Synthesized View Distortion Computation by Rendering for Depth Map Encoding

vorgelegt von
Dipl.-Ing.
Gerhard Tech
geb. in Aachen

von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Giuseppe Caire, Ph.D.
Gutachter:	Prof. Dr.-Ing. Jens-Rainer Ohm
	Prof. Mårten Sjöström, Ph.D.
	Prof. Dr.-Ing. Thomas Wiegand

Tag der wissenschaftlichen Aussprache: 3. Dezember 2018

Berlin 2019

To Charlotte

Acknowledgments

I would like to thank everybody at Fraunhofer HHI who supported me. In particular, I am grateful to Prof. Dr.-Ing. Thomas Wiegand, Dr.-Ing. Ralf Schäfer, Dr.-Ing. Detlev Marpe for giving me the opportunity to work on this thesis. Furthermore, I would like to thank Prof. Dr.-Ing. Heiko Schwarz and Dr.-Ing. Karsten Müller for reviewing this thesis and for their advice. My sincere thanks also go to Prof. Dr. Giuseppe Caire, Prof. Dr.-Ing. Jens-Rainer Ohm, Prof. Dr. Mårten Sjöström, and Prof. Dr.-Ing. Thomas Wiegand for their efforts as members of the thesis committee. I would like to thank also my parents, Dipl.-Ing. Johannes Gerhard Tech and Hildegard Tech, for their support and for encouraging me to become an engineer and writing a thesis. My biggest thanks go to Charlotte Pauli for supporting me almost the whole time I was concerned with this thesis with her love and patience.

Contents

Notations

Frequently used abbreviations

IV ..... *Input View, a recorded camera view used for rendering*

SV ..... *Synthesized View, a virtual view generated by rendering*

SVD ..... *Synthesized View Distortion, the distortion of the SV's texture*

SVDC ..... *Synthesized View Distortion Change, the measure presented in this thesis*

LCR ..... *Low Complexity Renderer, the rendering algorithm presented in this thesis*

RM ..... *Renderer Model, the SVDC derivation method presented in this thesis*

Symbols

s followed by a subscript ..... *signals and their types*

E.g.

s_T	texture	s_O	occlusion signal	s_H	hole map	s_D distortion
s_M	depth map	s_Z	distance	s_P	disparity	

x and y ..... *horizontal and vertical positions*

B ..... *regions*

D ..... *distortions*

R ..... *bit rates*

J ..... *rate-distortion costs*

T ..... *encoding times*

Lowercase Latin or Greek characters ..... *other constants and variables*

E.g. w is the picture width; λ is the Lagrange multiplier.

Lowercase bold characters ..... *vectors*

E.g. **c** is a vector denoting the camera projection center.

Uppercase bold characters ..... *matrices*

E.g. **P** is a projection matrix.

Calligraphic characters ..... *entities and sets*

E.g. $\mathcal{V}$ is a view synthesis method; $\mathcal{S}_\mathcal{R}$ is a set of intermediate rendering results.

Accents and subscripts

$'$ accent ..... *in SV*

When marked with $'$, a signal, position, or region belongs to an SV, otherwise to an IV.

E.g. s'_T, x', and B' are a texture, a position, and a region, respectively, of an SV; s_T, x, and B are a texture, a position, and a region, respectively, of an IV.

l or r subscripts ..... *left or right view*

When an IV signal is marked with l or r, it belongs to a left IV or right IV, respectively.

E.g. $s_{T,l}$ and $s_{T,r}$ are the SV textures of a left IV and right IV, respectively.

When an SV signal is marked with l or r, it is rendered from a left IV or right IV, respectively.

E.g. $s'_{T,l}$ is the SV texture rendered from the left IV.

$\tilde{}$ accent ..... *coded or distorted*

When a signal, position, or region is marked with $\tilde{}$, it is coded or distorted; or it is derived from coded or distorted data.

E.g. $\tilde{s}_M$ is a coded IV depth; $\tilde{s}'_T$ is an SV texture rendered from distorted data.

$\dot{}$ and $\ddot{}$ accents ..... *before and after a depth change*

In some contexts, signals, or variables are regarded before and after a change of the depth map. Then, $\dot{}$ marks the version before the depth change and $\ddot{}$ the version after the depth change.

E.g. $\dot{s}'_T$ and $\ddot{s}'_T$ are the SV textures before and after the depth change, respectively.

$\hat{}$ accent ..... *upsampled*

When a signal or position is marked with $\hat{}$, it is an upsampled signal or a position in an upsampled signal.

E.g. $\hat{s}_{T,l}$ is the upsampled version of the left IV texture $s_{T,l}$. $\hat{x}$ is a position in $\hat{s}_{T,l}$

$\check{}$ accent ..... *rounded*

When a position is marked with $\check{}$, it is a rounded position.

E.g. $\check{x}'$ is the rounded version of the SV position x'.

s or e subscripts ..... *start or end*

When a position is marked with s or e, it is the start or end of an (1D) region.

E.g. $x'_{C,s}$ is the start of the SV region $\tilde{B}' = [x'_{C,s}, x'_{C,e}]$.

The current most perfect photographic print only shows one aspect of reality; it reduces to a single image fixed in a plane, similar to a drawing or a hand-drawn painting. The direct view of reality offers, as we know, infinitely more variety. We see objects in space, in their true size, and with depth, not in a plane. Furthermore, their aspect changes with the location of the observer; the different layers of the view move with respect to one another; the perspective gets modified, the hidden parts do not stay the same; and finally, if the beholder looks at the exterior world through a window, he has the freedom to see the various parts of a landscape successively framed by the opening, and as a result, different objects appear to him successively.

GABRIEL LIPPMANN
Nobel laureate for reproducing colors photographically

Épreuves réversibles donnant la sensation du relief
Journal de Physique Théorique et Appliquée, 1908.
Translation by F. Durand [Dura 16]

1 Introduction

In recent years, stereoscopic 3D video became widely available in cinema and for home entertainment. Nevertheless, most of today's commercially available 3D displays are not capable of representing the reality as described by Lippmann [Lipp 08]. Although they achieve a 3D impression by stereopsis as they present different views to the eyes of an observer, the perceived scene is static with respect to his viewing position. Moreover, the observer usually requires glasses, which reduces the acceptance of 3D video particularly in home environments.

An alternative technology, which is based on Lippmann's pioneer work and reduces these drawbacks, are auto-stereoscopic 3D displays [Okos 80]. Auto-stereoscopic 3D displays emit different views of a captured scene to different spatial positions. This way, different views can be perceived by the left and the right eye of an observer at certain viewing positions. An enhancement of this technology are displays that not only emit two, but a range of views depicting the scene from a large number of slightly different perspectives. Such multiview auto-stereoscopic displays not only enable an observer to perceive the 3D video without glasses at an increased number of viewing positions, but also have the effect that he can slightly look around objects when moving his head [Dodg 05].

However, these advantages come at the cost that a large number of views need to be provided to an auto-stereoscopic display. As recording of a large number of views is not feasible in conventional application scenarios, depth-based 3D video formats have been proposed. More specifically, depth-based 3D video comprises recorded camera data depicting several views of

a 3D scene (called textures) and associated data representing the distance of the sampled 3D scene from the camera plane (called depth maps) [Smol 09]. The depth maps can be used to re-sample the recorded textures in order to render textures depicting the 3D scene from different perspectives, which is called depth image based rendering (DIBR). With DIBR, textures of additional views (called synthesized or virtual views) can be generated for an auto-stereoscopic display so that it is unnecessary to capture textures of a large number of views.

The reduction of the required number of views by depth-based formats is not only of advantage for 3D video acquisition, but also for its transmission. Considering e.g. that it is possible to run a 28-view auto-stereoscopic display with texture and depth of three views only, a significant bit rate reduction has already been achieved only by the choice of the data format. However, on the other hand, today's transmission channels are conventionally designed for bit rates of 2D content and thus for the texture of one view, and not for texture and depth of three views. From this, it becomes clear that there is a demand for improved compression techniques to further increase the transmission efficiency for 3D video.

In response to this demand, ITU-T VCEG and ISO/IEC MPEG formed the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V), which extended the existing AVC [ITU 07] and HEVC [ITU 13] video coding standards for efficient compression of depth-based 3D video. Two of the resulting extensions—3D-AVC and 3D-HEVC—provide dedicated coding tools for joint compression of texture and depth. However, the scope of standardization are the bitstream format and the decoding process. Thus, as complement to these new decoding processes, new encoding techniques especially tailored for depth-based 3D video can further increase compression efficiency.

An important encoding technique is rate-distortion optimization, which selects coding modes based on the number of required bits and the introduced distortion [Sull 98]. In texture coding, the distortion introduced by a coding mode can be computed directly by reconstructing the texture samples and comparing them to uncoded texture samples. In depth coding, such a conventional direct computation of the depth distortion is not reasonable because the depth is only transmitted for rendering and not presented directly to an observer. Thus, compression efficiency can be substantially increased by considering the distortion of the synthesized views while encoding the depth [Kim 09]. Consequently, several methods have been proposed that estimate the synthesized view distortion, e.g. [Oh 11, Kim 09]. However, as they provide only an estimate, an increased rate-distortion performance can be expected for an exact computation.

This insight is the motivation for the subject of this thesis—Synthesized View Distortion Computation by Rendering for Depth Map Encoding—which addresses in particular the following three questions:

1. How can a rendering-based synthesized view distortion measure be defined?
2. How can it be implemented with low complexity?
3. How can it be used optimally in encoding?

Answers to these questions enable an improved compression of depth-based 3D video and consequently a more realistic representation of 3D scenes.

1.1 Overview of the 3D Video System

Compared to 2D or stereo video, the generation, transmission, and presentation of depth-based 3D video requires additional processing steps. An exemplary 3D video system including these steps is shown in Figure 1.1 and discussed in the following to define the scope of and preliminaries for methods and evaluations presented in this thesis. The system is similar to the one assumed by JCT-3V for the development of the 3D extensions for AVC and HEVC [Mull 14].

1.1.1 Acquisition

The left side of Figure 1.1 shows the sender-side of the 3D video system. In its first processing step, the 3D scene is captured with a *multi-camera system*. In general, the number of cameras and their spatial positions depend on the application scenario: E.g., setups with cameras placed on a line or a plane in front of the 3D scene, or on an arc, circle, or sphere surrounding the scene have been proposed with parallel or non-parallel optical axes. Specifically, this thesis focuses on a setup with equally rotated cameras, which are placed on a horizontal line such that their optical axes are parallel. Such a linear and parallel setup simplifies view synthesis, as discussed later, and is required by most auto-stereoscopic displays. For simplification and without loss of generality, the test data used in this thesis comprises only views of two cameras. For recording, the cameras have been located in approximately double stereo distance ($\approx$ 13 cm), which is small enough to still allow reliable synthesis of intermediate views. Furthermore, the cameras have been calibrated so that their internal and external camera parameters are known, and they have been synchronized in a way that they sampled the 3D scene at identical instants of time.

After capturing the next step is conventionally *texture processing*, which improves the alignment of the captured texture views. An example for texture processing is the correction of color mismatches between cameras caused by different sensors or illumination. Another example is the rectification of textures, which is necessary in case that cameras have not been properly arranged. Such a rectification has been performed for test data used in this thesis such that a point of the 3D scene is projected to the same vertical position in all views.

As next step in the 3D video system, the rectified and color corrected textures are used for *depth estimation*. Depth estimation exploits the geometrical relationship between disparity and depth. The actual depth of a point in a 3D scene is calculated from the disparity between its projected position in the textures of different views using the camera parameters. For this, a major target of depth estimation is finding corresponding points in different views, which is commonly called stereo matching [Scha 02]. The algorithm that estimated the depth data of non-computer-generated sequences used in this thesis has been provided by MPEG and is known as Depth Estimation Reference Software (DERS) [Tani 09a]. DERS provides a full sample-dense depth map, i.e. one depth sample per sample of each texture picture of each view.

Depth estimation is not necessary when depth data is directly available, e.g. for computer-generated content or from technologies capturing the depth data directly from the 3D scene, as e.g. approaches based on structured light [Salv 04] or time-of-flight sensors [Foix 11]. Al-

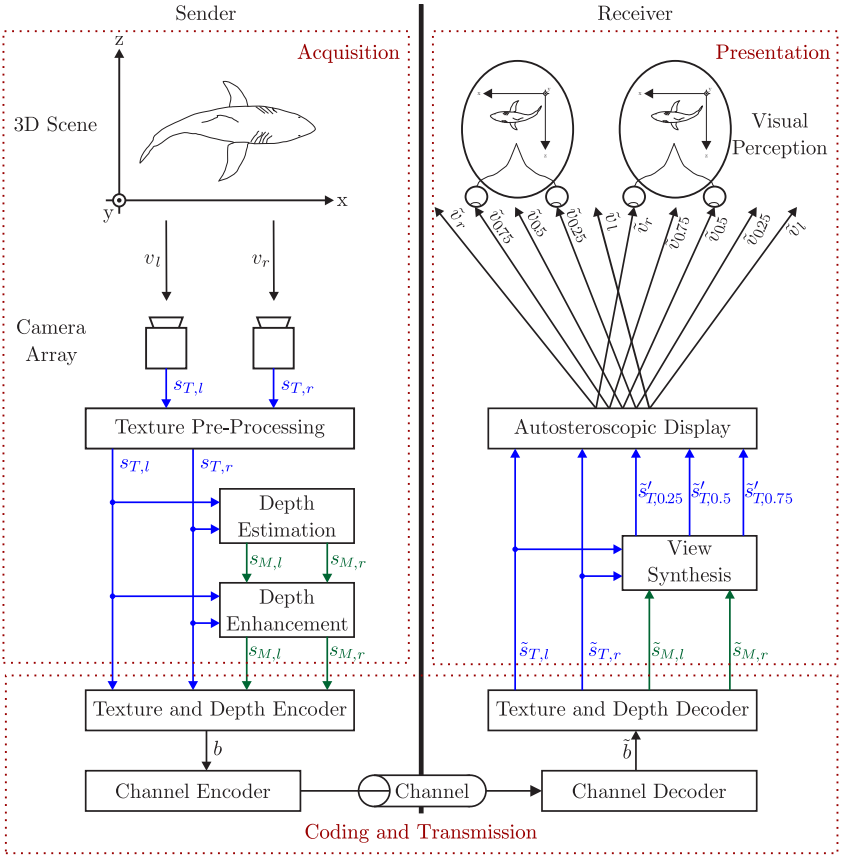


Figure 1.1: Depth-based 3D video system including acquisition, coding and transmission, and presentation of 3D video. Blue: Texture signals; Green: Depth signals.

though such methods for depth capturing showed progress in the recent years, achieved temporal or spatial resolutions are still not comparable to those provided by texture cameras [Lang 12]. Consequently, the recorded depth data needs to be interpolated before applying it in 3D video [Park 11]. The 3D video test set used in this thesis does not include sequences with depth data captured from natural scenes. However, it includes some computer generated sequences with computer generated depth maps.

After depth estimation (or capturing) depth maps may comprise signal parts irrelevant for rendering or noise. To reduce both an optional step is *depth enhancement*. A removal of irrelevant high-frequent signal parts, which not necessarily cause artifacts, is in particular beneficial to increase the efficiency of a subsequent encoding. The removal of noise primarily targets erro-

neous depth values that create artifacts in the synthesized views. A popular method for this is e.g. an alignment of texture and depth edges [Lai 10]. However, depth enhancement can also be (semi-automatic) manual editing, potentially even with feed-back to the estimation step. This has been applied to some depth maps in the test set used in this thesis [Tani 09a].

1.1.2 Coding and Transmission

After texture and depth have been acquired, they are encoded at the sender-side using a video and a channel encoder, transmitted via a channel, and decoded at the receiver-side using corresponding channel and video decoders.

For 3D *video coding*, actually any conventional 2D video codec could be used to compress texture and depth signals separately. However, higher compression can be achieved by using video codecs especially designed for multiview texture and depth data. Here, two different types of codecs can be distinguished. Codecs of the first type exploit redundancies between different views by inter-view prediction without modifying core parts of a corresponding single view base codec. Examples for this are MV-HEVC [Tech 16] or MVC [Vetr 11] conforming codecs. Advantage of these codecs is that they can easily be implemented based on existing HEVC [Sull 12] or AVC [Wieg 03] conforming hardware. Codecs of the second type, e.g. 3D-HEVC [Tech 16] or 3D-AVC conforming codecs, additionally use new coding tools for e.g. improved inter-view prediction, prediction between texture and depth or exploiting special signal characteristics of depth. This way, encoding efficiency increases at the cost of a more complex codec design. Specifically, this thesis employs 3D-HEVC, which is currently the most advanced 3D video coding standard. Evaluation and methods discussed in this thesis are based on the software that is provided by JCT-3V as reference for 3D-HEVC, and which is known as HTM [JCT 16].

For transmission over a lossy channel, *channel coding* is usually applied to the bitstream created by the video encoder, which adds signalling and redundancies to detect or correct losses. For 3D or stereoscopic video several methods [Akar 07] have been proposed, as e.g. unequal error protection of texture and depth [Hewa 09]. However, specifically this thesis assumes an error-free channel. Consequently, it does not address channel coding, although some of the synthesized view distortion metrics discussed in Part I could also be used to e.g. prioritize protection of certain parts of the bitstream.

1.1.3 Presentation

After decoding of the transmitted bitstream, textures of additional (virtual) views depicting the 3D scene from additional (virtual) camera positions are generated by DIBR-based *view synthesis*. The principle of DIBR is to render the virtual view's texture by projecting the texture samples of one or more recorded views to the coordinate system of the virtual view. Such a projection corresponds to a relative shift of the recorded texture samples by disparity vectors calculated from their associated depth sample values and the recorded and virtual view's camera

parameters. After projection further steps are conventionally applied, which are e.g. handling of occlusions (when two or more recorded texture samples are projected to the same position in the virtual view) or holes (when no recorded view's texture sample is projected to a particular position). These steps and view synthesis in general will be further discussed in Part I with focus on an own low-complexity algorithm, which is also suitable for encoder-side rendering, and a renderer provided by MPEG, which is available in a software package known as View Synthesis Reference Software (VSRS) [MPEG 10b].

The 3D video system synthesizes textures of virtual views as it conventionally transmits texture and depth of only a few recorded views, although auto-stereoscopic displays require usually more views from a linear and parallel camera setup. This means that virtual camera positions are located on a line defined by recorded view's camera positions and that all optical camera axes are perpendicular to this line and parallel to each other.

For presentation of the 3D scene, the synthesized and decoded textures of virtual and recorded views are provided to the multiview *auto-stereoscopic display*. Conventionally, auto-stereoscopic displays comprise two major components [Urey 11]: A display panel showing the provided views in an interleaved fashion (e.g. horizontally) and a system in front of the panel ensuring that only samples of one view are emitted to a certain point in front of the display (e.g. parallax barrier system or a lenticular lenses). This way, different spatial viewing cones occur in front of the display. In each cone, only one view is visible such that neighboring cones show neighboring views successively from left to right. This sequence from left to right is usually called viewing lobe and repeated multiple times, as depicted in Figure 1.1. The repetition can be avoided by increasing the viewing lobe size and thus by increasing the number of presented views. Motivated by this, prototypes of super-multiview displays have been proposed emitting up to 256 views [Taka 10].

The final step in the 3D video system is the *visual perception* of emitted views by an observer. More specifically, his eyes perceive two different views of a viewing lobe. These views are fused by the visual system to a single scene. Furthermore, the visual system creates a depth sensation of the scene by two different cues provided by the display: First, majorly by stereopsis, thus from the disparities between objects in the two perceived views [Whea 38]; and second, as an auto-stereoscopic display enables to move the head within a viewing lobe, also by motion parallax.

1.2 Scope and Contributions

This thesis targets to improve the 3D video system by increasing the rate-distortion (RD) performance in the encoding step. More specifically, its objective is to reduce the bit rate of the encoded depth data at constant quality of the synthesized views presented at the receiver-side. Processing steps that are relevant for this are shown in the sub-system in Figure 1.2. The sub-system encodes, transmits, and decodes texture and depth signals of a 3D video sequence, called input view (IV) textures and depth maps. Then, a view synthesis method uses the reconstructed IV textures and depths to render textures of virtual views, called synthesized view (SV) tex-

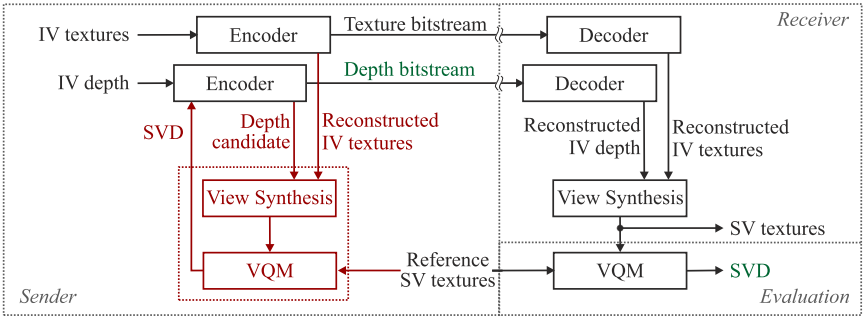


Figure 1.2: Sub-system of the 3D video system including components relevant in this thesis. Red: The synthesized view distortion (SVD) derivation method developed in this thesis.

tures. Besides these conventional processing steps, Figure 1.2 shows an additional step: For evaluation in this thesis, a video quality metric (VQM) compares the SV textures to reference SV textures, which can e.g. be rendered from uncoded texture and depth. The result of the comparison is the receiver-side synthesized view distortion (SVD).

Considering the sender-side, encoders perform usually rate-distortion optimization to increase the compression performance. This means that when conventional encoders are used for depth coding, they decide on whether to select a particular coding mode considering the distortion of the reconstructed depth values (called depth candidate) that are associated with that mode. Nevertheless, as the sub-system outputs not the reconstructed depth itself, but the rendered SV textures, an improvement can be achieved by considering the SVD that would occur at the receiver-side. For this, approaches have been proposed that estimate the receiver-side SVD [Oh 11, Kim 09, Kim 10, Wang 13, Yuan 14, Ma 13] at the encoder.

However, a higher compression efficiency can be achieved when the SVD determined at the encoder-side is not an estimate, but matches exactly with the receiver-side SVD. For this, Part I of this thesis develops an advanced encoder-side distortion derivation method. As shown in Figure 1.2, the advanced method provides an exact SVD measure by 1) rendering the SV texture with the same method as the receiver; and by 2) computation of the SVD with the same quality metric as the receiver. Then, Part II applies, optimizes, and evaluates the advanced method in depth encoding.

More specifically, contributions of this thesis addressing the development of such an advanced rendering-based method and its application in encoding are:

1. *A low-complexity view synthesis algorithm:* As computational resources are conventionally limited, complex view synthesis methods are not suitable for rendering in the rate-distortion optimization stage. To address this, this thesis develops a low-complexity rendering algorithm (LCR) by selecting, combining, and optimizing basic approaches. The algorithm comprises, on top of a minimal warping and hole filling approach, sub-sample precision, and view interpolation. This way, the complexity is minimized while preserving a sufficient

rendering quality for distorted depth and a high quality for undistorted depth. Key feature of the algorithm is that it can be easily extended by partial re-rendering, which is required for SVD computation in RD optimization. However, it can also be applied as receiver-side renderer.

2. *A concept for exact SVD derivation:* This thesis analyzes the relationship of distorted regions in the IV depth and the SV texture and shows that they do not map bijectively. This complicates the derivation of an additive and block related SVD measure, which is required in encoding. Therefore, this thesis develops the concept of the synthesized view distortion change (SVDC), which fulfills both requirements. The SVDC is the distortion change of a synthesized view texture that is related to an IV depth change. As exact measure, the SVDC outperforms estimation methods and cannot only be applied in encoding, but also by other depth processing algorithms, as e.g. depth estimation or enhancement.
3. *An extension of the LCR by partial re-rendering:* The SVDC can be determined by re-rendering only regions of the synthesized view that are affected by a depth change. To this end, this thesis provides an analysis revealing regions that actually change. The LCR is modified such that partial re-rendering of affected regions is enabled with low-complexity. More specifically, modifications allow a) to start re-rendering from an arbitrary position in the depth map, b) to determine while re-rendering the position at that the affected region is updated, and c) to stop at this position. The modified LCR is embedded in a framework, called renderer model (RM), which holds the current state of the rendering process and provides an interface to depth processing algorithms for SVDC computation.
4. *An optimization of SVDC-based encoding using the RM:* This thesis explores how to exploit the RM optimally in depth encoding. This includes the following contributions:

   - *An SVDC-based depth encoder:* The RM is integrated in an HEVC compliant encoder[1]. For this purpose, distortion computation modules conventionally used by the mode selection process are replaced with the RM. Additional functionality for synchronizing SVDC computation and encoding are added. The resulting SVDC-based encoder achieves (in an initial setup), depth bit rate reductions of about 74% at the same SV quality compared to a depth distortion-based encoder.
   - *An analysis how the RM's features contribute to the achieved depth bit rate savings:* For distortion computation, the RM renders with quarter sample precision, performs view combination, calculates the distortion of chroma components, and computes the SVDC. The evaluation shows how these different features of the RM contribute to the achieved depth bit rate savings and encoding complexity. It furthermore reveals a problem, called hiding effect, of SVDC-based mode selection that is caused by the coding order of texture and depth and the RM's view combination process.
   - *View combination variants mitigating the hiding effect:* To mitigate the hiding effect, different view combination variants are suggested and evaluated. The optimal variant achieves depth bit rate reductions of 9.6% compared to the initial setup.
   - *A classification and evaluation of different SV generation setups:* The RM computes the

[1] The encoder has been meanwhile further developed by JCT-3V to the 3D- and MV-HEVC reference encoder HTM.

SVDC by comparing a tested SV texture to a reference SV texture. The classification categorizes different options to generate them from coded or uncoded texture and depth and by view interpolation or extrapolation. The evaluation investigates their RD performance. The optimal setup achieves depth bit rate reductions of 10.4% compared to the initial setup. Results show furthermore, which RD performance losses occur when encoding views or texture and depth independently from each other.

- *An overall optimization of the SVDC-based encoder considering a depth distortion term:* To preserve the quality of the depth map in SVDC-based encoding, a depth distortion term has been proposed [Jung 12c]. The optimization evaluates different depth weights, in combination with different SV generation setups and view combination variants, with respect to their RD performance, encoding complexity, and depth quality. The optimal combination (referred to as optimized encoder setup in the following) achieves a depth bit rate reduction of 10.7% compared to the initial setup.
- *An analysis of different mode selection stage setups*: The initial setup uses SVDC computation only in the highest mode selection stage. The lower stages for the encoding of the residual quadtree; the pre-selection of intra-prediction modes; and the selection of wedgelet parameters, inter-prediction modes, and motion parameters use the depth distortion. For analysis, the RM is integrated to all selections stages such that it can be enabled individually for particular selection stages. An evaluation of different mode selection setups shows achievable complexity-RD performance trade-offs. Furthermore, the analysis shows that considering the SVDC in all selection stages achieves depth bit rate savings of 6.6% compared to the optimized encoder setup while the depth encoding time increases by a factor of 3.5.
- *An investigation of optimal Lagrange multipliers and quantization parameters (QPs)*: The initial SVDC-based encoder uses the SVDC only in the highest mode selection stage; lower-level stages employ the depth distortion. The investigation analyzes how the Lagrange multiplier used in SVDC-based and depth distortion-based stages are related to each other and to the QP. Results show that the RD performance of default values is close to the optimum and that consequently achievable coding gains using constant values are minor.

5. *Evaluations of the LCR, the RM, and the optimized depth encoder:* This thesis provides a broad assessment of the contributed methods in various application scenarios and under various test conditions:

   - *A complexity evaluation of the LCR and the RM:* The complexity of the LCR and the RM is assessed based on their required number of operations. For the RM, it is furthermore evaluated how the complexity depends on characteristics of the depth change, i.e. on its magnitude and on the size of the changed region in the depth map. Results enable a comparison of the RM's complexity to other distortion derivation methods. Furthermore, the effectiveness of the RM's optimizations is discussed.
   - *An analysis of mode selection characteristics when using SVDC-based encoding:* SVDC-based encoding reduces or distorts signal parts of the depth map that are irrelevant for rendering. An evaluation shows how this affects the statistics of chosen encoding modes.

- *An evaluation of different SV position setups*: Because of its computational complexity, SVDC computation can only regard a small number of SVs. The evaluation explores how the number and the positions of the considered SVs affect the coding gains at SV positions not regarded in SVDC computation and the encoding complexity. It reveals that considering three SVs provides a reasonable trade-off between complexity and RD performance.
- *A comparison of SVDC-based encoding to SVD estimation-based encoding:* An alternative to SVDC computation are SVD estimation methods. Three SVD estimation methods are discussed and optimized. In comparison to them, SVDC-based encoding achieves about 40% depth bit rate reduction.
- *An evaluation and optimization of SVDC-based encoding with respect to further aspects:* SVDC-based encoding is optimized and evaluated considering different receiver-side view synthesis methods and alternative distortion metrics. Furthermore, sample pictures revealing how PSNR gains are related to visual improvements are provided. A final evaluation shows that SVDC-based encoding achieves reductions of about 19% of the total bit rate compared to depth distortion-based encoding under test conditions provided by the JCT-3V.

Several contributions of this thesis have already shown their practical importance. Known as view synthesis optimization (VSO), SVDC computation with the RM is part of the 3D-HEVC reference software encoder HTM [JCT 16, Chen 15] since it was proposed as starting point for the development of 3D-HEVC [Schw 11a]. It is, according to JCT-3V's test conditions [Mull 14], the main method for distortion derivation in RD-based mode selection in depth coding [Chen 15]. Moreover, the LCR (and thus the RM) have been developed in parallel with the VSRS 1D-Fast software, which is an own contribution, operates similar to the LCR, and was the main renderer in the development of 3D-HEVC [Chen 15]. In contrast to the LCR, the VSRS 1D-Fast does not support re-rendering, but boundary noise removal.

Furthermore, several own papers concern or are related to the contributions of this thesis, e.g. [Tech 18, Tech 12c, Tech 12d, Tech 11, Tech 10]. In particular, major parts of the text and several figures in Chapters 3 to 5 have been published in [Tech 18]. A patent considering SVDC computation has been granted.

1.3 Outline

This thesis comprises two parts. *Part I "Synthesized View Distortion Computation"* addresses the questions on how to define and implement a rendering-based SVD measure.

- Chapter 2 "View Synthesis and the Synthesized View Distortion" reviews the basics of DIBR and existing view synthesis methods. Furthermore, it analyzes the SVD and existing methods for SVD derivation.
- Chapter 3 "Low-Complexity Depth Image Based Rendering Algorithm" develops a low-complexity algorithm, which is later extended in Chapter 5 by partial re-rendering for SVD

computation. The algorithm is evaluated with regard to complexity and synthesis quality.

- Chapter 4 “The Synthesized View Distortion Change” develops the concept of the synthesized view distortion change (SVDC), which is an additive and block related SVD measure and resolves the problem that distorted regions in a depth map and in a synthesized view do not map bijectively.
- Chapter 5 “Partial Depth Image Based Re-Rendering for SVDC Computation” analyzes the relationship between the depth map and the synthesized view to determine which parts of the depth map need to be processed for SVDC computation and extends the low-complexity rendering algorithm developed in Chapter 3 by partial re-rendering.

Part II “Synthesized View Distortion Based Encoding” discusses how SVDC computation can be used optimally in the scope of depth encoding.

- Chapter 6 “Basic 3D Video Coding Concepts and Methods” discusses 3D video coding on the example of 3D-HEVC.
- Chapter 7 “Encoder Integration and Initial Evaluation” discusses the encoder and the framework used for evaluations in this thesis, summarizes modifications of the coding mode selection process required for SVDC-based encoding using the RM, and discusses test conditions and configuration options. Furthermore, it provides an initial analysis of the encoder using the RM and evaluates how encoding and RD performance change when using the SVDC instead of the depth distortion.
- Chapter 8 “Encoder Optimization” optimizes and analyzes SVDC-based encoding with respect to complexity and RD performance considering 1) different features of the RM, 2) different options to generate the tested and the reference synthesized view textures used for SVDC computation, 3) an additional depth distortion term, 4) the usage of SVDC-based encoding in lower-level mode selection stages, and 5) the selection of Lagrange multipliers and quantization parameters.
- Chapter 9 “Evaluation and Optimization Using Other Test Conditions” analyzes the performance of SVDC-based encoding considering 1) varying numbers of SVs, 2) different rendering approaches, 3) estimation methods, 4) alternative distortion metrics, 5) sample pictures, and 6) JCT-3V’s test conditions.

Finally, a summary and the conclusions of this thesis’ two parts are provided.

Part I

Synthesized View Distortion Computation

2 View Synthesis and the Synthesized View Distortion

The methods presented in this thesis generate SV textures using DIBR. For this, this chapter provides an overview of commonly known (e.g. [Schr 05]) DIBR concepts and methods in Section 2.1. Then, it discusses properties of the SVD and methods to derive the SVD in Section 2.2.

2.1 View Synthesis by Depth Image Based Rendering

This section discusses plenoptic sampling and point correspondence in the 3D scene, i.e. the foundations of DIBR (Section 2.1.1). Furthermore, it provides an overview of problems that occur in DIBR and discusses basic and advanced methods to resolve them (Section 2.1.2).

2.1.1 Theoretical Foundations

A model describing the 3D scene is the plenoptic function [Adel 91], which provides the intensity I of light with a wavelength λ that is observed from a point at coordinates (x_s, y_s, z_s) in a 3D space at a time t when looking in the direction specified by the spherical coordinates (θ, φ):

$$I = \mathrm{f_P}(x_s, y_s, z_s, \theta, \varphi, \lambda, t) \tag{2.1}$$

As the plenoptic function describes the entire visual space, recording of a 3D scene corresponds to the sampling of a subset of the space spanned by the plenoptic function's parameters. Similarly, view synthesis can be interpreted as reconstruction of the plenoptic function at certain points from the sampled data and thus as re-sampling process. When the data of the sampled subset is represented by a set of images, view synthesis is also called Image Based Rendering (IBR). In the recent decades, a great variety of IBR methods and associated models to parametrize the plenoptic function have been proposed [Koch 05]. Methods differ in the number and positions of points that they sample from the plenoptic function and thus in the degree of freedom they provide for synthesizing additional views. Moreover, they can be distinguished on whether they use additional geometry information, as e.g. depth maps or additional 3D models.

Methods not using geometry information, but a dense sampling of the plenoptic function are e.g. Lightfield Rendering [Levo 96] or Lumigraph [Gort 96]. Both methods record a large set of pictures depicting the scene from a dense set of neighboring views in the 3D space. This dense sampling allows synthesis of additional views located in-between recorded views by conventional interpolation filters. A direct interpolation is not possible if the 3D space is only sparsely sampled [Chai 00]. To overcome this and enable view synthesis with a small number of views, methods exploiting information about the 3D scene's geometry have been proposed, as e.g. DIBR [Fehn 04]. The knowledge of the scene geometry enables to identify corresponding points in pictures of different views and thus to map sample values from one view to another. This way, parts of the plenoptic function can be reconstructed by compensating the parallax between different captured views, as it will be discussed in the following.

2.1.1.1 Plenoptic Sampling Based on a Pinhole Camera Model

The 3D system captures the 3D scene with two cameras, as shown in Figure 2.1. For capturing, a camera projects the points of the 3D scene to its image plane. With a pinhole model and homogeneous coordinates, the projection can be modeled by a 3×4 projection matrix [Ivek 05]

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]. \tag{2.2}$$

$\mathbf{P}$ can be separated to an (extrinsic) camera parameter matrix $\left[\mathbf{R}|\mathbf{t}\right]$ and an (intrinsic) camera calibration matrix $\mathbf{K}$. The 3×4 extrinsic camera parameter matrix describes the mapping of 3D scene coordinates to camera coordinates. It is composed of a 3×3 rotation matrix $\mathbf{R}$ and an 1×3 translation vector $\mathbf{t}$. The 3×3 camera calibration matrix describes the projection from the camera coordinates to the coordinates of the captured image, it is conventionally given by

$$\mathbf{K} = \begin{bmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

f is the focal length normalized by the sampling distance in the focal plane; o_x and o_y are the offsets of the captured image in the focal plane.

The projection matrix maps a point $\mathbf{m}_s = (x_s \; y_s \; z_s \; 1)^T$ in the 3D scene space to its corresponding projection position $\mathbf{m} = (x \; y \; 1)^T$ in the camera's image plane, as follows with w_n being a normalization factor:

$$w_n\mathbf{m} = \mathbf{P}\mathbf{m}_s = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]\mathbf{m}_s \tag{2.4}$$

Using (2.4) the set of points in the 3D space that is projected to a particular camera image plane position $(x \; y)^T$ can be derived as:

$$w_n\mathbf{m} = \mathbf{P}\mathbf{m}_s = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]\mathbf{m}_s = \left[\mathbf{K}\mathbf{R}|\mathbf{K}\mathbf{t}\right]\begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} = \mathbf{K}\mathbf{R}\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} + \mathbf{K}\mathbf{t} \tag{2.5}$$

$$\Leftrightarrow \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = (\mathbf{K}\mathbf{R})^{-1}(w_n\mathbf{m} - \mathbf{K}\mathbf{t}) = w_n(\mathbf{K}\mathbf{R})^{-1}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \mathbf{c} \tag{2.6}$$

When w_n is interpreted as free parameter, (2.6) provides the projection line in the 3D scene space that intersects with $(x \; y)^T$ in the camera image plane and the camera projection center $\mathbf{c} = -(\mathbf{K}\mathbf{R})^{-1}(\mathbf{K}\mathbf{t})$ in the 3D scene space. To find points of the plenoptic function that are sampled by the camera, projection lines intersecting with the camera projection center $\mathbf{c} = (x_c \; y_c \; z_c)^T$ can be parametrized in spherical coordinates $(r_c \; \theta_c \; \varphi_c)^T$ centered around $\mathbf{c}$. In this representation, r_c is the free parameter and θ_c and φ_c are given for completeness in Appendix A.1. Assuming that a camera captures the light rays that pass the camera projection center $\mathbf{c}$, the parametrization can be used to map the plenoptic function (2.1) to the image s_C at

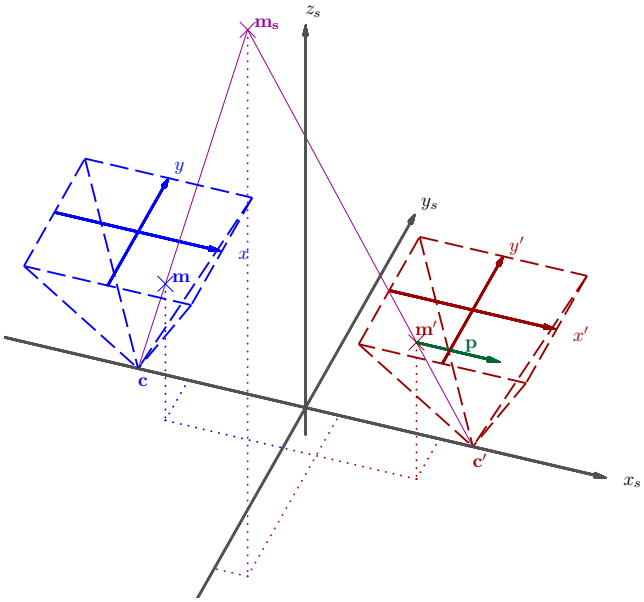


Figure 2.1: Point correspondence in a linear and parallel camera setup: Point $\mathbf{m}_s$ in the 3D scene space is projected to points $\mathbf{m}$ and $\mathbf{m}'$ in the cameras' focal planes. Consequently, the sample at $\mathbf{m}'$ in the right camera corresponds to the sample at $\mathbf{m} = \mathbf{m}' + \mathbf{p}$ in the left camera.

the camera plane:

$$s_C(x, y, \lambda, t) = \mathrm{f_P}(x_c, y_c, z_c, \theta_c, \varphi_c, \lambda, t) \tag{2.7}$$

From this image the camera captures a part of the received spectrum by integrating over a certain value range of λ. The resulting texture picture is in the following denoted as $s_T(x, y)$; t is dropped, since pictures that are discussed in the same context in this thesis are in general captured at the same point in time.

2.1.1.2 Point Correspondence

The last section assumed that the light rays passing through the camera projection center are captured in the focal plane of the camera. This is only true when there are no obstacles between the focal plane and the camera projection center attenuating or blocking the light rays. Under these conditions, geometrical optics define dependencies between the points of the plenoptic function. A similar dependency is given between points in the focal planes of two cameras. Consider a point in the 3D scene space that is captured by two identical cameras, as depicted in Figure 2.1. Under the condition that its luminance is isotropic (called Lambertian condition)

and that light rays emitted from this point are captured by two cameras without attenuation, its projection is identical in the focal planes of both cameras. This dependency can be used in two ways as the position of the point in the 3D scene and its position in the focal planes is linked by projective geometry. First, when two pictures are given, the position of the point in the 3D scene and thus its depth can be derived from its corresponding positions in the two camera planes. And second, when only one picture is given and its position in the 3D scene is known, its position in a second virtual camera can be derived. This is the main principle of DIBR.

Corresponding points in both cameras can be identified by the difference between the projection position $(x\ y)^T$ in the first camera and the projection position $(x'\ y')^T$ in the second camera, i.e. by their disparity. In the linear camera setup that is assumed in this thesis, cameras are only translated in x direction such that $\mathbf{t}$ is given by $(t_x\ 0\ 0)^T$. Considering furthermore that cameras are not rotated, i.e. that $\mathbf{R}$ is equal to the identity matrix, then (2.3) and (2.4) show that the position $\mathbf{m}$ in the first camera to that a 3D scene point $\mathbf{m}_s$ is projected is given by

$$w_n\mathbf{m} = \mathbf{P}\mathbf{m}_s = \begin{bmatrix} f & 0 & o_x \\ 0 & f & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} = \begin{bmatrix} f \cdot (t_x + x_s) + o_x \cdot z_s \\ f \cdot y_s + o_y \cdot z_s \\ z_s \end{bmatrix}. \tag{2.8}$$

Normalizing (2.8) by $w_n = z_s$ results in projection position

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \cdot (t_x + x_s)/z_s + o_x \\ f \cdot y_s/z_s + o_y \end{pmatrix}. \tag{2.9}$$

Assuming furthermore that the second camera has matrices $\mathbf{K}' = \mathbf{K}$ and $\mathbf{R}' = \mathbf{R}'$, and a translation vector $\mathbf{t}' = (t'_x\ 0\ 0)^T$ leads to a disparity of

$$\mathbf{p} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} f \cdot (t_x - t'_x)/z_s \\ 0 \end{pmatrix}. \tag{2.10}$$

(2.10) shows that in the linear and parallel setup corresponding points in different views are located at the same vertical position. Moreover, their horizontal position difference p_x is proportional to the focal length and the camera translation and inverse proportional to the depth z_s of the point in the 3D scene.

2.1.2 Basic and Advanced Rendering Techniques

A consequence of (2.10) is that, when the depth z_s of the point that is projected to $(x\ y)^T$ is given by a signal $s_Z(x, y)$, the sample in a captured texture picture $s_T(x, y)$ can be shifted by a distance given by the corresponding disparity signal

$$s_P(x, y) = f \cdot (t_x - t'_x)/s_Z(x, y). \tag{2.11}$$

to generate a sample of a picture $s'_T(x', y')$ of a virtual view. This is known as forward warping and is the key component of DIBR.

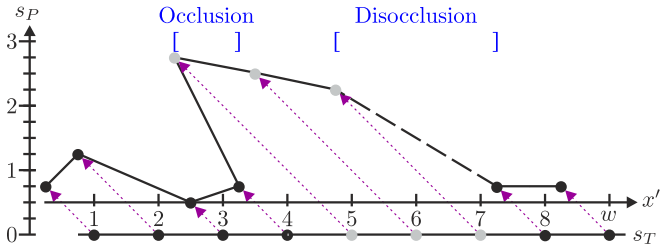


Figure 2.2: x'-s_P space: The IV samples s_T are shifted by the disparity s_P to SV positions x'.

In practice, DIBR conventionally does not use $s_Z(x)$[1] directly, but a non-linearly scaled version called depth map $s_M(x)$. Considering e.g. the representation format used by JCT-3V [MPEG 10a], $s_M(x)$ is given by

$$s_M(x) = 255 \cdot \frac{1/s_Z(x) - 1/z_f}{1/z_n - 1/z_f} \tag{2.12}$$

with z_n and z_f denoting the minimum and maximum of $s_Z(x)$. More specifically, (2.12) first inverts $s_Z(x)$ so that depth map values are proportional to the disparity. Then, it scales the inverted values to the range $[0, 255]$ to store them with 8-bit integer precision.

Given the IV depth map $s_M(x)$, DIBR can be implemented as follows: First, $s_P(x)$ is derived from $s_M(x)$ by scaling with a factor a and adding an offset b, i.e., corresponding to (2.11) and (2.12) by

$$s_P(x) = a \cdot s_M(x) + b \tag{2.13}$$

with $a = c \cdot (1/z_n - 1/z_f)$, $b = c/z_f$, and $c = f \cdot (t_x - t'_x)/255$. Then, a shifting function f uses $s_P(x)$ to identify the SV position x' that corresponds to the IV position x and the value of the IV sample at x is assigned to the SV position x':

$$x' = \mathrm{f}(x) = x - s_P(x) \tag{2.14}$$

$$s'_T(x') \coloneqq s_T(x) \tag{2.15}$$

This way, the IV texture s_T is warped such that its samples appear at their corresponding positions in the SV texture s'_T. Forward warping thus compensates parallax in s_T with respect to the SV camera position.

Figure 2.2 gives an example for forward warping in an x'-s_P space. It shows how IV texture samples $s_T(x)$ are shifted by their disparities $s_P(x)$ from IV positions x to SV positions x'. More specifically, a sample's vertical position indicates its disparity $s_P(x)$ used for shifting with (2.14). Thus, if a sample is located at the vertical position $s_P(x)$ equal to 0, its horizontal position in the x'-s_P space corresponds the IV position x. Otherwise, if $s_P(x)$ is not equal to 0, the sample is shifted and its horizontal position corresponds to its SV position x'. Beyond that,

[1] As p_y is always zero in the linear and parallel camera setup, the horizontal position y is dropped for clarity of description.

Problem	Caused by	Addressed by
Parallax	Varying depth of the 3D scene	Warping
Occlusions	Projection to same SV position	Detection of front-most point
Disocclusions	Scene points not visible in IV	Hole filling, view interpolation
Illumination-mismatch	Non-Lambertian reflectance	Illumination comp., blending
Superposition of points	Transparencies	Additional texture/depth layers
Non-uniform sampling	Perspective deformation	Sample interpolation, splatting
Texture/depth misalignment	Unreliable depth estimation	Boundary noise removal
Temporal inconsistent depth	Unreliable depth estimation	Temporal depth filtering

Table 2.1: Problems in DIBR and methods for their resolution.

the x'-s_P-space has two other properties: The first originates from (2.14): When the disparity $s_P(x)$ of a sample at IV position x changes, its position in the x'-s_P-space moves diagonally. The second is given by (2.11): The disparity s_P increases with decreasing depth s_Z. This means that when two shifted IV samples share the same SV position x', the upper sample in the x'-s_P-space is in the foreground. In summary, the x'-s_P-space visually links the IV samples' positions x, their warped SV positions x', and via s_P their depth in the 3D scene. It thus allows to understand the impact of depth changes easily so that it is a valuable tool which will be used in this thesis for discussion of warping related problems.

Some warping related problems are already shown in Figure 2.2: Since either multiple or none of the IV samples are shifted to particular regions or sampling positions in the SV (i.e. the occlusion and the disocclusion, respectively), the SV texture s'_T is neither completely nor uniquely defined by forward warping with (2.15). General reason for such problems is that the condition for correspondence (i.e. that a point of the 3D scene appears in texture pictures of both views equally) does not hold for all points. Why such problems occur and how DIBR-based approaches can address them by applying additional methods on top of forward warping is discussed in the next section.

2.1.2.1 Techniques Addressing DIBR Inherent Problems

As discussed in the last section, warping for parallax compensation is not sufficient for rendering an SV texture $s'_T(x)$ from a captured IV texture $s_T(x)$. Problems that need to be addressed additionally are summarized in Table 2.1. Some are inherent to the warping approach, others only occur when the provided geometry information, i.e. the depth map, is distorted. Usually conditions for point correspondence are not fulfilled for all points of a 3D scene. This causes disocclusions, occlusions, superpositions, and color mismatches. Furthermore, a re-sampling problem occurs as IV samples are not necessarily warped to the sampling grid of the SV. Whereas addressing the disocclusion, the occlusion, and the re-sampling problem is compulsory for DIBR methods to avoid severe artifacts, superpostion of projected points and color mismatches are usually ignored.

Occlusions
Occlusions occur when 3D scene points are projected to different IV points, but to the same SV point. An example is shown in Figure 2.3: The foreground point $\mathbf{i_s}$ and the background point $\mathbf{g_s}$ are projected to the IV points $\mathbf{i}$ and $\mathbf{g}$, respectively; in the SV, they are projected to the same point $\mathbf{i}' = \mathbf{g}'$. Thus, when warping samples from the IV to the SV, two IV samples correspond to the same SV position. As only the sample closer to the camera should be visible in the SV, the renderer must distinguish between foreground and background. As depth values of both warped samples are available in DIBR, a common solution is z-buffering. This means storing the warped samples depth values and discarding samples that are in the background.

Disocclusions
The disocclusion problem occurs when a point of the 3D scene is visible in the SV, but not in the IV. An example for this is given in Figure 2.3. There, the foreground object at point $\mathbf{j_s}$ prevents that light rays emitted from the point $\mathbf{h_s}$ reach $\mathbf{c}$. Consequently, the IV does not contain any point corresponding to $\mathbf{h_s}$. However, since $\mathbf{h_s}$ is projected to the SV position $\mathbf{h}'$, an IV sample value is required. This is the disocclusion problem.

A common solution is to capture the 3D scene with two cameras. This way, it is likely that 3D scene points that are not visible in one view are visible in another view. Consider e.g. in Figure 2.3, a second IV camera located to the right of the SV camera on the x_s axis. Such a camera would capture those background parts that are hidden from first IV camera by the foreground object and required for rendering the SV. Consequently, SV points that have no corresponding points in the first IV can be rendered from corresponding points in the second IV and vice versa. This approach is known as *view interpolation*.

View interpolation can usually fill a majority of disocclusions. However, in some scenarios parts of the 3D scene are not visible in both cameras. Consider e.g. a camera pair centered in front of a small and deep gap between two foreground objects such that they hide the gap's background. Then, IV samples for rendering a central SV would still be missing, and the disocclusion problem would not be solved by view interpolation. To handle such cases, view synthesis methods usually apply *inpainting* from SV samples adjacent to the disocclusion. Since holes that remain after view interpolation are in general small, simple inpainting algorithms are often sufficient to fill them. Sophisticated methods are conventionally only required when view interpolation is not applied, but only warping from one view (known as *view extrapolation*). A reason for view extrapolation can be that only one IV is available or that the SV camera is not located between the left and the right IV camera, but e.g. to the left of the left IV camera.

Illumination Mismatches
When the reflectance of a point $\mathbf{m_s}$ in the 3D scene does not meet the Lambertian condition, it reflects different intensities to the SV camera and the IV camera. Consequently, the sample values at the SV position $\mathbf{m}'$ and at the IV position $\mathbf{m}$ would differ. However, as DIBR uses the value of point $\mathbf{m}$ for point $\mathbf{m}'$ an illumination mismatch occurs.

This problem can actually be ignored for view extrapolation since it is generally not noticeable that the SV is shown with intensities received at the IV camera position. However, for view interpolation the problem occurs in a different manner: Consider e.g. a homogeneous

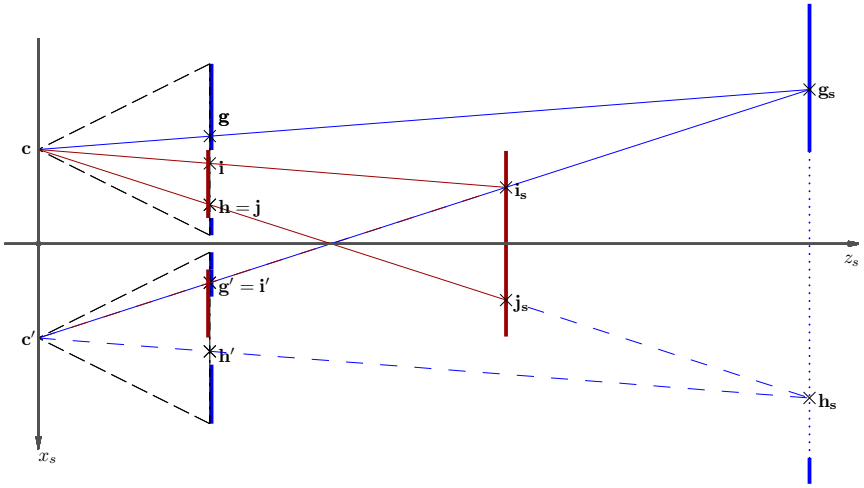


Figure 2.3: An IV picture captured at $\mathbf{c}$ is warped to generate an SV picture at $\mathbf{c}'$. Occlusion problem: The IV points $\mathbf{g}$ and $\mathbf{i}$ belonging to the background and the foreground object, respectively, correspond to the same SV point $\mathbf{g}' = \mathbf{i}'$. Disocclusion problem: The SV point $\mathbf{h}'$ has no corresponding IV point.

background of a 3D scene that reflects different intensities to the two IV cameras. Consider furthermore view interpolation that renders the SV in general from one IV and only fills holes from the other IV. Then different SV parts that should have the same intensity would have different intensities; distracting borderlines would occur.

Illumination compensation methods can mitigate such artifacts by local alignment of average intensities in corresponding regions of both IVs, e.g. as pre-processing step before rendering or encoding. However, they do not solve the illumination mismatch with respect to the SV as they do not reconstruct the intensities that would be captured by a real camera.

Superpositions

In the discussion of the occlusion and disocclusion problem, it was assumed that the foreground object is opaque. In case of a transparent foreground object, two additional problems arise. The first occurs when a projection line of an IV camera intersects with a transparent foreground point and a background point. Then intensities reflected from both points interfere in the corresponding IV point in the camera plane. The value captured at the IV point is thus related to both, the foreground and background point. Consequently, a correct depth value cannot be defined for the IV point. Furthermore, when the foreground and the background point are visible in an SV, a corresponding IV point exists for neither of them. The second problem occurs for the perspective of the SV when a projection line of the SV camera intersects with a transparent foreground point and a background point. Then the related SV point in the camera plane

should have a value determined by a superposition of intensities reflected from the foreground and background point. However, a correct value for the SV point cannot be rendered from the IV: First, because the projection of the transparent foreground point in the IV depends on the background point visible from the IV camera; and second, because the transparency of the foreground is not known.

Resolving such problems, which are in summary caused by superposition of intensities reflected by different points, requires a complex rendering process using additional transparency, depth, and texture layers. A preliminary is furthermore to generate such data, which actually introduces additional problems for natural 3D scenes. Consequently, most DIBR approaches ignore the superposition problem. Despite of this, they often perform well when a transparent foreground is diffuse or homogeneous, as it is e.g. the case for mist or fume. Then it is often sufficient to warp IV samples representing a superposition of foreground and background points with the depth values of the background points. In the SV, it is usually not noticeable that background points are superimposed by the wrong foreground points.

Re-sampling
In contrast to the other problems discussed so far, the sampling problem is not related to unfulfilled correspondence conditions. It occurs as IV objects appear perspectively displaced and deformed in the SV. Their warped IV sampling grid is thus irregular and does not match with the regular SV sampling grid. For this, re-sampling is required, which includes three problems.

The first arises since disparities s_P derived from the depth information s_Z are not necessarily quantized to values corresponding to the spatial sampling distance used in the SV and IV. IV samples warped with (2.15) can thus be located at fractional sampling positions in the SV. The simplest solution for this is rounding of warped IV sample positions to integer IV sample positions. However, as this is equivalent to increasing the quantization step size of the depth data, it reduces the number of depth levels that can be perceived. This can lead to artifacts, e.g. objects with a continuous depth variation can appear as if consisting of multiple stacked layers. For improvement, methods have been proposed that increase the IV and SV sampling rate by a factor N before warping and decimate the SV to the original size before output. This way, when rounding is performed in the upsampled domain, it corresponds to rounding to $1/N$-th positions. Beyond that, other methods apply e.g. linear or spline interpolation from warped IV samples adjacent to the SV sampling grid.

The second problem is that the distance between adjacent samples in the IV can increase when they are warped to the SV. The warped IV sampling grid can thus have locally a smaller density than the SV sampling grid and small holes occur in the SV, which need to be filled. To overcome this, several methods have been proposed. One of them is point splatting, i.e. the value of an IV sample is not only assigned to the sample at its corresponding SV position, but also to adjacent SV positions. Other methods fill holes in the SV using interpolation from SV neighboring positions after warping, e.g. with the same approach that is also applied to interpolate between fractional SV sample positions. Finally, methods for disocclusions filling can also be applied.

The third problem is the counterpart to the second: The distance between adjacent IV samples decreases when they are warped to the SV. The warped IV sampling grid can thus have a higher

density than the SV sampling grid. Consequently, more than one warped IV sample can correspond to the same SV position. Methods addressing this e.g. discard a second sample that is warped to the same position or compute the average of multiple samples warped to the same position. In doing so, conventionally the depth of warped samples is regarded so that occluding foreground samples overwrite the background.

2.1.2.2 Techniques for Rendering with Distorted Depth

Besides the problems that are inherent to the DIBR approach, further problems can occur when the depth data is distorted. Distortions can be introduced by depth estimation algorithms, by noisy depth sensors, or by encoding. Their impact on the SV is further discussed in Section 2.2.2.3. Typical issues that can be addressed in rendering are misalignment of edges in texture and depth, and temporal inconsistency of the depth data.

Methods to remove texture and depth misalignments are often applied while or after warping and called boundary noise removal. They usually remove samples that probably have an incorrect depth; subsequently, they fill the resulting gaps when processing holes caused by disocclusions or re-sampling. To reduce temporal inconsistencies, filtering methods for depth data have been proposed [Fu 10], which identify non-moving texture regions and apply temporal smoothing of the corresponding depth map regions.

Although such methods are part of some view synthesis approaches, they can in general also be regarded and performed as pre-processing steps prior to view synthesis. Applying such preprocessing before encoding at the sender-side of a 3D system has the advantage that receiver-side complexity is reduced. Furthermore, they can then be applied only for depth data that really need improvement. This is in particular of advantage because some depth correction methods can also introduce artifacts when applied on high quality depth data.

2.1.2.3 Other View Synthesis Methods

Beyond approaches discussed above, a great variety of methods have been proposed to address the DIBR inherent problems. They either increase rendering quality, e.g. by depth filtering [Mori 09], boundary noise removal [Zhao 11a], or advanced hole filling [Ndji 11]; or they target complexity reduction, as e.g. for hardware implementations [Horn 11], by switching between IVs [Tsun 09], or by input driven real-time implementations [Berr 06]. Some view synthesis techniques have been adopted by MPEG to a software package called *View Synthesis Reference Software (VSRS)* [Tian 09, MPEG 10b]. The VSRS comprises two main modes commonly called *general mode* and *1D mode*. Whereas the 1D mode only allows synthesis of views in a parallel and linear camera setup, as examined in this thesis, the general mode allows synthesis at arbitrary positions. As this software has been the basis for the LCR developed in this thesis and is used as reference for comparison, it is in detail described in Appendix A.2. In summary, prior art methods render whole SV textures, in contrast the LCR is developed in this thesis such that it can also be extended for re-rendering of only parts of the SV to determine the SVDC.

Step	Examples for introduction or reduction of distortions
Acquisition	
Camera capturing	Sensor noise, blur, aliasing, incorrect camera alignment
Texture pre-processing	Noise reduction, view rectification, colour alignment
Depth estimation	Misalignment at edges, wrong levels, temporal inconsistency
Depth enhancement	Smoothing of depth maps, alignment of edges
Coding and transmission	
Texture & depth coding	Video coding artifacts, e.g. blurring, ringing, blocking, flickering
Transmission	Data losses, jerking, delay
Presentation and perception	
View synthesis	Ampl. or suppres. of SVDs due to texture and depth distortion
Autosteroscopic display	Cross-talk, reduced resolution, insufficient view density
Visual perception	Amplification or suppression due to salience or binocular fusion

Table 2.2: Examples for distortions in the 3D video system.

2.2 Synthesized View Distortion Basics and Models

This section summarizes distortions occurring in the 3D video system (Section 2.2.1), elaborates on how the distortions affect the synthesized views (Section 2.2.2), and finally reviews existing models and methods for SVD derivation (Section 2.2.3).

2.2.1 Distortions in the 3D Video System

In the widest sense, the 3D video system's distortion can be defined as the (theoretically) visually perceivable difference between the actual 3D scene at the sender-side and the reconstructed 3D scene at the receiver-side. One source of this distortion are the DIBR limitations that have been discussed in Section 2.1.2.1: Distortions occur when DIBR inherent problems cannot be addressed perfectly because of lost 3D scene information. Beyond that the 3D video system's processing steps (as shown in Figure 1.1) can introduce other distortions, as summarized in Table 2.2 and discussed in the following[2].

Typical texture distortions introduced in the *capturing step* are e.g. noise or blurring that occur in the cameras' lens or sensor systems. Furthermore, when the cameras are not correctly calibrated, captured textures might not be properly aligned, which can be regarded as distortion, since the following processing steps require views from a parallel and linear setup. A *texture processing* step can reduce texture distortions, e.g. by denoising or rectification. The *depth estimation step*, which conventionally applies stereo matching, might not determine point correspondences correctly, which can cause misalignment of texture and depth edges, wrong disparity levels, or temporal inconsistent depth. Such distortions can be reduced in a *depth enhancement step*. The *texture and depth coding steps* introduce distortions by lossy coding, which appear as video coding artifacts in texture and depth. Examples are blurring, ringing,

[2] A similar but more extensive classification of artifacts can be found in [Boev 08].

blocking, or flickering [Zeng 14]. Furthermore, a lossy channel or bandwidth fluctuations in the *transmission step* can cause data packet losses and delays, which appear as artifacts or jerking. Such artifacts can be addressed by error concealment at the receiver-side. At the receiver-side, the distortions of the decoded IV texture and depth affect the quality of the SV textures rendered in the *view synthesis step*. As view synthesis warps the IV samples, IV texture distortions are directly visible in the SV textures. However, when warped IV textures are superimposed in view combination and their distortions are uncorrelated, the resulting SVD might be reduced. The impact of depth distortions on the SVD depends significantly on the texture characteristics. Depth distortions introduce little SVDs in homogeneous texture regions and significant SVDs when associated with texture regions carrying relevant information, as e.g. edges. When views are emitted by an *autostereoscopic display*, distortions are e.g. blurring caused by decimation required for fitting all SVs to the resolution of the display's panel, cross-talk between emitted views, and all kinds of distortions known from 2D displays. Considering the *visual perception* of the emitted views by an observer, the human visual system and psychovisual effects can amplify or attenuate distortions: Distortions in salient regions or the foreground might have a greater impact; or distortion in one view might be masked by another view, which is known as binoccular suppression [Stel 00].

In summary, the perceived distortion at the receiver-side of the 3D video system is caused by a great variety of distortions. Distortions introduced by mature techniques that are already employed for 2D video (i.e. texture capturing, coding, and transmission) are usually uncritical. In contrast, there is still demand and potential for distortion reduction in new techniques specifically required for 3D video (i.e. depth estimation, depth coding, and displaying).

2.2.2 Definition of the Synthesized View Distortion

As this thesis targets an optimization of depth coding by regarding the SVD occurring in view synthesis, this section discusses how the coded IV depth $\tilde{s}_M$ and the SVD D are related to each other, and how the SVD D is related to other distortions in the 3D video system; thus, how the SVD is defined.

2.2.2.1 Relationship to the Coded Depth

The SVD D occurs in the SV texture $\tilde{s}'_T$ rendered by a method $\mathcal{V}$ from a coded IV depth $\tilde{s}_M$ and a coded IV texture $\tilde{s}_T$. This can formally be expressed by the view synthesis function $\mathrm{f}_{\mathcal{V}}$ as

$$\tilde{s}'_T = \mathrm{f}_{\mathcal{V}}\left(\tilde{s}_T, \tilde{s}_M\right). \tag{2.16}$$

The distortion of a tested SV texture s'_T—the SVD D—with respect to a quality metric $\mathcal{Q}$ and a reference SV texture $s'_{T,Ref}$ (which can be chosen as it will discussed in Section 2.2.2.2) is then given by a distortion function $\mathrm{f}_{\mathcal{Q}}$ as

$$D = \mathrm{f}_{\mathcal{Q}}\left(\tilde{s}'_T, s'_{T,Ref}\right). \tag{2.17}$$

Combining (2.16) and (2.17) finally provides the joint view synthesis and distortion function $\mathrm{f}_{\mathcal{D}}$,

$$D = \mathrm{f}_{\mathcal{Q}}\left(\mathrm{f}_{\mathcal{V}}\left(\tilde{s}_T, \tilde{s}_M\right), s'_{T,Ref}\right) = \mathrm{f}_{\mathcal{D}}\left(\tilde{s}_T, \tilde{s}_M, s'_{T,Ref}\right), \tag{2.18}$$

which can be realized by a method $\mathcal{D}$ that derives the SVD D from the coded IV depth $\tilde{s}_M$. (2.18) shows that there is a certain degree of freedom to define the relationship between the SVD D and the distortion of the coded IV depth $\tilde{s}_M$. It not only depends on the quality metric $\mathcal{Q}$, but also on the used DIBR method $\mathcal{V}$ and the selected reference SV texture.

2.2.2.2 Distortions Represented by the Synthesized View Distortion

Distortions that may affect the SVD D are the distortions that are introduced by a) the acquisition stage to the IV depth s_M and IV texture s_T; b) video coding to the coded IV depth $\tilde{s}_M$ and texture $\tilde{s}_T$; c) channel errors; and d) the view synthesis method $\mathcal{V}$. As channel errors are outside the scope of this thesis, it is assumed that the coded IV depth $\tilde{s}_M$ and the coded IV texture $\tilde{s}_T$ correspond to the uncoded IV depth s_M and texture s_T signals plus the distortions introduced in the texture and depth coding steps. Whether and how the other distortions are represented by the SVD D depends on the choice of the reference SV texture $s'_{T,Ref}$:

1. When $s'_{T,Ref}$ is rendered from the uncoded IV texture s_T and the uncoded IV depth s_M, the SVD D represents only distortions introduced by texture and depth coding.
2. When $s'_{T,Ref}$ is rendered from the coded IV texture $\tilde{s}_T$ and the uncoded IV depth s_M, the SVD D represents only distortions introduced by depth coding while considering that a coded IV texture is used.
3. When $s'_{T,Ref}$ is rendered from the uncoded IV texture s_T and the coded IV depth $\tilde{s}_M$, the SVD D represents only distortions introduced by texture coding while considering that a coded IV depth is used.
4. When $s'_{T,Ref}$ is an original IV texture depicting the 3D scene from the same position as the SV, the SVD D represents not only the distortions related to the coding step, but also the distortions introduced by the acquisition stage, by the view synthesis method, and by DIBR inherent problems.

In summary, the choice of the reference SV texture $s'_{T,Ref}$ allows to determine which distortions are represented by the SVD D. This thesis will in general use option 1 in order to separately assess the SVD introduced by encoding at the receiver-side. However, encoder-side SVD derivation (in Section 8.2) uses also option 2 and further options in that the tested SV texture is rendered from uncoded IV texture data.

2.2.2.3 Perceived Synthesized View Distortion

After view synthesis, an autostreoscopic display emits the SV textures, which are then perceived by an observer. Distortions additionally introduced in the display step are outside the scope of this thesis. The distortion perceived by an observer is, however, of interest in this thesis.

Consequently, the SVD as defined by (2.18) should correlate with the perceived distortion, which requires a valid video quality metric $\mathcal{Q}$.

Similar to 2D video coding, most of the existing works related to 3D video use the PSNR (and thus the SSD). For example, the development of 3D-HEVC has been successfully based on quantifying the SVD using the PSNR [Mull 14]. However, it is known that the PSNR can underestimate or overestimate certain kinds of distortions. This can be in particular true for 3D video coding, in which other aspects, like the perception of a 3D impression and binocular suppression, play a role. Consequently, quality metrics for 3D video have been developed [Liu 15]. However, none of them has established itself by replacing the PSNR. This thesis goes beyond PSNR in Section 9.4 by additionally using the Structural Similarity (SSIM) index [Wang 04], representing the distortion of luminance, contrast, and structure; and a measure quantifying the distortion of edges, which is known as Pratt's Figure of Merit (FOM) [Prat 78].

To show how the SVD caused by depth coding appears to an observer, Figure 2.4 shows typical SV artifacts. Artifacts in Figure 2.4 (a) are commonly called boundary noise. They are caused by foreground samples that are adjacent to the background and that have erroneously the depth of the background. They can also occur when background samples adjacent to the foreground have the depth values of the foreground. For both types, occurrence and severity depend on the texture of the foreground object and the background, the extend of the misalignment, and the side of the foreground object where the misalignment occurs as e.g. SVDs in the background might be occluded by the foreground.

Figure 2.4 (b) shows blurring at object edges. It occurs since the left and right IV depth is distorted such that the left SV texture and the right SV texture do no longer match spatially. Consequently, blending them results in blurring. Such artifacts occur not only at depth edges, but also in homogenous depth regions when the depth DC is incorrectly represented. For small objects even double images can occur.

Beyond boundary noise, blurring, and double images, temporal inconsistent depth data can cause flickering in the SV textures. Such erroneously introduced motion is in particular annoying when the 3D scene is in general static. Furthermore, incorrect depth values can obviously lead to an incorrect depth impression. E.g. in objects with continuously varying depth, blocking artifacts in the depth can appear as distinct steps in the depth impression.

2.2.3 Synthesized View Distortion Derivation Methods

This thesis targets a measurement of the SVD by rendering and direct computation. An alternative approach is to estimate the SVD without full rendering. A great variety of such methods has been proposed. This section discusses some of them. For this, it classifies methods based on their objectives in four categories: *Local SVD estimation methods* target the quantification of the SVD caused by a local depth distortion, e.g. block-based for application in depth encoding or depth filtering. *Detection and threshold methods* only detect whether a depth distortion causes an SVD or provide depth distortion thresholds that guarantee that the introduced SVD is below a respective SVD threshold. *Complexity reduction methods* target the optimization of

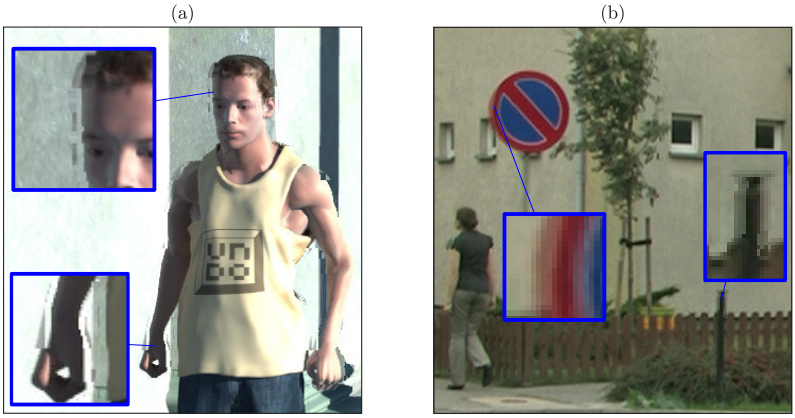


Figure 2.4: Typical SV artifacts caused by depth coding: (a) Boundary noise; (b) blurring.

existing other methods. Furthermore, *General SVD models* target to describe the relationship between the depth distortion and SV theoretically without targeting a local measurement. As the subject of this thesis is encoding, which requires a local measure, these methods are not further discussed; examples are [Rama 06, Zhan 15, Fang 14].

2.2.3.1 Local Estimation Methods

[Oh 11] presents an SVD estimation method, which is used as complement to SVDC computation by the HTM software. The method estimates the SVD as product of the magnitude of the local IV texture gradient and the disparity distortion. It thus takes into account that a disparity distortion affects the SV texture less in homogeneous IV texture regions than at IV texture edges. JCT-3V adopted this method—called VSD—in addition to the SVDC as metric for preselection of encoding modes [Oh 12c, Oh 12b, Oh 12d]. This way, the complexity of rendering can be reduced with a small reduction of RD performance [Oh 14]. Section 9.3 discusses this method further and compares it to SVDC computation.

Probably the first publication suggesting a local SVD estimation method is [Kim 09]. The method exploits the finding that the SVD is proportional to the disparity error. More specifically, it derives the factor between the SVD and the disparity distortion using a least squares fit and then estimates the SVD by multiplying the derived factor with the local disparity distortion. Also this method is further evaluated in Section 9.3.

[Kim 10] is a contribution of the same author. It includes two methods. The first method compares the IV texture to a texture obtained by shifting the IV texture samples by their respective disparity error. For complexity reduction, the second method avoids shifting, but uses an auto-regressive model. The first method is furthermore suggested in [Ma 13] and evaluated in Section 9.3. The second method is proposed in [Kim 15] as replacement for the VSD method.

[Silv 09] models the view synthesis process by shifting and heuristic assumptions on whether shifted samples are occluded. For initialization, the proposed method renders a reference SV texture. The method then derives the SVD related to a current IV sample as sum of two SVD terms by considering two respective positions in the reference SV texture. To obtain the first SV position, the method shifts the IV sample based on its coded IV depth value. Obviously, the SVD at this position, and thus the first SVD term, is the difference between the SV texture value at this position and the current IV sample's texture value. The method determines the second SV position by shifting the IV sample based on its uncoded IV depth value. Subsequently, the method distinguishes two cases: The first case occurs when the SV texture sample at this position is equal to the current IV texture sample. Then the method assumes that the current IV sample is not occluded in the reference SV texture, which implies that the depth distortion creates a hole. Therefore, it calculates the second SVD term as difference of the sample value used for hole filling and the current IV texture sample's value. When the SV texture sample at the second position is not equal to the current IV texture sample, the second case occurs: The method assumes that the current IV texture sample is occluded in the reference SV texture and that the depth distortion thus does not create an SVD at this position.

[Yang 15] derives an SVD model in the frequency domain. The derivation of the model first expresses the SVD as function of the depth distortion and the IV texture's Fourier transform. Then, it assumes a zero-mean Laplace distribution of the depth error. Finally, it combines both to describe the SVD as function of the energy spectral density of the local SV texture and the local expectation of the depth distortion. It then applies the model in depth coding, which includes a local adaptation of the Lagrange multiplier and the QP.

[Li 14] proposes a method that is based on the LCR and thus on one of the main contributions of this thesis. The method exploits the interval processing of the LCR (see Section 3.2.3). An initial step renders the SV texture and records the different SV interval types. Then for SVD derivation, it does not re-render the SV intervals, but estimates how their SVD changes. This means that it 1) assumes that no distortion is introduced in entirely occluded intervals, 2) estimates the SVD change in continuous intervals linearly, and 3) estimates the SVD change in right edge intervals considering interval length changes and filling by the background sample.

[Zhan 16] extends the SVDC computation and VSD estimation in the HTM software by a distortion term that targets to quantify subjectively perceivable temporal distortions. The extension derives the temporal distortion term by comparing two differences that are 1) the difference of two consecutive SV texture pictures rendered from uncoded IV data and 2) the difference of two consecutive SV texture pictures rendered from coded IV data. Furthermore, it includes an adaption of the Lagrange multiplier.

[Zhan 14] models the local depth distortion while 1) considering that multiple depth values are mapped to the same disparity value because of quantization differences, and 2) assuming a linear relationship between the disparity distortion and the SVD. The paper furthermore applies the model for mode selection, derives a suitable Lagrange multiplier, and suggests how the QP can be chosen based on the model. Furthermore, [Zhan 17] is based on the same method, but additionally suggests to use the method for early termination in mode selection.

2.2.3.2 Detection and Threshold Methods

[Zhao 11b] proposes a method to detect whether a depth change introduces an SVD change. The method exploits that the depth is generally finer quantized as the disparities required by the view synthesis method. It derives the range in that depth values can change without introducing a disparity distortion and thus an SVD. The determined range is then used to smooth the depth maps before encoding.

[Shao 14] proposes a similar method. In contrast to [Zhao 11b], the method allows a certain SVD level, which is predefined by an SVD threshold. For initialization, the method performs two steps. The first step shifts IV texture samples locally to find the maximal disparity distortion that causes an SVD below the predefined threshold. Based on the maximal disparity distortion, it then derives the related maximal tolerable depth distortion. The second step estimates a mask that indicates whether samples are occluded in the SV. The derived maximal disparity distortion and the mask are then used by an encoder to decide on whether to signal the residual and on how to quantize it.

[Lee 16] proposes a similar method, which estimates the local complexity of the IV texture by considering the magnitude of its local IV texture gradient. An encoder then skips residual coding when the estimate is below a given just notable difference (JND) threshold.

[Tech 10] is an own contribution for depth map filtering prior to encoding. It determines iteratively by rendering whether filtering of single depth samples with a diffusion approach changes the SV textures. When this is the case, the respective samples are not filtered.

2.2.3.3 Complexity Reduction Methods

Since the RM is part of the HTM software, several modifications have meanwhile been proposed by others, which target complexity reductions. More specifically, [Ma 14, Dou 15] propose to skip re-rendering for depth distortions that likely cause no SVD or a small SVD. The former can e.g. occur in occluded SV regions, or when the depth is quantized with a finer granularity than used by the view synthesis method such that a depth change causes no disparity change. The latter can be the case in smooth texture regions. From these methods JCT-3V adopted skipping re-rendering with the RM when the depth change causes no disparity change [Ma 14, Wang 12d, Oh 12a]. This method is called *VSO early skip* and is part of the HTM encoder.

2.3 Chapter Summary

The 3D scene can be modeled by the plenoptic function and consequently capturing of views by sampling the plenoptic function with a pinhole camera. Based on the pinhole camera model, point correspondences in different pictures and the 3D scene can be derived when the depth of the scene is known and when the condition for correspondence, i.e. points of the 3D scene are equally visible in the different pictures, is fulfilled.

DIBR uses these correspondences for parallax compensation to synthesize additional views of the 3D scene. When correspondence conditions are not fulfilled, DIBR inherent problems occur, i.e. occlusion, disocclussion, superposition, and illumination mismatches. These problems are typically resolved e.g. by occlusion detection, view interpolation, hole filling, splatting, sample interpolation, or illumination compensation. Additional problems are caused by incorrect depth data; they can be mitigated in DIBR by boundary noise removal or temporal filtering.

For DIBR, a great variety of methods has been proposed. From these, the VSRS provided by MPEG has been the basis for the development of the LCR. Existing methods differ from the LCR since they are not designed for re-rendering, as required for encoder-side distortion derivation.

Distortions in the 3D video system are inherent to the DIBR approach or introduced, reduced, or amplified in the system's processing steps. From these, this thesis focuses on the distortion of the SV texture—the SVD. Depending on the reference that is used for its computation, the SVD can jointly or solely represent distortions introduced in the acquisition stage, in depth coding, or in texture coding. Distortions introduced by the channel and by the autosteroscopic display are neglected in this thesis.

To quantify the SVD, this thesis uses in general the PSNR, but also two alternative metrics, the Structural Similarity index and Pratt's Figure of Merit, which quantify the perceptual quality and the alignment of edges, respectively. Perceivable SV artifacts due to depth coding are typically boundary noise, which is caused by texture and depth misalignment at edges, and blurring, which occurs when SV textures used in view combination do not match.

Existing methods for SVD derivation can be categorized in: Local SVD estimation methods, detection and threshold methods, complexity reduction methods, and general SVD models. SVDC computation with the RM—a main contribution of this thesis—differs from these other works because it allows an exact quantification of the change of the SVD using re-rendering, as it will be discussed in Chapters 4 and 5. However, since SVDC computation with the RM is part of the HTM software, it has actually become the basis for some of the existing methods.

3 Low-Complexity Depth Image Based Rendering Algorithm

Based on the DIBR discussion in the last chapter, this chapter presents one of the main contributions of this thesis—the low-complexity rendering algorithm (LCR). In particular, Section 3.1 discusses why such an LCR is needed and specifies requirements for its design. Section 3.2 then provides an overview of the LCR and elaborates on the implementation of its building blocks. Finally, Sections 3.3 and 3.4 evaluate the LCR's complexity and rendering quality, respectively.

Related works: The major part of this chapter (i.e. text and figures) has been presented in the own contribution [Tech 18]. Some aspects of the LCR also appear in the overview papers [Schw 11a, Schw 11b, Schw 11c, Schw 12b, Schw 12a, Mull 13], which describe the LCR as part of the RM in encoding. Other works related to the individual aspects of the LCR are discussed in Section 3.1.

3.1 Motivation and Requirements

As discussed in Section 2.1, view synthesis methods need to address several DIBR inherent problems. When doing so at the receiver-side of a 3D system, they must provide a sufficient SV texture quality, usually while not exceeding a certain computational complexity level. In principle, both is also required when view synthesis is applied according to the main idea of this thesis—by a depth encoder to select encoding modes based on their computed SVD.

In encoding, the computational complexity that can additionally be introduced for view synthesis is limited. Even without view synthesis, e.g. when mode selection is based on directly applied simple metrics, as e.g. the sum of squared differences (SSD) or absolute differences (SAD), distortion computation generates a significant part of the encoding complexity. Consequently, when an encoder additionally renders with a complex algorithm to calculate the distortion of the SV texture, its complexity can drastically increase. An important requirement for a view synthesis algorithm suitable for distortion computation in encoding is thus *low-complexity*.

Low-complexity view synthesis algorithms are usually optimized to render complete SV textures from complete IV textures and IV depth maps. In contrast to this, it is sufficient for SVD computation to render only parts of the SV that are related to the depth block processed and evaluated by the encoder. Actually, such *partial rendering* is not only sufficient, but also required since rendering of a whole SV texture for distortion computation of a single block is certainly too computational complex.

In summary, SVD computation in encoding requires a low-complexity algorithm that supports partial rendering and provides a sufficient rendering quality. The VSRS (as described in Appendix A.2) does not meet all three requirements. First, it comprises methods requiring a significant amount of memory bandwidth and computational operations, e.g.:

- The 1D and general mode buffer and access several intermediate results (i.e. results of a processing step that are required later).

- The 1D mode upsamples IV and SV signals for fractional sample accurate warping, which significantly increases the number of processed samples.
- In 1D and general mode, warping accesses hole and depth maps to evaluate if warped positions are already occupied by a foreground sample.
- The 1D mode evaluates for warping several SV positions adjacent to the shifted SV position to determine background samples.
- The general mode applies two warping steps (forward and backward).
- The general mode fills gaps in the SV depth applying a 3×3 median filter multiple times.
- In 1D mode, hole filling searches for the background side of holes.
- The general mode uses a complex inpainting algorithm for hole filling.
- In 1D mode, view combination detects holes in 7×7 windows.

Besides these methods with inherent high complexity, the VSRS comprises methods that are implemented in a non-optimized way. Therefore, rendering with the VSRS is not as low complex as required. Furthermore, it does not support partial rendering. Modifications to enable partial rendering on top of the VSRS methods would be hard to implement. As it becomes clear from the description in Appendix A.2, this would require to perform all steps partially. Sophisticated and computational complex methods would be needed to track or detect changes in the buffered intermediate results and to update them partially. Consequently, the VSRS is not suitable as renderer or as direct basis for a renderer in encoding.

However, as the VSRS renders with sufficient quality, its basic principles are the starting point for the development of the low complexity rendering algorithm (LCR), which is one of the main contributions of this thesis. For its development, the high complexity methods listed before are either replaced by simpler ones or optimized. Design criteria is complexity minimization while preserving a sufficient rendering quality for distorted depth and high quality for undistorted depth. Furthermore, it is taken care that the used methods can be easily modified to enable partial rendering.

More specifically, to develop the LCR, the following basic rendering techniques have been selected, modified, and combined:

- Techniques commonly used (e.g. [Tsun 09, Tian 09]) like warping, hole detection, and hole and margin filling.
- Occlusion detection from [Berr 06], but extended by detection of foreground (FG) edges.
- Correct rendering of edges [Tech 11] with an effect similar to boundary aware splatting [Tian 09].
- View blending similar to [Tian 09], but modified to operate after hole filling.
- A texture mapping approach, inspired by backward warping in [Tian 09], but embedded in an interval-wise processing scheme.

As these features are implemented in a modular way, the trade-off between rendering complexity and quality can be adjusted. How they address the DIBR inherent problems specifically is discussed in the next section.

3.2 Rendering Algorithm

As low-complexity is a major design aspect, this section discusses how the LCR can be implemented in an optimized way. The discussion allows to understand the required complexity down to single operations and provides thus a basis for the complexity analysis in Section 3.3. For clarity of description, it addresses design aspects for an extension by partial rendering only briefly; Chapter 5 elaborates on partial rendering separately. The focus in this section is on rendering of an entire SV texture by processing texture $s_{T,l}$ and depth $s_{M,l}$ of an entire left IV. Rendering from right IVs is performed accordingly, but with reversed processing direction.

The LCR exploits that disparities only have horizontal components due to the parallel camera setup and introduces no vertical dependencies. This means that it processes the views row-wise without referring to other rows. An overview how one row of the SV is rendered by processing one row of the IV is given in Figure 3.1. Central element of Figure 3.1 is an x'-$s_{P,l}$ chart, which shows how IV positions x are mapped in the first processing step to SV positions x'. Based on these positions further processing steps are applied in the SV to render the SV texture s'_T.

Usually, view synthesis methods (e.g. the VSRS) apply a single processing step to the entire IV or SV at once before continuing with the next step. As discussed in the last section, this has the drawback that results of the processing step that are required by following processing steps (i.e. intermediate results) need to be buffered and accessed later. To reduce buffering the LCR performs *interval-wise processing* while iterating over the IV samples. This means that in each iteration, it processes a pair of adjacent IV samples and renders all SV samples related to this pair at once by invoking all steps shown in Figure 3.1. Consequently, each iteration is associated with an IV interval defined by the two IV samples (e.g. b in Figure 3.1), and an SV interval that is affected by processing them (e.g. b'). This interval-wise rendering approach has two advantages: First, the number of signals that need to be stored and accessed is reduced; second, partial rendering can be performed by iterating once over a particular region in the IV.

More specifically, the LCR performs the following steps consecutively for each IV interval:

- *Warping*: IV positions x are shifted by disparities $s_{P,l}$ to SV positions x' to map the IV interval to its corresponding SV interval. (Section 3.2.1)
- *Render mode selection:* Depending on the SV interval boundaries, different scenarios can occur, as e.g. samples in the SV interval can be occluded (f'), disoccluded (b'), or belong to foreground object edges (d'). To adapt to these scenarios, the LCR has different modes. One of them is selected for the SV interval. (Section 3.2.2)
- *SV interval rendering:* The selected mode then derives sample values at all integer SV positions between the SV interval boundaries in an SV texture $s'_{T,l}$ (called left SV texture, as it is rendered from the left IV) from the corresponding IV positions in $s_{T,l}$. (Section 3.2.3)
- *View combination:* Some sample values rendered for an SV interval can be unreliable. Similar to the VSRS, such samples are enhanced by combining the left SV texture with a right SV texture $s'_{T,r}$, which has been rendered before (from the texture $s_{T,r}$ and the depth $s_{M,r}$ of a right IV) and is buffered. The result is the combined SV texture s'_T. (Section 3.2.4)

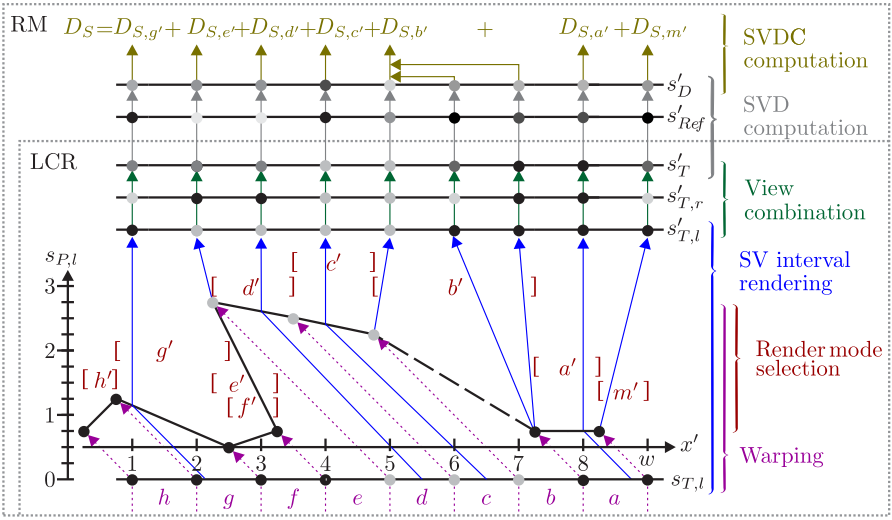


Figure 3.1: Bottom: The LCR's processing steps. Depicted signals represent one row of input, intermediate, or output data. Arrows show the relationship between samples or their positions. Top: Extension of the LCR to the RM (discussed in Chapter 5). ©IEEE

This way, the LCR can render the combined SV texture s'_T with low complexity and sufficient quality so that a basis for partial rendering and SVDC computation with the RM (as shown by the two topmost steps in Figure 3.1 and discussed in Chapter 5) is established.

3.2.1 Interval-wise Processing and Warping Step

As discussed above, the LCR operates interval-wise to facilitate partial rendering. When processing two adjacent samples of an IV interval, all SV samples of a related SV interval are rendered at once. Thus, an initial question is how to derive the SV interval with low complexity while iterating over the IV.

The LCR enables this as shown on the right of Figure 3.2. It iterates over the IV from right to left (so that occlusions can be detected with a simple method, as described later in Section 3.2.2.1). To this end, it starts with the current IV position x_s equal the row width w. After the LCR has initialized rendering of the current row (P1 in Figure 3.2) and after further iterations, the LCR stores x_s as the last processed IV positions x_e before decrementing x_s for the next iteration (P2). This way, an IV interval $[x_s, x_e]$ is provided in each iteration.

To render SV samples related to $[x_s, x_e]$ the LCR derives two SV positions (P3). The first, x'_s, is the SV position related to the current IV position x_s and thus derived using $x'_s = \mathrm{f}(x_s)$ with (2.14). This requires to convert the depth $s_{M,l}$ to disparity $s_{P,l}$ with (2.13), which is performed by a table look-up to reduce complexity. Table entries have a quantization step size of 0.25, so

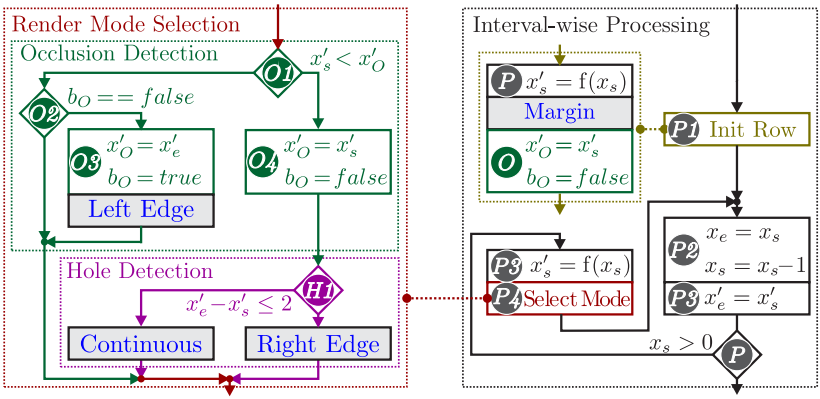


Figure 3.2: Right: Starting with $x_s = w$, interval-wise processing iteratively determines the IV interval $[x_s, x_e]$ and the SV interval $[x'_s, x'_e]$ (P2 and P3) and invokes the render mode selection step (P4) for each SV interval. Left: The render mode selection step performs occlusion detection (O1-O4) and hole detection (H1) and invokes the selected SV render mode. ©IEEE

that x'_s has quarter sample precision. The second, x'_e, is the SV position related to the last IV position x_e and is, as such, set equal to x'_s of the last iteration. Both positions provide the SV interval $[x'_s, x'_e]$.

Starting with the render mode selection (P4), the LCR then applies steps described in Sections 3.2.2 to 3.2.4 consecutively in the current iteration before continuing with the next. This way, the LCR can start and stop easily at arbitrary IV positions, as required for partial rendering.

3.2.2 Render Mode Selection Step

After mapping of IV positions to the SV, different DIBR inherent problems occur as discussed in Section 2.1.2.1. Specifically, when rendering an SV from a left IV, IV samples are shifted leftwards with (2.14). As foreground objects are shifted further leftwards than the background, they occlude the background on their left side and holes (i.e. disocclusions) occur on their right side. Moreover, no IV sample usually remains on the right SV margin and shifted IV sample position do not necessarily coincide with integer SV sampling grid. To adapt to these scenarios when rendering an SV interval, the LCR distinguishes between five different SV interval types, depending on the SV interval length and on whether its left boundary is occluded. In particular, it classifies the current SV interval's type as one of the following, which are exemplarily shown in Figures 3.1 and 3.3:

- A *Left edge interval* occurs on the left of a foreground object (e.g. e' in Figure 3.1). x'_e is the leftmost foreground object's position and is not occluded. Other positions in the left edge interval are occluded by the foreground object.

- An *entirely occluded interval* can occur on the left of a foreground object adjacent to a left edge interval (e.g. f') or an other entirely occluded interval. In contrast to left edge intervals (and other types), x'_s and x'_e are both occluded by the foreground object.
- A *right edge interval* occurs on the right of a foreground object (e.g. b'). x'_s is the rightmost foreground object position and is not occluded. x'_e belongs to the background. SV positions between x'_s and x'_e are disoccluded. The right side of a right edge interval may additionally be occluded by another foreground object on the right of x_e.
- In a *continuous interval* all positions in $[x'_s, x'_e]$ belong to the same object (e.g. a', c', d', g'). x'_s is not occluded. The right side of a continuous interval may additionally be occluded by another foreground object on the right of x_e (e.g. g').
- A *margin interval* occurs as gap on the right in the SV (e.g. m') since all IV samples are shifted leftwards. x'_s is given by shifting the rightmost IV position and thus by $\mathrm{f}(w)$. x'_e is the rightmost SV position w.

As margin intervals can only occur at the right SV margin, they are directly rendered as described in Section 3.2.3.5 when the LCR starts processing an IV row at $x_s = w$. In subsequent iterations, the question is how the LCR can differentiate between the other SV interval types to select the corresponding render mode. How this is done by *occlusion detection* and *hole detection* is described in the following. How the LCR then renders the samples related to the SV interval by adapting to its type is discussed in Section 3.2.3.

3.2.2.1 Occlusion Detection

The VSRS detects occlusions with a z-buffer method. For this, it evaluates additional intermediate SV signals, i.e. hole maps and depth maps, while warping to determine if SV positions at or adjacent to shifted IV positions are already occupied by a sample further in the foreground. As discussed in Section 3.1, this introduces additional reading and writing memory operations. To avoid them, the LCR detects whether a current SV interval or parts of it are occluded with an alternative occlusion detection method. The method has two objectives and uses two auxiliary variables, which are introduced in the following before giving an example.

The first objective is to detect left edge intervals and entirely occluded intervals. Both differ from other types in that the current SV position x'_s is occluded. To detect this, the LCR utilizes an auxiliary variable x'_O, which is called minimal occluding position. While processing the IV from right to left, the LCR sets x'_O to the leftmost fractional SV position that has already been rendered [Berr 06, Berr 03] in previous iterations. When x'_s is greater than or equal to x'_O, the position x'_s is occluded (see [Berr 03] or Appendix B.1) and the current SV interval is thus a left edge or an entirely occluded interval.

The second objective is to distinguish between left edge and entirely occluded intervals. They differ in that the last SV position x'_e is also occluded for entirely occluded intervals. To detect this, this thesis extends the method from [Berr 06] by a second auxiliary variable b_O, which is called occlusion flag and indicates whether x'_s of the SV interval rendered in the previous iteration (and thus x'_e in the current) is occluded.

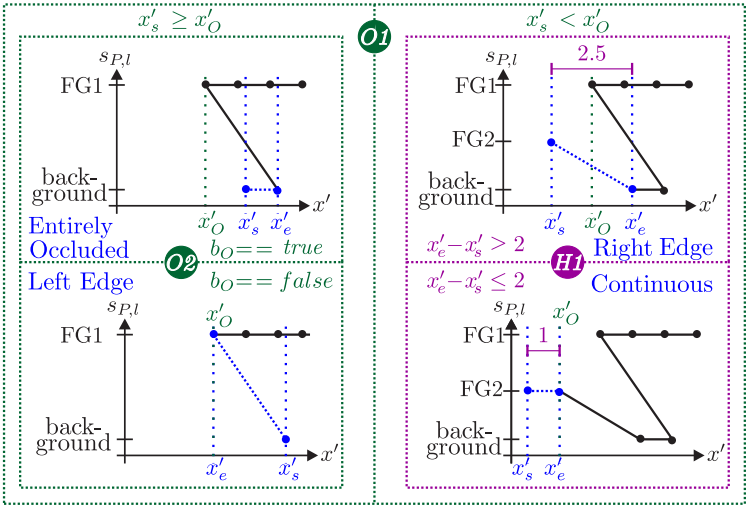


Figure 3.3: Examples for interval types, and occlusion and hole detection, as shown in Figure 3.2. Samples at FG1 and FG2 belong to two different foreground objects. ©IEEE

More specifically, the LCR addresses both objectives by evaluating x'_O and b_O in conditions (O1) and (O2) in Figures 3.2 and 3.3. The result determines how the LCR continues as shown in Figure 3.3:

- $x'_s \geq x'_O$ and b_O = false → x'_s occluded and x'_e not occluded → left edge interval. The LCR
  - sets x'_O equal to x'_e as it is now the leftmost SV position occluding other positions (O3);
  - sets the flag b_O equal to true to indicate that x'_s is occluded (O3); and
  - invokes the render mode for left edge intervals (Section 3.2.3.3).
- $x'_s \geq x'_O$ and $b_O = true$ → x'_s and x'_e occluded → entirely occluded interval.
  - Rendering or update of auxiliary variables is not required.
- $x'_s < x'_O$ → x'_s not occluded → right edge or continuous interval. The LCR
  - sets x'_O equal to x'_s since samples shifted further right are occluded (O4);
  - sets b_O equal to false to indicate that x'_s is not occluded (O4); and
  - determines a render mode for non-occluded parts of the SV interval by *hole detection* (Section 3.2.2.2).

This way, the LCR can detect occlusions from the boundary positions of an SV interval and the auxiliary variables only. Complex z-buffering methods are not required.

3.2.2.2 Hole Detection

As discussed in Section 2.1.2.1, IV samples can be missing at an SV position for two reasons: The first is the disocclusion problem, i.e. the corresponding sample is not visible in the IV; the second is the re-sampling problem, i.e. the corresponding position in the IV is not located on the integer IV sampling grid. The VSRS does not explicitly differentiate between these two reasons. It implicitly fills small holes, which usually occur because of the re-sampling problem, by median filtering or splatting in the warping step, larger holes due to disocclusions later in the view combination step, and finally remaining holes in a separate step after view combination. This final step requires additional iterations over the hole map to identify hole positions, and (in 1D mode) the detection of the background side that is adjacent to a hole.

In contrast, the LCR avoids the final hole filling step by addressing both, the disocclusion and the re-sampling problem, directly after warping. For this, hole detection distinguishes which of the two problems is present for the current SV interval. More specifically, the LCR evaluates the SV interval length $x'_e - x'_s$ (H1 in Figures 3.2 and 3.3), which is according to (2.12) to (2.14) proportional to the depth difference at x'_s and x'_e.

When the interval length is large (i.e. $x'_e - x'_s > 2$), and thus also the depth difference, x'_s belongs presumably to a foreground object and x'_e to a background object. A right edge interval is thus given and the hole in between its boundaries is caused by a disocclusion. Otherwise, in case of a small interval length (i.e. $x'_e - x'_s \leq 2$), x'_s and x'_e, presumably belong to the same object. A continuous interval is given and samples at integer SV position between its boundaries are presumably missing because of the re-sampling problem.

This way, the LCR can adapt to the disocclusion and the re-sampling problem by invoking different render modes for continuous and right edge intervals, as discussed in Sections 3.2.3.2 and 3.2.3.4, respectively. As discussed there, both modes derive sample values for all integer positions in the SV interval and provide those values to the view combination step. Since sample values are available for each integer SV position in the view combination step, a subsequent hole filling step is not necessary.

3.2.3 Synthesized View Interval Rendering Step

The warping and render mode selection steps have identified the SV interval's boundaries, x'_s and x'_e, and its type. The LCR can now adapt to this type to derive the SV texture in the SV interval. How this is done by interval type specific render modes is described in this section.

3.2.3.1 Occluded SV Interval Parts

By applying a render mode, the LCR renders $s'_{T,l}(\tilde{x}')$ for all integer positions $\tilde{x}'$ of the SV interval that are not occluded. Entirely occluded intervals are already skipped in the render mode selection step; for left edge intervals, it is clear that only their leftmost position is not occluded. However, right edge and continuous intervals can be occluded on their right side. In

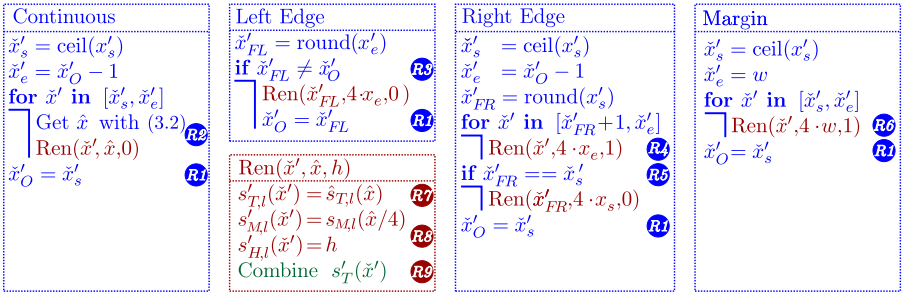


Figure 3.4: Render Modes for an SV Interval. Ren$(\check{x}', \hat{x}, h)$ maps a sample from the upsampled IV texture to the current SV position $\check{x}'$ (R7). The IV depth is mapped similarly, but for simplification only with integer precision (R8).

this case, the occluding SV positions have already been rendered in previous iterations as the LCR processes the IV from right to left (see Appendix B.1). They should not be overwritten when rendering the current SV interval. For this, the LCR uses an auxiliary variable $\check{x}'_O$ that tracks the leftmost integer SV position $\check{x}'$ rendered in iterations before [(R1) in Figure 3.4] and thus corresponds to x'_O, but in integer precision. The LCR must only render integer SV positions $\check{x}'$ of the current SV interval that are on the left of $\check{x}'_O$ [Berr 03].

3.2.3.2 Render Mode for Continuous Intervals

When the LCR selects the render mode for continuous intervals, the SV interval boundaries $[x'_s, x'_e]$ belong to the same object. The LCR must now address the re-sampling problem to derive the non-occluded samples in $[x'_s, x'_e]$. They are, as $\check{x}'_O$ is the leftmost occluding integer SV position that is already rendered, at integer SV positions $\check{x}'$ in

$$\left[\check{x}'_s, \check{x}'_e\right] = \left[\text{ceil}(x'_s), \check{x}'_O - 1\right]. \tag{3.1}$$

As discussed in Appendix A.2, the VSRS derives sample values at integer SV positions that do not match with the fractional positions of the warped IV samples by several approaches: E.g. warping an upsampled IV texture, rounding of warped sample positions, and splatting; or forward warping of the IV depth, median filtering, and backward warping. As these methods introduce additional complexity, the LCR avoids them by applying interval-wise texture mapping.

This means that the LCR interpolates the IV texture and maps positions in the SV interval to the interpolated samples in the IV. More specifically, on initialization the LCR quadruples the horizontal sampling rate of the IV texture $s_{T,l}$ to derive an upsampled texture $\hat{s}_{T,l}$ using the interpolation filter given in Table B.1. Then while rendering, the LCR derives the value of a sample in $[\check{x}'_s, \check{x}'_e]$ by mapping its SV position $\check{x}'$ to a position $\hat{x}$ in the corresponding IV interval

in $\hat{s}_{T,l}$ (R2) and setting $s'_{T,l}(\check{x}')$ to $\hat{s}_{T,l}(\hat{x})$ (R7).[1] The LCR maps positions with (3.2) so that the relative position of $\hat{x}$ in the upsampled IV interval in $\hat{s}_{T,l}$ is equal to the relative position of $\check{x}'$ in the SV interval.

$$\hat{x} = 4 \cdot \left(\frac{\check{x}' - x'_s}{x'_e - x'_s} + x_s \right) \tag{3.2}$$

As the LCR warps with quarter sample precision and applies the render mode for continuous intervals only when the distance between x'_e and x'_s is not greater than 2, the division in (3.2) can be avoided by using a look-up table (Table B.2).

In conclusion, this mapping has the advantage that the LCR can warp with quarter sample precision without applying complex methods as the VSRS. The computational complex interpolation is only required once for initializing the LCR and not during re-rendering because it is independent from the depth data.

3.2.3.3 Render Mode for Left Edge Intervals

When the LCR selects this mode, the leftmost foreground object position is the left SV boundary x'_e. This position is, in contrast to the remaining position of the left edge interval, not occluded. Consequently, the LCR must render the related SV integer position $\check{x}'_{FL} = \mathrm{round}(x'_e)$ correctly. The leftmost integer position already rendered is $\check{x}'_O$. Therefore, if $\check{x}'_{FL}$ is equal to $\check{x}'_O$, the LCR has already rendered the SV sample at $\check{x}'_{FL}$ in the last iteration. However, when $\check{x}'_{FL}$ is not equal to $\check{x}'_O$ (R3), the LCR sets the SV sample at $\check{x}'_{FL}$ to $s_{T,l}(x_e)$ [i.e. $\hat{s}_{T,l}(4{\cdot}x_e)$].

In summary, the fractional left foreground edge position is correctly mapped to the integer SV sampling grid. The effect is similar to boundary aware splatting as performed by the VSRS. Small but obtrusive visual artifacts at edges are avoided.

3.2.3.4 Render Mode for Right Edge Intervals

In this mode, x'_s is the rightmost SV position of a foreground object, whereas x'_e belongs to a background object. To derive SV samples within the integer SV boundaries given by (3.1), the LCR must fill the disocclusion and render the right foreground edge correctly.

The rightmost integer foreground object position is given by $\check{x}'_{FR} = \mathrm{round}(x'_s)$. Therefore, the disocclusion is in the SV interval $[\check{x}'_{FR}{+}1, \check{x}'_e]$. To fill it, the LCR extrapolates SV samples from the background sample $s_{T,l}(x_e)$ at the right (R4). Then, the LCR examines whether the left integer SV interval boundary $\check{x}'_s$ is equal to the rightmost integer foreground object position $\check{x}'_{FR}$ (R5). If this is true, $\check{x}'_{FR}$ belongs to the current SV interval and the LCR sets the SV sample at $\check{x}'_{FR}$ to $s_{T,l}(x_s)$. Otherwise, $\check{x}'_{FR}$ belongs to the next SV interval and will be rendered later.

[1] An alternative to deriving the whole texture $\hat{s}_{T,l}$ at initialization is to interpolate only required samples of $\hat{s}_{T,l}$ in the rendering step. When re-rendering is not performed, this approach reduces the total number of interpolated samples.

Case	1	1'	2	2'	3a	3a'	3b
$s'_{H,l}$	0	1	1	1	0	0	0
$s'_{H,r}$	1	0	1	1	0	0	0
$\lvert s'_{M,l} - s'_{M,r}\rvert < d_{th}$	*	*	*	*	0	0	1
$s'_{M,l} < s'_{M,r}$	*	*	0	1	0	1	*
s'_T	$s'_{T,l}$	$s'_{T,r}$	$s'_{T,r}$	$s'_{T,l}$	$s'_{T,l}$	$s'_{T,r}$	(3.3)

Table 3.1: s'_T depending on the hole maps $s'_{H,l}$ and $s'_{H,r}$, and the SV depths $s'_{M,l}$ and $s'_{M,r}$.

This way, the LCR provides an estimate for the hole so that no additional hole filling step is needed, and maps the right foreground edge positions correctly to the integer positions.

3.2.3.5 Render Mode for Margin Intervals

Because of different perspectives, IV samples are usually not warped to the right SV margin. Thus, when x_s is equal to w, a gap occurs to the right of $x'_s = \mathrm{f}(w)$. To fill it, the LCR sets the SV sample values in the margin interval $[\check{x}'_s, w]$ equal to the value of rightmost IV sample at w (R6). This way, the LCR provides a rough estimate for the gap.

3.2.3.6 Signals for View Combination

For right edge and margin intervals, some SV texture samples are extrapolated from neighboring samples and thus only roughly estimated. Moreover, in all render modes, SV samples might be rendered with erroneous depth. To identify such unreliable SV samples later in the view combination step (similarly to the VSRS), the LCR derives two additional signals while rendering an SV interval. The first signal, the hole map $s'_{H,l}$, indicates SV texture samples that are extrapolated in hole or margin filling. The second signal, the SV depth $s'_{M,l}$, is the warped IV depth. For simplification, $s'_{M,l}$ is derived in continuous intervals by mapping the IV depth $s_{M,l}$ to the SV position with integer precision. For other interval types, it is derived in the same way as $s'_{T,l}$. Both signals are used to identify and replace unreliable SV texture samples in the view combination step.

3.2.4 View Combination Step

As discussed above, some values of the left SV texture $s'_{T,l}$ can be unreliable. To replace such samples, the LCR combines the left SV texture $s'_{T,l}$ with a second SV texture $s'_{T,r}$ rendered from a right IV (i.e. the right SV texture) to a combined SV texture s'_T with higher reliability. This approach is similar to the one described for the VSRS in Appendix A.2.1. However, view combination in the VSRS and the LCR differs in three major aspects: First, the LCR performs the view combination step as point operation instantly after an SV sample $s'_{T,l}(\check{x}')$ has been rendered. Second, even when a sample is marked as hole in $s'_{H,l}$ or $s'_{H,r}$, the render modes for right edge or margin intervals have already provided a value for it, so that no further hole filling

operation is necessary, even if it is at a hole position in both views. And third, the LCR does not perform the complex evaluation of 7×7 windows to detect hole regions.

Table 3.1 shows the logic that results from these modifications and that the LCR uses for view combination. The decision how to combine samples is based on the SV depths $s'_{M,l}(\breve{x}')$ and $s'_{M,r}(\breve{x}')$, and the hole maps $s'_{H,l}(\breve{x}')$ and $s'_{H,r}(\breve{x}')$. The shown cases are motivated as follows:

1) When only one sample is disoccluded, take the other.

2) When both samples are disoccluded, take the background sample to avoid that foreground samples occur in the disocclusion.

3) When no sample is disoccluded, compare the depth difference $d_\Delta = |s'_{M,l}(\breve{x}') - s'_{M,r}(\breve{x}')|$ to a threshold $d_{th} = 0.3 \cdot (d_{max} - d_{min})$ with d_{max} and d_{min} denoting the maximal and minimal possible depth value (similar to the VSRS).

   a) If $d_\Delta > d_{th}$ is true, (which usually happens when foreground edges are not aligned in both SVs) perform background suppression by takeing the foreground sample to not impair the foreground object.

   b) Otherwise, consider both samples as reliable and combine them.

When both samples are reliable, as in case 3b), the LCR combines them similar as the VSRS. Obviously, when the distance of the SV to the right IV and the left IV differs, the SV texture rendered from the closer IV is more reliable. For this, the LCR derives the weighted average of both samples as follows:

$$s'_T(\breve{x}') = s'_{T,l}(\breve{x}') + \left[s'_{T,r}(\breve{x}') - s'_{T,l}(\breve{x}') \right] \cdot \frac{x'_V - x_{V,l}}{x_{V,r} - x_{V,l}} \tag{3.3}$$

with x'_V, $x_{V,l}$, and $x_{V,r}$ denoting the horizontal camera positions of the SV, the left IV, and the right IV, respectively. As the right factor in (3.3) is constant for a particular SV, the LCR can avoid the division and the multiplication by using a look-up table, which in particular reduces complexity of re-rendering.

In summary, with combination rules similar to the VSRS, the LCR exchanges or enhances unreliable samples with low complexity and provides an improved combined SV texture s'_T.

3.2.5 Rendering of Chroma Components

As this thesis assumes a 4:2:0 color format, the LCR needs to render two SV texture chroma components additionally. For this, it maps the IV texture luma and chroma samples jointly to the SV. However, as the chroma components have only quarter resolution, the LCR performs this only in each second row and column. This way, the LCR can render the SV texture luma and chroma components while iterating once over the IV.

Setup	Operations and memory accesses									Table
	AD	UA	CP	US	ML	LT	MR	MW	N_T	
Integer Extrapolation	2	2	7			1	2	1	15	C.1
+ Quarter Precision	+7.25	+1		+3.75	+4.5	+1			+17.5	C.2
Fractional Extrapolation	9.25	3	7	3.75	4.5	2	2	1	32.5	
+ Base Chroma			+1	+1			+0.5	+0.5	+3	C.5
+ Quarter Prec. Chroma	+5.25			+1.75	+4.5				+11.5	C.6
Full Extrapolation	14.5	3	8	6.5	9	2	2.5	1.5	47	

Table 3.2: Number of operations and memory accesses per SV sample for view extrapolation.

3.3 Complexity Evaluation

The LCR was developed as basis for partial re-rendering. How its design enables this with low complexity will be described in Chapter 4. However, as the LCR can also be applied as receiver-side renderer, this section discusses briefly its complexity for rendering of entire pictures.

Its complexity expressed in number of operations and memory accesses is provided in Tables 3.2 and 3.3. In particular, the tables show how many additions (AD), unary additions (UA), comparisons (CP), unary shifts (US), multiplications (ML), table look-ups (LT), reading memory accesses (MR), and writing memory access (MW) are required per SV sample. N_T denotes their total sum. Shown operations and memory accesses are logically needed and have been counted in simulations as discussed in Appendix C.1. Three setups—the *Integer*, the *Fractional* and the *Full* setup—are evaluated for view extrapolation and view interpolation. In the *Integer* setups, only luma samples are warped with integer sample precisions. In the *Fractional* setups, quarter sample precise is additionally enabled. Finally, the full setup also renders two additional chroma components.

Table 3.2 shows results for view extrapolation, i.e. for rendering using one IV only without view combination. The *Integer*, *Fractional*, and *Full* setup require about 15, 33, and 47 operations and memory accesses. Details how these are related to the rendering steps are given in Appendix C.2. There, it can be seen that about one third of the operations and memory accesses of the *Fractional* setup and about one half of the operations and memory accesses of the *Full* setup (and in particular all multiplications) are required for the interpolation of the IV texture $s_{T,l}$ to derive samples of $\hat{s}_{T,l}$ for quarter sample accuracy.

Table 3.3 provides results for view interpolation, which warps two IVs and combines the two resulting SVs. Consequently, its total number of operations and memory accesses is the double of that of view extrapolation plus the operations and memory accesses required by the view combination step. Total numbers for the *Integer*, *Fractional*, and *Full* setup are about 42, 77, and 108 operations and memory accesses per SV sample. Again, as for view extrapolation, a major part of operations and memory accesses in the *Fractional* and the *Full* setup is required for sample interpolation for quarter sample accuracy.

In summary, the complexity for rendering an entire picture varies widely from 15 operations and memory accesses for view extrapolation in the *Integer* setup to about 108 for view interpo-

Setup	Operations and memory accesses									Table
	AD	UA	CP	US	ML	LT	MR	MW	N_T	
2× *Integer Extrapolation*	4	4	14			2	4	2	30	C.1
+ View Combination	+2.9		+3			+1	+3	+2	+11.9	C.3
Integer Interpolation	6.9	4	17			3	7	4	41.9	
+ 2× Quarter Precision	+14.5	+2		+7.5	+9	+2			+35	C.2
Fractional Interpolation	21.4	6	17	7.5	9	5	7	4	76.9	
+ 2× Base Chroma			+2	+2			+1	+1	+6	C.5
+ 2× Q. Prec. Chroma	+10.5			+3.5	+9				+23	C.6
+ View Comb. Chroma	+0.98					+0.48	+0.48		+1.93	C.7
Full Interpolation	32.88	6	19	13	18	5.48	8.48	5	107.83	

Table 3.3: Number of operations and memory accesses per SV sample for view interpolation.

lation in the full setup. When employed for partial re-rendering by an encoder, the LCR thus allows trade-offs between enabled features and total complexity. Although the LCR does not use complex methods of the VSRS listed in Section 3.1, as e.g. warping of an increased number of samples, iterative median filtering, and separate processing steps, it requires at maximum of 108 operations and memory accesses in the full interpolation setup. However, some of these operations (in particular those for interpolation) are only required when the LCR renders entire pictures and not when it is used for its major purpose—partial re-rendering. Therefore, Chapter 4 will analyze its complexity in partial re-rendering.

3.4 Quality Evaluation

To limit computational complexity and to avoid over-fitting to particular rendering methods, the LCR only enables simple, fast, and geometrically correct rendering. Sophisticated methods for depth error concealment (e.g. boundary noise removal or temporal filtering, as used by VSRS) are not supported. In light of this, it is of interest which SV quality can be achieved when using low quality depth and when using high quality depth. For both cases, Figure 3.5 shows example pictures, which are originals; and rendered with VSRS, 1D-Fast (i.e. the LCR with boundary noise removal), and the LCR in the *full interpolation* setup.[2]

The pictures have been rendered from two neighboring IVs at stereo distance to the SV. As the *UndoDancer* sequence is computer generated, its depth has a high quality. Figure 3.5 (d) shows that the LCR provides an SV quality close to the original (a). Figures 3.5 (b) and (c), however, show some artifacts. These artifacts are caused by boundary noise removal that is applied by VSRS and 1D-Fast, although being harmful in case of undistorted depth. The *Balloons* sequence has distorted depth. Figures 3.5 (f), (g), and (h) thus show artifacts. However, the corona artifact marked with a pointer in Figure 3.5 (h) is not visible in (f) and (g) as boundary noise removal is effective there.

[2]The own contribution [Tech 18] provides further sample pictures and objective results.

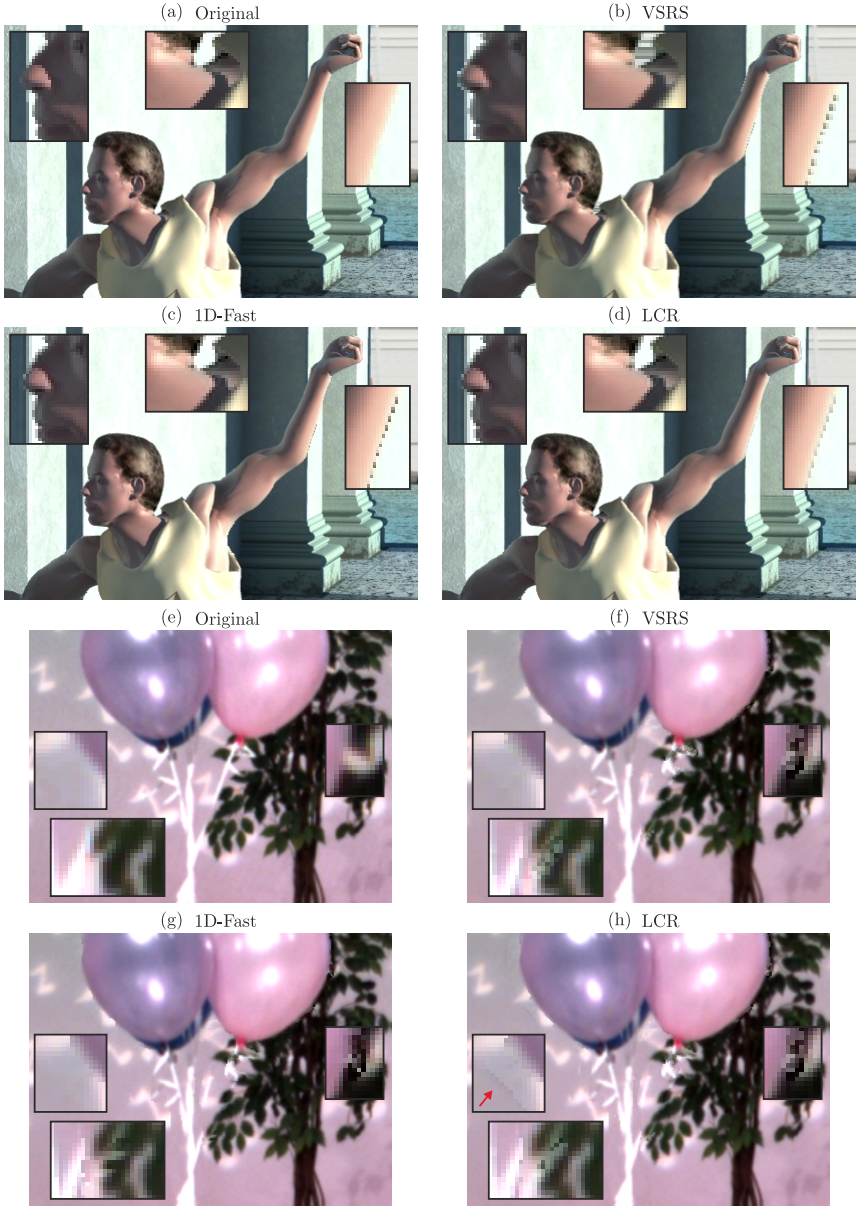


Figure 3.5: Comparison of rendering methods: (a)-(d) *UndoDancer* sequence (high quality depth); (e)-(h) *Balloons* sequence (low quality depth).

	Balloons	*Kendo*	*Newsp.*	*P.Hall2*	*P.Street*	*GTFly*	*Shark*	*UndoD.*	Mean
PSNR [dB]									
VSRS 1D	36.6	35.5	32.1	36.5	35.5	39.9	42.6	37.5	37.0
VSRS Gen	36.7	35.8	32.3	36.2	35.3	41.4	44.2	39.9	37.7
1D-Fast	36.9	35.9	32.4	36.2	35.2	42.7	46.1	40.5	38.3
LCR	37.0	36.0	32.4	36.3	35.4	42.9	46.1	40.6	38.3
MSSIM									
VSRS 1D	0.966	0.965	0.938	0.924	0.926	0.980	0.988	0.977	0.958
VSRS Gen	0.964	0.964	0.935	0.919	0.921	0.982	0.987	0.985	0.957
1D-Fast	0.965	0.965	0.935	0.920	0.922	0.984	0.991	0.986	0.958
LCR	0.966	0.965	0.934	0.920	0.923	0.984	0.991	0.986	0.959
FOM									
VSRS 1D	0.928	0.889	0.913	0.837	0.936	0.977	0.965	0.969	0.927
VSRS Gen	0.936	0.895	0.923	0.847	0.945	0.986	0.982	0.983	0.937
1D-Fast	0.934	0.893	0.920	0.847	0.943	0.988	0.985	0.986	0.937
LCR	0.934	0.893	0.918	0.846	0.943	0.989	0.985	0.986	0.937

Table 3.4: Rendering Quality; PSNR, MSSIM, and FOM; Reference: Original sequences.

In addition to the example pictures, Table 3.4 provides objective results of a comparison of the whole rendered sequences with the original sequences: The LCR and 1D-Fast achieve very similar results as the major difference between them is the boundary noise removal tool used by 1D-Fast. In terms of PSNR, the VSRS modes are outperformed by the other two methods. However, the mean Structural Similarity (MSSIM) index [Wang 04], as described in Appendix E.1.2, indicates that all three methods perform similarly. The same is true for Pratt's figure of merit (FOM) [Prat 78], which is a measure for the deviation of edge positions, as described in Appendix E.1.3.

In conclusion, example pictures and objective metrics show that the LCR renders geometrically correct with sufficient quality as targeted. In case of distorted depth, its rendering quality could be improved by boundary noise removal. For this, e.g. the JCT-3V uses the LCR (as part of the RM) for encoding while applying 1D-Fast at the receiver [Mull 14].

3.5 Chapter Summary

The chapter presented one of the main contributions of this thesis—the LCR. Motivation for its development is that rendering for SVD computation requires an algorithm that is capable of partial re-rendering with low-complexity. To enable this, the LCR avoids complex methods as performed by the VSRS and integrates several rendering steps into a single process, which can be applied per IV depth samples. More specifically, the LCR processes an IV interval for each depth sample, in which it

- maps the IV interval to an SV interval;

- identifies the SV interval's type by detecting occlusions and holes, and thus foreground object edges;
- adapts to the SV interval type and renders all samples related to the SV interval by quarter sample precise parallax compensation, hole filling, or special processing of foreground object edges;
- and combines rendered SV samples with SV samples rendered from another view to obtain the final synthesized texture by view interpolation.

This way, the LCR renders all SV samples related to an IV interval so that it is extensible for partial re-rendering with low complexity, as required for SVD computation. Furthermore, to allow a trade-off between complexity and rendering quality in SVD computation, the LCR design is modular. View interpolation, quarter sample warping precision, and processing of chroma components can be disabled.

After introducing the LCR, this chapter evaluated its complexity and view synthesis quality. The complexity evaluation focused on rendering of an entire view, thus for the case when the LCR is used as receiver-side renderer. Depending on the enabled features, it requires between 15 and 108 operations per SV sample, in which a significant part is required for sample interpolation to achieve quarter sample precise warping. When all features are enabled and high quality depth data is provided, the LCR provides a view synthesis quality similar to that of VSRS and 1D-Fast. However, in case of distorted depth data, it is outperformed by 1D-Fast because it does not include boundary noise removal.

In conclusion, the LCR renders with sufficient synthesized view quality and without using high complex rendering methods. Consequently, it is a suitable basis for partial re-rendering, as it will be discussed in Chapter 5. Beyond that, it is one of the receiver-side renderers used in evaluations presented in this thesis.

4 The Synthesized View Distortion Change

Section 2.2 discussed the SVD and how it can be estimated from the distorted IV depth by existing models and methods. Nevertheless, these models do not allow an exact and block-based SVD quantification. To overcome this, this chapter presents one of the main contributions of this thesis—the Synthesized View Distortion Change (SVDC). The SVDC is a concept for determining the change of the distortion of an SV texture that is related to a change of the IV depth map by rendering. To understand why it is needed, Sections 4.1 and 4.2 discuss requirements for a block-based distortion measure and how distorted regions in the IV depth and SV texture depend on each other, respectively. Based on these insights, Section 4.3 defines the SVDC. Finally, Section 4.4 discusses a framework and concepts that can be practically applied by depth encoders to compute the SVDC.

Related works: The SVDC and the RM have been presented in the own contribution [Tech 12c] and are discussed in the own contributions [Tech 18, Tech 12d, Schw 11a, Schw 11b, Schw 11c, Schw 12b, Schw 12a, Mull 13]. Some parts of this chapter are based on text and figures published in [Tech 18]. In contrast, this chapter discusses the SVDC and the RM in greater detail.

4.1 Requirements for a Distortion Function

This thesis targets to improve the performance of depth coding by regarding the SVD derived by rendering. This means that, when the depth encoder chooses between coding modes for a depth block, it needs to derive the related SVDs. Consequently, an SVD function providing a global SVD related to an entire IV depth $\tilde{s}_M$, an entire IV texture $\tilde{s}_T$, and an entire reference SV texture $s'_{T,Ref}$ is not useful. What is needed is a distortion function f fulfilling two requirements:

1. The distortion function f should provide the distortion D_B that is caused by the distortion of the IV depth map $\tilde{s}_M$ within a block B. This can be expressed as

$$D_B = \mathrm{f}\left[\tilde{s}_M(B)\right] \tag{4.1}$$

   with $\tilde{s}_M(B)$ denoting the part of $\tilde{s}_M(x, y)$ with $(x, y) \in B$.

2. The distortion function f should be additive. This means that the distortion computed from an IV depth block should be equal to the sum of the distortions computed independently from its sub-blocks. E.g. with $D_{B[1]}$ and $D_{B[2]}$ denoting the distortions of the sub-blocks $B[1]$ and $B[2]$, respectively; and $D_{B[1]\cup B[2]}$ denoting the distortion of a block $B[1] \cup B[2]$, the following should be true:

$$D_{B[1]\cup B[2]} = \mathrm{f}\left[\tilde{s}_M(B[1] \cup B[2])\right] \tag{4.2}$$

$$\stackrel{!}{=} D_{B[1]} + D_{B[2]} = \mathrm{f}\left[\tilde{s}_M(B[1])\right] + \mathrm{f}\left[\tilde{s}_M(B[2])\right] \tag{4.3}$$

In summary, a block-based and additive distortion function f is needed.

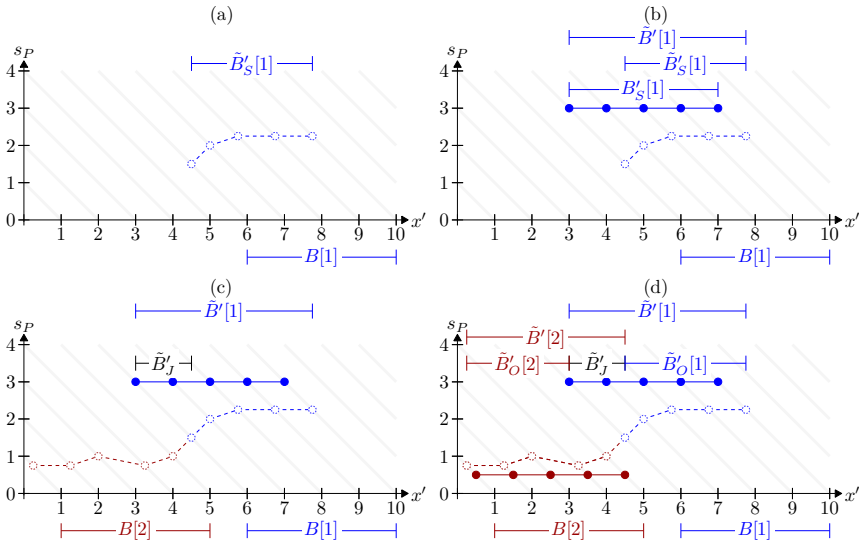


Figure 4.1: Problems in direct SVD measurement; Full dots and solid lines: Shifted with undistorted depth; Empty dots and dashed lines: Shifted with distorted depth; Blue: Related to $B[1]$; Red: Related to $B[2]$; (a) and (b): $\tilde{B}'$ can only be identified when undistorted depth is known for $B[1]$; (c) and (d): The SVD in $\tilde{B}'_J$ depends on data of $B[1]$ and $B[2]$.

4.2 Problems of Local Synthesized View Distortion Derivation

Given the two requirements, the question is how to define a function f. An approach that seems obvious is to identify the SV region $\tilde{B}'$ in that the SV texture $\tilde{s}'_T$ is distorted by the distorted depth in the IV region B and to sum the SVDs over $\tilde{B}'$. More specifically, this means by first calculating the per sample SVD

$$s'_D(x') = \mathrm{f}_{\mathcal{Q}}\left[\tilde{s}'_T(x'), s'_{T,Ref}(x')\right] \tag{4.4}$$

with a distortion function $\mathrm{f}_{\mathcal{Q}}$ operating sample-wise, as e.g. the squared differences $\left[\tilde{s}'_T(x') - s'_{T,Ref}(x')\right]^2$; and then by summing the per sample SVDs over $\tilde{B}'$, i.e.

$$D_B = \sum_{x' \in \tilde{B}'} s'_D(x'). \tag{4.5}$$

This approach leads directly to the question on how to identify $\tilde{B}'$. To answer it, Figure 4.1 shows an example, which is discussed in the following. Figure 4.1 (a) shows how the distorted depth $\tilde{s}_M$ in an IV region $B[1]$ shifts the IV texture samples s_T to their distorted SV positions. Obviously, the left- and rightmost distorted SV positions span an SV region $\tilde{B}'_S[1]$ which is affected by the distorted depth values in $B[1]$. However, the distorted depth values in $B[1]$ could

also affect SV positions outside $\tilde{B}'_S[1]$ because IV texture samples of $B[1]$ are erroneously not shifted there. Therefore, taking $\tilde{B}'_S[1]$ into account is not sufficient. What needs to be regarded additionally are the SV positions to that the IV texture samples are shifted with the undistorted depth, i.e. the depth that causes the per sample SVD $s'_D(x)$ to be zero at all SV positions x'.

Whether such undistorted depth is given depends on the reference SV texture $s'_{T,Ref}$. Two cases can be distinguished: In the first case $s'_{T,Ref}$ is a recorded texture. Then, the initial SV texture s'_T rendered from the initial IV depth s_M is in general already distorted with respect to $s'_{T,Ref}$. This means that it cannot be assumed that s_M provides correct SV positions. The SV region $\tilde{B}'[1]$ that comprises the SVD related to the depth distortion in $B[1]$ can thus not be identified. In the second case $s'_{T,Ref}$ is rendered from an initial IV depth s_M, which means that s_M can be considered as undistorted. Consequently, s_M in $B[1]$ defines the SV region $B'_S[1]$ that includes the SV positions to that the IV samples in $B[1]$ are shifted correctly. Knowing $B'_S[1]$, it is obvious that the SV region $\tilde{B}'[1]$ in that the SV texture is distorted by the depth distortion in $B[1]$ is given by the union of $B'_S[1]$ and $\tilde{B}'_S[1]$, as shown in Figure 4.1 (b). This means that a first problem for calculating the SVD in an SV region $\tilde{B}'$ is that

P1 the entire SV region $\tilde{B}'$ that is distorted by the depth distortion in B can only be identified when undistorted depth is known for B.

However, for the case that $\tilde{B}'$ is known, Figure 4.1 (c) shows that $\tilde{B}'[1]$ comprises a sub-region $\tilde{B}'_J$ in that the SV texture $\tilde{s}'_T$ not only depends on the IV data in $B[1]$, but also on IV data outside $B[1]$ in the IV region $B[2]$. More specifically, reasons for the SVD $D_{\tilde{B}'_J}$ in $\tilde{B}'_J$ are distortions in both IV depth regions $B[1]$ and $B[2]$: First, samples shifted from $B[2]$ to $\tilde{B}'_J$ have incorrect positions due to the depth distortion in $B[2]$; and second, they become only visible because IV samples of $B[1]$ are shifted to incorrect positions due to the depth distortion in $B[1]$. Consequently, the SVD $D_{\tilde{B}'_J}$ in $\tilde{B}'_J$ is not related to IV depth distortions in $B[1]$ or $B[2]$ solely, but is a joint SVD caused by both. In conclusion, as the distortion function f should reflect only the SVD related to $B[1]$ (see Item 1 in Section 4.1), another problem is that

P2 the SVD in $\tilde{B}'$ depends on data outside B.

Problem P2 has only been shown by the example in Figure 4.1, in which the joint SV region $\tilde{B}'_J$ becomes visible due to the depth distortion in $B[1]$. However, such regions with joint SVDs can also be caused by other effects: Consider e.g. the region $\tilde{B}'[1]$ is occluded by IV texture samples of further IV regions on the right of $B[1]$ that are erroneously shifted leftwards; or e.g. nonlinear processing of a view synthesis algorithm that renders SV samples in dependency of several IV depth values in different IV regions.

The problem P2 implies a third and a fourth problem. To calculate the SVD in $\tilde{B}'[1]$ in practice, the IV data outside $B[1]$ must be known. However, consider the case that $B[1]$ is processed before $B[2]$. Then the final IV depth in $B[2]$ is not yet known. Consequently, the third problem is that

P3 sub-regions $\tilde{B}'_J$ of $\tilde{B}'$ that not only depend on B and their joint SVD $D_{\tilde{B}'_J}$ might not be known when processing B.

To demonstrate the fourth problem, Figure 4.1 (d) shows two further sub-regions $\tilde{B}'_O[2]$ and

$\tilde{B}'_O[1]$ in that the SVD is solely related to $B[2]$ and $B[1]$, respectively. With (4.5) the SVD of $B[1]$ is the SVD in $\tilde{B}'[1]$ and thus the sum of the SVDs in $\tilde{B}'_J$ and $\tilde{B}'_O[1]$, i.e. $D_{\tilde{B}'_J} + D_{\tilde{B}'_{O[1]}}$. Accordingly, the SVD of $B[2]$ is the SVD in $\tilde{B}'[2]$ and thus the sum of the SVDs in $\tilde{B}'_O[2]$ and $\tilde{B}'_J$, i.e. $D_{\tilde{B}'_{O[2]}} + D_{\tilde{B}'_J}$. It is obvious that the sum of SVDs of $B[1]$ and $B[2]$ is not equal to the SVDs provided by (4.5) for the union of $B[1]$ and $B[2]$:

$$\mathrm{f}\left[\tilde{s}_M(B[1] \cup B[2])\right] = D_{\tilde{B}'_{O[1]}} + D_{\tilde{B}'_J} + D_{\tilde{B}'_{O[2]}} \tag{4.6}$$

$$\neq \mathrm{f}\left[\tilde{s}_M(B[1])\right] + \mathrm{f}\left[\tilde{s}_M(B[2])\right] = D_{\tilde{B}'_{O[1]}} + D_{\tilde{B}'_J} + D_{\tilde{B}'_{O[2]}} + D_{\tilde{B}'_J} \tag{4.7}$$

This means that, as additivity is required (see Item 2 in Section 4.1), the fourth problem is that

P4 summing over the SV region $\tilde{B}'$ leads to a non-additive function because joint SVDs are counted multiple times.

In summary, using the local SVD related to an IV depth region as distortion function has several problems. An undistorted depth map needs to be available to identify the distorted SV region $\tilde{B}'$ (P1). Furthermore, distorted SV regions $\tilde{B}'$ and IV regions B do not map bijectively, i.e. the SVD of a particular SV region $\tilde{B}'$ can be related to more than one IV region B (P2). This main problem implies that the SVD in such SV regions $\tilde{B}'$ cannot be taken into account for a current IV region B when the final depth of the other related IV regions is not known (P3). Moreover, the SVD in an SV region $\tilde{B}'$ might be counted multiple times for different IV regions B, which leads to a non-additive distortion measure (P4). For this, using a distortion function f providing the SVD in an SV region $\tilde{B}'$ related to a distorted IV depth region B is not feasible.

4.3 The Synthesized View Distortion Change

As the last section showed, it is not feasible to define an additive distortion function that provides the SVD in the SV region that is distorted by the depth distortion in the IV region B. However, the main target of an encoder is to minimize the overall SVD (under a particular bit rate constraint). For this, it is actually sufficient when a distortion function describes how the overall SVD depends on a depth distortion in an IV region B. This is the motivation for one of the main contributions of this thesis—the Synthesized View Distortion Change (SVDC).

4.3.1 Definition

The SVDC is the change of the total SVD of the whole SV texture that occurs when the IV depth map changes in an IV region B from initial to distorted depth while other IV depth regions contain distorted depth if already known, and initial depth otherwise. Consequently, the SVDC is conceptually agnostic about where the SVD related to a distorted IV region B is located in the SV, but depends on the processing state of the IV depth map.

Figure 4.2 shows how the SVDC D_S can be computed for an IV depth block B in four steps. The first step constructs two depth maps $\dot{s}_M$ and $\ddot{s}_M$. $\dot{s}_M$ contains the final distorted IV depth $\tilde{s}_M$ in regions that already have been processed by the encoder and the initial IV depth s_M in

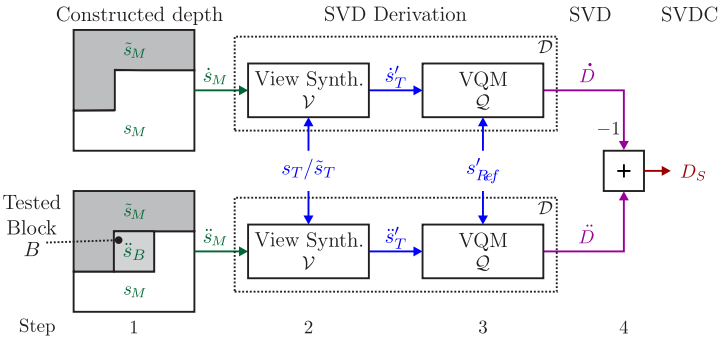


Figure 4.2: Computation of the SVDC.

other regions. $\ddot{s}_M$ contains the depth to test $\ddot{s}_B$ (called depth candidate) in the block B and is equal to $\dot{s}_M$ outside B. The second step renders two SV textures $\dot{s}'_T$ and $\ddot{s}'_T$ from $\dot{s}_M$ and $\ddot{s}_M$, respectively, with a view synthesis method $\mathcal{V}$ and using an IV texture (which can be the coded $\tilde{s}_T$ or the uncoded IV texture s_T as discussed later). The third step compares $\dot{s}'_T$ and $\ddot{s}'_T$ to a reference SV texture $s'_{T,Ref}$ with a video quality metric $\mathcal{Q}$, which provides the respective SVDs $\dot{D}$ and $\ddot{D}$. Finally, the fourth step subtracts $\dot{D}$ from $\ddot{D}$ to derive the SVDC D_S.

The four steps show that SVDC D_S can be interpreted as the SVD difference $\ddot{D} - \dot{D}$ of two SV textures, $\ddot{s}'_T$ and $\dot{s}'_T$, rendered from two IV depth maps, $\ddot{s}_M$ and $\dot{s}_M$, i.e.

$$\begin{aligned} D_S &= \ddot{D} && - \dot{D} \\ &= \mathrm{f}_{\mathcal{Q}}\left(\ddot{s}'_T, s'_{T,Ref}\right) && - \mathrm{f}_{\mathcal{Q}}\left(\dot{s}'_T, s'_{T,Ref}\right) \\ &= \mathrm{f}_{\mathcal{Q}}\left(\mathrm{f}_{\mathcal{V}}\left(s_T, \ddot{s}_M\right), s'_{T,Ref}\right) && - \mathrm{f}_{\mathcal{Q}}\left(\mathrm{f}_{\mathcal{V}}\left(s_T, \dot{s}_M\right), s'_{T,Ref}\right). \end{aligned} \tag{4.8}$$

Using the joint view synthesis and distortion function $\mathrm{f}_{\mathcal{D}}$, and omitting s_T, $s'_{T,Ref}$, and $\dot{s}_M$, the SVDC function $\mathrm{f}_{\mathcal{S}}$ can be defined as

$$D_S = \mathrm{f}_{\mathcal{S}}(\ddot{s}_M) = \mathrm{f}_{\mathcal{D}}(\ddot{s}_M) - \mathrm{f}_{\mathcal{D}}\left(\dot{s}_M\right). \tag{4.9}$$

The SVDC is thus an exact measure for the SVD introduced by the difference of the depth maps—the depth candidate $\ddot{s}_B$ in B. The SVDC depends on the IV data outside B and on the old IV depth in B. However, as it is derived by subtracting the SVD due to these conditions, it is the SVD that is solely caused by the depth change $\ddot{s}_B$ in B. It thus fulfills the first requirement given by Item 1 in Section 4.1.

This property implies directly that the SVDC is an additive measure as shown in the following. Consider that the IV depth map consists of N non-overlapping blocks $B[i]$ with $i = 1...N$. Consider furthermore that the encoder processes the IV depth map in multiple iterations. In the i-th iteration the encoder processes $B[i]$. Consequently, the distorted depth $\tilde{s}_M$ is given for blocks $B[n]$ with $n < i$; only initial depth s_M is given for blocks $B[n]$ with $n > i$; and the SVDC should be computed for $B[i]$ containing the depth candidate $\ddot{s}_B[i]$. Consider further-

more, without loss of generality, that the depth candidate $\ddot{s}_B[i]$ for iteration i corresponds to the final distorted depth $\tilde{s}_M$. Then, $\ddot{s}_M[i]$ is given as follows:

$$\ddot{s}_M[i] = \begin{cases} \tilde{s}_M(x,y) & \text{for } (x,y) \in \bigcup_{n=1}^{i} B[n] \\ s_M(x,y) & \text{for } (x,y) \in \bigcup_{n=i+1}^{N} B[n] \end{cases} \tag{4.10}$$

Furthermore, the SVD $\ddot{D}[i]$ of the SV texture $\ddot{s}'_T[i]$ rendered from $\ddot{s}_M[i]$ is given by

$$\ddot{D}[i] = \mathrm{f}_{\mathcal{D}}\left(\ddot{s}_M[i]\right). \tag{4.11}$$

(4.10) and (4.11) show that the SVDC $D_S[i][j]$ that occurs when jointly changing depth blocks $B[n]$ with $n \in (j+1)...i$ from s_M to $\tilde{s}_M$ is given by

$$D_S[i][j] = \ddot{D}[i] - \ddot{D}[j]. \tag{4.12}$$

On the other hand, when separately changing depth blocks $B[n]$ with $n \in (j+1)...i$ from s_M to $\tilde{s}_M$, the sum of the related SVDCs is given by

$$\sum_{n=j+1}^{i} D_S[n][n-1] = \sum_{n=j+1}^{i} \ddot{D}[n] - \ddot{D}[n-1] = \ddot{D}[i] - \ddot{D}[j] \tag{4.13}$$

and thus equal to the SVDC $D_S[i][j]$ of the joint depth change. Therefore, the SVDC is an additive measure and fulfills the requirement given by Item 2 in Section 4.1.

In summary, the SVDC quantifies the change of the overall SVD of the entire SV texture that occurs when an encoder changes a block of the IV depth map. It is—in contrast to the SVD in a distorted SV region—a block-based and additive measure.

4.3.2 Example and Local Interpretation

Figure 4.3 shows warping charts related to an SVDC computation for the scenario from Figure 4.1, as discussed in Section 4.2. More specifically, Figures 4.3 (a) to (c) show in three iterations ($i = 0...2$, respectively) how the SVDC can be computed for the IV blocks $B[1]$ and $B[2]$.

$i = 0$: $\ddot{s}_M[0]$ is equal to the initial depth s_M and the initial SVD $\ddot{D}[0]$ is computed by comparing the respected initial SV texture $\ddot{s}'_T[0]$ to the reference SV texture $s'_{T,Ref}$.

$i = 1$: According to (4.10), $\ddot{s}_M[1]$ contains the distorted IV depth that should be evaluated in $B[1]$. $B[2]$ contains the initial depth as the distorted depth is not yet known. The SVDC $D_S[1][0]$ that is related to the depth change in $B[1]$ is computed by calculating the overall SVD $\ddot{D}[1]$ of the related SV texture $\ddot{s}'_T[1]$ and by subtracting $\ddot{D}[1] = \ddot{D}[0]$.

$i = 2$: $\ddot{s}_M[2]$ contains the distorted IV depth that should be evaluated in $B[2]$. $B[1]$ contains the distorted depth determined in iteration 1. The SVDC $D_S[2][1]$ that is related to the

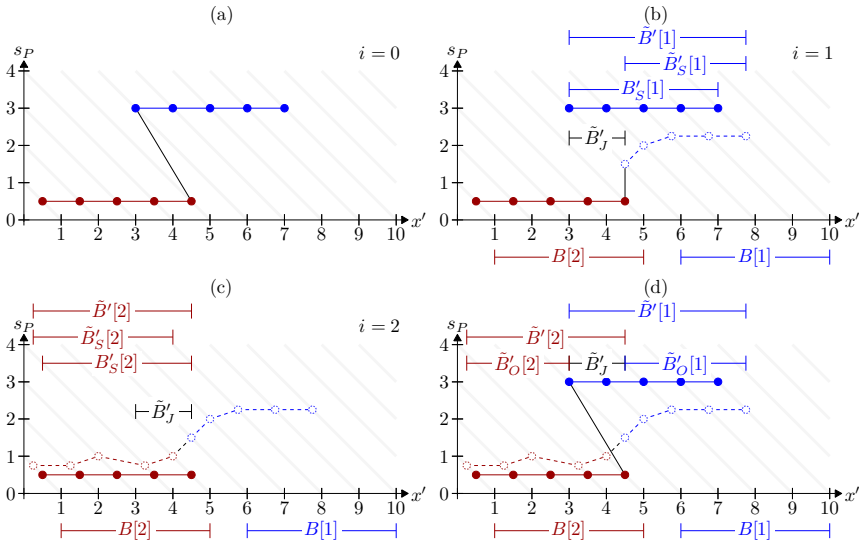


Figure 4.3: SVDC computation. Empty dots and dashed lines: Shifted with distorted depth; Full dots and solid lines: Shifted with initial depth. Blue: Related to $B[1]$; Red: Related to $B[2]$.

depth change in $B[2]$ is computed by calculating the overall SVD $\ddot{D}[2]$ of the related SV texture $\ddot{s}'_T[2]$ and by subtracting $\dot{D}[2] = \ddot{D}[1]$.

The SVDC could be computed in the three iterations by only considering the global SVDs. However, Figure 4.3 shows different SV regions in that the SV is locally affected. Based on these regions, the following discusses why the problems P1 to P4 that occur for the SVD measure, as discussed in Section 4.2, do not occur for the SVDC measure.

Problem P1 is that the distorted IV depth is required to identify the distorted SV region in that the SV texture is affected by the distortion in the IV depth block B. As SVDC computation regards the SVD of an entire SV texture by concept, P1 does not occur. However, Figures 4.3 (b) and (c) show two SV regions $\tilde{B}'[1]$ and $\tilde{B}'[2]$ in that the SV changes because of the depth change in $B[1]$ and $B[2]$, respectively. $\tilde{B}'[1]$ and $\tilde{B}'[2]$ are given by the left- and rightmost SV positions obtained by shifting IV texture samples with initial and distorted IV depth data in block $B[1]$ and $B[2]$, respectively. Identifying $\tilde{B}'[1]$ and $\tilde{B}'[2]$ requires the changed IV depth and the initial IV depth—but not the undistorted IV depth. As the changed and the initial IV depth are always given with respect to a current iteration, a changed SV region $\tilde{B}'$ that is related to the depth change in an IV region B can always be determined. Although SVDC computation does conceptually not require to determine the changed SV region $\tilde{B}'$, it is important for fast SVDC computation by partial re-rendering as it will be discussed in Section 4.4.

Problem P2 is that the SVD in an SV region $\tilde{B}'$ depends on data outside B. This implies that

Iteration i	$D_{\tilde{B}'_O[2]}$	$D_{\tilde{B}'_J}$	$D_{\tilde{B}'_O[1]}$	$D_S[i][i-1]$	$D_S[i][i-2]$
0	0	0	0	–	–
1	0	D_J	$\tilde{D}_1$	$D_J + \tilde{D}_1$	–
2	$\tilde{D}_2$	$\tilde{D}_J$	$\tilde{D}_1$	$\tilde{D}_2 + \tilde{D}_J - D_J$	$\tilde{D}_2 + \tilde{D}_J + \tilde{D}_1$

Table 4.1: SVDs of the SV regions in Figure 4.3 in iterations 0, 1, and 2, and related SVDCs.

the SVD cannot be computed as data outside B might not be known (P3) and that summing SVDs related to different IV regions B leads to a non-additive measure (P4). Joint SV regions $\tilde{B}'_J$ also occur in SVDC computation. However, they are not problematic, as shown in the following based on the example in Figure 4.3.

Problem P3 does not occur as SVDC computation assumes distorted depth when already known and initial depth otherwise. The example shows the effect of this approach on SVDC computation in $\tilde{B}'_J$. Iteration 1 calculates the SVDC in $\tilde{B}'_J$ for $B[1]$ assuming initial depth data in $B[2]$, i.e. that the SV texture that becomes visible in $\tilde{B}'_J$ (because of the IV depth change in $B[1]$) is the SV texture rendered from initial depth data in $B[2]$. Then in iteration 2, the distorted depth in $B[1]$ is already known, so that the SVD in $\tilde{B}'_J$ is calculated assuming that depth, i.e. that incorrectly shifted samples of $B[2]$ are visible in $\tilde{B}'_J$. In summary, the SVD in $\tilde{B}'_J$ is always calculated based on the actual state of IV depth outside B.

Problem P4 does not occur as a) finally not the SVD, but the SVDC is computed, and b) the state of the IV depth is iteratively changed from initial to distorted depth. The SVDC measure is thus additive, which can be easily shown for the example in Figure 4.3: Consider that the SVDs $D_{\tilde{B}'_O[2]}$, $D_{\tilde{B}'_J}$, and $D_{\tilde{B}'_O[1]}$ of the three SV regions are given in the three iterations as shown in Table 4.1, i.e. the initial SVDs are zero and the final SVDs are $\tilde{D}_2$, $\tilde{D}_J$, and $\tilde{D}_1$, respectively. Then, the additivity of the SVDC is shown by (4.14), which compares the sum of SVDCs $D_S[1][0] + D_S[2][1]$ due to consecutive changes of the depth in $B[1]$ and $B[2]$ to the SVDC $D_S[2][0]$ caused by a joint change.

$$\begin{aligned} &\mathrm{f}_{\mathcal{S}}\left[\tilde{s}_M(B[1])\right] + \mathrm{f}_{\mathcal{S}}\left[\tilde{s}_M(B[2])\right] &&= D_S[1][0] + D_S[2][1] &&= D_J + \tilde{D}_1 + \tilde{D}_2 + \tilde{D}_J - D_J \\ &= \mathrm{f}_{\mathcal{S}}\left[\tilde{s}_M(B[1] \cup B[2])\right] &&= D_S[2][0] &&= \tilde{D}_2 + \tilde{D}_J + \tilde{D}_1 \end{aligned} \tag{4.14}$$

From (4.14) it becomes clear that problem P4 does not occur as the SVD D_J in the overlapping SV region $\tilde{B}'_J$ that is calculated under the preliminary assumption that $B[2]$ contains initial depth is replaced in iteration 2 by the final SVD $\tilde{D}_J$ due to the final depth in $B[1]$ and $B[2]$.

4.3.3 Conclusion

As the SVDC is the change of the overall SVD, problems that occur when considering the local SVD related to a depth distortion are avoided. SVDC computation does not require an undistorted, but only an initial depth map. Furthermore, it bypasses the problem that SV regions can be related to more than one IV region by its iterative approach. More specifically, it establishes

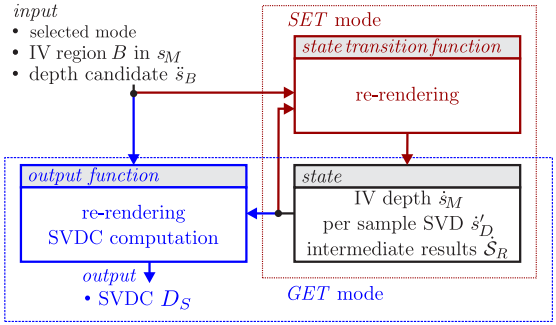


Figure 4.4: The Renderer Model (RM). When invoked in *GET* mode, the RM derives the SVDC D_S related to the depth candidate $\ddot{s}_B$. The derivation depends on the RM's state, but does not alter it. In *SET* mode, the RM adopts $\ddot{s}_B$ and performs re-rendering to update its state.

the relationship between the overall SVD and distorted IV depth regions not spatially, i.e. by identifying the SV region that contains the SVD related to a distorted IV depth region, but iteratively, i.e. by identifying the overall SVDC related to an IV depth change while considering the current processing state of the IV depth map. With this approach, the SVDC calculated for a particular IV depth region depends on the processing order of the IV depth map. This is similar to other signals derived in encoding, e.g. the bits required for a current block or prediction signals, which also depend on decisions taken before. As SVDC computation considers all information that are available in an iteration (i.e. the final distorted depth when already known and the initial depth otherwise) and since an exact SVD related solely to a distorted IV depth region can not be exactly defined, the SVDC can be considered as the best possible measure.

4.4 Synthesized View Distortion Change Computation

The last section discussed the main ideas behind the SVDC and their theoretical implications. In contrast, this section presents concepts for SVDC computation in practice. More specifically, it embeds SVDC computation in a framework that can be used by encoders and shows that the SVDC can be efficiently computed by partial re-rendering.

4.4.1 Framework

Section 4.3 showed that, in contrast to conventional distortion metrics, SVDC computation does not only depend on the depth candidate $\ddot{s}_B$ in the IV region B, but also on the IV depth $\dot{s}_M$ of the previous iteration. Therefore, SVDC computation can be modeled as a process having a state. In order to calculate the SVDC correctly, this state needs to be aligned with the decisions taken by the encoder such that $\dot{s}_M$ contains the final distorted depth $\tilde{s}_M$ when already known and the initial depth s_M otherwise. To do so, this section defines a framework—called Renderer

```
RM( INIT , s_M )
for each IV region B[i] to be processed in s_M
    for each depth candidate s̈_B[i][k] of B[i]
        D_S[i][i − 1][k] = RM( GET B[i], s̈_B[i][k] )
    Evaluate SVDCs D_S[i][i − 1][k] to select a candidate s̈_B[i][k̃]
    RM(SET, B[i], s̈_B[i][k̃] )
```

Figure 4.5: Example for the application of the RM in an encoder. After initialization, the encoder tests depth candidates $\ddot{s}_B[i][k]$ for the IV regions $B[i]$, with i indexing the IV regions and k their depth candidates. Finally, it selects and adopts $\ddot{s}_B[i][\tilde{k}]$.

Model (RM)—that provides an interface to an encoder for SVDC computation and updating its state.

Figures 4.4 and 4.5 show the RM and an example how an encoder can use it. For simplification, it is at first assumed that the RM's state only comprises the IV depth $\dot{s}_M$. Initially, the IV depth $\dot{s}_M$ is equal to the initial depth map s_M. While processing s_M, the encoder can then input an IV region B and the related depth candidate $\ddot{s}_B$ to the RM, before invoking it in one of two modes, called *GET* and *SET* mode. In *GET* mode, the RM computes the SVDC D_S caused by $\ddot{s}_B$ in B using its current state, which remains unchanged while doing so. This way, the encoder can compute the SVDC for multiple candidates $\ddot{s}_B$ for B before choosing a candidate. Figure 4.5 demonstrates this: The encoder tests several candidates $\ddot{s}_B[i][k]$ for $B[i]$, evaluates the related SVDCs $D_S[i][i-1][k]$, and finally selects $\ddot{s}_B[i][\tilde{k}]$. When the encoder has selected a candidate, the final depth for the respective IV region B is known and can be adopted to the RM's state, so that it is regarded when testing candidates for other IV regions. For this, the encoder can invoke the RM's *SET* mode, which replaces depth values of $\dot{s}_M$ in B by the depth candidate $\ddot{s}_B$, i.e. $\ddot{s}_B[i][\tilde{k}]$ in Figure 4.5. This way, the RM's state is always aligned with decisions taken by the encoder so that SVDC computation is always based on all adopted candidates.

4.4.2 Fast Computation by Partial Re-Rendering

So far it was assumed that the RM only comprises the IV depth $\dot{s}_M$ as its state. This is actually sufficient to derive the SVDC for a depth candidate $\ddot{s}_B$ in a *GET* operation, i.e. by 1) constructing the changed IV depth $\ddot{s}_M$ from the IV depth $\dot{s}_M$ and the depth candidate $\ddot{s}_B$ provided by the encoder; and 2) by applying (4.8), which includes full rendering of the two SV textures, $\dot{s}'_T$ and $\ddot{s}'_T$, and the computation of their SVDs, $\dot{D}$ and $\ddot{D}$. However, it is obvious that rendering of the full SV textures $\dot{s}'_T$ and $\ddot{s}'_T$ is not feasible in practical applications because of its computational complexity. Therefore, an alternative method for low complex SVDC computation is needed.

As analyzed in Section 4.3.2, the change of the IV depth $\dot{s}_M$ in B implies in general only a small change of the SV texture in a region $\tilde{B}'$. An optimization is thus to not render the whole SV texture, but to re-render only the SV region $\tilde{B}'$ that changes because of the depth change and to compute the SVDC in $\tilde{B}'$. Conceptually, this can be achieved by storing intermediate

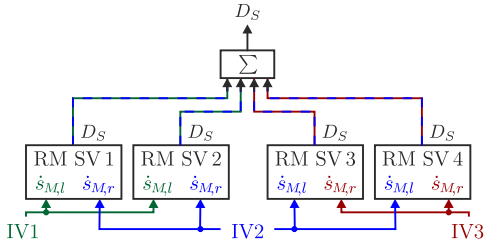


Figure 4.6: SVDC computation for three IVs and four SVs.

results, in the following denoted $\mathcal{S}_R$, of the rendering process that allow partial re-rendering as state of the RM. Which intermediate results $\mathcal{S}_R$ the RM actually stores depends on the applied rendering algorithm. (Section 5.1.1 will discuss this for an LCR-based RM.)

More specifically, the RM can be implemented as follows. On initialization the RM

I1 renders the SV texture s'_T and intermediate results $\mathcal{S}_R$ using the initial IV depth s_M,

I2 computes the initial per sample SVD $s'_D = [s'_T - s'_{T,Ref}]^2$,

I3 stores s'_D and $\mathcal{S}_R$ as state variables $\dot{s}'_D$ and $\dot{\mathcal{S}}_R$, respectively.

Then, in *GET* mode the RM

G1 uses $\dot{\mathcal{S}}_R$ and $\ddot{s}_B$ to determine the SV region $\tilde{B}'$ in that $\dot{s}'_T$ and $\ddot{s}'_T$ would differ,

G2 uses $\dot{\mathcal{S}}_R$ and $\ddot{s}_B$ to re-render $\ddot{s}'_T$ in $\tilde{B}'$,

G3 computes the per sample SVD $\ddot{s}'_D = [\ddot{s}'_T - s'_{T,Ref}]^2$ in $\tilde{B}'$,

G4 computes the SVDC

$$D_S = \sum_{x' \in \tilde{B}'} \left[\ddot{s}'_D(x') - \dot{s}'_D(x') \right]. \tag{4.15}$$

And in *SET* mode, the RM

S1 to **S3** performs **G1** to **G3**, respectively.

S4 stores $\ddot{s}'_D$ and $\ddot{\mathcal{S}}_R$ as state variables $\dot{s}'_D$ and $\dot{\mathcal{S}}_R$, respectively,

This way, the RM can calculate the SVDC by only re-rendering the part of SV texture $\dot{s}'_T$ that is affected by the depth candidate $\ddot{s}_B$. In contrast to rendering whole SV textures, the computational complexity is thus significantly reduced.

4.4.3 View Interpolation and Multiple Synthesized Views

So far, the RM uses one IV depth and one SV texture. However, the 3D video system regarded in this thesis renders multiple SVs using view interpolation. SVDC computation can be extended to this scenario by using one RM per SV and finally summing up SVDCs as shown in Figure 4.6. Furthermore, to take view interpolation into account, the encoder must signal in *SET* and *GET* mode which of the two IV depth maps, i.e. $s_{M,l}$ or $s_{M,r}$, is changed by the depth candidate. The concept of using initial depth when the final distorted depth is unknown can straightforwardly be extended to the two IV depth maps. This means that when the encoder e.g. processes the left IV depth $s_{M,l}$ first, the RM can assume the initial IV depth $s_{M,r}$ for the right view. Then, when it processes $s_{M,r}$ the RM can take the final distorted depth $\tilde{s}_{M,l}$ of the left view into account.

4.5 Chapter Summary

The chapter introduced a further main contribution of this thesis—the SVDC. Motivation for its development is that depth encoders require an SVD measure that is block-based and additive. These requirements are not fulfilled by the obvious approach of considering the SVD that occurs in the SV region that is distorted by the depth candidate in the evaluated IV region. The reasons for this are that the undistorted IV depth must be known to identify the distorted SV region and that this SV region can map to multiple IV regions.

The SVDC bypasses these problems by considering the change of the overall SVD of the entire SV texture that is caused when the initial depth in the evaluated IV region is changed to the depth candidate. In doing so, distorted depth is assumed outside the evaluated IV region if already known and initial depth otherwise. Consequently, the SVDC calculated for a particular depth candidate depends on the processing order of the IV depth. However, as SVDC computation is based on the best possible assumptions on IV depth data outside the evaluated IV region, it can be considered as the best possible measure.

SVDC computation depends on the state of the IV depth before the depth change; it can thus be modeled as process having a state. For this, this chapter introduced the Renderer Model. The RM is a framework that provides an interface to encoders for SVDC computation (*GET* mode) and updating the state of the SVDC computation process (*SET* mode). The SVDC is conceptually defined as the change of the overall SVD of an entire SV texture. However, as in general a depth candidate only implies the change of a small part of the SV texture, the SVDC can be computed by only re-rendering this part. To enable this, additional intermediate results of the rendering process can be stored as the state of the RM.

5 Partial Depth Image Based Re-Rendering for SVDC Computation

Chapter 4 discussed how an encoder can interact with the RM to obtain the SVDC for a depth candidate. In contrast to this discussion, which covered the question how the RM can compute the SVDC only conceptually, this chapter proposes and discusses an actual algorithm that enables SVDC computation by partial re-rendering—a further main contribution of this thesis. For this, Section 5.1 suggests how the LCR algorithm presented in Chapter 3 can be extended to the RM by enabling partial re-rendering and SVDC computation. Then, Section 5.2 discusses the complexity of the RM and compares the RM to an unoptimized variant.

Related works: The major part of this chapter (i.e. text and figures) has been presented in [Tech 18], which is an own contribution. Partial Re-rendering is also mentioned in the own contribution [Tech 12d] and overview papers [Schw 11a, Schw 11b, Schw 11c, Schw 12b, Schw 12a, Mull 13].

5.1 Extending the Low Complexity Renderer: The Renderer Model

The LCR, as discussed in Chapter 3, renders a complete SV by processing the complete IV depth map $s_{M,l}$. In contrast to this, the RM's objective is to quantify the SVDC in the SV region $\tilde{B}'$ that changes when the IV region B in $s_{M,l}$ changes to the depth candidate $\tilde{s}_B$[1]. This means that generally the RM does not need to render a complete SV texture, but only a part of it—the changed SV region $\tilde{B}'$. This section discusses how such partial re-rendering is enabled based on the LCR and thus extends the LCR to the RM.

The basic idea of partial re-rendering, as discussed in the last section, is as follows: First, the RM renders the complete SV and stores the intermediate results as its state. The RM does this only once as initialization step. Then, an encoder can repeatedly input different depth candidates for different IV regions. For each depth candidate, $\tilde{s}_B$ the RM re-renders the changed SV region $\tilde{B}'$ related to the depth candidate $\tilde{s}_B$ using the stored intermediate results. Depending on the mode indicated by the encoder—*GET* or *SET* mode—the RM either only computes the SVDC in the changed SV region $\tilde{B}'$ without changing its state or only updates its state by storing changed intermediate results to adopt the depth candidate.

How the RM performs partial re-rendering with low-complexity is addressed in the following. More specifically, it is answered which signals the RM stores as its state (Section 5.1.1), how the RM starts re-rendering at random access positions (Section 5.1.2), where to start and stop in the IV to re-render the entire changed SV region (Section 5.1.3), how the RM identifies the stop position while rendering (Section 5.1.4), and how the RM can exploit partial re-rendering for SVDC computation (Section 5.1.5).

[1] For clarity of description and without loss of generality, this chapter assumes that the left IV depth changes from uncoded to coded data, i.e. $\ddot{s}_B = \tilde{s}_B$, $\ddot{s}_M = \tilde{s}_{M,l}$, $\dot{s}_M = s_{M,l}$, and accordingly for other signals.

Signal			Introduced in	Scenario: Extrapolation Type	Extrapolation Pels/$(w\cdot h)$ Int.	Extrapolation Frac.	Extrapolation Full	Interpolation Type	Interpolation Pels/$(w\cdot h)$ Int.	Interpolation Frac.	Interpolation Full	Alt. interp. Type	Alt. interp. Pels/$(w\cdot h)$ Int.	Alt. interp. Frac.	Alt. interp. Full
IV depth	Left	$s_{M,l}$	3.2	S	1	1	1	S	1	1	1	S	1	1	1
	Right	$s_{M,r}$		–				–				S	1	1	1
Upsampled IV texture	Left	$\hat{s}_{T,l}$	3.2.3.2	C	1	4	6	C	1	4	6	C	1	4	6
	Right	$\hat{s}_{T,r}$		–				–				C	1	4	6
Occlusion signal	Left	$s_{O,l}$	5.1.2	S	1	1	1	S	1	1	1	S	1	1	1
	Right	$s_{O,r}$		–				–				S	1	1	1
SV texture	Left	$s'_{T,l}$	3.2	N				N				S	1	1	1.5
	Right	$s'_{T,r}$		–				C	1	1	1.5	S	1	1	1.5
SV depth	Left	$s'_{M,l}$	3.2.3.6	–				N				S	1	1	1
	Right	$s'_{M,r}$		–				C	1	1	1	S	1	1	1
Hole map	Left	$s'_{H,l}$	3.2.3.6	–				N				S	1	1	1
	Right	$s'_{H,r}$		–				C	1	1	1	S	1	1	1
Combined SV texture		s'_T	3.2	–				N				N			
Per sample SVD		s'_D	4.2	S	1	1	1	S	1	1	1	S	1	1	1
Reference SV texture		$s'_{T,Ref}$	4.3	C	1	1	1.5	C	1	1	1.5	C	1	1	1.5
Memory consumption		M_S			5	8	10.5		8	11	14		14	20	25.5

Table 5.1: Signals and memory consumption M_S in multiples of the picture size $w\cdot h$ (discussed in Section 5.2.3) for different RM setups; Type: – not use; N used, but no need to store; C constant; S state variable.

5.1.1 Initialization and State of the Renderer Model

For initialization, the RM renders the complete SV and stores several intermediate results of the rendering and the SVDC computation process as its state. Stored signals are shown in Table 5.1 for three different scenarios and marked with S and C.

In the extrapolation scenario, the RM applies view extrapolation. For this, it stores the left IV depth map $s_{M,l}$, the per sample distortion s'_D, and an occlusion signal $s_{O,l}$, which is introduced later. Beyond these state variables that change when the depth changes, the RM stores some variables, which are constant while testing different depth candidates. These are the upsampled IV texture $\hat{s}_{T,l}$ and the reference SV texture $s'_{T,Ref}$.

In the interpolation scenario, the RM applies view interpolation while depth changes can be applied to one IV depth only (here the left IV depth). This suffices generally as depth maps are usually encoded sequentially. In addition to signals of the extrapolation scenario, the RM stores signals of the right IV that are required for view combination. These are the hole map $s'_{H,r}$, the SV depth $s'_{M,r}$, and the SV texture $s'_{T,r}$, which are constant.

In the alternative interpolation scenario, the RM applies view interpolation while changes to

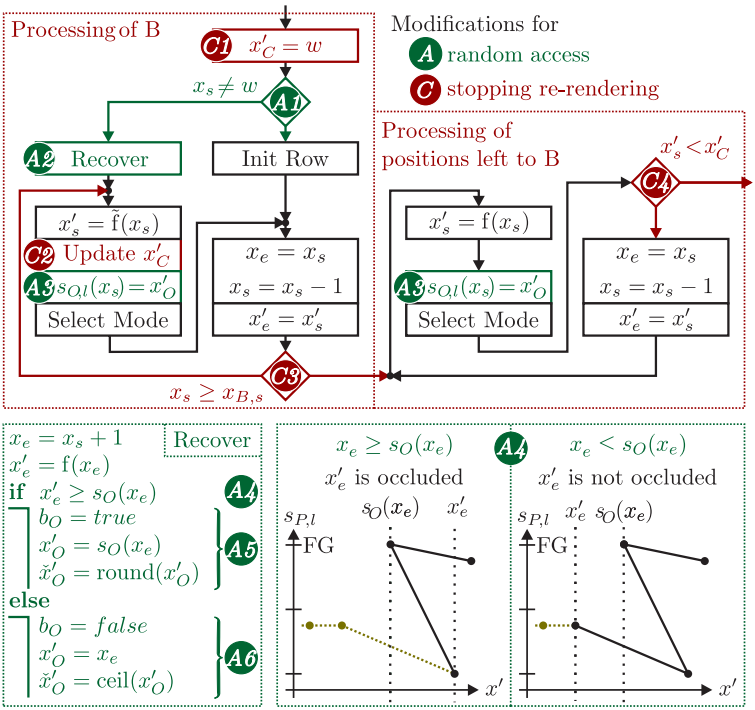


Figure 5.1: Extension of the LCR's interval-wise processing (see Figure 3.2) for partial re-rendering, i.e. random access and stopping. Processing starts at IV position x_s. ©IEEE

both IV depth maps can be performed, which might be useful e.g. for parallel encoding of depth maps or for depth filtering. For this, the RM stores additionally the right IV's depth $s_{M,r}$, texture $\hat{s}_{T,r}$, and occlusion signal $s_{O,r}$; furthermore, for view combination, it stores the left SV's texture $s'_{T,l}$, depth $s'_{M,l}$, and hole map $s'_{H,l}$.

Beyond signals that belong to the RM's state, Table 5.1 shows signals that are derived while rendering, but directly processed so that the RM does not store them (N). E.g. for the interpolation setup, the RM uses the SV texture $s'_{T,l}$, the SV depth $s'_{M,l}$, and the hole map $s'_{H,l}$ directly for view combination and then the combined SV texture s'_T directly for SVD computation.

5.1.2 Recovery of Auxiliary Variables for Occlusion Detection

Beyond the signals in Table 5.1, which are defined for each sample position of an SV or IV, the LCR employs three auxiliary variables, as presented in Section 3.2.2.1. x'_O and $\check{x}'_O$ track the leftmost fractional and integer SV positions that have already been processed; b_O indicates whether the left boundary of the last and thus the right boundary of the current SV interval is

occluded. These variables are initialized when the LCR starts at the right picture boundary w and are then continuously updated while iterating. However, when the RM starts re-rendering at a position not equal to w, they are unknown. To resolve this and recover them, the LCR's interval-wise processing (see Figure 3.2) is modified, as shown by items (A1), (A2), and (A3) on the left in Figure 5.1. One of the modifications derives an occlusion signal $s_{O,l}$, which is a state variable of the RM. More specifically, the RM sets $s_{O,l}(x_s)$ equal to the value that x'_O has when starting the iteration for x_s (A3). Another modification, called recovery process (A2) then employs $s_{O,l}$ to derive x'_O, $\check{x}'_O$, and b_O at random access (A1) at a particular position x_s.

When re-rendering starts at x_s, the first IV interval to be processed is $[x_s, x_e]$. The recovery process must consequently determine whether the right boundary $x'_e = \mathrm{f}(x_e)$ of the corresponding SV interval is occluded to derive b_O. This is done by comparing x'_e to $s_O(x_e)$ (A4 in Figure 5.1) since $s_O(x_e)$ was the minimal occluding position before x'_e was rendered. Figure 5.1 shows the two scenarios that can occur:

- When x'_e is greater than or equal to $s_O(x_e)$, x'_e is occluded by a foreground object that ends at $s_O(x_e)$. Consequently, the RM raises the flag b_O, sets x'_O equal to $s_O(x_e)$, and finally, in order to regard the correct position of the left foreground edge, $\check{x}'_O$ to $\mathrm{round}(x'_O)$ (A5).
- When x'_e is less than $s_O(x_e)$, x'_e is not occluded. The RM thus sets the flag b_O to zero, and finally x'_O and $\check{x}'_O$ to x'_e and $\mathrm{ceil}(x'_e)$, respectively, since these are the minimal fractional and integer SV position already rendered (A6).

This way, the RM can recover all three auxiliary variables at random access by only storing x'_O while rendering. Consequently, required memory bandwidth is reduced.

5.1.3 Determination of the Exact Changed Synthesized View Region

The previous section explained how the RM can start re-rendering at a random access position. The exact start and stop positions for re-rendering the whole SV region $\tilde{B}' = [x'_{C,s}, x'_{C,e}]$ to be changed, when the IV region $B = [x_{B,s}, x_{B,e}]$ changes to the depth candidate $\tilde{s}_B$ (and thus the IV depth from $s_{M,l}$ to $\tilde{s}_{M,l}$) is determined in this section. This analysis first neglects rounding to the SV integer grid.

The change of the IV depth has two effects on the SV. First, SV samples are removed from their old positions, and second, they are moved to new positions. To identify these positions, Figure 5.2 introduces two other regions. The first is the old SV region B'_S that includes the old sample positions. Its boundaries $x'_{S,s}$ and $x'_{S,e}$ are therefore given by the left- and the rightmost positions that are obtained by shifting samples of B with the unchanged IV depth $s_{M,l}$ as follows with f denoting the shifting function (2.14):

$$B'_S = [x'_{S,s}, x'_{S,e}] = \left[\min_{x \in B} \mathrm{f}(x), \max_{x \in B} \mathrm{f}(x)\right] \tag{5.1}$$

The second, the new SV region $\tilde{B}'_S$, includes the new sample positions. It is derived similar to

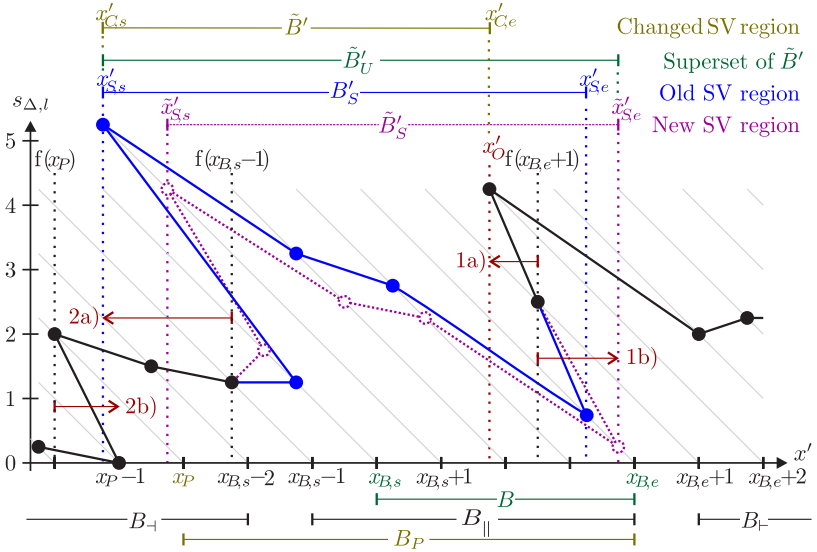


Figure 5.2: Regions in the SV texture $s'_{T,l}$ related to the depth change in B. ©IEEE

B'_S, but by shifting samples of B with the changed IV depth $\tilde{s}_{M,l}$, which is denoted by $\tilde{\mathrm{f}}$:

$$\tilde{B}'_S = [\tilde{x}'_{S,s}, \tilde{x}'_{S,e}] = \left[\min_{x\in B} \tilde{\mathrm{f}}(x), \max_{x\in B} \tilde{\mathrm{f}}(x)\right] \tag{5.2}$$

Obviously, $B'_S \cup \tilde{B}'_S$ is a subset of SV positions that may change. However, rendering of an SV interval depends on both of its boundaries. Thus, when the depth at $x_{B,s}$ and $x_{B,e}$ changes, the SV intervals related to $[x_{B,s}-1, x_{B,s}]$ and $[x_{B,e}, x_{B,e}+1]$ are affected. Regarding this additionally, the SV region that might change is $\tilde{B}'_U = [\tilde{x}'_{U,s}, \tilde{x}'_{U,e}]$ with

$$\tilde{x}'_{U,s} = \min\left(x'_{S,s}, \tilde{x}'_{S,s}, \mathrm{f}(x_{B,s}-1)\right) \tag{5.3}$$

$$\tilde{x}'_{U,e} = \max\left(x'_{S,e}, \tilde{x}'_{S,e}, \mathrm{f}(x_{B,e}+1)\right). \tag{5.4}$$

In summary, $\tilde{B}'_U$ is the region in that SV samples may depend on the IV depth change in B. However, IV samples outside B can also be warped to $\tilde{B}'_U$, as found in Section 4.2, and their SV intervals may overlap with and also affect samples in $\tilde{B}'_U$. In order to re-render $\tilde{B}'_U$, it must thus be answered a) which IV positions need to be processed to re-render SV intervals directly related to the depth change in B; and b) which IV positions need to be processed additionally to regard overlaps of their SV intervals with $\tilde{B}'_U$. The answer to a) is that processing IV positions in $B_\parallel = [x_{B,s}-1, x_{B,e}]$ is sufficient since an SV interval is rendered when its left IV boundary is processed. To answer b) and so to find where to start and stop in the IV, the following analyzes possible overlaps.

5.1.3.1 Start Position

To find the position to start, it is analyzed whether the RM needs to process any IV positions that is on the right of $B_{\parallel}$ in the region $B_{\vdash} = [x_{B,e}+1, w_B]$. This could be true when processing $B_{\parallel}$ would alter SV position that are also related to $B_{\vdash}$. When starting at $x_{B,e}$ the right boundary of its related SV interval is $\mathrm{f}(x_{B,e}+1)$. Taking this as reference position, SV intervals of $B_{\parallel}$ and $B_{\vdash}$ can only overlap when one or both of the following cases occur:

1a IV samples in $B_{\vdash}$ are shifted to the left of $\mathrm{f}(x_{B,e}+1)$

1b IV samples in $B_{\parallel}$ are shifted to the right of $\mathrm{f}(x_{B,e}+1)$

In case 1a, SV intervals of $B_{\vdash}$ (e.g. Figure 5.2 $x_{B,e}+2$) occlude SV intervals of $B_{\parallel}$ on the left of $\mathrm{f}(x_{B,e}+1)$. The occluding SV intervals of $B_{\vdash}$ are, however, not altered when processing $B_{\parallel}$ because the minimal occluding position x'_O (which is recovered at random access at $x_{B,e}$) excludes them from being re-rendered. Case 1b occurs when $x'_{S,e}$ or $\tilde{x}'_{S,e}$ (e.g. Figure 5.2 $x_{B,e}$) is greater than $\mathrm{f}(x_{B,e}+1)$. Then parts of the old or new SV region, and thus SV intervals of $B_{\parallel}$, are occluded by SV intervals of $B_{\vdash}$. However, as in case 1a, the SV intervals of $B_{\vdash}$ are preserved by using x'_O.

In conclusion, as in both cases processing $B_{\parallel}$ does not alter occluding SV intervals of $B_{\vdash}$, re-rendering can start at $x_{B,e}$.

5.1.3.2 Stop Position

After starting at $x_{B,e}$, the RM processes remaining IV positions in $B_{\parallel}$. The question is whether it needs to continue with IV positions in $B_{\dashv} = [1, x_{B,s}-2]$. This could be true when processing $B_{\parallel}$ alters SV positions that are also related to $B_{\dashv}$, which can be in case that

2a IV samples in $B_{\parallel}$ are shifted to the left of $\mathrm{f}(x_{B,s}-1)$.

Case 2a occurs, when the old or the new SV region boundary $x'_{S,s}$ or $\tilde{x}'_{S,s}$, is less than $\mathrm{f}(x_{B,s}-1)$. Then parts of the old or new SV region, and thus SV intervals of $B_{\parallel}$, occlude SV intervals of $B_{\dashv}$. This is analyzed distinguishing two sub-cases:

In the first sub-case, $x'_{S,s}$ is on the left of $\tilde{x}'_{S,s}$ (e.g. Figure 5.2 $x_{B,s}+1$). Thus, SV intervals of $B_{\dashv}$ that overlap with $[x'_{S,s}, \tilde{x}'_{S,s}]$ have been occluded previously by SV intervals of $B_{\parallel}$ and become visible. To re-render them, the RM needs to continue until the position x_P with $\mathrm{f}(x_P) < x'_{S,s}$ has been processed.

In the second sub-case, $x'_{S,s}$ is on the right of $\tilde{x}'_{S,s}$ (e.g. imagine Figure 5.2 with B'_S and $\tilde{B}'_S$ swapped). Then, SV intervals of $B_{\dashv}$ that overlap with $[\tilde{x}'_{S,s}, x'_{S,s}]$ have been visible previously and become occluded by SV intervals of $B_{\parallel}$. Thus, to re-render s'_T only, processing could be stopped at the position x_s in $B_{\parallel}$ with $\tilde{\mathrm{f}}(x_s) = \tilde{x}'_{S,s}$. However, as the SV region $[\tilde{x}'_{S,s}, x'_{S,s}]$ is occluded now, its occlusion signal $s_{O,l}$ changes and needs to be updated. Therefore, the RM needs to continue until a position x_P that is not occluded by $\tilde{B}'_S$ has been processed, which is given at the position x_P with $\mathrm{f}(x_P) < \tilde{x}'_{S,s}$.

In summary, the two sub-cases show that after processing $x_{B,e}-1$, the RM needs to continue until a position x_P with $\mathrm{f}(x_P) < \min(x'_{S,s}, \tilde{x}'_{S,s})$ has been processed. Having reached x_P (which can also be equal to $x_{B,e}-1$), a further overlap occurs when

2b IV positions in $[1,x_P-1]$ are shifted to the right of $\mathrm{f}(x_P)$

(e.g. Figure 5.2, x_P-1). However, since related SV intervals have been and are occluded, processing of positions in $[1,x_P-1]$ is not required and the RM can stop processing.

5.1.3.3 Processed Input View Region

Discussed cases show that the RM needs to process $B_{\parallel}$ first. Then it must continue in $B_{\dashv}$ until position x_P with $\mathrm{f}(x_P) < \min(x'_{S,s}, \tilde{x}'_{S,s})$ has been processed[2]. Therefore, the IV region to be processed for re-rendering $\tilde{B}'$ is

$$B_P = [\min(x_P, x_{B,s}-1), x_{B,e}]\,. \tag{5.5}$$

When processing B_P, the whole changed SV region $\tilde{B}'$ is re-rendered. Cases 1a and 1b show that $\tilde{B}'$ is a subset of $\tilde{B}'_U$ because its right side can be occluded. More specifically, with the minimal occluding position $s_O(x_{B,e})$ (as derived in Section 5.1.2), $\tilde{B}'$ is $[\tilde{x}'_{U,s}, s_O(x_{B,e})[$.

5.1.4 Re-Rendering the Changed Region

The last section specified B_P as the IV region that needs to be processed to re-render the changed region $\tilde{B}'$. How to start at $x_{B,e}$ was discussed in Section 5.1.2. The question is now how the RM can find x_P to stop re-rendering there. For this, Figure 5.1 shows further modifications to interval-wise processing.

As shown on the left, it starts with *processing* B. One of the modifications there is the derivation of the minimal changed position x'_C. In the recovery or initialization process, x'_C is set equal to w (C1). In subsequent iterations x'_C is updated (C2) to

$$x'_C = \begin{cases} \min\left(\tilde{\mathrm{f}}(x_s), x'_C\right) & \text{if } \tilde{s}_{M,l}(x_s) > s_{M,l}(x_s) \\ \min\left(\mathrm{f}(x_s), x'_C\right) & \text{otherwise} \end{cases}. \tag{5.6}$$

With (2.14), (5.1) and (5.2), it can easily be shown that the iterative derivation using (5.6) results in x'_C equal to $\min\left(x'_{S,s}, \tilde{x}'_{S,s}\right)$ after the RM has processed B (C3).

Then, the RM continues with *processing positions on the left of* B, as shown on the right in Figure 5.1. In the first iteration, the RM processes $x_{B,s}-1$. Subsequently, the RM evaluates the condition $x'_s < x'_C$ (C4). If this is true, x_s is equal to x_P, and the RM stops processing. Otherwise, it continues iterating.

[2] Although the analysis neglected rounding to the integer SV grid, it can be shown that this condition is still sufficient when rendering with quarter sample precision and rounding half sample position up.

This way, the RM can derive x'_C, which is required to find x_P and stop, directly while re-rendering. Consequently, a prior analysis of B to find x_P is not necessary.

5.1.5 Synthesized View Distortion Change Computation Step

This section discusses how the RM can use partial re-rendering for its major purpose—the quantification of the SVDC that is related to a depth candidate (indicated by the two topmost steps in Figure 3.1). As Section 4.4.2 discussed, this quantification is enabled by two modes, the *SET* and the *GET* mode. In both, a depth candidate $\tilde{s}_B$ for an IV region B is provided to the RM, either to adopt it (S4 in Section 4.4.2) or to calculate the related SVDC (G4).

To this end, the RM re-renders the changed SV region $\tilde{B}'$ (S2, G2) related to the depth candidate by iteratively processing the IV region B_P. In each iteration, the RM renders zero (e.g. f' in Figure 3.1) or more changed SV texture samples $\tilde{s}'_T(\breve{x}')$. Directly after a sample has been rendered, the RM computes its SVD $\tilde{s}'_D(\breve{x}') = [\tilde{s}'_T(\breve{x}') - s'_{T,Ref}(\breve{x}')]^2$ (G3, S3). In *SET* mode, the RM stores $\tilde{s}'_D(\breve{x}')$ as $s'_D(\breve{x}')$ (S4). In *GET* mode, the stored SVD is used to derive the SVDC (G4) according to (4.15) by incrementing D_S (which is set to zero when re-rendering starts) by

$$D_S(\breve{x}') = \tilde{s}'_D(\breve{x}') - s'_D(\breve{x}'). \tag{5.7}$$

This way, when the RM reaches position x_P and stops re-rendering, $\tilde{B}'$ has been entirely re-rendered, so that D_S is the SVDC related to the depth candidate $\tilde{s}_B$.

5.1.6 Further Optimization Options

Depending on how an encoder tests depth candidates, further optimizations are possible. In a scenario in that the encoder test several depth candidates successively for the same IV region, the RM could derive and store the left boundary of the old SV region $x'_{S,s}$ when testing the first candidate. Then, when testing further candidates, the stored value could be used to determine the changed SV region without accessing the old IV depth. This means by setting the minimal changed position x'_C to the stored value of $x'_{S,s}$ when re-rendering starts, and then to update x'_C only with the upper part of (5.6). Similarly, it would also be possible to sum up the SVD in the old SV region when processing the first depth candidate; and then to use this sum when testing further candidates, instead of using the sample wise difference (5.7). Both approaches would further decrease the number of reading memory accesses in *GET* mode.

A further option targets a scenario in that the encoder tests only one or two candidates for an IV region such that it is likely that the last one is selected. Then, it could be beneficial to perform the SVDC computation and the update of the RM jointly, i.e. a combined *SET* and *GET* operation. This can avoid a *GET* operation instantly followed by a *SET* operation for the same IV region. However, for testing multiple candidates, the option is not viable because of the higher complexity of the joint operation compared to a *GET* operation.

5.2 Complexity Evaluation

This section analyzes how the RM's complexity is related to its features (Section 5.2.1), depends on the input depth data (Section 5.2.2), affects memory requirements (Section 5.2.3), and is reduced by suggested optimizations (Section 5.2.4).

In general, the analysis neglects the complexity of the RM initialization, i.e. steps I1 to I3 in Section 4.4.2, since it is insignificant when the RM performs a huge number of *SET* and *GET* operations. This includes in particular also the interpolation of the IV texture $s_{T,l}$ to the upsampled IV texture $\hat{s}_{T,l}$.

5.2.1 Complexity of Different Renderer Model Setups and Modes

The modular RM design enables rendering with a subset of functionalities in different setups. This can avoid over-fitting or reduce complexity. Moreover, rendering can be applied in *GET* and *SET* mode. Which complexity is added by which functionality and how modes differ in complexity is thus of interest and evaluated in the following.

5.2.1.1 Methodology

The analysis counts the number of operations and memory accesses conceptually needed per IV depth sample as described in Appendix C.1. This provides a platform independent measure. Table 5.2 shows them for different setups and modes with N_T denoting the total number.

When re-rendering small IV regions B, the overhead for the recovery process (Section 5.1.2) can become significant. Furthermore, when $\tilde{s}_B$ is a large depth change, an overhead for processing positions on the left of B (Section 5.1.4) occurs. Both effects are evaluated in Section 5.2.2. To avoid them here, *GET* and *SET* operations are performed with B corresponding to the full size initial IV depth. This way, operations related to the depth candidate's characteristics are insignificant in the evaluation.

5.2.1.2 Renderer Model Setups

Table 5.2 shows results for different view extrapolation and view interpolation setups. For view interpolation, the analysis considers the case in that only one IV is changed (see Table 5.1). The following discusses the total number of operations and memory accesses that are required per processed IV depth sample, i.e. N_T, in average over *GET* and *SET* mode.

The *Base Extrapolation* setup renders the SV's luma component with integer sample precision. This requires on average 19 operations and memory accesses (i.e. 18 *GET*, 20 *SET*). Distortion computation on top of this requires on average 5 operations and memory accesses additionally (i.e. 6 *GET*, 4 *SET*). Consequently, the *Integer Extrapolation* setup—which enables SVDC computation with lowest complexity—performs 24 operations and memory accesses. When

Setup	Operations AD *GET*	AD *SET*	UA	CP	US	ML	LT	Memory accesses MR *GET*	MR *SET*	MW *GET*	MW *SET*	Σ N_T *GET*	Σ N_T *SET*	Table
Reference														
SSD	2		2	0	0	1	0	2		0		7		C.12
RM Extrapolation														
Base Extrapolation	2		3	9			1	3		0	2	18	20	C.1
+ Distortion Calculation	+3	+1				+1		+2	+1	0	+1	+6	+4	C.4
Integer Extrapolation	5	3	3	9		1	1	5	4	0	3	24		
+ Quarter Precision	+2		+1		+3		+1					+7		C.2
Fractional Extrapolation	7	5	4	9	3	1	2	5	4	0	3	31		
+ Base Chroma				+1	+1			+0.5				+2.5		C.5
+ Quarter Prec. Chroma					+1							+1		C.6
+ Dist. Calc. Chroma	+1				+0.25	+0.5		+0.5				+2.25		C.8
Full Extrapolation	8	6	4	10	5.25	1.5	2	6	5	0	3	36.75		
RM Interpolation														
Base Extrapolation	2		3	9			1	3		0	2	18	20	C.1
+ View Combination	+2.9			+3			+1	+3				+9.9		C.3
+ Distortion Calculation	+3	+1				+1		+2	+1	0	+1	+6	+4	C.4
Integer Interpolation	7.9	5.9	3	12		1	2	8	7	0	3	33.9		
+ Quarter Precision	+2		+1		+3		+1					+7		C.2
Fractional Interpolation	9.9	7.9	4	12	3	1	3	8	7	0	3	40.9		
+ Base Chroma				+1	+1			+0.5				+2.5		C.5
+ Q. Prec. Chroma					+1							+1		C.6
+ View Comb. Chroma	+0.98						+0.48	+0.48				+1.93		C.7
+ Dist. Calc. Chroma	+1				+0.25	+0.5		+0.5				+2.25		C.8
Full Interpolation	11.88	9.88	4	13	5.25	1.5	3.48	9.48	8.48	0	3	48.58		

Table 5.2: Number of operations and memory accesses per IV sample; additions (AD), unary additions (UD), comparisons (CP), unary shifts (US), multiplications (ML), table look-ups (LT), memory reads (MR) and writes (MW), and total N_T.

view interpolation is additionally enabled, as shown in the lower part in Table 5.2, 9.9 operations and memory accesses are required to combine the left SV texture $s'_{T,l}$ with the right SV texture $s'_{T,r}$. The *Integer Interpolation* setup thus requires 33.9 operations and memory accesses.

Quarter sample precision requires 7 operations and memory accesses. Consequently, the *Fractional Extrapolation* and *Fractional Interpolation* setups perform 31 and 40.9 operations and memory accesses. In contrast to the analysis of the LCR, this analysis neglects the operations and memory accesses required for the interpolation of the IV texture; as the RM performs them only once for initialization, they are insignificant for the complexity of SVDC computation.

The *Full Extrapolation* and the *Full Interpolation* setups render additionally two chroma components (with quarter resolution of the luma component) with quarter sample precision and calculate their distortions. They require 36.75 and 48.58 operations and memory accesses.

Compared to SSD computation, the total number of operations and memory accesses N_T (averaged over both modes) increases by factors of 3.4, 4.4, and 5.3 for the *Integer*, the *Fractional*, and the *Full Extrapolation* setup, respectively; and by about factors of 4.8, 5.8, and 6.9, respectively for the *Integer*, the *Fractional*, and the *Full Interpolation* setup, respectively. Considering that the RM performs full rendering, these factors are relatively low.

5.2.1.3 Renderer Model Modes

Table 5.2 shows that the *GET* mode does not write to the memory (column MR *GET*). In contrast, the *SET* mode writes to adopt the depth change and thus to update the RM's state. More specifically, it stores the changed state variables, as given in Table 5.1. When the *GET* mode renders a current interval, these variables change as well. However, their change is irrelevant for rendering further intervals, as this depends only on the old RM state, the auxiliary variables, and the RM input. Therefore, the *GET* mode does not store them and the RM state is preserved. This way, the RM can test further candidates without reverting its state. Moreover, the memory bandwidth is approximately reduced by 17% to 29% depending on the setup.

5.2.2 Dependency from the Depth Candidate

The preceding section analyzed re-rendering of a whole SV with an unchanged IV depth. However, re-rendering is usually performed for a small region B with changed depth $\tilde{s}_B$. So it is of interest how such characteristics increase the complexity. To answer this, it is analyzed how the average number of operations and memory accesses per IV depth sample increases a) for the recovery process (N_R); b) for processing B (N_B); c) for processing positions on the left of B (N_L); and d) in total ($N_T = N_B + N_L + N_R$). The evaluation uses the base extrapolation setup with quarter sample precision enabled.

To evaluate how complexity depends on the width $w_B = (x_{B,e} - x_{B,s} + 1)$ of B, the IV depth was first partitioned in regions B of size w_B. Subsequently, the RM performed a *GET* operation for each region B with a depth candidate $\tilde{s}_B$ corresponding to the depth values in B plus an

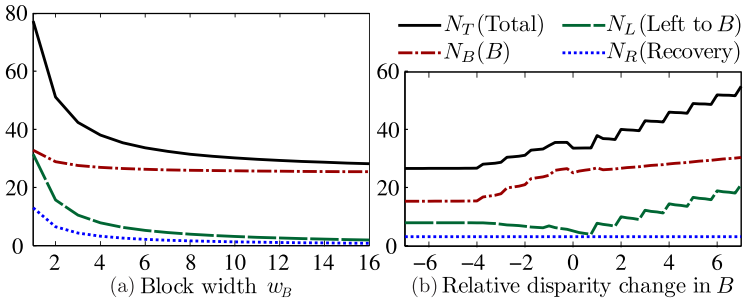


Figure 5.3: Average number of operations and memory accesses per IV depth sample depending on: (a) w_B; (b) disparity change. ©IEEE

offset leading to a disparity change of 1. Results in Figure 5.3 (a) show that when B corresponds to one IV sample only (w_B = 1), the additional complexity (N_R + N_L) introduced by the recovery process and by processing samples left of B is even larger than the complexity (N_B) required for processing B. However, N_L and N_R and thus N_T decrease fast with w_B. In conclusion, for wider IV regions (w_B >= 8) the additional complexity is insignificant.

A second aspect affecting the complexity is the magnitude of the depth change. The evaluation used the same approach as in the preceding section with two differences: w_B has been fixed to a value of 4, and the depth value offset and thus the disparity was varied. As Figure 5.3 (b) shows, N_T decreases for negative disparity changes. The reason is that the new SV region $\tilde{B}'_S$, as shown in Figure 5.2, moves to the right in $s'_{T,l}$ so that samples of the right side of $\tilde{B}'_S$ become occluded. Since occluded samples can be skipped while rendering, N_B decreases. Furthermore, as a disocclusion appears on the left of $\tilde{B}'_S$, hole filling increases N_L. When $\tilde{B}'_S$ is entirely occluded, the hole has its maximal size and N_B and N_L remain constant. For positive disparity changes N_B and N_L, and thus N_T, increase almost linearly. This is because $\tilde{B}'_S$ is shifted left in $s'_{T,l}$ and occludes left neighboring samples. As the additionally occluded positions need to be processed to update the occlusion signal $s_{O,l}$, N_L increases. Furthermore, the hole on the right of $\tilde{B}'_S$ gets larger and additional operations are required for hole filling, which increases N_B. In summary, the complexity increases significantly when testing large positive disparity changes.

5.2.3 Memory Requirements

Some application scenarios provide only limited memory resources. The RM, however, requires memory to store several signals as its state. Thus, it is of interest to quantify these requirements. For this, Table 5.1 shows the number of samples M_S that need to be stored. Shown values are calculated assuming that textures have a 4:2:0 color format and that the IV texture is upsampled by a factor of 4. The table shows that M_S—as multiple of the picture width and height product $w \cdot h$—is 5, 8, and 10.5, for the *Integer*, *Fractional*, and *Full Extrapolation* setup, respectively;

Feature	Enabled by	Factor OPs N_O	Factor MEM N_M
Partial re-rendering	Storage of state variables Recovery for random access Derivation of the changed range Interval-wise processing	64	64
Interval-wise processing	Occlusion handling Instant hole filling Instant view combination Instant distortion computation	1.3	1.8
GET mode	Interval-wise processing Preservation of state	1	1.4
Fast interpolation	Interpolation on initialization LUT based mapping	2.9	4

Table 5.3: Key optimizations of the RM and factor of the roughly estimated complexity increase without them.

and 8, 11, and 14, for the *Integer*, *Fractional*, and *Full Interpolation* setup, respectively. The major part of samples belong to the upsampled IV texture.

However, as the RM operates row-wise, it accesses only state variables of rows that belong to a currently processed IV block. Consequently, when an encoder processes the IV depth blocks in horizontal order, the required memory is reduced by a factor of h_B/h with h_B denoting the block height. In conclusion, with above optimizations and a small processing block height, the RM requires memory for less than a picture, which is provided in most application scenarios.

5.2.4 Complexity Reduction due to Key Features

As the RM design targets low complexity, it comprises several features for computational optimization. The question is if these features are effective. For this, this section compares the RM to a hypothetical unoptimized re-rendering algorithm. The comparison assumes that the unoptimized re-rendering algorithm and the RM provide the functionality of the *Integer Extrapolation* setup in *GET* mode. In this setup, the RM performs N_O = 19 operations and N_M = 5 memory accesses per IV depth sample[3]. How N_O and N_M increase when a particular RM feature is not supported by the unoptimized re-rendering algorithm is roughly estimated in the following and summarized in Table 5.3.

Partial Re-Rendering
Re-rendering is enabled by other features, which are: a) the storage of state variables as basis; b) the recovery process to start at a random access position; c) the derivation of the changed region to find out where to stop re-rendering; and d) interval-wise processing to simplify starting and stopping.

[3] See Table 5.2: 5 AD, 3 UA, 9 CP, 1 ML, and 1 LT operations; and 5 MR memory accesses.

Without these features, an unoptimized re-rendering algorithm would need to re-render a complete row upon a depth change in a small part of it because the unoptimized re-rendering algorithm would neither know whether samples at the start position are occluded, nor where to stop rendering. Assuming e.g. a row width of $w = 1024$ and a changed IV region of width $w_B = 16$, the number of processed IV samples and thus N_O and N_M would increase by a factor of $w/w_B = 64$ without partial re-rendering.

Interval-Wise Processing
Per IV depth sample, the RM performs all steps consecutively to derive the related SV samples and SVD. This is enabled by the combination and modification of basic rendering techniques, e.g. instant interpolation, hole filling, view combination, and SVD computation. A step instantly processes intermediate results of its preceding step, as depicted in Figure 3.1 by vertical arrows. Crucial for interval-wise processing is the occlusion detection method from [Berr 06]: No information of other intervals or z-buffering methods are required to handle occlusions because the minimal occluding position is tracked while rendering.

Without interval-wise processing, an unoptimized re-rendering algorithm would process all samples of the changed region consecutively in one step, before starting the next step. This way, an unoptimized re-rendering algorithm would need to store and read additional intermediate results (i.e. sample values before interpolation or hole filling). Moreover, additional operations would be necessary to iterate multiple times over the changed region.

Estimating that an unoptimized re-rendering algorithm performs three additional iterations over the changed region and that each iteration requires one addition and one comparison per sample, N_O increases by a factor of $25/19 \approx 1.3$. Assuming that two additional intermediate results are stored and read, N_M increases by a factor of $9/5 = 1.8$.

GET Mode
Interval-wise processing enables a further feature—the *GET* mode. In *GET* mode the RM can calculate the SVDC without changing its state; it thus does not perform writing accesses. Table 5.2 shows that for an unoptimized re-rendering algorithm with *SET* mode only, N_M would increase by a factor of $7/5 = 1.4$.

Fast Interpolation
For quarter sample precision, the RM maps SV positions to an upsampled IV texture, similar to backward warping [MPEG 10b], but embedded in the interval-wise processing scheme. In contrast, the VSRS 1D mode warps a complete IV texture, which has been upsampled to obtain an SV texture. This SV texture is then decimated. Assuming that an unoptimized re-rendering algorithm operates the same way in the base setup and upsamples by a factor of 4, it requires about $4 \cdot 19$ operations and $4 \cdot 5$ memory accesses (Integer Extrapolation in Table 5.2) when neglecting decimation and that SVD computation could be performed after decimation. The RM requires about 26 operations and 5 memory accesses (Fractional Extrapolation in Table 5.2) only. Therefore, for the unoptimized re-rendering algorithm N_O and N_M increase by a factor of $76/26 \approx 2.9$ and $20/5 \approx 4$, respectively.

Conclusion
As features are orthogonal, factors in Table 5.3 can be multiplied. Inverting them shows that

interval-wise processing, the GET mode, and fast interpolation reduce N_O and N_M to about $1/(1.3 \cdot 1 \cdot 2.9) \approx 26\%$ and $1/(1.8 \cdot 1.4 \cdot 4) \approx 10\%$, respectively, and in total N_T to about 23% (as N_M is about one quarter of N_T after reduction). On top of this, the most significant reduction is achieved by the main feature—partial re-rendering—as it avoids rendering of a complete row.

5.3 Chapter Summary

This chapter presented a further main contribution of this thesis—partial depth image based re-rendering. Motivation for partial depth image based re-rendering is that it enables efficient SVDC computation by re-rendering only the changed SV region, i.e. the region affected by an IV depth change. An analysis showed that re-rendering the changed SV region requires to process IV positions not only in the changed IV depth region, but also outside of it. The analysis furthermore revealed the IV positions that need to be processed and thus where to start and stop rendering in the IV.

Based on these findings, this chapter extended the LCR to the RM by the following methods:

1. Storage of intermediate results as the RM's state so that they can be used for re-rendering.
2. A recovery process to start re-rendering at a random access position.
3. Derivation of the minimal changed SV position while re-rendering and stopping there, i.e. when the entire changed SV region has been re-rendered.
4. Computation of the SVD and SVDC while re-rendering.

With these modifications, the RM needs to iterate only once over the IV region to either change its state in *SET* mode or to calculate the SVDC in *GET* mode.

Because of the modular RM design, subsets of the RM's features can be enabled selectively. This way, different RM setups allow trade-offs between rendering quality and complexity. A complexity analysis of different setups showed that compared to SSD computation, the number of required operations and memory accesses is increased by factors in the range of 3.4 (for integer extrapolation without chroma) to 6.9 (for fractional interpolation with chroma). The complexity furthermore depends on the characteristics of the depth change. It increases with small block sizes and large disparity changes.

Compared to an unoptimized re-rendering variant, the optimizations of the RM (i.e. interval-wise processing, *GET* mode, and fast interpolation) reduce its complexity to about 22%. On top of this, the most significant reduction is achieved by partial re-rendering because it avoids re-rendering the whole SV. How the RM compares to SVD estimation methods is evaluated for encoding in Section 9.3.

Part II

Synthesized View Distortion Based Encoding

6 Basic 3D Video Coding Concepts and Methods

Part I discussed the RM as method for SVDC computation. Part II evaluates the RM in the scope of 3D video coding. For this, it incorporates the RM in the rate-distortion optimization process of a 3D video encoder. As basis, this chapter discusses concepts and methods for 3D video coding considering 3D-HEVC, which is currently the most efficient standard for compression of depth-based 3D video. It provides a brief overview of HEVC and its 3D- and Multiview extensions (Section 6.1), and elaborates on how an encoder can select encoding modes while optimizing the trade-off between rate and distortion (Section 6.2). Further information on basics discussed in this chapter are given in Appendix D.

6.1 High Efficiency Video Coding

Although the available bandwidth increased in the recent years in most transmission scenarios, higher compression ratios are still required, e.g. for higher spatial and temporal resolutions, for higher sample bit depths, or for transmitting more channels at the same bandwidth. In response to these steady demands and to ensure interoperability, the ITU and ISO/IEC developed several video coding standards. H.265/HEVC [Sull 12, ITU 13, Wien 14, Sze 14]—the latest standard—was finalized in 2013 and reduces, at same quality, the bit rate about 50% [Tan 16] compared to its predecessor H.264/AVC [Wieg 03, ITU 07] from the year 2003. Beyond coding of conventional 2D video, HEVC has been extended for coding of scalable video (SHVC) [Boyc 16], screen content (SCC) [Xu 16], multiview video (MV-HEVC), and 3D video (3D-HEVC) [Tech 16]. The latter two extensions allow high efficient texture and depth compression, as it will be discussed in Section 6.1.2. Section 6.1.1 discusses the single layer HEVC coding techniques that are part of the HEVC main profile and thus a subset of both extensions.

6.1.1 Syntax and Decoding Processes

Figure 6.1 shows the basic building blocks of an HEVC encoder, bitstream, and decoder; and the data flow between them. The encoder analyzes the original input picture and selects values for HEVC syntax elements, which form a coded representation of the picture, i.e. the bitstream; the bitstream is then transmitted and interpreted by the decoder to reconstruct the picture. This section provides a brief overview of the bitstream format and the decoding processes. Further details can be found in Appendix D.1.

In HEVC, coded pictures are represented by nested syntax element structures. As shown in Figure 6.1, a picture is coded by one or more slices, which in turn consist of coding tree units (CTUs). CTUs comprise syntax elements to reconstruct picture blocks of fixed size, called coding tree blocks (CTBs). More specifically, a CTU contains a quadtree structure that subdivides the CTB in coding blocks (CBs), which are are represented by coding units (CUs) in the CTU. To decode a CB, HEVC decoders use a hybrid scheme: They first derive a prediction signal

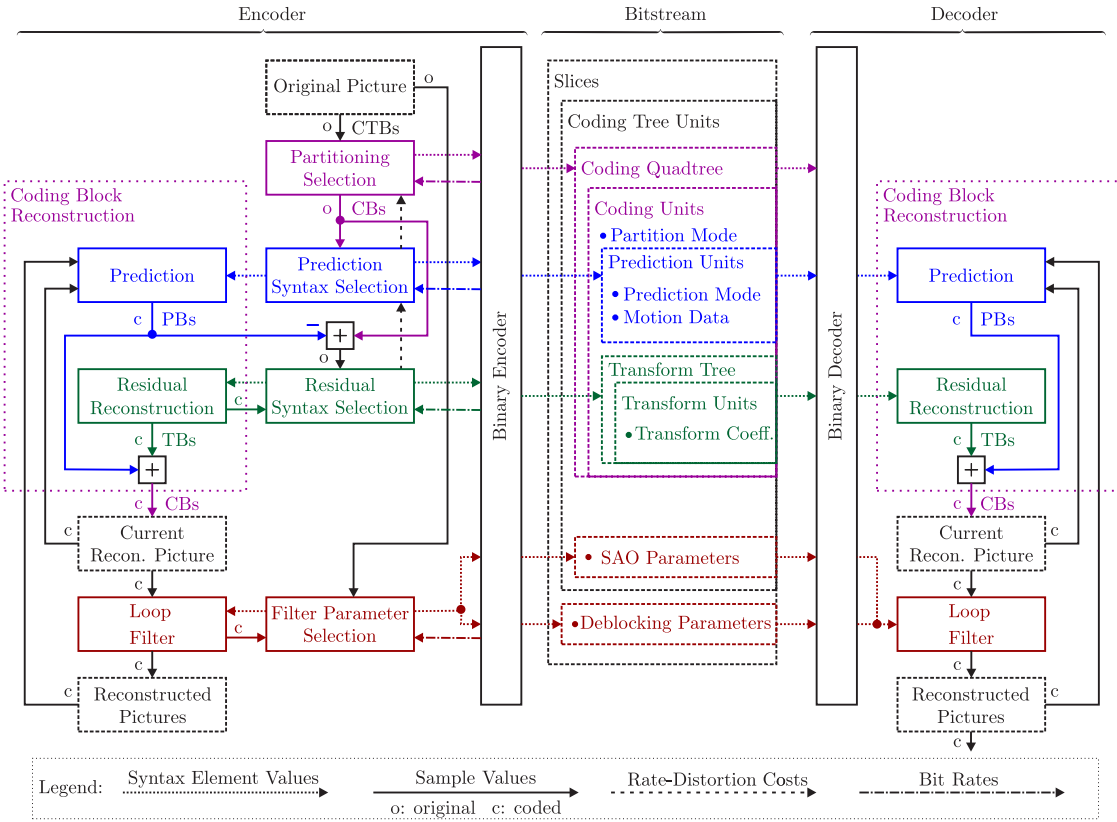


Figure 6.1: Exemplary HEVC encoder, bitstream, and decoder—Building blocks, modules, and data flow (simplified). A detailed description of the encoder is given in Appendix D.3.

for the CB from already reconstructed pictures or parts of the current picture; then, they add a residual signal, which is separately transform coded.

HEVC allows two types of prediction, intra- and inter-picture prediction. For the former, the decoder predicts the CB from adjacent parts in the current picture, and for the latter from pictures that have already been reconstructed before. A CB can either be inter- or intra-picture predicted. Nevertheless, different parts of a CB, called prediction blocks (PBs), can use different prediction parameters. For this, the CU comprises associated prediction units (PUs) that either signal different intra-picture prediction modes or, in case of inter-picture prediction, different motion data, which represent temporal and spatial offsets of reference blocks. (Prediction modes are further discussed in Appendix D.1.2.)

As the prediction conventionally differs from the original signal, HEVC allows to signal a residual. The residual is signaled for sub-partitions of the CB, called transform blocks (TBs). The partitioning of a CB in TBs is represented by a transform tree. TBs are represented by transform units (TUs), which comprise transform coefficients. To derive them, an encoder conventionally 1) applies a DCT-like integer transform to the difference between the prediction and the original signal and 2) quantizes the result, as specified by a quantization parameter (QP). Since the decoder cannot revert the quantization, residual coding is lossy. (Residual coding is further discussed in Appendix D.1.3.)

The decoder reconstructs the TBs from the TUs and adds them to the reconstructed PBs. The result are the reconstructed CBs, which the decoder inserts to the current reconstructed picture. When the decoder has reconstructed all CBs of the current picture, it applies two filters, called sample adaptive offset (SAO) and deblocking filter. (Both are discussed in Appendix D.1.4.) Furthermore, the decoder stores the final reconstructed picture in a buffer so that it can be used as reference for prediction of succeeding pictures. This way, the decoder processes all slices of the bitstream to reconstruct the input sequence.

Two further aspects of HEVC are the representation of the syntax elements in the bitstream and high level syntax (HLS). Considering the former, HEVC syntax elements are binarized and coded using context adaptive binary arithmetic coding (CABAC) [Marp 03]. Considering the latter, the bitstream comprises HLS structures, which hierarchically define parameters for parts of the picture (slice headers), one or more pictures (picture parameter set [PPS]), a sequences of pictures (sequence parameter set [SPS]), or the entire bitstream (video parameter set [VPS]). They define e.g. enabled coding tools and their parameters, the picture order in the bitstream, allowed prediction dependencies between pictures, quantization parameters for the transform coefficients, temporal layering of pictures, and random access or bitstream splicing points [Sjob 12].

6.1.2 Extensions for Multiview and 3D Video Coding

In principle, a 3D video system could encode the texture and depth layers of the 3D video signal independently using the HEVC main profile. However, such independent coding does not exploit the special characteristics of the 3D video signal. To overcome this and so to increase

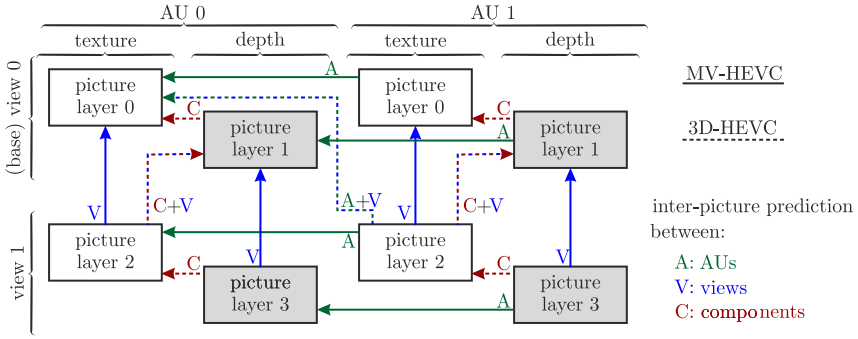


Figure 6.2: Overview of MV-HEVC and 3D-HEVC inter-picture dependencies [Tech 16].

the compression efficiency further, JCT-3V has extended HEVC by two multilayer extensions, MV- and 3D-HEVC, which are defined by the Multiview and the 3D main profile, respectively.

MV-HEVC extends only the HLS of HEVC. This means that JCT-3V has not defined new coding techniques, but has only modified the HLS such that inter-picture prediction can also refer to reference pictures of different views. More specifically, pictures of another view can be inserted to the reference picture lists of the current picture and then be used for inter-picture prediction in the same way as temporal reference pictures. This way, the similarity of different views can be exploited by inter-view prediction while reusing existing decoding cores. Figure 6.2 shows an example for such inter-view prediction. Pictures of the same time instant are contained in one access unit (AU), whereas pictures of the same view and component (i.e. texture or depth) are contained in the same layer. In addition to conventional temporal prediction within one layer, as performed in single layer HEVC coding (marked with [A]), MV-HEVC also allows inter-view prediction (marked with [V]).

Beyond the similarity of views, the 3D video signal has other special signal properties, which can be exploited by dedicated low-level coding tools. To do this, JCT-3V developed 3D-HEVC, which provides new coding tools that exploit 1) the similarity of views with enhanced methods, 2) the similarity of texture and depth, 3) the special signal characteristics of depth, and 4) that depth maps define point correspondences in different views.

To exploit similarities of different layers, some of the new coding tools use new types of inter-picture prediction, which are shown in Figure 6.2. They are inter-component prediction, i.e. prediction from texture to depth or vice versa (C), combined inter-component and inter-view prediction (C+V), and combined temporal and inter-view prediction (A+V).

Of particular relevance in this thesis are the new modes that exploit the special characteristics of the depth data (e.g. as shown in Figures E.1 and E.2). Depth maps comprise usually sharp edges in homogeneous areas. To preserve them, 3D-HEVC specifies modes that allow to split a CB in two sub block partitions, for which individual depth DCs can be predicted or signaled. The partitioning line can be straight (intra wedge mode) or predicted from the texture (intra contour mode). To represent entirely homogeneous CBs, 3D-HEVC furthermore allows the prediction

of a DC from selected neighboring samples (intra single mode), to skip transform coefficient (depth intra skip), or to signal a residual DC (depth DC offsets). Another new tool exploits the similarity of structures in texture and depth; it disallows to split a depth block further than its collocated texture block (quadtree limitation). (3D-HEVC coding tools are further discussed in Appendix D.2.)

6.2 Encoding Techniques

The encoder's objective is to select the syntax element values in a way that optimizes the trade-off between the bitstream's rate and the reconstructed picture's distortion. For this, the encoder tests different combinations of syntax element values (in the following called coding modes) that represent a current part of the input picture.

To give an example for this, Figure 6.1 shows a simplified encoding scheme[1]. The selection modules in the middle part of the encoder in Figure 6.1 iterate through different coding modes related to partitioning, prediction, residual coding, and filtering. While iterating they try to find the coding modes that provide the minimal rate-distortion cost with respect to a Lagrange multiplier approach, as it will be discussed in Section 6.2.1. To derive the rate-distortion cost of a current coding mode, they invoke 1) the binary encoder (right part of the encoder) to derive the related bit rate; 2) the reconstruction modules (left part of the encoder), which correspond to the decoder modules, to reconstruct blocks based on the tested coding mode such that the related distortion can be derived; and 3) lower-level selection modules to select dependent coding modes (e.g. TBs depending on a currently tested prediction) and to obtain their associated rate-distortion costs. When an optimal coding mode is found, the binary encoder is invoked to write the related syntax elements to the bitstream.

This way, the encoder processes all parts of a picture and all pictures of an input sequence. The resulting bitstream is, compared to the input sequence, significantly compressed.

6.2.1 Rate-Distortion Optimization based on Lagrange Multipliers

As discussed in the last section, the HTM encoder decides on how to encode a particular block by testing the RD performance of different coding modes. More specifically, to select the final coding mode m_i for a current block B_i among available coding modes m, the encoder operates as follows. It encodes block B_i with each mode m to determine the related distortion $D_i(m)$ and number of required bits $R_i(m)$. Then, it determines for each mode m the related RD cost $J_i(m)$ as

$$J_i(m) = D_i(m) + \lambda R_i(m) \tag{6.1}$$

with λ denoting the Lagrange multiplier. Finally, it selects the mode $m_i = m_i^*$, with m_i^* being the mode m that minimizes the RD cost $J_i(m)$ of the current block B_i [Shoh 88, Sull 98].

[1] A more detailed description of the exemplary encoder in Figure 6.1 is given in Appendix D.3. Furthermore, Section 7.1 provides a detailed description of the actual encoder used in this thesis.

The Lagrange multiplier approach is theoretically based on several ideas [Ever 63] that imply

1. that selecting m_i^* as final mode m_i for each block B_i is the selection that minimizes the sum of RD costs $J = \sum_i J_i(m_i)$,

2. and that minimizing J minimizes the sum of distortions $D = \sum_i D_i(m_i)$ while not exceeding a particular constraint R_C to the sum of bits $R = \sum_i R_i(m_i)$, i.e. $R \leq R_C$.

[Ever 63] shows that the constraint R_C depends on the choice of the Lagrange Multiplier λ. This means that sweeping through a range of values for λ and selecting modes for each λ based on their respective RD costs $J_i(m)$ reveals for each λ its associated constraint $R_C = \sum_i R_i(m_i^*)$ and minimal distortion $D = \sum_i D_i(m_i^*)$.

A further implication of findings in [Ever 63] concerns the relationship between two Lagrange multiplier values λ_1 and λ_2, their respective number of required bits $R(\lambda_1)$ and $R(\lambda_2)$, and their respective distortions $D(\lambda_1)$ and $D(\lambda_2)$: The quotient of the distortion difference and the difference of required bits is bound by the two multipliers, i.e. for the case that $R(\lambda_1) > R(\lambda_2)$:

$$-\lambda_1 \geq \frac{D(\lambda_1) - D(\lambda_2)}{R(\lambda_1) - R(\lambda_2)} \geq -\lambda_2 \tag{6.2}$$

This implication is in particular of importance when distortions D obtained by sweeping over a range of multipliers λ provide a differentiable function of their associated number of bits R. In this case (6.2) leads to

$$dD/dR = -\lambda \tag{6.3}$$

The Lagrange multiplier λ is thus the derivative of the function $D(R)$ obtained at the R that is associated with λ.

6.2.2 QP Selection Based on the Lagrange Multiplier

A parameter that must be determined for encoding is the QP (see Appendix D.1.3). When encoding a CB, an encoder could test different QPs to find the one that minimizes the RD cost. [Sull 98] studies this approach by a) allowing the encoder to select the QP of a current block in a QP range centered around the QP used by the preceding block; and b) analyzing the frequency distribution of the QPs that the encoder selects for a particular Lagrange multiplier. The analysis shows that for each particular Lagrange multiplier λ the related frequency distribution has a peak at a particular QP value. Consequently, [Sull 98] derives the optimal quantization step size q as function of the Lagrange multiplier λ:

$$q^2 = 12 \cdot a \cdot \lambda \tag{6.4}$$

with a denoting a constant. In addition to the experimental derivation, [Sull 98] derives (6.4) also from (6.3) by modeling the bit rate R as function of the distortion D, and the distortion D

as function of the quantization step size q, i.e.

$$R(D) = \quad a \cdot \ln[\sigma^2/D(q)], \tag{6.5}$$

$$D(q) = \quad q^2/12 \tag{6.6}$$

The QP selection of the HTM encoder is based on (6.4), which can also be expressed by substituting the quantization step size q with the QP:

$$\lambda = l_m \cdot 2^{(\mathrm{QP}-12)/3} \tag{6.7}$$

with l_m denoting an empirically found factor taking into account different picture types (i.e. intra- or inter-picture prediction) and prediction structures [Li 13].

In conclusion, (6.7) allows to select the QP based on the Lagrange multiplier without testing by the encoder. Whether such a relationship is also given for SVD-based encoding is analyzed in Section 8.5.

6.3 Chapter Summary

This chapter summarized basic 3D video concepts and methods with focus on HEVC. In HEVC, the picture is represent by coding blocks. Coding blocks are intra-picture predicted from already reconstructed coding blocks of the current picture or inter-picture predicted from other already decoded pictures. The prediction residual in represented by transform coefficients. Transform coefficient are quantized so that the picture is lossy coded. Reconstructed pictures are filtered.

Extensions of HEVC are MV-HEVC and 3D-HEVC, which target a higher compression efficiency for coding of multiple views with texture and depth data. MV-HEVC allows inter-picture prediction between pictures of different views. 3D-HEVC specifies additionally prediction between texture and depth data and introduces new coding modes exploiting the signal characteristics of the depth data. An overview of HEVC and 3D-HEVC coding modes is given in Appendix D.

To generate a bitstream, encoders select between coding modes considering their experimentally derived rate-distortion costs. For this, they employ conventionally a Lagrange multiplier approach, which minimizes the distortion with respect to a bit rate constraint, which depends on the multiplier. Without testing, the optimal quantization parameter (QP) can be derived directly from the Lagrange multiplier.

7 Encoder Integration and Initial Evaluation

Whereas the previous chapter focused on basic 3D video coding concepts and methods generally, this chapter discusses the actual encoder and framework that is used for evaluations in this thesis. It provides furthermore an initial analysis of the encoder and evaluates how encoding changes by the replacement of conventional depth distortion-based metrics with the SVDC. More specifically, Section 7.1 provides an overview of the coding mode selection process of the encoder and describes the modifications for SVDC-based encoding. Section 7.2 then defines test conditions for evaluations with this encoder and summarizes the design and configuration options that are evaluated in the following chapters of this thesis. Section 7.3 compares the RD performance and complexity of the SVDC-based encoder with conventional depth distortion-based encoding. Then, Section 7.4 discusses the reasons for the achieved coding gains by analyzing the impact of SVDC-based encoding on mode selection. Finally, Section 7.5 analyzes how SVDC-based encoding affects the depth quality. In summary, the results of this chapter provide a basis and a reference for further evaluations in this thesis.

Related works: An encoder including the RM (based on the HM software [Rose 16]) has been described in the own contributions [Tech 12c, Tech 12d]. The encoder has furthermore been part of the proposals [Schw 11a, Schw 11b] to MPEG and thus the basis for the HTM software [Chen 15, JCT 16], which has been used by JCT-3V for the development of 3D-HEVC. During the development of 3D-HEVC, additional encoding processes using the RM have been added [Chen 15]. The encoder version described in this section is based on the latest HTM version [JCT 16], but includes additional features for the evaluations in this thesis, as it e.g. enables to configure distortion metrics individually for different selection stages, to skip rate-distortion optimized quantization (RDOQ) based on the SVDC, and to use the RM also for motion estimation. [Tech 12c, Tech 12d] furthermore evaluate the RD performance of the RM in encoding. In contrast to these contributions, the evaluation in this chapter is not only based on latest HTM software, but also uses different test conditions and provides the additional analysis of mode selection and depth quality.

7.1 Integration of the Renderer Model to the HTM Software

The RM is part of the HTM encoder's processes for RD-based mode selection. The integration not only required the replacement of the conventional modules for SSD and SAD computation, but also invoking the RM in *SET* mode to align the RM with the state of the encoding process. The RM has been integrated such that it can be enabled or disabled in different selection stages, which are summarized in Table 7.1 and called *C*, *Q*, *P*, *D*, *R*, *I*, and *M* according to the related processes. This allows an exploration of different trade-offs between coding performance and complexity.

Figures 7.1 to 7.3 provide an overview of the HTM software's mode selection processes and show how they have been modified to integrate the RM. The overview is simplified; the en-

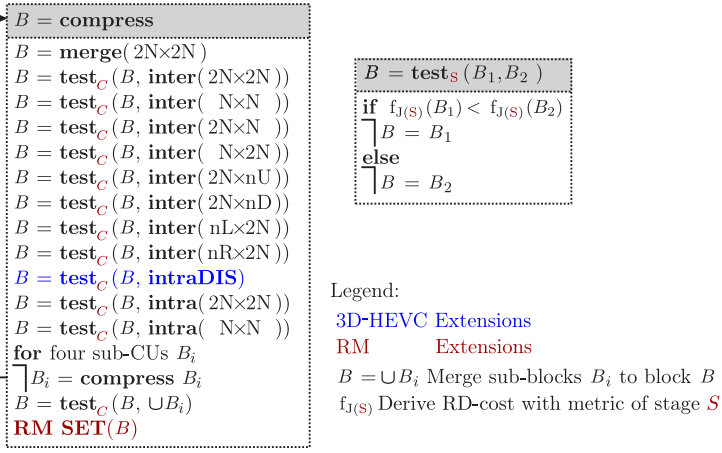


Figure 7.1: Overview of CU coding mode selection in HTM.

coder might not test all modes in all cases because some modes are e.g. not available in particular slices or skipped under particular conditions to reduce the computational complexity. A description of the coding modes is given in Appendix D.1.

Figure 7.1 shows **the CU coding mode selection process**, which belongs to selection stage *C*. For a particular CU, it first tests the merge mode for the partitioning mode $2N\times2N$. Then, it evaluates the inter-picture prediction modes for different CB partitionings, before testing the depth intra skip (DIS) mode, and the intra-picture prediction modes for different partitionings. After that, it evaluates whether to further split the block in four CBs. For this, the CU coding mode selection process invokes itself recursively. When it has determined the final mode for the current CU, it invokes the RM in *SET* mode to adopt its decision so that SVDC computation for encoding of the following CUs is based on the correct assumptions.

Figure 7.2 shows **the mode selection processes for intra-picture predicted CBs**. The **intra** process and the **intraDIS** process are invoked by the CU coding mode selection process for different partitionings; the **intra** process uses the **predIntraDepth** and the **intraRQT** process.

- The **intraDIS** process selects the best among the four prediction modes available for the depth intra skip mode (in 3D-HEVC only). It belongs to selection stage *C*.
- The **intra** process performs four steps for each PB of the current CB: 1) It pre-selects a set of several HEVC main profile intra-picture prediction modes. In doing so, it only tests the prediction without residual. 2) In 3D-HEVC, it selects parameters for the intra contour and the intra wedge mode, and adds both modes to the set of pre-selected modes. 3) It selects the best mode among the pre-selected modes considering their RD costs under the conditions that their residual is coded with a single TB, and in 3D-HEVC, skipped or coded with five different DC offsets. 4) Using the best mode, it tests whether to split the TB further.

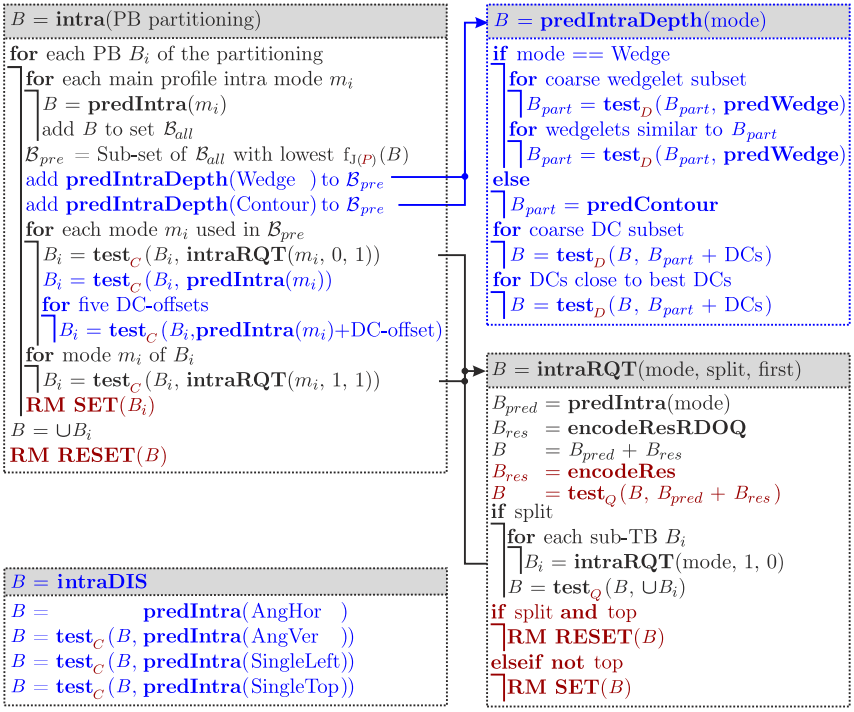


Figure 7.2: Mode selection processes for intra-picture predicted CBs.

The pre-selection in step 1 is selection stage *P*. Steps 3 and 4 belong to selection stage *C*. After the **intra** process has chosen the coding mode for a PB, it invokes the RM in *SET* mode to adopt its decision. This way, the SVDC computation for the next PB is based on the correct assumptions.

- The **predIntraDepth** process is selection stage *D* and determines parameters for the intra wedge and the intra contour mode. For the intra wedge mode, the **predIntraDepth** process selects a wedgelet and DC offsets for both wedgelet partitions using a coarse search followed by a refined search. For the intra contour mode, it only selects DC offsets, as the partitioning is predicted.

- The **intraRQT** process is selection stage *Q*. It predicts the current TB by invoking the **predIntra** process. Then, it selects between residual coding with and without rate-distortion optimized quantization (RDOQ) by invoking the **encodeResRDOQ** and **encodeRes** processes, respectively. The additional test of residual coding without RDOQ has been introduced as RDOQ is, for complexity reasons, not based on the SVDC. With the additional test, the **intraRQT** process can at least skip RDOQ when it is not beneficial. Subsequently, the **intraRQT** process tests recursively whether further splitting is better. After each invocation

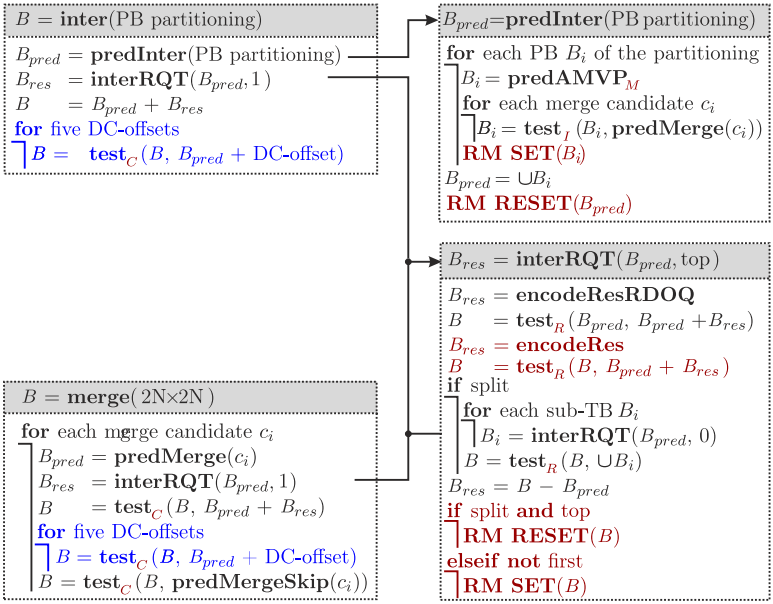


Figure 7.3: Mode selection processes for inter-picture predicted CBs.

(but not at the top level), it invokes the RM in *SET* mode so that SVDC computation for the following TBs is based on correct assumptions. At the top level, it resets the RM to uncoded data for following tests in the **intra** process.

- The **predIntra** process provides the prediction of the current intra-picture prediction mode.
- The **predWedge** process provides the partitioning of the current wedgelet.
- The **encodeResRDOQ** process transforms, quantizes, and inverse transforms the current residual. It uses rate-distortion optimized quantization (RDOQ), which considers the depth distortion in the transform domain, but, for complexity reasons, not the SVDC.
- The **encodeRes** process transforms, quantizes, and inverse transforms the current residual without using RDOQ.

Figure 7.3 shows the **mode selection processes for inter-picture predicted CBs**. They are the **inter** and the **merge** process. Both are invoked by the CU coding mode selection process and invoke the **predInter** and the **interRQT** process.

- The **merge** process belongs to selection stage *C*. It evaluates the merge candidates for CUs of size $2N\times2N$. Each merge candidate is tested while a) skipping the residual (i.e. merge skip mode), b) coding the residual, and c) with five different DC offsets (in 3D-HEVC).
- The **inter** process belongs to selection stage *C*. It first derives an inter-picture prediction for all PBs. It then encodes the residual and adds it to the prediction. In 3D-HEVC, it furthermore

Stage	Related processes	Mode Selection Setup												
		A (Intra)				E (Inter)				G (General)				
		1	2	3	4	1	2	3	4	0	1	2	3	4
C	**compress**, **intraDIS**, **intra** (without pre-selection), **inter**, **merge**	x	x	x	x	x	x	x	x	x	x	x	x	x
Q	**intraRQT**	x	x	x	x						x	x	x	x
–	**encodeRes** (intra)		x	x	x							x	x	x
P	**intra** (pre-selection)			x	x								x	x
D	**predIntraDepth**				x									x
R	**interRQT**					x	x	x	x		x	x	x	x
–	**encodeRes** (inter)						x	x	x			x	x	x
I	**predInter**							x	x				x	x
M	**predAMVP**								x					x

Table 7.1: Mode selection stages using the same distortion metric and mode selection setups.

tests whether it is better to use DC offsets instead of the transformed residual. In doing so, it evaluates five DC offsets.

- The **predInter** process is selection stage *I*. It derives an inter-picture prediction for each PB of the current CB. It first derives a prediction using the AMVP mode by invoking the **predAMVP** process. Then, it tests all merge candidates. For all modes, it tests only the prediction without residual. After having selected the prediction mode for a PB, it invokes the RM in *SET* mode to adopt the decision. After having processed all PBs, it resets the RM.
- The **interRQT** process is selection stage *R*. It decides how to encode the residual and operates similar to the **intraRQT** process. However, differences are that it a) does not derive the prediction signal as the prediction signal is, in contrast to the intra-picture prediction modes, decoupled from the TBs; and b) tests whether to skip the TB's residual.
- The **predMerge** process derives the prediction for the current merge candidate.
- The **predAMVP** is selection stage *M*. It derives a prediction using AMVP. In doing so, it performs motion estimation in one or more pictures.

In summary, the encoder invokes the RM in *GET* mode to derive the SVDC in various decision stages. When it evaluates sub-partitions of a block, it furthermore invokes the RM in *SET* mode so that SVDC computation for following partitions is based on the correct assumptions. After the encoder has coded all sub-partitions, it resets the RM to uncoded data for further evaluation of the block. It should be noted that Figures 7.1 to 7.3 show only in a simplified manner where *SET* operations are performed; the actual software includes further optimizations, as e.g. not carrying out *SET* operation for a last partition or skipping a reset when the encoder selects the last tested coding mode. In conclusion, the modified encoder can use the RM for SVDC computation in all mode selection stages, except in RDOQ, which can, however, be disabled based on the SVDC. Furthermore, different selection stages can use different distortion metrics, which is summarized in Table 7.1. This allows trade-offs between complexity and coding performance, which will be discussed in Section 8.4.

7.2 Test Conditions

This section defines the test conditions generally used by this thesis' evaluations and discusses configuration options that are additionally explored. To represent a realistic scenario, test conditions are based on the common test conditions provided by JCT-3V [Mull 14]. However, they have been modified to focus on the main aspect in this thesis—depth coding using the SVDC.

In particular, when comparing the RD performance of two methods, evaluations consider the SVD and the depth bit rate as follows: First, the depth bit rate and the SV texture PSNR are experimentally determined for each method using several depth QPs. In doing so, the texture QP is constant and texture coding operates from depth coding independently. Consequently, IV textures are identical. Then, the average depth bit rate difference (in the following called depth bit rate delta R_{Δ}) is obtained as described in Appendix E.2.1. This means that the depth bit rate is interpolated as function of the SV PSNR for both methods and that depth bit rate differences are averaged over a certain SV texture PSNR range. In summary, this approach has two advantages. Since IV textures are identical in all evaluations, the RD performance is not affected by indirect changes due to shifts of the bit rate ratio between texture and depth. Considering only depth bit rates avoids a bias by the texture bit rate. Nevertheless, as the RD performance considering texture coding is also of interest, it is finally evaluated in Section 9.6.

The following describes the test conditions more specifically. Table 7.2 classifies them in four categories concerning the test set, the encoder setup, the VSO setup, and the evaluation setup.

Testset
The test set consists of eight sequences provided by JCT-3V (see Appendix E.3). Five of them depict natural scenes and have estimated depth maps. Although suitable for rendering with acceptable quality, the depth maps can be regarded as noisy. Furthermore, these sequences contain other distortions of the acquisition step as summarized in Table 2.2. Three of the sequences are computer generated content. Depth and texture of these sequences can thus be regarded as distortion free. For simplification, only two views are used in evaluations in this thesis as results can generally be extrapolated to a larger number of views.

Encoder Configuration
The temporal and inter-view prediction structures are the same as in the JCT-3V test conditions: Temporal prediction uses a hierarchical IBP picture structure, a GOP size of 8, and a period of intra-picture predicted pictures of 24; inter-view prediction from the first to the second coded view is enabled for texture and depth. Inter-component prediction from texture to depth is applied. As this thesis targets depth coding only, the prediction from depth to texture is disabled (i.e. VSP, DBBP, and depth refinement; see Appendix D.2.2) so that texture coding is independent from depth coding. The texture QP is set to a constant value of 30. The depth QP is varied from 21 to 51 with a step size of 6. Evaluations use in general the 3D Main profile of HEVC since it is the most advanced coding method for 3D video. As in JCT-3V's common test conditions, sample adaptive offset (SAO) and the deblocking filter are disabled because these tools are not beneficial for depth coding [Mull 14]. Quadtree limitation as described in Appendix D.2.1 is enabled.

Configuration	*Default*	*Other Evaluated Options*
	Testset	
Sequences	JCT-3V sequences	–
Input views	Two at double stereo dist.	–
	Encoder Setup	
Syntax	3D-HEVC	MV-HEVC
Intra-period	24	∞
Temporal prediction	Hierarchical IBP GOP 8	IPP
Inter-View predicition	IP	–
Inter-Comp. prediction	Texture to depth	& Depth to texture
Texture QP	30	Variable
Depth QP	51, 45, 39, 33, 27, 21	–
SAO	Off	–
QT Limitation	On	–
	VSO Setup	
Mode selection setup	G0	G1–G4, A1–A4, E1–E4
RM setup	Full interpolation	Integer pel, SVDC off, View comb. off
SV generation setup	View interpolation	Extrap.; Interp./Extrap. combinations
Reference SV	Synth. from uncoded	Coded/uncoded combinations
Tested SV	Synth. from available coded	Uncoded
SV position setup	One SV at stereo distance	Multiple SV
Depth distortion term	Off	Various weights
Lagrange multiplier	Default	Optimized selection
VSO early skip	Off	–
	Evaluation Setup	
Renderer	LCR	1D-Fast, VSRS General, VSRS 1D
SV generation method	View interpolation	–
Reference SV	Synth. from uncoded	–
Tested SV	Synth. from coded	–
SV position setup	One at stereo distance	Multiple SVs
Distortion metric	PSNR	SSIM, Pratt's FOM, PSNR
RD perform. measure	Depth bit rate delta	Depth/SV PSNR delta
Complexity measure	Depth encoding time delta	Numer of operations

Table 7.2: Test conditions used in this thesis.

VSO Setup

The View Synthesis Optimization (VSO) setup is the subset of the encoder setup that is directly related to SVDC-based encoding. It comprises several other setups:

One is the mode selection setup as defined in Table 7.1. By default, mode selection setup G0 is used, i.e. VSO is only applied in the highest selection stage C. Effects of SVDC-based encoding in other mode selection setups are then evaluated in Section 8.4.

The RM setup defines the functionalities of the RM as described in Section 5.2.1. Evaluations use in general the *Full Interpolation* setup, i.e. quarter-sample precise warping including

chroma components is enabled. Other RM setups are then evaluated in Section 8.1.

The SV generation setup defines how the RM renders the tested and the reference SV texture, which is by default view interpolation. Other options, as e.g. view extrapolation or combination of view extrapolation and interpolation, are evaluated in Section 8.2. For SVDC computation, the RM renders the reference SV texture using uncoded texture and depth data, and the tested SV texture from coded data when already available and uncoded data otherwise. Other options are rendering from different combinations of coded and uncoded texture and depth data. These options are also further evaluated in Section 8.2.

The SV position setup defines the number and the positions of SVs. By default, only one SV in between the two IVs, i.e. at stereo distance, is rendered. Configurations with multiple views at different positions are evaluated in Section 9.1.

Two other parameters that affect SVDC-based encoding are a depth distortion term and the Lagrange multiplier used in rate-distortion optimization. The depth distortion term was suggested in [Jung 12c] in addition to the SVDC. By default, only the SVDC is used. Encoding with different depth distortion weights is analyzed in Section 8.3. Regarding the Lagrange multiplier, the default values of the HTM-software are used [Chen 15]. An evaluation that analyzes optimal combinations of Lagrange multipliers for depth and SVDC-based mode selection and related QPs is provided in Section 8.5. VSO early skip, as described in Section 2.2.3.3, is disabled.

Evaluation Setup
The evaluation setup defines how the encoding performance is measured. By default, the LCR is used as receiver-side renderer. The encoder and the receiver thus use the same rendering algorithm. An evaluation considering other rendering algorithms is then presented in Section 9.2. Furthermore, only one SV at stereo distance to the IV is regarded in general, whereas Section 9.1 discusses setups with multiple SVs. SVs are generated using view interpolation. The reference SV texture is rendered from uncoded IV texture and depth; the tested SV texture is rendered from coded IV texture and depth.

The default distortion metric is the PSNR of the tested SV texture with respect to the reference SV texture. Other metrics employed in Section 9.4 are the Structural Similarity (SSIM) Index and Pratt's Figure of Merit (FOM) (see Appendix E). The RD performance is measured as described in Appendix E.2: According to the Bjøntegaard Delta method [Bjon 01], depth bit rate differences are averaged over certain SV texture PSNR ranges to obtain the depth bit rate delta R_{Δ}. SV texture PSNRs ranges approximate the ranges used by JCT-3V's test conditions. They thus represents realistic conditions. Beyond the depth bit rate delta, some evaluations consider also the average SV PSNR or depth PSNR differences, called SV PSNR delta $D_{\Delta P}$ and depth PSNR delta $D_{\Delta P,M}$, respectively.

The encoding complexity is estimated by the average depth encoding time difference, which is called depth encoding time delta T_{Δ} and derived as presented in Appendix E.2.4. Furthermore, an analysis of the number of operations is performed in Section 9.3.

	3D-HEVC; SVDC vs. DD			MV-HEVC; SVDC vs. DD		
	$R_\Delta[\%]$	$T_\Delta[\%]$	$D_{\Delta P,M}$ [dB]	$R_\Delta[\%]$	$T_\Delta[\%]$	$D_{\Delta P,M}$ [dB]
Balloons	−77.5	+ 79	− 8.7	−37.6	+11	− 9.0
Kendo	−82.1	+ 58	−11.3	−44.2	+12	−13.4
Newsp.	−79.8	+ 86	−11.6	−53.5	+15	− 9.7
P.Hall2	−65.5	+105	−12.9	−51.6	+11	− 8.4
P.Street	−72.9	+103	− 7.7	−50.4	+12	− 5.0
GTFly	−75.4	+101	−29.7	−55.1	+16	−11.0
Shark	−82.9	+ 85	−14.9	−46.1	+12	−11.0
UndoD.	−63.3	+ 89	−20.0	−59.4	+12	−13.8
Mean	−74.9	+ 88	−14.6	−49.7	+13	−10.2

Table 7.3: Depth bit rate R_Δ, depth encoding time T_Δ, and depth PSNR $D_{\Delta P,M}$ deltas; Tested: SVDC-based encoding; Reference: Depth distortion (DD)-based encoding using the respective HEVC extension.

7.3 RD Performance and Encoding Complexity

This section compares depth distortion- and SVDC-based encoding under the default test conditions (see Section 7.2), which define the *initial encoder setup*. Figure 7.4 shows how the PSNR of the SV texture changes with the depth bit rate. In both shown scenarios—MV-HEVC and 3D-HEVC—the RM improves the RD performance significantly. For high rates, SVDC-based encoding using MV-HEVC even outperforms depth distortion-based encoding using 3D-HEVC. Table 7.3 shows that SVDC-based encoding reduces the depth bit rate about 75% and 50% in the 3D-HEVC and the MV-HEVC scenarios, respectively. Depending on the sequence, reductions vary from 63% to 83% for 3D-HEVC and 37% to 59% for MV-HEVC. Reductions for natural and synthetic sequences do not differ significantly. In summary, SVDC-based encoding increases the RD performance by a magnitude that roughly corresponds to what 3D-HEVC achieves in comparison with MV-HEVC, as Figure 7.4 shows.

To analyze the improved RD performance, Figures 7.5 (a) and (b) show how the depth bit rate and SV texture's PSNR differences depend on the QP. Results are averaged over all sequences. The depth bit rate decreases for all QPs when SVDC-based encoding is enabled. Relative reductions, however, decrease with increasing QP. On the other hand, the SV PSNR difference shows gains for low QPs, which decrease with increasing QPs so that PSNR losses occur at high QPs. The improved RD performance is thus caused by a decrease of the depth bit rate and an increase of the SV texture PSNR at low QPs and a decrease of the depth bit rate at high QPs.

Rendering for SVDC computation increases the number of required computational operations. To analyze this, Table 7.3 shows how the depth encoding time changes for SVDC-based encoding in comparison to depth distortion-based encoding. On average over sequences, it increases in the 3D- and MV-HEVC scenario by about 88% and 13%, respectively. The reason for this difference is that, relative to the total number of tested modes, the percentage of modes that are tested with the SVDC (in selection stage C) is higher in the 3D-HEVC scenario than in

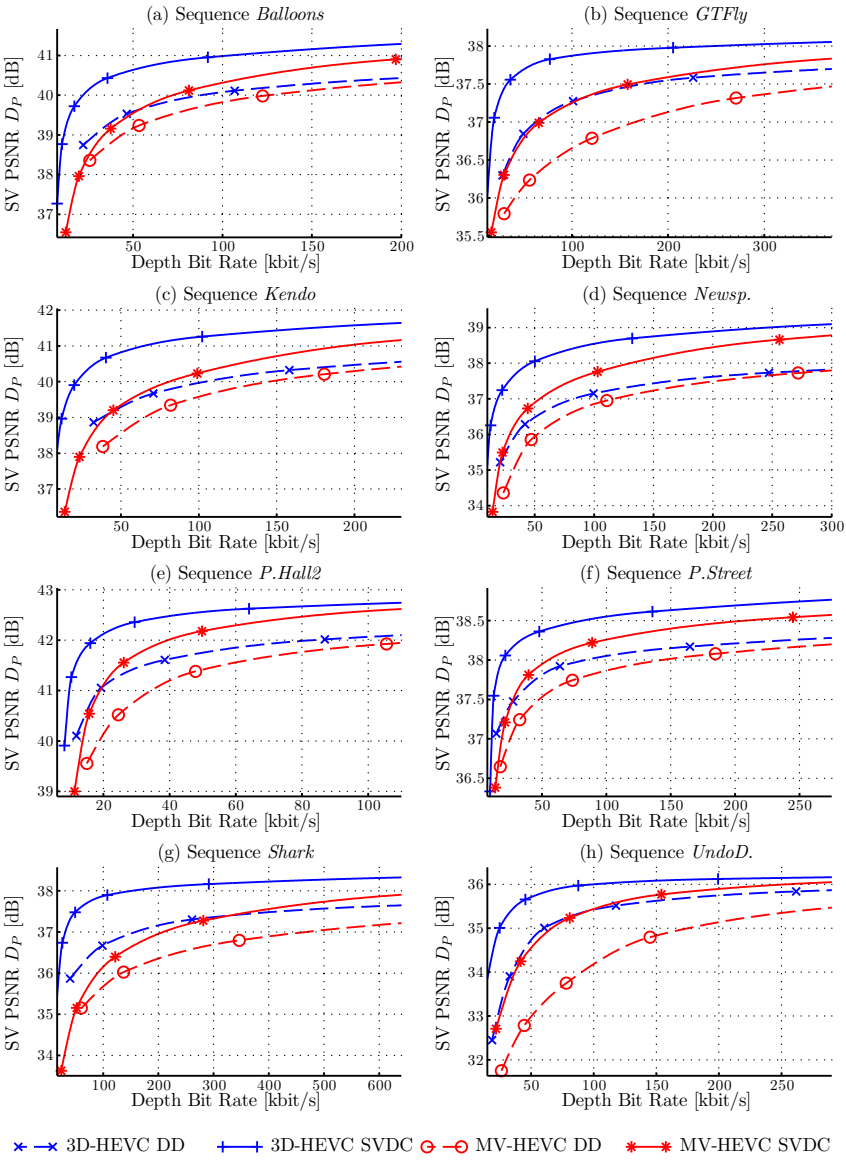


Figure 7.4: RD performance of depth distortion (DD)- and SVDC-based encoding; MV-HEVC and 3D-HEVC.

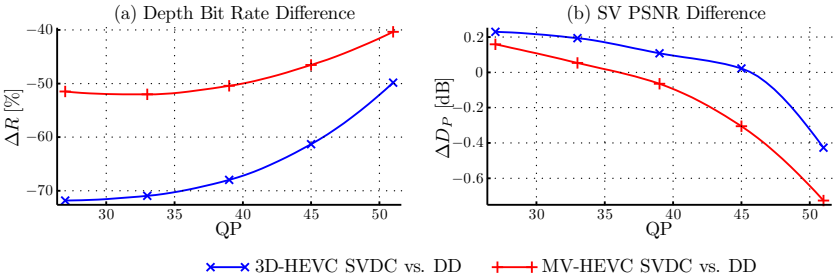


Figure 7.5: Difference of the SV texture PSNR and depth bit rate depending on the QP when using SVDC-based instead of depth distortion-based encoding.

the MV-HEVC scenario. More specifically, in the 3D-HEVC scenario, selection stage C tests additional modes (e.g. depth intra skip and depth DCs, see Section 7.1), whereas the lower-level stages test fewer modes (e.g. because of full sample motion accuracy and early skipping). Furthermore, the total number of tested modes is in the 3D-HEVC scenario lower than in the MV-HEVC scenario since quadtree limitation (see Appendix D.2.1) disallows to split the depth CB more than the texture CB.

A clearer conclusion can be drawn when comparing depth encoding times in percent of the MV-HEVC scenario's texture encoding time (see Table F.1): In the MV-HEVC scenario, the depth encoding time increases from 93% to 105%; in the 3D-HEVC scenario, it increases from 20% to 38%. Thus, although the complexity increases, numbers show that full rendering is feasible in real applications, in particular as SVDC computation can furthermore be parallelized.

7.4 Effects of SVDC-Based Encoding on Mode Selection

The last section showed that SVDC-based encoding improves the RD performance significantly compared to depth distortion-based encoding. The obvious reason is that the encoder allocates more bits to depth map regions that are relevant for the quality of the SV texture, whereas it encodes other regions with lower quality. This raises the questions if and how this changes the set of selected modes and the bit allocation to them and why this improves the RD performance.

To answer them, Figures 7.6 to 7.8 show various mode selection statistics for the four test cases, i.e. MV- and 3D-HEVC using the depth distortion and the SVDC. Results are averaged over the sequences of the test set and shown in dependence of a normalized depth bit rate. This means that before averaging results, depth bit rates have been normalized such that for each sequence 0% corresponds to the maximum of the four depth bit rates that are obtained when coding the sequence in the four test cases with the highest QP, and, accordingly, that 100% corresponds to the minimum of the four depth bit rates with the lowest QP.

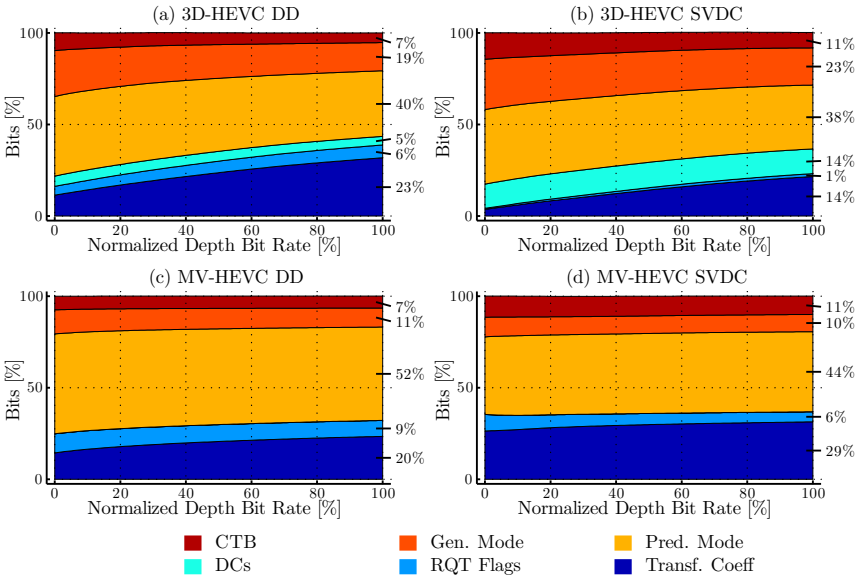


Figure 7.6: Percentage of depth bits allocated to different syntax element categories. Percentages on the right are averages over all bit rates.

7.4.1 Bit Rate Allocation

As discussed in Section 6.2, the encoder decides how to partition the picture in CBs, chooses prediction and residual coding modes for the CBs, and finally selects the respective parameters for prediction and for residual representation. The question how the allocation of bits to syntax elements in these basic categories changes is thus of interest to understand the differences of depth distortion- and SVDC-based encoding. It is answered by Figure 7.6 for 3D-HEVC and MV-HEVC. More specifically, Figure 7.6 shows the percentages of the total number of bits of the coded depth map that are allocated for the following categories:

- **CTB**: Coding Tree Block partitioning, i.e. splitting flags of the CTB.
- **Gen. Mode**: Coding mode signalization in general, i.e. syntax elements that signal the skip mode, intra-picture skip mode, whether inter-picture prediction or intra-picture prediction is used, and the presence of the residual quadtree or depth DCs.
- **Pred. Mode**: Parameters related to the selected prediction mode, i.e. the actual intra-picture prediction mode, AMVP or merge mode, the merge candidate, motion information, and the PB partitioning
- **DCs, RQT Flags, Transf. Coeff**: Residual representation, i.e. syntax elements representing the depth DCs (in 3D-HEVC), the partitioning of the residual quadtree (RQT), and the

presence of transform coefficients in the RQT, and transform coefficient, respectively.

Figure 7.6 shows that the percentages of bits in these categories vary monotonically in all four test cases: With decreasing depth bit rate, the percentage of bits spend for residual coding decreases and percentages of other categories increase accordingly.

Figures 7.6 (c) and (d) show results for the MV-HEVC test case. In depth distortion-based encoding the encoder allocates on average 7%, 11%, 52%, and 29% of all bits for CTB partitioning, general mode indication, prediction parameters, and residual representation, respectively. For SVDC-based encoding percentages change to 11%, 10%, 44%, and 35%, respectively. The encoder thus allocates more bits for partitioning of the picture and especially for residual coding at expense of bits it allocates for general mode signalling and prediction parameter signalling.

Results for the 3D-HEVC test case are shown in Figures 7.6 (a) and (b). In depth distortion-based encoding the encoder allocates 7%, 19%, 40%, and 34% of all bits, for CTB partitioning, general mode signalling, prediction parameter signalling, and residual representation, respectively. When SVDC-based encoding is enabled, percentages change to 11%, 23%, 38%, and 29%, respectively. Thus, the encoder allocates in particular more bits for CTB partitioning, while the number of bits spend for residual coding decreases.

In summary, the encoder achieves a higher SV quality at equal rate by a finer partitioning of the picture's CTBs. In doing so, it spends less bits for prediction parameters; for residual coding, it spends less bits (in the 3D-HEVC test case) or more bits (in the MV-HEVC test case). How these bit allocation shifts are actually related to the selection of particular encoding modes and how they lead to an SV quality gain is analyzed in the following two sections.

7.4.2 Residual Coding Modes

To analyze the mode selection related to residual coding, each depth sample of the coded sequence has been classified in one of six categories as shown in Figures 7.7 (a) and (b). Categories relevant in the MV-HEVC test case are and comprise samples of

- **Skip**: CBs coded in skip mode, i.e. not associated with any coded residual;
- **RQT 0**: CBs not coded in skip mode, but without transform tree;
- **RQT Cbf0**: CBs with transform tree that are at positions without transform coefficients;
- **RQT Cbf1**: TBs, i.e. associated with transform coefficients.

Figure 7.7 (c) shows the relative occurrence of samples in the four categories in depth distortion-based encoding. Averaged over the normalized bit rates, about 86%, 5%, and 6% of samples belong to categories Skip, RQT 0, and RQT Cbf0, respectively, and are thus coded without residual. Only 3% of samples are associated with transform coefficients. Reason for this low percentage compared to texture coding is that a large part of depth maps consist of homogeneous areas, which can be predicted with high accuracy, in general.

Figure 7.7 (d) shows how percentages change in SVDC-based encoding: The occurrence of samples coded in skip mode increases to about 96% at cost of a decreasing number of samples

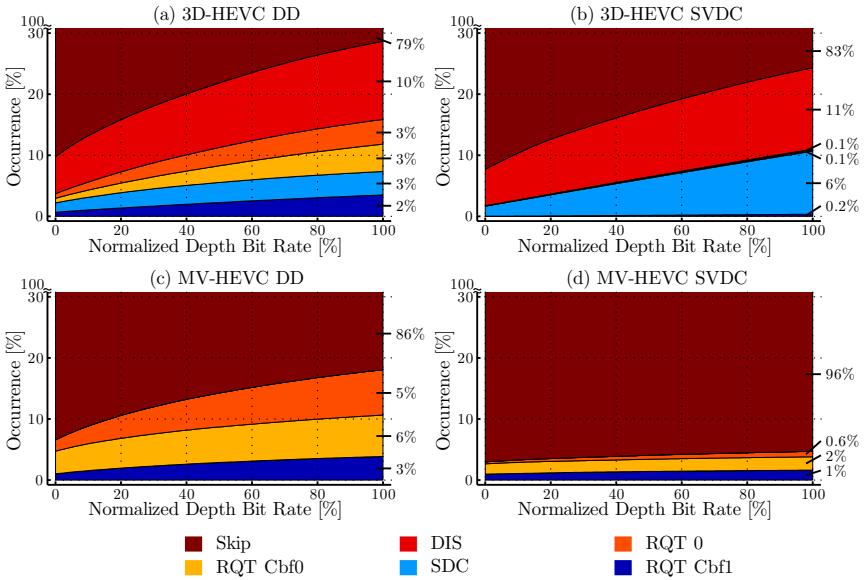


Figure 7.7: Areas show the relative occurrence of samples coded in different residual coding modes. The area from 30% to 100% belongs to the Skip Mode.

in the other categories. In particular, the percentage of samples associated with transform coefficients decreases to about 1% at all depth bit rates. This indicates that residuals transmitted in depth distortion-based encoding are in many cases not relevant for the SVD. Despite of that the occurrence of samples associated with transform coefficients drops from 3% to 1%, the number of bits allocated for transform coefficients increases from 20% to 29%, as Figures 7.6 (c) and (d) show. This means that the encoder achieves the SV quality gain at the same depth bit rate by signalling the residual for fewer blocks, but with higher quality.

Figures 7.7 (a) and (b) show results for the 3D-HEVC test cases. Compared to the MV-HEVC test cases, the number of samples in categories Skip, RQT 0, and RQT Cbf0 decreases. Reasons for this are the additionally available 3D-HEVC modes. Samples of blocks coded in these modes are classified in two additional categories.

- **DIS**: Samples of CBs coded in depth intra skip mode, which are not associated with any coded residual data (as described in Appendix D.2.1).
- **Depth DC**: Samples of CBs using the DC-only mode, i.e. CBs not associated with transform coefficients, but signalling DC values (see Appendix D.2.1).

Figure 7.7 (a) shows that in depth distortion-based encoding the number of samples in categories Skip, RQT 0, and RQT Cbf0 is about 79%, 3%, and 3%, respectively. Additionally, the DIS mode outperforms the other modes in 10% of all cases in that its efficient signalization in

combination with intra-picture prediction without residual is beneficial. In summary, 95% of all samples are not associated with residual data. The DC mode is selected for 3% of all samples for that the special signalling of a single DC value outperforms residual skipping or signalling of transform coefficients. Only 2% of all samples are associated with transform coefficients.

Figure 7.7 (b) shows that in SVDC-based encoding the percentage of samples not associated with residual data is with 94% about the same as in depth distortion-based encoding. Nevertheless, compared to depth distortion-based encoding, samples not associated with residual data are more often coded in skip or depth intra skip modes, whereas the number of samples in categories RQT 0, and RQT Cbf 0 decreases. A possible explanation for this (which will be further discussed in Section 7.4.3) is that the skip and the depth intra skip modes additionally allow an efficient signalization of the prediction parameters.

Regarding samples associated with residual data, it can be observed that in SVDC-based encoding the percentage of samples associated with transform coefficients decreases from 2% to 0.2%. Additionally, Figures 7.6 (a) and (b) show that the percentages of bits allocated for transform coefficients and for depth DCs decrease only from 23% to 14%. This indicates, as well as the MV-HEVC test case, that signalling the transform coefficients for fewer blocks with higher quality is beneficial for the SV quality. On the other hand, the percentages of samples associated with DC values increase from 3% to 6% while the percentage of bits allocated to code them increases from 5% to 14%. Thus, in SVDC-based encoding the alternative of signalling accurate DC values becomes more important for an increased number of depth blocks. In the MV-HEVC test case, the lack of this alternative is an explanation for the increased number of bits allocated for transform coefficients in SVDC-based encoding.

7.4.3 Prediction Modes

This subsection analyzes how the selection of prediction modes changes in SVDC-based encoding. Results for the MV-HEVC test case are shown in Figures 7.8 (e) to (h). Figures 7.8 (e) and (g) show that in depth distortion-based encoding about 38% of bits are allocated for syntax elements related to inter-picture prediction (i.e. to Merge and AMVP), which is used for 93% of all samples. In SVDC-based encoding, percentages change to 30% and 97%, respectively. More specifically, the occurrence of samples coded in skip mode increases from 86% to 96% and the percentage of bits allocated for their prediction parameters from 12% to 17%. In contrast, the occurrence of samples coded using AMVP decreases from 5% to 0.7%, whereas percentages of bits allocated for their motion information decreases only from 23% to 12%. Motion information in AMVP mode is thus signaled for a fewer number of blocks, but with higher accuracy, which is a similar effect as found for the transform coefficients. This effect occurs also for intra-picture prediction parameters: While the percentage of samples coded using intra-picture prediction decreases from 7% to 2.7%, bits allocated for their prediction parameters stay with 12% the same.

Results for the 3D-HEVC test case are shown in Figures 7.8 (a) to (d). In addition to the MV-HEVC prediction modes, two new modes are available—the depth intra skip mode and the intra

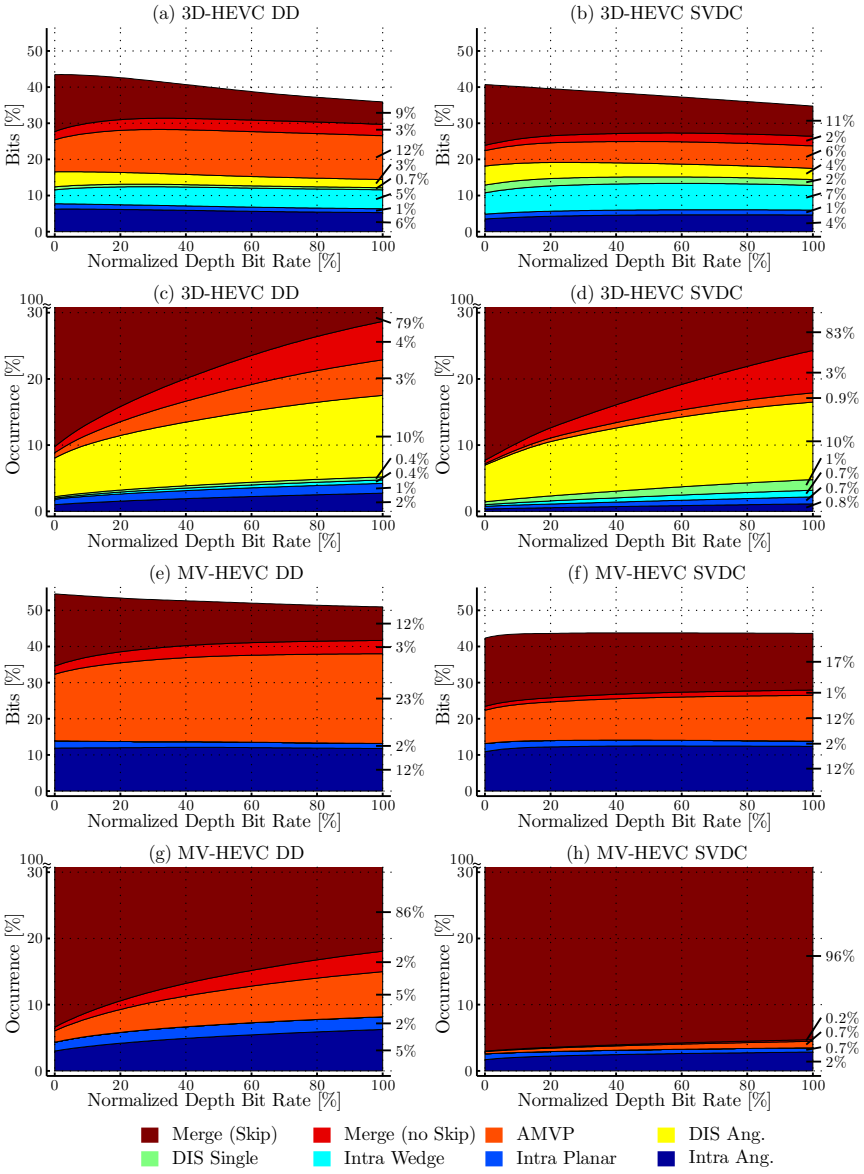


Figure 7.8: Percentage of depth bits for parameters of different prediction modes (a), (b), (e), and (f); and relative occurrence of samples coded in these modes (c), (d), (g), and (h).

wedge mode. As the depth intra skip mode is a viable alternative to the conventional inter-picture skip mode, the number of samples using inter-picture prediction decreases compared to MV-HEVC. More specifically, in depth distortion-based encoding 86% of all samples are coded using inter-picture prediction, which requires about 24% of all bits. When the RM is enabled, percentages change to 87% and 19%, respectively. Similar to the MV-HEVC test case, changes are primarily caused by changes for the AMVP mode, as its occurrence and bits allocated for its motion information decrease from 3% to 0.9% and 12% to 6%, respectively. Thus, again less blocks are coded in AMVP mode, while the bits spend per block for signalling motion parameters increase. Regarding the intra-picture prediction modes, it can be observed that percentages change only slightly, when the RM is enabled. In particular, the percentage of bits allocated for the intra wedge mode and the depth intra skip mode (i.e. DIS Ang. and Single) increases from 5% to 7% and 3.7% to 6%, respectively, while bits spend for the conventional intra modes (i.e. intra planar and ang.) decrease from 7% to 5%. Percentages of samples using these modes change similarly, from about 0.4% to 0.7%, 10% to 11%, and 3% to 1.5%. Thus, 3D-HEVC modes explicitly designed for SVD-based encoding are, as expected, favored over the conventional angular modes.

7.4.4 Conclusion

The analysis shows that SVDC-based encoding affects which modes are selected by the encoder. More specifically, the encoder selects prediction and residual representation modes that are expensive in terms of bits required for signalization, as e.g. AMVP, conventional intra modes, and transform coefficients less often. However, when they are selected, the encoder allocates more bits for them. On the other hand, the encoder selects modes that are inexpensive in terms of bits more often, as e.g. the skip modes or the signalization of depth DCs. In summary, bits are allocated more selectively, which also leads to a finer partitioning of the picture in CBs, as the increased number of bits for CTB partitioning shows.

These observations are in line with common assumptions on the importance of the depth map quality for the SV quality, i.e. that the depth quality is 1) rather irrelevant in the majority of depth regions that are collocated to low-frequent texture regions and 2) important in the minor part of the depth map that is collocated to texture regions with high variance as e.g. edges.

7.5 Effects on Depth Quality

SVDC-based encoding preserves the depth map quality only in parts that are required to maintain the SV quality. As discussed in the last section, this primarily means that motion information and in particular residual data is only signaled for a fraction of the depth map. How this affects the depth quality is shown in Table 7.3 (and Figure F.26). It can be observed that the depth PSNR decreases in average over sequences about 10 dB for MV-HEVC and 15 dB for 3D-HEVC. Reason for the differences is that the additional 3D-HEVC tools provide more

possibilities for the encoder to decrease the SVDC at expense of the depth distortion, e.g. using the DIS mode, or signalling only the residual DCs.

Rate savings achieved by SVDC-based encoding come thus at the expense of a highly reduced depth map quality, so the quality of the coded signal decreases whereas the quality of the finally presented signal increases. This is a fundamental difference to depth distortion-based encoding (and conventional texture coding), in which both signals are the same. In SVDC-based encoding, maximizing the RD cost will not necessarily also minimize the residual of the coded depth data. Furthermore, depth map parts with reduced depth quality could be a bad predictor for subsequent pictures, especially when using the depth distortion in motion estimation as performed in the initial evaluation.

7.6 Chapter Summary

This chapter demonstrated how the RM can be applied for mode selection in encoding. For this, it discussed the different selection stages of the HTM encoder and processes invoking the RM in *GET* mode to derive the SVDC and in *SET* mode to update the RM's state. The modified encoder can use the RM for SVDC computation in all mode selection stages relevant for depth coding, but in RDOQ, which can, however, be disabled based on the SVDC. Different selection stages can use different distortion metrics. This allows trade-offs between complexity and coding performance, which will be discussed in Section 8.4.

Furthermore, the chapter summarized test conditions and configuration options. Test conditions are based on those provided by JCT-3V, but modified to focus on depth coding. They comprise in particular the evaluation setup and the encoder setup. The encoder setup using the default options, as defined in Table 7.2, is called *initial encoder setup* and used for the initial evaluation.

The initial evaluation showed that SVDC-based encoding achieves depth bit rate reductions of 75% and 50% in average for 3D-HEVC and MV-HEVC, respectively, in comparison to depth distortion-based encoding. Depth bit rate savings come at the cost of a higher encoding complexity, which increases by about 88% and 13%, respectively, and a significant loss of the quality of the depth map (about 15 dB and 10 dB, respectively). Furthermore, this chapter showed that the set of selected modes differs in depth distortion-based encoding and SVDC-based encoding. In particular, an SVDC-based encoder allocates bits for signalling residuals and prediction parameters more selectively such that more bits are spend for a smaller fraction of all depth samples, whereas the percentage of samples coded in skip mode increases significantly. Results of this chapter are the basis for further evaluations in the following chapters that discuss different encoder and RM setups.

8 Encoder Optimization

The last chapter evaluated SVDC-based encoding using the *initial encoder setup* and pointed out further configuration options. This chapter explores how these options affect the RD performance and complexity to determine the optimal setup. More specifically, Section 8.1 discusses which features of the RM should be enabled. Section 8.2 classifies and discusses different options to generate the tested and the reference SV textures used for SVDC computation. Section 8.3 analyzes how an additional depth distortion term leads to coding gains. Section 8.4 explores the usage of the SVDC in lower-level mode selection stages and the complexity introduced by this. Finally, Section 8.5 addresses which Lagrange multipliers and quantization parameters are optimal in SVDC-based encoding.

8.1 Renderer Model Setups

As discussed in Section 5.2.1, the RM comprises basic techniques like warping, hole filling and occlusion detection. On top of that, it supports warping with quarter sample precision and view combination. Other features of the RM are SVD computation in chroma components and SVDC computation. The initial evaluation used all these features. A question left open is how these different methods contribute to the overall RD performance and the complexity. It is answered in this section, which addresses SVDC computation (Section 8.1.1), SVDC computation for chroma components (Section 8.1.2), quarter sample accurate warping (Section 8.1.3), and view combination (Section 8.1.4). *Related works*: An own evaluation using the RM with disabled SVD computation for chroma and disabled *SET* operations is provided in the own contribution [Tech 12a]. In contrast to this, the evaluation in this section is based on the latest HTM software and different test conditions.

8.1.1 Synthesized View Distortion Change

One of the contributions of this thesis is the SVDC for exact SVD quantification. As described in Chapter 4, the SVDC is the SVD in the SV region that changes because of a depth change minus the SVD in this SV region before the depth change. The SVDC is thus an additive measure by construction. A simplification is to give up additivity and to use only the SVD in the changed SV region. Coding results for this setup are shown in Figure 8.1 and Table 8.1 in comparison to the *initial encoder setup*, i.e. the setup using default parameters, as specified in Table 7.2: Not using the SVDC increases the depth bit rate by 6.7% and reduces the depth encoding time by about 2%. Reason for the latter is primarily that the encoder does not need to invoke the RM in *SET* mode to update its state; furthermore, memory bandwidth is reduced as the old SVD of the changed SV region is not required in distortion computation. In conclusion, the complexity reduction is relatively low considering the resulting depth bit rate increase. This justifies to use the SVDC instead of the SVD.

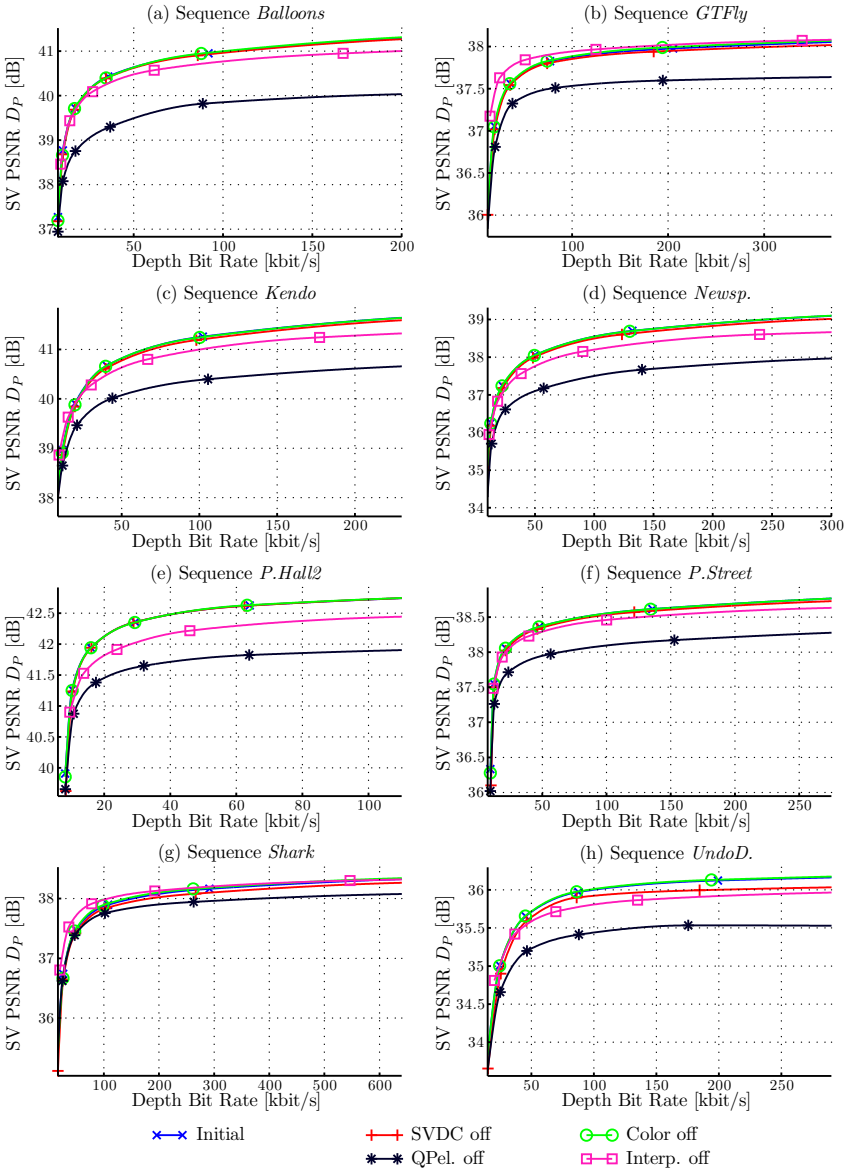


Figure 8.1: RD performance of SVDC-based encoding when disabling particular features of the RM.

	$R_\Delta[\%]$				$T_\Delta[\%]$			
	SVDC off	Color off	QPel. off	Interp. off	SVDC off	Color off	QPel. off	Interp. off
Balloons	+ 3.2	+0.3	+ 95.0	+20.0	−2	−4	−10	− 9
Kendo	+ 5.4	+1.0	+ 68.4	+19.3	−2	−3	−10	− 6
Newsp.	+ 6.8	−0.3	+101.1	+35.7	−2	−4	− 9	− 8
P.Hall2	+ 2.1	+0.2	+ 50.8	+57.2	−3	−4	−13	− 9
P.Street	+ 6.8	−0.1	+ 65.9	+25.3	−3	−4	−10	−11
GTFly	+ 4.9	−2.5	+ 42.8	−38.2	−3	−4	−10	−11
Shark	+10.2	−0.4	+ 12.9	−28.4	−2	−4	−12	− 9
UndoD.	+14.2	−0.8	+ 40.5	+32.3	−3	−4	−10	−11
Mean	+ 6.7	−0.3	+ 59.7	+15.4	−2	−4	−10	− 9

Table 8.1: Depth bit rate delta R_Δ and depth encoding time delta T_Δ when disabling particular RM features. Reference: All features enabled.

	$R_\Delta[\%]$ (YUV) Color off	$R_\Delta[\%]$ (Y) Color off	$R_\Delta[\%]$ (U) Color off	$R_\Delta[\%]$ (V) Color off
Balloons	+10.4	+0.3	+61.4	+54.4
Kendo	+ 2.7	+1.0	+19.4	+13.5
Newsp.	+ 4.0	−0.3	+23.8	+27.7
P.Hall2	+ 1.3	+0.2	+ 9.4	+12.5
P.Street	+ 1.3	−0.1	+22.1	+25.9
GTFly	+ 0.7	−2.5	+12.4	+12.7
Shark	+19.7	−0.4	+64.5	+38.1
UndoD.	− 0.4	−0.8	+ 6.0	+ 3.5
Mean	+ 5.0	−0.3	+27.4	+23.5

Table 8.2: Depth bit rate delta R_Δ with respect to a combined PSNR (YUV), the luma PSNR (Y), and the chroma PSNRs (U and V) when neglecting chroma in SVDC computation. Reference: Luma and chroma regarded.

8.1.2 SVDC Computation in Chroma Components

In the initial evaluation, the evaluation setup considered for simplification only the PSNR of the SV texture's luma component, whereas the encoder-side SVDC computation considered also the distortion of the SV texture's chroma components. This approach left open two questions: The first question is how not regarding the SV's chroma components in SVDC computation affects the coding results for luma components; the second is how it affects results that also consider chroma components.

To answer the first question, Figure 8.1 and Table 8.1 provide evaluation results for SVDC computation without chroma components. Averaged over sequences, the depth bit rate decreases by 0.3% with respect to the PSNR of SV texture's luma component. However, the depth bit rate increases for sequences *Balloons* and *PoznanHall2*. Considering the chroma component SVD

can thus lead to encoder decisions that are also beneficial for the performance related to the luma component only. An explanation for this might be that depth maps parts that are important for prediction are better preserved. Furthermore, results show a depth encoding time decrease of 4% as operations for blending and SVD computation in chroma components are not necessary.

To answer the second question, depth bit rate savings have been derived with respect to a combined PSNR that represents the quality of luma and chorma components jointly. The combined PSNR is the weighted average of the luma PSNR and the two chroma PSNR values in which the luma PSNR's weight is one and the weights of the chroma PSNRs are both 0.5. Table 8.2 shows that, when SVDC computation does not regard chroma components, the depth bit rate increases about 5% with respect to the combined PSNR. Considering individual sequences, results differ significantly: The depth bit rate increases by 19.7% for the *Shark* sequence, whereas it decreases by 0.4% for the *UndoDancer* sequence. These sequence dependent differences might be caused by different correlations of luma and chroma components.

In conclusion, the evaluation shows that the chroma SVDs should be regarded in SVDC-based encoding. Although coding losses due to not regarding them are in general small, high losses can occur for individual sequences when luma and chroma are weakly correlated.

8.1.3 Shift Precision

In the *initial encoder setup*, the RM warped with quarter sample precision. Figure 8.1 shows that when using integer precision instead, SV PSNRs decrease significantly (test case QPel. off). According to Table 8.1, the depth bit rate increases by 59.7%, which can be explained as follows: Since the RM renders the reference and the tested SV texture with integer precision, it is not sensitive to SVDs due to fractional disparity distortions. Consequently, depth encoding systematically introduces small disparity mismatches. Although small, disparity mismatches can cause large SVDs in regions with high contrast; furthermore, even when small SVDs are caused, they occur in a large number. Both leads to the SV PSNR drop.

Beyond the PSNR decrease, Table 8.1 shows a depth encoding time decrease by about 10% since the RM performs less rounding operations and does not need to perform the LUT based mapping, which accesses the upsampled IV texture. Consequently, not only the number of table look-ups decreases, but potentially also cache misses as the RM accesses the IV texture only, which requires significantly less memory than its upsampled version.

In conclusion, the high SV PSNR decrease shows that quarter sample precise warping is essential for SVDC-based encoding.

8.1.4 View Combination

In the *initial encoder setup*, the RM model performs view interpolation: It first extrapolates the left and the right SV texture from the left and the right IV texture, respectively, and then combines both as shown in Table 3.1. View combination includes distance depended blending

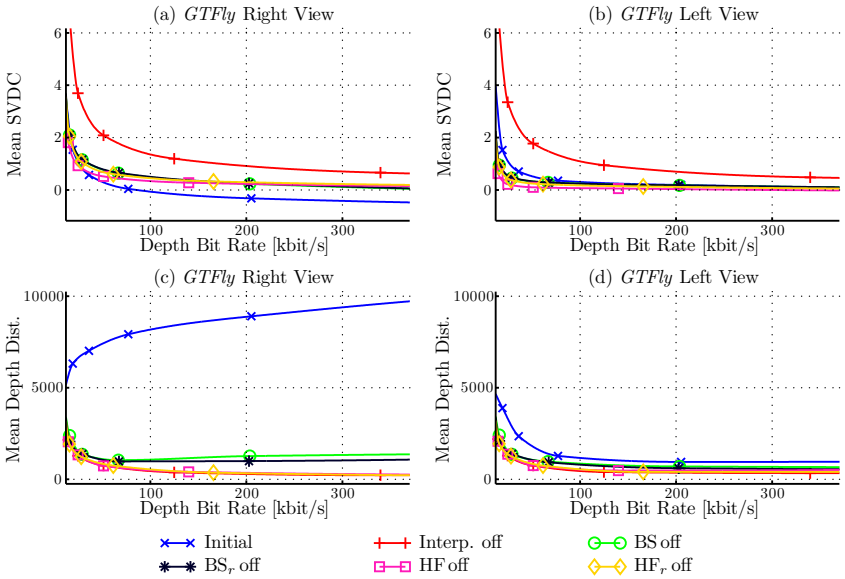


Figure 8.2: Implications of the hiding effect: When encoding the right IV depth in the *initial encoder setup*, the depth distortion, i.e. the mean squared error (MSE), increases with the depth bit rate while the measured SVDC is negative.

if extrapolated samples of both IVs are available for a particular SV position (case 3b), and background suppression in case of unreliable depth values (case 3a). An alternative, which is evaluated in this section, is neglecting the view combination step and using only view extrapolation. This means that the RM calculates the SVDC that occurs in the left SV when encoding the left IV, and, accordingly, the SVDC that occurs in the right SV when encoding the right IV.

Table 8.1 shows simulation results for this approach in comparison to the *initial encoder setup* (test case *Interp. off*). The depth bit rate increases by 15.4%. However, results vary extremely for individual sequences. Whereas depth bit rates of the natural sequences increase by about 25% to 57%, depth bit rates of the computer generated sequences *GTFly* and *Shark* decrease by about 38% and 28%, respectively.

In order to analyze the reasons for these differences, Figure 8.2 shows the SVDC and the squared depth difference (both averaged over all depth samples) that occur when encoding the right and the left IV depth of the *GtFly* sequence[1] (in which the right IV depth is coded first). Comparing results obtained with the *initial encoder setup* for the right IV and the left IV shows two major differences: First, for depth bit rates greater than approximately 75 kbit/s, the average SVDC of the right IV is negative, whereas the SVDC of the left IV is positive; and second, the depth

[1] Appendix F.3 provides results for the other sequences.

distortion of the right IV is not only significantly higher than for the left IV, but also increases with increasing depth bit rates. In summary, when encoding the right IV, the encoder selects coding modes in a way that distorts the depth map, but increases the SVDC.

A hypothesis why this is happening is that the encoder encodes the right IV depth such that parts of the right SV texture are hidden in the combined SV texture. This is enabled by three aspects related to SVDC computation, the view combination process and the test sequences.

As described in Section 4.4.3, the RM uses the uncoded left IV texture when encoding the right IV depth because the coded left IV texture is not available yet. More specifically, for SVDC computation, it renders the reference SV texture by combining the left SV and the right SV texture, which are extrapolated from their respective uncoded IV textures. It renders the tested SV texture, however, from the coded right IV texture and the uncoded left IV texture. This means that the view combination step combines the right SV texture, which is extrapolated from the coded right IV texture, with the left SV texture, which extrapolated from the uncoded left IV texture. The RM then calculates the local SVD as squared difference of the tested and the reference SV texture. From this, it can be concluded that the local SVD (and thus the SVDC) can decrease when the following two conditions are true:

1. The left SV texture, which is extrapolated from the uncoded left IV texture, is very similar to the reference SV texture.
2. The view combination step favors the left SV texture over the right SV texture, which is extrapolated from the coded right IV texture.

Whereas the first condition is generally not fulfilled for natural sequences because of noise in the IV depth maps and textures, it can be fulfilled for the synthetic sequences of the test set as they can be considered as noise free. The second condition is fulfilled in two cases: The first case occurs when the absolute depth difference of a left and a right SV texture sample that are collocated exceeds a particular threshold; then the view combination step discards the SV texture sample that is in the background, which is called background suppression (BS) (see Section 3.2.4). Consequently, the encoder can locally hide the right SV texture sample by choosing a coding mode that reduces its absolute depth value. The second case occurs when the right SV texture has a disocclusion. Then the RM fills this disocclusion with samples of the left SV texture. The encoder can thus shift distorted samples of the right IV texture so that they are occluded. In both cases, the left SV texture, which is extrapolated from the uncoded left IV texture, becomes visible and, for synthetic sequences, the SVDC is thus reduced.

However, when encoding the right IV depth, the uncoded left IV texture is only used as an assumption for rendering because its coded version is not yet available. This means that the SVDC reductions caused by favoring samples extrapolated from the uncoded left IV texture are only seemingly reductions (e.g. the negative average SVDC in Figure 8.2) because the coded IV texture is finally used by the receiver-side renderer. This hiding effect can lead to coding losses when the left IV texture is more distorted than the right IV texture.

To test the hypothesis that depth bit rate reductions observed for synthetic sequences in the extrapolation setup are rather related to the losses caused by the hiding effect in the *initial encoder setup*, further evaluations have been conducted. Table 8.3 and Figure 8.2 show results

	$R_\Delta[\%]$				
	Interp. off	*BS off*	*BS$_r$ off*	*HF off*	*HF$_r$ off*
Balloons	+20.0	− 2.8	− 2.4	+ 7.6	− 0.9
Kendo	+19.3	− 7.7	−10.5	+ 4.5	− 9.3
Newsp.	+35.7	− 1.3	− 2.4	+19.6	+ 1.1
P.Hall2	+57.2	+ 0.7	0.0	+29.5	+17.2
P.Street	+25.3	− 2.2	− 2.9	+ 7.1	− 1.3
GTFly	−38.2	−25.3	−27.4	−31.0	−32.6
Shark	−28.4	−18.2	−20.0	−25.9	−28.0
UndoD.	+32.3	−10.2	−10.8	+ 6.5	−13.9
Mean	+15.4	− 8.4	− 9.6	+ 2.2	− 8.5

Table 8.3: Depth bit rate delta R_Δ; Tested: Different view combination variants; Reference: The *initial encoder setup*, i.e. *Default* view combination as defined in Table 3.1.

for four additional view combination variants, in which the following aspects of the RM's view combination process have been disabled:

- *BS off*: For encoding of both IV depth maps, background suppression. Cases 3a and 3a' in Table 3.1 are treated as case 3b.
- *BS$_r$ off*: For encoding of the right IV depth, background suppression of the right SV texture. Case 3a is treated as case 3b.
- *HF off*: For encoding of both IV depth maps, background suppression and hole filling from the other view. All cases are treated as case 3b.
- *HF$_r$ off*: For encoding of the right IV depth, background suppression of the right SV texture and hole filling from the left SV texture. Cases 1, 2, and 3a are treated as case 3b.

Figure 8.2 shows for the *GTFly* sequence that for all view combination variants the average SVDC derived in depth coding of the right IV is no longer negative. Furthermore, the depth distortion of the right IV decreases to about the level of the depth distortion of the left IV. However, whereas the depth distortion decreases with the depth bit rate for *HF off* and *HF$_r$ off* view combination variants, it still increases for *BS off* and *BS$_r$ off* view combination variants, which means that the hiding effect still occurs, although less severe.

Table 8.3 shows the RD performance of the view combination variants. All view combination variants, but *HF off* view combination, achieve depth bit rate savings. Although results for *HF off* view combination show a decrease of the depth bit rate for the *GTFly* and *Shark* sequences (which are according to the reductions in the *Interp. off* test case most severely affected by the hiding effect), the depth bit rate increases for the other sequences. In summary, coding losses caused by neglecting background suppression and hole filling exceed coding gains due to avoiding the hiding effect. *HF$_r$ off* view combination achieves on average higher depth bit rate savings because of two reasons: It avoids coding losses introduced by background suppression and hole filling when coding the right IV depth; and it avoids coding losses caused by neglecting background suppression and neglecting hole filling when coding the left IV depth, which is

not affected by the hiding effect. Depth bit rate savings are again the highest for the *GtFly* and the *Shark* sequence. On the other hand, depth bit rates of the *PoznanHall2* and the *Newspaper* sequence increase, which indicates that neglecting background suppression and hole filling can also be disadvantageous when the hiding effect occurs only weakly.

Comparing *BS$_r$ off* and *HF$_r$ off* view combination shows that high depth bit rate reductions can already be achieved by disabling background suppression only. Coding gains for the synthetic sequences are lower for *BS$_r$ off* view combination, whereas losses for *PoznanHall2* and *Newspaper* decrease. Considering furthermore that *BS off* view combination achieves in contrast to *HF off* view combination still bit rate reductions, it can be concluded that neglecting background suppression leads to much lower losses than neglecting hole filling.

In summary, it is generally beneficial to consider view combination. However, coding losses can occur for sequences with high quality texture and depth as a hiding effect occurs when calculating the SVDC for the right IV depth while assuming an uncoded left IV texture. A method to resolve this issue is disabling the depth threshold based suppression of background samples. Other methods to mitigate this problem will be explored in Sections 8.2 and 8.3.

8.1.5 Conclusion

The section evaluated how different RM setups affect the coding performance and complexity of SVDC-based encoding and showed the following:

- Using the SVD instead of the *SVDC* results in a depth bit rate increase of about 6.7% and a depth encoding time decrease of about 2%.
- Neglecting the SV texture's *chroma components* reduces the depth encoding time by 4% and the RD performance for most of the sequences insignificantly. However, for one sequence the depth bit rate increases by about 20% with respect to a combined luma and chroma PSNR. To avoid artifacts in chroma components, they should be considered in SVDC computation.
- Rendering with integer sample precision instead of *quarter sample precision* increases the depth bit rate by about 60%. Quarter sample precise warping is thus essential.
- Without *view combination* the depth encoding time decreases by 6%. The depth bit rate increases for natural sequences by about 25%; for synthetic sequences, it decreases by up to 38%. Reason for this is a *hiding effect* in the view combination step, which occurs when encoding the right IV depth while assuming an uncoded left IV texture. The hiding effect can be avoided by disabling background suppression in the view combination step when encoding the right IV depth. With these modifications, the depth bit rate decreases by 9.6% on average over all sequences and about 19.4% on average over the synthetic sequences.

In summary, the evaluation confirmed that most of the RM's features effectively increase the RD performance with a moderate increase of complexity.

8.2 Synthesized View Generation Setups

In evaluations so far, the RM performed view interpolation. It interpolated the reference SV texture from uncoded IV data, and the tested SV texture from coded IV data when available and uncoded otherwise. The last section showed that this is not optimal when encoding the right IV depth because of the hiding effect. For this, this section evaluates other SV generation setups, as e.g. extrapolating or interpolating the tested and reference SV texture from different combinations of coded and uncoded IV textures and depths. It classifies different options (Section 8.2.1) and discusses their practical importance, RD performance, and complexity (Section 8.2.2).

Related works: Different SV generation setups for the RM have been suggested in the own contribution [Tech 12b], but they have not been evaluated.

8.2.1 Classification of Synthesized View Generation Setups

The classification considers the constraints imposed by 3D-HEVC and the test conditions: Layers are coded in the order right IV texture, right IV depth, left IV texture, and left IV depth; furthermore, the SV texture at the receiver-side is rendered by view interpolation.

8.2.1.1 Interpolation and Extrapolation

In the two view scenario, the encoder can employ four different combinations of interpolation and extrapolation, called *EE*, *II*, *EI*, and *IE* setup in the following:

- *EE*: When encoding the right and the left IV depth, the RM extrapolates the SV texture from the data of the respective IV only. SVDC computation thus does not depend on data of the respective other IV and ignores view interpolation, as conducted at the receiver-side.
- *II*: In encoding of both IV depths, the RM applies view interpolation, so encoding of the right IV depth depends also on data of the left IV and encoding of the left IV depth on data of the right IV. SVDC computation can fully regard view interpolation, but because of the coding order only use the uncoded texture and depth of the left IV, which leads to the hiding effect.
- *EI*: When encoding the right IV depth, the RM performs view extrapolation, which might be beneficial as coded data of the left IV is still unknown. When encoding the left IV depth, the RM performs view interpolation using the texture and depth of the right IV.
- *IE*: The RM interpolates the SV texture when encoding of the right IV depth and extrapolates it when encoding the left IV depth. Because such a setup has no benefits, it is not investigated.

8.2.1.2 Generation of the Tested and the Reference Synthesized View Texture

Beyond the choice between view interpolation and view extrapolation, there are multiple options to select the IV data for generation of the tested and the reference SV textures. The set of available options depends on whether the RM performs view extrapolation or interpolation.

Synthesized View Generation Setups for View Extrapolation

In view extrapolation, the SV textures depend on the IV texture and the IV depth that is currently encoded. According to the idea of the SVDC, the RM uses the partly coded IV depth for rendering the tested SV texture $\tilde{s}'_T$ and the uncoded IV depth for rendering the reference SV texture $s'_{T,Ref}$. What remains to be answered is whether the RM should use the coded or the uncoded IV texture for rendering the tested SV texture $\tilde{s}'_T$ and the reference SV texture $s'_{T,Ref}$. Possible combinations are shown in Figure 8.3 and denoted with o, v, n, and c.

- Combination o uses the uncoded IV texture for rendering $s'_{T,Ref}$ and $\tilde{s}'_T$, which allows texture independent depth coding. The SVD is related to depth coding while assuming the uncoded IV texture. SVDC computation not only neglects SVDs caused by texture coding, but also neglects that the IV texture is coded.
- Combination v compares $s'_{T,Ref}$ rendered with the uncoded IV texture to $\tilde{s}'_T$ rendered with the coded IV texture. SVDs can in principle be related to 1) texture coding, e.g. IV samples with incorrect values warped to correct SV positions; 2) depth coding, e.g. IV samples with correct values warped to incorrect SV positions; and 3) texture and depth coding jointly, e.g. IV samples with incorrect values warped to incorrect SV positions. Since the SVDC is the difference of the SVDs after and before a depth change, SVDs caused by texture coding only are canceled out. SVDCs are thus related to the SVDs caused by depth coding and caused by depth and texture coding jointly. The sensitivity to SVDs caused jointly by texture and depth coding allows the encoder to reduce SVDs due to texture coding in depth coding. E.g. it can encode the depth of a distorted IV texture sample such that it is occluded in the SV.
- Combination n uses the coded IV texture to render $s'_{T,Ref}$ and the uncoded IV texture to render $\tilde{s}'_T$. It has no practical use and is thus not investigated further.
- Combination c renders $s'_{T,Ref}$ and $\tilde{s}'_T$ using the coded IV texture. The SVD is related to depth coding while assuming coded IV texture, but without being biased by SVDs caused by texture coding only. SVDC computation thus regards that the IV texture is coded. This has the advantage that it is not sensitive to SVDs that are related to signal parts in the uncoded IV texture that are removed by texture coding. Signal parts of the IV depth that would cause such SVDs can thus be removed by the encoder. E.g. the encoder can remove details in the IV depth that become irrelevant for rendering because the collocated details in the IV texture are lost by texture coding.

In summary, there are three reasonable options for each of the two IVs in the scenario evaluated in this thesis. To distinguish between them, this thesis uses the following notation:

$$E(\tau_r\delta_r)E(\tau_l\delta_l) \text{ with } \tau_r, \tau_l \in \{o, v, c\} \text{ and } \delta_r, \delta_l \in \{x\} \tag{8.1}$$

$E(\tau_r\delta_r)$ specifies the SV generation setup when encoding the right IV depth and accordingly $E(\tau_l\delta_l)$ the setup for the left IV depth. E denotes that the RM performs view extrapolation. τ_r and τ_l specify the IV textures the RM uses for rendering $\tilde{s}'_T$ and $s'_{T,Ref}$, in which o, v, and c indicate the combinations as discussed above. δ_r and δ_l specify the IV depth maps used for rendering. For view extrapolation the only option is x, which indicates that the RM uses partly coded depth to render $\tilde{s}'_T$, and the uncoded depth to render the $s'_{T,Ref}$. Further options for δ will be introduced for view interpolation in Section 8.2.1.2.

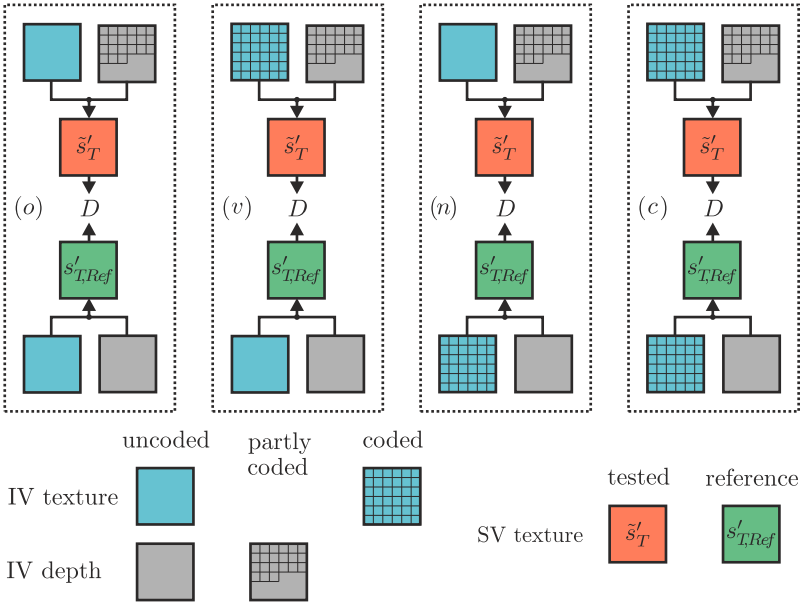


Figure 8.3: Setups to generate the reference SV texture $s'_{T,Ref}$ and the tested s'_T SV texture in view extrapolation.

Synthesized View Generation Setups for View Interpolation

As view interpolation requires two IV textures and two IV depths, there is a great variety of possible combinations. To distinguish between them, the notation

$$I(\tau_{rr}\delta_{rr}\tau_{rl}\delta_{rl})I(\tau_{lr}\delta_{lr}\tau_{ll}\delta_{ll}) \tag{8.2}$$

is used, in which I indicates view interpolation and $\tau_{\chi\rho}$ and $\delta_{\chi\rho}$ the following:

- $\tau_{\chi\rho}$ specifies for encoding of the left IV depth (if χ is l) or the right IV depth (if χ is r), the left IV textures (if ρ is l) or the right IV textures (if ρ is r) that the RM uses to interpolate the tested and reference SV textures; and
- $\delta_{\chi\rho}$ specifies for encoding of the left IV depth (if χ is l) or the right IV depth (if χ is r), the left IV depth maps (if ρ is l) or the right IV depth maps (if ρ is r) that the RM uses to interpolate the tested and reference SV textures.

Values of $\tau_{\chi\rho}$ and $\delta_{\chi\rho}$ can be selected from $\{o, v, c\}$ and $\{o, v, c, x\}$, respectively, in which the expressions specify the same as discussed above (i.e. $\tilde{s}'_T$ is rendered from coded data in cases c and v, partly coded data in case x, and uncoded data in case o; and $s'_{T,Ref}$ is rendered from coded data in cases c and v, and uncoded data in cases o and x). Not considered are combinations that

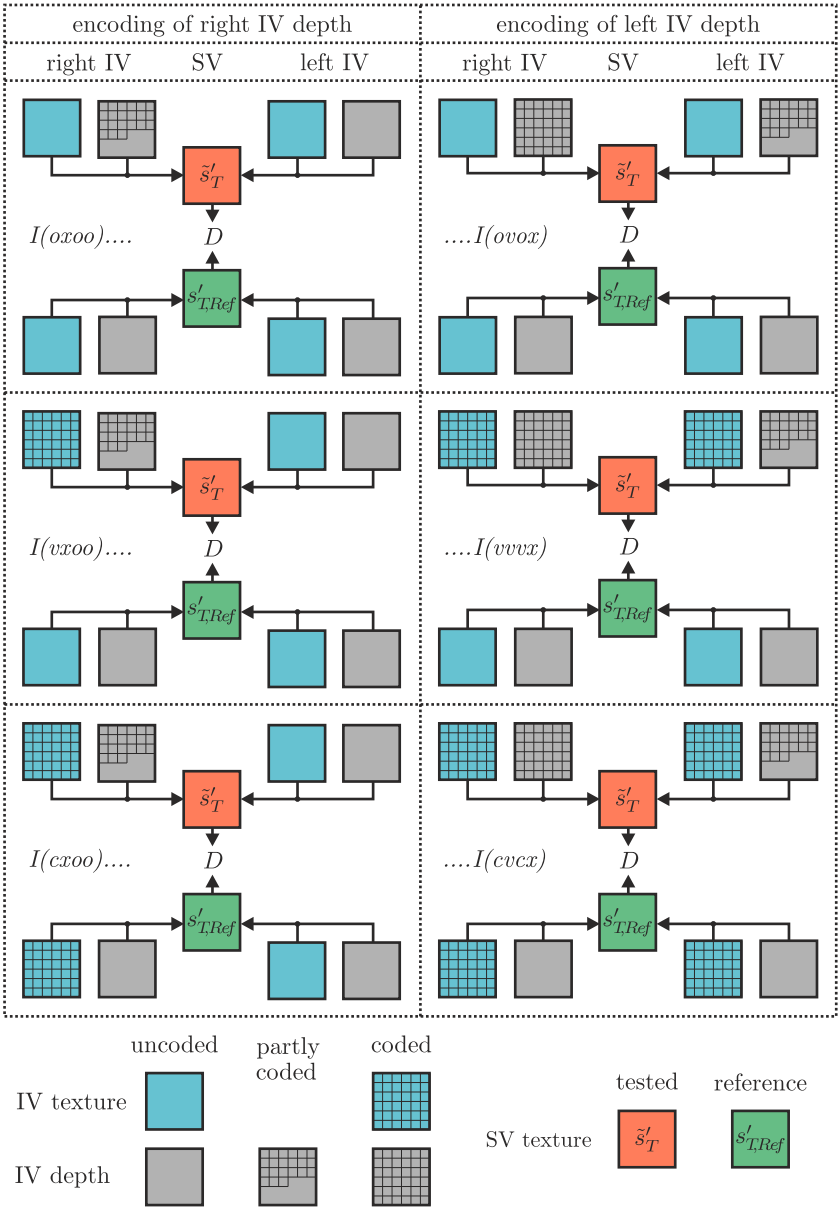


Figure 8.4: Setups to generate the reference SV texture $s'_{T,Ref}$ and the tested SV texture s'_T for view interpolation.

use uncoded data to render $\tilde{s}'_T$ and coded data to render $s'_{T,Ref}$ (i.e. case n).

When encoding the right IV depth, the RM uses the partly coded right IV depth to render $\tilde{s}'_T$ and uncoded right IV depth to render $s'_{T,Ref}$. As only the uncoded left IV texture and depth is available, possible combinations for encoding the right IV depth are given by $I(\tau_{rr}xoo)$, with $\tau_{rr} \in \{c, v, o\}$. SVDC computation can either neglect (o) or regard (c) that the right IV texture is coded, or can take SVDs jointly caused by depth and texture coding into account (v).

When encoding the left IV depth, the coded right and left IV textures are known, and also the coded right IV depth. Consequently, the RM can use one of 27 combinations which are given by $I(\tau_{lr}\delta_{lr}\tau_{ll}x)$ with $\tau_{lr}, \delta_{lr}, \tau_{ll} \in \{c, v, o\}$.

In conclusion, the SV generation setup for the right IV and left IV can be selected from three and 27 combinations, respectively. When considering both IVs, 81 combinations are available.

8.2.2 Evaluation

This section evaluates how SV generation setups differ in RD performance and complexity. It first answers for view interpolation which IV texture the RM should use when encoding the right IV depth (Section 8.2.2.1), and which IV textures (Section 8.2.2.2) and IV depths (Section 8.2.2.3) the RM should use when encoding the left IV depth. Then, Section 8.2.2.4 discusses setups using view extrapolation before Section 8.2.2.5 analyzes encoding complexity.

8.2.2.1 Selection of τ_{rr} in View Interpolation

Evaluated Setups

The *initial encoder setup* uses the $I(vxoo)I(vvvx)$ setup. When encoding the right IV depth, this setup renders $s'_{T,Ref}$ from uncoded data of both IVs, and $\tilde{s}'_T$ from the coded right IV texture and the uncoded left IV data. Motivation for this approach was that the initial SVD (i.e. the SVD before encoding of the right IV depth starts) includes SVDs introduced by coding of the right IV texture, so that an encoder can modify the right IV depth in a way that reduces these SVDs. However, the uncoded left IV data are only a surrogate that is used because the coded left IV data are not yet available; this leads to the hiding effect and coding losses.

When encoding the left IV depth, the $I(vxoo)I(vvvx)$ setup renders $s'_{T,Ref}$ from uncoded data of both IVs, and $\tilde{s}'_T$ from the coded right IV data and the coded left IV texture. SVDs introduced by encoding of preceding layers are thus included in the initial SVD when the encoder starts processing the left IV depth. The encoder can thus encode left IV depth in a way that reduces these SVDs. Unlike to encoding of the right IV depth, the coded IV data are not assumptions, the measured SVDC is correct with respect to the SVDC measured at the receiver-side renderer.

In summary, because of the hiding effect, the $I(vxoo)I(vvvx)$ setup is not optimal for encoding of the right IV depth. For this, the following evaluates setups $I(\boldsymbol{o}xoo)I(vvvx)$ and $I(\boldsymbol{c}xoo)I(vvvx)$ as alternatives.[2]

[2]Specifiers varied in an evaluation are in the following set in bold characters.

	$R_\Delta[\%]$							
	Default		*BS$_r$ off*			*HF$_r$ off*		
τ_{rr}	c	o	v	c	o	v	c	o
Balloons	− 0.3	+ 1.0	− 2.4	− 2.0	− 0.8	− 0.9	− 0.1	+ 1.1
Kendo	− 5.4	− 0.5	−10.5	−10.2	− 8.6	− 9.3	− 7.5	− 6.6
Newsp.	− 0.5	+ 1.2	− 2.4	− 1.6	− 0.2	+ 1.1	+ 1.6	+ 2.3
P.Hall2	− 0.9	+ 0.1	0.0	− 0.9	+ 0.1	+17.2	+14.7	+14.3
P.Street	− 1.5	− 0.9	− 2.9	− 3.4	− 2.7	− 1.3	− 1.0	− 0.6
GTFly	−23.5	−17.7	−27.4	−30.5	−33.1	−32.6	−32.5	−35.5
Shark	−12.5	−11.5	−20.0	−22.1	−22.9	−28.0	−25.0	−28.4
UndoD.	− 9.4	− 6.9	−10.8	−12.7	−12.9	−13.9	−14.7	−15.7
Mean	− 6.7	− 4.4	− 9.6	−10.4	−10.1	− 8.5	− 8.1	− 8.6

Table 8.4: Depth bit rate delta R_Δ for different view combination variants and SV generation setups. Reference: *Default* view combination and $\tau_{rr} = v$. All: $I(\tau_{rr}xoo)I(vvvx)$

In encoding of the right IV depth, the $I(\boldsymbol{o}xoo)I(vvvx)$ setup renders $s'_{T,Ref}$ from the uncoded data of both IVs, and $\tilde{s}'_T$ from the uncoded textures of both IVs, the uncoded left IV depth, and the partly coded right IV depth. When encoding of the right IV depth starts, the initial SVD is thus zero. The calculated SVDC is the SVDC that is introduced by depth coding for rendering with uncoded IV textures. This means that it neither includes SVDs introduced by texture coding, nor considers that IV textures are coded at the receiver.

The $I(\boldsymbol{c}xoo)I(vvvx)$ setup renders $s'_{T,Ref}$ from the coded right IV texture, the uncoded right IV depth, and the uncoded left IV data; furthermore, it renders $\tilde{s}'_T$ from the coded right IV texture, the partly coded right IV depth, and the uncoded left IV data. As the setup renders $s'_{T,Ref}$ and $\tilde{s}'_T$ from the same coded IV textures, the initial SVD is zero and the SVDC does not include SVDs caused by texture and depth coding jointly. However, in contrast to the $I(\boldsymbol{o}xoo)$ $I(vvvx)$ setup, it considers that the right IV texture is coded.

As the evaluated setups are given by $I(\tau_{rr}xoo)I(vvvx)$, with $\tau_{rr} \in \{v, o, c\}$, the remainder of this section calls them, for brevity of description, alternatively v, o, and c setup, respectively.

Evaluation Results

Table 8.4 compares the depth bit rate savings in different test cases obtained from combining the three SV generation setups—v, o, and c—with the three view combination variants discussed in Section 8.1.4—*Default*, *BS$_r$ off*, and *HF$_r$ off*. Reference in the evaluation is the v SV generation setup using *Default* view combination, i.e. the *initial encoder setup*. This way, effects related to the reduction of the hiding effect can be studied.

Results show that in particular the depth bit rate of synthetic sequences decreases. This indicates that the hiding effect is reduced in all test cases, either because of the view combination variant when the RM uses the v SV generation setup, or because of the modified SV generation setup when it uses *Default* view combination, or because of both. Possible explanations for the reduction of the hiding effect due to the modified SV generation setup differ for setups o and c. The o setup extrapolates in contrast to the v setup also the right SV texture from the uncoded

right IV texture to render $\tilde{s}'_T$. Consequently, the SVD of the right SV texture and thus the SVD difference between the left and the right SV texture, which causes the hiding effect, decreases. The c setup renders $s'_{T,Ref}$ in contrast to the v setup from the coded right IV texture. Thus, when the view combination variant favors the left SV texture, which is rendered from the uncoded left IV texture, the measured SVDC increases. This decreases the likelihood that the encoder hides the right SV texture by modifying the right IV depth.

Comparing *Default* view combination using the o and the c SV generation setup shows that the latter achieves higher depth bit rate reductions (i.e. 6.7% vs. 4.4%, respectively). An explanation for this is that the SVDC calculated in c setup matches better with the SVDC that occurs finally at the receiver-side renderer. In the c setup, the SVDC already reflects that high frequent parts in right SV texture are lost by coding of the right IV texture, so that the encoder can remove details in the right IV depth that are only important to preserve these parts in rendering.

Regarding the test cases using *BS$_r$ off* view combination, depth bit rate reductions for the v, the c, and the o setup are 9.6%, 10.4%, and 10.1%, respectively. This means that compared to *Default* view combination, depth bit rate reduction differences between the c and v setup decrease from 6.7% to 0.8% (i.e. $10.4\% - 9.6\%$) and between the o and v setup from 4.4% to 0.5% (i.e. $10.1\% - 9.6\%$). Obvious reason for this is that disabling background suppression already reduced the hiding effect; reducing it further in the c and the o setup only achieves moderate additional depth bit rate reductions. Considering individual sequences supports this finding. Whereas depth bit rates for sequences suffering severely from the hiding effect further decrease when using the c or o setup instead of the v setup (e.g. *GtFly* -30.5% and -33.1% instead of -27.4%), depth bit rates for sequences that are only marginally affected by the hiding effect increase (e.g. *Newspaper* -1.6% and -0.2% instead of -2.4%).

For test cases using *HF$_r$ off* view combination, depth bit rates decrease by 8.5%, 8.1%, and 8.6% for the v, c, and o setup, respectively. Thus, in contrast to the test cases using *Default* and *BS$_r$ off* view combination, the c setup does not reduce the depth bit rate compared to the v setup, whereas the o setup does only insignificantly—disabled hole filling already reduces the hiding effect. That the o setup outperforms the c setup might be a consequence of that the SV texture rendered from the uncoded IV texture comprises more high frequent signal parts. To preserve them, the encoder encodes the depth map with a higher quality, which could have the side-effect that SVDs caused by not regarding hole filling are mitigated.

Conclusion
Setups $I(\boldsymbol{o}xoo)I(vvvx)$ and $I(\boldsymbol{c}xoo)I(vvvx)$ outperform the $I(\boldsymbol{v}xoo)I(vvvx)$ setup when using *Default* view combination and when background suppression is disabled, as they do not regard the SVD introduced by coding of the right IV texture and thus reduce the hiding effect. When the hiding effect is already reduced by disabling hole filling, they do not achieve significant additional depth bit rate reductions. The $I(\boldsymbol{c}xoo)I(vvvx)$ setup with disabled background suppression for the right view achieves the highest depth bit rate reductions (10.4%) compared to the *initial encoder setup*. Nevertheless, the $I(\boldsymbol{o}xoo)I(vvvx)$ setup can be an option for scenarios requiring an independent encoding of the right IV's texture and depth.

	$R_\Delta[\%]$							
Right IV	$I(\boldsymbol{c}xoo)$			$I(\boldsymbol{o}xoo)$			$E(vx)$	
Left IV	$I(\boldsymbol{vvv}x)$	$I(\boldsymbol{cvc}x)$	$I(\boldsymbol{ccc}x)$	$I(\boldsymbol{vvv}x)$	$I(\boldsymbol{ovo}x)$	$I(\boldsymbol{ooo}x)$	$E(vx)$	$I(vvvx)$
Balloons	+0.5	+2.7	+ 7.2	+1.7	+3.6	+10.0	+22.8	+21.4
Kendo	+0.5	+4.9	+11.1	+2.7	+6.5	+23.6	+31.9	+21.5
Newsp.	+0.9	+4.7	+ 9.1	+2.3	+5.0	+15.0	+38.4	+27.0
P.Hall2	−0.9	+0.5	+ 1.7	0.0	+1.5	+ 3.5	+57.1	+37.3
P.Street	−0.6	+2.2	+ 5.0	−0.1	+3.4	+10.0	+27.1	+21.5
GTFly	−5.3	−1.8	+ 1.8	−6.7	−4.6	+ 8.4	− 5.4	− 5.7
Shark	−3.4	+0.5	+ 4.9	−3.7	−2.2	+32.9	− 3.7	− 7.0
UndoD.	−2.7	−1.9	+ 0.7	−2.8	−2.0	+16.9	+48.2	+14.4
Mean	−1.4	+1.5	+ 5.2	−0.8	+1.4	+15.0	+27.1	+16.3

Table 8.5: Depth bit rate delta R_Δ; Tested: Different SV generation setups; Reference: $I(vxoo)$ $I(vvvx)$ setup; All: *BS$_r$ off* view combination.

8.2.2.2 Selection of τ_{lr} and τ_{ll} in View Interpolation

The last section studied different SV generation setups for encoding of the right IV depth. For encoding of the left IV depth, the RM generated the tested SV texture by considering the coded IV textures of both IVs [i.e. $I(\cdot xoo)I(\boldsymbol{vvv}x)$]. This approach was based on the hypothesis that considering the SVDs already introduced by texture coding enables the depth encoder to reduce SVDs jointly caused by texture and depth coding. This section analyzes whether the hypothesis is true and how RD performance changes when rendering with either uncoded or coded IV textures only. For this, it evaluates two additional setups—$I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ and $I(\boldsymbol{c}xoo)I(\boldsymbol{cvc}x)$.

The $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ setup extends the $I(\boldsymbol{o}xoo)I(\boldsymbol{vvv}x)$ setup by using only uncoded IV textures also when encoding the left IV depth, so it neglects texture coding entirely. Similarly, the $I(\boldsymbol{c}xoo)I(\boldsymbol{cvc}x)$ setup extends the $I(\boldsymbol{c}xoo)I(\boldsymbol{vvv}x)$ setup by using coded IV textures. When encoding the right IV depth, it renders $\tilde{s}'_T$ and $s'_{T,Ref}$ from the coded right IV texture and the uncoded left IV texture, whereas it uses only coded IV textures when encoding the left IV depth. In both additional setups, the initial SVD solely represents the SVD due to depth coding of the right IV when the encoding of the left IV depth starts (however, with respect to different IV textures). Furthermore, SVDCs that are jointly caused by texture and depth coding are not measured.

Table 8.5 shows how the depth bit rate changes when the RM uses the discussed SV generation setups instead of the $I(\boldsymbol{v}xoo)I(\boldsymbol{vvv}x)$ setup. In all test cases, the RM uses *BS$_r$ off* view combination, which performed best so far. For all sequences, the $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ and the $I(\boldsymbol{c}xoo)$ $I(\boldsymbol{cvc}x)$ setup do not outperform the $I(\boldsymbol{o}xoo)I(\boldsymbol{vvv}x)$ and the $I(\boldsymbol{c}xoo)I(\boldsymbol{vvv}x)$ setup, respectively. Regarding the $I(\boldsymbol{c}xoo)I(\boldsymbol{vvv}x)$ setup and the $I(\boldsymbol{c}xoo)I(\boldsymbol{cvc}x)$ setup shows that the depth bit rate decreases by 1.4% and increases by 1.5%, respectively. Furthermore, the $I(\boldsymbol{o}xoo)I(\boldsymbol{vvv}x)$ setup achieves a depth bit rate reduction of 0.8%, whereas the depth bit rate increases in the $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ setup by 1.4%. These findings support the initial idea of

considering the SVDs caused by coding of the preceding layers in SVDC computation when encoding the left IV depth. Disadvantages like the hiding effect could not be found. Comparing furthermore the $I(\boldsymbol{c}xoo)I(\boldsymbol{vvv}x)$ setup to the $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ setup to analyze losses in scenarios in that the coded IV textures are not available shows that depth bit rate savings of 1.4% change to losses of 1.4%.

In conclusion, this section showed that using setups different from $I(\cdot x\cdot\cdot)I(\boldsymbol{v}\cdot\boldsymbol{v}x)$ for coding of the left IV is not advantageous, but that coding losses due to neglecting the coded IV textures are moderate.[3] SVDC computation is thus also suitable for scenarios with parallel encoding of texture and depth, which can e.g. be done in MV-HEVC.

8.2.2.3 Selection of δ_{lr} in View Interpolation

When encoding the left IV depth, all setups discussed so far render $s'_{T,Ref}$ from the uncoded right IV depth and $\tilde{s}'_T$ from coded right IV depth [i.e. $I(\cdot x\cdot\cdot)I(\cdot v\cdot x)$]. This section analyzes whether this selection is optimal. The analysis furthermore investigates how the RD performance is affected in scenarios that encode the left IV depth independently from all other layers. For this, it evaluates setups $I(\boldsymbol{o}xoo)I(\boldsymbol{ooo}x)$ and $I(\boldsymbol{c}xoo)I(\boldsymbol{ccc}x)$.

The $I(\boldsymbol{o}xoo)I(\boldsymbol{ooo}x)$ setup modifies the $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ setup by rendering also $\tilde{s}'_T$ from the uncoded right IV depth. When encoding an IV depth map, SVDC computation thus does not depend on any coded IV data of other layers. The $I(\boldsymbol{c}xoo)I(\boldsymbol{ccc}x)$ setup differs from the $I(\boldsymbol{c}xoo)I(\boldsymbol{cvc}x)$ setup similarly: When encoding the left IV depth, it also renders $s'_{T,Ref}$ from the coded right IV depth. $s'_{T,Ref}$ and $\tilde{s}'_T$ thus depend on the coded textures of both IVs and the coded right IV depth. In both setups, the initial SVD is zero when coding of an IV depth starts. Furthermore, the SVDC does not reflect any SVDCs caused jointly by depth coding of the right and left IV.

Table 8.5 shows the depth bit rate deltas for the evaluated setups in comparison to the $I(\boldsymbol{v}xoo)$ $I(\boldsymbol{vvv}x)$ setup. Comparing the $I(\boldsymbol{o}xoo)I(\boldsymbol{ovo}x)$ setup with the $I(\boldsymbol{o}xoo)I(\boldsymbol{ooo}x)$ setup shows that using the uncoded right IV depth further increases the depth bit rate from 1.4% to 15%. Moreover, the comparison of $I(\boldsymbol{c}xoo)I(\boldsymbol{cvc}x)$ and $I(\boldsymbol{c}xoo)I(\boldsymbol{ccc}x)$ setups shows that using only the coded right IV depth increases the depth bit rate further from 1.5% to 5.2%.[3]

In summary, the evaluation confirms that the best option for encoding of the left IV depth is to render $s'_{T,Ref}$ from the uncoded right IV depth and the $\tilde{s}'_T$ from the coded right IV depth as it is done in the *initial encoder setup*. Furthermore, results for the $I(\boldsymbol{o}xoo)I(\boldsymbol{ooo}x)$ setup indicate that the depth bit rate increases by about 15% when encoding of the left IV depth neglects that other layers are coded. An independent encoding of left IV depth, e.g. in a simulcast scenario, would thus not only lead to coding losses due to a restricted prediction structure, but also to losses due to a decreased performance of SVDC computation.

[3] Tables F.2 and F.3 show similar results for *Default* and HF_r *off* view combination.

	$T_\Delta[\%]$							
Right IV	$I(\boldsymbol{c}xoo)$			$I(\boldsymbol{o}xoo)$			$E(vx)$	
Left IV	$I(\boldsymbol{vvv}x)$	$I(\boldsymbol{cvc}x)$	$I(\boldsymbol{ccc}x)$	$I(\boldsymbol{vvv}x)$	$I(\boldsymbol{ovo}x)$	$I(\boldsymbol{ooo}x)$	$E(vx)$	$I(vvvx)$
Balloons	−4	− 7	− 8	− 6	− 9	−10	−10	−6
Kendo	−3	− 5	− 6	− 3	− 6	− 7	− 5	−2
Newsp.	−2	− 4	− 5	− 4	− 7	− 7	− 8	−3
P.Hall2	−6	−10	−11	−13	−18	−19	− 9	−5
P.Street	−4	− 6	− 7	− 8	−12	−12	−11	−7
GTFly	−6	− 7	− 8	−10	−12	−13	− 9	−6
Shark	−5	− 7	− 8	− 7	− 9	−10	− 7	−5
UndoD.	−6	− 9	−10	−10	−14	−14	−10	−7
Mean	−5	− 7	− 8	− 8	−11	−12	− 9	−5

Table 8.6: Depth encoding time delta T_Δ; Tested: Different SV generation setups; Reference: $I(vxoo)I(vvvx)$ setup; All: *BS$_r$ off* view combination.

8.2.2.4 Extrapolation Setups

This section studies two setups using view extrapolation—the $E(vx)I(vvvx)$ setup and the $E(vx)E(vx)$ setup. The *Interp. off* test case in Table 8.3 already compared the $E(vx)E(vx)$ setup to the $I(vxoo)I(vvvx)$ setup using *Default* view combination: It leads to coding losses as it does not regard view interpolation, but also to coding gains as it reduces the hiding effect. Table 8.5 shows the depth bit rate delta of the $E(vx)E(vx)$ setup relative to the $I(vxoo)$ $I(vvvx)$ setup using *BS$_r$ off* view combination. Compared to this setup, which is less affected by the hiding effect, the $E(vx)E(vx)$ setup leads to a depth bit rate increase of about 27.1% on average over sequences and a decrease of about 4.5% for the *GtFly* and *UndoDancer* sequences.

The $E(vx)I(vvvx)$ setup uses view extrapolation and interpolation when encoding the right and the left IV depth, respectively. This could avoid the hiding effect when coding the right IV depth while maintaining coding gains by regarding view interpolation when coding the left IV depth. Table 8.5 shows that the former is achieved as the depth bit rates for *GtFly* and *UndoDancer* still decrease. The latter is not entirely achieved, as the depth bit rate increases for the natural sequences still about 25%. On average, the depth bit rate increases by 16.3%.

In conclusion, using view extrapolation for encoding of the depth of both IVs or the right IV depth reduces the hiding effect and leads to gains for the synthetic sequences and losses for the natural sequences. However, when the hiding effect is already reduced by using *BS$_r$ off* view combination, the extrapolation setups lead to depth bit rate increases of about 27% and 16%, respectively, compared to the $I(vxoo)I(vvvx)$ SV generation setup.

8.2.2.5 Depth Encoding Times and Depth Quality

Table 8.6 shows that the depth encoding time decreases for all setups between 5% and 12%. The reason for setups using view extrapolation is that they do not require operations for view

	$D_{\Delta P,M}$ [dB]							
Right IV	$I(\boldsymbol{c}xoo)$			$I(\boldsymbol{o}xoo)$			$E(vx)$	
Left IV	$I(\boldsymbol{v}v\boldsymbol{v}x)$	$I(\boldsymbol{c}v\boldsymbol{c}x)$	$I(\boldsymbol{ccc}x)$	$I(\boldsymbol{v}v\boldsymbol{v}x)$	$I(\boldsymbol{o}v\boldsymbol{o}x)$	$I(\boldsymbol{ooo}x)$	$E(vx)$	$I(vvvx)$
Balloons	+ 0.8	+ 2.1	+ 2.2	+ 1.3	+ 3.1	+ 3.1	−0.2	−0.4
Kendo	+ 1.2	+ 3.0	+ 3.1	+ 1.9	+ 4.4	+ 4.5	−0.1	−0.4
Newsp.	+ 0.5	+ 1.2	+ 1.3	+ 0.5	+ 1.7	+ 1.7	+0.1	0.0
P.Hall2	+ 2.6	+ 4.6	+ 4.7	+ 5.5	+ 8.7	+ 8.7	−1.3	−1.2
P.Street	+ 1.1	+ 2.0	+ 2.1	+ 2.9	+ 4.2	+ 4.3	−1.0	−0.1
GTFly	+10.3	+12.3	+12.3	+13.3	+16.0	+15.9	+3.8	+4.1
Shark	+ 3.2	+ 5.6	+ 5.9	+ 2.9	+ 4.7	+ 4.9	−1.7	−1.6
UndoD.	+ 5.0	+ 6.1	+ 6.2	+ 5.5	+ 6.7	+ 6.4	+2.2	+2.4
Mean	+ 3.1	+ 4.6	+ 4.7	+ 4.2	+ 6.2	+ 6.2	+0.2	+0.3

Table 8.7: Depth PSNR delta $D_{\Delta P,M}$; Tested: Different SV generation setups; Reference: $I(vxoo)I(vvvx)$ setup; All: *BS$_r$ off* view combination.

combination. For the view interpolation setups, the depth encoding time decreases as the depth distortion (see Table 8.7) decreases. More specifically, because of an improved prediction of the depth signal, the magnitude of depth changes in SVDC computation decreases and thus, as discussed in Section 5.2.2, the size of the changed SV region that the RM re-renders. The increase of the depth quality on the other hand has two reasons: The first reason is that all setups reduce the hiding effect. The second reason concerns setups that use uncoded IV texture for rendering. Compared to the coded IV textures, the uncoded IV textures include more areas with high frequent signal parts. Correct rendering from these areas requires a higher depth quality, which is consequently preserved in encoding.

8.2.3 Conclusion

This section classified and analyzed different setups to generate tested and reference SV textures in SVDC computation. Setups can be distinguished on whether they use view interpolation, view extrapolation, or a combination of both and on whether they render the SV textures from coded or uncoded texture and depth. Depending on the setup, SVDC computation neglects or regards different SVD terms, as e.g. SVDs caused jointly by texture and depth coding. Setups using uncoded texture and depth of other layers furthermore allow SVDC computation in scenarios requiring independent encoding of texture and depth, or independent encoding of IV depth maps. The evaluation of a subset of possible setups provided the following results:

- When encoding the first (i.e. the right) IV depth, interpolating $\tilde{s}'_T$ and $s'_{T,Ref}$ from the same right IV texture reduces the hiding effect and can thus decrease the depth bit rate. Highest depth bit rate reductions (10.4%) compared to the *initial encoder setup* [$I(vxoo)I(vvvx)$] can be achieved by rendering with the coded right IV texture [$I(cxoo)I(vvvx)$] while disabling background suppression [*BS$_r$ off*]. Nevertheless, interpolation using the uncoded right IV texture [$I(oxoo)I(vvvx)$] can be an option for scenarios requiring an independent en-

coding of texture and depth of the right IV.

- When encoding the second (i.e. the left) IV depth, interpolating $\tilde{s}'_T$ from already coded textures and $s'_{T,Ref}$ from uncoded IV textures $[I(\cdot x\,\cdot\cdot)I(vvvx)]$ provides the highest depth bit rate reductions and outperforms setups that use either only the coded $[I(cxoo)I(cvcx)]$ or only the uncoded IV textures $[I(oxoo)I(ovox)]$. However, coding losses due to only using uncoded IV textures are moderate (< 2%); SVDC computation is thus also suitable for scenarios with parallel encoding of texture and depth.
- When encoding the second (i.e. the left) IV depth, interpolating $\tilde{s}'_T$ from the coded right IV depth and $s'_{T,Ref}$ from the uncoded right IV depth $[I(\cdot x\,\cdot\cdot)I(\cdot v\,\cdot x)]$ provides the highest depth bit rate reductions and outperforms setups that use either only the coded $[I(cxoo)$ $I(cccx)]$ or only the uncoded $[I(oxoo)I(ooox)]$ right IV depth. Losses due to only using uncoded depth, which occur in a scenario with independent encoding of the left IV depth, are significant (> 10%).
- Using view extrapolation for encoding of the depth of both IVs $[E(vx)E(vx)]$ or the right IV depth $[E(vx)I(vvvx)]$ reduces the hiding effect and achieves gains for the synthetic sequences and losses for the natural sequences. When the hiding effect is already reduced by disabled background suppression, the two setups lead to depth bit rate increases of about 19% and 14%, respectively, compared to the $I(cxoo)I(vvvx)$ setup.

8.3 Depth Distortion Term

The initial evaluation showed that SVDC-based encoding decreases the number of blocks coded with residual and thus the depth quality. To mitigate this, [Jung 12c] proposes to consider an additional depth distortion term in RD optimization to avoid over-fitting to particular SV positions and the RM's rendering algorithm [Son 12, Jung 12b]. These aspects are further investigated in Sections 9.1 and 9.2. In contrast, this section addresses the question whether the depth distortion term achieves coding gains when the RM's rendering algorithm is also used by the receiver-side renderer, as in the default evaluation setup. For this, it discusses the encoder modifications to include the depth distortion term (Section 8.3.1), evaluates the effect of different depth distortion weights on the RD performance (Section 8.3.2), compares the RD performance of different encoder setups using the depth distortion term (Section 8.3.3), and discusses encoding complexity and the depth map quality (Section 8.3.4).

Related works: In contrast to the contributions mentioned above [Jung 12c, Son 12, Jung 12b], this section not only focuses on a scenario with matching sender- and receiver-side renderers, but also takes different view combination variants and SV generation setups into account and uses the latest HTM software and different test conditions.

8.3.1 Modifications of Distortion Computation

The test cases evaluated in this section are based on the *initial encoder setup*. This means that only the highest mode selection stage C (as defined in Table 7.1) uses a weighted average

γ	$-\infty$	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
ρ	0.0	1.0	1.5	2.0	2.8	4.0	5.8	8.0	11.3	16.0	22.8
γ	5.0	5.5	6.0	6.5	7.0	7.5	8.0	8.5	9.0	9.5	∞
ρ	32.0	45.3	64.0	90.5	128.0	181.0	256.0	362.0	512.0	724.0	∞

Table 8.8: SVDC-depth distortion ratios ρ.

of depth distortion D_M and SVDC D_S, whereas other mode selection stages use the depth distortion only. Stage C computes the combined distortion D_C as suggested in [Jung 12b], i.e.

$$D_C = (D_S \cdot \omega_S^2 + D_M \cdot \omega_M^2)/(\omega_S^2 + \omega_M^2) \tag{8.3}$$

with ω_M and ω_S denoting the depth distortion and SVDC weight, respectively. The test cases in this section use different ratios $\rho = \omega_S/\omega_M$ that have been derived as $\rho \approx 2^\gamma$ with values of γ as shown in Table 8.8.

8.3.2 Effects on Coding Performance

Figure 8.5 shows the RD performance of the *initial encoder setup* for different sequences and a subset of the evaluated ratios ρ: Using the SVDC only [i.e. ρ equal to 2^∞] outperforms all test cases with ρ less or equal to 2^2 for all sequences. Moreover, for sequences *UndoDancer*, *GtFly*, and *Shark* the ratio ρ equal to 2^4 leads to coding gains compared to SVDC-based encoding only. This raises the question why these gains can be achieved and whether they can also be achieved in the other encoder setups.

To answer these questions, this section analyzes the impact of the depth distortion term on RD performance for three encoder setups that use the *Default*, *BS$_r$ off*, and *HF$_r$ off* view combination variant, as discussed in Section 8.1.4, together with the $I(cxoo)I(vvvx)$ SV generation setup. For analysis, Figures 8.6 (a) to (c) show how depth bit rate savings in these three setups change in dependency of $\log_2(\rho)$. The references are the respective setups using the SVDC only. [Figure F.34 shows results for the $I(vxoo)I(vvvx)$ setup.]

In general, it can be observed that for all setups the depth bit rate increases by more than 5% when using weights with ρ less than $2^{1.5}$ to $2^{4.5}$, depending on the sequence. In the range of 2^2 to 2^6 the depth bit rate of most sequences shows a minimum and coding gains compared to the reference using only the SVDC. For higher values of ρ, the depth bit rate delta varies, depending on the sequence, between $+1.5\%$ and -1.5%.

Considering the evaluated setups individually, it can be observed that for *Default* view combination, the depth bit rate decreases especially for synthetic sequences. E.g. *GtFly* and *UndoDancer* show minima of about -4% at ρ equal to $2^{5.5}$ and 2^3, respectively. This indicates that depth bit rate savings are again caused by reducing the hiding effect, as these sequences are severely affected by it. Figure 8.6 (b), which shows results for *BS$_r$ off* view combination, supports this hypothesis. Depth bit rate savings for the *GtFly* and *UndoDancer* sequence are only about 0.25% and 2%, respectively. The hiding effect is already reduced by disabling back-

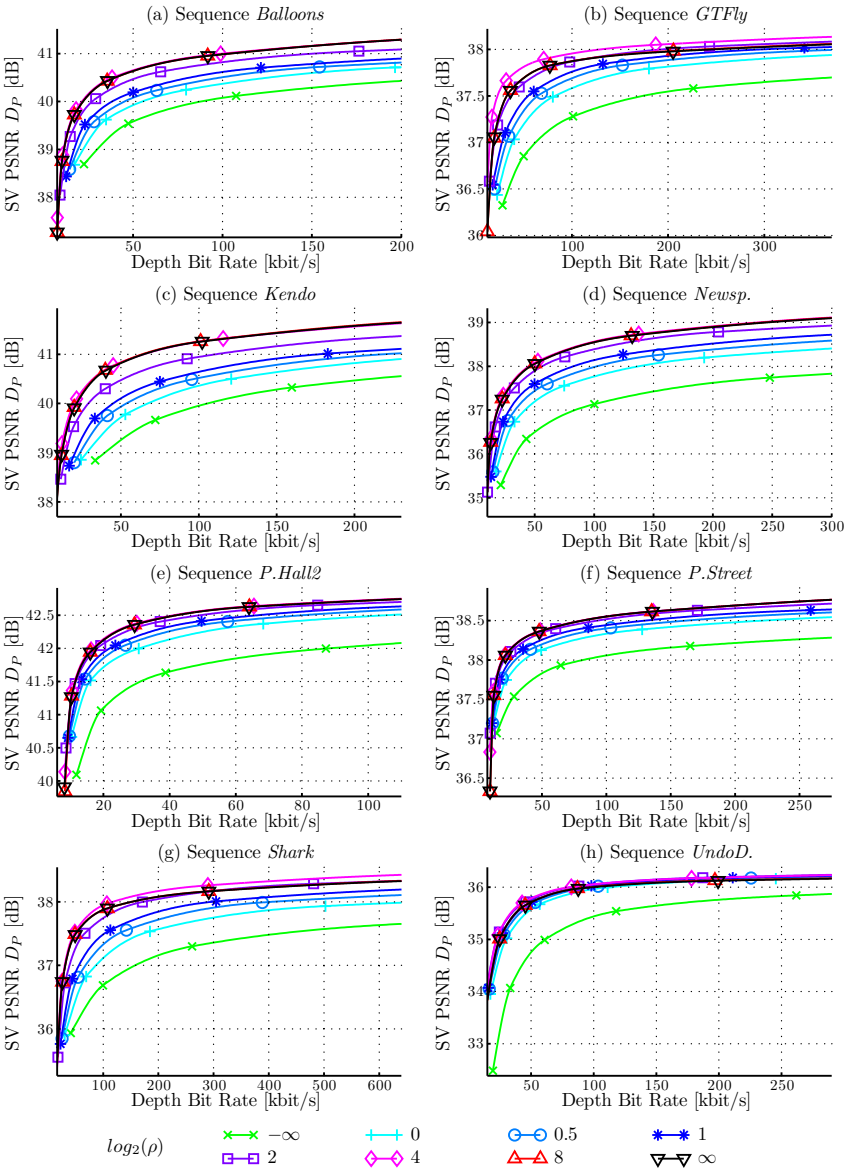


Figure 8.5: RD performance of the *initial encoder setup* using additionally a depth distortion term with different SVDC-depth distortion ratios ρ.

ground suppression; additional depth bit rate savings due to the depth distortion term are thus small. Also results for *HF$_r$ off* view combination in Figure 8.6 (c) show this behavior. Savings for the *GtFly* and the *UndoDancer* sequence are also only about 0.25% and 2%. On the other hand and in contrast to the other setups, the depth bit rate for the *PoznanHall2* sequence decreases. As this sequence suffers most from disabling hole filling (see Table 8.3), it can be concluded that the additional depth distortion term reduces coding losses caused by this.

In conclusion, the depth distortion term leads to depth bit rate savings for all three view combination variants. The amount of reductions depends on how much sequences suffer from the hiding effect or incorrect hole filling. For some sequences (e.g. *Shark*), the depth distortion term has no positive effect. Optimal ratios ρ depend on the sequence and vary from 2^3 to 2^6.

8.3.3 Comparison of Setups

So far, the evaluation compared different SVDC-depth distortion ratios while using the same SV generation setup and view combination variant. In contrast, this section compares combinations of SV generation setups and view combination variants that operate at their respective optimal depth distortion-SVDC ratios. The question which setup provides the best overall RD performance is thus answered.

Figure 8.7 and Table 8.9 show results of a comparison using the *initial encoder setup* [i.e. the $I(vxoo)I(vvvx)$ SV generation setup using *Default* view combination] as reference. Figure 8.7 shows for each setup how depth bit rate savings, averaged over the sequences, change with $\log_2(\rho)$. For each setup, Table 8.9 then summarizes the depth bit rate savings at the optimal value of ρ and when using the SVDC only. Comparing both shows that the depth distortion term does not change the rank order of test cases with respect to their RD performance. Also with the optimal ratio ρ, *BS$_r$ off* view combination performs best, followed by *HF$_r$ off* and *Default* view combination. Furthermore, considering the SV generations setups, the $I(cxoo)$ $I(vvvx)$ setup outperforms the $I(vxoo)I(vvvx)$ setup when using *Default* and *BS$_r$ off* view combination, and vice versa when using *HF$_r$ off* view combination.

A further observation can be made when relating the optimal ratios ρ to the difference of depth bit rate savings achieved by them and by using the SVDC only: The greater the difference the lower is the optimal ρ. For example, the $I(vxoo)I(vvvx)$ SV generation setup using *Default* view combination (which performs worst) achieves a difference of 8.1% with a ratio of $2^{4.5}$. In contrast, the $I(cxoo)I(vvvx)$ SV generation setup using *BS$_r$ off* view combination (which performs best) achieves a difference of 0.3% (i.e. 10.7% – 10.4%) with a ratio of $2^{5.5}$.

In conclusion, this evaluation supports the findings from Section 8.3.2: The more SVDC computation is affected by the hiding effect or by neglecting hole filling, the lower is the optimal ratio ρ and the higher are the depth bit rate reductions. However, when setups that suffer from the hiding effect or from neglecting hole filling use the depth distortion term, they do not outperform setups that address these issues already by the view combination variant or the SV generation setup. When these issues are already addressed, additional depth bit rate reductions are on average small [i.e. 0.3% more for the *BS$_r$ off* $I(cxoo)I(vvvx)$ setup].

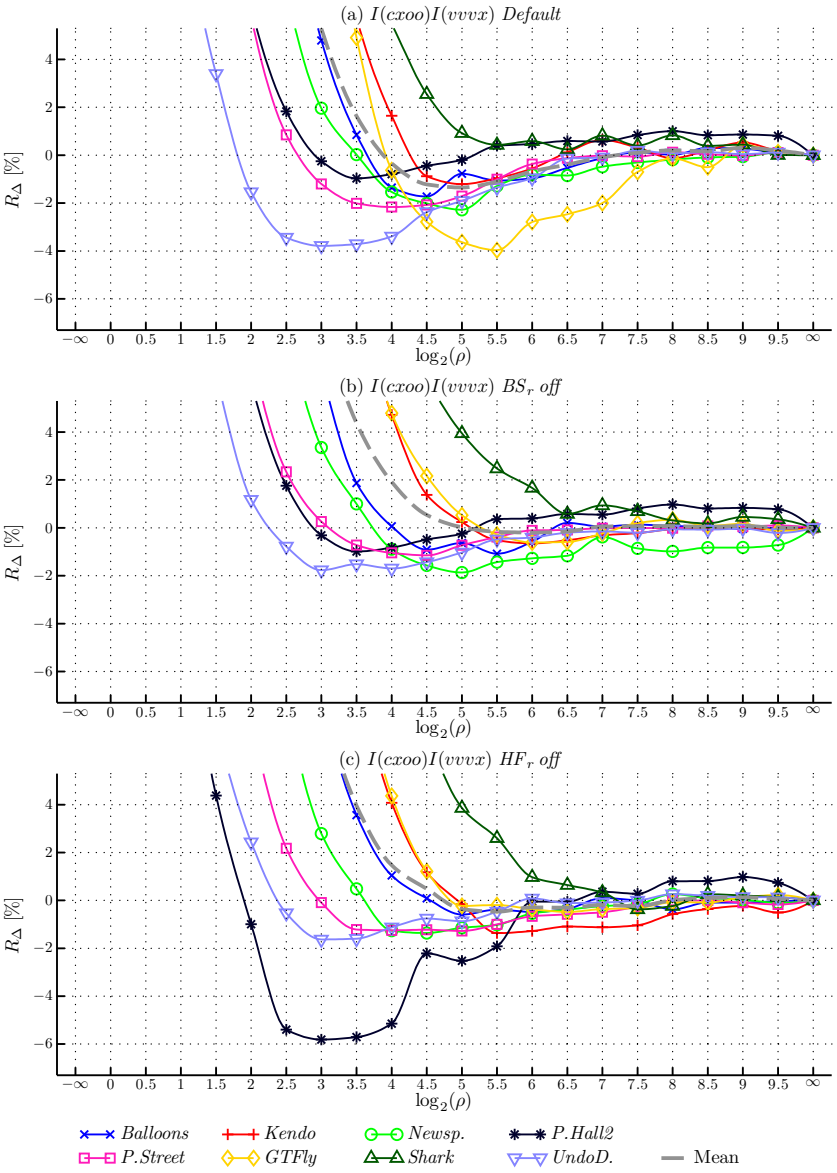


Figure 8.6: Depth bit rate savings averaged over sequences; Tested: View combination variants and ratios ρ; Reference: The respective view combination variant with ρ equal to ∞; All: $I(cxoo)I(vvvx)$ SV generation setup.

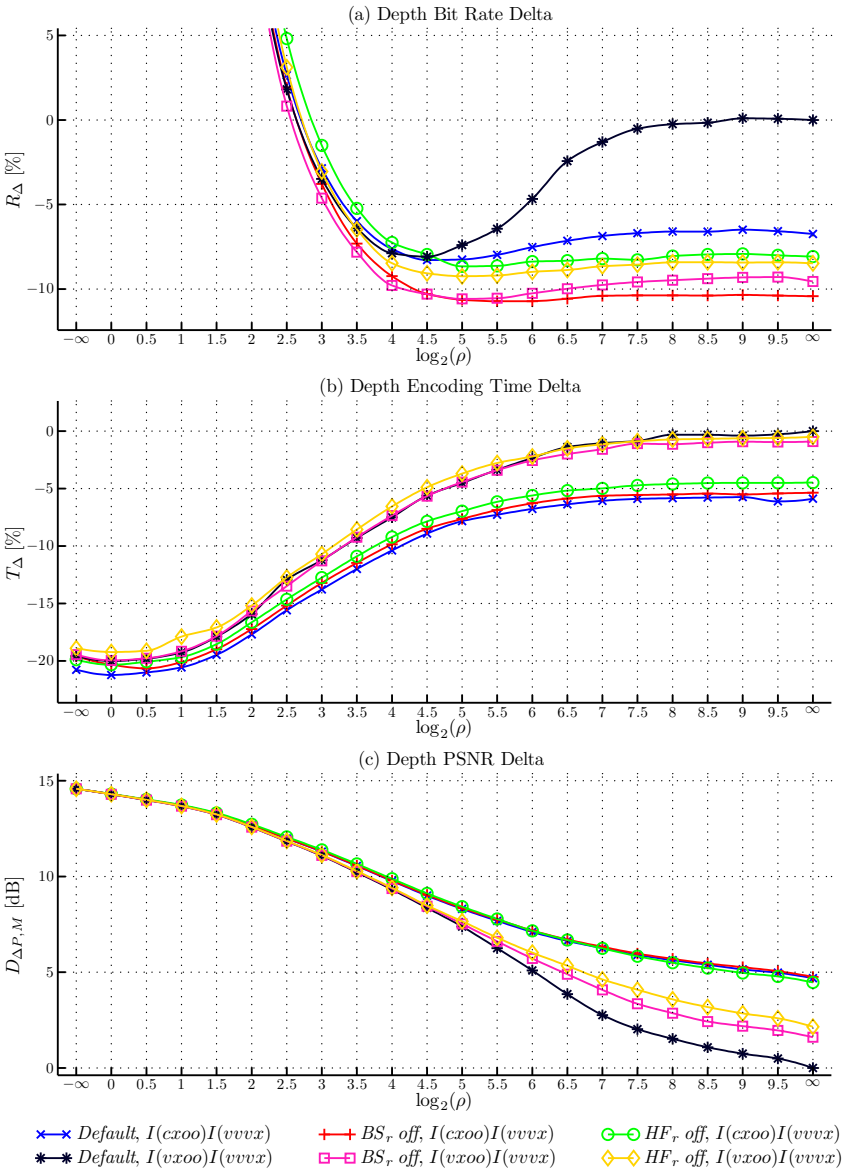


Figure 8.7: Depth bit rate, depth encoding time, and depth PSNR deltas, averaged over sequences; Tested: Different SV generation setups, view combination variants, and ratios ρ; Reference: The *initial encoder setup*.

	$R_\Delta[\%]$											
	$I(cxoo)I(vvvx)$						$I(vxoo)I(vvvx)$					
	Default		*BS$_r$ off*		*HF$_r$ off*		*Default*		*BS$_r$ off*		*HF$_r$ off*	
$\log_2(\rho)$	∞	4.5	∞	5.5	∞	5.0	∞	4.5	∞	5.0	∞	5.0
Balloons	− 0.3	− 2.0	− 2.0	− 3.0	− 0.1	− 0.6	0	− 2.4	− 2.4	− 3.6	− 0.9	− 1.2
Kendo	− 5.4	− 6.2	−10.2	−10.7	− 7.5	− 7.5	0	− 7.3	−10.5	−11.1	− 9.3	− 9.1
Newsp.	− 0.5	− 2.5	− 1.6	− 2.9	+ 1.6	+ 0.5	0	− 3.3	− 2.4	− 3.7	+ 1.1	− 0.2
P.Hall2	− 0.9	− 1.4	− 0.9	− 0.5	+14.7	+11.5	0	− 1.4	0.0	− 1.3	+17.2	+13.2
P.Street	− 1.5	− 3.5	− 3.4	− 3.7	− 1.0	− 1.9	0	− 3.6	− 2.9	− 3.8	− 1.3	− 2.3
GTFly	−23.5	−28.8	−30.5	−31.6	−32.5	−33.6	0	−28.7	−27.4	−30.7	−32.6	−34.5
Shark	−12.5	−10.2	−22.1	−20.2	−25.0	−22.3	0	− 7.9	−20.0	−18.6	−28.0	−25.1
UndoD.	− 9.4	−11.8	−12.7	−13.1	−14.7	−15.4	0	−10.1	−10.8	−12.0	−13.9	−14.8
Mean	− 6.7	− 8.3	−10.4	−10.7	− 8.1	− 8.7	0	− 8.1	− 9.6	−10.6	− 8.5	− 9.2

Table 8.9: Depth bit rate delta R_Δ; Tested: Different SV generation setups, view combination variants, and ratios ρ; Reference: The *initial encoder setup*.

8.3.4 Encoding Complexity and Depth Quality

The computation of the depth distortion and the weighted average requires additional operations. To analyze how this affects encoding complexity, Figure 8.7 compares the depth encoding times of the evaluated setups. The reference is the $I(cxoo)I(vvvx)$ SV generation setup using *Default* view combination and the SVDC only. Other than expected, depth encoding times decrease when using the depth distortion term. For the ratio ρ equal to 2^0 depth encoding times even drop below those obtained for the ratio $2^{-\infty}$ (for which the SVDC is computed, but not considered). Then for values of ρ greater than 2^1, they increase, but are still lower than the depth encoding times for using the SVDC only.

Comparing Figures 8.7 (b) and (c) shows a possible reason for the latter. The depth encoding time increases with decreasing depth PSNR gains. This indicates that the improved depth map quality leads to a better prediction, a decreased depth distortion when testing modes, and thus to a smaller changed SV region that needs to be re-rendered (see Section 8.2.2.5). Reason for the initial drop is, however, that when the SVDC is additionally regarded, the measured distortion decreases for blocks in that the depth quality is less relevant for the SV quality. Consequently, the encoder employs more often mechanisms that enforce an early mode selection, i.e. when the distortion is low for a mode tested early, the encoder selects it without testing of all possibilities.

In summary, at ratios that provide the maximal depth bit rate reductions (i.e. ρ about 2^5), the depth encoding time decreases by 4% to 8% while the depth PSNR increases by 6 dB to 9 dB.

8.3.5 Conclusion

This section showed that a depth distortion term reduces the depth bit rate only significantly (8.1% in average) when SVDC computation is affected by the hiding effect. In setups address-

ing this issue already by modified view combination or SV generation, it leads in average to only small additional depth bit rate reductions (less than 0.5%). Optimal SVDC-depth distortion ratios are between $2^{4.5}$ and $2^{5.5}$, in which the higher ratio is optimal in setups in that SVDC computation is less affected. Although depth distortion computation requires additional operations, depth encoding times decrease by 4% to 8%. Reason for this is that the depth PSNR increases by 6 dB to 9 dB, which reduces the rendering overhead due to the magnitude of the depth change. The depth distortion term is thus also beneficial in cases in that the receiver uses the RM's rendering algorithm. Whether it is beneficial when different renderers or SV positions are used, as intended in [Jung 12b, Jung 12c], is discussed in Sections 9.1 and 9.2.

In comparison to the *initial encoder setup*, the test case using *BS_r off* view combination, the $I(cxoo)I(vvvx)$ SV generation setup, and a ratio ρ equal to $2^{5.5}$ achieves with 10.7% the highest depth bit rate savings among all test cases. For this, following sections will use this combination—the *optimized encoder setup*—as basis for further evaluations.

8.4 Mode Selection Stage Setups

Encoders evaluated so far employed SVDC computation only in the highest mode selection stage C. Lower stages decided based on the depth distortion. This raises the questions whether using the SVDC also in these stages achieves further coding gains and which additional complexity is introduced by this. This section addresses SVDC-based encoding of the residual quadtree of inter- and intra-picture predicted CBs (Section 8.4.1); analyzes SVDC-based (pre-) selection of intra- and inter-picture prediction modes, wedgelet parameters, and motion parameters (Section 8.4.2); and explores test cases in that the SVDC is used jointly in different mode selection stages for encoding of intra- and inter-picture predicted CBs (Section 8.4.3).

Related works: In the own contributions [Tech 12c, Schw 11a] the RM was not applied for motion estimation. [Oh 12b, Wang 12c] replace the SVDC by estimated SVDs in several lower-level selection stages. In contrast to these papers, this section provides a comprehensive and systematical evaluation considering the replacement of the SVDC with the depth distortion.

8.4.1 Residual Quadtree Encoding

As discussed in Section 7.1, the encoder used in this thesis encodes the residual quadtree similarly for inter- and intra-picture predicted CBs. Related processes decide how to split the residual quadtree in TBs, whether to signal transform coefficients, and whether to apply RDOQ. In the G0 setup, they decide based on the depth distortion. To analyze effects when they consider the SVDC, this section discusses mode selection setups A1, A2, E1, and E2 as defined in Table 7.1. All test cases are based on the *optimized encoder setup*.

Setups A1 and E1 extend the G0 setup by using the SVDC also to decide whether to split the RQT and whether to signal transform coefficients for the TBs (Table 7.1, Q and R). Table 8.10 shows that the A1 setup achieves depending on the sequence minor depth bit rate savings and

	$R_\Delta[\%]$												
	A (Intra)				E (Inter)				G (Combined)				
	1	2	3	4	1	2	3	4	1	2	3	3D	4
Balloons	+0.1	+0.2	0.0	−0.8	−1.2	−1.9	−1.8	−5.4	−1.4	−1.6	−2.5	−2.8	−6.0
Kendo	−0.1	−0.4	−0.7	−1.4	−1.4	−1.5	−2.2	−7.8	−1.3	−1.7	−3.1	−3.4	−8.7
Newsp.	+0.2	+0.1	−0.8	−1.7	−1.5	−1.9	−2.2	−6.5	−1.6	−2.0	−3.3	−4.2	−8.2
P.Hall2	−0.1	−0.4	−1.0	−0.2	−0.7	−1.1	−1.3	−4.1	−0.8	−1.4	−1.7	−1.2	−4.2
P.Street	−0.1	0.0	−1.4	−1.9	−0.7	−0.8	−1.0	−3.2	−0.8	−0.7	−2.0	−2.7	−4.5
GTFly	+0.1	−0.1	−0.8	−1.1	−0.6	−0.7	−1.5	−6.4	−0.4	−1.0	−1.6	−2.5	−7.9
Shark	0.0	−0.1	−1.0	−1.3	−1.5	−1.2	−2.4	−5.4	−1.4	−1.4	−2.8	−3.4	−6.4
UndoD.	−0.2	−0.4	−1.5	−1.9	−1.0	−1.3	−1.6	−5.7	−1.2	−1.5	−3.2	−3.1	−7.2
Mean	0.0	−0.1	−0.9	−1.3	−1.1	−1.3	−1.8	−5.6	−1.1	−1.4	−2.5	−2.9	−6.6

Table 8.10: Depth bit rate delta R_Δ; Tested: Different mode selection setups; Reference: The *optimized encoder setup* (using the G0 setup).

	$T_\Delta[\%]$												
	A (Intra)				E (Inter)				G (Combined)				
	1	2	3	4	1	2	3	4	1	2	3	3D	4
Balloons	+3	+ 9	+13	+35	−4	+ 4	+ 8	+288	− 2	+12	+20	+41	+319
Kendo	+2	+ 8	+12	+32	−5	+ 4	+ 8	+302	− 3	+12	+19	+40	+335
Newsp.	+3	+ 9	+14	+40	−3	+ 7	+11	+304	0	+16	+24	+50	+344
P.Hall2	+3	+10	+15	+28	−3	+ 6	+ 9	+244	+ 1	+16	+24	+36	+274
P.Street	+3	+10	+15	+36	0	+ 9	+12	+280	+ 3	+18	+26	+47	+316
GTFly	+2	+ 8	+12	+34	−2	+ 8	+13	+328	0	+16	+25	+47	+352
Shark	+3	+ 9	+14	+36	+1	+12	+17	+372	+ 4	+21	+30	+52	+403
UndoD.	+3	+10	+14	+33	+7	+18	+23	+436	+10	+27	+37	+56	+464
Mean	+3	+ 9	+14	+34	−1	+ 9	+13	+319	+ 1	+17	+26	+46	+351

Table 8.11: Depth encoding time delta T_Δ; Tested: Different mode selection setups; Reference: The *optimized encoder setup* (using the G0 setup).

increases (both less than 0.3%). In contrast, the E1 setup reduces the depth bit rate in average over sequences by 1.1%. A potential explanation for this difference is that in comparison to depth distortion-based encoding, SVDC-based residual encoding modifies the depth such that it not only produces locally a higher SV quality, but also is a worse intra-picture predictor.

The A1 and E1 setups perform additional operations for SVDC computation. Furthermore, after the encoder finalized a part of the RQT, it needs to invoke the RM in *SET* mode to update its state. Table 8.11 shows that this leads for the A1 setup to a depth encoding time increase of about 3% consistently for all sequences. In contrast, depth encoding times in the E1 setup decrease for some sequences and increase for others. An explanation for this is the early mode selection the encoder performs for CBs with $2N\times2N$ partition size that are coded in merge mode: When the RQT encoding stage decides to skip transform coefficients entirely for such a CB, the encoder selects its merge candidate without testing others. As this case occurs more of-

ten for some sequences when using the SVDC in stage E1, their depth encoding times decrease despite of the additional operations required for SVDC computation.

The A2 and E2 setups extend the A1 and E1 setups, respectively, by an SVDC-based selection between a) transform coefficients coded without RDOQ and b) transform coefficients coded with RDOQ using the depth distortion. As discussed in Section 7.1, reason for these options is that SVDC computation is not directly possible in the transform domain. When the encoder can choose between the two options, depth bit rate savings increase from 0% to 0.1% for the A2 setup and 1.1% to 1.3% in the E2 setup while the depth encoding time increases from 3% to 9% and -1% to 9%, respectively. Thus, additional depth bit rate savings are rather minor considering the additional depth encoding time increase (which is in the E2 setup higher, as more CBs are inter-picture predicted).

In summary, for intra-picture predicted CBs, SVDC-based encoding of the RQT leads only to depth bit rate reductions for some sequences, which indicates that the coded depth becomes less suitable as intra-picture predictor. For inter-picture predicted CBs, the depth bit rate and the depth encoding time decrease by 1.1% and 1%, respectively. In both cases, enabling the encoder additionally to select transform coefficients quantized without RDOQ results in only small additional depth bit rate savings (about 0.2%), although the depth encoding time increases additionally by 6% and 10%, respectively.

8.4.2 Selection of Prediction Modes and Parameters

In the G0 setup (and the A1 and A2 setups), the encoder selects the final intra-picture prediction mode based on the SVDC from a set of pre-selected modes. However, the processes for pre-selection (stage P) and also for the selection of parameters for the wedglet modes (stage D) use the depth distortion. The A3 setup extends the A2 setup by using the SVDC in stage P to pre-select the intra-picture prediction mode. This leads to an increase of depth bit rate savings from 0.1% to 0.9% and depth encoding time from 9% to 14% (see Tables 8.10 and 8.11). The A4 setup extends the A3 setup by additionally selecting the wedgelet parameters in stage D based on the SVDC, so depth bit rate savings increase further from 0.9% to 1.3% and depth encoding times from 14% to 34%. Thus, compared to the A3 setup, additional depth bit rate savings in the A4 setup are rather minor considering the significant depth encoding time increase.

In encoding of inter-picture predicted CBs, the G0 setup (and the E1 and E2 setups) employ the depth distortion to select the motion derivation mode (i.e. merge or AMVP) and the merge candidate when the CB partitioning is not $2N\times2N$ (stage I), and to determine the motion parameters (stage M), i.e. to select between uni- and bi-directional prediction, reference pictures, motion predictor candidates, and (in motion estimation) motion vectors.

The E3 setup extends the E2 setup by using the SVDC also in stage I to select the motion derivation mode and the merge candidate. Tables 8.10 and 8.11 show that this increases the depth bit rate savings from 1.3% to 1.8% and the depth encoding time delta from 9% to 13%. Furthermore, the E4 setup extends the E3 setup by selecting the motion parameters in stage M based on the SVDC: Depth bit rate savings and the depth encoding time delta further increase

from 1.8% to 5.6% and 13% to 319%, respectively. Thus, although additional depth bit rate reductions are high, the significant increase of depth encoding time makes it questionable whether the E4 setup is suitable in practical use cases, unless they allow massive parallelization.

In summary, the SVDC-based pre-selection of the intra-picture prediction mode (stage P) and the SVDC-based selection of the motion derivation mode and the merge candidate for all CB partitionings (stage I) increase the depth bit rate savings moderately (additional 0.8% and 0.5% of the G0 depth bit rate, respectively) considering the depth encoding time increases (additional 5% and 4% of the G0 depth encoding time, respectively). In contrast, SVDC-based wedgelet (stage D) and motion parameter (stage M) selection leads to depth bit rate savings (additional 0.4% and 3.8% of the G0 depth bit rate, respectively), which are not justified by the depth encoding time increase (additional 20% and 306% of the G0 depth encoding time, respectively).

8.4.3 Combined Setups

Setups evaluated so far extend the G0 setup by using the SVDC for encoding of either inter-picture predicted CBs or intra-picture predicted CBs. Tables 8.10 and 8.11 show further results for setups G1, G2, G3, G3D, and G4, which use the SVDC for encoding of both types of CBs. For n in the range of 1 to 4, inclusive, setup Gn combines changes of setups An and En (see Table 7.1). An exception is the G3D setup, which corresponds to setup E3, but additionally selects wedgelet parameters in stage D based on the SVDC.

Tables 8.10 and 8.11 show that depth bit rate savings and depth encoding time deltas are approximately additive. The G1 setup achieves depth bit rate saving of 1.1% with a depth encoding time increase of 1% by using the SVDC also for RQT encoding. It is thus a reasonable alternative to the G0 setup when requiring low computational complexity. On the other hand, setup G4 can be an alternative for scenarios allowing a high computational complexity. It uses the SVDC in all selection stages (but for transform coefficients coded using RDOQ, which can, however, be discarded based on the SVDC) and achieves depth bit rate savings of 6.6% while the depth encoding time increases by 351%. Setups G2, G3, and G3D allow trade-offs between both.

8.4.4 Conclusion

This section explored the application of SVDC-based encoding in lower-level decision stages. Using the SVDC for encoding of the RQT of inter-picture predicted CBs decreases not only the depth bit rate by 1.1%, but also the depth encoding time by 1% because of an early mode selection method. For intra-picture predicted CBs, however, the depth encoding time increases by 3%, but no depth bit rate savings are achieved. Enabling the encoder additionally to select transform coefficients quantized without RDOQ leads to only small additional depth bit rate savings (about 0.2%); the depth encoding time further increases by 6% for intra-picture predicted CBs and 10% for inter-picture predicted CBs.

Whereas using the SVDC also for pre-selection of intra-picture predictions modes provides additional depth bit rate savings of 0.8% with a modest depth encoding time increase of 5%, the

selection of wedgelet parameters is computational complex (additionally 20% depth encoding time increase), and leads to only small additional bit rate savings of 0.4%. Using the SVDC also for all CB partitionings to select the motion derivation mode and the merge candidate increases depth bit rate savings by additionally 0.5% and the depth encoding time delta by additionally 4%. Using it on the other hand for determination of motion parameters provides with 3.8% the highest additional depth bit rate savings, but increases the depth encoding time by 306%.

In summary, SVDC-based encoding in all mode selection stages increases depth bit rate savings by 6.6% and the depth encoding time by a factor of about 3.5. Nevertheless, this section's findings show how other trade-offs between depth bit rate and encoding complexity can be achieved, e.g. to adapt to a given application scenario.

8.5 Lagrange Multiplier and QP Selection

For complexity reasons, the encoder evaluated in this thesis employs two distortion metrics in rate-distortion optimization: The SVDC in the highest mode selection stage C and the depth distortion in lower-level selection stages. Considering that SVDC-based mode selection uses a Lagrange multiplier λ_S, the question is how to optimally choose the quantization parameter (QP) and the Lagrange multiplier λ_M for depth distortion-based mode selection. This section addresses this question under the constraint that the three parameters are constant for the entire depth sequence. It derives their theoretical relationship (Section 8.5.1), determines optimal combinations experimentally (Section 8.5.2), and discusses the results (Section 8.5.3).

Related works: The Lagrange multiplier method is addressed in [Ever 63, Sull 98] (see Section 8.5). Furthermore, several papers suggesting SVD estimation methods, e.g. [Zhan 14, Yuan 14, Kim 10], analyze theoretically how to select λ_S or the QP based on their metrics. The analysis in Section 8.5.1 is similar, but regards additionally that the lower-level decision stages use the depth distortion and λ_M. An experimental evaluation of the relationship between λ_S and λ_M can be found in [Fank 12] and the own contribution [Tech 12c]. Differences to these works are that all three parameters—QP, λ_S, and λ_M—are regarded and that the depth distortion term is considered in Section 8.5.2. Furthermore, the own contribution [Tech 12b] uses the Lagrange multiplier to optimize the bit rate allocation between texture and depth, which is not intended in this section (and by purpose avoided by the test conditions).

8.5.1 Theoretical Background

This section derives the relationship between the three parameters discussed above, i.e. λ_S, λ_M, and the QP. More specifically, it discusses the case in that selection stage C uses the SVDC and the Lagrange multiplier λ_S, whereas other selection stages use the depth distortion and the Lagrange multiplier λ_M. Target is to derive λ_M and the depth QP such that the receiver-side RD performance is maximized.

The derivation of λ_M and the QP is based on two assumptions: The first is that the encoder

achieves the highest RD performance with respect to the receiver-side SVD when it minimizes the SVDC-based RD cost that is observable for the modes finally selected in stage C. The second assumption is that this is given when the encoder minimizes the SVDC-based RD cost J_S observable for the modes pre-selected by the lower-level selection stages, i.e.

$$J_S = D_S + \lambda_S \cdot R \tag{8.4}$$

with D_S representing the SVDC and R the total depth bit rate of the pre-selected modes. The conventional approach for minimizing (8.4) is to determine the root of its derivative with respect to R which leads to

$$\lambda_S = -\frac{\mathrm{d}D_S}{\mathrm{d}R}. \tag{8.5}$$

Assuming that the total SVDC D_S and the total depth bit rate R are differentiable functions of the total depth distortion D_M and considering that $\mathrm{d}D_M/\mathrm{d}R$ is equal to $-\lambda_M$ when the encoder selects the modes in lower-level selection stages based on the depth distortion and the Lagrange multiplier λ_M leads to

$$\lambda_S = -\frac{\mathrm{d}D_S}{\mathrm{d}D_M} \cdot \frac{\mathrm{d}D_M}{\mathrm{d}R} = \frac{\mathrm{d}D_S}{\mathrm{d}D_M} \cdot \lambda_M, \tag{8.6}$$

which can be written as

$$\lambda_M = \lambda_S \cdot \left(\frac{\mathrm{d}D_S}{\mathrm{d}D_M} \right)^{-1} = \lambda_S \cdot l_s \tag{8.7}$$

$$l_s = \left(\frac{\mathrm{d}D_S}{\mathrm{d}D_M} \right)^{-1}. \tag{8.8}$$

λ_M and λ_S are connected by the derivative of the total SVDC D_S with respect to the total depth distortion D_M. When not using the SVDC, but the combined distortion D_C including the depth distortion, a respective factor can be found by substituting D_S in (8.8) with D_C from (8.3).

The associated quantization step size q can be derived from (8.7) by assuming in analogy to [Sull 98] that R is given as function of D_M (6.5) and D_M as function of q (6.6). This leads to

$$q^2 = 12 \cdot a \cdot \lambda_S \cdot \left(\frac{\mathrm{d}D_S}{\mathrm{d}D_M} \right)^{-1} = 12 \cdot a \cdot \lambda_M. \tag{8.9}$$

Comparing (8.9) to (6.4) shows that when the Lagrange multiplier λ_M is selected such that it satisfies (8.7), the optimal quantization step size is given by the relationship as derived in [Sull 98]. Consequently, the depth QP can be chosen considering (6.7) with λ equal to λ_M.

8.5.2 Experimental Lagrange Multiplier and QP Selection

The last section gave a qualitative insight how the Lagrange multipliers λ_S and λ_M, and the QP are connected. This section discusses two methods to derive the optimal values experimentally. The first method is a coding tree block level search similar to the approach used in [Sull 98]. The second method is a full search on sequence level. Evaluations for both methods use the

optimized encoder setup, but for simplification an IPPP prediction structure, i.e. only the first picture is coded as intra-picture, whereas the following pictures can only refer to their respective preceding pictures. This way, a complex optimization of the bit rate allocation to different pictures, which would be required for hierarchical prediction structures, is not necessary.

8.5.2.1 Coding Tree Block-Based Search

As discussed in Section 6.2.1, [Sull 98] performs a search on the coding block level while encoding to find the optimal combination of the QP and the Lagrange multiplier in texture coding. The encoder operates with a fixed value of λ and is allowed to choose the QPs in a certain range around the QP value used by the last CB. Finally, the most frequent chosen QP for a particular λ value is selected for the entire sequence.

A similar method is applied in this thesis to determine the QP and λ_M for fixed values of λ_S: For each CTB the encoder evaluates nine different combinations of QP and λ_M that are given by the Cartesian product of $\{\mathrm{QP}_l - N,\ \mathrm{QP}_l,\ \mathrm{QP}_l + N\}$ and $\{\lambda_{M,\mathrm{l}} \cdot 2^{-N/3},\ \lambda_{M,\mathrm{l}},\ \lambda_{M,\mathrm{l}} \cdot 2^{N/3}\}$ with $\lambda_{M,\mathrm{l}}$ and QP_l denoting the values selected for the last CTB and N denoting the current step size. After testing all nine combinations for a CTB, the encoder selects the combination that achieves the minimal RD cost $J_S(\lambda_M, \mathrm{QP})$ while neglecting bits for signalling the delta QP value. When all combinations provide the same RD cost, the encoder reuses the last combination. To allow a fast convergence, the encoder starts with the step size N equal to 4 and then halves N after the 16-th and 32-nd picture of the sequence. The initial value for λ_M is λ_S; the initial QP corresponds to the default QP, which is given by (6.7) with l_m equal to 0.4625.

Figure 8.8 shows results of the evaluation for different values of λ_S and different sequences[4], i.e. histograms of the relative occurrence of different values of $\log_2(l_s) = \log_2(\lambda_M/\lambda_S)$ and QP values. The histograms consider only CTBs that actually signal transform coefficients or a mode selected based on λ_M, i.e. only CTBs that directly depend on the evaluated parameters. The following can be observed:

- The value range of relative occurrences shows that only a small fraction of CTBs depends on λ_M and the QP. Summing up all values leads to total values of about 16%, 8%, 3%, and 1% (in average over sequences) for the lowest to the highest shown λ_S value, respectively. Consequently, the smoothness of the histograms decreases with increasing λ_S values.
- Regarding the most frequent QP-l_s combinations, it can be furthermore observed for increasing λ_S values that the QP increases and that $\log_2(l_s)$ is constant at a value of about 0, which means that λ_M also increases. Both observations support the theoretical analysis in Section 8.5.1 qualitatively.
- At low depth bit rates, i.e. λ_S equal to 8.55, local maxima occur rather randomly, which might be caused by the low number of affected blocks. For medium depth bit rates, i.e. λ_S equal to 5.89 and 3.22, the histograms change such that they have rather a distinct single local maximum. At high depth bit rates, i.e. λ_S equal to 0.59, some sequences, however,

[4] Further evaluations showed that different initial QP and λ_M values lead to similar results, with an exception for λ_S equal to 8.55, which does not converge when starting e.g. from QP 0 and $\log_2(l_s)$ equal to -10.

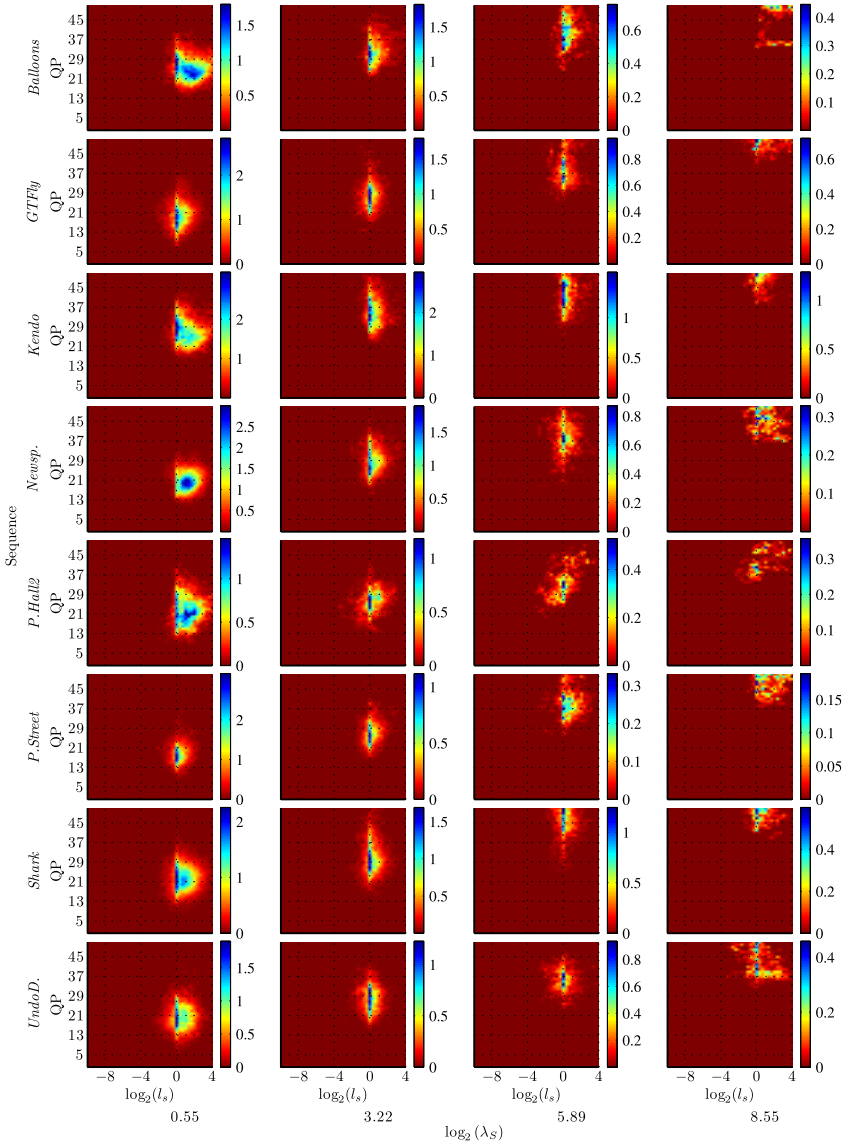


Figure 8.8: Relative occurrence (in ‰) of CTBs coded with different l_s-QP combinations for different Lagrange multipliers λ_S; Both views; λ_S values correspond to QP values of 17, 25, 33, and 41 according to (6.7).

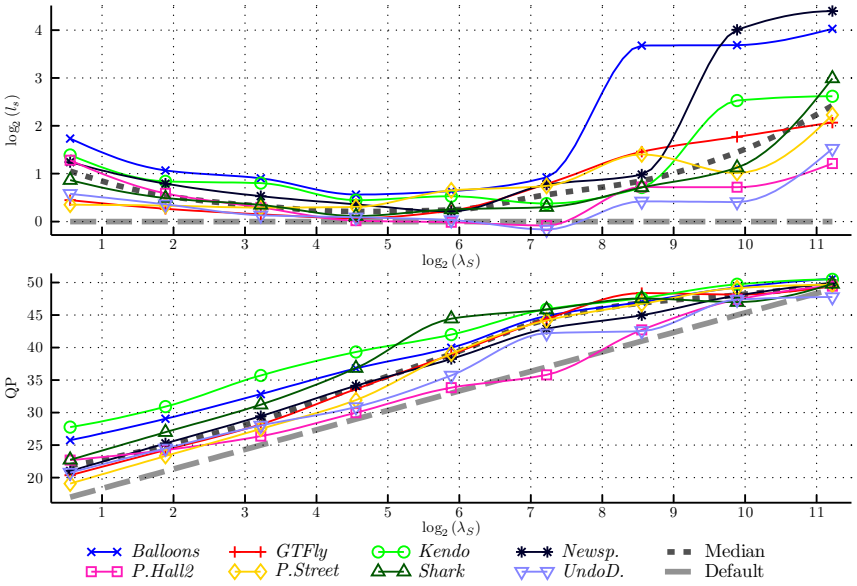


Figure 8.9: Optimal combinations of the QP and l_s for different λ_S derived by the CTB-based search; Center of mass.

show two distinct local maxima. These two local maxima occur also when considering views individually (see Figures F.35 and F.36). They are thus not caused by superposition of different characteristics of views and indicate that there is not a single optimal combination. This becomes also clear e.g. from (8.8): Considering $\mathrm{d}D_S/\mathrm{d}D_M$ not in total, but locally, it might significantly depend on local signal characteristics.

Figures 8.9 and 8.10 show for each of the evaluated λ_S values a QP-λ_M combination that will be further evaluated: The combinations in Figure 8.10 correspond to the actual maxima of the histograms; combinations in Figure 8.9 correspond to the centers of mass in Figure 8.8. It can be observed that optimal QP and l_s values change over λ_S qualitatively similar to the default combinations [i.e. $\log_2(l_s)$ equal to zero and the QP according to (6.7)]. However, at high λ_S values, l_s increases significantly. Reason for this might be a compensation for that the QP cannot exceed the value of 51: This means that l_s increases to satisfy (8.5) for a non-optimal selection of q. Figures 8.9 and 8.10 show furthermore medians of the sequences' optimal l_s and QP values. They are (apart from median l_s values obtained from the histograms' maxima) higher than the default values. This indicates [with (8.8)] that $\mathrm{d}D_S/\mathrm{d}D_M$ is less than 1.

In summary, the CTB-based encoder search provides two sets of λ_M and QP combinations that could be optimal and that will be further evaluated in Section 8.5.2.3.

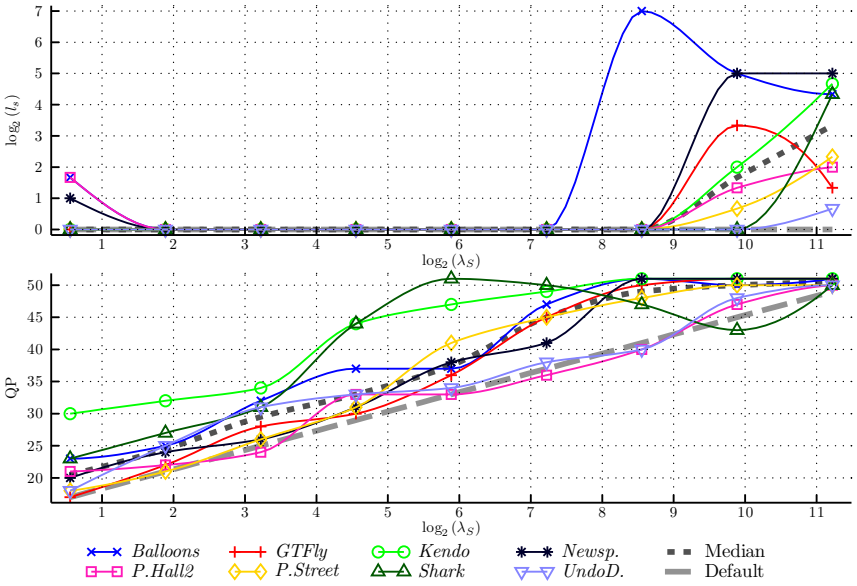


Figure 8.10: Optimal combinations of the QP and l_s for different λ_S derived by the CTB-based search; Maximum.

8.5.2.2 Sequence-Based Full Search

The assumption behind the CTB-based search is that using the λ_S-λ_M-QP combination that is most frequently selected will also minimize the receiver-side observable RD cost. To explore whether this is true, this section presents results of a sequence-based full search. The full search considers the receiver-side RD performance of different λ_S-λ_M-QP combinations. For this, it samples the space spanned by the three parameters in a simulation to determine SV PSNRs and depth bit rates. Evaluated points correspond to the combinations of the following values:

- $\log_2(\lambda_S) \in \{0.55, 1.88, 3.22, 4.55, 5.89, 7.22, 8.55, 9.89, 11.22\}$ [5]
- $\lambda_M = \lambda_S \cdot l_s$ with $\log_2(l_s) \in \{-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4\}$
- QP $\in \{1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49\}$

The SV texture PSNRs D_P achieved at $\log_2(\lambda_S) \in \{0.55, 3.22, 5.89, 8.55\}$ are shown in Figure 8.11. They have similar characteristics as the histograms of the CTB-based search: For high rates, D_P is a relative smooth function over QP and $\log_2(l_s)$ with a distinct maximum, while smoothness and distinction of the maximum decrease with increasing λ_S, as the number of blocks that depend on l_s and the QP decreases. Their impact on the overall RD performance

[5] The values of λ_S have been derived by (6.7) with $l_m = 0.4625$ and QP values 17, 21, 25, 29, 33, 37, 41, 45, and 49. They are thus the default Lagrange multipliers of the HTM software at these QPs.

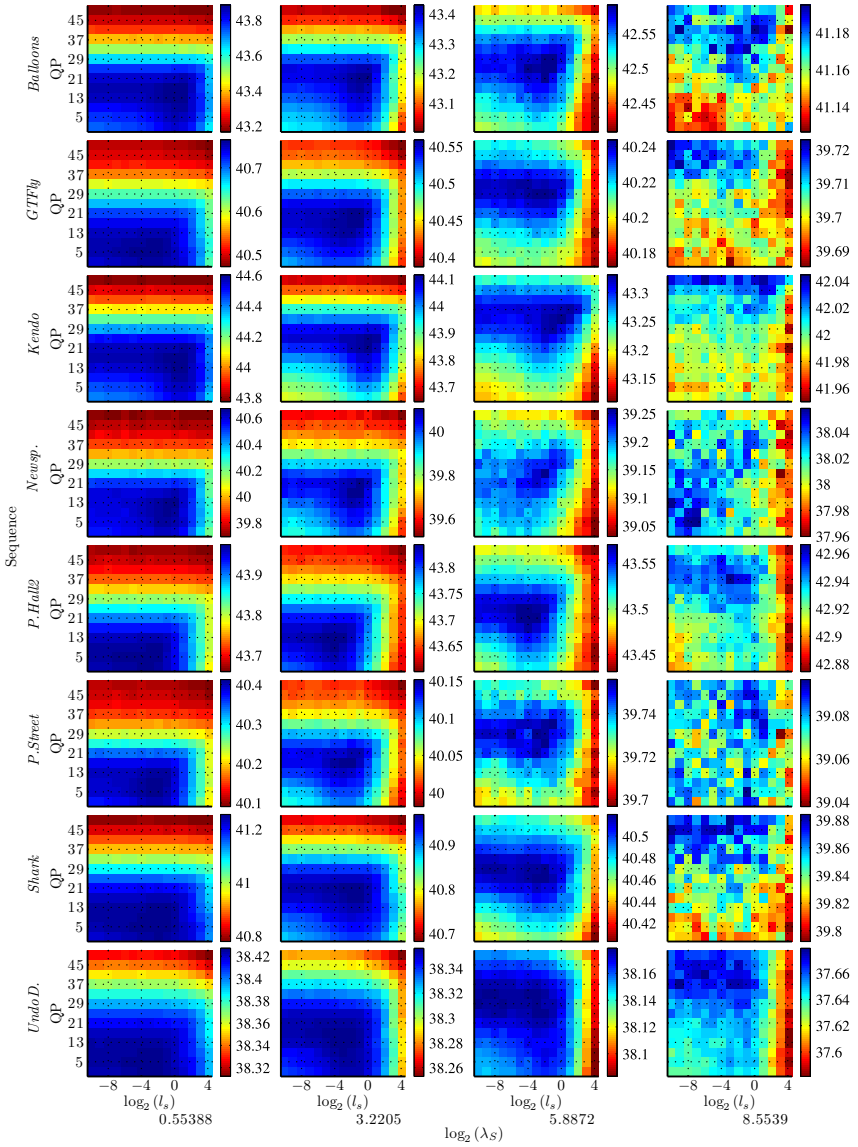


Figure 8.11: SV texture PSNR D_P [dB] in dependence of λ_S, l_s, and the QP; Full Sequence Search.

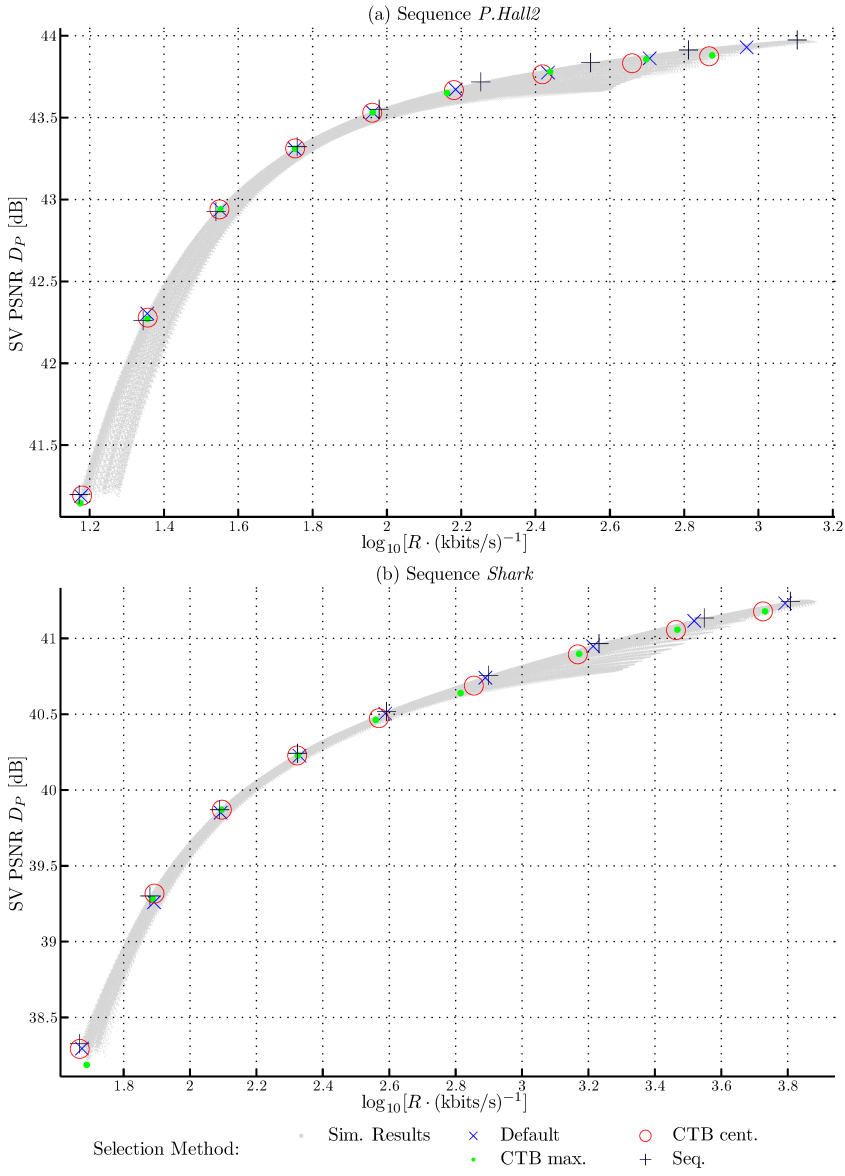


Figure 8.12: RD performance for different λ_S, λ_M, QP combinations; Gray: Interpolated from full search.

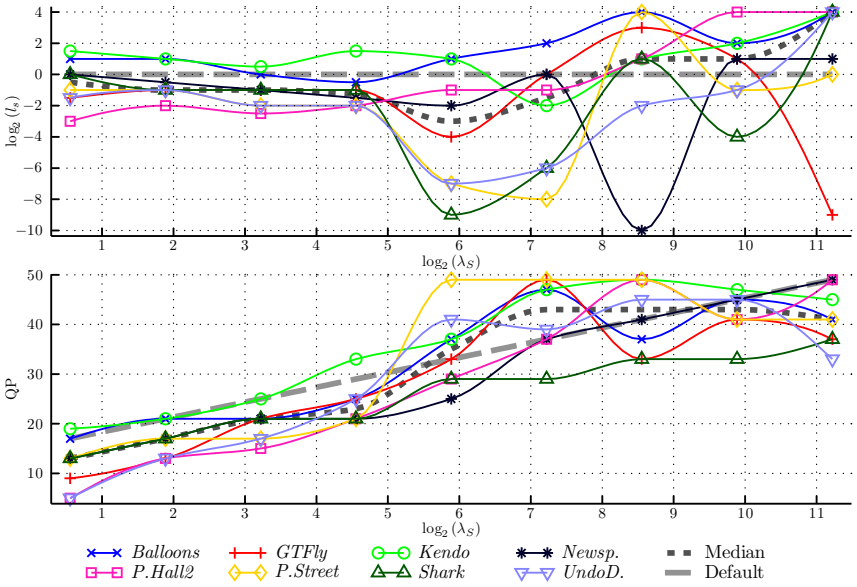


Figure 8.13: Optimal combinations of the QP and l_s for different λ_S (Sequence-based search).

becomes thus rather random. Similar observations can also be made in Figure F.37 for the obtained depth bit rates.

Based on the obtained SV PSNRs and depth bit rates, the full search determines the optimal λ_S-λ_M-QP combination in two steps: To fill gaps between the obtained RD points, the first step interpolates the SV texture PSNRs and depth bit rates at additional points in the λ_S-λ_M-QP space. For this, it applies cubic interpolation to increase the sampling rates of λ_S, λ_M, and the QP by factors of 64, 2, and 2, respectively. The second step determines λ_S, λ_M, and QP values on the hull of all sampled and interpolated RD points (by a Lagrangian approach).

As example, Figure 8.12 shows the interpolated and measured RD points and the RD points that the second step has selected for the evaluated λ_S values (Seq.). (Figures F.38 to F.40 show results for the other sequences.) Furthermore, Figure 8.13 summarizes the optimal combinations. Similar to the CTB-based approach, the found l_s and QP values follow the default values with increasing λ_S, but l_s increases disproportionally high at high λ_S values. However, in contrast to the CTB approach, values are in general lower than the defaults, which indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is greater than 1. Furthermore, they vary strongly at high λ_S values as the depth bit rate and SVD surfaces are not smooth.

In summary, this section provides a third set of combinations, which will be compared to results of the CTB-based search in the next section with respect to their RD performance.

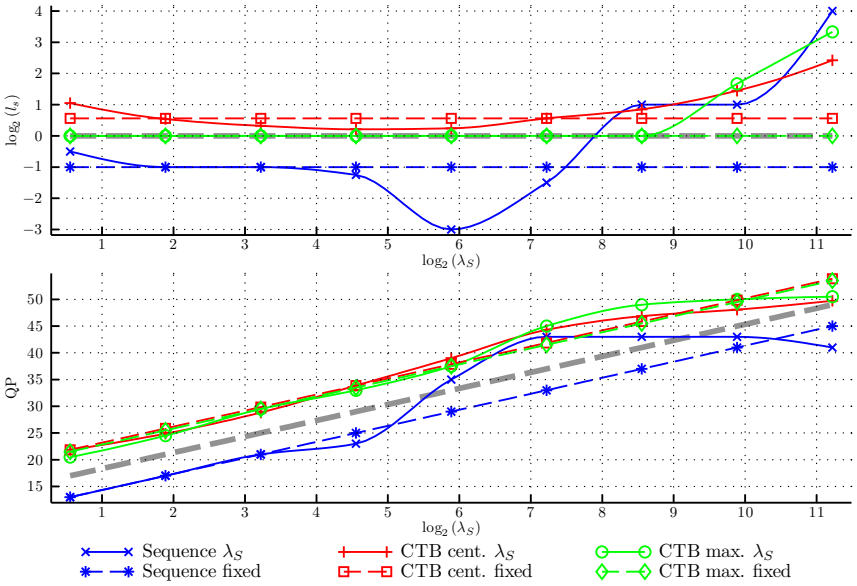


Figure 8.14: Optimal QP-l_s–$\log_2(\lambda_S)$ combinations; Medians

8.5.2.3 Evaluation of the Selected Combinations

The CTB-based and the sequence-based full search provided three sets of potentially optimal λ_S-l_s-QP combinations. However, the CTB-based search did not reveal the RD performance for using the found combinations constantly for the entire sequences; the results of the sequence-based full search might be unreliable as they are based on interpolated RD points. A question left open so far is thus how an encoder performs when using these combinations. To answer this, this section compares the results of nine test cases: Three test cases use the sequence and λ_S dependent values of the three sets of combinations as provided in Figures 8.9, 8.10, and 8.13. Another three test cases use the median values determined over sequences as also provided in Figures 8.9, 8.10, and 8.13, so that l_s and the QP only depend on λ_S. The last three test cases use the median l_s over all λ_S values and sequences, and a QP corresponding to the default QP plus an offset—the median difference between a) the QPs of all sequences and λ_S values and b) the default QP. The l_s value and the QP offset are thus constant for all sequences and λ_S values. Combinations for the latter six test cases are shown in Figure 8.14.

For assessment, Table 8.12 shows depth bit rate savings for all test cases in comparison to a test case using the default values. Savings are calculated based on the whole depth bit range provided by the evaluated λ_S range. This means that the limitations discussed in Appendix E.2 are not applied so that also the impact of changes at high depth bit rates can be analyzed. Results show that only combinations of the sequence-based full search outperform the default

QP&λ_M	$R_\Delta[\%]$ for λ_S & Sequences			$R_\Delta[\%]$ for λ_S			$R_\Delta[\%]$ fixed		
	Seq.	CTB max.	CTB center	Seq.	CTB max.	CTB center	Seq.	CTB max.	CTB center
Balloons	−2.4	−0.8	+1.3	0.0	+0.9	+1.1	+1.3	+1.6	+1.2
GTFly	−2.1	+0.4	+1.2	0.0	+1.2	+1.0	−0.9	+1.3	+1.1
Kendo	−0.5	+2.4	+2.0	+1.3	+0.3	+0.1	+0.9	+1.1	+0.8
Newsp.	−3.4	+3.4	+4.1	0.0	+4.9	+3.2	−1.4	+4.9	+3.7
P.Hall2	−1.7	0.0	+0.5	−0.1	+1.7	+1.7	+0.3	+2.0	+2.2
P.Street	−2.7	+0.9	+0.5	+0.2	+1.4	+1.1	−1.4	+1.4	+0.2
UndoD.	−1.8	−0.2	−0.1	−0.8	+0.1	+0.4	−0.4	−0.2	+0.3
Shark	−3.9	+1.1	+1.3	−2.7	+0.2	+0.7	−0.4	+1.7	+0.4
Mean	−2.3	+0.9	+1.3	−0.3	+1.3	+1.2	−0.2	+1.7	+1.2

Table 8.12: Depth bit rate delta R_Δ; Tested: Different λ_S, λ_M, and QP combinations; Reference: Default values; Full depth bit rate range.

combinations. The sequence and λ_S dependent combinations lead to the maximal depth bit rate reduction of 2.3%. The λ_S dependent combinations and the fixed combinations decrease the depth bit rate by about 0.3% and 0.2%, respectively. In contrast, test cases for the CTB-based search lead to depth bit rate increases of in average between 0.9% to 1.7%.

In summary, significant depth bit rate reductions are only achieved by sequence and λ_S depended optimization using the full search.

8.5.3 Discussion

Considering results of the sequenced-based full search, the encoder achieves depth bit rate reductions in all test cases by decreasing the QP and λ_M in comparison to the default values. This means by increasing the depth quality and the depth bit rate of modes that are pre-selected below the final selection stage C. This indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is greater than one. However, a value greater than one is not obvious considering that the depth distortion in major parts of the depth map is in general not relevant for the SVD as the collocated IV texture is homogenous. What must be considered furthermore is that for finding the optimal QP and λ_M combinations, only those blocks of the depth map are relevant that are actually encoded based on the QP and on λ_M. The majority of such blocks signal transform coefficients or motion vectors using AMVP. They are thus blocks for which the final mode selection stage C allocates an increased number of bits, which in turn means that they are associated with a higher SVD. Considering this, it is likely that their SVD depends stronger on the depth distortion than it is the case in other picture parts; therefore, a value of $\mathrm{d}D_S/\mathrm{d}D_M$ greater than one is reasonable.

The CTB-based search provides combinations that do not reduce the depth bit rate compared to the default values. The QP and λ_M values are in general higher as the default values, which indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is less than 1. Considering that $\mathrm{d}D_S/\mathrm{d}D_M$ differs locally, this result

does not necessarily contradict with the finding of the sequence-based full search: Given blocks with different values of $\mathrm{d}D_S/\mathrm{d}D_M$, the CTB-based search will converge to the optimal combination for the $\mathrm{d}D_S/\mathrm{d}D_M$ value that occurs most frequently. This combination is not necessarily optimal in an overall sense. It could e.g. be more effective to choose a combination optimal for blocks that have a $\mathrm{d}D_S/\mathrm{d}D_M$ value that occurs less frequently, but contributes more to the total RD cost. In summary, the evaluation results and the discussion indicate that the CTB search is not suitable to determine the optimal λ_M and QP combination.

8.5.4 Conclusion

This section provided a discussion and an experimental analysis of the optimal relationship between three encoder parameters: The Lagrange multipliers used in the SVDC-based and the depth distortion-based selection stages, i.e. λ_S and λ_M, and the depth QP. The discussion of the theoretical relationship showed that λ_M is given by the quotient of λ_S and the derivative of the SVD with respect to the depth distortion, i.e. $\mathrm{d}D_S/\mathrm{d}D_M$. Knowing λ_M the respective QP can be determined based on the conventional relationship used in texture coding. The experimental analysis targeted to find optimal λ_S-λ_M-QP combinations for a scenario without local or temporal adaptation. For this, it applied a CTB-based search similar to that used in [Sull 98] and a sequence-based full search.

The CTB-based search indicates that optimal QP and λ_M values are higher than the default values, which indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is less than 1. However, combinations provided by the CTB-based search do not lead to depth bit rate reductions. A possible explanation is that the values of $\mathrm{d}D_S/\mathrm{d}D_M$ vary locally: The CTB-based search rather converges to the combination that is optimal for the most frequently occurring value of $\mathrm{d}D_S/\mathrm{d}D_M$ than to the combination that provides the best overall RD performance. The sequence-based full search provides QP and λ_M values that are lower than the default values, which indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is greater than 1. This value seems reasonable considering that the selection of the QP and λ_M primarily affects blocks for which the SVD depends strongly on the depth distortion.

Significant improvements of the RD performance are only achieved by the sequence-based full search when using sequence dependent optimization. Depth bit rate savings considering unique values for all sequences are small ($\leq 0.3\%$). This means in conclusion that the default QP values and Lagrange multipliers provide already an RD performance that is close to the optimum for choosing unique values for all sequences. For this, following evaluations in this thesis will be further based on the *optimized encoder setup* using default QP and λ_M values.

8.6 Chapter Summary

This chapter evaluated several options to optimize SVDC-based encoding with respect to complexity and RD performance.

The analysis of the RM's features showed that they are effective and increase the RD performance with an acceptable complexity increase. An exception is view combination, which causes a hiding effect when encoding the (in coding order) first IV's depth while assuming an uncoded texture for the second IV. An evaluation of different view combination variants showed that the hiding effect can be mitigated by disabling background suppression in the view combination step when encoding the first IV. With these modifications, the depth bit rate decreases by 9.6% on average compared to the *initial encoder setup*.

The RM can use different SV generation setups to generate the tested and the reference SV texture by view interpolation and extrapolation from coded and uncoded texture and depth. Highest depth bit rate reductions (10.4% compared to the *initial encoder setup*) can be achieved by 1) disabling background suppression when encoding the first IV depth, 2) interpolating the tested SV texture from coded IV data when available and uncoded otherwise, and 3) interpolating the reference SV texture in general from uncoded IV data, but using the first IV's coded texture when encoding the first IV depth. When only using uncoded textures to render the tested and reference SV texture, coding losses are moderate ($< 2\%$). Losses when using uncoded depth, which would occur for a scenario with independent encoding of the second IV depth, are significant ($> 10\%$). View extrapolation setups lead to depth bit rate increases of about 15% to 20% when the hiding effect is already reduced by disabled background suppression. In summary, the evaluation showed how to improve the RD performance further and also quantified the losses occurring in scenarios with independent encoding of IVs or texture and depth.

In addition to the SVDC, the encoder can employ a depth distortion term [Jung 12c]. In a scenario with matching sender- and receiver-side renderers, the depth distortion term reduces the depth bit rate only significantly (8.1% on average) when SVDC computation is affected by the hiding effect. In setups addressing this issue already by a modified view combination process or by modified SV generation, it leads to only small additional depth bit rate reductions on average (less than 0.5%). However, the depth encoding times decrease about 4% to 8% when using the depth weight as depth PSNR increases by about 6 to 9 dB, which causes a decrease of the rendering overhead due to the magnitude of the depth change. In conclusion, the depth distortion term is thus also beneficial in cases in which the same renderer is used at the encoder and the receiver-side.

The encoder can employ the SVDC also in lower-level mode selection stages. Enabling the SVDC successively in intra-picture prediction for RQT coding, switching RDOQ, mode pre-selection, and wedgelet parameter selection leads to depth bit rate saving increments[6] of 0%, 0.1%, 0.8%, and 0.4%, respectively, and depth encoding time increments of 3%, 6%, 5%, and 20%, respectively. Thus, using the SVDC for mode pre-selection is beneficial, while using it in other stages leads to a rather worse complexity-RD performance trade-off. In inter coding, enabling the SVDC successively for RQT coding, switching RDOQ, AMVP/merge decisions for all CB partitionings, and selection motion parameters leads to depth bit rate saving increments of 1.1%, 0.2%, 0.5%, and 3.8%, respectively, and encoding time increments of -1%, 10%, 4%, and 306%, respectively. Considering again trade-offs between complexity and RD performance, using the SVDC for the inter RQT is most efficient. In summary, encoding using

[6] All percentages related to the depth bit rate or encoding time of the *optimized encoder setup*.

the SVDC in all mode selection stages leads to depth bit rate savings of 6.6% while the depth encoding time increases by a factor of approximately 3.5. Nevertheless, findings show how other trade-offs between depth bit rate and encoding complexity can be achieved, e.g. to adapt to a given application scenario.

Encoder parameters related to RD optimization are the Lagrange multipliers for the depth distortion-based and the SVDC-based selection stages and the depth QP. Considering their optimal relationship the Lagrange multiplier for depth-based selection stages λ_M is given by the quotient of the Lagrange multiplier for SVDC-based decision stages λ_S and the derivative of the SVDC with respect to the depth distortion. Knowing λ_M, the respective QP can be determined based on the conventional relationship used in texture coding. An experimental sequence-based full search provided QP and λ_M values that are lower than the default values, but only achieved significant improvements of the RD performance when using sequence depended optimization. Bit rate savings considering unique values for all sequences are small ($\leq 0.3\%$). The default QP values and Lagrange multipliers provide thus already an RD performance that is close to the optimal one for choosing unique values for all sequences.

In summary, this chapter answered the question on how to employ the RM optimally in encoding. It showed which RM features should be enabled, how to generate the tested and reference SVs, how to select the depth distortion term, and it clarified in which selection stages SVDC-based encoding should be applied and how to choose the Lagrange multipliers and QPs properly. The modified VSO setup—in the following referred to as *optimized encoder setup*—further decreases the depth bit rate in comparison to the *initial encoder setup* by 10.7%.

9 Evaluation and Optimization Using Other Test Conditions

Evaluations so far considered only one SV texture, used only the LCR as receiver-side renderer, compared SVDC-based encoding only with depth distortion-based encoding, presented depth bit rate savings only with respect to the PSNR, and considered depth bit rates only. To allow a broader assessment of the achieved coding gains that goes beyond these test conditions, this chapter presents further evaluations and optimizations: Section 9.1 evaluates how SVDC-based encoding performs when considering more than one SV position. Section 9.2 analyzes coding gains for scenarios in that the receiver-side rendering algorithm deviates from the rendering algorithms of the RM. Section 9.3 optimizes existing SVD estimation methods and compares them to SVDC-based encoding. Section 9.4 analyzes the coding gains achieved by SVDC computation with two alternative quality metrics—the Structural Similarity (SSIM) index and Pratt's Figure of Merit (FOM). Section 9.5 provides sample pictures, which indicate what PSNR gains actually mean for the SV quality. Finally, Section 9.6 analyzes the RD performance of SVDC-based encoding using JCT-3V's test conditions.

9.1 Synthesized View Position Setups

For simplification, evaluations so far used a rendering scenario with only one SV texture. However, SVDC computation targets systems with auto-stereoscopic displays, which require multiple SV textures depicting the 3D scenes from different viewing positions. A straightforward extension for such a scenario is to render also multiple SV textures in SVDC computation. However, because of the required computational resources, rendering textures for all SV positions (e.g. for a 28 view display) is not feasible in most application scenarios. This raises two questions: 1) How does the complexity increases when SVDC computation regards multiple SV positions? 2) How is the SV quality affected at SV positions that are not regarded in SVDC computation? Both questions are addressed in an experimental evaluation considering view extrapolation (Section 9.1.1) and view interpolation (Section 9.1.2). *Related works*: The text, figures, and the evaluations in this section are based on the own related contribution [Tech 12d]. In contrast to this paper, the evaluation uses the latest HTM software and different test conditions.

9.1.1 View Extrapolation

This section evaluates SVDC-based depth encoding in a scenario that extrapolates SVs from texture and depth of a single IV. This scenario is of practical importance e.g. when an auto-stereoscopic display requires SVs depicting the 3D scene from position outside the viewing range of the decoded IVs. It is also of theoretical interest as it is the basis for view interpolation. For this, the following analyzes the quality of multiple extrapolated SV textures that is achieved when SVDC computation regards only a subset of them. In the evaluation, view positions are numbered in the range of 0 to 1; view position 1 is the position of the right IV and view position

Test case	E_0	$E_{1,c}$	$E_{1,l}$	E_2	E_4
SV positions	–	0.5	0	$0, 0.5$	$0, 0.25, 0.5, 0.75$

Table 9.1: SV positions for view extrapolation test cases.

0 is at approximately two stereo baseline distances on the left of position 1 (i.e. the position of the left IV).

The evaluation uses the default test conditions (as defined in Table 7.2) with the following modifications:

- Input views: Only the right IV is encoded, the evaluation regards only its depth bit rate.
- VSO setup: The RM uses the full extrapolation setup, the ratio ρ for the depth distortion term is $2^{5.5}$ (as in the *optimized encoder setup*), the considered SV positions depend on the test case as defined in Table 9.1.
- Evaluation setup: 23 SVs equally spaced between the left and the right IV's positions and generated by view extrapolation are regarded.

To discuss typical effects of SVDC-based encoding in the view extrapolation scenario, Figure 9.1 shows the quality of SV textures extrapolated from the coded IV texture and depth of the *GtFly* sequence[1]. Figure 9.1 (a) shows as reference the RD performance for test case E_0, which uses only the depth distortion, but not the SVDC. As only the depth QP is varied, the right IV at view position 1 has the same quality at all depth bit rates. Furthermore, the SVD decreases the more with decreasing depth bit rates, the farther the distance of the SV's position to right IV's position. The obvious reason is that the disparity distortion increases with the view distance. For the *GTFly* sequence, the SV texture's PSNR decreases almost linearly[2] with increasing distance from the right IV. The effect of SVDC-based encoding is shown in Figure 9.1 (b) for the E_4 test case: The SVD increases less not only for decreasing depth bit rates but also for increasing distances of the SV from the IV.

9.1.1.1 Synthesized View Quality Gains

To analyze coding gains, Figure 9.1 (c) shows cross sections of the SV texture PSNR surfaces that are obtained in the different test cases at a depth bit rate of 41 kbit/s, i.e. the SV texture quality at different SV positions. Since the SV texture PSNR decreases less with decreasing SV positions, it is at SV positions close to 0 in test cases $E_{1,c}$, $E_{1,l}$, E_2, and E_4 about 2 dB higher than in the E_0 test case. Furthermore, the SV texture PSNR curves deviate locally from a linear development: test cases that consider SV position 0.5 in SVDC computation (i.e. $E_{1,c}$, E_2, and E_4) have a local maximum there; the SV texture PSNR close to position 0 is in test cases

[1] Results for other sequences are shown by Figures F.41 to F.47.

[2] [Kim 09] shows that the SV texture's MSE is proportional to the mean absolute disparity difference, which is itself proportional to the SV distance. From this, it can be easily shown that a) the SV texture PSNR should actually decrease by $\Delta\text{PSNR} = 10 \cdot \log_{10}(1 + n \cdot x)$ with n denoting the factor between the SV distance and SV texture's MSE and x denoting the SV distance; and b) ΔPSNR is approximately proportional to x for small values of n and x, which seem to be given in the shown test cases.

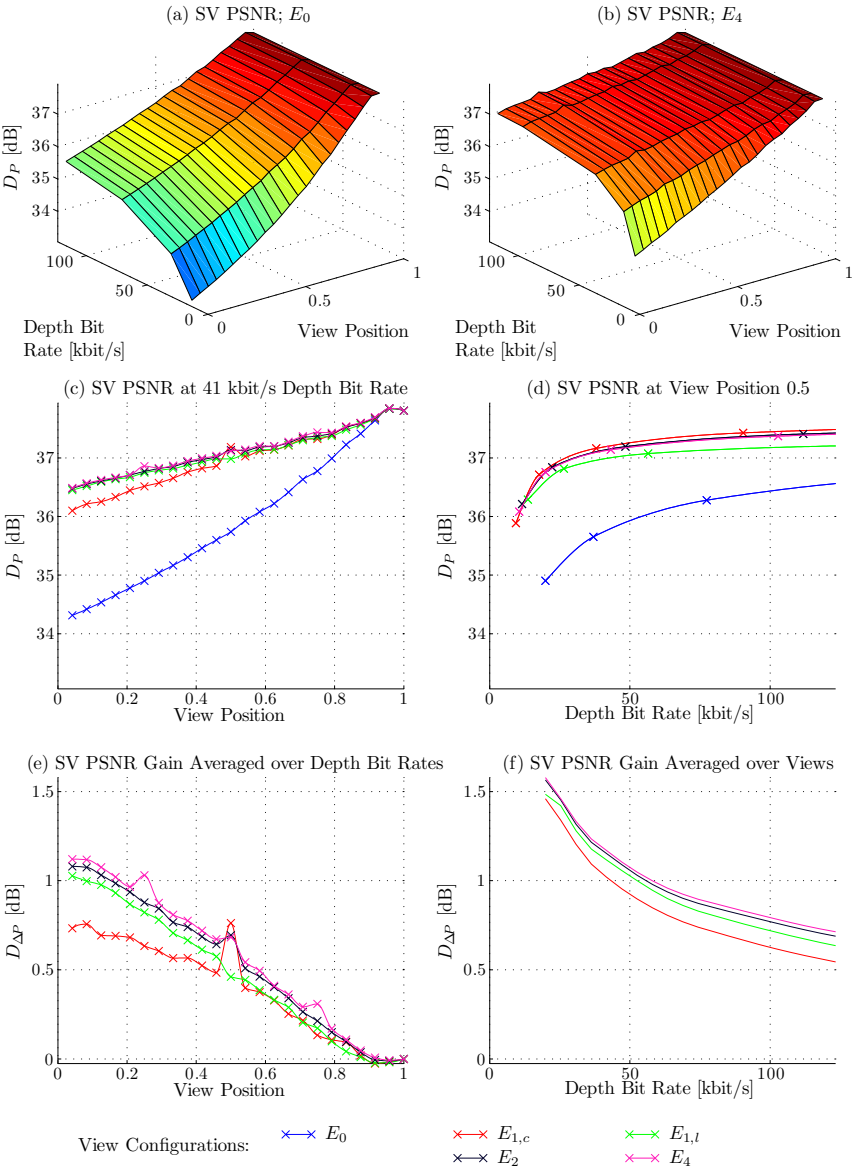


Figure 9.1: *GTFly*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

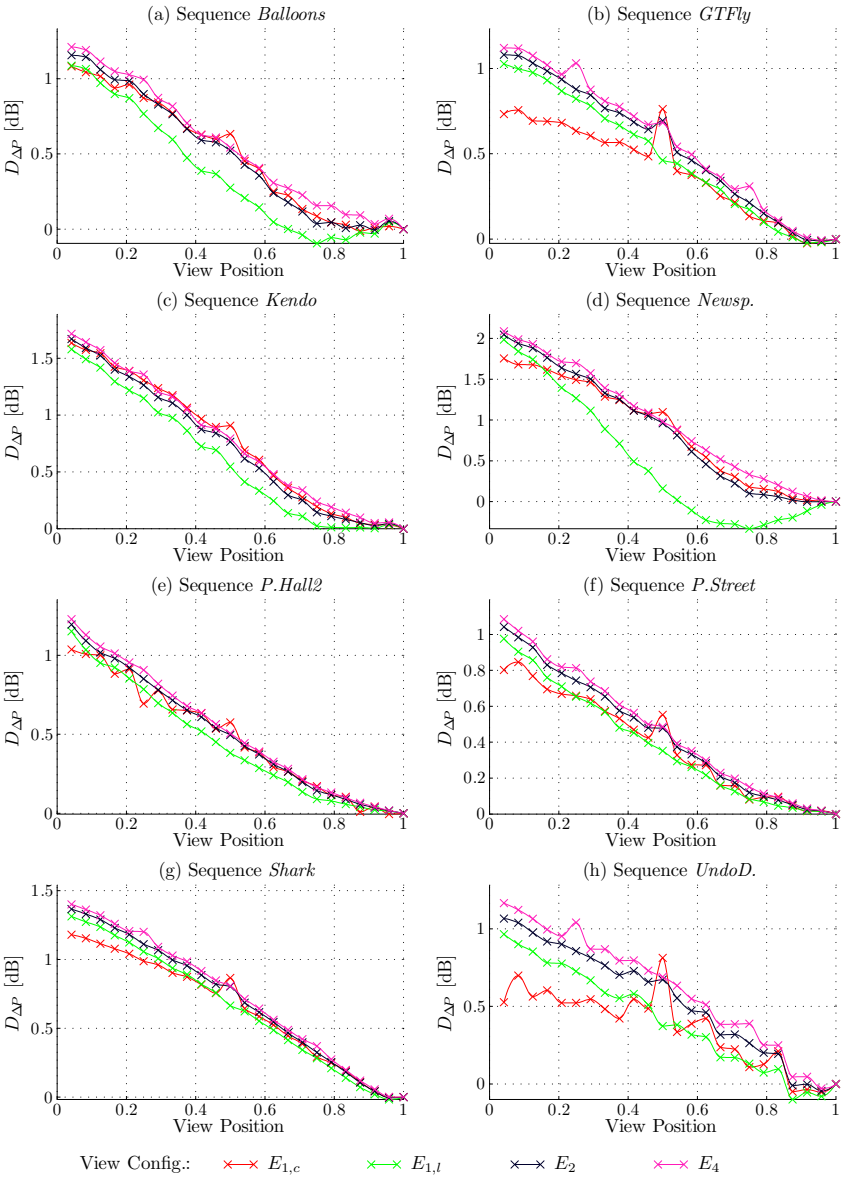


Figure 9.2: SV texture PSNR gains averaged over the depth bit rate at different SV positions in view extrapolation.

considering it (i.e. $E_{1,l}$, E_2, and E_4) greater than in test case $E_{1,c}$ (which does not consider it). These local peaks at optimized SV positions confirm that SVDC computation works effectively as intended. In addition, cross sections of the SV texture PSNR surfaces at SV position 0.5 in Figure 9.1 (d) show that this is true for all depth bit rates.

Whereas Figure 9.1 (c) analyzed the SV texture PSNR only at a particular depth bit rate, Figure 9.1 (e) shows SV texture PSNR gains in comparison to the E_0 setup and averaged over the evaluated depth bit rate ranges. Also here, the SV quality increases in general and in particular at SV positions regarded in SVDC computation. The same effect similarly occurs for other sequences as Figure 9.2 shows, although peaks are not as distinct as for the *GTFly* sequence. Beyond that two other observations can be made.

The first observation is that the SV texture gains increase with the number of SV positions considered in SVDC computation. However, the more SV positions are already considered, the lower is the increment of the SV texture PSNR gain. This can be observed in Figure 9.2 by comparing test case E_2 with test case $E_{1,l}$ (or $E_{1,c}$) and by comparing test case E_4 with test case E_2: Whereas using two SV positions instead of one SV position increases the SV texture PSNR gains additionally for most sequences and SV positions, using four instead of two SV positions leads to only minor differences. This indicates that the SV texture PSNR gain converges to a maximum that is already nearly reached when using four SV positions.

The second observation can be made when considering test case $E_{1,l}$. Although test case $E_{1,l}$ provides for most SV positions SV texture PSNR gains, losses occur for some sequences (e.g. *Newspaper* at SV positions 0.8). This indicates that the SVDC calculated at SV position 0 represents the SVDC for SV positions close to 1 not very accurately. The reason for this is that, when only considering SV position 0, the depth values of SV texture samples that are occluded at this SV position do not affect the SVDC measured there. Consequently, the encoder has a large degree of freedom to encode them without producing a local SVDC. Nevertheless, at SV positions closer to 1, these SV texture samples are not occluded and thus produce an SVD.

For further analysis, Figure 9.1 (f) shows the SV texture PSNR gain of the test cases compared to the E_0 test case and averaged over all SV positions: The order of performance is the same as in Figure 9.1 (e) and consistent for all depth bit rates. Furthermore, it can be observed that the SV texture PSNR gains increase with decreasing depth bit rates. Reason for this is that SV texture PSNR in the E_0 test case decreases faster with decreasing depth bit rates as in other test cases. In conclusion, Figure 9.1 (f) confirms the observations for all evaluated depth bit rates.

9.1.1.2 Depth Bit Rate Savings and Computational Complexity

The analysis of the SV quality gains left open two questions. The first is how SVDC-based encoding regarding multiple SVs reduces the depth bit rate at a constant quality of textures in the SV range; and the second is how it increases the computational complexity.

To answer the first question, Table 9.2 shows depth bit rate savings calculated with respect to the average PSNR of the SV textures at the evaluated SV positions and using the E_0 test case as reference. Depth bit rate savings are relative high, which is in particular caused by the SV

	$R_\Delta[\%]$				$D_{\Delta P}[\%]$ [dB]				$T_\Delta[\%]$			
	$E_{1,c}$	$E_{1,l}$	E_2	E_4	$E_{1,c}$	$E_{1,l}$	E_2	E_4	$E_{1,c}$	$E_{1,l}$	E_2	E_4
Balloons	−74.5	−68.4	−75.1	−76.7	+0.85	+0.77	+0.90	+0.94	+54	+48	+ 86	+161
Kendo	−83.0	−80.3	−83.3	−84.1	+1.29	+1.24	+1.34	+1.37	+39	+40	+ 72	+136
Newsp.	−80.7	−58.0	−81.5	−84.3	+1.22	+0.86	+1.29	+1.38	+64	+62	+102	+185
P.Hall2	−75.2	−73.8	−77.4	−77.9	+0.72	+0.73	+0.78	+0.80	+64	+59	+106	+194
P.Street	−74.6	−75.6	−78.2	−78.7	+0.58	+0.62	+0.66	+0.67	+73	+66	+112	+203
GTFly	−82.8	−86.4	−87.7	−88.0	+0.58	+0.68	+0.73	+0.75	+55	+55	+ 87	+156
Shark	−87.3	−87.7	−88.9	−89.3	+0.93	+0.97	+1.01	+1.03	+47	+48	+ 79	+147
UndoD.	−58.1	−63.7	−70.3	−72.3	+0.56	+0.70	+0.82	+0.86	+50	+46	+ 80	+149
Mean	−77.0	−74.2	−80.3	−81.4	+0.84	+0.82	+0.94	+0.98	+56	+53	+ 90	+166

Table 9.2: Depth bit rate R_Δ, SV PSNR $D_{\Delta P}$, and depth encoding time T_Δ deltas; Tested: Different SV position setups; Reference: E_0 setup; All: View extrapolation.

texture PSNR gain at high depth bit rates and the slow increase of the SV texture PSNR in the reference test case E_0 with increasing depth bit rates. In general, bit rate savings correspond to the SV texture PSNR gains that are also shown in Table 9.2 and averaged over view positions and depth bit rates. This means that the $E_{1,c}$ and $E_{1,l}$ test cases achieve the smallest depth bit rate savings (77% and 74.2%, respectively); using an additional SV position in the E_2 test case leads to some more savings (80.3%); and additional savings by using two more SV's in the E_4 test case are rather small (81.4%). Encoding times in Table 9.2 answer the second question: Results for the $E_{1,c}$ and $E_{1,l}$ test cases show that calculating the SVDC in one SV approximately increases the depth encoding time by about 54%. Other test cases indicate that each additional SV further increases the depth encoding time by about 36%.

In summary, the evaluation shows that depth bit rate reductions are with about 80% relatively high and that using more than two SVs is not reasonable as additional savings are small. Regarding the high depth bit rate savings, it should be noted that the LCR applies a very simple hole filling method. Because the size of the holes is proportional to the SV distance, this method is rather suitable for SV distances less than the two stereo baselines used in the evaluation. In such scenarios, SV gains and also the achieved depth bit rate savings are lower since the SV texture PSNR gains increase with the SV distance (see Figure 9.2). However, results in this section are conclusive as they still give an indication of what can be achieved in extrapolation scenarios that use different kinds of hole filling methods, as e.g. complex inpainting or hole filling from another IV without view combination [Boss 12].

9.1.2 View Interpolation

This section provides an evaluation similar to that of the last section, but for view interpolation as employed by the 3D video system discussed in this thesis. The RM and the receiver-side LCR interpolate the SVs from both IVs. Furthermore, SVDC computation regards the SV positions defined in Table 9.3, i.e. one to seven SVs in test cases I_1 to I_7, respectively.

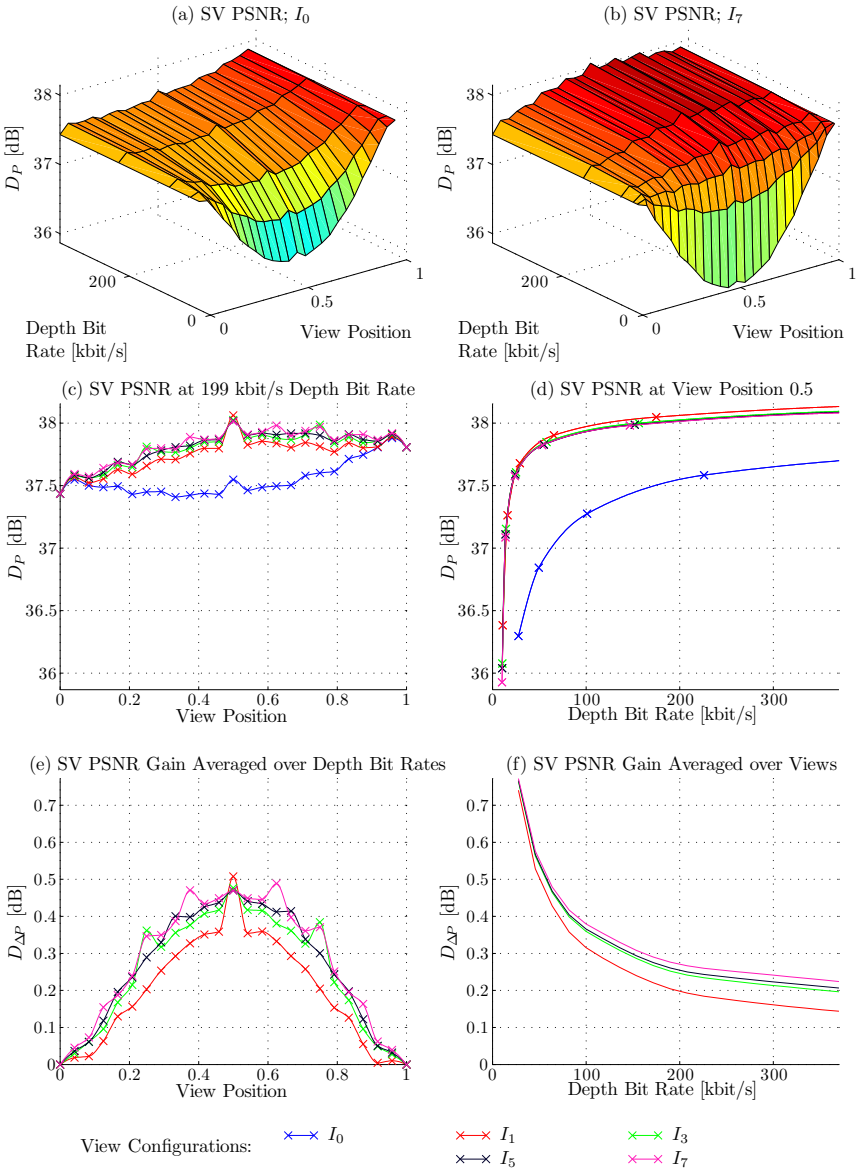


Figure 9.3: *GTFly*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

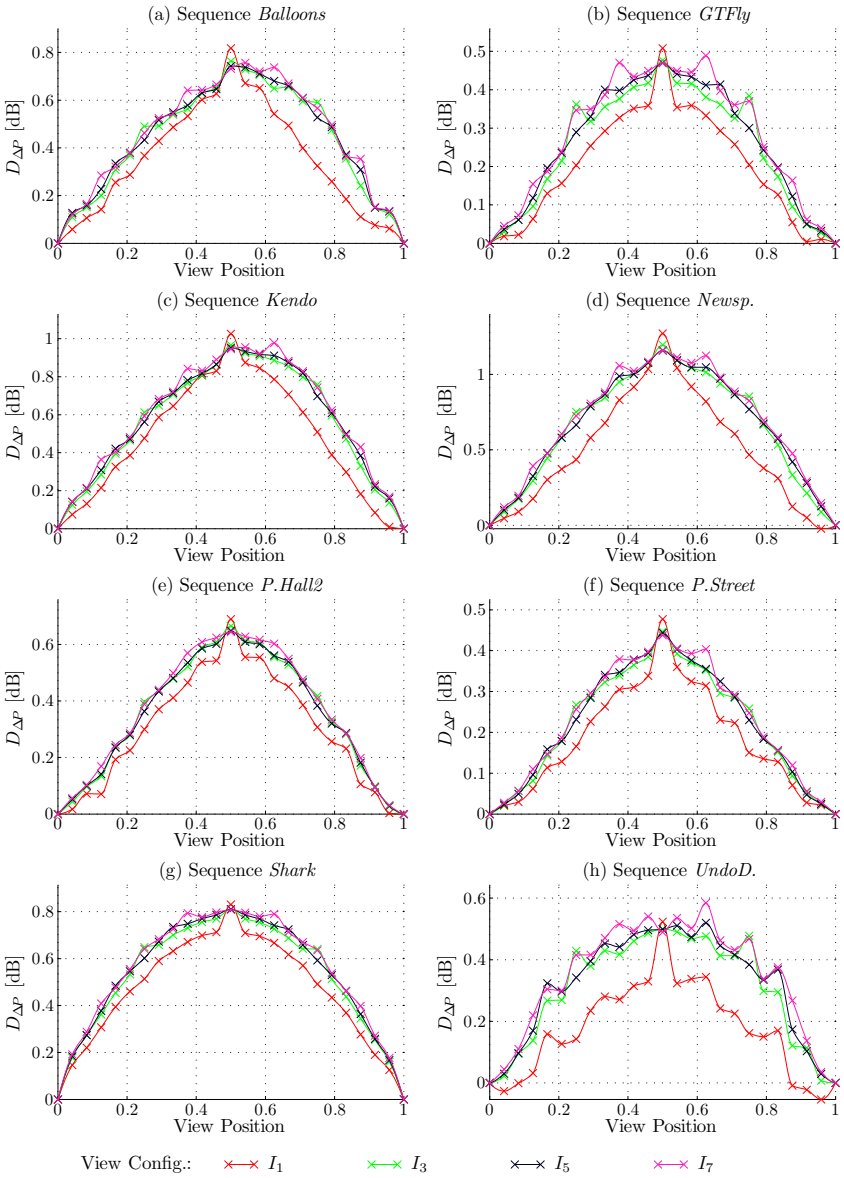


Figure 9.4: SV texture PSNR gains averaged over the depth bit rate at different SV positions in view interpolation.

Test case	SV position
I_0	–
I_1	0.5
I_3	$0.25, 0.5, 0.75$
I_5	$0.167, 0.333, 0.5, 0.667, 0.833$
I_7	$0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875$

Table 9.3: SV positions for view interpolation test cases.

As reference for the following evaluations, Figure 9.3 (a) shows for depth distortion-based encoding (i.e. test case I_0) how the SV texture PSNR of the *GTFly* sequence depends on the depth bit rate and the SV position. Some characteristics of the SV texture PSNR surface are caused by the view combination step, which derives the SV texture as weighted average of the SV textures extrapolated from the left and the right IV with weights depending on the distance of the SV position to the IV positions (see Section 3.2.4).

More specifically, the SV texture PSNRs at view positions 0 and 1 do not depend on the depth bit rate as the weights of the right and the left SV textures, respectively, are zero. Furthermore, the SV texture PSNR varies the more with the depth bit rate, the closer the SV position is to 0.5. Reason for this is that at view positions not equal to 0.5, the SV texture that is extrapolated from the farther IV (and thus depends more on the depth bit rate) has a smaller weight as the SV texture that is extrapolated from the closer IV (and thus depends less on the depth bit rate). At view position 0.5 the left and the right SV textures have equal weights; consequently, the depth bit rate has the strongest impact on the SV texture PSNR.

In Figure 9.3 (a) the SV texture PSNR decreases towards SV position 0.5. This indicates that, similar to view extrapolation, depth distortions have an increasing effect on the SV quality. However, an SV texture PSNR decrease towards SV position 0.5 does not occur for all test sequences. In particular, when sequences *PoznanHall2* and *PoznanStreet* are encoded at high depth bit rates, the SV texture PSNR increases towards SV position 0.5 (see Figures F.51 and F.52, respectively). Possible explanations for this are also related to the weighted averaging of the extrapolated SV textures. More specifically, when the left and the right SV textures have uncorrelated signal parts, the averaging damps these parts in the combined SV texture. This damping may have two effects, which are related to the reference and the tested SV texture: First, when the uncorrelated signal parts are caused by some kind of noise in the uncoded IV textures, weighted averaging reduces this noise in the reference SV texture. Consequently, when this noise is lost by encoding of the IV textures, the PSNR of tested SV texture with respect to the reference SV texture decreases less than the PSNR of the coded IV texture with respect to the uncoded IV texture. Second, when the uncorrelated signal parts are distortions introduced by encoding of the IV textures, weighted averaging reduces this noise in the tested SV texture. Consequently, the PSNR of the tested SV texture with respect to the reference SV texture is also less than the PSNR of the coded IV texture with respect to the uncoded IV texture.

Figure 9.3 (b) shows for the example of the I_7 test case that the SV texture PSNR increases by

SVDC-based encoding. SV texture PSNR losses due to disparity distortions decrease and the effect discussed above becomes visible: The SV texture PSNR at high depth bit rates is greater than the IV texture PSNRs. Furthermore, additional gains might be achieved by a positive hiding effect. This means that the encoder might encode the left IV depth (i.e. the second in coding order) such that the view combination step locally favors the extrapolated SV texture with less distortion. In summary, the I_7 test case achieves SV gains over the whole space spanned by SV positions and depth bit rates. The following sections analyze these gains further.

9.1.2.1 Synthesized View Quality Gains

Figure 9.3 (c) shows a cross section of the SV texture PSNR surface at a constant depth bit rate. The SV texture PSNR has not only local peaks at SV positions that are considered in SVDC computation, but also increases in general with the number of considered SV positions. Figure 9.3 (e) confirms this behavior also for the SV texture PSNR averaged over all depth bit rates and Figure 9.4 for the other test sequences. In the view extrapolation scenario, additional SV texture PSNR gains decrease with an increasing number of SV positions regarded in SVDC computation. Similarly, using more than three SV positions in the view interpolation scenario increases the SV texture PSNR gain for most sequences only insignificantly. On the other hand, considering only SV position 0.5 (i.e. test case I_1) achieves only very minor gains at SV positions close to 0 and 1, or even losses for the *UndoDancer* sequence.

Figures 9.3 (d) and (f) show cross sections of the SV texture PSNR surfaces at SV position 0.5 and the SV texture PSNR gains averaged over all SV positions, respectively. As in view extrapolation, SVDC-based encoding regarding multiple SVs achieves SV texture PSNR gains at all evaluated depth bit rates and the highest SV texture gains at low depth bit rates. Furthermore, the rank order of test cases with respect to their performance is the same at all depth bit rates.

9.1.2.2 Depth Bit Rate Savings and Computational Complexity

Table 9.4 shows depth bit rate savings with respect to the averaged PSNRs of all SV textures. Considering only one SV in test case I_1 already achieves high depth bit rate savings of 70%. Using three SV positions in test case I_3 increases the depth bit rate savings to 74.5%. Five and seven SV positions in test cases I_5 and I_7 lead to only small additional increases to 74.8% and 75.3%, respectively. Thus, the increase of depth bit rate savings is 0.3% when using five instead of three SVs and 0.5% when using seven instead of five SVs. Reason for this could be that, in contrast to the I_5 test case, the I_7 test case uses SV positions that coincide with the 23 SV positions that are sampled to derive the average SV texture PSNR.

To evaluate the computational complexity, Table 9.4 shows the depth encoding time deltas for the evaluated test cases. Using SVDC computation for one SV position (I_1) increases encoding depth time by about 82%. For each additional SV position, the depth encoding time then increases by 54%. These results show that for a trade-off between computational complexity and RD performance using more than three SV is not reasonable.

	$R_\Delta[\%]$				$D_{\Delta P}[\%]$ [dB]				$T_\Delta[\%]$			
	I_1	I_3	I_5	I_7	I_1	I_3	I_5	I_7	I_1	I_3	I_5	I_7
Balloons	−69.1	−73.7	−74.1	−74.3	+0.41	+0.49	+0.50	+0.51	+ 78	+180	+281	+386
Kendo	−77.0	−80.6	−80.7	−81.1	+0.59	+0.67	+0.68	+0.69	+ 55	+143	+229	+319
Newsp.	−68.6	−75.9	−76.3	−76.9	+0.54	+0.69	+0.70	+0.72	+ 86	+193	+301	+410
P.Hall2	−61.8	−64.4	−64.3	−64.9	+0.36	+0.41	+0.40	+0.41	+ 98	+222	+349	+476
P.Street	−65.9	−70.0	−70.3	−70.9	+0.20	+0.23	+0.23	+0.24	+101	+228	+352	+480
GTFly	−79.6	−82.4	−82.6	−83.0	+0.25	+0.30	+0.30	+0.32	+ 88	+195	+299	+405
Shark	−83.9	−85.9	−86.1	−86.2	+0.55	+0.60	+0.61	+0.62	+ 71	+171	+270	+371
UndoD.	−54.2	−63.4	−64.1	−65.0	+0.25	+0.38	+0.39	+0.41	+ 80	+181	+282	+386
Mean	−70.0	−74.5	−74.8	−75.3	+0.39	+0.47	+0.48	+0.49	+ 82	+189	+296	+404

Table 9.4: Depth bit rate R_Δ, SV PSNR $D_{\Delta P}$, and depth encoding time T_Δ deltas; Tested: Different SV position setups; Reference: I_0 setup; All: View interpolation.

9.1.3 Conclusion

This section evaluated the effects of considering multiple SV positions in SVDC computation. When considering one SV position, the viewing range's average SV texture quality increases in view extrapolation and view interpolation scenarios by about 0.8 dB and 0.4 dB, respectively. These SV PSNR gains correspond to depth bit rate savings of about 77% and 70%, respectively, and are achieved with a depth encoding time increase of about 56% and 82%, respectively. The SV PSNR increases most at SV positions considered in SVDC computation, but in general also at the other SV positions in the viewing range. However, for some sequences using one SV position for SVDC computation can lead to minor losses at SV positions farthest to the considered one. Such losses can be avoided by increasing the number of SV positions to e.g. two in the view extrapolation scenario or three in the view interpolation scenario, which increases the depth bit rate savings to 80.3% and 74.5%, respectively, but also depth encoding time by 90% and 189%, respectively. Using more than two or three SV positions increases depth bit rate savings only marginally further, but increases computational complexity linearly. More specifically, depth encoding times increase by about 37% and 54%, respectively, per additionally considered SV position. In a trade-off between computational complexity and RD performance using more than two or three SV positions is thus not reasonable in most application scenarios.

9.2 Alternative View Synthesis Algorithms

In all evaluations so far, the receiver-side used the LCR, i.e. the same view synthesis algorithm as the RM. This section answers the question how the RD performance is affected when the receiver-side employs a different algorithm. More specifically, it analyzes the RD performance of different encoder setups and different depth distortion weights with respect to four different receiver-side view synthesis algorithms. *Related works*: Own evaluations using the VSRS and 1D-Fast software as receiver-side renderers have been presented in [Tech 12c]. Furthermore, [Wang 12b] provides a similar evaluation. Different renderers and depth weights have been

evaluated in standardization [Son 12, Jung 12b, Jung 12c]. In contrast to these publications, Section 9.2 also evaluates different encoder setups, considers the general and the 1D mode of the VSRS individually, and is based on the latest HTM version and different test conditions. The own evaluation in [Tech 18] is related. It analyzes how the sender-side measured SVDC or an estimated SVD correlate with the receiver-side SVD when the renderers are not matching.

9.2.1 Tested Parameter Space

The analysis uses the receiver-side evaluation setup defined in Table 7.2, but with different view synthesis algorithms: 1) The LCR; 2) the LCR with enabled boundary noise removal [LCR (BNR)], i.e. 1D-Fast; 3) the VSRS in 1D mode [VSRS (1D)], described in Appendix A.2.1; and 4) the VSRS in general mode [VSRS (Gen.)], described in Appendix A.2.2.

With respect to the LCR the *optimized encoder setup* achieved the best performance so far. Nevertheless, it might not be optimal for the other view synthesis algorithms as they differ from the LCR. The majority of view synthesis algorithms similarly perform the basic features of the LCR like warping, occlusion and disocclusion handling. More significant deviations can occur for the view combination step. In that case, considering the LCR's view combination step in SVDC computation might not be optimal. To investigate this, the evaluation not only compares the *optimized encoder setup* to depth distortion-based encoding, but considers two further RM setups: 1) The *optimized encoder setup*, but using only view extrapolation, i.e. view combination is not performed; and 2) the *optimized encoder setup*, but using *BS off* view combination, i.e. no background suppression in encoding of both depth maps (see Section 8.1.4).

The depth distortion term can furthermore mitigate SVDs due to the mismatch of the RM and the receiver-side renderer as proposed in [Jung 12c]. The evaluation in Section 8.3 showed that for the *optimized encoder setup* an SVDC-depth distortion ratio ρ equal to $2^{5.5}$ is optimal for the LCR. However, when considering other view synthesis methods and encoder setups, other ratios ρ might provide better results. For this, ρ is the third parameter varied in the evaluation.

9.2.2 RD Performance

For each considered view synthesis method, the RD performance is analyzed in two steps: The first step compares the *optimized encoder setup* with depth distortion-based encoding (Table 9.5). The second step evaluates different test cases that combine the encoder setups defined in Section 9.2.1 with different SVDC-depth distortion ratios ρ. Figures 9.5 and 9.6 show the results of the second step, in which R_{Δ} is the depth bit rate delta in comparison to the *optimized encoder setup* with the optimal ratio of ρ equal to $2^{5.5}$. Table 9.6 summarizes the depth bit rate savings for the optimal ratios found in Figures 9.5 and 9.6. In each test case, the receiver-side renders the reference and the tested SV textures with the same method.

Results for the LCR confirm the findings of the preceding chapters: Table 9.5 shows that the depth bit rate savings of the *optimized encoder setup* are 77.8%. Figures 9.5 (a), (c), and (e) show that the *optimized encoder setup* with the optimal ratio ρ equal to $2^{5.5}$ outperforms other

	$R_\Delta[\%]$			
	LCR	LCR (BNR)	VSRS (1D)	VSRS (Gen.)
Balloons	−78.2	−77.0	−63.8	−66.9
Kendo	−83.8	−82.3	−74.7	−71.9
Newsp.	−81.1	−80.2	−66.1	−71.6
P.Hall2	−65.8	−64.1	−58.0	−62.1
P.Street	−73.9	−73.3	−59.0	−66.3
GTFly	−85.0	−84.5	−54.2	−78.1
Shark	−86.2	−86.2	−67.9	−81.7
UndoD.	−68.6	−66.8	−36.1	−54.9
Mean	−77.8	−76.8	−60.0	−69.2

Table 9.5: Depth bit rate delta R_Δ for different view synthesis methods; Tested: The *optimized encoder setup*; Reference: DD-based encoding.

encoder setups even when they use their respective optimal ratios. More specifically, in the comparison to the *optimized encoder setup* in Table 9.6 the Extrapolation and the *BS off* setups lead to depth bit rate increases of 24.7% and 0.9%, respectively.

When the receiver-side uses the LCR with boundary noise removal, coding results differ only marginally from the results for the LCR without boundary noise removal. As Table 9.5 shows, the *optimized encoder setup* still achieves bit rate savings of about 76.8% in comparison to depth distortion-based encoding and thus 1% less than as for the LCR. Furthermore, Figures 9.5 (b), (d), and (f) and Table 9.6 show that, as well as for the LCR, the Extrapolation and the *BS off* setups do not achieve further rate savings, but losses of about 22.7% and 0.7%, respectively. Optimal ratios ρ are lower as for the LCR, as the correlation of the SVDC measured by the RM and the actual receiver-side SVDCs decreases.

Major difference between the VSRS in 1D mode (as described in detail in Figure A.2) and the LCR is that the former renders the SV texture by warping the upsampled IV textures. As Table 9.5 shows, the *optimized encoder setup* achieves depth bit rate savings of 60% compared to depth distortion-based encoding. For the *optimized encoder setup*, coding losses due to the mismatches can be mitigated by reducing the SVDC-depth distortion ratio. As Figure 9.6 (e) and Table 9.6 show, the depth bit rate decreases by about 2.2% when changing ρ from $2^{5.5}$ to 2^4. Considering other encoder setups at their optimal values for ρ shows that, compared to the *optimized encoder setup*, the *BS off* and the Extrapolation setups achieve depth bit rate savings of 3.1% and 0.9%, respectively. However, for the Extrapolation setup, depth bit rate deltas vary significantly for different test sequences, as Figure 9.6 (a) shows.

Major differences between the VSRS in general mode (Appendix A.2.2) and the LCR are that the former uses a combination of forward and backward warping and does not perform background suppression. As Table 9.5 shows, the *optimized encoder setup* achieves depth bit rate savings of 69.2% compared to depth distortion-based encoding. As well as for the VSRS in 1D mode, reducing the SVDC-depth distortion ratio ρ achieves further depth bit rate savings for the *optimized encoder setup*. As Figure 9.6 (f) and Table 9.6 show, they are about 2.6%

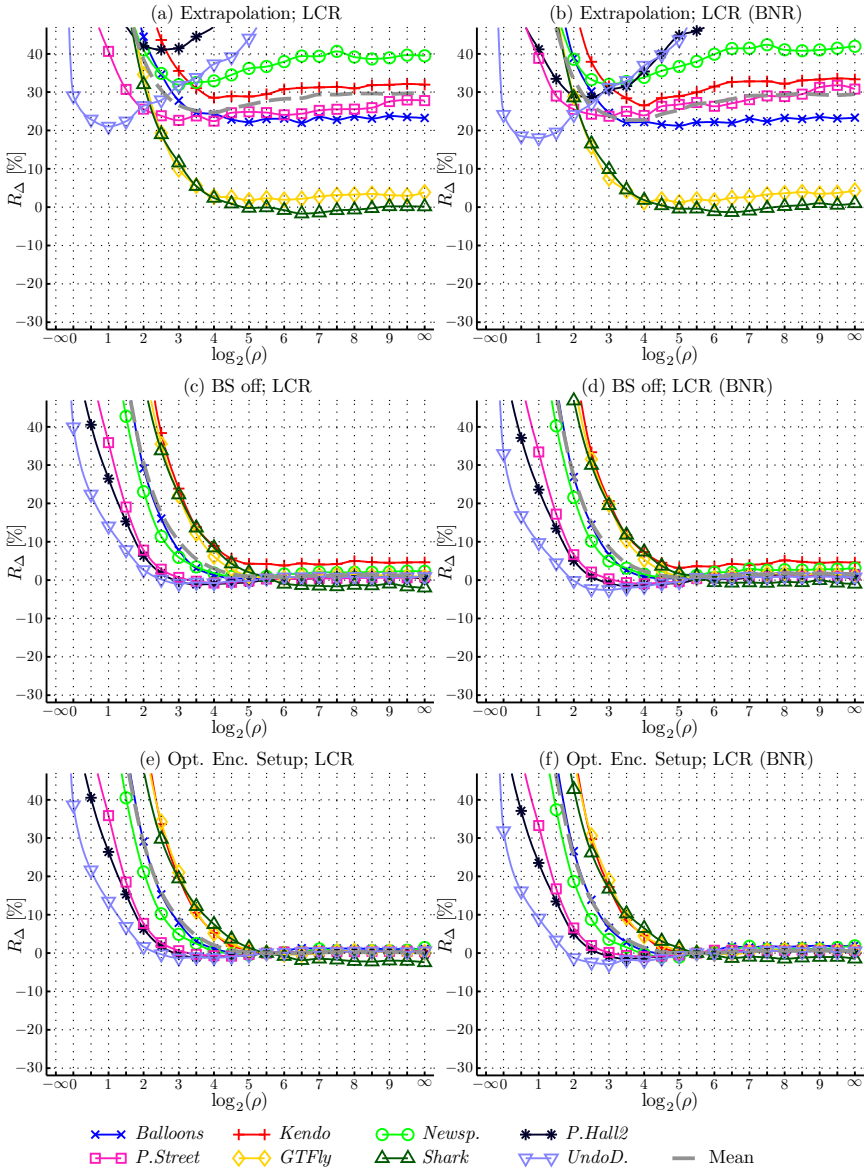


Figure 9.5: Depth bit rate deltas; Tested: Different view synthesis methods, encoder setups, and ratios ρ; Reference: The respective view synthesis method, and the *optimized encoder setup* with $\rho = 2^{5.5}$.

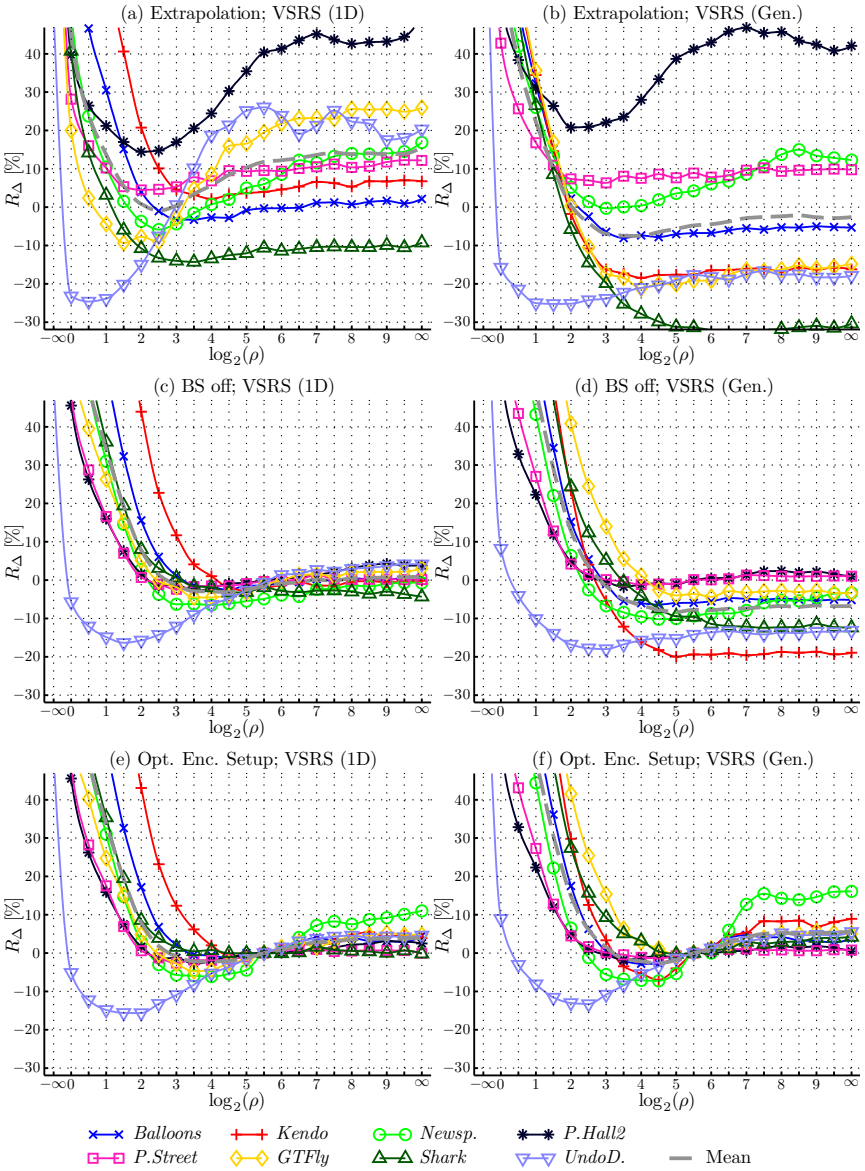


Figure 9.6: Depth bit rate deltas; Tested: Different view synthesis methods, encoder setups, and ratios ρ; Reference: The respective view synthesis method, and the *optimized encoder setup* with $\rho = 2^{5.5}$.

	$R_\Delta[\%]$											
	LCR			LCR (BNR)			VSRS (1D)			VSRS (Gen.)		
	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.
$\log_2(\rho)$	4.0	6.0	5.5	3.5	5.0	5.0	2.5	4.5	4.0	4.0	5.0	4.5
Balloons	+24.3	+0.5	0	+22.1	+0.5	+0.4	− 1.0	−2.6	−0.6	− 7.4	− 6.0	−2.8
Kendo	+28.5	+3.8	0	+28.4	+3.2	+0.4	+10.1	−2.0	+2.0	−18.4	−20.0	−7.0
Newsp.	+33.0	+1.6	0	+32.8	+1.0	−1.1	− 5.8	−6.1	−6.1	+ 0.4	−10.0	−7.2
P.Hall2	+47.1	+0.4	0	+31.6	−0.8	−0.7	+14.8	−0.9	−2.1	+28.0	− 1.0	−0.8
P.Street	+22.4	+0.3	0	+25.0	−0.4	−0.4	+ 4.7	−1.5	−1.9	+ 7.6	− 1.0	−1.4
GTFly	+ 2.8	+0.5	0	+ 4.3	+1.0	+0.6	− 9.0	−4.0	−4.4	−21.2	− 4.0	+1.2
Shark	+ 2.4	−1.0	0	+ 4.6	+1.7	+1.4	−13.2	−3.2	+0.8	−27.8	− 9.4	+0.4
UndoD.	+37.3	+0.8	0	+32.9	−0.5	−0.7	− 7.6	−4.8	−5.3	−21.0	−15.3	−3.4
Mean	+24.7	+0.9	0	+22.7	+0.7	0.0	− 0.9	−3.1	−2.2	− 7.5	− 8.3	−2.6

Table 9.6: Depth bit rate delta R_Δ; Tested: Different view synthesis methods and encoder setups; Reference: The *optimized encoder setup*.

when changing the ratio ρ from $2^{5.5}$ to $2^{4.5}$. In contrast to the VSRS 1D mode, disabling background suppression and view combination reduces the depth bit rate significantly, which can be explained with the different view combination methods applied by the LCR and VSRS in general mode. More specifically, Figures 9.6 (d) and (b) show that the *BS off* and the Extrapolation encoder setups using ratios ρ of 2^5 and 2^4, respectively, reduce the depth bit rate by about 8% in comparison to the *optimized encoder setup*. However, it can be observed that using the Extrapolation setup leads to losses for some sequences (e.g. *PoznanHall2*).

9.2.3 Conclusion

In comparison to depth distortion-based encoding, SVDC-based encoding using the *optimized encoder setup* achieves depth bit rate savings of about 76.8%, 60%, and 69.2% when using the LCR with boundary noise removal, the VSRS in 1D mode, and the VSRS in general mode, respectively. Depth bit rate savings decrease the more the receiver-side renderer deviates from the LCR. Coding losses due to these deviations can be mitigated, depending on the receiver-side renderer by decreasing the SVDC-depth distortion ratio ρ, by disabling background suppression in view combination, or view combination entirely. When optimizing for individual view synthesis methods, these modifications achieve about 3.1% and 8.3% depth bit rate savings in comparison to the *optimized encoder setup* for the VSRS in 1D mode, and the VSRS in general mode, respectively. When considering all view synthesis methods jointly[3], a reasonable setup is disabling background suppression and using an SVDC-depth distortion ratio of $\rho = 2^5$. This increases the depth bit rate by 1% and 0.7% for the LCR and the LCR with boundary noise removal, respectively, while it decreases the depth bit rate for the VSRS in 1D mode and the VSRS in general mode by 2.6% and 8.3%, respectively. In summary, the *optimized encoder*

[3] See Table F.4, which compares the depth bit rate savings when an unique value of ρ is selected for all view synthesis methods such that it maximizes the depth bit rate savings averaged over the receiver-side view synthesis methods.

setup achieves between 60% to 77.8% depth bit rate savings for all evaluated renderers. SVDC-based encoding is thus also effective when the receiver-side view synthesis algorithm does not match the LCR, which is used for SVDC computations.

9.3 Comparison with Estimation Methods

So far this thesis compared SVDC-based encoding with depth distortion-based encoding. A further alternative is encoding based on estimated SVD values. For this purpose, methods have been proposed that estimate the SVD from the depth distortion without considering the entire view synthesis process (see Section 2.2.3.1). This section evaluates how SVDC-based encoding performs in comparison to these methods. More specifically, it optimizes and analyzes a least squares method (Section 9.3.1), a method based on the texture gradient (Section 9.3.2) and a method based on shifting of samples (Section 9.3.3). Finally, Section 9.3.4 compares the RD performance and complexity of the estimation methods and the RM.

Related works: [Kim 10, Kim 09, Oh 11] describe the evaluated methods. Evaluations considering SVD estimation instead of the SVDC have been performed by MPEG [Oh 12c, Wang 12a, Jung 12a, Oh 12e] and others [Ma 13]. Some of them replace the SVDC with other metrics only in lower-level selection stages [Wang 12c, Oh 12b, Oh 12d, Kim 15]. [Fank 12] evaluates several SVD estimation methods in a transmission scenario considering mobile applications. This section differs from these publications as it uses a different set of estimation methods, optimizes them to the same extend, and compares them to SVDC-based encoding using the latest HTM software, the *optimized encoder setup* and different test conditions.

9.3.1 Least Squares Method

Kim et al. [Kim 09] propose a method for depth encoding that exploits their observation that the SVD is approximately proportional to the disparity distortion. Their method determines a factor α between the disparity distortion and the SVD with a least squares fit (and will thus be called LS method); then, it uses this factor in RD optimization to estimate the SVD.

The LS method derives α in two steps. For a vector $\mathbf{p}$ with horizontal disparities, the first step derives a vector $\mathbf{d_E}$ representing associated SVDs. More specifically, for each element p_i of $\mathbf{p}$, it derives the associated element $d_{E,i}$ of $\mathbf{d_E}$ as sum of squared differences between the IV texture $s_T(x)$ and the IV texture shifted by p_i, i.e.

$$d_{E,i} = \sum_{x \in B_I} \left[s_T(x) - s_T(x - p_i) \right]^2 \tag{9.1}$$

with B_I denoting all sample positions of the IV texture. The second step determines α as

$$\alpha = \frac{\mathbf{d_E}^T \mathbf{p}}{\mathbf{p}^T \mathbf{p}}, \tag{9.2}$$

		$R_\Delta[\%]$					$T_\Delta[\%]$				
N_P	→	1	2	4	7	15	1	2	4	7	15
	1	−53.3	−55.0	−55.2	−52.8	−43.4	+4	+8	+13	+18	+31
	2	−56.2	−56.4	−56.2	−53.7	−44.9	+2	+4	+ 7	+10	+15
w_B	4	−55.2	−55.7	−55.5	−53.7	−45.1	+1	+3	+ 6	+ 8	+11
	8	−51.1	−51.7	−51.8	−50.6	−43.4	+1	+3	+ 5	+ 6	+ 9
	64	−20.2	−20.3	−21.0	−20.9	−19.3	−1	0	+ 1	+ 3	+ 5
	256	− 2.0	− 1.8	− 2.3	− 3.5	0.0	−5	−4	− 4	− 3	0

Table 9.7: Least squares method; Depth bit rate deltas R_Δ averaged over sequences; Tested: Different block sizes w_B and maximal disparities ν_P; Reference: w_B = 256 and ν_P = 15.

i.e. by a least squares fit. When encoding, the LS method employes α to estimate the SVD d_E related to the absolute disparity difference $|\tilde{s}_P(x) - s_P(x)|$ of the currently coded block B as

$$d_E = \beta \cdot \sum_{x \in B} \alpha \cdot |\tilde{s}_P(x) - s_P(x)| \tag{9.3}$$

with β denoting a factor to consider view combination (e.g. 0.5 for the central SV); and $\tilde{s}_P$ and s_P denoting the coded and the uncoded disparity, respectively. In summary, the LS method estimates the SVD by simply scaling the absolute disparity difference, which is based on the assumption that the linear relationship is also given locally.

Selection of Optimal Parameters

The LS method has three parameters that can be varied: 1) the IV texture regions for which it calculates individual factors α; 2) the disparities p_i it uses for shifting to determine the SVDs $d_{E,i}$; and 3) the IV texture s_T and its shifted version, which can be coded or uncoded or combinations thereof. Furthermore, the encoder can use a depth distortion term; the ratio ρ between the SVD estimate and the depth distortion is thus a fourth parameter. This section analyzes how these parameters should be selected to maximize the RD performance of the LS method.

For complexity reasons, [Kim 09] proposes to calculate α from the entire first picture of a sequence and to use a vector of disparities $\mathbf{p}$ equal to $[1, 2, 3, ..., 30]^T$. However, authors also assert that using local and smaller IV texture parts improves the encoding performance. Therefore, the LS method, as evaluated in this thesis, determines an individual α for each $w_B \times w_B$ block B of the IV texture. Furthermore, to account for the smaller blocks size and locality, the least squares fit uses disparities relative to the center position of B, i.e. a vector of disparities $\mathbf{p}$ equal to $[-\nu_P, ..., -1, 0, 1, , ..., \nu_P]^T$. This requires to use absolute values of p_i in (9.2).

The optimal values of w_B and ν_P have been experimentally determined using the LS method as described above in selection stage C, but by considering the coded IV texture $\tilde{s}_T$, as this achieved highest depth bit rate savings in SVDC-based encoding (when no hiding effect occurs; see Section 8.1.4). More specifically, for calculating (9.3), the LS method employed the difference of the uncoded and the coded IV texture, i.e. $[s_T(x) - \tilde{s}_T(x - p_i)]$. Table 9.7 shows the coding results using the combination of w_B equal to 256 and ν_P equal to 15 as reference. Compared to the reference, the best combination—w_B equal to 2 and ν_P equal to 2—reduces the

	$R_\Delta[\%]$							
	Least Squares Method			VSD Method		Shifting Method		
	C	*R*	*R+W*	*C*	*C+W*	*C*	*R*	*C+W*
Balloons	0	− 6.6	− 9.1	0	+ 0.7	0	+ 2.9	− 0.9
Kendo	0	−10.6	−11.8	0	− 1.0	0	+ 1.8	− 0.4
Newsp.	0	−11.3	−14.4	0	− 5.4	0	+ 1.6	− 4.1
P.Hall2	0	−12.1	−16.2	0	− 7.5	0	+ 6.7	− 5.5
P.Street	0	− 8.7	− 9.4	0	− 3.0	0	+11.6	− 3.1
GTFly	0	− 6.7	− 6.6	0	− 0.4	0	+ 5.8	− 5.4
Shark	0	−11.1	−10.2	0	− 2.8	0	+ 6.2	− 5.9
UndoD.	0	−13.3	−16.4	0	−11.6	0	+ 8.9	−11.0
Mean	0	−10.0	−11.8	0	− 3.9	0	+ 5.7	− 4.5

Table 9.8: Depth bit rate delta R_Δ; Tested: Setups of different VSD estimation methods; Reference: Setup *C* of the respective VSD estimation method.

depth bit rate by 56.4%. In particular, it outperforms a block size w_B equal to 1. A sample-wise estimation is thus less reliable, although it allows to adapt α with lowest granularity.

The optimal test case found above, called test case *R* in the following, compared the uncoded IV texture to the shifted coded IV texture, i.e. $s_T(x) - \tilde{s}_T(x - p_i)$ to calculate (9.3). A further option is to compare the coded IV texture to the shifted coded IV texture, i.e. $\tilde{s}_T(x) - \tilde{s}_T(x - p_i)$. This option has been evaluated as test case *C*. However, as Table 9.8 shows, it is not beneficial since the depth bit rate in test case *R* is about 10% lower.

Since an additional depth distortion term is beneficial in SVDC-based encoding, the question is whether this is also true for the LS method. For investigation, Figure 9.7 (a) shows depth bit rate savings for different SVD estimate-depth distortion ratios ρ relative to the depth bit rate of the test case *R* (i.e. ρ equal to ∞). The test case with a ratio ρ equal to $2^{3.5}$ (called *R+W* in the following) is optimal and leads to a depth bit rate reduction of about 2%, as shown in Figure 9.7 (a), and to about 12% depth bit rate reduction in comparison to test case C.

In summary, this section shows that the selection of a small block size (i.e. 2×2) and a small disparity range (i.e. -2 to 2) provides depth bit rate reductions of more than 50% compared to parameters used in [Kim 09]. Similar to SVDC-based encoding, using an optimal depth distortion weight and comparing the uncoded IV texture with the coded IV texture achieves further depth bit rate reductions of about 12%. Optimizations enable a fair comparison of the LS method with SVDC-based encoding in Section 9.3.4.

Complexity

The computational complexity of the LS method is related to the derivation of α and to the actual SVD estimation. As the LS method performs the former only once per IV texture block B and in the optimal case with ν_P equal to 2, its complexity is rather small. The SVD estimation with (9.3), however, is repeated multiple times in RD optimization. For w_B equal to 2, it requires 8.5 operations and memory accesses per IV depth sample and is thus only marginally more complex than SSD computation, which requires 7 (see Tables C.9 and C.12).

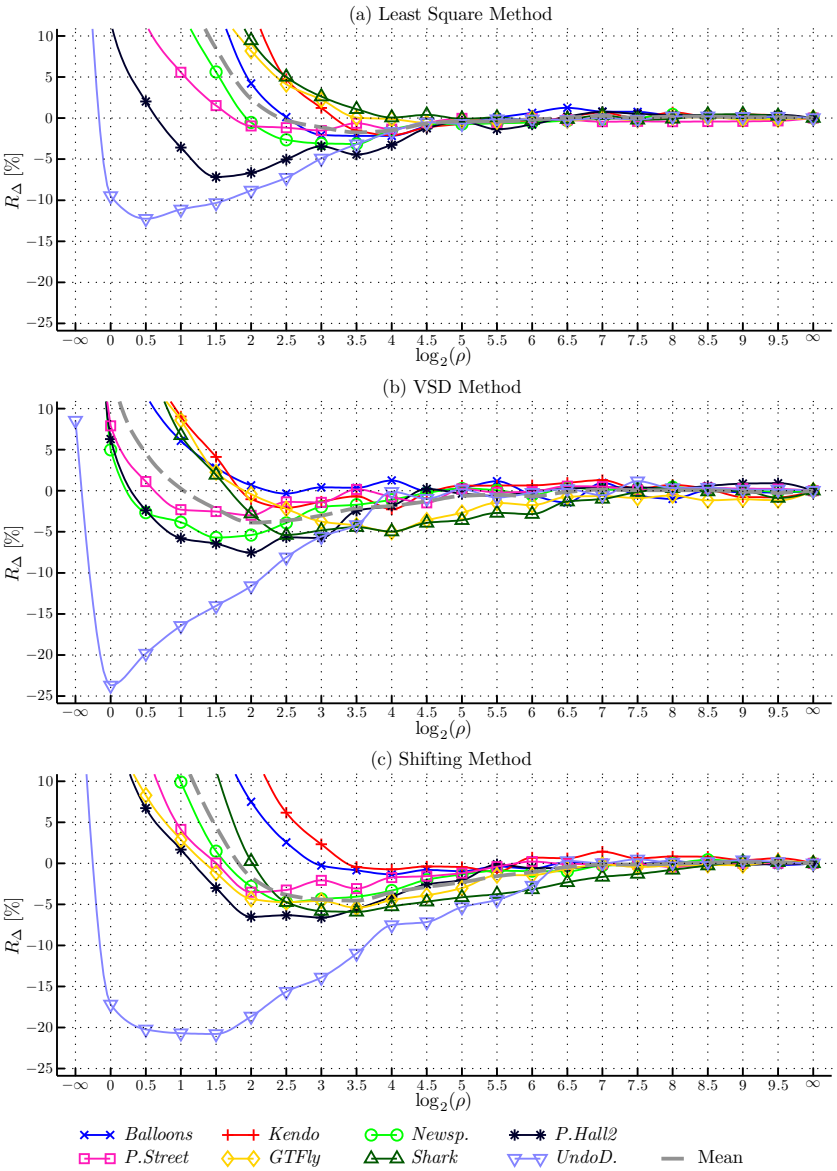


Figure 9.7: Depth bit rate deltas averaged over sequences; Tested: Different distortion derivation methods and ratios ρ; Reference: The respective distortion derivation methods with $\rho = \infty$.

9.3.2 Texture Gradient Approach

[Oh 11] proposes an SVD estimation method based on the local IV texture gradient. Known as VSD, the method is part of 3D-HEVC reference software HTM and, for complexity reduction, used under JCT-3V's test conditions as complement to the SVDC in mode selection stages P (intra pre-selection), D (intra wedge), and R (inter RQT) as defined in Table 7.1.

The derivation of the VSD method in [Oh 11] is based on the assumption of a linear SV texture interpolation function and leads to

$$d_E = \sum_{x \in B} \left(\alpha(x) \cdot \beta(x) \cdot [\tilde{s}_P(x) - s_P(x)] \right)^2 . \tag{9.4}$$

The SVD d_E related to the IV depth block B is thus the sum of weighted and squared disparity differences $\tilde{s}_P(x) - s_P(x)$ in B. Weights are the factors $\alpha(x)$, and $\beta(x)$. The factor $\alpha(x)$ approximates the horizontal gradient of the coded IV texture and is according to [Oh 11]

$$\alpha(x) = \frac{1}{2} \cdot \left(|\tilde{s}_T(x) - \tilde{s}_T(x-1)| + |\tilde{s}_T(x) - \tilde{s}_T(x+1)| \right) . \tag{9.5}$$

The factor $\beta(x)$ is actually not described in [Oh 11], but only used by HTM software's VSD implementation, which is provided by the authors of [Oh 11]. It is

$$\beta(x) = \begin{cases} 1/4 & \text{if } s_M(x) < 16 \\ 1 & \text{if } s_M(x) > 96 \text{ [sic]} \\ 1/4 + [s_M(x) - 16]/128 & \text{else} \end{cases} \tag{9.6}$$

and thus decreases the weight of IV texture samples that are closer to the background, which might be beneficial as these samples are more likely occluded.

In summary, the VSD method provides the squared disparity distortion weighted with the squared local texture gradient. This way, the IV depth is better preserved when collocated to IV texture edges, thus in regions where it is important for view synthesis.

Selection of Optimal Parameters
This thesis analyzes the VSD method as implemented by authors of [Oh 11] in the HTM software. However, as for the LS method, an additional depth distortion term is used. Figure 9.7 (b) shows that a ratio of $\rho = 2^2$ is optimal and leads to a depth bit rate decrease of 4% compared with encoding based on the SVD estimate only (i.e. $\rho = \infty$). Table 9.8 summarizes results, in which encoding without and with the optimal depth distortion weight are test cases *C* and *C+W*, respectively. As the VSD method is based on the texture gradient in a single IV texture, a comparison of the coded with the uncoded IV texture, i.e. a test case R, is not possible.

Complexity
In [Oh 11] and the HTM software, the VSD method calculates factors $\alpha(x)$ and $\beta(x)$ in RD optimization, i.e. when calculating the SVD d_E with (9.4). However, as for the LS method, the VSD method can derive $\alpha(x)$ and $\beta(x)$ in an initialization step to avoid repeated computations. With this approach, the VSD method's complexity is mainly related to (9.4), which

requires about 11 operations and memory accesses (see Table C.10), i.e. four more than SSD computation.

9.3.3 Shifting Method

The LS and the VSD methods use the disparity distortion only indirectly for SVD estimation by multiplying it with a weighting factor. In contrast, [Kim 10] presents a method that employs the disparity distortion directly to identify the IV texture sample that is shifted to the erroneous SV position of a current IV texture sample. (For complexity reduction, [Kim 10] presents a further method based on an auto-regressive model, which is not considered here as [Kim 10] asserts that the shifting method is more accurate.) The SVD estimate d_E for a block B is

$$\Delta s_P(x) = \tilde{s}_P(x) - s_P(x) \tag{9.7}$$

$$d_E = \sum_{x \in B} \left[s_T(x) - s_T\left(x - \Delta s_P\left[x\right]\right)\right]^2 . \tag{9.8}$$

Thus, considering that the current IV texture sample $s_T(x)$ is displaced by $\Delta s_P(x)$ in the SV, the shifting method assumes that the SV texture sample at the displaced SV position corresponds to the IV texture sample with the same displacement in the IV texture.

Selection of Optimal Parameters
To optimize the shifting method, Table 9.8 provides coding results for three test cases. Test case *C* uses the coded IV texture, i.e. $\tilde{s}_T(x) - \tilde{s}_T\left(x - \Delta s_P\left[x\right]\right)$ in (9.8), whereas test case *R* uses the difference of the coded and uncoded IV texture, i.e. $\tilde{s}_T(x) - s_T\left(x - \Delta s_P\left[x\right]\right)$. In comparison to test case C, test case R leads to a depth bit rate increase of 6%. Test case *C+W* is based on test case *C* and uses an SVD estimate-depth distortion ratio of ρ equal to $2^{3.5}$, which is optimal and leads to a depth bit rate decrease of 5% as Figure 9.7 (c) shows.

Complexity
Calculating the SVD estimate with (9.8) requires about 13 operations and memory accesses per IV depth sample (see Table C.11), i.e. twice the amount of SSD computation. However, the memory accesses to the IV texture are irregular as they depend on the disparities, which might increase the computational complexity further [Kim 10].

9.3.4 Comparison of Methods

So far the estimation methods have only been individually optimized. A question left open is how SVDC computation compares to them. For this, Figure 9.8 shows RD curves obtained with the optimized estimation methods. It can be observed that performance differences depend strongly on the sequence. Furthermore, Table 9.9 shows the depth bit rate savings of SVDC-based encoding with the *optimized encoder setup* when using the different estimation methods as reference. Compared to the VSD, the shifting (SH), and the LS method the depth bit rate decreases by 65.3%, 47.4% and 46.4%, respectively; the depth encoding time increases by about 47%, 39%, and 43%. The increased RD performance comes thus at an increased computational

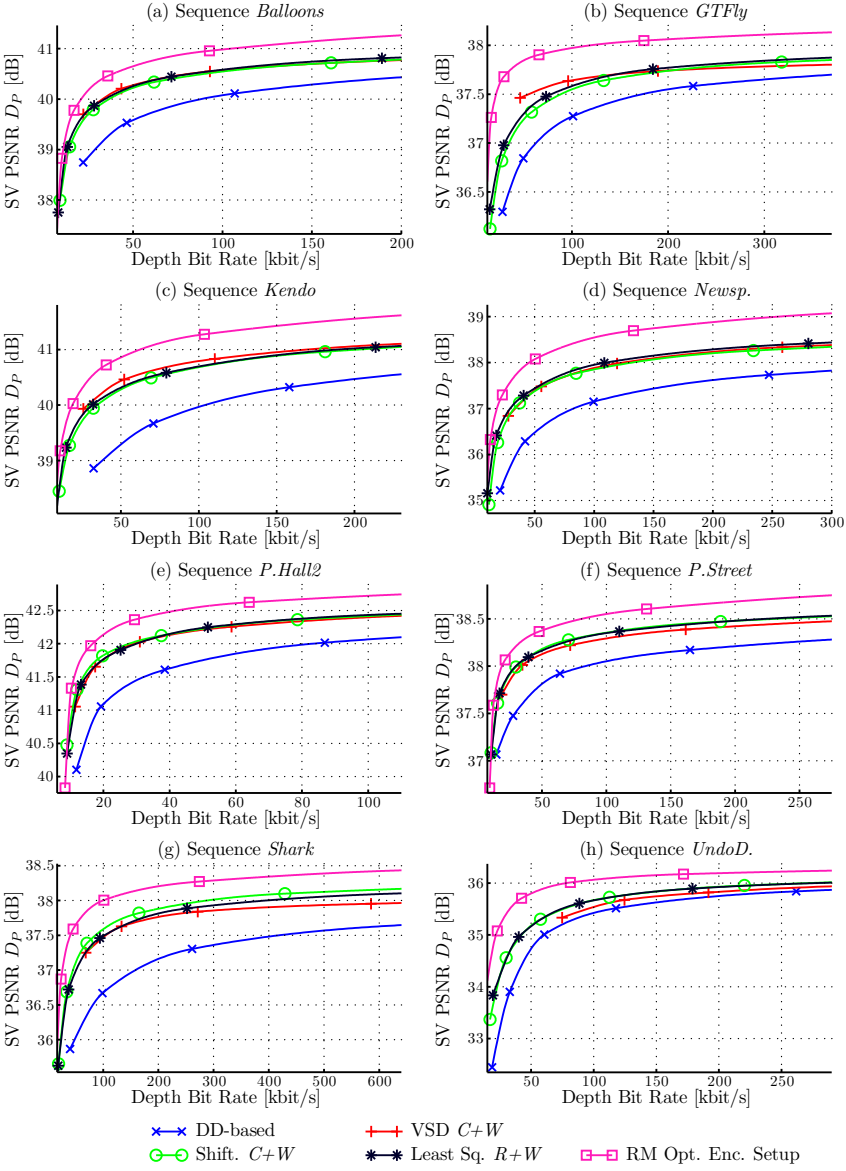


Figure 9.8: RD performance of the evaluated distortion derivation methods.

Reference	$R_\Delta[\%]$ DD	VSD *C+W*	SH *C+W*	LS *R+W*	$T_\Delta[\%]$ DD	VSD *C+W*	SH *C+W*	LS *R+W*
Balloons	−78.2	−60.0	−39.5	−32.9	+ 78	+49	+39	+39
Kendo	−83.8	−59.6	−49.8	−47.2	+ 57	+33	+33	+33
Newsp.	−81.1	−65.8	−51.9	−45.1	+ 83	+48	+38	+41
P.Hall2	−65.8	−45.1	−30.2	−32.8	+ 98	+59	+48	+54
P.Street	−73.9	−59.4	−36.1	−37.6	+101	+53	+40	+48
GTFly	−85.0	−83.2	−71.4	−68.3	+ 88	+47	+37	+41
Shark	−86.2	−77.8	−47.6	−54.5	+ 71	+40	+40	+42
UndoD.	−68.6	−71.3	−52.8	−52.4	+ 80	+43	+37	+44
Mean	−77.8	−65.3	−47.4	−46.4	+ 82	+47	+39	+43

Table 9.9: Depth bit rate delta R_Δ and depth encoding time delta T_Δ; Tested: *optimized encoder setup*; Reference: Optimized setups of different distortion derivation methods.

complexity. In summary, the evaluation shows that although SVD estimation achieves already significant depth bit rate reductions, it does not outperform SVDC-based encoding[4].

9.3.5 Conclusion

This section optimized three SVD estimation methods and compared them with SVDC computation. In general, the optimization analyzed whether comparing uncoded to coded IV texture is beneficial and determined the optimal SVD estimate-depth distortion ratios. For the LS method, it determined optimal block sizes and disparity ranges. In conclusion, SVDC-based encoding reduces the depth bit rate by 46.4% in comparison to encoding with the LS method, which performed best among SVD estimation methods. Exact SVDC computation achieves thus significant coding gains also in comparison to SVD estimation, however, with an increased computational complexity: with respect to the LS method, the depth encoding time increases by about 43%.

9.4 Alternative Distortion Metrics

Although commonly used, the PSNR might under- or overestimate certain kinds of SVDs. For an assessment that goes beyond the PSNR, this section provides SV quality gains and depth bit rate savings with respect to the Structural Similarity (SSIM) index and to Pratt's Figure of Merit (FOM), which measures the preservation of edges. It then analyzes how coding gains with respect to the alternative metrics depend on a depth distortion weight.

[4]Table F.5 shows that this is also true when the encoder uses the estimation methods in all selection stages: Additional coding gains are small, whereas complexity increases significantly.

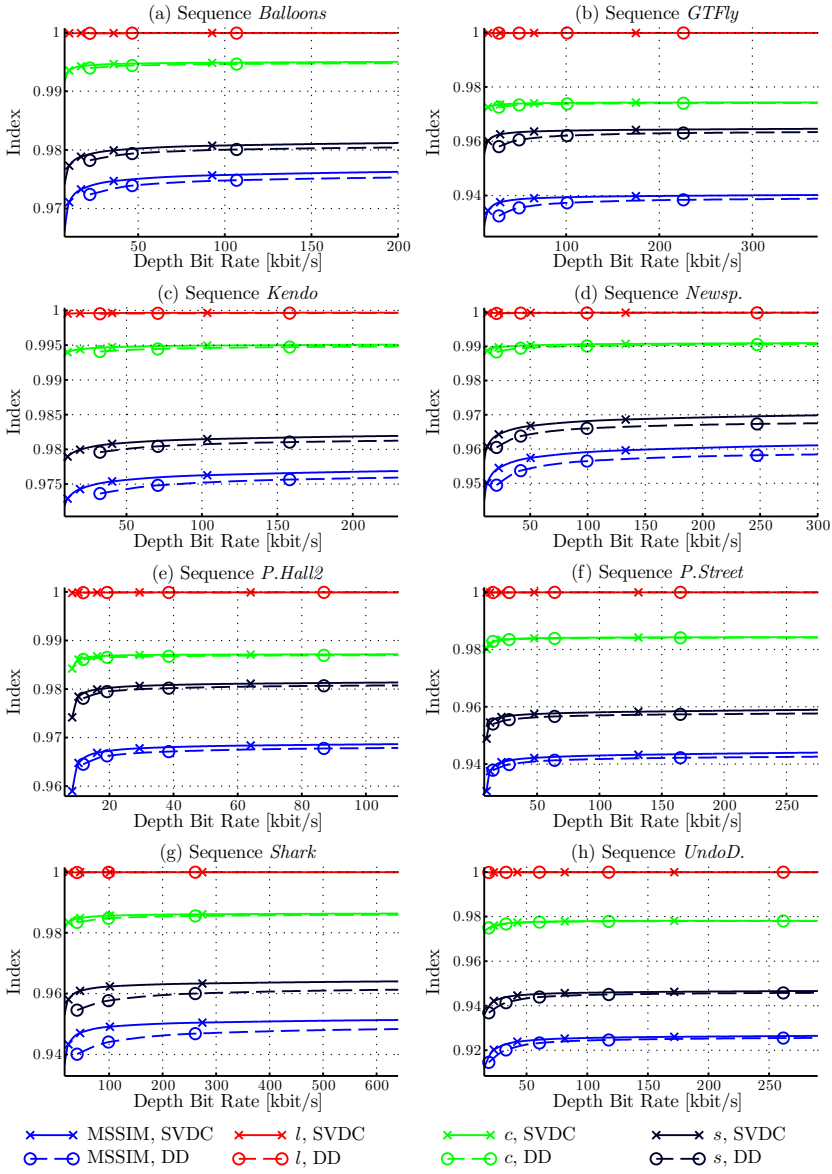


Figure 9.9: RD performance with respect to the MSSIM and its components. SVDC-based encoding (*optimized encoder setup*) vs. DD based encoding.

	$D_{\Delta P}$	MSSIM$_\Delta$ /10^{-2}	l_Δ /10^{-2}	c_Δ /10^{-2}	s_Δ /10^{-2}	FOM$_\Delta$ /10^{-2}
Balloons	+0.9	+0.102	+0.002	+0.027	+0.079	+0.277
Kendo	+1.3	+0.107	+0.006	+0.034	+0.076	+0.477
Newsp.	+1.4	+0.270	+0.005	+0.048	+0.226	+0.358
P.Hall2	+0.8	+0.087	+0.001	+0.026	+0.065	+0.627
P.Street	+0.5	+0.127	+0.001	+0.016	+0.114	+0.170
GTFly	+0.6	+0.185	+0.002	+0.032	+0.163	+0.197
Shark	+1.0	+0.366	+0.002	+0.056	+0.336	+0.411
UndoD.	+0.6	+0.118	+0.002	+0.015	+0.108	+0.050
Mean	+0.9	+0.170	+0.003	+0.032	+0.146	+0.321

Table 9.10: Delta of the SV quality with respect to different metrics. Tested: SVDC-based encoding using the *optimized encoder setup*; Reference: Depth distortion-based encoding.

9.4.1 Structural Similarity Index

The SSIM index [Wang 04] is asserted to be more representable for the visual perceived quality than e.g. the PSNR. It is the product of a luminance index l, a contrast index c, and a structure index s, which measure the local similarity of two pictures (see Appendix E.1.2). Figure 9.9 and Table 9.10 compare SVDC- and depth distortion-based encoding with respect to the mean SSIM (MSSIM) and its components. SVDC-based encoding uses the *optimized encoder setup*. Considering indices in Figure 9.9 individually shows that in both test cases the l index is close to 1 for all sequences. Depth coding affects the local average of the SV texture samples only insignificantly. Consequently, SVDC-based encoding increases the l index only slightly, as shown in Table 9.10, by $3 \cdot 10^{-5}$ in average over the evaluated depth bit rates. As the c indices in Figure 9.9 indicate, depth coding affects the local variance more. SVDC-based encoding achieves thus higher gains of about $3.2 \cdot 10^{-4}$. Finally, the s index indicates that depth coding reduces the local correlation of the tested and the reference SV texture most; it decreases significantly with decreasing rates. SVDC-based encoding can thus achieve gains of $1.46 \cdot 10^{-3}$.

In summary, SVDC-based encoding increases the average MSSIM by $1.7 \cdot 10^{-3}$, which corresponds, as Table 9.11 indicates, to depth bit rate savings of 60%. The gain is majorly caused by an increase of the s index and thus by an increasing local correlation of the tested and the reference SV texture.

9.4.2 Pratt's Figure of Merit

Edges in the SV texture are on the one hand important for the human perception and on the other hand, when collocated to SV positions with high depth gradients, prone to SVDs caused by depth coding. How they are affected by SVDC-based encoding is thus of particular interest and evaluated in this section using a measure for the similarity of edge positions in two pictures, called Pratt's Figure of Merit (FOM) [Prat 78], and described in Appendix E.1.3

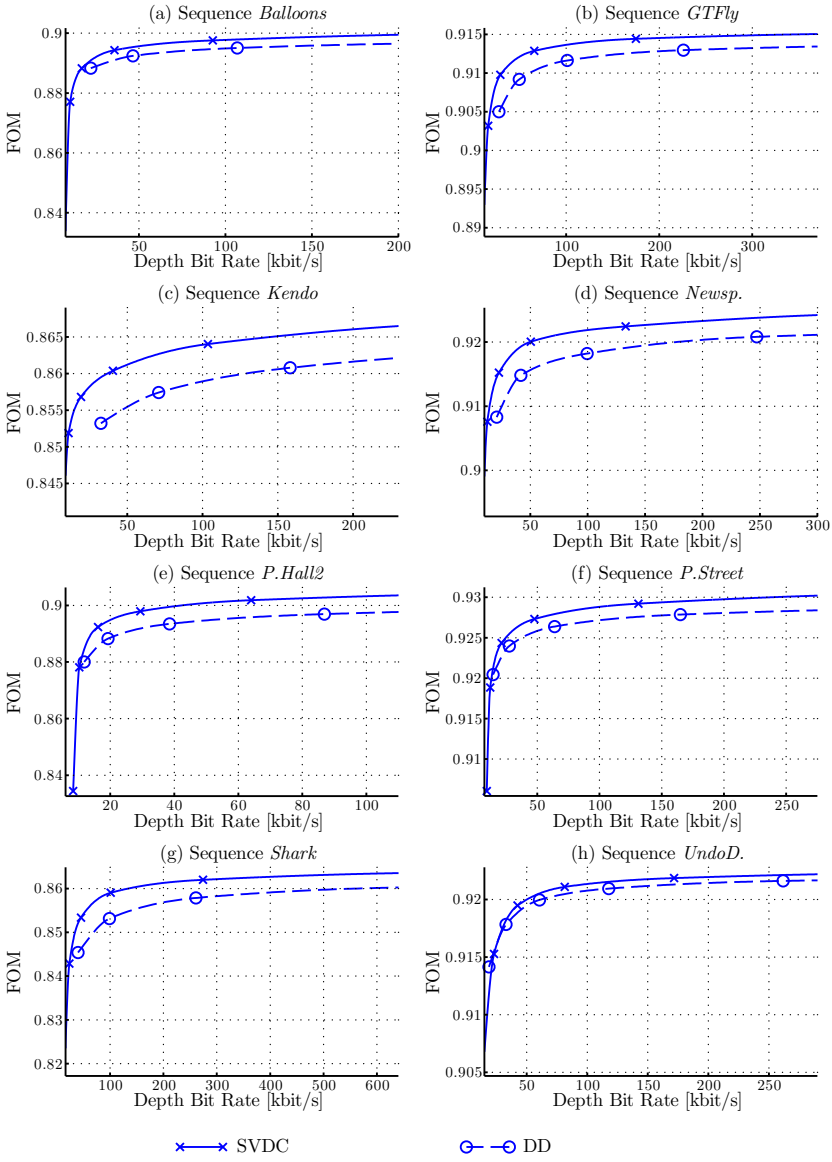


Figure 9.10: RD performance with respect to the FOM. SVDC-based encoding (*optimized encoder setup*) vs. DD based encoding.

	$R_\Delta[\%]$		
	PSNR	MSSIM	FOM
Balloons	−78.2	−61.1	−58.7
Kendo	−83.8	−66.3	−71.1
Newsp.	−81.1	−63.8	−59.8
P.Hall2	−65.8	−47.9	−52.0
P.Street	−73.9	−55.5	−43.9
GTFly	−85.0	−70.0	−53.8
Shark	−86.2	−81.0	−62.2
UndoD.	−68.6	−34.7	−13.1
Mean	−77.8	−60.0	−51.8

Table 9.11: Depth bit rate delta R_Δ with respect to different metrics. Tested: SVDC-based encoding using the *optimized encoder setup*; Reference: Depth distortion-based encoding.

Figure 9.10 shows how the FOM depends on the depth bit rates for depth distortion-based and SVDC-based encoding. SVDC-based encoding achieves FOM gains in particular at medium depth bit rates. Gains averaged over depth bit rates are about $0.3 \cdot 10^{-2}$, as shown in Table 9.10, and correspond to depth bit rate saving of about 51.8%, as shown in Table 9.11.

9.4.3 Depth Distortion Weight

In the evaluations discussed in Sections 9.4.1 and 9.4.2, SVDC-based encoding used an SVDC-depth distortion ratio of $\rho = 2^{5.5}$, which Section 8.3 identified as optimal with respect to the SV texture PSNR. This raises the question whether a different ratio is optimal with respect to the other distortion metrics. For this, Figure 9.11 shows depth bit rate savings with respect to the SSIM and the FOM in dependency of $\log_2(\rho)$ and using the *optimized encoder setup* (with $\rho = 2^{5.5}$) as reference. It can be observed that, as well as for the SV texture PSNR, the optimal value of ρ depends strongly on the tested sequence. On average over sequences, the optimal ρ with respect to the MSSIM is with $2^{4.5}$ lower as for the PSNR and leads to depth bit rate savings of about 1%. In contrast, depth bit rate savings with respect to the FOM do not increase for values of ρ different from $2^{5.5}$.

9.4.4 Conclusion

This section shows that although SVDC computation considers the SSD of the SV textures, not only the SV texture's PSNR increases, but also its MSSIM and FOM. SVDC-based encoding thus preserves structure and edges of the SV texture better than depth distortion-based encoding, which implies that also the perceived SV texture quality increases. However, depth bit rate savings with respect to the MSSIM and the FOM are with 60% and 51.8%, respectively, less than depth bit rate savings with respect to the SV PSNR.

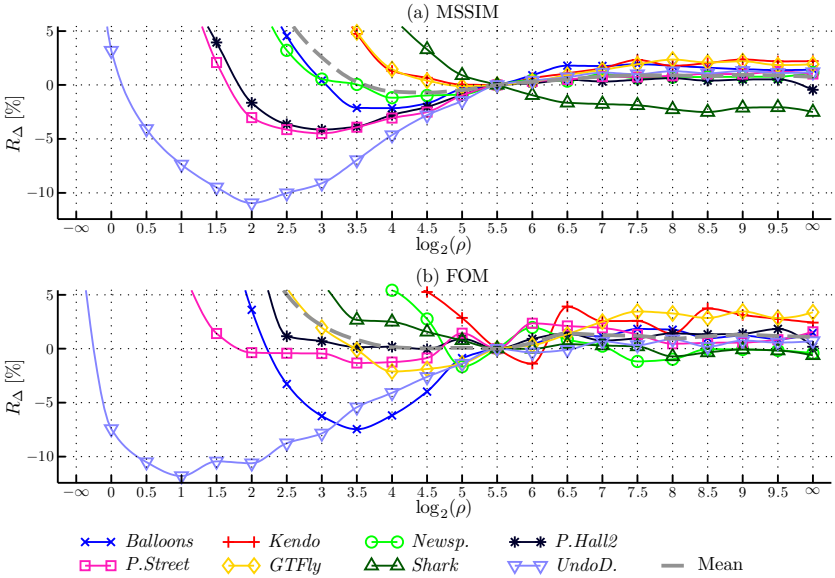


Figure 9.11: Depth bit rate deltas averaged over sequences; Tested: Different metrics and ratios ρ; Reference: The respective metric with $\rho = 2^{5.5}$.

9.5 Sample Pictures

So far, evaluations presented coding results in terms of depth bit rate savings and SV quality gains with respect to different objective metrics. To provide an impression what these results mean for the subjectively perceived SV quality, Figure 9.12 shows the SV textures rendered from the first pictures of the two IV textures and the two IV depths maps of the *PoznanHall2* sequence (Appendix F.10 provides further examples). More specifically, three test cases are shown in which the SV texture is rendered from IV texture and IV depth that are

- uncoded (*Uncoded* test case).
- coded based on the depth distortion only and using QPs 30 and 50 for the IV textures and IV depth maps, respectively (*Depth Dist.* test case).
- coded using the *optimized encoder setup*, a QP of 30 for the IV textures and an adapted Lagrange multiplier and QP for the IV depth maps such that they require approximately the same number of bits as in the *Depth Dist.* test case (*SVDC* test case).

Due to the selection of the depth QP, the SV textures are compared at the depth bit rate at which the highest quality differences occur. Moreover, as the right and left IV depth map's first pictures are intra-picture and inter-picture predicted, respectively, SV textures are exemplary for both prediction types.

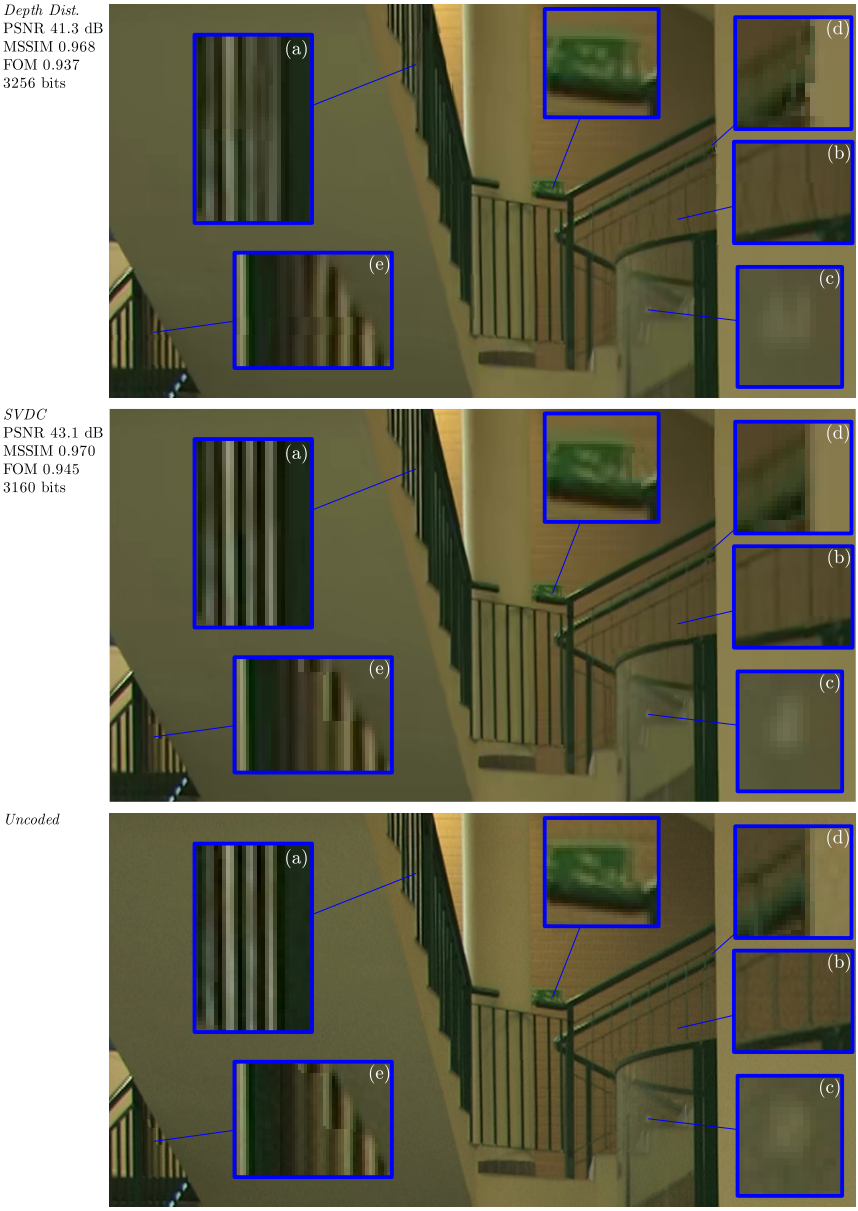


Figure 9.12: SVDC- vs. depth distortion-based encoding; Sequence: *P.Hall2*.

As annotated in Figure 9.12, the PSNRs in the *Depth Dist.* test case and the *SVDC* test case are 41.3 dB and 43.1 dB in comparison to the *Uncoded* test case, respectively, while the encoded pictures require 3256 and 3160 bits, respectively. The PSNR in the *SVDC* test case is thus higher, although the number of required bits is marginally lower. Comparing *Depth Dist.* and *SVDC* test cases shows that large parts of the sample pictures are very similar. PSNR improvements are thus rather locally concentrated. To demonstrate this, Figure 9.12 shows magnified cutouts of some typical artifacts and how SVDC-based encoding improves them. Figures F.55 to F.61 show further examples using other sequences.

Figure 9.12 (a) is blurred in the *Depth Dist.* test case. The reason is that depth distortion-based encoding distorts the left and right IV depth such that the left SV texture and the right SV texture do no longer match spatially. Consequently, blending them results in blurring. Whereas depth distortion-based encoding is not responsive to such artifacts when they are associated with a small depth distortion, SVDC-based encoding is responsive because of the large luminance difference of the tested SV texture and the reference SV texture.

Figures 9.12 (b) and (c) show double images, which are also caused by a mismatch of the left and the right SV texture, however, with a larger depth distortion than for the blurring artifacts.

Figure 9.12 (d) shows a typical boundary artifact. Samples of the foreground object remain in the background. SVDC-based encoding reduces such artifacts in particular between objects with height contrast.

Figures 9.12 (e) demonstrates an effect that can occur when the uncoded IV depth data is already distorted. In the *Uncoded* test case a kind of staircase occurs with a brightness difference to its left and right. Whereas SVDC-based encoding preserves this artifact, it is reduced by depth distortion-based encoding. However, as such improvements occur rather randomly, a more appropriate way would be to improve the IV depth maps before encoding instead of relying on an improvement in encoding.

In summary, the sample pictures show that at about the same depth bit rate SVDC-based encoding reduces boundary artifacts and blurring significantly. Sample pictures have been coded at a low depth bit rate point at which the PSNR difference between the SVDC and depth distortion test case is high. At higher depth bit rates, PSNR differences decrease as shown in Figure 7.4. Nevertheless, sample pictures give an indication how to interpret these PSNR gains.

Furthermore, sample pictures represent the perceived quality of the 3D video only in a limited way because they do not reflect motion and 3D related effects like flickering and an incorrect depth impression due to incorrect disparities. However, as the source for such effects are boundary artifacts and double images, they are reduced as well.

9.6 Evaluation using JCT-3V's Test Conditions

The test conditions used so far have been specifically selected in Section 7.2 to optimize and analyze depth coding. They did not regard texture coding and considered only depth bit rates and depth encoding times. Furthermore, they used only two IVs and one SV texture. Results

	This thesis: Depth; 2 IV; 1 SV				JCT-3V: Texture+Depth; 3 IV; 6 SV			
	$R_\Delta[\%]$		$T_\Delta[\%]$		$R_\Delta[\%]$		$T_\Delta[\%]$	
Ref.	DD	LS	DD	LS	DD	LS	DD	LS
Balloons	−78.2	−32.9	+ 78	+39	−15.6	− 8.9	+32	+21
Kendo	−83.8	−47.2	+ 57	+33	−23.4	−11.8	+33	+29
Newsp.	−81.1	−45.1	+ 83	+41	−24.5	−14.5	+43	+29
P.Hall2	−65.8	−32.8	+ 98	+54	−20.1	−10.0	+46	+32
P.Street	−73.9	−37.6	+101	+48	−11.5	− 8.6	+41	+28
GTFly	−85.0	−68.3	+ 88	+41	−16.7	−10.7	+39	+27
Shark	−86.2	−54.5	+ 71	+42	−20.1	−11.1	+42	+29
UndoD.	−68.6	−52.4	+ 80	+44	−22.0	− 6.8	+33	+24
Mean	−77.8	−46.4	+ 82	+43	−19.2	−10.3	+38	+27

Table 9.12: Bit rate deltas R_Δ and encoding time deltas T_Δ; Tested: SVDC-based encoding under this thesis' and JCT-3V's test conditions; Reference: Encoding using the depth distortion and LS method.

	$R_\Delta[\%]$							
	This thesis: Depth; 2 IV; 1 SV				JCT-3V: Texture+Depth; 3 IV; 6 SV			
	VSRS (1D)		VSRS (Gen.)		VSRS (1D)		VSRS (Gen.)	
Ref.	DD	LS	DD	LS	DD	LS	DD	LS
Balloons	−63.8	−12.1	−66.9	−10.9	−12.7	−4.4	−14.4	−6.0
Kendo	−74.7	−24.9	−71.9	−12.6	−20.5	−6.7	−21.7	−8.0
Newsp.	−66.1	−14.8	−71.6	−25.0	−19.8	−6.6	−20.4	−8.3
P.Hall2	−58.0	−18.4	−62.1	−23.5	−18.2	−4.9	−20.7	−8.9
P.Street	−59.0	−22.4	−66.3	−21.5	− 9.3	−6.2	−10.6	−6.7
GTFly	−54.2	−21.4	−78.1	−51.4	− 9.0	−4.8	−14.4	−5.9
Shark	−67.9	−24.0	−81.7	−32.8	−11.5	−5.9	−18.2	−7.2
UndoD.	−36.1	−15.1	−54.9	−22.8	−15.3	−0.1	−22.8	−1.9
Mean	−60.0	−19.1	−69.2	−25.0	−14.5	−5.0	−17.9	−6.6

Table 9.13: Bit rate deltas R_Δ and encoding time deltas T_Δ; Tested: SVDC-based encoding under this thesis' and JCT-3V's test conditions; Reference: Encoding using the depth distortion and LS method; VSRS.

of the *optimized encoder setup* under these conditions are summarized on the left in Table 9.12 using depth distortion-based encoding and the LS method (which performed best under the evaluated SVD estimation methods) as reference. With respect to these methods, SVDC-based encoding achieves depth bit rate reductions of 77.8% and 46.4% while the depth encoding time increases by about 82% and 43%.

To understand how these results are related to the actual scenario using texture and depth coding with multiple SVs, this section presents an evaluation under test conditions of the JCT-3V [Mull 14]. These conditions consider three IVs and six SVs. To encode the three IVs, the encoder uses varying QPs for the texture, as well as for the depth, and texture encoding tools

that depend on the depth map (i.e. VSP, DR, and DBBP; see Table D.1). SVDC computation considers the six SVs. The receiver-side uses the LCR with boundary noise removal enabled.

Figure 9.13 shows the RD performance under these conditions; Table 9.12 summarizes the deltas of the total bit rate (i.e. the depth plus the texture bit rate) and the total encoding time (i.e. the depth plus the texture encoding time). Results show that in comparison to depth distortion-based encoding SVDC-based encoding achieves total bit rate saving of about 19.2%. The total encoding time increases by 38%. In comparison to the LS method, depth bit rate savings are 10.3% with an encoding time increase of 27%.

Results for the VSRS in 1D and general mode are shown in Table 9.13: Under this thesis' test conditions, depth bit rate savings are 60% and 69.2%, respectively, in comparison to depth distortion-based encoding (see also Section 9.2); and 19.1% and 25%, respectively, in comparison to the LS method. Under JCT-3V's test conditions, SVDC-based encoding reduces the total bit rate by 14.5% and 17.9%, respectively, in comparison to depth distortion-based encoding; and by 5% and 6.6%, respectively, in comparison to the LS method. SVDC-based encoding thus outperforms depth distortion-based encoding and the LS method also in scenarios in that the RM's rendering algorithm does not match with receiver-side renderer.

In conclusion, SVDC computation achieves in comparison to SVD estimation also significant reductions of the total bit rate (5% to 10.3% depending on the view synthesis method). The highest total bit rate savings (10.3%) are achieved in scenarios in that the receiver-side renderer matches with the RM's algorithm. It should be noted that beyond increased coding gains on average, a scenario with matching algorithms has furthermore the advantage that the maximal introduced SVD can be controlled. In contrast, the depth distortion or an estimated SVD might not reflect that a local distortion of the depth map introduces a significant artifact in the SV. In particular, the LS method (with small blocks sizes) and the VSD method might be prone to such issues by their principle: Both estimate the SVD as the product of a factor derived from the local texture characteristics and the local depth distortion. When the factor is zero, e.g. in a smooth texture region, and a depth distortion shifts the local samples out of the region for that the factor is determined (e.g. across a neighboring texture edge), both methods would not detect any SVD, although it might be significant.

9.7 Chapter Summary

This chapter evaluated SVDC-based encoding using alternative test conditions. It considered 1) multiple SV positions, 2) alternative view synthesis methods, 3) estimation methods, 4) alternative distortion metrics, 5) sample pictures, and 6) texture coding.

When SVDC computation considers only one SV position, but the evaluation setup the average SV PSNR of the viewing range, depth bit rate savings of about 77% and 70% (compared to depth distortion-based encoding) are achieved in the view extrapolation and the view interpolation scenario, respectively, with an encoding time increase of about 56% and 82%, respectively. However, for some sequences using only one SV position for SVDC computation can lead to minor losses at SV positions farthest to it. Such losses can be avoided by increasing the number

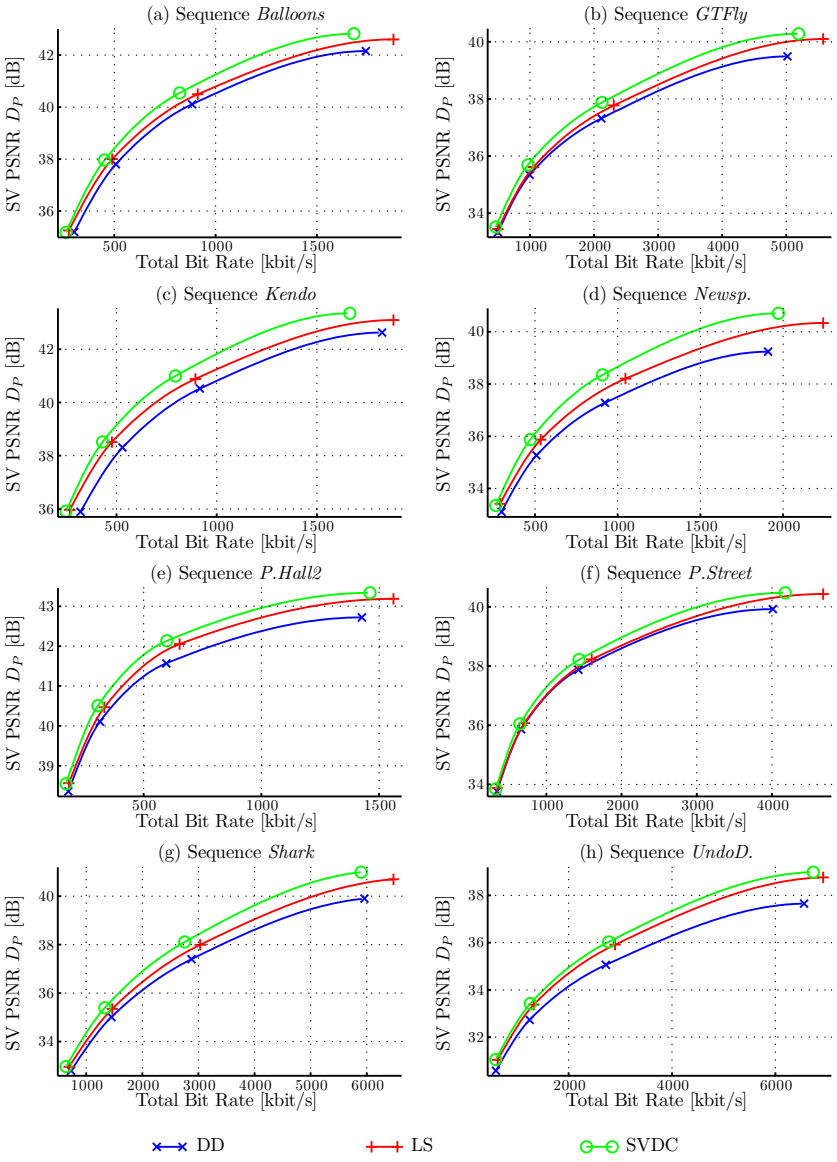


Figure 9.13: RD performance of different distortion derivation methods. JCT-3V's test conditions.

of SV positions to e.g. two in the view extrapolation scenario or three in the view interpolation scenario, which further increases the depth bit rate savings to 80.3% and 74.5%, but also encoding time deltas to 90% and 189%. Using more than two or three SV positions increases depth bit rate savings only marginally further, but increases computational complexity linearly. In a trade-off between computational complexity and RD performance, using more than two or three SV positions is thus not reasonable in most application scenarios.

With respect to the LCR with boundary noise removal, the VSRS in 1D mode, and the VSRS in general mode, SVDC-based encoding with the *optimized encoder setup* achieves depth bit rate savings of about 76.8%, 60%, and 69.2%, respectively, in comparison to depth distortion-based encoding. It is thus also efficient when the receiver-side view synthesis algorithm does not match the LCR, which is used for SVDC computation. Furthermore, coding losses due to deviations can be mitigated by decreasing the SVDC-depth distortion ratio ρ, as intended in [Jung 12c], and by disabling background suppression in view combination. A reasonable setup considering both achieves in comparison to the *optimized encoder setup* a depth bit rate decrease of 2.6% and 8.3% for the VSRS in 1D mode and general mode, respectively, while the depth bit rate increases by only about 1% for the LCR and the LCR with boundary noise removal.

For comparison with SVDC-based encoding, this chapter optimized three SVD estimation methods. Compared to the LS method, which performed best among them, SVDC-based encoding reduces the depth bit rate by 46.4%. Exact SVDC computation achieves thus significant coding gains also in comparison to SVD estimation, however, with an increased computational complexity: With respect to the LS method, the depth encoding time increases by about 43%.

SVDC-based encoding leads not only to SV texture PSNR gains, but also to MSSIM and FOM gains. Depth bit rate savings with respect to these metrics are 60% and 51.8%, respectively. SVDC-based encoding thus preserves structure and edges of the SV texture better than depth distortion-based encoding, which implies that also the perceived SV texture quality increases.

Sample pictures show that, at about the same depth bit rate, SVDC-based encoding significantly reduces boundary artifacts and since the left and the right SV match better, the occurrence of double images and blurring. However, it also preserves artifacts that are already present in the reference SV textures. As the reduced distortions are often the source of motion or 3D related artifacts, it can be assumed that those are reduced as well.

Under JCT-3V's test conditions, which also consider texture coding and a scenario with three IVs and six SVs, SVDC-based encoding reduces the total bit rate (i.e. the texture and depth bit rate) by 19.2% and 10.3% compared to using the depth distortion and the LS method, respectively. When the receiver-side uses the VSRS, total bit rate savings are 16% and 6%, respectively. Total encoding times increase by 38% and 27%, respectively. Results show that SVDC-based encoding achieves also significant bit rate reductions when considering the overall 3D system targeted in this thesis and for view synthesis algorithms not matching with the RM. These bit rate reductions justify the complexity introduced by SVDC-based encoding.

In conclusion, this chapter showed that SVDC-based encoding is effective in various scenarios deviating from the default test conditions used in this thesis.

Summary and Conclusions

This thesis presented and discussed an algorithm for SVD-aware encoding of depth maps. To this end, its two parts contributed and evaluated methods and approaches that define a rendering-based SVD measure, implement it with low complexity, and apply it optimally in depth encoding. This chapter summarizes these contributions and results.

The Synthesized View Distortion Change
This thesis answered the question how to define a rendering-based SVD measure by presenting the SVDC, which is a block-based and additive measure. In contrast to estimation methods, the SVDC is an exact measure with respect to the used rendering algorithm. In the scope of this thesis, the SVDC is applied in encoding only. However, beyond that other possible application scenarios are e.g. depth estimation, filtering, or enhancement.

Problems of a local SVD derivation: To motivate the SVDC, this thesis analyzed a local SVD derivation, i.e. the obvious approach of considering the SVD that occurs in the SV region that is distorted by a depth candidate in an evaluated IV region. This showed the following problems:

- The undistorted IV depth of the evaluated IV region must be known to identify the distorted SV region.
- The distorted SV region depends also on depth data outside the evaluated IV region. Consequently, the SVD in the distorted SV region is related to multiple IV regions.

Definition of the SVDC: To resolve these problems, this thesis presented the SVDC. The SVDC is the change of the overall SVD of the entire SV texture that is caused when the initial depth in the evaluated IV region is changed to the depth candidate. For computation, distorted depth is assumed outside the evaluated IV region if already known, and initial depth otherwise. This way, SVDC computation

- does not require an undistorted, but only an initial IV depth;
- establishes the relationship between the overall SVD and distorted IV depth regions not spatially, i.e. by identifying the SV region that contains the SVD related to a distorted IV depth region, but iteratively, i.e. by identifying the overall SVDC related to an IV depth change while considering the current processing state of the IV depth map;
- and depends on the processing order of the IV depth.

SVDC Computation: The SVDC differs from other distortion measures in that it depends on the processing state of the depth map. A question is thus how an SVDC computation process can interact with an encoder. For this, this thesis defined a framework for SVDC computation—the Renderer Model.

- The RM holds the state of the depth map and intermediate results that allow re-rendering.
- It provides an interface to encoders for SVDC computation.

- An encoder can invoke the RM in *SET* mode to update its state when a particular part of the depth map is finalized; or invoke it in *GET* mode to calculate the SVDC.

Low-Complexity Rendering Algorithm
Since computational resources are limited, SVDC computation in encoding requires a low-complexity rendering approach. By introducing the LCR, this thesis answered the questions how to implement such an approach and at which cost rendering can be performed. Key feature of the LCR is that it is extendable for encoder-side partial re-rendering. However, the rendering quality of the LCR suffices to use it as receiver-side renderer (e.g. the VSRS 1D-Fast software, an own contribution, is based on the LCR's methods). This way, a match between the encoder-side and receiver-side renderer can be achieved.

Algorithm: Requirements for the LCR are low complexity and extendability for partial re-rendering. This thesis showed how both can be achieved. For low complexity, its design

- avoids advanced methods, as e.g. used by the VSRS in general or 1D mode;
- combines, optimizes, and modifies several existing approaches to support interval-wise processing, quarter sample precise warping, occlusion detection, hole filling, correct rendering of foreground edges, and view interpolation;
- allows trade-offs between complexity and rendering quality since some of its features can selectively be disabled.

To allow extendability for partial re-rendering, the LCR integrates several rendering steps into a single process, which can be applied per IV depth sample so that all related SV samples are rendered. This facilitates to start and stop rendering at arbitrary positions. For each depth sample the process

- maps a related IV interval to an SV interval;
- identifies the SV interval's type by detecting occlusions, holes, and foreground object edges;
- adapts to the SV interval type and renders all samples related to the SV interval by quarter sample precise warping, hole filling, or special processing of foreground object edges;
- and combines the rendered SV samples with SV samples rendered from another view to obtain the final SV texture by view interpolation.

Complexity and rendering quality: This thesis answered the questions with which complexity rendering can be performed and which quality is achieved when the LCR is used at the receiver-side to render entire SV textures:

- Depending on enabled features, it performs between 15 and 108 operations per SV sample.
- From these, up to 44 operations are required for sample interpolation to achieve quarter sample precise warping. (In partial re-rendering, they need not be performed repeatedly.)
- Without using high complex rendering methods, the LCR provides a view synthesis quality similar to that of the VSRS and the 1D-Fast for high quality depth.
- In case of strongly distorted depth data, it can be outperformed by methods using boundary noise removal.

Partial Re-Rendering
Given the LCR, a question is how to employ it for SVDC computation. For this, this thesis extended the LCR by partial re-rendering, which enables efficient SVDC computation by re-rendering only the changed SV region, i.e. the region affected by an IV depth change. This way, complexity is reduced to a magnitude that becomes acceptable in encoding.

Analysis of the changed SV region: To find out which part of the IV depth map must be processed to ensure that the entire changed SV region is re-rendered, this thesis provided an analysis of the relationship of SV and IV regions:

- Because of the occlusion detection method, processing can be started at arbitrary positions.
- The position at that processing can be stopped depends on the old and new data within the changed IV depth block.
- The stop position must ensure that all parts of the SV that are no longer occluded after a depth change are updated.
- Furthermore, it must ensure that the occlusion signal, which is used by the RM to allow random access, is updated for IV samples that become occluded.

Extension of the LCR to the RM: This thesis answered the question on how to modify the LCR to enable partial re-rendering. Modifications allow the RM to iterate only once over the IV range to either change its state in *SET* mode or to calculate the SVDC in *GET* mode. More specifically, this thesis extended the LCR to the RM by

- storage of intermediate results as the RM's state so that they can be used for re-rendering;
- a recovery process to start re-rendering at a random access position;
- the derivation of the minimal changed SV position while re-rendering, and stopping there, i.e. when the entire changed SV region has been re-rendered;
- and the computation of the SVD and the SVDC while re-rendering.

Complexity: This thesis showed that the RM reduces the computational complexity of rendering to a magnitude that becomes acceptable in the scope of encoding.

- An analysis of different setups showed that, compared to a conventional SSD computation, the number of required operations and memory accesses is increased by factors in the range of 3.4 (for integer extrapolation without chroma) to 6.9 (for fractional interpolation with chroma).
- The complexity furthermore depends on the characteristics of the depth change; it increases with small block sizes and large disparity changes.
- Compared to an unoptimized re-rendering variant, the optimizations of the RM (i.e. interval-wise processing, *GET* mode, and texture mapping) reduce its complexity to about 20%.
- On top of this, the most significant reduction is achieved by partial re-rendering since it avoids re-rendering of the entire SV row.

Encoder Integration and Optimization
This thesis explored how to exploit the RM optimally in depth encoding. For this, the RM was integrated to a 3D-HEVC compliant encoder and an optimization with respect to various parameters has been performed.

Initial evaluation: The initial evaluation of the encoder showed that SVDC- instead of depth distortion-based encoding leads to a depth bit rate reduction of about the magnitude of what is achieved by 3D-HEVC over MV-HEVC.

- At same SV quality, SVDC-based encoding reduces the depth bit rate by 75% and 50% in scenarios using 3D-HEVC and MV-HEVC, respectively.
- Depth bit rate savings come at the costs of an increased depth encoding time of about 88% and 13%, respectively, and a loss of the quality of the depth map (about 15 dB and 10 dB).
- The set of selected modes differs in depth distortion- and SVDC-based encoding. An SVDC-based encoder allocates bits for signalling residuals and prediction parameters more selectively such that more bits are spend for a smaller fraction of all depth samples, whereas the percentage of samples coded in skip mode increases significantly.

Renderer model setups and the hiding effect: For rendering and SVDC computation, the RM uses different techniques, like e.g. quarter sample precise warping and view combination. A question is thus whether the complexity introduced by these techniques is justified considering the RD performance increase. This thesis showed that this is the case and revealed the hiding effect and ways to mitigate it.

- Using the SVD instead of the SVDC results in a depth bit rate increase of about 6% and a depth encoding time decrease of about 2%.
- Neglecting the SV's chroma components in SVDC computation reduces the depth encoding time by 4% and decreases the RD performance for the most of the evaluated sequences only insignificantly. However, to avoid artifacts in chroma components, they should be considered.
- Rendering with integer instead of quarter sample precision increases the depth bit rate by about 60%. Quarter sample precise warping is thus essential in SVDC-based encoding.
- SVDC computation without view combination decreases the depth encoding time by 9% and increases the average depth bit rate for natural sequences by about 20%. On the other hand, the depth bit rate of synthetic sequences decreases by up to 37%. Reason for this is a hiding effect caused by the view combination step, which occurs when encoding the (in coding order) first IV's depth while assuming an uncoded texture for the second IV.
- An evaluation of different view combination variants showed that the hiding effect can be mitigated by disabling background suppression in the view combination step when encoding the first IV. With this modifications, the depth bit rate decreases by 9.6% on average over all sequences and about 19% when averaged over the synthetic sequences.

SV generation setups: The RM can render the SV textures using view interpolation, view extrapolation, or a combination of both and using coded or uncoded texture and depth. A question is thus which SV generation setup is optimal. This thesis classified and analyzed different setups

and revealed the setup that achieves the highest coding gains. Furthermore, it showed which losses occur in scenarios with independent encoding of IVs or of texture and depth.

- When encoding the (in coding order) first IV, interpolating the tested and the reference SV textures from the same first IV texture, i.e. both either coded or uncoded, reduces the hiding effect and can thus decrease the depth bit rate. Highest depth bit rate reductions (10.4% compared to the initial encoder setup) can be achieved by rendering with the coded first IV texture while disabling background suppression. Nevertheless, interpolation using the uncoded first IV texture can be an option for scenarios requiring an independent encoding of texture and depth of the first IV.
- When encoding the second IV depth, interpolating the tested SV texture from already coded IV textures and the reference SV texture from uncoded IV textures provides the highest depth bit rate reductions and outperforms setups that use either only the coded or only the uncoded textures. Coding losses due to only using uncoded textures are moderate ($< 2\%$). SVDC computation is thus also suitable for scenarios with parallel encoding of texture and depth.
- When encoding the second IV depth, interpolating the tested SV texture from the already coded first IV depth and the reference SV texture from the uncoded first IV depth provides the highest depth bit rate reductions and outperforms setups that use either only the coded or only the uncoded first IV depth. Losses when using uncoded depth, which would occur for a scenario with independent encoding of the second IV depth, are significant ($> 10\%$).
- Using view extrapolation for encoding of the depth of both IVs or the first IV depth reduces the hiding effect and leads to gains for the synthetic sequences and losses for the natural sequences. When the hiding effect is already reduced by disabled background suppression, the extrapolation setups lead to depth bit rate increases of about 20% and 15%, respectively, compared to the initial encoder setup.

Depth distortion term: [Jung 12c] proposed to use a depth distortion term in addition to the SVDC, which is beneficial in case that the receiver-side renderer does not match with the RM. A question is thus whether it is also beneficial in a scenario with matching renderers. This thesis shows that this is the case while considering additionally view combination variants and SV generation setups.

- The depth distortion term reduces the depth bit rate only significantly (8.1% on average) when SVDC computation is affected by the hiding effect.
- In setups addressing these issues already by a modified view combination process or by modified SV generation, it leads to only small depth bit rate reductions (less than 0.5%).
- Depending on the setup, optimal depth distortion-SVDC ratios are in between $2^{4.5}$ and $2^{5.5}$, in which the higher weight is optimal when SVDC computation is less affected.
- Although additional operations are required for depth distortion computation, the depth encoding times decrease about 4% to 8% when using the depth distortion weight. Reason for this is that the depth PSNR increases by about 6 dB to 9 dB, which causes a decrease of the rendering overhead due to the magnitude of the depth change.

Furthermore, the evaluation revealed the optimal combination of SV generation setup, view

combination variant, and depth distortion weight—the optimized encoder setup—which disables background suppression, uses only coded texture data when coding the first depth map, and uses a depth distortion-SVDC ratio of $2^{5.5}$. It achieves 10.7% depth bit rate reductions compared with the initial encoder setup.

Mode selection stages: The encoder determines coding modes in several different selection stages, which can either use the SVDC (plus the depth distortion term) or the depth distortion solely. A question is thus which measure should be used in which selection stage. This thesis showed that using the SVDC in all selection stages leads to a depth bit rate decrease of 6.6% and an encoding time increase of factor 3.5 compared to the optimized encoder setup. Furthermore, it provided experimental results that allow to choose measures such that other trade-offs between depth bit rate savings and depth encoding time can be achieved (all changes are given in percent of the respective value in the optimized encoder setup):

- For intra-picture predicted CBs, enabling the SVDC successively for RQT coding, switching RDOQ, mode pre-selection, and wedgelet parameter selection leads to depth bit rate saving increments of 0%, 0.1%, 0.8%, and 0.4%, respectively, and depth encoding time increments of 3%, 6%, 5%, and 20%, respectively.
- Results show that using the SVDC for mode pre-selection is beneficial, while using it in other stages leads to a rather worse complexity-RD performance trade-off.
- For inter-picture predicted CBs, enabling the SVDC successively for RQT coding, switching RDOQ, motion derivation mode and merge candidate selection for all CB partitionings, and motion data selection leads to depth bit rate saving increments of 1.1%, 0.2%, 0.5%, and 3.8%, respectively, and encoding time increments of -1%, 10%, 4%, and 306%, respectively.
- Considering RD performance and complexity, using the SVDC for the inter RQT is thus most efficient; the depth encoding time decrease is caused by early mode decisions. On the other hand, using it to determine motion data provides with 3.8% the highest additional depth bit rate savings, but with 306% also the highest increase of the depth encoding time.
- Depth encoding time deltas and depth bit rate savings achieved for intra- and inter-picture predicted CBs are approximately additive.

Lagrange multipliers and QP: SVDC-based encoding depends on three encoder parameters: The Lagrange multipliers used in the SVDC-based and the depth distortion-based selection stages, i.e. λ_S and λ_M, and the depth QP. A question is thus how to choose them. To answer it, this thesis provided a discussion and an experimental analysis of the optimal relationship between them. It showed that the default QP values and Lagrange multipliers already provide an RD performance that is close to the optimal one for choosing unique values for all sequences.

- The theoretical analysis showed that λ_M is given by the quotient of λ_S and the derivative of the SVD with respect to the depth distortion, i.e. $\mathrm{d}D_S/\mathrm{d}D_M$. Knowing λ_M, the respective QP can be determined based on the conventional relationship used in texture coding.
- An experimental CTB-based search (similar to [Sull 98]) indicated that optimal QP and λ_M values are higher than the default values, which implies that $\mathrm{d}D_S/\mathrm{d}D_M$ is less than 1. However, combinations provided by the CTB-based search do not lead to depth bit rate reductions. A possible explanation is that the values of $\mathrm{d}D_S/\mathrm{d}D_M$ vary locally: The CTB-based

search rather converges to the optimal combination for the most frequently occurring value of $\mathrm{d}D_S/\mathrm{d}D_M$ than to the combination that provides the best overall RD performance.

- A sequence-based full search provided QP and λ_M values that are lower than the default values, which indicates that $\mathrm{d}D_S/\mathrm{d}D_M$ is greater than 1. This value seems to be reasonable considering that the selection of the QP and λ_M primarily affects blocks for that the SVD depends strongly on the depth distortion.
- Significant improvements of the RD performance are only achieved by the sequence-based full search when using sequence depended optimization. Bit rate savings considering unique values for all sequences are small ($\leq 0.3\%$).

Evaluation Using Other Test Conditions
To allow a broader assessment of the achieved coding gains that goes beyond the test conditions used for encoder optimizations, this thesis presented further evaluations.

SV position setups: SVDC computation increases the encoding complexity. For this, it can only consider a small number of SVs. A question is thus whether this is sufficient to increase the quality of the whole viewing range. This thesis showed that considering two or three SV positions for SVDC computation is sufficient; considering more SVs is not reasonable in a trade-off between computational complexity and RD performance. The evaluation regarding different SV position setups for view interpolation (VI) and view extrapolation (VE) showed the following in comparison to depth distortion-based encoding.

- Considering one SV position in SVDC computation increases the average SV texture quality of the viewing range by on average 0.8 dB (VE) and 0.4 dB (VI). These SV PSNR gains correspond to depth bit rate savings of about 77% (VE) and 70% (VI). They are achieved by a depth encoding time increase of about 56% (VE) and 82% (VI).
- The SV PSNR increases most at SV positions that are considered in SVDC computation, but in general also at the other SV positions. However, for some sequences using only one SV position for SVDC computation can lead to minor losses at SV positions farthest to it.
- Such losses can be avoided by increasing the number of SV positions to e.g. two (VE) or three (VI), which further increases the depth bit rate savings to 80.3% (VE) and 74.5% (VI), but also depth encoding time deltas to 90% (VE) and 189% (VI).
- Using more than two or three SV positions increases depth bit rate savings only little further, but increases computational complexity linearly by about 37% (VE) and 54% (VI) per additionally considered SV position.

Alternative rendering methods: The receiver-side renderer might not match with the LCR used for SVDC computation. A question is thus which performance losses occur in such a scenario. This thesis showed that SVDC-based encoding is still effective.

- Considering the LCR with boundary noise removal, the VSRS in 1D mode, and the VSRS in general mode showed that SVDC-based encoding with the optimized encoder setup achieves depth bit rate savings of about 76.8%, 60%, and 69.2%, respectively, in comparison to depth distortion-based encoding.

- Coding losses due to deviations can be mitigated by decreasing the SVDC-depth distortion ratio ρ [Jung 12c] and by disabling background suppression in view combination.
- A reasonable setup considering both achieves, in comparison to the optimized encoder setup, a depth bit rate decrease of 2.6% and 8.3% for the VSRS in 1D mode and general mode, respectively, while the depth bit rate increases by only 1% for the LCR and the LCR with boundary noise removal.

Comparison to SVD estimation methods: An alternative to SVDC-based encoding is the usage of SVD estimation methods. A further question is thus how methods compare to each other. This thesis showed that SVDC computation outperforms the SVD estimation methods.

- For a comparison, this thesis optimized three SVD estimation methods. The optimization analyzed whether comparing uncoded to coded IV texture is beneficial, determined SVD estimate-depth distortion ratios, and, for the LS method, block sizes and disparity ranges.
- In comparison to encoding with the LS method, which performed best among SVD estimation methods, SVDC-based encoding reduces the depth bit rate by 46.4%. However, depth encoding times also increase by 43%.
- In contrast to SVD estimation methods, SVDC-based encoding has furthermore the advantage that it allows to control the maximal introduced distortion.

SV quality evaluation: SVDC-based encoding leads to significant coding gains with respect to the PSNR. A question is thus whether gains can also be achieved when considering alternative distortion metrics and sample pictures. This was confirmed by this thesis.

- An evaluation using the MSSIM and the FOM showed depth bit rate savings of about 60% and 51.8% compared to depth distortion-based encoding, respectively. SVDC-based encoding thus preserves structures and edges of the SV texture better.
- This was also shown by sample pictures: At about the same depth bit rate, SVDC-based encoding reduces boundary artifacts. Furthermore, it reduces double images and blurring significantly because it achieves a better match of the left and the right SV.
- As the reduced distortions are often the source of motion or 3D related artifacts, it can be assumed that those are reduced as well.
- When artifacts are already present in the reference SV textures, they are also preserved.

Final evaluation using JCT-3V's test conditions: 3D video comprises texture and depth. A question is thus which coding performance is achieved when also considering texture coding. This thesis showed that the complexity of SVDC-based encoding is still justified by its high coding gains, even in comparison to estimation methods and when using alternative view synthesis algorithms. Compared to encoding using the depth distortion (DD) and the LS method, SVDC-based encoding with the optimized encoder setup leads to the following changes:

- Under default test conditions used in this thesis, the depth bit rates decrease by 77.8% (DD) and 46.4% (LS). Depth encoding times increase by 82% (DD) and 43% (LS).
- Under test conditions of the JCT-3V, which also consider texture coding and a scenario with three IVs and six SVs, total (i.e. texture and depth) bit rates decrease by 19.2% (DD) and

10.3% (LS). Total encoding times increase by 38% (DD) and 27% (LS).

- When the receiver-side uses the VSRS, depth bit rate savings are 65% (DD) and 23% (LS). Total bit rate savings are 16% (DD) and 6% (LS).

Outlook and Further Works

Possible extensions of SVDC-based encoding concern the SVD metric, the camera setup, and the texture encoder. As SVD metric, the RM uses the SSD, which might not represent the visually perceived quality perfectly. An improvement would thus be more advanced distortion metrics. In principle and in contrast to estimation methods, the RM can be the basis for computation of any distortion metric since it provides the actual SV texture. Considering the camera setup, the LCR and the RM support solely parallel arrangements as required by autostereoscopic displays. Another possible extension of the RM would thus be partial re-rendering for other camera arrangements, e.g. toed-in or arc-setups. Considering texture, encoders perform RD optimization with respect to its distortion. However, in a 3D video system, texture coding also leads to SVD changes. A texture encoder could thus consider these changes additionally, i.e. the SVDC and the IV texture distortion jointly. This would require a different RM design since the disparities are constant while the IV texture changes. Nevertheless, in summary, all suggested extensions would increase encoding complexity further.

Conclusion

Methods, optimizations, and evaluations in this thesis answered the initial questions on how to 1) define a rendering-based SVD measure, 2) implement it with low complexity, and 3) use it optimally in encoding. How to define a rendering-based SVD measure has been answered with the SVDC. The SVDC is an additive and exact measure that resolves the issue that distorted depth regions can not be mapped bijectivly to respective SV regions. How to implement SVDC computation with low complexity has been demonstrated by the development of the LCR and the RM. The LCR renders with low complexity and is extendable for partial re-rendering. This way, it can be used as receiver-side render and by the RM to compute the exact SVDC. For other receiver-side renderers, it is still representable since it comprises only very basic features. To compute the SVDC, the RM employs partial re-rendering, which allows to identify and process SV regions that are affected by a depth change. Consequently, computational complexity for rendering is reduced to a magnitude that is acceptable for the application in encoding. How to use the SVDC optimally in encoding has been demonstrated by the integration, optimization, and evaluation of the RM in a 3D-HEVC compliant encoder. The initial SVDC-based encoder reduced the depth bit rate in comparison to depth distortion-based encoding by a magnitude of about that what is achieved by 3D-HEVC over MV-HEVC. Optimizations of the view combination variant, the SV generation setup, and the depth weight achieved further depth bit rate reductions. Moreover, evaluations of the RM's features, SV generation setups, mode selection stage setups, SV positions setups, and Lagrange multipliers showed how to select other encoding parameters, how to achieve trade-offs between encoding complexity and RD performance, and how to adapt SVDC-based encoding to different scenarios. Other evaluations verified the effectiveness of SVDC-based encoding considering independent coding of texture and depth, multiple SV positions, alternative distortion metrics, alternative receiver-side rendering methods, and the total bit rate of texture and depth.

In summary, the contributions of this thesis enable an exact measurement of the SVD and its efficient exploitation in encoding. They achieve significant coding gains that allow the transmission of 3D video at a reduced bit rate or a higher quality and thus improve 3D video systems with auto-stereoscopic displays. This may help to increase the acceptance of such systems, which allow a more realistic representation of reality.

Appendix

A Depth Image Based Rendering Basics and Techniques

A.1 Coordinate Mapping

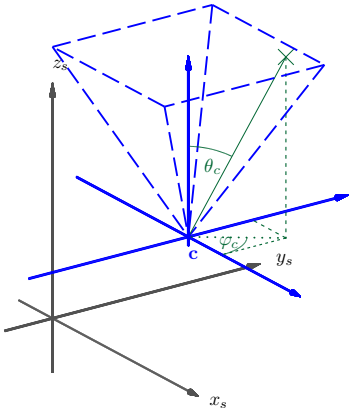


Figure A.1: Camera sampling the plenoptic function at the projection center $\mathbf{c}$.

To find points of the plenoptic function that are sampled by the camera, projection lines intersecting with its projection center $\mathbf{c} = (x_c \; y_c \; z_c)^T$ can also be parametrized in spherical coordinates $(r_c \; \theta_c \; \varphi_c)^T$ centered around $\mathbf{c}$. In this representation, r_c is the free parameter. Using (2.6) and defining θ_c and φ_c as shown in Figure A.1, they are given as follows with $\mathbf{e}_x$, $\mathbf{e}_y$, and $\mathbf{e}_z$ representing the unit vectors in the 3D space:

$$\theta_c = \arccos \frac{z_s - z_c}{\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2}} = \arccos \frac{\left((\mathbf{KR})^{-1}\mathbf{m}\right)\mathbf{e}_z}{\left\|(\mathbf{KR})^{-1}\mathbf{m}\right\|} \tag{A.1}$$

$$\varphi_c = \arctan \frac{y_s - y_c}{x_s - x_c} = \arctan \frac{\left((\mathbf{KR})^{-1}\mathbf{m}\right)\mathbf{e}_y}{\left((\mathbf{KR})^{-1}\mathbf{m}\right)\mathbf{e}_x} \tag{A.2}$$

A.2 View Synthesis Reference Software

As the View Synthesis Reference Software (VSRS) [Tian 09, MPEG 10b, Chen 15] has been the basis for the view synthesis method developed in this thesis and is used as reference for comparison, this section discusses its methods.

The VSRS comprises two main modes, which are commonly called *general mode* and *1D mode*. Whereas the 1D mode only allows synthesis of views in a parallel and linear camera setup, as examined in this thesis, the general mode allows synthesis at arbitrary positions. Overviews are

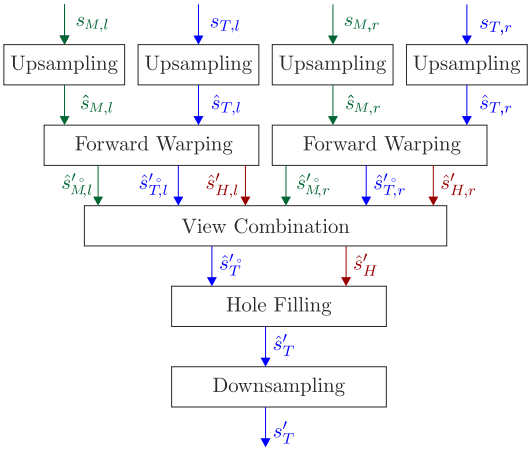


Figure A.2: Processing steps and signals of the VSRS in 1D mode.

shown in Figures A.2 and A.3 and discussed in the following. The description focuses on tools active in typical configurations. Some of them are optional and can be disabled.

A.2.1 1D Mode

Figure A.2 shows processing steps of the 1D mode. The 1D mode requires texture and depth of a left IV ($s_{T,l}$ and $s_{M,l}$, respectively) and a right IV ($s_{T,r}$ and $s_{M,r}$, respectively). Both IVs are upsampled and warped independently to create two different SV textures, $\hat{s}'^{\circ}_{T,l}$ and $\hat{s}'^{\circ}_{T,r}$, depicting the 3D scene from the same SV camera perspective. These texture are then combined to a single texture $\hat{s}'^{\circ}_{T}$, which is further processed in a hole filling and a downsampling step to create the SV texture s'_T—the output of the 1D mode. How the processing steps of the 1D mode address the problems inherent to DIBR is described in the following, in which upsampling and warping steps are only discussed for the left IV as identical steps are performed for the right IV.

The first step of the 1D mode is the *upsampling* step, which doubles the sampling rate of the IV texture $s_{T,l}$ and depth $s_{M,l}$ in horizontal direction. The following *forward warping* step shifts the samples of the upsampled IV texture $\hat{s}_{T,l}$ by disparities derived from the upsampled depth $\hat{s}_{M,l}$ to integer positions in the SV texture $\hat{s}'^{\circ}_{T,l}$, which has also a doubled horizontal sampling-rate compared to the IV texture and depth. This way, the warping step compensates the parallax of the IV with half sample distance precision relative to sampling distances in the input signals. Beyond that, the warping step addresses DIBR problems related to re-sampling and occlusions.

As discussed in Section 2.1.2.1, a problem in re-sampling is that small gaps occur when the distance of warped IV samples in the SV is greater than the SV sampling distance. To address this, the 1D mode performs point splatting. More specifically, the warping step is invoked twice, in which warped IV sample positions are rounded down in the first invocation and rounded up

in the second invocation. This way, an IV sample warped to a fractional SV position is assigned to the two adjacent integer SV positions in $\hat{s}'^{\circ}_{T,l}$ so that small gaps in the SV are filled. In the 1D mode this splatting can be enabled for regions close to depth boundaries only.

The warping step furthermore addresses the occlusion problem by two means: First, a sample is not warped when its SV position in $\hat{s}'^{\circ}_{T,l}$ is already occupied by a sample closer to the camera. And second, it is not warped when neighboring SV sample positions in $\hat{s}'^{\circ}_{T,l}$ are already occupied with samples having a significant different depth value. To enable both two further signals are generated in the forward warping step, which are a binary map $\hat{s}'_{H,l}$ indicating holes in $\hat{s}'^{\circ}_{T,l}$ and the depth map $\hat{s}'^{\circ}_{M,l}$ representing the depth of samples in $\hat{s}'^{\circ}_{T,l}$. In short, the 1D mode applies z-buffering to resolve the occlusion problem.

The SV texture $\hat{s}'^{\circ}_{T,l}$ rendered in the forward warping step comprises conventionally holes, which are caused by disocclusions. A majority of these holes are filled by the 1D mode in the *view combination* step. More specifically, this step combines the SV texture $\hat{s}'^{\circ}_{T,l}$, which is rendered from the left view, with an SV texture $\hat{s}'^{\circ}_{T,r}$, which is rendered from a right view, to derive the SV texture $\hat{s}'^{\circ}_{T}$. How the view combination step derives the value of a sample $\hat{s}'^{\circ}_{T}(x,y)$ depends on the binary hole maps $\hat{s}'_{H,l}$ and $\hat{s}'_{H,r}$, and the SV depth maps $\hat{s}'^{\circ}_{M,l}$ and $\hat{s}'^{\circ}_{M,r}$ as follows:

1. When $\hat{s}'_{H,l}(x,y)$ and $\hat{s}'_{H,r}(x,y)$ indicate that both samples $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$ are at hole positions, none of them is taken, and a hole remains at $\hat{s}'^{\circ}_{T}(x,y)$, which is marked in the combined hole map at $\hat{s}'_{H}(x,y)$.
2. When $\hat{s}'_{H,l}(x,y)$ and $\hat{s}'_{H,r}(x,y)$ indicate that only one of $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$ is not at a hole position, the value of the respective sample is taken.
3. When $\hat{s}'_{H,l}(x,y)$ and $\hat{s}'_{H,r}(x,y)$ indicate that both samples $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$ are not at hole positions, $\hat{s}'^{\circ}_{T}(x,y)$ is derived in a way that increases its reliability as follows:
   (a) When $\hat{s}'^{\circ}_{M,l}(x,y)$ and $\hat{s}'^{\circ}_{M,r}(x,y)$ indicate that the depth of $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$ differ significantly, the value of the sample further in the foreground is taken to prevent that erroneous background samples occur in foreground objects.
   (b) When $\hat{s}'_{H,l}$ and $\hat{s}'_{H,r}$ indicate that there is a significant difference in the number of holes in a 7×7 window around $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$, the sample value surrounded by less holes is taken, as hole regions conventionally occur at depth edges, which are more likely distorted.
   (c) In other cases, the values of $\hat{s}'^{\circ}_{T,l}(x,y)$ and $\hat{s}'^{\circ}_{T,r}(x,y)$ are averaged with distance dependent weights. The weight for $\hat{s}'^{\circ}_{T,l}(x,y)$ is higher than the weight for $\hat{s}'^{\circ}_{T,r}(x,y)$ when the SV camera is closer to the left IV camera than to the right IV camera. This reduces uncorrelated distortions in $\hat{s}'^{\circ}_{M,l}(x,y)$ and $\hat{s}'^{\circ}_{M,r}(x,y)$ while regarding that the SV distortion increases with increasing distance between the IV and SV camera positions.

This way, the view combination step not only addresses the disocclusion problem, but also problems caused by distorted depth. Consequently, the reliability of the combined texture $\hat{s}'^{\circ}_{T}(x,y)$ is increased. Furthermore, the view combination step applies the same combination rules to render a combined depth map $\hat{s}'^{\circ}_{M}$ from $\hat{s}'^{\circ}_{M,l}$ and $\hat{s}'^{\circ}_{M,r}$, which is required in the following hole filling process.

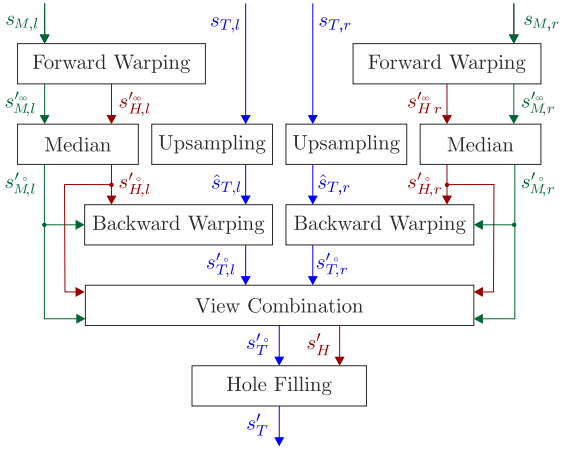


Figure A.3: Processing steps and signals of the VSRS in general mode.

View combination conventionally fills the majority of holes. However, depending on the depth structure of the 3D scene, there can be samples that are neither visible in $\hat{s}'^{\circ}_{T,l}$ nor in $\hat{s}'^{\circ}_{T,r}$. Consequently, holes can remain in $\hat{s}'^{\circ}_{T}$. For this, the *hole filling* step processes $\hat{s}'^{\circ}_{T}$ to render a texture $\hat{s}'_{T}$ without holes. More specifically, it detects the background side of a hole using $\hat{s}'^{\circ}_{M}$ and $\hat{s}'_{H}$. Then, it chooses the value of the first sample of $\hat{s}'^{\circ}_{T}$ on the background side as value for all samples in $\hat{s}'_{T}$ corresponding to the hole. This way, holes are filled by their adjacent background samples values. As holes usually occur in the background, this is a better estimate than the value of the adjacent foreground samples.

The final *downsampling* step reduces the sampling rate, which was initially increased for half-sample accurate warping, to the sampling rate of the IV signals. It thus decimates $\hat{s}'_{T}$ by a factor of two to generate the final SV texture s'_{T}.

A.2.2 General Mode

An overview of processing steps of the general mode is shown in Figure A.3. Similar to the 1D mode the general mode performs view interpolation. For this, the general mode renders two SV textures $s'^{\circ}_{T,l}$ and $s'^{\circ}_{T,r}$ from the left and right IV data, respectively, merges them to an SV texture s'°_{T} in a view combination step, and derives the final SV texture s'_{T} in a hole filling step. However, in contrast to the 1D mode, the general mode does not render the SV textures $s'^{\circ}_{T,l}$ and $s'^{\circ}_{T,r}$, which are used in the view combination step, by forward warping with upsampled IV signals, but by several processing steps performing depth forward and texture backward warping. How this is done specifically and how other processing steps differ from the 1D mode is discussed in the following.

As Figure A.3 shows, the general mode first processes the IV texture $s_{T,l}$ and IV depth $s_{M,l}$

independently to render input signals for the backward warping step. More specifically, the *forward warping* step renders the SV depth map $s'^{\circ}_{M,l}$ by shifting the IV depth samples $s_{M,l}$ with disparities derived from $s_{M,l}$ itself. In this process, shifted IV sample positions are rounded to the integer SV grid positions. Furthermore, the value of the foreground sample is chosen when several IV samples are shifted to the same SV position and positions that remain vacant are marked in the hole map $s'^{\circ}_{H,l}$.

The next step—the *median filtering* step—filters the SV depth $s'^{\circ}_{M,l}$ and the hole map $s'^{\circ}_{H,l}$ four times with a median like-filter. Output of this filter is either the median of a 3×3 window if the majority of samples within the 3×3 are non-hole samples, or the value of the center sample otherwise. This way, small holes due to disocclusions and increased distances of warped samples are removed from $s'^{\circ}_{M,l}$ and accordingly from $s'^{\circ}_{H,l}$.

Besides and independent from the derivation of $s'^{\circ}_{M,l}$ and $s'^{\circ}_{H,l}$, the *upsampling* step quadruples the sampling rate of the IV texture $s_{T,l}$. This results in the upsampled IV texture $\hat{s}_{T,l}$. The three signals $s'^{\circ}_{M,l}$, $s'^{\circ}_{H,l}$, and $\hat{s}_{T,l}$ are then used by the *backward warping* step to render the SV texture $s'^{\circ}_{T,l}$. For this, the backward warping step processes all positions in the SV texture $s'^{\circ}_{T,l}$ that are not marked as holes in $s'^{\circ}_{H,l}$. When processing a particular SV position, it derives a disparity value from the SV depth $s'^{\circ}_{M,l}$, uses the disparity to identify a corresponding IV position in $\hat{s}_{T,l}$ and assigns the sample value at the corresponding IV position in $\hat{s}_{T,l}$ to the particular SV position in $s'^{\circ}_{T,l}$. This way, the general mode renders $s'^{\circ}_{T,l}$ with quarter sample accurate parallax compensation without operating with SV signals with increased sampling rate.

Similar to the 1D mode, the general mode combines the SV texture $s'^{\circ}_{T,l}$ rendered from the left IV with an SV texture $s'^{\circ}_{T,r}$ rendered from the right IV in the *view combination step*. However, in contrast to the 1D mode, the view combination step does not discard samples that have a significant deviating background depth value or are located close to hole regions (i.e. items 3a and 3b, respectively, in Appendix A.2.1. It thus only applies camera distance dependent weighting when two samples are given for an SV position and is therefore more prone to depth distortions.

The final step of the general mode is the *hole filling* step. In contrast to the 1D mode, it fills holes that remain after blending regardless of whether neighboring samples belong to the back or the foreground by a complex inpainting method based on Navier-Stokes equations [Bert 01]. The output of the hole filling step is the final synthesized texture s'_T.

Besides the described processing steps, the VSRS comprises boundary noise removal methods to address misaligned depth and texture edges, as discussed in Section 2.1.2.2. These methods are based on removing SV texture samples or not warping IV texture samples in edge regions that are potentially associated with incorrect depth data. Resulting vacant SV positions are then treated as holes, and as such filled in the view combination or hole filling steps.

B Low-Complexity Depth Image Based Rendering Algorithm

B.1 Proof Occlusion Detection

The proof shows that an IV sample shifted from the IV position $x-n$, with $n > 0$, is occluded in the SV when $\mathrm{f}(x-n) \geq \mathrm{f}(x)$ is true, in which f denotes the shifting function (2.14). The proof assumes that a left IV is processed to render an SV on its right and that background samples on the left of a foreground object in the IV do not appear in a hole at the right side of the foreground object in the SV. [Berr 03] shows the same differently.

Proof:

$$
\begin{aligned}
& \mathrm{f}(x-n) \geq \mathrm{f}(x) \\
\Leftrightarrow \quad & x - n - s_P(x-n) \geq x - s_P(x) && \text{with (2.14)} \\
\Rightarrow \quad & s_P(x-n) < s_P(x) \\
\Leftrightarrow \quad & s_Z(x-n) > s_Z(x) && \text{with (2.11)}
\end{aligned}
$$

This means that the sample at IV position x is located closer to the camera than the sample at IV position $x-n$. Therefore, the sample from IV position $x-n$ is shifted behind a foreground object and thus occluded in the SV. □

Consequently, when processing the IV from right to left, a rendered SV sample will never be occluded by any SV sample rendered later and can thus be regarded as final [Berr 03].

Example:

In Figure 2.2, the sample at the IV position 5 is shifted to the SV position $\mathrm{f}(5) = 5 - s_P(5) = 2.25$. All samples with IV position $x < 5$ are occluded when they are shifted to SV position with $\mathrm{f}(x) < 2.25$ as they can only move on the diagonal lines in the x'-s_P space.

B.2 Tables Quarter Sample Precision

Position	0	1	2	3	4	5	6	7	Div
1/4	− 1	4	−10	57	19	− 7	3	− 1	64
2/4	− 1	4	−11	40	40	−11	4	− 1	64
3/4	− 1	3	− 7	19	57	−10	4	− 1	64

Table B.1: Luma upsampling; HM software [McCa 11].

$x'_e - x'_s$ \ $\check{x}' - x'_s$	0	0.25	0.5	0.75	1	1.25	1.5	1.75	2
0	0	–	–	–	–	–	–	–	–
0.25	0	1	–	–	–	–	–	–	–
0.5	0	0.5	1	–	–	–	–	–	–
0.75	0	0.25	0.5	1	–	–	–	–	–
1	0	0.25	0.5	0.75	1	–	–	–	–
1.25	0	0.25	0.5	0.5	0.75	1	–	–	–
1.5	0	0.25	0.25	0.5	0.75	0.75	1	–	–
1.75	0	0.25	0.25	0.5	0.5	0.75	0.75	1	–
2	0	0.25	0.25	0.5	0.5	0.75	0.75	1	1

Table B.2: Look-up table for the fraction in (3.2).

C Complexity Analysis of Rendering and Distortion Derivation

C.1 Methodology

To obtain a platform independent complexity measure, the analysis counts the number of operations and memory accesses that are logically needed to implement the algorithms. For the LCR and the RM some numbers can depend on the input data. For this, they have been determined in a simulation using the JCT-3V test set. The analysis neglects steps that the distortion derivation methods perform only once per picture for initialization. It considers only operations and memory accesses that are repeatedly performed for distortion computation.

The analysis distinguishes the following operations:

- AD: Additions and subtraction
- UA: Unary additions and unary subtractions
- US: Unary bit shifts
- ML: Multiplications
- CP: Comparisons
- LT: Table look-ups
- MR: Reading memory accesses
- MW: Writing memory accesses

A comparison can be a comparison of two operands or a comparison of one operand with a fixed value. The analysis neglects memory accesses to variables in the current work flow, i.e. variables that are locally available (as e.g. results of previous operations).

For the LCR and the RM operations and memory accesses are categorized in three different modes in that they are required:

- R: General rendering with the LCR
- S: Re-rendering with the RM in *SET* mode
- G: Re-rendering with the RM in *GET* mode

Furthermore, tables indicate in which steps the operations and memory accesses are performed:

- W: Warping step (Section 3.2.1 and modifications in Section 5.1.4)
- S: Render mode selection step (Section 3.2.2)
- R: SV interval rendering step (Section 3.2.3)
- C: View combination step (Section 3.2.4)
- D: SVD and SVDC computation step (Section 5.1.5)

C.2 Low Complexity Renderer and Renderer Model

Mode R	Mode S	Mode G	Step	Action	Operations and memory accesses AD	UA	CP	US	ML	LT	MR	MW	N_T
×	×	×	W	Decrement positions x_s in $s_{M,l}$.		1							1
×	×	×	W	Read IV depth data $s_{M,l}$.							1		1
	×	×	W	Decrement the positions in $\ddot{s}_B$.		1							1
	×	×	W	Read new depth $\ddot{s}_B$.							1		1
×	×	×	W	Convert $s_{M,l}$ to $s_{P,l}$.						1			1
×	×	×	W	Compute x'_s with (2.14).	1								1
×	×	×	W	Stop condition P, C4 in Figures 3.2 and 5.1.			1						1
	×	×	W	Compute minimal changed position x'_C (5.6).			2						2
	×		W	Write new input depth data $s_{M,l}$.								1	1
	×		W	Write occlusion signal $s_{O,l}$.								1	1
×	×	×	S	Compute interval width $x'_e - x'_s$.	1								1
×	×	×	S	Most probable path O1, H1 in Figure 3.2.			2						2
×	×	×	R	Decrement positions $\check{x}'$ in SV $s'_{T,l}$.		1							1
×	×	×	R	Read IV texture $\hat{s}_{T,l}$ or $s_{T,l}$.							1		1
×	×	×	R	Check range and picture size limits.			4						4
×			R	Write SV texture $s'_{T,l}$								1	1
×			$\sum$		2	2	7			1	2	1	15
	×				2	3	9			1	3	2	20
		×			2	3	9			1	3		18

Table C.1: Operations and memory accesses; *Base setup*

Mode R	Mode S	Mode G	Step	Action	Operations and memory accesses AD	UA	CP	US	ML	LT	MR	MW	N_T
×	×	×	W	Shift x_s to quarter prec.				1					1
×	×	×	R	Round up x'_s to get $\check{x}'_s$ (3.1).		1		1					2
×	×	×	R	Shift $\check{x}'$ to quarter prec.				1					1
×	×	×	R	Compute $\check{x}' - x'_s$ in quarter prec. for (3.2).	1								1
×	×	×	R	Get fraction in quarter prec. for (3.2).						1			1
×	×	×	R	Compute $\hat{x}$ from frac. and x_s in quar. prec. (3.2).	1								1
×			R	Interpolate $\hat{s}_{T,l}$ (required for 3 of 4 samples).	5.25			0.75	4.5				10.5
×			Σ		7.25	1		3.75	4.5	1			17.5
	×	×			2	1		3		1			7

Table C.2: Additional operations and memory accesses; *quarter sample precision*.

Mode R	Mode S	Mode G	Step	Action	Operations and memory accesses AD	UA	CP	US	ML	LT	MR	MW	N_T
×			R	No need to write $s'_{T,l}$ in the *Base Setup*.								−1	−1
×			R	Write $s'_{M,r}$, $s'_{H,r}$ when processing right view.								2	2
×	×	×	C	Read signals from right IV $s'_{T,r}$, $s'_{M,r}$, $s'_{H,r}$.							3		3
×	×	×	C	Compute depth difference $s'_{M,l} - s'_{M,r}$.	1								1
×	×	×	C	Combination logic in Table 3.1.			3						3
×	×	×	C	Distance dependent weighting using LUT (3.3).	1.9					1			2.9
×			C	Write final texture s'_T.								1	1
×			Σ		2.9		3			1	3	2	11.9
	×	×			2.9		3			1	3		9.9

Table C.3: Additional operations and memory accesses; *view combination*.

Mode R	Mode S	Mode G	Step	Action	Operations and memory accesses AD	UA	CP	US	ML	LT	MR	MW	N_T
	×	×	D	Read reference SV texture $s'_{T,Ref}$.							1		1
	×	×	D	Derive new squared distortion s'_D.	1				1				2
	×		D	Write new squared distortion s'_D.								1	1
		×	D	Read old squared distortion s'_D.							1		1
		×	D	Compute current SVDC $D_S(x')$.	1								1
		×	D	Add current SVDC $D_S(x')$ to total SVDC.	1								1
×			Σ										
	×				1				1		1	1	4
		×			3				1		2		6

Table C.4: Additional operations and memory accesses; *distortion computation*.

Mode R	Mode S	Mode G	Step	Action	Operations and memory accesses AD	UA	CP	US	ML	LT	MR	MW	N_T
×	×	×	R	Skip each second position.			1	1					2
×	×	×	R	Read chroma related to $\hat{s}_{T,l}$.							0.5		0.5
×			R	Write chroma related to $s'_{T,l}$.								0.5	0.5
×			Σ				1	1			0.5	0.5	3
	×	×					1	1			0.5		2.5

Table C.5: Additional operations and memory accesses; chroma in *base setup*.

Mode					Operations and memory accesses								
R	S	G	Step	Action	AD	UA	CP	US	ML	LT	MR	MW	N_T
×	×	×	C	Round $\hat{x}$ to chroma precision				1					1
×			C	Interpolate chroma related to $\hat{s}_{T,l}$.	5.25			0.75	4.5				10.5
×			Σ		5.25			1.75	4.5				11.5
	×	×						1					1

Table C.6: Additional operations and memory accesses; chroma in *quarter sample precision*.

Mode					Operations and memory accesses								
R	S	G	Step	Action	AD	UA	CP	US	ML	LT	MR	MW	N_T
×			R	No need to write chroma related to $s'_{T,l}$.								–0.25	–0.25
×	×	×	C	Read chroma from right IV related to $s'_{T,r}$.							0.48		0.48
×	×	×	C	Distance dependent chroma weighting (3.3).	0.98					0.48			1.45
×			C	Write chroma related to texture s'_T.								0.25	0.25
×	×	×	Σ		0.98					0.48	0.48		1.93

Table C.7: Additional operations and memory accesses; chroma in *view combination*.

Mode					Operations and memory accesses								
R	S	G	Step	Action	AD	UA	CP	US	ML	LT	MR	MW	N_T
	×	×	D	Read chroma reference related to $s'_{T,Ref}$.							0.5		0.5
	×	×	D	Compute squared distortion of chroma textures.	0.5				0.5				1
	×	×	D	Add and scale squared chroma distortion to s'_D.	0.5			0.25					0.75
×			Σ										
	×	×			1			0.25	0.5		0.5		2.25

Table C.8: Additional operations and memory accesses; chroma in *distortion computation*.

C.3 Estimation Methods

Action	Operations and memory accesses								
	AD	UA	CP	US	ML	LT	MR	MW	N_T
Iterate over and read $\tilde{s}_{M,l}$ and $s_{P,l}$		2					2		4
Iterate over and read factor $\alpha \cdot \beta$ (for each 4-th sample)		0.25					0.25		0.5
Convert $\tilde{s}_{M,l}$ to $\tilde{s}_{P,l}$						1			1
Derive absolute difference $\lvert\tilde{s}_{P,l} - s_{P,l}\rvert$	1								1
Multiply with $\alpha \cdot \beta$					1				1
Sum differences	1								1
$\sum$	2	2.25			1	1	2.25		8.5

Table C.9: Operations and memory accesses; *least squares method*.

Action	Operations and memory accesses								
	AD	UA	CP	US	ML	LT	MR	MW	N_T
Iterate over and read $\tilde{s}_{M,l}$ and $s_{P,l}$		2					2		4
Iterate over and read factor $\alpha \cdot \beta$ values		1					1		2
Convert $\tilde{s}_{M,l}$ to $\tilde{s}_{P,l}$						1			1
Derive $\left[\alpha \cdot \beta \cdot (\tilde{s}_{P,l} - s_{P,l})\right]^2$	1				2				3
Sum differences	1								1
$\sum$	2	3			2	1	3		11

Table C.10: Operations and memory accesses; *VSD method*.

Action	Operations and memory accesses								
	AD	UA	CP	US	ML	LT	MR	MW	N_T
Iterate over and read $\tilde{s}_{M,l}$, $s_{P,l}$, and $\tilde{s}_{T,l}$		3					3		6
Convert $\tilde{s}_{M,l}$ to $\tilde{s}_{P,l}$						1			1
Derive $\tilde{x} = x - [\tilde{s}_P(x) - s_P(x)]$	2								2
Read $s_{T,l}(\tilde{x})$							1		1
Derive $[\tilde{s}_{T,l}(x) - s_{T,l}(\tilde{x})]^2$	1				1				2
Sum differences	1								1
$\sum$	4	3			1	1	4		13

Table C.11: Operations and memory accesses; *shifting method*.

C.4 Sum of Squared Differences

Action	Operations and memory accesses								
	AD	UA	CP	US	ML	LT	MR	MW	N_T
Iterate over $\tilde{s}_{M,l}$ and $s_{M,l}$		2							2
Read $\tilde{s}_{M,l}$ and $s_{M,l}$							2		2
Derive squared difference $(\tilde{s}_{M,l} - s_{M,l})^2$	1				1				2
Sum differences	1								1
$\sum$	2	2			1		2		7

Table C.12: Operations and memory accesses; *SSD computation*.

D High Efficiency Video Coding Techniques

D.1 Single Layer High Efficiency Video Coding Techniques

D.1.1 Partitioning in Coding Blocks

In HEVC a picture consists conventionally of one or more slices. All slices of a picture can be decoded independently from each other, which is of advantage in case of data losses. Whether and how a picture is divided in slices is thus conventionally decided based on the application scenario. A slice consists of multiple square coding tree blocks (CTB) of size 16×16, 32×32, or 64×64, which are represented by coding tree units (CTUs) in the bitstream. The CTB size is conventionally constant for all pictures of a sequence. Each CTU contains the root node of the coding tree—a quadtree structure, which defines the split of the CTB in square coding blocks (CBs) of a minimum size of 8×8 in terms of luma samples. Motivation for the partitioning in CBs is to select coding modes based on the locals signal characteristics. Consequently, the partitioning in CBs conventionally differs from CTB to CTB and is selected based on rate-distortion costs.

D.1.2 Prediction

In HEVC, prediction of the current CB's sample values can be categorized in intra-picture prediction, i.e. prediction from already reconstructed parts of the current picture, and inter-picture prediction, i.e. prediction from already reconstructed reference pictures. Depending on how their CBs are predicted, HEVC distinguishes three slice types: I slices only contain intra-picture predicted CBs. As I slices do not refer to other pictures, they are used to encode pictures at that a decoder can start decoding [called Intra Random Access Point (IRAP) pictures]. P slices can additionally contain unipredicted CBs, i.e. CBs inter-picture predicted from one reference picture. On top of this, B slices can contain bipredicted CBs, i.e. CBs inter-picture predicted from two reference pictures.

Slice types are conventionally selected based on requirements imposed by the video system, as e.g. delay, maximal temporal distance of random access points, or complexity. In contrast, an encoder conventionally selects the prediction type and parameters in order to optimize rate-distortion costs. To achieve this, it can further divide CBs in prediction blocks (PBs) and select different prediction parameters for each PB. However, all PBs of a CB have to use either intra-picture or inter-picture prediction.

Intra-picture Prediction
Intra-picture predicted CBs can be associate with one or, at minimum CB size, four PBs using different prediction modes. When a PB is predicted, its prediction mode is applied to all its TBs. This means that for each TB a prediction is derived from the TB's neighboring samples that are spatially adjacent to its left and top margin (and called reference samples). The rational behind

applying the prediction based on the TBs and not on the PBs is that this way the residual of preceding TBs can already be decoded and taken into account. For prediction, HEVC provides three basic techniques:

- *Angular modes:* The angular modes extrapolate the reference sample values to the current PB. The extrapolation is performed along a straight line having a particular prediction direction that is signaled as one out of 33 specified by HEVC.
- *DC mode:* The DC mode predicts the current PB with a constant value that is the average of the reference samples.
- *Planar mode:* The planar mode averages two linear predictions that are based on 1) the reference sample at the PB's top right corner and the reference samples on its left and 2) the reference sample at the PB's bottom left and reference samples at its top.

In summary, HEVC provides 35 different possibilities to adapt intra-picture prediction of a PB to the local signal characteristics.

Inter-picture Prediction
Inter-picture predicted CBs can be divided in one, two, or four PBs. In case of two PBs, the split can be horizontal or vertical and furthermore asymmetrical, i.e. not only at half, but also at quarter or three-quarter width or height. In case of four PBs, the CB is horizontally and vertically split at half width and height. This way, HEVC supports eight different CB partitioning schemes.

Each of the PBs can be predicted differently from one or two reference pictures. Which reference pictures can be used is determined by the HLS. Available reference pictures for the current picture are ordered in two reference picture lists. A decoder predicts a current PB from the reference pictures as follows:

1. It derives motion information, i.e. a reference picture list indication, a reference picture index, and a motion vector, based on syntax element values of the current PU.
2. It uses the list indication and the reference picture index to identify a reference picture.
3. It identifies the position of a reference block in the reference picture by adding the motion vector to the current PB's position.
4. It uses sample values of the reference block as prediction of the current PB, in which fractional positions might be interpolated.
5. When the PU uses a second reference picture, the decoder performs steps 2 to 4 for the second set of motion information and averages both predictions.

This way, HEVC can exploit the similarities of different pictures by motion compensated prediction. However, a question left open so far is how the decoder derives the motion information of the current PB in step 1. HEVC provides two motion derivation modes for this, called Advance Motion Vector Prediction (AMVP) and merge mode.

In *AMVP mode*, the current PU contains the reference picture index and the prediction list indication. To derive the motion vector, the AMVP decoding process first derives a candidate list, which contains up to three MVs from spatially and temporally adjacent PUs. It then selects

an MV predictor from this list based on an indication signaled in the current PU. When the MV predictor refers to a reference picture different from the reference picture signaled for the current PU, AMVP linearly scales the MV to adjust it to the different temporal distance. Finally, AMVP adds an MV difference, which is also signaled in the PU, to the predictor to derive the final MV. This way, AMVP provides the possibility to signal arbitrary MVs for a PU while reducing the signalling overhead by motion prediction.

The second mode for motion derivation is the *merge mode*. The merge process first derives a list of merge candidates, i.e. a list containing the motion information of up to five spatially or temporally adjacent PUs. It then selects the candidate from the merge candidate list that is indexed by a value signaled in the PU. This way, the merge mode not only predicts the MVs, but also their reference pictures from neighboring PUs. As the merge mode furthermore does not use MV differences, it reduces, compared to AMVP, the bit rate for signalling MVs at the cost of signalling flexibility.

When an encoder applies the merge mode, it is often not necessary to split the CB or to transmit the residual. To reduce the signalling cost for this frequent case, HEVC provides additionally a signalling alternative—the skip mode. In skip mode, the CU only contains the skip flag (equal to 1) and a merge index. Quantized transform coefficients are not present. The decoding process then predicts the entire CB using the merge mode.

D.1.3 Residual Coding

For residual coding, HEVC allows to quarter the CB in square blocks and to recursively quarter the sub-blocks further. This way, HEVC represents the partitioning of the CB's residual by a residual quadtree (RQT) whose leaf nodes are the transform blocks (TBs). TBs can have a size of 4×4, 8×8, 16×16, or 32×32 in terms of luma samples. To code a TB's residual, the encoder first applies an integer DCT-like transform. It then quantizes transform coefficients with an uniform reconstruction quantizer, encodes the quantized transform coefficients, and stores them in the respective TU in the bitstream.

HEVC represents the quantization step size logarithmically with a quantization parameter (QP). The QP range is from 0 to 51, inclusive, in which a QP change of 6 approximately doubles the quantization step size. As the decoder requires the used QP for scaling, the encoder signals it in the SPS, in the slice header or at CU level. However, to avoid signalling overhead at the CU level, encoders conventionally use the same QP for the entire sequence. Furthermore, HEVC supports to explicitly signal quantization and scaling matrices.

D.1.4 Loop Filters

After a decoder has decoded all CBs of a picture, it applies two loop filters, called deblocking filter and Sample Adaptive Offset (SAO) to the reconstructed picture before storing it in the reference picture buffer or output.

The deblocking filter operates on samples adjacent to TU and PU boundaries at an 8×8 grid. Whether filtering is applied and the strength of filtering depends on how adjacent blocks are coded, i.e. on the presence of non-zero transform coefficients or on whether the same motion information have been used. Furthermore, it depends on signaled HLS syntax elements.

Sample Adaptive Offset (SAO) is a nonlinear filter that adds a signaled offset to the reconstructed samples based on the local signal characteristics. More specifically, an encoder can signal one of two modes, called band offset mode and edge offset mode, and related parameters on a CTU basis. In the band offset mode the sample value range is divided in 32 sub-ranges called bands. For four of these bands, which are consecutive in order, the encoder signals offsets. The SAO process then adds these offsets to samples with values in the respective bands. In the edge offset mode, the encoder also signals four different offsets and additionally a gradient direction. The SAO process then classifies the samples based on the local signal characteristic in gradient direction, i.e. based on whether they are related to a falling or raising edge; or to a local minimum or maximum. Depending on the detected case, the SAO process adds one of the offsets. In summary, the two SAO modes can reduce artifacts that occur at edges and in homogeneous areas. The SAO parameters can be selected by the encoder based on RD costs.

D.2 3D-High Efficiency Video Coding Techniques

Table D.1 shows an overview of the 3D-HEVC coding techniques. As this thesis focuses on depth coding, the following subsection will discuss the 3D-HEVC depth coding techniques, while Appendix D.2.2 will shortly address the texture coding techniques.

D.2.1 Depth Coding Techniques

For depth coding, 3D-HEVC modifies HEVC single-layer processes for CTU partitioning, prediction, and residual coding.

D.2.1.1 CTU and CB Partitioning

A 3D-HEVC technique that modifies CTU partitioning in CBs and also the division of CBs in PBs is quadtree limitation (QTL). QTL disallows to split CTBs and CBs in depth layers further than their co-located CTBs and CBs in the texture layer of the same view. This way, QTL reduces costs for partitioning signalization by assuming that structures in texture and depth are similar. A VPS flag can enable or disable QTL for the entire sequence.

D.2.1.2 Inter-picture Prediction

For inter-picture prediction, 3D-HEVC specifies two new candidates—the texture and the inter-view merge candidate. Both are added to the merge list so that an encoder can select them by

	Technique	Abbr.	Description	Depend.
Texture	Neighboring block disparity vector	NBDV	Derives predicted disparity information (PDI) for a CU	V, A, I
	Extended TMVP[1] for merge	-	Extends TMVP to also operate for inter-view prediction	V, A
	Inter-view motion prediction[2,3]	IV	Uses PDI for inter-view prediction of merge candidates	V
	Disparity info. merge cand.[3]	DI	Uses PDI directly as a merge candidate	-
	Residual prediction[3]	RP	Predicts residual from a different view or AU	V, A, A+V
	Illumination compensation	IC	Adapts an inter-view sample prediction to the current view	V, I
	Depth refinement[3]	DR	Improves PDI using disparity from a depth map	C+V
	View synthesis prediction[2,3]	VSP	Derives merge candidates from samples of a depth block	C+V, I
	Depth based block partitioning[3]	DBBP	Subdivides an inter-picture predicted CB based on a depth block	C+V, V, A
Depth	Inter-view motion prediction	IV	Uses a default disparity value to predict a merge candidate	V
	Full sample motion accuracy	-	Reduces ringing artifacts and complexity	-
	Intra_Wedge mode	-	Subdivides a PB by a straight line and predicts DCs	I
	Intra-picture skip / Intra_Single mode	-	Signals an angular mode or a single value for the entire CB	I
	DC offsets / DC-only mode	-	Codes residual DC explicitly / skips transform coefficients	-
	Depth look-up table	DLT	Quantizes the residual DC non-linearly	-
	Quadtree limitation	QTL	Derives a depth CB partitioning from texture	C
	Texture merge candidate[2]	T	Derives a merge candidate from texture	C
	Intra_Contour mode	-	Predicts a PB subdivision from texture and predicts DCs	C, I

Dependency from picture in different: V) View; C) Component A) AU; A+V) AU and View; C+V) Component and View; I) Intra-picture. [1]Temporal motion vector prediction. [2]Using sub-block partition (SBP) motion accuracy. [3]Depends on NBDV. [4]For RP from different view only.

Table D.1: 3D-HEVC texture and depth coding tools [Tech 16]. ©IEEE

signalling the respective merge index. The texture merge candidate corresponds to the motion information of the block that is collocated to the current CB in the texture layer of the current view. The inter-view merge candidate is derived from the motion information of a reference block in the depth layer of the current view. This way, 3D-HEVC exploits the similarity of motion in different views and components by inter-view and inter-component motion prediction. A further modification of inter-picture prediction in depth layers is that 3D-HEVC applies integer motion accuracy. This reduces singalling costs and avoids interpolation artifacts at sharp edges.

D.2.1.3 Intra-picture Prediction

For intra-picture prediction of depth maps, 3D-HEVC provides three new prediction modes—the intra wedge, the intra contour, and the intra single mode—and a new intra skip mode. The intra wedge and intra contour mode divide the PB in two sub block partitions (SBPs). After partitioning, they derive a DC value for each SBP from its adjacent reference samples and use this DC value as prediction. The intra contour mode derives the partitioning from a collocated block in the texture layer of the same view. This way, it exploits the similarity between texture and depth by inter-component prediction. In intra wedge mode, the encoder signals the partitioning explicitly by an index that refers to a set of predefined binary patterns, which are called wedgelets. The wedgelets divide the PB by a straight line.

The intra single mode uses the value of a particular reference sample as prediction for the entire PB. For this, the encoder can signal either the central sample on top of the PB or the central sample on the left of the PB. An encoder can signal the intra single mode only when it also selects the new intra skip mode. The intra skip mode, also called depth intra skip (DIS), can be signaled at CU level. When it is used, the CU consist of only one PB that is either predicted using the intra single mode or the vertical or horizontal angular mode. More specifically, in intra skip mode the CU only contains the conventional inter-picture skip flag (equal to 0), the intra-picture skip flag (equal to 1), and the index that selects among the horizontal intra angular mode, the vertical intra angular mode, the intra single mode using the top sample, or the intra single mode using the left sample. Other information like transform coefficients are not present. The intra skip mode thus provides an efficient way to signalize most common intra-picture prediction modes used for depth coding.

In summary, the 3D-HEVC depth intra-picture modes provide an efficient way to represent and signal sharp edges and homogeneous areas, which are common in depth maps.

D.2.1.4 Residual Coding

3D-HEVC exploits the special depth characteristics also in residual coding. For this, it specifies techniques for DC offset coding and the depth look-up table.

To refine the DC of PBs or SBPs, 3D-HEVC allows to signal DC offset values and to add them to the prediction signal. In intra wedge and intra contour mode an encoder can signal two DC offsets, one for each SBP. In addition both modes can use quantized transform coefficients. For

other prediction modes, an encoder can signal a single DC offset for the entire PB, but only when it signals DC-only, which means that quantized transform coefficients are not present.

To efficiently represent DC offsets, 3D-HEVC provides a depth look-up table that can be enabled and specified in the PPS. When the DLT is enabled, DC offsets are transmitted in a compressed value range to exploit that depth maps often use the available sample value range (of e.g. 8 bit) only sparsely.

D.2.2 Texture Coding Techniques

In addition to advanced depth coding, 3D-HEVC also targets improved texture coding. For this, it specifies several tools as listed in Table D.1.

A new technique for texture coding, which is the basis for several other new tools, is neighbouring block disparity vector derivation (NBDV). The decoder invokes NBDV at CU level to derive a disparity vector and a respective index to a reference view [jointly called predicted disparity information (PDI)] from temporal and spatial neighboring blocks. Subsequently, when depth maps are present, the decoder applies another new technique called depth refinement (DR). DR uses depth maps to refine the PDI. The PDI is then used by other texture coding techniques for inter-view prediction of samples, motion, or partitioning information.

Techniques for motion derivation that are based on the PDI and that are applied in merge mode are inter-view motion prediction, view synthesis prediction, and the disparity information candidate. Inter-view motion prediction derives motion vectors in SBP granularity from the texture of another view. View synthesis prediction converts depth values of a depth map of another view to motion vectors, which can then be used for inter-view sample prediction. Finally, the PDI can be directly added to the merge list and is then called disparity information (DI) candidate.

A technique that uses the PDI for the prediction of partitioning information is depth-based block partitioning (DBBP). More specifically, DBBP uses the PDI to identify a block in depth map of another view and divides the current PB based on a contour derived from the reference depth block. DBBP derives the sample values of the resulting SBPs independently using inter-picture prediction.

Another tool that is used in inter-picture sample prediction is residual prediction. Residual prediction uses the PDI to identify two blocks in two different pictures of a reference view. The blocks correspond to the current PB and its reference block. A prediction for the current PB's residual can be derived by subtracting their sample values.

The 3D-HEVC coding tools discussed above exploit the depth information either directly (e.g. VSP, DR, and DBBP) or indirectly by using the PDI derived from by DR (e.g. VSP, the IV and DI merge candidates). However, 3D-HEVC also comprises two techniques—extended TMVP and illumination compensation—that do not require the PDI. Extended TMVP scales motion vectors used for inter-view prediction based on view index values; illumination compensation adapts inter-view predicted samples to the current view by scaling and offsetting.

D.3 Description of the Exemplary Encoder

The selection modules in middle part of the encoder in Figure 6.1 test different coding modes, i.e. syntax element combinations. The reconstruction modules (left) correspond to the modules of the decoder. They are used to derive the reconstruction related to the tested coding modes. The binary encoder (right) provides bit rates of the tested coding modes and writes the syntax element values of the finally selected coding modes to the bitstream.

The *partitioning selection* module tests different partitionings of the CTB in CBs. For each partitioning, it operates as follows:

1. For each CB, it invokes the prediction syntax selection step to derive the associate RD costs for encoding it.
2. It invokes the binary encoder to determine the bit-rate for signalling the partitioning.
3. It derives the total RD cost of the partitioning as sum of the RD costs of its CBs and the RD cost for signaling the partitioning parameters.

Finally, it selects the partitioning with the lowest RD cost.

The *prediction syntax selection* module tests different combinations of PB partitionings, prediction types, and prediction parameters. For each tested combination, it operates as follows:

1. It invokes the prediction module, which performs the associated prediction.
2. It invokes the residual syntax selection module to encode the prediction's residual.
3. It invokes the binary encoder to determine the bit-rate for signalling the tested combination.
4. It derives the total RD cost as sum of the RD cost provided by residual coding module and the RD cost required for signaling the tested combination.

Finally, it selects the combination with the lowest RD cost.

Input to *residual syntax selection module* is the difference of the original signal and prediction, in the following called original residual. To encode it, the module tests different combinations of TB partitionings and quantized transform coefficients. For each tested combination, it operates as follows:

1. It invokes the residual reconstruction module to obtain the reconstructed residual.
2. It compares the reconstructed and the original residual to determine the distortion.
3. It invokes the binary encoder to determine the bit rate for signalling the tested combination.
4. It determines the total RD cost of the combination from the bit rate and the distortion.

Finally, it selects the combination with the lowest RD cost.

The *filter parameter selection* module tests different filter parameters. For the deblocking filter, encoders conventionally signal fixed parameters depending on the QP. SAO parameters are conventionally directly determined from the reconstructed and the original picture, before deciding whether to signaled them based on their RD costs.

E Measures and Test Sequences

E.1 Distortion Measures

E.1.1 Peak Signal-to-Noise Ratio

The SV texture PSNR D_P represents the ratio between the SV texture's maximal possible distortion and SV texture's distortion with respect to the reference SV texture in decibels, i.e.

$$D_P = 10 \cdot \log_{10} \frac{255^2 \cdot w \cdot h}{D} \tag{E.1}$$

with 255 as maximal sample value, D as sum of squared differences between the tested and the reference SV texture, and w and h denoting the SV texture width and height, respectively.

E.1.2 Structural Similarity Index

The Structural Similarity index [Wang 04] is the product of a luminance index l, a contrast index c, and a structure index s, which measure the local similarity of two pictures s_X and s_Y:

$$\text{SSIM} = l \cdot c \cdot s \tag{E.2}$$

Index l measures the similarity between the local mean values μ_X and μ_Y of s_X and s_Y, respectively as

$$l = \frac{2\mu_X\mu_Y + C_1}{\mu_X^2 + \mu_Y^2 + C_1}. \tag{E.3}$$

C_1 (and C_2 and C_3 in the following) are constants introduced for stability for cases in which the denominator is close to zero.[1] Index c represents the similarity between the local standard deviations σ_X and σ_Y of the two pictures s_X and s_Y, respectively, and is given by

$$c = \frac{2\sigma_X\sigma_Y + C_2}{\sigma_X^2 + \sigma_Y^2 + C_2}. \tag{E.4}$$

Index s is a measure for local correlation of s_X and s_Y:

$$s = \frac{\sigma_{XY} + C_3}{\sigma_X \cdot \sigma_Y + C_3} \tag{E.5}$$

with σ_{XY} denoting the local covariance between s_X and s_Y. As proposed in [Wang 04], SSIM computation in this thesis first derives the local SSIM indices (and thus local averages, vari-

[1] As proposed in [Wang 04], SSIM computation in this thesis uses $C_1 = (0.01 \cdot 255)^2$, $C_2 = (0.03 \cdot 255)^2$, and $C_3 = C_2/2$.

ances, and covariances) in overlapping windows of sizes 11×11 and using a Gaussian weighting function with a standard deviation of 1.5 samples. Then, it averages the local SSIM indices to a mean SSIM index (MSSIM). In the evaluations, s_X and s_Y represent the tested and the reference SV textures $\tilde{s}'_T$ and $s'_{T,Ref}$, respectively.

E.1.3 Pratt's Figure of Merit

Pratt's Figure of Merit (FOM) [Prat 78] is a measure for the similarity of edge locations in two pictures. FOM computation requires to identify edges in the reference and the tested SV texture. For this thesis, this was done with Canny's filter [Cann 86].[2] Knowing the edge positions the FOM is then given by

$$\text{FOM} = \frac{1}{\max(N_x, N_y)} \cdot \sum_{i=1}^{N_x} \frac{1}{1 + 0.11 \cdot r_i^2} \tag{E.6}$$

with N_x and N_y denoting the numbers of edge samples in the reference and the tested SV texture, respectively, and r_i denoting the Euclidean distance between the edge sample i in the tested SV texture to its closest edge sample in the reference SV texture. The FOM is thus 1 when edge positions in the reference and the tested SV texture are identical and less than 1, otherwise. E.g. when N_x is equal to N_y and the average edge position deviation is one sample distance, the FOM is 0.9.

E.2 Rate-Distortion Performance and Complexity Measures

E.2.1 Depth Bit Rate Delta

The delta of the depth bit rate R_Δ achieved by a tested method in comparison to a reference method is calculated according to the Bjøntegaard Delta method [Bjon 01]. With $\bar{R}_T(D_P)$ and $\bar{R}_R(D_P)$ denoting the decadic logarithm of the depth bit rates achieved by the tested and the reference method, respectively, at the SV texture PSNR D_P, the difference of $\bar{R}_T(D_P)$ and $\bar{R}_R(D_P)$ is averaged over an SV texture PSNR range $[D_{P,s}, D_{P,e}]$:

$$\delta = \int_{D_{P,s}}^{D_{P,e}} \left[\bar{R}_T(D_P) - \bar{R}_R(D_P)\right] dD_P \tag{E.7}$$

$$R_\Delta = 10^\delta - 1 \tag{E.8}$$

$\bar{R}_T(D_P)$ and $\bar{R}_R(D_P)$ are interpolated using piecewise cubic polynomials from the RD points obtained in the simulations.

[2] With the standard deviation of 1.5 for the Gaussian filter and lower and higher hysteresis threshold chosen as 10 and 15, respectively.

Sequences	Depth bit rate [kbit/s] Lower limit	Upper limit
Ballons	8	200
Kendo	9	230
Newspaper	10	300
PoznanHall2	6	110
PoznanStreet	8	275
GTFly	12	370
Shark	16	640
UndoDancer	15	290

Table E.1: Depth bit rate limits for BD computations.

E.2.2 Synthesized View Texture PSNR Delta

The Bjøntegaard Delta method is also used to calculate the delta of the SV texture PSNR $D_{\Delta P}$. With $D_{P,T}(\bar{R}_T)$ and $D_{P,R}(\bar{R}_T)$ denoting the SV texture PSNR at the logarithmized depth bit rate $\bar{R}_T$ achieved by the tested and the reference method, respectively, the difference of $D_{P,T}(\bar{R}_T)$ and $D_{P,R}(\bar{R}_T)$ is averaged over the logarithmized depth bit rate range $\left[\bar{R}_s, \bar{R}_e\right]$:

$$D_{\Delta P} = \int_{\bar{R}_s}^{\bar{R}_e} \left[D_{P,T}(\bar{R}_T) - D_{P,R}(\bar{R}_T)\right] d\bar{R}_T \tag{E.9}$$

$D_{P,T}(\bar{R}_T)$ and $D_{P,R}(\bar{R}_T)$ are interpolated using piecewise cubic polynomials from the RD points obtained in the simulations.

E.2.3 Evaluated Depth Bit Rate and PSNR Ranges

Conventionally, average depth bit rate and PSNR deltas are calculated using for each evaluated method four RD points defined by fixed QPs in encoding. The minimal and maximal PSNRs and bit rates achieved by these QPs define the respective evaluated ranges. However, in evaluations in this thesis SVDs and depth bit rates can vary significantly at fixed QPs for different tested methods. When using four QPs, this can cause an insufficient overlap of the tested and the reference method's RD curves. For this, simulations in this thesis use six depth QPs (see Table 7.2). This leads to large depth bit rate and SV PSNR ranges. To evaluate changes in parts of these that are of practical relevance, $[D_{P,s}, D_{P,e}]$ and $\left[\bar{R}_s, \bar{R}_e\right]$ have been limited in this thesis. Depth bit rate limits used to determine $\left[\bar{R}_s, \bar{R}_e\right]$ are given in Table E.1. They are approximately the depth bit rates achieved for coding two IVs with common test condition defined by the JCT-3V [Mull 14]. Limits for $[D_{P,s}, D_{P,e}]$ are given by the PSNR values that are achieved by the reference method at these depth bit rates. (Exceptions are comparisons of SVDC- to depth distortion- or SVD estimation-based encoding, which use always the SVDC-based methods instead of the reference method.) This way, problems due to insufficiently overlapping RD curves are mitigated while still considering reasonable depth bit rate and SV PSNR ranges.

Name	Abbreviation	Type	Pictures	Format	Provided by
Balloons	–	Natural	300	1024×768	[Tani 09b]
Kendo	–	Natural	300	1024×768	[Tani 09b]
Newspaper	*Newsp.*	Natural	300	1024×768	[Lee 09]
PoznanHall2	*P.Hall2*	Natural	200	1920×1088	[Doma 09]
PoznanStreet	*P.Street*	Natural	250	1920×1088	[Doma 09]
GTFly	–	CG	250	1920×1088	[Hann 11]
Shark	–	CG	300	1920×1088	[Shim 13]
UndoDancer	*UndoD.*	CG	250	1920×1088	[Hann 11]

Table E.2: Overview of test sequences.

E.2.4 Encoding Time Delta

The encoding time delta T_Δ achieved by a tested method in comparison to a reference method is calculated as follows: 1) For each method, the geometric mean of encoding times at the evaluated QPs is calculated (as in test conditions of the JCT-3V [Mull 14]). 2) The reference method's geometric mean is substracted from the tested method's geometric mean. 3) The difference is expressed in percent of the reference method's geometric mean.

Encoding times have been measured on systems with Intel Xeon E5-2697A v4 2.60 GHz processors with disabled turbo boost and 2400 MHz memory speed. The systems used Scientific Linux 7.

E.3 Test Sequences

The test sequences used in this thesis correspond to the test set used by JCT-3V [Mull 14]. The set includes five natural and three computer generated sequences as shown in Table E.2.

PoznanHall2

PoznanStreet

UndoDancer

GTFly

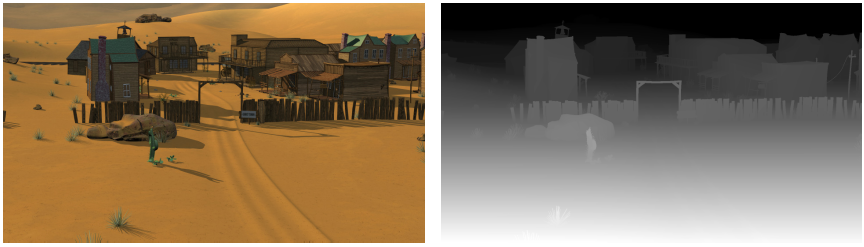
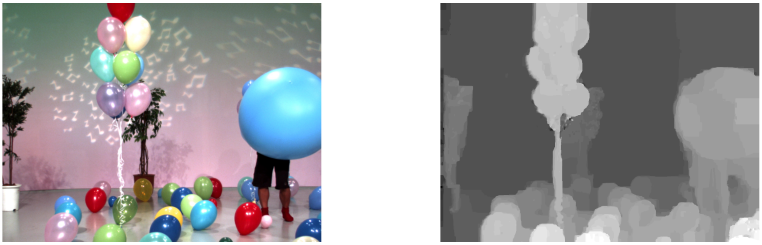


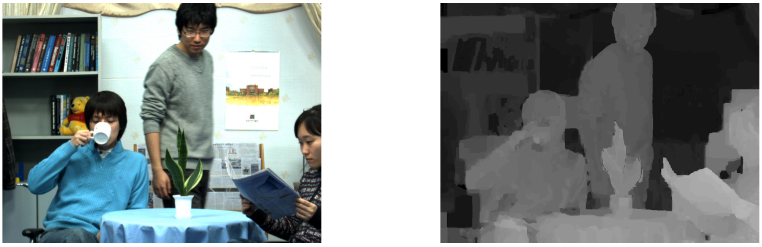
Figure E.1: Sample pictures of test sequences.

Kendo

Balloons



Newspaper



Shark

Figure E.2: Sample pictures of test sequences.

F Additional Coding Results

F.1 Initial Evaluation

	Depth encoding time in % of MV-HEVC texture encoding time			
	MV-HEVC		3D-HEVC	
	DD	SVDC	DD	SVDC
Balloons	93	104	20	37
Kendo	98	110	23	37
Newsp.	98	112	22	41
P.Hall2	95	106	18	36
P.Street	95	107	19	38
GTFly	88	103	21	42
Shark	91	102	21	40
UndoD.	84	95	18	33
Mean	93	105	20	38

Table F.1: Depth encoding time of depth distortion (DD)- and SVDC-based encoding in % of the texture encoding time in MV-HEVC.

F.2 Effects of SVDC-Based Encoding on Mode Selection

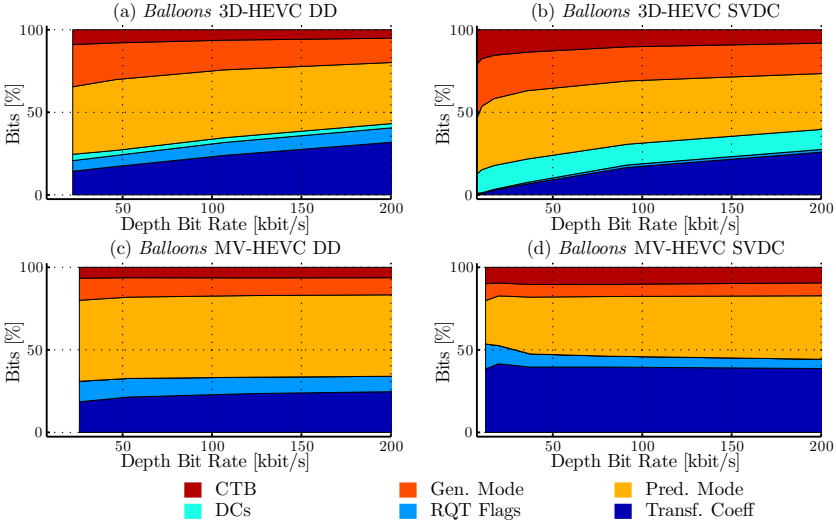


Figure F.1: Bit allocation to different syntax element categories. *Balloons*.

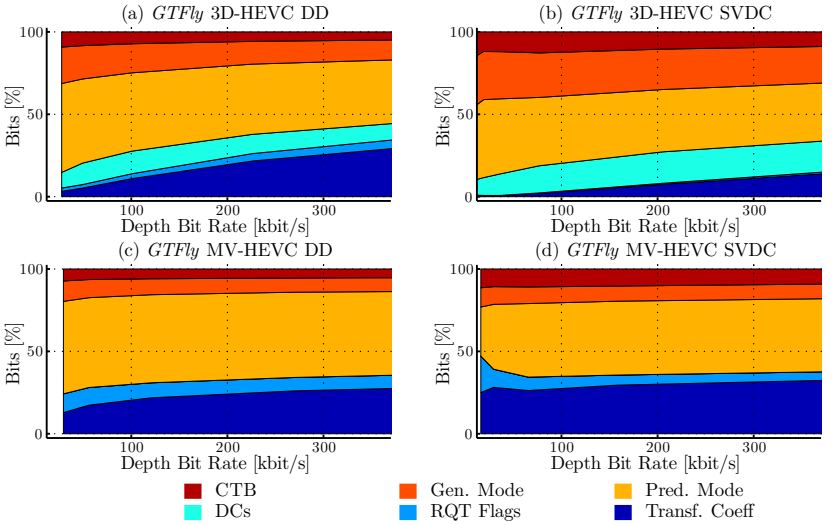


Figure F.2: Bit allocation to different syntax element categories. *GTFly*.

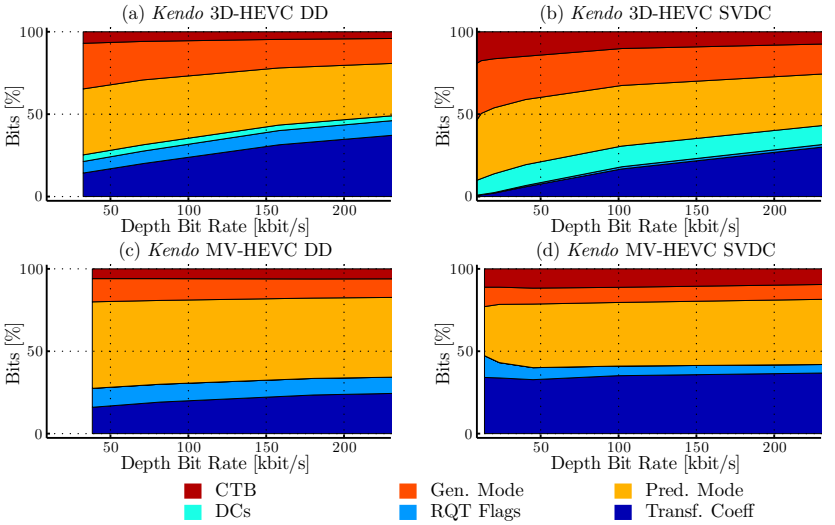


Figure F.3: Bit allocation to different syntax element categories. *Kendo*.

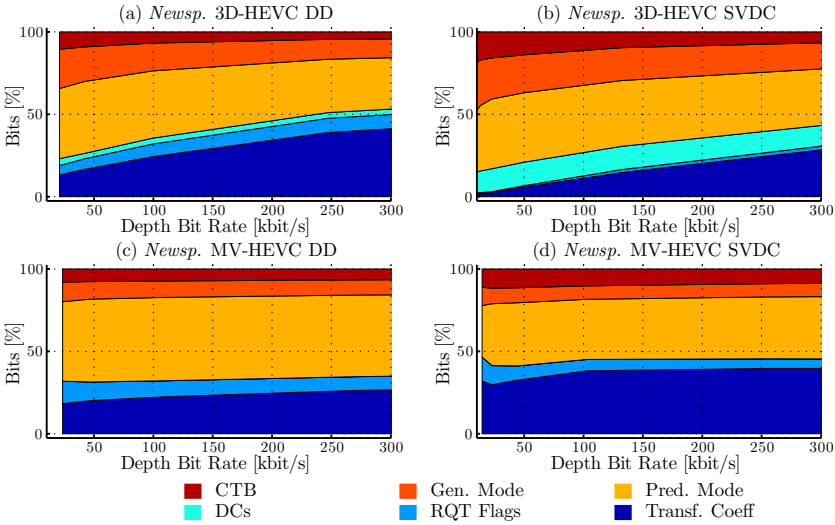


Figure F.4: Bit allocation to different syntax element categories. *Newsp*.

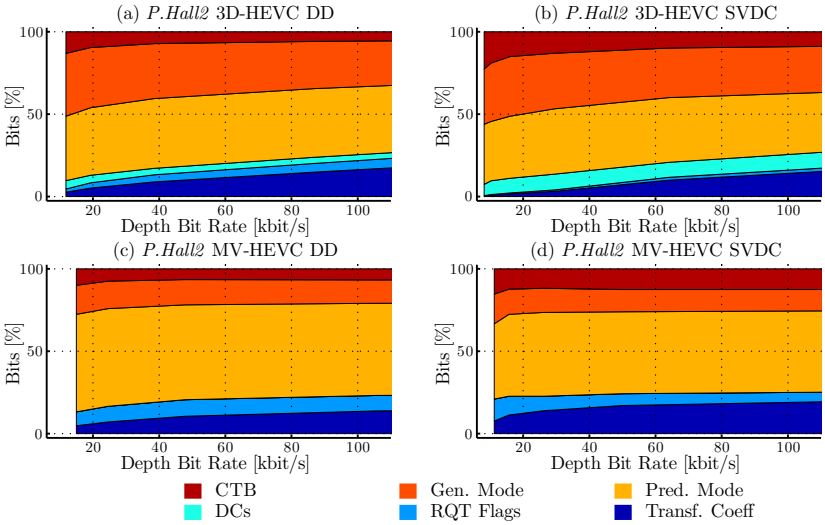


Figure F.5: Bit allocation to different syntax element categories. *P.Hall2*.

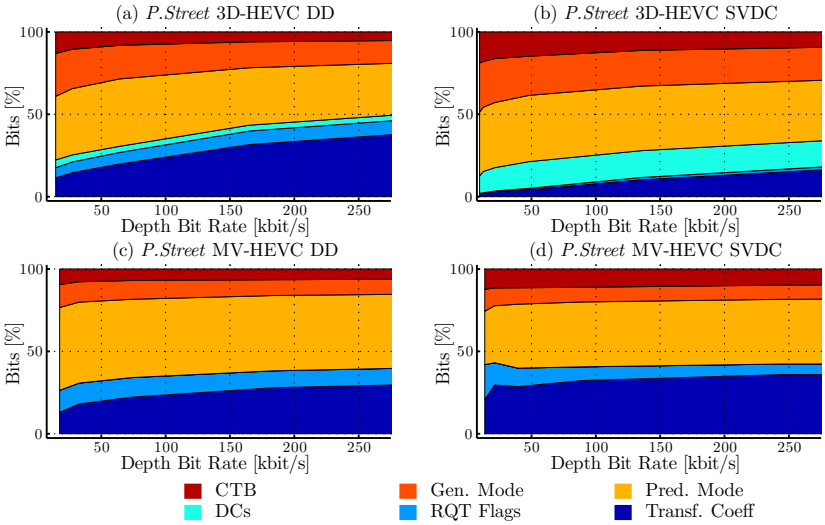


Figure F.6: Bit allocation to different syntax element categories. *P.Street*.

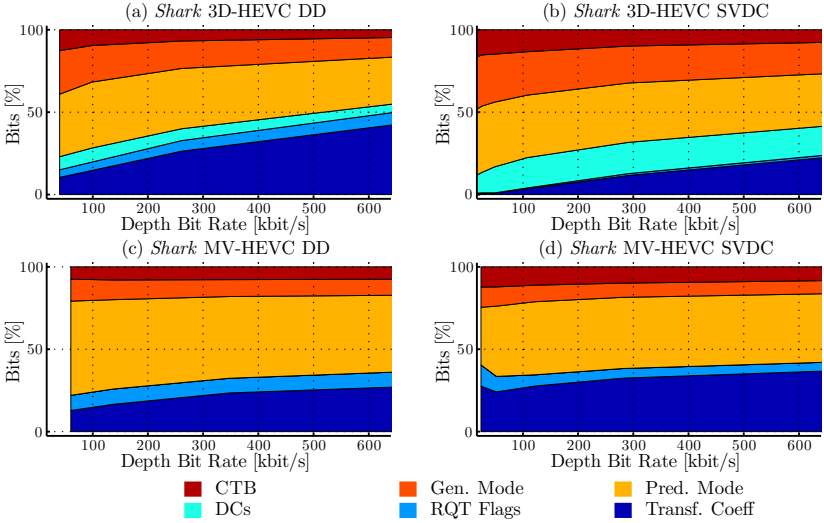


Figure F.7: Bit allocation to different syntax element categories. *Shark*.

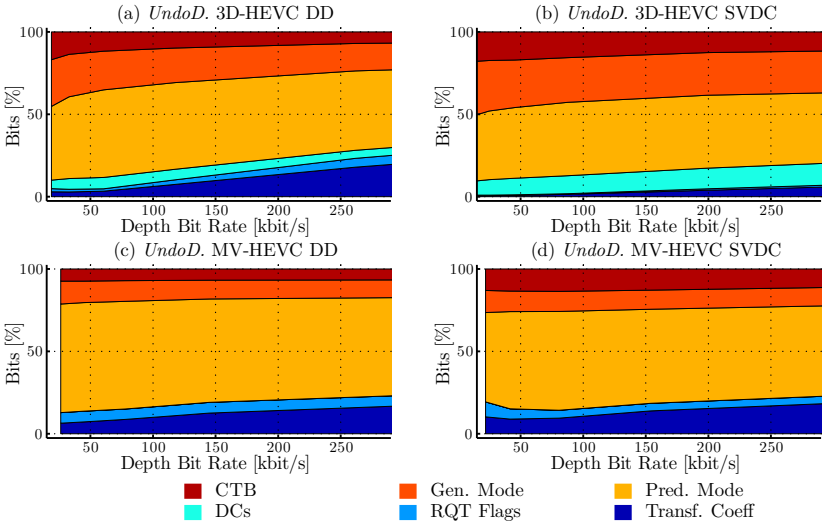


Figure F.8: Bit allocation to different syntax element categories. *UndoD.*

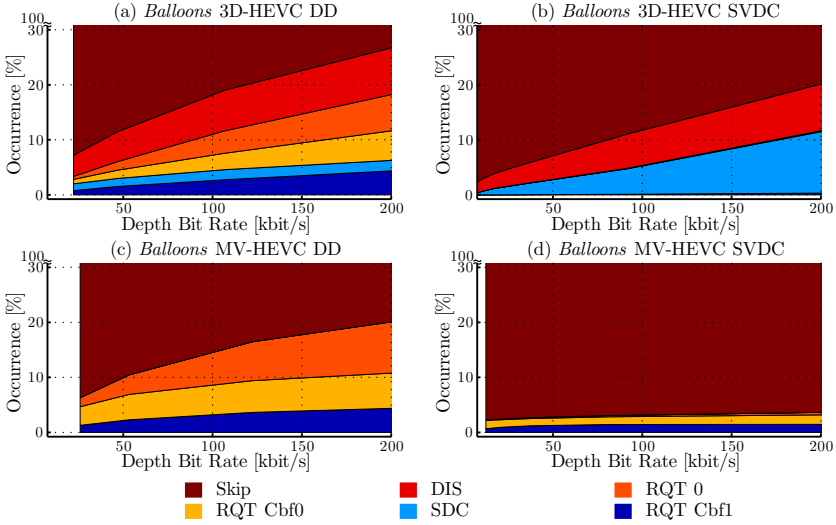


Figure F.9: Occurrence of samples coded in different residual coding modes. *Balloons*

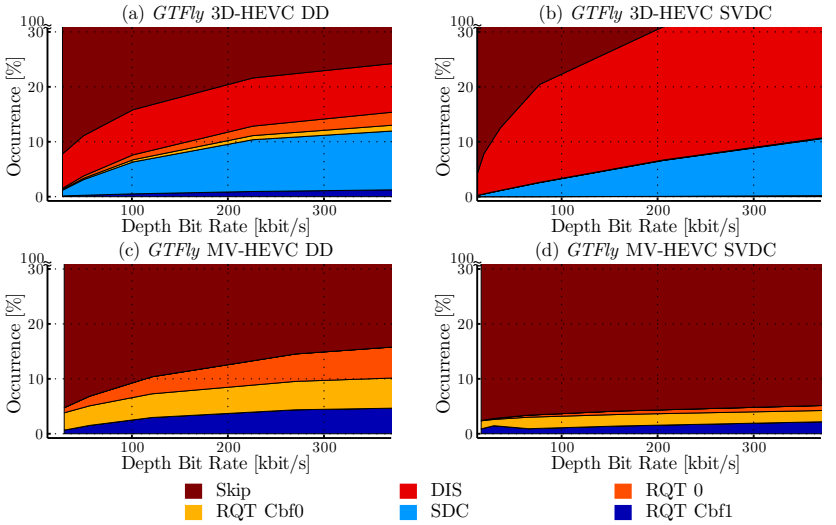


Figure F.10: Occurrence of samples coded in different residual coding modes. *GTFly*

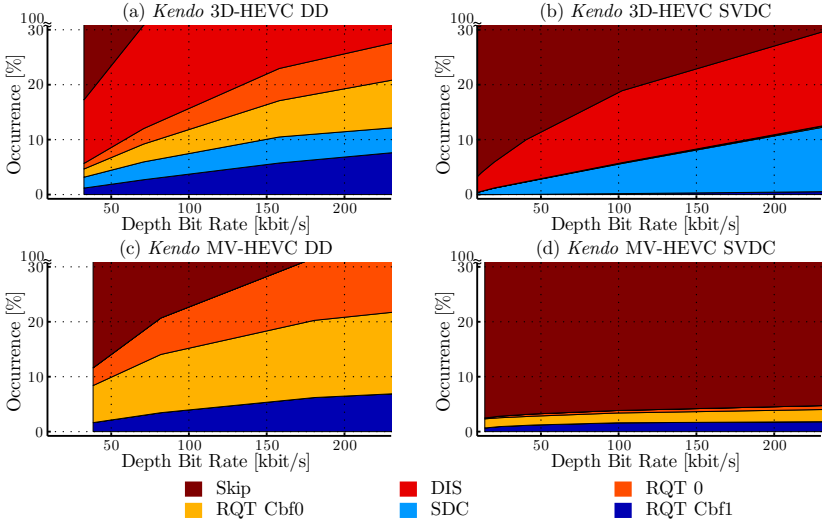


Figure F.11: Occurrence of samples coded in different residual coding modes. *Kendo*

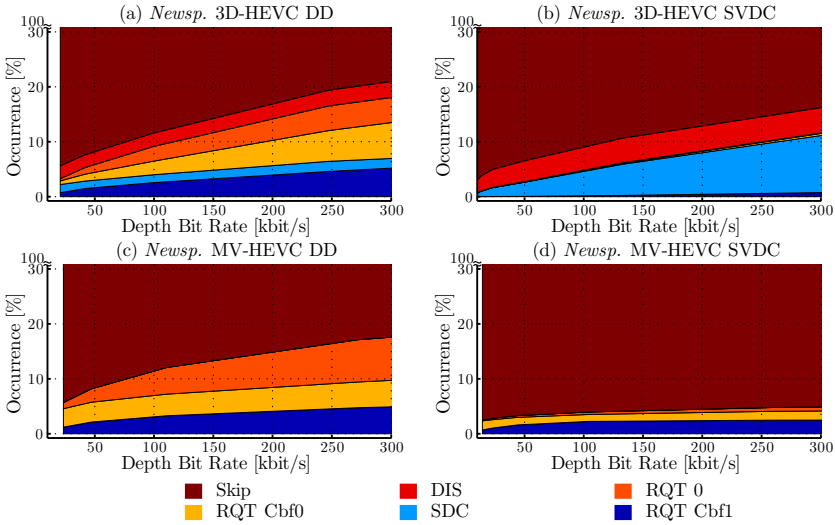


Figure F.12: Occurrence of samples coded in different residual coding modes. *Newsp.*

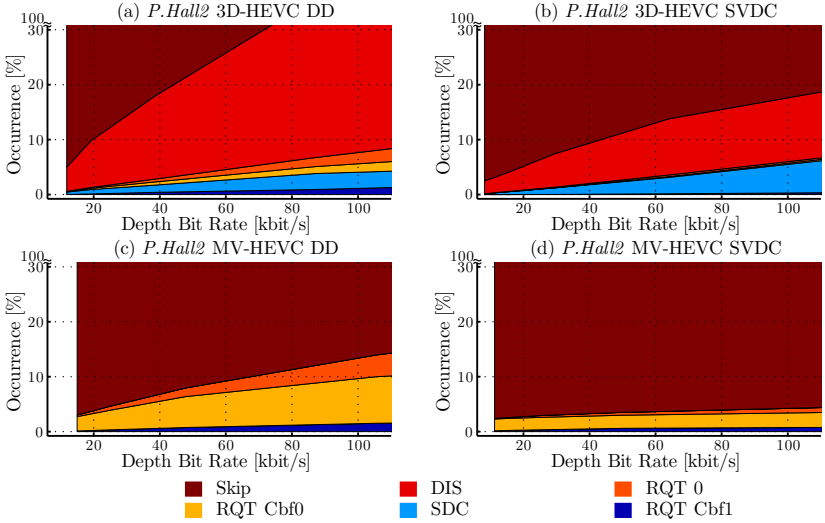


Figure F.13: Occurrence of samples coded in different residual coding modes. *P.Hall2*

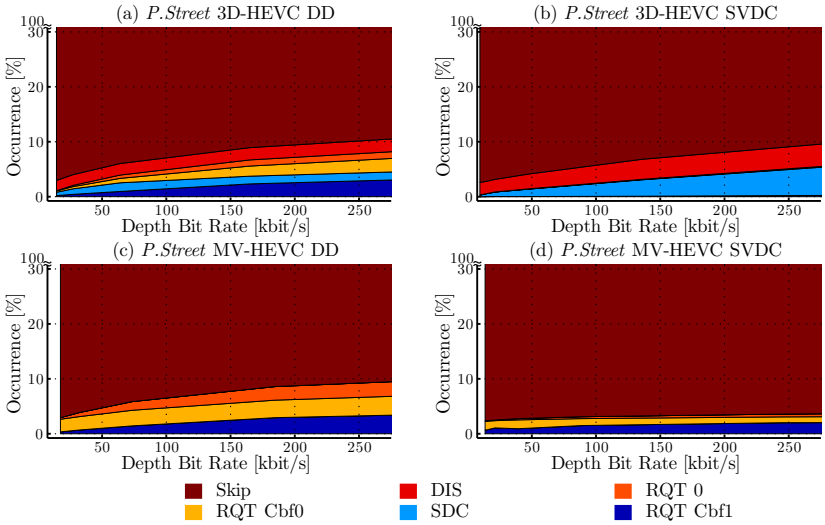


Figure F.14: Occurrence of samples coded in different residual coding modes. *P.Street*

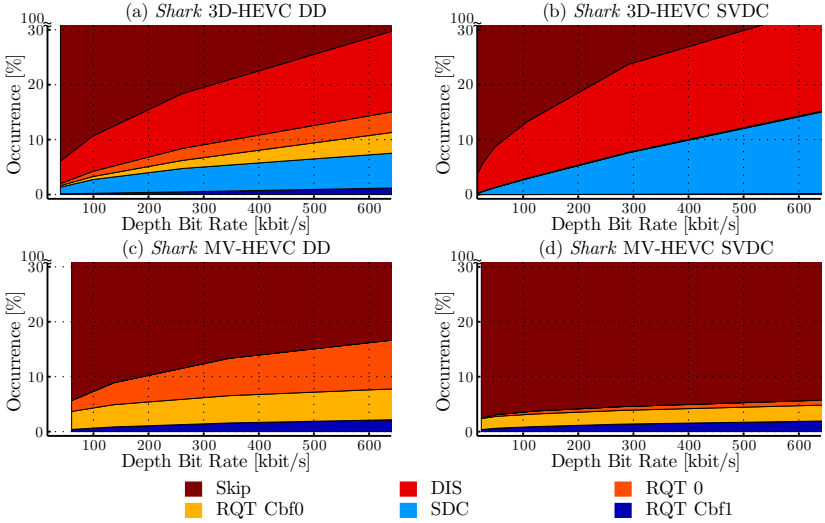


Figure F.15: Occurrence of samples coded in different residual coding modes. *Shark*

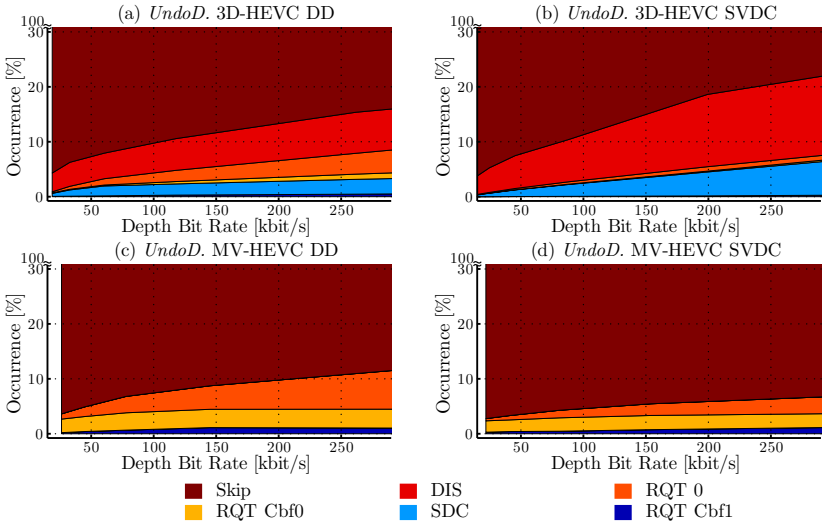


Figure F.16: Occurrence of samples coded in different residual coding modes. *UndoD.*

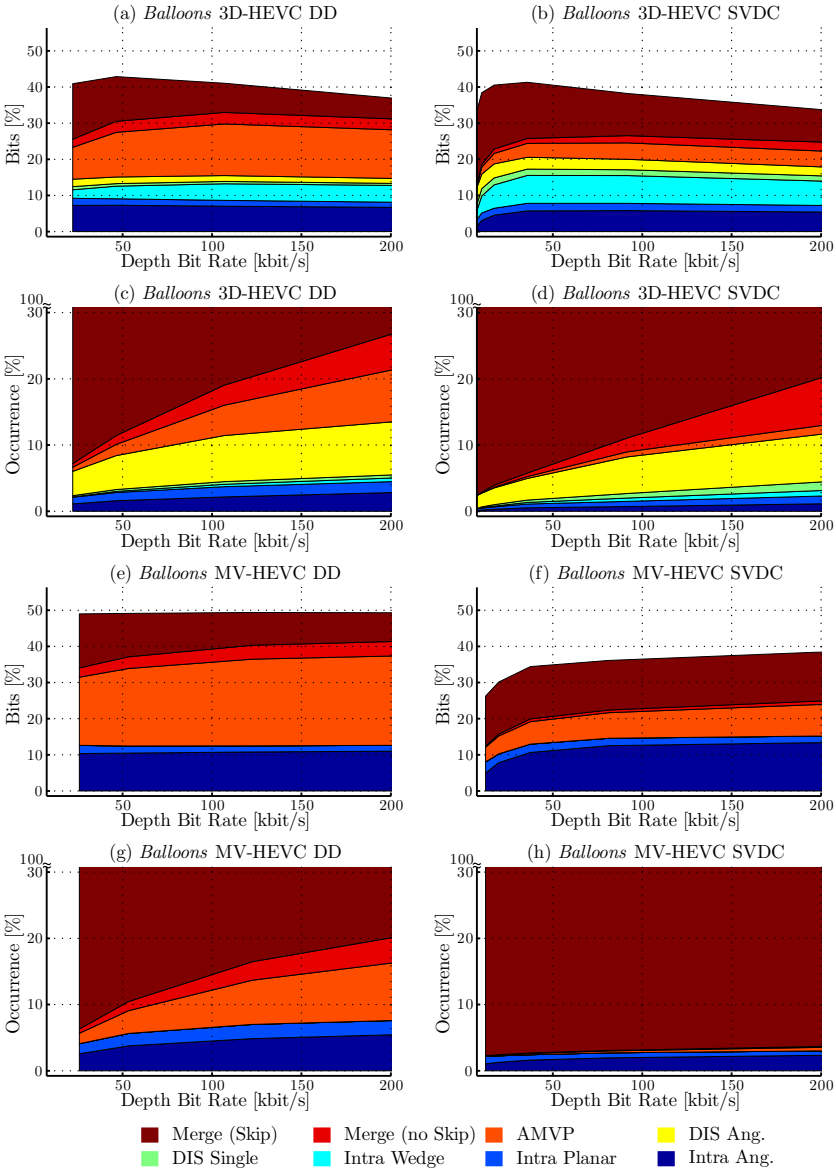


Figure F.17: Bit rate allocation and occurrence, prediction modes. *Balloons*.

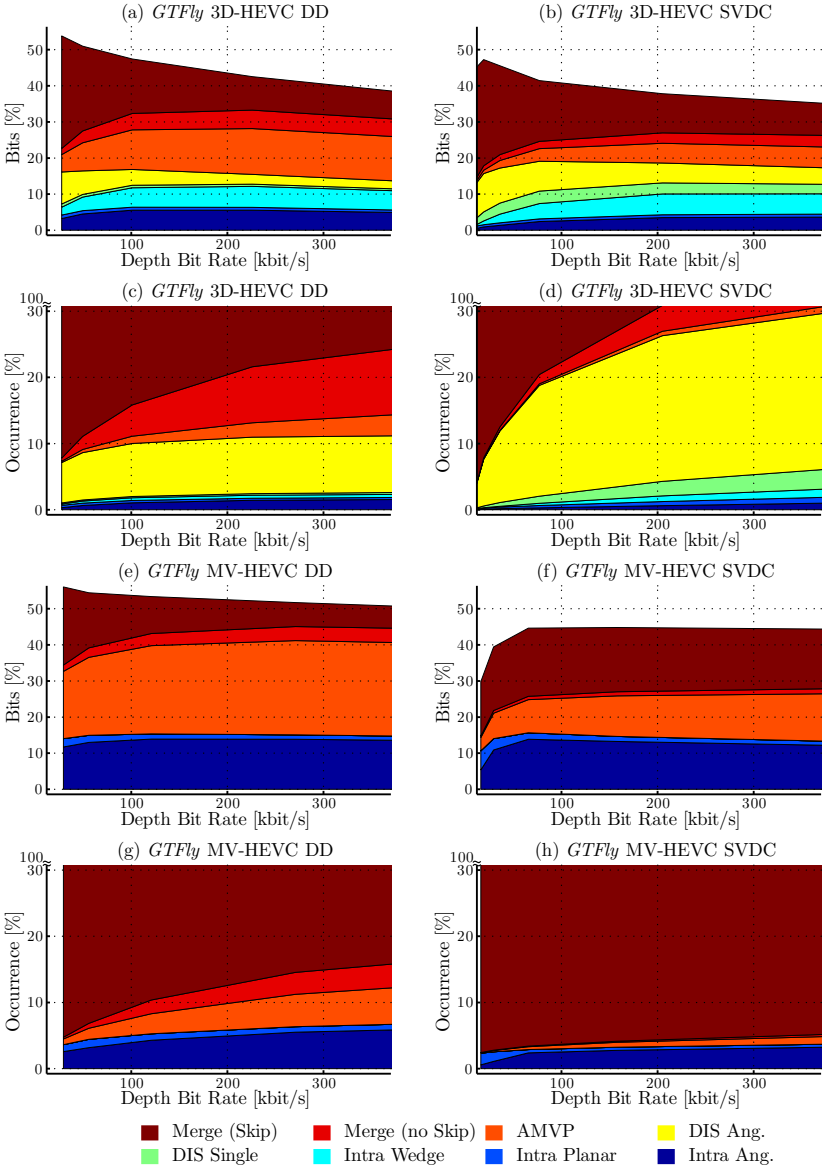


Figure F.18: Bit rate allocation and occurrence, prediction modes. *GTFly*.

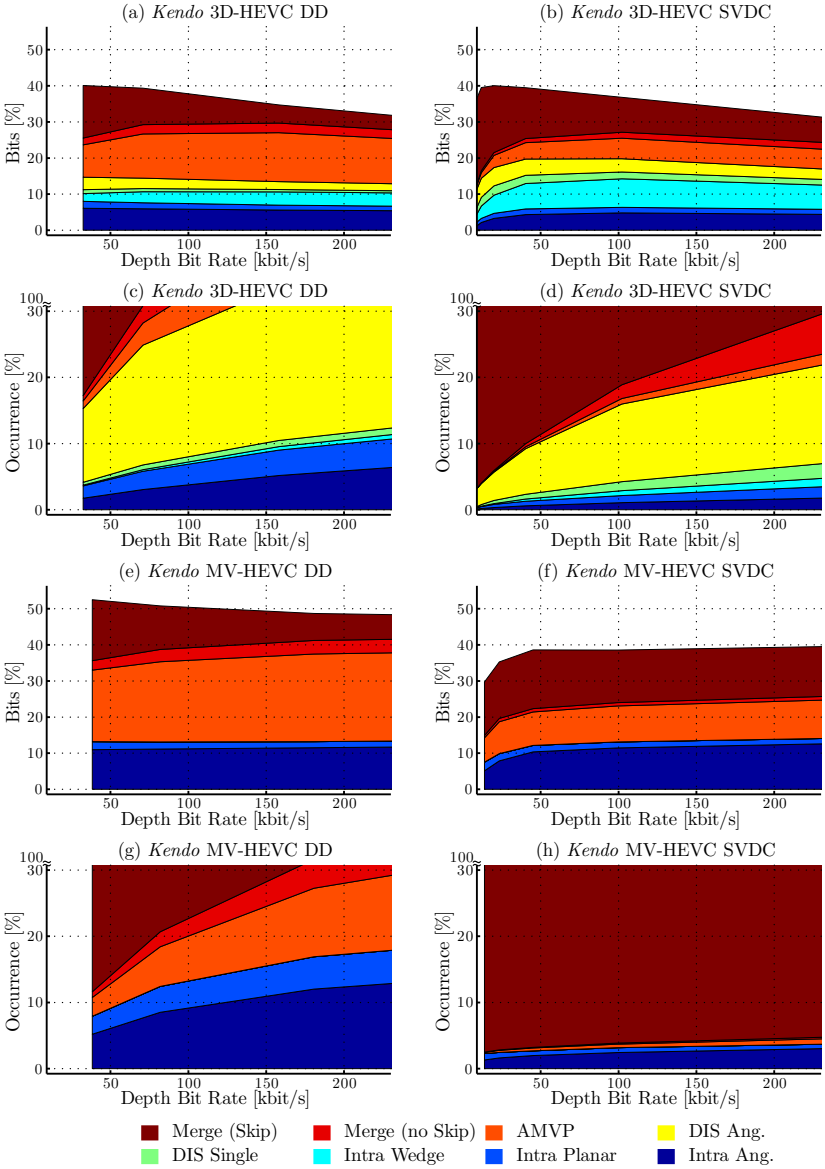


Figure F.19: Bit rate allocation and occurrence, prediction modes. *Kendo*.

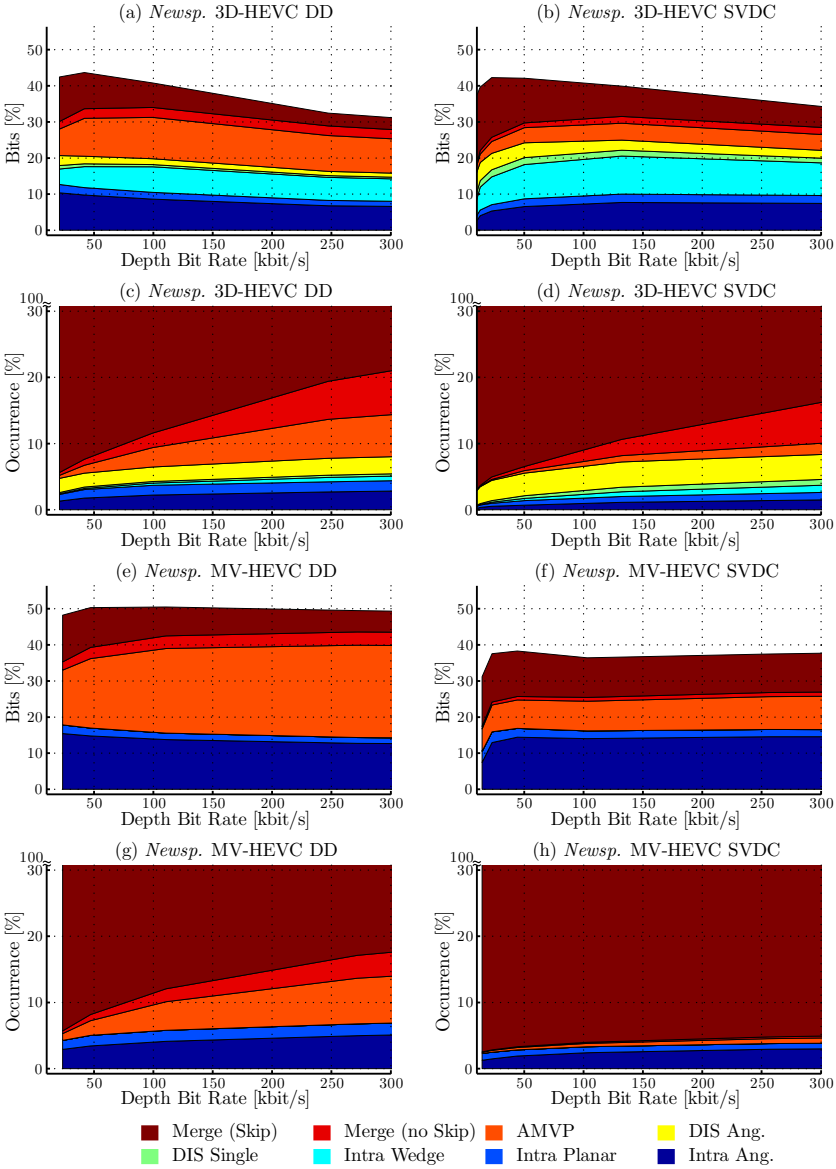


Figure F.20: Bit rate allocation and occurrence, prediction modes. *Newsp*.

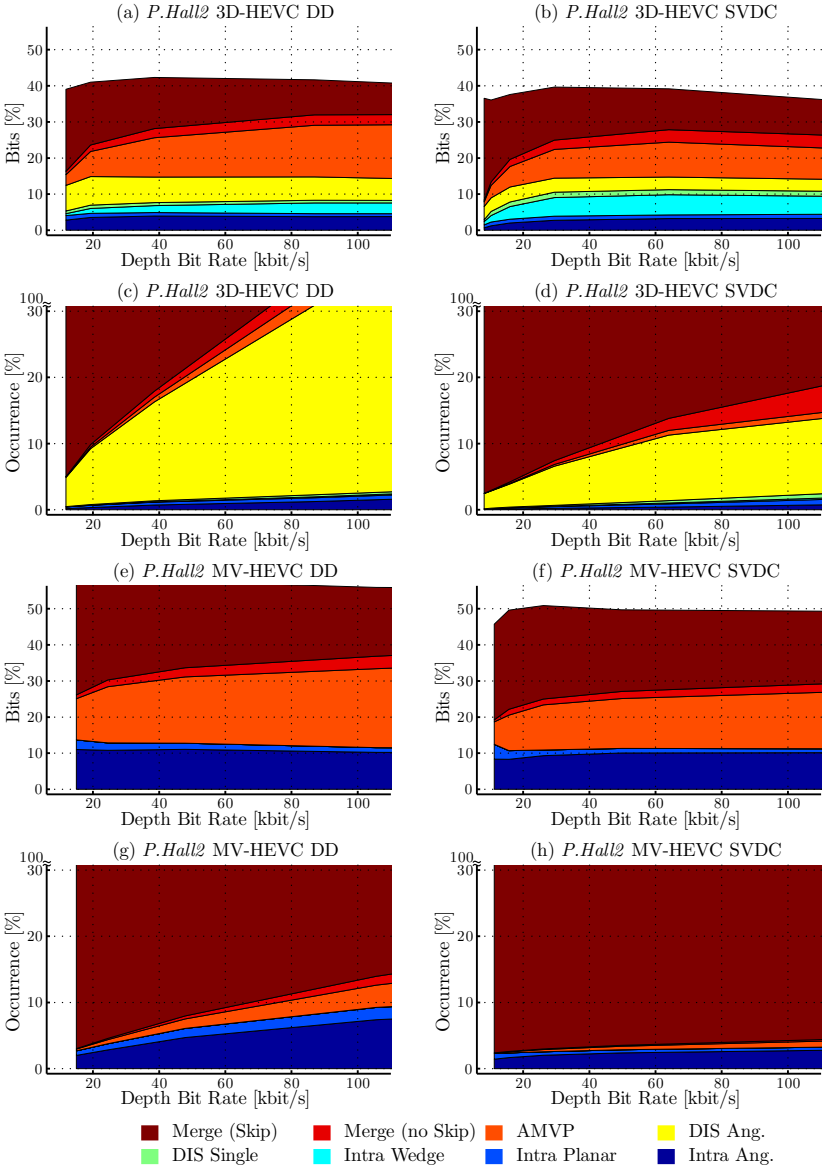


Figure F.21: Bit rate allocation and occurrence, prediction modes. *P.Hall2*.

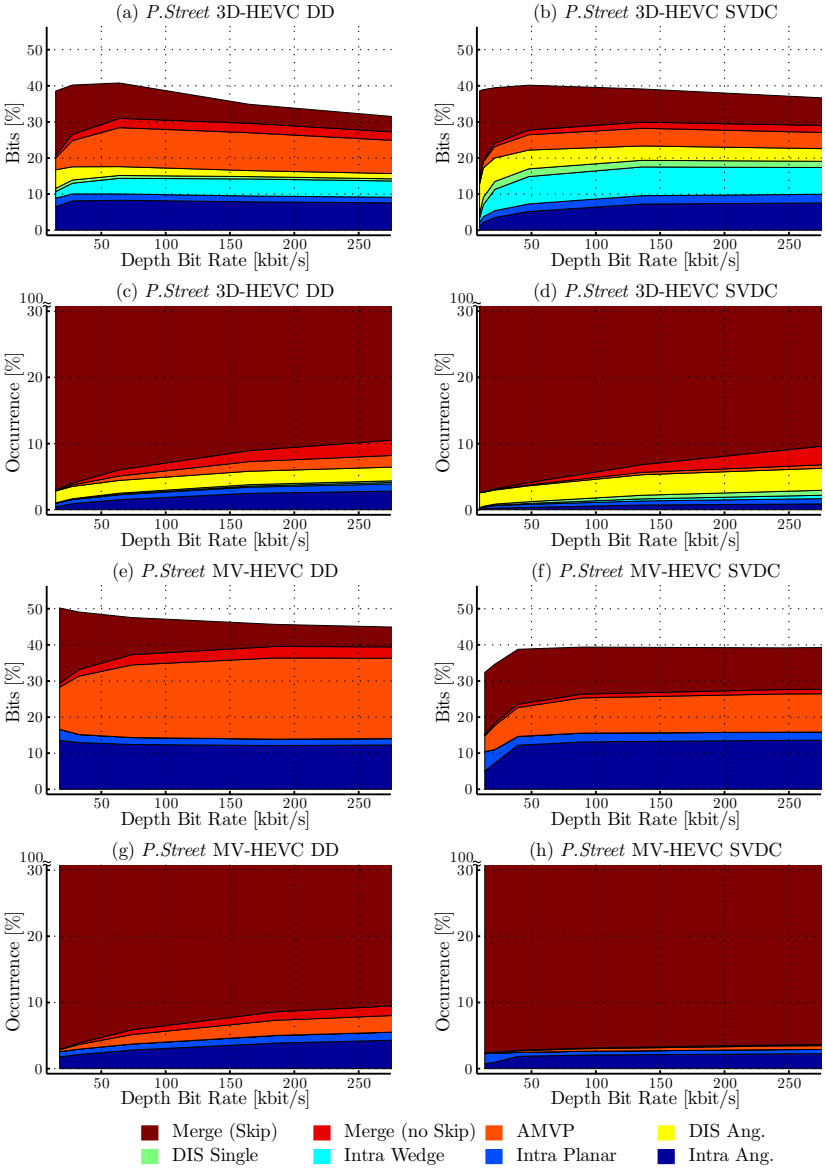


Figure F.22: Bit rate allocation and occurrence, prediction modes. *P.Street*.

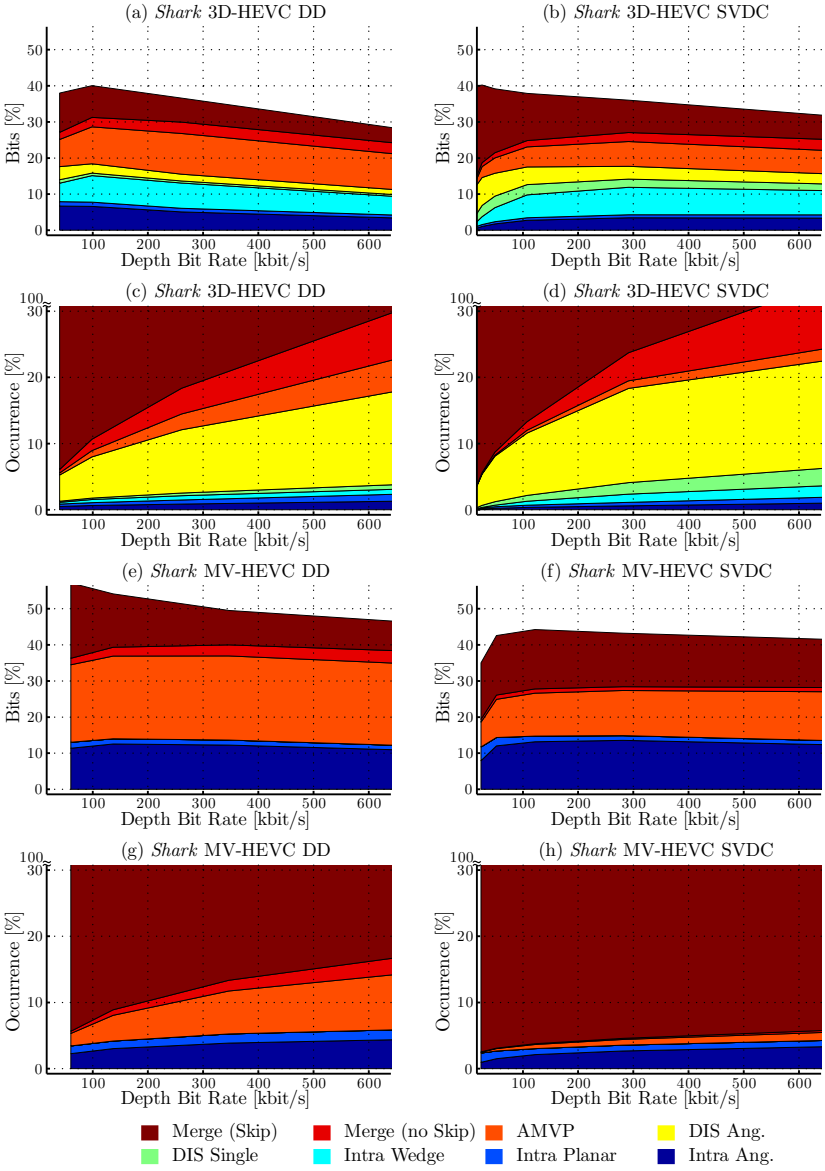


Figure F.23: Bit rate allocation and occurrence, prediction modes. *Shark*.

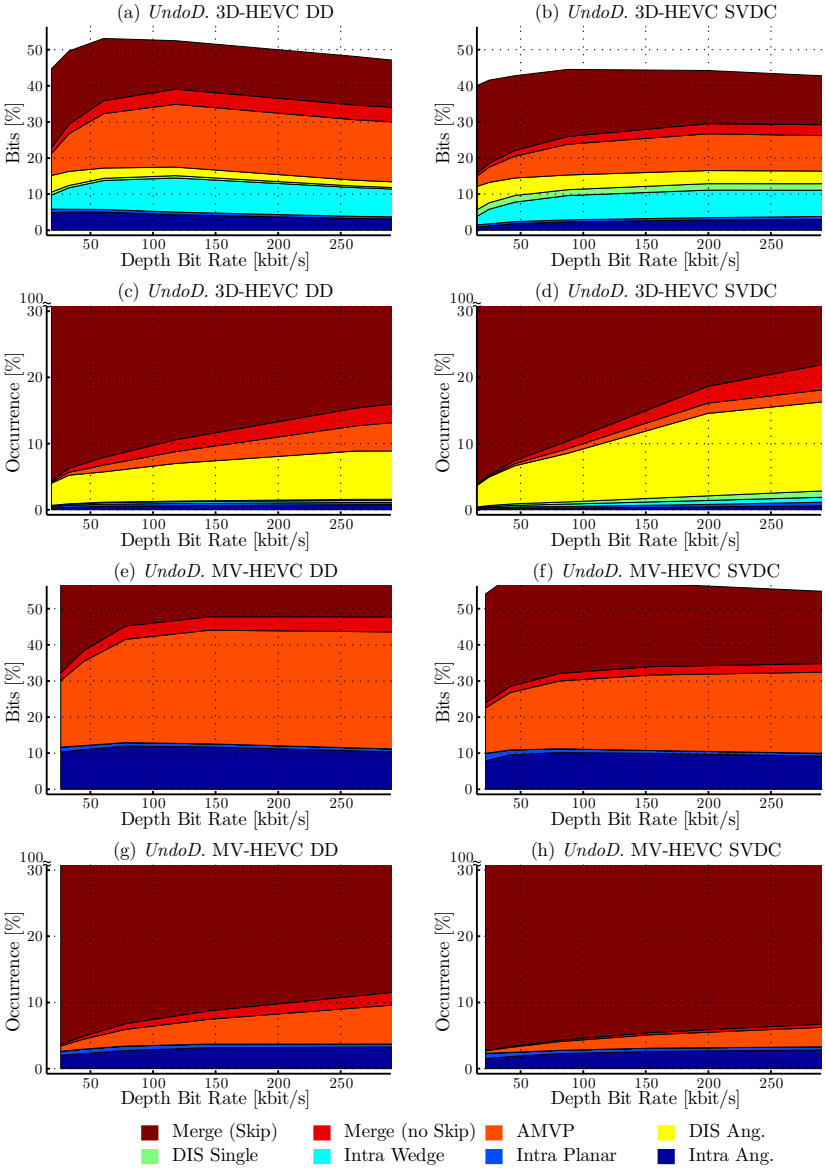


Figure F.24: Bit rate allocation and occurrence, prediction modes. *UndoD.*

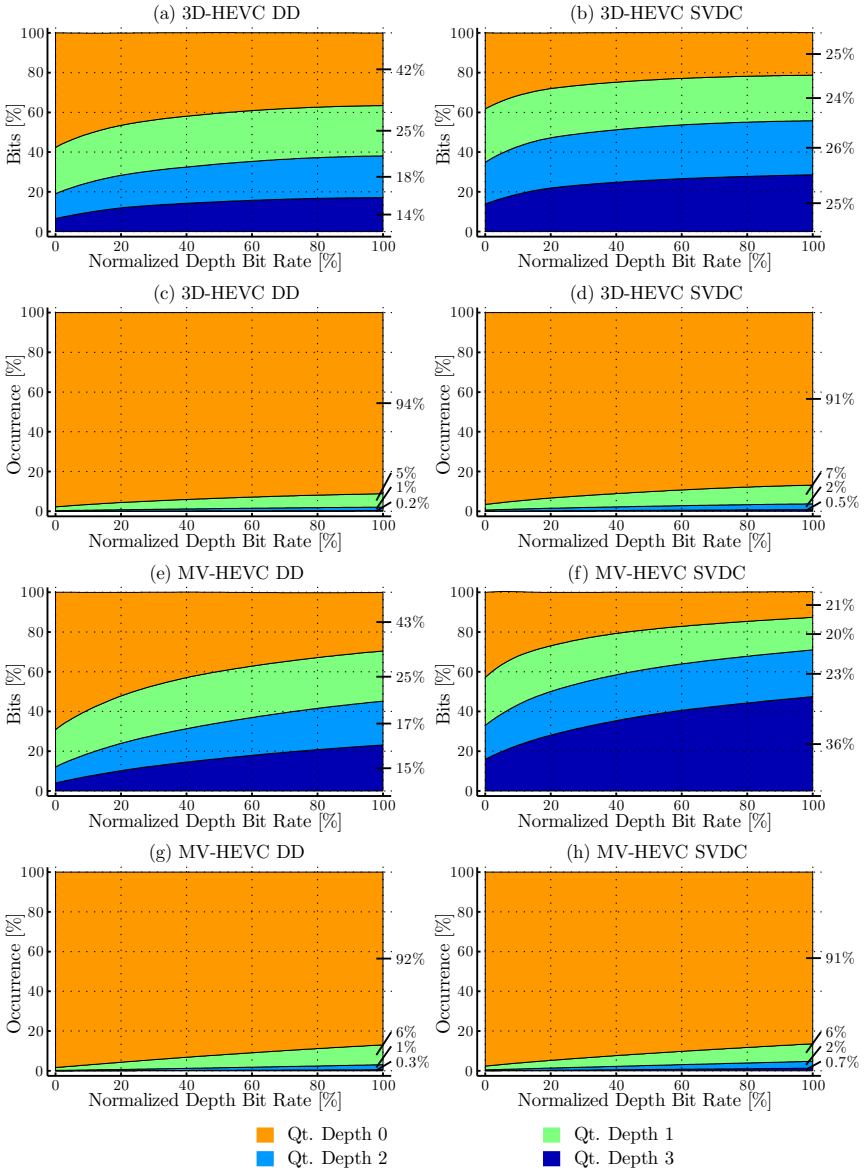


Figure F.25: Occurrence of and bit allocation to different quadtree depths.

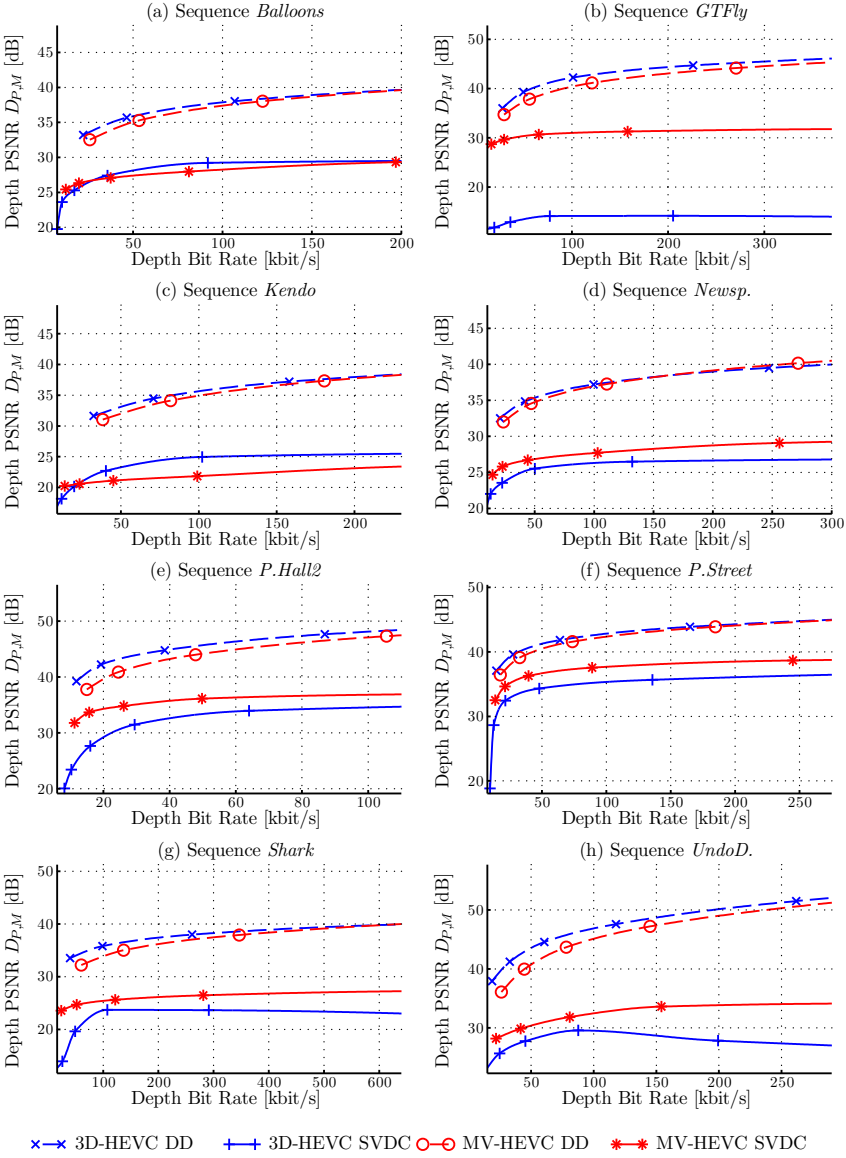


Figure F.26: Depth map PSNR $D_{P,M}$ in depth distortion-based and SVDC-based encoding for MV-HEVC and 3D-HEVC.

F.3 Renderer Model Setups

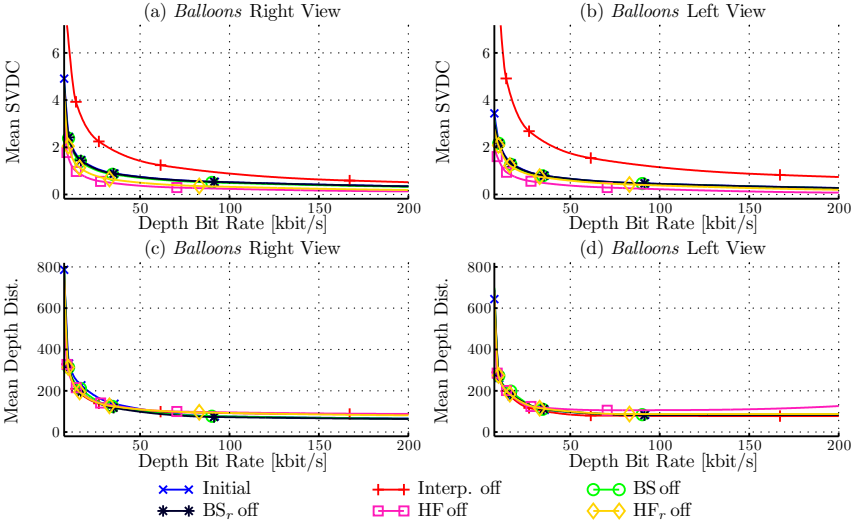


Figure F.27: Implications of the hiding effect; *Balloons*.

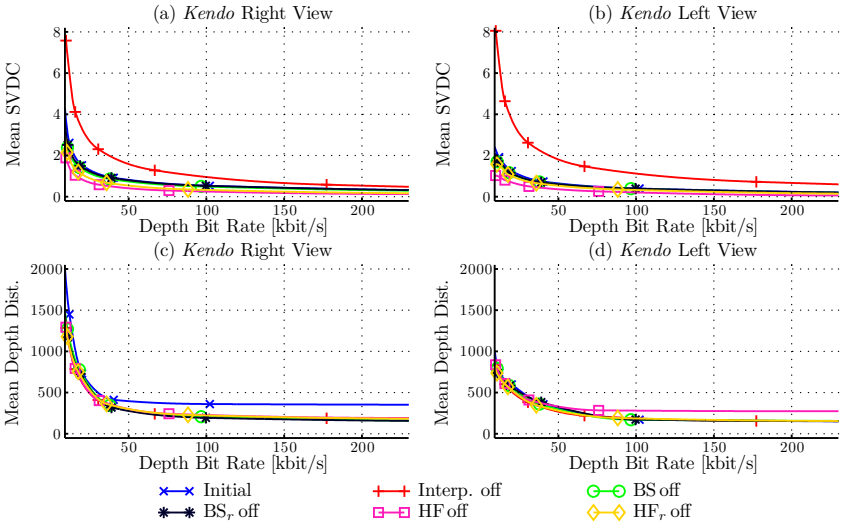


Figure F.28: Implications of the hiding effect; *Kendo*.

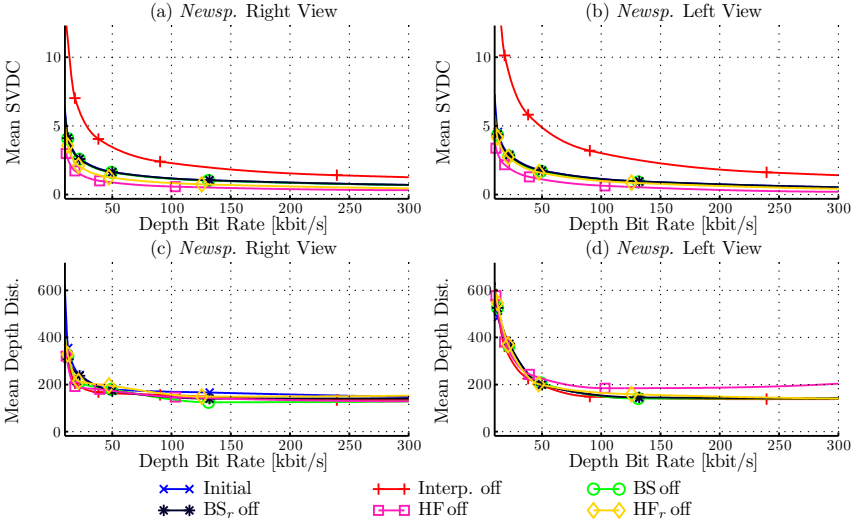


Figure F.29: Implications of the hiding effect; *Newsp*.

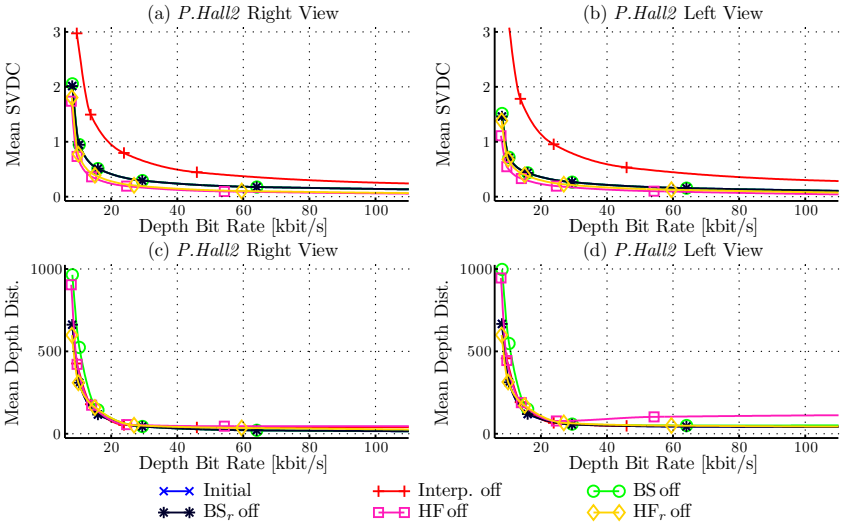


Figure F.30: Implications of the hiding effect; *P.Hall2*.

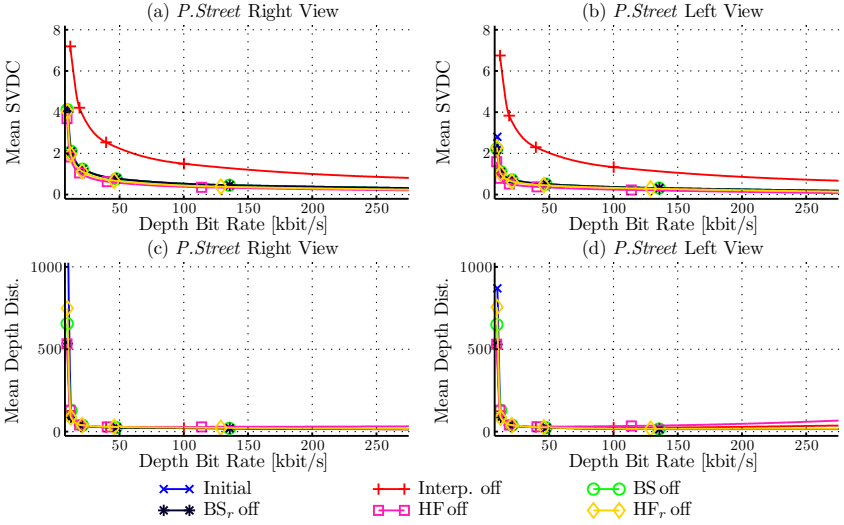


Figure F.31: Implications of the hiding effect; *P.Street*.

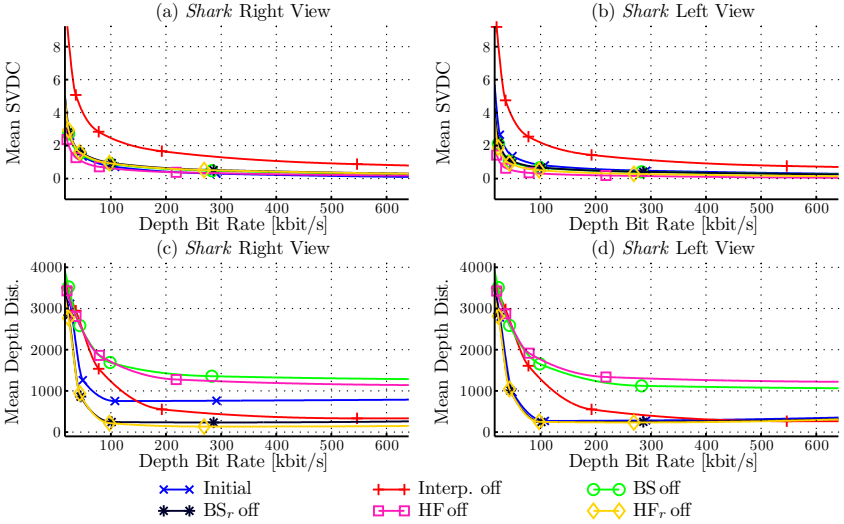


Figure F.32: Implications of the hiding effect; *Shark*.

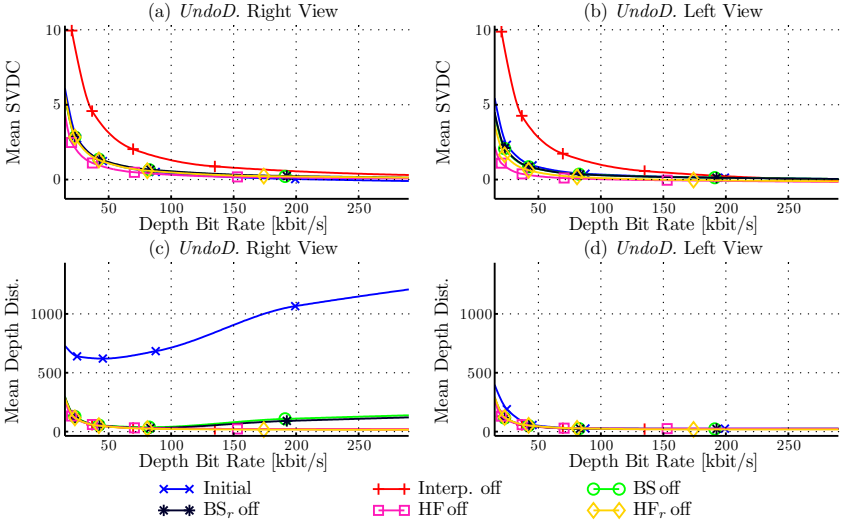


Figure F.33: Implications of the hiding effect; *UndoD*.

F.4 Synthesized View Generation Setups

	$R_\Delta[\%]$							
Right IV	$I(\boldsymbol{c}xoo)$			$I(\boldsymbol{o}xoo)$			$E(vx)$	
Left IV	$I(\boldsymbol{vv}vx)$	$I(\boldsymbol{cvc}x)$	$I(\boldsymbol{ccc}x)$	$I(\boldsymbol{vv}vx)$	$I(\boldsymbol{ov}ox)$	$I(\boldsymbol{ooo}x)$	$E(vx)$	$I(vvvx)$
Balloons	− 0.3	+ 1.7	+ 5.1	+ 1.0	+ 2.8	+ 18.8	+20.0	+18.6
Kendo	− 5.4	− 1.8	+ 3.0	− 0.5	+ 2.7	+ 56.7	+19.3	+ 9.3
Newsp.	− 0.5	+ 3.0	+ 6.9	+ 1.2	+ 3.9	+ 31.7	+35.7	+24.2
P.Hall2	− 0.9	+ 0.5	+ 1.7	+ 0.1	+ 1.5	+ 3.5	+57.2	+37.3
P.Street	− 1.5	+ 1.2	+ 3.6	− 0.9	+ 2.6	+ 12.8	+25.3	+19.6
GTFly	−23.5	−22.2	−19.7	−17.7	−18.1	+ 55.4	−38.2	−37.2
Shark	−12.5	−10.2	− 7.2	−11.5	−11.0	+126.0	−28.4	−29.1
UndoD.	− 9.4	− 8.9	− 6.9	− 6.9	− 6.6	+ 58.3	+32.3	+ 0.3
Mean	− 6.7	− 4.6	− 1.7	− 4.4	− 2.8	+ 45.4	+15.4	+ 5.4

Table F.2: Depth bit rate delta R_Δ; Tested: Different SV generation setups; Reference: $I(vxoo)$ $I(vvvx)$ setup; All: *Default* view combination.

	$R_\Delta[\%]$							
Right IV	$I(\boldsymbol{c}xoo)$			$I(\boldsymbol{o}xoo)$			$E(vx)$	
Left IV	$I(\boldsymbol{vv}vx)$	$I(\boldsymbol{cvc}x)$	$I(\boldsymbol{ccc}x)$	$I(\boldsymbol{vv}vx)$	$I(\boldsymbol{ov}ox)$	$I(\boldsymbol{ooo}x)$	$E(vx)$	$I(vvvx)$
Balloons	+0.8	+2.7	+ 7.8	+1.8	+3.6	+11.2	+18.2	+18.6
Kendo	+2.1	+5.8	+14.5	+3.2	+7.1	+22.4	+28.3	+18.7
Newsp.	+0.6	+4.4	+12.9	+1.3	+4.4	+14.6	+33.1	+22.3
P.Hall2	−1.9	−1.1	+ 1.9	−2.8	−1.8	+ 1.5	+21.2	+ 7.0
P.Street	+0.1	+2.9	+ 7.5	−0.1	+3.4	+ 9.6	+22.8	+17.9
GTFly	+1.6	+4.5	+ 9.2	−1.5	+0.5	+ 8.1	+10.6	+ 7.7
Shark	+5.0	+9.2	+15.3	+0.7	+2.6	+19.2	+12.3	+ 6.3
UndoD.	−1.4	−0.2	+ 6.4	−2.2	−1.2	+10.0	+52.0	+17.3
Mean	+0.9	+3.5	+ 9.5	+0.1	+2.3	+12.1	+24.8	+14.5

Table F.3: Depth bit rate delta R_Δ; Tested: Different SV generation setups; Reference: $I(vxoo)$ $I(vvvx)$ setup; All: *HF$_r$ off* view combination.

F.5 Depth Distortion Term

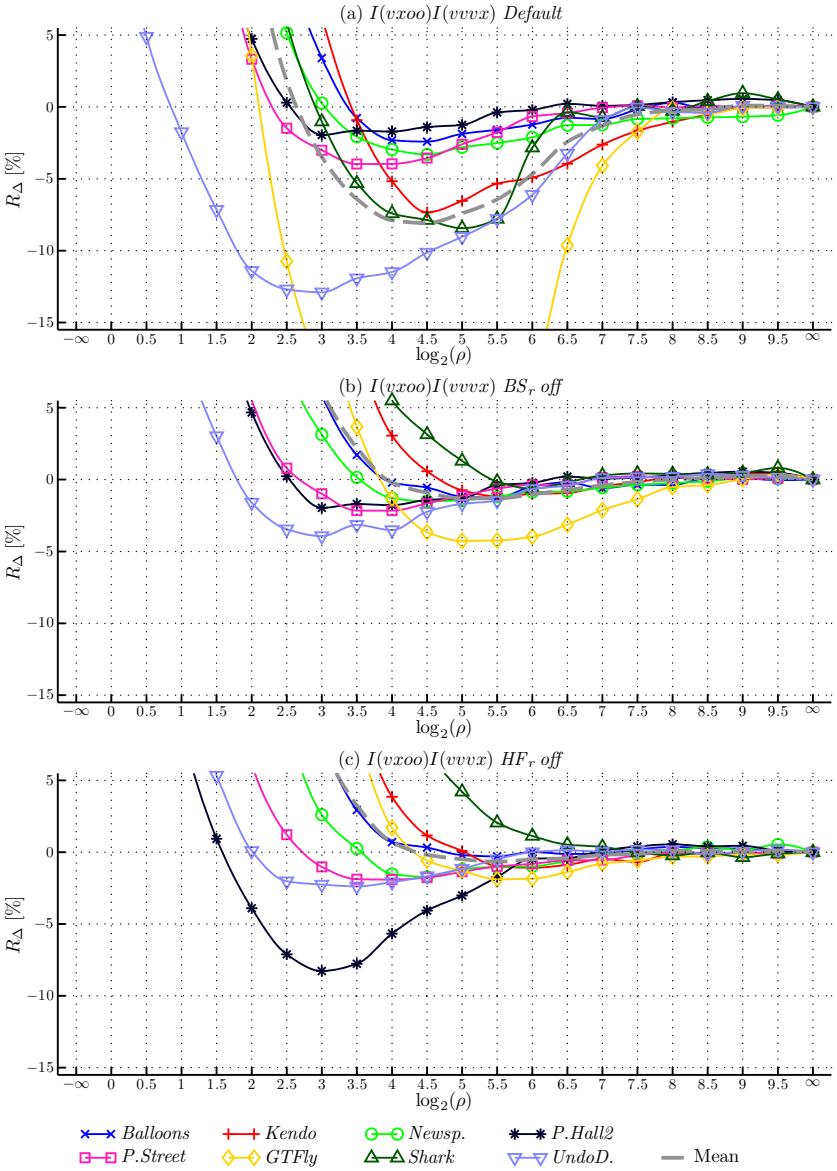


Figure F.34: R_Δ; Tested: View combinations variants and ratios ρ; Reference: The respective view combination variant with $\rho = \infty$; All: $I(vxoo)I(vvvx)$ SV generation setup.

F.6 Lagrange Multiplier and QP Selection

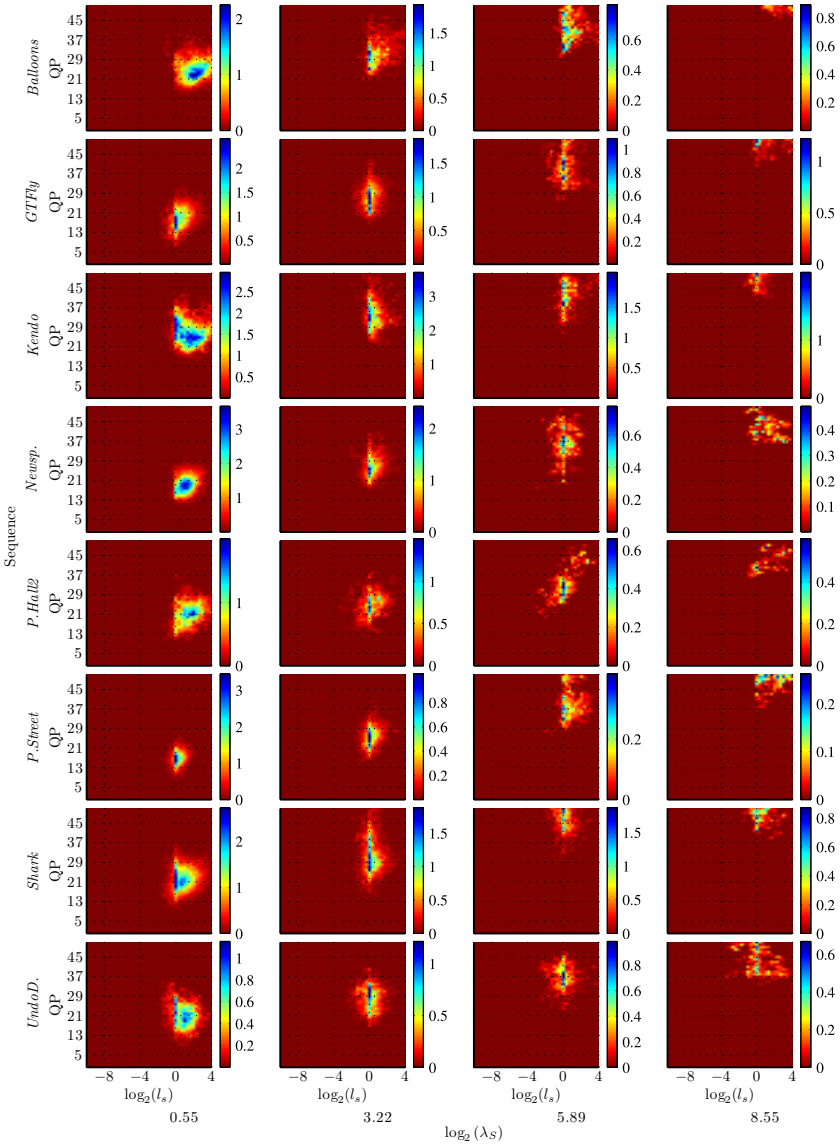


Figure F.35: Relative occurrence (in ‰) of CTBs coded with different l_s-QP combinations for different Lagrange multipliers λ_S; right view.

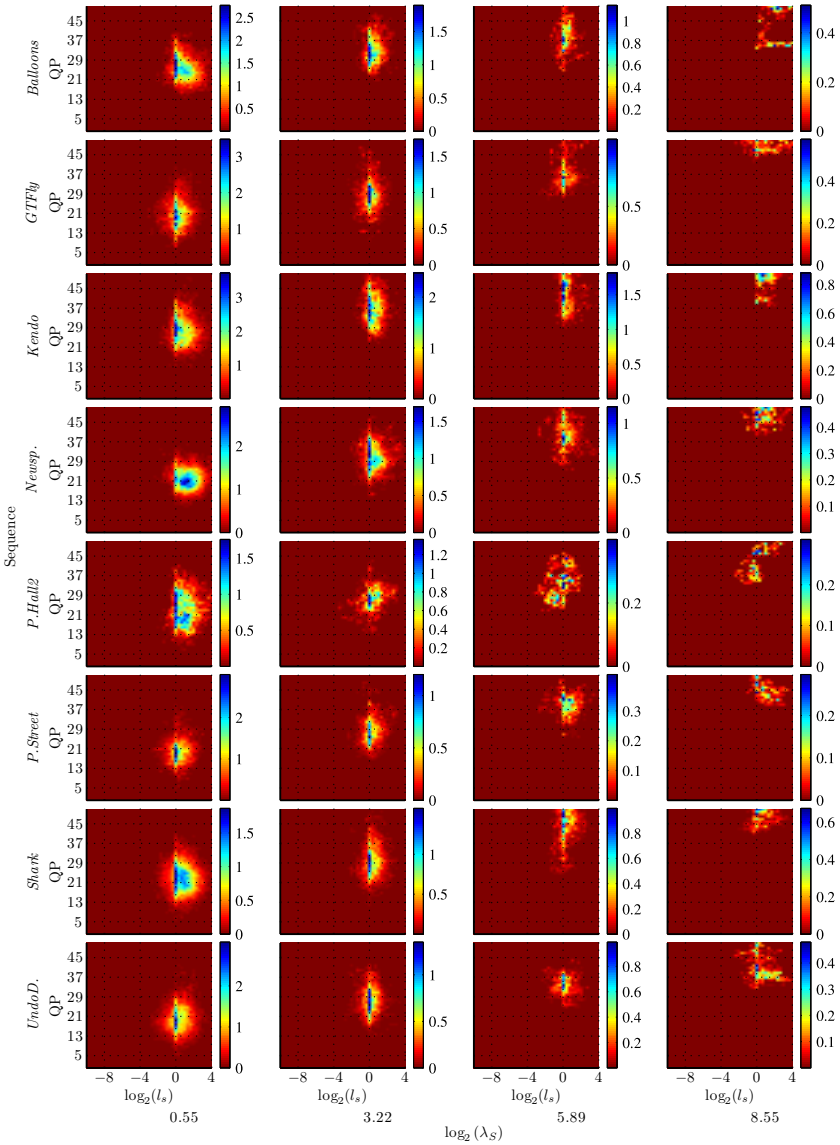


Figure F.36: Relative occurrence (in ‰) of CTBs coded with different l_s-QP combinations for different Lagrange multipliers λ_S; left view.

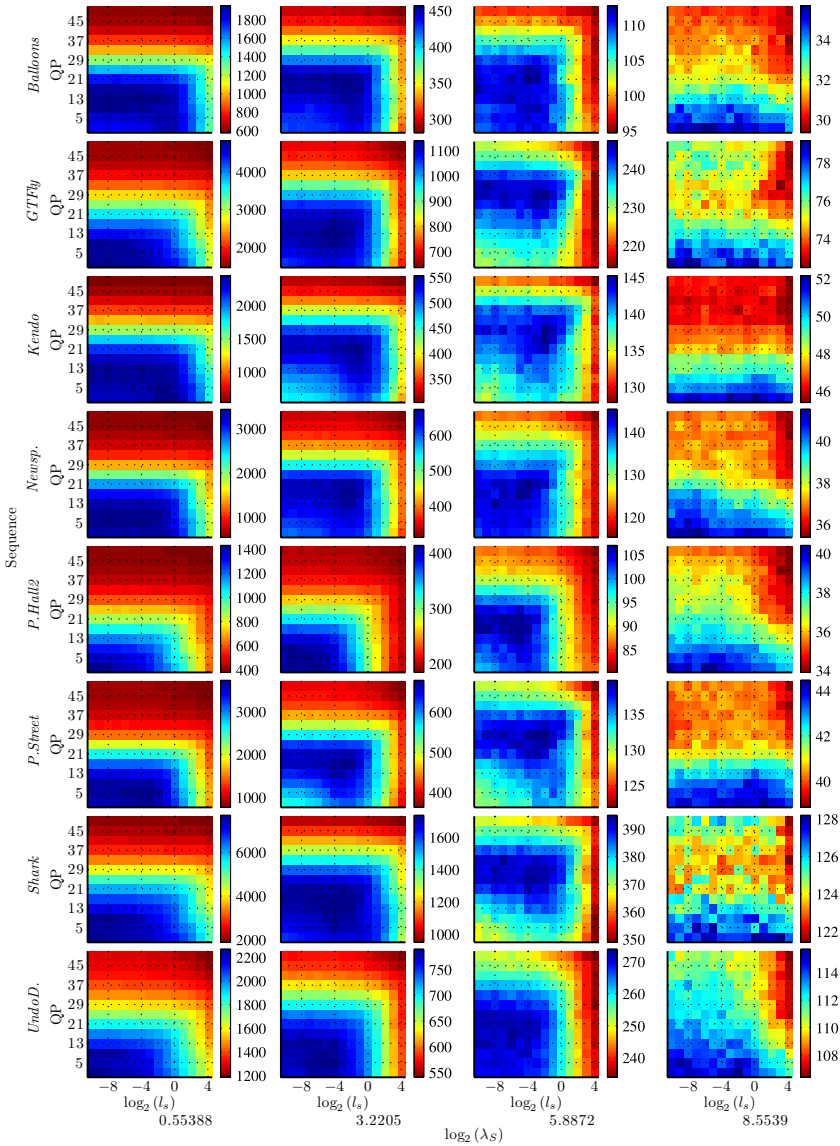


Figure F.37: Depth bit rate R [kbit/s] in dependence of λ_S, l_s, and the QP; Full Sequence Search.

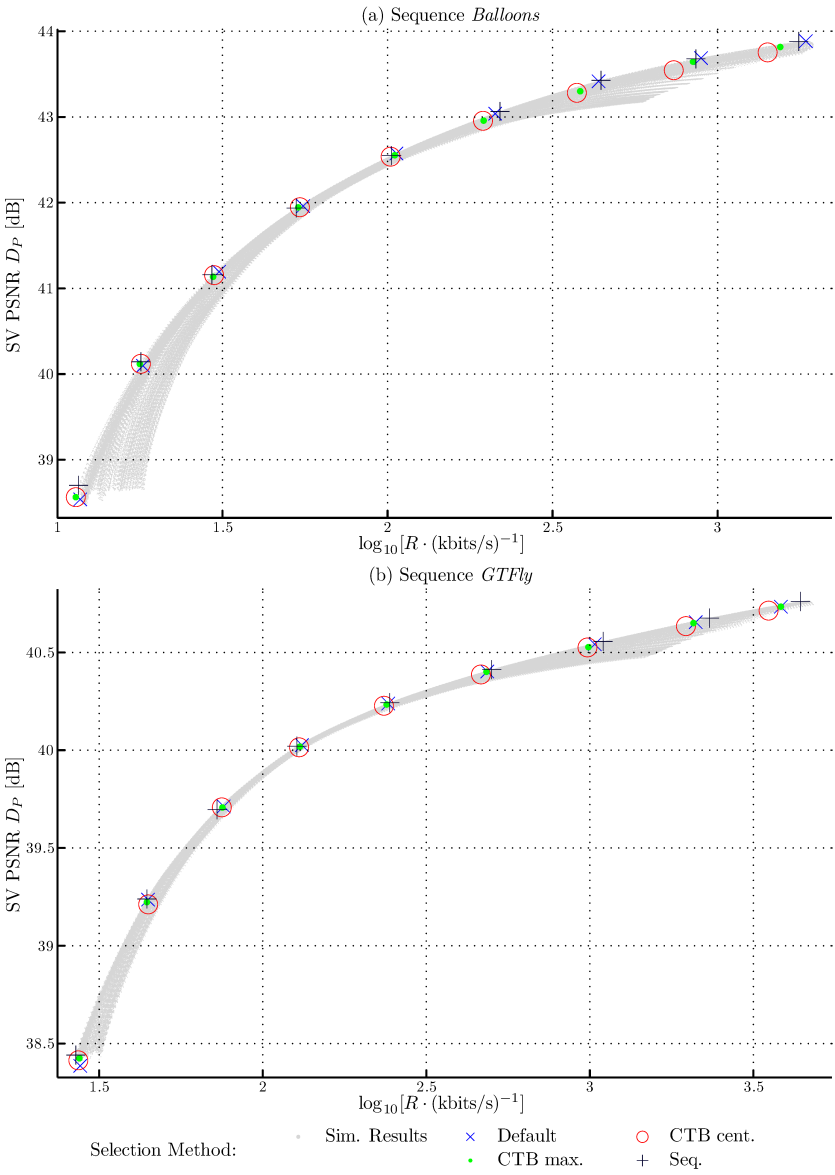


Figure F.38: RD performance for different λ_S, λ_M, QP combinations; Gray: Interpolated from full search.

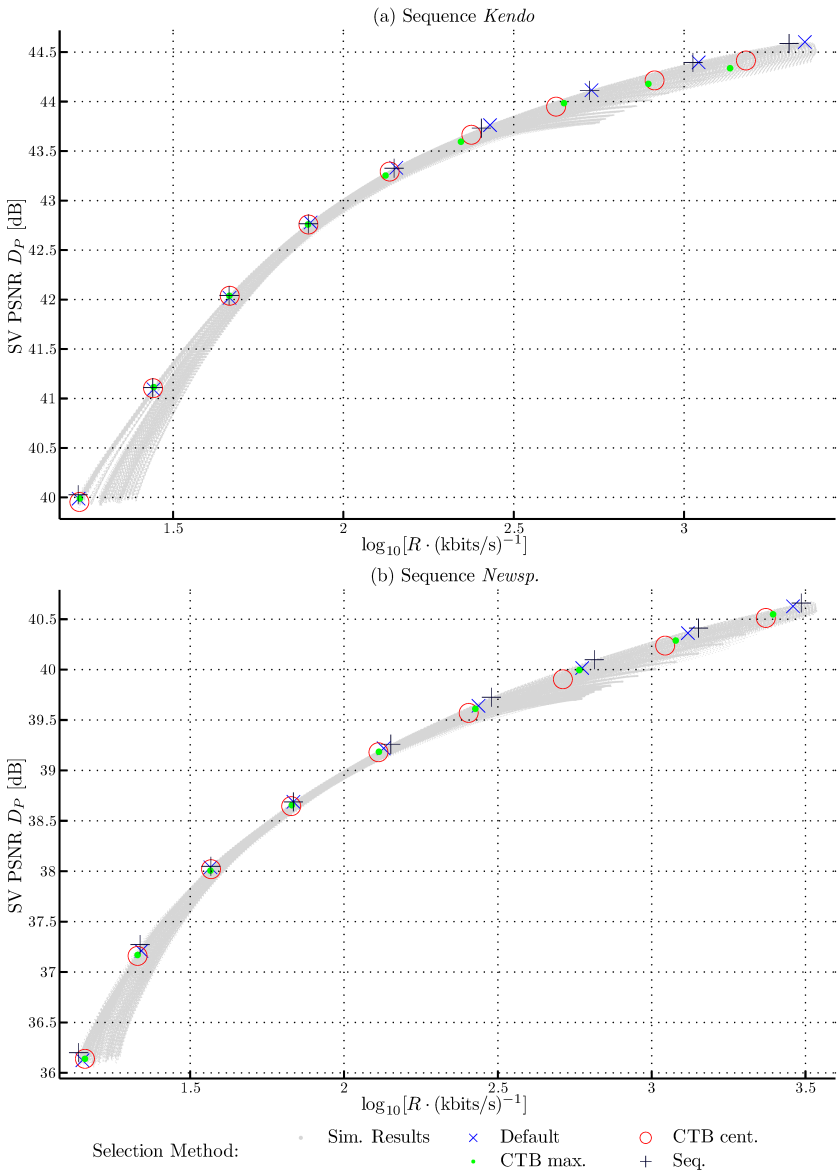


Figure F.39: RD performance for different λ_S, λ_M, QP combinations; Gray: Interpolated from full search.

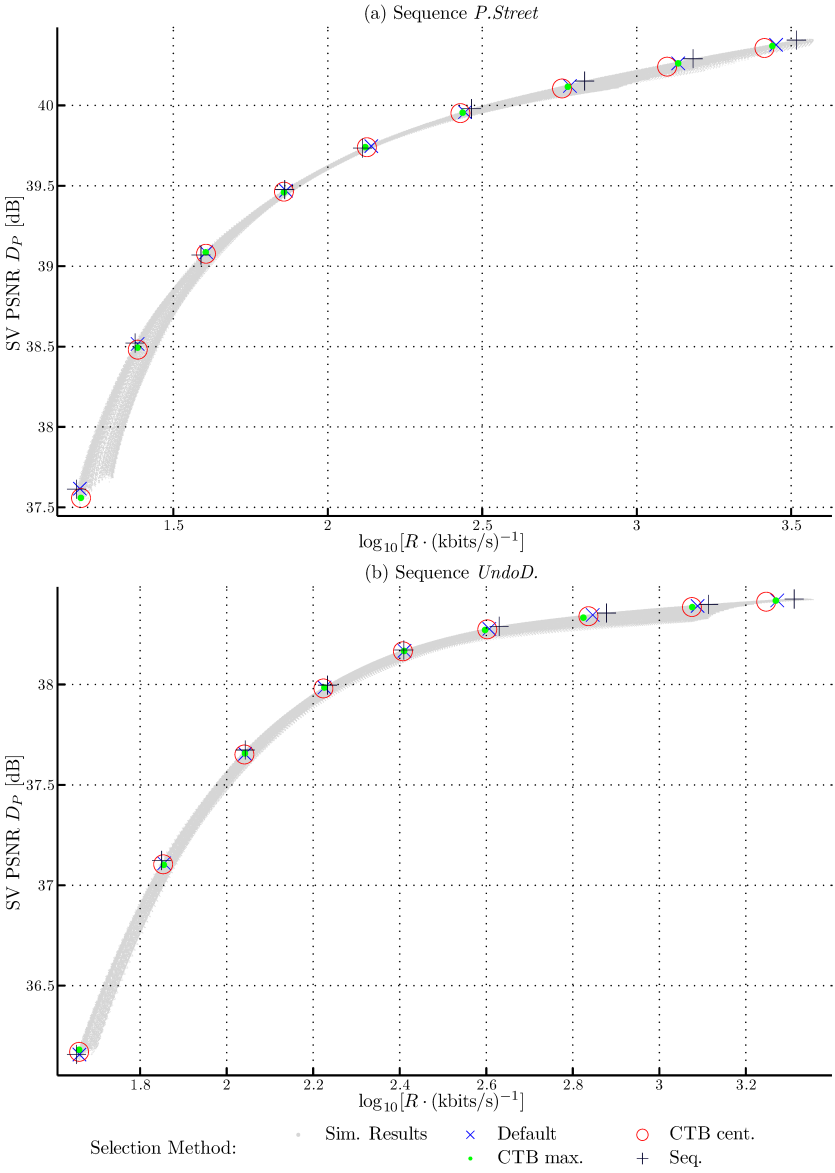


Figure F.40: RD performance for different λ_S, λ_M, QP combinations; Gray: Interpolated from full search.

F.7 Synthesized View Position Setups

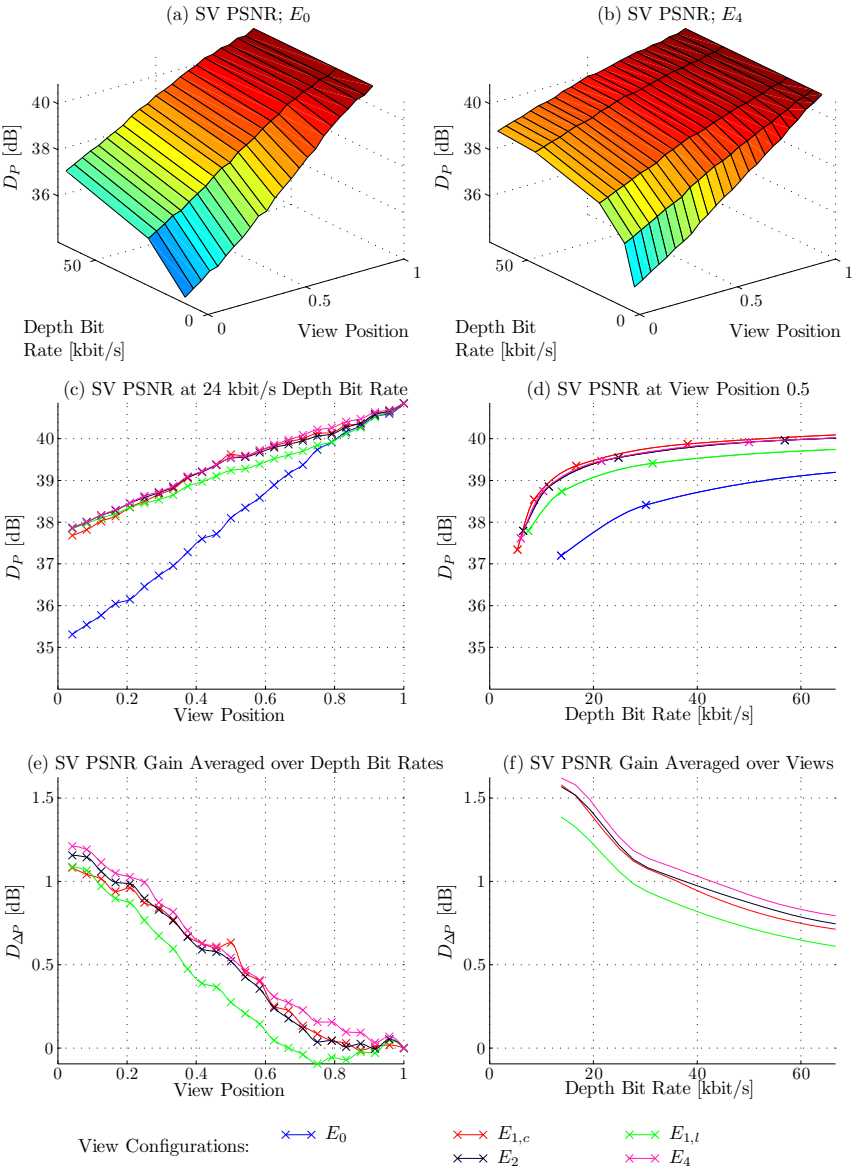


Figure F.41: *Balloons*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

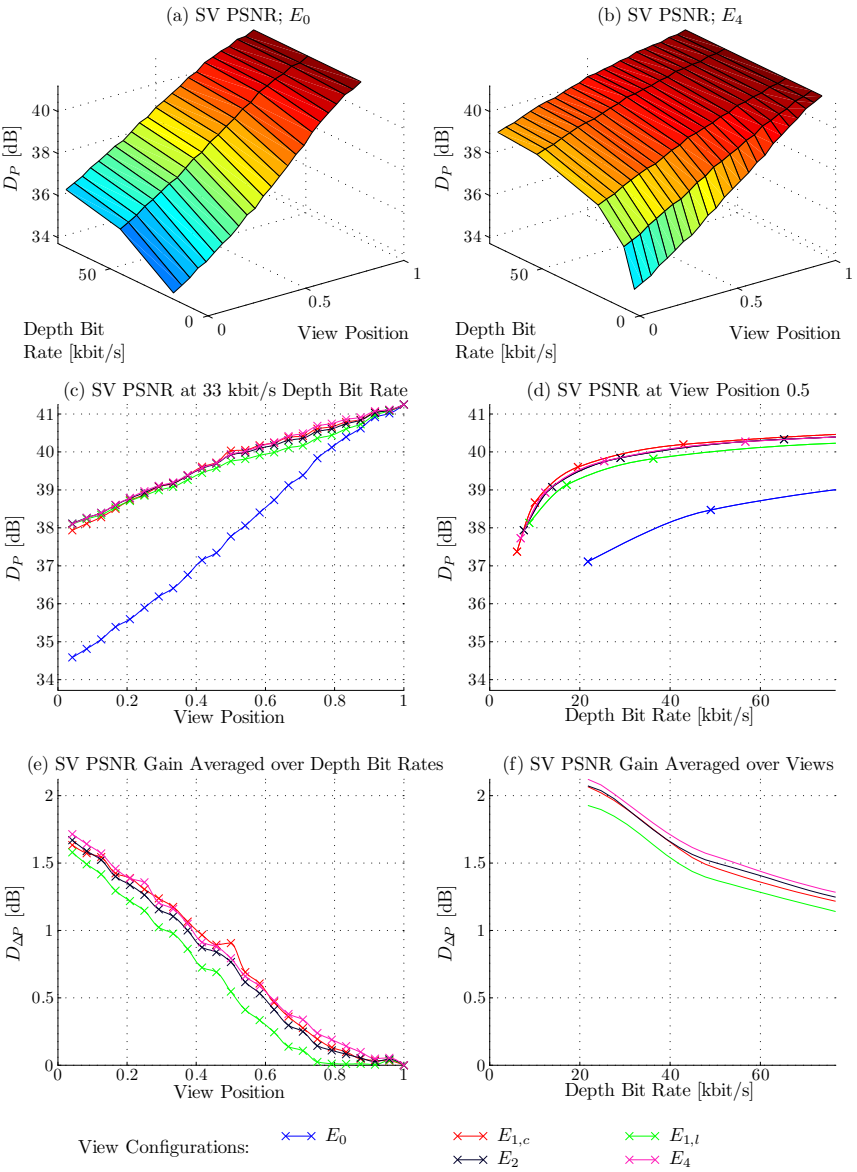


Figure F.42: *Kendo*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

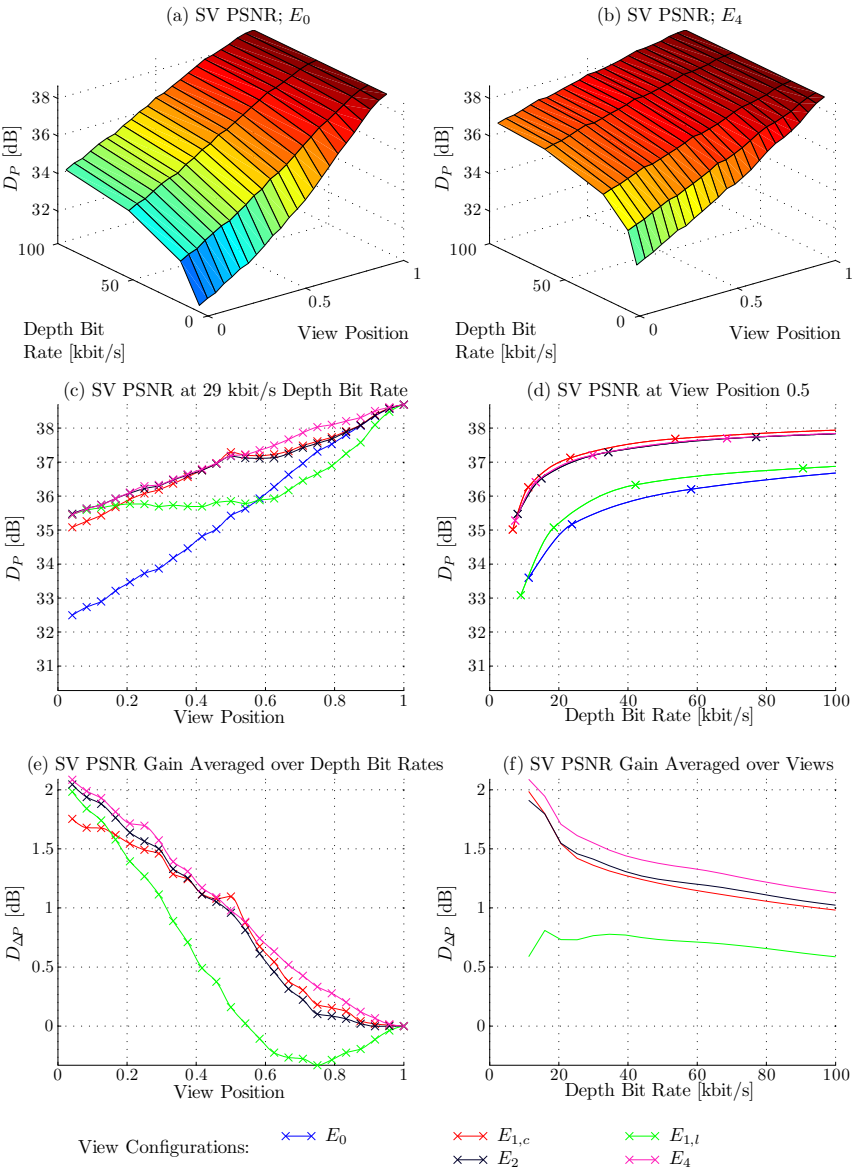


Figure F.43: *Newsp.*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

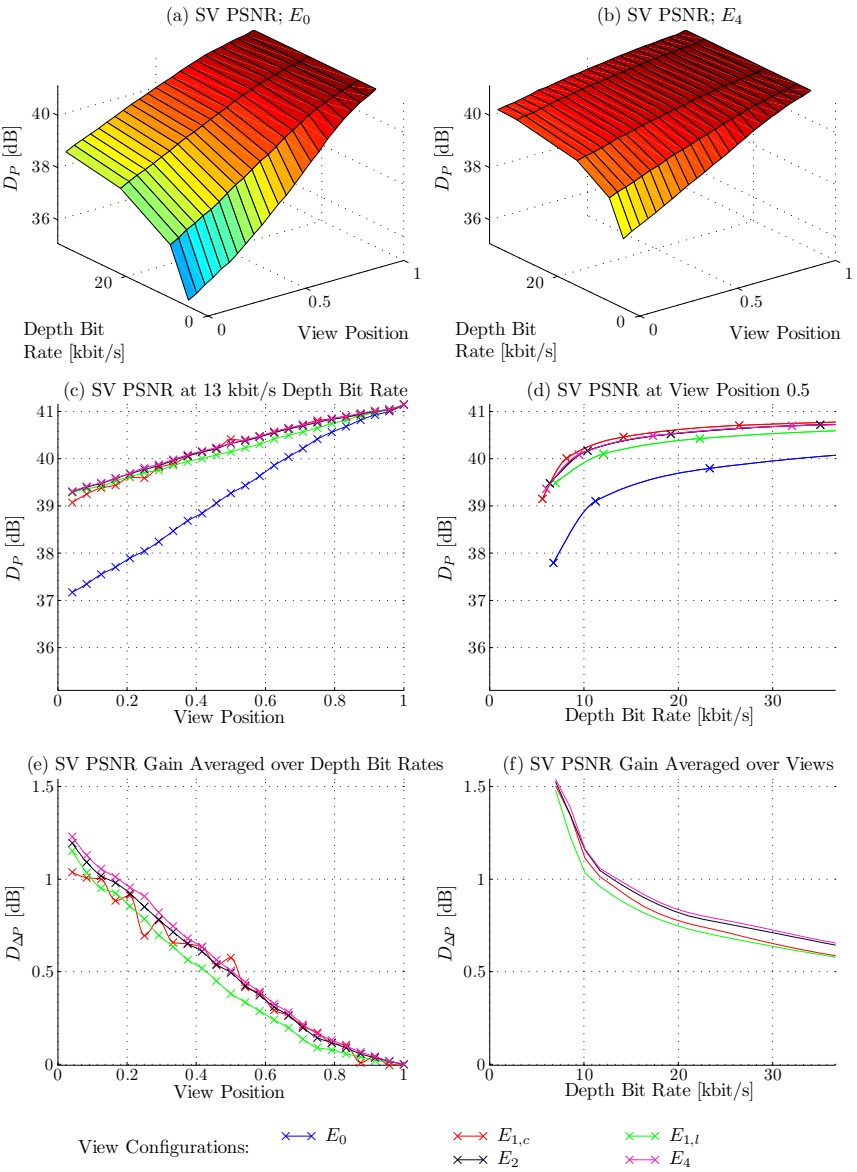


Figure F.44: *P.Hall2*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

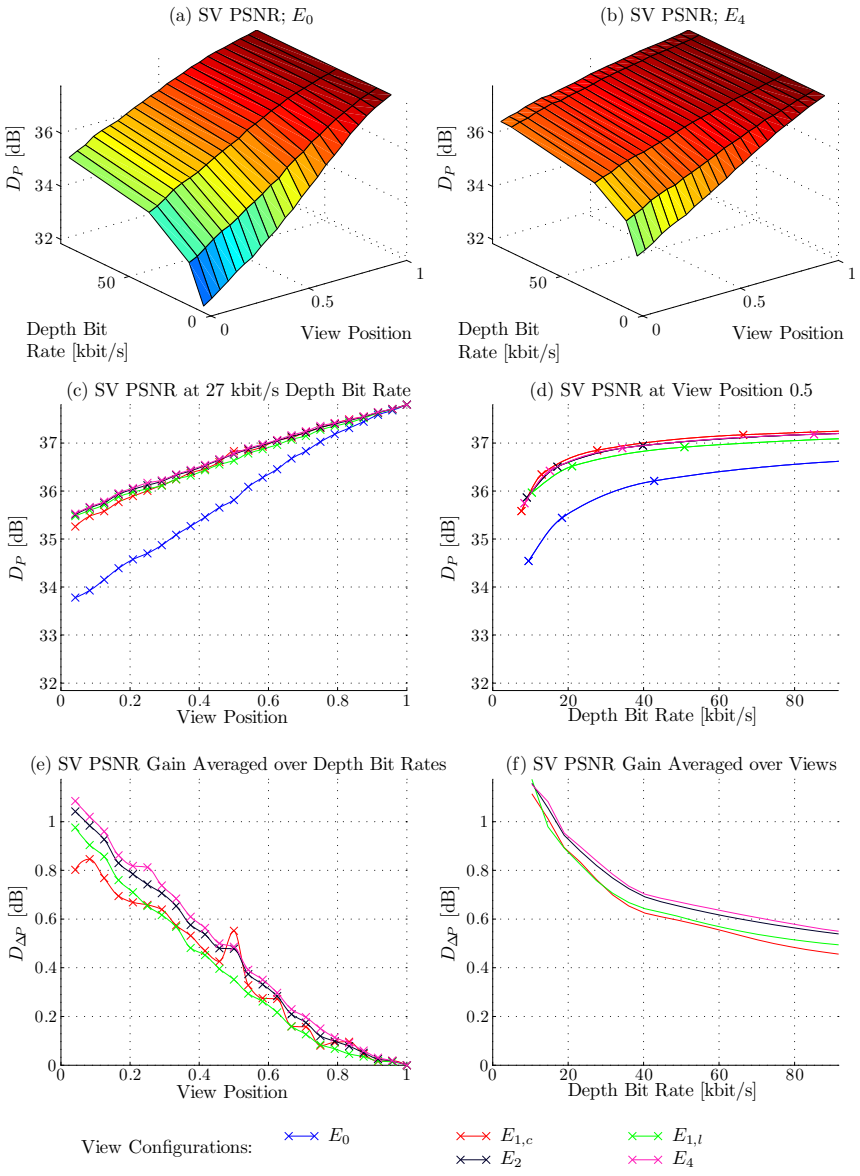


Figure F.45: *P.Street*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

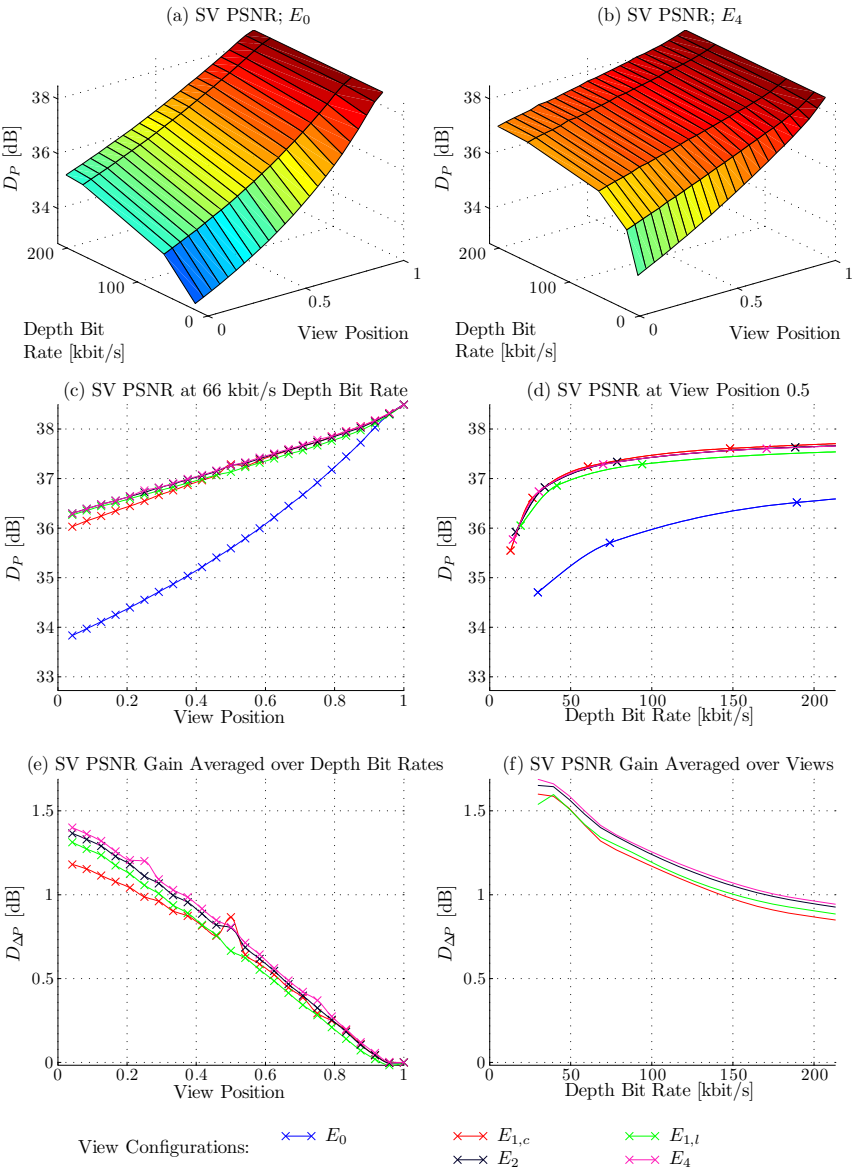


Figure F.46: *Shark*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

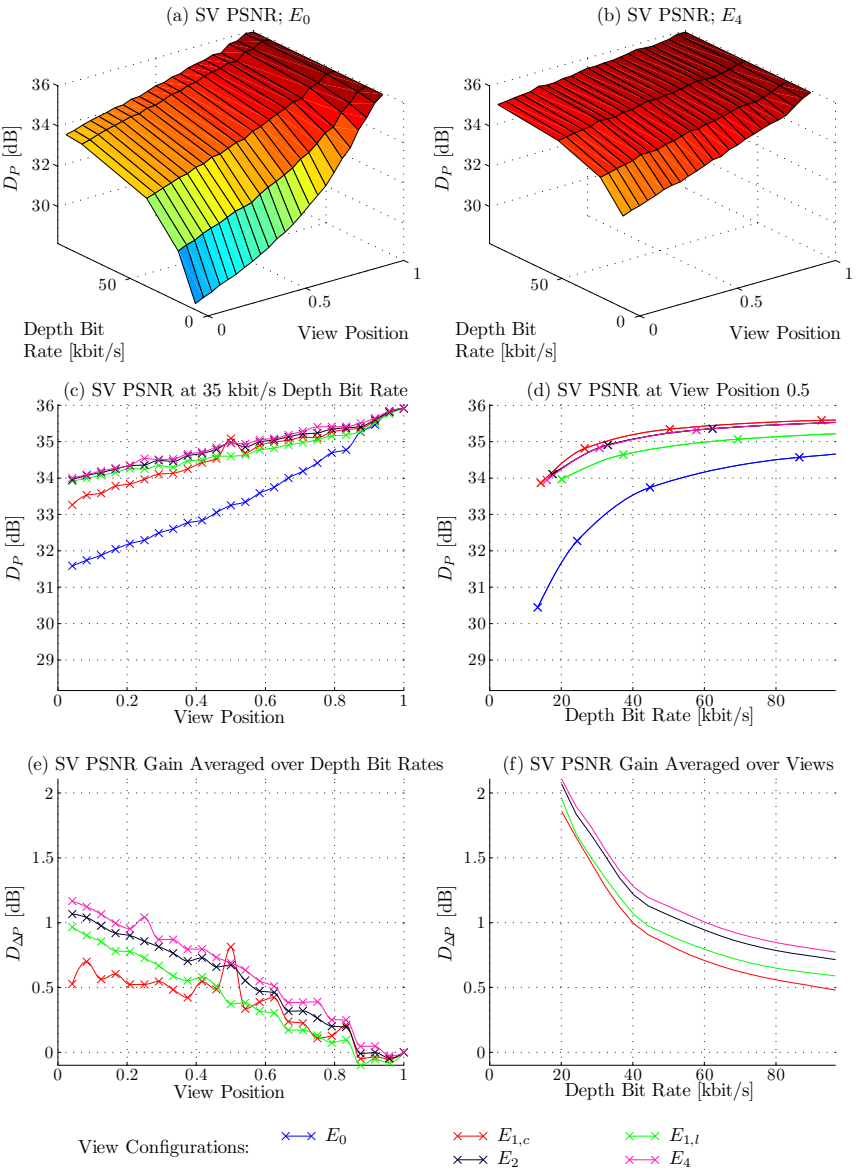


Figure F.47: *UndoD.*, view extrapolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

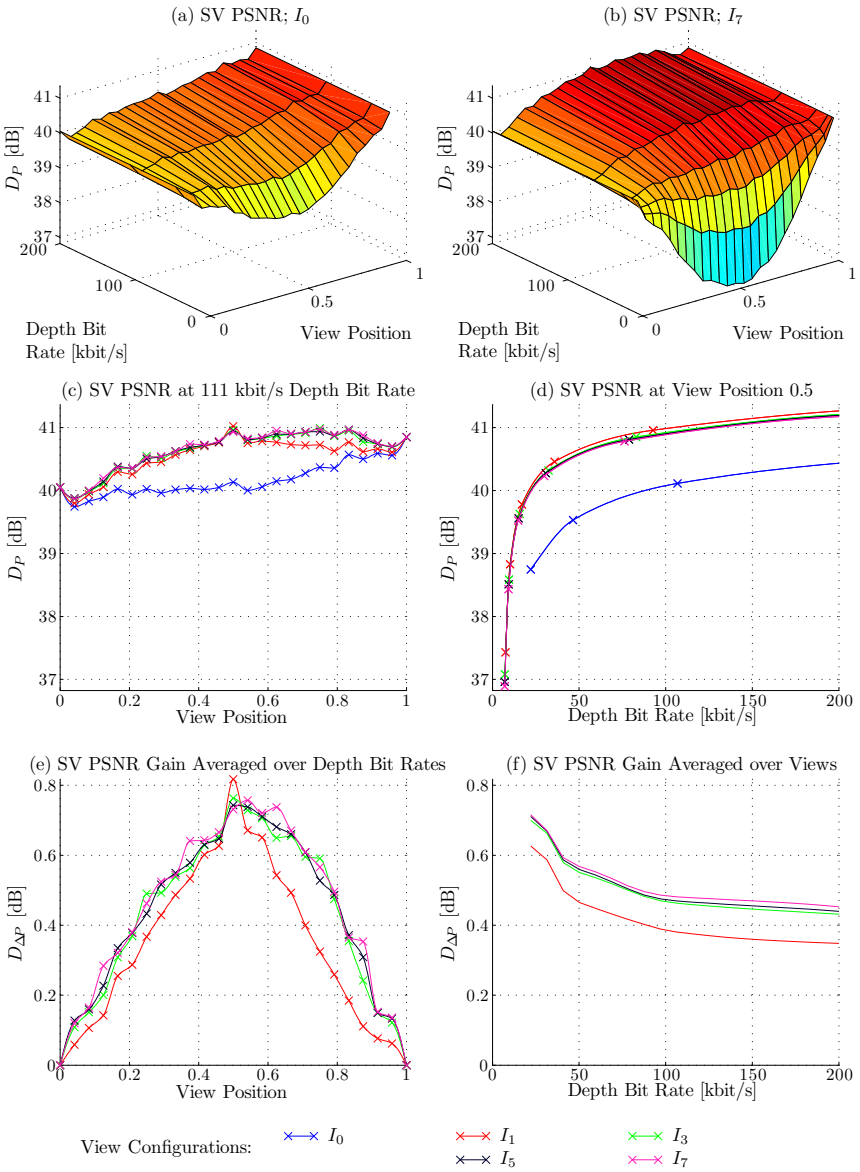


Figure F.48: *Balloons*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

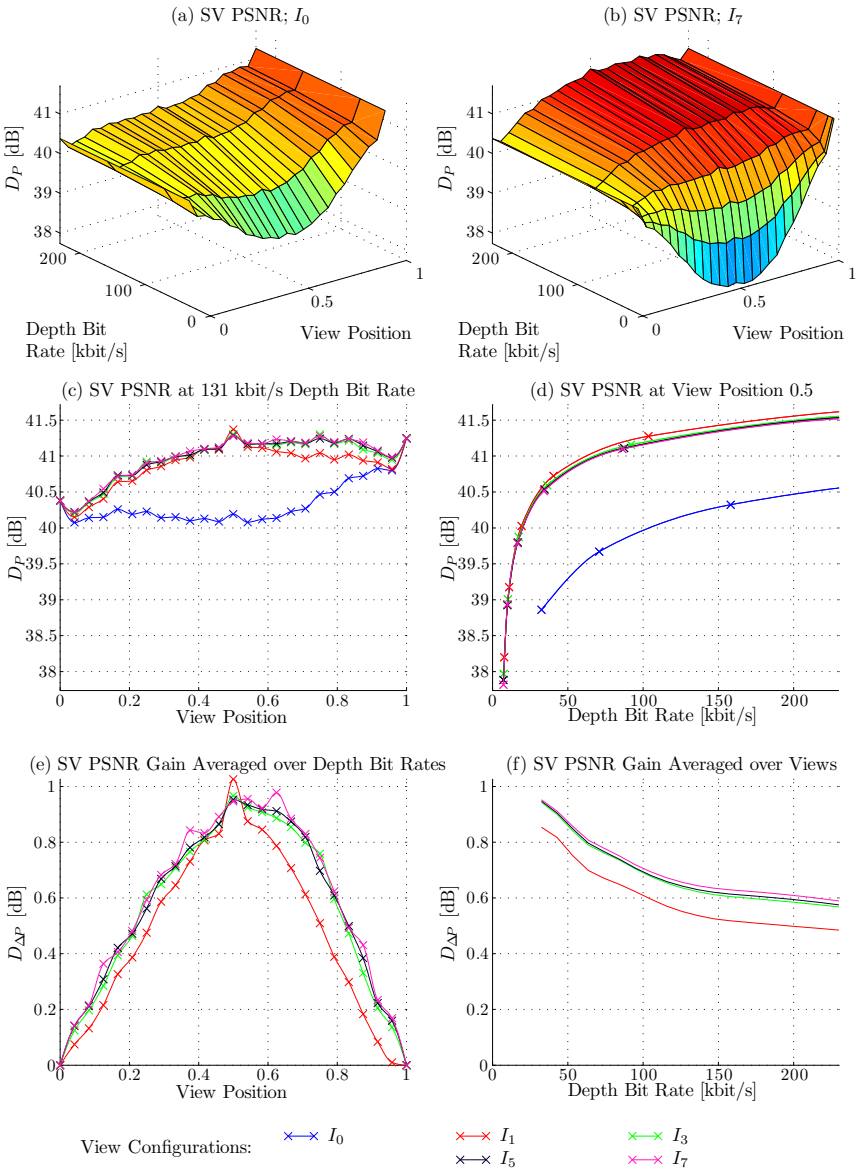


Figure F.49: *Kendo*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

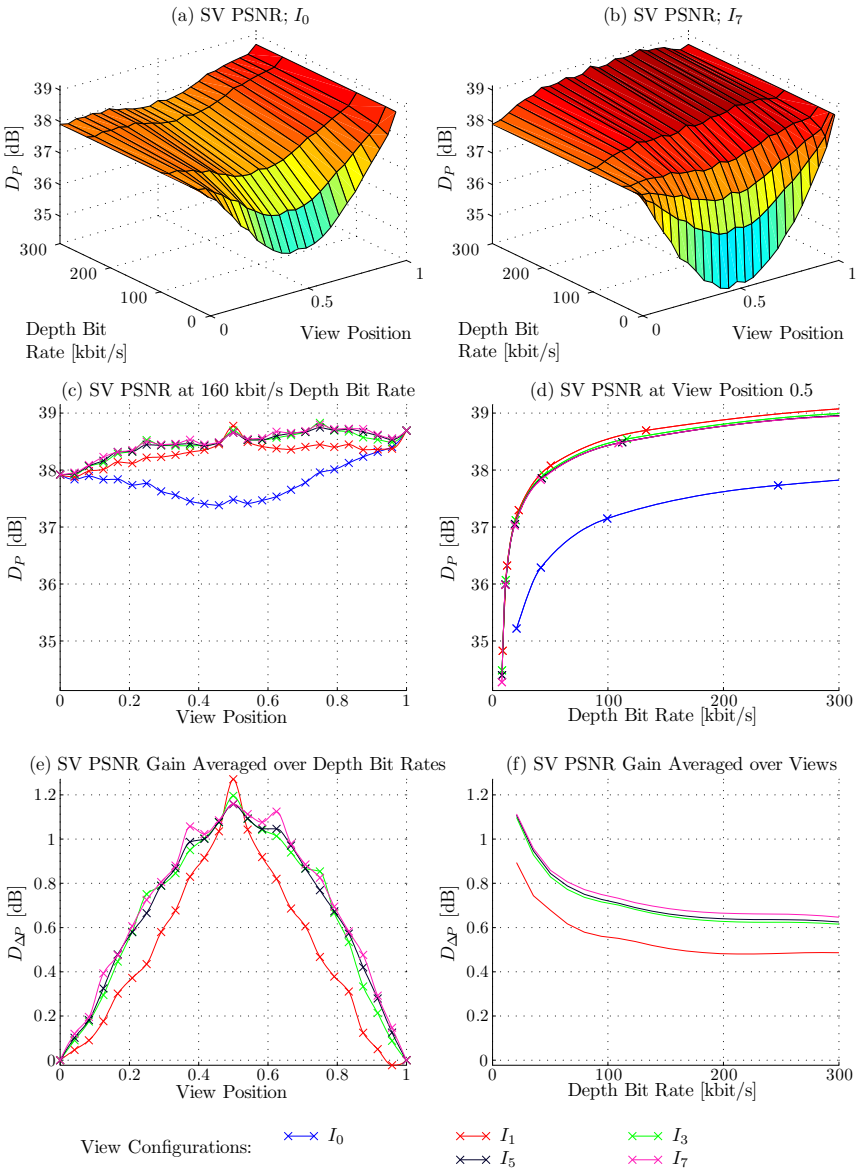


Figure F.50: *Newsp.*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

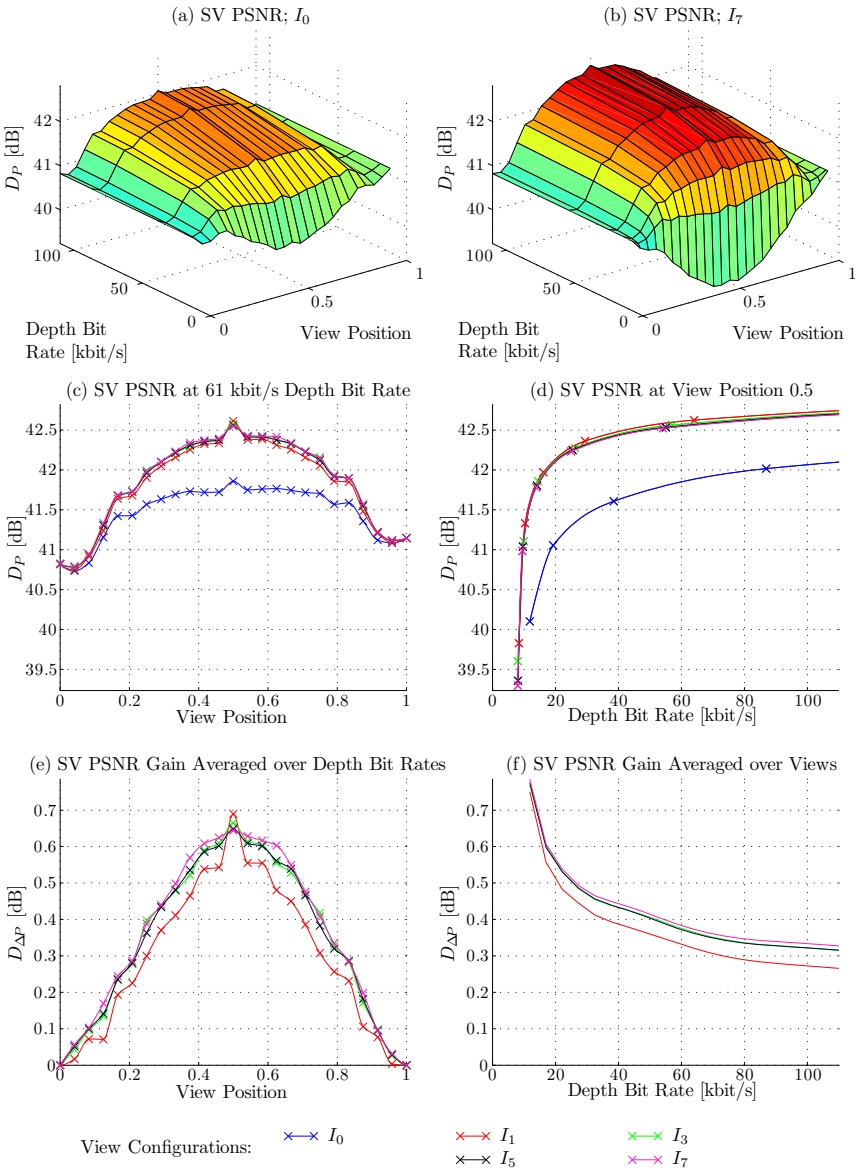


Figure F.51: *P.Hall2*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

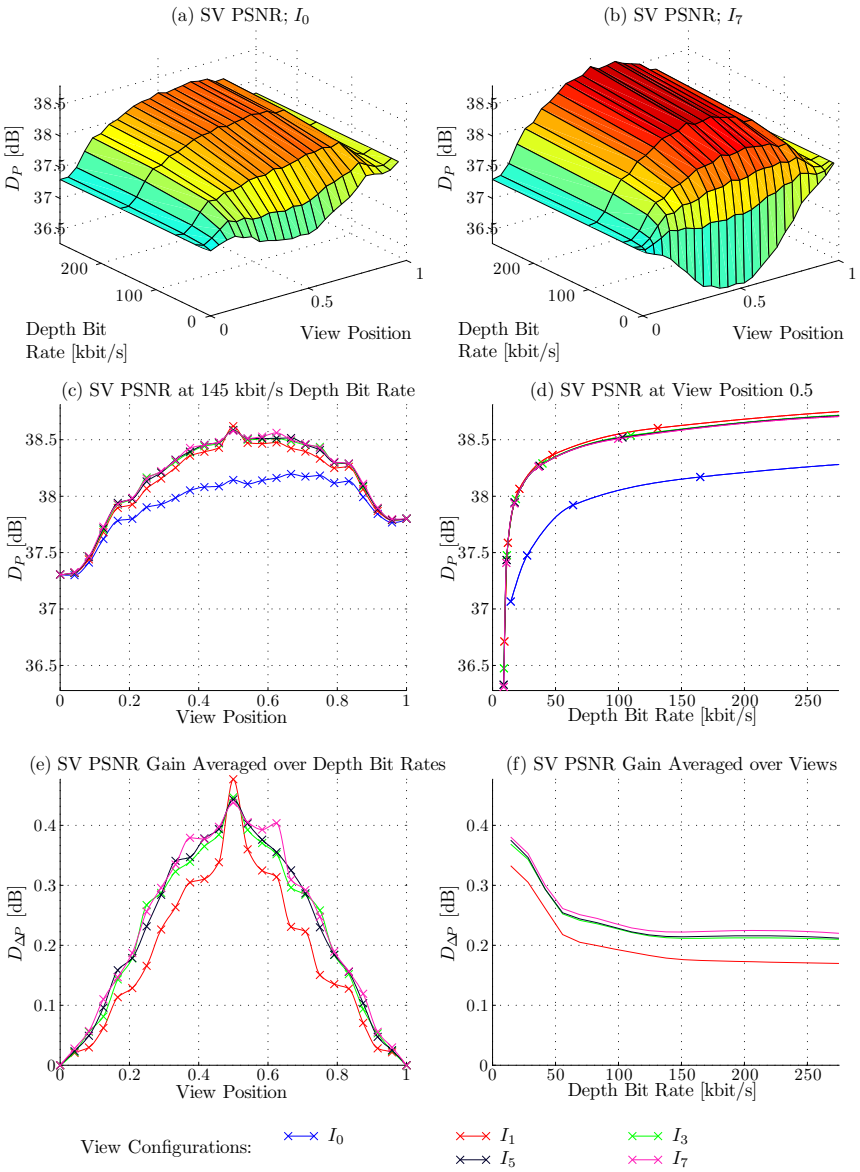


Figure F.52: *P.Street*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

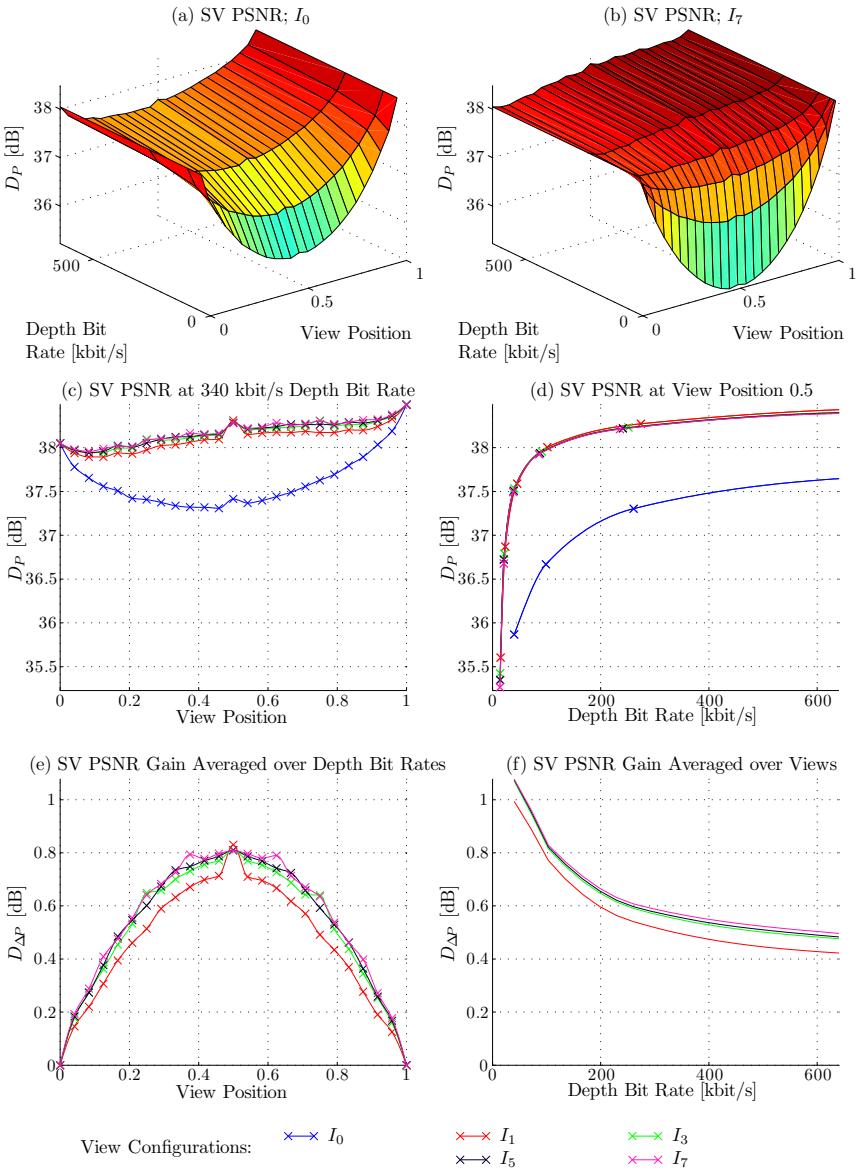


Figure F.53: *Shark*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

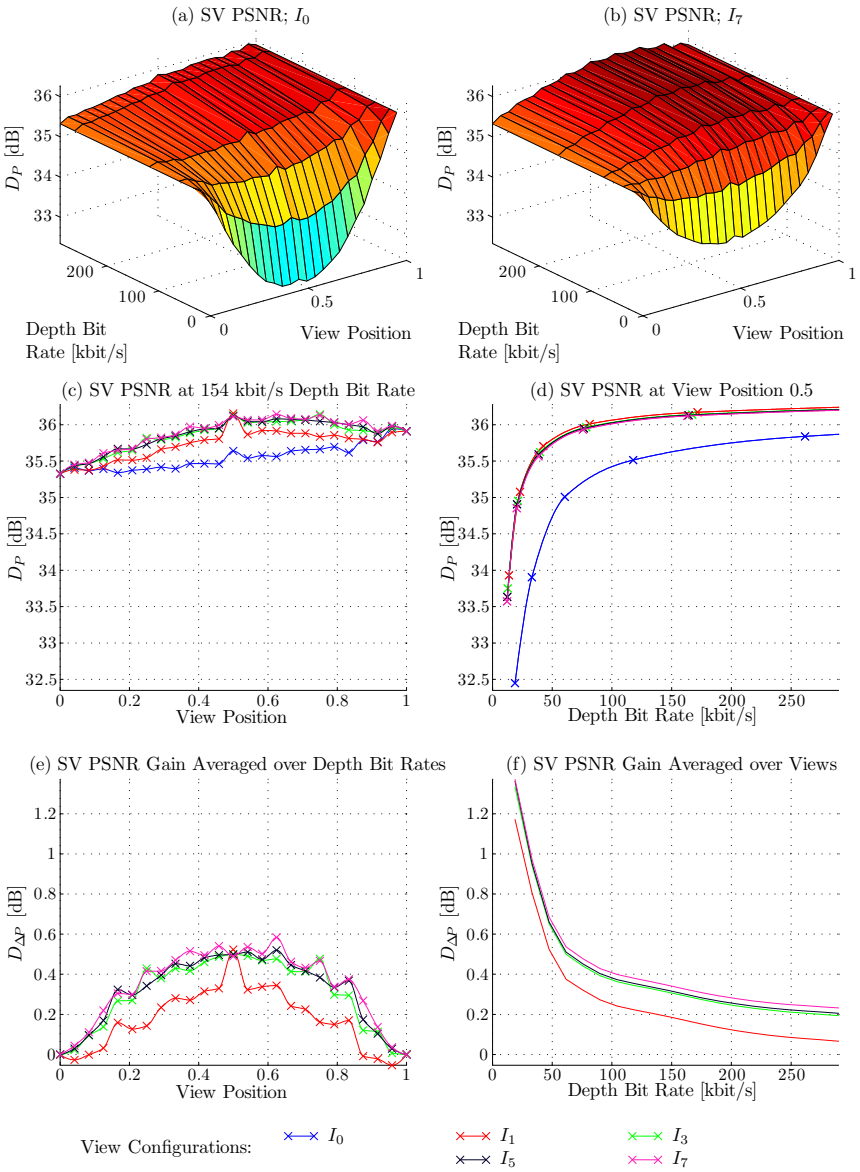


Figure F.54: *UndoD.*, view interpolation: (a) and (b) SV PSNR in dependence of SV position and depth bit rate; (c) and (d) at fixed depth bit rate and SV position; (e) and (f) gains averaged over depth bit rates and SV positions.

F.8 Alternative View Synthesis Algorithms

	$R_\Delta[\%]$											
	LCR			LCR (BNR)			VSRS (1D)			VSRS (Gen.)		
	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.	Ext.	*BS off*	Opt.
$\log_2(\rho)$	3.5	5.0	4.5	3.5	5.0	4.5	3.5	5.0	4.5	3.5	5.0	4.5
Balloons	+24.7	+0.4	+0.2	+22.1	+0.5	−0.1	− 3.3	−1.4	−1.3	− 8.0	− 6.0	−2.8
Kendo	+31.1	+4.3	+1.9	+28.4	+3.2	+1.3	+ 3.2	−2.7	−0.6	−17.3	−20.0	−7.0
Newsp.	+32.5	+1.2	−0.1	+32.8	+1.0	−1.0	− 1.6	−5.5	−5.4	0.0	−10.0	−7.2
P.Hall2	+44.5	−0.6	−0.8	+31.6	−0.8	−1.0	+20.5	−0.7	−0.9	+23.5	− 1.0	−0.8
P.Street	+23.8	−0.2	−0.8	+25.0	−0.4	−0.8	+ 7.8	−0.9	−1.6	+ 8.1	− 1.0	−1.4
GTFly	+ 5.6	+1.0	+2.5	+ 4.3	+1.0	+2.2	+ 4.8	−3.4	−2.5	−18.4	− 4.0	+1.2
Shark	+ 5.5	+2.1	+3.7	+ 4.6	+1.7	+3.1	−14.3	−3.1	0.0	−25.2	− 9.4	+0.4
UndoD.	+33.8	−0.2	−1.0	+32.9	−0.5	−1.4	+10.2	−2.7	−3.6	−22.3	−15.3	−3.4
Mean	+25.2	+1.0	+0.7	+22.7	+0.7	+0.3	+ 3.4	−2.6	−2.0	− 7.4	− 8.3	−2.6

Table F.4: Depth bit rate delta R_Δ; Tested: Different view synthesis methods and encoder setups using the same value for ρ ; Reference: Depth distortion-based encoding.

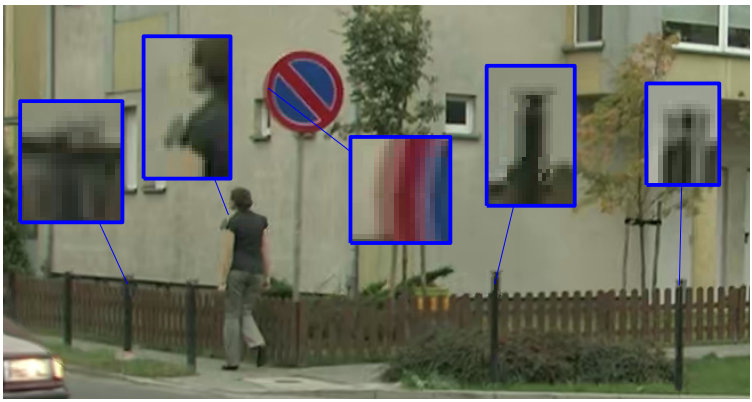
F.9 Comparison with Estimation Methods

	$R_\Delta[\%]$				$T_\Delta[\%]$			
Ref.	DD	VSD *C+W*	SH *C+W*	LS *R+W*	DD	VSD *C+W*	SH *C+W*	LS *R+W*
Balloons	−78.2	−52.6	−45.5	−30.9	+ 78	−66	−39	−27
Kendo	−83.8	−52.7	−56.7	−45.2	+ 57	−70	−43	−27
Newsp.	−81.1	−66.2	−50.3	−43.6	+ 83	−69	−42	−31
P.Hall2	−65.8	−43.9	−31.2	−32.4	+ 98	−57	−30	−18
P.Street	−73.9	−59.4	−37.3	−37.3	+101	−62	−34	−23
GTFly	−85.0	−83.1	−71.5	−67.3	+ 88	−69	−40	−29
Shark	−86.2	−72.7	−50.1	−51.5	+ 71	−69	−44	−33
UndoD.	−68.6	−72.2	−46.9	−51.7	+ 80	−74	−50	−40
Mean	−77.8	−62.9	−48.7	−45.0	+ 82	−67	−40	−29

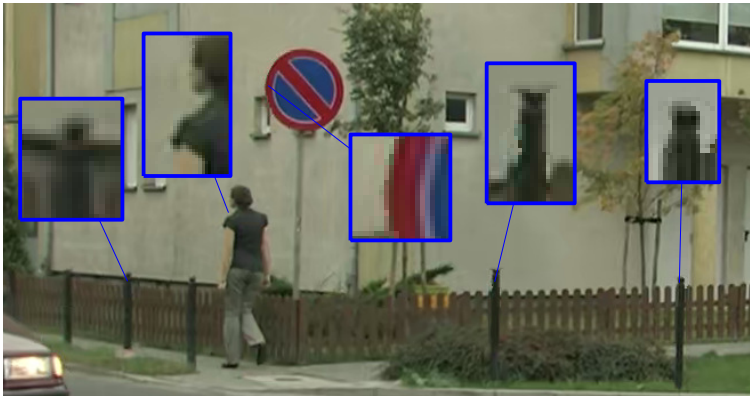
Table F.5: Depth bit rate delta R_Δ and encoding time delta T_Δ; Tested: *optimized encoder setup*; Reference: Optimized setups of different distortion derivation methods; Selection stage setup G4.

F.10 Sample Pictures

Depth Dist.
PSNR 37.7 dB
MSSIM 0.944
FOM 0.946
5640 bits



SVDC
PSNR 38.5 dB
MSSIM 0.946
FOM 0.947
5488 bits



Uncoded

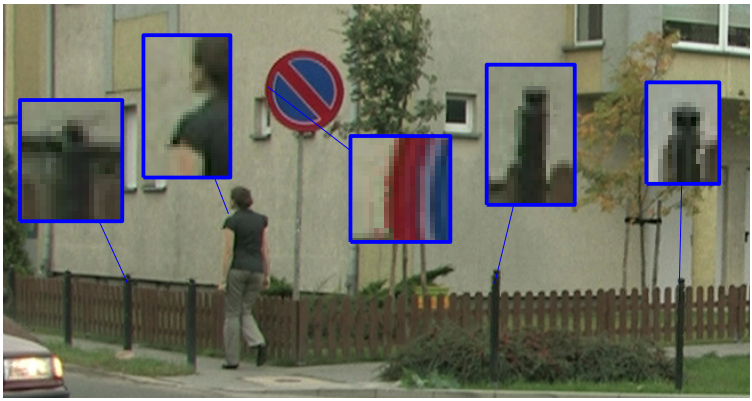


Figure F.55: SVDC- vs. depth distortion-based encoding; Sequence: *P.Street*.

Figure F.56: SVDC- vs. depth distortion-based encoding; Sequence: UndoD.

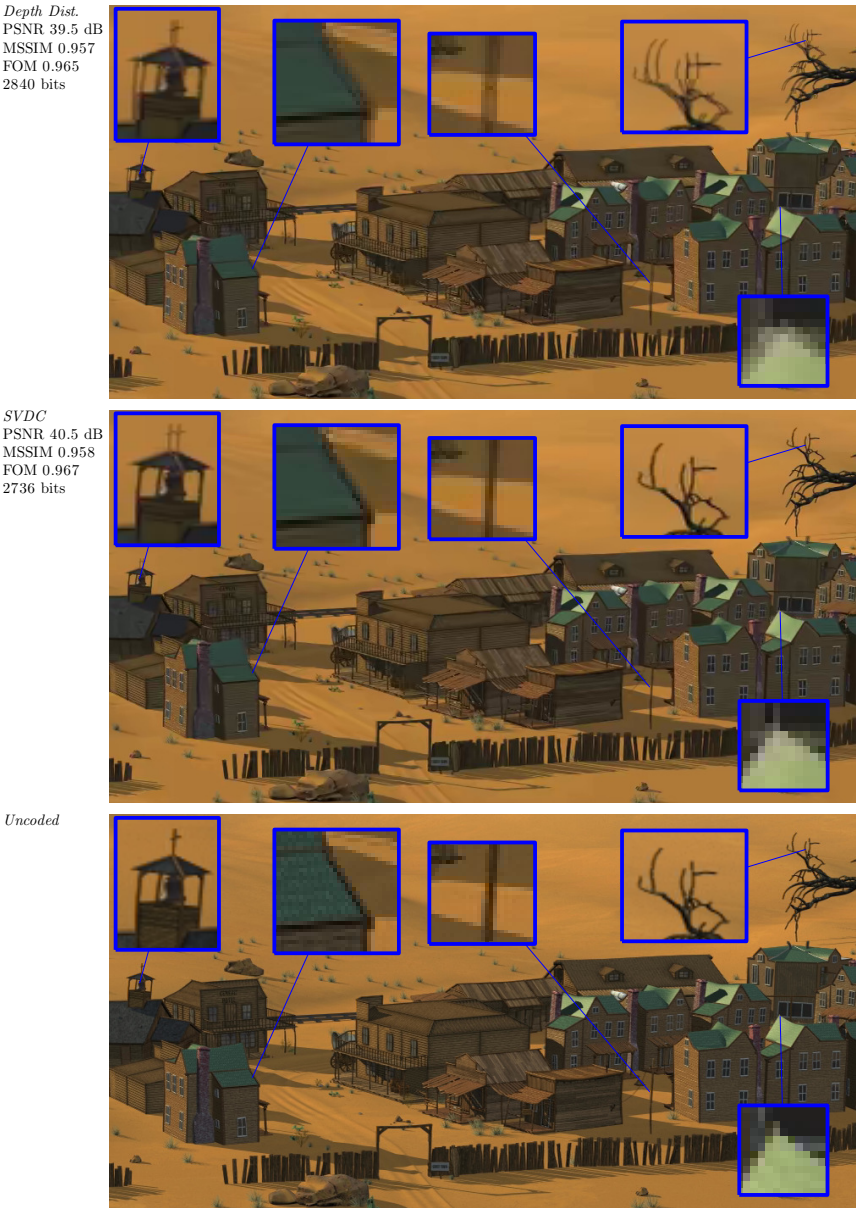


Figure F.57: SVDC- vs. depth distortion-based encoding; Sequence: *GTFly*.

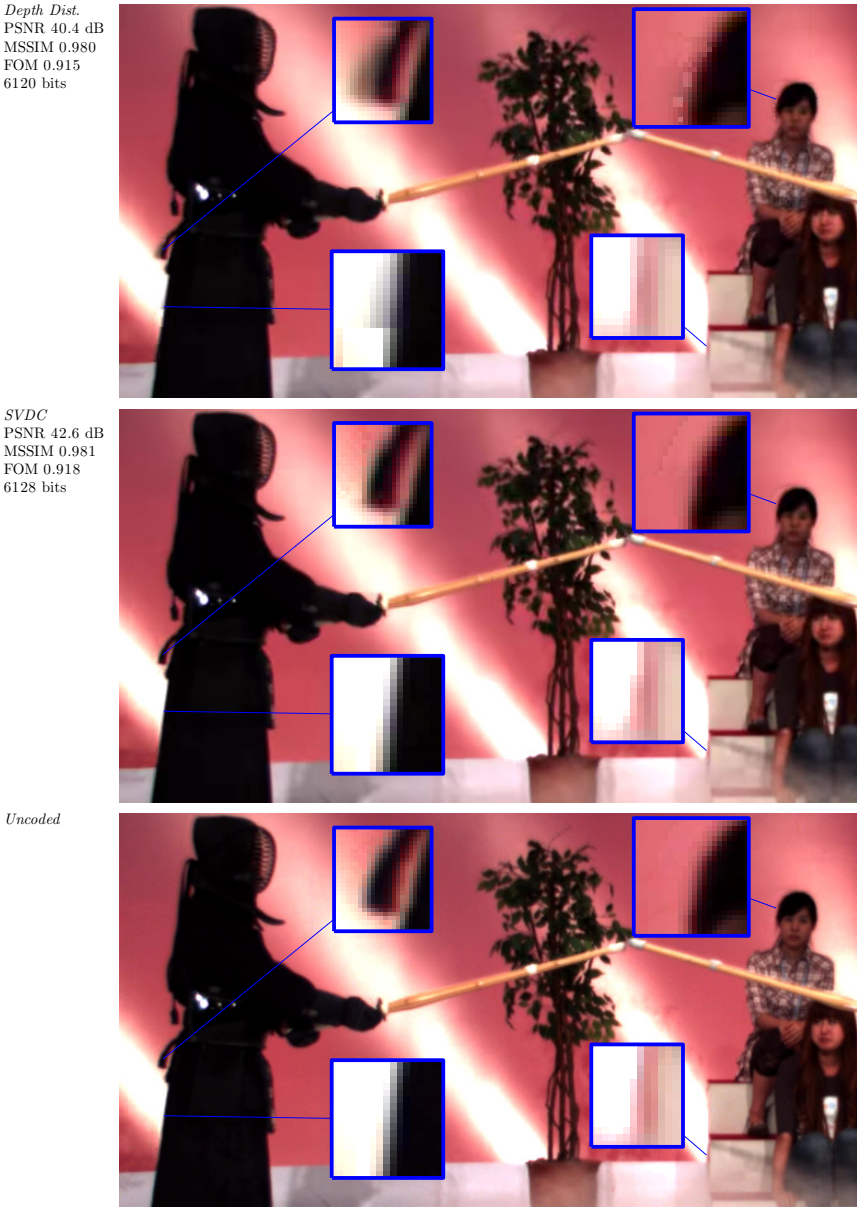


Figure F.58: SVDC- vs. depth distortion-based encoding; Sequence: *Kendo*.

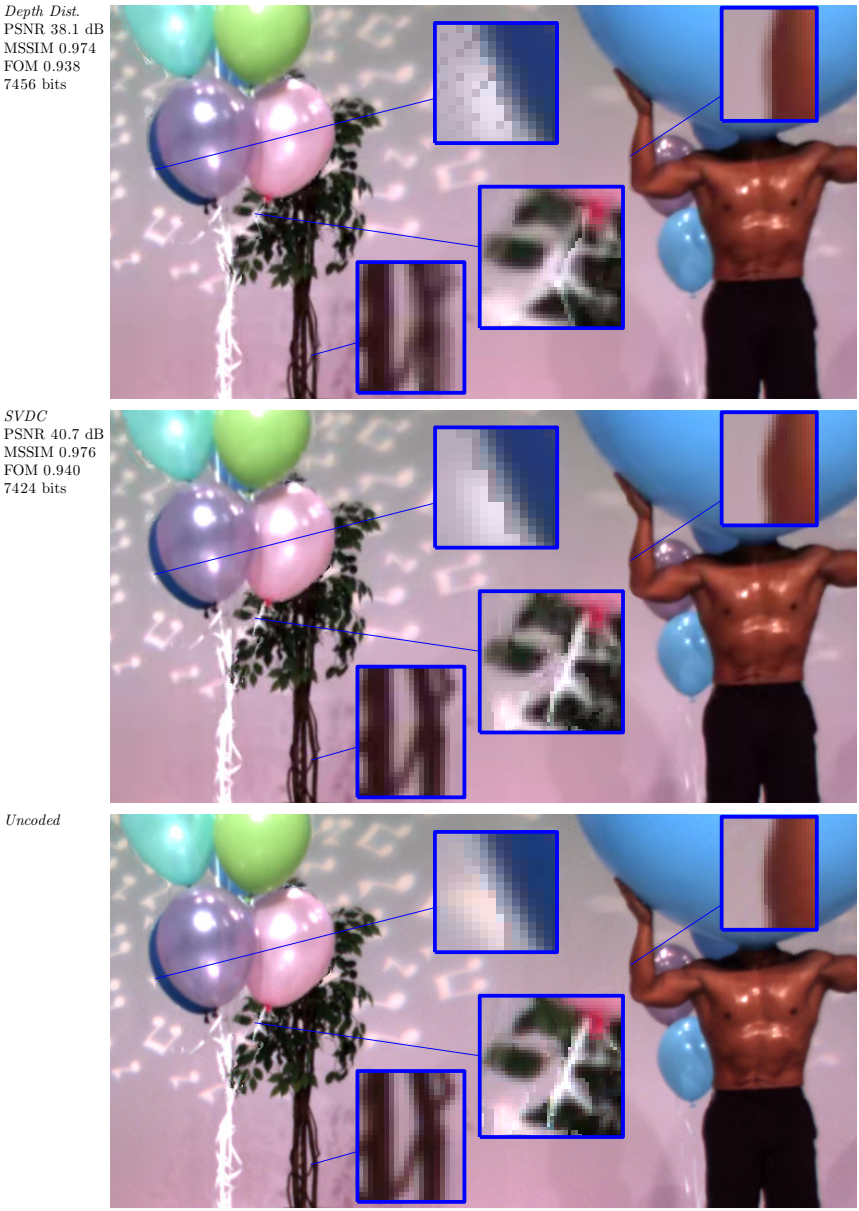


Figure F.59: SVDC- vs. depth distortion-based encoding; Sequence: *Ballons*.

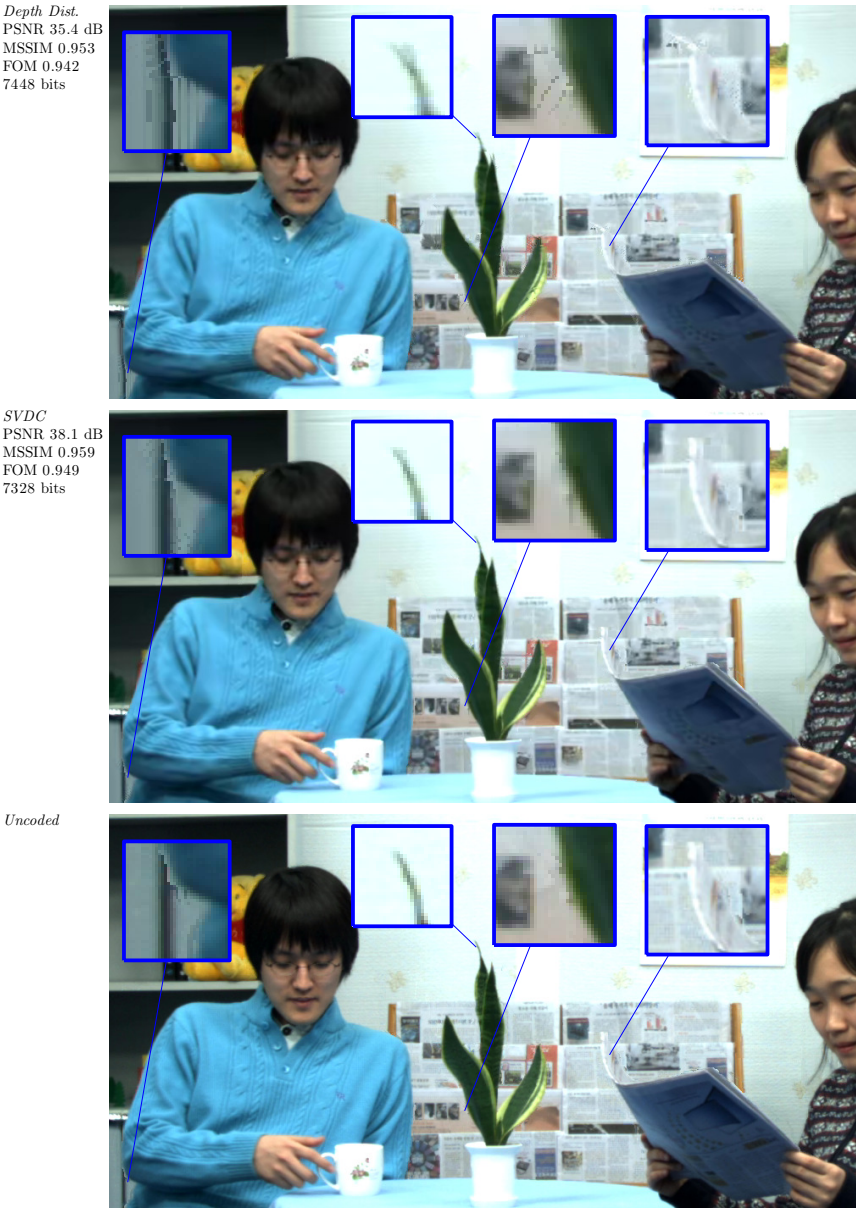


Figure F.60: SVDC- vs. depth distortion-based encoding; Sequence: *Newsp.*.

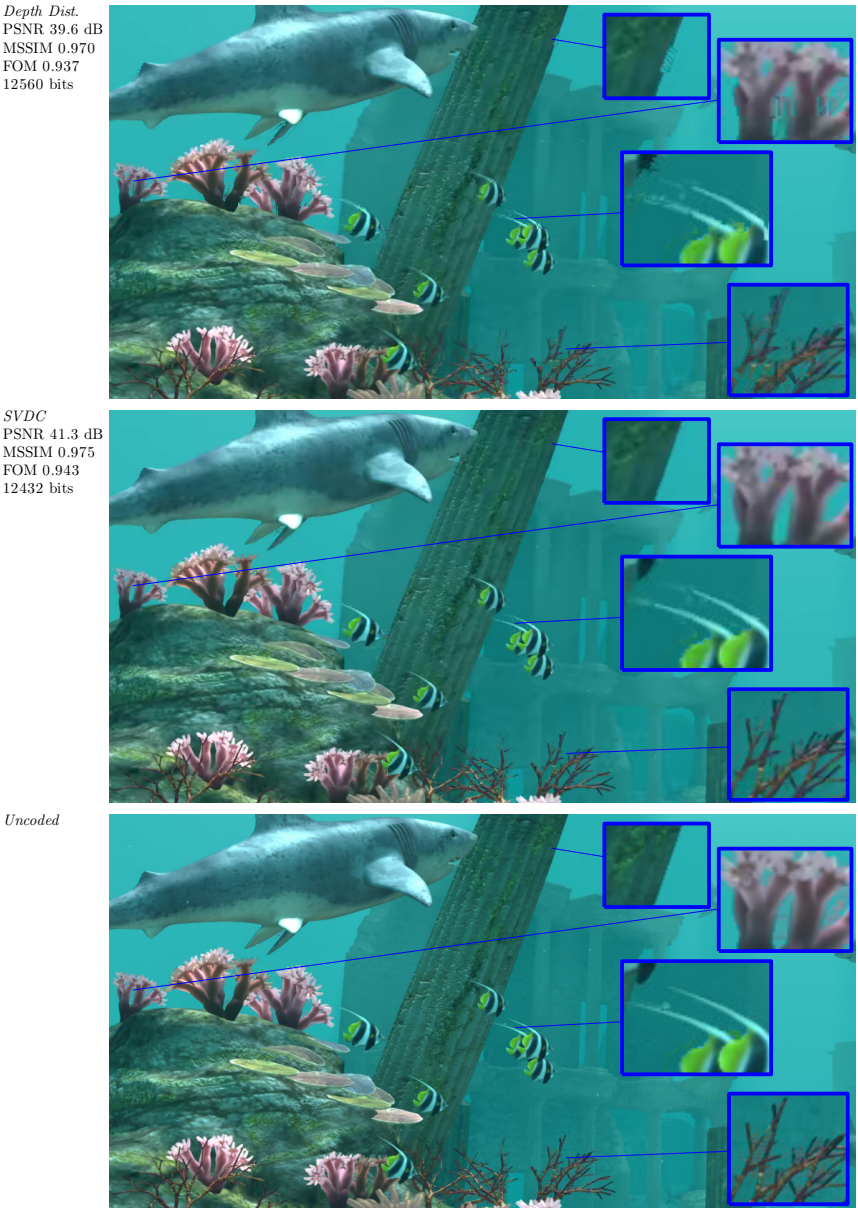


Figure F.61: SVDC- vs. depth distortion-based encoding; Sequence: *Shark*.

Glossary

Background suppression: Discarding of samples of the left or right SV texture based on their depth in view combination.

Bit rate delta: The bit rate difference of two RD curves averaged over a distortion range.

Changed SV region: The region in the SV that changes because of an IV depth change.

Depth candidate: The IV depth provided by the encoder to the RM for SVDC computation.

Depth distortion (DD): The distortion of the coded IV depth.

Depth distortion term: A depth distortion considered by the encoder in mode selection.

Encoding time delta: The difference of average encoding times in percent of the reference.

Evaluation setup: The setup used for evaluation of encoding, as defined in Section 7.2.

Hiding effect: The effect that the encoder encodes IV depth samples such that related IV texture samples are not visible in the SV texture because of background suppression.

Hole map: Values associated with the SV samples that indicate whether they belong to a hole.

IV depth change: The change of the IV depth, e.g. due to a depth candidate.

IV interval: An interval in the IV defined by two consecutive samples.

Initial encoder setup: The encoder setup using the default parameters defined in Section 7.2.

Input view (IV): A view generated in the acquisition step of the 3D video system.

Intermediate result: Result of a processing step required by following processing steps.

LS method: The least squares SVD estimation method as proposed in [Kim 09].

Left and right SV texture: The SV texture extrapolated from the left IV and the right IV, respectively.

Left edge interval: An SV interval on the left of a foreground object.

Low-complexity rendering algorithm (LCR): The renderer developed in this thesis.

Margin interval: An SV interval adjacent to the right margin of the picture.

Mode selection setup: The subset of the encoder setup that specifies the distortion metrics used by the mode selection stages, as defined in Table 7.1.

Mode selection stage: A subset of the encoding process using the same distortion metric, as defined in Table 7.1.

Occluded interval: An SV interval occluded by a foreground object.

Occlusion signal: Values associated with the IV that indicate the leftmost SV position already rendered.

Optimized encoder setup: The encoder setup using the default parameters given in Section 7.2 and the modifications given in Section 8.3.5

PSNR delta: The PSNR difference of two RD curves averaged over a bit rate range in percent of the reference.

Quadtree limitation: An encoder tool that disallows to partition the IV depth with a finer granularity than the IV texture.

RM setup: A subset of the VSO setup specifying the enabled features of the RM.

Reference SV texture: The SV texture to that SVD computation compares the tested SV texture.

Renderer model (RM): The SVDC computation method developed in this thesis.

Right edge interval: An SV interval on the right of a foreground object.

SVDC-based encoding: Encoding using the SVDC in rate-distortion optimization.

SVDC computation: The computation of the SVDC by comparing two SVDs.

SVD computation: The computation of the SVD by comparing the reference SV texture to the tested SV texture.

SVD estimation: The estimation of the SVD without view synthesis.

SV generation setup: The subset of the RM setup or the evaluation setup that specifies how the tested and the reference SV textures are rendered.

SV interval: An interval in the SV defined by warping an IV interval.

SV position setup: The subset of the VSO setup that specifies the number and position of the SVs regarded in SVDC computation.

Synthesized view (SV): A view generated by view synthesis.

Synthesized view distortion (SVD): The distortion of the SV texture.

Synthesized view distortion change (SVDC): The change of the SVD caused by an IV depth change.

Tested SV texture: The SV texture whose SVD is derived by SVD computation.

VSO early skip: An encoder tool that skips re-rendering when a depth change does not cause a disparity change.

VSO setup: A subset of the encoder setup specifying how the RM is used in RD optimization.

View combination: The process in view synthesis that combines the left and the right SV textures to the combined SV texture.

View combination variant: A modified view combination process a defined in Section 8.1.4

View extrapolation: View synthesis using the texture and depth of one IV.

View interpolation: View synthesis using the texture and depth of two IVs.

View synthesis: A process that renders SV textures from IV textures and IV depth.

View synthesis optimization (VSO): The application of the RM in encoding.

Abbreviations

AMVP	advanced motion vector prediction
AVC	Advanced Video Coding
BD	Bjøntegaard Delta
BNR	boundary noise removal
BS	background suppression
CABAC	context-adaptive binary arithmetic coding
CB	coding block
CBF	coded block flag
CTB	coding tree block
CTU	coding tree unit
CU	coding unit
DBBP	depth-based block partitioning
DCT	discrete cosine transform
DD	depth distortion
DIBR	depth image based rendering
DLT	depth look-up table
DR	depth refinement
FOM	Figure of Merit (Pratt's)
GOP	group of pictures
HEVC	High Efficiency Video Coding
HLS	high-level syntax
HM	HEVC test model
HTM	HEVC-based 3D test model
IBR	image based rendering
IEC	International Electrotechnical Commission
IRAP	intra random access point
ISO	International Organization for Standardization
ITU	International Telecommunications Union
ITU-T	ITU Telecommunication Standardization Sector
IV	input view
JCT-3V	Joint Collaborative Team on 3D Video Coding Extension Development
JCT-VC	Joint Collaborative Team on Video Coding
LCR	low-complexity rendering algorithm
LS	least squares
MPEG	Moving Picture Experts Group
MSE	mean squared error
MSSIM	mean structural similarity
MV	motion vector
MV	multiview
MV-HEVC	Multiview High Efficiency Video Coding

NBDV	neighboring block disparity vector
PB	prediction block
PDI	predicted disparity information
PPS	picture parameter set
PSNR	peak signal to noise ratio
PU	prediction unit
QP	quantization parameter
QT	quadtree
QTL	quadtree limitation
RD	rate-distortion
RDO	rate-distortion optimization
RDOQ	rate-distortion optimized quantization
RM	renderer model
RQT	residual quadtree
SAD	sum of absolute differences
SAO	sample adaptive offset
SBP	sub-block partition
SHVC	Scalable High Efficiency Video Coding
SPS	sequence parameter set
SSD	sum of squared differences
SSIM	Structural Similarity
SV	synthesized view
SVD	synthesized view distortion
SVDC	synthesized view distortion change
TB	transform block
TMVP	temporal motion vector prediction
TU	transform unit
VCEG	Visual Coding Experts Group
VPS	video parameter set
VQM	video quality metric
VSO	view synthesis optimization
VSP	view synthesis prediction
VSRS	view synthesis reference software

Symbols

α	Factor in SVD estimation
β	Factor in SVD estimation
θ	Spherical coordinate (3D scene)
λ	Wavelength (3D scene)
λ_M	Lagrange multiplier, depth distortion-based selection stages
λ_S	Lagrange multiplier, SVDC-based selection stages
μ_X	Local average
μ_Y	Local average
ν_P	Maximal disparity in SVD estimation
φ	Spherical coordinate (3D scene)
ρ	SVDC-depth distortion ratio
σ_X	Local standard deviation
σ_{XY}	Local cross-correlation
σ_Y	Local standard deviation
ω_M	Depth distortion weight
ω_S	SVDC weight
$\mathcal{D}$	Distortion derivation
$\mathcal{Q}$	Quality metric
$\mathcal{S}_R$	Intermediate rendering results, initial
$\dot{\mathcal{S}}_R$	Intermediate rendering results, before a depth change
$\ddot{\mathcal{S}}_R$	Intermediate rendering results, after a depth change
$\mathcal{V}$	View synthesis
$\mathbf{K}$	Camera calibration matrix (3D scene)
$\mathbf{P}$	Projection matrix (3D scene)
$\mathbf{R}$	Rotation matrix (3D scene)
$\mathbf{c}$	Camera projection center (3D scene)
$\mathbf{d_E}$	Vector of estimated SVDs
$\mathbf{e}_x$	Unit vector, x direction (3D scene)
$\mathbf{e}_y$	Unit vector, y direction (3D scene)
$\mathbf{e}_z$	Unit vector, z direction (3D scene)
$\mathbf{m}$	Point, 2D space (3D scene)
$\mathbf{m}_s$	Point, 3D space (3D scene)
$\mathbf{p}$	Disparity vector
$\mathbf{t}$	Translation vector (3D scene)
$[x'_{C,s}, x'_{C,e}]$	Changed SV region, i.e. $\tilde{B}'$

$[x'_{S,s}, x'_{S,e}]$	Old SV region, i.e. B'_S
$[\tilde{x}'_{S,s}, \tilde{x}'_{S,e}]$	New SV region, i.e. $\tilde{B}'_S$
$[x_s, x_e]$	IV interval
$[x'_s, x'_e]$	SV interval related to $[x_s, x_e]$
B	IV region
$\tilde{B}'$	Changed SV region
$B_{\parallel}$	Processed IV region
$B_{\dashv}$	Region on the left of $B_{\parallel}$
$B_{\vdash}$	Region on the right of $B_{\parallel}$
B_I	Picture
$\tilde{B}'_O[1]$	SV region related to $B[1]$ only
$\tilde{B}'_O[2]$	SV region related to $B[2]$ only
B'_S	Old SV region
$\tilde{B}'_S$	New SV region
$\tilde{B}'_U$	Superset of $\tilde{B}'$
$B[1]$	IV region
$\tilde{B}'[1]$	SV region related to $B[1]$
$B[2]$	IV region
$\tilde{B}'[2]$	SV region related to $B[2]$
$\tilde{B}'_J$	SV region related to $B[1]$ and $B[2]$
D	Texture distortion (in Section 6.2.1)
D	SVD
$\dot{D}$	SVD, before a depth change
$\ddot{D}$	SVD, after a depth change
D_C	Combined SVDC and depth distortion
$D_{\Delta P}$	SV PSNR delta (BD-delta)
$D_{\Delta P,M}$	Depth PSNR delta (BD-delta)
D_M	Depth distortion
D_P	PSNR
$D_{P,M}$	Depth PSNR
$D_{P,R}$	Reference SV texture PSNR
$D_{P,T}$	Test SV texture PSNR
$D_{P,e}$	Upper bound evaluated PSNR range
$D_{P,s}$	Lower bound evaluated PSNR range
D_S	Synthesized View Distortion Change
J_S	RD cost with respect to the SVDC
R	Texture bit rate (in Section 6.2.1)
R	Bit rate
R_{Δ}	Bit rate delta (BD-Delta)
$\bar{R}_R$	Logarithmized reference bit rate
$\bar{R}_T$	Logarithmized tested bit rate

$\bar{R}_e$	Upper bound evaluated logarithmized depth bit rate range
$\bar{R}_s$	Lower bound evaluated logarithmized depth bit rate range
T_Δ	Encoding time delta
b_O	Occlusion flag
d_E	Element of $\mathbf{d_E}$
d_{th}	Threshold in view combination
f	Focal length (3D scene)
h	Picture height in samples
l_m	Scale Lagrange multiplier
l_s	Scaling factor for Lagrange multiplier
o	Picture center coordinate (3D scene)
p	Disparity vector element
q	Quantization step size
$\ddot{s}_B$	Candidate IV depth in B
$\tilde{s}_B$	Candidate IV depth in B
s'_D	SVD, initial
$\dot{s}'_D$	SVD, before a depth change
$\ddot{s}'_D$	SVD, after a depth change
$\tilde{s}'_D$	SVD, final
$\hat{s}'_H$	Upsampled SV hole map
$s'_{H,l}$	Left SV hole map
$\hat{s}'_{H,l}$	Left upsampled SV hole map
$s'^{\infty}_{H,l}$	Left SV hole map with many holes
$s'^{\circ}_{H,l}$	Left SV hole map with holes
$s'_{H,r}$	Right SV hole map
$\hat{s}'_{H,r}$	Right upsampled hole map
s_M	IV depth
$\dot{s}_M$	IV depth, before a depth change
$\ddot{s}_M$	IV depth, after a depth change
$\tilde{s}_M$	Distorted IV depth
$\hat{s}'^{\circ}_M$	Upsampled SV depth with holes
$s_{M,l}$	Left IV depth
$\tilde{s}_{M,l}$	Left distorted IV depth
$\hat{s}_{M,l}$	Left upsampled IV depth
$s'_{M,l}$	Left SV depth
$s'^{\circ}_{M,l}$	Left SV depth with holes
$s'^{\infty}_{M,l}$	Left SV depth with many holes
$\hat{s}'^{\circ}_{M,l}$	Left Upsampled SV depth with holes
$s_{M,r}$	Right IV depth
$s'_{M,r}$	Right SV depth
$\hat{s}'^{\circ}_{M,r}$	Right upsampled SV depth with holes
s_O	Occlusion signal
$s_{O,l}$	Left IV occlusion signal

$s_{O,r}$	Right IV occlusion signal
s_P	IV disparity
$\tilde{s}_P$	Distorted IV disparity
$s_{P,l}$	Left IV disparity
$\tilde{s}_{P,l}$	Left distorted IV disparity
s_T	IV texture
$\tilde{s}_T$	Distorted IV texture
s'_T	SV texture
$\dot{s}'_T$	SV texture, before a depth change
$\ddot{s}'_T$	SV texture, after a depth change
$\tilde{s}'_T$	Distorted SV texture
$\hat{s}'_T$	Upsampled SV texture
s'°_T	SV texture with holes
$\hat{s}'^{\circ}_T$	Upsampled SV texture with holes
$s'_{T,Ref}$	Reference SV texture
$s_{T,l}$	Left IV texture
$\tilde{s}_{T,l}$	Left distorted IV texture
$\hat{s}_{T,l}$	Left upsampled IV texture
$s'_{T,l}$	Left SV texture
$s'^{\circ}_{T,l}$	Left SV texture with holes
$\hat{s}'^{\circ}_{T,l}$	Left upsampled SV texture with holes
$s_{T,r}$	Right IV texture
$\hat{s}_{T,r}$	Right upsampled IV texture
$s'_{T,r}$	Right SV texture
$s'^{\circ}_{T,r}$	Right SV texture with holes
$\hat{s}'^{\circ}_{T,r}$	Right upsampled SV texture with holes
s_Z	IV z values
t	Time
t	Translation vector element (3D scene)
w	Picture width in samples
w_B	Region width
w_n	Norm of homogeneous vector (3D scene)
x	Horizontal IV position
$\hat{x}$	Horizontal IV position, upsampled IV
x'	Horizontal SV position
$\check{x}'$	Horizontal SV position, rounded
$x_{B,e}$	Horizontal IV end position of an IV region B
$x_{B,s}$	Horizontal IV start position of an IV region B
x'_C	Minimal changed SV position
$x'_{C,e}$	Horizontal SV end position of the changed SV region $\tilde{B}'$
$x'_{C,s}$	Horizontal SV start position of the changed SV region $\tilde{B}'$
$\check{x}'_{FL}$	Hor. SV position of a foreground object's left edge, integer prec.
$\check{x}'_{FR}$	Hor. SV position of a foreground object's right edge, integer prec.

x'_O	Minimal occluded SV position
$\check{x}'_O$	Minimal occluded SV position, integer precsion
x_P	IV positions to terminate rendering
$x'_{S,e}$	Horizontal SV end position of the old SV region B'_S
$\tilde{x}'_{S,e}$	Horizontal SV end position of the new SV region $\tilde{B}'_S$
$x'_{S,s}$	Horizontal SV start position of the old SV region B'_S
$\tilde{x}'_{S,s}$	Horizontal SV start position of the new SV region $\tilde{B}'_S$
$\tilde{x}'_{U,e}$	Horizontal SV end position of the SV region $\tilde{B}'_U$
$\tilde{x}'_{U,s}$	Horizontal SV start position of the SV region $\tilde{B}'_U$
x'_V	Horizontal position of the SV (3D scene)
$x_{V,l}$	Horizontal position of the left IV (3D scene)
$x_{V,r}$	Horizontal position of the right IV (3D scene)
x_e	IV end position of an interval
x'_e	SV end position of an interval
$\check{x}'_e$	SV end position of an interval, rounded
x_s	x position (3D scene)
x_s	IV start position of an interval
x'_s	SV start position of an interval
$\check{x}'_s$	SV start position of an interval, rounded
y_s	y position (3D scene)
z_f	Maximum distance from the camera (3D scene)
z_n	Minimum distance from the camera (3D scene)
z_s	z position (3D scene)

Tables

Figures

Bibliography

[Adel 91] E. H. Adelson and J. R. Bergen. “The Plenoptic Function and the Elements of Early Vision”. In M. Landy and J. A. Movshon, Eds., *Computational Models of Visual Processing*, pp. 3–20, MIT Press, Camebridge, USA, 1991.

[Akar 07] G. B. Akar, A. M. Tekalp, C. Fehn, and M. R. Civanlar. “Transport Methods in 3DTV-a survey”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 11, pp. 1622–1630, Nov. 2007.

[Berr 03] R. Berretty and F. Ernst. “High Quality Images from 2.5D Video”. In *Proceedings of Eurographics 2003 Short Presentations*, pp. 255–262, Granada, Spain, Sep. 2003.

[Berr 06] R. Berretty, F. J. Peters, and G. T. G. Volleberg. “Real-Time Rendering for Multiview Autostereoscopic Displays”. In *Proceedings of the SPIE Stereoscopic Displays and Virtual Reality Systems XIII*, vol. 6055, pp. 208–219, San Jose, USA, Feb. 2006.

[Bert 01] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. “Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting”. In *Proceedings of the IEEE Society Conference on Computer Vision and Pattern Recognition*, pp. I–355, Kauai, USA, Dec. 2001.

[Bjon 01] G. Bjøntegaard. “Calculation of Average PSNR Difference between RD Curves”. VCEG-M33, Video Coding Experts Group, Austin, USA, Oct. 2001.

[Boev 08] A. Boev, D. Hollosi, and A. Gotchev. “Classification of Stereoscopic Artefacts”. Tech. Rep. D5.1, MOBILE 3DTV, July 2008.

[Boss 12] S. Bosse, H. Schwarz, T. Hinz, and T. Wiegand. “Encoder Control for Renderable Regions in High Efficiency Multiview Video Plus Depth Coding”. In *Proceedings of the Picture Coding Symposium*, pp. 129–132, Krakow, Poland, May 2012.

[Boyc 16] J. M. Boyce, Y. Ye, J. Chen, and A. K. Ramasubramonian. “Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding Standard”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 20–34, Jan. 2016.

[Cann 86] J. Canny. “A Computational Approach to Edge Detection”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[Chai 00] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. “Plenoptic Sampling”.

In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pp. 307–318, New Orleans, USA, July 2000.

[Chen 15] Y. Chen, G. Tech, K. Wegner, and S. Yea. "Test Model 9 of 3D-HEVC and MV-HEVC". JCT3V-K1003, Joint Collaborative Team on 3D Video Coding Extension Development, Geneva, Switzerland, Feb. 2015.

[Dodg 05] N. A. Dodgson. "Autostereoscopic 3D Displays". *IEEE Computer*, vol. 38, no. 8, pp. 31–36, Aug. 2005.

[Doma 09] M. Domanski, T. Grajek, K. Klimaszewski, M. Kurc, O. Stankiewicz, J. Stankowski, and K. Wegner. "Poznan Multiview Video Test Sequences and Camera Parameters". MPEG M17050, ISO/IEC JTC1/SC29/WG11, Xian, China, Oct. 2009.

[Dou 15] H. Dou, Y. L. Chan, K. B. Jia, and W. C. Siu. "View Synthesis Optimization Based on Texture Smoothness for 3D-HEVC". In *Proceedings of the IEEE International Conference on Conference on Acoustics, Speech and Signal Processing*, pp. 1443–1447, Brisbane, Australia, Apr. 2015.

[Dura 16] F. Durand. "Translation of Lippmann's 1908 Paper on Integral Photographs". http://people.csail.mit.edu/fredo/PUBLI/Lippmann.pdf, Accessed: April 2016.

[Ever 63] H. Everett. "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources". *Operations Research*, vol. 11, no. 3, pp. 399–417, May 1963.

[Fang 14] L. Fang, N. M. Cheung, D. Tian, A. Vetro, H. Sun, and O. C. Au. "An Analytical Model for Synthesis Distortion Estimation in 3D Video". *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 185–199, Jan. 2014.

[Fank 12] M. Fankam-Rabsch. *Kodierung von Tiefendaten unter Berücksichtigung von Bildsynthese-Verfahren*. Diplomarbeit, Institut für Technische Informatik und Mikroelektronik, TU Berlin, 2012.

[Fehn 04] C. Fehn. "3D-TV Using Depth-Image-Based Rendering (DIBR)". In *Proceedings of the Picture Coding Symposium*, pp. 307–312, San Francisco, USA, Dec. 2004.

[Foix 11] S. Foix, G. Alenya, and C. Torras. "Lock-in Time-of-Flight (ToF) Cameras: A Survey". *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, Sep. 2011.

[Fu 10] D. Fu, Y. Zhao, and L. Yu. "Temporal Consistency Enhancement on Depth Sequences". In *Proceedings of the Picture Coding Symposium*, pp. 342–345, Nagoya, Japan, May 2010.

[Gort 96] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. "The Lumigraph". In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pp. 43–54, New Orleans, USA, Aug. 1996.

[Hann 11] M. Hannuksela and D. Rusanovskyy. “Extension of Existing 3DV Test Set Toward Synthetic 3D Video Content”. MPEG M19221, ISO/IEC JTC1/SC29/WG11, Daegu, Korea, Jan. 2011.

[Hewa 09] C. T. Hewage, Z. Ahmad, S. T. Worrall, S. Dogan, W. A. C. Fernando, and A. Kondoz. “Unequal Error Protection for Backward Compatible 3-D Video Transmission over WiMAX”. In *Proceedings of the IEEE International Symposium on Circuits and System*, pp. 125–128, Taipei, Taiwan, May 2009.

[Horn 11] Y. R. Horng, Y. C. Tseng, and T. S. Chang. “VLSI Architecture for Real-Time HD1080p View Synthesis Engine”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 9, pp. 1329–1340, Sep. 2011.

[ITU 07] ITU-T and ISO/IEC JTC 1. “Advanced Video Coding for Generic Audiovisual Services (ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC)”. Nov. 2007.

[ITU 13] ITU-T and ISO/IEC JTC 1. “High Efficiency Video Coding (ITU-T Rec. H.265 and ISO/IEC 23008-2)”. Apr. 2013.

[Ivek 05] S. Ivekovic, A. Fusiello, and E. Trucco. “Fundamentals of Multiple-View Geometry”. In O. Schreer, P. Kauff, and T. Sikora, Eds., *3D Video Communication*, Chap. 6, John Wiley & Sons, New York, USA, 2005.

[JCT 16] JCT-3V. “3D-HEVC Reference Software, HTM-16.2”. https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-16.2, Joint Collaborative Team on 3D Video Coding Extension Development, Accessed: May 2016.

[Jung 12a] J. Jung, E. Son, and S. Yea. “3D-HEVC-CE10 Results on Modified Depth Distortion Measure by LG”. MPEG M23856, ISO/IEC JTC1/SC29/WG11, San Jose, US, Feb. 2012.

[Jung 12b] J. Jung, S. Yea, S. Ryu, D. Kim, and K. Sohn. “CE8.h Depth Distortion Metric with a Weighted Depth Fidelity Term”. JCT3V-A0119, Joint Collaborative Team on 3D Video Coding Extension Development, Stockholm, Sweden, July 2012.

[Jung 12c] J. W. Jung and S. Yea. “3D-CE4.h Results on Depth Distortion Metric with a Weighted Depth Fidelity Term”. JCT3V-B0131, Joint Collaborative Team on 3D Video Coding Extension Development, Shanghai, China, Oct. 2012.

[Kim 09] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila. “Depth Map Distortion Analysis for View Rendering and Depth Coding”. In *Proceedings of the IEEE International Conference on Image Processing*, pp. 721 –724, Cairo, Egypt, Nov. 2009.

[Kim 10] W.-S. Kim, A. Ortega, P. Lai, D. Tian, and C. Gomila. “Depth Map Coding with Distortion Estimation of Rendered View”. In *Proceedings of the SPIE Visual Information Processing and Communication*, vol. 7543, p. 75430B, San Jose, USA, Jan. 2010.

[Kim 15] W. S. Kim, A. Ortega, P. Lai, and D. Tian. “Depth Map Coding Optimization Using Rendered View Distortion for 3D Video Coding”. *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3534–3545, Nov. 2015.

[Koch 05] R. Koch and J.-F. Evers-Senne. “View Synthesis and Rendering Methods”. In O. Schreer, P. Kauff, and T. Sikora, Eds., *3D Video Communication*, Chap. 9, John Wiley & Sons, New York, USA, 2005.

[Lai 10] P. Lai, D. Tian, and P. Lopez. “Depth Map Processing with Iterative Joint Multilateral Filtering”. In *Proceedings of the Picture Coding Symposium*, pp. 9–12, Nagoya, Japan, May 2010.

[Lang 12] B. Langmann, K. Hartmann, and O. Loffeld. “Depth Camera Technology Comparison and Performance Evaluation”. In *Proceedings of the IEEE International Conference on International Conference on Pattern Recognition*, pp. 438–444, Vilamoura, Portugal, Feb. 2012.

[Lee 09] E. Lee and Y. Ho. “3-D Test Sequence - Multiview Video and Depth Map”. MPEG M16396, ISO/IEC JTC1/SC29/WG11, Maui, USA, Apr. 2009.

[Lee 16] J. Y. Lee and H. W. Park. “Efficient Synthesis-Based Depth Map Coding in AVC-Compatible 3D Video Coding”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 6, pp. 1107–1116, June 2016.

[Levo 96] M. Levoy and P. Hanrahan. “Light Field Rendering”. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques*, pp. 31–42, New Orleans, USA, Aug. 1996.

[Li 13] B. Li, J. Xu, D. Zhang, and H. Li. “QP Refinement According to Lagrange Multiplier for High Efficiency Video Coding”. In *Proceedings of the IEEE International Symposium on Circuits and System*, pp. 477–480, Beijing, China, May 2013.

[Li 14] C. Li, X. Jin, and Q. Dai. “A Novel Distortion Model for Depth Coding in 3D-HEVC”. In *Proceedings of the IEEE International Conference on Image Processing*, pp. 3228–3232, Paris, France, Oct. 2014.

[Lipp 08] G. Lippmann. “Épreuves réversibles donnant la sensation du relief”. *Journal de Physique Théorique et Appliquée*, vol. 7, no. 4, pp. 821–825, 1908.

[Liu 15] X. Liu, Y. Zhang, S. Hu, S. Kwong, C.-C. J. Kuo, and Q. Peng. “Subjective and Objective Video Quality Assessment of 3D Synthesized Views with Texture/Depth Compression Distortion”. *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4847–4861, Dec. 2015.

[Ma 13] R. Ma, N. M. Cheung, O. C. Au, and D. Tian. “Novel Distortion Metric for Depth Coding of 3D Video”. In *Proceedings of the IEEE International Conference on Image Processing*, pp. 1714–1718, Melbourne, Australia, Sep. 2013.

[Ma 14] S. Ma, S. Wang, and W. Gao. “Low Complexity Adaptive View Synthesis Optimization in HEVC Based 3D Video Coding”. *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 266–271, Jan. 2014.

[Marp 03] 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, no. 7, pp. 620–636, July 2003.

[McCa 11] K. McCann, S. Sekiguci, B. Bross, and W.-J. Han. “HEVC Test Model 3 (HM 3) Encoder Description”. JCTVC-E602, Joint Collaborative Team on Video Coding, Geneva, Switzerland, June 2011.

[Mori 09] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto. “View Generation with 3D Warping Using Depth Information for FTV”. *Signal Processing: Image Communication*, vol. 24, no. 1, pp. 65–72, Jan. 2009.

[MPEG 10a] MPEG. “Report on Experimental Framework for 3D Video Coding”. MPEG N11631, ISO/IEC JTC1/SC29/WG11, Guangzhou, China, 2010.

[MPEG 10b] MPEG. “View Synthesis Reference Software (VSRS) 3.5”. http://wg11.sc29.org/svn/repos/MPEG-4/test/trunk/3D/view_synthesis/VSRS, ISO/IEC JTC1/SC29/WG11, Accessed: March 2010.

[Mull 13] K. Müller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. H. Rhee, G. Tech, M. Winken, and T. Wiegand. “3D High-Efficiency Video Coding for Multi-View Video and Depth Data”. *IEEE Transactions on Image Processing*, vol. 22, no. 9, pp. 3366–3378, Sep. 2013.

[Mull 14] K. Müller and A. Vetro. “Common Test Conditions of 3DV Core Experiments”. JCT3V-G1100, Joint Collaborative Team on 3D Video Coding Extension Development, San Jose, USA, Jan. 2014.

[Ndji 11] P. Ndjiki-Nya, M. Köppel, D. Doshkov, H. Lakshman, P. Merkle, K. Müller, and T. Wiegand. “Depth Image-Based Rendering with Advanced Texture Synthesis for 3-D Video”. *IEEE Transactions on Multimedia*, vol. 13, no. 3, pp. 453–465, June 2011.

[Oh 11] B. T. Oh, J. Lee, and D.-S. Park. “Depth Map Coding Based on Synthesized View Distortion Function”. *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 7, pp. 1344 –1352, Nov. 2011.

[Oh 12a] B. Oh, J. Lee, D. Park, G. Tech, K. Müller, T. Wiegand, S. Wang, S. Ma, H. Liu, J. Jia, and J. Jung. “3D-CE8.h Results on View Synthesis Optimization by Samsung, HHI and LG-PKU”. JCT3V-A0093, Joint Collaborative Team on 3D Video Coding Extension Development, Stockholm, Sweden, July 2012.

[Oh 12b] B. T. Oh, J. Lee, and D. S. Park. "3D-CE8.h Results on View Synthesis Optimization Using Distortion in Synthesized Views by Samsung". MPEG M24830, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Apr. 2012.

[Oh 12c] B. T. Oh, J. Lee, and D. S. Park. "3D-HEVC-CE10 Results on Samsung's New Distortion Function for R-D Mode Decision". MPEG M23668, ISO/IEC JTC1/SC29/WG11, San Jose, US, Feb. 2012.

[Oh 12d] B. T. Oh, J. Lee, D. S. Park, G. Tech, K. Müller, and T. Wiegand. "3D-CE8.h Results on View Synthesis Optimization". JCT3V-A0033, Joint Collaborative Team on 3D Video Coding Extension Development, Stockholm, Sweden, July 2012.

[Oh 12e] K. Oh. "3D-HEVC-CE10 Summary Report on Distortion Measure for Depth Coding". MPEG M23688, ISO/IEC JTC1/SC29/WG11, San Jose, US, Feb. 2012.

[Oh 14] B. T. Oh and K. J. Oh. "View Synthesis Distortion Estimation for AVC- and HEVC-Compatible 3-D Video Coding". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 1006–1015, June 2014.

[Okos 80] T. Okoshi. "Three-Dimensional Displays". *Proceedings of the IEEE*, vol. 68, no. 5, pp. 548–564, May 1980.

[Park 11] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. "High Quality Depth Map Upsampling for 3D-TOF Cameras". In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1623–1630, Barcelona, Spain, Nov. 2011.

[Prat 78] W. K. Pratt. *Digital Image Processing*. John Wiley & Sons, New York, USA, 1978.

[Rama 06] P. Ramanathan and B. Girod. "Rate-Distortion Analysis for Light Field Coding and Streaming". *Signal Processing: Image Communication*, vol. 21, no. 6, pp. 462 – 475, July 2006.

[Rose 16] C. Rosewarne, B. Bross, M. Naccari, K. Sharman, and G. J. Sullivan. "High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Encoder Description Update 7". JCTVC-Y1002, Joint Collaborative Team on Video Coding, Chengdu, China, Oct. 2016.

[Salv 04] J. Salvi, J. Pages, and J. Batlle. "Pattern Codification Strategies in Structured Light Systems". *Pattern Recognition*, vol. 37, no. 4, pp. 827–849, Apr. 2004.

[Scha 02] D. Scharstein and R. Szeliski. "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms". *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, Apr. 2002.

[Schr 05] O. Schreer, P. Kauff, and T. Sikora. *3D Videocommunication: Algorithms,*

Concepts and Real-Time Systems in Human Centred Communication. John Wiley & Sons, New York, USA, 2005.

[Schw 11a] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Müller, H. Rhee, G. Tech, M. Winken, and T. Wiegand. "Description of 3D Video Coding Technology Proposal by Fraunhofer HHI (HEVC Compatible, Configuration A)". MPEG M22570, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Nov. 2011.

[Schw 11b] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Müller, H. Rhee, G. Tech, M. Winken, and T. Wiegand. "Description of 3D Video Coding Technology Proposal by Fraunhofer HHI (HEVC Compatible, Configuration B)". MPEG M22571, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Nov. 2011.

[Schw 11c] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Müller, H. Rhee, G. Tech, M. Winken, and T. Wiegand. "Description of 3D Video Coding Technology Proposal by Fraunhofer HHI (MVC Compatible)". MPEG M22569, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Nov. 2011.

[Schw 12a] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, D. Marpe, P. Merkle, K. Müller, H. Rhee, G. Tech, M. Winken, and T. Wiegand. "3D Video Coding Using Advanced Prediction, Depth Modeling, and Encoder Control Methods". In *Proceedings of the Picture Coding Symposium*, pp. 1–4, Krakow, Poland, May 2012.

[Schw 12b] H. Schwarz, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, K. Müller, H. Rhee, G. Tech, M. Winken, D. Marpe, and T. Wiegand. "Extension of High Efficiency Video Coding (HEVC) for Multiview Video and Depth Data". In *Proceedings of the IEEE International Conference on Image Processing*, pp. 205–208, Orlando, USA, Sep. 2012.

[Shao 14] F. Shao, W. Lin, G. Jiang, M. Yu, and Q. Dai. "Depth Map Coding for View Synthesis Based on Distortion Analyses". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 1, pp. 106–117, Mar. 2014.

[Shim 13] S. Shimizu, S. Sugimoto, and H. Kimata. "AHG11: 3D Test Materials from NICT-3D data set". JCT3V-C0101, Joint Collaborative Team on 3D Video Coding Extension Development, Geneva, Switzerland, Jan. 2013.

[Shoh 88] Y. Shoham and A. Gersho. "Efficient Bit Allocation for an Arbitrary Set of Quantizers". *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 9, pp. 1445–1453, Sep. 1988.

[Silv 09] D. V. S. X. D. Silva and W. A. C. Fernando. "Intra Mode Selection for Depth Map Coding to Minimize Rendering Distortions in 3D Video". *IEEE*

Transactions on Consumer Electronics, vol. 55, no. 4, pp. 2385–2393, Nov. 2009.

[Sjob 12] R. Sjoberg, Y. Chen, A. Fujibayashi, M. M. Hannuksela, J. Samuelsson, T. K. Tan, Y. K. Wang, and S. Wenger. "Overview of HEVC High-Level Syntax and Reference Picture Management". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1858–1870, Dec. 2012.

[Smol 09] A. Smolic, K. Müller, P. Merkle, P. Kauff, and T. Wiegand. "An Overview of Available and Emerging 3D Video Formats and Depth Enhanced Stereo as Efficient Generic Solution". In *Proceedings of the Picture Coding Symposium*, pp. 389–392, Chicago, USA, May 2009.

[Son 12] E. Son and S. Yea. "3D-CE8.h Results: On the Adequacy of the RD-Measure in VSO". MPEG M25010, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Apr. 2012.

[Stel 00] L. Stelmach, W. J. Tam, D. Meegan, and A. Vincent. "Stereo Image Quality: Effects of Mixed Spatio-Temporal Resolution". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 188–193, Mar. 2000.

[Sull 12] 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, no. 12, pp. 1649–1668, Dec. 2012.

[Sull 98] G. J. Sullivan and T. Wiegand. "Rate-Distortion Optimization for Video Compression". *IEEE Signal Processing Magazine*, vol. 15, pp. 74–90, Nov. 1998.

[Sze 14] V. Sze, M. Budagavi, and G. J. Sullivan. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Springer, 2014.

[Taka 10] Y. Takaki and N. Nago. "Multi-Projection of Lenticular Displays to Construct a 256-View Super Multi-View Display". *Optics express*, vol. 18, no. 9, pp. 8824–8835, Apr. 2010.

[Tan 16] T. K. Tan, R. Weerakkody, M. Mrak, N. Ramzan, V. Baroncini, J. R. Ohm, and G. J. Sullivan. "Video Quality Evaluation Methodology and Verification Testing of HEVC Compression Performance". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 76–90, Jan. 2016.

[Tani 09a] M. Tanimoto, T. Fujii, M. P. Tehrani, and M. Wildeboer. "Depth Estimation Reference Software (DERS) 4.0". MPEG M16605, ISO/IEC JTC1/SC29/WG11, London, UK, June 2009.

[Tani 09b] M. Tanimoto, T. Fujii, M. P. Tehrani, M. Wildeboer, N. Fukushima, and

H. Furihata. “Moving Multiview Camera Test Sequences for MPEG-FTV”. MPEG M16922, ISO/IEC JTC1/SC29/WG11, Xian, China, Oct. 2009.

[Tech 10] G. Tech, K. Müller, and T. Wiegand. “Diffusion Filtering of Depth Maps in Stereo Video Coding”. In *Proceedings of the Picture Coding Symposium*, pp. 306–309, Nagoya, Japan, Dec. 2010.

[Tech 11] G. Tech, K. Müller, and T. Wiegand. “Evaluation of View Synthesis Algorithms for Mobile 3DTV”. In *Proceedings of the IEEE 3DTV Conference*, Antalya, Turkey, May 2011.

[Tech 12a] G. Tech, K. Müller, and T. Wiegand. “3D-CE8.h Results on View Synthesis Optimization by HHI”. MPEG M24865, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Apr. 2012.

[Tech 12b] G. Tech, K. Müller, and T. Wiegand. “3D-HEVC-CE10 Results on View Synthesis Optimization by HHI”. MPEG M23714, ISO/IEC JTC1/SC29/WG11, San Jose, US, Feb. 2012.

[Tech 12c] G. Tech, H. Schwarz, K. Müller, and T. Wiegand. “3D Video Coding Using the Synthesized View Distortion Change”. In *Proceedings of the Picture Coding Symposium*, pp. 25–28, Krakow, Poland, May 2012.

[Tech 12d] G. Tech, H. Schwarz, K. Müller, and T. Wiegand. “Synthesized View Distortion Based 3D Video Coding for Extrapolation and Interpolation of Views”. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pp. 634–639, Melbourne, Australia, July 2012.

[Tech 16] 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, no. 1, pp. 35–49, Jan. 2016.

[Tech 18] G. Tech, K. Muller, 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, no. 6, pp. 1273–1287, June 2018. ©IEEE.

[Tian 09] D. Tian, P. Lai, P. Lopez, and C. Gomila. “View Synthesis Techniques for 3D Video”. In A. G. Tescher, Ed., *Proceedings of the SPIE Applications of Digital Image Processing*, vol. 7443, p. 74430T, San Diego, USA, Sep. 2009.

[Tsun 09] P.-K. Tsung, P.-C. Lin, L.-F. Ding, S.-Y. Chien, and L.-G. Chen. “Single Iteration View Interpolation for Multiview Video Applications”. In *Proceedings of the IEEE 3DTV Conference*, Potsdam, Germany, May 2009.

[Urey 11] H. Urey, K. V. Chellappan, E. Erden, and P. Surman. “State of the Art in Stereoscopic and Autostereoscopic Displays”. *Proceedings of the IEEE*, vol. 99, no. 4, pp. 540–555, Apr. 2011.

[Vetr 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, no. 4, pp. 626–642, Apr. 2011.

[Wang 04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. “Image Quality Assessment: From Error Visibility to Structural Similarity”. *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[Wang 12a] L. Wang, D. Fu, Y. Zhang, and L. Yu. “3D-HEVC-CE10 Results on Joint RDO for Depth Coding of 3D Video by ZJU”. MPEG M23829, ISO/IEC JTC1/SC29/WG11, San Jose, US, Feb. 2012.

[Wang 12b] L. Wang, D. Fu, Y. Zhao, and L. Yu. “3D-CE8.h: Results on JRDO and VSO with Different View Synthesis Algorithms”. MPEG M24899, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, Apr. 2012.

[Wang 12c] L. Wang and L. Yu. “3D-CE8.h Results on JRDO”. JCT3V-A0057, Joint Collaborative Team on 3D Video Coding Extension Development, Stockholm, Sweden, July 2012.

[Wang 12d] S. Wang, S. Ma, H. Liu, and J. Jia. “CE8.h: Results of Simplification of View Synthesis Optimization by Detection of Zero Distortion Change in Synthesized View”. JCT3V-A0083, Joint Collaborative Team on 3D Video Coding Extension Development, Stockholm, Sweden, July 2012.

[Wang 13] L. Wang and L. Yu. “Rate-Distortion Optimization for Depth Map Coding with Distortion Estimation of Synthesized View”. In *Proceedings of the IEEE International Symposium on Circuits and System*, pp. 17–20, Beijing, China, May 2013.

[Whea 38] C. Wheatstone. “Contributions to the Physiology of Vision.–Part the First. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision”. *Philosophical Transactions of the Royal Society of London*, vol. 128, pp. 371–394, 1838.

[Wieg 03] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra. “Overview of the H.264/AVC Video Coding Standard”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[Wien 14] M. Wien. *High Efficiency Video Coding: Coding Tools and Specification*. Springer, 2014.

[Xu 16] J. Xu, R. Joshi, and R. A. Cohen. “Overview of the Emerging HEVC Screen Content Coding Extension”. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 50–62, Jan. 2016.

[Yang 15] M. Yang, C. Zhu, X. Lan, and N. Zheng. “Parameter-Free View Synthesis Distortion Model with Application to Depth Video Coding”. In *Proceed-*

ings of the IEEE International Symposium on Circuits and System, pp. 2812–2815, Montreal, Canada, May 2015.

[Yuan 14] H. Yuan, S. Kwong, J. Liu, and J. Sun. "A Novel Distortion Model and Lagrangian Multiplier for Depth Maps Coding". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 3, pp. 443–451, Mar. 2014.

[Zeng 14] K. Zeng, T. Zhao, A. Rehman, and Z. Wang. "Characterizing Perceptual Artifacts in Compressed Video Streams". In *Proceedings of the SPIE Human Vision and Electronic Imaging XIX*, p. 90140Q, San Fransisco, USA, Mar. 2014.

[Zhan 14] Y. Zhang, S. Kwong, S. Hu, and C. C. J. Kuo. "Efficient Multiview Depth Coding Optimization Based on Allowable Depth Distortion in View Synthesis". *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4879–4892, Nov. 2014.

[Zhan 15] D. Zhang and J. Liang. "View Synthesis Distortion Estimation with a Graphical Model and Recursive Calculation of Probability Distribution". *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 5, pp. 827–840, May 2015.

[Zhan 16] Y. Zhang, X. Yang, X. Liu, Y. Zhang, G. Jiang, and S. Kwong. "High-Efficiency 3D Depth Coding Based on Perceptual Quality of Synthesized Video". *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5877–5891, Dec. 2016.

[Zhan 17] Y. Zhang, Z. Pan, Y. Zhou, and L. Zhu. "Allowable Depth Distortion Based Fast Mode Decision and Reference Frame Selection for 3D Depth Coding". *Multimedia Tools and Applications*, vol. 76, no. 1, pp. 1101–1120, Jan. 2017.

[Zhao 11a] Y. Zhao, C. Zhu, Z. Chen, D. Tian, and L. Yu. "Boundary Artifact Reduction in View Synthesis of 3D Video: From Perspective of Texture-Depth Alignment". *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 510–522, June 2011.

[Zhao 11b] Y. Zhao, C. Zhu, Z. Chen, and L. Yu. "Depth No-Synthesis-Error Model for View Synthesis in 3-D Video". *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2221–2228, Aug. 2011.