Compression and View Interpolation for Multiview Imagery

Stefan Richter

KTH Electrical Engineering

Master Thesis
Stockholm, Sweden, March 2011

urn:nbn:se:kth:diva-37699

Declaration / Erklärung

To the best of my knowledge and belief this work was prepared without aid from any other sources except where indicated. Any reference to material previously published by any other person has been duly acknowledged. This work contains no material which has been submitted or accepted for the award of any other degree in any institution.

Hiermit versichere ich die vorliegende Arbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Stockholm, den 6. März 2011

(Stefan Richter)

Abstract

This thesis deals with the tremendous amounts of data produced by camera arrays used in emerging technologies like 3D-TV and free-viewpoint TV. Being highly redundant, the imagery is eligible for efficient compression. A commonly used data format is N-texture/N-depth where each camera image is accompanied by pixelwise scene depth values. For this format, we present a novel codec. Tree-structured depth images are proposed to encode the depth information. Their advantage is efficient self-sustained compression without side information. To process the texture, we extend the class of Motion-Compensated Orthogonal Transforms by a wavelet decomposition along the view axis using depth image based rendering (DIBR). To cope with prevalent visibility issues of DIBR, orthogonal shape-adaptive transforms have been developed. The combination of the proposals yields a rate-distortion optimal codec. This system exploits the specific properties of depth images as opposed to previous approaches which rather attach a depth coder to a texture coder. The performance is evaluated by encoding multiview testsets and measuring the MSE after decoding. In the process, a MEX-file implementation of the Embedded Block Coding with Optimized Truncation algorithm has been created. We believe its speed boost can prove valuable for other's projects in the field.

Abbreviations, Acronyms and Symbols

DERS	depth estimation reference software [31]
DIBR	depth image based rendering [19]
EBCOT	embedded block coding with optimized truncation [32]
KLT	Karhunen Loève transform
MCOT	motion-compensated orthogonal transform [12, 13]
MSE	mean square error
MVC	multiview video coding
PSNR	peak signal-to-noise ratio
RDI	root node's depth image
TSDI	tree-structured depth image

N	number of viewpoints
H	height of the camera image
W	width of the camera image
u,v	camera pixel coordinates
X,Y,Z	world coordinates

$\mathbf{x}$	vector of pixel values
$\mathbf{y}$	vector of transform coefficients
$\hat{\mathbf{x}}, \hat{\mathbf{y}}$	respective quantized value
$\mathbf{T}$	transform matrix
$\mathbf{T}_\kappa$	incremental transform matrix

R_T	bitrate of the texture
R_D	bitrate of the depth map
R_{tot}	total bitrate
D	distortion

Contents

1 Introduction

Continuing improvements of camera and display technology meet the demand of more and more realistic visual presentation. Signal processing closes the gap between acquisition and reproduction of images to yield visual communication systems. Together, these increasingly powerful capabilities regularly boost the perceived quality.

While planar presentations like photography and television have become ubiquitous, emerging technologies strive to enable immersive experience of natural scenes. The most important applications are 3D-TV and free-viewpoint TV (FTV). The former is based on the binocular vision of human beings where two slightly different images are fused by the brain to yield depth perception. Hence, 3D-TV may use the same processing as FTV but simultaneously presents two aligned viewpoints.

Usually, multiple video cameras are used to simultaneously acquire various viewpoints of a scene [14]. The resulting set of images is referred to as multiview imagery. An example is presented in Fig. 1.2. As more camera viewpoints help to improve 3D realism and freedom of viewpoint choice, a huge amount of data is produced that calls for efficient compression [14, 18, 37]. We can exploit that the images are taken from the same scene and have inter-view as well as intra-view, temporal correlation [14, 37, 5].

In contrast to computer graphics, no geometric model is available in general. We have to base the processing on the camera images and possibly auxiliary data like, e.g., range measurements of a laser scanner. The resulting high-level system is depicted in Fig. 1.1. The available data is encoded into a bitstream which is then decoded. However, the viewpoints or their number are not necessarily identical at both ends. The view requested by the user or required for the faced display technology may be not contained in the input.

To obtain unavailable views, view interpolation using depth image based rendering (DIBR) [19] was proposed [20, 29]. However, high-quality synthesis requires information about the scene depth. It is usually structured as pixelwise depth map aligned with the texture pixels. To code the resulting data, several approaches with both predictive coding [21, 22, 38, 40] and subband coding [4, 5, 6, 15, 37] are available. They are essentially derived from single-view codecs and attach a subordinate depth map coding to the texture coding.

One aim of this thesis is exploiting the special characteristics of depth maps. We propose a datastructure for depth maps that requires no side information for encoding. Contrary to most existing algorithms, we encode the depth map first and then use it as side information in texture processing. Hence, the overall system also needs no side information and we save the respective bitrate.

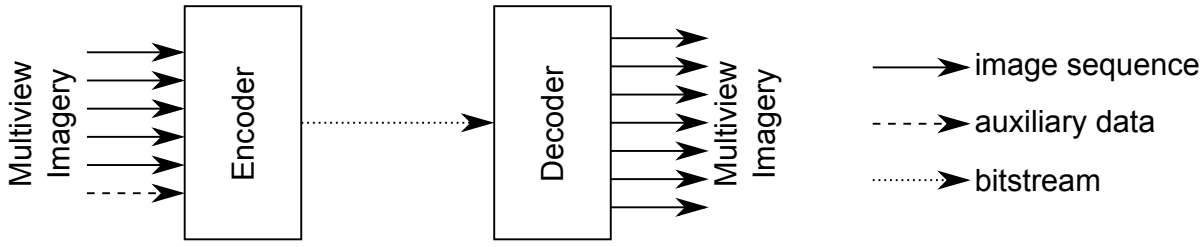


Figure 1.1: Signal processing of multiview imagery.

For texture processing, we extend the class of motion-compensated orthogonal transforms [12, 13] to use DIBR for the derivation of the compensation. Being strictly orthogonal, the transform enables relatively simple distortion calculation and embedded coding. In addition, an algorithm for joint rate-distortion optimization is presented that finds optimal bitrates of the depth map and the texture.

A common problem with DIBR is disocclusion, i.e., that a previously covered area becomes visible in a rendered view. For both depth map and texture processing, we present algorithms to cope with the disocclusion while the transform remains orthogonal.

The thesis is organized as follows. Chapter 2 presents background knowledge. The proposed depth image coder and texture coder are described in Chapters 3 and 4, respectively. In Chapter 5, we present the rate-distortion optimization and hence the concatenation of the two algorithms. Chapter 6 covers the experimental results followed by the conclusions in Chapter 7.

Figure 1.2: Views 3 to 6 of *Newspaper* test sequence.

2 Background

The goal of this chapter is a summary of some previous work found in literature. It introduces a number of concepts which are extended and combined to form a codec in Chapters 3, 4, and 5.

2.1 Multiview Imagery

For humans, the natural experience of a real-world scene is a three-dimensional (3D) impression where the viewpoint can be altered by moving oneself. Communication systems should be able to reproduce this for an immersive experience. The 3D perception is achieved if the two eyes are supplied with slightly different images like the brain would expect them for binocular vision [10]. Changing the viewpoint becomes possible if multiple viewpoints are captured and means to select them are provided. Some emerging applications are 3D-TV, free-viewpoint TV (FTV), and in teleconferencing as well as in surveillance [37, 18, 15, 14, 21].

Multiview imagery is a set of pictures taken from a dynamic scene. The pictures are captured at different viewpoints and/or consecutive time instants. They are typically generated with an array of synchronized video cameras [14, 21]. Further issues to obtain high quality are calibration of color, positions, and orientations of the cameras [21, 36].

A presentation on an autostereoscopic display, which is more convenient since it requires no glasses, may display 9 or more views simultaneously [29]. Free-viewpoint TV needs image samples of all surface areas that are visible in desired views. Hence, it also requires a large number of views and both applications create a huge amount of data [14, 37, 18]. However, the images are taken from the same scene and have inherent similarities which can be used for efficient compression [14, 37, 5]. Exploiting the also present temporal redundancies is well understood and a vast amount of literature exists. Additionally, the temporal correlations are known to be typically stronger [5] with certain exceptions [38].

2.2 Depth Image Based Rendering

Approaches like Light Field and Lumigraph [28] allow rendering views without knowing the geometry of a scene. However, this requires significant oversampling and thus high bandwidth for the transmission of a large number of views. Other approaches use information about the geometry to reduce the need of oversampling. They are based on the notion that a camera image is the projection of a 3D real-world scene on a 2D plane. The inverse of this projection and thus a mapping to another view becomes possible with appropriate auxiliary information. It can be encoded as point clouds [34], mesh-based representations [23] or depth images. The latter associates a depth value with each pixel of the image. This may be considered as an additional component alongside intensity and color data. A variant are Layered Depth Images (LDI) [26] which also store objects invisible from the LDI's viewpoint. Fig. 2.1 gives an example of a depth image.

Depth image based rendering (DIBR) [19] is used to obtain new viewpoints from given texture and depth images. It uses the computer-vision community's ideal planar pinhole model depicted in Fig. 2.2. The point of intersection of all rays is called center of projection [19] or optical center [21]. The distance

Figure 2.1: Example of depth image (*Newspaper*, frame 50, view 3).

between the optical center and the image plane is called focal length and denoted f. The relationship of a 3D world point $(X,Y,Z)^T$ and camera pixel $(u,v)^T$ is given by

$$u = \frac{Xf}{Z} \text{ and } v = \frac{Yf}{Z}. \tag{2.1}$$

Rewriting this to matrix notation using homogeneous coordinates for the pixel yields

$$\begin{pmatrix} uZ \\ vZ \\ Z \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \text{ where } \mathbf{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

is called intrinsic camera parameters. Different camera models are described by generalizing $\mathbf{K}$. To change the viewpoint, a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$ are introduced. Using homogeneous coordinates for the world point, (2.2) becomes

$$\begin{pmatrix} uZ \\ vZ \\ Z \end{pmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.3}$$

The reverse projection of the pixel coordinates to the 3D world is obtained by solving (2.3) for the world coordinates.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \mathbf{K}^{-1}\mathbf{R}^{-1} \begin{pmatrix} uZ \\ vZ \\ Z \end{pmatrix} - \mathbf{R}^{-1}\mathbf{t} \tag{2.4}$$

When (2.4) with one set of camera parameters $\{\mathbf{K}, \mathbf{R}, \mathbf{t}\}$ is inserted into (2.3) with the set of a second camera, we obtain

$$\begin{pmatrix} u_2 Z_2 \\ v_2 Z_2 \\ Z_2 \end{pmatrix} = \mathbf{K}_2 \begin{bmatrix} \mathbf{R}_2 | \mathbf{t}_2 \end{bmatrix} \begin{bmatrix} \mathbf{K}_1^{-1}\mathbf{R}_1^{-1} \begin{pmatrix} u_1 Z_1 \\ v_1 Z_1 \\ Z_1 \end{pmatrix} - \mathbf{R}_1^{-1}\mathbf{t}_1 \end{bmatrix}. \tag{2.5}$$

This process is called *3D image warping* in computer graphics.

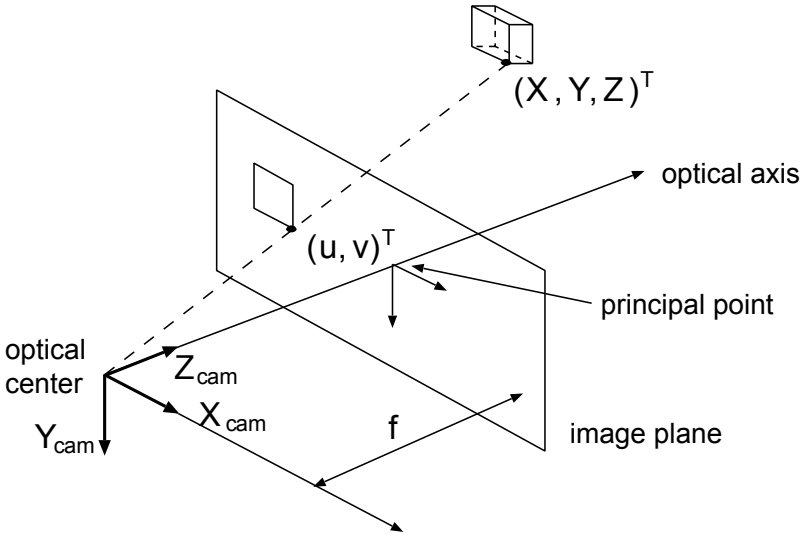


Figure 2.2: The ideal pinhole camera model describes the perspective projection $(u, v)^T$ on the camera's image plane of 3D world point $(X, Y, Z)^T$ (Source: [21]).

The depth is consistent, i.e. $Z = Z_1 = Z_2$, for planar camera setups where all optical centers are on one plane. Z is then the shortest distance from a surface point to that plane. The depth values of one view are found in all other views where the same surface point is visible, i.e., a single view can define the entire visible surface [24]. However, this only holds for continuous values. Problems arise for integer pixel coordinates and quantized depth values. A heuristic technique has to be used to 'bend' the result of (2.5) to the pixel grid, e.g., round it to the closest integer [21]. As a consequence, we have broken connections where (2.5) no longer establishes a connection between two pixels in two views. Instead, a connection to neighboring pixels may be claimed. On the other hand, it has been shown that limited quantization [16] and subsampling [3, 8] may be applied without disturbing perception quality.

3D warping may be applied to all pixel components. Since it only establishes the relationship between pixels, there are two possible directions to transfer the signal. So-called *forward warping* will use the depth value of a source view to determine the related pixel in the target view. The signal will be taken from the same view as the depth. The second approach is *backward warping* or *inverse warping*. Here, depth value and signal are taken from different views. The advantage of the former is that a single view can render any nearby view. However, this leaves pixels undefined, i.e., it introduces small holes of typically a single pixel because of the problems explained above. For inverse warping, all pixels will be defined since the depth is taken from the target view. On the other hand, this depth has to be known by some means.

While depth images may be derived by other methods like, e.g., LIDAR range measuring, they are typically obtained from the camera images using vision algorithms [24, 17, 21]. Depth Estimation Reference Software (DERS) developed by MPEG combines a number of approaches [31]. Despite the complexity, the results are rather poor. Most importantly, they are not consistent between the views because only neighboring views are used [24].

A common problem with DIBR is disocclusion. As demonstrated with the example in Fig. 2.3, objects in the foreground may cover areas farther away in a particular view. However, the areas may be visible at a different viewpoint. This inhibits simple rectangular datastructures as used for still images. In addition, it causes some uncertainty. Camera 1 cannot see if the gray object is present. Using just this camera, we hence cannot determine if camera 2 sees the area marked undetermined. It might just as well point on the possible occluder.

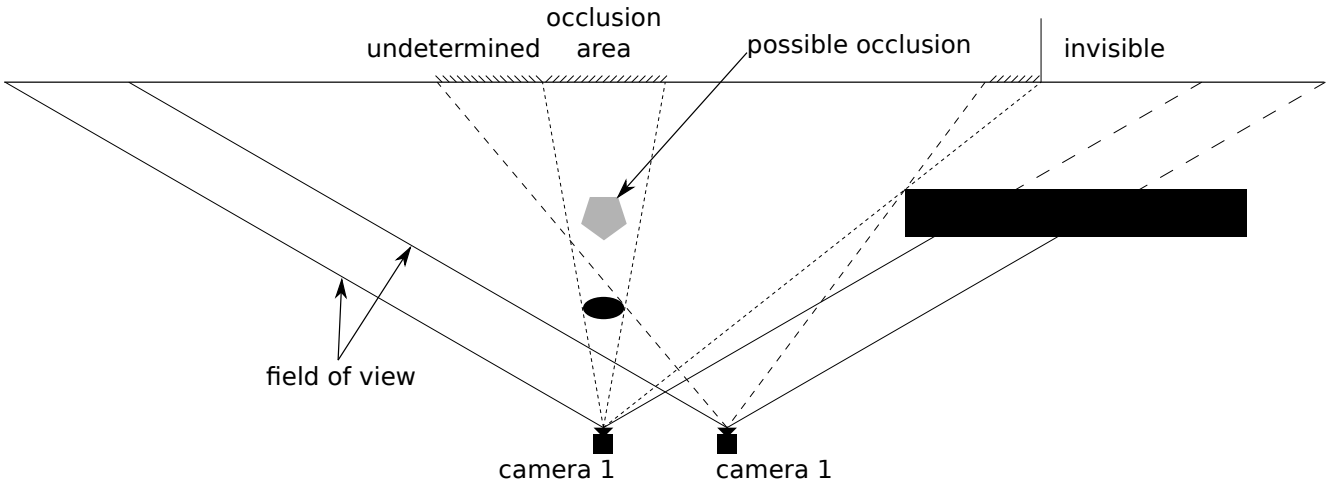
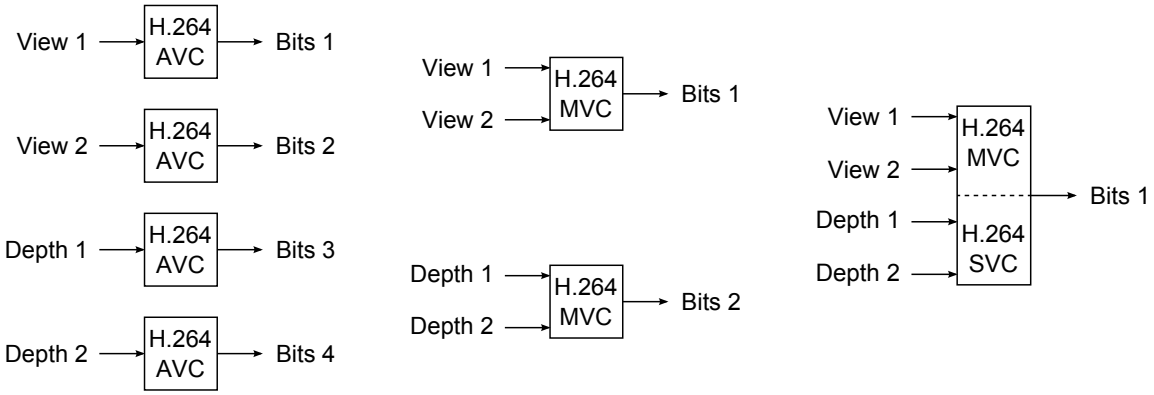


Figure 2.3: Occlusion. The black objects create areas visible in just one of the two views.



(a) Simulcast **(b)** Multiview coding **(c)** Joint MV+depth coding

Figure 2.4: Prediction structures for H.264 codec family.

2.3 Predictive coding

The H.264 video coding standard is widely used in predictive approaches to multiview video coding (MVC) since it is very efficient and has a large set of tools that can assist MVC. The most basic structure called *simulcast* (Fig. 2.4a) treats each view and each depth map as an independent sequence. Since the inter-view correlation is not exploited, the bitrate is relatively high [21]. H.264/MVC (Fig. 2.4b) encodes all views together. It extends the motion compensation such that it may choose a reference picture in a different view. This yields a disparity-compensated prediction and exploits the inter-view correlation [21, 40]. Morvan proposes to use a rendered view as additional reference [21, 38]. To enable high-quality, flexible view synthesis, depth maps are added as an independent MVC bitstream [40]. The motion vectors and diparity vectors of the texture are correlated to those of the depth map. Ref. [22] therefore uses the texture's side information as candidate for the respective depth map. Using H.264/SVC tools, this is extended to *joint multiview video plus depth coding* as depicted in Fig. 2.4c and achieves about 10 to 20% saving in depth bitrate [40].

The most important disadvantage of predictive coding is its foundation on closed-loop architecture. This denotes a system with feedback, i.e., it uses quantized and inverse-transformed output to process consecutive input. The quantization must not change later on since this error would accumulate other-

wise and the reconstruction fails. In contrast, an open-loop system does not mix the transform result with any part of the input signal in any way.

2.4 Lifted Wavelets

An open-loop system enables retroactive change of the transform coefficients. This is called scalability and may concern the rate by changing the quantization or the temporal resolution by omiting temporal highbands. The spatial resolution also may be reduced correspondingly but also improved locally when a selected set of coefficients is coded at better quality. The affected area is called *region of interest*. These features enable low-complexity adaptation to heterogeneous enviroments with varying channel bandwidths, different display resolutions, and diverse computation power.

A possibility is subband coding with wavelets. A number of wavelet-based multiview video codecs are discussed in literature [4, 5, 6, 15, 37] as well as related problems on rate-distortion optimization [33, 18]. The codecs usually use any or all of Haar, 5/3 - , and 9/7 - Cohen-Daubechies-Feauveau wavelets in lifting implementation. This ladder structure by Sweldend [30] decomposes the wavelet into prediction and update steps. Lifting allows nonlinear operations in the steps while the transform remains reversible. This feature is used to integrate motion compensation and disparity compensation. A problem is the different number of references for each pixel depending on the motion and disparity fields. The transform is only approximately orthogonal [12] and the resulting difficulty of rate-distortion optimization reduces the performance.

2.5 Orthogonal Transform

For all cases, the intended goal of transformation of the input images is a preparation that renders efficient compression as easy as possible. The optimal solution is the Karhunen Loève Transform (KLT) since it yields a decorrelated signal and achieves optimum energy concentration at the same time. However, we face the problem that the KLT depends on the signal statistics. These are diverse for images and only an estimate is available. Moreover, this has to be present for the inverse transform and thus needs bitrate. Since the KLT objects to be performed as sparse matrix multiplications, the computational load is high. For most applications, an approximation is inevitable. Common examples are the discrete cosine transform and wavelet transforms.

For orthogonal transforms, Parseval's theorem holds and we can calculate the MSE of the reconstructed images using only the error in the transform coefficients. Hence, the effect of quantization is known without performing the inverse transformation. This becomes very important in Sec. 2.6 to obtain embedded bitsreams. Another valuable feature is the separability of the 2D transform into two 1D transforms. Together with sparse transform matrices, this reduces the computational load significantly.

As an exemple, we consider the Haar wavelet. Let $\mathbf{x}$ and $\mathbf{y}$ denote vectors of the image values and the transform coefficients, respectively. Then, we have $\mathbf{y} = \mathbf{T}\mathbf{x}$ where $\mathbf{T}$ is the transform and may be factored into transform steps

$$\mathbf{T}_\kappa = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \tag{2.6}$$

Each transform step takes two input values and results in two coefficients. To form $\mathbf{T}$, we have to extend all $\mathbf{T}_\kappa$ to identity matrix except for the two components under transformation. This sparcity is exploitable since we may apply $\mathbf{T}_\kappa$ directly to the two respective components of $\mathbf{x}$. The separated 2D

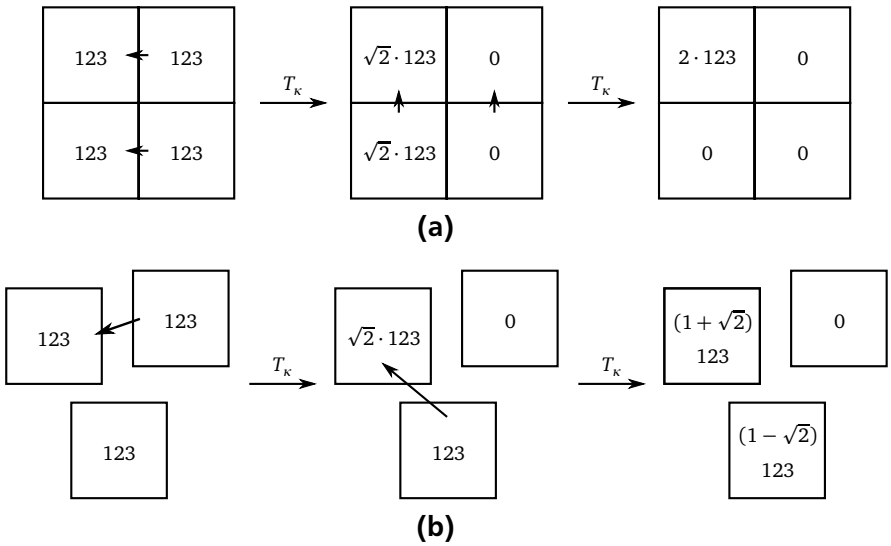


Figure 2.5: Separated transform of (a) regularly and (b) irregularly structured data.

transform decomposes the image horizontally first and then vertically. Hence, we processed rectangular blocks with four pixels when both horizontal and vertical wavelet are done. These blocks are non-overlapping. Fig. 2.5a illustrates the process. Since the values of the example are set identical, the so-called *approximation* takes all energy while all three so-called *detail coefficients* become zero. In general, they will be non-zero but nonetheless convey little energy for correlated signals.

A problem arises for irregularly structured data. Let a process like, e.g., motion compensation or DIBR declare the pixels in Fig. 2.5b to be correlated. One of the intermediate coefficients has twice the energy after one transform step, i.e., in the figure's middle. Multiplication with $\mathbf{T}_\kappa$ would yield non-zero detail. To overcome this problem, Ref. [12] introduces an adaptive decorrelation factor a_n and replaces $\mathbf{T}_\kappa$ by

$$\mathbf{H} = \frac{1}{\sqrt{1+a_n^2}} \begin{bmatrix} 1 & a_n \\ -a_n & 1 \end{bmatrix} \tag{2.7}$$

and we thus have

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \tag{2.8}$$

The transform (2.7) can be made strictly orthogonal by

$$a_n = \frac{\sqrt{n_2+1}}{\sqrt{n_1+1}} \tag{2.9}$$

unlike lifted wavelets [12]. The so-called *scale counters* n_1 and n_2 count the number of transformations performed on x_1 and x_2, respectively. The lowband's scale counter n_1 is update to m_1 according to $m_1 = n_1 + n_2 - 1$ since the energy is concentrated there. Both are initialized by zero and n_2 is also reinitialized to zero if further processing like spatial decomposition is desired.

The scale counters are reconstructed using the motion vectors or DIBR at the decoder. No additional information is needed compared to lifting approaches. However, these transforms are harder to construct and fewer wavelets are available.

The double motion-compensated orthogonal transforms (MCOT) is the second member of this family of MCOTs [13]. In similar fashion, two adaptive decorrelation factors are found to yield a strictly orthogonal transform with two hypotheses. Compatible spatial wavelets are also available [11]. We will reuse the transform steps and combine these transforms to obtain a view decomposition.

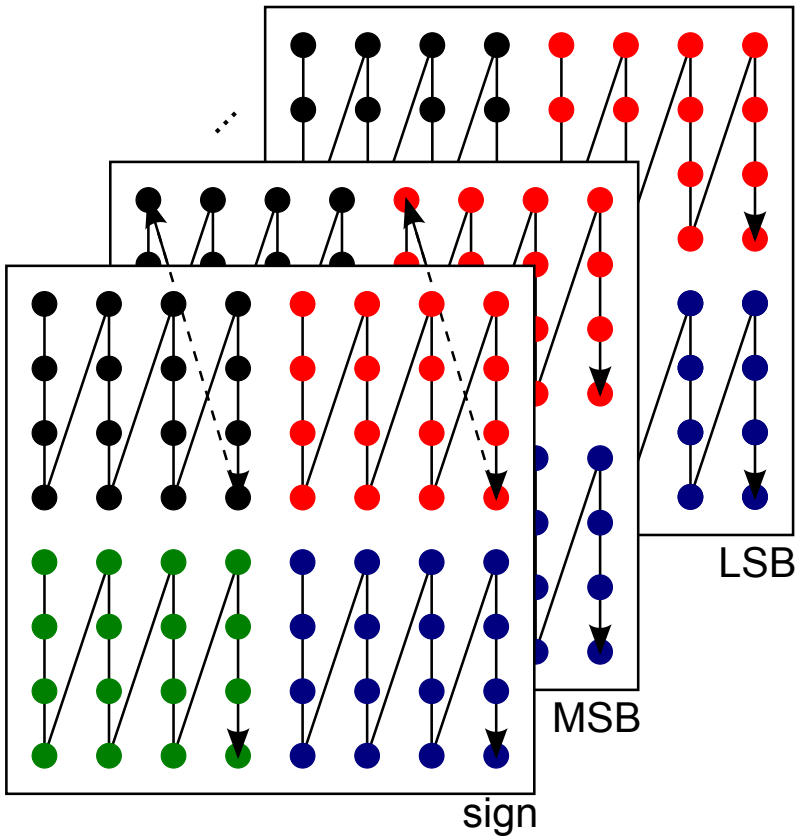


Figure 2.6: Bitplane coding along the arrows; colors indicate sets of coefficients.

2.6 Entropy Coding

The basic principle of compression is removing information considered redundant and finding the smallest but lossless description of the remaining important data. The former is achieved by preparing the signal and quantizing it. Appropriate entropy coding then exploits the coefficient statistics to assign a minimum length code for the values.

A well-known algorithm for entropy coding is Huffman coding. It optimally encodes individual source symbols by a variable-length string of output symbols (virtually always bits). The mapping table is constructed using the source symbol probabilities such that the result has minimal length on average.

Even after transformation, some spatial correlation of the transform coefficients in general remains for image data. Hence, a joint coding of the source symbols is desirable since it outperforms Huffman coding. Meanwhile, the scalability features of wavelet approaches should remain. So-called *embedded coding* sorts the bits according to their importance. Truncating this single embedded bitstream, different bitrates and spatial or temporal resolutions are obtained. Additionally, a particular version contains all lower qualities as prefixes.

A prerequisite of embedded bitstreams is embedded quantization. The bits of some set of coefficients are arranged in so-called *bitplanes*. Each bitplane contains all bits of the same significance, i.e., the first has all *most significant bits*, the last has all *least significant bits* etc. as depicted in Fig. 2.6. By defining a scan order on the bitplanes, a stream of bits is obtained. This bitstream may be truncated after one of the bitplanes. Thereby, the effective step size of the quantizer is controlled.

Many popular wavelet entropy coders exploit a property of images called *self-similarity* or *cross-subband similarity*. The hypothesis is that all finer details are irrelevant if a threshold at a coarse level

indicates irrelevance. A tree ordered by importance is obtained by decreasing the threshold. This so-called *zerotree* can be coded efficiently. Examples for this approach are EZW [27] and SPIHT [25]. The otherwise similar WDR [39] efficiently encodes indices of significant wavelet coefficients instead of a zerotree which actually indicates insignificant parts.

Another encoder for wavelet coefficients is EBCOT [32]. As the first of two steps, the quantizer indices for each subband are partitioned into code blocks. Code blocks are rectangular in shape, and their nominal size is a free parameter of the coding process [2]. Fig. 2.6 depicts four code blocks indicated by the four colors. EBCOT passes through the bitplanes of each code block and tries to predict the next bit using the already encoded. This yields information for context-adaptive arithmetic coding that in turn allows efficient coding with no or few prior knowledge about the signal. The produced embedded bitstreams for the code blocks are mutually independent. Thus, EBCOT is very flexible for features like scalability, regions-of-interest, or rate-distortion optimization. Although abandoning cross-subband correlation, the algorithm exhibits equal or slightly better compression performance than the previous. The drawback is the implementation and computation complexity. However, powerful implementations exist since a variant of EBCOT is the entropy coder of ISO 15444, known as JPEG2000 [1]. Moreover, JPEG2000 was adopted for digital cinema [7] suggesting that it is suitable for similar applications.

The EBCOT framework provides means to optimize the distortion D of the transform coefficients $\mathbf{y}$ for a given rate. The assumptions are that the distortion metric is calculated in transform domain and the distortion of the individual code blocks is additive, i.e.,

$$D = \sum_{\text{all } i} D_i \tag{2.10}$$

where $D_i = \mathrm{f}(\mathbf{y_i}, \mathbf{y_i})$ and i is the code block index. We spare the computation effort of the inverse transform and can successively derive the distortion for each truncation point during bitstream creation. A prerequisite is an open-loop system. The additive distortion simplifies the optimization problem since it becomes an easier independent optimization for each code block. One possible metric is the mean squared error (MSE). Since its calculation is very simple, MSE is very popular for image and video coding. However, MSE does not take the human distortion perception into account. Taubman shows that, e.g., the visual distortion metric [32] improves subjective quality while fulfilling the above assumptions.

To account for the characteristics of 3D subband decompositions, 3-D ESCOT was proposed [35]. It improves EBCOT by introducing new contexts. Further extension to even more dimensions suggests itself and presumably improves performance. Nevertheless, we limit the remainder of this thesis to EBCOT since all basic principles are covered and an implementation is available.

3 Multiview Depth Image Coding

A depth image is sufficient to render new depth images since the depth value is the signal but it is also possible to deduce the relationship to pixels in other views with it. Hence, all information to predict a different view is contained and we can build a compression system with the depth map alone. In contrast, the texture needs side information for parallax compensation which can be derived from depth images. Moreover, we ultimately need the depth map for view synthesis for FTV or adaptation to the heterogeneous display technology. These reasons motivate an overall system in two parts as illustrated in Fig. 3.1. The first part will independently process the depth. The result will be used as side information in both encoding and decoding of the second part which handles the texture. Thereby, we can use the depth map both as side information and as output.

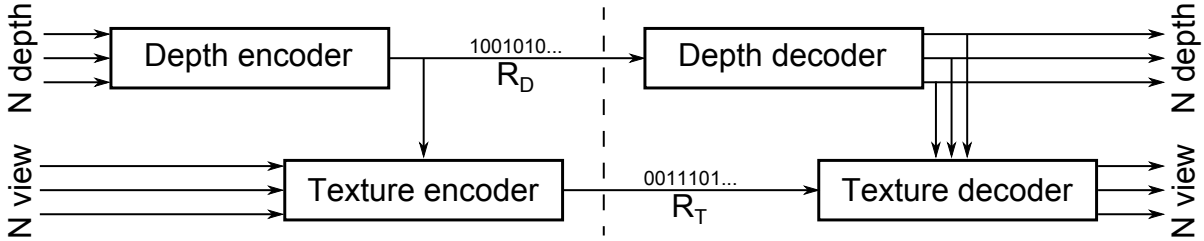


Figure 3.1: Proposed structure of the complete system.

3.1 Tree-Structured Depth Image

Each pixel of a depth image represents a point in the 3D world. In a non-sampled and non-quantized case, the identical value is found in the depth image of a different view for planar camera arrays or it can be calculated in the general case. Assuming this holds approximately in the sampled and quantized case, each depth value should be coded once and only once. This assumption is supported by the observation that limited errors are tolerated by humans and do not disturb perceived quality.

However, a problem occurs for occlusions. Some areas which were previously invisible since covered by foreground objects might be visible in a different perspective. The shape of these areas is arbitrary. Thus, the concerned pixels cannot be coded in the rectangular structure of a depth image. We therefore propose the concept of tree-structured depth images (TSDI) as illustrated in Fig. 3.2. The rectangles represent four views of a scene. Note that their arrangement is not merely caused by the tree but rather due to the spatial alignment of the viewpoints. We can maintain this and hence there is no need to tamper with the geometry of the processed pixels.

The black pixels of Fig. 3.2 indicate pixels that are encoded, white pixels are rendered using the information of parents in the tree. However, the structure of the tree is pixel dependent. Some might be rendered using the depth values in an auxiliary view, since they are only visible there. This is indicated by the dashed arrow and the gray pixels in Fig. 3.2.

The tree can be serialized. The root node's depth image (RDI) defines occlusion areas in its descendants. The depth values of one child (e.g., view 3 in Fig. 3.2) have to be coded but we can simply

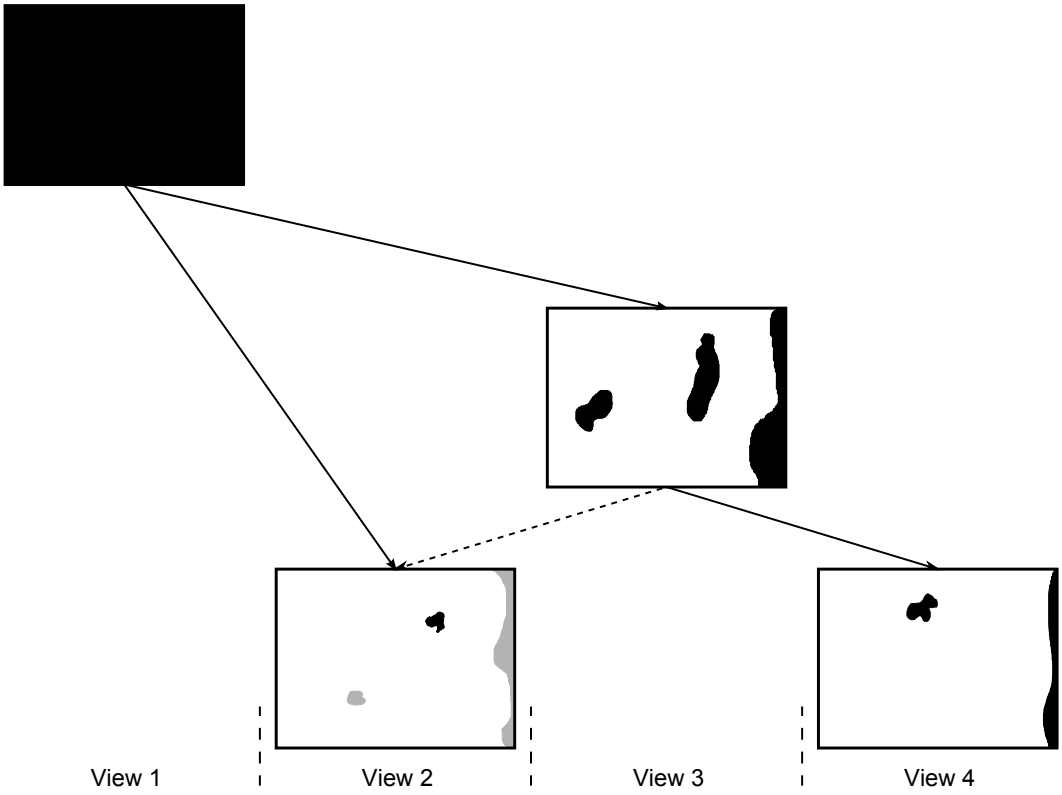


Figure 3.2: Structure of tree-structured depth images (black: encoded pixel, white: rendered pixel, gray: rendered with auxiliary view).

append them to a bitstream of the RDI since we now know the shape of the occlusion areas of this view. This process is then applied to the subtree at the child and so on until the tree has been traversed.

3.2 Generation of TSDI

If we transform connected pixels of consistent depth maps, the highbands are zero as the depth values are identical. This is good since we need no bitrate to transfer zeros. However, the decoder has to know the connections for this to work. In practical application, it has to derived them from the transform coefficients. However, this is ambivalent. As an example, a transform coefficient of the orthogonal transform can have the value $100\sqrt{2}$. Without further information, this can indicate two connected pixels with a depth value of 100 or actually be the result of $25\sqrt{8}$, i.e., eight connected pixel and depth value 25. Solving this ambiguity seems to require extra bitrate for open-loop systems. On the other hand, closed-loop approaches do not face this problem. The encoded data is a single view first and each coefficient represents exactly on pixel. The other views are added consecutively.

We generate TSDIs as depicted in Fig. 3.3. The RDI is encoded as the first step. Using forward warping of the decoded RDI depth values, the view of the highest subtree, i.e., view 3 in Fig. 3.2, is rendered. To cope with holes, we apply a 3×3 median filter two times to a copy of this intermediate result. The undefined pixels are then replaced by the filter result. When the same warping error occurs for foreground objects, the background will become visible. Therefore, we median-filter the second intermediate result. Additionally, this will remove scattered pixels from the occlusion areas. They appear depending on the gradient at the edge of an object in the source view. In general, some pixels remain undefined after this processing. They are considered as the occlusion areas and are encoded. Particularly,

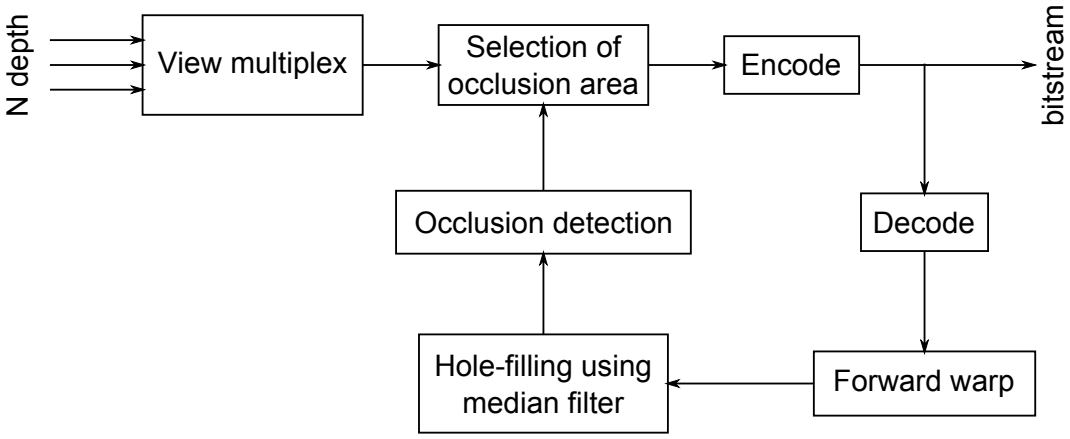


Figure 3.3: Generation of TSDI as closed-loop process.

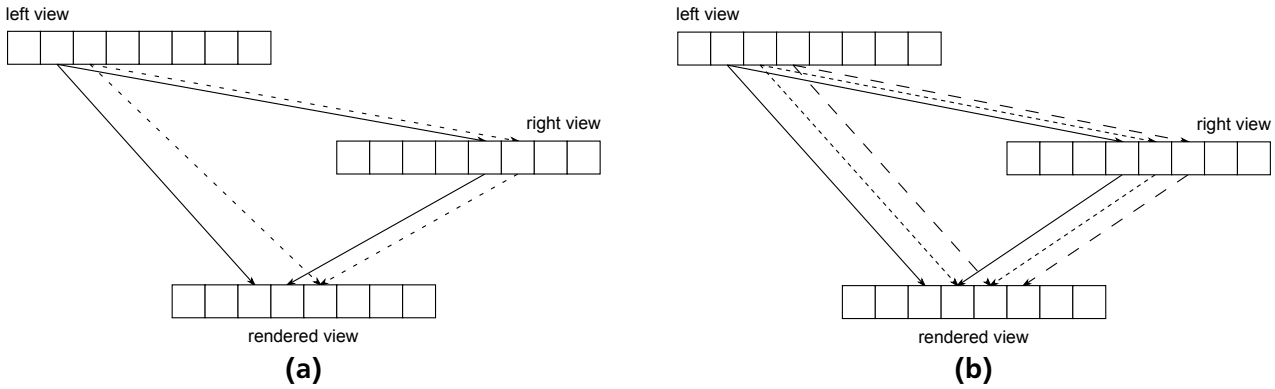


Figure 3.4: Mitigating round-off problems with second warp (a) A hole is filled because of different round-off (solid arrows) (b) Filter effect due to displacement of 3D warps.

we discard the depth values of the auxiliary view which are outside the occlusion mask. This reduction of the dimensionality will decrease the required bitrate while we are still able to reconstruct the depth value because of the consistency property.

The remaining views are treated in similar fashion. However, since rendering results are better for small baselines, we use the closest view for the rendering. Particularly, we expect to reduce the number of holes. For example, view 4 in Fig. 3.2 is rendered using view 3 despite many values are actually derived from view 1. To further mitigate the round-off problems, both views will be rendered to create a new view when the baselines are equal. The result is then derived as the average of the two depth values. If no round-off error or occlusion is involved, i.e., dashed arrow in Fig. 3.4a, the two values and hence also the result are identical. Otherwise, the second warp will either fill a hole or result in a spatial filtering, as illustrated in Fig. 3.4a and 3.4b, respectively.

Since the input depth images are usually not consistent, we preprocess them before we actually generate the TSDI. All views are warped to the RDI's view. For each pixel, we then average over all obtained samples. We exclude depth values outside an interval of two times the standard deviation. The calculation of the final average, mean, and standard deviation ignore undefined values. The auxiliary views are processed in the same fashion except that views of parent nodes are not included. The latter will not contribute since only the occlusion areas are relevant which in turn are defined as invisible in the parents. Thereby, we save computation time for the warps.

3.3 Reconstruction of Multiview Depth Image

Since we know the rectangular shape of the RDI, we can extract it from the beginning of the bitstream and decode it. The same warping and filtering process and occlusion detection as above is used to obtain the next view of the tree. Since the same process and data as during generation is used, the shape of the occlusion area will be the same. Thus, we can decode the depth values for this area and insert them to complete the view. Finally, the original order of the views is restored in a demultiplexer.

3.4 Coding of TSDI

3.4.1 Orthonormal Transformation

To exploit spatial correlations of the depth images, they are decomposed using a type-1 spatial wavelet transform [11]. It becomes the Haar wavelet for the RDI where a regular dyadic decomposition can be used. We use four resolution levels, i.e., three decomposition stages. The benefit of using an orthonormal transform is the simple quantizer. All coefficients are quantized with the same step size in a uniform midthread quantizer. The same quantization parameters are used for the coefficients of the occlusion areas. However, a different transform has to be used since they have irregular shapes in general. Moreover, we cannot produce an embedded bitstream since we are forced to choose a closed-loop approach.

3.4.2 Shape Adaptation

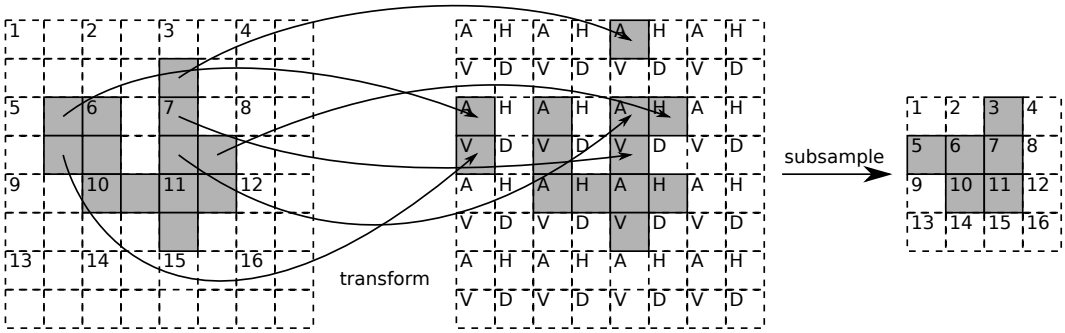
Both at the encoder and the decoder, the occlusion mask is known. Hence, no additional bits to encode the shape are required if we use this mask to adapt the transform to the shape. The occlusion area is divided into 2×2 blocks at the same positions of the non-adaptive transform. Only the blocks where some of the pixels are not part of the occlusion area are treated specially. All complete blocks are processed as above.

Incomplete blocks may have one, two or three defined pixels and we consider these cases separately. We try to optimize the shape adaptation at each decomposition level such that later decompositions are as tidy as possible. An example of the shape-adaptive transformation is depicted in Fig. 3.5a and the resulting second-level decomposition in Fig. 3.5b.

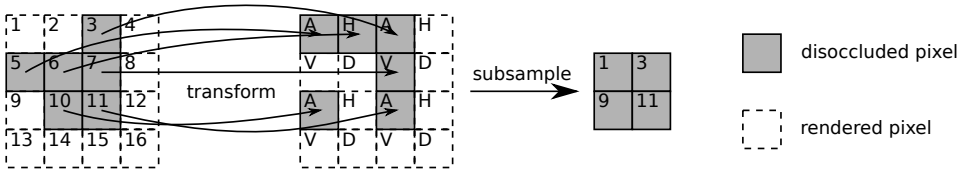
A single pixel is the simplest case. We pass on its value as approximation coefficient during downsampling. The motivation here as well as in the two other cases is that it seems arbitrary which pixel becomes the approximation for a complete block. All pixels contribute equally to the approximation for the non-adaptive transform. The detail coefficients are then determined such that they actually extract the horizontal, vertical, or diagonal detail.

Two pixels are found either at the left or right vertical edge, the top or bottom horizontal edge, or in a diagonal configuration in opposite corners of a block. A single 1D transform is applied. One of the two coefficients is passed on as the approximation such that the energy will get concentrated in the lowband. The other pixel is passed on as the vertical, horizontal, or diagonal detail, respectively. Thereby, the spatial correlation of the detail coefficients after decomposition should be better. This improves the performance of entropy coders which take spatial correlations into account.

In the case of three defined pixels, there will be one principal pixel which is next to the two others. The latter are in turn diagonal to each other. To reduce the bitrate, we want to concentrate the energy but also maintain sparsity in the coefficient domain, i.e., there should be only three coefficients after the



(a) First decomposition level.



(b) Second decomposition level.

Figure 3.5: Example of shape-adaptive transformation.

transform. Thus, we should use the approximation and two detail coefficients. By choosing the vertical and horizontal coefficients, the transform remains separable into a vertical and a horizontal transform pass. The transform is performed with the principal pixel as to-be low band. First, we transform the principal pixel and the one to the left or right to yield an intermediate low band and the horizontal detail. We also update the scale counters. Then, the transformation of the intermediate low band and the third pixel yields the vertical detail and the approximation. Again, we have to update scale counters.

This shape adaptation is efficiently implemented as permutations of the signal space before dimensionality reduction. We exploit that the output values outside the mask may take any value since they are discarded anyway. Additionally, it is possible to combine occlusion mask and scale counters. We set the scale counter to a special ignore value like Not-a-Number or -1 where indicated by the occlusion mask. When we apply the permutations to both the signal and the scale counter, we can keep track which values to discard and which to transform and encode without using additional memory. Meanwhile, we can still easily perform the subsampling since all approximations are found where they are expected in the non-adaptive case. This is required for additional levels of decomposition.

3.4.3 Entropy Coding

The arbitrary shape of the occlusion areas also introduces problems for entropy coding. The encoder should exploit spatial correlation of the coefficients to obtain a good performance. This becomes a problem at borders where some surrounding values are not available. For normal images, this occurs at the edges of the image. However, this may be solved, e.g., by appropriate block sizes, symmetric extensions etc. since the edges are straight lines. In case of shape adaptation, the problem becomes more complex.

For example, we have to derive the context of each coefficient for the EBCOT approach. However, values included in this calculation may be missing and we have to either modify the rules to select a certain context or introduce new contexts. The former has the risk of impairing the probability estimation when the context is misclassified. For the second solution, we have to estimate adaptively more probabilities. However, these estimates are expected to be poor since only few samples will be available when we

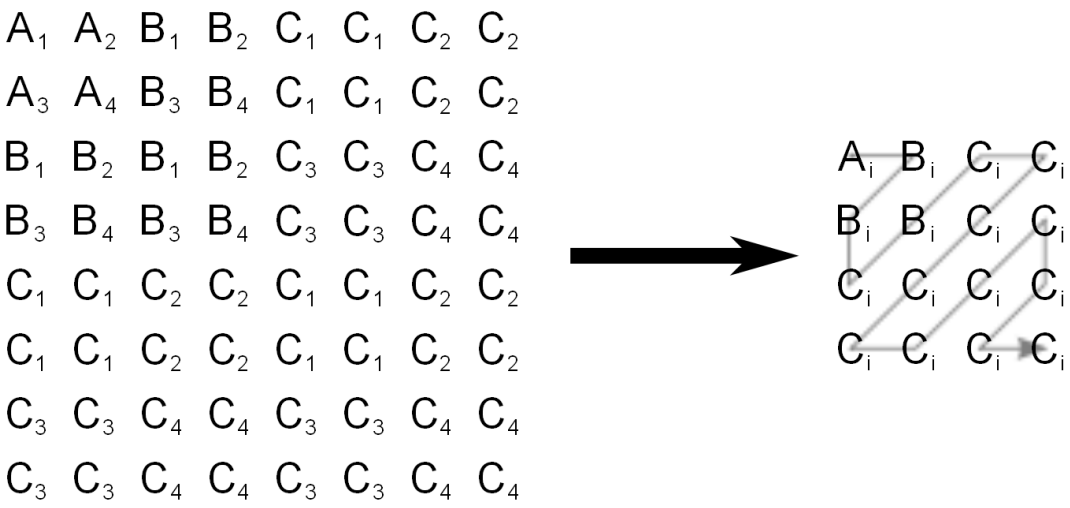


Figure 3.6: Conversion of wavelet decomposition (left) to zig-zag scans with $i \in 1, 2, 3, 4$. Equal letters belong to the same decomposition level, equal numbers to the same wavelet tree.

have too many contexts. A similar problem occurs for scattered small occlusion areas. In this case, only a few coefficients in each block are defined and thus few samples are available for probability estimation. Additionally, all algorithms in Section 2.6 introduce implementation and computation complexity to achieve scalability features. Since the TSDI generation is closed-loop, they are no benefit. In light of this, these algorithms seem unsuitable to encode a TSDI.

We propose an entropy coding scheme similar to the one of JPEG. The quantized coefficients are rearranged as depicted in Fig. 3.6 such that all belong to the same wavelet tree. We expect to create longer tails of zeros by zig-zag scanning this. The top left coefficient is the low band. It is least likely to be zero since the energy is concentrated there. The remaining coefficients are scanned roughly in order of their decomposition level. The lower the decomposition level, the less energy is present and we hope for early termination of the scan, i.e., all remaining coefficients equal to zero. We code this end with a special *end-of-scan* codeword. The scan is converted to run-length symbols which are then Huffman coded. Undefined coefficients are ignored. We use the occlusion mask to insert them at the respective positions in the decoder.

4 Multiview Texture Coding

To achieve efficient compression of the multiview texture, we have to reduce the redundancy. We use orthonormal wavelet transforms with subsequent quantization and entropy coding to do this. However, to obtain high performance, we have to compensate the parallax motion. With the approach outlined in Chapter 3, we can use the already decoded depth map for this compensation. At the encoder, the same information is available for free before occlusion detection in the TSDI generator loop.

We propose the structure in Fig. 4.1 for texture coding. The depth is sent as described above and converted to disparity at both ends. We use inverse warping and thus have the disparity for all pixels of the target view. Thus, there is no need of filtering which would disturb the visual quality. However, occlusion processing is required for reasons described in Sec. 4.3. Moreover, a structured decomposition which allows exploiting cross-subband correlation seems challenging. The flexibility of EBCOT circumvents this problem and we favor it instead of WDR, EZW, or SPIHT.

We transform the views with either a 1-hypothesis transform or 2-hypothesis transform to exploit the inter-view redundancy. For N views, this results in one low band having most of the energy and $N-1$ energy reduced high bands. All bands are then spatially decomposed and the result is quantized. We use a scalar uniform midthread quantizer with step size 1. Finally, EBCOT is used as entropy coder. The decoder does the inverse steps in reversed order. Without loss of generality, we only consider the processing of the intensity.

We use the JPC codec of JasPer [1]. It implements the EBCOT entropy coder including a built-in quantizer and MSE calculation. The output is a code stream including data, height, width, number of resolution levels, etc. according to the JPEG2000 standard, i.e., ISO/IEC 15444-1. While coping with the spatial decomposition is an intrinsic feature of EBCOT, we store each view decomposition as one components. While only three components are usually required to encode color still images, JPEG2000 allows up to 2^{14} components [2]. This leaves sufficient space for temporal decomposition and color processing.

JasPer has been included as an official reference implementation in the ISO/IEC 15444-5 standard. The JPEG2000 transforms have been stripped from version 1.900.1 and replaced by an interface to MATLAB. Additionally, we have added means to access the rate-distortion control.

4.1 1-Hypothesis Transform

Our 1-hypothesis transform is similar to the motion-compensated orthogonal transform in [12]. Instead of motion compensation, we use disparity compensation. In particular, the inverse warp will establish a relationship between each pixel in the target view and a pixel in the reference view. Each of these pairs is processed with an incremental transform T_κ as in [12]. The transform removes energy from the target view turning it into the view high band. The energy is concentrated in the reference view which thus becomes the view low band.

To be able to process more than 2 views, we use a cascade of this transform as depicted in Fig. 4.2. At the start, all pairs of input views are decomposed into an intermediate low band and a high band. Pairs of the intermediate low bands are further transformed in successive decomposition levels until only one low band is left.

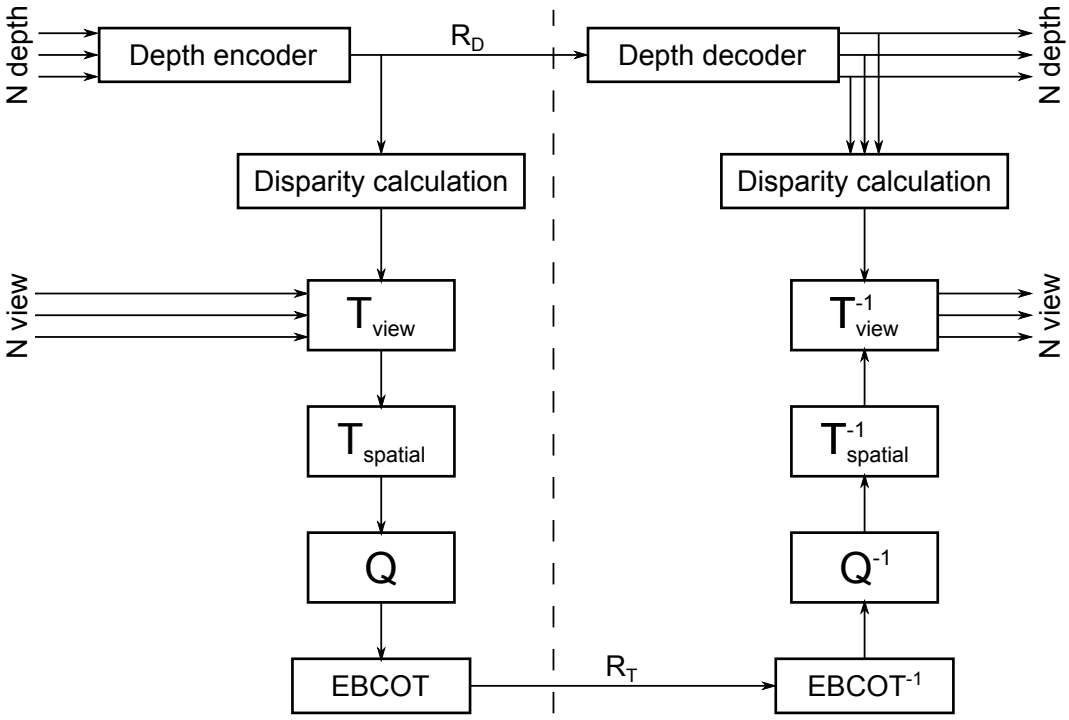


Figure 4.1: Proposed texture coding approach.

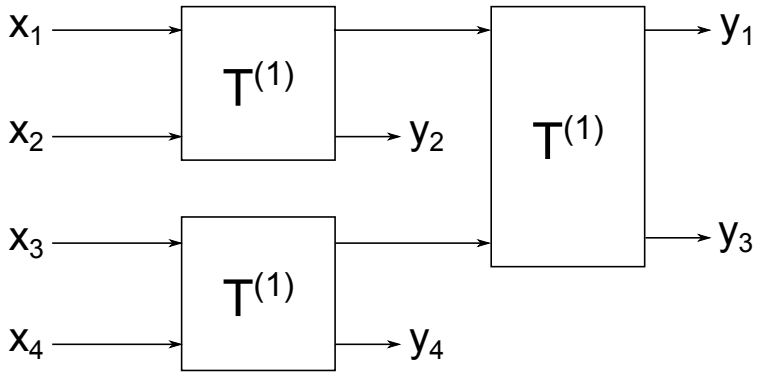


Figure 4.2: Cascade of 1-hypothesis transforms for $N = 4$.

The spatial decomposition uses the adaptive spatial transform of [12]. Incremental transforms of the same type as during view processing are used. However, they process pairs of horizontally neighboring pixels in the first run, then vertically neighboring pixels.

4.2 2-Hypothesis Transform

Since a linear combination of multiple motion-compensated signals improves the accuracy of motion compensation [13], we expect the equivalent for parallax motion compensation. A particular advantage of compensation by warping is that a single set of depth images will convey all relationships between the pixels in all views. Thus, we can use a 2-hypothesis transform without spending extra bits on a second set of side information. This is in general not the case for the motion vectors of motion compensation.

We replace the motion compensation in the double motion-compensated orthogonal transform [13] with disparity compensation to obtain a disparity-compensated 2-hypothesis transform. It may be used

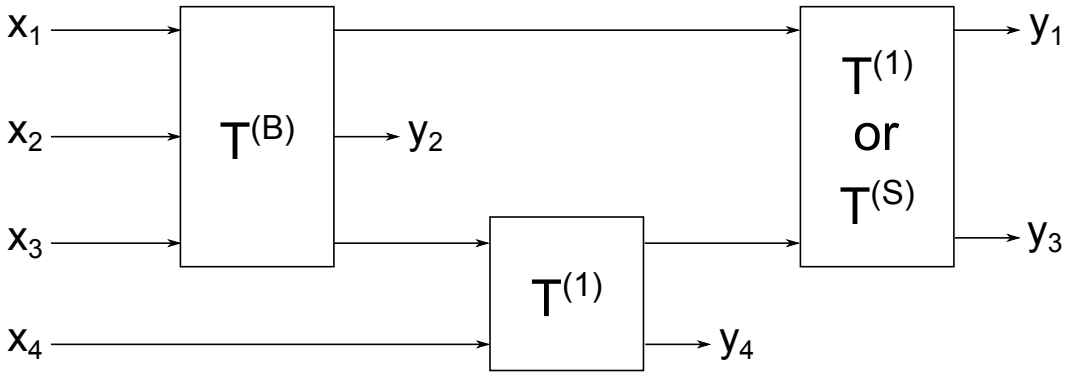


Figure 4.3: Cascade of bidirectional 2-hypothesis ($T^{(B)}$) or 2-hypothesis sub-pixel ($T^{(S)}$) transform and 1-hypothesis ($T^{(1)}$) transforms to process $N = 4$ views.

to do a bidirectional transform $T^{(B)}$ with two different reference views or to achieve sub-pixel accuracy (denoted $T^{(S)}$) with a single reference view.

If a view has a left and a right neighbor, we can find a corresponding pixel in both neighbors for most pixels of said view. The exception are pixels which are only visible in either of the two neighbors (or even none) due to occlusions. They are detected using the occlusion mask and processed with the 1-hypothesis transform. Besides the dense disparity field, inverse warping has a second advantage in this case. The more complex reverse projection has to be performed just once. The same 3D world point is then projected to the left and right views. The respective pixels in each view are used as the references for the transform. Thus, we obtain two intermediate low bands and one high band. To get a single view low band, the last step must be a 1-hypothesis transform and the resulting cascade for $N = 4$ is depicted in Fig. 4.3.

The warp results virtually never point to integer pixel positions. This motivates sub-pixel accuracy where we use the two pixels closest to the warp result as the two hypotheses. For rectified views of linear camera arrays, these are $\lfloor u \rfloor$ and $\lceil u \rceil$. However, we use two hypotheses only for $0.25 < |u - \lfloor u \rfloor| < 0.75$ since the transform tries to equally distribute the energy between the references. If the warp result is closer than this threshold to either of the references, it seems more reasonable to send as much energy as possible to that single reference. However, none of the choices for the transform is guaranteed to improve performance. For motion compensation, this is typically solved by determining the better choice and explicitly signaling it. Although a block based decision suggests itself, we intend to save the computation effort and the bandwidth.

4.3 Shape Adaptation

Because the decoder filled in the auxiliary information, all pixels of the decoded depth maps are defined. The warping calculation is also successful for all pixels. However, the proposed pixel may belong to a different 3D world point if an occlusion happened or may lie outside the view's image boundary. In the former case, a relationship between the pixels is claimed but there actually is no correlation. In the second case, there is no reference unless we enforce one by, e.g., clipping to the image boundary.

The naive solutions are no transformation of the affected pixel at all or complete processing, i.e., the transformation of the uncorrelated pixels. The former will leave the untouched pixel values and thus the energy in the high band. Because of this increased energy, a spatial decomposition of the high bands is necessary. The other approach ignores the z-buffer and hence the possibility that there can be additional invisible layers in a view.

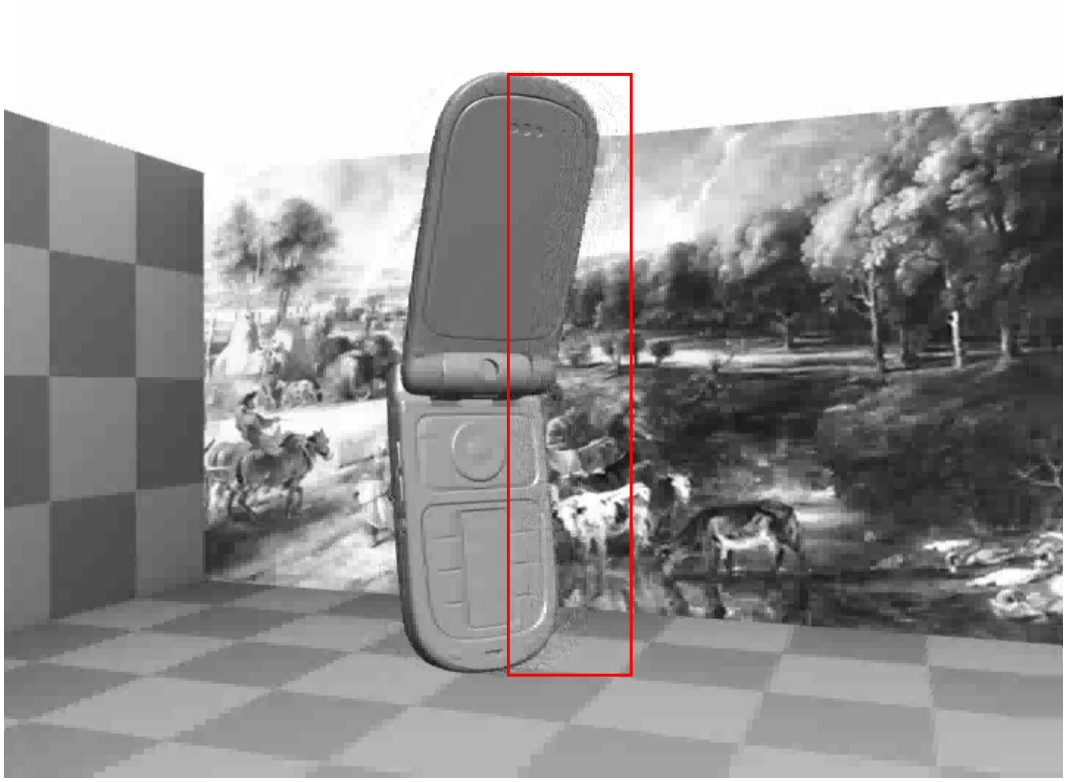


Figure 4.4: Large-scale distortion caused by z-buffer-free transformation of occluded pixels.

Without transformation, the distorted area is much smaller than for the complete transformation depicted in Fig. 4.4. Nonetheless, there are visible artifacts at the border of the occlusion area as shown in Fig. 4.5. They occur because of different signal properties when view-transformed, energy-removed pixels are spatially decomposed together with unprocessed high-energy pixels. We conclude that we should use a shape-adaptive transform.

A more complex shape-adaptive transform than in Sec. 3.4.2 is required. In contrast to the depth, the view high band of the texture is not sparse since it has to model non-Lambertian surfaces, semi-transparent objects, and so on. On the other hand, the dense high band allows the use of a fairly regular transform once the energy is removed. To avoid trouble when quantizing, we must achieve the removal in orthonormal fashion. The energy should also be concentrated to improve the efficiency of entropy coding. To achieve these goals, we use the 4-pixel blocks of the regular transform as the guideline similar to Sec. 3.4.2. The blocks are divided into the three subsets energy-removed, high-energy and mixed.

The basic principle is to turn the mixed into energy-removed blocks by moving the energy to the closest high-energy block. We can achieve this when we apply transformation with appropriately chosen reference pixels due to the spatial correlation. When using the orthogonal transform step and appropriate scale counters, the signal space is also preserved. Since we process values of either high bands or original pixels, the scale counters are initialized with zero. An example of an occlusion mask is given in Fig. 4.6. Black pixels had their energy removed by the view transform. The occlusion areas are indicated by colored pixels where the color indicates the mode as outlined below.

Nearby pixels are usually more similar than those farther apart. Hence, we try to process neighboring pixels first. Therefore, we search for complete blocks which are horizontally, vertically, or diagonally misaligned, i.e., shifted by one pixel in x-direction, y-direction, or both directions, respectively. They are indicated by the blue and purple pixels in Fig. 4.6. Particularly for multiple resolution levels, we

Figure 4.5: Visible artifacts at the border of untransformed occlusion areas.

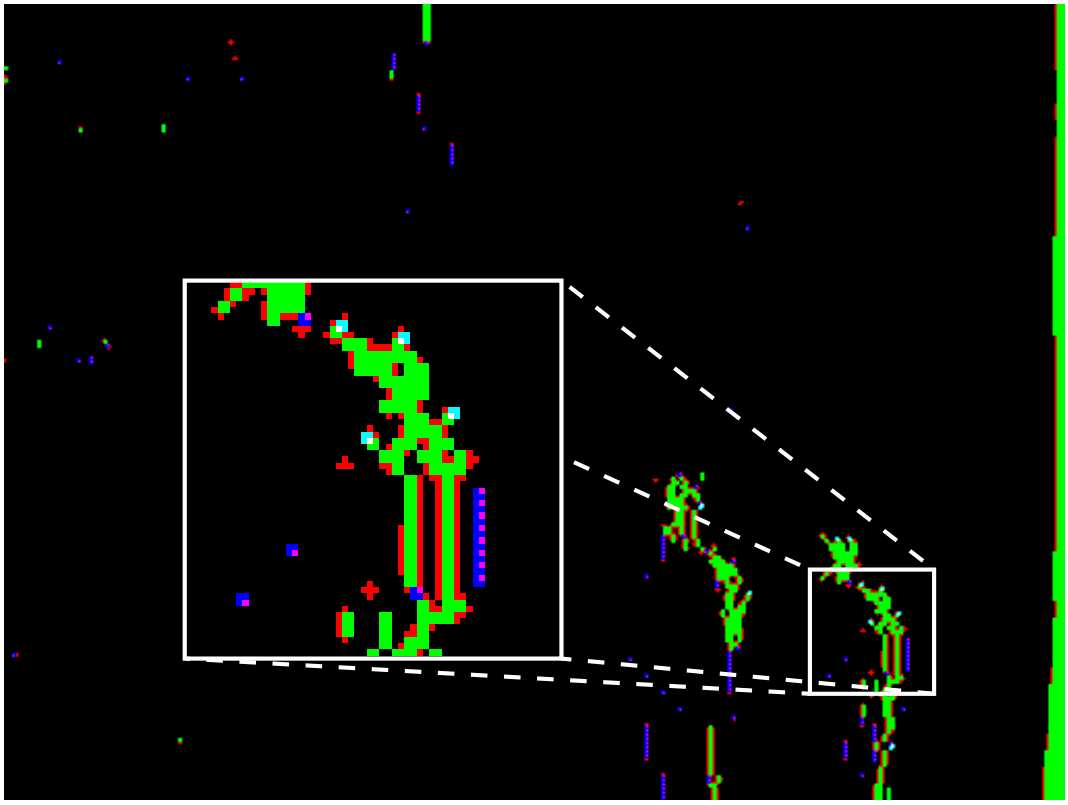


Figure 4.6: Example of a mask for occlusion processing (black: energy-removed pixel, green: aligned high-energy block, blue/purple: shifted high-energy block, cyan/white: overlapping blocks coinciding at the white pixel, red or purple: transformed with closest high-energy block).

improve the correlation of the subsampled bands and thus compression if we keep the approximation and the three detail coefficients of a shifted block at the proper positions of the respective aligned blocks. An aligned block containing a purple pixel would have its approximation coefficient there. Analogous, the blue pixels are detail coefficients in the respective blocks. With appropriate permutations before and after the orthogonal transform step, we achieve that the purple pixel really is the approximation and the detail ends up at the correct position.

A complete high-energy block is sometimes only found if we allow two blocks to overlap. They are marked by cyan and white pixels in Fig. 4.6 where the two involved blocks coincide at the white pixel. We can permute and transform one of the blocks so that the white pixel becomes the approximation. When we proceed and transform the other, we are left with just one high-energy coefficient that requires further processing. Frequently, there is an aligned high-energy block involved and we are done with the clean-up operations.

The scattered red pixels in Fig. 4.6 also have high energy but could not be assigned to any block. They are transformed with the closest pixel of a high-energy block. Actually, they are processed before the above steps so that we can also use shifted blocks as reference which may be closer than an aligned block. Similarly, some pixels are left with high energy after processing the shifted blocks. They are marked purple in Fig. 4.6 and we use the closest pixel of an aligned high-energy block for the transform. Finally, we separately transform the sets of energy-removed and high-energy blocks with the adaptive spatial transform [12].

5 Optimal Rate Allocation

Since the available bandwidth R_{max} is limited for many applications, we need means to control the bitrate. At the same time, we want to obtain the best possible quality for a given rate. Rate-distortion optimization achieves both objectives.

While EBCOT can perform rate-distortion optimization in the context of a single still image, it assumes that the transform requires no side information, i.e., has a predefined structure as, e.g., Mallat or Spacl structure in [32]. However, our system proposed in Fig. 3.1 uses an adaptive transform and consequentially has the two rates R_D and R_T for the depth map and the texture, respectively. The quality of the depth information influences the output quality due to its use as side information during texture processing. The distortion metric is then $D = \mathrm{f}(R_T, R_D)$ and we should also vary R_D.

Analogous to still images, masking effects for binocular vision are known [3] and suggest an appropriate visual distortion metric. However, we face the problem that no such measure seems to be readily available. Since we lack resources to conduct the subjective evaluation, we stick to MSE for this work. Hence, we can use the same assumptions as outlined in Sec. 2.6 and also the generalized Lagrange multiplier method [9].

The relation between R_D and D involves a complex combination of (2.5), (2.7), and Huffman codes. A analytical expression seems intractable and in fact unnecessary since the generalized Lagrange multiplier method is applicable to discrete sets [9]. Additionally, a similar restriction exists anyway because we may only assign integer bits in general and they are our single resource to dispense. The particular implementation is restricted to bytes, i.e., the code words grow in multiples of 8 bits. Thus, we choose a set of K quantizer step sizes for the TSDI for simplicity. The produced TSDIs result in the rates R_D^k where $k = 1, ..., K$. Since we construct the texture transform using the depth map, we have to perform the texture transform for each quantizer step size using the respective depth maps. R_D is implied to be constant during the subsequent bitstream generation and distortion calculation. The optimal tradeoff between the two rates is then the solution to

$$\min_{\text{all } k, \text{all } R_T} D(R_T, R_D^k) \tag{5.1}$$

subject to

$$R_T + R_D^k \leq R_{max}. \tag{5.2}$$

Owing to the additivity assumption, (5.1) is the same as

$$\operatorname*{argmin}_{\text{all } k, \text{all } R_T} \sum_{\text{all } i} D_i(R_T, R_D^k) \tag{5.3}$$

where D_i is as in Sec. 2.6. The texture rate is analogous to [32] the sum

$$R_T = \sum_{\text{all } i} n_i \tag{5.4}$$

where n_i is the truncation point of the respective block's codeword. By Everett's main theorem [9], the optimum for any desired so-called *Lagrange multiplier* $\lambda \geq 0$ is

$$\underset{\text{all } k,\text{all } n_i(\lambda)}{\operatorname{argmin}} \sum_{\text{all } i} \left[D_i(n_i(\lambda), R_D^k) + \lambda n_i(\lambda) \right] + \lambda R_D^k. \tag{5.5}$$

Since $\lambda R_D^k \geq 0$ and is thus just an offset for constant k, finding the minimum decomposes to

$$\delta^k(\lambda) = \min_{\text{all } n_i(\lambda)} D_i(n_i(\lambda), R_D^k) + \lambda n_i(\lambda) \quad \text{for each } i \text{ and each } k \in 1, ..., K \tag{5.6}$$

and subsequently

$$\underset{k}{\operatorname{argmin}}\ \delta^k(\lambda) + \lambda R_D^k \tag{5.7}$$

to find the final truncation points for each block and the quantizer step size of the TSDI.

The solution to (5.6) is identical to the one in [32] and hence obviously inherits its features. They include that it is rather efficient since it is sufficient to obtain the convex envelope of the truncation points only once. Without this, the search for the minima (5.6) would be to complex. Additionally, this allows to rerun the rate allocation later on if one stores the produced information about the code blocks. The latter seems reasonable since this data is small for still images. Since we also have to cope with R_D and each transform yields different EBCOT codewords, we need to store not only the block information but multiple sets of the entire data. However, the epsilon theorem [9] suggests that we can find almost optimal allocation even if we store only few data sets. Thereby, we achieve SNR scalability.

We sweep through λ as proposed by Everett to obtain multiple solutions. Using a set of λ's, we find the respective texture rates which add up with the depth map's rate to the surface in Fig. 5.1. The optimal points for this example are marked by dots. The multiplex of the two bitstreams results in a total bitrate $R_{tot} = R_T + R_D$ and a rate-distortion curve like in Fig. 5.2.

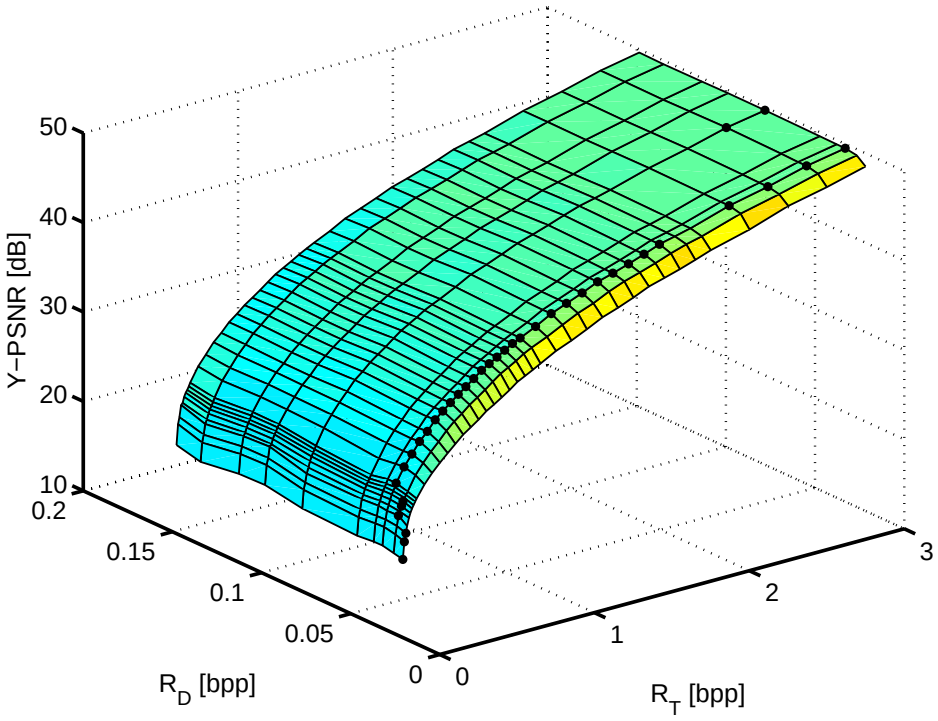


Figure 5.1: Luminance PSNR as function of texture rate R_T and depth map rate R_D.

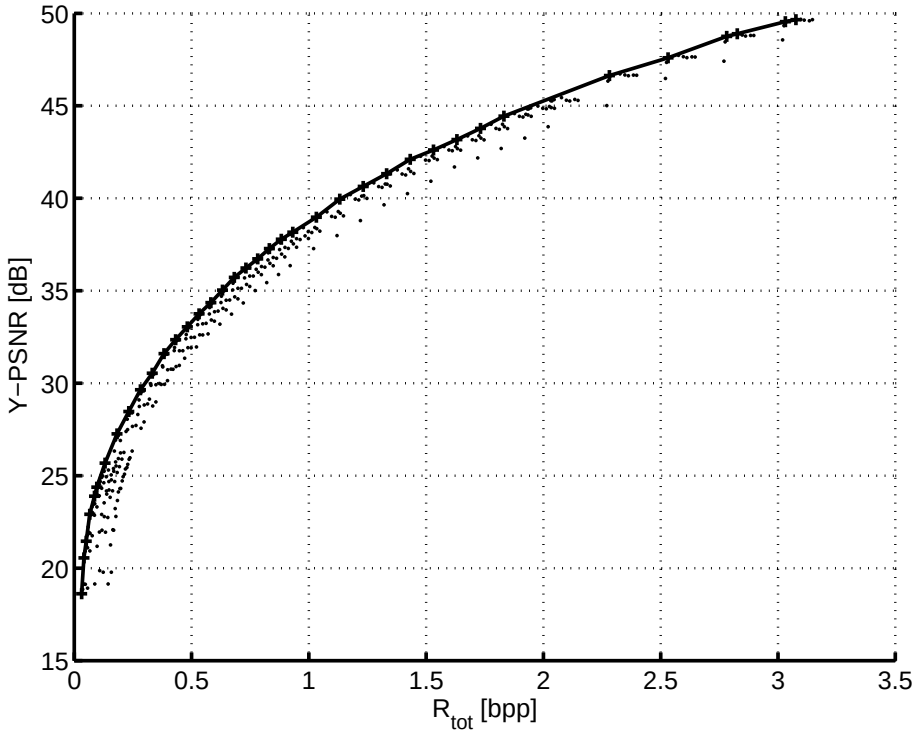


Figure 5.2: Optimal (+) and suboptimal (·) distortion for respective total bitrate.

6 Results

The MPEG test sequences *Lovebird 1*, *Newspaper*, and *Mobile* are used to test the proposed algorithms. The two former are rectified and calibrated images of a linear camera array. They are provided with calibration data and depth maps obtained with DERS [31]. *Mobile* is a computer generated sequence and thus has better depth maps. Its challenge is the shape edge between the rotating mobile in the foreground and the background. *Newspaper* has the most complex scenery of the three sequences. A person is passing between persons and objects in the foreground and background objects. Thus, there are many transitions in the depth. The two other sequences both show a static background and some motion in the foreground.

We use the first 50 frames of each sequence and the views 4, 6, 8, and 10 of *Lovebird 1* and 3 to 6 of *Mobile*. *Newspaper* is used with a narrow baseline (views 2 to 5) and a wide baseline (views 2, 3, 5, and 6). *Lovebird 1* and *Newspaper* are downsampled from 1024×768 pixel to 512×384 to reduce computation time and because of memory constraints. *Mobile* is left at its original resolution of 720×528. All sequences are YUV files with 8 bit per channel and 4:2:0 chroma subsampling. However, we ignore the color to further reduce time and memory demands. The additive distortion metric makes the extension to color simple. We also ignore the temporal correlation and perform only view decomposition since the typically stronger temporal correlations [5] might mask results otherwise.

6.1 Depth Image Coding

In Fig. 6.1, we evaluate the effectiveness of the dimensionality reduction of the TSDI using the quotient

$$k = \frac{\text{coded coefficients}}{\text{total number of pixels}} = \frac{HW + M}{HWN} \tag{6.1}$$

where H, W denote the width and height of the images, N the number of views, and M the number of auxiliary pixels. The product HW in the nominator accounts for the RDI which is always coded entirely. Hence, the minimal value of k is 0.25. k is approximately constant for sufficiently high bitrate. This indicates that we are only coding almost the minimum of auxiliary values if the RDI quality is good enough. For low bitrates, the approach randomly either breaks down and produces larger holes or illicitly removes fine details. The number of coded pixels increases or decreases, respectively. For *Mobile*, the individual k for each frame is peculiar as depicted in Fig. 6.2. Since the mobile spins around its own axis, the occlusion area decreases for the considered range of frames. Being computer generated, the sequence also is noise free and in total somewhat degenerated.

For TSDIs, the preprocessing and the dimensionality reduction prevent perfect reconstruction. This is illustrated by the minimal reconstruction error in Fig. 6.3. It is derived by performing the generation process as in Sec. 3.2 but without quantization and compression of the transform coefficients. However, we compare the initial measurements and final result. Hence, we see a combination of desired measurement error reduction and unwanted coding distortion. Caused by any of the two, the MSE of the depth values shows the desired initial rapid decay at low rates which suggests good compression. Its function over bitrate is approximately convex.

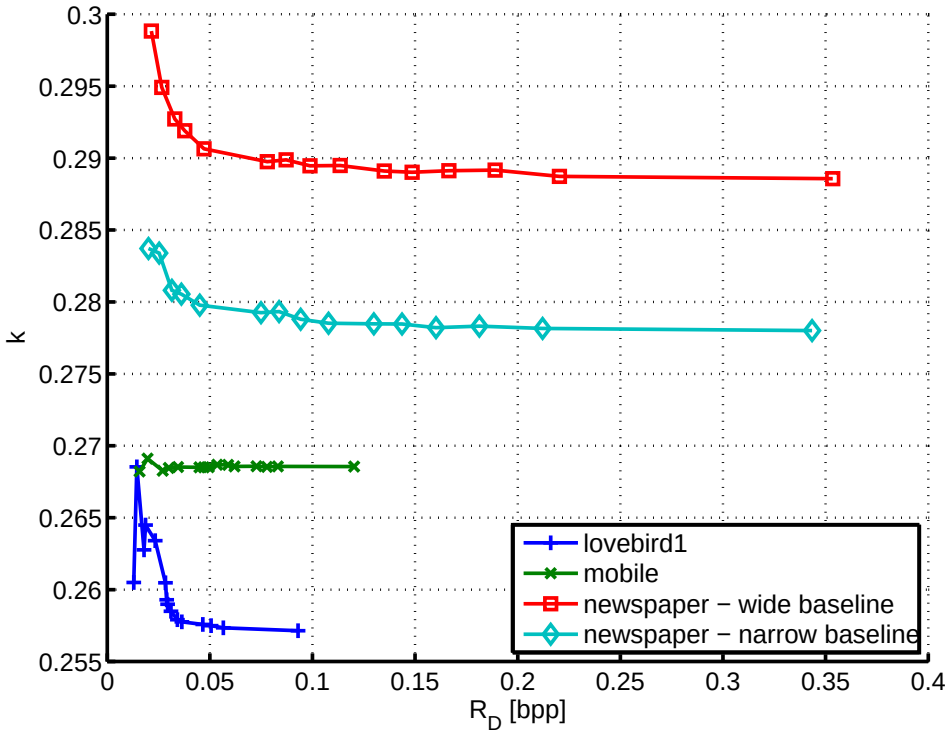


Figure 6.1: Fraction of actually coded pixels (representing the dimensionality reduction) over rate.

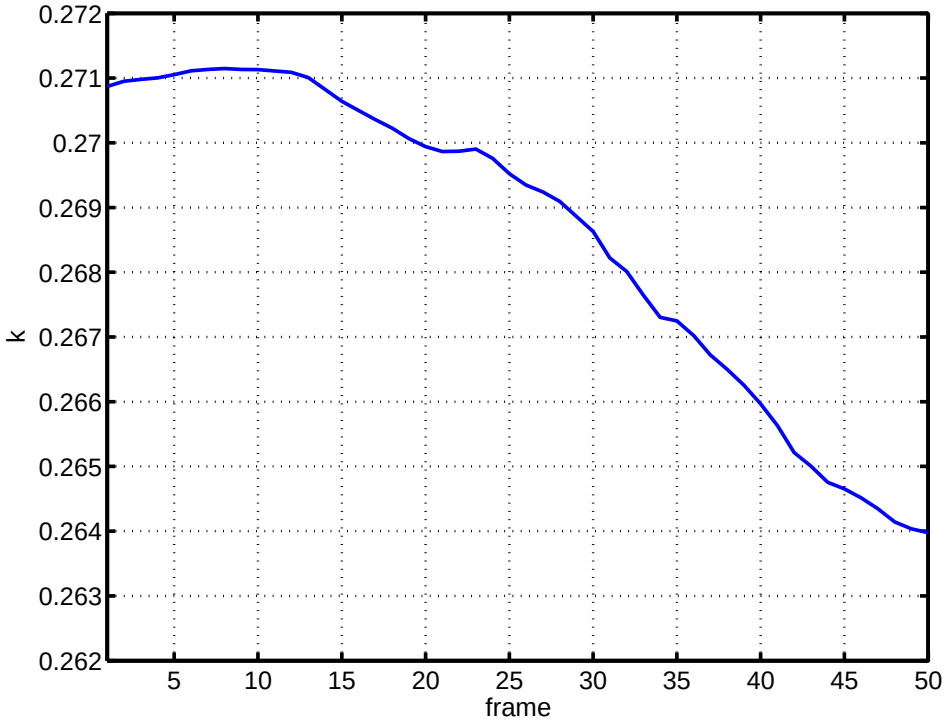


Figure 6.2: Percentage of coded pixels for each depth map of *Mobile*.

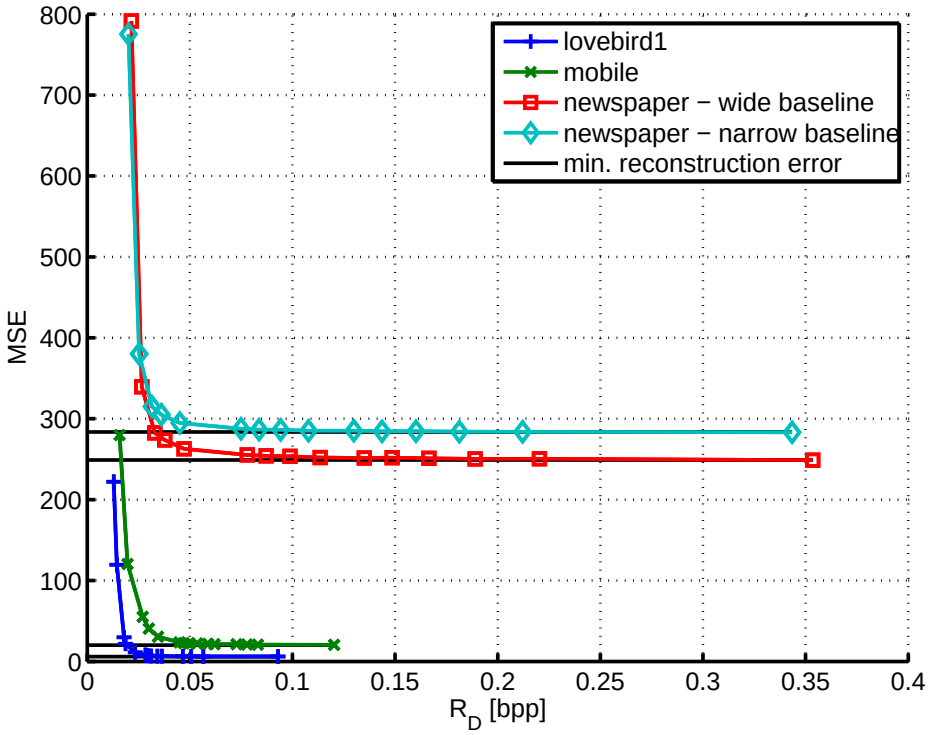


Figure 6.3: MSE of depth values over bitrate of entire depth map.

If we only consider the actually coded coefficients, a result as in Fig. 6.4 is obtained. The bitrate is now the total number of bits spend divided by the number of coded coefficients. Since we used an orthogonal transform, the PSNR of the coefficients and depth values are related. The result is even more similar to what is typically expected for image coding and converges towards 6dB per bit per pixel. An interesting detail of *Newspaper* is the similarity of the PSNR for the same quantizer step size despite different baseline configurations. Meanwhile, the minimal reconstruction error in Fig. 6.3 is not independent of the baseline. Its reduction for wider baselines seems plausible since more pixels appear at the edge of the image than for a narrow baseline. This is illustrated by the black area at the left edge of view 3 and 4 in Fig. 3.2 and reflected in smaller k for the narrow baseline in Fig. 6.1. They are intra-coded and hence have in general smaller error.

6.2 Texture Coding

The performance is evaluated using the distortion in the luminance channel. In particular, we use the PSNR where

$$\mathrm{PSNR} = 10 \log \left(\frac{255^2}{MSE} \right). \tag{6.2}$$

Figures 6.5, 6.6, and 6.7 show the PSNR vs. the total bitrate for *Mobile*, *Lovebird 1*, and *Newspaper* sequences, respectively. The latter is used only with wide baseline. We compare our shape-adaptive transform to a reference where the disocclusion areas are blindly transformed together with the rest of the view. In addition, we add sub-pixel accuracy to the shape-adaptive transform. All resulting PSNR curves are virtually cocave.

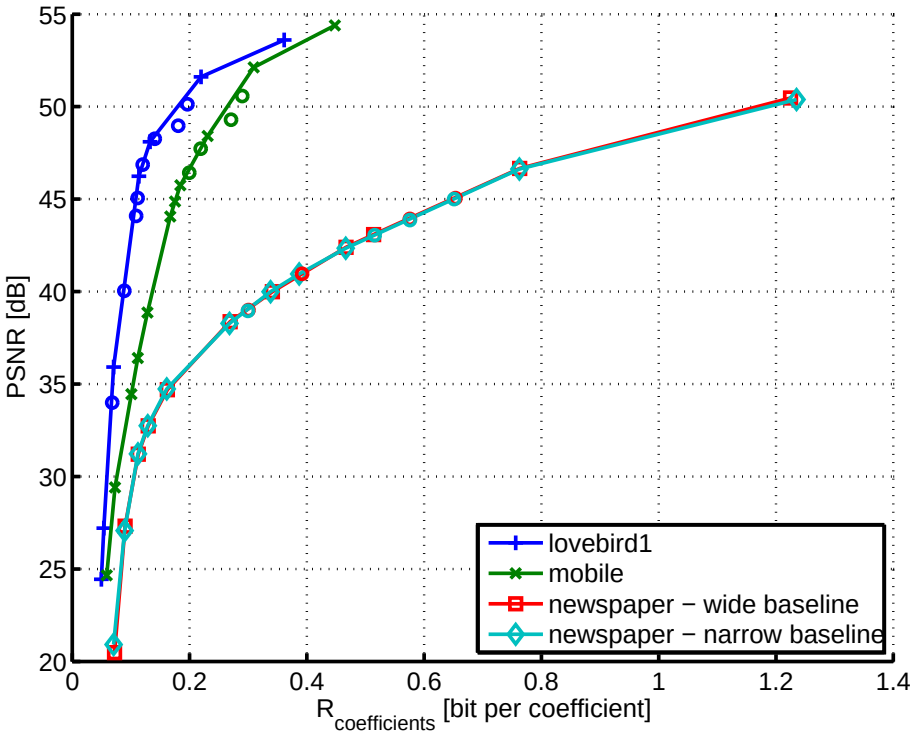


Figure 6.4: PSNR of coded coefficients over their bitrate (○ not on convex hull).

The shape-adaptive transform proposed in Sec. 4.3 yields the improvement depicted in Fig. 6.8. The apparent degradation of *Lovebird 1* between 38dB and 48dB is at closer look also an improvement. We argue that about 0.3dB improvement in the disocclusion area outweighs slight degradation otherwise in term of visual perception. In the overall PSNR, the relatively few occluded pixels are masked. Although this effect does not become evident if all parts are enhanced, it is in fact present for all qualities. The improvement of the occlusion area is in general significantly larger, e.g., about 1.5dB vs. in total 0.6dB for *Lovebird 1* at 30dB. This is also reflected in the energy compaction in the lowbands of the spatial decompositions of the view highbands. It increases from about 34% to about 57% with shape adaptation for 6.6.

Sub-pixel accuracy does not improve the PSNR compared to shape-adaptive transformation as depicted in Fig. 6.9. The exception is *Mobile* being up to 1.2dB better. We conclude that superior quality of the depth map is required for our blind approach and mode decision is inevitable otherwise.

Fig. 6.10 shows the optimal bitrate of the depth map depending on the texture rate. The *Newspaper* curve corresponds to the dots in Fig. 5.1. Finding a quantizer step for a particular bitrate is sometimes difficult since the MSE is soon largely flat. Moreover, the interesting operation range is usually considered to be about 25dB to 45dB. Hence, we ignore the sawtooth for *Newspaper* to the right. An additional R_D may yield a steady rate-distortion decision but the small PSNR improvement appears unreasonably complex for practical applications. Since the algorithm only takes few of the offered values for R_D and R_D is constant on intervals, we conclude that the approach to SNR scalability in Chapter 5 is indeed reasonable.

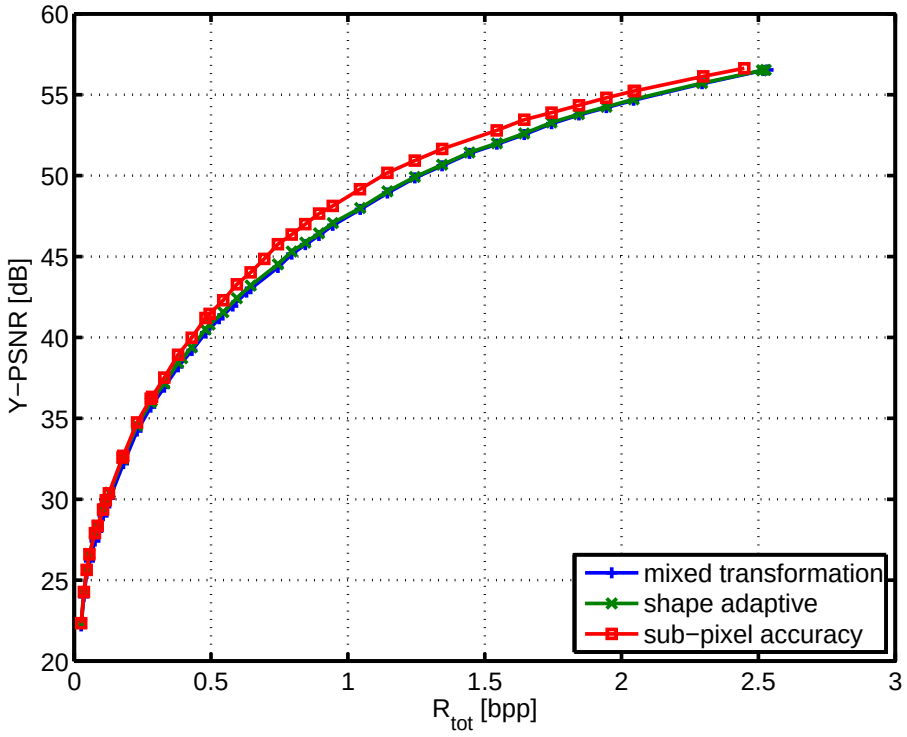


Figure 6.5: Luminance PSNR over bitrate for reference and proposed occlusion processing as well as half-pel accurate transformation (*Mobile* sequence).

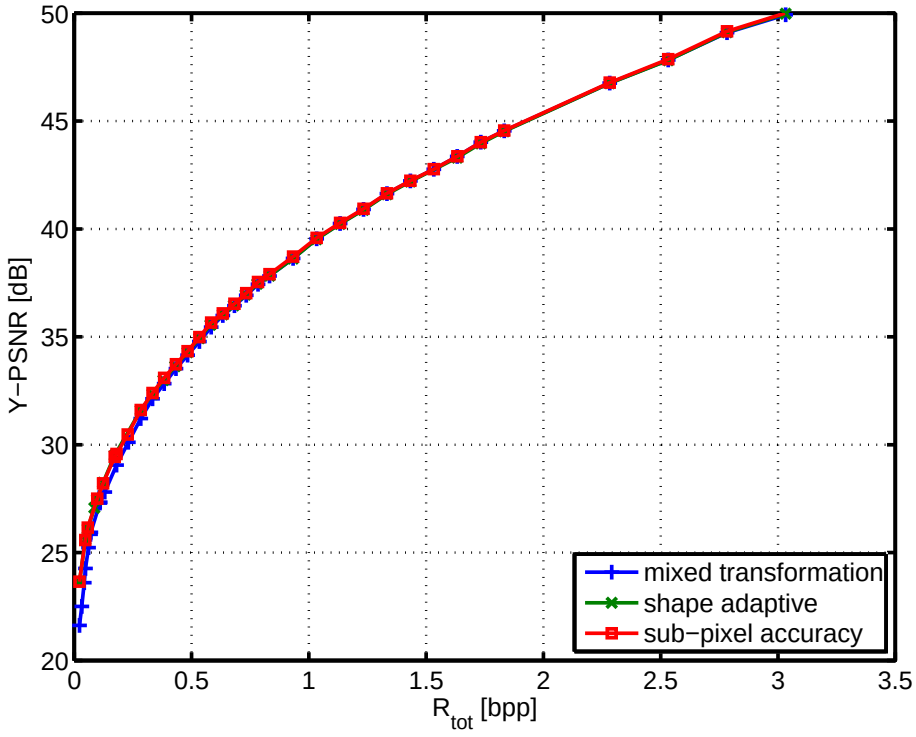


Figure 6.6: Luminance PSNR over bitrate for reference and proposed occlusion processing as well as half-pel accurate transformation (*Lovebird 1* sequence).

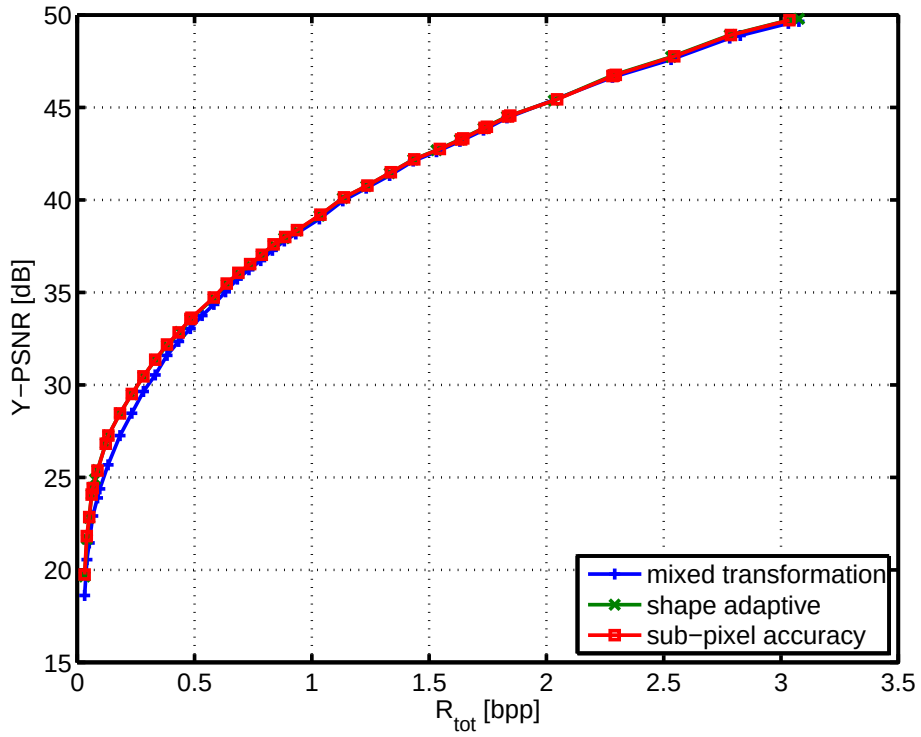


Figure 6.7: Luminance PSNR over bitrate for reference and proposed occlusion processing as well as half-pel accurate transformation (*Newspaper* sequence).

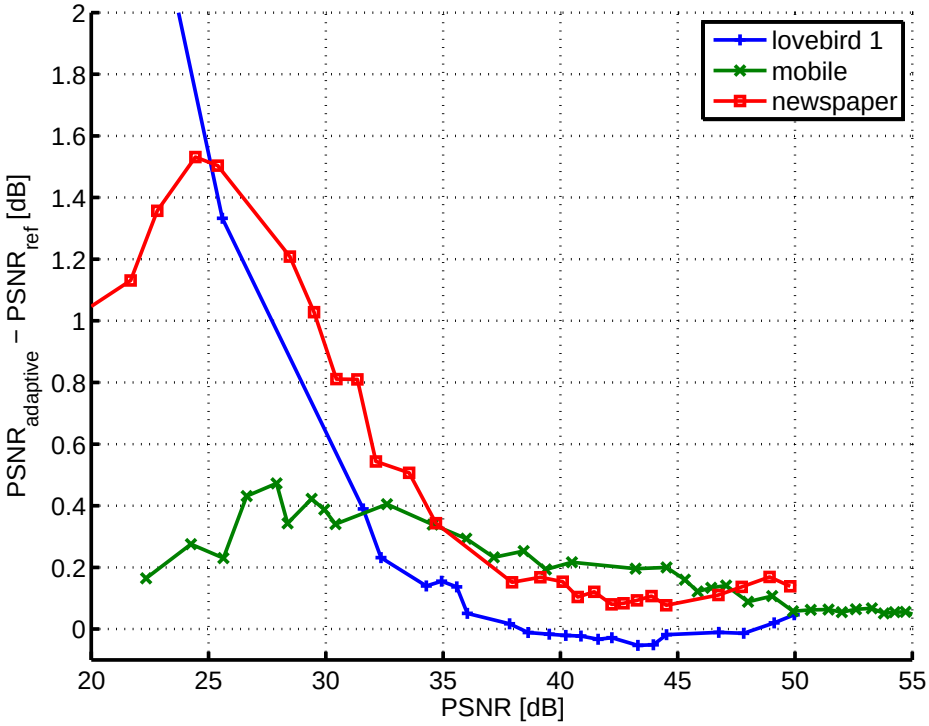


Figure 6.8: Improvement due to shape-adaptive transform over PSNR.

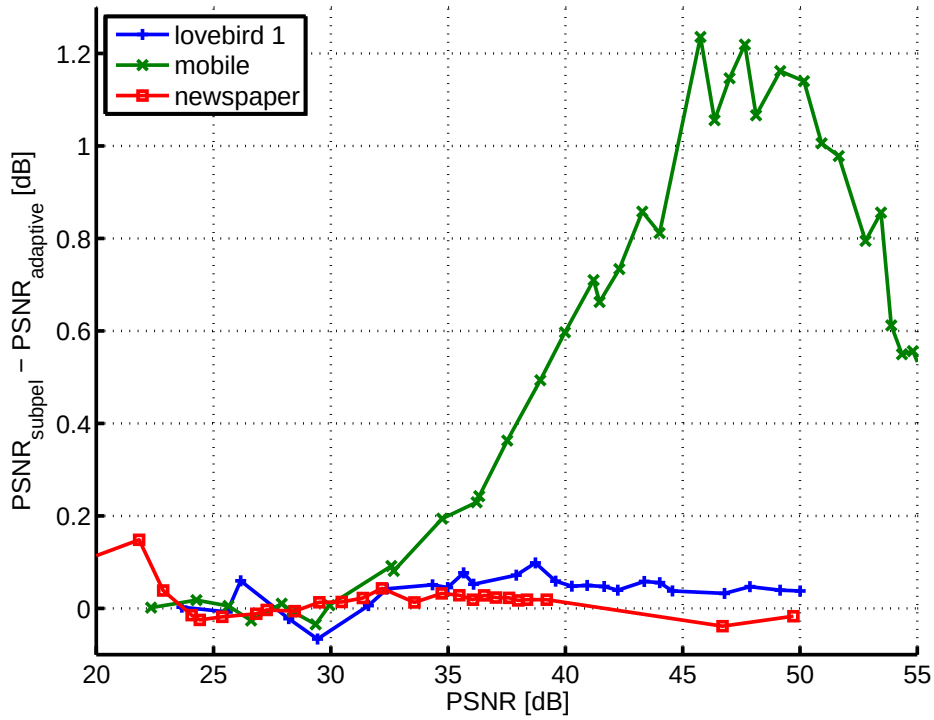


Figure 6.9: Change between sub-pixel accuracy and shape-adaptive transform over PSNR.

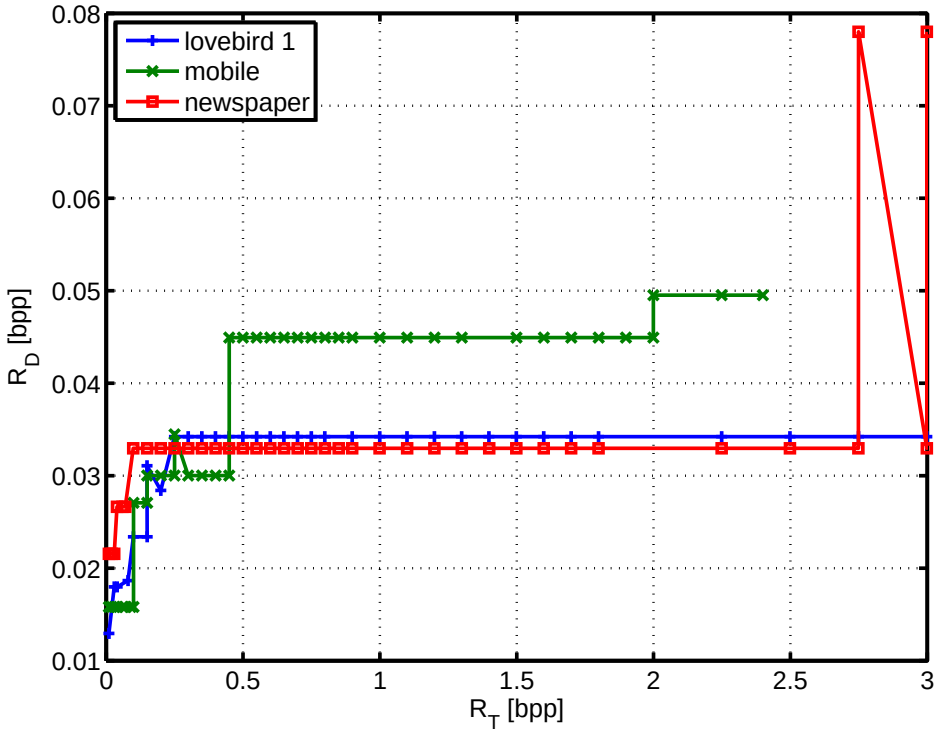


Figure 6.10: Bitrate of depth map as function of texture rate.

7 Conclusions

Emerging technologies like 3D-TV and free-viewpoint TV require image sequences from multiple viewpoints. Captured by camera arrays, this multiview imagery produces a tremendous amount of data. On the other hand, this data is highly redundant and hence eligible for efficient compression. The diverse applications and display technologies demand adaptation to the specific conditions. Adding relatively little auxiliary information to the images enables powerful and high-quality algorithms that can perform such adaptation. The most common approaches are scene depth information for each pixel and post-processing using depth image based rendering (DIBR).

We have investigated the properties of depth images and propose tree-structured depth images as a novel datastructure to represent them. TSDIs exploit the consistency of depth values, i.e., that linked pixels have in theory the same value. This enables dimensionality reduction of the signal space. The compression of TSDIs is shown to be self-sustained, i.e., without the usual need of side information.

For texture processing, the class of motion-compensated orthogonal transforms [12, 13] has been extended by a new member using DIBR for disparity compensation. When combined with state-of-the-art entropy coding using the EBCOT algorithm, the resulting open-loop subband coding enables SNR scalability and potentially spatio-temporal scalability. In the process, an interface between MATLAB and the JasPer [1] open-source reference implementation of JPEG2000 has been developed as a MEX-file. We believe its speed boost can prove valuable for related projects in the field.

A common problem with DIBR are disocclusions, i.e., that a previously covered area becomes visible in a rendered view. For both depth map and texture processing, we have presented orthogonal processing that yields better energy compaction in the disoccluded areas.

The combination of the proposals yields a rate-distortion optimal codec for multiview plus depth imagery. TSDIs are used to encode the depth images. The result is used as side information in the texture processing. As the previous proposals rather attach a depth coder to a texture coder, this is an alternate system structure. It can use depth information to improve the efficincy of texture coding but meanwhile exploits the correlation between them. Thereby, we achieve a compact joint representation. Also being orthogonal, it enables efficient algorithms for optimal rate allocation.

Future work consists of improving the energy compaction of the transforms. Motion compensation, well-known for exploiting temporal redundancy in singleview video, and block-based selection of different transforms suggest themselves. Since EBCOT is designed for still images, it should be further optimized for the multiview imagery.

Bibliography

[1] M. D. Adams. The JasPer project home page. `http://www.ece.uvic.ca/~mdadams/jasper/`.

[2] M. D. Adams. The JPEG-2000 still image compression standard. Technical Report ISO/IEC JTC 1/SC 29/WG 1 N 2412, Dept. of Electrical and Computer Engineering, University of Victoria, Canada, Dec. 2002.

[3] P. Aflaki, M. Hannuksela, J. Häkkinen, P. Lindroos, and M. Gabbouj. Impact of downsampling ratio in mixed-resolution stereoscopic video. In *3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2010*, pages 1–4, June 2010.

[4] A. Akbari, N. Canagarajah, D. Redmill, and D. Bull. Disparity compensated view filtering wavelet based multiview image code using lagrangian optimization. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 309 –312, May 2008.

[5] N. Anantrasirichai, C. Canagarajah, D. Redmill, and D. Bull. In-band disparity compensation for multiview image compression and view synthesis. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(4):473–484, Apr. 2010.

[6] C.-L. Chang, X. Zhu, P. Ramanathan, and B. Girod. Light field compression using disparity-compensated lifting and shape adaptation. *IEEE Transactions on Image Processing*, 15(4):793 –806, Apr. 2006.

[7] Digital Cinema Initiatives. *Digital Cinema System Specification Version 1.2*, Mar. 2008.

[8] E. Ekmekcioglu, S. Worrall, and A. Kondoz. Bit-rate adaptive downsampling for the coding of multi-view video with depth information. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 137–140, May 2008.

[9] H. Everett. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Oper. Res.*, 11:399–417, 1963.

[10] C. Fehn. A 3D-TV system based on video plus depth information. In *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1529–1533 Vol.2, Nov. 2003.

[11] M. Flierl. Adaptive spatialwavelets for motion-compensated orthogonal video transforms. In *16th IEEE International Conference on Image Processing (ICIP)*, pages 1045–1048, Nov. 2009.

[12] M. Flierl and B. Girod. A motion-compensated orthogonal transform with energy-concentration constraint. In *Proc. of the IEEE International Workshop on Multimedia Signal Processing*, pages 391–394, Oct. 2006.

[13] M. Flierl and B. Girod. A double motion-compensated orthogonal transform with energy concentration constraint. In *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, 2007.

[14] M. Flierl and B. Girod. Multiview video compression. *IEEE Signal Processing Magazine*, 24(6):66–76, Nov. 2007.

[15] J.-U. Garbas and A. Kaup. Enhancing coding efficiency in spatial scalable multiview video coding with wavelets. In *IEEE International Workshop on Multimedia Signal Processing*, pages 1–6, Oct. 2009.

[16] I. Ideses, L. Yaroslavsky, I. Amit, and B. Fishbain. Depth map quantization - How much is sufficient? In *3DTV Conference, 2007*, pages 1–4, May 2007.

[17] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger. Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability. *Signal Processing: Image Communication*, 22(2):217–234, 2007. Special issue on three-dimensional video and television.

[18] W. Kumwilaisak and P. Lasang. Optimal subband bit allocation for multi-view image coding with disparity compensated wavelet lifting technique. In *Proceedings of Picture Coding Symposium 2007*, 2007.

[19] L. McMillan Jr. *An Image-Based Approach To Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina, Chapel Hill, USA, 1997.

[20] Y. Mori, N. Fukushima, T. Fujii, and M. Tanimoto. View generation with 3d warping using depth information for ftv. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2008*, pages 229 –232, May 2008.

[21] Y. Morvan. *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2009.

[22] H. Oh and Y.-S. Ho. H.264-based depth map sequence coding using motion information of corresponding texture video. In *Proceedings of Pacific-Rim Symposium on Image and Video Technology*, pages 898–907, 2006.

[23] J. H. Park and H. W. Park. A mesh-based disparity representation method for view interpolation and stereo image compression. *IEEE Transactions on Image Processing*, 15(7):1751–1762, July 2006.

[24] P. Rana and M. Flierl. Depth consistency testing for improved view interpolation. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 384–389, Oct. 2010.

[25] A. Said and W. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243 –250, June 1996.

[26] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 231–242, 1998.

[27] J. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445 –3462, Dec. 1993.

[28] H.-Y. Shum, S. B. Kang, and S.-C. Chan. Survey of image-based representations and compression techniques. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(11):1020–1037, Nov. 2003.

[29] A. Smolic, K. Müller, K. Dix, P. Merkle, P. Kauff, and T. Wiegand. Intermediate view interpolation based on multiview video plus depth for advanced 3d video systems. In *15th IEEE International Conference on Image Processing, 2008. ICIP 2008.*, pages 2448 –2451, Oct. 2008.

[30] W. Sweldens. The lifting scheme: a construction of second generation wavelets. *SIAM J. Math. Anal.*, 29:511–546, Mar. 1998.

[31] M. Tanimoto, T. Fujii, K. Suzuki, N. Fukushima, and Y. Mori. Reference softwares for depth estimation and view synthesis. Technical Report M15377, ISO/IEC JTC1/SC29/WG11, Apr. 2008.

[32] D. Taubman. High performance scalable image compression with EBCOT. *IEEE Transactions on Image Processing*, 9(7):1158–1170, July 2000.

[33] M. Wang and M. van der Schaar. Operational rate-distortion modeling for wavelet video coders. *IEEE Transactions on Signal Processing*, 54(9):3505 –3517, 2006.

[34] S. Würmlin, E. Lamboray, and M. H. Gross. 3D video fragments: dynamic point samples for real-time free-viewpoint video. *Computers & Graphics*, 28(1):3–14, 2004.

[35] J. Xu, Z. Xiong, S. Li, and Y. qin Zhang. Three-dimensional embedded subband coding with optimized truncation (3-d escot). In *Applied and Computational Harmonic Analysis 10*, pages 290–315, 2001.

[36] K. Yamamoto, M. Kitahara, H. Kimata, T. Yendo, T. Fujii, M. Tanimoto, S. Shimizu, K. Kamikura, and Y. Yashima. Multiview video coding using view interpolation and color correction. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11):1436–1449, Nov. 2007.

[37] W. Yang, Y. Lu, F. Wu, J. Cai, K. N. Ngan, and S. Li. 4-D wavelet-based multiview video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(11):1385–1396, Nov. 2006.

[38] S. Yea and A. Vetro. View synthesis prediction for multiview video coding. *Image Commun.*, 24:89–100, Jan. 2009.

[39] Y. Yuan and M. K. Mandal. Fast embedded image coding technique using wavelet difference reduction. volume 4925, pages 200–208. SPIE, 2002.

[40] J. Zhang, M. Hannuksela, and H. Li. Joint multiview video plus depth coding. In *17th IEEE International Conference on Image Processing (ICIP)*, pages 2865 –2868, Sept. 2010.