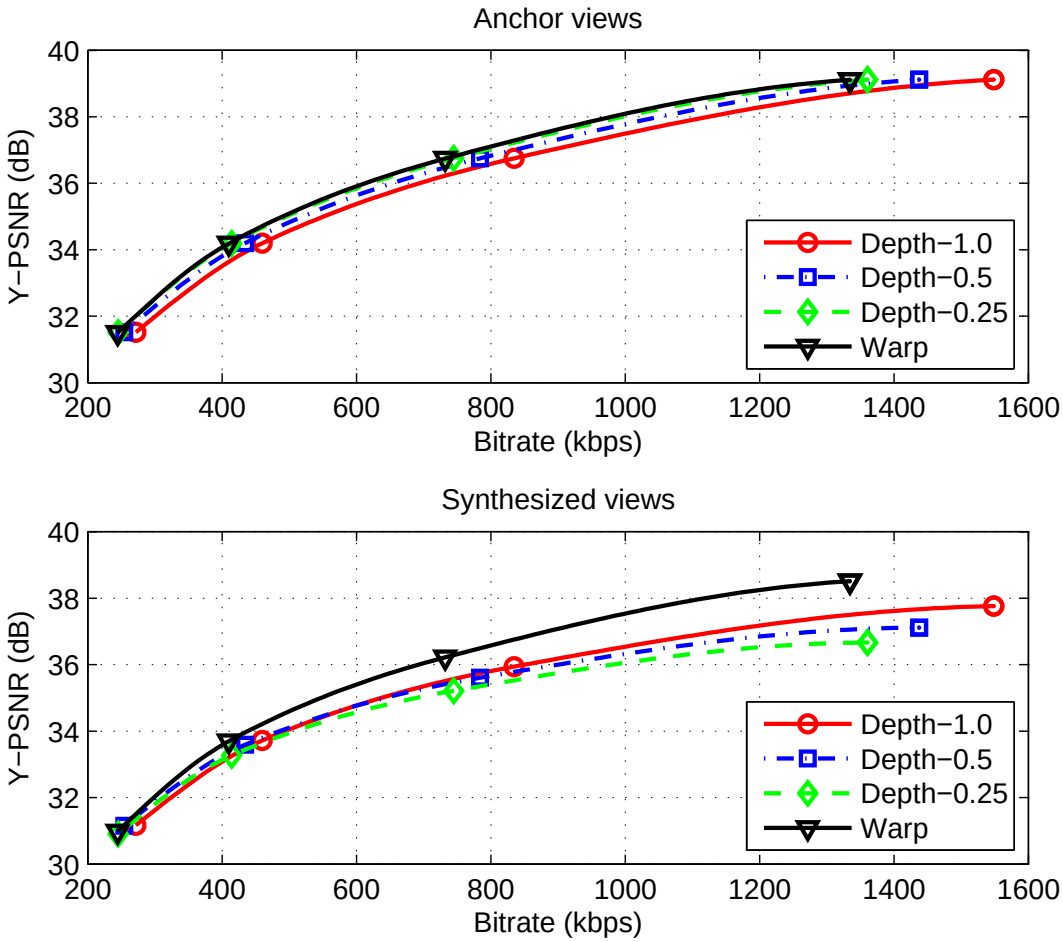


Figure 5.5: RD curves for the *Newspaper* sequence under CTC.

Table 5.3: BD-Rate (%) for anchor views for CTC QPs - measured against Depth-1.0 using texture bitrates only.

	Depth-0.5	Depth-0.25	**Warp**
Balloons	-0.09	0.07	0.47
Kendo	-0.03	0.21	0.94
Newspaper	0.08	0.06	0.48
UndoDancer	-0.10	0.19	2.63
GTFly	-0.18	0.40	2.80
PoznanHall2	-0.12	-0.17	1.84
PoznanStreet	-0.18	0.27	2.45
Average	-0.09	0.15	1.66

5.3.4 Performance of the View Synthesis Distortion in 3D-AVC for Warp Maps

In 3D-AVC, depth maps are encoded with block-based coding tools, and the mode decision, block size selection, etc. are decided by standard Lagrangian RD optimization functions. Since the depth-maps are not observed but only used for view synthesis, using typical distortion metrics such as *Sum of Squared Errors* (SSE) or *Mean Squared Error* (MSE) do not necessarily yield the best synthesis results. To address this, researchers have proposed various new metrics for measuring the distortion of a depth map in terms of the geometrical distortions it will introduce to the synthesized views [55–59]. The 3DV-ATM codec software also employs such a distortion metric called the *View Synthesis Distortion* (VSD) [76], and is adopted by JCT-3V because of its effectiveness and simplicity.

For a block in a $K \times L$ depth map frame with a corresponding $M \times N$ texture frame, VSD is calculated as:

$$\begin{aligned} \text{VSD} = \sum_{k,l} \Big[& \underbrace{s|D(k,l) - \overline{D}(k,l)|}_{\text{disparity error}} \cdot \\ & \underbrace{\frac{M}{K}\frac{N}{L}\sum_{i,j}\frac{1}{2}(|\overline{I}(i,j) - \overline{I}(i-1,j)| + |\overline{I}(i,j) - \overline{I}(i+1,j)|)}_{\text{simple measure of corresponding amount of texture}} \Big]^2 \end{aligned} \tag{5.9}$$

where $i \in [\frac{k \cdot M}{K}, \frac{(k+1) \cdot M}{K})$, $j \in [\frac{l \cdot N}{L}, \frac{(l+1) \cdot N}{L})$.

VSD consists of the multiplication of two parts. The first part calculates the disparity error, which translates to the amount of error in shifting the corresponding texture pixels. The second part is a simple measure of the amount of texture corresponding that specific depth map pixel. With this design, VSD aims to yield high values only when there is a large error in disparity that also corresponds to a highly textured area.

Since VSD does not use any depth-specific assumptions, it works effectively for warp map coding as well. When enabled, it yields some coding gain for Warp both for anchor and synthesized views, and this can be seen from Table 5.1 and 5.2.

Table 5.4: Average computational complexity of the coding processes in comparison to Depth-1.0.

Operation	Depth-0.5	Depth-0.25	**Warp**
Encoder	74.1%	68.4%	68.6%
Decoder	102.4%	184.4%	64.8%
Synthesis	99.9%	99.6%	83.2%

5.3.5 Computational Complexity

As discussed in Section 5.2, the proposed codec is implemented in C++ without any specialized libraries and does not use GPU hardware acceleration. This ensured that we can reliably compare complexity of the proposed codec with its depth-based counterpart directly from the run-time measurements. The computational complexity comparison of the competing coding methods against Depth-1.0 is reported in Table 5.4. These results are the average complexities calculated over all sequences and QP levels.

As expected the encoding process with reduced resolution depth map and warp map inputs require less computation. This is because of the reduction in the size of the input data and the encoder complexities are actually proportional to the resolution of the depth/warp maps.

Contrarily, due to the upsampling and post dilation filtering operations, the decoding process for Depth-0.5 and Depth-0.25 require more computation than Depth-1.0. Therefore, the bitrate savings from using reduced resolution depth maps come at the expense of increased computational complexity. On the other hand, since Warp does not require any post processing on the decoded warp maps, its decoding complexity is only 65% of Depth-1.0.

Finally, the complexity of the view synthesis is the same for all depth-based methods as the depth maps are upsampled to the same resolution prior to this operation. In contrast, Warp requires less computation also for view synthesis, only 83% of the depth-based methods. This is because IDW uses stitching instead of blending, and unlike DIBR, it does not require a hole-filling operation.

5.3.6 Optimal Depth/Warp QP Selection

In both depth and warp-based coding, the allocation of the total bitrate between the texture videos and their supplementary data is crucial. Every sequence has different statistics, such as varying amounts of texture, motion, and depth, and the optimal bitrate allocation depends on these statistics. Hence, the optimal choice of the set of QPs for texture videos and the depth/warp maps also varies with the sequence. For depth-based coding, various models have been proposed to help the encoder determine the optimal allocation for the best view synthesis quality [50–54]. This problem has not been investigated for warp-based coding.

The CTC predetermines a set of depth QPs for every test sequence and it is clear that these choices might be suboptimal. In this section, we used a full-search algorithm to find the optimal set of QPs for each coding method. To reduce the search space we fixed the texture QPs to the ones from CTC ({26,31,36,41}). Then, for every sequence, we searched for the best set of depth/warp QPs from a set of QPs calculated with offsets {-15,-10,-5,0,4,8,12} from each corresponding texture QP. The set of best depth/warp QPs are simply determined as the combination that yielded the best RD-curve for the synthesized views; and this selection process is depicted in Fig. 5.6. The optimal depth/warp QPs selected by the full-search algorithm are listed in Table 5.5. In order to differentiate with our previous convention, we pad a "*" to the names of each coding method in order to indicate they use the optimal QPs (e.g. Warp becomes Warp*).

As discussed earlier, the statistics of warp maps are very different than depth maps. They contain fewer high frequency elements, yet the distortion introduced for every pixel has a larger impact on the view synthesis. Consequently, the optimal QPs for the warp maps tend to be smaller than the ones from CTC, which can be seen in Table 5.5.

To analyze the RD performance of the compared coding methods, we calculated the BD-Rates again for the optimal set of QPs. To keep the anchor coding method consistent across the paper, the BD-Rates are again calculated with Depth-1.0 as the anchor. These results are provided in Table 5.6 and 5.7.

With the optimal depth/warp QPs, Depth-0.25* still yields the best RD

performance for anchor views with average bitrate savings of 11.5%. This bitrate savings is only marginally better than the performance of Warp* with a difference of 0.05%. For the synthesized views, Depth-0.5* also remains as the best performing method with bitrate savings with an average of 5.2%, whereas Warp* yields 3.4%.

As discussed in Section 5.3.2, BD-Rates for the *Poznan* sequences are skewed for the synthesized views to the advantage of depth-based coding methods. When the average BD-Rates are calculated without these sequences, as presented in Table 5.6 and 5.7, the proposed codec becomes the best performing coding method both for the anchor and the synthesized views with average bitrate savings of 12.1% and 10.3% respectively.

For the optimal QPs, we also calculated the percentage of the depth/warp map bitrates with respect to their corresponding texture bitrates. We observed that these percentages are consistent for different texture QPs per sequence, thus we only report the average percentages in Table 5.8.

Using the same fixed set of texture QPs across all coding methods produces very similar texture bitrates for all of them, with a variation of less than 1%. Therefore, this choice makes it possible to compare the percentages reported in Table 5.8 directly to each other. Based on these results, it is apparent that warp maps can be coded more effectively than any resolution of the tested depth maps. The supplementary data for Warp* brings only 5.5% overhead to the bitstream on average, whereas for Depth-1.0*, Depth-0.5* and Depth-0.25* this overhead is 17.3%, 8.7% and 7.3% respectively.

5.4 Conclusion

In this chapter, we proposed a novel 3DV codec based on warp map coding using 3D-AVC. We showed that due to their small resolution and low frequency content, warp maps can be very efficiently compressed, accounting for an average bitrate savings of 12.8% and 1.5% in comparison to coding full resolution depth maps for anchor and synthesized views, respectively.

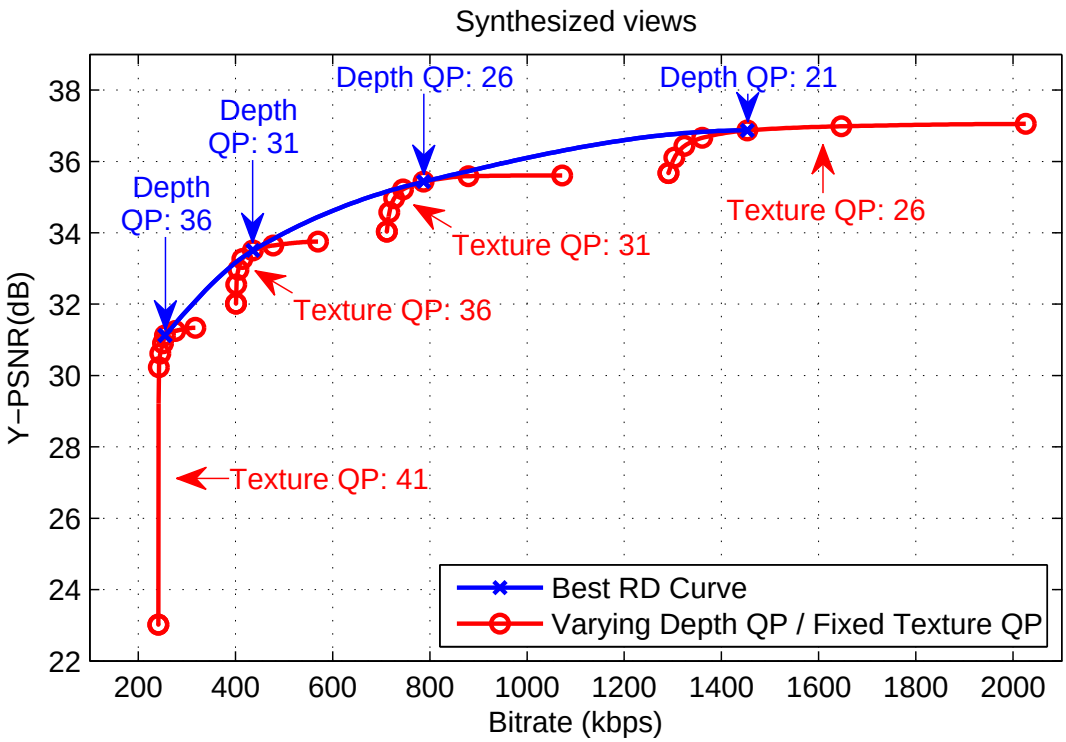


Figure 5.6: Demonstration of the optimal depth QP selection for the *Newspaper* sequence and Depth-0.25.

Due to their similarity, we also compared the proposed codec with depth-based 3D-AVC using reduced resolution depth maps. For anchor views, the proposed codec performed very close to the best performing depth map resolution with only a 0.2% difference in bitrate savings.

In comparison, for the synthesized views, reduced resolution depth maps yielded the best results. We showed that this is due to the bias camera noise brings on the metric we use to compare the quality of the synthesized views. Once the noisy sequences are removed from our calculations, the proposed codec became the best for synthesized views with bitrate savings of 8.8% on average.

Since the proposed codec does not introduce changes to the 3D-AVC syntax, warp maps are treated as depth maps for the depth-based coding tools present in 3D-AVC. We showed that these tools continue working for warp map inputs with only minimal bitrate loss, which is well-compensated for by the gains from warp map coding.

Table 5.5: Optimal depth/warp QPs selected by the full-search algorithm.

Texture QPs	26	31	36	41	26	31	36	41
	Depth-1.0*				Depth-0.5*			
Balloons	34	39	44	49	30	35	40	45
Kendo	34	43	48	49	34	39	44	45
Newspaper	30	35	40	45	26	31	36	41
UndoDancer	21	26	40	45	21	26	31	41
GTFly	30	39	48	49	26	31	40	45
PoznanHall2	30	35	40	45	26	31	40	41
PoznanStreet	34	39	44	45	30	35	40	45
	Depth-0.25*				**Warp***			
Balloons	26	31	36	41	26	31	36	41
Kendo	26	31	36	41	26	31	36	41
Newspaper	21	26	31	36	21	26	31	36
UndoDancer	16	21	26	31	11	21	26	31
GTFly	21	26	31	41	16	21	31	36
PoznanHall2	21	26	31	41	21	21	31	36
PoznanStreet	26	31	36	36	21	21	31	36

Table 5.6: BD-Rate (%) for anchor views for optimal depth/warp QP selection - measured against Depth-1.0 with CTC QPs.

	Depth-1.0*	Depth-0.5*	Depth-0.25*	**Warp***
Balloons	-6.44	-10.75	-12.74	-12.52
Kendo	-12.09	-17.03	-16.48	-16.73
Newspaper	0.00	-6.02	-5.61	-8.33
UndoDancer	6.01	-7.70	-10.80	-10.76
GTFly	-5.62	-11.20	-13.00	-11.91
PoznanHall2	0.00	-9.77	-11.32	-11.75
PoznanStreet	-3.67	-8.93	-10.67	-8.32
Average	-3.12	-10.07	**-11.52**	-11.47
Average w/o Poznan seq.	-3.63	-10.61	-11.72	**-12.05**

Additionally, we analyzed the proposed codec in terms of computational complexity. We showed that it is significantly less complex than all competing methods we compared, for all of the coding processes. We also showed that using reduced resolution depth maps increases the decoder complexity significantly in

Table 5.7: BD-Rate (%) for synthesized views for optimal depth/warp QP selection - measured against Depth-1.0 with CTC QPs.

	Depth-1.0*	Depth-0.5*	Depth-0.25*	**Warp***
Balloons	-2.94	-4.76	-4.83	-4.40
Kendo	-5.73	-9.34	-10.44	-5.81
Newspaper	0.00	0.60	4.45	-14.86
UndoDancer	-2.94	-0.12	4.07	-14.94
GTFly	-2.91	-10.13	-10.80	-11.25
PoznanHall2	0.00	-5.53	-6.71	11.68
PoznanStreet	-1.32	-6.92	-7.62	16.10
Average	-2.26	**-5.17**	-4.55	-3.35
Average w/o Poznan seq.	-2.90	-4.75	-3.51	**-10.25**

Table 5.8: Percentage (%) of the optimal depth/warp map bitrates with respect to their corresponding texture bitrates.

	Depth-1.0*	Depth-0.5*	Depth-0.25*	**Warp***
Balloons	13.52	8.15	5.70	5.52
Kendo	15.72	8.12	8.72	7.45
Newspaper	19.86	12.28	12.80	9.24
UndoDancer	25.10	9.58	7.14	3.57
GTFly	14.34	6.93	4.84	3.52
PoznanHall2	20.94	10.38	7.62	5.49
PoznanStreet	11.93	5.49	3.95	3.93
Average	17.34%	8.70%	7.25%	5.53%

comparison to using full resolution depth maps, so the bitrate savings they bring comes at the expense of increased complexity. On the other hand, the proposed codec reduces the computational complexity while achieving similar or better coding performance relative to using the reduced resolution depth maps.

We also observed that using the same set of QPs for warp and depth maps is suboptimal. We identified the optimal bitrate ratio between texture video and the depth/warp maps, and found that optimal QPs for warp maps tend to be smaller than they are for depth maps. With the optimal set of QPs, in comparison to using full resolution depth maps, the proposed codec yielded average bitrate savings of

11.5% for the anchor views, making it the second best performing coding method but only with a marginal difference of 0.05%.

For the synthesized views, reduced resolution depth maps still yielded better bitrate savings, but once the bitrate savings are recalculated without the noisy sequences, the proposed codec became the best performing coding method by a large margin with an average bitrate savings of 10.3%.

Finally, with the optimal set of QPs, we calculated the bitrate ratio between the depth/warp maps and texture videos. We found that warp maps can be compressed more efficiently than any of the tested depth map resolutions, requiring only a 5.5% overhead to the bitstream.

Acknowledgments

This chapter, in full, has been submitted for publication of the material as it may appear in IEEE Transactions on Circuits and Systems for Video Technology. Bal, Can; Nguyen, Truong Q., IEEE, 2015 [1]. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Conclusion

In this dissertation we developed a framework for comparing Depth Image-Based Rendering and Image Domain Warping in the context of 3D video coding. First, in Chapter 2 we established a common formulation for DIBR and IDW and showed that both rendering methods can be posed as a "reverse mapping" operation. When their corresponding reverse mapping operators are compared, it can be observed that IDW generates renders with significantly lower frequency depth content than DIBR. Due to this characteristic and also by using the "warp map" representation, it is possible to use existing block-based video codecs to compress warps very efficiently, and use warps as an alternative to depth maps in the next generation 3DV codecs.

Then, in Chapter 3, we proposed a fast DIBR-based view synthesis method that synthesizes views with good objective quality and also provides inter-view consistency. The proposed view synthesis method is designed such majority of its main processing blocks can be parallelized and therefore implemented on CUDA to take advantage of multiple cores of the GPU. We also showed that the proposed method requires almost constant time in terms of the number of synthesized views; hence it is very effective for generating content for autostereoscopic displays that require many viewpoints. This chapter is intended to serve as a proof of concept for the feasibility of fast view synthesis with good synthesis quality.

Moreover, in Chapter 4, we presented a new coding tool called the "Depth-based Prediction Mode" (DBPM), and incorporated it into the coding loop of

MVC. Using DBPM, we realized a novel MVD codec and we showed that view synthesis can also be used for better prediction of the anchor views. DBPM uses the supplementary depth data and DIBR to achieve up to 9.2%, 9.9% and 6.7% bitrate savings over MVC for coding MVD data, depth maps and multiview videos respectively. In this chapter, we also provided a complexity analysis for the proposed codec using run time measurements, and showed that its encoder complexity is comparable to MVC. On the other hand, due to the DIBR its decoder complexity is about twice of the decoder complexity of MVC. In this implementation we did not employ any hardware acceleration to enhance the speed of DIBR, but as we showed in Chapter 3, it is possible to take advantage of GPUs to significantly reduce the decoder complexity of the proposed codec.

Finally, in Chapter 5, we established a codec framework based on 3D-AVC that allowed a direct comparison of DIBR and IDW in terms of coding efficiency and computational complexity. 3D-AVC is designed for MVD input and has depth-based coding tools similar to the DBPM already incorporated. It also has support for reduced resolution depth maps; hence warp maps can simply be replaced with depth maps and compressed using 3D-AVC directly without changes to its syntax. According to our findings, depth-based coding tools present in 3D-AVC continue working effectively even with warp map inputs. Using this framework, we showed that IDW achieves better coding performance than DIBR with average bitrate savings of 12.8% for anchor views and 1.5% for synthesized views with significantly lower computational complexity. We also provided an analysis on the effect of camera noise on the view synthesis quality metric and show that noise produces a bias towards better measurements for DIBR. Once the bitrate savings are recalculated using only the sequences without camera noise, IDW starts to provide average bitrate savings of 8.8% in the synthesized views instead of 1.5%. The QPs used in these experiments were based on CTC, but these fixed set of QPs are suboptimal. Instead we determined the optimal bitrate allocation between texture video and the supplementary depth/warp maps, and showed that the proposed codec can achieve average bitrate savings of 12.1% for the anchor and 10.3% for the synthesized views respectively. With this optimal bitrate allocation, in aver-

age warp maps only require a 5.5% bitrate overhead to be incorporated into the bitstream, whereas depth maps require up to 17.3% depending on their resolution.

Bibliography

[1] C. Bal and T. Q. Nguyen, "A Novel 3D Video Codec Based on 3D-AVC and Image Domain Warping," *IEEE Trans. Circuits Syst. Video Technol.*, submitted, unpublished.

[2] L. C. Tran, C. Bal, C. J. Pal, and T. Q. Nguyen, "On consistent inter-view synthesis for autostereoscopic displays," *3D Research*, vol. 3, no. 1, pp. 1–10, Jan. 2012.

[3] C. Bal and T. Q. Nguyen, "Multiview Video Plus Depth Coding With Depth-Based Prediction Mode," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 995–1005, Jun. 2014.

[4] ——, "Depth-based Prediction Mode for 3D Video Coding," in *IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2013, pp. 2187–2191.

[5] Dimenco Displays — 3D Displays. Accessed: 2014-08-30. [Online]. Available: http://www.dimenco.eu/dimencodisplays/3d-displays/

[6] ITU-T, "Advanced video coding for generic audiovisual services," ITU-T Recommendation H.264, Apr. 2013.

[7] K. Müller, P. Merkle, and T. Wiegand, "3-D Video Representation Using Depth Maps," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 643–656, Apr. 2011.

[8] "Call for Proposals on 3D Video Coding Technology," ISO/IEC JTC1/SC29/WG11, Doc. M12036, Mar. 2011.

[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[10] J.-R. Ohm, D. Rusanovskyy, A. Vetro, and K. Müller, "Work Plan in 3D Standards Development," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-B1006, Oct. 2012.

[11] T. Senoh, Y. Ichihashi, H. Sasaki, K. Yamamoto, M. Tanimoto, and K. Suzuki, "AHG8: Report on Relation of GVD Format with Current 3D Video Standardization Tracks," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-C0043, Jan. 2013.

[12] N. Stefanoski and A. Smolic, "AHG8: New warp data for IDW-based view synthesis experiments," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-B0074, Oct. 2012.

[13] C. Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV," in *SPIE Stereoscopic Displays and Virtual Reality Systems XI*, vol. 5291, May 2004, pp. 93–104.

[14] K. Suzuki, M. Tanimoto, and T. Senoh, "AHG08: Technical Description of residual data generation and target view synthesis in GVD (Global View and Depth) 3D Format," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-C0058, Jan. 2013.

[15] M. Lang, A. Hornung, O. Wang, S. Poulakos, A. Smolic, and M. Gross, "Nonlinear Disparity Mapping for Stereoscopic 3D," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 75:1–75:10, Jul. 2010.

[16] M. Farre, O. Wang, M. Lang, N. Stefanoski, A. Hornung, and A. Smolic, "Automatic content creation for multiview autostereoscopic displays using image domain warping," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, Jul. 2011, pp. 1–6.

[17] A. K. Jain, L. C. Tran, R. Khoshabeh, and T. Q. Nguyen, "Efficient Stereo-to-Multiview Synthesis," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2011, pp. 889–892.

[18] W. Su, D. Rusanovskyy, and H. M.M., "3DV-CE1.a: Block-based View Synthesis Prediction for 3DV-ATM," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-A0107, Jul. 2012.

[19] P. Krähenbühl, M. Lang, A. Hornung, and M. Gross, "A System for Retargeting of Streaming Video," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 126:1–126:10, Dec. 2009.

[20] N. Stefanoski, C. Bal, and A. Smolic, "3DV: Results on coding of warps using HEVC," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-A0004, Jul. 2012.

[21] N. Stefanoski, "Report: Subjective testing results on comparing synthesis quality of IDWR and VSRS 1D Fast," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-B0225, Oct. 2012.

[22] ——, "Report: Subjective testing results on comparing warp-based with depth-based synthesis from coded data at same bit-rate," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-C0240, Jan. 2013.

[23] N. Stefanoski, O. Wang, M. Lang, P. Greisen, S. Heinzle, and A. Smolic, "Automatic View Synthesis by Image-Domain-Warping," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3329–3341, Sep. 2013.

[24] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality Video View Interpolation Using a Layered Representation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, Aug. 2004.

[25] K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "View synthesis for advanced 3D video systems," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 438148, 2009.

[26] S. Zinger, L. Do, and P. H. N. de With, "Free-viewpoint depth image based rendering," *Journal of Visual Communication and Image Representation*, vol. 21, no. 56, pp. 533–541, Jul./Aug. 2010.

[27] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "View generation with 3D warping using depth information for FTV," *Signal Processing: Image Communication*, vol. 24, no. 12, pp. 65–72, Jan. 2009.

[28] L. C. Tran, C. J. Pal, and T. Q. Nguyen, "View synthesis based on Conditional Random Fields and graph cuts," in *IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2010, pp. 433–436.

[29] L. Tran, R. Khoshabeh, A. Jain, C. Pal, and T. Nguyen, "Spatially Consistent View Synthesis with Coordinate Alignment," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2011, pp. 905–908.

[30] P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, and T. Wiegand, "Depth image based rendering with advanced texture synthesis," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, Jul. 2010, pp. 424–429.

[31] S.-J. Lin, C.-M. Cheng, and S.-H. Lai, "Spatio-temporally Consistent Multi-view Video Synthesis for Autostereoscopic Displays," in *Advances in Multimedia Information Processing - PCM 2009*. Springer Berlin Heidelberg, 2009, vol. 5879, pp. 532–542.

[32] J. Lu, S. Rogmans, G. Lafruit, and F. Catthoor, "High-Speed Stream-Centric Dense Stereo and View Synthesis on Graphics Hardware," in *IEEE Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2007, pp. 243–246.

[33] S. Rogmans, J. Lu, P. Bekaert, and G. Lafruit, "Real-time stereo-based view synthesis algorithms: A unified framework and evaluation on commodity GPUs," *Signal Processing: Image Communication*, vol. 24, no. 12, pp. 49–64, Jan. 2009.

[34] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Processing*, vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[35] D. Scharstein and C. Pal, "Learning Conditional Random Fields for Stereo," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, Jun. 2007, pp. 1–8.

[36] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Müller, P. H. N. de With, and T. Wiegand, "The effects of multiview depth video compression on multiview rendering," *Signal Processing: Image Communication*, vol. 24, no. 1-2, pp. 73–88, Jan. 2009.

[37] S. Liu, P. Lai, D. Tian, and C. W. Chen, "New Depth Coding Techniques With Utilization of Corresponding Video," *IEEE Trans. Broadcast.*, vol. 57, no. 2, pp. 551–561, Jun. 2011.

[38] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Müller, and H. Rhee, "3D video coding using advanced prediction, depth modeling, and encoder control methods," in *Picture Coding Symposium (PCS)*, May 2012, pp. 1–4.

[39] K. Müller, P. Merkle, G. Tech, and T. Wiegand, "3D video coding with depth modeling modes and view synthesis optimization," in *Asia-Pacific Signal & Information Processing Association Annual Summit and Conf. (APSIPA ASC)*, Dec. 2012, pp. 1–4.

[40] J. Ruiz-Hidalgo, J. R. Morros, P. Aflaki, F. Calderero, and F. Marqués, "Multiview depth coding based on combined color/depth segmentation," *Journal of Visual Communication and Image Representation*, vol. 23, no. 1, pp. 42–52, Jan. 2012.

[41] B. T. Oh, H.-C. Wey, and D.-S. Park, "Plane segmentation based intra prediction for depth map coding," in *Picture Coding Symposium (PCS)*, May 2012, pp. 41–44.

[42] K.-J. Oh, J. Lee, and D.-S. Park, "Depth intra skip prediction for 3D video coding," in *Asia-Pacific Signal & Information Processing Association Annual Summit and Conf. (APSIPA ASC)*, Dec. 2012, pp. 1–4.

[43] K.-J. Oh, S. Yea, A. Vetro, and Y.-S. Ho, "Depth Reconstruction Filter and Down/Up Sampling for Depth Coding in 3-D Video," *IEEE Signal Process. Lett.*, vol. 16, no. 9, pp. 747–750, Sep. 2009.

[44] D. Rusanovskyy, K. Müller, and A. Vetro, "Common Test Conditions for 3DV Core Experiments," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-D1100, Apr. 2013.

[45] P. Merkle, C. Bartnik, K. Müller, D. Marpe, and T. Wiegand, "3D video: Depth coding based on inter-component prediction of block partitions," in *Picture Coding Symposium (PCS)*, May 2012, pp. 149–152.

[46] H. Oh and Y.-S. Ho, "H.264-Based Depth Map Sequence Coding Using Motion Information of Corresponding Texture Video," in *Advances in Image and Video Technology*. Springer Berlin Heidelberg, 2006, vol. 4319, pp. 898–907.

[47] J. Konieczny and M. Domanski, "Extended Inter-View Direct mode for Multiview Video Coding," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 845–848.

[48] S. Ryu and K. Sohn, "Depth-based direct mode for multiview video coding," *Signal Processing: Image Communication*, vol. 27, no. 6, pp. 571–578, Jul. 2012.

[49] W. Su, D. Rusanovskyy, M. M. Hannuksela, and H. Li, "Depth-based motion vector prediction in 3D video coding," in *Picture Coding Symposium (PCS)*, May 2012, pp. 37–40.

[50] Y. Morvan, D. Farin, and P. H. N. de With, "Joint Depth/Texture Bit-Allocation For Multi-View Video Compression," in *Picture Coding Symposium (PCS)*, Nov. 2007.

[51] Y. Liu, Q. Huang, S. Ma, D. Zhao, and W. Gao, "Joint video/depth rate allocation for 3D video coding based on view synthesis distortion model," *Signal Processing: Image Communication*, vol. 24, no. 8, pp. 666–681, Sep. 2009.

[52] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map distortion analysis for view rendering and depth coding," in *IEEE Int. Conf. on Image Processing (ICIP)*, Nov. 2009, pp. 721–724.

[53] H. Yuan, Y. Chang, J. Huo, F. Yang, and Z. Lu, "Model-Based Joint Bit Allocation Between Texture Videos and Depth Maps for 3-D Video Coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 4, pp. 485–497, Apr. 2011.

[54] J. Xiao, T. Tillo, and H. Yuan, "Real-Time Macroblock Level Bits Allocation for Depth Maps in 3-D Video Coding," in *Advances in Multimedia Information Processing - PCM 2012*. Springer Berlin Heidelberg, 2012, vol. 7674, pp. 232–240.

[55] T.-Y. Chung, W.-D. Jang, and C.-S. Kim, "Efficient depth video coding based on view synthesis distortion estimation," *IEEE Visual Communications and Image Processing (VCIP)*, pp. 1–4, Nov. 2012.

[56] Q. Zhang, P. An, Y. Zhang, and Z. Zhang, "Efficient rendering distortion estimation for depth map compression," in *IEEE Int. Conf. on Image Processing (ICIP)*, Sep. 2011, pp. 1105–1108.

[57] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila, "Depth map coding with distortion estimation of rendered view," in *Proc. SPIE Visual Information Processing and Communication*, vol. 7543, Jan. 2010, pp. 75 430B–75 430B–10.

[58] G. Tech, H. Schwarz, K. Müller, and T. Wiegand, "3D video coding using the synthesized view distortion change," in *Picture Coding Symposium (PCS)*, May 2012, pp. 25–28.

[59] H.-P. Deng, L. Yu, B. Feng, and Q. Liu, "Structural similarity-based synthesized view distortion estimation for depth map coding," *IEEE Trans. Consum. Electron.*, vol. 58, no. 4, pp. 1338–1344, Nov. 2012.

[60] E. Martinian, A. Behrens, J. Xin, A. Vetro, and H. Sun, "Extensions of H.264/AVC for multiview video compression," in *IEEE Int. Conf. on Image Processing (ICIP)*, Oct. 2006, pp. 2981–2984.

[61] S. Yea and A. Vetro, "View synthesis prediction for multiview video coding," *Signal Processing: Image Communication*, vol. 24, no. 1-2, pp. 89–100, Jan. 2009.

[62] C. Lee and Y.-S. Ho, "A framework of 3D video coding using view synthesis prediction," in *Picture Coding Symposium (PCS)*, May 2012, pp. 9–12.

[63] F. Jäger and C. Feldmann, "Warped-skip mode for 3D video coding," in *Picture Coding Symposium (PCS)*, May 2012, pp. 145–148.

[64] S. Bosse, H. Schwarz, T. Hinz, and T. Wiegand, "Encoder Control for Renderable Regions in High Efficiency Multiview Video plus Depth Coding," in *Picture Coding Symposium (PCS)*, May 2012, pp. 129–132.

[65] D. Rusanovskyy, M. M. Hannuksela, and W. Su, "Depth-based coding of MVD data for 3D video extension of H.264/AVC," *3D Research*, vol. 4, no. 2, pp. 1–10, Jun. 2013.

[66] Y. Morvan, D. Farin, and P. H. N. de With, "Multiview Depth-Image Compression Using an Extended H.264 Encoder," in *Int'l Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, ser. ACIVS'07. Berlin, Heidelberg: Springer-Verlag, Aug. 2007, pp. 675–686.

[67] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU VCEG, Doc. VCEG-M33, Apr. 2001.

[68] ITU-R, "Methodology for the subjective assessment of the quality of television pictures," Recommendation ITU-R BT.500-13, Jan. 2012.

[69] A. K. Jain, C. Bal, and T. Q. Nguyen, "Tally: A Web-based Subjective Testing Tool," in *Int. Workshop on Quality of Multimedia Experience (QoMEX)*, Jul. 2013, pp. 128,129.

[70] D. Rusanovskyy, F.-C. Chen, L. Zhang, and T. Suzuki, "3D-AVC Test Model 9," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-G1003, Jan. 2014.

[71] P. Aflaki, M. M. Hannuksela, and X. Huang, "CE7: Removal of texture-to-depth resolution ratio restrictions," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-E0035, Jul./Aug. 2013.

[72] S. Lee, S. Lee, H. Wey, and J. Lee, "3D-CE3.a results on dilation filter for depth post processing," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-A0038, Jul. 2012.

[73] D. Rusanovskyy, K. Müller, and A. Vetro, "Common Test Conditions of 3DV Core Experiments," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-E1100, Jul./Aug. 2013.

[74] D. Rusanovskyy, "JCT-3V AHG Report: 3D-AVC Software Integration (AHG3)," JCT on 3DV Coding Ext. Dev. of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Doc. JCT3V-I0003, Jul. 2014.

[75] S. Mitra, *Digital Signal Processing: A Computer-based Approach*, 4th ed. McGraw-Hill, 2010.

[76] B. T. Oh, J. Lee, and D. S. Park, "3D-CE8.a results on view synthesis optimization using distortion in synthesized views by Samsung," ISO/IEC JTC1/SC29/WG11, Doc. M24826, May 2012.