Depth Coding based on Binary Tree Decomposition

Gonçalo Nuno Silva do Carmo

Dissertation submitted for obtaining the degree of
Master in Electrical and Computer Engineering

Jury

President: Prof. José Bioucas Dias
Supervisor: Prof. Fernando Pereira
Co-Supervisor: Dr. Matteo Naccari
Members: Prof. Luís Ducla Soares

October 2010
Acknowledgments

First and foremost, I would like to thank both my parents for their love and outstanding effort to support me in all aspects, namely during my Master Degree, since without it none of this would be possible. I would like to thank my brother for his love and for cheering me up on my weekends at home. I'm also very thankful to my family for their constant messages of encouragement and for their constant support along the past years.

Regarding IT Image Group members, I would like to show my deepest gratitude to: Prof. Fernando Pereira - for his guidance, enthusiasm and perseverance towards the success of this Thesis – to Matteo Naccari and Catarina Brites – for helping me to develop my programming skills and also for their admirable devotion, patience and vision – and finally to all Image Group members – for providing an exciting and friendly atmosphere to work with.

Last but not the least, I want to thank my IST friends (Nuno Couto, Rui Trindade, Filipa Henriques, André Neves, André Chiubeles, Tiago Henriquez, André Esteves, Tiago Veiga, Tiago Simão, Tiago Correia, João Vicente, Daniel Quina, Hugo Correia and Filipe Wiener) and to my friends from Évora (Judite Salavessa, Vasco Monteiro, Nélsom Cachapa, Pedro Raimundo, Ricardo Polha and Nuno Caraça) for their friendship and constant support.
Abstract

Since the world is a 3D world, there has been a growing interest in 3D video systems, largely motivated by the novel capabilities that acquisition, representation and display technologies are boosting and thus targeting the improvement of the multimedia user experiences. Both highly immersive audio and video are essential in 3D systems for a powerful user experience and, therefore, its design and optimization are central in efficient 3D systems. Since depth data brings several advantages to 3D systems, it may need to be involved in the 3D video content representation process; in this context, a 3D system using depth information - besides color information - is able to more efficiently support the necessary 3D features and capabilities (such as Head Motion Parallax), notably if an efficient depth coding solution is available.

Depth coding is a research field with growing interest by the video coding community, due to the emerging related needs in 3D systems. Therefore, in this Thesis is proposed a depth coding solution - based on a content adaptive mesh approach – which has as main objectives to: i) contribute for the research field of depth coding; ii) compete with already implemented depth coding solutions; iii) introduce depth coding novel tools that may lead to a better support of 3D features; iv) help to further meet the expectations of the user’s demand for more immersive 3D experiences.

The proposed depth coding solution showed encouraging results by outperforming JPEG and H.264/AVC without inter prediction (i.e. intra coding only) codecs based on a PERR assessment. For some conditions, the proposed depth coding solution outperforms H.264/AVC with inter prediction and MVC codecs which is impressive since this depth coding solution only has intra prediction mode. A PSNR assessment is also made in order to ease the comparison of this Thesis results with other works.

Keywords: Depth coding, Binary Tree Decomposition, 3D Systems, Depth Assessment
Resumo

Uma vez que o mundo é um mundo 3D, tem havido um interesse crescente em sistemas de vídeo 3D, em grande parte motivado pelas novas capacidades que as tecnologias de aquisição, representação e exibição estão a proporcionar e, desta forma, a melhorar as experiências de multimédia do user. Tanto áudio como vídeo altamente imersivos são essenciais em sistemas 3D para uma poderosa experiência por parte do user e, portanto, a sua concepção e optimização são centrais em sistemas 3D eficientes. Como a informação de depth traz inúmeras vantagens para os sistemas de 3D, ela pode precisar de ser envolvida no processo de representação de vídeo 3D; neste contexto, um sistema 3D que usa informação de depth - além da informação de cor - é capaz de apoiar de forma mais eficiente as capacidades e features 3D (como por exemplo a Head Motion Parallax), nomeadamente se houver uma solução de codificação de depth eficiente.

A codificação de depth é um campo de investigação com interesse crescente por parte da comunidade de codificação de vídeo, devido às suas necessidades emergentes relacionadas com sistemas 3D. Portanto, nesta Tese é proposta uma solução de codificação para depth - com base em uma abordagem de mesh adaptativo ao conteúdo - que tem como principais objectivos: i) contribuir para o campo de pesquisa da codificação de depth; ii) competir com soluções de codificação de depth já implementadas iii) introduzir novas ferramentas de codificação de profundidade que pode levar a um melhor suporte de recursos 3D; iv) contribuir para uma melhor resposta às expectativas de experiências 3D mais imersivas parte do user.

Os resultados da solução proposta de codificação de depth são encorajadores, uma vez que esta solução tem melhor desempenho que os codificadores JPEG e H.264/AVC sem inter prediction (ou seja, apenas codificação intra) baseados numa avaliação PERR (Percentage of Errored Pixels). Para algumas condições, a solução proposta de codificação depth supera os codificadores H.264/AVC com inter prediction e o MVC, que é impressionante, pois esta solução proposta não tem inter prediction nem interview prediction. Uma avaliação baseada em PSNR também é feita para facilitar a comparação dos resultados desta Tese com outras soluções da literatura.

Palavras Chave: Codificação de depth; Decomposições em árvore binária; Sistemas 3D, Avaliação da Qualidade de depth.
# Table of Contents

1. **Introduction** ............................................................................................................................... 1  
   1.1. Context and Motivation .................................................................................................................. 1  
   1.2. Objectives .................................................................................................................................. 3  
   1.3. Thesis Organization .................................................................................................................... 4  

2. **Reviewing 3D Video Systems** ..................................................................................................... 5  
   2.1. About Depth Data ........................................................................................................................ 6  
   2.2. Relevant 3D Video Systems ......................................................................................................... 9  
      2.2.1. Stereo Video Systems ........................................................................................................... 11  
      2.2.2. Multiview Video Systems ................................................................................................... 13  
      2.2.3. Video Plus Depth Systems .................................................................................................. 15  
      2.2.4. Multiview Video Plus Depth Systems ............................................................................... 16  
   2.3. Summary .................................................................................................................................... 19  

3. **Reviewing Depth Coding Techniques** .......................................................................................... 21  
   3.1. H.264/AVC Depth Coding ........................................................................................................... 21  
      3.1.1. Basic Technical Approach ................................................................................................... 21  
      3.1.2. Architecture ....................................................................................................................... 22  
      3.1.3. Main Coding Tools .............................................................................................................. 23  
      3.1.4. Performance Evaluation ..................................................................................................... 24  
      3.1.5. Strengths and Weaknesses .................................................................................................... 25  
   3.2. Hierarchical Decomposition Mesh-based Depth Coding ............................................................. 25  
      3.2.1. Basic Technical Approach ................................................................................................... 25  
      3.2.2. Architecture ....................................................................................................................... 26
3.2.3. Main Coding Tools ................................................................. 27
3.2.4. Performance Evaluation ......................................................... 29
3.2.5. Strengths and Weaknesses ..................................................... 30
3.3. Platelet-based Depth Coding ...................................................... 30
  3.3.1. Basic Technical Approach .................................................... 30
  3.3.2. Architecture ......................................................................... 31
  3.3.3. Main Coding Tools ............................................................... 32
  3.3.4. Performance Evaluation ....................................................... 33
  3.3.5. Strengths and Weaknesses .................................................... 35
3.4. Content Adaptive Mesh-based Depth Coding ................................... 35
  3.4.1. Basic Technical Approach .................................................... 35
  3.4.2. Architecture ......................................................................... 36
  3.4.3. Main Coding Tools ............................................................... 37
  3.4.4. Performance Evaluation ....................................................... 38
  3.4.5. Strengths and Weaknesses .................................................... 40
3.5. Summary .................................................................................. 41

4. Proposing Depth Coding Solution: IST-Depth ....................................43
  4.1. Architecture ............................................................................ 44
  4.2. Functional Description by Module ............................................. 46
  4.3. Algorithmic Description by Module .......................................... 47
    4.3.1. Depth Analysis .................................................................. 47
      4.3.1.1. Initial Triangular Decomposition .................................. 48
      4.3.1.2. Triangular Decomposition .......................................... 49
      4.3.1.3. Planar Approximation ................................................. 53
      4.3.1.4. Assessing the Triangle’s Quality and Size ...................... 60
    4.3.2. Binary Tree Encoding .......................................................... 62
      4.3.2.1. Binary Tree Symbol Creation ..................................... 63
      4.3.2.2. Binary Tree Symbol Entropy Encoding .......................... 64
    4.3.3. Depth Values Encoding ....................................................... 65
# Index of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Stereoscope, one of the first devices which dealt with 3D content [2]</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Hyundai 3D TV, HP project to provide 3D from a laptop LCD display and Samsung 3D TVs [9] [10] [11]</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Philips 3DWOW TV [12]</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Depth perception</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Depth image example [14]</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Zcam for 3DV Systems [17]</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>HDTV Axi-Vision Camera [18]</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Stereo pair capture example [21]</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>Basic 3D video system data processing chain</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>CSV system data: two color signals [25]</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Stereo camera developed for the Avatar movie [26]</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>Combined temporal and interview prediction for stereo video [8]</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>MVV system data: multiple color signals [25]</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>Example of MVC temporal and interview prediction structure [27]</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Example of outputs and 3D views for a MVV system</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>V+D system data: one color and one depth signals [25]</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>MV+D system data: multiple color and multiple depth signals [25]</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>MV+D and view synthesis (DIBR) for the efficient support of multiview autostereoscopic displays [8]</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>LVD system data: Two color signals and two depth signals, each one of the same type containing information regarding the main and the background layers [25]</td>
<td>19</td>
</tr>
<tr>
<td>20</td>
<td>H.264/AVC encoder architecture</td>
<td>22</td>
</tr>
<tr>
<td>21</td>
<td>H.264/AVC based depth coding solution</td>
<td>23</td>
</tr>
</tbody>
</table>
Figure 46 – Defining the triangle sides based on the numbered vertices. ................................................................. 51

Figure 47 – a) Triangle to be decomposed (input) where position 3 (in this case) corresponds to the midpoint in the largest edge; b) Left/upper depth triangle; c) Right/lower depth triangle; d) Renumbered left/upper depth triangle (output); e) Renumbered right/lower depth triangle (output). .................................................................................. 52

Figure 48 - Planar Approximation module's algorithm flowchart .................................................................................. 54

Figure 49 – Example of the creation of a depth window (transparent green) based on the Cartesian coordinates of a triangle's vertices. ........................................................................................................................................ 55

Figure 50 - a) Top-bottom, from left to right (raster scanning order); b) Top-bottom, from right to left; c) Left-right, top to bottom; d) Left-right, bottom to top. .......................................................................................................................... 56

Figure 51 - Example of chosen scanning pattern (red) for a triangle (blue) inside its depth window (green). .... 57

Figure 52 – Example binary trees representing the decomposition of a (randomly) decomposed a depth map. .. 61

Figure 53 – Binary Tree Encoding architectural module's flowchart .............................................................................. 62

Figure 54 – BBS and DBS binary tree encoding examples and their corresponding bitstreams ........................................ 64

Figure 55 – Depth Values Encoding architectural module's flowchart .......................................................................... 65

Figure 56 – Different decompositions using the same depth position ........................................................................... 68

Figure 57 – “Beer Garden” color and depth example frames .......................................................................................... 72

Figure 58 - "Breakdancers" color and depth example frames. ......................................................................................... 72

Figure 59 - "Newspaper" color and depth example frames. ............................................................................................... 72

Figure 60 – PERR results for the “Beer Garden” multiview sequence. ............................................................................. 78

Figure 61 - PERR results for the “Breakdancers” multiview sequence .............................................................................. 79

Figure 62 – PERR results for the “Newspaper” multiview sequence. ............................................................................... 80

Figure 63 – PSNR results for the “Beer Garden” multiview sequence. ........................................................................... 81

Figure 64 – PSNR results for the “Breakdancers” multiview sequence ............................................................................. 82

Figure 65 – PSNR results for the “Newspaper” multiview sequence. ............................................................................... 83
Index of Tables

Table 1 – PSNR differences for virtual view rendering with low, middle and high bitrates between H.264/AVC Intra and platelet-based depth coding [36]........................................................................................................35

Table 2 – Changed parameter for H.264/AVC Inter............................................................................................76

Table 3 – Changed parameters for MVC Inter.....................................................................................................77
### List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DMC</td>
<td>3D Mesh Coding</td>
</tr>
<tr>
<td>AVC</td>
<td>Advanced Video Coding</td>
</tr>
<tr>
<td>BBS</td>
<td>Breadth Based Scanning</td>
</tr>
<tr>
<td>BFS</td>
<td>Breadth First Search</td>
</tr>
<tr>
<td>BPP</td>
<td>Bit per Pixel</td>
</tr>
<tr>
<td>BSP</td>
<td>Binary Space Partitions</td>
</tr>
<tr>
<td>CSV</td>
<td>Conventional Stereo Video</td>
</tr>
<tr>
<td>DBS</td>
<td>Depth Based Scanning</td>
</tr>
<tr>
<td>DFS</td>
<td>Depth First Search</td>
</tr>
<tr>
<td>FIFO</td>
<td>First In First Out</td>
</tr>
<tr>
<td>FVV</td>
<td>Free Viewpoint Video</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Pictures</td>
</tr>
<tr>
<td>HD</td>
<td>High Definition</td>
</tr>
<tr>
<td>HDTV</td>
<td>High Definition Television</td>
</tr>
<tr>
<td>HVS</td>
<td>Human Visual System</td>
</tr>
<tr>
<td>HMPN</td>
<td>Head Motion Parallax Navigation</td>
</tr>
<tr>
<td>LDV</td>
<td>Layered Depth Video</td>
</tr>
<tr>
<td>LIFO</td>
<td>Last In First Out</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Pictures Experts Group</td>
</tr>
<tr>
<td>MV+D</td>
<td>Multiview Video Plus Depth</td>
</tr>
<tr>
<td>MVC</td>
<td>Multiview Video Coding</td>
</tr>
<tr>
<td>MVV</td>
<td>Multiview Video</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse-Code Modulation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>PERR</td>
<td>Percentage of Errored Pixels</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>RD</td>
<td>Rate Distortion</td>
</tr>
<tr>
<td>SAOC</td>
<td>Spatial Audio Object Coding</td>
</tr>
<tr>
<td>SEI</td>
<td>Supplemental Enhanced Information</td>
</tr>
<tr>
<td>TOF</td>
<td>Time-of-Flight</td>
</tr>
<tr>
<td>V+D</td>
<td>Video Plus Depth</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This chapter has the objective to introduce this Thesis, notably by describing its context and motivation, its goals and, finally, its structure.

Most of the concepts and entities, firstly introduced in this chapter, will be frequently used in the next chapters, thus having a central role for the understanding of the work developed in this Thesis.

1.1. Context and Motivation

The interest in 3D systems dates back to 1844, when the first 3D photo camera was developed: the stereoscope [1], shown in Figure 1. Since then, the interest in 3D systems has been increasing over time; this growth of interest in 3D technologies is deeply related to the obvious fact that we live in a 3D world and not in a 2D world, which is the type of visual world typically presented to the users by the current displays and systems.

Figure 1 - Stereoscope, one of the first devices which dealt with 3D content [2].
Stereoscopy or stereoscopic is any technique that can create the illusion of depth perception and, thus, a 3D perception where a third spatial dimension comes to life [3]. The most common stereoscopic techniques are characterized by providing a 3D perception using an image pair where each image is directed to one eye. The image pair presents the same scenario shot by two different angles – corresponding to the left and right human eye positions – and then providing the 3D perception through the specific processing happening in the human brain, which processes this image pair to provide the users with the depth perception of the scene. A simple way to provide the users 3D video experiences consists in showing a sequence of these image pairs which is the most commonly used stereoscopic process.

The grown interest in 3D systems has been mainly reflected in the following markets:

- **3D TV** – now available in the user’s mass market, e.g. the last World Cup (2010), which is the most watched event in the whole world, had twenty five games available in 3D [4] that were transmitted in (stereo) 3D for many countries around the world, such as Portugal, Spain, UK and USA [5] [6] [7];

- **3D Cinema** – Regular cinema had a huge impact in the current society since its starting; nowadays, 3D cinema is quickly expanding and this trend is expected to continue in the next years; a strong evidence of this boost is the announcement made by Disney that in the future all its animated movies will be released in 3D [8];

- **3D Displays** – Naturally, there are no 3D experiences without 3D displays; nowadays there are more and more 3D displays in the market such as the Hyundai 3D TV, shown in Figure 2, which still require glasses; most of these displays are able to deal with High Definition (HD) content, thus increasing the user experience quality and impact.

![Hyundai 3D TV, HP project to provide 3D from a laptop LCD display and Samsung 3D TVs](image)

*Figure 2 - Hyundai 3D TV, HP project to provide 3D from a laptop LCD display and Samsung 3D TVs [9] [10] [11].*

The new 3D displays can be classified in two main types: stereoscopic and multi-stereoscopic. While the stereoscopic displays only provide one 3D view, the multi-stereoscopic displays are capable of providing multiple 3D views in the sense that the user does not only get a depth sensation but can also navigate around the (3D) scene by changing the viewpoint he/she is consuming.

The multi-stereoscopic displays - such as the Philips 3DWOW TV which does not require glasses anymore, see Figure 3 - have the Head Motion Parallax Navigation (HMPN) functionality which enables the users to
change its physical position, parallel to the 3D display, and still have a 3D perception of the content. This is particularly interesting in the sense that multiple users can see 3D video in different positions and without glasses.

![Philips 3DWOW TV](image)

Figure 3 – Philips 3DWOW TV [12].

One application that is important for a complete 3D experience is Free Viewpoint Video (FVV), which enables the user to change position of the viewpoint interactively, i.e., as if changing position of the capturing camera. This application requires having $N$ cameras capturing the scene and calibration information to synchronize the views [13].

3D systems must be able to support 3D features and application efficiently, meaning that the tools in 3D systems are responsible for the effectiveness of the resources (namely bits). In this context, the presence of depth data – which provides information regarding the geometry of the scene and will be thoroughly described in Chapter 2 – plays a special role, since it brings two advantages:

- It strongly enhances the user’s 3D experience by increasing the number of available 3D views by synthesizing new, additional views based on the transmitted views (this process will be further explained in Chapter 2);
- It allows reducing the cost of the 3D experiences for the users and service providers since, in terms of bits, some 3D tools become more efficient (as it be again thoroughly explained in Chapter 2).

In this context, a 3D system using depth information besides the color information is able to more efficiently support the necessary 3D features and capabilities (such as HMPN), notably if an efficient depth coding solution is available.

1.2. Objectives

In the literature, some depth coding solutions have been proposed, some of them exploring new mathematical concepts in the field, as it will be shown in the next chapter, and also exploring the codecs available for traditional video data. However, depth coding is a research field with growing interest by the video coding community, due to the emerging related needs in 3D systems, and the fact that there is until now no single depth
coded largely accepted by the experts as a mature specific solution since depth properties are very specific and they have not yet been completely explored in terms of depth compression efficiency; in this context, higher compressions rations for depth coding can be achieved.

Therefore, the main objective of this Thesis is to design, implement and evaluate an efficient depth coding solution to be integrated in 3D systems such that it can:

- Contribute for a better understanding of the depth specific properties;
- Compete with already proposed depth coding solutions;
- Introduce depth coding novel tools that may lead to a better support of 3D features;
- Help to further meet the expectations of the user’s demand for more immersive 3D experiences.

In this context, the proposed depth coding solution is based on the solution in the literature which the author of this Thesis considered to be one of the most promising depth solutions available; this solution has been complemented with some novelties added by the author, based on the knowledge acquired by reviewing the state-of-the-art on this subject.

1.3. Thesis Organization

This Thesis is organized in six chapters, including this first one introducing the Thesis.

Chapter 2 defines thoroughly what depth data is and reviews the most relevant 3D systems to provide a better understanding why the presence of depth data is important in 3D systems and what type of data has to be efficiently coded.

Chapter 3 describes some of the most relevant depth coding solutions available in the literature. This description intends to highlight the strengths and weaknesses of each solution in order to reach a good understanding of the state-of-the-art in terms of depth coding.

Chapter 4 describes the adopted depth coding solution which takes into account specific depth data properties and should be able to compete with the relevant alternative depth coding solutions already available.

Chapter 5 presents the performance evaluation of the adopted depth coding solution - proposed in Chapter 4 - by using appropriate test conditions and assessment metrics to compare it with the relevant depth coding benchmarks.

Finally, Chapter 6 is dedicated to the summary of the work performed in this Thesis and to the proposal of relevant future work in the following of this Thesis.
Chapter 2

Reviewing 3D Video Systems

Since the world is a 3D world, there has been a growing interest in 3D video systems, largely motivated by the novel capabilities that acquisition, representation and display technologies are boosting and thus targeting the improvement of the multimedia user experiences.

A 3D video system must support some 3D video services and yet be efficient from the representation point of view, meaning that it should use as few resources as possible to fulfill the relevant requirements.

Both highly immersive audio and video are essential in 3D systems for a powerful user experience and, therefore, its design and optimization are central in efficient 3D systems. New audio representation techniques have been developed recently, notably by the MPEG (Moving Pictures Experts Group) Audio group. Among them, it is worthwhile to mention the so-called Spatial Audio Object Coding (SAOC) standard which is a unique tool designed to suit a wide variety of applications – from conferencing and broadcast to music remixing and even karaoke. Combining a compact representation of the audio data with user interactivity, SAOC enables highly efficient representation, transport and user-controllable rendering of individual audio input signals in a compatible mono or stereo downmix, making it particularly attractive for online music distribution or transmission over channels with limited bandwidth.

However, while the relevance of audio in 3D systems is unquestionable, this Thesis will focus on the video component which is also under intense research and scrutiny. On the visual side of the 3D user experience, depth information is playing an increasingly important role to express the scene geometry. In this context, this chapter will first introduce the reader to basic depth concepts and tools in order to understand later the impact of depth data in 3D video systems.
After, this chapter will provide an overview on the most relevant types of 3D video systems, highlighting their evolution in terms of main concepts, features, tools and, finally, advantages and limitations.

2.1. About Depth Data

The main objective of any video system is to offer the user the best possible video experience under the relevant constraints, e.g. bandwidth, computational power, etc. So far, all displays and video systems in the mass market offer 2D videos to the user and, thus, 2D experiences. However, because we live in a 3D world, the provision of 3D video improves the user video experience; for these 3D experiences, information regarding a third dimension, notably expressing the scene geometry, is needed. One way to provide this information is by using depth maps which basically carry the information regarding the third spatial axis, \( z \), beyond the usual 2D screen plane axis, as represented in Figure 4. In fact, depth perception refers to the visual ability to perceive the world in three dimensions. In this context, depth data is responsible for providing geometrical information about the scene, thus regarding the missing component in 2D images.

![Figure 4 – Depth perception.](image)

In a more precise way, a depth map is a pixel based map where each value measures the distance between the object and the camera, as represented in Figure 5; hence, the depth values range varies from \( z_{\text{near}} \) to \( z_{\text{far}} \) and corresponds respectively, to the minimum and maximum distances between the camera and the objects in the scene. Typically, each depth sample is quantized with 8 bits, i.e. the closest point is associated with the value 255 and the most distant point with the value 0 as shown in Figure 5 [8]. This association is in contradiction with the definition above since the shorter the distance the higher is the depth value; this change seems to be related to the improvement of the visual impact of depth images highlighting the objects closer in the scene.
Generally, depth maps have very smooth depth changes for each object and sudden changes in their edges. Figure 5 exemplifies well some of the depth maps main properties, notably:

- Smooth regions in the objects, e.g. in the bottle and in the glasses;
- Sharp edges between smooth regions corresponding to the objects’ boundaries, e.g. between the top of the bottle and the background.

Note that a depth image is basically an image with a scale of grays, very similar to a luminance (texture) image. This is why it is currently rather common to encode depth maps with luminance codecs, e.g. MPEG-2 Video or H.264/AVC (Advanced Video Coding), as in the MPEG-C Part 3 standard\(^1\). However, to efficiently encode depth, the codecs designed for luminance coding are not the most adequate, and thus efficient, since the two types of signals have rather different properties. The most important difference between them is related to the fact that depth images are not to be seen directly by the user, contrary to what happens with luminance signals. This has a major impact on the type of metrics appropriate to measure the quality of depth images, since the human visual system (HVS) is still a decision factor but not as directly as for the luminance; in fact, depth is important to create synthesized views from available views and, naturally, their quality to be measured through the HVS depends on the depth quality. For example, a good preservation of the depth discontinuities will lead to a better quality of the synthesized views\(^2\); thus, this should be taken into account in efficient depth coding. Next chapter will discuss the state-of-the-art on depth coding as well as the best metrics to apply to appropriately evaluate depth quality while maximizing its benefits.

**Acquisition**

So far it has been introduced the concept of a depth map, how it is defined and its properties. The following paragraphs of this section describe examples of the two most common solutions to obtain depth:

- Acquiring depth directly with a depth sensor;
- Computing depth based on a disparity map.

Depth sensors have the capability to measure the distance between the sensor and the object that is being shot. Figure 6 shows a commercial version of a Z camera\(^2\), which is a depth sensor’s type, named Zcam\(^3\). This kind of technology has seen great developments in recent years and is becoming more and more popular.

---

1. MPEG-C Part 3 standard allows coding one color video (luminance and chrominances) and the corresponding depth map using available video coding standards such as MPEG-2 Video and H.264/AVC.
2. A Z camera provides color video with real-time depth information, allowing 3D video.
Basically, this type of cameras uses the principle of time-of-flight (TOF) which measures the time that a narrow laser beam takes to be reflected at an object and then detected by the sensor to compute its distance. This allows creating distance data providing real time depth video of a scene with good quality.

Another type of depth sensor called Axi-Vision camera is shown in Figure 7. This camera is capable of acquiring HDTV (High Definition Television) color image and a depth image of objects on more than 1280x720 pixels at a frame rate of 29.97 Hz. The principle to measure the distance between objects and the camera is also the TOF principle because this method has the advantage of quick and straightforward information processing.

Summarizing this technology main advantages and disadvantages; depth sensors are at great development in the recent years and are already capable of providing depth maps in real time and with good quality. However, and as any technology under great development, more time is needed to mature this technology and fully understand its capabilities and limitations.

There is another solution to acquire depth, which is by computing depth through the so-called disparity map. A disparity map is a pixel based map which each sample value corresponds to the distance between corresponding pixels or features in stereo images, this means two images with different viewpoints. Figure 8 shows the example of a stereo pair capturing which is commonly used to generate a disparity map and, therefore, a depth map since they are analytically related and, thus, it is possible to create one from the other.
Using stereo vision from two different cameras separated by a distance called baseline, it is possible to estimate the depth following three main steps [19] [20]:

- Establish correspondences between the two images, e.g. based on salient features;
- Compute the displacements (disparity) corresponding to the Euclidian distance between corresponding object pixels in the two images;
- Estimate the depth relative to the cameras based on the epipolar geometry concept which requires knowledge regarding the geometry (position) and focal length of the cameras.

The algorithms to estimate depth from a stereo pair, such as feature-based or local intensity-based methods [22], can be highly complex and also error prone, namely when there are no correspondences between the two images for some objects or for large stereo baseline distances [23]. To reduce the errors in this process, a solution can be implemented by adding more capturing cameras in order to provide more information about the scene and, thus to create more accurate correspondences between views; however, the complexity once again increases.

The reason why in the literature depth maps are often associated with disparity maps is because the two concepts are closely related and it is possible to derive one from the other under some specific conditions. The relationship between them depends on the focal length of the cameras and on the geometry of the cameras [24].

Summarizing this technology main advantages and disadvantages; computing depth based on a disparity map it is usually done by error prone and highly complex processes which represents a problem regarding the final quality of depth maps. This solution avoids the usage of new devices, such as depth sensors, to capture depth; allowing a more smooth transition to its integration on conventional acquisition systems.

2.2. Relevant 3D Video Systems

A 3D video system is a system able to provide to the user some type of 3D feeling and experience. The various types of 3D systems available are closely related to the data representation approach adopted; which determines the information processing chain and also the functionalities provided and the data coding.
requirements. The basic 3D video system processing chain is shown in Figure 9, where each module has the following functions:

- **Acquisition:** This module is associated to some device(s) capturing the view(s) data for the relevant scene, notably the color video (luminance and chrominances) signal and possibly depth video signal;

- **Post-Processing:** The data captured may have to be processed in some way, e.g. noise reduction, to provide the encoder the type of information and the associated format it is expecting;

- **Encoding:** This module changes the data representation from a raw, very inefficient format into a much more efficient format; notably, in terms of representation resources, this means compression efficiency. However, requirements such as random access, scalability and error resilience may be relevant depending on the application;

- **Storage or Transmission:** The data resulting from the encoding process can be stored e.g. in a hard drive, or transmitted through a channel until it reaches its destination.

- **Decoding:** This module reconstructs the view(s) based on the data produced by the encoder in a way that the display can interpret it. Error concealment techniques are applied when errors introduced by the channel are detected.

- **View Synthesis:** This module enables the synthesis of new views only when depth data is also available besides the color video, based on the information gathered by the decoder. According to the application requirements, different algorithms are applied to create new views. Some algorithms can be found in [19].

- **Display:** Where the video views are finally displayed, for example using a mono or stereo approach.

![Figure 9 - Basic 3D video system data processing chain.](image)

Since there are currently several relevant types of 3D video systems, the following sections will review the most relevant 3D systems by increasing order of complexity. This reviewing will mainly focus on data coding issues since this is the main topic of this Thesis.
2.2.1. Stereo Video Systems

**System Data**

A Conventional Stereo Video (CSV) system is a type of 3D video system based on the transmission of two color video signals, each one targeting one eye, see Figure 10. The color signals in this Thesis are considered to be composed by one luminance and two chrominance signals.

![CSV system data: two color signals](image)

*Figure 10 - CSV system data: two color signals [25].*

The two color signals can be captured by two cameras or by a single camera able to capture two video signals at once. An example of how to capture two views with a single camera is shown in Figure 11; this camera has been developed for the very popular 3D movie, *Avatar*.

![Stereo camera developed for the Avatar movie](image)

*Figure 11 - Stereo camera developed for the Avatar movie [26].*

**Data Representation**

In the Telecommunications business, it is important to use the smallest possible amount of resources to reach the defined targets, in this case to represent the stereo data to be transmitted or stored. Therefore, to be efficient in stereo representation, it is necessary to exploit the effects that could not be exploited in regular 2D video
systems, notably the redundancy between stereo views. The exploitation of the redundancy between two correlated views of the same scene (for the same time instant) is typically performed in the same way as the exploitation of temporal redundancy between two frames of the same view. While this process corresponds to temporal prediction with motion estimation in terms of temporal redundancy, it corresponds to view prediction with disparity estimation in terms of interview redundancy; the well known motion vectors in video coding correspond to disparity vectors in stereo coding. Figure 12 show an example of a stereo prediction structure where both interview prediction (one view is predicted from the other exploiting the redundancy between the two view) and temporal prediction (one frame is predicted from one or two frames in the same view) are present.

Looking to Figure 12, it is possible to see that the frames from the left view are used to predict frames in the right view; however, the opposite is not true allowing this prediction scheme to provide backward compatibility to monoview systems which means a monoview system may simply decode the left view.

Figure 12 - Combined temporal and interview prediction for stereo video [8].

The coding process mentioned above has been adopted for stereo video coding in standards such as MPEG-2 Video and MPEG-4 Visual (Part 2). In November 2009, a stereo Supplemental Enhancement Information (SEI) message was added to the latest and most efficient H.264/AVC video coding standard to deal with stereo coding. The objective of these SEI messages is to insert extra information into the bitstream, in this case related to the second video channel, to enhance the use of the video for a wide variety of purposes usually with the final objective of sparing resources.

Functionalities and Features

The output of any 3D system is clearly limited by the type of input information selected. The stereo approach is indeed a good example of these limitations. The most reliable information that can be expected at the output is the one captured by the cameras, meaning that basically only two color signals are available; thus, two different views are expected. Although in theory it is possible to estimate depth at the decoder by computing depth based on the disparity map produced by these two views (as mentioned in the previous section), this process is highly complex and rather inaccurate since the estimation would be (very likely) based on lossy coded images. This rigidity is a problem regarding the support of 3D features such as HMPN; since it is rather expensive to provide more 3D views, it is assumed that only one stereo pair is available; therefore, only one 3D view is available too.

Another major drawback, probably the most important, is the fact that it is not possible to easily adapt the data to get depth perception with different display types and sizes [8]. All these drawbacks together make this approach rather limited if a 3D video system supporting a large range of 3D applications and able to be used for different types of displays is needed.
2.2.2. Multiview Video Systems

**System Data**

To support a larger range of 3D applications, Multiview Video (MVV) systems accommodate more than two color signals meaning that this type of solution is clearly an extension of the above CSV systems by considering more input views. For example, Figure 13 shows a situation where five color signals from different views are taken as the MVV system input to be encoded and, eventually, transmitted. This choice will naturally result in a major rate increase for transmission or storage.

![MVV System Data: Multiple Color Signals](image)

*Figure 13 - MVV system data: multiple color signals [25].*

**Data Representation**

MVV systems have much more scene information available than CSV systems, provided by the increased number of views; therefore, it is expected to provide better results, notably in terms of the final 3D video experience provided to the user.

As for stereo coding, efficient MVV coding has to exploit the available interview redundancy; otherwise, the high number of views will lead to an unacceptably high coding rate.

The first MVV standard coding solution has been defined in the context of the so-called Multiview profile in the MPEG-2 standard; however, for several reasons, this solution has never been much used in practice.

More recently, the Multiview Video Coding (MVC) standard, which is an extension of the H.264/AVC standard, released in January 2008, targeted the efficient coding of multiview video; more specifically, this coding standard targets the efficient representation of N input views with a quality depending on the rate invested [27]. Although the total rate for this approach is naturally higher than for CSV systems when the number of views is higher than 2 and for the same conditions, the rate cost per view may be lower since there is very likely a higher correlation/similarity between the views, thus leading to a higher compression ratio. Figure 14 shows an example of a MVC prediction structure.
The red arrows in Figure 14 represent the interview predictions while the black arrows represent the temporal predictions, e.g. the second B frame of camera two will be predicted based on the second B frame of cameras one and three (interview prediction) and based on the first and third B frames of camera two (temporal prediction).

![Figure 14 – Example of MVC temporal and interview prediction structure [27].](image)

Following the example described earlier, and taking into account the inputs for this MVV approach, the outputs are also five color videos which can provide four different 3D views, as shown in Figure 15.

![Figure 15 - Example of outputs and 3D views for a MVV system.](image)

A trade-off of the distance between 3D views and the feeling of a sudden change between views is typical of MVV systems; e.g. in order to make a smooth transition between views, the cameras must be close enough in order the HVS cannot notice the change between 3D views, therefore the 3D views must be close to each other, limiting the distance between the position where the user may see the first view and the position where the user may see the fourth view. So, regarding 3D video features requiring a continuum or a very large number of 3D viewpoints, they are typically not supported unless some additional views are synthesized at the receiver again at the cost of high complexity and error prone processes. Naturally, the more views are coded, the more information is available about the scene geometry and, thus, the better may be the quality of the additional synthesized views.
**Functionalities and Features**

In summary, there is a main problem regarding the support of 3D features such as and HMPN since MVC does not directly support a continuum of views and, thus, it may be too expensive in terms of rate to increase the number of inputs; naturally, additional views may always be synthesized from the available views with more or less quality, e.g. depending on the quality and ‘distance’ between the available views.

As for the CSV case, it is difficult to adapt the depth perception to different display types and sizes due to the associated complexity [8]; however, this process should provide better results considering more scene information is available through the multiple views.

### 2.2.3. Video Plus Depth Systems

**System Data**

Since the transmission of multiple views, notably to provide a smooth transition between views, may be rather expensive in terms of total rate, a new type of 3D video system has emerged trying to make available multiple 3D viewpoints based on the effective representation of a much smaller number of input views.

Video Plus Depth (V+D) systems are 3D video systems based on information associated to a single view as shown in Figure 16; however, this type of system requires a single input view consisting on not only the corresponding color signal but also the associated depth map.

![Figure 16 - V+D system data: one color and one depth signals [25].](image)

**Data Representation**

The addition of depth maps as inputs in the V+D approach changes the data processing chain when compared with the previous 3D systems types since now a View Synthesis module is required. The View Synthesis module is responsible for creating additional output views based on the available decoded view and its associated depth map. 3D warping algorithms are usually applied to create those views, e.g. planar-to-planar warping algorithm [28].

As mentioned before, depth and luminance signals use a grey scale and, therefore, depth can be fed into the luminance channel of a video signal (eventually setting the chrominance to a constant value) and then be coded...
by any state-of-the-art video codec such as H.264/AVC; this is basically the solution adopted in the MPEG-C Part 3 standard. This process typically results in a compression at 10-20% of the bit rate needed to code a video view while still providing a good quality for the rendered views; this shows that it is more efficient to transmit or store depth data than a color signal, therefore, making the view synthesis approach vital for saving resources in a multiview context [8].

**Functionalities and Features**

The information carried out by the depth map brings to the V+D systems two major improvements when comparing to previous approaches:

- Rendering of a continuum of output views although within a rather limited view angle;
- Reduction of the overall bit rate for the same number of 3D views.

Since the depth map provides information about the scene geometry, this type of 3D system allows the stereo impression to be adjusted and customized after transmission [8], thus enabling the depth perception to be adapted to different display types and sizes, including multiview displays. Moreover, the *View Synthesis* module allows synthesizing multiple new views with less complex and less error prone processes thus supporting HMPN in a smooth fashion within a practical short range since *View Synthesis' artifacts* may dramatically increase with the distance of the synthesized to the reference views [8]. In theory, it is possible to extract depth from video e.g. the two views in a CSV system could be used at the decoder to extract depth maps and after generate more video views. However, and as mentioned in the previous section, this solution is practically less interesting since extracting the depth data from decoded video leads typically to a depth quality which is not of much usage in terms of minimum quality of the synthesized views and its associated processes are highly complex. This paved the way to encoder side depth map capturing or extraction based on original video.

### 2.2.4. Multiview Video Plus Depth Systems

**System Data**

This type of 3D video systems, known as MV+D or MVD, can be understood as an extension of the V+D system in terms of the number of inputs views. As illustrated in Figure 17, and following the example previously used, the MV+D system considers multiple color videos and their associated depth maps; naturally, there are not only more views available but the system has also globally a much better knowledge on the scene geometry.
Data Representation

This 3D system, when comparing to the previous 3D systems, has a better knowledge regarding the geometry of the whole scene at the expense of an increase of data to be stored or transmitted. This increase of data will eventually increase the number of 3D views significantly; however, the question regarding the efficiency of this process is legitimate. Therefore, to show that 3D video systems using depth data are more efficient than those not using it, follow this example: Assume that an available 3D display is able to show nine different 2D views (V1 - V9) simultaneously as in Figure 18. From a specific position, a user can only see a stereo pair of the eight possible 3D viewpoints, e.g. in position 1 (Pos1), the user may see the views 1 and 2 (V1 and V2) which provide one 3D view and so on. Encoding and transmitting these nine views directly, e.g. using the MVC standard, would require a rather high rate; for example, assuming one single 2D view composed by a single color video signal requires about 3 Mbit/s for the chosen resolution using the H.264/AVC standard, a MVC solution would require $9 \times 3 \text{Mbit/s} \times 0.5 = 13.5 \text{Mbit/s}$ assuming that the MVC is 50% more efficient than the independent H.264/AVC coding of each 2D view. Assuming an alternative MV+D solution with only three 2D views V1, V5, and V9 each one composed by a color signal and its corresponding depth signal, there is also the need to code the depths D1, D5, and D9 since the remaining 2D views can be synthesized by a depth image based rendering (DIBR) process as shown in Figure 18. Assuming that the depth can be compressed at 20% of the bitrate necessary to encode a color video, while still providing good quality for the rendered 2D views, and still assuming that one single color video signal requires 3 Mbit/s, then the MV+D solution would require a rate of $3 \times 3 \text{Mbit/s} \times 0.5 = 4.5 \text{Mbit/s}$ for the color video signals and $3 \times 0.2 \times 3 \text{Mbit/s} = 1.8 \text{Mbit/s}$ for the depth video signals. In summary, the total rate is $6.3 \text{Mbit/s}$ for the MV+D case which is less than half the required rate for the MVC solution (13.5 Mbit/s).
Functionalities and Features

In V+D systems, HMPN is supported but within rather short ranges since View Synthesis artifacts may dramatically increase with the distance to the newly synthesized viewpoint. To overcome this problem, MV+D systems use more than one color signal as well as the associated depth signals and, therefore, may better support HMPN.

Generally, it is convenient to use depth maps in 3D systems because they have the necessary scene geometry information to better support multiview displays and 3D applications, notably in a more rate efficient way. Particularly when many 2D views are required at the display, depth based 3D systems are able to provide a much more efficient solution in terms of coding rate even though at the cost of some additional complexity. This increased complexity may become a problem, notably for small devices such as mobile phones. To overcome this problem, there are two main solutions:

- **Implement a depth quality scalability coding solution** meaning that the depth information is sent by layers and each layer level contains more accurate depth data;

- **Use a less complex codec**, which will very likely be less efficient, proposing a different efficiency-complexity trade-off.

Another scalable depth solution, but different from the previous, is the Layered Depth Video (LDV) approach; typically, the data inputs in this type of system are two layers:

- **Main layer**: composed by one view’s color signal with its associated depth signal, just like in the V+D approach;

- **Background layer**: also composed by a color signal with its associated depth map; however, and as shown in Figure 19, these signals have information regarding the occluded areas by the foreground objects in the main layer.
The LDV approach supports rendering virtual views and, therefore, multiview displays and 3D applications as well. Although LVD might be more efficient than MV+D because less data may have to be transmitted, and thus also less decoding complexity may be needed, it is also possible that additional error resilience capabilities are needed since the processing tools have to operate on less reliable depth data. Finally, it is possible to note that LVD can be generated from MV+D by warping the main layer view onto other contributing input views because the occluded areas by the objects of a central view are usually available at the outermost views of the same scene.

Regarding a less complex depth coding solution and the possible compression efficiency-complexity-quality trade-offs, these will be the focus of the next chapter.

2.3. Summary

The main goal of this chapter was to review the main types of 3D video systems with emphasis on the depth related aspects. The role and impact of depth in 3D video systems was analyzed, showing that properly used depth offers an efficient solution to:

- Render a continuum of output views;
- Reduce the overall bit rate;
- Support a large range of multiview displays and 3D features.
After reviewing the main types of 3D video systems, the conclusion was that the MV+D approach is the most complete and, therefore, the one maximizing the 3D video user experience for a limited, acceptable rate. However, coding depth with a luminance codec, e.g. the H.264/AVC standard, is not a good depth coding approach since it does not take into account any depth specific features and requirements. It is always important to remind that depth is not directly seen by the users and, therefore, direct HVS considerations including PSNR (Peak Signal to Noise Ratio) optimizations may be inappropriate. Next chapter is precisely dedicated to review depth coding concepts, coding tools and performance.
Chapter 3

Reviewing Depth Coding Techniques

Depth data may need to be involved in the 3D video content representation process and thus efficiently coded to be transmitted or simply stored. Since depth data has different properties than regular video, it has to be treated accordingly, meaning that already existing state-of-the-art video codecs may not be appropriate for such type of data. Moreover, depth data is not directly consumed by the user as regular video, so new methodologies and metrics are needed to optimize and assess the depth coding performance. This chapter has the main objective of reviewing the current state-of-the-art on depth coding by presenting the most relevant solutions and also reporting their performance.

3.1. H.264/AVC Depth Coding

Among the available video codecs, MPEG and ITU-T standards play an especial role, with the most efficient video coding standard available being clearly the H.264/AVC standard [29]. In this context, the depth coding approach presented in this section corresponds to the simple adoption of the best available video codec, which is mainly lossy although – under certain conditions – it can achieve lossless quality. In this way, depth coding does not introduce any novelty burden, exploiting the existing state-of-the-art 2D video coding technology to encode both video and depth sequences without any additional tools.

3.1.1. Basic Technical Approach

A depth map is a 2D representation of the depth of a scene; however, it may be taken as a texture/luminance map by setting the chrominances to a constant value and coded using any available video codecs which are able to code luminance data. The H.264/AVC standard was jointly developed by the MPEG and ITU-T standardization bodies and has a special relevance because it provides the same quality for approximately half
the rate, thus doubling the compression factors, regarding previously available standards. In this context, the first depth coding tool to be reviewed in this Thesis is precisely the H.264/AVC standard when it gets as input a depth map sequence to be coded as a luminance sequence. In the multiple views case, multiple depth maps are available and may be coded with the MVC standard which is an extension of the H.264/AVC standard able to exploit the interview correlation, in this case the inter depth map correlation.

As mentioned above, the coding approach here it to treat the depth maps as a luminance signal since depth images are ‘monochromatic’ too. Naturally, the decoder must be informed that a depth map is being transmitted along with the usual color video. To make available a standard format following this approach, MPEG has specified, in 2007, a container format, known as MPEG-C Part 3 but formally named as “ISO/IEC 23002-3 Representation of Auxiliary Video and Supplemental Information” [30]; this standard format defines auxiliary video streams and specifies its syntax and semantics according to the ‘depth taken as luminance’ approach. The only additional information regarding depth that needs to be transmitted is the depth range ($z_{\text{near}}$ and $z_{\text{far}}$) since this is essential for any receive usage of the depth data. This standard enables rather efficient coding of the 3D video data for V+D systems [8].

3.1.2. Architecture

The H.264/AVC standard is a predictive video coding standard mainly based on the combination of a spatial transform with quantization, temporal prediction with 1/4 pel motion compensation and entropy coding. The general architecture of the H.264/AVC standard is presented in Figure 20.

![H.264/AVC encoder architecture.](image)

The depth coding solution considered in this section is very simple and it is illustrated in Figure 21; basically, depth information are coded with H.264/AVC, meaning that this coding solution does not acknowledged in any way the specific coding characteristics of depths maps and simply consider they are similar to a luminance map. Since the depth maps are treated as luminance maps, all coding tools used for the luminance are also used for the depth.
The next section describes some of the tools which made H.264/AVC codec so popular.

3.1.3. Main Coding Tools

H.264/AVC has a compression gain of 50% when compared with previous video coding standards like H.263, MPEG-2 Video and MPEG-4 Visual. Although many technical novelties were introduced in H.264/AVC, those mainly responsible for doubling the compression gain are [31]:

- **Variable (and smaller) block size motion compensation** – Allows to more efficiently explore the temporal redundancy, providing better predictions and resulting in a decrease of bit rate for the same quality;

- **Multiple reference frames** – This feature is especially useful for cyclic content, where a frame still on the memory can provide a particularly good estimation for a new frame with big similarities with a frames in the past although not necessarily the most recent;

- **Hierarchical transform with smaller block sizes** – An exact-match integer 4×4 spatial block transform, allowing precise placement of residual signals with little of the "ringing" often found with prior codec designs; a second transform layer corresponding to a secondary Hadamard transform is performed on the DC coefficients of the primary spatial transform applied to luminance to obtain even more compression in smooth regions;

- **Deblocking filter in the prediction loop** – Besides improving the subjective impact in terms of removing the block artifacts, this filter is also able to reduce the residues after the motion compensated prediction due to the fact that it smooths the block edges;

- **Improved, adaptive entropy coding** – A solution based on Context-based Adaptive Binary Arithmetic Codes is available leading to a 5-15% reduction of the bit rate regarding the alternative non arithmetic coding based solution.

In particular, some of these tools boosted the efficiency of the temporal prediction and thus, the efficiency of the codec in general.
3.1.4. Performance Evaluation

Two test sequences are shown in Figure 22: “Homeshopping” and “Breakdancers” which were coded using the H.264/AVC. While “Homeshopping” has 100 frames each with 720×480 pixels, “Breakdancers” has 100 frames each with 1024×768 pixels.

![Figure 22 – “Homeshopping” (left) and “Breakdancers” (right) color (top) and depth (below) test sequences [32].](image)

The correspondent rate-distortion (RD) performance is shown in Figure 23, which was obtained with the JM 9.6 reference software when coding the depth data for the two test sequences. As expected, the depth PSNR in dB grows linearly with the rate.

![Figure 23 – H.264/AVC based depth coding results [33].](image)

By considering this as a rather basic coding solution, since only are exploited available luminance tools and not considering any depth specific features, it justifies why this solution is commonly used as benchmark to assess the RD performance gains when using depth specific coding solutions. As mentioned before, this is not an
adequate performance assessment approach since depth maps are not directly visualized and thus it does not make much sense to optimize and assess their coding as for the luminance, e.g. using the $\text{PSNR}$.

### 3.1.5. Strengths and Weaknesses

Concerning the coding process, one of the main strengths of this approach is the fact that powerful spatial and temporal correlation exploitation tools are used; on the other hand, these tools do not consider in any way the depth specific properties which may mean that more efficient tools may be used. The fact that this codec only exploits texture properties instead of depth properties may result in ringing artifacts along the depth edges, leading to errors in depth positions which may be especially critical in terms of strong visual synthesis artifacts.

On a different perspective, this approach may be beneficial from an economical point of view due to its compatibility with already implemented video coding systems; thus, no radical technological investment would be needed to introduce this solution in the market.

### 3.2. Hierarchical Decomposition Mesh-based Depth Coding

In [33], Kim and Ho have proposed a lossy depth coding solution based on a hierarchical decomposition of the depth map, i.e., decomposing a depth map four different kinds of images. At the decoder’s side, these four images are able to reconstruct a depth map which - through an interpolation technique that was not described - generates a mesh. After the hierarchical decomposition, three of these four images are merged into a single image; then, the two resulting images are coded with $\text{H.264/AVC}$.

#### 3.2.1. Basic Technical Approach

Most of the complexity of this approach is at the hierarchical decomposition, which generates three disjointed images and a layer descriptor. These four images are the following and have the respective function:

- **Regular Mesh (RM)** – provides general information of the depth map content;
- **Edge-Region (ER)** – provides more detailed information regarding the regions with edges in the depth map;
- **No-Edge-Region (NER)** – provides more detailed information regarding the region without edges in the depth map;
- **Number-of-Layer (NOL)** – provides information to manage the three disjointed images (which are the previously thee described images).

The three disjointed images have very specific information directly related to the depth map itself, since their pixels are feature points of the depth map, i.e., depth samples that critically influence the generation of the mesh representing the depth map; thus, they have smaller dimensions than the depth map and therefore they are be merged into a single image.

The information provided by the NOL and the three disjointed images is able to generate a triangular mesh. The advantage of the mesh-based representation is the high rendering speed which allows enjoying real time 3D video. However, it is challenging to compress the feature point data due to its irregularity. One solution to remove this irregularity is to regard all depth information in a depth map as features points at the cost of missing
real-time 3D video because a huge number of triangles must be generated. Another solution is to apply a 3D warping technique, although the absence of good hole-filling algorithms and more stable warping systems represents a major problem to its implementation. Therefore, Kim and Ho decided to employ a hierarchical decomposition.

### 3.2.2. Architecture

The architecture of this depth coding approach is represented in Figure 24 which has as input the depth video. Two phases can be noticed in this architecture, notably: the hierarchical decomposition and the 3D video compression described in the following. Note that some of the modules do not correspond directly to a process but to types of data too; notably, the RM Images, ER Images, NER Images and NOL images are types of data and not processes.

![Hierarchical decomposition depth-based 3D Video coding architecture](image)

**Figure 24 – Hierarchical decomposition depth-based 3D Video coding architecture [33].**

- **Hierarchical Decomposition**

  1. **Downsampling** - The depth image is downsampled according to the predefined grid cell size, which is the decomposition unit, and thus generating a RM image. At the decoder’s side, the RM generates a so-called regular mesh with the same dimensions of a full depth map though an interpolation technique (which was not described in [33]);

  2. **Edge extraction** - The depth pixels at the edges of the image are extracted in order to create the ER images and the NER images. The edge extraction is made by applying a Sobel filter to the depth image.

After those two previously described processes, a NOL image is employed to manage the hierarchical decomposition of the depth maps, i.e., they have information regarding the three disjointed images and the grid cell size. The pixel intensities in these images contain the information regarding the number of feature points and the ER images modes (those modes will be described in the next section).
• 3D Video Compression

1. **Data Aggregation** - To prevent low coding efficiency due to the scattered pixels in the NER images, these depth pixels are aggregated in a way that the decoder can recover their original positions;

2. **Entropy Coding** - NOL images are directly entropy encoded;

3. **Image Merge** - ER, NER and RM images are merged into a single image as shown in Figure 25;

4. **2D Video Coder** - The resulting two images, i.e. the merged and the NOL images, are encoded using H.264/AVC.

![Figure 25 – Example image merging result [15].](image)

Although this depth coding solution does not have a codec focused in depth data to lead with depth data, it has changed the depth data representation into a different representation that, hopefully, will be more adequate for the H.264/AVC coding process than depth by itself.

### 3.2.3. Main Coding Tools

This section intends to provide some more details for the most important coding tools in this depth coding solution; however, just the edge extraction module will be described since the author of this Thesis has not found enough details in [33] to be able to describe the other tools in present depth coding solution. Furthermore, there are no references available regarding the coding process of this depth coding solution.

**Edge Extraction**

Extracting edges is one of the most important coding tools in this depth coding solution; as shown in Figure 24, the edge extraction module has the task to create the ER images and also the NER images. However, the description of this module will focus on the ER image generation since it is more complex than the NER image generation.

1. **Detecting Edges** – The Sobel filter is applied vertically and horizontally into the depth map in order to extract the edges;

2. **Depth map Partitioning** – The depth map is partitioned accordingly to the grid cell size;
3. **Region Analysis** – All partitions are analyzed independently. If a partition does not have edges, it is considered as a NER. Otherwise, it is defined as ER and a modeling technique is employed to extract the feature points of the ER;

4. **Modeling Modes for ER** - According to the distribution of the ER in the grid cell, one of the five modes shown in Figure 28 is applied.

![Figure 28](image_url)

*Figure 28 – ER images modeling modes for a 5x5 grid cell size [34].*

The black dots in Figure 26 represent the feature points which contain the information regarding the region of edges. Generally, feature points are depth pixels which can create mesh. Consequently, the modeling modes are chosen accordingly to the zone where the depth points are located: e.g., if more than half of the grid cell has depth points regarding the region of edges, the full modeling mode in d) Figure 26 is applied to avoid losing important information; otherwise, one of the other modes is used;

5. **Generating the ER image** - The generation of an ER image is made using those previously described feature points along the raster scanning order as shown in Figure 27. This process was not further described by the *Kim* and *Ho* and no references about it were found.

![Figure 27](image_url)

*Figure 27 - Generation of an ER image [34]*
An example of the final representation of an ER image may be seen in Figure 28.

![Figure 28 - ER image (left) [25].](image)

The next section is dedicated to the performance evaluation of the proposed depth coding solution by Kim and Ho.

### 3.2.4. Performance Evaluation

Kim and Ho have conducted performance assessment experiments with two test sequences, notably: “Homeshopping” and “Breakdancers” which have the same resolutions described for the previous depth coding solution. The grid cell size was defined as 8×8. To encode the decomposed images with the H.264/AVC standard, the JM 9.6 reference software has been used.

For “Homeshopping”, one RM image, one ER image and one NER image have been generated for each frame; the resolution of the “Homeshopping” merged images is 270×60 pixels. For “Breakdancers”, one RM image, two ER images and one NER image have been generated for each frame; the resolution of the “Breakdancers” merged images is 512×96.

For benchmarking, the depth solution describe in this section has been compared with the H.264/AVC coding approach, which conceptually regards all depth information as feature points - i.e. all depth samples are coded instead of choosing just the more influent depth samples in a depth map - and to the Grewatsch approach which uses MPEG-4 3D mesh coding (3DMC) to code the depth data [35]. The results shown in Figure 29 allow concluding that the H.264/AVC approach (in red) has the best coding efficiency followed by the proposed scheme (in blue) and, finally, the MPEG-4 3DMC approach (in green). H.264/AVC performs better than the proposed scheme because the depth maps spatial and temporal correlations were higher than the decomposed images even thought the resolution of the decomposed image is much smaller.
Kim and Ho stated that a mesh based depth coding solution may provide high rendering speed; however, the main focus of this Thesis is depth compression; therefore, the study regarding the rendering times not relevant.

### 3.2.5. Strengths and Weaknesses

This approach - because uses conventional 2D video codecs - is compatible with already implemented 2D video systems, thus reducing the initial investment for the deployment of this technology. Furthermore, the coding performance for the scheme proposed in this section is slightly worse than the coding performance of H.264/AVC, since depth have very high temporal and spatial correlations.

Note that depth maps are decomposed into four new images, each one with different properties, and these new images are coded with an encoder that mainly takes into account visual properties despite these new images are not to be seen by the users, leading to the conclusion that this encoder is not optimized to encode these images. Thus, a better compression ratio can be expected if an encoder taking these image’s properties into account is used, although at the cost of losing compatibility with already implemented 2D systems.

### 3.3. Platelet-based Depth Coding

Platelet-based coding schemes try to take advantage of depth properties by selecting different modeling functions chosen accordingly to the already described depth properties.

#### 3.3.1. Basic Technical Approach

In [36], Merkle et al. propose a novel lossy depth coding approach taking into account an important depth property which is the fact that smooth regions are delineated by sharp edges. Therefore, the algorithm models smooth regions with a piecewise constant or a linear function and sharp edges with a straight line. Moreover, a quad-tree decomposition is employed to divide the image into blocks of variable size to define the area of support of each modeling function. The quad-tree subdivision and the selection of the modeling functions for each region are optimized such that a RD criterion is satisfied.
3.3.2. Architecture

The architecture of the platelet-based depth coding scheme is showed in Figure 30. The dark arrows are associated to the depth data processing flow while the blue arrows are associated to the processing control.

![Quad-tree decomposition example](image)

*Figure 30 - Platelet-based depth coding architecture.*

The walkthrough for the proposed depth coding algorithm is as follows:

- **Quad-tree decomposition** - The depth image is recursively decomposed in four equal depth block until no more decomposition are possible, i.e. a full quad-tree decomposition. In this case, each block is a parent node and its four sub-blocks are its four children as shown in Figure 31 (this is not a full quad-tree decomposition example).

![Quad-tree decompositions example](image)

*Figure 31 - Quad-tree decompositions example [31].*

- **Modeling function selection** - A modeling function is chosen for all nodes in the quad-tree, according to the content within each node; this module will be presented with more detail in the next section;

- **Quantization** - The modeling functions’ coefficients are quantized;

- **Tree-pruning and decision** - A tree-pruning technique is applied to each parent node and its respective four children, to choose the option leading to the lowest coding cost. The tree-pruning process is
repeated for each quantization value; thus producing a different tree for each quantization value. Finally, the tree leading to the smallest coding cost is selected to be coded;

- **Entropy coding** – Entropy coding is implemented to reduce the redundancy between nodes in the tree, thus improving coding efficiency; more details about this process can be found in [37].

The next section is dedicated to the most relevant coding tools in the present depth coding solution.

### 3.3.3. Main Coding Tools

This section intends to provide some more details for the most important coding tools in this depth coding solution.

**Modeling Function Selection**

The most novel module when compared to the previously presented depth coding schemes is clearly the *Modeling Function Selection*. With the purpose of exploiting more efficiently the depth properties, two classes of modeling functions are used: a class of piecewise constant functions and a class of piecewise linear functions. Consequently, four modeling functions are available for each node in the quad-tree:

- $f_1$ – The node content is approximated by a constant function;
- $f_2$ – The node content is approximated by a linear function;
- $f_3$ – The node is subdivided into two regions separated by a straight line and each region is approximated by a piecewise-constant function (known as a *wedgelet function*);
- $f_4$ – The node is subdivided into two regions separated by a straight line and each region is approximated by a piecewise-linear function (known as a *platelet function*).

The example in Figure 32 shows four squares associated to four regions with different properties.

![Figure 32 - Different modeled regions.](image)

Regarding the example in Figure 32, assuming that these virtual colored squares correspond to a block obtained by the quad-tree decomposition, they need to be modeled; in this case, the blue square would be modeled by $f_1$, the red square by $f_2$, the green square by $f_3$ and the yellow square by $f_4$. This is true if also assuming that the background and the bottleneck are almost constant regions and the table and the rest of the bottle are smooth regions which seems reasonable in this example. In practice, the selection of modeling functions is based on a RD decision criterion.
3.3.4. Performance Evaluation

*Merkle et al.* conducted some performance assessment experiments for two MV+D test data sets, notably “Breakdancers” and “Ballet”. Figure 33 shows a sequence of depth images corresponding to the “Ballet” test sequence.

*Figure 33 – “Ballet” test sequence: depth video.*

RD performance for the proposed depth coding scheme and straight *H.264/AVC* coding using only I frames are shown in Figure 34. It is possible to conclude that only for very low bit rates the platelet-based coding approach has a similar or better Rd performance than the *H.264/AVC* approach; otherwise, *H.264/AVC* outperforms the platelet-based coding approach.

Since a better *PSNR* for the depth maps does not necessarily imply a better image rendering result, a new study should be made to analyze the rendering results obtained with the coded depth maps. Therefore, a series of virtual views along the camera path were rendered. These experiments were made for the RD points marked in Figure 34, where both depth coding solutions produce the same bit rate. The results are shown in Figure 35 and in Table 1 and lead to the conclusion that, although *H.264/AVC* outperforms the platelet-based approach regarding the PSNR quality of depth images, the quality of the rendered images using the Platelet-based approach is better than using *H.264/AVC*. This conclusion results from the fact that depth properties were better explored in the novels approach and the *PSNR* metric is not able to express this effect.
Figure 34 - RD curves for “Breakdancers” and “Ballet” depth sequences [36].

Figure 35 – PSNR rendering results for “Breakdancers” and “Ballet” [36].
Table 1 - PSNR differences for virtual view rendering with low, middle and high bitrates between H.264/AVC Intra and platelet-based depth coding [36].

<table>
<thead>
<tr>
<th>Bit rate [kbps]</th>
<th>Breakdancers</th>
<th>Ballet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>150</td>
<td>400</td>
</tr>
<tr>
<td>ΔPSNR [dB] Coding</td>
<td>0.80</td>
<td>-1.04</td>
</tr>
<tr>
<td>ΔPSNR [dB] Rendering</td>
<td>0.90</td>
<td>-0.13</td>
</tr>
</tbody>
</table>

The next section is dedicated to the analysis of the positive and negative effects of the present depth coding solution.

3.3.5. Strengths and Weaknesses

This depth coding approach seems to consider more the depth properties than the previous ones; interestingly, this is shown through the rendered images and not directly through the depth RD performance which highlights the issue regarding the best way to evaluate the performance of a depth coding solution. However, due to its high complexity, this depth coding solution still cannot be used for real time operation [36]. Concerning the compatibility, this new approach is not compatible with any already available coding systems since new concepts and tools are used.

3.4. Content Adaptive Mesh-based Depth Coding

The preservation of depth discontinuities is essential to reach the high quality view synthesis necessary for advanced 3D applications. Ideally, the sample rate for depth maps should be high near the edges and low in the smoother regions. The depth coding solution to be described in this section uses the concept of content adaptive meshing to take advantage of this fact.

3.4.1. Basic Technical Approach

In [38], Sarkis, Zia and Diepold propose a lossy image codec based on the content adaptive mesh-based depth coding scheme, which takes the challenge to define the non-uniform sampling process for a depth image, in order to reconstruct the depth content with high quality.

In this context, depth maps are adaptively divided into an irregular mesh which has the structure of a binary triangular tree, the so-called tritree or simply trie. In the edge regions, the number of triangles generated by the mesh is higher since the sampling rate is higher too when compared to the smoother regions. The nodes of the mesh must be able to interpolate all the other depth samples up to a predefined error. For a high quality representation of the depth map, the mesh and its corresponding node’s depth values will be encoded in a lossless fashion.
3.4.2. Architecture

The content adaptive mesh-based depth coding scheme has the architecture shown in Figure 36.

The walkthrough for this depth coding solution is as follows:

- **Tritree** - The binary space partitions (BSP) concept is utilized to generate a trie (for details consult [39]). This means that the depth map is divided along its longest diagonal into two triangles recursively for each triangle until a predefined threshold is reached. The criteria influencing this threshold are related to the maximum number of points that the triangle can hold and the reconstruction error of the pixels intensities [39].

At this point, there two types of data that are going to be compressed independently: sparse disparities and the mesh.

- **Sparse Disparities Differential Coding** - Depth sparse disparities are differentially encoded to remove the remaining redundancy among them;
- **Sparse Disparities Huffman Coding** - Entropy coding is performed to compress the symbols created by the differential process above;
- **Mesh Binary Trie Encoding** - The mesh is compressed via binary trie coding to condense its representation. Since the sparse binary trees - obtained with Tritree - histograms are almost flat, no entropy encoding is needed;
- **Compressed Depth Map** – This corresponds to the compressed format including all the data necessary to represent the depth map in an efficient way.

Note that the depth map size must be present at the decoder’s side in order for this depth coding solution be able to reconstruct the depth map; however, the coding of the depth map size will not be addressed since this is a metadata coding issue and it does not belong to the scope of this Thesis.
3.4.3. Main Coding Tools

This section intends to provide more details about the novel – and most complex - coding tool in this depth coding solution.

**Tritree**

The main innovation in this depth coding solution is the content adaptive meshing via the binary tritree (Tritree module in Figure 36). The idea is to use the tritree for a more efficient representation of the depth by implementing the adaptive mesh structure which has the shape of a binary tritree.

As mentioned before, a predefined threshold $\varepsilon$ must be used for the division process to create the adaptive mesh. Since depth maps are not directly seen by the user, its visual quality is not as important as the preservation of its discontinuities. Therefore, according to the authors, $\varepsilon$ should not be based on PSNR values but rather on the percentage of the depth errors ($PERR$) which measures the quality of the triangles that will be generated to create the adaptive mesh. $PERR$ is commonly used for stereo vision [19]. In this way, $PERR$ is defined as

$$PERR = \frac{1}{\Delta T} \sum_{i=0}^{\lfloor \Delta T \rfloor - 1} f(d(x_i, y_i), \tilde{d}(x_i, y_i))\%,$$  

where $\Delta T$ represents the area of a triangle $T$, $\lfloor \cdot \rfloor$ is the floor operator and $(x_i, y_i)$ are the coordinates of a pixel in $T$. The function $f$ returns the value 0 if the module’s difference between the original depth value $d$ and the reconstructed depth value $\tilde{d}$ is strictly less than one; otherwise, $f$ returns 1. In this context, $PERR$ expresses the accuracy of each triangle after the interpolation process at the decoder.

To obtain the adaptive mesh, the $PERR$ in each triangle should be minimized using the BSP concept as follows:

- The depth map is divided along one of its diagonals into two sub-triangles;
- Each sub-triangle is analyzed to check if $PERR$ is lower than $\varepsilon$ thus satisfying or not the division criterion;
- If the division conditions are not satisfied, the triangle is recursively subdivided until the division conditions are fulfilled for all triangles.

With this process, a binary tritree is generated representing the decomposition made in the depth map because:

- Each triangle can be considered as a node (thus justifying prefix ‘tri’ in the tritree word) in the tree;
- Once that a triangle (node) is decomposed into two more sub-triangles (thus justifying the word binary), those two sub-triangles will be the children nodes of the decomposed triangle (parent node) in the tree.
Note that each node in the binary has three associated depth samples - in order to define a triangle - and is connected to other two nodes (in the case it was decomposed); thus, the leaf nodes in the binary tritree corresponds to the triangles that will be used to interpolate the remaining depth values, i.e. the depth values which are not associated to any node of the tritree).

3.4.4. Performance Evaluation

Sarkis, Zia and Diepold have evaluated the performance of their proposed depth coding scheme in [38] using the “Middlebury” depth maps test bench, notably “Cones” and Teddy” sequences, as illustrated in Figure 37, and the depth maps of the Microsoft “Breakdancers” and “Ballet” sequences, which are computed using the stereo matching technique in [40].

![Figure 37 – Test sequences “Cones” (left) and “Teddy” (right) texture images (top) and depth images (below) [41].](image)

The performance results for the proposed depth coding solution are compared with the compression results for the JPEG and the JPEG 2000 image compression standards. Three quality metrics will be used, notably PERR, MSE and the RD performance curve through the average number of bits used to represent each pixel, this means the bits per pixel (BPP). The performance has also been studied varying the compression ratios.

The results displayed in Figure 38 and Figure 39 show that the proposed depth coding scheme performs better than the JPEG and JPEG 2000 standards. Considering the coded depth maps, the PERR performance is more important than the MSE performance because the first metric may lead to better synthesized views. So, although the proposed performance shows worse MSE performance than the others, it shows a better PERR performance which in this case is more relevant.
Figure 38 - Comparison of various depth coding algorithms for the “Middlebury” (first row), and “Teddy” (second row) depth sequences: (a) Depth error rate in % versus the compression ratio; (b) MSE versus the compression ratio; (c) RD performance curve [38].

Figure 39 - Comparison of the compression algorithms on the depth images computed by the stereo scheme of [40]. First row: “Breakdancers”. Second Row: “Ballet”. (a) Depth error rate in % versus the compression ratio; (b) MSE versus the compression ratio; (c) RD performance curve [38].

Rendered 3D views for some depth images overlayed with the adaptive meshes obtained with the proposed scheme are shown in Figure 40. The example shows that this algorithm can be used not only to compress depth maps but also to create a mesh representation that can be used in 3D rendering.
Concerning the encoding complexity of a frame, the following time measurements were obtained:

- 80 ms with the “Teddy” and “Cones” images
- 90 ms for the “Dolls” image, and 88 ms for the “Art” and “Moebius” images;
- 110 ms for the “Ballet” image at half the resolution (the full resolution is 1024x768) and 390 ms at the full resolution (1024x768);
- 100 ms for the “Breakdancers” at half the resolution and 350 ms at the full resolution (1024x768).

The previously enumerated time measurements were made on an AMD Opteron 64 bit quad core PC at 2.2 GHz. The algorithm has been implemented using the C++ programming language.

### 3.4.5. Strengths and Weaknesses

The adaptive sampling process – which follows the Nyquist theorem - proves to handle well depth properties. Moreover, using lossless encoding allows preserving the depth discontinuities which leads to better rendered views. Using a metric like PERR instead of PSNR is essential for measuring the quality/accuracy of the coded depth image and, therefore, to guarantee the quality of rendered views.
Content adaptive meshing via binary tritree has a corresponding binary *tritree* that allows coding the mesh in a very compact format. Also, the tritree algorithm can be easily parallelized and, therefore, each piece of a pre-divided depth map can be processed on a different CPU. The results can be then simply padded at the end.

### 3.5. Summary

This chapter has reviewed the main approaches in the literature for depth coding with emphasis on the concepts, tools and algorithms behind those approaches. Although performance evaluation results are also reported for each approach, they were not properly compared because the test material and conditions were not all the same in the papers used. For example, not all the depth coding schemes are tested with the same test sequences, with the same spatial resolutions, and the same quality metrics; while some experiments measured the PSNR of the coded depth images, others measured the PSNR of rendered images obtained based on the decoded depth maps; while both may be relevant, it has to be acknowledged that users see the synthesized images and not directly the depth maps.

Overall, the *Content Adaptive Mesh-based Depth Coding* scheme seems to be focused on dealing with depth data since:

1. Performs its operations based on a *PERR* criterion;
2. The depth samples rate changes accordingly to depth properties;
3. The created mesh can be easily used to render the 3D scenes represented by the depth map [38].

In this context, the proposed depth coding solution in the next chapter, among other facts, it also has into account these three specific previously described facts.
Proposing Depth Coding Solution: IST-Depth

The main objective of the proposed depth coding solution – designated as IST-Depth from now on - is to provide an intra depth coding solution that can compete with the H.264/AVC Intra (coding mode); this objective intends to avoid performing motion estimation for the depth coding since this is usually a very computationally intensive process.

One of the most promising approaches to code depth information is based on content adaptive meshes (described in Chapter 3) which are able to exploit the depth properties (described in Chapter 2) by changing the depth samples’ rate depending on the depth variations. Therefore, the depth coding solution proposed, implemented and evaluated in this Thesis is based on this promising approach.

IST-Depth adjusts the depth samples’ rate accordingly to the depth regions and its variations; thus, this implies using the depth samples to obtain the remaining depth data (the non-sampled depth). This is done by creating triangular depth regions – or simply triangles - where the triangle’s vertices corresponds to the depth samples and the remaining depth data is obtained by an estimation/reconstruction function which only uses the triangular depth region’s vertices information. This solution is based on the depth coding solution proposed by Sarkis in [38] (and briefly described in Chapter 3); however, it includes some algorithmic novelties by the author of this Thesis, as will be explained along this chapter.

Throughout the course of this chapter, the adopted depth coding solution architecture is firstly introduced, followed by its modules’ functional description; then, a high detailed description of the algorithms proposed for
the various architectural modules is made in order to the reader may fully understand and replicate the designed solution.

4.1. Architecture

The architecture of IST-Depth is presented in Figure 41 - as mentioned above, this architecture is rather similar to the depth coding solution architecture proposed by Sarkis in [38] - and regards the coding process of a single depth map without exploiting the temporal correlation if a depth map sequence is to be coded, as it would happen while using a H.264/AVC Intra coding solution. The interview correlation is also not explored by this codec.

![Encoder and decoder architecture of the IST-Depth codec.](image)

Naturally, the encoder and decoder processes are very complementary in terms of the various modules functionalities, except for two architectural modules: Depth Analysis and Depth Map Reconstruction. All the other architectural modules are responsible for the lossless encoding/decoding processes of the associated information, this means the selected depth samples’ values and the binary tree; this implies the amount of depth distortion introduced in the coding process, essentially depends on the Depth Analysis and Depth Map Reconstruction modules.

While the Depth Map Reconstruction architectural module has only reconstruction purposes, naturally fitting at the decoder's side, the Depth Analysis architectural module includes tools with two different purposes:
• **Analysis purposes** - Based on one or more criteria, a selection of the depth samples to be coded in a lossless fashion is made; this implies that the remaining depth data has to be approximated at the decoder by means of a reconstruction function based on the encoded depth samples. The more depth samples are encoded, the easier should be the task of the reconstruction process but the higher is the bitrate used to encode the depth samples. This analysis/selection process is typical of the encoder's side.

• **Reconstruction purposes** – Since **IST-Depth** adopts an analysis by synthesis approach, this means the encoder takes decisions after knowing what these decisions will imply at the decoder’s side, the reconstruction tools to be used at the decoder’s side have also to be present at encoder’s side. At the encoder's side, the reconstruction tools are employed to determine the quality of each (depth) triangle reconstructed at the decoder and thus also to determine the final quality of a depth map (corresponding to the quality at the decoder’s side) according to the chosen quality metrics; naturally, it is assumed here that no depth data is lost during its transmission.

Following the description above it is possible to conclude that the encoder includes most the tools of **IST-Depth** architecture: analysis, reconstruction and encoding tools. As the decoding tools are the natural counterparts of the corresponding encoding tools; thus, this chapter will only describe the encoder's modules in order to avoid repeating information.

Although **IST-Depth** architecture only includes lossless encoding processes, overall, this codec may be lossy since the data selection made in the **Depth Analysis** module, implies the remaining depth data to be approximated/interpolated and therefore making the final depth quality very dependent on the accuracy of the approximation algorithm. Summarizing, **IST-Depth** coding solution is both lossy and lossless depending on the used coding parameters. In the context of this coding paradigm, the lossless case occurs when all depth regions are perfectly approximated, when all depth regions are just composed by its vertices, or when the two previous conditions happen simultaneously.

**Required Metadata**

Some metadata is also necessary to be transmitted even though not mentioned in the architecture, this happens because this is a metadata coding issue - thus out of the scope of this Thesis since it is not related with depth coding - and also because, naturally, its impact on the final results is negligible since the only metadata that needs to be transmitted is the sequence view’s size i.e. width and height.

**Architectural Novelties**

No architectural novelties were introduced at this stage, the only difference - in architectural terms – between **IST-Depth** and the solution proposed by Sarkis in [38] is the names of the modules and how they are structured. Although, by comparing the modules in both encoding architectures, they have similar but different tools - and thus producing also similar but different results – even though they conceptually they do the same. The novelties of **IST-Depth** codec are inside each architectural module; thus, they will be emphasized as soon as they are introduced.
Tested Alternative Solutions

The implementation of two particular modules would influence the previously presented architecture; thus, they are described in this section and as follows:

1. **Inter Mode**: In order to try to explore temporal correlation, it seemed to be a good idea to code the difference between the current frame and the previous instead of the current frame, as intra mode does. However, the difference between frames produces a very different type of data than the type of data present in depth maps; and although the present architecture could encode that difference, the results would be worse than the intra mode because the difference between depth maps produces a very detailed image in some regions; therefore, for this solution achieve such level of detail, the number of decomposition would need to increase significantly (note that the number of depth samples also increases with the number of decompositions) and thus spending a greater quantity of bits than it would be spent by coding a non-differential frame;

2. **PERR Refinement**: The main idea of this solution was to decompose recursively the triangle which had worse PERR quality among all triangles until a PERR threshold for the frame was achieved; naturally, this was an alternative for the Depth Analysis architectural module. However, as the tree grows, the complexity to order a list by PERR quality grows significantly.

Taking into account the drawbacks of these two modules, their implementation was discarded.

4.2. Functional Description by Module

As mentioned before, the depth encoder includes most of the tools present at the decoder and, thus, by describing the function of each of the encoder’s modules, the function of the decoder’s modules is also implicitly described.

The list of the functions at the encoder’s side by module is the following:

- **Depth Analysis** – Selects from the raw depth map the relevant samples for the coding process accordingly to a set of input parameters related with the final quality desired;

- **Binary Tree Encoding** – Changes the representation of the binary tree, produced in the Depth Analysis module, into another representation which is more efficient in terms of bits and in a lossless fashion;

- **Depth Values Encoding** – Changes the representation of the depth values, produced in the Depth Analysis module, into another representation which is more efficient in terms of bits and in a lossless fashion;

- **Multiplexer** – Aggregates and orders all data produced by the Depth Values Encoding and the Binary Tree Encoding modules.

In the next section, the tools to implement these functions are described thoroughly.
4.3. Algorithmic Description by Module

This section intends to provide a detailed algorithmic description for each architectural module in of the IST-Depth codec; the goal is not only to detail the implemented solution but also to motivate and justify the solutions adopted and options made.

4.3.1. Depth Analysis

The Depth Analysis module is by far the most complex module in the proposed architecture; throughout this section, besides the algorithmic concepts, some mathematical concepts are also introduced whenever they have a relevant role for the understanding of the algorithmic tools.

The overall flowchart for the Depth Analysis module is shown in Figure 42 while the description of each of its modules is made in the four following sections.

![Figure 42 - Depth Analysis module flowchart.](image)

The input to this module is a full depth map and the output is a sequence of symbols resulting from the operations performed by this architectural module on the full depth map, notably a set of depth samples and a binary tree with the depth samples associated location in the full depth map.
4.3.1.1. Initial Triangular Decomposition

This module’s purpose is to decompose the original rectangular shaped depth map into two triangular shaped regions since the Planar Approximation module only operates over triangles. The new triangular shaped regions must contain all depth pixels confined by the initial rectangular shaped data; therefore, two triangular regions are needed to perform this decomposition; this module’s flowchart is shown in Figure 43 and its walkthrough is the following:

- **Input**: The rectangular shaped **Original Depth Map** is a matrix - with the same dimensions of a frame - containing all the original depth information in a frame; this allows to define the location of the triangle’s vertices using their Cartesian coordinates (x,y);

- **Defining the Left/Upper Triangle**: The initial top frame’s corners and bottom-left frame’s corner correspond to the initial left/upper triangle;

- **Defining the Right/Lower Triangle**: The bottom frame’s corners and the top-right frame’s corner correspond to the initial right/lower triangle;

- **Numbering the Triangles Vertices**: The vertices of each of the two triangles are numbered independently using a raster scanning order over those vertices in the **Original Depth Map**; this means that the first vertex of each triangle to be found using a raster scanning order is numerated as 1, the second as 2 and the third as 3. The numeration is important in order to define which side of the triangle should be divided in case of a further decomposition;

- **Output**: The final output of this module are two sets of three ordered depth samples and positions in which each set defines a triangle.

An example of this process input and output is shown in Figure 44.
Module’s Novelties

Comparing to the work described by Sarkis in [38], the fact that the vertices are numbered may be a novelty since it is not clear how Sarkis computes the partition line in the decomposition of a triangle. Numbering the triangle vertices allows to, at a later stage, avoid computations and tests by defining a rectangular region that contains the triangle with these vertices.

4.3.1.2. Triangular Decomposition

The Triangular Decomposition module has two main purposes:

- Decompose any given triangle (input) into two triangles (outputs) if required by the quality/distortion control metric;

- Renumber the new two triangles’ vertices using a raster scanning order.

To accomplish these purposes, the operations in the flowchart presented in Figure 45 are described as follows:
- **Input**: A set of three depth samples defining the triangle to be further decomposed;

- **Computing the Triangles Sides Euclidean Distances**: The Euclidean distances between all the triangles vertices are computed as follows to determine the longest side of the triangle:

\[
s_1 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2}
\]

\[
s_2 = \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2} \tag{3}
\]

\[
s_3 = \sqrt{(x_3 - x_1)^2 + (y_3 - y_1)^2} \tag{4}
\]

where \((x_1,y_1), (x_2,y_2)\) and \((x_3,y_3)\) are the Cartesian coordinates of the first, second and third triangle vertices, respectively, and \(s_1, s_2\) and \(s_3\) are the length of the first, second and third triangle sides. The first side is defined by the first and second triangle's vertices, the second side is defined by the second and third triangle's vertices and, finally, the third side is defined by the third and first triangle's vertices as shown in Figure 46.
Having computed the Euclidean distances of all the triangles, it is easy to identify the longest triangle's side, which is the side corresponding to the biggest distance;

- **Computing the Longest Side Midpoint**: This module computes the longest side midpoint coordinates using Equations 4 and 5, where the vertices \( a \) and \( b \) define the longest side.

\[
x_m = \frac{x_a + x_b}{2} \tag{5}
\]

\[
y_m = \frac{y_a + y_b}{2} \tag{6}
\]

After this computation, it is necessary to sort the new midpoint vertex among other vertices according to a raster scanning order; therefore, the following tests are made (by omission, go to the next operation after each positive test):

1. Is the first triangle side the longest? No: Go to 3;
2. The midpoint vertex is numbered as 2 and the second and third triangle vertices are numbered as 3 and 4, respectively;
3. Is the second triangle side the longest? No: Go to 5;
4. The midpoint vertex is numbered as 3 and third vertex is numbered as 4;
5. Is the third triangle side the longest? No: Exit;
6. Is the midpoint vertex before (according to the raster scanning order) the second vertex? No: Go to 8;
7. Execute 2 and then exit;
8. Is the midpoint vertex after (according to the raster scanning order) the second vertex? No: Exit;
9. Execute 4 and then exit.
- **Defining the Left/Upper Triangle**: Now that all vertices and the midpoint are numbered accordingly to the raster scanning order, the depth positions numbered as 1, 2 and 3 (if 3 is not collinear to the line defined by 1 and 2) or 4 (if 3 is collinear to the line defined by 1 and 2) define the left/upper triangle;

- **Defining a Right/Lower Triangle**: Now that all vertices and the midpoint are numbered accordingly to the raster scanning order, the depth positions numbered as 4, 3 and 2 (if 2 is not collinear to the line defined by 3 and 4) or 1 (if 2 is collinear to the line defined by 3 and 4) define the right/lower triangle;

- **Renumbering the New Triangles Vertices**: In order to this process can be repeated for every triangle numbered accordingly to the raster scanning order, the vertices in each triangle are renumbered as follows (by omission, go to the next operation after each positive test):
  1. Is this the left/upper triangle? No: Go to 3;
  2. Vertices numeration is already correctly ordered;
  3. Is this a right/lower triangle? No: Exit;
  4. The vertex numbered as 2 is renumbered as 1 (first vertex);
  5. The vertex numbered as 3 is renumbered as 2 (second vertex);
  6. The vertex numbered as 4 is renumbered as 3 (third vertex);
  7. Exit

- **Output**: The final output of this module are two sets of three ordered depth samples and positions in which each set defines a triangle.

An example of this algorithm's input and output is shown in Figure 47.

![Figure 47](image_url)

*Figure 47 – a) Triangle to be decomposed (input) where position 3 (in this case) corresponds to the midpoint in the largest edge; b) Left/upper depth triangle; c) Right/lower depth triangle; d) Renumbered left/upper depth triangle (output); e) Renumbered right/lower depth triangle (output).*
After this decomposition process, each triangle can be independently analyzed, meaning that parallel processing can be explored; therefore, each triangle can be processed by different CPUs and the results can be gathered at the end of each independent process.

**Module’s Novelties**

Comparing to the work described by Sarkis in [38], the fact that the vertices are renumbered may be a novelty since it is not clear how Sarkis computes the partition line in a decomposition of a triangle. Renumbering the triangle vertices allows to continuously avoiding useless computations and tests at later stages.

**Tested Alternative Solutions**

An alternative solution was tested to replace the Computing the Longest Side Midpoint module above which is the module determining the novel vertex position for the creation of the two new triangles. The main idea was to perform the decomposition using not the longest side as above but rather the side leading to a specific reconstruction quality property, such as:

- The lowest distortion metric average value for the new pair of triangles; or
- The lowest distortion metric value for each single triangle; or
- The minimum between the three (because there are three sides and thus three possible ways to divide a triangle) maximum distortion metric values of each pair of triangles.

This alternative solution module was discarded since - for all the three cases - it originates final triangles with very long sides (in particular, two of the three sides were too long when compared to the other side); and this led to a bad planar approximation since the depth values used in the approximation are too far away from each other and thus is highly probable that they are very different. By decomposing the longest side, this problem is solved; assuring that the quality of the triangles is, although maybe not immediately, decreasing over decompositions.

**4.3.1.3. Planar Approximation**

The Planar Approximation module has the important target to estimate the depth values inside each intermediate and final triangle generated from the full depth map decomposition, thus strongly determining the final reconstructed depth quality. As mentioned before, the quality of a triangle will be measured both in terms of PERR and MSE/PSNR. The flowchart in Figure 48 shows all operations executed in this module; each operation is described in the following:
• **Input:** The input to this module is sets of three depth samples and their location which it is the minimum necessary to define a triangle. The previous modules always provided two sets of three depth values and their locations – one set representing a left/upper triangle and the other representing a right/lower triangle – therefore, it was defined that the left/upper triangles have priority over the right/lower triangles, this type of information is not relevant here, however will be useful for the *Binary Tree Encoding* module which follows this same definition when the triangles are decomposed;

• **Defining the Counting Process:** This module’s goal is to avoid useless computations and tests in the next modules over some depth positions (a depth position is any type of coordinates that can provide the location of a depth value). This implies to select the smallest rectangular region (depth window) as possible, instead of the full depth map, which contains the triangle under study in order to be able to efficiently count the number of depth samples inside the triangle and to estimate their reconstructed depth value. With this purpose in mind, two distinct phases are present in this sub-module:

I. **Depth Window Creation:** The depth window defines the smallest rectangular region in the *Original Depth Data* which contains the triangle under processing. To accomplish this, that window must be defined based on the Cartesian coordinates of the triangle’s vertices as follows:

   o $x_{min}$ - The minimum Cartesian coordinate in the xx axis among all triangle’s vertices is assigned as the minimum value of the depth window in the xx axis;
- $x_{\text{max}}$ - The maximum Cartesian coordinate in the xx axis among all triangle’s vertices is assigned as the maximum value of the depth window in the xx axis;
- $y_{\text{min}}$ - The minimum Cartesian coordinate in the yy axis among all triangle’s vertices is assigned as the minimum value of the depth window in the yy axis;
- $y_{\text{max}}$ - The maximum Cartesian coordinate in the yy axis among all triangle’s vertices is assigned as the maximum value of the depth window in the yy axis.

An example of the creation of a depth window based on the Cartesian coordinates of a triangle’s vertices is shown in Figure 49;

![Figure 49 – Example of the creation of a depth window (transparent green) based on the Cartesian coordinates of a triangle's vertices.](image)

**II. Defining a Scanning Pattern:** Also based on the Cartesian coordinates of the triangle's vertices, a scanning pattern is defined allowing some depth values to be skipped in further computation/tests. Four different scanning patterns are illustrated in Figure 50 and defined as:

- Top-bottom, from left to right (raster scanning order);
- Top-bottom, from right to left;
- Left-right, top to bottom;
- Left-right, bottom to top.

A scanning pattern corresponds to a primary direction and a secondary direction; the primary direction is defined as the direction that is never repeated and the secondary direction is defined as the direction that is repeated successively along the primary direction.
The decision on the scanning pattern to be used is only based on the Cartesian coordinates of the triangles’ vertices as follows (by omission, go to the next operation after each positive test):

1. Are there any vertices with a common xx coordinate (meaning that the side defined by those vertices is superimposed to a row of \textit{Original Depth Data})? No: Go to 5;

2. Define the primary direction as \textit{top-down};

3. Has the vertex with the different xx coordinate a higher xx coordinate than the others which have a common coordinate? No: Define \textit{right-left} as a secondary direction and exit;

4. Define \textit{left-right} as a secondary direction and exit;

5. Are there any vertices with a common yy coordinate (meaning that the side defined by those vertices is superimposed to a column of \textit{Original Depth Data})? No: Choose default pattern (\textit{raster scanning pattern}) since is expected to the computation and tests are approximately the same for all patterns. Then exit;

6. Define the primary direction as \textit{left-right};

7. Has the vertex with the different yy coordinate a higher yy coordinate than the others which have a common coordinate? No: Define top-down as a secondary direction and exit;

8. Define down-top as a secondary direction and exit.

An example on how this previously described process works is shown in Figure 51 where two vertices (vertex 2 and 3 which define the second triangle’s side) have the same yy coordinate value – meaning that left-right is the primary direction – and where the first vertex (which is not used to define the second triangle’s side) has a lower yy coordinate value than the vertices that define the second side – meaning that the secondary direction is bottom to top – leading to the pattern shown in Figure 50 d).
Figure 51 - Example of chosen scanning pattern (red) for a triangle (blue) inside its depth window (green).

Now that a scanning pattern has already been defined, the number of depth positions to be tested can be reduced to almost half (for the best case scenario) of the total different depth positions inside the depth window by performing the following operations (by omission, go to the next operation after each positive test):

1. Chose the first depth position, accordingly to the scanning pattern selected, and then execute 2;
2. Does the current depth position belong to the triangle? Yes: Execute 4. No: Execute 3;
3. Did the previous depth position belong to the triangle? No: Execute 5;
4. Is there more depth positions to test accordingly to the scanning pattern and that belong to the depth window? Yes: Select the next depth position and execute 2. No: Exit;
5. Skip the rest of the depth positions in this current pattern’s secondary direction and then execute 4.

All the algorithms present in this module seems to be novelties of IST-Depth codec since in the proposed codec by Sarkis, nothing was mentioned regarding any module or any process that similarly avoids the computations and tests over the full depth map.

- **Barycentric Coordinates Converter**: In order to count the all the depth positions that were inside a triangle, it is essential to know if a certain depth position is inside the triangle or not. By employing the Barycentric coordinates [42], which are homogeneous coordinates defined by the vertices of a ‘simplex’ (in this case a triangle), it is easy to detect if a sample \( a \), with coordinates \( (x_a, y_a) \), belongs to a triangle \( T^{123} \), with vertices \( (x_1, y_1) \), \( (x_2, y_2) \) and \( (x_3, y_3) \), just by checking the values of the sample’s Barycentric coordinates (which is done in the next module). The following linear transformation is employed to convert depth sample position \( a \) from Cartesian coordinates to Barycentric coordinates:

\[
w_1 = \frac{(y_2 - y_3)(x_a - x_3) - (x_3 - x_2)(y_a - y_3)}{(x_1 - x_3)(y_2 - y_3) - (y_1 - y_3)(x_3 - x_2)} \quad (7)
\]
\[
w_2 = \frac{(y_3 - y_1)(x_a - x_3) - (x_1 - x_3)(y_a - y_3)}{(x_2 - x_3)(y_3 - y_1) - (y_2 - y_3)(x_3 - x_1)} \quad (8)
\]
\[
w_3 = 1 - w_1 - w_2 \quad (9)
\]
where \((w_2, w_2, w_3)\) are the Barycentric coordinates of depth sample \(a\);

- **Testing Triangle’s Depth Positions:** Using the Barycentric coordinates to check if a sample \(a\) belongs to a triangle \(T^{123}\) boils down to a simple test as follows:

\[
0 \leq w_i \leq 1, \quad \forall \ i = 1, 2, 3
\]  

(10)

with a positive answer meaning that the (“mass”) center of sample \(a\) is inside triangle \(T^{123}\). This method is extremely helpful to compute the triangle’s size (\(TriSize\)) which is defined as the number of depth positions contained in the triangle under study. The minimum number of depth positions inside a triangle is 3, with each depth position corresponding to a triangle’s vertex and thus achieving 100% reconstruction accuracy since the vertices’ depth values are encoded in a lossless fashion. Regardless he outcome of this process, the **Defining the Counting** module is notified to provide the next depth position to be tested.

- **Updating Triangle Size:** This sub-module just counts the number of depth values which passed the previous test, thus determining the size (\(TriSize\)) of the triangle under study;

- **Estimating Samples Depth:** Once again, the Barycentric coordinates eases yet another problem; in this case, the problem to estimate the depth value of a given depth position. Being \(d_a\) [39] the depth value estimated for the sample \(a\), \(\hat{d}_a\) is simply computed as:

\[
\hat{d}_a = w_1 \cdot d_1 + w_2 \cdot d_2 + w_3 \cdot d_3
\]  

(11)

where \(d_1\), \(d_2\) and \(d_3\) are the depth values of the triangle’s vertices and \((w_1, w_2, w_3)\) are the Barycentric coordinates of reconstructed depth sample \(a\);

- **Updating Triangle Quality:** Another novelty of **IST-Depth** is in the way that the \(PERR\) is computed. The \(PERR\) proposed by Sarkis represents the percentage of errored pixels in a region relatively this same area’s region (i.e. relatively to the number of pixels inside that region); this means that \(PERR\) changes very quickly, up and down, over decompositions since the area is decreasing and thus one errored pixel starts to have a heavy weight in the final \(PERR\). In order to solve this issue, the percentage of errored pixels is computed relatively to the area of the full depth map, i.e. to the total number of pixels in a depth map, which is constant. This new type of \(PERR\) computation corresponds to the modified \(PERR\) (for the rest of this Thesis, the acronym \(PERR\) refers always to the modified version). Note that for the full depth map \(PERR\) analysis both \(PERR\) metrics have the same value since the area in both cases is the same; thus providing the compatibility of analysis between these two metrics.
The quality of a triangle reconstruction may be measured in terms of its MSE and PERR (modified); both metrics are cumulative, meaning that they can be computed independently for each region and then summing all regions’ MSE and PERR, the overall depth map’s MSE and PERR may be obtained. Therefore, the triangle’s MSE and PERR are computed as follows:

\[
MSE = \sum_{i=1}^{\text{TriSize}} (d(x_i, y_i) - \hat{d}(x_i, y_i))^2
\]

\[
PERR = \frac{1}{W \times H} \sum_{i=1}^{\text{TriSize}} f(d(x_i, y_i), \hat{d}(x_i, y_i))\%
\]

where \(\text{TriSize}\) represents the triangle’s size, \(W\) and \(H\) are the depth map width and height, respectively, \((x_i, y_i)\) are the coordinates of a depth value belonging to the triangle, \(d(x_i, y_i)\) is the original depth value for the corresponding coordinates and \(\hat{d}(x_i, y_i)\) is the estimated depth value for the corresponding coordinates. The function \(f\) returns the value 0 if the original depth value \(d\) and the reconstructed depth value \(\hat{d}\) are the same; otherwise, if these values are different, \(f\) returns the value 1;

- **Output**: The output of this module is a set of depth positions (under the shape of a triangle) with an associated/estimated depth value.

**Module’s Novelties**

Comparing to the work described by Sarkis in [38], two more novelties are introduced:

- **PERR** is computed differently from Sarkis since this PERR computes the number of errored depth values relatively to the dimensions of a depth map instead to the dimensions of the current triangle; thus increasing the control over the quality of the triangles;

- **MSE** was added as a quality control parameter and thus enabling to perform decompositions based on a more visual related metric.

The **Defining the Counting Process** module may be a novelty since it is nothing is mentioned on how Sarkis reduced the number of depth positions to test in a triangle; thus, seeming that Sarkis tests all depth map’s position in order to know which of them belong to the triangle under study.

**Tested Alternative Solutions**

An alternative solution has been tested for the **Defining the Counting Process** sub-module, in Figure 48, which consisted in, besides all the same previous described restrictions, avoiding the transmission of the depth positions corresponding to the triangle’s edges (i.e. the delimiting pixels of the triangle), meaning that the depth positions that were superimposed to the triangles’ edges (including the triangle’s vertices) would be skipped for the computation of the triangle’s quality and size since they were common to another adjacent triangle(s). The main idea was to avoid the case where an edge of a triangle and the immediate inside depth values are
superimposed to a depth discontinuity; this case would imply a large number of decompositions to try correctly approximating the depth discontinuity.

This solution also required the Planar Approximation module to use three depth values (virtual vertices) belonging to the triangle in order to approximate the remaining depth values inside the triangle instead of the triangle real vertices since the access to the real vertices was not possible (because they belong to the triangle edge region which was skipped). The virtual vertices were the depth values which were the closest as possible to the triangle vertices and inside the triangle (and thus not on the triangle edge region).

This idea did not solve the case that was intended to solve because the reconstructed depth map had to be reconstructed using the “real” triangles vertices since the virtual triangle vertices were not available at the encoder’s side and thus, although this idea reduced the number of executed decompositions, the final quality had worsen significantly. Therefore, this idea was abandoned.

4.3.1.4. Assessing the Triangle’s Quality and Size

This sub-module has the target to assess each triangle’s quality to check if the user defined quality targets have been reached or not, meaning that further triangle’s decomposition(s) would be required for the reconstruction function to perform a better approximation. This is clearly an analysis by synthesis solution where the encoder simulates the decoder’s behavior to check if the target performance is being achieved.

As mentioned before, the quality/distortion control parameters adopted are the {PERR} and the {MSE/PSNR} simultaneously. To avoid very small triangles from being generated - since this would increase too much the bitrate - a quality/complexity tradeoff is also available: the user may also control the minimum size a triangle may achieve. In summary, two types of controls are possible:

1. Quality Thresholds
   - \textit{TriPERR\_threshold}: This threshold defines the minimum required \textit{PERR} quality for all triangles, meaning that every triangle must have a \textit{PERR} quality lower or equal to this \textit{PERR} threshold;
   - \textit{TriMSE\_threshold}: This threshold defines the minimum required \textit{MSE} quality for all triangles, meaning that every triangle must have a \textit{MSE} quality lower or equal to this \textit{MSE} threshold.

2. Complexity Thresholds
   - \textit{TriSize\_threshold}: This threshold defines the number of different depth positions inside a triangle after which the triangle cannot be further decomposed. Due to topological reasons, this value must not be lower than three (which is the case where the triangle is composed only by its vertices); any value above that provides a complexity/quality tradeoff.

In this context, this module’s effective function is to assess the triangle’s quality and to check if it has to be further decomposed in order to achieve the target quality and if it can be further decomposed in terms of complexity; therefore, the following tests are performed:
\[ \text{TriMSE} > \text{TriMSE}_{\text{threshold}} \land \text{TriPERR} > \text{TriPERR}_{\text{threshold}} \]
\[ \text{TriSize} > \text{TriSize}_{\text{threshold}} \]

A positive answer to both checks means that the triangle must be further decomposed to achieve the target quality and can be further decomposed in terms of complexity; otherwise, the triangle is considered to be a final triangle, this means a triangle part of the final depth map decomposition.

Regardless the outcome of these checks, a label is generated for each triangle; the label ‘0’ is generated whenever the triangle fulfils the both test, this means it is not final, and the label ‘1’ is generated otherwise, this means the triangle is final.

Since each decomposition generates two (children) triangles for each (parent) triangle, the tree used to represent the decompositions should be binary; therefore, this binary tree is capable of representing all the decompositions performed in this architectural sub-module. An example of a binary tree representing a (randomly) decomposed depth map is shown in Figure 52. Note that the two top nodes of the binary tree do not have ‘real’ parent node although they can be considered as children of the same “virtual” parent corresponding to the full depth map. In this example, the final triangles are represented with the label ‘1’ but with different colors in order to enhance its location on the full depth map; also, the decompositions to obtain those final triangles are represented in the binary tree. For each pair of labels in the same level of the tree, the left label corresponds to a left/upper triangle and the right label to a right/lower triangle.

![Figure 52 – Example binary trees representing the decomposition of a (randomly) decomposed a depth map.](image)

When all regions of the reconstructed depth map are approximated by final triangles, the final reconstructed depth map is obtained; it is then necessary to encode in a lossless fashion the information regarding the vertices’ depth values of each final triangle and also the information regarding the Cartesian coordinates of those same vertices, which are indirectly represented by the binary tree since one more decomposition means another more depth value to be coded; thus, by knowing the current triangles vertices’ coordinates, it can be computed the new depth value coordinate; which is the midpoint of the longest side of the triangle.

**Module’s Novelties**

Comparing to the work described by Sarkis in [38], two main changes were introduced:
The quality test takes into account the two quality metrics (PERR and MSE) simultaneously; naturally, it is also possible to test the impact of a single quality parameter by setting the other to a very high threshold value;

Only two labels were used to represent the binary tree; although this is a novelty it does not bring any advantages since this process is equivalent to the process used by Sarkis.

In summary, with IST-Depth codec it is possible to test the impact in depth maps of mixing two different quality metrics (PERR and MSE).

Tested Alternative Solutions

Before the employment of the PERR (modified), it was the original PERR proposed by Sarkis [38] that was being used. The main difference between these two definitions is the fact that the PERR (modified) by definition computes the percentage of errored pixels over the total number of depth samples in the depth map while the original PERR computes the percentage of errored depth sample over the number of depth samples in the current triangle. The modified PERR is a better metric because it tends to decrease with the number of decompositions while the original PERR is highly irregular since it strongly depends on the size of the triangles which continuously get smaller with further decompositions. The original PERR may increase significantly after one decomposition and then decrease as expected; the original PERR may also remain approximately the same (most common case) after a decomposition. Therefore, the modified PERR is proposed since it presents a more consistency behavior to be used as a quality/distortion control parameter.

4.3.2. Binary Tree Encoding

The Binary Tree Encoding module has the target to efficiently encode the generated binary tree which is indirectly representing the location of the final triangles vertices. This typically implies two phases:

I. Defining the precise symbols to represent the binary tree; and
II. Efficiently encoding these symbols exploiting their statistics, this means using some kind of entropy coding.

In this context, the following flowchart in the Figure 53 represents the two previous described phases respectively.

Figure 53 – Binary Tree Encoding architectural module's flowchart.
In the next two sections it will be described in detail each module of the Binary Tree Encoding architectural module’s flowchart.

4.3.2.1. Binary Tree Symbol Creation

Well known searching algorithms for trees, such as Breadth First Search (BFS) and Depth First Search (DFS), can be employed to pre-sort the whole binary tree’s bitstream by defining none search parameters; thus scanning the binary tree until all of its nodes are visited (a node in this tree corresponds to a label as it could be inferred, for example, from Figure 52). In this context, those two search algorithms are transformed, respectively, into two scanning algorithms:

1. **Breadth Based Scanning algorithm**

   The algorithm for the BBS binary tree encoding is the following (note that the label’s symbol is immediately written to a file/stream as the node is visited):
   
   1. Push the first left/upper node before pushing the first right/lower node into a FIFO (First In First Out) list;
   2. Pull and visit a node from the FIFO list;
   3. If the current visited node has children, push them into the list (always push the left/upper child before the right/lower child);
   4. Exit if the FIFO list is empty; otherwise, go to 2.

2. **Depth Based Scanning algorithm**

   The algorithm for the DBS binary tree encoding is the following (note that the label’s symbol is immediately written to a file/stream as the node is visited):
   
   1. Push the first left/upper node after pushing the first right/lower nodes into a LIFO (Last In First Out) list;
   2. Pull and visit a node from the LIFO list;
   3. If the current visited node has children, push them into the list (always push the left/upper child after the right/lower child);
   4. Exit if the LIFO list is empty; otherwise, go to 2.

Using the same binary tree of the (randomly) decomposed depth map in Figure 52, the result of its encoding process by employing both the BBS and DBS algorithms is shown in Figure 54; to ease the visualization of the visiting order of both algorithms, the trees’ nodes are numbered according to the order that they are visited.
It can be concluded, from Figure 54, that the BBS and BDS algorithms produce different bitstreams and thus, for the same entropy encoder, the compression ratios will be different.

Module’s Novelties

In [38], Sarkis first codes “the binary tree as a pre-ordered bit stream” but it is not described how the bitrate is pre-ordered; instead, Sarkis refers to two literature works: [43] and [44], which describes more than one solution to pre-order the bitstreams of binary trees.

In this module, the labels are pre-ordered using two different scanning types: breadth and depth based. With these two algorithms it is expected to pre-order the bitstream and thus benefiting from the future entropy encoding.

4.3.2.2. Binary Tree Symbol Entropy Encoding

After the BBS or DBS symbol streams are generated, representing the full binary tree, these symbols have to be encoded using some entropy encoder. Due to time constraints, no specific entropy coder was developed here and it was decided to use the available 7-Zip software [45]; this entropy coding solution is thus able to create the binary tree bitstream to be provided to the Multiplexer architectural module.

The 7-Zip software uses the Lempel-Ziv-Markov chain algorithm (LZMA) which is a variant of the LZ77 algorithm. LZ77 algorithms achieve compression by replacing portions of the data with references to matching data that have already passed through both the encoder and decoder. A match is encoded by a pair of numbers called a length-distance pair, which is equivalent to the statement "each of the next length characters is equal to the character exactly distance characters behind it in the uncompressed stream" [46].

LZMA uses an adaptive binary range encoder to encode the bitstream which is divided into packets; each packet describes either a single byte or an LZ77 sequence with its length and distance implicitly or explicitly encoded. Each part of each packet is modeled with independent contexts, so the probability predictions for each bit are correlated with the values of that bit in previous packets of the same type [47].

To encode a bitstream representing a binary tree with the 7-Zip software, the following parameters were used:
The number of fast bytes is an advanced compression parameter which can be in the range from 3 to 255. Bigger numbers provide slightly better compression strength at the expense of longer compression time. A bigger value of the fast bytes parameter can significantly increase the compression ratio when files being compressed contain long identical sequences of bytes [48].

Module’s Novelties

In [38], it is mentioned that “no considerable gain can be expected if entropy coding is performed” and probably is due to how Sarkis pre-orders his binary tree bitstream. Due to the implemented pre-ordered algorithms, it is expected to have some significant compression gains.

4.3.3. Depth Values Encoding

The Depth Values Encoding module has the target to efficiently encode the generated depth values which are representing the vertices of the final triangles. This typically implies two phases:

I. Defining the precise symbols to represent the binary tree; and
II. Efficiently encoding these symbols exploiting their statistics, this means using some kind of entropy coding.

The flowchart of the binary tree encoding algorithm proposed is shown in Figure 53.
In the next two sections it will be described in detail each module of the Depth Values Encoding architectural module’s flowchart.

4.3.3.1. Depth Values Differential Encoding

A differential representation is proposed for the triangles’ depth values. Since the depth values, in PCM (Pulse-Code Modulation) format, vary from 0 to 255 (8 bit/depth sample), the range of depth’s differences is larger, this means from -255 to +255, corresponding to the transitions from 255 to 0 and from 0 to 255, respectively. Therefore, when having a sequence of depth values as input, this module’s algorithm operates as follows:

1. Transmit the first depth value to the next module to be entropy coded;

2. Store the value of the current depth value and read the next depth value (which becomes the current depth value from now on);

3. Transmit the difference between the current depth value and the stored depth value;

4. Go to 2 until all depth values of the current depth map are read; otherwise, exit this process.

The output of this module is a set of differential depth values, and thus, the number of output values is equal to the number of input values less one. Generally, this differential process changes the statistical distribution of the depth values to be entropy coded into a peak-shaped histogram, which is more suitable for entropy encoding processes.

Module’s Novelties

Comparing to the work described by Sarkis in [38], no novelties are introduced in this module.

4.3.3.2. Depth Values Entropy Encoding

This architectural module defines an alphabet capable of exploiting the statistical distribution of the previous generated differential depth values in order to, in a later stage, be able to code them efficiently. A statistical distribution study for the various differential depth values was made in order to find their probabilities; however, their statistical distribution is very dependent on the target quality. Therefore, in order to avoid defining a different entropy encoder, e.g. a Huffman code, for each target quality, the differential depth values are coded as if they have at their disposal an ideal code; this means using their ‘entropy’ computed for the current depth map.

Defining the Alphabet

The alphabet chosen to represent the differential depth values has seventeen different symbols which are the following: ‘0’, ‘+1’, ‘+2’, ‘+4’, ‘+8’, ‘+16’, ‘+32’, ‘+64’, ‘+128’, ‘-1’, ‘-2’, ‘-4’, ‘-8’, ‘-16’, ‘-32’, ‘-64’ and ‘-128’. These symbols have two main properties:

- Each symbol of the alphabet is representing its associated value;
- Each symbol is associated to a power of two except the symbol ‘0’ which is an end symbol to signal the end of a differential value.
This alphabet can represent values varying from -255 up to +255, by performing the following algorithm (which is similar to a bitplane approximation) on a differential depth value:

1. Check the signal of the differential depth value and act accordingly to the following and then proceed to the next step:
   - Positive Signal: Use symbols with associated positive values;
   - Negative Signal: Use the symbols with associated negative values;

2. Select symbol with the highest associated absolute value;

3. For the current symbol, check if the symbol’s associated absolute value is inferior to the differential depth absolute value. Yes: The current symbol is used to represent the differential depth value. No: Select the next symbol with highest associated absolute depth and repeat this process, go to 4 in case there are no more symbols to select;

4. The differential depth value is fully represented.

To clarify this differential depth value approximation using the defined alphabet, let us assume that the differential depth value 135 has to be coded. 135, accordingly to the previous algorithm, is represented by the following symbols: ‘+128’, ‘+4’, ‘+2’, ‘+1’ and ‘0’, which by summing their associated values equals 135. Therefore, the bitstream for the 135 differential depth value is composed by the correspondent codewords for the ‘+128’, ‘+4’, ‘+2’, ‘+1’ and ‘0’ symbols respectively. The codewords for each symbol is defined accordingly to the entropy of the defined alphabet.

**Computing the Entropy of the Alphabet**

The entropy coding algorithm for the differential depth values for a depth map is the following:

1. Count the number of times that each one of the seventeen symbol is used (freq);

2. Count the total number of used symbols:

\[
#total = \sum_{i=1}^{Nr\,CW} freq_i
\]  

(16)

3. Compute the probabilities for each symbols taken as their relative frequency:

\[
p_i = \frac{\sum_{i=1}^{Nr\,CW} freq_i}{#total}
\]  

(17)

4. The total number of bits spent by all symbols with this ideal code is computed as:

\[
#bits = \sum_{i=1}^{Nr\,CW} freq_i \times \log_2(\frac{1}{p_i})
\]  

(18)
Since there are optimal codes that can achieve approximately 98% of the maximum efficiency corresponding to the approach above, coding depth values in this ideal way can be considered to be a realistic approximation since a real code with a very similar representation efficiency may be defined.

Module’s Novelties

Comparing to the work described by Sarkis in [38], the novelty present in this module is the fact that IST-Depth codec uses the codewords with the ideal length (based on the entropy of the alphabet) to code the differential depth values instead of real codewords. This was done to avoid defining a different code for each quality parameter and also because the ideal code provides a good estimation of the expected number of bits spent to code the current frame.

4.3.4. Multiplexer

The Multiplexer module has the target to put together the depth values and binary tree corresponding bitstreams. Due to the binary tree’s properties, by keeping track of the bits provided by Binary Tree Encoding module, it is possible to know when all the information regarding the encoding process of a single depth map is complete by the following reasons:

- The binary tree has always one more leaf (corresponding to the number of decomposition labels with value ‘1’) then the total number of nodes (corresponding to the number of decomposition labels with value ‘0’); therefore, when the number of labels ‘1’ is superior by one to the number of labels ‘0’, means the bitstream of the binary tree have ended since (for the previous defined BBS and DBS algorithms) this case only occurs when the final node is scanned;

- The number of expected depth values can be indirectly known by accessing the binary tree labels.

Due to these two reasons, the bitstream provided by Binary Tree Encoding architectural module is firstly received then the bitstream provided by the Depth Values Encoding architectural module.

Once the full information regarding the binary tree is received, this information is decoded in order to know the total number of expected depth values (or the number of differential depth values since they are intrinsically related) encoded by the Depth Values Encoding module; this decoding sequence must be followed since it is not possible to know the number of depth positions based on the number of decompositions, because there are different decompositions that can use the same depth position, as shown in Figure 56.

![Figure 56 – Different decompositions using the same depth position.](image-url)
Therefore, once all the depth positions are obtained through the binary tree (since each decomposition has an associated depth position) then - by removing the repeated depth positions - it is obtained the number of expected differential depth values. This means that the decoder knows the number of expected depth values by performing this analysis.

The differential depth values has an end symbol (‘0’), when the number of end symbols is equal to the number of expected differential depth values, means that the bitstream regarding the differential depth values has ended and thus, the bitstream of the current depth map has also ended.

Overall, at the decoder’s side, it is necessary to have the binary tree to know the number of depth values to be expected and the binary tree is a type of data which automatically flags its own end, the multiplexing in this architectural module is simply done by firstly gathering all the binary tree’s bits and after all the depth values’ bits for each depth map.

4.4. Summary

This chapter details the proposed depth map coding process from the concepts to the algorithms and their implementation. This encoding process may be both lossless and lossy and it is applied to a single depth map; the simplest way to encode a sequence of depth maps, associated to a video sequence, would be by repeating the described coding process to each depth map in the video sequence; this is how the IST-Depth codec encodes a sequence of depth maps.

In the next chapter, a performance evaluation to the proposed depth coding solution is made and compared to some relevant alternative coding solutions.
Chapter 5

Performance Evaluation

The main objective of this chapter is to evaluate the performance of the so-called IST-Depth codec, which was described in the previous chapter; as mentioned before, this codec was designed to take into account the specific depth properties.

The IST-Depth codec performance evaluation study is made by comparing the results obtained for the IST-Depth codec with the results obtained for well know coding standards; these results correspond to relevant performance metrics to be presented in the following. In order to have an accurate comparison between all codecs and to ease the comparison of this Thesis with other previous works, only the mandatory parameters, i.e. the parameters related with direct properties of the sequences, were altered to run the tests for the standards, since the study of the optimization of parameters for depth coding for each the standard is not part of the scope of this Thesis.

In summary, this performance evaluation should allow to understand the strengths and weaknesses of the IST-Depth codec regarding the relevant alternative codecs.

5.1. Test Conditions and Metrics

Defining the test conditions in a very precise way is important in order it is fully clear how the performance evaluation is made and thus how meaningful are the results obtained; in this context, this section targets to fully present and motivate the test conditions and metrics used to assess the performance of the developed IST-Depth codec.
1. Test sequences

Three multiview test sequences have been selected and they were chosen taking into account their different properties. Note that only the depth views are coded. The multiview sequences used for the performance evaluation are:

- **“Beer Garden”** – HD multiview sequence where two individuals make a toast with their cups and drink a little bit of beer. This sequence has a steady background and all of its objects are perfectly defined, see Figure 57;

![Figure 57 - “Beer Garden” color and depth example frames.](image)

- **“Breakdancers”** – HD multiview sequence where a group of breakdancers is performing. This sequence has a lot of quick movements and all objects are rather blurry, see Figure 58;

![Figure 58 - “Breakdancers” color and depth example frames.](image)

- **“Newspaper”** – HD multiview sequence with three individuals; one is reading the newspaper, one is holding a drink and later talking to a third individual which is passing by. The background of this sequence is relatively steady and its objects are a little blurry, see Figure 59.

![Figure 59 - “Newspaper” color and depth example frames.](image)
The blurriness in the “Breakdancers” sequence is related to the fact that its depth data was acquired using a stereo technique in [40] (other sequences do not provide information about their acquisition method); however, it seems that “Beer Garden” and “Newspaper” were acquired by a depth sensor because their depth maps are significantly less noisy than the depth maps of “Breakdancers”. Remember - from Chapter 2 - that the depth acquisition through stereo techniques are typically more error prone than the acquisition through a depth sensor.

2. Depth Spatial Resolution

Each multiview sequence has a single spatial resolution, meaning that all views of each multiview sequence have the same spatial resolution. Two different depth spatial resolutions are used for this performance evaluation:

- “Beer Garden” - 1920×1080 spatial resolution;
- “Breakdancers” and “Newspaper” - 1024×768 spatial resolution.

3. Depth Temporal Resolution

As for the depth spatial resolution, the depth temporal resolution is also the same for all views of each multiview sequence. In this case, each multiview sequence has a different temporal resolution:

- “Beer Garden” - 25 Hz temporal resolution;
- “Breakdancers” - 15 Hz temporal resolution;
- “Newspaper” - 30 Hz temporal resolution.

4. Number of Views

Each multiview sequence must be composed by at least two different views and, thus, it is important to know how many views are in each multiview sequence since the rate for each multiview sequence is the sum of all views’ rates. The number of views of each multiview sequence is the following:

- “Beer Garden” - 2 views;
- “Breakdancers” - 8 views;
- “Newspaper” – 3 views.

5. Number of Frames per View

As for the depth spatial and temporal resolutions, the number of frames per view is the same for all views of each multiview sequence. Thus, the number of frames per view is:

- “Beer Garden” – 150 frames per view;
- “Breakdancers” – 100 frames per view;
- “Newspaper” – 300 frames per view.
6. Performance Metrics

Taking into account that depth views are not supposed to be directly seen the human eye, previous metrics such as the MSE and the PSNR which typically are used to assess video quality, may be unfit to evaluate the quality of depth maps due to their specific properties.

In this context, several metrics have been suggested to evaluate the depth coding performance, such as the PERR and PSNR variants (by, for example, Noha et al. in [49]); however, none of these metrics is consensual in the literature as a good way to assess the depth coding performance. In this context, two depth quality/accuracy metrics were chosen to be used in this performance evaluation:

I. **PERR** (Percentage of Errored pixels) – This metric simply expresses the number of errored reconstructed pixels. The PERR in a sequence is the average of all PERR’s views. For a single depth map, it is defined as

\[
PERR = \frac{1}{W \times H} \sum_{i=1}^{Area} f(d(x_i,y_i), \hat{d}(x_i,y_i)) \%
\]

where \(Area\) represents the size - in pixels - of the depth map, \(W\) and \(H\) are the frame width and height, respectively, \((x_i,y_i)\) are the coordinates of a sample, \(d(x_i,y_i)\) is a original depth sample and \(\hat{d}(x_i,y_i)\) is the estimated depth value for the corresponding coordinates. The function \(f\) returns the value 0 if the original value \(d\) and the reconstructed value \(\hat{d}\) are the same; otherwise, if these values are different, \(f\) returns the value 1.

While the PERR is capable of providing the exact number of errored depths samples in a frame which is rather relevant information, the PERR does not take into account by how much the errored pixels are errored and thus it is insensitive to the size of the errors which seems to be a significant weakness.

II. **PSNR** – Since this metric is commonly used in literature for the accuracy assessment of many types of signals, it is considered here that it would be interesting to study the PSNR performance for depth coding. PSNR is a metric based on the MSE which is defined for a frame as

\[
MSE = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} [o_{i,j} - r_{i,j}]^2
\]

\[
PSNR = 10 \times \log_{10} \left( \frac{255^2}{MSE} \right)
\]

where \(W\) and \(H\) are the depth map width and height, respectively, \(i\) and \(j\) are the coordinates of a depth sample, \(o_{i,j}\) is a original depth value and \(r_{i,j}\) is the reconstructed depth value after coding. Therefore, the PSNR for a frame is defined as

While the PSNR measures the quality of the approximation between depth values - i.e., a depth sample which is approximated by a value more similar to its original value, has less impact than a pixel
approximated by a less similar value - it is not able to express the number of errored pixels, i.e. all pixels in an image can be errored and still a high PSNR value is obtained if the errors, although many, are small.

For the multiview sequences, the final quality is measured as the mean quality of all the views.

5.2. Depth Codecs Under Comparison

To fully assess the performance of the depth codecs proposed in this Thesis, appropriate metrics have to be gathered - not only for those codecs but also for relevant benchmark codecs - in order their relative performance is obtained. The benchmarking codecs typically correspond to state-of-the-art and well know codecs, such as standard codecs, which are commonly used in literature; the proposed codecs are labeled in the following as IST-Depth codecs.

For the multiview sequences, the bitrate is measured as the sum of all the coded views bitrates.

5.2.1. Benchmarking Codecs

Four different benchmarking codecs will be used in this performance evaluation. While two of the benchmarking codecs are derived from the same standard codec - the H.264/AVC coding standard - the other benchmarks are associated to other standard codecs with rather different levels of complexity and architectures, the JPEG and the MVC standards.

5.2.1.1. JPEG Standard

Although this codec is typically applied for images (since it does not explore temporal redundancy), some depth coding schemes in literature have been treating depth images of the multiview sequences to be tested; thus justifying its presence. Furthermore, it allows – at a later stage - providing a better understanding of the PSNR results of the IST-Depth codec.

RD points

Besides the natural changes regarding the test conditions described earlier, no more changes were made in its default parameters, except for the quality parameter; since the RD points for this codec were obtained by varying the quality parameter from 10 (associated with lower quality) up to 90 (associated with higher quality) with a quality step of 10.

5.2.1.2. H.264/AVC Intra Standard

This codec corresponds to the most efficient video coding standard available without exploring the temporal correlation between frames, i.e., H.264/AVC using just the intra mode to code frames.

RD points

Besides the inherent changes due to the test conditions previously described, no more changes were made in its the default configuration parameters, except for the quantization parameter; since the RD points for this codec are obtained by varying the QP (quantization parameter) from 10 up to 40 with a quantization step of 5.
5.2.1.3. H.264/AVC Inter Standard

This codec corresponds to the most efficient video coding standard available which explores, in a very efficient way, the temporal correlation between frames in the same view.

**RD points**

In order to define an intra period - which is necessary for non-intra coding solutions such as this one - it was followed the condition by ISO/IEC, described in [50], which states that “The GOP (Group of Pictures) size for each sequence shall not be less than 0.5 seconds”, i.e., the intra period shall not be less than 0.5 second. Therefore, accordingly to each sequence’s frame rate, the chosen intra period for each multiview sequence is defined by the author of this Thesis as the minimum integer value which follows the previously described condition; corresponding to the values shown in Table 2.

<table>
<thead>
<tr>
<th>Parameter/Test Sequence</th>
<th>Beer Garden</th>
<th>Breakdancers</th>
<th>Newspaper</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Intra Period</strong> (in frames)</td>
<td>13</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td><strong>Intra Period</strong> (in seconds)</td>
<td>0.52</td>
<td>0.5(3)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Besides the inherent changes due to the test conditions and parameters previously described, no more changes were made to the default configuration parameters, except for the quantization parameter; since the RD points for this codec are obtained by varying the QP (quantization parameter) from 10 up to 40 with a quantization step of 5.

5.2.1.4. MVC Standard

MVC is an extension of H.264/AVC which targets 3D video coding since it explores the correlation between the views of the same sequence.

**RD points**

In order to define an intra period - which is necessary for non-intra coding solutions such as this one - it was also followed the condition by ISO/IEC, described in [50], which states that “The GOP size for each sequence shall not be less than 0.5 seconds”, i.e., the intra period shall not be less than 0.5 second. The same line of thought was followed for the anchor period (parameter related with the interview). Therefore, accordingly to each sequences frame rate, the chosen intra and anchor periods for each multiview sequence’s view is the minimum integer value which follows the previously described condition; corresponding to the values shown in Table 3.
Table 3 – Changed parameters for MVC

<table>
<thead>
<tr>
<th>Parameter/Test Sequence</th>
<th>Beer Garden</th>
<th>Breakdancers</th>
<th>Newspaper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anchor Period (in frames/in seconds)</td>
<td>13 / 0.52</td>
<td>8 / 0.5(3)</td>
<td>15 / 0.5</td>
</tr>
<tr>
<td>Intra Period (in frames/in seconds)</td>
<td>13 / 0.52</td>
<td>8 / 0.5(3)</td>
<td>15 / 0.5</td>
</tr>
</tbody>
</table>

All of the MVC multiview coding parameters - for all sequences - are presented in Annex A. It was followed the MVC’s typical prediction scheme shown in Figure 14.

Besides the inherent changes due to the test conditions and the parameters previously described, no more changes were made to its default configuration parameters, except for the quantization parameter; since the RD points for this codec are obtained by varying the BasisQP (parameter that controls the quantization parameters) from 10 up to 40 with a quantization step of 5.

5.2.2. Proposed Depth Codecs

The performance of the two depth codecs proposed in Chapter 4 is here assessed in comparison with the benchmarks presented above; while these two codecs derive from the same basic codec, they have some significant differences, notably in terms of the triangle decomposition criteria related parameters.

Regarding the binary trees encoding process, the BBS and DBS processes produce very similar results in terms of compression efficiency (after the 7-zip encoder); however, the results for the DFS were slightly better; thus, those will be the presented results.

5.2.2.1. IST-Depth/PERR Codec

The IST-Depth/PERR codec has the purpose to maximize the PERR performance of IST-Depth codec; meaning that the decomposition criteria are mainly PERR based by setting the MSE threshold to high value; thus minimizing its effect in the decompositions.

RD points

For this codec, the RD points to be evaluated are defined by varying the PERR threshold exponentially as follows:

\[ TriPERR_{\text{threshold}} = 10^{-x}, x = \{0, 1, 2, 3, 4\} \]

(22)

Note that this PERR threshold is applied to assess the quality in a triangle and not in the whole depth map.
Regarding the remaining parameters; to minimize the TriMSE$_{threshold}$ effect in the decompositions, the TriMSE$_{threshold}$ was set to 100 and, to avoid complexity restrictions, the TriSize$_{threshold}$ was set to 3 (corresponding to its minimum possible value).

5.2.2.2. IST-Depth/MSE Codec

The IST-Depth/MSE codec has the purpose to maximize the MSE performance of IST-Depth codec; meaning that the decomposition criteria are mainly MSE based by setting the PERR threshold to high value; thus minimizing its effect in the decompositions.

RD points

For this codec, the RD points to be evaluated are defined by varying the MSE threshold exponentially as follows:

\[
\text{TriMSE}_{\text{threshold}} = 10^{-x}, x = \{0, 1, 2, 3, 4, 5\}
\]

(23)

Note that this MSE threshold is applied to assess the quality in a triangle and not in the whole depth map.

Regarding the remaining parameters; to minimize the TriPERR$_{threshold}$ effect in the decompositions, the TriPERR$_{threshold}$ was set to 100 and, to avoid complexity restrictions, the TriSize$_{threshold}$ was set to 3 (corresponding to its minimum possible value).

5.3. PERR Performance and Analysis

The PERR results obtained for the three multiview test sequences for all the codecs under study are shown in Figure 60, Figure 61 and Figure 62.

![Figure 60 – PERR results for the “Beer Garden” multiview sequence.](image-url)
The analysis of the PERR chart in Figure 60 for the “Beer Garden” sequence motivates the following remarks (keep in mind that for the PERR metric, a lower value corresponds to a better performance):

- Both the IST-Depth/PERR and the IST-Depth/MSE codecs outperform the JPEG codec, which is an important conclusion since the very low complexity JPEG codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding;
- IST-Depth/PERR can compete with the H.264/AVC Intra codec and for lower and higher bitrates can even outperform the H.264/AVC Intra codec;
- The IST-Depth/MSE has a PERR performance worse than the H.264/AVC Intra codec but much better than the JPEG codec;
- The inter mode of H.264/AVC seems to improve the results in terms of PERR for the same bitrate;
- JPEG has a strange PERR performance behavior, this can happen because making better approximations does not necessarily mean that more pixels are being correctly approximated, it just means that the quality of the approximation is increasing.

![Figure 61 - PERR results for the “Breakdancers” multiview sequence.](image)

The analysis of the PERR chart in Figure 61 for the “Breakdancers” sequence motivates the following remarks (keep in mind that for the PERR metric, a lower value corresponds to a better performance):

- IST-Depth/PERR outperforms all other codecs without exception, which is an important fact since this codec just have intra mode and is outperforming codecs with inter mode and interview prediction, i.e., H.264/AVC Inter and MVC respectively;
- The IST-Depth/MSE has a PERR performance much better than the JPEG codec, which is an important conclusion since the very low complexity JPEG codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding.
• The *IST-Depth/MSE* has a *PERR* performance worse than the *H.264/AVC Intra*, *H.264/AVC Inter* and *MVC* codecs;

• The inter mode of *H.264/AVC* does not improve significantly the results in terms of *PERR* for the same bitrate.

![Figure 62 – PERR results for the “Newspaper” multiview sequence.](image)

The analysis of the *PERR* chart for the “Newspaper” sequence, in Figure 62, motivates the following remarks (keep in mind that for the *PERR* metric, a lower value corresponds to a better performance):

• Both the *IST-Depth/PERR* and the *IST-Depth/MSE* codecs outperform the *JPEG* codec, which is an important conclusion since the very low complexity *JPEG* codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding;

• *IST-Depth/PERR* can compete with the *H.264/AVC Intra* codec and - for lower and higher bitrates - can even outperform the *H.264/AVC Intra* codec, which is an important fact since both codecs have only the intra mode;

• The *IST-Depth/MSE* has an overall *PERR* performance worse than the *H.264/AVC Intra* codec but much better than the *JPEG* codec; however, for very lower and higher bitrates, it can outperform the *H.264/AVC Intra* codec;

• The inter mode of *H.264/AVC* seems to improve the results in terms of *PERR* for the same bitrate;

• *JPEG* has a strange *PERR* performance behavior, this can happen because making better approximations does not necessarily mean that more pixels are being correctly approximated, it just means that the quality of the approximation is increasing.
General PERR Performance Conclusions

Based on the PERR results above, the following main conclusion may be derived:

- From a PERR perspective, the performance of IST-Depth/PERR codec was always better than the performance for the IST-Depth/MSE codec, which was expected since the IST-Depth/PERR codec uses PERR as its main decomposition criterion as thus it is naturally more ‘adaptive’ to this performance criterion;

- In terms of PERR performance, the most similar codec to the IST-Depth/PERR codec is the H.264/AVC Intra codec although most of the times the IST-Depth/PERR codec performed even better than the H.264/AVC Intra codec, especially for the Breakdancers sequence at lower bitrates. This means that, from a PERR perspective, the IST-Depth/PERR codec (which is also only intra mode) is a good alternative to the H.264/AVC Intra codec;

- The IST-Depth/MSE codec showed a PERR performance behavior somewhere between the H.264/AVC Intra and the JPEG codecs although closer to the H.264/AVC Intra codec. This means that, from a PERR perspective, the IST-Depth/MSE codec (which is also only intra mode) is not a good alternative to the H.264/AVC Intra codec;

- From a PERR perspective, the H.264/AVC Inter and MVC codecs provide the most efficient depth coding solutions while JPEG is clearly the least efficient coding solution.

This concludes the PERR performance analysis; in the next section the PSNR performance will be analyzed and then, the final conclusions - which take into account both PERR and PSNR performances - are presented.

5.4. PSNR Performance and Analysis

The PSNR results obtained for the three multiview sequences for all the codecs under study, according to the previously described test conditions, are shown in Figure 63, Figure 64 and Figure 65.

Figure 63 – PSNR results for the “Beer Garden” multiview sequence.
The analysis of the PSNR chart in Figure 63 for the “Beer Garden” sequence motivates the following remarks:

- **IST-Depth/MSE** starts by outperforming JPEG codec for lower bitrates (no more points representing even lower bitrates could be obtained for JPEG) and then has very similar PSNR performance, which is an important conclusion as the very low complexity JPEG codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding;
- **IST-Depth/MSE** is outperformed by H.264/AVC Intra, H.264/AVC Inter and MVC;
- **IST-Depth/PEER** is outperformed by all other codecs;
- The inter mode of H.264/AVC seems to improve significantly the results in terms of PSNR for the same bitrate.

![Figure 64 – PSNR results for the “Breakdancers” multiview sequence.](image)

The analysis of the PSNR chart in Figure 64 for the “Breakdancers” sequence motivates the following remarks:

- **IST-Depth/MSE** outperforms JPEG codec, which is an important conclusion as the very low complexity JPEG codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding;
- In terms of PSNR performance, **IST-Depth/MSE** is similar to H.264/AVC Intra, H.264/AVC Inter and MVC for intermediate bitrates, which is an important conclusion since this codec just have intra mode and is competing codecs with inter mode and interview prediction, i.e., H.264/AVC Inter and MVC respectively;
- **IST-Depth/PEER** only outperforms JPEG, for higher bitrates;
- The inter mode of H.264/AVC does not improve significantly the results in terms of PSNR for the same bitrate.

![Figure 65 – PSNR results for the “Newspaper” multiview sequence.](image)

The analysis of the PSNR chart in Figure 65 for the “Newspaper” sequence motivates the following remarks:

- **IST-Depth/MSE** outperforms JPEG codec, which is an important conclusion as the very low complexity JPEG codec (also not exploiting the temporal redundancy as the two proposed depth codecs) is still often used for depth maps coding;

- In terms of PSNR performance, **IST-Depth/MSE** is similar to H.264/AVC Intra for higher bitrates, which is an important conclusion since both codecs just have intra mode;

- **IST-Depth/PEER** is outperformed by all other codecs;

- The inter mode of H.264/AVC seems to improve significantly the results in terms of PSNR for the same bitrate.

**General PSNR Performance Conclusions**

Based on the PSNR results above, the following main conclusion may be derived:

- From a PSNR perspective, the performance of **IST-Depth/MSE** codec was almost always better than the **IST-Depth/PEER** codec performance, which was expected since the **IST-Depth/MSE** codec uses MSE as its main decomposition criterion as thus it is naturally more ‘adaptive’ to this performance criterion;

- The **IST-Depth/MSE** codec shows a PSNR performance behavior somewhere between the H.264/AVC Intra and the JPEG codecs although closer to the H.264/AVC Intra codec for higher bitrates. This
means that, from a PSNR perspective, the IST-Depth/MSE codec (which has also only intra mode) is a fairly good alternative to the H.264/AVC Intra codec and a good alternative to the JPEG codec;

- The IST-Depth/PERR codec performance is the worse in terms of PSNR when compared to all the other codecs;
- From a PERR perspective, the H.264/AVC Inter and the MVC codecs are the best performers with the H.264/AVC Intra codec some way off behind, except for the “Breakdancers” sequence.

This concludes the PSNR performance analysis; in the next section the final conclusions - which take into account both PERR and PSNR performances - are presented.

5.5. General Conclusions

Taking into account the results, comments and conclusions previously presented, i.e. from a depth coding perspective, and regarding the proposed codecs it can be concluded the following:

- The IST-Depth/MSE solution is a weak solution since it is not the leader for any of the performed tests and it is always rather behind the other codecs;
- The IST-Depth/PERR codec leads some of the performed tests, especially in terms of PERR performance and, thus, it is an interesting solution for the cases where the PERR is considered to be a good metric for depth coding assessment.

Regarding the IST-Depth/PERR codec, it provides a good depth coding solution in some conditions because:

- From a PERR perspective, it always outperforms its direct competitors, the H.264/AVC Intra and the JPEG codecs, since codecs without temporal prediction typically has a lesser associated complexity;
- From a PERR perspective, the IST-Depth/PERR codec outperforms some of the its lesser direct competitors - the H.264/AVC Inter and MVC codecs – for the “Breakdancers” sequence which is a very important conclusion, since this seems to corroborate one of the main objectives in this Thesis: to propose a depth coding solution that can compete with already proposed depth coding solutions;
- Also from a PERR perspective, the IST-Depth/PERR codec constantly outperforms its direct competitor - the H.264/AVC Intra codec – which is a very important conclusion, since this seems to corroborate one of the main objectives in this Thesis: to propose a depth coding solution that can to compete with already proposed depth coding solutions;
- Although it has an inferior performance than most of the other codecs from a PSNR perspective, the IST-Depth (both PERR and MSE based codecs) provides the location of the lossless coded samples and therefore providing vital information regarding reliable depth data in a depth map at the decoder’s side. For view synthesis purposes, this is a critical issue in order to be able to perform a better reconstruction of the view [51].

Regarding binary tree coding, the compression factor of the 7-zip encoding process varied from 20 up to 70, for higher and lower bitrates respectively. Regardless the success of this encoding process, it has a weak impact
(approximately 7% for the best case) regarding the IST-Depth performance; meaning that the encoding process of depth samples has a stronger impact for the IST-Depth performance than the binary tree encoding process.

Taking all these conclusions into account, it can be considered that these Thesis objectives were accomplished.
Chapter 6

Summary and Future Work

This chapter summarizes the work done in this Thesis and its main conclusions. Then, it suggests new ideas for the improvement of the proposed depth coding solution - IST-Depth - described in the chapter four.

6.1. Summary and Conclusions

The Thesis started by stating the fact that 3D experiences are becoming more common to experience in our daily lives: through cinema, television, computers, etc. In order to support those 3D experiences, by allowing a wide and a better variety of 3D features, depth data had to be present in the 3D systems along with color video data instead of just having color video data; since the processes to compute depth data through color video data are - for now – highly complex and also are highly error prone.

Besides the support of 3D features, it was concluded that a 3D systems using depth data and color video data is more efficient in terms of bitrate then a 3D system that only uses color video data since depth can be compressed at 20% of the bitrate necessary to encode a color video, while still providing good quality for the synthesized views. So, depth data may lower the complexity required to support 3D features and, at the same time, can lower the necessary bit rate for the same number of views provided by the 3D system, when comparing with systems that only use color video.

Depth data has very specific properties (smooth surfaces and sharp edges delimiting the smooth surfaces) that are very different from the properties of normal color video; thus, a study regarding depth properties was made in which was concluded that using color video codecs, such as H.264/AVC, it may be less efficient than using codecs which could explore those depth specific properties. It was also concluded that using a PSNR based metric it is not appropriate for measuring the quality of depth maps, since the encoded depth map do not to be
visually similar to the originals; instead, they need to be able to provide synthesized view that are similar to the original; therefore, it is issued - that to obtain this result – a PERR based metric is more appropriate.

In this context, in order to enhance the efficiency of 3D systems using depth data, it is essential to encode depth data in an efficient fashion; thus, this Thesis proposed a depth coding solution which explores specific depth properties and also its appropriate evaluation with the purpose of trying to overcome already implemented depth coding solutions and also well know standards (normally used to code color video data).

The performance evaluation of IST-Depth showed encouraging results, even though this solution can be improved in several aspects. The aspects that can be improved and were noticed by the Author are briefly described in the following section.

6.2. Future Work

Although the results of IST-Depth were encouraging, there is still room for improvements in some aspects, notably in the following areas:

- **Adding a Inter Coding Mode** – The temporal redundancy between the successive depth frames is currently not exploited as the current version of IST-Depth codec only performs intra based coding, thus only exploiting the spatial redundancy. As mentioned before, a simple inter coding solution has been tested which basically used the presented depth coding process to code not a depth frame but rather a depth residual frame, i.e. the depth frame resulting from subtracting a depth frame from its previous depth frame, to exploit the temporal redundancy in a rather simple and low complexity way. However, since the resulting residual frame was very fragmented, the depth needed for the coding tree was too large which resulted in a worst performance than the pure intra mode coding. Therefore, exploiting the depth temporal redundancy is still an objective to be addressed. A possible solution to explore the redundancy between frames is to code the inter frame as an intra frame, then analyze the differences between the trees of the previous and the current frames and only code the differences (and naturally also the different depth values), i.e. instead of encoding the residual frame, the residual tree with the correspondent depth values would be coded; this could improve the current performance of the IST-Depth codec;

- **Adding a Interview Coding Mode** – Since the redundancy between the various views is currently not exploited by the IST-Depth codec, this is another topic to be addressed. A possible interview prediction structure to be integrated in the IST-Depth codec could be similar to the MVC interview prediction structure. Then, after interview prediction, the coding process would be somehow similar to the coding process and temporal prediction, i.e. a comparison between depth coding trees and depth values to only code the relevant differences;

- **‘Playing’ with Lossless Depth Values to Maximize Triangle’s Quality** – Another idea to improve the current performance of IST-Depth codec is to manipulate the depth values to be transmitted in a lossless fashion in order to maximize the decomposition triangles quality; for example, after performing all decompositions, the depth map would be analyzed to choose the depth value for each triangle’s vertex in order to maximize, for example, the number of accurate depth values inside the triangles to which
this vertex is used to approximate the remaining depth values. Although this process may improve, in general, the quality of a reconstructed depth map, it brings a limitation since the ability to locate - in a simple way - the depth values which are equal to the originals is lost; this may also affect the synthesis reconstruction results since they are very sensitive to unreliable depth data and now this data cannot be easily detected;

- **Alternative Decomposition Data Coding Approach** – An alternative approach to code the depth tree and depth values after the decompositions is obtained is to use (run, level) pairs where the run regards the number of positions until the next vertex and the level regards its associated depth value. These (run, level) pairs may be differentially encoded, thus resulting in a better performance.

To conclude, there is still a lot of research regarding depth coding that can improve the compression performance of the *IST-Depth* codec.
References


Annex A

Regarding the MVC coding performed in Chapter 5, this Annex presents the configuration of the multiview parameters - of the MVC codec - for all test sequences. Thus, it includes three images showing a print of the employed multiview coding parameters in MVC, one for each sequence.

```
#------------- MULTIVIEW CODING PARAMETERS ---------------
NumViewsMinusOne  1  # (Number of view to be coded minus 1)
ViewOrder         0-1  # (Order in which view_ids are coded)
View_ID           0  # view_id (0..1024): valid range
Fwd_NumAnchorRefs 0  # (number of list_0 references for anchor)
Bwd_NumAnchorRefs 0  # (number of list_1 references for anchor)
Fwd_NumNonAnchorRefs 0  # (number of list_0 references for non-anchor)
Bwd_NumNonAnchorRefs 0  # (number of list_1 references for non-anchor)

View_ID           1
Fwd_NumAnchorRefs  1
Bwd_NumAnchorRefs  0
Fwd_NumNonAnchorRefs 0
Bwd_NumNonAnchorRefs 0
Fwd_AnchorRefs     0 0
```

Figure A.1 - MVC's Multiview Coding Parameters set for the "Beer Garden" sequence.
Figure A-2 - MVC’s Multiview Coding Parameters set for the “Breakdancers” sequence.
Figure A-3 - MVC's Multiview Coding Parameters set for the "Newspaper" sequence.