



**HAL**  
open science

# Compression and synthesis for representation of immersive content adapted to 6DoF

Patrick Garus

► **To cite this version:**

Patrick Garus. Compression and synthesis for representation of immersive content adapted to 6DoF. Image Processing [eess.IV]. Université Rennes 1, 2022. English. NNT: 2022REN1S056 . tel-03938942

**HAL Id: tel-03938942**

**<https://theses.hal.science/tel-03938942v1>**

Submitted on 14 Jan 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601  
*Mathématiques et Sciences et Technologies  
de l'Information et de la Communication*  
Spécialité : *Signal, Image, Vision*

Par

« **Patrick GARUS** »

« **Compression et synthèse pour la représentation de contenus  
immersifs adaptés à 6DoF** »

Thèse présentée et soutenue à « Rennes », le « vendredi 07 octobre 2022 »  
Unité de recherche : « Centre de Recherche Inria Rennes Bretagne Atlantique »

## Rapporteurs avant soutenance :

Marek DOMAŃSKI      Professeur, Université de Poznań  
Mohamed-Chaker LARABI      Maître de conférences, Université Poitiers

## Composition du Jury :

Président :	Luce MORIN	Professeur, INSA de Rennes
Examineurs :	Mathias WIEN	Chercheur, RWTH Aachen University
	Marek DOMAŃSKI	Professeur, Université de Poznań
	Mohamed-Chaker LARABI	Maître de conférences, Université Poitiers
Dir. de thèse :	Christine GUILLEMOT	Directeur de recherche, INRIA de Rennes Bretagne Atlantique
Co-dir. de thèse :	Félix HENRY	Ingenieur de recherche, Orange Labs

## Invité :

Thomas MAUGEY      Chercheur, INRIA de Rennes Bretagne Atlantique



# TABLE OF CONTENTS

---

<b>Résumé en Français</b>	<b>7</b>
<b>Introduction</b>	<b>12</b>
<b>1 Principles of Immersive Video Coding</b>	<b>17</b>
1.1 Introduction . . . . .	17
1.2 Immersive Video Formats . . . . .	19
1.2.1 Point Cloud . . . . .	19
1.2.2 Multiview video plus Depth . . . . .	20
1.2.3 Multiplane Images . . . . .	21
1.3 Volumetric Video Coding . . . . .	23
1.3.1 3D-HEVC . . . . .	24
1.4 Depth Estimation . . . . .	31
1.4.1 Cost Volume Construction . . . . .	31
1.4.2 Graph cut optimization . . . . .	35
1.5 View Synthesis . . . . .	36
1.5.1 View Sorting and pre-processing . . . . .	37
1.5.2 Conditional depth merging and warping . . . . .	38
1.5.3 Texture processing . . . . .	39
1.6 Conclusion . . . . .	41
<b>2 The MPEG Immersive Video Standard</b>	<b>43</b>
2.1 Motivation and Limitation of a video-based immersive video standard . . .	43
2.1.1 Simulcast 2D and Multiview Coding . . . . .	44
2.2 The Test Model for MPEG Immersive Video . . . . .	47
2.2.1 MIV Overview . . . . .	47
2.2.2 MIV Profiles . . . . .	47
2.2.3 TMIV Encoder . . . . .	49
2.2.4 TMIV Decoder and Renderer . . . . .	56
2.3 The Common Test Conditions . . . . .	58



2.4	Conclusion . . . . .	63
<b>3</b>	<b>Basic Decoder Side Depth Estimation (B-DSDE)</b>	<b>65</b>
3.1	Motivation for Decoder Side Depth Estimation . . . . .	65
3.2	Experimental Setup to Evaluate B-DSDE . . . . .	67
3.3	Comparative Analysis of the Performance of ESDE and B-DSDE . . . . .	69
3.3.1	Depth Quality . . . . .	69
3.3.2	Synthesis Quality . . . . .	78
3.3.3	Coding Efficiency . . . . .	83
3.4	Opportunities in the B-DSDE system . . . . .	86
3.5	Conclusion . . . . .	88
<b>4</b>	<b>Hybrid Decoder-Side Depth Estimation (H-DSDE)</b>	<b>91</b>
4.1	Motivation for a hybrid DSDE solution . . . . .	91
4.2	Description of the H-DSDE system . . . . .	92
4.3	Experimental results . . . . .	95
4.3.1	Partial Depth Decisions . . . . .	95
4.3.2	Synthesis Quality . . . . .	98
4.3.3	Coding Efficiency of H-DSDE . . . . .	107
4.4	Future Work . . . . .	108
4.5	Conclusion . . . . .	110
<b>5</b>	<b>Feature-Driven Decoder Side Depth Estimation (FD-DSDE)</b>	<b>111</b>
5.1	Motivation for a feature-driven DSDE solution . . . . .	111
5.2	Description of the FD-DSDE system . . . . .	112
5.3	Impact of quantization . . . . .	120
5.4	Coding Efficiency of FD-DSDE . . . . .	121
5.4.1	Depth maps quality . . . . .	123
5.4.2	Synthesis Quality . . . . .	130
5.4.3	Coding Efficiency of FD-DSDE . . . . .	137
5.4.4	Complexity Reduction . . . . .	137
5.5	FD-DSDE with high quality depth maps . . . . .	138
5.6	Comparison of FD-DSDE with 3D-HEVC . . . . .	139
5.6.1	Experimental results . . . . .	141
5.7	Conclusion . . . . .	150

<b>6</b>	<b>The Geometry Absent Profile and the Geometry Assistance SEI message of MPEG Immersive Video</b>	<b>153</b>
6.1	The Geometry Absent Profile of MPEG Immersive Video . . . . .	153
6.2	The Geometry Assistance SEI message of MPEG Immersive Video . . . . .	155
6.2.1	Depth range . . . . .	156
6.2.2	Partitioning . . . . .	156
6.2.3	Depth Estimation Skip . . . . .	157
6.2.4	Depth Estimation Parameters . . . . .	157
6.3	Ongoing research . . . . .	158
6.4	Conclusion . . . . .	158
<b>7</b>	<b>Low Complexity Decoder Side Depth Estimation (LC-DSDE)</b>	<b>159</b>
7.1	Description of the LC-DSDE system . . . . .	160
7.2	Preparation for Evaluation of LC-DSDE . . . . .	161
7.3	Experimental Results . . . . .	163
7.3.1	Depth Maps Quality . . . . .	163
7.3.2	View Synthesis Quality . . . . .	170
7.3.3	Complexity-Distortion Compromise . . . . .	176
7.4	Conclusion . . . . .	177
<b>8</b>	<b>Decoder Side Multi Plane Images (DSMPI) with Geometry Assistance SEI message</b>	<b>179</b>
8.1	Motivation for decoder side MPIs . . . . .	179
8.2	Adapted Test Conditions and Experimental Setup . . . . .	180
8.3	Block-based Basic Decoder Side Multi Plane Images (B-DSMPI) . . . . .	181
8.4	Enhanced B-DSMPI with GA SEI . . . . .	185
8.5	Coding Efficiency of B-DSMPI with Geometry Assistance SEI . . . . .	185
8.5.1	View Synthesis Quality . . . . .	187
8.5.2	Efficiency over all rate points . . . . .	187
8.6	Conclusion . . . . .	200
	<b>Conclusion</b>	<b>201</b>
	<b>Bibliography</b>	<b>207</b>



# RÉSUMÉ EN FRANÇAIS

---

## Contexte

Les normes de codage de la vidéo immersive sont dans une position difficile car le contenu à coder, la vidéo volumétrique ou le light field, n'est pas encore largement utilisé par les consommateurs. L'une des raisons en est que ce contenu nécessite d'importantes quantités de débit binaire pour être traité par les systèmes de décodage classiques déployés dans les appareils des utilisateurs. Ce problème de la poule et de l'œuf est l'une des raisons pour lesquelles les normes précédentes de vidéo immersive, comme les extensions 3D d'AVC et HEVC, n'ont eu qu'un succès commercial limité, voire nul : le coût du déploiement de 3D-AVC ou 3D-HEVC dans les appareils des utilisateurs était trop élevé par rapport à la demande de décodage de ce contenu. La prochaine itération d'une norme de codage vidéo immersive par MPEG est appelée MPEG Immersive Video (MIV) et vise à être beaucoup plus simple et moins chère à déployer que ses prédécesseurs. Cet objectif est atteint en exploitant les codecs vidéo 2D déjà déployés sans nécessiter de modification de ces derniers. Toutefois, cette approche présente plusieurs inconvénients : la quantité de données que ces codecs peuvent traiter est limitée et quantifiée par ce que l'on appelle le taux de pixels. Le taux de pixels est défini comme le nombre de pixels luma par seconde, qui peuvent être décodés par un décodeur vidéo 2D matériel donné. Le respect de la contrainte du taux de pixel est un défi supplémentaire que les normes de codage vidéo immersif doivent relever, contrairement aux codecs 2D. Un autre inconvénient est que les différents types d'attributs ainsi que la géométrie de la vidéo volumétrique sont codés par le codec vidéo 2D déployé. Les formats vidéo volumétriques avancés comportent plusieurs attributs supplémentaires en plus de la couleur. La géométrie est souvent représentée en termes de cartes de profondeur, qui contribuent également au taux de pixels. Par conséquent, le nombre de caméras pouvant être codées dans un tel système est limité par le nombre d'attributs supplémentaires, outre la couleur, la géométrie et d'autres propriétés comme la résolution et la fréquence d'images. Toutefois, pour obtenir une véritable expérience immersive (*i.e.* *6 Degrees of Freedom (6DoF)*), plusieurs caméras doivent être utilisées pour capturer la scène. Enfin, on s'attend à ce qu'un tel système soit inefficace en termes

de codage de profondeur, car les codecs vidéo 2D déployés n'utilisent pas d'outils de codage de profondeur dédiés.

## Motivation et objectifs

Ce travail est initié au tout début des activités de normalisation du MIV. Les recherches et propositions menées doivent respecter les contraintes données de cette activité et visent à réaliser un système de codage vidéo immersif de haute qualité, qui atténue l'inconvénient d'un codec vidéo immersif basé sur la vidéo, *i.e.* reposant sur les codecs 2D existants. La qualité d'un tel système est mesurée en termes d'efficacité de codage, en réduisant le débit binaire requis pour coder la vidéo volumétrique tout en améliorant simultanément les performances de synthèse de la vue. Les inconvénients du système donné sont l'exigence du taux de pixels et le traitement inefficace de l'information de profondeur par les codecs 2D classiques.

Tout d'abord, nous comprenons que les systèmes précédents de codage vidéo immersif (l'"état de l'art") sont de nature complètement différente car ils suivent des contraintes différentes. 3D-HEVC n'a pas été contraint par des limitations de taux de pixel pendant le développement, puisque des configurations à 3 vues ont été testées. En outre, les cartes de profondeur ont été codées efficacement à l'aide d'outils de codage de la profondeur. Ces normes précédentes n'ont donc pas été confrontées à la situation à laquelle nous sommes actuellement confrontés dans le contexte de la MIV. Comme nous avons affaire à des contraintes différentes, le système fondamental qui sous-tend les normes précédentes peut être entièrement remis en question. Un point commun entre les systèmes précédents, et aussi le point de départ de la MIV, est le codage des cartes de profondeur. Les cartes de profondeur peuvent généralement être dérivées des textures par l'estimation de la profondeur et donc, en principe, être dérivées du côté du décodeur. Toutefois, cela n'a pas été considéré comme une solution pratique, car l'estimation de la profondeur est généralement un processus complexe, qui peut être impossible à réaliser du côté du décodeur. En outre, des outils spécialisés dans le codage de la profondeur et de l'inter-composante peuvent compresser ces cartes de profondeur de manière efficace. Ceci est certainement vrai compte tenu des contraintes du codec et de la complexité de la vidéo volumétrique traitée lors du développement de 3D-HEVC. Cependant, dans le cadre de MIV, nous avons l'intention de reconsidérer le système, dans lequel la profondeur est estimée du côté du décodeur. Ceci est motivé par la situation différente de MIV par rapport à 3D-HEVC : les outils de codage

de la profondeur ne peuvent pas être utilisés et par conséquent, les cartes de profondeur seront codées de manière nuisible et inefficace. En utilisant l'estimation de la profondeur côté décodeur, cette situation peut être entièrement évitée. Un autre aspect est le taux de pixel : dans le système conventionnel, les cartes de profondeur sont codées à l'aide de codecs 2D et contribuent au taux de pixel dans les mêmes proportions que la composante de texture correspondante. Comme le taux de pixels est une quantité cruciale à gérer, l'estimation de la profondeur côté décodeur pourrait immédiatement doubler le nombre de caméras qui peuvent être codées, contribuant ainsi à l'objectif d'atteindre 6DoF.

Par conséquent, nous anticipons plusieurs avantages de l'estimation de la profondeur côté décodeur (DSDE) par rapport à l'estimation de la profondeur côté codeur (ESDE) compte tenu des contraintes MIV. Dans cette thèse, nous montrons que la DSDE est plus efficace que l'ESDE et a un potentiel significatif dans le contexte MIV. Plusieurs améliorations du système DSDE sont proposées et évaluées, qui abordent en outre le problème de la complexité. En raison du lien étroit avec le processus de normalisation, du suivi strict des conditions de test communes et des résultats prometteurs, ce travail a conduit à l'adoption du *MIV Geometry Absent Profile* et du *MIV Geometry Assistance SEI message* de la prochaine norme MIV.

## Résumé des Contributions

Ce manuscrit comprend 8 chapitres, dont les deux premiers résument les éléments fondamentaux nécessaires à l'initiation de ce travail.

**Chapitre 1:** Ce chapitre résume les principes du codage vidéo immersif. Les formats les plus courants et les plus récents, les stratégies de codage et les méthodes de rendu sont présentés.

**Chapitre 2:** Ce chapitre présente la norme MPEG de vidéo immersive et les conditions de test courantes.

**Chapitre 3:** Ce chapitre présente une analyse comparative et détaillée du système DSDE de base proposé (B-DSDE) avec le système ESDE actuellement étudié par le MPEG. Nous montrons que le système B-DSDE est nettement supérieur en termes d'efficacité de codage, d'économie de débit de pixels et de qualité de synthèse de la vue. Nous développons des améliorations potentielles, qui sont présentées dans les chapitres suivants.

**Chapitre 4:** Nous déduisons du chapitre précédent que la B-DSDE est globalement plus efficace que l'ESDE, cependant, localement, le système ESDE peut encore offrir certains avantages. Dans ce chapitre, nous proposons le système DSDE hybride (H-DSDE), dans lequel les informations de profondeur locales sont codées, ce qui a été identifié comme étant supérieur à B-DSDE. La décision est prise sur la base d'un bloc en utilisant l'optimisation du taux de distorsion avec le changement de distorsion de la vue synthétisée. Cette approximation nous permet d'identifier les blocs de profondeur, qui sont plus utiles pour la synthèse de vue, même s'ils sont déformés par un codec vidéo 2D classique. Néanmoins, ce chapitre confirme l'hypothèse selon laquelle le codage des cartes de profondeur à l'aide d'un tel codec est tout simplement inefficace, même s'il est effectué partiellement.

**Chapitre 5:** Dans le cadre des résultats du chapitre précédent, nous proposons un système qui continue à contourner complètement le codage de la profondeur, tout en bénéficiant simultanément des cartes de profondeur de haute qualité présentes du côté de l'encodeur. Nous désignons ce système sous le nom de Feature-Driven DSDE (FD-DSDE), dans lequel des informations secondaires utiles (*features*) sont identifiées dans la profondeur côté codeur. Ces informations sont codées et fournies à l'estimateur de profondeur côté décodeur afin d'améliorer la qualité de l'estimation de la profondeur et, par conséquent, les performances de la synthèse de vues, tout en réduisant considérablement le temps d'exécution de l'estimateur de profondeur.

**Chapitre 6:** Etant étroitement lié aux activités de normalisation du MPEG, ce chapitre résume l'impact des résultats des chapitres précédents sur la norme MIV. Nous montrons comment cette recherche a conduit à l'adoption du profil MIV Geometry Absent, du message MIV Geometry Assistance SEI et de plusieurs expériences et résultats supplémentaires menés par d'autres recherches.

**Chapitre 7:** À partir de ce chapitre, certaines contraintes qui ont été données dans les chapitres précédents sont levées. Nous étendons nos recherches à un format plus récent, appelé images multiplans (MPI), et étudions les performances dans le contexte MIV, si elles sont utilisées du côté du décodeur. Nous introduisons une approche d'estimation MPI basée sur les blocs et montrons que l'impact sur la qualité MPI est mineur par rapport à la référence d'estimation MPI plein cadre. C'est la première condition pour fournir à chaque bloc les informations secondaires identifiées au chapitre 4. Nous montrons enfin que les MPI peuvent bénéficier de manière significative du message SEI d'assistance à la géométrie (ou FD-DSDE),

en améliorant la qualité de la synthèse de vue compte tenu d'un faible nombre de plans de profondeur.

**Chapitre 8:** Ce chapitre propose le DSDE à faible complexité (LC-DSDE). Dans les chapitres précédents, nous avons l'intention de conserver (ou d'améliorer) les cartes de profondeur, et donc la qualité de la synthèse de vue, tout en réduisant la complexité. Cependant, il est assez courant qu'une accélération de l'estimation de la profondeur soit obtenue en sacrifiant la précision. Dans LC-DSDE, nous proposons d'utiliser les informations de mouvement codées dans le flux binaire de texture pour récupérer les cartes de profondeur temporellement par compensation de mouvement. De cette façon, la majorité des étapes d'estimation de la profondeur sont sautées pour les inter-images, ce qui réduit considérablement la complexité. Cependant, comme aucun résidu n'est codé (étant donné la contrainte de ne pas coder la profondeur), une distorsion est introduite, que nous concluons être tolérable étant donné la rapidité obtenue.



# INTRODUCTION

---

## Context

Immersive video coding standards have a difficult position as the content to be coded, the volumetric video or light field, is not yet widely utilized by consumers. One reason is that this content requires significant amount of bitrate in order to be handled by conventional coding systems deployed in user devices. Consequently, an efficient compression system is required in order to enable the spread of immersive video. This chicken-and-egg problem is one of several reasons why previous immersive video standards like the 3D-extensions of AVC and HEVC had little to no commercial success: the cost for deploying 3D-AVC or 3D-HEVC in user-devices have been too high given the demand for decoding such content. The first digital 2D video codecs, starting from H.261, have not been in such a position, as 2D videos have already been widely consumed.

The next iteration of an immersive video coding standard by MPEG is denoted as MPEG Immersive Video (MIV) and aims at being significantly simpler and cheaper to deploy in comparison to its predecessors. This is achieved by exploiting already deployed 2D video codecs without requiring any modification of the latter. However, such an approach comes with several disadvantages: the amount of data these codecs can process is limited and quantified by the so called *pixel rate*. Pixel rate is defined as the number of luma pixels per second, which can be decoded by a given hardware 2D video decoder. Meeting the pixel rate constraint is an additional challenge, immersive video coding standards have to deal with, in contrast to 2D codecs. Another disadvantage is that the different types of attributes as well as the geometry of the volumetric video are coded by the deployed 2D video codec and therefore, contribute to the overall pixel rate. Advanced volumetric video formats come with several additional attributes in addition to color, while geometry is often represented in terms of depth maps. Consequently, the number of cameras that can be coded in such a system is limited by the number of additional attributes beyond color, the geometry and other properties like resolution and frame rate. However, in order to achieve a true immersive experience (*i.e.* *6 Degrees of Freedom (6DoF)*), multiple cameras have to be used to capture the scene. Finally, it is

expected that such a system is inefficient in terms of depth coding, as deployed 2D video codecs do not utilize dedicated depth coding tools.

## Motivation and Goals

This work is initiated at the very start of the standardization activities of MIV. The conducted research and proposals need to respect the given constraints of this activity and aim at achieving a high-quality immersive video coding system, which mitigates the disadvantage of a video-based immersive video codec, *i.e.* relying on existing 2D codecs. The quality of such a system is measured in terms of coding efficiency, reducing the required bitrate to code the volumetric video while simultaneously improving the view synthesis performance. The disadvantages of the given system are the requirement of pixel rate and the inefficient processing of depth information by conventional 2D codecs.

First of all, we understand that the previous immersive video coding systems (the "state of the art") are of completely different nature as they follow different constraints. 3D-HEVC has not been constraint by pixel rate limitations during development, as typically, 3-View configurations were tested. Furthermore, the depth maps were efficiently coded using depth coding tools. These previous standards have therefore not been confronted with the situation we are currently facing in the context of MIV. Since we are dealing with different constraints, the fundamental system behind the previous standards may be questioned entirely. A commonality between the previous systems, and also the starting point of MIV, is the coding of depth maps. Depth maps can typically be derived from the textures through depth estimation and therefore, in principle, be derived at the decoder side. However, this has not been considered a practical solution, since depth estimation is typically a complex process, which may be unfeasible to perform on the decoder-side. Furthermore, dedicated depth and inter-component coding tools may compress these depth maps in an efficient matter. This is certainly true given the constraints of the codec and the complexity of the volumetric video handled during the development of 3D-HEVC. However, in the scope of MIV, we intent to reconsider the system, in which depth are estimated on the decoder side. This is motivated by the different situation of MIV in comparison to 3D-HEVC: depth coding tools cannot be used and therefore, the depth maps will be coded in a harmful and inefficient matter. Using decoder-side depth estimation, this situation may be entirely avoided. Another aspect is the pixel rate: in the conventional system, depth maps are coded using 2D codecs and contribute to the

pixel rate in the same amounts as the corresponding texture component. As pixel rate is a crucial quantity to handle, decoder-side depth estimation could immediately double the amount of cameras that can be coded, contributing to the goal of achieving 6DoF.

Consequently, we anticipate several advantages of decoder-side depth estimation (DSDE) as opposed to encoder-side depth estimation (ESDE) given the MIV constraints. In this thesis, we show that DSDE is more efficient than ESDE and has significant potential in the MIV context. Several improvements of the DSDE system are proposed and evaluated, which additionally address the problem of complexity. Due to the tight link with the standardization process, strictly following the common test conditions and the promising results, this work has led to the adoption of the *MIV Geometry Absent Profile* and the *MIV Geometry Assistance SEI message* of the upcoming MIV standard.

## Thesis Roadmap

This manuscript covers 8 chapters, of which the first two summarize the fundamentals required to initiate this work.

**Chapter 1:** This chapter summarizes the principles of immersive video coding. The most common and recent formats, coding strategies and rendering methods are presented.

**Chapter 2:** This chapter presents the MPEG Immersive Video standard and the common test conditions.

**Chapter 3:** This chapter presents a comparative and detailed analysis of the proposed basic DSDE (B-DSDE) system with the ESDE system currently investigated in MPEG. We show, that B-DSDE is clearly superior in terms of coding efficiency, pixel rate savings and view synthesis quality. We elaborate on potential improvements, which are presented in subsequent chapters.

**Chapter 4:** We deduce from the previous chapter, that B-DSDE is globally more efficient than ESDE, however, locally, the ESDE system may still provide some benefits. In this chapter, we propose the Hybrid DSDE (H-DSDE) system, in which local depth information is being coded, which has been identified to be superior over B-DSDE. The decision is done on a block-basis using Rate-Distortion optimization together with the synthesized view distortion change. This approximation allows us to identify depth blocks, which are more useful for view synthesis even if distorted

through a conventional 2D video codec. Nevertheless, this chapter confirms the assumption that coding depth maps using such a codec is just too inefficient, even if conducted partially.

**Chapter 5:** In scope of the results of the previous chapter, we propose a system, which continuously completely bypasses the depth coding, while simultaneously benefiting from high-quality depth maps present at the encoder-side. We denote this system as Feature-Driven DSDE (FD-DSDE), in which useful side-information (*features*) is identified in the encoder-side depth. This information is encoded and provided to the decoder-side depth estimator in order to improve the depth estimation quality, and therefore, the view synthesis performance while significantly reducing the runtime of the depth estimator.

**Chapter 6:** Being closely related to the standardization activities of MPEG, this chapter summarizes the impact of the findings of the previous chapters on the MIV standard. We show, how this research led to the adoption of the MIV Geometry Absent profile, the MIV Geometry Assistance SEI message and several additional experiments and results conducted by other researchers. Finally, the Geometry Absent profile and Geometry Absent SEI message are presented, which are utilized in Chapter 7 and 8.

**Chapter 7:** Starting from this chapter, some constraints that have been given in the previous chapters are lifted. We extend our research to a more recent format, denoted as Multiplane Images (MPI) and investigate the performance in the MIV context, if utilized on the decoder-side. We introduce a block-based MPI estimation approach and show that the impact on the MPI quality is minor compared to the full-frame MPI estimation reference. This is the first condition, in order to provide each block with the side-information identified in chapter 4. We finally show that MPIs can significantly benefit from the Geometry Assistance SEI message (or FD-DSDE), improving the view synthesis quality given a low amount of depth planes.

**Chapter 8:** This chapter proposes Low-Complexity DSDE (LC-DSDE). In the previous chapters, we intended to retain (or improve) the depth maps, and therefore, view synthesis quality, while reducing the complexity. However, it is quite common that a speed-up of depth estimation is achieved by sacrificing accuracy. In LC-DSDE, we propose to utilize the encoded motion information in the texture-bitstream to recover the depth maps temporally through motion compensation. This way, the majority of depth estimation steps are skipped for inter-frames, re-

ducing the complexity significantly. However, since no residual is coded (given the constraint of not coding depth), a distortion is introduced, which we conclude is tolerable given the achieved speedup.

# PRINCIPLES OF IMMERSIVE VIDEO CODING

---

## 1.1 Introduction

Immersive video coding covers a wide range of topics beyond coding. Particularly depth estimation and rendering, which are huge research domains by themselves, are dominantly discussed in computer graphics rather than in coding. Consequently, research and progress in computer graphics often neglects the requirements and constraints of coding, and vice versa. It can therefore happen that the rendering in coding systems, especially in scope of the standardization activities of the Motion Picture Experts Group (MPEG), may be perceived as too *traditional* by computer graphics experts. At the same time, the huge amount of data and rendering time that is tolerated in computer graphics may seem too *unpractical* by coding experts. Consequently, immersive video coding takes a very peculiar position in between coding and computer graphics as expertise in both are required to design a useful system.

This thesis is strongly related to the design of the overall immersive video system and therefore, every module of the system may play a role in the overall improvement. The main goal is to avoid the coding of geometry, as it is identified as one of the biggest bottleneck in terms of coding gain and limits the amount of color information that can be coded. Consequently, the intention of this chapter is to present the principles of immersive video coding systems, which are necessary in order to initiate research on the system design. Fig. 1.1 depicts the high-level scheme of immersive video. The capturing aspect

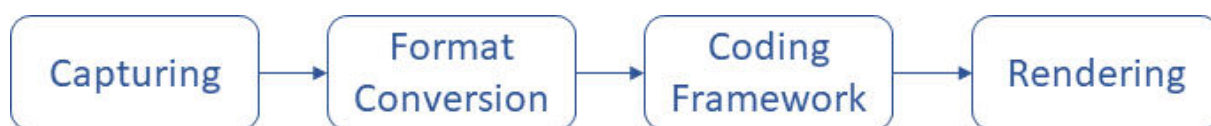


Figure 1.1 – High-level Immersive Video Framework from capturing to rendering.

of immersive video is only marginally relevant for this thesis. However, it is important to be aware of the most important camera types and setups. The main scope of this thesis is multiview video sequences, which have been captured with multiple omnidirectional or perspective cameras. These cameras are typically embedded in a rack in order to fix their positions. The orientation can be divergent and convergent. We deal with sequences with up to 42 cameras. The resolution is typically HD for perspective and up to UHD for omnidirectional cameras. The frame rate is 25 or 30 fps. The cameras can be placed in the real world or virtually in a computer generated imagery (CGI) environment. In addition to the color (or texture<sup>1</sup>) information, depth sensors may also capture the distance of the captured content, which is often stored as depth maps. Depth (or geometry) is a crucial information for the reconstruction of the captured content. Besides of active methods like depth sensors, passive methods like disparity estimation can be used to compute the depth directly from the captured textures. Captured textures, together with the measured or estimated depth maps are often referred to as Multiview plus Depth.

After capturing, an optional format conversion can be performed. The textures and depth maps may be converted into a format, which is expected to be more friendly with the coding framework or enable mathematical tools considered advantageous. As an example, the texture information can be re-projected, based on the given geometry, into 3D space leading to the point cloud format. Neural networks can be used to convert the multiview sequences into more sophisticated formats like multiplane images, which provide transparencies in addition to color information for every considered depth plane. More details on these formats are given in the subsequent section. The encoder of the test model of MPEG Immersive Video can be seen as a format converter, which translates multiview plus depth into an atlas format, which is easier to compress with 2D codecs (see Chapter 3). It is also an example of how the format conversion can lead to a representation with reduced redundancy and therefore lower coding cost - prior to the actual coding.

The coding framework is responsible for the compression of the sequence. In this thesis, we differentiate between two strategies of coding: The *geometry-based* and the *video-based* immersive video coding systems. Geometry-based solutions operate in the domain of the given format. Consequently, coding tools for point clouds are operated in 3D space. In case the geometry is represented as depth maps, dedicated depth coding tools are utilized. Video-based solutions utilize existing 2D codecs in their framework. For that purpose, the

---

1. *Texture* is a commonly used MPEG terminology referring to the color attribute of volumetric video and, throughout this dissertation, shall not be mistaken with its very specific meaning in image processing.

input sequence is typically converted into a format which is more appropriately coded by a 2D codec. In case of point clouds, a special projection of the sequence is performed using virtual cameras. The entire sequence is represented as so called *atlases*. These atlases are then coded by a ordinary 2D video codec. The novel MPEG Immersive Video Standard also falls into the category of video-based solutions. Section 1.3 further details the most recent geometry-based and video-based immersive video coding standards.

The rendering is the final stage of the immersive video pipeline. The goal of the renderer is to synthesize novel, intermediate views, which have not been coded. The view is typically requested by a viewer. The goal of the MPEG group is to achieve 6 Degrees of Freedom (6DoF), which is an often used MPEG terminology referring to the ability of the client to navigate freely through the captured video sequence.

## 1.2 Immersive Video Formats

The continuous light field contains the information of all light rays, flowing through all points and all directions in 3D space. It can be described by the 5D plenoptic function, defining each ray by three spatial coordinates and two angular directions. A sampled representation of the continuous light field can be captured by multiple cameras. In that case, the radiance along the ray is considered constant and the 5D plenoptic function reduces to a 4D light field. The cameras can be omnidirectional or perspective cameras. They can be oriented to a convergent or divergent setup. Active or passive geometry reconstruction allows to associate the captured images with the 3D space. Since immersive video refers to the captured 3D space, it is also often referred to as volumetric video. Different representations of the sampled light field are possible, of which the most prominent are summarized in the following.

### 1.2.1 Point Cloud

During or after capturing of the scene, the geometry is actively measured through depth sensors or passively computed, *e.g.* through depth estimation [1]. The volumetric video can then be represented as a point cloud by unprojecting the captured scene to 3D space, where at each spatial position  $(x, y, z)$  an attribute is given. Attributes describe the properties of the surface through the color information, transparencies, reflectance and many more. Point clouds are often classified in categories like dense, sparse, static



or dynamic. In immersive video, point clouds are typically dense and dynamic [2]. Sparse and static point clouds are typically found in the context of preserving architecture or objects as well as in autonomous driving, in which LiDaR scanners are used to capture surrounding data in realtime [3]. Examples of dense, dynamic and sparse, static point cloud videos are shown in Fig. 1.2 and Fig. 1.3 respectively.



Figure 1.2 – Example for dense and dynamic point cloud video sequences : longdress, loot, redandblack and soldier sequences [2].

### 1.2.2 Multiview video plus Depth

Similar to point clouds, the multiview video plus depth (MVD) format is generated by capturing the light field through multiple cameras and the geometry is either measured or estimated. The geometry is represented as depth maps, which indicate for every color pixel the distance to the captured object. Depth maps are often stored as 8, 10 or 16 bit  $Y, C_b, C_r$  (or YUV) video<sup>2</sup>, where the depth is represented in the luma component or as grey level image. MVD is a very flexible format and has been used for over a decade in the standardization activities of MPEG. Also in the ongoing standardization activity of the next generation immersive video codec and this thesis, the MVD format plays the main role. Examples are shown in Fig. 1.4. The Chess sequence [4] at the top is represented in a equirectangular projection (ERP). The cameras are positioned in a divergent setup, *i.e.* are positioned on a sphere, pointing outwards. This is a computer-generated content and therefore, the depth maps have been generated using the computer model. In this

---

<sup>2</sup>.  $Y, C_b, C_r$  is commonly shortened as YUV in the context of standardization and used in the remainder of this manuscript.

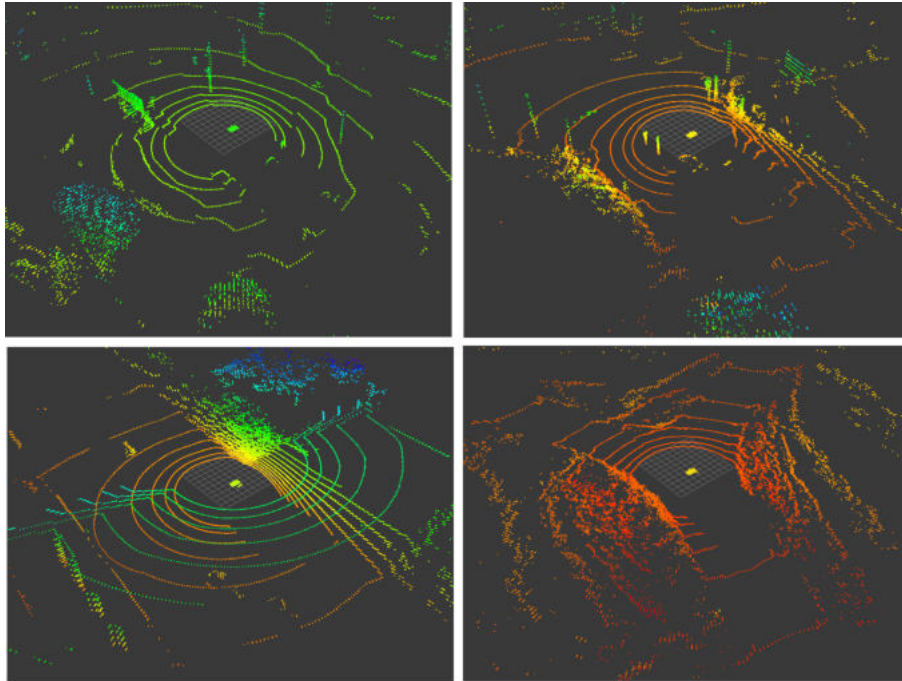


Figure 1.3 – Example of sparse point clouds in autonomous driving, clockwise from top-left : junction approach, junction exit, motorway bridge and navigation turns [3].

thesis, we exclude ERP content from our analysis, because they were not supported by the used depth estimator. The Painter sequence [5] at the bottom is represented using a linear perspective projection (LPP). The cameras are positioned in a convergent setup, *i.e.* are positioned on a rack, pointing towards the same direction.

### 1.2.3 Multiplane Images

Multiplane Images (MPIs) have gained popularity recently among computer graphics expert for enabling high-quality rendering despite the usage of low-complex and simple view synthesis approach [6] [7]. Furthermore, it is very straightforward to train neural networks to estimate this format. Again, the starting point is a set of captured images of the scene. A network estimates MPIs from these images, converting each image towards a volume, in which the third dimension represents a fixed distance between camera and object. For each of these depth layers, a color mage and a transparency map are computed. The transparency map represents the *visibility* of the corresponding color values at this depth level. The amount of data stored in this format is significant and has motivated work towards a compressed representation of the MPI, which is simple, as MPIs are very

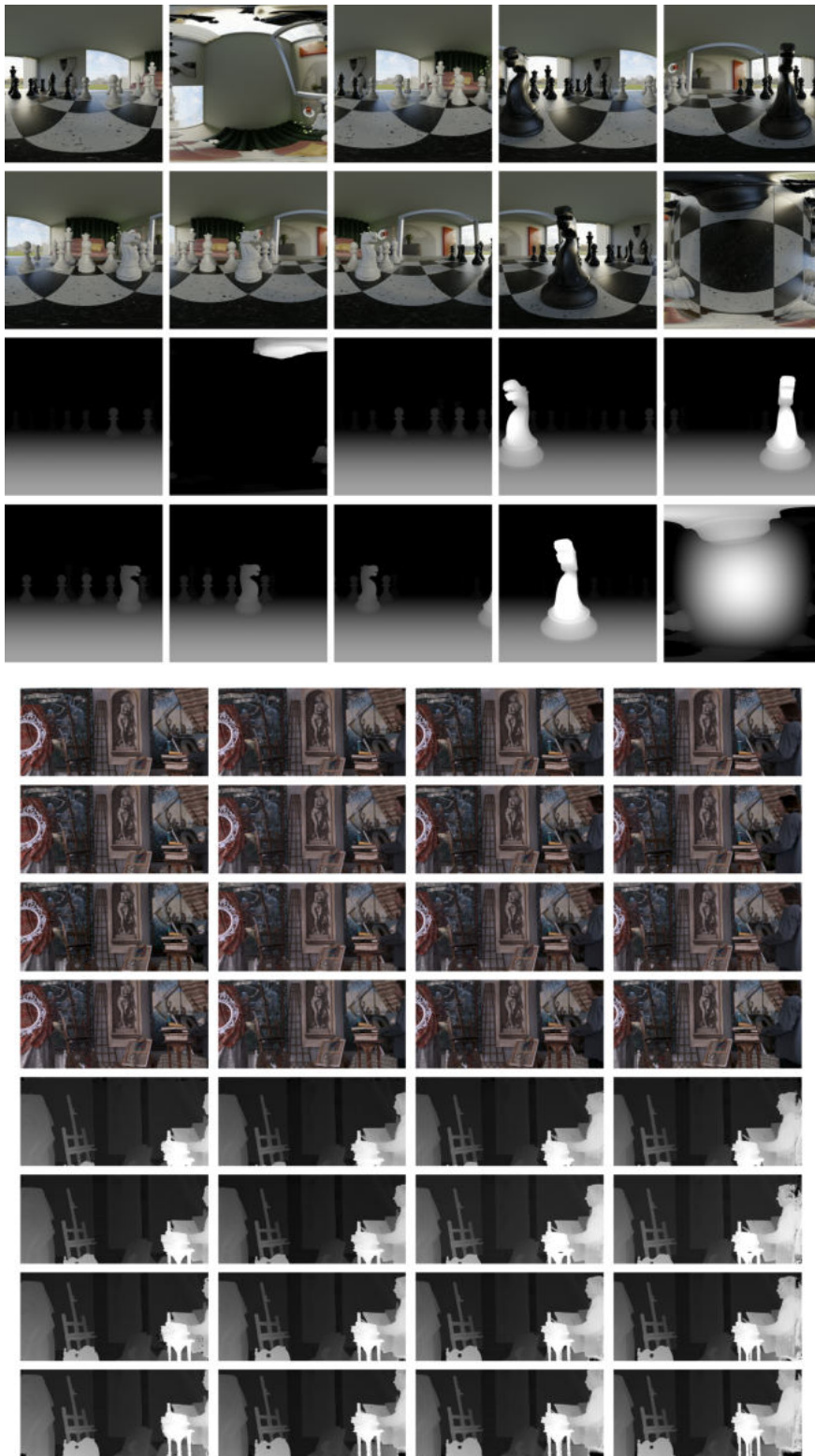


Figure 1.4 – Two examples of the MVD format. Top: Chess sequence. Bottom: Painter sequence.

sparse, *i.e.* containing many zero-valued transparencies [8] [9]. Two examples are shown in Fig. 1.5. Each indicated slice represents a depth plane, for which a color and transparency map is given.

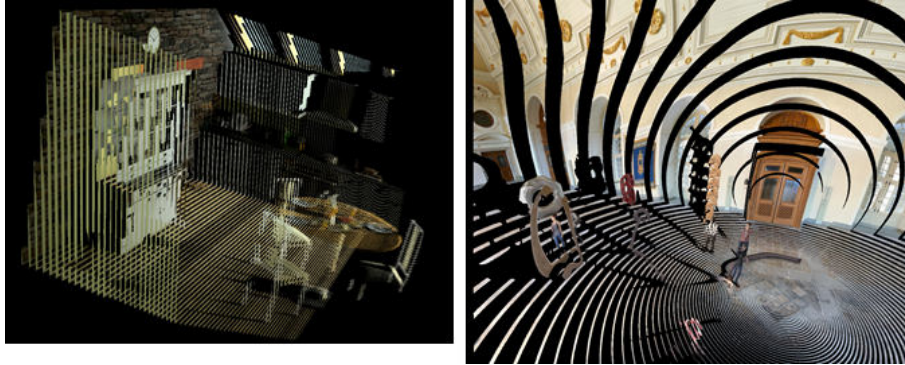


Figure 1.5 – Two examples of the MPI format: Left: Kitchen sequence. Right: Museum sequence.

## 1.3 Volumetric Video Coding

For point clouds, two different standards have been developed by MPEG: the Video-based Point Cloud Compression V-PCC MPEG standard (ISO/IEC 23090-5) and the Geometry based Point Cloud Compression (G-PCC) MPEG standard (ISO/IEC DIS 23090-9). These two standards correspond to two fundamentally different strategies of coding the same format: A video-based and a geometry-based solution.

### Video-based Coding

*Video-based* immersive video coding strategies are characterized by utilizing existing, unmodified 2D codecs to compress the volumetric video [10]. The motivation for such a type of system is the simplicity of deployment. Hardware-based 2D decoders are widely spread in user devices. The immersive video decoder, as it is handling a small amount of data, relative to the volumetric video, can be deployed, e.g. in software as the complexity is minor. The latter does not consider the rendering process, which is not normative. A simple example for a video-based solution is the compression of a MVD sequence by simulcast 2D coding each texture and depth map independently. The 2D coding may have some capabilities to remove inter-view redundancy. Multiview Video Coding [11] [12] (MVC) utilizing h.264 [13] or Multiview High Efficiency Video Coding [14] (MV-HEVC)

utilizing h.265 [15] are examples for that, where removal of inter-view redundancy is possible by simple high-level syntax modifications. V-PCC as well as MIV fall in the category of video-based solutions as well. The point cloud or MVD, in case of V-PCC or MIV respectively, are converted to a format which is considered more friendly to a 2D codec. The V-PCC and MIV bitstreams are merely describing how these formats have to be interpreted by the renderer.

## Geometry-based Coding

*Geometry-based* immersive video coding strategies operate directly in the domain of the format. This typically requires to design and include additional coding tools, leading to significant changes or a completely new decoder. The immersive video decoder is handling all the data, including the volumetric video. Consequently, the deployment of such a decoder may be cost-intensive and may slow down or prevent the deployment of a standard by the industry. In case of point clouds the G-PCC tools operate directly on the attributes and geometry in 3D space. As motion compensation turned out to be very challenging in this domain, G-PCC is typically recommended for static point clouds. For coding of dynamic point clouds, V-PCC seem to be more efficient. Consequently, both standards may not compete with each other, but find their own applications and advantages. In case of MVD the 3D-HEVC depth coding tools operate on the depth maps as well. They are exploiting the properties of depth, *i.e.* smoothness and sharp edges, and utilize view synthesis optimization to compress the volumetric video. However, the additional cost and effort to deploy such a complex decoder on top of already existing codecs has been too big a hurdle considering the expected coding gain. Consequently, 3D-HEVC cannot be considered a successful standard and even with the finalisation of Versatile Video Coding (VVC) in 2021, there are currently no plans towards a 3D extension for VVC. However, 3D-HEVC is the most recent solution for MVD coding and especially the MV-HEVC part, which is still in the video-based category, can be a promising starting point for defining the next iteration of a MVD immersive video codec and is therefore elaborated in more detail in the following section.

### 1.3.1 3D-HEVC

The 3D-HEVC standard can be roughly separated into three components: inter-view and inter-component prediction as well as depth coding tools. Examples for each of these



components are given in the following and based on the test model 11 of 3D-HEVC (HTM11). HTM11 is elaborated in detail in [16].

### Inter-view prediction

Similar to motion compensated prediction (MCP) in temporal coding of 2D video, given a disparity vector, the current view can be compensated from already encoded, neighboring views. It is therefore denoted as disparity compensated prediction (DCP) and is used as an alternative to MCP, as illustrated in Fig. 1.6 [17]. The motion vector

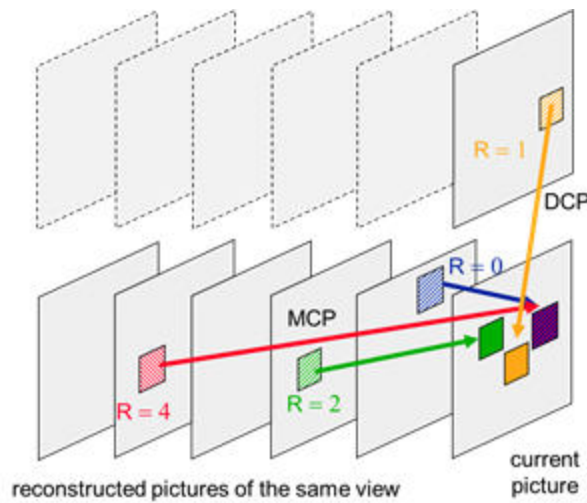


Figure 1.6 – Disparity-compensated prediction and motion-compensated prediction.

prediction is adapted accordingly, predicting motion vectors from motion-compensated blocks using only neighboring blocks that refer to temporal reference frames and prediction disparity vectors from disparity-compensated blocks using only neighboring blocks that refer to reference views. The category of inter-view prediction itself is the main feature of MV-HEVC [14], which is used in the main anchor used in scope of this thesis (see Chapter 2). As already mentioned, MV-HEVC can be realized by mere high-level syntax changes and no additional coding tools are required to perform inter-view prediction.

### Inter-component prediction

Inter-component prediction refers to techniques that exploit the correlation between video and depth signal [18]. In case of 3D-HEVC, already coded depth maps may be available and can be used to improve the texture coding. The disparity vector is derived

from neighboring blocks that use inter-view prediction or from motion vectors obtained by inter-view prediction. This disparity vector is used to identify the corresponding depth block in the decoded depth maps in order to perform backward-warping, allowing to further refine the estimated disparity vector. This concept is illustrated in Fig. 1.7. In

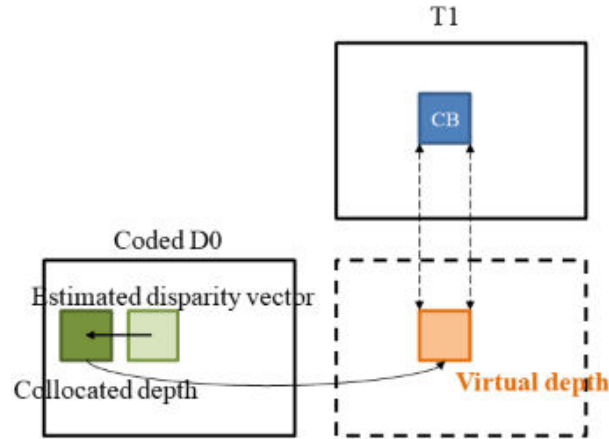


Figure 1.7 – Refinement of the disparity vector utilizing decoded depth maps.

view synthesis prediction (VSP), backward-warping of texture information is used as a predictor for the current prediction unit. The concept is shown in Fig. 1.8. The process entails three steps:

1. A disparity Vector (DV) is derived from a neighboring block.
2. A depth block is fetched from a reference depth view and used as an estimator for the depth information in the current prediction unit.
3. Texture information is backward-warped from a texture reference view according to the estimated depth block. The block derived from this simple synthesis process serves as the predictor for the current block

In the depth-based block partitioning (DBBP) approach, the prediction signal of the texture is constructed based on the corresponding decoded depth map. A segmentation mask  $m_D(x, y)$  is derived from the depth, trying to separate back- and foreground, as indicated in Fig. 1.9. Motion estimation is performed for the corresponding texture, which is partitioned into two  $N \times 2N$  blocks and therefore, two motion parameters are available. Two prediction signals  $p_{T0}(x, y)$  and  $p_{T1}(x, y)$  are derived using the same  $2N \times 2N$  block with both motion parameters, leading to a background and a foreground prediction. Both are merged using  $m_D(x, y)$  and the inversion  $\overline{m_D}(x, y)$ , leading to the final predicted signal  $p_T(x, y)$  as described in Fig. 1.10

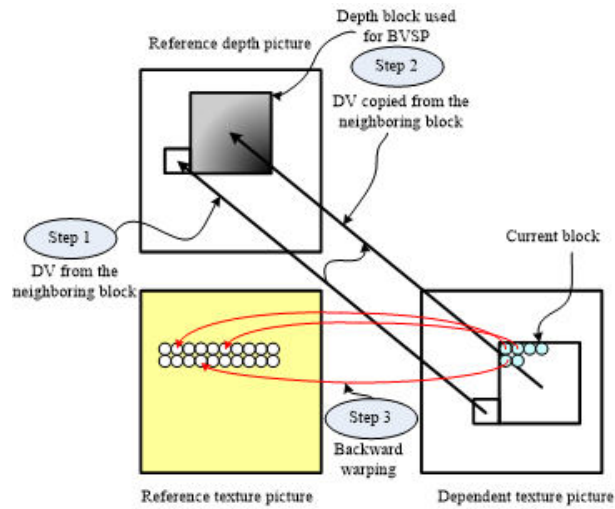


Figure 1.8 – Concept of view synthesis prediction (VSP).

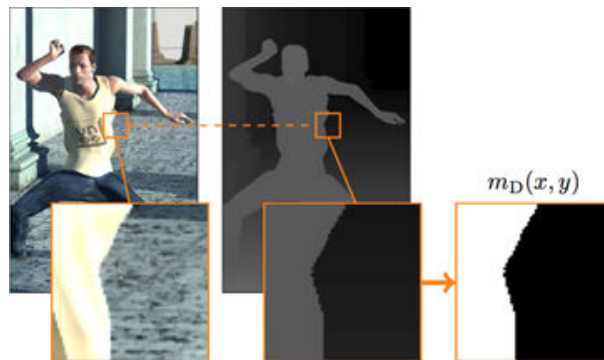


Figure 1.9 – Generation of the segmentation mask using the depth map, in order to compute the prediction signal for the corresponding texture.

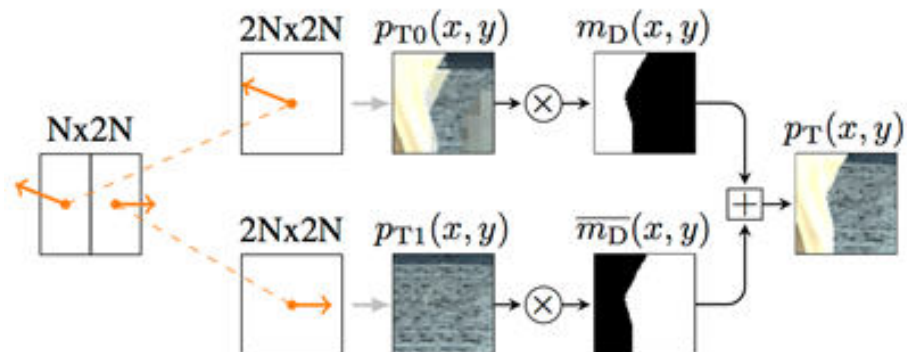


Figure 1.10 – Construction of the prediction signal in DBBP.



3D-HEVC further allows to utilize the motion parameters estimated during texture coding for the subsequent depth inter coding, denoted as motion parameter inheritance. Since the motion of the object may be reflected similarly in the depth maps as well as in the textures, the inherited motion information may serve as an appropriate candidate.

### Depth coding

In order to better adapt to the properties of depth maps, *i.e.* sharp edges and smooth object areas, additional coding tools are included in 3D-HEVC. This is necessary, since intra prediction and transform coding can lead to coding artifacts at object boundaries in depth maps, translating to double-contouring in synthesized views. Two strategies for partitioning are used: wedgelets and contours.

Wedgelets separate a block into two regions  $P_1$  and  $P_2$  by a straight line  $\overline{SE}$ , see Fig. 1.11. The separation between both areas is coded as a partition pattern, which is a binary array, indicating to which area each pixel belongs.

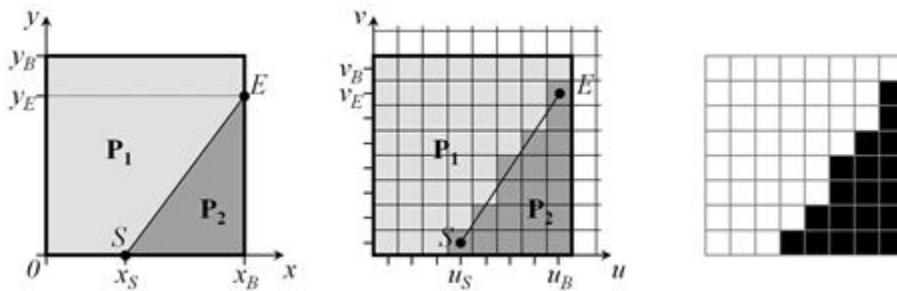


Figure 1.11 – Wedglet-based partitioning. Continuous (left), discrete (middle) and binary partition pattern (right) separation of the areas  $P_1$  and  $P_2$ .

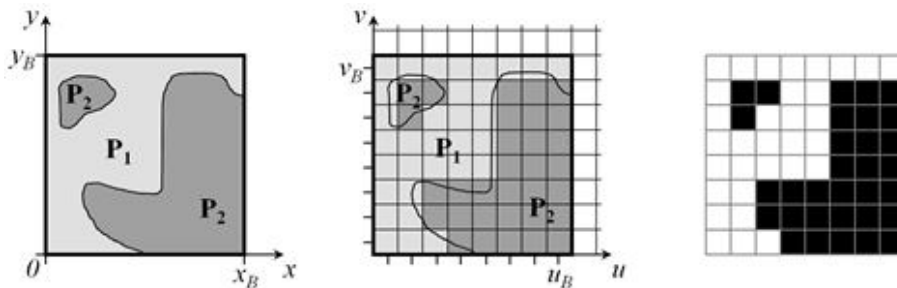


Figure 1.12 – Contour-based partitioning. Continuous (left), discrete (middle) and binary partition pattern (right) separation of the areas  $P_1$  and  $P_2$ .

In addition of the partitioning, a constant partition value (CPV) is used as a predictor, representing the mean depth value of an area. This is motivated by the expected smoothness of depth maps.

The quantized depth maps may not require the full 8bit bitdepth and therefore, the effective bitdepth of the residual can be reduced using a depth lookup table (DLT). The DLT is dynamically constructed and encoded into the bitstream through differential coding or bit map coding.

### Encoder control

In general, the decision of coding modes is done through a rate-distortion optimization (RDO) process using the cost measure

$$D + \lambda R, \quad (1.1)$$

with the estimated rate  $R$  and distortion  $D$  if a certain coding mode is used [19].  $\lambda$  is the lagrangian multiplier that is computed based on the selected quantization parameter. It controls the compromise between the introduced distortion and invested bitrate. This process is usable locally, *e.g.* on a block-level, at the condition that the local distortion and rates are additive and sum up to the total distortion and bitrate of the content. Typically, the distortion between the reconstructed block and the original block is used. In texture coding, the sum of squared differences (SSD) or the sum of absolute differences (SAD) are the most common metrics. The goal of immersive video codecs however, is not primarily the accurate coding of depth maps, but rather high quality view synthesis. While it is an often reasonable assumption, that the most accurate depth translate to the most accurate synthesized view, it cannot be generalized. A more robust option is to additionally consider the view synthesis distortion.

The synthesized view distortion change (SVDC) considers the impact of the currently tested block on the overall synthesis quality. It is a more robust approach to avoid the problem of evaluating occlusions in the synthesized view. A single depth block cannot recover the full synthesized texture block in the target view, as it may require an additional reference view to fully recover the occluded areas. The concept of the SVDC computation is illustrated in Fig. 1.13. The currently tested block is surrounded by decoded blocks and uncoded original blocks. The reference signal  $s_D$  considers the currently tested block as uncoded. The whole depth frame is used in view synthesis leading to the synthesized

view  $s'_T$ . The SSD is computed between  $S'_T$  and the uncoded reference texture  $S'_{T,ref}$  at the same position as the synthesized target view. This distortion is denoted as  $D$ . Then, for every possible coding mode, the currently tested block is reconstructed as above, the whole depth view  $\tilde{s}_D$  is used to compute the synthesized view  $\tilde{s}'_T$ . The distortion is denoted as  $\tilde{D}$ . The SVDC is then defined as

$$\Delta D = \tilde{D} - D. \quad (1.2)$$

$\Delta D$  replaces the distortion measure in the RDO. As every tested coding tool requires re-performing of view synthesis, this approach can quickly get too complex. A partial re-rendering is therefore proposed [20]

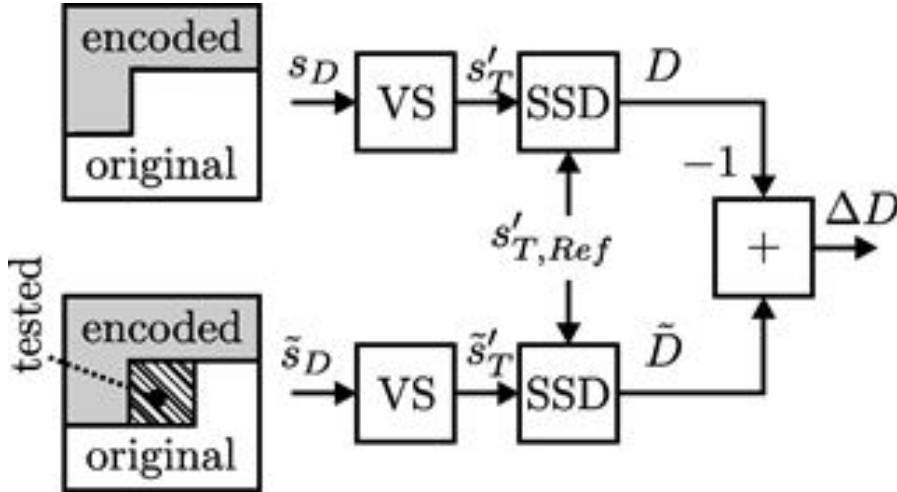


Figure 1.13 – Computation of the SVDC.

The view synthesis distortion can alternatively be estimated using a model, omitting the rendering process. In that case, the horizontal texture gradient is additionally used to modulate the distortion of the depth signal.

$$VSD = \sum_{(x,y) \in B} \left( \frac{1}{2} \alpha |s_D(x,y) - \tilde{s}_D(x,y)| \times (|\tilde{s}_T(x,y) - \tilde{s}_T(x-1,y)| + |\tilde{s}_T(x,y) - \tilde{s}_T(x+1,y)|)^2 \right), \quad (1.3)$$

with the reconstructed texture  $\tilde{s}_T$  and

$$\alpha = \frac{f}{B} \left( \frac{1}{Z_{near}} - \frac{1}{Z_{far}} \right), \quad (1.4)$$

with the focal length  $f$ , the baseline  $B$  between the currently coded and the rendered view and the maximum and minimum depth in the scenery,  $Z_{far}$  and  $Z_{near}$  respectively. The final distortion is then a normalized, weighted sum of the depth distortion and the synthesis distortion [18].

$$D = \frac{w_{syn} \times D_{syn}(m_{DL}) + w_{dep} \times D_{dep}(m_{DL})}{w_{syn} + w_{dep}} \quad (1.5)$$

The presented tools and strategies of 3D-HEVC are considered a summary and are not complete. The standardization process has motivated a lot of research in depth coding tools [21], continuously improving compression performance [22] or reducing encoder complexity [23] [24].

## 1.4 Depth Estimation

Depth estimation refers to a set of tools or algorithms which extract scene geometry from a single or from multiple cameras. The category of monocular depth estimation is more specific and has gained significant progress using convolutional neural networks. Stereo and Multi-view depth estimation are often based on the foundation of epipolar geometry. This perspective opens strategies in which depth estimation is done through disparity estimation. Several strategies exist to estimate disparity. In the following sections, the algorithm of the Depth Estimation Reference Software 8 (DERS 8) is explained. The algorithm has a long history in MPEG and has been refined over several years. It is used in all upcoming experiments in this thesis. Beyond DERS8, Immersive Video Depth Estimation (IVDE) [25] has been extensively used throughout the MIV standardization, particularly with rising interest in decoder side depth estimation.

The following block diagram summarizes the process of disparity estimation through global optimization.

### 1.4.1 Cost Volume Construction

The goal is to estimate the disparity value for each pixel of the image. That means, finding the spatial shift of the pixel of the current view relative to another reference view. If it is assumed, that no prior information is available, any possible disparity value must

be tested which points to an existing color value in the target view.

The view for which the depth map is to be estimated is denoted as input view, other available views are denoted as target views. The views have a width  $w$  and a height  $h$ . Every possible disparity value to be tested is denoted as disparity candidate. Without any additional information, the possible range of the disparity candidates is  $d = [0, \dots, w]$ . However, the maximum and minimum disparity candidates are often known from prior analysis or automatically estimated, and therefore  $d = [d_{min}, \dots, d_{max}]$ . Then, the number of disparity candidates to be tested is  $N = (d_{max} - d_{min})p + 1$ , with an optional sub-pixel precision  $p$ . For every image pixel and for every disparity candidate, a (dis-)similarity measure is to be associated. The chosen metric is evaluated between the color value of the current pixel of the the input view and the color value at the target view, which is associated with the current disparity candidate. Often, patches instead of pixels are evaluated in order to reduce noise. The association from the input view to the target view is performed by re-projecting the current pixel of the input view to 3D space and projecting it to the target view. The projection from 3D space to the input view is described by the projection equation [26]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = [I] [R] [W - T] = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} X - t_x \\ Y - t_y \\ Z - t_z \end{bmatrix}, \quad (1.6)$$

with an object point  $(X, Y, Z)$  in 3D world coordinates and the image pixel  $(u, v)$  of the input view.  $[R] = (r_{ij})$  is the rotation matrix,  $[T] = (t_i)$  is the translation vector of the camera,  $f$  is the focal length and  $(p_u, p_v)$  is the principal point of the input view.  $s$  is the distance between the object and the camera lens center.  $[R]$  and  $[T]$  are denoted as extrinsic,  $f$  and  $(p_u, p_v)$  as intrinsic camera parameters. As all other parameters are assumed to be available,  $s$  is the only unknown variable in depth estimation. This situation is further sketched in Fig. 1.14 [26].

In order to describe the re-projection from the input image to 3D space, equation 1.6 needs to be inverted. For that purpose, the equation is modified as follows [26]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \left( \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) \quad (1.7)$$

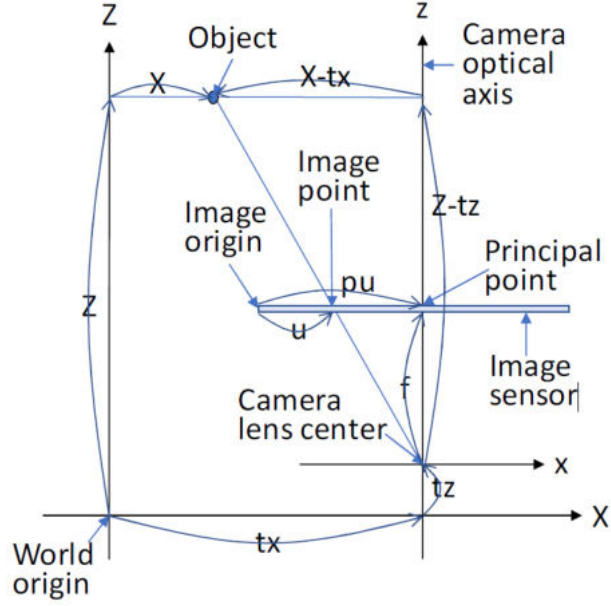


Figure 1.14 – Sketched relationship between the image plane and 3D space [26].

By setting  $[V] = \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$ , equation 1.7 can be simplified to:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \left( \begin{bmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix} \right) = [I] ([R] [W] - [V]). \quad (1.8)$$

By modifying  $[W]$ , the expression can be merged and the projection matrix  $[P]$  can be defined as:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & -v_0 \\ r_{10} & r_{11} & r_{12} & -v_1 \\ r_{20} & r_{21} & r_{22} & -v_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (1.9)$$

Finally,  $[P]$  is extended as follows, enabling to compute the inverse:

$$s \begin{bmatrix} u \\ v \\ 1 \\ 1/s \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = [P] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (1.10)$$

The final re-projection equation projects an image pixel to 3D space:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = s \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \\ 1/s \end{bmatrix} = s [P]^{-1} \begin{bmatrix} u \\ v \\ 1 \\ 1/s \end{bmatrix} \quad (1.11)$$

In order to describe to re-projection from the input view to 3D space, followed by the projection of the 3D space to the target view, both steps can be combined:

$$r \begin{bmatrix} u \\ v \\ 1 \\ 1/r \end{bmatrix} = s [P] [P]^{-1} \begin{bmatrix} i \\ j \\ 1 \\ 1/s \end{bmatrix} = s [H] \begin{bmatrix} i \\ j \\ 1 \\ 1/s \end{bmatrix} = s \begin{bmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ h_{10} & h_{11} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} & h_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \\ 1/s \end{bmatrix} \quad (1.12)$$

The homography matrix  $[H]$  contains only known camera parameters from the input view and the target view. The advantage of this description is, that  $[H]$  can be pre-computed and re-used for any disparity candidate, reducing the complexity. The disparity candidate is converted to depth using the following relationship:

$$s = \frac{fB}{d} \quad (1.13)$$

In DERS8, a weighted sum of absolute difference (SAD) in a window of fixed size of  $3 \times 3$  is used. The cost considers the luma as well as the chroma component of the YUV sequences:

$$C_{u,v} = C_Y + C_U + C_V. \quad (1.14)$$

The luma cost is computed as follows

$$C_Y = \sum K \odot |Y_w^{input}(u, v) - Y_w^{target}(i, j)|, \quad (1.15)$$

with a normally distributed kernel  $K = \begin{pmatrix} 0.05 & 0.09 & 0.05 \\ 0.09 & 0.14 & 0.09 \\ 0.05 & 0.09 & 0.05 \end{pmatrix}$  and the element-wise matrix multiplication  $\odot$ . Due to weighting the SAD with the matrix K, the center pixel contributes more to the luma cost. The chroma cost is considering a single pixel and is set as follows:

$$C_U = 0.15 \times |U^{input}(u, v) - U^{target}(i, j)| \quad (1.16)$$

$$C_V = 0.15 \times |V^{input}(u, v) - V^{target}(i, j)|. \quad (1.17)$$

Given multiple target views, the cost is computed independently for each view. The cost included in the cost volume is the minimal cost among all views. This way, erroneous matches due to occlusions are excluded. The size of the cost volume is finally  $CV = Nwh$

## 1.4.2 Graph cut optimization

Selecting the depth candidate with the minimum cost is not adequate as multiple depth candidates may be associated with a similarly low cost. This is typically the case for low textured areas. DERS8 uses a reliability weight, in order increase the cost of depth candidates associated with homogeneous, low-textured regions. First, the slope is analysed as follows:

$$S = 1.2 \times slope_y^Y + 0.6 \times (slope_{y-1}^Y + slope_{y+1}^Y) + 0.3 \times (slope_y^U + slope_y^V) \quad (1.18)$$

Each cost value is then weighted by

$$R_w = \begin{cases} \frac{R_{th}}{0.3} & S < 0.3 \\ 1 & S > R_{th} , \\ \frac{R_{th}}{S} & else \end{cases} \quad (1.19)$$

with a manually selected threshold  $R_{th}$ . The weight  $R_w$  is higher, the smaller the slope S. As low-textured areas have a smaller slope,  $R_w$  is higher and therefore, the cost of the corresponding disparity candidate is higher, making it less likely to be selected than other candidates with similar cost, but higher reliability.

After filtering the cost volume with the reliability weight, the disparity candidate has



to be selected. In order to achieve a coherence between neighboring selected depth values, a dependency between the depth candidates has to be established. A popular approach is to use a Markov Random Field graph [27] [28], initialized with the cost volume and minimize the total energy using Graph cut with  $\alpha$ -expansion. In addition to the cost volume ( $E_{data}$ ), the energy function is extended by a smoothing constraint ( $E_{smooth}$ ):

$$E = E_{data} + S_w \times \lambda \times E_{smooth}, \quad (1.20)$$

with the depth smoothing weight  $S_w$  and the smoothing coefficient  $\lambda$ . The smoothing term  $E_{smooth}$  increases linearly with the difference of the disparity value of the vertical and horizontal neighbor. Therefore,  $E_{smooth}$  increases the overall energy if the disparity candidate differs to much from the disparity in the neighborhood. Consequently, this term encourages the selection of similar disparity values spatially, contributing to the smoothness of the final depth map. However, this effect has to be attenuated in case a transition between disparity values is wanted. This is done by  $S_w$  defined as:

$$S_w = \begin{cases} 1 & S < \rho \times S_{th} \\ \rho & S > S_{th} \\ \rho \times \frac{S_{th}}{S} & else \end{cases}, \quad (1.21)$$

with a second smoothing coefficient  $\rho$ , a smoothing threshold  $S_{th}$  and the slope  $S$  defined in equation 1.18. A strong slope (large  $S$ ) can indicate a transitioning between foreground and background objects, in which case  $S_w$  will be small, reducing the impact of  $E_{smooth}$ . However, this approach can fail, if an edge is identified, even though a transitioning between foreground and background is not occurring, *e.g.* in checkerboard-patterns.

## 1.5 View Synthesis

View synthesis refers to a set of tools or algorithms, which synthesize novel views from available views. The novel view is typically positioned between the available views, *i.e.* as if captured by a virtual camera. The synthesizer may infer geometry internally or is utilizing explicit geometry, *e.g.* in terms of depth maps. Therefore, depth estimation and view synthesis are often jointly utilized tools. Utilizing depth maps and textures for view synthesis is often referred to as depth image-based rendering (DIBR). In the following, the algorithm of the Versatile View Synthesis (VVS) software is explained. VVS has been

adopted for the standardization process of MIV for 6DoF. Another synthesizer is reference view synthesizer (RVS), which has been used for 3DoF+ sequences, *i.e.* equirectangular sequences. In scope of this thesis, we focus on perspective sequences and therefore 6DoF. Consequently, VVS is used in all upcoming experiments in this thesis.

### 1.5.1 View Sorting and pre-processing

The input to VVS are all (decoded) views, *i.e.* textures  $T^i$ , depth maps  $D^i$ , their camera parameters  $C^i$  and the camera parameters  $C^V$  of the view  $V$  to be synthesized.

The very first step is the sorting of the reference views in descending order based on their closeness to the target view  $V$ . A metric  $\Delta D^i$  is computed reflecting the minimum variation of  $p$  of a depth value that can yield to a displacement  $\delta_p$  in the texture, if a new depth value  $D^i(x, y) + p$  is used instead of  $D^i(x, y)$  for projection:

$$\Delta D^i = \min_{p=1}^N \frac{\{p\}}{\Delta_p > \Theta}, \quad (1.22)$$

with the number of reference views  $N$ .

Next, the depth maps  $D^i$  are re-sampled to 16 bit precision. If the depth maps are already given in 16 bit precision, they are instead re-sampled to 20 bit.  $Z_{near}$  and  $Z_{far}$  of the target view  $V$  are computed, since the depth range can differ from one view to another:

$$Z_{near}^V = \min Z_{near i=1..N}^i \quad (1.23)$$

$$Z_{far}^V = \max Z_{far i=1..N}^i \quad (1.24)$$

The textures  $t^i$  are up-sampled at quarter-pel precision using the H.264/AVC up-sampling filter.

Finally, safe warping areas are identified as it cannot be assumed that the depth maps  $D_i$  are flawless. For that purpose, binary maps  $S^i$  are constructed, indicating for each pixel if it "safe" or "unsafe" to warp. By computing a spatial gradient of the depth maps  $D^i$  on eight different angles, an amplitude and gradient is given for each pixel and thresholded by  $\Delta^i$ . Only the most salient points are retained by filtering the amplitudes in the gradient

direction. Then,  $S^i$  is defined using a  $3 \times 3$  window  $W_3$  centered on a pixel  $(x, y)$ :

$$S^i(x, y) = \begin{cases} 1 & D^i(x, y) > \Delta D^i \forall (x, y) \in W_3 \\ 0 & \text{else} \end{cases}, \quad (1.25)$$

If the ratio of "unsafe" pixels exceed 20%, the corresponding reference view is no longer considered for view synthesis.

### 1.5.2 Conditional depth merging and warping

The depth maps  $D^i$  are warped to the target view position  $V$  using the camera parameters  $C^i$  and binary maps  $S^i$ . In a first pass, a simple pixel-based projection is used. In a second pass, two triangles  $ABC$  and  $ACD$ , which together form a square  $ABCD$ , are projected, with the pixel positions  $[(x_A, y_A), \dots, (x_D, y_D)]$ . For each point of the triangles, it is evaluated using  $S^i$  if the corresponding position is considered "safe" or "unsafe". An entire triangle is only warped, if all positions are considered "safe". If  $(x_A, y_A)$  is the only "safe" position in the triangle, it is solely warped. For the triangle  $ABC$ , the depth value  $d_{new}(x, y)$  is interpolated using the weights  $w_A^V, w_B^V, w_C^V$  (with  $w_A^V + w_B^V + w_C^V = 1$ ), computed based on the distance between  $(x, y)$  and the projected positions  $[A^V, \dots, C^V]$ . Some holes in the projected depth maps  $D_V^i$  can be filled with neighboring values. For each pixel, the number of holes is counted in a  $3 \times 3$  window. If the number of holes in this window is above  $\Theta_{hole1} = 4$ , the hole at the center is not replaced. If the number of holes in a  $5 \times 5$  window is below  $\Theta_{hole2} = 10$ , the hole is replaced with the median of the depth values in this window. The availability of a depth value for each reference is stored in a binary map  $O^i$ .

The next step is denoted as conditional depth merging and further improves the quality of the projected depth maps  $D_V^i$  to derive  $D_{V+}^i$ . A confidence map  $CM^i = (\Delta D^i)^2$  is defined, indicating the robustness of depth values towards variations. A binary reliability map  $F^i$  is computed as:

$$D_{max} = \max_{i=1..N}(O^i(x, y) \times D_V^i)(x, y) \quad (1.26)$$

$$D_{min} = \min_{i=1..N}(O^i(x, y) \times D_V^i)(x, y) \quad (1.27)$$

$$F^i(x, y) = ((D_{max}(x, y) - D_{min}(x, y)) < \Theta_{merge}). \quad (1.28)$$

$\Theta_{merge}$  is defined as

$$\Theta_{merge} = \max(2 \times \sqrt{\sigma}, \frac{1}{N} \sum_{i=1}^N \Delta D^i), \quad (1.29)$$

with the projected inter-view variance  $\sigma^2$ :

$$\sigma^2 = \sum_{y=0}^{height-1} \sum_{x=0}^{width-1} \sum_{i=1}^N (D_V^i(x, y) - \frac{1}{N} \sum_{i=1}^N D_V^i(x, y))^2, \quad (1.30)$$

which is an indicator for noisy depth maps. Large variations require significant merging of the depth maps, which can introduce blurriness.  $F^i(x, y) = 1$  indicates a reliable depth value. The enhanced projected depth maps are finally derived as:

$$D_{V+}^i(x, y) = \sum_{i=1}^N F^i(x, y) \times (CM^i \times D_V^i(x, y) \times O^i(x, y)) + (1 - F^i(x, y))(D_V^i(x, y)) \quad (1.31)$$

### 1.5.3 Texture processing

Each up-sampled texture  $T^i$  is backward-warped to the target view  $V$  using the depth maps  $D_{V+}^i$ , the camera parameters  $C^i$  and the binary maps  $O^i$ . Only texture values are projected, for which a valid depth value is given (indicated by  $O^i$ ). Consequently, the projected textures  $T_V^i$  contain holes and are merged in a next step.

Only a subset of the textures  $T_V^i$ , projected to the target view  $V$ , are used. The decision is based on the depth maps quality. The following condition needs to hold for a reference view to be used:

$$\|d_{max}(x, y) - D_{V+}^i(x, y)\| < \Delta D, \quad (1.32)$$

with  $\Delta D$  computed similar to equation 1.22 with a threshold  $\Theta = 12$  and  $d_{max}$  defined as:

$$d_{max}(x, y) = \max_{i..N}(D_{V+}^i(x, y)) \quad (1.33)$$

The subset of reference views  $sRef$  are used to merge the synthesized view and depth maps as follows:

$$T_V(x, y) = \frac{1}{\text{cards}(sRef)} \sum_{i \in sRef} T_V^i(x, y) \quad (1.34)$$

$$D_V(x, y) = \max_{i \in sRef} (D_{V+}^i(x, y)). \quad (1.35)$$

The remaining holes in the merged texture view  $T_V$  are filled using inpainting.  $\zeta(x, y)$  is initialized to  $O^i(x, y)$ , indicating holes in the merged texture. Three different inpainting techniques are used subsequently. First, the texture boundaries are filled using a line expansion algorithm. A  $11 \times 11$  median filter is used. The left boundary, starting at  $\zeta(0, y)$  is filled with the value generated by the median filter starting from the right boundary of the line of holes. Similarly, the holes in the right boundary  $\zeta(width - 1, y)$  are filled in reverse direction. Second, a template matching inpainting process is used to fill the majority of holes. A confidence map  $CI(x, y) = \zeta(x, y)$  is initialized. A pixel (x,y) is considered as an edge if the four following conditions are met:

1.  $\zeta(x, y) = 1$ : the pixel is not a "hole".
2.  $CI(x, y) \geq 0$ : the confidence is sufficient to consider this edge pixel in the next iteration.
3.  $\zeta(x - 1, y) = 1$  or  $\zeta(x + 1, y) = 1$  or  $\zeta(x, y - 1) = 1$  or  $\zeta(x, y + 1) = 1$ : at least one point, in 4 connectivity, is not a "hole".
4.  $\zeta(x, y + 1) = 0$ : at least one point, in 4 connectivity is a "hole".

For each identified edge pixel a confidence value  $C_{edge}$  is computed as the average of the  $CI(x, y)$  values in a  $11 \times 11$  window. The edge normal  $(n_x, n_y)$  and the gradient  $(g_x, g_y)$  are computed for each pixel. The edge orthogonal is computed as:

$$Orth = n_x \times g_y + n_y \times g_x, \quad (1.36)$$

which is used to give higher weight to gradients that are perpendicular to the edge. A value  $Backg$  is computed as the normalized maximum depth around the position (x,y) in the  $11 \times 11$  window, which corresponds to the background area. If the confidence  $C_{edge} < 0.1$ , the edge is discarded, else a Priority  $P$  is computed as:

$$P = C_{edge} \times (Orth + Backg) \quad (1.37)$$

The pixel with the highest priority is kept for inpainting. In a full search, using a  $11 \times 11$  patch size, the edge pixel is replaced with the average of the two texture candidates associated with the lowest SSD. The template matching inpainting is not performed if only one patch is found and the edge confidence  $CI(x, y)$  is set to -1. The confidence

value is set to the selected edge average confidence. The process is iterated until all points are filled, or if the confidence falls below 0.1. Finally, the last inpainting method fills the remaining holes, where the template matching failed. A spiral search is performed for each line segment with  $\zeta(x, y) = 0$ , where 16 directions are tested. The search ends after a radius of 128 has been reached or if all directions are tested. The texture is filled with the median texture inside a  $7 \times 7$  window centered around the position with the smallest depth identified by the spiral search. Inpainted pixels are indicated by an inpainting map  $I_V$ . For video sequence, the inpainted texture  $T_V$  can be improved by utilizing the information of the synthesized view of the previous frame  $\overline{T_V}$ ,  $\overline{CI_V}$  and  $\overline{I_V}$ . Temporal inpainting is only performed on pixels for which the confidence and inpainting maps of both time instances are equal. In that case, the synthesized texture is modified as follows:

$$T_V(x, y) = \frac{T_V(x, y) + 15 \times \overline{T_V(x, y)}}{16} \quad (1.38)$$

A common artifact in DIBR methods is a tearing on object boundaries. VVS is applying a gaussian filter of size  $5 \times 5$  on pixels, for which a separation between fore- and background is detected. For this purpose, a boundary map  $B_{map}$  using the depth map  $D_V$ .  $\Theta^i$  is used to define a new threshold

$$\Theta_{safe}^V = \min(\Theta_{safe}^i). \quad (1.39)$$

Again, a pixel is considered as "unsafe" if  $D^i(a, b) > \Theta_{safe}^i$  for each  $(a, b)$  in a  $3 \times 3$  window.

## 1.6 Conclusion

In this chapter, the fundamentals of volumetric video compression have been introduced. Two different strategies have been introduced: geometry-based and video-based coding. G-PCC and 3D-HEVC have been summarized as the most recent examples for geometry-based volumetric video compression. In contrast, V-PCC and MIV have been summarized for video-based solutions respectively. In that context, the most relevant depth estimator and view synthesizer, DERS 8 and VVS, have been presented. These algorithms will be used to evaluate the proposals presented in this thesis, following the MPEG common test conditions, which define the configurations for all sequences, which is a significant advantage over alternative methods outside the MPEG context. The MIV activity in MPEG-I Visual is the main motivation for the research presented in this thesis. Consequently, the upcoming chapter will elaborate on the test model for MIV (TMIV) in

detail, the anchors and most importantly, the common test conditions, defining the rules for evaluating proposals.

# THE MPEG IMMERSIVE VIDEO STANDARD

---

The standardization activities related to the MPEG Immersive Video standard has significantly driven the direction of this thesis. The first response to the Call for Proposals (CfP) was evaluated early 2019, several months after the start of this thesis. Consequently, the common test conditions as well as the software is not related to the matured version of MIV, finalized in 2021. Nevertheless, the generic system and its implications were known with the description of the CfP and enabled us predict the disadvantages of the MIV, to find solutions in scope of this thesis and to shape a better version of the MIV. In this section, the incentives of the MIV are described and set in contrast to the previous immersive video standard 3D-HEVC. The MV-HEVC system and anchor are described, which is the governing system for most of this thesis. The technical aspects of the final reference software of MIV are described. Finally, the common test conditions are described, which are strictly followed in scope of this thesis, giving fidelity and robustness to the experimental results.

## 2.1 Motivation and Limitation of a video-based immersive video standard

As the spiritual successor of 3D-HEVC, the MIV standard may seem to take several steps backwards in technical terms. At the same time, the ongoing publication of research papers related to improvements of the 3D-HEVC standard and related encoder optimizations may induce the impression of their relevance and potential. However, since the finalization of the 3D extension of HEVC in 2015, a widely spread deployment failed to materialize. Consequently, 3D-HEVC cannot be considered a successful, industrial standard. A paradigm shift was required in order to satisfy the industrial needs and to lower



the hurdle for implementation. The most significant change is the separation of the 2D video codec from the immersive video standard. In the case of 3D-HEVC, the 3D extension becomes obsolete if the underlying 2D codec is not deployed. Consequently, MIV has been designed to be compatible with any 2D video codecs that are or will be deployed. While this choice may increase the chance of industrial success, it sets limitations on the available coding tools. The most characteristic aspects of 3D-HEVC had to be omitted: inter-view and inter-component prediction as well as depth coding tools. Even though prediction-based coding is known to be very efficient, it requires long prediction structures when coding light fields with many views, which cannot be processed by hardware decoders currently deployed in consumer devices. This matter is quantified by the *pixel rate*, which is defined as the number of luma pixels that have to be decoded per second in order to play a video in real-time. Furthermore, memory limitations make it impossible to store all decoded pictures required to predict all views and frames. However, many views are needed in order to perform high-quality view synthesis and to achieve Six Degrees of Freedom (6DoF), *i.e.*, a free navigation through the scene. As a consequence, the reference implementation of the MIV codec, denoted as Test Model for MPEG Immersive Video (TMIV) [29], is no longer coding all pixels by prediction, but instead discarding some of them through a pruning process. The goal of the pruning is to remove redundant information among the views, which is not required to render any point of view of the scene. What remains are views containing multiple patches of pixels. The patches are packed together into an entity called the atlas, which is the output format of the TMIV encoder. Therefore, an atlas can be seen as a sparse representation of a collection of images. The atlas description and related information is the basis of the MIV bitstream. The pruning process is designed to respect a given pixel rate constraint, enabling the MIV codec implementation to adapt the bitstream to current as well as future capabilities of hardware decoders. The atlas construction is applied to the depth maps as well. An example of texture and depth atlases is shown in Fig. 2.1. The selected full views (*basic views*) are packed into the first atlases and the remaining space is filled with patches (originating from *additional views*). Consequently, one atlas contains mainly full frames, while the other atlas contains mainly patches.

### 2.1.1 Simulcast 2D and Multiview Coding

The most straight-forward way to encode MVD is by using simulcast 2D coding. Every texture and depth view is encoded by a single, independent bitstream. Consequently,

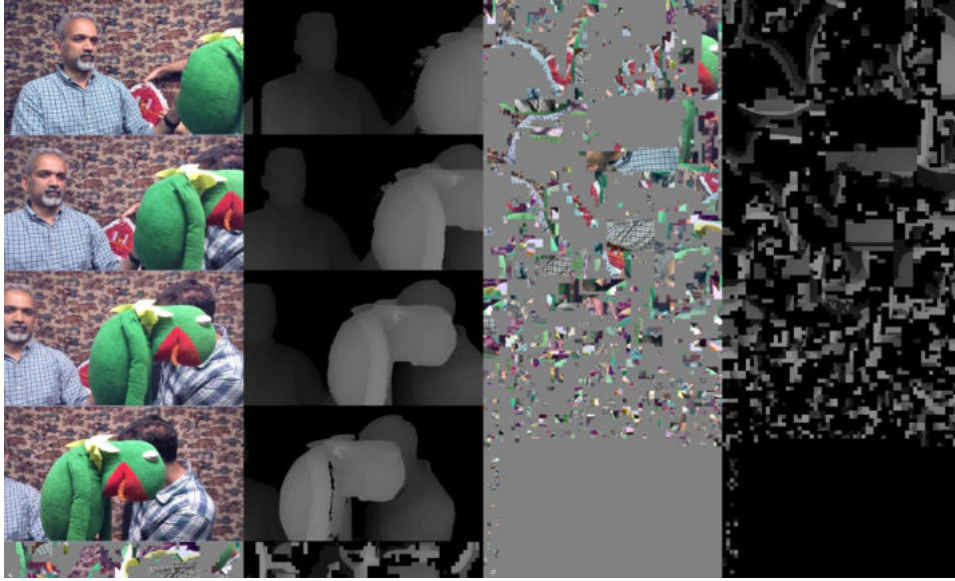


Figure 2.1 – An example of texture and depth atlases for Frog sequence. First column: texture atlas with four basic views and patches, second column: depth atlas with four basic views and patches, third column: texture atlas with patches, last column: depth atlas with patches.

each bitstream has to be decoded by a single decoder instantiation. The MPEG-I Visual group works with the assumption, that four decoder instantiations are reasonable. However, in that case only two textures and two corresponding depth maps can be encoded. Fig. 2.2 illustrates this situation. The two cameras would need to be chosen through a view selection process. Without inter-view coding, the selected views may be selected with minimized redundancy. It can be assumed that the four decoders have multiview capabilities. In that case more views can be coded per decoder. More redundancy among the views can be tolerated as it is efficiently removed through inter-view prediction. The situation is illustrated in Fig. 2.3 with a multiview codec for texture and depth and a total number of codable views  $N$ . The MPEG-I Visual group assumes, that the total decoding capability is limited to 32 Megapixels at 30 frames per second (fps). As a consequence, assuming a typical test sequence resolution of  $2048 \times 1088$  at 30 fps, only  $N=8$  views can be coded. These views have to be spread over the available 4 codecs and therefore, 4 textures or depth maps can be coded with multiview capabilities into a single bitstream.

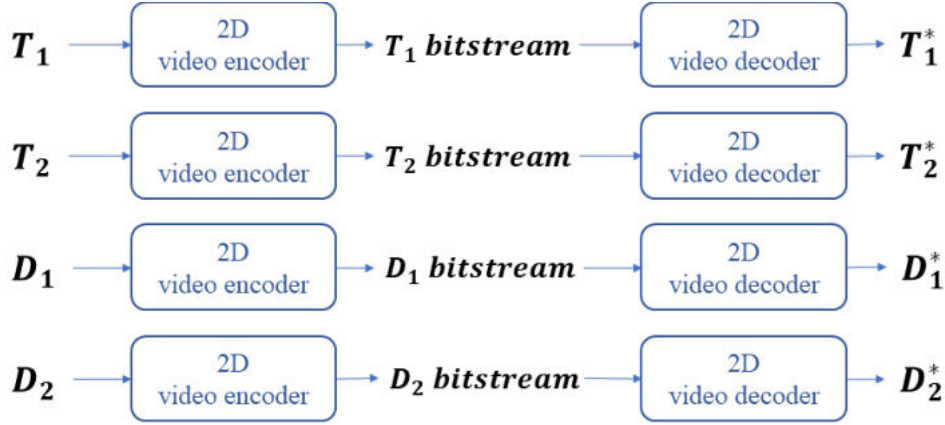


Figure 2.2 – Video-based coding system using a simulcast of 2D codecs. In this setup, two views of the volumetric video can be coded.

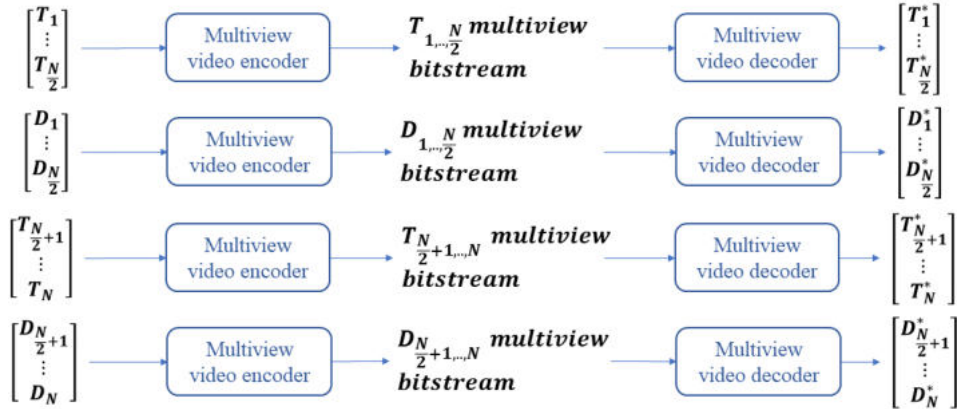


Figure 2.3 – Video-based coding system using a simulcast of 2D codecs with multiview coding tools. In this setup,  $N$  views of the volumetric video can be coded.  $N$  depends on the pixel rate constraint of each decoder instantiation.

Table 2.1 – The MIV profiles. D - depth, T - texture, A - transparency, O - occupancy sub-bitstreams.

Name	Profiles			
	Main	Extended	Extended - restricted subprofile	Geometry Absent
Video bitstreams	D, T	D, T, A, O	T, A	T
Input format	MVD	MVD+	MPI	MVV MVD

## 2.2 The Test Model for MPEG Immersive Video

### 2.2.1 MIV Overview

MIV (ISO/IEC 23090-12) is an extension of the Visual Volumetric Video-based Coding (V3C) MPEG standard (ISO/IEC 23090-5 2nd edition), which describes the syntax, semantics and decoding of any volumetric media. *Volumetric media* refers to formats, which represent a sampled version of the continuous light field or plenoptic function, *e.g.*, MVD, lenslet images or point clouds. The term *video-based* implies that the format is ultimately treated as a 2D video. Another example is the Video-based Point Cloud Compression (V-PCC) MPEG standard (ISO/IEC 23090-5), which is common with V3C and shares many similarities with MIV (the MIV specification is referencing V3C). In contrast, the Geometry based Point Cloud Compression (G-PCC) MPEG standard (ISO/IEC DIS 23090-9) processes the point cloud in its own domain, without converting it to a 2D video format.

### 2.2.2 MIV Profiles

Several profiles are defined, allowing adaptation to the input format. The latter is quite flexible since, as shown in Tab. 2.1, several combinations of texture, geometry, occupancy and transparency are possible.

The first profile is called Main Profile and is designed for MVD coding: in this case, the only attribute is texture, and the geometry is described by depth information. The Extended Profile targets more advanced formats, which, in addition to texture and depth, also include transparency and occupancy. They are consequently referred to as MVD+.

They can be useful to design more sophisticated synthesis algorithms, because DIBR cannot appropriately handle transparent objects like windows.

The restricted subprofile of the Extended Profile has been especially adapted to the Multi-Plane Image (MPI) format [6], which has gained popularity in recent years for providing high-quality view synthesis despite the very simple synthesis process [7]. Instead of providing a single texture value for an explicitly coded depth value (as in MVD), MPIs provide texture for multiple depth levels, each weighted with a certain alpha/transparency value. In order to synthesize a view, the MPI is first generated at the target view and each plane is alpha-blended.

Any additional attribute and occupancy in the extended profile is coded as a video bitstream. While these attributes can improve the view synthesis performance, they increase the general pixel rate and consequently less texture can be coded. Therefore, the overall light field reconstruction can suffer. Especially for MPIs, minimizing the amount of data prior to video-based coding is currently researched [8]. However, removing the redundancy of MPIs for natural content is particularly challenging [9]. As a result, advanced formats have not been able to outperform any MVD-based anchors under CTCs conditions (including MPIs) to this date.

The Geometry Absent (GA) Profile enables the encoding of light fields without geometry, simply referred to as multiview video (MVV). One application concerns densely sampled light fields, for which image-based rendering can lead to sufficient synthesis quality. Another use case is Decoder-Side Depth Estimation (DSDE), in which depth maps are estimated at the decoder-side or in the cloud prior to the rendering. This profile provides the most pixel rate for coding of texture as no other attribute is coded. Given that 2D codecs are particularly designed for texture coding, the GA profile allows for coding of the light field with the least harm. The most important requirement for this goal is to have sufficient redundancy in the coded light field in order to perform depth estimation, which is based on finding matching pixels or blocks among the views. With progress in machine learning-based monocular depth estimation, it can be expected that this requirement will be less stringent in the future.

DSDE is a novel architecture that has been introduced into the MIV standard for the first time based on the research conducted in the scope of this thesis. Accordingly, we denote the system, in which depth maps are encoded as Encoder-Side Depth Estimation (ESDE). During the standardization activities, significant BD-rate performance gains have been reported [30] [31] for DSDE compared to ESDE. Consequently, the Geometry

Absent profile has been adopted into the MIV. The current implementation of DSDE, with a depth estimator that simultaneously estimates the depth maps of all reference views, does not take into account that the DSDE system can take advantage of the knowledge of the desired viewport during the depth estimation process, which would result in lower complexity. With complexity being the main concern of DSDE, an extension of DSDE has been proposed in [32], which reduces the runtime of DSDE significantly while simultaneously improving the view synthesis performance. This approach has been adopted as the Geometry Assistance SEI message in the MIV standard.

The GA Profile is based on the results presented in Chapter 3. The Geometry Assistance SEI message is based on the Feature-Driven DSDE system presented in Chapter 5. For better consistency between the chapters, the TMIV description is separated into two parts. In the current chapter, the ESDE-profile of TMIV is explained. Chapter 6 extends the TMIV description to the Geometry Absent profile as well as the Geometry Assistance SEI message, the standardization process around it, proposals and the ongoing research beyond this thesis conducted by other researchers. It will be outlined, how the adopted technology in the MIV standard differs from the proposals described in Chapter 3 and 5. It furthermore sets the foundation for the experiments conducted in the subsequent chapter 7, in which the Geometry Absent profile as well as the Geometry Assistance SEI message of TMIV are utilized in the context of decoder-side Multiplane Images.

The following description of the TMIV refers to version 11, which is the latest implementation at the time of writing this thesis. As the MIV has reached the final draft international standard (FDIS) at the end of 2021, TMIV11 represents a complete test model implementation of the standardization process.

### 2.2.3 TMIV Encoder

The TMIV Encoder implements one way to derive the MIV Metadata bitstream and to convert the input format into an atlas representation. Even though it is able to include additional attributes like transparencies into the atlases, for the sake of clarity, the following explanations will assume solely texture and depth maps as an input. A high-level block scheme of the TMIV encoder is shown in Fig. 2.4. The input to the TMIV encoder are all uncoded textures, depth maps and camera parameters. Texture and depth atlases are constructed, which contain a subset of views and/or patches. The MIV Metadata encodes the camera parameters and the atlas description. The atlases are independently encoded by a 2D video encoder, which produce the video sub-bitstreams.

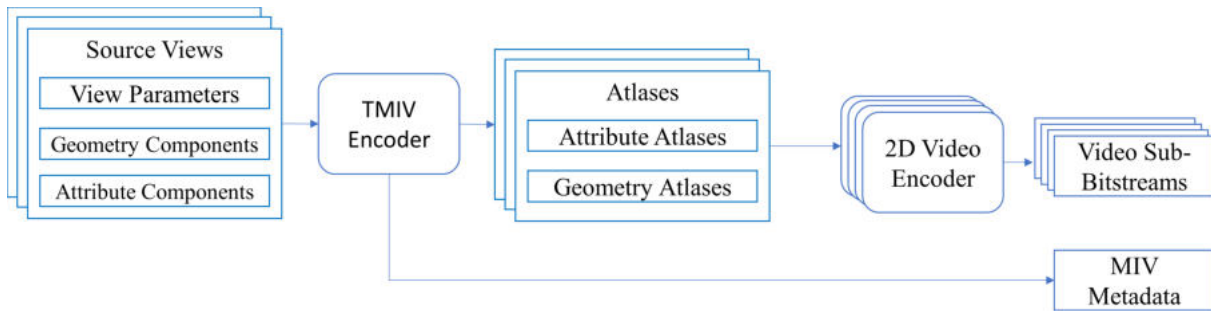


Figure 2.4 – The high-level block scheme of the TMIV encoder [33].

More details of the TMIV encoder are given in the block diagram shown in Fig. 2.5. The encoding is separated into two major steps: the group-based encoding and the single-group encoding.

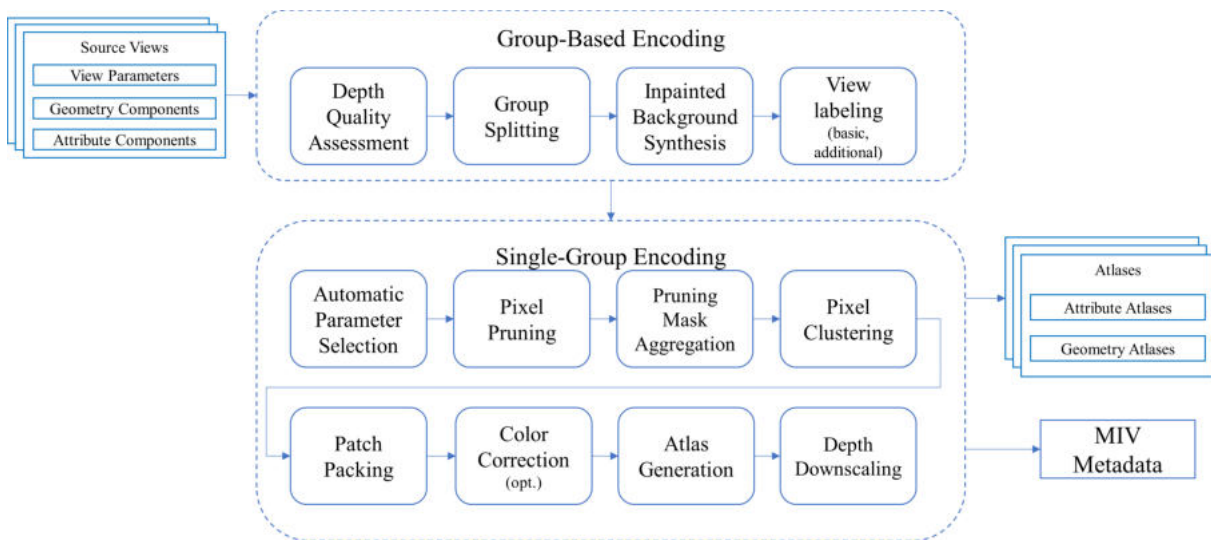


Figure 2.5 – Detailed block scheme of the TMIV Encoder in ESDE mode: group-based and single-group encoding [33].

## Group-based encoding

Group-based encoding refers to a set of steps, which are applied on all source views. One step is the separation of these source views into groups. The number of groups is at least one, which means that all source views are considered a group. The motivation behind the grouping of the source views is to define clusters of views and to process these clusters independently in the single-group encoding stage.

The very first step of the group-based encoding is the assessment of the depth maps quality. The quality is defined by the inter-view consistency of the depth maps. This is done as follows: the very first frame of every depth video sequence is re-projected to the positions of all other source views. The re-projected depth values are compared with the depth values in the target view. This information is used in several later stages of the single-group encoding process, in order to increase the robustness of the TMIV encoder if erroneous depth maps are given.

The next step is the splitting of the source views into groups. Using the camera parameters, the position of each view in 3D space is known. The dominant cartesian axis is selected and certain key points are defined along this axis. The distance of all views to these key points are computed and the views are assigned to these key points in iterations. Finally, clusters of views are given, with each cluster forming a group. The number of groups is defined by the user. The advantage of the grouping process is that processing of views is avoided, which have a long distance to each other and may therefore share little to no commonality.

The goal of the inpainted background synthesis module is to reduce the complexity of the inpainting operation at the decoder-side. First, an intermediate view at the center of all cameras is synthesized. Minor modifications of the internal synthesizer are taken in order to render the background over the foreground. Afterwards, several steps are taken in order to remove all foreground pixels, that may be present on top of the background. The remaining identified background pixels are inpainted using a variation of the push-pull inpainter [34] [35]:

1. **Push:** Starting from the input resolution, a pyramid is constructed of the texture and depth components using linear interpolation. The top of the pyramid is of size  $1 \times 1$ .
2. **Pull:** The starting point is the second-smallest Image. If the depth is larger than zero, the texture and depth is preserved. Otherwise, the texture and depth is averaged using the neighboring samples of the higher layer. The zero depth is maintained, if the higher layer has zero depth as well.
3. The output of the push-pull inpainter is the filtered image at input resolution.

The final, inpainted background view is encoded and used at the decoder-side to reconstruct occluded areas and therefore, avoiding the complex inpainting process.

The push-pull inpainter is initialized with the synthesized background view. In the push process, the resolution is repeatedly halved with a linear interpolation of the texture and



depth, until the top of the pyramid is an image of size  $1 \times 1$ . In the pull process, the process is reverted starting with the second smallest resolution. If a depth value is larger than zero, the texture and depth is preserved and otherwise averaged over the neighboring samples of the higher layer, which do not have zero-valued depth. If the sample does not exist in the higher layer, the zero depth is maintained. The output of the push-pull inpainter is the filtered frame at input resolution.

The final step of the group-based encoding is the view labeling. This module classifies each view into *basic views* and *additional views*. Basic views are preserved during the single-group encoding, while additional views are pruned. Two modes are implemented, which are both used in different anchors of the common test conditions (See section CTC). The first mode is illustrated in Fig. 2.6, in which the category of additional views is enabled. This mode refers to the main anchor of the MIV (denoted as *MIV anchor*). The second mode is illustrated in Fig. 2.7, in which the category of additional views is disabled. Consequently, only basic views are coded, which implies, that the single-group encoder does not require to perform any pruning. This mode is used in an alternative anchor of the MIV (denoted as *MIV View anchor*). The number of basic views can be controlled



Figure 2.6 – View labeling mode 1. Additional views are enabled.

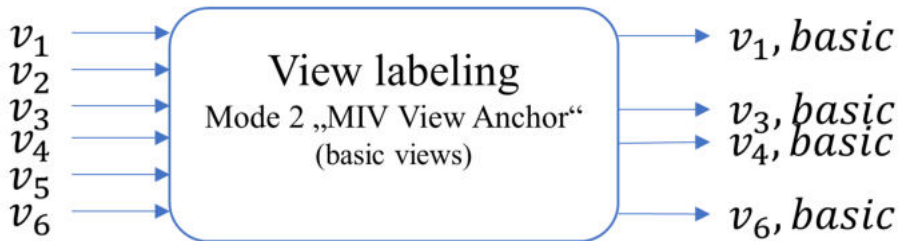


Figure 2.7 – View labeling mode 2. Additional views are disabled.

by the user. The encoder confirms through internal calculations, that the number of basic views can respect the pixel rate constraint considering the number of requested groups.

In case of mode 1, at least one view is classified as additional view, in order to have at least some degree of view pruning. With the finalization of the view labeling, each group is further processed by the single-group encoding.

### Single-group encoding

The very first step is the automatic computation of parameters and the preparation for the atlas construction. All components of the volumetric video, including atlas data, occupancy video data, geometry video data and attribute video data is stored in the atlas. In V3C, some of the components are rescaled as part of the reconstruction. In MIV, the attribute video data is always stored at original resolution, while the geometry can be downsampled by an user-defined integer factor. The number of atlases per group and the resolution of each atlas is calculated automatically. Three restrictions define the atlas resolution:

1. The maximum sample rate (in Hz) of the luma component.
2. The maximum resolution of a frame considering the luma component only. This constraint is given by the decoder capabilities.
3. The total number of allowed decoder instantiations.

The computation is then as follows:

1. number of atlases = number of atlases per group  $\times$  number of groups.
2. luma picture size = atlas frame width  $\times$  atlas frame height.
3. luma sample rate =  $1 + \frac{1}{N^2} \times$  luma picture size  $\times$  frame rate  $\times$  number of atlases.
4. number of decoder instantiations =  $2 \times$  number of atlases.

The following steps are taken to meet these constraints:

1. First, the atlas frame width is set to the widest source view width.
2. The number of atlases per group is set to reach or exceed the maximum luma sample rate, but not more than the maximum number of atlases.
3. The atlases frame height is maximized within the constraints.

The second step is the atlas construction, which consists of pruning and packing. The pixel pruning has two goals: first, remove the inter-view redundancy among the volumetric video, as this task is not assumed to be performed by a multiview codec. Second, the pruning is a complete deletion of the corresponding pixels, as they are not

coded in a predictive way. Consequently, the pixel is not decodable and needs to be reconstructed during the TMIV rendering. However, besides of reducing the bitrate, the pruning reduces the pixel rate required to code the volumetric video and therefore enables a richer representation of the light field. The pruning further aims at considering temporal

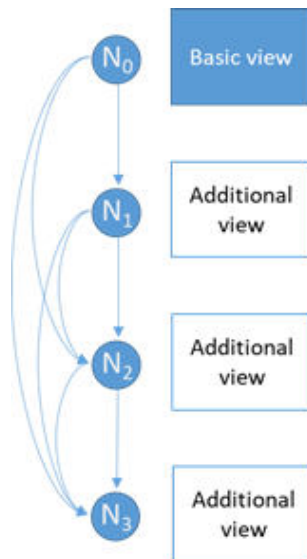


Figure 2.8 – Construction of the pruning graph.

consistency and to retain more, smaller patches. The complexity of the pruning shall remain realistic. The process is indicated in Fig. 2.8 assuming a single basic view and three additional views. First, the basic view is re-projected to all three additional views and a pruning mask is constructed for each. Afterwards, the additional view with the most preserved pixels is used, to further prune the remaining additional views and the pruning mask is further updated. This process is done for all additional views constructing a hierarchical graph. This hierarchy is maintained for future frames, preserving the temporal consistency. The additional views and basic views are clustered, in order to limit the size of the graph and therefore, reduce the complexity of the pruning process. Since the decision to prune a pixel is based on the similarity identified through re-projection, the pruning performance and therefore the bitrate and pixelrate requirement depend on the depth maps quality. The pruning mask is refined in a second-pass considering the global color component differences among the source views. Finally, the pruning masks are aggregated frame-by-frame and reset after each intra period. The thresholds used in the pruning operation can be set by the user and is further adapted using the global luma standard deviation in order to consider different levels of noise in the sequence.

The pruning is performed on a pixel-basis and connected pixels have to be identified in order to define clusters. If a pixel has another clustered pixel in the immediate neighborhood, the pixel is added to that cluster. In order to reduce the number of cluster and therefore, the number of side information to encode, clusters are merged. A cluster is described by a bounding box, including the top-left pixel position, width and height of the box, see Fig. 2.9. In case, the cluster inside the box is irregularly-shaped, a splitting of the cluster is possible, if the bounding box of the resulting two split cluster is smaller by a threshold compared to the original bounding box. The final step of the atlas generation

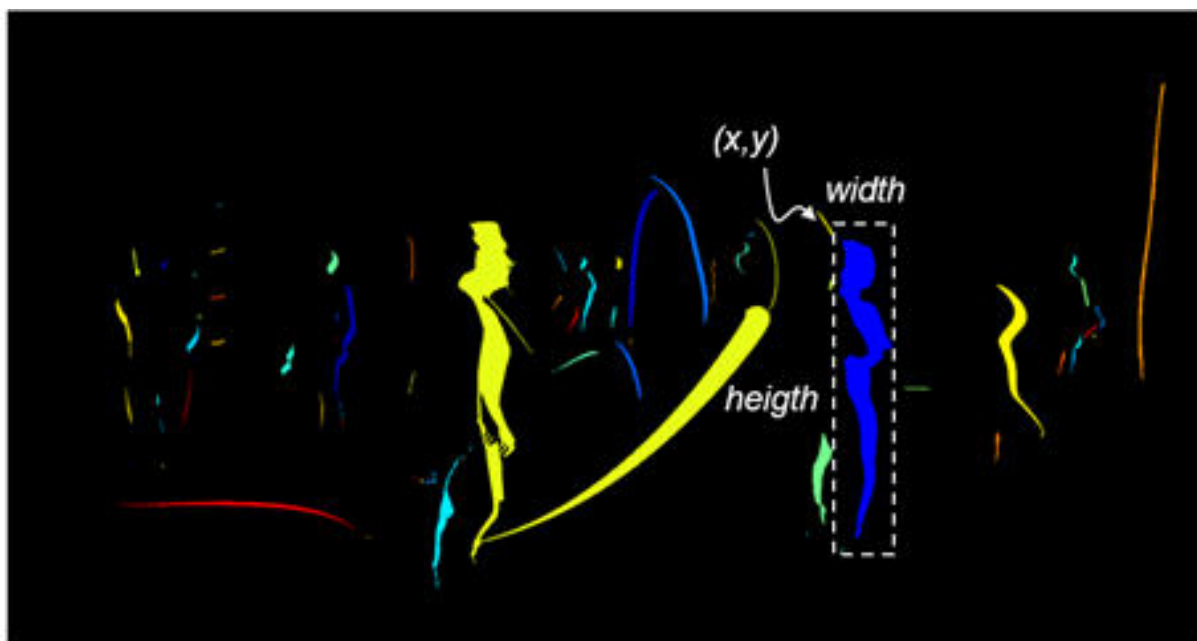


Figure 2.9 – Packing of patches into the atlases, considering free space in the block boundary of merged clusters.

is the packing of clusters into the atlas, which is done using the MaxRect algorithm [36]. The packing process allows for splitting, rotating and flipping of clusters, in order to fit them as best as possible into the available atlas frame. This excludes basic views, because the atlas frame resolution is always computed based on their width. Therefore, packing basic views is simple, since they are included first. Since a cluster is described through a boundary box, free space may be available inside that box, which is used first, as shown in Fig. 2.10. If there is no space left for a cluster, despite rotations, a splitting is performed on the longer border. If one part is smaller than a threshold, it is discarded. The constructed atlas video data comes along with side information, required to recover the patches at the decoder-side. Each patch is indexed and for each patch it is coded to which atlas it

belongs to, the size, position, possible flipping and rotation and the view it originated from. It is further signalled, if a patch is related to the inpainted background view. An optional entity ID can be included. The color correction module is an optional step, which

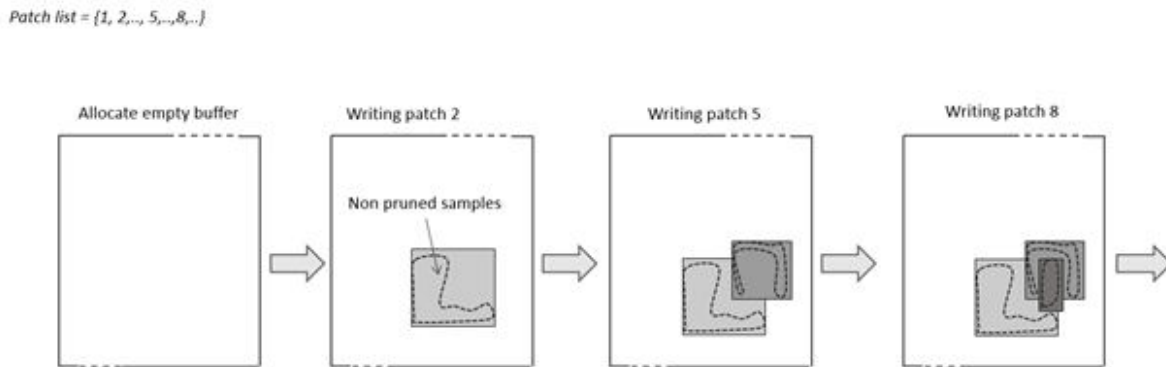


Figure 2.10 – Packing of patches into the atlases, considering free space in the block boundary of merged clusters.

allows for alignment of the color characteristics towards the center view, improving the coding efficiency. The color offset is coded with each atlas allowing for reconstruction of the original color value.

The constructed geometry atlas can be downscaled by a factor of 2 if enabled. A  $2 \times 2$  max pooling filter is used. Geometry downscaling allows for coding of depth with lower pixel rate cost and mitigating impact of compression, since similar bitrate can be spend on a smaller frame. However, the downscaling also reduces the original depth quality and therefore synthesis quality. The consideration of such a technique reflects the urgent need of saving pixel rate. An example for the final constructed atlases are shown in Fig. 2.11.

## 2.2.4 TMIV Decoder and Renderer

The TMIV decoder includes demultiplexing, bitstream parsing, video decoding (*e.g.* using HM oder VTM), frame unpacking and block to patch map decoding following the MIV specification [33]. The TMIV renderer outputs a synthesized view according to the requested target views camera parameters. Similar to 3D-HEVC, the process of rendering is non-normative, which means, any other method can be used. The full TMIV renderer is shown in Fig. 2.12. The patch culling module identifies patches, which have no overlap with the requested target view in order to speed-up the synthesis process as they can be subsequently ignored. All other patches are reconstructed by copying them to the views

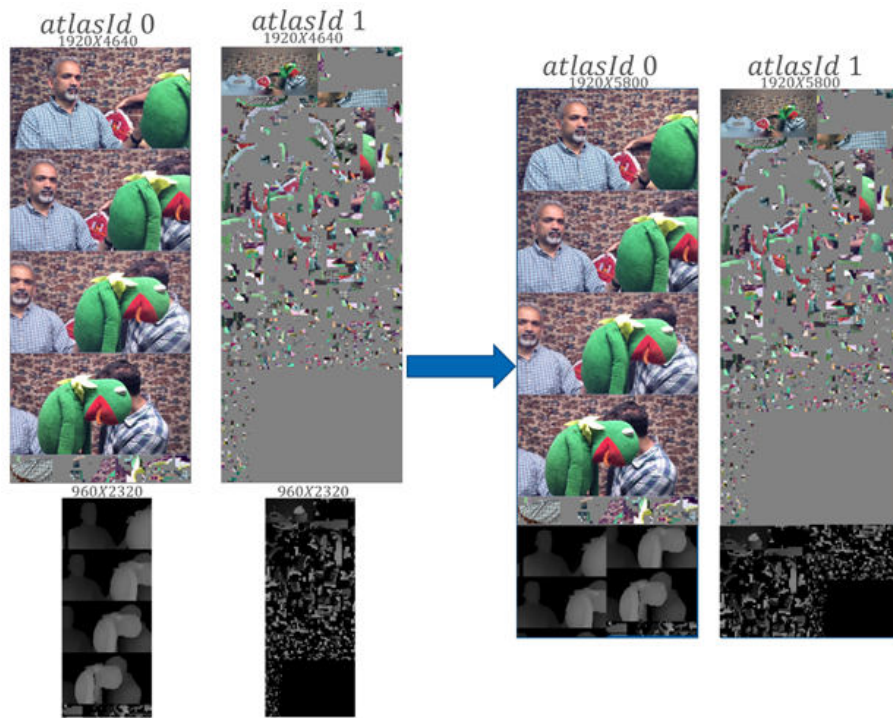


Figure 2.11 – Attribute and downscaled geometry atlases.

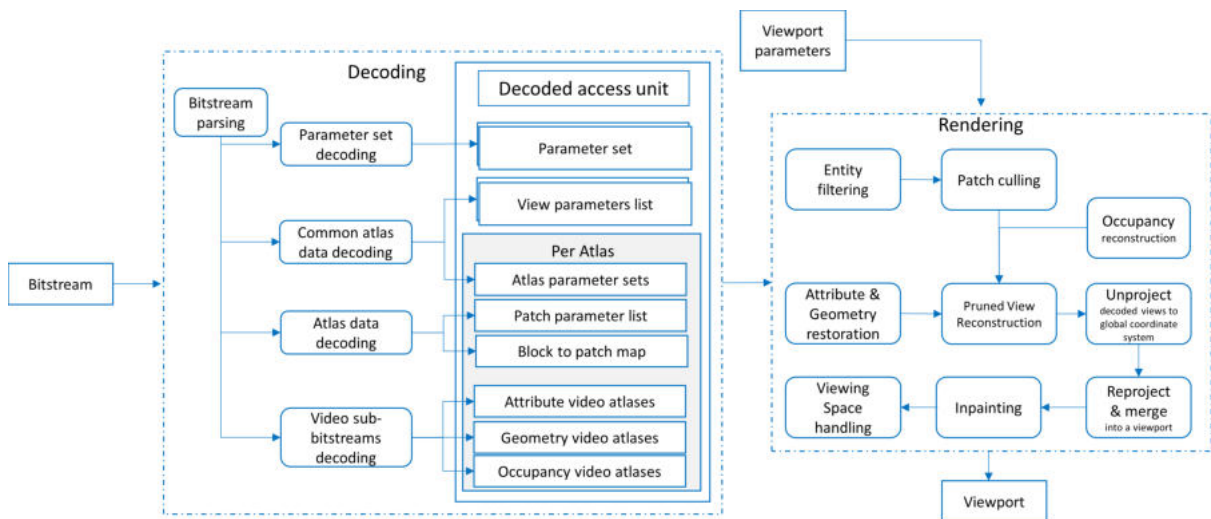


Figure 2.12 – Block diagram of the TMIV decoder and renderer in ESDE mode. The corresponding DSDE mode system is presented in Chapter 6 [33].

they originally belong to, see Fig. 2.13. In case of *Decoder-Side Depth Estimation*, a depth

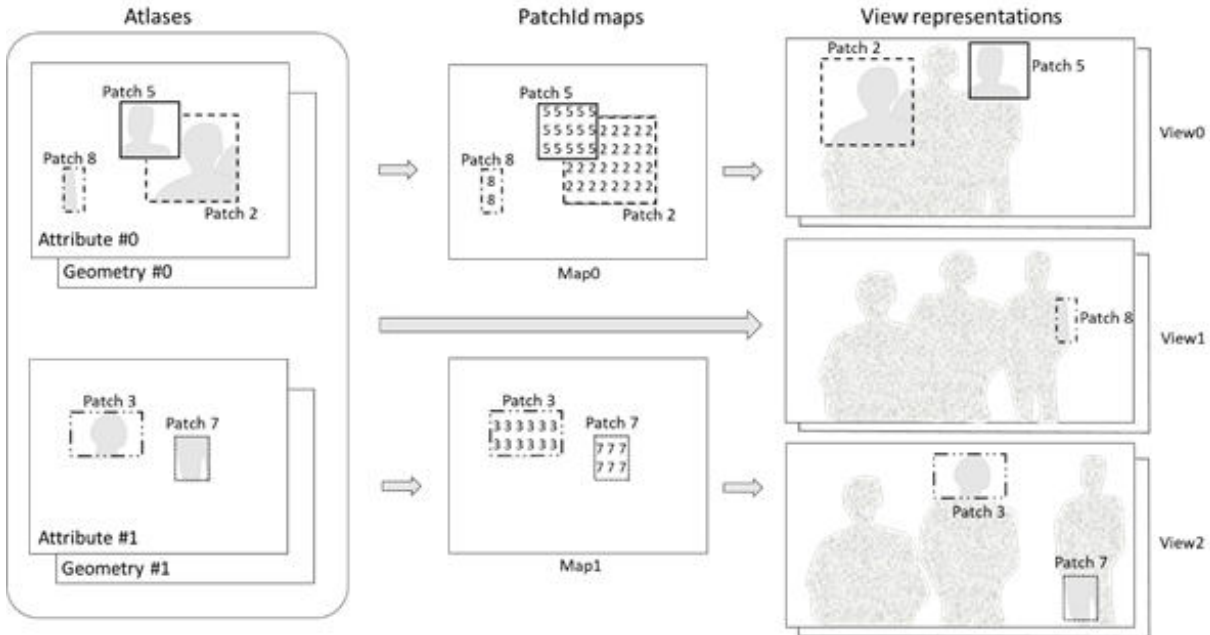


Figure 2.13 – Reconstruction of the pruned views.

estimation module is evoked (See Chapter 6).

Two alternative synthesizers are implemented in TMIV : an approach based on RVS [37] and the View Weighing Synthesizer (VWS). RVS uses similar and typical DIBR-based techniques as VVS. However, it is less robust to erroneous depth maps. VWS follows two steps: first, a visibility map is computed by warping and merging the geometry information towards the target view. This process is already considering the weight of each view. Next, the texture is warped considering the consistency with the visibility map. The weight is based on the distance between each reconstructed view and the target view. The decoded inpainted background view is only supported by VWS and used to recover occluded information, if possible. The remaining holes are filled through inpainting. Each hole is replaced by the distance-weighted average of the closest available left and right pixel.

## 2.3 The Common Test Conditions

The previous section described the latest version of TMIV at the time of writing this chapter. The Common Test Conditions (CTC) [38] described in the following, however,



Sequence	Type	#Views (Setup)	Resolution (frame rate)
Painter	Natural	16 (4x4 planar)	2048x1088 (30)
UnicornA	Natural	25 (5x5 planar)	1920x1080 (still image)
UnicornB	Natural	15 (5x3 planar)	1920x1080 (still image)
Shaman	CGI	25 (5x5 planar)	1920x1080 (30)
Kitchen	CGI	25 (5x5 planar)	1920x1080 (30)
Dancing	CGI	42 (14x3 planar)	1920x1080 (30)
Chef2	Natural	20 (5x4 planar)	1920x1048 (30)
Frog	Natural	15 (15x1 linear)	1920x1080 (30)
Fencing	Natural	10 (10x1 linear)	1920x1080 (25)

Table 2.2 – Test sequences of the CTC for MPEG Immersive Video.

are based on the very early stages of the standardization activity. The 6DoF and 3DoF+ exploration experiments were separated early 2019 and have been merged at a later development stage of MIV. The CTC used in this thesis [38] is based on the first version released in 2019, at the time of the beginning of this thesis. It will be used in chapters 3, 4 and 5 and will be less strictly followed in chapters 7 and 8. Later CTCs, used in more progressed stages of the MIV, included additional objective metrics in order to evaluate proposals. We have applied them all experiments retrospectively. Novel sequences are used in chapter 8. The CTC defines nine test sequences covering a wide spectrum of challenges. The number of cameras, their positioning as well as scene complexity varies for each sequence. A summary of the test sequences is provided in Tab. 2.2. Depth maps are provided originating from different sources. In case of CGI, the depth maps are typically computer generated as well. In case of natural content, the depth maps may be estimated using software not publicly disclosed. They may further be refined by additional algorithms. As we are required to estimate the depth maps ourselves for all sequences, we use DERS8, which is the reference software for depth estimation in the MPEG-I Visual group. Configurations for all sequences are provided, which we adopt for all our experiments without any modification. However, DERS8 provides inconclusive results for the Fencing sequence and has been since then removed from the mandatory sequences. Consequently, we exclude it from our experiments as well. The MV-HEVC anchor block diagram is shown in Fig. 2.14. The 6DoF anchor is generated using MV-HEVC version 13.0. A modification is applied to enable the coding for more than 16 views. A subset of views is pre-selected, however, for most sequences, all views are coded. The coding order for all sequences is based on the serpentine scan, as indicated in Fig. 2.15. Textures and depth maps are encoded separately using predefined quantization parameters (QP) for textures ( $QP_T$ )



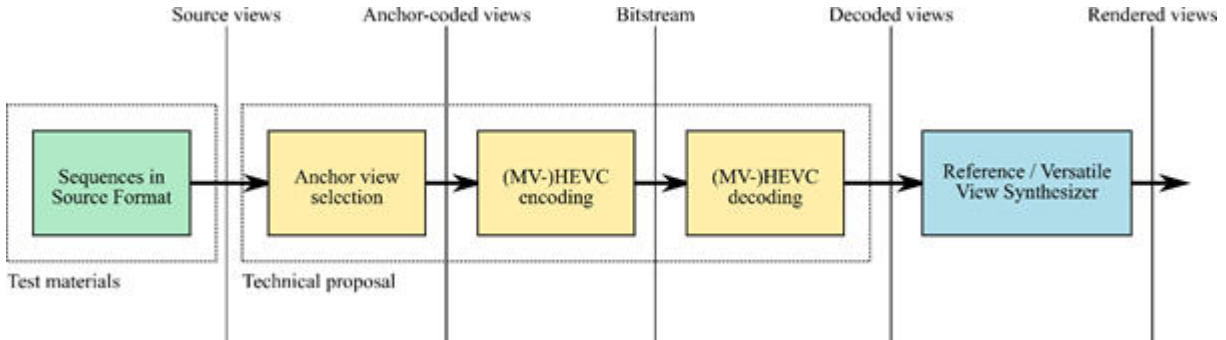


Figure 2.14 – The MV-HEVC anchor.

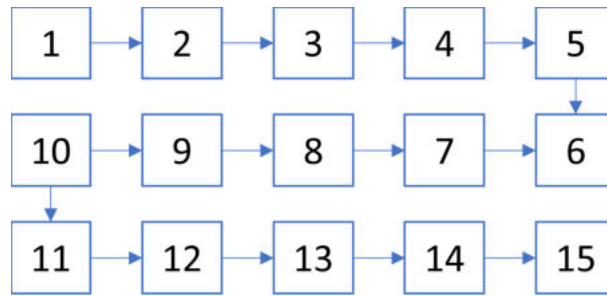


Figure 2.15 – "Serpentine" scan coding order for MV-HEVC based coding of MVD.

and depth maps ( $QP_D$ ). Five  $QP_T/QP_D$ -pairs are tested for texture and depth coding. Codecs are compared using the Bjøntegaard delta (BD) [39]. The four upper QP-pairs define the medium-bitrate range and the four lower QP-pairs define the low-bitrate range, which are used for BD-Rate computation. Transmitted views reside at source positions. For subjective evaluation, views at intermediate positions are synthesized following a pose trace. For objective evaluation, views at source position are synthesized assuming the texture and depth map at the currently synthesized source position do not exist. In this way, the synthesized view can be directly compared to the uncompressed texture at the source position. View Synthesis is performed by VVS. The evaluation procedure is shown in Fig. 2.16 In contrast to classical 2D video coding, two types of BD-Rates are reported, which consider the Y-PSNR of either decoded textures (video BD-Rate) or of synthesized textures (synth BD-Rate) together with the total bitrate. Beyond the CTC, MPEG-I

Table 2.3 – Quantization parameter pairs for the MV-HEVC anchor.

	QP1	QP2	QP3	QP4	QP5
$QP_T$	25	30	35	40	45
$QP_D$	34	39	42	45	48



Figure 2.16 – Objective evaluation: allowed reference views (blue) to be used to synthesize a target view (red) at source position.

Visual further studies the combination of additional metrics with the BD-Rate by replacing the PSNR with Video Multimethod Assessment Fusion (VMAF) [40] and Multi-Scale Structural Similarity (MS-SSIM) [41]. This is motivated by the fact that the evaluation of view synthesis performance can be misleading if merely PSNR is considered [42]. While a common agreement of multiple metrics may confirm an actual improvement, subjective evaluation of synthesized video segments is still the proper way of evaluating changes in the design. We additionally report the LPIPS metric [43], which evaluates the human perception of synthesized views. The overall objective performance of the system is always evaluated considering the bitrate together with quality of synthesized textures compared to the original source textures. According to the conventions of the CTC, negative BD-Rate values indicate an improvement of the proposal compared to the anchor.

### Pixel Rate Constraint

The limitation of immersive video coding due to the pixel rate constraint has been mentioned already multiple times in previous sections. This constraint [44] has significantly defined the chosen strategies during the MIV development, like pruning and geometry downscaling. A low pixel rate configuration is constrained to 32 Megapixels at 30 frames per second, motivated by decoding capabilities of modern smartphones. A typical test sequence of size 2048x1088 and 16 views requires 64 Megapixels at 30 frames per second and can therefore not be decoded at the low pixel rate constraints. This motivates solutions which allow transmission of less pixels in terms of texture or depth maps. Consequently, the concept is ordered as follows:

1. First, meet the low pixel rate constraint. Going further down is not considered beneficial at this point.
2. Second, minimize bitrate requirements and maximize synthesis performance, *i.e.* optimize synth BD-Rate.

Due to the increased relevance of pixel rate, an approach that performs worse in terms of BD-Rate may be preferred if it manages to meet the low pixel rate constraint.

### **$QP_D$ Allocation**

A common difficulty in the ESDE system is the allocation of  $QP_D$  depending on  $QP_T$ . Typically,  $QP_D$  is derived using

$$QP_D = QP_T + \Delta_{QP}, \quad (2.1)$$

with a fixed  $\Delta_{QP}$  for all sequences. The challenge in allocating  $QP_D$  arises from its dependency on multiple additional aspects besides the current  $QP_T$ : the quality of the given depth maps, the coding method used, and the sequence itself. The allocation problem has been intensively studied in [45]. Recently, the authors of [46] propose to model the relationship between  $QP_T$  and  $QP_D$  using linear regression, assuming high quality depth maps. Yet, deriving sequence-optimum  $QP_D$  remains a challenging task. The  $QP_T/QP_D$ -pairs have been selected based on experience from 3D-HEVC standardization.

## **2.4 Conclusion**

In this chapter, the incentives and motivation of the MPEG Immersive Video standard have been elaborated, particularly emphasizing the advantages and compromises in video-based immersive video coding. The TMIV 11 encoder, decoder and renderer have been presented. TMIV 11 is the most recent and polished achievement of the MPEG-I Visual group. The Geometry Absent Profile and the Geometry Assistance SEI message have been adopted based on the work conducted in scope of this thesis, which is presented in the subsequent chapters.



# BASIC DECODER SIDE DEPTH ESTIMATION (B-DSDE)

---

This chapter introduces the *Decoder Side Depth Estimation (DSDE)* system and embeds it in the context of the MIV. DSDE is compared to the traditional *Encoder Side Depth Estimation (ESDE)* system. The motivation and advantages of DSDE are discussed, supported by the experimental results based on the common test conditions of MPEG-I Visual. Possible improvements and extensions of DSDE are discussed, which are detailed in later chapters. It is the first time DSDE has been evaluated in the context of immersive video coding, making this chapter the foundation of this thesis.

## 3.1 Motivation for Decoder Side Depth Estimation

The next iteration of an MVD-based immersive video coding has been chosen to be based on a video-based architecture. This decision has been motivated by the lack of industrial success of 3D-HEVC and several technical limitations of this codec design. Consequently, the mistakes of 3D-HEVC should not be repeated. As a video-based solution, existing 2D codecs shall be utilized for MVD coding and the usage of dedicated depth coding tools is not allowed. Consequently, the geometry is coded inefficiently leading to a degradation of view synthesis quality. Simultaneously, pixel rate is a crucial quantity, which has to be minimized. Furthermore, synthesis-degrading techniques like downscaling of the geometry show the urgent need of saving pixel rate. With the goal of achieving 6DoF in mind, many views have to be coded. With these premises in mind, we propose to investigate an alternative architecture for video-based coding of volumetric video.

Fig. 3.1 summarizes three possible systems for DIBR-based volumetric video cod-

B-DSDE was presented at the Picture Coding Symposium 2019

Patrick Garus, Joel Jung, Thomas Maugey and Christine Guillemot, "Bypassing Depth Maps Transmission For Immersive Video Coding," 2019 Picture Coding Symposium (PCS), 2019, pp. 1-5.

ing [47], mainly defined by the positioning of the depth estimator and the view synthesizer in the codec pipeline. The scene is captured and stored as multiview textures  $T$ . They are compressed using a texture encoder. This element is common for all system. The first system (Fig. 3.1a)) requires the depth maps to be estimated (or captured) on the encoder side and to encode them subsequently. Consequently, we refer to this system as *Encoder Side Depth Estimation (ESDE)*. The decoded depth maps  $D^*$  and textures  $T^*$  are used to generate novel views  $S_{ESDE}$  using a synthesizer. This is the system used in all previous immersive video standards, presented in the previous chapters. In a video-based solution, the original depth maps are encoded using the same video codec as for textures. Dziembowski et al. [48] have tested the impact of compressing the textures on the accuracy of depth estimation and showed that it is limited to low bitrates. Their investigation is targeting the application of rendering servers [49] [50] (or edge servers), in which depth estimation is performed on compressed textures to save storage, while synthesized views are transferred to the client [51]. This solution is based on the third architecture shown in Fig. 3.1c), which is significantly different from the design currently considered in MPEG.

In the system shown in Fig. 3.1b), the depth maps are not transmitted and the depth estimator is moved to the decoder side. Consequently, we refer to this system as *Decoder Side Depth Estimation (DSDE)*. Depth maps  $D^+$  are estimated from decoded textures  $T^*$  to synthesize  $S_{DSDE}$ . In contrast to ESDE, little investigation [52] and improvement have been done towards the DSDE system up to this date, because the compression of depth has always been the main concern. In the literature, several reasons are mentioned as for why the DSDE system has not been considered for immersive video [53]:

1. The result of depth estimation varies depending on the method. Consequently, a content provider may not have full control of the view synthesis quality on the consumer's display.
2. Quantization noise and other compression artifacts present in decoded textures make depth estimation more challenging and may harm the view synthesis process.
3. Depth estimation is typically a complex process and may therefore be inappropriate to be performed on the decoder side.

The first argument has only minor relevance because the renderer is purposely not part of the specification of 3D-HEVC and MIV: while this prevents a guarantee of a certain level of quality, it leaves room for improvement as any type of future synthesizer can be used. The second argument is the main concern of this chapter and will be addressed in the

context of the MV-HEVC anchor of the MIV standardization process. The third argument is a valid concern and its mitigation is one of the main goals of this thesis.

The DSDE system has the potential to overcome the disadvantages of video-based solutions, which are generally related to the inefficient compression of the geometry and the pixel rate cost. With the development of video-based solutions, the timing may be right to investigate and improve DSDE as an alternative to ESDE.

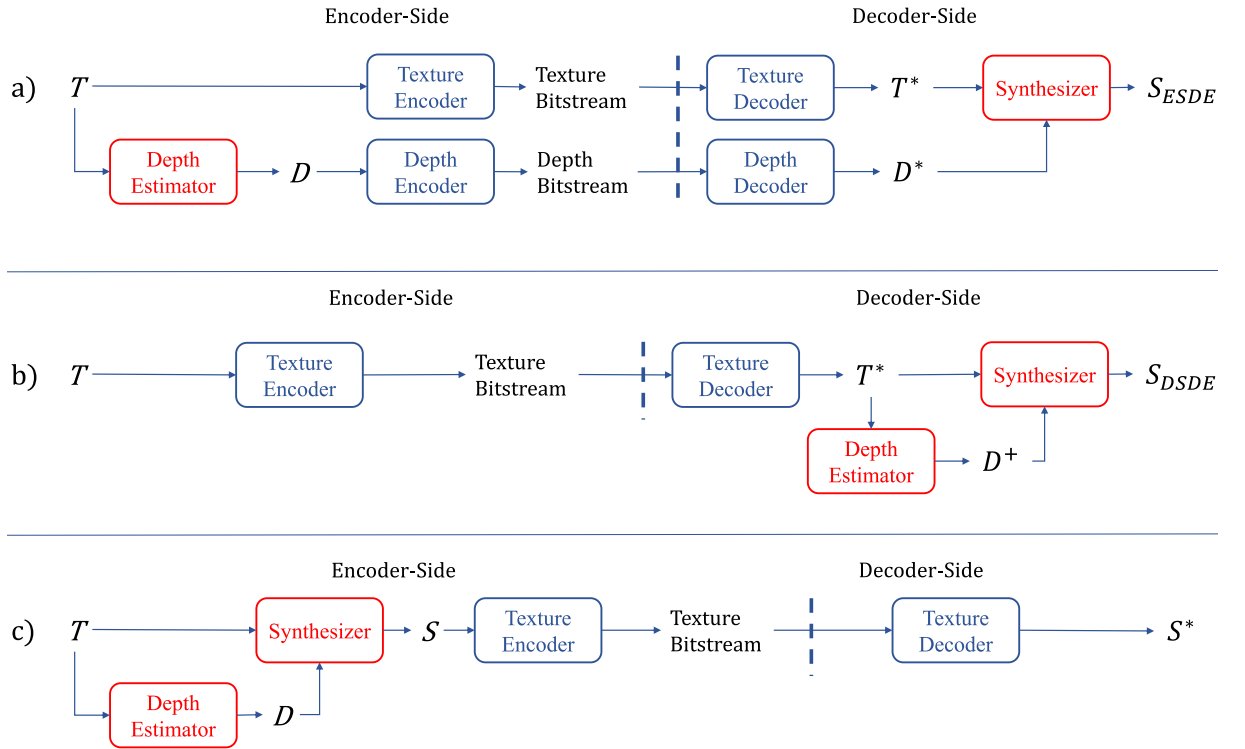


Figure 3.1 – Three immersive video coding architectures with different positioning of the depth estimator and the view synthesizer. The input signals are original multiview textures  $T$  and multiview depth maps  $D$ . The superscript  $*$  indicates decoded signals. a) shows the ESDE system with synthesized views  $S_{ESDE}$ , b) shows the DSDE system with decoder-derived depth  $D^+$  and synthesized views  $S_{DSDE}$  and c) shows a third solution, which directly transmits synthesized views  $S$ .

## 3.2 Experimental Setup to Evaluate B-DSDE

Since all experiments in this thesis are related to DSDE in general, for easier separation, we refer to the unmodified, simplest form of DSDE as Basic-DSDE (B-DSDE), shown in Fig. 3.1. The goal of this chapter is to investigate the coding efficiency of B-DSDE in



comparison to ESDE. Our evaluation procedure is described in chapter 3. In order to perform a fair comparison, the depth maps are estimated from the same source: DERS8. The configuration is unmodified in ESDE as well as B-DSDE, as proposed by the MPEG-I Visual group. View synthesis is performed with VVS, the reference view synthesizer for 6DoF, which has been designed considering compression artifacts in the depth maps. Despite possible bias of DERS8 and VVS towards ESDE, we do not perform any tuning or adaptations. For all other configurations, we strictly follow the CTC. In the following, all visual examples show a view, surrounded by reference views, *i.e.* view  $x1y1$  for 2D setups or  $v1$  for 1D setups.

First, all depth maps are estimated with DERS8 using the uncoded source textures. The results are shown in Fig. 3.2 and are qualitatively high. Especially for natural content, they are equivalent to the depth maps used in the standardization process of MIV. For the evaluation of ESDE, these depth maps are coded with MV-HEVC. In contrast, the B-DSDE depth maps are estimated from decoded textures, compressed with MV-HEVC as well.

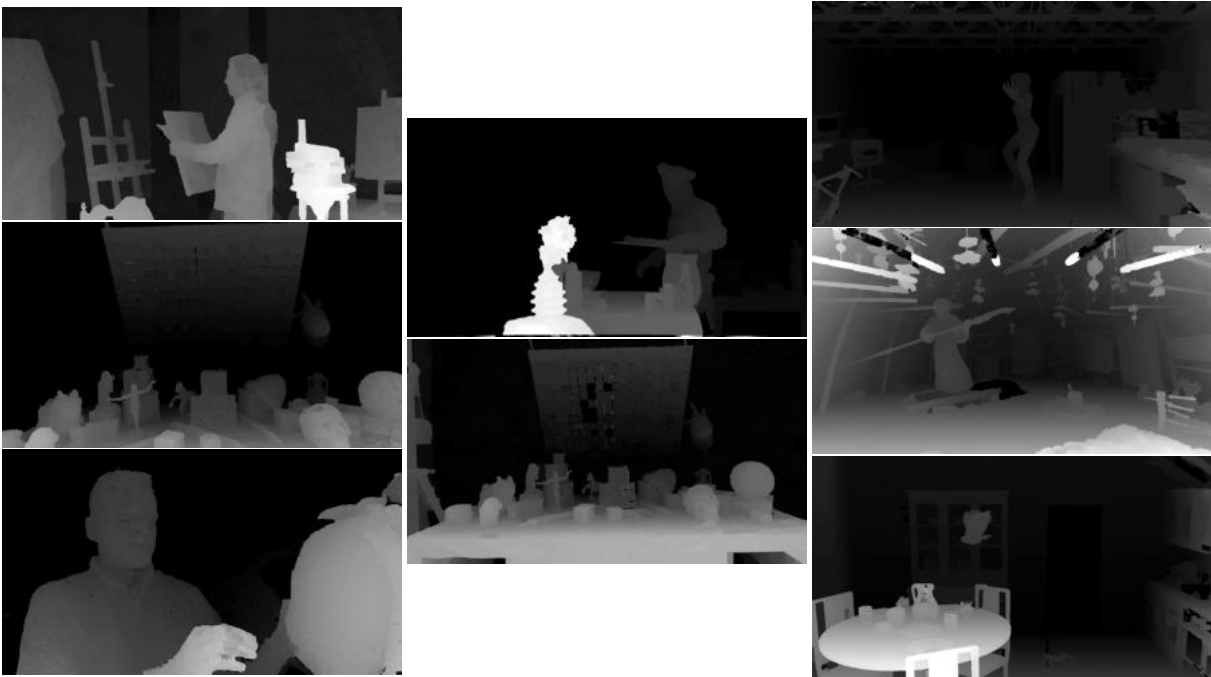


Figure 3.2 – Depth maps estimated by DERS8 using source textures. Top row: Painter, Fencing, Dancing. Mid row: Chef2, UnicornA, Shaman. Bottom row: Frog, UnicornB, Kitchen.

## 3.3 Comparative Analysis of the Performance of ESDE and B-DSDE

### 3.3.1 Depth Quality

First, we investigate the quality of the depth maps. Fig. 3.3 shows the continuous degradation of the depth maps quality over all QPs for the Frog sequence. We observe, that the performance of the depth estimator in case of B-DSDE is barely impacted at high bitrates. This is expected, since the textures coded at high bitrates are still at very high quality and comparable to the uncoded textures. It can even be expected, that the depth estimation performs better in some cases at high bitrates, in contrast to uncoded textures. This may be especially the case for noisy, natural content, since the coding at high bitrates can be seen as a slight low pass filter. Consequently, the mitigated noise levels at high bitrates may serve in favor of the depth estimator. In case of ESDE, the impact of compression is already visible at high bitrates. Block artifacts start to appear, boundaries get blurred and all depth values may be altered, as they are treated like textures. Naturally, these effects become more severe the lower the bitrates. However, B-DSDE seem to be more robust towards depth degradation, if textures coded at lower bitrates are used. Nonetheless, at the lowest bitrate, the degradation becomes apparent. While sharp object boundaries are preserved, a strong "blockiness" starts to appear in the B-DSDE depth maps. The impression is, as if the block artifacts in the textures start to become visible in the estimated depth maps. This can be explained by understanding the depth estimation algorithm described in chapter 1. The smoothing coefficient is modulated by edge maps, computed from the textures. Therefore, the blockiness of the strongly coded textures may be interpreted as a transition between fore- and background increasing the probability for a transition between depth values. Simultaneously, the more structures are lost due to strong texture compression, the more difficult it becomes for the depth estimator to find accurate matches. This observation can provide clues for possible, architectural improvements of B-DSDE.

Another example is shown in Fig. 3.4 for the Kitchen sequence, which is a CGI sequence. In this case, we observe a stronger degradation of the depth maps quality in case of B-DSDE, starting from the high bitrate case. In comparison to natural content, CGI content is far easier to compress, since the textures are more consistent with each other. As we are comparing QPs and not rate points, this outcome may indicate that the "high

bitrate" case of Kitchen may already relate to a lower bitrate scenario of a natural sequence. However, we see more "holes" appearing in the B-DSDE depth maps, indicating difficulties with the selection of the smoothing coefficient. At the same time, the quality of the ESDE depth maps seem more stable over the bitrates. Consequently, we can make the observation that in case of ESDE, the depth maps have the same, initial quality, which is subsequently degraded through compression. While trivial, it is fundamentally different to B-DSDE, where the depth maps quality may change unexpectedly and is mainly determined by the settings and performance of the depth estimator. The change is unexpected, because the depth estimator is operated "blindly", without control, adaptations or any other prior knowledge. In contrast, the impact of compression with decreasing bitrates is more deterministic.

We have seen, that for a comparative analysis, most information can be drawn from the highest and the lowest bitrate scenarios. Consequently, we will limit the provision of visual results on these two cases. Fig. 3.5 and Fig. 3.6 show additional ESDE and B-DSDE depth maps for the remaining sequences. In some cases (e.g. Shaman and Chef2), the depth maps may be normalized differently due to DERS8 decisions, but have no impact on the synthesis performance, as the depth range is adapted. Again, we observe significantly sharper object boundaries in the B-DSDE depth maps with erroneous depth decisions in some cases.

The remaining depth maps in the low bitrate scenario are shown in Fig. 3.7 and Fig. 3.8. Again, strong and different kinds of artifacts in ESDE as well as B-DSDE. The Painter sequence reveals another type of challenge in the B-DSDE case: strong compression of the textures may make a differentiation between fore- and background more challenging (especially if both have a similar color). The back of the painter is therefore partially merged with the background.

We can conclude, that both systems, ESDE as well as B-DSDE introduce degradations in the depth maps. In the high bitrate scenario, the structural properties of the depth maps of the B-DSDE system are very similar to the uncoded depth maps. In case of ESDE however, despite the high bitrate, the compression artifacts become apparent in the depth maps. In the low bitrate scenario, the structural properties of the depth maps can get strongly degraded in case of B-DSDE. While they are more retained in ESDE, because they were originally estimated from uncoded textures, the strong impact of compression may have a more severe impact on the quality of the synthesized view. Finally, the rendering quality and the compromise in invested bitrate are more important in scope of a

coding system.

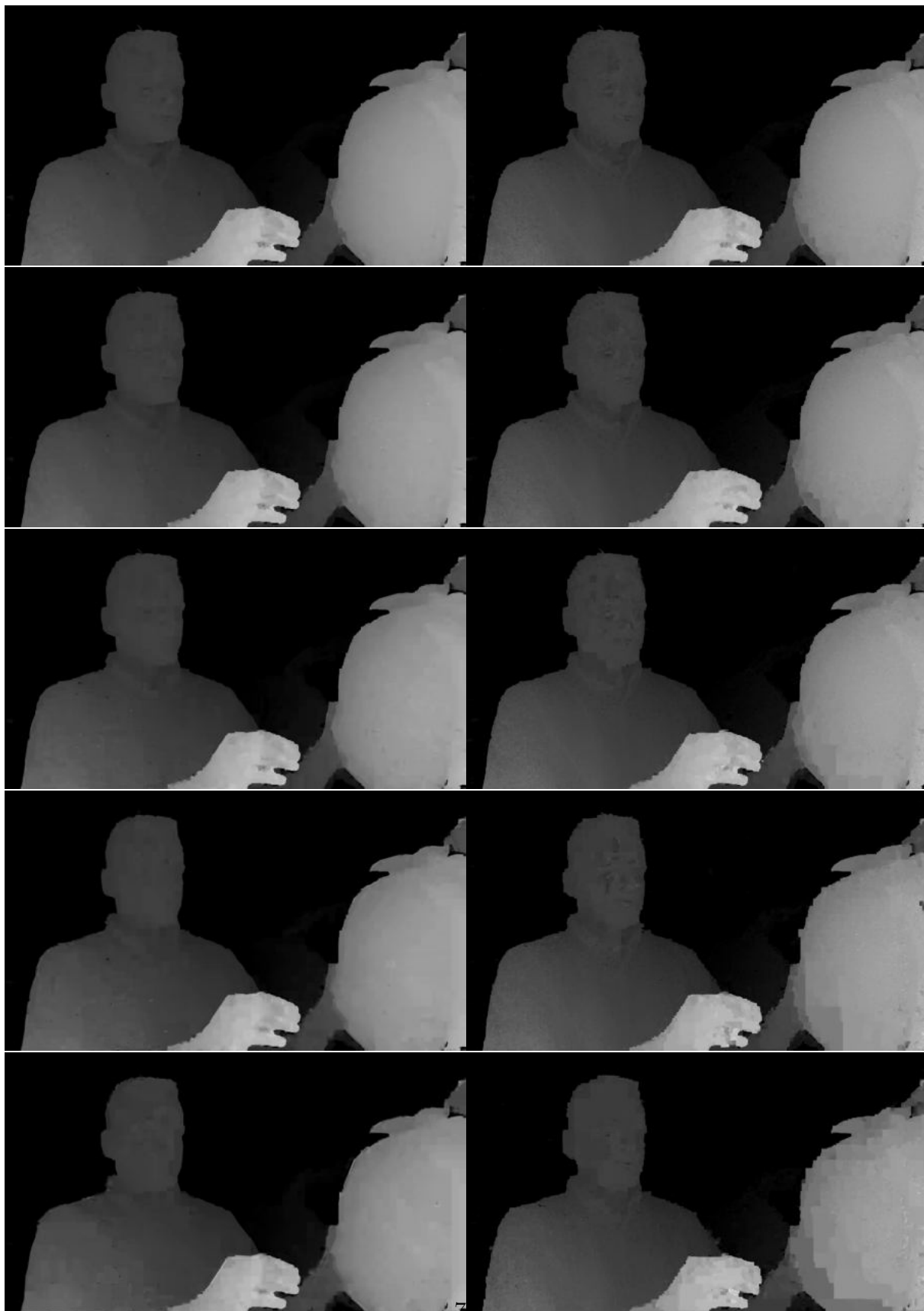


Figure 3.3 – Evolution of depth map quality for ESDE (left) and DSDE (right) from high bitrate (top) to low bitrate (bottom) for natural content (Frog sequence).

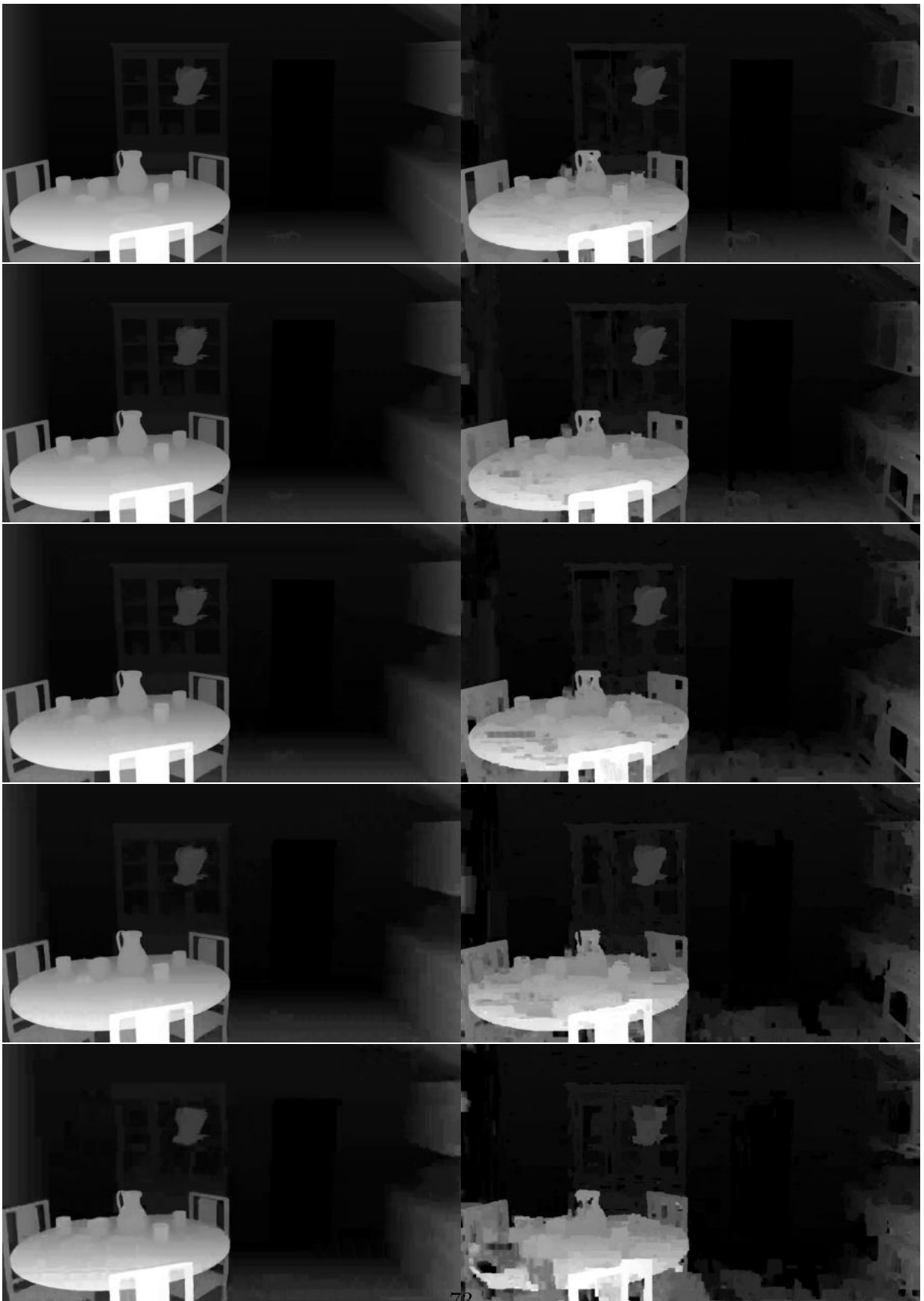


Figure 3.4 – Evolution of depth map quality for ESDE (left) and DSDE (right) from high bitrate (top) to low bitrate (bottom) for CGI content (Kitchen sequence).



Figure 3.5 – ESDE vs DSDE Depth maps quality at high bitrate (1/2).

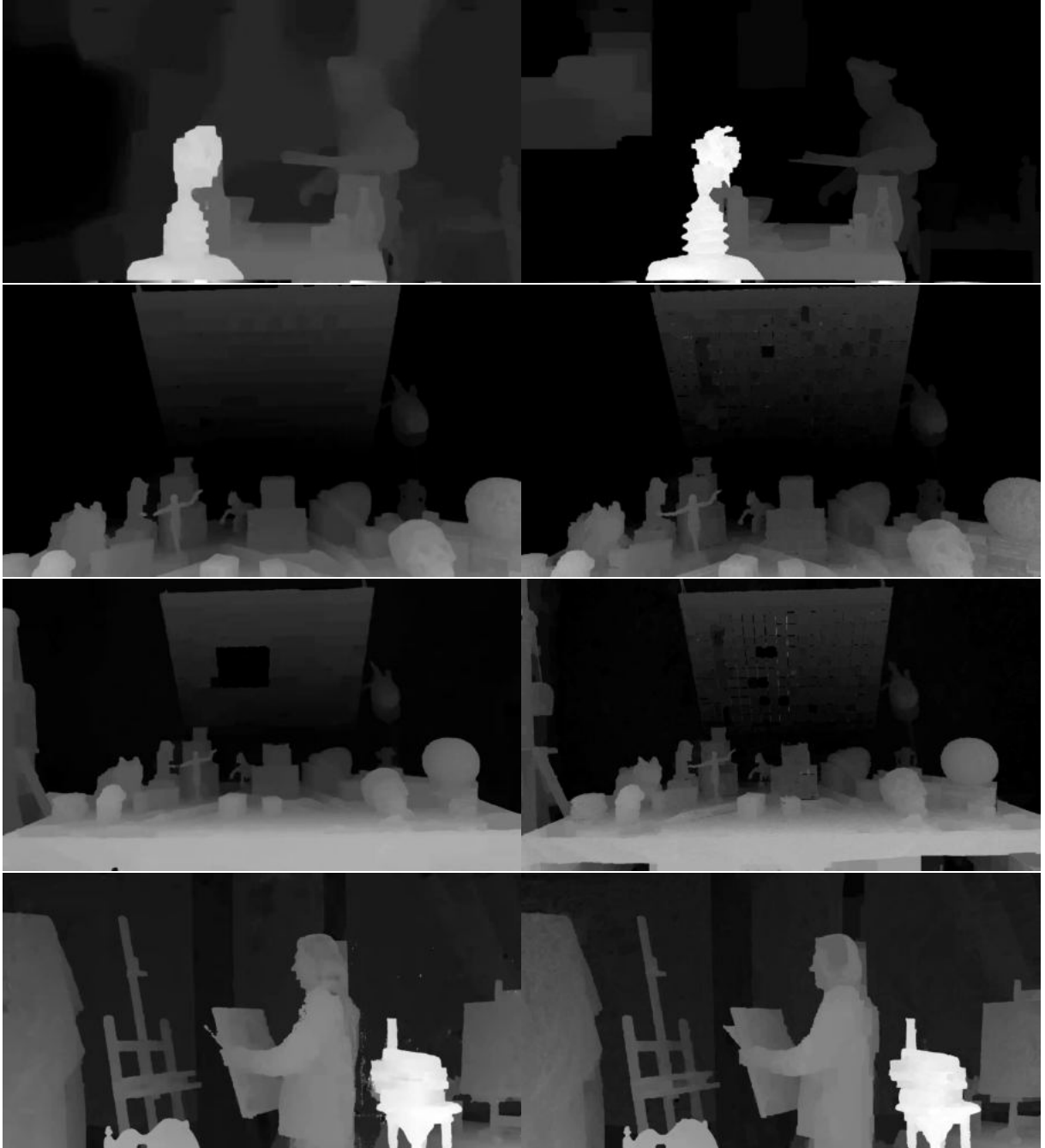


Figure 3.6 – ESDE vs DSDE Depth maps quality at high bitrate (2/2).





Figure 3.7 – ESDE vs DSDE depth maps quality at low bitrate (1/2).



Figure 3.8 – ESDE vs DSDE depth maps quality at low bitrate (2/2).

### 3.3.2 Synthesis Quality

The view synthesis quality is shown in Fig. 3.9 and Fig. 3.10 in the high bitrate case. Cropped areas for all sequences are shown. Even at high bitrates, double-contouring and ghosting artifacts are visible in many cases in the ESDE system. Examples are the oven in the Kitchen sequence, the tools, the roof and the Sintel character in the Dancing sequence, the knife and the pots in the Shaman sequence and the sculpture and the cook in the Chef2 sequence. In contrast, the B-DSDE system provides significantly less blurry, however in some cases more "noisy" results, especially well visible on the shirt in the Frog sequence and the roof of the Dancing sequence.

The low bitrate scenario is shown in Fig. 3.11 and Fig. 3.12. The degradation of many objects seem more severe in the ESDE in comparison to the B-DSDE system. Several structures become completely unrecognizable. Examples are the red button in the Kitchen sequence, the fingers of Sintel and the bulb in the kitchen sequence, the knife in the Shaman sequence. While more sharp and more structures are retained, strong noise is noticeable especially in the roof of the Dancing sequence.



Figure 3.9 – ESDE vs DSDE view synthesis quality at high bitrate (1/2).



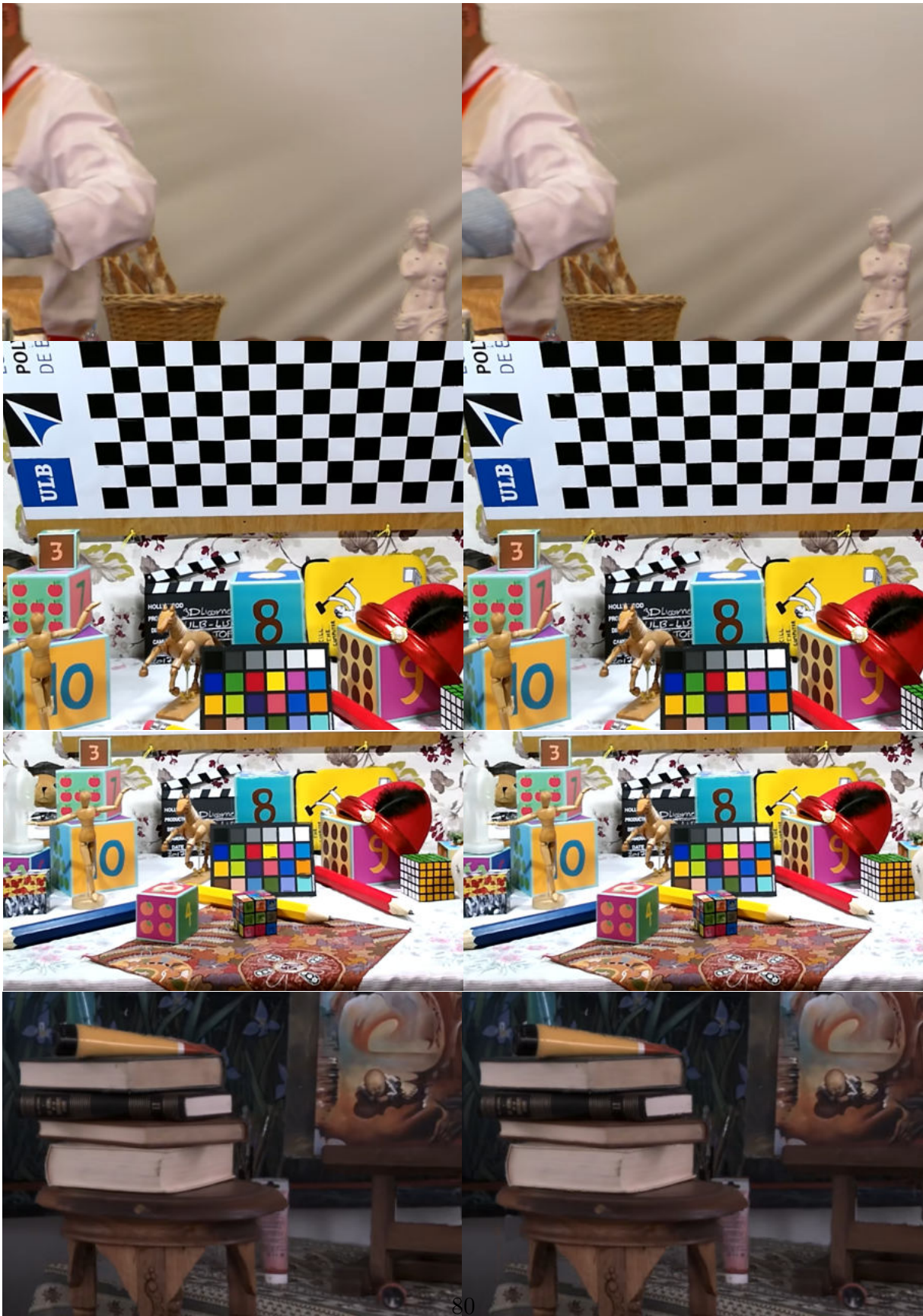


Figure 3.10 – ESDE vs DSDE view synthesis quality at high bitrate (2/2).



Figure 3.11 – ESDE vs DSDE view synthesis quality at low bitrate (1/2).





Figure 3.12 – ESDE vs DSDE view synthesis quality at low bitrate (2/2).

Tab. 3.1 presents the synthesis PSNR and the LPIPS metrics for ESDE and B-DSDE. In terms of PSNR, ESDE is marginally better for Painter, UnicornA and UnicornB in the high bitrates and UnicornA for the low bitrate. For all other sequences, we compute significant PSNR improvements in the B-DSDE system compared to ESDE. On average, B-DSDE performs 0.45 dB better than ESDE at high and 0.34 dB better at low bitrates. The best PSNR gain of B-DSDE is seen in the Dancing sequence with 1.82 dB at high and 1.09 dB at low bitrates. The LPIPS metric is mostly in line with the PSNR indicating a better view synthesis performance in high as well as low bitrates. The metrics confirm the better perceptual quality of B-DSDE over ESDE in all bitrates.

### 3.3.3 Coding Efficiency

Naturally, the bitrate has to be taken into account when evaluating coding systems. Tab. 3.2 summarizes several BD-Rate values computed with different metrics. In B-DSDE, the depth maps coding is entirely omitted. The bitrate savings are visible in the *video* BD-Rate metric, which uses decoded texture PSNR, instead of view synthesis PSNR. Since the decoded textures are equivalent in ESDE as well as B-DSDE, this metric solely reflects the bitrate savings. While it is apparent that B-DSDE saves more bitrate than ESDE, it is still remarkable that the bitrate can be quite significant in a video-based solution: 36%, 32.9% and 27.8% bitrate savings at high bitrates and 39.2%, 39.7% and 31.2% at lower bitrates for Painter, Shaman and Chef2 respectively. On average, 17.4% and 21.0% bitrate are saved at high and low bitrange ranges respectively. This indicates, that ESDE becomes less efficient at lower bitrate ranges.

More importantly, the synthesis PSNR BD-Rate, which indicates the performance of all aspects of the system, shows that B-DSDE outperforms ESDE significantly. If the synth PSNR BD-Rate indicate bigger gains than the video PSNR BD-Rate, it can be concluded that the PSNR of B-DSDE is higher than for ESDE, which is in line with Tab. 3.1. Remarkable coding gains are shown for the Dancing sequence ( $-72.4\%$ ) at high bitrates, while the results are less conclusive for the two still image sequences UnicornA and UnicornB. On average, B-DSDE outperforms ESDE by  $-39\%$  at high and  $-36.3\%$  at low bitrates respectively. In terms of synthesis MS-SSIM BD-Rate, the biggest improvement is measured for the Shaman sequence for high ( $-72.4\%$ ) and low ( $-70.1\%$ ) bitrates. The average synthesis MS-SSIM BD-Rate gain of B-DSDE is  $-33.4\%$  for high and  $-31.3\%$  for low bitrates. The VMAF confirms the previous metrics with the best performance gain measured for the Dancing sequence ( $-74.5\%$ ) at high and for the Shaman



Table 3.1 – Average synthesis PSNR and LPIPS of B-DSDE and ESDE. The bold values indicate the better solution.

Config	Sequence	synth PSNR [dB]		LPIPS	
		ESDE	B-DSDE	ESDE	B-DSDE
Medium Bitrate	Painter	<b>34.38</b>	34.23	0.220	<b>0.218</b>
	UnicornA	<b>29.58</b>	29.38	<b>0.059</b>	0.063
	UnicornB	<b>29.88</b>	29.84	<b>0.058</b>	0.059
	Shaman	33.61	<b>34.21</b>	0.255	<b>0.243</b>
	Kitchen	30.55	<b>31.05</b>	0.181	<b>0.176</b>
	Dancing	28.15	<b>29.97</b>	0.218	<b>0.191</b>
	Chef2	31.50	<b>31.91</b>	<b>0.260</b>	<b>0.260</b>
	Frog	26.95	<b>27.59</b>	0.229	<b>0.208</b>
	Average	30.57	<b>31.02</b>	0.185	<b>0.177</b>
Low Bitrate	Painter	32.96	32.75	0.334	<b>0.331</b>
	UnicornA	<b>28.21</b>	28.20	0.123	<b>0.122</b>
	UnicornB	28.56	<b>28.65</b>	0.123	<b>0.121</b>
	Shaman	32.46	<b>32.75</b>	0.430	<b>0.418</b>
	Kitchen	29.42	<b>29.79</b>	0.324	<b>0.317</b>
	Dancing	27.02	<b>28.11</b>	0.386	<b>0.374</b>
	Chef2	30.82	<b>31.29</b>	0.323	<b>0.322</b>
	Frog	25.99	<b>26.59</b>	0.363	<b>0.337</b>
	Average	29.43	<b>29.77</b>	0.301	<b>0.293</b>

sequence ( $-60.0$ ) at low bitrates. The average synthesis VMAF BD-Rate gain of B-DSDE is  $-40.8\%$  for high and  $-39.3\%$  for low bitrates.

Table 3.2 – Objective coding performance of B-DSDE compared to ESDE.

Config	Sequence	video	synth PSNR	synth MS-SSIM	synth VMAF
		BD-Rate [%]	BD-Rate [%]	BD-Rate [%]	BD-Rate [%]
Medium Bitrate	Painter	-36.0	-31.7	-30.9	-35.4
	UnicornA	-4.7	6.0	-5.9	5.5
	UnicornB	-5.6	-3.6	-7.8	-4.4
	Shaman	-32.9	<b>-55.2</b>	-72.4	-62.7
	Kitchen	-18.2	<b>-39.2</b>	-24.3	-43.3
	Dancing	-5.3	<b>-72.4</b>	-48.7	-74.5
	Chef2	-27.8	<b>-58.4</b>	-40.7	-53.3
	Frog	-11.4	<b>-57.8</b>	-36.2	-53.3
Average	-17.4	<b>-39.0</b>	-33.4	-40.8	
Low Bitrate	Painter	-39.2	-35.1	-34.8	-39.8
	UnicornA	-5.4	<b>-6.7</b>	-8.3	-5.0
	UnicornB	-6.7	<b>-13.8</b>	-9.2	-14.3
	Shaman	-39.7	<b>-50.0</b>	-70.1	-60.0
	Kitchen	-22.3	<b>-36.0</b>	-26.0	-41.8
	Dancing	-8.0	<b>-51.4</b>	-32.9	-59.6
	Chef2	-31.2	<b>-54.3</b>	-40.9	-52.8
	Frog	-15.0	<b>-42.8</b>	-28.5	-41.5
Average	-21.0	<b>-36.3</b>	-31.3	-39.3	

One of the most important advantages of the B-DSDE system is the saving of pixel rate. In our experimental setup, **50%** pixel rate is saved compared to ESDE, while simultaneously improving view synthesis performance and coding efficiency. In other words, B-DSDE allows for coding of double the number of cameras, double the framerate or double the width/height.

### Depth Estimation Complexity

Nevertheless, when it comes to comparing codecs or coding systems with each other, the decoder complexity is an important constraint and must always be considered. The runtimes of DERS8 are shown in Tab. 3.3. With an average runtime of around 1400s, DERS8 is far beyond any reasonable complexity for a client-side renderer. However, DERS8 has never been optimized for runtime and the software mainly used for research. Consequently, these runtimes are given for completeness and not meant to be used to draw conclusions. Significantly faster software exist, especially in the context of GPU-driven machine learning and FPGA-implementations. While a hardware-accelerated solution may

Table 3.3 – Average runtime per frame and view using the unmodified DERS8.

Sequence	DERS8 [s]
Painter	698.3
UnicornA	291.4
UnicornB	270.5
Shaman	3933.9
Kitchen	1051.5
Dancing	1195.0
Chef2	2565.1
Frog	1107.3
<b>Average</b>	<b>1389.1</b>

be necessary for a realistic solution, it does not prevent us to improve the B-DSDE system with the tools provided by the MPEG-I Visual group. Particularly, if these improvements are system-related and reasonable generalized.

### 3.4 Opportunities in the B-DSDE system

Fig. 3.13 indicates several aspects of the B-DSDE system, which can be addressed. Four scopes are described:

1. Scope A: modifications of the texture decoder or the texture bitstream may apply adaptations of the texture coding system to the peculiar characteristics of the MIV. For example, adaptations to the atlas format. However, the MIV is supposed to be agnostic to the used 2D codec and the presence of such a codec cannot be expected. Instead, the information coded into the texture bitstream could be exploited to improve the depth estimator in its task. Possible information could be partitioning and local bitrate cost, which could imply complexity of the textures in the corresponding block. Such an information could be used to adapt the smoothing coefficient of the depth estimator. Another possibility is the provision of motion information, which could be used to enhance the depth estimation temporally. This scope is investigated in Chapter 7: low complexity decoder side depth estimation (LC-DSDE).
2. Scope B: modifications of the TMIV decoder or the MIV Bitstream may refer to the inclusion of relevant side information, which can be used by the depth estimator to improve the depth estimation performance. This scope is investigated in Chapter

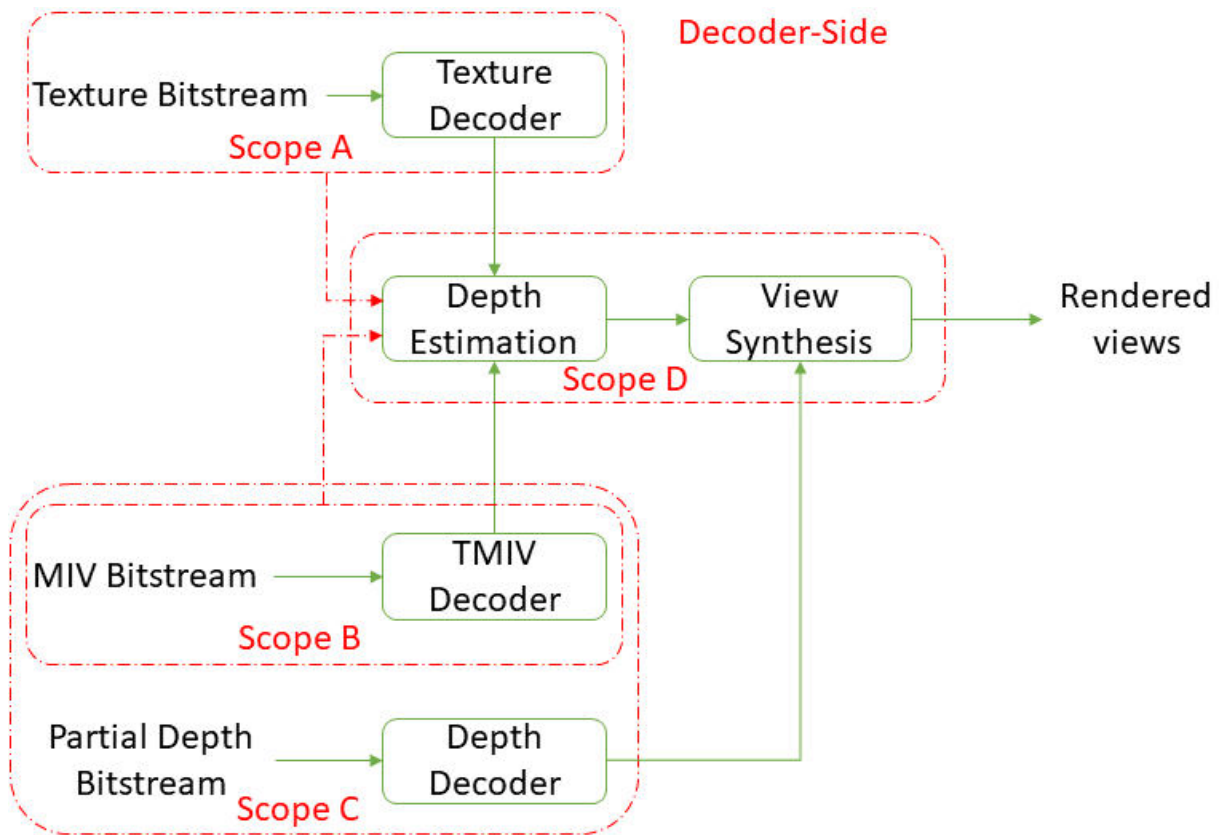


Figure 3.13 – Extensions to B-DSDE.

- 5: feature-driven decoder side depth estimation (FD-DSDE).
3. Scope C: under the assumption that a depth decoder is present, a partial depth bitstream could be provided. Depth maps of a single view could be provided (possibly lossless) to be used as a trustful reference for the estimation of depth maps of all other views. Another possibility is the transmission of partial depth information per view, including only the depth that could not be appropriately recovered at the decoder-side. This scope is investigated in the subsequent chapter 4: hybrid decoder side depth estimation (H-DSDE).
4. Scope D: the depth estimator is directly followed by the view synthesizer in B-DSDE. Therefore, this system is much closer to typical rendering systems developed in the computer graphics community, where a coding and transmission module is not considered between depth estimation and view synthesis. Consequently, B-DSDE opens opportunities for joint operations of these two tools. For example, in the context of machine learning, a CNN-based depth estimator is commonly trained by back-propagating the view synthesis error of a subsequent CNN-based synthesizer. Depth estimation and view synthesis can therefore be considered as a single rendering module. MPIs are one example, where a geometry analysis optimized for  $\alpha$ -blending based view synthesis is performed. This scope is investigated in chapter 8: decoder side multi plane images (DSMPI).

### 3.5 Conclusion

In this chapter, B-DSDE has been presented and analysed comparatively with the ESDE equivalent defined in the CTC of MIV. Despite the claims found in literature, this system provides significantly better coding efficiency with similar or better subjective synthesis quality. At the same time, this system is very promising given the pixel rate constraint, which mainly define the tools utilized in the upcoming MIV standard. It therefore allows to encode twice the number of cameras, making it more suitable to achieve true 6DoF. Furthermore, these additional textures are consequently coded in an appropriate manner using deployed 2D codecs. Finally, this system potentially simplifies the encoder optimization, since a difficult search for the optimal  $QP_D/QP_T$  is no longer required.

In B-DSDE, this system may not guarantee a certain depth maps quality, since the depth estimator is not intended to be part of the standard. However, we do not see this as an

obstacle, because the same holds for the synthesizer. A content provider can therefore not guarantee a certain quality in any case. Nevertheless, our proposals presented in the subsequent chapters will step-by-step establish a link between the depth estimator and the encoder-side.

The final and only concern of this system is the complexity of the depth estimator. As a consequence, extensions and improvements to B-DSDE have to take this matter into consideration.



# HYBRID DECODER-SIDE DEPTH ESTIMATION (H-DSDE)

---

This chapter introduces the hybrid DSDE (H-DSDE) system as an extension of the B-DSDE system. In H-DSDE, depth coding is partially allowed using a 2D codec. Such a hybrid system provides more opportunities to the encoder optimization to make the local decision if a depth block should be coded or instead be estimated at the decoder side.

## 4.1 Motivation for a hybrid DSDE solution

B-DSDE has shown significant quality improvements over ESDE. However, depth estimation is performed completely uncontrolled and the encoder has no choice but to accept the B-DSDE estimated depth maps as they are. From the observation of the B-DSDE depth maps quality we have concluded that in some cases the structural properties cannot be retrieved, due to the distortions in the texture views. Therefore it is still possible that *local depth map* transmission is more beneficial than B-DSDE, if the encoder has the possibility to carefully select the areas where a depth map is transmitted. To test this hypothesis, we apply a Rate-Distortion (RD) based decision between DSDE and ESDE at the block level (Coding Unit, or CU) during HM-encoding. This H-DSDE approach aims at transmitting only the depth CUs which are improving the RD criterion. Consequently, there are potentially two different depth estimation processes at the encoder side: a first one to produce the original depth maps (as in ESDE), and a second one which aims at simulating the depth estimation process at the decoder side. Such an approach could merge the benefits of both systems and further improve the objective performance.

H-DSDE was presented in the IEEE Transactions on Circuits and Systems for Video Technology  
 Patrick Garus, Felix Henry, Joel Jung, Thomas Maugey and Christine Guillemot, "Immersive Video Coding: Should Geometry Information Be Transmitted as Depth Maps?," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 5, pp. 3250-3264, May 2022.



## 4.2 Description of the H-DSDE system

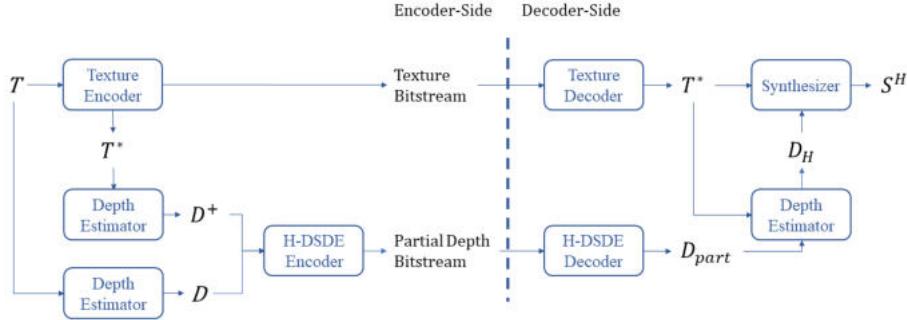


Figure 4.1 – Overview of the proposed H-DSDE system. The input signals are original multiview textures  $T$  and multiview depth maps  $D$ .  $T^*$  are coded textures.  $D^+$  are depth maps estimated from decoded textures.  $D_{part}^*$  are coded partial depth maps and  $D_H$  are depth maps composed of decoded and decoder derived-depth.  $S^H$  are synthesized views.

The H-DSDE system is shown in Fig. 4.1. Texture compression is not modified. We assume textures to be encoded prior to the depth maps. Using the decoded textures, the encoder can estimate the same depth maps as at the decoder side using DERS8. Inputs to the H-DSDE encoder are the original depth  $D$  and DSDE depth  $D^+$ . First, an ESDE CU is processed by the video codec. The bitrate required to compress the depth CU is denoted as  $R_{ESDE}$ . Then, in order to estimate the bitrate required for the DSDE case, an empty (medium grey level) CU is encoded. The goal is to signal the DSDE-decision in the cheapest manner, without disrupting the subsequent compression process. The corresponding bitrate for the DSDE CU is denoted as  $R_{DSDE}$ . In order to model the distortion, we follow the concept of the synthesized view distortion change (SVDC) [54] using VVS to synthesize views at source positions. The computation is illustrated in Fig. 4.3. We construct a complete depth map in order to perform view synthesis. All decisions prior to the current CU being processed are fixed during synthesis and are either ESDE- or DSDE-type depth. All subsequent CUs are replaced by the original depth. Therefore, three depth maps have to be evaluated, in which the current CU is either DSDE, ESDE or original depth. We synthesize a neighboring target view using VVS and compute the sum of squared differences (SSD) between the synthesized view and the corresponding original view  $T_{Ref}$  for all three cases. Synthesizing a complete view ensures that the full impact of the CU decision is taken into account in the synthesized view. In practice, the complexity of our encoder could be reduced by synthesizing only the portion of the view that is affected by the CU decision, similar to the partial re-rendering concept of

3D-HEVC [55].

Depending on the depth type, we denote the corresponding distortion measures as  $e_{ESDE}$ ,  $e_{DSDE}$  and  $e_{orig}$ . The quantity used in the RD-formulation is the distortion change  $\Delta E_{ESDE} = e_{ESDE} - e_{orig}$  and  $\Delta E_{DSDE} = e_{DSDE} - e_{orig}$  for ESDE and DSDE accordingly. The decision is taken by choosing the minimum between the cost functions:

$$\begin{aligned} C_{ESDE} &= \Delta E_{ESDE} + \gamma \lambda R_{ESDE}, \\ C_{DSDE} &= \Delta E_{DSDE} + \gamma \lambda R_{DSDE}, \end{aligned} \quad (4.1)$$

with the lagrangian multiplier  $\lambda$  and a scaling factor  $\gamma$ .  $\gamma$  adapts the lagrangian multiplier in order to optimize the coding performance using the SVDC. This modification allows to take into account that the distortion is evaluated on the synthesized view, instead of the decoded texture, as it is done in 2D coding. We set  $\gamma = 0.5$  as in [54]. The decision is signalled through a flag per CU.

At the decoder side, the partial depth map  $D_{part}^*$  is decoded. A depth estimator reconstructs the depth CUs which were bypassed using the decoded flag. The resulting depth map  $D_H$  is a composition of ESDE and DSDE depth CUs, that can be used to render novel views. Fig. 4.2 demonstrates the benefit of H-DSDE for the Kitchen sequence. Our



Figure 4.2 – Example for different depth signals in the H-DSDE system. From left to right: source depth ( $D$ ), DSDE depth ( $D^+$ ), hybrid depth ( $D_H$ ), decoded partial depth ( $D_{part}^*$ ).

H-DSDE encoder is based on HM16.6 [56]. Consequently, decisions in other views are not

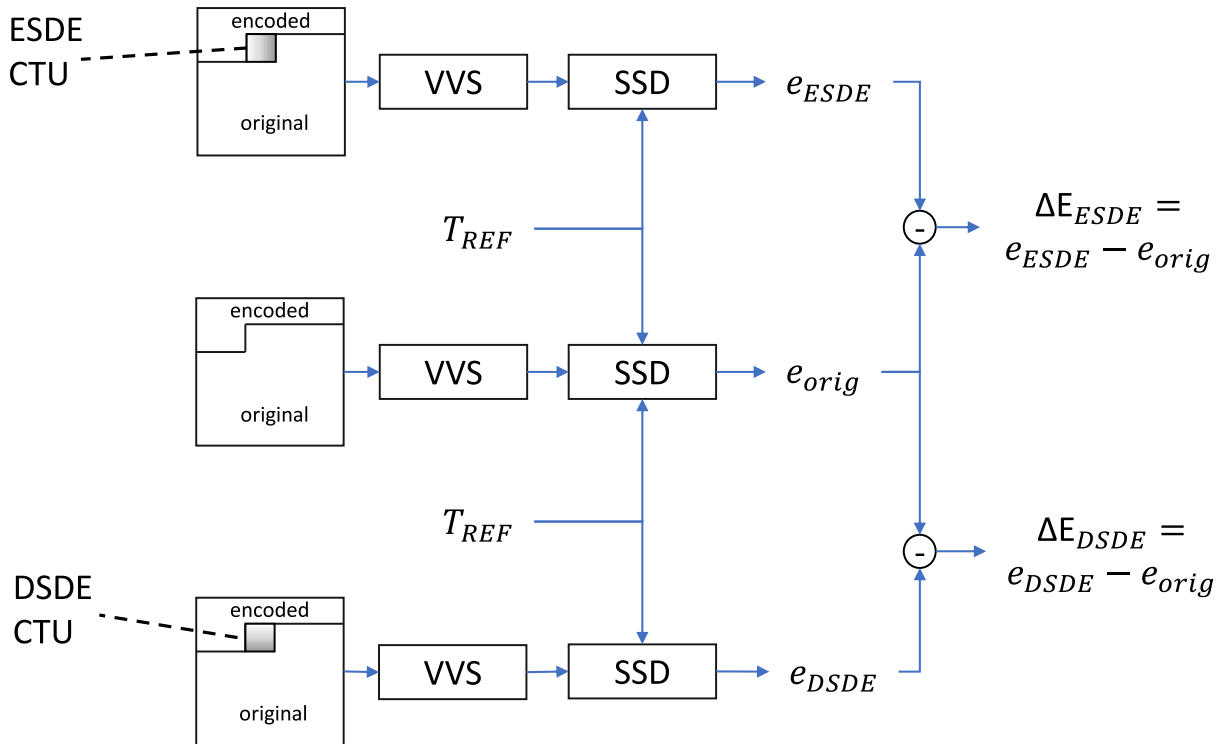


Figure 4.3 – Overview of the SVDC computation. View synthesis is performed by VVS. The currently encoded CU is either ESDE, DSDE or original. The sum of squared differences (SSD) between the synthesized view and the original view  $T_{ref}$  leads to the distortions  $e_{ESDE}$ ,  $e_{DSDE}$  and  $e_{orig}$  for all three cases respectively. The differences  $\Delta E_{ESDE}$  and  $\Delta E_{DSDE}$  are used in the RD cost computation.

taken into account, which is a disadvantage as the anchor is utilizing MV-HEVC. Another limitation is the evaluation during the SVDC calculation, which is not equivalent to the final evaluation of the proposal after finalizing the compression of all views. Fig. 4.4 visualized the situation during H-DSDE coding for two examples, views  $x_0y_0$  and  $x_1y_0$ . For each depth map being coded, the target view for view synthesis is selected based on the serpentine order. The reference views that are used to generate the target view are the same as in the final evaluation, which means target view  $x_1y_0$  is synthesized using the reference views  $x_0y_0$ ,  $x_1y_1$  and  $x_2y_0$ . The depth maps at  $x_1y_1$  and  $x_2y_0$  contain solely uncoded depth blocks, while the view being coded,  $x_0y_0$  contains three types of blocks: ESDE, DSDE and uncoded. The block being currently processed (indicated with ?) is tested for all three cases in order to compute the SVDC. For clarity, the situation is shown for position  $x_1y_0$  as well. At the same time, a coded depth map may be used for up to 4 different target views in the final, objective evaluation. Consequently, the synthesis distortion taken into account during coding does not reflect the synthesis distortion in the final evaluation. In principle, these limitations could have been addressed in several ways. For the decision of the current depth block, the contribution of every close target view could have been considered. At the same time, the coding of a depth map could have been conditioned on the finalization of the previous view. This would allow to take previously coded depth maps into account in the synthesis process during coding of a certain view. However, all of these options would increase the runtime of the simulations by several factors, which is considered unreasonable for the expected benefit.

## 4.3 Experimental results

We follow the common test conditions and compare to the same ESDE results provided in the B-DSDE chapter. Therefore, the performance of H-DSDE can be directly compared to ESDE and B-DSDE.

### 4.3.1 Partial Depth Decisions

First, we investigate the choice between ESDE and DSDE in the depth maps. Fig. 4.5 show the selection of the blocks across all bitrates for the Frog sequence. For the highest bitrate, a preference towards DSDE for object boundaries can be observed. This is expected, because the object boundary has a significant impact on the view synthesis

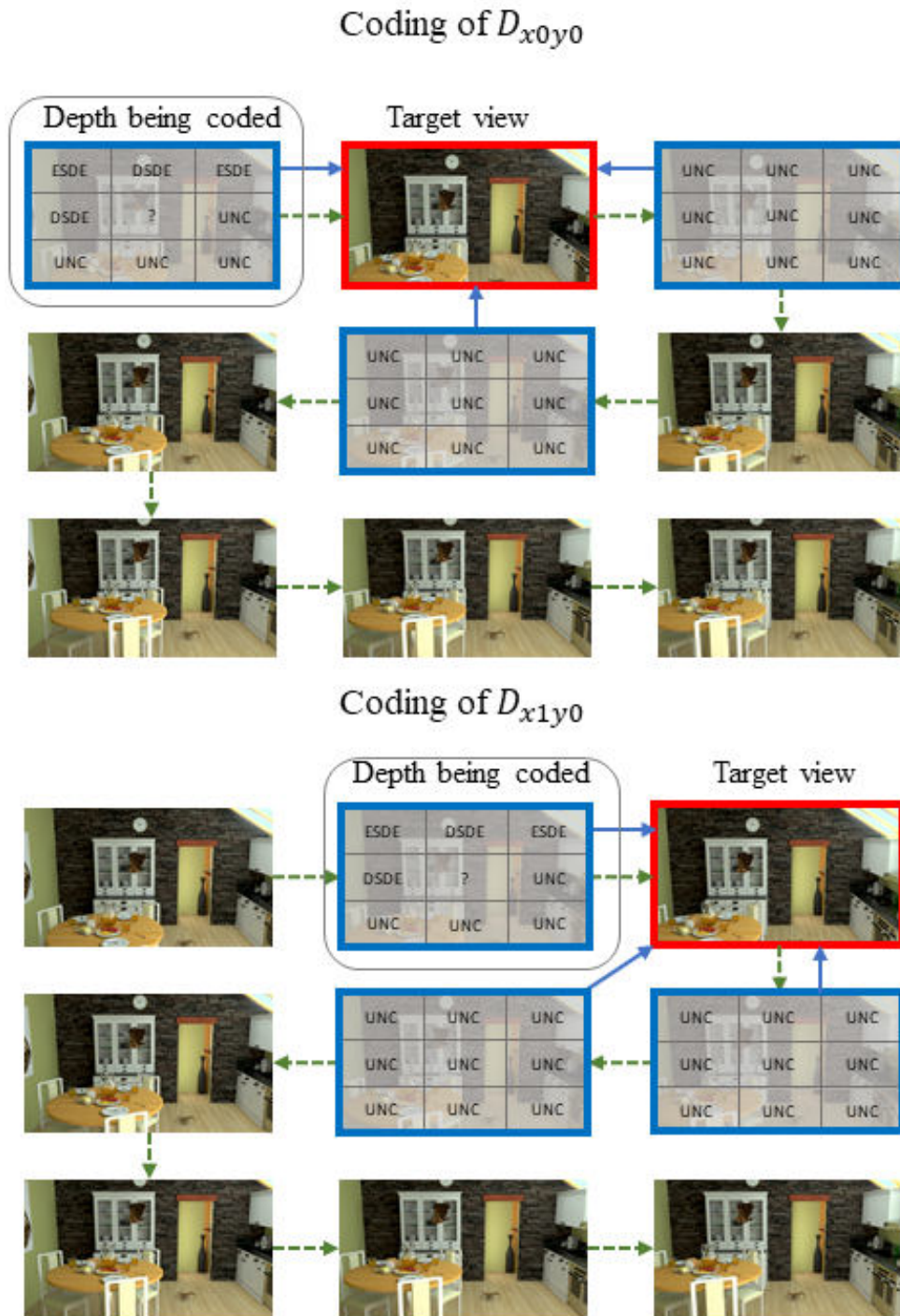


Figure 4.4 – Illustration of the quality evaluation during H-DSDE encoding for two examples. The green, dashed arrows indicate the "coding order", *i.e.* the selection of the target view given the currently coded view. The blue arrows indicate the reference views (blue boundaries) used for view synthesis of the target view (red boundary). In each reference view, the current decision of the coded block is shown. Only the view currently being coded contains either ESDE, DSDE or uncoded blocks, while all other reference depth maps contain solely uncoded blocks.

quality and is already at high bitrates strongly impacted by compression artifacts. This behaviour stays consistent over all bitrates, which implies that the erroneous DSDE-blocks are still less harmful than the corresponding ESDE blocks. With lower bitrates, more and more blocks inside the objects are selected in favor of DSDE. However, when it comes to the interpretation of these depth maps, the decisions are always a compromise between spent bitrate and measured benefit for view synthesis. Sometimes, the additional benefit of quality may not be worth the cost for coding a ESDE depth. Nevertheless, the "cost" of a "grey" DSDE block may actually be surprisingly high, in comparison to smooth ESDE block, which may be cheaper to predict from its surroundings. This situation could be improved by making the selection of the constant value for DSDE blocks dependent on the already coded neighborhood.

Figures 4.6 and 4.7 show additional depth decisions of all other sequences for the highest and lowest bitrates respectively. In terms of PSNR, B-DSDE performed the worst for UnicornA and Painter. Consequently, a reasonable expectation could be that for these two sequences, most decisions should be ESDE. However, we do not observe obvious decisions in these views. For Painter, only a small number of blocks are chosen as ESDE. This would imply, that barely any useful information was found in the uncoded Painter depth, which is more efficient after compression in comparison to DSDE depth. It implies further, that even though some useful information may be present in the uncoded depth, the compression is too costly. Nevertheless, for UnicornA it seems that more structures are chosen to be coded, especially the figures and objects on the table. Remarkably, the least amount of ESDE blocks are coded in the Shaman sequence, especially at low bitrates. ESDE as well as B-DSDE have shown the best performance in terms of synthesis PSNR for Shaman (and Painter) with over 34 dB compared to the average of 31 dB for all sequences. Therefore, if there is not much more performance to gain from coding the depth, it is explainable that barely any ESDE blocks are selected, since, in case of similar performance, B-DSDE will always win the competition.

### 4.3.2 Synthesis Quality

The view synthesis quality is evaluated for the lowest and highest bitrates. Fig. 4.8 and Fig. 4.9 show cropped areas in case of high bitrates for B-DSDE (left) and H-DSDE (right). Since the reference is B-DSDE, any change in the picture on the right is due to the partial transmission of depth. Several structures have been better recovered by the selected depth blocks, however, the differences are rather minor. In cases, where the noise-like characteristics of B-DSDE were overwhelming, the coded depth provide slightly better results, for example in the shirt of the Frog or the roof of the Dancing sequence. Due to "holes" in the B-DSDE depth maps of UnicornA, some artifacts were visible in the corresponding checkerboard, which has been corrected in H-DSDE. Structures like pots in Shaman or the objects in Kitchen have been better recovered in H-DSDE. In lower bitrates, see Fig. 4.10 and Fig. 4.11, the benefit of H-DSDE becomes even more apparent. This is expected as in high bitrates, the difference between B-DSDE and ESDE depth maps are rather minor. The benefit in low bitrates is very well visible in the Dancing sequence, where the B-DSDE synthesis suffers from strong distortions, which is mostly corrected in H-DSDE.

The objective performance of H-DSDE in comparison to B-DSDE is shown in Tab. 4.1. The objective performance is improved for the majority of sequences, confirming the observations in the provided view synthesis examples. Some strong fluctuations are visible in the metrics, *e.g.* high losses for Dancing (0.89 dB) but moderate improvements for UnicornA (0.4 dB) are given. Consequently, an improvement of merely 0.04 dB is seen for high bitrates. LPIPS generally confirms the PSNR values, indicating similar performance on average. It is reasonable, that at higher bitrates, the PSNR-performance of B-DSDE and ESDE are rather similar, leading to barely any benefit in H-DSDE. In low bitrates however, the benefit of H-DSDE is more apparent. While gains and losses are seen for the same sequences, the fluctuations are weaker. On average, a PSNR improvement of 0.12 dB and an LPIPS improvement of 0.002 is given for H-DSDE.

Table 4.1 – Average synthesis PSNR and LPIPS for B-DSDE and H-DSDE. The best performance is indicated in bold.

Config	Sequence	synth PSNR [dB]		LPIPS	
		B-DSDE	H-DSDE	B-DSDE	H-DSDE
Medium Bitrate	Painter	34.23	<b>34.34</b>	0.218	<b>0.217</b>
	UnicornA	29.38	<b>29.78</b>	0.063	<b>0.056</b>
	UnicornB	29.84	<b>30.18</b>	0.059	<b>0.056</b>
	Shaman	34.21	<b>34.26</b>	<b>0.243</b>	<b>0.243</b>
	Kitchen	31.05	<b>31.41</b>	0.176	<b>0.170</b>
	Dancing	<b>29.97</b>	29.08	<b>0.191</b>	0.202
	Chef2	31.91	<b>32.03</b>	0.260	<b>0.258</b>
	Frog	<b>27.59</b>	27.41	<b>0.208</b>	0.215
	Average	31.02	<b>31.06</b>	<b>0.177</b>	<b>0.177</b>
Low Bitrate	Painter	32.75	<b>32.87</b>	<b>0.331</b>	<b>0.331</b>
	UnicornA	28.20	<b>28.54</b>	0.122	<b>0.116</b>
	UnicornB	28.65	<b>28.90</b>	0.121	<b>0.119</b>
	Shaman	32.75	<b>32.87</b>	<b>0.418</b>	0.419
	Kitchen	29.79	<b>30.17</b>	0.317	<b>0.311</b>
	Dancing	28.11	27.93	0.374	<b>0.370</b>
	Chef2	31.29	<b>31.41</b>	0.322	<b>0.321</b>
	Frog	<b>26.59</b>	26.44	<b>0.337</b>	0.344
	Average	29.77	<b>29.89</b>	0.293	<b>0.291</b>





Figure 4.5 – Evolution of H-DSDE selection (left) and corresponding view synthesis quality (right) with falling bitrate for natural content (Frog sequence). The grey blocks indicate omitted depth CTUs, which have to be recovered at the decoder side.

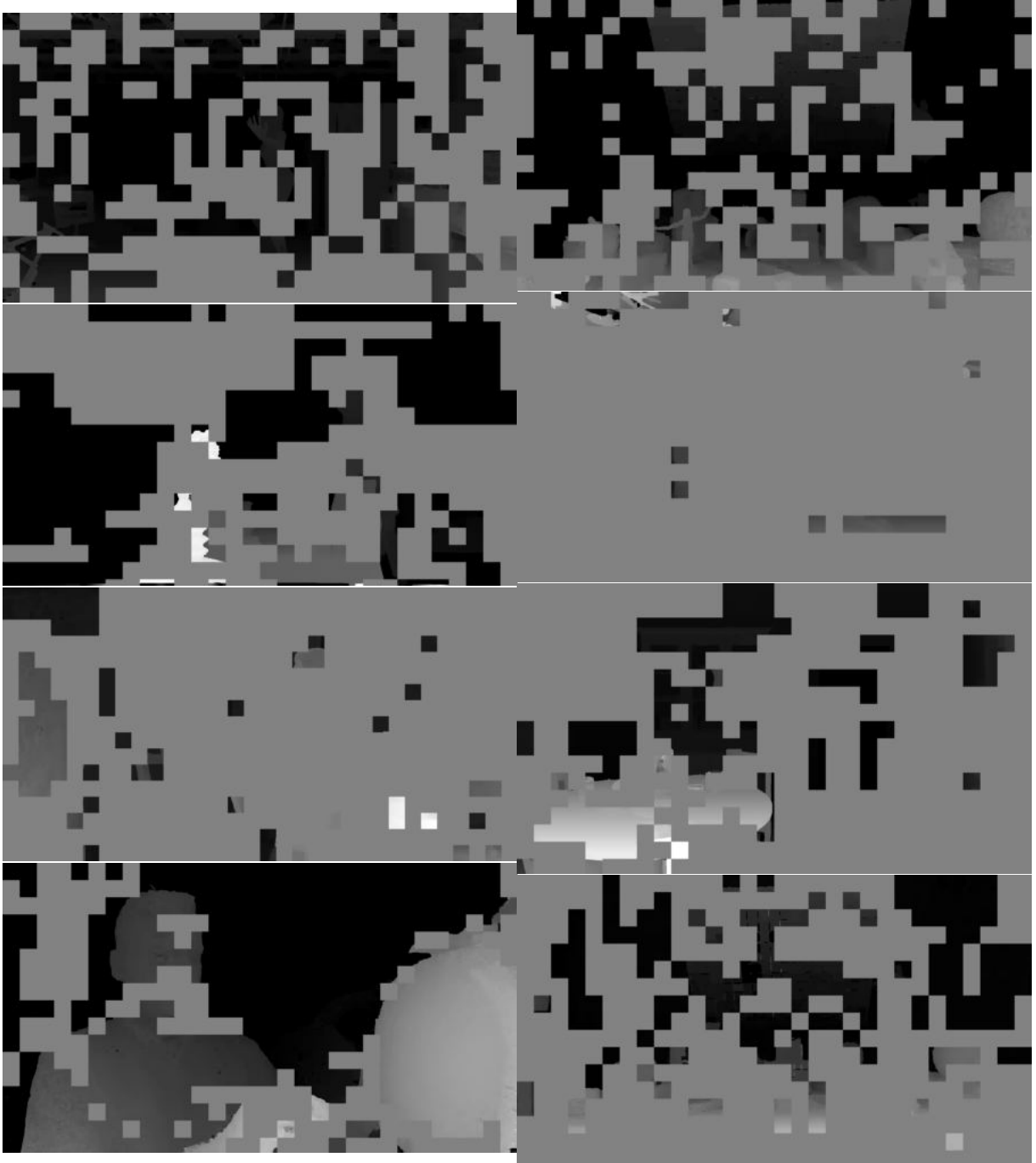


Figure 4.6 – H-DSDE depth maps for high bitrate.

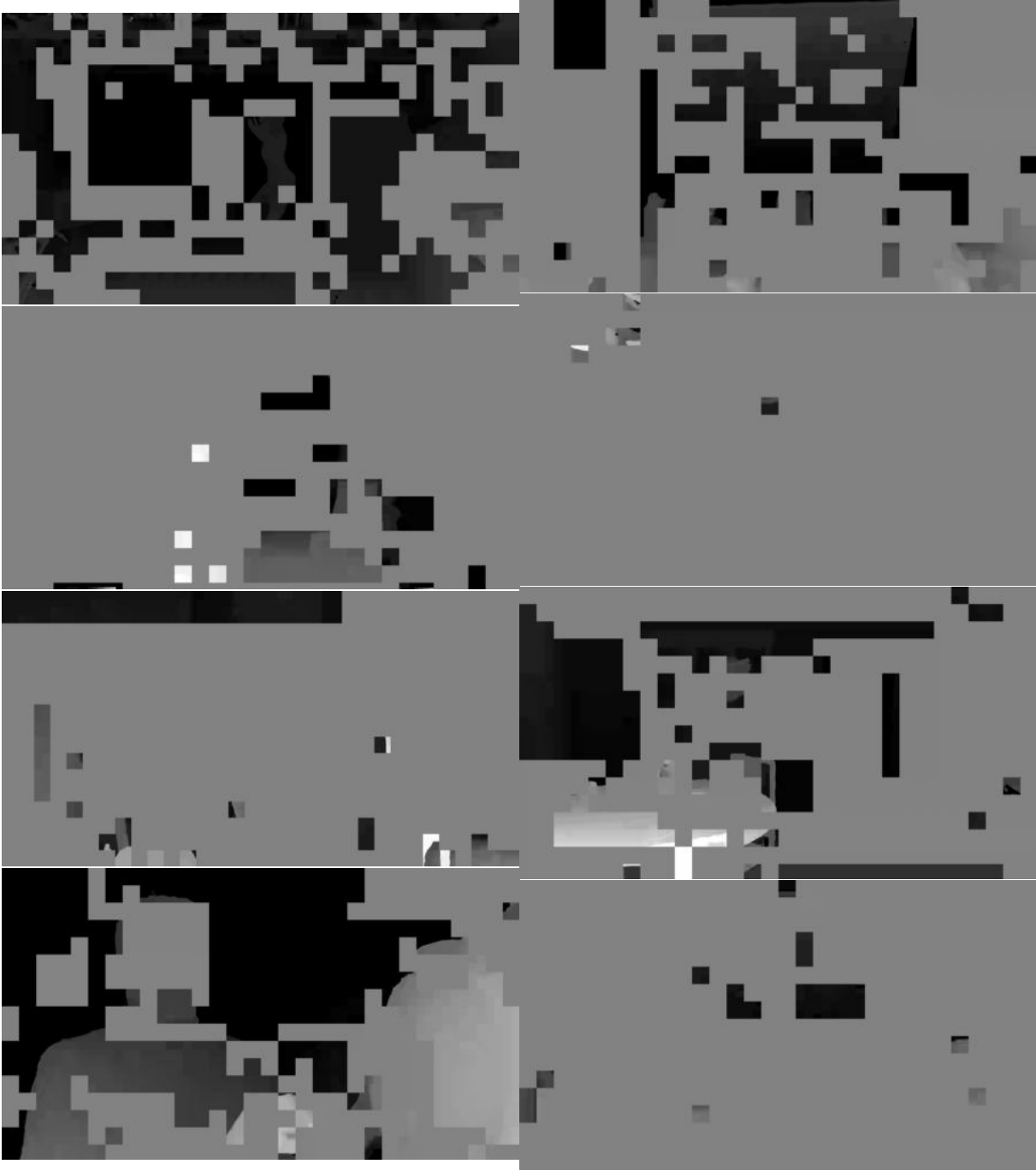


Figure 4.7 – H-DSDE depth maps for low bitrate.

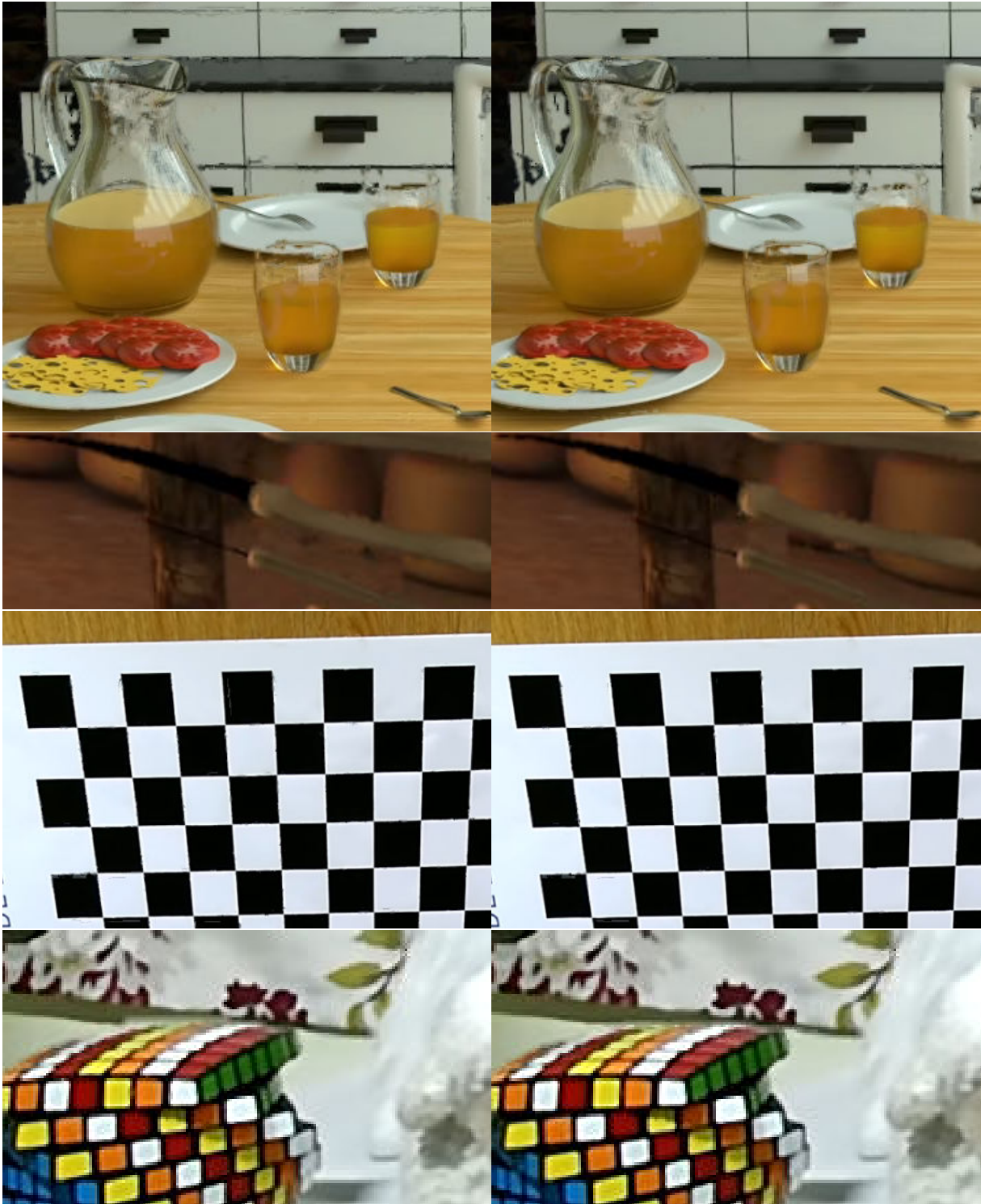


Figure 4.8 – H-DSDE synthesized views for high bitrate (1/2).



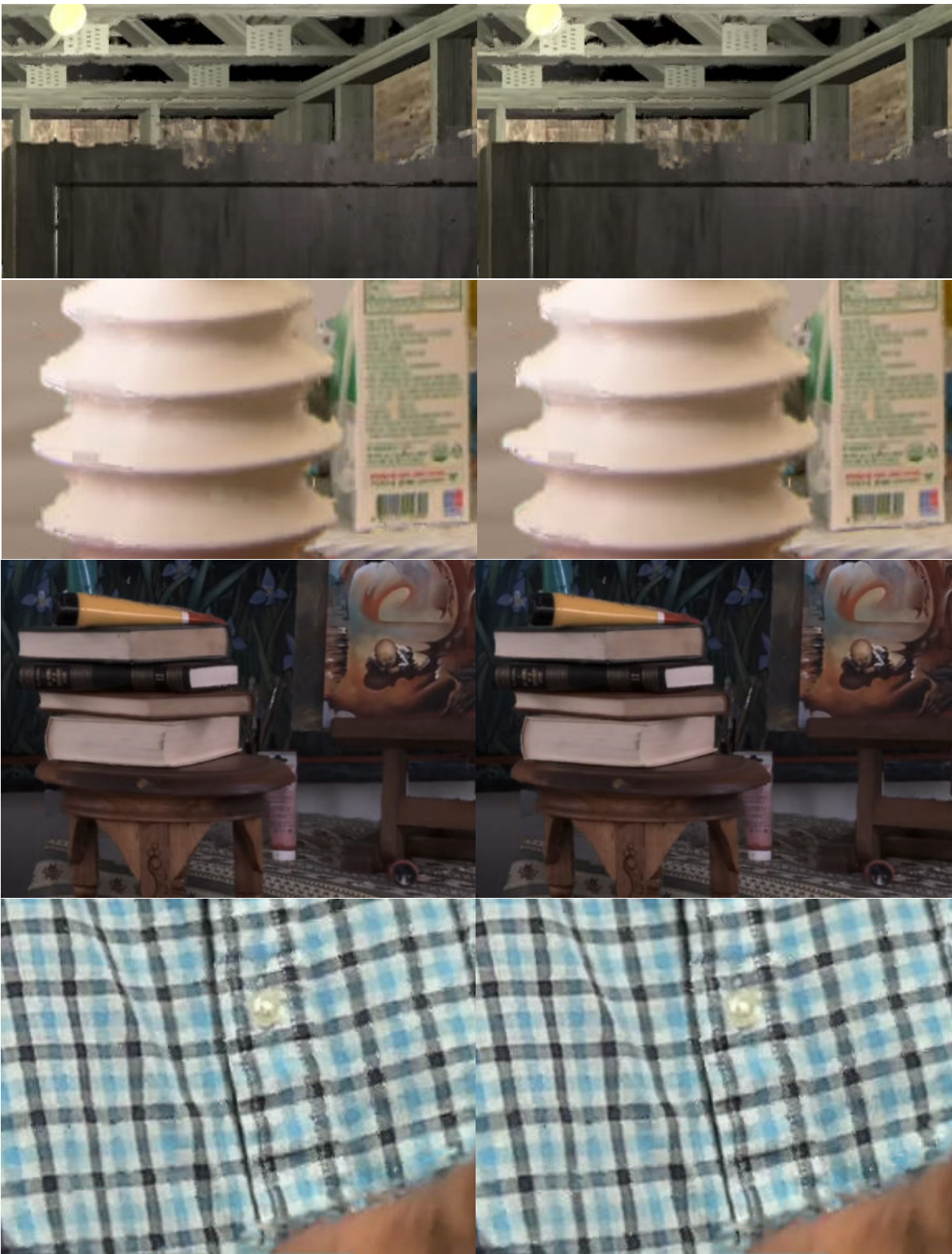


Figure 4.9 – H-DSDE synthesized views for high bitrate (2/2).

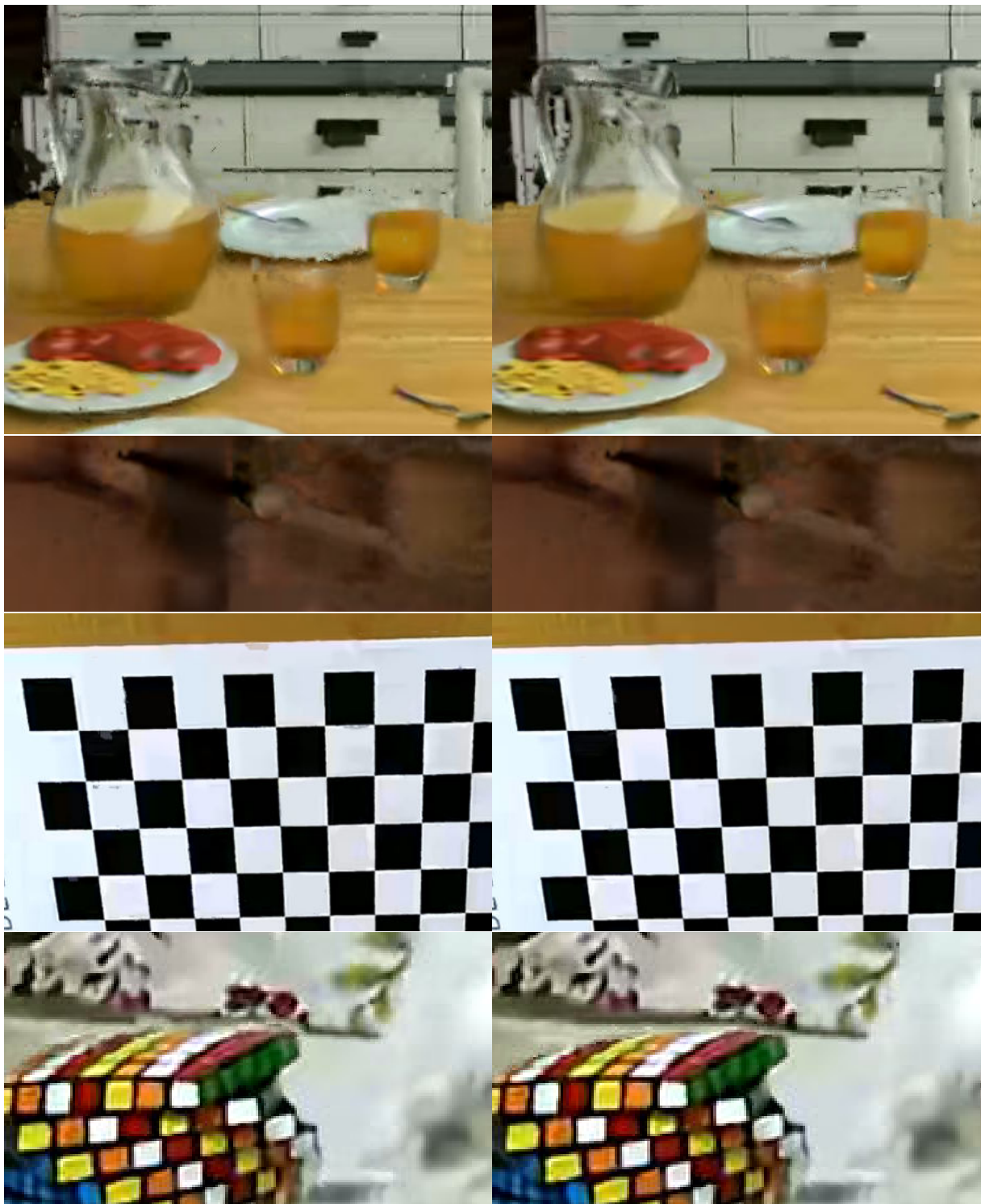


Figure 4.10 – H-DSDE synthesized views for low bitrate (1/2).

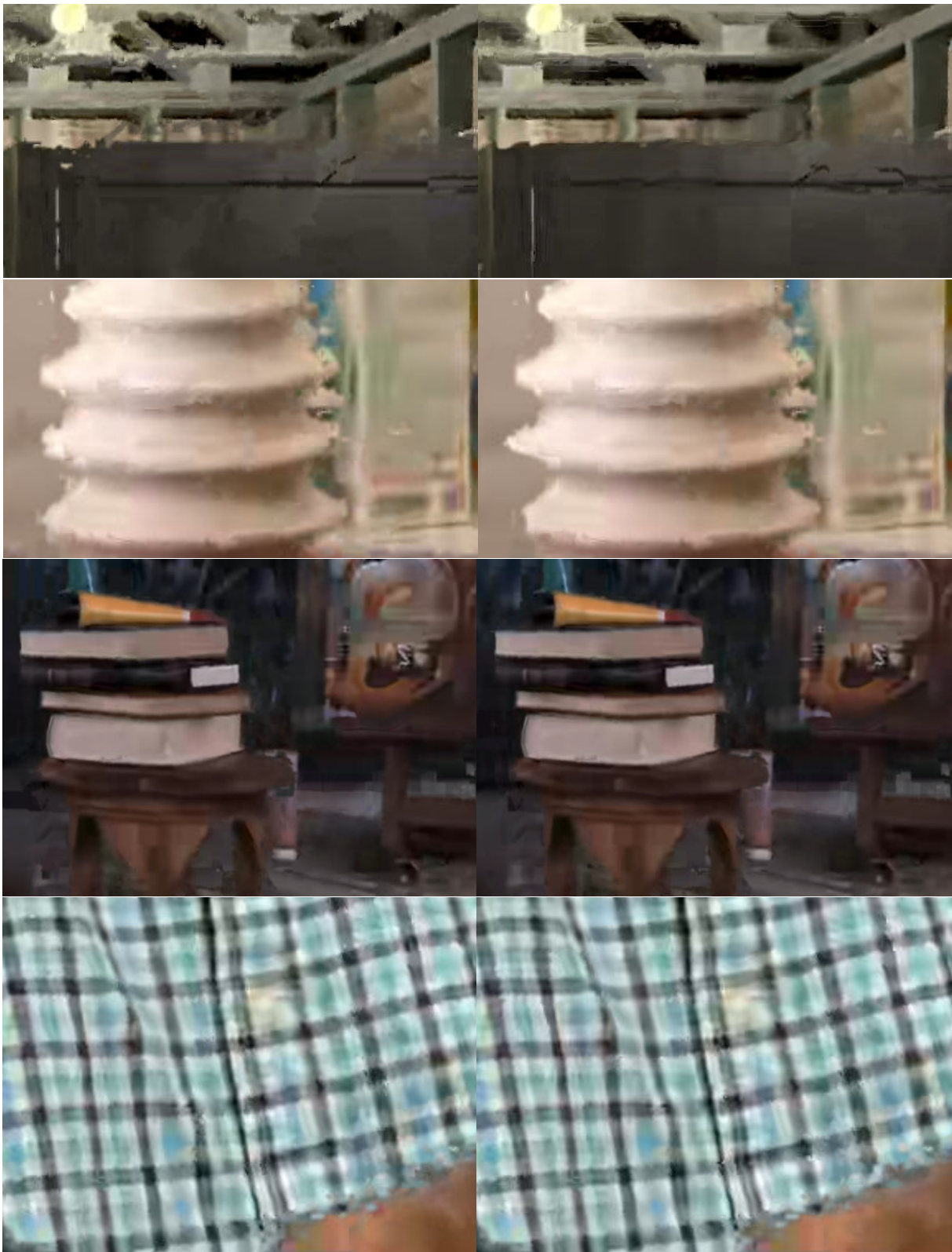


Figure 4.11 – H-DSDE synthesized views for low bitrate (2/2).

### 4.3.3 Coding Efficiency of H-DSDE

Finally, all metrics have to be considered together to judge the performance of H-DSDE as a coding system. Tab. 4.3 summarizes the BD-Rate results for several metrics. The BD-Rate is computed for B-DSDE (for reference) and H-DSDE compared to ESDE. The video BD-Rate shows that the bitrate cost for partial coding of depth is higher than for B-DSDE, which is natural. In some cases, losses compared to ESDE are seen, which means, the cost for the partial depth is higher than the MV-HEVC coded full depth maps. In a classical rate-distortion optimization scenario, this behaviour is not expected. However, in this hybrid scheme, the distortion has to rely on an approximation, which is a limitation of the approach itself. This is also clear, as our design does not include any removal of inter-view redundancy, hence, for sequences like Dancing, the highest loss of 8.1% and 14.7% are seen for high and low bitrates respectively, since it this sequence consists of the most views. Due to the additional bitrate required to transport the local depth maps, H-DSDE does not outperform B-DSDE in terms of overall compression efficiency. The limitations of the rate-distortion formulation and derivation of the distortion value have been elaborated earlier. This is a limitation inherent to DIBR systems and we believe this is the reason why the encoder fails to select the ESDE blocks most efficiently. Moreover, the additional redundancy in the H-DSDE makes a competition with MV-HEVC challenging. Overall, H-DSDE outperforms ESDE by  $-34.9\%$  and  $-24.8\%$  synthesis PSNR BD-Rate for high and low bitrates respectively. Similarly,  $-18.0\%$  and  $-10.2\%$  MS-SSIM BD-Rate improvement as well as  $-33.1\%$  and  $-24.4\%$  VMAF BD-Rate improvement is reported.

The H-DSDE system re-introduces the video-based depth coding in comparison to B-DSDE. Consequently, pixel rate is invested for depth maps. Simultaneously, the coded depth maps do not need to be estimated at the decoder-side. Therefore, H-DSDE has two effects: the pixel rate is increased, but the decoder-side depth estimation complexity is reduced. Tab. 4.2 summarizes the hypothetical reduction in pixel rate and cost volume size relative to ESDE. The values are hypothetical, since the coding of "grey blocks" does not truly save pixel rate and the decoder side depth estimator does not truly estimate only the B-DSDE blocks in our experiments (a limitation that is overcome in Chapter 5). The required pixel rate for depth decoding is different for each sequence. Since the amount of such blocks is unknown, the decoder must have capabilities for the worst case, and therefore have the complexity for pixel rates that are equivalent to the ESDE design. The cost volume reduction of  $-38.49\%$  and  $-36.11\%$  implies, that in case of complexity constraints on the decoder-side, partial depth transmission could be utilized to make the



Table 4.2 – Pixel Rate and Cost Volume Reduction for H-DSDE.

Config	Sequence	PRR [%]	CVR [%]
Medium Bitrate	Painter	-39.43	-21.13
	UnicornA	-22.55	-54.89
	UnicornB	-21.13	-28.83
	Shaman	-41.05	-17.89
	Kitchen	-32.24	-35.51
	Dancing	-14.78	-70.42
	Chef2	-30.64	-34.58
	Frog	-27.65	-44.68
	<b>Average</b>	<b>-28.68</b>	<b>-38.49</b>
Low Bitrate	Painter	-39.78	-20.42
	UnicornA	-19.30	-61.39
	UnicornB	-32.16	-35.7
	Shaman	-45.13	-9.72
	Kitchen	-35.60	-28.79
	Dancing	-18.46	-63.07
	Chef2	-30.64	-20.42
	Frog	-26.80	-46.39
	<b>Average</b>	<b>-30.98</b>	<b>-36.11</b>

system feasible.

## 4.4 Future Work

H-DSDE has been developed during the early stages of the MIV and most tools described in Chapter 2 were not available in the TMIV. The design of H-DSDE presented in this chapter breaks an important premise, which is the agnosticism of the 2D codec towards MIV. However, this section shall outline, how H-DSDE could be embedded in the current design of the MIV standard.

In order to evaluate the ESDE candidate, the block has to go through a coding process as described in this chapter. Subsequently, the blocks of the *uncoded* depth map are packed into a patch atlas based on the decisions of the H-DSDE encoding. The metadata related to the patch atlas is encoded through the MIV bitstream. The resulting patch atlas is encoded by an unmodified 2D codec. In such an approach, the provided hypothetical pixel rate and cost volume reductions can be approximately achieved.

Table 4.3 – Summary of the BD-Rate objective metrics for B-DSDE and H-DSDE compared to ESDE. The best synthesis performance is indicated in bold.

Config	Sequence	video BD-Rate [%]		synth PSNR BD-Rate [%]		synth MS-SSIM BD-Rate [%]		synth VMAF BD-Rate [%]	
		B-DSDE	H-DSDE	B-DSDE	H-DSDE	B-DSDE	H-DSDE	B-DSDE	H-DSDE
Medium Bitrate	Painter	<b>-36.0</b>	-24.0	<b>-31.7</b>	-23.0	<b>-30.9</b>	-19.4	<b>-35.4</b>	-25.1
	UnicornA	<b>-4.7</b>	2.0	6.0	<b>-14.6</b>	<b>-5.9</b>	-4.2	5.5	<b>-9.8</b>
	UnicornB	<b>-5.6</b>	-0.8	-3.6	<b>-22.7</b>	<b>-7.8</b>	<b>-7.8</b>	-4.4	<b>-20.2</b>
	Shaman	<b>-32.9</b>	-20.9	<b>-55.2</b>	-46.7	<b>-72.4</b>	-31.5	<b>-62.7</b>	-53.3
	Kitchen	<b>-18.2</b>	1.3	<b>-39.2</b>	-37.0	<b>-24.3</b>	-12.4	<b>-43.3</b>	-34.6
	Dancing	<b>-5.3</b>	8.1	<b>-72.4</b>	-42.6	<b>-48.7</b>	-22.8	<b>-74.5</b>	-42.8
	Chef2	<b>-27.8</b>	-5.3	<b>-58.4</b>	-50.9	<b>-40.7</b>	-25.7	<b>-53.3</b>	-43.5
	Frog	<b>-11.4</b>	-1.3	<b>-57.8</b>	-41.9	<b>-36.2</b>	-20.8	<b>-53.3</b>	-35.7
	Average	<b>-17.4</b>	-5.1	<b>-39.0</b>	-34.9	<b>-33.4</b>	-18.0	<b>-40.8</b>	-33.1
Low Bitrate	Painter	<b>-39.2</b>	-23.2	<b>-35.1</b>	-20.8	<b>-34.8</b>	-18.0	<b>-39.8</b>	-24.3
	UnicornA	<b>-5.4</b>	3.7	-6.7	<b>-11.1</b>	<b>-8.3</b>	-1.1	-5.0	<b>-7.1</b>
	UnicornB	<b>-6.7</b>	-0.2	-13.8	<b>-15.9</b>	<b>-9.2</b>	-3.8	-14.3	<b>-14.7</b>
	Shaman	<b>-39.7</b>	-20.9	<b>-50.0</b>	-34.0	<b>-70.1</b>	-23.0	<b>-60.0</b>	-42.9
	Kitchen	<b>-22.3</b>	6.8	<b>-36.0</b>	-21.3	<b>-26.0</b>	0.5	<b>-41.8</b>	-21.0
	Dancing	<b>-8.0</b>	14.7	<b>-51.4</b>	-35.1	<b>-32.9</b>	-11.1	<b>-59.6</b>	-34.4
	Chef2	<b>-31.2</b>	-2.2	<b>-54.3</b>	-36.0	<b>-40.9</b>	-14.9	<b>-52.8</b>	-28.8
	Frog	<b>-15.0</b>	0.0	<b>-42.8</b>	-24.8	<b>-28.5</b>	-9.8	<b>-41.5</b>	-22.2
	Average	<b>-21.0</b>	-2.7	<b>-36.3</b>	-24.8	<b>-31.3</b>	-10.2	<b>-39.3</b>	-24.4

## 4.5 Conclusion

This chapter has presented a possible implementation of an H-DSDE system using HM. The proposed system has effectively chosen depth blocks to be coded, which improve the quality of the view synthesis at the decoder-side. This has been achieved using an adapted SVDC-based approach. However, the inefficiencies in simulcast HM and limitations of the rate-distortion formulation prevented to outperform B-DSDE in terms of BD-Rate. Additional steps are required in order to include H-DSDE into the MIV standard. However, this system maintains the disadvantages of both: ESDE and B-DSDE. All the problems that were solved through B-DSDE have re-appeared, including  $QP_D$  allocation for the partial depth, complex encoder optimization, pixel rate cost and inefficient depth coding. Simultaneously, the B-DSDE problems are only partially addressed: depth estimation and decoder complexity. In H-DSDE, a decoder has to be ready to decode depth information and at the same time estimate the missing depth. For these reasons, we conclude from the hybrid approach that depth map transmission, even when limited to local cases, does not seem to be a promising design. Furthermore, additional steps are required in order to embed H-DSDE into the context MIV. However, during the final stages of the MIV standard, H-DSDE has been proposed for MIV and is still under investigation in 2022 at the time of writing this thesis (see Chapter 6).

# FEATURE-DRIVEN DECODER SIDE DEPTH ESTIMATION (FD-DSDE)

---

The H-DSDE method considers the video-based coding of partial depth information, which enables to benefit from the advantages of both, the ESDE and the B-DSDE system. However, the disadvantages are maintained as well. This chapter introduces the Feature-Driven Decoder Side Depth Estimation (FD-DSDE) system. The concept of this approach is to identify relevant side information, denoted as *features*, which support the depth estimator at the decoder-side to perform depth estimation significantly faster and with higher accuracy. By avoiding the video-based coding of depth entirely, the disadvantages of ESDE are avoided while simultaneously, the presence of high quality depth maps on the encoder-side is exploited.

## 5.1 Motivation for a feature-driven DSDE solution

We conclude from the H-DSDE experiments (see chapter 4), that depth map transmission using a video codec, globally or locally, is inefficient. Significant modifications of the depth map encoder may be required, including the usage of depth coding tools. However, such an approach goes against the premise of a video-based volumetric video coding solution and may lead to a dead-end (similar to 3D-HEVC). Alternatively, H-DSDE may be included into a MIV bitstream, however, the disadvantages of ESDE, including  $QP_D$  allocation and pixel rate cost would be maintained. Consequently, we do not deepen our investigation into the concept of H-DSDE. However, we learned that the depth maps, present at the encoder-side, may contain information which cannot be recovered by the same depth estimator at the decoder-side using decoded textures. Furthermore, we un-

derstand that the task of decoder-side depth estimator is not similar to depth estimation typically discussed in literature: there is no necessity to perform the depth estimation "blindly", because the true depth maps are already known (at the encoder-side). Therefore, the question to address is: given that the depth maps are already known, what kind of information could be provided to a depth estimator, in order to estimate the depth faster and in higher accuracy? The goal is to identify a strategy, which takes advantage of the presence of high quality depth maps at the encoder side, without coding them using a 2D codec, without losing the advantages of B-DSDE and without introducing the disadvantages of ESDE. Even further, we intent to attenuate the downsides of B-DSDE, which is the high complexity at the decoder side.

## 5.2 Description of the FD-DSDE system

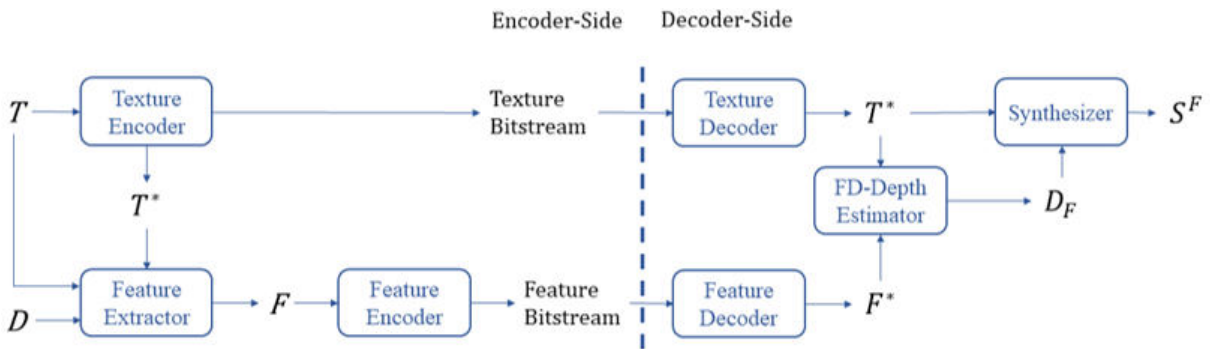


Figure 5.1 – Overview of the proposed FD-DSDE system. The input signals are original multiview textures  $T$  and multiview depth maps  $D$ .  $T^*$  are coded textures  $F$  and  $F^*$  are the original and coded feature sets respectively.  $D_F^+$  are feature-enhanced decoder-derived depth maps.  $S^F$  are synthesized views.

FD-DSDE is a proposed, novel system based on B-DSDE and addresses two major drawbacks of the latter: decoder complexity, and depth estimation inaccuracy. Considering the BD-Rate performance of B-DSDE presented in Chapter 3, complexity may be the biggest concern of DSDE in general. The Feature Driven DSDE (FD-DSDE) system is shown in Fig. 5.1. A feature extractor identifies features that help the depth estimation at the decoder side. The features are chosen during encoding time to generate a depth map which minimize the distance to the original depth map. The encoder has access to the decoded version of the textures and can thus simulate the behavior of the decoder side depth estimator in order to correctly choose the feature values. Since the features

are extracted and optimized at the block level, the depth estimator has been modified to operate on a block-basis. For each block, the following features are extracted and transmitted: depth ranges, optimal depth estimation parameters, depth estimation skip flag and a partition flag, which are detailed in the following paragraphs.

**Depth Range** traditionally, the camera distance  $Z_{min}$  and  $Z_{max}$  are required by most depth estimators, in order to define the search range, which can alternatively be described in terms of disparity to a fixed reference view as  $d_{max} = \frac{f \cdot b}{Z_{min}}$  and  $d_{min} = \frac{f \cdot b}{Z_{max}}$  with the focal length  $f$  and the baseline to a reference view  $b$ . The disparity range  $[d_{min}, \dots, d_{max}]$  defines the number of disparity candidates that need to be tested. The number of candidates is  $N = (d_{max} - d_{min}) \cdot p + 1$  with optional sub-pixel precision  $p$ . The overall size of the cost volume is therefore  $CV = N \cdot w \cdot h$  with width  $w$  and height  $h$  of a view. Each pixel requires an analysis over the full range, which is defined by the farthest and closest object in the view.

For each block, the feature extractor simply selects  $Z_{min}$  and  $Z_{max}$  from the original depth map  $D$ . In the same spirit, the available geometry has been utilized previously in order to optimize the computation of plane-sweep volumes in the context of CNN-based view synthesis [57] [58].

Narrowing down depth ranges per block aims at reducing the complexity by minimizing the cost volume that has to be computed. Enabling partial cost volumes increases the speed of block-matching, depth selection (*e.g.* by graph-cuts) and makes the estimated depth maps more accurate by avoiding depth candidates that are out of the correct range. This is illustrated in Fig. 5.2, where the number of candidates is minimized for smooth areas. The much larger default number of depth candidates is only justified where a block contains a transition between foreground and background.

**Depth Estimation Skip** the depth estimation skip is signaled through a single flag which indicates if a block requires re-estimation. If a block is skipped, the depth of the previous frame is re-used and the encoder does not transmit any additional features. Fig. 5.3 indicates, which blocks require re-estimation. In the given example, only around 7.5% of the blocks require a re-estimation of the depth map. The skip flag aims at minimizing complexity and maintaining temporal stability in the depth maps and synthesized textures. While it would be possible for the decoder side depth estimator to determine whether a block depth should be updated, this would require additional computation and

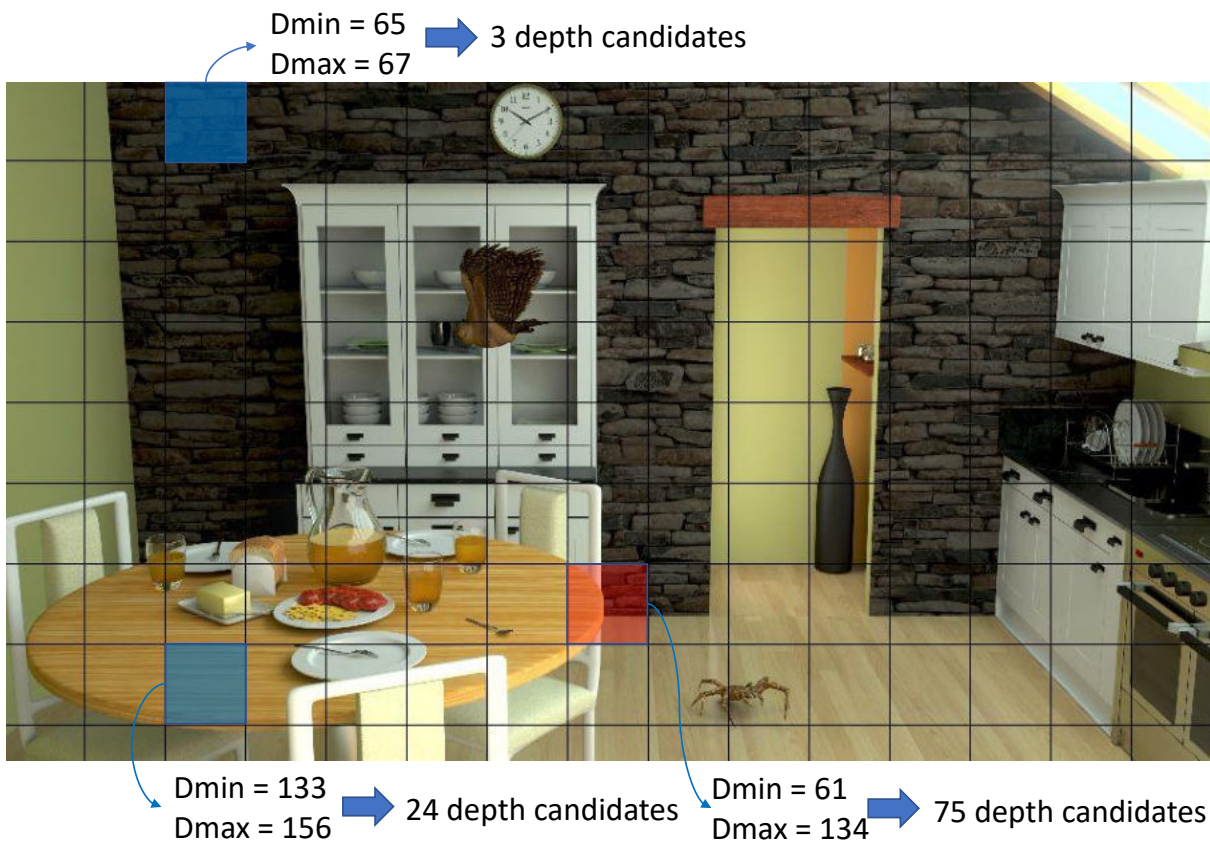


Figure 5.2 – Illustration of required number of depth candidates per block, specified by the transmitted depth range ( $D_{min}$  and  $D_{max}$ ).

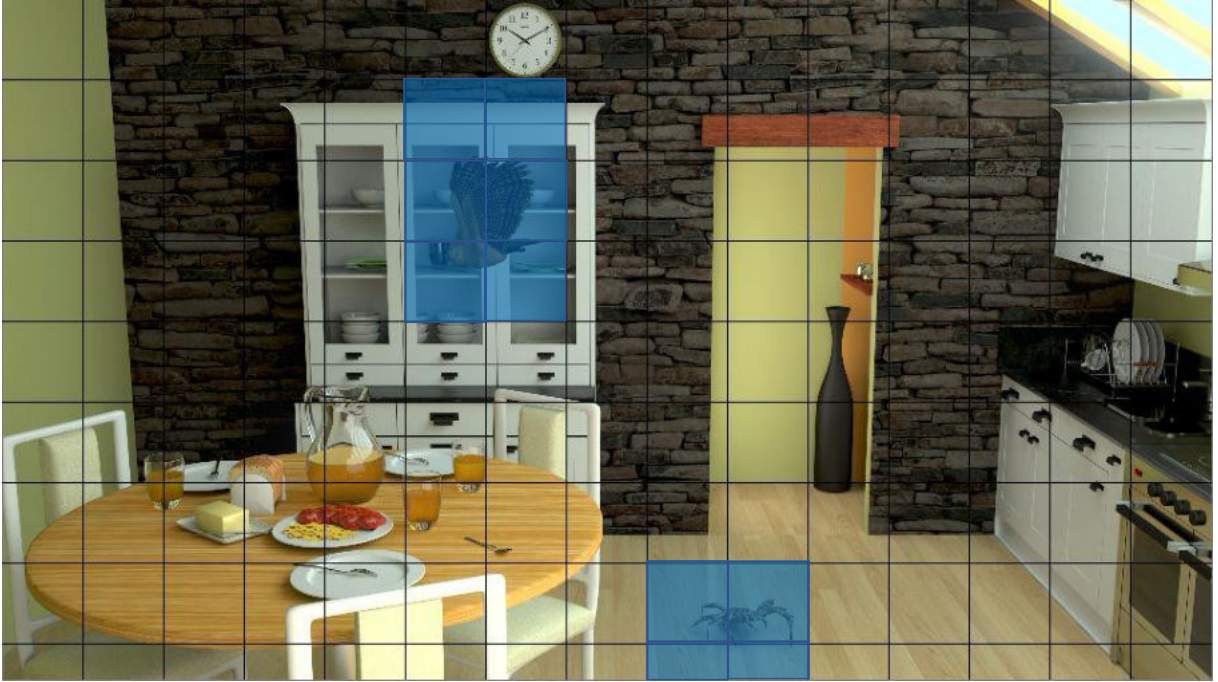


Figure 5.3 – Illustration of depth estimation skip. Highlighted blocks are re-estimated while all other blocks are skipped.

would go against our objective to reduce the complexity.

**Partitioning** with the design of a block-based approach, a maximum block size has to be selected. Partitioning enables the feature extractor to transmit features at a finer granularity if beneficial. We enable a simple partitioning in a quad-tree fashion. The benefit is illustrated in Fig. 5.4. In the given example, a range of [61,...,134] has to be analyzed in a block of size  $N \times N$ . The range is large, because the block contains parts of the table (close to camera) and part of the wall behind the chair (far from camera). By partitioning, each block can further minimize the search range, leading to a reduction of depth candidates that have to be tested by an additional 50%.

**Depth Estimation Parameters** most Multiview or Stereo Depth Estimation methods include a form of regularization which favors homogeneous regions, unless a substantially large change is detected locally. This type of regularization is usually dependent on parameters determining the amount of change requested to consider a transition as valid, such as smoothing coefficients in energy-minimization methods. We choose to transmit optimized smoothing coefficients as features, because the strength of the regularization is



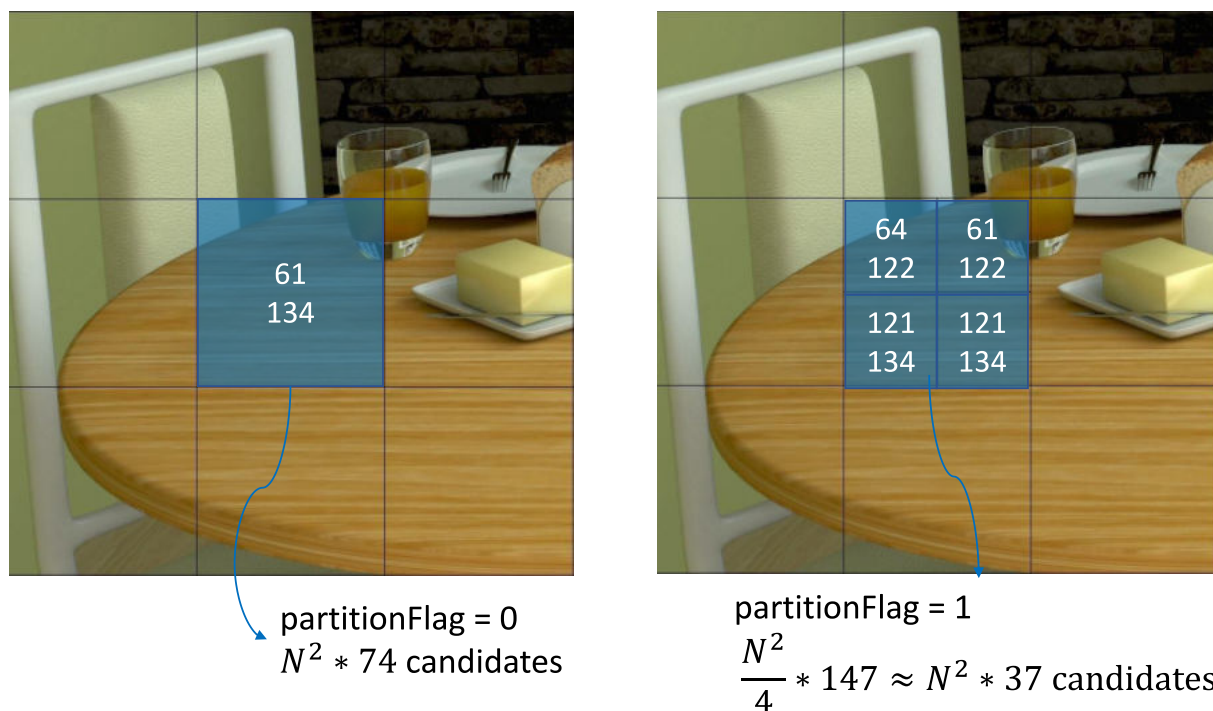


Figure 5.4 – Illustration of the partition flag. The highlighted block shows  $D_{min}$  and  $D_{max}$ . On the left, parts of the background are in the block and 74 depth candidates have to be tested. On the right, the partition flag is used and the back- and foreground are more efficiently separated, reducing the total number of depth candidates to 37.

heavily signal-dependent. In our feature-extractor design, several smoothing coefficients are tested by simulating the depth estimation process and selecting the value which minimizes the error with the block in the reference depth.

In this work, our feature extractor selects the features by minimizing the L2 distance of a block between the estimated and the reference depth. In other words, we *encode* the reference depth into a feature set, which helps the depth estimator at the decoder side to estimate depth maps using decoded textures. This criterion is used to identify partitioning and depth estimation parameters. The depth estimation skip is performed if the L2 distance between two temporally adjacent blocks in the original source textures is below a threshold. For CGI content, the depth maps can be used directly to identify the depth estimation skip, where any temporal change implies re-estimation.

Our feature extractor is based on DERS8, where every possible feature set is tested by estimating the depth on a block-basis. The DERS8 used at the decoder side has been modified accordingly, reading the received features and estimating the depth faster and with higher performance. One should note that FD-DSDE does not require any synthesis during encoding, so it is relatively light when compared to other processing stages such as texture coding. The features are compressed using Context-based Binary Arithmetic Coding (CABAC) [59] with a single context assigned for each feature. We encode the features losslessly and therefore optimize for complexity minimization instead of BD-Rate performance, since complexity is the biggest concern of the DSDE system.

Fig. 5.5 illustrates the impact of our features on depth estimation and synthesis performance. In this example, uncoded textures are used for depth estimation and synthesis. A) shows the performance of unmodified DERS8. F) shows the performance of synthesis if blender depth is used. B)-E) gradually include additional features, extracted from the blender depth. We observe a degradation of quality by moving from a full-frame to a purely block-based method in B). This is expected, as the limited texture available in certain blocks can make the depth estimation more challenging. However, by transmitting the depth range per block in C), many artifacts are removed from the estimated depth map and an improvement of the visual quality is obtained. In D) the depth estimation is enhanced by additional smoothing coefficients, which further reduces the distance to the blender reference and has the biggest impact on the objective quality. In E), partitioning is included, which further refines the depth map locally.

While the difference in objective metrics may seem minor compared to the unmodified DERS8, the impact increases with decoded textures, which we show in section V us-

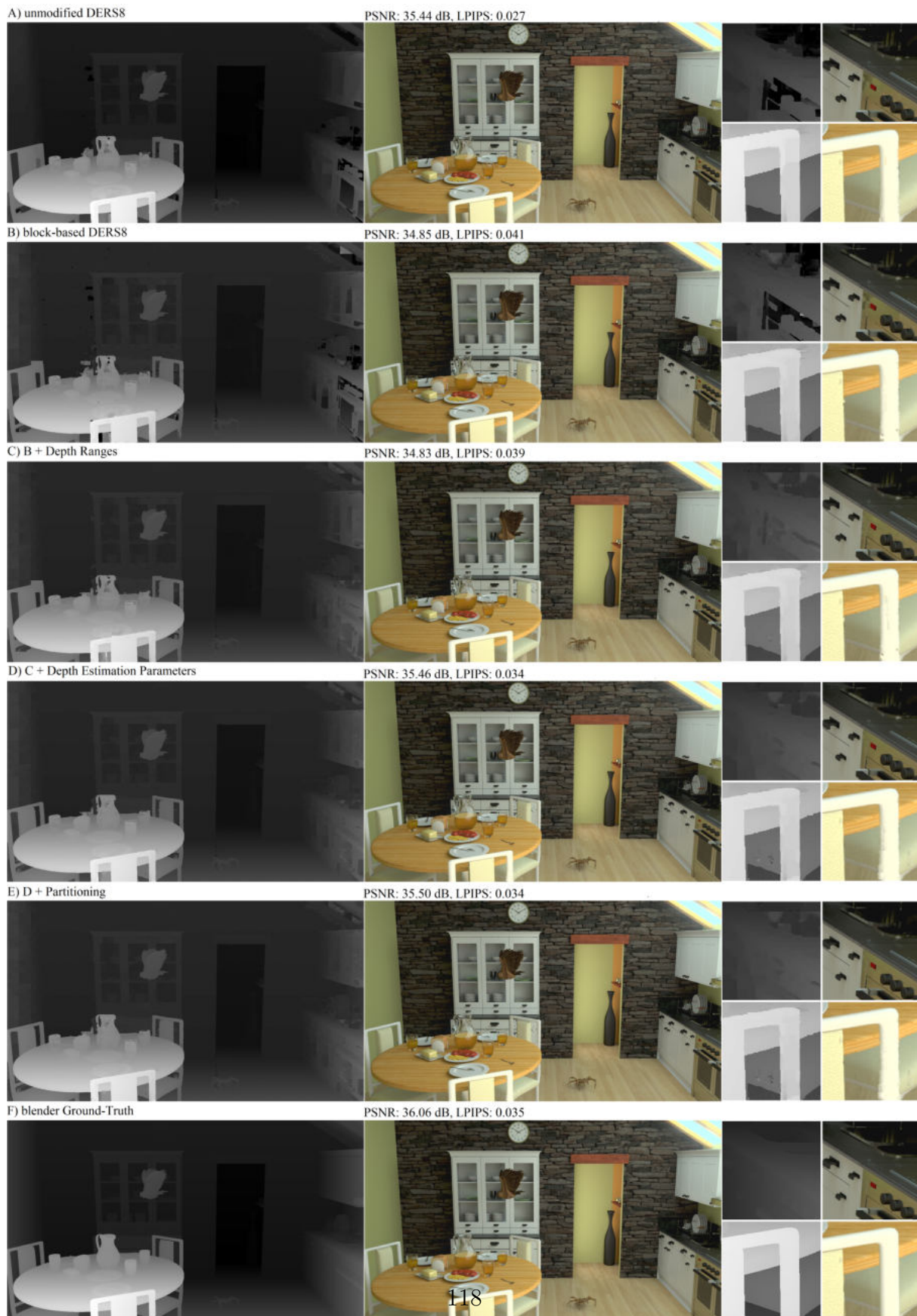


Figure 5.5 – Illustration of the impact of features, extracted from high quality blender depth. Depth Estimation Parameters and Partitioning are chosen to maximize PSNR.

Table 5.1 – Entropy Rate (in kbps) and their fraction of the features of experiments B-E.

Experiment	Depth Ranges	Depth Estimation Parameters	Partitioning	Depth Estimation Skip	Total
B	1203 (100%)				1203
C	1203 (73%)	441 (27%)			1644
D	1278 (72%)	466 (26.5%)	13 (0.8%)		1757
E	120 (52%)	45 (19.5%)	1.0 (0.4%)	65 (28%)	230

ing DERS8 depth maps as a reference. Additionally, when high-quality depth maps are available (*e.g.* for CGI), this approach can guide the depth estimator to achieve a higher quality depth maps.

In order to provide a first guess of the required bitrate of these features, we compute the entropy rate for all experiments B-E. The average entropy rate and their fractions are summarized in Tab. 5.1. The assumed block size is 128. The number of frames is 16. Regarding the coding of depth ranges (experiment E), we observe that the required rate (1203 kbps) is not insignificant. The required bitrate is high as the depth ranges can strongly differ among all blocks of a frame and GOP. The coding of two smoothing coefficients (experiment C) is much cheaper, as only pre-defined coefficients are tested. The rate required for partitioning (experiment D) is negligible as it is represented by a single flag per block. Furthermore, it is rarely selected, indicating that a block size of 128 is appropriate. The depth estimation skip (experiment E) has a significant impact on the required bitrate for all features, as it enables us to omit the coding of the other features, if a block or sub-block is indicated as a block to be skipped. The total entropic rate is therefore significantly reduced, making the skip flag a valuable addition to the features. Further experiments show that the bitrate of the features is roughly proportional to the number of blocks. Consequently, a blocksize of 64 is considered too expensive as the required rate is approximately four times larger than using a block size of 128. Therefore, the following experiments continue to assume an initial block size of 128.

The depth ranges are directly extracted from the depth maps. The depth estimation parameters are selected by simulating the depth estimation for each parameter set. The depth estimation skip and the partitioning have more options for the selection process. The motivation of the depth estimation skip is to only estimate depth blocks at the decoder side, if changes in the depth have been identified by the feature extractor. In case of CGI depth maps, the situation is typically simple, as a change in depth pixels is truly caused by a change of depth of the corresponding object. In case of natural content

and estimated depth maps however, depth values may change temporally, even for static content, for which the depth did not truly change. Some level of noise is especially visible at object boundaries, leading to flickering in the corresponding synthesized view. This situation can occur if temporal coherence is not sufficiently or not at all considered in the depth estimator. Consequently, a threshold  $\text{skipTh}$  is defined in order to classify a block as "requires re-estimation". A constant  $\text{skipTh}$  is identified empirically for each sequence. Note, that these values have empirically shown good result, but may be imperfect in certain cases: some blocks may be re-estimated, even though it is not truly necessary (negatively impacting temporal stability, CVR and runtime) and some blocks may not be re-estimated, even though it is necessary (negatively impacting depth maps and view synthesis quality). However, the thresholds have been chosen in favor of the quality.

Alternatively, the uncoded textures could be used to identify the skip flag. However, as textures may contain significantly different levels of noise, a threshold cannot be avoided. Furthermore, a change in the textures may not necessarily require a re-estimation of depth (e.g. if the illumination changes, but the objects depth does not). More sophisticated methods may be developed to identify these features more robustly, and there is significant room for encoder optimization. The partition flag can similarly chosen based on different criteria, using either uncoded depth maps or textures. In our case, we chose to partition if the resulting depth map further minimizes the L2-distance to the reference depth. Alternatively, a minimization of the complexity (as implied in Fig. 5.4) or a synthesis-based optimization could be considered.

### 5.3 Impact of quantization

The feature extractors allows for quantization of the depth ranges according to the following formulas

$$Z_{min,q} = (Z_{min})/q^2 \tag{5.1}$$

$$Z_{max,q} = (Z_{max} + q - 1)/q^2 \tag{5.2}$$

The quantized depth ranges must now further narrow down the depth ranges, as this will certainly introduce errors in the estimated depth map. Stronger quantizations, which increase the depth range interval, will mainly attenuate the quality enhancing and complexity reducing effects of the depth ranges. We test several quantization levels  $q = 1, 2, 4, 8$

with  $q=1$  referring to lossless compression.

Tab. 5.2 summarizes the average video and synthesis BD-Rate, the PSNR and the CVR for all quantization levels and for high and low bitrates. In this scenario, we compare with B-DSDE. Consequently, the video BD-Rate will always indicate a loss and reflects the bitrate which has been invested into the features. As expected, the video BD-Rate gradually decreases with higher quantization. More importantly, the synthesis BD-Rate for high bitrates reveals that the best performance is achieved for lossless compression of the depth ranges. This is a surprising result, as the cost of the features may have been expected to be high. However, as indicated by the PSNR difference, providing accurate depth ranges is highly important for pixel-accurate performance. Minor deviations in the depth ranges may already introduce pixel-shifts in the corresponding synthesized view, which has a high impact on PSNR. Naturally, the CVR is the best without quantization. In the low bitrate case, higher levels of compression may be advised, as  $q=8$  indicates the best BD-Rate performance. This result is expected, as with lower bitrates, higher degradations in the features may be tolerated in favor of bitrate savings.

In light of these results, we conclude that lossless compression of the depth range is advised. It may be possible to tune the quantization parameter for lower bitrates. However, the benefit seems to be minor (-2.3% lossless vs. -3%  $q=8$ ) and vary across the sequences. This is also supported by our belief that the high bitrate case (and therefore high quality / high level of immersion) is the most important scenario for immersive video. Finally, it is an interesting outcome that quantization of the depth ranges is of minor impact, as it allows us to maintain a great advantage of B-DSDE: no tuning of any quantization level is required. Note, that in ESDE as well as in H-DSDE, the selection of the quantization parameter ( $QP_D$ ) is a major challenge. Consequently, FD-DSDE preserves this advantage of B-DSDE. In the following sections, quantization is no longer considered and the features are coded losslessly.

## 5.4 Coding Efficiency of FD-DSDE

In this section, the performance of the FD-DSDE system is evaluated by extracting the features from uncoded DERS8 depth maps present at the encoder-side. In that case, the decoder-side performance and the accuracy of the features are limited by the quality of the encoder-side depth maps. Consequently, erroneous blocks of the encoder-side depth maps may even attenuate the quality.

Table 5.2 – Evaluation of different quantization levels for the depth range quantization. The comparison is done with B-DSDE.

Quantization level	Metric	High bitrate	Low bitrate
q=1 (lossless)	video BD-Rate [%]	4.6	8.8
	synth BD-Rate [%]	-14.6	-2.3
	synth PSNR diff [dB]	-0.34	-0.40
	CVR (t=0)		-83.4
q=2	video BD-Rate [%]	4.0	7.7
	synth BD-Rate [%]	-10.1	-1.0
	synth PSNR diff [dB]	-0.29	-0.35
	CVR (t=0)		-82.1
q=4	video BD-Rate [%]	3.4	6.6
	synth BD-Rate [%]	-10.3	-2.1
	synth PSNR diff [dB]	-0.28	-0.34
	CVR (t=0)		-80.3
q=8	video BD-Rate [%]	3.0	5.7
	synth BD-Rate [%]	-10.8	-3.0
	synth PSNR diff [dB]	-0.27	-0.33
	CVR (t=0)		-78.0

### 5.4.1 Depth maps quality

First, we investigate the impact of FD-DSDE on the depth maps quality over all bitrates using the Kitchen sequence as an example. Fig. 5.6 shows the uncoded DERS8 estimated depth map, which is used by the feature extractor as a reference. Fig. 5.7 shows the corresponding decoder-side estimated depth map in comparison to the B-DSDE depth maps over all rate points. We observe a significantly attenuated degradation of depth maps quality with decreasing bitrate. Furthermore, the integrity of the structures in the depth maps are much better retained, especially at low bitrates. Most "holes" in the depth maps are filled. This shows that for every rate point the features can be adapted to minimize the L2 distance towards the reference depth, leading to a more robust and performant system. As there is no quantization applied to the features, the improvement is guaranteed even for low bitrates. Additional comparisons between B-DSDE and FD-DSDE are shown in Fig. 5.8 and Fig. 5.9 as well as in Fig. 5.10 and Fig. 5.11 for the highest and lowest bitrates respectively. The benefit of sending adapted smoothing coefficients is especially visible in the checkerboard of the UnicornA sequence, in which the "holes" are mostly filled in FD-DSDE. However, we observe that this is not the case for UnicornB. This can be easily explained by referring to the uncoded DERS8 depth maps, shown in Fig. 3.2 in Chapter 3. The "holes" are already present in the uncoded ders8 depth maps and consequently, they will be carried over to the FD-DSDE depth maps. In fact, more "holes" appeared in the checkerboard of the UnicornB sequence, which is consistent with the uncoded reference depth maps. While this may seem at first disappointing, it is actually a wanted and encouraging result, as it shows a strong relationship between the encoder-side features and the decoder-side depth maps quality. Naturally, reference depth maps of low quality will translate in a worse performance. The results in the low bitrate range are particularly conclusive, as the level of degradation is significantly reduced for all sequences. However, some level of "blockiness" is retained, originating from the block-based approach. This blockiness is especially visible in the Chef2 sequence. However, the performance has to be evaluated in terms of view synthesis quality.





Figure 5.6 – Uncoded DERS8 estimated Kitchen depth map.

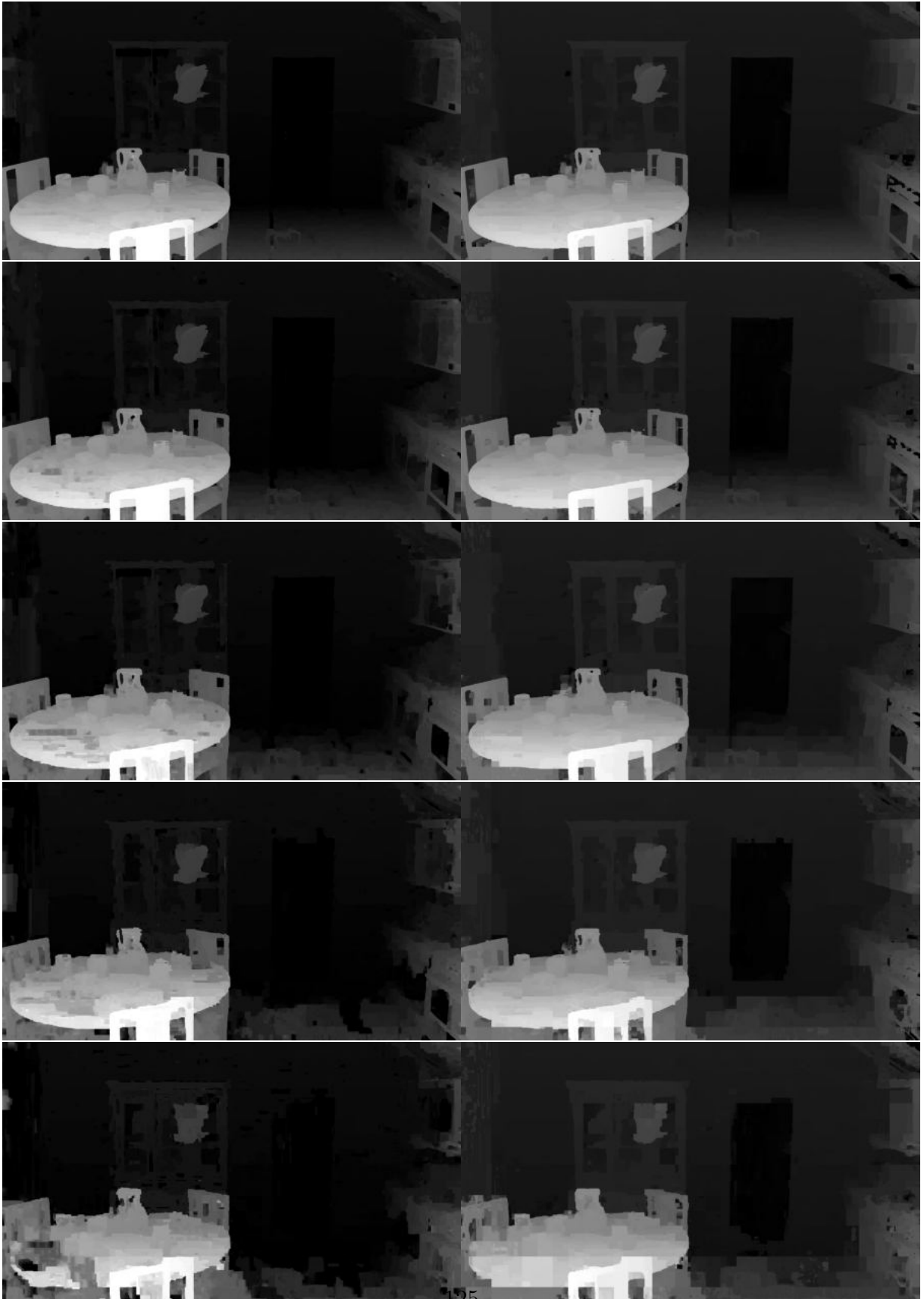


Figure 5.7 – Left: B-DSDE depth maps. Right: FD-DSDE depth maps.



Figure 5.8 – B-DSDE (left) and FD-DSDE (right) for high bitrates using DERS8 as reference (1/2).

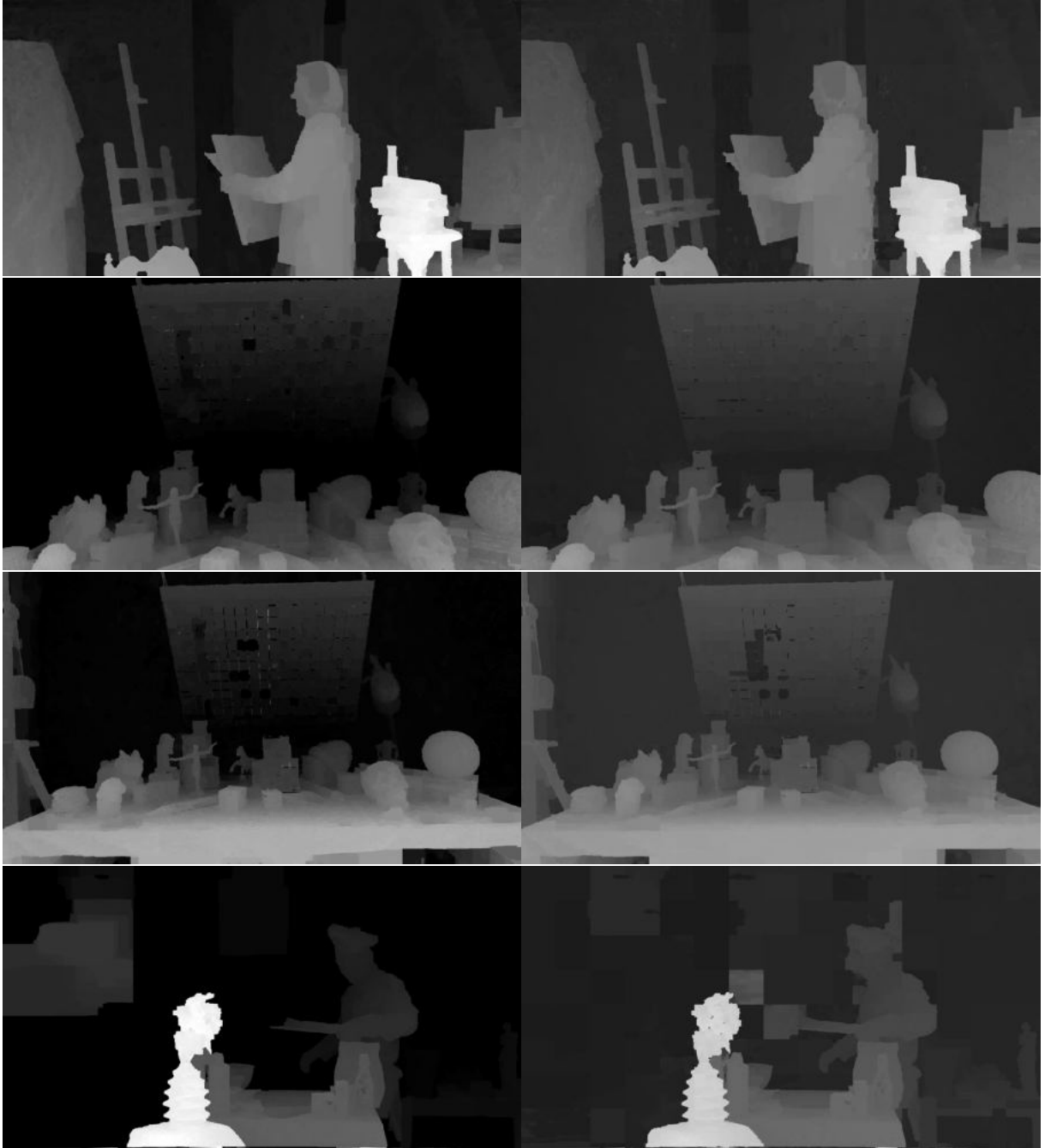


Figure 5.9 – B-DSDE (left) and FD-DSDE (right) for high bitrates using DERS8 as reference (2/2).



Figure 5.10 – B-DSDE (left) and FD-DSDE (right) for low bitrates using DERS8 as reference (1/2).

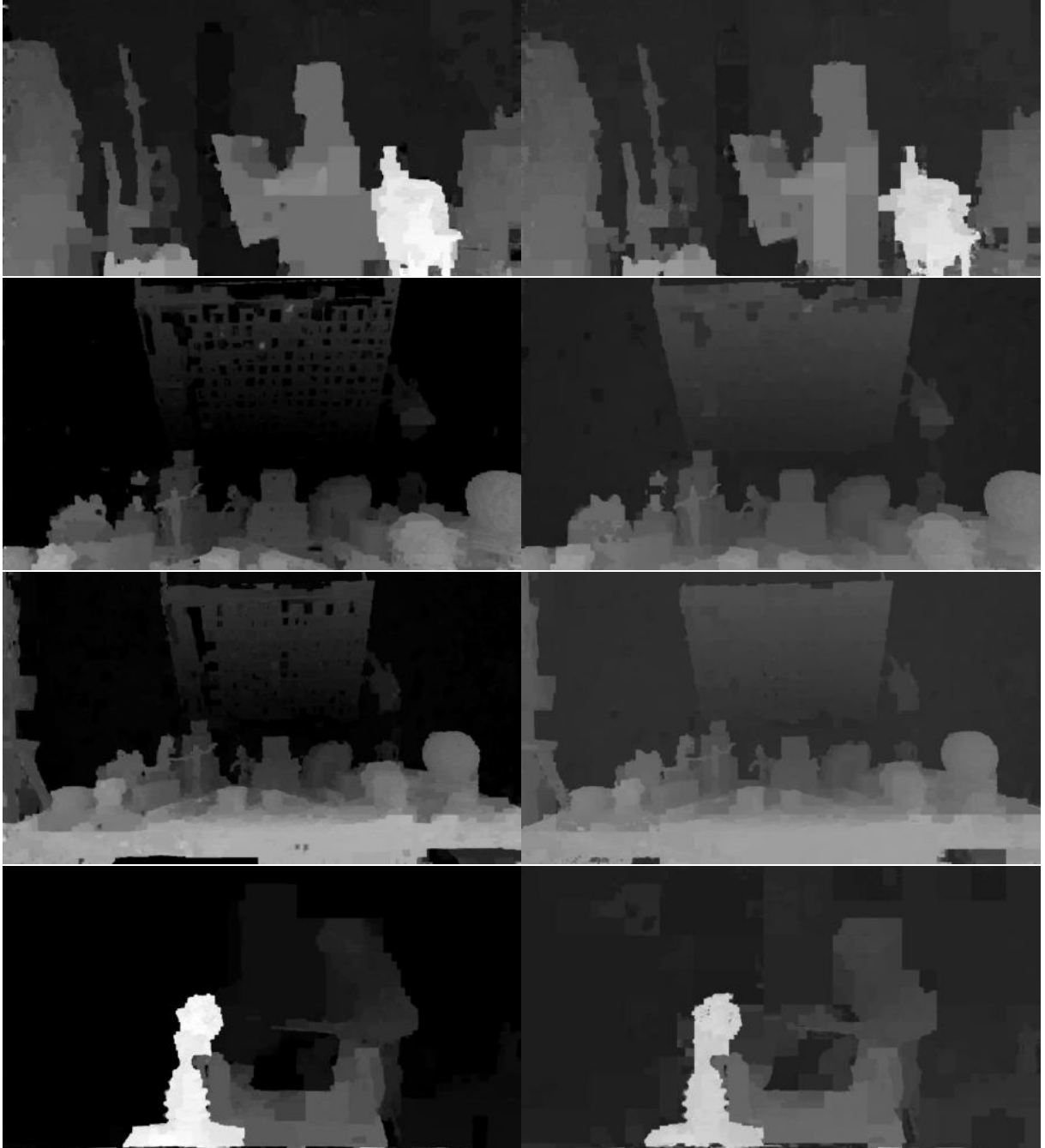


Figure 5.11 – B-DSDE (left) and FD-DSDE (right) for low bitrates using DERS8 as reference (2/2).

### 5.4.2 Synthesis Quality

The view synthesis performance per rate point is visualized in Fig. 5.12, corresponding to the depth maps of Kitchen shown in Fig. 5.7. By comparing the B-DSDE and FD-DSDE synthesis we observe, that the reduction of degradation in the depth maps directly translates into a perceptually improved view synthesis quality. This is particularly well visible in the low bitrate ranges, as all artifacts, *e.g.* around the table, are significantly attenuated. Additional results are shown in Fig. 5.13 and Fig. 5.14 as well as in Fig. 5.15 and Fig. 5.16 for the highest and lowest bitrates respectively. The objective quality with FD-DSDE is improved further compared with B-DSDE and H-DSDE. PSNR gains increase on average by 0.63 dB and 0.62 dB for medium and low bitrate respectively. Similar to H-DSDE, the benefit of the method heavily depends on the quality of the depth map at the encoder side. In the case of the Chef2 sequence, we observe that neither of our extensions led to a benefit over B-DSDE, which indicates that the encoder side depth maps did not contain relevant additional information compared to the B-DSDE depth maps. However, the Dancing sequence increases in PSNR by 1.91 dB, surpassing B-DSDE and H-DSDE. For the majority of sequences, the LPIPS metric is better with FD-DSDE, showing that the synthesis quality is improved.



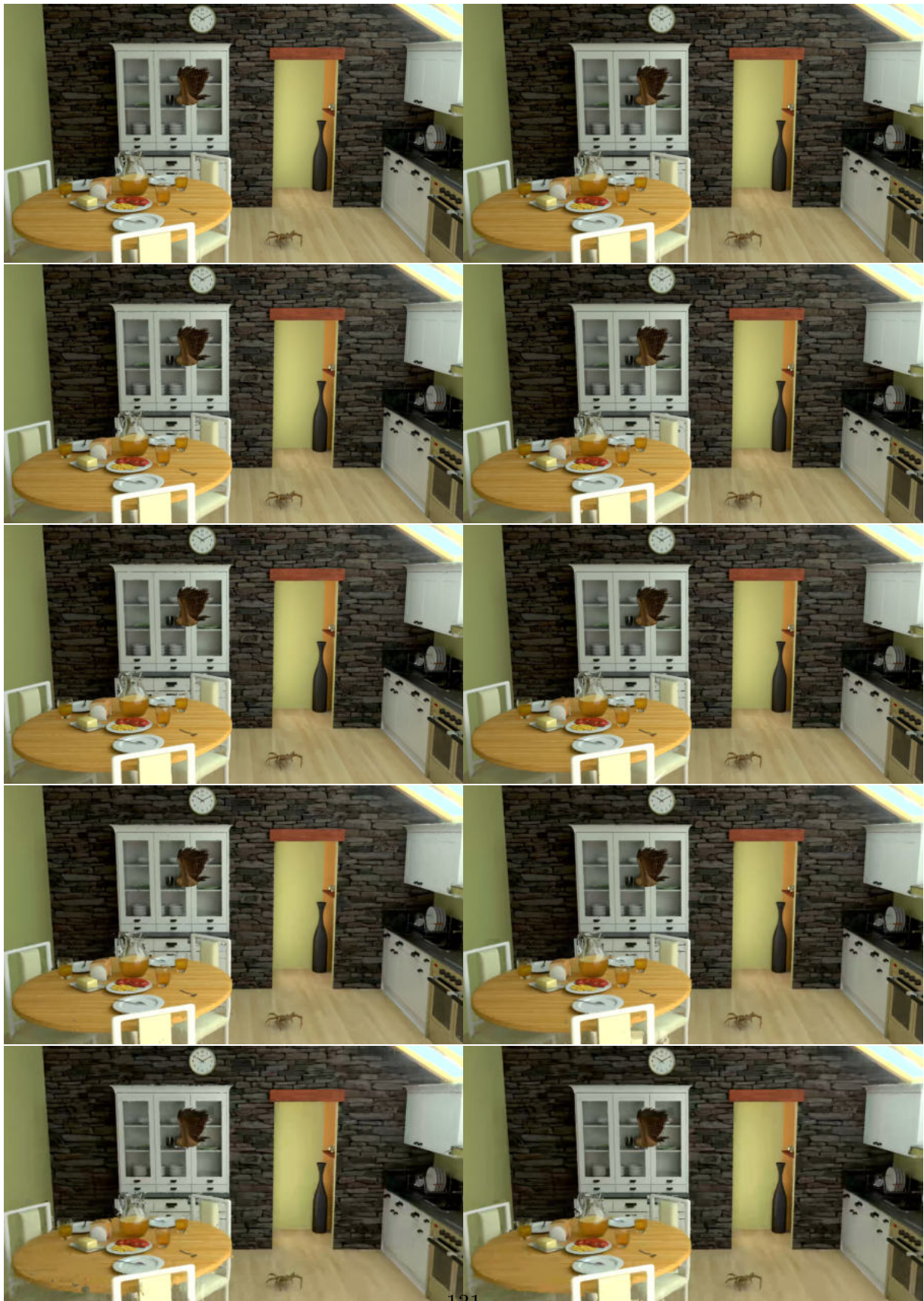


Figure 5.12 – Synthesis performance of Kitchen over all rate points. Left: B-DSDE synthesis. Right: FD-DSDE synthesis using DERS8 depth maps as reference.





Figure 5.13 – B-DSDE (left) and FD-DSDE (right) for high bitrates using DERS8 as reference (1/2).



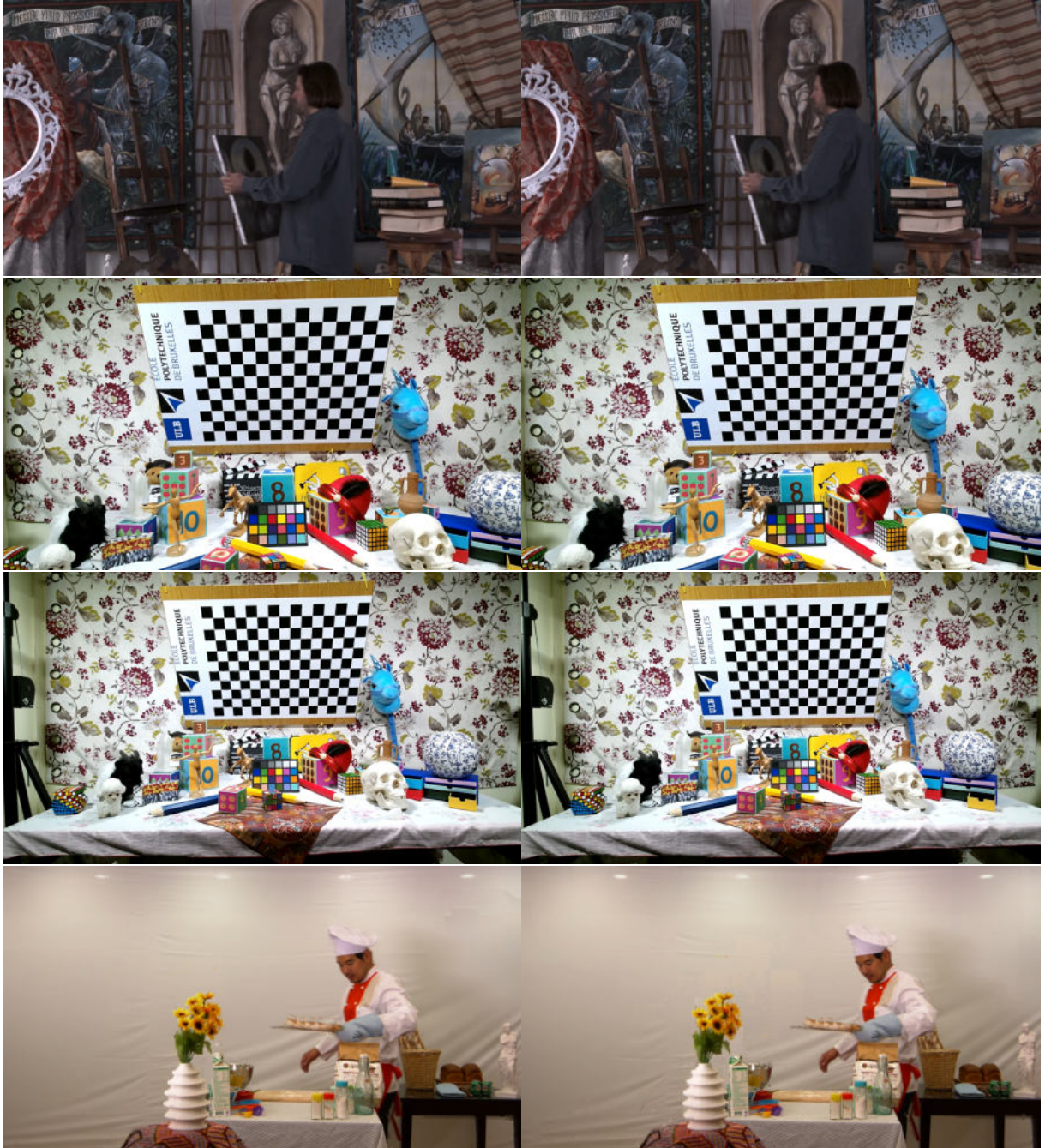


Figure 5.14 – B-DSDE (left) and FD-DSDE (right) for high bitrates using DERS8 as reference (2/2).





Figure 5.15 – Synthesis quality of B-DSDE (left) and FD-DSDE (right) for low bitrates using DERS8 as reference (1/2).



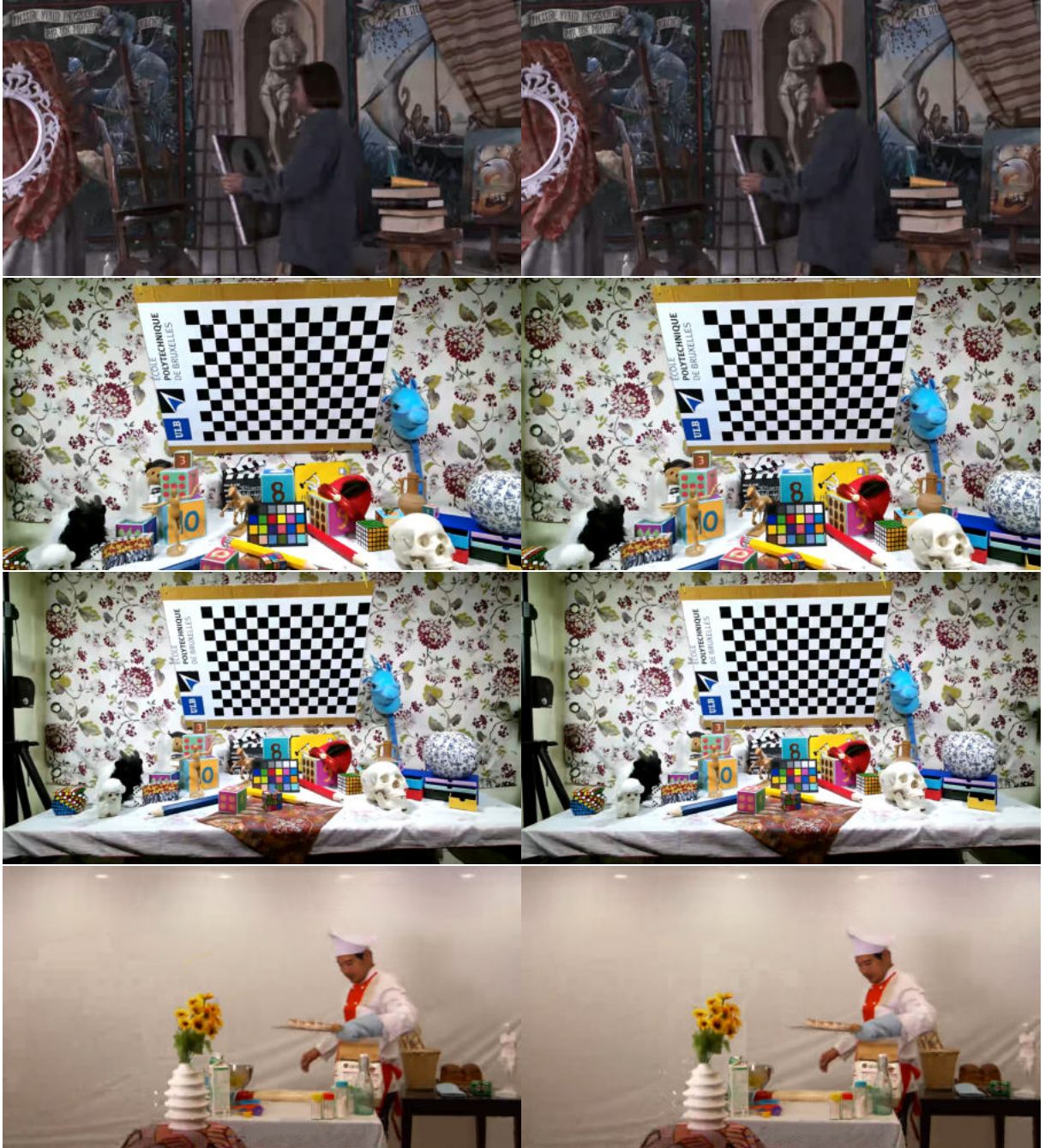


Figure 5.16 – Synthesis quality of B-DSDE (left) and FD-DSDE (right) for low bitrates using DERS8 as reference (2/2).

Table 5.3 – Summary of all BD-Rate objective metrics for FD-DSDE compared to ESDE. The best synthesis performance is indicated in bold. In addition, PSNR and LPIPS are provided. The value in brackets indicate the difference compared to ESDE.

Config	Sequence	video BD-Rate [%]	synth PSNR BD-Rate [%]	synth MS-SSIM BD-Rate [%]	synth VMAF BD-Rate [%]	synth PSNR diff [dB]	synth LPIPS diff
Medium Bitrate	Painter	-29.3	-25.0	-23.4	-28.6	34.25 (-0.13)	0.218 (-0.002)
	UnicornA	-3.8	-28.5	-13.2	-22.4	29.92 (+0.34)	0.056 (-0.003)
	UnicornB	-4.6	-40.4	-14.5	-35.9	30.42 (+0.54)	0.055 (-0.003)
	Shaman	-28.1	-52.1	-36.8	-50.8	34.24 (+0.61)	0.250 (-0.005)
	Kitchen	-14.8	-51.5	-28.6	-52.1	31.45 (+0.90)	0.169 (-0.018)
	Dancing	-2.7	-75.0	-49.0	-73.8	30.06 (+1.91)	0.191 (-0.027)
	Chef2	-22.8	-42.2	-32.8	-47.5	31.70 (+0.20)	0.257 (-0.003)
	Frog	-5.8	-53.1	-28.6	-48.2	27.53 (+0.58)	0.215 (-0.014)
	Average	-14.0	-46.0	-28.3	-44.9	31.20 (+0.63)	0.176 (-0.009)
Low Bitrate	Painter	-27.6	-23.6	-21.5	-27.7	32.80 (-0.16)	0.332 (-0.002)
	UnicornA	-3.4	-23.1	-10.6	-18.7	28.67 (+0.46)	0.115 (-0.008)
	UnicornB	-4.7	-26.5	-10.0	-25.2	29.12 (+0.56)	0.118 (-0.005)
	Shaman	-30.4	-44.8	-33.1	-47.3	32.93 (+0.47)	0.423 (-0.007)
	Kitchen	-16.0	-43.6	-25.4	-45.1	30.27 (+0.85)	0.308 (-0.016)
	Dancing	-2.5	-68.0	-40.0	-64.2	28.88 (+1.86)	0.356 (-0.030)
	Chef2	-22.6	-40.0	-23.4	-36.7	31.13 (+0.31)	0.319 (-0.004)
	Frog	-3.7	-34.0	-16.1	-31.3	26.60 (+0.61)	0.346 (-0.017)
	Average	-13.9	-37.9	-22.5	-37.0	30.05 (+0.62)	0.290 (-0.011)

Table 5.4 – Average runtime per frame and view using the unmodified DERS8 and FD-DSDE.

Sequence	DERS8 [s]	FD-DSDE [s]	speedup
Painter	698.3	111.0	6.3
UnicornA	291.4	40.4	7.2
UnicornB	270.5	35.2	7.7
Shaman	3933.9	169.1	23.3
Kitchen	1051.5	22.6	46.5
Dancing	1195.0	30.2	39.6
Chef2	2565.1	228.8	11.2
Frog	1107.3	177.6	6.2
<b>Average</b>	<b>1389.1</b>	<b>101.85</b>	<b>18.5</b>

### 5.4.3 Coding Efficiency of FD-DSDE

Tab. 5.3 summarizes the BD-Rate values of FD-DSDE in comparison to ESDE. Overall, we observe an average synth BD-Rate reduction of 46% and 37.9% showing a clear benefit over both B-DSDE and H-DSDE. In contrast to ESDE and H-DSDE, FD-DSDE efficiently transmits the relevant depth information in terms of features to the client side. The MS-SSIM and VMAF metrics confirm significant improvements compared to ESDE, with equivalent performance compared to B-DSDE.

### 5.4.4 Complexity Reduction

One of the motivations for designing the FD-DSDE system was the reduction of complexity, as it is the main concern of B-DSDE. The pixel rate and cost volume reduction is shown in Tab. 5.5. Since no depth is transmitted, the pixel rate reduction is equivalent to B-DSDE, *i.e.*  $-50\%$ . Therefore, one of the main advantages of B-DSDE is maintained. The cost volume is reduced by around  $-88\%$  on average. That means, that due to the adapted depth ranges and the skip flag in combination with a block-based approach, the far majority of depth candidates do not need to be tested. This translates into a speedup of 18.5 on average as shown in Tab. 5.4, emphasizing that FD-DSDE can provide a significant contribution in making decoder side depth estimation feasible in terms of complexity. Furthermore, the block based approach is highly parallelizable, as every block can be processed independently. In our experiments, we did not take advantage of this aspect. However, a speedup of several factors can be expected.

Table 5.5 – Pixel Rate and Cost Volume Reduction FD-DSDE.

Config	Sequence	PRR [%]	CVR [%]
Medium Bitrate	Painter	-50.0	-83.6
	UnicornA	-50.0	-76.8
	UnicornB	-50.0	-84.0
	Shaman	-50.0	-94.8
	Kitchen	-50.0	-97.7
	Dancing	-50.0	-97.3
	Chef2	-50.0	-88.8
	Frog	-50.0	-82.4
	<b>Average</b>	<b>-50.0</b>	<b>-88.3</b>
Low Bitrate	Painter	-50.0	-83.8
	UnicornA	-50.0	-77.4
	UnicornB	-50.0	-84.4
	Shaman	-50.0	-95.0
	Kitchen	-50.0	-97.9
	Dancing	-50.0	-97.3
	Chef2	-50.0	-88.7
	Frog	-50.0	-82.4
	<b>Average</b>	<b>-50.0</b>	<b>-88.4</b>

## 5.5 FD-DSDE with high quality depth maps

Up to this point, all depth maps used in our experiments originated from the same algorithm (DERS8). In this way we show that the changes in performance of all proposed systems do not come from changes in the depth estimation algorithm. Yet, in practice, one would use the best depth maps that are available. Due to the high potential of FD-DSDE, shown in the last sections, we evaluate it once again using high quality blender depth for CGI content. Tab. 5.6 shows the results of all objective metrics, where the Kitchen, Shaman and Dancing sequences have been re-computed using high-quality depth maps as a reference. Synth BD-Rate is further improved compared to the FD-DSDE results with DERS8-reference. The PSNR of the Dancing sequence has improved by 2.86 dB on average, which is 1 dB more than B-DSDE and 2 dB more than H-DSDE, while simultaneously keeping the bitrate low. This further underlines that FD-DSDE is a more efficient way of taking advantage from high quality depth maps by encoding the relevant geometry information into features.

## 5.6 Comparison of FD-DSDE with 3D-HEVC

The previous section reports experimental results using MPEG-I Visual CTC. As clarified, MPEG-I Visual chose to facilitate the adoption of its standard by considering existing 2D video codec for depth and textures, thus preventing the use of specific depth coding and interview prediction tools. However, since MIV is the spiritual successor of 3D-HEVC, there seems to be a regular interest to request a comparison with 3D-HEVC when proposing improvements to MIV. Judging a proposal for a video-based system based on a comparison with a geometry-based system, which may be at a different stage of development, is not fruitful. Nevertheless, in this subsection, we provide a comparison of FD-DSDE with 3D-HEVC for informative reasons. As there are inherent limitations of the 3D-HEVC standard and its reference software that prevent us from setting it into a configuration that matches the MPEG-I Visual CTC, we chose to use the 3D-HEVC test conditions as described in [60] and set the configuration of FD-DSDE accordingly: in particular, we limit ourselves to a 3 view configuration, using the first three horizontal views of the MPEG-I Visual test sequences. The results are summarized in Tab. 5.7 and Fig. 5.26. Naturally, this experiment is in favor of 3D-HEVC, as the 3-View configuration is designed for this scenario. It is therefore inconclusive when it comes to 6DoF immersive video. Since 3 views are used, the anchor has to be re-computed. Depth estimation is re-performed on 3 views, leading to novel uncoded depth maps. To be consistent with our previous section, the MV-HEVC anchor in 3-view configuration is set as an anchor for the video-based ESDE approach. In total, the following experiments were re-computed given 3 views per sequence:

1. Depth estimation from uncoded textures.
2. 3D-HEVC-coded texture+depth.
3. Simulcast MV-HEVC-coded texture+depth.
4. MV-HEVC coded texture + decoder-side estimated depth.
5. MV-HEVC coded texture + feature-driven decoder side estimated depth.

As the FD-DSDE results are the most interesting, we will omit the presentation of the B-DSDE and video-based ESDE results. Instead, we compare the geometry-based ESDE (full 3D-HEVC) with FD-DSDE, with textures coded by MV-HEVC. Numerical results are provided for all experiments.



Table 5.6 – Objective metrics, where features are extracted from high quality depth maps created with blender, which are available for the CGI sequences (bold). The value in brackets indicate the change to the results of Tab. 5.3 and ??.

Config	Reference Depth	Sequence	video BD-Rate [%]	synth PSNR BD-Rate [%]	synth MS-SSIM BD-Rate [%]	synth VMAF BD-Rate [%]	synth PSNR [dB]	synth LPIPS
Medium Bitrate	DERS8	Painter	-29.3	-25.0	-23.4	-28.6	34.25	0.218
		UnicornA	-3.8	-28.5	-13.2	-22.4	29.92	0.054
		UnicornB	-4.7	-40.4	-14.5	-35.9	30.42	0.055
		Chef2	-22.8	-42.2	-32.8	-47.5	31.70	0.256
		Frog	-5.8	-53.1	-28.6	-48.2	27.53	0.216
	Blender	Shaman	<b>-22.0 (+6.1)</b>	<b>-58.3 (-6.2)</b>	<b>-40.6 (-3.2)</b>	<b>-48.9 (+1.9)</b>	<b>34.91 (+0.67)</b>	<b>0.243 (-0.007)</b>
		Kitchen	<b>-15.2 (-0.4)</b>	<b>-56.7 (-5.2)</b>	<b>-32.6 (-4.0)</b>	<b>-53.7 (-1.6)</b>	<b>31.74 (+0.29)</b>	<b>0.166 (-0.003)</b>
		Dancing	<b>-3.4 (-0.7)</b>	<b>-81.4 (-6.4)</b>	<b>-56.6 (-7.6)</b>	<b>-82.4 (-8.6)</b>	<b>31.03 (+0.97)</b>	<b>0.172 (-0.018)</b>
		<b>Average</b>	<b>-13.4</b>	<b>-48.2</b>	<b>-30.3</b>	<b>-45.9</b>	<b>31.44</b>	<b>0.173</b>
		Painter	-27.6	-23.6	-21.5	-27.7	32.80	0.332
Low Bitrate	DERS8	UnicornA	-3.4	-23.1	-10.6	-18.7	28.67	0.114
		UnicornB	-4.7	-26.5	-10.0	-25.2	29.12	0.118
		Chef2	-22.6	-40.0	-23.4	-36.7	31.13	0.319
		Frog	-3.7	-34.0	-16.1	-31.3	26.60	0.345
		Shaman	<b>-18.7 (+11.7)</b>	<b>-40.1 (+4.7)</b>	<b>-24.5 (+8.6)</b>	<b>-36.3 (+11.0)</b>	<b>33.36 (+0.43)</b>	<b>0.423 (-0.000)</b>
	Blender	Kitchen	<b>-16.8 (-0.8)</b>	<b>-47.7 (-4.1)</b>	<b>-28.2 (-2.8)</b>	<b>-46.4 (-1.3)</b>	<b>30.46 (+0.19)</b>	<b>0.306 (-0.002)</b>
		Dancing	<b>-4.1 (-1.6)</b>	<b>-73.6 (-5.6)</b>	<b>-46.2 (-6.2)</b>	<b>-72.0 (-7.8)</b>	<b>29.51 (+0.63)</b>	<b>0.346 (-0.010)</b>
<b>Average</b>	<b>-11.9</b>	<b>-38.6</b>	<b>-22.6</b>	<b>-36.8</b>	<b>30.21</b>	<b>0.288</b>		

### 5.6.1 Experimental results

Fig. 5.17 summarizes the DERS8 depth maps quality for 3 input textures. The center view is shown and therefore, may contain less artifacts than the boundary views. Naturally, the performance of 3D-HEVC will be impacted by the quality of these depth maps. However, the same applies for the corresponding B-DSDE and FD-DSDE results.

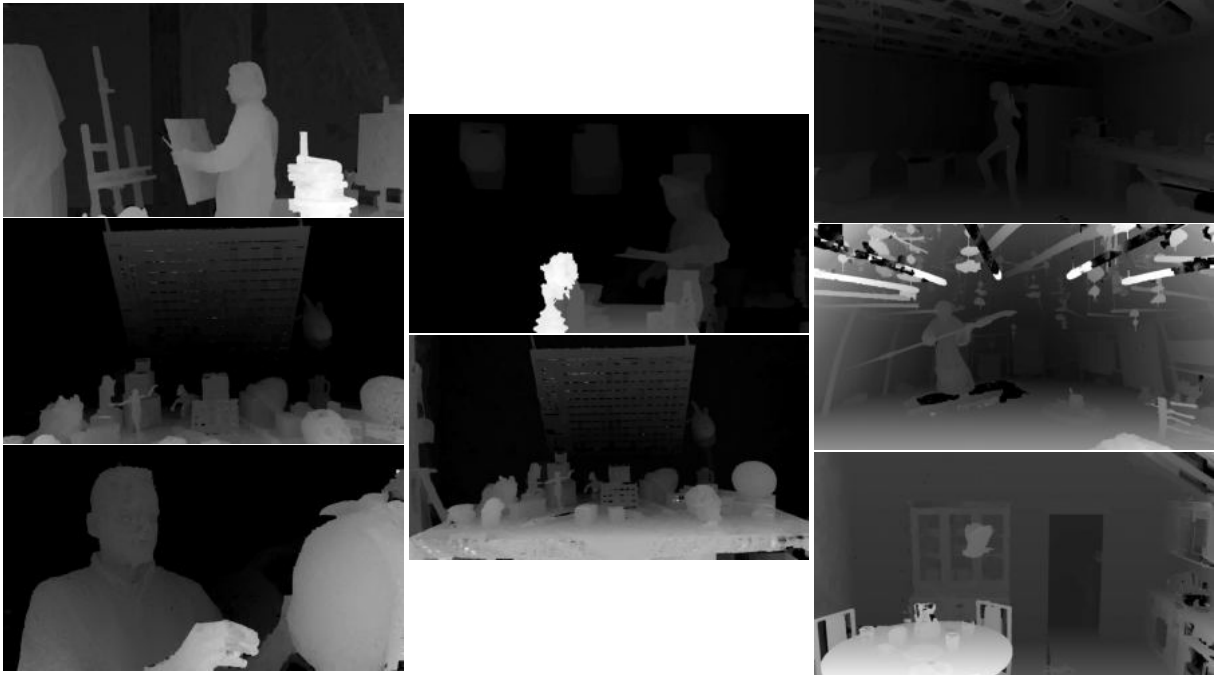


Figure 5.17 – Depth maps estimated by DERS8 using 3 source textures for 3D-HEVC experiments.. Top row: Painter, Fencing, Dancing. Mid row: Chef2, UnicornA, Shaman. Bottom row: Frog, UnicornB, Kitchen.

As expected, 3D-HEVC outperforms MVD compression with MV-HEVC simulcast. In many cases B-DSDE can still outperform 3D-HEVC. Since depth maps are estimated from 3 views only, FD-DSDE may perform worse, as the features are extracted from depth maps with more artifacts. We show the RD curves in Fig. 5.26.

#### Depth Quality of FD-DSDE against 3D-HEVC

3D-HEVC and FD-DSDE depth maps are shown in Figs. 5.18- 5.21. As expected, 3D-HEVC preserves the object boundaries much better than MV-HEVC. Especially at low bitrates, the superiority in depth preservation is visible.



Figure 5.18 – 3D-HEVC based ESDE (left) and FD-DSDE (right) for high bitrates (1/2).

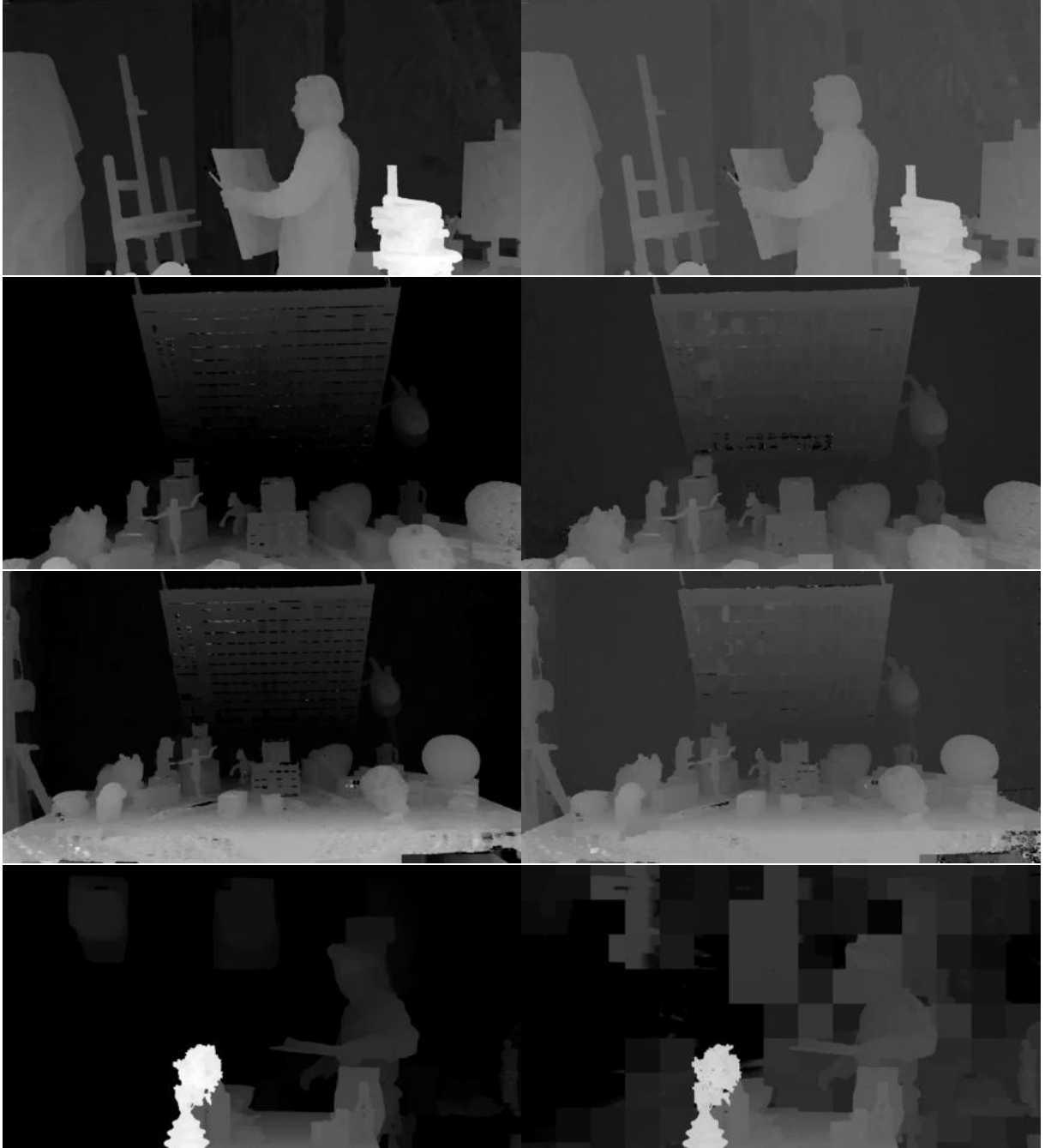


Figure 5.19 – 3D-HEVC based ESDE (left) and FD-DSDE (right) for high bitrates (2/2).



Figure 5.20 – 3D-HEVC based ESDE (left) and FD-DSDE (right) for low bitrates (1/2).

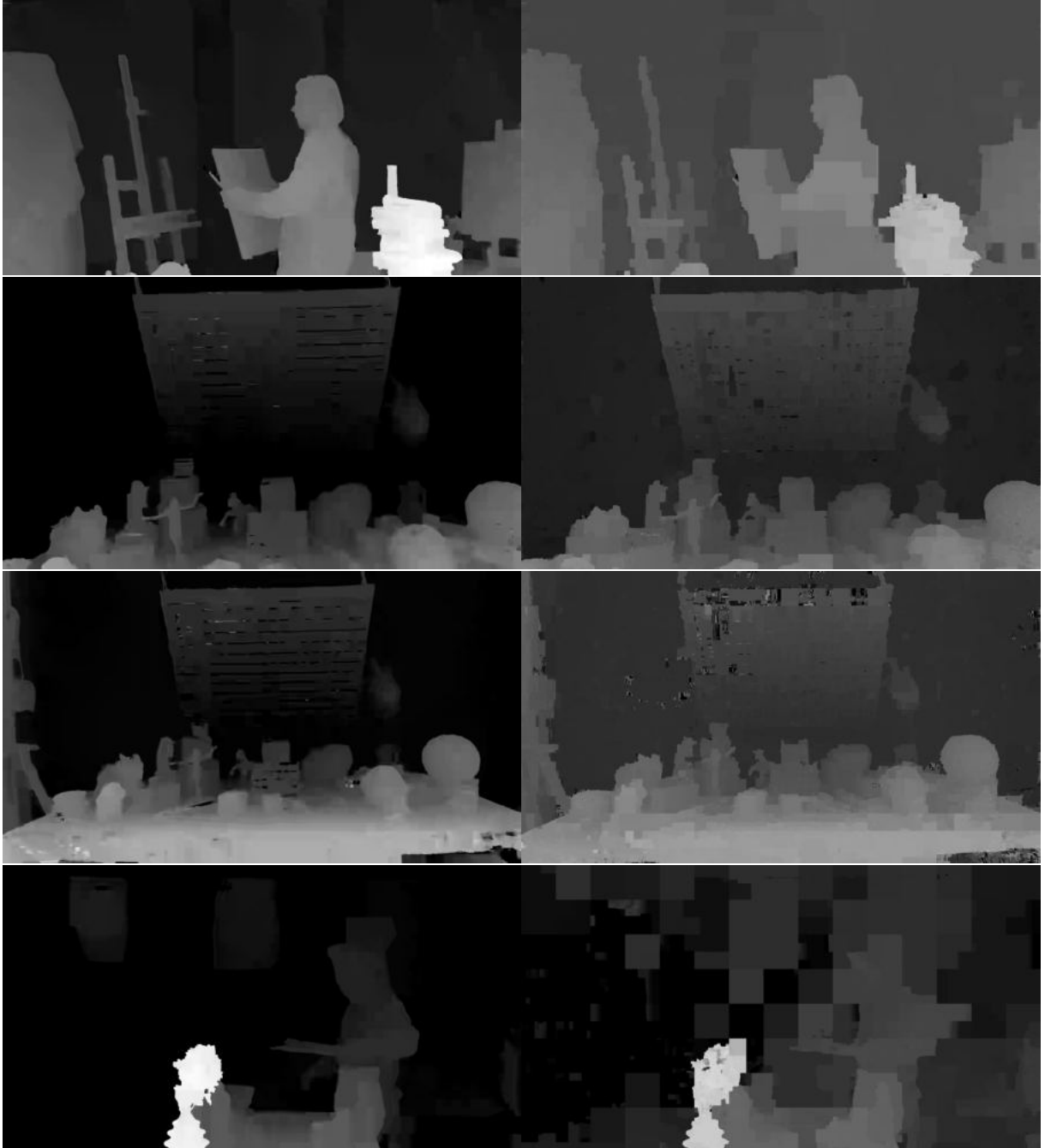


Figure 5.21 – 3D-HEVC based ESDE (left) and FD-DSDE (right) for low bitrates (2/2).

### View Synthesis Quality of 3D-HEVC against FD-DSDE

The visual quality of the synthesized views is shown in Figs. 5.22-5.25.



Figure 5.22 – Synthesis quality for 3D-HEVC based ESDE (left) and FD-DSDE (right) for high bitrates (1/2).



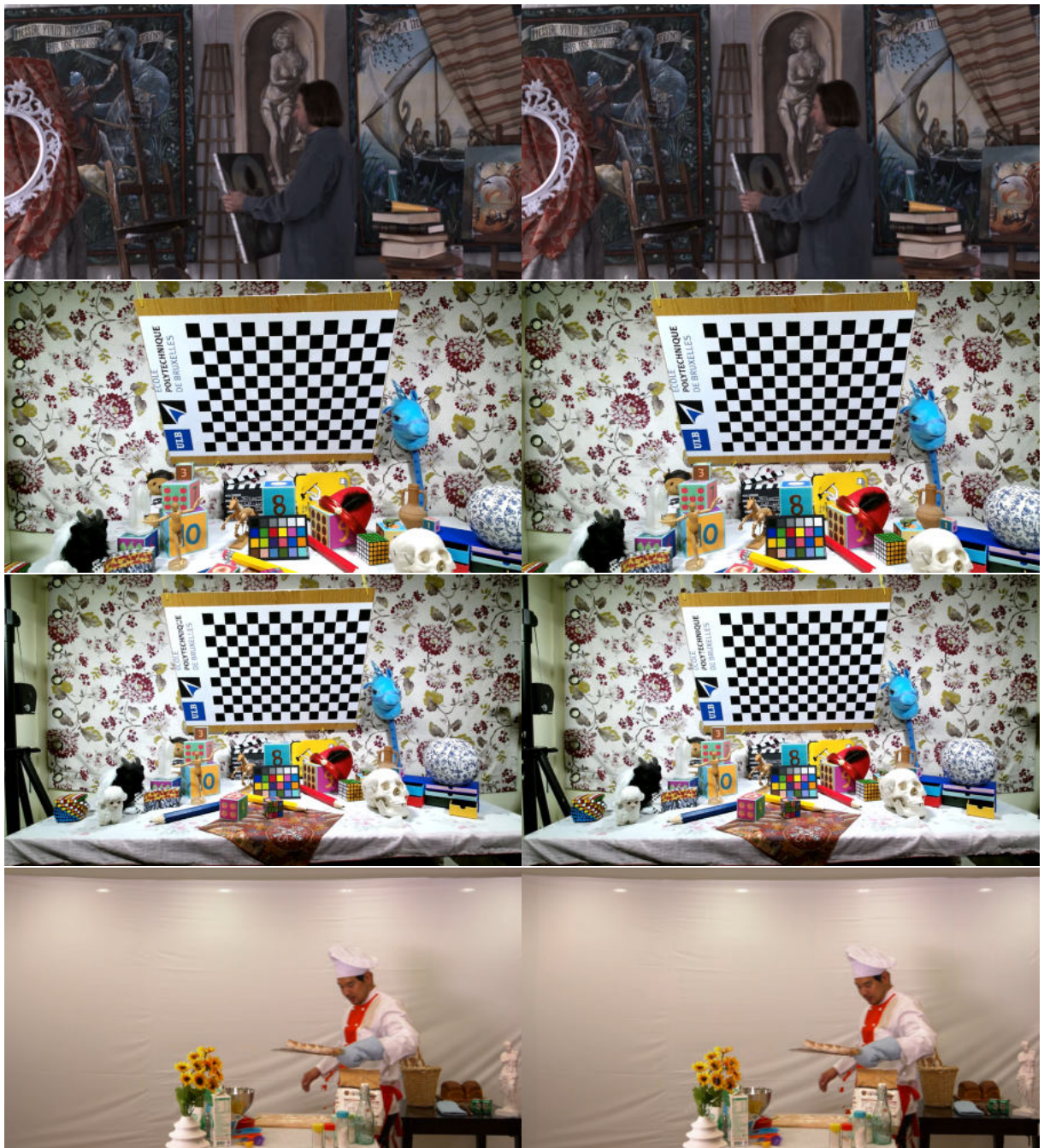


Figure 5.23 – Synthesis quality for 3D-HEVC based ESDE (left) and FD-DSDE (right) for high bitrates (2/2).



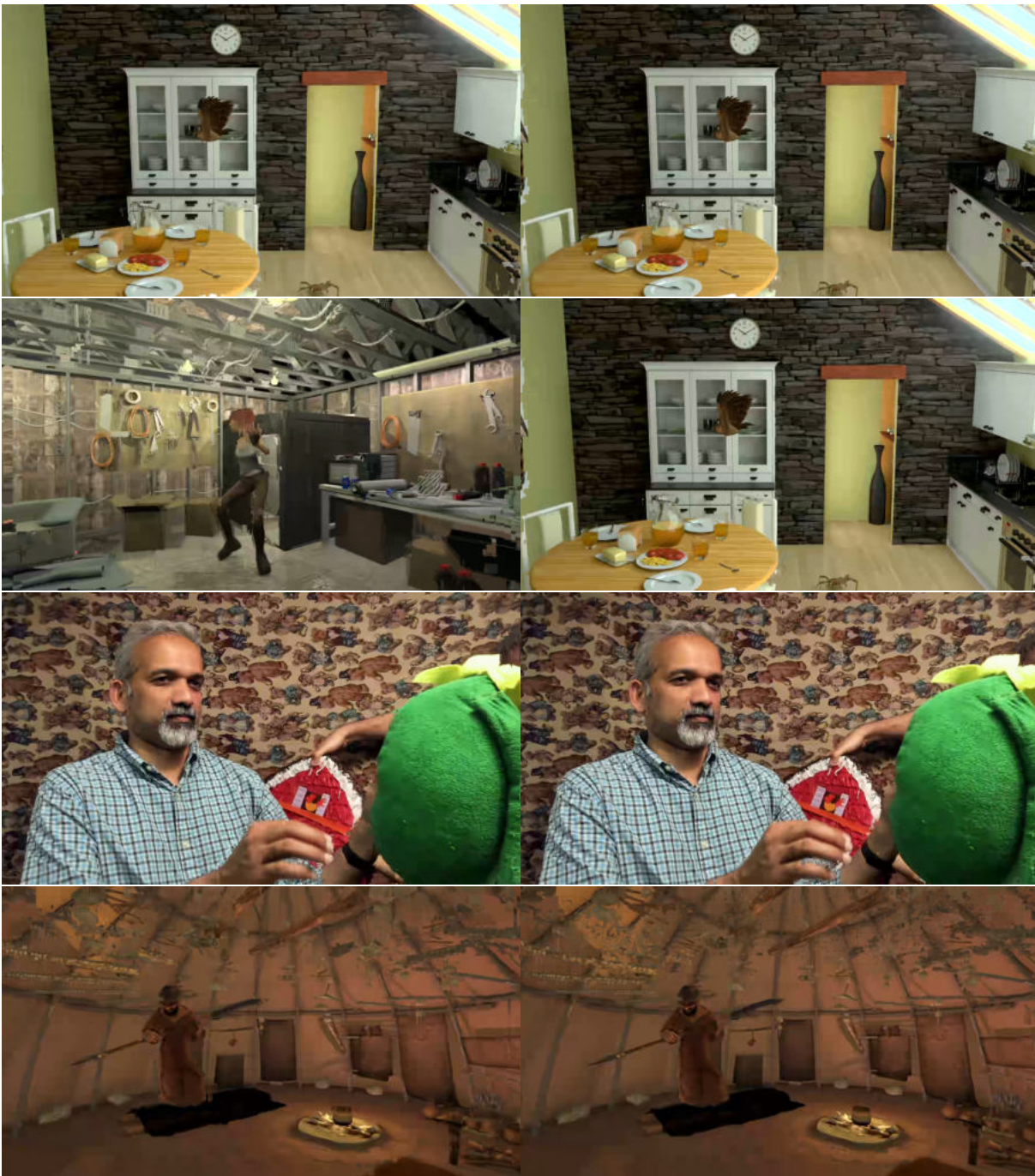


Figure 5.24 – Synthesis quality for 3D-HEVC based ESDE (left) and B-DSDE (right) for low bitrates (1/2).

### Coding Efficiency of FD-DSDE against 3D-HEVC

The synthesis PSNR BD-Rate is shown in Tab. 5.7. We observe, that on average, 3D-HEVC and B-DSDE perform very similar, which is surprising considering the years of



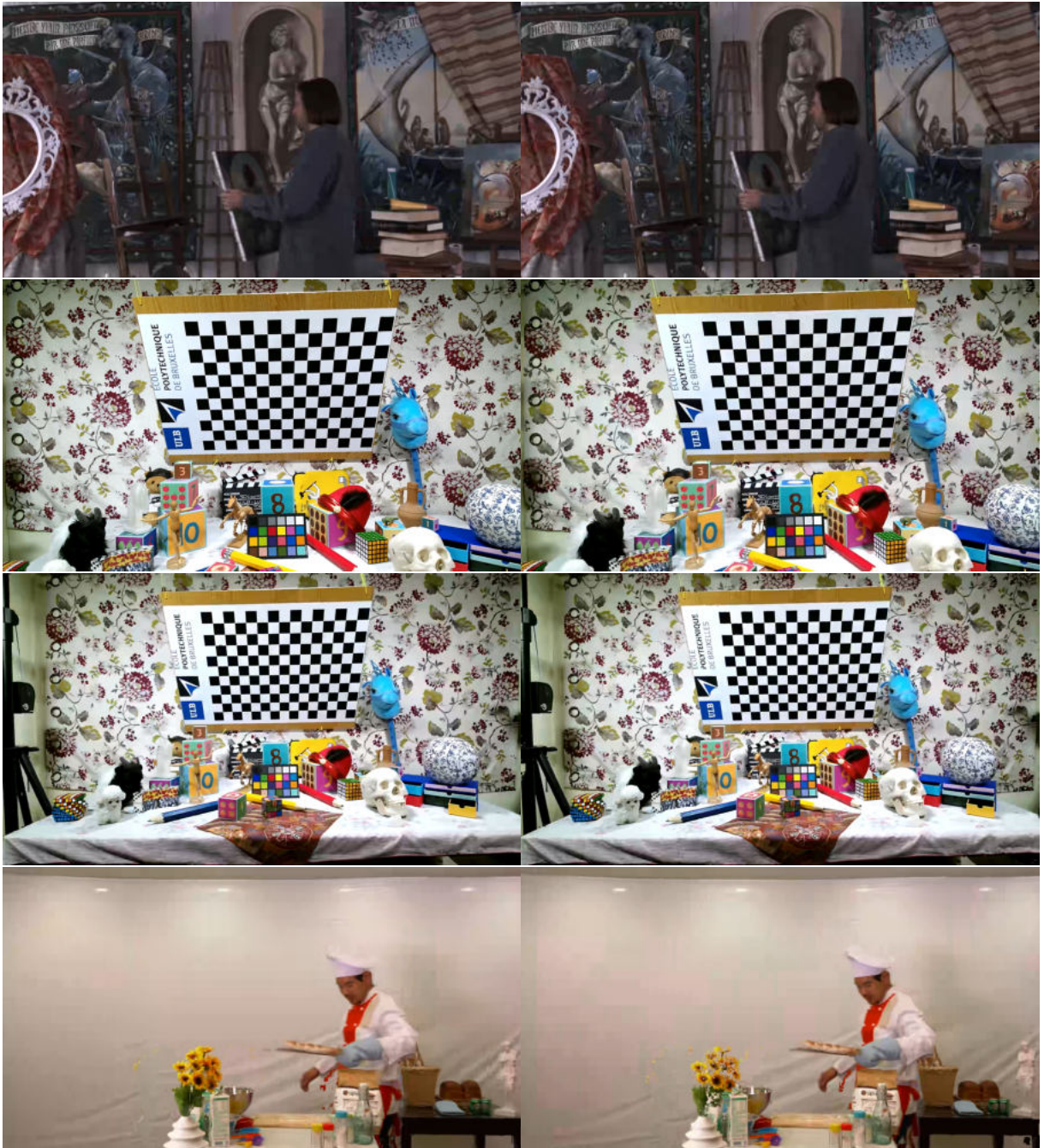


Figure 5.25 – Synthesis quality for 3D-HEVC based ESDE (left) and B-DSDE (right) for low bitrates (2/2).

research behind 3D-HEVC in comparison to the simplicity of B-DSDE. Furthermore, 3D-HEVC seems to be more robust towards erroneous depth maps, while FD-DSDE suffers more from the degraded features, extracted from the 3-view configuration-based depth

Table 5.7 – Synthesis PSNR BD-Rates using the 3-View configuration of the 3D-HEVC CTC. FD-DSDE average (\*) is underestimated due to missing overlaps for Frog and Dancing. Instead, their RD-curves are shown in Fig. 5.26.

Config	Sequence	synth PSNR BD-Rate [%]		
		3D-HEVC	B-DSDE	FD-DSDE
Medium Bitrate	Painter	-24.3	<b>-36.4</b>	-27.7
	UnicornA	-58.3	-42.3	<b>-64.6</b>
	UnicornB	<b>-59.2</b>	-33.8	-44.1
	Shaman	<b>-59.5</b>	-46.8	-35.9
	Kitchen	<b>-66.4</b>	-61.1	-57.2
	Dancing	-62.2	-83.4	<b>no overlap</b>
	Chef2	-67.1	<b>-82.3</b>	-47.9
	Frog	-62.8	-79.9	<b>no overlap</b>
	Average	-57.5	-58.2	-46.0*
Low Bitrate	Painter	-25.0	<b>-36.5</b>	-27.0
	UnicornA	-39.3	-38.3	<b>-47.5</b>
	UnicornB	<b>-41.6</b>	-37.4	-40.0
	Shaman	<b>-50.6</b>	-25.5	-22.7
	Kitchen	<b>-57.4</b>	-50.2	-53.6
	Dancing	-32.8	-82.2	<b>no overlap</b>
	Chef2	-55.3	<b>-61.5</b>	-40.7
	Frog	-40.5	-58.6	<b>-77.8</b>
	Average	-42.8	-48.8	-44.2*

maps. Fig. 5.26 shows additionally the RD-curves for the Dancing and Frog sequence, for which a BD-Rate value could not be computed.

## 5.7 Conclusion

In this chapter, FD-DSDE has been presented. It is a solution which identifies useful features that support the depth estimator at the decoder side in estimating depth maps faster and with higher accuracy. In that context, a block-based depth estimation strategy has been proposed. The proposal outperforms B-DSDE without re-introducing the disadvantages of ESDE. Therefore, a 50% pixel rate reduction is maintained. Furthermore, a comparison to 3D-HEVC has been presented, proving that B-DSDE and FD-DSDE have the potential to outperform 3D-HEVC in terms of coding efficiency. However, independent of these results, DSDE does not actually compete with 3D-HEVC in practice, since the latter has not been adopted by the industry.

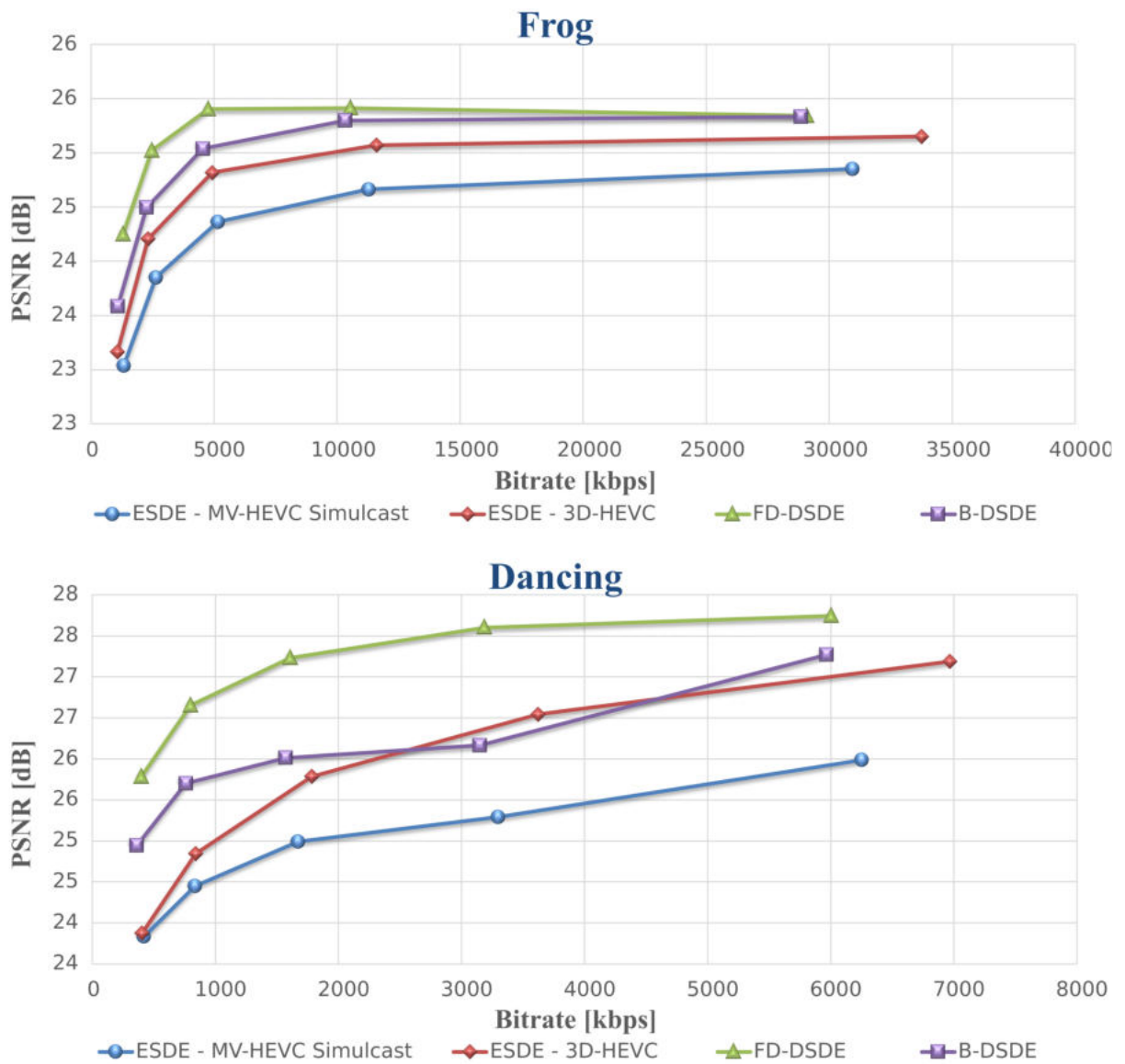


Figure 5.26 – RD-curves for the Dancing and Frog sequences.



# THE GEOMETRY ABSENT PROFILE AND THE GEOMETRY ASSISTANCE SEI MESSAGE OF MPEG IMMERSIVE VIDEO

---

This chapter is a direct continuation of Chapter 2, MPEG Immersive Video, focusing on two features of the standard, which have been adopted as a result of the research conducted in Chapter 3 and 5: the Geometry Absent Profile and the Geometry Assistance SEI message. Both will be utilized in the subsequent chapters 7 and 8.

## 6.1 The Geometry Absent Profile of MPEG Immersive Video

It was shown during the standardization process that the coding gain of the TMIV can be significantly improved, if the depth maps are estimated on the decoder-side using Immersive Video Depth Estimation (IVDE), instead of coding the CTC depth maps [61] [62]. The latter include high-quality depth maps, originating from the blender software ("Ground-Truth") in case of CGI or refined several times, *e.g.* through manual tuning of the software. As a consequence, the B-DSDE concept has been adopted into the MIV standard as the MIV Geometry Absent (GA) Profile. The specification around this profile is designed under the assumption, that depth maps are not present at the encoder-side [63], which has several consequences. First of all, it is not possible to design an anchor, which corresponds to the MIV Anchor in ESDE-mode. The latter utilizes depth information in order to perform the pruning process. This is not possible in the GA Profile. Therefore, the

GA & GA SEI were presented in the IEEE Transactions on Circuits and Systems for Video Technology  
Dawid Mieloch, Patrick Garus et al., "Overview and Efficiency of Decoder-Side Depth Estimation in MPEG Immersive Video," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 9, pp. 6360-6374, Sept. 2022.

anchor related to the GA profile corresponds to the MIV View profile, in which the view labeling is run in mode 2, effectively turning it into a view selector. However, since up to 50% of pixel rate is saved in the GA Profile, up to twice the number of basic views (= basic "textures") can be included into the atlases compared to the MIV View anchor. In addition, all four decoder instantiations are now available for texture decoding. Finally, the encoders will compress the texture atlases as efficient as possible, as they are designed for texture compression. The potential of multiview-prediction is also high in this setup, because more similarity among the views can be expected, given that double the views of the original light field are coded. This scenario is similar to the one covered throughout this thesis, in which MV-HEVC is utilized to code the MVD. The anchor has finally been added to the CTC, denoted as *MIV DSDE anchor* [64].

Modifications to the specification are mostly related to excluding the syntax elements related to geometry coding. A remarkable premise is, however, that the depth estimator remains non-normative, similar to the rendering process. Consequently, the GA Profile of TMIV is compatible with any depth estimator, as long as it does not require any additional information beyond texture and camera parameters. The block diagram of the simplified TMIV Encoder is shown in Fig. 6.1. Simplified means, that any module related to ESDE has been removed (see Chapter 2, Fig. 2.5) [65] [63]. It shall be noted that the encoder in DSDE-mode is significantly simpler and mostly dependent on the view selection process. Despite the simple architecture, the coding gain of the DSDE-mode utilizing the GA Profile is much higher. The corresponding simplified TMIV Decoder,

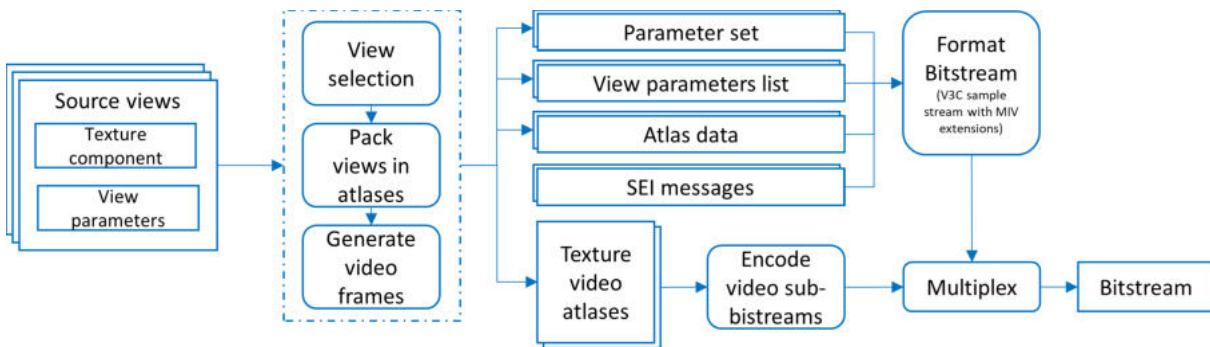


Figure 6.1 – The simplified TMIV Encoder reflecting the DSDE mode using the GA Profile [65].

run in DSDE mode, is shown in Fig. 6.2. It reflects, how the depth-related information is removed from the decoding process and how the depth processing related modules are replaced by a simple depth estimation process ((see Chapter 2, Fig. 2.12) [65]). Needless



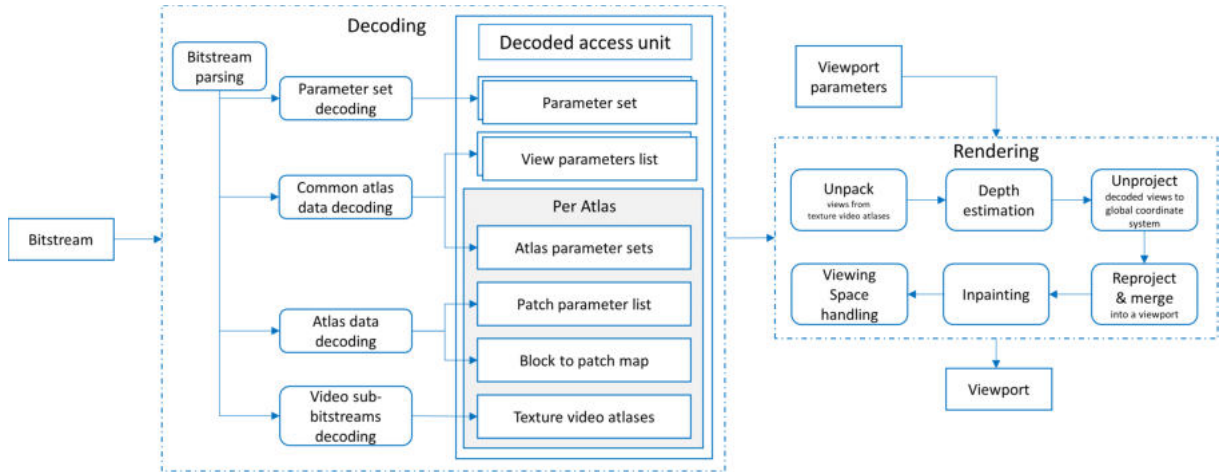


Figure 6.2 – The simplified TMIV Decoder reflecting the DSDE mode using the GA Profile [65].

to say, the price to pay for such a simplified system is the requirement of an accurate and fast depth estimator at the decoder-side and as well as a smart and reliable view selector at the encoder-side.

## 6.2 The Geometry Assistance SEI message of MPEG Immersive Video

One way to support the depth estimator in its challenging task is to move most of the thinking to the encoder-side and provide relevant side information instead. FD-DSDE has shown, that a significant speedup and coding gain can be achieved by providing certain features to the depth estimator at the decoder-side [66]. This concept has been adopted into the MIV Standard as the Geometry Assistance SEI message. It is mostly similar to the proposal described in Chapter 5. Therefore, only the main differences are summarized here.

The feature extractor has been significantly simplified in order to derive the features directly from the depth maps. The derivation process and criteria are explained in for each feature below.



### 6.2.1 Depth range

For a given block partition, the  $Z_{min}$  and  $Z_{max}$  values the block are extracted from the reference depth map, similar to FD-DSDE.

In the MIV Standard, the depth ranges are coded in a predictive manner using already coded values in the neighboring left ( $Z_{min,left}$ ,  $Z_{max,left}$ ) and top ( $Z_{min,top}$ ,  $Z_{max,top}$ ) block indicated by the  $gas\_ltmin\_flag$  and  $gas\_ltmax\_flag$  respectively. For each block, the depth ranges are derived as:

$$Z_{min} = (gas\_ltmin\_flag == 1 ? Z_{min,top} : Z_{min,left} + gas\_qs \times gas\_zmin\_delta) \quad (6.1)$$

$$Z_{max} = (gas\_ltmax\_flag == 1 ? Z_{max,top} : Z_{max,left} + gas\_qs \times gas\_zmax\_delta) \quad (6.2)$$

with  $gas\_qs$  indicating the quantization step, while the differences between the current block and the indicated neighboring blocks depth range are  $gas\_zmax\_delta$  and  $gas\_zmin\_delta$ . The consideration of quantization for the depth range coding is the main difference with FD-DSDE. The reason for that is that additional depth estimation parameters (like smoothing coefficients) are not part of the Geometry Assistance SEI message, see Section below. For each sequence, the quantization step  $gas\_qs$  can be automatically adapted in order to limit the overall bitrate to not exceed  $1Mbps$ . This constraint was set by the group, as it is unusual that a SEI message involves the coding of this big amount of data.

### 6.2.2 Partitioning

The original FD-DSDE, which supports a quadtree partitioning, is extended to rectangular shapes as shown in Fig. 6.3. This codes associated with each partition type are based on the expected occurrence. The latter has been derived through simulations. As can be seen, the quadtree split is the most occurring partitioning type, proving, that the most important case has already been covered by FD-DSDE. It is coded through a single  $gas\_split\_flag$ -flag. Additional signalling is required for less often occurring partition choices.

The feature extractor selects the partitioning based on the computed cost volume reduction if a given partition is utilized. A partitioning is performed, if the amount of









split type \ flag								
gas_split_flag	0	1	1	1	1	1	1	1
gas_quad_split_flag		1	0	0	0	0	0	0
gas_split_orientation_flag			0	0	0	1	1	1
gas_split_symmetry_flag			1	0	0	1	0	0
gas_split_first_block_bigger				0	1		0	1

Figure 6.3 – All supported split types, their corresponding flag description and code, in the Geometry Assistance SEI message.

cost volume reduction is below  $T_{split}$ :

$$\frac{CV_N}{CV_{split}} < T_{split}, \quad (6.3)$$

with the initial cost volume  $CV_N$  of the initial block of size  $N$  and the resulting cost volume  $CV_{split}$  for a certain partitioning type. The threshold  $T_{split}$  has to be manually selected. The most recent experiments utilize  $T_{split} = 4\%$ .

### 6.2.3 Depth Estimation Skip

The depth estimation skip is similar to the one used in FD-DSDE. The depth estimation skip is indicated by a single *gas\_skip\_flag*. If it is equal to zero, *gas\_zmax\_delta* and *gas\_zmin\_delta* have to be present in the bitstream, while *gas\_ltmin\_flag* and *gas\_ltmax\_flag* may be present. If the skip flag is equal to one, none of these syntax elements are present in the bitstream, saving significant amount of bitrate for the coding of the features. A depth estimation skip is performed for a block or sub-block if the  $L_1$  distance between the current and the previous depth block or sub-block is below  $T_{skip}$ . The most recent experiments utilize  $T_{skip} = 0.2\%$ .

### 6.2.4 Depth Estimation Parameters

The depth estimation parameters, particularly the two smoothing coefficients of DERS8 have not been adopted in the GA SEI message. As previously explained, the depth estimation is supposed to stay non-normative. While normalized smoothing coefficients, valid for multiple energy-minimization-based depth estimators could be designed, it could not be proven at the time of adoption. As a consequence, the performance of the GA SEI

message is lower than FD-DSDE. Nevertheless, the implementation in the novel depth estimator is not truly block-based. Significant advantages in terms of speed-up are lost, but the requirement of transmitting smoothing coefficients is lower.

### 6.3 Ongoing research

Various companies and institutions have shown interest in the DSDE system, which contributed to the adoption of the Geometry Absent profile [67] [68]. The system has been investigated over several meetings in scope of exploration experiments [69]. The IVDE software has been continuously upgraded to support DSDE by including the ERP format, automatizing the global depth range estimation (in GA, the depth range of the sequence is not signalled), refinement steps prior to depth estimation [70] [71] [72] and more robustness for compressed input textures [73].

After revealing FD-DSDE to the group, the technology has been continuously improved in scope of exploration experiments [74]: enhanced partitioning by rectangular shapes [75] [76], deeper splitting [77] [78] [79] and partial feature transmission [80].

Variations of H-DSDE and combinations with FD-DSDE are under investigation [81], in which full depth maps are send for a subset of views, supporting the depth estimator to estimate depth of remaining views [71].

### 6.4 Conclusion

In this chapter, the Geometry Absent Profile and the Geometry Assistance SEI message have been presented, which will be utilized in the subsequent chapters. The currently adopted technology as well as the ongoing research related to the B-DSDE, H-DSDE as well as the FD-DSDE system have been outlined.

# LOW COMPLEXITY DECODER SIDE DEPTH ESTIMATION (LC-DSDE)

---

In the previous chapter, FD-DSDE has introduced two concepts to reduce the depth estimation complexity: the first is the depth range, which reduces the local number of depth candidates. The second is the skip Flag, which allows to skip the depth estimation for future frames, if the depth did not change. The current chapter is driven by the idea to find alternative ways of recovering the depth on the decoder side and perceiving depth estimation as a "last resort" method. Instead, other strategies should be taken advantage of, which are significantly less complex in contrast to the distortion they may introduce. Indeed, instead of rate-distortion, DSDE systems may require a complexity-distortion criterion, judging an approach based on the invested complexity and the achieved distortion in the synthesized view. Typically, speed-up approaches sacrifice accuracy, *i.e.* by operating in low resolutions or by considering segments/superpixels instead of pixels. We have seen with FD-DSDE that a less complex approach does not necessarily provide worse results. B-DSDE is has already shown impressive performance over ESDE. FD-DSDE has further improved the quality and the question of "complexity" remains as the most prevalent argument against DSDE.

This chapter introduces the *low complexity decoder side depth estimation* (LC-DSDE) system, which is taking advantage of the presence of the texture decoder at the decoder-side. Particularly, motion information contained in the texture bitstream is utilized to recover depth maps temporally using motion compensation, avoiding the process of depth estimation. This way, significant complexity reduction can be expected, however, as it is not intended to send a depth residual - which would contradict the concept of DSDE - a lossy approach must be expected.

## 7.1 Description of the LC-DSDE system

It is known from 3D-HEVC studies that the motion in textures often correlate with the motion in the depth maps and therefore may serve as a useful predictor (see Chapter 2). In LC-DSDE, we try to utilize the motion information already coded in the texture bitstream to motion compensate the depth maps and therefore, reconstruct depth for future frames, without using depth estimation. The system is shown in Fig. 7.1. Again, we assume that the textures have been coded prior to the depth analysis on the encoder-side. This allows us to fetch the motion information from the texture bitstream and test it on the encoder-side for their accuracy of depth reconstruction. For this, we adopt the partitioning provided by the texture encoder, where a motion vector is given for any prediction unit. Consequently, we can decide in the corresponding depth map, for any patch as small as a prediction unit, if it shall be compensated or estimated. In order to design such a decision, the block-based approach, presented in Chapter 5, has to be reconfigured to be applicable for very small, varying patches. We know already that such a strategy introduces artifacts in the corresponding depth. Therefore, besides of the distortion introduces by motion compensation, the distortion introduced by a block-based approach has to be considered as well.

The depth recovery module on the decoder-side is utilizing the modified block-based depth estimator. The bitstream contains the result of the depth analysis and communicates to the depth recovery, which prediction units shall be compensated using the decoded depth at a previous time instance and which shall be computed using depth estimation. The very first frame is estimated as usual with the unmodified depth estimation. For subsequent frames, the proposal is used. A 2D encoder compresses the textures. For all

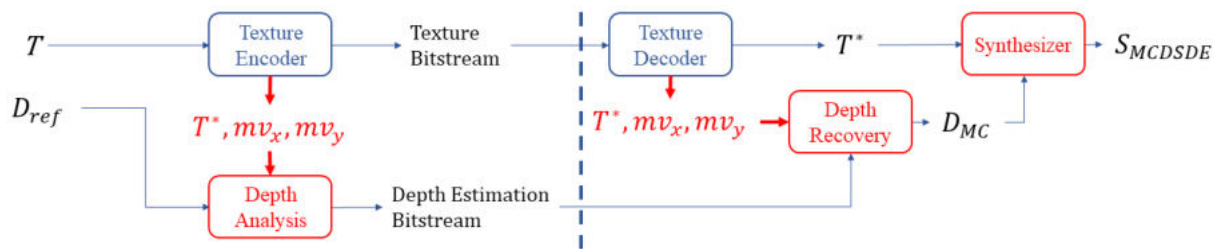


Figure 7.1 – Our proposed LC-DSDE system.  $T$  and  $T^*$  are uncoded and coded textures.  $D_{ref}$  are reference depth maps.  $D_{MC}$  are depth maps, where some blocks have been recovered by motion compensation.  $D^+$  are DSDE depth maps estimated by ffDE. Motion vectors are denoted as  $mv_x$  and  $mv_y$ . The resulting synthesized view is  $S_{MCDSDE}$  for the LC-DSDE system involving motion compensated depth.

inter-coded CUs a displacement vector  $[mv_x, mv_y]$  is present. The FFMPEG opensource library provides tools to extract these displacement vectors from the bitstream in a simple way for h.264/AVC [13]. As TMIV is compatible with any 2D codec the usage of AVC in the following experiments is possible. Furthermore, the partitioning associated with all displacement vectors is available. The displacement vectors, the decoded Textures  $T^*$  as well as the reference depth maps  $D_{ref}$  are provided to the Depth Analysis module. This module performs the following tests for all inter CUs<sup>1</sup>:

1. Perform block-based depth estimation on the CU.
2. Perform motion compensation utilizing the previous intra-coded depth frame and the associated displacement vectors  $[mv_x, mv_y]$ .
3. Compute the  $L_2$ -distance between the reference depth  $D_{ref}$  and the two depth map candidates  $D_{mc}$  and  $D_{bbDE}$  as .
4. Select the method based on the evaluation  $L_{2,bbDE} > sL_{2,mc}$ , with the error tolerance factor  $s$ . If the statement is true, the CU is recovered at the decoder-side through motion compensation. If the statement is false the CU is estimated by the block-based depth estimator.

The decision is signalled as part of the MIV bitstream and does not require normative changes to the 2D encoder.

## 7.2 Preparation for Evaluation of LC-DSDE

Starting from this chapter, we release some of the constraints set by the CTC. In order to show a proof of concept the configuration of the encoder has been drastically modified to disable intra coding in inter-frames. Furthermore, we disable B-Frames, as it will simplify the interpretation of the extracted motion information. In practice, the number of valid prediction units for motion compensation has to be expected to be smaller than in our experiments. On the other hand, the prediction is expected to be more accurate if B-Frames are enabled. Furthermore, more recent coding standards may provide displacement vectors with higher accuracy. Furthermore, we will investigate the high-bitrate scenario ( $Q_T = 25$ ) for 5 sequences, excluding the still images UnicornA and UnicornB, as they do not benefit from this approach and Shaman, as it has been removed from the CTC in the meantime, due to being too similar to other sequences.

---

1. We continue the usage of *Coding Tree Unit* and *Coding Unit* to stay consistent with the text and the usage in more recent codecs like HEVC in VVC. In AVC, the CTU is denoted as a macroblock.

Next, we look at the modifications of DERS8 to small patches. In AVC, the block size of motion compensated CUs can vary between  $4 \times 4$  to  $16 \times 16$ . We enable DERS8 to extract the partitioning from the AVC bitstream and to estimate each motion compensated patch individually. It is therefore an extension to the block-based depth estimator utilized in Chapter 5, however, adapted to micro blocks. An example of the resulting depth quality is shown in Fig. 7.2. The left depth map originates from the unmodified DERS8, while the right depth map was generated using the block-based DERS8. The "blockiness" can be clearly seen, surprisingly however, the depth map seems overall correct.

The amount of motion compensated blocks is controlled by tolerating a higher error, *i.e.* we define a block as a motion compensated block if  $L_{2,bbDE} > sL_{2,mc}$  with an error tolerance  $s$  and the  $L_2$  distance to a reference depth  $D_{ref}$ . Here, we use the unmodified DERS8 depth maps as a reference. We test the values  $s = [0.1, 0.25, 0.50, 0.75]$ . An example of the chosen blocks is shown in Fig. 7.3 For each value  $s$ , we compute the amount of motion compensated blocks as well as the resulting synthesis PSNR and MS-SSIM. Furthermore, the depth estimation runtime is provided comparing the block based DERS8 with different thresholds  $s$  and the unmodified DERS8. Computation was performed on an Intel Xeon (R) CPU E5-2520 @ 2.00 GHz. As the bitrate required to signal this decision is negligible, we focus on the objective and subjective quality of the depth maps and the synthesizes views. The more blocks are motion compensated, the better the reduction in complexity. Simultaneously, more errors are introduced into the depth maps.

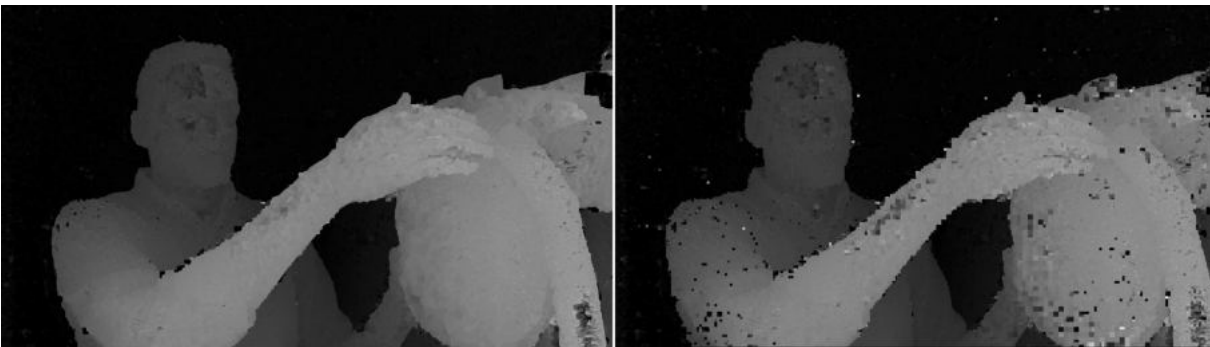


Figure 7.2 – Left: unmodified DERS8 depth. Right: modified block based DERS8 depth.





Figure 7.3 – Example for identified Prediction Units with non-zero motion information. Green blocks are motion compensated, while red blocks are estimated using the block-based DERS8 (case  $s = 0.25$ , Painter sequence).

## 7.3 Experimental Results

As usual, we first investigate the impact of our proposal on the depth maps quality, secondly on the view synthesis quality and finally investigate the benefit on the main goal of the proposal: the complexity reduction.

### 7.3.1 Depth Maps Quality

Instead of different QPs, we investigate the influence on the depth maps quality for different values of the error threshold  $s$ . The Figs. 7.4- 7.8 show the change of degradation starting from frame 0, to frame 8 and 16. In general, a quality degradation is expected in all cases, since no residual is allowed to be send, as otherwise the approach would steer towards depth coding. The impact of the last frame in the GoP is most severe for  $s = 0.1$ , also due to error propagation. However, it is apparent that even for  $s = 0.75$  an increase in artifacts may appear, for example in the Chef2 sequence. This is due to erroneously estimated depth in a block-based approach. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by

motion compensation.

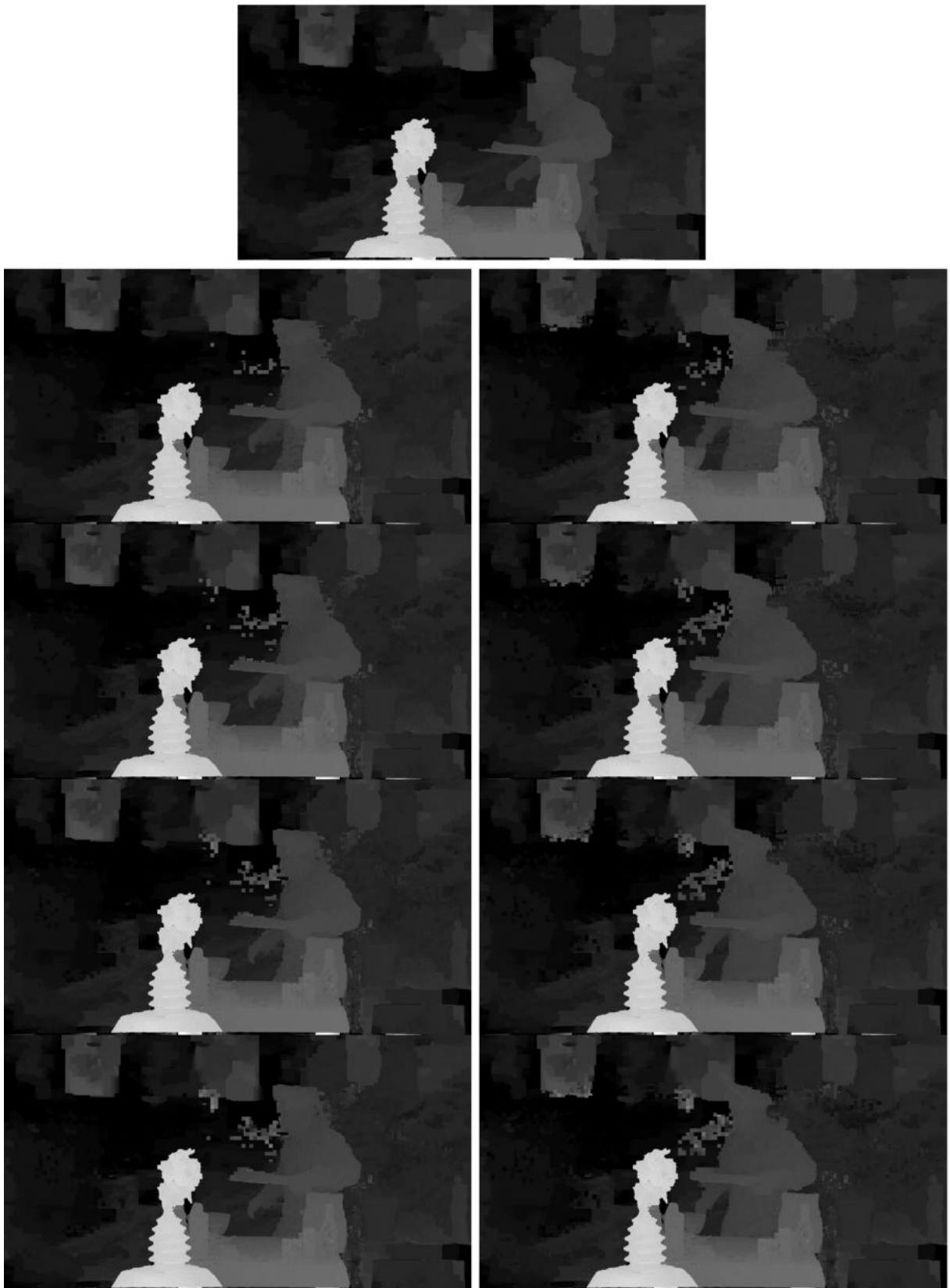


Figure 7.4 – Chef2 Sequence. Top: frame 0, estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75

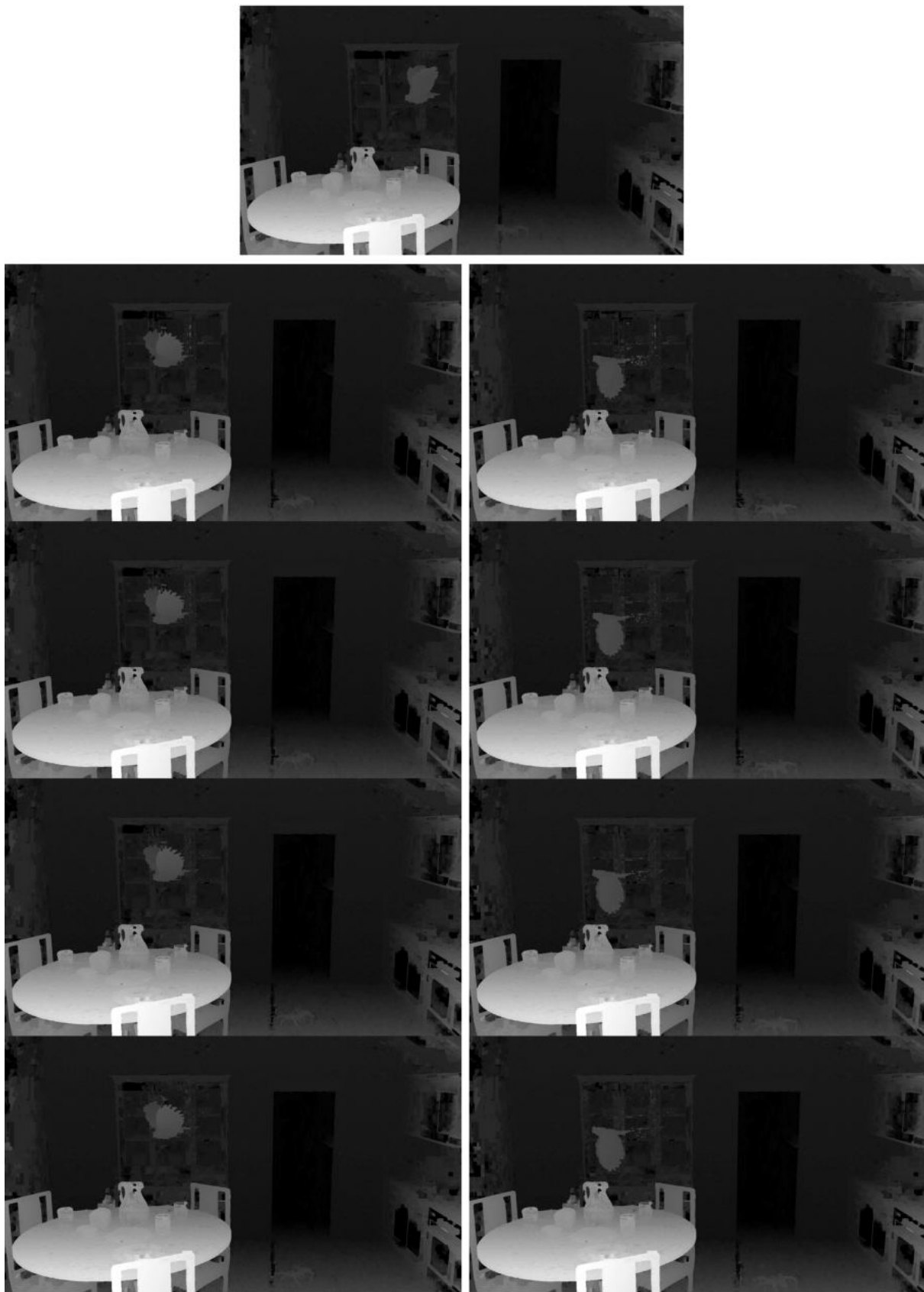


Figure 7.5 – Kitchen Sequence. Top: frame 0, estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75

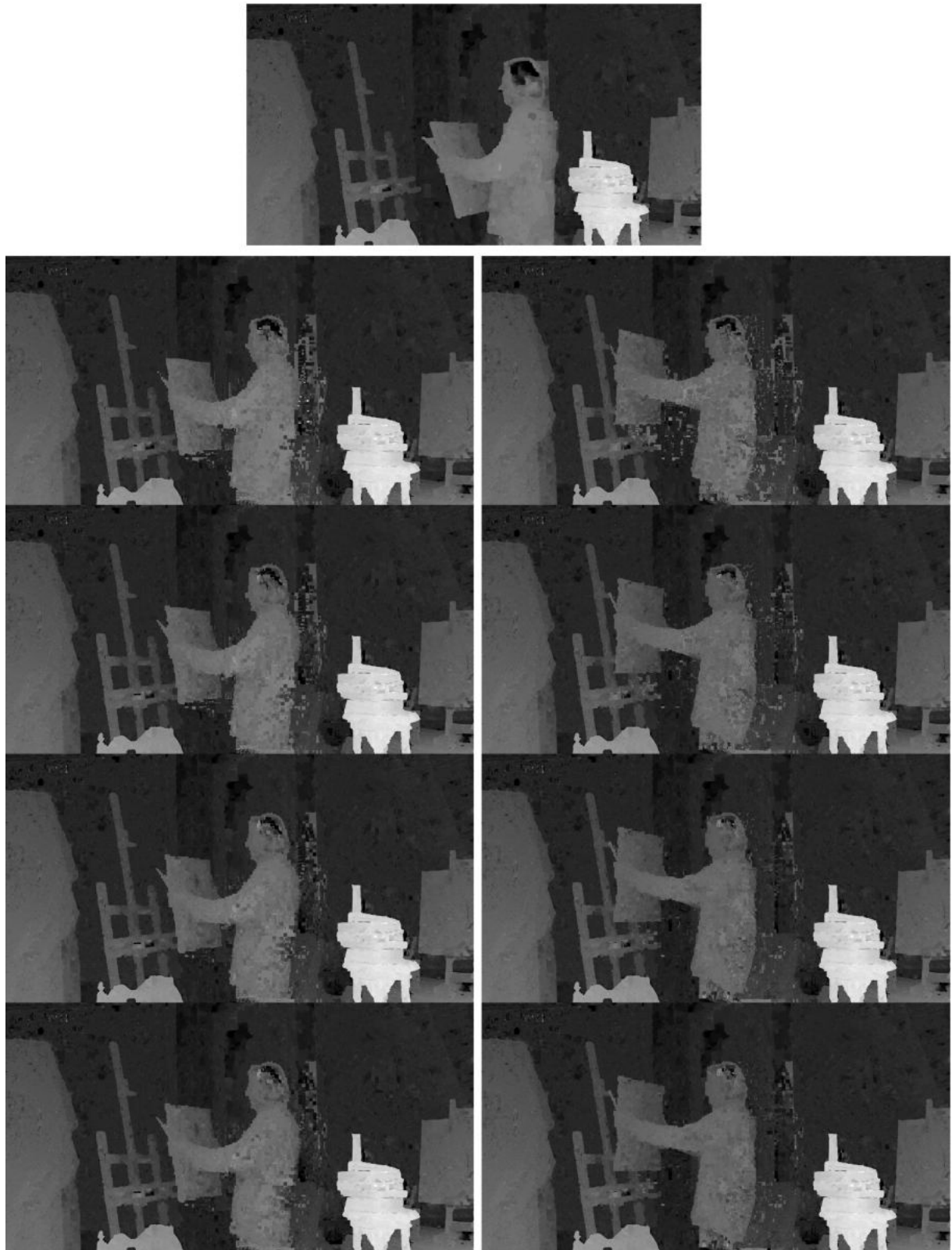


Figure 7.6 – Painter Sequence. Top: frame 0, estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75

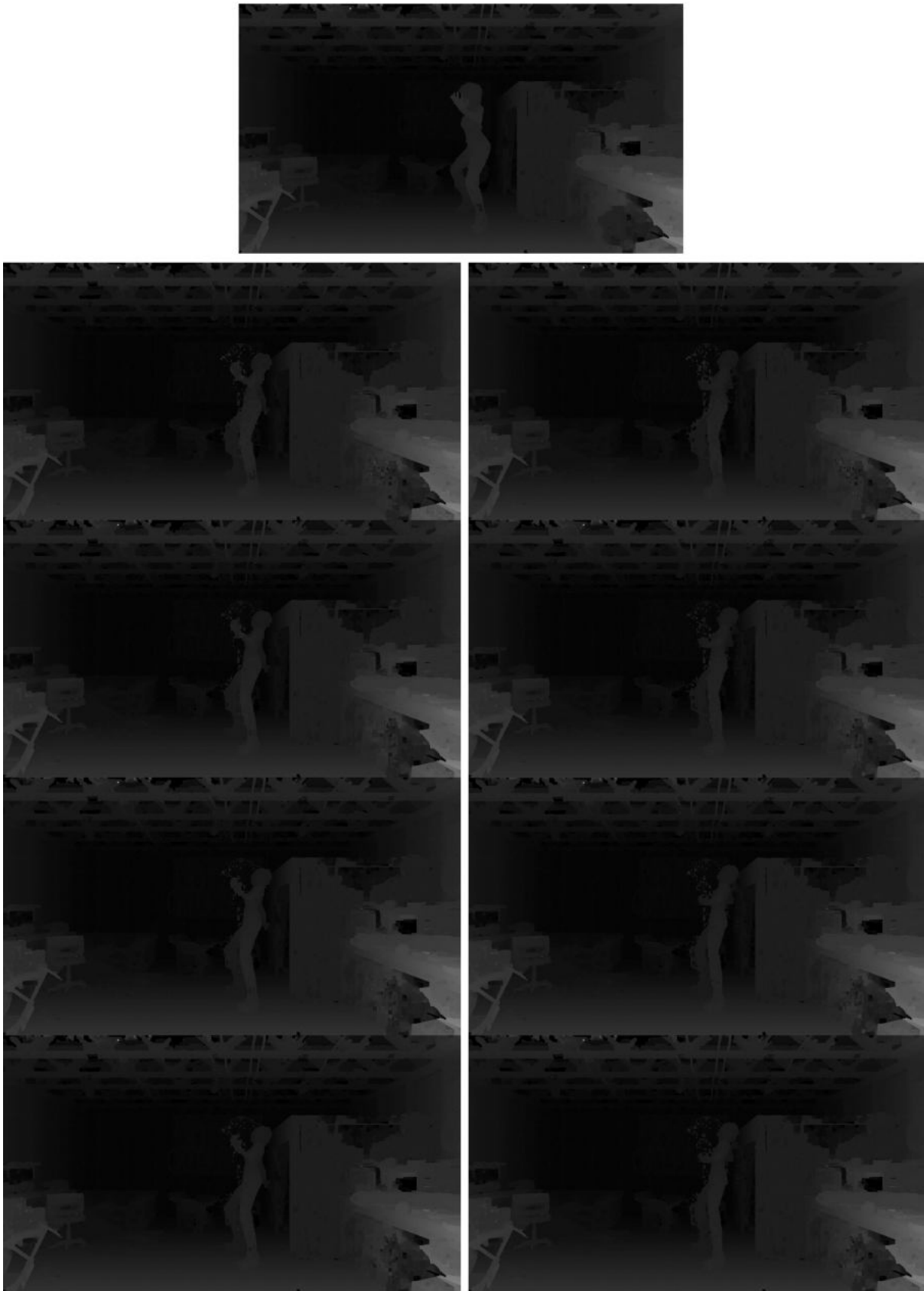


Figure 7.7 – Dancing Sequence. Top: frame 0, estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75

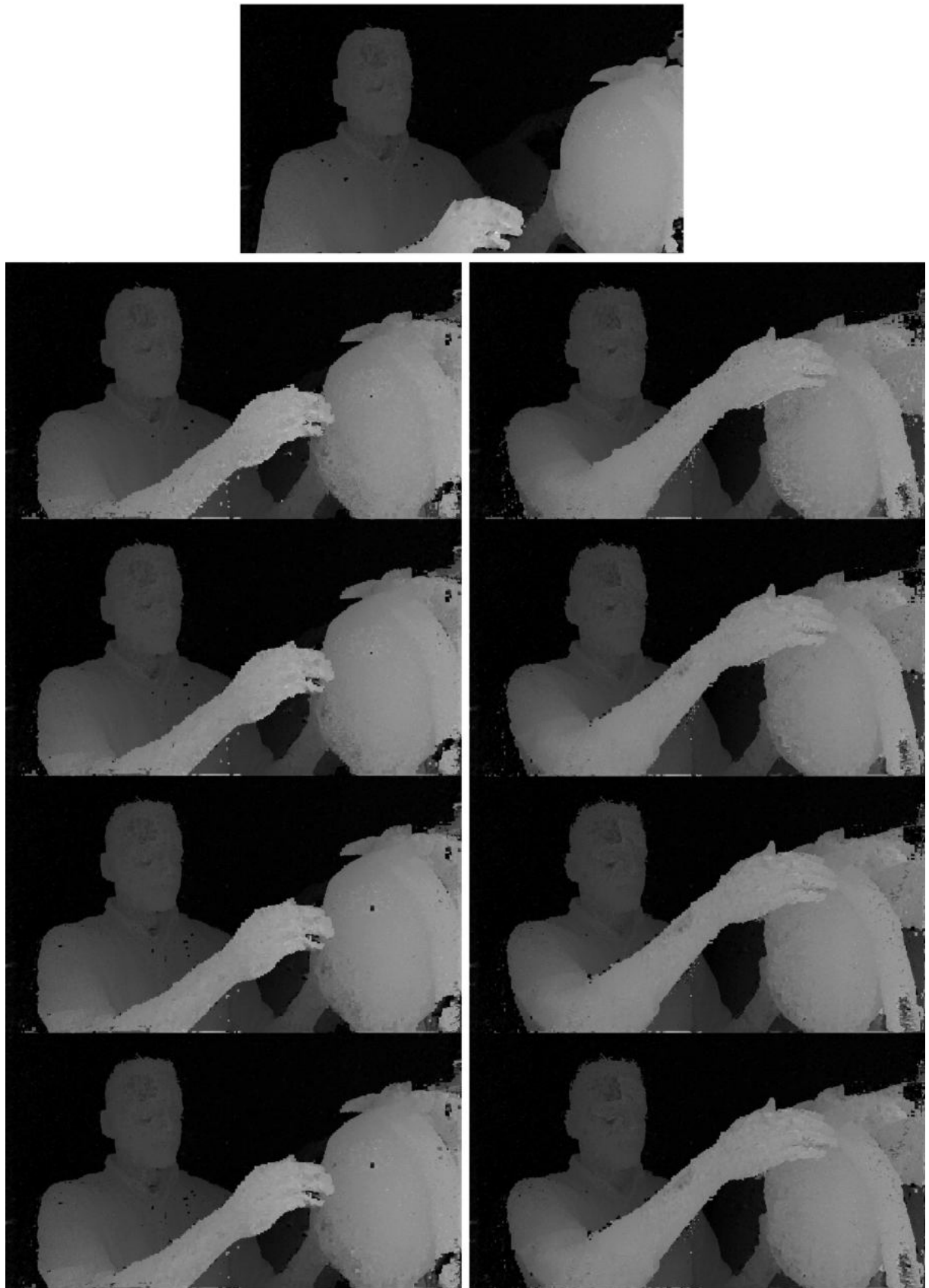


Figure 7.8 – Frog Sequence. Top: frame 0, estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75



### 7.3.2 View Synthesis Quality

The synthesized views are shown in Figs. 7.9- 7.13 for all thresholds  $s$  and the frames 0, 8 and 16. A zoom towards certain patches (with motion) is shown in Fig. 7.14 for view 16, together with their corresponding depth maps. The degradation of view synthesis quality is quite severe for  $s = 0.1$ , with strong artifacts and visible "holes" for Sequences like Frog. Nevertheless, we observe that many artifacts are not due to the motion-compensation of depth, but rather due to the block-based approach of DERS8, because several artifacts appear at areas without any motion. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by motion compensation. If a high amount of error is tolerated ( $s = 0.1$ ), the typical motion compensation artifacts in depth maps translate into an increased level of noise-like artifacts in the synthesized textures. Yet, even for  $s = 0.1$  most of the synthesized objects are reconstructed accurately.



Figure 7.9 – Chef2 Sequence. Top: frame 0, synthesized using depth maps estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75



172  
Figure 7.10 – Kitchen Sequence. Top: frame 0, synthesized using depth maps estimated unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75



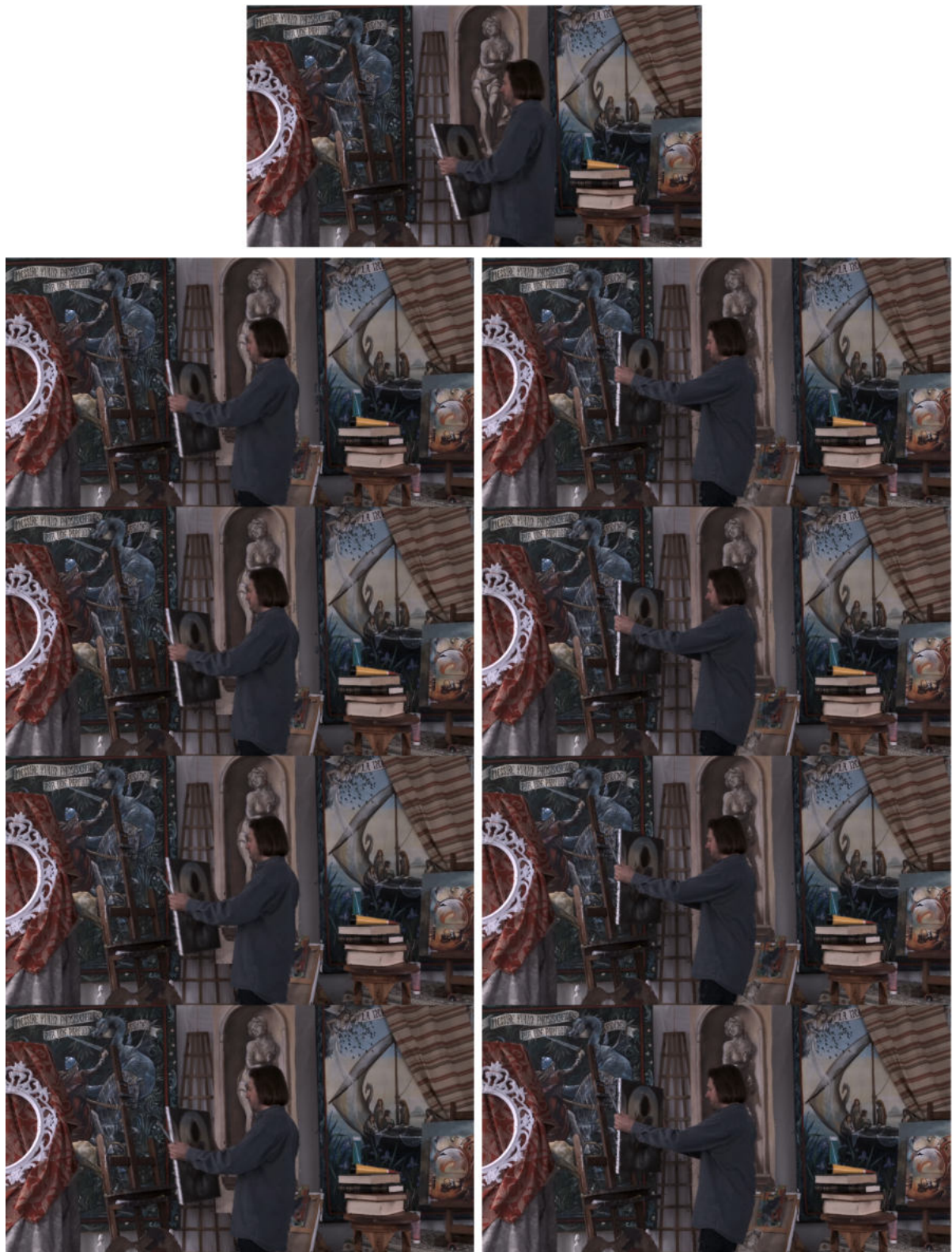


Figure 7.11 – Painter Sequence. Top: frame 0, synthesized using depth maps estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75





Figure 7.12 – Dancing Sequence. Top: frame 0, synthesized using depth maps estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75





175  
Figure 7.13 – Frog Sequence. Top: frame 0, synthesized using depth maps estimated by the unmodified DERS8. left: Frame 8. Right: Frame 16. From top to bottom: increasing threshold  $s$ , from 0.1 to 0.75

### 7.3.3 Complexity-Distortion Compromise

The main goal of the proposed LC-DSDE is to minimize the complexity (with minor impact on quality), as it is the main concern of DSDE. The objective metrics and the runtimes are shown in Tab. 7.1 for all tested sequences and thresholds  $s$ . The unmodified DERS8 is denoted as ffDE. We observe the following: first, the synthesis PSNR degrades with a lower threshold, and therefore, with more motion compensated blocks, as expected. Naturally, this is dependent on the amount of motion in sequence. Therefore, Frog has a loss of  $1.63dB$ , Painter and Chef2 losses of around  $1dB$ . In contrast, sequences with little motion have a lower degradation at  $s = 0.1$ . This is confirmed by the MS-SSIM metric. However,  $s = 0.1$  is rather an extreme-case. Another interesting observation is, that the synthesis PSNR is rather similar between  $s = 0.5$  and  $s = 0.75$ . This indicates that with  $s = 0.5$ , the majority of "safe to use" motion vectors are found, which do not significantly impact the quality in contrast to depth estimation. Again, this is confirmed by the MS-SSIM metric, which indicate the same value of 0.960 for  $s = 0.5$  and  $s = 0.75$ . The next observation is, that despite the saturation of quality from  $s = 0.5$ , there is still a large gap towards the unmodified, full frame DERS8 synthesis PSNR. On average, a loss of  $0.44dB$  is seen, with the smallest loss of  $0.12dB$  for Kitchen and the largest loss of  $0.73dB$  for Chef2. Due to the saturation of quality starting from  $s = 0.5$ , we conclude, that large parts of the degradation starting from this threshold are due to the block-based DERS8 approach and not due to the applied motion compensation.

Tab. 7.1 also summarizes the amount of motion compensated blocks, taking only into account the blocks with non-zero motion vectors. In case of  $s = 0.1$ , on average about 80% of blocks with indicated motion are used to motion compensate the depth blocks, instead of estimating them. The fraction gradually increases with the threshold to around 34% with  $s = 0.75$ . While we observed a saturation of quality loss from  $s = 0.5$ , there is indeed more blocks being compensated from  $s = 0.75$  to  $s = 0.5$ . The runtime is also shown in this Table for all thresholds and compared to the unmodified DERS8. Naturally, the runtime of the unmodified DERS8 is much larger, as it is applied on the full frame. Nevertheless, focusing the depth estimation on blocks, which are not compensated, can provide a speed-up from 105 ( $s = 0.1$ ) to 18 ( $s = 0.75$ ). Considering the quality degradation, the optimal point among the tested thresholds is  $s = 0.5$ , for which a speed-up of 22.2 for inter-frames is measured. The proposed method shows potential to reduce the depth estimation complexity significantly. Tab. 7.1 provides the runtime for inter-Frames for both, bbDE for several thresholds as well as the unmodified ffDE.



Table 7.1 – Average synthesis PSNR and MS-SSIM as well as the amount of motion compensated blocks and runtime different thresholds  $s$  and the fFDE reference. The percentage refers only to blocks which have an identified motion.

Sequence	synth PSNR [dB]					synth MS-SSIM					Motion Compensated [%]				Depth Estimation runtime [s]				
	0.1	0.25	0.5	0.75	fFDE	0.1	0.25	0.5	0.75	fFDE	0.1	0.25	0.5	0.75	0.1	0.25	0.5	0.75	fFDE
Painter	33.89	34.23	34.36	34.39	34.88	0.968	0.972	0.973	0.973	0.976	88.71	77.37	60.32	47.32	2.7	8.1	15.2	22.3	412
Kitchen	31.69	31.84	31.89	31.90	32.02	0.975	0.976	0.977	0.977	0.977	56.41	42.48	29.19	20.29	1.9	3.6	5.9	6.5	594
Dancing	30.45	30.54	30.59	30.60	30.76	0.967	0.968	0.968	0.968	0.970	79.57	65.62	49.19	37.45	2.3	4.9	9.6	14.0	1467
Chef2	30.50	30.61	30.66	30.63	31.36	0.948	0.950	0.951	0.950	0.958	89.81	76.33	55.11	42.39	2.5	11.8	27.7	42.0	1485
Frog	26.27	26.96	27.17	27.21	27.90	0.913	0.927	0.930	0.930	0.940	76.65	51.97	31.09	21.70	35	98	149	170	657
Average	30.56	30.83	30.93	30.94	31.38	0.954	0.957	0.960	0.960	0.964	78.23	62.75	44.98	33.83	8.8	25.28	41.5	51.0	923

While fFDE provides the highest synthesis PSNR performance, the computational cost is drastically increased. Depending on the threshold value, we report a speedup of a factor 18 ( $s = 0.75$ ) to 104 ( $s = 0.1$ ). For the best threshold of  $s = 0.5$ , the runtime is reduced by a factor of 22.

## 7.4 Conclusion

In this chapter, LC-DSDE is presented, which takes advantage of motion information present in the texture bitstream in order to motion compensate the depth information instead of estimating it. It was shown that a significant speed-up can be achieved for minor degradation of view synthesis quality. However, the presented solution is considered a proof of concept and more research is required to make it usable. This includes an optimized block-based depth estimation approach. However, such an approach could take advantage of the FD-DSDE proposal in Chapter 5. By minimizing the search range, the estimation will be less often erroneous for very small patches. Furthermore, the approach becomes more accurate with more advanced codecs, as in VVC, the motion can be represented with higher order models.



Figure 7.14 – Depth maps and synthesized view of two sequences. Frame 16 is shown for the unmodified reference in the B-DSDE system using ffDE as well as the MC-DSDE proposal for different thresholds.

# DECODER SIDE MULTI PLANE IMAGES (DSMPI) WITH GEOMETRY ASSISTANCE SEI MESSAGE

---

In this chapter, decoder side MPIs (DSMPIs) are proposed and enhanced utilizing the Geometry Assistance SEI. It is shown, that without modifications of the CNN-based model, a block-based MPI construction is feasible without loss of performance. Furthermore, each block can be supported by the Geometry Assistance SEI message. DSMPI allows to bypass all the disadvantages of challenges of encoder-side MPI, which has up to today not been able to outperform the MVD-based anchors of the MIV.

## 8.1 Motivation for decoder side MPIs

Our studies on the proposed DSDE systems has shown, that the transmission of depth maps through a video-based coding system is inefficient. However, particularly FD-DSDE has shown that the transmission of some information related to the geometry is very helpful. Therefore, in this chapter, we investigate to what extend our conclusions related to DSDE can be extended to and exploited by recent texture-based neural synthesizers. Several neural network-based, viewport synthesis techniques have emerged recently. Typical examples are Stereo Magnification [6] and [7] based on MPIs or IBRNet [82] based on ray marching. MPIs have been introduced in Chapter 1 as an alternative format to MVD. MPIs have also been investigated by the MPEG-I Visual group, who has decided to support the MPI format in MIV and to include a dedicated MPI encoder in TMIV. However, the application of the latter is limited to a specialized representation of the MPI,

which has shown interesting perceptual quality in case of CGI. However, significant coding losses have been shown in comparison to the MVD-based solution, following the CTC. Furthermore, the specialized representation is still difficult to achieve for natural content and the amount of data in terms of pixel rate is too large for reasonable reconstruction. Because the conversion to this specialized format is not publicly available and because shown results for MPI encoding are not very promising (following the CTC) we decide to study the concept of DSDE using MPIs directly. DSDE is a system, which is in favor of MPIs, because the huge amount of data is not required to be coded. And similar to DSDE, the complexity may be the biggest concern of decoder side MPI (DSMPI). Therefore, another goal is to investigate if MPIs can similarly benefit from the Geometry Assistance SEI message as depth estimators do. This would reveal, that our study on DSDE and our proposed FD-DSDE (or Geometry Assistance SEI) is not tied to traditional depth estimation, but can similarly be utilized with neural networks and alternative techniques which utilize geometry to perform view synthesis.

## 8.2 Adapted Test Conditions and Experimental Setup

We use the Stereo Magnification network [6] to show the benefit of the GA SEI message. The stereo magnification network estimates a single MPI given two texture views. This allows us to set up a simplified test environment using only two coded texture views. We select 8 sequences, which are compatible with this network (rectified, perspective) and 4 QP values. These sequences differ mostly from the previously presented sequences, as they are taken from more recent common test conditions. We decided to update the sequences, because MPIs have shown some benefit with non-lambertian surfaces and these new sequences reflect this difficulty. Their properties are summarized in Tab. 8.1. Stereo Magnification is by design limited to 32 planes. Methods exist, that are in principle not limited to any number planes and can provide almost perfect results [7] (if any memory and runtime limitation is ignored). However, as complexity is more crucial on the decoder side, we constrain ourselves at 32 planes, which is equivalent to using 32 depth candidates per pixel in depth estimation. Therefore, we investigate whether the GA SEI message can improve the performance given this constraint.

We encode the first two texture-views of each sequence using TMIV 8.0 and HM 16.23. Depth maps are not coded, as we operate completely in the geometry absent profile. The evaluation of an experiment is done as follows. Using two decoded textures, the

Table 8.1 – Test sequences.

Sequence	Type	Format	Resolution	NumViews
Painter	NC	LPP	2048 × 1088	16
Frog	NC	LPP	1920 × 1080	13
Carpark	NC	LPP	1920 × 1088	9
Fan	CG	LPP	1920 × 1080	15
Kitchen	CG	LPP	1920 × 1080	25
Hall	NC	LPP	1920 × 1088	9
Street	NC	LPP	1920 × 1088	9
Mirror	NC	LPP	1920 × 1080	15

MPI at view  $(x_0y_0/v_0)$  is constructed. It is subsequently used to synthesize a view at  $(x_1y_0/v_1)$ . The synthesis PSNR is computed between the synthesized view  $(x_1y_0/v_1)$  and the corresponding uncoded reference.

In total, three experiments are performed and evaluated (see Fig. 8.1):

1. Anchor generation. SM is applied without modification (Fig. 8.1a)).
2. Block-based MPI construction. The input textures are divided into equally sized blocks and padded. The padding mitigates the artifacts introduced by the extrapolation method (see next Section). The MPI construction and synthesis is applied on each block independently. The padding is removed from each synthesized block and they are stitched together. This experiment shows the impact of moving to a block-based MPI approach (Fig. 8.1b)).
3. Block-Based MPI construction using the features provided by the Geometry Assistance SEI (FD-MPI). Similar to the block-based MPI construction, however, for each block, the depth planes are positioned as proposed by the depth ranges contained in the Geometry Assistance SEI message (Fig. 8.1c)).

### 8.3 Block-based Basic Decoder Side Multi Plane Images (B-DSMPI)

The construction of the MPI on the decoder side is analogue to the B-DSDE discussed in Chapter 3, and therefore, we denote this system as basic DSMPI (B-DSMPI). In order to take advantage of the GA SEI message, the MPI construction needs to be applied

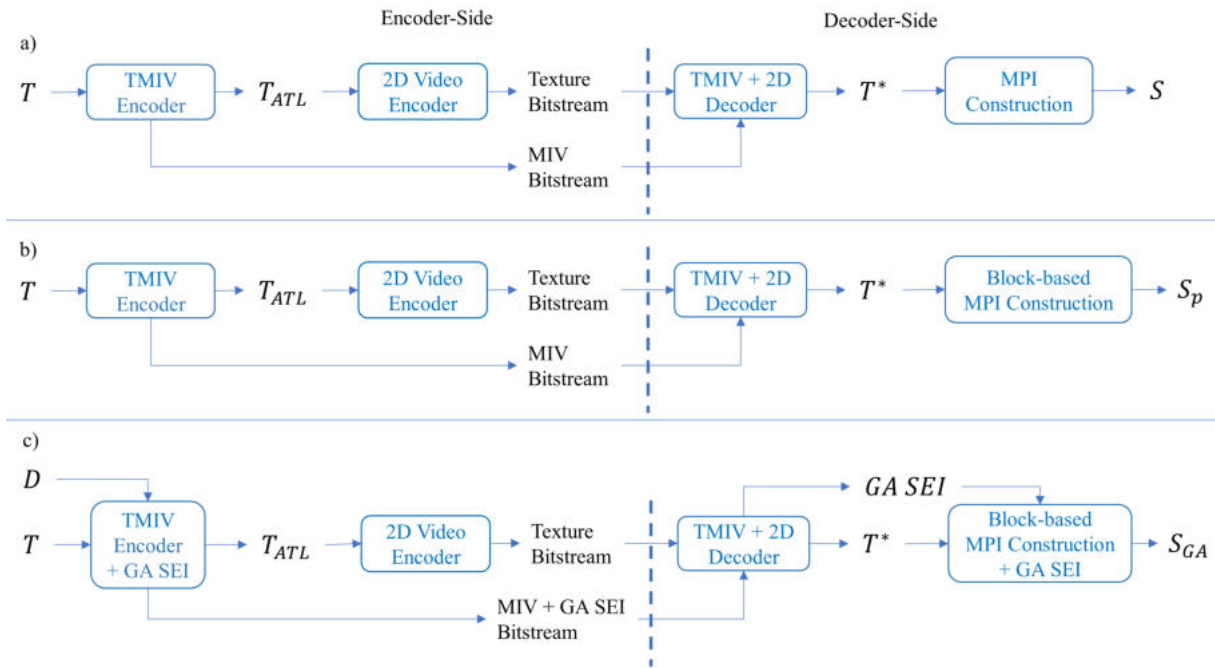


Figure 8.1 – The three decoder-side MPI systems discussed in this chapter: The unmodified system in a), the block-based system in b) and the block-based and GA-SEI enhanced system in c). Input to all systems are textures  $T$ , which are converted into Atlases  $T_{ATL}$  using the TMIV Encoder and subsequently compressed using a 2D video encoder. Depth maps  $D$  are used by the TMIV-internal feature extractor in c), in order to derive the GA SEI message. The texture bitstream is decoded by the corresponding 2D video decoder and back-converted to the multiview format  $T^*$  by the TMIV decoder. A MPI construction algorithm utilizes these textures to derive the MPIs and to synthesize novel views  $S$ ,  $S_p$  and  $S_{GA}$  in systems a), b) and c) respectively. The latter utilizes additionally the GA SEI provided by the TMIV decoder.

on a block-basis. The content is separated into blocks of size  $128 \times 128$ . In order to avoid boundary effects, it is further padded by 32 pixels on the left, top and bottom boundaries and by 224 on the right boundary, see Fig. 8.2. These numbers have been derived empirically and are motivated by the patch-based synthesis, described below. Given a full HD frame, 135 MPI-patches are estimated independently. The view synthesis

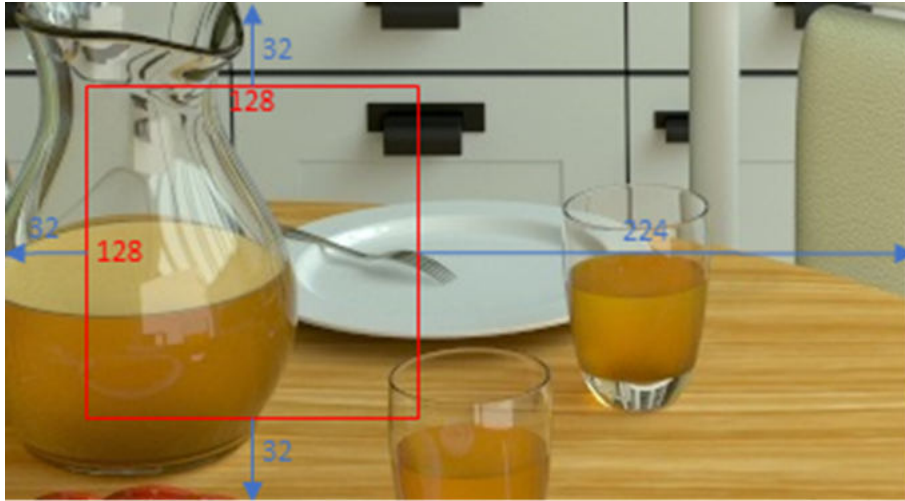


Figure 8.2 – Construction of the padding for a certain block (red box).

is performed on a MPI-patch as well. Each MPI-patch is projected to the target view and the padded area is removed. The process is shown in Fig. 8.3. Because the input of the MPI network are two patches, each synthesized patch will end up with an area that could not be recovered (grey area) and possibly an area with strong artifacts, reflecting the uncertainties in the MPI-patch. This situation motivated the one-sided strong padding. The padded synthesized patch is cropped to the final, synthesized patch. This process is repeated for all MPI-patches. The cropped, synthesized patches are stitched together to create the final synthesized view, see Fig. 8.4. The boundary effects (due to the block based approach) are barely visible. The main artifact remains at the far right view, which remains as the consequence of the chosen extrapolation approach. Due to this artifact, we restrain our PSNR evaluation on  $height \times (width - 200)$ , excluding this artifact in all experiments.



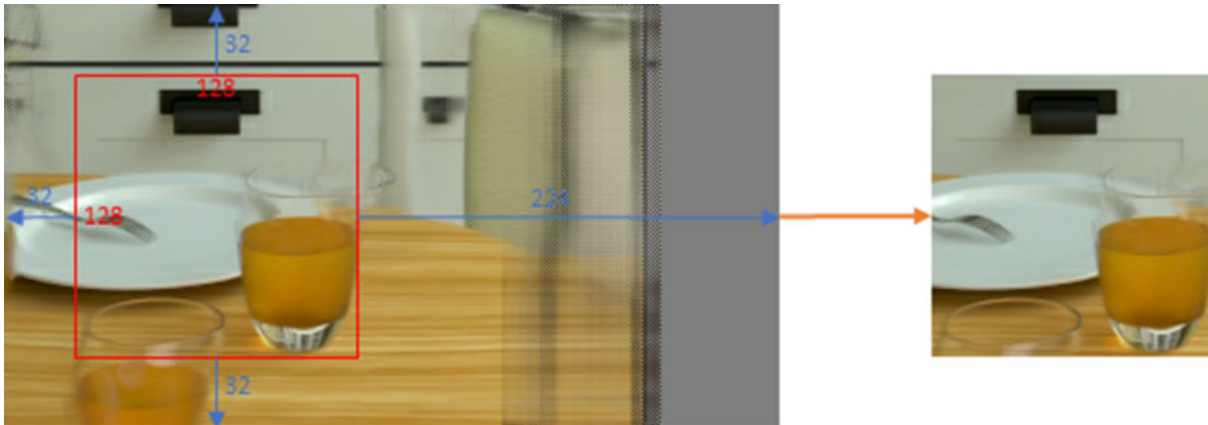


Figure 8.3 – Left: synthesized patch using one patch-MPI. Right: final cropped, synthesized patch.



Figure 8.4 – Final synthesized view, stitched together from synthesized patches.

## 8.4 Enhanced B-DSMPI with GA SEI

The last section described the prerequisite for the utilization of the GA SEI. Fig. 8.1c) shows the system overview of our proposed approach. The GA SEI is derived on the encoder-side utilizing the feature extractor of TMIV (see Chapter 6). It is encoded into the MIV bitstream and therefore increases the total bitrate of the proposal, which is considered in the evaluation section. However, it does not affect the coded texture bitstream and therefore, all synthesis PSNR improvements are due to the GA SEI. On the decoder-side, the GA SEI is provided to the block-based MPI construction algorithm. Utilizing the GA SEI, the relevant depth range is known to the MPI constructor. The 32 planes, available by the Stereo Magnification algorithm, can therefore be placed in-between the relevant depth range in every block. Consequently, a denser sampling in the depth space can be achieved from content, which is actually relevant. In the unmodified system, content outside the valid depth range, and therefore, outside the volumetric video will be sampled. This situation is demonstrated in Fig. 8.5, which shows the transparency maps for all 32 depth planes sampled in the reference and our proposal for a given block. The depth range of the Carpark sequence is  $[D_{min}^{global}, D_{max}^{global}] = [0.34m, 27.6m]$ . Consequently, an MPI layer is constructed approximately every meter in the given example. It can be observed, that the majority of transparency layers are zero, because the corresponding content is not present in processed block. zero-layers do not contribute to the alpha-blending process during view synthesis and are therefore irrelevant. In the GA SEI enhanced proposal however, the relevant depth range for this block  $b$  is identified as  $[D_{min}^{b,GA}, D_{max}^{b,GA}] = [21.8m, 26.1m]$  and the available 32 depth planes are placed in this range. Consequently, the sampling is more dense in the range, in which relevant content exists and the corresponding transparency layers are non-zero. In other words, more relevant information from the volumetric video is captured in the MPI representation utilizing the GA SEI.

## 8.5 Coding Efficiency of B-DSMPI with Geometry Assistance SEI

When analysing the results, we have observed, that the MPI-specific artifacts are rather constant among all QPs. Even further, we have observed a strong robustness of the CNN-based stereo magnification approach towards compression input. Consequently, the visual results shown in the following are computed using *uncoded* input textures.

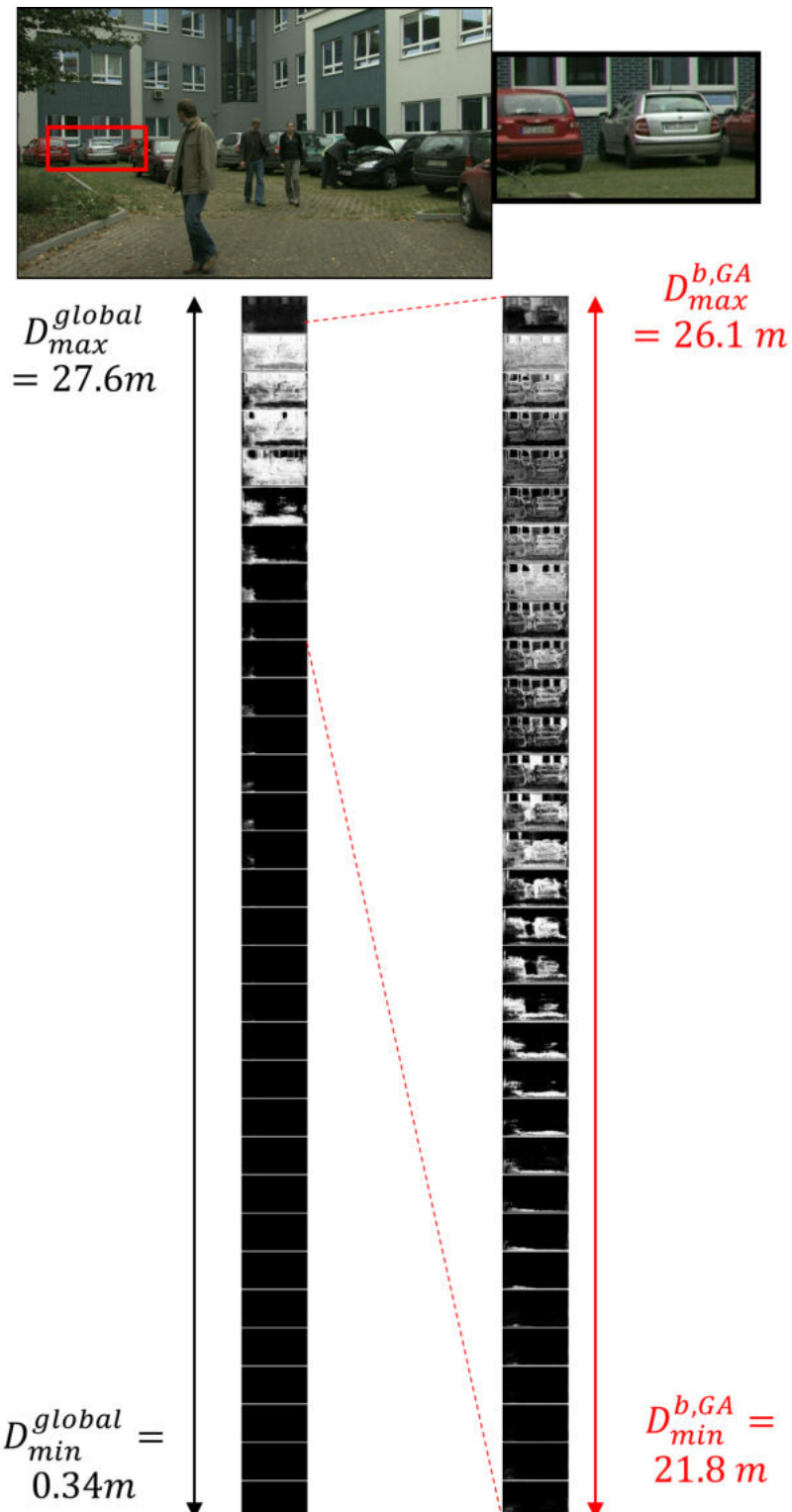


Figure 8.5 – Illustration of the benefit of the GA SEI. Real example based on the Carpark sequence. For the given block, the GA SEI allows to analyze the relevant depth range, leading to more meaningful transparency layers.

### 8.5.1 View Synthesis Quality

The Figs. 8.6- 8.12 show the view synthesis quality of the anchor and the proposal utilizing the Geometry Assistance SEI message. Without increasing the number of depth planes, the resulting synthesized images become significantly sharper. This is because the depth planes have been locally placed, where the content actually exists and therefore, the 32 planes carry more relevant information. If the depth range of the full frame is used, the depth planes may be locally placed without covering any existing information, as they are distributed uniformly. In the estimated MPIs, those planes are often found to be "empty", meaning, the transparencies are zero and the area does not carry meaningful information. Furthermore, we observe that by narrowing down the range, the extrapolation works better and more information can be recovered. The fan sequence is particularly interesting as it shows where the depth-driven approach may fail. In case of a "fence"-like object, which allows to see through the object, the depth ranges become useless. The same would hold for the FD-DSDE approach presented in Chapter 5.

Tab. 8.2 summarizes the synthesis PSNR for the uncoded sequences. As expected, due to the limitation to 32 planes, the anchor performance is relatively low and vary strongly from sequence to sequence ( $20.26dB$  to  $33.58dB$ ). The Table confirms, that the impact of moving to a block-based MPI approach is relatively low. On average, a loss of  $-0.2dB$  originates from the block-based computation of the MPIs. The Geometry Assistance SEI improves the quality from  $1.75dB$  for the Carpark sequence to up to  $6.16dB$  for the Hall sequence. The only exception is the Painter sequence, which show a loss of  $-0.63dB$ . This odd result is however consistent with Chapter 3-5, in which Painter has already shown losses. On average, a quality improvement of  $2.74dB$  is observed if the Geometry Assistance SEI is utilized.

### 8.5.2 Efficiency over all rate points

The quality improvement in terms of synthesis PSNR is too large to compute any synthesis BD-Rate values. Therefore, the RD curves are shown instead in Figs. 8.13-8.16. We see, that the bitrate increase due to the Geometry Assistance SEI is marginal and therefore, almost all improvements are due to the quality increase. Furthermore, the quality improvement is consistent across all bitrates.

Table 8.2 – PSNR performance of the anchor, block-based MPI with and without Geometry Assistance SEI.

Sequence	Anchor	Block-based MPI (without GA SEI)	Difference	GA-MPI	Difference
Painter	33.58	33.37	-0.21	32.94	-0.63
Mirror	23.98	23.56	-0.41	25.83	+1.86
Kitchen	27.45	27.40	-0.05	32.18	+4.87
Frog	22.96	22.89	-0.07	24.72	+1.75
Fan	20.26	20.07	-0.18	22.82	+2.57
Carpark	28.98	28.79	-0.19	30.69	+1.70
Hall	29.25	28.92	-0.32	35.42	+6.16
Street	30.49	30.34	-0.15	34.13	+3.64
Average	27.12	26.92	-0,20	29.84	+2.74





Figure 8.6 – View synthesis result of the Carpark sequence. Top: anchor. Bottom: proposal.



Figure 8.7 – View synthesis result of the Fan sequence. Top: anchor. Bottom: proposal.



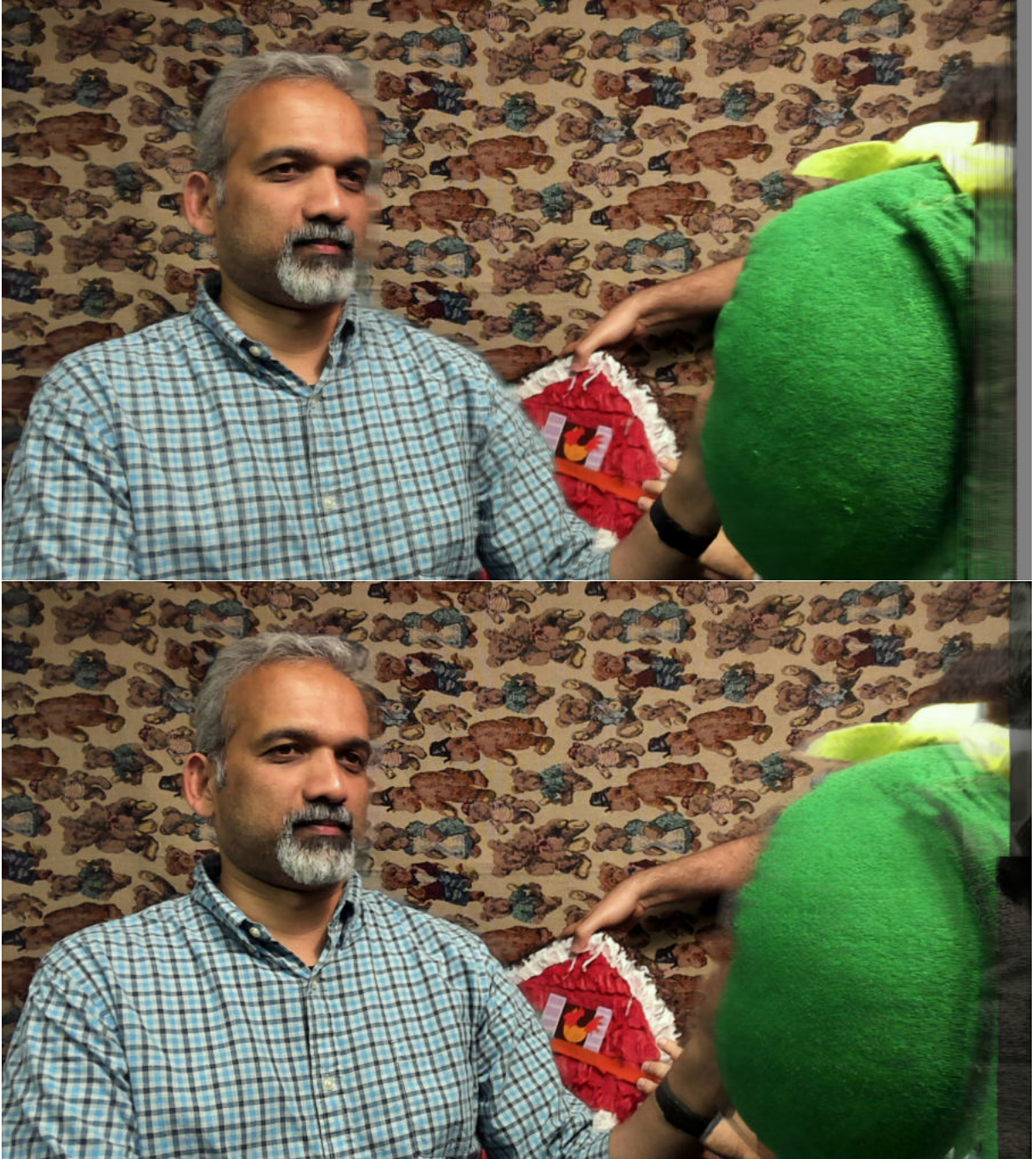


Figure 8.8 – View synthesis result of the Frog sequence. Top: anchor. Bottom: proposal.





Figure 8.9 – View synthesis result of the Kitchen sequence. Top: anchor. Bottom: proposal.



Figure 8.10 – View synthesis result of the Mirror sequence. Top: anchor. Bottom: proposal.





Figure 8.11 – View synthesis result of the Painter sequence. Top: anchor. Bottom: proposal.



Figure 8.12 – View synthesis result of the Street sequence. Top: anchor. Bottom: proposal.

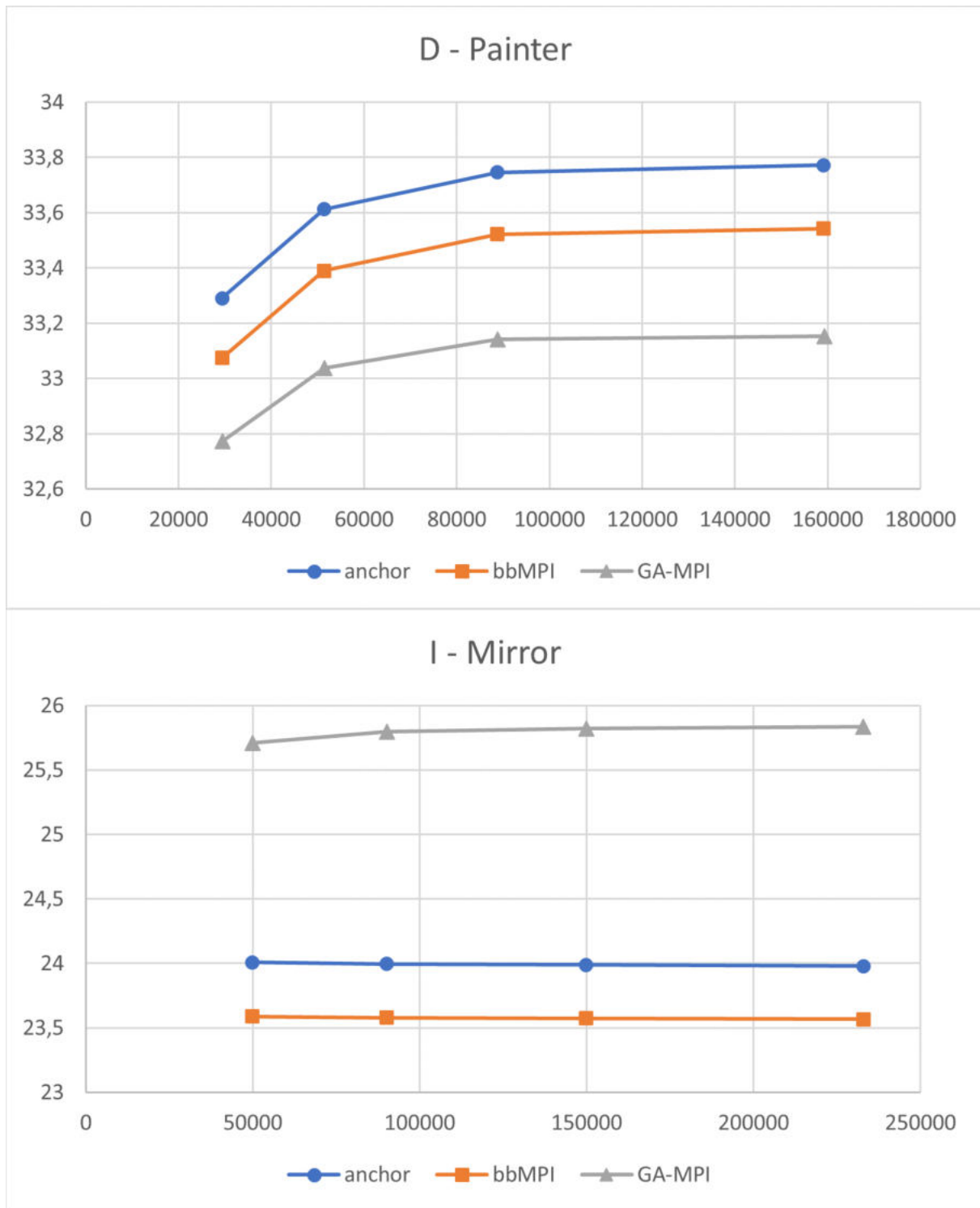


Figure 8.13 – RD-curves of the Painter and Mirror sequences of the anchor, the block-based MPI (bbMPI) and the proposal using the Geometry Assistance SEI message (GA-MPI).



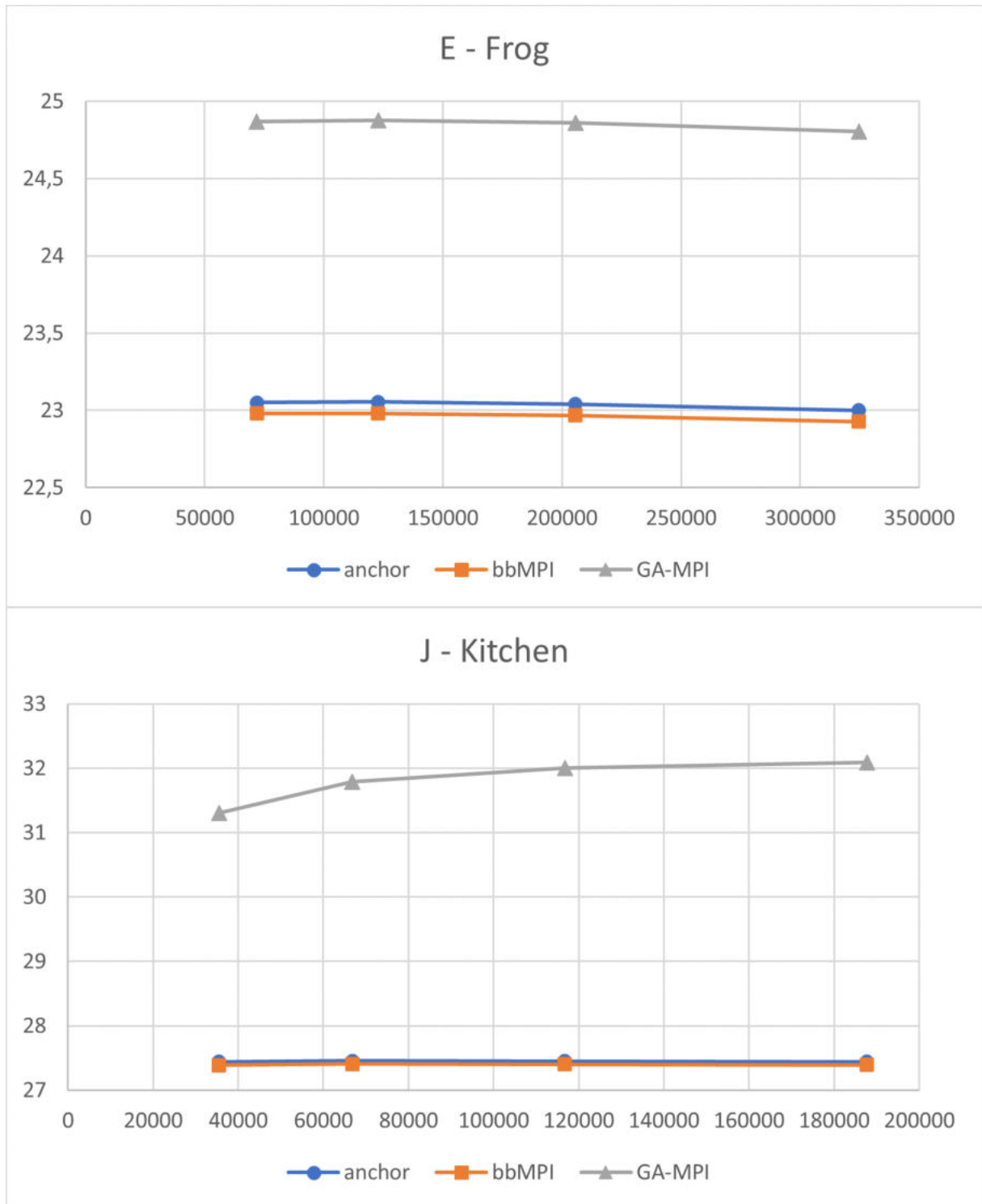


Figure 8.14 – RD-curves of the Frog and Kitchen sequences of the anchor, the block-based MPI (bbMPI) and the proposal using the Geometry Assistance SEI message (GA-MPI).



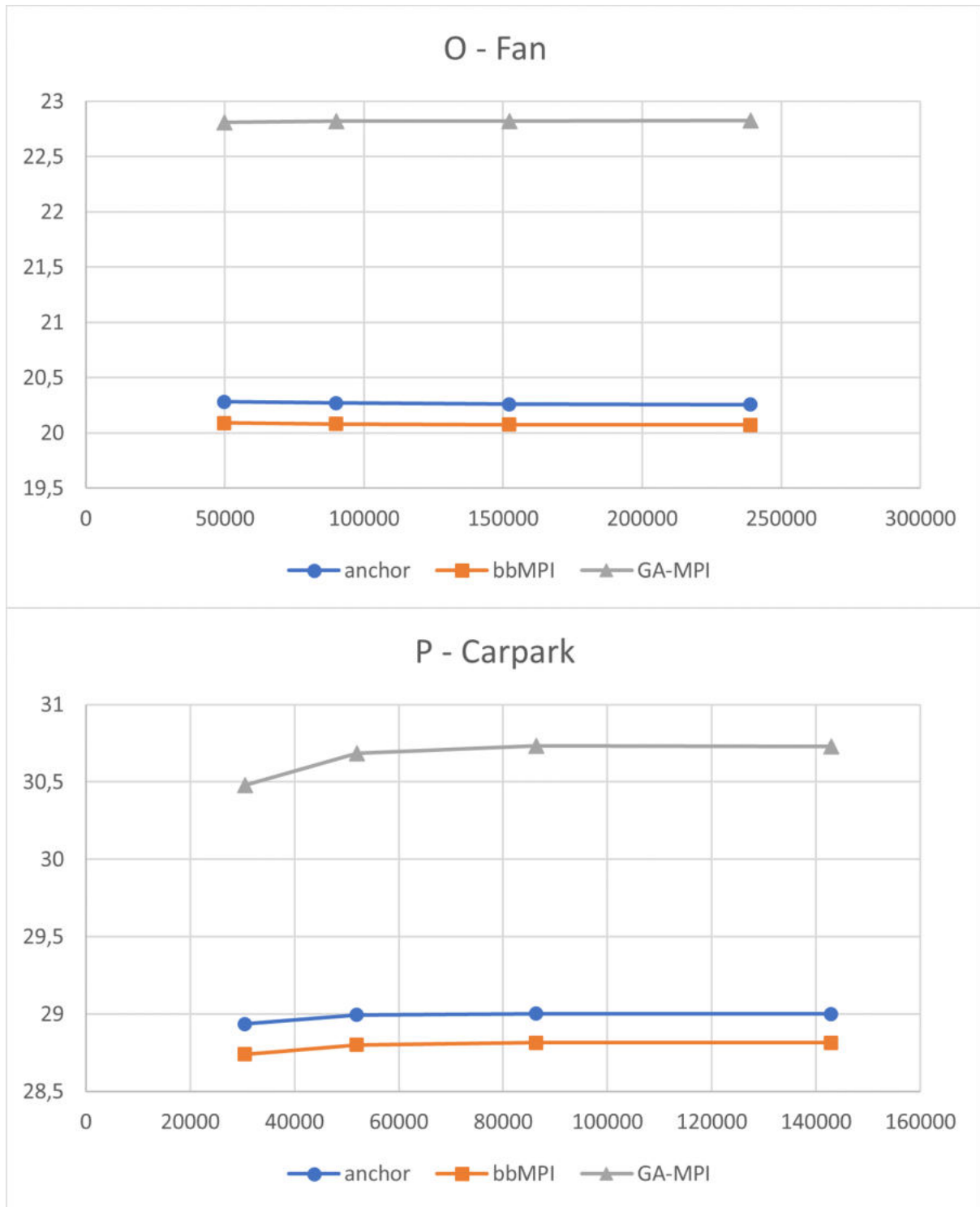


Figure 8.15 – RD-curves of the Fan and Carpark sequences of the anchor, the block-based MPI (bbMPI) and the proposal using the Geometry Assistance SEI message (GA-MPI).

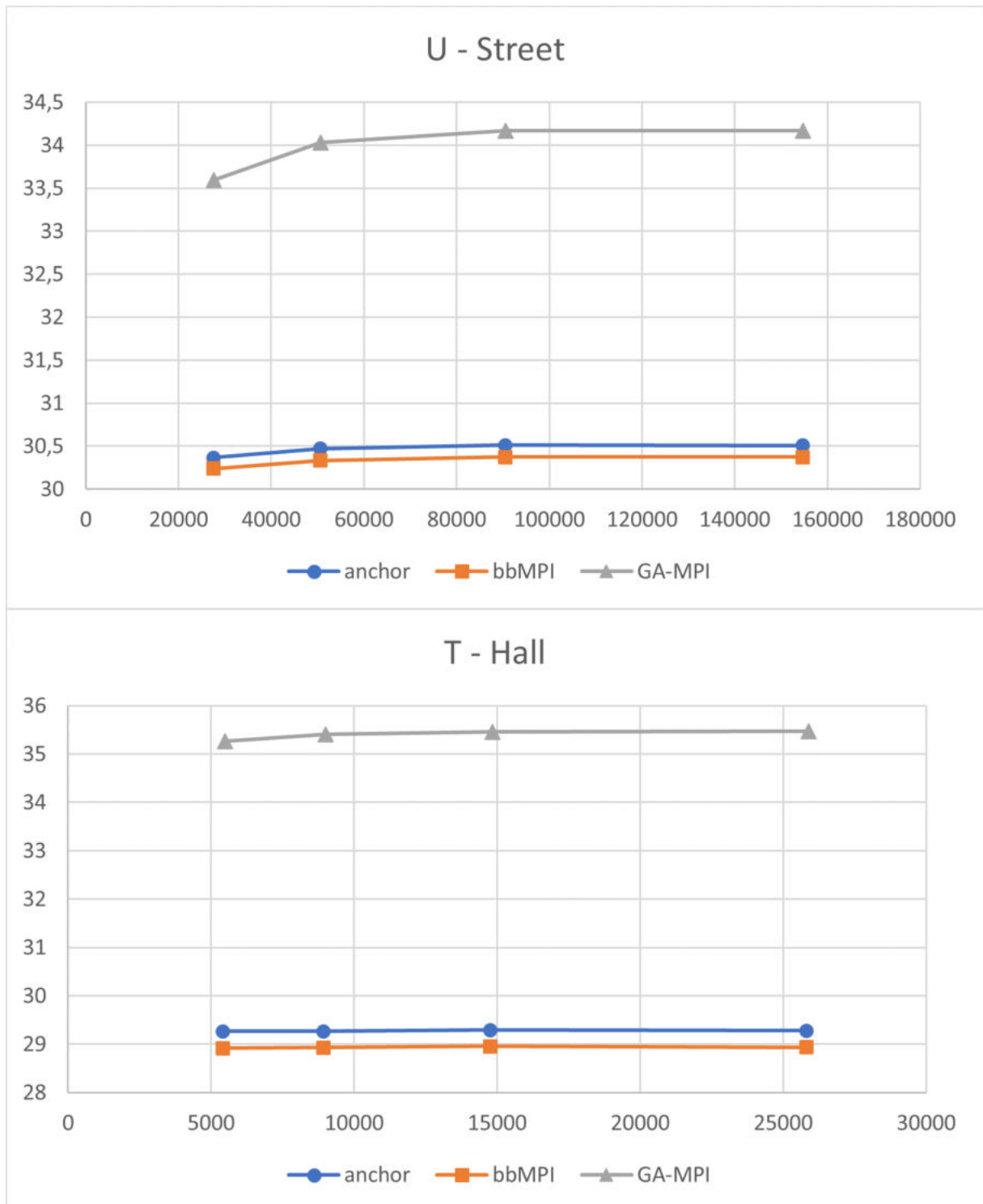


Figure 8.16 – RD-curves of the Hall and Street sequences of the anchor, the block-based MPI (bbMPI) and the proposal using the Geometry Assistance SEI message (GA-MPI).

## 8.6 Conclusion

In this Chapter, B-DSMPI has been proposed, which considers the MPI construction on the decoder-side. We have shown, that neural network-based methods are more robust to compression artifacts in the input textures, leading to a more consistent synthesis quality over the QPs. We have shown, that a block-based application of the MPI construction network can be performed with minor quality losses, which is the foundation to support each block with supplemental information or to skip the construction temporally. Finally, we have enhanced the block-based MPI construction using the Geometry Assistance SEI message, which has been originally designed for depth estimation. The objective quality has been improved by up to 6.16 dB and 2.74 dB on average. The subjective quality is significantly higher with sharper synthesis results and mitigated artifacts. This improvement has been achieved using a limitation of 32 planes, which better respect the complexity-critical situation on the decoder-side.

# CONCLUSION

---

The research conducted in scope of this thesis has brought Decoder-Side Depth Estimation to life. DSDE has been a system, which did not find its place during the standardization activities of 3D-HEVC. However, we have foreseen the consequences of the design choices for the novel MPEG Immersive Video standard and understood that the challenge of reducing pixel rate as well as the inappropriate coding of depth maps through 2D codecs can be solved by DSDE. Convinced by the impressive 37% synthesis BD-Rate improvement and 50% pixel rate savings by B-DSDE compared to the MV-HEVC anchor, several opportunities of improving and extending the simple B-DSDE architecture have been proposed. After several meeting cycles after proposal, the B-DSDE system has been adopted as the *Geometry Absent profile* in the novel MIV standard.

In H-DSDE, a CTU-based Rate-Distortion optimized decision has been performed to select between ESDE and DSDE mode per CTU. For this manner, the SVDC has been utilized. We have shown, that some parts of estimated depth at the encoder-side may still contain useful information in contrast to the decoder-side depth. As a result the subjective performance has improved as several DSDE-related artifacts have been resolved. At the same time, many ESDE-CTUs, which introduce strong distortion, could be avoided. A more efficient, video-based coding with multiview capabilities as opposed to HM simulcast could make this kind of solution more attractive in terms of coding efficiency. Furthermore, while it avoids the problems of DSDE, it maintains the problems of ESDE. As of writing this thesis, H-DSDE is still being investigated by MPEG.

In FD-DSDE, the disadvantages of ESDE shall not be re-introduced. We maintain a fully DSDE-type system and do not code any geometry by spending pixel rate. Instead, several features have been identified, which support the depth estimator at the decoder side to perform faster depth estimation with higher accuracy. The features include block-based, adapted depth ranges, a partitioning concept and a skip flag, which indicates for inter-frames to avoid depth estimation for a certain block. This metadata is coded into the MIV bitstream through CABAC losslessly. Therefore, not tuning of quantization parameters is required. All objective metrics have improved with 46% synthesis BD-Rate gain over ESDE. At the same time, the amount of data required to be processed by the

---

decoder-side depth estimator is reduced by 90% and a speedup of around x20 is achieved compared to the reference depth estimator. This proposal has been, together with some modifications and several improvements, adopted as the *Geometry Assistance SEI message* in the novel MIV standard. As of writing this thesis, FD-DSDE is still being investigated by MPEG, including synergies with H-DSDE.

In LC-DSDE we utilize the displacement information present in the bitstream in order to motion compensate certain depth CTUs for inter-frames, instead of estimating them. This choice is done on the encoder-side. We have shown, that a significant speed-up between 18 and 104 can be achieved, depending on the tolerated synthesis PSNR loss.

In DSMPI it has been shown that the benefit of Geometry Assistance SEI message of TMIV is not limited to depth estimation. Instead, the presence of the depth estimator on the decoder-side can be utilized to merge the geometry estimation and the view synthesis into a joint method. The MPI construction is one example for such a solution. We show, that a block-based MPI approach is possible and enhance each block by utilizing the Geometry Assistance SEI. Without increasing the number of analysed depth planes, the objective quality is improved by up to 6 dB and 2.74 dB on average.

# LIST OF PUBLICATIONS

---

## Contributions to Standardization

1. **P. Garus**, J. Jung, and P. Boissonade, [MPEG-I Visual] DERS7 anchor results, ISO/IEC JTC1/SC29/WG11 MPEG2019/m46716, Mar. 2019.
2. **P. Garus**, J. Jung, and P. Boissonade, [MPEG-I Visual] Evaluation of DERS modifications proposed in [m45265], ISO/IEC JTC1/SC29/WG11 MPEG2019/m46799, Mar. 2019.
3. R. Laot, P. Boissonade, J. Jung, **P. Garus**, [MPEG-I Visual] Code base clean-up of DERS proposed in [m45265], ISO/IEC JTC1/SC29/WG11 MPEG2019/m47761, Mar. 2019.
4. **P. Garus** and J. Jung, [MPEG-I Visual] A 37% MV-HEVC+VVS anchor improvement with 50% pixel rate reduction, ISO/IEC JTC1/SC29/WG11 MPEG2019/m49153, Jul. 2019.
5. M. Milovanovic, **P. Garus**, P. Boissonade and J. Jung, [MPEG-I Visual] MV-HEVC anchor results for immersive video CTCs template (w18443), ISO/IEC JTC1/SC29/WG11 MPEG2019/m49093, Mar. 2019.
6. **P. Garus**, F. Henry, G. Clare, and J. Jung, Feature-driven decoder side depth estimation, ISO/IEC JTC 1/SC 29/WG 4 M54994, Oct. 2020
7. G. Clare, **P. Garus**, and F. Henry, [MIV] Geometry Assistance SEI message, ISO/IEC JTC1/SC29/WG04 MPEG/M56626, Apr. 2021.
8. G. Clare, **P. Garus**, F. Henry, et al., [MIV] Combination of m56626 and m56335 for Geometry Assistance SEI message, ISO/IEC JTC1/SC29/WG04 MPEG/M56950, Apr. 2021.



---

## Publications

1. **P. Garus**, J. Jung, T. Maugey and Christine Guillemot, « Bypassing depth maps transmission for immersive video coding », in 2019 Picture Coding Symposium (PCS), 2019, pp. 1–5. doi: 10.1109/PCS48520.2019.8954543.
2. **P. Garus**, F. Henry, J. Jung, T. Maugey and C. Guillemot, « Immersive video coding: should geometry information be transmitted as depth maps? », IEEE Transactions on Circuits and Systems for Video Technology, pp. 1–1, 2021. doi: 10.1109/TCSVT.2021.3100006.
3. D. Mieloch, **P. Garus**, M. Milanovic, Joel Jung, Jun Young Jeong, Smitha Lingadahalli Ravi and Basel Salahieh, « Overview and efficiency of decoder-side depth estimation in MPEG immersive video », IEEE Transactions on Circuits and Systems for Video Technology, 2022.
4. **P. Garus**, M. Milanoanovic, J. Jung and Marco Cagnazzo, « MPEG Immersive Video », in Immersive Video Technologies, G. Valenzise, M. Alain, E. Zerman, et al., Eds., to be published November 1st, 2022, Elsevier Science & Technology, 2022.

---

## Patents

1. **P. Garus**, P. Nikitin, J. Jung, "Procédé et dispositif de traitement de données de vidéo multi-vues", French Patent FR3096538A1, 27.11.2020.
2. **P. Garus**, P. Nikitin, J. Jung, "Procédé et dispositif de traitement de données de vidéo multi-vues", French Patent FR3107383A1, 19.08.2021.
3. **P. Garus**, F. Henry, G. Clare, "Codage et décodage d'une vidéo multi-vues", French Patent FR3114716A1, 01.04.2022.
4. Jung, Joël, Pavel Nikitin, and **Patrick Garus**. "Method and device for processing multi-view video data." U.S. Patent Application No. 17/622,486.



# BIBLIOGRAPHY

---

- [1] D. Graziosi, O. Nakagami, S. Kuma, A. Zagheto, T. Suzuki, and A. Tabatabai, « An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc) », *APSIPA Transactions on Signal and Information Processing*, vol. 9, e13, 2020. DOI: 10.1017/ATSIP.2020.12.
- [2] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, *Title 8i voxelized full bodies – a voxelized point cloud dataset*, ISO/IEC JTC1/SC29/WG11 m40059, Jan. 2017.
- [3] D. Flynn, S. Lasserre, and G. Martin-Cocher, *Pcc cat3 test sequences from blackberry/qnx*, ISO/IEC JCTC1/SC29/WG11 MPEG/m23647, Jul. 2018.
- [4] L. Ilola, V. Kumar Malamal Vadakital, K. Roimela, and J. Keränen, *New test content for immersive video – nokia chess*, ISO/IEC JTC1/SC29/WG11 MPEG2019/M50787, Oct. 2019.
- [5] D. Doyen, G. Boisson, and R. Gendrot, *[mpeg-i-visual] ee depth: new version of the pseudo-rectified technicolorpainter content*, ISO/IEC JTC1/SC29/WG11 MPEG2018/m43366, Oct. 2019.
- [6] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, *Stereo Magnification: Learning View Synthesis using Multiplane Images*, 2018. arXiv: 1805.09817 [cs.CV].
- [7] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, *et al.*, « Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines », *ACM Trans. Graph.*, vol. 38, 4, Jul. 2019, ISSN: 0730-0301. DOI: 10.1145/3306346.3322980.
- [8] J. Navarro and N. Sabater, « Compact and adaptive multiplane images for view synthesis », in *2021 IEEE International Conference on Image Processing (ICIP)*, 2021, pp. 3403–3407. DOI: 10.1109/ICIP42928.2021.9506403.
- [9] J. Fleureau, R. Doré, B. Chupeau, F. Thudor, G. Briand, and T. Tapie, *MIV CE1-Related - Activation of a transparency attribute and application to MPI encoding*, ISO/IEC JTC 1/SC 29/WG 04 MPEG/m55089, Oct. 2020.

- 
- [10] D. B. Graziosi and B. Kroon, « Video-based coding of volumetric data », in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2706–2710. DOI: 10.1109/ICIP40778.2020.9190689.
- [11] A. Vetro, T. Wiegand, and G. J. Sullivan, « Overview of the stereo and multiview video coding extensions of the h.264/mpeg-4 avc standard », *Proceedings of the IEEE*, vol. 99, 4, pp. 626–642, 2011. DOI: 10.1109/JPROC.2010.2098830.
- [12] Y. Chen, Y.-K. Wang, K. Ugur, M. M. Hannuksela, J. Lainema, and M. Gabbouj, « The emerging mvc standard for 3d video services », vol. 2009, Jan. 2008, ISSN: 1110-8657. DOI: 10.1155/2009/786015. [Online]. Available: <https://doi.org/10.1155/2009/786015>.
- [13] D. Marpe, T. Wiegand, and G. Sullivan, « The h.264/mpeg4 advanced video coding standard and its applications », *IEEE Communications Magazine*, vol. 44, 8, pp. 134–143, 2006. DOI: 10.1109/MCOM.2006.1678121.
- [14] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, and Y.-K. Wang, « Overview of the multiview and 3d extensions of high efficiency video coding », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, 1, pp. 35–49, 2016. DOI: 10.1109/TCSVT.2015.2477935.
- [15] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, « Overview of the high efficiency video coding (hevc) standard », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, 12, pp. 1649–1668, 2012. DOI: 10.1109/TCSVT.2012.2221191.
- [16] Y. Chen, G. Tech, K. Wegner, and Y. Sehoon, *Test model 11 of 3d-hevc and mv-hevc*, JCT-3V document JCT3V-G1100, Feb. 2015.
- [17] P. Merkle, C. Bartnik, K. Müller, D. Marpe, and T. Wiegand, « 3d video: depth coding based on inter-component prediction of block partitions », in *2012 Picture Coding Symposium*, 2012, pp. 149–152. DOI: 10.1109/PCS.2012.6213308.
- [18] Y.-L. Chan, C. Fu, H. Chen, and S.-H. Tsang, « Overview of current development in depth map coding of 3d video and its future », *IET Signal Process.*, vol. 14, pp. 1–14, 2020.
- [19] Y. Shoham and A. Gersho, « Efficient bit allocation for an arbitrary set of quantizers (speech coding) », *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, 9, pp. 1445–1453, 1988. DOI: 10.1109/29.90373.

- 
- [20] G. Tech, K. Müller, H. Schwarz, and T. Wiegand, « Partial depth image based re-rendering for synthesized view distortion computation », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, 6, pp. 1273–1287, 2018. DOI: 10.1109/TCSVT.2016.2631568.
- [21] G. Sanchez, L. Agostini, and C. Marcon, « Algorithms for Efficient and Fast 3D-HEVC Depth Map Encoding », in *Springer Nature*, 2019.
- [22] O. Stankiewicz, K. Wegner, and M. Domański, « Study of 3d video compression using nonlinear depth representation », *IEEE Access*, vol. 7, pp. 31 110–31 122, 2019. DOI: 10.1109/ACCESS.2019.2903087.
- [23] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, « Fast 3d-hevc depth map encoding using machine learning », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, 3, pp. 850–861, 2020. DOI: 10.1109/TCSVT.2019.2898122.
- [24] J. Lei, J. Duan, F. Wu, N. Ling, and C. Hou, « Fast mode decision based on grayscale similarity and inter-view correlation for depth map coding in 3d-hevc », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, 3, pp. 706–718, 2018. DOI: 10.1109/TCSVT.2016.2617332.
- [25] *Manual of ivde 3.0*, document ISO/IEC JTC1/SC29/WG4 MPEG2020/ N0058, Jan. 2021.
- [26] T. Senoh, N. Tetsutani, and H. Yasuda, « Depth estimation and view synthesis for immersive media », in *2018 International Conference on 3D Immersion (IC3D)*, 2018, pp. 1–8. DOI: 10.1109/IC3D.2018.8657842.
- [27] A. Blake, P. Kohli, and C. Rother, *Markov Random Fields for Vision and Image Processing*, ser. EBSCO ebook academic collection. MIT Press, 2011, ISBN: 9780262015776. [Online]. Available: <https://books.google.fr/books?id=dQE5k7DY-dAC>.
- [28] Y. Boykov and V. Kolmogorov, « An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, 9, pp. 1124–1137, 2004. DOI: 10.1109/TPAMI.2004.60.
- [29] *Test Model 10 for MPEG Immersive Video*, ISO/IEC JTC1/SC29/WG4 MPEG2021/ N0112, Jul. 2021.



- 
- [30] P. Garus, J. Jung, T. Maugey, and C. Guillemot, « Bypassing depth maps transmission for immersive video coding », in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5. DOI: 10.1109/PCS48520.2019.8954543.
- [31] P. Garus and J. Jung, *[mpeg-i visual] a 37% mv-hevc+vvs anchor improvement with 50% pixel rate reduction*, ISO/IEC JTC1/SC29/WG11 MPEG2019/m49153, Jul. 2019.
- [32] P. Garus, F. Henry, J. Jung, T. Maugey, and C. Guillemot, « Immersive video coding: should geometry information be transmitted as depth maps? », *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021. DOI: 10.1109/TCSVT.2021.3100006.
- [33] J. Boyce, B. Chupeau, and L. Kondrad, *Text of iso/iec fdis 23090-12 mpeg immersive video*, ISO/IEC JTC1/SC29/WG04 N0111, Jul. 2021.
- [34] D. P. Mitchell, « Generating antialiased images at low sampling densities », *SIGGRAPH Comput. Graph.*, vol. 21, 4, pp. 65–72, Aug. 1987, ISSN: 0097-8930. DOI: 10.1145/37402.37410. [Online]. Available: <https://doi.org/10.1145/37402.37410>.
- [35] P. Burt, « Moment images, polynomial fit filters. and the problem of surface interpolation », in *Proceedings CVPR '88: The Computer Society Conference on Computer Vision and Pattern Recognition*, 1988, pp. 144–152. DOI: 10.1109/CVPR.1988.196228.
- [36] J. Jylänki, *A thousand ways to pack the bin – a practical approach to two-dimensional rectangle bin packing*, 2010.
- [37] *Reference view synthesizer (rvs) manual*, ISO/IEC JTC1/SC29/WG11 N18068, Oct. 2018.
- [38] J. Jung, B. Kroon, R. Doré, G. Lafruit, and J. Boyce, *Ctc on 3dof+ and windowed 6dof (v2)*, ISO/IEC JTC1/SC29/WG11/MPEG2018/N17726, Jul. 2018.
- [39] G. Bjontegaard, *Calculation of average psnr differences between rd-curves*, ITU-T Q.6/16, Doc. VCEG-M33, Mar. 2001.
- [40] M. Orduna, C. Díaz, L. Muñoz, P. Pérez, I. Benito, and N. García, « Video multi-method assessment fusion (vmaf) on 360vr contents », *IEEE Transactions on Consumer Electronics*, vol. 66, 1, pp. 22–31, 2020. DOI: 10.1109/TCE.2019.2957987.

- 
- [41] Z. Wang, E. Simoncelli, and A. Bovik, « Multiscale structural similarity for image quality assessment », in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, 2003, 1398–1402 Vol.2. DOI: 10.1109/ACSSC.2003.1292216.
- [42] S. Tian, L. Zhang, L. Morin, and O. Déforges, « A benchmark of dibr synthesized view quality assessment metrics on a new database for immersive media applications », *IEEE Transactions on Multimedia*, vol. 21, 5, pp. 1235–1247, 2019. DOI: 10.1109/TMM.2018.2875307.
- [43] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, *The unreasonable effectiveness of deep features as a perceptual metric*, 2018. arXiv: 1801.03924 [cs.CV].
- [44] B. Kroon, V. Kumar Malamal Vadakital, and J. Jung, *Recommended pixel rate limits for the ctc for immersive video*, ISO/IEC JTC1/SC29/WG11/MPEG2019/M49826, Jul. 209.
- [45] E. Bosc, F. Racape, V. Jantet, P. Riou, M. Pressigout, and L. Morin, « A study of depth/texture bit-rate allocation in multi-view video plus depth compression », *3D video technologies and services, special issue of Annals of Telecommunications*, vol. 68, Jan. 2012. DOI: 10.1007/s12243-013-0363-x.
- [46] Y. Al-Obaidi, T. Grajek, and M. Domański, « Quantization of depth in simulcast and multiview coding of stereoscopic video plus depth using hevcc, vvc and mv-hevc », in *2019 Picture Coding Symposium (PCS)*, 2019, pp. 1–5. DOI: 10.1109/PCS48520.2019.8954539.
- [47] M. Tanimoto, « Ftv (free viewpoint television) creating ray-based image engineering », in *IEEE International Conference on Image Processing 2005*, vol. 2, 2005, pp. II–25. DOI: 10.1109/ICIP.2005.1529982.
- [48] A. Dziembowski, M. Domański, A. Grzelka, D. Mieloch, J. Stankowski, and K. Wegner, « The influence of a lossy compression on the quality of estimated depth maps », May 2016, pp. 1–4. DOI: 10.1109/IWSSIP.2016.7502730.
- [49] Y. Liu, J. Liu, A. Argyriou, L. Wang, and Z. Xu, « Rendering-aware vr video caching over multi-cell mec networks », *IEEE Transactions on Vehicular Technology*, vol. 70, 3, pp. 2728–2742, 2021. DOI: 10.1109/TVT.2021.3057684.

- 
- [50] Y. Liu, J. Liu, A. Argyriou, and S. Ci, « Mec-assisted panoramic vr video streaming over millimeter wave mobile networks », *IEEE Transactions on Multimedia*, vol. 21, 5, pp. 1302–1316, 2019. DOI: 10.1109/TMM.2018.2876044.
- [51] O. Stankiewicz, M. Domański, A. Dziembowski, A. Grzelka, D. Mieloch, and J. Samelak, « A free-viewpoint television system for horizontal virtual navigation », *IEEE Transactions on Multimedia*, vol. 20, 8, pp. 2182–2195, 2018. DOI: 10.1109/TMM.2018.2790162.
- [52] C. Brites, J. Ascenso, and F. Pereira, « Epipolar plane image based rendering for 3d video coding », in *2015 IEEE 17th International Workshop on Multimedia Signal Processing (MMSP)*, 2015, pp. 1–6. DOI: 10.1109/MMSP.2015.7340854.
- [53] K. Müller, P. Merkle, and T. Wiegand, « 3-d video representation using depth maps », *Proceedings of the IEEE*, vol. 99, 4, pp. 643–656, 2011. DOI: 10.1109/JPROC.2010.2091090.
- [54] G. Tech, H. Schwarz, K. Müller, and T. Wiegand, « 3d video coding using the synthesized view distortion change », in *2012 Picture Coding Symposium*, 2012, pp. 25–28. DOI: 10.1109/PCS.2012.6213277.
- [55] G. Tech, K. Müller, H. Schwarz, and T. Wiegand, « Partial depth image based re-rendering for synthesized view distortion computation », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, 6, pp. 1273–1287, 2018. DOI: 10.1109/TCSVT.2016.2631568.
- [56] K. Suehring, L. B., V. Seregin, K. Sharman, G. Tech, and A. Tourapis, *Jct-vc ahg report: software development and software technical evaluation*, ISO/IEC JTC1/SC29/WG11/MPEG20 AL0003, Jan. 2020.
- [57] K. Takeuchi, K. Okami, D. Ochi, and H. Kimata, « Partial plane sweep volume for deep learning based view synthesis », in *ACM SIGGRAPH 2017 Posters*, ser. SIGGRAPH '17, Los Angeles, California: Association for Computing Machinery, 2017, ISBN: 9781450350150. DOI: 10.1145/3102163.3102220. [Online]. Available: <https://doi.org/10.1145/3102163.3102220>.
- [58] A. Li, L. Fang, L. Ye, W. Zhong, and Q. Zhang, « Geometry-guided view synthesis with local nonuniform plane-sweep volume », in *Digital TV and Wireless Multimedia Communication*, G. Zhai, J. Zhou, H. Yang, P. An, and X. Yang, Eds., Singapore: Springer Singapore, 2020, pp. 393–403, ISBN: 978-981-15-3341-9.

- 
- [59] D. Marpe, H. Schwarz, and T. Wiegand, « Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard », *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, 7, pp. 620–636, 2003. DOI: 10.1109/TCSVT.2003.815173.
- [60] K. Müller and A. Vetro, *Common test conditions of 3dv core experiments*, Document JCT3V-G1100, JCTVC ITU-T SG16 WP3ISO/IECJTC1/SC29/WG11, Jan. 2014.
- [61] D. Mieloch and A. Dziembowski, *Put results for ee2 on coding for future mpeg immersive video*, ISO/IEC JTC1/SC29/WG4 MPEG/M54944, Oct. 2020.
- [62] J. Jung and V. Kumar Malamal Vadakital, *Report of the exploration experiments on coding for future immersive video*, ISO/IEC JTC 1/SC 29/WG 4 m54941, Oct. 2020.
- [63] B. Salahieh and J. Boyce, *Miv geometry absent*, ISO/IEC JTC 1/SC 29/WG 4 m 54874, Oct. 2020.
- [64] J. Jung and B. Kroon, *Common test conditions for mpeg immersive video*, ISO/IEC JTC 1/SC 29/WG 4/N0006, Oct. 2020.
- [65] D. Mieloch, P. Garus, M. Milovanovic, *et al.*, « Overview and efficiency of decoder-side depth estimation in mpeg immersive video », *IEEE Transactions on Circuits and Systems for Video Technology*, 2022.
- [66] P. Garus, F. Henry, G. Clare, and J. Jung, *Feature-driven decoder side depth estimation*, ISO/IEC JTC 1/SC 29/WG 4 M54994, Oct. 2020.
- [67] B. Salahieh, J. Boyce, F. Julien, C. Bertrand, and D. Renaud, *Miv view with decoder-derived depth*, ISO/IEC JTC1/SC29/WG11 MPEG2019/M54492-v2, Jul. 2020.
- [68] D. Mieloch and A. Dziembowski, *[mpeg-i visual] miv geometry absent for perspective content*, ISO/IEC JTC1/SC29/WG11 MPEG2020/M54755, Jul. 2020.
- [69] J. Jung and V. Kumar Malamal Vadakital, *Bog report: coding of future immersive video*, ISO/IEC JTC1/SC29/WG11 MPEG2020/m54787, Jun. 2020.
- [70] D. Mieloch and A. Dziembowski, *The comparisons of encoder-side and decoder-side depth estimation*, ISO/IEC JTC1/SC29/WG04 MPEG/57349, Jul. 2021.
- [71] D. Mieloch, A. Dziembowski, B. Szydełko, D. Klóska, G. Lee, and J. Y. Jeong, *[miv] decoder-side depth estimation with extended input depth assistance*, ISO/IEC JTC 1/SC 29/WG 04 m59516, Apr. 2022.

- 
- [72] D. Mieloch, A. Dziembowski, and M. Domański, « Depth map refinement for immersive video », *IEEE Access*, vol. 9, pp. 10 778–10 788, 2021.
- [73] D. Mieloch, D. Klóska, and M. Woźniak, « Point-to-block matching in depth estimation », 2021.
- [74] J. Jung, V. Kumar Malamal Vadakital, and D. Mieloch, *Exploration experiments on future mpeg immersive video*, ISO/IEC JTC1/SC29/WG04 MPEG/N0115, Jul. 2021.
- [75] B. Szydełko, D. Mieloch, D. Adrian, G. Lee, and J. Young Jeong, *Title rectangular blocks in encoder-derived features for decoder-side depth estimation*, ISO/IEC JTC1/SC29/WG04 MPEG/M56335, Apr. 2021.
- [76] G. Clare, P. Garus, F. Henry, *et al.*, *[miv] combination of m56626 and m56335 for geometry assistance sei message*, ISO/IEC JTC1/SC29/WG04 MPEG/M56950, Apr. 2021.
- [77] B. Szydełko, A. Dziembowski, and D. Mieloch, *Analysis of further block splitting in feature-driven dsde*, ISO/IEC JTC1/SC29/WG04 MPEG/m57347, Jul. 2021.
- [78] B. Szydełko, A. Dziembowski, D. Mieloch, M. Domański, G. Lee, and J. Young Jeong, *Effectiveness of recursive splitting in feature extraction for subset of views*, ISO/IEC JTC1/SC29/WG04 MPEG/m58334, Oct. 2021.
- [79] B. Szydełko, A. Dziembowski, D. Mieloch, M. Domański, and G. Lee, « Recursive block splitting in feature-driven decoder-side depth estimation », *Etri Journal*, vol. 44, Feb. 2022. DOI: 10.4218/etrij.2021-0308.
- [80] B. Szydełko, A. Dziembowski, D. Mieloch, M. Domański, G. Lee, and J. Young Jeong, *Partial geometry assistance information*, ISO/IEC JTC1/SC29/WG04 MPEG/m58047, Oct. 2021.
- [81] D. Klóska, D. Mieloch, A. Dziembowski, and M. Domański, *Decoder-side depth estimation with input depth assistance*, ISO/IEC JTC1/SC29/WG04 MPEG/m58048, Oct. 2021.
- [82] Q. Wang, Z. Wang, K. Genova, *et al.*, *Ibrnet: learning multi-view image-based rendering*, 2021. DOI: 10.48550/ARXIV.2102.13090. [Online]. Available: <https://arxiv.org/abs/2102.13090>.



---

**Titre :** Compression et synthèse pour la représentation de contenus immersifs adaptés à 6DoF

**Mot clés :** Immersive Video Coding, Video Compression, Decoder Side Depth Estimation, MPEG Immersive Video, Rendering, Multiplane Images

**Résumé :** Une nouvelle norme de codage vidéo immersive a été finalisée par le Moving Picture Experts Group (MPEG). Il s'agit de la norme MPEG Immersive Video (MIV), MPEG-I Part 12. La norme MIV peut être utilisée pour permettre une navigation libre dans une scène. Cependant, il ne faut pas s'attendre à une compression appropriée de la géométrie, car les codecs vidéo 2D largement utilisés ne prennent pas en charge les outils dédiés au codage de la profondeur. En outre, la compression de la géométrie présente plusieurs inconvénients, tels que des exigences plus élevées en termes de débit binaire et de taux de pixel.

Dans cette thèse, *Decoder Side Depth Es-*

*imation (DSDE)* est proposé et développé comme un système de codage alternatif au MIV, qui offre un gain de codage significatif, des économies de taux de pixel et une meilleure qualité perceptuelle. Nous proposons en outre plusieurs nouvelles améliorations de DSDE, impliquant la transmission partielle de la géométrie, la transmission d'informations latérales et l'exploitation du flux binaire de la texture afin d'améliorer encore le gain de codage et de réduire la complexité. Enfin, nous montrons que nos propositions peuvent être utilisées pour améliorer les performances de méthodes de rendu plus récentes, basées sur les réseaux neuronaux, comme les images multiplans.

---

**Title:** Compression and Synthesis for Representation of Immersive Content for 6DoF

**Keywords:** Immersive Video Coding, Video Compression, Decoder Side Depth Estimation, MPEG Immersive Video, Rendering, Multiplane Images

**Abstract:** A novel immersive video coding standard has been finalized 2022 by the Moving Picture Experts Group (MPEG) denoted as MPEG Immersive Video (MIV), MPEG-I Part 12. The MIV standard can be used to enable free navigation within a scene. However, appropriate compression of the geometry cannot be expected as widely used 2D video codecs do not support dedicated depth coding tools. Furthermore, several disadvantages like higher bit- and pixel rate requirements are connected with the compression of geometry.

In this thesis *Decoder Side Depth Estimation (DSDE)* is proposed and further devel-

oped as an alternative coding system to MIV, which provides significant coding gain, pixel rate savings and improved perceptual quality. We further propose various novel improvements of DSDE, involving the partial transmission of geometry, the transmission of side information and the exploitation of the texture bitstream in order to further improve the coding gain and to reduce the complexity. Finally, we show that our proposals can be used to enhance the performance of more recent, neural network-based rendering methods like multiplane images.