UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Multi-view depth coding based on a region representation combining color and depth information

by

Marc Maceira Duch

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Universitat Politècnica de Catalunya
Teoria del Senyal i Comunicacions

This thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD)

Multi-view depth coding based on a region representation combining color and depth information

---

by Marc Maceira Duch

Advisors: Josep Ramon Morros i Rubió and Javier Ruiz Hidalgo
Barcelona, April 2017

Abstract

Depth map data is used to supplement the color data in multi-view sequences. As depth maps present distinct characteristics than natural color images, new coding techniques are required to represent their smooth regions and sharp edges. In this thesis, segmentation-based coding techniques are proposed to encode depth maps by exploiting the redundancy between color and depth information. Methods developed combine partitions obtained from color and depth images to find efficient representations. The color image is assumed to be available before the depth map coding process, therefore a color partition can be obtained at the decoder without introducing coding cost.

Two hierarchical image segmentation algorithms are proposed to generate color and depth partitions for coding applications. The color segmentation obtains a super-pixel representation using color information, spatial distribution and shape complexity. The depth segmentation uses a 3D planar model for each region to extract the structure of the scene. Color and depth partitions are combined in depth map coding methods to find the final coding partition.

Different methods for texture representation have been explored in this thesis. Initial approaches used 2D coding methods, while a 3D representation have been proposed to represent depth maps from multiple views with a unique segmentation. This 3D representation is used to segment depth maps in single-view and multi-view configurations. Final coding partitions are obtained with a rate-distortion optimization over a hierarchy of regions. Segmentation-based coding techniques proposed obtain competitive results with HEVC coding standards.

Acknowledgments

This thesis was conducted at the UPC Image and Video Processing Group. None of this would have been possible without the help of a lot of people. First and foremost, I would like to begin by expressing my gratitude to my supervisors, Ramon Morros and Javier Ruiz, for our enriching endless discussions, for the uncountable number of revisions (and punctuation marks) on our drafts, for their patience, and specially for stepping up at crucial times (2nd year PhD crisis anyone?). Also thanks to Josep Ramon Casas for having introduced me into the world of research and for giving me the opportunity to continue with the PhD. I would like take this opportunity to thank Ferran Marques and David Varas for their collaboration in the last articles.

Being now a quite experienced researcher, I have had the opportunity to observe in others how determinant is the influence of senior engineers in juniors and the importance of landing in the right environment. An important part of this thesis has been developed with my friends at our office, D5-120. We have shared our *tu que en saps*, baking *negritus*, *constitucions*, *tractores* and *exclusivas*. I sincerely thank you all for creating such a great place to work. I want to acknowledge Albert and Josep, from whom I have learned many *mandangues*; for setting up an awesome technical -and *happy*- environment so we had to think only about research. I should also mention Eva for sharing her precious time with me in the first years of this thesis and Joana for her sharp comments about articles and life.

I am grateful to my parents for giving me (while pushing me and my *parsimònia*) the opportunity to continue my studies all these years. I would also to specially

thank my brother, for being always there and endure all my bad humor in the early and late hours of every day. I also need to thank the rest of the family, Àvia, Padrí, uncles and cousins, but specially the lost ones in this years: Padrina, Bueli, Joan.

It is difficult to go through this without your peers comprehension, so is lovely to be able to count upon friends who have gone through similar experiences. Thanks to all my undergraduate friends for being there during the last 12 years, and for the *divendres*, *accessòries* and *abrils* to come. I also take this opportunity to greet my friends in the gimnàs for helping me to damage my knee. And of course my friends from Tarragona / Andorra.

I also acknowledge that this thesis would not have been possible without the financial assistance of projects TEC2010-18094 (FPI grant BES-2011-045678), TEC2013-43935-R and TEC2016-75976-R, financed by the Spanish Ministerio de Economía y Competitividad and the European Regional Development Fund (ERDF).

Thanks to all, and to the others too. Reinforcements ... or refreshments?

Marc Maceira Duch

Contents

List of Figures

Glossary

3D-HEVC 3D extension of HEVC. 15, 17, 64, 65, 67, 70, 88, 89, 94–97, 103, 130

3DTV 3D TeleVision. 1, 25

AVC/H.264 Advanced Video Coding. 1, 12–15, 24, 64, 67, 70, 107

BD-PSNR Bjontegaard delta PSNR. 57, 96, 114

BD-RATE Bjontegaard delta RATE. 96, 114

BPT Binary Partition Tree. 23, 29–31, 34, 36, 37, 41, 44, 45, 47, 49, 126

CABAC Context-Adaptive Binary Arithmetic Coding. 14

DCT Discrete Cosine Transform. 13, 44, 47, 120

DIBR Depth Image Based Rendering. 3, 24, 63

FVV Free Viewpoint Video. 1

HEVC High Efficiency Video Coding. 1, 12–17, 55–58, 64, 67, 70–72, 88, 94–97, 107, 130

MPEG Moving Picture Experts Group. 2, 13, 14, 103

MV-HEVC Multi-View extension of HEVC. 2, 14, 15, 64, 67, 88, 95–97

MVC Multi-View video Coding. 2, 14, 15

MVD Multi-View video plus Depth. 2, 3, 15, 25, 26, 38, 103, 117

MVV Multi-View Video. 1, 2, 14

PSNR Peak Signal-to-Noise Ratio. 56, 57, 65, 70, 71, 93, 96, 111–114, 120, 128

QSAP Quadratic Semi-Assignment Problem. 6, 75, 100, 101

RANSAC RANdom SAmple Consensus. 37, 60, 121, 126

SA-DCT Shape Adaptive Discrete Cosine Transform. 20, 47–49, 52, 54, 120, 122

SEEDS Superpixels Extracted via Energy-Driven Sampling. 21, 22, 33, 34

SLICS Simple Linear Iterative Clustering. 21, 33, 34, 99

SSIM Structural SIMilarity. 55, 57, 70–72, 93, 112, 113

UCM Ultrametric Contour Map. 23, 27, 33–35, 39, 41

Chapter 1

Introduction

The extension of current visual displays and systems to the third dimension aims to convey depth perception to the viewer. Advances in computer graphics, computer vision, 3D display devices and interactive multimedia systems have promoted the development of new means of storing and transmitting video information. Many applications exploiting 3D video such as 3D video games, 3D films (IMAX cinemas), medical imaging and virtual or augmented reality have arisen over the last years.

3D video applications such as 3D TeleVision (3DTV) [Dod05] and Free Viewpoint Video (FVV) [SMM+06] are supported by multi-view video. The possibilities of both technologies are complementary and can be combined within the same system. 3DTV offers a depth impression of the observed scenery from a single static position in the space. On the other hand, FVV allows the user an interactive selection of the viewpoint and direction within the available views covered by the acquisition cameras. New virtual views can be synthesized in intermediate position between the encoded viewpoints from a smaller set of multi-view inputs.

Multi-view video coding systems can be classified among those that only use color data and the ones that also make use of depth data for each viewpoint. Among the ones using only color data -referred as Multi-View Video (MVV)- stand out the respective extensions of Advanced Video Coding

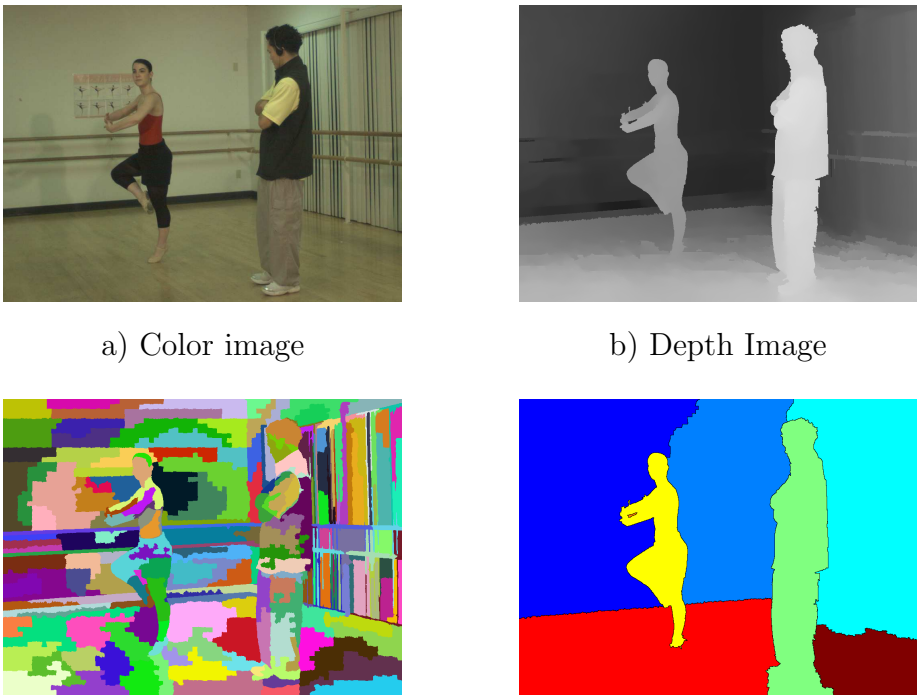
(AVC/H.264) [OBL+04] and High Efficiency Video Coding (HEVC) [SOHW12] to multi-view environments (Multi-View video Coding (MVC) [Mea07] and Multi-View extension of HEVC (MV-HEVC) [TCM+16]respectively). Both extensions use inter view prediction to exploit the multi-view redundancy of the N cameras employed and are back-compatible with the single-view standards.

(a) MVV sequence: Color frame for all cameras at the same time instant

(b) Corresponding depth information for each camera at the same time instant

Figure 1.1: MVD ballet sequence composed of 8 camera views.

The Moving Picture Experts Group (MPEG) specified a standard for efficient compression and transmission of color and depth data referred as Multi-View video plus Depth (MVD). In this configuration, color information and depth maps from several closely situated viewpoints are transmitted. A depth map is an image that contains information relating to the distance of objects from a viewpoint. Depth maps are stored as gray-scale images where the value at each pixel represents the distance (or depth) between the camera and the object. The depth value of each pixel ranges between the maximum and the minimum



a) Color image b) Depth Image

c) Color Partition d) Depth Partition

Figure 1.2: Color-based partition and depth-based partition for an image of the sequence *Ballet*.

distance to the camera position. Depth map values are quantized with 8 bits, with the points closer to the camera having values near 255 and the furthest near 0. Depth map information can be back-projected to the 3D world enabling encoders to establish relations between views. Figure 1.1 shows one frame for each view and the associated depth map of the Multi-View video plus Depth (MVD) sequence *Ballet* [ZKU+04]. The addition of depth maps allows the rendering of virtual views in-between of the encoded camera positions with Depth Image Based Rendering (DIBR) [Feh04].

Depth maps are not intended to be viewed by the user but to render new images. Thus, the aim when coding depth maps is to maximize the perceived visual quality of the rendered virtual color views instead of the visual characteristics of decoded depth maps themselves. Conventional image or video compression techniques have been designed for high visual quality, and are not well adapted to depth coding. The encoding of depth maps has to exploit the characteristics

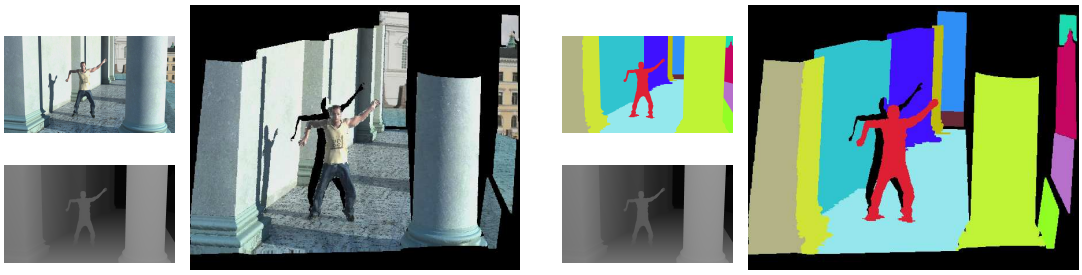
of depth maps and reduce the transmission cost when a large number of captured views are employed [Mea07].

Depth maps are characterized by the presence of large homogeneous areas separated by sharp edges. Errors close to sharp edges lead to severe rendering artifacts, while errors on areas without important transition may have negligible influence on the final quality. Block-based transformations such the ones found in natural video coders are unable to represent depth edges efficiently. High frequency components are needed to represent depth edges in block-based transforms.

The particular characteristics of depth maps suggest the possibility of using a segmentation-based technique to encode depth maps. Segmentation-Based coding [KIK85] consists in describing the image using arbitrarily shaped contours and coding the textured regions in-between them. An image segmentation that separates the main depth transitions in-between different regions will contain regions without sharp transitions. The depth texture information of these regions will have smooth changes that can be represented compactly.

The main drawback of region-based coding representations consists in having to code the position of contours in addition of the texture information. A partition built from the color image can be used to reduce the cost of coding depth boundaries. Color and their associated depth image present structural similarities since they observe the scene from the same location. In Figure 1.2 a pair of partitions for the color image and the depth map are shown.

In this thesis, different options have been considered for combining color and depth partitions efficiently. All the proposed methods use a region-based coding representation of depth maps. The 3D geometry of the scene is exploited by generating a 3D representation from the segmentation of several depth maps, obtaining a unique representation of the scene in the 3D domain. Figure 1.3 shows a 3D representation generated from an input view.



a) b)

Figure 1.3: a) Depth maps can be back-projected to the 3D world using the camera parameters. In the figure, each point is represented with the corresponding color of the color image. b) A 2D image partition is used to build a 3D model of the scene.

1.1 Context and Contributions

The purpose of this thesis is to build an encoding system for multi-view depth maps. To do so, a 3D representation of the scene is proposed to encode the information of the multiple views. The different algorithms developed use a region-based coding approach, where a 2D partition obtained on the depth maps signals the contours between regions and the texture of the region is modeled and encoded. To reduce the cost of encoding the depth boundaries, a color image partition is used to predict the location of depth edges.

More concretely, the contributions of this work and their position in this document are the following:

- Color image partition (Section 3.1): a new similarity measure using region centroids for a binary merging algorithm to find super-pixel color partitions.
- Depth image partition (Section 3.3): depth map data is back-projected to 3D where regions are represented with 3D planar models. A similarity measure for a binary merging algorithm is defined in the 3D domain to obtain planar representations of the scene.
- 2D Single-view coding (Chapter 4): two different single-view coding meth-

ods have been proposed using 2D models to represent the texture information.

- 3D Single-view coding (Chapter 5): color and depth map partitions are merged in a single-view coding technique adding progressively the main depth edges and representing the corresponding regions using 3D models.

- Rate-distortion Hierarchy Optimization (Section 6.1): the rate-distortion problem is solved as a Quadratic Semi-Assignment Problem (QSAP) which defines the relation between regions in terms of their common contour.

- Scene Representation (Appendix E and Section 6.2): the optimization finds a unique partition across views using using only depth map partitions (Appendix E) and color and depth map partitions (Section 6.2), generating a consistent 3D scene representation in each case.

- Multi-view coding (Section 6.2): the rate-distortion optimization combines color and depth boundaries to find the optimal partition that represents the depth information of the multiple views. The obtained representation is used to efficiently encode the depth information of the multiple views.

The work done in this thesis has been published in the following conference articles:

- a.1: Depth map coding based on a optimal hierarchical region representation [MRHM12], 3DTV Conference, Zurich 2012.

- a.2: Fusion of colour and depth partitions for depth map coding [MMRH13], Digital Signal Processing, Santorini 2013.

- a.3: Region-based depth map coding using a 3D scene representation [MMRH15] ICASSP, Brisbane 2015.

- a.4: 3D Scene representation via rate-distortion optimization [SUBMITTED], ICIP 2017.

And in the following journal articles:

- j.1: Depth map compression via 3D region-based representation [MMRH16], Multimedia Tools and Applications 2016.
- j.2: 3D hierarchical optimization for Multi-view depth map coding [SUBMITTED], Multimedia Tools and Applications.

	Segmentation	**Objective**	**Approach**	**Texture**
a.1	Color	2D SV coding	Optimization	2D
a.2	Color + depth	2D SV coding	Fusion	2D
a.3	Color + depth	3D SV coding	Fusion selected	3D
j.1	Color + depth	3D SV coding	Fusion selected	3D
a.4	Depth	3D MV Representation	Optimization	3D
j.2	Color + depth	3D MV coding	Optimization	3D

Table 1.1: Overview of the articles developed during this thesis. SV stands for single-view and MV for multi-view.

An overview of the different articles are summarized in table 1.1. In a.1, a single color segmentation is used to approximate the depth map contours. As the color segmentation could not find all the depth edges, in a.2 a fusion of color and depth partitions is proposed. In these first articles, a 2D region model is used to encode the depth maps.

To be able to exploit the 3D characteristics of the multi-view sequences, in a.3 and j.1, a 3D region model is proposed to encode the texture of the depth maps in a single-view approach, having in mind a future multi-view extension. This extension is explored in a.4 and j.2 articles.

This thesis is organized as follows: In Chapter 2 a review of the state of the art relevant for this work is provided. Chapter 3 tackles the problem of generating segmentations adapted to depth map coding, both for color images and depth map images. Chapter 4 presents the two methods of 2D coding (a.1 and a.2). In

Chapter 5, 3D models for single view region-based coding of depth map images are presented (a.3 and j.1). In Appendix E the initial exploration of 3D multi-view is explored (a.4). The multi-view depth map coding based on 3D models is presented in Chapter 6 (j.2). Finally conclusions and future work are presented in Chapter 7.

Chapter 2

State of the art

In this Chapter, a review of the state of the art relevant for this Thesis is provided. An introduction of video coding is provided in Section 2.1. The block-based video coding approach used in coding standards is summarized in Chapter 2.2. A definition of segmentation-based coding techniques as the ones that will be used in this work are presented in Section 2.3. Finally depth map coding techniques and 3D scene representation methods in the literature are studied in Sections 2.4 and 2.5.

2.1 Video Coding

The communication problem of transmitting video over a channel can be stated as sending the source data with the highest fidelity within the available bit rate, or alternatively, sending the source data using the lowest bit rate possible while maintaining a certain reproduction fidelity. In both options, a trade-off is made between bit rate and fidelity. The behavior of a video codec (a video system including a coder and a decoder) is determined by the performance of the rate-distortion trade-off.

Hence, video codecs are mainly characterized by:

- Throughput of the channel: average bit rate over a given channel, in-

fluenced by the transmission channel bit rate and the error-correction overheads.

- Distortion of the decoded video: distortion is introduced by the video codec and also by errors in the channel.

In practical applications other factors have to be taken into account:

- System delay: can be restricted by factors as the encoding or decoding processing delay, buffering of needed data or the speed at which data are conveyed through the transmission channel.
- Complexity: have to be evaluated in different aspects including the computational requirements, memory capacity and memory access requirement.

Thus, the practical source coding design problem is posed as follows: given a maximum allowed delay and a maximum allowed complexity, achieve an optimal trade-off between bit rate and distortion for the range of environments envisioned in the scope of the applications.

2.1.1 Source Coding Strategies

Techniques for digital compression can be classified as follows:

- Prediction: A process where a set of prediction values is used to predict the values of the input samples. In this procedure only the differences between the prediction values and the input samples have to be represented, which are called the residual values.
- Transformation: A process consisting of forming a new set of samples from a combination of input samples. A transformation usually avoids representing similar values repeatedly by using frequency analysis. A typical benefit of transformation is that the most relevant aspects of the

set of input samples are typically concentrated into a small number of variables.

- Quantization: A process by which the precision used for the representation of a sample value is reduced in order to decrease the amount of data needed to encode the representation. The precision can be controlled by a step size that specifies the smallest representable value increment. Among the techniques listed here for video compression, quantization is typically the only one that is inherently non-invertible: quantization produces a mapping that involves some loss of fidelity.

- Entropy coding: A process by which discrete-valued source symbols are represented in a manner that takes advantage of the relative probabilities of the various possible values of each source symbol. Types of entropy coding are the the variable-length code (VLC), which establish a code table that associate short codes to values that are highly likely to occur and longer to represent less likely symbol values, and the arithmetic coding, which encodes the entire message into a single number.

Video coding techniques can be classified depending on the prediction techniques used. One way of compressing video is by compressing each picture separately. This type of coding is referred as intra-picture or Intra coding, since the picture is coded without any reference to other images of the sequence. However, higher compression figures are obtained by using the redundancy among consecutive images in a sequence. Video can be represented more efficiently by sending only the changes in the video scene rather than coding all regions repeatedly. This coding is referred as inter-picture or Inter coding. This ability to use temporal redundancy to improve coding efficiency is what distinguishes video compression from the Intra compression exemplified by still images standards such as JPEG.

2.2 Block-Based Coding

Nowadays, the two main video coding standards are the commonly used AVC/H.264 [WSBL03] and the High Efficiency Video Coding (HEVC) / h.265 [SOHW12]. In this Section a review of block-based coding characteristics is provided before centering to HEVC. The main characteristics of block-based standards are listed below:

- Hybrid codec: inter/intra-picture prediction and 2D transform coding. Regions can be predicted using inter-picture prediction, and a spatial frequency transform is applied to the Intra-coded regions. The encoding process for inter-picture prediction consists of choosing motion data comprising the selected reference picture and motion vector to be applied for predicting the samples of each block. The residual signal of the intra or inter prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The resulting coefficients are then scaled, quantized, entropy coded, and transmitted together with the prediction information.
- Macroblocks and Slices: Each picture is partitioned into slices which in turn are subdivided into macroblocks. Each slice can be parsed independently of the other slices in the picture. There are three fundamental slice types depending on the freedom for the prediction signal:
  - I slices: each macroblock uses intrapicture coding using spatial prediction from neighbouring regions.
  - P slices: support intrapicture and interpicture predictive coding.
  - B slices: support intrapicture coding, interpicture predictive coding, and also interpicture bipredictive coding.
- Quantization: A Quantization Parameter is used for determining the quantization of transform coefficients.
- Adaptive Deblocking Filter: Block edges are typically predicted with less accuracy than interior samples, and block transforms also produce block

edge discontinuities. The filter reduces blockiness while retaining the sharpness of true edges in the scene.

- Profiles and Levels: Profiles and levels specify conformance points to facilitate interoperability for various applications.

2.2.0.1 High Efficiency Video Coding (H.265)

HEVC [SOHW12] is a video compression standard developed jointly by Video Coding Experts Group and MPEG through their Joint Collaborative Team on Video Coding. The HEVC project was launched to achieve major savings for equivalent visual quality relative to the bit rate needed by the widely used AVC standard. HEVC has been designed to address essentially all existing applications of previous standards and to particularly focus on two key issues: increased video resolution and the increased use of parallel processing architectures. The main features of the HEVC are:

- The video coding layer of HEVC employs the same hybrid approach used in AVC/H.264 (inter/intra-picture prediction and 2D transform coding).
- Coding Tree Units and Coding Units: The core of the coding layer in previous standards was the macroblock, whereas the analogous structure in HEVC is the coding tree unit, which has a size selected by the encoder.
- Prediction Blocks: The decision whether to code a picture area using inter-picture or intra-picture prediction is made at the coding unit level. Luma and chroma coding blocks can then be further split in size and predicted from luma and chroma prediction blocks. HEVC supports variable Prediction Blocks sizes from 64×64 down to 4×4 samples.
- Transform Blocks: The prediction residual is coded using block transforms. Integer basis functions similar to those of a Discrete Cosine Transform (DCT) are defined for the square transform block sizes.
- Motion compensation: Quarter-sample precision is used for the motion vectors, filters are used for interpolation of fractional-sample positions.

Multiple reference pictures are used. For each prediction block, either uni-predictive or bi-predictive coding can be done.

- Entropy coding: Context-Adaptive Binary Arithmetic Coding (CABAC) is used for entropy coding. It is a lossless compression technique providing much better compression than other entropy encoding algorithms for video encoding. CABAC encodes binary symbols with an arithmetic coding approach using local context.

2.2.1 Multi-view Video Coding

In addiction at their work in video coding, the Joint Video Team of VCEG and MPEG has also promoted the standardization of video coding with multiple views. An extension of AVC/H.264 that is referred to as Multi-View video Coding (MVC) was added in July 2008 and an additional stereo high profile was completed one year later [Mea07]. The standard enables inter-view prediction to improve compression capability, as well as supporting ordinary temporal and spatial prediction. For Multi-View Video (MVV) with up to eight views, an average of 20% reduction in bit rate was reported, relative to the total simulcast bit rate [MMW11].

MV-HEVC

HEVC standard has been extended to support efficient representation of multi-view video and depth-based 3D video formats [TCM+16]. The multi-view extension, Multi-View extension of HEVC (MV-HEVC), allows efficient coding of multiple camera views and associated auxiliary pictures, and can be implemented by reusing single-layer decoders without changing the block-level processing modules since block-level syntax and decoding processes remain unchanged. Bit rate savings compared with HEVC simulcast are achieved by enabling the use of inter-view references in motion-compensated prediction.

MV-HEVC follows the same design principle as MVC. The design enables a single texture base view to be extracted from MV-HEVC bitstreams, which is

decodable by a HEVC decoder. MV-HEVC comprises only high-level syntax additions and can thus be implemented using existing single-layer decoding cores. Higher compression (compared with simulcast) is achieved by exploiting redundancy between different camera views of the same scene.

A key benefit of the MV-HEVC architecture is that it does not change the syntax or decoding process required for HEVC single-layer coding below the slice level. This allows the reuse of existing implementations without major changes for building MV-HEVC decoders. Motion vectors in MV-HEVC may represent temporal or disparity information, depending whether they are computed between frames in different temporal instants for the same view or between views in the same temporal instant. Block-level HEVC motion compensation modules can be used which operate the same way regardless of whether prediction have been used. The average gain for MV-HEVC compared with HEVC coding are over 30% [TCM+16].

3D-HEVC

The more advanced 3D video extension, 3D extension of HEVC (3D-HEVC) uses the Multi-View video plus Depth (MVD) format. Additional bit rate reduction compared with MV-HEVC is achieved by exploiting statistical dependencies between video texture and depth and specifically adapt to the properties of depth maps. The coding scheme represents an extension of HEVC, similar to the MVC extension of AVC/H.264. In addition to the disparity-compensated prediction advanced techniques, the representation of depth blocks, and the encoder control for depth signals have been integrated.

In 3D-HEVC all video pictures and depth maps that represent the video scene at a given time instant are encoded in an access unit similarly to MVC. Inside an access unit, the video picture of the so-called independent view is transmitted first directly followed by the associated depth map. Thereafter, the video pictures and depth maps of other views are transmitted.

Each independent view is coded using a unmodified HEVC coder. The corre-

sponding sub-bitstream can be extracted from the 3D bitstream to be displayed on a conventional 2D screen. The other components are coded using modified HEVC coders, which are extended by including additional coding tools and inter-component prediction techniques that employ previously coded data inside the same access unit. The principal characteristics are listed below:

- Coding of Dependent Views: the disparity-compensated prediction has been added as alternative to motion-compensated prediction in a similar way as for MVC.
- Coding of depth maps: The HEVC design has been optimized for natural video. To optimize the coding of depth maps, two families of modes have been added to represent the sharp edges and large regions with nearly constant values characteristics of depth maps:
  - Depth Modeling Modes: New intra coding modes have been added to enable a better representation of depth edges. Additional modes partition a depth block into two non-rectangular regions and represent each of these regions by a constant value. Two types of partitioning are used, namely Wedgelets, for segmentations using a straight line, and Contours, for arbitrary segmentations. The constant values for the two regions are predicted based on the reconstructed samples in neighboring blocks and the remaining difference is quantized and coded in the bitstream.
  - Motion Parameter Inheritance: Since video pictures and depth maps represent different aspects of the same video scene, the motion characteristics should be similar. A inter coding mode for depth maps is added in which the partitioning of a block into sub-blocks, as well as the associated motion parameters, are inferred from the co-located block in the associated video picture.
- View Synthesis Optimization: Coding artifacts in depth data are only indirectly perceivable in the synthesized video data. In the encoder control for depth maps, the distortion is not directly measured in the depth map

domain, but instead the resulting distortion in one or more synthesized views is analyzed.

- Coding Performance: In comparison to simulcasting the different signals using HEVC, 3D-HEVC approach provides about 40% and 50% bit rate savings for configurations with 2 and 3 views, respectively [TCM+16].

2.3 Segmentation-Based Coding

As presented in Section 2.1, video compression standards divide the images in rectangular blocks. Each block is coded with a hybrid prediction-transform approach. An alternative approach consist in describing the image using arbitrarily shaped contours and coding the textured regions in-between those contours. These methods are called Segmentation-based coding methods since a complete segmentation of the image is needed in order to code the image. In segmentation-based coding methods the shape and size of each segment are arbitrary and ideally are adapted to the image content.

The principal drawback of segmentation-based coding methods is the need of coding the contour information in addition to the texture information. Errors in contours are extremely noticeable to the human visual system, leading to the use of lossless or near-lossless coding techniques for contour coding. The joint coding of texture and contour creates a trade-off in the bit rate assigned to each term. A partition with many regions is desirable since the texture inside of each region will be more homogeneous and therefore could be coded using less bits, but the coding cost of the shape will increase. On the other hand, reducing the numbers of contours will create regions with more texture variations that will need more coefficients to provide the desired quality.

The usual process to perform the image coding consist in encoding first the contours and, once the final contours are obtained, encoding the texture to avoid errors if a lossy contour coding technique is used.

2.3.1 Contour Coding

In the literature it is possible to find a wide range of contour coding techniques [Mor04]. Among the lossless techniques, can be pointed out Chain Code [Fre61], Morphological Skeleton [MS86], transition points [Pin98] and Quadtree decomposition [KM95]. Multigrid Chain Code (MGCC) [MG96] is an example of a near-lossless technique (it is an extension of Chain Code). Some examples of lossy techniques are Fourier descriptors [ZR72], and polygonal [Dun86] and Spline [Sch07] approximations.

Other option to signal contours include binary coders such as JBIG2 [HKM+98] or PAQ [Mah05]. JBIG2 [HKM+98] is an image compression standard designed for bi-level images suitable for both lossless and lossy compression. PAQ [Mah05] are a series of open source data compression techniques which use context mixing: a large number of models estimate the probability that the next bit of data will be a 0 or 1. These predictions are combined and arithmetic coded.

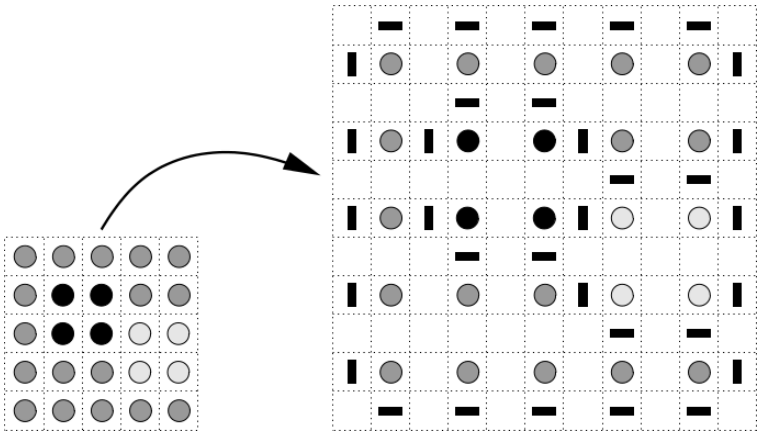


Figure 2.1: Chain code contour coding: relationship between partition and contour grid sites.

Chain Code techniques [Fre61] allow lossless coding of image partitions. Regions are represented by their boundaries and the coding process consists on tracking and encoding the boundaries. The process used to encode a partition is:

- Define an appropriate boundary representation. This usually leads to

constructing a contour image.

- Using the fact that two consecutive boundary points in a discrete grid are neighbours, encode the movements from one contour point to another, starting from an initial point, until the complete contour is tracked.

The contour of a given partition can be represented using an hexagonal contour grid. In this case, a one to one relationship between partitions and their boundary representation can be achieved. In this type of representation, each pair of neighbor pixels in the partition is separated by a contour grid site. This contour grid site is labeled as active if the associated pair of pixels have different labels, otherwise it is labeled as inactive.

This concept is illustrated in the example of Figure 2.1, where circular and line segment elements represent sites from the partition and contour grids, respectively. Encoding of the contours is done by specifying the set of movements necessary to track the complete set of active contour grid sites. Shape and position information are jointly coded by introducing the location information in the chain code itself.

2.3.2 Texture Coding

The regions resulting after a segmentation process are statistically quasi-stationary and should therefore enable higher data compression ratios. However, traditional block-based texture coding techniques can not exploit efficiently this quasi-stationarity because they perform poorly on border blocks.

Different texture coding methods can be used in this stage, provided they are region oriented. The simplest texture coding method consists of the coding of the mean value for every region, thus leading to a very cheap texture coding cost. Several more complex methods for the efficient coding of texture information have been proposed so far. The polynomial method represents an image segment using polynomial functions of different degrees [SC96, BMC88]. Gilge et al. [GEM89] proposed a more accurate method in which a generalized or-

thogonal transformation can be constructed with respect to the shape of the image segment at the cost of calculating the orthogonalizing basis images. Generalized orthogonal transforms rely on an approximation of the texture within each region by an orthogonal polynomial function. Only the coefficients of this function are sent to the receiver.

Low complexity Shape Adaptive Discrete Cosine Transform (SA-DCT) algorithms have been introduced to efficiently transform the image blocks that contain object boundaries [SM95, SBM95]. These algorithms first divide an image into blocks and then code the boundary blocks using the SA-DCT. Although boundary blocks are encoded effectively, the problem of blocking effects inside the image segments remains unsolved. Another approach to encode the image segments is the region-based wavelet transform [KIAK97]. The advantages of the wavelet-based scheme are its low computational complexity, ability to code the details inside the segment, and absence of blocking effects in the reconstructed images.

2.3.3 Image Segmentation

Segmentation-based coding techniques require a image partition to encode the image. Image segmentation is one of the most basic and studied problems in computer vision. It is the process of partitioning a digital image into multiple segments. The goal of segmentation is to simplify the representation of an image into something that is more meaningful and easier to analyze. Here a brief review of segmentation techniques is provided.

2.3.3.1 Superpixels Methods

Superpixels are popular in computer vision applications for their aim to oversegment the image by grouping pixels that belong to the same object. Superpixels provide a convenient primitive from which to compute local image features. They capture redundancy in the image and greatly reduce the complexity of subsequent image processing tasks. Superpixels have proved increasingly useful for a variety of applications such as depth estimation, image segmentation,

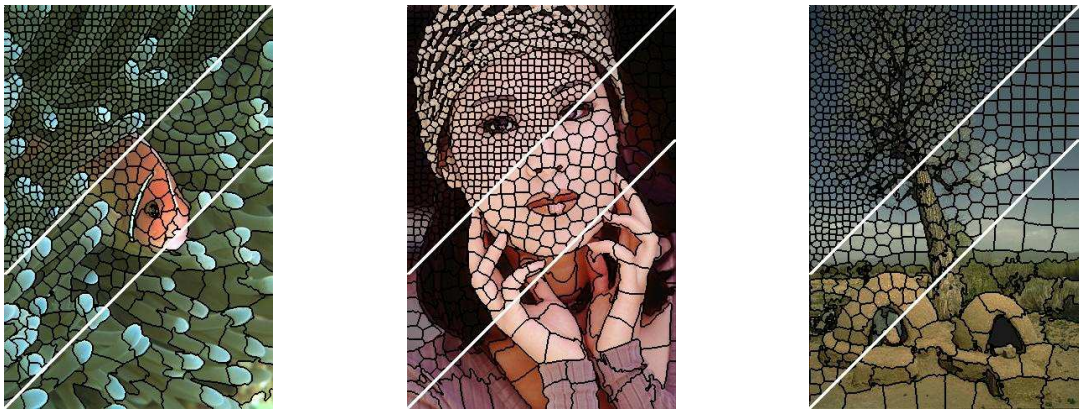


Figure 2.2: Superpixel segmentation with different number of regions obtained with Simple Linear Iterative Clustering (SLICS). Images obtained from [ASS+12].

skeletonization, body model estimation, or object localization. Figure 2.2 shows an example of superpixel segmentation.

Superpixel representations have the following desired properties:

- Computationally efficient: reduces the complexity of images from hundreds of thousands of pixels to only a few hundred superpixels.
- Representationally efficient: pairwise constraints between pixels, can model much longer-range interactions between superpixels.
- Perceptually meaningful: each superpixel is a perceptually consistent unit. All pixels are most likely uniform in color and texture.

The algorithm Simple Linear Iterative Clustering (SLICS) [ASS+12] performs a local clustering of pixels in the 5-D space defined by the L, a, b values of the CIELAB color space and the x, y pixel coordinates. SLICS start from a regular grid of centers or segments, and grow the superpixels by clustering pixels around the centers. At each iteration, the centers are updated, and the superpixels are grown again. A distance measure enforces compactness and regularity in the superpixel shapes.

Superpixels Extracted via Energy-Driven Sampling (SEEDS) [BBR+13] introduce an approach based on a hill-climbing optimization. Starting from an initial superpixel partitioning, it continuously refines the superpixels by modifying the boundaries evaluating the changes with a energy function, based on enforcing color similarity between the boundaries and the superpixel color histogram. The procedure to tackle the superpixel problem in SEEDS starts from a complete superpixel partitioning, which is iteratively refined.

2.3.3.2 Hierarchical Segmentation

Image segmentation methods can be classified according to whether they provide a single image partition or a hierarchy of regions [PT14]. Flat Segmentation methods produce a single image partition for each parametrization, they represent different scales by varying some of these parameters. On the other hand, hierarchical partitions provide a pyramidal structure that represents regions at all scales. Hierarchical representations contain partitions of the image at different levels of detail in a single structure.

Hierarchies may be constructed in Top-down or Bottom-up algorithms. In top-down algorithms prior knowledge about objects such as its possible shape, color, or texture are used to guide the segmentation. On the other hand, bottom-up algorithms are constructed in an iterative process, as illustrated in Figure 2.3. On the bottom extreme of the diagram, the algorithm starts by a partition at the maximum level of detail (pixels, superpixels, etc.), representing each region as a node. Then, it iteratively merges those sets of regions that are more similar according to a given criterion.

Some of the most successful hierarchical segmentation methods in the literature include Ultrametric Contour Map (UCM) [AMFM11] or Binary Partition Tree (BPT) [SG00]. In Ultrametric Contour Map (UCM), a contour detector combines multiple local clues into a globalization framework to obtain the probability of contour for each boundary point. UCM algorithm transforms the contour map into a hierarchical representation. Partitions at different level of detail are extracted from the hierarchy according to the contour probability

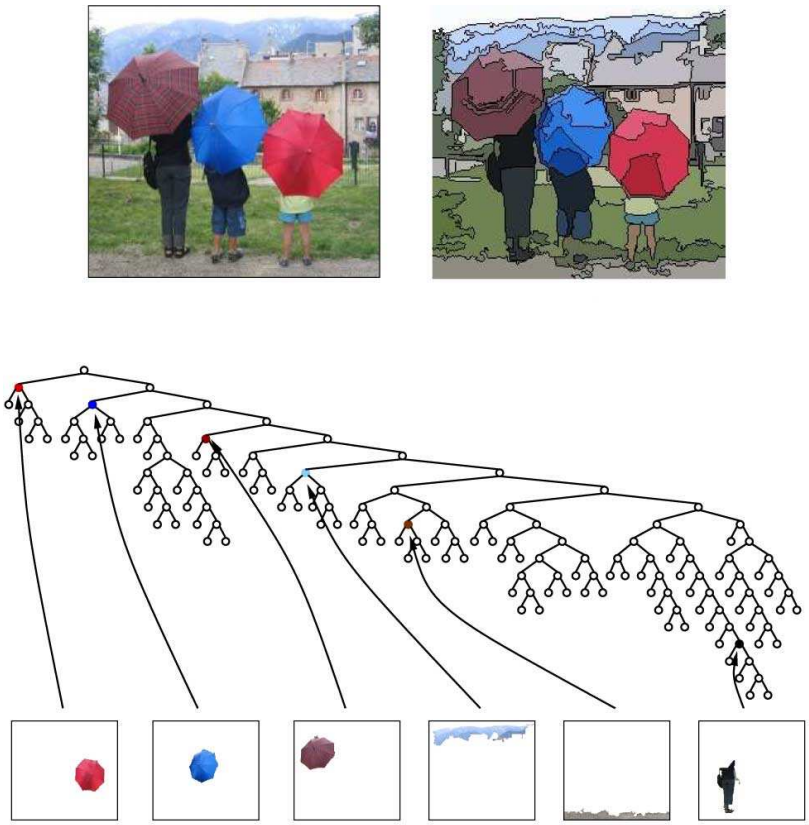


Figure 2.3: Hierarchical representation of an image highlighting object nodes. Figure obtained from [VMS08].

among regions. Binary Partition Tree (BPT) algorithm creates a hierarchy of regions by a merging algorithm that can make use of any similarity measure. More details about the BPT method are given in Section 3.

2.4 Depth Map Coding

Depth data present a set of characteristics that diverge from color images. Typically, they are composed of large smooth regions separated by sharp depth transitions. Classical video compression techniques for color images have been designed to achieve high visual quality on the encoded signal. For this purpose, the image is divided in blocks and each block is coded using a transform and a quantization step [OBL+04]. However, the direct application of these techniques on depth maps leads to coding artifacts in the edges due to quanti-

zation [MMS+08].

Many techniques aimed at preserving the depth map edges while reducing the cost of coding the smooth regions have been presented. Edge-adaptive wavelets have been applied for depth map coding by using shape-adaptive lifting [MD08] or by switching between long filters in homogeneous areas and short filters over the edges [DTPP08]. Graph-based wavelets have been proposed to preserve edge information in depth maps [SSO09]. All these approaches try to avoid applying transform coding across the edges.

Other family of approaches explicitly encode the position of the most significant depth edges (see for e.g. [Jag11]). The main depth contours are signaled and the in-between texture is encoded with piecewise-linear functions. In [CKO+11], two different modes are proposed to signal the depth edges depending of their complexity over one unified framework. Then, the simplest transform is chosen for each block. Instead of explicitly representing the depth maps boundaries, [SMAP14] proposes encoding the residual prediction errors with quantization at pixel domain rather than in the transform domain. The coding of prediction errors in the spatial domain is also used in [MMMW15].

The fact that color images and depth maps represent the scene from the same viewpoint leads to a high structural similarity between both images. The edges in depth are often located in the same location of color discontinuities. In order to avoid signaling the explicit location of depth edges, this similarity between depth maps and the corresponding texture image can be used. This strategy is used in [LLZ+15], where skip-coding mode and motion vectors in the coded texture are used in the depth map. The similarity between color information and depth maps to avoid encoding the location of depth map edges is also used in [DYQZ12]. A joint color and depth coding for multi-view video is proposed in [GCM+16]. Making use of DIBR, the color and depth information is projected obtaining a inter-view prediction, missing pixels are in-painted using minimum explicitly encoding.

A color image segmentation is proposed in [MZZF11] to predict the shape of the

different surfaces in the depth map. Then, each region is approximated either by a parametrized plane or by AVC/H.264 Intra coder. An approximation of a depth partition using the color image also is used in [RHMA+12], where the depth texture is encoded with orthogonal basis.

The reduction of the redundancy present in multi-view image sets have been studied in several works. This inter-view redundancy can be removed by extracting the geometrical structure of the scene. In [MOF15], a geometrical representation is introduced to describe the multi-view information with a graph. Starting from an initial view, inter-view redundancy is avoided by adding new graph nodes only if new information appears in the subsequent views. Similarly, a novel 3D video coding technique based on the creation of a panorama view is detailed in [FLG15]. This view represents most of the visual information acquired from multiple views using a single virtual view characterized by a larger field of view. Moreover, in [SLJ+14], the inter-view redundancy is removed by an analysis performed over the occluded areas. In the second view, only new areas which were occluded in the first view are coded.

Since depth maps are not directly displayed but used to render new images, the usual rate-distortion criterion over the depth map may not give a proper measure of the quality of the representation. Different distortion measures are studied to better relate the errors in depth maps with their effect on the synthesized color images. To this end, it is preferable to optimize the rate-distortion measure over the synthesized views rather than over the depth map directly [KOLT15]. The main advantage of modeling the coding error on the synthesized view instead of calculating it on the depth map is that the impact of the coding errors can be determined in the generated virtual view. Moreover, in [TSMW12, ZYL+16], new distortion metrics are proposed to measure the influence of depth errors in the synthesized virtual view.

Planar models have been also proposed for depth representation on 3DTV applications. In [OA14], a markov random field model that mimics a rate-distortion trade-off is used in a stereo configuration to obtain a co-segmentation with planar approximations. A single reference MVD format is also proposed to fuse the

information of the stereo views in a single reference view. Two different compression techniques are explored, the first one follows the two-view structure while the second fuses the data in a single-reference MVD format.

2.5 3D Representation

RGB-D data provides a dense representation of the scene from a single-view. However, in multi-view sequences, depth data associated with each view heavily augments the amount of data needed to store the 3D information. In this context, extracting a 3D model of a scene from multiple depth maps removes redundant information of the views while obtaining a unique 3D representation of the scene. This representation can be further used in tasks as action detection [LZ15], scene recognition [GAM13], scene labeling [WLC+15] or robotics [KATS11, KS14].

The use of 3D planar models have been used to recover the structure of the scene from panoramic sequences [MK09] or from a multi-camera environment [YVEM15]. With the raise in popularity of 3D sensors, 3D planar models have been used to segment images while reducing the noisy nature of the obtained depth maps. In [SMLN11] multiple planes are detected and tracked in Time of Flight depth images. In [LKF16] a layered scene decomposition extracts the structure of the scene handling the occlusions using priors. Each pixel in the image can be associated to multiple labels which represent the occlusion of background regions, each region is represented whether for a planar or a b-spline model.

Using MVD data, in [BP14] the 3D point cloud from multiple RGB-D cameras is back-projected to the 3D world and used to generate a set of 3D planar patches consistent among the views. These patches are obtained with a Markov random fields using a voxelization of the scene and several 3D planar candidates. More recently, in [VDV16] the same problem is tackled in a wide-baseline stereo configuration. Each image is segmented independently and a fitting process assigns a 3D plane to each region. Planes are used as candidates in an

energy-minimization problem, which optimizes the error of the model and the smoothness over neighboring regions. 3D planar models have shown also to be useful to extract 3D planes from stereo configurations [SSS09] or to co-segment multiple view objects [KSS12].

Segmenting point clouds coming from RGB-D sensors has been tackled as a part of semantic labeling of indoor scenes and scene understanding. In [SHKF12] a hierarchical segmentation is proposed using depth clues to infer the relations between objects and using priors to classify the different objects. In [RBF12] an initial superpixel segmentation is used to compute kernel descriptors such as gradient, color and surface normal to build a hierarchy that is used to assign the labels to each node using markov random field. Gupta et al. [GAM13] generalize the *UCM* hierarchical segmentation to incorporate depth information for semantic segmentation. Lately, in [WLC+15] an unsupervised framework where joint feature learning and encoding is proposed for RGB-D scene labeling.

Chapter 3

Proposed Segmentation Techniques

In this thesis, two criteria for obtaining a hierarchical segmentation of images have been developed, one for color images and the other for depth maps. Both hierarchical segmentations obtain regions adapted for coding purposes. These methods use the Binary Partition Tree (BPT) algorithm presented in 2.3.3.2. First, the BPT algorithm is reviewed. Then, the two techniques are presented and evaluated in terms of segmentation accuracy.

BPTs are hierarchical region-based representations of images [SG00]. The representation of a hierarchy can be depicted with nodes as shown in Figure 3.1. Each node of the hierarchy represents a region in the image, and the parent node of a set of regions represents their merging. In all stages it is assumed that this hierarchy is binary (regions are merged by pairs). Commonly, such hierarchies are created using a greedy region merging algorithm that, starting from an initial leave partition LP, iteratively merges the most similar pair of neighboring regions according to a region similarity criterion (referred as $O_{criterion}$ in the following).

The algorithm works on a Region Adjacency Graph, that is, a set of nodes representing regions and a set of links defining the connectivity between regions.

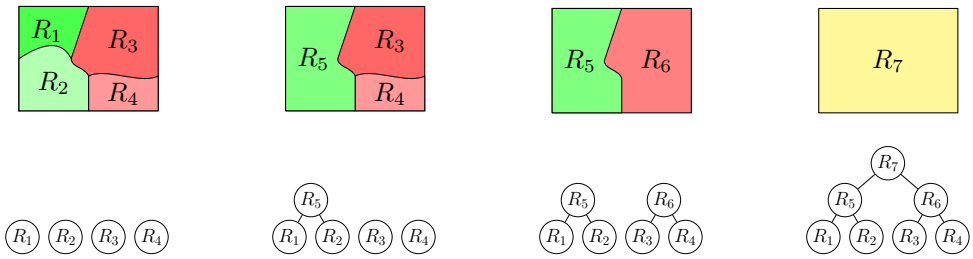


Figure 3.1: Hierarchical representation of an image. From left to right, the two most-similar neighboring regions are merged at each step. The hierarchical representation is depicted as a tree, where the region formed by merging two segments is represented as the parent of the respective nodes.

A node of the graph can represent either a region, a flat zone or even a single pixel. A merging algorithm on this graph is a technique that removes some of the links and merges the corresponding nodes. The steps in each merging step are the following:

- compute a similarity measure for each pair of neighbor regions.
- select the most similar pair of regions and merge them into a new region.
- update the neighborhood and the similarity measures. The algorithm iterates until the desired number of regions is obtained.

The merging process ends when the whole image is represented by a single region, which is the root of the tree. The set of mergings that creates the tree, from the leaves to the root, is denoted as merging sequence. In a binary hierarchy, a merging sequence contains N partitions, where N is the number of leaves (regions in LP). This is the set of partitions that is usually analyzed when working with hierarchies. Still, once the hierarchy is built, an analysis on the whole hierarchy may obtain partitions which are not included in the merging sequence.

The creation of a BPT relies on two major notions: the merging criterion and the region model. The merging criterion defines the similarity of neighboring regions and, therefore, the order in which regions are going to be merged. The region

model specifies how regions are represented and how to compute the model of the union of two regions. Several criteria and models are listed in [VMS08].

3.1 Color Segmentation

The objective of the color segmentation is to obtain a P^{Color} partition similar to superpixel partitions in the literature [BBR+13, ASS+12]. Superpixels provide a useful representation of the image with a reduced number of entities with respect to the pixel representation. The P^{Color} partition will be used as a base partition to encode depth maps. Thus, it will determine the texture rate needed to encode the set of regions. For HD sequences, experimental results developed have shown that P^{Color} has to be composed of a few hundred regions. At that number of regions, superpixels techniques fail at retrieving accurately the discontinuities in the image. To overcome that, a superpixel inspired technique is proposed, which obtains higher boundary retrieval than current superpixel techniques in the literature.

Superpixels techniques (see Section 2.3.3.1) perform a clustering process using the color similarities and spatial proximity. In this Section, these two characteristics are introduced in the BPT. The new similarity measure for the color image is derived from the *bpt_nwmc* measure in [VMS08].

In *bpt_nwmc*, the region model is constant for all the pixels of the region and the model M_R is obtained by averaging the values of the pixels $p \in R$, in the YCbCr color space:

$$M_R = \frac{1}{N_R} \sum_{p \in R} I(p) \tag{3.1}$$

where N_R is the number of pixels of region R and I is the image in the YCbCr color space.

The *bpt_nwmc* similarity criterion consists of two terms: The first one, based on color similarity, is the Weighted Euclidean Distance between Models (*wedm*) which compares the models of the original regions, R_1 and R_2, with the model

of the region obtained after the merging $R_1 \cup R_2$:

$$O_{wedm}(R_1, R_2) = N_{R_1}||M_{R_1} - M_{R_1 \cup R_2}||_2 + N_{R_2}||M_{R_2} - M_{R_1 \cup R_2}||_2 \quad (3.2)$$

The second term is related to the contour complexity of the merged regions. The measure computes the increase in perimeter $\Delta P(R_1, R_2)$ of the new region with respect to the largest of the two merged regions: $\Delta P(R_1, R_2) = min(P_1, P_2) - 2P_{12}$ where P_1 and P_2 are the perimeters of the regions R_1 and R_2 and R_{12} is the common perimeter between regions.

The term that measures contour complexity is:

$$O_{cont}(R_1, R_2) = max(0, \Delta P(R_1, R_2)) \quad (3.3)$$

The contour term promotes the creation of smooth contours between regions. Since most objects are regular and compact (that is, tend to have simple contours), the analysis of shape complexity can provide additional information for the mergings.

Color and contour similarity measures are linearly combined to form the *bpt_nwmc* similarity criterion:

$$O_{bpt\_nwmc}(R_1, R_2) = \alpha\, O_{wedm}(R_1, R_2) + (1 - \alpha)\, O_{cont}(R_1, R_2) \quad (3.4)$$

The *bpt_nwmc* similarity criterion creates color homogeneous regions with smooth contours, but tends to create elongated regions. As the regions will be used for coding purposes, more compact regions are desirable. To this end, a new term which measures the spatial proximity is added based on the distance between region centroids. The centroid of a region is defined as:

$$Cent(R_1) = \frac{1}{N_{R_1}} \sum_{p \in R_1} Coord(p) \quad (3.5)$$

where $Coord$ are the coordinates of the pixels p in the region R_1. The *cent* is defined as the euclidean distance d between centroids:

$$O_{cent}(R_1, R_2) = d\,(\,Cent(R_1),\, Cent(R_2)) \quad (3.6)$$

The *Cent* criterion is combined with the *bpt_nwmc* to form the *bpt_spx* superpixel similarity criterion:

$$O_{bpt\_spx}(R_1, R_2) = \beta\, O_{Cent}(R_1, R_2) + (1-\beta)\, O_{bpt\_nwmc}(R_1, R_2) \tag{3.7}$$

After testing different weights for α and β, it was found that the weight factors in equations (3.4) and (3.7) obtain similar results in terms of boundary retrieval. For simplicity, the three terms are combined without weights, creating the superpixels criterion *bpt_spx* (3.8):

$$O_{bpt\_spx}(R_1, R_2) = O_{Cent}(R_1, R_2) + O_{wedm}(R_1, R_2) + O_{Cont}(R_1, R_2) \tag{3.8}$$

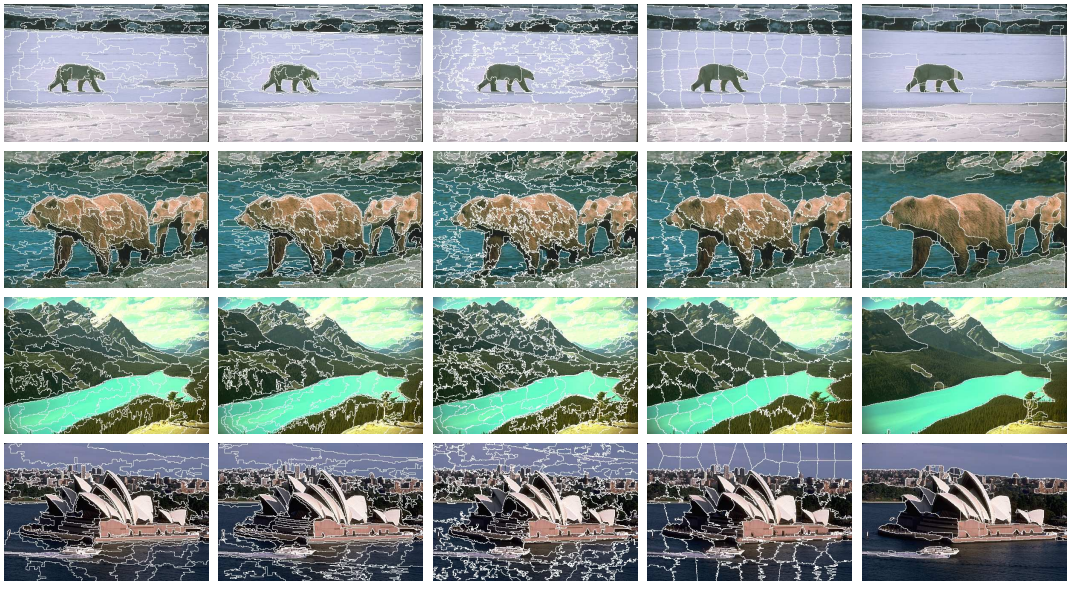


Figure 3.2: Color segmentation: visual comparison.

As stated before, the number of regions in the color partition, N_{color}^{regs}, is an important parameter of the coding systems proposed. The *bpt_spx* criterion builds the hierarchy until N_{color}^{regs} regions are obtained. Visual results for the different methods are shown in Figure 3.2.

3.2 Color Segmentation Evaluation

The color image segmentation is evaluated against other superpixels segmentations to validate the use of the new centroid distance term. The benchmark for this work consist of the BSDS500 dataset [MFTM01] which contains 500 images with human-marked-boundaries as ground-truth. The *bpt_spx* method is compared with the BPT merging criterion *bpt_nwmc* [VMS08], with two state of the art superpixel methods, SEEDS [BBR+13] and SLICS [ASS+12], and with the *UCM* [AMFM09], that obtains a hierarchical structure similar as the one derived with the BPT.

Figure 3.2 shows visual results of the methods compared in this work. The regions generated with *bpt_spx* present smoother contours than the *bpt_nwmc* criterion due to the centroid term. This term promotes compactness in the first stages of the hierarchical segmentation which leads to smoother contours. SLICS segmentation recovers even simpler contours, achieving partitions with less false boundaries at the cost of losing also some meaningful contours. The objective of *bpt_spx* differs from the one of *UCM* since, on the one hand, the *UCM* partition aims to obtain a partition that represent the objects of the scene with minimal regions while, on the other hand, the aim of the color segmentation is to achieve a superpixel representation for coding purposes.

Figure 3.3 shows numerical results in terms of precision, recall and F-measure (see Appendix C.2 for an explanation of the evaluation metrics used in this Thesis) for boundaries between segmentation and ground-truth. In *bpt_spx*, the use of the centroid slightly decreases the recall with respect to the *bpt_nwmc* criterion but the precision is improved, obtaining a result similar to the segmentation obtained with SLICS. Globally, the usage of the centroid criterion achieves a better trade-off between superpixel compactness and boundary adherence than *bpt_nwmc*.

The F-measure of *bpt_spx* results are comparable to SLICS, but as the main objective of the color segmentation is to procure the maximum number of contours, a higher recall is desired. Since the *UCM* representation promotes a represen-

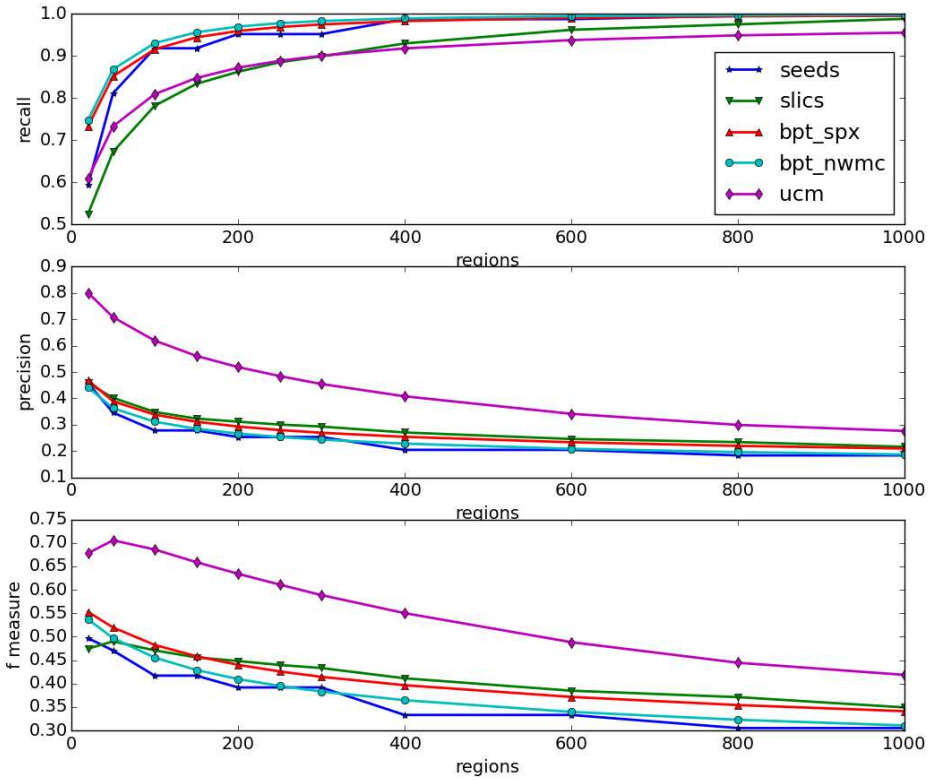


Figure 3.3: Evaluation of different methods for the color segmentation

tation where each region is meaningful, the number of boundaries that are not from the object is lower, obtaining a much higher precision figure. On the other hand, the *UCM* obtains smooth contours which occasionally are slightly displaced from the ground-truth, leading to lower recall measure. Moreover, the high computational requirements of the *UCM* make their use costly for a video coding scheme.

3.3 Depth Map Segmentation

The color segmentation provides an initial segmentation for depth maps. However, when color and depth images present structural differences, not all the depth edges can be retrieved from the color image. Since edges from depth

segmentation have to be encoded, the objective is to find a depth partition able to represent the depth map image using the lowest number of depth edges. In this Section, a method that uses a 3D planar model to represent the regions in the BPT is proposed.

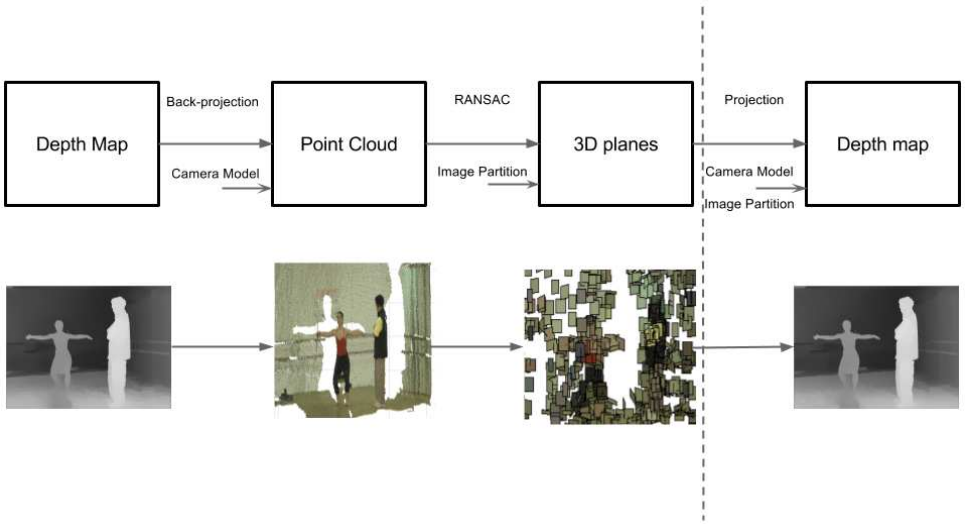


Figure 3.4: Depth maps are back-projected to the 3D world using the camera parameters of the view. Using an initial segmentation, each region is represented with a 3D planar model. From the 3D planar models the depth map can be projected to recuperate the depth map.

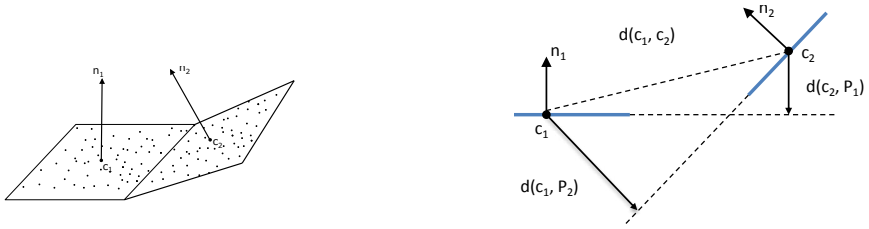
The depth map partition is created in three steps. Depth maps used in multi-view scenarios are often noisy and quantized into 8 bits. To avoid discontinuities in the surface normals, an initial superpixel over-segmentation of the depth map is performed as an initial step. This severely over-segmented partition ($\sim$10000 regions) removes noise and quantization errors while capturing the scene with sufficient detail.

Using this initial over-segmentation, the 3D point cloud is generated by computing the centroid of all the points in the region and back-projecting them to the 3D world using the mean value of all the pixels in the region, as shown in Figure 3.4. This creates a 3D point cloud where each point of the point cloud is related to a region. As the point cloud is formed with the depth map of one viewpoint, only the nearest surfaces of the scene are present in the point cloud.

The second step consist in performing the region growing algorithm in [RvdHV06] to the 3D point cloud generated in the first step to obtain smoothly connected areas. The algorithm computes the local surface normals and uses the point connectivity to join 3D points that have the same orientation. The method requires a small number of parameters -number of neighbors, angle orientation tolerance and a smoothness threshold to start new regions-, which provide a trade-off between under- and over-segmentation.

The partition obtained after the region growing step is able to recover the structure of the scene but still has some small-sized regions which have not been merged as shown in Figure 3.6.a. The third step of the algorithm finds the final partition with a BPT. A new model and similarity criterion is proposed to deal with depth maps.

A 3D planar region model is chosen as a model for the merging process. In this model, each region is characterized by the centroid of the 3D points of the region c_i and the normal orientation n_i of the plane that best fits the 3D points, as shown in Figure 3.5.a. Each region model is formed by fitting a 3D plane to all the 3D points of the region using RANdom SAmple Consensus (RANSAC) [FB81].



a) b)

Figure 3.5: a) Region model: plane with normal n_i and centered in c_i. b) Distances in the 3D similarity criterion.

The similarity criterion combines two different similarity measures between regions R_1 and R_2: *orientation* and *3d-distance*. The *orientation* similarity indicates whether the centroid of one plane is well approximated by the neighboring

plane equation:

$$O_{orientation}(R_1, R_2) = a_1 * d(c_1, P_2) + a_2 * d(c_2, P_1) \tag{3.9}$$

where a_i is the area of the region in number of pixels, $d(c, P)$ is the euclidean distance between a point c and a plane P.

The measure *3d-distance* is based on the euclidean distance between centroids. This measure has a similar effect than the *cent* term in the *bpt_spx* criterion. *3d-distance* promotes the creation of regions that are closer in the 3D space.

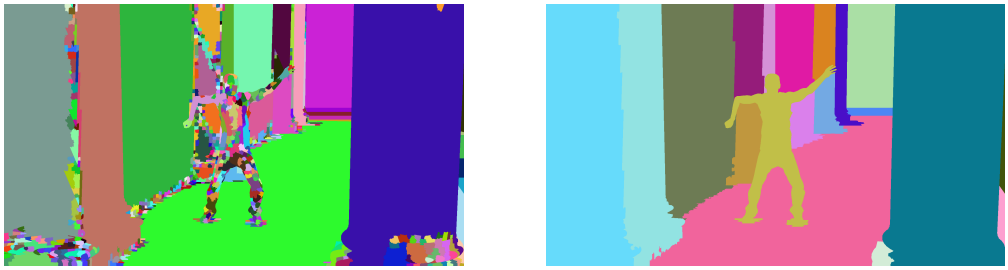
$$O_{3d-distance}(R_1, R_2) = \frac{a_1 + a_2}{2} * d(c_1, c_2) \tag{3.10}$$

The final similarity criterion is defined as:

$$O_{3d-bpt}(R_1, R_2) = O_{orientation}(R_1, R_2) + O_{3d-distance}(R_1, R_2) \tag{3.11}$$

Figure 3.5.b shows a graphical example of the *3d-bpt* criterion.

At each iteration of the merging process, the algorithm selects the pair of regions with the lowest O_{3d-bpt}, which corresponds to the most similar pair of regions, and merges them into a new region. An example of final partition obtained with the algorithm is depicted in Figure 3.6.b.



a) b)

Figure 3.6: Depth Map Partition process. a) Result of applying the region growing algorithm. b) Final coding partition after the 3*d_bpt* algorithm.

3.4 Depth Map Segmentation Evaluation

The depth map segmentation proposed is evaluated with 25 depth maps from the MVD sequences *undo dancer*, *balloons*, *kendo*, *breakdancers* and *ballet*. For

more details about the MVD sequences see Appendix A. Results generated with the proposed scheme are compared against the segmentation produced with rgbd-UCM [GAM13] and with UCM computed using only the depth image. For rgbd-UCM, a hierarchical segmentation is generated using color and depth clues. Two variants of the 3D merging process are examined, with and without the region growing step, named 3d-rgb-rgrow and 3d-bpt respectively.

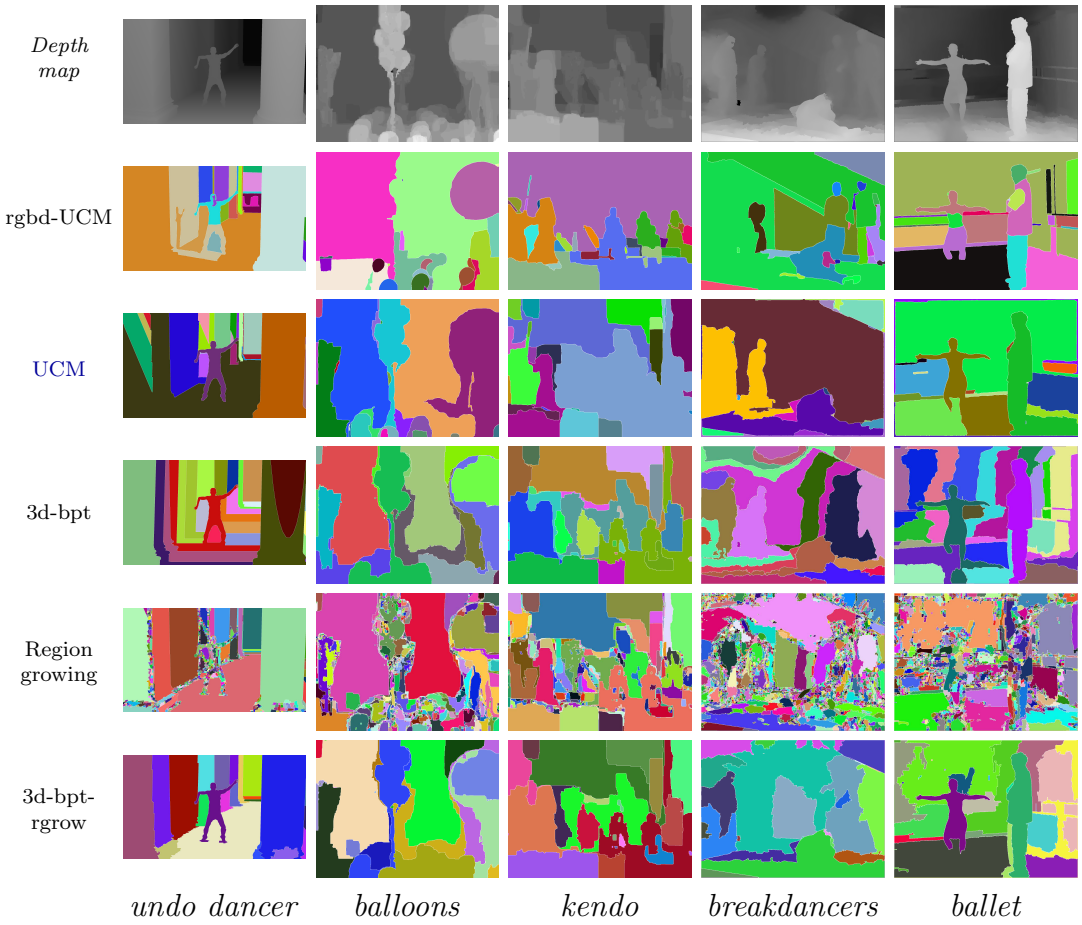


Figure 3.7: Visual comparison between the depth segmentation techniques. In row ascendent order: Depth map to segment, rgbd-UCM, UCM, 3d-bpt, Output of Region growing segmentation stage and 3d-bpt-rgrow.

Figure 3.7 shows the partitions obtained with the different methods. By using the color image in addition of the depth image, the rgbd-UCM generally is able to represent the foreground objects with more regions. Despite that, when the depth map is noisy, the rgbd-UCM fails at obtaining the main depth edges as

can be seen in the balloons image in Figure 3.7. Using the standard UCM on the depth image this over-segmentation is reduced.

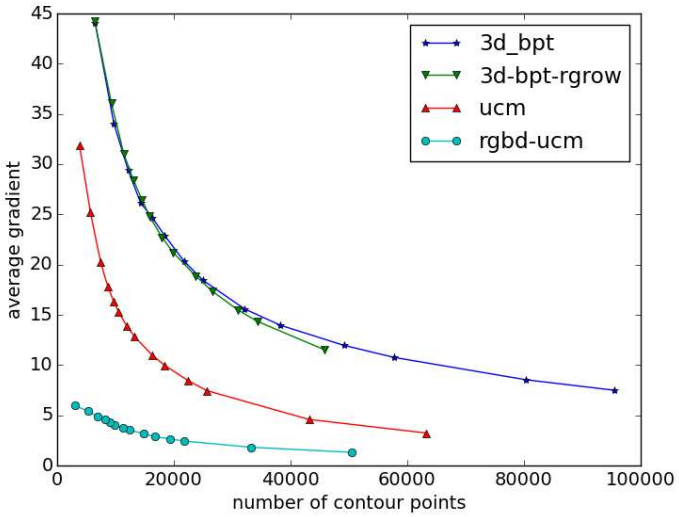


Figure 3.8: Gradient measure in function of the number of contour points in the segmentation.

The *3d-bpt* method obtains a representation of the scene where objects in the same depth are correctly separated but has problems at recovering the overall structure of the scene. This can be seen in the *undo dancer* image in Figure 3.7 where regions are created at increasing depths values, joining the walls and the floor. The region growing stage creates an initial segmentation where the flat areas of the scene are joined in a unique region. From the region growing segmentation, the *3d-bpt* criterion merges the regions in non-smooth areas with the best 3D plane model.

The evaluation of the different depth segmentation techniques is computed using a gradient measure in the contour points. The purpose of the depth segmentation is to provide the main depth edges of the depth map with the minimum contour points. The maximum directional gradient (horizontal or vertical) is computed for each contour point and then averaged. This measure is computed at different cuts of the hierarchy. Notice that this metric is helpful to determine if areas at different depth distances are in different regions. However it

cannot measure the areas where there is not a depth edge but a change of the orientation, as the joints between floor and walls.

In Figure 3.8, the average results for the different sequences are shown. The 3d-bpt and the 3d-bpt-rgrow obtain better results than the rgbd-UCM and UCM. In the two UCM options the contours obtained are smooth, losing some meaningful depth boundaries. The rgbd-UCM obtains even lower gradient measure since it uses also the color image to generate the hierarchical partition.

While the results are similar for the two BPT options, the 3d-bpt-rgrow is selected by its better 3D scene reconstruction. The separation of walls and floor is more desirable from a conceptual point of view more than the edges that are in the middle of them.

Chapter 4

2D Single-View

In this Chapter, the two 2D depth coding methods developed during this Thesis for single-view coding are presented. The first one, *Color-Based Hierarchical Optimization*, finds the final coding partition with an optimization over a hierarchy of regions using only a color partition. By using only the color partition, the depth map can be encoded without explicitly signaling the depth boundaries. However, as the color partition is not always able to retrieve all depth edges, depth contours are present inside the regions. This leads to use complex texture models, increasing the texture bitrate. In the second method,*Fusion of Color and Depth Partitions*, depth contours are encoded explicitly, providing depth edges when color and depth boundaries are inconsistent.

4.1 Color-Based Hierarchical Optimization

The first method is based on segmenting the depth map using only color information. Depth maps are segmented using a color partition into homogeneous regions of arbitrary shape. The contents of these regions are coded using texture coding techniques. Having uniform regions allows coding the texture inside the region with few coefficients. Using only color partition saves the high cost associated to coding the resulting partitions (region shape) in segmentation-based coding techniques. Instead of directly segmenting the depth map and sending the partition to the receiver, an approximate depth partition is constructed

using the decoded color image which is supposed to be available at the receiver.

The final coding partition is obtained with an optimal lagrangian approach applied to a hierarchical region representation. The lagrangian approach (presented in Subsection 4.1.1) presents two benefits. Firstly, the encoding parameters are selected optimally for each region and secondly, the optimal final number of regions needed to encode the depth map can be automatically obtained. Furthermore, two modes of the Shape Adaptive Discrete Cosine Transform (DCT) [SM95] are used to obtain a better rate-distortion trade-off.

The segmentation technique used in this work is the BPT with the similarity *bpt_nwmc* criterion defined in Section 3.1. The proposed *bpt_spx* similarity criterion in 3.1 was developed later in the thesis. The *bpt_nwmc* can be computed using color information (*bpt_nwmc_color*) or using depth information (*bpt_nwmc_depth*).

4.1.1 Hierarchical Rate-distortion Problem Definition

The rate-distortion problem consists in finding the optimal coding that minimizes the distortion D of the image with the constrain that the total cost R is below a given budget [OR98]. In the hierarchical case, this optimal coding consists in finding the optimal partition with the coding technique for each region to describe the image, this is, a set of regions in the hierarchy H. The rate-distortion function has to be defined (see [OR98]) with R and D additive measures in H:

$$R_H = \sum_{regions} R_k \tag{4.1}$$

$$D_H = \sum_{regions} D_k \tag{4.2}$$

Each node of the hierarchy can be encoded using a set of coding techniques generating a set of $R_{k,q}$ and $D_{k,q}$ for each region k and each coding technique q. In this Thesis, the distortion measure $D_{k,q}$ is computed as the Square Error

between the estimated texture values and the real ones:

$$D_{k,q} = \sum_{n=1}^{N_k} |\hat{g}_q(n) - g(n)|^2 \tag{4.3}$$

where N_k is the number of pixels of the region k, g is the texture value of the pixels of the region and $\hat{g}_q$ is the estimated values with the q texture coding option.

Typically, in segmentation-based coding, the rate to encode a region is composed of two terms: The cost of the contours of the region ($R^C_{k,q}$) and the cost of the texture coefficients ($R^T_{k,q}$). Hence, for each region their rate cost $R_{k,q}$ is defined as:

$$R_{k,q} = R^C_{k,q} + R^T_{k,q} \tag{4.4}$$

The constrained problem can be converted into an equivalent unconstrained one by using Lagrangian relaxation. Cost and distortion are combined using a positive multiplier λ that defines the trade-off between the distortion allowed and the number of bits spent:

$$J_{k,q} = D_{k,q} + \lambda R_{k,q} \tag{4.5}$$

Figure 4.1: Lagrangian optimization over hierarchies. The bottom-up analysis follows the merging sequence from Figure 3.1 activating the nodes with minimum lagrangian. The algorithm finds the best partition (regions 3, 4 and 5) and the best coding strategy.

Assuming that the optimum λ^* is known, the first step is to make a local analysis and to compute, for each node of the BPT, the Lagrangian for each coding technique (or quantizer level). The coding technique giving the minimum Lagrangian is considered as the optimum one for this node:

$$J_k = \arg\min_q J_{k,q} \tag{4.6}$$

The second step is to define the best partition. This can be done by a bottom-up analysis of H. Starting from the lowest level, one checks if it is better to code the area represented by the children regions as a single parent region or as individual regions $child_1$ and $child_2$. The best choice is selected by comparing the Lagrangian of the parent with the children's one:

$$J_{parent} \leq J_{child_1} + J_{child_2} \tag{4.7}$$

where J_{child} is the cost of the cheapest path under the child node.

If the left term is lower than the right term, the parent node is activated and the children ones are deactivated. This inequality is evaluated following the merging sequence up to the root node. At the end of the procedure, the best partition for the given budget is defined by picking up all the regions corresponding to the activated nodes together with their corresponding best coding technique (defined during the first step of the algorithm). The definition of the optimum parameter λ^* can be done, for instance, with a gradient search algorithm. Figure 4.1 shows a result of a lagrangian optimization applied to the hierarchy depicted in Figure 3.1.

4.1.2 Encoding process

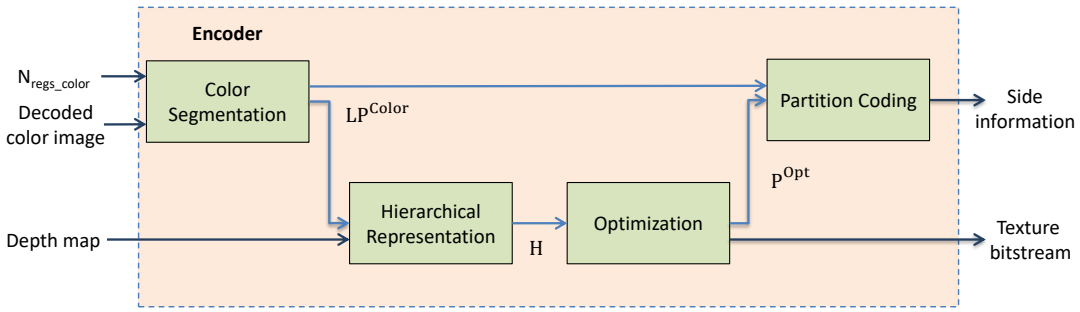


Figure 4.2: Block diagram for the hierarchical representation.

The creation of the coding partition is depicted in Figure 4.2 and can be described in four steps:

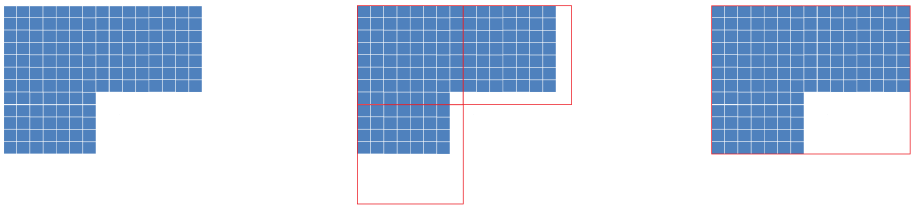
Color segmentation: A fine initial color partition (LP^{Color}) is built in the same way both at the encoder and the decoder using the decoded color view (using the *bpt_nwmc_color* similarity criterion from Section 3.1). LP^{Color} is an over-segmented approximation of the final coding partition. The assumption here is that depth transitions coincide in most situations with color transitions. Since LP^{Color} can be constructed both at the encoder at the decoder, it has no coding cost. Experiments have shown that the proposed coding technique is robust to the number of regions in this initial partition and simply, a high enough number to prevent under-segmentation is enough (see Figure 4.4.a).

Hierarchical representation (depth criterion): Starting from LP^{Color}, a BPT hierarchy H is obtained at the encoder side by using a depth-based similarity (*bpt_nwmc_depth*) as a merging criterion. Using an initial partition constructed using only depth information (instead of LP^{Color}) would have all depth edges, but in that case, the shape of this partition would need to be sent, at a high coding cost.

Optimization: The rate-distortion optimization presented in Subsection 4.1.1 selects the optimum combination of regions and coding techniques/quantizer over H. In this work, a SA-DCT [SM95] has been used as the texture coding strategy. The SA-DCT encodes the texture (in this case the depth associated with the region) of any arbitrary shaped segment.

Two modes of the SA-DCT have been used. The first mode, $SA\text{-}DCT_{blocks}$, divides the segmented region in blocks of $8x8$ pixels and each block is encoded using SA-DCT or standard DCT. The SA-DCT is used in blocks that correspond to a boundary of the region while the standard DCT is used in blocks where all pixels belong to the region. The second mode, $SA\text{-}DCT_{single}$, is intended to handle homogeneous regions in an efficient manner, hence encodes all the segmented region in a single SA-DCT, giving a rough approximation to the texture with few coefficients. Example blocks for each SA-DCT configuration are shown in Figure 4.3.

Both modes are available to the lagrangian multiplier optimization process that



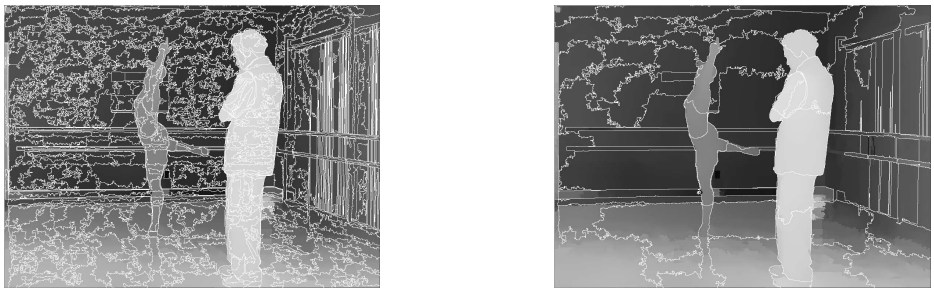
a) Example block b) $SA\text{-}DCT_{blocks}$ c) $SA\text{-}DCT_{single}$

Figure 4.3: Example of coding a block with $SA\text{-}DCT$ methods. $SA\text{-}DCT_{blocks}$ divides the segmented region in blocks of $8x8$ pixels while $SA\text{-}DCT_{single}$ encodes all the segmented region in a single SA-DCT.

selects the optimal one, region by region, following the optimization described in Subsection 4.1.1. As the partition is fully color-based, the contour cost $R^C_{k,q}$ is 0 bits for all regions. Once the optimization process finds the optimal partition P^{Opt}, the texture coefficients from active regions are grouped and entropy coded using an adaptive arithmetic codec [WNC87].

Partition Coding: As the hierarchical representation and optimization steps use depth information (not available at the decoder) to construct P^{Opt}, side information has to be provided to generate P^{Opt} at the decoder. The side information consist on the sequence of mergings that the decoder has to perform to generate P^{Opt} from LP^{Color}. To do that, an alternative hierarchy H^C is constructed starting from LP^{Color}, but this time, a color criterion (*bpt_nwmc_color*) is used instead of the depth (*bpt_nwmc_depth*) criterion. Being purely based on color information, H^C can be constructed at the decoder.

At the encoder, the sequence of mergings in H^C and H are compared. For each merging in H^C, a bit of side information is sent. If the regions merged at a given step in H^C are also merged in the P^{Opt}, a '0' is sent. Otherwise, a '1' is sent. This way, the decoder can prevent the mergings in H^C that are not used to construct P^{Opt}, and this way, P^{Opt} can be constructed at the decoder. Experimental results have shown that using this approach instead of directly encoding the region contours greatly reduces the amount of bits needed to encode the partition. The amount of bits needed to replicate the hierarchical structure is less than 0.01 bits per pixel in the sequences used.



a) LP^{Color} b) P^{Opt}

Figure 4.4: a) Example of a LP^{Color} with 500 regions for image *Ballet*. b) P^{Opt} selected by the Lagrange optimization for the same image.

Figure 4.4.b shows an example of the partition selected by the lagrangian optimization for the image *Ballet*. At the end, the total encoding rate for all regions is formed by the sum of each of the regions texture cost R_k^T plus the coding cost of sending the side information that will allow the decoder to build P^{Opt}.

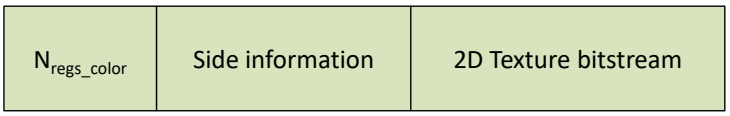


Figure 4.5: Bitstream containing the number of regions for the color image, the side information needed to recover P^{Opt} and the SA-DCT coefficients.

4.1.3 Decoding Process

The decoder process is depicted in Figure 4.6. LP^{Color} is built from the decoded color image as done in the encoder. Then, the side information is used to replicate P^{Opt}: the decoder proceeds merging further regions using the color information (*bpt_nwmc_color*) criterion but, each time a merging is proposed by this criterion, a bit is pulled from the side information to decide if the merging is allowed. In case it is not, the algorithm proceeds with the next merging in the BPT.

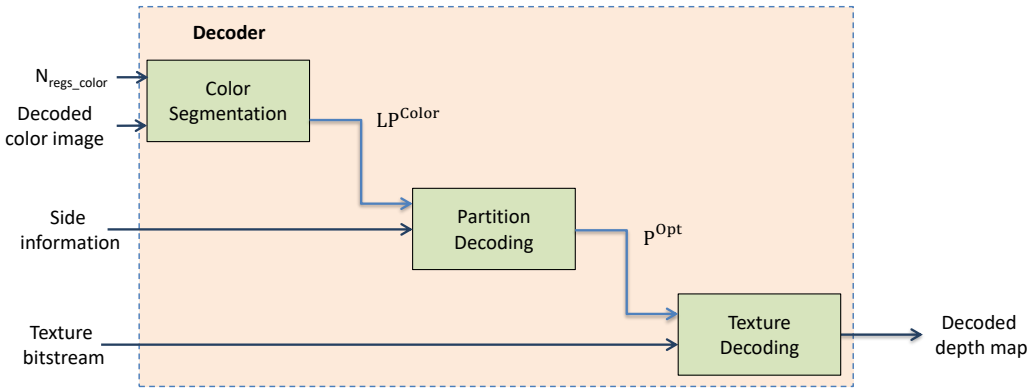


Figure 4.6: Block diagram for the decoding process.

4.2 Fusion of Color and Depth Partitions

In Section 4.1, a segmentation technique has been used to create a partition of the decoded color image, already available at the decoder. This partition is used to approximate the location of the depth edges. This allows using region-based texture coding techniques without the burden to encode the full depth partition. An evolution of this method was presented in [MMRH13] with two main contributions.

The first contribution is to use the color segmentation criterion *bpt_spx* presented in Section 3.1. It was introduced to promote compact-shaped regions. The second contribution is related to the assumption of co-occurrence of depth and color contours in [MRHM12]. While in many cases depth and color contours are located at similar locations, there are small but noticeable differences that result in regions that contain depth discontinuities. In this case, the lack of homogeneity results in a poor performance of the region-based texture coding techniques. To solve that, it is proposed to use a combination (fusion) of a fine color partition (denoted as P^{Color}) and a coarse depth partition (denoted as P^{Depth}) which contains the location of the main edges, providing information about the boundaries when the color and depth discontinuities are inconsistent. In Figure 4.8, partitions obtained for the color image and the depth map are shown.

Unlike the previous work presented in Section 4.1, a rate-distortion optimization

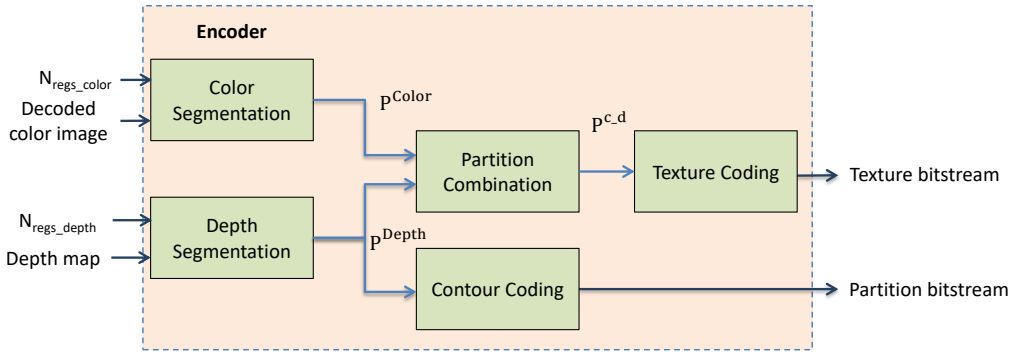
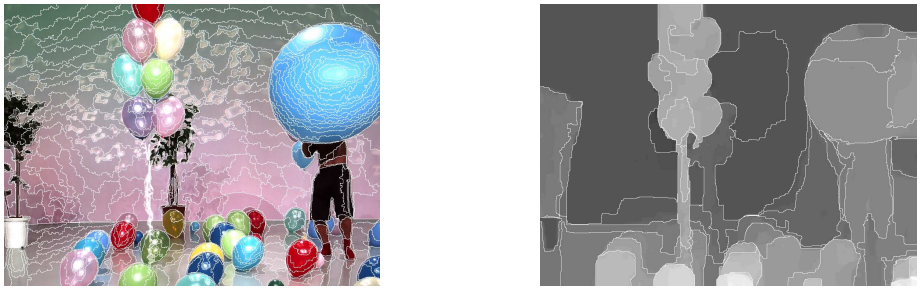


Figure 4.7: Scheme for the encoding process. The two partitions obtained with the color image and the depth map are fused to a single partition which is used in the encoding of the depth map.

is not performed in the hierarchical structure obtained with the region-merging algorithm. Here the focus is on improving the coding segmentation by adding depth boundaries, therefore the texture coding is performed in a single layer of the hierarchical representation.

4.2.1 Encoding process



a) P^{Color} b) P^{Depth}

Figure 4.8: a) Color-based partition for sequence *balloons*. b) Depth-based partition for sequence balloons.

The scheme of the encoder process is shown in Figure 4.7. P^{Color} is built with the *bpt_spx* presented in Section 3.1. For P^{Depth}, the similarity criterion involves only texture information (from the depth map). The criterion used is the squared error (*bpt_se*) [VMS08] which do not take into account the size of the regions to be merged, which is preferable in the scarcely textured depth

maps. The region model M (see Section 3.1) is constant for all the pixels of the region $R_1 \cup R_2$ and is compared to the original depth image I:

$$O_{bpt\_se}(R_1, R_2) = \sum_{p \in R_1 \cup R_2} (I(p) - M_{R_1 \cup R_2})^2 \tag{4.8}$$

4.2.1.1 Partition Combination

Both partitions P^{Color} and P^{Depth} are combined to form a new partition P^{c_d}. The P^{c_d} partition is built by taking all the P^{Color} and P^{Depth} boundaries:

$$P_i^{c\_d} = P_i^{color} \bigcap P_i^{depth} \tag{4.9}$$

Typically, P^{Color} presents a more regular segmentation and also contains a good number of depth edges, while P^{Depth} contains the more important depth edges. To ensure the creation of meaningful regions, only new regions that are larger than a certain size are created. Figure 5.2.b shows the resulting P^{c_d}, distinguishing the boundaries from P^{Color} and P^{Depth}. With the addition of the edges from P^{Depth}, the inconsistencies between color and depth map that lead to under-segmentation errors are solved.

4.2.1.2 Contour and texture coding

The partition contours are coded with a modified Freeman Chain Code technique [Fre61] [MSG93]. See Subsection 2.3.1 for details of Freeman Chain Code technique.

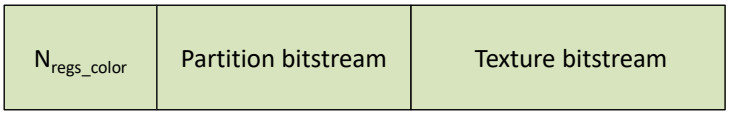


Figure 4.9: Bitstream containing the number of regions for the color image, the contour coefficients to recover P^{Depth} at the decoder and the SA-DCT coefficients.

The SA-DCT technique [SM95] is used to encode the texture within each region. The absence of transitions inside regions allows coding the texture information with few coefficients. In this case the mode used is the $SA\text{-}DCT_{blocks}$ (see 4.1),

which divides the segmented region in blocks of $8x8$ pixels and each block is encoded using either $SA\text{-}DCT$ or standard DCT. $SA\text{-}DCT_{blocks}$ have been used because achieves higher quality than $SA\text{-}DCT_{single}$.

4.2.2 Decoding Process

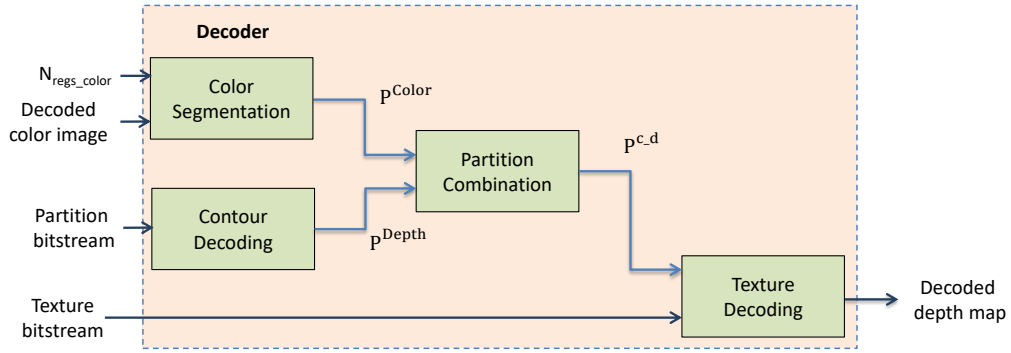


Figure 4.10: Scheme for the decoding process. The color partition is obtained from the color image while the depth partition is received for the decoder.

The decoder receives the bitstream depicted in Figure 4.9. The decoder is able to replicate the color partition as it has been created from the encoder since solely the color image is used. Using the contour information received, the depth map partition is generated. The fusion segmentation is built using the same process as described for the encoder. The complete design of the decoder is presented in Figure 4.10.

4.3 2D Single-View Depth Map Coding Results

The 2D coding methods presented in this Chapter are evaluated using 10 images of each of the multi-view sequences sets *ballet*, *undo dancer*, *kendo* and *balloons*. The Color-Based hierarchical optimization method (Section 4.1) will be referred as *Color opt*, and the fusion of color and depth partition method (Section 4.2) will be referred as *Fusion*. *Color opt* results are obtained with initial partitions

of 2000 regions. Results for the *Fusion* method are obtained using a fixed number of regions in the partitions of 50 regions in the depth segmentation and 100 regions in the color segmentation. The number of regions in the depth segmentation has been found to be sufficient to find the main edges on the image. Notice the difference of the number or color regions used in *Color opt* and *Fusion*. As the *Color opt* method does not use a depth partition, the number of regions in the color partition is higher to obtain as many depth contours as possible.

Color opt and *Fusion* methods use SA-DCT as a coding technique. Figure 4.11 shows the coding performance of the two SA-DCT configurations for the *Color opt* method. Depth map coding results are obtained with different configurations: $SA\text{-}DCT_{single}$ and $SA\text{-}DCT_{blocks}$ encode the LP^{Color} partition with a fixed coding technique (varying the quantizer level) whereas the other approaches uses the lagrangian optimization to select the best choice of coding technique/quantizer level. The *lagrangian* optimization finds the best trade-off for each region using the hierarchy H. The *lagrangian no hierarchical* uses the lagrangian optimization but only exploring LP^{Color}. The lagrangian

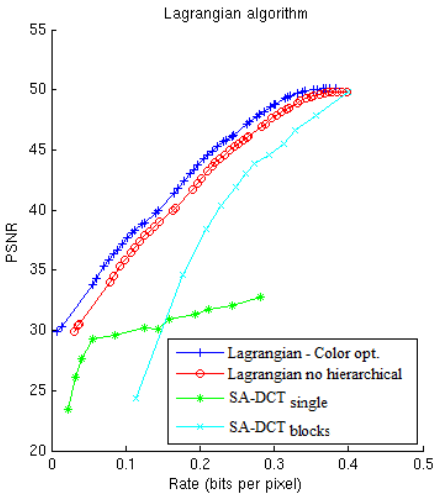
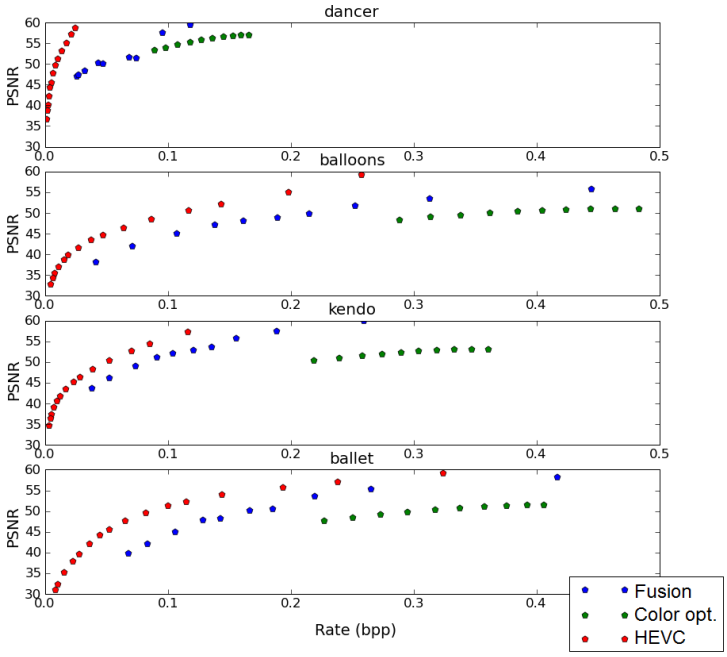
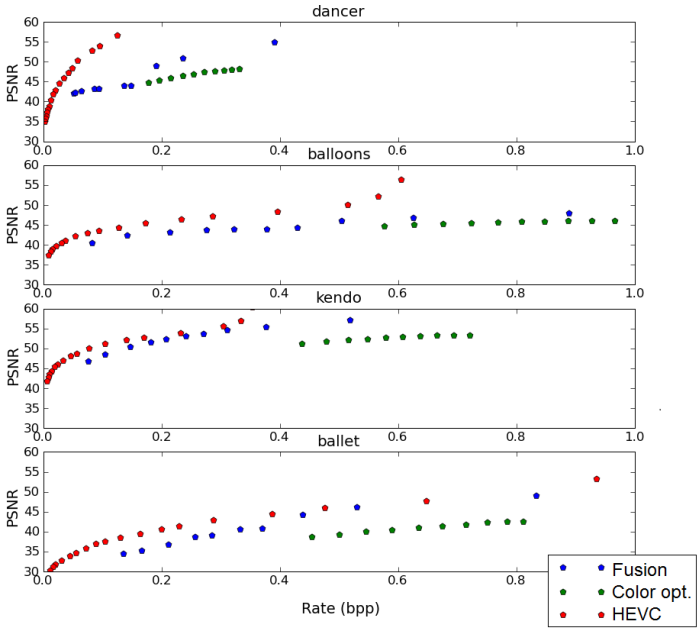


Figure 4.11: Rate-distortion over depth maps. The proposed *lagrangian* optimization (*Color opt*) is compared with using only the leaves partition in the lagrangian optimization (*lagrangian no hierarchical*), with $SA\text{-}DCT_{single}$ and with $SA\text{-}DCT_{blocks}$



PSNR over depth map



PSNR over virtual view

Figure 4.12: Peak Signal-to-Noise Ratio (PSNR) over depth maps and in virtual views of *Color opt*, *Fusion* and HEVC.

optimization combines regions coded with $SA\text{-}DCT_{single}$ and $SA\text{-}DCT_{blocks}$ optimally, obtaining increased rate-distortion figures than obtaining any of the methods separately. Moreover, the use of the hierarchy explores a larger number of coding options, leading to an increase of 1dB compared of using the *lagrangian no hierarchical* configuration.

As the proposed algorithms are image based, intra-mode comparison with the main intra-mode profile HEVC is provided. A description of the sequences used is provided in Appendix A. The description of the experimental configuration is described in Appendix B. Error measures (defined in Appendix C) are computed between pairs of synthesized virtual views. The first one is always obtained using the uncompressed depth maps, while the second one is generated, respectively, with HEVC, *Color opt* and *Fusion* methods.

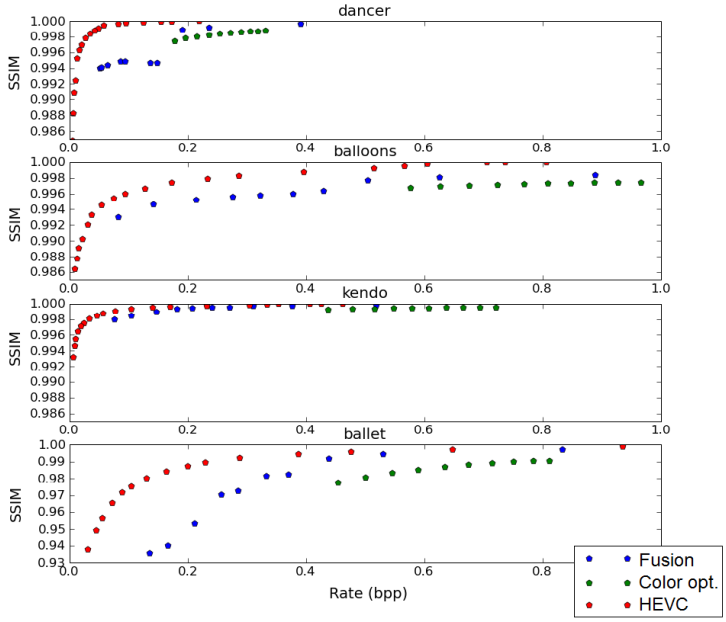


Figure 4.13: Structural SIMilarity (SSIM) over virtual views of *Color opt*, *Fusion* and HEVC.

Rate-distortion results are presented in both Peak Signal-to-Noise Ratio (PSNR) and Structural SIMilarity (SSIM) depending on the number of bits per pixel used to encode the depth map. For the *Color opt* different rate-distortion points are found using different λ parameters. For the *Fusion* method, the

	undo dancer	balloons	kendo	ballet
Depth Map				
Color opt	-12.39	-10.93	-12.76	-8.79
Fusion	-15.81	-6.94	-6.70	-5.00
Virtual View				
Color opt	-8.43	-6.48	-7.91	-4.97
Fusion	-12.66	-4.46	-3.42	-2.96

Table 4.1: Bjontegaard measures with respect to HEVC.

cost of encoding the contours given the depth segmentation is fixed whereas the texture cost depends of the quantization step selected for each region. Results are shown in Figure 4.13 and Figure 4.12. Bjontegaard delta PSNR (BD-PSNR) [Bjo01] (see Appendix C) results are summarized in table 4.1. BD-PSNR figures are presented with respect to HEVC.

In both SSIM and PSNR the *Fusion* proposal outperforms *Color opt*, showing the need of using depth contours to increase the quality of the color partition. The BD-PSNR improvement of the *Fusion* method with respect of the *Color opt* is in the margin of 3 dB. The only exception is for the *undo dancer* sequence where, for low rates, the quality is similar to the previous method, thus diminishing the BD-PSNR measure for that sequence. In the *Color opt* method, the rate-distortion optimization prevents that behavior.

The advantages of a segmentation-based technique can be seen in the better performance obtained with the proposed techniques in virtual views. Despite results of the two 2D methods are below HEVC ones, segmentation-based coding techniques explored in this Chapter provide foundation for next Chapters showing the necessity to combine color and depth map partitions to increase the coding efficiency and the benefits in using optimization methods over a hierarchy of regions.

Chapter 5

3D Single-View

In this Chapter, a 3D depth map coding technique for a single-view is presented (*3d Single-View*). The segmentation algorithms presented in Chapter 3 are used to independently segment the color image and the depth data. Similarly to the *Fusion* method presented in Section 4.2, the color and depth partitions are combined to obtain the final coding partition. In this case, to reduce the cost of encoding depth boundaries, only the main depth edges are added. From the final partition, a 3D plane-based representation is introduced to store the scene structure. The main contribution of this Chapter is the introduction of the 3D planes to represent the depth map, which facilitates the multi-view extension in Chapter 6.

5.1 3D Single-View Depth Map Coding

The *3d Single-View* depth map coding technique uses an image partition to fit a 3D plane for each region. Pixels belonging to each region are back-projected into the 3D space and then encoded using a 3D plane. 3D planes are able to represent the smooth texture of depth maps with few coefficients. Assuming that most depth edges in the depth maps are located in the same position as color discontinuities, a segmentation technique using the decoded color image allows to recover most of these depth edges.

5.1.1 Encoding Process

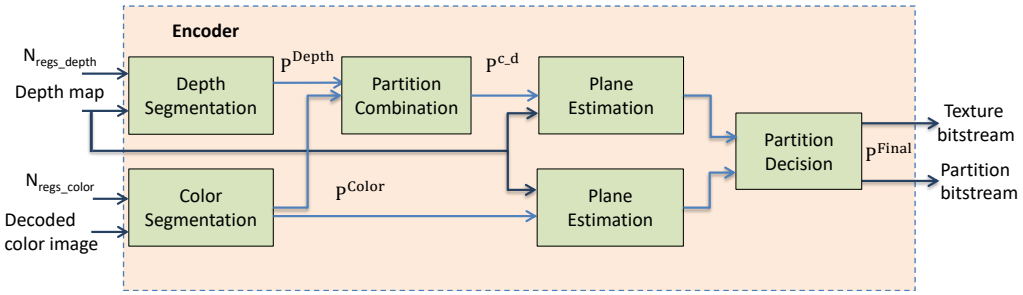


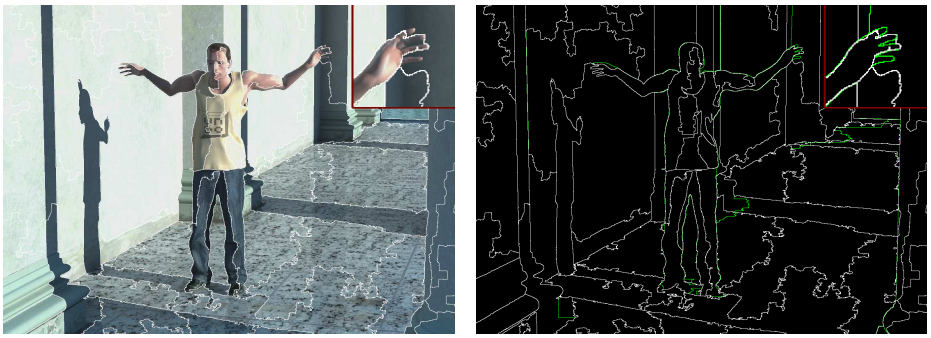
Figure 5.1: Encoder scheme. Color image and depth map are used to build two independent partitions. 3D planes are fitted using the color partition and the intersection of both partitions. The depth boundaries that solve the inconsistencies between color and depth partitions are found in a rate-distortion fashion and sent to the decoder.

The encoding process is depicted in Figure 5.1. The encoder uses both the decoded color image and the original unencoded depth map to build two partitions with the *bpt_spx* and *3d-bpt-rgrow* methods explained in Sections 3.1 and 3.3 (referred as P^{Color} and P^{Depth}). P^{Color} provides most of the depth boundaries and can be built also at the decoder without any extra coding cost. P^{Depth} contains the most important depth boundaries (including the ones not present at the color partition). P^{Color} and P^{Depth} are combined to form a new partition P^{c_d} by taking all boundaries of both partitions as explained in Section 4.2.1.1. An example of the partition combination process is shown in Figure 5.2.

A difference of the *fusion* method of Section 4.2, the *3d Single-View* only encodes the most relevant depth edges. To obtain these edges, two 3d planes representations are built, one using P^{Depth} and the other using P^{c_d}. Once each 3d plane estimation is done (Section 5.1.1.1), the main depth edges are obtained analyzing both 3D representation with a Partition Decision (Section 5.1.1.2).

5.1.1.1 Plane Estimation

A 3D representation is formed by fitting a 3D plane for each region using RANSAC [FB81]. The performance of RANSAC algorithm relies heavily on



a) P^{Color} b) P^{c_d}

Figure 5.2: Partition combination. Contours from P^{Color} depicted in white; P^{Depth} contours depicted in green.

the threshold to determine the outliers. If the threshold is too strict, few points in every iteration will be found as inliers, thus the plane will be estimated with a reduced subset of the points of the region. On the contrary, if the threshold do not separate between inliers and outliers, the fitting will include the outliers, resulting in a poor plane estimation. The dependence of choosing an adequate threshold is removed by selecting different values and keeping the one that provides the best model, that is, the one that has lower mean square distance between the plane and the points of the region:

$$D_k = \sum_{n=1}^{N_k} |Proj(\Pi_k^{3D}, n) - g(n)|^2 \tag{5.1}$$

where Π_k^{3D} is the plane model for region k, N_k is the number of pixels of region k, $Proj(\Pi_k^{3D}, n)$ is the value of the projected 3D planar model to the pixel n of region k and $g(n)$ is the depth value of n.

In order to obtain a compact representation the 3D plane coefficients are represented using the distance from the plane to the camera and the plane orientation. The distance from the plane to the camera is converted to an alternative quantized representation using the distance to depth map conversion:

$$C_{dist} = \frac{1.0}{\frac{d(pl,c)}{(2^{N_{dist}}-1)} * (\frac{1.0}{MinZ} - \frac{1.0}{MaxZ}) + \frac{1.0}{MaxZ}} \tag{5.2}$$

where $d(pl, c)$ is the euclidean distance between the region plane and the camera, N_{dist} is the number of bits to be used in the quantization and $MaxZ$ and $MinZ$ are the maximum and minimum depth values of the image.

The plane orientation is stored in spherical coordinates with their 3D angles θ and ϕ:

$$\theta = arccos\left(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) \tag{5.3}$$

$$\phi = arctan\left(\frac{n_y}{n_x}\right) \tag{5.4}$$

The n_z component is pointed towards $z > 0$. The resulting angles have the following dynamic range: $0 \leq \theta \leq \frac{\pi}{2}$ and $0 \leq \phi \leq 2\pi$. Each angle is encoded with equal precision with a uniform quantizer.

5.1.1.2 Partition Decision

While adding edges from P^{Depth} removes under-segmentation and thus, reduces the coding distortion, these contours must be explicitly encoded, which increases the coding cost. Unlike the 2D method in Section 4.1, to achieve the budget rate, the method controls how the new boundaries are added, prioritizing the boundaries that have a larger impact to the coding of the depth map. New regions in P^{c_d} are classified according to the distortion reduction obtained when adding the corresponding depth boundary to P^{Color}. Depth distortion is computed with the Equation 5.1.

Different rate-distortion points are obtained by progressively adding region boundaries to P^{Color} until the budget rate for this image is reached, resulting in the final partition P^{Final}. These added region boundaries should be also encoded (with lossless Freeman Chain-Code technique [Fre61], see Section 2.3.1) and sent to the decoder. The bitstream sent to the decoder is depicted in Figure 5.3.

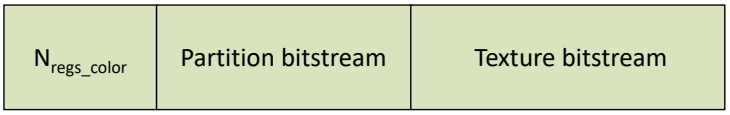
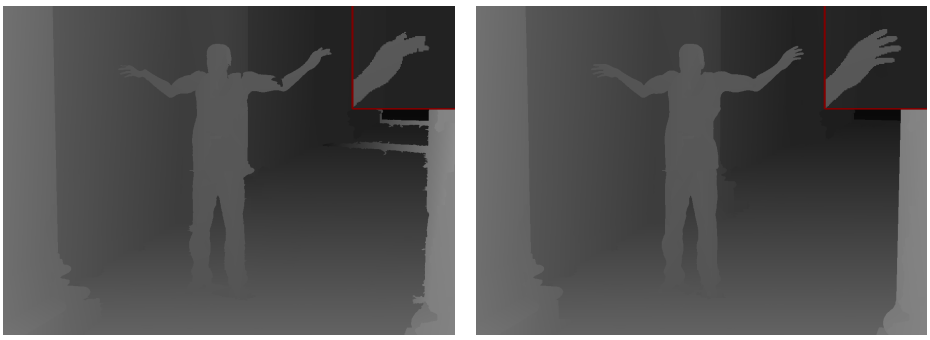


Figure 5.3: Bitstream containing the number of regions for the color image, the depth boundaries coded with chain-code and the 3D plane coefficients.

Figure 5.4 shows the resulting reconstructed planes projected onto the decoded depth map image. In Figure 5.4.a no depth contours are added to P^{Color} while in 5.4.b the P^{Final} is used. Using directly P^{Color} results in depth discontinuities inside the regions that lead to poorly fitted 3D planes and in high errors in the decoded depth map. However, the P^{Final} partition corrects under-segmentation errors with the added depth contours. That leads to a more efficient representation in the decoded depth map (see for instance the detail in the hand).



a) P^{Color} b) P^{Final}

Figure 5.4: Partition decision coding example. 3D planes coding example using P^{Color} and P^{Final}.

5.1.2 Decoder Scheme

The *3d Single-View* bitstream in Figure 5.3 is decoded with the scheme shown in Figure 5.5. Using the number of regions for the color partition, the decoder is able to build P^{Color} as done in the encoder without any added cost. Then, P^{Final} is built by decoding the additional boundaries and adding them to the P^{Color} partition. Combining the color partition and the depth edges, the decoder obtains P^{Final} reproducing the structure of the depth map without explicitly encoding the position of all the depth edges. Once P^{Final} is obtained, the

decoder projects the 3D planes back to the 2D depth map. Besides the color image, the camera model of the viewpoint is required to project the 3D planes. Both color images and camera model are used in the DIBR process. Thus, no extra information needs to be sent to the decoder.

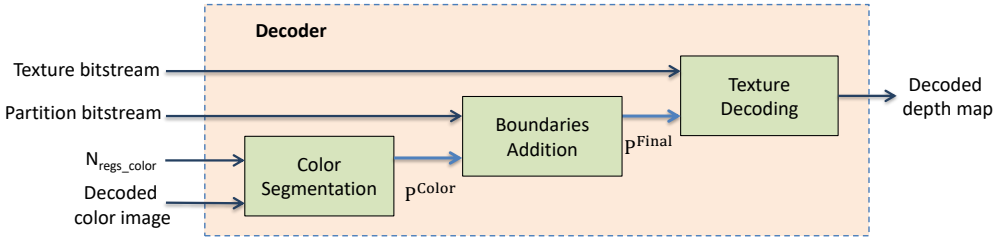


Figure 5.5: Decoder scheme. By adding the transmitted depth boundaries to the color partition the decoder obtains the final coding partition.

5.2 3D Single-View Depth Map Coding Results

The *3d Single-View* method is evaluated using 10 frames of the 3D multi-view sequence sets *undo dancer*, *ballet*, *kendo*, *breakdancers* and *balloons*. As the *3d Single-View* method does not have temporal prediction, only intra modes for the different methods are taken. Sequences used, information of generating the virtual view and computing the error measures are shown in Appendices A, B and C.

The different design parameters are discussed in section 5.2.1. Then, the complete coding scheme is compared against AVC/H.264, HEVC, 3D-HEVC and MV-HEVC and the state of the art method [OA14] in Section 5.2.2.

5.2.1 Configuration

Figure 5.6 shows the averaged cost of sending the texture and the contour information for the sequences *undo dancer*, *balloons*, *kendo*, *breakdancers* and *ballet*. The starting point corresponds to use only the P^{Color}, thus the rate for the partition is 0 bits. The number of regions in P^{Color} is determined according to the

budget rate. Adding an increasing number of contours increments the rate for both the contour and the texture, since new regions are created. The contour cost grows at a higher pace. Notice that, in the last rate-distortion point, the rate employed for texture has increased by a half of the starting rate while the contour cost is more than 5 times larger.

Since the cost of adding new boundaries rapidly surpasses the initial texture cost, it is compulsory to add only the boundaries that improve greatly the distortion figure. This behavior motivated this *3d Single-View* method, which only adds new boundaries in regions where the distortion is reduced heavily. With this methodology, the increased rate is employed solely in regions where the P^{Color} has problems representing the depth map.

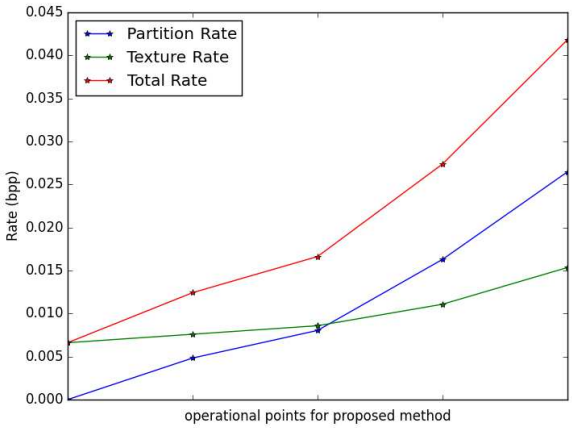
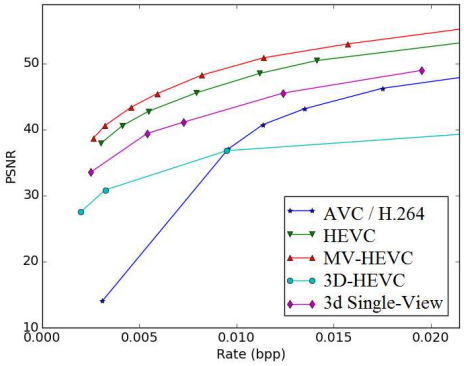


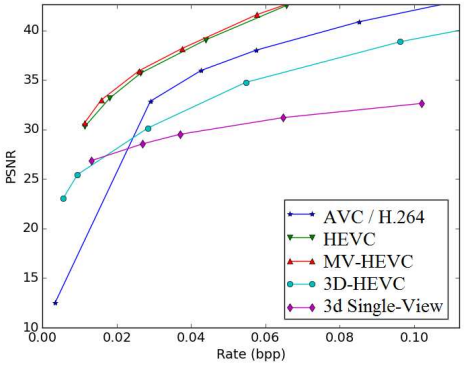
Figure 5.6: Comparative between the rate employed for coding the texture and the contour for different rate-distortion points obtained.

5.2.2 Coding performance

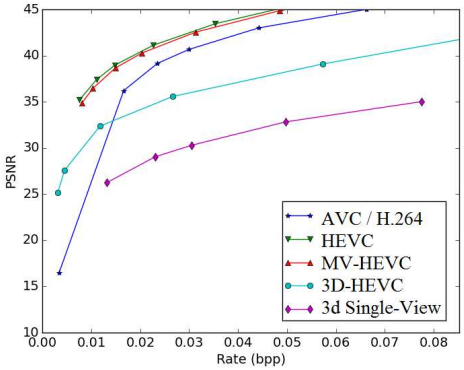
To objectively evaluate the *3d Single-View* method, error measures are taken both in the depth map and in the synthesized virtual view. The valuable measure is in the virtual view but measuring directly on the depth map gives an overview of how good can the original depth map be represented with planes. The PSNR measure is taken to evaluate the error in the depth map directly and the results for the different sequences are shown in Figure 5.7. In that



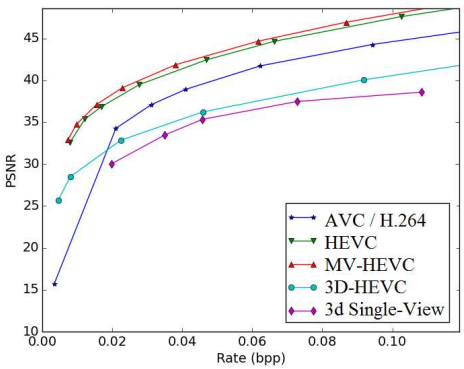
a) *undo dancer*



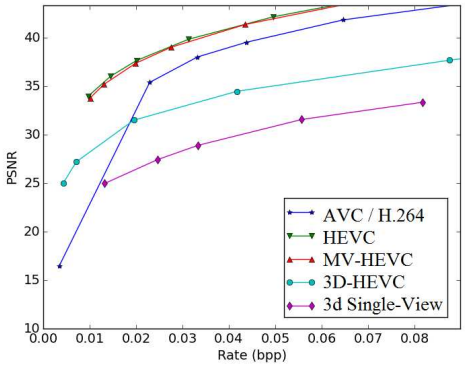
b) *ballet*



c) *kendo*



d) *breakdancers*



e) *balloons*

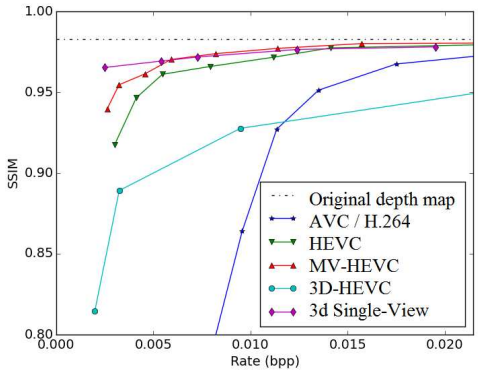
Figure 5.7: 3D Single-View: rate-distortion results over depth map.

comparison, the *3D-HEVC* performs worse than the other methods of the literature. Since color and depth map for two views are encoded altogether in *3D-HEVC*, the view synthesis optimization maximize the quality in the virtual view and not directly in the depth map. The *3d Single-View* method also is in that category and the results in depth map are below the other methods.

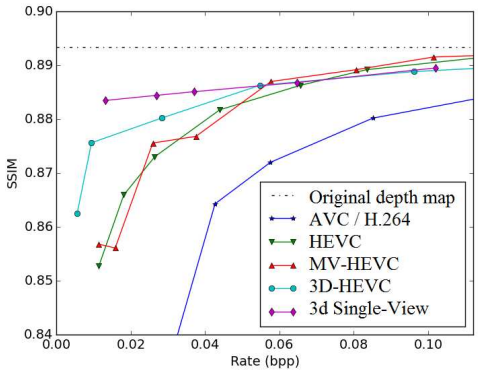
To compare the results in the virtual view, for each frame of the sequence, the virtual view is synthesized using the original depth maps and the decoded depth maps (using the *3d Single-View* method and the intra mode of AVC/H.264, *HEVC*, *3D-HEVC* and *MV-HEVC*). The color images for the synthesized process are encoded using the same quality for all the experiments. Notice that *3D-HEVC* encodes color and depth altogether but only the depth is used in the comparison.

Figure 5.8 and Figure 5.9 show the rate-distortion results for the sequences evaluated. The vertical axis shows the average SSIM of the synthesized virtual view, while the horizontal axis corresponds to the bitrate needed to encode the depth maps. The *3d Single-View* method is able to obtain better rate-distortion efficiency than the *HEVC* for low bitrates in the sequences *undo dancer*, *ballet*, *breakdancers*. For *kendo*, and *balloons* the result obtained is not as good as their depth maps are noisy and the planarity assumption does not hold. In these sequences, the depth boundaries are not well defined and often are not in correspondence with color edges. Fitting planes in those areas results in a poorly 3D estimated representations.

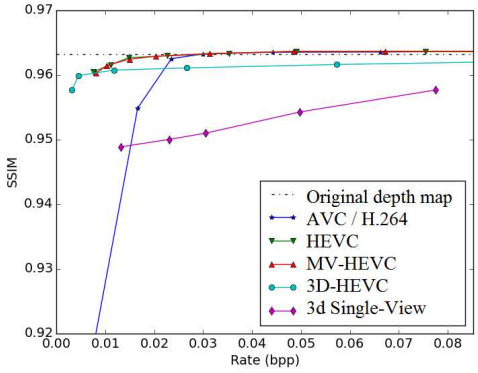
Furthermore, it can be seen that the *3d Single-View* method achieves better performance when measuring SSIM in the original view rather than in the virtual view. This means that, by using the color segmentation as a base partition for the 3D representation, the estimated planes are able to solve original inconsistencies in the depth map, obtaining a better performance than *HEVC* which is unaware of color transitions.



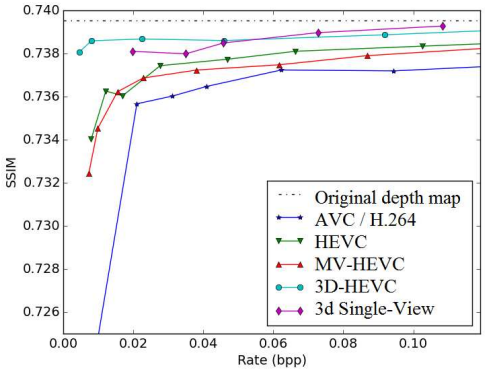
a) *undo dancer*



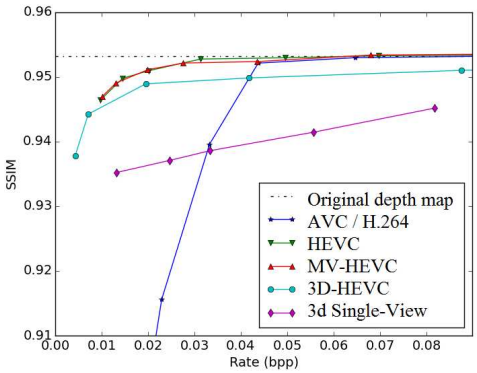
b) *ballet*



c) *kendo*

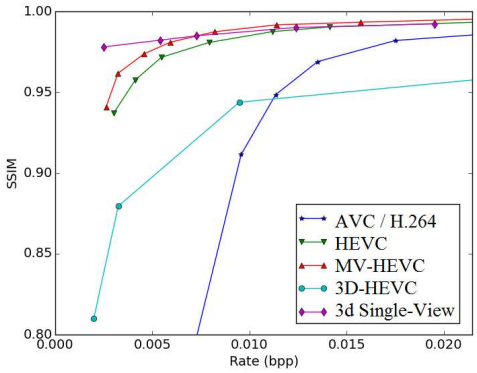


d) *breakdancers*

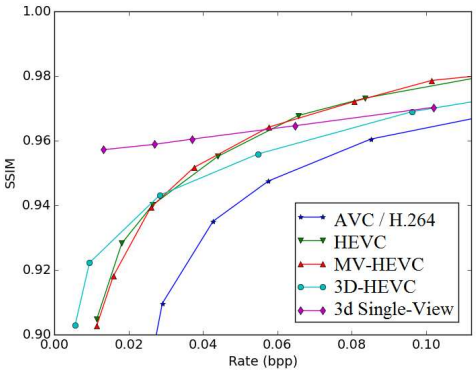


e) *balloons*

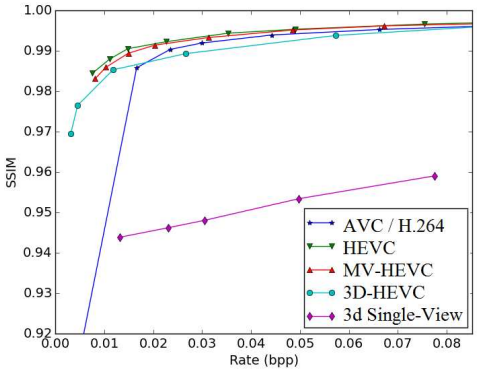
Figure 5.8: 3D Single-View: rate-distortion results over original view.



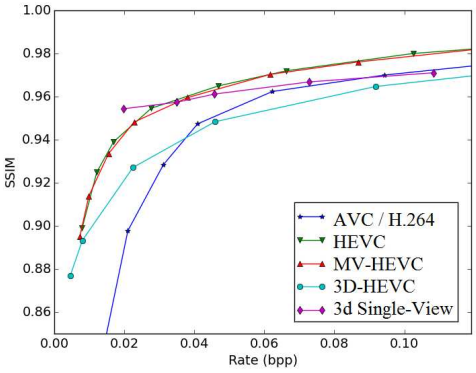
a) *undo dancer*



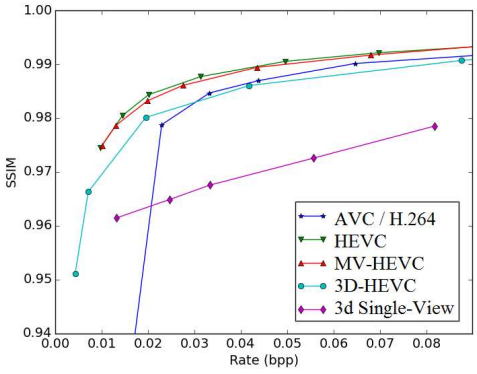
b) *ballet*



c) *kendo*



d) *breakdancers*

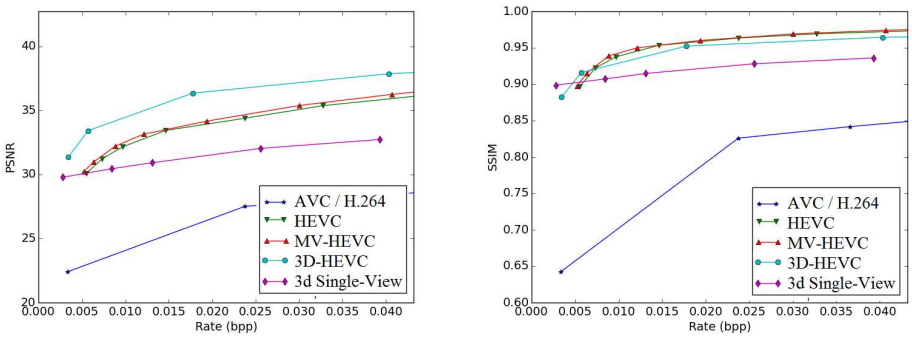


e) *balloons*

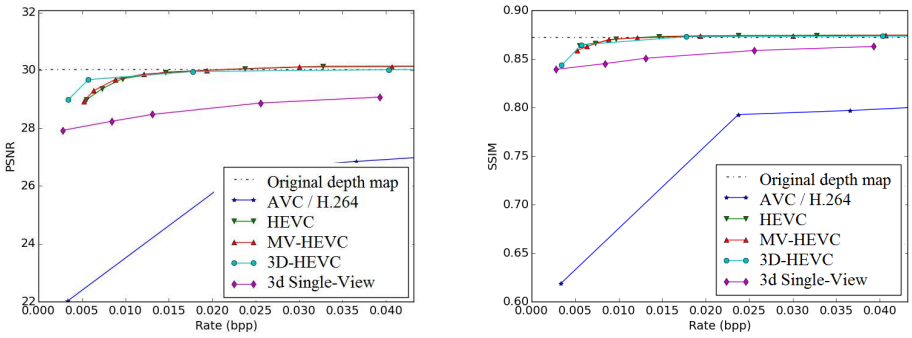
Figure 5.9: 3D Single-View: rate-distortion results over rendered view.

5.2.2.1 Results in Middlebury dataset

In order to compare against the state of the art method [OA14] described in Section 2.4, results on the Middlebury Dataset were generated. The Middlebury Stereo Dataset [SS03] consists of many stereo images with ground-truth disparities between several view-points. The datasets chosen from the website are the 2003 [SS03], 2005 [SP07] and 2006 [HS07] which were the ones used in [OA14]. Each sequence contains color information from several viewpoints and disparity for two of them. These ground-truth disparities have some unknown values which have been filled using the same in-painting method than [OA14] to have a fair comparison between the methods. The images are cropped to a multiple of 8 in order to be able to be encoded with the AVC/H.264 and *HEVC* encoders.



a) Rendered view

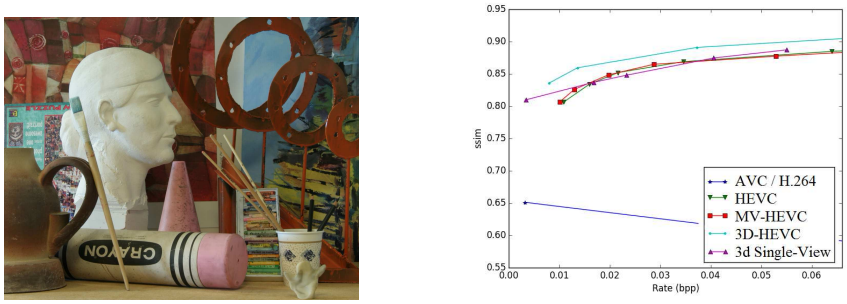


b) Original view

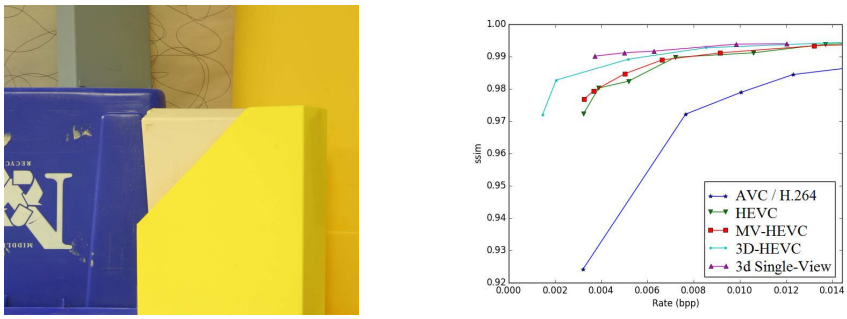
Figure 5.10: 3D Single-View: results for the middlebury dataset.

The results obtained in the full dataset at maximum resolution are shown in Figure 5.10. In the first row, the rendered image obtained with the depth maps

coding using different methods is compared with the rendered image using the original depth map. The *3d Single-View* method obtains PSNR and SSIM values comparable to HEVC for low bitrates. In the original view comparison, the values for the different methods saturate at 30 dB which is the maximum quality achievable with the given in-painted depth maps. Is worth noting that for the two comparisons the *3d Single-View* method achieves better results when evaluated with SSIM rather than with PSNR. Also, the 3D-HEVC method is clearly the best in PSNR over rendered views, since the 3D-HEVC encoder finds the best rate-distortion points for each block in terms of PSNR. Despite that, when comparing with SSIM, the *3d Single-View* method obtains similar results than other *HEVC* configurations. A direct comparison with [OA14] is not provided in Figure 5.10 since they provide results just on 6 single selected images, while in Figure 5.10 results are averaged over the full dataset. However, the *3d Single-View* method obtains results comparable to their proposed MVD method.



a) Results for Art sequence



b) Results for sequence Plastic

Figure 5.11: Results obtained for two sequences of the middlebury dataset with different coding performance.

Similar to [OA14], the performance of the *3d Single-View* method varies depending on the characteristics of the depth map. To illustrate that behavior, Figure 5.11 shows two original color images from the sequences Plastic and Art and the coding results for that sequence. In the top row, the sequence Art has a very textured color image, which leads to problems of under-segmentation in some regions. Adding depth boundaries progressively, the main under-segmentation problems in the color image are solved with depth edges, increasing the SSIM figure. On the other hand, the Plastic sequence is a sequence with few color texture, thus it is easier to segment solely with the color image. Using only color edges the performance of the *3d Single-View* method is better than the HEVC standards.

The proposed *3d Single-View* scheme obtains coding performances similar to HEVC, showing a notable improvement compared to the ones presented in Chapter 4. A more complete evaluation of contour and texture techniques is performed in Appendix D. Moreover, the 3D coding technique opens up the encoding of multiple views sharing texture coefficients, as will be explored in Chapter 6.

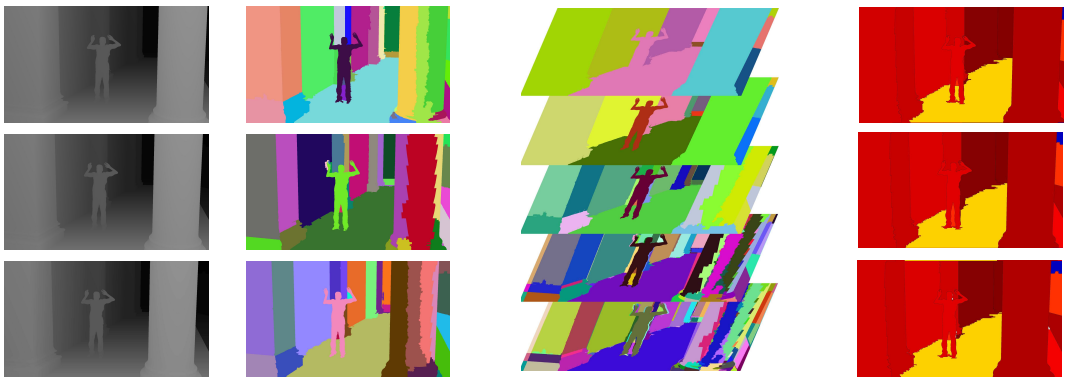
Chapter 6

3D Multi-View

In this Chapter, a complete 3D multi-view depth map coding technique is proposed. It uses the spatial redundancy among views to represent the depth maps of multiple viewpoints. The proposed scheme jointly extracts a unique 3D planar representation and a consistent segmentation of the scene from multiple depth maps. The region based representation proposed in Chapter 5 is extended to work with multiple views. A unique 3D region model is used for represent regions in different views.

In Chapter 5, the final coding partition was determined using a distortion based approach (adding depth boundaries progressively). In this Chapter, the number of encoding regions for each depth map is determined by a rate-distortion optimization process. The optimization uses the *3d-bpt* hierarchy from Section 3.3 to extract a 3D representation for the multiple views with a variable number of regions. The process of building the 3D representation is depicted in Figure 6.1.

This Chapter is structured as follows. The lagrangian rate-distortion optimization of Section 4.1.1 is stated in a global problem in Section 6.1. The optimization method is applied to find a 3D Multi-View representation using color and depth partitions in Section 6.2. This representation is used to encode depth maps of multiple views in Section 6.3. Experimental results are provided in Section 6.4.



a) b) c) d)

Figure 6.1: a) Depth images of multiple views b) For each view a combined color and depth partition P_i is obtained c) P_i partitions are combined in a reference view where a hierarchy of regions is built d) A rate-distortion optimization finds the optimal partition in the hierarchy which defines a partition in each of the input views.

6.1 Rate-Distortion Hierarchy Optimization

Inspired by [CGW03, GVB11, VAM15], the rate-distortion optimization problem presented in Section 4.1.1 is stated as a Quadratic Semi-Assignment Problem (QSAP), restricting the solution to nodes of the hierarchy. Partitions are defined in terms of boundaries between regions and hierarchical constraints are added to solve the Quadratic Semi-Assignment Problem (QSAP) problem with a linear programming relaxation approach. The notation used is consistent with [VAM15], where an optimization on multiple hierarchies is used for semantic segmentation of sequences.

The QSAP approach has two main advantages over other common approaches such as the dynamic programming solution from Section 4.1.1. Firstly, QSAP defines the relation between regions (whether they are merged or not) in terms of their common contour. This relation allows to represent the contour cost of the union of two regions easily. Secondly, QSAP can be generalized to work with multiple hierarchies, which would allow applying the procedure defined in this work to relate frames of multiple time instants in order to remove temporal redundancy.

6.1.1 Hierarchy Optimization

The objective of the hierarchy optimization is to find the optimal boundary configuration that defines a partition using nodes from the hierarchy H using the rate-distortion constrains. This optimization can be stated as a constrained minimization problem:

$$\begin{aligned} &\min_{B} \; tr(QB) = \min_{B} \sum_{m,n} q_{m,n} b_{m,n} \\ &s.t. \quad b_{m,n} \in \{0,1\} \; \forall m,n \; b_{m,m} = 0 \end{aligned} \tag{6.1}$$

where matrix Q is an affinity matrix that measures the quality of all the possible partitions in the hierarchy H. This partitions are encoded in a binary matrix B, where elements $b_{m,n} = 1$ if the boundary between leaves m and n is active (regions m and n have not been merged) and $b_{m,n} = 0$ otherwise.

For the rate-distortion optimization, the elements in matrix Q can be defined to mimic the rate-distortion lagrangian decision as:

$$q_{child_1,child_2} = J_{parent} - (J_{child_1} + J_{child_2}) \tag{6.2}$$

This allows to introduce the local node analysis of equation 4.7 into the global framework optimization of equation 6.1. Nodes with minimum Lagrangian J are favored, thus leading to the optimal partition for a given λ (see Figure 4.1).

Note that, by correctly zeroing some elements of matrix B, the whole set of partitions in H can be unequivocally described. This allows the optimization to fully exploit the richness of the hierarchical representation. In practice, not all the variables represented in this matrix are useful, as boundaries between non adjacent leave regions are not considered in the process. Thus, the maximum number of hierarchical constrains is proportional to the number of regions in the leave partition.

The hierarchy H contributes in two aspects to the optimization process. Firstly, it defines the mergings between regions of its leaves partition LP to form clusters. Secondly, it also includes the order in which these regions should be merged

to represent each node of the tree. As in [VAM15], these restrictions are encoded using two coupled constraints per node. The first constraint imposes that all the variables representing boundaries between two siblings should have the same value. The second constraint imposes that for a given node, a variable representing a boundary between two siblings can only impose a merging if all the leaves associated with the node are merged.

First, for a given parent node and in order to merge its two siblings, all the leaves that form the boundaries between these two siblings should be merged. This is imposed by:

$$\sum_{n \neq l}^{m,n} b_{m,n} = (N_c - 1) b_{m,l} \tag{6.3}$$

where N_c is the total number of common region boundaries from the leave partition that represents the union of both siblings, m is a region from the first sibling and n, l are regions from the second sibling.

Second, for a given parent node and in order to merge its two siblings, the leaves that form their respective subtrees must also be merged:

$$\sum^{n,l} b_{n,l} \leq N_m b_{m,o} \tag{6.4}$$

where N_m is the total number of inner region boundaries from the leaves partition of both siblings, m and n are regions from the first sibling and n, l are regions from the second sibling.

Note that Equation 6.3 guarantees that all boundaries between two siblings are either active or non active at the same time. Therefore, the second constraint 6.4, coupled with the first one, ensures that the optimization process propagates the second condition to all the node boundaries.

Adding these two constraints to the optimization process of Equation 6.1 results

in:

$$\min_{B} \sum_{m,n} q_{m,n} b_{m,n} \tag{6.5}$$

$$s.t. \quad b_{m,n} \in \{0,1\} \quad b_{m,n} = b_{n,m} \quad \forall n, m$$

$$\sum_{n \neq l}^{m,n} b_{m,n} = (N_c - 1) b_{m,l}, \quad \sum^{n,l} b_{n,l} \leq N_m b_{m,o} \quad \forall p \in \{H\}$$

where p represents any parent node in the collection of hierarchies. The result of this optimization is a binary matrix B^* that describes the optimal partition $\{P^*(\lambda)\}$. Varying λ different optimal rate-distortion partitions are found. Once the final partition is found, the vector b that encodes the active boundaries between leaves is stored. This information allows to recover the final coding partition from the leaves partition.

6.2 3D Multi-View Scene Representation using Color and Depth Partitions

The optimization method presented in the previous Section is applied to a 3D Multi-View representation using color and depth partitions. Prior to the method presented in this Section, a scene representation method was developed using only depth map partitions (see Appendix E). In this initial method, multiple depth partitions were combined in a reference view. Then, an optimization found a consistent partition across the views. The initial scene representation method showed the computational complexity and limitations of merging multiple partitions using a unique hierarchy. This motivated the method presented here to perform a single-view optimization before combining the information of multiple views.

The complete multi-view scene representation method is depicted in Figure 6.2. It starts by finding a partition P_i for each individual view i which merges color and depth partitions optimally. The single-view processing is done independently for each view as shown in Figure 6.3 and explained in Section 6.2.1. This process provides an optimal color-depth partition for each of the views,

thus reducing the number of regions of each view before combining the information of the multiple views.

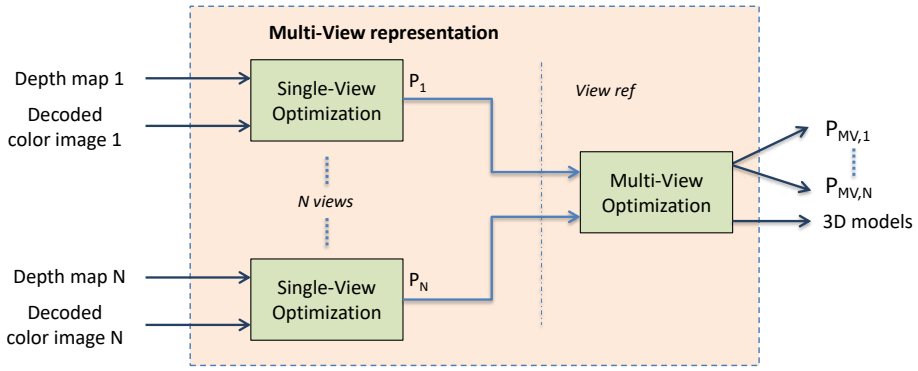


Figure 6.2: Multi-View representation. For each view i, a joint color and depth optimization finds a P_i partition. The N P_i partitions are projected to the reference view where a multi-view optimization finds a consistent partition $P_{MV,i}$ for the N views.

The information of the multiple views is accumulated into a common reference view $view_{ref}$ by projecting the P_i into $view_{ref}$. Then, a multi-view optimization process (depicted in Figure 6.6) obtains P_{MV} which defines the final coding partitions for each view. Complete details on this multi-view optimization are given at Section 6.2.2.

The optimization process of equation 6.5 is referred as:

$$P^*(\lambda) = Opt_\lambda(H) \tag{6.6}$$

where, for a given λ, an optimal partition P^* is extracted from H. For simplicity, the notation of optimal partitions is reduced from $P^*(\lambda)$ to P.

Hierarchies in the single-view and the multi-view stages are built as in Chapter 5: Initial color and depth map partitions (LP_i^{Color} and LP_i^{depth} respectively) are generated with the methods presented in Chapter 3. Hierarchical representations H are generated with *3d_bpt* method presented in Section 3.3. A 3D region model is used to represent each region in H as presented in Subsection 5.1.1.1.

6.2.1 Single-View Optimization

In the Single-View Optimization LP_i^{Color} and LP_i^{depth} are combined into an optimal partition P_i for each view i as shown in Figure 6.3.

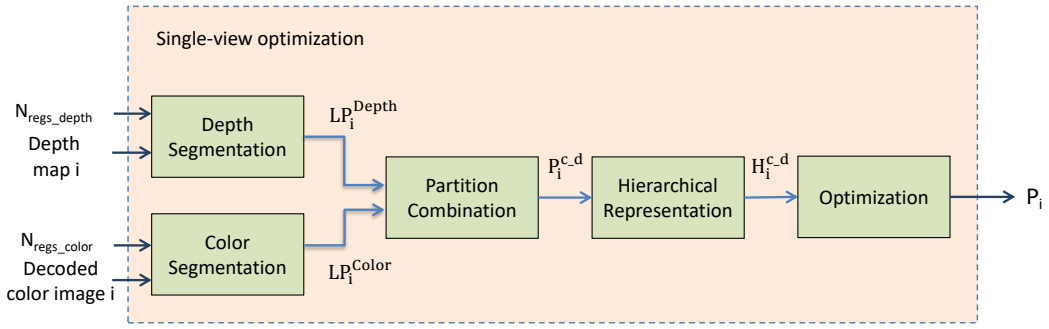


Figure 6.3: Optimization process for each view. A combined color and depth optimization is performed finding an optimal combined partition.

LP_i^{Color} and LP_i^{depth} are fused into a partition $P_i^{c\_d} = LP_i^{color} \bigcap LP_i^{depth}$ that preserves all the boundaries from both partitions as presented in Subsection 4.2.1.1. The $P_i^{c\_d}$ is used to build a hierarchy $H_i^{c\_d}$. The optimization procedure introduced in Section 6.1 over that hierarchy obtains an optimal rate-distortion partition P_i:

$$P_i = Opt_\lambda(H^{c\_d}) \tag{6.7}$$

The equation 6.2 can be expressed as follows:

$$\begin{gathered} q_{ch_{i,1},ch_{i,2}} = J_{p_i} - (J_{ch_{i,1}} + J_{ch_{i,2}}) = \\ (D_{p_i} + \lambda R_{p_i}) - (D_{ch_{i,1}} + \lambda R_{ch_{i,1}}) - (D_{ch_{i,2}} + \lambda R_{ch_{i,2}}) = \\ \triangle Dist_{ch_{i,1},ch_{i,2}} + \lambda \triangle Rate_{ch_{i,1},ch_{i,2}} \end{gathered} \tag{6.8}$$

where p_i is the parent region resulting of the union of children $ch_{i,1}$ and $ch_{i,2}$: $p_i = ch_{i,1} \bigcup ch_{i,2}$. $\triangle Dist_{ch_{i,1},ch_{i,2}}$ is the increment in distortion caused by the union and $\triangle Rate_{ch_{i,1},ch_{i,2}}$ is the increment in rate. Typically, the $\triangle Dist_{ch_{i,1},ch_{i,2}}$ is positive as representing the depth map with less regions will increase the distortion and $\triangle Rate_{ch_{i,1},ch_{i,2}}$ is negative.

The distortion of each region k of the hierarchy is computed as the square error between the 3D planar model and the pixel values of the region, particularizing

Equation 4.3 as:

$$D_{i,k} = \sum_{n=1}^{N_{i,k}^{Pix}} |Proj_i(\Pi_k, n) - g_i(n)|^2 \tag{6.9}$$

where Π_k^{3D} is the plane model for region k, $N_{i,k}^{Pix}$ is the number of pixels of region k and $Proj_i(\Pi_k, n)$ is the value of the projected 3D planar model to the pixel n of region k and $g_i(n)$ is the depth value of pixel n. An example of the projection step is shown in Figure 6.4.

The $\triangle Dist_{ch_{i,1},ch_{i,2}}$ is computed as:

$$\triangle Dist_{ch_{i,1},ch_{i,2}} = D_{p_i} - D_{ch_{i,1}} - D_{ch_{i,2}} \tag{6.10}$$

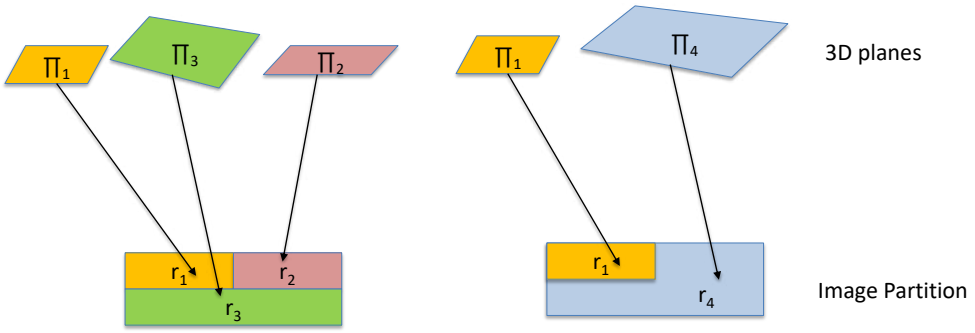


Figure 6.4: The Single-View distortion is computed projecting each 3D planar model to all the pixels of the corresponding region. The r_4 distortion generated with Π_4 is compared with a representation using the child regions r_2 and r_3, represented with models Π_2 and Π_3.

To compute the $\triangle Rate_{ch_{i,1},ch_{i,2}}$ two terms are present: texture and contour. The texture $R^T_{ch_{i,1},ch_{i,2}}$ is a constant -R^T, since with the union only one model (instead of one for each child) is needed.

Contours of a region may come from two partitions; some from LP_i^{Color} and some from LP_i^{depth}. The contour cost $R^C_{ch_{i,1},ch_{i,2}}$ is computed by counting the number of contour elements (two neighboring pixels with different labels define one contour element between them) of the regions $ch_{i,1}$, $ch_{i,2}$ and multiplying for the average cost of encoding each contour element:

$$R^C_{ch_{i,1},ch_{i,2}} = -c_A\, N_{ch_{i,1},ch_{i,2}} - c'_A\, N'_{ch_{i,1},ch_{i,2}} \tag{6.11}$$

where $N_{ch_{i,1},ch_{i,2}}$ and $N'_{ch_{i,1},ch_{i,2}}$ are the numbers of common contours of the two children from the depth and color partitions respectively, while c_A and c'_A are the average costs to encode a contour element for either the depth and the color partitions. Note that contours from the color partition do not introduce any cost, therefore $c'_A = 0$. The c_A has been obtained experimentally using the multi-view sequences resulting in $c_A = 1.2$ bits per contour position.

Equation 4.4 is computed as:

$$\triangle Rate_{ch_{i,1},ch_{i,2}} = -R^C_{ch_{i,1},ch_{i,2}} - R^T_{ch_{i,1},ch_{i,2}} = -c_A\, N_{ch_{i,1},ch_{i,2}} - R^T \qquad (6.12)$$

Figure 6.5 shows an example of merging two regions. Encoding r_4 instead of r_2 and r_3 saves the cost of encoding the boundary between the regions and the texture coefficients of one region.

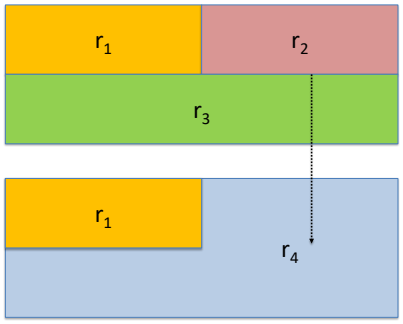


Figure 6.5: The Single-View rate for each region is the rate saving promoted by the merging. Regions r_2 and r_3 are merged into r_4, leading to represent the depth map without the contour between r_2 and r_3 and saving one 3D plane.

The color-depth optimization in each view addresses the addition of the depth boundaries in the color partition in an optimal fashion while reducing heavily the total number of regions. Doing this procedure in each view also reduces the number of occluded regions in the Multi-View Optimization.

6.2.2 Multi-View Optimization

The multi-view optimization process is shown in Figure 6.6. The partition P_i of each of the N views is projected to $view_{ref}$, obtaining an initial partition P_i^{ini} for each view:

$$P_i^{ini} = Proj_{ref}(P_i) \qquad (6.13)$$

The projection step uses the camera parameters to translate the information of each view to the reference view $view_{ref}$. Pixels in each partition P_i are processed in scan order and the corresponding pixel labels are projected to $view_{ref}$. Each individual projected partition P_i^{ini} is defined by assigning to each pixel position the label of the nearest projected $view_i$ pixel. With this process the regions that are near to the camera position prevail over the ones that are further away.

In this projection step, regions of $view_i$ that are occluded in the reference view are detected. A region is considered occluded if the number of pixels of this region in $view_{ref}$ is less than half the number of pixels in $view_i$. Occluded regions have no valid correspondence in $view_{ref}$. To solve this, these regions will be encoded independently in that view and removed from the projected partition in $view_{ref}$.

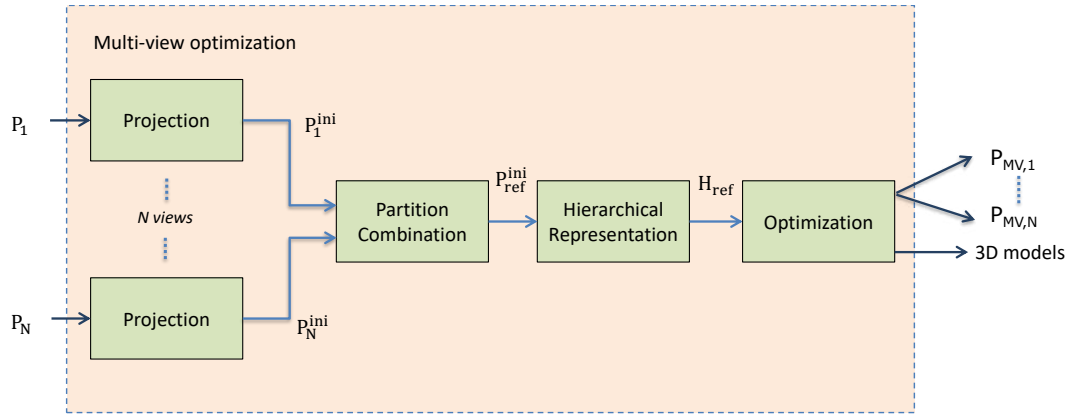


Figure 6.6: Multi-View Optimization process for each view. Partitions from multiple views are combined in a unique hierarchy. Rate-distortion measures are computed using all the input views and stored in the hierarchy to find the joint optimal partition for the multiple views.

The individual projected partitions P_i^{ini} in $view_{ref}$ are accumulated in a unique partition as stated in Equation 6.14. Unlike the single-view optimization, here in the projected $view_{ref}$ not all the pixels have a projected label. In order to obtain a partition with all the pixels assigned to a label, a hole filling algorithm creates new labels in-between regions with different label. For each combination

of labels in each $view_i$, a new label is created in P_{ref}^{ini}.

$$P_{ref}^{ini} = \bigcap_i P_i^{ini} \tag{6.14}$$

A hierarchical depth representation H_{ref} is built using the accumulated partition P_{ref}^{ini}. Rate and distortion values for each region in H_{ref} are found examining each partition generated in the N views. The increment of distortion for each union is computed as:

$$\triangle Dist_{ch_1,ch_2} = \sum_{i=1}^{N} \triangle Dist_{ch_{i,1},ch_{i,2}} \tag{6.15}$$

where $\triangle Dist_{ch_{i,1},ch_{i,2}}$ represents the increment of distortion of each view i as in Equation 6.10. A graphical example of equation 6.15 is shown in Figure 6.7. Notice that, in the example, $r_{ocl\,1}$ is occluded in $view_{ref}$. Therefore, $r_{ocl\,1}$ does not participate of the optimization process and is encoded with a independent plane.

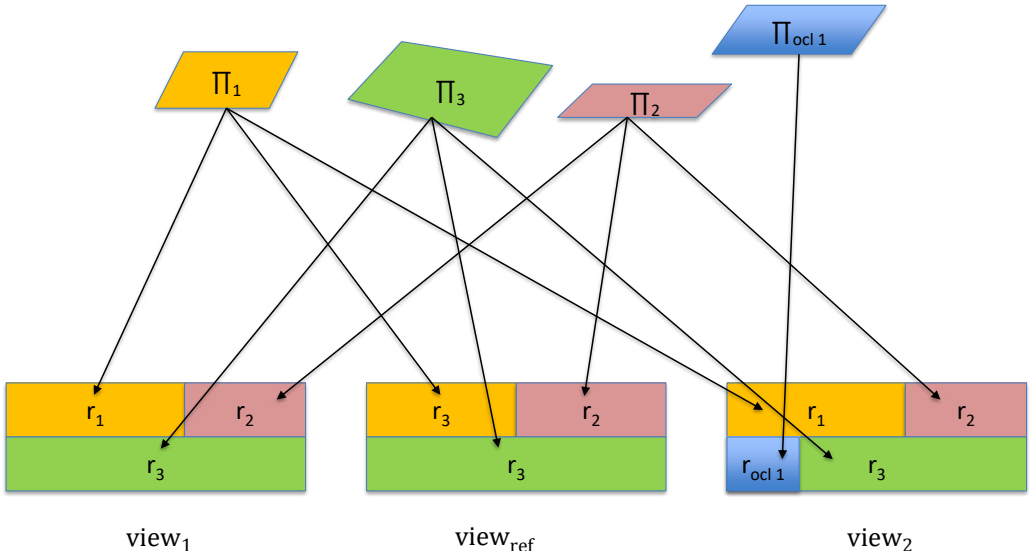


Figure 6.7: The Multi-View distortion is computed projecting each 3D planar model to all the pixels of the corresponding region of each image.

Similarly, the rate is computed using the information of all views, adding the boundary cost (from depth boundaries only, as $c'_A = 0$ for color contours):

$$\triangle Rate_{ch_1,ch_2} = \sum_{i=1}^{N} \triangle Rate_{ch_{i,1},ch_{i,2}} \tag{6.16}$$

As the same region model encode regions of multiple views, the texture rate is the same than in the single view optimization:

$$\triangle Rate_{ch_1,ch_2} = -c_A \sum_{i=1}^{N} N_{ch_{i,1},ch_{i,2}} - R^T \tag{6.17}$$

An example of computing the rate for the Multi-View optimization is shown in Figure 6.8.

The P_{MV} partition is obtained using the hierarchical optimization of section 6.1:

$$P_{MV} = Opt_\lambda(H_{ref}) \tag{6.18}$$

Since the hierarchy H_{ref} is build in the $view_{ref}$, the merging orders are determined in $view_{ref}$. By incorporating the information of the N views to the optimization process, the resulting partition P_{MV} defines an optimal partition in all the views.

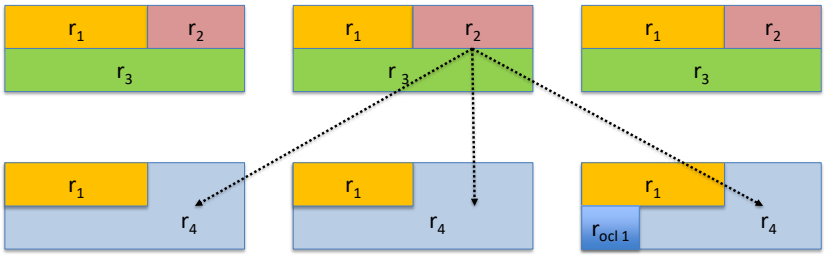
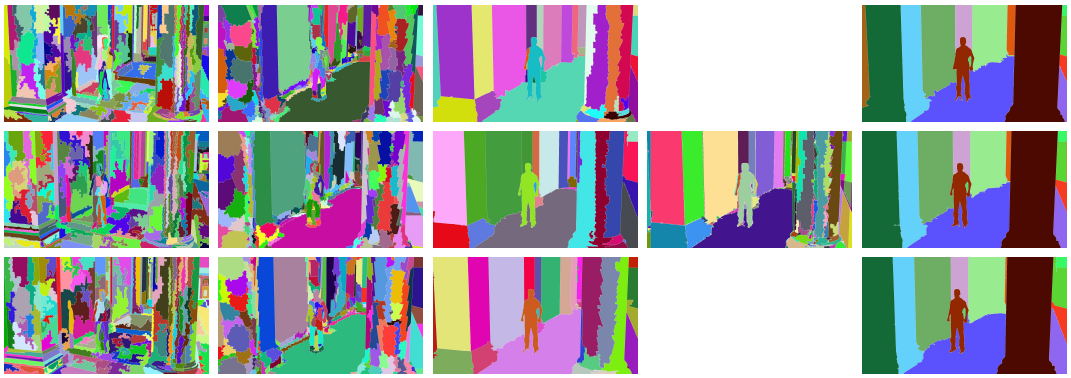


Figure 6.8: The Multi-View rate for each region is the rate saving obtained with the merging. A merging in $view_{ref}$ may force a merging in the other views.

The optimization parameter λ that defines the rate-distortion trade-off is the quality parameter. By varying λ, different rate-distortion points are found, being able to obtain 3D scene representations with different level of detail. The multi-view optimization obtains a set of consistent partitions referred as $P_{MV,i}$. An overview of the partitions obtained at every stage is shown in Figure 6.9.

6.3 3D Multi-View Depth Map Coding

In the previous Section, a set of $P_{MV,i}$ partitions and 3D models was found. In order to convey that information to the decoder a method to pack the informa-



a) b) c) d) e)

Figure 6.9: Example of the different partitions obtained in the Multi-View 3D Representation a) Input Color partition LP_i^{Color} b) Input Depth partition LP_i^{depth} c) Result of the Single-View Optimization P_i d) P_{ref}^{ini} partition e) Result of the Multi-View Optimization process $P_{MV,i}$.

tion of the multiple views is proposed. Figure 6.10 shows the encoding process for each view i.

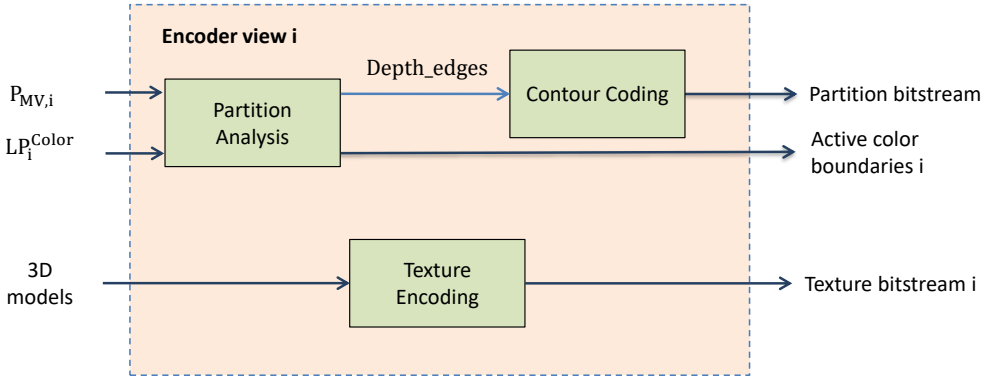


Figure 6.10: Encoder scheme. Partition information is generated independently for each view. Texture encoding encodes the 3D plane in INTRA if it is the first view where it appears or INTER if it has been encoded previously.

To generate the partition information, the first step consists in analyze which contours from LP_i^{color} and LP_i^{depth} are present in $P_{MV,i}$. Contours in $P_{MV,i}$ from LP_i^{depth} are encoded with Freeman contour coding [Fre61] independently for each view. For the color contours, information of color boundaries that are not active (because of region mergings) is sent to the decoder indicating if the

merging has been done or not. Active information is stored with 1 bit per each merging in H_{ref} as done with the side information in Section 4.1.

Once the partition information has been encoded, the texture coefficients are generated. The first view encodes the texture information of all the planes associated to that view in INTRA mode, while planes from the subsequent views will be encoded using INTER view prediction when possible. In the INTRA mode, the 3D planes are represented by using the distance of that plane to a camera plus the plane orientation. The INTRA mode uses the same representation as the 3D Single-View depth map coding presented in Section 5.1.1.1.

For subsequent views i, the INTER mode is used. Planes from the first view are projected to view i as done in the construction of the multi-view partition in $view_{ref}$. Each region in partition $view_i$ has a candidate plane from $view_1$. This plane is compared with a plane coded with INTRA in $view_i$. The plane with the lower error is kept. If the plane is coded with INTER, a SKIP mode is sent, if not the region is INTRA coded.

The bitstream (depicted in Figure 6.11) provides the decoder with the information needed to recover the partition and the texture for each region. The partition bitstream, active color boundaries and texture bitstream for each view are provided in addition of the number of regions N_{regs_color} in each initial color partition LP_i^{color}. N_{regs_color} is the same for the N views.

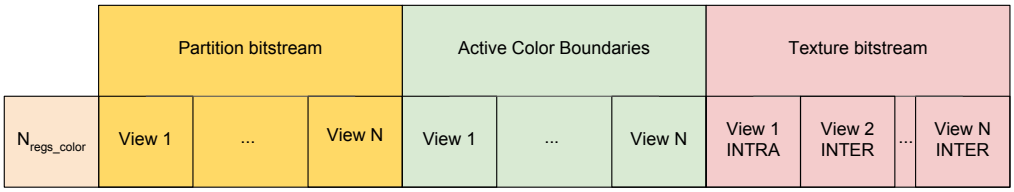


Figure 6.11: Bitstream with the information of the N views. The depth partition information signal the depth boundaries added to the color information. With the active boundaries information the final coding partition is obtained. The texture coefficients for views 2 to N make use of the previous transmitted coefficients.

The decoder process is depicted in Figure 6.12. For each view, the LP_i^{color}

is built from the decoded color image. From that partition, color edges are removed using the active color boundaries information. Then new edges are added from the partition bitstream. Depth map is recovered by decoding the texture bitstream and projecting each 3D plane to $P_{MV,i}$.

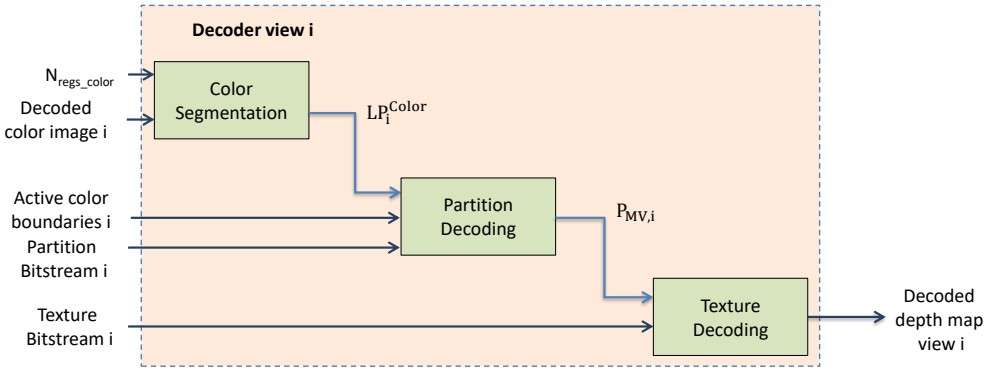


Figure 6.12: Decoder scheme. LP_i^{color} is generated from the decoded color image using N_{regs_color}. Using the information of active color boundaries and the partition bitstream the encoding partition $P_{MV,i}$ is generated. The decoded depth map for view i obtained by projecting the 3D planes to $P_{MV,i}$.

6.4 3D Multi-View Results

In this Section, a quantitative evaluation of the 3D Multi-View depth map representation and coding methods are presented. The RGB-D multi-view sequences *ballet*, *undo dancer*, *breakdancers* and *ghost town* are used for evaluation. Each sequence is composed of several color and depth images from multiple views, as well as camera parameters for each frame (see Appendix A). The different stages of the 3D Multi-View method are evaluated using 25 frames of each sequence.

First, different configurations for the 3D multi-view scene representation method are studied in Section 6.4.1. Then, the 3D multi-view depth map coding method is compared against HEVC, 3D-HEVC and MV-HEVC in Section 6.4.2.

Experiments will make use of three views as shown in Figure 6.13: $view_{left}$, $view_{right}$ and $view_{ref}$ using the same configuration of 3D-HEVC. $view_{left}$ and

$view_{right}$ are the two base views to encode and $view_{ref}$ is the location selected to perform the optimization process in the multi-view optimization block (see Section 6.2). This view corresponds to the virtual view of 3D-HEVC. Color and depth map images as well as the camera parameters are available in the three views considered.

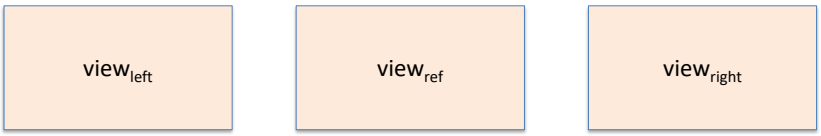


Figure 6.13: Three views are considered in the generation of results: $view_{left}$, $view_{right}$ and $view_{ref}$. $view_{ref}$ is the intermediate view where the Multi-View optimization is performed.

6.4.1 3D Multi-View Scene Representation Evaluation

In this Section, the different blocks of the 3D Multi-View Scene Representation system are analyzed. Results for the different sequences are averaged into a single rate-distortion curve.

6.4.1.1 Combining Multiple Views

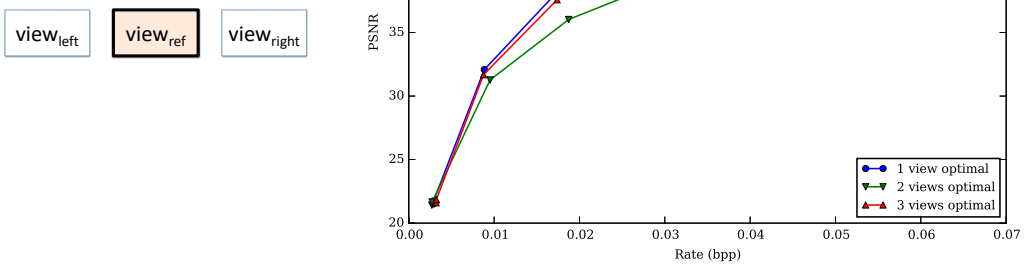


Figure 6.14: Results in the $view_{ref}$. 1 view configuration have only information of $view_{ref}$, 2 view configuration use the 2 views projected to $view_{ref}$ and 3 view configuration uses the three of them.

The first experiment of the 3D Scene Representation recuperates the configuration used in Appendix E to analyze their limitations. It combines depth partitions (LP_i^{depth}) from multiple views without the single-view optimization. Thus, the multi-view optimization stage of Section 6.2.2 is evaluated using LP_i^{depth} as P_i partitions for the different views. Different configurations are used to analyze their performance: the *1 view optimal* configuration uses only information from $view_{ref}$ in the multi-view optimization process (Equations 6.14 to 6.18 with $P_{ref}^{ini} = LP_{ref}^{depth}$). The *2 views optimal* configuration uses information of $view_{left}$ and $view_{right}$ ($P_{ref}^{ini} = LP_{left}^{depth} \bigcap LP_{right}^{depth}$) and in the *3 views optimal* all 3 views are used.

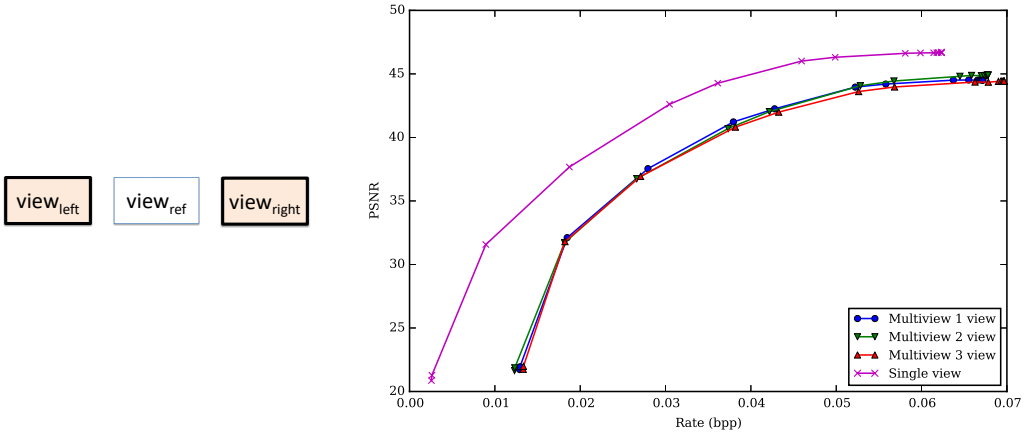


Figure 6.15: Results from Figure 6.14 are translated to $view_{left}$ and $view_{right}$. The three multi-view configuration (all with hierarchy built in $view_{ref}$) are compared with a hierarchy in each view independently.

Results are analyzed first in $view_{ref}$. Figure 6.14 shows the results of different configurations. The *1 view optimal* configuration, which only uses information from $view_{ref}$, obtains the higher rate-distortion figures. In the *3 views optimal*, results are slightly below the *1 view optimal*. As the projection from these views to $view_{ref}$ is not able to match all the depth boundaries in $view_{ref}$, the results for the *2 views optimal* are below the 1 view configuration.

The resulting partitions are then evaluated in $view_{left}$ and $view_{right}$ comparing also to a single-view optimization done in each view independently. Results are shown in Figure 6.15. Each region that is not present in $view_{ref}$ (occluded

regions) is encoded with an additional 3D planar model in that view. The different multi-view configurations achieve similar results, all of them below the results of the single-view configuration. This behavior is due to the fact that the hierarchical optimization in $view_{ref}$ restricts the possible mergings in the other views and that an extra rate is needed to encode occluded regions.

The use of a single-view optimization prior to the multi-view stage overcomes the limitations of restricting mergings in other views. With the single-view optimization, the number of regions in the different views is reduced before building the shared hierarchy between views in $view_{ref}$. Being able to merge regions in a single-view reduces the number of regions that are occluded in $view_{ref}$ while creating a hierarchy H_{ref} in $view_{ref}$ with relevant regions.

6.4.1.2 Single View Optimization: Rate-distortion

In this section the performance of the rate-distortion optimization method in the Single-View Configuration is studied. Different cuts of the hierarchy $H_i^{c\text{-}d}$ are extracted following the merging sequence are compared with optimal partitions P_i with different quality parameter λ. By using the 3D Multi-View optimization, a gain of 2-3 dB over using the H_i^{depth} merging sequence is achieved.

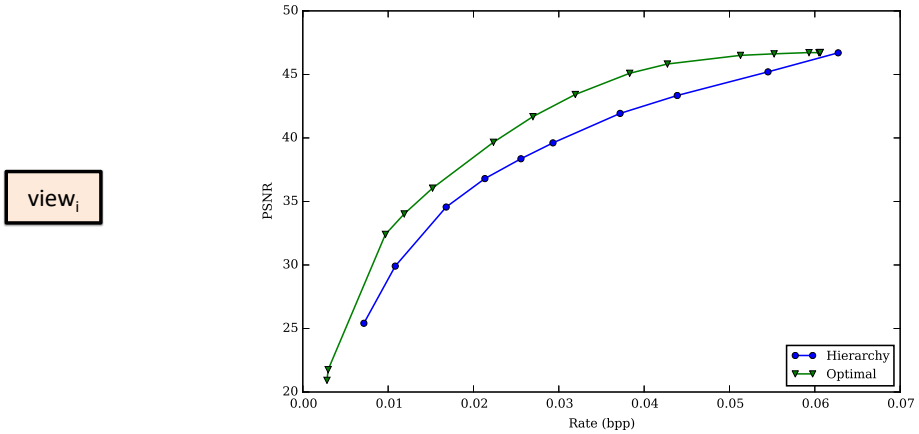


Figure 6.16: Rate-distortion results with the 3D Multi-View optimization method. Using an initial H^{depth}, results from coding the partition with 3D planes following the merging sequence and with the proposed optimization.

6.4.1.3 Single View Optimization: Color and Depth Combination

Figure 6.17 shows the R-D curves for three different options: using only a LP^{Depth}, using only a LP^{Color} and using a P^{c_d}. Using only color partition has the advantage that no contour information has to be sent. Its a good option for very low bit-rates but the lack of precision in the region boundaries limits the maximum achievable quality. Use only depth information increase the quality obtained at the cost of coding all depth contours (increased rate). The combination of color and depth provides a good trade-off because most of the depth contours can be approximated using color boundaries and only a reduced set of depth contours not present in the color partition must be sent.

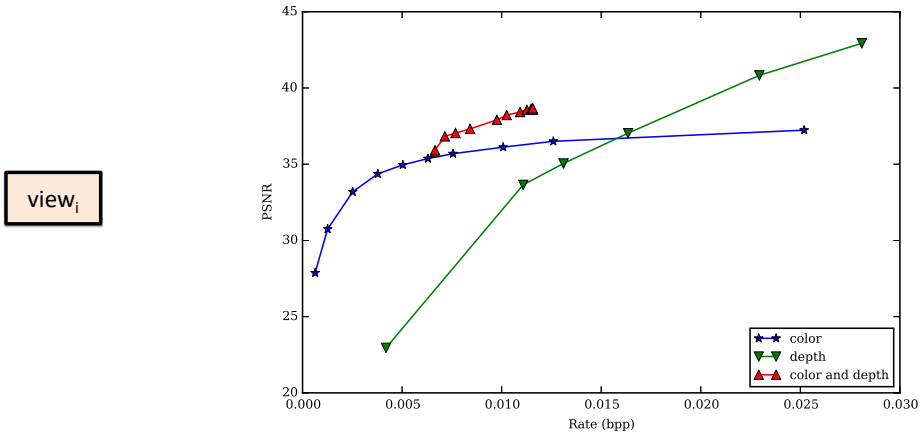


Figure 6.17: R-D curves using LP^{Color}, LP^{Depth} and combined P^{c_d} to encode depth maps. Partitions using depth include the cost of encoding texture and boundary while the colors only need texture information.

6.4.2 3D Multi-View Depth Map Coding Results

Configuration setup for the experiment is provided in Appendix B. For each sequence, three views are used, $view_{left}$ and $view_{right}$ are encoded (each view independently) and the middle one is employed as the location for the $view_{ref}$. The view rendered using the original depth maps serves as a reference for the different methods in the virtual view. As the 3D multi-view method does not have temporal prediction, only intra modes for the different methods are taken. To evaluate the performance of the 3D multi-view system, PSNR and SSIM

measures (see Appendix C) were taken. As both measures show similar tendencies, only the PSNR will be shown to evaluate the different techniques.

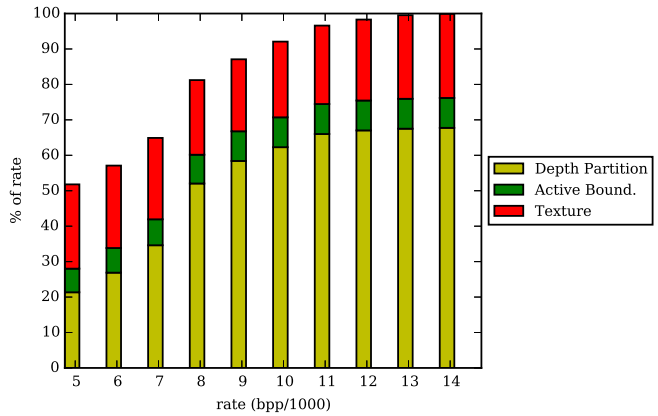
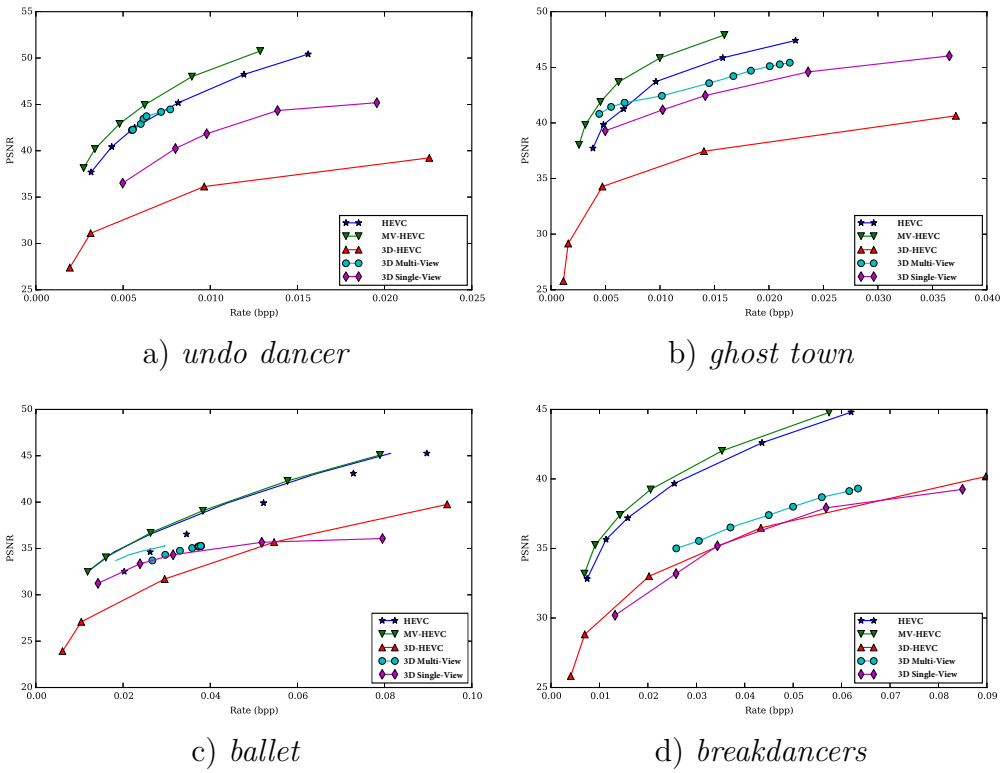


Figure 6.18: Distribution of rate in texture, partition and active boundaries information for different rate-distortion points.

To evaluate the distribution of rate among texture, boundaries and active boundaries information, the bitstream presented in Section 6.3 was analyzed to determine the different contributions. Figure 6.18 shows the rate distribution among each category. Each bar is normalized with the maximum rate in the last column. The cost of encoding the texture and the information of active boundaries remains fairly constant among all the different rate-distortion points. However, the partition cost becomes the prevailing one as the rate increases. The results of the optimization depend on the given budget rate. When the budget is low, only color boundaries are added. When more bits can be allocated, the optimization adds the costly depth boundaries.

Results measured on the depth map for the different sequences are shown in Figure 6.19. In that comparison, the 3D-HEVC performs worse than the other HEVC configurations methods of the literature. As explained in Section 5.2, this is because color and depth map for two views are encoded together in 3D-HEVC. The view synthesis optimization in 3D-HEVC maximize the quality in the virtual view and not directly in the depth map. The 3D multi-view method outperforms the single-view approach presented in Chapter 5 (3d Single-View) for all the sequences. The use of the 3D multi-view optimization allows a gain



a) *undo dancer* b) *ghost town*

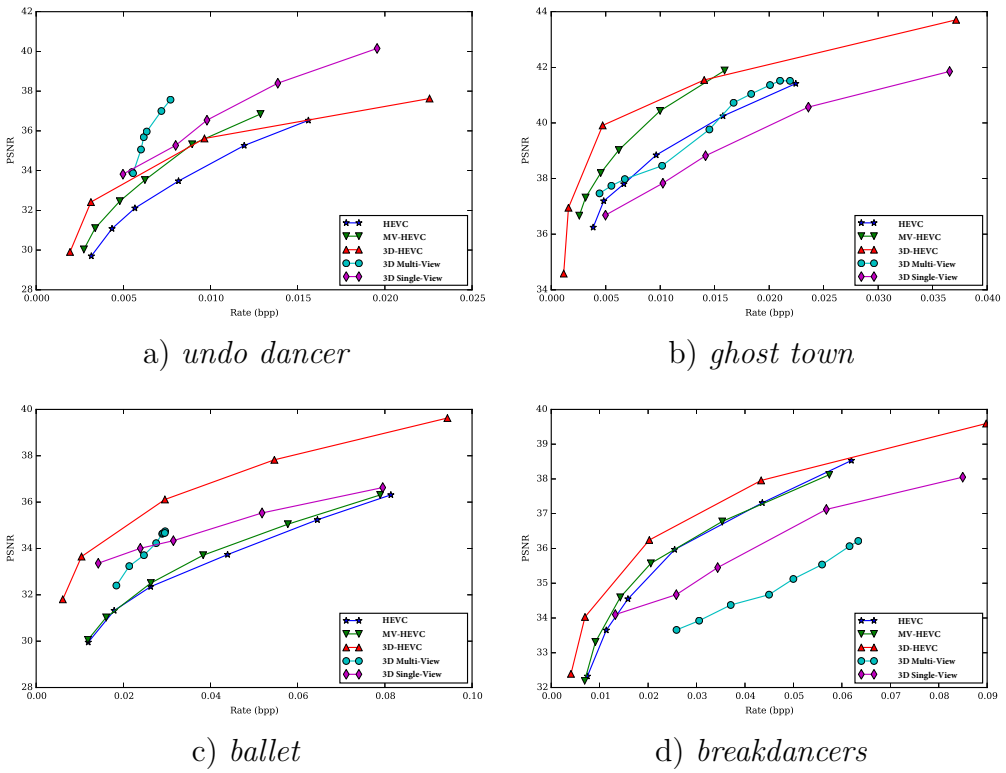
c) *ballet* d) *breakdancers*

Figure 6.19: Multi-View results: rate-distortion evaluated over depth map.

over the 2 dB for the different sequences. This improvement is noteworthy for the *undo dancer* sequence, where the planar characteristic of the scene can be fully exploited. On the other hand, the 3D multi-view method has more problems with the *ghost town* sequence. In that sequence, the color segmentation is not able to properly segment elements at different depths since all of them have the same tonality. In that case the number of contours added for the Multi-View approach is similar to the Single-View approach, thus achieving comparable results.

In sequences where the depth map is noisier, as *ballet* and *breakdancers*, the 3D planar segments are not able to represent correctly the depth maps, thus obtaining worse results than the HEVC standards.

Results in the rendered virtual view are shown in Figure 6.20. The results of



a) *undo dancer* b) *ghost town*

c) *ballet* d) *breakdancers*

Figure 6.20: Multi-View results: rate-distortion evaluated over rendered view.

the planar implementations achieve better results than HEVC and MV-HEVC in *ballet* and *undo dancer* and are closer than in the depth map comparison in the other sequences. Notice that in *undo dancer*, 3D-HEVC does not improve HEVC or MV-HEVC. As explained in the depth map domain, this is due to 3D-HEVC encoding together color and depth information. Here only depth maps from 3D-HEVC are taken, using the same color image than the other methods for generating the rendered view. The results for Figure 6.20 are summarized in Tables 1 and 2, where the Bjontegaard's metric to compute the average gain in PSNR Bjontegaard delta PSNR (BD-PSNR) and to compute the average saving in bitrate Bjontegaard delta RATE (BD-RATE) are shown, taking the HEVC as a reference.

The 3D multi-view method improves the rate-distortion results of the 3D single-view method in Chapter 5 for all the sequences in the depth map. This result

Table 6.1: BD-RATE

	MV-HEVC	3D-HEVC	3D Single-View	3D Multi-View
undo dancer	-23.69	-40.97	-36.94	-49.16
ghost town	-39.96	-64.50	42.03	2.91
ballet	-6.03	-71.89	-37.16	-41.63
breakdancers	-6.16	-35.71	48.46	150.13

Table 6.2: BD-PSNR

	MV-HEVC	3D-HEVC	3D Single-View	3D Multi-View
undo dancer	1.16	1.80	2.12	3.38
ghost town	1.43	2.16	-1.03	0.01
ballet	0.20	3.47	1.57	1.51
breakdancers	0.17	1.02	-1.03	-2.57

is not translated equally to the virtual view. For *ballet* and *breakdancers*, the results in the virtual view are below the single-view algorithm. This is because the multi-view algorithm reduces the number of 3D planes when encoding the depth map. *ballet* and *breakdancers* are noisier and present a larger base-line between views. The reduced number of segments in the multi-view scheme penalizes the performance of the method. On the other hand, in *undo dancer* the planar characteristics of their depth maps allow an improvement over the HEVC standards.

Chapter 7

Conclusions

This thesis is divided in three main parts. The first part tackles the problem of generating image partitions adapted to depth map coding, both for color images and for depth map images. The second and third parts uses the set of partitions generated to encode depth maps. In the second part, 2D segmentation-based techniques are used to encode depth maps. In the third part, 3D region models obtain 3D representations of the scene while encoding the depth maps.

Two image segmentation algorithms have been introduced for generating the color and depth partitions independently. For the color segmentation, the *bpt_spx* criterion has been proposed which adds a new term based on the distance between region centroids in order to obtain compact regions has been proposed. The segmentation results improve the previous criteria in terms of precision and F-measure over boundaries, obtaining results similar to partitions obtained with state-of-the-art SLICS method.

The *bpt_3d_rgrow* depth segmentation method uses a 3D planar region model alike to the 3D representation desired for the multi-view depth map encoding. In this model, each region is characterized by the centroid of the 3D points of the region and the normal orientation of the plane. Comparing with different state of the art segmentation methods, it is shown the benefits of using the proposed method for depth map compression.

The proposed 2D Single-View Depth Map Coding techniques validate the use of the redundancy of color images and depth maps to improve depth map coding efficiency. In the Color-Based hierarchical optimization, the benefits of using a hierarchy of regions to extract the final coding partition have been shown. In the fusion of color and depth partition method, the necessity to explicitly signal depth edges to overcome color-based segmentation limitations have been studied. All depth edges are added to the color partition to obtain the coding partition.

The 3D single-view coding proposed uses the two segmentation techniques introduced in this work. It combines the color partition and the depth map partition to obtain the final coding partition that properly segments the depth map. A difference of the fusion method, not all depth edges are signaled, a partition decision step selects the most important ones.

In the multi-view domain, in order to work with the multiple images the rate-distortion optimization problem is formulated as a QSAP. Partitions are defined in terms of boundaries between regions and hierarchical constraints are added to solve the QSAP with a linear programming relaxation approach. A consistent segmentation among the different views is obtained with the rate-distortion optimization. This representation retrieves a unique 3D planar decomposition of the scene. A method to encode the 3D planar decomposition for depth map coding application has been proposed.

Depth map coding techniques proposed in this thesis have been validated against current standards showing competitive results against HEVC,MV-HEVC and 3D-HEVC standards.

Future Work

In this Thesis the coding of depth maps have been carried out over a single-view and a multi-view configuration. Some of the future work possibilities include:

- Exploiting temporal redundancy: video can be represented more efficiently by sending only changes between consecutive images instead of coding all regions repeatedly. An extension of this work to handle video sequence would allow to reduce the coding cost for each frame individually. The QSAP optimization problem can be formulated to work with multiple hierarchies of different frames.

- Deep learning: the problems of depth estimation and super-resolution from a single monocular image is being studied with deep learning techniques. In addition of multi-view clues, an estimated depth map obtained with monocular techniques or an up-sampled depth map can be used as extra information in the construction of the depth map coding partition.

- Delay and complexity: the different algorithms presented in this work have been developed without constraints in the delay or complexity of the different elements. The integration of the different parts as part of a feasible coding scheme should consider these limitations.

- Depth inconsistencies: the performance of the described methods varies depending on the sequences considered. The depth maps used in this work are considered as ground truth and are not modified. Inconsistencies between views or between color images and depth map are not preprocessed. These inconsistencies penalize the proposed encoding techniques as it is assumed that color images and depth maps are well aligned as well as that depth of different views is consistent across views. Creating new synthetic sequences or improving the current depth estimation techniques will provide a larger database to analyze the performance of the different depth map coding methods.

- Content-based representation: the 3D representation introduced in this thesis can be used in further task as object recognition or gesture detection.

- Adaptive encoding: in the depth map coding methods of this thesis all regions are treated equally. Segmentation-based coding techniques can

select different quality parameters for each region, assigning more bits in the most relevant areas of the image.

- 3D region models: the 3D scene representation and multi-view coding use 3D plane models to represent the scene. Region models of greater complexity such as quadric surfaces can be added on the generic optimization step to represent the scene.

Appendix A

Databases

In this appendix the MVD sequences used in the thesis are presented. In first place the MVD sequences used in the development of the MVC and 3D-HEVC standards from MPEG are presented and then the Middlebury Stereo Dataset.

A.1 MVD Sequences

The MVD camera sequences in this Section were captured by synchronized cameras with camera arrays. The MVD sequence in different configurations. The MVD sequences that were used in this work are the following: *kendo* [TFT+09], *balloons* [TFT+09], *breakdancers* [ZKU+04], *ballet* [ZKU+04], *undo dancer* [Nok09] and *ghost town* [Nok09]. In the following it is detailed the views used in the experiments and characteristics of each sequence. For each sequence samples frame (frame 51) of the three views used are shown.

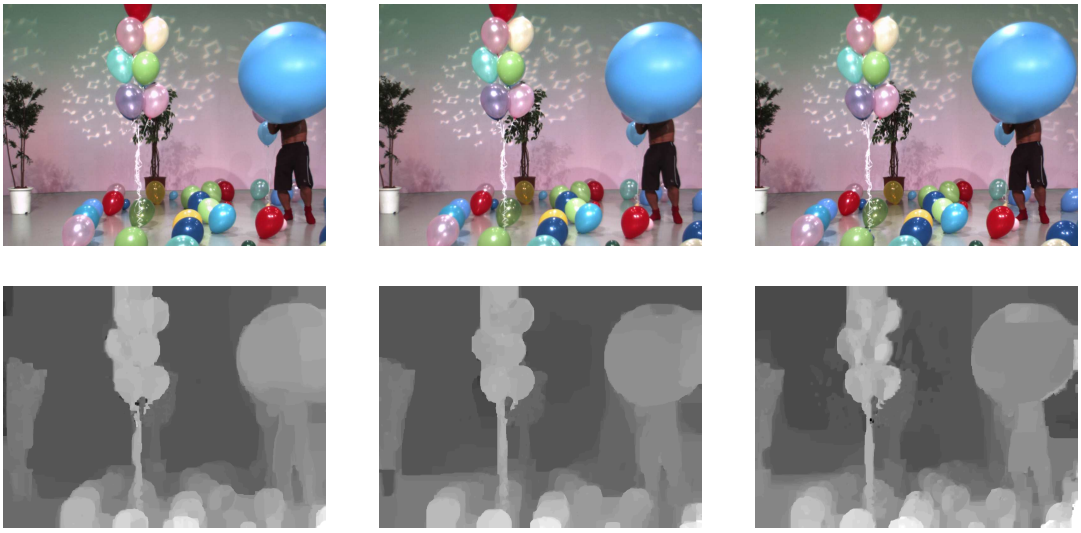
A.1.1 *Kendo* and *Balloons*

- Number of cameras: 7
- Spatial resolution: 1024x768
- Frame rate: 30 fps

- Left - Reference - Right Camera: 1 - 3 - 5

Left view Reference view Right view

Figure A.1: *Kendo* sequence.



Left view Reference view Right view

Figure A.2: *Balloons* sequence.

A.1.2 *Ballet* and *Breakdancers*

- Number of cameras: 8

- Spatial resolution: 1024x768
- Frame rate: 15 fps
- Left - Reference - Right Camera: 1 - 2 - 3

Left view Reference view Right view

Figure A.3: *Ballet* sequence.

Left view Reference view Right view

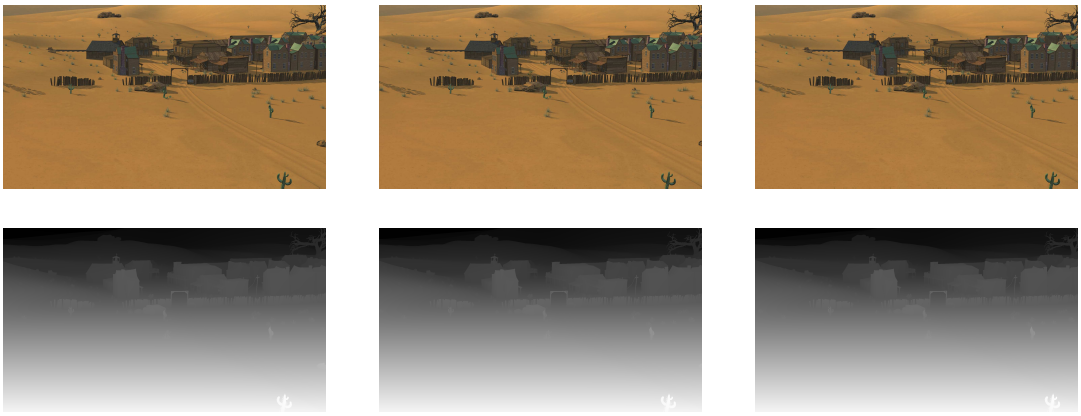
Figure A.4: *Breakdancers* sequence.

A.1.3 *Undo Dancer* and *Ghost Town*

- Number of cameras: 5
- Spatial resolution: 1920x1088
- Frame rate: 25 fps
- Left - Reference - Right Camera: 1 - 5 - 9

Left view Reference view Right view

Figure A.5: *Undo Dancer* sequence.



Left view Reference view Right view

Figure A.6: *Ghost Town* sequence.

A.2 Middlebury Dataset

The Middlebury Stereo Dataset [SS03] consists in many stereo images with ground-truth disparities between several view-points. It is widely used as a database to evaluate different methods of computing disparities. Here the ground-truth disparities are used to generate the depth map for two viewpoints with the disparities to depth transformation:

$$depth(x,y) = \frac{f * B}{disp(x,y)} \tag{A.1}$$

where f is the focal length in pixels, B is the Baseline (distance between the two cameras), disp(x,y) is the disparity of the pixel x,y and depth(x,y) is the absolute depth of the pixel x,y.

The absolute depth is quantized and stored in 8 bits:

$$depth(x,y)_{0to255} = \frac{1.0}{\frac{depth(x,y)}{255.0} * (\frac{1.0}{MinZ} - \frac{1.0}{MaxZ}) + \frac{1.0}{MaxZ}} \tag{A.2}$$

where MinZ and MaxZ is the minimum and maximum depth for the sequence.

The datasets chosen from the website are the 2003 [SS03], 2005 [SP07] and 2006 [HS07] which were the ones used in [OA14]. Each sequence contains color information from several viewpoints and disparity for two of them. These ground-truth disparities have some unknown values which have been filled using the same in-painting method than [OA14] to have a fair comparison between the methods. The images are cropped to a multiple of 8 in order to be able to be encoded with the AVC/H.264 and HEVC encoders.

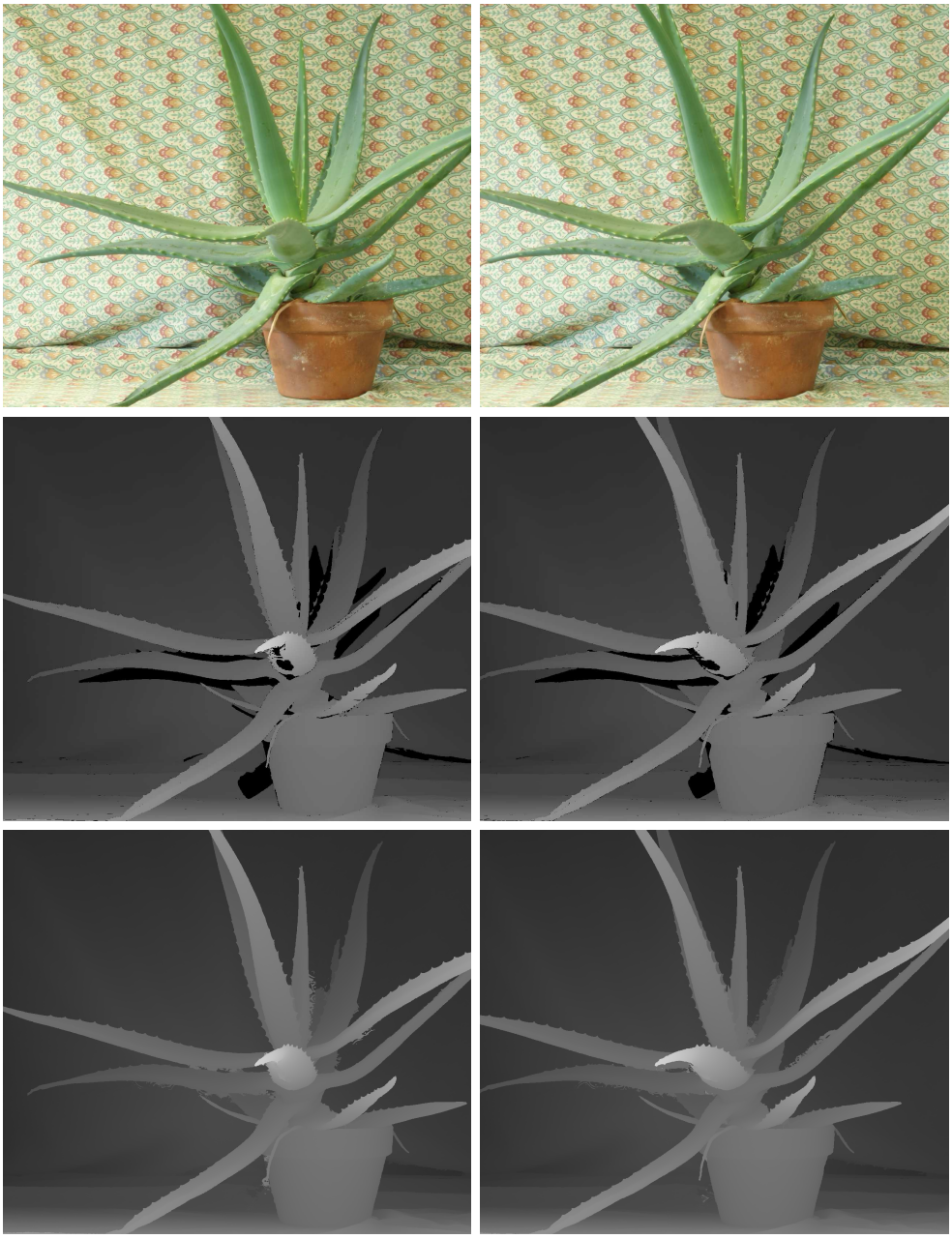


Figure A.7: Color images, depth with holes and depth in-painted for the two views of Aloe sequence.

Appendix B

Experiment Configuration

The configuration to evaluate the performance of the depth map coding techniques proposed in this thesis are shown in Figure B.1. For each sequence, three views are used, the left and right views are encoded and the middle one is employed as the location for the virtual view. Color and depth map images for the three views are available.

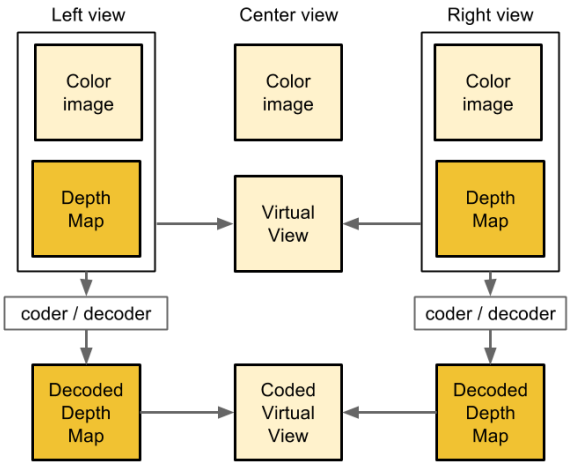


Figure B.1: Virtual view generation.

The performance of the depth map coding technique can be compared in the depth map, giving an overview of the compression efficiency of the method. As depth maps are used to render new images and not to be viewed directly, the relevant comparison is performed in the virtual view. The generation of the

rendered virtual view is done using the color image and the depth map from left and right view.

Software used for the generation of virtual views are:

- View synthesis reference software (VSRS 3.5), in Tech. Rep. ISO/IEC JTC1/SC29/WG11, March 2010.
- HM Software for the HEVC project. Version 16.4 of the software, March 2015.

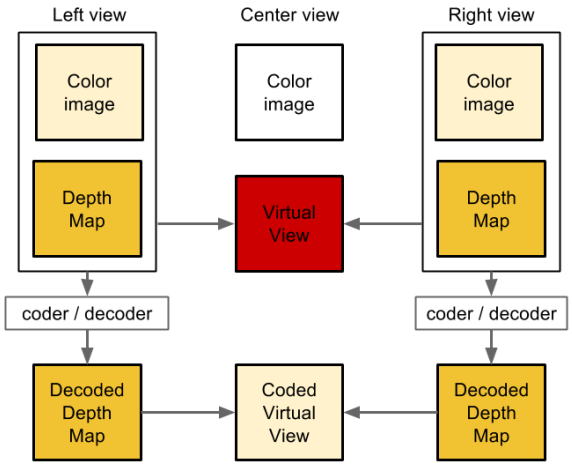


Figure B.2: Evaluation in the rendered virtual view.

VSRS 3.5 is used for *ballet* and *breakdancers* sequences and the HEVC software for the other sequences. The comparative with the rendered view is shown in Figure B.2. Additionally the comparison can be done with the original color image.

Appendix C

Error Measures

C.1 Image Quality Measures

PSNR

PSNR is a term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. PSNR is commonly used to measure the quality of reconstruction of lossy compression codecs. The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation to human perception of reconstruction quality. Although a higher PSNR generally indicates that the reconstruction is of higher quality, in some cases is not true, it is only conclusively valid when compares results from the same codec type and same content.

PSNR is most easily defined via the mean squared error (MSE). Given a noise-free $m \times n$ monochrome image x and its noisy approximation y, MSE is defined as:

$$MSE(x, y) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [x(i, j) - y(i, j)]^2 \tag{C.1}$$

The PSNR is defined as:

$$PSNR(x, y) = 10 \log_{10} \left(\frac{MAX_x^2}{MSE(x, y)} \right) \tag{C.2}$$

MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three. Alternately, for color images the image is converted to a different color space and PSNR is reported against each channel of that color space.

SSIM

The SSIM index [WBSS04] is a method for measuring the similarity between two images. SSIM is designed to improve on traditional methods like PSNR and MSE, which have proven to be inconsistent with human eye perception. A difference of PSNR and MSE, SSIM considers image degradation as perceived change in structural information. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene. The SSIM metric is calculated on various windows of an image.

x and y are discrete non-negative signals; μ_x, σ_x^2, and σ_{xy} are the mean value of x, the variance of x, and the covariance of x and y, respectively. According to the luminance, contrast, and structure comparison measures were given as follows:

$$l(x, y) = \frac{2\,\mu_x \mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{C.3}$$

$$c(x, y) = \frac{2\,\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{C.4}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3} \tag{C.5}$$

where C_1, C_2 and C_3 are small constants given by $C_1 = (K_1 \cdot L)^2$; $C_2 = (K_2 \cdot L)^2$ and $C_3 = C_2/2$. Here L is the dynamic range of the pixel values, and $K_1 \ll 1$ and $K_2 \ll 1$ are two scalar constants. The general form of the SSIM index between signal x and y is defined as:

$$SSIM(x,y) = l(x,y)c(x,y)s(x,y) = \frac{(2\,\mu_x\mu_y + C_1)(2\,\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x + \sigma_y + C_2)} \tag{C.6}$$

In order to evaluate the image quality this formula is applied only on luma. The resultant SSIM index is a decimal value between -1 and 1, and value 1 is only reachable in the case of two identical sets of data. Typically it is calculated on window sizes of 8×8. The window can be displaced pixel-by-pixel on the image but the authors propose to use only a subgroup of the possible windows to reduce the complexity of the calculation. SSIM is maximal when two images are coinciding.

Bjontegaard metrics

Bjontegaard metric are metrics to compute the difference between two rate-distortion plots. PSNR has been a widely used metric for objective test of video quality. The encoding quality of two codecs can also be compared by using their rate-distortion curves and Bjontegaard proposed a method of calculation of average PSNR differences between curves in [Bjo01].

The method is based on fitting a curve through multiple data points consisting of PSNR-bitrate obtained by encoding the video sequence at different quantization parameter values. An expression for the integral of such curve is formulated for two different codecs. The average difference is then calculated by taking difference between the integrals, divided by the integration interval.

The Bjontegaard model is used to calculate the average PSNR and bit rate differences between two rate-distortion curves obtained from the PSNR measurement when encoding a content at different bit rates. The model reports two values:

- Bjontegaard delta PSNR (BD-PSNR), which corresponds to the average PSNR difference in dB for the same bit rate.
- Bjontegaard delta RATE (BD-RATE), which corresponds to the average bit rate difference in percent for the same PSNR.

C.2 Segmentation Quality Measures

Precision and recall are the basic measures used in evaluating search strategies. In pattern recognition and information retrieval, precision is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved.

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been):

$$Precision = \frac{t_p}{t_p + f_p} \tag{C.7}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{C.8}$$

where t_p,f_p,f_n are the true positives, the false positive and the false negatives respectively.

A precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C).

A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2\,\frac{precision \times recall}{precision + recall} \tag{C.9}$$

This is also known as the F_1 measure, because recall and precision are evenly weighted.

Appendix D

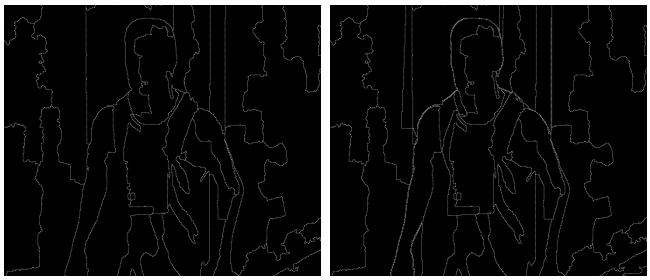
Segmentation-Based Coding Analysis

The choice of methods to perform the texture and contour coding in Chapter 4 was performed without an exhaustive analysis of the alternatives. In this Appendix an analysis of different coding options for contour and texture is developed. A selection of methods presented in Section 2.3 were studied to find the most suitable for depth map coding. Experiments in this Appendix have been performed using the proposed color segmentation and depth map segmentation techniques presented in Chapter 3.

D.1 Contour Coding

The evaluation of the different methods have been carried out over the multiview sequences *undo dancer*, *balloons*, *kendo*, *breakdancers* and *ballet*. Sample images of the MVD sequences and the configuration used are found in Appendix A, B and C. Using an initial color partition, new regions are progressively added by coding the position of the boundaries with the different methods: Chain code, JBIG2 and PAQ (version 8). Intra coding mode and inter mode are explored. In the intra mode, all the boundaries of the partition are encoded without any reference partition. The inter mode uses the color partition as a support to add depth boundaries. The inter mode is obtained just encoding the

location of the new contours (with chain code coding) or by signaling all the pixels that have a new label within the region. Figure D.1 shows an example of the different coding options considered. The new label option has the limitation that two new adjacent regions can not be created inside the same region.



Intra partition for 2 rate-distortion points

Inter partition for 2 rate-distortion points

New labels for 2 rate-distortion points

Figure D.1: Different modes used for adding new regions given an initial partition. In the Inter Partition the boundaries of the original partitions are not coded. In the first two options the image size is (2N*1)x(2M+1) to store all the possible position of the boundaries. In the new labels option the image have size NxM.

The experiments are performed over 20 depth images for the different sequences using an increasing number of new contours. The intra and inter modes are coded with chain code, jbig2 and paq8 while the new labels only for jbig2 and

paq8. Figure D.2 shows the results for each sequence. The horizontal axis shows different percentage of contours added while the vertical axis shows the rate of coding this boundaries in bits per pixel. The intra modes perform worst than the inter modes, which is expectable since little new boundaries are added. The PAQ8 contour intra doubles in rate all the other modes and is removed for the comparison averaging all the sequences in Figure D.3.

As can be seen in the average plot, the chain code inter method is the one that performs better followed closely for the two JBIG2 inter methods. As a result, the chain code inter method is taken as the option to code the contours generated with the depth partition.

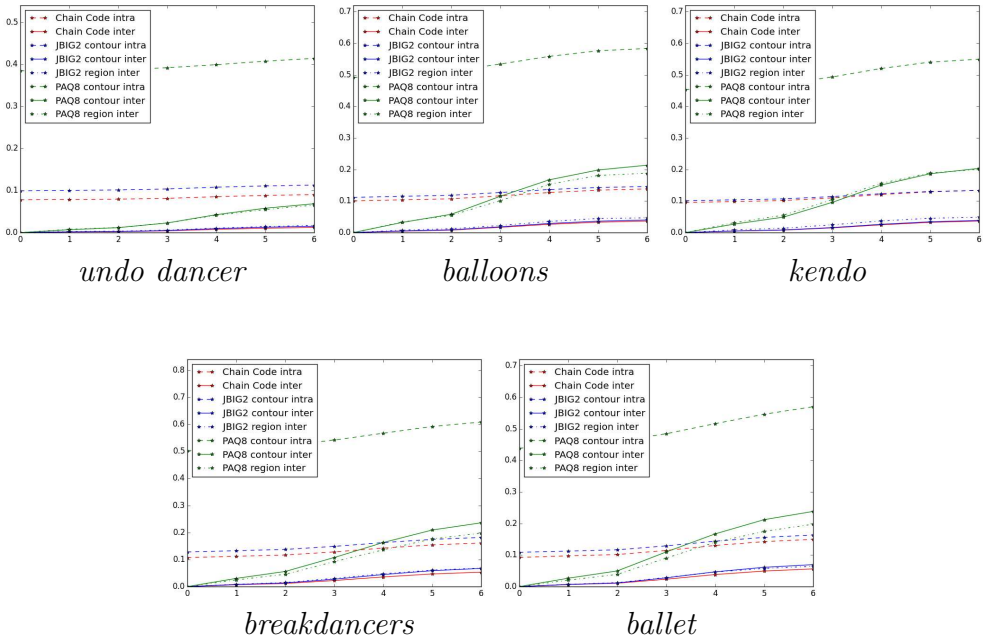


Figure D.2: Results of contour coding for the different sequences.

D.2 Texture Coding

In this section, different methods have been studied for coding the texture in each region. Different partitions have been used to encode the depth maps. Starting from a color based partition, new depth boundaries are progressively added. The different options considered were:

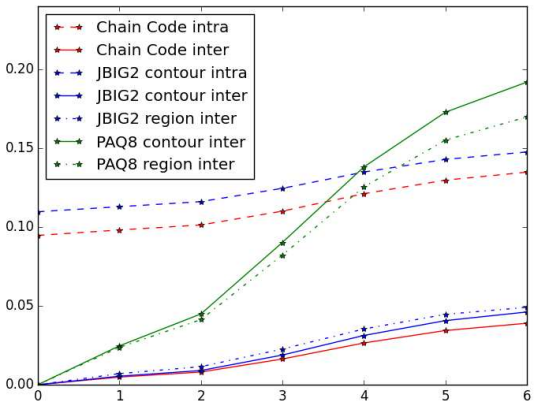


Figure D.3: Contour coding results averaged for all sequences.

- SA-DCT blocks: a SA-DCT with blocks of 8x8 in each region. In blocks where all the 8x8 pixels are in the same region a normal DCT is performed, in border blocks a Shape Adaptive DCT is used. This method is able to obtain high PSNR figures at the cost of employing several coefficients for each region.

- SA-DCT region: a SA-DCT for the whole region together. With this method a unique SA-DCT is performed, but only smooth transitions in the region can be represented.

- Orthobasis: an orthonormal basis is build depending of the shape of the region. It can represent smooth transitions, but their behavior is inconsistent. Increasing the number of bits of each coefficient not always is translated to better distortion values.

- Region Mean: using only the mean depth value to encode each region. It is the simplest codification option and is used as a baseline.

- 3D planes: model used in the 3D depth map segmentation introduced in Section 3.3 and detailed in Subsection 5.1.1.1. Texture information of each region is back-projected to the 3D domain and a plane is fitted with RANSAC. Plane coefficients are stored using the distance of the plane with the camera and their orientation.

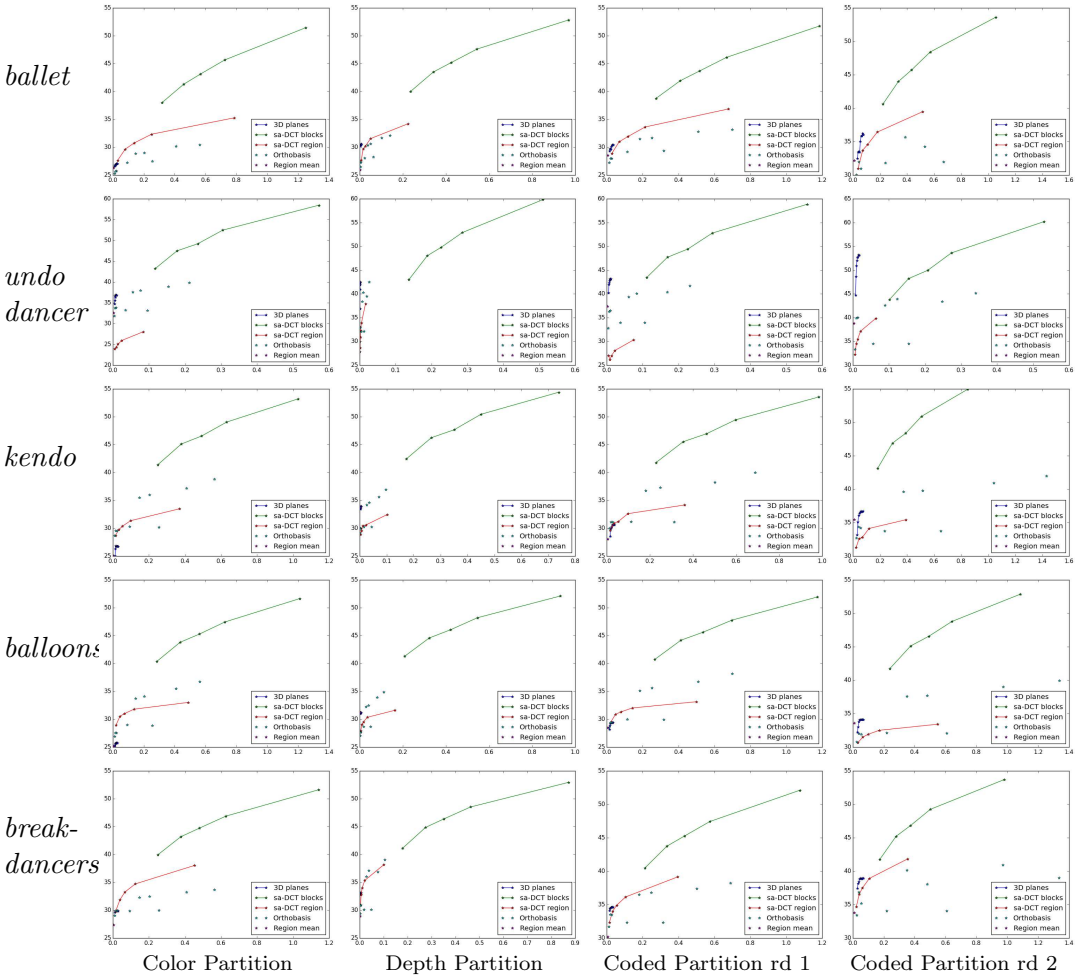


Figure D.4: Texture coding: Results evaluated with the different partitions over depth maps.

The performance of each method is evaluated with the different partitions obtained with: the initial color partition, the initial depth partition and two partitions combining color and depth partitions for two rate-distortion points. In all the plots only the texture cost is represented, as here the interest is on finding the best coding option for a given partition.

Figure D.4 shows the performance of the different partitions evaluated directly over the depth map. The first column shows the result using the color partition. As some meaningful depth edges are missing in these partitions, results using 3D planes are worse than other coding options that are better at representing sharp

edges. In sequences where depth maps are noisier (kendo and balloons) the planes fail at representing that behavior, leading to low rate-distortion figures. Notice that in these sequences the region mean method is better at representing the depth map since none technique can represent correctly the transitions.

The second column shows the results using the depth partition. The performance of the proposed technique is better than the alternatives. Since a hierarchical method that merges segments with a 3D planar model is proposed, the resulting segments can be approximated correctly with a 3D plane. This behavior can be observed also in the other two rows since in there also depth edges are used.

The method that achieves similar results of 3D planes is the SA-DCT region. It is able to obtain similar distortion figures with the same rate. However the SA-DCT blocks needs a much higher rate to represent the depth map. The orthobasis method achieves lower performance across different rates.

The main advantages of the 3D planes method can be observed in Figure D.5. In this Figure the distortion measures are found in the virtual view. The virtual view is generated with the original depth map and with the depth obtained with the different techniques. The proposed texture coding technique is able to obtain better results than the other coding techniques. Even in sequences where the results in D.4 where much lower than the others (for low bitrates) the 3D planes method is able to generate a better representation when evaluated in virtual view.

The proposed 3D coding technique do not represent the depth map with all the small variations inside each region leading to lower results when comparing in depth map, but is able to represent the 3D structure of the scene correctly, obtaining improved results in the virtual view. The result is even better in *undo dancer* or *ballet* where the depth is smoother than the other sequences, resulting in regions quasi-planar.

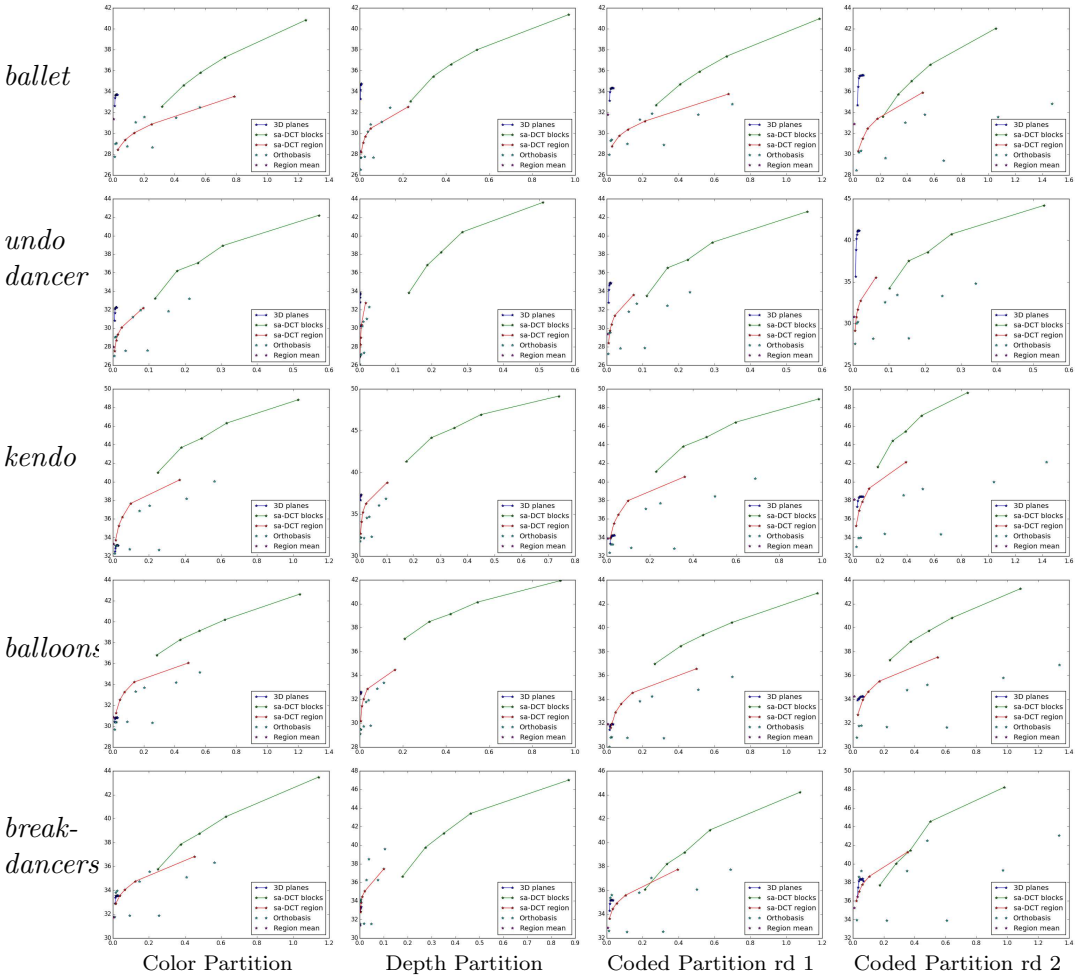


Figure D.5: Texture coding: results evaluated with the different partitions over virtual views.

D.3 Contour and Texture Coding Cost

In this experiment the color and depth partitions are encoded as the previous experiment but in this case contour and texture costs are accumulated in a unique figure. The color partition is assumed to be available and therefore their cost is zero. Results are shown in Figure D.6. The hybrid color+depth technique obtains the final coding partition achieve better results than coding directly the ideal depth map. In the sequences where the differences between color and depth are greater -kendo and balloons- this results are comparable, but in sequences with acceptable depth map the hybrid approach is better than

using only the depth partition.

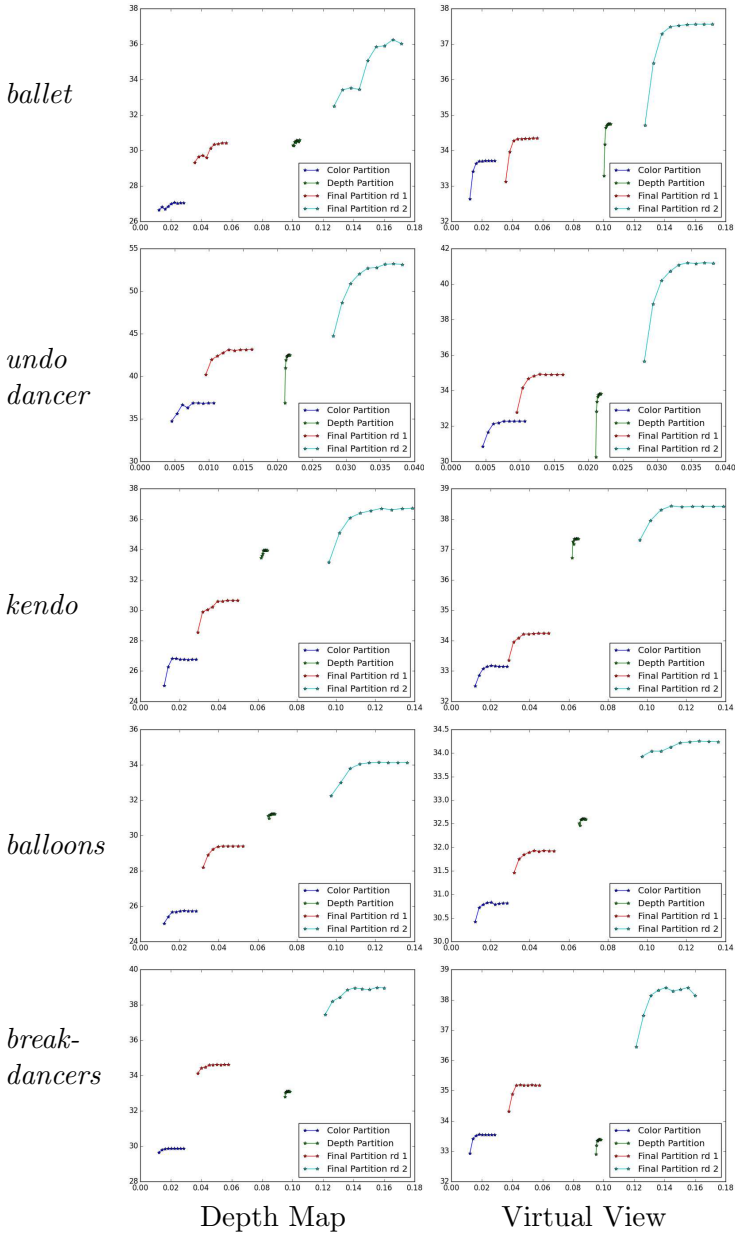


Figure D.6: Contour and texture coding: Results evaluated with the different partitions adding the contour coding cost.

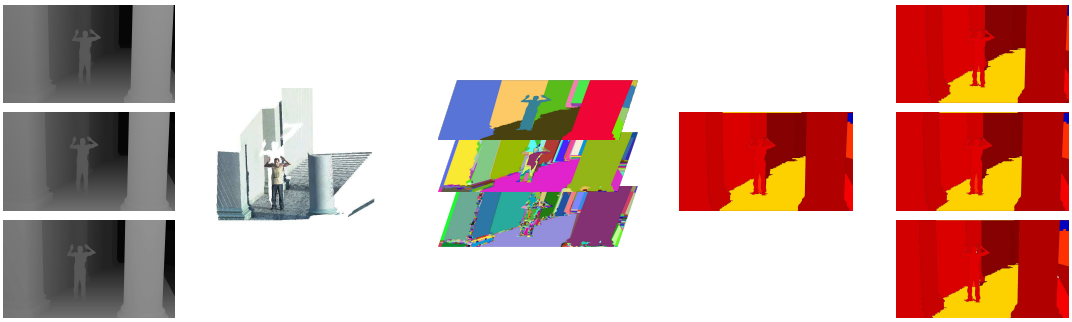
Appendix E

3D Multi-View Scene Representation using Depth Partitions

As a first application for the hierarchical optimization presented in Section 6.1, a method to jointly extract a 3D planar representation and a consistent segmentation of the scene from multiple views is proposed.

The pipeline of the segmentation process can be seen in Figure E.1. The information of each depth map is back-projected to the 3D domain using the camera parameters generating a unique point cloud for all the views. Using the camera model of each view, pixels from the multiple depth maps are back-projected to the 3D world, as shown in Figure E.1.b.

The information of the multiple views will be accumulated to $view_{ref}$. However, the handling of occlusions can not be done in the 2D image domain [LKF16]. The overcome that, a region growing algorithm [RvdHV06] obtains an initial segmentation directly on the 3D point cloud. The initial partition of the hierarchy for $view_{ref}$ is obtained by projecting this point cloud segmentation into $view_{ref}$. In this process, 3D points are projected keeping the foreground elements at each pixel location of $view_{ref}$.



a) $view_i$ depth maps | b) 3D Point Cloud | c) Hierarchical Representation | d) Rate-distortion optimization | e) $view_i$ Final Partitions

Figure E.1: a) Depth images of multiple views. b) The depth information of the multiple views are back-projected to the 3D world to generate a unique point cloud. c) The point cloud is segmented in the 3D domain and projected to a reference view where a hierarchy of regions is built. d) A rate-distortion optimization finds the optimal partition in the hierarchy. e) The partition in the reference view defines a partition in each of the input views.

As the projected partition has holes in pixels where no 3D point from the point cloud is projected, a hole filling over the projected partition obtains the leaves partition LP. The hole filling stage creates a partition with new labels in-between regions with different label while filling the interior of regions.

The hierarchy is built with the BPT merging criterion presented in Section 3.3 using LP as initial partition. Iteratively, the two most similar regions at each step are merged until one region represents the whole image, as shown in Figure 3.1. The merging process uses a 3D planar model to represent each region and a similarity measure that computes the distance between planes. For each node, a 3D planar model is fitted with RANSAC [FB81] using the corresponding points in the 3D point cloud.

By assigning a rate-distortion measure to each node of the hierarchy in $view_{ref}$, the optimal representation in terms of rate-distortion is retrieved. In this application, the main objective is the consistent segmentation of multiple depth maps. Thus, the parameter relevant is the number of regions to represent the

whole multi-view scene. To this end, the cost of coding the depth edges is not incorporated to the rate-distortion optimization.

The optimization parameter λ that defines the rate-distortion trade-off is the quality parameter. By varying λ, different rate-distortion points are found, being able to obtain 3D scene representations with different level of detail.

A search in the hierarchical representation finds the optimal partition in terms of rate-distortion as presented in Section 6.1. The optimal partition obtained defines a partition for $view_{ref}$. To find the final partitions for the set of input $view_i$ views, the point cloud created using the partitioned $view_{ref}$ is projected to each $view_i$. The optimal partition is obtained by replicating the information of active boundaries B^* to each $view_i$, obtaining a consistent segmentation across the views.

Scene Representation Results

In this section, quantitative evaluations of the consistent segmentation results are provided. As the technique aims to segment RGB-D multi-view sequences, the multi-view sequences *ballet*, *undo dancer* and *ghost town* are used. Sequences include color images, depth maps and camera parameters for each view. Sample images and cameras taken for each sequence are detailed in Appendix A.

In order to assess the scene representation technique, different configurations to build the hierarchy are explored. A *2 views* configuration which uses two $view_i$ different than $view_{ref}$ is compared to a *3 views* configuration that also incorporates the $view_{ref}$. A hierarchy obtained solely in $view_{ref}$ is added to observe the benefits of the multi-view approach (*1 view*). To compare the different configurations, each region is represented with the proposed 3D planar model.

Results in reference view

Two results are generated for each configuration. The ones obtained with the hierarchy presented in Section 4.1.1 (following the merging sequence) and the

optimal ones obtained in the rate-distortion optimization.

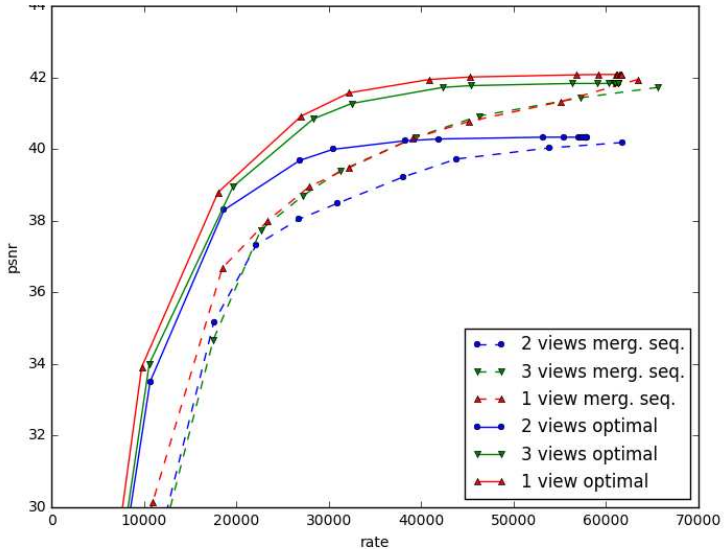


Figure E.2: Rate-distortion results in the $view_{ref}$. For each configuration, the non-optimal solution following the merging sequence and the optimal results are shown.

The optimization process allows keeping the maximal PSNR achievable with a reduction of the number of regions, yielding gains above 3 dB compared to the non optimal approach. The hierarchy obtained by the *1 view* configuration results in higher PSNR since the partition is generated directly in that view. It is noticeable the gain when adding a third view, since the *3 views* configuration obtains an average gain of 2dB's at high rates with respect to the *2 views* configuration.

Results in side views

The optimal partitions found in $view_{ref}$ are translated to the different views $view_i$. As the *1 view* configuration only uses the $view_{ref}$, the $view_i$ partitions are not available, hence the partitions of the *3 views* configuration are used. Figure E.3 shows that the *3 views* configuration obtains higher rate-distortion figures than the *1 view* configuration, supporting the multi-view approach of the proposed algorithm.

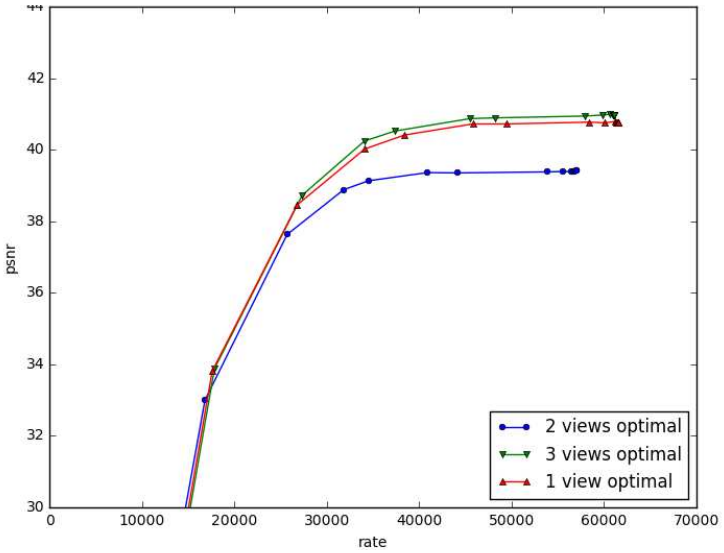


Figure E.3: Averaged Rate-distortion results in the views $view_i$.

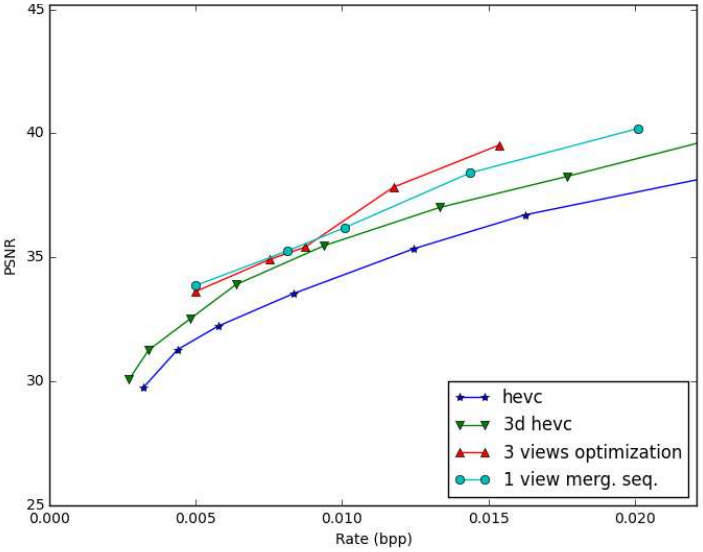


Figure E.4: Depth map coding. Depth maps of $view_i$ are used to render a color image in $view_{ref}$ position.

Depth map coding

As an application of the proposed 3D scene representation, the resulting partitions in $view_i$ are used to encode depth maps in multi-view sequences. The methodology used is as [MMRH16] where, starting from a color partition, the main depth edges are added progressively, obtaining different rate-distortion

points. Once the final partition is build, a 3D plane fitting stage finds the best plane for each region. Here it is compared the method presented in Chapter 5 (the depth partition is obtained independently for each view using the non optimal solution following the merging sequence: 1 view merg.seq.), the optimal 3 views configuration proposed in this work (3 views optimization) and the HEVC standard and 3D-HEVC.

Each depth map is coded in the two $view_i$ and are used to generate a rendered color image in the $view_{ref}$ position (see Appendix B. The resulting color images are compared with the original color image of the $view_{ref}$. In Figure E.4 the rate-distortion results of the method compared with HEVC standards is shown. Since the *3 views* configuration is able to find a better depth partition across views, the depth map obtained with that representation is able to over-perform both [MMRH16] and HEVC results.

Bibliography

[AMFM09] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, *From contours to regions: An empirical evaluation*, IEEE Conference on Computer Vision and Pattern Recognition (Miami, USA), June 2009, pp. 2294–2301.

[AMFM11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik, *Contour detection and hierarchical image segmentation*, IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011), no. 5, 898–916.

[ASS+12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, *SLIC superpixels compared to state-of-the-art superpixel methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence **34** (2012), no. 11, 2274–2282.

[BBR+13] Michael Bergh, Xavier Boix, Gemma Roig, Benjamin Capitani, and Luc Gool, *Seeds: Superpixels extracted via energy-driven sampling*, CoRR **7578** (2013), 13–26.

[Bjo01] G. Bjontegaard, *Calculation of average psnr differences between rd-curves*, April 2001.

[BMC88] M.J. Biggar, O.J. Morris, and A.G. Constantinides, *Segmented-image coding: performance comparison with the discrete cosine transform*, Radar and Signal Processing, IEE Proceedings F **135** (1988), no. 2, 121–132.

[BP14] F. Barrera and N. Padoy, *Piecewise planar decomposition of 3d point clouds obtained from multiple static rgb-d cameras*, 2014 2nd

International Conference on 3D Vision, vol. 1, Dec 2014, pp. 194–201.

[CGW03] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth, *Clustering with qualitative information*, Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), FOCS '03, IEEE Computer Society, 2003, pp. 524–533.

[CKO+11] G Cheung, W. S. Kim, A. Ortega, J. Ishida, and A. Kubota, *Depth map coding using graph based transform and transform domain sparsification*, International Workshop on Multimedia Signal Processing, October 2011, pp. 1–6.

[Dod05] N.A. Dodgson, *Autostereoscopic 3D displays*, Computer **38** (2005), no. 8, 31–36.

[DTPP08] I. Daribo, C. Tillier, and B. Pesquet-Popescu, *Adaptive wavelet coding of the depth map for stereoscopic view synthesis*, Multimedia Signal Processing, 2008 IEEE 10th Workshop on, 2008, pp. 413–417.

[Dun86] J.G. Dunham, *Optimum uniform piecewise linear approximation of planar curves*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **PAMI-8** (1986), no. 1, 67–75.

[DYQZ12] Huiping Deng, Li Yu, Jinbo Qiu, and Juntao Zhang, *A joint texture/depth edge-directed up-sampling algorithm for depth map coding*, Multimedia and Expo (ICME), 2012 IEEE International Conference on, july 2012, pp. 646 –650.

[FB81] Martin A. Fischler and Robert C. Bolles, *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Commun. ACM **24** (1981), no. 6, 381–395.

[Feh04] Christoph Fehn, *Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv*, Proc. SPIE **5291** (2004), 93–104.

[FLG15] M.S. Farid, M. Lucenteforte, and M. Grangetto, *Panorama view with spatiotemporal occlusion compensation for 3D video coding*, IEEE Transactions on Image Processing **24** (2015), no. 1, 205–219.

[Fre61] H. Freeman, *On the encoding of arbitrary geometric configurations*, IRE Transactions on Electronic Computers **EC-10** (1961), no. 2, 260–268.

[GAM13] S. Gupta, P. Arbelaez, and J. Malik, *Perceptual organization and recognition of indoor scenes from RGB-D images*, IEEE Conference on Computer Vision and Pattern Recognition (Portland, USA), June 2013, pp. 564–571.

[GCM+16] Y. Gao, G. Cheung, T. Maugey, P. Frossard, and J. Liang, *Encoder-driven inpainting strategy in multiview video compression*, IEEE Transactions on Image Processing **25** (2016), no. 1, 134–149.

[GEM89] Michael Gilge, Thomas Engelhardt, and Ralf Mehlan, *Coding of arbitrarily shaped image segments based on a generalized orthogonal transform*, Signal Processing: Image Communication **1** (1989), no. 2, 153 – 180, ¡ce:title¿64 kbit/s Coding of Moving Video¡/ce:title¿.

[GVB11] D. Glasner, S. N. Vitaladevuni, and R. Basri, *Contour-based joint clustering of multiple segmentations*, Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (Washington, DC, USA), CVPR '11, IEEE Computer Society, 2011, pp. 2385–2392.

[HKM+98] P.G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W.J. Rucklidge, *The emerging jbig2 standard*, Circuits and Sys-

tems for Video Technology, IEEE Transactions on **8** (1998), no. 7, 838–848.

[HS07] H. Hirschmuller and D. Scharstein, *Evaluation of cost functions for stereo matching*, IEEE Conference on Computer Vision and Pattern Recognition, June 2007, pp. 1–8.

[Jag11] F. Jager, *Contour-based segmentation and coding for depth map compression*, VCIP 2011, Nov. 2011, pp. 1–4.

[KATS11] Hema S. Koppula, Abhishek Anand, Joachims Thorsten, and Ashutosh Saxena, *Semantic labeling of 3d point clouds for indoor scenes*, Advances in Neural Information Processing Systems 24 (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), Curran Associates, Inc., 2011, pp. 244–252.

[KIAK97] H. Katata, N. Ito, T. Aono, and H. Kusao, *Object wavelet transform for coding of arbitrarily shaped image segments*, Circuits and Systems for Video Technology, IEEE Transactions on **7** (1997), no. 1, 234–237.

[KIK85] Murat Kunt, Athanassios Ikonomopoulos, and Michel Kocher, *Second-generation image-coding techniques*, Proceedings of the IEEE **73** (1985), no. 4, 549–574.

[KM95] R. Kresch and D. Malah, *Quadtree and bit-plane decompositions as particular cases of the generalized morphological skeleton*, 1995, pp. 995–999.

[KOLT15] 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 **24** (2015), no. 11, 3534–3545.

[KS14] D. I. Kim and G. S. Sukhatme, *Semantic labeling of 3d point clouds with object affordance for robot manipulation*, 2014 IEEE Inter-

national Conference on Robotics and Automation (ICRA), May 2014, pp. 5578–5584.

[KSS12] A. Kowdle, S.N. Sinha, and R. Szeliski, *Multiple view object cosegmentation using appearance and stereo cues*, European Conference on Computer Vision (Firenze, Italy), October 2012, pp. 789–803.

[LKF16] Chen Liu, Pushmeet Kohli, and Yasutaka Furukawa, *Layered scene decomposition via the occlusion-crf*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[LLZ+15] J. Lei, S. Li, C. Zhu, M.T. Sun, and C. Hou, *Depth coding based on depth-texture motion and structure similarities*, IEEE Transactions on Circuits and Systems for Video Technology **25** (2015), no. 2, 275–286.

[LZ15] B. Liang and L. Zheng, *A survey on human action recognition using depth sensors*, Digital Image Computing: Techniques and Applications (DICTA), 2015 International Conference on, Nov 2015, pp. 1–8.

[Mah05] Matthew V Mahoney, *Adaptive weighing of context models for lossless data compression*, 2005.

[MD08] M. Maitre and M.N. Do, *Joint encoding of the depth image based representation using shape-adaptive wavelets*, Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on, 2008, pp. 1768–1771.

[Mea07] P. Merkle et al., *Multi-view video plus depth representation and coding*, ICIP, Oct 2007.

[MFTM01] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, International Conference on Computer Vision (Vancouver, Canada), vol. 2, July 2001, pp. 416–423.

[MG96] Ferran Marques and Antoni Gasull, *Partition coding using multigrid chain code and motion compensation*, Proceedings of the 1996 IEEE International Conference on Image Processing, 1996, pp. 935–938.

[MK09] B. Micusik and J. Kosecka, *Piecewise planar city 3d modeling from street view panoramic sequences*, Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, June 2009, pp. 2906–2912.

[MMMW15] P. Merkle, K. Muller, D. Marpe, and T. Wiegand, *Depth intra coding for 3D video based on geometric primitives*, IEEE Transactions on Circuits and Systems for Video Technology **PP** (2015), no. 99, 1–1.

[MMRH13] M. Maceira, J.R. Morros, and J. Ruiz-Hidalgo, *Fusion of colour and depth partitions for depth map coding*, Digital Signal Processing (Santorini, Greece), 07/2013 2013.

[MMRH15] ———, *Region-based depth map coding using a 3d scene representation*, IEEE International Conference on Acoustics, Speech and Signal Processing (Brisbane, Australia), 04/2015 2015.

[MMRH16] Marc Maceira, Josep-Ramon Morros, and Javier Ruiz-Hidalgo, *Depth map compression via 3d region-based representation*, Multimedia Tools and Applications (2016), 1–24.

[MMS+08] P. Merkle, Y. Morvan, A. Smolic, D. Farin, K. Muller, P.H.N. de With, and T. Wiegand, *The effect of depth compression on multiview rendering quality*, 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, May 2008, pp. 245–248.

[MMW11] K. Muller, P. Merkle, and T. Wiegand, *3-D video representation using depth maps*, Proceedings of the IEEE **99** (2011), no. 4, 643–656.

[MOF15] T. Maugey, A. Ortega, and P. Frossard, *Graph-based representation for multiview image geometry*, IEEE Transactions on Image Processing **24** (2015), no. 5, 1573–1586.

[Mor04] J.R. Morros, *Optimization of segmentation based video sequence coding techniques: Application to content based functionalities*, phd, Universitat Politècnica de Catalunya (UPC), 2004.

[MRHM12] M. Maceira, J. Ruiz-Hidalgo, and J.R. Morros, *Depth map coding based on a optimal hierarchical region representation*, 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2012, oct. 2012, pp. 1 –4.

[MS86] P. Maragos and R.W. Schafer, *Morphological skeleton representation and coding of binary images*, Acoustics, Speech and Signal Processing, IEEE Transactions on **34** (1986), no. 5, 1228–1244.

[MSG93] F. Marqués, J. Sauleda, and T. Gasull, *Shape and location coding for contour images*, Picture Coding Symposium (Lausanne, Switzerland), March 1993, pp. 18.6.1–18.6.2.

[MZZF11] S. Milani, P. Zanuttigh, M. Zamarin, and S. Forchhammer, *Efficient depth map compression exploiting segmented color data*, IEEE International Conference on Multimedia and Expo, July 2011, pp. 1–6.

[Nok09] Nokia, *3dv test sequences*, 2009.

[OA14] B.O. Ozkalayci and A.A. Alatan, *3D planar representation of stereo depth images for 3DTV applications*, IEEE Transactions on Image Processing **23** (2014), no. 12, 5222–5232.

[OBL+04] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, *Video coding with H.264/AVC: tools, performance, and complexity*, IEEE Circuits and Systems Magazine **4** (2004), no. 1, 7–28.

[OR98] A. Ortega and K. Ramchandran, *Rate-distortion methods for image and video compression*, IEEE Signal Processing Magazine **15** (1998), no. 6, 23–50.

[Pin98] Armando J. Pinho, *A method for encoding region boundaries based on transition points*, Image and Vision Computing **16** (1998), no. 3, 213 – 218.

[PT14] Jordi Pont-Tuset, *Image segmentation evaluation and its application to object detection*, Ph.D. thesis, Universitat Politècnica de Catalunya, UPC BarcelonaTech, 2014.

[RBF12] X. Ren, L. Bo, and D. Fox, *Rgb-(d) scene labeling: Features and algorithms*, Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, June 2012, pp. 2759–2766.

[RHMA+12] J. Ruiz-Hidalgo, J.R. Morros, P. Aflaki, F. Calderero, and F. Marqus, *Multiview depth coding based on combined color/depth segmentation*, Journal of Visual Communication and Image Representation **23** (2012), no. 1, 42 – 52.

[RvdHV06] T. Rabbani, F. A. van den Heuvel, and G. Vosselman, *Segmentation of point clouds using smoothness constraint*, ISPRS Commission V Symposium 'Image Engineering and Vision Metrology', January 2006, pp. 248–253.

[SBM95] T. Sikora, S. Bauer, and B. Makai, *Efficiency of shape-adaptive 2-d transforms for coding of arbitrarily shaped image segments*, Circuits and Systems for Video Technology, IEEE Transactions on **5** (1995), no. 3, 254–258.

[SC96] H. Sanderson and G. Crebbin, *Region-based image coding using polynomial intensity functions*, Vision, Image and Signal Processing, IEE Proceedings - **143** (1996), no. 1, 15–22.

[Sch07] Larry Schumaker, *Spline Functions: Basic Theory*, 3 ed., Cambridge Mathematical Library, Cambridge University Press, Cambridge, 2007.

[SG00] P. Salembier and L. Garrido, *Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval*, IEEE Trans. on IP **9** (2000), no. 4, 561–576.

[SHKF12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, *Indoor segmentation and support inference from RGBD images*, European Conference on Computer Vision (Firenze, Italy), vol. 5, October 2012, pp. 746–760.

[SLJ+14] F. Shao, W. Lin, G. Jiang, M. Yu, and Q. Dai, *Depth map coding for view synthesis based on distortion analyses*, IEEE Journal on Emerging and Selected Topics in Circuits and Systems **4** (2014), no. 1, 106–117.

[SM95] T. Sikora and B. Makai, *Shape-adaptive dct for generic coding of video*, IEEE Trans. on CSVT **5** (1995), no. 1, 59–62.

[SMAP14] S. Shahriyar, M. Murshed, M. Ali, and M. Paul, *Efficient coding of depth map by exploiting temporal correlation*, International Conference on Digital lmage Computing: Techniques and Applications, November 2014, pp. 1–8.

[SMLN11] L. A. Schwarz, D. Mateus, J. Lallemand, and N. Navab, *Tracking planes with time of flight cameras and j-linkage*, Applications of Computer Vision (WACV), 2011 IEEE Workshop on, Jan 2011, pp. 664–671.

[SMM+06] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand, *3D video and free viewpoint video - technologies, applications and MPEG standards*, IEEE International Conference on Multimedia and Expo (Toronto, Canada), July 2006, pp. 2161–2164.

[SOHW12] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand, *Overview of the high efficiency video coding (HEVC) standard*, IEEE Transactions on Circuits and Systems for Video Technology **22** (2012), no. 12, 1649–1668.

[SP07] D. Scharstein and C. Pal, *Learning conditional random fields for stereo*, IEEE Conference on Computer Vision and Pattern Recognition, June 2007, pp. 1–8.

[SS03] D. Scharstein and R. Szeliski, *High-accuracy stereo depth maps using structured light*, IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, June 2003, pp. 195–202.

[SSO09] A. Sanchez, G. Shen, and A. Ortega, *Edge-preserving depth-map coding using graph-based wavelets*, Asilomar Conference on Signals, Systems and Computers, Nov. 2009, pp. 578–582.

[SSS09] S.N. Sinha, D. Steedly, and R. Szeliski, *Piecewise planar stereo for image-based rendering*, International Conference on Computer Vision (Kyoto, Japan), September 2009, pp. 1881–1888.

[TCM+16] G. Tech, Y. Chen, K. Mller, 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 **26** (2016), no. 1, 35–49.

[TFT+09] Masayuki Tanimoto, Toshiaki Fujii, Mehrdad Panahpour Tehrani, Menno Wildeboer, Norishige Fukushima, and Hisayoshi Furihata, *Moving multiview camera test sequences for mpeg-ftv*, ISO/IEC JTC1/SC29/WG11 M **16922** (2009).

[TSMW12] G. Tech, H. Schwarz, K. Muller, and T. Wiegand, *3D video coding using the synthesized view distortion change*, Picture Coding Symposium, May 2012, pp. 25–28.

[VAM15] D. Varas, M. Alfaro, and F. Marques, *Multiresolution hierarchy co-clustering for semantic segmentation in sequences with small*

variations, 2015 IEEE International Conference on Computer Vision (ICCV), Dec 2015, pp. 4579–4587.

[VDV16] Cedric Verleysen and Christophe De Vleeschouwer, *Piecewise-planar 3d approximation from wide-baseline stereo*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[VMS08] V. Vilaplana, F. Marqués, and P. Salembier, *Binary partition trees for object detection*, IEEE Transactions on Image Processing **17** (2008), no. 11, 2201–2216.

[WBSS04] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, Image Processing, IEEE Transactions on **13** (2004), no. 4, 600–612.

[WLC+15] A. Wang, J. Lu, J. Cai, G. Wang, and T. J. Cham, *Unsupervised joint feature learning and encoding for rgb-d scene labeling*, IEEE Transactions on Image Processing **24** (2015), no. 11, 4459–4473.

[WNC87] Ian H Witten, Radford M Neal, and John G Cleary, *Arithmetic coding for data compression*, Communications of the ACM **30** (1987), no. 6, 520–540.

[WSBL03] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra, *Overview of the h.264/avc video coding standard*, Circuits and Systems for Video Technology, IEEE Transactions on **13** (2003), no. 7, 560–576.

[YVEM15] F. Yin, S. A. Velastin, T. Ellis, and D. Makris, *Learning multi-planar scene models in multi-camera videos*, IET Computer Vision **9** (2015), no. 1, 25–40.

[ZKU+04] Larry Zitnick, Sing Bing Kang, Matt Uyttendaele, Simon Winder, and Rick Szeliski, *High-quality video view interpolation using a*

layered representation, ACM SIGGRAPH, vol. 23, Association for Computing Machinery, Inc., August 2004, p. 600608.

[ZR72] Charles T. Zahn and Ralph Z. Roskies, *Fourier descriptors for plane closed curves*, Computers, IEEE Transactions on **C-21** (1972), no. 3, 269–281.

[ZYL+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 **25** (2016), no. 12, 5877–5891.