ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO FREITAS SANCHEZ

EXPLORATION OF ALGORITHMS AND IMPLEMENTATIONS FOR EFFICIENT 3D-HEVC DEPTH MAP ENCODING

Porto Alegre

2019

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
COMPUTER SCIENCE GRADUATE PROGRAM

EXPLORATION OF ALGORITHMS AND IMPLEMENTATIONS FOR EFFICIENT 3D-HEVC DEPTH MAP ENCODING

GUSTAVO FREITAS SANCHEZ

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. César Augusto Missio Marcon
Co-Advisor: Prof. Luciano Volcan Agostini

Porto Alegre
2019

Ficha Catalográfica

S211e Sanchez, Gustavo Freitas

Exploration of Algorithms and Implementations for Efficient 3D-HEVC Depth Map Encoding / Gustavo Freitas Sanchez . – 2019.
137 f.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. César Augusto Missio Marcon.
Co-orientador: Prof. Dr. Luciano Volcan Agostini.

1. 3D-HEVC. 2. Codificação de mapas de profundidade. 3. Predição intra-quadro. 4. Predição inter-quadro. 5. Minimização do esforço computacional. I. Marcon, César Augusto Missio. II. Agostini, Luciano Volcan. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Salete Maria Sartori CRB-10/1363

Gustavo Freitas Sanchez

EXPLORATION OF ALGORITHMS AND IMPLEMENTATIONS FOR EFFICIENT 3D-HEVC DEPTH MAP ENCODING

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Computer Science, of the Graduate Program in Computer Science, School of Computer Science of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 12, 2019.

“Risk comes from not knowing what you’re doing.”
(Warren Buffett)

ACKNOWLEDGMENTS

Although it was a long journey these four years in which I developed this Thesis, it was one of the best stages of my life. This is thanks to all the people I have lived with during this period because they have made both my professional and personal life extremely enjoyable.

First of all, I thank my friends, advisors and from now collaborators Marcon and Luciano. I am grateful to Marcon for having opened the door for me to enter the Ph.D. at PUCRS, giving me total independence and supporting me to work on the subject that I had been working for a while, besides always being present and helping in any personal or professional problems. To Luciano, whom I have known for many years, I thank you for giving me the opportunity to work with video coding (from many years ago). Besides, during the Ph.D. period, Luciano always helped in the most difficult and necessary situations. I also had the opportunity of an international exchange of six months in Lisbon, where I thank Professor Leonel for having received me very well and made himself available to collaborate with my work.

In this period of Lisbon, I also made several friends (too many to list name by name) and thank you all for helping me adapt quickly in another country with a different culture. I have no doubt that these friendships will continue on for my entire life.

Most of my work was not built only along with my advisors, but rather with teamwork with other friends. So, I thank the partnership of Ramon and Cataldo who have been working together with me in some periods of this Ph.D. course. And also, thank Mario, whom I have been working together since my master's degree, and is now a friend that I can talk about any subject including work, gym, politics, economy, and anything.

My childhood friends were also essential for making this Ph.D. period so pleasant. Any time we met, it was impossible not to laugh and forget about all the obstacles. So I thank Joaquim, Igor, Alexandre, Eduardo, Pedro, and Alessandro for being so present at this stage of my life.

Finally, I could not fail to thank my parents and my family for always motivating me to study and try to try to improve more and more.

Thank you all!

EXPLORAÇÃO DE ALGORITMOS E IMPLEMENTAÇÕES PARA CODIFICAÇÃO EFICIENTE DE MAPAS DE PROFUNDIDADE NO PADRÃO 3D-HEVC

RESUMO

O padrão 3D High Efficiency Video Coding (3D-HEVC) foi o primeiro padrão a usar a codificação de mapas de profundidade juntamente com a codificação de textura para obter alta taxa de compressão e qualidade em vídeos 3D. A codificação de mapas de profundidade introduziu ferramentas que requerem maior esforço computacional se comparado com a codificação de textura, implicando novos desafios para codificar vídeos 3D eficientemente, principalmente quando a codificação for efetuada em sistemas embarcados ou requerer tempo real. Esta Tese propõe algoritmos e implementações que permitem aprimorar a codificação de mapas de profundidade no padrão 3D-HEVC. As contribuições principais da Tese são agrupadas em três partes. (i) Redução do esforço de codificação intra-quadro, onde foram projetados quatro novos algoritmos com base em uma profunda análise da distribuição de tempos e de modos usados na codificação. (ii) Aprimoramento dos modos de bipartição, onde foram explorados (a) o controle do esforço de codificação dos modos de bipartição, (b) o paralelismo para o modo Intra_Wedge, (c) a compressão de padrões para redução de armazenamento de wedgelets, e (d) o projeto de hardware para a decodificação dos modos de bipartição. (iii) Redução do esforço de codificação inter-quadros, onde foram projetados três novos algoritmos para acelerar a codificação de quadros do tipo P e B. O conjunto de algoritmos e técnicas desenvolvidos podem ser aplicados individualmente ou combinados para melhorar o uso dos recursos computacionais ao codificar mapas de profundidade.

Palavras-Chave: 3D-HEVC, codificação de mapas de profundidade, predição intra-quadro, predição inter-quadro, minimização do esforço computacional.

EXPLORATION OF ALGORITHMS AND IMPLEMENTATIONS FOR EFFICIENT 3D-HEVC DEPTH MAP ENCODING

ABSTRACT

3D High Efficiency Video Coding (3D-HEVC) standard was the pioneer standard to use depth map coding along with texture coding to achieve high compression ratio and quality in 3D videos. Depth map coding introduced new tools requiring more computational effort compared to texture coding, implying new challenges to encode 3D videos efficiently, especially when the encoding is performed on embedded systems or requiring real-time processing. This Thesis proposes algorithms and implementations to improve the depth map coding in the 3D-HEVC standard. The main contributions are divided into three parts. (i) Reduction of intra-frame coding effort, where four new algorithms were designed based on a deep analysis of the time distribution and modes used in the coding process. (ii) Enhancement of bipartition modes, exploring (a) the encoding effort control of the bipartition modes, (b) the parallelism for Intra_Wedge mode, (c) the compression of patterns for wedgelet storage reduction, and d) the hardware design for the bipartition modes decoding. (iii) Reduction of inter-frame coding effort, where three new algorithms were designed to speed up the coding of P- and B-frames. The set of algorithms and techniques developed can be applied individually or combined to improve the use of computational resources when encoding depth maps.

Keywords: 3D-HEVC, depth maps coding, intra-frame prediction, inter-frame prediction, encoding effort reduction.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AVS2 – Audio Video Coding Standard 2

BCM – Block Change Map

BD-RATE – Bjontegaard Delta rate

BER – Bipartition modes Evaluation Rate

B-LCM – Block Line Change Map

CABAC – Context Adaptive Binary Arithmetic Coding

CMOS – Complementary Metal Oxide Semiconductor

CTC – Common Test Conditions

CU – Coding Unit

CPV – Constant Partition Value

CPU – Central Processing Unit

CTU – Coding Tree Unit

DC – Direct Component

DCT – Discrete Cosine Transform

DE – Disparity Estimation

D-FB&C – Dual First Bit and Change

DFPS – DMM-1 Fast Pattern Selection

DIBR – Depth Image Based Rendering

DIS – Depth Intra Skip

DMM – Depth Modeling Modes

DDF-GMOF – Double Degree of Freedom GMOF

DS – Diamond Search

E-ADME – Edge-Aware Depth Motion Estimation

EC – Entropy Coding

ED-RMD – Enhanced Depth Rough Mode Decision

EEC – Encoding Effort Control

ERR – Ending Rows Removal

FB&C – First Bit and Change

FS – Full Search

FPS – Frames per Second

GMOF – Gradient-based Mode One Filter

GPU – Graphics Processing Unit

HEVC – High Efficiency Video Coding

HEVC-SCC – High Efficiency Video Coding Screen Content

IG – Information Gain

I-SDSP – Iterative Small Diamond Search Pattern

ISP – Initial Search Point

JCT-3V – Joint Collaborative Team on 3D Video Coding Extension Development

LCM – Line Change Map

MAD – Maximum Absolute Deviation

ME – Motion Estimation

MPM – Most Probable Modes

MSM – Merge/Skip Mode

MVD – Multiview Video plus Depth

PDF – Probability Density Function

PID – Proportional-Integral-Derivative

PSNR – Peak Signal-to-Noise Ratio

PU – Prediction Unit

P&GMOF – Pattern-based Gradient Mode One Filter

QP – Quantization Parameter

RA – Random Access

RD-COST – Rate-Distortion cost

RD-LIST – Rate-Distortion list

REP – Reduced Error Pruning

RMD – Rough Mode Decision

SAD – Sum of Absolute Differences

SATD – Sum of Absolute Transformed Differences

SDC – Segment-wise Direct Component Coding

SDF-GMOF – Single Degree of Freedom GMOF

SED – Simplified Edge Detector

S-GMOF – Strong GMOF

SIMD – Single Instruction Multiple Data

SSD – Sum of Squared Differences

SVDC – Synthesized View Distortion Change

TH – Threshold

TQ – Transform-Quantization

TU – Transform Unit

TZS – Test Zone Search

VHDL – Very High Speed Integrated Circuits Hardware Description Language

WEKA – Waikato Environment for Knowledge Analysis

WMEM – Wedgelet Memory

2D – Two-dimensional

3D – Three-dimensional

3D-HEVC – 3D-High Efficiency Video Coding Screen Content

3D-HTM – 3D-HEVC Test Model

CONTENTS

1. INTRODUCTION

Several digital video applications have arisen in the past few years, demanding high quality and high definition videos. Besides, these videos are stored in several media and places and streamed over several heterogeneous communication systems distributed at the internet. Therefore, a high effort was spent in the standardization of the modern two-dimensional (2D) video coding such as High Efficiency Video Coding (HEVC) [73], VP9 [23], Audio Video Coding Standard 2 (AVS2) [39] and probably the next emergent video coding standards, allowing to maintain a good video quality without penalizing the stream size. However, currently, video coding applications go beyond capturing simple 2D scenes. This kind of video application allows sharing computer screens or enjoying an experience that goes beyond 2D videos by providing a depth perception of the scene. These new characteristics are not well-explored by the 2D video coding standards since they focus on capture texture aspects of a single view only; consequently, limiting the efficiency on capturing depth aspects of each video's scene. To fulfill this requirement, video coding experts designed HEVC extensions that included HEVC Screen Content (HEVC-SCC) [80], which allows achieving a higher performance when sharing computer screens or similar videos, and the three-dimensional (3D) High Efficiency Video Coding (3D-HEVC) [40] [76], which explores 3D video redundancies. Since this thesis focus on 3D videos, the explanations herein considered only 3D-HEVC, which is the most advanced 3D video coding standard.

3D-HEVC was based in the Multiview Video plus Depth (MVD) representation [29], where each texture (regular data seen in video sequences) frame is associated with a depth map captured from the same camera that captures the texture videos. The texture frames are the natural images with colors that are presented at the TV, composed of luminance and chrominance samples. The depth maps are represented by gray shades using only lumincance samples. These depth maps provide the geometrical distance between objects and the camera [86]. Figure 1.1(a) shows an example of a texture frame; whereas Figure 1.1(b) illustrates its associated depth map, both figures are extracted from Kendo video sequence [75].

The motivation of using MVD is the bandwidth reduction for a 3D video transmission because the decoder can synthesize high-quality virtual views interpolating texture views based on the depth data, using techniques such as Depth Image Based Rendering (DIBR) [16]. These techniques allow synthesizing a dense set of high-quality texture views of the scene, reducing the number of transmitted texture views.

Figure 1.2 shows an abstraction of the entire processes of 3D-HEVC coding and decoding, which starts from the scene capturing, passing by the 3D-HEVC video encoding, the video storing/transmitting, the video decoding and the intermediary synthesized views generating.

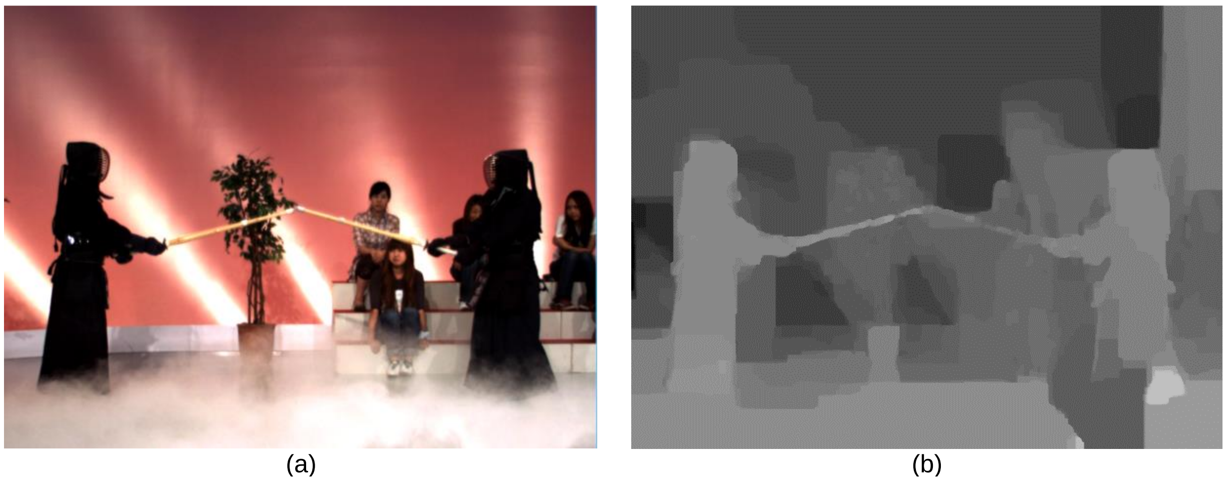


Figure 1.1 – (a) Texture view and its associated (b) depth map obtained from Kendo video sequence (source: [75]).

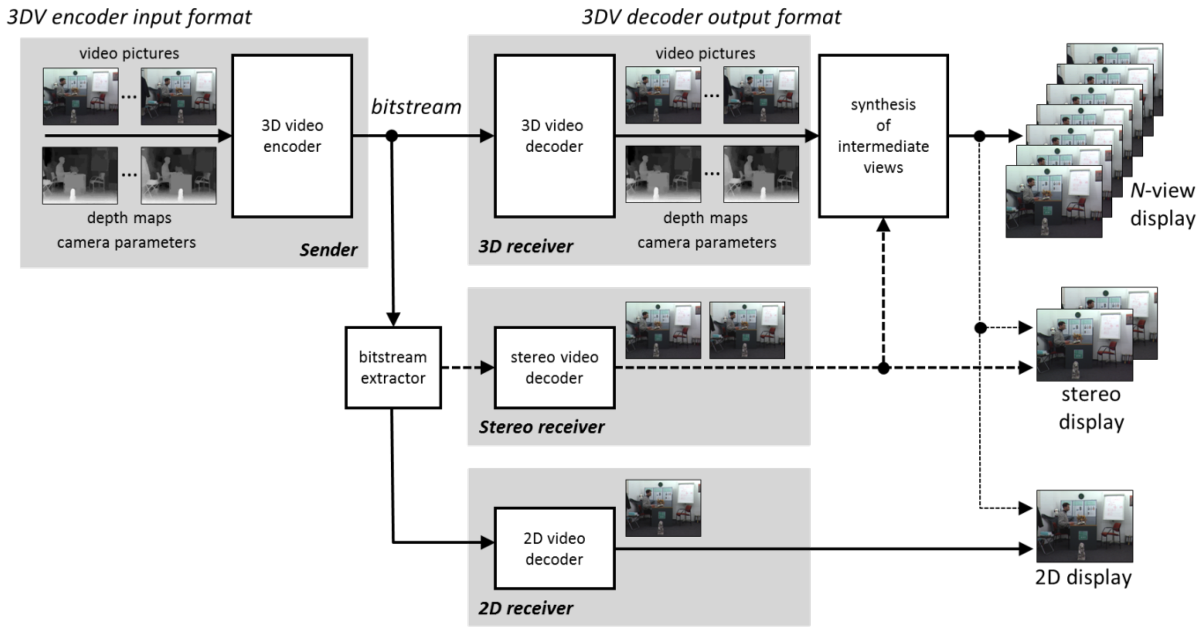


Figure 1.2 – 3D-HEVC encoding and decoding flows (source: [8]).

Firstly, a scene is captured by multiple cameras simultaneously. Each camera provides a raw texture and depth data from the same perspective, which are encoded together in a 3D-HEVC encoder, packing the data inside an encoded stream. Each camera is called a view, where the texture and the depth content are included.

The encoded stream can be stored in the device memory or cloud, broadcasted, transmitted, and/or others. Whenever another application requires decoding this stream, the 3D-HEVC decoder is responsible for reconstructing the original encoded views, providing the reconstructed raw texture and depth data. However, a 3D application may require more than the number of transmitted views. Therefore, a view-synthesis is applied to generate innumerous texture views located between the original encoded views. Notice the 3D-HEVC bitstream can also provide information for stereo or 2D videos decoder.

Figure 1.1 shows depth maps contain different characteristics from texture, i.e., depth maps are characterized by containing large regions with homogeneous values (background or objects bodies) and sharp edges (borders of objects) [72]. Considering these characteristics, the algorithms can encode depth maps with higher efficiency and quality. The quality of the encoded depth maps is crucial to allow generating high-quality synthesized views, which leads to a significant challenge in this scenario, because, encoding depth maps using only traditional 2D video coding techniques introduces errors in the encoding process.

The quality of the encoded depth maps influences the 3D video quality indirectly since depth maps are used to synthesize virtual views that are seen by the application viewers. The highest difference is noticed in the depth map edges; therefore, it is essential to encode as precisely as possible the depth maps, preserving these edges (i.e., without smoothing them) and avoiding errors in the video synthesis process.

Taking into account the importance of edge preservation and the requirement of obtaining high compression rates without decreasing the video quality, the Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) experts have proposed new tools for 3D-HEVC depth map coding. These new coding tools include the Intra-picture skip [31], Direct Component (DC)-only [33] and the bipartition modes (composed of the algorithms Intra_Wedge and Intra_Contour [37]). It is essential to emphasize during the 3D-HEVC standardization, the names of these tools have changed; therefore, some timesaving algorithms designed in this Thesis are refereeing the old name to maintain the original algorithm name. The bipartition modes are also found in the literature as Depth Modeling Modes (DMMs), where Intra_Wedge was called DMM-1 and Intra_Contour was called DMM-4. The Intra-picture skip is also found in the literature as Depth Intra Skip (DIS), and the DC-only is found in the literature as Segment-wise DC Coding (SDC).

Considering MVD is a relative new encoding format, the depth map coding has not been so investigated as texture coding over the last decade, having significant space for researchers provide new enhancements for its algorithms. Besides, several encoding tools were inherited from the HEVC texture coding without considering the depth map simplicity, and some new tools were designed without significant exploration on depth map coding; therefore, opening space for deeper exploration of the depth map coding. Thus, this Thesis explores depth map characteristics and coding algorithms in several levels aiming to enhance depth map coding implementations.

1.1 Main Objective

A high-level diagram of a 3D encoder is presented in Figure 1.3, where the sections or chapters of this Thesis contributions are indicated in the focused encoding tool.

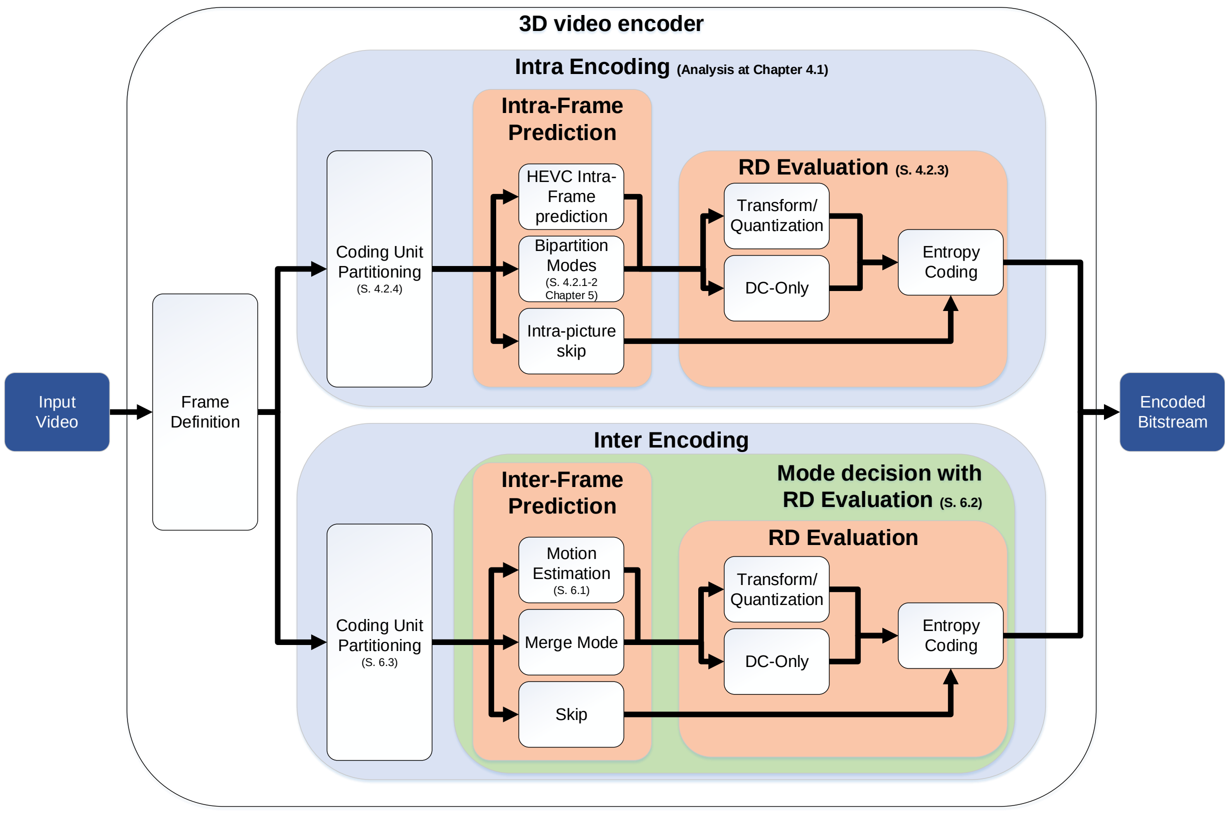


Figure 1.3 – High-level diagram of the 3D encoder with the Thesis contributions.

The primary objectives of this Thesis are to explore new algorithms and implementations to enhance depth maps coding. This thesis lies in the exploration of new techniques and algorithms for the design of efficient 3D-HEVC depth maps coding tools at several levels. Among the technologies used are machine learning, algorithmic parallelization, statistical analysis, and hardware/software implementations; the application of these technologies based on an in-depth analysis of the primary features of the depth maps allowed us to achieve significant results in timesaving and memory usage requirements. The main contributions of this Thesis are summarized as follows:

- Detailed analysis of the encoding time distribution in the encoding tools for depth map intra-frame prediction [60] [69];
- Detailed analysis of the encoding modes usage in the depth map intra-frame prediction [69];
- Four timesaving algorithms for coding the depth map intra-frame prediction [48] [54] [55] [62];
- Specific implementation explorations on bipartition modes including encoding effort control [65], parallelism exploration for Intra_Wedge in multicore [58] and Graphics Processing Unit (GPU) systems [59], methods for reducing its memory usage in Intra_-

Wedge [53] [64], and a hardware implementation of a bipartition mode decoder [56] [61]; and

- Three timesaving algorithms for inter-frame prediction coding [47] [49] [68].

1.2 Specific Objectives

To achieve the primary objective of this work the following specific objectives were accomplished:

- Analysis of the use of the encoding modes;
- Analysis of the execution time of the encoding modes;
- Investigating related works concerning depth map timesaving;
- Statistical analysis of the most time-consuming encoding modes;
- Decision algorithms creation to speed up the intra-frame prediction;
- Exploration of the implementation characteristics of the bipartition modes; and
- Decision algorithm design to speed up the inter-frame prediction.

1.3 Thesis' Structure

This Thesis is divided into seven chapters to present the 3D-HEVC encoding concepts and contributions. Chapter 2 presents the theoretical background showing the 3D-HEVC depth map encoding structure and explaining the main tools used during its processing. Chapter 3 presents and discusses a dense set of related work. Chapter 4 describes the contributions related to the encoding effort reduction in the intra-frame prediction, showing an analysis of the distribution and encoding time of the depth map encoding tools followed by the designed timesaving algorithms for intra-frame prediction. Chapter 5 presents the contributions specifically at bipartition modes including parallel investigation, encoding effort control, Intra_Wedge memory aware solutions and hardware design for its decoding. Chapter 6 presents the contributions focused on reducing the computational effort of inter-frame coding. Finally, Chapter 7 concludes this Thesis.

2. THEORETICAL BACKGROUND

This chapter contains the 3D-HEVC theoretical background employed on this Thesis. Section 2.1 describes the 3D-HEVC basic encoding structure. Section 2.2 presents and discusses the intra-frame prediction algorithms applied to depth maps. Section 2.3 details the inter-frame and inter-view encoding algorithms. Finally, Section 2.4 discusses the common encoding steps applied in the resulting of these prediction algorithms.

2.1 3D-HEVC Basic Encoding Structure

The 3D-HEVC encoding is based on (i) an efficient prediction structure, where the predictions strategies are employed according to the encoding frame type; and (ii) a flexible quadtree structure, where different block sizes are evaluated searching for the best combination of block partitioning.

2.1.1 Temporal and Spatial Prediction Structure

Figure 2.1 shows the 3D-HEVC prediction structure. For a given *k* instant of time (*Time*$_k$), all texture data and their associated depth maps are encoded by the 3D-HEVC encoder in a structure called access unit.

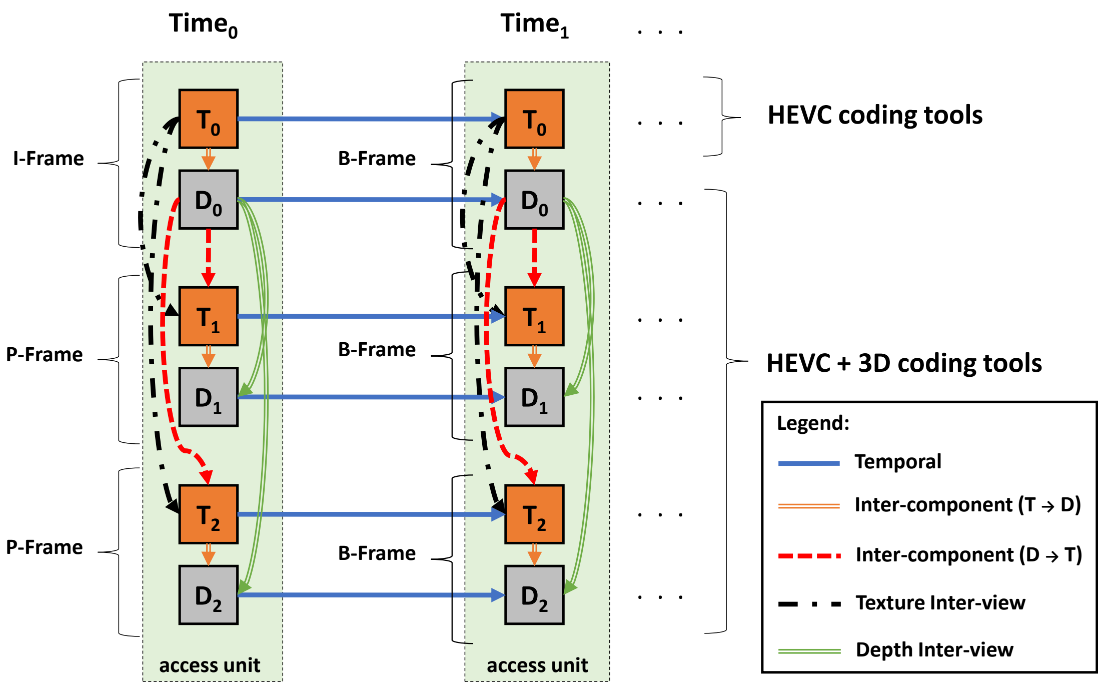


Figure 2.1 – Basic 3D-HEVC prediction structure (Source: [49]).

The access unit is a structure containing texture and depth maps of all available views in a given instant of time. Inside an access unit, a base texture view (T_0) is used as the reference for the dependent texture views (T_1, T_2) for exploring the similarities among views. The base view, i.e. T_0, is independent of the other views being encoded by a conventional HEVC encoder since an HEVC decoder can provide images for the 2D-conventional display from a 3D-HEVC compatible bitstream. The dependent views are encoded using information from the base view (T_0) to reduce the inter-view redundancy. Figure 2.1 also shows the 3D-HEVC inter-frame dependencies, allowing exploring inter-frame redundancies among access units; i.e., exploring the redundancies among time instants.

3D-HEVC works employing three frame types: I-, P-, and B-frames. I-frames are present only in the base view, and they are encoded using the information inside the current encoding frame, i.e., employing the intra-frame prediction algorithms. I-frames are used to encode few base views - in the first video frame and from time to time (a specific amount of time defined by the encoder) to refresh the current scene. I-frames allow to reconstruct the video without any information of past-encoded frames; therefore, I-frames can recover the video quality that eventually was lost among several frames propagation. Besides, I-frames help to recover from transmission faults that lose packages. P-frames are employed in the same access unities that encoded the base view using I-frames. They can be encoded applying intra-frame and inter-view predictions. Finally, B-frames can be encoded using intra-, inter-frame, and inter-view predictions. Inter-layer prediction techniques can be also used in all types of frames.

The depth map (D_0, D_1, and D_2) receives the same frame type of its associated texture frame; however, 3D-HEVC allows using inter-component prediction and other encoding tools such as Intra_Wedge, Intra_Contour, DC-only, and Intra-picture skip.

2.1.2 Quadtree Structure

The 3D-HEVC depth map coding applies the same quadtree structure of the HEVC texture coding [35]; i.e., before being encoded, the current frame is divided into Coding Tree Units (CTUs) - a smaller and basic coding structure. Subsequently, each CTU can be split into four Coding Units (CUs), which can be divided recursively into four CUs until reaching a minimum pre-determined quadtree level.

Figure 2.2 exemplifies a 64×64 CTU partitioning, where the encoder selects the final quadtree structure. The quadtree leaves are the final selected CU to encode the current CTU. It is worth to mention this high flexibility is responsible for achieving a high encoding efficiency in modern video coding standards since regions with fewer details can be encoded with larger blocks, while regions with more details can be further refined using smaller blocks.

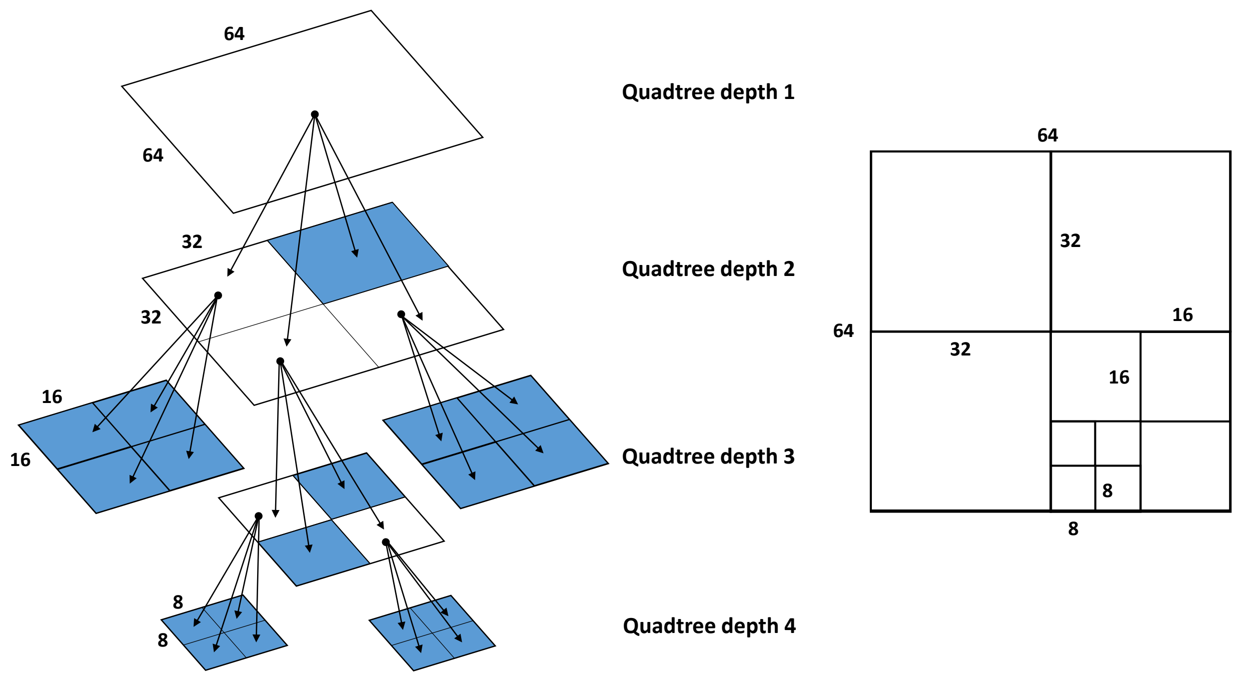


Figure 2.2 – Quadtree structure.

A CU can be further partitioned into Prediction Units (PUs), where intra-, inter-frame, and/or inter-view predictions encode the blocks according to the encoding frame type. The intra-frame prediction encodes a PU using the information contained inside the encoding frame. The inter-frame prediction explores the temporal redundancy among frames; therefore, past encoded frames in the same view are used as the reference to perform predictions. The inter-view prediction is similar to the inter-frame prediction; however, it uses other views inside the access unit as a reference, instead of a past encoded frame.

Figure 2.3 shows the possibilities of PU partitioning sizes according to the encoding type. Notice the PUs can assume only quadratic sizes (i.e., blocks with the same width and height) [73] in the intra-frame prediction. While in the inter-frame and inter-view predictions, rectangular partitions are allowed. In all these predictions, the maximum and minimum sizes of the blocks are 64 $\times$ 64 and 4 $\times$ 4, respectively.

The Rate-Distortion cost (RD-cost) is a function that balances the visual quality and the number of bits required for a given tool encode a block. The encoder computes the RD-cost of each block size to select the best tool for encoding the block. By combining the RD-cost of smaller blocks, the encoder can compare several partitions possibilities and find the partition that leads to the smaller RD-cost for the resulting quadtree. One can notice this evaluation requires a high computational effort since there are several encoding modes and block sizes that are evaluated in this quadtree evaluation.

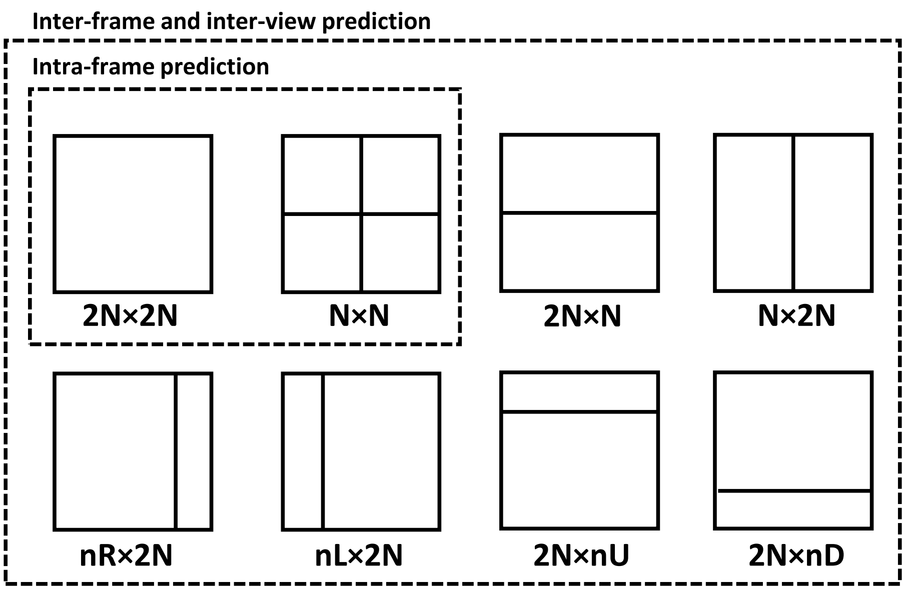


Figure 2.3 – Partitioning of PUs.

2.2 Intra-Frame Prediction

Figure 2.4 shows the dataflow model of the depth map intra-frame prediction for a given depth block used in the 3D-HEVC Test Model (3D-HTM), which is the 3D-HEVC reference software [27]. The best encoding mode is evaluated using its RD-cost, and the mode with the lowest RD-cost is selected to encode the block. Each encoded block is evaluated using: (i) HEVC intra-frame prediction in Transform-Quantization (TQ) and DC-only flows; (ii) bipartition modes (grouping Intra_Wedge and Intra_Contour) in both TQ and DC-only flows, and; (iii) Intra-picture skip mode, which uses only Entropy Coding (EC).

The HEVC intra prediction tool used by 3D-HEVC is the same defined to predict texture in HEVC. Both, Intra_Wedge and Intra_Contour [37] segment the encoding block into two regions, where each region is predicted with the Constant Partition Value (CPV), which contains the average of all values of the region. Intra_Wedge divides the encoding block using a straight line called wedgelet, while Intra_Contour creates a contour segmentation dynamically. The Intra_Wedge and Intra_Contour were mainly designed for achieving high encoding efficiency when encoding edges regions. The Intra-picture skip encoding tool defines four extra prediction modes for obtaining high bitrate reduction at homogeneous regions [31].

The HEVC intra-frame prediction and the bipartition modes use a Rate-Distortion list (RD-list) to define some prediction modes that are further evaluated by their RD-cost, since evaluating all encoding possibilities by their RD-cost implies a prohibitive computation effort. After processing the prediction tools, all modes inserted into the RD-list are evaluated using TQ and DC-only flows. Subsequently, the RD-costs are obtained using entropy coding in the results of TQ and DC-only flows. The TQ flow is the same applied in the HEVC texture

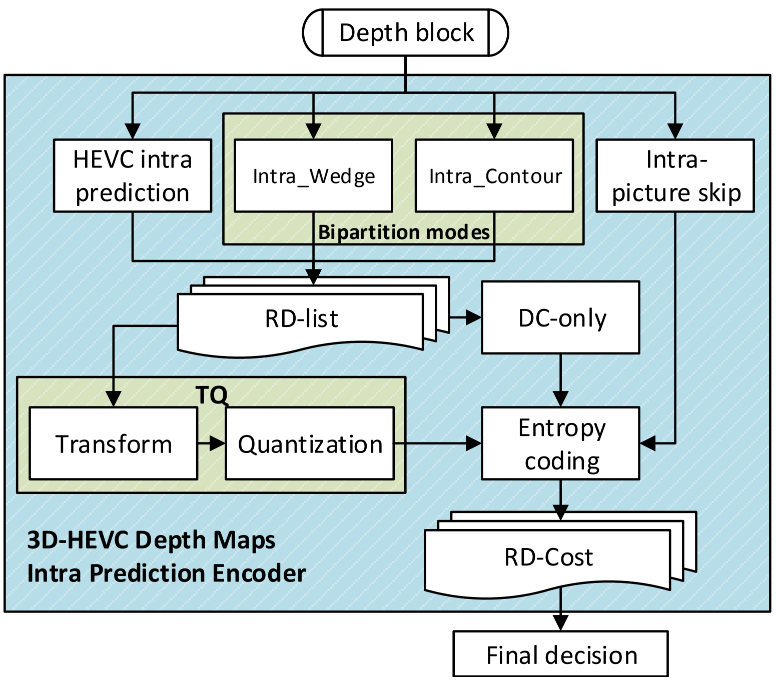


Figure 2.4 – Dataflow model used for the 3D-HEVC depth map intra-frame prediction (source: [60]).

coding, and the DC-only flow [33] has been designed as a new encoding alternative to the TQ flow, focused on obtaining higher efficiency in the homogeneous regions of the depth maps. Besides, the entropy encoder is directly applied in the Intra-picture skip modes to obtain the RD-cost without passing through the RD-list evaluation.

Next subsections detail the depth map intra-frame prediction tools. The common encoding tools between intra-, inter-frames and inter-views predictions (i.e., TQ and DC-only flow, and Entropy Coding) are later described in Section 2.4.

2.2.1 HEVC Intra-Frame Prediction

The 3D-HEVC depth map intra-frame prediction defines the planar mode, DC mode and, 33 directional modes, whose directions are presented in Figure 2.5 [30]. These modes use samples of spatially neighboring blocks as a reference for creating a predicted block.

All available block sizes of the intra-frame prediction apply this encoding tool using a different number of directions; i.e., (i) 18 directions for 4×4 blocks, (ii) 35 directions for 8×8, 16×16 and 32×32 blocks; and (iii) 4 directions for 64×64 blocks.

The most intuitive encoding approach is to evaluate the RD-cost of all available prediction modes, which may be prohibitive for real-time applications. 3D-HTM employs the heuristic proposed by Zhao et al. [85] to encode texture and depth maps. An RD-list is generated (see Figure 2.4) containing few modes among all available ones and RD-cost is

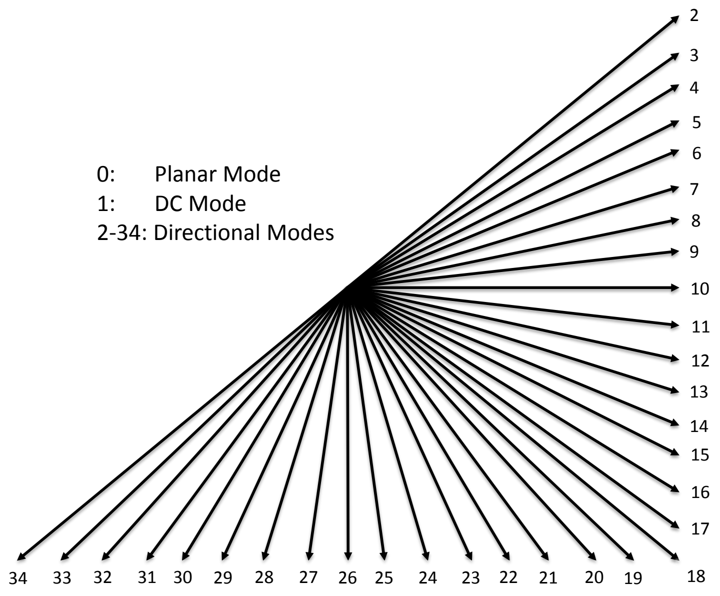


Figure 2.5 – Directional modes of the HEVC intra-frame prediction (source: [30]).

computed only for these selected modes. This heuristic uses two algorithms: Rough Mode Decision (RMD) and Most Probable Modes (MPM). The RMD algorithm applies the Sum of Absolute Transformed Differences (SATD) between the original block and the predicted one to early evaluate all HEVC modes (without the complete RD-cost evaluation). The algorithm orders the modes according to their SATDs and inserts the modes with the lowest SATDs ordered into the RD-list (8 modes for 4×4 and 8×8 blocks, and 3 modes for 16×16, 32×32, and 64×64 blocks). Subsequently, the MPM algorithm gets the information of the modes used in the left and above encoded neighbor blocks and can inserts up to two modes into the RD-list.

2.2.2 The Bipartition Modes

The bipartition modes use the Intra_Wedge and Intra_Contour algorithms for block sizes from 4×4 to 32×32, which produce wedgelet exemplified at Figure 2.6 and contour segmentation presented at Figure 2.7. Notice these modes are not available for 64×64 blocks.

Figure 2.6 (a) shows the Intra_Wedge divides the encoding block into two regions separated by a wedgelet. Since there are samples near the wedgelet that are not entirely inside a region, Figure 2.6 (b) illustrates a discretization performed to select which region these samples belong.

The Intra_Wedge algorithm defines the evaluation of many wedgelets for each block, but only a subset is effectively evaluated, intending to reduce the Intra_Wedge com-

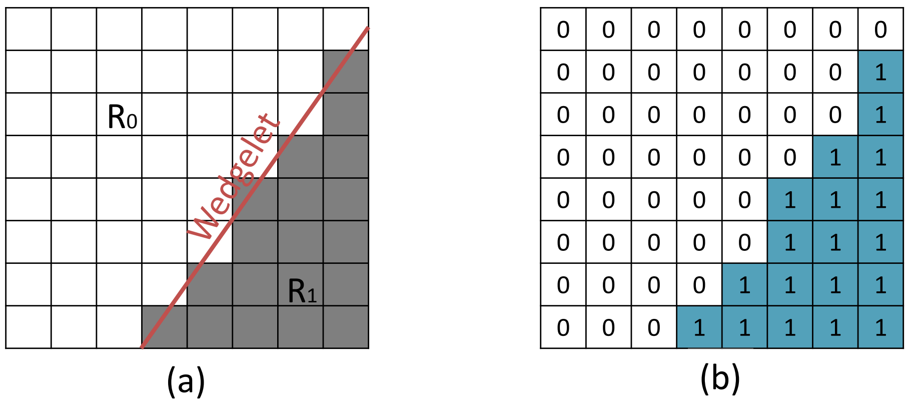


Figure 2.6 – Wedgelet segmentation model of a depth block: (a) Pattern with R_0 and R_1 and (b) discretization with constant values (source: [63]).

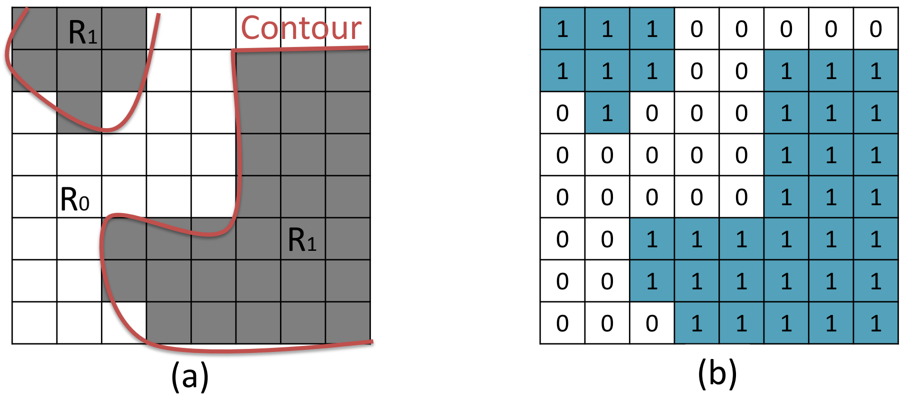


Figure 2.7 – Contour segmentation model of a depth block: (a) Pattern with R_0 and R_1 and (b) discretization with constant values (source: [63]).

putational effort. Table 2.1 shows the possible wedgelets and the storage requirement (in bits) according to the block size.

Table 2.1 – Number of wedgelets evaluated in the Intra_Wedge algorithm.

Block size	**Total number of possible wedgelets**	**Number of evaluated wedgelets in the main stage**	**Storage requirements (bits)**
4×4	86	58	1,376
8×8	802	314	51,328
$\geq 16 \times 16$	510	384	130,560

Figure 2.8 presents a high-level diagram of the Intra_Wedge encoding algorithm as applied in 3D-HTM. Three stages compose the Intra_Wedge algorithm: (i) Main, (ii) Refinement and (iii) Residue. The Main Stage evaluates the initial wedgelet set (i.e., wedgelets assessed before the refinement) and finds the best wedgelet partition among the available ones. The process of finding the best wedgelet requires mapping the encoding block into the binary pattern defined by each wedgelet. According to this mapping, the average values of all samples mapped into regions R_0 and R_1 are computed, and the predicted block is defined as the average value of each region (Prediction step). For each wedgelet pattern, a similar-

ity criterion is applied to determine the encoding efficiency of this pattern compared to the other wedgelet patterns, generating a local distortion value. All distortions are compared, and the pattern with the lowest distortion is selected as the best wedgelet (Distortion step). There are many possibilities for similarity criterion usage, such as Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), and Synthesized View Distortion Change (SVDC) [77]. Each similarity criterion can obtain different encoding impact and encoding effort when profiling. 3D-HTM uses the SVDC algorithm as default [77].

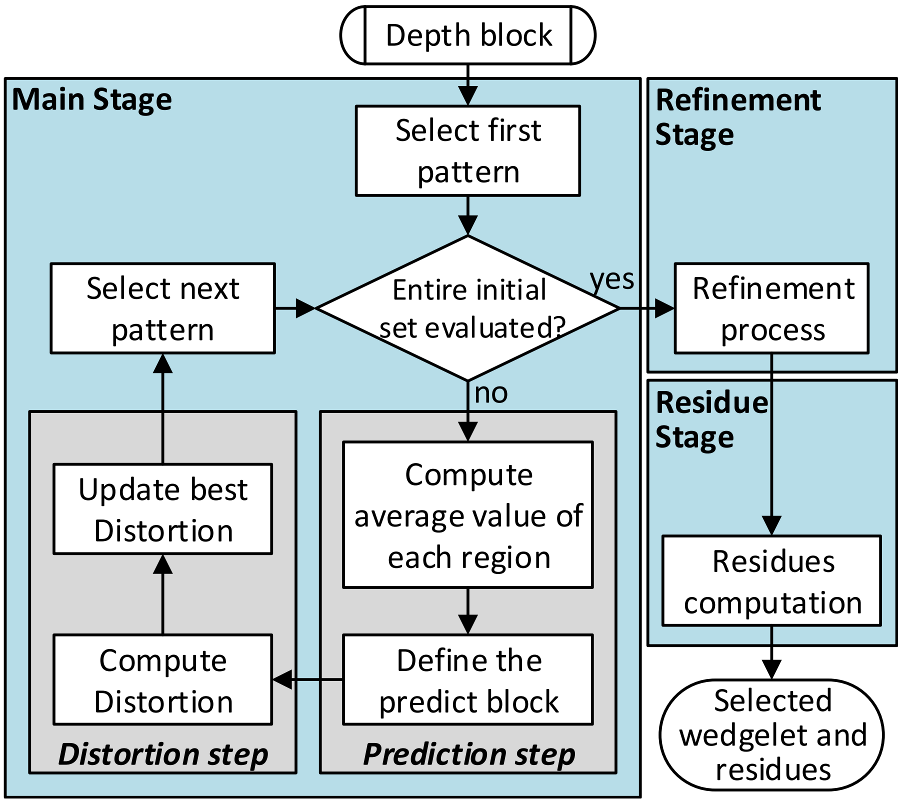


Figure 2.8 – Main blocks of the Intra_Wedge encoding algorithm (source: [63]).

The Intra_Wedge Refinement Stage evaluates up to eight wedgelets with patterns similar to the one selected in the previous operation. Again, the wedgelet with the lowest distortion among these eight possibilities, along with the wedgelet selected in Main Stage, is selected as the best one. Finally, the Residue Stage subtracts the predicted block of the elected wedgelet from the original one and adds this wedgelet into the RD-list.

While Intra_Wedge assumes only pre-defined patterns, Intra_Contour employs a contour segmentation that can model arbitrary patterns and even consist of several parts (but only two regions with constant values are allowed). Figure 2.7 (a) exemplifies the contour segmenting a block into two regions and Figure 2.7 (b) presents the contour discretization with the constant value of each region. Intra_Contour is not purely an intra-frame prediction algorithm because it uses a technique called inter-component prediction to find the best contour partition, using previously encoded information from one component (i.e., texture) during the prediction of other components (i.e., depth maps) [40].

JCT-3V experts designed the Intra_Contour algorithm motivated by the fact the depth block represents the same scene of the texture block, which is already encoded. Although depth maps and texture contain different characteristics, they have high structural similarities that can be explored. For instance, an edge in the depth component usually corresponds to an edge in the texture component. Figure 2.9 highlights these similarities and correlation.

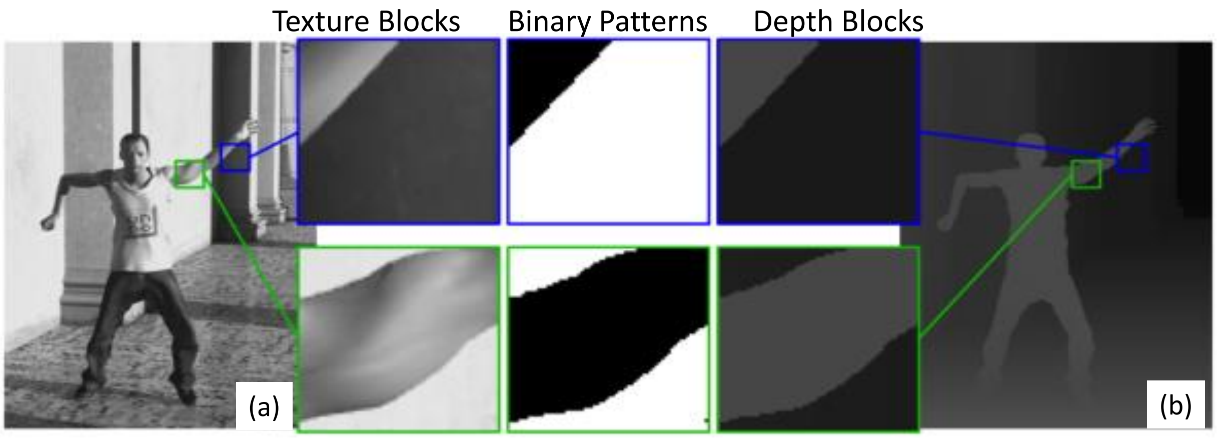


Figure 2.9 – Correlation between (a) texture and (b) depth components (source: [8]).

Figure 2.10 portrays the Intra_Contour encoding flow. The only information known at the beginning of Intra_Contour execution is the texture and the depth blocks. Besides, only the reconstructed luminance signal (previously encoded) composes the texture block used as a reference in the Intra_Contour algorithm. Three stages compose the Intra_Contour algorithm: Texture Average, Prediction, and Residue. The Texture Average Stage starts computing the average value (u) of the corners samples in the texture block. The pseudo-code described in Figure 2.11 is applied to construct a binary map that determines the Intra_-Contour contour partition. From this pseudo-code, one can conclude all texture samples smaller than u are mapped into R_0, while the others are mapped into R_1, generating a binary map.

The Intra_Contour encoding does not require to use depth information during the binary map generation. Besides, the Intra_Contour pattern is not transmitted because the decoder can use the texture block, which is decoded before the depth block, and adaptively generate the Intra_Contour pattern at the decoder side, without increasing the bitrate.

The Prediction Stage processes the depth maps using the binary map determined in the previous stage and calculates the average values of all depth block samples mapped in the regions R_0 and R_1. These average values are selected as the predicted block, according to the binary map value.

Finally, the Residue Stage of Intra_Contour subtracts the predicted block from the original depth block, generating the residues and inserting this mode into the RD-list.

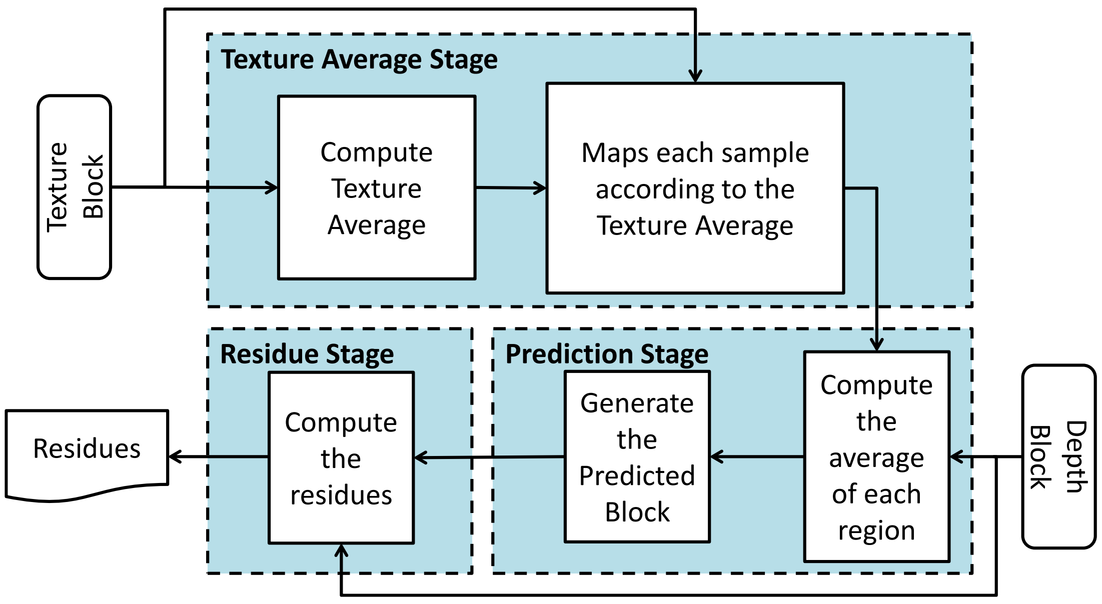


Figure 2.10 – Intra_Contour encoding flow (source: [63]).

```
1.  for i = 1 to N
2.      for j = 1 to N
3.          if sample[i][j] < u
4.              $R_0 \leftarrow$ sample[i][j]
5.          else
6.              $R_1 \leftarrow$ sample[i][j]
```

Figure 2.11 – Pseudo-code representing the mapping of samples in the regions R0 and R1 in the Intra_Contour algorithm (source: [63]).

The bipartition modes can produce CPVs that require several bits for transmission and storage. The 3D-HEVC standard adopted the delta CPV encoding, which computes the predicted CPV using the information of the neighbor pixels to mitigate this problem. Figure 2.12 shows the samples used in delta CPV computation, where the encoding block is detached in red.

The CPV prediction requires the following items: (i) binary pattern information of the three corners positions (P0, P1, and P2); (ii) three samples of the upper PU block (TL, TM, and TR); (iii) one sample from the upper right PU (TRR); (iv) three samples from the left PU (LT, LM, and LB); and (v) one sample from the left bottom PU (LBB). When some of these samples are not available (i.e., the necessary neighbor pixel has not been codified yet), the nearest available sample is used instead of that.

Figure 2.13 illustrates the pseudo-code for delta CPV generation that assigns the references ref_0 and ref_1 to the correct prediction CPV (i.e., pred_0 and pred_1). Case 1 arises when the three binary patterns are equal; then, ref_0 is computed using the average value of TL and LT. In this case, ref_1 is obtained from LBB or TRR, according to the highest absolute difference between TRR and TL or LBB and LB. Cases 2 occurs when P0 is

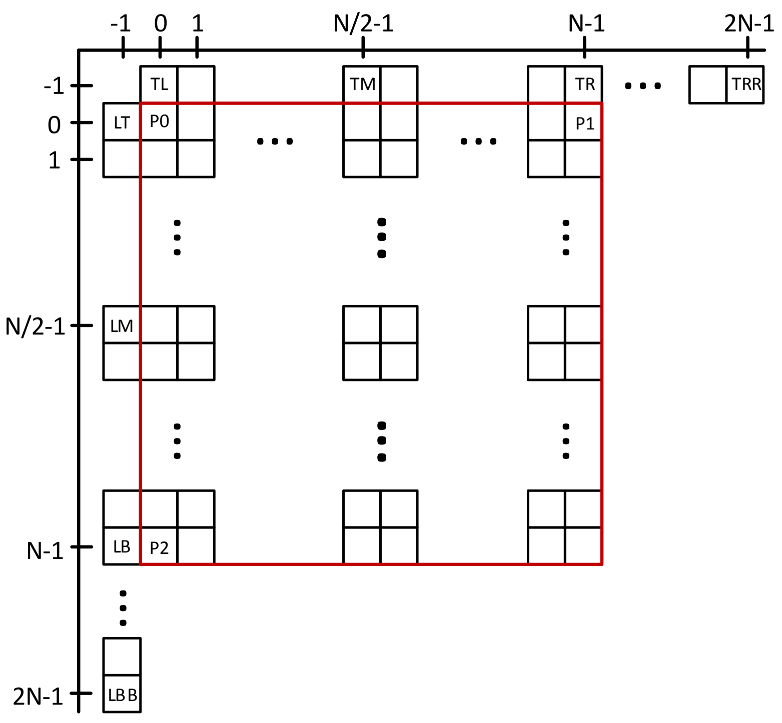


Figure 2.12 – Encoding block and its boundary samples (source: [56]).

different of P1 and P2; then, ref_0 and ref_1 are computed using the average value of the pairs (TL, LT) and (TR, LB), respectively. Case 3 arises when P0 is equal to P1 but differs from the pattern of P2. In this case, TM is selected as ref_0 and LB as ref_1. Finally, Case 4 occurs when P0 is equal to P2 but differs to the pattern of P1; then, LM is selected as ref_0 and TR as ref_1.

```
if P0 = P1 = P2                          → Cases 1a, 1b
   ref_0 ← average(TL, LT)
   if abs(TRR – TL) > abs(LBB – LB)
      ref_1 ← TRR                        → Case 1a
   else
      ref_1 ← LBB                        → Case 1b
elsif P0 ≠ P1 and P1 = P2                → Case 2
   ref_0 ← average(TL, LT)
   ref_1 ← average(LB, TR)
else
   ref_0 ← P0 ≠ P2 ? TM : LM             → Case 3 (TM, LB)
   ref_1 ← P0 ≠ P2 ? LB : LR             → Case 4 (LM, LR)
pred_0 ← P0 ≠ P2 ? ref_1 : ref_0
pred_1 ← P0 ≠ P2 ? ref_0 : ref_1
```

Figure 2.13 – Pseudo-code for delta CPV computation (source: [56]).

The delta CPVs are computed by subtracting the original CPV from the predicted CPV and only delta CPVs are transmitted in the bitstream. In the decoding process, the predicted CPV can be computed using the same algorithm employed in the encoding process. It requires accessing the wedgelet memory to obtain P0, P1, and P2 values, or generating the

DMM-4 pattern dynamically to request the same sample values of neighbor samples used by the encoder. The predicted CPV is added with the delta CPV, recovering the original CPV.

2.2.3 Intra-Picture Skip

Intra-picture skip is a novel tool of the 3D-HEVC depth map intra-frame prediction, which was developed considering most of the depth maps are smooth areas for block sizes ranging from 8×8 to 64×64. In general, slight changes of depth values in smooth areas are not relevant to synthesized views quality [31]. Thus, Intra-picture skip ignores the residual coding for the smooth areas providing a significant bitrate reduction.

Intra-picture skip includes four prediction modes [31]: (i) Mode 0 (vertical intra-frame), which copies all neighbor upper samples to the current encoding block; (ii) Mode 1 (horizontal intra-frame), which copies all neighbor left samples to the entire current encoding block; (iii) Mode 2 (single vertical depth), which copies only the middle upper neighbor sample to the entire current encoding block and; (iv) Mode 3 (single horizontal depth), which copies only the middle left neighbor sample to the entire current encoding block. Figure 2.14 (a) to (d) exemplify the Intra-picture skip modes 0, 1, 2 and 3, respectively, regarding an 8×8 encoding block; the red boxes defines the predicted blocks. Only the N above and N left neighbor samples are used in the prediction (considering N^2 the number of samples of the square encoding block). The HEVC vertical (Figure 2.5, direction 26) and horizontal (Figure 2.5, direction 10) intra directions are used to generate Modes 0 and 1. The Modes 2 and 3 fill the predicted block with a single depth value derived from the upper and left neighbor blocks, respectively. By using Intra-picture skip, it is not necessary to transmit the encoding block nor extra information. Only side information is required to be entropy coded and packed in the bitstream, providing significant bitrate reduction.

2.3 Inter-Frame and Inter-View Predictions

Figure 2.15 shows the Motion Estimation (ME) and the Disparity Estimation (DE) referencing other frames. Notice ME explores the temporal redundancies among frames, while DE searches for the redundancies among views, i.e., inside an access unit.

Similarly to texture inter-frame/view prediction, ME/DE is applied in depth map prediction using a search algorithm to detect the block that contains the most similarity to the encoding block inside a search area in the reference frame, as displayed in Figure 2.16. It is performed mapping the movement/disparity from the current encoding block in a reference frame using a 2D vector that points to the most similar block in the reference frame by apply-

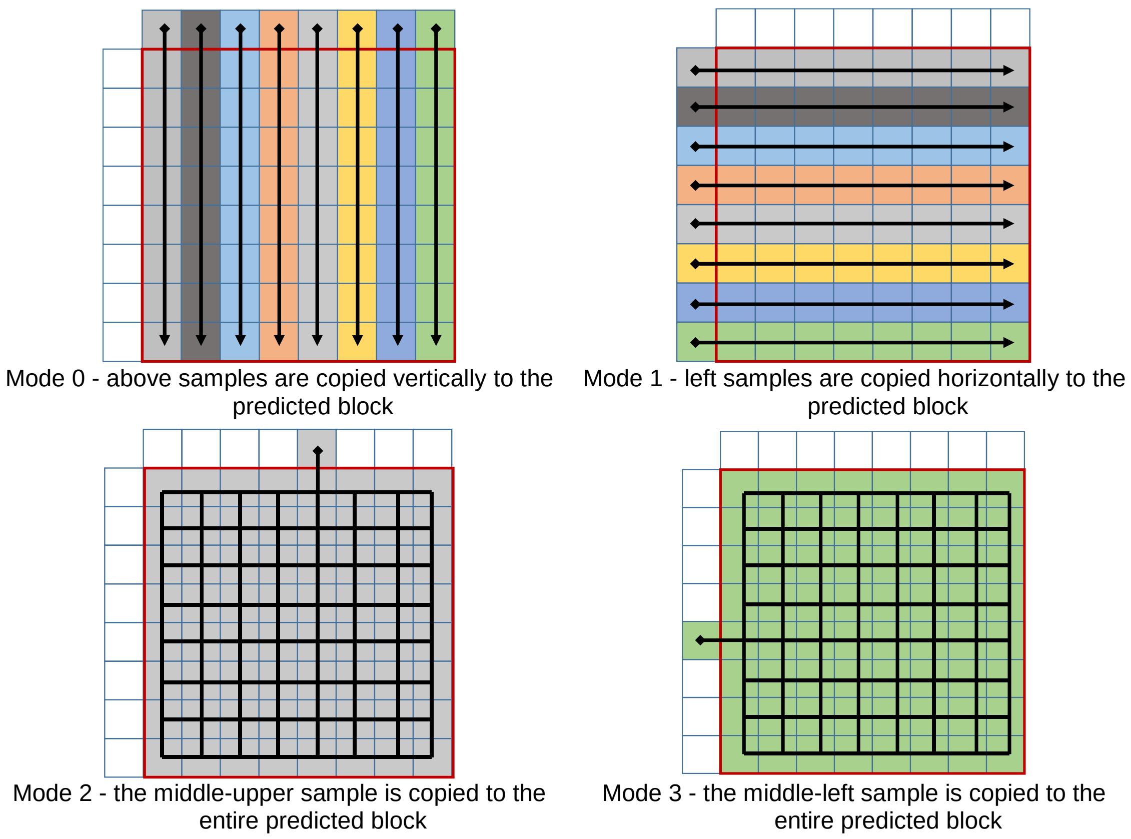

Mode 0 - above samples are copied vertically to the predicted block
Mode 1 - left samples are copied horizontally to the predicted block
Mode 2 - the middle-upper sample is copied to the entire predicted block
Mode 3 - the middle-left sample is copied to the entire predicted block

Figure 2.14 – Example of Intra-picture skip encoding modes for an 8×8 block (source: [69]).

ing a search process, which requires a considerable encoding effort. Therefore, the search process is a critical task inside block-based video encoders [9].

3D-HTM uses Test Zone Search (TZS) as the ME/DE search algorithm and SAD as the similarity criterion. The TZS algorithm can reach a near optimal performance (i.e., the TZS can obtain results very similar to a brute force approach) regarding the quality of the search process. TZS employs an Initial Search Point (ISP) decision and then performs an iterative approach around the best ISP.

TZS selects the best ISP comparing the SAD results of the: (i) vector pointing to the co-located block (i.e., the block in the same position in the reference frame); (ii) median vector of the encoding block neighborhood (composed of previously encoded blocks around of the current block); and (iii) vector of the largest block for the current CU [74]. Besides, TZS uses different patterns in the search process, such as expansive diamond, raster search and refinement step [74]. The ISP technique combined with these search machines allows reducing around of 23 times the Full Search (FS) (i.e., the brute force approach) encoding time without significant impact on the encoded video quality [74]. Even so, TZS still requires a considerably high number of SAD comparisons when compared to lightweight algorithms.

The inter-frame/view prediction also allows the usage of Merge/Skip Mode (MSM), where the encoder derives information of spatial or temporal neighbor blocks. The Merge

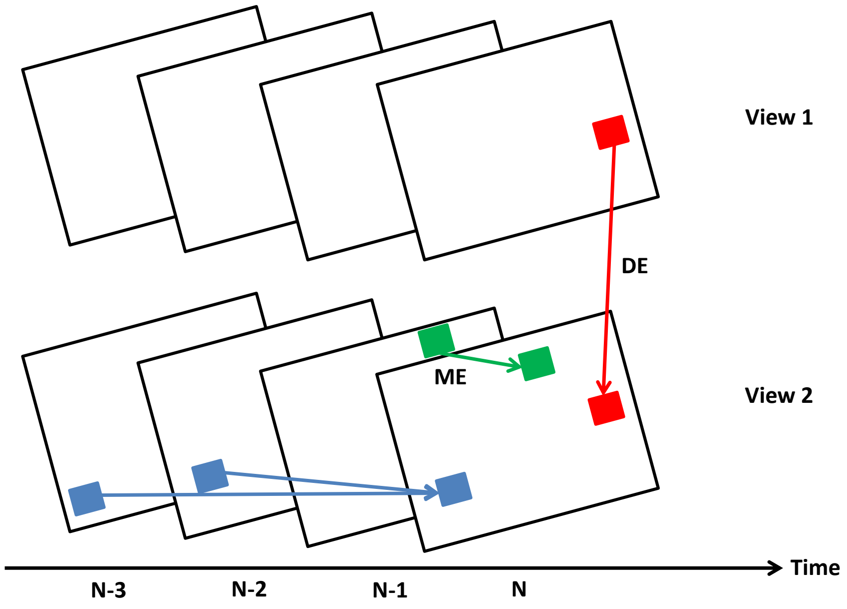


Figure 2.15 – Example of ME/DE referencing blocks from other frames.

mode uses a merge list containing the neighbor blocks, whose index is used during the encoding process. The Skip mode is a particular case of MSM, where no residual information is required, reducing the final stream size significantly. Therefore, the encoder uses one of the following information for every inter-frame PU: (i) explicit encoding of motion parameters; (ii) motion merge mode; or (iii) Skip mode.

2.4 Common Encoding Steps: Transform-Quantization, DC-only and Entropy Coding

After the execution of the prediction steps described in Sections 2.2 and 2.3, the depth map encoding finishes using three additional tools described next: TQ, DC-only, and Entropy Coding (EC).

2.4.1 Transform-Quantization (TQ)

The TQ flow of the 3D-HEVC depth maps uses the same process applied in the HEVC texture coding; this flow divides the CU in a quadtree of Transform Unities (TUs) [79], which allows TQ to range from 4×4 to 32×32 samples. Discrete Cosine Transform (DCT) and a quantization module process the residual information of the predicted block under evaluation. So, when encoding texture it attenuates or removes the less relevant frequencies to the human vision [6]; thus, generating small quality losses in the coding process. A Quantization Parameter (QP) defines the quantization strength, and as bigger is

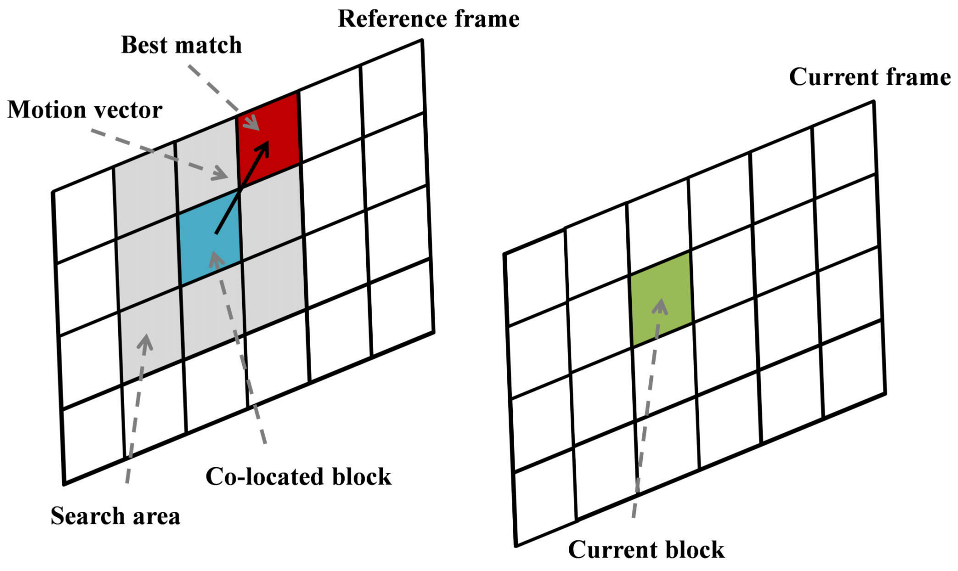


Figure 2.16 – Example of ME/DE search process (source: [65]).

the QP as higher is the compression rate, but also as higher is the image quality degradation [6].

2.4.2 DC-Only Coding

As an alternative to TQ flow, 3D-HEVC uses DC-only to obtain further bitrate gains in smooth regions [33] for block sizes ranging from 8×8 to 64×64, allowing only quadratic blocks. Instead of using the residual information of depth samples, DC-only considers a single DC residue encodes the HEVC intra-frame, inter-frame, and inter-view predictions, and considers two DC residues encode the bipartition modes. The DC residue generation requires to compute a predicted DC (dc_p), the original DC (dc_o), and then to perform a subtraction operation.

It uses the average value of the original block for HEVC intra- and inter-frame predictions, and the average value of each region for the case of bipartition modes to calculate dc_o. The use of the average value of the four corners samples of the predicted block for intra-frame prediction generates dc_p; additionally, dc_p uses the predicted CPV obtained by applying the procedure described in Subsection 2.2.2 for bipartition modes.

2.4.3 Entropy Coding

The depth map EC uses the same tools defined in the conventional HEVC texture encoding process. Context Adaptive Binary Arithmetic Coding (CABAC) [36] is the primary tool used, which provides a significant bitstream reduction by performing lossless entropy compression in the series of syntax elements delivered by the previous encoding tools. Thus, EC processes all information generated by the previously described encoding tools. The EC results allow calculating the RD-cost of all possible modes, enabling to compare those results and select the one that will be used to encode the current block.

2.5 Evaluation Methodology - Common Test Conditions (CTC)

The Common Test Conditions (CTC) [41] were developed by JCT-3V experts to allow a fair comparison among the 3D video coding works. CTC includes eight different video sequences that should be encoded in four quantization scenarios. Table 2.2 presents a summary of these sequences characteristics.

Table 2.2 – Test sequences details.

Test Sequence	**Resolution**	**Total Frames Encoded**	**Environment**	**Sequence Details**	**Depth Structure**
Balloons	1024 × 768	300	Inside a room	Medium	Medium
Kendo	1024 × 768	300	Inside a room	High	Medium
Newspaper_CC	1024 × 768	300	Inside a room	High	Medium
GT_Fly	1920 × 1088	250	Computer Graphics	Medium	Complex
Poznan_Hall2	1920 × 1088	200	Inside a room	Medium	Complex
Poznan_Street	1920 × 1088	250	Outside	High	Complex
Undo_Dancer	1920 × 1088	250	Computer Graphics	High	Simple
Shark	1920 × 1088	300	Computer Graphics	Medium	Complex

Each video sequence possesses specific characteristics such as high movement, transparency, details and camera noise. Figure 2.17 and Figure 2.18 present the first texture frame and the depth map of the central view, respectively. The following set of videos are evaluated in CTCs: (a) Balloons [75], (b) Newspaper_CC [25], (c) Kendo [75], (d) GT_Fly [84], (e) Poznan_Hall2 [15], (f) Poznan_Street [15], (g) Undo_Dancer [46], and (h) Shark [42].

The experts designed the 3D-HTM [24], which is a 3D-HEVC reference software that provides a reference software implementing the 3D-HEVC encoder, used to evaluate these 3D video sequences following CTC. 3D-HTM allows obtaining relevant information, such as texture encoding time, depth encoding time, bitrate for each encoded view, and the resulting Peak Signal-to-Noise Ratio (PSNR), i.e., subjective video quality, of each view. After encoding a video with 3D-HTM, it is possible to use another part of this reference software containing a 3D-HEVC decoder and decode the views, and posteriorly generate

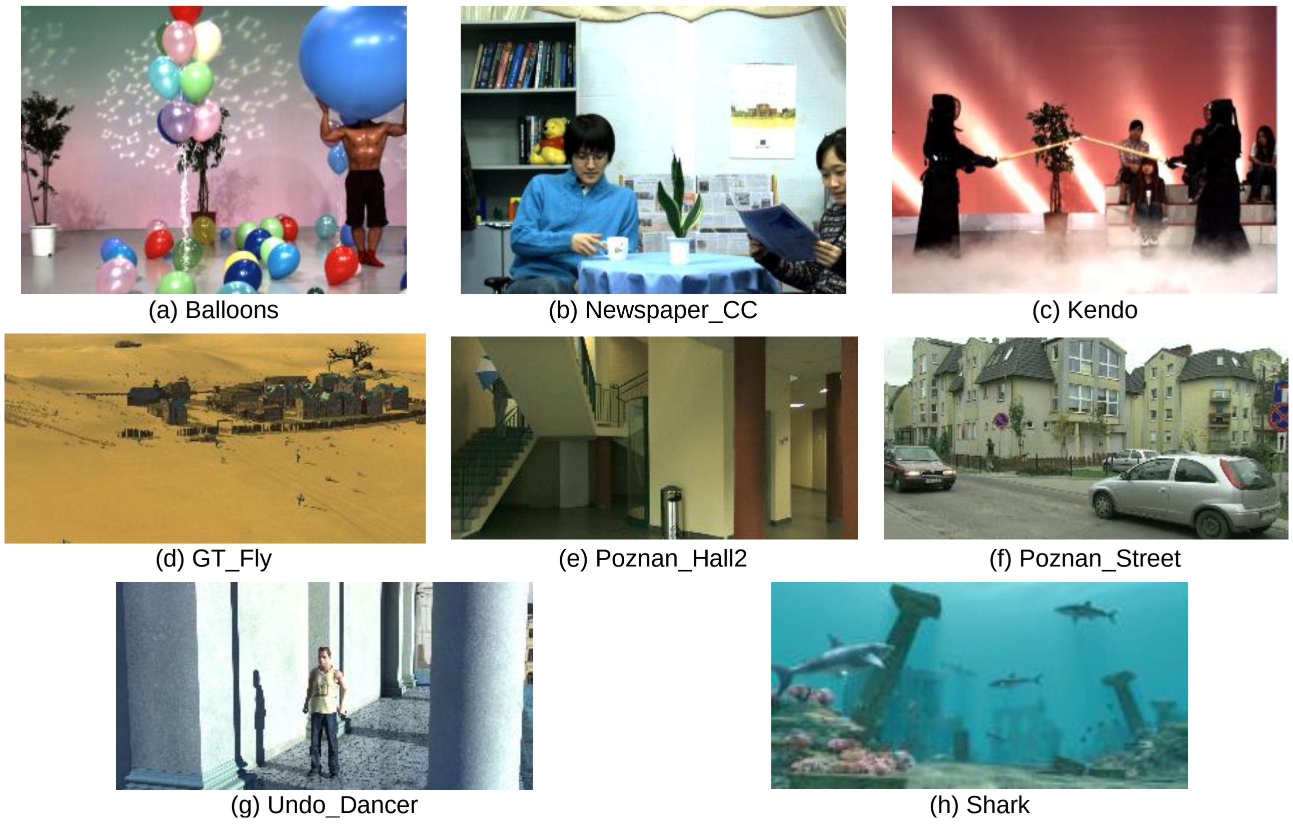


Figure 2.17 – First frame of the texture of each video present in CTC.

intermediary view by using a view synthesis interpolator. Consequently, one can evaluate the visual quality of the synthesized view and compare with a related work fairly.

When following CTC definition, the 3D-HTM is evaluated using four QP pairs (QP-texture, QP-depth) for each video sequence: (25, 34), (30, 39), (35, 42), and (40, 45), where the QP indicates the quantization impact in the encoding process. The higher the QP value, the higher are the reductions in the video quality, also resulting in a bitrate reduction.

The evaluation of these four QP pairs allows drawing Rate-Distortion (RD) curves to compare different encoding algorithms. Moreover, the technique proposed by G. Bjontegaard [4] can be applied to obtain the Bjontegaard Delta rate (BD-rate). This BD-rate metric indicates the bitrate reduction for similar video quality and it is computed by interpolating the resulting Peak Signal-to-Noise Ratio (PSNR) and the bitrate obtained in one experiment and comparing this result with a baseline implementation. However, in depth maps coding, the PSNR of the depth encoded view does not provide relevant information for analysis consequently, the BD-rate of the depth maps are not relevant. Therefore, in MVD systems, for computing the BD-rate it is necessary to synthesize intermediary views and use their PSNR as the reference for this evaluation. So, in this Thesis, the evaluated BD-rate in all experiments is obtained using the average PSNR of the synthesized views and considering the total bitrate, i.e., the sum of the bitrate of all texture views and depth maps.

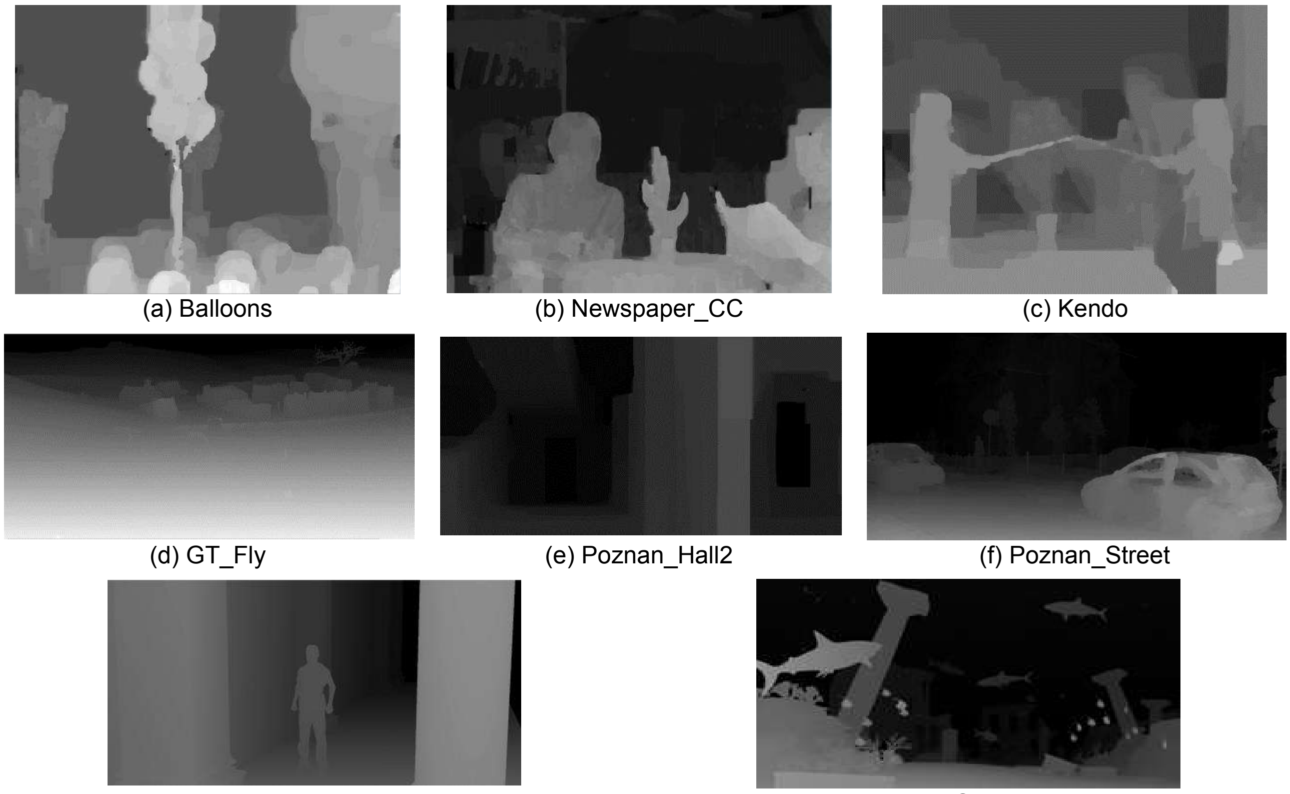


Figure 2.18 – First frame of depth maps of each video present in CTC.

Table 2.3 shows the main configurations applied to the experiments of this Thesis. These configurations are in accordance with the CTC definitions.

Table 2.3 – Configurations used in the experiments.

Profile	**All Intra**	**Random Access**
GOP size	1	8
Intra-period	1	24
Bits per pixel	8	
Texture/Depth views	3/3	
QP-pair	(25, 34), (30, 39), (35, 42), (40, 45)	
DMM evaluation	Enabled	
Depth intra skip	Enabled	
VSO	Enabled	
Rate control	Disabled	
3D-HTM version	16.0	

3. RELATED WORK

This chapter describes works targeting four topics of depth map coding. Sections 3.1 and 3.2 discuss the works aiming to reduce the computational effort of the intra-frame and inter-frame/view predictions, respectively. Section 3.3 shows works focused on controlling the encoding effort. Finally, Section 3.4 describes the main related work focusing on designing hardware and/or dealing with the 3D-HEVC memory requirements.

Sections 3.1 and 3.2 classify the timesaving algorithms in (i) mode-level, (ii) block-level, and (iii) quadtree-level. The timesaving technique of the mode-level focuses on a single encoding mode. The block-level algorithms aim at reducing the complete evaluation of some encoding steps if a given criterion is met. For example, if a block and its neighbor are highly correlated with the current encoding block (i.e., highly similar characteristics), then it is necessary only to evaluate the current block by the encoding mode used in the neighbor block, skipping the remaining possibilities. The quadtree-level algorithms limit the minimum and maximum quadtree-level adaptively according to the encoding region. For example, if the encoding region is almost homogeneous, there is no reason to evaluate small block sizes since a higher block size can represent nearly the same information without influencing the quality and providing a smaller bitrate. Besides, coding an area with many details is more difficult when using large block sizes.

3.1 Related Work on Reducing the Computational Effort on the Intra-Frame Prediction

Several works detail solutions for reducing the computational effort of the 3D-HEVC depth map intra-frame prediction. The mode-, block-, and quadtree-level decision algorithms are presented in Subsections 3.1.1, 3.1.2, and 3.1.3, respectively.

3.1.1 Related Work in Mode-Level Decision Algorithms

Regarding mode-level decision algorithms, some works propose to reduce the encoding effort of RMD and MPM selection (e.g., [82] and [83]), some other works propose to diminish the encoding effort of the TQ and/or DC-only flows (e.g., [20], [44], and [83]), and others focus on reducing the wedgelet list evaluations (e.g., [18] and [82]). We have also proposed two previous [52] and [66] focusing on reducing the wedgelet list evaluation.

The work proposed by Zhang et al. [82] simplifies the RMD and MPM selection, and the Intra_Wedge wedgelet list evaluation. It classifies the current block in one of three types using the variance value of neighbor reference samples: (i) smooth, (ii) edge, or (iii) normal. A block classified as smooth requires to insert into RD-list only the planar mode; if the block is classified as an edge, the algorithm uses the orientation of the intra-frame direction to decide the modes inserted into RD-list. It also uses the orientation of Intra_-Wedge patterns to avoid evaluating the entire Intra_Wedge set. If the block is classified as normal, the original 3D-HTM encoding flow is performed. It reached a timesaving of 27.9% in the depth maps coding while increased the BD-rate in 1.03%.

The work [83] performs mode- and block-level (described in Subsection 3.1.2) decisions. The mode-level decision always inserts into RD-list the modes 10 (horizontal), 26 (vertical), planar, DC, and bipartition. The mode-level decision also verifies the parental block (a larger block containing current block) encoding mode. If the planar, DC or bipartition modes do not encode its parental block, then the mode-level decision also inserts two extra intra-frame prediction modes into RD-list. Otherwise, only the modes initially inserted into the RD-list are evaluated. Its mode-level decision was capable of achieving a reduction of 24.4% in depth maps coding with an increase of 0.51% in BD-rate.

The work [44] proposed a timesaving algorithm in the TQ and DC-only flows using mode-, block- and quadtree-level decisions. Since its mode- and block-level decisions work together, they are explained together here, while its quadtree-level decision is explained in Subsection 3.1.3. In step 1, it evaluates only planar, DC, horizontal, vertical and MPM intra-frame prediction modes and compares the obtained results with a threshold. The process stops if the threshold criterion is met, skipping the remaining HEVC intra prediction directions and the bipartition modes evaluation; else, step 2 is executed, where bipartition modes are evaluated generating a new RD-cost. If the new RD-cost of the bipartition modes is smaller than the previous RD-cost obtained in step 1, then the process finishes. On the contrary, step 3 is executed, computing the RD-cost for the remaining intra-frame prediction modes that share a similar direction to the best wedgelet obtained in Intra_Wedge. Again, the obtained RD-cost is compared with the RD-cost obtained in Step 1, ending the process when the new RD-cost is smaller. Otherwise, all the remaining modes are evaluated. This timesaving scheme obtained 21.6% with a BD-rate increase of 0.5%.

Gu et al. [20] is a default algorithm implemented in 3D-HTM for simplifying the DC-only evaluation without modifying the TQ encoding flow. The three best-encoded modes in TQ flow are stored during their assessment. If the best-encoded mode is planar or DC and the encoding block variance is lower than a threshold, then only planar and DC modes are evaluated in the DC-only flow. If the best-encoded mode by TQ flow is not planar or DC, then only the three best-encoded modes are evaluated by their RD-cost in the DC-only flow. Since it is already executed by default in 3D-HTM, then its encoding time is considered as the baseline for the remaining evaluations. However, before being integrated with the reference software, it was capable of reducing the encoding time in 17.0% with an increase of 0.46% in BD-rate.

The remaining mode-level decision algorithms focus on reducing the number of wedgelets evaluated by the Intra_Wedge. Our previous work [66] propose a gradient filter in the borders of the block to detect the most promising positions and to evaluate Intra_Wedge patterns. It requires evaluating only wedgelets that touch the positions with high gradients values in the borders. Another of our previous work [52] present three further advances providing higher wedgelets evaluation skips when analyzing the two highest gradients of different borders and evaluating wedgelets that touch near positions. The results encoding timesaving in these evaluations vary between 4.9% and 8.2% with a BD-rate increase between 0.33% and 1.47% when evaluated in Random Access configuration.

Fu et al. [18] propose dividing the encoding block into two regions using a vertical wedgelet, a horizontal wedgelet, a wedgelet with 45º and a wedgelet with 135º passing in the middle of the block. In each division, the variance of each region is computed aiming to verify if each region has a strong potential of containing an edge region. With that analysis, Intra_Wedge wedgelet search process is reduced by only evaluating wedgelgets with a high potential. This technique was capable of reducing 52.9% of the evaluated wedgelets with an increase of 0.49% in BD-rate. Fu et al. [18] do not present any evaluation regarding the timesaving provided by their solution.

3.1.2 Related Work in Block-Level Decision Algorithms

Regarding block-level decision algorithms, the work [19] and our previous work [67] proposed skipping the entire bipartition modes evaluation when the traditional HEVC intra-frame prediction encodes the block effectively. Therefore, these works do not search for the best Intra_Wedge wedgelet and do not generate the Intra_Contour pattern when the skip is performed. Besides, some works focus on removing the TQ flow evaluation [83], and on encoding only one flow [10] if a given criterion is met.

The work of Gu et al. [19] compute the variance of the encoding block and use a threshold based on QP for performing fast decisions (algorithm used by default in 3D-HTM). If the encoding block has a low variance or if the best mode in RD-list inserted by HEVC intra-frame prediction is planar, then the bipartition modes are rarely selected. In this case, Gu et al. remove the bipartition modes evaluation. Since it is already executed by default in 3D-HTM, then its encoding time is considered as the baseline for the remaining evaluations. Before its integration with the reference software, it was capable of reducing the encoding time in 26.5% with an increase of 0.02% in BD-rate.

Our previous work [67] explore an approach similar to that described in [19]; however, instead of using the variance of the encoding block, it uses a Simplified Edge Detector (SED), which compares the highest difference among the four corners samples with a threshold decided according to the block size and video resolution. If the value is lower than the threshold, then the encoding block represents a nearly constant region, and the bipartition modes evaluation is skipped. When evaluated in random access configuration, it was capable of reducing the BD-rate in 0.064% and also reducing the encoding time in 5.9% considering the whole encoder and 23.8% considering only depth maps encoding time.

Zhang et al. [83] analyze the correlation between the parental PU and the children PUs (i.e., smaller blocks inside the given PU) mode selection. It concluded that when the parental PU best results was obtained using DC-only, then more than 94% of its child PUs are also encoded with DC-only. Therefore, Zhang et al. [83] propose a block-level to skip only the TQ flow in the smaller blocks evaluation if the current block parent is DC-only coded. When this decision was integrated with their mode-level described in Subsection 3.1.1, it was capable of reaching a depth maps encoding time reduction of 32.87% with a BD-rate increase of 0.54%.

Conceição et al. [10] propose to avoid evaluating the remaining encoding modes according to the RD-cost obtained by Skip and Intra-picture skip modes. The main idea is using the RD-cost obtained in these modes as a decision to avoid the computation of the remaining encoding modes. Since the proposed technique requires working with Skip and Intra-picture skip modes together, then it will be better described in Section 3.2 because Skip mode only works in random access configuration. Besides, the work does not mention how intra encoding frames are evaluated.

3.1.3 Related Work in Quadtree-Level Decision Algorithms

There are works proposing a quadtree-level decision algorithm [7], [44], and [81].

Peng et al. [44] propose a quadtree-level decision that compute the block variance of the current block. If the variance of the current block is higher than a threshold that is defined according to the encoding QP, then it allows splitting the quadtree into smaller blocks. Otherwise, the variance of the four subblocks inside the encoding block are computed. Then the algorithm decides to stop the quadtree expansion when the highest variance in these subblocks are lower than the current block variance. Otherwise, the quadtree expansion is performed. Together with the mode decision explained in Subsection 3.1.1, Peng et al. [44] was capable of reducing 37.6% in the encoding time with 0.8% of BD-rate increase.

H. Zhang et al. [81] propose a QP-based depth quadtree limit for splitting the quadtree only when the encoding block contains relevant information for smaller blocks usage. Additionally, the authors use a classical corner detection algorithm [22] to detect if the information is relevant for smaller block sizes. It achieved a reduction of 41% in the encoding time with an increase of 0.44% in BD-rate.

H. Chen et al. [7] proposed an early termination algorithm for intra CU splitting. When a CU is split into four new CUs, they check if the first sub-CU selects Intra-picture skip as the best encoding mode. Case it is encoded using intra-picture skip, the RD_cost obtained in the parental CU is compared to four times the RD-cost obtained by Intra-picture skip mode. In the case the RD-cost of the parental CU is smaller, then the parental CU is used in the encoding process and no more evaluation is required. Otherwise, the remaining sub-CUs are evaluated and the sum of all sub-CU costs is compared with the parental CU cost. Again, in case the parental CU cost is smaller, then it is selected as the best partition. It reduced 45.1% of depth map encoding time with a BD-rate increase of 0.09%.

3.1.4 Summary of Intra-Frame Prediction Related Work

Table 3.1 presents a summary of the related work where the focused tool are highlighted for mode-level, the skipped tools are highlighted for block-level and it is detached the quadtree-level characteristic. Among the related work, we can see that there were several distinct characteristics that were explored in depth maps coding. In mode-level decisions, the highest focused tool was Intra_Wedge due to its high encoding effort associated with it. In block-level decisions, skipping bipartition modes was the most focused encoding tool. And only a few works focused on designing quadtree-level decisions for depth maps coding; however the highest gains regarding depth maps timesaving where achieved in these works.

Table 3.1 – Summary of related work on reducing the intra-frame prediction computational effort

Work	Mode-level focused tool				Block-level skipped tool				Quadtree decision
	Intra_Wedge	RDM/MPM	TQ	DC-only	Bipartition modes	RMD/MPM	TQ	DC-only	
H. Chen et al. [7]									X
R. Conceição et al. [10]					X	X	X	X	
C.-H. Fu et al. [18]	X								
Z. Gu et al. [19]					X				
Z. Gu et al.[20]				X					
K.-K Peng et al.[44]		X	X	X	X				X
M. Saldanha et al. [52]*	X								
G. Sanchez et al. [66]*	X								
G. Sanchez et al. [67]*					X				
H.-B. Zhang et al. [81]									X
H.-B. Zhang et al.[82]	X	X							
H.-B. Zhang et al.[83]		X	X	X			X		

*our previous works

3.2 Related Work on Reducing the Computational Effort on the Inter-Frame/View Prediction

We found far fewer works on depth map inter-frame/view prediction than depth map intra-frame prediction because most of the encoding tools were inherited from texture, while the intra-frame prediction inserted new encoding modes to be explored. Three classes of works are found here: (i) mode-level, which focuses on fast ME/DE algorithms;(ii) block-level, which avoids computing all available modes for a given block, and; (iii) quadtree-level, which focuses on pruning the quadtree expansion.

Afonso et al. [2] designed an algorithm for accelerating the Disparity Estimation (DE). Their algorithm explores the 3D-HEVC horizontal camera arrangement by performing a fast horizontal-only DE search. This search is performed in three steps, where the first step searches for the best block match horizontally using a high subsampling. Then, the next steps perform a refinement in the previous step result by performing smaller subsampling evaluations around the best block match obtained in the previous step. It achieved a reduction between 32.7% and 61.8% in the SAD computation with an increase in BD-rate between 0.3123% and 0.4803%.

As previously mentioned in Section 3.1.2, Conceição et al. [10] proposed a block-level decision scheme that exploits the occurrence of Skip and Intra-picture skip for 3D-HEVC depth map encoding. The idea explored in this work is that these are the most used modes and when the RD-cost obtained encoding them are low, there is a high probability of them be selected as the final decision. Therefore, the RD-cost was used as a decision for removing the remaining modes evaluation. They provided a reduction between 24.4% and 33.7% in depth maps encoding time with an increase of BD-rate between 0.067% and 0.409%. Only this work was found proposing block-level decisions for random access modes.

Two works focus on inter-view/frame encoding at quadtree-level [32] and [38].

Mora et al. [38] propose to use inter-component for accelerating the depth map coding by limiting the quadtree expansion based on the CU partition information of the collocated texture. Considering the texture and depth maps represent the same scene from the same viewpoint at the same time, then the quadtrees of both are highly correlated. Thus, when encoding P- and B-frames, depth map quadtree expansion is limited to the correlated texture quadtree. This solution is integrated into the 3D-HTM reference software. When enabled, this solution is capable of achieving 52% of reduction in encoding time with an increase of 1.024% in BD-rate, when compared with disabling this technique.

Lei et al. [32] designed a fast mode decision algorithm for depth map coding based on the grayscale similarities and the inter-view correlation. The mode decision is composed of an early CU termination and an early PU mode decision for dependent views encoding.

In the CU termination decision, a threshold is responsible for classifying the grayscale similarities between the current encoding CU and the co-located block in the reference frame. When there is a high similarity, the quadtree depth level of the current CU is constrained by the level of the co-located quadtree in the reference frame. A grayscale similarity and an inter-view correlation are applied in the PU mode decision. The PU mode decision finalizes when the co-located CU in reference view selects merge or inter $2N \times 2N$ as best mode. This work reached an encoding time reduction of 41.5% with 2.06% increase in BD-rate.

Therefore, we can notice that there are a few works focused on inter-frame/view predictions in the literature, where only one was focused on block-level decisions and two focused on quadtree-level decisions. Moreover, we could not find any work in the literature proposing to accelerate ME/DE in depth maps coding.

3.3 Related Work on Encoding Effort Control

Although the works described in Sections 3.1 and 3.2 can reduce the 3D-HEVC encoding time meaningfully, there are applications where the Encoding Effort Control (EEC) is necessary. For popularization of the 3D video systems, it is desired that many device types could be capable of encoding/decoding 3D videos. Since different device types can have specific computational capacities and power restrictions, an EEC system needs to operate aware of the target device to ensure encoding efficiency. Moreover, considering many future battery-based devices will have an embedded 3D-HEVC encoder, then an EEC system will require dynamic adaptation of the encoding effort according to the available battery charge.

Some EEC solutions are found for the 2D version of HEVC. For instance, Corrêa et al. [11] propose an EEC for 2D-HEVC based on the coding tree structures adjustment according to a targeted effort. Besides, [14] and [71] proposed EEC techniques to adjust the HEVC encoding effort limiting the CTU maximum depth. In [14] a region-of-interest model is established to define different weights for regions according to their importance, whereas in [71] the certain saliency threshold is used to decide whether the CU is split. However, at this moment, there is no published work presenting methods to deal with EEC targeting any tool of the 3D-HEVC standard.

3.4 Related Work on Hardware Design

Since 3D-HEVC is a relatively new standard, only a few works have proposed a hardware design for its implementation. Moreover, considering that several encoding tools are new, previous hardware design for older standards cannot encode using the new encoding tools of 3D-HEVC.

Amish et al. [3] and our previous work [63] designed architectures capable of encoding both Intra_Wedge and Intra_Contour together. An Intra_Contour encoder architecture was designed in another of our previous work [70], and an Intra_Wedge encoder architecture was designed in our previous work [57]. Afonso et al. [1] designed an encoder architecture for the Intra-picture skip. However, no work in the literature was found proposing architectures for decoding the 3D-HEVC new encoding tools. Therefore, there is a space for designing efficient solutions aiming depth maps decoding in 3D-HEVC standard.

Besides, when designing a hardware architecture, the storage of all wedgelet patterns implies an Intra_Wedge hardware implementation with a high area and power dissipation. To overcome the storage requirement problem, some works proposed approaches to reduce the memory used for wedgelet pattern storage. Our previous work [63] removed the refinement step in the Intra_Wedge hardware design, reducing 30% of the memory requirement with a BD-rate loss of 0.25%. The work [34] proposed a technique that reduces 27.8% of the memory size with 0.03% BD-rate loss. This result was obtained storing the entire 16×16 wedgelet patterns and dynamically down-sampling these patterns for other block sizes. All these techniques related to reducing wedgelets memory requirements can be applied only to the encoder hardware since they cannot create all available wedgelets required for decoding a 3D-HEVC compliant bitstream.

However, the patterns inside Intra_Wedge wedgelet list contains several redundancies, and therefore, it has a high potential to be compressed at higher compression rate using more sophisticated compression algorithms.

4. CONTRIBUTIONS ON ENCODING TIME REDUCTION FOR DEPTH MAP INTRA-FRAME PREDICTION

This chapter presents the thesis contributions related to encoding time reduction for intra-frame prediction. Section 4.1 presents an in-depth analysis of encoding time and mode distribution for depth map intra-frame prediction. These analyses were contributions published in author works [60] and [69]. Section 4.2 presents algorithms designed for reducing the encoding time of depth map intra-frame prediction, and Section 4.3 shows results and discussion about the encoding time reduction.

4.1 Performance Analysis

This section is divided as follows. Subsection 4.1.1 discusses the time allocation of each encoding mode in some quantization scenarios. Section 4.1.2 presents the encoding mode distribution of all block sizes and some QPs. All videos of this section have been evaluated under all-intra-frame scenario using 3D-HTM 16.0 using the configurations described in Section 2.5, which already include by default the techniques proposed by the works [19] for skipping bipartition modes evaluations and [20] for reducing the DC-only evaluation encoding time. Finally, all experiments described in this section were conducted in a server containing two Xeon E5-2660 processors (10 cores and 20 threads) with 96 GB of main memory.

4.1.1 Encoding Time Distribution

Table 4.1 presents the results of an experiment considering the percentage of the encoding time distribution for each component per video and per QP-pair. Table 4.1 also presents the average time for encoding one texture frame and its associated depth map. The encoding time varies according to the video characteristics; however, in average the encoding time decreases with the QP increase.

The time spent during texture coding ranges from 11.0% to 18.2%, which is much lower than depth map encoding time. It occurs in all-intra-frame scenario because texture coding only applies traditional HEVC intra-frame prediction following TQ flow, while depth map coding still uses Intra_Wedge, Intra_Contour, Intra-picture skip, and DC-only evaluations. On average, depth maps coding is 5.8 times more time-consuming than the texture coding in the all-intra-frame scenario.

Table 4.1 – Encoding time evaluation for texture and depth coding in the 3D-HEVC all-intra-frame.

Videos	Encoding time											
	QP-pair = (25/34)			QP-pair = (30/39)			QP-pair = (35/42)			QP-pair = (40/45)		
	Texture (%)	Depth (%)	Total (s)	Texture (%)	Depth (%)	Total (s)	Texture (%)	Depth (%)	Total (s)	Texture (%)	Depth (%)	Total (s)
Balloons	15.7	84.3	33.72	14.4	85.6	34.78	14.9	85.1	33.64	14.5	85.5	34.48
Kendo	15.9	84.1	31.62	15.2	84.8	32.90	15.4	84.6	32.52	13.4	86.6	32.77
Newspaper	13.2	86.8	45.62	11.0	89.0	45.53	11.7	88.3	42.66	11.8	88.2	42.21
GT_Fly	14.8	85.2	83.11	13.3	86.7	82.58	14.5	85.5	70.58	14.8	85.2	67.04
Poznan_Hall2	18.2	81.8	47.67	16.2	83.8	49.60	16.8	83.2	47.72	15.6	84.4	49.20
Poznan_Street	13.6	86.4	95.89	12.2	87.8	93.01	13.7	86.3	77.32	13.5	86.5	73.96
Undo_Dancer	18.0	82.0	78.98	16.7	83.3	73.53	16.0	84.0	68.47	14.8	85.2	67.37
Shark	16.1	83.9	87.93	14.7	85.3	88.99	15.2	84.8	80.80	14.5	85.5	80.22
Average	15.7	84.3	63.07	14.2	85.8	62.62	14.8	85.2	56.71	14.1	85.9	52.43

Two other encoding time evaluations complement this experiment: (i) one describing the percentage of encoding time used per block size regarding the QP-depth, and (ii) another one describing the encoding time behavior inside a given block size. In both experiments, all video sequences and frames were evaluated and the encoder parameters were set according to our description in Section 2.5. Figure 4.1 illustrates the first evaluation. The encoding time axis represents the percentage of time required for depth map encoding for a given scenario.

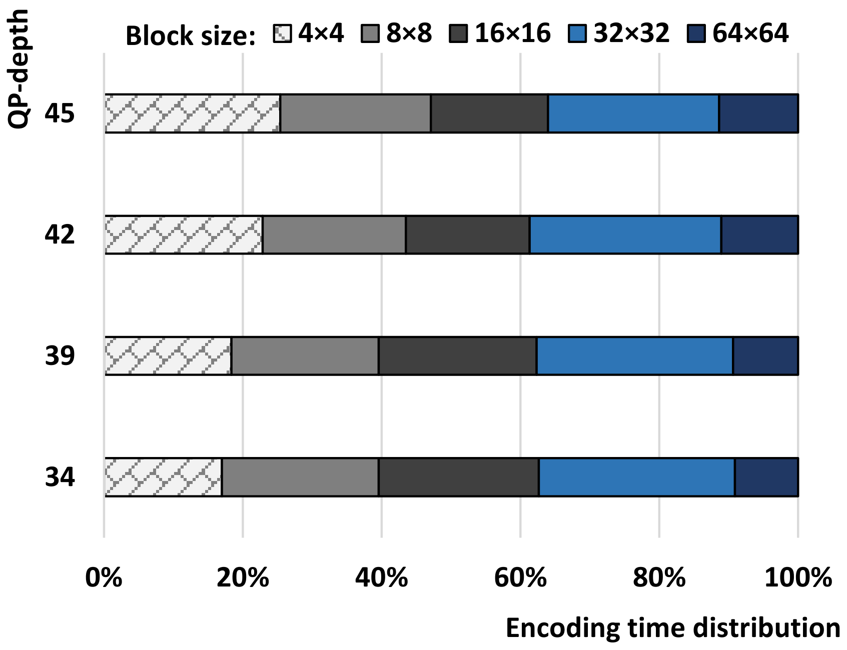


Figure 4.1 – Encoding time distribution according to the block size and QP-depth (source: [69]).

The QP-depth variation produces a slightly different encoding time distribution because QP-based timesaving algorithms ([19] and [20]) are being applied. However, it is expected the use of different block sizes varies according to the QP-depth, as discussed in Subsection 4.1.2 and it can bring several benefits when designing a timesaving solution. Besides, it is vital to identify the time spent on the encoding tools. For instance, some modes share encoding steps (e.g., the RMD and MPM selection in HEVC intra-frame prediction are used for both TQ and DC-only flows); therefore, the next analysis focuses on the encoding steps instead of the encoding tools. The following steps were considered for this encoding time distribution evaluation: (i) RMD and MPM selection; (ii) Intra_Wedge wedgelet search; (iii) Intra_Contour pattern generation; (iv) RD-list evaluation in TQ flow; (v) RD-list assessment in DC-only flow; and (vi) Intra-picture skip evaluation. Steps (iv) to (vi) also consider the time spent in entropy coding to generate the RD-cost. Figure 4.2 shows the encoding time distribution among the intra-frame prediction steps for each block size, regarding the QP-depth values of the corners, which complements the previous experiment.

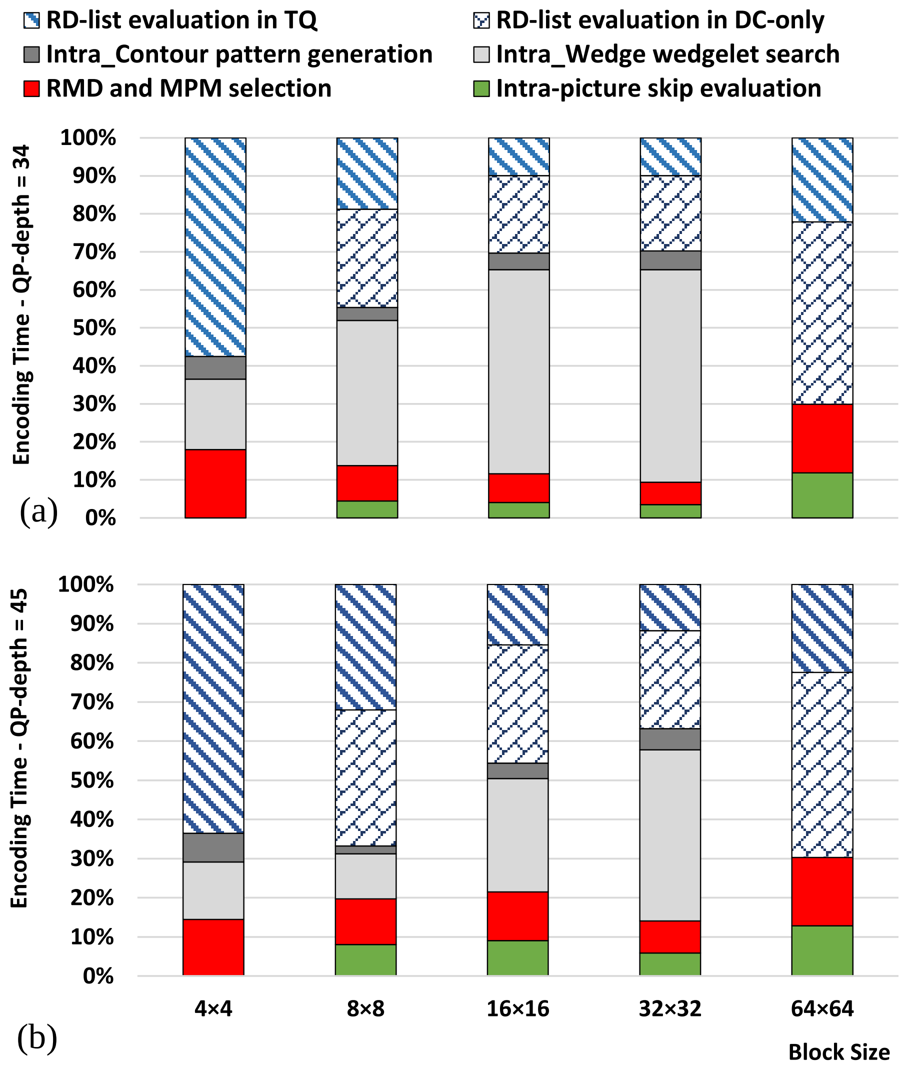


Figure 4.2 – Encoding time distribution according to the block size with (a) QP-depth=34 and (b) QP-depth=45 (source: [69]).

Firstly, one can notice the 4 × 4 blocks do not present Intra-picture skip and DC-only evaluations because these modes are applied only in block sizes ranging from 8 × 8 to

64×64. Besides, Intra_Wedge and Intra_Contour are not present in 64×64 blocks because bipartition modes are only applied for block sizes ranging from 4×4 to 32×32.

Regarding 4×4 blocks, where Intra-picture skip and DC-only evaluations are not available, the RD-list evaluation in TQ flow is the most time-demanding operation for both evaluated QP-depths. Intra_Wedge presents the second highest percentage of encoding time. The RMD and MPM selection and Intra_Contour pattern generation also use a significant amount of time to be processed, when compared to the other block sizes.

The 8×8 block size shows a significant time distribution variation of the encoding tools when QP-depth changes. For lower QP-depths, the Intra_Wedge wedgelet search represents the highest percentage of time among the encoding steps. This behavior changes entirely for high QP-depths (where the RD-list evaluation in DC-only and TQ have the highest percentage of time demanded) because the bipartition modes were proposed to handle with the high detail levels of depth maps, intending to preserve the borders, as previously discussed. Since higher QP-depths reduce the image details, then, the effort spent in bipartition modes is reduced by applying the heuristic proposed by [19].

The Intra_Wedge wedgelet search and the RD-list evaluation in DC-only present the first and second position, respectively, in the percentage of time spent to encode 16×16 and 32×32 block sizes for the evaluated QP-depths. The Intra_Wedge evaluation spends a higher percentage of time for higher block sizes. It occurs because the solution proposed in [19], which is present by default in 3D-HTM, skips the bipartition modes evaluation for low variance blocks. Since larger blocks tend to have higher variance, then fewer skips of bipartition modes occur.

When 64×64 blocks are processed, and bipartition modes are not available, the RD-list evaluation in DC-only is the most time-consuming operation, representing almost 50% of the total encoding time for both QP-depths. The DC-only execution time is most relevant for bigger block sizes because DC-only was planned to be used in homogeneous blocks or blocks that can be partitioned into two homogeneous areas. Thus, the heuristic proposed by [20] tends to skip the DC-only evaluation for heterogeneous regions. The execution times of Intra_Contour and Intra-picture skip prediction modes are very low for almost all block sizes. The Intra-picture skip encoding time is only higher than 10% for the 64×64 encoding blocks.

Meaningful conclusions are also obtained regarding the time distribution of encoding steps versus the QP-depth values. For example, the percentage of time spent in the Intra_Wedge calculation is inversely proportional to the QP-depth values used for all block sizes, because the heuristic proposed in [19] reduces the bipartition modes evaluations when high QP-depth values are used. Thus, lower QP-depths imply lower bipartition mode skips. The opposite behavior occurs for the RD-list evaluation in DC-only, which is higher for higher QP-depths for all block sizes. It happens because if the encoding block is not homogeneous, the DC-only evaluation is skipped [20] and higher QP-depth values tend to generate more homogeneous areas, which are encoded efficiently by DC-only. The RD-list evaluation in TQ also presents a similar behavior than the RD-list evaluation in DC-only where the increase in QP-depth values raises the percentage of time spent in the RD-list evaluation. Lastly, the Intra-picture skip evaluation, the RMD/MPM selection and the Intra_-Contour pattern generation do not variate the encoding time expressively with the QP-depth variation.

This subsection described the most time-consuming steps among depth map intra-frame predictions. However, it is also important to investigate when the encoding step is used for an encoding block. This analysis is presented next.

4.1.2 Block Sizes and Encoding Mode Distribution

Figure 4.3 shows the percentage of block size used to encode the depth maps according to the QP-depths. The experiment considers average results for all videos inside CTC when assessed under all-intra-frame configuration.

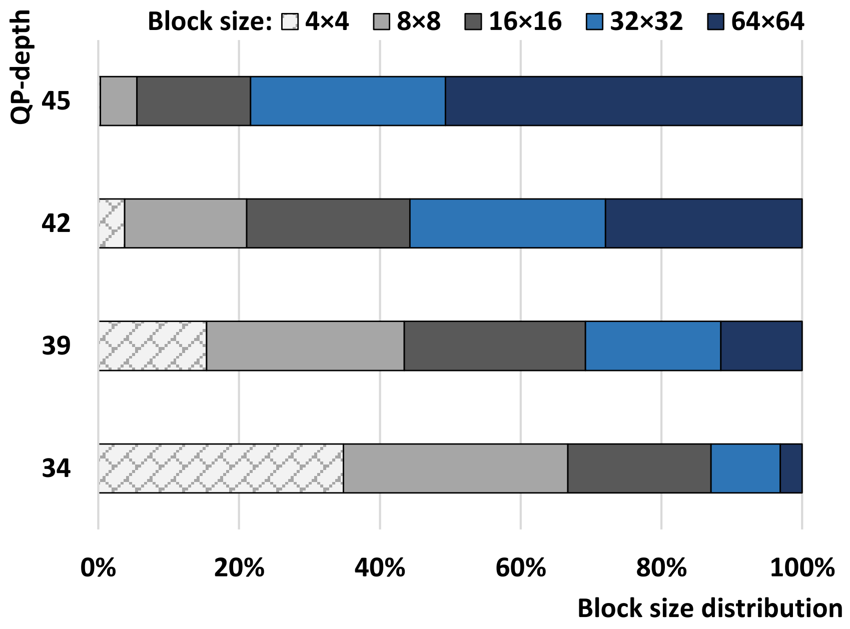


Figure 4.3 – Block size distribution for some quantization scenarios (source: [69]).

The selection of block size depends on the QP-depth value; high QP-depth values imply selecting bigger blocks (e.g., 32 $\times$ 32 and 64 $\times$ 64 block sizes are used more than 75% of times for QP-depth=45), and the opposite is also true (e.g., 4 $\times$ 4 and 8 $\times$ 8 block sizes are used more than 65% of times for QP-depth=34). Besides, the percentage of 16 $\times$ 16 blocks selection remains nearly constant, i.e., independently of the QP-depth. This happens because low QP-depths tend to preserve the depth map details, generating heterogeneous regions, which are encoded better with smaller block sizes. On the other hand, high QP-depths tend to attenuate the depth map details, creating homogeneous regions, which are encoded better with bigger block sizes. Besides, it is essential to understand the encoder decisions according to the available encoding flows. This analysis is presented next, considering five possibilities of encoding flows:

- **Bp-TQ** – a bipartition mode encodes the block using the TQ flow;
- **Intra-TQ** – the HEVC intra-frame prediction encodes the block using the TQ flow;
- **Bp-DC** – a bipartition mode encodes the block applying the DC-only flow;
- **Intra-DC** – the HEVC intra-frame prediction encodes the block applying the DC-only flow, and;
- **Intra-picture skip** – any Intra-picture skip mode encodes the block.

Given the two corners of QP-depths in CTC, Figure 4.4 shows the mode distribution selection for each block size has a low level of dependency with the QP-depth value.

Regarding 4×4 blocks, on average, Intra- and Bp-TQ flows were selected 68% and 32% of times, respectively. This is the unique block size where the Intra- or Bp-TQ flows are used in a relevant number of times since Intra-picture skip and DC-only are not available.

Concerning blocks ranging from 8×8 to 32×32, all the five flows are evaluated. In this range, independently of QP-depth, the Intra-picture skip encoding flow was selected more than 50% of times. The DC-only evaluation through Bp-DC and Intra-DC are also significant in this distribution since both encoded flows are selected 33% of times, on average. Intra-TQ and Bp-TQ are the less representative encoding flows. Intra-TQ is selected a maximum of 11% of times (QP-depth=34 and block size of 32×32) and a minimum of less than 1% (QP-depth=45 and block size of 8×8). Bp-TQ was selected less than 1% of times for all scenarios. For these intermediate block sizes, the higher the QP, the higher the use of Intra-picture skip and Bp-DC and the smaller the use of Intra-DC, Intra-TQ, and Bp-TQ. Besides, for all block sizes, Bp-TQ tends to zero for the highest value of QP-depth evaluated.

Only Intra-picture skip, Intra-DC, and Intra-TQ modes are evaluated in the 64×64 blocks. However, the experimental results show the Intra-TQ is almost not used. The Intra-picture skip mode is selected more than 85% of times, followed by Intra-DC. It happens mainly because larger blocks are chosen when encoding a region with low details, such as the background or the body of objects composed of large areas of homogeneous values in depth maps. In this case, Intra-TQ is almost irrelevant since Intra-picture skip and Intra-DC can reduce the bitrate expressively without affecting the visual quality.

Figure 4.5 (a) to (d) exemplify the selection of the five encoding flows when encoding the first frame central view of two CTC videos (Balloons and Shark). The boxes of the figures containing the colors (the same of Figure 4.4) of the used modes were plotted in front of the original depth image using 50% of transparency. The frames encoded using the

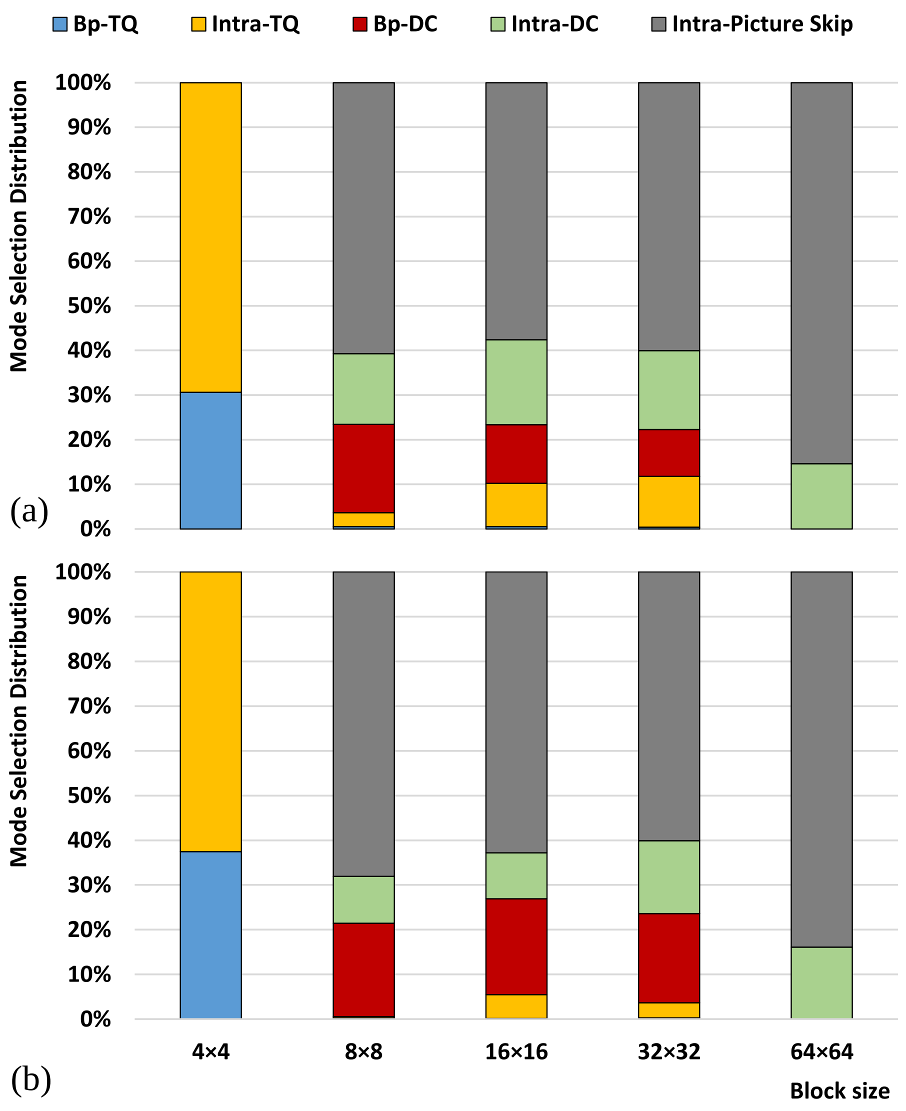


Figure 4.4 – Average mode distribution for each block size with (a) QP-depth=34 and (b) QP-depth=45 (source: [69]).

lowest, and the highest QP-depth values show that higher values of QP-depth significantly reduces the image details, generating smooth regions. This means the efficient encoding of depth maps with high QP-depths requires bigger block sizes, as presented in Figure 4.5. On the other hand, small block sizes are suitable to encode the depth map details using low QP-depths.

Additionally, Figure 4.5 shows Bp-TQ, Bp-DC, and Intra-TQ are used mainly in the borders of the objects, while Intra-picture skip and Intra-DC are selected in regions with smooth areas such as bodies and backgrounds. Intra-DC can also achieve a high usage in blocks that can be partitioned into two homogeneous regions. As the QP-depth increases, the quantity of smooth area also increases, raising the usage of Intra-picture skip and Intra-DC. It is crucial to identify the high use of Intra-picture skip encoding mode is concentrated in the background and the bodies of the objects. Intra-picture skip is hardly used in the borders of the objects since it does not encode residual information. However, Intra-picture skip is

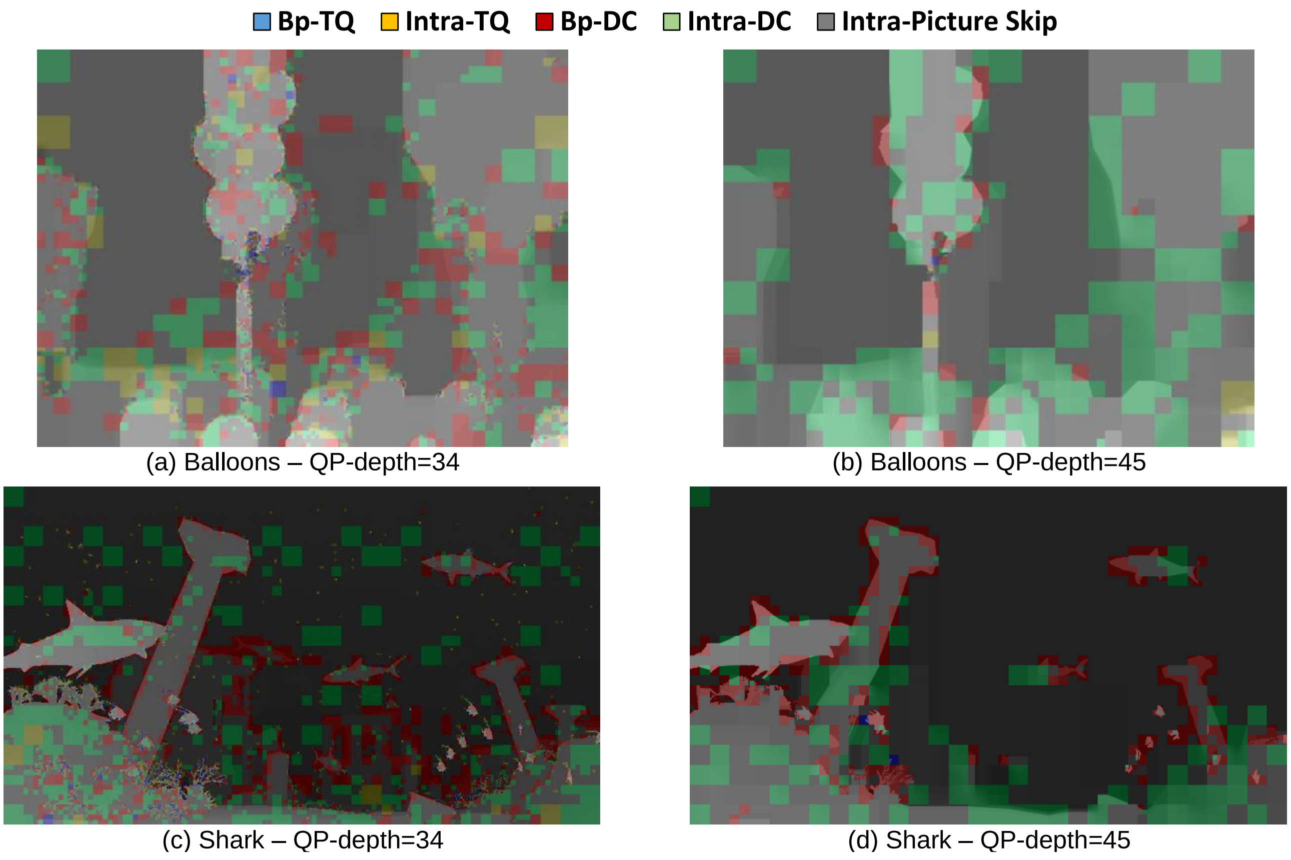


Figure 4.5 – Examples of mode distribution of the first frame central view of the videos Balloons (a) QP-depth=34, (b) QP-depth=45, and shark (c) QP-depth=34 and (d) QP-depth=45 (source: [69]).

efficient on the object borders, when a horizontal or vertical line segments the block. In these cases, Intra-picture skip Mode 0 or Mode 1 are selected, as can be seen in Balloons video sequence.

4.2 Designed Solutions

This Thesis proposed the design of four solutions aiming to reduce the encoding time of the intra-frame prediction: (i) DMM-1 Fast Pattern Selection (DFPS) [62]; (ii) Pattern-based Gradient Mode One Filter (P&GMOF) [55]; (iii) Enhanced Depth Rough Mode Decision (ED-RMD) [54]; and (iv) quadtree limitation using decision trees [48].

The design of the DFPS and P&GMOF algorithms arises to speed up the Intra_-Wedge tool since it is the most time-consuming operation. ED-RMD reduces the RD-list size; consequently, reducing both TQ and DC-only flows encoding effort. The quadtree limitation was designed using decision trees to skip unnecessary evaluations of small blocks inside the quadtree.

4.2.1 DMM-1 Fast Pattern Selector (DFPS)

Figure 4.6 shows the current encoding PU, and the upper and left PU neighbor. A vector called *PatternVector* was designed, and it was initially filled with zeros, where at the end of the Main Stage of Intra_Wedge, the selected pattern number ($Pattern_{curr}$) is inserted into the corresponding vector position.

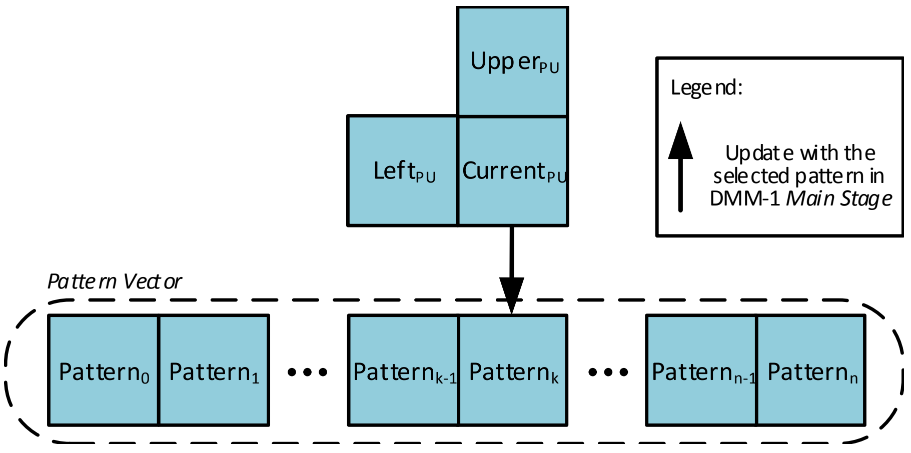


Figure 4.6 – An example of neighbor PUs and selected patterns in Intra_Wedge main stage (source: [55]).

Let $Pattern_{up}$, $Pattern_{left}$ and, $Pattern_{curr}$ be the patterns selected by the upper, left and current PU in the Main Stage of Intra_Wedge, respectively. Let P_{copy} be the predictor that searches for the pattern with the smallest distortion between $Pattern_{left}$ and $Pattern_{up}$ to the current PU. Let P_{extend} be the predictor that searches for the pattern with the smallest distortion among extended wedgelet orientations of $Pattern_{left}$ and $Pattern_{up}$. The following experiment associating each block size (i.e., 4×4, 8×8, 16×16 and 32×32) with a different *PatternVector* was proposed to explain the relevance of the predictors P_{copy} and P_{extend}. These predictors are defined in equations 4.1 and 4.2.

$$P_{copy} : Pattern_{curr} = PMinDist(Pattern_{up}, Pattern_{left}) \tag{4.1}$$

$$P_{extend} : Pattern_{curr} = PMinDist(Extend(Pattern_{up}, Pattern_{left})) \tag{4.2}$$

Where *PMinDist* searches for the Pattern with the smallest distortion, and *Extend* returns all patterns with the extended orientation.

The main idea of this experiment is to correlate the decisions of previous evaluated neighbor PUs with the current evaluation. When evaluating the K^{th} PU in the row, the pattern selected by the $Left_{PU}$ is stored in the position $Pattern_{k-1}$, and the pattern selected by the $Upper_{PU}$ is stored in the position $Pattern_k$ in the *PatternVector*. Therefore, by accessing the

PatternVector, the $Current_{PU}$ can retrieve the necessary information for the prediction based on P_{copy} and P_{extend}. At the end of the encoding, the $Current_{PU}$ writes this information in $Pattern_k$, since the pattern selected by the $Upper_{PU}$ is no longer necessary for the prediction of any other block. Then, when evaluating the next PU (K+1 in the row), the necessary information for its prediction are available in the *PatternVector* in the positions $Pattern_k$ and $Pattern_{k+1}$.

Every video sequence in the CTC was evaluated analyzing the success rate of each part of each predictor according to the block size. Equation 4.3 describes the success rate, which is the number of cases the proposed predictors had success (i.e., when the predictor obtained the same result of the original encoding flow) divided by the total number of evaluations. In this experiment, the P_{extend} predictor was only evaluated when P_{copy} fails.

$$Success\ rate = \frac{Number\ of\ cases\ the\ predictor\ had\ success}{Total\ number\ of\ evaluations}\ (\%) \qquad (4.3)$$

Figure 4.7 display these statistical results, according to the block size. The total success rate is the union of all predictors success rate (i.e., considerers that any of the predictors had success), which is a relevant information when combining these predictors in a single solution. Analyzing these results, one can note the smaller PUs obtain higher success in P_{copy}, while larger PUs get better results in P_{extend}. Moreover, the combination of both predictors allows obtaining high success rate independent of the encoding block size.

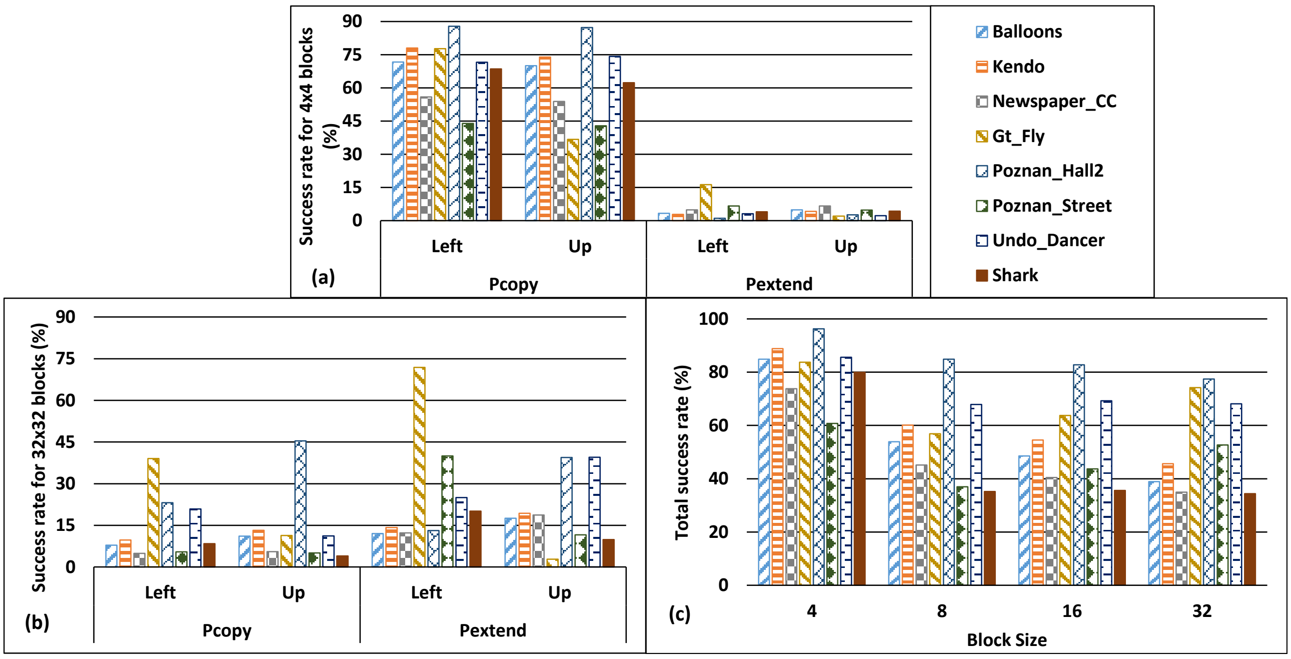


Figure 4.7 – Success rate for P_{copy} and P_{extend} for (a) 4×4 and (b) 32×32 blocks. (c) Average total success rate for all available block sizes (source: [55]).

Along with this analysis, Figure 4.8 presents the probability density function of having success in P_{copy} according to the obtained distortion divided by the block size. The distortion divided by the block size was used as a criterion since larger block sizes tend to have larger distortion values. This function, which follows a Gaussian distribution, was achieved by executing the GT_Fly video sequence under all-intra-frame mode. This video was randomly selected to perform this analysis.

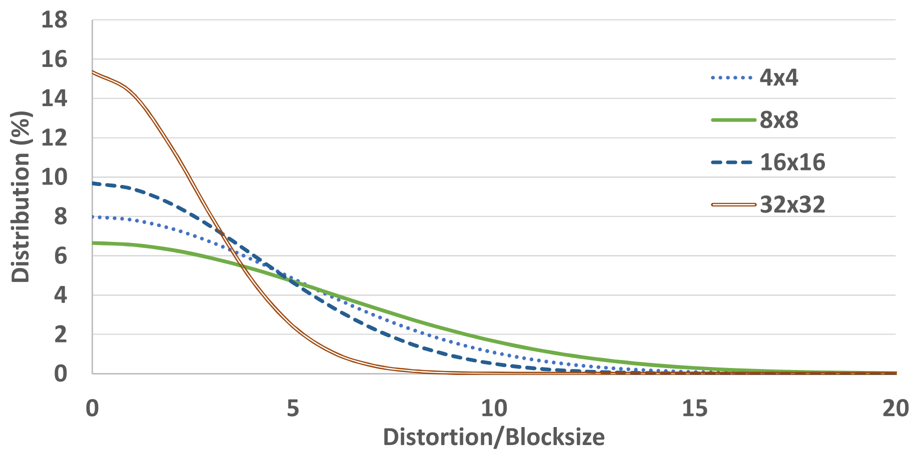


Figure 4.8 – Probability density function of P_{copy} have success according to the distortion divided by the block size in GT_FLY under all-intra-frame mode (source: [62]).

The results show a high probability of these predictors in having success for small values of distortion, having almost no chance to had success for values of distortion divided by the block size larger than 15 since larger distortion values mean a different pattern encodes the PU.

A similar analysis was done for P_{extend} resulting in similar conclusions. Thus, the values of distortion divided by the block size, obtained by the predictors, are directly related with the predictor success or fail. Therefore, they can be used to perform an early Intra_Wedge termination decision according to a threshold criterion. Moreover, these experiments enable to conclude there is a high probability of these predictors having success by performing at most two wedgelet evaluations in P_{copy}, or performing more than two wedgelet evaluations in P_{extend}, but lower than the traditional encoding flow, for all available block sizes.

Figure 4.9 shows the dataflow model of DFPS inside the Intra_Wedge encoding algorithm. Looking for a lightweight solution, capable of skipping many wedgelet evaluations, DFPS starts finding the minimum distortion in P_{copy} performing at most two pattern evaluations. If the distortion divided by the block size is lower than the THreshold 1 (TH1), defined by the offline analysis described later, then the Intra_Wedge main process is finalized, and the refinement is performed followed by the residue computation.

Further evaluations are required to obtain a reliable prediction on Intra_Wedge Main Stage when TH1 criterion is not met. Therefore, instead of evaluating the entire Intra_-Wedge initial set, a medium-weight solution is designed, where P_{extend} is assessed extending

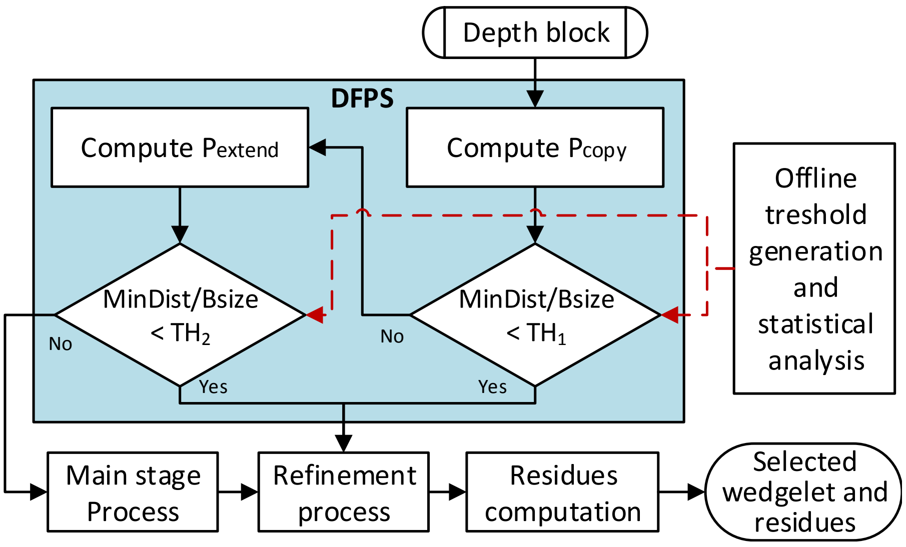


Figure 4.9 – DFPS dataflow model (source: [62]).

left and upper PUs wedgelet orientation. Once more, the minimum distortion divided by the block size is compared to THreshold 2 (TH2). If it is smaller than the threshold, then the Main Stage is finalized, and the Refinement Stage is performed. Otherwise, further evaluations are required, and the remaining wedgelets are evaluated without any simplification.

In the worst case (i.e., no simplification is performed) the RD-costs calculated by DFPS is the same than the traditional approach. Notice this solution always stores in a vector the information of the pattern selected in the Main Stage. Consequently, even if other modes has been selected in the complete RD-cost evaluation, this information can be used to accelerate the encoding of neighbor PUs.

The threshold values affect the encoding timesaving significantly. Thus, we made an extensive experimental analysis of 16 scenarios to explain the impact of threshold variation using the GT_Fly video sequence again. In each scenario, 10 frames of GT_Fly were encoded and compared to the original 3D-HTM results. The average and standard deviation of the distortion divided by the block size has been saved in the previous evaluation of GT_Fly video sequence, and they were used to determine the new evaluation scenarios.

The new evaluation scenarios suppose equation 4.4 for each threshold (represented by TH_n), where k is selected empirically ranging from 0 to 3, and u_n and std_n represent the average and standard deviation value for the TH_n computation, respectively. The threshold definition and k values were selected for ensuring the predictors have success in 50% (k = 0), 84% (k = 1), 97.5% (k = 2) and, 99.85% (k = 3) of cases. Table 4.2 presents all evaluated thresholds, with $u_1 = 1$, $u_2 = 2$, $std_1 = 5$, and $std_2 = 25$.

$$TH_n = u_n + k \times std_n \tag{4.4}$$

Table 4.2 – Thresholds for the new scenarios.

Case	TH1	TH2	Case	TH1	TH2
Case-1	1	2	**Case-9**	1	52
Case-2	6	2	**Case-10**	6	52
Case-3	11	2	**Case-11**	11	52
Case-4	16	2	**Case-12**	16	52
Case-5	1	27	**Case-13**	1	77
Case-6	6	27	**Case-14**	6	77
Case-7	11	27	**Case-15**	11	77
Case-8	16	27	**Case-16**	16	77

Figure 4.10 displays the percentage of wedgelet evaluations skipped according to the BD-rate criterion. DFPS allows some operation points capable of providing better coding efficiency or higher pattern skips. Case-2 was selected empirically (that is highlighted in Figure 4.10) as the best operation point to be further evaluated. However, other operation points could lead to higher encoding timesaving and higher impact in video quality.

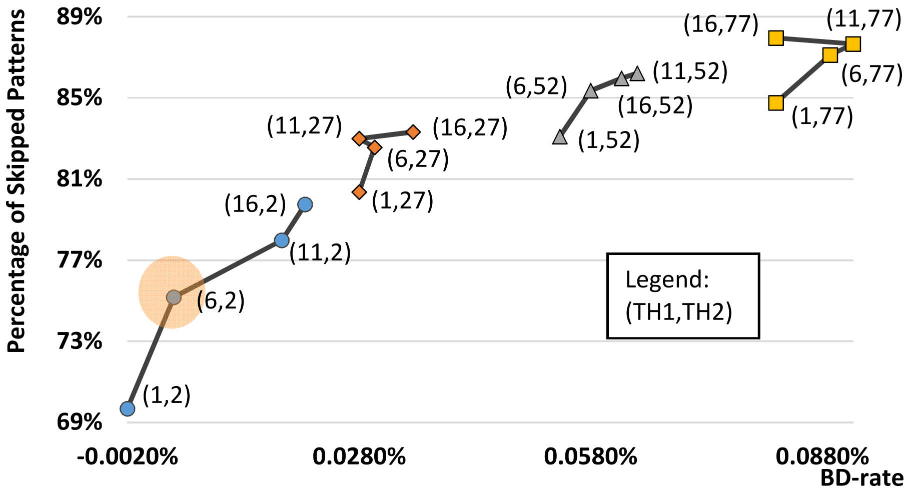


Figure 4.10 – Percentage of skips on Intra_Wedge patterns evaluation according to the BD-rate impact (source: [62]).

4.2.2 Pattern-Based Gradient Mode One Filter (P&GMOF)

Figure 4.11 exemplifies an 8×8 depth block with the selected wedgelet and the gradient values of the block borders when the Intra_Wedge original algorithm is applied.

The gradient vectors of the upper (*Grad_Upper*) and bottom (*Grad_Bottom*) rows, and left (*Grad_Left*) and right (*Grad_Right*) columns are obtained by applying the equations 4.5, 4.6, 4.7, and 4.8, respectively; where $P(x, y)$ represents the luminance sample pixel in the position (x, y) of the block. The x value in these equations ranges from 1 to $size - 1$,

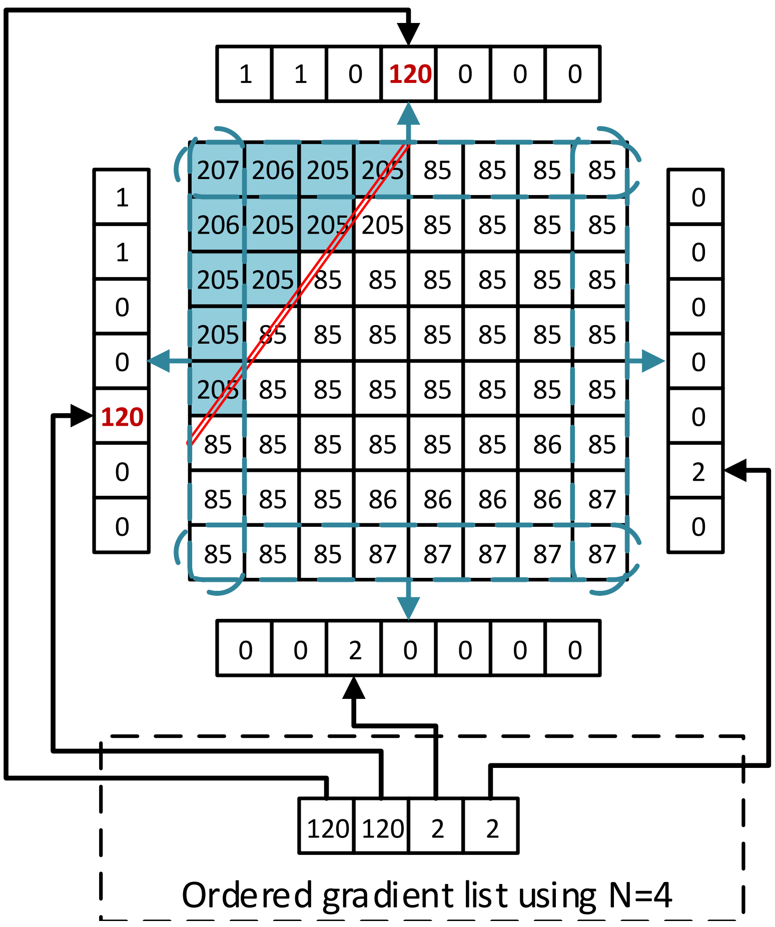


Figure 4.11 – Example of a wedgelet selection and an analysis of its border gradient (source: [55]).

where *size* represents the width of the block. Notice in the example in Figure 4.11 the best encoding wedgelet is found near the positions with the highest gradient. Consequently, one can conclude, the borders positions with high gradient values tend to be good candidates for Intra_Wedge wedgelet decision process.

$$Grad_{Upper}(x) = |P(1, x) - P(1, x + 1)| \tag{4.5}$$

$$Grad_{Left}(x) = |P(x, 1) - P(x + 1, 1)| \tag{4.6}$$

$$Grad_{Bottom}(x) = |P(size, x) - P(size, x + 1)| \tag{4.7}$$

$$Grad_{Right}(x) = |P(x, size) - P(x + 1, size)| \tag{4.8}$$

Since the Intra_Wedge algorithm is one of the most time-consuming operations in depth map intra-frame prediction, our work [66] has proposed the Gradient-Based Mode One

Filter (GMOF) algorithm to reduce the number of evaluated wedgelets considerably in Intra_-Wedge, and some variations of this algorithm are presented in [52]. All these algorithms reduce the number of wedgelets by searching for wedgelets whose straight line starts in a position with a significant change of the gradient value. The solutions proposed by [66] are line-based because it considers the position of the stored straight line, while the original execution of Intra_Wedge is performed using the stored binary pattern.

The original line-based Gradient-based Mode One Filter (GMOF) algorithm creates a gradient list with *N* positions ordered by the highest gradients of the borders. The positions in this list point to the position change obtained that gradient, as presented at the bottom of Figure 4.11. The algorithm evaluates only wedgelets whose straight line is located in those positions. By applying intense experimentation, the best *N* selected in [66] was 8.

The other GMOF variations are based on the original GMOF algorithm. The Strong GMOF (S-GMOF) algorithm selects the two highest gradients from two distinct borders. The Single Degree of Freedom GMOF (SDF-GMOF) selects the highest gradient among all possible positions and set a wedgelet line start in it. The ending of the wedgelet is selected searching in the wedgelet list for the nearest end position compared to the second gradient position. The Double Degree of Freedom GMOF (DDF-GMOF) searches the entire wedgelet list for the wedgelet that has the lowest distance compared to the two highest gradients positions. The S-GMOF, SDF-GMOF and DDF-GMOF algorithms can reduce the wedgelet list to a single wedgelet evaluation. After this initial assessment, the original Intra_-Wedge refinement is applied in all these algorithms. However, often when using these line-based algorithms (including the original GMOF), it is impossible to reach the best wedgelet approximation. Figure 4.12 exemplifies the encoding a 4×4 depth block with the best pattern selected by the original Intra_Wedge algorithm. This example shows the best wedgelet pattern is obtained starting the wedgelet in an intermediary pixel and there is a gradient value only in a half pixel of distance from there. It occurs because wedgelet evaluations are pattern-based and not line-based as previous work explored.

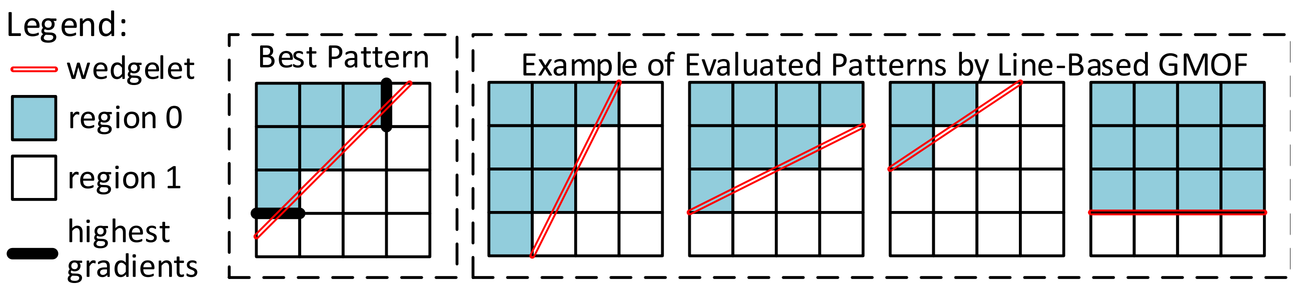


Figure 4.12 – Best pattern and the evaluated patterns using Pattern-Based GMOF of a 4×4 encoding depth block (source: [55]).

Considering the Intra_Wedge evaluation does not use the wedgelet line position, but uses only the binary pattern, a new P&GMOF algorithm is proposed. Figure 4.13 depicts the dataflow model of the P&GMOF.

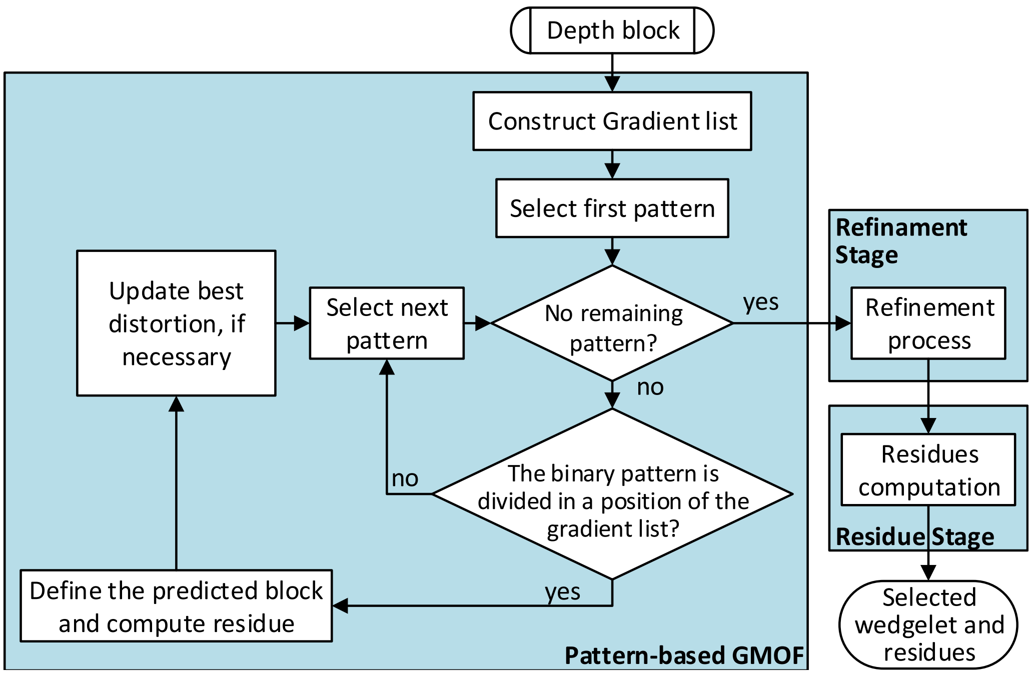


Figure 4.13 – P&GMOF dataflow model (source: [55]).

The gradient list constructed by the original line-based GMOF is maintained in this algorithm by applying the equations 4.5 to 4.8. After constructing this list, the pattern of each wedgelet in the initial wedgelet pattern set is analyzed with the positions in the gradient list. If there is a division in the pattern in any of those positions (i.e., it changes from one region to another in that position), the wedgelet is evaluated by its RD-cost. Otherwise, the next wedgelet is applied to the same process. Furthermore, when the entire initial wedgelet set is analyzed, the original refinement is applied, and the wedgelet that obtained the lowest RD-cost is selected as the best-encoded wedgelet, sending its residues and encoding information to the next encoder modules.

The same example presented in Figure 4.12 for line-based GMOF is also presented for P&GMOF in Figure 4.14. In this case, the pattern-based version can achieve the same wedgelet that would be selected by the original Intra_Wedge algorithm.

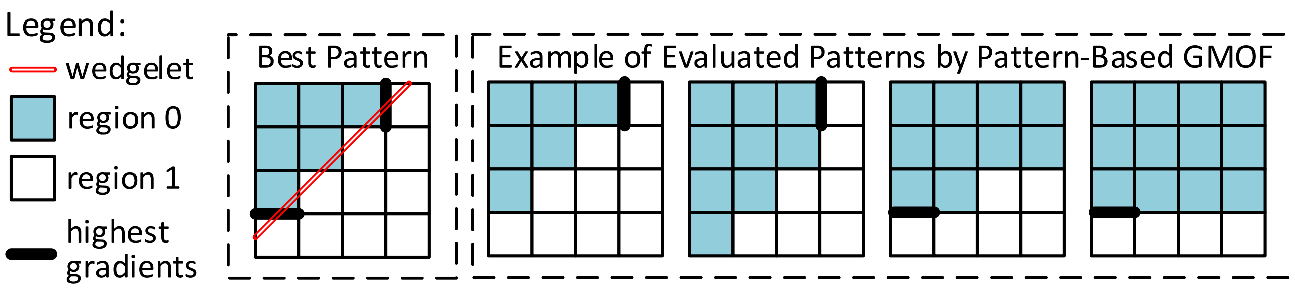
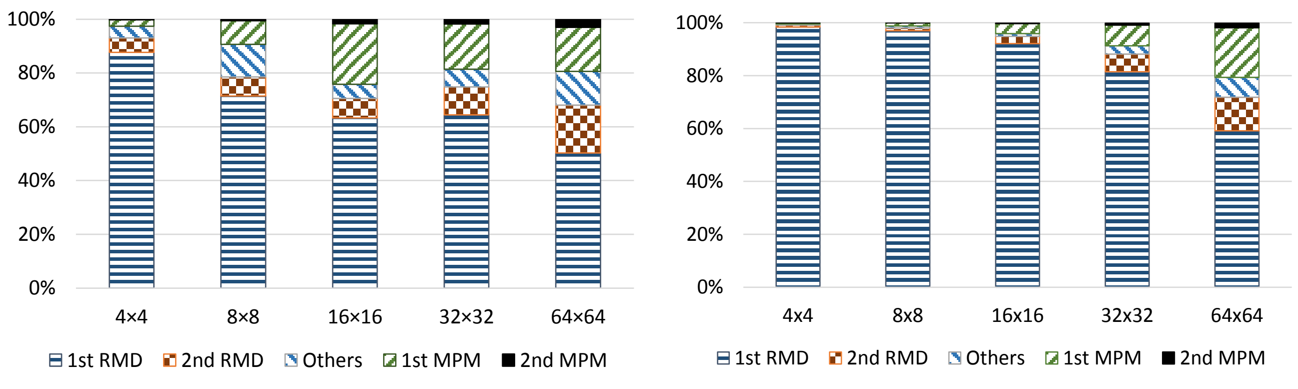


Figure 4.14 – Best pattern and the evaluated patterns using Pattern-Based GMOF of a 4×4 encoding depth block (source: [55]).

4.2.3 Enhanced Depth Rough Mode Decision (ED-RMD)

All videos inside CTC were encoded using the corners QP-depth in the all-intra-frame scenario. The best directional mode selected after computing its RD-cost during each block computation was stored and compared to its position in the RD-list for depth maps coding. The modes inside RD-list have the possibility of being inserted by the RMD or the MPM algorithms. When inserted by RMD, they are previously ordered by their SATD, as previous described in Section 2.2.1. Here, the position of these in the RD-list is called RMD rank.

Figure 4.15(a) and (b) present the statistical usages of each rank in the RMD selection and two MPM usage for QP-depth equal to 34 and 45, respectively. In 4×4 and 8×8 blocks, the legend Others groups the modes ranked in the positions 3 to 8. In 16×16 to 64×64 blocks, Others represents only the mode ranked in position 3, since these block sizes allows inserting only 3 modes and the MPMs inside RD-list.



(a) QP-depth = 34 (b) QP-depth = 45

Figure 4.15 – Statistical analysis of RMD and MPMs selection (source: [54]).

Notice the 1st RMD possesses the most representative selection signifying more than 50% in all encoding cases. Moreover, when increasing the encoding QP-depth, the selection of the 1st RMD mode is still higher. Furthermore, there are still positions in RD-list that contains a significant selection, i.e., the 2nd RMD mode and the 1st MPM.

The statistical analyses demonstrate the depth map coding does not require the complete RD evaluation for all modes inside RD-list. It happens because the RMD and MPM pre-analysis were designed considering texture coding that contains a more complex behavior. Besides, due to many new modes inserted into depth map intra-frame prediction, the removal of some HEVC intra-frame prediction modes in RD evaluation direct to significant encoding timesaving results without considerable impact on the encoding efficiency.

Since many encoding modes selected by the RMD and MPM algorithms have almost no impact in the encoding selection, we performed 48 experiments varying the number of modes selected by RMD and MPM to identify the encoding efficiency and timesaving of the proposed ED-RMD heuristic. These experiments encoded ten frames of Balloons and Undo_Dancer sequences under the all-intra configuration. In these experiments, we vary the number of modes evaluated in RD-list, between one and eight for 4×4 and 8×8 blocks, while vary between one and three the number of modes evaluated in 16×16 to 64×64 blocks. Besides, we analyses the use of one and two MPMs.

Table 4.3 and Table 4.4 describe the results of the experiments using one and two MPMs, respectively. The Encoding time column represents the percentage of encoding time compared with the original 3D-HTM execution, considering both texture and depth encoding time. Notice that removing one intra-frame mode from larger block sizes implies in higher computational effort reduction than removing one mode from smaller block sizes. Moreover, removing one evaluation of MPM implies in a small computational reduction and a negligible impact on BD-rate. Consequently, in real-time systems, reducing the MPM evaluation to a single mode results in a sound tradeoff.

Table 4.3 – ED-RMD evaluation using one MPM (source: [55]).

Modes selected for 4×4 and 8×8	**Modes selected for 16×16 to 64×64**					
	1		**2**		**3**	
	BD-rate	**Encoding time**	**BD-rate**	**Encoding time**	**BD-rate**	**Encoding time**
8	0.164%	90.2%	0.072%	95.5%	0.043%	99.3%
7	0.146%	87.8%	0.120%	92.7%	0.052%	97.0%
6	0.219%	85.1%	0.159%	90.4%	0.097%	94.5%
5	0.223%	82.6%	0.143%	87.7%	0.109%	91.9%
4	0.290%	80.2%	0.206%	85.3%	0.175%	89.5%
3	0.380%	77.6%	0.263%	82.7%	0.239%	87.0%
2	0.440%	74.9%	0.407%	80.1%	0.326%	84.2%
1	0.577%	71.4%	0.455%	76.4%	0.451%	80.5%

Table 4.4 – ED-RMD evaluation using two MPM (source: [55]).

Modes selected for 4×4 and 8×8	**Modes selected for 16×16 to 64×64**					
	1		**2**		**3**	
	BD-rate	**Encoding time**	**BD-rate**	**Encoding time**	**BD-rate**	**Encoding time**
8	0.108%	90.7%	0.050%	95.8%	0.000%	100.0%
7	0.103%	88.3%	0.053%	93.5%	-0.002%	97.7%
6	0.149%	85.9%	0.136%	91.0%	0.020%	95.3%
5	0.176%	83.3%	0.128%	88.4%	0.068%	92.6%
4	0.248%	80.8%	0.194%	86.1%	0.120%	90.2%
3	0.365%	78.4%	0.274%	83.5%	0.235%	87.7%
2	0.432%	75.5%	0.347%	80.7%	0.280%	85.0%
1	0.558%	71.7%	0.478%	77.1%	0.397%	81.3%

These experiments show different computational effort reduction could be achieved according to the system configuration. Notice there is no optimum configuration for the proposed algorithm, and the system can be configured according to its application constraints such as maximum BD-rate, encoding time, and maximum power dissipation. If a higher encoding efficiency is desired, then more modes can be inserted into the RD-list; however, when requirements of reducing the encoding time or power dissipation are critical, fewer modes can be inserted in the RD-list with small impact on the encoding efficiency.

The ED-RMD heuristic was defined using the four best-ranked modes in RMD and one in MPM for 4×4 and 8×8 blocks, and only the best-ranked mode in RMD and one in MPM for 16×16 to 64×64 blocks, considering the analysis explained before. Other operation points could also be explored, but this configuration presented the best tradeoff between computational effort reduction and coding efficiency degradation. The selected configuration saved almost 20% of the time, with a BD-rate increase under 0.3%, which is a considerable computational effort reduction and a negligible impact on the encoding efficiency.

4.2.4 Applying Machine Learning for Quadtree Limitation

Many encoder attributes were evaluated to define the ones most relevant to build the static CU trees for depth map encoding. A significant amount of data from the depth video sequences and internal encoding variables were collected to find features that could identify the splitting decisions. The attributes listed into Table 4.5 were evaluated and stored for each CU size during the 3D-HTM encoder execution.

These attributes verify depth map regions that tend to be harder to encode and, consequently, tend to cause a splitting decision. Figure 4.16 presents the density probability of the 64×64 CUs do not be split into smaller CUs for some collected attributes. Figure 4.16(a) and (b) show *MaxDiff* and *VAR*_64 have lower values for those CUs that are not split into smaller CUs. The distribution of RD-cost is shown in Figure 4.16(c) and reveals a correlation with the splitting decision. Figure 4.16(d) shows the distribution of *VAR*_16, which provides essential information for sub-blocks inside a 64×64 CU since high values of *VAR*_16 can indicate the presence of edges in the current CU and its information can be hidden in generated information for larger blocks. In the case of sub-blocks with low-variance values, the current encoding CU tends not to be split into smaller CUs.

The attribute evaluation allows concluding only *QP-depth*, *RD-cost*, *VAR*, *VAR_-size*, *Average*, and *MaxDiff* are relevant to build the static CU decision trees. Then, only these attributes were used in the data mining training process. Since 3D-HEVC depth map intra-prediction allows square CU sizes from 8×8 up to 64×64, we designed three static decision trees defining when CUs of sizes 16×16, 32×32 and 64×64 should be or not split into smaller CUs.

We encoded the Kendo video sequence in all-intra configuration, considering all CTC QP values, for the data mining process. The CTU size has been limited to 16×16,

Table 4.5 – Parameters evaluated in I-frames and their description.

Parameter	Description
QP-depth	The current QP-depth value, which defines the compression rate and has much impact on the CU split decision.
RD-cost	The RD-cost obtained when encoding the current CU size, which was used to evaluate better the relations among the different encoder decisions and the encoding efficiency.
VAR	The variance of the original samples inside the current CU, which indicates the block homogeneity, and then, if the block should or not be split.
VAR_size	The maximum variance of smaller blocks inside of the current CU, which represents the maximum variance of the samples inside a block. For a 64×64 CU, there are four instances of this attribute, one for each possible block partition (4×4, 8×8, 16×16 and 32×32). This information indicates the homogeneity or presence of edges into smaller blocks.
Average	The average value of the current CU, which is the average of all samples inside each CU. This information indicates if the encoding CU is near or far from the camera. Additional details of near objects should be maintained and, in this case, it is interesting to evaluate lower CUs sizes.
MaxDiff	The single maximum difference between samples of the current CU, which is useful in the CU split decisions since this information indicates sudden variations in samples values.
Corners_grad	The maximum absolute difference of the four corners in the current CU. This information indicates the presence of edges in the current CU.
Max_Grad	Maximum gradient is the maximum absolute difference of the four corners of smaller blocks inside of the current CU. As VAR_size, there are four instances of this attribute for 64×64 CU, and they indicate when a CU should be split or not.

32×32 and 64×64 pixels for each evaluation. We stored for each encoded CU: (i) all information presented earlier, and (ii) the information indicating if the CU has been split or not. The Kendo video sequence was selected from the CTC dataset randomly, and it was used to extract the data necessary to the offline training process. Although only one sequence was used in this training, similar results were obtained repeating the same process with more video sequences or using a different video sequence.

We employed the Waikato Environment for Knowledge Analysis (WEKA) [21], version 3.8, to train each decision tree with the J48 algorithm, which is an open-source implementation of the C4.5 algorithm [45] available on WEKA. Seeking for a better data balancing, the input files were organized in two data sets with equal sizes having inputs that result in splitting and not the CUs. Besides, to avoid the overfitting problem on the train data set, the Reduced Error Pruning (REP) [5] was performed in each tree, reducing the depth of decision trees and allowing a better generalization.

Figure 4.17 illustrates the static decision tree generated for 64×64 CUs, where the leaves “N” and “S” correspond to the not split and split decisions, respectively. The decision trees for 32×32 and 16×16 CUs are composed of five and eight decision levels, respectively. The attributes in each decision tree were selected through the information gain, which is used by the training algorithm of the WEKA decision trees [21].

Table 4.6 presents the complete list of attributes and the corresponding usage in the three proposed decision trees. These attributes were selected in the decision trees

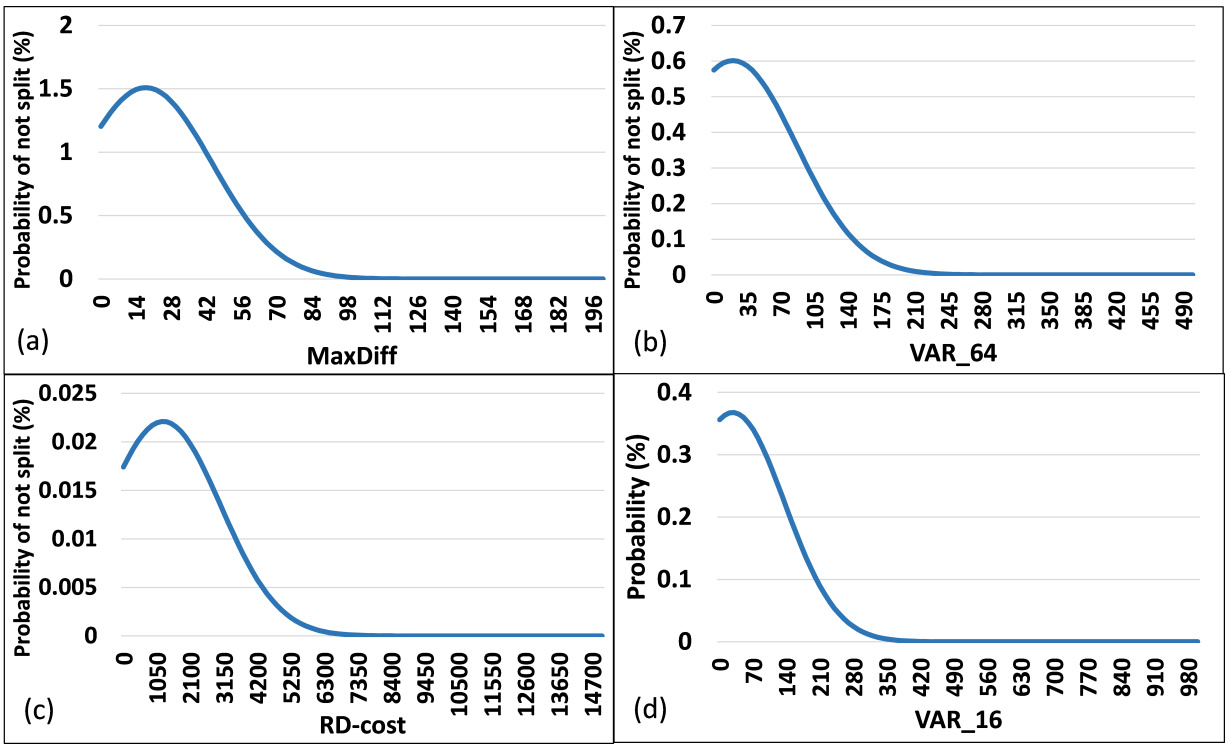


Figure 4.16 – Probability density of the analyzed attributes does not divide the current 64×64 CUs (source: [48]).

construction that uses the Information Gain (IG) during the tree definition. The IG of each attribute refers to the difference between the order of entropy for the entire data set and the entropy of the partitioned subset for the attribute being evaluated. The IG of each used attributed is also presented in Table 4.6.

Table 4.6 – Attributes used in decision trees for I-frames.

Attributes	**64 × 64**		**32 × 32**		**16 × 16**	
	Used	**IG**	**Used**	**IG**	**Used**	**IG**
QP-depth	×	0.089	×	0.123	×	0.039
RD-cost	×	0.312	×	0.203	×	0.293
Var	×	0.395	×	0.17	×	0.027
Var_16	×	0.397		-		-
Var_8		-	×	0.171	×	0.025
Var_4		-		-	×	0.025
MaxDiff		-		-	×	0.026
Average		-	×	0.11	×	0.061
Accuracy	**92.68%**		**83.59%**		**84.54%**	

The proposed solution does not add new computational effort to the 3D-HEVC encoder since the training is an offline operation, which is performed only once to define the static trees, and the attributes are easily obtained. Table 4.6 presents the accuracies of the constructed decision trees, which are 84.54%, 83.59% and 92.68% for 16 × 16, 32 × 32, and 64 × 64 CUs, respectively.

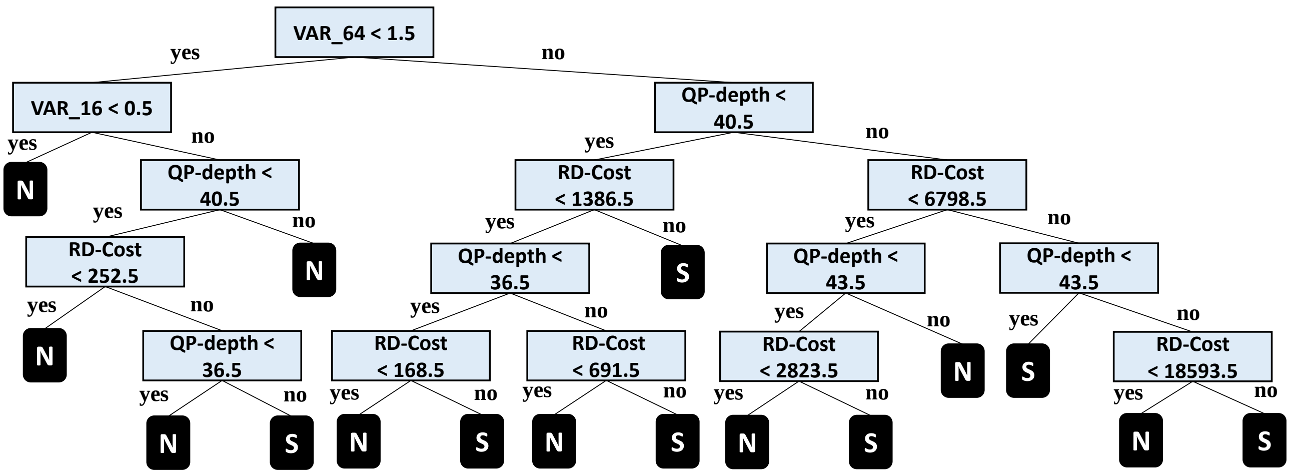


Figure 4.17 – Decision tree for splitting decision in 64 $\times$ 64 CUs (source: [48]).

4.3 Experimental Results

Table 4.7 illustrates the results of all-intra-frame evaluation for DFPS Case-2 and P&GMOF, and Table 4.8 presents the results for ED-RMD and quadtree limitation using machine learning algorithms. Both results considers the CTC described in Section 2.5. One can notice there is some significant performance difference in different videos execution, which is explained by the fact that each video posses different characteristics as highlighted in Section 2.5.

In average, the DFPS solution reduces 11.7% the encoder time considering texture and depth coding execution time when the all-intra-frame mode is considered, with a drawback of 0.0467% in BD-rate. The P&GMOF was evaluated using $N = 8$ since this value was already studied in a previous work [66]. This algorithm saves 6.5% of the encoding time considering texture and depth coding execution time with a BD-rate increase of 0.0119%. Notice the DFPS and P&GMOF algorithms, which are focused on Intra_Wedge, can be applied together. In this case, instead of evaluating the remaining patterns when the P_{extend} predictor fails, it is possible to evaluate only the positions indicated by the Pattern-based GMOF. Figure 4.18 shows the percentage of wedgelets skipped for P&GMOF and DFPS, according to the evaluated video sequence. On average, P&GMOF reduces the wedgelet evaluation by 58%, while DFPS reduces the wedgelet evaluation by 71%. In the DFPS, the highest gains are obtained in higher resolution videos because the first predictor succeeds more often in this kind of videos. Although P&GMOF and DFPS obtain a similar average reduction in wedgelet evaluations for lower resolution videos, the DFPS superior timesaving reduction is explained because it minimizes the effort spent in smaller block sizes more than the traditional GMOF.

ED-RMD was configured with the highest encoding timesaving obtained in the analysis with a BD-rate under 0.3% (in Subsection 4.2.3). The selected condition uses the four

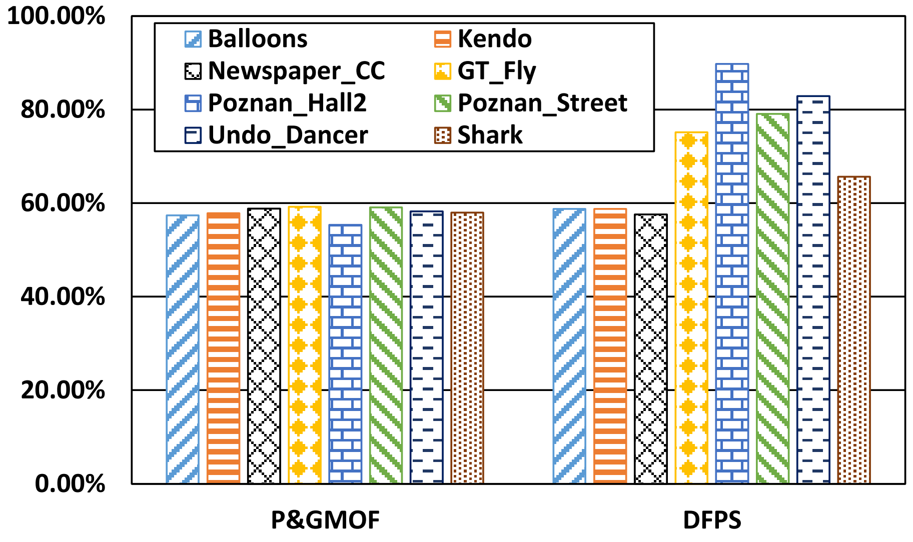


Figure 4.18 – P&GMOF and DFPS solutions for reducing the wedgelet evaluation (source: [55]).

Table 4.7 – DFPS and Pattern-based GMOF evaluation in the all-intra-frame scenario.

Resolution	Videos	DFPS Case-2		P&GMOF	
		Synthesis only BD-rate	Timesaving	Synthesis only BD-rate	Timesaving
1024 × 768	Balloons	0.0152%	8.1%	0.0158%	6.8%
	Kendo	0.0189%	6.5%	0.0141%	5.1%
	Newspaper_cc	0.0733%	10.8%	0.0344%	6.6%
	Average	0.0358%	8.5%	0.0214%	6.2%
1920 × 1088	GT_Fly	0.0045%	15.3%	0.0049%	8.3%
	Poznan_Hall2	0.1847%	13.3%	-0.0018%	6.1%
	Poznan_Street	0.0380%	14.6%	0.0051%	7.2%
	Undo_Dancer	0.0298%	13.7%	0.0107%	5.7%
	Shark	0.0152%	11.0%	0.0049%	8.3%
	Average	0.0533%	13.6%	0.0047%	6.8%
Average		0.0467%	11.7%	0.0119%	6.5%

best-ranked modes in the RMD selection and one in the MPM for 4 × 4 and 8 × 8 blocks. For 16 × 16 to 64 × 64 blocks, the proposed algorithm uses only the best-ranked mode in the RMD selection and the first MPM since this configuration reduces almost 20% the encoding time in the experiments described in Subsection 4.2.3. The previous evaluation of ED-RMD was applied only to ten frames of two videos. Then, this case study evaluated all videos and frames in CTC all-intra-frame test case. On average, it reduced 20.6% of the encoder execution time considering texture and depth execution time with a drawback of 0.1671% in the BD-rate of synthesized views.

The quadtree limitation with machine learning reduces 52.4% of the encoding time considering texture and depth execution time, on average with 0.1770% of BD-rate increase. When considering only depth maps execution time this solution is capable of achieving 59% of execution time reduction. Figure 4.19 illustrates the percentage of CUs that were not

Table 4.8 – ED-RMD and quadtree limitation using machine learning in the all-intra-frame scenario.

Resolution	Videos	ED-RMD		Quadtree limitation using machine learning	
		Synthesis only BD-rate	Timesaving	Synthesis only BD-rate	Timesaving
1024 × 768	Balloons	0.2662%	19.9%	0.1425%	45.7%
	Kendo	0.2844%	19.4%	0.1909%	49.8%
	Newspaper_cc	0.3064%	20.3%	0.1062%	37.3%
	Average	0.2857%	19.9%	0.1465%	44.3%
1920 × 1088	GT_Fly	0.0965%	21.0%	0.0561%	58.8%
	Poznan_Hall2	0.2046%	22.0%	0.6242%	63.4%
	Poznan_Street	0.1484%	22.7%	0.1204%	55.8%
	Undo_Dancer	0.0974%	19.0%	0.1406%	56.5%
	Shark	0.0829%	20.3%	0.0352%	51.9%
	Average	0.1367%	21.0%	0.2353%	57.3%
Average		0.1671%	20.6%	0.1770%	52.4%

split according to the CU size and QP-depth value. Higher QPs achieve higher not splitting percentages. For instance, regarding 64 × 64 CUs and QP=45, the percentage of CUs that were not split is 85%, on average; it happens because this QP causes higher compression rates and tends to encode the CTUs with larger CU sizes. This solution obtained the highest reduction in the encoding time among the designed ones because it works in a high-level, pruning some levels of quadtree evaluations. However, all proposed solutions can be applied in different scenarios. For example, DFPS and P&GMOF can be applied in the design of an Intra_Wedge architecture focusing on reducing the hardware usage, required frequency, and power dissipation.

Table 4.9 compares the proposed algorithms with related works mentioned in Section 3. Some solutions present results for Random access evaluation only, and some other only count depth map coding (without considering the whole encoder time).

The work [10] does not provide results for the intra-frame prediction algorithm individually; thus, it will be compared in Chapter 6 that explores inter-frame prediction timesaving algorithms.

The proposed solutions present a small impact in the encoding performance, where solutions such as DFPS and P&GMOF that require less than 0.05% of BD-rate increase can be considered negligible in this aspect while providing a significant encoding timesaving. Comparing these solutions to other works that focus only on Intra_Wedge, such as [18], [52], and [66], one can notice the timesaving results provided by our solutions achieve a worthy tradeoff reducing Intra-Wedge encoding effort.

ED-RMD and the quadtree limitation also provides competitive results regarding BD-rate, where an increase of less than 0.18% is required with a significant encoding time-

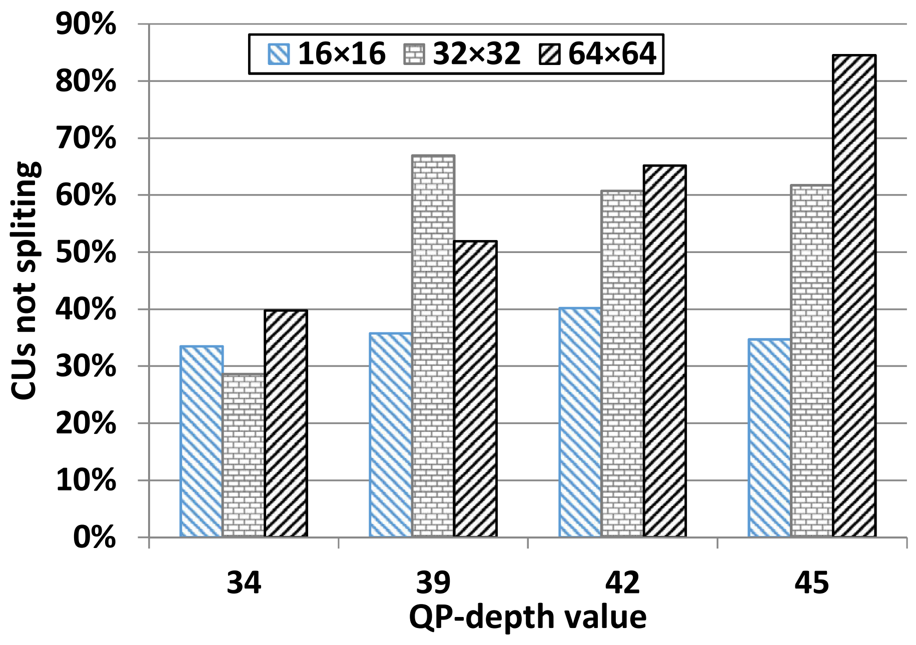


Figure 4.19 – Percentage of non-splitting CUs according to the block size and QP-depth for AI configuration (source: [49]).

Table 4.9 – Comparisons of intra-frame prediction timesaving algorithms.

Work	3D-HTM version	Decision level	Synthesis only BD-rate	Encoding timesaving
DFPS	16.0	Mode	0.047%	11.7%
P&GMOF		Mode	0.012%	6.5%
ED-RMD		Mode	0.167%	20.6%
Quadtree limitation using machine learning		Quadtree	0.177%	52.4%
H. Chen et al. [7]	16.1	Quadtree	0.09%	45.1%
R. Conceição et al. [10]*	16.0 – RA	Block	0.062% - 0.409%	9.6% - 13.8%
C.-H. Fu et al. [18]	8.1	Mode	0.49%	55% (Intra_Wedge only)
Z. Gu et al. [19]	-	Block	Default	Default
Z. Gu et al. [20]	-	Mode	Default	Default
K.-K. Peng et al. [44]	13.0	Mode, Block and Quadtree	0.8%	37.6%
M. Saldanha et al. [52]	10.2 – RA	Mode	0.33% - 1.47%	4.9% - 8.2% (depth only)
G. Sanchez et al. [66]	7.0 – RA	Mode	-0.047%	9.8% (depth only)
G. Sanchez et al. [67]	7.0	Block	-0.064%	5.9%
H.-B. Zhang et al. [81]	13.0	Quadtree	0.44%	41%
H.-B. Zhang et al. [82]	8.1	Mode	1.03%	27.9% (depth only)
H.-B. Zhang et al. [83]	13.0	Mode and block	0.54%	32.9% (depth only)

Note: *this result considered the intra- and inter-frame technique explored in that work.

saving, mainly in the quadtree limitation solution. Typically, solutions that provide higher encoding time-savings, such as ED-RMD and the quadtree limitation, require a significant BD-rate increase, such as [44], [81], [82], and [83].

Analyzing the quadtree limitation individually with other solutions that proposed to limit the quadtree, such as [44] and [81], our solution requires a smaller BD-rate increase and provides a superior timesaving result. Compared to [7], our quadtree solution obtained a higher BD-rate increase; however, we were capable of reducing significantly more the encoding timesaving, since their results considered only depth map encoding time.

5. CONTRIBUTIONS ON BIPARTITION MODE ALGORITHMS

Considering the bipartition modes are new encoding tools for exploring depth map encoding characteristics, this chapter presents some specific contributions to these tools including (i) Encoding effort control for both bipartition modes as presented in Section 5.1 [65]; (ii) Parallelism exploration for Intra_Wedge presented in Section 5.2 [58][59]; (iii) A study of compression in the Wedgelets patterns presented in Section 5.3 [53][64]; and (iv) A decoder hardware design for both bipartition modes presented in Section 5.4 [56][61].

5.1 Encoding Effort Control for Bipartition Modes

We selected the Simplified Edge Detector (SED) algorithm proposed in [67] as the base for designing the encoding effort control because SED, whose flowchart is presented in Figure 5.1, contains some characteristics that can be explored in this scenario such as the selection of skipping the entire evaluation of bipartition modes and decision based on a threshold. However, different block-level decision algorithms can be used following a similar approach.

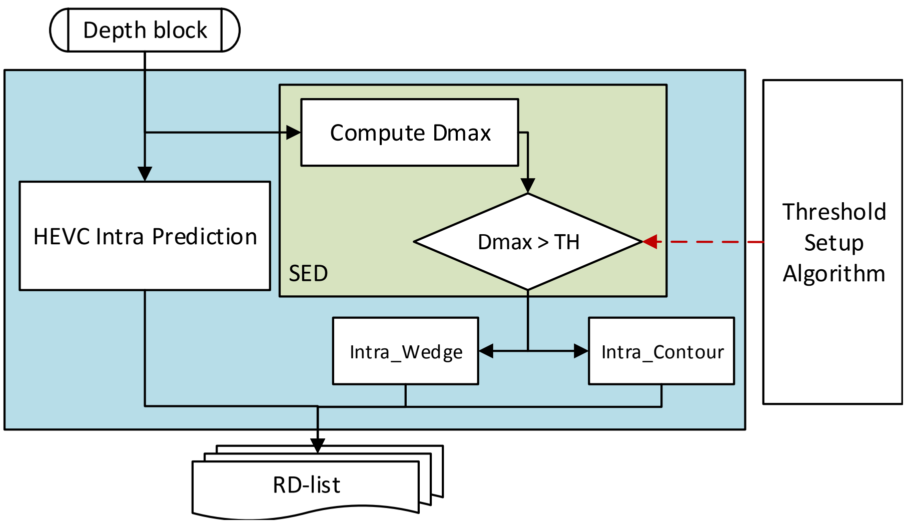


Figure 5.1 – Flowchart of the SED algorithm (source: [65]).

In our previous work [67], we demonstrated the highest difference of the four corner samples of a block, called D_{max}, contains significant information to classify this block as homogeneous or edge. Figure 5.2 depicts a slice from Undo_Dancer depth map with four edge blocks and four homogeneous blocks detached. The D_{max} values of those detached blocks are provided below. One can notice the edge blocks contain D_{max} with high values, while homogeneous regions contain low D_{max} values. Motivated by this analysis, the SED algorithm compares D_{max} with a fixed threshold to perform this classification. When D_{max} is higher than the threshold, the encoding block is classified as an edge block, and bipartition modes evaluation is necessary. Otherwise, the encoding block is classified as a homogeneous region and the bipartition modes evaluations can be skipped without affecting the encoding efficiency significantly.

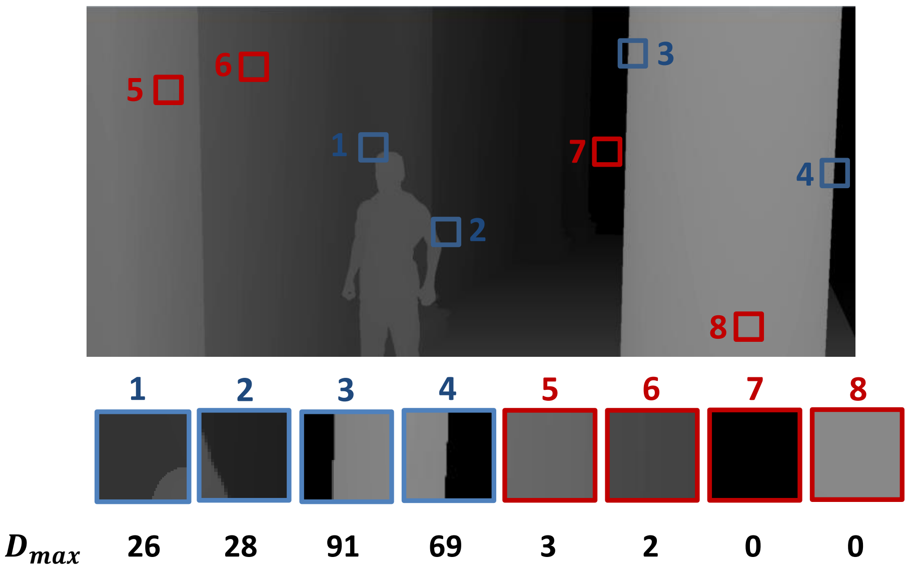


Figure 5.2 – Depth map example with detached edges (1-4) and homogeneous regions (5-8) (source: [65]).

We proposed in [67] the SED classification based on a fixed threshold value according to the video resolution and the block size. The thresholds were defined in an offline statistical analysis regarding the resolutions 1024×768 and 1920×1088. Therefore, SED thresholds are not defined in lower video resolutions such as CIF (352×288) and SD (720×480), or higher video resolutions such as 4K (3840×2160) and UHD (7680×4320). Moreover, the efficiency of the SED decision depends on the encoding video characteristic, which introduces non-determinism regarding the encoding effort spent in the bipartition modes. Figure 5.3 illustrates this idea presenting the D_{max} heat map for two frames of Shark and Poznan_Street video sequences considering 8×8 blocks, where red colored blocks indicate the highest values of D_{max}, green regions means intermediary values, and blue regions show the lowest values.

When SED uses a fixed threshold proposed in [67], a variable number of bipartition mode evaluation are skipped for different types of videos since each video has a different quantity of borders and details as exemplified in Figure 5.3.

Figure 5.4 exemplifies the 100 first encoding frames of the CTC sequences using the fixed thresholds SED [67]. Notice that each video has its specific profile as previously highlighted in the example shown in Figure 5.3. Moreover, during the execution of a small part of a video, there are high differences in the percentage of bipartition modes evaluation, ranging from 3% to 20%. It happens because each video has different characteristics as described in Section 2.5. Besides, the percentage of bipartition modes evaluated varies significantelly according to the block size, as demonstrate in Figure 5.4(a) (4×4 blocks) and Figure 5.4(b) (32×32 blocks), when the SED threshold is fixed. For example, Shark video sequence achieves about 20% of bipartition modes evaluated considering 32×32 blocks, whereas, for 4×4 blocks, the highest bipartition modes evaluation achieved is about 6%. Since each video has a specific profile and there are a variety of devices providing different computational capacities and power restrictions for encoding 3D videos, it is crucial to build solutions for controlling the encoding effort allowing performing bipartition modes operation for any encoding video characteristics, without surpassing the system resources limit. In the case of the system executes fewer evaluations than its processing capacity, then it is possible to compensate it by applying further evaluations in the next frames, increasing the encoding efficiency.

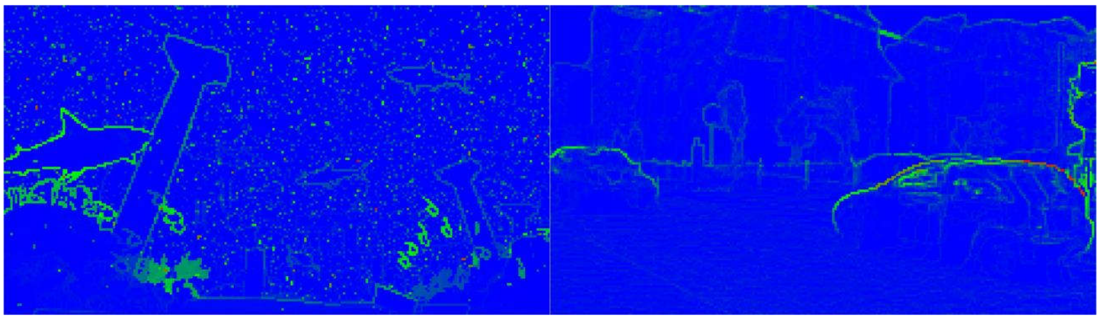


Figure 5.3 – The SED heat map for the first frame of the central view of (a) Shark and (b) Poznan_street video sequences (source: [65]).

5.1.1 The Design of the Complexity Control System

A Proportional-Integral-Derivative (PID) controller was proposed to adjust the SED threshold for controlling the bipartition modes evaluation according to the desired target encoding effort dynamically. Figure 5.5 displays the diagram of the proposed control system, which is replicated for each block size. The delay box in Figure 5.5 outputs the timing value from the previously encoded frame.

Figure 5.6 presents the pseudo-code of the proposed control system to complement the control block diagram illustrated in Figure 5.5.

The algorithm selects the evaluation rate of a target bipartition mode ($Target_{rate}$) as the input of the system. For example, when a target of 5% is selected, then 95% of the bipartition mode evaluation are skipped. This $Target_{rate}$ is selected according to the available resources of the system, and it can be changed dynamically from frame to frame, according to the available resources (e.g., it can be reduced if the system is running with low battery). Let f_n be the number of the current frame being encoded, then the error of f_n ($e[f_n]$) can be

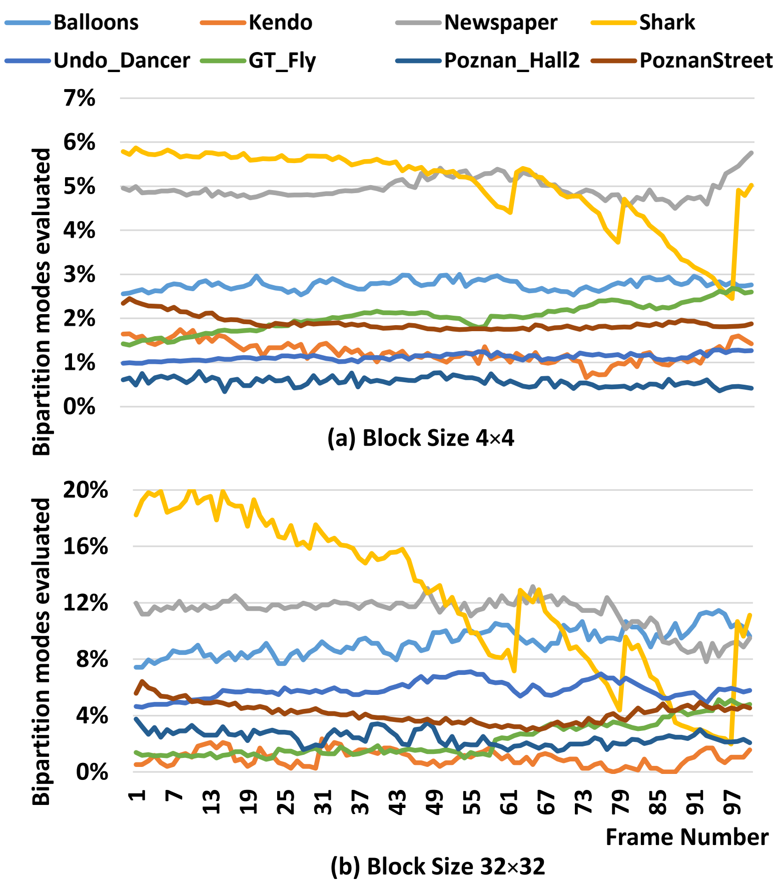


Figure 5.4 – SED bipartition mode evaluation (source: [65]).

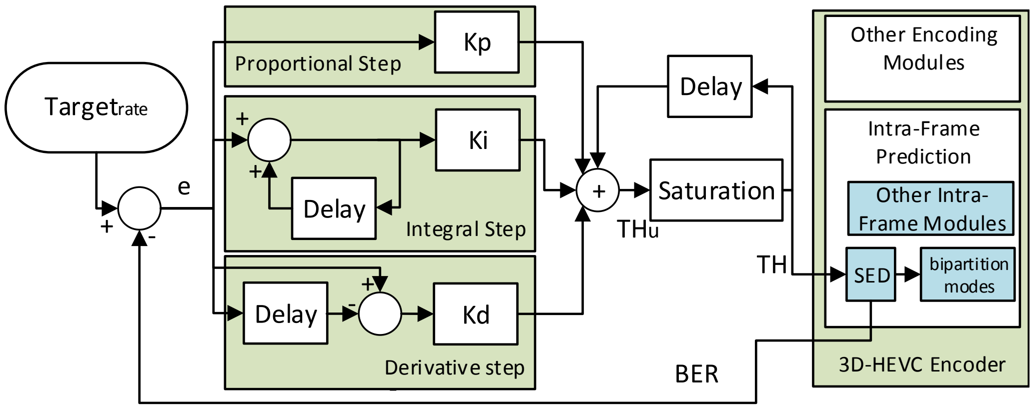


Figure 5.5 – Diagram of the proposed PID control system (source: [65]).

computed using equation 5.1, where the $Target_{rate}$ is subtracted from the Bipartition mode Evaluation Rate (BER) of the previous frame. BER is always delivered by SED when a frame encoding is finished.

$$e[fn] = Target_{rate} - BER \tag{5.1}$$

```
1.  TH               ← initial threshold value
2.  $Target_{rate}$  ← desired rate
3.  N                ← number of frames
4.  e_sum            ← 0
5.  last_e           ← 0
6.  for fn = 1 to N
7.      Encode frame using TH
8.      BER ← percentage of bipartition modes evaluated
9.      e[fn] ← $Target_{rate}$ – BER
10.     e_sum ← e_sum + e[fn]
11.     THu[fn] ← Kp × e[fn] + Ki × e_sum + Kd × (e[fn] - last_e) + TH
12.     last_e ← e[fn]
13.     if (THu[fn] < 1)
14.         TH ← 1
15.     else
16.         TH ← THu[fn]
```

Figure 5.6 – Pseudo-code describing the algorithm of the control system (source: [65]).

The computed error follows three paths (i) a proportional step, (ii) an integration step, and (iii) a derivation step, whose results are added to the threshold of the previous frame ($TH[f_n - 1]$) in equation 5.2 aiming to generate the unbounded threshold (TH_u).

$$TH_u[f_n] = Kp \times e[f_n] + Ki \times \sum_{i=0}^{n} e[i] + Kd \times (e[f_n] - e[f_n - 1]) + TH[f_n - 1] \qquad (5.2)$$

In the proportional step, the error value is multiplied only by a *Kp* constant, changing the threshold proportional to the error obtained by the current frame evaluation.

In the integration step, all the previous error values are added and multiplied by a *Ki* constant, eliminating the residual evaluation error of the system. On first analysis, the integration step may sound a complex operation because all the previous errors need to be added. However, this operation can be reduced to a single sum multiplied by the *Ki* constant because all errors until the current frame have already been added and this information is available, as one can see in the algorithm displayed in Figure 5.6.

In the derivative step, the current error is subtracted from the error obtained in the previous frame and multiplied by a *Kd* constant. The derivative step helps to predict the system behavior promptly, enabling a fast control actuation.

TH_u is generated by adding the results of these three steps with the last frame threshold. A lower bound saturation is applied before delivering this threshold to SED. This saturation prevents the threshold to go below one when 100% of bipartition mode evaluations would be performed. Moreover, in case D_{max} = 0, the region has the highest probability to be a homogeneous region, then the bipartition mode evaluations are always skipped.

5.1.2 Results and Discussion

The proposed control system was implemented in 3D-HTM 16.0 and evaluated with the CTC video sequences were in all-intra configuration with 5%, 10%, and 15% of target rates. These target rates, which considered the information presented in previous experiments, reduce the number of bipartition mode evaluation aggressively. Figure 5.7 depicts the convergence rate of the bipartition mode evaluation to the desired target rate. This experiment considered the first 100 frames of the CTC depth video sequences for block sizes 4×4 and 32×32 (considering only the central view). Several experiments were performed in Balloons video sequence to select the values of the constants *Kp*, *Ki*, and *Kd* to well fit the system without leading to unstable behavior. These experiments led to the values 10, 10, and 1 for *Kp*, *Ki*, and *Kd*, respectively.

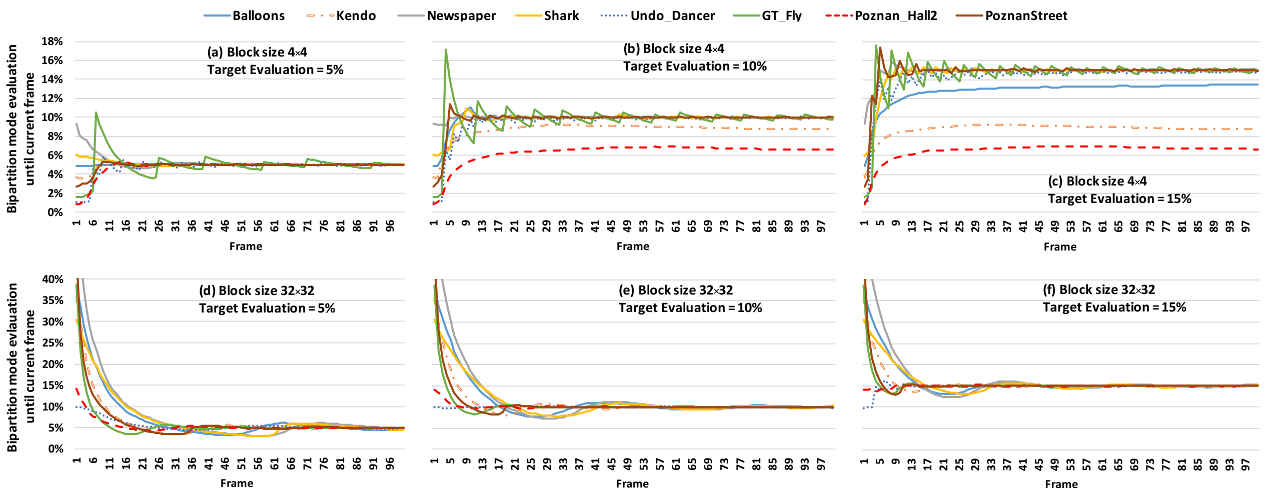


Figure 5.7 – Complexity control results for some scenarios and target evaluations (source: [65]).

Figure 5.7 shows the bipartition mode evaluations have a slower convergence for smaller blocks than for larger ones. However, after a short period, all videos have converged to the target evaluation rate. In some cases, such as the encoding Kendo video sequence with a target rate of 15%, the assessments are limited to a maximum of 7.5%. This limitation occurs because the proposed control system has a lower bound saturation preventing the encoder from performing the bipartition mode evaluation for blocks with $D_{max} = 0$, which has the highest chance to be a homogeneous block.

The target rate affects the encoding efficiency of the proposed complexity control system; targeting a higher number of bipartition mode evaluation increases the encoding efficiency. Table 5.1 displays the encoding efficiency regarding BD-rate versus target rate when comparing our proposal to the original 3D-HTM 16.0, keeping the target rate fixed for all block sizes. For all cases, the BD-rate decreases when the target rate increases; a target rate of 5% implies a BD-rate of 0.5%, on average, and with 15% of target rate, the BD-rate reaches 0.20%, on average.

Table 5.1 – BD-rate results according to three complexity target rates.

Videos	**BD-rate**		
	Target rate = 5%	**Target rate = 10%**	**Target rate = 15%**
Balloons	0.59%	0.35%	0.27%
Kendo	0.57%	0.36%	0.25%
Newspaper	1.11%	0.78%	0.62%
GT_Fly	0.36%	0.16%	0.10%
Poznan_Hall2	0.39%	0.22%	0.13%
Poznan_Street	0.16%	0.09%	0.08%
Undo_Dancer	0.23%	0.08%	0.06%
Shark	0.60%	0.22%	0.10%
Average	0.50%	0.28%	0.20%

The worst result is noticed in Newspaper video sequence with a BD-rate increase ranging from 0.62% up to 1.11% regarding complexity targets. This impact occurs because the original video sequence has a poor depth map quality and regions where the bipartition modes tend to be chosen (such as sharp edges) are composed of many distortions, causing more coding efficiency losses. When analyzing video sequences with a good depth map quality, e.g., GT_Fly, one can notice the BD-rate increase is negligible.

These impacts are acceptable since more than 85% of bipartition modes evaluations were skipped. Considering the bipartition modes use 35.34% of the intra-frame encoder time of the 3D-HEVC depth maps, on average, as described in Chapter 4, the proposed solution can drastically reduce the encoder complexity, and maintain this complexity over a well-defined range of values. Table 5.2 displays the timesaving of the depth map coding provided by our solution in the three target rates, considering texture and depth execution time. Notice our solution achieved an average timesaving of 29.9%, 27.9% and, 19.3%, according to the selected target rate. Although Figure 5.7 demonstrates the bipartition mode evaluations are stabilized, Table 5.2 shows there is still significant variation in the timesaving because the evaluated videos have different content characteristics. Therefore, other tools used in the depth map coding are responsible for different complexities (that are not stabilized) since we only employ the control system for bipartition modes.

In this evaluation, the Newspaper video sequence achieved the highest timesaving gains. Again, it occurs because of the low quality of the original depth map. The timesaving obtained by this sequence ranges from 24.5% to 35.4%, according to the target rate.

Table 5.2 – Timesaving of the depth map coding according to complexity target rates.

Videos	Timesaving		
	Target rate = 5%	Target rate = 10%	Target rate = 15%
Balloons	32.0%	31.8%	21.5%
Kendo	32.3%	31.4%	20.6%
Newspaper	35.4%	34.9%	24.5%
GT_Fly	29.2%	25.9%	18.7%
Poznan_Hall2	26.2%	22.5%	13.1%
Poznan_Street	28.5%	26.5%	20.4%
Undo_Dancer	28.0%	25.2%	17.6%
Shark	27.3%	25.0%	18.2%
Average	29.9%	27.9%	19.3%

5.2 Parallelism Exploration for Intra_Wedge

Two parallel architectures were used in this evaluation: (i) a symmetric multiprocessor, with few but powerful multi-cores containing memory coherence supported by hardware, as the current multi-core systems; and (ii) a massively parallel GPU, with thousands of simple cores operating under the Single Instruction Multiple Data (SIMD) paradigm [17], which enables to explore data parallelism and stream computing. Additionally, two parallel approaches were investigated to handle the Intra_Wedge computation, allowing the parallelism exploration at block-based and pattern-based granularities.

Figure 5.8(a) illustrates the main idea of the block-based approach. Since Intra_-Wedge evaluates a block independent of its neighborhood, each block can be assigned to a given processing unity for a massive parallel encoding. This approach considers a thread is responsible for the entire Intra_Wedge encoding of a depth block, applying the Main, Refinement and Residue stages.

Since Intra_Wedge evaluates several wedgelet patterns, the proposed pattern-based approach assigns the evaluation of each pattern to a thread, as shown in Figure 5.8(b). The wedgelets evaluated in the Main stage are distributed to the threads for balancing their computational effort. Therefore, the loop presented in the original Intra_Wedge algorithm is no longer required, and each thread evaluates the encoding blocks for a given pattern using the flow presented in Figure 5.8(c). After computing the distortion of all wedgelets, the threads are synchronized, and the distortion results are compared for selecting the best one. In the Refinement stage, the wedgelet patterns are also assigned to multiple threads with similar processing.

The original SED [67] and GMOF [66] heuristics were applied to the block-based parallel approach to evaluating how the Intra_Wedge simplification could bring gains in a parallel environment. These two algorithms were selected due to their lightweight implementations. However, the same analysis could be extended to other algorithms with similar

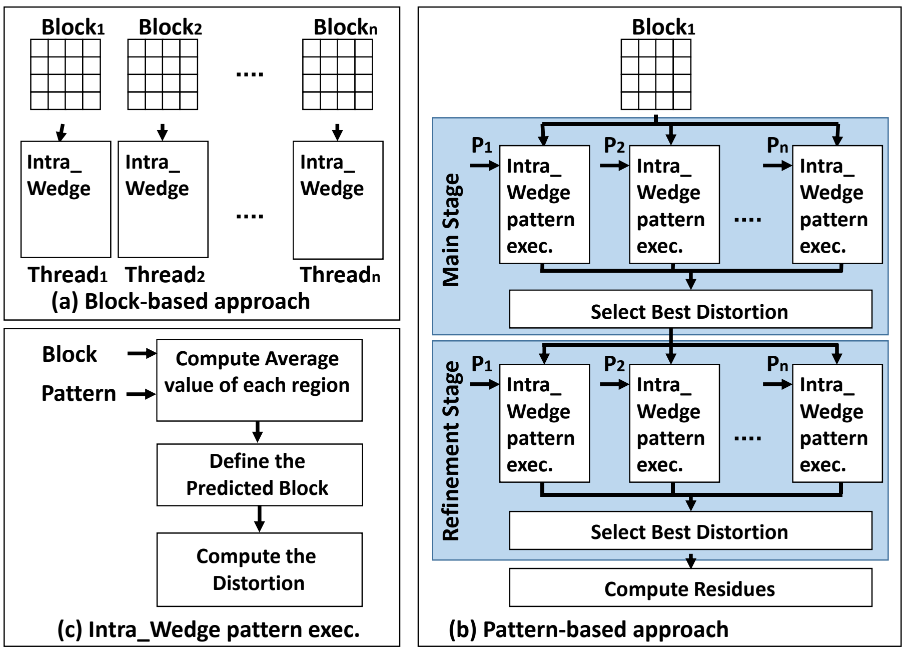


Figure 5.8 – Parallelism exploration: (a) block-based, (b) pattern-based, and (c) Intra_Wedge pattern execution (source: [59]).

characteristics that use block- or mode-level approaches; except if the algorithm contains data dependencies among the blocks inside a given frame, which is the case of DFPS presented in Section 4.2.1.

We are investigating the characteristics (block- or mode-level) of the simplification algorithms that allow obtaining the best benefits for each parallel architecture. Considering 3D-HTM was developed to evaluate the efficiency of the encoder tools, and it is not suitable for evaluating the computational performance since its execution time is far from being optimal, we compute the Intra_Wedge with a new C++ sequential program. Besides, we implemented two other versions, one using the SED heuristic and another one using the GMOF heuristic. OpenMP 7.0 [13] and CUDA 3.1 [43] were used to program the proposed algorithms for the Central Processing Unit (CPU) and Graphics Processing Unit (GPU), respectively.

The experiments encompass programs encoding the central view of the eight available videos at CTC. The software inputs are the raw data of depth maps and the reconstructed texture video obtained with 3D-HTM. The software provides the residual data and the selected Intra_Wedge pattern.

We executed the experimental evaluation on an Intel Core i7 (Skylake) 6700K with 4 cores (8 threads) running at 3.5 GHz, with a 32 GB DDR3 memory. Two GPUs were used in this evaluation: (i) NVidia GTX Titan X (GM200 – Maxwell) with 3072 CUDA cores running at 1.08 GHz (called Titan X in the rest of this Section) and; (ii) NVidia Titan Xp (GP102 – Pascal) with 3840 CUDA cores running at 1.48 GHz (called Titan Xp in the rest of this Section). Although we performed the evaluations using a four-core CPU and two

Nvidia GPUs, the experiment aims to show the encoding blocks can be assigned to multiple cores to speed up the encoding process. Besides, the proposed methods can be applied in systems with different levels of parallelism, resources and programmability characteristics, such as Multiprocessor System-on-Chip (MPSoC) or dedicated hardware designs.

The experimental results cover the following analysis: (i) parallelism granularity; (ii) scalability; and (iii) Intra_Wedge simplifications in multicore systems.

5.2.1 Parallelism Granularity Analysis

Figure 5.9 displays the evaluation of both block- and pattern-based approaches in the CPU, comparing the execution time of eight threads to a single thread. The average time required to process a frame with 1024 $\times$ 768 pixels is 5.0 s and 5.9 s for block- and pattern-based approaches, respectively; whereas, for a single thread, this time is 30.5 s. The time required to process a frame with 1920 $\times$ 1088 pixels was reduced from 83.2 s to 16.2 s and 13.6 s, when the pattern- and block-based approaches are applied, respectively. These results highlight the timesaving when encoding blocks in parallel on a multicore CPU.

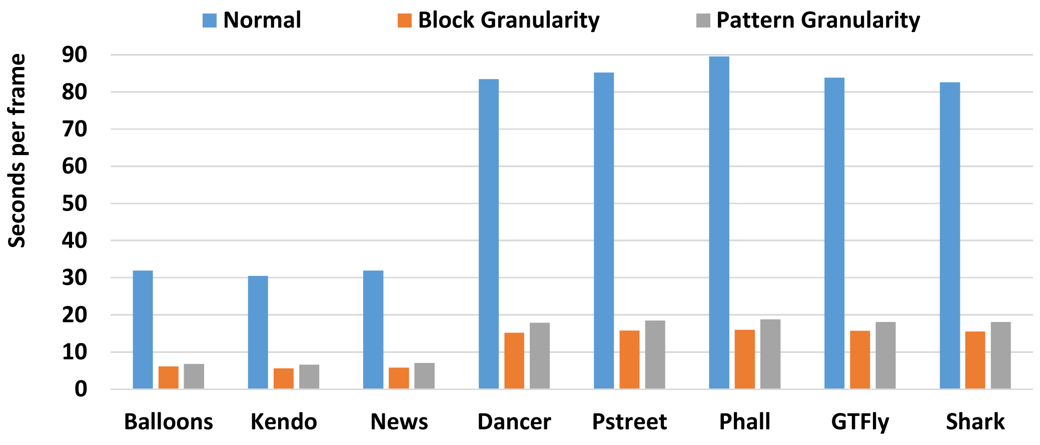


Figure 5.9 – Encoding time per frame considering the evaluated approaches (source: [59]).

The attained speedup over the single thread version is similar for the two frame sizes, reaching on average 6.1 and 5.1 for block- and pattern-based approaches, respectively. Since the best results were always achieved with the block-based approach, this approach is used in the next experiments, where scalability and the usage of Intra_Wedge simplifications are evaluated. It is important to mention the proposed parallel approaches do not insert losses, regarding the 3D-HEVC encoded video quality and bitrate.

5.2.2 Scalability Analysis

This section analyzes the scalability of the block-based approach by varying from one to eight the number of CPU threads. Figure 5.10(a) shows the speedup results of this scalability evaluation taking the single-thread as reference. One can notice that the results obtained in different video sequences are very similar (the curves are almost overlapped).

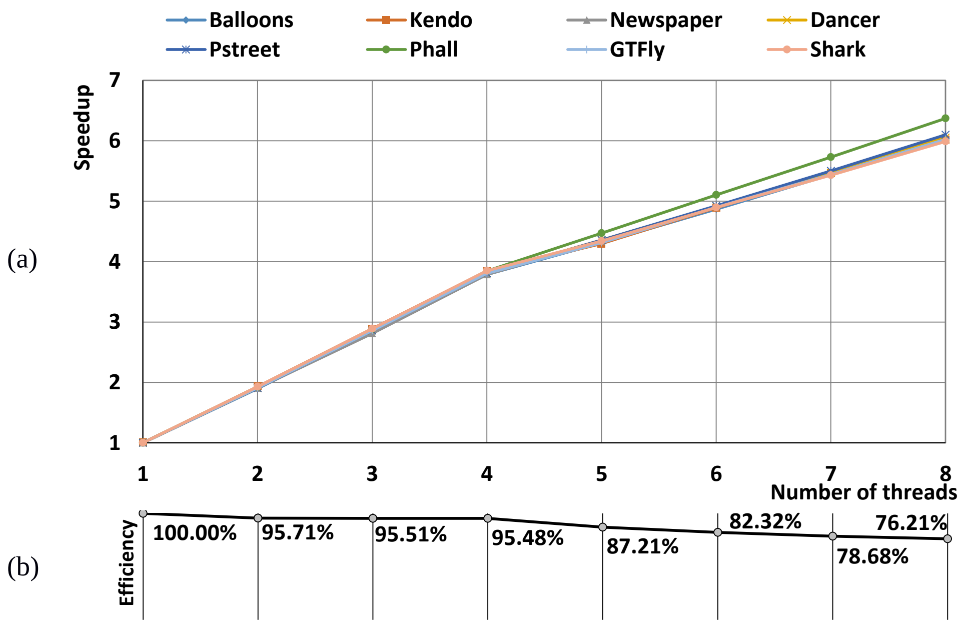


Figure 5.10 – Scalability Analysis - (a) eight speedups and (b) the efficiency according to the number of threads for the block granularity (source: [59]).

Figure 5.10(b) shows the efficiency regarding the number of threads that is given by equation 5.3, where E_N is the efficiency using N threads, $Time_1$ is the average time required to encode with a single thread and $Time_N$ is the average time required to encode using N threads.

$$E_N = \frac{Time_1}{Time_N \times N} \times 100\% \qquad (5.3)$$

Notice when using eight threads, the proposed approach speeds up 6.1 the single thread version, on average. Besides, the speedup decreases after five threads because the system has only four physical cores; when more than four threads are launched, the hyper-threading is activated limiting the speedup. However, the speedup curve does not saturate, demonstrating that higher speedups could be attained when using processors with a higher number of CPU cores. Besides, the efficiency of our parallelism decreases with the grow of the number of threads. It happens because in our evaluation it was considered the encoding time spent in frame reading and data transfers.

5.2.3 Intra_Wedge Simplification Analysis in Multicore CPU

We integrated SED and GMOF into our Intra_Wedge parallel programs to show the proposed parallel approaches can be further integrated with Intra_Wedge simplification algorithms. Figure 5.11 depicts the speedup results using the two simplification schemes for a target architecture varying from 1 to 8 threads. The results are normalized according to the computation time of a single thread Intra_Wedge algorithm without simplification.

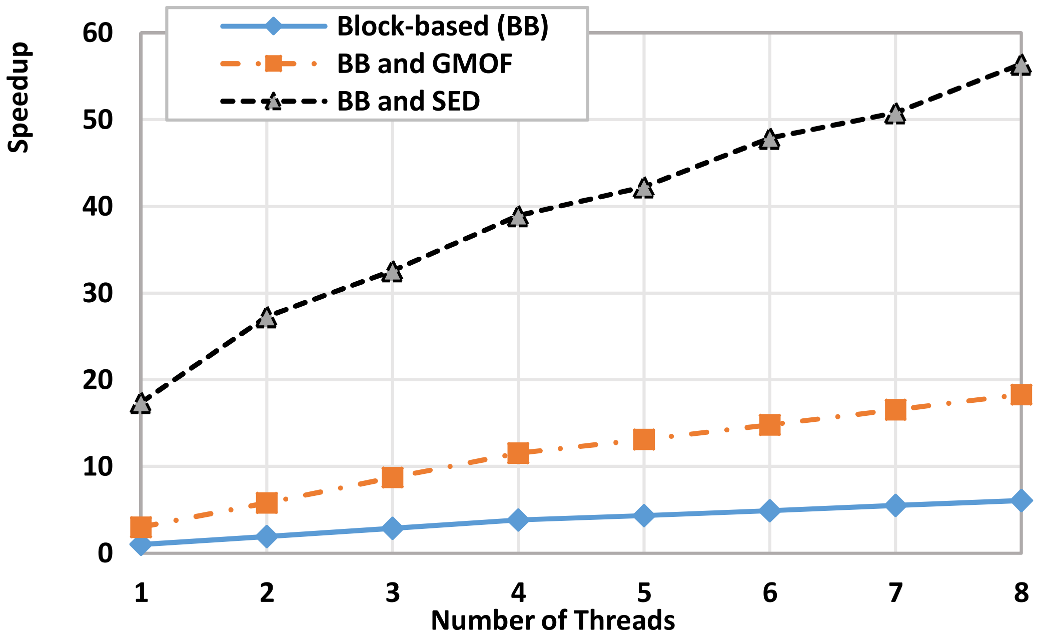


Figure 5.11 – Scalability analysis with Intra_Wedge simplifications (source: [59]).

The y-axis of Figure 5.11 shows both algorithms scale with the number of cores. Table 5.3 summarizes the average time per encoded frame in the multicore CPU. For the 1024 × 768 videos, the encoding time per frame can be further reduced to 1.7 s and 0.6 s when the GMOF and SED simplifications are applied, respectively. While for the 1920 × 1088 frames, GMOF and SED can reduce the encoding time per frame to 4.5 s and 0.5 s, respectively.

Table 5.3 – Multicore CPU results.

Implementation		**Time per frame (seconds)**	
		1024 × 768	**1920 × 1088**
Single thread		30.5	83.2
Eight threads	**Block-based approach**	5.0	13.6
	Block-based approach with GMOF	1.7	4.5
	Block-based approach with SED	0.6	0.5

5.2.4 Intra_Wedge Implementation using GPU

We first used the TITAN X GPU in our analysis and later we used the TITAN Xp GPU to obtain the final results, in order to show the potential of further data parallelism in Intra_Wedge execution. We implemented the Intra_Wedge algorithm with TITAN X adopting the same block-based approach employed in the multicore CPU implementation. It is expected this approach provides even better results since data parallelism is fundamental to use the GPU's resources efficiently. We first evaluated the processing rate in frames per second (fps). Table 5.4 summarizes these results along with the time spent during the data transfer between CPU and GPU, allowing visualizing the percentages of processing and data transfer.

Table 5.4 – Results for the basic GPU implementation.

Resolution	**Videos**	**Processing rate (fps)**	**Communication and frame read percentage**
1024 $\times$ 768	**Balloons**	18.6	0.9%
	Kendo	18.6	0.9%
	Newspaper_cc	18.5	0.9%
	Average	18.6	0.9%
1920 $\times$ 1088	**GT_Fly**	14.6	1.8%
	Poznan_Hall2	14.6	1.8%
	Poznan_Street	14.7	1.7%
	Undo_Dancer	14.7	1.8%
	Shark	14.6	2.0%
	Average	14.6	1.8%

The GPU implementation achieves 18.6 fps@1024 $\times$ 768 and 14.6 fps@1920 $\times$ 1088. Comparing the processing rate with the CPUs one, one can notice that a higher increase in the processing rate is obtained for higher resolution videos, because more blocks are encoded by the CUDA cores, increasing the data parallelism explored. Moreover, the time spent on data transfers represents less than 2%, which means that most of the time is spent on the GPU processing. However, further investigation is required to obtain an improved processing rate for real-time encoding of HD 1080p videos.

We improved the basic GPU implementation by increasing the number of CUDA streams to maximize the usage of CUDA cores and by taking advantage of the GPU constant memory. The optimization on CUDA streams allows transferring data between CPU and GPU asynchronously and executing each kernel in a different stream, improving the use of the CUDA cores. Moreover, the constant memory was used to store the 4 $\times$ 4 and 8 $\times$ 8 pattern blocks. The 16 $\times$ 16 patterns were not stored in the constant memory because there was not enough space for them in the TITAN X (neither in TITAN Xp that will be used later). Therefore, higher speedup can be achieved by employing a GPU with larger constant memory.

Figure 5.12 shows the processing rate and the speedup obtained by comparing the optimized GPU implementation to the original CPU version. On average, the system was capable of reaching 26.3 fps for 1024 $\times$ 768 videos and 18.2 fps for 1920 $\times$ 1088 videos.

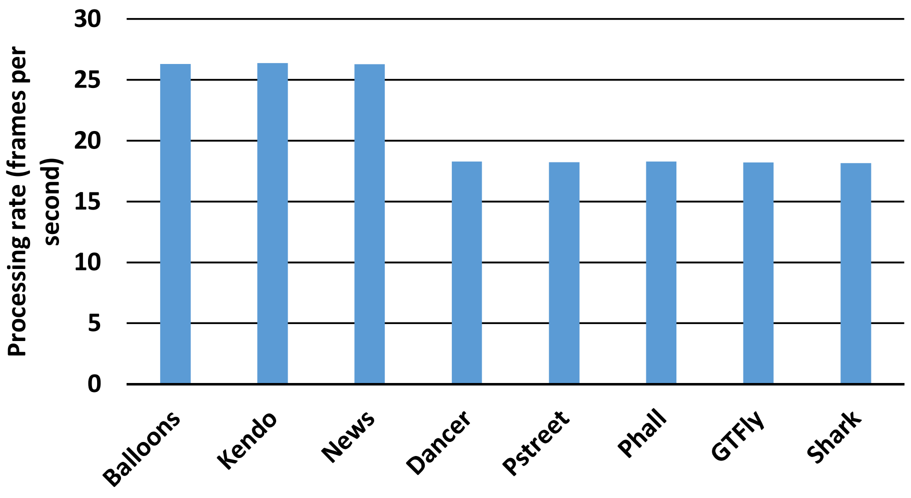


Figure 5.12 – Results of the GPU optimization (source: [59]).

We have also implemented and evaluated the two simplification techniques for Intra_Wedge, GMOF and SED, in the TITAN X GPU, keeping the streams and the constant memory optimizations. Table 5.5 shows that, on average, GMOF and SED allow encoding 1024 $\times$ 768 resolution videos at 48.1 and 30.1 fps, respectively, while around 30 fps are encoded for 1920 $\times$ 1088 resolution videos. On 1024 $\times$ 768 videos, the best performance is achieved with GMOF, unlike the multicore CPU that takes more advantage of the SED. It happens because all threads of the GPU have to execute the same code in parallel in each warp. Consequently, if the SED algorithm of a thread decides to skip the Intra_Wedge evaluation, threads diverge and have to wait until the remaining blocks finish their computation. In this context, SED only takes full advantage of the GPU resources when all blocks inside a warp are skipped, i.e., when all blocks in the warp are classified as homogeneous. However, GMOF algorithm always provides a significant speedup. Considering the different characteristic of the two types of architectures, one can conclude that a multicore CPU execution can obtain higher benefits from algorithms that skips the entire Intra_Wedge evaluation, while GPUs obtain better results when applying algorithms that accelerate the processing of every executing block in a convergent way.

We also evaluated these two Intra_Wedge simplification techniques into a quite recent GPU platform: the NVidia TITAN Xp. The reached results for this second GPU implementation is also presented in Table 5.5, considering the block-based approach with optimizations, and with the usage of GMOF and SED techniques. Similar conclusions than that reached for the previous experiment with the TITAN X GPU can be drawn. Again, GMOF algorithm presented higher processing rates because SED algorithm requires that some

Table 5.5 – GPU results for the GMOF and SED implementations.

Resolution	Videos	TITAN X (fps)			TITAN Xp (fps)		
		Block-based	GMOF	SED	Block-based	GMOF	SED
1024 × 768	Balloons	26.3	50.3	30.2	40.5	106.8	42.3
	Kendo	26.4	49.2	31.7	40.6	102.4	45.0
	Newspaper_CC	26.3	44.9	28.7	40.4	87.2	40.3
	Average	26.3	48.1	30.1	40.5	98.8	42.5
1920 × 1088	Undo_Dancer	18.3	30.9	27.3	26.3	56.1	36.9
	Poznan_Street	18.2	30.1	37.1	26.2	51.7	56.1
	Poznan_Hall2	18.3	31.3	32.9	26.5	57.7	46.8
	GT_Fly	18.2	30.0	31.6	26.2	53.3	56.3
	Shark	18.1	29.9	27.0	26.0	54.2	37.6
	Average	18.2	30.4	30.8	26.2	54.6	44.6

threads perform the full Intra_Wedge algorithm without any simplification. Therefore, the threads that skip Intra_Wedge processing early wait until all threads inside the warp finish. The processing rates in this evaluation reached up to 98.8 fps for 1024 × 768 videos and 54.6 fps for 1920 × 1088 videos using GMOF algorithm.

Finally, Table 5.6 summarizes the multicore CPU and GPU results obtained along this work. Our GPU implementation increases the processing rate from 0.03/0.01 fps (for 1024 × 768 / 1920 × 1088 videos) to 98.8/54.6 for these resolutions when using GMOF technique running on a TITAN Xp.

Table 5.6 – Summary of results obtained with CPU and GPU implementations.

Implementation		Frames per seconds	
		1024 × 768	1920 × 1088
Single thread		0.03	0.01
CPU - Eight threads	Block-based approach	0.22	0.07
	Block-based approach with GMOF	0.59	0.22
	Block-based approach with SED	1.80	2.05
TITAN X	Block-based approach	26.30	18.20
	Block-based approach with GMOF	48.10	30.10
	Block-based approach with SED	30.40	30.80
TITAN Xp	Block-based approach	40.50	26.20
	Block-based approach with GMOF	98.80	54.60
	Block-based approach with SED	42.50	44.60

Experimental evaluation was made for a specific CPU and two specific GPUs. However, one can conclude that the proposed parallel approach is scalable, then it can be extended to speed up the Intra_Wedge execution on other parallel systems, with different characteristics, such as MPSoCs or dedicated hardware design.

The best performance was achieved by applying the proposed parallel strategies and exploring the SED and GMOF simplification techniques. The parallelism exploration using both SED and GMOF techniques can be reached because these simplifications do not contain dependencies with the remaining blocks in the current frame.

Considering the previous discussions about the reached results one can conclude that Intra_Wedge can be a good candidate for being accelerated using massive parallel approaches. In all cases, the frame rates were scalable with the explored parallelism according to the target CPU/GPU architectures. It is important to emphasize that the Intra_Wedge tool represents around 20% of the 3D-HEVC computational effort according to our previous evaluations, then this is an important tool that must be accelerated to allow the design of real-time encoders able to process high resolution videos at 30 fps or more. A similar approach than that used here for Intra_Wedge tool can be used for other encoder tools allowing a high throughput also for these modules. Besides, other solutions can also be explored in the other encoder modules, targeting the system acceleration, including the use of multiple GPUs, MPSoCs, dedicated designs or other high-performance solutions. Then, integrating these solutions will be possible to have a complete 3D-HEVC encoder processing high resolution videos in real-time.

5.3 Intra_Wedge Pattern Compression

One of the bottlenecks in Intra_Wedge hardware design is the efficient storage of all wedgelet patterns due to a large number of allowed patterns. The 3D-HTM software stores all wedgelet patterns without compression using $N \times N$ bits for each pattern to fulfill the beginning of the encoder or decoder execution. 3D-HTM uses this strategy to keep high performance during software execution. However, this strategy is highly inefficient when the Intra_Wedge encoder or decoder is implemented in dedicated hardware design.

Table 2.1 (Section 2.2.2) illustrated the Intra_Wedge implemented inside 3D-HTM requires 183,264 bits to store all wedgelet patterns of all block sizes. Notice the wedgelet patterns of 32×32 blocks are not stored because they are obtained upscaling the wedgelets of 16×16 blocks. Furthermore, the Intra_Wedge encoder algorithm implemented in 3D-HTM requires frequent access to the wedgelet patterns, implying in a significant power dissipation caused by these memory accesses. Thus, essential requirements for dedicated Intra_-Wedge encoder design are memory size reduction and efficient memory access. For the remainder of this section, the memory that stores the wedgelet patterns is called *WMem*.

5.3.1 First Bit and Change (FB&C) Algorithm

The First Bit and Change (FB&C) algorithm is grounded in the fact the Intra_Wedge divides each block into two and only two regions. Thus, Intra_Wedge does not represent interlaced ones and zeros in a single line leading to an inefficient storage model if $N \times N$ bits are stored per pattern. For example, in a 4×4 block, each line should never contain the pattern "0010" because no wedgelet can describe this line. It happens because the Intra_-Wedge algorithm was designed for partitioning and encoding blocks into two regions using a straight line. Consequently, patterns containing more than one change of region in a line are forbidden. Thus, $N \times N$ blocks allow only $2 \times N$ lines in Intra_Wedge patterns. Table 5.7 presents all allowed lines for 4×4 blocks along with the respective FB&C codes. Note the half of the bits combination (patterns) are forbidden.

Table 5.7 – FB&C coding for line patterns of a 4×4 block.

Allowed line	**FB&C code**
0111	000
0011	001
0001	010
0000	011
1000	100
1100	101
1110	110
1111	111

Figure 5.13(a) illustrates the FB&C technique that codifies each pattern copying the 1st bit content of the original pattern to the coded one. Let RB be the number of Remaining Bits required for coding a line of the $N \times N$ block, which can be obtained applying equation 5.4. Then, RB specifies in how many subsequent bits the coded line will change to the other region. Figure 5.13(b) displays two encoding examples of 8×8 block lines. The first example shows how to encode the "00000011" line with FB&C representation. The 1st bit "0" is just copied, and the encoding algorithm seeks for the first occurrence of "1"; i.e., in the 5th position. Then, the value five (related to the 5th position) is stored after the 1st bit, completing the FB&C representation of the first example. The second example shows a line containing only a single region of "1s". Therefore, FB&C codifies this line copying the 1st bit followed by the $N - 1$ value (seven).

$$RB = log_2(N) \tag{5.4}$$

5.3.2 Huffman Code

Huffman Code is a traditional algorithm in the data compression field, which creates a prefix binary code tree with minimum expected code-word length [26]. However, this is the first work in literature proposing the usage of huffman applied to the compression of wedgelets patterns. Here, Huffman works reducing the representation of the most-frequent

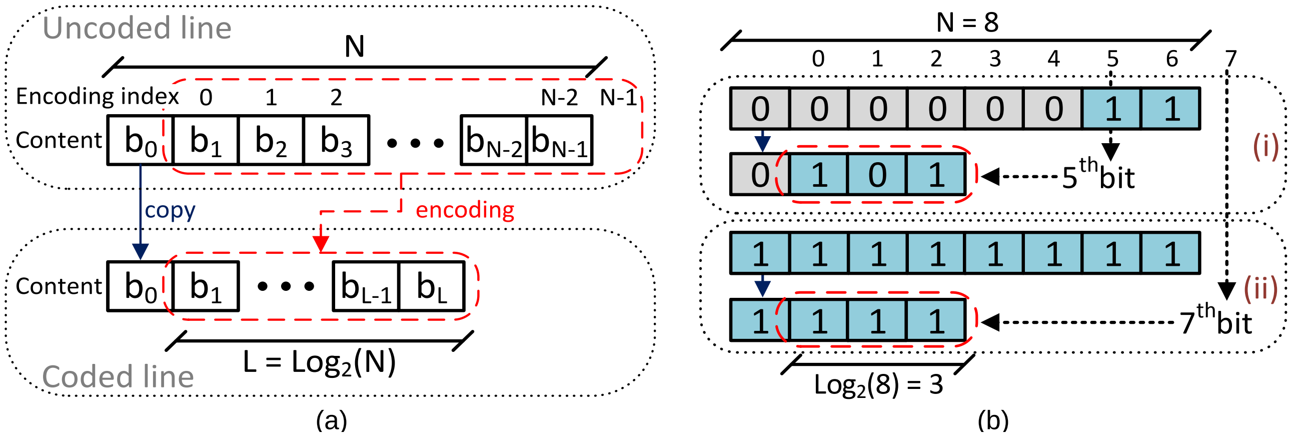


Figure 5.13 – (a) FB&C encoding model and (b) two pattern examples for 8 $\times$ 8 blocks (source: [53]).

used lines and increasing the representation of the less-frequent used lines. The statistical analysis to create the Huffman Code tree is performed statically because wedgelets are predefined before the encoding execution and then a statistical analysis at runtime is not necessary.

Table 5.8 shows the allowed lines, its occurrence probability (extracted from the static analysis of all wedgelet patterns) and the Huffman Code for all possible patterns of 4 $\times$ 4 block lines. One can notice the “0000” and “1111” are the most often used patterns being represented by only 2 bits, while “0011” and “0001”, which are the less often patterns, are represented by 5 bits, resulting in a reduction in the storage requirements.

Table 5.8 – Occurrence probability and Huffman Code for the allowed lines of 4 $\times$ 4 blocks.

Allowed line	**Probability**	**Huffman Code**
0111	6.10%	0001
0011	5.81%	00001
0001	6.10%	00000
0000	21.52%	10
1000	12.50%	010
1100	15.12%	001
1110	12.50%	011
1111	20.35%	11

5.3.3 Block Change Map Algorithm

Block Change Map (BCM) is an algorithm planned to reduce the entropy of the wedgelet patterns employing an auxiliary matrix containing the differences between subsequent wedgelets. For each wedgelet coding, the BCM algorithm stores the wedgelet pattern read from the standard *WMem* into the $Pattern_{k+1}$ buffer. Additionally, the content of the $Pattern_{k+1}$ is copied to the $Pattern_k$ buffer. Consequently, $Pattern_{k+1}$ contains the most recently read wedgelet, whereas $Pattern_k$ contains the previous one.

The algorithm fills an auxiliary matrix of bits with the comparison between $Pattern_k$ and $Pattern_{k+1}$. The positions containing a change are signalized with "1" while the remaining positions are signalized with "0". Subsequently, the algorithm applies Huffman Code to the content of this matrix, producing the wedgelet coded in the BCM format. The exception occurs only in the 1st wedgelet since at the first time only $Pattern_{k+1}$ contains a valid pattern; thus, the algorithm applies Huffman Code on the content of $Pattern_{k+1}$ instead of on the auxiliary matrix.

The coded patterns are stored into the coded *WMem* reducing the storage requirement significantly when compared to the approach that uses only Huffman Code, as demonstrated in the section of experimental results.

Figure 5.14 shows a block diagram of the BCM architecture, exemplifying the coding of the first two 4 × 4 wedgelet patterns. This example encodes the first wedgelet pattern with Huffman. Subsequently, the Change bitmap generator fills the auxiliary matrix comparing bit by bit the first wedgelet with the subsequent one, resulting in a single bit difference. The content of the matrix is encoded using Huffman. Finally, all encoded data are inserted into the coded *WMem*. Remark the output of Huffman code is different from the one presented in Table 5.8 because BCM produces other probabilities of pattern occurrence.

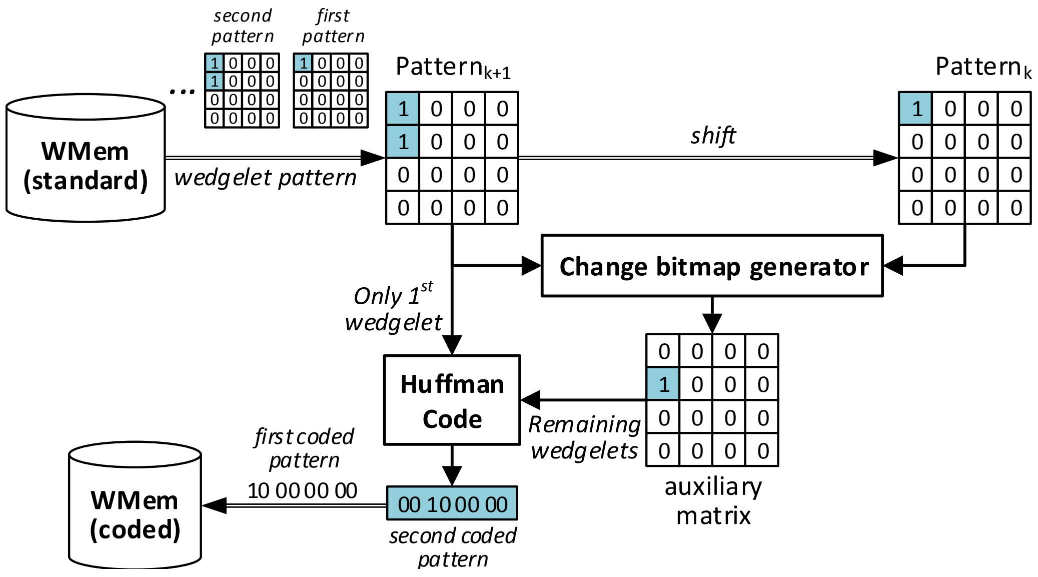


Figure 5.14 – Block diagram of the BCM architecture exemplifying the coding of two 4 × 4 wedgelet patterns (source: [64]).

5.3.4 Line Change Map Algorithm

The Line Change Map (LCM) algorithm works similarly to the BCM algorithm; however, LCM generates a bitmap change for each line, exploring the vertical redundancies among all lines of all blocks of the same size; i.e., the algorithm connects all wedgelets considering the first line of a subsequent wedgelet needs to be compared to the last line of the previous wedgelet pattern. Instead of having $Pattern_k$, $Pattern_{k+1}$ and auxiliary matrix, the LCM algorithm requires only vectors as buffers, thus, corresponding the analogous $Line_k$, $Line_{k+1}$, and auxiliary vector, respectively. Notice the BCM algorithm reads only lines from the standard *WMem*; besides, it stores only coded lines into the coded *WMem*.

Figure 5.15 exemplifies this technique applied to the five first lines of the standard *WMem* regarding 4×4 block sizes, which implies the coding of two subsequent wedgelets.

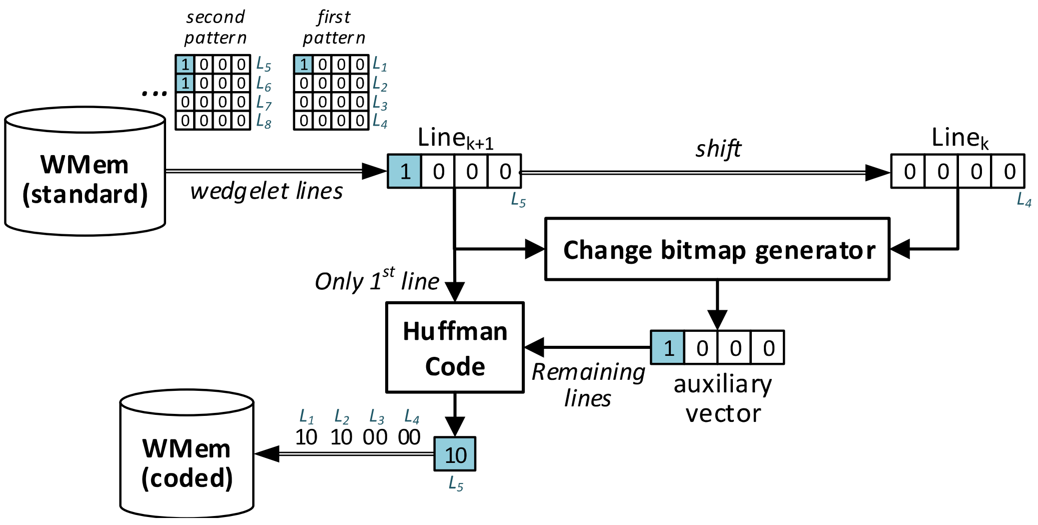


Figure 5.15 – Block diagram of the LCM architecture exemplifying the coding of five lines of the 4×4 wedgelet patterns. Line buffers are filled for generating the L_5 coded line (source: [64]).

5.3.5 Block Line Change Map Algorithm

The LCM algorithm groups logically the wedgelet list before performing the bitmap change. When the first line pattern of a subsequent wedgelet differs significantly from the last line pattern of its predecessor wedgelet, the Change bitmap generator produces more bit-changes, reducing the encoding efficiency. Block Line Change Map (B-LCM) was proposed to mitigate this problem, as an improvement of the LCM algorithm. B-LCM applies FB&C to encode the first line of all wedgelets, and the remaining lines of each wedgelet are encoded using LCM. This procedure makes the encoding of the wedgelet pattern independent from previous or subsequent patterns.

5.3.6 Dual First Bit and Change Algorithm

FB&C is grounded on the fact that Intra_Wedge cannot contain interlaced ones and zeros in a single row, enabling to explore horizontal redundancies. However, the same restriction (inexistence of interlaced ones and zeros) occurs in the column patterns. Thus, Dual First Bit and Change (D-FB&C) was proposed to explore both horizontal and vertical redundancies. Figure 5.16(a) exemplifies this coding technique in an 8×8 block size, where the first bit of the block is stored without compression; then, the first column of the block is encoded with the position of the first bit-change (similar to the explanation of FB&C), which happens in the 2*nd* bit of the first column of the example. Therefore, each line of the pattern is inserted into the memory being encoded with the position of the bit-change as FB&C does; however, without requiring the storage of the 1*st* bit of each row, since the encoded first column can provide this information.

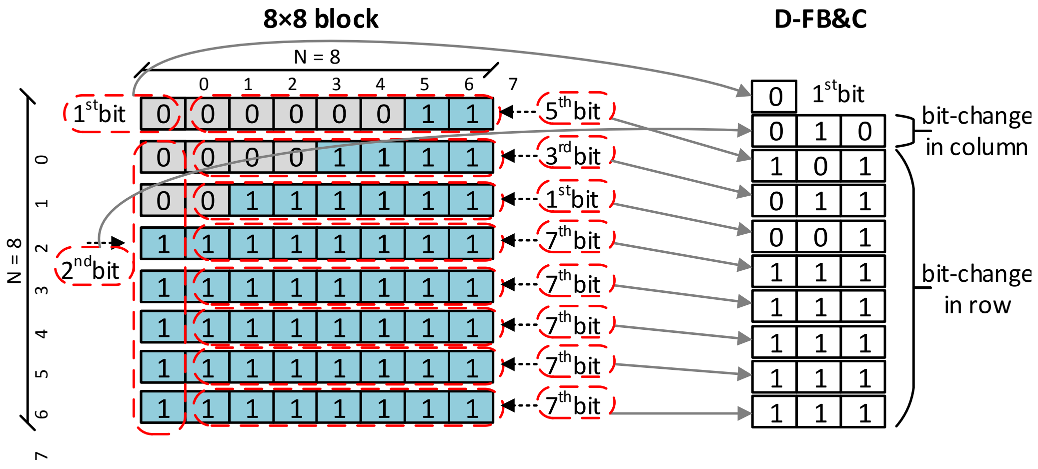


Figure 5.16 – D-FB&C encoding example with an 8×8 wedgelet pattern (source: [64]).

5.3.7 Ending Rows Removal Technique

Ending Rows Removal (ERR) is a technique planned to reduce the storage area removing consecutive vectors with the same bit pattern, especially when large blocks are encoded. The technique is based on two aspects: (i) Intra_Wedge defines only a single bit variation in each row or column, and (ii) this variation may result in many consecutive rows or columns with the same pattern. Correctly, the ERR technique is applied for row removal, but it could be applied to column reduction, due to the symmetrical characteristics of the blocks and wedgelets.

Let (R_j, C_k) be the row-column pair representing the position of a bit inside a block with $1 \leq j \leq N$ and $1 \leq k \leq N$, then C_1 and C_N represent the leftmost and rightmost columns of the block, respectively. Whenever a wedgelet crosses C_1 or C_N, there is at least one row with a pattern where all bits are equal (i.e., all bits of the row are "0s" or "1s"). Additionally, there is a bit-change (i.e., $0 \rightarrow 1$ or $1 \rightarrow 0$) for each crossing. Considering these bit-changes represented by the pairs (R_a, C_1) and (R_b, C_N), then any and all row between $Max(R_a, R_b)$ and R_N have the same pattern where all bits are equal, allowing to apply the ERR technique. Based on this observation, the ERR algorithm skips adding information into *WMem* when all subsequent rows have the same pattern of the current one.

Figure 5.17 exemplifies three wedgelet patterns for 8×8 blocks. Figure 5.17(a) shows a wedgelet pattern that crosses only one column, the rightmost one (C_N), whose bit change is placed in the pair (R_5, C_8); thus, all rows from R_5 to R_8 have the same pattern "11111111". Figure 5.17(b) exemplifies a wedgelet pattern with bit-changes placed on pairs $(3, 1)$ and $(5, 8)$; since $Max(3, 5)$ is 5, the rows 5 to 8 have the same pattern. However, Figure 5.17(c) shows a wedgelet pattern that does not cross the columns C_1 and C_N; in this case, there is no occurrence of repeated patterns.

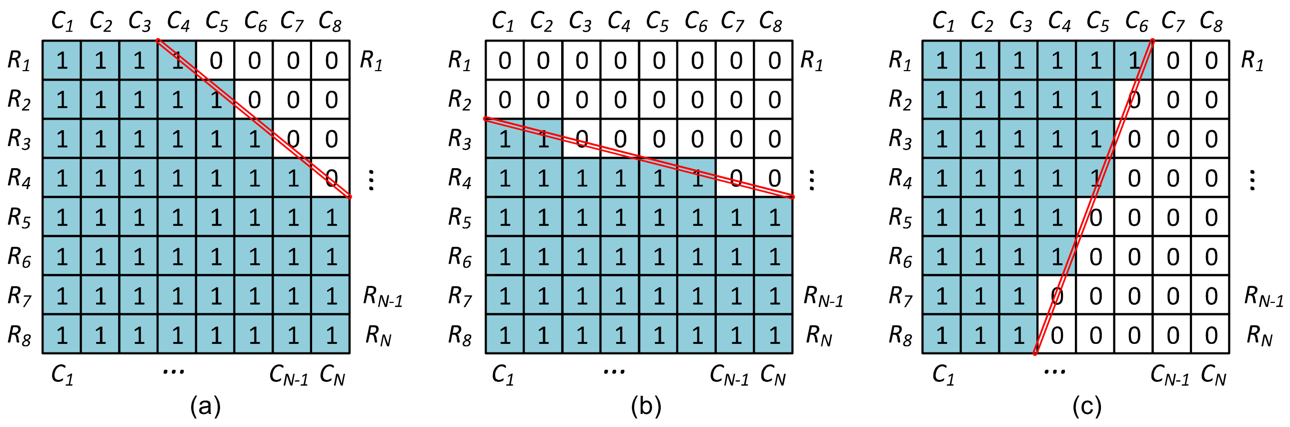


Figure 5.17 – Example of wedgelet patterns for 8×8 blocks (source: [64]).

The ERR technique can be applied together with all other encoding algorithms. This technique is signalized inserting the symbol (+) after the basic encoding algorithm; for instance, D-FB&C+ means the ERR technique is applied together with D-FB&C.

Figure 5.18 shows the same wedgelet pattern of Figure 5.16 to compare FB&C, D-FB&C, and D-FB&C+, considering an 8×8 block. The example shows D-FB&C+ reduces 12 bits compared to D-FB&C, representing 42.8% of compression for this 8×8 block. Besides, when D-FB&C+ is compared to FB&C, which does not explore vertical redundancies, it reduces by half the memory required to store that pattern (i.e., from 32 bits to 16 bits).

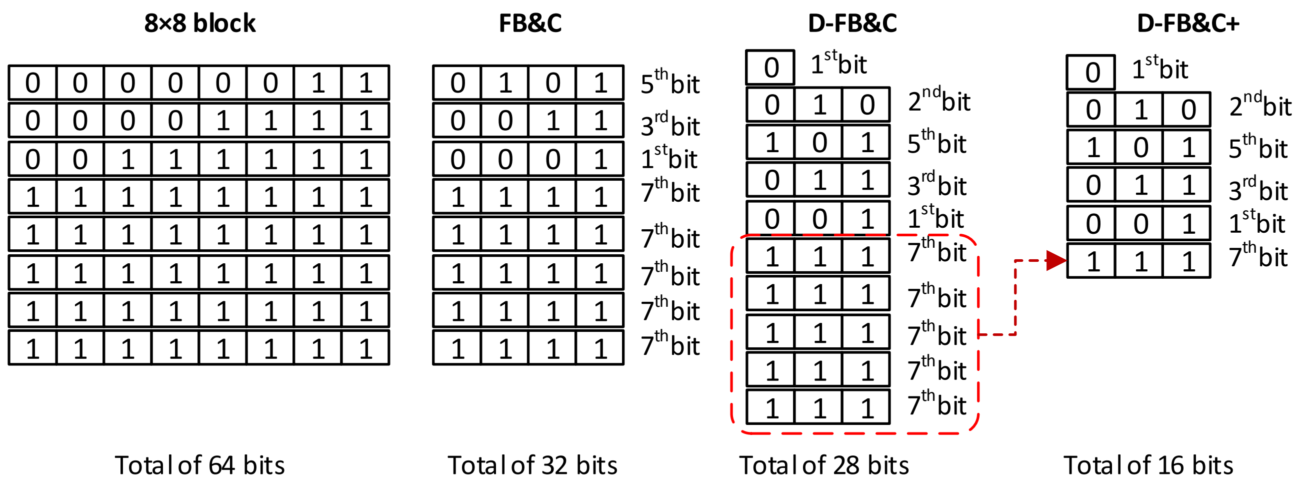


Figure 5.18 – Wedgelet pattern codified with FB&C, D-FB&C, and D-FB&C+ (source: [64]).

5.3.8 Evaluation of the Proposed Algorithms

All wedgelet patterns employed on the experimental results were extracted from 3D-HTM, and the proposed solutions were implemented using Matlab. Additionally, D-FB&C+ was implemented using 65 nm CMOS technology to extract synthesis results.

Table 5.9 illustrates the number of bits required to store all wedgelet patterns of all block sizes of the standard Intra_Wedge, the proposals of [34] and [63], and the solutions proposed here. Besides, the ERR technique was evaluated with FB&C, B-LCM, and D-FB&C (i.e., FB&C+, B-LCM+, and D-FB&C+). Notice all the memory compression techniques proposed here do not change the 3D-HEVC processing flow. Our solutions only reduce the *WMem* size maintaining 0% of BD-rate; consequently, minimizing the required bandwidth between the encoder/decoder and the *WMem*.

Table 5.9 – Number of bits required to store all wedgelets.

Work	Algorithm	4×4	8×8	16×16	Total	BD-rate
	Standard	1,376	51,328	130,560	183,264	0.00%
[34]	**Down-sampling**	0	0	130,560	130,560	0.05%
[63]	**No refinement**	928	20,096	98,304	119,328	0.25%
This Thesis	**Huffman**	991	23,503	34,298	58,792	0.00%
	BCM	761	14,428	27,175	42,364	
	LCM	1,086	22,301	27,108	50,495	
	FB&C	1,032	25,664	40,800	67,496	
	B-LCM	865	19,944	25,259	46,068	
	D-FB&C	946	22,456	35,190	58,592	
	FB&C+	825	17,256	24,225	42,306	
	B-LCM+	755	17,107	21,700	39,562	
	D-FB&C+	808	16,150	21,930	38,888	

Figure 5.19 shows the impact of the presented compression solutions using the standard Intra_Wedge as reference. In the down-sampling solution proposed in [34], only the total compression rate is presented since it removes the memory area required for storing 4×4 and 8×8 wedgelet patterns, maintaining all wedgelet patterns for 16×16 blocks. B-LMC improves the LCM algorithm increasing more than 2% the compression rate. D-FB&C provides almost 5% of gains when compared to FB&C because of its vertical-aware propriety. The ERR technique applied to FB&C, B-LCM and D-FB&C increases 9.3% the compression rate, on average. Besides, D-FB&C+ reaches a total memory reduction of 78.8% compared to the standard approach; thus, D-FB&C+ is the best compression result among all the solutions proposed here and, also among all the published works.

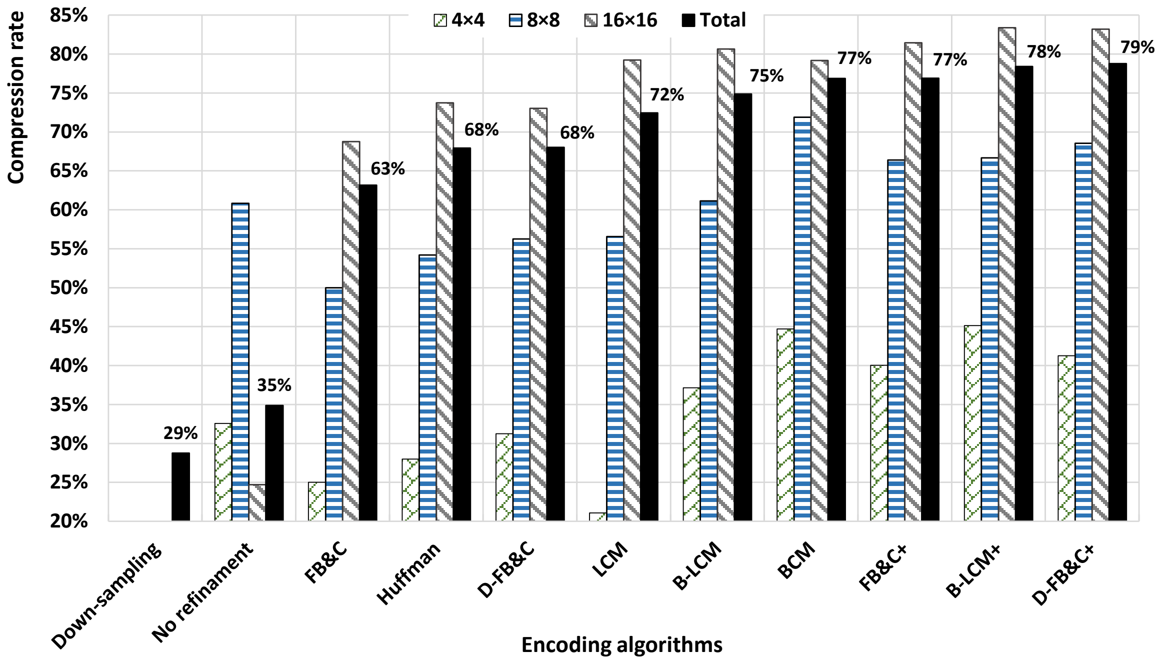


Figure 5.19 – Compression rates for all stored patterns of all proposed technique (source: [64]).

5.3.9 DFB&C+ Hardware Implementation

We chose to implement the D-FB&C+ algorithm in hardware because it is the most efficient technique for the wedgelet compression, as presented in the last subsection. D-FB&C+ can be implemented with lightweight hardware due to the low complexity of this algorithm. The D-FB&C+ coded wedgelet are generated in an off-line process, as explained previously. These codes are stored in the *WMem* and remain the same during the system execution. Therefore, any Intra_Wedge encoder or decoder designed in hardware requires only an additional circuit to support the D-FB&C+ decoding, since when the Intra_Wedge encoder or decoder requires a wedgelet from *WMem*, the coded wedgelet must be decoded to be used.

The standard *WMem* contains all wedgelets in the uncompressed format defined by the 3D-HEVC standard. The Intra_Wedge encoder or decoder can directly assess these wedgelets. The solution proposed here defines the wedgelets are stored in a compressed way inside *WMem*. Then, extra hardware (the D-FB&C+ decoder) is necessary to interface the Intra_Wedge encoder and decoder and *WMem*. The D-FB&C+ decoder reduces the *WMem* area and decreases the memory communication since the flow between the memory and the encoder/decoder is composed of compressed wedgelets. Notice that both the Intra_Wedge encoder and decoder can share a single circuit containing the decoder of the compressed data (i.e., D-FB&C+ decoder) if they are inside the same integrated circuit.

We designed a D-FB&C+ decoder architecture using VHDL to evaluate the area consumption and power dissipation, considering the Intra_Wedge encoder scenario. Figure 5.20 illustrates the block diagram of the proposed architecture.

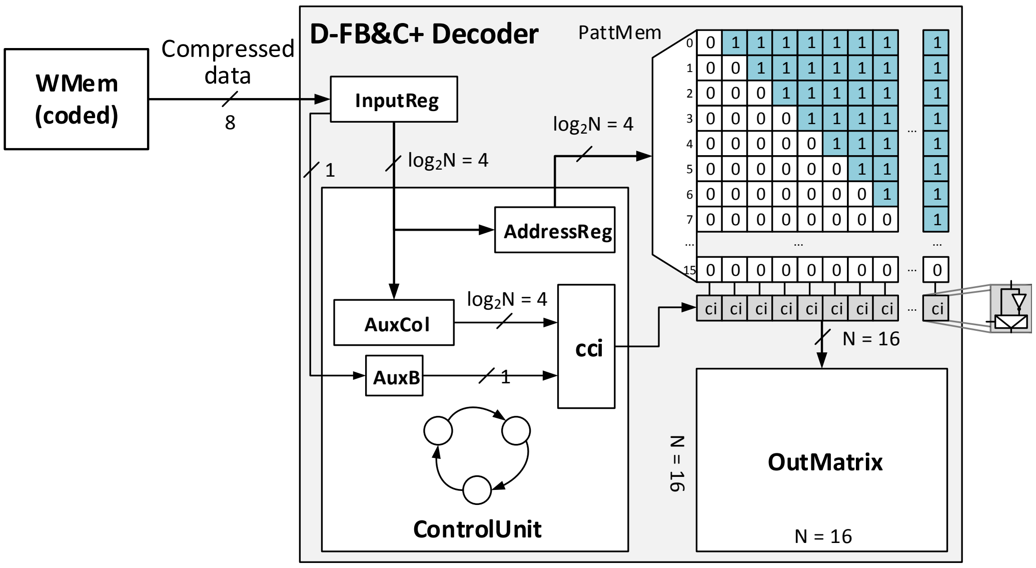


Figure 5.20 – Partial block diagram of the D-FB&C+ decoder architecture (source: [64]).

The D-FB&C+ algorithm stores the wedgelet pattern in an irregular format. The first bit of the D-FB&C+ format is the top left corner of the wedgelet pattern block, and the following bits are grouped into sets of $log2(N) - bit$ (*ICodes*) to code bit-changes in the rows and columns of the block. Notice the *ICode* size depends on the block size is being decoded; for instance, *ICode* is a 3-bit size for 8×8 blocks since N is 8 and $log2(N)$ is 3. Therefore, this section exemplifies the D-FB&C+ decoder considering 8×8 blocks.

The D-FB&C+ decoder has a one-byte register (*InputReg*) that receives 8 bits from the coded *WMem*. Due to the D-FB&C+ irregular format, in the first reading, *InputReg* stores the first coded bit of the block, two *ICodes*, plus one bit that is part of the third *ICode*. The first *ICode* indicates the line of the first column where the bit-change occurs and the second *ICode* indicates the position of the bit-change in the first row. This last bit of *InputReg* is not handled in this first reading since the third *lCode* is incomplete. However, this remaining bit together with the next byte reading allows the construction of three more *lCodes*.

The *ControlUnit* circuit controls the bit stream read from the coded *WMem* to keep updated the information being received and check if a re-reading of *WMem* is required to complete the wedgelet pattern decoding. Thus, the reading of the coded *WMem* follows as long as there are wedgelet patterns that are read by the D-FB&C+ decoder.

Decoding the pattern requires an output matrix (*OutMatrix*) that must have the size of the block being decompressed. In practice, this matrix must have the largest block size to be decompressed (i.e., 16 $\times$ 16), since smaller blocks are stored within this same area.

ControlUnit copies the first bit of *InputReg* into a one-bit auxiliary register (*AuxB*) and the next *lCode* into an auxiliary register (*AuxCol*) that stores the number of the row where occurs the bit-change in the first column. All subsequent *lCodes* encode the bit-change position in each row of the block (*rowBlock*); thus, *ControlUnit* copies this data into a register (*AddressReg*) that address a pattern memory (*PattMem*) containing the wedgelet pattern of a row. Similarly to the coded *WMem*, *PattMem* is also filled only once during the design time.

The output row of *PattMem* passes by a vector of controlled inverters (*ci*) that maintain the pattern or invert all bits. The vector of *cis* is managed by a circuit (*cci*) that reads *AuxB* and *AuxCol*, and according to both information decides if the patterns have to be inverted. For instance, if *AuxB* = 0 and *AuxCol* $>$ *rowBlock*, the pattern is preserved; however, if *AuxB* = 1 and *AuxCol* $>$ *rowBlock*, the pattern is inverted.

All patterns passing by the vector of *cis* are stored into *OutMatrix*, and the process of reading new *lCodes* is repeated until the pointer of the *OutMatrix* row is smaller than the height of the block is being decoded.

The D-FB&C+ decoder architecture was synthesized using standard cells ST 65 nm CMOS technology (logical synthesis), and the area consumption and power dissipation of *WMem* were estimated using CACTI [28]. The parameters used in the synthesis were focused on achieving real-time processing in the Intra_Wedge main stage. Table 5.10 displays the area consumption and power dissipation results and compares the proposed solution to the standard approach.

Table 5.10 – D-FB&C+ synthesis results compared to the standard approach.

Algorithm	**Resource**	**Area (µm2)**				**Power (mW)**			
		4 $\times$ 4	**8 $\times$ 8**	**16 $\times$ 16**	**32 $\times$ 32**	**4 $\times$ 4**	**8 $\times$ 8**	**16 $\times$ 16**	**32 $\times$ 32**
Standard	**Memory**	1,523	30,118	68,378	68,378	4.2	217.3	325.5	81.4
D-FB&C+	**Memory**	1,021	8,496	11,833	11,833	1.8	40.1	38.5	9.6
	Hardware	710	2,617	7,256	6,526	2.3	13.4	34.7	16.8
	Total	1,731	11,113	19,089	18,359	4.1	53.5	73.2	26.4
Gain		-13.7%	63.1%	72.1%	73.2%	2.4%	75.4%	77.5%	67.6%

Firstly, notice the *WMem* area required for storage 16×16 and 32×32 blocks is the same because the wedgelet patterns for 32×32 blocks are obtained from the same memory information employed on 16×16 blocks, and it is just required to upscale the patterns. Moreover, the circuits employed on encoding 32×32 blocks dissipate less power than 16×16 blocks because there are four times less 32×32 blocks in a frame than 16×16 blocks and both require the same computation effort.

Regarding 4×4 blocks, our proposal reduces the dissipated power by only 2.4% when compared to the standard approach, because the power dissipation of the decoder is considerably high. Additionally, the hardware of the D-FB&C+ decoder has an extra area overhead implying our solution consumes 13.7% more area than the standard approach.

Regarding blocks greater or equal to 8×8, the gains of the D-FB&C+ solution are expressive reducing between 63.1% and 73.2% the area consumption and between 67.6% and 77.5% the power dissipation, even considering the hardware overhead of the D-FB&C+ decoder. Considering the coding of all block sizes, the D-FB&C+ approach reduces the power dissipation from 628.4 mW to 157.2 mW when compared to the standard approach, which represents a reduction of almost 75%.

5.4 Bipartition Mode Hardware Design

Figure 5.21(a) and (b) exemplify the decoding of two 4×4 blocks, one encoded by Intra_Wedge and the other encoded by Intra_Contour, respectively. For a given block size, the input of the Intra_Wedge decoding algorithm requires the number of the selected pattern, the CPV of each region, and the residual block. An Intra_Wedge decoder requires accessing the wedgelet memory for reading the wedgelet pattern. Then, the CPV of each region is mapped into this Intra_Wedge pattern to reconstruct the predicted block. Finally, the decoder adds the residues to the predicted block, generating the reconstructed depth block.

The Intra_Contour decoder requires the residual block, CPV, and the correlated texture block. The main difference between Intra_Wedge and Intra_Contour decoding is the first mode obtains its pattern by accessing a memory with a pre-defined set of patterns, while the second mode rebuilds the pattern dynamically doing the same operation performed in the Intra_Contour encoder. To rebuild the pattern, Intra_Contour firstly computes the average of the four corner texture samples (*AVG_TEXT*); subsequently, it compares each texture sample with *AVG_TEXT* and sets samples above this average to one region and the remaining samples to the other region.

After building the pattern, Intra_Contour reconstructs the block similarly to the Intra_Wedge process, mapping the CPVs according to the Intra_Contour pattern for reconstructing the predicted block and adding the residual block to the predicted one, obtaining

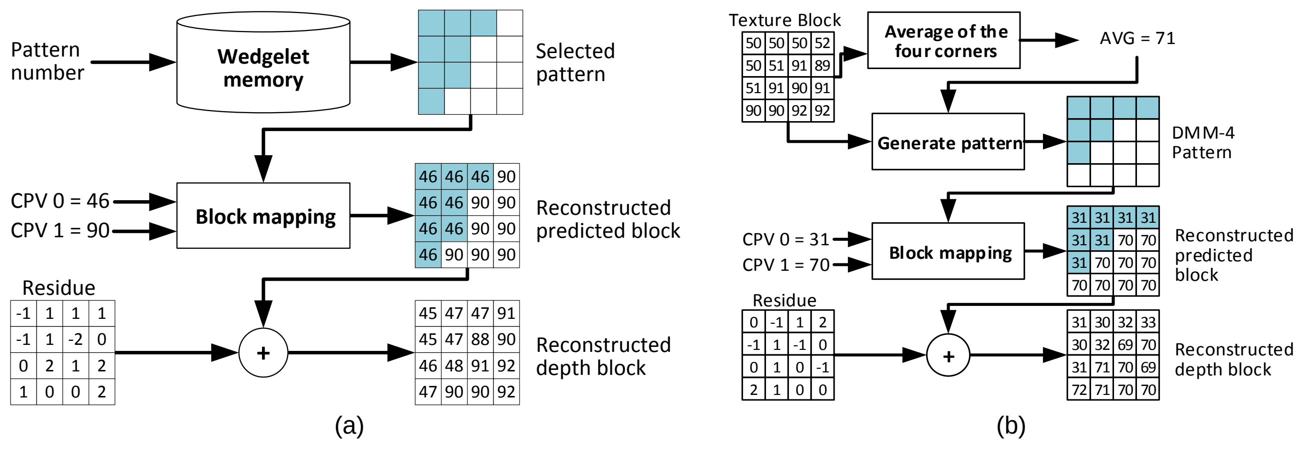


Figure 5.21 – Example of a 4 $\times$ 4 depth block decoding with Intra_Wedge (source: [61]).

the final reconstructed block. When comparing the Intra_Wedge decoding to the Intra_Contour decoding, it is clear this last requires more operations, because it rebuilds the pattern dynamically, while the Intra_Wedge decoding obtains the pattern by performing one read in the wedgelet memory.

The encoder transmit only the delta CPV to reduce the size of the bitstream. Thus, before the Intra_Wedge and Intra_Contour decoding, it is necessary to reconstruct the CPV by computing the CPV prediction and adding to the delta CPV prediction. The architecture designed here reconstructs the CPV information before decoding the bipartition modes.

Figure 5.22 displays the high-level block diagram of the decoder architecture designed for the bipartition modes containing six modules: (i) CPV Reconstruction, (ii) Register bank, (iii) Block Reconstruction, (iv) Wedgelet Decision, (v) Wedgelet Memories, and (vi) Control. We designed the decoder requiring at most 36 bits per cycle as input (four 9-bits data) to create a low area and low energy consumption architecture. Consequently, this design reduces the communication with the main memory of the decoder, increasing the total decoder efficiency.

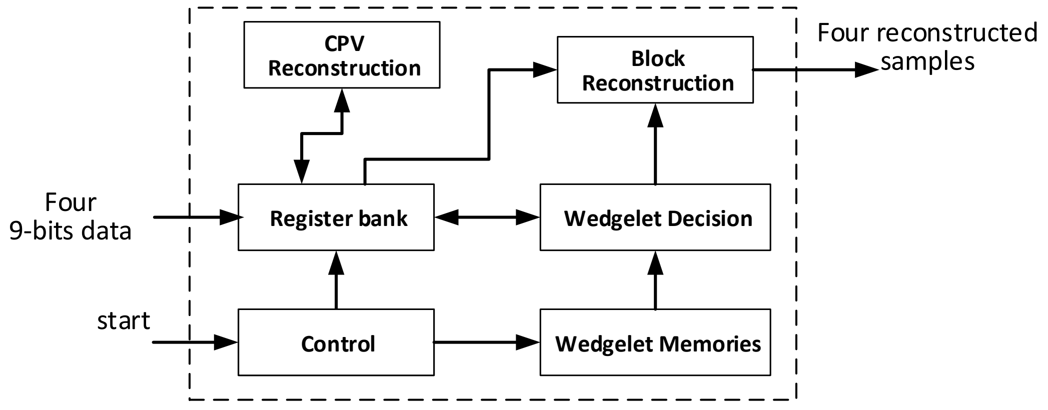


Figure 5.22 – High-level block diagram of the Intra_Wedge decoder (source: [61]).

Figure 5.23 displays both the architecture of the Intra_Wedge and Intra_Contour decoding flows. Initially, the global decoder requires decoding a block encoded by the bipartition mode signalizing the architecture to start processing. In the next cycle, the decoder receives five data among the input information, namely: DeltaCPV_0, DeltaCPV_1, Block_size, Mode, and Pattern_num. Notice the input of the architecture can receive four chunks of 9-bits data. However, Block_size and Mode require few bits allowing integrating both in a single chunk of 9-bits. Besides, Pattern_num is only used in case of Intra_Wedge being selected, and discarded when the Intra_Contour is selected, without being stored in the Register Bank. According to the Mode value, the block follows Intra_Wedge or Intra_Contour decoding.

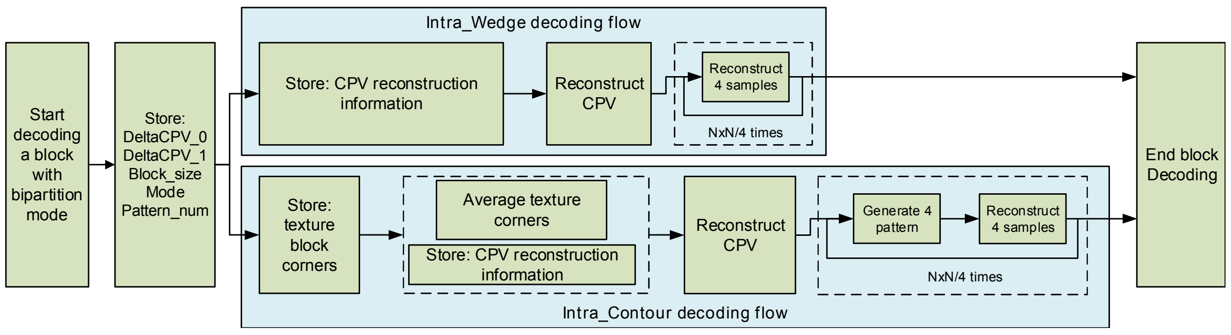

Start decoding a block with bipartition mode
Store: DeltaCPV_0 DeltaCPV_1 Block_size Mode Pattern_num
Intra_Wedge decoding flow
Store: CPV reconstruction information
Reconstruct CPV
Reconstruct 4 samples
NxN/4 times
Store: texture block corners
Average texture corners
Store: CPV reconstruction information
Reconstruct CPV
Generate 4 pattern
Reconstruct 4 samples
NxN/4 times
Intra_Contour decoding flow
End block Decoding

Figure 5.23 – Diagram of the architecture decoding flow for both bipartition modes (source: [61]).

Following the Intra_Wedge flow, in the next two cycles, the architecture stores information of neighbor blocks required to reconstruct CPV and accesses the wedgelet memory for reading the corners information of the Intra_Wedge pattern. In the subsequent cycle, the CPV Reconstruction module rebuilds the original CPV, generating all information required to decode the Intra_Wedge block. Afterward, the next cycles are used to recreate the encoding block, four samples per cycle using the Block Reconstruction module, which requires $N \times N/4$ cycles to rebuild the entire block. Finally, after completing the block reconstruction, the architecture signalizes the Intra_Wedge decoding is finished.

Following the Intra_Contour flow, in the next cycle, the architecture stores the four corners samples of the associated texture block. In the subsequent two cycles, while the architecture receives the neighbor samples required to rebuild the CPVs, the texture corner samples are averaged using the Block Reconstruction module, since it already contains adders and, therefore, it is necessary to insert only one more additional adder instead of three adders (see Figure 5.24). Additionally, the CPV Reconstruction module generates the binary pattern of the corner samples required for recreating the CPV value. Then, in the next cycle, the architecture rebuilds the original CPV value in the CPV Reconstruction module followed by a two-cycle operation that generates the four binary patterns, rebuilding the four samples. This two-cycle operation is repeated $N \times N/4$ times for reconstructing the entire block. Finally, the decoder signalizes the Intra_Contour decoding is completed.

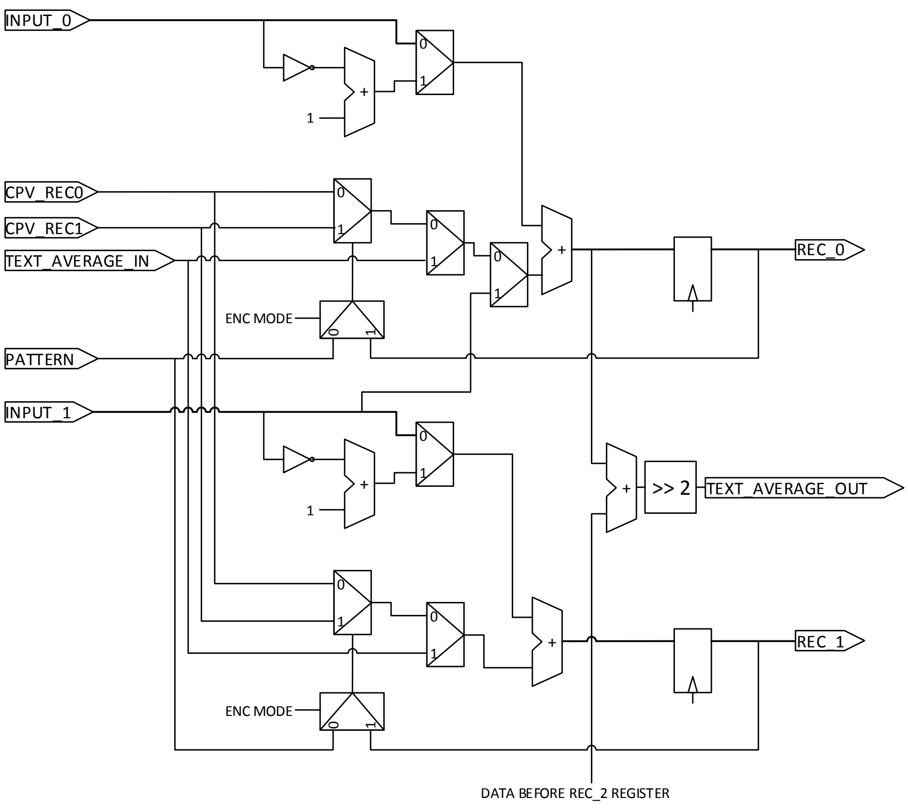


Figure 5.24 – Diagram of the block reconstruction for the first two samples. REC_2 and REC_3 diagram is similar to REC_0 and REC_1, respectively (source: [61]).

Notice that several operations performed in Intra_Wedge and Intra_Contour decoding are similar, enabling a hardware design with lower area requirements than implementing two independent architectures, one for each mode.

Figure 5.24 presents the Block Reconstruction module for the first two reconstruction samples (REC_0 and REC_1). Since REC_2 and REC_0 are produced similarly, and REC_3 and REC_1 are also generated similarly, the diagram omits them. According to the cycle, the INPUT value is a texture sample or the depth residue. In the texture average computation, the four INPUT information are the texture corner samples that are added and shifted right 2 bits (i.e., divided by 4), generating the TEXT_AVERAGE_OUT information that is stored in the register bank. When making the four Intra_Contour binary pattern information, the INPUT receives four texture samples, which are subtracted from TEXT_-AVERAGE_IN (that comes from the register bank). The subtraction is stored in the register at the right side of the architecture, and the information is loopback for the next cycle to select if CPV_REC0 or CPV_REC1 is added with the INPUT value that contains a residue, outputting the reconstructed sample.

The Intra_Wedge process is simpler than the Intra_Contour one, as the PATTERN is read from the wedgelet memory and decoded in the wedgelet decision (because it was stored using FB&C). Then, the pattern is used to select the CPV_REC to be added to the residue delivered to this module in the INPUT variable.

Figure 5.25 depicts the block diagram of the DC Reconstruction architecture that implements the delta CPV algorithm. All data required in this process (i.e., P0, P1, P2, DELTA_DC0, DELTA_DC1, TL, TM, TR, TRR, LT, LM, LB, LBB) are previously stored in the decoder Register bank. This module takes one clock cycle to generate DC_REC0 and DC_REC1, representing the first DC computed by the encoding process.

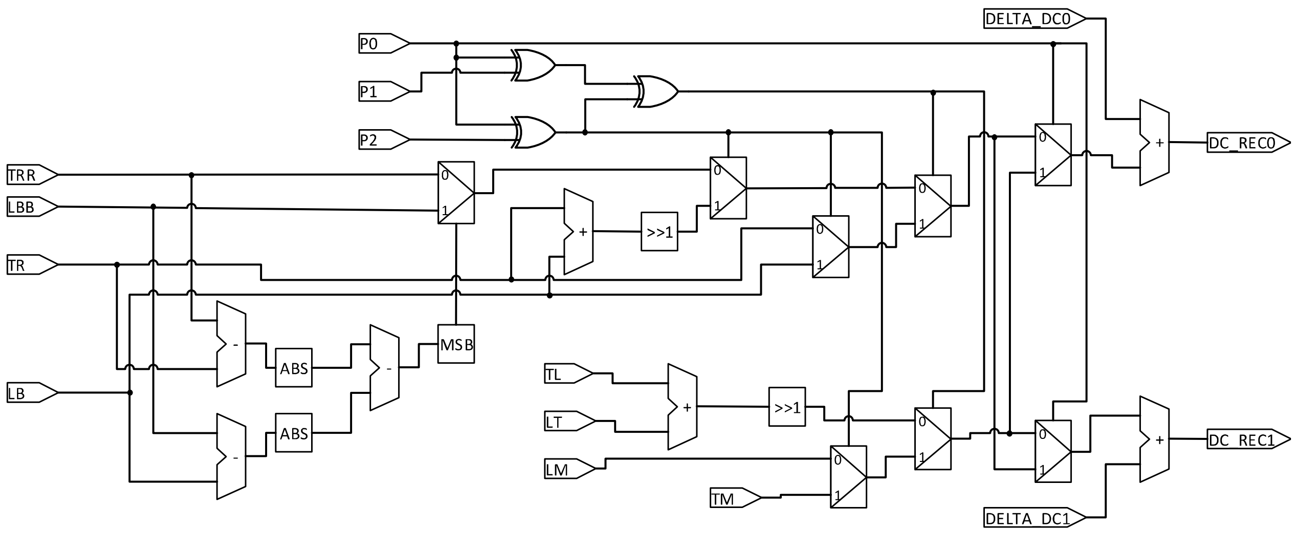


Figure 5.25 – DC Reconstruction architecture (source: [61]).

5.4.1 Synthesis Results

We synthesized three versions of this architecture: (i) a Intra_Wedge only architecture with the traditional wedgelet storing approach (i.e., storing all binary information); (ii) a Intra_Wedge only architecture using FB&C to achieve a considerable memory reduction; and (iii) a bipartition mode architecture using FB&C, which was selected in this evaluation due to its simplicity to be integrated with the decoder hardware design. Table 5.11 shows the logical synthesis results for standard cells ST 65nm CMOS technology.

Table 5.11 – Logical synthesis results of the bipartition modes decoder.

Solution	(i)	(ii)	(iii)
Implemented algorithms	Intra_Wedge	Intra_Wedge (FB&C)	Intra_Wedge (FB&C) & Intra_Contour
Area (gates)	4,533	4,047	5,165
Decoding block sizes	All	All	All
Frequency (MHz)	38.9	38.9	58.3
Cycles per block – Intra_Wedge	10/22/70/262	10/22/70/262	10/22/70/262
Cycles per block – Intra_Contour	-	-	15/39/135/519
Processing rate HD 1080p fps	30.0	30.0	30.0
Power (mW)	1.34	0.95	1.57

The designed architectures were described in VHDL and synthesized for standard cells ST 65nm CMOS technology. The frequency used for synthesis considered the worst case for the proposed architectures; i.e., in cases (i) and (ii) all blocks were selected with size 4×4 using Intra_Wedge, and in case (iii) all blocks were selected with size 4×4 and using the Intra_Contour mode, since 4×4 would produce the worse processing rate. In all cases, the synthesis was performed to decode 1080p videos at 30 frames per second (fps). However, these assumptions overestimates the required area, processing rate, and power dissipation of the designed architecture, because (i) a real implementation requires decoding higher block sizes, which needs lower frequency, and (ii) the mode selection is shared with other modes, which allows maintaining this architecture idle with a significant power dissipation saving, for example.

The Intra_Wedge decoder with traditional memory approach requires 4,533 gates and dissipates 1.34 mW of power. When applying FB&C, the architecture area and power dissipation are reduced in 10.7% and 29.1%, respectively. This result demonstrates that reducing the wedgelet memory brings significant benefits for hardware design regarding area occupation and power dissipation as already demonstrated in Section 5.3.

In the bipartition mode architecture, even with this overestimated assumption, the area required by the proposed architecture is of only 5,165 gates with a power dissipation of 1.57 mW. Notice the bipartition mode architecture consumes 27.6% more area and dissipates 65.2% more power than that of the best version of Intra_Wedge architecture. However, the bipartition mode architecture contains two modes that can be decoded, and its synthesis results considered the worst case when Intra_Contour was always selected. As previously described, the Intra_Contour decoding implies more computational effort than the Intra_Wedge decoding because Intra_Contour performs several computations for rebuilding its pattern, unlike the memory read operation employed in Intra_Wedge. Besides, the designed architecture shares resources between Intra_Wedge and Intra_Contour due to the shared operations. As such, independent Intra_Wedge and Intra_Contour architectures probably would require more area than this design.

6. CONTRIBUTIONS ON INTER-FRAME PREDICTION ENCODING TIME REDUCTION

This chapter presents the thesis contributions related to the encoding time reduction of the inter-frame prediction. Three algorithms were designed here: (i) Edge-Aware Depth Motion Estimation (E-ADME) (Section 6.1) [68], (ii) an early termination block-level decision (Section 6.2) [47], and (iii) a quadtree limitation (Section 6.3) [49].

6.1 Edge-Aware Depth Motion Estimation (E-ADME)

Figure 6.1(a) and (b) show a slice of two consecutive frames from the depth maps of Shark video sequence (frames 2 and 3 of the view 9). Two detached blocks with 16×16 pixels in Figure 6.1 (b) were applied to the FS algorithm, whose search area of [-40, 41] are displayed in Figure 6.1(a) with the same color. The yellow box in Figure 6.1(b) starts in the pixel (240, 340) and the red box starts in the pixel (240, 440). Figure 6.1(c) and (d) show the SAD heat map for the ME process of a given encoding block in homogeneous (red box in Figure 6.1(b)) and edge regions (yellow box in Figure 6.1(b)), respectively. Dark blue regions denote the lowest SAD values, whereas red regions represent the higher SAD values, meaning the best and worst values, respectively.

Figure 6.1 shows when encoding a homogeneous region block, the heat map has smooth changes, with large regions containing low SAD values around the center of the search area. Thus, homogeneous regions in the depth maps can be encoded using center-biased (i.e., a procedure that starts searching in the center of the search area) lightweight fast ME algorithms, such as Iterative Small Diamond Search Pattern (I-SDSP) or Diamond Search (DS) [87], as described in the works [50] and [51], achieving near-optimal results with less SAD comparisons than TZS. However, the SAD analysis of the edge heat map (Figure 6.1(d)) shows higher pattern variability, requiring more sophisticated ME algorithms for providing higher quality on the synthesized views.

Therefore, designing a scheme able to identify if the encoding block contains an edge or homogeneous region can reduce the computational effort of the ME and also preserve the encoding efficiency. This scheme can perform a more sophisticated algorithm for edge regions or a simple algorithm for homogeneous regions.

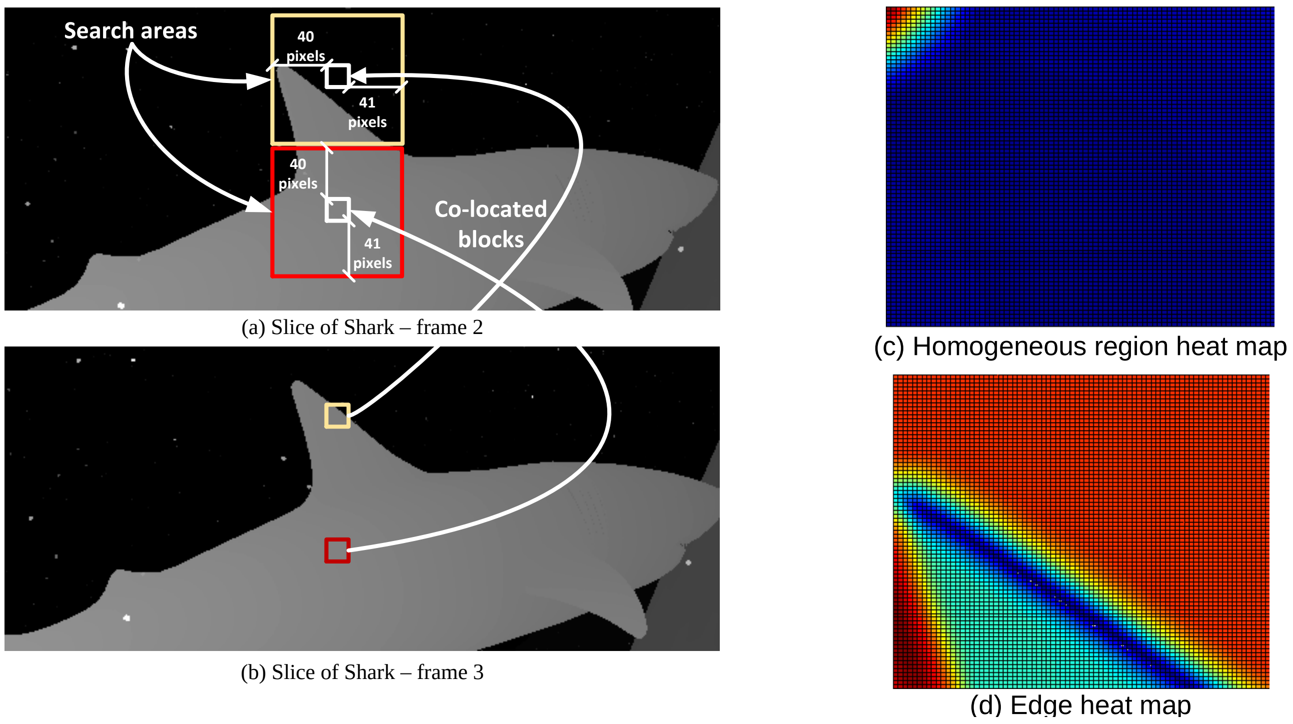


Figure 6.1 – Slices of the Shark video with (a) two detached search areas and (b) two detached encoding blocks. SAD heat map for (c) homogeneous and (d) edge regions (source: [68]).

6.1.1 Designed Scheme

Figure 6.2 illustrates the E-ADME scheme, which starts using the SED algorithm to identify if the encoding block contains an edge or a homogeneous region. When the encoding block is classified as homogeneous, the lightweight I-SDSP algorithm is applied due to its efficiency for this kind of scenario. I-SDSP can find low SAD values around the center-biased position; thus, accelerating the ME encoding flow. When SED classifies the encoding block as an edge, the conventional 3D-HEVC ME encoding flow is performed using the TZS algorithm, because it raises the probability of obtaining higher SAD values distant from the co-located block.

The work [50] shows using I-SDSP instead of TZS in ME of depth maps tends to obtain a good tradeoff between computational effort and encoding efficiency. However, the ME process over depth map blocks in edge regions is harder than for homogeneous regions because many candidate blocks with low SAD values are found distant from the center-biased block and, thus, I-SDSP can be inefficient. Considering these two aspects, the usage of TZS in edge blocks tends to obtain a sound tradeoff regarding encoding quality and computational effort. One can notice the presented solution does not add any computational effort or a memory access to the 3D-HEVC depth map inter-prediction.

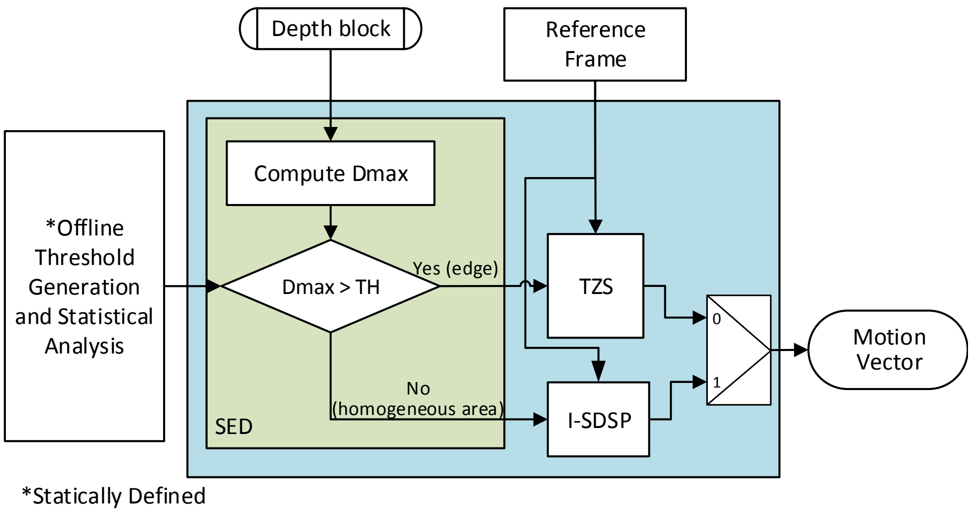


Figure 6.2 – Scheme proposed for reducing the E-ADME depth map complexity (source: [68]).

The SED algorithm used as a base in this scheme was described earlier in this thesis. The efficiency of the SED algorithm is dependent on the threshold defined statically according to the block size and the video resolution [67]. The encoding effort control system designed in Section 5.1 could also be used for controlling the thresholds here; however, their efficiency was not evaluated for inter-frame prediction. In [67], the thresholds (*TH*) for classifying a block as homogeneous or edges were determined regarding the block size. These thresholds were interpolated, generating equations 6.1 and 6.2 that shows the polynomials for 1024 $\times$ 768 and 1920 $\times$ 1088 video resolutions, respectively; *W* indicates the encoding block width and *TH* the resulting threshold.

$$TH = ceil(-0.0186 \times W^2 + 2.2 \times W + 3.5) \tag{6.1}$$

$$TH = ceil(-0.0038 \times W^2 + 0.74 \times W + 5.1) \tag{6.2}$$

Sanchez et al. [67] use only four blocks sizes: 4 $\times$ 4, 8 $\times$ 8, 16 $\times$ 16 and 32 $\times$ 32, while ME also requires encoding 64 $\times$ 64 blocks, asymmetrical blocks such as 64 $\times$ 32, 32 $\times$ 24, 16 $\times$ 12, and others. Thus, it is necessary to define new thresholds for those block sizes. The thresholds for the new block sizes were generated using the polynomial interpolation of Lagrange to make a second-degree polynomial equation, which computes the new thresholds based on the fixed thresholds described in [67] and the correspondent block width. The new thresholds were generated for blocks with the widths of 12, 24, and 64 samples.

6.2 Early Termination Block-Level Decision

Figure 6.3 presents experimental results for analyzing the 3D-HEVC encoder behavior in inter-frame prediction. These experiments were performed following the CTC (described in Section 2.5) considering Random Access (RA) encoder configuration. The first part of the experiment allows evaluating the Skip mode occurrence in texture encoding considering different QPs. Figure 6.3(a) shows the Skip mode has a high probability of occurrence in texture encoding for all cases evaluated, ranging from 27% (8×8 CUs and QP=40) to 95% (64×64 CUs and QP=40). Besides, for all evaluated QPs, more than 90% of the 64×64 CUs were encoded using the Skip mode.

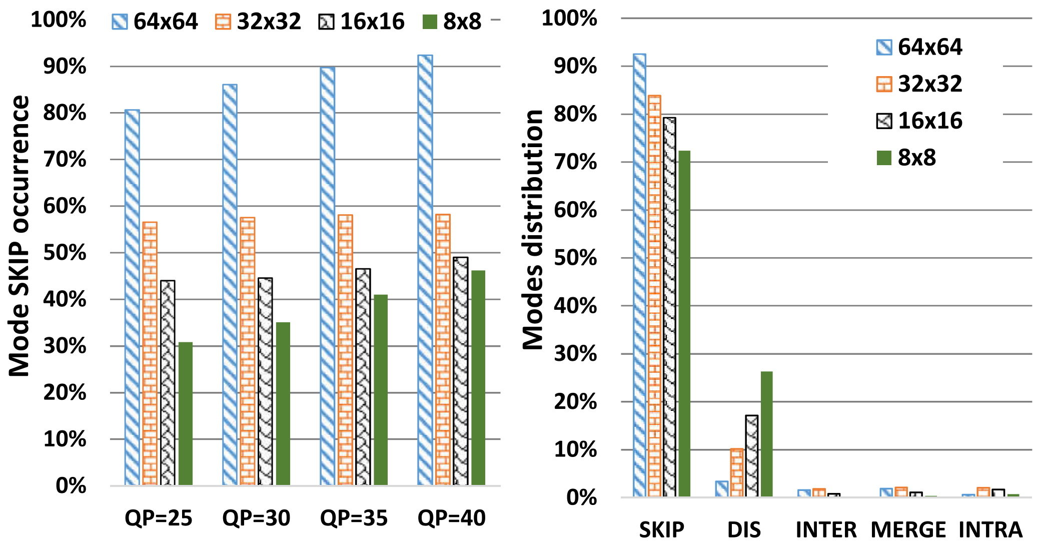


Figure 6.3 – (a) Occurrence of Skip mode for texture encoding and (b) modes distribution for depth map encoding when M_{text} = *Skip* (source: [47]).

The second part of the experiment verifies the depth map coding behavior when the correlated reference block in texture view is encoded with the Skip mode. For convenience, let M_{text} be the selected mode by the correlated reference block in texture view. Figure 6.3(b) presents the mode distribution for depth map encoding when M_{text} = *Skip*, showing a low probability, less than 4% (on average), of the modes INTER, MERGE or, INTRA being chosen for encoding a current depth map CU. On the other hand, 93% of 64×64 CUs and more than 80% (on average) of depth maps CUs are encoded using Skip mode. Besides, the Intra-picture skip mode is selected more than 15% (on average) for encoding the depth map CUs. Thus, when M_{text} = *Skip*, the probability of the encoding depth map CU be Skip or Intra-picture skip is more than 95%, on average.

Figure 6.4 presents the Probability Density Function (PDF) of Skip or Intra-picture skip mode (Skip/Intra-picture skip) being selected as the best encoding mode for the current CU, when M_{text} = *Skip*, according to the RD-cost (divided by CU width size) obtained by evaluating only these two modes. The division of RD-cost by the CU width size has been selected as an evaluation criterion since larger CU sizes tend to have larger RD-cost values to compute. The analysis presented in Figure 6.4 considers 64 $\times$ 64 CUs with the Undo_Dancer video sequence encoding under RA configuration with QP=30/39 (QPtexture/QPdepth). Similar results were obtained for the other CU sizes and QPs.

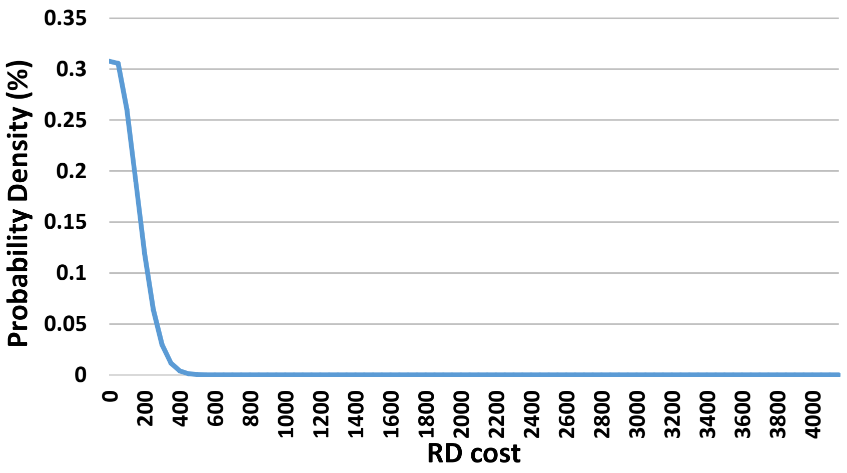


Figure 6.4 – PDF of encoding depth map CU with Skip or Intra-picture skip according to the RD-cost divided by the CU width size (source: [47]).

These results show a high probability of the current CU being encoded using Skip or Intra-picture skip mode for small values of RD-cost when M_{text} = *Skip*. On the other hand, for higher values of RD-cost, Skip and Intra-picture skip have no chance to be selected, since higher values of RD-cost mean a different encoding mode should encode the CU. Thus, when the correlated block in texture view is encoded with Skip mode (M_{text} = *Skip*), and the RD-cost obtained by Skip and Intra-picture skip has a low value, it is high the probability of selecting Skip as the best encoding mode. Hence, it can be used to perform an early termination decision according to a threshold criterion, avoiding the evaluations of the remaining encoding modes.

6.2.1 Design of the Early Termination Block-level Decision

Looking for a lightweight solution for avoiding the excessive evaluations of the encoding modes in the traditional 3D-HTM encoding flow, the early termination block-level decision scheme privileges the modes with a high probability of being selected, with low computational effort and using fewer bits to encode blocks. The flowchart of the proposed scheme is presented in Figure 6.5.

Based on our previous analysis, the scheme starts computing the RD-cost for Skip and Intra-picture skip modes for the given depth map CU. If the M_{text} = *Skip*, the minimum RD-cost between Skip and Intra-picture skip modes is computed. If the minimum RD-cost divided by CU width size (Wsize) is lower than the threshold (*TH*) defined by in an offline

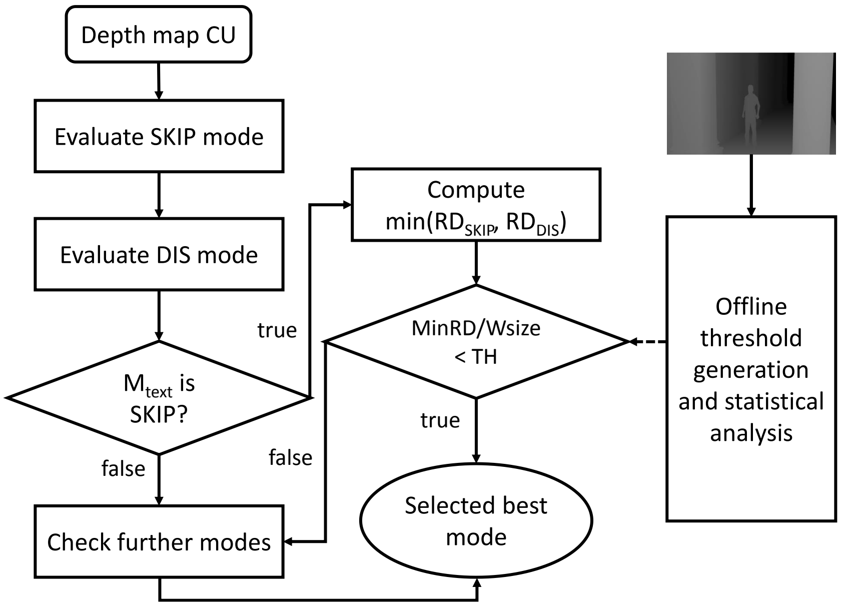


Figure 6.5 – Dataflow model for early termination scheme (source: [47]).

analysis, then the previous evaluation mode that obtained the lowest RD-cost is selected. Otherwise, further mode evaluations are required, and the encoding process follows without simplification, requiring evaluating INTER, MERGE and INTRA encoding modes seeking for better encoding results. Equation 6.3 summarizes the decision process.

$$\text{Decision} = \begin{cases} \mathit{lowestRD}(\mathit{DIS}, \text{SKIP}), & \frac{\text{MinRD}}{\text{Wsize}} < \text{TH}, \\ \mathit{lowestRD}(\mathit{DIS}, \text{SKIP}, \mathit{INTER}, \mathit{MERGE}, \mathit{INTRA}), & \frac{\text{MinRD}}{\text{Wsize}} \geq \text{TH} \end{cases} \quad (6.3)$$

In the best case, only Skip and Intra-picture skip are required to be evaluated in our scheme. However, in the worst case, the RD-cost calculated by our solution for that depth map CU is the same as the conventional 3D-HTM encoding flow, without inserting additional computational effort than the default approach would require.

The threshold values lead to light or aggressive solutions regarding both encoding time and video quality. Therefore, we employed an experimental analysis that evaluates some scenarios to explain the impact of the threshold variation.

We selected seven THs as evaluation targets, ranging from 100 to 400, with step 50. The corner values were selected by analyzing the PDF presented in Figure 6.4, while the step 50 was empirically selected to refine its results because smaller values would lead to a minimal variation. We evaluated 10 frames of the Undo_Dancer and GT_Fly 3D video sequences, which were selected randomly among the available sequences. Only two 3D video sequences were used in this analysis to avoid overfitting the designed scheme.

Figure 6.6 depicts the results of the threshold scenario evaluations, highlighting the percentage of early termination of encoding mode evaluation according to the well-accepted

BD-rate criterion. The percentage of early termination was computed as a division of the total cases our scheme skips the remaining evaluations by the total cases of M_{text} = *Skip*.

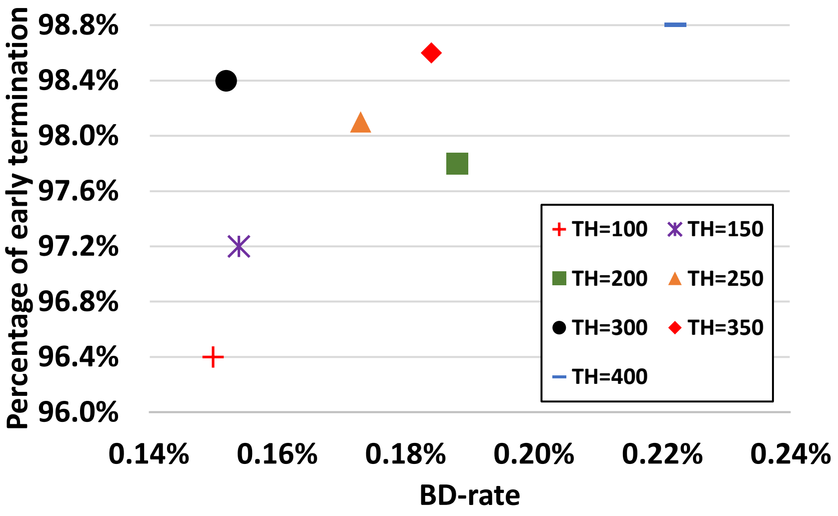


Figure 6.6 – Percentage of early termination of the encoding mode evaluation according to the BD-rate impact (source: [47]).

Figure 6.6 shows the evaluated THs provide some operation points considering encoding efficiency and with a high percentage of early termination in the traditional 3D-HTM encoding flow (more than 96% of all evaluated cases). Based on this analysis, *TH* = 300 was the best operating point for this early termination scheme.

6.3 Decision Trees for P- and B-Frames

The decision trees of the P- and B-frames definition process are similar to that used on the I-frames definition process. The features and attributes that could lead to effective I-frames splitting decisions were also evaluated for P- and B-frames. However, we assessed new parameters since there are many different features in P- and B-frames when compared to I-frames, such as inter-frame or inter-view dependencies. These new parameters are listed in Table 6.1, along with their description.

Figure 6.7 exemplifies the probability density functions regarding three of selected parameters, to show the correlation among the parameters and the CU splitting decision. Figure 6.7(a) shows the probability of the CU not be split according to the MAD_4 (MAD of its 4 $\times$ 4 sub-blocks); Figure 6.7(b) illustrates the probability of the CU not be split according to the Neigh_depth, and Figure 6.7(c) is the probability of the CU be split according to RelRatio. For the first two attributes (MAD_4 and Neigh_depth), low values tend to keep the encoding CU size. On the other hand, low values of the RelRatio tends to split the encoding CU. Therefore, the knowledge of these attributes is crucial for achieving efficient decisions on current CU splitting.

Table 6.1 – Parameters evaluated in P- and B-frames and their description.

Parameter	Description
RD_MSM	RD-cost of Merge/Skip Mode. It indicates the efficiency of using Merge/Skip mode; generally, high efficiency is obtained in slow-motion or homogeneous regions. In this case, bigger CUs are used.
RD_DIS	RD-cost obtained in Intra-picture skip mode; note the name of this mode can also be found in the literature as DIS. Using it, the decision tree can verifies if the current encoding CU using Intra-picture skip can be an attractive alternative to indicate if the current CU is composed of homogeneous regions.
Ratio	RD-cost obtained in intra-frame prediction divided by the RD_DIS. It allows comparing the efficiency of the whole inter-frame prediction and DIS encoding mode.
RatioInter	RD-cost of inter $2N \times 2N$ evaluation divided by RD_MSM, which can be used to identify if the encoding region is complex or contains a high movement intensity. In this case, smaller CUs tends to be selected [12].
RelRatio	The normalized difference between the RD-cost of inter $2N \times 2N$ and RD_MSM. This attribute is also used to identify if the region is complex or contains a high movement intensity such as RatioInter.
Neigh_depth	The average quadtree depth of the top, left, top-left, top-right, and co-located CTUs of both reference lists. This parameter is used because neighbor CUs tends to have a high correlation.
SKIP_flag	The binary flag notifies if the CU has been encoded using the Skip mode. In cases the CU is encoded with Skip, there is a high probability of not using smaller CUs.
DIS_flag	The binary flag notifies if the CU has been encoded using the DIS mode. In cases the CU is encoded with Intra-picture skip, there is a high probability of not using smaller CUs.
MAD	Maximum mean Absolute Deviation (MAD) of smaller blocks inside of the current CU (4×4 up to current CU size) - Allows identifying the dispersion of the values inside a given CU.

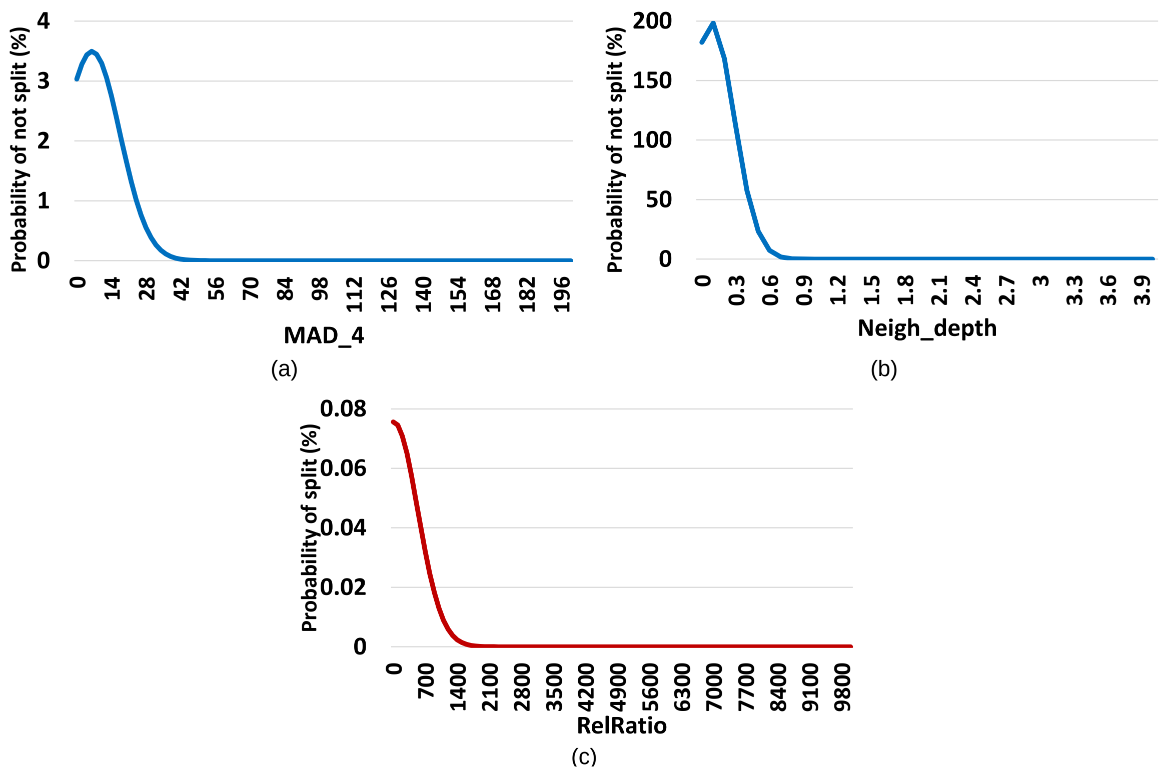


Figure 6.7 – Probability density function of the current 64×64 CU in RA configuration not be split regarding (a) MAD_4 and (b) Neigh_depth, and (c) be split regarding RelRatio (source: [49]).

Again, the experiment was trained using the Kendo video sequence. However, similar results were obtained repeating the same process with more video sequences or using a different video sequence. Table 6.2 illustrates the parameters used in the three decision trees for P- and B-frames. Table 6.2 also presents the IG of the attributes and the accuracy obtained by the constructed trees, where the designed trees for 16 $\times$ 16, 32 $\times$ 32 e 64 $\times$ 64 CUs obtained an accuracy of 82.13%, 83.08% e 91.65%, respectively.

Table 6.2 – Parameters used in P- and B-frames decision trees.

Attributes	**64 × 64**		**32 × 32**		**16 × 16**	
	Used	**IG**	**Used**	**IG**	**Used**	**IG**
QP-depth		-	×	0.032	×	0.022
R-D cost	×	0.194	×	0.223	×	0.216
RD_MSM	×	0.198	×	0.182	×	0.154
RD_DIS		-	×	0.214		-
Ratio	×	0.450	×	0.258	×	0.228
RatioInter		-	×	0.194		-
RelRatio	×	0.483	×	0.194	×	0.139
Neigh_depth	×	0.493		-		-
SKIP_flag	×	0.257	×	0.068	×	0.050
DIS_flag		-	×	0.063		-
VAR	×	0.269	×	0.209		-
VAR_16	×	0.274		-		-
MAD_4	×	0.271		-		-
MaxDiff	×	0.278		-		-
Accuracy	**91.65%**		**83.08%**		**82.13%**	

Figure 6.8 shows the decision tree built by WEKA for 64 $\times$ 64 CUs. The built 32 $\times$ 32 and 16 $\times$ 16 CUs decision trees (that are omitted in this thesis) required six decision levels.

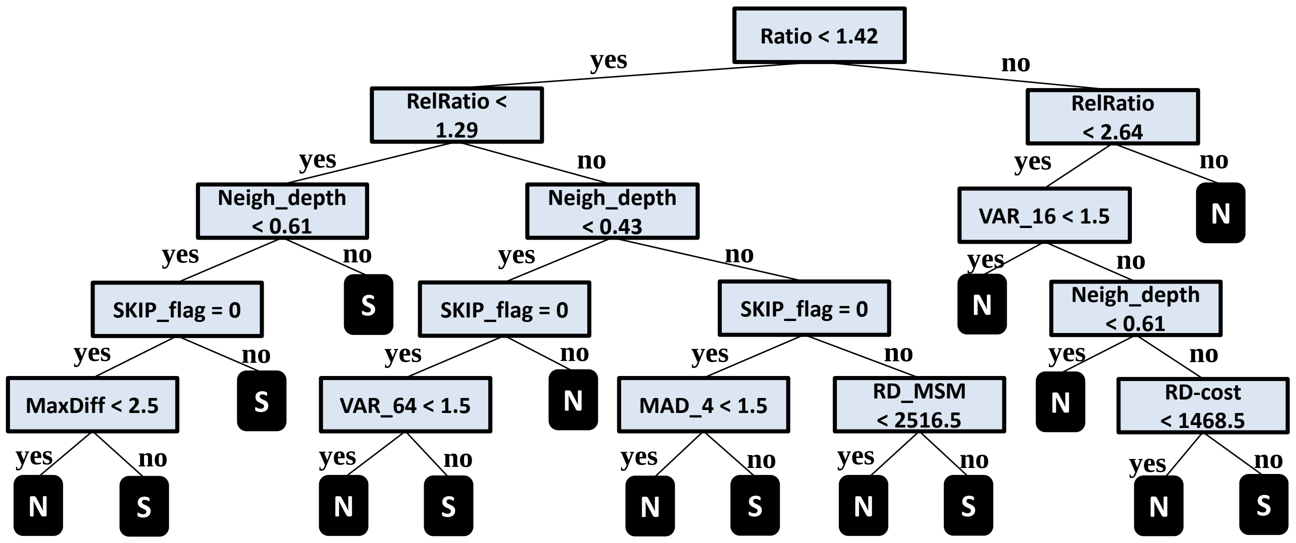


Figure 6.8 – Decision tree for splitting decision in 64 $\times$ 64 CUs for P- and B-frames (source: [49]).

6.4 Results and Comparisons

These three algorithms have been inserted into 3D-HTM and evaluated under CTC in RA configuration. Table 6.3 illustrates the results obtained with these algorithms. The performance difference in different videos execution occurs because each video posses different characteristics as highlighted in Section 2.5.

Table 6.3 – Evaluations in the random access scenario.

Resolution	Videos	E-ADME		Early termination Block-level decision		Decision tree with machine learning	
		Synthesis only BD-rate	Timesaving	Synthesis only BD-rate	Timesaving	Synthesis only BD-rate	Timesaving
1024 × 768	Balloons	0.0206%	3.6%	0.2084%	12.3%	0.0582%	47.7%
	Kendo	0.0246%	3.4%	0.2948%	11.1%	0.0267%	47.1%
	Newspaper_cc	0.1057%	3.7%	0.2620%	11.8%	0.0531%	48.1%
	Average	0.0503%	3.6%	0.2551%	11.7%	0.0460%	47.6%
1920 × 1088	GT_Fly	0.0948%	3.3%	0.0265%	14.8%	0.0747%	52.2%
	Poznan_Hall2	0.2900%	3.2%	0.3863%	15.2%	0.3921%	59.4%
	Poznan_Street	0.0713%	3.6%	0.1452%	15.8%	0.1017%	57.0%
	Undo_Dancer	0.2036%	2.3%	0.1327%	14.0%	0.5534%	53.3%
	Shark	0.3739%	2.9%	0.1252%	14.2%	0.0103%	51.4%
	Average	0.2067%	3.1%	0.1632%	14.8%	0.2264%	54.7%
Average		0.1481%	3.2%	0.1976%	13.6%	0.1588%	52.0%

The E-ADME algorithm reduces 3.2% the encoding time considering texture and depth map execution time with an increase of 0.1481% in the BD-rate; although it is not a significant reduction, if only ME/DE complexity is considered the 3D-HTM encoding time is reduced significantly as shown in Figure 6.9, the SAD computation is reduced 68.2%, on average.

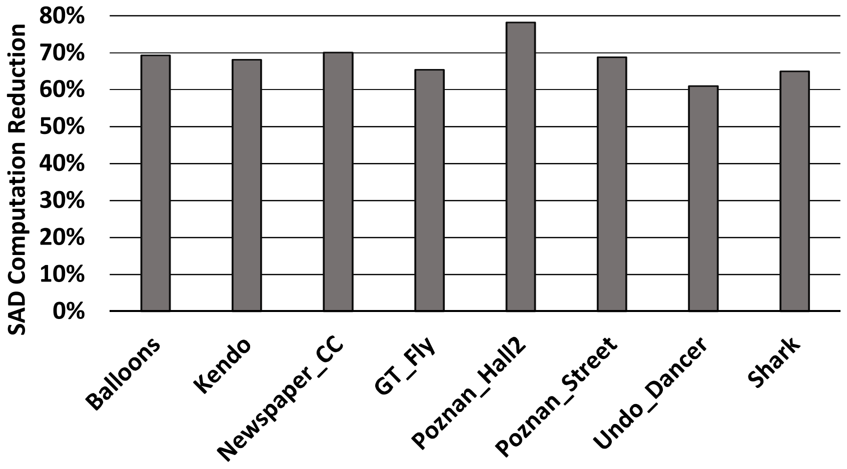


Figure 6.9 – SAD computation reduction obtained using E-ADME (source: [68]).

Comparing E-ADME to the other proposed techniques presented in Table 6.3, its results may seem worse than the others solutions; however, the BD-rate increase is still small and also, when designing real-time hardware it may be interesting to limit the ME/DE complexity by introducing this kind of technique. This reduction in SAD computations reduces the necessary accesses to the memory with reference samples and then, contributing to reduce the power dissipation. Consequently, the E-ADME scheme contributes to the development of real-time encoding systems with negligible impact on the encoding efficiency.

Furthermore, one of the most significant problems in the ME algorithms relies on the fact that it has a high I/O communication with the reference memory. This I/O communication is strictly related to fetching information from previous reference frames to use in its SAD computation. Therefore, reducing SAD computation in 68.2% tends to reduce the memory access in the same proportion. Since the encoder bottleneck is the I/O communication with the memory [78], the proposed scheme tends to increase the performance of the entire system when implemented into dedicated hardware design.

The block-level early termination decision reduces 13.6% the encoding time considering the texture and depth execution time, increasing 0.1976% the BD-rate, with some variations around it. These variations happen due to the differences among the encoding sequences, and higher resolution video sequences tend to select more Skip than other modes in the texture coding. Figure 6.10 shows the significant timesaving results are obtained because our scheme can evaluate only Skip and Intra-picture skip in 96.33% (on average) of the cases when M_{text} = *Skip*.

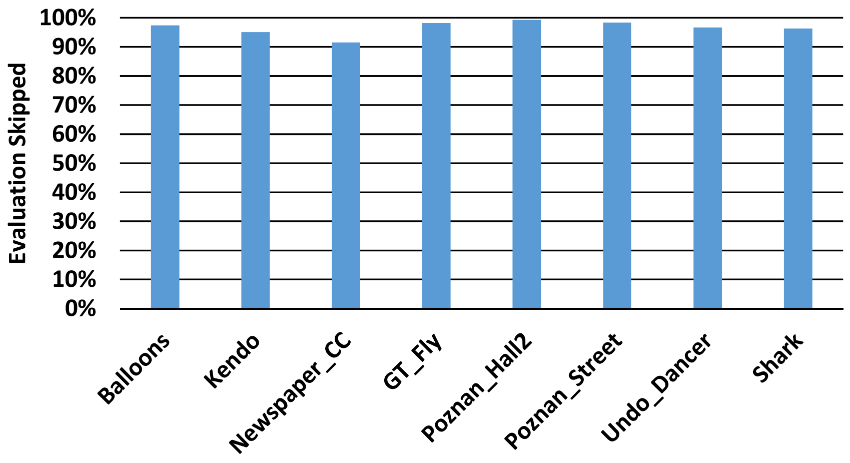


Figure 6.10 – Evaluation skipped in block-level early termination scheme (source: [47]).

This result is better than the one achieved with E-ADME when considering only this tradeoff; however, its implementation in a real-time system is harder since it contains dependencies among encoding modules. Besides, this scheme reduces 27.7% the depth map encoding time. These results are explained by showing when this technique skips the remaining encoding modules (more than 96% of evaluations are skipped, on average).

The last algorithm using a quadtree limitation based on machine learning reduced 52.0% the encoding time considering texture and depth map encoding time, with a drawback of increasing only 0.1588% the BD-rate in synthesized views. This was our best result obtained with RA when considering only the tradeoff between encoding timesaving and BD-rate. One drawback of this technique is the impossibility of encoding multiple block sizes together since the decision tree needs a given level to be encoded before encoding or skipping a lower quadtree level. Figure 6.11 shows the percentage of CUs were not split according to the CU size and QP-depth value when evaluated in RA configuration. The higher is the QP-depth value, the higher is the use of larger CUs and, consequently, the higher is the CUs with not split decisions. The highest not splitting percentage reaches 97.7% with QP=45 when processing 64 $\times$ 64 CUs, explaining the high encoding time reduction. When only depth map coding is considered, this solution reduces 68.0% the encoding time.

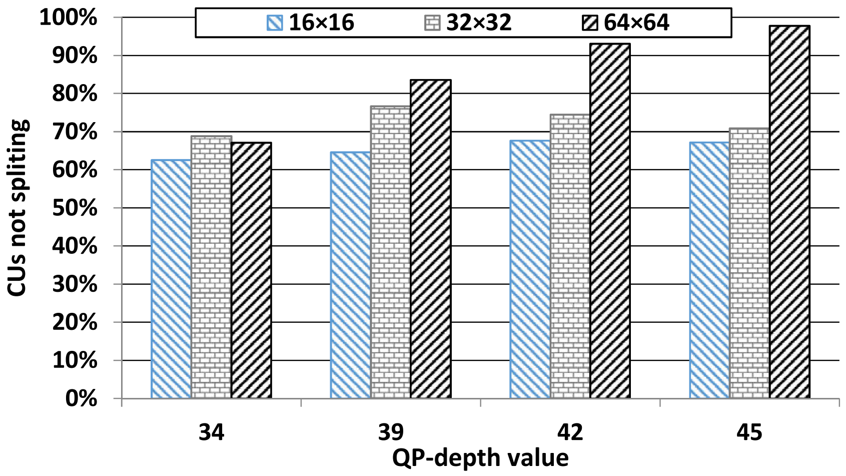


Figure 6.11 – Percentage of not splitting CUs using the designed quadtree limitation (source: [49]).

Table 6.4 compares our results to the related work.

Table 6.4 – Comparison with related works in random access scenario.

Work	Decision level	Synthesis only BD-rate	Encoding timesaving
E-ADME	Mode	0.1481%	3.2%
Early termination Block-level decision	Block	0.1976%	13.6%
Quadtree limitation using machine learning	Quadtree	0.1588%	52.0%
V. Afonso et al. [2]	Mode	0.3123% - 0.4803%	32.7% - 61.8% (reduction in SAD computation)
R. Conceição et al. [10]	Block	0.062% - 0.409%	9.6% - 13.8%
J. Lei et al. [32]	Quadtree	2.0600%	41.5%
E. G. Mora et al. [38]	Quadtree	1.0240%	52.0%

The E-ADME algorithm can be fairly compared only with [2], which is focused on reducing the DE encoding effort. It does not present their timesaving results, but only the reductions in SAD computation. Besides, it is only focused on DE, while our work is focused on ME and DE. It obtained a BD-rate increase ranging between 0.3123% and 0.4803%, with a reduction between 32.7% and 61.8% in SAD computation. The results show E-ADME provides lower BD-rate and higher reductions on SAD computation (68.2%) compared to [2].

Our early termination block-level decision can be compared to [10] that also proposes a block-level decision. Its results reduce the encoding time between 9.6% and 13.8% with a BD-rate increase of 0.062% to 0.409%. This high encoding timesaving is similar to the one provided by this thesis; however, their solution requires more than the double of the BD-rate increase, showing the advantage of our solution.

Comparing our quadtree limitation using machine learning with the related work focusing on quadtree-level, one can notice our solution provides competitive encoding time-saving gains, reaching the same results than [38] and better results than [32]. Additionally, our solution reached a much smaller BD-rate increase than those solutions.

7. CONCLUSIONS AND FUTURE WORKS

This thesis proposed several algorithms at different granularity levels to reduce the encoding effort at intra- and inter-frame prediction of depth maps. This Thesis was motivated by the introduction of depth maps in 3D-HEVC along with the texture views aiming to achieve high quality and compression of 3D videos. The 3D-HEVC standardization proposed and adopted several new tools to encode depth maps efficiently, increasing the encoding effort of 3D videos. Consequently, 3D-HEVC introduced new challenges regarding efficient and efficacious video coding, especially for real-time system design.

By designing timesaving algorithms to the most time-consuming encoding tools and implementing enhancements to the bipartition modes, which are new encoding tools for depth map coding, this thesis contributes to designing efficient 3D video coding systems. The main contributions of this Thesis are: (i) the encoding effort reduction of the intra-frame prediction; (ii) implementations and enhancements on the bipartition modes; and (iii) the encoding effort reduction of the inter-frame prediction.

The topic described in (i) englobes a deep encoding time and mode distribution analysis, which boost the design of four algorithms for encoding effort reduction of the intra-frame prediction: DFPS, P&GMOF, ED-RMD, and quadtree limitation with machine learning. DFPS and P&GMOF were designed focusing on reducing the Intra_Wedge encoding effort, ED-RMD was planned for reducing the encoding effort spent in Transform-Quantization and DC-only evaluations. The quadtree limitation with machine learning was designed as a high-level decision approach, where the encoding effort spent in lower levels of the quadtree were skipped in some scenarios. Our results demonstrate the proposed algorithms reduce 6.5% to 52.4% the encoding effort, impacting only 0.0119% to 0.1770% the BD-rate of the synthesized views. These results are very competitive when compared to the related work.

Four contributions related to implementing enhancements for the bipartition modes were described in the topic (ii) of this Thesis. An encoding effort control system was designed for maintaining the bipartition modes evaluation stable over the encoded frames. This system was designed using a PID controller that controls the threshold of a timesaving technique. Experimental results demonstrated the system was capable of stabilizing in a few frames. We explored two parallelism strategies into a multicore CPU with an extended analysis into a GPU to speed up the Intra_Wedge execution without changing the algorithm. With the GPU optimizations and the application of a timesaving algorithm in Intra_Wedge execution, it was possible to achieve real-time encoding of 1080p depth maps. Considering Intra_Wedge requires too much memory to store the wedgelet patterns, we designed seven lossless strategies to compress these patterns. Experimental results demonstrated 79% of compression was achieved using our best technique, which can save 75% of power when designed in hardware. The last contribution of the topic (ii) was the design of a decoder for bipartition modes. This architecture was designed using a wedgelet compression strategy, and it was synthesized for CMOS 65nm ST technology targeting achieving real-time decoding, requiring 5,165 logic gates and dissipating 1.57 mW of power, considering the worst case scenario.

Finally, the contributions related to the encoding effort reduction for inter-frame prediction were presented in the topic (iii). Here, three algorithms were designed, each one at a different granularity. E-ADME considered the depth map simplicity to allow speeding up the depth map ME when encoding homogeneous regions while keeping the original evaluation when encoding edges. This algorithm was based in an edge classifier of a previous work that speeds up the homogeneous regions evaluation. We designed a block-level scheme for early termination decisions. Based on the results of the most used modes (i.e., Skip and Intra-picture skip), this algorithm determines if the remaining evaluations inside an encoding block are required or not. Therefore, when Skip and Intra-picture skip results are satisfactory, the block is encoded with these modes and the encoding process is finalized. Our last contribution inside this topic was designing specific decision trees using machine learning for pruning the quadtree at inter-frame prediction. This contribution was similar to the intra-frame prediction contribution; however, in inter-frame prediction more data were evaluated to achieve better results. Our experimental results obtained between 3.2% and 52.0% of encoding effort reduction, with a drawback of increasing the BD-rate in a range from 0.1481% to 0.1976% in the synthesized views.

Summarizing this Thesis, it was designed several algorithms and encoder/decoder enhancements. Our evaluations demonstrated that significant timesaving can be achieved in depth map encoding when considering its characteristics with minor impact in the quality of synthesized views. However, there are several points that can still be explored for achieving higher performance in depth maps coding as described next Section.

7.1 Future Works and Open Research Possibilities

We can see several open research possibilities that were not covered by the literature yet in depth map coding. Our highest timesaving algorithms in both intra- and inter-frame prediction achieved around 50% of encoding effort reduction, which is already a significant reduction. Higher reductions can be achieved by using more sophisticated machine learning techniques such as deep learning. However, for allowing real-time applications using 3D-HEVC encoders, it is necessary to reduce even more the encoding effort or speeding up the encoding by using parallelism. Therefore, there still significant space for outcoming works to allow this real-time design.

On hardware design, one can find only a few works in the literature designing small solutions for 3D-HEVC depth maps coding. However, there is still space for design fast and low-power systems exploring the parallelism in the depth map encoding. For the parallelism exploration, only this Thesis has already explored it at 3D-HEVC and focusing on only one module. Therefore, several encoding/decoding tools can be the target for this parallelism exploration. Besides, tiles were not much explored in the literature in both texture and depth map encoding.

REFERENCES

[1] Afonso, V.; Susin, A.; Audibert, L.; Saldanha, M.; Conceição, R.; Porto, M.; Zatt, B.; Agostini, L. “Low-power and high-throughput hardware design for the 3D-HEVC depth intra skip”. In: IEEE International Symposium on Circuits and Systems, 2017, pp. 1–4.

[2] Afonso, V.; Susin, A.; Perleberg, M.; Conceição, R.; Corrêa, G.; Agostini, L.; Zatt, B.; Porto, M. “Hardware-friendly unidirectional disparity-search algorithm for 3d-hevc”. In: IEEE International Symposium on Circuits and Systems, 2018, pp. 1–5.

[3] Amish, F.; Bourennane, E.-B. “An efficient hardware solution for 3D-HEVC intra-prediction”, *Journal of Real-Time Image Processing*, vol. 1–1, Jan 2017, pp. 1–13.

[4] Bjontegaard, G. “Calculation of average PSNR differences between RD-curves,”, Technical Report, ITU-T SC16/SG16 VCEG-M33, 2001, 4p.

[5] Brunk, C. A.; Pazzani, M. J. “An investigation of noise-tolerant relational concept learning algorithms”. In: Machine Learning Proceedings, 1991, pp. 389–393.

[6] Budagavi, M.; Fuldseth, A.; Bjontegaard, G. “High efficiency video coding (HEVC): Algorithms and Architectures”. Cambridge, Massachusetts, USA: Springer, 2014, chap. 6, pp. 141–169.

[7] Chen, H.; Fu, C.-H.; Chan, Y.-L.; Zhu, X. “Early intra block partition decision for depth maps in 3d-hevc”. In: IEEE International Conference on Image Processing, 2018, pp. 1777–1781.

[8] Chen, Y.; Tech, G.; Wegner, K.; Yea, S. “Test model 11 of 3D-HEVC and MV-HEVC”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2015, 58p.

[9] Cheng, Y.-S.; Chen, Z.-Y.; Chang, P.-C. “An H.264 spatio-temporal hierarchical fast motion estimation algorithm for high-definition video”. In: IEEE International Symposium on Circuits and Systems, 2009, pp. 880–883.

[10] Conceição, R.; Avila, G.; Corrêa, G.; Porto, M.; Zatt, B.; Agostini, L. “Complexity reduction for 3D-HEVC depth map coding based on early skip and early DIS scheme”. In: IEEE International Conference on Image Processing, 2016, pp. 1116–1120.

[11] Correa, G.; Assuncao, P.; Agostini, L.; da Silva Cruz, L. A. “Complexity control of HEVC through quadtree depth estimation”. In: EUROCON, 2013, pp. 81–86.

[12] Correa, G.; Assuncao, P. A.; Agostini, L. V.; da Silva Cruz, L. A. “Fast HEVC encoding decisions using data mining”, *IEEE transactions on circuits and systems for video technology*, vol. 25–4, Oct 2015, pp. 660–673.

[13] Dagum, L.; Menon, R. “OpenMP: an industry standard API for shared-memory programming”, *IEEE computational science and engineering*, vol. 5–1, Jan 1998, pp. 46–55.

[14] Deng, X.; Xu, M.; Li, S.; Wang, Z. “Complexity control of HEVC based on region-of-interest attention model”. In: IEEE Visual Communications and Image Processing Conference, 2014, pp. 225–228.

[15] Domañski, M.; Grajek, T.; Klimaszewski, K.; Kurc, M.; Stankiewicz, O.; Stankowski, J.; Wegner, K. “Poznan multiview video test sequences and camera parameters”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2009, 6p.

[16] Fehn, C. “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV”. In: Stereoscopic Displays and Virtual Reality Systems XI, 2004, pp. 93–105.

[17] Flynn, M. J. “Very high-speed computing systems”, *Proceedings of the IEEE*, vol. 54–12, Dec 1966, pp. 1901–1909.

[18] Fu, C.-H.; Zhang, H.-B.; Su, W.-M.; Tsang, S.-H.; Chan, Y.-L. “Fast wedgelet pattern decision for DMM in 3D-HEVC”. In: IEEE International Conference on Digital Signal Processing, 2015, pp. 477–481.

[19] Gu, Z.; Zheng, J.; Ling, N.; Zhang, P. “Fast intra prediction mode selection for intra depth map coding”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2013, 4p.

[20] Gu, Z.; Zheng, J.; Ling, N.; Zhang, P. “Fast segment-wise DC coding for 3D video compression”. In: IEEE International Symposium on Circuits and Systems, 2015, pp. 2780–2783.

[21] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. “The WEKA data mining software: an update”, *ACM SIGKDD explorations newsletter*, vol. 11–1, Jun 2009, pp. 10–18.

[22] Harris, C.; Stephens, M. “A combined corner and edge detector”. In: Alvey vision conference, 1988, pp. 147–151.

[23] He, Z.; Yu, L.; Zheng, X.; Ma, S.; He, Y. “Framework of AVS2-video coding”. In: IEEE International Conference on Image Processing, 2013, pp. 1515–1519.

[24] HHI. “3D-HEVC Test Model”. Source: hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/, Jan 2018.

[25] Ho, Y.-S.; Lee, E.-K.; Lee, C. “M15419, multiview video test sequence and camera parameters”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2008, 6p.

[26] Huffman, D. A. “A method for the construction of minimum-redundancy codes”, *Proceedings of the IRE*, vol. 40–9, Sep 1952, pp. 1098–1101.

[27] ITU-T VCEG and ISO/IEC MPEG. “3D-HEVC Test Model”. Source: https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/HTM-16.0/, Sep 2017.

[28] Jouppi, N. P.; Kahng, A. B.; Muralimanohar, N.; Srinivas, V. “CACTI-IO: CACTI with off-chip power-area-timing models”, *IEEE Transactions on Very Large Scale Integration Systems*, vol. 23–7, Jul 2015, pp. 1254–1267.

[29] Kauff, P.; Atzpadin, N.; Fehn, C.; Müller, M.; Schreer, O.; Smolic, A.; Tanger, R. “Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability”, *Signal Processing: Image Communication*, vol. 22–2, Feb 2007, pp. 217–234.

[30] Lainema, J.; Bossen, F.; Han, W.-J.; Min, J.; Ugur, K. “Intra coding of the HEVC standard”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22–12, Dec 2012, pp. 1792–1801.

[31] Lee, J.; Park, M.; Kim, C. “3d-ce1: depth intra skip (dis) mode”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2015, 5p.

[32] Lei, J.; Duan, J.; Wu, F.; Ling, N.; Hou, C. “Fast mode decision based on grayscale similarity and inter-view correlation for depth map coding in 3D-HEVC”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28–3, Sep 2018, pp. 706–718.

[33] Liu, H.; Chen, Y. “Generic segment-wise DC for 3D-HEVC depth intra coding”. In: IEEE International Conference on Image Processing, 2014, pp. 3219–3222.

[34] Ma, S.; Wang, Y.; Zhu, C.; Lin, Y.; Zheng, J. “Reducing Wedgelet lookup table size with down-sampling for depth map coding in 3D-HEVC”. In: IEEE International Workshop on Multimedia Signal Processing, 2015, pp. 1–5.

[35] Marpe, D.; Schwarz, H.; Bosse, S.; Bross, B.; Helle, P.; Hinz, T.; Kirchhoffer, H.; Lakshman, H.; Nguyen, T.; Oudin, S.; et al.. “Video compression using nested quadtree structures, leaf merging, and improved techniques for motion representation and entropy coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20–12, Dec 2010, pp. 1676–1687.

[36] Marpe, D.; Schwarz, H.; Wiegand, T. “Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard”, *IEEE Transactions on circuits and systems for video technology*, vol. 13–7, Jul 2003, pp. 620–636.

[37] Merkle, P.; Müller, K.; Marpe, D.; Wiegand, T. “Depth intra coding for 3D video based on geometric primitives”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26–3, Feb 2016, pp. 570–582.

[38] Mora, E. G.; Jung, J.; Cagnazzo, M.; Pesquet-Popescu, B. “Initialization, limitation, and predictive coding of the depth and texture quadtree in 3D-HEVC”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24–9, Sep 2014, pp. 1554–1565.

[39] Mukherjee, D.; Bankoski, J.; Grange, A.; Han, J.; Koleszar, J.; Wilkins, P.; Xu, Y.; Bultje, R. “The latest open-source video codec VP9-an overview and preliminary results”. In: Picture Coding Symposium, 2013, pp. 390–393.

[40] Müller, K.; Schwarz, H.; Marpe, D.; Bartnik, C.; Bosse, S.; Brust, H.; Hinz, T.; Lakshman, H.; Merkle, P.; Rhee, F. H.; et al.. “3D high-efficiency video coding for multi-view video and depth data”, *IEEE Transactions on Image Processing*, vol. 22–9, Sep 2013, pp. 3366–3378.

[41] Müller, K.; Vetro, A. “Common test conditions of 3DV Core experiments”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2014, 7p.

[42] NICT. “National Institute of Information and Communication Technology”. Source: ftp://ftp.merl.com/pub/tian/NICT-3D/Shark/, Sep 2017.

[43] Nvidia Corporation. “NVIDIA CUDA Programming Guide”, Technical Report, NVIDIA Corporation Santa Clara, CA, 2011, 173p.

[44] Peng, K.-K.; Chiang, J.-C.; Lie, W.-N. “Low complexity depth intra coding combining fast intra mode and fast CU size decision in 3D-HEVC”. In: IEEE International Conference on Image Processing, 2016, pp. 1126–1130.

[45] Quinlan, J. R. “C4. 5: programs for machine learning”. Elsevier, 2014, 302p.

[46] Rusanovskyy, D.; Aflaki, P.; Hannuksela, M. “Undo Dancer 3DV sequence for purposes of 3DV standardization”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2011, 6p.

[47] Saldanha, M.; Sanchez, G.; Marcon, C.; Agostini, L. “Block-level fast coding scheme for depth maps in three-dimensional high efficiency video coding”, *Journal of Electronic Imaging*, vol. 27–1, Feb 2018, pp. 010502.

[48] Saldanha, M.; Sanchez, G.; Marcon, C.; Agostini, L. “Fast 3D-Hevc Depth Maps Intra-Frame Prediction Using Data Mining”. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2018, pp. 1738–1742.

[49] Saldanha, M.; Sanchez, G.; Marcon, C.; Agostini, L. “Fast 3D-HEVC Depth Maps Encoding Using Machine Learning”, *IEEE Transactions on Circuits and Systems for Video Technology (submitted)*, vol. 1–1, Jan 2019, pp. 12.

[50] Saldanha, M.; Sanchez, G.; Zatt, B.; Porto, M.; Agostini, L. “Complexity reduction for the 3D-HEVC depth maps coding”. In: IEEE International Symposium on Circuits and Systems, 2015, pp. 621–624.

[51] Saldanha, M.; Sanchez, G.; Zatt, B.; Porto, M.; Agostini, L. “Energy-aware scheme for the 3D-HEVC depth maps prediction”, *Journal of Real-Time Image Processing*, vol. 13–1, Mar 2017, pp. 55–69.

[52] Saldanha, M.; Zatt, B.; Porto, M.; Agostini, L.; Sanchez, G. “Solutions for DMM-1 complexity reduction in 3D-HEVC based on gradient calculation”. In: IEEE 7th Latin American Symposium on Circuits & Systems, 2016, pp. 211–214.

[53] Sanchez, G.; Agostini, L.; Marcon, C. “Energy-aware light-weight DMM-1 patterns decoders with efficiently storage in 3D-HEVC”. In: Symposium on Integrated Circuits and Systems Design, 2016, pp. 1–6.

[54] Sanchez, G.; Agostini, L.; Marcon, C. “Complexity reduction by modes reduction in RD-list for intra-frame prediction in 3D-HEVC depth maps”. In: IEEE International Symposium on Circuits and Systems, 2017, pp. 1–4.

[55] Sanchez, G.; Agostini, L.; Marcon, C. “A reduced computational effort mode-level scheme for 3D-HEVC depth maps intra-frame prediction”, *Journal of Visual Communication and Image Representation*, vol. 54–1, Jul 2018, pp. 193–203.

[56] Sanchez, G.; Agostini, L.; Marcon, C. “High Efficient Architecture for 3D-HEVC DMM-1 Decoder Targeting 1080p Videos”. In: IEEE International Symposium on Circuits and Systems, 2018, pp. 1–5.

[57] Sanchez, G.; Agostini, L.; Mór, F.; Marcon, C. “Low-area scalable hardware architecture for DMM-1 encoder of 3D-HEVC video coding standard”. In: Symposium on Integrated Circuits and Systems Design, 2017, pp. 36–40.

[58] Sanchez, G.; Agostini, L.; Sousa, L.; Marcon, C. “3D-HEVC DMM-1 Parallelism Exploration Targeting Multicore Systems”. In: Symposium on Integrated Circuits and Systems, 2018, pp. 1–5.

[59] Sanchez, G.; Agostini, L.; Sousa, L.; Marcon, C. “Parallelism exploration for 3D high-efficiency video coding depth modeling mode one”, *Journal of Real-Time Image Processing*, vol. 1–1, Sep 2018, pp. 1–11.

[60] Sanchez, G.; Cataldo, R.; Fernandes, R.; Agostini, L.; Marcon, C. “3D-HEVC depth maps intra prediction complexity analysis”. In: IEEE International Conference on Electronics, Circuits and Systems, 2016, pp. 348–351.

[61] Sanchez, G.; Fernandes, R.; Cataldo, R.; Agostini, L.; Marcon, C. “Low Area Reconfigurable Architecture for 3D-HEVC DMMs Decoder Targeting 1080p Videos”. In: IEEE International Conference on Electronics, Circuits and Systems, 2018, pp. 1–4.

[62] Sanchez, G.; Jordani, L.; Marcon, C.; Agostini, L. “DFPS: a fast pattern selector for depth modeling mode 1 in three-dimensional high-efficiency video coding standard”, *Journal of Electronic Imaging*, vol. 25–6, Nov 2016, pp. 063011.

[63] Sanchez, G.; Marcon, C.; Agostini, L. “Real-time scalable hardware architecture for 3D-HEVC bipartition modes”, *Journal of Real-Time Image Processing*, vol. 13–1, Mar 2017, pp. 71–83.

[64] Sanchez, G.; Marcon, C.; Agostini, L. V. “Exploration of depth modeling mode one lossless wedgelets storage strategies for 3D-high efficiency video coding”, *Journal of Electronic Imaging*, vol. 27–1, Feb 2018, pp. 013025.

[65] Sanchez, G.; Saldanha, M.; Agostini, L.; Marcon, C. “Depth modeling modes complexity control system for the 3D-HEVC video encoder”. In: European Signal Processing Conference, 2017, pp. 1021–1025.

[66] Sanchez, G.; Saldanha, M.; Balota, G.; Zatt, B.; Porto, M.; Agostini, L. “A complexity reduction algorithm for depth maps intra prediction on the 3D-HEVC”. In: IEEE Visual Communications and Image Processing Conference, 2014, pp. 137–140.

[67] Sanchez, G.; Saldanha, M.; Balota, G.; Zatt, B.; Porto, M.; Agostini, L. “Complexity reduction for 3D-HEVC depth maps intra-frame prediction using simplified edge detector algorithm”. In: IEEE International Conference on Image Processing, 2014, pp. 3209–3213.

[68] Sanchez, G.; Saldanha, M.; Zatt, B.; Porto, M.; Agostini, L.; Marcon, C. “Edge-aware depth motion estimation—A complexity reduction scheme for 3D-HEVC”. In: European Signal Processing Conference, 2017, pp. 1524–1528.

[69] Sanchez, G.; Silveira, J.; Agostini, L.; Marcon, C. “Performance Analysis of Depth Intra Coding in 3D-HEVC”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1–1, Aug 2018, pp. 1–12.

[70] Sanchez, G.; Zatt, B.; Porto, M.; Agostini, L. “A real-time 5-views HD 1080p architecture for 3D-HEVC Depth Modeling Mode 4”. In: Symposium on Integrated Circuits and Systems Design, 2014, pp. 1–6.

[71] Shan, N.; Zhou, W.; Duan, Z. “Complexity control for HEVC by visual saliency detection”. In: IEEE International Conference on Signal Processing, Communications and Computing, 2016, pp. 1–5.

[72] Smolic, A.; Muller, K.; Dix, K.; Merkle, P.; Kauff, P.; Wiegand, T. “Intermediate view interpolation based on multiview video plus depth for advanced 3D video systems”. In: IEEE International Conference on Image Processing, 2008, pp. 2448–2451.

[73] Sullivan, G. J.; Ohm, J.-R.; Han, W.-J.; Wiegand, T.; et al.. “Overview of the high efficiency video coding (HEVC) standard”, *IEEE Transactions on circuits and systems for video technology*, vol. 22–12, Dec 2012, pp. 1649–1668.

[74] Tang, X.-l.; Dai, S.-k.; Cai, C.-h. “An analysis of TZSearch algorithm in JMVC”. In: International Conference on Green Circuits and Systems, 2010, pp. 516–520.

[75] Tanimoto Lab NICT. “National Institute of Information and Communication Technology”. Source: http://www.tanimoto.nuee.nagoya-u.ac.jp/, Sep 2017.

[76] Tech, G.; Chen, Y.; Müller, K.; Ohm, J.-R.; Vetro, A.; Wang, Y.-K. “Overview of the multiview and 3D extensions of high efficiency video coding”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26–1, Jan 2016, pp. 35–49.

[77] Tech, G.; Schwarz, H.; Müller, K.; Wiegand, T. “3D video coding using the synthesized view distortion change”. In: Picture Coding Symposium, 2012, pp. 25–28.

[78] Tuan, J.-C.; Chang, T.-S.; Jen, C.-W. “On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture”, *IEEE Transactions on circuits and systems for video technology*, vol. 12–1, Jan 2002, pp. 61–72.

[79] Winken, M.; Helle, P.; Marpe, D.; Schwarz, H.; Wiegand, T. “Transform coding in the HEVC test model”. In: IEEE International Conference on Image Processing, 2011, pp. 3693–3696.

[80] Xu, J.; Joshi, R.; Cohen, R. A. “Overview of the emerging HEVC screen content coding extension”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26–1, Jan 2016, pp. 50–62.

[81] Zhang, H.-B.; Chan, Y.-L.; Fu, C.-H.; Tsang, S.-H.; Siu, W.-C. “Quadtree decision for depth intra coding in 3D-HEVC by good feature”. In: IEEE International Conference on Acoustics, Speech and Signal Processing, 2016, pp. 1481–1485.

[82] Zhang, H.-B.; Fu, C.-H.; Chan, Y.-L.; Tsang, S.-H.; Siu, W.-C. “Efficient depth intra mode decision by reference pixels classification in 3D-HEVC”. In: IEEE International Conference on Image Processing, 2015, pp. 961–965.

[83] Zhang, H.-B.; Tsang, S.-H.; Chan, Y.-L.; Fu, C.-H.; Su, W.-M. “Early determination of intra mode and segment-wise DC coding for depth map based on hierarchical coding structure in 3D-HEVC”. In: Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, 2015, pp. 374–378.

[84] Zhang, J.; Li, R.; Li, H.; Rusanovskyy, D.; Hannuksela, M. M. “Ghost Town Fly 3DV sequence for purposes of 3DV standardization”, Technical Report, ISO/IEC JTC1/SC29/WG11, 2011, 5p.

[85] Zhao, L.; Zhang, L.; Ma, S.; Zhao, D. “Fast mode decision algorithm for intra prediction in HEVC”. In: IEEE Visual Communications and Image Processing, 2011, pp. 1–4.

[86] Zhao, Y.; Zhu, C.; Chen, Z.; Tian, D.; Yu, L. “Boundary artifact reduction in view synthesis of 3D video: from perspective of texture-depth alignment”, *IEEE Transactions on Broadcasting*, vol. 57–2, Jun 2011, pp. 510–522.

[87] Zhu, S.; Ma, K.-K. “A new diamond search algorithm for fast block-matching motion estimation”, *IEEE transactions on Image Processing*, vol. 9–2, Feb 2000, pp. 287–290.

APPENDIX A – PUBLICATION LIST

Patent applications during the Ph.D. period:

1. Title: Método de otimização em codificação de imagens tridimensionais e codificador para imagens tridimensionais
   Authors: G. Sanchez, C. Marcon, M. Saldanha, L. Agostini
   Country: Brazil
   Process Number: BR 10 2018 005971 8
   Date: 2018

Articles and papers published or accepted during the Ph.D. period:

1. Title: Real-Time Scalable Hardware Architecture for 3D-HEVC Bipartition Modes
   Authors: G. Sanchez, C. Marcon, L. Agostini
   Journal: Journal of Real-Time Image Processing (JRTIP)
   Date: 2016

2. Title: DFPS: A Fast Pattern Selector for Depth Modeling Mode 1 in Three-Dimensional High Efficiency Video Coding Standard
   Authors: G. Sanchez, L. Jordani, C. Marcon, L. Agostini
   Journal: Journal of Electronic Imaging (JEI)
   Date: 2016

3. Title: Energy-Aware Light-Weight DMM-1 Patterns Decoders with Efficiently Storage in 3D-HEVC
   Authors: G. Sanchez, L. Agostini, C. Marcon
   Conference: Symposium on Integrated Circuits and Systems Design (SBCCI)
   Date: 2016

4. Title: Real-Time Simplified Edge Detector Architecture for 3D-HEVC Depth Maps Coding
   Authors: G. Sanchez, M. Saldanha, M. Porto, B. Zatt, L. Agostini, C. Marcon
   Conference: International Conference on Electronics, Circuits and Systems (ICECS)
   Date: 2016

5. Title: 3D-HEVC Depth Maps Intra Prediction Complexity Analysis.
   Authors: G. Sanchez, R. Cataldo, R. Fernandes, L. Agostini, C. Marcon.
   Conference: International Conference on Electronics, Circuits and Systems (ICECS)
   Date: 2016

6. Title: Evaluation of Emerging TSV-enabled Main Memories on the PARSEC Benchmark

Authors: R. Cataldo, G. Korol, R. Fernandes, G. Sanchez, D. Matos, C. Marcon
Conference: International Conference on Electronics, Circuits and Systems (ICECS)
Date: 2016

7. Title: Complexity Reduction by Modes Reduction in RD-List for Intra-Frame Prediction in 3D-HEVC Depth Maps
Authors: G. Sanchez, L. Agostini, C. Marcon
Conference: IEEE International Symposium Circuits and Systems (ISCAS)
Date: 2017

8. Title: Depth Modeling Modes Complexity Control System in 3D-HEVC
Authors: G. Sanchez, M. Saldanha, L. Agostini, C. Marcon
Conference: European Signal Processing Conference (EUSIPCO)
Date: 2017

9. Title: Edge-Aware Depth Motion Estimation – a Complexity Reduction Scheme for 3D-HEVC
Authors: G. Sanchez, M. Saldanha, B. Zatt, M. Porto, L. Agostini, C. Marcon
Conference: European Signal Processing Conference (EUSIPCO)
Date: 2017

10. Title: Low-Area Scalable Hardware Architecture for DMM-1 Encoder of 3D-HEVC Video Coding Standard
Authors: G. Sanchez, F. Mór, L. Agostini, C. Marcon
Conference: Symposium on Integrated Circuits and Systems Design (SBCCI)
Date: 2017

11. Title: Block-level fast coding scheme for depth maps in three-dimensional high efficiency video coding
Authors: M. Saldanha, G. Sanchez, C. Marcon, L. Agostini
Journal: Journal of Electronic Imaging (JEI)
Date: 2018

12. Title: Exploration of depth modeling mode one lossless wedgelets storage strategies for 3D-high efficiency video coding
Authors: G. Sanchez, C. Marcon, L. Agostini
Journal: Journal of Electronic Imaging (JEI)
Date: 2018

13. Title: Fast 3D-HEVC depth maps intra-frame prediction using data mining
Authors: M. Saldanha, G. Sanchez, C. Marcon, L. Agostini
Conference: International Conference on Acoustics, Speech, and Signal Processing

(ICASSP)
Date: 2018

14. Title: High Efficient Architecture for 3D-HEVC DMM-1 Decoder Targeting 1080p Videos
Authors: G. Sanchez, C. Marcon, L. Agostini
Conference: IEEE International Symposium Circuits and Systems (ISCAS)
Date: 2018

15. Title: Performance Analysis of Depth Intra Coding in 3D-HEVC
Authors: G. Sanchez, J. Silveira, L. Agostini, C. Marcon
Journal: IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)
Date: 2018

16. Title: A Reduced Computational Effort Mode-Level Scheme for 3D-HEVC Depth Maps Intra-Frame Prediction
Authors: G. Sanchez, L. Agostini, C. Marcon
Journal: Journal of Visual Communication and Image Representation (JVCIR)
Date: 2018

17. Title: Exploration of Parallel Architectures for 3D-HEVC Depth Modeling Mode One
Authors: G. Sanchez, L. Agostini, L. Sousa, C. Marcon
Journal: Journal of Real-time Image Processing (JRTIP)
Date: 2018

18. Title: 3D-HEVC DMM-1 Parallelism Exploration Targeting Multicore Systems
Authors: G. Sanchez, L. Agostini, L. Sousa, C. Marcon
Conference: Symposium on Integrated Circuits and Systems Design (SBCCI)
Date: 2018

19. Title: Hardware-Oriented Wedgelet Evaluation Skip for DMM-1 in 3D-HEVC
Authors: G. Sanchez, M. Saldanha, L. Agostini, C. Marcon
Conference: Symposium on Integrated Circuits and Systems Design (SBCCI)
Date: 2018

20. Title: DCDM-INTRA: Dynamically Configurable 3D-HEVC Depth Maps Intra-Frame Prediction Algorithm
Authors: G. Sanchez, R. Fernandes, L. Agostini, C. Marcon
Conference: IEEE International Conference on Image Processing (ICIP)
Date: 2018

21. Title: Energy saving on DTN using trajectory inference model
Authors: A. de Vit, C. Marcon, R. Nunes, T. Webber, G. Sanchez, C. Rolim
Conference: Annual ACM Symposium on Applied Computing
Date: 2018

22. Title: Efficient HEVC intra-frame prediction using curved angular modes
    Authors: R. Fernandes, G. Sanchez, R. Cataldo, T. Webber, C. Marcon
    Conference: Electronic Letters
    Date: 2018

23. Title: Low Area Reconfigurable Architecture for 3D-HEVC DMMs Decoder Targeting 1080p Videos
    Authors: G. Sanchez, R. Fernandes, R. Cataldo, L. Agostini, C. Marcon
    Conference: International Conference on Electronics, Circuits and Systems (ICECS)
    Date: 2018

24. Title: Least-Squares Approximation Surfaces for High Quality Intra-Frame Prediction in Future Video Standards
    Authors: R. Fernandes, G. Sanchez, R. Cataldo, L. Agostini, C. Marcon
    Conference: International Conference on Electronics, Circuits and Systems (ICECS)
    Date: 2018

25. Title: Fast 3D-HEVC Depth Maps Encoding Using Machine Learning
    Authors: M. Saldanha, G. Sanchez, C. Marcon, L. Agostini
    Journal: IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)
    Date: 2019

26. Title: Analysis of Parallel Encoding using Tiles in 3D-High Efficiency Video Coding
    Authors: G. Sanchez, M. Saldanha, L. Agostini, C. Marcon
    Journal: Signal, Image and Video Processing
    Date: 2019

27. Title: TITAN: Tile Timing-Aware Balancing Algorithm for Speeding Up the 3D-HEVC Intra Coding
    Authors: M. Saldanha, G. Sanchez, L. Agostini, C. Marcon
    Conference: IEEE International Symposium Circuits and Systems (ISCAS)
    Date: 2019

Articles and papers under review:

1. Title: Multicore Parallelism Exploration Targeting 3D-HEVC Intra-Frame Prediction
   Authors: G. Sanchez, R. Fernandes, R. Cataldo, L. Agostini, L. Sousa, C. Marcon
   Journal: IEEE Design & Test of Computers
   Date: Regular submission – without specific date

2. Title: Curved Intra-Frame Extensions for Angular Prediction Modes in HEVC
   Authors: R. Fernandes, G. Sanchez, R. Cataldo, C. Marcon
   Journal: IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)
   Date: Regular submission – without specific date

3. Title: 3D-HEVC Bipartition Modes Encoder and Decoder Design Targeting High Resolution Videos
   Authors: G. Sanchez, M. Saldanha, R. Fernandes, R. Cataldo, L. Agostini, C. Marcon
   Journal: IEEE Transactions on Circuits and Systems I
   Date: Regular submission – without specific date

4. Title: Subutai: Speeding up Legacy Parallel Applications through Data Synchronization
   Authors: R. Cataldo, R. Fernandes, K. Martin, J. Sepulveda, G. Sanchez, C. Marcon, J.-P. Diguet
   Journal: IEEE Transactions on Computers
   Date: Regular submission – without specific date