UNIVERSITAT POLITÈCNICA DE CATALUNYA

Department of Signal Theory and Communications

M.Sc. dissertation

Segmentation based coding of depth Information for 3D video

Author: **Payman Aflaki Beni**

Advisor: **Professor Javier Ruiz Hidalgo**

Barcelona, July 2009

Contents

1. Abstract

Increased interest in 3D artifact and the need of transmitting, broadcasting and saving the whole information that represents the 3D view, has been a hot topic in recent years. Knowing that adding the depth information to the views will increase the encoding bitrate considerably, we decided to find a new approach to encode/decode the depth information for 3D video.

In this project, different approaches to encode/decode the depth information are experienced and a new method is implemented which its result is compared to the best previously developed method considering both bitrate and quality (PSNR).

First the depth image is segmented using a BPT (binary partition tree) method which divides the image to different homogeneous partitions based on gray level value of its pixels. Then using this partitioned image, the contours are coded exploiting chain code contour coding and to encode each partition Orthobasis method is exploited. (**Fig. 1.1**)

Fig. 1.1: Steps of a successful segmentation based coding

Considering that in such a way the quality of final image is high and the bitrate is high as well, we had to make decision whether this approach is suitable or not. Regarding the very high bitrate which was mostly devoted to contour coding while using the lossless mode (depending on the number of regions, contour coding takes even more than 90% of the total bitrate), we tried to tackle the problem in such a way that the bitrate decreases along with having a satisfactory quality. In the proposed method, a combination of decoded view segmentation with completion by original depth segmentation is explained and implemented.

The reference for our results is H.264/AVC, which is the pioneer in the field of video coding nowadays. In this thesis, because of the work load, no time interpolation has been exploited. It means without using the dependency of images through the time, a very satisfactory result has been achieved. So, the outcome of H.264 was processed in two cases: first, while using it as an I frame intra coder and second, while using the optimized H.264. This means, using encoded depth information of each original camera, we tried to create the virtual views based on distance distribution between each pair of nearby original cameras. In this case we could study the PSNR of different coding methods based on the quality of virtual rendered images.

The results were satisfactory and for specific and reasonable quality of decoded depth images, our new segmentation based coding method out performed H.264. This means, while having the same PSNR, we managed to achieve a lower bitrate.

2. Introduction

It is believed that 3D is the next major revolution in the history of TV. Both at professional and consumer electronics exhibitions, companies are eager to show their new 3D products that always attract a lot of interest. Obviously, if a workable and commercially acceptable solution can be found, the introduction of 3D-TV will generate a huge replacement market for the current 2D-TV sets. In this decade, it is expected that technology will progress far enough to make a full 3D-TV application available on the mass consumer market, including content generation, coding, transmission and display.

In recent years, various multimedia services have become available and the demand for realistic multimedia systems is growing rapidly. A number of three-dimensional (3D) video technologies, such as holography, two-view stereoscopic system with special glasses, 3D wide screen cinema, and multi-view video have been studied. Among them, multi-view video (MVV) is the key technology for various applications including free-viewpoint video (FVV) [1], [2], free-viewpoint television (FVT), 3DTV[3], teleconference, and surveillance systems. The traditional video is a two-dimensional (2D) medium and only provides a passive way for viewers to observe the scene. However, 3D and free viewpoint video are new types of natural video media that expand the user's sensation far beyond what is offered by traditional media. The first offers a 3D depth impression of the observed scenery, and the second one, on the other hand, is characterized by providing the user the ability to interactively select an arbitrary viewpoint (depending on the number of views that has been put in the structure of the FVT) in the video scene as known from computer graphics. Both technologies do not exclude each other, but rather can be combined into one system. A common characteristic of such technologies is that they use MVV data, where a real world scene is recorded by multiple synchronized cameras. MVV can offer arbitrary viewpoints of dynamic scenes and thus allow more realistic video.

Target applications include broadcast television and other forms of video entertainment, as well as surveillance. These applications are enabled through convergence of technologies from computer graphics, computer vision, multimedia and related fields, and rapid progress in research covering the whole processing chain from capturing, signal processing, data representation, compression, transmission, display and interaction. Some of these application scenarios may be based on proprietary systems, as for instance already employed for (post-) production of movies and TV content. On the other hand there are also application scenarios that require interoperable systems, such as 3DTV broadcast or free viewpoint video on DVD. This may open huge consumer market for 3D displays, set-top boxes, media, content, DVDs, HD-DVDs, BRDs, etc., along with the corresponding equipment for production, transmission, etc.

The multi-view video includes multi-viewpoint video sequences captured by multiple cameras at the same time, but at different positions. In the context of MVV, this format is combined with multi-view video to the multi-view video + depth (MVD) format, consisting of multiple color videos with associated multiple depth data of one scene. MVD representations cause a vast amount of data to be stored or transmitted to the user.

It means that because of the increased number of cameras, the MVD contains a large amount of data. Since this system has serious limitations on information distribution applications, such as broadcasting, network streaming services, and other commercial applications, we need to compress the multi-view sequence efficiently without sacrificing visual quality significantly. As a result, efficient compression techniques are essential for realizing such applications. Previous work presented various solutions for multi-view video coding (MVC), mostly based on H.264 with combined temporal and inter-view prediction, as well as different approaches for depth image coding, like transform- or wavelet-based depth compression.

In this step, since depth-maps are not directly viewed and knowing that traditional image compression methods have been designed to provide maximum perceived visual quality, a direct application is suboptimal for depth-map compression. In other words, the sensitivity of the rendering error depends on image content as well as on the depth map. The depth maps are never directly viewed-rather they are used for rendering new images whose quality is of importance in the application, so, in this case, having the highest perceived visual quality of depth image is not the most important issue. For example: errors in the depth map close to an intensity edge can result in ugly rendering artifacts, while errors on smooth surface will be unnoticed.

This urged us to try to exploit a segmentation based approach to encode/decode the depth video. So while preserving the contours, using different possible methods like chain code or multi grid chain code, we can send the whole information of each partition with a texture coding method like Polynomial or Orthobasis. Combining this segmentation method we can preserve the good quality of the final rendered images while having the contours of the depth image as much similar as possible to the original ones.

Using this approach on two sequences of “Ballet ” and “Breakdancers “ (which are the two well known and commonly used sequences in the field of depth coding), the bitrate and quality for different combinations of mentioned methods will be calculated and the results are compared with each other and considering the nicety of final images the best one will be chosen. As well, as a reference for the comparison, the results will be compared with those of H.264 achieved by two different configurations, to see the performance of the proposed methods. Finally in this project a satisfactory method to compress the depth information is achieved which can compete or even outperform the reference in specific qualities.

This report is organized as follows. Section 3 is about the background needed for this research. The segmentation methods are introduced in section 4. Section 5 covers the segmentation based coding subject while in section 6 the entropy coding is explained. Section 7 is about virtual view rendering, especially its application to evaluate the effects of depth-image compression. In section 8 different approaches that leaded us to the new segmentation based coding method are discussed and the proposed method is explained. Experimental results are completely presented in section 9. Finally, the paper concludes with Section 10.

3. Background

3.1. Encoding methods

To encode a video, different methods have been exploited so far, which mostly are concentrating on best perceived visual quality of the decoded video.

State-of-the-art video coding, such as H.264, addresses coding of natural scenes and is good at exploiting similarities between neighboring frames containing smooth structures.

Video compression refers to reducing the quantity of data used to represent digital video images, and is a combination of spatial image compression and temporal motion compensation. Video compression is an example of the concept of source coding in information theory. Compressed video can effectively reduce the bandwidth required to transmit video via terrestrial broadcast, via cable TV, or via satellite TV services. Particularly in this project, we use an efficient method to encode the images who will create the video. As well the middle virtual images will be rendered using some algebraic methods and image projection approaches.

H.264 employs state-of-the-art video coding technologies and promises to have significant impact on future generations of video coding products, markets and services. So in this thesis a method of encoding/decoding for depth images will be introduced and explained. This will be followed by the results and the comparison with the reference.

For video coding, we can name two major standards of MPEG and H.26x which have a joint point in H.264. Below is a summary of both standards and a short description about their history and different versions:

MPEG

(Moving **P**ictures **E**xperts **G**roup) An ISO/ITU standard for compressing digital video. It is the universal standard for digital terrestrial, cable and satellite TV, DVDs and digital video recorders (DVRs).

MPEG is an asymmetrical system. It takes longer to compress the video than it does to decompress it in the DVD player, PC, set-top box or digital TV set. As a result, in the early days, compression was performed only in the studio. As chips advanced and became less costly, they enabled digital video recorders, such as Tivos, to convert analog TV to MPEG and record it on disk in real time.

The MPEG compression methodology is considered asymmetric while the encoder is more complex than the decoder. The encoder needs to be algorithmic or adaptive whereas the decoder is 'dumb' and carries out fixed actions. This is considered advantageous in applications such as broadcasting where the number of expensive complex encoders are small but the number of simple inexpensive decoders is large. This approach of the ISO

to standardization in MPEG is considered novel because it is not the encoder which is standardized; instead, the way in which a decoder shall interpret the bit stream is defined. A decoder which can successfully interpret the bitstream is said to be compliant. The advantage of standardizing the decoder is that over time encoding algorithms can improve yet compliant decoders will continue to function with them. The MPEG standards give very little information regarding structure and operation of the encoder and implementers can supply encoders using proprietary algorithms. This gives scope for competition between different encoder designs which means that better designs can evolve and users will have greater choice because of different levels of cost and complexity can exist in a range of coders yet a compliant decoder will operate with them all.

MPEG uses lossy compression within each frame similar to JPEG, which means pixels from the original images are permanently discarded. It also uses inter-frame coding, which further compresses the data by encoding only the differences between periodic frames. MPEG performs the actual compression using the discrete cosine transform (DCT) method.

The MPEG standards consist of different parts where each part covers a certain aspect of the whole specification. MPEG standardization can be devided as below:

- **MPEG-1 (Video CDs)**

  Although MPEG-1 supports higher resolutions, it is typically coded at 352x240 x 30fps (NTSC) or 352x288 x 25fps (PAL/SECAM). Full 704x480 and 704x576 frames (BT.601) were scaled down for encoding and scaled up for playback. MPEG-1 uses the YCbCr color space with 4:2:0 sampling, but did not provide a standard way of handling interlaced video. Data rates were limited to 1.8 Mbps, but often exceeded.

- **MPEG-2 (DVD, Digital TV)**

  MPEG-2 provides broadcast quality video with resolutions up to 1920x1080. It supports a variety of audio/video formats, including legacy TV, HDTV and five channel surround sound. MPEG-2 uses the YCbCr color space with 4:2:0, 4:2:2 and 4:4:4 sampling and supports interlaced video. Data rates are from 1.5 to 60 Mbps.

- **MPEG-4 (All Inclusive and Interactive)**

  MPEG-4 is an extremely comprehensive system for multimedia representation and distribution. Based on a variation of Apple's QuickTime file format, MPEG-4 offers a variety of compression options, including low-bandwidth formats for transmitting to wireless devices as well as high-bandwidth for studio processing..

  MPEG-4 also incorporates AAC, which is a high-quality audio encoder. MPEG-4 AAC is widely used as an audio-only format.

A major feature of MPEG-4 is its ability to identify and deal with separate audio and video objects in the frame, which allows separate elements to be compressed more efficiently and dealt with independently. User-controlled interactive sequences that include audio, video, text, 2D and 3D objects and animations are all part of the MPEG-4 framework.

MPEG-4 uses further coding tools with additional complexity to achieve higher compression factors than MPEG-2. In addition to more efficient coding of video, MPEG-4 moves closer to computer graphics applications. In more complex profiles, the MPEG-4 decoder effectively becomes a rendering processor and the compressed bitstream describes three-dimensional shapes and surface texture. MPEG-4 also provides Intellectual Property Management and Protection (IPMP) which provides the facility to use proprietary technologies to manage and protect content like digital rights management. Several new higher-efficiency video standards (newer than MPEG-2 Video) are included (an alternative to MPEG-2 Video), notably:

- **MPEG-4 Part 2:**

  Despite allowing much more efficient Compression than MPEG-2 (as used in DVDs and most DTV), it lacks the advanced technology for which AVC (Advanced Video Coding) is named, including CABAC, InLoop Deblocking, and Partitions. The best known implementations of MPEG-4 Part 2 are the DivX and XviD codecs, which began primarily as computer-bound formats that required a computer to watch but are now playable by a large number of standalone DVD players.

- **MPEG-4 Part 10:**

  MPEG-4 Part 10 (or Advanced Video Coding or H.264) may be used on HD DVD and Blu-ray discs, along with VC-1. As well MPEG-2. 10 was designed specifically with High Definition (HD) video for home theater applications in mind as one potential application. It also includes specifications for lower quality video for portable devices, allowing a single standard to be compatible across a wide range of different needs.

  Advanced Video Coding (AVC) is the newest entry in the series of international video coding standards. It is currently the most powerful and state-of-the-art standard, and was developed by a Joint Video Team (JVT) consisting of experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG) created in 2001. The ITU-T H.264 standard and the ISO/IEC MPEG-4 Part 10 standard (formally, ISO/IEC 14496-10) are technically identical. The final drafting work on the first version of the standard was completed in May 2003. As has been the case with past standards, its design provides the most current balance between the coding efficiency, implementation complexity, and cost based on state of VLSI design technology (CPUs, DSPs, ASICs, FPGAs, etc).

H.264/MPEG-4 Part 10 contains a number of new features that allow it to compress video much more effectively than older standards and to provide more flexibility for application to a wide variety of network environments.

The main goals of this standardization effort are to develop a simple and straightforward video coding design, with enhanced compression performance, and to provide a “network-friendly” video representation which addresses “conversational” (video telephony) and “non-conversational” (storage, broadcast or streaming) applications. H.264 has achieved a significant improvement in the rate-distortion efficiency – providing, typically, a factor of two in bitrate savings when compared with existing standards

It presents some new features like :

- Multi-picture inter-picture prediction
- Spatial prediction from the edges of neighboring blocks for "intra" coding, rather than the "DC"-only prediction found in MPEG-2 Part 2 and the transform coefficient prediction found in H.263v2 and MPEG-4 Part 2
- Lossless macroblock coding features
- Flexible interlaced-scan video coding features
- New transform design features
- An entropy coding design
- Loss resilience features

Note that MPEG-4 Part 10 defines a different format than MPEG-4 Part 2 and should not be confused with it. MPEG-4 Part 10 is commonly referred to as H.264 or AVC, and was jointly developed by ITU-T and MPEG.

- **MPEG-7 (Meta-Data)**

  Formally called the Multimedia Content Description Interface, MPEG-7 provides a tool set for completely describing multimedia content. MPEG-7 is designed to be generic and not targeted to a specific application.

  MPEG-7 is about describing multimedia objects and has nothing to do with compression. It provides a library of core description tools and an XML-based Description Definition Language (DDL) for extending the library with additional multimedia objects. Color, texture, shape and motion are examples of characteristics defined by MPEG-7.

ITU-T

The **Video Coding Experts Group** or **Visual Coding Experts Group** (**VCEG**) is the informal name of Question 6 (Video coding) of Working Party 3 (Media coding) of Study Group 16 (Multimedia coding, systems and applications) of the ITU-T.This organization

has standardized (and is responsible for the maintenance of) the following video compression formats and ancillary standards:

- **H.120:** the first digital video coding standard. v1 (1984) featured conditional replenishment, differential PCM, scalar quantization, variable-length coding and a switch for quincunx sampling. v2 (1988) added motion compensation and background prediction. This standard was little-used and no codecs exist.

- **H.261:** was the first practical digital video coding standard (late 1990). This design was a pioneering effort, and all subsequent international video coding standards have been based closely on its design.

- **H.262:** it is identical in content to the video part of the ISO/IEC MPEG-2 standard (ISO/IEC 13818-2). This standard was developed in a joint partnership between VCEG and MPEG, and thus it became published as a standard of both organizations. ITU-T Recommendation H.262 and ISO/IEC 13818-2 were developed and published as "common text" international standards. As a result, the two documents are completely identical in all aspects.

- **H.263:** was developed as an evolutionary improvement based on experience from H.261, and the MPEG-1 and MPEG-2 standards. Its first version was completed in 1995 and provided a suitable replacement for H.261 at all bitrates.

- **H.263v2:** also known as H.263+ or as the 1998 version of H.263, is the informal name of the second edition of the H.263 international video coding standard. It retains the entire technical content of the original version of the standard, but enhances H.263 capabilities by adding several annexes which substantially improve encoding efficiency and provide other capabilities (such as enhanced robustness against data loss in the transmission channel). The H.263+ project was completed in late 1997 or early 1998, and was then followed by an "H.263++" project that added a few more enhancements in late 2000.

- **H.264:** Will be discussed in the MPEG standardization section. (MPEG 4, part 10)

3.2. MVC (Multi-view video coding)

3.2.1. Introduction

Since our world is not a 2D world, 3D has always been an exciting target in visual representation. 3D experiences may be provided through multi-view video (MVV), notably: i) 3D video (also called stereo) which brings a depth impression of a scene, and ii) free viewpoint video (FVV) which allows an interactive selection of the viewpoint and direction within certain ranges. Of course, 3D experiences may require special 3D display technology but many new products have been announced and exhibited recently. In fact, there are new 3D display technologies with interesting features such as no glasses, multi-

persons displays, higher display resolutions, and also avoiding uneasy feelings (headaches, nausea, eye strain, etc.).

3D technology is relevant for broadcasting, teleconference, surveillance, interactive video, cinema, gaming and other immersive video applications. While there is already a MPEG-2 Video multi-view profile, the truth is that it passed almost unnoticed, especially due to its low compression efficiency in comparison with the simulcasting (or independent view coding) alternative. Multi-view video refers to a set of N temporal synchronized video streams coming from cameras that capture the same real world scene from different viewpoints. MVV provides the ability to change viewpoint freely when multiple views are available; it is also possible to render one view (real or virtual) to legacy 2D displays. Clearly, the most well known case is stereo video (N = 2), with each view derived for projection into one eye, in order to create the illusion of depth. Since 2001, MPEG has been studying 3D based applications and their needs in terms of content representation. While many good tools developed both by the Video and SNHC MPEG subgroups are already available in MPEG-4 Visual, it was recognized that the MVV case was becoming relevant enough to develop a novel, more efficient standard. Following the same process as for SVC, an AVC backward compatibility requirement was identified which again led to a Multi-view Video Coding (MVC) standard built on the top of the AVC standard, again in the context of the JVT.

Multi-view video coding (MVC) is one of the most interesting technologies for future multimedia applications such as free-viewpoint video or photorealistic rendering of 3D scenes. Multi-view video will be used to drive next-generation 3D displays. These next-generation displays enable viewing of high resolution stereoscopic images from arbitrary positions and without need for glasses. The multi-view video provides the 3D display with view-dependent pixels that render a different color to the observer based on viewing angle.

3.2.2. Coding

For 3D systems based on multi-view video to become feasible, we need simultaneous recording of a moving scene with several cameras that will generate huge amounts of data which makes efficient compression necessary. The most straightforward solution to MVC is to independently code each video of each camera with a conventional video codec, e.g. H.264. As the efficiency of this so-called simulcast coding is not optimal, much effort has been spent to find better solutions. In many publications, it could be shown that there exists significant correlation across the different views at the same or similar time instances[59, 60, 61].

In order to achieve efficient compression results, exploitation of these inter-view dependencies is indispensable. Most multi-view codecs that have been published recently use some kind of inter-view prediction and show superior performance compared to simulcast coding. The new MVC reference software used for MPEG evaluations is an example for this strategy.

Compression is certainly needed; since the increase in the amount of data and storage requirements, new compression techniques that exploit not only the temporal correlation in the video, but also the inter view correlation between the videos are being extensively studied in the context of H.264[4].

As mentioned, a simple for coding multi-view video is to encode each view independently using a state-of-art video codec such as H.264/AVC[4]. The advantage of this approach is that it could be achieved with current standards and existing hardware. However, it does not exploit the redundancy across views, thereby not realizing the most efficient representation in compressed form.

The basic idea employed in all related works on efficient multi-view video coding is to exploit spatial and temporal redundancy for compression, since all cameras capture the same scene from different viewpoints, inter view redundancy is present. A sample prediction structure is shown in **Fig. 3.1.** Pictures are not only predicted from temporal neighbors, but also from spatial neighbors in adjacent views.

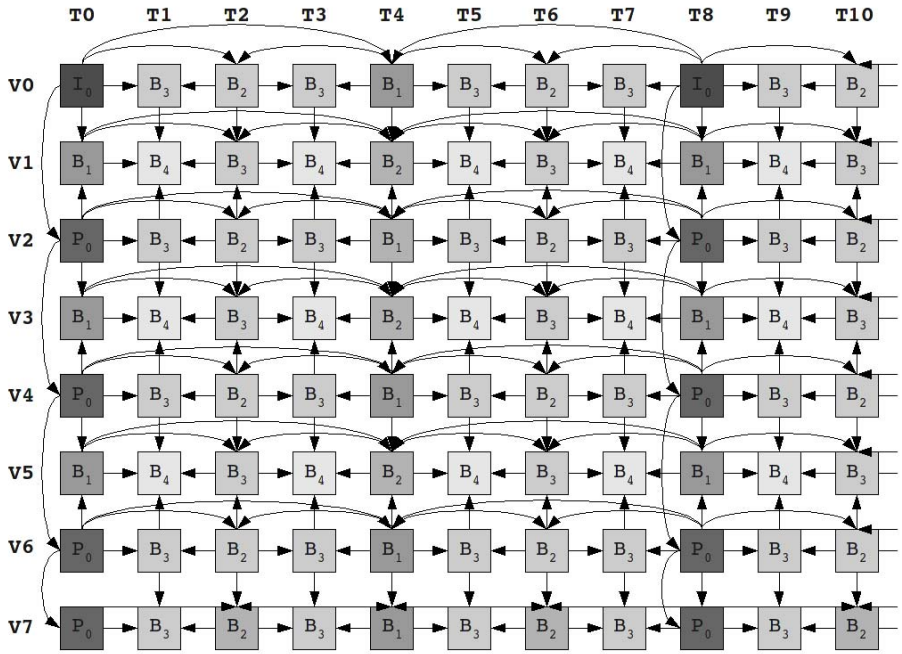


Fig. 3.1: Sample prediction structure for multi-view video coding that exploits both temporal and spatial redundancy

It has been shown that coding multi-view video in this way does give significantly better results compared to the simple H.264 simulcast solution [5]. Improvements of more than 2 dB have been reported for the same bitrate, and subjective testing has indicated that the same quality could be achieved with approximately half the bitrate for a number of test sequences.

There are many variations on the prediction structure considering both temporal and spatial dependencies. The structure not only affects coding performance, but has notable impact on delay, memory requirements and random access. It has been confirmed that the majority of gains are obtained using interview prediction at anchor positions, i.e., set of pictures that have no temporal prediction, only spatial prediction such as T0 in **Fig. 3.1** .

Besides the exploitation of the temporal and spatial redundancy within each view, redundancy can also be exploited across the different views to achieve higher coding gains. The current AVC multi-view extension, MVC, does not require any changes to lower-level AVC syntax, and thus it is very compatible with single-layer AVC hardware, since no new coding tools have already been accepted in MVC. Inter-view prediction is enabled just with the flexible design of decoded reference picture management which allows decoded pictures from other views to be inserted and removed from the reference picture buffer. Small changes to the high-level AVC syntax have been introduced, e.g. to specify view dependency. Some of the tools which are still under consideration are illumination compensation (to incorporate illumination changes into the motion compensation process), adaptive reference filtering (to compensate for the focus mismatches between views), motion skip mode (to infer motion information from the corresponding block in neighboring view), and view synthesis prediction (to generate synthesized views from neighboring views using estimated depth, then use synthesized view for prediction). All these tools would require changes to slice/macroblock level AVC syntax. Additional 10-15% gains are expected depending on the content. It is still not clear if this amount of gains will be enough to make worthwhile to include these tools in the MVC standard. Since MVV video codecs are required to store/transmit huge amounts of data, the standardization goal is to reach 50% bitrate savings over independent coding of views (simulcasting) with same quality by defining another extension to the AVC standard.

3.3. MVD (MVC + Depth)

The MVD format consists of several camera sequences of color texture images and associated per sample depth images or depth maps as illustrated in **Fig. 3.2.** Depth images are a 2D representation of the 3D surface of the scene. Therefore the depth range is restricted to a range in between two extremes Z_{near} and Z_{far}, indicating the minimum and maximum distance of the corresponding 3D point from the camera respectively. By quantizing the values in this range, the depth image in **Fig. 3.2** is specified which will result to a grey scale image. A sequence of such depth images can be converted into a YUV 4:0:0 format video signal and compressed by any state-of-the-art video codec.

Since depth images represent the scene depth, their characteristics differ from texture images. Encoding depth images with video codecs that are highly optimized to the statistical properties and human perception of color or texture video sequences, might be efficient but results in disturbing artifacts. Therefore novel algorithms for depth image compression are developed, that are adapted to their special characteristics, namely smooth regions delineated by sharp edges.

(a) **(b)**

Fig. 3.2: Example for the MVD format with texture image **(b)** and corresponding depth-image **(a)**.

We should be aware of the fully effectiveness of traditional image compression methods in providing maximum perceived visual quality, and as well know that depth-maps are not directly viewed. But depth images will be used to render some virtual images. It means they will be used to create some view images which their quality is of importance to us, not the quality of decoded depth maps themselves. Knowing this, we will result that having the highest perceived visual quality is not the most important issue. And especially for depth images preserving the contours is one of the fundamental tasks that should be considered in all the phases of the encoding/decoding process.

Bearing in mind the up-mentioned resolution, we were urged to try to exploit a segmentation based approach to encode/decode the depth video. As a result, the image had to be divided to different regions and contours and partitions had to be encoded separately.

Using this approach on two sequences of “Ballet ” and “Breakdancers “, different combinations of the encoding methods will be tested and the best approach regarding both a low bitrate and a high quality will be chosen and in order to evaluate the properties of segmentation based method for depth coding algorithm, the coding results are compared to H.264 as a reference.

4. Segmentation

4.1. Introduction

Image segmentation is a long standing problem in computer vision and is a fundamental step in many areas of this aspect including stereo vision and object recognition. In computer vision, segmentation refers to the process of partitioning a digital image into multiple segments (sets of pixels). The segmentation is based on measurements taken from the image and might be grey level, colour, texture, depth, motion, etc. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful with respect to a particular application and easier to analyze. It provides additional information about the contents of an image by identifying edges and regions of similar color and texture, while simplifying the image from thousands of pixels to less than a few hundred segments. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

The result of image segmentation is a set of partitions that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

Additionally, image segmentation has applications separate from computer vision; it is frequently used to aid in isolating or removing specific portions of an image. From those applications we can name Content Based Image Retrieval (CBIR), object recognition, matching of stereo pairs for 3-D reconstruction, navigation and artificial expert medical. Although a seemingly simple first step in high level computer vision tasks, there are many challenges to an ideal image segmentation. There are many possible correct segmentations, and the one we are looking for depends on the application. For example, in object recognition, the goal is often to find segments that describe objects. However, the idea of “objects” is itself vague. Most of the time objects can be subdivided into smaller objects. For example, a tree can further be divided into a trunk and its leaves. To clearly define an ideal segmentation we must decide on the level of detail that we want. Another application of segmentation is as a preliminary step to stereo vision. Here, we want to group together pixels that are the same distance from the camera, and thus an ideal segmentation has relatively small segments where many put together define an object. Even if we can formulate a definition for an ideal segmentation, there are many difficulties to overcome. The information we know directly about an image is the intensity/color and location of each pixel, and therefore these two variables are the central source of data for any segmentation. However, color and intensity can vary significantly over a single object. Also, shadows, reflections, and textures add sharp color contrasts to the same surface. Additionally, the image can be distorted or blurry. Edges that look clearly defined from far away are often blurry and hard to distinguish at the pixel level. Similarly, an area that looks like a solid color at a distance may be seen as mix of many

colors at the pixel level and may be interpreted as a number of segments. Shortly we can say a segmentation of an image is a partition of the image that reveals some of its content.

The confusion in defining image segmentation (and also perceptual grouping) simply reflects the following two facts:

1) It is difficult to model all types of stochastic patterns in generic vision.

Real world images consist of multiple layers of stochastic processes, such as texture, texton, stochastic point, line, curve, graph, region, and object processes, which generate images through spatial organizations. Thus an appropriate formulation should be image decomposition or parsing, which decomposes an image into its natural constituents as various stochastic processes.

This subsumes image segmentation as region process, and naturally integrates object recognition and perceptual organization. The latter deal with point, line, curve, and object processes. Implicit in this formulation is the notion of generative models for image interpretation in contrast to classification and discrimination methods. A segmentation algorithm must be general enough to handle many families of image models in a principled way.

2) Image segmentation is a computational process and should NOT be treated as a task.

Real images are intrinsically ambiguous, and our perception changes dynamically, even in a very short time duration, depending on our attention. Generally speaking, the more one looks at an image, the more one sees. It seems narrow-minded to think that a segmentation algorithm just outputs one final result. Instead a segmentation algorithm should realize the intrinsic ambiguities characterized, say, in a Bayesian posterior probability, and outputs multiple distinct solutions dynamically and endlessly so that these solutions, as samples, “best preserve” the posterior density. Then we need a mathematical principle for pruning and selecting the outputs.

4.2. Segmentation techniques

Image segmentation is an ill-posed problem [6]. For a given image, different partitions can be obtained depending on the homogeneity criteria. So, the segmentation approach has to be selected depending on the final specific application where it is to be used. In the framework of segmentation-based video coding the goal is the quality of the final image representation as well as its coding cost. This means that, to obtain a suitable partition for coding purposes, the special characteristics of the coding techniques that are used should be taken into account. Thus the segmentation process not only has to describe the information in the scene, but also has to lead to a low cost representation.

Three major steps can be outlined in a segmentation process [7]: **simplification**, **feature extraction** and **decision**. The simplification step aims at reducing the complexity of the process by removing irrelevant information. The feature extraction step involves the

choice of the type of homogeneity to be considered (this is, the choice of the feature space) and the process of estimation of the feature (or features) that will be analyzed looking for homogeneity. This analysis of the data in the feature space is performed later in the decision step, where the position of the boundaries in the decision space is established. These boundaries separate data areas that contain elements sharing the same characteristics in the feature space.

Then, a classification is done according to the decision viewpoint, with two main categories: **transition based** and **homogeneity based** segmentation techniques. Transition based techniques intend to estimate the position of discontinuities (that are evaluated in the feature space) in the decision space. To do this, a pre-processing step that defines transition zones is followed by a process of reduction of uncertainty in these zones that provides the final region borders.

The second category is formed by techniques that work by looking, in the decision step, for regions where elements are homogeneous with respect to the feature space. In the literature one can find two main types of homogeneity criteria: deterministic and probabilistic. In both cases, a model is given for each region. In deterministic models, the decision on to which region belongs each pixel is taken by computing the distance between the actual pixel value and the different model estimations (one per region) for this pixel. In probabilistic schemes, regions are modeled by a random field and also a posteriori likelihood of the data to be a sample of this random field is computed.

In spatial segmentation, features are estimated from spatial information (color or gray level). Features such as size, contrast, spectral content or dynamic characterize visually relevant objects. For each feature, a different homogeneity criterion can be applied.

Coding systems that only utilize spatial segmentation face the problem of exploiting temporal redundancy. Although some different ways to solve this problem exist [8, 9, 10, 11, 12, 13], perhaps the most useful method uses a time recursive segmentation of the sequence [14, 15, 16]. In this case the sequence is regarded as a 3D signal and the segmentation is carried out over a window of several frames that is shifted along the sequence as the processing of the sequence advances along the time axis. In this approach, the partitions of former frames are used to initialize the segmentation procedure of the new frames.

There are also segmentation techniques that directly combine both spatial and temporal homogeneity in the same partition[17]. This spatio-temporal approach seems to provide the most promising results. As there is a strong interaction between the estimation of the moving objects boundaries and their motions, it makes sense to use spatial and motion information in the segmentation step. This is the case of the joint estimation of motion and segmentation. The segmentation is expressed as a relaxation problem based on a Markov Random Field (MRF) modeling and a Bayesian criterion [18]. In this context, the spatio temporal segmentation and the motion are simultaneously estimated [19].

Only spatial and spatio-temporal homogeneity based segmentation methods are better suited for coding purposes because of its superior accuracy on the position of boundaries. In general, spatial and spatio-temporal segmentation techniques use homogeneity-based methods in the decision step. This is because the transition-based approach presents problems when used in spatial segmentation (lack of robustness, lack of accuracy at the transitions). However, some spatial transition-based segmentation techniques have been presented [20, 21], these methods are mostly intended for video analysis and indexing and are not well suited for coding purposes.

Even though spatio-temporal segmentation provides better result in video coding applications, purely spatial segmentation techniques are important by several reasons [22]. Any coding technique needs an intra-frame coding mode, and intra-frame segmentations can utilize only spatial information. Moreover, spatial segmentation is useful even in systems that use motion-based segmentation because there are always regions in which the motion approach will fail. This is the case, for instance of new objects appearing in the scene.

4.3. BPT method

Shortly, we can say Binary Partition Trees method is a region-oriented image representation[24]. BPT concentrates on a compact and structured representation of a set of meaningful regions that can be extracted from an image. It offers a multi-scale representation of the image and define a translation invariant 2-connectivity rule among regions. This representation can be used for a large number of processing goals such as filtering, segmentation, information retrieval and visual browsing. Furthermore, the processing of the tree representation leads to very efficient algorithms. Finally, for some applications, it may be interesting to compute the Binary Partition Tree once and to store it for subsequent use for various applications.

In a region-based image representation, objects in the scene are obtained by the union of regions in an initial partition. Since an arbitrary partition may contain about a hundred of regions, the number of different possible unions among these regions can be large. Actually, a region-based representation implies a compromise between accuracy (related to the number of regions in the partition) and processing efficiency (related to the number of unions of regions to be analyzed).

From this set of regions at various resolution levels, an application dependent algorithm can select the most convenient one(s) for its concrete application.

There exist different approaches to hierarchical region-based image representation, usually related to tree data structures. In these structures, tree nodes are associated with regions in the image whereas tree links represent the inclusion relation.

Especially in the Binary Partition Tree (BPT) [23] reflects the similarity between neighboring regions. It proposes a hierarchy of regions created by a merging algorithm that can make use of any similarity measure. Starting from a given partition (that can be as fine as assuming that each pixel or each flat zone is a region), the region merging

algorithm proceeds iteratively by 1) computing a similarity measure (merging criterion) for all pair of neighbor regions, 2) selecting the most similar pair of regions and merging them into a new region, and 3) updating the neighborhood and the similarity measures. The algorithm iterates steps 2) and 3) until all regions are merged into a single region.

The BPT stores the whole merging sequence from an initial partition to the one-single region representation. The leaves in the tree are the regions in the initial partition. Merging is represented by creating a parent node (the new region resulting from the merging) and linking it to its two children nodes (the pair of regions that are merged). An example of BPT is shown in **Fig. 4.1** [24]. As with the other tree representations, the nodes of the tree represent regions and the links of the inclusion relationship.

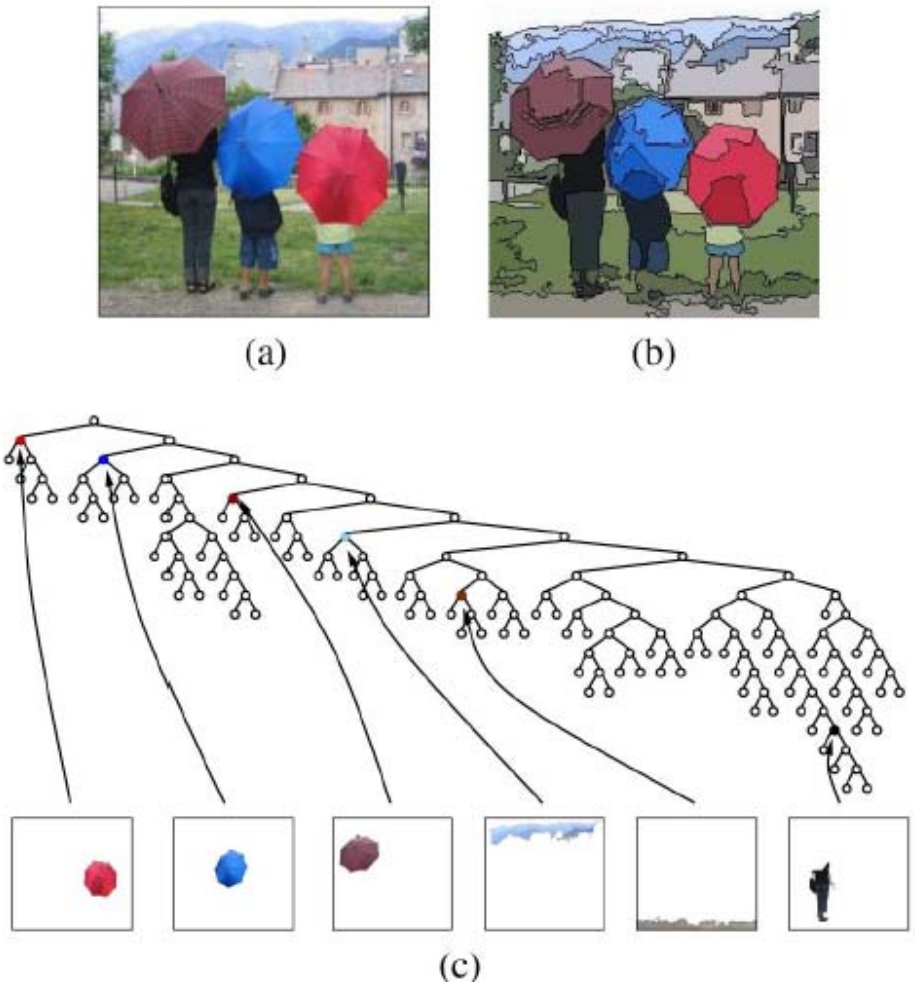


Fig. 4.1: Example of BPT and illustration of its ability to represent objects in the scene: **(a)** Original image, **(b)** Initial partition where each region has been filled with its mean color, **(c)** Binary Partition Tree and examples of nodes representing objects in the scene

The BPT represents a set of regions at different scales of resolution and its nodes provide good estimates of the objects in the scene. Classical object detection algorithms scan the image at numerous positions and scales looking for the possible representations of the object in the scene. Using the BPT representation, the image has to be analyzed only at the positions and scales that are proposed by the BPT nodes. Therefore, the BPT can be considered as a means of reducing the search space in object detection tasks.

5. Segmentation based coding

5.1. Introduction

Coding of arbitrarily shaped image segments is an important tool to achieve object-based coding, which is becoming more and more popular in todays multimedia applications. Segmentation-based coding methods (SBCS) [25, 26, 27, 28, 29, 30] rely on an adaptive partition of the image, that can be described as a set of textured regions with arbitrarily shaped contours. In this case unlike block based methods, the size and shape of each partition is not fixed and dependes on the specific image and its content. Texture and contour information can be adaptively coded using an appropriate representation model for each image segment. So, region based image coding methods divide an image into homogeneous regions with coding of contour and texture information done separately [62, 63]. These techniques have the advantage of adaptive allocation of bit budget to code the texture according to the activity level of each segment.

The goal is efficient sequence coding, i.e., maximal quality of the decoded images for a given coding cost (or the dual problem, minimal coding cost for a given image quality). The segmentation has to produce a set of regions (partitions) suitable for coding purposes. This can be done by ensuring that the regions are homogeneous in some sense (e.g. gray level,color or motion). Due to this homogeneity, the information of each region can be separately coded in an efficient manner, while there is not that much change inside each partition.

A common problem found in segmentation-based coding systems is the high cost of coding the contour information in comparison to texture coding. Accurate contour representation is very important for the quality of the decoded images because the errors that appear in the decoded images have an important visual effect. This leads to the use of lossless or near-lossless coding techniques. However, a partition of the image with many regions will produce highly homogeneous regions, with an easily codable texture which will decrease the amount of data that should be sent for texture coding; but the cost of contours will be very high. On the contrary, a partition with few regions will lower the cost of contours but the regions can be rather inhomogeneous and a lot more coefficients are needed to encode each partition in an efficient and acceptable way. So there is a trade-off between these two options.

In this thesis different combinations of number of regions and number of coefficients related to each region are practiced to make the best possible ratio between these two, in order to find the optimized possible quality with a specified cost. This will be done by choosing a fix PSNR and changing the number of regions along with cost of encoding of each region to gain different bitrates. As well the dual problem can be implemented and while having more or less the same bitrate, changing the quality and comparing the achieved results to find the best trade off

.

5.2. Coding of contour and texture

Once the segmentation step has been performed, the resulting partition (shape and localization of the regions) and the texture (gray level or color information) of the resulting regions have to be encoded and transmitted[56]. Usually, the process is to encode first the contour and later the texture, in a closed loop approach. This way, errors due to lossy contour encoding are avoided.

Two main differences can be noticed with respect to block-based coding systems. The first one is that in block-based systems there is no need of the partition coding step because it is fixed a priori (rectangular blocks) while in segmentation based method the shape of contours is arbitrary and depends on the content of the image. The second difference is that the arbitrarily-shaped regions that appear in SBCS make possible a better adaption to the local image characteristics but also increase the complexity of the texture coding methods with respect to the block-based techniques.

The regions resulting after a segmentation process are statistically quasi-stationary and should therefore enable higher data compression ratios [31]. However, traditional blockbased texture coding techniques can not exploit efficiently this quasi-stationarity because they perform poorly on border blocks. In this case, the overhead imposed by the need to code the contour information, can result in a performance loss.

There are many texture and contour coding techniques that can be used in this kind of systems. Some of them are an evolution of existing block-based techniques, with modifications to deal with blocks located at region borders, while others are designed specially for SBCS.

We can cite for example, Shape-Adaptive DCT [32, 33], Shape-Adaptive Wavelet [34] and padding methods [35] as examples of the first category, while the generalized orthogonal transform method in [36] is on the second category.

Due to non-stationarity behavior of images, some texture coding techniques may be appropriate in some regions, whereas they may be particularly inefficient in some others. But still all the above commented techniques are adequate because they can be used in a scalable coding and the trade-off between coding cost and quality can be chosen for rate-control purposes.

Partition coding is necessary on SBCS to inform the receiver of the shape (contour) and spatial/temporal localization of the regions. Contour coding can be lossless or lossy. Because the human visual system is very sensitive to errors in the localization of the contours of the regions, lossless or near-lossless approaches are popular. However, these methods are expensive fron the bitrate point of view.

In the literature it is possible to find a wide range of contour coding techniques. Among the lossless techniques, can be pointed out Chain Code (CC) [37, 68], Morphological Skeleton [38, 39], transition points [40] and Quadtree decomposition (a particular case of

Skeleton [41]). Multigrid Chain Code (MGCC) [42, 43] is an example of a near-lossless technique (it is an extension of Chain Code). Finally, some examples of lossless techniques are Fourier descriptors [44, 45], and polygonal [46] and Spline [47] approximations. Note that partition coding has to deal not only with the contour of the regions, but also with the temporal evolution along the video sequence. This is important because it will be the basis for an efficient inter-frame mode, thus helping to effectively remove the temporal partition redundancy.

In this thesis for contour coding method Chain Code has been selected as a lossless method to preserve the contours completely and making it possible to compare the results and see the maximum quality that can be achieved. As well for texture coding different orders of the Orthobasis method has been used. So, considering the possibility of choosing different number of regions and also different orders of texture coding, it would be possible to find the best mixture for the specific application of depth coding.

5.2.1. Chain code contour coding

Chain Code techniques [37, 48] 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[56] 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 neighbors, encode the movements from one contour point to another, starting from a given initial point (or various), until the complete contour is tracked.

The contour of a given partition can be represented using a hexagonal contour grid [49]. 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 first example of **Fig. 5.1**, where circular and line segment elements represent sites from the partition and contour grids, respectively.

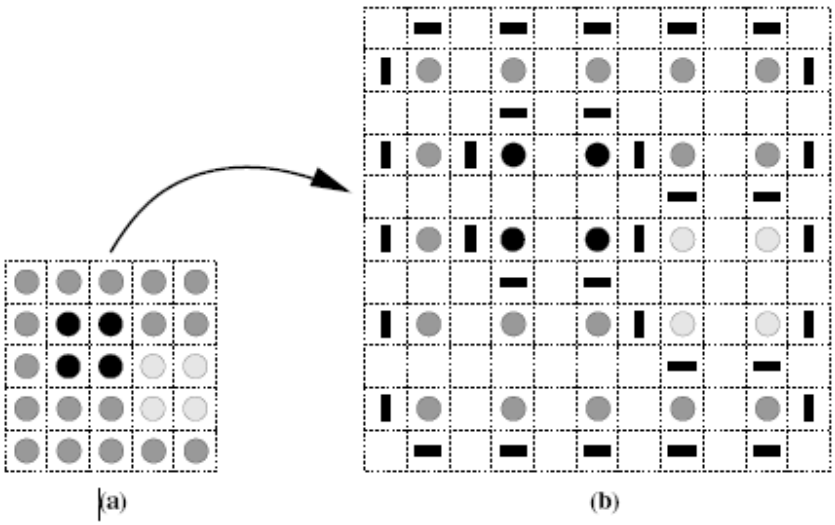


Fig. 5.1: Relationship between partition and contour grid sites[56]

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 is jointly coded by introducing the location information in the chain code itself. The neighborhood system is 6-connected and, therefore, there are 6 basic movements. Nevertheless, as a derivative chain code is used [37], the amount of possible movements reduces to 3: turn right, turn left and straight ahead (r,l,s) (See **Fig. 5.2**).

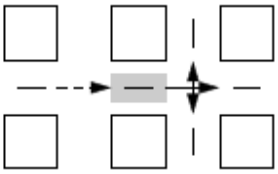


Fig. 5.2: Movements in a hexagonal grid for DCC[56]

The intersections of contours are encoded by using triple points. A triple point is a site in the contour grid that has at least three active neighbor contour sites (See **Fig. 5.3**) where at least three contours will meet each other in one point. Triple points can be used to locate the initial point of a new contour segment. Triple points are encoded by introducing a new symbol, M, in the chain itself. Only a subset of the triple points are actually necessary to describe the whole set of contours. These triple points are called active triple points and only active triple points are marked in the chain by the symbol M while the rest of the triple points will be recognized using only active triple points.

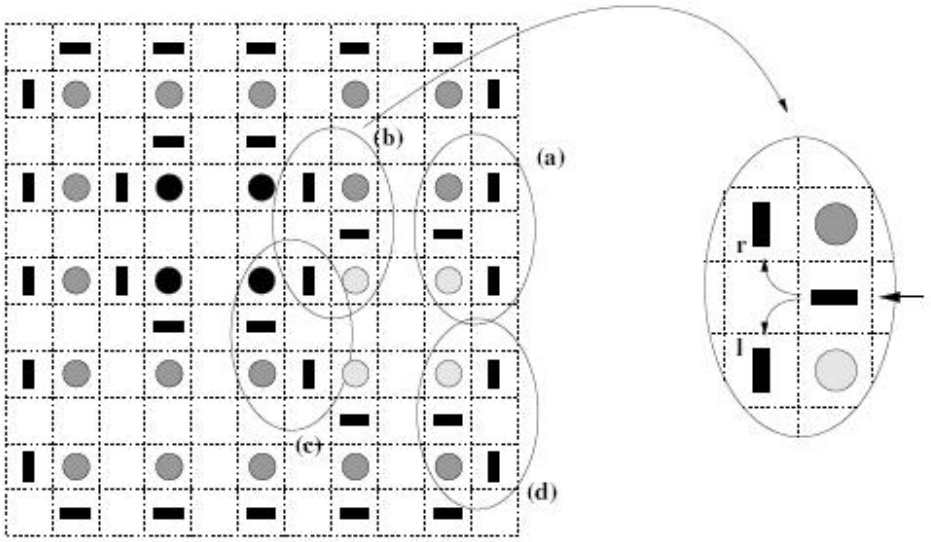


Fig. 5.3: Example of triple points. The ellipses mark the triple points in the example partition. The right figure shows a detail of a triple point[56]

Regions that have no contact with other regions do not produce any triple point and have to be directly addressed by the coordinates of their initial points.

In the inter-frame case, differential coding of the contours is used. That is, as the decoder already knows the contours that correspond to the compensated partition, only the new contours have to be sent. The contour segments are encoded by sending the initial point (always located on already known contours) and the set of movements describing the segment. Note that it is not necessary to send the ending point. The decoder can detect an end of segment when an already known contour is reached by the new contour.

5.2.2. Orthobasis texture coding

The boundaries are approximated by contour coding methods , and the remaining information to be coded consists of the inside of the regions, that is, their texture. 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 degree 0–3 [64], [65]. Gilge et al. [36] proposed a more accurate method in which a generalized orthogonal 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 [31] 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. Then, low complexity shape-adaptive DCT algorithms have been introduced to efficiently transform the image blocks that contain object boundaries [65, 66]. These

algorithms first divide an image into blocks and then code the boundary blocks using the shape-adaptive 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 [67, 34], which was implemented by Barnard [67] using a dyadic wavelet transform. 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.

In this thesis Orthobasis texture coding is used. This method operates segment by segment and needs the shape information which by an approximative contour description can be reconstructed at the decoder site. Orthogonalization schemes can be used to obtain a set of basis functions which is orthogonal with respect to the shape of the region, resulting in a generalized region-oriented transform coder. In this case the transmission of the used basis functional set is not necessary, because if the receiver has the same basic knowledge, e.g. uses the same starting polynomials, the orthogonal basis functional set can be re-generated at the receiver. The coded and transmitted segment shape contains the information how to derive the basis functions. Finally, the produced orthogonal basis functions are used for the segment content approximation coefficients and they determine the amount of each basis function needed for the content reconstruction, in other words, the segment is recovered by the weighted summation of the basis functional set with the approximation coefficients as the weighting factors. The algorithm then proceeds to the next segment. For further information the reader is refered to section 5 of reference [57].

6. Entropy Coding

6.1. Introduction

In information theory, **entropy** is a measure of the uncertainty associated with a random variable. The term by itself in this context usually refers to the **Shannon entropy** [50], which quantifies, in the sense of an expected value, the information contained in a message, usually in units such as bits. Equivalently, the Shannon entropy is a measure of the average information content that one is missing when one does not know the value of the random variable. Shannon entropy equation is shown in **Eq. 6.1**:

$$H(X) = -\sum_{i=0}^{N-1} p_i \log_2 p_i$$

Eq. 6.1: Shanon entropy equation, p is the probability of a given symbol

Shannon's entropy represents an absolute limit on the best possible lossless compression of any communication under certain constraints: treating messages to be encoded as a sequence of independent and identically-distributed random variables; Shannon's source coding theorem shows that, in the limit, the average length of the shortest possible representation to encode the messages in a given alphabet is their entropy divided by the logarithm of the number of symbols in the target alphabet.

The process of entropy coding (EC) can be split in two parts: modeling and coding. Modeling assigns probabilities to the symbols, and coding produces a bit sequence from these probabilities. As established in Shannon's source coding theorem, there is a relationship between the symbol probability and its corresponding bit sequence. In order to achieve a good compression rate, an exact probability estimation is needed. Since the model is responsible for the probability of each symbol, modeling is one of the most important tasks in data compression.

Entropy coding can be done with a coding scheme, which uses a discrete number of bits for each symbol, for example Huffman coding, or with a coding scheme, which uses a discrete number of bits for several symbols. In the last case we get arithmetic coding if the number of symbols is equal to the total number of symbols to encode.

The notion of compression systems captures the idea that data may be transformed into something which is encoded, then transmitted to a destination, and finally transformed back into the original data. Any data compression approach, whether employing arithmetic coding, Huffman codes, or any other coding technique, has a model which makes some assumptions about the data and the events encoded. And finally the goal is to compress data, which might either be stored on a computer-readable media or be sent

over some form of stream. This data could represent anything, reaching from simple text up to graphics, binary executable programs, etc.

6.2. Different methods

There are few main techniques for achieving entropy coding:

- **Huffman Coding**

  The basic idea of Huffman coding, which is one of the simplest variable length coding schemes, is to encode symbols that have higher probability to appear with fewer bits of output. Since each symbol can be represented by different amount of bits, it is important that no symbol code is a prefix of other symbol code. A code with this property is else called the **Prefix Code**.

  Static Huffman coding uses the symbols probabilities and then it makes the codes, which are of variable length. Note that one code can only have one symbol. The tree for static Huffman Coding is proved [51] to provide with an optimal Prefix Code for any given input stream. The main disadvantage of Static Huffman Coding is a need to care statistics information together with encoded stream of symbols. Adaptive Huffman Coding allows to avoid transmitting statistics data. Statistics information is gathered in the same pass and Huffman tree is updated accordingly. This way encoder and decoder both build Huffman tree while reading stream of data. Adaptive scheme is intended to avoid the input stream preprocessing during the encoding. Adaptive scheme gain us with two main benefits:

  - Decompression can be started before compression is finished.
  - No need to pass statistics data together with compressed data stream.

  The main disadvantages are:

  - Complexity: Statistics data must be calculated and interpreted on-the-fly.
  - Compression ratio is generally lower then in static coding. Although for short input sequences benefit of not sending statistics data can be more significant.

- **Run-length Coding (RLC)**

  RLC is a very simple form of data compression in which runs of data are stored as a single data value and count, rather than as the original run. This is most useful on data that contains many such runs: for example, relatively simple graphic images such as icons, line drawings, and animations. It is not recommended for use with files that don't have many runs as it could potentially double the file size. (very useful for binary data containing long runs of ones or zeros

- **Arithmetic Coding**

  This type of coding is a widely used method and relatively new variable length coding scheme that can combine the best features of Huffman and run-length coding, and also adapt to data with non-stationary statistics. Arithmetic coding completely bypasses the idea of replacing an input symbol with a specific code. Instead, it takes a stream of input symbols and replaces it with a single floating point output number. The only drawback is its computational complexity although compression tends to be better than what Huffman can achieve.

  Arithmetic coding encodes data (the data string) by creating a code string which represents a fractional value on the number line between 0 and 1. This single number can be uniquely decoded to create the exact stream of symbols. The coding algorithm is symbolwise recursive; i.e., it operates upon and encodes (decodes) one data symbol per iteration or recursion. So, the encoding and decoding algorithms perform arithmetic operations on the code string. On each recursion, the algorithm successively partitions an interval.

  In comparison to the well-known Huffman Coding algorithm, Arithmetic Coding overcomes the constraint that the symbol to be encoded has to be coded by a whole number of bits. This leads to higher efficiency and a better compression ratio in general. Indeed Arithmetic Coding can be proven to almost reach the best compression ratio possible, which is bounded by the entropy of the data being encoded. Though during encoding the algorithm generates one code for the whole input stream, this is done in a fully sequential manner, symbol after symbol.

- **Lempel-Ziv Compression algorithms**

  Lempel-Ziv Compression algorithms are easily divided in two main groups: LZ77 and LZ78. The bold difference between these two groups is that LZ77 do not need an explicit dictionary where LZ78 do need it.

  - **LZ77**

  The main idea of LZ77 is to find the longest match to the current part of the input stream in the already passed part of the input stream.

  - **LZ78**

  The main idea of LZ78 is to enhance dictionary with appearance of each unencoded symbol. Each time, if symbol found that can not be encoded, the dictionary is enhanced with new entity that is the last found entity plus one symbol.

In this thesis for entropy coding, an arithmetic coding will be used to decrease the bitrate to the lowest possible level.

7. Virtual view rendering

7.1. Introduction

The generation of virtual environments for telepresence systems, interactive viewing, and immersion in remote 3D-scenarios is an expanding and very promising research field. The challenge of this technology is to design systems capable of synthesizing views from any desired perspective using a set of real-scene perspectives. In such a system, two main parts can be distinguished: multi-view image-analysis and viewpoint synthesis.

The main advantage of MVD representations in contrast to MVV is that due to the availability of depth information, 3D rendering based applications like FVV can be realized. Multi-view video + depth data together (MVD) with camera geometry provide the possibility to synthesize or render arbitrary intermediate views from a 3D representation of the scene. Virtual view rendering uses pairs of neighboring original camera views to render arbitrary virtual views on a specified camera path between them. The relation between points in 3D scene space and the values in a depth image is defined by the projection matrix, allowing for projecting and unprojecting depth data. First the depth images are unprojected, resulting in a colored 3D particle cloud for each original camera. Next the projection matrix of a virtual camera is calculated from the two original cameras projection matrices by spherical linear interpolation (SLERP) and linear interpolation (LERP). These methods originate from computer graphics in the context of quaternion interpolation. Using the projection matrix of virtual cameras we have the ability to render the virtual view weighting according to the virtual camera's position relative to the original cameras.

The final goal is to render a sequence of virtual views in between each two original camera, and using this method we can implement the possibility of having MVD. (**Fig. 7.1**)

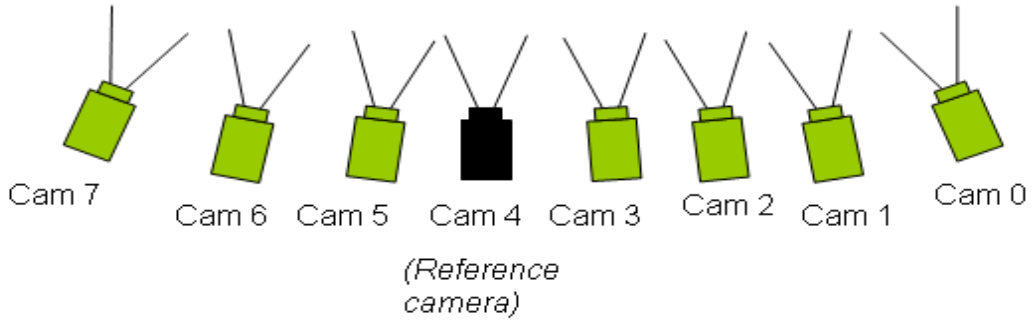


Fig. 7.1: The positioning of 8 cameras for Breakdancers sequence and respective reference camera

7.2. Virtual camera

For all the cameras aligned in this method, we are using the same coordinate system which is referred as a 3D world. In this general coordinate system all the positions of original and virtual cameras will be calculated based on a reference and afterwards all the calculation will be according to this global positioning. For example in the Breakdancers this reference (origin) point is the location of camera 4.

Each camera has three matrices which by, it is possible to calculate the projection matrix respected to that camera:

- **Rotation matrix:** also called a coordinate transformation matrix is a 3*3 matrix and by using a conversion, it will be converted to three Euler Angles regarding specific direction of each camera in the 3D world (which direction the camera is aiming for). Euler Angles consist of the *Pitch*, *Roll* and *Yaw* angles (or equivalently, the *Elevation*, *Bank*, and *Heading*). The formula is as below and it can be used to convert the angles to rotation matrix and vice versa:

$$M = \begin{bmatrix} \cos(\psi)\cos(\theta) & \sin(\psi)\cos(\theta) & -\sin(\theta) \\ \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \cos(\theta)\sin(\phi) \\ \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix}$$

  Using this formula we can find the three Euler angles of each real camera, and then the same Euler angles for each virtual camera should be calculated. To do this algebra while considering the distance distribution of virtual cameras in between each two original camera and using a simple linear and algebraic division, we can find the respective angles for the considered virtual camera. In this step using the reverse usage of the above formula we can create the rotation matrix of each virtual camera.

- **Translation Vector:** a 3 items vector which will represent the exact position of the camera in 3D world in the global coordinate system and in respect to the reference of this system (ex. camera 4 in the Breakdancers sequence). Which consist of X,Y and Z for each camera. So considering the distance from each one of original cameras and using a simple linear distance distribution in between those two, we can calculate the translation vector of each virtual camera.

- **Calibration matrix:** a unique 3*3 matrix which includes the intrinsic parameters of each camera which will be calculated based on the physical structure of each camera and is different from one camera to another one. The parameters which are used to create this matrix are as below :

  - Position of image center in the image **(**It is typically not at (width/2, height/2) of image**)**

- Focal length
- Different scaling factors for row pixels and column pixels
- Skew factor
- Lens distortion (pin-cushion effect)

This information is available based on each original camera's intrinsic characteristics (provided by producer company) and can not be calculated for the virtual cameras. As a result and knowing that this matrix can not be created or calculated for the virtual cameras, in this thesis, the calibration matrix related to closest original camera will be used for each one of the virtual cameras in order to have the best approximation.

Using these three steps, we have the rotation matrix, translation vector and calibration matrix of each virtual camera and using the below equation, we can calculate the projection matrix of that virtual camera:

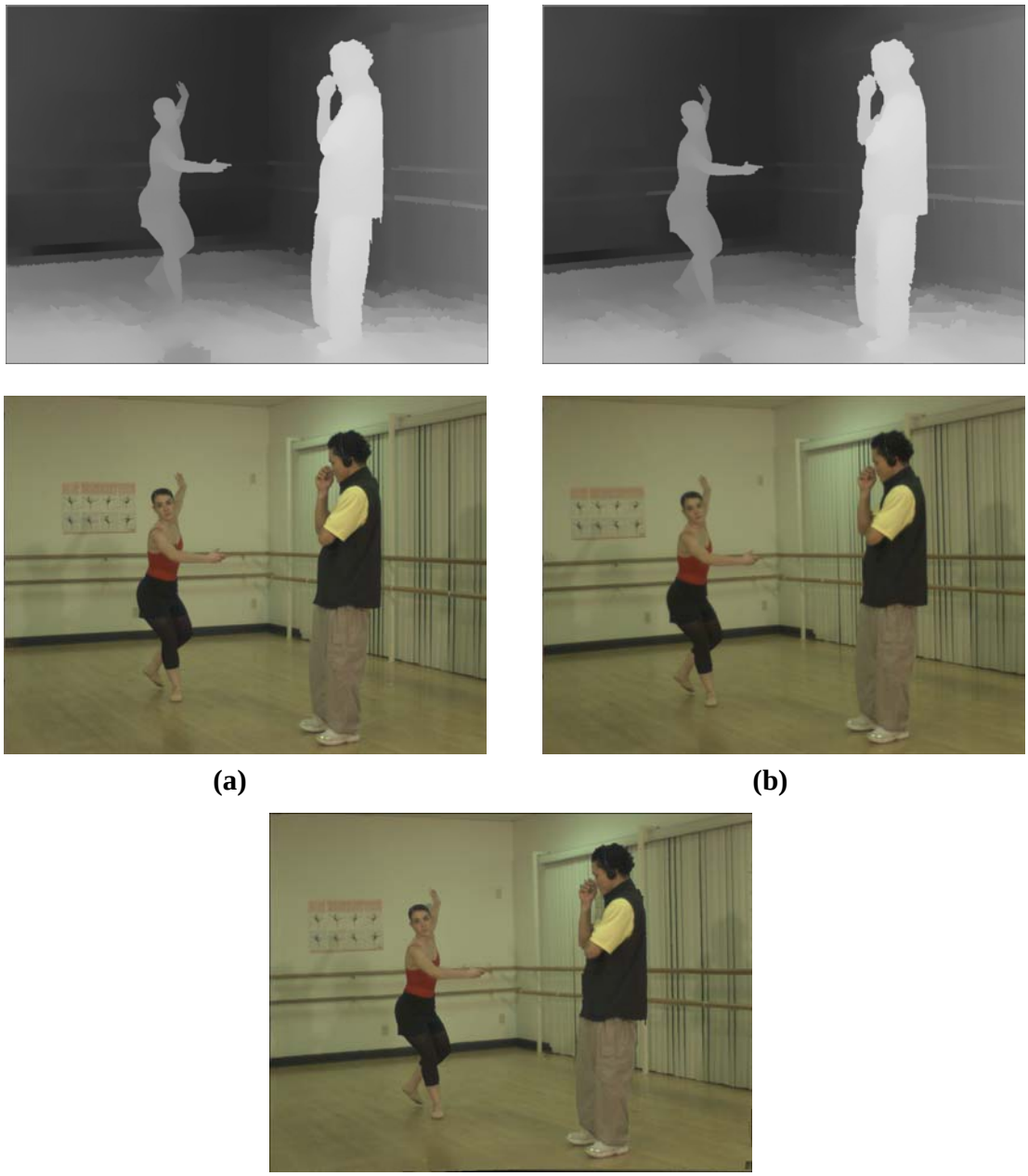
Projection matrix P = K * [Rot | Trans]

Where K = calibration matrix, Rot = Rotation matrix and Trans = Translation vector. This matrix will be used in the next section to create the virtual views.

7.3. Virtual view

The goal of this step is to create the virtual color view related to each virtual camera [55]. Here, we use for each pixel of the color image related to each original camera, the appropriate "depth" value in the same location at the depth image. At this point, we will use the projection matrix of each camera to project this point to the 3D world. Note that this 3D point is with respect to a reference camera which specifies a global coordinate system as mentioned in section 7.2. For the Breakdancers sequence we have the camera 4 as the reference.

Using this method, we can create 2 virtual views for each virtual camera. This will be done using the information of left and right original cameras separately. Joining these two images and trying to filter the blurred areas and filling the gaps, cracks and occlusions, we can achieve a good presentation of the middle virtual view. In **Fig. 7.2** we can see the depth maps, and color view of two consecutive cameras (camera 3 & 4) and the rendered virtual view in the middle of them.



(a) **(b)**

(c)

Fig. 7.2: Original view and depth image for **(a)** camera 3 and **(b)** camera 4 for ballet sequence, **(c)** Virtual view rendered in between them

7.4. PSNR calculation

Considering that depth images are not perceived directly and they will be used to create these rendered images which their quality is of importance to us, ordinary comparison of quality of depth images and calculating the traditional PSNR is not the best possible way to evaluate the exactness of our depth map coding method. **Fig. 7.3** is calculated based on

different quantization steps of H.264. For this specific figure 7 quantization steps of 46,44,40,36,32,28 and 24 are used which by increasing this factor, we will decrease the quality and bitrate. As it can be been below, the quality of decoded images for even quantization step equal to 44 is more than 34 db which is a completely satisfactory result.

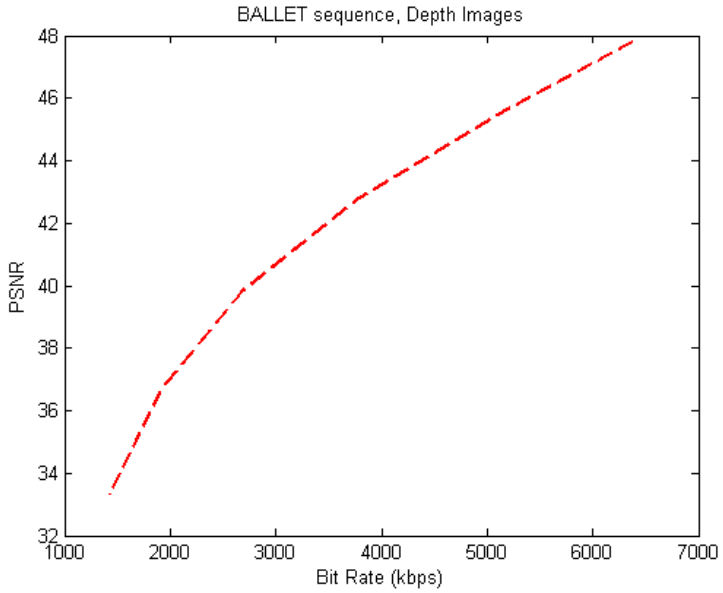


Fig. 7.3: Traditional PSNR for depth images

In this step and to calculate the PSNR, the rendering technique described in the previous section is first applied to uncompressed input data, resulting in a reference output image. In a second try the same virtual view is rendered by using compressed input data. The impact of coding artifacts on the quality of rendered views can now be analyzed by comparing the reference picture with the one rendered with compressed input data in terms of objective and subjective quality [58].

Using this method and considering that while going further from each original camera, the effect of depth quality will be more visible (considering that in farther points, error regarding depth maps will affect the rendered view more), we expect the PSNR to decrease while we approach to the middle position of each two original camera and to be maximum in the closest position to each original camera. (**Fig. 7.4**)

As well in the exact position of each original camera, the PSNR goes to infinite. This is because we are projecting a color image in a specific point using two different depth maps (original depth and compressed one) into the exact same place. So the difference of depth images won't be sensed while we are using the same position of rendering the virtual image as the one that we have taken the original view itself. Actually in the exact position of original cameras, if we use two completely different depth maps, we get the same infinite PSNR because we use the same original view and each pixel of that will be projected into the exact same place of original view.

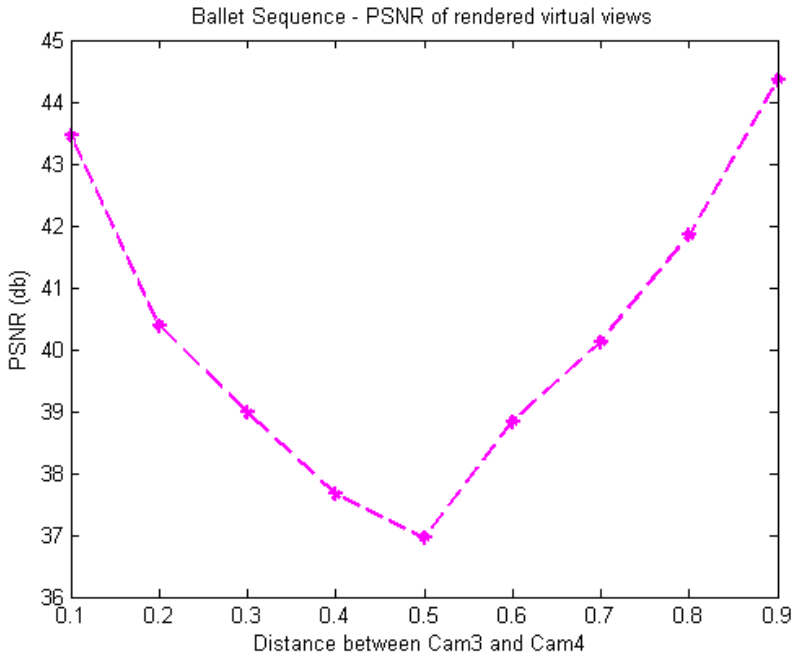


Fig. 7.4: PSNR calculation method for depth maps

Using this method, we can achieve a good evaluation of depth compression and considering that in real world, depth maps are used to create the virtual views (as we implemented in this quality comparison technique), we can be sure that this is the correct way to evaluate our segmentation based depth coding approach.

8. Proposed segmentation based coding method

8.1. Abstract

As explained in section 1, using chain code contour coding method along with Orthobasis partition coding is very costly and generates a high rate of bits which won't make it possible to compete with H.264 at all. Also having in mind that most of this bitrate is used by contour coding part of data, we thought of new possible approaches to make it possible to reduce this bitrate to a reasonable number that has the potential to compete our reference.

It is clear that the decoder is provided with the decoded view image and having only the depth images alone doesn't have any meaning. In other words, a receiver without having the decoded color images is incomplete and doesn't provide any perceivable information to the observer. This said along with thinking about a possible adjacent method of contour coding and considering the similarities between segmentation of depth image and color images, we thought about the possibility of exploiting this resemblance to do some part of the segmentation based on decoded color images and complete it with some information regarding original depth image segmentation. Using this new technique will produce some extra information which will be sent to the decoder along with the rest of the data for partition coding. So the quality of decoded depth images will increase considerably along with a reasonable raise in bitrate. The bitrate related to this extra information is not comparable to what we get using contour coding. Here we have a much lower bitrate which makes it possible to overcome H.264 with the same quality of rendered virtual images.

Using this combination method, the results outperform the outcome while only using view segmentation (based on the quality point of view). As well, while they are using view segmentation for some of the process, their quality is not as high as the results achieved by using only original depth segmentation (by exploiting contour coding which takes a high bitrate). The experimental results in section 9.5 will show a good view and understanding about this unique method of segmentation based coding of depth information.

In this thesis, for partition coding Orthobasis method will be used and considering that it is the same for all the experimental results, we don't focus on it. (This is one of the important aspects of future research that can be done). The most important part of this research deals with the way that segmentation will be made and how it is possible to decrease the bitrate while having a good final quality; this said, we try to tackle this problem based on the contour coding part rather than partition coding and as a result different approaches in this case are implemented and compared.

8.2. H.264 coding

As any other novel method, our new approach should be compared to the best already implemented and accessible algorithm that does the same process. In the case of video coding, the pioneer is H.264/AVC which our results will be compared to it as the reference. If we achieve a better bitrate while having the same quality, or the dual situation (gaining a better quality while having the same bitrate), we can consider our new method a success.

As reviewed in section 3.1, H.264 [52, 54] is the best video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group and represents a major step forward in the development of video coding standards. As already mentioned in section 3.1, it uses state-of-the-art coding tools and provides enhanced coding efficiency for a wide range of applications, including video telephony, video conferencing, TV, storage (DVD and/or hard disk based, especially high-definition DVD), streaming video, digital video authoring, digital cinema, and many others. In **Fig. 8.1** we can see the performance of this algorithm compared to other already implemented methods. As it can be obviously seen for the same bitrate the quality of decoded images, using H.264, is higher than the rest.

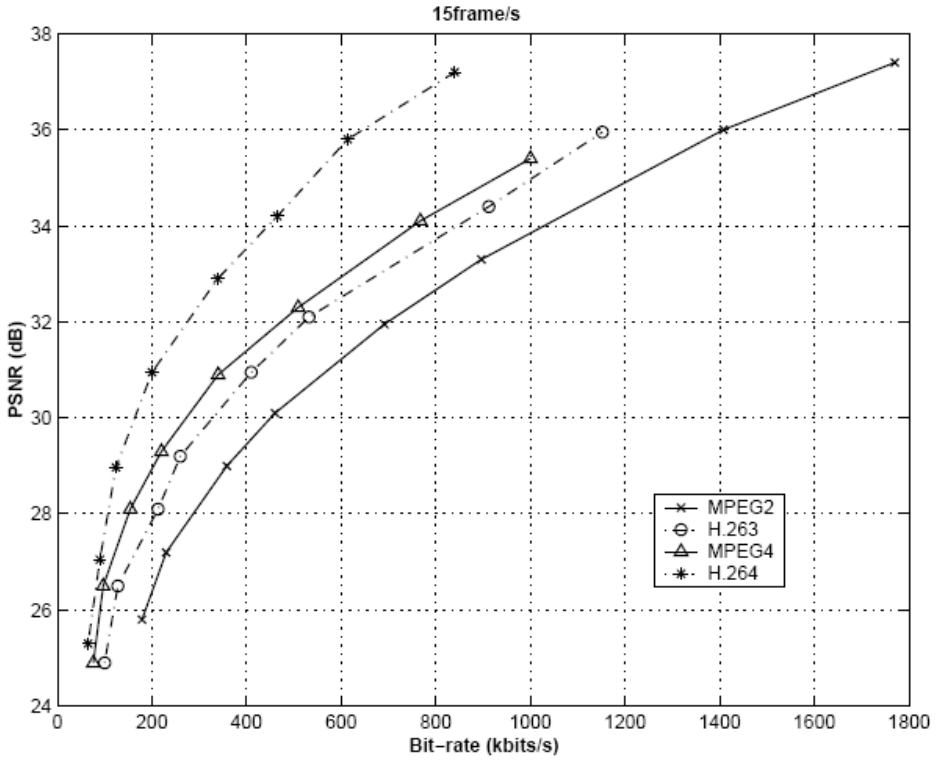


Fig. 8.1: Comparison between different coding methods already implemented [53]

The main goals of the H.264 standardization effort have been enhanced compression performance and provision of a “network-friendly” video representation addressing “conversational” (video telephony) and “nonconversational” (storage, broadcast, or streaming) applications. Also, H.264 has achieved a significant improvement in rate-distortion efficiency relative to existing standards.

At a basic overview level, the coding structure of this standard is similar to that of all prior major digital video standards. The architecture and the core building blocks of the encoder show that it is also based on motion-compensated DCT-like transform coding. Each picture is compressed by partitioning it as one or more slices; each slice consists of macroblocks, which are blocks of 16x16 luma samples with corresponding chroma samples. However, each macroblock is also divided into sub-macroblock partitions for motion-compensated prediction. The prediction partitions can have seven different sizes – 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. The spatial transform for the residual data is then either 8x8 or 4x4. In past major standards, the transform block size has always been 8x8, so the 4x4 block size provides an enhanced specificity in locating residual difference signals. The block size used for the spatial transform is always either the same or smaller than the block size used for prediction. The hierarchy of a video sequence, from sequence to samples is given by **Fig. 8.2**.

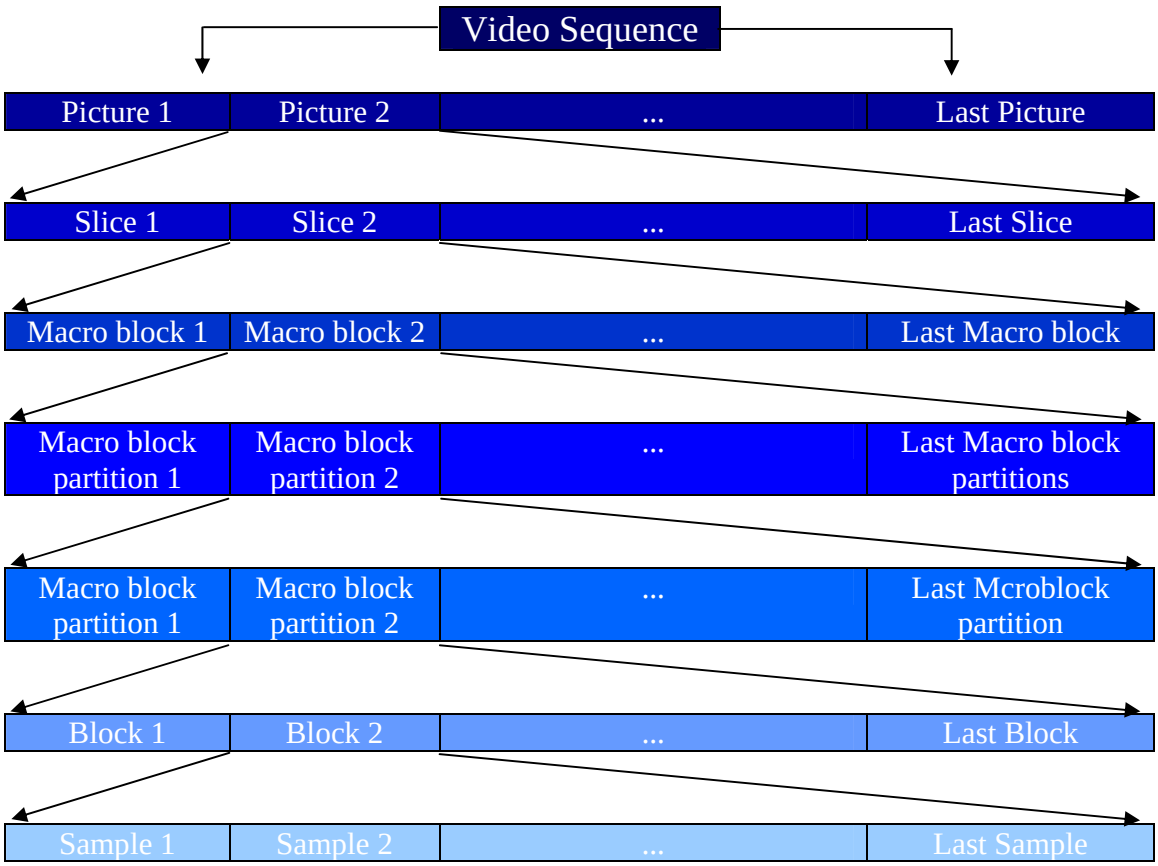


Fig 8.2.: The hierarchy of a video sequence, from sequence to samples

Considering that H.264 is the reference and it will be compared with our method, it is necessary to explain its coding tools more in detail. This known, we can mention that slices in a picture are compressed by using the following coding tools:

- "Intra" spatial (block based) prediction
  - Full-macroblock luma or chroma prediction – 4 modes (directions) for prediction
  - 8x8 (FRExt-only) or 4x4 luma prediction – 9 modes (directions) for prediction

- "Inter" temporal prediction – block based motion estimation and compensation
  - Multiple reference pictures
  - Reference B pictures
  - Arbitrary referencing order
  - Variable block sizes for motion compensation

- Seven block sizes: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4
  - 1/4-sample luma interpolation (1/4 or 1/8th-sample chroma interpolation)
  - Weighted prediction
  - Frame or Field based motion estimation for interlaced scanned video

- Interlaced coding features
  - Frame-field adaptation

- Picture Adaptive Frame Field (PicAFF)

- MacroBlock Adaptive Frame Field (MBAFF)
  - Field scan

- Lossless representation capability
  - Intra PCM raw sample-value macroblocks
  - Entropy-coded transform-bypass lossless macroblocks (FRExt-only)
  - 8x8 (FRExt-only) or 4x4 integer inverse transform (conceptually similar to the well-known DCT)
  - Residual color transform for efficient RGB coding without conversion loss or bit expansion (FRExt-only)

- Scalar quantization

- Encoder-specified perceptually weighted quantization scaling matrices (FRExt-only)

- Logarithmic control of quantization step size as a function of quantization control parameter

- Deblocking filter (within the motion compensation loop)

- Coefficient scanning, Zig-Zag

- Lossless Entropy coding
  - Universal Variable Length Coding (UVLC) using Exp-Golomb codes
  - Context Adaptive VLC (CAVLC)
  - Context-based Adaptive Binary Arithmetic Coding (CABAC)

- Error Resilience Tools
  - Flexible Macro block Ordering (FMO)
  - Arbitrary Slice Order (ASO)
  - Redundant Slices

- SP and SI synchronization pictures for streaming and other uses

For further information, the reader is referred to [52] to have an idea about different methods used in H.264 to increase the performance.

H.264 is completely based on block matching methods and doesn't apply segmentation encoding approaches at all; While in our new proposed method, the base of encoding is due to use of segmentation and uses the information of contours and partitions to recreate the final decoded image. This shows that our approach for depth coding is fundamentally different from that of our reference and is not just a modification of it.

8.3. View segmentation

Using segmentation and possible object detection methods, we can have a general understanding of what is going on in the image. While just using view segmentation, we can segment the image based on the different color levels of pixels and texture construction of the view. As shown in the **Fig. 8.3** (created using BPT method [24]) we can see that view segmentation is able to partition the image and show its content to an acceptable extent.

(a) **(b)**

Fig. 8.3: View segmentation gives a good understanding of the image
(a) Original image, **(b)** View segmentation

Considering that view and depth images are taken from the same scene and the objects, foreground and background are the same for both of them, we thought of the possibility of exploiting the view image segmentation for the decoding part of our segmentation based coding of depth information. Using this method, the whole created contours will not be the same as those of depth segmentation, but those contours that overlap between these two, are exactly the same. As well in this approach we can have a good partitioning of objects of the image.

Using view segmentation, we only need to send the information related to that of partition coding of the depth image, while the contours will be re created in decoder using the decoded color image. So while using the segmentation of decoded view in the receiver, we will be able to decrease the amount of being transmitted data to a considerable level. Having this in mind and considering that contour coding takes most of the transmitted data, this approach will be a very successful step from bitrate point of view.

While decreasing the bitrate and by achieving a good result, we are supposed to pay a cost. Here, quality of the final rendered virtual views will decrease comparing to the quality of virtual views rendered by the decoded depth images using lossless contour coding algorithms; and it is logical while we are using color views instead of the original depth maps to segment the image. A complete set of experimental results of this approach are available in section 9.5 which will show that this approach is applicable but has some restrictions that will result in low quality of final rendered virtual images. The exact effects on bitrate and PSNR will be pursued in the following sections.

8.4. Comparison with depth segmentation

As mentioned in the previous section, we considered the possibility of segmenting the decoded view images instead of sending the encoded data regarding the contours of the depth images which is very costly. Using this method, obviously the quality of final rendered virtual images will decrease, while we are not using the exact contours of the depth images in the receiver part in order to recreate the depth images.

There are different reasons that the segmentation applied on the decoded view images is not as exact as the segmentation of original depth images. Some of them deal with lack of segmentation which will not segment some parts of the image where they are supposed to be partitioned and some others will over segment the image. The latter will result in increasing the bitrate because regarding the increased number of segments, more coefficients should be sent for partition coding. Over all, we can divide the problems of view segmentation to two general below branches:

8.4.1. Over segmentation

Over segmentation deals with the possibility of segmenting some objects to two or more regions while considering the depth point of view, they should be all segmented as a homogeneous region. This happens while we have different colors for a region with the

same depth. For example if we have a person having a pants and coat with different colors; using view segmentation method, they will be segmented to two different regions. In the same case and using depth segmentation the whole person will probably be segmented as one object where it has a constant distance from camera for all of its pixels. This is shown in **Fig. 8.4**. In this figure you can see the original depth and relative segmentation and also the encoded color image and its segmentation. They both are partitioned to 20 regions using BPT method [24] and it is noticeable that the man standing at the right and also the dancing lady have almost one segment in depth but they are divided to different regions in view segmentation.

(a)

(b)

(c)

(d)

Fig. 8.4: (a) Original depth image, **(b)** Original encoded color image
(c) Segmented depth image, **(d)** Segmented color image
We can see that the same objects in **(d)** are over segmented in comparison to **(c)**

This problem results in increasing the bitrate but it doesn't affect the quality of the final rendered virtual images that much because we are just dividing a homogenous region to two or more partitions and relatively we send the information of each partition using texture coding methods separately; so the final depth image will be created almost the same as the original one while bitrate has increased.

Having this in mind, if the final bitrate is not that restricted, this problem may not cause such a big problem in the quality of final rendered images. But still finding new methods and solutions to decrease the number of regions to that of the original depth segmentation would be the ideal case and is of interest. This approach can be implemented in future possible research in this field.

8.4.2. Under segmentation

Another problem deals with some parts that are supposed to be partitioned from depth point of view, but using view segmentation they are not segmented. This problem results in a less acceptable final virtual rendered image, which some parts are not shown in their correct 3D position. This problem might happen in some different situations which two of the most important and common ones will be explained below:

- First case is while having an object in the foreground which has the same color as the background. In this case, while we are using view segmentation, the algorithm doesn't segment the closer object separately from background because the decision will be made based on the color level of the pixels. So, something that should be segmented separately will be merged with the background or other objects with different depth levels. Using the depth image, while knowing that these objects with the same color in view image have different presentations in depth image with different gray level values, they will be segmented separately. In **Fig. 8.4** and considering the left leg of the dancing lady in image **(d),** we can see the effect of this problem in the segmented image. In this case, while the color of the skin of the lady is similar to the background, the two regions are merged into one homogenous region in view segmentation while are separately segmented using depth image.

  This problem will affect the final PSNR a lot, as well the final rendered image doesn't show the scene with an acceptable quality. So, while this defect deals with not exact presentation of virtual views along with a decrease in the quality of the final rendered images, we should think of an applicable way to make our algorithm able to overcome it.

- Another possibility is the case when we have a smooth and homogenous area in color image which will go from a shallow depth to a deep one. In this case as well, the color image segmentation will segment the whole object as a single region, while depth image segmentation will partition in based on the distance from the camera.

  As an example, we can think of a wall in the scene. The whole wall has the same color (sometimes illumination and brightness may change the color and as a result

segmentation), so using the view segmentation it will be considered as a homogenous area and will not be segmented; but while two opposite ends of the wall are not located at the same distance from camera, it will be segmented to different regions based on the gray level of the depth image and depending on the number of regions for that image. In this case, partition coding (in our case Orthobasis) will help us create the content of that specific region close to that of the original depth image. This means, for example, the gray level value of the pixels of the wall will be recreated based on the information that will be sent to the receiver using partition coding. This happens while we segment the wall as a homogenous and relatively larger region, which is not the perfect partitioning. As a result, considering that the region is bigger in comparison to the depth segmentation of the same object, maybe our partition coding methods won't be able to present it as good as if it was segmented to some different partitions. Anyway, as well in this case, the view segmentation won't give a perfect perceived quality of the 3D location of the wall.

Again in **Fig. 8.4**, and paying attention to the right wall in the image **(c)**, easily we can observe the segmentation of the wall based on gray level value of the pixels in depth image; while in image **(d)** the segmentation is based on the different colors of the wall (depends on brightness, illumination and other factors) and has nothing to do with the distance from camera. In this specific example and based on the **Fig. 8.4 (d)**, considering that the wall is segmented in a completely different way, the effect of color partition coding will be even worse than when the whole wall is segmented as a homogenous region. Same as the previous explained problem, this aspect of under segmentation is important too. If this problem remains un-touched, we can completely remove the depth feeling of uniform objects with different depth levels of their two ends which is also related to some regions that should be segmented to different regions but will not. So the solution should be able to handle this problem too.

Considering these two major problems and knowing that exploiting a contour coding method is not logical because of the high cost, the best possible approach would be a mixture of these two methods which can achieve a low bitrate along with a good approximation of the depth segmentation while solving the problems of view partitioning. This said, we are ready to go through the proposed segmentation based coding of depth information for 3D video.

8.5. New segmentation based coding method

Using direct segmentation of original depth images by means of lossless contour coding and Orthobasis partition coding, forces us to send both of these two set of encoded data to the receiver which means very high bitrate as discusses in section 8.4; As well, if we just focus on partition coding and exploit the segmentation from decoded color image, the bitrate will be very appropriate but the quality of rendered images is not high enough to give an acceptable understanding about the 3D aspect of the scene. As a result a new method should be implemented which while keeping the quality high, doesn't introduce a costly transmission bitrate.

In our novel method of segmentation based coding of depth information, we exploit the decoded color image along with original depth image. This means, we do a base segmentation regarding the decoded color image and try to create the BPT branches in such a way that final result matches with that of original depth image segmentation. Obviously the final partitioning will not be exactly the same as original depth image segmentation, but a reasonable approximation of that with respect to the fact that bitrate will be much less is a satisfactory result. In other words, we can consider this joint usage of decoded color image and original depth image partitioning to achieve a segmentation of depth image, as a contour coding method which will represent the contours as much close as possible to those of original depth segments. This can be done using two different methods (both based on BPT segmentation method):

a. Make the initial segmentation with a high number of partitions using decoded color image in the receiver. The stop point of this initial segmentation will be based on a comparison with original depth image segmentation and having a difference error between them less than a specific threshold (i.e. 15%). Shortly, this means the initial segmentation in decoded color image can be up to 15% different from that of original depth image. Afterward, based on the information of original depth segmentation, we will send all the remained merging sequence from that initial segmentation to the final partitioning of the image which will be a smaller number of regions (i.e. initial segmentation with 1500 regions will be converted to a partitioning with only 80 regions). Using this method, we will skip a large part on initial segmentation and only a sequence of numbers will be sent to decoder which will inform the receiver the exact mergings that should be applied after the initial segmentation. By means of this technique, except the first threshold that would be an approximation of the correct initial partitioning, all the other steps will be the same as original depth segmentation steps. The effects of quality and bitrate will be explained later in section 9.6.

b. In the second approach, a base segmentation will be done using the decoded color image in the receiver and it will be mapped on the original depth image afterwards. So, we have the same base segmentation for both original depth and decoded color image. Form this point on, the BPT segmentation will be applied to each image separately and considering that some of the initial partitions won't be merged in the depth segmentation but they will be merged in the decoded color image, there will be a sequence of numbers which will inform the receiver about some mergings that shouldn't be implemented. Using this method all the segmentation will be based on decoded color image, and sometimes when the merging shouldn't be applied (base on the information of original depth segmentation), it will be skipped. As a result in this method the amount of information that should be sent to the receiver will decrease a lot while the implementation of the whole process is based on the decoded color image and will be completed in the decoder.

   Even with applying this approach, we don't merge some partitions that are separated in the decoded color partitioning but not in the original depth partitioning. This fault is due to the over segmentation explained in section 8.4. This happens because we are

not informing the decoder about the **merging sequence**, but the **non merging sequence**. As explained before this defect can be somehow covered using the Orthobasis partition coding method but still won't be perfectly reconstructed. So, we still will have some partitions which are over segmented in the decoder. Considering partition coding, this defect results in some more coefficients to be sent to the receiver, but comparing the benefit that we achieve regarding contour coding this increase of bitrate is completely negligible; and the final quality of rendered virtual images are almost the same(even considering over segmentation), so this is not a big problem. Still and in the future work, there must be an algorithm to make our technique able to overcome this defect too.

Using each one of these two methods, we will decrease the bitrate by a very considerable amount, but we have to keep in mind that both PSNR and bitrate along with comparison between them and our reference is of our interest. This will be explained completely in the following section.

8.6. Effects of new method on quality and bitrate

Results show that both of these methods will give a nearly similar PSNR, because they both use initial segmentation of decoded color image and then they try to create the rest of BPT segmentation sequence using the original depth image. So the final result will be the same as mapping the initial decoded color image segmentation to the original depth image, and applying BPT partitioning method. In this approach, the quality is not the highest achievable one because of the first segmentation that we apply while doing the initial partitioning using decoded color image and not original depth image; but still considering the final results, we can see that this is completely acceptable in the sense of both quality and bitrate.

From this point of view, the quality will remain almost equal for both methods and the preference and decision will be made based on the bitrate. Considering that the sequence of numbers that should be sent to the receiver is different for these two methods, they will be discussed separately in terms of bitrate:

a. Using the first method mentioned in previous section, we don't achieve the lowest bitrate possible and the merging sequence that should be sent to the decoder includes a lot of information so it will increase the bitrate with a considerable amount. This is because we have to send all the merging sequence in between of the two initial partitioning and desired partitioning while for every merging at least two numbers should be sent; each number represents the label of one of the two regions that should be merged. For example if the initial segmentation has 1000 regions and the best partitioning (final one) has 80 regions, we need to send 2*920=1840 numbers. We found out, as expected, that this method is not as efficient as the second one from bitrate point of view; so the second method has been implemented.

b. Based on the fundamental algorithm of the second method, we will not try to send all of the merging sequence as the first approach. Here we will concentrate on the

differences between two segmentations. As explained before, only a sequence of numbers regarding the non merging nodes will be sent to the decoder. For every step, only one number represents the specific merging that shouldn't be implemented. By applying this method, while achieving the same final result, we can decrease the amount of the information to be sent because only a portion of the total mergings has to be marked and for each one only one number should be transmitted. Considering the same example as the previous method, going from 1000 regions to 80 regions, less than 920 numbers should be sent; in practice the real approximation is around 600 numbers which comparing 1840 is much less. Each member of this sequence of numbers, represents the difference of each two consecutive node numbers not the number of each node itself (to minimize the bitrate as much as possible). So, theoretically and mathematically, the bitrate will decrease at least to one half of the previous approach. This depends on the depth image and the decrease can be much more than this.

So, the second approach is the winner because of the better bitrate. Using this method of "segmentation based coding of depth information for 3D video", we have to send two sequences of data: First, the coefficients regarding partition coding, which in this thesis Orthobasis partition coding has been exploited. Second, a sequence of numbers which will inform the decoder about the non merging nodes for the creation of BPT segmentation.

9. Experimental results

9.1. Calculations of PSNR and bitrate

All the experimental results in this thesis will be done for two sequences of Ballet and Breakdancers. Each sequence includes eight synchronized cameras that take the video of the same scene for both depth and color view through the time. We have all the information regarding calibration matrix, translation vector and rotation matrix of these eight cameras, so the projection matrix of them and virtual cameras in between can be calculated. The duration of the whole sequence will be 100 frames for each camera. The rate is 30 frames per second and the bitrate will be calculated based on this frame rate. Based on the frame rate and number of frames, the whole sequence will be almost 3.3 second long which is long enough to give us a good observation about the effectiveness of coding methods.

In this thesis, as mentioned in section 7, the distance between each two original camera will be covered with 9 virtual cameras and the virtual views will be rendered using the projection matrix of these virtual cameras. Using the same method for different coding methods, we can make the comparison between each two desired technique.

By means of this technique of PSNR calculation, we will compare the qualities of final rendered virtual images based on the distribution of virtual cameras in between the original cameras. As a result of this fact, for each figure we should fix the quality and compare the bitrate regarding that specified quality and observe the difference as a measurement of comparison.

The final results will be calculated for the novel proposed method in this thesis and also for two different cases of H.264 coding method as the reference. First, while it is coded only using I frames and second while it is using I, B and P frames for the regular and optimized coding (GOP with length 12). Obviously in the first configuration of H.264, the bitrate will be more comparing the second one. The results will be compared to both of these references to show its efficiency perfectly

9.2. H.264 coding

As mentioned in section 8.2, our reference of comparison is H.264 which is the pioneer in video coding nowadays. The comparison will be done as the encoding/decoding will be applied by using reference as well as our new proposed method on depth images. Then by creating the virtual views using two different decoded depth images and comparing the quality of those rendered color images with the same virtual views rendered by the original depth images, we can achieve a good PSNR comparison. Using this method, we can fix one of bitrate or PSNR for both methods and check the performance on the other one.

Using H.264 method, we can exploit different quantization steps (in the range of 1 to 51) which will result in decoded images with different qualities and different bitrates. Obviously while increasing the quantization step the quality and bitrate will decrease.

To have a good understanding about the effect of quantization step, in **Fig. 9.1** you can see the dependency of bitrate and quality on quantization step for H.264 algorithm (for depth images of Breakdancers sequence). While the quantization step increases, the exactness of the algorithm declines and as a result, as shown on the left Y axis, the PSNR decreases. At the same time, the amount of the information that should be sent reduces, so as shown on the right Y axis the bitrate decreases too.

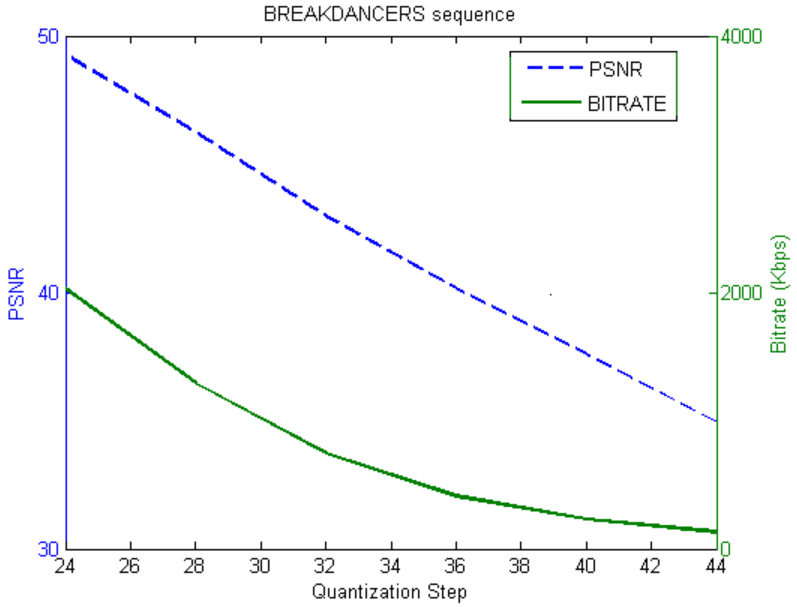


Fig. 9.1: Dependency of bitrate and PSNR on the quantization step in H.264 coding method

In **Fig. 9.2** we can see the traditional graphs of PSNR and Bitrate for two sequences of depth images, based on 8 cameras and 100 frames per camera. Different points in these graphs are calculated based on different quantization steps used to encode the depth images for the whole video sequences. The different quantization steps of H.264 used in this research are 24, 28, 32, 36, 40, 44 and 46 which will represent the best quality to worst quality respectively. As well, you can see the figures for both cases of H.264 configuration based on the number of I frames used to encode the depth images in each GOP. Once, we use H.264 as an I intra coder which all the images will be coded in intra mode (quantization step from 44 to 24). As the second experience, the same procedure will be done with GOPs with length of 12 images(quantization step from 46 to 24).

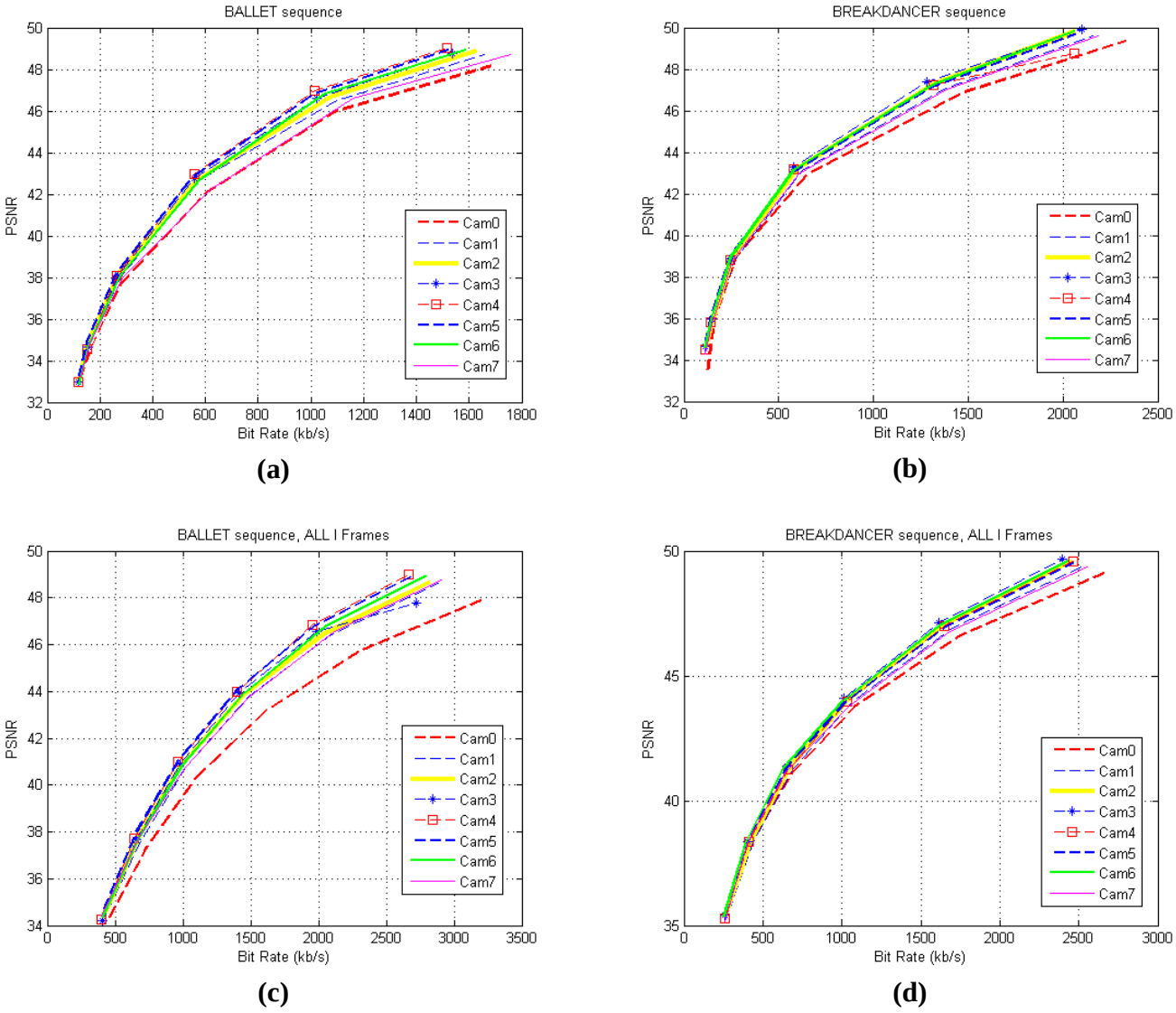
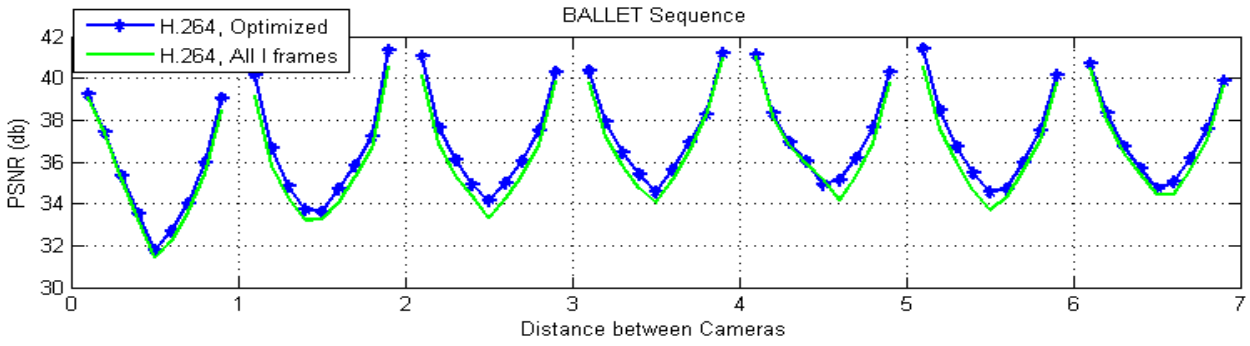


Fig. 9.2: Shows the traditional PSNR/Bitrate graphs of H.264 for two sequences of Ballet and Breakdancers for the depth images - **(a)** , **(b)** using optimized H.264 with all I, B and P frames - **(c)** , **(d)** only using I frames

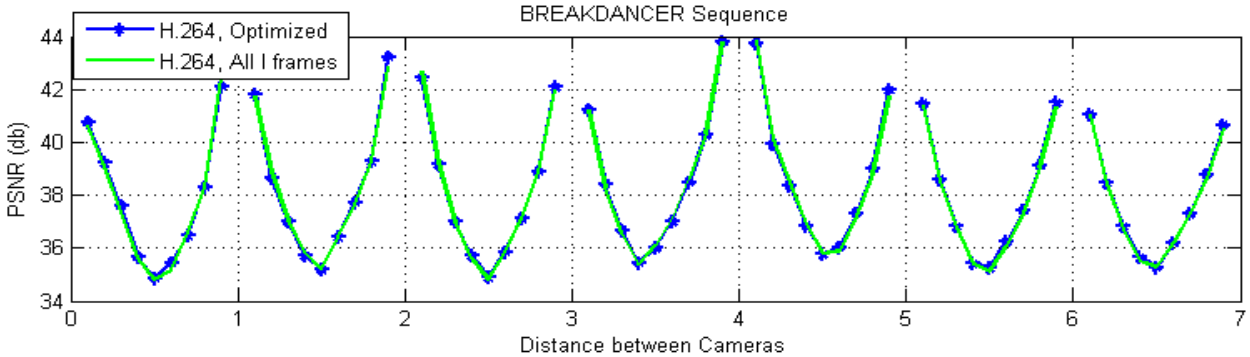
As it can be seen in **Fig. 9.2**, while we are using the H.264 coder as I frames, the bitrate will increase considerably (almost twice in Ballet sequence). We compare the results of both of these configurations of H.264 with the new proposed method.

To do the comparison in this thesis, the quantization step 44 has been used for H.264 algorithm and using the exact method explained in section 7.4, we can find the U shape graphs of quality with respect to the distance between each two original camera. The quantization step equal to **44** will give a decoded depth image with a quality of **34.69** db for Ballet and **35.89 db** for Breakdancers. This is a completely acceptable PSNR and as a

result, the decoded depth images with this quality will be used as the reference for the comparison in this work. In **Fig. 9.7**, a comparison between the decoded depth images with original ones is shown for H.264 coding method with quantization step of 44 and our proposed method. The final graphs are as shown in **Fig. 9.3** for two configurations of H.264. First, while using it as an I intra coder and second, use the optimized configuration of H.264 which exploits I, P and B frames. These graphs are created using the original and H.264 encoded depth images of Ballet and Breakdancers sequences; considering that we have 8 original cameras, there will be 7 possible positions between each two original camera to create the virtual views and render the U shape PSNR graphs (each one like **Fig. 7.4**). In this distance between each two original camera, 9 virtual views have been rendered to create the graphs as shown with * in **Fig. 9.3.** It can be easily seen that there is not much difference between these two configurations of H.264 while the bitrate is completely different (**Table. 9.1** and also shown in **Fig. 9.2**) This result will be compared to that of our segmentation based coding method of depth information in the section 9.6 and the difference between PSNR and bitrate will show the performance of each algorithm.



(a)



(b)

Fig. 9.3: Comparison of PSNR using two configurations of H.264 coding method with quantization step 44 for **(a)** Ballet, **(b)** Breakdancers sequence.

	Ballet (Kbps**)**	**Breakdancers (**Kbps**)**
H.264, All I Intra coding	**414.144**	**262.55**
H.264, Optimized	**155.75**	**148.905**

Table. 9.1: Comparison between the bitrate of two configurations of H.264

9.3. Original depth segmentation

First approach for the new segmentation base coding method of depth information is to segment the depth images separately and only based on the gray level value of its pixels. Then try to send the encoded contour and partitions separately and recreate the depth image in the receiver. Having this in mind, we applied BPT as the segmentation algorithm on the depth image; then using contour chain code method, the boundaries were encoded and put in a buffer. Afterward, each region was encoded using the order two of Orthobasis partition coding algorithm. Both these information should be transmitted to the receiver to make it possible to recreate the depth images.

As mentioned before, this way of coding won't be optimized because a huge amount of bitrate will be devoted to the contour coded information. This known, we can have a good understanding regarding the costliness of this method. In **Fig. 9.4,** we can see the comparison of bitrates between H.264 (QS=44) and original depth segmentation coding for Ballet sequence. In this experiment, chain code contour coding as a lossless method is used to encode the contours and Orthobasis method with order two is used as the partition coding method. As it can be seen, the bitrate for partition coding is acceptable (**Fig. 9.4, b**) while contour coding takes too many bits (**Fig. 9.4, a**). As well for the more or less same PSNR based on virtual rendered images, we can see that the difference between bitrate of this method and H.264 is not acceptable (**Fig. 9.4, c**). For the same quality which will be achieved using 71 regions, we will have a bitrate equal to 121 Kbps for H.264 while it is 3740 Kbps for original depth image segmentation using lossless contour coding along with partition coding method (**Table. 9.2** shows the bitrates relative to the **Fig. 9.4**). This means, using the direct method doesn't result in a good way of coding the depth information because it is too costly to be accepted as an applicable transmitting method.

Just as an example, and to make the pass more clear for the rest of the experimental results, a virtual rendered PSNR graph including the comparison of the results regarding the above mentioned method and both configurations of H.264 is shown in **Fig. 9.5.** Considering that using our method we try to approximate the original depth coding segmentation, the graph that we can see in **Fig. 9.**5 shows the highest possible gain that can be achieved by our method. In this figure the bitrate is completely different and is much more for the proposed method (**Table. 9.3** shows the bitrates relative to the **Fig. 9.5**), but we can have a good understanding about the quality of the rendered images using exact contour and partition coding of original depth images. Also, this graph will be

used afterwards in the next sections to compare our proposed method with the best final result that can be achieved. (Using this specific number of region and algorithm of partition coding - order two of Orthobasis).

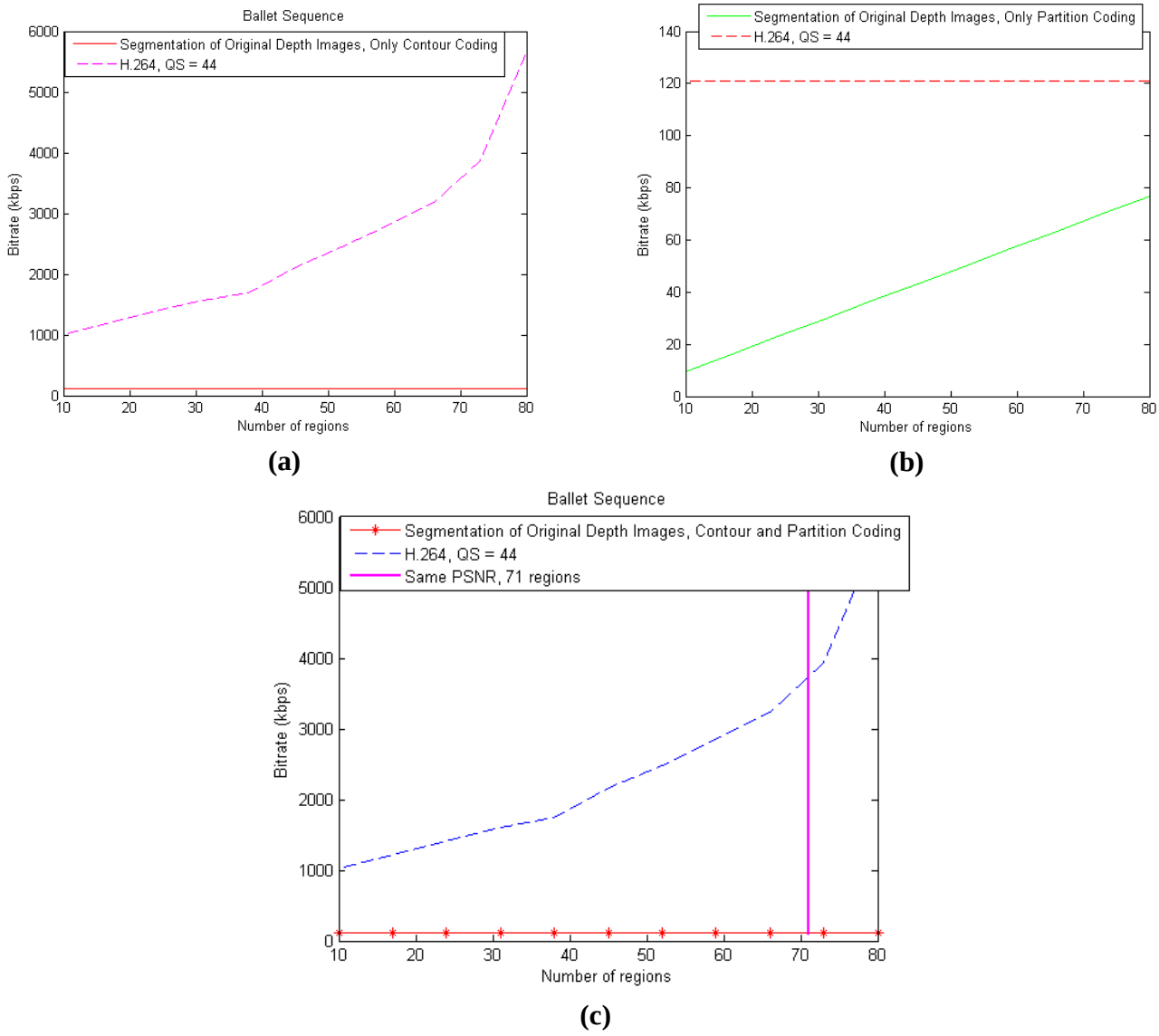
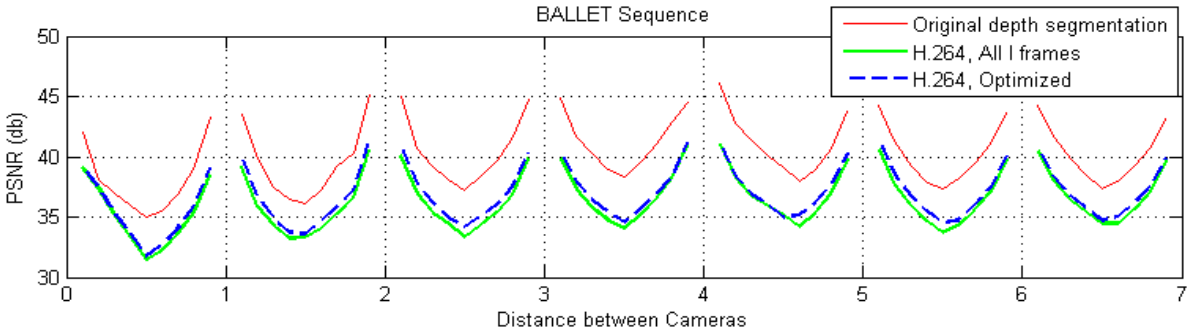


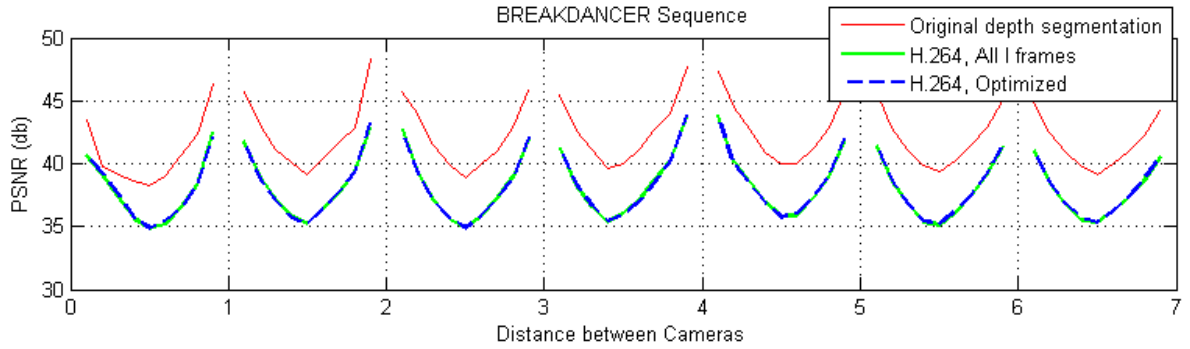
Fig. 9.4: Comparison of bitrate for H.264 and direct segmentation of original depth images, **(a)** Showing only contour coding bitrate **(b)** partition coding bitrate **(c)** Sum of contour and partition coding bitrate along with the same PSNR line

	Contour Coding	Partition Coding	Partition + Contour Coding (Kbps)	H.264 (Kbps) **Optimized.**	H.264 (Kbps) **All I frames**
Ballet sequence	3675	65	3740	155	414

Table. 9.2: The Bitrates for the primary segmentation based method while having the same PSNR with two configurations of H.264 (Ballet sequence)



(a)



(b)

Fig. 9.5: Comparison between PSNR using original depth segmentation (Using lossless chain code contour coding and Orthobasis partition coding with order two) and two configurations of H.264 with quantization step = 44 for two sequences of **(a)** Ballet, **(b)** Breakdancers

	Ballet (Kbps)	**Breakdancers** (Kbps)
H.264, All I Intra coding	**414.144**	**262.55**
H.264, Optimized	**155.75**	**148.905**
Original depth segmentation	**3740**	**3160**

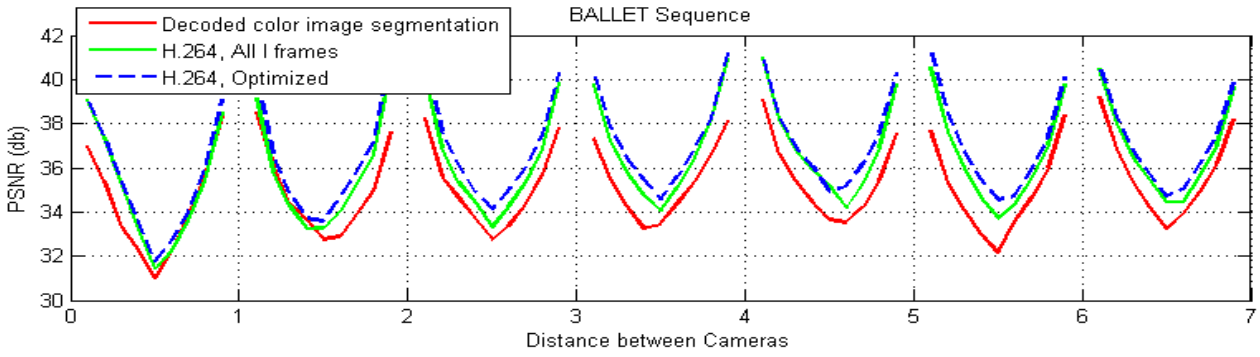
Table. 9.3: Comparison between the bitrate of two configurations of H.264 and Original depth segmentation

9.4. Only view segmentation

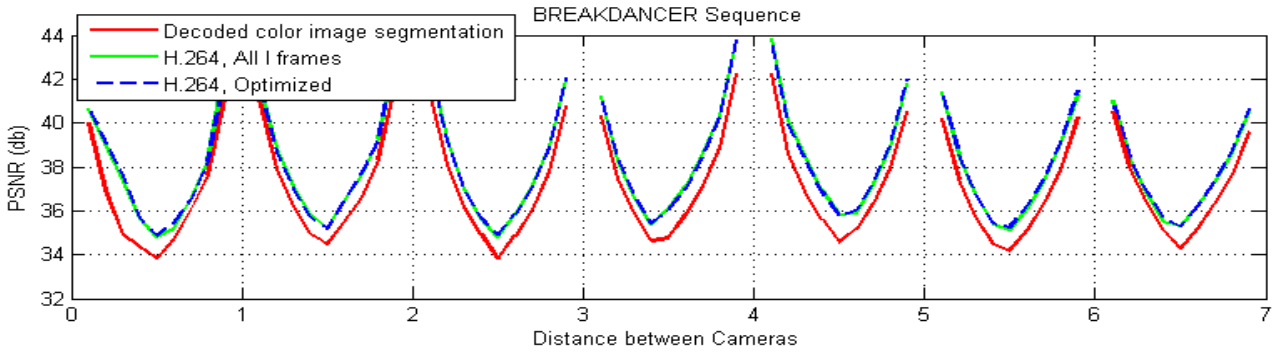
Having in mind that sending the data regarding the contour coding takes most of the bitrate, so a primary approach would be considering segmentation based on the decoded color images and using them to recreate the depth images in the receiver

side. In this approach, while we are already provided with the encoded color images in the receiver, we can use this information to achieve a good approximation of the contours of the original depth image (as explained in section 8.3).

In this case, the only information that should be transmitted is the partition coding of the depth images while their contours will be approximated using the encoded color images at the receiver. **Table. 9.4** shows the difference between the bitrates in this mode. Even though the bitrate regarding the decoded color image segmentation is very low in this experience, but considering the reasons explained in sections 8.3 and 8.4 using the view segmentation alone is not practically useful. As a result, the quality of the rendered images will be less than that of the virtual images rendered using lossless contour coding due to the lack of exactness while using the segmentation of decoded color views. **Fig. 9.6** shows the result of using this method in comparison to H.264 using both configurations.



(a)



(b)

Fig. 9.6: Using only view segmentation to calculate the PSNR and its comparison with both configurations of H.264 for **(a)** Ballet **(b)** Breakdancers sequences

	Ballet (Kbps**)**	**Breakdancers (**Kbps**)**
H.264, All I Intra coding	**414.144**	**262.55**
H.264, Optimized	**155.75**	**148.905**
Decoded color image segmentation	**69.297**	**71.27**

Table. 9.4: Comparison between the bitrate of two configurations of H.264 and Decoded color image segmentation

9.5. Segmentation based coding method

Based on using only view segmentation, we know that it has some defects comparing the original depth segmentation and doesn't represent the depth image as exact as we expect it to be (as described in section 8.4). So, we thought about the new method (explained in section 8.5 & 8.6) to decrease the bitrate as much as possible. Considering that we are basing our initial segmentation on the decoded color image in the receiver, it can not achieve the highest quality for the final rendered images. The reason is that, even if we start with a very high number of regions in base segmentation of the decoded color image, it still has some fundamental and basic differences from that of original depth image which will result in some differences that can not be fixed even after using the non merging sequence. So these fundamental differences will remain and will limit the final quality to some extent.

We know that the depth images have very smooth structures and many parts of the image are homogeneous (which doesn't necessarily happen for color images). As well, we know that these depth images are not directly seen, but are used in the middle steps to create the correct virtual view in between the original cameras. In this thesis, the quantization step 44 will be used for encoding depth images using H.264 which will result in a quality of **34.69 db** for the ballet sequence and **35.89 db** for the Breakdancers sequence. This is a completely acceptable PSNR (in respect to the original depth images) which represents the logicalness of using such bitrates to encode depth images. The above reasons justify the relatively low bitrates used in this research. In **Fig. 9.7**, you can see a comparison between the original depth image and the decoded images using H.264 with quantization step 44 and our proposed method.

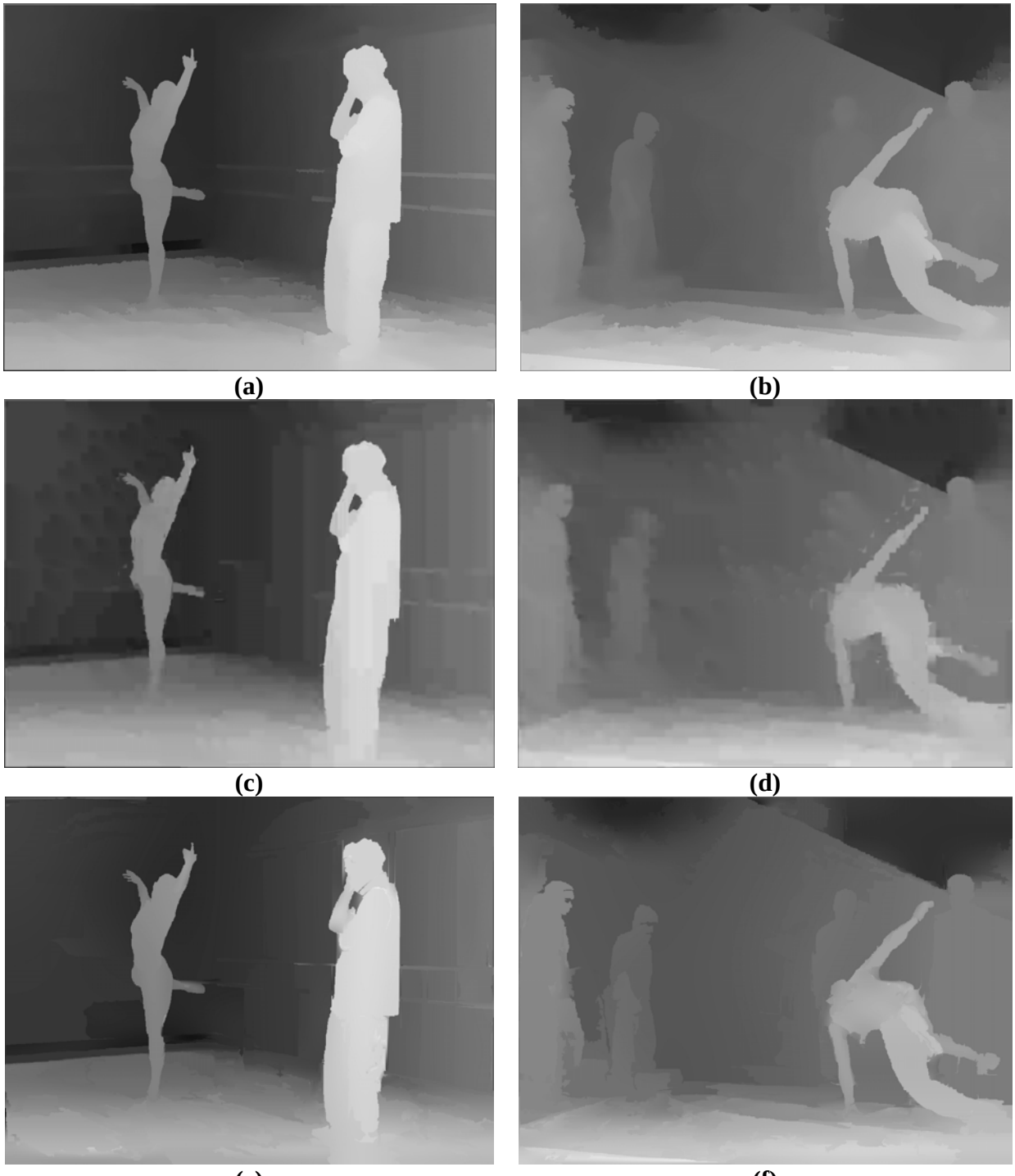


Fig. 9.7: Comparison of original and decoded depth images using H.264 coding method with quantization step equal to 44 and proposed method
(a) Original depth image **(c)** H.264 decoded image **(e)** Proposed method decoded image for **Ballet** Sequence
(b) Original depth image **(d)** H.264 decoded image **(f)** Proposed method decoded image for **Breakdancers** Sequence

The final results, which will use the segmentation based approach, are compared to both configurations of H.264 with quantization step 44 in **Fig. 9.8.** As it can be seen on this figure, the PSNR of our proposed method and optimized configuration of H.264 are close and in some parts even the segmentation based approach outperforms the reference. The PSNR H.264 while using only I frames is lower comparing the other methods and its bitrate is higher, this makes it not a rival with optimized H.264 and segmentation based coding method.

Considering the bitrate of Segmentation based coding method, we have two series of data that should be sent to the decoder. First is the partition coding which includes the coefficients of Orthobasis coding method using order two; second set of data that represents the non-merging sequence (as explained in section 8.5). In **Table. 9.5** we can compare the bitrate of reference to that of the proposed method breaked up to its two fundamental sets of data. To calculate the bitrate of proposed method for the non merging sequence, while having the sequence of numbers representing the nodes, the difference of these numbers has been written to a file as a binary sequence and the result has been entropy coded. The same process has happened for the coefficients regarding partition coding created by Orthobasis method.

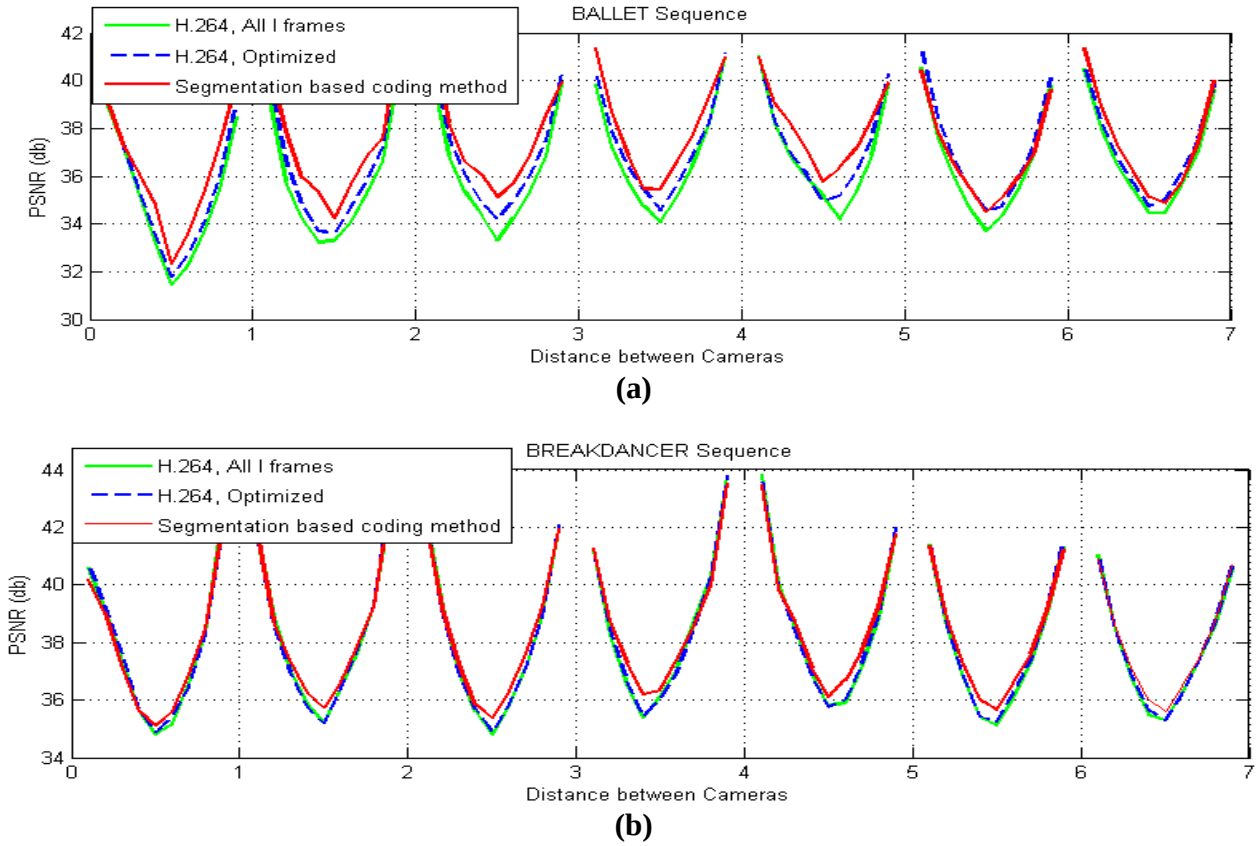


Fig. 9.8: Comparison between Segmentation based coding method with both configurations of H.264 using quantization step equal to 44 for sequences **(a)** Ballet, **(b)** Breakdancers

	Ballet	**Breakdancers**
Segmentation based method (partition coding)	69.297 Kbps	71.27 Kbps
Segmentation based method (non-merging sequence)	51.03	50.93
Segmentation based method (Total)	**120.327**	**122.2**
H.264, All I Intra coding	414.144	262.55
H.264, Optimized	**155.75**	**148.905**

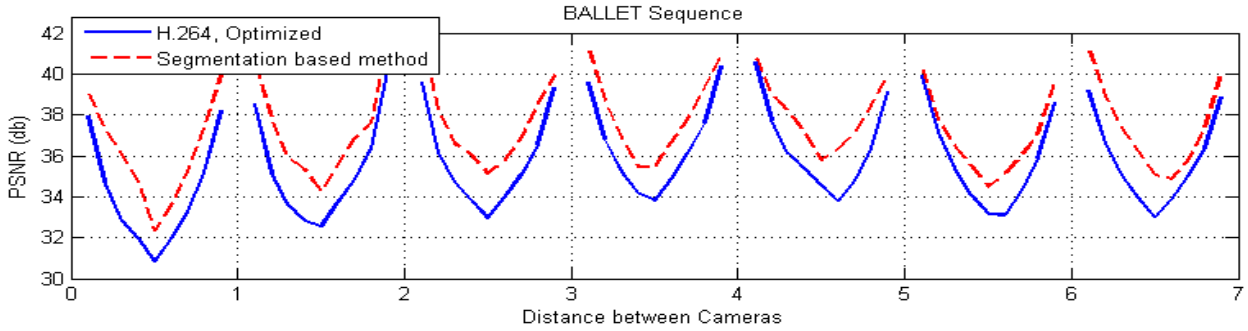
Table. 9.5: Bitrate comparison between H.264 (QS=44) and segmentation based coding method, (All bitrates are Kilo bits per second)

As shown in **Table. 9.5**, the total bitrate of segmentation based coding method is much less than H.264 while it is exploited as an I intra coder. Even while using it as an optimized coder of depth information with GOPs of length 12, still the bitrate achieved by proposed method outperforms that of reference.

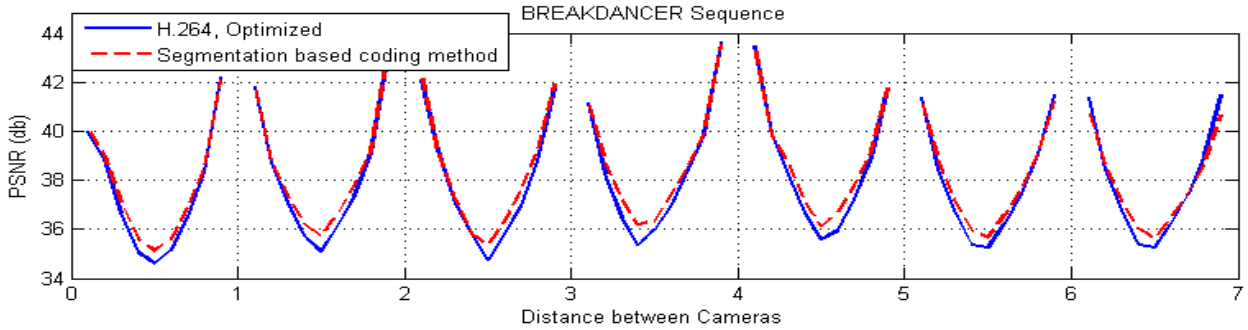
We can do the dual process of the same experiment; it means while having the same bitrate, comparing the PSNR. To achieve the same bitrate as ours by mean of H.264, we have to adjust the quantization step equal to 46 for ballet sequence and 45 for Breakdancers sequence. **Fig. 9.9** shows the comparison of these two methods while having almost the same bitrate. As well, **Table. 9.6** contains the bitrate information related to **Fig. 9.9.** It is important to pay attention to the fact that, using H.264 as an I intra coder increases the bitrate so much that only by using very poor quality of decoded image we can achieve the same bitrate as that of ours; as a result in **Fig. 9.9** only the comparison with optimized H.264 is presented.

	Ballet(Kbps**)**	**Breakdancers(**Kbps**)**
Segmentation based method (Total)	**120.327**	**122.2**
H.264, Optimized	**121.059**	**133.11**

Table. 9.6: Bitrate comparison between H.264 and segmentation based coding method, (All bitrates are Kilo bits per second), Ballet sequence with quantization step equal to 46 and Breakdancers equal to 45



(a)



(b)

Fig. 9.9: Comparison between same bitrate of segmentation based coding method with optimized H.264 using quantization step equal to **(a)** 46 for Ballet sequence, and **(b)** 45 for Breakdancers sequence

Considering **Fig 9.9** along with **Table. 9.6** it is known that while having the same bitrate segmentation based coding method of depth information overcomes H.264.

10. Conclusion and future work

In this work, we have presented a comparative study on depth image compression which deals with a completely new approach for depth coding of 3D video. This new method was implemented and the results were compared to that of our reference. Knowing that H.264 uses block matching method while our new method uses segmentation based algorithms, this method is not an edition or complication of the previously implemented method, but it is a method implemented for the first time in this field.

This method did the partitioning of original depth image based on segmentation of both decoded color image and original depth image. This was done using an initial segmentation based on decoded color image; Then trying to complete the BPT sequence using some extra information received from encoder. This data informs the decoder which nodes in BPT creation shouldn't be merged together while doing the BPT segmentation process. While only with some limited information, we can instruct the decoder how to implement the rest of the segmentation steps on the decoded color image, the bitrate in this approach won't be high at all. This happens while we have achieved a good quality if rendered virtual views.

Based on the results achieved, segmentation based coding method of depth images is able to overcome H.264 for the specific and reasonable qualities. This means, while having the same quality, we can achieve a better bitrate for the virtual rendered images in between original cameras which has been the final goal of this thesis from the beginning.

We have to keep in mind that this result has been achieved without exploiting the relevancies between consecutive images in time. This said, we can mention some of the possible future work possibilities:

- **Using time interpolation**

Knowing that both of the sequences of depth images are taken with a frame-rate equal to 30 fps, it is obvious that there is a very high dependency between images that occur in consecutive frames. This can be done, by processing the coefficients regarding partition coding. First a logical dependency between segmentation of consecutive frames should be pursued. Then the coefficients regarding same partitions will be processes to find out the correlations between them. Regarding lack of time for this thesis, we couldn't manage to exploit this dependency which will result in a considerable decrease in bitrate. This is the most important future work that can reduce the cost of the method.

- **Handling the "to be merged" regions**

As mentioned in section 8.5, it is possible to think of possible algorithms to send some information to receiver informing it about the need of merging some segments with each other. In explained approach in this thesis, we are concentrating on non merging sequence which will have a very good effect on the outcome of the system. But still there will be some partitions which will be over segmented using decoded color image segmentation (explained in section 8.4). Implementing this algorithm will result in an

even more decreased bitrate based on the partition coding, while some data should be sent to decoder informing it about the needed mergings.

- **Using rate-distortion methods**

There are some numbers in this thesis which has been achieved and used based on some experimental results. i.e. initial segmentation of decoded color image, best segmentation number for Breakdancers and Ballet sequences and order of Orthobasis partition coding method. While using rate-distortion algorithms we will be able to find the best values for these which will result in a better quality and lower bitrate.

- **Alternative partition coding methods**

In this thesis, second order of Orthobasis partition coding has been exploited to encode the information of each region. Using more optimized methods and even some techniques adjusted to specific characteristics of depth images will be able to improve the results in the sense of both quality and bitrate.

11. References

[1] A. Smolic, K. Müller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and Thomas Wiegand, 3D Video and Free Viewpoint Video –Technologies, Applications and MPEG Standards, ICME 2006, IEEE International Conference on Multimedia and Expo, Toronto, Ontario, Canada, July 2006.

[2] A. Smolic, and P. Kauff, Interactive 3D Video Representation and Coding Technologies, Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery, vol. 93, no. 1, Jan. 2005.

[3] H. Ozaktas, and L. Onural, Three-Dimensional Television: Capture, Transmission, and Display, Springer, Heidelberg, December 2007.

[4] ITU-T Rec. & ISO/IEC 14496-10 AVC., Advanced video coding for generic audiovisual services, 2005

[5] P.Merkle, K. Muler, A. Smolic, and T. Wiegand. Efficient compression of multi-view video exploiting interview dependencies based on H.264 in proc. IEEE int'l Conf. Multimedia & Expo, 2006

[6] M. Bertero, T.A. Poggio, and V. Torre., Ill-posed problems in early vision. Proceedings of the IEEE, 76:869–887, 1988.

[7] P. Salembier and F. Marqués. Region-based representation of image and video: Segmentation tools for multimedia services. IEEE Trans. on Circuit and Systems for Video Technology, 9(8):1147–1169, December 1999.

[8] J. K. Aggarwal and W. N. Martin., Analyzing dynamic scenes containing multiple moving objects. In T. S. Huang, editor, Image sequence analysis, pages 355–380. Springer- Verlag, New York, 1981.

[9] P. Salembier and M. Pardàs. Hierarchical morphological segmentation for image sequence coding. IEEE Trans. on Image Processing, 3(5):639–651, September 1994.

[10] P. Salembier., Morphological multiscale segmentation for image coding. EURASIP Signal Processing, 38(3):359–386, September 1994.

[11] P. Willemin, T. Reed, and M. Kunt. Image sequence coding by split and merge. IEEE Transactions on Communications, 39(12):1845–1855, 1991.

[12] M. Pardàs and P. Salembier., 3D morphological segmentation and motion estimation for image sequences. EURASIP Signal Processing, 38(2):31–43, September 1994.

[13] S. Rajala, M. Civanlar, and W. Lee., Video data compression using three-dimensional segmentation based on HVS properties. In IEEE, editor, International Conference on Acoustics, Speech and Signal Processing, pages 1092–1905, New York (NY), USA, 1988.

[14] M. Pardàs and P. Salembier., Time-recursive segmentation of image sequences. In EURASIP, editor, EUSIPCO 94, VII European Signal Processing Conference, pages 8–21, Edinburgh, U.K., September 13-16 1994.

[15] F. Marqués, V. Vera, and A. Gasull., Recursive image sequence segmentation by hierarchical models. In Proc. of the 12th International Conference on Pattern Recognition, pages 523–525, Oct 1994.

[16] B. Marcotegui and F. Meyer., Morphological segmentation of image sequences. In J. Serra and P. Soille, editors, Mathematical morphology and its applications to image processing, pages 101–108. Kluwer Academic Publishers, 1994.

[17] P. Salembier, F. Marqués, M. Pardàs, R. Morros, I. Corset, S. Jeannin, L. Bouchard, F. Meyer, and B. Marcotegui., Segmentation-based video coding system allowing the manipulation of objects. IEEE Trans. on Circuits and Systems for Video Technology, 7(1):60–73, February 1997.

[18] F. Dufaux and F. Moscheni., Segmentation-based motion estimation for second generation video coding techniques. In L. Torres and M. Kunt, editors, Video Coding: The Second Generation Approach, pages 79–124. Kluwer Academic Publishers, 1996. ISBN: 0 7923 9680 4.

[19] F. Dufaux and F. Moscheni., Spatio-temporal segmentation based on motion and static segmentation. In IEEE International Conference on Image Processing, ICIP'95, Washington, DC, October 1995.

[20] T. Meier and K.N. Ngan., Automatic segmentation of moving objects for video object plane generation. IEEE Trans. on Circuits and Systems for Video Technology, 8(5):525–538, September 1998.

[21] Y. Deng and B.S. Manjunath Netra-v., Toward an object-based video representation. IEEE Trans. on Circuits and Systems for Video Technology, 8(5):616–627, September 1998.

[22] F. Marqués, M. Pardàs, and P. Salembier. Coding-oriented segmentation of video sequences.In L. Torres and M. Kunt, editors, Video Coding: The Second Generation Approach, pages 79–124. Kluwer Academic Publishers, 1996. ISBN: 0 7923 9680 4.

[23] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation and information retrieval," IEEE Trans. Image Process., vol. 9, no. 4, pp. 561–575, Apr. 2000.

[24] Veronica Vilaplana, Member, IEEE, Ferran Marques, Member, IEEE, and Philippe Salembier, Member, IEEE, "Binary Partition Trees for Object Detection", IEEE Trans. Image Process., vol. 17, no. 11, Nov 2008

[25] M. Kunt, A. Ikonomopoulos, and M. Kocher., Second generation image coding techniques. Proceedings of the IEEE, 73(4):549–575, April 1985.

[26] H.G. Musmann, M. Hotter, and J. Ostermann., Object-oriented analysis-synthesis coding of moving images. Signal Processing: Image Communication, 1(2):117–138, October 1989.

[27] L. Torres and M. Kunt. Video Coding: The second generation approach. Kluwer Academic Publishers, Boston, 1996.

[28] C. Gu., 3D contour image coding by morphological filters and motion estimation. In IEEE, editor, International Conference on Acoustics, Speech and Signal Processing, ICASSP'94, Australia, April 1994.

[29] P. Salembier, L. Torres, F. Meyer, and C. Gu. Region-based video coding using mathematical morphology. Proceedings of IEEE (Invited Paper), 83(6):843–857, June 1995.

[30] E. Reusens. Joint optimization of representation model and frame segmentation for generic video compression. Signal Processing, 46:105–117, September 1995.

[31] M. Gilge. Region-oriented texture coding. In L. Torres and M. Kunt, editors, Video Coding: The Second Generation Approach, pages 171–218. Kluwer Academic Publishers, 1996. ISBN: 0 7923 9680 4.

[32] T. Sikora, S. Bauer, and B. Makai. Efficiency of shape-adaptive 2-d transforms for coding of arbitrarily shaped image segments. IEEE Trans. on Circuits and Systems for Video Technology, 1995.

[33] E. Jensen, K. Rijkse, I. Lagendijk, and P. van Beek. Coding of arbitrarily sahaped image segments. In Proceedings of Workshop on Image Analysis and Synthesis in Image Coding, Berlin, October 1994.

[34] H. Katata, N. Ito, T.Aono, and H.Kusao. Object Wavelet Transform for coding of arbitrarily-shaped image segments. IEEE Trans. on Circuits and Systems for Video Technology, 7(1):234–237, February 1997.

[35] A. Kaup., Object-based texture coding of moving video in MPEG-4. IEEE Trans. On Circuits and Systems for Video Technology, 9(1):5–15, February 1999.

[36] M. Gilge, T. Engelhardt, and Mehlan R. Coding of arbitrarily shaped image segments based on a generalized orthogonal transform. EURASIP, Image Communications, 1(2):153–180, October 1989.

[37] H. Freeman. On the coding of arbitrary geometric configurations. IRE Trans. Electronic, Comp., EC(10):260–268, June 1961.

[38] J. Serra. Image Analysis and Mathematical Morphology. Academic Press, 1982.

[39] F. Marqués, S. Fioravanti, and P. Brigger., Coding of image partitions by morphological skeletons using overlapping structuring elements. In IEEE, editor, 1995 IEEE Workshop on Nonlinear Signal and Image Processing, pages 250–253, Halkidiki, Greece, June 20-22 1995.

[40] A.J. Pinho. A method for encoding region boundaries based on transition points. Image and Vision Computing, 16(3):213–218, March 1998.

[41] R. Kresch and D. Malah., Quadtree and bitplane decompositions as particular cases of the generalized morphological skeleton. In IEEE, editor, 1995 IEEE Workshop on Nonlinear Signal and Image Processing, Halkidiki, Greece, June 1995.

[42] T. Minami and K. Shinohara., Encoding of line drawings with multiple grid chain code. IEEE Trans. on Pattern Analysis and Machine Intelligence, 8:265–276, March 1986.

[43] A. Gasull, F. Marqués, and J. A. Garc´ıa. Lossy, image contour coding with multiple grid chain code. In Workshop on Image Analysis and Synthesis in Image Coding94, WIASIC'94, pages B4.1–B4.4, Berlin, Germany, October 1994

[44] C.T. Zahn and R.Z. Roskies. Fourier descriptors for plane closed curves. IEEE Trans. on Computers, 21(3):269–281, March 1972.

[45] P. Van Otterloo., A contour-oriented approach for shape analysis. Prentice Hall International(UK), 1991.

[46] J.G. Dunham., Optimum unifirm piecewise linear approximation of planar curves. IEEE Trans. on Pattern Analysis and Machine Intelligence, 8:67–75, January 1986.

[47] L.L. Schumaker. Spline functions: basic theory. Wiley-Interscience, New York, 1981.

[48] P. Salembier, F. Marqués, and A. Gasull., Coding of partition sequences. In L. Torres and M. Kunt, editors, Video Coding: The Second Generation Approach. Kluwer, 1996. ISBN: 0 7923 9680 4.

[49] F. Marqués, J. Sauleda, and T. Gasull. Shape and location coding for contour images.In Picture Coding Symposium, pages 18.6.1–18.6.2, Lausanne, Switzerland, March 1993.

[50] Claude E. Shannon: A Mathematical Theory of Communication, Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, 1948.

[51] A Method for the Construction of Minimum-Redundancy Codes. David A. Huffman. http://compression.graphicon.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf

[52] ThomasWiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthxra, Overview of the H.264 Video Coding Standard , IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 13, NO. 7, JULY 2003

[53] Introduction to H.264, Bo Hong Multimedia Communications Laboratory University of Texas at Dallas, November 22, 2002

[54] ATI Technologies Inc., Introduction to H.264, 09/05. PN: 129-40558-10

[55] Carolina Martínez Ordinas , Virtual View Generation for Free Viewpoint Applications, Master thesis to be completed and available in ETSETB library by July 2009

[56] Josep Ramon Morros, Optimization of Segmentation Based Video Sequence Coding Techniques: Application to Content Based Functionalities, Ph.D. Dissertation, October 2004

[57] Video coding, the second Generation Approach, Edited by Luis Torres and Muraat Kunt, November, 1995

[58] P. Merkle, Y. Morvan, A. Smolic, D. Farin1, K. Müller, P.H.N. de With, and T. Wiegand , The Effect of Depth Compression on Multi-view Rendering Quality

[59] N. Garc´ıa, F. Jaureguizar, and J.I. Ronda. Pixel-based video compression schemes. In L. Torres and M. Kunt, editors, Video Coding: The Second Generation Approach, pages 31–78. Kluwer Academic Publishers, 1996. ISBN: 0 7923 9680 4.

[60] C. Stiller and J. Konrad. Estimating motion in image sequences: A tutorial on modeling and computation of 2D motion. IEEE Signal Processing Magazine, 16(4):70–91, July 1999.

[61] Thomas Wiegand and Bern Girod. Multi-Frame Motion-Compensated Prediction for Video Transmission. Kluwer Academic Publishers, Boston, 2001.

[62] M. Kunt, A. Ikonomopoulous, and M. Kocher, Second-generation image coding, Proc. IEEE, vol. 73, pp. 549–574, Apr. 1985.

[63] R. Talluri et al., A robust, scalable, object-based video compression technique for very low bit-rate coding, IEEE Trans. Circuits Syst. Video Technol., vol. 7, pp. 221–234, Feb. 1997.

[64] H. Sanderson and G. Crebbin, Region-based image coding using polynomial intensity functions, Proc. IEE Vision Image Signal Processing, vol. 143, no. 1, pp. 15–22, Feb. 1996.

[65] G. Biggar et al., Segmented-image coding: performance comparison with the discrete cosine transform, Proc. Inst. Elect. Eng.—F, vol. 135, no. 2, pp. 121–132, Apr. 1988.

[65] T. Sikora and B. Makai, Shape-adaptive DCT for generic coding of video, IEEE Trans. Circuits Syst. Video Tech., vol. 5, no. 1, pp. 59–62, Feb. 1995.

[66] P. Kauff et al., Functional coding of video using a shape-adaptive DCT algorithm and an object-based motion prediction toolbox, IEEE Trans. Circuits Syst. Video Technol., vol. 7, pp. 181–197, Feb. 1997.

[67] H. J. Barnard, Image and video coding using a wavelet decomposition, Ph.D dissertation, Dept. Elect. Eng., Delft Univ. Technol., The Netherlands, May 1994.

[68] D. Saupe, M. Ruhl, Evolutionary fractal image compression, IEEE International Conference on Image Processing (ICIP'96), Lausanne, Sept. 1996 1