UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

VLADIMIR AFONSO

High-Throughput Dedicated Hardware Design Targeting the 3D-HEVC-Prediction Coding Tools

Thesis presented as partial requirement for the Ph.D. degree in Microelectronics.

Advisor: Prof. Dr. Altamiro Amadeu Susin
Co-advisor: Prof. Dr. Luciano Volcan Agostini

Porto Alegre
2019

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Afonso, Vladimir

High-Throughput Dedicated Hardware Design Targeting the 3D-HEVC-Prediction Coding Tools / Vladimir Afonso. – Porto Alegre: PGMicro da UFRGS, 2019.

204 f.:il.

Orientador: Altamiro Amadeu Susin; Coorientador: Luciano Volcan Agostini.

Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2019.

1. Video Coding. 2. 3D-HEVC. 3. Depth Maps. 4. Real-Time Embedded System. 5. Low-Energy Hardware Design. I. Susin, Altamiro Amadeu, orient. II. Agostini, Luciano Volcan, coorient. III. Título.

AGRADECIMENTOS

Quando se inicia um curso de doutorado, várias são as incomodações que permeiam os nossos pensamentos. E uma das dúvidas predominantes é a de como lidar com o tempo que temos para trabalhar, que por horas parece tão pouco e por horas parece mais do que o necessário. Um doutorado talvez seja a mais perfeita analogia para o entendimento da relatividade, usando o tempo (depois do exemplo dado por Albert Einstein), pois em um relance de pensamento pode nos parecer que vivemos outra vida ou, ainda, que não vivemos um dia sequer. Por isso, toda tese de doutorado (também) deveria se chamar: Uma Breve História do Tempo.

Qual será o tempo de um doutorado? É o tempo suficiente para comprometer o corpo. É o tempo suficiente para comprometer a mente. É o tempo suficiente para atravessar uma separação. É o tempo suficiente para se afastar de um irmão. É o tempo suficiente para ver a pessoa mais forte que conhecia desabar. É o tempo suficiente para ver várias pessoas queridas partir. É o tempo suficiente para ver a pessoa mais corajosa que conhecia findar. É o tempo suficiente para ver o mundo ruir.

Mas o tempo de um doutorado também é o tempo suficiente para se levantar. É o tempo suficiente para se sentir literalmente mais leve. É o tempo suficiente para correr longas distâncias. É o tempo suficiente para se reinventar. É o tempo suficiente para se apaixonar e se desapaixonar. É o tempo suficiente para ser perdoado e para perdoar. É o tempo suficiente para entender o valor das coisas, da vida e das pessoas. Enfim, é o tempo suficiente para recomeçar.

Qual será o tempo de um doutorado? É o tempo suficiente para entender que o nosso tempo é finito. É o tempo suficiente para entender que o momento derradeiro é imprevisível. É o tempo suficiente para entender o tempo.

Por entender o valor da vida e das pessoas, em primeiro lugar, agradeço aos meus pais, Eledir Lobo Afonso e Neli Afonso (*in memoriam*), que apesar de todas as adversidades que lhes foram impostas pela vida, tomaram a decisão de permitir que eu deixasse a pequena cidade de Herval/RS em 1997, ainda adolescente, para cursar um curso técnico em Eletrônica na antiga ETFPel (Escola Técnica Federal) em Pelotas/RS. Por sabedoria, eles me deram uma das coisas mais importantes, senão a mais importante, que pais podem dar aos seus filhos, a oportunidade de estudar e ter uma profissão. Deram-me a oportunidade a qual eles mesmos não puderam usufruir plenamente durante suas vidas. O destino impôs que minha mãe partisse antes de ver a concretização dessa etapa da minha vida, mas antes ficou claro para nós dois a importância de cada um na vida do outro. Sou grato por isso. Agradeço também a minha avó Sirena, meu irmão Rogério, minha sobrinha Ana Júlia e aos meus demais familiares, assim como aos meus amigos, pela compreensão da ausência em diversos momentos durante o desenvolvimento desse trabalho.

Faço um agradecimento especial ao meu orientador, Prof. Altamiro Susin, e ao meu coorientador, Prof. Luciano Agostini, que além de me oportunizarem a pesquisa de um tema extremamente relevante, disponibilizaram parte do seu tempo para me orientar no desenvolvimento de cada etapa desse trabalho. Sou grato pelas discussões, revisões e todas outras contribuições dadas por vocês. Ao Prof. Luciano Agostini também agradeço por ter colaborado para o meu desenvolvimento como ser humano, e para o meu melhor entendimento sobre todo o contexto em que as relações humanas se estabelecem.

Agradeço ao Prof. Marcelo Porto, com quem tenho uma relação de amizade desde 1998, ou seja, há mais de 20 anos, por ter participado ativamente do meu doutorado, dando contribuições para o trabalho mesmo sem ter orientação formal. Da mesma forma, agradeço enormemente ao Prof. Bruno Zatt, que por diversos momentos, quando eu tinha dúvidas gigantescas sobre o desenvolvimento do trabalho, me direcionou para as escolhas que hoje entendo terem sido as mais acertadas.

A todos os professores, bolsistas de iniciação científica, mestrandos e doutorandos do GACI (Grupo de Arquiteturas e Circuitos Integrados) e do ViTech (*Video Technology Research Group*) da UFPel, fica o meu agradecimento pela oportunidade de aprendizado e pela boa convivência estabelecida durante esses anos de pesquisa. Poderia citar dezenas de nomes de amigos que fiz durante o doutorado, fica aqui o meu abraço fraterno em todos vocês. Em especial, agradeço aos amigos doutorandos Ruhan Conceição (a quem chamo de maninho) e Mario Saldanha (um craque de bola) e ao amigo mestrando Murilo Perleberg (o golden boy) que colaboraram do início ao fim para que este trabalho se tornasse realidade. Não posso deixar de agradecer também a Mariana Ucker, o Luan Audibert, o Henrique Maich e o Jones Goebel, que colaboraram ou participaram de forma importante nesse trabalho, em diferentes momentos.

Deixo o meu agradecimento aos colegas de doutorado do PGMicro e do PPGC da UFRGS, em especial, aos amigos Alexandra Zimpeck, Louise Etcheverry, Renato Campana, Guillermo Daniel Ortega Galeano e Fabio Irigon com quem dividi muitos instantes, várias tarefas, momentos de estudo e de trabalho pela universidade.

Agradeço aos colegas professores do Curso Técnico em Eletromecânica do IFSul que ministraram as aulas da disciplina de Eletrônica Industrial enquanto eu estava afastado para o doutorado. Faço um agradecimento especial aos professores e amigos Mario Farias, Jair Jonko Araujo, Andrea Fisher, Claudio Machado e Igor Bederode por terem me dado apoio incondicional enquanto não conseguia o afastamento no IFSul para cursar o doutorado em Microeletrônica em Porto Alegre/RS. Agradeço também aos professores Carlos Corrêa, José Octávio Badia, Rafael Griep, Rodrigo Souza, Velington Neumman e André Oldoni, que juntamente com os professores Andrea, Claudio e Igor se comprometeram em assumir a minha carga horária (2015) até que um professor substituto fosse disponibilizado pelo IFSul.

Por fim, agradeço a todos aqueles, que de alguma forma, seja participando da minha formação profissional ou como ser humano, contribuíram para que este objetivo fosse alcançado.

ABSTRACT

The popularization of multimedia services has pushed forward the development of 2D/3D (Two and Three Dimensional) video-capable embedded mobile devices. In this scenario, 3D-video systems based on the simultaneous exhibition of multiple views are also expected, including systems capable of dealing with high and ultra-high resolutions. To meet this demand and the huge amount of data to be processed and stored, an extension for the HEVC (High Efficiency Video Coding) standard targeting three-dimensional video coding was developed by the ISO/IEC (International Organization for Standardization / International Electrotechnical Commission) and ITU-T (International Telecommunication Union – Telecommunication) experts. The HEVC is state-of-the-art for 2D video coding, and its 3D extension is called 3D-HEVC. The 3D-HEVC uses the MVD (Multi-view plus Depth) concept which associates a depth-map with each texture frame for each view that composes the video sequence. Because of that, 3D-HEVC defines several novel coding tools to make possible the 3D-video processing with multiple views with increasing resolutions under this novel perspective. As a result, the 3D-HEVC extension requires a high computational effort. Since 2D/3D video-capable embedded mobile devices require efficient energy/memory-management strategies to deal with severe memory/processing requirements and limited energy supply, the development of energy and memory-aware systems targeting the 3D-HEVC is essential.

This thesis brings contributions for high-throughput and low-power architectures targeting the 3D-HEVC. These contributions are mainly centered on four designed architectures, as follows: (i) a low-power and high-throughput hardware design for the DIS (Depth Intra Skip) coding tool; (ii) a low-power and memory-aware depth-map Intra-frame prediction system based on complexity-reduction strategies; (iii) a ME/DE (Motion and Disparity Estimation) system that was designed for low-energy consumption, featuring a run-time adaptive memory hierarchy; (iv) a low-power and coding-efficient disparity estimation architecture based on the proposed iUDS (Improved Unidirectional Disparity-Search) algorithm. The contributions of these architectures to the state-of-the-art are confirmed by the publications made so far in high-quality conferences/journals.

Keywords: Video Coding. 3D-HEVC. Depth Maps. Real-Time Embedded System. Low-Energy Hardware Design.

Projeto de Hardware Dedicado de Elevada Taxa de Processamento para as Ferramentas de Codificação da Predição do 3D-HEVC

RESUMO

A popularização de serviços multimídia tem alavancado o desenvolvimento de dispositivos portáteis capazes de lidar com vídeos 2D/3D. Nesse cenário, sistemas de vídeo 3D baseados na exibição simultânea de múltiplas vistas também são esperados, incluindo sistemas capazes de lidar com altas e ultra-altas resoluções. Para atender essa demanda e a grande quantidade de dados que precisa ser processada e armazenada, uma extensão do padrão HEVC (*High Efficiency Video Coding*) visando à codificação de vídeos 3D foi desenvolvida por especialistas da ISO/IEC (*International Organization for Standardization / International Electrotechnical Commission*) e da ITU-T (*International Telecommunication Union – Telecommunication*). O HEVC é o estado da arte para codificação de vídeos 2D e sua extensão para 3D é chamada 3D-HEVC. O 3D-HEVC usa o conceito MVD (*Multi-view plus Depth*) em que um mapa de profundidade é associado a cada quadro de textura para cada vista que compõem a sequência de vídeo. Devido a isso, o 3D-HEVC define diversas novas ferramentas de codificação para tornar possível o processamento de vídeos 3D com múltiplas vistas com resoluções crescentes sob essa nova perspectiva. Como um resultado, a extensão 3D-HEVC requer um elevado esforço computacional. Uma vez que dispositivos móveis embarcados capazes de lidar com vídeos 2D/3D requerem estratégias eficientes para gerenciamento de memória/energia e para lidar com os severos requisitos de processamento/memória e fornecimento limitado de energia, o desenvolvimento de sistemas voltados para eficiência em energia e memória visando o 3D-HEVC é essencial.

Esta tese traz contribuições para arquiteturas com alta taxa de processamento e baixa dissipação de potência visando o 3D-HEVC. Estas contribuições estão principalmente centradas em quatro arquiteturas, como segue: (i) um projeto de hardware com baixa dissipação de potência e alta taxa de processamento para a ferramenta de codificação DIS (*Depth Intra Skip*); (ii) um sistema para predição *Intra-frame* de mapas de profundidade com baixa dissipação de potência e voltado para eficiência em memória baseado em estratégias de redução de complexidade; (iii) um sistema para a ME/DE (*Motion/Disparity Estimation*) projetado para baixo consumo de energia, apresentando uma hierarquia de memória adaptável em tempo real; (iv) uma arquitetura para estimação de disparidade com baixa dissipação de potência e eficiência em codificação baseada no algoritmo proposto iUDS (*Improved*

Unidirectional Disparity-Search). As contribuições destas arquiteturas para o estado da arte são confirmadas pelas publicações feitas até o momento em conferências e revistas qualificadas.

Palavras-chave: Codificação de Vídeos. 3D-HEVC. Mapas de Profundidade. Sistema Embarcado de Tempo Real. Projeto de Hardware de Baixo Consumo Energético.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

2D	Two Dimensional
3D	Three Dimensional
3D-HEVC	3D-High Efficiency Video Coding
3D-HTM	3D-HEVC Test Model
AI	All Intra
ASIC	Application Specific Integrated Circuit
ASW	Adaptive Search Window
AU	Access Unit
AVC	Advanced Video Coding
BD	Bjontegaard Difference
BD-Rate	Bjontegaard Difference Bit Rate
BD-PSNR	Bjontegaard Difference Peak Signal-to-Noise Ratio
B-Encoder	Baseline Encoder
BMA	Block Matching Algorithm
BMH	Base Memory Hierarchy
BSM	Binary Segmentation Mask
BV	Base View
CABAC	Context-Based Adaptive Binary Arithmetic Coding
CAVLC	Context-Based Adaptive Variable Length Coding
CCO	Conventional Coding Order
CPV	Constant Partition Value
CPU	Central Processing Unit
CTC	Common Test Conditions of 3DV Core Experiments
CTU	Coding Tree Unit
CU	Coding Unit
dB	decibel
DBBP	Depth-Based Block Partitioning
DBF	Deblocking Filter
DCP	Disparity-Compensated Prediction
DE	Disparity Estimation
DIBR	Depth-image-based Rendering
DIS	Depth Intra Skip

DLT	Depth Lookup Table
DMM	Depth Modeling Modes
DMMFast	Depth Modeling Modes Fast Prediction
DPB	Decoded Picture Buffer
DSWR	Depth-Based Dynamic Search Window Resizing
DV	Dependent View
DVS	Dynamic Voltage Scaling
EWS	Explicit Wedgelet Signaling
FCO	Flexible Coding Order
FIR	Finite Impulse Response
FME	Fractional Motion Estimation
FPGA	Field Programmable Gate Array
fps	frames per second
FS	Full Search
FTV	Free Viewpoint Television
GoP	Group of Pictures
GPB	Generalized P and B Picture
GPU	Graphics Processing Unit
HC	Hardware-oriented Constraint
HD	High Definition
HDS	Horizontal Disparity Search
HEVC	High Efficiency Video Coding
HEVC SC	HEVC Simulcast
HM	HEVC Test Model
HOTZS	Hardware-oriented Test Zone Search
ICPCP	Inter-component-predicted Contour Partitioning
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IME	Integer Motion Estimation
IO	Intra Only
IPH	Horizontal Intra Prediction
IPV	Vertical Intra Prediction
ISDC	Inter-mode segment-wise DC coding
ISO	International Organization for Standardization

ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
iUDS	Improved Unidirectional Disparity Search
JCT-3V	Joint Collaborative Team on 3D Video Coding Extension Development
JCT-VC	Joint Collaborative Team on Video Coding
LCD	Liquid Crystal Display
MC	Motion Compensation
MCL	Merge Candidate List
MCP	Motion-compensated Prediction
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MSB	Most Significant Bit
MSE	Mean Squared Error
MVC	Multi-view Video Coding
MVD	Multi-view Video plus Depth
MV-HEVC	Multi-view HEVC
PC	Processing Core
PS	Power State
PSNR	Peak Signal-to-Noise Ratio
PU	Prediction Unit
QP	Quantization Parameter
RA	Random Access
RAH	Run-Time Adaptive Hierarchy
RDO	Rate-distortion Optimization
RGB	Red, Green, Blue
RPS	Reference Picture Set
RSH	Reduced-Size Hierarchy
SA	Search Area
SAD	Sum of Absolute Differences
SAO	Sample Adaptive Offset
SATD	Sum of Absolute Transformed Differences
SC	Sleep Circuitry
SCU	Search and Comparison Unit
SD	Single Depth

SDC	Segment-wise DC coding
SDH	Horizontal Single Depth
SDV	Vertical Single Depth
SED	Simplified Edge Detection
SR	Search Range
SRAM	Static Random Access Memory
SSD	Sum of Squared Differences
SVDC	Synthesized View Distortion Change
SW	Search Window
TMVI	Texture-Based Motion Vector Inheritance
TQ	Transform/Quantization
TV	Television
TU	Transform Unit
TZS	Test Zone Search
UDS	Unidirectional Disparity Search
UHD	Ultra High Definition
VCEG	Video Coding Experts Group
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration
VSD	View Synthesis Distortion
VSO	View Synthesis Optimization
YCbCr	Luminance, Chrominance blue, Chrominance red

SUMMARY

1 INTRODUCTION

Three-dimensional (3D) videos allow the spectators to perceive the distance (or depth) of the objects in a scene, with a more realistic representation than using the conventional 2D (two dimensional) videos.

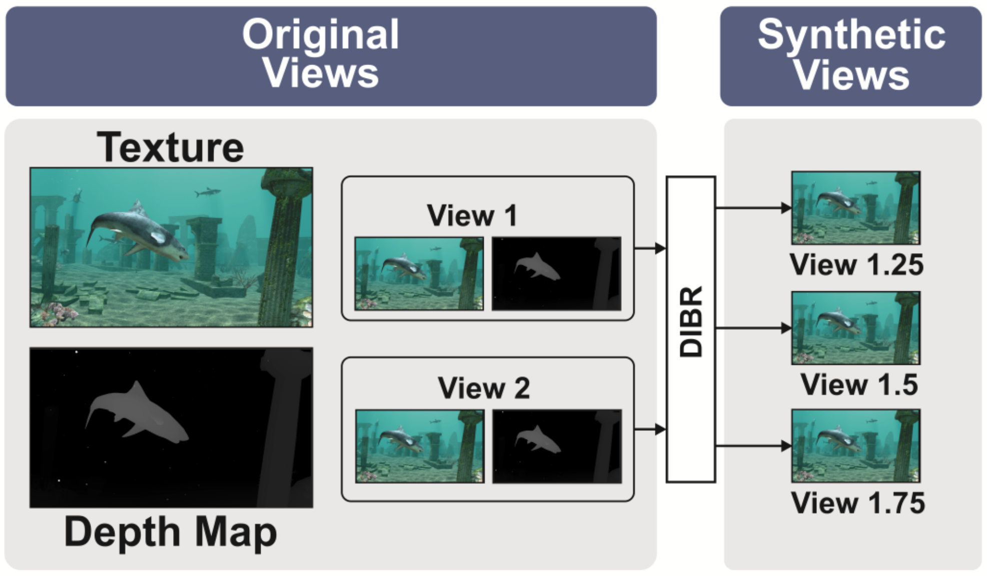
In the last decade, the industry invested in the design and manufacturing of electronic devices with support to record and reproduce 3D videos. In 2009 the success of the 3D movie Avatar in the cinema (SONY, 2019) was a strong booster for this investment. At that time 3D TVs, 3D Blu-Ray players, and portable devices, such as smartphones, gaming consoles and camcorders (TOSHIBA, 2019) (LG, 2019) (NINTENDO, 2019) (PANASONIC, 2019) – usually based on a stereoscopic approach – were heavily commercialized, allowing people to have access to 3D technology in their daily lives and their homes. At a first moment, it seemed that 3D videos and their related products would be finally accepted by the consumers. However, over the years, the shortage of high-quality 3D-video content, and some limitations presented by the technology, such as the huge amount of data to be transmitted, limited view synthesis performance and the need for a pair of glasses (major part of technologies), led the sales to a slowing down due to the visual discomfort and poor user experience (PEREIRA *et al.*, 2016). Despite the effort to provide an immersive experience, the first generation of devices capable of dealing with 3D videos has failed, and the interruption of 3D-TVs production by the major manufacturers occurred at the end of 2015 (LIFEWIRE, 2019).

In contrast to the manufacturing interruption of 3D-TVs production, the 3D cinema remains to be a widely accepted application for 3D videos. Also, there is a massive popularization of multimedia services and video-capable mobile devices in the last years. This popularization is pushed by the concern of the users in high-quality and immersive video-related experiences, and by the platforms with support for video sharing, such as social networks (Facebook, Instagram, etc.), messaging platforms (WhatsApp, Facebook Messenger, Viber, etc.), and video-streaming services (Netflix, YouTube, Amazon Prime Video, etc.). In 2016, the data traffic related to mobile video on the internet accounted for 60% of total mobile data traffic (CISCO, 2019). Cisco (2019) forecasts that by the year 2021 more than three-fourths of the total mobile data traffic will be video, accounting for 78% of total mobile data traffic. Concurrently, handheld devices, such as smartphones and tablets, are advancing along with the emerging of new high-quality and immersive video technologies, such as 3D-movies, living games, virtual reality, augmented reality, mixed reality, among other technologies.

As an alternative to some of the problems presented by the previous 3D-video technology, the MVD (Multi-View plus Depth) concept has emerged along with 3D-HEVC (3D- High Efficiency Video Coding) (TECH *et al.*, 2016) – the 3D extension of the HEVC (High Efficiency Video Coding) standard (SULLIVAN *et al.*, 2012). Due to the huge amount of data to be processed, stored, and transmitted, the processing of digital videos with increased resolutions considering 3D videos is unfeasible without the use of novel and efficient compression techniques in addition to the ones normally applied in the HEVC and in other previous standards. The HEVC was published in April 2013 (ITU-T, 2015), initially developed to encode 2D videos. The 3D-HEVC development was finished in February 2015 (ITU-T, 2015), with the main purpose of making feasible efficient 3D-video storage and transmission for multi-view systems, such as Free-viewpoint Television (FTV) and Three-dimensional Television (3D-TV), besides supporting emerging technologies such as multi-view auto-stereoscopic displays that do not require the usage of glasses (DIMENCO, 2019).

The MVD format introduces the concept of depth maps, which are typically represented as grayscale pictures that depict the distance between each point in the scene and the camera (KAUFF *et al.*, 2007). The objects farthest from the camera are represented by darker shades of gray while objects (or part of objects) nearest to the camera are represented by light shades of gray. For each video view, there is a depth map in addition to the texture picture (pictures conventionally showed to the spectators that contain the brightness and color information), both of which are used to synthesize additional virtual views through DIBR (Depth Image-Based Rendering) techniques (KAUFF *et al.*, 2007). Thus, considering an MVD video system, the transmitter encodes and sends a reduced set of views in comparison to previous texture-only approaches, such as the MVC (Multi-view Video Coding) (ITU-T, 2008), for obtaining the same number of views after the decoding process, since the receiver decodes the transmitted views and synthesizes virtual views when necessary. Figure 1.1 shows an example of two texture pictures and their respective depth maps, considering a system with two original views, and the processing to generate three synthetic views from them using DIBR.

Figure 1.1 – Example of texture pictures and their respective depth maps considering a system with two original and three synthetic views

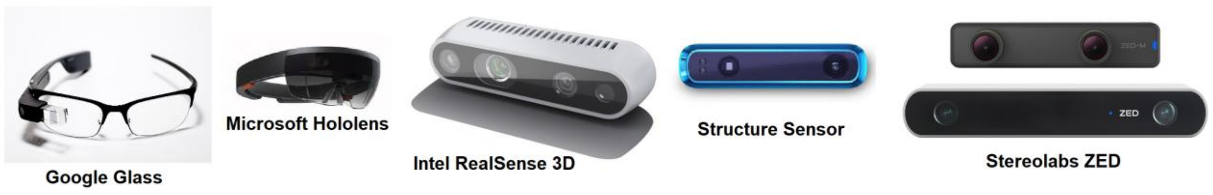


Source: Author.

The pictures that compose the depth maps present well-defined characteristics, such as large homogeneous regions and sharp edges. This way, the encoding tools normally applied in HEVC, and in other previous video coding standards, are not efficient enough to deal with depth-map encoding. 3D-HEVC also allows exploiting the redundancies between different views through the technique called Disparity Estimation (DE), which does not occur in HEVC. Therefore, the new coding tools introduced by 3D-HEVC to efficiently encode 3D-videos along with an MVD content result in a higher complexity when compared to the HEVC (TECH *et al.*, 2016). It is important to mention that the HEVC already has a high complexity when compared with previous standards (CORRÊA *et al.*, 2012).

Meeting real-time and energy constraints while sustaining high coding efficiency becomes more challenging when considering mobile battery-powered devices. While texture-only multiple view handheld camcorders are available in the market since 2010 (PANASONIC, 2019), mobile devices able to capture MVD content are expected to be widely available shortly. The popularization of handheld devices capable of dealing with 3D content is expected driven by emerging acquisition and displaying technologies like Google Glass (GOOGLE GLASS, 2019), Microsoft HoloLens (MICROSOFT HOLOLENS, 2019), Intel RealSense 3D (INTEL REALSENSE, 2019), Structure Sensor (STRUCTURE SENSOR, 2019), and Stereolabs ZED (STEREOLABS ZED, 2019). Figure 1.2 shows examples of electronic devices based on emerging acquisition and displaying technologies.

Figure 1.2 – Examples of electronic devices based on emerging acquisition and displaying technologies



Source: (GOOGLE GLASS, 2019) (MICROSOFT HOLOLENS, 2019) (INTEL REALSENSE, 2019) (STRUCTURE SENSOR, 2019) (STEREOLABS ZED, 2019) Modified by the author.

These emerging video technologies tend to be increasingly integrated with the handheld devices, which face severe constraints in terms of performance, storage, bandwidth and, mainly, energy consumption since these devices are battery powered. Thereby, dedicated VLSI design focusing on video coding is mandatory to provide high-throughput and energy efficiency but becomes more challenging when considering handheld devices performing real-time encoding along with emerging video technologies and 3D-HEVC. The memory and computational effort required for the novel 3D-HEVC prediction tools evidence the need for the development of complexity-reduction strategies and VLSI designs, which are the focus of this thesis. The main challenges regarding memory, processing, and complexity for the 3D-HEVC are discussed with more details in Section 6.1.

1.1 Goals of this Thesis

This thesis presents dedicated algorithms and architectures focusing on the 3D-HEVC tools.

The **major goal** of this doctorate was to improve the energy efficiency of the most energy demanding 3D-HEVC encoding tools to make feasible the real-time processing of high and ultra-high definition 3D-videos focusing on battery-powered applications. This goal was reached through the exploration of memory/processing aspects, as well as the characteristics of the encoding tools under an MVD approach.

The specific goals of this work are listed below and they guided the insights and ideas exploited in the proposed algorithms and in the designed architectures.

Goal 1: Identify the most time demanding encoding tools according to the 3D-HEVC for both texture and depth maps to assist the understanding of its energy requirements.

Goal 2: Identify the most selected and representative encoding tools during a 3D-HEVC-based coding process to discover which tools have the most impact regarding compression and image quality.

Goal 3: Evaluate the impact regarding compression by constraining specific encoding tools as well as the block sizes supported by them during a 3D-HEVC-based coding process.

Goal 4: Analyze, at run time, on-/off-chip memory access behaviors related to 3D-HEVC encoding process to define memory management and hardware design methodologies capable of attempting the requirements regarding throughput and energy consumption.

Goal 5: Take advantage of application-specific knowledge of 3D-HEVC (i.e., its new coding tools) and video content properties to develop hardware-oriented algorithms capable of increasing the throughput and improving the energy efficiency.

Goal 6: Use the developed hardware-oriented algorithms and the clock gating technique to develop energy-efficient video memory and processing architectures.

Goal 7: Exploit a flexible coding order between texture and depth maps to propose heuristics and memory management capable of reducing the energy consumption required for the developed energy-efficient video memory and processing architectures.

1.2 Contributions of this Thesis

Figure 1.3 summarizes the main contributions of this thesis, including both algorithmic and architectural levels. The proposal of algorithms and architectures was based on exhaustive evaluations using the 3D-HEVC reference software and this extensive evaluation, itself, is the first contribution of this thesis.

As presented in Figure 1.3, four architectures/systems exploiting memory/processing aspects were proposed, two of them focusing on the Intra-frame prediction tools and other two focusing on the Inter prediction tools, the latter including both Inter-frames and Inter-view predictions. All the developed architectures take advantage of application-specific knowledge of 3D-HEVC along with the MVD approach, i.e., knowledge of its new coding tools and video content properties.

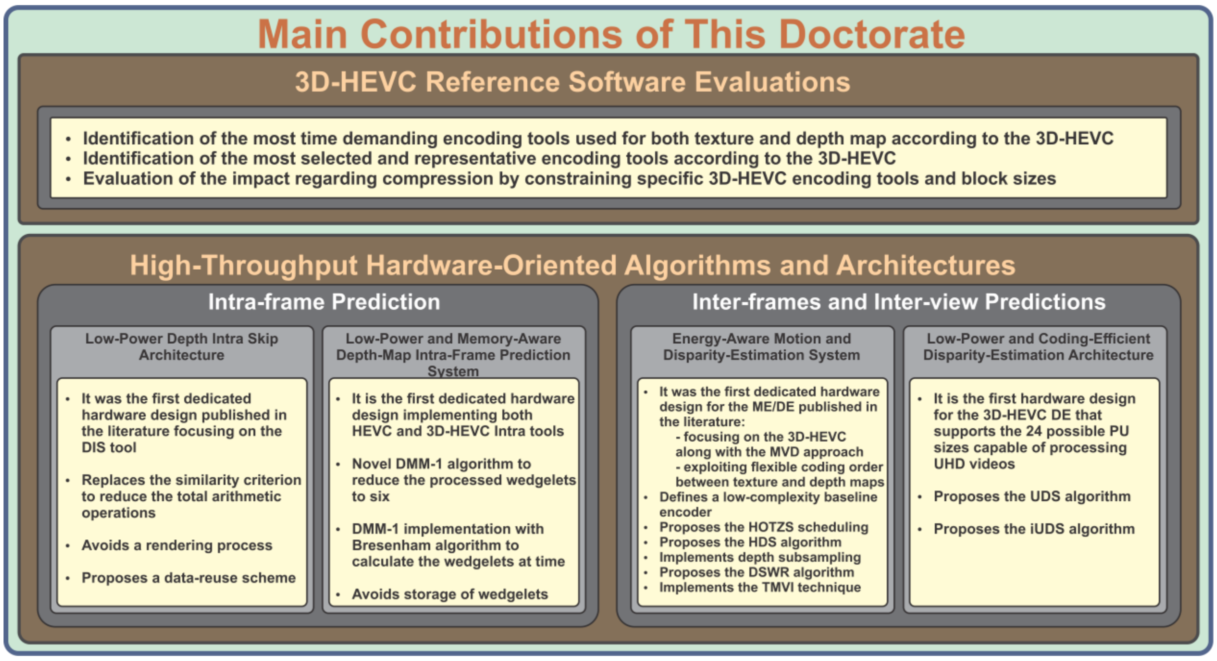
The first development contribution of this work is a low-power Depth Intra Skip (DIS) architecture that was the first dedicated hardware design published in the literature for the DIS tool, and it is capable of processing five UHD 2160p views at 60 frames per second. This architecture used a simpler similarity criterion in order to reduce the computational effort. This strategy reduced the number of arithmetic operations and avoided a rendering process. Data-reuse schemes were also proposed in order to reduce energy consumption. This architecture is detailed in section 7.1.

The second development contribution presented in this thesis is a low-power and memory-aware depth-map Intra-frame prediction system which is also the first published solution supporting both the novel 3D-HEVC and the conventional HEVC intra prediction tools. This architecture is capable of processing nine HD 1080p views at 30 fps. Hardware-oriented heuristics that consists of removing the less important modes and block sizes were proposed to reduce the computational effort. Also, a specific algorithm was applied to the Depth Modeling Mode–1 (DMM-1) to reduce the computational effort related to this tool. Section 7.2 presents this architecture.

The third development contribution presented in this work is an energy-aware Motion and Disparity Estimation (ME/DE) system that was the first hardware design focusing on the 3D-HEVC (with the MVD approach) published in the literature. This design is capable of processing three HD 1080p views at 30 frames per second. Also, this system was the first to take advantage of a flexible coding order between texture and depth maps to reduce energy consumption related to the 3D-HEVC ME/DE encoding tools. The architecture was designed for low-energy consumption, featuring a run-time adaptive memory hierarchy. Several heuristics and memory management capable of reducing the energy consumption were proposed. These and the other techniques proposed in this work are described with more details in section 8.1.

The last development contribution presented in this thesis is a low-power and coding-efficient Disparity Estimation (DE) architecture which is the first published hardware design focusing on 3D-HEVC that supports 24 possible PU sizes capable of processing five UHD 2160p views at 40 frames per second. Two low-complexity algorithms for the disparity estimation were proposed, prioritizing horizontal search instead of using conventional search algorithms in two dimensions. Section 8.2 presents this architecture.

Figure 1.3 – Main contributions of this doctorate



Main Contributions of This Doctorate

3D-HEVC Reference Software Evaluations

- Identification of the most time demanding encoding tools used for both texture and depth map according to the 3D-HEVC
- Identification of the most selected and representative encoding tools according to the 3D-HEVC
- Evaluation of the impact regarding compression by constraining specific 3D-HEVC encoding tools and block sizes

High-Throughput Hardware-Oriented Algorithms and Architectures

Intra-frame Prediction

Low-Power Depth Intra Skip Architecture

- It was the first dedicated hardware design published in the literature focusing on the DIS tool
- Replaces the similarity criterion to reduce the total arithmetic operations
- Avoids a rendering process
- Proposes a data-reuse scheme

Low-Power and Memory-Aware Depth-Map Intra-Frame Prediction System

- It is the first dedicated hardware design implementing both HEVC and 3D-HEVC Intra tools
- Novel DMM-1 algorithm to reduce the processed wedgelets to six
- DMM-1 implementation with Bresenham algorithm to calculate the wedgelets at time
- Avoids storage of wedgelets

Inter-frames and Inter-view Predictions

Energy-Aware Motion and Disparity-Estimation System

- It was the first dedicated hardware design for the ME/DE published in the literature:
  - focusing on the 3D-HEVC along with the MVD approach
  - exploiting flexible coding order between texture and depth maps
- Defines a low-complexity baseline encoder
- Proposes the HOTZS scheduling
- Proposes the HDS algorithm
- Implements depth subsampling
- Proposes the DSWR algorithm
- Implements the TMVI technique

Low-Power and Coding-Efficient Disparity-Estimation Architecture

- It is the first hardware design for the 3D-HEVC DE that supports the 24 possible PU sizes capable of processing UHD videos
- Proposes the UDS algorithm
- Proposes the iUDS algorithm

Source: Author.

1.3 Text Organization

This thesis is organized into nine chapters, as follows:

Chapter 2 brings the basics related to video coding applications. Some general and fundamental concepts like lossy and lossless compression, color space and subsampling, quality assessment in digital videos, and an overview about the current 3D video coding standards are presented. This revision includes the 3D-HEVC extension, state-of-the-art regarding 3D-video coding and the focus of the algorithms and architectures developed in this work.

Chapter 3 presents an HEVC revision in order to give an overview about the block partition and temporal prediction structures employed by the HEVC standard as well as its most important encoding tools.

Chapter 4 further details the 3D-HEVC extension. First, a historical on the 3D-HEVC extension development is presented. In the sequence, the coding structures of the 3D extension, as well as its novel encoding tools, are presented.

Chapter 5 presents the main ideas of state-of-the-art works regarding the development of algorithms and dedicated hardware architectures focusing on 3D-video coding. The works focusing on the development of energy-efficient systems and focusing on the 3D-HEVC are prioritized for such discussion. Promising works focusing on previous 2D/3D video coding standards are also presented.

Chapter 6 shows detailed 3D-HEVC reference software evaluations to motivate the algorithms and dedicated architectures developed during this work. The main goal is to have a better understanding of the 3D-HEVC encoding-tools behavior to support the ideas developed in this work.

Chapter 7 presents the two dedicated architectures designed during this doctorate focusing on the 3D-HEVC Intra-frame prediction. The first one is a low-power and high-throughput hardware design for the DIS coding tool. The second architecture is a low-power and memory-aware depth-map Intra-frame prediction system.

Chapter 8 presents the two dedicated architectures designed during this doctorate focusing on the 3D-HEVC Inter-frames and Inter-view predictions. The first architecture is a low-power and coding-efficient Disparity Estimation architecture based on the developed iUDS (Improved Unidirectional Disparity-Search) algorithm. The second one is a ME/DE system designed for low-energy consumption, featuring a run-time adaptive memory hierarchy using the developed DSWR (Depth-Based Dynamic Search Window Resizing) algorithm.

Chapter 9 concludes this thesis by presenting the final remarks. All contributions are summarized, and the initially defined goals are discussed, based on the achieved results. As reflexive analysis from the doctorate path, future research perspectives are presented.

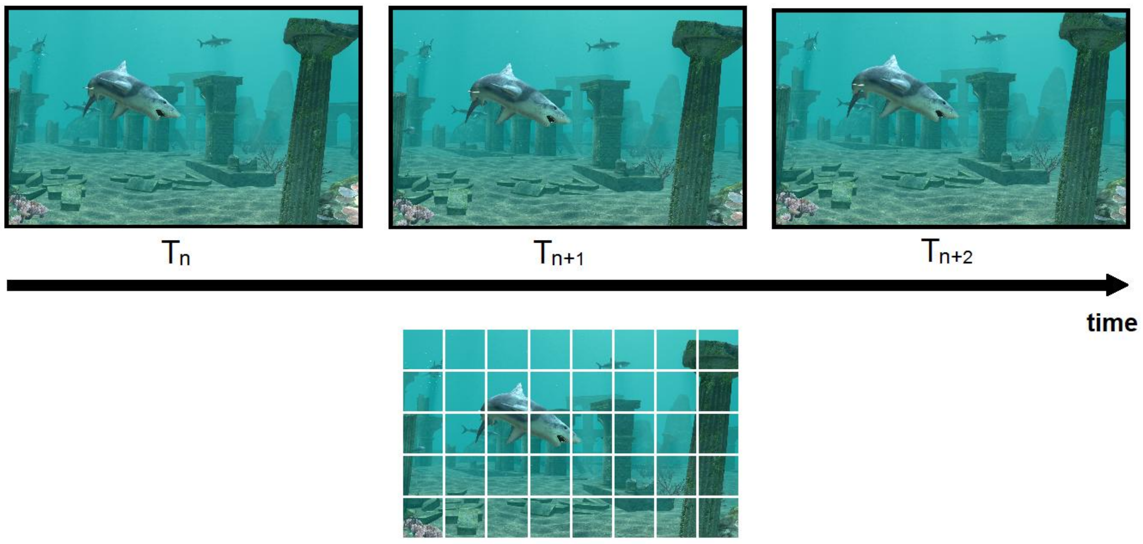
2 CONCEPTUAL REVIEW OF 3D-VIDEO ENCODING

This Chapter brings the basics related to video coding applications focusing on 3D-video coding. Besides some general and fundamental concepts, a revision about 3D technology is also presented.

2.1 Fundamental Concepts about Digital Videos

A digital video consists of a sequence of independent pictures, captured with a specific time interval. These pictures are commonly called frames, and frames are composed of pixels. Pixels are physical points that compose the image, generally using three samples that correspond to the brightness and color information. These samples can change according to the system used to represent the colors, i.e., according to the Color Space (OHM, 2015). The frames can also be divided into blocks by the video encoders. These blocks can present different sizes according to the video-coding standard and, for the current standards, the block size varies during the coding process in order to improve the efficiency of the encoding tools. Figure 2.1 presents a sequence of temporal-neighboring frames where one of them is divided into blocks. The representation of a scene with 2D digital videos is based on two types of sampling: (i) spatial and (ii) temporal. The spatial sampling uses a matrix of pixels defined as video resolution, which is responsible for delimiting the frame size. Considering that the samples of two different videos, recorded under the same conditions, have the same bit wide, the quality of the video is better the bigger is the video resolution since a higher number of pixels is used to represent the same image. The temporal sampling is related to the time interval used to capture the images that compose the video sequence. The higher the rate used to capture the images, i.e., the higher the temporal sampling, the better is the perception of movement. However, this improvement on the perception of movement reaches a limit according to the capacity of the human visual system. In general, the literature indicates a minimal rate from 24 to 30 frames per second in the image capturing to guarantee the sensation of movement to the spectators (GONZALES, 2003). However, high and ultra-high resolutions, such as UHD 2160p, require a minimum of 50 to 60 frames per second to allow a proper viewing (GOLDMAN *et al.*, 2015).

Figure 2.1 – Sequence of temporal neighboring frames and a frame divided into blocks

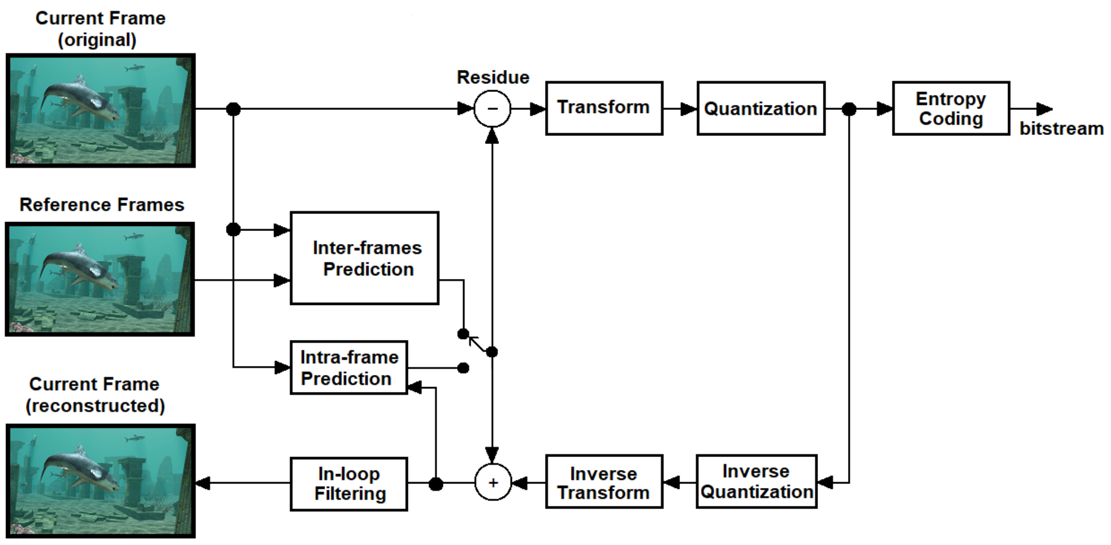


Source: Author.

2.2 Generic Video Encoder Scheme and Data Redundancies

Figure 2.2 shows a block diagram of a generic 2D-video encoder (compliant with current video standards, such as the HEVC and H.264/AVC) aiming at identifying the main type of video redundancies explored by each coding tool of a video encoder.

Figure 2.2 – Generic 2D-video encoder model



Source: Author.

The Inter-frames prediction is responsible for exploiting the temporal redundancies. Temporal redundancies are those similarities that appear in neighboring frames in a video. These frames tend to have many similar or even identical regions. The Inter-frames prediction is composed of two main steps: the Motion Estimation (ME) and the Motion

Compensation (MC) (AGOSTINI, 2007). ME is responsible for finding the block previously processed in the reference frames that is most similar with the block being encoded, considering some parameters to constrain the search to limit the amount of evaluate blocks and reduces the computational effort. MC is used to reconstruct the frames estimated from the data generated by the ME, allowing the residue generation (the subtraction between the current block and the predicted block). By using the ME and MC, only the residues and the relative localization (motion vector) of the block selected in the reference frame are transmitted.

The Intra-frame prediction exploits the spatial redundancies presented in a video. Considering the same frame of a video, most of neighboring pixels tend to present similar values (or even similar behavior when the values smoothly change in a given direction).

A control step in the encoder is responsible for choosing the prediction tool that will be used to encode each block according to video characteristics intending to maximize the relation between compression and image quality.

After the prediction, a subtraction operation is performed between each original image block and the respective predicted block to obtain the difference between these blocks, which is called residue.

The residue passes through a Transform step that transforms the information from the spatial domain to the frequency domain. This way, an additional step called Quantization eliminates those frequencies less relevant to the human visual system. This operation introduces losses in terms of image quality (AGOSTINI, 2007). Nevertheless, these losses can be controlled by the encoder optimizing the relation between the image quality and the compression rate. The Quantization Parameter (QP) defines the used quantization level and as higher is the QP value, as higher is the image compression and as lower is the image quality. Then, the encoder can define an operation point where the image degradations are imperceptible. Transform and Quantization operations together also contribute to reduce the spatial redundancies.

The Entropy Encoding is applied after the Quantization to change the representation of the symbols using the variable-length code. Thus, the symbols with higher occurrences are represented with a smaller bit width. On the contrary, symbols with lower occurrences are represented with a higher bit width. Then, this tool is responsible for reducing the entropic redundancy, the third redundancy presented in digital videos. This redundancy is directly related with the number of bits necessary to represent the video information. The HEVC main tool used in the Entropy Encoding is the CABAC (Context-Based Adaptive Binary Arithmetic Coding) (McCANN *et al.*, 2014).

The encoder also has the Inverse Transform and Inverse Quantization steps that are used in the generation of the reconstructed image blocks. The reconstructed image blocks compose the reconstructed frame, which allows the next frames to be encoded and guarantees that both the encoder and the decoder use the same references since the quantization generates losses in the encoding process (AGOSTINI, 2007).

Finally, the In-loop filtering step is used to reduce the block effects introduced due to the block-based coding, differing a real image edge from an artifact generated by the Quantization. Two filters are used in HEVC before the reconstruction of samples: (i) the Deblocking Filter and (ii) the SAO (Sample Adaptive Offset). These filters increase the subjective image quality (McCANN *et al.*, 2014).

2.3 Lossy and Lossless Compression

The processing of digital videos involves the manipulation of a huge amount of data, which would make infeasible this kind of media considering real-time applications without the use of compression techniques.

Considering that these compression techniques can introduce data losses in the encoding process, the compression methods are classified into: (i) lossy; or (ii) lossless (RICHARDSON, 2002). When the lossless compression method is applied, the decompressed data are identical to the original data before the compression process. However, the lossless compression method reaches reduced compression rates when applied to digital videos. Since both image and video compression require higher compression rates due to the huge amount of data transmitted and stored, the lossy compression method is the most adequate method (RICHARDSON, 2003). By using lossy compression methods, the H.264/AVC standard reaches compression rates of 50 times, while maintaining the image quality perception similar to the original video (AGOSTINI, 2007) while HEVC reduces twice the bit rate when compared to H.264/AVC for a same image quality (SULLIVAN *et al.*, 2012).

2.4 Color Space and Subsampling

The color space consists of a system able to represent the colors in an image (OHM, 2015). The color space YCbCr is composed of three components: (i) Luminance, denoted by Y; (ii) Chrominance blue, denoted by Cb; and (iii) Chrominance red, denoted by Cr (OHM, 2015). This color space is one of the main systems used in video coding (OHM, 2015) since the brightness information (luminance channel) is separated from the color information (chrominance channels) and then, the different channels can be independently processed without losses in the visual quality. Other well-known color spaces, such as the RGB (Red, Green, and Blue), do not have such characteristic. This way, the encoders take advantage of the human visual system which is more sensitive to the brightness information than the color information to reduce the amount of data used to represent the image (OHM, 2015). For example, the video encoders apply a technique called color subsampling that consists on reducing the spatial sampling of the chrominance information when compared to the luminance, increasing the coding efficiency (OHM, 2015).

The color subsampling can be applied with different spatial subsampling, which defines the color subsampling formats 4:4:4, 4:2:2, and 4:2:0. In the format 4:4:4, there are four samples of Cb and four samples of Cr for every four samples of Y. The format 4:2:2 considers two samples of Cb and two samples of Cr for every four samples of Y. Finally, the format 4:2:0 uses only one sample of Cb and one sample of Cr for every four samples of Y. The format 4:4:4 maintains the same spatial sampling for both luminance and chrominance samples and, therefore, it does not change the image quality but also it does not provide any compression. By using the format 4:2:0, the image quality decreases but the total video size is reduced to the half without the use of any other compression technique. The quality losses are almost imperceptible for the human visual system even with this dramatic discard of data. By default, the HEVC standard uses the format 4:2:0, but the formats 4:4:4 and 4:2:2 can also be used when configured for that. The HEVC also allows for disabling the use of chrominance samples (monochromatic option) (BROSS *et al.*, 2012).

2.5 Quality Comparison Metrics Used Inside Video Encoders

The video encoders can exploit several techniques in order to obtain higher coding efficiency. These techniques use metrics to compare the original video and the compressed video and, therefore, to define which coding mode must be used for each block being encoded. The coding decision is based on the relation between the reduction of data to be transmitted/stored (measuring the bit rate), and the impact of this compression in the image quality.

Current video-coding standards, such as the HEVC, use the Rate Distortion Optimization (RDO) technique (3D-HEVC Reference Software, 2019) to measure the cost of the different encoding modes when applied to the candidate blocks in the encoding process, and decide which one returns the smallest cost. For obtaining the distortion, RDO can apply different methods between the original and the reconstructed sample values, as will be explained in the following.

The objective criterion PSNR (Peak Signal-to-Noise Ratio) (GHANBARI, 2003), defined in equation (1), is the most used criterion in current video encoders (3D-HEVC Reference Software, 2019). The PSNR can be applied in different levels from a block or frame to a complete video and it is represented in decibels (dB) that indicates the image-quality evaluated in the coding process. The PSNR calculation uses the similarity criterion called MSE (Mean Squared Error), as defined in equation (2). The MSE represents the mean squared error among the pixels of two frames. R and O represent the samples of the reconstructed and the original frames, respectively. The variables m and n determine the frame dimension, and MAX represents the maximum value of the samples. The higher is the PSNR, the better is the objective image quality (RICHARDSON, 2002).

$$PSNR_{dB} = 20 * log_{10}\left(\frac{MAX}{\sqrt{MSE}}\right) \tag{1}$$

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(R_{i,j} - O_{i,j}\right)^2 \tag{2}$$

Besides the MSE, there are several other similarity criterions available in the literature. Among these criterions, the Sum of Absolute Differences (SAD) (SEIDEL *et al.*, 2016) is one of the most important in current video encoders (3D-HEVC Reference Software, 2019). The SAD is a similarity criterion widely known in this community and it is commonly used in the prediction steps of many codec implementations. One advantage of SAD over MSE is that SAD allows an easier hardware implementation than the MSE, where only sums and subtractions are needed. The MSE requires costly processes like division and exponentiation operations. The equation (3) defines the SAD calculation, where R and O represent the samples from the reconstructed and the original frames, respectively. The variables m and n represent the frame dimensions.

$$SAD = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left|R_{i,j} - O_{i,j}\right| \tag{3}$$

Current video encoders (3D-HEVC Reference Software, 2019) also employ a modified version of the SAD criterion, the Sum of Absolute Transformed Differences (SATD) (SEIDEL *et al.*, 2016) that uses a Hadamard transform and imposes a higher computational effort than the SAD criterion. In other words, the SATD criterion delivers better results than the SAD, but the SATD also uses four times more arithmetic operations than the SAD (GRELLERT, 2014). Equation (4) defines the SATD calculation where HT(i,j) represents the application of the Hadamard Transform to the residual block obtained from the reconstructed samples and the original samples. The variables m and n represent the frame dimensions.

$$SATD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |HT_{i,j}| \tag{4}$$

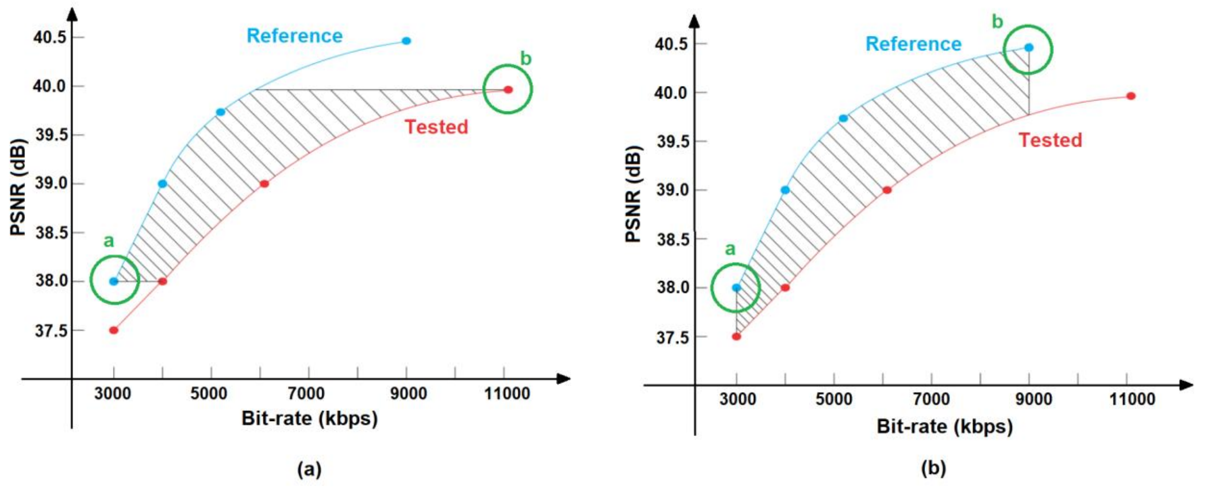
2.6 Coding Efficiency Comparisons among Different Encoder Implementations

There are several metrics in the literature targeting the coding efficiency assessment of digital videos. These metrics uses two well-known main approaches that consist of: (i) a subjective assessment; (ii) an objective assessment. In summary, the metrics based on a subjective assessment take in account the perception of the spectators of both the original and the modified video in a given set of test conditions while the objective approach uses equations to compare the original and the modified video.

Considering the objective metrics, such as the bit rate and the PSNR, the tradeoff between compression and image quality cannot be accurately verified when they are separately analyzed. This way, the metrics BD bit rate (or BD-BR or BD-Rate) and BD Peak Signal-to-Noise Ratio (BD-PSNR), based on the BjØntegaard Difference (BD) (BJΦNTEGAARD, 2001), are widely adopted in the video-coding research area. The BD-Rate measures the percentage variation of the bit rate between two test cases considering encoded videos with the same objective image quality. Positive values of BD-Rate indicate low efficiency on the compression since it means a bit-rate increase to obtain the same image quality. In the BD-PSNR metric, the idea is similar, where the percentage variation of PSNR values (in decibels) between two test cases is evaluated considering the same bit rate. This way, negative values of BD-PSNR indicate an image degradation to obtain the same bit rate. The test cases should be encoded considering the Common Test Conditions (MÜLLER *et al.*, 2014), which are presented in Appendix A.

To obtain the BD-PSNR and the BD-Rate values between two test cases (here called Reference and Tested), these test cases are encoded with four different Quantization Parameters (QPs), which results in eight pairs (PSNR, bit rate). From these eight pairs, two rate-distortion (RD) curves are generated with a third order interpolation function, as presented in Figure 2.3. The area between these two RD curves passes through an integration that considers the y-axis as the reference for BD-Rate calculation (Figure 2.3-a), and considers the x-axis as the reference for BD-PSNR calculation (Figure 2.3-b). Equations (5) and (6) can be used to calculate the BD-Rate and the BD-PSNR, respectively. REF and TEST in (5) and (6) represent PSNR or bit rate values for the curves related to the test case Reference and Tested, respectively. The variables a,b represent the second minimum and the second maximum PSNR values from both curves for the BD-BR calculation, or they represent the second minimum and the second maximum bit-rate values from both curves for the BD-PSNR calculation, respectively.

Figure 2.3 – RD curves used to (a) BD-BR and (b) BD-PSNR calculations



Source: Author.

$$BD_{BR} = \frac{\int_a^b (REF_{PSNR}(y) - TEST_{PSNR}(y))\,dy}{b-a} \tag{5}$$

$$BD_{PSNR} = \frac{\int_a^b (REF_{BR}(x) - TEST_{BR}(x))\,dx}{b-a} \tag{6}$$

2.7 3D Video-Coding Standards

Three-dimensional videos require much more data to be represented than two-dimensional videos since 3D videos are constructed using multiple views, commonly captured from multiple cameras, which represent the same scene at different viewpoints. Thus, a set of 2D videos are encoded and simultaneously transmitted resulting in a considerable increase in the computational effort of the 3D encoder and the bandwidth for 3D-video transmission, when compared to 2D videos. Therefore, new approaches to surpass this drawback and enable real-time 3D-video processing have been investigated in the last years.

The first intuitive solution to encode 3D-videos is a simulcast coding scheme (TECH *et al.*, 2016), where the frames are predicted from frames belonging to the same view, i.e., each view is independently encoded as a 2D video. Although simulcast is less complex than other encoding models that exploit the redundancies between neighboring views, this coding scheme is much less efficient.

On the other hand, the multiple views of the same scene (multi-view) allow applications like stereoscopic and autostereoscopic displaying, Three-Dimensional Television (3D-TV), and Free-viewpoint Television (FTV) that may require a large set of views. Such applications become unfeasible with simulcast approach regarding high resolutions because the 3D encoded video information tends to increase proportionally with the number of views. In this case, the final bitstream is the sum of all bitstreams of each view, which results in a huge amount of data to be transmitted/stored, mainly considering videos at high and ultra-high resolutions.

In order to efficiently exploit the coding process of three-dimensional videos the first standardized solution was the Multiview Video Coding (MVC) (ITU-T, 2008) that was defined as an extension of the H.264/AVC (Advance Video Coding) (ITU-T, 2017). The main idea of MVC was to exploit the redundancies among neighboring views reducing the encoded data. The MVC uses the H.264/AVC tools to encode these multiple videos and explore the redundancies between neighboring views in a new process called Inter-view prediction (ITU-T, 2008). The main tool used in the Inter-view prediction is the Disparity Estimation (DE), which is based on the ME (Motion Estimation) used to encode two-dimensional videos (SULLIVAN *et al.*, 2013). For each block of the current frame in a view, DE searches for a similar block in neighboring views (SULLIVAN *et al.*, 2013). DE application increases the coding efficiency in the multi-view context, and this was the first tool included in video coding standards to explore the specificities of 3D information. According to the Merkle *et al.* (2007), at least 20% of the image blocks that composes the frames are more efficiently encoded by using Inter-view prediction. When compared to simulcast approach using H.264/AVC, the MVC can reduce the data needed for the video representation in about 20-50% (ZATT *et al.*, 2013). However, in MVC, the bit rate increases in a linear proportion with the number of views (MERKLE *et al.*, 2007). Since multi-view applications require many views to guarantee high quality in the viewing experience, the MVC should be improved to handle with this scenario.

The HEVC standard also defined an extension for multi-view tridimensional videos named as MV-HEVC (Multi-view HEVC) (TECH *et al.*, 2016). MV-HEVC follows a similar encoding structure of H.264/AVC MVC but using the new HEVC tools. Then, the MV-HEVC is also able to explore inter-view redundancies, increasing the coding efficiency. The MV-HEVC requires a higher computational effort when compared to MVC because the HEVC encoding tools are much more complex (and efficient) than the H.264/AVC encoding tools. But the scenario with many views is also a problem difficult to be handled for the MV-HEVC.

Aiming at solving this problem, experts of the JCT-3V (Joint Collaborative Team on 3D Video Coding Extension Development) developed a second HEVC extension targeting 3D videos, the 3D-HEVC (TECH *et al.*, 2016). By using the Multi-view plus Depth (MVD) concept (KAUFF *et al.*, 2007), where a depth map is associated with each texture frame, the 3D-HEVC can be considered the state-of-the-art of 3D video coding standard. The use of MVD allows reducing the number of captured and encoded texture views since the texture views together with its depth maps can be used at the decoder side to generate synthesized (or virtual) views, as previously explained.

Since the 3D-HEVC must consider the depth maps together with texture, novel encoding tools were also defined to efficiently deal with this type of information. These tools will be detailed in Chapter 4 including the new encoding modes for Intra-frame prediction: Depth Modeling Modes 1 and 4 (DMM-1 and DMM-4) and Depth Intra Skip (DIS); and an alternative residual flow Segment-wise DC coding (SDC) (TECH *et al.*, 2016). All the original HEVC tools were also inherited and are available to be used in the 3D-HEVC.

Another important point considered in the development of the 3D-HEVC extension is the compatibility with the current technologies besides the technologies in development. In other words, 3D-HEVC is compliant with conventional 2D displays, stereoscopic displays, and autostereoscopic displays with a different number of views. The decoder can select a sub-bitstream according to the technology used in the display.

3 BRIEF REVISION ABOUT THE HEVC

To help the understanding of the 3D-HEVC extension and its specific video-coding tools, this chapter presents some details about the HEVC, including: (i) the block-partition structure, (ii) the temporal prediction structure and (iii) the used video-coding prediction tools. These HEVC features were inherited by the 3D-HEVC.

3.1 Block-Partition Structure

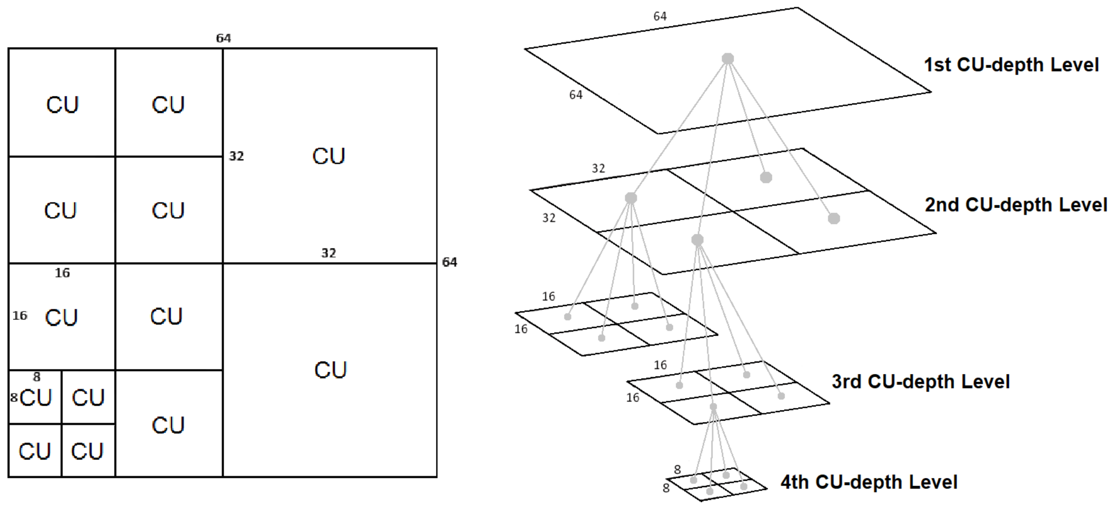
One of the main innovations presented by the HEVC when compared to the H.264/AVC is in the structure of compression. The adopted structure uses a highly flexible scheme of representation with three different concepts of blocks: (i) Coding Units (CUs); (ii) Prediction Units (PUs); and (iii) Transform Units (TUs) (McCANN *et al.*, 2014). This way, each coding tool is employed over a different block structure, allowing more efficiency in the coding process according to the tool.

Video coding standards older than the H.264/AVC used a fixed block size. The H.264/AVC introduces the concept of block partition by adopting a macroblock of 16x16 samples that can be divided into smaller blocks until the minimum size of 4x4 samples. In turn, HEVC enables a maximum block size of 64x64 samples, while the minimum block size was maintained with 4x4 samples. Therefore, the HEVC allows a range of block sizes greater than the H.264/AVC in the predictions, and transforms/quantization steps, as will be better explained in the following.

The coding structure used in the HEVC initially allows the partition of the frames into square-shaped blocks called Coding Tree Units (CTUs) (McCANN *et al.*, 2014). The CTU size can be configured. However, the maximum block size used to the coding process must be smaller or equal to 64x64 samples. The CTU is composed of one luminance block along with two chrominance blocks. The concept of CTU is analog with the concept of a macroblock in H.264/AVC.

So, on each CTU, the highly flexible scheme of representation previously mentioned is implemented. Each CTU is composed of one or more CUs. CUs always are square-shaped blocks with size 2Nx2N, where N can assume the values 4, 8, 16, and 32. Each CU can be recursively divided into four blocks of the same size considering up to four CU-depth levels from the CTU size. Hence, these divisions form a quaternary tree coding structure (quadtree). Figure 3.1 shows a CTU divided into CUs and its respective quaternary tree.

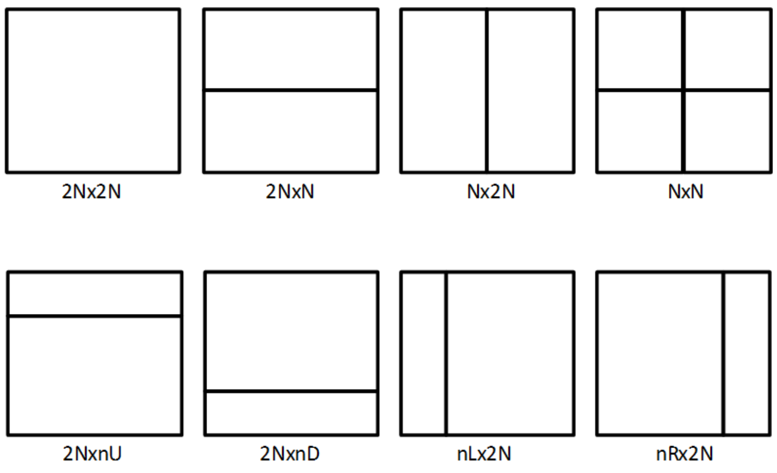
Figure 3.1 – Example of a CTU divided into CUs and its respective quaternary tree



Source: Author.

Each CU can have one or more PUs. The PU is the base block unit used for the Inter-frames and Intra-frame predictions. Unlike the CUs, PUs allow rectangular-shaped blocks. Hence, the blocks can be partitioned according to the real limits of the objects that compose the image. Figure 3.2 shows all the different ways that a CU can be divided into PUs. The values of N can be the same values previously presented for the CU sizes (4, 8, 16, and 32). The division NxN is only enabled in the Intra-frame prediction and when N is equal to 4. In the Intra-frame prediction, only NxN and 2Nx2N partitions can be used, while the Inter-frames prediction can also use symmetric partitions (2NxN, and Nx2N) and asymmetric partitions (2NxnU, 2NxnD, nLx2N and nRx2N). Asymmetric partitions are available only if N is bigger than four.

Figure 3.2 – Types of partition of a CU in PUs



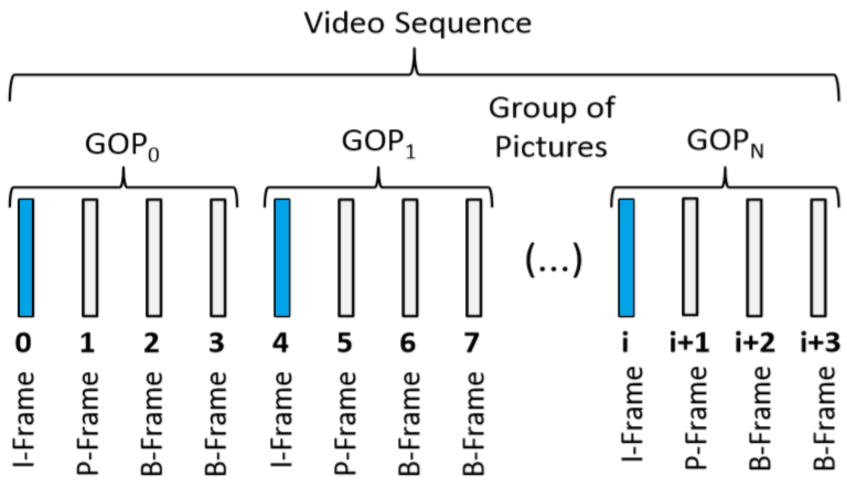
Source: McCann *et al.* (2014, p. 14).

The TU is the base block unit used to the processes of Transform and Quantization. TUs are always square-shaped blocks of size NxN, where N can have the same sizes applied to the CUs (4, 8, 16, and 32). Each CU can have one or more TUs so that the TUs can also be disposed of a quaternary tree structure such as the CUs.

As above mentioned, the HEVC allows the partition of frames into blocks. However, the HEVC also allows the division of the video sequence into other hierarchy levels. In HEVC, Group of Pictures (GoP) are composed of frames, frames are divided into CTUs, CTUs are partitioned into CUs and, finally, the CUs are partitioned into PUs and TUs intending to reach the best coding efficiency. One GoP uses as references only frames inside the same GoP.

Figure 3.3 illustrates a video sequence divided into GoPs of size four. The concepts of I, P and B-frames are presented in the next subsection.

Figure 3.3 – Example of a video sequence divided into GoPs



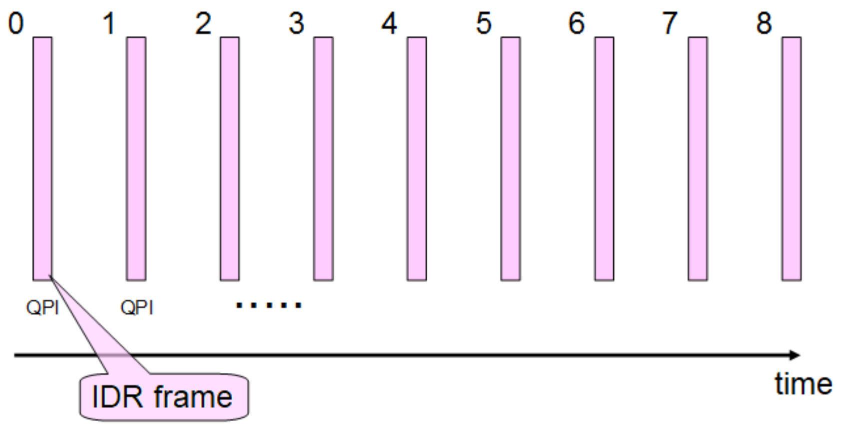
Source: Author.

3.2 Temporal Prediction Structure

The HEVC Reference Software (HM – HEVC Test Model) can operate under three different temporal prediction structures according to the experimental conditions, such defined in the document JCT-VC L1100 (BOSSEN, 2013). These structures are called intra-only (IO – or All Intra depending on the JCT-VC document), low-delay (LD) and random-access (RA). HEVC extensions, such as the 3D-HEVC, only can operate under RA and IO temporal structures, then, only these two structures will be detailed in this text.

When operating under the test configuration IO, each frame that composes the video sequence is encoded as an IDR (Instantaneous Decoding Refresh). IDRs are frames predicted using only the Intra-frame tool. This way, no reference frames are used in IO temporal structure. The QP value remains the same during the encoding process. Figure 3.4 represents the IO configuration, where the number associated with each frame represents the encoding order.

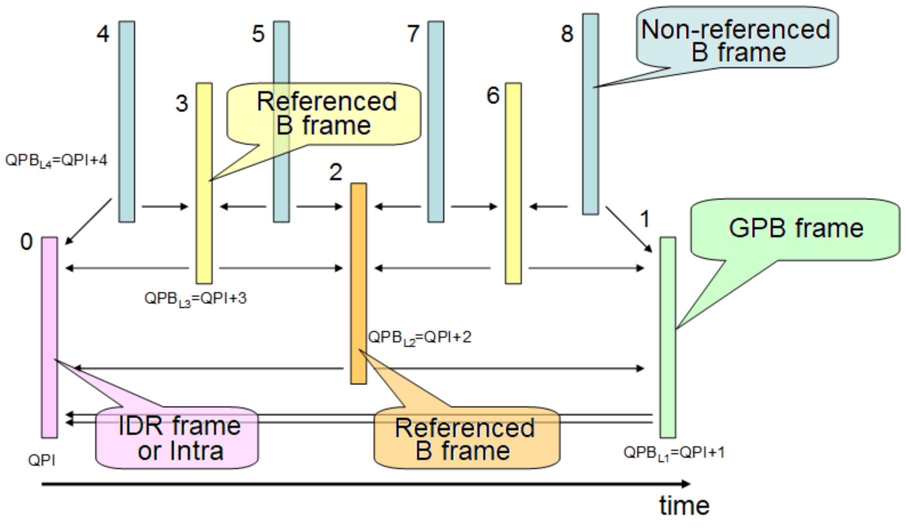
Figure 3.4 – IO temporal configuration



Source: (McCANN *et al.*, 2014).

Considering the RA configuration, a hierarchy structure composed of bi-predicted frames (B frames) is used in the encoding process. B frames use two reference frames together to generate the prediction. One IDR frame is inserted every second approximately. The frames located between two IDR frames in the viewing order are encoded as B frames. The Generalized P and B pictures (GPB) are used in the lowest level of the temporal layer. GPB frames use only temporally previous frames in relation to the current frame. The second and the third temporal layers consist of referenced B frames whereas the highest level of the temporal layer contains non-referenced B frames. The QP value related to each Inter-coded frame is obtaining from an off-set summed to the QP of the Intra frame. This off-set depends on the temporal layer. The viewing order in the RA configuration is not the same used in the encoding process. Figure 3.5 shows the RA configurations and the number associated with each frame represents the coding order.

Figure 3.5 – RA Temporal Configuration



Source: (McCANN *et al.*, 2014).

3.3 Encoding Tools

In terms of coding tools, HEVC introduces some novelties when compared to the H.264/AVC. The following subsections discuss these novelties focusing on both Intra-frame and Inter-frames predictions which are the focus of the contributions presented in this thesis.

3.3.1 Intra-frame Prediction

In the Intra-frame prediction, the HEVC maintained the DC mode and the eight directional modes used in the H.264/AVC, but introduced the Planar mode and other 25 new directional modes. Therefore, the total number of modes increased from nine in the H.264/AVC to 35 in the HEVC. As previously explained, the Intra-frame prediction is responsible for exploiting the spatial redundancies presented in neighboring regions inside the same frame.

The next subsections explain the intra-prediction encoding tools.

3.3.1.1 Planar Mode (Mode 0)

The Planar prediction mode is efficient to predict regions with smooth textures or complex textures that are not efficiently predicted by the angular modes (LAINEMA *et al.*, 2014), and it results in predicted blocks with surfaces without discontinuities in the edges.

The Planar mode is calculated through the average of the results provided from the interpolation using two filters, called here: (i) Horizontal prediction (P_h), defined by equation (7); and (ii) Vertical prediction (P_v), defined by equation (8). These filters calculate their values from the multiplication of reference samples belonging to the adjacent blocks, located above and to the left of the block being encoded. Equation (9) is used to calculate the final value of the Planar mode for each position.

The value of N in equations (7)-(9) represents the size of the block to be predicted, where x and y ϵ 0, …, N-1, while Lat[x] and Lat[N] are reference samples located to the left of the current block, Sup[y] and Sup[N] are samples located above to the current block and $P_h[x][y]$ and $P_v[x][y]$ are the samples generated by the filters.

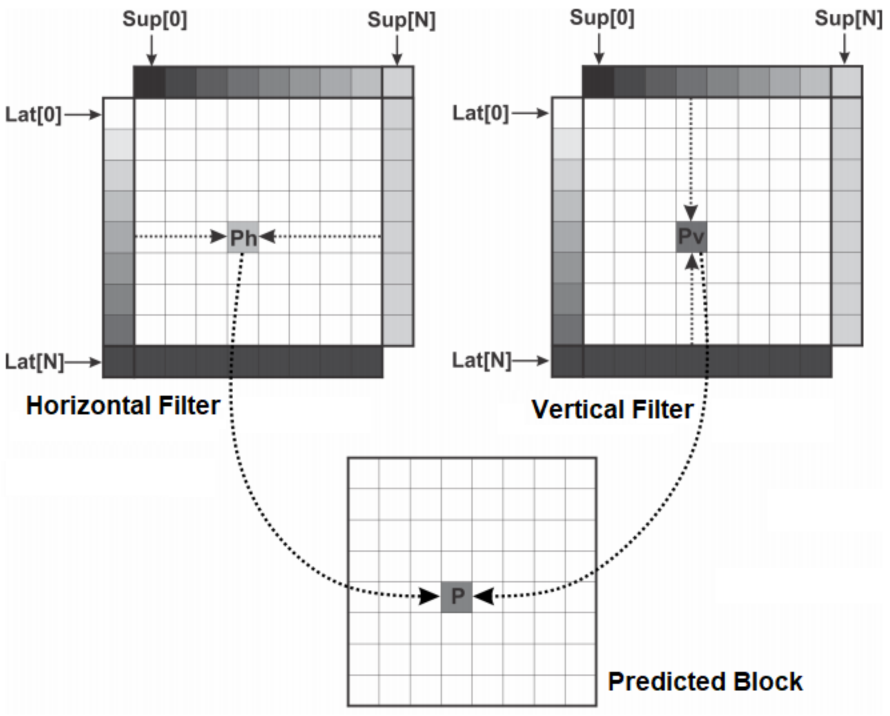
$$\mathrm{Ph}[x][y] = (N-1-x) * \mathrm{Lat}[x] + (x+1) * \mathrm{Sup}[N] \tag{7}$$

$$\mathrm{Pv}[x][y] = (N-1-y) * \mathrm{Sup}[y] + (y+1) * \mathrm{Lat}[N] \tag{8}$$

$$\mathrm{P}[x][y] = (\mathrm{Ph}[x][y] + \mathrm{Pv}[x][y] >> (log2(N)+1) \tag{9}$$

The Planar mode results are obtained by the average of horizontal and vertical filters results, as demonstrated in Equation (9), where P[x][y] is the final result of the Planar mode. Figure 3.6 shows the calculation of Horizontal and Vertical filters in an 8x8 block, as well as the use of the partial results for the Planar calculation.

Figure 3.6 – Example of an 8x8 block calculated using the Planar mode



Source: Author.

3.3.1.2 DC Mode (Mode 1)

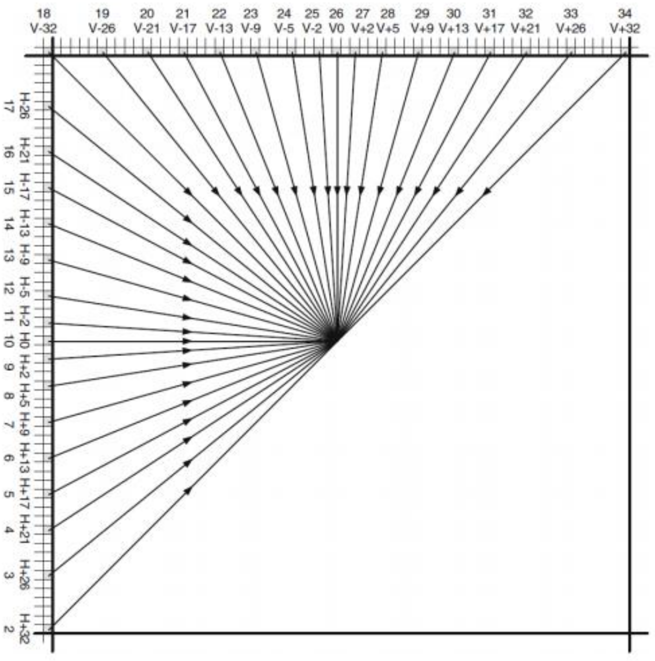
The DC prediction mode is capable of efficiently encoding homogeneous regions (LAINEMA *et al.*, 2014). DC mode is based on the calculation of the value denoted *dcval* and, after that, this value is copied for all samples of the predicted block. If the block does not have any reference sample, the *dcval* value is changed for the middle value considering the bit-depth of the samples. That is, if the samples are eight bit, *dcval* is set to 128.

3.3.1.3 Angular Modes (Modes from 2 to 34)

The Angular modes were defined aiming at efficiency modeling the directional structures presented in video textures (LAINEMA *et al.*, 2014), where well-defined edges are presented (SULLIVAN *et al.*, 2012). For that, angular modes generate the prediction blocks by projecting the neighboring samples in different directions.

The prediction with the Angular modes in HEVC is done similarly to its predecessor standard, the H.264/AVC, but with a larger number of reference samples, angles and block sizes. Each angular mode works with a precision of 1/32 (LAINEMA et al., 2014). The angles used in the HEVC standard were selected after evaluations and, therefore, they are not equidistant. The angles associated with angular modes can be seen in Figure 3.7.

Figure 3.7 – Angular modes definition



Source: (McCANN *et al.*, 2014).

The angular modes can be classified into vertical modes and horizontal modes. The horizontal modes (from 2 to 17) use mainly samples from the left of the block being predicted whereas the vertical modes (from 18 to 34) use mainly samples above the block being predicted to represent the current block. To obtain the value of each sample of the predicted block, a set of equations are used to define which neighboring sample is used to compute the value of each sample from the predicted block, and also the multiplication factor when the weighted average of two samples is used.

Each angular mode has an associated pre-defined value called Angular Parameter (A) that varies from -32 up to 32 representing the direction of the prediction process. The neighboring samples of the current block are used to create a reference vector by simply copying the neighboring samples, based on equations (10) and (11). However, the modes that have negative values for A (from 11 to 25) have also an associated pre-defined parameter B used to project some samples required for the prediction. In other words, the B parameter becomes necessary to select the samples that will be used to complete the reference vector in each mode, as presented in Figure 3.8, where some specific neighboring samples are projected to complete the reference vector. The projection of the required samples to complete the reference vector can be computed by equations (12) and (13), being *x* and *y* the indexes of the reference vector.

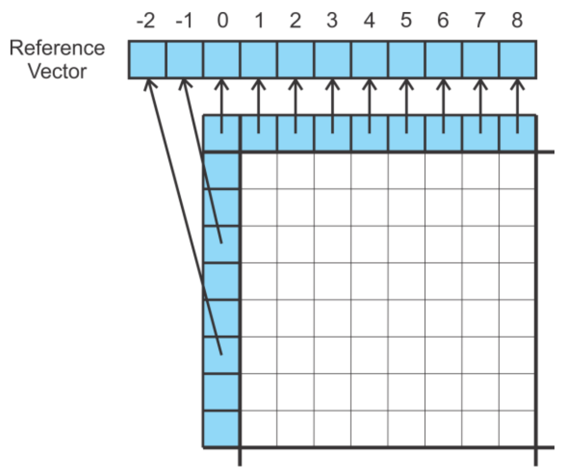
$$ref[x] = side[x], \text{ for horizontal modes} \tag{10}$$

$$\mathrm{ref[y]} = \mathrm{top[y]}, \text{for vertical modes} \tag{11}$$

$$ref[x] = top[((x * B) + 128) \gg 8], \text{for horizontal modes} \tag{12}$$

$$ref[y] = side[((y * B) + 128) \gg 8], \text{for vertical modes} \tag{13}$$

Figure 3.8 – Projection of samples to complete the reference vector



Source: Author.

After generating the reference vector, the block is predicted by projecting the samples of the reference vector in the respective angle. In this projection, the value of the samples from the predicted block is computed by equations (14) and (15), which are a weighted average of the two samples from the reference vector located in the direction of the respective angle.

$$p[x][y] = \big((32 - f) * ref[y + i + 1] + f * ref[y + i + 2] + 16\big) \gg 5, \text{ for horizontal modes} \quad (14)$$

$$p[x][y] = \big((32 - f) * ref[x + i + 1] + f * ref[x + i + 2] + 16\big) \gg 5, \text{ for vertical modes} \quad (15)$$

In equations (14) and (15), i represents a projection integer displacement on row y or column x, which are used to select different samples from the reference vector depending on which sample of the predicted block is being computed. The value of i can be computed by equations (16) and (17).

$$i = \big((x + 1) * A\big) \gg 5, \text{ for horizontal modes} \quad (16)$$

$$i = \big((y + 1) * A\big) \gg 5\ , \text{ for vertical modes} \quad (17)$$

In addition, the equations (14) and (15) select two samples from the reference vector to compute each sample of the predicted block. These two samples are multiplied by an f value, which is the weight of each of the two selected neighboring samples, based on the distance from the predicted sample to the neighboring samples, represented by y or x. The value of f can be obtained by the equations (18) and (19).

$$f = \big((x + 1) * A\big)\ \&\ 31, \text{ for horizontal modes} \quad (18)$$

$$f = \big((y + 1) * A\big)\ \&\ 31, \text{ for vertical modes} \quad (19)$$

Based on equations (10)–(19), the current block can be predicted using the neighboring samples of the current block, based on any of the 33 angular modes used in the HEVC standard.

Despite the HEVC standard defining 33 angular prediction modes, the Mode 10 (Horizontal) and the Mode 26 (Vertical) have a particularity since they only perform a simple copy of the reference samples in the horizontal direction, for the Mode 10, and in the vertical direction, for the Mode 26.

3.3.2 Inter-frames Prediction

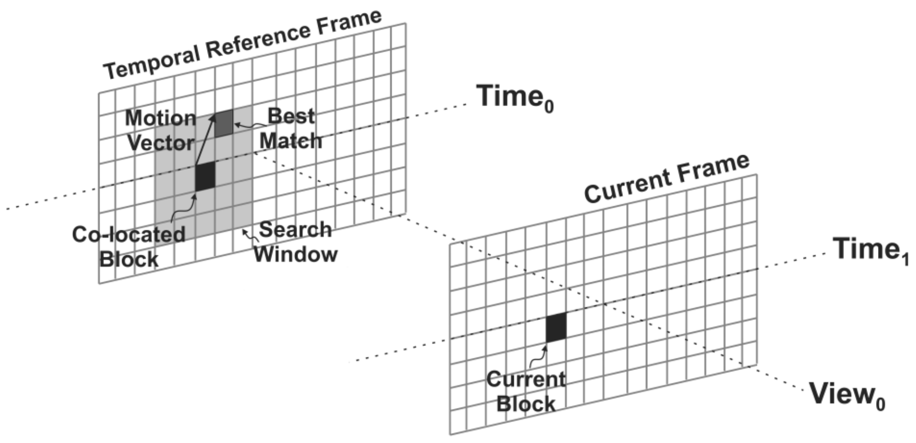
As previously mentioned, HEVC employs a block-based hybrid video coding scheme. Each block is evaluated by several coding tools, and the most efficient one is selected. Among these tools, the Inter-frames prediction, with the ME and Merge modes, take a prominent position in current video encoders due to the large compression gains achieved and the high energy consumption related to their intense processing and memory communication (as discussed in Section 6.1). Both ME and Merge are responsible for exploring temporal redundancies by performing block matching between frames at different time instants, as will be better explained in the sequence.

3.3.2.1 Motion Estimation (ME)

The Motion Estimation (ME) stands as the most computational/energy/memory intensive task within the encoder (ZATT *et al.*, 2011b). Its goal is to exploit temporal redundancy by searching for similar blocks in previously encoded frames.

The ME consists of comparing the current block with reference blocks of previously processed frames in order to find the most similar one under a similarity criterion. The search pattern is determined by a BMA (Block Matching Algorithm) and it is typically constrained within a Search Window (SW). The TZS (Test Zone Search) algorithm (JIA et al., 2013) has become the most widely used BMA in the latest encoder implementations, whereas the SAD (Sum of Absolute Differences) is the most commonly used similarity criterion when real-time systems are targeted. The ME process can be observed in Figure 3.9.

Figure 3.9 – Illustration of the ME process



Source: Author.

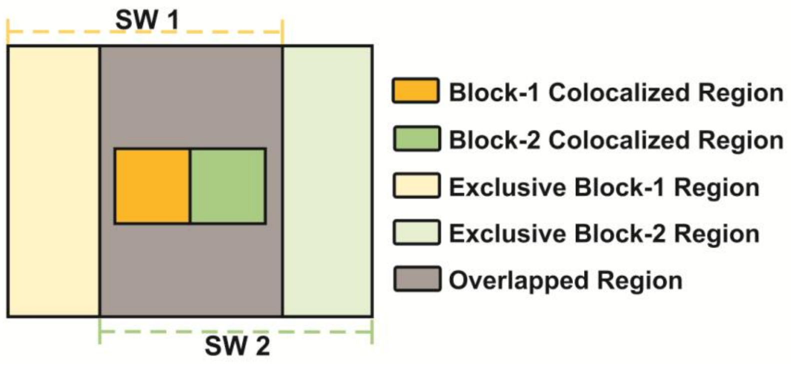
Additionally, to improve coding efficiency, HEVC define the use of different block sizes (known as PUs) in the ME, ranging from 64x64 down to 4x8 pixels. Considering the full Rate-Distortion Optimization (RDO) process (3D-HEVC Reference Software, 2019), the decision on which PU size must be used occurs after the complete evaluation of all possible combinations considering the 24 PU sizes, so that the best tradeoff between compression and distortion is achieved. Note that even when only 32x32 blocks are considered, the BMA should evaluate 90 million candidate blocks per second in order to allow real-time processing of HD (High Definition) 1080p 3D videos with three views (AFONSO *et al.*, 2019). In this scenario, each evaluated candidate requires the fetching of 32x32 8-bit samples from the reference memory leading to memory traffic of 92 GB/s.

Memory access emerges as one of the main bottlenecks regarding energy consumption and processing rate when considering an ME hardware design. On the one hand, a limited memory bandwidth harms the system performance due to the existence of many candidates to be compared by the BMA. On the other hand, huge memory bandwidth results in a significant power dissipation/energy consumption of the system bus (CHEN *et al.*, 2006). Therefore, data reuse must be adopted to reduce memory bandwidth while leaving the system performance unhindered.

The Level-C scheme allows the reuse of overlapping memory regions between neighboring SWs (CHEN *et al.*, 2006). It considers that, during the coding process, neighboring blocks share much of the SW. Thus, this data may be kept in the on-chip memory avoiding external memory retransmission. Figure 3.10 depicts this process. When the encoder requests the Block-2 SW, the Exclusive Block-1 Region is discarded from on-chip memory, the Exclusive Block-2 Region is brought from the external memory, and the Overlapped Region is reused.

Note that although Level-C was proposed a few years ago, it is still widely used in dedicated ME solutions due to its great tradeoff between efficiency and ease of implementation. Furthermore, current related works are still proposing and implementing data-reuse schemes for on-chip memories based on Level-C (JIA *et al.*, 2015) for their ME systems. Other works adopt Level-C as the baseline for comparison (ZATT *et al.*, 2011a) (SAMPAIO *et al.*, 2013). Therefore, this data-reuse scheme was adopted as the baseline implementation in this doctorate, which allows demonstrating the gains of the developed solutions for memory hierarchy and management implemented in this doctorate, and also comparisons with the main related works.

Figure 3.10 – Level-C scheme overview



Source: (AFONSO *et al.*, 2019).

As explained before, ME applies the BMA to select a set of reference blocks to be compared with the current block, and to find the most similar reference block to the current block. The output of the ME step is the information about the motion vector related to the best matching block, i.e., the most similar processed reference block to the current block. This information is used to reduce the data transmission/storage, since only the difference between the current block and the most similar reference block, and the motion vector related to this block, are necessary to represent the block being encoded.

To reduce the complexity of the ME, the BMA is applied only to a pre-defined and reduced search window, which size depends on the CTU size and also on the search range (SR). There are several BMA in the literature with the focus on the ME step (FAN *et al.*, 2018) (PERLEBERG *et al.*, 2018), each one with its particularities. The HEVC Reference Software, the HM, adopts the TZS algorithm to perform the ME step. TZS is classified as a fast algorithm (LI *et al.*, 2014b) because it processes a small set of reference blocks of a given SW, while it maintains the visual distortion practically unchanged when compared to the Full Search (FS) algorithm (LI *et al.*, 2014b), which generates the optimal result for a given SW.

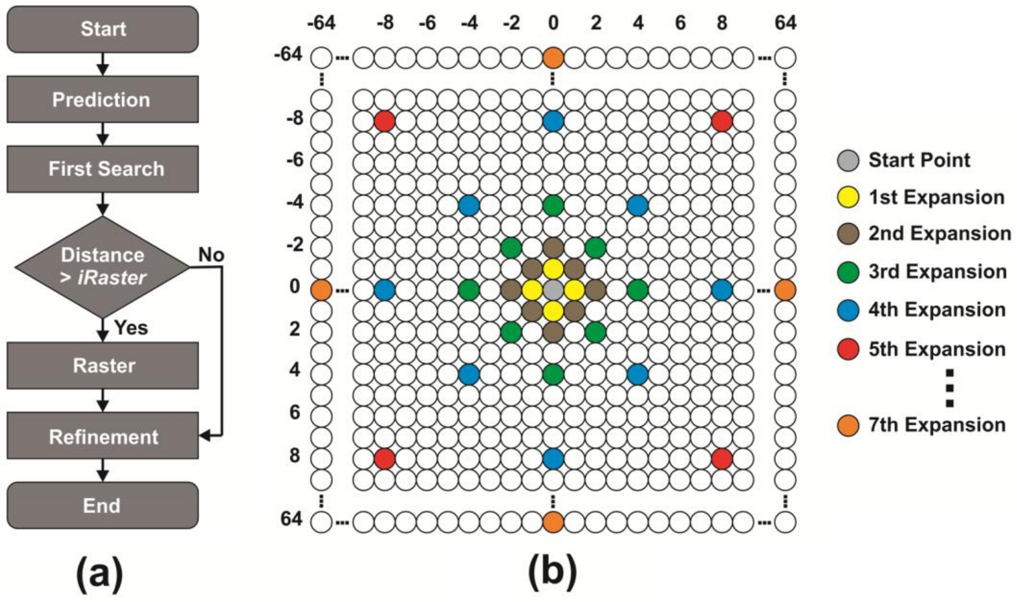
The TZS algorithm is composed of four steps, where each step result defines the start position of the next step. These steps are the Prediction, First Search, Raster and Refinement (LI *et al.*, 2014b), and the flowchart of TZS steps is present in Figure 3.11-a. The Prediction is the first step of the TZS. It is composed of five different predictors that evaluate specific blocks based on their probability of being the best matching block. After this, the First Search is applied.

Both First Search and Refinement steps perform an expansive search which compares several blocks around their start point. The scheme of the expansive search is presented in Figure 3.11-b, where the colored circles represent the first sample of each candidate block. It starts evaluating the four blocks around the position of the start point (highlighted in yellow). Then, the search expands in a diamond scheme (the scheme can vary according to the default configurations of the HM) to evaluate candidates farther from the center until it reaches a stop condition.

The First Search and Refinement step has two stop conditions that occur when the SW limit is reached, or when the algorithm expands three consecutive times without finding a better matching block. The First Search advances to the next step when any of these conditions occur. However, the Refinement only ends the iteration (set of expansions around a given position) when the SW limit is reached, or when the algorithm expands three consecutive times without finding a better matching block. In other words, the Refinement starts a new iteration around the best result found in the previous iteration when those stop conditions occur. This characteristic of the Refinement step leads to unpredictable behavior of the TZS algorithm. The TZS only ends when the Refinement does not find a better result through the current iteration.

The Raster step is a sub-sampled full search that can be applied between the First Search and the Refinement. It is used only when the vector from the resulting block of the First Search is larger than the *iRaster* constant (*iRaster* is defined as 5 by default in the HM).

Figure 3.11 – TZS Algorithm: (a) Flowchart; (b) Expansions



Source: Author.

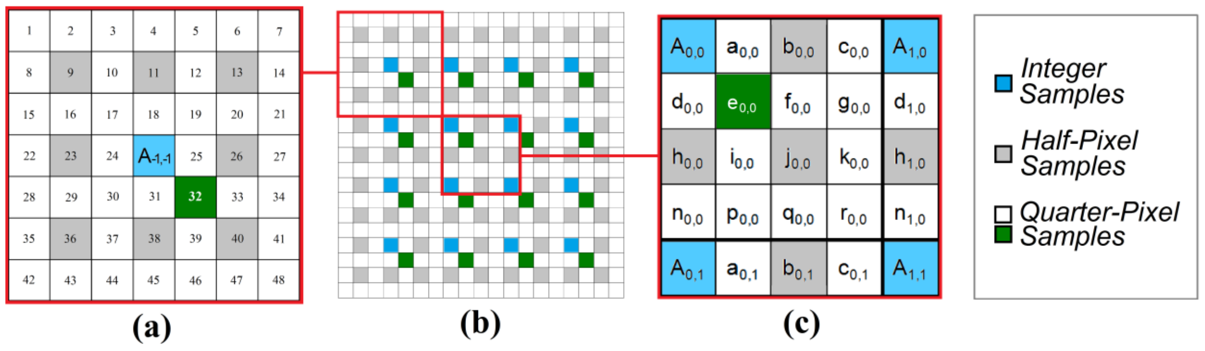
After the BMA finds the most similar block using the ME step, HEVC employs a refinement process over this block. This process is known as the FME (Fractional Motion Estimation) whereas it is common for authors to refer the ME at integer positions as the IME (Integer Motion Estimation). The HEVC standard employs the interpolation of samples in order to generate additional blocks at fractional positions for new comparisons. The HEVC uses FIR (Finite Impulse Response) filters with seven or eight taps to calculate both samples at quarter-pixel or half-pixel positions, which depends on the calculated position.

The filter inputs can receive samples at integer positions or fractional positions previously calculated, depending on the position. Furthermore, the data dependence also varies according to the position to be calculated, i.e., the calculation of a specific sample at a fractional position depends on samples at half-pixel or quarter-pixel positions.

After the interpolation, a search-and-comparison process using samples at half-pixel and quarter-pixel is performed (AFONSO *et al.*, 2016). The search using the fractional samples occurs around the block with the best result considering integer-pixel positions. By default, in the HEVC reference-software implementation, a search with the eight blocks composed of half-pixel positions is performed firstly in the FME, and after that, a search with the eight blocks around the best match of half-pixel blocks is performed using quarter-pixel positions.

Figure 3.12 represents the samples at integer positions (blue squares and uppercase letters), as well as the samples at fractional positions (non-blue squares) for the interpolation process based on the HEVC standard. In Figure 3.12-b, a 4x4 block is represented (due to the space limitation). When fractional samples are generated, 48 new fractional blocks are formed for a new comparison, as presented in Figure 3.12. In Figure 3.12-a, the values from 1 to 48 inside the squares represent the first sample of each new fractional block. The gray squares represent the half-pixel samples, and the white squares represent the quarter-pixel samples. In Figure 3.12-c, the fractional samples are detailed with lowercase letters. As an example, a fractional block with quarter-pixel precision is highlighted in green in Figure 3.12-b. It is important to note that the number of new blocks for comparison (48 fractional blocks) does not depend on the PU size.

Figure 3.12 – Fractional Motion Estimation: (a) First samples of the 48 fractional blocks; (b) A 4x4 block; (c) Samples at integer and fractional positions


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
$A_{-1,-1}$
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
(a)
(b)
$A_{0,0}$ $a_{0,0}$ $b_{0,0}$ $c_{0,0}$ $A_{1,0}$
$d_{0,0}$ $e_{0,0}$ $f_{0,0}$ $g_{0,0}$ $d_{1,0}$
$h_{0,0}$ $i_{0,0}$ $j_{0,0}$ $k_{0,0}$ $h_{1,0}$
$n_{0,0}$ $p_{0,0}$ $q_{0,0}$ $r_{0,0}$ $n_{1,0}$
$A_{0,1}$ $a_{0,1}$ $b_{0,1}$ $c_{0,1}$ $A_{1,1}$
(c)
Integer Samples
Half-Pixel Samples
Quarter-Pixel Samples

Source: (AFONSO *et al.*, 2016).

Fifteen equations are used to calculate the fractional positions (ITU-T, 2015) based on the FIR filters with 7-taps or 8-taps. The fractional positions $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$ and $n_{0,0}$ are calculated from the luminance values at integer positions. The calculation for determining the fractional positions $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$ and $r_{0,0}$ requires values of the positions $a_{0,i}$, $b_{0,i}$ and $c_{0,i}$ previously calculated, where *i* varies from -3 to 4 in the vertical direction (ITU-T, 2015). It is important to notice that, during the interpolation process, some samples around the block are used to calculate the fractional samples. Since the filter inputs require seven or eight samples, a border of samples is needed to calculate the fractional samples located at the borders of the blocks.

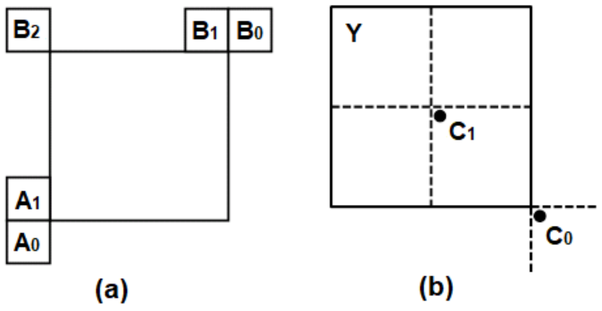
3.3.2.2 Merge and Skip Modes

The HEVC, as well as the H.264/AVC, uses the Skip mode in the case of the PU being encoded is very similar to the collocated block in the reference frame. The main characteristic of the blocks encoded with Skip mode is that no information is transmitted by the encoder for representing the PU. In the HEVC, the concept of Skip mode is incremented through the Merge mode. In the Merge mode, the motion parameters of the PU being encoded are obtained from the neighboring spatial and temporal PUs previously processed, as presented in Figure 3.13.

The main difference of the Merge mode when compared to the Skip mode is that the motion parameters must be transmitted for each PU, including motion vectors and a reference frame index. When a PU is predicted with Merge mode, an index pointing to a Merge candidate list is analyzed from the bitstream and so, used to recover the motion information. The Merge candidate list construction is specified in the HEVC standard document (ITU-T, 2015) and this list has a maximum number of candidates considering the blocks at neighboring positions of the block being encoded. Considering the spatial-neighboring blocks, until four candidates are selected among five possible candidates located in specific positions. In the case of the temporal-neighboring blocks, only one candidate can be selected among the two possible candidates. Since the decoder assumes a constant number of Merge candidates for each PU, additional candidates are generated when the number of candidates does not reach the maximum number signalized in the Slice Header.

The Skip mode can be applied to the 2Nx2N PU partition, i.e., to four block sizes, while the Merge mode can be applied to anyone of the all 24 possible PU sizes. More details about the coding tools employed in the HEVC standard can be obtained in (SULLIVAN *et al.*, 2012), (McCANN *et al.*, 2014), and in HEVC standard document (ITU-T, 2015).

Figure 3.13 – Merge candidates: (a) Spatial-neighboring candidates; (b) Temporal-neighboring candidates



Source: Author.

4 3D-HEVC EXTENSION

This chapter presents in the details the extension 3D-HEVC. Initially, a historical summary of the 3D-HEVC is presented and, in the sequence, the coding structure and the encoding tools are explained.

4.1 Historical Summary of the 3D-HEVC

In March 2011, a call for proposals (ISO/IEC, 2011) was disclosure in parallel to the HEVC standardization aiming the developing of a 3D video-coding standard capable of working with an arbitrary number of views, when maintaining high efficiency in compression and image quality. Among several proposals, one proposal based on the HEVC standard and in the concept of depth maps was chosen to be the initial point to the development of a new 3D video-coding standard.

After the choose of the baseline proposal, the group JCT-3V was created in 2012 from the collaboration between the experts of the VCEG (*Video Coding Experts Group*) of ITU-T (*International Telecommunication Union – Telecommunication Standardization Sector*) and the MPEG (*Moving Picture Experts Group*) of ISO/IEC (*International Organization for Standardization/International Electrotechnical Commission*) having as goal the development of a 3D video-coding extension in the top of the HEVC standard. Before, VCEG and MPEG groups had collaborated in the HEVC standardization through the JCT-VC (*Joint Collaborative Team on Video Coding*), which starts its activities in 2010 and generates the first version of the HEVC standard in April 2013.

The development of this 3D extension for the HEVC standard was needed since the algorithms used in the 2D video-coding tools are inefficient in a 3D video-coding context when an arbitrary number of views is considered. Even the MVC extension of the H.264/AVC, developed focusing on the coding of multi-view videos, presented non satisfactory results for the 3D technology advance, mainly in function of the current demand for high and ultra-high-resolution videos and the rise of multi-view auto-stereoscopic displays (MÜLLER *et al.*, 2013). The first version of the HEVC standard containing the 3D extension was published in February 2015 (ITU-T, 2015).

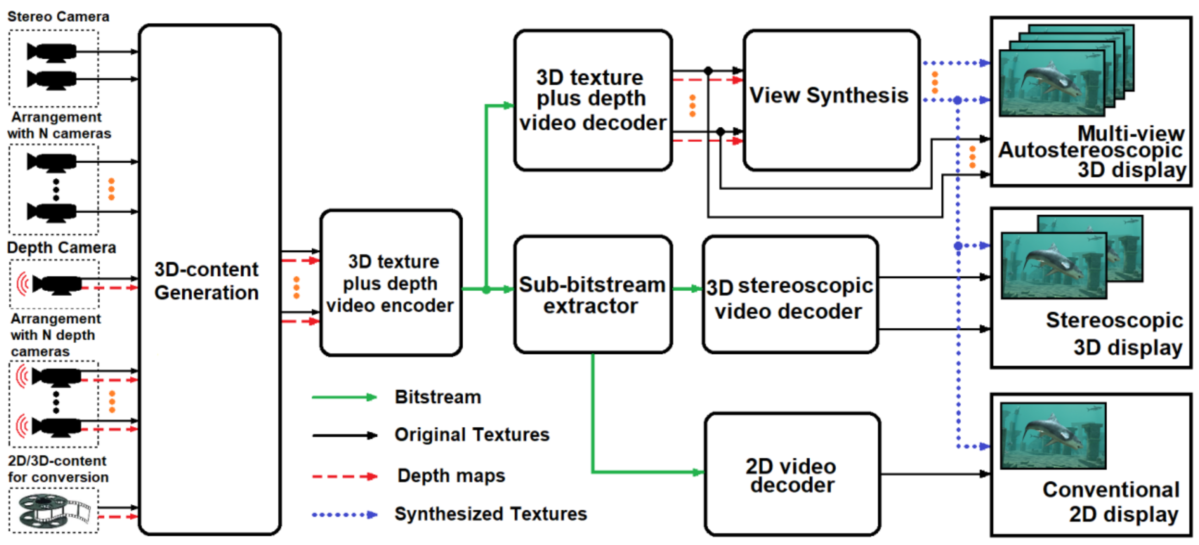
In turn, 3D-HEVC introduces the concept of depth maps for the coding of 3D videos along with new video-coding tools in order to reach high efficiency regarding compression. The use of depth maps allows that, from two or three views (composed of texture pictures plus their respective depth maps), several synthetic views representing intermediate positions can be generated maintaining image quality in this processing (MÜLLER *et al.*, 2013).

The following explanations on the 3D-HEVC extension and its coding tools are in agreement with the documents provided by the JCT-3V: the document “*Test Model 11 of 3D-HEVC and MV-HEVC*” (CHEN *et al.*, 2015), and the 3D-HEVC standard document (TECH *et al.*, 2015).

4.2 3D-HEVC Basic Coding Structure

Figure 4.1 presents an overview of a 3D-HEVC-based system that can be used to encode 3D videos, following an MVD format (Multi-view Video plus Depth). As presented in Figure 4.1, 3D videos can be generated from different camera arrangements and signal sources containing either of the following: (i) a stereoscopic camera; (ii) a depth camera or a time-of-flight camera; (iii) an arrangement with n texture cameras; (iv) an arrangement with n depth cameras; (v) 3D content generated from a 2D-content conversion.

Figure 4.1 – Overview of the basic 3D coding structure in conformance with the 3D-HEVC extension


Stereo Camera
Arrangement with N cameras
Depth Camera
Arrangement with N depth cameras
2D/3D-content for conversion
3D-content Generation
3D texture plus depth video encoder
3D texture plus depth video decoder
View Synthesis
Sub-bitstream extractor
3D stereoscopic video decoder
2D video decoder
Multi-view Autostereoscopic 3D display
Stereoscopic 3D display
Conventional 2D display
Bitstream
Original Textures
Depth maps
Synthesized Textures

Source: Author.

This way, before the pictures are coded with the 3D-video coder, they need to pass through a step of 3D-content generation. If the pictures have depth information (depth maps) (from depth cameras, for example), they can be directly processed by the 3D encoder. The depth maps associated with the texture pictures can be directly generated from cameras such as the time-of-flight (LI, 2014a). But if the depth map is not available, the texture pictures must pass by a depth-estimation step (SMOLIC *et al.*, 2011).

Figure 4.1 also shows that the 3D-HEVC encoder generates a bitstream which can be completely decoded to reconstruct the MVD video for a multi-view autostereoscopic display, but it also can be used to generate stereoscopic videos (only two views) or even 2D videos.

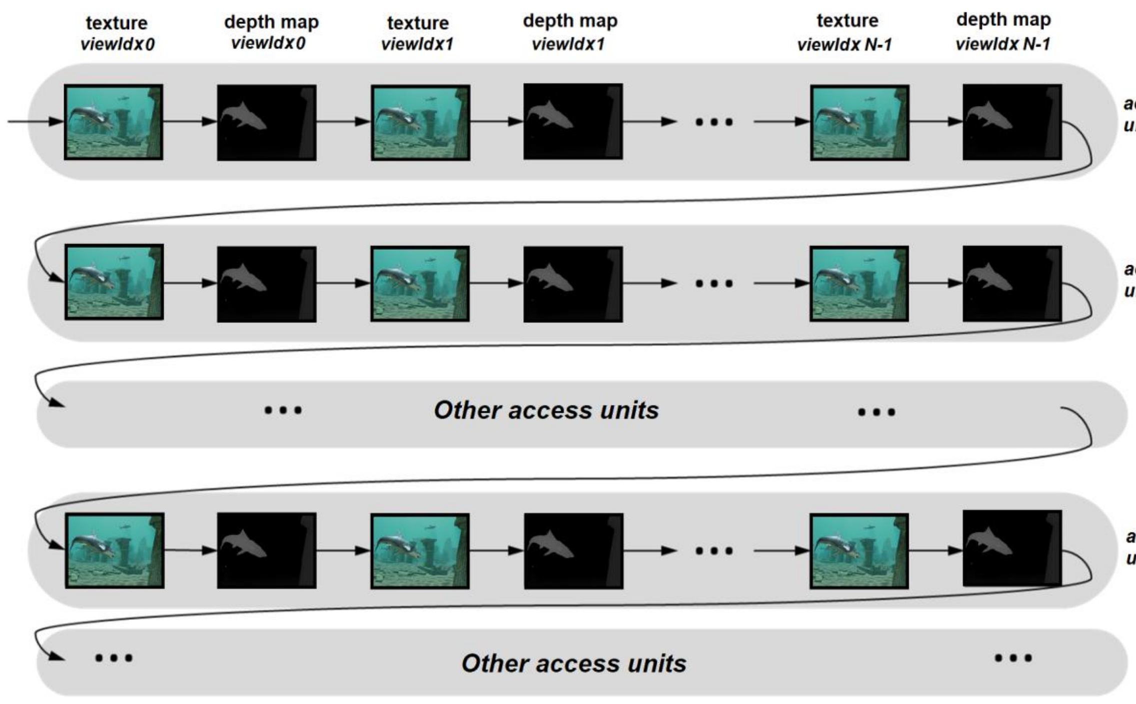
This approach allows the reproduction of this content in different display technologies. After the decoding of the complete bitstream, the 3D decoder delivers a specific number of texture pictures, and their respective depth maps to the View Synthesis step. In turn, a stereo decoder can receive a sub-bitstream and deliver two channels of texture based on the original views to a stereoscopic display. Finally, a 2D decoder can provide 2D-content for a 2D conventional display from a sub-bitstream extracted from the bitstream transmitted by the 3D coder.

The View Synthesis step uses a rendering process called DIBR (Depth-Image-Based Rendering) (MÜLLER *et al.*, 2013) to generate intermediate synthetic texture views between the original texture views. This way, a multi-view autostereoscopic display can be used to reproduce this content.

Figure 4.1 also shows that texture views can be provided for both 3D stereoscopic and 2D displays from the View Synthesis step.

The 3D-HEVC encoder organizes the input texture and depth maps information using the concept of Access Units (AUs), as presented in Figure 4.2. An AU contains all texture pictures and their respective depth maps belonging to the same time instant.

Figure 4.2 – Encoding process based on Access Units



Source: Author.

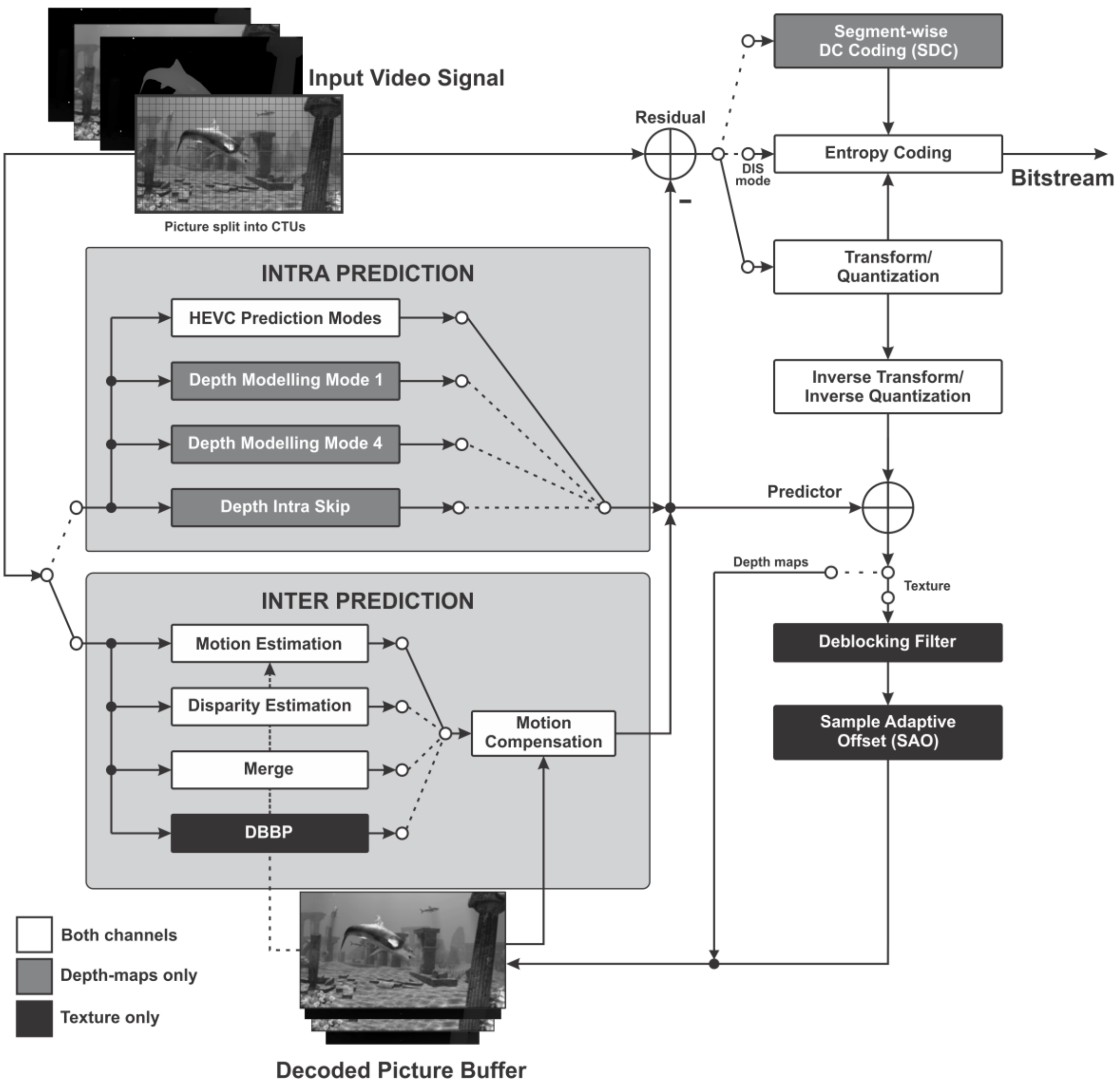
Each AU can have multiple views so that an index (***viewIdx***) is used to identify these views during the coding process. A number is attributed to each view index, i.e., the texture picture and its respective depth-map receive the same number. The index can range from *0* to *n-1*, and the view that receives the index *0* is always the first one to be encoded in each AU. This first view to be encoded is called Base View (BV), or Independent View, and it is encoded without using information from other views. The other views belonging to an AU are called Dependent Views (DVs), and they receive index values between *1* and *n-1* according to the coding order. It is important to notice that in the BV, the texture picture is always encoded before its depth map. By default, texture pictures are also encoded before depth maps in DVs, however, this order can be inverted enabling the Flexible Coding Order (see Section 4.2.2). As presented in Figure 4.2, each AU is also represented by an index number. The first AU receives the index *0*, whereas the following AUs receive indexes between *1* and *n-1*.

The 3D-HEVC inherits all block partition structures and the temporal prediction structure defined in the HEVC. In other words, an encoder based on the 3D-HEVC also divides the pictures (both textures and depth maps) into smaller square-shaped block sizes before applying the encoding tools. The same HEVC concepts are presented in 3D-HEVC, which covers the Coding Tree Units (CTUs), Coding Units (CUs), Prediction Units (PUs), Transform Units (TUs), and Group of Pictures (GoPs), among others.

Figure 4.3 shows a simplified 3D-HEVC Encoding Model in which some novel encoding tools and methods can be observed. Considering the Intra-frame prediction, the 3D-HEVC inherited all the HEVC prediction modes, which are used to encode both texture and depth frames. Besides, the 3D-HEVC Intra-frame prediction includes new tools to deal with depth maps, comprising the depth modeling modes one and four (DMM-1 and DMM-4) and the Depth Intra Skip (DIS). In the Inter-frames and Inter-view predictions, Motion and Disparity Estimations are used to encode both texture pictures and depth maps using the same tools defined for the HEVC Inter-frames prediction. The Depth-Based Block Partitioning (DBBP) mode is applied only to the texture pictures.

The residual encoding supports the HEVC Transform and Quantization steps and also other two alternative methods. The first one is the Segment-wise DC Coding (SDC) which can be applied to all prediction modes used in the encoding of depth maps, except the DIS tool. SDC method consists of a residual encoding method in which the residual data bypass transform and quantization steps. The DC residue of a PU is calculated from the average of the differences between the original sample values and the predicted sample values. SDC can be applied only when a PU uses the 2Nx2N partition. The second one is applied only for blocks predicted with DIS mode, where the residual encoding is bypassed, and the residues are directly processed by the Entropy Coding.

Figure 4.3 – Simplified 3D-HEVC encoding model



Source: Author.

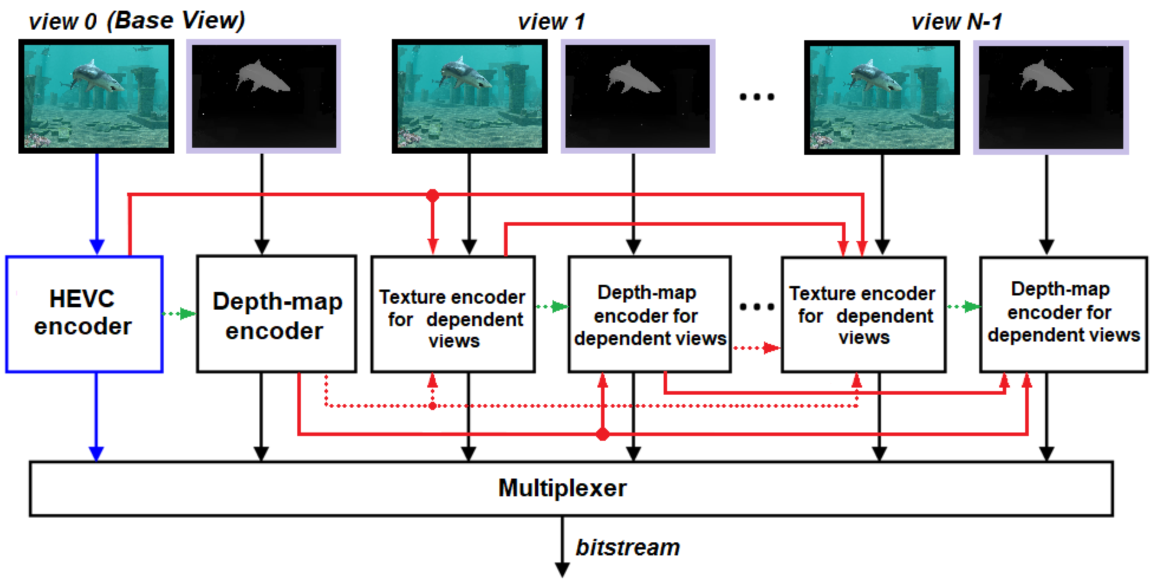
4.2.1 Conventional 3D-HEVC-based Encoding Process

Figure 4.4 shows a basic structure of the 3D-HEVC encoder where one can observe that the texture picture of the BV (***view 0***) is processed with a conventional HEVC encoder. This way, the decoder can provide pictures to a conventional 2D display. The other texture pictures and the depth maps are encoded using HEVC-based encoders, where additional coding tools are inserted to explore the inter views and inter components redundancies and, also the depth maps characteristics, as explained in the next paragraphs. The final bitstream is provided after a multiplexing step where the outputs of each encoder (one for each channel of each view) are switched for the system output.

The Inter-component prediction is the tool responsible for exploiting the redundancies between different channels, i.e., between texture and depth maps channels. Texture information can be used to encode the depth maps in the same view, represented by the dotted green arrows in (used in the DMM-4 mode, for example – see Section 4.4.1.2) and, also, the depth-map information can be used to encode the texture of other views, represented by the dotted red arrows in (used in the DBBP mode, for example – see Section 4.3.3). However, the Inter-component prediction can be disabled and, in this case, the texture pictures are decoded without using the depth maps information. The Inter-view prediction explores the redundancies between different views but considering the same channel and it is represented by the continuous red arrows in Figure 4.4. Basically, the Inter-view prediction is composed of the Disparity Estimation tool (see Section 4.3.1).

Figure 4.4 shows the Inter-component and Inter-view predictions considering the conventional 3D-HEVC encoding process, here called conventional coding order (CCO).

Figure 4.4 – CCO base structure of the encoder with Inter-component and Inter-view predictions


view 0 (Base View)
view 1
view N-1
HEVC encoder
Depth-map encoder
Texture encoder for dependent views
Depth-map encoder for dependent views
Texture encoder for dependent views
Depth-map encoder for dependent views
Multiplexer
bitstream

Source: Author.

As previously mentioned, BVs are encoded through an encoder in conformance with the HEVC. In the case of DVs, the same concepts and encoding tools are used. However, data of views previously encoded are used to enhance the coding efficiency of DVs by adding new coding tools to the HEVC encoder. Therefore, Sections 4.3 and 4.4 presents this novel encoding tools in details.

4.2.2 Encoding Using Flexible Coding Order (FCO)

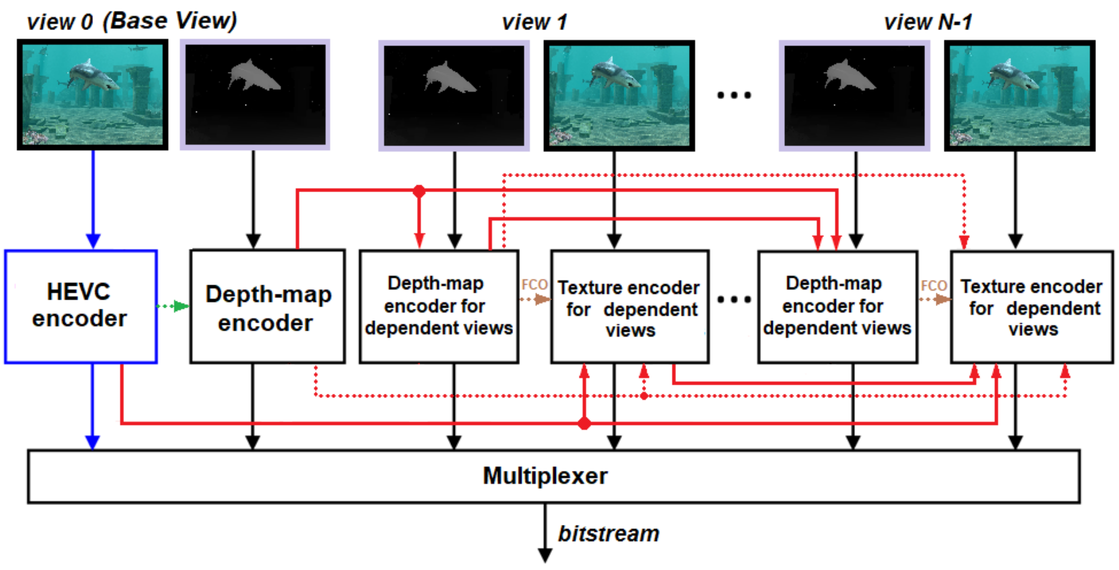
The Flexible Coding Order (FCO) for the 3D-HEVC was proposed in the work (GOPALAKRISHNA *et al.*, 2013). FCO features the encoding of depth maps before their associated texture pictures for the dependent views, i.e., the first view (denoted Base View or

Independent View) is encoded with a conventional way (texture before depth map) and in the remaining views, that used information from other views, the depth maps can be encoded first, as depicted in Figure 4.5.

A conventional 3D-HEVC coding order (CCO) improves the compression efficiency of dependent views by using disparity information derived from the motion information of the neighboring blocks (additional candidates tested in the Merge mode). The idea brought by the FCO configuration is that disparity information can be derived directly from the depth data increasing the accuracy of the disparity vectors. This method with a different encoding order is depicted by the dotted brown arrows in Figure 4.5, where disparity information is used to aid the prediction of the current texture block. The work (GOPALAKRISHNA *et al*., 2013) measured the impact of the FCO method regarding compression and it decreases by 0.6% the BD-Rate of video total, on average. In other words, this approach improves the compression performance of 3D-HEVC. Also, the proposed configuration reduced the complexity of decoding by over 8%.

In this thesis, the FCO configuration was exploited in the hardware design presented in Section 8.1 with the goal of obtaining energy efficiency for the developed ME/DE system. This feature enables the depth information to manage the memory and processing units of the developed system according to the channel that is encoded first.

Figure 4.5 – FCO base structure of the encoder with Inter-component and Inter-view predictions


view 0 (Base View)
view 1
view N-1
HEVC encoder
Depth-map encoder
Depth-map encoder for dependent views
FCO
Texture encoder for dependent views
Depth-map encoder for dependent views
FCO
Texture encoder for dependent views
Multiplexer
bitstream

Source: Author.

4.3 3D-HEVC Dependent-view Encoding

This section presents the novel encoding tools introduced by the 3D-HEVC encoder to better deal with the dependent-view encoding.

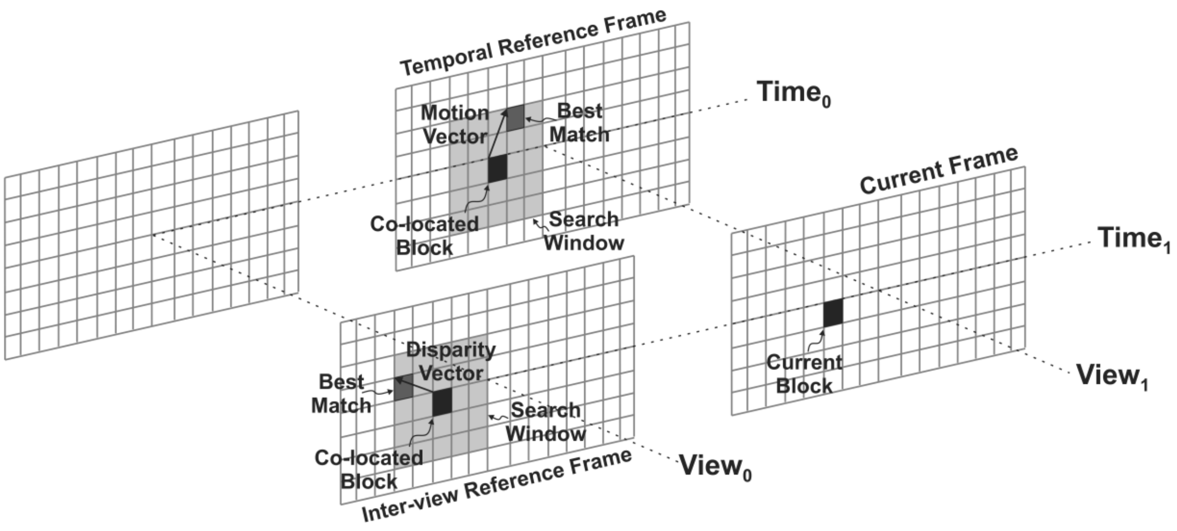
4.3.1 Disparity-Compensated Prediction (DCP)

The Disparity Estimation (DE) technique is used in the 3D-HEVC Inter-view prediction to exploit inter-view redundancies. This technique was introduced by the MVC, and it uses a concept quite similar to the ME, widely used in the current video coding standards, such as the H.264/AVC and HEVC. The difference is that the DE uses previously processed neighboring views as reference frames and the position is called disparity vector in this case.

Although DE and ME have many similarities, these techniques also present some important differences. Whereas ME explores the temporal redundancy of the same view, i.e., similarities between the temporally neighboring frames, DE explores the redundancy between different views, both in the texture and depth maps, considering the same time instant (the same Access Unit).

Figure 4.6 presents examples of the predictions based on DE and ME techniques. In the 3D-HEVC documents, these predictions are called DCP (Disparity-compensated Prediction) and MCP (Motion-compensated Prediction), respectively.

Figure 4.6 – Examples of MCP and DCP predictions



Source: Author.

Other important difference between ME and DE is in the process used to generate the vectors. In general, ME generates vectors with reduced values for the block matching, which allows smaller search areas in the reference frames and, consequently, lower complexity in the process. Furthermore, in the ME, the search performed around the position of the collocated block in the reference frame tends to present a good result. On the other hand, DE tends to generate vectors with higher values than the ME for the block matching. This way, searches around the position of the collocated block can result in an increased search window to reach a satisfactory coding efficiency. The bigger is the search window, the higher is the computational effort needed for the encoding process. The 3D-HEVC coding tools were designed to be more efficient when the views are aligned in 1D linear and coplanar arrangements (TECH *et al.*, 2016) and the camera arrangement used to record the videos follows a horizontal displacement (MÜLLER *et al.*, 2014) (TECH *et al.*, 2016). However, by default, the 3D-HEVC Reference Software (3D-HTM – 3D-HEVC Test Model) defines a vertical search range equal to 56 and the TZS algorithm as the BMA for the DE, i.e., it performs a 2D search in a scenario where only 1D displacement is expected, demanding unnecessary computation by using an inappropriate algorithm for DE prediction. This thesis proposes some heuristics to exploit this issue in Chapter 8.

Given the similarities between DE and ME previously mentioned, these steps share the Motion Compensation (MC) step. It is important to notice that the DE is applied only in the dependent views.

In 3D-HEVC, as two different Inter predictions can be performed, the Inter-frames and the Inter-view predictions, two types of reference frames are included in two different reference lists (called L0 and L1). For the MCP, Inter-temporal reference frames are included and, for the DCP, Inter-view reference frames are included (previously processed frames that bellows to the same Access Unit). More details about the steps used to construct the reference lists can be found in (ITU-T, 2015) and (CHEN *et al.*, 2015).

Despite the DE increasing the coding efficiency, DE competes with the ME in the encoding process. Since the ME presents better prediction than the DE in most cases, the DE is used in smaller regions of the frame. This way, the DE is mainly used to predict occluded regions due to the temporal motion (MÜLLER *et al.*, 2013).

Also, several techniques applied in the 3D-HEVC can use the obtained disparity vector to improve coding efficiency. More details about these techniques used in the 3D-HEVC can be found in Chen *et al.* (2015).

4.3.2 Modification in the construction of the Merge Candidate List

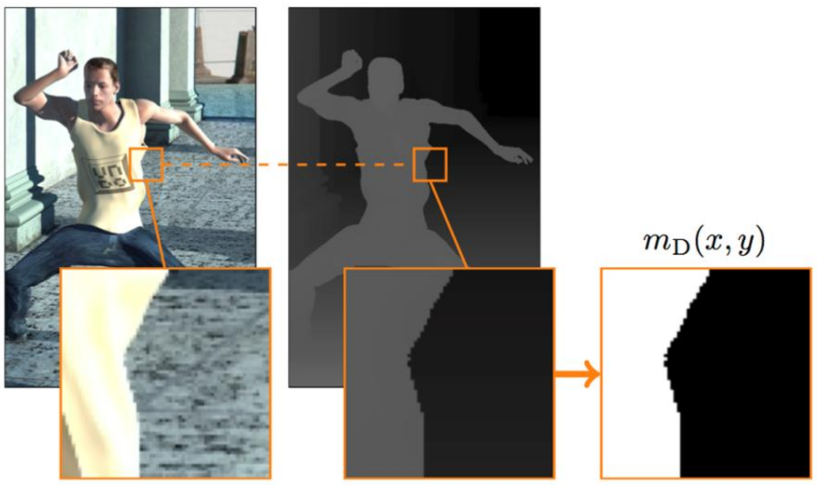
The process used to construct the Merge Candidate List (MCL) is modified in the 3D-HEVC when compared to the HEVC. This modification aiming at obtaining improved efficiency in the prediction with a motion vector in the dependent views since the correlation of the information between the views can also be exploited. The maximum number of candidates in the MCL is increased to six in 3D-HEVC. Additional candidates are included in the MCL list considering the high probability of two associated blocks in the BV and the dependent views have the same motion information since these two views represent the same scene captured with synchronized cameras, only under different perspectives. These candidates consider the motion information of previously encoded blocks in the reference view and the disparity information that points to the Inter-view block in the reference view. The motion information of one of the dependent views can be predicted through the BV previously encoded since a disparity vector linking these two associated blocks is known. The steps to construct the MCL list for the dependent views of texture in the 3D-HEVC is described with details in Chen *et al.* (2015).

4.3.3 Depth-Based Block Partitioning (DBBP)

The Depth-Based Block Partitioning (DBBP) consists of a mode that derives an arbitrary partition for a given collocated texture block based on a Binary Segmentation Mask (BSM) calculated from the correspondent depth map. The two partitions obtained from this process are similar to the foreground and the background images, in which a merge step based on the BSM is applied. The process of generation of the masks starts getting the disparity vector in order to identify the correspondent depth-map block in the reference view with the same size of the current texture block. After the depth-map block is found, a threshold value is calculated based on the average value of the samples inside the correspondent depth-map block. This threshold value is used together the depth values to generate the BSM m_D (x,y). This process is relatively simple. If the depth value located in the (x,y) coordinates is higher than the threshold value, the m_D (x,y) is equal to 1 and, otherwise, m_D (x,y) is equal to 0. Figure 4.7 depicts the process used for the generation of the masks.

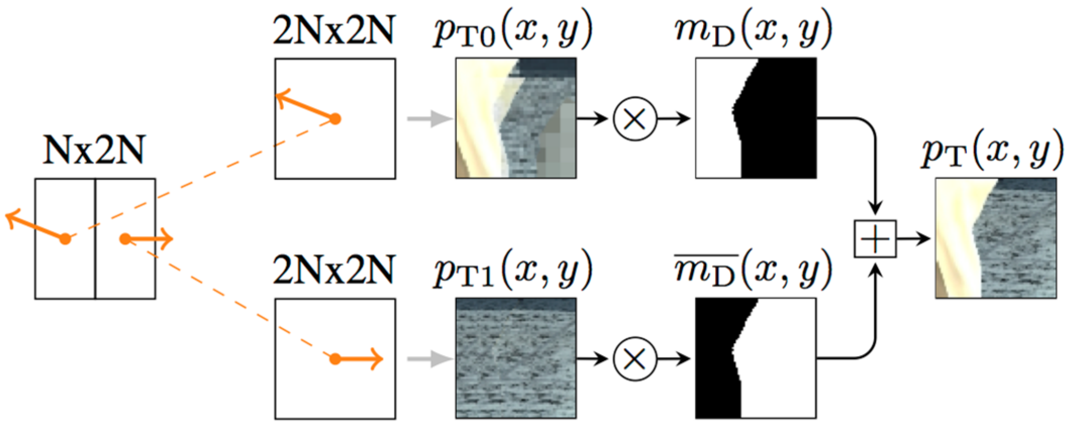
In the case of BSM is taken into account, two motion parameters are estimated in the encoder using a conventional ME and SAD as the similarity criterion. As one can observe in Figure 4.8, a 2Nx2N block predicted using MCP is performed for each one of the two decoded motion parameters. In the following, the resultant prediction signals p_{T0} (x,y) e p_{T1} (x,y) are combined using the mask m_D (x,y). The two sets of motion parameters are predicted considering that the current block is encoded with two PUs of 2NxN or Nx2N sizes. Finally, the output p_T (x,y) is equal to p_{T0} (x,y), if m_D (x,y) is equal to 1. Otherwise, the output p_T (x,y) is equal to p_{T1} (x,y). It is important to notice that the DBBP mode is only applied to the 2NxN and Nx2N PU partitions, and to the 64x64, 32x32, 16x16 CU-depth levels.

Figure 4.7 – Detailing of the mask generation using DBBP and the sequence *Undo_Dancer* (MÜLLER *et al.*, 2014)



Source: Chen *et al.* (2015, p. 29).

Figure 4.8 –Merging process used in the DBBP



Source: Chen *et al.* (2015, p. 30).

4.4 3D-HEVC Depth-Map Encoding

The depth-map encoding uses the same base concepts used in the coding of texture videos, such as the Intra-frame prediction, MCP prediction, DCP prediction, and Transform-based coding. However, some tools are modified, other tools are disabled, and novel encoding tools are introduced.

Due to the fact that one of the main aspects related to the depth maps is the presence of sharp edges, the interpolation filters of 7 and 8 taps used in the HEVC MC (samples with quarter-pixel precision) can produce artifacts when applied to the sharp edges. These artifacts can generate significant errors in the synthesized views. In order to avoid these errors in the synthesis process and reduce the encoder/decoder complexities, the MCP and DCP predictions are modified in the Inter-frames and Inter-view predictions, where the fractional prediction is not allowed.

Also, the in-loop filters used in HEVC are developed targeting the encoding of texture videos, which makes these filters inefficient in the depth-map encoding. This way, all in-looping filters (Deblocking Filter, and Sample-adaptive Loop Filter) are disabled in the depth-map encoding. Hence, the complexities of the encoder and the decoder can also be reduced.

The size of the Merge Candidate List for depth maps is also extended in one position, as in the texture encoding.

Since the HEVC Intra-frame prediction encoding tools are not so efficient to deal with the sharp edges, neither with very homogeneous regions, 3D-HEVC introduces novel encoding tools along with the ones inherited of HEVC to enhance the efficiency of depth maps in the Intra-frame prediction. These novel encoding tools are explained in details in the following sub-sections.

4.4.1 Depth Modeling Modes (DMM)

As previously mentioned, depth maps have some specific characteristics, such as the presence of sharp edges (represent the edges of the objects) and large areas with practically constant values or values with smooth variation (represent the inner parts of the objects).

The Intra-frame prediction and the transform-based encoding of the HEVC are efficient to encode image regions with smooth variation. Therefore, the 35 Intra modes used in HEVC (33 directional modes, DC mode, and Planar mode) were maintained in 3D-HEVC to efficiently encode those regions. However, the use of those modes can generate artifacts by encoding sharp edges. Such artifacts result in significant errors in the process used for generating the synthetic intermediate views. To deal with this problem, new Intra modes called Depth Modeling Modes (DMM) were proposed for coding of depth maps in 3D-HEVC.

Two DMMs are integrated with the conventional Intra-frame prediction used in HEVC. The DMMs generate a residue that represents the difference between the approximation of the depth map used in the encoding process and the original depth map. This residue can be transmitted through a transform-based coding, as in the Intra modes inherited from the HEVC. In these two new DMMs, each block of a given depth map is approximated using a model that partitions the internal area of the block into two regions represented by two constant values. Therefore, two information are needed: (i) the information that allows the identification of what partition the samples belong (called partition information); (ii) the constant value of the samples that belong to each region (called region value information). The information that corresponds to the value applied to each region is indicated by a CPV (Constant Partition Value). For a given partition, the better approximation is obtained by using the average value of the original depth values for the correspondent region as the CPV.

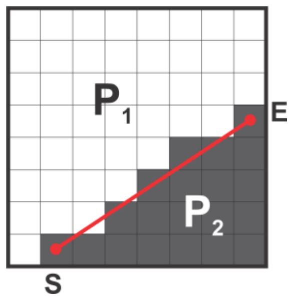
The DMMs differ in the segmentation of the block into two regions. These segmentations can be done using *Wedgelets* and *Contours*.

While the 3D-HEVC development, four Depth Modeling Modes were proposed for the Intra-frame prediction, but only two of those modes remains in the current version of the 3D-HTM. These two modes are called: (i) DMM-1 or EWS (Explicit Wedgelet Signaling); and (ii) DMM-4 or ICPCP (Inter-component-predicted Contour Partitioning). Both DMM-1 and DMM-4 modes support 4x4, 8x8, 16x16 and 32x32 block sizes. Additional information about both DMM-1 and DMM-4 is given in the following.

4.4.1.1 DMM-1: EWS (Explicit Wedgelet Signaling)

When the DMM-1 is applied, the block is divided into two regions using a straight line, as presented in Figure 4.9, which considers the partition of an 8x8 block as an example. The two regions are called P_1 and P_2. The start point of the straight line is represented by *S*, whereas the end point of this line is represented by *E*. These two points are always located on two different external borders of the block.

Figure 4.9 –Wedgelet partition of an 8x8 depth-map block



Source: Author.

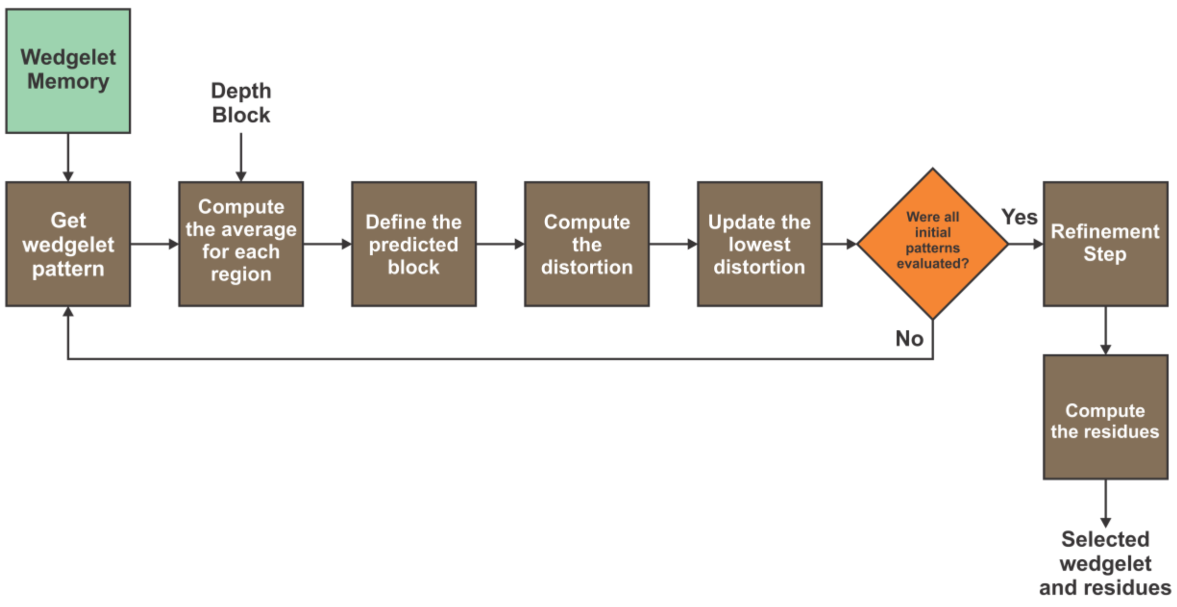
The DMM-1 mode finds the best match for each evaluated block using wedgelet partitions. The encoder performs a search over a set of wedgelet partitions using the original depth values of the current block as the reference. The wedgelet partition that presents a minimal distortion with the original depth signal is selected, and the correspondent partition information is transmitted in the bitstream. Each wedgelet partition is decided using the start and the end positions of the segmentation line along the borders of the current block.

The 3D-HTM uses different lists to store the wedgelets evaluated by the DMM-1 mode according to the block size. These sets of wedgelets are generated before the coding process, and all the wedgelets inside a specific list are evaluated during the prediction of its respective block size. The generation of the wedgelets occurs into two steps, one initial step followed by a refinement step. In the first step, all wedgelets between two adjacent and between two opposite borders are created and stored for each list according to the block size. This process starts with two adjacent borders and, as soon as these wedgelets are obtained, this set of wedgelets is rotated in order to generate all wedgelets considering adjacent borders. The generation of wedgelets for opposite borders is similar (ZHANG *et al.*, 2014) (IKAI *et al.*, 2015). It is important to notice that the generation of wedgelets for different block sizes applies different sampling to determine the start and end points. The 16x16 and 32x32 block sizes consider start and end points every two samples along the borders, whereas the 4x4 and 8x8 block sizes consider start and end points every sample along the borders, i.e., 4x4 and 8x8 block sizes consider all possible wedgelets. However, after the generation of the wedgelets among adjacent and opposite borders, the 3D-HTM implementation removes the wedgelets with the same start and end points or the ones with higher similarity.

Figure 4.10 presents the algorithm applied to the DMM-1 mode according to the 3D-HTM implementation. For each DMM-1 partition to be tested, the associated wedgelet pattern is searched in the Wedgelet Memory. Basically, each storage wedgelet consists of a binary mask and, an average value to be used in each region (CPV value) is computed based on this binary mask from the original depth block. After the calculation of the average values, the predicted block is derived where each region assumes the respective CPV value previously calculated. The following step consists of the distortion computation in which the lowest distortion is updated until all initial patterns defined in 3D-HTM are evaluated. When all initial patterns are evaluated, the wedgelet that presents the lowest distortion in previous steps is used for a Refinement Step, where additional patterns are tested. The refinement step consists of generating eight new wedgelets patterns to be evaluated around the previously selected wedgelet. These new wedgelets are generated apart one sample of the previously selected wedgelet covering the eight possible cases. After all evaluations, the selected wedgelet and the residues are delivered by the Residue Computation.

The optimal results for the DMM-1 are reached testing all possible wedgelets combining start and end points from the different block borders, which requires a high computational effort. As previously mentioned, the 3D-HTM implementation applies some restrictions to the start and end positions of the wedgelet partitions in order to reduce the number of wedgelet partitions to be evaluated, the storing of wedgelets in the Wedgelet Memory, and consequently, the complexity associated with the DMM-1 mode. The reduced number of wedgelet partitions used for comparisons, according to the block size, is presented in Table 3.1. Still, DMM-1 is a bottleneck in both processing and memory since a large number of wedgelet patterns are processed and 183,264 bits are required to store all wedgelets patterns (ZHANG *et al.*, 2014) (IKAI *et al.*, 2015) of all block sizes supported by the encoding tool, according to the 3D-HTM implementation.

Figure 4.10 – DMM-1 encoding algorithm according to the 3D-HTM implementation


Wedgelet Memory
Depth Block
Get wedgelet pattern
Compute the average for each region
Define the predicted block
Compute the distortion
Update the lowest distortion
Were all initial patterns evaluated?
Yes
No
Refinement Step
Compute the residues
Selected wedgelet and residues

Source: Author.

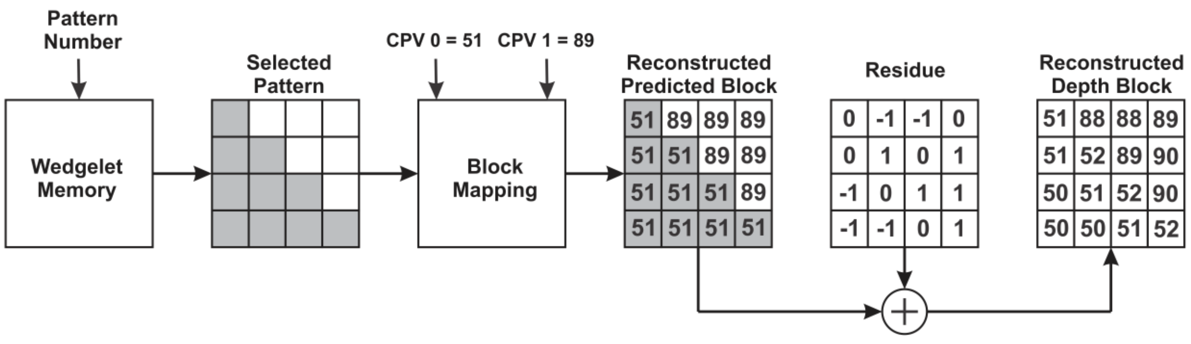
Table 3.1 – Number of DMM-1 wedgelets according to the 3D-HTM implementation

Block size	Total of possible wedgelets
4x4	86
8x8	802
16x16	510
32x32	510

Source: (IKAI *et al.*, 2015).

In the decoding side, the predicted block is reconstructed using the partition information transmitted in the bitstream. In DMM-1, the partition information is not predicted, therefore this mode is called Explicit Wedgelet Signaling. Figure 4.11 shows DMM-1 decoding process for a 4x4 depth-map block, as an example. For a given depth map block, the decoding algorithm requires the selected number of the wedgelet pattern, the CPV for each encoded region, and the residual values. While the DMM-1 encoding process evaluates an expressive amount of wedgelet patterns, which results in many memory accesses, the decoding process only requires the access of the wedgelet indexed by the pattern number transmitted to the decoder. Then, the wedgelet pattern that was selected in the encoding process is retrieved. After that, the CPV values are mapped into this DMM-1 pattern to generate the reconstructed predicted block. Finally, the residues values are added to the reconstructed predicted block to generate the reconstructed depth map block.

Figure 4.11 – Example of the DMM-1 decoding process for a 4x4 depth-map block

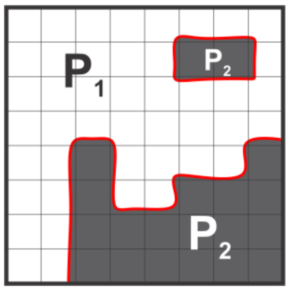


Source: Author.

4.4.1.2 DMM-4: ICPCP (Inter-component-predicted Contour Partitioning)

In the case of the Contour partition, the segmentation of the two regions cannot be easily described by a geometric function, since the regions P_1 and P_2 can have arbitrary forms and several parts. An example of Contour partition can be observed in Figure 4.12. The partition scheme considering the contour partition is individually obtained for each block from the reference block, i.e., no list can be used.

Figure 4.12 – Contour partition of an 8x8 depth-map block



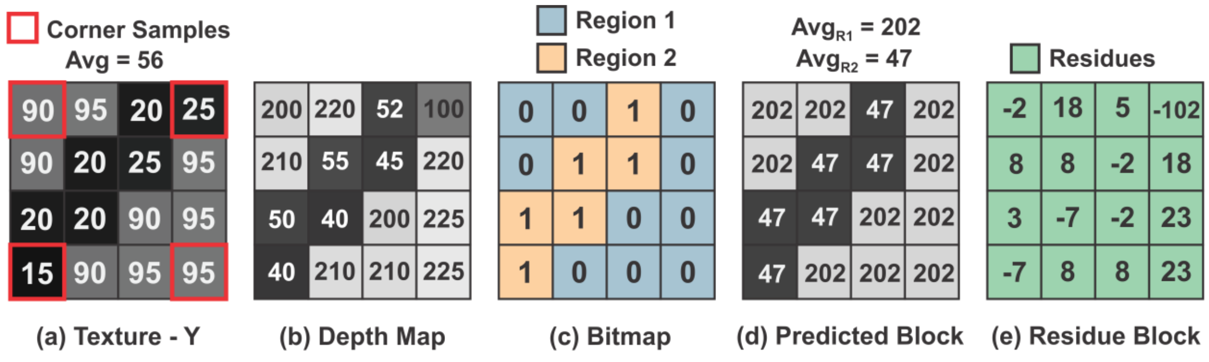
Source: Author.

The DMM-4 mode consists of the prediction of the Contour partition (see Figure 4.12) from a texture reference block using Inter-component prediction. The reconstructed luminance signal of the collocated block in the associated texture frame is used as the reference. The DMM-4 is based on the Rate-Distortion (RD) cost computation, i.e., according to the RD-cost results, the selection among DMM-4 and the other intra prediction modes is made based on the lowest RD cost.

As previously explained, by using Contour partition along the DMM-4, the depth block to be predicted is divided into two regions (or partitions) in which discontinuous regions may belong to each partition, which is not possible with the DMM-1. This division is done using a threshold value from the texture samples of the collocated block inside the texture picture associated with the depth map. From the threshold value, a contour of the objects is constructed as presented in Figure 4.13. This figure shows an example of prediction with the DMM-4 mode considering a 4x4 block size. As one can observe in Figure 4.13-a, the threshold is obtained from the average using the four corner samples presented in the collocated texture reference block (considering the luminance channel – Y), resulting in an average value equal to 56 for this example. The threshold value is then used to divide the depth block represented in Figure 4.13-b into two regions from a bitmap. Those samples that present values higher than the threshold value are identified as the samples that belong to a region 1, whereas the samples with values smaller than the threshold are associated to the region 2 in the Contour partition.

Figure 4.13-c shows the bitmap obtained from the depth block given as an example in Figure 4.13-b. Note that the depth-block values are compared with the threshold value obtained from the texture in order to define if the bitmap receives *0* or *1*. Once this is done, the DMM-4 mode calculates the arithmetic average of all depth block samples of each region from the bitmap to determine the value that will entirely represent each region. This way, the Bitmap can be replaced with the average of the partitions to generate the predicted block, as presented in the example shown in Figure 4.13-d, where the two regions result in average values equal to 202 and 47. Finally, the DMM-4 calculates the difference between the original and predicted samples and delivers the residues, as shown in Figure 4.13-e.

Figure 4.13 – Example of the DMM-4 coding process for a 4x4 depth-map block



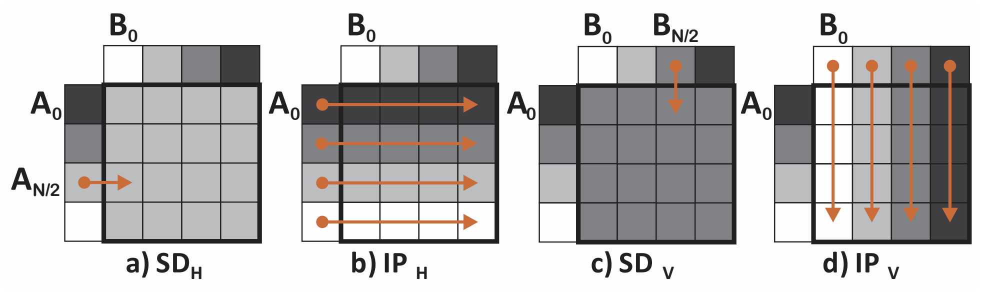
Source: Author.

4.4.2 DIS (Depth Intra Skip) Mode

As previously mentioned, one of the main aspects in the depth maps is the presence of many regions with similar values and, frequently, with the same value. In most cases, small variations of the sample values of homogeneous regions insignificantly affect the quality of the synthesized views. With this in mind, the DIS mode was introduced in the 3D-HEVC. This coding tool avoids the residual coding of homogeneous areas in depth maps, drastically reducing the amount of data required to represent those regions.

DIS presents four prediction sub-modes as presented in Figure 4.14: (i) Horizontal Single Depth (SD_H); (ii) Horizontal Intra Prediction (IP_H); (iii) Vertical Single Depth (SD_V); and (iv) Vertical Intra Prediction (IP_V). SD modes consider only one depth sample of previously processed neighboring blocks to perform the prediction of the current block, i.e., the whole block is predicted with the same value. While the SD_H considers the sample from the left neighboring column located in the position $A_{N/2}$, SD_V uses the sample from the above neighboring row located in the position $B_{N/2}$, where N represents the block size. Note that Figure 4.14 uses an example with 4x4 blocks only due the space limitation. When a spatial neighboring sample is not available for the SD modes, the middle value of the depth range is used for the whole block (e.g., 128 for 8 bits). The IP_H and IP_V prediction modes apply the horizontal and vertical modes commonly used in the HEVC. It consists of copying the reference samples in horizontal and vertical directions, respectively.

Figure 4.14 –Depth Intra Skip sub-modes: (a) SD_H; (b) IP_H; (c) SD_V; (d) IP_V



Source: (AFONSO *et al.*, 2017).

By default, DIS uses SVDC (Synthesized View Distortion Change) as the distortion metric to decide the best mode to encode a given block based on RDO (3D-HEVC Reference Software, 2019). SVDC uses rendering functionalities, as it will be explained in Section 4.5.1.1.

DIS is applied over blocks in the level of CUs, and it can encode any of the possible CU sizes (8x8, 16x16, 32x32, and 64x64 blocks). An index is used to identify the prediction mode of the candidate list of the CU encoded using DIS.

4.5 Control of the 3D-HEVC Encoder

The Rate Distortion Optimization (RDO) technique is used in the decision mode to measure the cost of the different encoding modes and parameters applied to the candidate blocks in the encoding process. The mode or parameter that returns the smallest cost according to the equation (20) is select.

$$\mathrm{J} = \mathrm{D} + \lambda . \mathrm{R} \tag{20}$$

In equation (20), *J* represents the cost, *D* represents the distortion obtained by encoding a given block with a specific mode, *R* represents the number of bits needed to represent the block in this specific mode, and λ is the Lagrangian multiplier obtained based on the QP. For obtaining the distortion, methods such as SSD (Sum of Squared Differences), SAD, or SATD are applied between the original and the reconstructed sample values.

For the depth-map encoding, the same decision process is applied. However, the distortion is measured using a modified process, and it also considers the distortion of the synthesized views, which increases the coding efficiency. This modification is necessary since the geometric information of the depth maps is indirectly explored during the rendering process and the decoded depth map is not visible during this process.

4.5.1 View Synthesis Optimization (VSO)

Occlusions and disocclusions avoid a bijective mapping of depth-map areas with distortion regions in the synthesized view. This occurs when the block (or part of this block) is occluded in the synthesized view. Hence, an exact mapping between the depth-block distortion and the associated distortion in the synthesized view is not possible by using only the information provided by the current block. To measure the distortion of the synthesized views, two different metrics can be applied in RDO.

4.5.1.1 Synthesized View Distortion Change (SVDC)

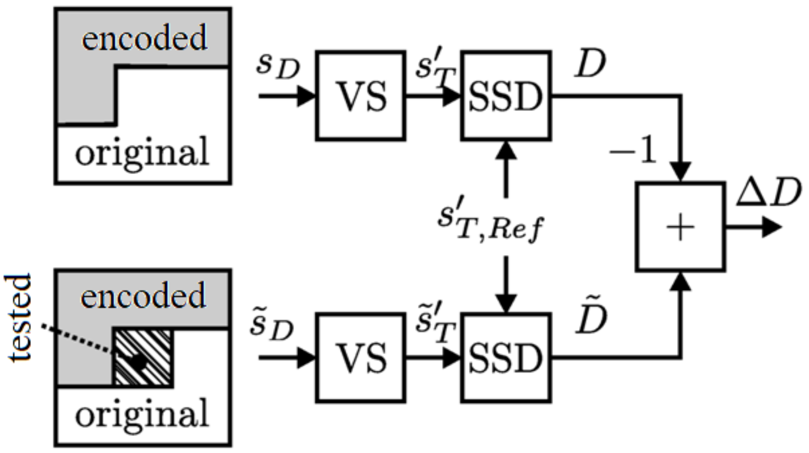
The Synthesized View Distortion Change (SVDC) (CHEN *et al.*, 2015) allows an exact measuring of the synthesized-view distortion that considers the occlusions and disocclusions. As presented in Figure 4.15, the distortion of the synthesized view as a function of the changes inside the depth block B is calculated considering the depth information outside the block B. For that, SVDC is defined as the difference in distortion $\Delta\mathbf{D}$ of two texture synthesized views $\boldsymbol{s}'_{\boldsymbol{T}}$ and $\tilde{\mathbf{s}}'_{\boldsymbol{T}}$ and it is mathematically represented by (21). $\boldsymbol{s}'_{\boldsymbol{T}}$ represents a texture rendered from the depth map $\boldsymbol{s}_{\boldsymbol{D}}$ which consists of depth information reconstructed by blocks previously encoded and original depth information for the remaining blocks. $\tilde{\mathbf{s}}'_{\boldsymbol{T}}$ represents a texture rendered from the depth map $\tilde{\mathbf{s}}_{\boldsymbol{D}}$. In this case, the reconstructed depth values from the current encoding mode are used considering the current block B. The distortions $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ are calculated using the SSD metric. For calculating the

SSD, $\boldsymbol{s'_{T,Ref}}$ represents a reference texture in the initialization rendered using the original video and the depth information. Finally, the difference between $\mathbf{D}$ and $\widetilde{\mathbf{D}}$ is represented by the $\Delta\mathbf{D}$, and it measures the depth distortion. The VS blocks in Figure 4.15 represents the Views Synthesis steps. I represents the set of all samples in the synthesized view.

$$SVDC = \Delta D = \widetilde{D} - D = \sum_{(x,y)\in I}\left[\tilde{s}_T(x,y) - s'_{T,Ref}(x,y)\right]^2 - \sum_{(x,y)\in I}\left[s'_T(x,y) - s'_{T,Ref}(x,y)\right]^2 \tag{21}$$

The SVDC calculation requires a rendering process during the encoding. Given that the required computational effort is a critical factor in the encoding, an alternative method is used. This approach uses fast rendering in order to avoid re-rendering of synthesized views for each distortion calculation. That method supports the base functionalities used in the processing of view synthesis algorithms of the most rendering approaches such as the Sub-Sample Accurate Warping, Hole Filling, and View Blending. More details about the SVDC metric can be obtained in Chen et al. (2015).

Figure 4.15 – SVDC definition related to the distorted depth information of the block B (represented by the hatched area)



Source: Adapted from Chen *et al.* (2015, p. 44).

4.5.1.2 View Synthesis Distortion (VSD)

The View Synthesis Distortion (VSD) is a model based on the distortion estimation without rendering. The distortion estimation of the synthesized views is obtained from the pondering of the depth distortion using a factor obtained from the texture view in a horizontal direction. The depth-map distortion does not affect linearly the synthetic distortion since the impact in the depth-map distortion varies according to the correspondent texture information. Thus, the VSD performs the absolute sum of horizontal texture gradients in order to ponder the depth distortion and estimate the synthesized view distortion with higher efficiency than using SSD.

Equation (22) demonstrates the VSD calculation where $s_D(x,y)$ and $\tilde{s}_D(x,y)$ indicate the original and reconstructed depth map, respectively, and (x,y) means the sample position inside the block. Also, $\tilde{s}_T$ indicates the reconstructed texture, and α is proportional coefficient that depends on the focal length, the baseline between the current and the rendered view, and the values of the nearest and farthest depth of the scene, respectively. More details about the VSD metric can be obtained in Chen *et al.* (2015).

$$VSD = \sum_{(x,y)\in B} \left(\frac{1}{2} \cdot \alpha \cdot |s_D(x,y) - \tilde{s}_D(x,y)| \cdot [|\tilde{s}_T(x,y) - \tilde{s}_T(x-1,y)| + |\tilde{s}_T(x,y) - \tilde{s}_T(x+1,y)|]^2\right) \tag{22}$$

5 3D-HEVC RELATED WORKS

Since the 3D-HEVC was recently released, there are a few published papers targeting efficient hardware designs for the new proposed encoding tools or specifically focusing on the 3D-HEVC challenges. However, since the HEVC tools are also required in the 3D-HEVC implementations, previously published works targeting the HEVC can also be used in 3D-HEVC implementations with some adaptations, as it will be discussed in the following.

5.1 HEVC Related Works

All prediction modes available at the HEVC Intra-frame prediction are also used in the 3D-HEVC. Then, the hardware designed targeting the HEVC Intra-frame prediction can be used in 3D-HEVC implementations. Some works like (MIN *et al.*, 2017) (CORREA *et al.*, 2017) (PALOMINO *et al.*, 2012) (PASTUSZAK *et al.*, 2016a) should be considered when designing dedicated hardware for a complete 3D-HEVC codec. These architectures can be adapted to include the new prediction modes defined by the 3D-HEVC.

The hardware designed targeting the HEVC Inter-frames prediction (motion estimation and motion compensation) can be used in the 3D-HEVC Inter-frames prediction. Since the Inter-frames and Inter-view predictions have very similar behavior, solutions designed targeting the HEVC Inter-frames prediction can be replicated to work in the 3D-HEVC Inter-view prediction with adaptations. Thus, works (PERLEBERG *et al.*, 2018) (SANCHEZ *et al.*, 2015b) (KALALI *et al.*, 2018) (FAN *et al.*, 2018) can be used to design a 3D-HEVC codec.

The hardware designed targeting other HEVC encoder modules: forward and inverse transforms, forward and inverse quantization, entropy coding, and in-loop filters, also can be reused in the 3D-HEVC context with minor adaptations, including the works (KALALI *et al.*, 2014) (LEE *et al.*, 2016) (BRAATZ *et al.*, 2018) (GOEBEL *et al.*, 2016). The HEVC entropy coding, due to its natural control flow behavior, is harder to be parallelized, then designs of high throughput hardware are not easy to be found in the literature, but a few works (RAMOS *et al.*, 2016) (KIM *et al.*, 2015b) (ZHANG *et al.*, 2018) can be used in this scenario. Finally, the published hardware designs targeting the HEVC in-looping filter also can be reused in 3D-HEVC implementations, including works (SHEN *et al.*, 2016) (CHO *et al.*, 2015) (ZHOU *et al.*, 2016). However, none of these solutions is capable of handling depth maps coding and most of them are not able to reach the required throughput to process 3D videos in real time.

In turn, (LEE *et al.*, 2017) implements depth maps coding using the baseline HEVC standard. It proposes an Adaptive Search Window (ASW) that explores the temporal correlation between depth map and texture. The correlation is used to predict the probable range of movements of an object throughout consecutive frames, which results in an encoding time reduction of up to 53% in comparison to TZS.

Moreover, such solutions targeting the HEVC do not feature optimizations considering the runtime 3D-HEVC behavior and processing/memory requirements. Multiple related works proposing memory and energy-aware hardware designs are available in the literature, but focusing on previous 3D video-coding standards, as it will be explained in the next section.

5.2 H.264/AVC and MVC Related Works

Several related works proposing memory and energy-aware hardware designs for ME and DE have been published lately. In the following, some of the most prominent works targeting Inter-frames and Inter-view predictions for previous video coding standards are discussed.

Aiming at reducing the number of block matching operations, many solutions propose Adaptive Search Window (ASW) algorithms. The work (JIA *et al.*, 2013) proposes applying the exhaustive full search (FS) algorithm within a diamond-shaped ASW that is dynamically resized at the frame level. When implemented on an H.264/AVC encoder, the solution reached an encoding time reduction of 80%. The authors in (KIM *et al.*, 2014) propose an ASW algorithm and use a fast ME algorithm to accelerate the HEVC ME based on Graphics Processing Units (GPU). A scalable fast search algorithm suitable for a massively parallel GPU architecture focusing on MVC ME/DE is proposed in (JIANG *et al.*, 2016). A hardware-oriented fast Integer ME (IME) algorithm is proposed in (DOAN *et al.*, 2017). It targets a future parallel ME design capable of processing all HEVC block sizes while reducing the computational complexity by 54.54%. None of the available proposals is able to exploit the tools provided by 3D-HEVC. Moreover, the solutions are not focused on real-time processing and neither consider memory-related issues.

A handful of solutions to reduce and control video on-chip memories are already available. The works (CHEN *et al.*, 2006) and (TSUNG *et al.*, 2016) are among the most widely used on-chip memory data reuse strategies, in which the SW is fully stored on-chip to reduce the accesses to the external memory. However, several samples within the SW are not used due to the adaptive characteristic of the current ME/DE BMAs. The work (ZATT *et al.*, 2011a) proposes on-chip video memory architecture for ME/DE in MVC, in which an application-aware power management scheme based on a multiple-sleep state model is employed. In (SAMPAIO *et al.*, 2013), an energy-efficient memory hierarchy for ME/DE on

MVC was presented. The authors employed a *Reference Frames-Centered Data Reuse* scheme to avoid multiple search window retransmissions leading to a reduced number of external memory accesses and, consequently, memory energy reduction. The on-chip video memory energy was reduced by employing a statistical power gating scheme and candidate blocks reordering. Therefore, up to 71% for external memory energy, 88% on-chip memory static energy, and 65% on-chip memory dynamic energy were saved when compared to the Level-C approach. Unfortunately, none of the described solutions considers depth maps and are unable to exploit motion and disparity characteristics to further reduce energy consumption.

A few complete systems for MVC have also been proposed. In (DING *et al.*, 2010) the authors present an MVC encoder with low energy consumption able to real-time encode four high definition views. In (ZATT *et al.*, 2011b), a run-time adaptive energy-aware ME/DE architecture for the MVC standard is presented, which uses memory access and data prefetching techniques for jointly reducing the on-/off-chip memory energy consumption. The work (ZATT *et al.*, 2011b) also proposes a run-time dynamically expanding SW to reduce the off-chip memory accesses and a power-gating scheme to reduce on-chip memory power dissipation. As a result, (ZATT *et al.*, 2011b) provides a dynamic energy reduction of 82-96% for the off-chip memory and a leakage energy reduction of 57-75% for the on-chip memory in comparison to Level-C and Level-C+. The authors in (CHOI *et al.*, 2013) propose two MVC frame scheduling schemes considering MVC with depth information. These scheduling schemes avoid that the different encoding times of each channel reduce the coding efficiency.

5.3 MV-HEVC and 3D-HEVC Related Works

Although the 3D extension of the HEVC standard was published in 2015, there exist some prominent works in the literature proposing dedicated architectures and algorithmic solutions for the novel Intra-frame encoding tools of the 3D-HEVC.

To efficiently encode depth maps, the 3D-HEVC adopts new encoding tools, as previously discussed. In Intra-frame prediction, 3D-HEVC introduces: (i) Depth Intra Skip (DIS); (ii) Depth Modeling Mode-1 (DMM-1); and (iii) Depth Modeling Mode-4 (DMM-4). These tools increase the coding efficiency but also increase the required computational effort.

The works (KIM *et al.*, 2015a) and (CONCEICÃO *et al.*, 2016) present algorithm solutions to the 3D-HEVC DIS. The work (KIM *et al.*, 2015a) proposes a fast mode decision for the Single Depth (SD) modes based on a decision criterion to detect smooth regions in depth maps. It early decides the use of SD modes by using statistics of smooth signals for depth intra modes and analysis of the distortion metrics. As a result, this work presents a 25.6% encoding-time saving at the cost of a 0.18% increase in the BD-Rate. The work (CONCEICÃO *et al.*, 2016) proposes an Early Skip/DIS mode decision for 3D-HEVC to reduce the complexity of the depth-map coding process. This work is based on an adaptive threshold model that considers the Skip/DIS occurrences as a function of its generated rate-distortion cost. This solution reduces in 33.7% the depth complexity on average with a mean BD-rate increase of up to 0.409%. There is no work in the literature proposing a dedicated hardware designs for the 3D-HEVC DIS coding tool that besides the ones developed in this doctorate. However, the work (AHMAD *et al.*, 2015) presented a complexity, and a hardware architecture analyses for the implementation of Synthesized View Distortion Estimation that is the distortion metric adopted in RDO to evaluate the encoding of blocks using the DIS tool.

The main works published in the literature that propose hardware designs for the 3D-HEVC depth modeling modes, DMM-1 and/or the DMM-4, are (AMISH *et al.*, 2017) (SANCHEZ *et al.*, 2014b) (SANCHEZ *et al.*, 2016) (SANCHEZ *et al.*, 2017b) (SANCHEZ *et al.*, 2018a). It is important to notice that there are no published works relating hardware designs supporting all Intra-frame prediction modes, i.e., the Intra modes inherited from the HEVC and the ones developed to deal with depth maps. The work presented in (SANCHEZ *et al.*, 2014b) implements the DMM-4 prediction mode for 8x8, 16x16, and 32x32 block sizes. The architecture was synthesized for an Altera Stratix V FPGA, and it is capable of processing HD 1080p videos (1920x1080 pixels) with five views at 31.39 frames per second (fps). The work presented in (SANCHEZ *et al.*, 2016) implements the DMM-1 and DMM-4 prediction modes, which can be scaled for all block sizes. DMM-1 memory issues are not covered by this work. The architecture was synthesized for ASIC, and it is capable of processing HD 1080p videos at 30 fps considering one view. The work presented in (AMISH *et al.*, 2017) implements the three modes introduced in the 3D-HEVC extension: DIS, DMM-1, and DMM-4 considering all possible block sizes. A strategy to reduce the DMM-1 complexity was implemented, but memory issues were not covered. The developed architecture was synthesized for a Virtex 6 FPGA, and it can process HD 1080p videos at 30 fps, considering six views. The works (AMISH *et al.*, 2017), (SANCHEZ *et al.*, 2014b), and (SANCHEZ *et al.*, 2016) are not fully compliant with the 3D-HEVC standard. The works (SANCHEZ *et al.*, 2014b), and (SANCHEZ *et al.*, 2016) present differences in the method used to calculate the texture block average in DMM-4 besides the work (AMISH *et al.*, 2017) uses an alternative method to calculate the DIS. A DMM-1 decoder implemented in hardware and supporting the processing of high-resolution videos in real-time is presented in

(SANCHEZ *et al.*, 2018a). The work (SANCHEZ *et al.*, 2017b) presents architecture for the Simplified Edge Detector (SED) algorithm employed in 3D-HEVC depth-maps Intra-frame prediction.

Only one work was found in the literature focused on hardware design targeting the MV-HEVC. The work (LIU *et al.*, 2017) presents a hardware design for the MV-HEVC focusing on the real-time decoding of multi-view videos in mobile devices.

5.4 Related Works Summary

Table 5.1 summarizes some important information related to the most prominent related works found in the literature. Even though there are related works (TSUNG et al., 2016) (ZATT *et al.*, 2011a) (ZATT *et al.*, 2011b) (SAMPAIO *et al.*, 2013) (DING *et al.*, 2010) that propose memory and energy-aware hardware systems for ME/DE of previous video coding standards (MVC), none of them focuses on 3D-HEVC. This means that many of them do not support depth maps coding (or apply texture-only video codecs to encode depth maps) and, as a result, these works do not exploit the specific depth-map characteristics. Additionally, no solutions that fully exploit the MVD correlation space (spatial, temporal, disparity and inter-channel) to jointly reduce computation and memory-related energy consumption were found in the literature.

Considering the hardware designs (AMISH *et al.*, 2017) (SANCHEZ *et al.*, 2014b) (SANCHEZ *et al.*, 2016) (SANCHEZ *et al.*, 2018a) which are capable of dealing with depth-map processing using the 3D-HEVC (MVD approach), all works implemented Intra-frame solutions for specific encoding tools without consider general aspects related to the Intra-frame prediction nor some important issues related to the memory usage and energy consumption required by the encoding tools.

Table 5.1 – Related Works Summary

Related Work	**Video-coding Standard**	**Encoding Tools**	**Depth-map Processing**	**Memory Implementation**
(MIN *et al.*, 2017)	HEVC	Intra-frame Prediction	No	No
(CORREA *et al.*, 2017)	HEVC	Intra-frame Prediction	No	No
(PALOMINO *et al.*, 2012)	HEVC	Intra-frame Prediction	No	No
(PASTUSZAK *et al.*, 2016a)	HEVC	Intra-frame Prediction	No	No
(PERLEBERG *et al.*, 2018)	HEVC	Inter-frames Prediction	No	No
(SANCHEZ *et al.*, 2015b)	HEVC	Inter-frames Prediction	No	No
(KALALI *et al.*, 2018)	HEVC	Inter-frames Prediction	No	No
(FAN *et al.*, 2018)	HEVC	Inter-frames Prediction	No	No
(KIM *et al.*, 2014)	HEVC	Inter-frames Prediction	No	No
(DOAN *et al.*, 2017)	HEVC	Inter-frames Prediction	No	No
(LEE *et al.*, 2017)	HEVC	Inter-frames Prediction	Yes	No
(JIA *et al.*, 2013)	H.264/AVC	Inter-frames Prediction	No	No
(CHEN *et al.*, 2006)	H.264/AVC	Inter-frames Prediction	No	No
(TSUNG *et al.*, 2016)	MVC	Inter-frames and Inter-view Predictions	No	No
(ZATT *et al.*, 2011a)	MVC	Inter-frames and Inter-view Predictions	No	Yes
(ZATT *et al.*, 2011b)	MVC	Inter-frames and Inter-view Predictions	No	Yes
(SAMPAIO *et al.*, 2013)	MVC	Inter-frames and Inter-view Predictions	No	Yes
(DING *et al.*, 2010)	MVC	Encoder	No	Yes
(KIM *et al.*, 2015a)	3D-HEVC	*Intra-frame DIS tool	Yes	No
(CONCEIÇÃO *et al.*, 2016)	3D-HEVC	*Intra-frame DIS tool	Yes	No
(AMISH *et al.*, 2017)	3D-HEVC	** Intra-frame DIS, DMM-1, and DMM-4 tools	Yes	No
(SANCHEZ *et al.*, 2014b)	3D-HEVC	***Intra-frame DMM-4 tool	Yes	No
(SANCHEZ *et al.*, 2016)	3D-HEVC	***Intra-frame DMM-1, and DMM-4 tools	Yes	No
(SANCHEZ *et al.*, 2018a)	3D-HEVC	Intra-frame DMM-1 tool (decoder)	Yes	No
(LIU *et al.*, 2017)	MV-HEVC	Decoder	No	No

*Algorithmic solution

** Not fully compliant with the 3D-HEVC standard (DIS encoding tool)

*** Not fully compliant with the 3D-HEVC standard (DMM-4 encoding tool)

Source: Author.

6 3D-HEVC REFERENCE SOFTWARE EVALUATIONS

Given that the 3D-HEVC extension can be considered state of the art in terms of 3D-video coding, and it brought innovations along with its depth-map processing and novel encoding tools, an investigation of the 3D-HEVC encoding tools and the development of algorithms and hardware architectures capable of processing multi-view 3D-videos in real time are motivating and challenging activities.

This chapter presents the evaluations performed through the 3D-HEVC Reference Software, the 3D-HTM, during this doctorate, and it is divided into three main sections. The first one presents 3D-HTM Time Profiling and Memory Analyses. The second section presents a statistical analysis based on the encoding tools considering both the CCO (Conventional Coding Order) and the FCO (Flexible Coding Order) configurations. These statistical analyses were performed with the goal of identifying the importance of each encoding tool and, this way, to propitiate the proposal of efficient complexity-reduction strategies for the 3D-HEVC encoder. The third one discusses the obtained results. It is important to notice that the major part of the experiments performed during this work followed the test recommendations provided by the JCT-3V as well as used version 16.0 of the 3D-HEVC reference software. This experimental setup is shown in details in Appendix A. There are a few exceptions from evaluations where the experimental setup or the 3D-HEVC reference software version is different from those above mentioned. In these specific cases, the differences are highlighted throughout the text and discussed.

6.1 3D-HTM Time Profiling and Memory Analyses

The new coding tools introduced by 3D-HEVC to efficiently encode MVD content result in a higher complexity when compared to the HEVC. It is important to mention that the HEVC already has a high complexity when compared with previous standards. Therefore, the development of real-time 3D systems, which is the focus of this thesis, is a very challenging task, demanding dedicated energy-aware hardware solutions to enable real-time encoding, especially when portable devices are considered. The relevance of this research topic is placed in evidence due to the high 3D-HEVC complexity and its features.

The memory and computational effort required for both the novel depth 3D-HEVC intra prediction tools, such as the DMM-1 mode, and the 3D-HEVC ME/DE steps evidences the need for the development of complexity-reduction strategies and VLSI designs focusing on both 3D-HEVC Intra-frame prediction, and Inter-frames and Inter-view predictions.

The main challenges regarding memory, processing, and complexity for the 3D-HEVC predictions (Intra-frame, Inter-frames, and Inter-view) are discussed in the next two subsections.

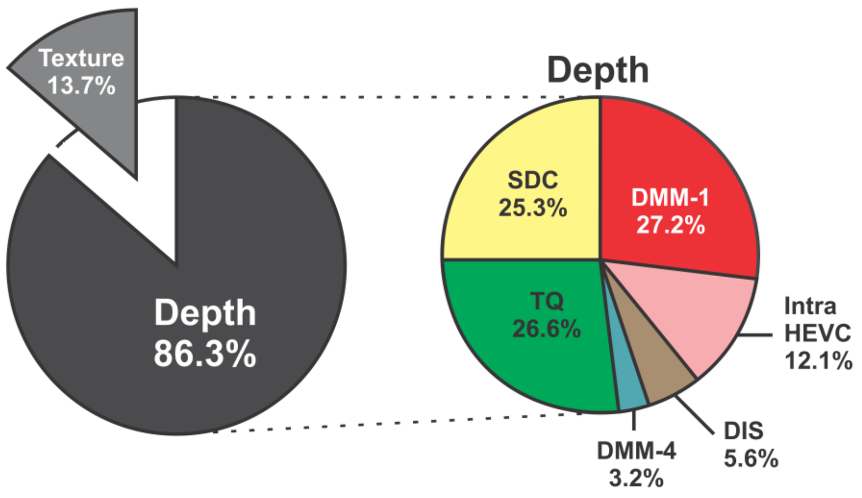
6.1.1 3D-HEVC Intra-frame Prediction

Figure 6.1 presents an analysis of 3D-HEVC Intra-frame prediction regarding the time profiling of 3D-HEVC intra coding. These results were obtained through evaluations using the 3D-HTM (3D-HEVC Test Model) reference software version 16.0 (3D-HEVC Reference Software, 2019) under AI (All-Intra) encoder configuration (see Section 3.2), i.e., only Intra-frame prediction tools are available to encode texture and depth maps. The Common Test Conditions (CTC) for 3D-HEVC presented in the Appendix A were adopted as the experimental setup.

Analyzing the time profiling on the left-hand side of Figure 6.1, it is possible to note that the texture coding represents only 13.7% of the encoder computational effort, whereas depth maps coding represents 86.3% when considering the AI scenario. Detailed profiling of depth coding time is also presented in Figure 6.1, where DMM-1 is the most time-consuming encoding tool being responsible for 27.2% of the depth maps intra computational effort, followed by residual encoding tools with 26.6% and 25.3% for TQ (Transform/Quantization) and SDC, respectively. The “Intra HEVC” considers the evaluation of the prediction modes inherited from HEVC texture coding and represents 12.1% of the time spent in depth maps intra coding. The remaining encoding modes together, DMM-4 and DIS, represent less than 9%. Even though these modes do not represent high encoding time when compared to other tools, they can provide significant improvements in 3D-HEVC encoding efficiency (SANCHEZ *et al.*, 2018b).

These results can be explained since depth maps are composed of large homogenous regions and the DIS typically fits better with these kinds of regions; however, depth maps also present edge and gradient regions, where DMMs and Intra HEVC are crucial to provide high encoding efficiency.

Figure 6.1 – Computational effort distribution in 3D-HEVC Intra-frame prediction



Source: Author.

Therefore, according to the data presented in Figure 6.1, one can conclude that depth maps intra coding is a costly computational task for the 3D-HEVC encoder. The highest computational effort is mainly concentrated in DMM-1 and residual encoding flows. In order to reduce this computational effort, some works proposed to avoid the entire DMM-1 evaluation (SANCHEZ *et al.*, 2015a) and decrease the number of modes evaluated in residual encoding flow (SANCHEZ *et al.*, 2017a).

However, when hardware architectures able to achieve real-time processing with low power dissipation are considered, new challenges can be noticed. For instance, in the residual coding flow, limited parallelism can be exploited since these steps share data with entropy encoding that works with a sequential approach. Besides, in DMM-1 the number of wedgelets to be evaluated depends on the block size being predicted (1,908 in total – see Section 4.4.1) and these wedgelets must be stored in memory to be evaluated.

Many wedgelets patterns for both DMM-1 encoding and decoding processes are allowed. This storage of all wedgelets patterns is undesirable, mainly considering 3D-HEVC systems implemented in dedicated hardware. As discussed before, DMM-1 requires 183,264 bits to store all wedgelets patterns (ZHANG *et al.*, 2014) (IKAI *et al.*, 2015) of all block sizes supported by the encoding tool (i.e., square-shaped block sizes ranging from 4x4 to 32x32), according to the 3D-HTM implementation. Also, the DMM-1 algorithm requires continuous access to the wedgelets patterns on memory and intensive processing, which demands high energy consumption. Thus, to obtain performance and energy consumption compatible with real-time processing of 3D-HEVC high definition videos, it is necessary to reduce memory size, memory accesses and processing effort related to DMM-1 calculations.

6.1.2 3D-HEVC Inter-frames and Inter-view Predictions

ME and DE are among the costly steps of video encoders regarding energy and memory (AFONSO *et al.*, 2019), as previously discussed. Considering the similar behavior of ME and DE in the Inter-frames and Inter-view predictions, these two encoding tools will be analyzed together in this section.

Both ME and DE encoding tools are evaluated in the encoder at the level of PU (Prediction Units) blocks, and the 3D-HEVC allows ME/DE to be performed with 24 block sizes, ranging from 4x8 up to 64x64 pixels in order to improve coding efficiency. In this scenario, the selection of the best match is a costly evaluation process, which is required to achieve the best tradeoff between compression and distortion. This way, ME/DE demands a huge number of block matching operations leading to intense processing, memory communication and, consequently, high energy consumption.

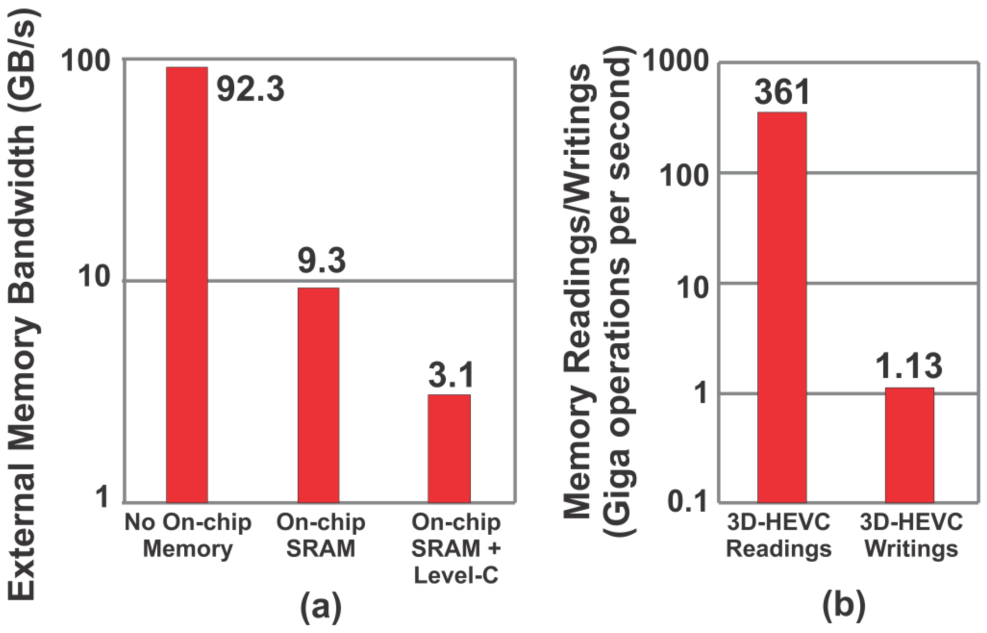
Memory access is one of the main components regarding energy consumption and performance considering ME/DE hardware designs due to the number of candidate blocks to be compared by the BMA (ZATT et al., 2011b). Data-reuse schemes such as the Level-C (CHEN *et al.*, 2006) and other dedicated solutions (ZATT et al., 2011a) (ZATT et al., 2011b) (SAMPAIO et al., 2013) have been used in ME and ME/DE to reduce the memory bandwidth, as previously presented.

The work (AFONSO *et al.,* 2019) presents an evaluation of the external memory traffic considering 3D-HEVC encoding of three-view (three texture pictures plus their respective depth maps) HD (High Definition) 1080p MVD videos using HOTZS (Hardware-Oriented TZS) static scheduling and HDS (Horizontal Disparity Search) fast block-matching algorithm (this thesis details this work in Section 8.1). In this work, search patterns much less complex than the original TZS algorithm were adopted and these algorithms considered only 32x32 and 16x16 block sizes. When no on-chip SRAM is employed, i.e., the ME/DE processing unit communicates directly to the external memory, the demanded communication for real-time encoding would reach 92GB/s, as demonstrated in Figure 6.2. Given that current memory technologies such as the embedded low-power memories provide no more than 25.6GB/s in ideal cases (MICRON, 2014), this target performance is unfeasible. Considering the use of an on-chip SRAM to prefetch and store the SW used for the block-matching (360kB), the communication is reduced to 9.3GB/s. When the Level-C scheme (see Section 3.3.2.1) is adopted in addition to the on-chip SRAM memory, the memory bandwidth drops to 3.1GB/s, leading to a total communication reduction of 96%, as depicted in Figure 6.2. Nevertheless, the energy consumption related to the external memory communication, and on-chip SRAM dynamic/static consumption is still high, and it hinders the implementation of 3D real-time systems, mainly focusing on handheld devices.

Regarding memory readings and writings, the work (AFONSO *et al.*, 2019) estimated the number of DE/ME memory operations considering the 3D-HEVC encoding of HD 1080p MVD videos with the 24 possible PU sizes (full RDO cost). The number of encoded views was defined in order to attempt a system capable of delivering nine views after the decoding process. 3D-HEVC requires 1.13×10^9 memory writings and 361×10^9 memory readings per second to encode three HD 1080p views (texture plus depth maps).

Although feasible, the energy consumption related to memory hierarchy remains high due to external memory communication and on-chip SRAM dynamic and static consumption. Thus, a memory hierarchy featuring on-chip SRAM storage is mandatory. However, there is still a need for further reducing external memory communication, reducing on-chip SRAM size (to reduce static/leakage consumption), and managing the memory hierarchy.

Figure 6.2 – (a) ME/DE external memory bandwidth for three memory organizations (without on-chip memory, with on-chip SRAM and with on-chip SRAM using the Level-C data reuse scheme); and (b) ME/DE memory reading and writing for 3D-HEVC



Source: Author.

Computational effort analyses considering 3D-HEVC ME/DE steps are incipient in the literature. Actually, some works as (AFONSO *et al.*, 2016) evaluated ME computational effort considering a 2D approach with the HEVC Reference Software (HM – HEVC Test Model) using the Common Test Conditions for HEVC standard (2D videos) (BOSSEN, 2013), but the presented results cannot be extrapolated to the ME/DE behavior in a 3D-HEVC context along with MVD format. Therefore, this chapter presents an evaluation of the computational effort related to the ME and DE steps as well as the relation between ME, DE, and Intra-frame prediction and the other tools used in the 3D-HTM. The experiments considered the RA (Random-Access) configuration (BOSSEN, 2013), and two of the video sequences (*Balloons* and *Undo_Dancer*) defined in CTC (MÜLLER et al., 2014) with two QP (Quantization Parameter) values (30 and 39) also defined in CTC. It is important to emphasize that, when considering the RA configuration, Inter-frames prediction, Inter-view prediction, and Intra-frame prediction tools are available to encode texture and depth maps.

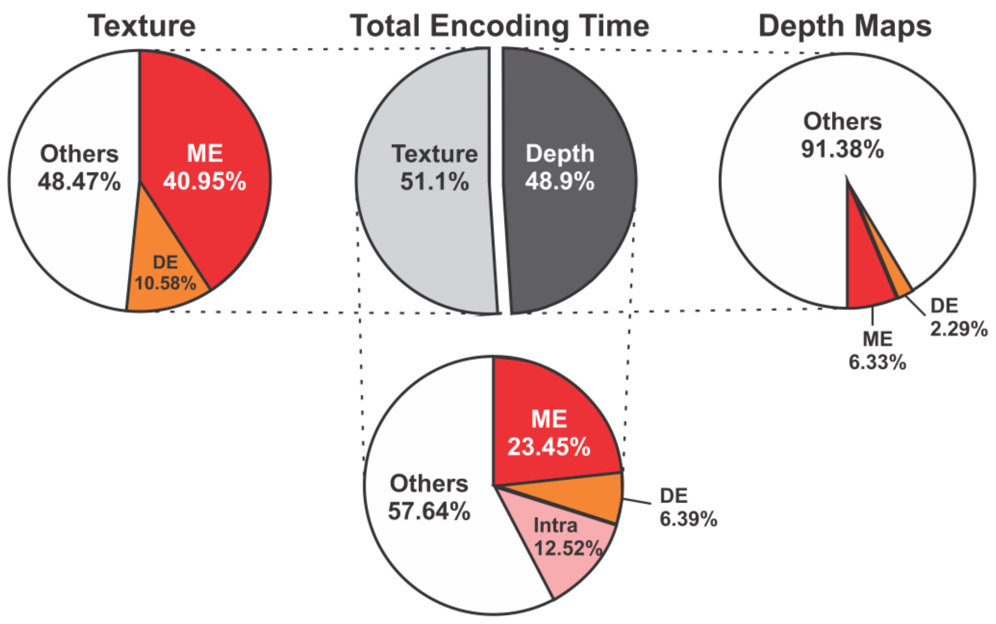
As one can notice in Figure 6.3, the time spent with each one of the channels (texture and depth maps) is similar considering an encoder configuration that uses the ME/DE steps, so texture used 51.1% of the encoding time whereas depth maps used 48.9%.

Regarding the texture channel, a significant portion of the computational effort is due to the ME/DE steps, where ME is responsible for 40.95% of the spent time, DE uses 10.58% of encoding effort, and the other encoding tools (Intra HEVC, Transform, Quantization, etc) use the remaining 48.47% of the time, as depicted in Figure 6.3. Considering the depth maps, the portion of computational effort due to the ME/DE is much smaller, where ME uses 6.33% of the time, DE uses 2.29% of the time, and the other encoding tools use 91.38% of the encoding time (Intra HEVC, DMM-1, DMM-4, DIS, Transform, Quantization, SDC, etc). This behavior is expected since Intra-frame prediction tools are available in the RA configuration and the depth maps tends to be more efficiently encoded when using the novel Intra-frame prediction tools designed for this type of information, as discussed in the previous subsection (Section 6.1.1).

Figure 6.3 also presents the total encoding time related to ME/DE and Intra-frame prediction. Note that since ME/DE steps increase the complexity of the texture encoding and the novel 3D-HEVC intra encoding tools increases the complexity of the depth-map encoding, the time spent with predictions are significant considering all the encoding process. ME/DE predictions spent 29.84% of the time, in which Intra-frame prediction spent 12.52% of the time. The remained encoding tools spent 57.64% of the encoding time.

The memory and computational effort results presented and discussed in this chapter for both the novel depth 3D-HEVC intra prediction tools and the 3D-HEVC ME/DE steps evidenced the need for the development of complexity-reduction strategies and VLSI designs focusing on both 3D-HEVC Intra-frame prediction, and Inter-frames and Inter-view predictions.

Figure 6.3 – 3D-HEVC computational effort distribution when running in RA configuration



Source: Author.

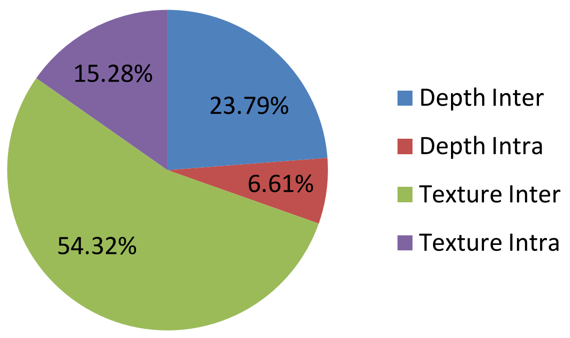
6.2 3D-HTM Encoding-tool Analyses

This Section presents a wide encoding-tool analysis through the 3D-HTM reference software, focusing on both the CCO and FCO configurations employed in the 3D-HEVC extension. The results for each 3D-HEVC coding-order approach are presented in the following, and they are also divided into subsections, one related to the Inter-frames and Inter-view predictions and another to the Intra-frame prediction. All results consider the experimental setup presented in Appendix A and the RA temporal configuration. The encoding tools are evaluated from the percentage of pixels encoded with them, called of "Representativeness" in this thesis. It is important to notice that each texture picture has an associated depth-map and, therefore, the same number of pixels is encoded by each channel considering a complete video. The difference is in the fact that depth maps have only luminance samples while texture pictures have three different samples: luminance, chrominance blue, and chrominance red.

6.2.1 3D-HTM Encoding-tool Analyses Based on the CCO Configuration

As previously explained, the Conventional Coding Order (CCO) configuration adopts the encoding of the texture pictures before their associated depth maps for all views that composes the 3D-video sequences. Figure 6.4 shows the selection of Inter (Inter-frames and Inter-view predictions) and Intra encoding tools according to the PUs for both texture pictures and depth maps. Note that the number of PUs is different for each channel because the depth maps tend to use bigger PU sizes than the texture pictures. This way, the texture channel uses much more PUs to be encoded. Texture encodes 69.6% of the PUs whereas depth maps use 30.4% of the PUs. Therefore, to analyze only the selection of the PUs according to the encoding tools can bring some imprecision to the assessment. For this reason, the percentage of pixels encoded with the encoding tools, here called of "Representativeness", was prioritized for the 3D-HTM Encoding-tool analyses.

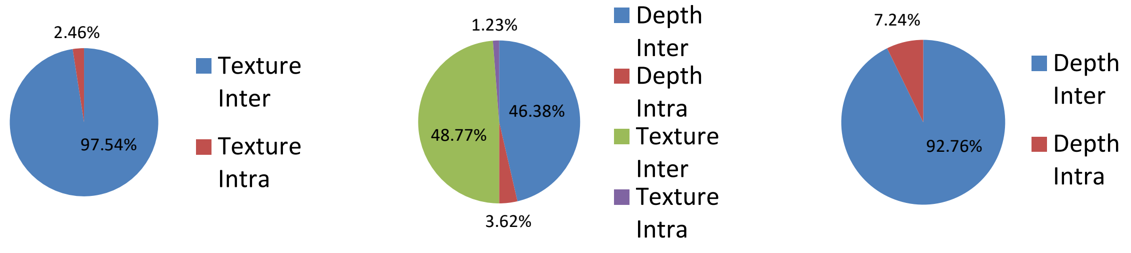
Figure 6.4 – Inter prediction (Inter-frames and Inter-view), and Intra-frame PU selection with the CCO approach



Source: Author.

The results obtained with the CCO approach were divided according to the type of prediction, Intra-frame or Inter prediction. As presented in Figure 6.5, the representativeness of the Inter prediction (Inter-frames and Inter-view predictions) is predominant for both the texture and the depth-map pictures. Considering the texture frames (Figure 6.5-a), 97.54% of the pixels are encoded by Inter prediction while Intra-frame prediction is responsible for encoding only 2.46% of the pixels. In the depth maps (Figure 6.5-c), 92.76% of the pixels are encoded with Inter prediction, and the Intra-frame prediction encodes 7.24% of them. It is important to notice that intra-encoded frames are applied to encode the first frame and a new frame at every second approximately to guarantee the references during the encoding process of the other frames. Furthermore, Intra-frame prediction improves the compression gains in some specific situations, such as the coding of object edges. The coding of sharp edges is even more important in an MVD context, where high-quality depth-map coding is needed to efficiently synthesize the intermediate texture views.

Figure 6.5 – Inter and Intra-frame representativeness with a CCO approach: a) Texture only; b) Both texture and depth; and c) Depth-map only



(a) (b) (c)

Source: Author.

Regarding encoding time, Inter prediction is well-known as the most costly step of the encoders (ZATT *et al.*, 2013). However, in an MVD context, the Intra-frame prediction is also responsible for an important part of the complexity. The work (SANCHEZ *et al.* 2018b) states that 34.51% of the depth-map encoding time is due to the Intra-frame prediction in 3D-HEVC, which surpass the Inter prediction considering only depth maps (about 19.74%). This way, complexity reduction strategies are important for both the Inter and the Intra-frame predictions of both texture and depth-map coding considering the 3D-HEVC extension. The next subsections present the results related to the Intra-frame and Inter prediction encoding tools separately.

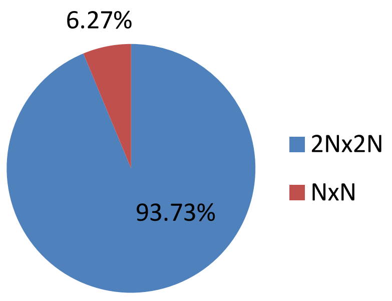
6.2.1.1 Intra-frame Prediction

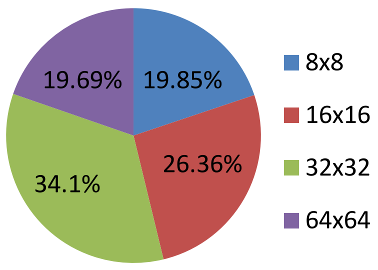
This subsection presents in details the results obtained from the experiments considering only the Intra-frame prediction and its respective modes. Since the texture and the depth-map pictures present specific characteristics, and the Intra-frame prediction of depth-maps introduces other prediction modes besides the ones used in the texture pictures, the experimental results for the Intra-frame prediction were divided into Texture and Depth-maps to better explore their characteristics and allow the identification of efficient complexity-reduction strategies. The results related to the Texture are showed and discussed in the next subsection. Afterward, the data related to the depth maps are presented.

6.2.1.1.1 Texture

According to the experiments performed in this work, 93.73% of the pixels are encoded with 2Nx2N partitions when only Intra-frame prediction and texture pictures are considered, as presented in Figure 6.6-a. The NxN partitions are responsible for encoding the other 6.27% of the pixels (used only in the fourth quad-tree level). Figure 6.6-b also shows these data according to the quad-tree level at which these partitions occurred. Considering the results, no CU-depth level presented negligible results, encoding between 19.69% and 34.10% of the Intra-predicted texture pixels. Therefore, Intra-frame prediction simplifications such as to disable a given partition and/or a quad-tree level tend to present important impacts when applied to texture pictures, i.e., compression and image-quality losses.

Figure 6.6 –Intra-frame prediction representativeness with a CCO approach considering texture pictures: a) According to the partition type; b) According to the CU-depth level



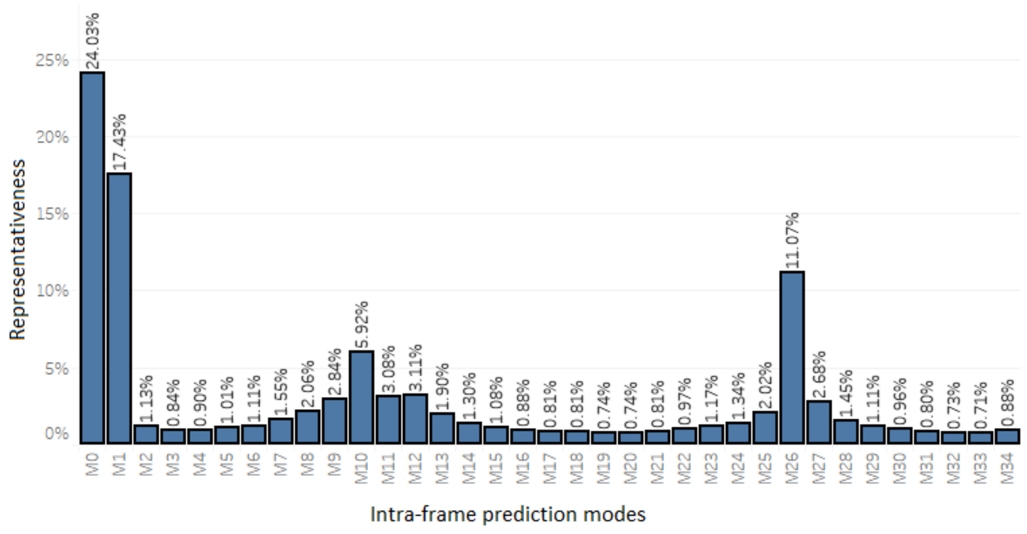


(a) (b)

Source: Author.

Other data arrangement was used to verify the number of Intra-predicted pixels that were encoded with all 35 modes supported in the texture coding (Planar, DC, and Angular modes). Figure 6.7 shows the percentage of pixels related to each one of these modes. The Planar mode (mode 0 according to the 3D-HTM) is the most important mode according to the results since it encodes 24.03% of the pixels, followed by the DC mode (mode 1), Vertical mode (mode 26), and Horizontal mode (mode 10) with 17.43%, 11.07%, and 5.92%, respectively. Anyone of the other modes encodes up to 3.11% of the pixels. Also, the four modes that encode more pixels account for 58.45% of the pixels, which is an important result in a scenario with 35 modes. The next subsection presents the Intra-frame prediction results considering depth maps.

Figure 6.7 –Intra-frame prediction representativeness considering texture pictures according to the encoding mode with a CCO approach



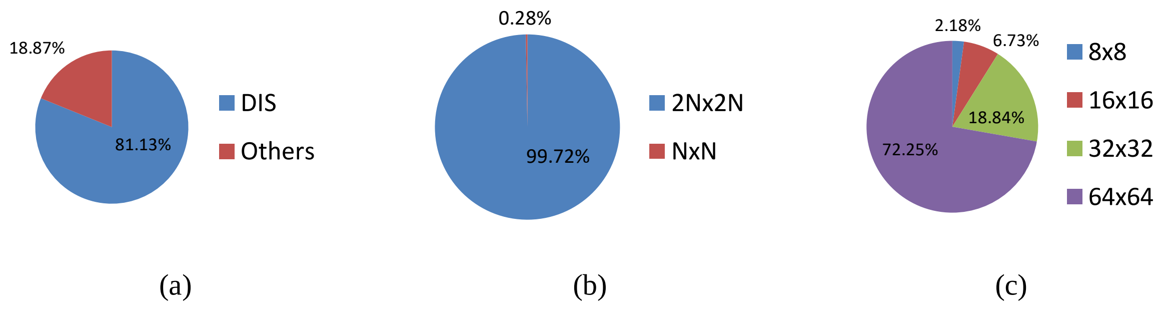
Source: Author.

6.2.1.1.2 Depth Maps

This subsection presents and discusses the results correspondent to the 3D-HEVC Intra-frame prediction and depth-map pictures.

In the case of the depth-map coding with Intra-frame prediction, it is necessary to analyze the DIS mode firstly. As previously explained, the DIS mode was introduced in 3D-HEVC to encode depth-maps, and it tends to present better results in this type of picture. Figure 6.8-a separately shows the percentage of pixels encoded with the DIS mode and the other Intra-frame tools for depth maps. It is important to notice that the DIS mode is widely used, encoding 81.13% of the Intra-predicted depth-map pixels (used in 2Nx2N partitions only because it is used at the CU level). Considering the remaining pixels and modes, 18.87% of the pixels are encoded distributed for 37 Intra modes for depth maps (35 HEVC modes and two depth modeling modes).

Figure 6.8 – Intra-frame representativeness with a CCO approach for depth maps: a) Using DIS tool; b) According to the partition type; c) According to the CU-depth level



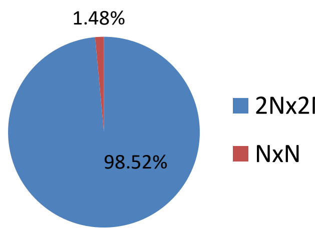
Source: Author.

To better evaluate the Intra-predicted depth-map results and, given the wide usage of the DIS mode, the following results disregard the DIS mode.

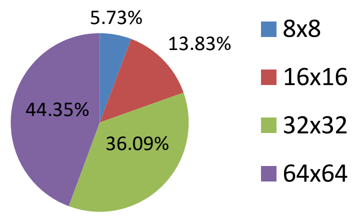
Even disregarding the DIS mode by considering only the other modes, the NxN partitions present an insignificant impact in the coding of depth maps, accounting for about 1.48% of the pixels, as presented in Figure 6.9-a. However, considering the CU depth selection, the DIS mode pushes up the 64x64 CU-depth level representativeness when compared to the remained CU depths, as one can observe comparing Figure 6.8-c and Figure 6.9-b.

Figure 6.9-b shows the representativeness of the Intra-predicted depth maps according to the CU-depth level. As presented in Figure 6.9-b the depth-maps behavior is so different when compared with the CCO Intra-predicted pixels considering texture pictures. While in the CCO texture pictures, the Intra-predicted blocks are usually encoded with all quad-tree levels almost equally distributed, in the depth maps, the encoding with smaller block partitions is less frequent than the encoding with bigger partitions. Furthermore, considering the DIS mode, this predominance is even higher as previously presented (Figure 6.8-c). Nevertheless, conclusions about the importance order of the CU-depth levels in depth maps must be carefully drawn due to the different aspects of the depth maps in which borders must be preserved.

Figure 6.9 – Intra-frame prediction representativeness with a CCO approach for depth maps (except DIS) according to: a) Partition mode; b) CU-depth level



(a)

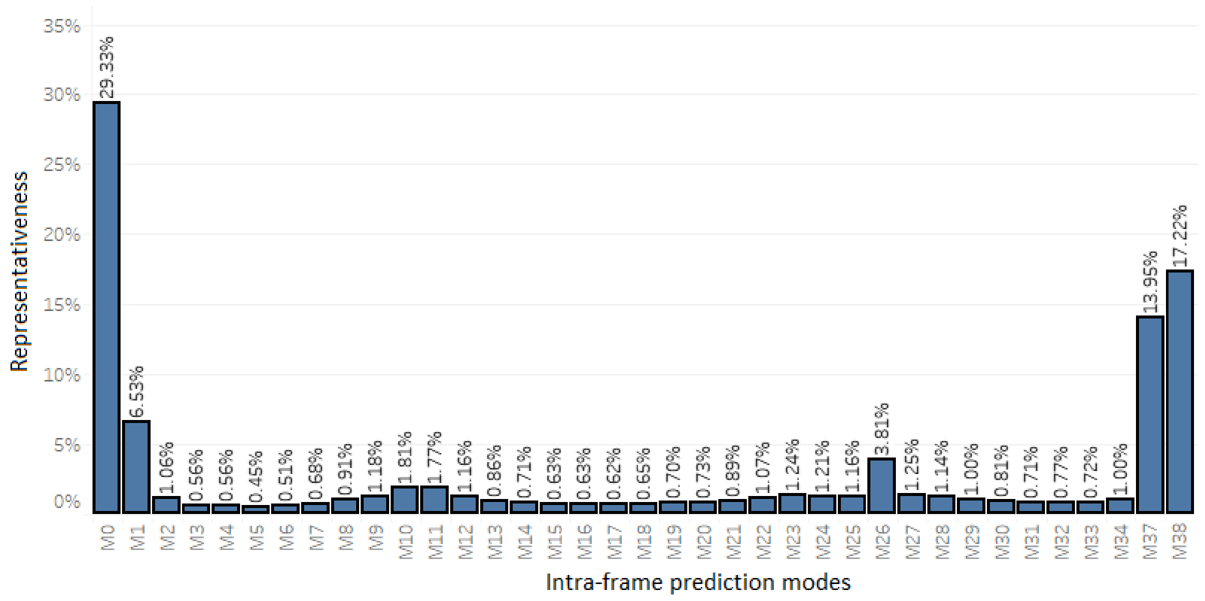


(b)

Source: Author.

Also, the depth-map modes more important in the encoding process were evaluated. As the DIS mode is the most representative Intra-frame depth-map mode, this mode was disregard in this analysis. Figure 6.10 shows the representativeness of the Intra-frame depth-map modes at which the four most representative modes are Planar (mode 0), DMM-4 (mode 38), DMM-1 (mode 37), and DC (mode 1), with 29.33%, 17.22%, 13.95%, and 6.53% of representativeness, respectively. These four modes together are responsible for 67.03% of the Intra-predicted depth-map pixels disregarding the DIS mode. It is important to notice that no other mode encodes more than 3.81% of the pixels, as well as, 37 modes can be employed in the depth-maps disregarding the DIS mode (the same texture modes and two DMM modes). Also, when these four most representative modes and the DIS mode are considered, they are responsible for 93.78% of the encoded pixels. By including the six more representative modes, which includes the Horizontal (mode 10) and Vertical (mode 26) modes, the percentage of encoded pixels increases to 72.65% disregarding the DIS mode and increases to 94.84% when considering the DIS mode. The DMM modes (mode 37 and 38) were introduced in the 3D-HEVC specially to deal with depth-map pictures, and they present important results in the compression. However, these modes are associated with a large complexity so that they are not applied in the first CU-depth level (64x64 CUs).

Figure 6.10 –Intra-frame prediction representativeness considering depth-maps according to the encoding mode (except DIS) with a CCO approach



Source: Author.

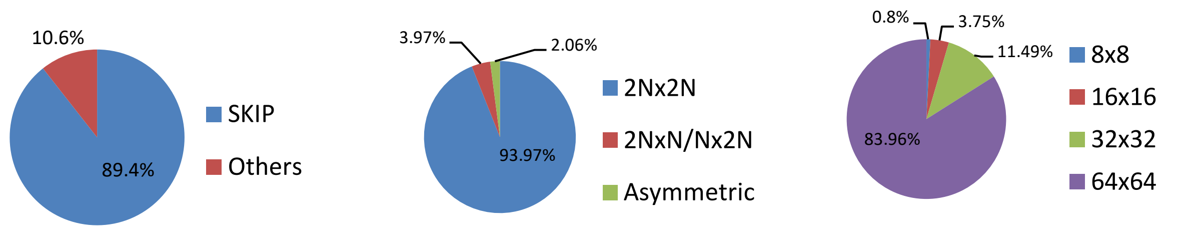
6.2.1.2 Inter-frames and Inter-view Predictions

This subsection presents the results focusing on the CCO approach and in the Inter-frames and Inter-view predictions. As previously mentioned, given the different aspects related to the Texture and Depth Maps, the encoding process of each type of picture adopts different choices along the encoding process. This way, the results are also presented separately in two subsections to Inter prediction, the first considering texture pictures, and the second, depth maps.

6.2.1.2.1 Texture

As previously mentioned, about 95.15% of the pixels are encoded using the Inter Prediction (Inter-frames and Inter-view predictions), as presented in Figure 6.5. Considering only CCO Inter-predicted pixels and the texture pictures, the percentage of pixels is even higher, about 97.54%. As previously explained, the Inter prediction presents several modes and partitions. However, this expressive representativeness of the Inter prediction is mainly due to the Skip mode, as presented in Figure 6.11-a. The percentage of Inter-predicted texture pixels considering the Skip mode is 89.40%, which is applied to any CU-depth level of 2Nx2N partitions, dealing the 2Nx2N partition to reach 93.97% of the pixels, as presented in Figure 6.11-b. The representativeness of the Inter prediction according to the CU-depth level can be observed in Figure 6.11-c. When the results are evaluated considering the quad-tree level of the CUs, it is possible to observe that the first CU-depth level is responsible for encoding 83.96% of the CCO Inter-predicted texture pixels. Considering the two initially quad-tree levels, the percentage of pixels is 95.44% and, if the three first quad-tree levels are considered, this percentage is 99.2%. This way, the last quad-tree level (8x8 CUs) is responsible for only 0.8% of the encoded pixels. The 64x64 CU-depth level corresponds to the major part of the pixels pushed up by the Skip mode, as well as the 2Nx2N partition.

Figure 6.11 – CCO Inter prediction representativeness considering texture pictures: a) Using Skip mode; b) According to the partition types; c) According to the CU-depth level

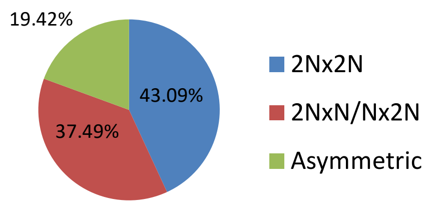


(a) (b) (c)

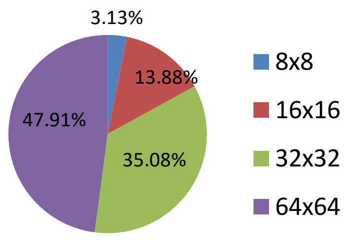
Source: Author.

By analyzing the results regarding the selected partitions and CU-depth levels, the huge representativeness of the Skip mode deals the 2Nx2N partition and 64x64 CU-depth level to encode the major part of the Inter-predicted texture pixels, as previously discussed. This way, the data were also organized disregarding the Skip and detailing the partitions and CU-depth levels selected with the other modes, as presented in Figure 6.12. With this data arrangement showed in Figure 6.12, it is possible to verify that the DBBP, Merge, ME/DE encoding tools have a considerable use of the 2NxN and Nx2N partitions. Despite these partitions being less representative than 2Nx2N partitions considering the CCO Inter prediction of texture pictures, they can present important impacts on the image quality and compression. It is important to notice that 2Nx2N, 2NxN, and Nx2N partitions are responsible for 97.94% of the total CCO Inter-predicted texture pixels (or 80.65% disregarding Skip mode – see Figure 6.12-a). However, the importance of the smaller CU-depth levels is notable disregarding the Skip mode since the 8x8 and 16x16 CU depths are responsible for encoding 17.01% of the pixels even covering a smaller area per block, as depicted in Figure 6.12-b.

Figure 6.12 – CCO Inter prediction representativeness considering texture pictures (except Skip): a) According to the partition type; b) According to the CU-depth level



(a)

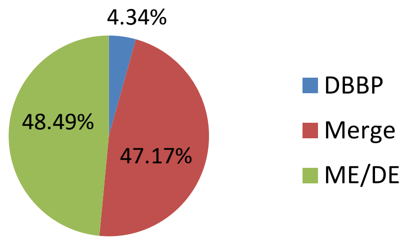


(b)

Source: Author.

In Figure 6.13, the data are presented disregarding the Skip mode but detailing the other encoding tools. It is possible to notice that Merge and Disparity/Motion Estimations present a similar behavior, whereas encodes a significant area when compared to the DBBP mode. Whereas the Merge and ME/DE encodes 47.17% and 48.49% of pixels, respectively, the DBBP is responsible for encoding 4.34% of the pixels. Note that the DBBP mode is only applied to the 2NxN and Nx2N PU partitions, and to the 64x64, 32x32, 16x16 CU depths. The next subsection focuses on the results for the CCO Inter prediction considering depth-map pictures.

Figure 6.13 – CCO Inter prediction representativeness of texture pictures according to the encoding mode (except Skip)



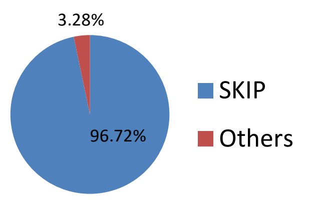
Source: Author.

6.2.1.2.2 Depth Maps

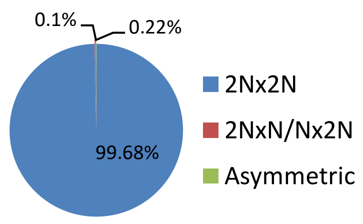
Considering the depth-map encoding process, the CCO Inter prediction corresponds by 92.77% of the encoded pixels. Similarly, with the Inter prediction of the texture, this important value is mainly due to the Skip mode. In the depth maps, 96.72% of the CCO Inter-predicted pixels used Skip mode, as presented in Figure 6.14-a. Note that the Skip mode can be applied to any quad-tree level of partitions 2Nx2N.

Regarding partition types, the 2Nx2N partition is responsible for encoding 99.68% of the CCO Inter-predicted depth-map pixels, as presented in Figure 6.14-b. By analyzing the data regarding the quad-tree level at which the blocks were encoded in Figure 6.14-c, it is possible to conclude that the absolute majority of pixels are encoded with the initially quad-tree levels. The 64x64 and 32x32 CU depths are responsible for encoding 99.27% of the pixels. In depth maps, this predominance of areas encoded with 2Nx2N partitions and CU-depth levels with larger CU sizes tends to be pushed up by the Skip mode, as well as in the texture pictures. This way, an evaluation disregarding the Skip mode is presented in the following.

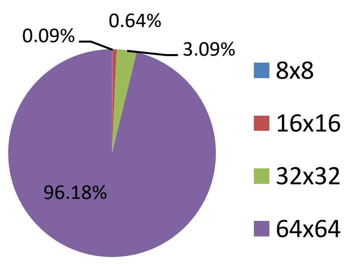
Figure 6.14 – CCO Inter prediction representativeness considering depth maps: a) Using Skip mode; b) According to the partition type; c) According to the CU-depth level



(a)



(b)

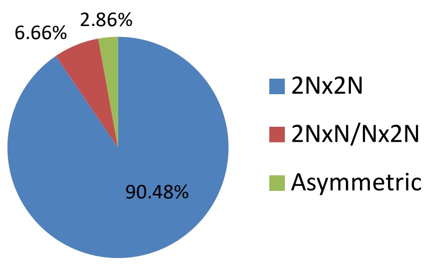


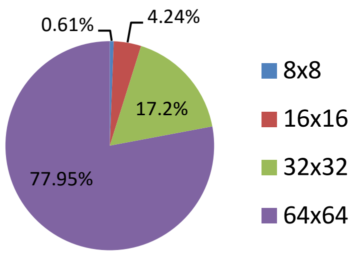
(c)

Source: Author.

A data arrangement that disregards the Skip mode is presented in Figure 6.15. This way, the predominance of the 2Nx2N partitions can be really verified for the other encoding tools. Even disregarding the Skip mode, the 2Nx2N partition encodes 90.48% of the CCO Inter-predicted depth-map pixels. Similarly, even disregarding the Skip mode, the 64x64 and 32x32 CU depths are responsible for encoding 95.15% of the pixels. In the sequence, a further analysis according to the encoding mode is presented.

Figure 6.15 – CCO Inter prediction representativeness considering depth maps (except Skip): a) According to the partition type; b) According to the CU-depth level



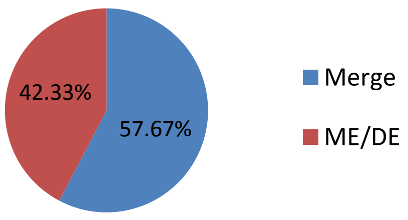


(a) (b)

Source: Author.

The results disregarding the Skip mode are also separated according to the encoding tools, and they are presented in Figure 6.16. Over again, it is possible to observe that the tool behaviors, for depth maps, are similar with and without the Skip mode. Merge mode encodes more pixels than ME/DE encoding tools, but both Merge, and ME/DE steps are significant in the encoding process.

Figure 6.16 – CCO Inter prediction representativeness of depth maps according to the encoding mode (except Skip)



Source: Author.

6.2.1.2.3 Block-Size Analysis

As previously mentioned, the major part of the computational effort of the 3D-HEVC is due to the decision of which methods of encoding and PU sizes must be used in the Inter prediction, since until 24 PU sizes can be evaluated during the encoding process if the full RDO cost was considered. All these 24 PU sizes must be processed by other encoding tools (Transforms and Quantization, for instance) to define which size presents the best tradeoff between compression and image quality. The Merge-based encoding along with Motion and Disparity Estimations are important steps in the current video coding standards due to the compression rates propitiated by them. To reach such performance, Merge mode, Motion and Disparity Estimations support 24 PU sizes.

In conclusion, this process has a high cost, and a reduction in this computational effort is highly desirable to allow energy-efficient and real-time systems. To allow the identification of computational-effort reduction strategies for the Inter prediction (both Inter-frames and Inter-view predictions), block-size analyses are presented in the following for both texture and depth-maps. These analyses disregarded the Skip mode (less complex mode applied to only four block sizes) and they focused on the more complex steps.

It is possible to infer that a simple way to reduce the computational effort is reducing the PU sizes that must be compared in the Inter prediction, mainly due to the ME/DE steps. However, the real impact in terms of compression and image quality of using some specific PU must be evaluated. To support this idea, the incidence of each PU size in the Inter prediction and its representativeness on the frames were investigated. Hence, the 3D-HTM code was modified with the aim of extracting those data.

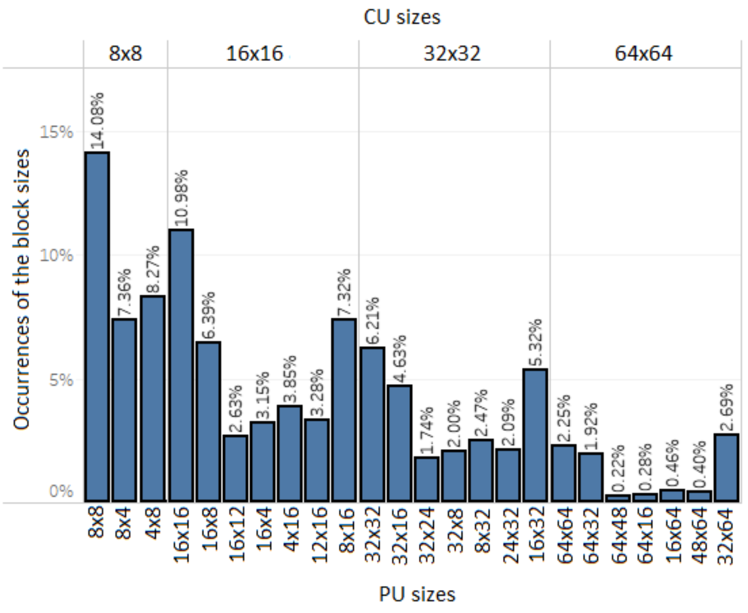
All the eight sequences defined by the CTC were encoded in the RA temporal configuration and according to the configurations recommended in that document. Figure 6.17 shows the percentage of PU sizes selection in the Inter prediction, on average, considering the texture pictures. The values are presented separately for all 24 PU sizes according to the CU-depth level. Considering only the data referred to the ME/DE and Merge steps (and DBBP mode for the texture), i.e., disregarding Skip mode, the 8x8 PU size is the most frequently selected block size with 14.08% of selections and the second most often selected size is the 16x16 with 10.98% of selections. Note that the other two square-shaped block sizes allowed in the Inter prediction (32x32 and 64x64 PU sizes) are poorly selected when compared to other sizes (seventh and fifteenth more selected sizes only).

The percentage of PU sizes selection suggests that some PU sizes such as the 8x8 PU size have great importance during the coding process. However, since bigger PUs are more representative in the image, evaluate the percentage of pixels that were covered by each PU size is important. Bigger PUs, such as the 16x16, even being less frequent, may cover a larger area and, therefore, they can be more relevant to the coding process.

To further evaluate this hypothesis, the data about the selection of the PU sizes were adjusted considering the image representation of each PU size. This analysis, as depicted in 6.18, shows that bigger sizes (such as 64x64 and 32x32) are more representative in the video sequences, even being less frequent and even when the skip mode is not considered. Whereas 64x64 PU size is the most representative size encoding 20.61% of the pixels, 32x32 PUs are the second most representative PUs by encoding 14.2% of the pixels. Note that the square-shaped PU sizes are the most representative sizes in each one of the CU-depth levels. Figure

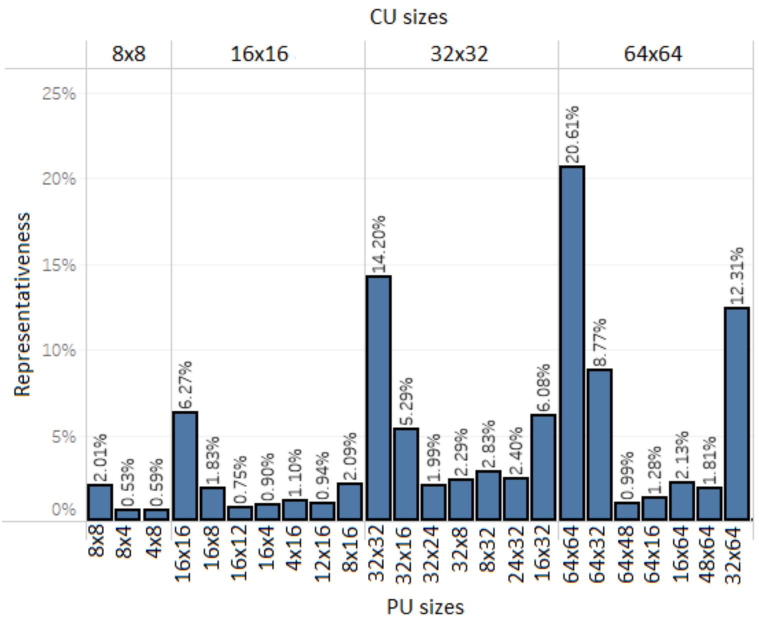
6.17 and 6.18 show that square-shaped PUs are both frequent and representative when compared to the non-square-shaped PUs in the texture picture encoding process. Note that 8x8 PU size is the most frequent and the 16x16 PU size is the second most frequent, whereas the square-shaped sizes 64x64, and 32x32 are the most representative sizes.

Figure 6.17 – Occurrences of the block sizes in the CCO Inter prediction considering texture pictures (except Skip mode)



Source: Author.

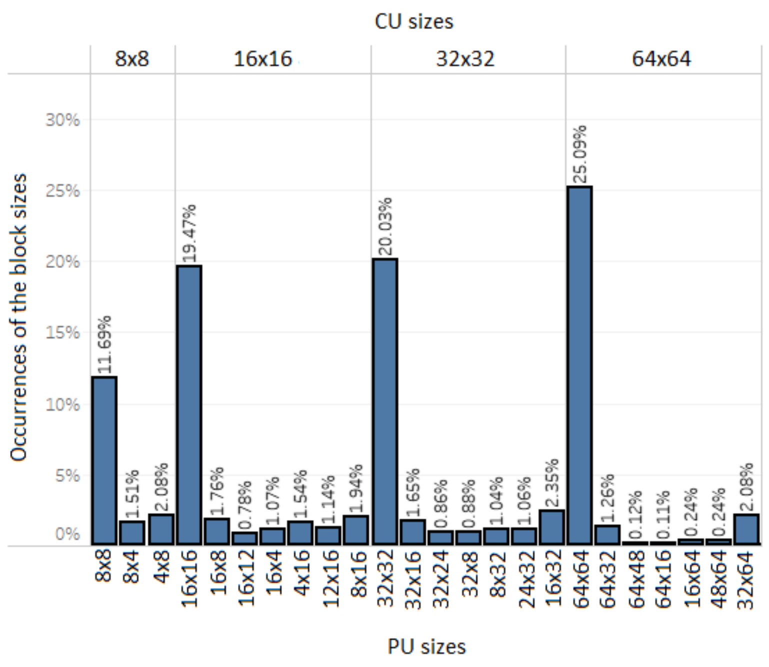
Figure 6.18 –Representativeness of the block sizes in the CCO Inter prediction considering texture pictures (except Skip mode)



Source: Author.

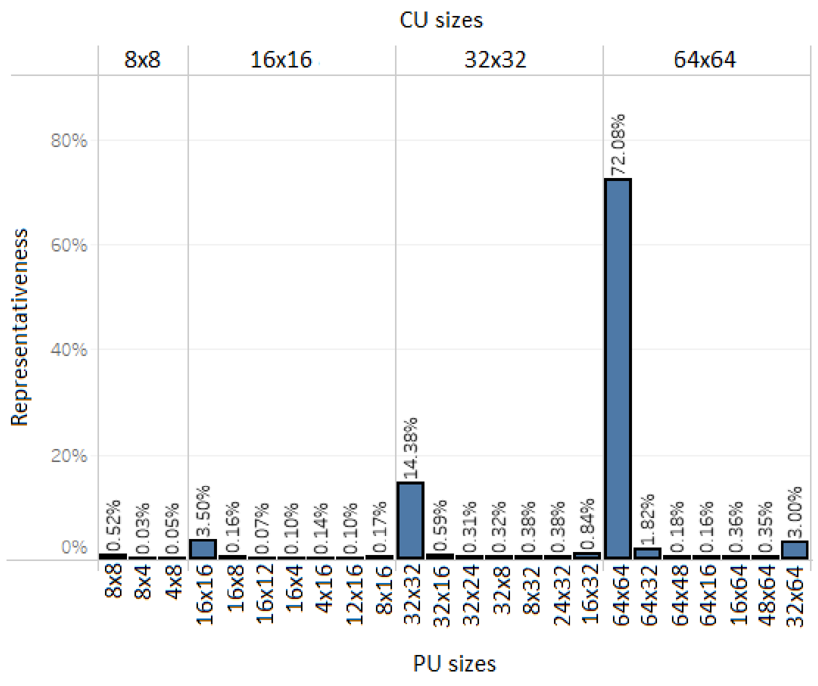
Figure 6.19 shows the average percentage of PU sizes selection in the Inter prediction considering the depth maps. The values related to the depth maps are presented separately for all 24 PU sizes according to the CU-depth level. Considering only the data referred to the ME/DE and Merge steps, i.e., disregarding Skip mode, the 64x64 PU size is the most frequently selected block size with 25.09% of selections, followed by the other three square-shaped block sizes allowed in the Inter prediction, 32x32, 16x16, and 8x8 PU sizes. The 32x32 PU size is the second most often selected size with 20.03% of selections, the 16x16 PU size is the third most often selected PU size with 19.47% of selections, and the 8x8 PU size is the fourth most often selected PU size with 11.69% of selections. Note that this behavior is different from that obtained in the texture pictures since bigger PUs are more selected when considering depth maps. Furthermore, it is possible to observe that the four square-shaped block sizes cover 76.28% of the selections related with depth maps, and the other 23.72% are distributed among 20 PU sizes. For depth maps, the data about the PU sizes selection were also adjusted considering the representativeness. Figure 6.20 depicts the percentage of pixels that were encoded by each PU size. This analysis shows that the three bigger square-shaped sizes 64x64, 32x32, and 16x16 are the three most representative sizes, encoding 72.08%, 14.38% and 3.5% of the pixels, respectively. The other square-shaped PU size, the 8x8, is the eighth most representative size. Note that the square-shaped PU sizes are the most representative sizes in each one of the CU-depth levels and, together, they cover 90.48% of the encoded pixels considering depth maps. Figure 6.19 and Figure 6.20 showed that square-shaped PUs are both frequent and representative when compared to the non-square-shaped PUs in the depth-map encoding process. Note that the three bigger square-shaped sizes, 64x64, 32x32, and 16x16 PU sizes, are both the three most frequent and the three most representative sizes whereas the 8x8 PU size is the fourth most frequent.

Figure 6.19 – Occurrences of the block sizes in the CCO Inter prediction considering depth maps



Source: Author.

Figure 6.20 – Representativeness of the block sizes in the CCO Inter prediction considering depth maps



Source: Author.

The 3D-HEVC evaluations allowed verifying the occurrences of the PU sizes that are most selected and most representative during the encoding considering the CCO Inter prediction. From these evaluations, it is possible to note that the square-shaped PU sizes have the two most selected sizes (16x16, and 8x8) and they have the two most representative sizes

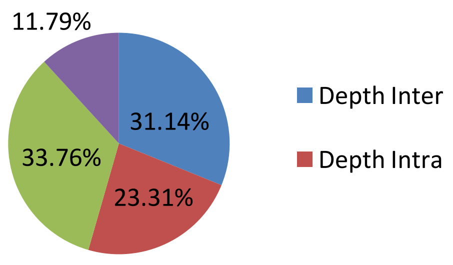
(64x64, 32x32) considering the texture pictures. Considering depth maps, it is possible to note that the square-shaped PU sizes are the four most selected sizes (64x64, 32x32, 16x16, and 8x8) and they have the three most representative sizes (64x64, 32x32, and 16x16). Finally, the four square-shaped PU sizes are the most representative sizes in each one of the CU depth-levels for both texture pictures and depth maps. Based on these observations, some scenarios that limit the PU sizes in the Inter prediction were investigated in this work targeting a complexity reduction to support energy-efficient hardware design (see Chapter 8).

6.2.2 3D-HTM Encoding-tool Analyses Based on FCO configuration

As previously explained, the FCO configuration adopts the encoding of the texture pictures before their associated depth maps for BVs and depth maps before texture pictures for all dependent views that composes the 3D-video sequences. Figure 6.21 shows the selection of Inter prediction (Inter-frames and Inter-view), and Intra-frame prediction encoding tools according to the PUs for both texture pictures and depth maps.

Note that the behavior of the selection is different considering the FCO configuration when compared with the CCO approach. In FCO, depth maps encode a higher number of PUs than the texture due to use of smaller PU sizes. This way, texture encodes 45.55% of the PUs whereas depth maps use 54.45% of the PUs. In order to obtain a more precise assessment about the encoding-tool behavior, the percentage of pixels encoded with the encoding tools was prioritized for the 3D-HTM encoding-tool analyses in the sequence of the text.

Figure 6.21 – Inter prediction (Inter-frames and Inter-view), and Intra-frame prediction PU selection with the FCO approach



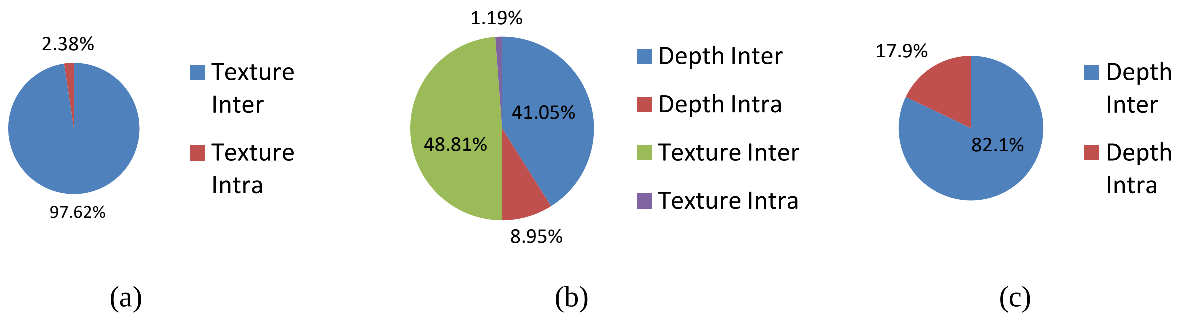
Source: Author.

The results obtained with the FCO approach were also divided according to the type of prediction, Intra-frame or Inter prediction. As presented in Figure 6.22, the representativeness of the Inter Prediction is predominant for both the texture and the depth-map pictures. Considering the texture frames (Figure 6.22-a), 97.62% of the pixels are encoded by Inter prediction while Intra-frame prediction is responsible for encoding only 2.38% of the pixels.

In the depth maps, 82.1% of the pixels are encoded with Inter prediction, and the Intra-frame prediction encodes 17.9% of them (see Figure 6.22-c).

It is important to notice that the the depth intra tools importance is higher in an FCO approach than in a CCO approach, which is expected since dependent views encode the depth maps before texture frames. This way, DIS and DMM-1 modes tend to obtain a better match than Inter prediction tools (especially than the Disparity Estimation and Merge mode) preserving the depth-map characteristics needed to efficiently synthesize the intermediate texture views. The next subsection presents an Intra-frame prediction analysis considering the FCO approach.

Figure 6.22 – Inter and Intra-frame prediction representativeness with an FCO approach: a) Texture pictures; b) Texture and depth maps; c) Depth maps



Source: Author.

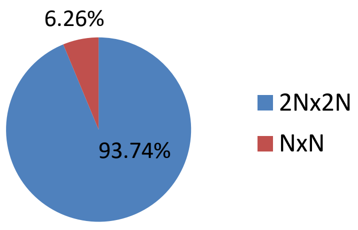
6.2.2.1 Intra-frame Prediction

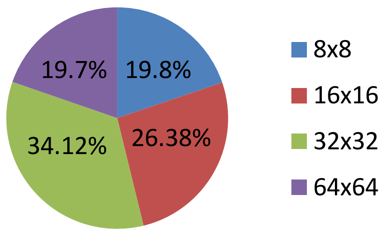
This subsection presents in details the results obtained from the experiments considering only the Intra-frame prediction and its respective modes under the FCO configuration. First, the texture encoding process with Intra-frame prediction is exploited and, in the sequence, the depth-map encoding is analyzed using the experimental results. This way, both texture, and depth-map characteristics can be exploited separately for the identification of efficient complexity-reduction strategies for the Intra-frame prediction in an FCO context.

6.2.2.1.1 Texture

According to the experiments developed in this work, 93.74% of the pixels are encoded with 2Nx2N partitions when only Intra-predicted texture pixels are considered using the FCO configuration, as presented in Figure6.23-a. The NxN partitions are responsible for encoding the others 6.26% of the pixels (used only in the fourth quad-tree level). Figure 6.23-b also shows these data according to the CU-depth level at which these partitions occurred. Considering the results, no quad-tree level presented negligible results, since the different levels encode between 19.7% and 34.12% of the Intra-predicted texture pixels. In other words, the behavior of the texture picture encoding is very similar for both CCO and FCO approaches.

Figure 6.23 – FCO Intra-frame prediction representativeness considering texture pictures: a) According to the partition type; b) According to the CU-depth level





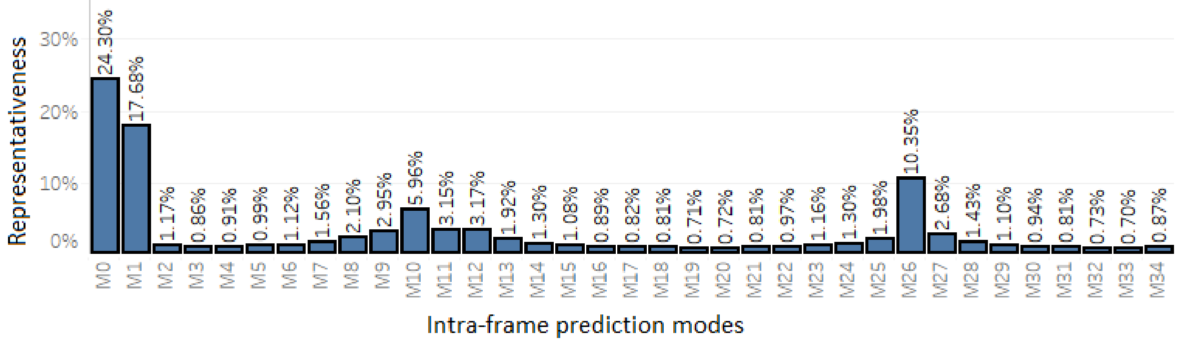
(a) (b)

Source: Author.

Other data arrangement was used to verify the number of Intra-predicted pixels that were encoded with all 35 modes supported by the texture coding (Planar, DC, and Angular). Figure 6.24 shows the percentage of pixels encoded with each one of these modes. The Planar mode (mode 0 according to the 3D-HTM) is the most important mode since it encodes 24.30% of the pixels, followed by the DC mode (mode 1), Vertical mode (mode 26), and Horizontal mode (mode 10) with 17.68%, 10.35%, and 5.96%, respectively. The other modes encode a maximum of 3.17% of the pixels. Also, the four modes that encode more pixels account for 58.29% of the pixels, which is an important result in a scenario with 35 modes. Note that the behavior of the texture picture encoding related to the mode distribution is also very similar for both CCO and FCO approaches.

The next subsection presents the Intra-frame prediction results considering depth maps.

Figure 6.24 – Intra-frame prediction representativeness according to the encoding mode for the texture pictures

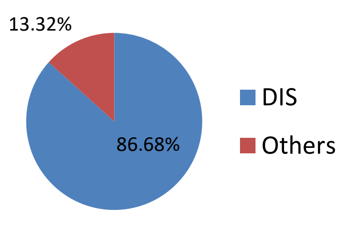


Source: Author.

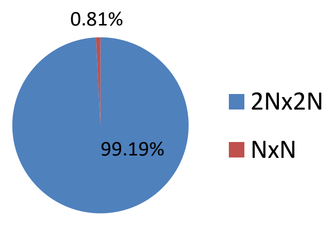
6.2.2.1.2 Depth Maps

This subsection presents and discusses the results correspondent to the Intra-predicted depth-map pixels. As previously explained, in the case of the depth-map coding with Intra-frame prediction is necessary to analyze the DIS mode firstly. Figure 6.25-a separately shows the percentage of pixels encoded with the DIS mode and the other Intra-frame tools for depth maps in an FCO approach. It is important to notice that the DIS mode is widely used in FCO approach as well as in the CCO approach, encoding 86.68% of the Intra-predicted depth-map pixels when using FCO. The 13.32% of remaining pixels are distributed to be encoded by the 37 Intra modes defined for depth maps (35 HEVC modes and two depth modeling modes).

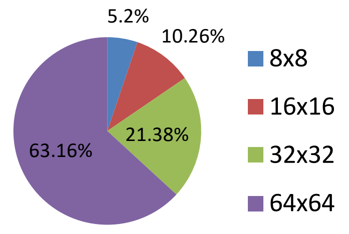
Figure 6.25 – Intra-frame prediction representativeness with an FCO approach for depth maps: a) Using the DIS mode; b) According to the partition type; c) According to the CU-depth level



(a)



(b)



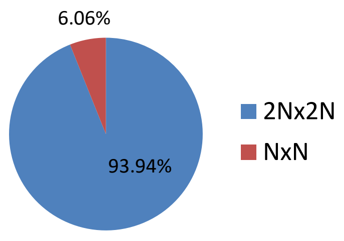
(c)

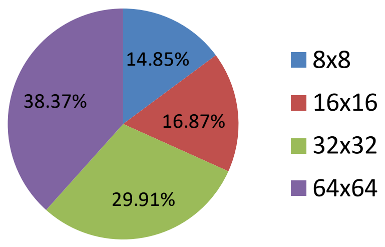
Source: Author.

Even disregarding the DIS mode by considering only the other modes, the NxN partitions present a small impact in the coding of depth maps, accounting for about 6.06% of the pixels, as presented in Figure 6.26-a. However, the NxN partition is not negligible in an FCO approach when compared with a CCO approach (increases from 1.48% to 6.06%) if the DIS encoding tool was disregard. In terms of CU-depth distribution, the DIS mode pushes up the use of 64x64 CU-depth level, as presented in Figure 6.25-c and Figure 6.26-b.

To better evaluate the Intra-encoded depth-map representativeness, Figure 6.26 and Figure 6.27 disregard the DIS mode. Figure 6.26-b shows the representativeness of the Intra-predicted depth maps according to the quad-tree level. As presented in Figure 6.26-b, the FCO depth-map encoding has a different behavior when compared with the CCO. While in the CCO approach, the Intra-predicted depth blocks encoded with bigger sizes tend to be more representative, in the FCO, the area covered by each CU-depth level is distributed more evenly.

Figure 6.26 – Intra-frame prediction representativeness with an FCO approach for depth maps (except DIS) according to the: a) Partition mode; b) CU-depth level





(a) (b)

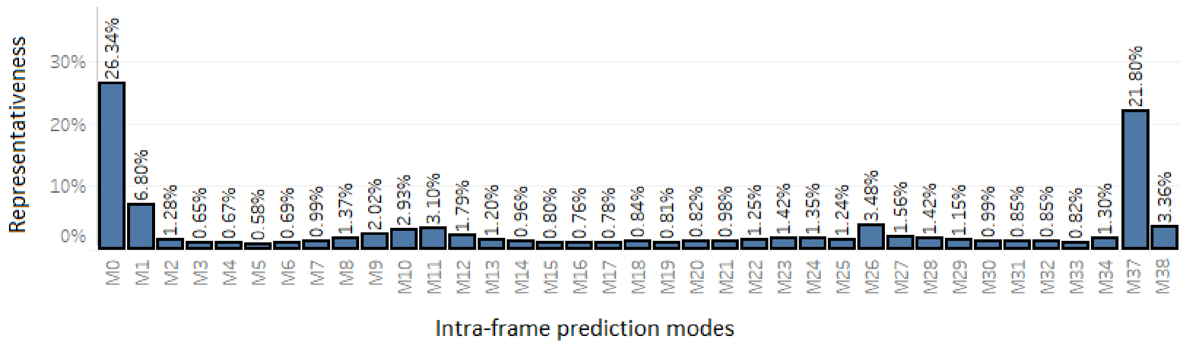
Source: Author.

The most important depth-map modes more important in the encoding process were also evaluated in this subsection. Since the DIS mode is the most representative Intra-frame depth-map mode, this mode was disregard. Figure 6.27 shows the representativeness of the Intra-frame depth-map modes where the five most representative modes are Planar (mode 0), DMM-1 (mode 37), DC (mode 1), Vertical (mode 26), and DMM-4 (mode 38), with 26.34%, 21.80%, 6.80%, 3.48%, and 3.36% of representativeness, respectively. The sixth most representative mode is the Angular 11 with 3.10% whereas the Horizontal mode (M10) is the seventh most representative mode with 2.93% of representativeness.

These seven modes together are responsible for 67.81% of the Intra-predicted depth-map pixels disregarding the DIS mode. It is important to notice that no other mode encodes more than 2.02% of the pixels as well as there are 37 modes that can be employed in the depth-maps disregarding the DIS mode (the same texture modes and two DMM modes).

By including six modes among the seven most representative modes, i.e., Planar, DC, DMM-1, DMM-4, Horizontal, and Vertical modes, the percentage of encoded pixels disregarding the DIS mode is 64.71%, and this number increases to 95.3% when considering the DIS mode. As previously stated, the DMM modes (modes 37 and 38) are associated with a large complexity so that they are not applied in the first quad-tree level (64x64 CU-depth level). The coding with DMM-4 mode covered a reduced area considering the FCO approach, only 3.36% (decreases from 17.22% in CCO to 3.36% in FCO). This behavior is expected since the DMM-4 requires information of texture collocated blocks and, in the FCO approach, this is possible only in the Base View. In dependent views, depth-maps are encoded before texture frames which disabled the use of DMM-4.

Figure 6.27 – FCO Intra-frame prediction representativeness considering depth-maps according to the encoding mode (except DIS)



Source: Author.

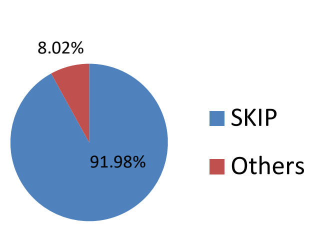
6.2.2.2 Inter-frames and Inter-view Predictions

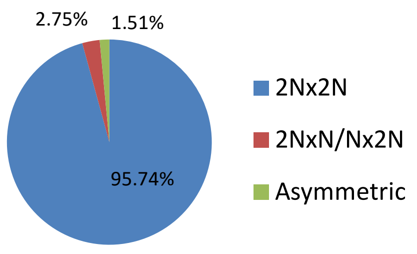
This subsection presents the results focusing on the FCO approach and in the Inter Prediction, i.e., focusing on the Inter-view and Inter-frames predictions. The results are also presented separately in two subsections: one considering texture pictures and another considering depth maps.

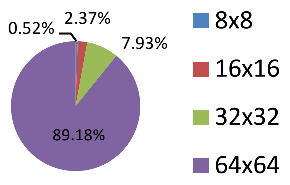
6.2.2.2.1 Texture

As previously mentioned, most of the pixels are encoded using the Inter prediction in FCO approach (about 89.86%, see Figure 6.22). This is also true for FCO Inter-predicted texture pixels, where the percentage of pixels is even higher: 97.62%. As previously explained, this expressive representativeness of the Inter prediction is mainly due to the Skip mode, as presented in Figure 6.28-a. The percentage of Inter-predicted texture pixels considering the Skip mode is 91.98%, which is applied to any quad-tree level of partitions 2Nx2N, dealing the 2Nx2N partition to reach 95.74% of the pixels, as presented in Figure 6.28-b. The representativeness of the Inter prediction according to the CU-depth level can be observed in Figure 6.28-c. When the results are evaluated considering the quad-tree level of the CUs, it is possible to observe that the first quad-tree level is responsible for encoding 89.18% of the FCO Inter-predicted texture pixels. Considering the first and the second quad-tree levels, the percentage of encoded pixels is 97.11% and, if the three first quad-tree levels are considered, this percentage grows to 99.48%. This way, the last quad-tree level (8x8 CUs) is responsible for only 0.52% of the encoded pixels. Both the 2Nx2N partition and the 64x64 CU-depth level representativeness are pushed up by the Skip mode.

Figure 6.28 – FCO Inter prediction representativeness considering texture pictures: a) Using the Skip mode; b) According to the partition type; c) According to the CU-depth level





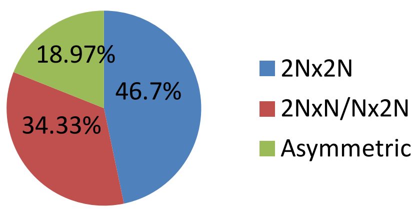


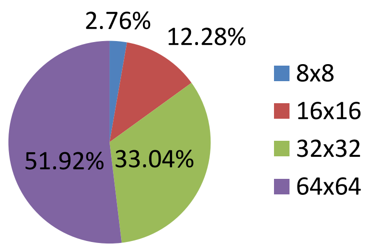
(a) (b) (c)

Source: Author.

This analysis was also done disregarding the Skip and detailing the partitions and CU-depths selected with the other modes, as presented in Figure 6.29. With this data arrangement showed in Figure 6.29-a, it is possible to verify that the DBBP, Merge, ME/DE encoding tools are considerably used in the 2NxN and Nx2N partitions. Despite these partitions being less representative than 2Nx2N partitions considering the FCO Inter prediction of texture pictures, they can present important impacts on the image quality and compression. It is important to notice that 2Nx2N, 2NxN, and Nx2N partitions are responsible for 98.49% of the total FCO Inter-predicted texture pixels (or 81.03% disregarding Skip mode). However, the importance of the smaller CU-depth levels is notable disregarding the Skip mode since the 8x8 and 16x16 CU depths are responsible for encoding 15.04% of the pixels even covering a smaller area per block, as depicted in Figure 6.29-b.

Figure 6.29 – FCO Inter prediction representativeness considering texture pictures (except Skip): a) According to the partition type; b) According to the CU-depth level



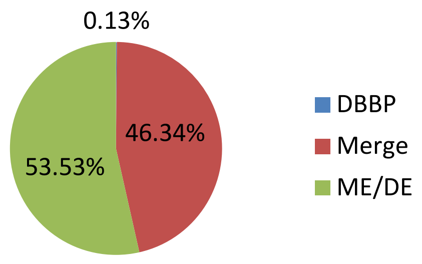


(a) (b)

Source: Author.

In Figure 6.30, the data are presented disregarding the Skip mode but detailing the other encoding tools. It is possible to notice that Merge and Disparity/Motion Estimations present a similar behavior, and also they encode a significant number of pixels when compared to the DBBP mode. Whereas the Merge and ME/DE encodes 46.34% and 53.53% of pixels, respectively, the DBBP is responsible for encoding only 0.13% of pixels. Note that the number of pixels encoded using DBBP decreases in an FCO approach when compared to the CCO approach (from 4.34% to 0.13%). The next subsection focuses on the results for the FCO Inter prediction of depth-map.

Figure 6.30 – FCO Inter prediction representativeness of texture pictures according to the encoding mode (except Skip)

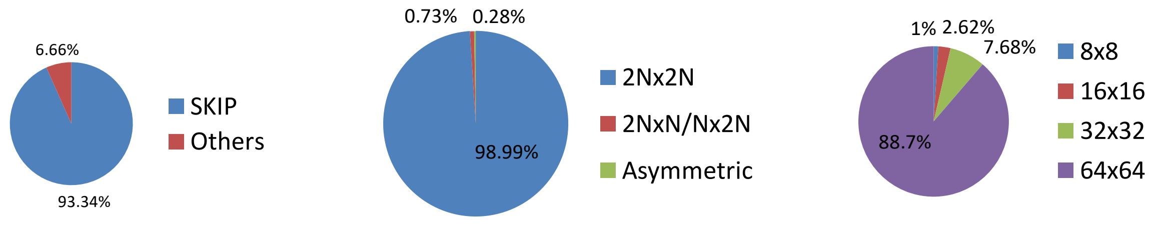


Source: Author.

6.2.2.2.2 Depth Maps

Considering the FCO approach and the depth-map encoding process, Inter prediction (Inter-view and Inter-frames) corresponds by 82.1% of the encoded pixels. This higher value is mainly due to the Skip mode. In the depth maps, 93.34% of the FCO Inter-predicted pixels used Skip mode, as presented in Figure 6.31-a. Regarding partition types, the 2Nx2N partition is responsible for encoding 98.99% of the FCO Inter-predicted depth-map pixels, as presented in Figure 6.31-b. The analysis regarding the quad-tree level at which the blocks were encoded (see Figure 6.31-c) shows that the absolute majority of pixels are encoded with the first quad-tree levels. The 64x64 and 32x32 CU depths are responsible for encoding 96.38% of the pixels considering depth maps. This predominance of areas encoded with 2Nx2N partitions and CU-depth levels with larger CU sizes tends to be pushed up by the Skip mode, as in the previous cases. This way, again, an evaluation disregarding the Skip mode is presented.

Figure 6.31 – FCO Inter prediction representativeness considering depth maps: a) Using Skip mode; b) According to the partition type; c) According to the CU-depth level

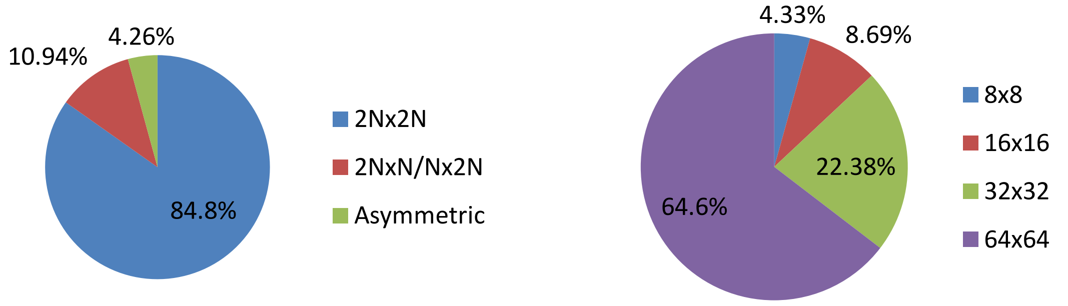


(a) (b) (c)

Source: Author.

A data arrangement that disregards the Skip mode is presented in Figure 6.32. This way, the predominance of the 2Nx2N partitions can be confirmed for the other encoding tools. Even disregarding the Skip mode, the 2Nx2N partition encodes 84.8% of the FCO Inter-predicted depth-map pixels. Similarly, even disregarding the Skip mode, the 64x64 and 32x32 CU-depth levels are responsible for encoding 86.98% of the pixels. In the sequence, a further analysis according to the encoding mode is presented.

Figure 6.32 – FCO Inter prediction representativeness considering depth maps (except Skip): a) According to the partition type; b) According to the CU-depth level

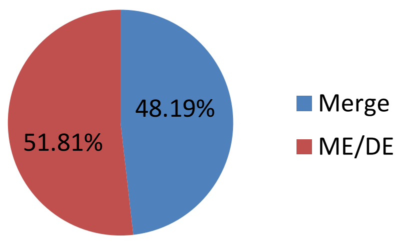


(a) (b)

Source: Author.

The results disregarding the Skip mode and separated according to the encoding tools are presented in Figure 6.33. Again, it is possible to observe that the tool behavior, for depth maps, is similar with and without the Skip mode. Whereas Merge encodes 48.19% of the pixels, ME/DE Estimations are responsible for encoding 51.81% of the pixels. Both Merge mode and ME/DE steps tend to encode more with 2Nx2N partitions and with the first CU-depth level.

Figure 6.33 – FCO Inter prediction representativeness of depth maps according to the encoding mode (except Skip)



Source: Author.

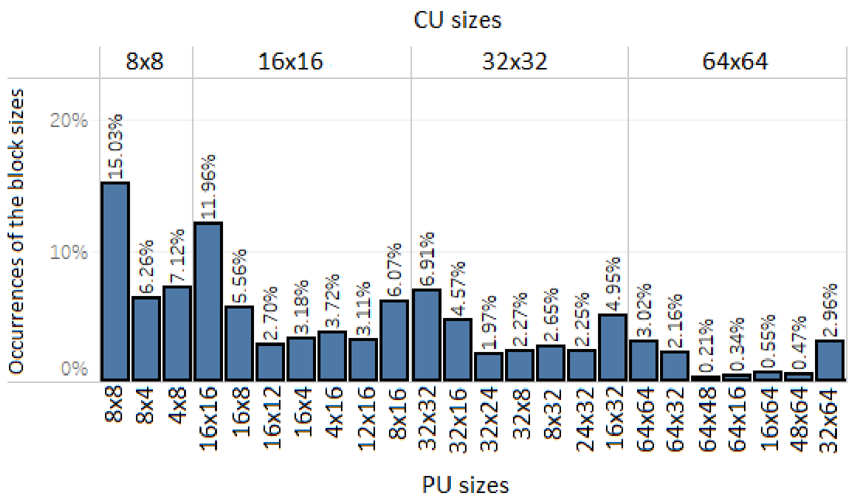
6.2.2.2.3 Block-Size Analysis

Figure 6.34 shows the average percentage of PU sizes selection in the Inter prediction considering the texture pictures and the FCO approach. The values are presented separately for all 24 PU sizes according to the CU-depth level. Considering only the data referred to the ME/DE and Merge steps (and DBBP mode for the texture), i.e., disregarding Skip mode, the 8x8 PU size is the most frequently selected block size with 15.03% of selections. The second most often selected size is the 16x16 with 11.96% of selections. Note that the other two square-shaped block sizes allowed in the Inter prediction (32x32 and 64x64 PU sizes) are less selected than the 8x8 and 16x16 sizes. Nevertheless, 32x32 is the fourth size most selected with 6.91%. The 64x64 block size is the thirteenth most selected size with 3.02% of the PUs.

Since bigger PUs are more representative in the image than the smaller PUs, the percentage of pixels that were covered by each PU size was evaluate. The data about the PU sizes selection were adjusted considering the percentage of pixels that were encoded by each PU size, considering an average of all tested conditions. This analysis, as depicted in 6.35, shows that bigger sizes (such as 64x64 and 32x32) are more representative in the video sequences, even being less frequent. Whereas 64x64 PU size is the most representative size encoding 24.64% of the pixels, 32x32 PUs are the second most representative PUs by encoding 14.06% of the pixels. Note that the square-shaped PU sizes are the most representative sizes in each one of the CU-depth levels.

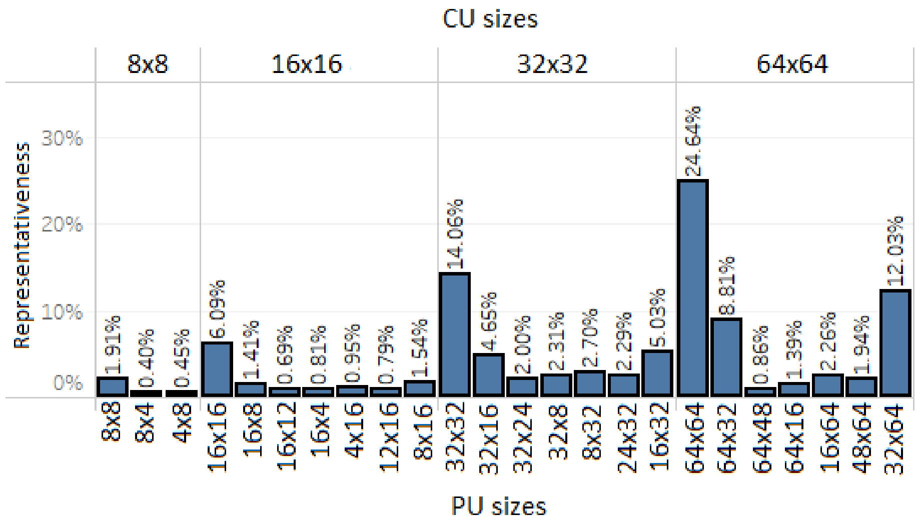
Figure 6.34 and 6.35 show that square-shaped PUs are both frequent and representative when compared to the non-square-shaped PUs in the texture picture encoding process with the FCO approach. Note that 8x8 PU size is the most frequent and the 16x16 PU size is the second most frequent, whereas the square-shaped sizes 64x64, and 32x32 are the most representative sizes.

Figure 6.34 – Selection of PU sizes for the Inter prediction considering texture pictures and FCO approach



Source: Author.

Figure 6.35 – Representativeness of PU sizes for the Inter prediction considering texture pictures and FCO approach

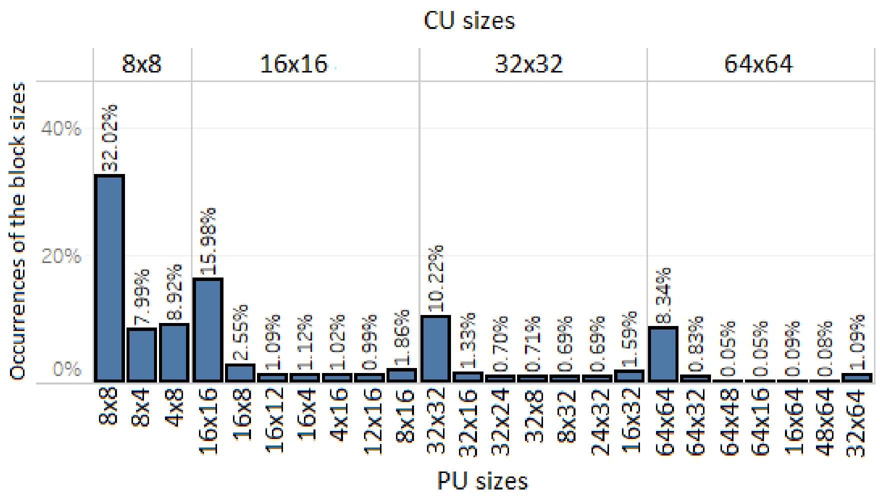


Source: Author.

Figure 6.36 shows the average percentage of PU sizes selection in the Inter prediction considering the depth maps and the FCO approach. The values related to the depth maps are presented separately for all 24 PU sizes according to the CU-depth level. Considering only the data referred to the ME/DE and Merge steps, i.e., disregarding Skip mode, the 8x8 PU size is the most frequently selected block size with 32.02% of selections, followed by other two square-shaped block sizes, 16x16, and 32x32 PU sizes. The 16x16 PU size is the second most often selected size with 15.98% of selections and the 32x32 PU size is the third most often selected PU size with 10.22% of selections. Another square-shaped block size, the 64x64, is the fifth most often selected PU size with 8.34% of selections. Note that in an FCO approach, the behavior correspondent to the block sizes is similar for texture pictures and depth maps. The four square-shaped block sizes cover 66.56% of the selections related to depth maps and the 33.44% are distributed for the other 20 PU sizes.

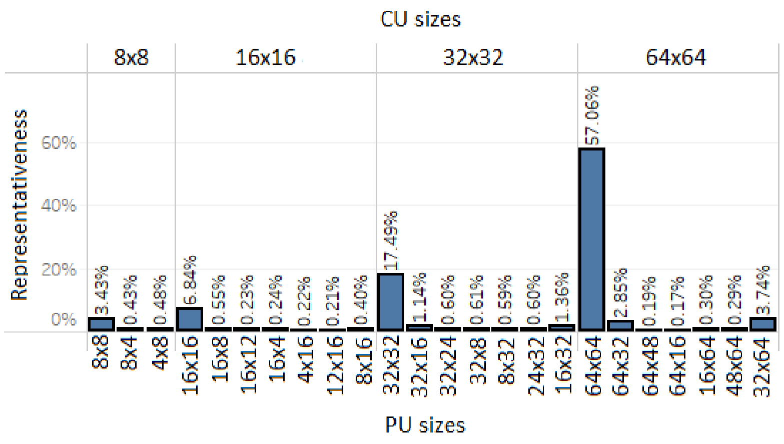
For depth maps, the data about the PU sizes selection were also adjusted considering the image representativeness, as presented in Figure 6.37 . This analysis shows that the three bigger sizes (64x64, 32x32, and 16x16) are the most representative sizes by encoding 57.06%, 17.49% and 6.84% of the pixels, respectively. The other square-shaped PU size, the 8x8, is the fifth most representative size with 3.43%. Note that the square-shaped PU sizes are the most representative sizes in each one of the CU-depth levels and, together, they cover 84.82% of the encoded pixels considering depth maps. Figure 6.36 and Figure 6.37 show that square-shaped PUs are both frequent and representative when compared to the non-square-shaped PUs in the depth-map encoding process. Note that the three bigger square-shaped sizes (64x64, 32x32, and 16x16 PU sizes) are the three most representative sizes whereas 8x8, 16x16, and 32x32 PU sizes are the three most frequent sizes. Furthermore, 64x64 PU size is the fifth most frequent whereas the 8x8 PU size is the fifth most representative size.

Figure 6.36 – Selection of PU sizes for the Inter prediction considering depth maps and FCO approach



Source: Author.

Figure 6.37 – Representativeness of PU sizes for the Inter prediction considering depth maps and FCO approach



Source: Author.

Considering the presented results, one can conclude that the square-shaped PU sizes have the two most selected sizes (16x16, and 8x8) and they have the two most representative sizes (64x64, 32x32) considering both the texture and the depth maps. Considering depth maps, one can conclude that the square-shaped PU sizes are the three most selected sizes (32x32, 16x16, and 8x8) and they have the three most representative sizes (64x64, 32x32, and 16x16). Finally, the four square-shaped PU sizes are the most representative sizes in each one of the CU-depth levels for both texture pictures and depth maps.

6.3 General Discussion about the Experimental Results

Comparing the results obtained for both CCO and FCO approaches, it is possible to observe some differences between these two encoding order configurations, mainly in the results related to the depth-map encoding.

One of the main differences between CCO and FCO approaches is related to the usage of depth intra tools to encode the depth-map PUs, where only 6.61% of the PUs are encoded using depth intra tools in a CCO approach (considering both the texture pictures and the depth maps) and 23.31% of the PUs are encoded with depth intra tools in an FCO approach. This way, the percentage of pixels covered by depth intra tools increases from 3.62% to 8.95% when the CCO is changed to the FCO approach. Part of this variation is due to the increasing in the number of pixels encoded using DIS mode in depth maps, which increases from 81.3% in the CCO approach to 86.68% in the FCO approach. The modification in the depth encoding process behavior is expected since the FCO dependent views encode depth maps before the texture pictures and, therefore, the encoder tends to select Intra-predicted candidates rather than using Inter-predicted candidates obtained with Merge mode or from disparity vectors.

Another important observation is related to inexpressive use of NxN partitions in the depth-map coding in CCO and FCO where less than 1% of the depth-map pixels use this type of partition. However, the results also show that the NxN partition is more important in an FCO approach. Considering only the depth-map pixels and disregarding the DIS mode, the NxN partition increases the encoded pixels from 1.48% in a CCO approach to 6.06% in an FCO approach.

The distribution of pixels according to the CU-depth level in the depth-map encoding is also different in the FCO approach when compared with the CCO approach. Whereas CCO encodes more pixels in bigger CU sizes, FCO approach encodes the depth maps in a more balanced way in terms of CU-depth levels, ranging from 14.85% to 38.37% according to the CU-depth level. In CCO configuration this range is from 5.73% to 44.35%. Similarly, the usage of Intra modes in the depth-map encoding is different in an FCO approach. Whereas both DMM-1 and DMM-4 modes are largely used in a CCO approach, in an FCO approach, DMM-1 mode tends to be more representative than the DMM-4 mode. In a CCO approach, the DMM-4 and the DMM-1 are the third and the fourth most selected modes among the Intra modes for depth maps, encoding 17.22% and 13.95% of the pixels, respectively. Considering an FCO approach, the DMM-1 is the second most selected mode, and the DMM-4 is only the fifth most selected mode among the Intra tools for depth maps, encoding 21.80% and 3.36% of the pixels, respectively. This different behavior of the DMM-4 encoding tool is also expected since DMM-4 requires the use of collocated texture samples which makes impossible its use considering the dependent views in an FCO approach.

Focusing on the Inter prediction of texture pictures, the behavior in terms of block sizes selection and representativeness is practically the same in both FCO and CCO approaches evidencing the importance of the square-shaped block sizes where the 8x8 and the 16x16 block sizes are the most often selected sizes and the 32x32 and the 64x64 block sizes are the most representative sizes for both FCO and CCO approaches disregarding the Skip mode. Focusing on the Inter prediction of depth maps, the behavior in terms of block sizes selection and representativeness is a little bit different for FCO and CCO approaches. The importance of the smaller block sizes is higher in the FCO approach so that the two most selected block sizes are the 8x8 and the 16x16, with 32.02% and 15.98%, respectively. The depth-map encoding with Inter prediction showed the importance of the square-shaped block sizes (64x64, 32x32, 16x16, and 8x8) in the coding process so that at least three square-shaped block sizes are the most selected and the most representative sizes for both coding order configurations disregarding the Skip mode. Notice that there are 24 PU sizes in the Inter prediction.

The ME importance is evident for both CCO and FCO approaches, where more than 80% of the Inter-predicted PUs in CCO and more than 90% of the Inter-predicted PUs in FCO use this type of prediction (disregarding Skip mode). DE is also important, being part of at least 20.62% of the pixels encoded considering the depth maps in the CCO approach and at least 10.7% of the pixels encoded considering the depth maps in the FCO approach. Notice that there is a significant decrease in the use of the DE considering the FCO. Such behavior can also be observed in the texture encoding, and it is related to the coding order of the dependent views, that tends to choose intra modes rather using Merge mode and DE. As previously explained, some Merge candidates are provided by techniques that depend on the channel coding order. This way, the configuration is directly related to the choice of the modes. Note that DE stills to be important to the encoding of regions with occlusion and disocclusion.

Focusing on the texture behavior, the use of DBBP mode in an FCO approach also varies a lot. Whereas in a CCO approach, DBBP encodes 4.34% of the texture pixels considering only the Inter prediction and disregarding the Skip mode, in an FCO approach, DBBP is practically not used, encoding only 0.13% of the pixels considering the same scenario. Considering the fractional ME and DE, the significance of the quarter-pixel vectors is evident in texture encoding process since 56.54% of the pixels are encoded using quarter-pixel FME and 16.54% of the pixels are encoded with quarter-pixel DE considering the CCO approach and disregarding the Skip mode. Considering the FCO approach, the importance of the quarter-pixel prediction is also notable. In the texture encoding process, 63.45% of the pixels are encoded using quarter-pixel ME, and 8.89% of the pixels are encoded with quarter-pixel DE, disregarding the Skip mode. It is important to highlight that depth maps do not use fractional-precision vectors.

7 INTRA-FRAME PREDICTION DEVELOPED ARCHITECTURES

This chapter presents in details the two high-throughput architectures developed during this doctorate for the Intra-frame prediction. The first architecture consists of a Depth Intra Skip (DIS) hardware design that adopts a distortion metric replacing, and data reuse in order to obtain energy efficiency. The second hardware design consists of a system capable of dealing with both the novel depth-map Intra-frame prediction modes, and the modes inherited by the HEVC standard, while reduces energy and memory requirements. Both architectures used pipeline and the clock-gating low-power technique in order to reach processing rates capable of processing high and ultra-high definition videos and, also, to save energy consumption when it is possible. It is important to mention that the complexity-reduction heuristics proposed for the Intra-frame prediction in this chapter are anchored in the statistical results for the CCO approach in Chapter 6 and, therefore, the impacts on the compression of them also consider the CCO approach. However, all the algorithms and architectures can be used along an FCO approach without drastic modifications.

7.1 Low-Power Depth Intra Skip Architecture based on Distortion Metric Replacing

As previously discussed, no work proposing a dedicated hardware design for the 3D-HEVC DIS coding tool was found in the literature during the development of this doctorate. Aiming at addressing this issue, the first dedicated hardware design for the DIS tool was developed considering all possible block sizes. A complexity-reduction strategy to obtain a low-power and high-throughput architecture was implemented to process the similarity criterion. Also, strategies of data reuse were used in the hardware design. Such strategies were adopted based on evaluations performed with the 3D-HTM reference software, as follows.

7.1.1 3D-HEVC DIS Complexity-Reduction Strategy and Evaluation

As presented in Section 4.5.1.1, SVDC requires a considerable number of arithmetic operations to decide the best mode to encode a given block. Furthermore, SVDC also imposes multiplications and costly rendering processes for hardware implementations. This way, replacing the SVDC by other similarity criterion that allows a hardware-friendly design is a promising complexity-reduction strategy. The use of SAD (see Section 2.5) as the similarity criterion directly applied to the depth-map samples avoids both multiplications and the rendering process.

Comparing (3) and (21) that mathematically represent the SAD and the SVDC, respectively, it is possible to notice that the total arithmetic operations can be reduced at least

71.52% (depending on the block size) using SAD as similarity criterion. For instance, considering a 64x64 block, the total arithmetic operations can be reduced of 28,544 to 8,128. Table 7.1 presents in details the total arithmetic operations as a function of the block size for the SAD and SVDC distortion metrics, given an NxN block, where N represents the block height/width.

Table 7.1 –Total arithmetic operations for an NxN block according to the distortion metric

Arithmetic Operations	**Distortion Metric**		**Δ (SVDC – SAD)**
	SAD	**SVDC**	
Subtractions	N^2	$3N^2$	$2N^2$
Sums	N^2-N	$2N^2$-2N	N^2-N
Multiplications	–	$2N^2$	$2N^2$
Total	$2N^2$-N	$7N^2$-2N	$5N^2$-N

Source: Author.

Despite replacing SAD as the similarity criterion rather than using SVDC being a promising complexity-reduction strategy, the trade-off between coding efficiency and computational effort must be carefully considered. This way, the coding-efficiency impact of this strategy was evaluated with the 3D-HTM reference software in the version 16.0 (3D-HEVC Reference Software, 2019), the IO temporal configuration and the test conditions recommended in the CTC document by the JCT-3V and presented in Appendix A. Therefore, the eight 3D video sequences were used, including five 1920x1088 sequences and three 1024x768 sequences as well as the four QP sets for texture/depth defined in the CTC document. Furthermore, it is worth saying that between 200 and 300 AUs were encoded according to these video sequences.

Table 7.2 presents the coding-efficiency degradation results for this analysis regarding BD-rate. Throughout this thesis, the coding efficiency results are mainly discussed for video total and synthesized frames. When *Video Total* is referred, the quality related to texture channel (disregarding synthesized frames) considering the total bit rate of the video (texture + depth) is measured. When *Synthesized Frames* are referred, the quality of frames synthesized with encoded texture and encoded depth maps, using DIBR, is measured in comparison with frames synthesized with original texture and original depth maps. Occasionally, quality results are also presented for *Video Only*, which represents the quality to the texture channel using the texture bit rate of the video (disregarding synthesized frames). According to the results presented in Table 7.2, using the SAD as the similarity criterion brings negligible losses at the 3D-HEVC coding efficiency, increasing the BD-rate only by 0.21% for synthesized views, on average. When the video total is considered, the result is similar, with a 0.19% BD-Rate increase. It should be noticed that these compression losses are acceptable when a significant computational effort is saved.

Table 7.2 –BD-rate increase using SAD as similarity criterion

Test Sequences	**BD-Rate increase using SAD rather than SVDC (%)**	
	Video Total	**Synthesized Only**
Balloons	0.16	0.23
Kendo	0.22	0.34
Newspaper_CC	0.22	0.26
GT_Fly	0.14	0.20
Poznan_Hall2	0.40	0.29
Poznan_Street	0.10	0.14
Undo_Dancer	0.07	0.02
Shark	0.17	0.21
1024x768	0.20	0.28
1920x1088	0.18	0.17
Average	0.19	0.21

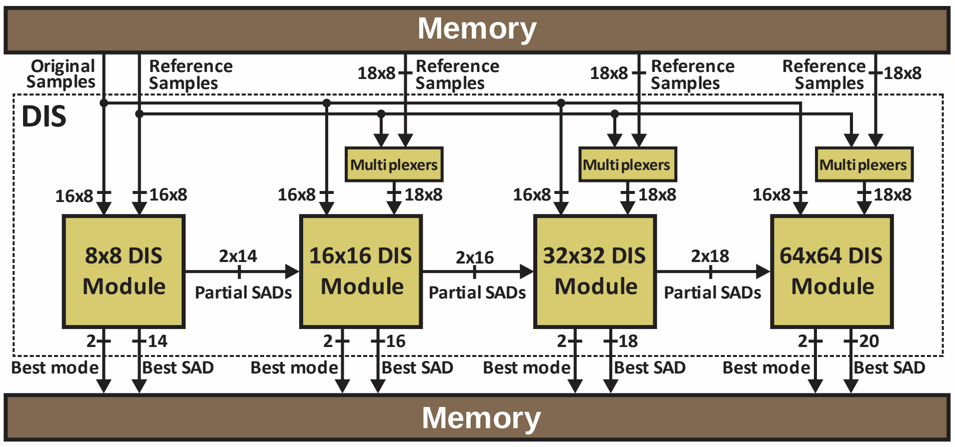
Source: Author.

Note that the complexity-reduction strategy presented in this section is capable of decreasing the computational effort of the DIS coding tool while providing a negligible coding loss in terms of compression efficiency. Since the SAD similarity criterion is friendlier than the SVDC to hardware design, a dedicated 3D-HEVC DIS architecture, which supports the four CU sizes, is presented in the next subsection.

7.1.2 Developed 3D-HEVC DIS Architecture

The developed architecture was designed to process the four block sizes supported by the DIS coding tool in parallel. For that, four DIS modules process and deliver the best SADs and DIS modes, one for each CU size, as presented in Figure 7.1. In order to reach a high parallelism level, the proposed design was made in a bottom-up approach, where the DIS modules that process CUs bigger than 8x8 pixels compose their CUs by the processing of 8x8 blocks.

Figure 7.1 – Complete diagram for the developed DIS hardware design

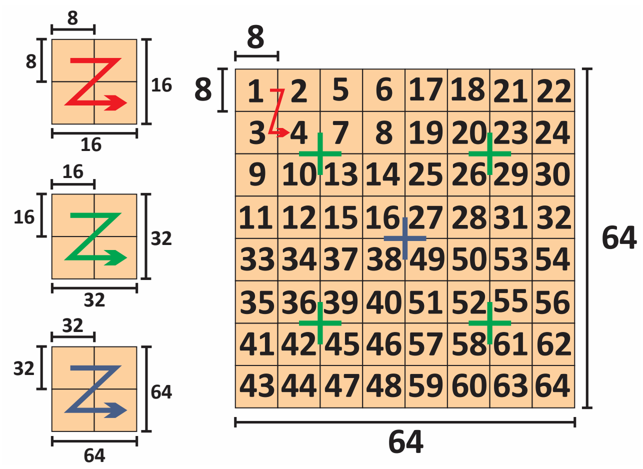


Source: (AFONSO *et al.*, 2017).

Note that the bottom-up processing strategy allows a similar hardware design for all DIS modules. The main difference among these modules is in the internal control module and the internal architecture bit depth. Once this strategy was adopted, it also provides a reduction of external-memory communication and computing by reusing input samples and partial SAD calculations. Therefore, this bottom-up processing strategy results in an efficient hardware design by allowing better control of the trade-off among hardware cost, power dissipation, and throughput.

Figure 7.2 shows the processing order (from 1 to 64) of 8x8 blocks aiming at composing bigger CU sizes. Note that this processing order follows a Z format, as highlighted in red (blocks 1, 2, 3, and 4), where a 16x16 block is composed by processing 8x8 blocks. Furthermore, the strategy of using a Z-format is extended to other CU sizes, i.e., 32x32 blocks can be composed by 16x16 blocks (green), and 64x64 by 32x32 blocks (blue). The Z-format scheme was adopted to make the DIS hardware design compliant with other Intra-frame coding tools, since these modes have data dependence due to the use of left neighboring column and above neighboring row as reference samples. It is worth saying the adopted approach makes feasible the global decision on which Intra tool presents better result based on RDO.

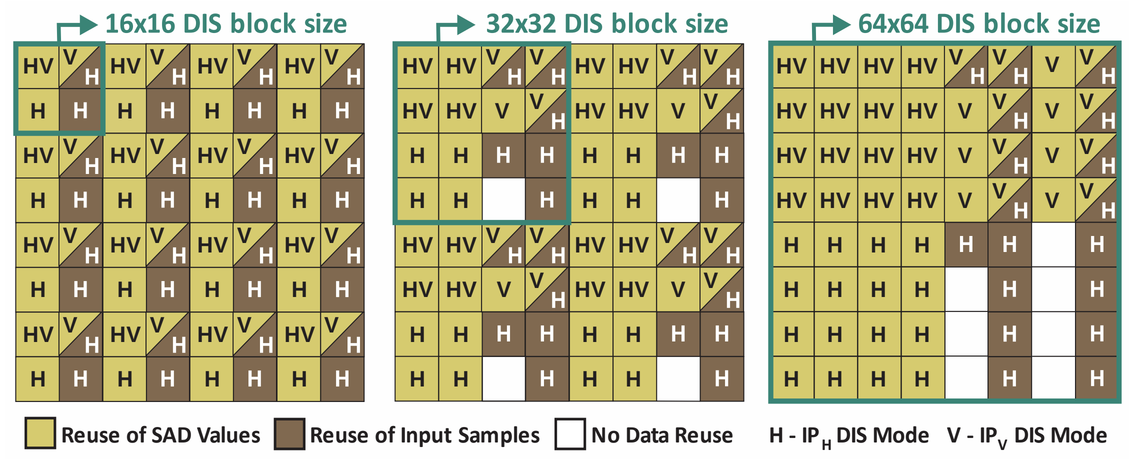
Figure 7.2 – Processing order of 8x8 blocks to compose the bigger CUs



Source: (AFONSO *et al.*, 2017).

The data-reuse scheme of bigger DIS block sizes by processing 8x8 blocks is detailed in Figure 7.3. Reuse of SAD values is possible for the IP_H and IP_V modes, while reuse of input samples is possible for the IP_H mode. In Figure 7.3, yellow squares represent the reuse of SAD values, brown squares represent the reuse of input samples, and white squares show the blocks where no data reuse is possible. Note that the reuse of SAD values reduces about 50% the arithmetic operations needed to process 16x16, 32x32, and 64x64 DIS block sizes when the IP_H and IP_V modes are considered. Also, the reuse of input samples reduces 28.91% the memory accesses considering the IP_H mode when applied to the 16x16, 32x32, and 64x64 DIS block sizes.

Figure 7.3 – Data-reuse scheme for16x16 DIS blocks (left), 32x32 DIS blocks (center), and 64x64 DIS blocks (right)



Source: (AFONSO *et al.*, 2017).

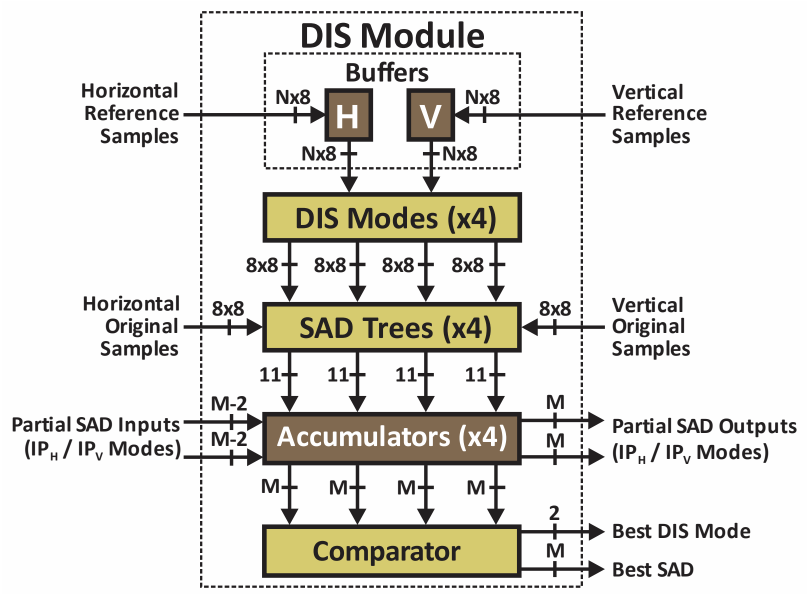
It is also important to notice that the Figure 7.3 depicts the data-reuse scheme based on 8x8 blocks, but the reuse of SAD values can be provided by any DIS Module that processes smaller CUs. However, using the SAD values from the immediately smaller module, unnecessary computing is saved. This way, the reuse of input samples is based on 8x8 blocks while the reuse of SAD values is based on the immediately smaller module.

In Figure 7.4, it is shown a generic block diagram that represents any DIS Module. In general, the differences among all four DIS Modules (one for each CU size) are in the control of each module, the I/O bit depth of partial SADs, the internal bit depth from the Accumulators, and the Comparator. For example, the 8x8 DIS Module does not apply data reuse; therefore, this module has no partial SAD inputs. Anyhow, it provides partial SADs and reference samples for the DIS Modules that process bigger CUs. Additionally, each DIS Module has two Buffers, one to store horizontal reference samples and another to store vertical reference samples. Thus, these reference samples are selected by the DIS Modes stage using multiplexers.

DIS Modes stage delivers the outputs of the four prediction modes in parallel, row after row or column after column (both with eight samples of 8 bits) according to the mode. In the following, four SAD Trees compute – in parallel – the difference between the samples predicted by the four DIS Modes and the original samples for one row or column (according to the DIS prediction mode). Then, the Accumulators accumulate the obtained SAD of each row/column until computing the SAD of the entire block. Inside of the Accumulators stage, there are multiplexers which allow the reuse of SAD from smaller DIS Modules as well as partial SAD outputs that provide the SAD results for other greater DIS Modules. Given the data-reuse scheme, the clock-gating technique is applied to the DIS Modes, SAD Trees and Accumulators stages of the 16x16, 32x32, and 64x64 DIS Modules.

Finally, the Comparator compares the obtained SAD from the four DIS prediction modes, delivering the best SAD and the best mode for each processed CU. The complete encoding of an 8x8 block requires 13 clock cycles. In Figure 7.4, **M** can assume values as 14, 16, 18, and 20 (bit depths), depending to the DIS Module whereas **N** can assume two values: eight or nine (samples) depending to the DIS Module as well.

Figure 7.4 – Generic diagram of the DIS Modules


DIS Module
Buffers
Horizontal Reference Samples
Vertical Reference Samples
Nx8
H
V
DIS Modes (x4)
8x8
Horizontal Original Samples
Vertical Original Samples
SAD Trees (x4)
11
M-2
M
Partial SAD Inputs (IP_H / IP_V Modes)
Partial SAD Outputs (IP_H / IP_V Modes)
Accumulators (x4)
2
Best DIS Mode
Best SAD
Comparator

Source: (AFONSO *et al.*, 2017).

7.1.3 Synthesis Results and Comparisons

The synthesis results were generated targeting an ASIC technology according to the experimental setup presented in Appendix B. These results are detailed in Table 7.3. It should be noticed that the hardware design presented in this thesis is the first dedicated solution for the 3D-HEVC DIS coding tool. Since direct comparisons with other works are not possible due to this fact, synthesis results of two different versions of the developed architecture were generated and compared. These two versions are called here as Non-optimized and Optimized. Both the Non-optimized and Optimized versions use the same hardware structure based on SAD as the similarity criterion presented in Section 7.1. The difference in terms of implementation is that the Non-optimized version does not implement the data-reuse scheme (SAD values and input samples) proposed in this thesis neither low-power techniques. This way, the Non-optimized version is used as a baseline to better evaluate the data-reuse and low-power strategies.

The maximum frequency obtained by the two different versions is superior to 550MHz. This operation frequency is enough to process 60 UHD 2160p (3840x2160 pixels) fps considering five views. Since the two versions present the same parallelism, both versions reach real-time processing with the same operation frequency for HD and UHD resolutions. Another common point related to both versions of the architecture is the BD-Rate increase of about 0.21% in the synthesized views, on average. Note that this BD-Rate increase is acceptable since a high amount of computations was saved (at least 71.52%).

The Optimized version adopts the reuse of SAD values and input samples as presented in Figure 7.3. Also, it implements the clock-gating register technique. As Table 7.3 shows, the Optimized version reduces the power dissipation in 20.58% considering 2160p@60fps resolution (five views) when compared to the Non-optimized version, reaching a power dissipation of 19.57mW. Considering the processing of 1080p@120fps resolution (five views), the power-dissipation reduction achieves 23.05%. The hardware resource usage of the two versions is similar. However, it is important to highlight that the data-reuse scheme and the clock-gating technique increase by 1.09% the hardware resource usage. It is important to notice that this power-dissipation reduction is due to the data-reuse scheme and low-power techniques. The power-dissipation reduction due to the use of SAD as the similarity criterion rather than using SVDC was not measured in this doctorate. By using SAD as the similarity criterion a costly rendering process related to the SVDC is also avoid.

Table 7.3 – Synthesis results for ASIC technology

Architecture		**Non-Optimized**	**Optimized**
Total Area (gates)		34.80k	**35.37k**
1080p@120fps (five views)	**Freq. (MHz)**	252.72	**252.72**
	Power (mW)	14.95	**11.52**
2160p@60fps (five views)	**Freq. (MHz)**	505.44	**505.44**
	Power (mW)	24.64	**19.57**

Source: Author.

7.2 Low-Power and Memory-Aware Depth-map Intra-frame Prediction System based on Complexity Reduction Heuristics

The development of hardware designs for the novel Intra-frame prediction modes have some gaps that remain uncovered among the works published in the literature, as follows: (i) there are no works that implement both the novel 3D-HEVC Intra-frame prediction modes and the modes inherited from HEVC; (ii) some works are not fully compliant with the 3D-HEVC standard; (iii) DMM-1 hardware designs usually store all possible wedgelets with a significant cost in the memory-related energy (SANCHEZ *et al.*, 2016). In this doctorate, an intensive effort is done in order to overcome these gaps through the development of a low-power and memory-aware depth-map Intra-frame prediction system, as follows. The next subsection presents complexity-reduction strategies and their evaluation and, after, another subsection presents the hardware design.

7.2.1 3D-HEVC Depth-Map Intra Prediction Heuristics and Evaluation

Section 6.2.1 presented a comprehensive statistical analysis of the 3D-HEVC depth-map intra-prediction tools usage. Among the data provided by the experiments, the most prominent modes and block sizes used in the depth-map coding were identified.

The statistical analysis of the intra-prediction tools usage was based on the percentage of pixels that are encoded using each encoding tool, mode and block size. Considering the CCO approach, the 3D-HTM depth-map coding used the DIS tool to encode 81.13% of the depth-map pixels, i.e., all the remaining 37 prediction modes are responsible for 18.87% of the pixels. Then, the high importance of the DIS tool becomes evident. But this fact cannot lead us to consider that the other encoding tools are less important than the DIS, since the other tools better deal with sharp edges and directional structures which are extremely important to the view synthesis process.

Since the DIS tool is much more used than the others, the Intra-frame prediction distribution was also analyzed disregarding the DIS tool in Section 6.2.1. Considering the remaining 37 prediction modes, the Planar mode is the one with higher representativeness, encoding 29.33% of the depth-map pixels, followed by the DMM-4 mode with 17.22%, the DMM-1 mode with 13.95%, and the DC mode with 6.53%. The fifth mode with higher representativeness is the Vertical mode, encoding 3.81% of the depth-map pixels, followed by the Horizontal mode with 1.81%. If these six modes are considered together, they are responsible for encoding 72.65% of the depth-map pixels when DIS is not considered.

Regarding the block sizes used for the depth maps in the Intra-frame prediction, the 4x4 block size is unrepresentative, corresponding to only 0.28% of the pixels in the depth-map coding. The results show that the bigger block sizes encode a higher amount of depth-map pixels when compared to the smaller sizes. Disregarding the DIS tool, the 4x4 block size encodes only 1.48% in this case whereas 32x32 and 64x64 block sizes have representativeness in the encoding of 36.09% and 44.35% of the depth-map pixels, respectively. The 16x16 block size encodes other 13.83% whereas the 8x8 block size encodes 5.73% of the depth-map pixels disregarding the DIS tool.

Based on the analyses presented in Section 6.2.1, some hardware-oriented heuristics are proposed in this doctorate in order to reduce the computational effort for the depth-maps Intra-frame prediction. These strategies consist in to remove the less important prediction modes and block sizes and in a specific strategy applied to the DMM-1 mode.

The depth-map Intra-frame prediction was restricted to the seven most representative prediction modes and the four most representative block sizes considering the CCO approach, which corresponds to 94.84% of all encoded depth-map pixels considering the regular encoding process. Therefore, DIS, Planar, DC, Horizontal, Vertical, DMM-1, and DMM-4 modes and 8x8, 16x16, 32x32, and 64x64 block sizes were selected to be supported in the hardware design.

Furthermore, other simplifications were introduced to the DMM-1 mode in order to reduce its processing and memory requirements, since this is the most computational intensive mode among the supported ones. The used heuristic reduces the number of wedgelets tested during the encoding process to six. The six wedgelets are defined using a pre-processing through a gradient calculation using the samples along the four neighboring-block borders. Then, all DMM-1 wedgelets are available, but only six will be evaluated. The heuristic also avoids memory usage for wedgelet patterns storage using the Bresenham (BRESENHAM, 1965) algorithm to compute at run-time the DMM-1 bitmaps for the six evaluated wedgelets. The Bresenham is widely used in Computer Graphics to perform the rasterization of lines and polygons. A modified Bresenham implementation receives the start and end points of the wedgelet and processes a pixel of the line representing the wedgelet at the time. Then, the 1,908 bitmap wedgelet patterns that must be stored when using the conventional approach were reduced to only six evaluations without any storage.

In order to verify the impact of the developed heuristics on the coding efficiency, these hardware oriented heuristics were inserted in the 3D-HTM in version 15.1 (3D-HEVC Reference Software, 2019) and compared with the original results using the same test conditions presented in Appendix A. Therefore, the experiments considered the 3D-HEVC CTC document, i.e., the eight videos at 1920x1088 and 1024x768 resolutions and four QP sets. For this evaluation, two parameters are considered, as follows: (i) the encoding time, used to estimate the computational effort reduction; and (ii) the BD-rate parameter, to evaluate the bit-rate variation for the same image quality.

Considering the average of all videos and QPs running under RA temporal configuration, the BD-Rate increased 2.64% in the synthetic views, on average, as presented in Table 7.4. On the other hand, the computational effort was reduced in 32.82% when encoding depth maps and in 16.17% for the global 3D-HEVC encoder. Regarding the IO temporal configuration, i.e., only intra-prediction tools are used in the encoding process, the simplifications increase the BD-Rate by 7.16% in the synthetic views, on average. In this case, the encoding-time of depth maps is reduced by 53.98% and the total encoding time is reduced by 46.58%, as shown in Table 7.4.

Table 7.4 – BD-Rate variation by using the hardware-oriented heuristics

Sequence	IO Temporal Configuration				RA Temporal Configuration			
	BD-Rate Variation		Encoding-Time Reduction		BD-Rate Variation		Encoding-Time Reduction	
	Video Total	Synthesized	Total	Depth Only	Video Total	Synthesized	Total	Depth Only
Balloons	-0.02%	7.96%	45.86%	53.26%	0.22%	2.73%	14.95%	32.21%
Kendo	0.02%	8.04%	46.29%	53.81%	0.61%	2.31%	14.88%	31.13%
Newspaper_CC	-0.89%	12.09%	48.35%	54.46%	-0.32%	6.48%	18.01%	35.03%
GT_Fly	-0.61%	3.82%	46.68%	54.01%	-0.05%	0.88%	15.48%	31.92%
Poznan_Hall2	-0.29%	7.98%	46.39%	54.69%	-0.15%	2.32%	17.01%	33.94%
Poznan_Street	-0.39%	3.31%	47.27%	53.88%	0.00%	2.13%	18.57%	35.61%
Undo_Dancer	-0.18%	7.55%	45.20%	53.45%	0.53%	2.28%	14.56%	31.35%
Shark	-0.87%	6.58%	46.64%	54.30%	0.23%	1.98%	15.89%	31.34%
1024x768	-0.30%	9.36%	46.83%	53.84%	0.17%	3.84%	15.94%	32.79%
1920x1088	-0.47%	5.85%	46.44%	54.07%	0.11%	1.92%	16.30%	32.83%
Average	-0.41%	7.16%	46.58%	53.98%	0.14%	2.64%	16.17%	32.82%

Source: Author.

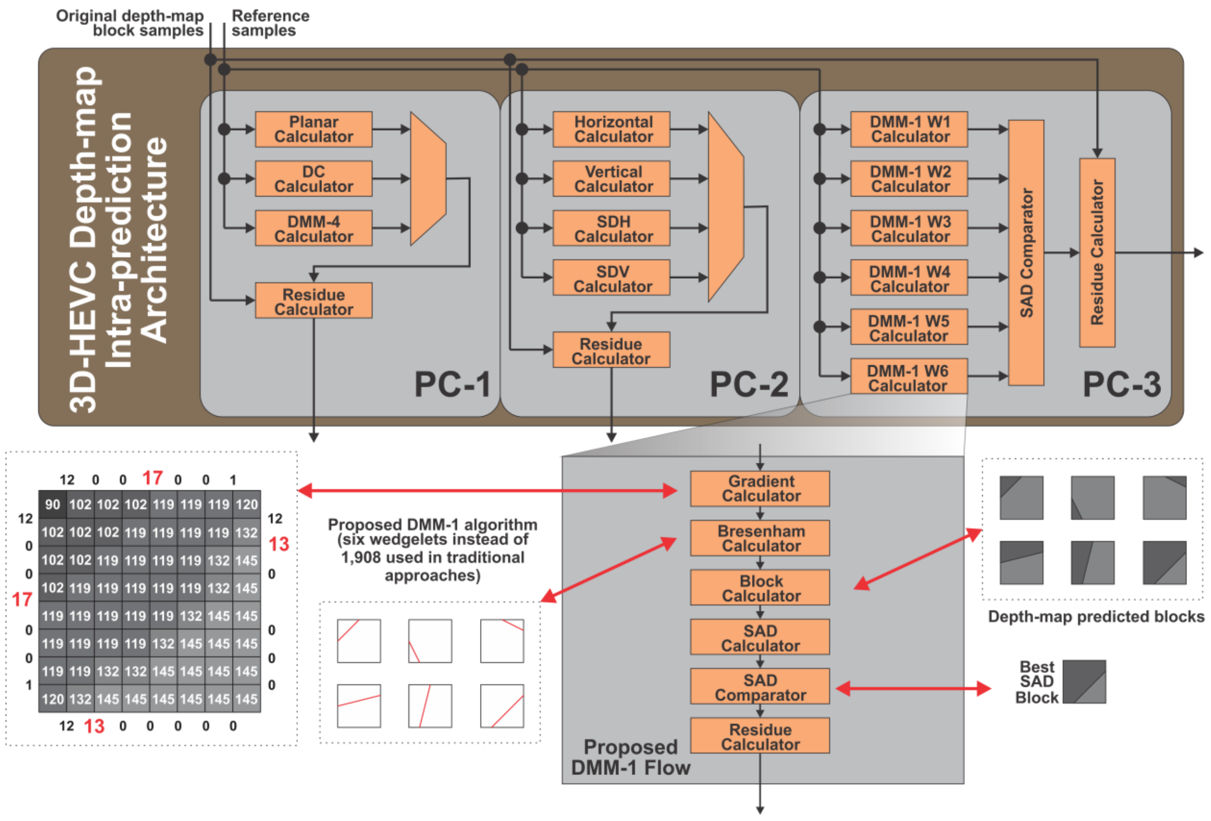
7.2.2 Developed 3D-HEVC Depth-Map Intra Prediction Architecture

This subsection presents the architecture designed for the depth-map Intra-frame prediction using the heuristics presented in the last subsection. The architecture was designed to efficiently process the modes Planar, DC, Horizontal, Vertical, DIS, DMM-1, and DMM-4, supporting 8x8, 16x16, 32x32, and 64x64 block sizes. The heuristic to simplify the DMM-1 mode was also considered.

The designed architecture is presented in Figure 7.5 and it is divided into Processing Cores (PC), where each PC is responsible for processing a subset of the pre-defined intra-frame prediction modes in an interleaved way, as presented in Figure 7.5. Then it is possible to reach the desired throughput, avoiding unnecessary calculations.

The seven supported modes are distributed in three processing units that work in parallel, as presented in Figure 7.5. The intra tools are grouped according to their similarities, which allows a reduction in area usage (and power dissipation) by reusing the hardware inside each PC. In the high-level diagram of the architecture presented in Figure 7.5, one can see the mode distribution on the three PCs: (i) Planar, DC and DMM-4; (ii) IP_H, IP_V, SD_H, SD_V; (iii) DMM-1. The IP_H and IP_V modes are used both as HEVC conventional modes as DIS sub-modes, allowing the reuse of PC-2 results. Although the DMM-1 and DMM-4 modes have similarities, these modes were not grouped in the same PC in order to obtain the desired throughput, since DMM-1 is much more complex than the other modes.

Figure 7.5 –High-level block diagram of the developed architecture

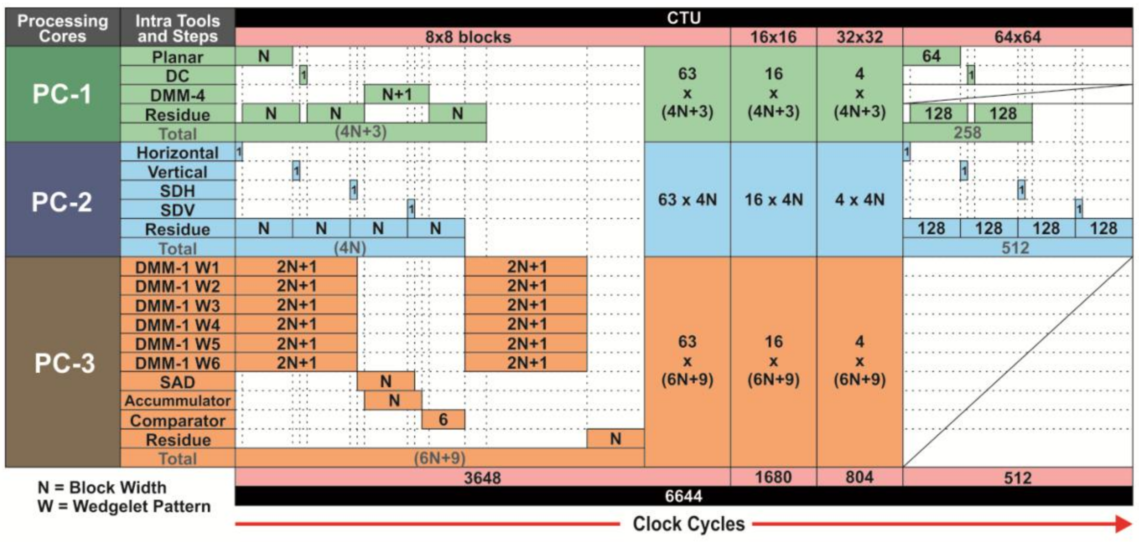


Source: Author.

The designed hardware is a multiplierless solution, where the multiplications were replaced by shift-adds to save hardware resources (and power). Clock-gating was also widely used to idle modules during the architecture operation. The architecture has a parallelism level enough to deliver one block line per clock cycle independently of the block size.

The Processing Core-1 (PC-1) is responsible for processing the Planar, DC, and DMM-4 modes. Each mode is also processed by an independent module, as one can observe in Figure 7.5. As inputs, the PC-1 receives the left and upper edges of neighboring blocks to the block being encoded (previously reconstructed blocks), which are used for DC and Planar calculations. For the DMM-4 calculations, the PC-1 architecture also receives four 8-bit width samples from the texture, which are required for the threshold calculation. PC-1 also receives as inputs the samples of the depth-block being predicted to calculate the residues. As one can see in the Timing Analysis presented in Figure 7.6, the residue calculation starts at the moment that the Planar mode evaluates the first line of samples. The DC calculator needs one cycle before the residue calculation.

Figure 7.6 – Timing analysis of the developed architecture



Source: Author.

The PC-2 is responsible for calculating the IP_H, IP_V, SD_H and SD_V modes and each mode is processed by an independent module, as presented in Figure 7.5. As inputs, PC-2 receives the left and upper edges of neighboring blocks to the block being encoded (previously reconstructed blocks). Also, the PC-2 receives as inputs the samples of the block being predicted to calculate the residues. A multiplexer selects which samples can follow to the Residue Calculator at each moment. The coding of the four modes starts along with the residue calculation, avoiding any additional clock cycle besides the ones used by the Residue Calculator, as presented in Figure 7.6.

The PC-3 is responsible for calculating the DMM-1. As discussed in the last subsection, a heuristic to drastically reduce the DMM-1 computational effort and to avoid the storage of wedgelet patterns in memory was used. The heuristic consists of processing only the six wedgelets formed from the four higher gradients along the four borders of the block to be predicted (one gradient per border), as detailed in the example given in Figure 7.5. PC-3 processes these six wedgelets in parallel, as presented in Figure 7.5 and Figure 7.6. As inputs, PC-3 receives the samples of the block being predicted to calculate the residues.

The first DMM-1 module consists of a Gradient Calculator that generates six outputs determining the start and end points of the six wedgelets. Once the positions to compose the wedgelet are available, the bitmap of the predicted block can be obtained using the Bresenham algorithm in order to indicate which samples belong to each partition. It is important to notice that the Bresenham algorithm does not predict the number of samples the line will have. Thus, it is not possible to predict the total number of required clock cycles. The architecture was designed to cover the worst case of the Bresenham algorithm. The next step is the Block Calculator that starts its operation processing the bitmap line by line. After the Block Calculator generates the predicted block lines, the SAD calculator generates the sum of absolute differences related with each one of the six evaluated wedgelets. Finally, PC-3 has a comparator to compare the six SAD values and decide the most efficient wedgelet before calculating its residue.

7.2.3 Synthesis Results and Comparisons

The developed architecture was synthesized targeting an ASIC technology according to the experimental setup presented in Appendix B. Table 7.5 summarizes the hardware results in the rightmost column. Table 7.5 also presents comparisons with related works. This hardware design is the first to process depth maps that supports both the novel 3D-HEVC Intra-frame prediction modes and the HEVC inherited Intra-frame prediction modes. Although the literature presents some works with hardware designs targeting the HEVC Intra-frame prediction modes, those works do not consider the MVD approach, making unfair the comparisons with the work developed in this doctorate.

Among the related works targeting 3D-HEVC intra-prediction tools, three hardware designs found in the literature were selected for comparisons. It is important to emphasize that the work (SANCHEZ *et al.*, 2014b) implements only the DMM-4, the work (SANCHEZ *et al.*, 2016) implements both DMM-1 and DMM-4, and the work (AMISH *et al.*, 2014) implements DIS, DMM-1, and DMM-4. It is also important to emphasize that all the works (SANCHEZ *et al.*, 2014b), (SANCHEZ *et al.*, 2016) and (AMISH *et al.*, 2017) are not fully compliant with the 3D-HEVC standard.

Table 7.5 – Synthesis results for ASIC technology

Work	Sanchez (2014b)	Sanchez (2016)	Amish (2017)	This Work
Encoding Tools	DMM-4	DMM-1 and DMM-4	DIS, DMM-1, and DMM-4	DIS, DMM-1, DMM-4, H, V, Planar, and DC
Technology	FPGA Altera StratixV 5SGXMABN3F45I32 (28nm)	ASIC 65nm	FPGA Xilinx Virtex 6 (40nm)	ASIC Nangate 45nm
Area	5,568 ALMs 4,405 Registers	219.95 k gates* (2-input NAND)	55 k LUTs 67 k Registers	486,804 gates (2-input NAND)
Memory for DMM-1 wedgelets	-	Yes (101,352 bits*)	Yes (1,947 k bits)	No
Maximum Frequency	31.3 MHz	53.2 MHz*	275 MHz	940 MHz
Frequency for HD1080p@30@5views	22.47 MHz	Not reached	165 MHz	504.5 MHz
Power for HD1080p@30fps	Not available	166.5 mW* (1 view)	Not available	41.57 mW / 0.95 V (9 views)
3D-video processing at UHD2160p@30fps	No (1.74 views)	No (0.25 views)	Yes (2.08 views)	Yes (2.33 views)

*Values consider only the architecture to process 32x32 blocks.

Source: Author.

As one can observe in Table 7.5, only the architecture designed in this work processes HEVC inherited modes (Planar, DC, Horizontal, Vertical) besides the DIS, DMM-1 and DMM-4 modes. Whereas the works (SANCHEZ *et al.*, 2014b) and (AMISH *et al.*, 2017) focused on FPGA devices, this architecture and the work (SANCHEZ *et al.*, 2016) are focused on ASIC technology. This hardware design and the other ASIC-based work present power results and even the hardware developed in this work supporting a high number of prediction modes, it presents promising results.

As one can notice in Table 7.5, only this hardware design and the design presented in (AMISH *et al.*, 2017) are able to process 3D-videos at UHD2160p@30fps (with at least two views). However, the work (AMISH *et al.*, 2017) requires an external memory to store all the possible DMM-1 wedgelets patters, which is not necessary for the architecture developed in this work. Similarly, the work (SANCHEZ *et al.*, 2016) also needs external memory. These works also did not present the power dissipation related to the memory used to store the wedgelets. The developed architecture can process 3D videos at HD1080p@30fps (nine views) with a power dissipation of 41.57 mW whereas the architecture developed by (SANCHEZ *et al.*, 2016) needs 166.5 mW only to process the 32x32 blocks and one view at HD1080p@30fps. It is important to notice that the power dissipation of the architecture developed in this work covers the processing of all supported block sizes (from 4x4 to 64x64) whereas (SANCHEZ *et al.*, 2016) does not present the total power dissipation by processing all block sizes.

Since the other works only support a few Intra-frame prediction modes and do not present the BD-rate results for the complete Intra-frame prediction, as this thesis presented, it was not possible to make a fair comparison of the BD-rate results of these works with the architecture developed in this work. Considering that the developed architecture covers both the novel Intra-frame prediction modes and the ones inherited from the HEVC, the BD-Rate increases 2.64% in the synthetic views under RA temporal configuration. The work (SANCHEZ *et al.*, 2016) increases by 0.09% the BD-Rate in the synthetic views but it supports only DMM-1 and DMM-4 modes. The work (AMISH *et al.*, 2017) increases from 0.436% to 1.906% the BD-Rate in the synthetic views but it covers only DMM-1, DMM-4, and an older DIS implementation.

8 INTER-FRAMES AND INTER-VIEW PREDICTIONS DEVELOPED ARCHITECTURES

This chapter presents in details the two high-throughput architectures developed during this doctorate for the Inter-frames and Inter-view predictions. The first architecture consists of a Motion/Disparity Estimation system with run-time adaptive memory hierarchy that proposes several complexity-reduction heuristics to reduce energy and memory requirements. The second hardware design consists of a coding-efficient Disparity Estimation architecture based on the Improved Unidirectional Disparity-Search Algorithm (iUDS). It is important to mention that the complexity-reduction heuristics proposed for these architectures considers different contexts. The first architecture, i.e., the ME/DE system with run-time adaptive memory hierarchy focuses on an FCO approach in order to take advantage of the channel coding order to save energy, and, therefore, its heuristics are also proposed and evaluated in terms of compression based on the statistical results for the FCO approach in Chapter 6. In the second proposed architecture, i.e., the DE architecture based on the iUDS Algorithm, the focus is on the coding efficiency by considering all PU block sizes. Therefore, the impacts on the compression of the second architecture consider the CCO approach. However, it can be used along with an FCO approach without drastic modifications.

As well as in the Intra-frame developed architectures, both architectures developed for the Inter-frames and Inter-View predictions used pipeline and the clock-gating low-power technique in order to reach processing rates capable of processing high and ultra-high definition videos and, also, to save energy consumption when it is possible.

8.1 Energy-Aware Motion and Disparity Estimation System with Run-time Adaptive Memory Hierarchy

Given the processing/memory-related challenges posed by ME and DE, several related works that propose memory and energy-aware hardware designs for ME and DE targeting previous video coding standards have been published in recent years. Some proposals focus on optimizing the processing-related issues by reducing the computational effort (KIM *et al.*, 2014) or designing energy-efficient hardware architectures (DING *et al.*, 2010). Other works propose solutions aiming at efficient memory organization, sizing, and management (ZATT *et al.*, 2011a) (SAMPAIO *et al.*, 2013) (SONG *et al.*, 2015). Finally, a set of works proposes complete ME and/or DE systems by jointly discussing memory and processing-unit designs (ZATT *et al.*, 2011b). There are many insightful published hardware solutions focusing on energy efficiency, i.e., taking into account processing and memory issues. However, there are no publications based on these approaches for 3D-HEVC.

Actually, most of the 3D-HEVC-related papers regard algorithmic solutions only, and several of them present solutions already implemented in the 3D-HEVC Reference Software. ME and DE are applied to both texture and depth-map channels in 3D-HEVC. However, there are no ME/DE hardware designs capable of handling MVD format in the current literature. Since these channels present completely different characteristics, it may be beneficial to choose the channel to be first processed. 3D-HEVC enables this feature through the Flexible Coding Order (FCO) tool (GOPALAKRISHNA *et al.*, 2013) that allows the encoding of depth maps before their associated texture frames, except for the Base View. In this work, the FCO is exploited regarding memory and processing issues to reduce energy consumption.

Therefore, this thesis proposes a 3D-HEVC Motion and Disparity Estimation system designed for low-energy consumption. This system features a dedicated memory hierarchy capable of run-time adaptation. The processing unit employs FCO along with a series of simplifications and optimizations targeting at computational effort reduction through data behavior observations and inter-channel/inter-view redundancies exploration. Statistical analysis was performed and motivated the proposal of an efficient average-case hardware design. Among the main contributions are the proposed of the Hardware-Oriented Test Zone Search (HOTZS) with static scheduling, which is a hardware-friendly version of the conventional HEVC/3D-HEVC fast motion estimation algorithm. It was proposed to allow the design of a high-throughput ME architecture. Also, the Horizontal Disparity Search (HDS) technique is proposed to avoid vertical search based on the horizontal-only camera displacement and propitiate a simplified and efficient disparity-estimation hardware design. The proposed system employs distributed on-chip memories associated with the processing units. Each memory features window-based data reuse and is composed of multiple sectors enabling independent control via Dynamic Voltage Scaling (DVS). Sub-sampling is applied to depth-map memories whereas Horizontal Disparity Search (HDS) allows memory size reduction. A Depth-Based Dynamic Search Window Resizing (DSWR) algorithm is proposed to dynamically reduce the search window during texture ME/DE based on depth information, allowing dynamic power management through DVS of idle memory regions. The Texture-Based Motion Vector Inheritance (TMVI) technique takes advantage of the ME/DE behavior and inter-channel correlation to reduce energy consumption. The next subsection presents the definition of a Baseline Encoder (B-Encoder) configuration used throughout this system development based on extensive tests performed using the 3D-HTM reference software.

8.1.1 3D-HEVC Baseline Encoder Definition

As previously discussed, the intensive computation and the large amount of data handled by ME and DE lead to a large amount of memory access and high energy consumption in 3D-HEVC. These tools must be simplified by applying constraints/modifications to their processes in order to make the system implementation feasible. This means there is a need to define a set of hardware-oriented constraints focusing on complexity and communication reductions. However, applying such constraints usually impact negatively on the compression rates and image quality. Therefore, extensive tests were performed to evaluate the impact of different constraints on the 3D-HEVC encoder. This evaluation allowed the definition of a B-Encoder configuration used throughout this system development and evaluation.

The experimental setup used to define the B-Encoder configuration, and an analysis that led to the definition of the ME/DE hardware-oriented constraints were performed with the 3D-HTM in version 16.2 (3D-HEVC Reference Software, 2019). The experiments follow the 3D-HEVC CTC document, recommended by the JCT-3V (see Appendix A). Therefore, the eight 3D video sequences (with three texture plus depth views), including five 1920x1088 (HD 1080p) sequences and three 1024x768 sequences (HD 760p) were used under the four recommended sets of QPs to encode texture/depth: 25/34, 30/39, 35/42, and 40/45. Also, the experiments considered the RA temporal configuration with 48 AUs, the three views defined by the CTC, and the Search Range (SR) of 64 pixels.

To define the ME/DE hardware-oriented constraints (HC), an incremental analysis was performed to verify the impact of them on the coding efficiency. After an extensive analysis of each HC considering different scenarios, one of these scenarios was adopted as the baseline for the next HC evaluation, and so forth. Each one of the ME/DE HCs is discussed in the following, and all results are summarized at the end of this section. The coding efficiency degradation is measured using the BD-Rate metric, which represents the percentage variation in bit rate considering the same objective image quality in comparison to the original 3D-HTM encoder. The coding efficiency results are discussed for video total (i.e., the quality related to texture channel and the total bit rate of the video) and for synthesized frames (i.e., the quality of frames synthesized with texture and depth using DIBR).

8.1.1.1 Block-Size Constraint (HC1)

The first evaluation was performed to choose a subset of block sizes (known as PU – Prediction Unit) to be used in ME/DE process and avoid the evaluation of all 24 sizes possible in 3D-HEVC. In this hardware-constraint definition, the possible sizes were reduced to two. The choice of two PU sizes among 24 possibilities requires a proper analysis to identify the ones that yield a better tradeoff between complexity reduction and bit-rate increase. Since 276 combinations of two PU sizes are possible considering 24 PU sizes, a complete analysis of this space using the 3D-HTM is unfeasible. Given that the four square-shaped blocks (8x8, 16x16, 32x32, and 64x64) are the most frequently used and representative sizes in 3D-HEVC (see Section 6.2), the best combination of two square-shaped blocks (six possible combinations) was evaluated in this work. The results regarding BD-Rate for these six combinations are presented in Table 8.1. Note that two test cases present a BD-Rate increase lower than 7% considering synthesized views. These two promising test cases consider only 16x16 and 32x32 PU sizes, or only 16x16 and 64x64 PU sizes in ME/DE and present very similar results. Thus, based on the BD-Rate results obtained, the 16x16 and 32x32 block sizes were adopted as the ME/DE block-size constraint since this approach allows the development of more efficient hardware solutions considering processing and memory-related issues.

The complete BD-Rate results, according to the 3D-video sequences, can be observed in Appendix C.

Table 8.1 – BD-Rate increase according to the Block-Size Constraint (HC1)

Block-Size Constraint **(HC1)**	**Type of Pictures**	**Average BD-Rate Increase (%)**		
		1024x768	**1920x1088**	**All Sequences**
8x8 and 16x16	Video Total	9.07	15.12	**12.85**
	Synth.	8.44	14.07	**11.96**
8x8 and 32x32	Video Total	5.44	10.17	**8.39**
	Synth.	5.12	9.43	**7.81**
8x8 and 64x64	Video Total	7.56	13.42	**11.22**
	Synth.	7.08	12.55	**10.49**
16x16 and 32x32	Video Total	3.81	8.92	**7.00**
	Synth.	4.19	8.44	**6.85**
16x16 and 64x64	Video Total	4.38	8.54	**6.98**
	Synth.	4.58	8.10	**6.78**
32x32 and 64x64	Video Total	5.30	14.11	**10.81**
	Synth.	5.92	13.64	**10.75**

Source: Author.

8.1.1.2 Block-Matching Algorithm Constraint (HC2)

A major portion of the 3D-HEVC ME/DE complexity is related to the TZS algorithm. Although TZS does not compare all reference blocks in a given SW, the computational effort related to TZS remains large in the context of real-time 3D-HEVC encoding. As previously explained, TZS is composed of four main steps: (i) Prediction; (ii) First Search; (iii) Raster; (iv) Refinement. This way, in this analysis some of the TZS steps were disabled, and the different combinations of steps were evaluated. Prediction is the first TZS step, and it employs five different ways to select the SW position in the reference frame: one based on the collocated block (a.k.a., Zero Predictor) and four predictors based on neighboring blocks. By applying predictors based on neighboring blocks, TZS can move the SW to a distant region from the collocated block position and, consequently, increase the memory access overhead. This way, only the Zero Predictor (which does not move the SW) was maintained in TZS for all evaluations.

Additionally, three different levels of TZS simplifications were tested, called TZS1, TZS2, and TZS3, as defined in Table 8.2. The table shows which steps are enabled in each test case. Based on the results of the experiments, the use of TZS without Prediction (except Zero Predictor) and Raster steps was adopted, since this approach presented better trade-off among complexity reduction, BD-Rate increase, and hardware resource usage. TZS3 simplification presents very aggressive image degradation, increasing the BD-Rate in about 40%. TZS1 and TZS2 simplifications present similar BD-Rate results, from 10.14% to 12.09% for video total, and similar total encoding time reduction from 37.74% to 41.10%, respectively. However, by removing the Raster step (TZS2), hardware resource usage, processing, and memory requirements are saved since a considerable number of candidate blocks are avoided. It is important to notice that the Raster step is responsible for testing all candidate blocks inside the search window considering a subsampling parameter. Table 8.3 presents the results in terms of BD-Rate when the constrained TZS is applied over the block-size constraint (HC1). The complete BD-Rate results, according to the 3D-video sequences, can be observed in Appendix C.

Table 8.2 – TZS simplifications according to the enabling/disabling steps

TZS Steps	**BMA Constraint**		
	TZS1	**TZS2**	**TZS3**
Zero Predictor Only	X	X	X
First Search	X	X	X
Raster	X		
Refinement	X	X	

Source: Author.

Table 8.3 – BD-Rate increase according to the Block-Matching Algorithm Constraint (HC2)

BMA Constraint **(HC2)**	**Type of Pictures**	**Average BD-Rate Increase (%)**		
		1024x768	**1920x1088**	**All Sequences**
TZS1 + HC1	Video Total	4.54	13.50	**10.14**
	Synth.	4.75	12.81	**9.79**
TZS2 + HC1	Video Total	4.85	16.44	**12.09**
	Synth.	5.05	15.57	**11.63**
TZS3 + HC1	Video Total	12.28	55.92	**39.55**
	Synth.	11.11	53.56	**37.64**

Source: Author.

8.1.1.3 Vertical-Disparity Constraint (HC3)

Although 3D-HEVC does not impose any limitation regarding the view arrangement (i.e., the position of the cameras), its coding tools were designed to be more efficient when the views are aligned in 1D linear and coplanar arrangements (TECH *et al.*, 2016). Furthermore, the camera arrangement used to record the videos follows a horizontal displacement (MÜLLER *et al.*, 2014)(TECH *et al.*, 2016). Nevertheless, 3D-HTM defines TZS as the BMA applied to the DE prediction and a vertical disparity range equals to 56 by default. In other words, 3D-HTM performs a 2D search in a scenario where only 1D displacement is expected, demanding unnecessary computation by using an inappropriate algorithm for DE prediction.

In summary, given the horizontal-only displacement between cameras, such range is unnecessary for most cases. Therefore, an evaluation considering the horizontal-only disparity was tested, aiming at reducing the memory traffic and its related issues. The strategy increases the accumulated BD-Rate to 13.22% and 14.65% for synthesized views and video total, respectively, when applied over the block-matching algorithm constraint (HC2), as presented in Table 8.4. Based on these results, the use of horizontal disparity was adopted, since it presented a small impact in the accumulated BD-Rate.

Table 8.4 – BD-Rate increase according to the Vertical-Disparity Constraint (HC3)

Sequence	**Average BD-Rate Increase (%) according to the HC3 (Vertical-Disparity Constraint + HC2)**	
	Video Total	**Synthesized**
Balloons	5.76	5.74
Kendo	9.08	8.02
Newspaper_CC	6.20	5.47
GT_Fly	26.84	26.10
Poznan_Hall2	18.81	14.45
Poznan_Street	17.11	13.20
Undo_Dancer	8.86	8.06
Shark	24.58	24.72
1024x768	7.01	6.41
1920x1088	19.24	17.31
All sequences	14.65	13.22

Source: Author.

8.1.1.4 Additional ME/DE Constraints (HC4)

3D-HTM defines SAD as similarity criterion for the Integer ME/DE, while the Sum of Absolute Transformed Differences (SATD), based on the Hadamard Transform, is applied for the Fractional ME/DE (not used in depth maps), increasing the computational effort. Also, 3D-HTM supports bi-directional prediction and several reference frames for each direction. To reduce the ME/DE computational effort, memory and hardware requirements, three additional modifications were performed: SAD was used for Fractional ME/DE instead of SATD, the support to bi-directional prediction was disabled, and the search was limited to one reference frame per direction (both in ME and DE). The accumulated results show a BD-Rate increase of up to 24.16% when these modifications are applied over the vertical-disparity constraint (HC3), as presented in Table 8.5. When these additional ME/DE constraints were implemented in the 3D-HTM reference software, the total encoding time was reduced by 59%. Based on that, the additional ME/DE constraints were adopted due to the great reduction in the 3D-HEVC ME/DE complexity.

Table 8.5 – BD-Rate increase according to the Additional ME/DE Constraints (HC4)

Sequence	**Average BD-Rate Increase (%) according to the HC4 (Additional ME/DE Constraints + HC3)**	
	Video Total	**Synthesized**
Balloons	7.21	8.38
Kendo	21.51	19.86
Newspaper_CC	6.90	7.14
GT_Fly	42.96	42.23
Poznan_Hall2	22.89	20.08
Poznan_Street	18.99	17.53
Undo_Dancer	17.31	16.10
Shark	55.51	54.47
1024x768	11.87	11.79
1920x1088	31.53	30.08
All sequences	24.16	23.22

Source: Author.

8.1.1.5 ME/DE Constraint Analyses Summary

Table 8.6 summarizes the BD-Rate increase of all ME/DE hardware-oriented constraints. One may notice that by using all constraints (HC4), BD-rate is increased by 23.22% and 24.16% for synthesized views and video total, respectively. The simplifications were implemented in the 3D-HTM reference software, and the encoding computational effort was reduced by 59%. Therefore, the modified 3D-HEVC with the applied ME/DE constraints was adopted as the Baseline Encoder for the Energy-Aware 3D-HEVC ME/DE System developed in this work and presented in the next section. The complete BD-Rate results, according to the 3D-video sequences, can be observed in Appendix C.

The accumulated coding-efficiency losses are not negligible, but the constraints are necessary when real-time encoders for mobile devices are considered. The need for simplifications becomes clear when we observe that related works focusing on HEVC (less complex than 3D-HEVC) also employ similar constraints, such as a limited number of PU sizes (AFONSO *et al.*, 2013)(AFONSO *et al.*, 2016), simplified BMAs (DOAN *et al.*, 2017)(HE *et al.*, 2015)(PASTUSZAK *et al.*, 2016b), SAD as the unique similarity criterion (PASTUSZAK *et al.*, 2016b), and a limited number of reference frames (PASTUSZAK *et al.*, 2016b). Unfortunately, most of these works do not present detailed results regarding the losses in terms of compression efficiency due to the adopted simplifications. Other works present comparisons with configurations that do not consider all encoding tools available in the reference software by default so that the full impact of the simplifications cannot be estimated. In this work, the ME/DE system was developed after the base configuration is carefully defined and characterized.

Table 8.6 – BD-Rate increase according to the ME/DE Hardware-oriented Constraint (HC)

HC	**Constraints**	**Type of Pictures**	**Average BD-Rate Increase (%)**		
			1024x768	**1920x1088**	**All sequences**
1	**Block sizes (16x16 and 32x32)**	Video Total	3.81	8.92	**7.00**
		Synth.	4.19	8.44	**6.85**
2	**BMA (TZS without *Predictors* and *Raster*) + HC1**	VideoTotal	4.85	16.44	**12.09**
		Synth.	5.05	15.57	**11.63**
3	**Horizontal Disparity Only + HC2**	VideoTotal	7.01	19.24	**14.65**
		Synth.	6.41	17.31	**13.22**
4	**Additional ME/DE Constraints + HC3**	VideoTotal	11.87	31.53	**24.16**
		Synth.	11.79	30.08	**23.22**

Source: Author.

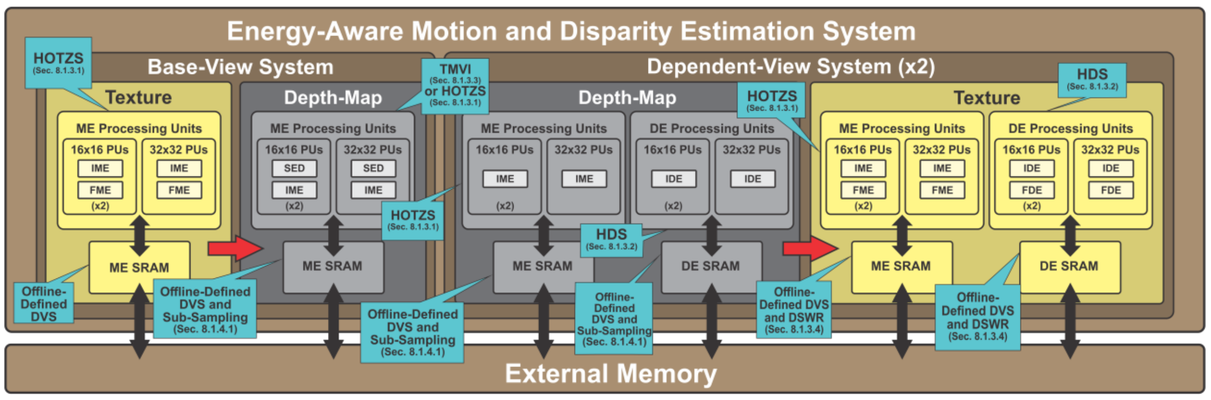
8.1.2 Developed 3D-HEVC ME/DE System

In order to make clear the understanding of the system development, this section is divided into two subsections. The first one presents an overview of the developed energy-aware ME/DE system, whereas the second describes the design of the dedicated processing units. After, Section 8.1.3 shows the developed hardware-oriented BMA and scheduling, and Section 8.1.4 presents the developed on-chip memory design and management.

8.1.2.1 Energy-Aware ME/DE System Overview

Figure 8.1 shows the diagram of the developed energy-aware 3D-HEVC Motion and Disparity Estimation architecture. The architecture is composed of three main hardware units, one responsible for processing the Base View (BV) and two for the Dependent Views (DVs) (due to space restrictions only one instance is depicted in Figure 8.1). The Base View, or Independent View, is the first one to be encoded and it does not depend on other views, allowing compatibility with 2D HEVC-based systems. The remaining ones are Dependent Views, i.e., those whose encoding process depends on information from other views. The Base-View unit has one ME processing unit per channel (texture and depth), whereas each Dependent-View unit has one ME and one DE processing unit per channel. Each processing unit has a private on-chip SRAM memory. Every processing unit and on-chip memory features specific optimizations and observe distinct requirements according to the data characteristics, processing requirements, and available information from previously processed data. Figure 8.1 summarizes the techniques applied to each unit, which are detailed in Sections 8.1.3 and 8.1.4.

Figure 8.1 – Energy-aware ME/DE system

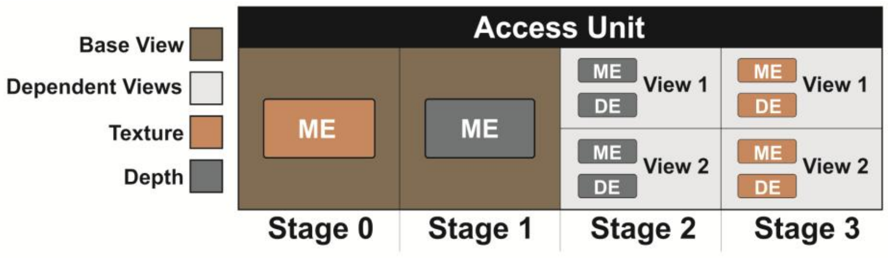

Energy-Aware Motion and Disparity Estimation System
Base-View System
Dependent-View System (x2)
HOTZS (Sec. 8.1.3.1)
Texture
Depth-Map
TMVI (Sec. 8.1.3.3) or HOTZS (Sec. 8.1.3.1)
HDS (Sec. 8.1.3.2)
ME Processing Units
DE Processing Units
16x16 PUs
32x32 PUs
IME
FME
SED
IDE
FDE
(x2)
ME SRAM
DE SRAM
Offline-Defined DVS
Offline-Defined DVS and Sub-Sampling (Sec. 8.1.4.1)
Offline-Defined DVS and DSWR (Sec. 8.1.3.4)
External Memory

Source: (AFONSO *et al.*, 2019).

To guarantee HEVC backward compatibility, it is mandatory for the Base View to process the texture channel before the depth channel. Observing this restriction and aiming at efficiently exploiting the inter-channel correlation, the Texture unit provides valuable information related to the ME result to the Depth-Map unit that, under specific situations, may completely skip the search process according to the proposed Texture-Based Motion Vector Inheritance (TMVI) algorithm (Section 8.1.3.3). To further exploit the inter-channel correlation, the proposed system employs the Flexible Coding Order for the Dependent Views. This allows the Depth-Based Dynamic Search Window Resizing (DSWR) to adapt the SW with proper knowledge of the scene as both disparity and motion displacements are related to depth (Section 8.1.3.4). The red arrows in Figure 8.1 represent auxiliary information flow - used by the proposed algorithms.

The proposed system is implemented in a 4-stage macro-pipeline at the frame level, as depicted in Figure 8.2. The first stage processes texture for the Base View and the second processes depth for the Base View. In turn, the third and fourth stages process (in parallel) ME and DE of all Dependent Views for depth maps and texture, respectively. As the number of cycles spent in each stage varies according to data characteristics, once a unit has concluded its task, clock gating is applied for energy reduction.

Figure 8.2 – ME/DE system macro-pipeline



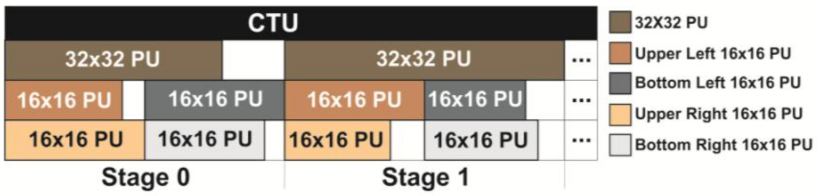
Source: (AFONSO *et al.*, 2019).

8.1.2.2 Processing Units Design

The dedicated processing units were developed according to average-case design strategy observing the 3D-HEVC reference software behavior. Each processing unit was defined to process 16 lines in parallel, where each line is composed of 16 or 32 samples, depending on the block size (Section 8.1.1.1). Whenever 16x16 blocks are processed, part of the logic circuit is clock-gated. Based on a statistical analysis of the number of candidate blocks during the BMA process, 240 candidates was defined as the target performance. Such performance was calculated considering the adoption of the previously described constraints.

ME/DE processing units present a similar behavior. They both divide each 64x64 CTU into four 32x32 CUs, sequentially processing each one. The CU is evaluated as one 32x32 block and as four 16x16 blocks. At the beginning of the process, the 32x32 and both upper 16x16 blocks begin to be processed in parallel. Immediately after, the remaining ones start to be processed in parallel, as detailed in Figure 8.3.

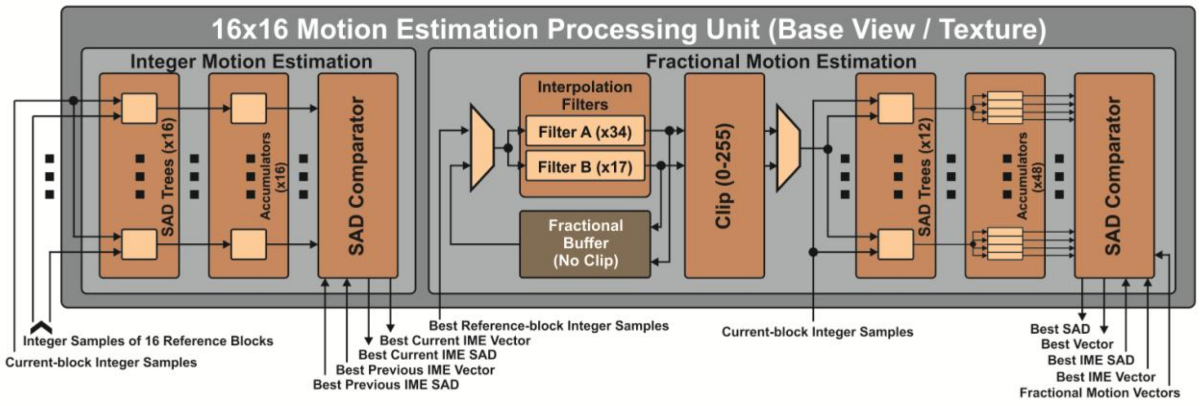
Figure 8.3 – ME/DE Block-encoding process macro-pipeline



Source: (AFONSO *et al.*, 2019).

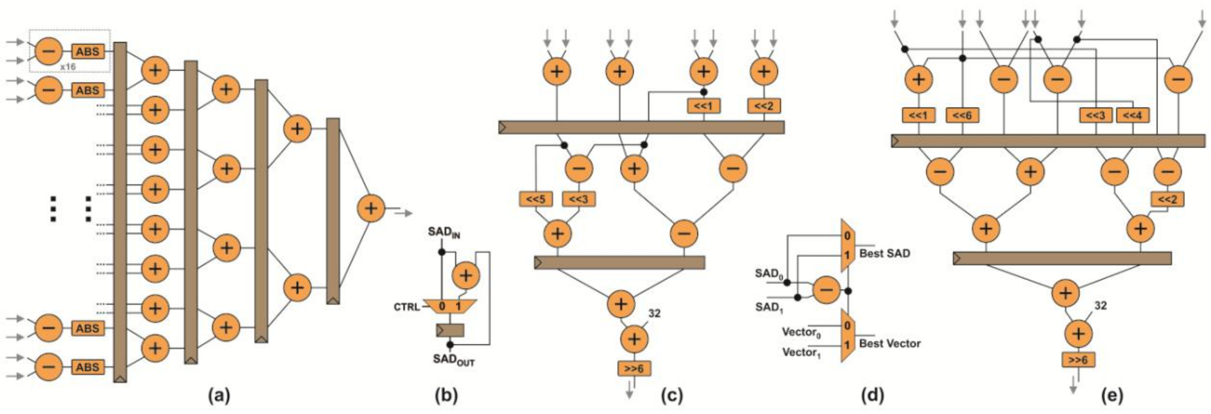
Figure 8.4 presents the architecture for the operative part of a 16x16 ME processing unit (a similar structure is used for DE and 32x32 units) used in the coding process of texture channel for the Base Views. Figure 8.5 details the basic hardware structures that compose the processing unit: (a) SAD tree; (b) SAD accumulator; (c) Type-A interpolation filter; (d) SAD comparator; and (e) Type-B interpolation filter. The interpolation filters are used in the Fractional Motion Estimation (FME) processing. This step is applied to the best candidate found by the IME in order to improve the coding efficiency. By default, FME is not applied to the depth map.

Figure 8.4 – 16x16 ME processing unit used in the texture processing for the Base View


16x16 Motion Estimation Processing Unit (Base View / Texture)
Integer Motion Estimation
SAD Trees (x16)
Accumulators (x16)
SAD Comparator
Fractional Motion Estimation
Interpolation Filters
Filter A (x34)
Filter B (x17)
Fractional Buffer (No Clip)
Clip (0-255)
SAD Trees (x12)
Accumulators (x48)
SAD Comparator
Integer Samples of 16 Reference Blocks
Current-block Integer Samples
Best Reference-block Integer Samples
Best Current IME Vector
Best Current IME SAD
Best Previous IME Vector
Best Previous IME SAD
Current-block Integer Samples
Best SAD
Best Vector
Best IME SAD
Best IME Vector
Fractional Motion Vectors

Source: (AFONSO *et al*., 2019).

Figure 8.5 – Basic hardware structures: (a) SAD Tree of 16x16 processing units; (b) SAD accumulator; (c) Type-A interpolation filter; (d) SAD comparator of two blocks; (e) Type-B interpolation filter.



Source: (AFONSO *et al*., 2019).

8.1.3 Hardware-Oriented BMA and Scheduling

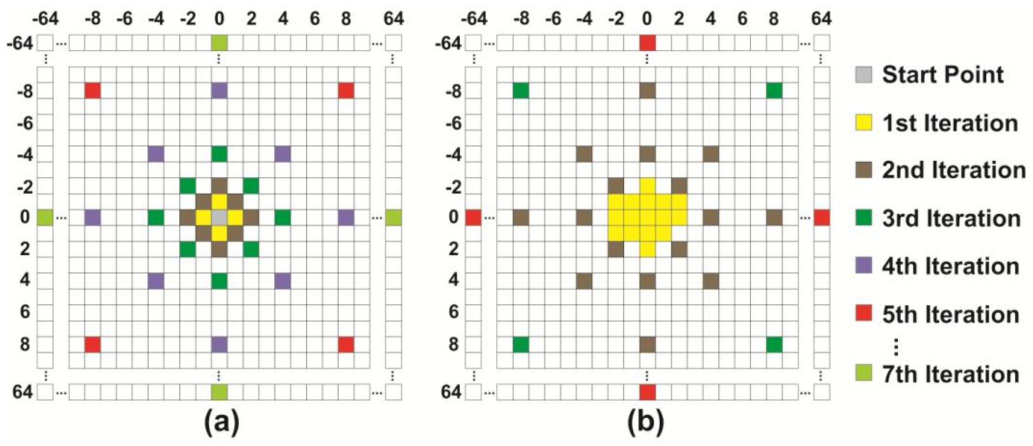
This section is divided into four subsections. The first one presents the Hardware-Oriented TZS static scheduling (HOTZS). The second shows the developed Horizontal Disparity Search (HDS) algorithm. The third describes the developed Texture-Based Motion

Vector Inheritance (TMVI) algorithm. Finally, the fourth subsection presents the Depth-Based Dynamic Search Window Resizing (DSWR) algorithm.

8.1.3.1 Hardware-Oriented TZS Static Schedule (HOTZS)

As described in Section 8.1.1.2, the TZS algorithm was modified by removing two main steps. By removing the Prediction step, search window displacement is avoided, and data reuse is improved. Computational effort is reduced by removing the Raster step. On top of that, a static TZ search scheduling (HOTZS) to provide a more regular memory access pattern and improve the Processing Units usage was also defined. As it can be observed in Figure 8.6-a, the TZS is composed of multiple iterations composed of 1, 4, 8 or 16 candidate blocks each. To use the 16-line parallelism, HOTZS rearranges the processing schedule to process 16 candidates per iteration, as shown in Figure 8.6-b.

Figure 8.6 – Processing of candidates according to each iteration: (a) TZS; (b) HOTZS



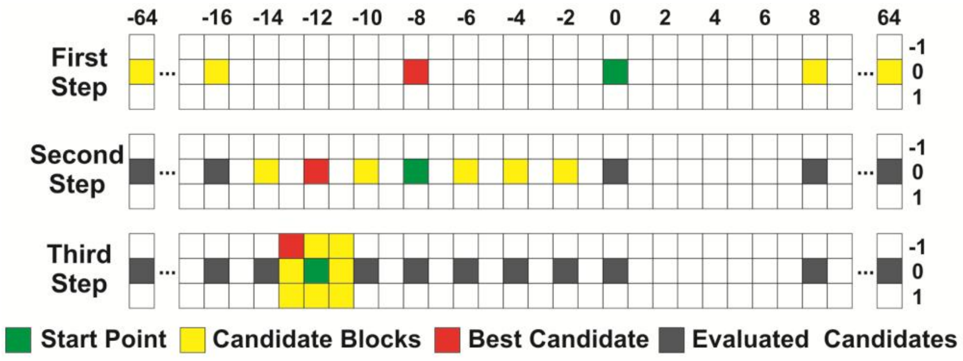
Source: (AFONSO *et al*., 2019).

8.1.3.2 Horizontal Disparity Search (HDS)

Since 3D-HEVC is meant to encode horizontally displaced video streams, the diamond-shaped search pattern employed by TZS leads to two drawbacks during the DE process: (i) unnecessary computation and increased search window, resulting from vertical block matchings; (ii) local minima trapping when TZS converges in the first iterations, leading to inaccurate disparity field estimation.

Therefore, the Horizontal Disparity Search (HDS) algorithm, composed of three steps, was proposed to exploit the horizontally displaced video characteristic. The First Step consists of a horizontal search from the start point comparing the current block with candidate blocks eight samples apart from each other (i.e., a subsampled horizontal-only raster search), represented in the upper part of Figure 8.7 by the colored boxes in the First Step. Similarly, the Second Step performs the horizontal search, but the distance between candidates is two samples. The Second Step is centered at the best block matching of the First Step (red box in the example of Figure 8.7). Finally, the Third Step performs a bidirectional search around the best matching obtained from the Second Step. Although there is no real vertical disparity, a 1-sample vertical search refinement is employed to assure good matching under slight cameras vibrations or captured noise. The HDS allows reducing the 192x192-sample memory for DE to a 192x66-sample memory. Additionally, the number of block matchings during HDS is constant, simplifying memory-access pattern and processing-unit design.

Figure 8.7 – Horizontal Disparity Search algorithm



Source: (AFONSO *et al.*, 2019).

8.1.3.3 Texture-Based Motion Vector Inheritance (TMVI)

Texture-Based Motion Vector Inheritance (TMVI) was implemented aiming at exploring the correlation between texture and depth map for the Base View. Although there is a correlation between the texture and the depth channels, texture information does not exist in depth maps. This way, different motion vectors lead to good matchings for homogeneous regions of a depth map (regions generally inside the objects), which includes the motion vector selected during the texture encoding of the collocated block.

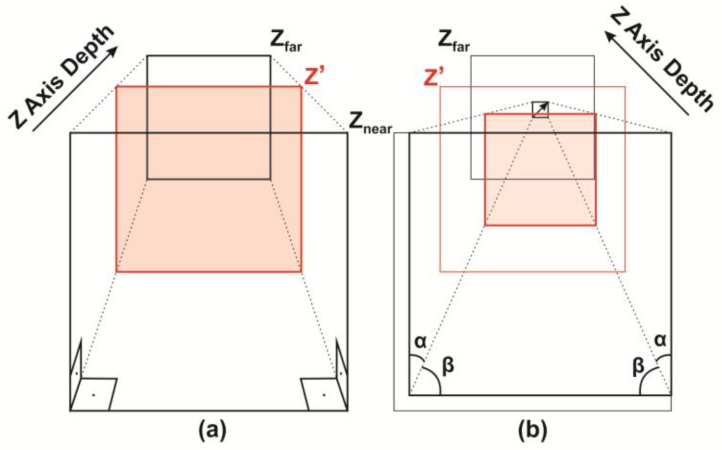
The TMVI calculates the Simplified Edge Detector (SED) metric (SANCHEZ *et al.*, 2014a) to define if the depth block under processing is a sharp-edge or homogeneous region. In the case of sharp-edge detection, the HOTZS is normally applied to the depth map ME. Otherwise, for homogeneous regions, the motion vector used for the collocated block in the texture channel is inherited by the depth block. Thus, a complete search is skipped (leading to block matching complexity reduction) and the memory associated with the depth ME at the Base View may be scaled to idle mode.

8.1.3.4 Depth-Based Dynamic Search Window Resizing (DSWR)

DSWR is implemented to dynamically reduce the search window by exploiting depth information in the texture ME and DE processes. Typical implementations employ a constant search window along the whole ME/DE process, disregarding the block/object depth within the scene, as shown in Figure 8.8-a. During the ME process, DSWR exploits the fact that the displacement (motion vectors) of objects/blocks moving at the same speed is inversely proportional to the distance between object and camera (depth), i.e., the closer the object, the larger the motion vector (for objects moving at same speed) and, thus, a larger SW is necessary. The same applies to DE estimation, i.e., closer objects present larger disparity vectors.

Therefore, DSWR scales the active SW (shaded area in Figure 8.8) according to a perspective projection, as depicted in Figure 8.8-b. Additionally, DSWR considers the disparity/motion vectors selected during the depth-map ME/DE to displace the active SW by applying a projection distortion. This behavior is based on the observation that collocated blocks of the same view (texture and depth) represent the same scene projection and, consequently, tend to present similar motion and disparity vectors. In Figure 8.8, the Z_{near} and Z_{far} correspond to the depth-sample values of the closest and farthest objects in the scene respectively. *Z'* corresponds to the current block depth value to be considered for the DSWR resizing.

Figure 8.8 –Search window size: (a) Fixed search window size; (b) Depth-Based Dynamic Search Window Resizing



Source: (AFONSO *et al.*, 2019).

Figure 8.9 presents the algorithm used to define the SW resizing simplified to one dimension (horizontal in this case). First, the horizontal component of the vector (MV_x) of the collocated block is obtained (line 2). Afterward, the average depth of the collocated depth-map block is stored in the S variable (line 3). Then, an offset is added to the module of MV_x and stored in the absMV (line 4). After, the vector orientation is extracted (line 5). The search range right side (srchRngPosSide) is defined using Z_{near} and S (lines 6-7). Similarly, the search range left side (srchRngNegSide) is defined (lines 8-12). Finally, the Search Range is placed according to the vector orientation (lines 13-19) and mapped to an SW rounded to multiple of 16 samples (lines 20-21) to fit the memory sector organization.

Figure 8.9 – DSWR algorithm pseudo-code.

```
1.  //Define the X dimension of texture search range
2.  $MV_x$ ← getCorrelatedDepthVector();// get correlated vector in depth map
3.  S ← getCorrelatedDepthAverage();//get correlated depth-map block average
4.  absMV ← abs($MV_x$) + OFFSET;//insert an offset in the obtained vector
5.  signal ← ($MV_x$≥0) ? 1 : -1;  //save the obtained vector signal
6.  a ← $\frac{(64-absMV)}{Z_{near}}$;//define the a constant
7.  srchRngPosSide ← $a \times (S - Z_{near}) + 64$;  //define size of positive side ofSR
8.  If($MV_x$≠0)Then//define the size of negative side of SR
9.      srchRngNegSide ← srchRngPosSide–absMV;
10. Else
11.     srchRngNegSide ← srchRngPosSide;
12. End If
13. If(signal = 1)Then//define the new search range
14.     $SR_{RIGHT}$ ← srchRngPosSide;
15.     $SR_{LEFT}$ ← srchRngNegSide×(-1);
16. Else
17.     $SR_{RIGHT}$ ← srchRngNegSide;
18.     $SR_{LEFT}$ ← srchRngPosSide×(-1);
19. End If
20. $SW_{RIGHT}$ ← roundSrchWin($SR_{RIGHT}$); //round the new search window
21. $SW_{LEFT}$ ← roundSrchWin($SR_{LEFT}$);
```

Source: (AFONSO *et al.*, 2019).

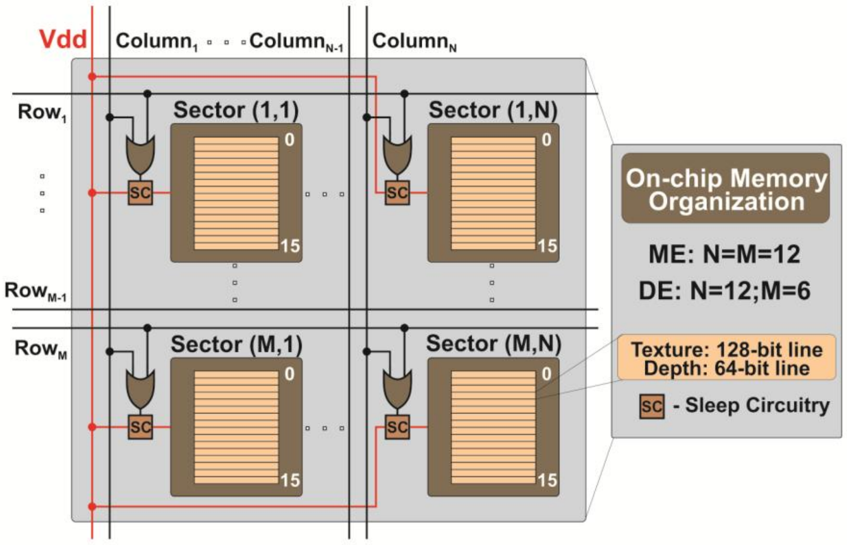
8.1.4 On-chip Memory Design and Management

This section is divided into two main subsections. The first subsection presents the memory organization and sizing, and the second subsection shows the techniques proposed for memory management.

8.1.4.1 Memory Organization and Sizing

The proposed system employs four different types of on-chip memories. However, both ME and DE memories follow the same memory organization presented in Figure 8.10, independently of the channel (texture or depth). The ME on-chip memories cover an SW of 192x192 samples and are composed of 144 sectors, organized in 12 columns and 12 rows. Each sector stores 16 128-bit words, i.e., each word is composed of 16 samples of 8 bits for texture. As the main benefit of our HDS algorithm (Section 8.1.3.2), the DE memories only need to store 192x66 samples within 72 sectors, organized in 12 columns and six rows (upper and bottom sectors distributed in the six rows store only one additional word). By applying the Depth Sub-Sampling technique presented in the following, which reduces the sample representation to 4 bits, the memories for depth maps comprise 64-bit lines.

Figure 8.10 – Organization of on-chip memory units



Source: (AFONSO *et al.*, 2019).

In the MVD format, depth map pixels are represented using one channel composed of 8-bit samples. Additionally, depth map content is dominated by smooth regions (within objects and background) and very sharp edges (object borders), i.e., there are no complex-textured regions. Observing these characteristics, a bit-depth subsampling is implemented to reduce the size of on-chip memories for depth maps by 50% (subsampling from 8 bits down to 4 bits) with small coding efficiency drawback. For that, four sub-sampling patterns were evaluated. The evaluated patterns remove the (i) four more significant bits; (ii) four less significant bits; (iii) odd bits; and (iv) even bits, as presented in Table 8.7. As the elimination of the four less significant bits demonstrated the lowest drawback in coding efficiency (0.59% in BD-Rate), it was adopted for the design of depth maps memories. This evaluation used the experimental setup shown in Section 8.1.1.

Table 8.7 – BD-Rate increase according to the depth maps sub-sampling

Sub-Sampling Strategy	**Type of Pictures**	**Average BD-Rate Increase (%)**		
		1024x768	**1920x1088**	**All Sequences**
SubSamp = Samp & 0x0F	Video Total	0.28	2.52	1.68
	Synth.	1.16	3.87	2.85
SubSamp = Samp & 0xF0	Video Total	-0.06	0.50	0.29
	Synth.	0.21	0.81	0.59
SubSamp = Samp & 0x55	Video Total	-0.18	0.94	0.52
	Synth.	0.31	1.60	1.12
SubSamp = Samp & 0xAA	Video Total	-0.06	1.33	0.81
	Synth.	0.34	1.87	1.30

Source: Author.

Table 8.8 presents the total memory size for each memory type. Note that DE memories are smaller than ME memories, and depth-map memories are smaller than texture memories due to the proposed techniques.

Table 8.8 – Memory sizing

Memory	**#samples**	**#bits per sample**	**#bytes per instance**	**#sectors (rows x columns)**	**Instances**	**Total Size**
ME – Texture	192x192	8	36.86kB	144 (12x12)	3	110.58kB
ME – Depth Map	192x192	4	18.43kB	144 (12x12)	3	55.29kB
DE – Texture	192x66	8	12.67kB	72 (6x12)	2	25.34kB
DE – Depth Map	192x66	4	6.34kB	72 (6x12)	2	12.68kB
Total	-	-	**74.3kB**	-	-	**203.89kB**

Source: Author.

8.1.4.2 Run-Time Adaptive Memory Management

The following paragraphs present the power model considered in this work and the proposed run-time adaptive memory management.

Sector-level dynamic voltage scaling (DVS) is applied to reduce leakage energy in idle sectors. The power state management is performed using a power-state matrix signaled by Column_x and Row_y in Figure 8.10. The Sleep Circuitry (SC in Figure 8.10) is responsible for actuating on the sector voltage (V_{dd}) switching between three power states: PS_{on} (V_{dd}=1V), PS_{idle} (V_{dd}=0.6V), and PS_{off} (V_{dd}=0V), where PS_{on} and PS_{idle} are data retentive. The nvsim (NVSIM, 2019) simulator along with the 45nm CMOS technology power models (WANG et al., 2015) was used to estimate the energy consumption for leakage, reading, writing, and power state transition, as summarized in Table 8.9. The wakeup latency was considered negligible for this analysis.

Table 8.9 – Power states energy

Power State	**Vdd(V)**	**Energy (J)**					
		Per bit			**Per Transition**		
		Leak	**Write**	**Read**	PS_{on}	PS_{idle}	PS_{off}
PS_{on}	1.0	8.467f	0.082p	0.127p	-	0	0
PS_{idle}	0.6	4.472f	-	-	0.062f	-	
PS_{off}	0	0	-	-	0.113f	0.038f	-

Source: Author.

As previously mentioned, Depth-Based Dynamic Search Window Resizing (DSWR – Section 8.1.3.4) and Texture-Based Motion Vector Inheritance (TMVI – Section 8.1.3.3) are proposed to allow dynamic power management through DVS of idle memory regions taking advantage of the inter-channel correlations. DSWR dynamically reduces the search-window during texture ME/DE processing based on the depth ME/DE encoding decisions for the Dependent Views, disabling texture-memory sectors based on the depth-block motion vector and sample average. TMVI prioritizes the use of ME/DE computational effort over the sharp-edge regions of depth maps. This way, for homogeneous regions, TMVI inherits the motion vector used for the collocated block in texture channel, disabling the depth-memory sectors.

8.1.5 Results

In order to make the discussion of results clearer, this section was divided into four subsections. Firstly, the run-time adaptive memory hierarchy results are presented, followed by the processing-unit results, system results, and the comparison with related works.

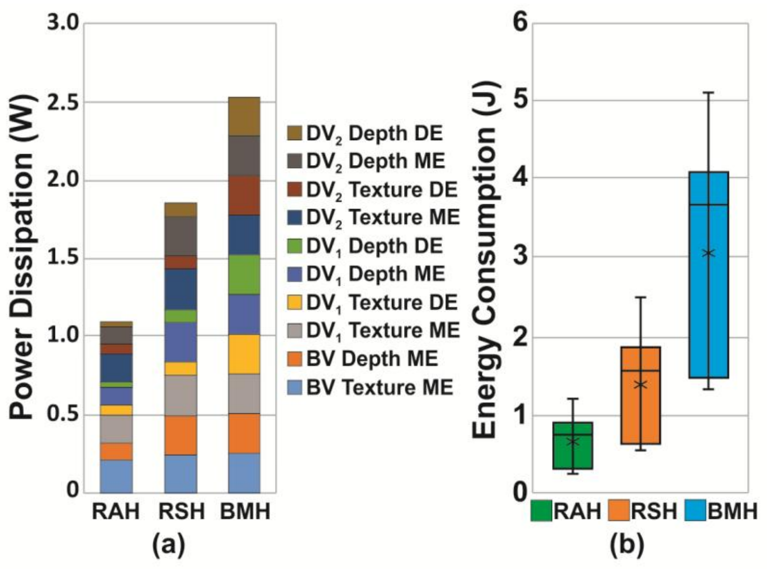
8.1.5.1 Adaptive Memory Results

To perform a fair energy consumption analysis for the on-chip memories, a memory simulator capable of emulating their behavior was developed (in C#) (VITECH, 2019), allowing the collection of the following numbers for each memory sector: cycles on, cycles sleep and toggles. Additionally, the total number of readings and writings were collected for each memory line. The simulator input is a trace file indicating all memory requests extracted using the 3D-HTM reference software during the ME/DE processing. It is worth saying that two traces were generated: (i) using 3D-HTM considering only hardware-oriented simplifications (Section 8.1.1); (ii) considering all techniques proposed in this thesis (Section 8.1.2). The use of two different traces is needed since the proposed algorithms change the BMA behavior leading to different memory requests. The energy consumption results provided by the simulator were estimated considering the power model presented in (Section 8.1.4.2).

Three memory scenarios were considered in this work and were applied on top of the B-Encoder configuration. The Base Memory Hierarchy (BMH) consists in the proposed hierarchy without memory size reduction techniques, adaptive management algorithms or data reuse; the Reduced-Size Hierarchy (RSH) features the proposed memory hierarchy with size reduction but without adaptive algorithms; and the Run-Time Adaptive Hierarchy (RAH) comprises the proposed memory hierarchy with run-time adaptive management. For any of these memory scenarios, the impact in BD-Rate was obtained with the same experimental environment presented in Section 8.1.1. It is important to notice that BMH presents the same BD-Rate increase of the B-Encoder (see Section 8.1.1.5), RSH increases the BD-Rate by up to 0.59% when compared to the B-Encoder (see Table 8.7), and RAH increases the BD-Rate by 1.54% and 1.47% for synthesized views and video total, respectively (when compared to the B-Encoder). The complete BD-Rate results, according to the 3D-video sequences, can be observed in Appendix C.

Figure 8.11-a presents a bar chart of the power dissipation for each of the 10 ME/DE memories in the three mentioned scenarios (DV and BV stand for Dependent and Base Views, respectively). As expected, the BMH solution presented the highest power dissipation (2.53W). Furthermore, one may notice that the memory size reduction (RSH solution) leads to a power dissipation of 1.86W, which means a decrease of 26.5%. The RAH solution led to a power dissipation of 1.10W, i.e., a reduction of 56.5% when compared to BMH. Additionally, Figure 8.11-b presents the energy consumption distribution of each experiment considering all AUs. Note that, in addition to energy reduction, the Run-Time Adaptive Hierarchy leads to a more concentrated distribution, demonstrating the low energy consumption along different video sequences and QPs.

Figure 8.11 – On-chip memory results: (a) Power dissipation breakdown and (b) Energy consumption boxplot



Source: (AFONSO *et al.*, 2019).

It is worth mentioning that the contributions regarding on-chip memory hierarchy and management of this work consider SRAM memories. However, these contributions focused on the architectural and control levels, so the significant gains achieved by the RAH over RSH and BMH memory hierarchies are independent of the technology. Thus, the proposed solution can be applied to on-chip memory technologies other than SRAM.

8.1.5.2 Processing Units Synthesis Results

The developed hardware was synthesized targeting an ASIC technology according to the experimental setup presented in Appendix B. Table 8.10 presents the synthesis results obtained for each design (ME/DE and texture/depth). It is important to highlight that the results for each ME/DE architecture consider only one instance of the processing units. In other words, each result is related to the ME or DE processing of one channel, texture or depth, and the number of instances needed in the system varies according to the encoding tool and the channel, as previously discussed in Section 8.1.2.1. As expected, both texture architectures consumed a more on-chip area compared to depth ones due to the internal bit depth. The full hardware design that considers all instances of the ME/DE architectures has 51.2mm² of on-chip area (40.9M Gates, considering the NAND-2 gate size) while presenting a total dissipation of 6.45W@100MHz.

Table 8.10 – Hardware design synthesis results

Architecture	Area (μm^2)	#Gates ($\times 10^6$)	Power (W)			Instances
			Leak.	Dyn.	Leak. + Dyn.	
ME – Texture	8,542,735	6.817	0.147	0.984	1.131	3
ME – Depth Map	2,797,576	2.232	0.047	0.422	0.469	3
DE – Texture	6,388,479	5.098	0.123	0.509	0.631	2
DE – Depth Map	2,220,003	1.772	0.041	0.152	0.193	2
SubTotal	**19,948,793**	**15.919**	**0.358**	**2.067**	**2.424**	**-**
****Total***	**51,237,897**	**40.888**	**0.908**	**5.539**	**6.447**	**-**

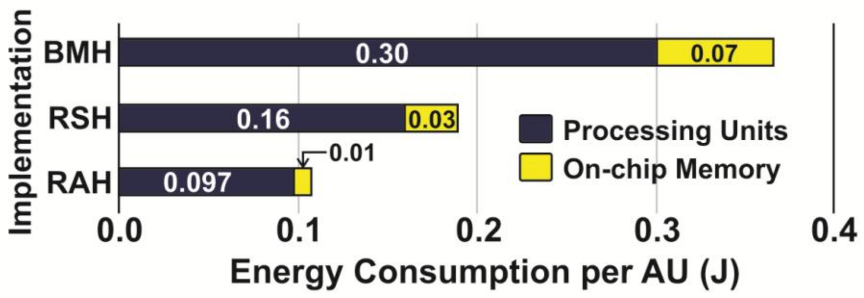
* The total results consider all instances of ME/DE architectures.

Source: Author.

8.1.5.3 System Results

Figure 8.12 presents a comparison regarding energy consumption per AU between the three scenarios considered in this work: (i) BMH, (ii) RSH, and (iii) RAH. Note that the BMH implementation presented a mean energy consumption of 0.37J per AU, whereas RSH consumes 0.19J (reduction of 48.6%). Finally, the developed Energy-Aware Motion and Disparity Estimation system, considering all proposed techniques, presents average consumption of only 0.107J per AU, thus reducing the total energy consumption by 71.1% in comparison to the BMH implementation, with a BD-Rate increase of only 1.54%.

Figure 8.12 – Comparison among Base Memory Hierarchy, Reduced-Size Hierarchy, and Run-Time Adaptive Hierarchy energy consumption per AU



Source: (AFONSO *et al.*, 2019).

Current commercial mobile devices include on-chip video codecs in addition to other modules responsible for specific functionalities or general purpose. However, manufacturers do not release detailed results of their hardware accelerators. Furthermore, works that present a power analysis for the different components implemented inside a device are rarely found in the literature, mainly considering current devices and current video-coding standards. These facts avoid that a comprehensive analysis regarding power savings by the proposed methodology in comparison with the overall power dissipated by a mobile device is made.

The work (CARROLL *et al.*, 2013) presents a power analysis of the Samsung Galaxy S III smartphone (launched in 2012) (SAMSUNG, 2019), in which the power consumption is measured by instrumentation at circuit level for the major components. Also, this work analyzes power consumption while decoding two 1280x720 (HD 720p) H.264 videos. For each video, the authors measured the consumption of the hardware video decoder and the software decoder. For a high-quality HD 720p video, more than 1W is saved by using the dedicated video compression hardware, reducing about 45% of the overall power dissipation. It is important to notice that a recording process (considering the video encoding instead of the decoding) demands higher power dissipation and up to 98% of the total energy consumption is related to the ME/DE steps (ZATT *et al.*, 2011b) considering off-chip memory access, on-chip memory and computation under the MVC standard. In the context of 3D-HEVC, it is also expected that the major part of the energy consumption is due to the ME/DE steps. Therefore, as the main proposed strategy reduces the energy consumption by 71.1% compared to BMH (related to the on-chip memory and computation), it is possible to infer that the power dissipation saved is relevant when current mobile devices are considered.

8.1.6 Main Related Works and Comparisons

As mentioned before, there are no related works proposing dedicated systems for 3D-HEVC ME/DE found in the literature. Even though a direct comparison is not possible, this section compares the proposed system with systems designed for previous standards. The on-chip memory solution proposed in (ZATT *et al.*, 2011b) can achieve up to 65% of energy consumption in comparison to Level-C. The Reduced-Size Hierarchy solution presents an average energy-consumption reduction of 79% when compared to Base Memory Hierarchy (which basically employs the Level-C). The Multi-view Video Encoder proposed in (DING *et al.*, 2010) dissipates 366mW@166MHz when processing two-view 1920x1080 3D videos. Although presenting less power dissipation when compared to the developed work, note that the solution proposed in (DING *et al.*, 2010) was built for a previous and less complex standard (MVC) where only two views (without depth maps) are considered, while the system developed in this work allows the processing of three texture frames and their respective depth maps. In other words, the system developed in this work enables the viewing of multiple views at the decoder side after view synthesis. Finally, the solution in (SAMPAIO *et al.*, 2013) reduces the energy consumption by 77%-88% when compared to the Level-C approach (considering the MVC standard), whereas the developed system reaches a reduction of 79% for 3D-HEVC compared to Level-C.

In conclusion, the developed 3D-HEVC Motion and Disparity Estimation system was designed for low-energy consumption featuring a dedicated memory hierarchy capable of run-time adaptation. Through several proposed algorithms on-chip memory reduction and dynamic power management were allowed. Three different memory hierarchy approaches were developed aiming at reducing energy consumption. Experimental results demonstrated an average energy consumption reduction of 79% for the memory when compared to the widely used Level-C scheme. By using the run-time adaptive management combined with the memory-size reduction techniques, the total ME/DE energy consumption is reduced by 71.1%. The developed work is the only solution able to process ME/DE for 3D-HEVC supporting up to three HD 1080p views (three texture views and their associated depth maps).

8.2 Low-power and Coding-Efficient Disparity Estimation Architecture based on Improved Unidirectional Disparity-Search Algorithm

Few works proposing hardware solutions for the DE step can be found in the literature and none of them was focusing on the 3D-HEVC. The works (FAN *et al.*, 2018) and (PERLEBERG *et al.* 2018) successfully proposed hardware solutions for the ME step of the HEVC standard considering modified versions of the Test Zone Search (TZS) algorithm (LI *et al.*, 2014b), which are compatible to the 3D-HEVC DE step. But solutions able to jointly propose hardware-friendly algorithms and efficient hardware architectures are still required to allow real-time processing of all 24 PU sizes defined by the 3D-HEVC. In this work, an intensive effort is done in order to overcome these gaps and a low-power and coding-efficient Disparity Estimation architecture based on a novel algorithm is proposed, as follows.

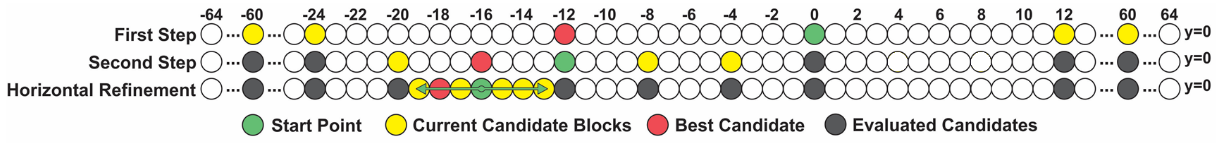
8.2.1 An Efficient Low-complexity BMA for the 3D-HEVC DE

As previously explained, the DE step is responsible for reducing the Inter-view redundancy, in a computationally intensive process. To constrain the DE computational effort, the BMA is applied within a pre-defined and reduced search area (SA), which size depends on the size of the CTU and the user-defined Search Range (SR). There are several BMAs in the literature designed focusing on the ME step which are also used in the DE step (FAN *et al.*, 2018) (PERLEBERG *et al.* 2018). The 3D-HEVC Reference Software, the 3D-HTM (3D-HEVC Reference Software, 2019), adopts the TZS to perform both, ME and DE steps. TZS can maintain near-optimal performance whereas reducing 23x the computational effort when compared to the Full Search (FS) algorithm (LI *et al.*, 2014b). However, once it was designed considering ME characteristics, TZS processes candidates in both horizontal and vertical positions related to the TZS start point. This way, TZS is unable to take advantage of DE specificities because the candidates at vertical positions have a low probability of being selected to represent the disparity. This way, BMAs focusing on the processing of candidates located in horizontal positions are more efficient to be used in the DE step since the 3D-HEVC was developed focusing on views captured with horizontal displacement (TECH *et al.*, 2016)(MÜLLER *et al*. 2014). The diamond-shaped search pattern employed by TZS leads to two drawbacks during DE process: (i) unnecessary computation and increased search window; (ii) potential local minima trapping when the TZS converges in the first iterations. Therefore, this thesis proposes an efficient low-complexity BMA for the 3D-HEVC DE, the Unidirectional Disparity-Search (UDS) algorithm. The developed algorithm is composed of three steps, and it considers a multi-view approach with a horizontal camera arrangement, as presented in the following. However, it is important to highlight that the algorithm can be easily adapted for a possible vertical camera arrangement.

8.2.1.1 UDS: Unidirectional Disparity Search Algorithm

The first UDS step, called the First Step, consists of a horizontal search from the start point (collocated block in the reference frame), which compares candidate blocks with a block sub-sampling defined by the S_1 parameter. The value of S_1 was chosen based on a wide offline evaluation using the 3D-HTM reference software as will be discussed in the following. Figure 8.13 presents an example with S_1=12, i.e., a block sub-sampling of 12:1, where the green circle represents the start point, the yellow circles represent the candidate blocks, and the red circle represents the best candidate. The First Step considers all available horizontal candidate blocks inside the search-window, concerning the S_1 block sub-sampling criteria.

Figure 8.13 – Steps of the proposed UDS algorithm



Source: (AFONSO *et al.*, 2018).

Similarly, the Second Step also performs a horizontal search, but with a lower sub-sampling rate, defined by S_2. This sub-sampling factor was also defined based on evaluations with the 3D-HTM. S_2=4 is used as an example in Figure 8.13 . The Second Step is centered at the best block matching of the First Step, and it is performed until reaching the search window limit or the limits given by the two neighboring candidate blocks evaluated in the previous step. The gray circles represent the candidate blocks evaluated along previous steps of the UDS algorithm. Finally, the third step consists of a Horizontal Refinement centered at the best block matching of the Second Step, and it considers all candidate blocks between the two neighboring evaluated candidates from the Second Step, as illustrated in Figure 8.13.

The UDS algorithm evaluates a constant number of candidate blocks during the disparity search, simplifying the memory-access pattern and the processing-unit design. The number of candidate blocks evaluated by the UDS algorithm can be calculated using the equations (23)–(26). The First step is performed with the number of candidate blocks denoted by $\#CB_{S1}$ in equation (23). The Search Range (SR) can be modified, however, SR=64 was used, as defined in the 3D-HTM default configuration. This means that candidate blocks located 64 samples to the left and 63 samples to the right of the collocated block can be compared in a search using the UDS algorithm. In equation (24), $\#CB_{S2}$ represents the candidate blocks evaluated in the Second Step, whereas in equation (25) $\#CB_{S3}$ denotes the candidate blocks compared in the Horizontal Refinement step. Equation (26) is used to calculate the total number of candidate blocks.

$$\#CB_{S1} = \left\lfloor \frac{2 \times SR}{S1} \right\rfloor + 1 \tag{23}$$

$$\#CB_{S2} = 2 \times \left(\left\lfloor \frac{S1}{S2} \right\rfloor - 1 \right) \tag{24}$$

$$\#CB_{S3} = 2 \times (S2 - 1) \tag{25}$$

$$\#CB_{Total} = CB_{S1} + CB_{S2} + CB_{S3} \tag{26}$$

In order to evaluate the UDS algorithm and to define the S_1 and S_2 values, some experiments were performed based on the CTC document recommended by JCT-3V. The experiments were conducted using the 3D-HTM reference software (3D-HEVC Reference Software, 2019) and the RA temporal configuration. The eight 3D video sequences and the three views (texture + depth map) of these video sequences were used along with the four recommended QP sets defined in the CTC document (see Appendix A). The SR remains the same defined in the 3D-HTM default configuration (64 samples).

The evaluations performed to assess the UDS performance considered the following different values of S_1 and S_2 parameters: S_1={4,8,12,16,20,24,28,32} and S_2={2,4,8,12,16}. These evaluations were done without any modification in the 3D-HEVC quadtree structure, i.e., DE prediction can adopt any of all 24 possible PU sizes. The only change was related to the disparity estimation, which uses UDS instead TZS for both texture and depth-map DE. This evaluation was anchored in two important variables: (i) the impact in coding efficiency, measured using the BD-Rate metric, and (ii) the number of candidate blocks, which strongly impacts in throughput, power and memory traffic.

Twenty-one test cases were applied to the UDS algorithm using a notation (S_1-S_2). The BD-Rate increase was evaluated considering the average values for all videos. The lowest impact in coding efficiency was observed for case 4-2, with a BD-Rate increase of 0.4037%. However, case 4-2 is also among those cases that evaluate more candidate blocks (37). In turn, the 12-4 case is one of those cases that evaluate fewer candidate blocks (21), and it presents the lowest impact regarding compression efficiency among those cases that evaluate 21 candidate blocks, with a BD-Rate increase of 1.7288%. Based on the results, these two test cases were adopted as the operation points in the UDS algorithm: Operation Point (4-2), Operation Point (12-4).

Table 8.11 summarizes the number of candidate blocks evaluated (based on equations (23)–(26)) for each step of the UDS algorithm according to these two operation points.

Table 8.11 – Number of candidate blocks evaluated by the UDS algorithm according to the operation point

UDS steps	**Number of candidate blocks according to the operation point (S_1-S_2)**	
	4-2	**12-4**
First Step	33	11
Second Step	2	4
Horizontal Refinement	2	6
Total	37	21

Source: Author.

8.2.1.2 iUDS: Improved Unidirectional Disparity Search Algorithm

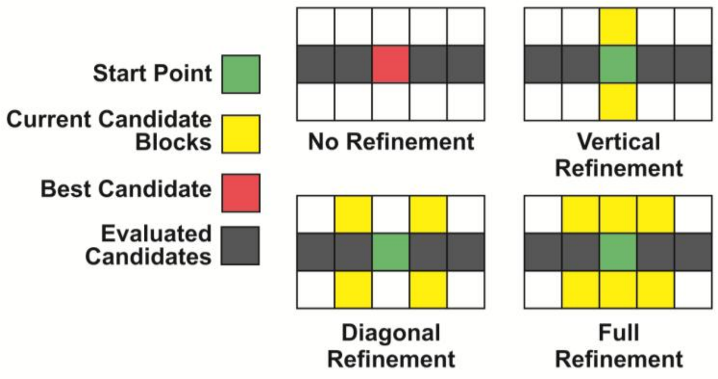
The UDS was developed to perform three steps of subsampled FS only in the horizontal direction. The three steps are similar, however, each step has a different subsampling parameter, which represents the distance that each candidate block will be apart from the neighboring candidate blocks. To get advantages of some vertical displacements exploration, which are not supported by UDS, the HDS was proposed during the development of this doctorate (see Section 8.1.3.2) in which performs a vertical refinement to the UDS heuristic. For that, the third step from UDS has been changed, and this step also evaluates the six vertical candidates around the best horizontal candidate results from the second step. Candidates at vertical positions have a low probability to be selected to represent the disparity, however, to cover little vertical displacements it is important to assure good matching under slight cameras vibrations or captured noise.

The HDS algorithm proposed in this doctorate implements a vertical refinement, but both horizontal distances that each candidate block will be apart from the neighboring candidate blocks and the refinement vertical are proposed empirically. The iUDS algorithm brings a proper analysis to verify the impact of this approach in video coding. Therefore, to propose the iUDS algorithm (Improved UDS), three vertical refinement alternatives along with the UDS original operation points were evaluated.

All experiments to evaluate the iUDS were performed using the 3D-HTM in version 16.2 (3D-HEVC Reference Software, 2019), considering the 3D-HEVC CTC document. The results presented in this Section consider the average of all videos and QPs defined in the CTC (see Appendix A) running under the RA temporal configuration. Furthermore, it is worth mentioning that between 200 and 300 AUs were encoded from each video sequence. The SR remains the same defined in the 3D-HTM default configuration (64 samples).

The iUDS vertical refinement is treated as a fourth UDS step, where the algorithm can evaluate two, four or six additional neighboring blocks in the vertical direction (three above and three below the current best block), as presented in Figure 8.14. The vertical-refinement patterns are called here: (i) Vertical; (ii) Diagonal; (iii) Full Refinement (Vertical + Diagonal). Vertical pattern selects the two blocks located directly above and below the best matching block of the previous steps for comparison. Diagonal pattern selects for comparison the four blocks located diagonally from the best matching block. Full-refinement pattern selects both the Vertical and Diagonal blocks, totalizing six vertical neighboring blocks of the best matching block chose in the previous steps. The definition of which mode will be used is made by the N_4 parameter. After this step, the iUDS completes its execution, and the result is the block most similar to the block being encoded among the evaluated blocks.

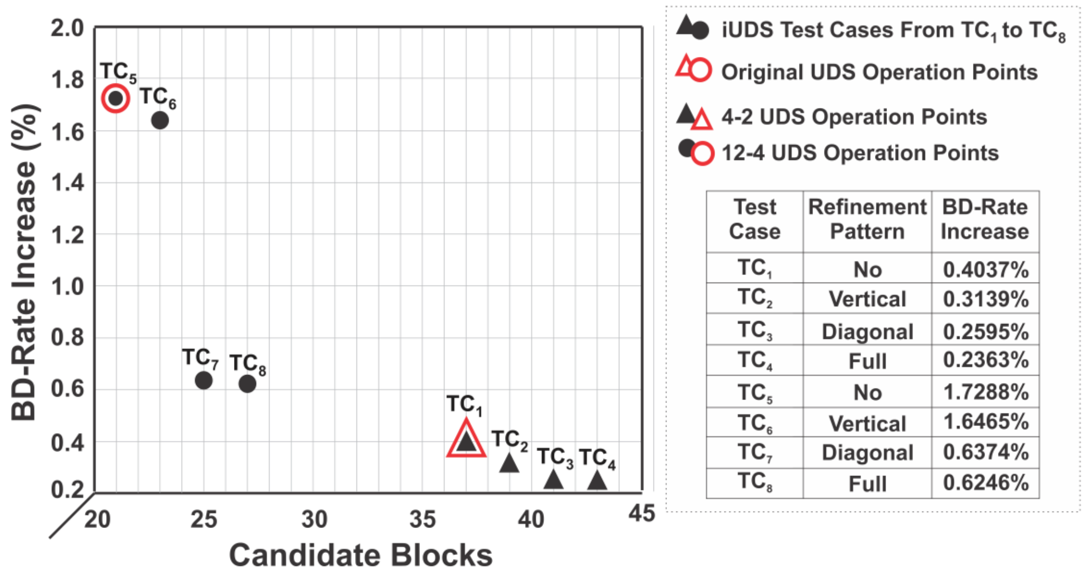
Figure 8.14 – iUDS vertical refinement analysis: Refinement patters



Source: Author.

Based on the proposed vertical-refinement patterns, a set of evaluations were performed to assess the iUDS performance when the iUDS replaces the TZS for both texture and depth-map DE. This evaluation was anchored in two important variables: (i) the impact in coding efficiency, measured using the BD-Rate metric, and (ii) the number of candidate blocks, which strongly impacts in throughput, power, and memory access. Figure 8.15 presents a scatter graph containing eight test cases (TC) applied to the iUDS algorithm (black shapes). Red shapes denote the original UDS operation points, i.e., iUDS with no refinement. The BD-Rate increase is presented considering the average values for all videos of the experimental setup. The lowest impact in coding efficiency was observed for TC_4 operating with Full Refinement, with a BD-Rate increase of 0.2363%. However, TC_4 is also the case that evaluates more candidate blocks (43). In turn, TC_7 with Diagonal Refinement presents a better balance between evaluated candidate blocks (25) and BD-Rate increase (0.6374%). The complete BD-Rate results, according to the 3D-video sequences, can be observed in Appendix C.

Figure 8.15 – iUDS Vertical Refinement Analysis: Scatter Graph Containing Eight iUDS Test Cases



Test Case	Refinement Pattern	BD-Rate Increase
TC_1	No	0.4037%
TC_2	Vertical	0.3139%
TC_3	Diagonal	0.2595%
TC_4	Full	0.2363%
TC_5	No	1.7288%
TC_6	Vertical	1.6465%
TC_7	Diagonal	0.6374%
TC_8	Full	0.6246%

Source: Author.

Comparing the iUDS approach with the TZS algorithm, iUDS has a low impact on the coding efficiency and delivers an expressive reduction in the number of evaluated blocks. In (AFONSO *et al*., 2019), the number of candidate blocks evaluated by the TZS was analyzed. TZS evaluates 55 candidate blocks on average (standard deviation equal to 149) in the DE process, ranging from 10 up to 1,484 in the performed experiments. As one can notice in Figure 8.15), iUDS algorithm evaluates between 21 and 43 candidate blocks for the eight

TCs, and it causes a reduction between 21.8% and 61.8% in the number of evaluated blocks, considering the TZS average results, whereas presenting a predictable behavior in terms of computational effort.

8.2.2 iUDS 3D-HEVC DE Architecture

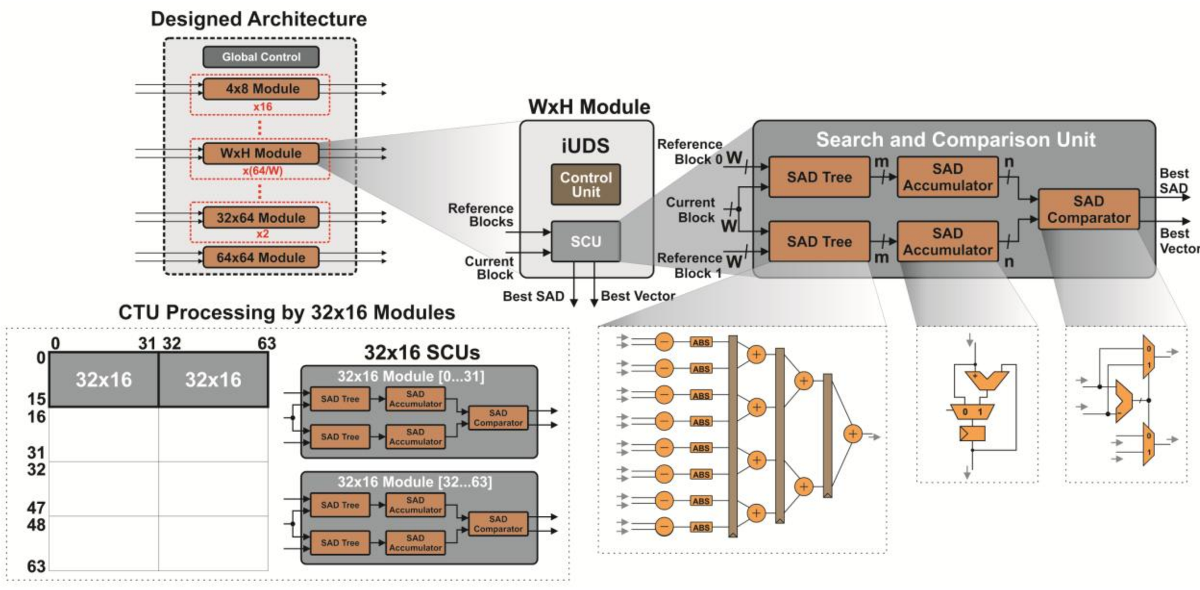
The architecture designed in this work for the iUDS algorithm is presented in Figure 8.16, and it supports the eight test cases presented in Section 8.2.1.2. Also, it can process the 24 possible PU sizes with 24 specialized datapaths for each PU size (red dotted lines in Figure 8.16). All datapaths work in parallel ensuring the processing of high-resolution 3D videos. Each specialized datapath has a variable number of internal modules, and each module is an instance of the iUDS, developed to efficiently deal with only one PU size. The number of modules to process each PU size is based on the PU width, where each datapath has the number of modules needed to parallelize all the PUs of a CTU with 64x64 samples, as given by equation (27). So, the number of iterations for each module completes the processing of a CTU depends on the PU height, as given by equation (28).

$$Modules_{Datapath} = \frac{CTU_{Width}}{Width_{Datapath}} \quad (27)$$

$$Iterations_{Module} = \frac{CTU_{Height}}{Height_{Datapath}} \quad (28)$$

As presented in Figure 8.16, each module of this architecture can be divided into two units. The first unit is the control unit, which selects the candidate blocks using any subsampling parameter. The control unit also compares the results provided by different steps of iUDS. The second unit is the Search and Comparison Unit (SCU), which compares the two candidate blocks selected by the control unit with the current block following the SAD similarity criterion. The SAD was adopted in this work as the similarity criterion because it can be easily implemented in hardware, and it is the similarity criterion most commonly used in hardware implementations (FAN *et al.*, 2018) (PERLEBERG *et al.* 2018). Also, it is the similarity criterion adopted in the 3D-HTM to deal with the DE step, by default.

Figure 8.16 – iUDS developed architecture



Source: Author.

8.2.2.1 Search and Comparison Unit

The block diagram of one SCU is also presented in Figure 8.16. Targeting the best tradeoff between processing rate and the idle time of the circuit, each SCU processes, in parallel, two candidate blocks so that one line of each candidate block is processed at each clock cycle, independently of the block size. For that, each SCU was composed of two SAD Trees, two Accumulators and one Comparator of two inputs, as presented in Figure 8.16.

Each SAD Tree is responsible for calculating the SAD value between one line of one reference block and one line of the current block. Figure 8.16 exemplifies the SAD Tree that deals with PUs with 8-sample width. One can notice that each SAD Tree receives the samples of each line of these candidate blocks and calculates the difference between them and the samples of the current block. After that, the absolute value of that difference is obtained. Finally, these absolute differences between the samples are added two by two, so that each SAD Tree results in only one SAD value. These sums are separated into three pipeline stages. Since the SAD Trees of other PU sizes will have a different number of inputs, they will also require a different number of pipeline stages to maintain the desired throughput.

After the SAD Trees compute the SAD value related to one line, the processed SAD values are accumulated by two SAD Accumulators, while the SAD Trees compute the rest of the block lines. After the SAD Trees compute all lines of the candidate blocks, the output value in each Accumulator is exactly the SAD value between each candidate block and the current block. Figure 8.16 shows the architecture of one SAD Accumulator. The Control signal is responsible for resetting the value accumulated by the SAD Accumulator when the SAD Trees start the processing of a new block. The value of **m** (input sample bit depth) depends on the processed PU width, while the value of **n** (output bit depth) depends on the width and height of the processed PU. The values of **m** and **n** are given by equations (29) and (30), respectively.

$$m = 8 + log_2(PU_{WIDTH}) \tag{29}$$

$$n = m + log_2(PU_{HEIGHT}) \tag{30}$$

Finally, when the SAD Trees finish the blocks processing, the Comparator stage processes the SAD values along with the disparity vectors of each block and delivers the smallest SAD value and the related motion vector. The architecture of a SAD Comparator is presented in Figure 8.16. The most significant bit (MSB) of the difference between the two SAD values is used as the selector in two multiplexers that output the smallest SAD and its disparity vector.

8.2.2.2 Control Unit

The control unit selects the candidates to be processed by the SCU. The selection is made from left to right, starting with the candidates on the left side of the SA, and finishing by the candidates on the right side of the SA.

The number of candidate blocks that the first three iUDS steps need to select for comparison is dependent on the subsampling parameters of the adopted operation point, as it can be obtained by equation (31), where **N#** is the subsampling parameter from the iUDS step. However, the number of candidates from the vertical refinement are 2, 4, or 6 for the Vertical, Diagonal, or Full Refinement, respectively. The control unit selects the candidate blocks two by two, from the left of the SA to the right of the SA, and send the selected blocks to the SCU. All results from SCU will be compared by the control unit. So, when the search covers the SA, the search can pass to the processing of the next step candidates.

$$h(N_{\#}) = {(SA_{Width_{\#}} - CTU_{Width} + N_{\#})}/{N_{\#}} \tag{31}$$

The control unit is also responsible for managing the memory communication. The memory communication affects the processing of each CTU, which will need a few more clock cycles to be processed. Two additional cycles are required at the end of the processing of each PU, and one additional cycle is required when ending the process of each CTU.

8.2.2.3 Temporal Analysis

The number of clock cycles that a module of a specific datapath needs to process one specific iUDS step is given by equation (32), being **h** the number of candidates from the respective step. The equation considers the filling of the pipeline stages in the first iteration of each iUDS step, the processing rate of one block line per clock cycle, and also the two clock cycles due to the memory communication. When considering the four iUDS steps, the number of clock cycles to process each block can be obtained by the equation (33).

$$p(N_{\#}) = log_2\left(PU_{Width}\right) + \frac{h(N_{\#})}{2}\left(PU_{Height}\right) + 1 + 1 \tag{32}$$

$$r(N_1, N_2, N_3, N_4) = p(N_1) + p(N_2) + p(N_3) + p(N_4) \tag{33}$$

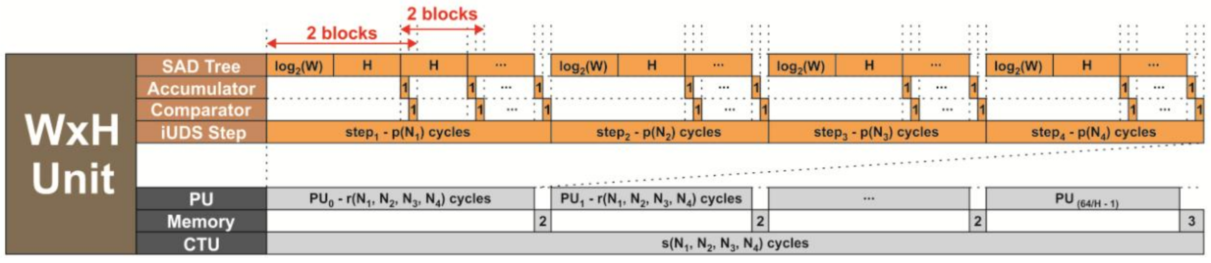
As previously mentioned, some modules were replicated targeting a minimum idle time to each module, instead of using only one module per PU size to process the CTU. This choice resulted in a balanced number of clock cycles, as it can be confirmed using the equation (34), that represents the total clock cycles that each module needs to fully complete the processing of a CTU.

$$s(N_1, N_2, N_3, N_4) = \left(\frac{CTU_{Height}}{PU_{Height}}\right)\left(r(N_1, N_2, N_3, N_4) + 2\right) + 1 \tag{34}$$

As one can notice from equation (34), the modules which process the 8x4 datapath are the modules that need more clock cycles to process a CTU, considering any iUDS operation point among the eight test cases evaluated in Section 8.2.1.2. So, the delay of this module is used to determine the throughput of the developed hardware. The slower modules that have finished the processing of the CTU were disabled using the clock-gating technique, reducing energy consumption.

The clock cycle diagram of one WxH module is presented in Figure 8.17. As one can notice, the processing of each PU depends on the block size and the subsampling parameters. Similarly, the number of iterations from a module to process a given PU depends on the block Height.

Figure 8.17 – Temporal analysis of the iUDS developed architecture



Source: Author.

8.2.3 Synthesis Results and Comparisons

The developed hardware was synthesized targeting an ASIC technology according to the experimental setup presented in Appendix B. The results are presented in Table 8.12, along with hardware results of the main related works. Although the two related works target the ME step, some values related to the hardware results (such as the power dissipation and area usage) can be compared, since both ME and DE are similar encoding tools. Other comparisons, as the compression efficiency obtained with the BMA developed by (FAN *et al.*, 2018) and (PERLEBERG *et al.*, 2018) and BMA proposed in this thesis, cannot be made since they were evaluated in a different encoding context, using different encoders, with different video sequences. Furthermore, as our hardware uses the MVD format which requires processing of both texture and depth maps, the throughput of related works was divided by two.

As presented in Table 8.12, iUDS architecture needs more resources than (FAN *et al.*, 2018) and less than (PERLEBERG *et al.*, 2018). Except (FAN *et al.*, 2018), that used 65nm standard cells, all other works used 45nm. However, the results show that our design reaches power dissipation ranging from 29% to 62% smaller than (FAN *et al.*, 2018), and from 35% to 66% smaller than (PERLEBERG *et al.*, 2018). The power results related to the developed architecture considered the Test Cases TC_4 and TC_5, which represent the best operation points considering coding efficiency and computationally effort, respectively. Besides, the designed hardware reaches throughput to process UHD 2160p 3D videos at 60fps (3.4 views) while these two related works did not reach this throughput.

Table 8.12 – Hardware design synthesis results

Related Works		**Fan (2018)**	**Perleberg (2018)**	**This Thesis**	
				TC_5	**TC_4**
Target Encoding Tool		HEVC ME	HEVC ME	3D-HEVC DE	
Block Sizes		All 24 possible	Four	All 24 possible	
BMA Algorithm		Modified TZS	Modified TZS	iUDS	
ASIC Technology		65nm	Nangate 45nm	Nangate 45nm	
Total Area (gates)		489.4k	18,103k	3,243k	
SRAM		18.4kB	no	16.77kB*	
1080p@60fps 1 view	**Frequency**	500MHz	144.92MHz	58.8MHz	105.8 MHz
	Power/ Voltage	128.5mW/ N.A.	140.84mW/ N.A.	47.82mW/ 0.95V	90.72mW/ 0.95V
3D-video processing at UHD 2160p@60fps		No (0.25 views)	No (1.76 views)	Yes (3.4 views)	

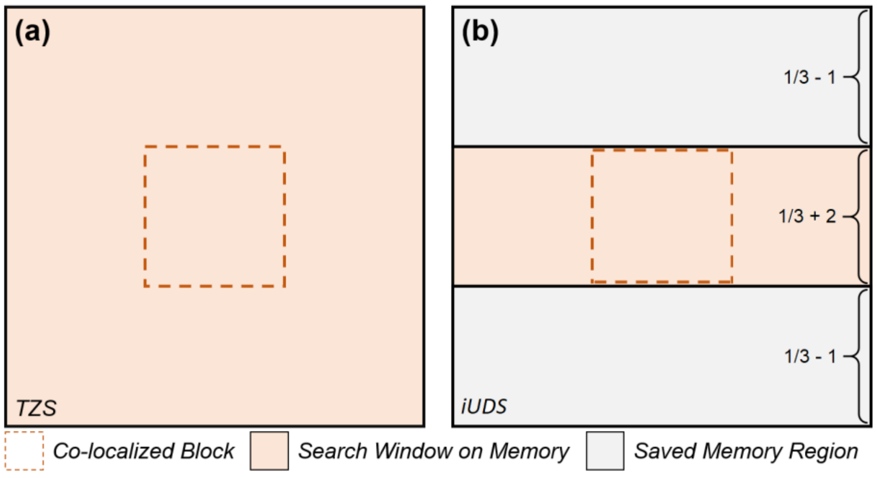
*estimated

Source: Author.

The replacement of the TZS by the iUDS can bring better hardware results when compared to TZS related works. Also, this replacement affects the on-chip memory characteristics. The iUDS needs to process candidate blocks from only three lines considering a CTU, reducing the samples stored in the on-chip memory. Considering the CTU size of 64x64 samples, and an SR of 64 samples, the iUDS needs an SA of 192×66 samples, which represents a memory of 16.77kB. This value represents an on-chip memory reduction of 65.62% since the storage of all TZS candidates requires a memory of 192×192 samples. Figure 8.18 helps to understand this reduction, where a search area (a.k.a. Search Window – SW) with the size of 192×192 samples (necessary to employ 64×64 CTU-level data reuse with search range of 64 samples) considering the TZS algorithm is reduced to an SW of 192×66 samples when using the iUDS algorithm. Additionally, after processing the search algorithm for a given block, the on-chip memory must be refreshed with the SW of the new block to be predicted. Considering a search window-based data reuse strategy, such as Level-C (CHEN et al., 2006), this fact would increases in three the total number of memory accesses for the TZS algorithm, and in one additional access considering the iUDS algorithm in anyone operation point.

The on-chip memory reduction is important to allow a hardware area reduction, but mainly to allow lower energy consumption, as discussed in the following.

Figure 8.18 – On-chip memory required to store the search window for one CTU: (a) Using TZS algorithm; (b) Using the iUDS algorithm.



Source: Author.

An estimation of the impact in terms of energy consumption on memory-related issues when the iUDS algorithm replaces the TZS algorithm on the 3D-HEVC disparity-estimation prediction is presented in Table 8.13 presents a comparison between the test case TC4 (4-2

UDS with full vertical refinement), test case TC5 (12-4 UDS without refinement), and TZS algorithms regarding energy consumption of on-chip memory estimated by looking at the characteristics of these algorithms, such as the number of candidate blocks and search range. Note that these results were normalized concerning TZS algorithm since it evaluates more candidate blocks among the algorithms presented in Table 8.13.

One may notice in Figure 8.18 that TZS requires an on-chip memory almost three times larger than that of iUDS to store the SW. This fact indicates that the static energy consumption tends to be almost three times greater when employing TZS. Actually, the iUDS leakage energy consumption is 65.62% lower than in TZS according to the energy estimation by considering an iUDS operation with full vertical refinement.

Based on these values, the normalized dynamic energy consumption of reading/writing operations was estimated for TZS (1), TC4 iUDS (0.759), and TC5 iUDS (0.379). Note that iUDS propitiates a reduction of dynamic energy from 24.1% up to 62.1% in comparison with TZS. It is worth mentioning that TZS has a dynamic number of evaluated candidates, that depends on the video content and the prediction tool evaluated. Considering the DE context, the TZS evaluates 55 candidates (AFONSO *et al.*, 2019) on average. However, the iUDS evaluates from 21 to 43 candidates, which represents a reduction from 21.81% to 61.81% in the number of evaluated candidates. Considering 130 candidates for this search algorithm (average + 0.5 standard deviation), the dynamic energy consumption would be 6.04 times higher than that consumed by TC5 iUDS. Moreover, considering the worst TZS case (1,484 candidates), the memory energy consumption would be 67.59 times higher than that consumed by TC5 iUDS in this scenario.

The relationship between dynamic (reading and writing) and static (leakage) energy consumption varies according to the technology. This fact hampers fair estimative regarding total energy consumption. Still, one may notice that the hardware implementation of the iUDS algorithm would strongly reduce the energy consumption and size of on-chip memory to store the disparity SW. Since TZS requires the evaluation of more candidate blocks than iUDS, it demands higher parallelism considering a hardware implementation to maintain the same architecture throughput. Note that increasing the architecture parallelism will lead to greater energy consumption by DE processing units.

Table 8.13 – Hardware design synthesis results

Algorithm	**#Cand. Blocks**	**Estimated Normalized Energy Consumption**		
		Leakage	***Dynamic***	***SAD Operations***
TZS	55	1	1	1
TC4 iUDS (4-2 UDS with full vertical refinement)	37	0.344	0.759	0.782
TC5 iUDS (12-4 UDS without refinement)	21	0.344	0.379	0.382

Source: Author.

Therefore, this dedicated hardware design for the 3D-HEVC Disparity Estimation developed in this work presented a high-throughput and low-power, using the proposed iUDS algorithm. A software analysis was performed to investigate the optimal operation points considering a vertical refinement step, allowing the selection of the operation points from an external control signal according to the desired trade-off between coding efficiency and power dissipation. The synthesis results showed that the developed hardware obtains higher throughput than the related works using the TZS algorithm. Also, by using the iUDS instead of TZS, the architecture reduces in 65.62% the required memory size. Finally, the designed architecture is the first dedicated hardware design for 3D-HEVC DE step that supports all 24 possible PU sizes capable of processing UHD2160p 3D-videos at 60fps.

9 CONCLUSIONS AND FUTURE WORKS

3D-HEVC enables the exhibition 3D-videos of multiple views at the same time with high and ultra-high resolutions at the cost of a high computational effort increase due to its novel encoding tools. Three-dimension video-capable embedded mobile devices based on the 3D-HEVC are expected pushed forward by the popularization of multimedia services and emerging video technologies. These three-dimension video-capable embedded mobile devices demand efficient energy/memory-management systems to deal with severe memory/processing requirements and limited energy supply. To meet these challenges, this thesis presented dedicated energy-efficient algorithms and architectures focusing on the most energy demanding 3D-HEVC encoding steps along with its MVD approach.

Four architectures/systems exploiting memory/processing aspects, and the characteristics of the encoding tools under an MVD approach were developed, two of them focused on the Intra-frame prediction tools and other two focused on the Inter prediction, which includes both Inter-frames and Inter-view predictions. All the decisions taken during the development of the work were marked in exhaustive simulations using the 3D-HEVC reference software where the following data were extracted: (i) the most time demanding encoding tools used for both texture and depth-map coding according to the 3D-HEVC to allow the understanding of its energy requirements; (ii) the most selected and representative 3D-HEVC encoding tools used during the coding process to identify the tools with more impact regarding compression and image quality; (iii) the impact regarding compression by constraining specific 3D-HEVC encoding tools and the block sizes supported by these encoding tools during the coding process.

All the developed architectures take advantage of application-specific knowledge of 3D-HEVC, i.e., its new coding tools and video content properties.

Also, on-/off-chip memory-related access behaviors related to 3D-HEVC encoding process were analyzed at run time to adapt the memory management and to save energy consumption for the ME/DE system developed in this doctorate. This system also takes advantage of a flexible coding order between texture and depth maps to propose heuristics and memory management capable of reducing the energy consumption of its video memory and processing architectures.

The low-power hardware design for the Depth Intra Skip coding tool reduces the computational effort based on a strategy that consisted of replacing the SVDC for the SAD as the similarity criterion. This strategy reduced the number of arithmetic operations related to the similarity criterion by over 71%, and it avoided a rendering process. This architecture was the first dedicated hardware design published in the literature for the DIS tool, and it is capable of processing five UHD 2160p views at 60 frames per second.

The low-power and memory-aware depth-map Intra-frame prediction system was developed based on hardware-oriented heuristics to reduce the computational effort. These strategies consisted of removing less important prediction modes and block sizes. Also, a specific complexity-reduction strategy applied to the DMM-1 along with a modified Bresenham implementation for representing the wedgelet at the time allowed the calculation of only six wedgelet patterns without any storage prediction. This architecture is the first supporting both the novel 3D-HEVC and the conventional HEVC intra prediction tools, and it is capable of processing HD 1080p nine views at 30 fps.

The Motion and Disparity Estimation system was designed for low-energy consumption, featuring a run-time adaptive memory hierarchy. The memory hierarchy featured window-based prefetching, data reuse, subsampling, and dynamic voltage scaling controlled by the developed Depth-Based Dynamic Search Window Resizing algorithm. The developed energy-aware Motion and Disparity Estimation system was the first work published in the literature that proposed a real-time ME/DE system for the 3D-HEVC standard with an adaptive memory hierarchy capable of processing three HD 1080p views at 30 frames per second. Memory results demonstrated an average on-chip energy reduction of 79% in comparison to the Level-C solution.

The low-power and coding-efficient disparity estimation architecture based on the developed Improved Unidirectional Disparity-Search algorithm (iUDS) prioritized the horizontal search instead of using conventional search algorithms in two dimensions to take advantage of the fact that horizontal disparities are more common. Results showed negligible impact over the encoding efficiency and enough throughput to encode five UHD 2160p views at 40 frames per second.

The algorithms and architectures proposed during this doctorate widely exploited 3D-HEVC memory/processing aspects, and the characteristics of the encoding tools under an MVD approach. One of the developed systems also had taken advantage of a flexible coding order between texture and depth maps to reduce the energy consumption, however, focusing only on the Motion and Disparity Estimations, i.e., only focusing on the Inter-frames and Inter-view predictions. As a future work, it is possible to extend this approach to the Intra-frame prediction, covering both the Intra tools inherited by the HEVC standard as the ones introduced by the 3D-HEVC. This way, the decisions used for the coding process of a specific channel can be used to avoid or simplify the coding process of the other channel. An example of this idea is related to the angular mode inherited by the HEVC and the DMM-1 mode introduced by the 3D-HEVC. Both encoding tools are based on the prediction of directional structures, which assumes that these modes can also take advantage of sharing their decisions.

REFERENCES

3D-HEVC Reference Software. Available in: <https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/tags/>. Retrieved in: Jan. 2019.

AFONSO, V.; CONCEIÇÃO, R.; SALDANHA, M.; BRAATZ, L.; PERLEBERG, M.; CORRÊA, G.; PORTO, M.; AGOSTINI, L.; ZATT, B.; SUSIN, A. Energy-Aware Motion and Disparity Estimation System for 3D-HEVC with Run-Time Adaptive Memory Hierarchy. **Transactions on circuits and systems for video technology (TCSVT)**, v. 29, n. 6, pp. 1878–1892, 2019.

AFONSO, V.; MAICH H.; AGOSTINI, L.; FRANCO, D. Low Cost and High Throughput FME Interpolation for the HEVC Emerging Video Coding Standard. In: IEEE 4th Latin American Symposium on Circuits and Systems, LASCAS, 2013. **Proceedings...** Cusco: Feb. 27 - Mar. 1, 2013.

AFONSO, V.; MAICH H.; AUDIBERT, L.; ZATT, B.; PORTO, M.; AGOSTINI, L.; SUSIN, A. Hardware Implementation for the HEVC Fractional Motion Estimation Targeting Real-Time and Low-Energy**. Journal of Integrated Circuits and Systems (JICS)**, v.11, n. 2, pp. 106-120, Apr. 2016.

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, ISCAS, 2017. **Proceedings...** Baltimore: IEEE, 2017.

AFONSO, V.; SUSIN, A.; PERLEBERG, M.; CONCEIÇÃO, R.; CORREA, G.; AGOSTINI, L.; ZATT, B.; PORTO, M. Hardware-Friendly Unidirectional Disparity-Search Algorithm for 3D-HEVC. In: 2018 IEEE International Symposium on Circuits and Systems, ISCAS, 2018. **Proceedings...** Florence: IEEE, 2018.

AGOSTINI, L. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas a Compressão de Vídeo Segundo o Padrão H.264/AVC**. 2007. 172f. Ph.D. Thesis (Doctorate in Computing Science) – Instituto de Informática, UFRGS, Porto Alegre.

AHMAD, W.; MARTINA, M.; MASERA, G. Complexity and Implementation Analysis of synthesized view distortion estimation architecture in 3D High Efficiency Video Coding. In: International Conference on 3D Imaging, IC3D, 2015. **Proceedings...** Liege: 2015.

AMISH, F.; BOURENNANE E. An efficient hardware solution for 3D-HEVC intra-prediction. **Journal of Real-Time Image Processing**, pp 1-13. 2017

BJONTEGAARD, G. **Calculation of average PSNR differences between RD curves (VCEG-M33)**, Austin, 2001.

BOSSEN, F. **Common test conditions and software reference configurations (JCTVC-L1100)**, Jan. 2013.

BRAATZ, L.; ZATT, B.; PALOMINO, D.; AGOSTINI, L.; PORTO, M. High-Throughput and Low-Power Integrated Direct/Inverse HEVC Quantization Hardware Design. In: International Symposium on Circuits and Systems, ISCAS, 2018. **Proceedings...** Florence: IEEE, 2018.

BRESENHAM, J. Algorithm for computer control of a digital plotter. **IBM Systems Journal**, [S.l.], v.4, n.1, p.25–30, 1965.

BROSS, B.; HAN, W.; OHM, J.; SULLIVAN, G.; WIEGAND, T. **High Efficiency Video Coding text specification draft 9 (JCTVC-K1003)**, Shangai, 2012.

CADENCE. **Encounter RTL Compiler.** Available in: <https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84441.html>. Retrieved in: Jan., 2019.

CARROLL A.; HEISER, G. The Systems Hacker's Guide to the Galaxy Energy Usage in a Modern Smartphone. In: 4th Asia-Pacific Workshop on Systems, APSys, 2013. **Proceedings...** Singapore: ACM, 2013.

CHEN, C.; HUANG, C.; CHEN, Y.; CHEN, L. Level C+ Data Reuse Scheme for Motion Estimation With Corresponding Coding Orders. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 16, n. 4, pp. 553-558, Apr. 2006.

CHEN, Y.; TECH, G.; WEGNER, K.; YEA, S. **Test Model 11 of 3D-HEVC and MV-HEVC (JCT3V-K1003)**, Geneva, 2015.

CHO, S.; KIM, H.; KIM, H. Y.; KIM, M. Efficient In-Loop Filtering Across Tile Boundaries for Multi-Core HEVC Hardware Decoders With 4 K/8 K- UHD Video Applications. **Transactions on Multimedia (TMM)**, v. 17, n. 6, pp. 778- 791, 2015.

CHOI, M.; CHANG, I.; KIM, J. High Performance and Hardware Efficient Multiview Video Coding Frame Scheduling Algorithms and Architectures. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 23, n. 8, pp. 1312-1321, Aug. 2013.

CISCO. **Visual Networking Index: Forecast and Methodology, 2016–2021**. Available in: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html> Retrieved in Jan., 2019.

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, ICIP, 2016. **Proceedings...** Phoenix: IEEE, 2016.

CORRÊA, G.; ASSUNÇÃO, P.; AGOSTINI, L.; CRUZ, L. A. S.. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. **IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, [S.l.], v. 22, n. 12, p. 1899-1909, Dec. 2012.

CORREA, M.; ZATT, B.; PORTO, M.; AGOSTINI, L. High-throughput HEVC intrapicture prediction hardware design targeting UHD 8K videos. In: International Symposium on Circuits and Systems, ISCAS, 2017. **Proceedings...** Baltimore: IEEE, 2017.

DHINGRA, P. **Real 3D or Fake 3D**. Available in: <http://realorfake3d.com/>. Retrieved in: Jan. 2019.

DIMENCO. Available in: <http://www.dimenco.eu>. Retrieved in: Jan. 2019.

DING, L.; CHEN, W.; TSUNG, P.; CHUANG, T.; HSIAO, P.; CHEN, Y.; CHIU, H.; CHIEN, S.; CHEN, L. A 212 MPixels/s 4096 2160p Multiview Video Encoder Chip for 3D/Quad Full HDTV Applications. **IEEE Journal of Solid-State Circuits**, v. 45, n. 1, pp. 46-58, Jan. 2010.

DOAN, N.; KIM, T.; RHEE, C.; LEE, H. A hardware-oriented concurrent TZ search algorithm for High-Efficiency Video Coding. **EURASIP Journal on Advances in Signal Processing**, v. 2017, n. 1, Nov. 2017.

FAN, Y.; HUANG, L.; HAO, B.; ZENG, X. A Hardware-Oriented IME Algorithm for HEVC and Its Hardware Implementation. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 28, n. 8, pp. 2048-2057, 2018.

GENG, J. Three-dimensional display technologies. **Advances in Optics and Photonics**, [S.l.], v. 5, n. 4, p. 456-535, Nov. 2013.

GHANBARI, M. **Standard Codecs:** Image Compression to Advanced Video Coding. United Kingdom: The Institution of Electrical Engineers, 2003.

GOEBEL, J.; PAIM, G.; AGOSTINI, L.; ZATT, B.; PORTO, M. An HEVC multi-size DCT hardware with constant throughput and supporting heterogenous CUs. In: International Symposium on Circuits and Systems, ISCAS, 2016. **Proceedings...** Montreal: IEEE, 2016.

GOLDMAN, M. S.; LITWIC, L.; BAUMANN, O. ULTRA-HD Content Acquisition and Exchange Using HEVC Range Extensions. **SMPTE Motion Imaging Journal**, v. 124, n. 3, p. 28–36, Apr. 2015.

GONZALEZ, R.; WOODS, R. **Processamento de Imagens Digitais.** São Paulo: Edgard Blücher, 2003.

GOOGLE GLASS TECHNOLOGY. Available in: <https://googleglassesbus237.weebly.com/ >. Retrieved in: Jan., 2019.

GOPALAKRISHNA, S.; HANNUKSELA, M.; GABBOUJ, M. Flexible Coding Order for 3D Video Extension of H.265/HEVC. In: IEEE Picture Coding Symposium, PCS, 2013. **Proceedings...** San Jose: IEEE, Dec. 8-11, 2013, pp. 253-256.

GRELLERT, M. **Computational Effort Analysis and Control in High Efficiency Video Coding.** 2014. 89f. Dissertação de Mestrado. Mestrado em Ciência da Computação – Instituto de Informática, UFRGS, Porto Alegre.

HE, G.; ZHOU, D.; LI, Y.; CHEN, Z.; ZHANG T.; GOTO, S. High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v. 23, n. 12, pp.3138-3142, Dec. 2015.

IKAI, T.; TSUKUBA, T. **Simplification of DMM table derivation (JCT3V-K0042)**, Geneva, 2015.

INTEL REALSENSE TECHNOLOGY. Available in: <https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html>. Retrieved in: Jan., 2019

ISO/IEC. **Call for Proposals on 3D Video Coding Technology (JTC1/SC29/WG11)**, Geneva, 2011.

ITU-T. **Joint Draft 8.0 on Multiview video coding (JVT-AB204)**, 2008.

ITU-T. **Recommendation H.264: Advanced video coding for generic audiovisual services,** Aug., 2017.

ITU-T. **Recommendation ITU-T H.265: High efficiency video coding,** Apr. 2015.

JIA, L.; AU, O. C.; TSU, C.; SHI, Y.; MA, R.; ZHANG, H. A diamond search windowbased adaptive search range algorithm. In: IEEE International Conference on Multimedia & Expo Workshops, ICMEW, 2013. **Proceedings...** San Jose: IEEE, 2013.

JIA, L.; TSUI, C.; AU, O.; ZHENG, A. A Fast Variable Block Size Motion Estimation Algorithm with Refined Search Range for A Two-layer Data Reuse Scheme. In: IEEE International Symposium on Circuits and Systems, ISCAS, 2015. **Proceedings...** Lisbon: IEEE, May. 24-27, 2015, pp. 1206-1209.

JIANG, C.; NOOSHABADI, S. A Scalable Massively Parallel Motion and Disparity Estimation Scheme for Multiview Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, vol. 26, no. 2, pp. 346-359, Feb. 2016.

KALALI, E.; HAMZAOGLU, I. Approximate HEVC fractional Interpolation Filters and Their Hardware Implementations. **Transactions on Consumers Electronics (TCE)**, v. 64, n. 3, pp. 285-291, 2018.

KALALI, E.; OZCAN, E.; YALCINKAYA, O.; HAMZAOGLU, I. A low energy HEVC inverse transform hardware. **Transactions on Consumer Electronics (TCE)**, v. 60, n. 4, pp. 754-761, 2014.

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 (SPIC)**, v. 22, n. 2, Feb. 2007.

KIM, S.; LEE, D.; SOHN, C.; OH, S. Fast motion estimation for HEVC with adaptive search range decision on CPU and GPU. In: IEEE China Summit & International Conference on Signal and Information Processing, ChinaSIP, 2014. **Proceedings...** Xi’an: IEEE, July 9-13, 2014, pp. 349-353.

KIM, M.; LING, N.; SONG, L. Fast single depth intra mode decision for depth map coding in 3D-HEVC. In: IEEE International Conference on Multimedia & Expo Workshops, ICMEW, 2015. **Proceedings...** Turin: IEEE, 2015a.

KIM, D.; MOON, J.; LEE, S. Hardware implementation of HEVC CABAC encoder. In: International SoC Design Conference, ISOCC, 2015. **Proceedings...** Gyungju: IEEE, 2015b.

LAINEMA, J. Intra-Picture Prediction in HEVC. In: SZE, V.; BUDAGAVI, M.; SULLIVAN, G. (Ed.). High efficiency video coding (HEVC). [S.l.]: Springer, 2014. p.91–112.

LEE, B.; KIM, M. A CU-Level Rate and Distortion Estimation Scheme for RDO of Hardware-Friendly HEVC Encoders Using Low-Complexity Integer DCTs. **Transactions on Image Processing (TIP)**, v. 25, n. 8, pp. 3787-3800, 2016.

LEE, T.; CHAN, Y.; SIU, W. Adaptive Search Range for HEVC Motion Estimation Based on Depth Information. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 27, n. 10, pp. 2216-2230, Oct. 2017.

LG. Available in: <http://www.lge.com/br/celulares/lg-P920-smartphone>. Retrieved in: Jan. 2019.

LI, L. **Time-of-Flight Camera – An Introduction (Texas Instruments – Technical White Paper)**, Dallas, 2014a.

LI, X.; WANG, R.; WANG, W.; WANG Z.; DONG, S. Fast motion estimation methods for HEVC. In: IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2014. **Proceedings...** Beijing: IEEE, 2014b.

LIFEWIRE. Available in: < https://www.lifewire.com/why-3d-tv-died-4126776>. Retrieved in: Jan. 2019.

LIU, W.; LI, J.; CHO, Y. B. A novel architecture for parallel multi-view HEVC decoder on mobile device. **EURASIP Journal on Image and Video Processing**, Mar. 2017.

McCANN, K.; ROSEWARNE, C.; BROSS, B.; NACCARI, M.; SHARMAN, K.; SULLIVAN, G. **High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description (JCTVC-S1002)**, Strasbourg, 2014.

MERKLE, P.; SMOLIC, A.; MÜLLER, K.; WIEGAND, T. Efficient Prediction Structures for Multiview Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 17, n. 11, p. 1461-1473, nov. 2007.

MICRON TECHNOLOGY INC., **272b: x64 Mobile LPDDR4 SDRAM Features**. MT53B384M64D4, 2014.

MICROSOFT HOLOLENS TECHNOLOGY. Available in: <https://www.microsoft.com/en-us/hololens>. Retrieved in: Jan., 2019.

MIN, B.; XU, Z.; CHEUNG, R. A Fully Pipelined Hardware Architecture for Intra Prediction of HEVC. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v.27, n. 12, Dec. 2017.

MÜLLER, K.; SCHWARZ, H.; MARPE, D.; BARTNIK, C.; BOSSE, S.; BRUST, H.; HINZ, T.; LAKSHMAN, H.; MERKLE, P.; RHEE, F. H.; TECH, G.; WINKEN, M.; WIEGAND, T. 3D High-Efficiency Video Coding for Multi-View Video and Depth Data. **IEEE Transactions on Image Processing**, [S.l.], v. 22, n. 9, p. 3366-3378, sep. 2013.

MÜLLER, K.; VETRO, A. **Common Test Conditions of 3DV Core Experiments (JCT3V-G1100)**, San José, 2014.

NANGATE. FreePDK45 Generic Open Cell Library. Available in: <http://projects.si2.org/openeda.si2.org/projects/nangatelib>. Retrieved in: Jan., 2019.

NINTENDO. Available in: <https://www.nintendo.com/3ds/features>. Retrieved in: Jan. 2019.

NVSIM. Available in: <http://nvsim.org/>. Retrieved in: Jan., 2019.

OHM, J.-R. **Multimedia Signal Coding and Transmission.** Aachen: Springer , 2015.

PALOMINO, D.; SAMPAIO, F.; AGOSTINI, L.; BAMPI, S.; SUSIN, S. A memory aware multiplierless VLSI architecture for the complete Intra Prediction of the HEVC emerging standard. In: International Conference on Image Processing, ICIP, 2012. **Proceedings...** Orlando: IEEE, Oct. 2012.

PANASONIC HDC-SDT750K. Available in: <ftp://ftp.panasonic.com/camcorder/om/hdc-sdt750_en_om.pdf>. Retrieved Jan., 2019:

PASTUSZAK, G.; ABRAMOWSKI, A. Algorithm and Architecture Design of the H.265/HEVC Intra Encoder. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 26, n. 1, pp. 210-222, 2016a.

PASTUSZAK G.; TROCHIMIUK, M. Algorithm and architecture design of the motion estimation for the H.265/HEVC 4K-UHD encoder. **Journal of Real-Time Image Processing**, v. 12, n. 2, pp. 517-529, Aug. 2016b.

PDM. Avaiable in: < https://pmdtec.com/picofamily/ >. Retrieved in: Jan. 2019.

PEREIRA, F.; DA SILVA, E. A. B. Efficient Plenoptic Imaging Representation: Why Do We Need It?. In: IEEE International Conference on Multimedia and Expo, ICME, 2016. **Proceedings...** Seattle: IEEE, 2016.

PERLEBERG, M.; AFONSO, V.; CONCEIÇÃO, R.; SUSIN, A.; AGOSTINI, L.; ZATT, B.; PORTO, M. Energy and Rate-Aware Design for HEVC Motion Estimation Based on Pareto Efficiency. **Journal of Integrated Circuits and Systems (JICS)**, v. 13, pp. 1-12, 2018.

RAMOS, F.; GOEBEL, J.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power hardware design for the HEVC binary arithmetic encoder targeting 8K videos. In: Symposium on Integrated Circuits and Systems Design, SBCCI, 2016. **Proceedings...** Belo Horizonte: ACM/IEEE, sep. 2016.

RICHARDSON, I. **H.264 and MPEG-4 Video Compression**: Video Coding for Next-Generation Multimedia. Chichester: John Wiley and Sons, 2003.

RICHARDSON, I. **Video Codec Design**: Developing Image and Video Compression Systems. Chichester: John Wiley and Sons, 2002.

SAMPAIO, F.; ZATT, B.; SHAFIQUE, M.; AGOSTINI, L.; BAMPI, S.; HENKEL, J. Energy-Efficient Memory Hierarchy for Motion and Disparity Estimation in Multiview Video Coding. In: Design, Automation & Test in Europe Conf. & Exhibition, DATE, 2013. **Proceedings...** Grenoble: Mar. 18-22, 2013, pp. 665-670.

SAMSUNG. **Galaxy SIII**. Available in: <https://www.samsung.com/us/support/owners/product/galaxy-s-iii-sprint>. Retrieved in: Jan. 2019.

SANCHEZ, G.; AGOSTINI, L.; MARCON, C. Complexity reduction by modes reduction in RD-list for intra-frame prediction in 3D-HEVC depth maps. In: International Symposium on Circuits and Systems, ISCAS, 2017. **Proceedings...** Baltimore: IEEE, 2017a.

SANCHEZ, G.; MARCON, C.; AGOSTINI, L. High Efficient Architecture for 3D-HEVC DMM-1 Decoder Targeting 1080p Videos. In: International Symposium on Circuits and Systems, ISCAS, 2018. **Proceedings...** Florence: IEEE, 2018a.

SANCHEZ, G.; MARCON, C.; AGOSTINI, L. Real-time scalable hardware architecture for 3D-HEVC bipartition modes. **Journal of Real-Time Image Processing**, v.13, n.1, pp. 71-83, jun. 2016.

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, ICIP, 2014. **Proceedings...** Paris: Oct 27-30. 2014a, pp. 3209-3213.

SANCHEZ, G.; SALDANHA, M.; BALOTA, G.; ZATT, B.; PORTO, M.; AGOSTINI, L. DMMFast: a complexity reduction scheme for three-dimensional high-efficiency video coding intraframe depth map coding. **Journal of Electronic Imaging**, v. 24, n. 2, p. 1-15, apr. 2015a.

SANCHEZ, G.; SALDANHA, M.; PORTO, M.; ZATT, B.; AGOSTINI, L. Real-time simplified edge detector architecture for 3D-HEVC depth maps coding. In: International Conference on Electronics, Circuits and Systems, ICECS, 2017. **Proceedings...** Monte Carlo: IEEE, 2017b.

SANCHEZ, G.; SILVEIRA, J.; AGOSTINI, L.; MARCON, C. Performance Analysis of Depth Intra Coding in 3D-HEVC. **Transactions on circuits and systems for video technology (TCSVT)**, Early Access, aug. 2018b.

SANCHEZ, G.; ZATT, B.; PORTO, M.; AGOSTINI, L. A Real-Time 5-Views HD 1080p Architecture for 3D-HEVC Depth Modeling Mode 4. In: ACM/IEEE Symposium on Integrated Circuits and Systems Design, SBCCI, 2014. **Proceedings...** Aracajú: ACM/IEEE, 2014b. p. 1-6.

SANCHEZ, G.; ZATT, B.; PORTO, M.; AGOSTINI, L. Hardware-friendly HEVC motion estimation: new algorithms and efficient VLSI designs targeting high definition videos. **Analog Integrated Circuits and Signal Processing**, v. 82, n. 1, pp. 135-146, 2015b.

SEIDEL, I.; BRÄSCHER, A. B.; GÜNTZEL, J. L.; AGOSTINI, L. Energy-efficient SATD for beyond HEVC. In: IEEE International Symposium on Circuits and Systems, ISCAS, 2016. **Proceedings...** Montreal: ACM/IEEE, 2016.

SHEN, W.; FAN, Y.; BAI, Y.; HUANG, L.; SHANG, Q.; LIU, C.; ZENG, X. A Combined Deblocking Filter and SAO Hardware Architecture for HEVC. **Transactions on Multimedia (TMM)**, v. 18, n. 6, pp. 1022-1033, 2016.

SMOLIC, A.; KAUFF, P.; KNORR, S.; HORNUNG, A.; KUNTER, M.; MÜLLER, M.; LANG, M. Three-Dimensional Video Postproduction and Processing. **Proceedings of the IEEE**, [S.l.], v. 99, n. 4, p. 607-625, apr. 2011.

SONG, C.; JU, L.; JIA, Z. Hybrid Scratchpad and Cache Memory Management for Energy-Efficient Parallel HEVC Encoding. In: 33rd IEEE In: International Conference on Computer Design, ICCD, 2015. **Proceedings...** New York: IEEE, Oct. 18-21, 2015, pp. 712-719.

SONY. Available in: <http://www.sonyrumors.net/wp-content/uploads/2011/06/3D-Movie-Timeline-Medium.jpg?bdc16b>. Retrieved in: Jan. 2019.

STEREOLABS ZED. Available in: <https://www.stereolabs.com/zed/>. Retrieved in: Jan. 2019.

STRUCTURE INC. SENSOR. Available in: <https://structure.io/>. Retrieved in: Jan. 2019.

SULLIVAN, G.; OHM, J.; HAN, W.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v. 22, n. 12, p. 1649–1668. sep. 2012.

SULLIVAN, G.; BOYCE, J.; CHEN, Y.; OHM, J.; SEGALL, C.; VETRO, A. Standardized extensions of High Efficiency Video Coding (HEVC). **Journal of Selected Topics in Signal Processing**, v. 7, n. 6, pp. 1001-1016, 2013.

TECH, G.; CHEN, Y.; MÜLLER, K.; OHM, J.; VETRO, A.; WANG, Y. Overview of the Multiview and 3D extensions of High Efficiency Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, [S.l.], v. 26, n. 1, p. 35-49, Jan. 2016.

TECH, G.; WEGNER, K.; CHEN, Y.; YEA, S. **3D-HEVC Draft Text 7 (JCT3V-K1001)**, Geneva, 2015.

TOSHIBA. Available in: <http://www.toshiba-om.net/pdf/manuals/lcdtv/English/Country_Specific/ZL2-55-English-Specific.pdf>. Retrieved in: Jan. 2019.

TSUNG, P.; CHEN, W.; DING, L.; CHIEN, S.; CHEN, L. Cache-based integer motion/disparity estimation for quad-HD H.264/AVC and HD multiview video coding. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2009. **Proceedings...** Taipei: IEEE, Apr. 19-24, 2009, pp. 2013-2016.

X-BOX. Available in: <http://www.xbox.com/pt-BR/Xbox-One/accessories/kinect-for-xbox-one>. Retrieved in: Jan. 2019.

VITECH. 3D-HEVC Memory Simulator. Available in: <https://wp.ufpel.edu.br/vitech/en/downloads/>. Retrieved in: Jan. 2019.

WANG, B.; ZHOU, J.; KIM, T. SRAM devices and circuits optimization toward energy efficiency in multi-Vth CMOS. **Microelectronics Journal**, v. 46, n. 3, pp. 265-272, mar. 2015.

ZATT, B. **Energy-Efficient Algorithms and Architectures for Multiview Video Coding**. 2012. 236 f. Ph.D. Thesis (Doctorate on Microelectronics) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2012.

ZATT, B.; SHAFIQUE, M.; BAMPI, S.; HENKEL, J. **3D Video Coding for Embedded Devices**: Energy Efficient Algorithms and Architectures. Springer Science, New York, 2013.

ZATT, B.; SHAFIQUE, M.; BAMPI, S.; HENKEL, J. A low-power memory architecture with application-aware power management for motion & disparity estimation in Multiview Video Coding. In: IEEE/ACM Int'l Conf. Computer-Aided Design, ICCAD, 2011. **Proceedings...** San Jose: IEEE/ACM, Nov. 7-10, 2011a, pp. 40-47.

ZATT, B.; SHAFIQUE, M.; SAMPAIO, F.; AGOSTINI, L.; BAMPI, S.; HENKEL, J. Run-time Adaptive Energy-Aware Motion and Disparity Estimation in Multiview Video Coding. In: 48th ACM/EDAC/IEEE Design Automation Conference, DAC, 2011. **Proceedings...** New York: ACM/EDAC/IEEE, Jun. 5-9, 2011b, pp. 1026-1031.

ZHANG, Y.; LU, C. A Highly-Parallel Hardware Architecture of Table-Based CABAC Bit Rate Estimator in HEVC intra encoder. **Transactions on Circuits and Systems for Video Technology (TCSVT)**, Early Access, apr. 2018.

ZHANG, X.; ZHANG, K.; AN, J.; HUANG, H.; LIN, J.-L.; LEI, S. **On Lookup Table Size Reduction for DMM1 (JCT3V-J0035),** Strasbourg, 2014.

ZHOU, W.; ZHANG, J.; ZHOU, X.; LIU, Z.; Liu, X.; A High-Throughput and Multi-Parallel VLSI Architecture for HEVC Deblocking Filter. **Transactions on Multimedia (TMM)**, v. 18, n. 6, pp. 1034-1047, 2016.

APPENDIX A – 3D-HTM EVALUATIONS EXPERIMENTAL SETUP

The experimental conditions of the evaluations performed along this doctorate and presented in this thesis are based on documents available by the JCT-3V group. In the document JCT3V-G1100 (MÜLLER *et al.*, 2014), the JCT-3V recommended test conditions to conduct experiments with the 3D-HTM reference software. Among these conditions, the document defines eight 3D-video sequences using MVD format to be used, as presented in Table A.1. In Table A.1, it is also possible to observe the views to be used as inputs in the experiments using each video sequence, given these 3D-videos have a different number of views. This way, any researcher can conduct the experiments over the same conditions than the others.

Table A.1 – Recommended 3D-video sequences and the views to be used in experiments

ID	**Sequence**	**Views to be used (left-center-right)**
S01	Poznan_Hall2	7-6-5
S02	Poznan_Street	5-4-3
S03	Undo_Dancer	1-5-9
S04	GT_Fly	9-5-1
S05	Kendo	1-3-5
S06	Balloons	1-3-5
S08	Newspaper	2-4-6
S10	Shark	1-5-9

Source: (MÜLLER *et al.*, 2014).

The configurations and test conditions provided by the JCT-3V recommends the use of an Inter-View encoding structure composed of three views with a coding order center/left/right and P-I-P Inter-View prediction. The JCT-3V document also recommends the use of a temporal prediction structure with GoP containing eight frames and Intra-frame occurrences every 24 frames (Random Access every second). Also, the document defines the use of inputs with 8-bit wide samples, internal processing with 8-bit wide samples, and complete resolution for both texture and depth maps.

The QP values to be used in the experiments are also defined in the JCT-3V document. The QP values defined by the JCT-3V to be used in the Texture Independent Views are 25, 30, 35, and 40. The QP values to be used in the depth maps are fixed according to the texture QP values, as presented in Table A.2. QP_{V0} represents the QP values for the texture whereas QP_{D0} represents the QP values for their respective depth maps. Therefore, for the texture QP values recommended by the JCT-3V the correspondent depth-map QP values are 34, 39, 42, and 45.

Table A.2 – QP values to be used for texture and depth maps

QP_{V0}	51	50	49	48	47	46	45	44	43	42	41	**40**	39	38	37	36	**35**	34	33	32	31	**30**	29	28	27	26	**25**
QP_{D0}	51	50	50	50	50	49	48	47	47	46	45	**45**	44	44	43	43	**42**	42	41	41	40	**39**	38	37	36	35	**34**

Source: (MÜLLER *et al.*, 2014).

The eight 3D-video sequences with MVD format recommended by the JCT-3V have between 200 and 300 AUs so that five of them are in the 1920x1088 resolution and the other three are in the 1024x768 resolution, as presented in Table A.3. The frame rate of the sequences varies between 25 and 30 frames per second (fps), which results in 10 seconds of video, except for the sequence *Poznan_Hall2*. Five sequences are real sequences (S01, S02, S05, S06, and S08) recorded using seven or nines cameras spaced every five or 13.75 centimeters whereas the other three sequences are artificially generated using computer graphics (S03, S04, and S10), as shown in Table A.3.

Table A.3 – Technical aspects of the 3D-video sequences

Test Sequence	**Resolution**	**AUs**	**Frame rate (fps)**	**Cameras**	**Inter-lens Camera Spacing**
S01 – Poznan_Hall2	1920x1088	200	25	9	13,75 cm
S02 – Poznan_Street	1920x1088	250	25	9	13,75 cm
S03 – Undo_Dancer	1920x1088	250	25	9	–
S04 – GT_Fly	1920x1088	250	25	9	–
S05 – Kendo	1024x768	300	30	7	5 cm
S06 – Balloons	1024x768	300	30	7	5 cm
S08 – Newspaper1	1024x768	300	30	9	5 cm
S10 – Shark	1920x1088	300	30	9	–

Source: Author.

The main image aspects related to each one of the sequences are presented in Table A.4. Note that the sequences provided by the JCT-3V present well-diversified characteristics: sequences with internal recording environment, external recording environment, or computer graphics; sequences with or without camera movement; sequences with different levels of object movement; sequences with natural or artificial lighting; as well as different image details and depth structures.

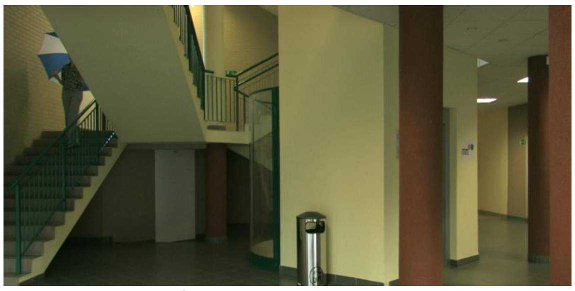
Table A.4 – Main image aspects of the 3D-video sequences

Test Sequence	Recording Environment	Object Movement	Camera Movement	Image Detailing Level	Depth Structures	Light
S01 – Poznan_Hall2	Internal	Complex (reflections and transparences)	Yes	Intermediate	Complex	Artificial
S02 – Poznan_Street	External	Complex (reflections and transparences)	No	High	Complex	Natural
S03 – Undo_Dancer	Computer Graphics	Complex	–	High	Simple	Computer Graphics
S04 – GT_Fly	Computer Graphics	Simple	–	Intermediate	Complex	Computer Graphics
S05 – Kendo	Internal	Complex (smoke, reflections and transparences)	Yes	High	Intermediate	Artificial Studio Light
S06 – Balloons	Internal	Complex (reflections and transparences)	Yes	Intermediate	Intermediate	Artificial Studio Light
S08 – Newspaper	Internal	Simple	No	High	Intermediate	Artificial Studio Light
S10 – Shark	Computer Graphics	Complex	–	High	Complex	Computer Graphics

Source: Author.

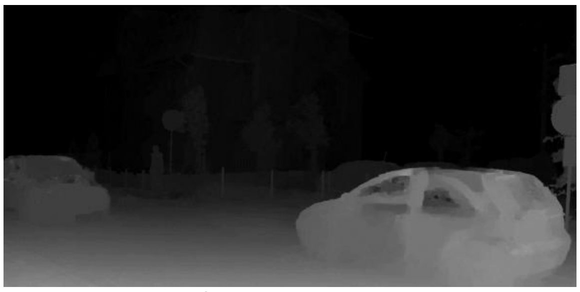
The first frames of the 3D-video test sequences provided by the JCT-3V can be observed in Figures A.1 and A.2, in which are presented the texture pictures and their respective depth maps. Figure A.1 shows the 1920x1088 sequences, whereas Figure A.2 shows the 1024x768 sequences.

Figure A.1 – First frames of the 1920x1088 3D-video test sequences

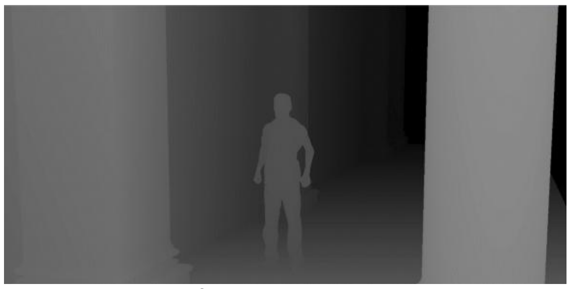

Texture of Poznan_Hall2 test sequence.

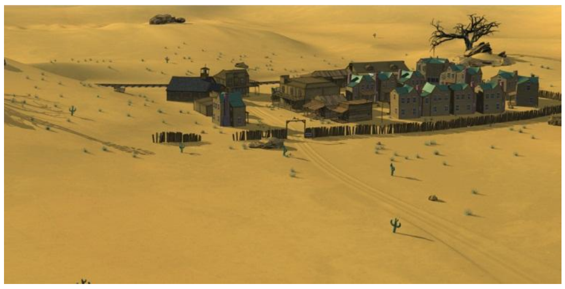
Depth-map of Poznan_ Hall2 test sequence.

Texture of Poznan_street test sequence.


Depth-map of Poznan_street test sequence.

Texture of Undo_Dancer test sequence.


Depth-map of Undo_Dancer test sequence.


Texture of GT_Fly test sequence.

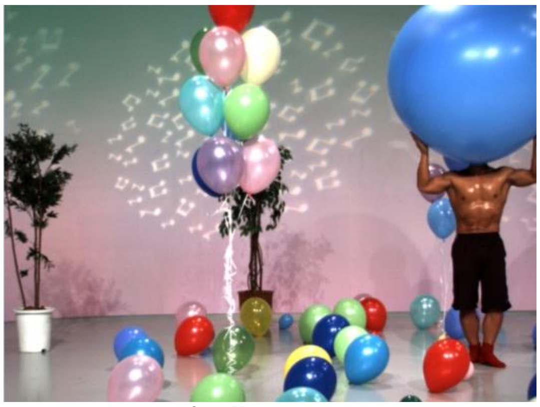
Depth-map of GT_Fly test sequence.

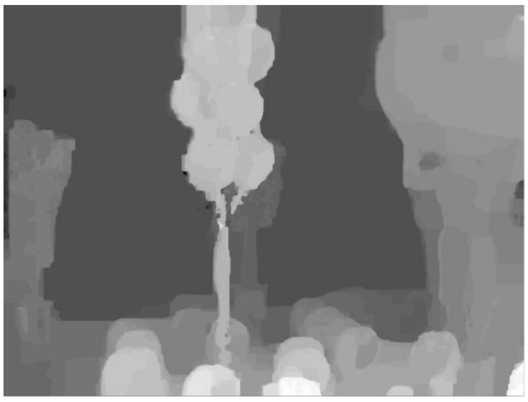
Texture of Shark test sequence.

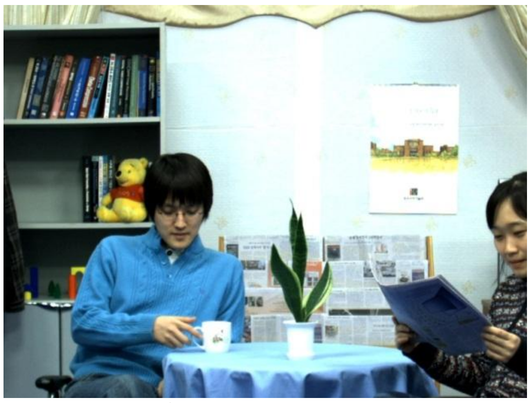
Depth-map of Shark test sequence.

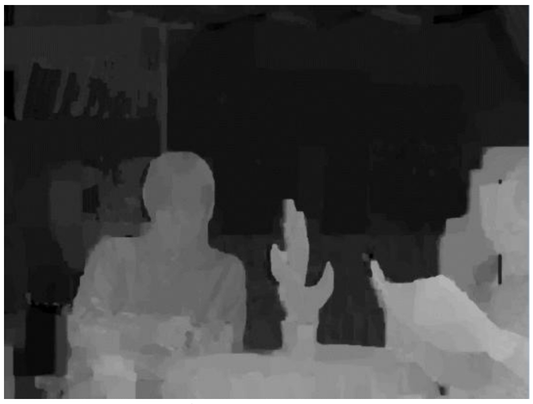
Source: (MÜLLER *et al.*, 2014).

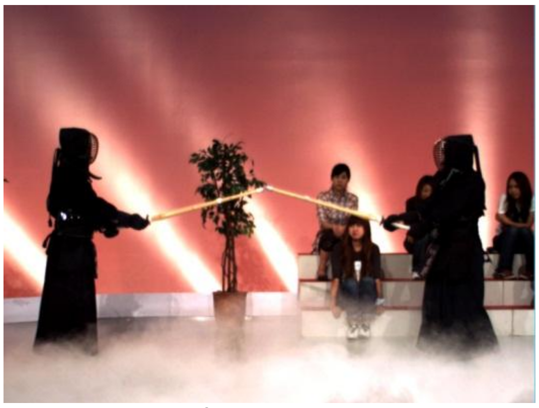
Figure A.2 – First frames of the 1024x768 3D-video Test Sequences

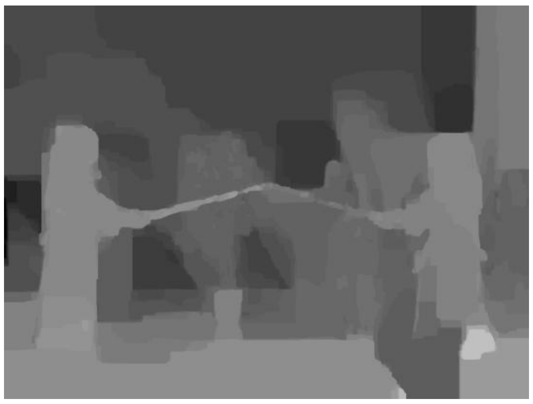

Texture of Balloons test sequence.


Depth-map of Balloons test sequence.


Texture of Newspaper1 test sequence.


Depth-map of Newspaper1 test sequence.


Texture of Kendo test sequence.


Depth-map of Kendo test sequence.

Source: (MÜLLER *et al.*, 2014).

Several architectures were developed during the doctorate based on heuristics and techniques evaluated with exhaustive software simulations. Once these software evaluations occurred in different moments of the work, the experimental setup adopted for each one of the experiments had slight variations. These variations occurred because new versions of the 3D-HEVC reference software and standardization documents emerged during the development of this work. Anyway, the absolute majority of the experiments followed the recommendations given in the common test conditions, i.e., they used all eight video sequences and four QP pairs recommended by the JCT-3V, as presented in Table A.5. Also, almost all experiments used the 3D-HTM in version 16 and considered all AUs of the sequences. Table A.5 depicts the exact experimental setup used to guide each evaluation done using the 3D-HTM.

Table A.5 – Experimental setup summary according to each experiment

Evaluations	**3D-HEVC Reference Software Version**	**Temporal Configuration**	**Coding Order Configuration**	**Sequences**	**Quantization Parameters**	**Number of Access Units**
Section 6.1.1	3D-HTM 16.0	All Intra	Conventional Coding Order	Eight (according to the CTC)	Four pairs (according to the CTC)	All AUs (between 200 and 300 AUs)
Section 6.1.2	3D-HTM 16.0	Random Access	Conventional Coding Order	Two sequences (Ballons and Undo_Dancer) recommended in the CTC	One QP pair recommended in the CTC (30/39)	All AUs (between 200 and 300 AUs)
Section 6.2	3D-HTM 16.0	Random Access	Conventional and Flexible Coding Orders	Eight (according to the CTC)	Four pairs (according to the CTC)	All AUs (between 200 and 300 AUs)
Section 7.1	3D-HTM 16.0	All Intra	Conventional Coding Order	Eight (according to the CTC)	Four pairs (according to the CTC)	All AUs (between 200 and 300 AUs)
Section 7.2	3D-HTM 15.1	All Intra and Random Access	Conventional Coding Order	Eight (according to the CTC)	Four sets (according to the CTC)	All AUs (between 200 and 300 AUs)
Section 8.1	3D-HTM 16.2	Random Access	Flexible Coding Order	Eight (according to the CTC)	Four pairs (according to the CTC)	48 AUs
Section 8.2	3D-HTM 16.2	Random Access	Conventional Coding Order	Eight (according to the CTC)	Four pairs (according to the CTC)	All AUs (between 200 and 300 AUs)

Source: Author.

APPENDIX B – SYNTHESIS METHODOLOGY APPLIED TO THE DEVELOPED ARCHITECTURES/SYSTEMS

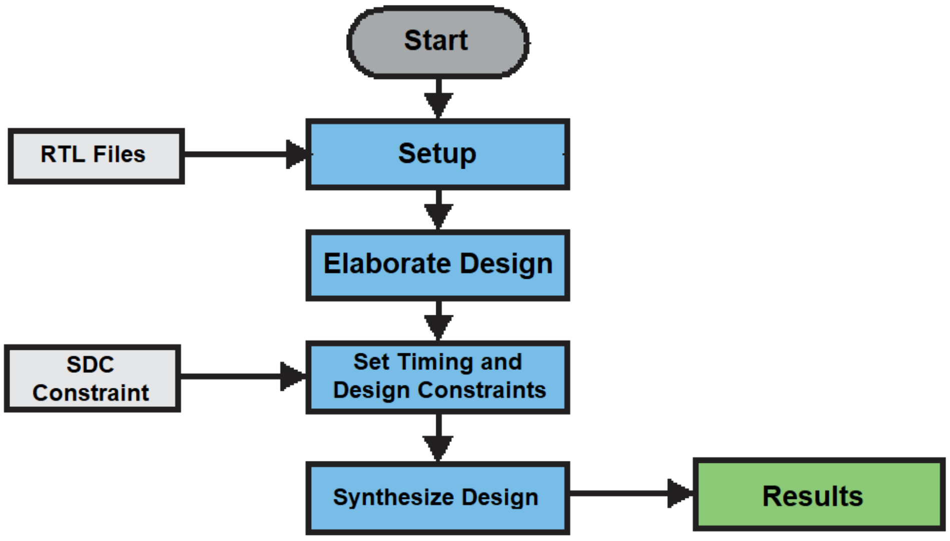
All architectures and systems developed in this doctorate were described in VHDL, and the synthesis results were generated targeting an ASIC technology using the Cadence Encounter RTL Compiler 11.10 tool (CADENCE, 2019). The ASIC synthesis results were obtained for 45nm Nangate standard cells (NANGATE, 2019). Also, the gate count presented along this thesis is calculated based on 2-input NANDs.

The power results of all developed architectures were also estimated using the Cadence RTL Compiler 11.10 tool. These power results were provided by RTL Files and default switching activity targeting 45nm standard-cells and they considered a voltage supply of 0.95V. All developed architectures used the clock-gating low-power technique enabled for the automatic logic synthesis.

The default flow for the logic synthesis provided by RTL Compiler tool is basically composed of four steps, as follows: (i) Setup, responsible for set the target library and activate some improvements as clock gating; (ii) Elaborate Design, responsible for identifying the structure and building the architecture; (iii) Set Timing and Design Constraints, which defines timing and operating conditions for the design; (iv) Synthesize Design step, responsible for the design synthesis and mapping to the target technology, which generates the gate level net list of standard cells available in the library (NANGATE, 2019). After these steps, the power analysis can be done, which is responsible for analyzing and reporting the power estimation.

Figure B.1 depicts the default flow provided by RTL Compiler tool for power estimation. It requires, as inputs the RTL files (HDL files), and the Synopsys Design Constraints file (SDC file), containing the operating conditions for the design, as target frequency, input and output delay, clock latency, etc. This flow considers a default switching-activity value, defined by the RTL Compiler tool. The default toggle percentage is 20% on each input pin, where the toggle percentage is a multiplication factor (scale factor – 0.2) to be used with the clock to modify the toggle rate.

Figure B.1 – Default flow for power estimation provided by the RTL Compiler tool



Source: Author.

APPENDIX C – BD-RATE RESULTS ACCORDING TO THE 3D-VIDEO SEQUENCES

This appendix shows the impact on the compression according to the 3D-video sequences for the four architectures/systems presented in this thesis. The results consider the BD-Rate metric. When *Video Total* is referred, the quality related to texture channel (disregarding synthesized frames) considering the total bit rate of the video (texture + depth) is measured. When *Synthesized Frames* are referred, the quality of frames synthesized with encoded texture and encoded depth maps, using DIBR, is measured in comparison with frames synthesized with original texture and original depth maps. Finally, when *Video Only* is referred, the quality related to the texture channel using the texture bit rate of the video (disregarding synthesized frames) is measured. The BD-Rate results are showed for each architecture/system divided into four subsections, as follows.

C.1 Low-Power Depth Intra Skip Architecture based on Distortion Metric Replacing

This subsection presents the BD-Rate increase according to the 3D-video sequences for the Low-Power Depth Intra Skip Architecture presented in Chapter 7. Table C.1 presents the BD-Rate increase of replacing the SVDC by the SAD as the similarity criterion.

Table C.1 – BD-Rate increase according to the 3D-video sequences of replacing the SVDC by the SAD as the similarity criterion

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	0.00%	0.00%	0.00%	0.00%	0.16%	0.23%
Kendo	0.00%	0.00%	0.00%	0.00%	0.22%	0.34%
Newspaper_CC	0.00%	0.00%	0.00%	0.00%	0.22%	0.26%
GT_Fly	0.00%	0.00%	0.00%	0.00%	0.14%	0.20%
Poznan_Hall2	0.00%	0.00%	0.00%	0.00%	0.40%	0.29%
Poznan_Street	0.00%	0.00%	0.00%	0.00%	0.10%	0.14%
Undo_Dancer	0.00%	0.00%	0.00%	0.00%	0.07%	0.02%
Shark	0.00%	0.00%	0.00%	0.00%	0.17%	0.21%
1024x768	0.00%	0.00%	0.00%	0.00%	0.20%	0.28%
1920x1088	0.00%	0.00%	0.00%	0.00%	0.18%	0.17%
Average	0.00%	0.00%	0.00%	0.00%	0.19%	0.21%

Source: Author.

C.2 Low-Power and Memory-Aware Depth-map Intra-frame Prediction System based on Complexity Reduction Heuristics

This subsection presents the BD-Rate variation according to the 3D-video sequences for the Low-Power and Memory-Aware Depth-map Intra-frame Prediction System based on Complexity Reduction Heuristics presented in Chapter 7. Table C.2 presents the BD-Rate variation of combining all complexity-reduction strategies considering the AI configuration. Table C.3 also shows the BD-Rate variation of combining all these strategies but considering the RA configuration. Also, tables C.2 and C.3 present the total and depth time reductions by adopting all complexity-reduction heuristics.

Table C.2 – BD-Rate variation according to the 3D-video sequences in AI configuration

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized	Time Reduction	Depth Reduction
Balloons	0.00%	0.00%	0.00%	0.00%	-0.02%	7.96%	45.86%	53.26%
Kendo	0.00%	0.00%	0.00%	0.00%	0.02%	8.04%	46.29%	53.81%
Newspaper_CC	0.00%	0.00%	0.00%	0.00%	-0.89%	12.09%	48.35%	54.46%
GT_Fly	0.00%	0.00%	0.00%	0.00%	-0.61%	3.82%	46.68%	54.01%
Poznan_Hall2	0.00%	0.00%	0.00%	0.00%	-0.29%	7.98%	46.39%	54.69%
Poznan_Street	0.00%	0.00%	0.00%	0.00%	-0.39%	3.31%	47.27%	53.88%
Undo_Dancer	0.00%	0.00%	0.00%	0.00%	-0.18%	7.55%	45.20%	53.45%
Shark	0.00%	0.00%	0.00%	0.00%	-0.87%	6.58%	46.64%	54.30%
1024x768	0.00%	0.00%	0.00%	0.00%	-0.30%	9.36%	46.83%	53.84%
1920x1088	0.00%	0.00%	0.00%	0.00%	-0.47%	5.85%	46.44%	54.07%
Average	0.00%	0.00%	0.00%	0.00%	-0.41%	7.16%	46.58%	53.98%

Source: Author.

Table C.3 – BD-Rate variation according to the 3D-video sequences in RA configuration

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized	Time Reduction	Depth Reduction
Balloons	0.00%	0.15%	0.32%	0.08%	0.22%	2.73%	14.95%	32.21%
Kendo	0.00%	0.12%	0.11%	0.04%	0.61%	2.31%	14.88%	31.13%
Newspaper_CC	0.00%	0.21%	0.16%	0.06%	-0.32%	6.48%	18.01%	35.03%
GT_Fly	0.00%	0.75%	0.54%	0.13%	-0.05%	0.88%	15.48%	31.92%
Poznan_Hall2	0.00%	0.08%	0.05%	0.00%	-0.15%	2.32%	17.01%	33.94%
Poznan_Street	0.00%	0.57%	0.71%	0.18%	0.00%	2.13%	18.57%	35.61%
Undo_Dancer	0.00%	1.30%	1.23%	0.33%	0.53%	2.28%	14.56%	31.35%
Shark	0.00%	1.02%	1.04%	0.22%	0.23%	1.98%	15.89%	31.34%
1024x768	0.00%	0.16%	0.20%	0.06%	0.17%	3.84%	15.94%	32.79%
1920x1088	0.00%	0.74%	0.72%	0.17%	0.11%	1.92%	16.30%	32.83%
Average	0.00%	0.52%	0.52%	0.13%	0.14%	2.64%	16.17%	32.82%

Source: Author.

C.3 Energy-Aware Motion and Disparity Estimation System with Run-time Adaptive Memory Hierarchy

This subsection presents the BD-Rate increase according to the 3D-video sequences by adopting the proposed hardware-oriented constraints (HCs) and the developed algorithms/techniques for the Energy-Aware Motion and Disparity Estimation System presented in Chapter 8. Tables from Table C.4 to Table C.9 present the results for the six scenarios tested with the HC1. Tables from Table C.10 to Table C.12 present the BD-Rate increase for the three scenarios tested with the HC2. Table C.13 presents the results for the

HC3. Table C.14 shows the BD-Rate increase for the HC4. Tables from Table C.15 to Table C.18 present the impact on compression by using the depth subsampling considering four different test scenarios. Finally, Table C.19 shows the BD-Rate increase of combining all adopted algorithms/techniques along with all hardware-oriented constraints (HC4).

Table C.4 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 8x8 and 16x16

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	6.5%	7.2%	5.5%	7.4%	8.1%	7.65%
Kendo	9.6%	14.3%	9.3%	11.7%	13.6%	12.65%
Newspaper_CC	2.5%	5.3%	3.9%	4.1%	5.5%	5.01%
GT_Fly	13.7%	7.0%	5.6%	14.4%	15.7%	15.02%
Poznan_Hall2	13.8%	16.9%	17.4%	19.0%	20.9%	18.92%
Poznan_Street	4.7%	8.9%	10.7%	8.6%	10.5%	8.48%
Undo_Dancer	6.3%	2.2%	1.6%	7.5%	9.8%	9.05%
Shark	13.2%	8.4%	8.5%	13.5%	18.7%	18.89%
1024x768	6.2%	8.9%	6.2%	7.8%	9.1%	8.44%
1920x1088	10.4%	8.7%	8.8%	12.6%	15.1%	14.07%
Average	8.8%	8.8%	7.8%	10.8%	12.8%	11.96%

Source: Author.

Table C.5 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 8x8 and 32x32

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	3.8%	4.0%	3.8%	4.8%	5.4%	5.04%
Kendo	5.9%	8.2%	4.4%	6.8%	7.4%	7.00%
Newspaper_CC	2.1%	3.0%	2.5%	3.1%	3.6%	3.30%
GT_Fly	10.7%	2.7%	2.9%	10.8%	11.5%	11.08%
Poznan_Hall2	8.1%	9.3%	10.3%	11.7%	12.4%	10.90%
Poznan_Street	3.9%	3.3%	5.6%	6.3%	7.0%	5.70%
Undo_Dancer	5.3%	0.8%	0.9%	6.2%	7.3%	6.72%
Shark	10.7%	5.5%	4.6%	10.7%	12.6%	12.76%
1024x768	3.9%	5.1%	3.6%	4.9%	5.4%	5.12%
1920x1088	7.7%	4.3%	4.9%	9.1%	10.2%	9.43%
Average	6.3%	4.6%	4.4%	7.6%	8.4%	7.81%

Source: Author.

Table C.6 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 8x8 and 64x64

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.5%	6.8%	6.1%	7.7%	8.0%	7.59%
Kendo	8.4%	10.4%	5.8%	9.3%	9.6%	8.98%
Newspaper_CC	3.8%	5.2%	2.8%	4.9%	5.0%	4.65%
GT_Fly	15.1%	5.7%	6.5%	14.8%	15.4%	14.89%
Poznan_Hall2	8.9%	8.3%	9.5%	11.9%	12.3%	11.06%
Poznan_Street	6.4%	4.3%	7.3%	9.4%	10.1%	8.41%
Undo_Dancer	7.2%	1.4%	2.3%	8.4%	9.1%	8.32%
Shark	19.2%	11.0%	9.5%	18.8%	20.2%	20.05%
1024x768	6.2%	7.5%	4.9%	7.3%	7.6%	7.08%
1920x1088	11.3%	6.1%	7.0%	12.7%	13.4%	12.55%
Average	9.4%	6.6%	6.2%	10.7%	11.2%	10.49%

Source: Author.

Table C.7 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 16x16 and 32x32

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	3.09%	3.80%	3.17%	3.97%	3.98%	4.92%
Kendo	4.55%	7.20%	3.18%	5.28%	5.36%	5.22%
Newspaper_CC	1.78%	2.81%	2.07%	2.52%	2.10%	2.43%
GT_Fly	12.88%	2.59%	1.44%	12.90%	12.53%	12.35%
Poznan_Hall2	6.82%	8.77%	9.23%	9.92%	10.40%	9.12%
Poznan_Street	3.02%	3.31%	5.46%	4.99%	5.12%	4.34%
Undo_Dancer	5.47%	0.12%	-0.12%	6.17%	6.51%	5.97%
Shark	9.40%	4.26%	3.29%	9.30%	10.02%	10.40%
1024x768	3.14%	4.60%	2.81%	3.92%	3.81%	4.19%
1920x1088	7.52%	3.81%	3.86%	8.66%	8.92%	8.44%
Average	5.88%	4.11%	3.46%	6.88%	7.00%	6.85%

Source: Author.

Table C.8 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 16x16 and 64x64

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	3.5%	4.6%	3.3%	4.7%	4.7%	5.34%
Kendo	4.7%	7.0%	3.5%	5.5%	5.6%	5.36%
Newspaper_CC	2.2%	3.1%	2.0%	3.1%	2.8%	3.03%
GT_Fly	13.6%	2.9%	2.6%	13.4%	12.9%	12.57%
Poznan_Hall2	4.7%	4.6%	6.5%	6.9%	7.2%	6.66%
Poznan_Street	3.3%	2.7%	3.4%	5.2%	5.2%	4.28%
Undo_Dancer	5.7%	0.2%	0.1%	6.7%	6.6%	5.96%
Shark	11.1%	4.4%	4.0%	10.8%	10.8%	11.05%
1024x768	3.5%	4.9%	2.9%	4.5%	4.4%	4.58%
1920x1088	7.7%	2.9%	3.3%	8.6%	8.5%	8.10%
Average	6.1%	3.7%	3.2%	7.0%	7.0%	6.78%

Source: Author.

Table C.9 – BD-Rate increase according to the 3D-video sequences for the HC1 by constraining the block sizes to 32x32 and 64x64

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	5.0%	4.4%	4.9%	5.9%	5.8%	6.94%
Kendo	7.5%	8.5%	4.2%	7.9%	7.2%	7.28%
Newspaper_CC	3.7%	3.6%	2.1%	4.4%	2.9%	3.55%
GT_Fly	31.1%	5.3%	5.1%	29.8%	27.1%	26.72%
Poznan_Hall2	7.5%	5.8%	6.3%	8.9%	8.5%	7.74%
Poznan_Street	4.9%	2.0%	4.7%	6.9%	6.5%	5.56%
Undo_Dancer	12.1%	2.8%	4.8%	12.9%	11.8%	10.99%
Shark	19.5%	7.9%	6.2%	18.8%	16.7%	17.22%
1024x768	5.4%	5.5%	3.7%	6.1%	5.3%	5.92%
1920x1088	15.0%	4.7%	5.4%	15.5%	14.1%	13.64%
Average	11.4%	5.0%	4.8%	11.9%	10.8%	10.75%

Source: Author.

Table C.10 – BD-Rate increase according to the 3D-video sequences for the HC2 by removing the TZS predictors (except the *Zero Predictor*) (in addition to the HC1)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	3.3%	4.1%	3.1%	4.1%	4.1%	4.90%
Kendo	5.3%	7.7%	3.8%	6.1%	6.5%	6.29%
Newspaper_CC	1.8%	4.1%	2.5%	3.2%	3.0%	3.07%
GT_Fly	21.4%	5.0%	5.6%	21.1%	19.8%	19.41%
Poznan_Hall2	7.7%	9.8%	12.7%	11.9%	11.9%	10.33%
Poznan_Street	8.0%	4.2%	5.4%	10.0%	9.7%	8.53%
Undo_Dancer	7.5%	0.9%	1.1%	8.3%	8.4%	7.72%
Shark	19.6%	7.6%	4.7%	19.2%	17.7%	18.07%
1024x768	3.5%	5.3%	3.1%	4.5%	4.5%	4.75%
1920x1088	12.8%	5.5%	5.9%	14.1%	13.5%	12.81%
Average	9.3%	5.4%	4.9%	10.5%	10.1%	9.79%

Source: Author.

Table C.11 – BD-Rate increase according to the 3D-video sequences for the HC2 by removing the TZS predictors (except the *Zero Predictor*), and the *Raster* step (in addition to the HC1)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	3.24%	4.15%	2.85%	4.19%	4.36%	4.93%
Kendo	5.91%	7.95%	4.64%	6.83%	7.02%	6.70%
Newspaper_CC	1.79%	4.62%	3.04%	3.31%	3.17%	3.51%
GT_Fly	29.16%	7.34%	6.79%	28.56%	26.21%	25.58%
Poznan_Hall2	8.55%	11.14%	13.57%	13.21%	13.45%	11.18%
Poznan_Street	9.16%	4.16%	5.99%	11.08%	10.56%	9.43%
Undo_Dancer	7.51%	1.10%	0.96%	8.04%	8.09%	7.63%
Shark	26.30%	11.29%	7.06%	25.78%	23.86%	24.05%
1024x768	3.65%	5.57%	3.51%	4.77%	4.85%	5.05%
1920x1088	16.14%	7.01%	6.87%	17.33%	16.44%	15.57%
Average	11.45%	6.47%	5.61%	12.62%	12.09%	11.63%

Source: Author.

Table C.12 – BD-Rate increase according to the 3D-video sequences for the HC2 by removing the TZS predictors (except the *Zero Predictor*), the *Raster* step, and the *Refinement* step (in addition to the HC1)

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	7.0%	10.4%	8.2%	9.8%	9.5%	9.35%
Kendo	15.4%	18.4%	14.2%	18.5%	18.5%	16.51%
Newspaper_CC	3.2%	12.3%	18.5%	8.7%	8.8%	7.49%
GT_Fly	97.5%	44.2%	25.9%	96.6%	88.8%	87.07%
Poznan_Hall2	29.3%	30.7%	39.4%	41.0%	38.6%	33.21%
Poznan_Street	25.1%	26.9%	27.8%	33.0%	31.9%	28.03%
Undo_Dancer	29.2%	6.5%	10.3%	30.5%	28.8%	27.93%
Shark	95.1%	47.5%	35.1%	93.3%	91.4%	91.57%
1024x768	8.5%	13.7%	13.6%	12.3%	12.3%	11.11%
1920x1088	55.3%	31.2%	27.7%	58.9%	55.9%	53.56%
Average	37.7%	24.6%	22.4%	41.4%	39.6%	37.64%

Source: Author.

Table C.13 – BD-Rate increase according to the 3D-video sequences for the HC3 by considering the horizontal-only disparity search (in addition to the HC2)

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	3.24%	4.57%	6.63%	5.31%	5.76%	5.74%
Kendo	5.91%	8.06%	11.28%	8.41%	9.08%	8.02%
Newspaper_CC	1.79%	4.49%	19.00%	5.96%	6.20%	5.47%
GT_Fly	29.16%	8.10%	5.25%	28.86%	26.84%	26.10%
Poznan_Hall2	8.55%	12.86%	28.57%	18.29%	18.81%	14.45%
Poznan_Street	9.16%	5.02%	35.45%	16.93%	17.11%	13.20%
Undo_Dancer	7.51%	0.89%	0.30%	8.72%	8.86%	8.06%
Shark	26.30%	11.38%	5.55%	25.89%	24.58%	24.72%
1024x768	3.65%	5.71%	12.30%	6.56%	7.01%	6.41%
1920x1088	16.14%	7.65%	15.02%	19.74%	19.24%	17.31%
Average	11.45%	6.92%	14.00%	14.79%	14.65%	13.22%

Source: Author.

Table C.14 – BD-Rate increase according to the 3D-video sequences for the HC4 by considering the SAD as the only similarity criterion for ME/DE, bi-directional prediction disabling, and the search limitation to one reference frame per direction (in addition to the HC3, defined as the B-Encoder configuration and the Base Memory Hierarchy – BMH)

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	6.02%	8.05%	7.17%	6.88%	7.21%	8.38%
Kendo	22.01%	18.79%	17.16%	22.43%	21.51%	19.86%
Newspaper_CC	5.11%	10.06%	5.97%	6.42%	6.90%	7.14%
GT_Fly	46.79%	18.99%	19.68%	45.57%	42.96%	42.23%
Poznan_Hall2	17.32%	21.14%	21.94%	22.80%	22.89%	20.08%
Poznan_Street	17.27%	12.96%	16.66%	19.50%	18.99%	17.53%
Undo_Dancer	17.31%	7.73%	7.21%	18.09%	17.31%	16.10%
Shark	58.71%	32.28%	30.78%	57.90%	55.51%	54.47%
1024x768	11.05%	12.30%	10.10%	11.91%	11.87%	11.79%
1920x1088	31.48%	18.62%	19.25%	32.77%	31.53%	30.08%
Average	23.82%	16.25%	15.82%	24.95%	24.16%	23.22%

Source: Author.

Table C.15 – BD-Rate increase according to the 3D-video sequences for the depth subsampling technique by removing the four less significant bits (in addition to the HC4, defined as the Reduced-Size Hierarchy – RSH)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.02%	7.18%	7.60%	6.76%	6.87%	8.21%
Kendo	22.01%	19.32%	16.69%	22.26%	21.76%	20.48%
Newspaper_CC	5.11%	9.58%	6.06%	6.38%	6.81%	7.31%
GT_Fly	46.79%	20.44%	19.84%	45.68%	43.47%	42.96%
Poznan_Hall2	17.32%	20.97%	22.87%	22.70%	23.19%	21.26%
Poznan_Street	17.27%	13.21%	16.90%	19.71%	19.53%	18.04%
Undo_Dancer	17.31%	8.65%	7.47%	18.23%	17.93%	17.04%
Shark	58.71%	32.89%	31.83%	57.92%	56.05%	55.17%
1024x768	11.05%	12.02%	10.12%	11.80%	11.81%	12.00%
1920x1088	31.48%	19.23%	19.78%	32.85%	32.03%	30.89%
Average	23.82%	16.53%	16.16%	24.95%	24.45%	23.81%

Source: Author.

Table C.16 – BD-Rate increase according to the 3D-video sequences for the depth subsampling technique by removing the four more significant bits (in addition to the HC4)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.02%	8.06%	6.87%	6.83%	6.95%	8.94%
Kendo	22.01%	18.58%	16.90%	22.32%	22.43%	21.61%
Newspaper_CC	5.11%	9.65%	5.87%	6.45%	7.08%	8.29%
GT_Fly	46.79%	21.34%	20.22%	45.70%	44.62%	44.49%
Poznan_Hall2	17.32%	21.28%	22.96%	22.66%	23.23%	21.87%
Poznan_Street	17.27%	13.09%	16.65%	19.56%	19.99%	18.78%
Undo_Dancer	17.31%	8.56%	7.43%	18.20%	19.75%	20.82%
Shark	58.71%	32.35%	30.89%	58.11%	62.68%	63.77%
1024x768	11.05%	12.10%	9.88%	11.87%	12.15%	12.95%
1920x1088	31.48%	19.32%	19.63%	32.84%	34.05%	33.95%
Average	23.82%	16.61%	15.97%	24.98%	25.84%	26.07%

Source: Author.

Table C.17 – BD-Rate increase according to the 3D-video sequences for the depth subsampling technique by removing the even bits (in addition to the HC4)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.02%	7.44%	6.83%	6.70%	6.94%	8.25%
Kendo	22.01%	18.62%	17.07%	22.38%	21.78%	20.57%
Newspaper_CC	5.11%	10.47%	5.92%	6.35%	6.71%	7.57%
GT_Fly	46.79%	20.40%	19.89%	45.56%	43.54%	42.99%
Poznan_Hall2	17.32%	21.43%	22.23%	22.84%	23.36%	21.27%
Poznan_Street	17.27%	12.58%	16.87%	19.53%	19.71%	18.45%
Undo_Dancer	17.31%	7.95%	7.09%	18.18%	17.90%	17.03%
Shark	58.71%	32.56%	30.36%	58.10%	59.80%	60.00%
1024x768	11.05%	12.18%	9.94%	11.81%	11.81%	12.13%
1920x1088	31.48%	18.98%	19.29%	32.84%	32.86%	31.95%
Average	23.82%	16.43%	15.78%	24.96%	24.97%	24.52%

Source: Author.

Table C.18 – BD-Rate increase according to the 3D-video sequences for the depth subsampling technique by removing the odd bits (in addition to the HC4)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.02%	7.87%	7.16%	6.76%	6.91%	8.35%
Kendo	22.01%	19.00%	16.58%	22.24%	21.59%	20.36%
Newspaper_CC	5.11%	10.21%	6.24%	6.32%	6.56%	7.58%
GT_Fly	46.79%	19.80%	18.91%	45.41%	43.44%	43.10%
Poznan_Hall2	17.32%	21.47%	22.57%	22.75%	23.39%	21.79%
Poznan_Street	17.27%	13.53%	16.48%	19.72%	19.51%	18.03%
Undo_Dancer	17.31%	8.56%	7.69%	18.18%	19.19%	19.23%
Shark	58.71%	32.79%	31.44%	57.96%	56.84%	56.27%
1024x768	11.05%	12.36%	9.99%	11.78%	11.69%	12.10%
1920x1088	31.48%	19.23%	19.42%	32.80%	32.47%	31.68%
Average	23.82%	16.65%	15.88%	24.92%	24.68%	24.34%

Source: Author.

Table C.19 – BD-Rate increase according to the 3D-video sequences using all proposed techniques (in addition to the HC4 and depth subsampling, defined as the Run-Time Adaptive Hierarchy – RAH)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	6.02%	6.72%	7.21%	6.86%	7.05%	8.30%
Kendo	22.01%	19.42%	17.19%	22.57%	22.14%	20.66%
Newspaper_CC	5.11%	12.58%	6.46%	7.00%	7.47%	7.74%
GT_Fly	46.79%	21.54%	20.99%	45.72%	43.45%	43.00%
Poznan_Hall2	17.32%	28.34%	33.15%	28.69%	30.02%	25.71%
Poznan_Street	17.27%	12.27%	16.62%	19.37%	19.04%	17.69%
Undo_Dancer	17.31%	8.23%	7.17%	18.35%	18.36%	17.66%
Shark	58.71%	32.57%	31.43%	57.95%	57.50%	57.30%
1024x768	11.05%	12.91%	10.29%	12.14%	12.22%	12.24%
1920x1088	31.48%	20.59%	21.87%	34.02%	33.67%	32.27%
Average	23.82%	17.71%	17.53%	25.81%	25.63%	24.76%

Source: Author.

C.4 Low-power and Coding-Efficient Disparity Estimation Architecture based on Improved Unidirectional Disparity-Search Algorithm

This subsection presents the BD-Rate variation according to the 3D-video sequences by adopting the iUDS algorithm rather using the TZS algorithm in the 3D-HEVC DE. This approach was integrated on the Low-power and Coding-Efficient Disparity Estimation Architecture presented in Chapter 8. Four test cases (TCs) are considered in this appendix: TC1, TC4, TC5, and TC8.

Table C.20 – BD-Rate variation according to the 3D-video sequences by using the iUDS under the Test Case 1 (UDS 4-2 without refinement step)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	0.0000%	0.4814%	0.3425%	0.1883%	0.1815%	0.1360%
Kendo	0.0000%	0.2038%	0.0117%	0.0637%	0.0410%	0.0377%
Newspaper_CC	0.0000%	0.5097%	0.8659%	0.2989%	0.2785%	0.2552%
GT_Fly	0.0000%	0.9818%	1.1340%	0.4393%	0.4473%	0.3361%
Poznan_Hall2	0.0000%	3.0303%	1.8482%	1.1318%	1.0747%	0.9198%
Poznan_Street	0.0000%	2.4736%	2.6187%	0.8525%	0.8382%	0.7493%
Undo_Dancer	0.0000%	1.3111%	1.1077%	0.4653%	0.5237%	0.4672%
Shark	0.0000%	-0.4126%	-1.1487%	-0.0960%	-0.1552%	-0.0565%
1024x768	0.0000%	0.3983%	0.4067%	0.1836%	0.1670%	0.1430%
1920x1088	0.0000%	1.4768%	1.1120%	0.5586%	0.5458%	0.4832%
Average	0.0000%	1.0724%	0.8475%	0.4180%	0.4037%	0.3556%

Source: Author.

Table C.21 – BD-Rate variation according to the 3D-video sequences by using the iUDS under the Test Case 4 (UDS 4-2 + full refinement)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	0.0000%	0.1524%	0.2596%	0.0945%	0.0963%	0.0566%
Kendo	0.0000%	0.2428%	-0.1538%	0.0289%	0.0291%	0.0061%
Newspaper_CC	0.0000%	0.2470%	0.2112%	0.1164%	0.1090%	0.0350%
GT_Fly	0.0000%	1.1140%	1.0396%	0.3691%	0.3739%	0.2700%
Poznan_Hall2	0.0000%	1.8637%	1.0671%	0.6753%	0.6339%	0.5633%
Poznan_Street	0.0000%	1.0427%	1.3690%	0.3911%	0.3846%	0.3233%
Undo_Dancer	0.0000%	1.0488%	0.8795%	0.3573%	0.4099%	0.2868%
Shark	0.0000%	-0.2346%	-1.0003%	-0.0842%	-0.1464%	-0.1160%
1024x768	0.0000%	0.2141%	0.1057%	0.0800%	0.0781%	0.0325%
1920x1088	0.0000%	0.9669%	0.6710%	0.3417%	0.3312%	0.2655%
Average	0.0000%	0.6846%	0.4590%	0.2436%	0.2363%	0.1781%

Source: Author.

Table C.22 – BD-Rate variation according to the 3D-video sequences by using the iUDS under the Test Case 5 (UDS 12-4 without refinement step)

Sequence	Video 0	Video 1	Video 2	Video Only	Video Total	Synthesized
Balloons	0.0000%	2.9720%	3.6236%	1.3217%	1.2620%	0.8549%
Kendo	0.0000%	3.2785%	2.6801%	1.2071%	1.1174%	0.7409%
Newspaper_CC	0.0000%	3.7622%	8.3066%	2.4617%	2.2788%	1.7575%
GT_Fly	0.0000%	2.4013%	2.6036%	1.0351%	1.0931%	0.6835%
Poznan_Hall2	0.0000%	7.2381%	5.4676%	2.8249%	2.6908%	2.1790%
Poznan_Street	0.0000%	12.9342%	12.7752%	3.9473%	3.8302%	3.0250%
Undo_Dancer	0.0000%	2.1452%	2.4020%	0.9447%	1.0601%	0.8743%
Shark	0.0000%	1.0775%	0.6685%	0.4344%	0.4983%	0.3868%
1024x768	0.0000%	3.3376%	4.8701%	1.6635%	1.5527%	1.1178%
1920x1088	0.0000%	5.1592%	4.7834%	1.8373%	1.8345%	1.4297%
Average	0.0000%	4.4761%	4.8159%	1.7721%	1.7288%	1.3127%

Source: Author.

Table C.23 – BD-Rate variation according to the 3D-video sequences by using the iUDS under the Test Case 8 (UDS 12-4 + full refinement)

Sequence	**Video 0**	**Video 1**	**Video 2**	**Video Only**	**Video Total**	**Synthesized**
Balloons	0.0000%	0.5460%	0.7834%	0.2955%	0.2875%	0.2089%
Kendo	0.0000%	0.7863%	0.2575%	0.2417%	0.2332%	0.1295%
Newspaper_CC	0.0000%	1.3536%	1.6544%	0.6440%	0.6014%	0.4292%
GT_Fly	0.0000%	1.7452%	1.9356%	0.7626%	0.7982%	0.4630%
Poznan_Hall2	0.0000%	2.8343%	2.1075%	1.1305%	1.0715%	0.8913%
Poznan_Street	0.0000%	2.0740%	2.8851%	0.9051%	0.9075%	0.6714%
Undo_Dancer	0.0000%	1.6175%	1.4540%	0.6705%	0.7609%	0.5576%
Shark	0.0000%	0.7848%	0.4468%	0.3090%	0.3368%	0.2005%
1024x768	0.0000%	0.8953%	0.8984%	0.3937%	0.3740%	0.2559%
1920x1088	0.0000%	1.8112%	1.7658%	0.7555%	0.7750%	0.5568%
Average	0.0000%	1.4677%	1.4406%	0.6199%	0.6246%	0.4439%

Source: Author.

ANNEX A – LIST OF THE MAIN PUBLICATIONS DURING THIS DOCTORATE

AFONSO, V.; CONCEIÇÃO, R. A.; SALDANHA, M. R. F.; BRAATZ, L. A.; PERLEBERG, M. R.; CORREA, G. R.; PORTO, M. S.; AGOSTINI, L. V.; ZATT, B.; SUSIN, A. A. Energy-Aware Motion and Disparity Estimation System for 3D-HEVC with Run-Time Adaptive Memory Hierarchy. **IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT)**, v. 29, n. 6, pp. 1878–1892, 2019.

ZATT, B.; PERLEBERG, M. R.; SUSIN, A.; AFONSO, V.; PORTO, M.; CONCEIÇÃO, R.; AGOSTINI, L.. Energy and Rate-Aware Design for HEVC Motion Estimation Based on Pareto Efficiency. **JICS. JOURNAL OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 13, p. 1-12, 2018.

AFONSO, V.; SUSIN, A.; PERLEBERG, M.; CONCEIÇÃO, R.; CORREA, G.; AGOSTINI, L.; ZATT, B.; PORTO, M. Hardware-Friendly Unidirectional Disparity-Search Algorithm for 3D-HEVC. In: 2018 IEEE International Symposium on Circuits and Systems, ISCAS, 2018. **Proceedings...** Florence: IEEE, 2018.

PERLEBERG, M. R.; GOEBEL, J. W.; MELO, M. S.; AFONSO, V.; AGOSTINI, L. V.; ZATT, B.; PORTO, M. ASIC power-estimation accuracy evaluation: A case study using video-coding architectures. In: 9th Latin American Symposium on Circuits & Systems, LASCAS, 2018. **Proceedings...** Puerto Vallarta: IEEE, 2018.

UCKER, M.; AFONSO, V.; AUDIBERT, L.; SUSIN, A.; ZATT, B.; PORTO, M.; AGOSTINI, L. Low-Power and High-Throughput Architecture for 3D-HEVC Depth Modeling Mode 4. In: 31st Symposium on Integrated Circuits and Systems Design, SBCCI, 2018. **Proceedings...** Bento Gonçalves: ACM/IEEE, 2018.

PERLEBERG, M.; AFONSO, V.; CONCEIÇÃO, R.; SUSIN, A.; AGOSTINI, L.; ZATT, B.; PORTO, M. A Power-Efficient and High-Throughput Hardware Design for 3D-HEVC Disparity Estimation. In: 31st Symposium on Integrated Circuits and Systems Design, SBCCI, 2018. **Proceedings...** Bento Gonçalves: ACM/IEEE, 2018.

AFONSO, V.; CONCEIÇÃO, R.; PERLEBERG, M.; PORTO, M.; ZATT, B.; AGOSTINI, L. V.; SUSIN, A. Fast and Low-Power Hardware Design for HEVC Fractional Motion Estimation. **IEEE COMSOC MMTC Communications - Frontiers**, v. 12, p. 6-11, 2017.

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, ISCAS, 2017. **Proceedings...** Baltimore: IEEE, 2017.

AFONSO, V.; MAICH, H.; AUDIBERT, L.; ZATT, B.; PORTO, M. S.; AGOSTINI, L. V.; SUSIN, A. Hardware Implementation for the HEVC Fractional Motion Estimation Targeting Real-Time and Low-Energy. **JICS. JOURNAL OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 11, p. 106-120, 2016.

PAIM, G.; PENNY, W.; GOEBEL, J.; AFONSO, V.; SUSIN, A.; PORTO, M.; ZATT, B.; AGOSTINI, L. An efficient sub-sample interpolator hardware for VP9-10 standards. In: IEEE International Conference on Image Processing, ICIP, 2016. **Proceedings...** Phoenix: IEEE, 2016.

AFONSO, V.; MAICH, H.; AUDIBERT, L.; ZATT, B.; PORTO, M.; AGOSTINI, L. Memory-Aware and High-Throughput Hardware Design for the HEVC Fractional Motion Estimation. In: 28th Symposium on Integrated Circuits and Systems Design, SBCCI, 2015. **Proceedings...** Salvador: ACM/IEEE, 2015.

MAICH, H.; PAIM, G.; AFONSO, V.; AGOSTINI, L.; ZATT, B.; PORTO, M. A multi-standard interpolation hardware solution for H.264 and HEVC. In: IEEE International Conference on Image Processing, ICIP, 2015. **Proceedings...** Quebec City: IEEE, 2015.

MAICH, H.; PAIM, G.; AFONSO, V.; AGOSTINI, L.; ZATT, B.; PORTO, M. A multi-standard interpolation filter for motion compensated prediction on high definition videos. In: IEEE 6th Latin American Symposium on Circuits & Systems, LASCAS, 2015. **Proceedings...** Montevideo: IEEE, 2015.