Low Complexity HEVC Intra Coding

TESIS DOCTORAL
PRESENTADA AL DEPARTAMENTO DE
SISTEMAS INFORMÁTICOS DE LA
UNIVERSIDAD DE CASTILLA-LA MANCHA
PARA LA OBTENCIÓN DEL TÍTULO DE
DOCTOR POR LA UNIVERSIDAD DE CASTILLA-LA MANCHA

José Damián Ruiz Coll

Escuela Politécnica Superior de Albacete

Campus Universitario

Avenida de España s/n

02071 Albacete

DIRIGIDA POR

Gerardo Fernández Escribano y José Luis Martínez Martínez

Albacete, Noviembre de 2015

CONTENTS

AGRADECIMIENTOS

Me gustaría mostrar mi agradecimiento a todas las personas que han hecho posible este trabajo en estas breves líneas.

En primer lugar, quiero agradecer a mis tutores, Gerardo Fernández, José Luis Martínez y Pedro Cuenca, su ayuda, apoyo y dedicación durante estos años, así como el haberme permitido trabajar y hacerme participe de éste gran grupo de investigación. Por supuesto, a mis *supervisors* al otro lado del Atlántico, Hari Kalva y Velibor Adzic, sin su confianza e infatigable ayuda en la fase inicial de este trabajo no hubiera sido posible alcanzar este objetivo. Me gustaría agradecer también el apoyo a compañeros de la FAU, Ray García, del iTEAM, Jordi Joan y Jaime López, de RTVE, Eladio Gutiérrez y Javier Sánchez, de la UPM, Giuseppe, Natalia y Álvaro, de SAPEC, a Juanjo Anaya y Miguel Ángel Cristóbal, y en especial a Oscar Patiño, por su confianza y ayuda para resolver muchas dudas y dificultades que han surgido en el camino.

Y finalmente, pero no menos importante, a toda mi familia, a mi madre y hermanos por su aliento e infinita confianza en que podía cumplir este sueño, y por supuesto y muy especialmente a mi mujer, y a mis hijos, Andrea y Daniel, por haber sido mi motivación durante estos años para conseguirlo. Por último, a mi padre que desde ahí arriba se que estará muy orgulloso de este trabajo, y a quien me ha permitido firmar este trabajo.

RESUMEN

En la actualidad, los contenidos multimedia son consumidos sobre un gran número de dispositivos y terminales diferentes, desde Smart TVs a smartphones, ordenadores y tabletas. Este hecho es motivado en gran medida por la irrupción en los últimos años de tecnologías de compresión muy eficientes que han revolucionado el concepto de distribución, almacenamiento e intercambio de contenidos, respondiendo al denominado en la actualidad *mundo conectado*. Diversos estudios han confirmado que la distribución de vídeo se ha convertido en el principal tráfico de datos a nivel mundial, en especial sobre redes celulares e inalámbricas, y su volumen continua creciendo rápidamente, impulsado por el auge en el número de servicios y usuarios, pero también por el incremento de la resolución espacial y temporal que han experimentado los formatos de vídeo por encima de la *Alta Definición* (AD), tal como el nuevo formato de *Ultra Alta Definición* (UAD). El formato de UAD tiene una resolución espacial cuatro veces superior al del formato de AD, pero también duplica su resolución temporal, amplia significativamente la capacidad de representación de su colorimetría, e incrementa el rango dinámico de representación de los niveles muy bajos y muy altos de luminosidad.

Con el objetivo de proporcionar una solución a esta demanda, en junio del 2013 el ITU-T y el ISO/IEC aprobaron un nuevo estándar de compresión de vídeo conocido como *High Efficiency Video Codin* (HEVC). Evaluaciones científicas de calidad de vídeo han demostrado que el estándar HEVC puede reducir a la mitad la tasa binaria de su antecesor, el estándar H.264/AVC, ofreciendo la misma calidad perceptual. El elevado rendimiento de compresión de HEVC ha conseguido que este sea seleccionado como estándar para los futuros servicios de UAD por los dos organismos internacionales de radiodifusión de mayor implantación a nivel mundial, el *Digital Video Broadcasting* (DVB) y el *Advanced Television Systems Committee* (ATSC). Este hecho acelerará el desarrollo y expansión de HEVC en el mercado de consumo, provocando la rápida sustitución de H.264/AVC en servicios multimedia de uso masivo tales como el streaming, el VoD y los servicios *Over the Top* (OTT).

Sin embargo, la alta eficiencia de compresión de HEVC tiene como contrapartida una elevada complejidad computacional comparada con la de otros estándares, lo cual es un

impedimento para su rápida adopción en el mercado profesional. En los últimos años, la comunidad científica ha llevado a cabo un considerable esfuerzo para abordar este problema, y se han presentado numerosas propuestas para la aceleración de las diferentes herramientas de HEVC. Esta tesis se enmarca en este campo de investigación, centrado en el diseño de algoritmos de baja complejidad para la reducción del tiempo de codificación de HEVC, presentando varias propuestas para la codificación *intra-frame* basado en técnicas alternativas a las utilizadas habitualmente en los estándares de compresión de vídeo, tales como el *Machine Learning (ML)* y los algoritmos de procesado de imágenes.

La primera propuesta presentada en esta tesis es un algoritmo de baja complejidad para la decisión del particionado de la unidad de codificación, la cual es la tarea más intensa en la predicción *intra-frame* en HEVC. El algoritmo propuesto está basado en herramientas de ML las cuales explotan las características de la textura de la imagen y obtienen un árbol de decisión para clasificar la unidad de codificación como una clase *Split* o *Non-Split.* Este enfoque usa un esquema *Top-Bottom* con tres niveles o nodos, compuestos por un árbol de decisión en cada nodo. La arquitectura propuesta reduce la complejidad de la predicción *intra-frame* por encima del 50%, sin pérdidas de calidad en términos de PSNR, y con una penalización de la tasa binaria de tan solo el 2%.

Como segunda propuesta, se presenta un algoritmo de detección de la orientación de la textura para la decisión rápida del modo de predicción *intra-frame* en HEVC, el cual calcula la *Mean Directional Variance* (MDV) a largo de un conjunto de co-lineas. El elemento clave de esta propuesta se sustenta en la hipótesis de que la correlación de los píxeles es máxima en la dirección de la textura dominante, y por lo tanto la varianza calculada en dicha dirección será muy baja. Una característica destacable de esta propuesta es el uso de un conjunto de pendientes racionales, las cuales son definidas exclusivamente en posiciones enteras del espacio discreto Λ, por lo que no es preciso llevar a cabo interpolación entre píxeles. En esta tesis también se presenta una mejora al algoritmo de MDV, el cual introduce el concepto de ventana deslizante (MDV-SD). Este nuevo enfoque mejora significativamente el rendimiento del algoritmo para elevados QP, al considerar la distorsión sobre los pixeles de referencia en el cálculo de la MDV. Los resultados de las simulaciones llevadas a cabo muestran que se obtiene una reducción de la complejidad computacional próxima al 30%, con una penalización de tasa binaria despreciable del 0.4%.

Por último, ambas propuestas son combinadas creando un algoritmo rápido de predicción *intra-frame* denominado *Fast Partitioning and Mode Decision* (FPMD), el cual se considera una solución flexible de baja complejidad al permitir ser implementado usando cualquier combinación de nodos, obteniendo un amplio rango de aceleraciones entre el 44% y el 67%, con un aligera penalización en la tasa binaria entre el 1.1% y el 4.6%.

ABSTRACT

Today, media content is consumed on a huge number of different devices and terminals from smart TVs and smartphones, to computers and tablets. This has been motivated by the emergence, over the last few years, of high efficiency digital compression technologies that have revolutionized the concept of content delivery, storage and exchange, answering the needs of today's connected online world. According to various studies, video delivery now makes up the majority of network traffic world-wide, especially over wireless and cellular networks, and its volume will continue to grow at a very fast pace, driven by the ever larger number of services and users, but also by the increasing spatial and temporal resolution of video formats beyond High Definition (HD), such as the new Ultra-HD format. The UHD format increases the spatial resolution of HD by two, involving also a higher temporal resolution, greater colour fidelity, and a higher dynamic range.

To address these special requirements, in January of 2013, a new video coding standard was approved from the ITU-T and ISO/IEC organizations known as *High Efficiency Video Coding* (HEVC). Exhaustive quality assessments have proved that HEVC can reduce the bit rate of its predecessor, H.264/AVC, by half for the same perceptual quality.

The high performance of HEVC has led to this standard being adopted for future TV services using the new UHD format by the two worldwide broadcasting organisations, *Digital Video Broadcasting* (DVB) and the *Advanced Television Systems Committee* (ATSC). This fact will probably accelerate HEVC's expansion and development, prompting a rapid replacement of the H.264/AVC standard in the consumer market and in massive multimedia services such as video streaming, VoD and *Over the Top* (OTT).

However, the high compression performance of HEVC comes at the expense of a huge computational complexity increase compared with other codecs, which is hindering the rapid adoption of HEVC by the professional market. In the last few years, the research community has made a considerable effort to address this problem, and a number of new approaches and algorithms have been proposed for speeding up the different HEVC tools. This thesis lies within this research area, mainly focusing on low complexity algorithm design for HEVC time coding reduction, presenting several approaches for *intra-prediction* coding, based on non-traditional techniques used in the video coding standards, such as *Machine Learning* (ML) and image processing algorithms.

The first approach proposed in this thesis is a low complexity algorithm for the coding unit partitioning decision, which is the most computationally intensive task in the *intra-prediction* process in HEVC. The proposed algorithm is based on *ML* tools, which exploit the content texture attributes and derive a set of decision trees to classify the coding unit into *Split* or *Non-Split* classes. This approach uses a *Top-Bottom* scheme with three depth levels or nodes, comprising a decision tree in each node. The proposed architecture reduces the *intra-prediction* complexity by over 50%, with no quality penalty in terms of the PSNR, and near a 2% of bit rate increase, compared to the HEVC reference model.

As a second approach, a novel texture orientation detection algorithm is proposed for the fast *intra-prediction* mode decision in HEVC, which computes the *Mean Directional Variance* (MDV) along a set of *co-lines*. The key point of the proposed algorithm is based on the hypothesis that pixel correlation is maximum in the dominant texture orientation, and consequently the variance computed in that direction will obtain a low value. A noteworthy feature of this proposal is the use of a set of rational slopes, which are exclusively defined in an integer position of the discrete lattice Λ; thus no pixel interpolation is required. An enhancement to the MDV algorithm is also presented in this thesis, which introduces the concept of *Sliding Window* (MDV-SW). This approach significantly improves the algorithm's performance for high QPs, because the distortion over the reference pixels is considered in the MDV computation. Simulation results of MDV-SW report a complexity reduction near 30%, with a negligible rate penalty of 0.4%.

Finally, the two proposals presented in this thesis, the fast partitioning based on *ML* techniques and fast mode decision based on the MDV-SW algorithm, are combined, creating an overall fast *intra-prediction* algorithm, called *Fast Partitioning and Mode Decision* (FPMD), which achieves a very high overall performance. This approach arises as a flexible low complexity solution because it can be implemented using any combination of nodes, obtaining a wide range of speed-ups, from 44% to 67%, and light penalties from 1.1% to 4.6%.

GLOSSARY

ADP	Angular Directional Prediction
AMVP	Advanced Motion Vector Prediction
ASP	Advanced Simple Profile
ATSC	Advanced Television Systems Committee
BDA	Blu-ray Disc Association
CABAC	Context Adaptive Binary Aritmetic Coding
CAVLC	Context-Adaptive Variable Length Coding
CB	Coding Block
CIF	Common Intermediate Format (352x288 pixels)
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
DCT	Discrete Cosine Transform
DEA	Dominant Direction Assents
DFT	Discrete Fourier Transform
DM	Data Mining
DPB	Decoded Picture Buffer
DSAD	Directional Sum of Absolutes Differences
DST	Discrete Sine Transform
DVB	Digital Video Broadcasting

DVD	Digital Versatile Disc
DWT	Directional Wavelet Transform
FEOD	Four Edge Orientation Detection
FME	Fractional Motion Estimation
FPM	Fractional Position Modes
FPMD	Fast Partitioning and Mode Decision
FRext	Fidelity Range Extensions
GOP	Group Of Pictures
GPB	Generalized P and B
GPU	Graphics Processor Unit
H.261	A video coding standard
H.263	A video coding standard
H.264	A video coding standard
H.265	A video coding standard
HD	High Definition
HEVC	High Efficiency Video Coding
HOG	Histogram of Oriented Gradients
HT	Hadamard Transform
HVS	Human Visual System
IDR	Instaneous Decoding Refresh
IME	Integer Motion Estimation
IP	Internet Protocol
IPM	Integer Position Modes
ISDN	Integrated Services Digital Network
ISO/IEC	International Organization for Standardization and International Electrotechnical Commission
ITU	International Telecommunication Union
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
JCT-VC	Joint Collaborative Team Video Coding
KLT	Karhunen/Loeve Transform
LCTD	Limited Coding Tree Depth
LUT	Look-Up Table
MAD	Mean of Absolute Differences
MC	Motion compensation
MDV	Mean Directional Variance
MDV-SW	Mean Directional Variance with Sliding Window
ME	Motion Estimation
MPEG	Motion Pictures Expert Group
MPEG-1	A multimedia coding standard
MPEG-2	A multimedia coding standard
MPEG-4	A multimedia coding standard

MPM	Most Probable Mode
MPS	Most Probable Symbol
MV	Motion Vector
MVC	Multiview Video Coding
NAL	Network Adaptation Layer
OTT	Over-The-Top
PB	Prediction Block
POC	Picture Order Count
PU	Prediction Unit
QCIF	Quarter Common Intermediate Format (176x144 pixels)
QP	Quantification Parameters
RDO	Rate Distortion Optimization
RExt	Range Extensions
RMD	Rough Mode Decision
RPS	Reference Picture Set
RQT	Residual Quad Tree
SAD	Sum of Absolute Differences
SAO	Sample Adaptive Offset
SI	Spatial Index
SD	Standard Definition
SHVC	Scalable HEVC Video Coding
SPS	Sequence Parameter Set
SSE	Sum of Square Error
TB	Transform Block
TCP	Transmission Control Protocol
TI	Temporal Index
TU	Transform Unit
UHD	Ultra High Definition
VCL	Video Coding Layer
VLC	Variable Length Coding
VoD	Video on Demand
SVC	Scalable Video Coding
WP	Weighted Prediction
WPP	Wavefront Parallel Processing

FIGURE INDEX

TABLE INDEX

CHAPTER 1

INTRODUCTION, MOTIVATION, AND OBJECTIVES

1.1 INTRODUCTION

Today, media content is consumed on a huge number of different devices and terminals from smart TVs and smartphones, to computers and tablets. This has been motivated by the emergence, over the last few years, of high efficiency digital compression technologies that have revolutionized the concept of content delivery, storage and exchange. At the same time, the use of video services such as VoD, video streaming and personal video communications, is becoming common among users, answering the needs of the today's connected online world.

According to various studies, video delivery now makes up the majority of network traffic world-wide, especially over wireless and cellular networks, and its volume will continue to grow at a very fast pace, driven by the ever larger number of services and users.

Nevertheless, this traffic increase is also due to the spatial and temporal resolution of the new video formats, and by the demand for a better quality experience.

The increasing resolution phenomenon has experienced an exponential growth in the last decade, driven by the new capabilities of the acquisition terminals, mainly video cameras, smartphone cameras and even sport cameras. These devices can support resolutions from the *Standard Definition* (SD) format comprising nearly 0.5 Megapixels per picture, through the *High Definition* (HD) format with 2 Megapixels, up to the current *Ultra High Definition* (UHD) format with over 8 Megapixels, which will soon be surpassed by the forthcoming 8k format, which quadruples the previous resolution of the UHD format.

From a network point of view, this formats evolution demands higher bandwidth networks, which has led to the quick development of a new generation of wireless and cellular networks such as 3G and 4G, but also to a new generation of broadcasting technologies for satellite, terrestrial and cable networks, known as DVB-X2. At the same time as this network evolution, the advances in video coding technologies have played a fundamental role in the media revolution. The MPEG-2 video coding standard was the driver for the development of Digital TV and media exchange on DVDs in the 90s. this standard allowed the delivery of TV services using the SD format at a bit rate of 4Mbps. On the other hand, the H.264/AVC video coding standard, approved in 2003, has become the main driver for supporting the HD format on a wide range of networks and services with average bit rates in the range of 4Mbps to 10Mbps. The success of H.264/AVC standard is undoubted, and it has become the mass-market codec, giving support to the current streaming, broadcasting and IPTV services, and being used worldwide on billions of mobile devices.

The recent arrival of UHD technology brings with it not only a spatial resolution increase, but also rather a general quality enhancement, involving a higher temporal resolution, a greater colour fidelity, known as WCG (Wider Colour Gamut) and a higher dynamic range, known as HDR (High Dynamic Range). To address these special requirements, in 2010 the scientific community joined their forces for the development of a new generation of video compression standard, with the aim of paving the way for new UHD services. That research was concluded in 2013 with a new video coding standard, known as *High Efficiency Video Coding* (HEVC), which provides about a factor of two improvements in compression efficiency compared to its predecessor, the H.264/AVC standard, for a comparable visual quality. The objective in the long term is that HEVC will provide UHD services using the same bandwidth that H.264/AVC needs to compress HD formats.

The increased coding efficiency needed to achieve such performance is obtained by introducing new coding tools as well as by improving certain algorithms already used in previous standards. Some of the most noteworthy tools introduced in HEVC are: the new

quadtree block partitioning which covers a wide range of block sizes, a high density of angular modes for *intra-coding*, a larger number of transform and prediction sizes, a new additional in-loop filter for perceptual quality improvement, and an advanced motion vector mode for *inter-coding* that enables neighbouring blocks to share the same motion information, among others.

Nevertheless, the new coding tool do not provide such high efficiency by themselves; on the contrary, high efficiency can only be achieved if an *optimal decision* is taken in every stage and tool of the encoder, i.e. the number and sizes of the sub-blocks into which a block has to be quadtree split, and for each of such sub-blocks of the quadtree, the optimal transform size that should be used, the optimal angular mode for each prediction unit, or the optimal motion vector used in the *motion-estimation* stage. Consequently, the implementation of an *optimal decision* is the key element in the architecture of the encoder which makes it possible to achieve the best encoding performance.

Traditionally, that task has been carried out by an exhaustive evaluation of all available parameter combinations, and, by using a cost function, the combination with the best performance, that is the best quality with the lowest distortion, is selected as the *optimal decision.* However, the huge number of combinations available in the new standard makes this *optimal decision* based on brute force an impractical solution for most of the applications in this field, due to the extremely high computational complexity required.

In the last few years, the research community has made a considerable effort to address this problem, and a number of new approaches and algorithms have been proposed for speeding up the different HEVC tools. This thesis lies within this research area, mainly focusing on low complexity algorithm design for HEVC coding time reduction, presenting several novel approaches for *intra-prediction* coding, as will be shown in the following chapters.

1.2 MOTIVATION

Over the last few decades, much research has focused on the development and optimization of video codecs for media distribution to end-users via the Internet, broadcasts or mobile networks, but also for videoconferencing and for the recording on optical disks for media distribution. Most of the video coding standards for delivery are characterized by using a high efficiency hybrid schema, based on *inter-prediction* coding for temporal picture decorrelation, and *intra-prediction* coding for spatial picture decorrelation. As is well known, this schema achieves a high performance at the expense of several drawbacks, such as a high coding latency, low stream robustness, and a high coding complexity due to the *motion-estimation* process of the *inter-prediction*, among others.

Nevertheless, high efficiency video and image codecs are also required in other fields with quite different requirements, such as still-picture photography storage, live TV interviews where low latency is needed for natural communication between the interlocutors, and also for professional edition and post-production tasks commonly used in the TV and cinema industries, where high quality and fast access to the individual pictures are required. These production codecs, named *mezzanine* codecs, are highly relevant for the audio-visual industry, which demands two things: very high compression efficiency and a low computational burden.

A high video coding performance allows a reduction in the storage space for archiving applications, and also a long recording capability on small physical supports, i.e. optical disks and solid state memories. Regarding low computational complexity, this is mainly motived by the fact that these *mezzanine* codecs are used on portable devices, mainly camcorders, and thus one wishes that the codec can be implemented on low cost processors, and just as importantly, with low power consumption, allowing a long operational autonomy.

The latest video coding standard was approved in January of 2013, and is the product of the collaboration between the ITU-T and the ISO/IEC international organizations grouped into the Joint Collaborative Team Video Coding group (JCT-VC). This standard is officially called *Recommendation ITU-T H.265* and *ISO/IEC 23008-2*, and informally known as HEVC. Exhaustive quality assessments have proved that HEVC can reduce the bit rate of its predecessor, namely H.264/AVC, by half for the same perceptual quality.

The high compression efficiency of HEVC has led to this standard being adopted for future TV services using the new UHD format by the two worldwide broadcasting organisations, *Digital Video Broadcasting* (DVB) and the *Advanced Television Systems Committee* (ATSC). The Blu-ray Disc Association (BDA) has also adopted it in other application fields, such as for the next generation of *Ultra-HD Blu-ray*. This fact will probably accelerate HEVC's expansion and development, prompting a rapid replacement of the H.264/AVC standard in the consumer markets and in massive services such as video streaming, VoD and OTT (Over the Top).

HEVC is also considered by the industry as the best candidate to replace the current *mezzanine* compression codecs, due to the high performance of the novel HEVC *intra-prediction* coding. This schema significatively improves H.264/AVC's performance, mostly by using a high density of angular predictors. For this reason, the HEVC standard has approved a set of specific profiles which exclusively use the *intra-prediction* schema, known as "*Main still Picture*" and the "*Main Intra*" profiles, with support for different bit-depths and chroma sampling formats.

However, the high compression performance of HEVC comes at the expense of a huge computational complexity increase compared to other codecs, which is hindering the rapid adoption of HEVC by the professional market. The speeding up of the *intra-prediction* coding can be achieved by applying advanced techniques that allow the taking of decisions that are needed in the different stages of *intra-prediction* with low complexity. This approach constitutes the basis of this thesis, which addresses the complexity reduction of HEVC intra-coding by using non-traditional techniques used in the video coding standards, such as *Machine Learning* and image processing algorithms for the texture orientation detection.

1.3 OBJECTIVES

In this dissertation, four main objectives have been proposed for the computational reduction of the real time implementation of the *intra-prediction* coding in HEVC, as following:

1. *Study of state-of-the-art of HEVC and fast intra-prediction approaches*. As an initial aim we have defined an in-depth study of the HEVC architecture and the new coding tools introduced in this standard, paying special attention to the performance differences between H.264/AVC and HEVC. One of the main objectives is the detailed study of the different algorithms that comprises the *intra-prediction* in HEVC, and the analysis of the computational complexities required for these. Then, we will conduct a detailed study of the most relevant approaches proposed in the literature for fast partitioning and mode decision in *intra-prediction*, with particular attention to the algorithms already adopted in the *Rate Distortion Optimization* (RDO) stage of the test model proposed in the HEVC reference software. The conclusion of this study will constitute the basis for the design of the efficient low complexity algorithms proposed in this thesis.

2. *Development and evaluation of a fast partitioning decision algorithm*. Being aware that the partitioning of the coding units is the most critical decision that the encoder has to take in terms of computational burden but also regarding the impact on quality, the second objective proposed in this thesis is the complexity reduction of the partitioning decision. With the aim of tackling this task with high efficiency, the use of the *Machine Learning* techniques for the design of a decision tree will be studied. Special efforts will be made in the training stage of the decision trees, selecting the optimal number and type of attributes, allowing a high precision classifier with the minimum computational cost. It required that the architecture of the proposed approach permit a scalable implementation with different decision nodes, achieving

different levels of speed-up, and as a counterpart, different levels of performance reductions compared to the HEVC reference software.

3. *Development and evaluation of a fast mode decision algorithm*. With the aim of exploiting the strong correlation between the texture orientation of the image and the angular modes defined in HEVC, a fast mode decision algorithm based on texture orientation will be proposed and developed. In order to achieve this objective, an in-depth study of the different texture and edge detection techniques published in the literature will be carried out. The proposal is based on the computation of the *Mean Directional Variance* (MDV) along a set of *co-lines*, and it will be highly efficient for the detection of the dominant gradient in all the range of coding unit sizes, from 64x64 to 4x4 pixels. The concept of *Sliding Window* for the improvement of the MDV algorithm, denoted as MDV-SW, will be also presented in this thesis. The aim of the MDV-SW approach will be to achieve a significant speed-up of *intra-prediction* coding by reducing the number of modes to be evaluated, with a very low performance penalty.

4. *Combination of both proposals in a full fast intra-prediction algorithm*. As a last aim, the integration of both fast decision approaches in a unified architecture, called *Fast Partitioning and Mode Decision* (FPMD), is proposed. The performance evaluation of the combined proposal will allow knowing the best performance that can be achieved by speeding up both decisions. The analysis of the results will also show the mutual inference that exist between the wrong classification decisions of both algorithms in terms of bit rate penalty and quality degradation. That is, how a wrong mode decision affects the overall encoding performance when a wrong partitioning decision is taken, or vice versa, how a wrong partitioning decision affects the encoding performance when a wrong mode decision is taken.

1.4 THESIS ORGANIZATION

This thesis has been structured into the following chapters:

- *Chapter 1*. This chapter presents the framework of the thesis, describing the motivation, and setting the main objectives proposed in this thesis.

- *Chapter 2*. A general review of video coding standards is presented in this chapter, with special emphasis on the HEVC architecture and the coding tools, which constitutes the *state-of-the-art* in video coding. An in-depth description of the HEVC *intra-prediction* coding architecture is given, comparing it to the H.264/AVC

architecture, and reporting the computational complexity of both standards. This chapter concludes with a description and analysis of the fast *Rate Distortion Optimization* architecture implemented in the HEVC reference software.

- *Chapter 3*. In this chapter, a review of the most relevant approaches proposed by the scientific community is presented. The architecture and design of a fast CTU partitioning algorithm based on *Machine Learning* techniques is described. Finally, experimental results and a comparison to other fast partitioning approaches are also shown, proving the high performance of the proposed algorithm.

- *Chapter 4*. In the scenario of the *intra-prediction*, the different approaches proposed in the literature for the complexity reduction of the mode decision is summarized in this chapter. The design of a novel architecture, MDV, for the fast mode decision is described. The performance of this algorithm based on the directional variance computation along the co-lines, and an improved version of that by using a new sliding window approach, MDV-SW, is reported. To conclude, a comparison to other fast partitioning approaches is shown.

- *Chapter 5*. Using the two approaches of fast partitioning and fast mode decision proposed in the previous chapters, a full implementation of a fast *intra-prediction* approach is proposed and evaluated. The implementation details and the simulation results of the overall architecture are reported.

- *Chapter 6*. This chapter summarizes the novel algorithms proposed in this thesis for the complexity reduction of the *intra-prediction* in HEVC, highlighting the main conclusion derived from the results obtained. The publications resulting from the studies carried out in this thesis are also presented, and the lines for future research are finally proposed as a starting point for the development of the new approaches in *intra-prediction* coding.

- The bibliography and references used in the elaboration of this thesis are finally presented.

CHAPTER 2

OVERVIEW OF THE HEVC VIDEO CODING STANDARD

Este capítulo comienza presentando una revisión histórica de los distintos estándares de codificación de vídeo, para a continuación profundizar en la arquitectura, funcionalidades y prestaciones del último estándar aprobado, denominado HEVC, en el cual se centra esta tesis. Se hará especial hincapié en las novedades que este estándar introduce, como el nuevo modelo de particionado de la unidad de codificación basado en *quadtrees*, las novedades que incorporan el resto de etapas de predicción *intra-frame*, predicción *inter-frame*, codificación transformacional y codificación entrópica. A continuación se describen los detalles de la codificación *intra-frame* en HEVC, donde se abordan las particularidades de la selección y filtrado de las muestras de referencia utilizadas para la construcción de los predictores, las estrategias adoptadas para la mejora de la calidad perceptual, así como una estimación de la complejidad computacional requerida por este esquema de codificación. Por último, se describe la arquitectura y algoritmos utilizados en la implementación del software de referencia de HEVC.

2.1 INTRODUCCIÓN A LOS ESTÁNDARES DE COMPRESIÓN DE VÍDEO

En las últimas décadas, diversos estándares de compresión de vídeo han sido aprobados por los dos organismos internacionales competentes en este ámbito, el *International Telecommunication Union*, en su sección de *Telecommunications* (ITU-T) y el *International Organization for Standardization and International Electrotechnical Commission* (ISO/IEC). En ciertas ocasiones, estos organismos han aprobado sus estándares de modo independiente como son las recomendaciones H.261 [ITU-T H.261] y H.263 [ITU-T H.263] del ITU, y los estándares MPEG-1 [ISO-11172] y MPEG-4 Parte 2 [ISO-14496-2] del ISO-IEC.

Sin embargo, en las tres ocasiones que ambas organizaciones han colaborado para el desarrollo conjunto de un estándar, éstos han tenido un éxito indudable en toda la industria audiovisual, como ha ocurrido con los estándares MPEG-2 [ISO-13818-2] [ITU-T H.262], el conocido como H.264/AVC, cuyo nombre deriva de la denominación oficial de dicho estándar en el ITU-T como H.264 [ITU-T H.264] y en el ISO/IEC como MPEG-4 AVC [ISO-14496-10]; y el estándar recientemente aprobado por el ITU como H.265 [ITU-T H.265] y por el ISO/IEC como MPEG-H [ISO-23008-2]. Sin embargo este último estándar es ampliamente referenciado en la literatura como *High Efficiency Video Coding* (HEVC) y en esta tesis se utilizará HEVC como notación para este estándar.

Todos los estándares mencionados anteriormente comparten un mismo objetivo: reducir la tasa binaria necesaria para transmitir o almacenar una secuencia de vídeo con las mínimas pérdidas perceptuales posibles. A pesar de los más de 25 años que han transcurrido entre el primer y último estándar, todos ellos utilizan el mismo conjunto de herramientas de codificación que aplican una reducción de la alta correlación espacial y temporal que presentan las imágenes, o frames, de la secuencia de vídeo, y técnicas de codificación perceptual que explotan características del sistema de percepción visual humano como son las características cromáticas, la agudeza visual en determinadas orientaciones o el enmascaramiento entre componentes espectrales de distinta energía.

Estas herramientas se estructuran en una arquitectura que se denominada *híbrida predictivo-transformacional*, utilizada por la totalidad de los estándares de compresión de vídeo anteriormente mencionados. Estos codificadores en ocasiones son referenciados como codificadores con pérdidas matemáticas o, desde otra perspectiva, como codificadores sin pérdidas perceptuales. En lo concerniente a la etapa predictiva en el dominio espacial, denominada codificación *intra-frame*, ésta se implementa por medio de sencillas técnicas predictivas de primer orden, mientras que la predicción temporal, denominada codificación *inter-frame*, se aplican las técnicas de *Motion Estimation* (ME) y *Motion Compensation* (MC).

Las técnicas de ME tienen como objetivo la búsqueda de uno o varios bloques de píxeles que correlen fuertemente con el bloque bajo codificación, para ser utilizados como predictor temporal. La búsqueda se puede llevar a cabo sobre frames ya codificados, o sobre frames posteriores al actual frame, en orden de visualización. Una vez identificado ese bloque o bloques, éstos son identificados por su vector o vectores de movimiento, denominados *Motion Vector* (MV), y por el índice temporal del frame al que pertenece, con el objetivo de poder ser enviado al decodificador y que éste pueda llevar a cabo la MC.

La etapa transformacional es aplicada al residuo obtenido como resultado de las etapas predictivas, con la finalidad de convertir la imagen al dominio frecuencial donde se pueden aplicar sencillas técnicas de cuantificación que permitan obtener una ganancia de compresión. Entre las transformaciones propuestas en la literatura para la codificación de imágenes [Rao76], cabe destacar la transformada de *Hadamard*, la transformada *Discrete Fourier Transform* (DFT), la *Karhunen-Loeve Transform* (KLT) y la *Discrete Cosine Transform* (DCT) entre otras. Esta última es la elegida de modo casi generalizado por el conjunto de estándares de codificación de vídeo al ofrecer un buen balance entre complejidad y compactación energética. La eficiencia de la DCT es considerada muy próxima a la KLT, y ha demostrado ampliamente ser una transformación óptima para imágenes con alta correlación espacial [Rao74].

Otra característica destacable de este tipo de transformadas bidimensionales para su utilización en la codificación de imágenes, es su característica de *ortonormalidad*, que permite que su implementación se pueda llevar a cabo de modo separable como dos transformaciones 1D, reduciendo significativamente el número de operaciones matemáticas necesarias para su cómputo [Bha97], y por otro lado son transformaciones invertibles al conservar su energía, permitiendo su reconstrucción sin pérdidas en ausencia de cuantificación.

La DCT es considerada como una transformación expansiva, ya que su representación en el dominio transformado requiere de una representación de sus coeficientes con mayor rango dinámico que el de entrada a la transformación, y hasta la aprobación del estándar H.264/AVC, su cómputo se define con aritmética en coma flotante. La compresión en estos esquemas de codificación híbrida se introduce realmente en la etapa de cuantificación, que provoca que muchos de los coeficientes transformados sean nulos, generando las pérdidas matemáticas que no permiten una reconstrucción perfecta de la imagen en el decodificador. Conforme han ido evolucionando los estándares de compresión, estos han ido introduciendo distintas opciones de cuantificación perceptual por medio de matrices de cuantificación no lineales, las cuales permiten cuantificar más severamente las componentes frecuenciales menos sensibles para el HVS, obteniendo fuertes ratios de compresión.

La arquitectura de codificación *híbrida* incluye habitualmente como última etapa un codificador estadístico o entrópico que lleva a cabo la compresión sin pérdidas de los coeficientes cuantificados, así como del resto de los elementos sintácticos generados en la codificación que son imprescindibles para la decodificación de la secuencia de vídeo. El codificador entrópico más utilizado en los distintos estándares ha sido el codificador de longitud variable, denominado *Variable Length Coding (VLC),* que en ocasiones es denominado en la literatura como codificador *Huffman,* el cual permite una implementación de muy baja complejidad por medio de tablas o *Look-Up Table (*LUT). Sin embargo los últimos estándares han ido incorporando primero como opción (H.264/AVC) y después como normativos (HEVC), un codificador aritmético que mejora la eficiencia del VLC, pero que requiere de un incremento sustancial de la carga computacional.

En 1988, el estándar ITU-T H.261 fue el primero en utilizar este tipo de arquitecturas en su modalidad más básica debido a las limitaciones computacionales de aquella época. Su objetivo era proveer servicios de videoconferencia para formatos de muy baja resolución sobre las redes digitales conmutadas, que proporcionaban anchos de bandas múltiplos de 64Kbps, con velocidades máximas de 1920Kbps, conocidas como redes ISDN (*Integrated Services Digital Network*). H.261 utiliza como unidad de codificación bloques de 16x16 píxeles, denominados *Macroblocks* (MB), definición que perdurará durante 25 años hasta la aprobación del estándar HEVC con una ME muy sencilla que permite un único MV exclusivamente para posiciones enteras, y una DCT de tamaño 8x8.

Ese mismo año, pero promovido por el *Motion Pictures Expert Group* (MPEG) perteneciente al ISO/IEC, se aprobaba el estándar MPEG-1 para la codificación de contenidos multimedia sobre dispositivos ópticos, conocidos como CD Video, con un límite máximo en su tasa binaria de 1.5Mbps. MPEG-1 define herramientas de codificación muy similares a las utilizadas por H.261, empelando la misma estructura de MB y DCT, pero mejoraba la ME permitiendo la utilización de posiciones con precisión fraccionaria de ½ de píxel, e introducía la predicción temporal bidireccional.

Sin lugar a dudas, con la aprobación en 1996 del estándar MPEG-2, de modo conjunto por ambos organismos internacionales, ITU-T e ISO/IEC, se inicia una revolución multimedia que alcanza hasta nuestros días. MPEG-2 ha sido el propulsor de servicios y productos tan exitosos como la TV digital o el DVD, y ha sido utilizado como referencia para la evolución de los siguientes estándares.

En lo concerniente a las herramientas de codificación utilizadas por MPEG-2, sus diferencias con respecto a MPEG-1 son mínimas. MPEG-2 mantiene como unidad de codificación el MB, con transformación por DCT de 8x8, y ME con precisión de ½ de píxel, introduciendo ligeras mejoras en la codificación entrópica y la flexibilidad en la utilización de las matrices

de cuantificación. MPEG-2 introduce mejoras en aspectos operativos, como los formatos de vídeo soportados, permitiendo resoluciones de *Standard Definition* (SD) y HD con escaneo entrelazado, y mayores tasas binarias. Pero sin lugar a dudas MPEG-2 puede ser considerado como un estándar adelantado a su tiempo al introducir herramientas de escalabilidad espacial, temporal y de calidad, así como la codificación multivista, herramientas que por su anticipación a las capacidades de las redes y capacidades de procesado, nunca llegaron a implementarse en productos comerciales en el mercado.

Una vez ratificado MPEG-2, ambos organismos trabajaron en el desarrollo de estándares de codificación de modo independiente. El ITU-T aprobó ese mismo año el estándar H.263, concebido como una evolución del H.261, enfocado a servicios de videocomunicaciones personales de muy baja tasa binaria y que introdujo innumerables mejoras en sus distintas versiones H.263+ y H.263++, especialmente en la estimación de movimiento, que años más tarde constituyó los pilares del estándar H.264/AVC. En paralelo, el grupo MPEG trabajó en el desarrollo de un estándar para el entorno multimedia, a priori ciertamente novedoso, al proponer la codificación independiente de los objetos que componen la imagen. La primera versión de este estándar fue aprobada en 1999, conocido como MPEG-4 Visual, pero su implantación comercial fue un fracaso debido a las dificultades que conlleva la segmentación de los objetos de la escena, a lo costoso de la descripción sintáctica de dichos objetos en términos de tasa binaria, y a la imposibilidad de su implementación en tiempo real. Sin embargo, MPEG-4 Visual incluyó un perfil denominado *Advanced Simple Profile* (ASP) [Sto01] cuyo enfoque es el tradicional, basado en la codificación por bloques de la imagen, y cuya aportación más destacable fue la introducción de la ME con precisión fraccionaria de ¼ de píxel, la cual fue adoptada años después en el estándar H.264/AVC.

La irrupción de los servicios multimedia en Internet y la proliferación de dispositivos que soportaban formatos de HD, fueron los detonantes para que el ITU-T e ISO/IEC unieran de nuevos sus fuerzas en el *Joint Video Team Video Coding* (JVT-VC) para el desarrollo de un nuevo estándar de vídeo que mejorara la eficiencia de los estándares previos, adecuándose a las particularidades de las nuevas redes de comunicaciones basadas principalmente en protocolos TCP/IP. Utilizando como base para su desarrollo las herramientas definidas en los estándares ITU-T H.263 y MPEG-4 Visual, en 2003 fue aprobado el estándar H.264/AVC, el cual, a pesar de mantener el mismo esquema de codificación *híbrido* de su predecesor, puede ser considerado como una segunda generación de codificadores al introducir innumerables novedades respecto a sus predecesores.

H.264/AVC establece dos capas independientes para adaptarse mejor a las distintas topologías de red, definiendo una capa de codificación denominada *Video Coding Layer* (VCL), y una capa de red denominada *Network Adaptation Layer* (NAL) [Wie03]. H.264/AVC introduce por primera vez una transformada entera de tamaños 8x8 y 4x4, como

una aproximación de las funciones base de la DCT, permitiendo la implementación de los decodificadores con aritmética entera de 16 bits. Además, define una predicción *intra-frame* avanzada, más allá de la simple predicción de los coeficientes de DC entre bloques vecinos utilizada hasta entonces. De igual modo, introduce mejoras sustanciales en la codificación *inter-frame* con predicciones de tamaño rectangular, la utilización de múltiples frames de referencia, mejoras en los filtros de interpolación fraccionaria y el uso de modos de alta eficiencia como el modo *Skip* y *Direct* [Tou05]. Y como se describió anteriormente, H.264/AVC incorpora como opción la utilización de un modelo muy eficiente de codificación entrópica basado en la codificación aritmética adaptativa denominada *Context Adaptive Binary Aritmetic Coding* (CABAC) [Mar03]. La estandarización de H.264/AVC se extendió durante más de 6 años con la aprobación de perfiles profesionales, denominados *Fidelity Range Extensions* (FRext) [Mar05], perfiles escalables, denominados *Scalable Video Coding* (SVC) [Sch07], y perfiles para la codificación multivista denominados *Multiview Video Coding* (MVC) [Neḿ10].

Motivados por la fuerte demanda y consumo de contenidos en la red, así como por el incremento de resolución espacial y temporal de los nuevos formatos de vídeo superiores a los de HD, en 2010 el ITU-T e ISO/IEC volvieron a unir sus fuerzas en el *Joint Collaborative Team Video Coding* (JCT-VC) para el desarrollo de un nuevo estándar de vídeo que duplicara la eficiencia de compresión de H.264/AVC, y que facilitara la paralelización de ciertas herramientas que no permite H.264/AVC. Esto se convertía en un cuello de botella para la implementación en tiempo real de formatos de vídeo de muy alta resolución espacio-temporal. La petición de *Call for Proposals* lanzada desde e JCT-VC [VCEG-AM91] para el desarrollo del nuevo estándar, fue respondida por 27 propuestas que demostraron cómo la inclusión de ciertas herramientas podía mejorar significativamente la calidad de su predecesor, especialmente desde un punto de vista subjetivo. Tres años después, en 2013, ambos organismos aprobaron el estándar HEVC, que ha demostrado en numerosas evaluaciones científicas que puede alcanzar reducciones del ancho de banda por encima del objetivo fijado del 50% con respecto a H.264/AVC [Ohm12][Han12]. En 2014, un conjunto de perfiles profesionales denominados como *Range Extensions* (REext) [Fly15], fueron aprobados por el JCT-VC con el objetivo de cubrir las necesidades del mercado profesional, dando soporte a formatos con profundidades de píxel superiores a 10-bits y esquemas de color 4:2:0 y 4:4:4. Finalmente, en 2015, nuevos perfiles escalables han sido aprobados en HEVC denominados *Scalable HEVC Video Coding* (SHVC) [Boy15], facilitando las escalabilidad espacial, de calidad, de profundidad de píxel y de color.

2.2 ARQUITECTURA Y HERRAMIENTAS DE HEVC

En esta sección se aborda la arquitectura del último estándar de codificación de vídeo, HEVC, y se describen las nuevas herramientas que incorpora HEVC con respecto a estándares anteriores, las cuales son analizadas en detalle en los siguientes apartados. Una descripción detallada de la estructura y algoritmos que incorpora HEVC son presentados en [Sul12][Ohm13].

HEVC mantiene los principios basados en la arquitectura tradicional de codificación *híbrida predictiva-transformacional* descrita en al anterior apartado, en la cual se aplican técnicas predictivas *intra-frame* e *inter-frame*, para la obtención de un residuo de mínima energía, el cual es transformado al dominio frecuencial por medio de una transformación lineal basada en núcleos de transformación ortogonales. En el dominio frecuencial un proceso de cuantificación con pérdidas es aplicado a los coeficientes del residuo transformado, los cuales son posteriormente codificados entrópicamente sin pérdidas, conformando la información de salida del codificador denominado *bitstream*. Todo codificador *híbrido predictivo-transformacional* incluye los bloques funcionales necesarios para la reconstrucción/decodificación de la secuencia de vídeo codificada, con el objetivo de poder ser utilizada como información de referencia en las etapas predictivas. La Figura 2.1, muestra la arquitectura genérica de un codificador de vídeo HEVC, que sigue los fundamentos descritos de los codificadores *híbridos* con pérdidas.

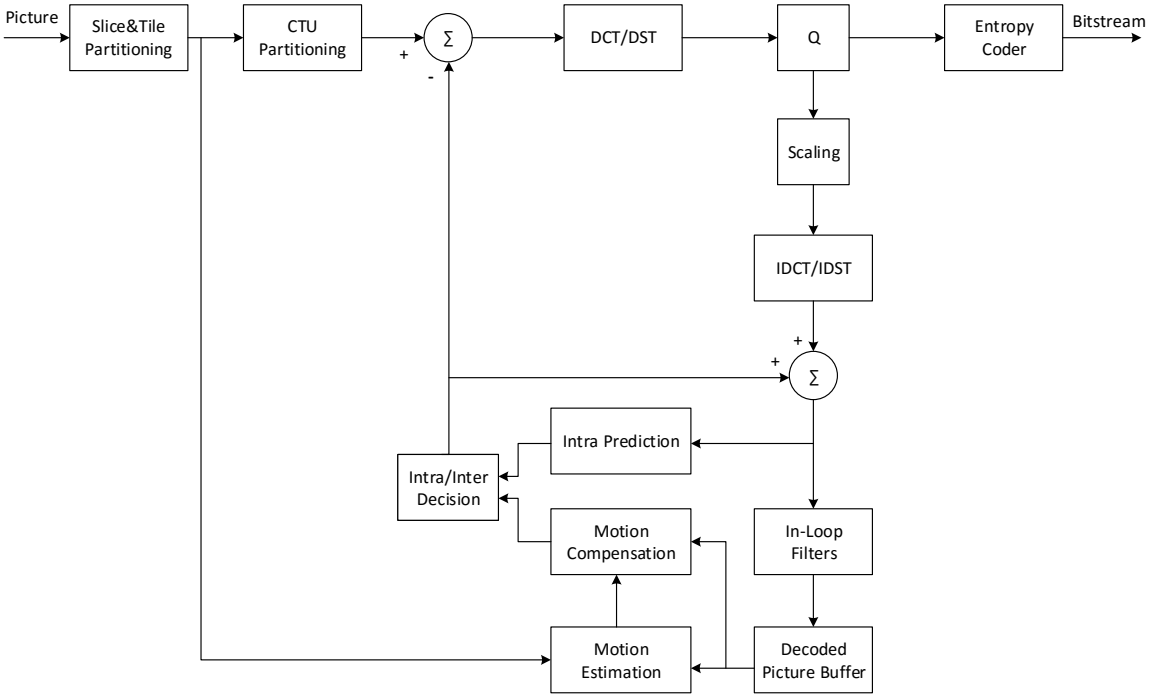


Figura 2.1. Arquitectura del codificador híbrido HEVC

Como se puede apreciar en la Figura 2.1, cada uno de los frames de la secuencia de vídeo puede ser inicialmente particionado en dos tipos de unidades denominadas *Slices*, similares a los ya definidos en H.264/AVC, y *Tiles* [Mis13], cuya finalidad es la de permitir un procesado paralelo y una menor latencia de codificación. Ambas estructuras pueden ser codificadas y decodificadas de modo independiente. Sin embargo, como contrapartida a su capacidad de paralelización éstas presentan una ligera pérdida de eficiencia de compresión debido a que las etapas predictivas están restringidas a su *Slice* o *Tile*. El particionado de la imagen en *Slices*, también habilita una segunda paralelización a nivel de filas, denominada *Wavefront Parallel Processing* (WPP) [Che14], que permite que las filas de las distintas unidades de codificación puedan ser codificadas tan pronto se disponga de la información necesaria obtenida desde la fila superior.

El segundo nivel de particionado es aplicable a los *Slices* o *Tiles* de una imagen, el cual constituye una de las principales novedades introducidas en HEVC con respecto a los estándares previos, (véase el apartado 2.2.1). Este particionado sustituye a los MB definidos en la totalidad de estándares previos a HEVC, por una nueva estructura jerárquica denominada *Coding Tree Unit* (CTU), que puede tener una resolución de hasta 64x64 píxeles, lo cual significa cubrir un área equivalente a 16 MB. Esta nueva unidad de codificación, CTU, puede ser particionada iterativamente en unidades de resolución mitad denominadas *Coding Unit* (CU), las cuales se estructuran como un árbol con distintos niveles de profundidad, conteniendo cada uno de ellos las CUs de su respectiva resolución. Este incremento en la granularidad de las unidades de codificación permite a HEVC ser más eficiente al adaptarse mejor a las características del contenido de la imagen, con bloques más pequeños para las zonas de elevado detalle y bloques más grandes para las zonas homogéneas.

Las etapas predictivas *inter-frame* e *intra-frame*, si bien mantienen los modelos básicos utilizados en H.264/AVC, ambas introducen mejoras significativas responsables en gran medida de la elevada eficiencia de HEVC. La predicción *intra-frame* se fundamenta en la utilización de los píxeles decodificados de las CUs vecinas, superior e izquierda, para la construcción de un predictor con distintas orientaciones, de modo que se pueda obtener un residuo de mínima energía como resultado de la decorrelación espacial de la CU. Las novedades introducidas con respecto H.264/AVC son el incremento del número de predictores direccionales, su aplicación a bloques de mayores dimensiones, 32x32 y 64x64, así como la mejora en los procesos de filtrado de los píxeles de referencia y de los predictores, que se traduce en una reducción de artefactos visuales que se producen en H.264/AVC cuando las imágenes presentan ciertos patrones. Una amplia descripción de las distintas etapas y procesos que comprende la codificación *intra-frame* es presentada en los apartados 2.2.2 y 2.3 de esta Tesis.

En lo que respecta a la predicción *inter-frame*, como se puede apreciar en la Figura 2.1, HEVC aplica el esquema clásico de estimación y compensación de movimiento, el cual utiliza como referencia para la estimación una imagen anterior o posterior a la actual que ha sido previamente codificada y decodificada, y convenientemente almacenada como referencia en el *Decoded Picture Buffer* (DPB). HEVC mantiene las precisiones fraccionarias de ½ de píxel y ¼ de píxel utilizadas en H.264/AVC, así como los modelos espacio-temporales predictivos avanzados para reducir la información de los MVs. No obstante, HEVC consigue una mejor eficiencia de codificación *inter-frame* al utilizar un amplio rango de tamaños de bloque, en especial de bloques rectangulares asimétricos no permitidos en H.264/AVC, que se restringe a bloques con un tamaño máximo de 16x16 píxeles y siempre con particionado simétrico. Otro aspecto relevante introducido en HEVC, es la mayor precisión de los filtros utilizados para la interpolación de píxeles en posiciones fraccionarias, los cuales están basados en el muestreo de las funciones del coseno, y a su vez permiten reducir la latencia de su cómputo al permitir el cálculo simultáneo de los píxeles en posiciones fraccionarias de ½ y ¼. Una descripción de los detalles relativos a la codificación *inter-frame* se presenta en al apartado 2.2.3 de esta Tesis.

El residuo generado como resultado de ambas etapas predictivas es transformado en el dominio frecuencial, utilizando unas versiones de aproximaciones enteras de la DCT, de modo similar a lo introducido en H.264/AVC. Tan sólo para los residuos provenientes de las predicciones *intra-frame* de tamaño 4x4, se aplica una transformada alternativa a la DCT, la *Discrete Sine Transform* (DST), la cual ha demostrado ser más eficiente en la compactación energética de dicho residuo. Los detalles de la implementación de ambas funciones son descritos en [Bri06]. Ambas transformadas bidimensionales utilizan núcleos de transformación ortogonales, permitiendo su implementación de modo separable como transformadas 1D, en sentido horizontal y vertical, y por lo tanto con menor coste computacional que el requerido por la transformación bidimensional de un solo paso.

HEVC define tamaños de transformación superiores al tamaño máximo de 8x8 permitido en H.264/AVC. En concreto, HEVC establece transformaciones de tamaños $2^N \; \forall \; N = 2,..,5$, por lo tanto mantiene tamaños pequeños de 4x4 y 8x8 para bloques de alta complejidad, e incrementa el rango de transformadas hasta 32x32 para mejorar en la compactación energética en amplias áreas de la imagen de muy baja complejidad con predominio de bajas frecuencias. Los coeficientes del residuo transformado son cuantificados utilizando el mismo esquema de cuantificación lineal utilizado previamente por H.264/AVC, compuesto por 52 niveles de cuantificación. Los detalles de los núcleos de transformación y la etapa de cuantificación definidos en el estándar, así como del proceso de escalado necesario para la reconstrucción de la imagen, son descritos en el apartado 2.2.4 de esta Tesis.

Como etapa final del codificador, tanto los coeficientes cuantificados como el resto de elementos sintácticos, imprescindibles para la correcta decodificación de la imagen, son codificados sin pérdidas por el codificador entrópico CABAC, que puede considerarse como una mejora de la codificación aritmética utilizada opcionalmente en H.264/AVC.

A diferencia de las etapas previamente descritas, cuyo objetivo principal es la mejora de la eficiencia con respecto a anteriores estándares, la codificación entrópica utilizada en HEVC mejora los dos elementos más críticos del algoritmo de CABAC utilizado en H.264/AVC, los cuales han impedido su utilización en tiempo real para formatos con elevadas resoluciones, como es el formato de UHD. Estos elementos se pueden resumir en la alta complejidad computacional debido al elevado número de elementos sintácticos definidos y a la elevada velocidad de procesado para altas tasas binarias, así como la falta de paralelización, al necesitar actualizar los modelos estadísticos de cada símbolo, como paso previo a la codificación de un nuevo elemento sintáctico. HEVC mantiene la eficiencia de compresión de su antecesor utilizando el mismo esquema de tres etapas, *Binarización*, *Modelado Contextual* y *Codificación Aritmética*, pero reduce su complejidad de modo notable. Esto reduce tanto el número de elementos sintácticos como muchas de las dependencias entre contextos, lo que le permite incrementar sustancialmente su velocidad de procesado para poder ser eficiente ante requerimientos de altas tasas binarias. En el apartado 2.2.5 se describen los detalles de la codificación entrópica utilizada por HEVC.

En lo que respecta a la arquitectura del decodificador que se incluye en la estructura del codificador, como se muestra en la Figura 2.1, éste mantiene su estructura tradicional ya definida en H.264/AVC, incluyendo un buffer para el almacenamiento de los frames decodificados, utilizados en la compensación de movimiento de la codificación *inter-frame*, y una etapa de post-filtrado, denominado *In-Loop Filter*, que permite una notable mejora de la calidad perceptual al reducir las típicas distorsiones entre bloques cuando se aplican fuertes niveles de cuantificación. H.264/AVC incorpora un único filtro de post-procesado denominado *Deblocking Filter*, con el objetivo de reducir las distorsiones entre bloques de codificación en zonas de la imagen relativamente homogéneas, donde el ojo percibe fácilmente si existen diferencias de nivel entre ambos bloques. HEVC mejora dicho *Deblocking Filter* [Nor12] e introduce un segundo post-filtrado, en este caso, para la reducción de las distorsiones internas entre los bloques debidas en gran medida a los efectos de la cuantificación cuando se utilizan transformadas de gran tamaño, mayores de 8x8. Si bien estas transformadas son muy eficientes desde la perspectiva de la compactación energética, pueden provocar efectos visuales de *ringing*, cuando se aplican elevados factores de compresión como se describe en [Han10].

Este filtro se denomina *Sample Adaptive Offset* (SAO) y permite una implementación independiente por CTU, lo que facilita su paralelización a nivel de CTU cuando se utilizan

las opciones de particionado tipo *Slice* o *Tiles*. La arquitectura definida en el SAO surge como una simplificación de la complejidad de 3 técnicas de post-filtrado propuestas en el desarrollo inicial del estándar, *Picture-based Band Offset* (PBO), *Picture-based Edge Offset* (PEO) y *Picture-based Adaptive Clipping* (PAC) [JCTVC- D122]. Así mismo se ha reducido a tan sólo 4 el número de patrones y sub-bandas a los que se aplica el filtro, y el tipo de offset puede ser deducido de modo intrínseco en el decodificador sin necesidad de señalización adicional. Una descripción detallada de la implementación y prestaciones del SAO es descrita en [Chi12].

Como resumen de este apartado, la Tabla 2.1 recoge una comparativa de las herramientas de codificación que caracterizan la arquitectura de HEVC, con tres de los estándares de compresión de vídeo más representativos de la evolución de estas tecnologías, H.261, MPEG-2 y H.264/AVC.

Tabla 2.1. Comparación de las herramientas de codificación incluidas en los estándares H.261, MPEG-2, H.264/AVC y HEVC

	H.261	**MPEG-2**	**H.264/AVC**	**HEVC**
Resoluciones	QCIF (172x144), CIG (352x288)	QCIF (172x144) hasta HD (1920x1080)	QCIF (172x144) hasta (4096x2304)	QCIF (172x144) hasta UHD 8K (8192x4320)
Unidades de Codificación	Macro Bloque (16x16)	Macro Bloque (16x16)	Macro Bloque (16x16)	Coding Tree Unit (8x8, 16x16, 32x32, 64x64)
Tamaños de predicción *intra-frame*	16x16	16x16	16x16, 8x8, 4x4	64x64, 32x32, 16x16, 8x8, 4x4
Predicción *intra-frame*	Modo DC	Modoe DC	16x16 y 4x4: 4 modos 8x8: 9 modos	35 modos para todos los tamaños
Predicción *inter-frame*	Solo posiciones enteras	Precisión ½ pixel	Precisión ½ pixel Precisión ¼ pixel	Precisión ½ pixel Precisión ¼ pixel
Vectores de Movimiento	1 MV	1MV	3 MV Modos Direct/Skip	AMVP Modos Merge/Skip
Filtros interpoladores *inter-frame*	-	Bilineal	Lum. ½: 6-taps Lum. ¼ : bilineal Crominan.: bilineal	Lum. ½: 8-taps Lum. ¼ : 7-taps Crominan.: 4-tap
Transformación	DCT (8x8)	DCT (8x8)	DCT entera (4x4, 8x8)	DCT entera (4x4, 8x8, 16x16, 32x32) DST entera (4x4)
Codificación entrópica	VLC	VLC	CAVLC, CABAC	CABAC
In Loop Filter	Filtro suavizado MB	-	Deblocking Filter	Deblocking Filter Sample Adaptive Filter

2.2.1 Particionado de las Unidades de Codificación, Predicción y Transformación

Como ya se ha mencionado anteriormente, una de las herramientas más novedosas que introduce el estándar HEVC es la nueva unidad de codificación CTU, la cual sustituye al tradicional MB definido en todos los estándares previos, que utiliza un bloque de 16x16 píxeles. La CTU es una estructura más flexible, que permite una adaptación más precisa a los detalles y textura de la imagen, con el objetivo de mejorar la eficiencia de compresión. Una descripción detallada del modelo de particionado en HEVC es presentado en [Mar10][Han10].

Una CTU está compuesta por tres bloques denominados *Coding Tree Blocks* (CTB), uno para la componente de la luminancia, y otros dos bloques para las dos componentes de color, también denominadas crominancia o color. La gran ventaja que presenta la CTU es que esta puede ser iterativamente particionada en cuatro sub-bloques cuadrados de resolución mitad denominados CUs, e igualmente, cada CU está a su vez compuesta por tres bloques denominados CBs, para las correspondientes componentes de luminancia y color. La estructura de particionado jerárquico de la CTU puede considerarse como un árbol donde cada rama finaliza en un nodo que posee una CU cuyo tamaño depende del nivel de profundidad en el que se encuentra su respectiva rama. Esta estructura es denominada en la literatura como *quadtree*.

El máximo tamaño permitido para la CTU en HEVC es de 64x64 píxeles, y puede ser particionado en 3 niveles de profundidad, denominados como d, y por consiguiente el tamaño mínimo que puede alcanzar una CU es de 8x8 ($64/2^3$). En cada nivel de profundidad del árbol de particionado de una CTU, puede haber 4^d CUs, denominándose a cada CU como $CU_{d,k}$ ($d = 0,..,3, \mid k = 0,..,4^d - 1$), siendo d el nivel de profundidad y K el índice de la CU en dicho nivel. Por consiguiente, las cuatro sub-CUs que dependen de una $CU_{d,k}$ son denominadas como $CU_{d+1,4k+i}$ ($i = 0,..,3$).

Para incrementar la flexibilidad de particionado de las CTUs y mejorar así la eficiencia de codificación, cada CU se define como la raíz de dos nuevos árboles que contienen dos nuevos tipos de unidades denominadas *Prediction Units* (PU) y *Transform Units* (TU), las cuales constituyen las unidades básicas de predicción, tanto para la codificación *intra-frame* como *inter-frame*, y de transformación del residuo obtenido como resultado de la predicción. De igual modo, cada PU y TU está compuesta por tres bloques, uno para la componente de luminancia y dos para las componentes de crominancia, denominados *Prediction Blocks* (PB) y *Transform Blocks* (TB), respectivamente.

En HEVC se establece que las PUs pertenecientes a una predicción *intra-frame* deben tener el mismo tamaño que sus respectivas CUs de las que dependen, y pueden ser nuevamente particionadas en 4 sub-PUs de resolución mitad, tan sólo si éstas pertenecen a una CU de tamaño 8x8. Por consiguiente, dichas Pus pueden tener un tamaño que cubre el rango de 64x64 píxeles a 4x4 píxeles. Las PUs pertenecientes a una predicción *inter-frame* pueden ser particionadas en sub-PUs rectangulares simétricas o asimétricas, como se describe más adelante en este mismo apartado.

En lo que respecta a las TUs, éstas pueden ser particionadas en una estructura de árbol denominado *Residual Quad Tree* (RQT), el cual puede disponer como máximo de 3 niveles de profundidad. Debido a que el estándar solo permite matrices de transformación con tamaños máximos de 32x32 y mínimos de 4x4, por motivos de eficiencia y de coste computacional, los tamaños de las TUs están restringidos a estas resoluciones. Por consiguiente una PU perteneciente a una CU de tamaño 8x8, sólo podrá particionarse en dos niveles de profundidad, esto es, con TUs mínimas de tamaño 4x4, y una PU de 64x64, siempre es codificada con un tamaño máximo de 4 TUs de 32x32.

La Figura 2.2 muestra un ejemplo del particionado de una CTU de 64x64 en modo *intra-frame*, en sus tres arboles jerárquicos de CUs, PUs y TUs, indicados en color rojo, verde y azul respectivamente. Se puede comprobar cómo los arboles de PUs y TUs tienen como raíz los nodos de una CU.

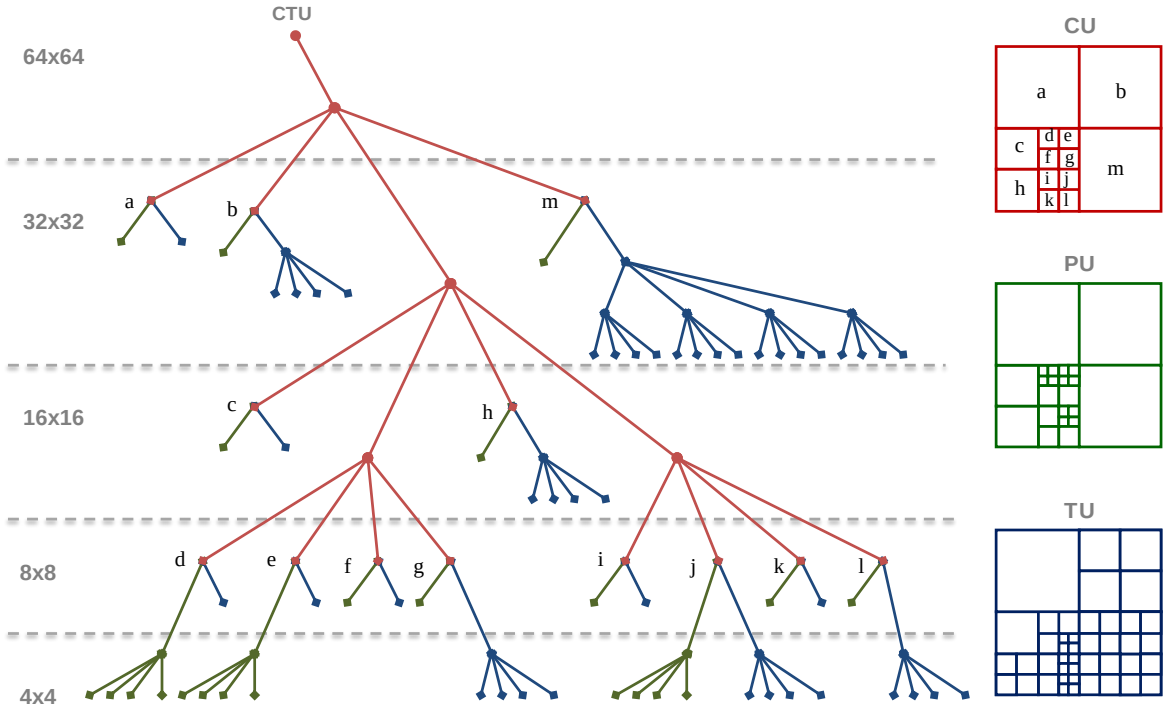


Figura 2.2. Ejemplo de particionado de una CTU en sus respectivos árboles de CUs, PUs y TUs

Con el objetivo de facilitar la compresión del esquema de particionado se han etiquetado las correspondientes CUs por orden de escaneo de izquierda a derecha y arriba abajo, alfabéticamente desde la "*a*" a la "*m*". Se puede comprobar cómo en el segundo nivel de profundidad del árbol ($d = 1$) existen 3 CUs con tamaño 32x32, que se corresponde con la CU "*a*" cuya TU no es particionada, la CU "*b*" cuya TU ha sido dividida en 4 TUs de 16x16, y la CU "*m*" cuyas TUs han sido particionadas con dos niveles de profundidad en 16 TUs de 8x8. En los tres casos, al no pertenecer sus respectivas PUs a una CU de tamaño 8x8 no pueden ser particionadas. El mismo razonamiento puede ser aplicado al resto de CUs en los distintos niveles, destacando cómo las PUs pertenecientes al último nivel de profundidad, dependientes de las CUs "*d*", "*e*" y "*j*", son particionadas en 4 PUs alcanzando el tamaño mínimo permitido de 4x4. De igual modo, tan sólo las TUs pertenecientes a las CUs "*g*", "*j*" y "*l*" son particionadas con un único nivel de profundidad en 4 TUs de tamaño 4x4.

La Figura 2.3 muestra un ejemplo real de particionado de CTUs en HEVC, donde se puede apreciar cómo los tamaños de las CUs se ajustan al contenido de la imagen, utilizando CUs de tamaño 64x64 y 32x32 para zonas texturadas pero con bajo detalle, como son las zonas del césped de la parte superior izquierda, mientras que las CU más pequeñas de 16x16 y 8x8 se seleccionan en los bordes que delimitan los objetos de la escena, permitiendo una codificación más precisa y eficiente.

Figura 2.3. Ejemplo de particionado de CTUs en HEVC

Como ya se ha mencionado, las PUs pertenecientes a la predicción *inter-frame* pueden ser particionadas en regiones no cuadradas, los cuales no están permitidas al resto de CUs, PUs *intra-frame* y TUs. En concreto, HEVC define 4 tipos de particionado simétrico de resoluciones $2Nx2N$, NxN, $2NxN$ y $Nx2N$, y 4 tipos de particionados asimétricos con resoluciones de $2NxnU$, $2NxnD$, $nLx2N$, y $nRx2$N, $\forall N = 4, 8, 16, 32 \mid n = N/2$. La Figura 2.4 muestra los 8 tipos distintos de particionado que puede adoptar una PU *inter-frame*, donde U, D, L y R denotan donde se sitúa la porción asimétrica más pequeña superior, derecha, izquierda y derecha respectivamente.

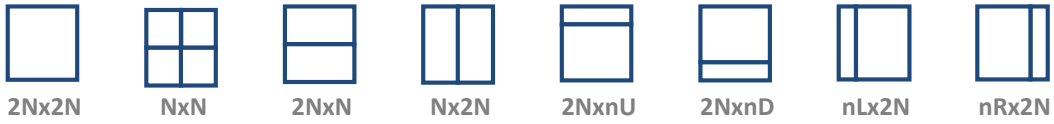


Figura 2.4. Particionados simétricos y asimétricos de PUs en predicción *inter-frame*

Las particiones asimétricas mejoran significativamente la eficiencia de la predicción *inter-frame* de HEVC con respecto a H.264/AVC, al adaptarse a los contornos de las imágenes de modo más preciso. En la Figura 2.5 se muestra un ejemplo de particionado asimétrico, donde se puede apreciar como su correcta selección permite generar residuos más compactos, que son codificados de modo más eficiente.

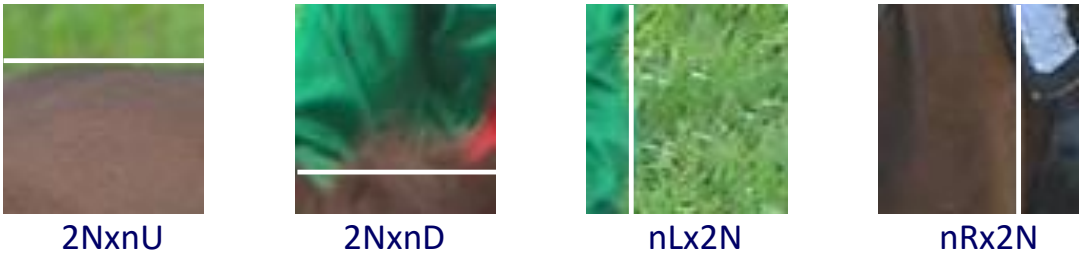


Figura 2.5. Ejemplo de particionado asimétrico de una PU en predicción *inter-frame*

El estándar define una serie de restricciones al particionado de PUs en modo *inter-frame*, como la utilización del particionado NxN tan solo en PUs pertenecientes a CUs de tamaño 8x8, o la prohibición de utilizar particionados asimétricos en PUs pertenecientes a CUs de tamaño 8x8. La Tabla 2.2, muestra el conjunto de particionados permitidos para las PUs pertenecientes a la predicción *inter-frame*, en función del tamaño de la CU de la que depende.

Tabla 2.2. Particionados de PUs en predicción inter-frame

	PU inter-frame							
CU	**2NX2N**	**2nXn**	**Nx2N**	**NxN**	**2NxnU**	**2NxnD**	**nLx2N**	**nRx2N**
64	64x64	64x32	32x64	-	64x16 64x48	64x48 64x16	16x64 48x64	48x64 16x64
32	32x32	32x16	16x32	-	32x8 32x24	32x24 32x8	8x32 24x32	24x32 8x32
16	16x16	16x8	8x16	-	16x4 16x12	16x12 16x4	4x16 12x16	12x16 4x16
8	8x8	8x4	4x8	4x4	-	-	-	-

2.2.2 Predicción Intra-frame

En este apartado se introducen los fundamentos básicos de la codificación *intra-frame* en HEVC, mientras que en el apartado 2.3 se describirá en detalle cada una de las etapas que componen su arquitectura, que incluye la selección de muestras de referencia, la construcción de los predictores y un análisis de su complejidad computacional, al ser ésta una de las principales motivaciones de la presente Tesis.

El esquema de predicción *intra-frame* de HEVC se basa en el modelo utilizado previamente por H.264/AVC, pero introduce mejoras sustanciales en la mayoría de las etapas que lo componen. La codificación *intra-frame* explota la elevada correlación espacial entre un PBs y los píxeles frontera de sus PBs vecinos, los cuales son utilizados para la construcción de los predictores. La predicción *intra-frame* adoptada finalmente en HEVC deriva de la unificación de un conjunto de propuestas [Min10][Pia10][Ugu10][Kan11] que presentaban elementos muy similares en su arquitectura, y que finalmente convergieron en un modelo avanzado de predicción *intra-frame*.

Al igual que H.264/AVC, HEVC establece dos tipos de predictores, los no-angulares, compuestos por los modos *Planar* y *DC*, y los angulares, compuestos por 33 modos direccionales, divididos en dos grupos en función de su orientación, 16 modos horizontales definidos de H_2 a H_{17}, y 17 modos verticales definidos de V_{18} a V_{34}. La Figura 2.6 muestra la orientación de los 33 modos angulares, donde se puede apreciar como la distribución de los modos no es homogénea, sino que se concentra en las orientaciones horizontal y vertical, modos H_{10} y V_{26} respectivamente.

Los modos no-angulares, *Planar* y *DC*, están diseñados para obtener una elevada eficiencia de codificación con bloques que presenten una alta homogeneidad entre sus píxeles, o bloques con gradientes muy suavizados. Por el contrario, los modos angulares son muy eficientes con bloques cuyo contenido presenta bordes o patrones direccionales muy fuertes, es decir bloques que contengan una elevada energía en altas frecuencias con una misma orientación espacial. Las mejoras más destacables que presenta el nuevo modelo de predicción *intra-frame*, es el incremento del número de predictores angulares, que se eleva desde los 8 modos definidos en H.264/AVC, hasta los 33 modos. Esta alta densidad de predictores direccionales, permite encontrar un predictor que describa con elevada precisión la orientación de los posibles patrones existentes en las imágenes, lo cual mejora de modo notable la eficiencia de compresión.

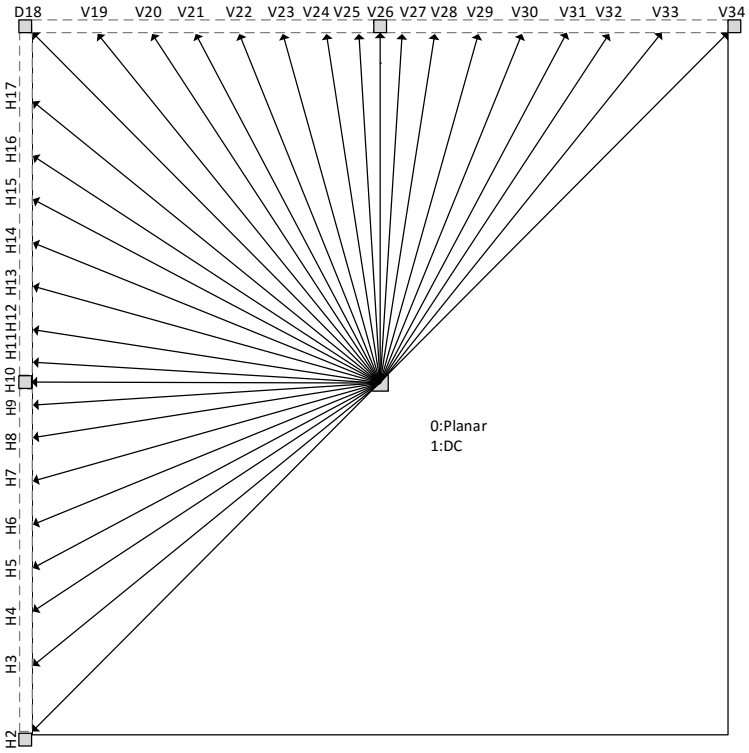


Figura 2.6. Modos de predicción intra-frame en HEVC

HEVC también incrementa el tamaño máximo de los bloques de predicción por encima de los 16x16 píxeles soportado H.264/AVC, con PBs de tamaños de 32x32 y 64x64 píxeles, lo que le permite mejorar la eficiencia de predicción en formatos de elevada resolución, como son los formatos de HD y el emergente formato de UHD. En este tipo de imágenes, la probabilidad de encontrar áreas de la imagen con un mismo patrón homogéneo es mucho más elevada que en formatos de menor resolución. Debido a que el tamaño máximo permitido de transformada en HEVC se limita a 32x32, la predicción en PBs de 64x64 se implementa como 4 sub-PBs de 32x32, con el mismo modo de predicción.

La utilización de PBs de gran tamaño mejora la eficiencia de compresión, como ya se ha indicado, pero también puede introducir distorsiones visuales en los bordes de las PBs, en función del contenido de la imagen. Por este motivo, HEVC ha introducido un nuevo filtro de suavizado de mayor energía para los PBs de 32x32, dentro de la propia predicción *intra-frame*. En el apartado 2.3 se describen otras mejoras y novedades introducidas en la predicción *intra-frame* en HEVC, como la aplicación selectiva del pre-filtrado de los píxeles de referencia, un nuevo post-filtrado de ciertos predictores, un nuevo algoritmo para el cómputo del predictor *Planar* alternativo a la predicción bilineal utilizado en H.264/AVC, un nuevo algoritmo para el cálculo de los predictores angulares, o la utilización de una codificación transformacional basada en la DST alternativa a la DCT, para codificar el residuo de los PB de tamaño 4x4.

HEVC también mejora la codificación contextual tanto de los modos de predicción, utilizando una mejora del concepto de *Most Probable Mode* (MPM) sobre el definido en H.264/AVC, como en la codificación del residuo transformado, donde se amplían los modos de escaneo de los coeficientes en virtud del modo de predicción y del tamaño del TB. En el apartado 2.3.8 se presenta una estimación de la complejidad computacional requerida por las distintas etapas que comprende la codificación *intra-frame* en HEVC, donde se muestra cómo su coste computacional es mucho más elevado que en el esquema de codificación *intra-frame* de H.264/AVC.

2.2.3 Predicción Inter-frame

Como se describió en el apartado 2.2, la codificación i*nter-frame* definida en HEVC se basa en los principios tradicionales de estimación y compensación de movimiento utilizados de modo amplio en los anteriores estándares de codificación de vídeo.

La predicción *inter-frame* se lleva a cabo dentro de su nueva estructura jerárquica de particionado del CTB, y su tamaño inicial viene definido por el tamaño del CB al que pertenece, pudiendo ser particionada de modo simétrico o asimétrico. A diferencia de H.264/AVC, la predicción i*nter-frame* calcula la ME y MC en un rango de tamaños de bloque más amplios, que van desde 64x64 píxeles a 4x4 píxeles, lo cual le permite mejorar su eficiencia de predicción para formatos de alta resolución.

La predicción *inter-frame* se aplica igualmente a posiciones enteras, denominadas IME (*Integer Motion Estimation*) y posiciones fraccionarias, denominadas FME (*Fractional Motion Estimation*), manteniendo la precisión fraccionaria de ½ y ¼ de píxel establecida en H.264/AVC. Una de las principales diferencias que presenta la predicción *inter-frame* con respecto a H.264/AVC son los filtros utilizados para la FME. Las componentes de luminancia utilizan filtros separables de longitudes de 8-tap o 7-taps, que sustituyen al filtro de dos etapas definidas en H.264/AVC compuesto por un filtrado de 6-tap para la precisión de ½ de píxel, seguido de una interpolación bilineal para el cálculo de los píxeles con precisión de ¼ de píxel. Con respecto a la predicción de las componentes de crominancia, éstas siguen los particionados del PB de luminancia, compartiendo los correspondientes vectores de movimiento, pero con el correspondiente factor de escala en función del formato de submuestreo de color aplicado. El filtro utilizado en este caso es de 4-taps.

Otra diferencia destacable que presenta la predicción *inter-frame* en HEVC con respecto a su predecesor, es el modo en que la información adicional al residuo de predicción es inferida, incluyendo la información de sus MVs, los índices de los frames de referencia de las respectivas listas, y los flag de las listas de frames de referencias utilizadas, entre otras.

En concreto, HEVC ha introducido dos modelos de codificación de la información de estimación de movimiento adicional al residuo:

- *Merge:* es un modo implícito que sustituye a los modos *Skip/Direct* definidos en H.264/AVC. Este modo evita la codificación de los parámetros de la ME, transmitiendo exclusivamente un índice que identifica dicha información en una lista de parámetros candidatos que ha sido derivada de los bloques vecinos, tanto espaciales como temporales, reduciendo significativamente la tasa de bits utilizada por bloque.
- *Inter Mode:* es un modo de inferencia explícito, y que incluye un nuevo modo de inferencia de los MVs denominado AMVP (Advanced MV Prediction), similar al modo *Merge*, pero restringiendo el número de MVs candidatos a dos espaciales y uno temporal. En ambos casos se utiliza como candidatos los MVs del mismo frame o los de frames previos, y la información de las componentes de crominancia se derivan de la componente de luminancia.

En lo que respecta a la estructura de la codificación *inter-frame*, HEVC define igualmente *Slices* tipo *I, P* y *B*, pero introduce el concepto de *Slices* GPB (*Generalized P* and *B*), desacoplando el tipo de frame y el orden de codificación, y permitiendo que un frame contenga *Slices* de diferente tipo (*I, P, B*). HEVC también mantiene la estructura de *Slices B* jerárquicos, donde un *Slice B* puede ser utilizado como referencia de otros *B*s.

Otros aspectos que HEVC hereda de su predecesor son el concepto de múltiples frames de referencia, denominados IDRs (*Instaneous Decoding Refresh*), y la predicción ponderada o WP (*Weighted Prediction*). HEVC mantiene las dos listas de frames de referencia (*list0* y *list1*) permitiendo la utilización de cualquiera de ellas para los *Slices* no-bidireccionales, *Slices P*, y utiliza ambas listas para los *Slices* con predicción bidireccional, *Slices B.*

La gestión de los frames decodificados y almacenados en el DPB para su utilización como frames de referencia de futuros *Slices Ps* o *Bs*, se realiza señalizando una lista denominada RPS (*Reference Picture Set*) en la cabecera de su correspondiente *Slice*, con los índices o POCs (*Picture Order Count*) de los frames utilizados como referencia. La sintaxis utilizada para la identificación de las RPS se simplifica enormemente con respecto a H.264/AVC, sin necesidad de inferencias al estado interno de los frames almacenados en el DPB, lo cual facilita la implementación de los modos de acceso y control al stream para funciones de búsqueda rápida (*rewind, seeking, o fast forward*).

2.2.3.1 Filtros interpoladores para FME

La interpolación de píxeles para posiciones fraccionarias en HEVC se puede considerar como una versión simplificada, pero más eficiente, de la utilizada en H.264/AVC, la cual define tres filtros distintos de interpolación: dos para la componente de luminancia de 8-taps para las posiciones ½ de píxel y de 7-taps para las posiciones de ¼ de píxel, y un conjunto de filtros de 4-taps para las componentes de crominancia con 1/8 de precisión.

En HEVC se reducen los tiempos de latencia de la ME al poderse computar la interpolación en una sola etapa para las posiciones de ½ y ¼ de píxel, a diferencia de H.264/AVC donde primero se computan las posiciones de ½ de píxel (filtro de 6-taps en H.264/AVC) y después las de ¼ de píxel (filtro bilineal en H.264/AVC). El incremento en el número de taps de los filtros tanto de luminancia como de crominancia (2-taps para la crominancia en H.264), permite obtener un interpolador de mayor precisión, no requiriendo de procesos de redondeo intermedios. Esto aumenta la eficiencia de compresión de la predicción *inter-frame*, consiguiendo un residuo de menor energía, el cual es codificado de modo más eficiente por la etapa transformacional.

Los filtros interpoladores son derivados de las funciones base de la DCT. Sus valores son mostrados en la Tabla 2.3, donde $L_{i/j}$ y $C_{i/j}$ denotan los filtros de luminancia y crominancia respectivamente, i es la posición del píxel interpolado y j la precisión de la interpolación.

Tabla 2.3. Filtros de interpolación de luminancia y crominancia para posiciones fraccionarias de la ME

	Localización píxeles							
Filtro	**-3**	**-2**	**-1**	**0**	**+1**	**+2**	**+3**	**+4**
$L_{1/4}$	-1	4	-10	58	17	-5	1	
$L_{1/2}$	-1	4	-11	40	40	-11	4	1
$L_{3/4}$		1	-5	17	58	-10	4	-1
$C_{1/8}$			-2	58	10	-2		
$C_{2/8}$			-4	54	16	-2		
$C_{3/8}$			-6	46	28	-4		
$C_{4/8}$			-4	36	36	-4		
$C_{5/8}$			-4	28	46	-6		
$C_{6/8}$			-2	16	54	-4		
$C_{7/8}$			-2	10	58	-2		

2.2.4 Transformación y Cuantificación

Dentro del esquema de codificación híbrido definido en HEVC, la codificación transformacional mantiene los mismos principios aplicados por estándares previos del ITU-T y MPEG. Dicha transformación utiliza una transformación bidimensional separable en sentido horizontal y vertical, pero en este caso derivados de una aproximación entera de los coeficientes de la DCT, con la excepción de los TBs de 4x4 procedentes de la predicción *intra-frame*, ya que éstos utilizan una aproximación de las funciones base de la DST, tal y como se explica en el apartado 2.2.3.

Como se describió en el apartado 2.2.2, los bloques de transformación son denominadas TUs, y se aplican a tamaños de bloques, siempre cuadrados, que cubren el rango de 4x4 hasta 32x32. HEVC es el primer estándar que utiliza tamaños de transformadas superiores a 8x8, con tamaños de 16x16 y 32x32 que son muy eficientes compactando la energía en el dominio transformado en áreas de la imagen que presentan patrones con una cierta homogeneidad frecuencial, como ocurre con los formatos de elevada resolución de HD y UHD.

Una mejora introducida en el núcleo de la transformación en HEVC consiste en que sus vectores base se ajustan mejor a las funciones teóricas definidas en la DCT, por lo que no es preciso compensarlos para cada coeficiente en una etapa posterior de escalado, con el objetivo de cumplir el requisito de ortonormalidad como ocurre en H.264/AVC. Por simplicidad, el estándar sólo recoge la matriz de transformación de 32x32, y las distintas matrices de menor tamaño son calculadas por submuestreo de las filas y columnas de dicha matriz. El detalle de las matrices de transformación para el conjunto de los tamaños de TBs disponibles se describe en [JCTVC-K1002]. A modo de ejemplo, la Figura 2.7 muestra las matrices de transformación para la DCT de tamaño 8x8 y 4x4, así como la DST de 4x4.

DCT 8x8

64	64	64	64	64	64	64	64
89	75	50	18	-18	-50	-75	-89
83	36	-36	-83	-83	-36	36	83
75	-18	-89	-50	50	89	18	-75
64	-64	-64	64	64	-64	-64	64
50	-89	18	75	-75	-18	89	-50
36	-83	83	-36	-36	83	-83	36
18	-50	75	-89	89	-75	50	-18

DCT 4x4

64	64	64	64
83	36	-36	-83
64	-64	-64	64
36	-83	83	-36

DST 4x4

29	55	74	84
74	74	0	-74
84	-29	-74	55
55	85	74	-29

Figura 2.7. Matrices de transformación para DCT 8x8, DCT 4x4 y DST 4x4, en HEVC

HEVC utiliza el mismo esquema de cuantificación lineal, y de reconstrucción por escalado uniforme utilizado por H.264/AVC, con 52 niveles determinados por el parámetro QP, $QP = 0,..,51$, el cual duplica su tamaño de cuantificación cada 6 niveles, permitiendo un control logarítmico de la cuantificación. De modo genérico, el proceso de cuantificación se obtiene aplicando la expresión matemática descrita en la Ecuación (2.1), donde $C(x,y)$ es la matriz de coeficientes transformados, $F(qp)$ es el vector con los factores de cuantificación definidos en la Ecuación (2.2), N es el tamaño del TB, y $C_{qp}(x,y)$ es la matriz de coeficientes transformados:

$$C_{qp}(x,y) = round\left(\frac{C(x,y)*F[QP\%6]}{2^{\left(29+\frac{QP}{6}-N-BitDepth\right)}}\right) \quad (2.1)$$

$$F(qp) = [26214,\ 23302,\ 20560,\ 18396,\ 16384,\ 14564\ ] \quad (2.2)$$

De modo adicional a los dos modos de predicción descritos, *intra-frame* e *inter-frame*, HEVC soporta dos modos de codificación para escenarios particulares, como son el modo *Transform_Skip*, que permite cuantificar directamente la señal del residuo sin haber sido transformada, y el modo *Lossless* que le aplica la codificación entrópica al residuo, sin haber sido transformado ni cuantificado.

Debido al elevado tamaño de los núcleos de transformación, HEVC especifica que los resultados intermedios llevados a cabo en el proceso de transformación se puedan llevar a cabo con aritmética entera de 16-bits para el perfil *Main*, y 32-bits para el perfil *Main1*0, por lo que entre cada uno de los procesos de transformación unidimensional, es preciso aplicar el correspondiente recorte a 16-bits y 32-bits. De modo similar, en el proceso de decodificación posterior a la transformación y cuantificación inversa, es preciso aplicar una función de redondeo, denominado *Shift*, que permite devolver las muestras decodificadas al margen dinámico original de la secuencia, como se define en la ecuación (2.3), donde r(x,y) son los coeficientes del residuo resultado de la transformación y cuantificación inversa, y $BitDepth$ la profundidad de píxel de la secuencia de vídeo:

$$R(x,y) = round\left(\frac{r(x,y)}{2^{(20-BitDepth)}}\right) \quad (2.3)$$

Por último, un aspecto relevante en la etapa de transformación es el reordenamiento de los coeficientes transformados, similar a la estrategia de *zig-zag* ya utilizada en estándares previos. HEVC introduce un complejo método de scaneo de patrones en función del tamaño de los TBs, el modo de codificación *inter-frame* o *intra-frame*, incluso del modo de predicción utilizado, que permite mejorar la eficiencia de la codificación estadística. Un descripción de tallada de este proceso es presentado en [Sol12].

2.2.5 Codificación Entrópica

A diferencia de H.264/AVC, HEVC sólo implementa un algoritmo de codificación estadística basado en la codificación aritmética adaptativa, denominado CABAC, motivado por su demostrada eficiencia de compresión superior a la codificación entrópica basada en códigos de longitud variable o VLC, y que mejora igualmente la versión adaptativa utilizada en H.264/AVC denominada CAVLC (*Context-Adaptive Variable Length Coding*).

Para corregir los problemas de baja velocidad de procesado, o *throughput*, que presenta el algoritmo de CABAC utilizado por H.264/AVC, HEVC introduce numerosas mejoras como la reducción del número de contextos de los elementos sintácticos a codificar o *bin*s, y la reducción de sus dependencias, permitiendo incrementar significativamente su *throughput*, manteniendo la eficiencia de codificación sin pérdidas y reduciendo los requerimientos de memoria, entre otros beneficios. Una descripción detallada del algoritmo de CABAC definido en HEVC es descrita en [Sze12].

En general, HEVC reduce el número de contexto de los *bins* en una proporción de 8 con respecto a H.264/AVC. La memoria requerida para el almacenamiento de los contextos se reduce igualmente en un factor de 3, eliminando el cuello de botella que suponía el modelo de CABAC implementado en H.264/AVC, el cual limitaba la codificación en tiempo real para formatos de elevada resolución como el UHD, muy especialmente cuando se requerían codificaciones con altas tasas binarias que demandan procesar un elevado número de elementos sintácticos.

En HEVC, el CABAC se aplica a todos los elementos sintácticos o símbolos para ser codificados de modo muy eficiente y próximo a su entropía por medio de un número no-entero de bits. Estos elementos sintácticos incluyen la información de los elementos generados para cada una de las etapas del codificador definidas en el estándar, entre otros:

- *CU:* la sintaxis que escribe su estructura de particionado, y tipo de predicción utilizada, *inter-frame* o *intra-frame.*
- *PU*: la información referente al modo de predicción *intra-frame* si este modo es utilizado, o por el contrario la información del particionado de la PU así como la información de los MVs pertenecientes a la etapa de ME de la predicción *inter-frame.*
- *TU*: la información referente al tamaño y tipo de transformación aplicado a cada PU, describiendo para cada uno de ellos la posición del último coeficiente distinto de cero, su mapa de significatividad, así como su magnitud y signo.

El modelo de CABAC se estructura en las siguientes tres etapas, que requieren de su procesado secuencial:

- *Binarización*: proceso que mapea los elementos sintácticos a *bins*.
- *Modelado Contextual*: proceso en el que se estiman de modo adaptativo las probabilidades de cada *bin*.
- *Codificación aritmética*: en este proceso se codifican sin pérdidas los *bins* en función de sus probabilidades estimadas.

En lo que respecta al proceso de binarización, éste mantiene los cuatro tipos de binarización utilizados en H.264/AVC, *Fixed Length, Exp-Golomb, Unary y Truncated Unary*, los cuales son aplicados de modo individual o combinando varios de ellos, con dependencias en función de valores previamente codificados. El *Modelado Contextual*, estima la probabilidad de los símbolos a codificar basados en los estadísticos de los símbolos vecinos previamente codificados, siendo éste el factor clave para alcanzar una elevada ganancia de compresión. HEVC aplica la misma metodología de modelado utilizada en H.264/AVC, pero reduce sustancialmente el número de contextos, pero mejorando la eficiencia obtenida en H.264/AVC.

El cuello de botella de la codificación entrópica viene determinado por el número de símbolos por segundo que se requiere procesar por segundo, lo cual determina el *throughput* del codificador. En H.264/AVC, las fuertes dependencias en el modelo de contexto, evita que se puedan procesar multiplex bins en paralelo. HEVC incrementa el número de bins codificados en modo *bypass*, donde no existen dependencias con respecto a la selección del contexto, permitiendo su codificación en paralelo.

A modo de ejemplo, en la Tabla 2.4 se muestra una comparativa entre H.264/AVC y HEVC llevada a cabo en [Sze13], en la que se recoge el número de contextos y el máximo número de *bins* codificados en modo contextual por PU, en cada estándar. Como se puede apreciar, el número de contextos totales se reduce casi a la mitad en HEVC, con 14 contextos frente a 26 en H.264/AVC, mientras que el número máximo de *bins* por PU es 8 veces inferior en HEVC con respecto a H.264/AVC.

Tabla 2.4. Comparativa de los modelos contextuales de CABAC en H.264/AVC y HEVC

	H.264/AVC		**HEVC**		
	Intra	Inter	Intra	AMVP	Merge
Número de candidatos	1	1	3	2	5
Número de contexto	6	20	2	10	2
Número máximo de contextos por PU	7	98	5	12	2

2.3 PREDICCIÓN INTRA-FRAME

Como se ha descrito en el apartado 2.2.2, con el objetivo de explotar la fuerte correlación espacial que presentan de modo habitual las imágenes naturales, HEVC incorpora una etapa avanzada de predicción *intra-frame* que mejora significativamente su eficiencia de codificación. Su arquitectura es muy similar a la de su predecesor, el estándar H.264/AVC, que se fundamenta en la utilización de los píxeles previamente decodificados de los bloques vecinos al actual PB bajo codificación, en concreto los bloques superior e izquierdo. Dichos píxeles son utilizados como muestras de referencia para la construcción de los 35 predictores que define el estándar.

La comunidad científica ha llevado a cabo distintas evaluaciones para conocer la eficiencia de HEVC en modo *intra-frame*. Los autores en [Li11], analizan la eficiencia de HEVC con respecto a H.264/AVC, mostrando que un ahorro en la tasa binaria del 25% puede ser alcanzado para la misma calidad objetiva. De modo similar, el estudio presentado en [Ngu12] comparando HEVC con otros estándares tradicionales de compresión de imágenes como son JPEG2000 [ITU-T T.800] y JPEG XR [Duf09], revela que HEVC mejora la eficiencia de compresión de dichos estándares en un 35% y 74% respectivamente, cuando se utiliza el modo de codificación *intra-frame* en HEVC.

Este incremento sustancial de la eficiencia de compresión, se debe en gran medida al conjunto de pequeñas mejoras realizadas sobre las distintas etapas que introduce el nuevo estándar, tal y como se resume a continuación:

- El incremento del número de modos angulares, que pasa de un máximo de 8 modos en H.264/AVC, a 33 modos en HEVC, permitiendo obtener una estimación mucho más precisa de los patrones direccionales que presenta la imagen.
- El incremento en el rango de tamaños de bloques de predicción, que incorpora los tamaños de 32x32 y 64x64 no disponibles en H.264/AVC, los cuales son muy eficientes con formatos de gran resolución como son los de HD y UHD.
- Mejoras en los filtros de pre-filtrado y post-filtrados de las muestras de referencia utilizadas para la construcción de los predictores, aplicados de modo selectivo en función de los modos angulares, evitando ciertas distorsiones que se hacen perceptibles en H.264/AVC.
- Mejora de la predicción *Planar*, utilizando uno esquema alternativo a la interpolación bilineal aplicada en H.264/AVC.
- Utilización de transformada DST alternativa a la DCT para la codificación del residuo de los bloques 4x4, permitiendo una compresión más eficiente del residuo.

2.3.1 Arquitectura de la Predicción Intra-frame

Como ya se ha descrito, la predicción *intra-frame* comprende un amplio conjunto de herramientas adicionales a la simple construcción de los predictores, que permiten obtener una elevada eficiencia de compresión, pero que requieren igualmente de un elevado coste computacional debido al elevado número de operaciones que exige cada etapa.

La Figura 2.8 muestra el diagrama de flujo del proceso de codificación *intra-frame* en HEVC, el cual se inicia con la selección de las muestras de referencia desde los píxeles de los bloques frontera y la reconstrucción de dichas muestras en caso de no estar disponibles. En función del tamaño del PB y del modo de predicción, y como paso previo a la construcción de los predictores, un pre-filtrado es aplicado a las muestras de referencia con el objetivo de evitar posibles artefactos en los píxeles reconstruidos. Igualmente, de modo selectivo para cada uno de los modos de predicción y tamaño de los PBs, un post-filtrado es aplicado al predictor con el objetivo de evitar discontinuidades que serían percibidas visualmente entre los bloques vecinos que utilicen modos con distinta orientación.

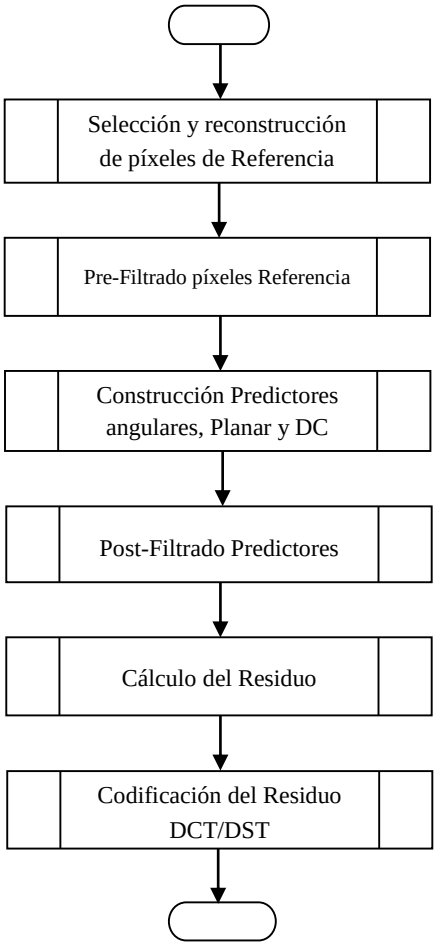


Figura 2.8. Diagrama de flujo de la predicción intra-frame en HEVC

Una vez construido el predictor para el conjunto de píxeles del PB, el residuo es calculado como la diferencia entre el predictor y el bloque original, y éste es codificado transformacionalmente por medio de la DCT, a excepción de los PBs de tamaño 4x4 que como ya se ha descrito en el apartado anterior, se les aplica una DST para obtener una codificación más eficiente del residuo como se demostró en [Sax11].

En la anterior Figura 2.8, puede comprobarse como el flujo de ejecución de las distintas etapas del algoritmo de predicción *intra-frame* es completamente lineal, lo que impide la introducir estrategias de paralelización entre procesos, dificultando su aceleración con este tipo de estrategias. En los siguientes apartados se describen los detalles de cada uno de los procesos que componen los bloques funcionales de la predicción *intra-frame*, y una información más detallada es descrita en [Lai12].

2.3.2 Selección y Reconstrucción de los Píxeles de Referencia

La selección de los píxeles de referencia utilizados para la construcción de los predictores *intra-frame*, difiere significativamente del modelo aplicado en H.264/AVC. A modo de ejemplo, la Figura 2.9 muestra los píxeles de referencia $d(x, y)$, sombreados en gris, utilizados para la construcción de los predictores en la codificación *intra-frame*, pertenecientes a los bloques vecinos al PB, representado por los píxeles $p(x, y)$ en color azul, y con dimensiones $NxN \mid N = 8$. El estándar define como píxeles de referencia para los 17 predictores direccionales verticales el píxel del vértice superior izquierdo, y los $2N$ píxeles superiores indicados en la Figura 2.9 en la fila superior al PB desde $d(0,0)$ a $d(2N, 0)$.

De modo similar, para la construcción de los 16 predictores angulares verticales se definen como píxeles de referencia los $2N$ píxeles del lateral izquierdo al PB, es decir, los indicados en la figura de $d(0,1)$ a $d(0,2N)$. Al igual que en H.264/AVC, las muestras de referencia $d(x, y)$ son las muestras previamente decodificadas en sus respectivos PBs, con el objetivo de que estos predictores puedan ser construidos de modo preciso en el decodificador, donde sólo están disponibles los píxeles.

Para los PBs donde los píxeles de referencia horizontales o verticales con índices de $d(x, y)$ superiores a N, no están disponibles, el estándar establece que dichos píxeles deben ser reconstruidos por simple repetición de la última muestra disponible, izquierda para la predicción vertical, o superior para la predicción horizontal. En el supuesto de que ninguno de los píxeles de referencia horizontal o vertical estuviera disponible, como ocurre en la primera fila y columna del frame, se reconstruyen todos los píxeles de referencia con el valor $2^{(b-1)}$, siendo b la profundidad de píxel de la imagen.

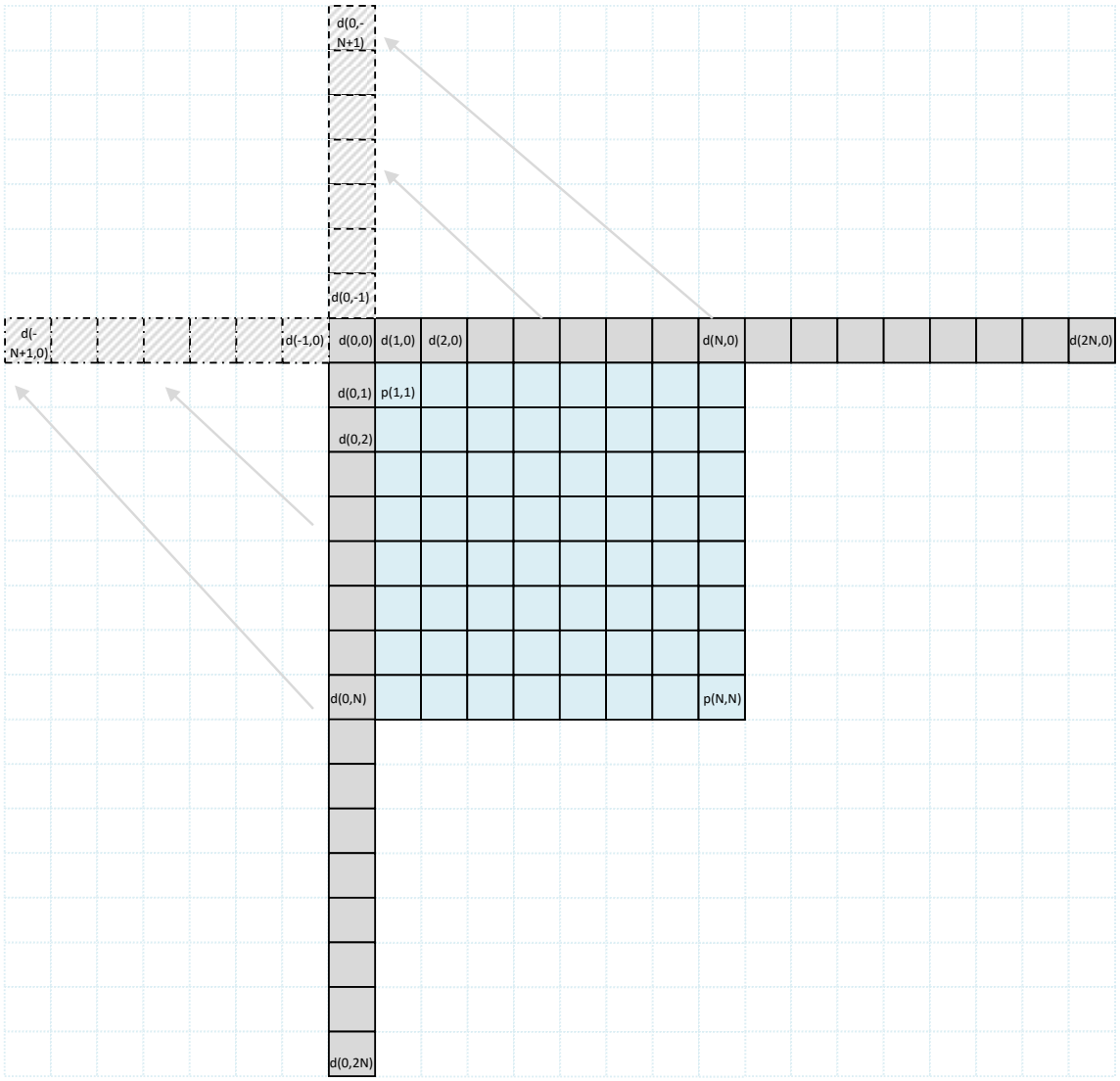


Figura 2.9. Muestras de referencia $d(x, y)$ utilizadas para la construcción de los predictores angulares del PB_{8x8}

Por último, como se puede apreciar en la Figura 2.9, en función del ángulo del predictor direccional pueden existir algunos píxeles de referencia que se situarían fuera de los $2N$ píxeles de referencia definidos en la fila superior o columna izquierda descritos anteriormente, siendo representados en la Figura 2.9 con trazas discontinuas en color gris.

En dicho escenario, el cálculo de las predicciones verticales donde el píxel de referencia requerido obtenga un índice negativo, $d(x, 0) \mid x < 0$, los píxeles de referencia ausentes se reconstruyen como proyección de las muestras de referencia horizontales $\mathrm{d}(0,1)\ a\ d(0,2N)$, utilizando para ello el mismo cálculo que se utiliza para calcular los predictores direccionales.

El mismo método es aplicable para los píxeles de referencia no disponibles en la predicción horizontal con $d(0,y) \mid y < 0$, reconstruyendo los píxeles de referencia por proyección de las muestras de referencia verticales disponibles, $d(1,0)$ a $d(2N,0)$.

2.3.3 Pre-filtrado de los Píxeles de Referencia

Con el objetivo de evitar los efectos de *contouring* que aparecen en las imágenes que presentan fuertes patrones direccionales, como se describe en [Wie03], HEVC lleva a cabo un pre-filtrado en las muestras de referencias, $d(x,y)$, de modo previo a la construcción de los predictores, el cual provoca un efecto de suavizado de dichos píxeles. HEVC utiliza un filtro de 3 taps caracterizado por el *kernel* $[1\ 2\ 1]/4$, que había sido utilizado en H.264/AVC en modo *intra-frame* para el pre-*filtrado* del tamaño de sub-bloques de $8x8$, de aplicación exclusiva para la familia de perfiles *High*.

La aplicación de este pre-filtrado se lleva a cabo de modo selectivo en función del tamaño del PB y del modo de predicción, sujeto a las siguientes restricciones:

- *Modo DC*: no se aplica el pre-filtrado para ningún tamaño de PB.
- *Modo Planar*: no se aplica el pre-filtrado para ningún tamaño de PB.
- PB_{4x4}: no se aplica el pre-filtrado para ningún modo de predicción.
- PB_{8x8}: solo se aplica a los modos con orientación diagonal H_2, V_{18} y V_{34}.
- PB_{16x16}: se aplica el pre-filtrado a todos los modos excepto a los modos horizontales H_9, H_{10} y H_{11}; y a los modos verticales V_{25}, V_{26} y V_{27}.
- PB_{32x32}: se aplica el pre-filtrado a todos los modos excepto al modo horizontal H_{10} y al modo vertical V_{26}.
- Los últimos píxeles de referencia, $d(0,2N)$ y $d(2N,0)$, no son pre-filtrados

Las PBs con tamaños 32x32, constituyen un caso especial en lo concerniente al pre-filtrado de los píxeles de referencia. Con el objetivo de reducir posibles distorsiones que se hagan apreciables visualmente, un filtro de suavizamiento más severo puede ser aplicado, el cual debe ser señalizo como metadata en la estructura denominada *Sequence Parameter Set* (SPS). Este filtro es aplicado si se detecta que la actividad de las muestras de referencia está por debajo de un umbral, el cual es calculado a partir de la derivada segunda sobre los píxeles extremos y central de las muestras de referencia en ambos sentidos, horizontal y vertical.

2.3.4 Post-filtrado de las muestras de predicción

Con el objetivo de mejorar la calidad perceptual de la predicción, mediante la reducción de la discontinuidad que en ocasiones puede aparecer entre los bordes de los PBs, como ha sido demostrado en [Min11], el estándar establece un post-filtrado sobre los píxeles que constituyen las muestras de predicción $P(x, y)$, el cual es aplicado de modo selectivo exclusivamente para las componentes de luminancia, y para los tres modos de predicción que se describen a continuación:

- *DC*: se filtran tan sólo las muestras correspondientes a los píxeles de predicción de la primera fila $P(x, 1)$ y columna $P(1, y)$, siendo filtradas con un filtro de 2-taps, definido por el *kernel* $[3\ 1]/4$, que se aplica entre los píxeles afectados y las muestras de referencia $d(x, y)$ colindantes a dichos píxeles.

- *H_{10}:* se aplica un filtro de gradiente basado en la suma a cada uno de los píxeles de predicción de la primera columna $P(1, y)$, la mitad de la diferencia entre el píxel de referencia $d(0,0)$ y su correspondiente píxel de referencia $d(0, y)$, como se describe en la Ecuación (2.4).

$$P(1, y) = P(1, y) + [d(0,0) - d(0, y)]/2 \quad (2.4)$$

- *V_{26}*: se aplica un filtro de gradiente basado en la suma a cada uno de los píxeles de predicción de la primera fila $P(x, 1)$, la mitad de la diferencia entre el píxel de referencia $d(0,0)$ y su correspondiente píxel de referencia $d(x, 0)$, como se describe en la Ecuación (2.5).

$$P(x, 1) = P(x, 1) + [d(0,0) - d(x, 0)]/2 \quad (2.5)$$

2.3.5 Predictores Angulares

Como se describió en el apartado 2.3, HEVC define 17 predicciones angulares verticales, con orientaciones que cubren un sector angular de 90º comprendidos entre 45º y 135º, y 16 modos angulares horizontales cubriendo igualmente un sector angular de 90º, comprendido entre 135º y 225º. Estos 33 modos angulares no han sido definidos con un mismo intervalo angular, por el contrario, han sido especificados por medio de los parámetros de desplazamiento $\pm dx$ para los modos horizontales, y $\pm dy$ para los modos verticales, con una precisión en ambos casos de 1/32 de píxel, con respecto al modo central horizontal, H_{10}, y al modo central vertical, V_{26}. Las Tablas 2.5 y 2.6 muestran los desplazamientos $\pm dx$ y $\pm dy$ para el conjunto de modos de predicción angular horizontales, H_2 a H_{17}, y verticales, V_{18} a V_{34}.

Tabla 2.5. Valores del parámetro de desplazamiento $\boldsymbol{dx}$ para los modos de predicción angular horizontales

Modo	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}	H_{12}	H_{13}	H_{14}	H_{15}	H_{16}	H_{17}
dy	+32	+26	+21	+17	+13	+9	+5	+2	0	-2	-5	-9	-13	-17	-21	-26

Tabla 2.6. Valores del parámetro de desplazamiento $\boldsymbol{dy}$ para los modos de predicción angular verticales

Modo	V_{18}	V_{19}	V_{20}	V_{21}	V_{22}	V_{23}	V_{24}	V_{25}	V_{26}	V_{27}	V_{28}	V_{29}	V_{30}	V_{31}	V_{32}	V_{33}	V_{34}
dx	-32	-26	-21	-17	-13	-9	-5	-2	0	+2	+5	+9	+13	+17	+21	+26	+32

La Figura 2.10, muestra gráficamente la distribución de modos angulares y su correspondencia con los respectivos valores de desplazamiento $\pm dx$ y $\pm dy$.

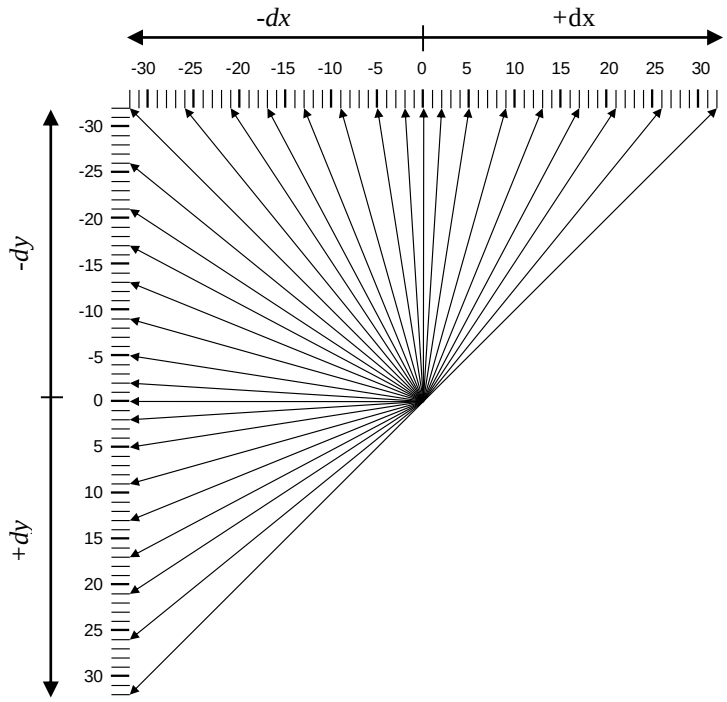


Figura 2.10. Correspondencia de los parámetros $\pm\boldsymbol{dx}$ y $\pm\boldsymbol{dy}$ con los modos de predicción angular

Para cada píxel $p(x, y)$ de un PB, el cálculo de su correspondiente predictor $P(x, y)$ se lleva a cabo en dos etapas, la primera de ellas donde se calcula el desplazamiento entero Δe, y posteriormente se calcula el correspondiente incremento fraccionario Δf, el cual es común a todos los píxeles de una fila para las predicciones verticales, o de una columna para las predicciones horizontales.

La Figura 2.11 muestra un ejemplo de cómo se obtienen los valores de $\Delta e + \Delta f$ por proyección del desplazamiento dx sobre la fila superior de píxeles de referencia $d(x, y)$, para un predictor vertical. El mismo concepto es aplicable para el cálculo de predictores horizontales.

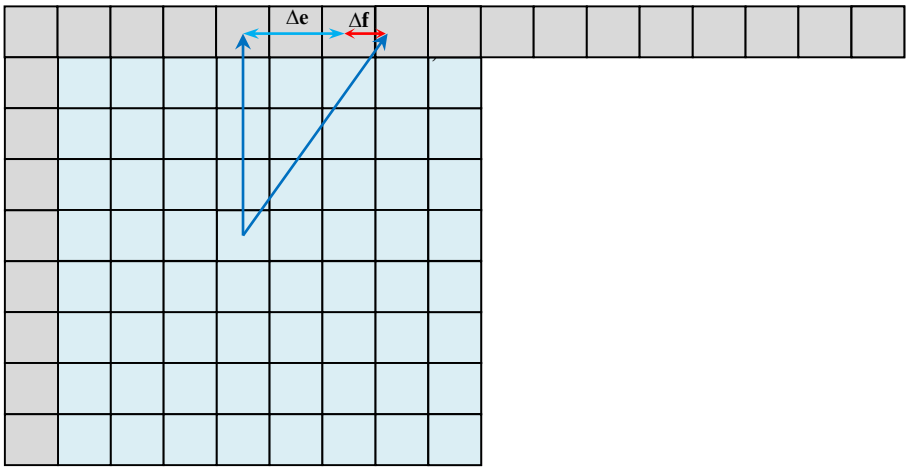


Figura 2.11. Ejemplo de la proyección de los desplazamientos enteros $\Delta \boldsymbol{e}$, y fraccionarios $\Delta \boldsymbol{f}$, para la predicción de modos verticales

De modo numérico, dichas proyecciones pueden obtenerse con muy bajo coste por medio de simples multiplicaciones, desplazamientos y funciones lógicas "*AND*", aplicando las Ecuaciones (2.6) y (2.7) para los predictores horizontales, y las Ecuaciones (2.8) y (2.9) para los predictores verticales respectivamente:

$$\Delta e_h = (x * dy) \gg 5 \quad (2.6)$$
$$\Delta f_h = (x * dy) \& 31 \quad (2.7)$$

$$\Delta e_V = (y * dx) \gg 5 \quad (2.8)$$
$$\Delta f_v = (y * dx) \& 31 \quad (2.9)$$

Finalmente, el predictor es construido por interpolación bilineal entre los dos píxeles de referencia situados entre la proyección de dx o dy, ponderados por el correspondiente valor fraccionario Δf obtenido, como se describe en las Ecuaciones (2.10) y (2.11), para los modos de predicción horizontal definido como $P_H(x, y)$, y vertical definido como $P_V(x, y)$:

$$P_H(x, y) = [(32 - \Delta f_h) * d(0, y + \Delta e_h) + \Delta f_h * d(0, y + \Delta e_h + 1) + 16] \gg 5 \quad (2.10)$$

$$P_V(x, y) = [(32 - \Delta f_v) * d(x + \Delta e_V, 0) + \Delta f_v * d(x + \Delta e_V + 1{,}0) + 16] \gg 5 \quad (2.11)$$

2.3.6 Predictores Planar y DC

En lo que respecta a los predictores no-direccionales, el modo *Planar* puede considerarse como una mejora del modo *Plane* definido en H.264/AVC que utiliza una predicción bilineal, y que se evidenció que provocaba ciertas distorsiones apreciables visualmente como discontinuidades entre los bordes de los bloques. Para solventar este problema, HEVC construye dicho predictor promediando dos predicciones lineales P_{H_Planar} y P_{V_Planar} con orientaciones horizontales y verticales respectivamente, definidas por en las Ecuaciones (2.12) y (2.13), siendo N es el tamaño del PB a codificar. Se define $P_{Planar}(x,y)$ como el predictor obtenido para cada uno de los píxeles del PB, conforme a la Ecuación (2.14):

$$P_{H\_Planar}(x,y) = (N-x) * d(0,y) + x * d(N+1,0) \quad (2.12)$$

$$P_{V\_Planar}(x,y) = (N-y) * d(x,0) + y * d(0,N+1) \quad (2.13)$$

$$P_{Planar}(x,y) = (P_V(x,y) + P_H(x,y) + N) \gg (log_2(N)+1) \quad (2.14)$$

El modo de predicción *DC* no presenta ninguna variación con respecto al definido en el estándar H.264/AVC, y por consiguiente su construcción se lleva a cabo como el valor medio de los píxeles de referencia disponibles, como se muestra en la Ecuación (2.15), siendo N el tamaño del PB:

$$DC = \frac{1}{2N}\left(\sum_{i=1}^{N} d(0,i) + \sum_{j=1}^{N} d(j,0)\right) \quad (2.15)$$

2.3.7 Predicción Intra-frame de las componentes de color

Debido a las particularidades que presentan las componentes de crominancia, con una gran concentración de bajas frecuencias, éstas solo pueden utilizar el mismo modo de predicción utilizado por la componente de luminancia, o bien uno de los dos modos no-angulares, *Planar* y DC, o los modos angulares ortogonales, H_{10} y V_{26}, y el modo diagonal V_{34}. La señalización de este último modo es un tanto particular, ya que sólo puede ser utilizada por la crominancia si la luminancia está utilizando uno de los cuatro modos indicados, *Planar* y DC, H_{10} y V_{26}, como se recoge en la Tabla 2.7.

Tabla 2.7. Modos de predicción *intra-frame* para las componentes de crominancia

Modo seleccionado	**Predictor**
0	Planar
1	V_{26}
2	H_{10}
3	DC
4	Mismo modo seleccionado por la luminancia
0-4	Modo V_{34} (Se debe señalizar uno de los modos 0 a 4 que coincida con el modo de luminancia)

2.3.8 Complejidad de la predicción intra-frame

Un aspecto relevante de la nueva arquitectura de codificación *intra-frame*, es la cuantificación de la complejidad requerida por sus algoritmos. Como ya se mencionó en el apartado 2.2.1, el conjunto de herramientas que componen este esquema de codificación deben ejecutarse de modo estrictamente secuencial, lo que impide la aplicación de estrategias de paralelización de las distintas etapas. Por el contrario, el proceso que se ejecuta sobre cada uno de los píxeles del PB no presenta ningún tipo de interdependencias con otros píxeles del bloque, por lo cual se puede llevar a cabo una paralelización a nivel de píxel con arquitecturas muti-cores como GPUs o similares.

Con el objetivo de caracterizar la complejidad computacional de la predicción *intra-frame* por píxel, la Tabla 2.8 muestra una estimación del número de operaciones aritméticas y lógicas que componen las distintas etapas de la predicción *intra-frame*, y que básicamente comprende el pre-filtrado y post-filtrado de las muestras de referencia, y la construcción de los predictores angulares y no-angulares.

Sin considerar el coste de las etapas de pre-filtrados y post-filtrados, se puede observar como el cómputo de los predictores angulares $P(x, y)$ puede ser representado como por la función $(ab + cd + 16)\text{››}5$, y por consiguiente requiere como mínimo de 2 multiplicaciones, 3 sumas y un desplazamiento, más el cálculo del desplazamiento entero y fraccionario ($\Delta e\ y\ \Delta f$), que implican 2 multiplicaciones, un desplazamiento y una operación lógica AND. En total se precisa de 3 sumas, 4 multiplicaciones, 2 desplazamientos y una operación lógica. Un estudio detallado de los costes computacionales de los distintos modos de predicción para las componentes de luminancia y crominancia para cada tamaño de PB, medidos sobre un decodificador software de HEVC optimizado, puede ser consultado en [Bos11].

Tabla 2.8. Estimación de la complejidad computacional por píxel en la predicción intra-frame en HEVC

Modo señalizado	Función	Sumas/Restas	Multiplicaciones	Divisiones	Operaciones L'ogicas	Desplazamientos
Pre-filtrado	$[1\ 2\ 1]/4$	2	1	1		
Desplazamiento entero (Δe)	Ec. 2.6, 2.8		1			1
Desplazamiento fraccionario (Δf)	Ec. 2.7, 2.9		1		1	
Predictor Angular	Ec. 10, 2.11	3	2			1
Predictor Planar	Ec. 2.12 - 2.14	7	4			1
Predictor DC	Ec. 2.15	2N-1			1	
Post-filtrado DC	$[3\ 1]/4$	1	1	1		
Post-filtrado H_{10} y V_{26}	Ec. 2.4, 2.5		2	1		
Calculo del residuo	$p(x,y) - P(x,y)$	1				

Si se comparan estos resultados con el cálculo de los modos de predicción angulares en H.264/AVC, representados como una función $(a + b + b + c + 2)\text{››}2$, donde únicamente se requieren 4 sumas y un desplazamiento, se puede apreciar cómo la complejidad de HEVC es muy superior a la de su predecesor. A todo ello, es preciso añadir el incremento de la complejidad de la predicción *Planar* y de los procesos de pre-filtrado y post-filtrado, así como el incremento del número de predictores y de los tamaños de bloques a evaluar para seleccionar el modo de predicción óptimo para cada CTB.

Como ya se describió en el primer capítulo, la reducción de la complejidad computacional de la predicción *intra-frame* constituye el principal objetivo fijado de esta Tesis, cuyas propuestas se describen en los siguientes capítulos.

2.4 IMPLEMENTACION DE LA PREDICCIÓN INTRA-FRAME EN EL SOFTWARE DE REFERENCIA DE HEVC

2.4.1 Modelo de Rate-Distortion Optimization

El incremento de la eficiencia de compresión que han experimentando los estándares de compresión de vídeo, ha ido acompañada de un significante incremento de su complejidad, debido en gran medida al incremento en el número de parámetros de codificación que pueden ser seleccionados. La selección óptima de este conjunto de parámetros es la clave para alcanzar la máxima eficiencia de compresión, entendida como aquella que satisfaga el equilibrio entre mínima distorsión y mínima tasa binaria. Este problema ha sido formulado como un ejercicio de optimización, en el que se busca los parámetros de codificación que minimiza la distorsión de codificación, sujeto a la restricción de que la tasa binaria, se ajuste a la tasa binaria fijada como objetivo de la codificación.

En [Ort99][Sul99] se propone el transformar este problema de minimización de la distorsión con restricciones (la tasa binaria), en un problema de minimización sin restricciones, definiendo una función de coste, J, que combina la tasa binaria, R, y la distorsión de codificación, denominada D, por medio del multiplicador de Lagrangian, λ. Este modelo es conocido como *Rate-Distortion Optimization* (RDO) en la literatura, y su expresión genérica se muestra en la Ecuación 2.16.

$$min\{\, J(parámetros\ codificación) = D + \lambda \cdot R \,\} \quad (2.16)$$

Como medida de distorsión pueden ser utilizadas distintas métricas aplicadas tradicionalmente en el campo del procesado y la codificación de imágenes como la MSE, la SAD o la SATD. Habitualmente, la selección de la métrica de distorsión se lleva a cabo en función de los requisitos computacionales disponibles, ya que algunas de ellas, como la MSE, pueden requerir la codificación y decodificación completa de la imagen, lo cual puede hacer que este modelo sea inviable en la práctica, si el número de posibles combinaciones de compresión es muy elevado. De igual modo, la obtención de la tasa binaria requerida en para la codificación con cada uno de los posibles parámetros, requiere de la codificación completa, lo cual puede ser igualmente costoso en ciertos escenarios. Por este motivo, en muchas ocasiones se utilizan métricas basadas en estimaciones de la distorsión y la tasa binaria, obteniendo una selección sub-óptima de los parámetros de codificación, lo cual reduce la eficiencia de comprensión.

Un factor clave para alcanzar en la eficiencia del RDO estriba en la selección del parámetro λ. En [Sul99] se ha demostrado que dicho parámetro de modo genérico es proporcional al

valor cuadrático del escalón de cuantificación, ΔQ, el cual es a su vez proporcional al parámetro de cuantificación QP.
En la Ecuación (2.17) se muestra dicha aproximación, siendo α es una constante que depende del modo de codificación el cual se desea optimizar:

$$\lambda = \alpha \cdot \Delta Q^2 \quad (2.17)$$

En [Ohm12] se ha demostrado que tanto en H.264/AVC como en HEVC, ya que ambos comparte el modelo de cuantificación, la relación entre el escalón de cuantificación y el parámetro de cuantificación es proporcional a $2^{(QP-12)/6}$, y por consiguiente la función de coste utilizada de modo genérico para la optimización del codificador HEVC queda determinada por la expresión indicada en la Ecuación (2.18):

$$J(parámetros) = D + \alpha \cdot 2^{(QP-12)/3} \cdot R \quad (2.18)$$

La forma tradicional de aplicar el modelo del RDO en la codificación de vídeo consiste en llevara a cabo una evaluación exhaustiva de la función de coste para todas las posibles combinaciones de parámetros disponibles en el codificador, y seleccionar aquella que obtenga el mínimo coste. Los codificador de nueva generación, como son el H.264/AVC y HEVC permiten la selección un elevado número de parámetros: el particionado del bloque de codificación, el tamaño del bloque de predicción, el tamaño de la transformada, el predictor en la codificación *intra-frame*, los MVs y la selección de múltiples frames de referencia en la codificación *inter-frame*, entre otros.

El número total de combinaciones posibles se obtiene como el producto del número de parámetros por el número de posibles valores que pueden adoptar cada uno de ellos, convierten al RDO en la etapa más costosa computacionalmente de un codificador. A su vez, el RDO es considerado como la etapa más crítica, ya que de la correcta selección de dichos parámetros depende en gran medida la eficiencia del codificador.

Este hecho ha despertado un gran interés en la industria y en la comunidad científica, motivado el desarrollo numerosas investigaciones con el objetivo de obtener nuevos modelos y algoritmos alternativos a la utilización de la fuerza bruta requerida por el modelo de RDO. Estos algoritmos son conocidos en la literatura como *Fast* o *Low Complexity*, ya que requieren de un menor coste computacional que el RDO, pero como contrapartida obtienen una selección de parámetros sub-óptima que repercute en una pérdida de eficiencia de codificación.

Como ya se ha descrito, los trabajos presentados en esta Tesis se encuadran en este tipo de investigaciones, y abordan la reducción de la complejidad del modelo de RDO utilizado para la predicción *intra-frame* en HEVC.

2.4.2 Arquitectura de la Predicción Intra-frame en el HM

En este apartado se describe la arquitectura de la predicción *intra-frame* implementada en el software de referencia de HEVC, denominado HM [JCT-VC Reference Software]. Esta predicción requiere de la selección de tres parámetros claves: la selección óptima del particionado de la CTU en PUs, la selección del modo de predicción óptimo para cada una de las PUs, y por último el particionado óptimo del RQT que define los tamaños de TU aplicados a cada PU.

La aplicación del modelo de RDO exhaustivo en la predicción *intra-frame* requiere de la evaluación de los 5 tamaños distintos que puede adoptar una PU ($2^N \; \forall \; N = 2,..,6$). Esto implica que para una CTU de 64x64 se deben evaluar 341 PUs ($\sum_{d=0}^{4} 4^d$). De igual modo, para cada una de las 341 PUs se deben evaluar los 35 predictores direccionales definidos en el estándar. Además, el residuo obtenido en cada una de estas 341x35 combinaciones, deben ser evaluados para un máximo de 3 tamaños distintos de transformada, con el objetivo de conocer cuál de ellos obtienen un menor coste.

Por todo ello, la codificación *intra-frame* en HEVC es considerada como una etapa de elevada complejidad, que ha motivado que el JCT-VC decidiera incluir en el software de referencia de HEVC una seria de algoritmos rápidos en su modelo de RDO, con el objetivo de reducir su coste computacional. En [Rui15d], se muestra como el modelo de RDO implementado en el HM consigue una aceleración del 66% con respecto a un RDO en el que se evalúan todas las combinaciones, obteniendo como contrapartida una ligera penalización en su eficiencia de compresión del 0.4%.

La arquitectura de la predicción *intra-frame* implementada en el software de HEVC sustituye el modelo de RDO con evaluación exhaustiva de todas las combinaciones por el esquema propuesto por Piao *et al.* en [Pia10], al que se le han incorporado ciertas optimizaciones propuestas en [Lia11], y cuya arquitectura se muestra en la Figura 2.12.

Este esquema lleva a cabo la evaluación de todos los tamaños de PUs, pero debido a la complejidad que requiere el evaluar todos los modos de predicción efectuando la codificación y decodificación completa para cada una de las combinaciones, procede a efectuar una pre-estimación de modos candidatos, por medio del cálculo de una función de coste de baja complejidad, como se describe a continuación.

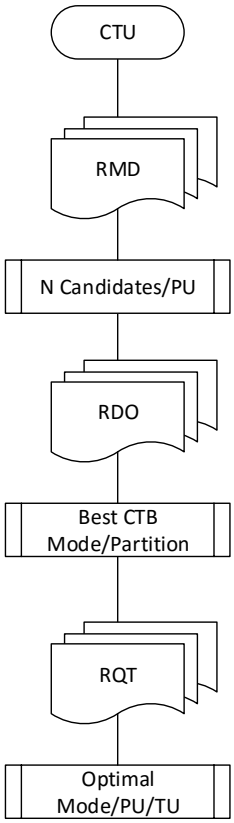


Figura 2.12. Arquitectura de la predicción intra-frame implementada en el HM

El algoritmo se inicia con una etapa que evalúa todos los modos de predicción para cada una de las PUs, denominada *Rough Mode Decision* (RMD), que utiliza una función de coste, J_{SATD}, que computa la distorsión en el dominio transformado por medio de la transformada de *Hadamard*. Su expresión se muestra en las Ecuaciones (2.18) y (2.19), siendo SATD la *Sum of Absolute Transform Difference*, λ_{pred} el multiplicador de Lagrangian y R_{pred} la tasa binaria utilizada en la señalización de dicho modo de predicción. La justificación del valor asignado a constante λ_{pred} es descrita en [JCTVC-K1002].

$$J_{SATD} = SATD + \lambda_{pred} \cdot R_{pred} \quad (2.18)$$

$$\lambda_{pred} = \sqrt{0.57 \cdot 2^{(QP-12)/3}} \quad (2.19)$$

El cálculo de la J_{SATD} no requiere el llevar a cabo el proceso completo de codificación y decodificación de la CTU, ya que tan solo efectúa la transformación entera del residuo para estimar la distorsión por medio de la SATD, por lo que el RMD es considerado como un modelo de RDO de bajo coste computacional. El algoritmo selecciona como modos candidatos los N modos cuya función de coste sea menor, siendo $N = 3$ para las PUs con tamaños de 64x64, 32x32 y 16x16, y $N = 8$ para PUs de tamaño 8x8 y 4x4.

Sin embargo, el modo óptimo podría no estar seleccionado entre los N candidatos seleccionados, al haberse utilizado métricas que proporcionan valores estimados de la distorsión y la tasa binaria.

Con el objetivo de mejorar la eficiencia del RMD, hasta 2 modos candidatos adicionales pueden ser añadidos como candidatos a los N modos previamente seleccionados, denominados *Most Probable Modes* (MPMs). Estos son derivados de los modos utilizados por las PUs vecinas, superior e izquierda, previamente codificadas. En la Tabla 2.9 se recoge el número total de modos que pueden ser seleccionados por el RMD para cada tamaño de PU, para el caso más favorable y el más desfavorable. En la cuarta columna se muestra el número mínimo de modos candidatos que pueden ser seleccionados, asumiendo que los MPMs ya han sido seleccionados como candidatos por el RMD, y en la última columna el número máximo de modos candidatos, en el supuesto de que ninguno de los MPMs haya sido previamente seleccionado por el RMD.

Table 2.9. Número de modos candidatos seleccionados por le RMD

PU	#Candidatos	#PUs	Mínimo #Modos	Máximo #Modos
64x64	3 ∪ MPMs	1	3	5
32x32	3 ∪ MPMs	4	12	20
16x16	3 ∪ MPMs	16	48	80
8x8	8 ∪ MPMs	64	512	640
4x4	8 ∪ MPMs	256	2048	2560
TOTAL			**2623**	**3305**

Como se puede observar, el número total de modos candidatos propuestos para ser nuevamente evaluados se sitúa entre 2623 y 3305 modos por CTU. Con el propósito de seleccionar un único modo de predicción para cada una de las 341 PUs, en la segunda etapa del algoritmo implementado en el software de referencia se calcula un RDO cuya función de coste, J_{mode}, es definida conforme a las expresiones mostradas en las Ecuaciones (2.20) a (2.22). Como métrica de distorsión se utiliza la distorsión real entre la PU original y la PU reconstruida, calculada por medio de la SSE para ambas componentes de luminancia, SSE_{luma}, y crominancia, SSE_{chroma}, siendo W_{chroma} un factor de ponderación aplicado a la distorsión de la componente de crominancia, y λ_{mode} el multiplicador de Lagrangian. Como métrica de tasa binaria en la función de coste se utiliza la tasa binaria real obtenida en la codificación de cada modo, R_{mode}.

$$J_{mode} = (SSE_{luma} + W_{chroma} \cdot SSE_{chroma}) + \lambda_{mode} \cdot R_{mode} \quad (2.20)$$

$$W_{chroma} = 2^{(QP - QP_{chroma})/3} \quad (2.21)$$

$$\lambda_{mode} = 0.57 \cdot 2^{(QP-12)/3} \quad (2.22)$$

Hay que destacar que si bien el RDO es mas exhaustivo que el RMD en su procedimiento, su decisión del modo de predicción no puede considerarse como óptima, al llevar a cabo la codificación y decodificación utilizando tan sólo como tamaño de transformada el tamaño de la PU bajo análisis, sin evaluar el coste para los distintos tamaños de particionados de TU que permite el estándar, con el objetivo de reducir su complejidad.

La decisión del tamaño de particionado se toma basándose en una estrategia *Bottom-Up*, que compara el coste de una $PU_{d,k}$ con la de la suma de los costes de las 4 sub-PUs ($PU_{d+1,4k+i}$ $\forall\, i = 0,..3$) derivadas de dicha $PU_{d,k}$. El RDO selecciona recursivamente la PU con menor coste J_{mode}, entre la PU de tamaño $2Nx2N$ y sus 4 sub-PUs de NxN conforme a la norma mostrada en la Ecuación 2.23.

$$IF \quad \left[J_{mode}\left(PU_{d,k}\right) \leq \sum_{i=0}^{3} J_{mode}\left(PU_{d+1,4k+i}\right)\right] \quad (2.23)$$

$$Best_{Partition} = 2Nx2N_{mode}$$

$$ELSE$$

$$Best_{Partition} = SPLIT\_NxN_{mode}$$

Finalmente, se lleva a cabo el cómputo del RQT, el cual codifica el residuo de predicción obtenido para el tamaño de PU y modo de predicción seleccionado por el RDO, para los tres niveles de profundidad del RQT permitidos. Durante el proceso de RQT, el tamaño óptimo de TU es obtenido por medio de la evaluación de la función de coste descrita en la Ecuación (2.20), seleccionando el particionado que la minimice.

Como se ha descrito en este apartado, el algoritmo de predicción *intra-frame* implementado en el HM, introduce distintas estrategias de aceleración con el objetivo de reducir su complejidad computacional. A pesar de ello, éste sigue considerándose como muy intensivo computacionalmente, por lo que su aceleración sigue siendo una de las principales demandas de la industria.

CHAPTER 3

FAST CODING TREE BLOCK PARTITIONING DECISION USING MACHINE LEARNING

As aforementioned in Chapter 2, the *intra-prediction* stage in the HEVC reference software [JCT-VC Reference Software] achieves high coding performance through the evaluation of all available combinations of CU sizes and prediction modes, selecting the one with the lowest rate distortion cost, which requires a huge computational task. In this Chapter, a low complexity algorithm for the CTU partitioning decision is proposed, which is the most computationally intensive task in the *intra-prediction* coding; while a novel low complexity mode decision will be proposed in the next Chapter. The proposed algorithm is based on the hypothesis that the optimal CTU partitioning decision in HEVC has a strong correlation with the content texture complexity distribution of CU and the corresponding four sub-CUs. In this thesis, *Machine Learning (ML)* tools are used to exploit that correlation and derive a set of decision trees to classify the CUs into *Split* or *Non-Split* classes. This approach uses a *top-bottom* scheme with three depth levels or nodes, comprising a decision tree in each node. The proposed architecture achieves a sub-optimal CTU partitioning map with low complexity, while maintaining the coding efficiency.

Experimental results show that the proposed approach reduces the *intra-prediction* complexity by over 50%, with no quality penalty in terms of the PSNR, and very nearly a 2% of bit rate increase, compared to the HEVC reference model. Finally, a performance comparison to state-of-the-art proposals shows that the proposed algorithm surpasses the best proposal in terms of time reduction, for the same coding performance penalty.

3.1 OBSERVATIONS AND MOTIVATION

As mentioned above, the HEVC *intra-prediction* can achieve a high level of performance compared to the traditional image coding standards, such as JPEG [ITU-T T. T.81], JPEG2000 [ITU-T T.800] and the new JPEG XR [ITU-T T.832], outperforming them by 40%. However, the high efficiency of the new HEVC *intra-prediction* algorithm requires a huge encoding burden due to the high number of combinations involved in the *intra-prediction* coding, comprising the evaluation of all possible prediction modes and the full range of CU sizes, which may not be suitable for real-time application scenarios.

The *Intra-prediction* scheme achieves optimal coding performance by using an exhaustive evaluation of the 35 prediction modes, and all possible partitions sizes from 64x64 to 4x4, which *means* the evaluation of 341 different blocks by CTU. Table 3.1 depicts the *intra-coding* complexity comparison between H.264/AVC and HEVC for an exhaustive search for the optimal mode/partitioning combination, revealing that HEVC requires five times more evaluations (15,610) than H.264/AVC (2,944). That high number of combinations derives from the number of prediction sizes (#PUs for HEVC and #Blocks for H.264/AVC), the number of prediction modes (#modes) and the number of different transform sizes (#TUs for HEVC and #Transforms for H.264/AVC).

Table 3.1. Complexity comparison between *intra-prediction* in HEVC and H.264/AVC, using an exhustive evaluation

	HEVC				H.264			
	# PUs	# Modes	#TUs	Total	# Blocks	# Modes	#Transforms	Total
64x64	1	35	2	70	NA	NA	NA	NA
32x32	4	35	3	420	NA	NA	NA	NA
16x16	16	35	3	1680	16	4	1(4x4)	64
8x8	64	35	2	4480	64	9	1(8x8)	576
4x4	256	35	1	8960	256	9	1(4x4)	2304
TOTAL				**15610**				**2944**

Based on these observations a number of proposals have been presented by the research community for the speeding up of the *intra-prediction* process. According to the algorithm approaches, proposals can be mainly classified into two categories, the first one reducing the number of prediction modes to be checked for the RMD, and eventually the number of

candidate modes in the RDO stage is also limited. The second category evaluates all the prediction modes but reduces the number of CU sizes required to be checked by both the RMD and the RDO stages, mostly by limiting the depth of the CU tree.

The proposed approach in this thesis is derived from the analysis of the computational complexity of the *intra-prediction* algorithm implemented in the HM reference software, which is composed of three stages: the Rough Mode Decision (RMD), the Rate Distortion Optimization (RDO) and the Residual Quad Tree (RQT), as was described in Section 2.4.2. For this purpose, one sequence from each class of the JCT-VC test sequences [JCTVC-L1100] has been encoded using the HM 16.6 reference model and the computing time of each *intra-prediction* stage has been collected.

Table 3.2 depicts the results for two QPs, QP22 and QP37, showing that the complexity of the RMD and RDO stages makes up over 80% of the total *intra-prediction* computation, and the remainder of the time is spent by the RQT stage and other auxiliary tasks. It is worth highlighting that most time consumption corresponds to the RDO, by over 60%, followed by the RMD, in the range of 17% to 25%, and lastly the RQT process with around 17%.

Table 3.2. Computational Complexity analysis of RMD, RDO and RQT stages in the HM 16.6

	QP22			QP37		
	RMD	RDO	RQT	RMD	RDO	RQT
Traffic	19.5%	63.1%	17.4%	27.1%	57.0%	15.7%
BQTerrace	15.9%	65.7%	18.3%	24.9%	59.2%	15.9%
PartyScene	13.6%	67.5%	18.8%	20.7%	62.1%	17.1%
BasketballPass	17.2%	65.9%	16.8%	26.0%	58.1%	15.8%
FourPeople	21.1%	62.1%	16.7%	27.6%	56.9%	15.4%
Average	**17.5%**	**64.9%**	**17.6%**	**25.3%**	**58.7%**	**16.0%**

This fact has motivated the approach presented in this chapter, which replaces the brute force scheme used in the HEVC reference model with a low complexity algorithm based on a fast CU size classifier, which was previously trained using *Machine Learning*. The classifier selects a sub-optimal CTU partitioning, thus avoiding the exhaustive evaluation of all available CU sizes.

This proposal is based on the hypothesis that the CU partitioning decision can be taken using the local CU features, considering that there exists a strong correlation between the optimal partitioning size and the texture complexity of the CU. It is well-know that homogeneous blocks achieve best performance when being encoded by large block sizes. Otherwise, highly textured blocks can be efficiently encoded by using small sizes which adjust their size to the details of the image. Figure 3.1 depicts the partitioning map obtained for the encoding of the first frame of the sequence *BasketballDrillTex* belonging to the JCT-VC test sequences, using the HM16.6 reference software.

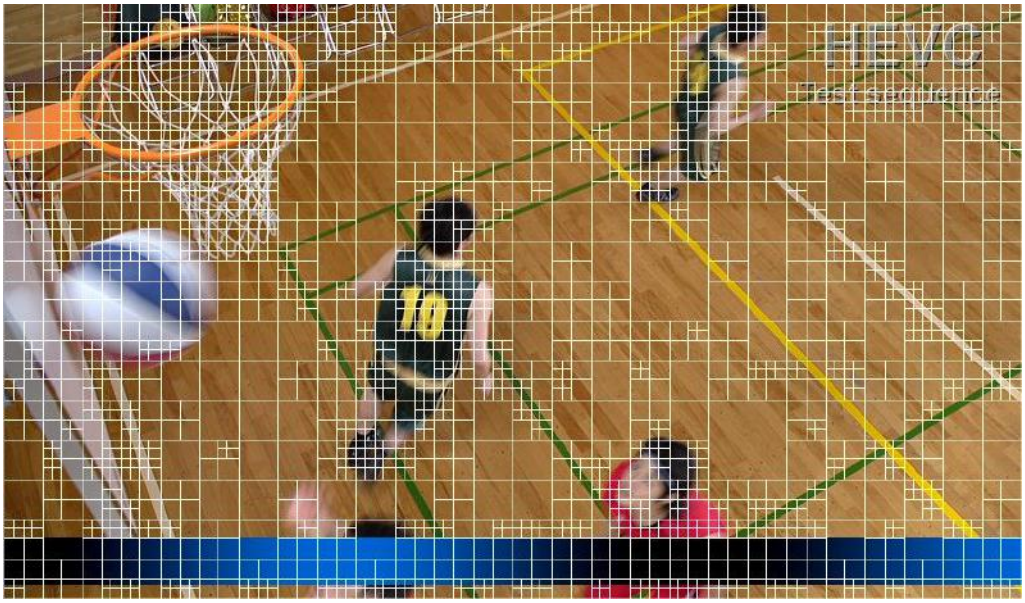


Figure 3.1. Partitioning map of the first frame of the sequence *BasketballDrillTex* using the HM16.6

As can be observed, exhaustive partitioning selects the smaller block sizes for the detailed elements of the image, such as the basket, the players, the court lines and the HEVC logo. Otherwise, for the flat areas such as the court floor, bigger CU sizes of 64x64 and 32x32 are more likely to be selected by the exhaustive RDO. However, it is worth noting that very similar homogeneous areas, such as the court floor in the centre of the image, achieve different partitioning structures, turning the partitioning decision into a non-trivial process.

Based on such observations, the algorithm proposed uses a binary classifier based on a *decision tree* with two classes, *Split* and *Non-Split*, applying, unlike the HM reference software, a *top-bottom* approach with one node for the top CU tree depths ($i = 0, .., 2$). The algorithm starts with the biggest CU size, namely 64x64, and using several attributes of the CU the *decision tree* takes the partitioning decision. If a CU has been split, the respective four sub-CUs are driven to the next node and a new binary classification is performed by the second level of the decision tree. The process is iterated till the last depth node level is reached, thus obtaining the final CU partitioning map.

The decision trees of each node are designed in a training stage by using *Machine Learning* methodology, which selects the number of the inner nodes of the decision trees and the rules applied in each inner node.

Because the *intra-prediction* stage does not have additional information regarding the CU feature, as happens in other processes such as transcoding or the inter-prediction stage, the feature selection for the training of the decision trees becomes the other key point of the proposal. There are several statistics that are commonly used in image processing and image

classification [Mar04] that can be extracted from a CU providing a high level of CU characterization. However, the computational complexity in the implementation of these metrics has to be considered in order to achieve a good balance between CU classification accuracy and the computational burden of the CU attributes.

The proposal presented in this chapter will use low complexity first-order spatial metrics such as the *mean and variance* of the CUs and sub-CUs which were selected from the training stage in the ML process. These metrics are computed with a very low complexity in a scalable way, and can be re-used from the top nodes to the bottom ones, as will be described in Section 3.3.6.

3.2 RELATED WORKS FOR FAST CTU PARTITIONING IN HEVC

In this section, the most relevant fast partitioning proposals for HEVC *intra-prediction* in the literature are introduced. Although CTU partitioning is a process that is highly dependent on other encoding decisions such as the angular prediction mode and residual quadtree partitioning, a large number of approaches have been proposed by the scientific community by addressing this issue as a standalone approach.

The proposals framed as fast CTU partitioning decisions are characterized by using similar approaches. All the *intra-prediction* modes are evaluated but only a small number of CU sizes are selected to be further checked by both the RMD and the RDO stages, mostly by limiting the depth of the CU tree, and thus they are usually named as *tree pruning* or *early termination* algorithms. Our approach fits in this group of proposals since our CU *decision tree* classifier determines the most probable CU sizes to be evaluated for both stages.

The proposals can be classified into two different categories. The first contains those in which the partitioning decision is taken exclusively on the basis of the RD cost value of the different CU sizes. The approaches in [Sun12][Hua13][Kim13][She13][Cen15] follow that schema, where the strategy used often achieves a negligible rate penalty because two depth levels for each CU are evaluated, at the expense of moderate complexity reduction, as is shown below. A workflow example of this approach is depicted in Figure 3.2, *d* being the tree depth and *k* the CU number at depth *d*.

The approaches in the second group [Tia12][Kha13][Cho13] mainly use the same architecture, but the optimal CTU partitioning prediction is based on some content features extracted from the CUs. Since our approach uses some CU attributes to select the CTU partitioning map, it can be framed in the second category.

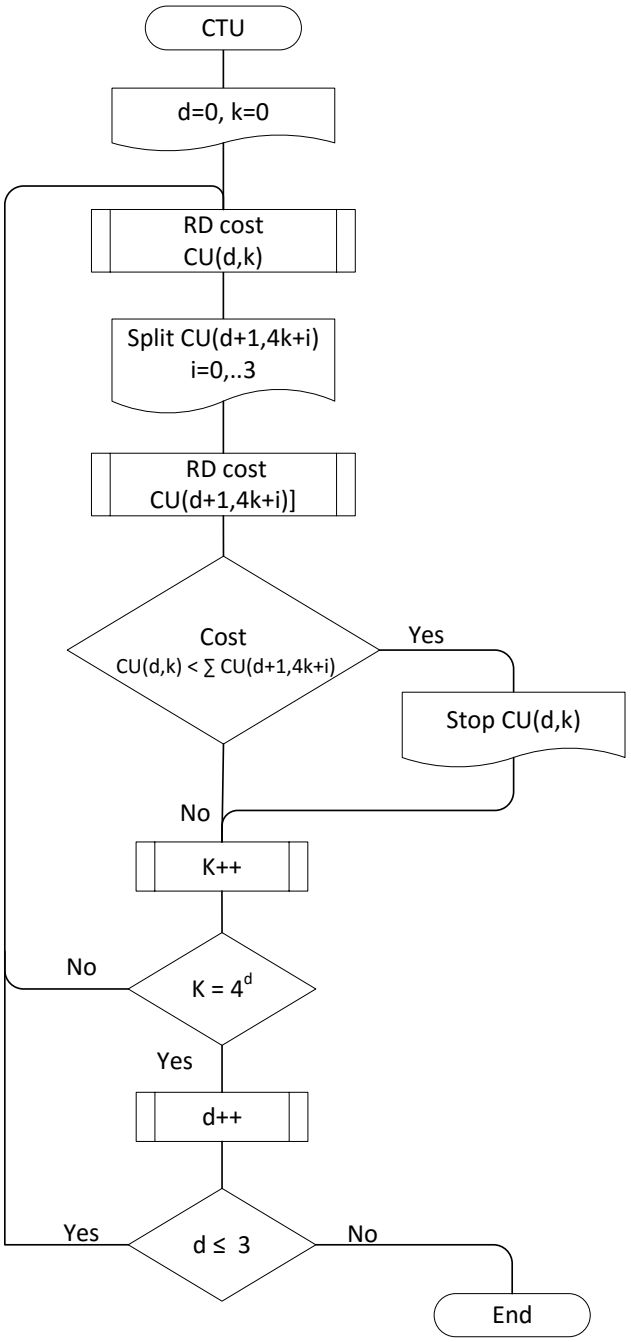


Figure 3.2. Workflow of the first category of tree pruning approaches using the RD cost

The authors in [Sun12] present a novel tree pruning approach using the RD cost difference between *Split* and *Non-Split* of a CTU, denoted as ΔC_{2N}. By applying an off-line training, they have defined an error function $E(T)$ in terms of RD cost using the $P(Split|\Delta C_{2N})$ and $P(Split|\Delta C_{2N})$ conditional probability distributions, as is shown in Equation (3.1), and a threshold *T* for the *Split* decision is derived. In the encoding process, if the ΔC_{2N} is lower than *T* the CU is *Non-Split*, instead the CU is divided in four sub-CUs, and the method is repeated for each of those sub-CUs.

$$E(T) = \int_{\Delta C_{2N}=0}^{T} P(Split|\Delta C_{2N}) + \int_{\Delta C_{2N}=T}^{\infty} P(Non-Split|\Delta C_{2N}) \qquad (3.1)$$

Huang *et al.* in [Hau13] proposed a straightforward bottom-up pruning scheme for the first tree depths (16x16 to 64×64), which decides that a $CU_{d,k}$ is not *Split* in a quad tree if its RD cost for the best mode (J_{mode}) is lower than the sum of RD costs of theirs four sub-CUs ($CU_{d+1,4k+i} \mid i=0,1,..3$). This algorithm achieves a moderate time saving of around 20% with a penalty of 0.46% in terms of increased bit rate. By using a similar approach, a fast *intra-prediction* algorithm is presented in [Kim13] based on the RD cost difference between the first and second candidate modes computed in the RMD stage. It reduces the number of RDO candidates to one mode (Best Candidate) or three modes (Best Candidate, DC, and MPM) instead of three or eight, as was proposed in the [Pia10] algorithm. In addition, if the RD cost of the best mode of the RDO stage is below a given threshold, the algorithm decides to apply an early termination tree, thus avoiding the evaluation of lower CU sizes. This algorithm achieves a better computational complexity reduction of 30.5%, but in return, it also obtains a higher rate increase of 1.2%.

Shen *et al.* [She13] suggested a tree pruning method based on the correlation of the depths of the neighboring CTUs from Up , Left , Up-Left, and Up-Right CTUs, and the depth of the current CTU, as is depicted in Figure 3.3 (dashed CTUs). By using weighting functions from the RD cost of the neighbouring CTUs, an early termination decision is applied in order to prune the tree depth. The results show a complexity time reduction of 21% with a moderate 1.7% rate increase. A similar approach is presented by Cen *et al.* in [Cen15], which also uses the same up and left neighboring CTUs to predict the CTU depth range of the current CU. The algorithm sets two depth ranges of [0-2] and [1-3], skipping only one depth level, and the computational burden reduction obtained is just 16%, with a significant 2.8% rate increase.

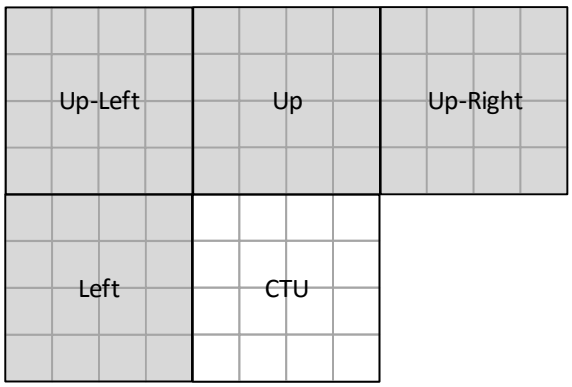


Figure 3.3. Neighbouring CTUs used for the depth estimation in the Shen *et al.* algorithm

Regarding the second category, the authors in [Tia12] proposed a fast partitioning decision based on CU texture complexity using a normalized metric of the logarithmic *variance*. The complexity is computed over a down-sampled version of the CTU with a 16×16 resolution (CU_{16}), as is shown in Equation (3.2), where E_{max} is the maximum energy of the 16x16 checkerboard CU according to Equation (3.3), being $log(\sigma^2[CU_{16}])$ the mentioned logarithmic *variance*. The 64×64 and 32×32 CU sizes are skipped if the complexity is above a predefined threshold; otherwise, it computes the complexity of the down-sampled version of 16×16 CUs with a resolution of 8×8. Again, the 8×8 and 4×4 partitions are skipped if the complexity is below a second predefined threshold. The simulation results show a similar time saving to that of [Kim13] of around 29%, with a 0.47% data rate penalty.

$$\mathrm{CTU}_{\mathrm{Complexity}} = \frac{\log(\sigma^2[\mathrm{CU}_{16}])}{\log(\mathrm{E}_{\max})} \tag{3.2}$$

$$\mathrm{E}_{\max} = \frac{1}{2}\sum_{j=0}^{16}\sum_{j=0}^{16}\left[(255^2 + 0^2) - \frac{1}{2}(255 + 0)^2\right] \tag{3.3}$$

The approach in [Kha13] proposes a bottom-up approach to determine the full partitioning map of the CTU using the CU *variance* as metric. Starting from the 4x4 CUs, the algorithm merges the four 4x4 sub-CUs into an 8x8 CU if any of them have a *variance* below a given threshold, repeating the process for quad 8x8 PUs into 16x16 PUs, and so on. For a reduced set of sequences, the algorithm obtains a complexity gain of 42% with a 1.2% rate increase.

Using a similar approach, Cho [Cho13] proposes a fast partitioning algorithm according to a Bayes decision method that assumes the RD cost to be a Gaussian random variable. Both, the Bayes decision and the prior classes (*Split* and *Non-Split*) need to be computed in a periodic learning phase *on the fly* over a set of training frames, in order to estimate a threshold for early CU splitting and early CU pruning decisions. This algorithm achieves a huge average time saving of 63.5% with a 3.5% rate increase. However, as mentioned above, during the on-line training phase, the encoding complexity is the same as the HM implementation, and therefore the encoding burden of training frames and non-training frames is highly unbalanced.

It should be noted that this proposal differs from the rest of the proposals in the sense that it obtains the thresholds for the partitioning decision in an *on-line* stage. That allows it to achieve huge computational burden reductions of over 60%, outperforming the other proposals by more than 20%. Nevertheless, during the *on-line* training phase, which could

take several frames, the encoding complexity is not reduced. It can become in a hindrance for real-time hardware architectures due to the highly unbalanced burden. In addition, in order to update the model and compute new thresholds, a scene cut detector is required to start a new on-line training at the beginning of the scene.

3.3 FAST CODING TREE UNIT PARTITIONING DECISION APPROACH

In this section, a novel coding tree unit partitioning decision algorithm based on ML techniques is presented, which is used as part of a low complexity *intra-prediction* algorithm in HEVC.

3.3.1 Machine Learning Methodology

Finding patterns in data is known by different names. *Data Mining* (DM) and *Machine Learning* are mainly used by statisticians, database researchers, and business communities, while Knowledge Discovery in Databases (KDD) is generally used to refer to the overall process of discovering useful knowledge from data, where *Data Mining* is a particular step in this process. ML can be considered as an extension of traditional statistical and data analysis approaches, by introducing analytical techniques from a range of disciplines such as numerical analysis, pattern matching, neural networks and genetic algorithms, among others.

Two types of ML approaches can be identified on the basis of the methodology used to build the model and the pattern finding.

- *Model building*: the objective is to produce an overall summary of a set of data to identify and describe the main features of the shape of the distribution [Han98]. Such models include a cluster analysis partition of a set of data, a regression model for prediction, and a tree-based classification rules.

- *Pattern detection*: this approach tries to detect small, relevant or unusual patterns of behaviour, which arises from anomalies such as discontinuity, noise, ambiguity, and incompleteness [Fay96]. Examples include objects with patterns of characteristics unlike others, sporadic waveforms in biomedical traces, or unusual spending patterns in credit card usage.

The ML methodology used in this work matches with the first approach by using the tree-based classification rules, which use the analysis of data sets in a training stage to create a set of rules which allow the taking of decisions in a *decision tree* [Wit05]. Since the inductive learning consists of obtaining generalization from samples, the data set characterization in terms of data size and distribution of the classes is the key factor in achieving a high precision classifier. The data set needs to have a sufficiently large number of examples to cover a wide range of representative patterns, and the classes of the training data set should be well-balanced in order to avoid performance classification, because the standard learning methods assume a balanced class distribution [Was10].

The ML models are obtained by recursively partitioning the data set space and fitting a simple prediction model within each partition. As a result, the partitioning can be represented as a decision tree. Classification trees are designed for dependent variables that take a finite number of unordered values, and the prediction error is measured in terms of misclassification cost [Loh11].

ML is being used for an extensive application spectrum, covering the scientific field, such as the prediction of climatic disasters, object recognition in computer vision, tumour diagnosis in medical images, neural and genetic classification, but ML is also becoming common in areas such as web search engines, face recognition in social networks, identification of purchasing habits, or even in the detection of review manipulation in social communities and e-business.

In the last few years the use of ML has also been applied to video coding and video transcoding processes [Fer08a][Fer10][Mar08][Jil09][Car10], proving that it is possible to find patterns in the image that describe image complexity and/or homogeneity, which can facilitate the decision made by the encoder. *Machine learning* is also considered a powerful tool in terms of complexity reduction because it can be implemented as a simple *decision tree* with a set of rules; implemented with low computational cost instructions (*if-else).*

Fernández-Escribano *et al.* in [Fer08a][Fer10] propose a non-traditional use of ML for MPEG-2 to H.264/AVC video transcoding, showing that the encoding complexity can be reduced without significant quality losses. In addition, Distributed Video Coding complexity reduction was proposed in [Mar08] by using ML.

In [Jil09], a novel *Machine Learning* scheme is applied to reduce the complexity of *intra-prediction* coding in H.264/AVC. With a similar approach, a fast H.264/AVC *inter-picture* prediction algorithm is reported in [Car10], which uses *Data Mining* for the encoding complexity reduction, achieving 65% with a negligible bit rate penalty. As has been

described, ML methodology has been proved as a useful tool for MPEG-2 and H.264/AVC video coding, but not for HEVC intra-prediction.

Hence, the approach presented in this section uses *Machine Learning* methodology to obtain a CTU classifier, which makes it possible to take a binary decision to *Split* or *Non-Split* a CU, avoiding the huge burden involved in the exhaustive evaluation of CU sizes and prediction modes. A *decision tree* is modelled as a set of multilevel inner-nodes, and each node has a rule which represents a decision based on the value of the input variables, and the thresholds defined in the training stage for each inner-node. Machine learning allows the building of a *decision tree* using knowledge acquired in the training phase from the training set. Thus the ML approach can address the CTU partitioning decision issue in HEVC with low computational complexity, since the inner-rules can be implemented as simple comparisons between a variable and a threshold.

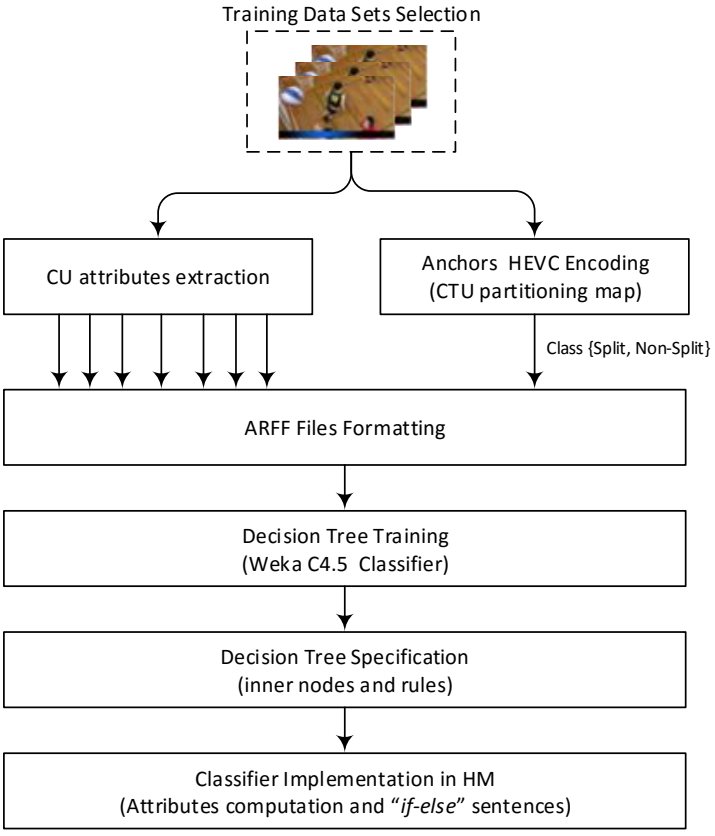


Figure 3.4.Machine Learning methodology used for the design of the decision trees for the proposed algorithm

The fast *intra-prediction* algorithm addressed in this section proposes the use of ML to classify each CU size in the *Split* or *Non-Split* class using a decision tree, avoiding the full evaluation of the compute-intensive algorithms which comprise *intra-prediction* (RMD, RDO and RQT). The variables for the classifier are obtained from the features of the CU under evaluation. The feature selection is based on the observed behaviour that a CU with

rich textured regions tends to be chosen for *Split*, whereas CUs with homogeneous regions are often chosen for non-split.

As was mentioned in Section 3.2, there are a large number of local statistics commonly used in image processing and image classification which can be extracted from the CUs. However, a trade-off between statistical accuracy in the CU characterization and the computational burden of the metric used should be carefully considered.

The ML methodology used for the design of the proposed algorithm is depicted in Figure 3.4, and can be described as follows:

a. Training Data Sets Selection: with the aim of obtaining representative data sets for the training process, a set of test sequences were selected using the Spatial Information and Temporal Information metrics as criteria, according to the ITU-T P.910 recommendation [ITU-T P.910]. The test sequences selection and the size of the training data sets selection are described in Section 3.3.2.

b. CU Attributes Extraction: The attributes selection for the training stage is a key factor in obtaining a high accuracy classifier. Such selection is based on relevant data, and a small set of attributes are chosen for the construction of the decision rules, as is explained in Section 3.3.3.

c. Anchors HEVC Encoding: with the aim of obtaining the anchor of the optimal partitioning map of each CTU, the test sequences were encoded using the HEVC reference model. The partitioning map has been obtained for the four distortion levels QP22, QP27, QP32 and QP37 recommended in the "*Common Test Conditions and Software Reference Configurations*" (CTC) by the JCT-VC [49] .

d. ARFF File Formatting: The CU attributes are ordered and combined with their correlative CU class obtained from the partitioning map, comprising an instance for the training stage. Those data are formatted into a file in accordance with the *Attribute-Relation File Format* (ARFF) [Wit05], whose structure will be presented in Section 3.3.4.

e. Decision Tree Training: the tree training was performed using the *Waikato Environment for Knowledge Analysis* (WEKA) Version 3.6.10 [Wit05] *Machine Learning* tool. WEKA includes a set of ML algorithms such as classifiers, regression and clustering, but it also has several attribute tools for data pre-processing, attribute selection and data visualization. WEKA is open source software issued under the *General Public License (*GNU).

f. Decision Tree specification: the C4.5 classifier proposed by Ross Quinlan [Qui93] and implemented in WEKA was used for the decision tree specification, which includes a set of interconnected inner nodes and a set of rules, with a threshold for each rule. Only one attribute is evaluated in each inner node, and the rule is based on a simple comparison between the value of the attribute and a threshold which defines the rules. The details of the *decision tree* specifications are described in Section 3.3.5.

g. Decision Tree Implementation in HM: once the *decision tree* accuracy has been measured by applying a 10-fold cross-validation [Efr83], the decision rules are converted to simply "*if-else*" sentences and they are implemented in the HM reference software. Details of ML classifier implementation, including the computation of the thresholds for the non-trained QPs, are described in Section 3.3.6.

Taking into account the fact that each CU size could need a specific *decision tree* which best describes its features, the ML methodology described above has been carried out once for each available size of 64x64, 32x32, and 16x16. Consequently, we have obtained three different decision trees, which we have called nodes of decision, denoted as *Node64*, *Node32* and *Node16*. Each of those nodes have their specific attributes, inner nodes and rules, but all of them have the same aim, the decision of splitting or not splitting a CU into its respective sub-CUs.

3.3.2 Training Data Set

With the aim of obtaining a training set covering a wide range of content complexities, the Spatial Information (*SI*) and Temporal Information (*TI*) metrics were computed for all JCT-VC test sequences [JCTVC-L1100], according to the ITU-T P.910 recommendation [ITU-T P.910]. *SI* is a good indicator of the edge energy commonly used as the basis for estimating image complexity. Honghai and Winkler in [Hon13] showed that *SI* measures strongly correlate with compression-based complexity measures. The *SI* metric is suitable for image activity characterization in image and video quality classification, and a number of video quality metrics use it for that purpose [Pin04][Kor10].

It is useful to compare the relative *SI* and *TI* metrics found in the test sequences, since generally compression difficulty is directly related to both spatial complexity and the temporal variability of the pictures of the video sequence. Eight sequences were chosen: two Class A sequences (*Traffic* and *PeopleOnStreet*), four Class B sequences (*ParkScene*,

Cactus, *BQTerrace*, and *BasketballDrive*), and two Class F sequences (*BasketballDrillText* and *SliceEditing*) which have hybrid content and non-camera-captured content (synthetic content from a computer screen), respectively. Figure 3.5 shows the first frame of the 8 selected training set sequences.

Figure 3.5. First frame of training set sequences

Figure 3.6 shows the spatio-temporal (*ST*) information of the selected training set sequences, where low values of the vertical axis correspond to sequences with very limited motion and high values indicate that a sequence contains scenes with a lot of motion. For the horizontal *SI* axis, low values correspond to scenes having minimal spatial detail and high values are found in scenes having significant spatial textured areas. As can be observed, the selected sequences cover the diagonal area from very low SI and TI values (*ParkScene, Cactus*, and *BasketballDrive*) to very high SI and TI complexity (*Traffic*, *PeopleOnStreet*, and *SliceEditing*), as well as medium complexity (*BQTerrace* and *BasketballDrillText*).

Only the first picture of each selected sequence is used to training the classifier, which is considered enough to achieve a high number of representative CTU samples of both, blocks with homogeneous areas and blocks with highly detailed textured areas.

The trained sequences were used for the extraction of attributes from the CUs, but they are also encoded with the HM encoder with the aim of achieving the optimal partitioning map of each CTU which will be used as anchors. The partitioning map has been obtained for the four distortion levels QP22, QP27, QP32 and QP37 recommended in the "*Common Test Conditions and Software Reference Configurations*" (CTCs) by the JCT-VC [JCTVC-L1100] .The partitioning maps allow us to classify each CU in a binary class that take the *Split* or *Non-Split* value for each QP in an independent way. The attributes information from each CU is ordered, and they are combined with their correlative CU class comprising an

instance for the training stage. Those data are formatted into a file according to the ARFF format, which is the default input in the WEKA tool for the training of the decision tree.

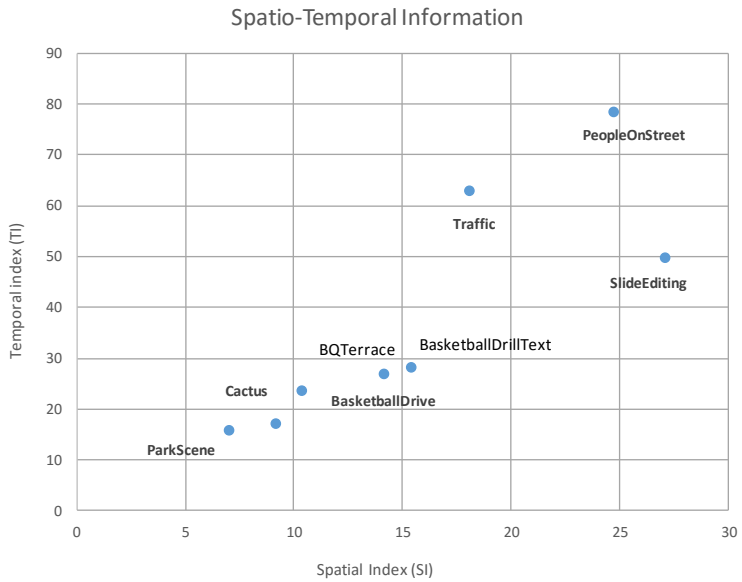


Figure 3.1. ST information of the training sequences

5.3.3 Attributes Selection

Attributes selection aims to choose a subset of features which leads to the best performance in the CU classification. Many features can be extracted from a CU to describe its content using the first and second order metrics in the spatial domain, such as the *Mean*, *Standard Deviation*, *Skewness*, *Kurtosis*, *Entropy*, *Autocorrelation*, *Inertia*, or *Covariance* [Pra01]. There are also useful attributes describing the CU features which can be computed in the Fourier domain, such as the vertical and horizontal energy, the DC energy, or the *mean* and *variance* of the AC coefficients.

Taking into account that the proposed approach defines the classes *Split* and *Non-Split*, a high accuracy classification of the CU can be achieved by comparing some features from CU with *2Nx2N* size and the four *NxN* sub-CUs. Initially, we extracted a large number of statistics commonly used in image processing and image classification, clustered in the following categories:

- Spatial domain attributes: *mean* of the CU, *variance* of the CU, horizontal and vertical *variance* of the CU, *variance* of the *means* of the four Sub-CUs, the *variance* of the *variance* of the CUs, and CU entropy.

– Transformed domain attributes: the energy of the AC coefficients of the Discrete Cosine Transform (DCT) of the CU, the number of non-zero AC coefficients from the DCT of the CU, the horizontal and vertical energy of the AC coefficients from the DCT (first row and column coefficients), and the *mean* and *variance* of the DC coefficients from the four sub-CUs.

WEKA provides a set of tools for attribute selection that use different strategies to rank the usefulness of the attributes, denoted as *Attributes Evaluator*, in conjunction with different search algorithms. In our training process, we have used the *GainRatioAttributeEval* tool, which evaluates the worth of an attribute by measuring the gain ratio with respect to the class, together with the ranker search method. The results of the attributes selection reported that both sets of attributes, spatial and transformed, achieve quite similar ranking. However, the domain transformed attributes require much more computation than the spatial ones. It should be noted that unlike other ML approaches, in our proposal the attributes are not implicitly calculated in the encoding process, so they have to be specifically computed to be available for the classifier, thus the time consumption is also a key factor in attribute selection.

Considering both factors, attribute ranking and computational complexity, we finally selected the three attributes from the spatial domain with the best performance in terms of ranking evaluation, which are based on the *variance* and *mean* computation, which are depicted in Figure 3.7, and are the following:

– The *variance* of the *2Nx2N* CUs, denoted as σ^2_{2N}

– The *variance* of *variances* of the *NxN* 4 sub-CUs, denoted as $\sigma^2[\sigma^2_n]$

– The *variance* of the *mean*s of the 4 *NxN* sub-CUs, denoted as $\sigma^2[\mu_N)]$

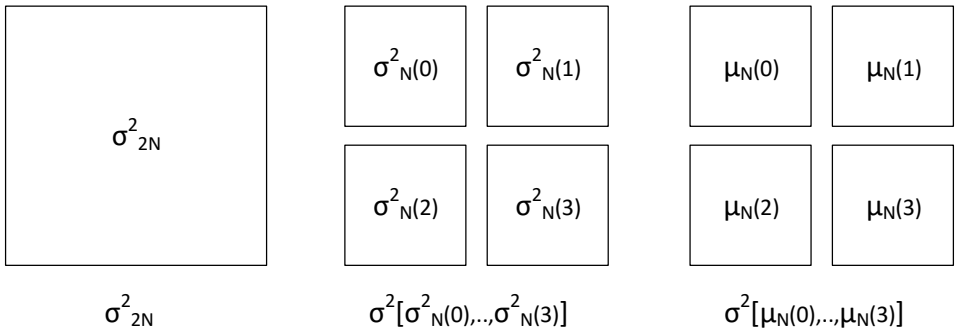


Figure 3.2. Selected attributes for the training of decision trees

The attributes selection is coherent with the well-known observation that homogeneous areas in the image tends to be intra-coded by large blocks, which have low *variance*, whereas rich and complex textured ones are usually coded by small blocks with high *variance* values.

In order to reduce the complexity of the *variance* computation, it is computed by the *variance* algorithm named *textbook one-pass* described by Chan *et al.* in [Cha79], which does not require passing through the data twice, once to calculate *mean* and again to compute sum of squares of deviations from *mean*. Equation (3.4) shows the generic *textbook one-pass variance* computation, for a data set of *N* samples, denoted as x_j:

$$\sigma^2[\mathrm{x}] = \frac{1}{N}\left[\sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{x_j}^2 - \frac{1}{\mathrm{N}}\left(\sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{x_j}\right)^2\right] \tag{3.4}$$

By adapting the Chan *et al.* algorithm to the three selected attributes σ_{2N}^2, $\sigma^2[\sigma_N^2]$, and $\sigma^2[\mu_N]$, Equations 3.5 to 3.9 show the low complexity *variance* computation used in this approach, *p(i,j)* being the pixels belonging to the *NxN* or 2Nx2N CUs respectively, and *k* the number of sub-CU, $\forall k = 0,..,3$:

$$\sigma_{\mathrm{N}}^2 = \frac{1}{N^2}\left[\sum_{\mathrm{i}=0}^{\mathrm{N}-1}\sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{p(i,j)}^2 - \frac{1}{N^2}\left(\sum_{\mathrm{i}=0}^{\mathrm{N}-1}\sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{p(i,j)}\right)^2\right] \tag{3.5}$$

$$\mu_{\mathrm{N}} = \frac{1}{N^2}\left[\sum_{\mathrm{i}=0}^{\mathrm{N}-1}\sum_{\mathrm{j}=0}^{\mathrm{N}-1} \mathrm{p(i,j)}\right] \tag{3.6}$$

$$\sigma_{\mathrm{2N}}^2 = \frac{1}{2N^2}\left[\sum_{\mathrm{i}=0}^{\mathrm{2N}-1}\sum_{\mathrm{j}=0}^{\mathrm{2N}-1} \mathrm{p(i,j)}^2 - \frac{1}{2N^2}\left(\sum_{\mathrm{i}=0}^{\mathrm{2N}-1}\sum_{\mathrm{j}=0}^{\mathrm{2N}-1} \mathrm{p(i,j)}\right)^2\right] \tag{3.7}$$

$$\sigma^2[\sigma_{\mathrm{N}}^2] = \frac{1}{4}\left[\sum_{\mathrm{k}=0}^{3} [\sigma_{\mathrm{N}}^2(\mathrm{k})]^2 - \frac{1}{4}\left(\sum_{\mathrm{k}=0}^{3} \sigma_{\mathrm{N}}^2(\mathrm{k})\right)^2\right] \tag{3.8}$$

$$\sigma^2[\mu_{\mathrm{N}}^2] = \frac{1}{4}\left[\sum_{\mathrm{k}=0}^{3} [\mu_{\mathrm{N}}(\mathrm{k})]^2 - \frac{1}{4}\left(\sum_{\mathrm{k}=0}^{3} \mu_{\mathrm{N}}(\mathrm{k})\right)^2\right] \tag{3.9}$$

3.3.4 ARFF Files Formatting

The initial attributes extracted from the training set are formatted to default WEKA ARFF files [Wit05], which is an ASCII text file showing the relationship between the attributes and their classes. The ARFF file is structured in two sections; the attributes declaration and the data section. The attributes declaration is a header which includes the attribute names and types, which can be binary, numerical o nominal. The data section comprises the raw data of attributes, ordered as one instances per row, with one attribute per column, where the class attribute is always the last column.

Figure 3.8 exemplifies an ARFF file comprising nine numeric attributes, where the tenth attribute is the nominal *Non-Split* and *Split* class, denoted as 2N and N respectively.

```
@relation NODE64_QP22

@attribute EnerAC_V numeric
@attribute EnerAC_H numeric
@attribute DC_DCT numeric
@attribute NZACCoeff numeric
@attribute VarAC_V numeric
@attribute VarAC_H numeric
@attribute Var2N numeric
@attribute VarVarN numeric
@attribute VarMeanN numeric

@attribute Class {2N, N}
-----------------------------------
@DATA
0,59,294,169,8,20,140,7675,30,N,
33,25,291,163,9,6, 1513,267626,271,N,
29,50,290,179,7,25, 1968,472420,102,N,
```

Figure 3.3. Example of ARFF file.

3.3.5 Decision Trees Specification

As mentioned above, the training process was carried out using the open source WEKA, an effective *ML* tool which includes a large number of tree classifiers, such us the *C.4.5*, *RandomForest*, *REPTree* and *LMT*, among others. This tool also includes filters for instance preprocessing, the ranking of attributes and data visualization. Prior to the training of the decision tree, and considering that it is one of the key factors in *Machine Learning*, we proceeded to classifier selection. From among several classifiers, *C4.5* [Qui93] is a well-known classifier for general-purpose classifying problems [Lon13] as well as for video coding purposes [Fer08b]. It uses the *Kullback–Leibler* divergence (or relative-entropy), which is an asymmetric dissimilarity measure between two probability distributions, "*p*" and "*q*". *C4.5* works by iterating all training instances, and it searches for the threshold for each

attribute which achieves the optimal classification recursively. Consequently, we have used the *C4.5* classifier for our approach.

The most common mechanism for measuring the trained *decision tree* accuracy is the 10-fold cross-validation process available in WEKA, which provides the prediction error measured in terms misclassification instances or percentage of *Correctly Classified Instances* (CCI). The following sections present the decision tree topology and the classification accuracy for each decision node, *Node64*, *Node32* and *Node16*.

3.3.5.1 Decision Tree Specification for Node64

This section discusses the proposed low complexity CTU partitioning decision for the first decision node, *Node64* [Rui14a] [Rui15b]. This node is the most critical because a wrong *Split* or *Non-Split* decision at the highest partitioning level can cause a 64x64 CTU that should be divided into several smaller CUs not to be split, and therefore the compression efficiency will be reduced, increasing the penalty in terms of bit rate and quality. In contrast, a correct classification of the *Non-Split* CTUs can achieve a high complexity reduction, thus avoiding the evaluation of a huge number of smaller CUs (32x32, 16x16 and 8x8).

The following sections describe the data set size, its class distribution and the set of attributes selected in the training process for *Node64*, as well as the *decision tree* specification in term of inner nodes and rules.

3.3.5.1.1 Classifier Training

As was mentioned in Section 3.3.2, the data set is composed of the 8 test sequences comprising a total of 4,231 64x64 CTUs, and therefore that is the data set for *Node64*. The last column of Table 3.3 shows the data set size for this node. The attributes of each 64x64 CU are an instance for the classifier, which finds the correlation between the CU's features and its class.

A further relevant aspect of the selected data sets in ML is the class balance of the data, thus the class distribution of the training set for the *Node64* is analysed for the four QPs recommended in the CTCs. As can be observed in Table 3.4, data sets suffer the well-known imbalance problem because there are significantly more instances belonging to *Split*, by over 80%, than *Non-Split* with just 8% for the QP22.

Table 3.3. Data set size for *Node64*

Training Sequences class	Number of Sequences	Horizontal resolution	Vertical resolution	Data set size for *Node64*
Class A	2	2560	1600	2000
Class B	4	1920	1080	1920
Class F (BasketballDrillText)	1	832	480	91
Class F (SliceEditing)	1	1280	720	220
TOTAL				**4231**

This issue has been addressed in the last few years by the research community, and it is assumed that a classifier that attempts to correctly classify the minority class will likely achieve poor accuracy [Jap00]. The most common technique used to obtain a balanced distribution is to apply an undersampling to the majority class, or on the contrary an oversampling to the minority class. Hulse *et al.* in [Hul07] show that the *Random* undersampling method is one of the best algorithms to address the imbalance issue, thus previously to the training of the *decision tree* for *Node64*, an "*Unsupervised instance random subsample*" filter available in WEKA has been applied. That filter produces a random subsample of a data set using either sampling with replacement or without replacement. Finally, a balanced data set is obtained and the tree is trained with the balanced class data set.

Table 3.4. Class distribution of training instances for the *Node64* decision tree

	Node64			
Class	QP22	QP27	QP32	QP37
Split	91.73%	87.17%	85.91%	81.00%
Non-Split	8.27%	12.83%	14.09%	19.00%

3.3.5.1.2 Decision Tree Specification

The training results obtained after the 10-fold cross-validation step for the Decision Trees of the *Node64* are shown in Table 3.5, collecting the CCIs, the trained tree size and the number of leaves of the decision trees. As can be noted, the CCIs are over 90%, which can be considered a high accuracy classification. However, it is worth noting the huge dispersion that exists between the tree topologies for the different QPs, with a tree size in the range of 7 to 13, and the number of leaves from 3 to 7. The use of different *decision tree* topologies for different QPs would make the design unapproachable, and consequently tree topology fusion was performed in a trade-off between accuracy and losses. There is not an automatic tool to address this work in WEKA, thus that work was carried out empirically by analysing the rules and the accuracy in each inner node.

A pruning process was also carried out with the aim of reducing the implementation complexity. As a final result, the *decision tree* topology for each QP was unified, but the threshold used in each inner node for each QP is kept according to the value obtained in the training.

Table 3.5. *Decision tree* accuracy and topology for the *Node64*

	Node64			
	QP22	QP27	QP32	QP37
CCI	92.86%	90.54%	91.08%	89.83%
Tree size	7	9	5	13
# Leaves	4	6	3	7

Attribute evaluation carried out in the training stage selected as the most relevant attributes the *variance* of the 64x64 CU, denoted as σ_{64}^2, the *variance* of the *variances* of the four 32×32 Sub-CUs, denoted as $\sigma^2[\sigma_{32}^2]$, and computed as $\sigma^2\left[\sigma^2\left(CU_{1,k}\right)\ \forall k = 0,..,3\right]$, and the *variance* of *means* of the 32×32 sub-blocks, denoted as $\sigma^2[\mu_{32}^2]$ and computed as $\sigma^2\left[\mu\left(CU_{1,k}\right)\ \forall k = 0,..,3\right]$. The classifier determined that a CU is homogeneous when σ_{64}^2, $\sigma^2[\mu_{32}]$ and $\sigma^2[\sigma_{32}^2]$ are low, classifying it as *Non-Split*. Otherwise, if large differences exist between the *variance* in the four sub-blocks or between their *means*, the classifier interprets the sub-CUs as having different textures and complexities and therefore suggests a CU must be partitioned to be encoded efficiently.

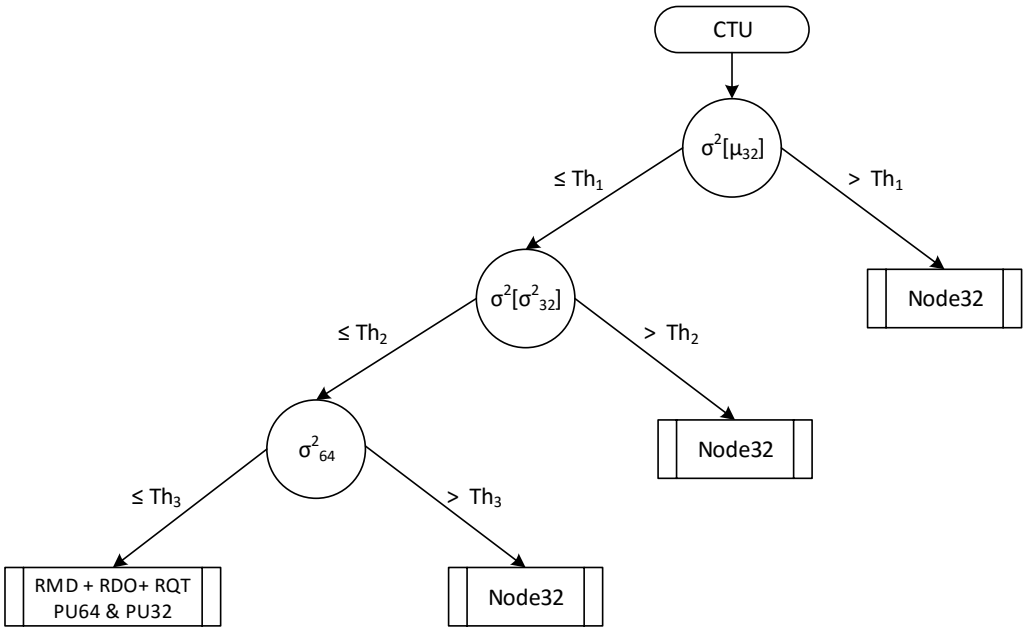


Figure 3.4. *Decision tree* for *Node64*

The *decision tree* for *Node64* was created using the WEKA *Machine Learning* tool, and is shown in Figure 3.9. Based on the different values of the *variance* and *mean* of the CU itself and the four 32x32 sub-CUs, the *decision tree* classifies each 64x64 CU into *Split* or *Non-Split* classes.

Node64 is defined with 3 inner nodes, three rules, and one condition for each rule with a specific threshold Th_i ($\forall i = 0,1,2.$) which determines the binary decision within the inner nodes. The *decision tree* methodology is described below:

- *Inner Node 1*. This inner node receives all 64x64 CTUs from the image slice, and computes the *mean* of the four 32x32 sub-CUs $\{\mu(CU_{1,k}) | \forall\, k = 0,..,3\}$ following Equation (3.6). Thereafter, using those *means* the $\sigma^2[\mu_{32}]$ is computed using the Equation (3.9), and the results are compared to Th_1. If the $\sigma^2[\mu_{32}] > Th_1$, the CTU is classified as Split and it is partitioned into four 32x32 Sub-CUs, which are sent to the *Node32* to be further processed. Otherwise, the CTU whose *means* of the four sub-CUs are quite similar (low *variance* of *means*) is sent to *Inner Node 2* for additional evaluation.

- *Inner Node 2*. This node works in a similar manner to the previous one, but the *variances* of the four 32x32 sub-CUs $\{\sigma^2(CU_{1,k}) | \forall\, k = 0,..,3\}$, are computed following Equation (3.5), instead of the *means*. Thereafter, using those *means* the $\sigma^2[\sigma_{32}^2]$ is computed using the Equation (3.8), and the result is compared to Th_2. Again, if the $\sigma^2[\sigma_{32}^2] > Th_2$ the CTU is classified as Split because the complexity of the sub-CUS is quite different (high *variance* of the sub-CUs *variance*), and it is partitioned into four 32x32 Sub-CUs which are sent to the *Node32* to be further processed. Otherwise, the CTU is sent to the last inner node for further evaluation.

- *Inner Node 3.* The last inner node just computes the *variance* of the CTU (σ_{64}^2,) following the Equation (3.7). Its value is compared to Th_3, and the CTU is classified as *Split* if the $\sigma_{64}^2 > Th_3$, meaning that it has a high texture complexity in overall, and it will be more efficiency encoded if it is divided into smaller sub-CUs. Otherwise, all CTU attributes are below the respective thresholds (Th_1, Th_2, Th_3), and therefore the CTU can be considered as likely being a high homogeneity region of the image. If that condition is met, the CTU is sent to the default HM *intra-prediction* stage for an exhaustive evaluation comprised of the RMD, RDO and RQT stages.

In this case the *intra-prediction* is evaluated but just for the 64x64 and 32x32 PU sizes, avoiding the evaluation of the other 336 PUs, sixteen 16x16, 64 16x16 PUs, and 256 4x4 PUs. The CTUs which reach this third inner node are the ones that achieve the highest complexity reduction, but it is also true that natural images usually have few homogeneous areas covering 64x64 pixels.

As mentioned above, in order to achieve the highest classification accuracy, the thresholds obtained with WEKA for each inner node were different for each QP. It is known that optimal partitioning in *intra-picture* prediction has a strong dependence on the encoding distortion level (QP), as it is described in [Ruiz14]. This QP dependency occurs not only in HEVC, but also in H.264/AVC. This observation is consistent with the Th_i thresholds obtained in the *decision tree* training for trained QPs, and summarized in Table 3.6, which shows that the thresholds increase with QP value (except Th_1), favouring the classification of the largest block size.

Table 3.6. Thresholds value for the *decision tree* of *Node64*

QP	Th_1	Th_2	Th_3
22	1	20	20
27	1	28	30
32	1	110	43
37	1	380	74

In order to analyse the partitioning performance of the fast *intra-prediction* proposal, the optimal CTU partitioning map obtained from the HM reference software as anchors, and the CTU partitioning decision using the Decision Tree of *Node64* are obtained. Both results are depicted in Figure 3.10 and Figure 3.11 for the first frame of the BQMall sequence and two QPs (QP22 and QP37).

As can be observed, the biggest CU sizes, namely 64x64 and 32x32, are mainly selected just in flat areas of the image, and it can also be seen that the classification decision in *Node64* matches with the high precision of the CTU partitioning carried out by the HM reference software, especially for QP22.

(a)

(b)

Figure 3.5. Performance partitioning of first frame of BQMall for QP22. (a) HM. (b) *Node64* proposal

(a)

(b)

Figure 3.6. Performance partitioning of first frame of BQMall sequence for QP37. (a) HM. (b) *Node64* proposal

To summarize, the *decision tree* for *Node64* was trained using the WEKA *Machine Learning* tool and three attributes, σ_{64}^2, $\sigma^2[\mu_{32}]$ and $\sigma^2[\sigma_{32}^2]$ from 64x64 CTU and four 32x32 sub-CUs, obtained from the first frame of the eight training sequences, which involves the use of a data set of 4,231. The unbalanced data set was undersampled using the *Random* method, and the new balanced class distribution was used for the *decision tree* training. Based on that data set, the WEKA tool was used to create four Decision Trees, one for each training QP, achieving a high classification accuracy of over 90% of CCI. The four trees were unified in a final *decision tree* with three inner nodes, a rule for each inner node, and a threshold for each rule and each QP. The visual results shown the partitioning performance of the proposed classifier for *Node64*, fits with the results obtained from the optimal partitioning using the HM reference software for *intra-prediction*, in terms of CTU classification sizes.

3.3.5.2 Decision Tree Specification for Node32

This section discusses the low complexity CTU partitioning decision proposal for the second decision node, *Node32* [Rui14a][Rui15b]. This node processes the split 32x32 CUs from *Node64*, and the decision of *Split* or *Non-Split* the CUs is taken, forwarding the CUs classified as *Split* to *Node16*, and otherwise sending the *Non-Split* CUs to the HM *intra-prediction* stage for an exhaustive evaluation of the 32x32 and 16x16 PU sizes. It should be noted that some of the 32x32 CUs reaching this node may have been wrongly classified from the previous node (*Node64*). For this reason this node has to be especially efficient in order to address this issue, because a new wrong classification of those CUs could lead to those CUs being classified as sixteen 16x16 CUs, or even smaller sizes, causing high quality degradation and a bit rate increase. On the other hand, correctly *Non-Split* classified CUs will obtain a high computational burden reduction, because they are not evaluated for the smaller 16x16 to 4x4 sizes.

The following sections describe the data set size, its class distribution and the set of attributes selected in the training process for *Node32*, as well as the *decision tree* specification in terms of inner nodes and rules.

3.3.5.2.1 Classifier Training

The 4,231 CTUs which comprise the initial data set, belonging to the 8 test sequences, are divided into 32x32 CUs for Node32 training, and therefore the total training data set size for this node is 16,924, as shown in Table 3.7. The attributes of each 32x32 CU are an instance for the classifier, which finds the correlation between the CU's features and its class.

Table 3.7. Data set size for *Node32*

Training Sequences class	Number of Sequences	Horizontal resolution	Vertical resolution	Data set size for *Node32*
Class A	2	2560	1600	8000
Class B	4	1920	1080	7680
Class F (BasketballDrillText)	1	832	480	364
Class F (SliceEditing)	1	1280	720	880
TOTAL				**16924**

As mentioned above, the class balance of the data set is a key factor for correct training and classification in ML, so the class distribution of the training set for *Node32* is also analysed for the four QPs recommended in the CTCs, and it is shown in Table 3.8. In this case, only the class distribution for the 22 and 27 QPs are imbalanced, because there are significantly more instances belonging to *Split*, by over 60%, than *Non-Split*, with around 30%. The class distribution for QP32 and QP37 is practically 50% balanced, thus an "*Unsupervised instance random subsample*" filter has been applied just to the instances of the QP22 and QP27 data sets. Finally, the *decision tree* is trained with the balanced class.

Table 3.8. Class distribution of training instances for the *Node32* decision tree

	Node32			
Class	QP22	QP27	QP32	QP37
Split	70.18%	62.28%	56.06%	45.76%
Non-Split	29.82%	37.72%	43.94%	54.24%

3.3.5.2.2 Decision Tree Specification

The training results obtained after the 10-fold cross-validation step for the Decision Trees of *Node32* are shown in Table 3.9, which includes the CCI, the trained tree sizes and the number of leaves of the decision trees. As can be noted, the CCIs are in the range of 83% to 89%, which is around 10% lower than the CCIs obtained in *Node64*, which can still be considered a high accuracy classification. As for the tree topology for the different QPs, it also presents a high dispersion with tree sizes in the range of 11 to 17 and the number of leaves from 6 to 9. Following the same methodology that was used for *Node64*, the four decision trees were unified into just one topology, but the threshold used in each inner node for each QP is kept according to the value obtained in the training.

Table 3.9. *Decision tree* accuracy and topology for the *Node32*

	Node32			
	QP22	QP27	QP32	QP37
CCI	89.10%	86.26%	85.02%	83.40%
Tree size	11	13	13	17
# Leaves	6	7	7	9

The attributes evaluation carried out in the training stage, selected as the most relevant attributes the *variance* of the 32x32 CUs, denoted as σ_{32}^2, and the *variance* of the *variances* in the four 16x16 Sub-CUs, denoted as $\sigma^2[\sigma_{16}^2]$ and computed as the $\sigma^2\left[\sigma^2\left(\mathrm{CU}_{2,4k+i}\right) \mid \forall\, i = 0,..,3\right]$. The classifier determines that a CU is homogeneous when σ_{32}^2, and $\sigma^2[\sigma_{16}^2]$ are low, classifying it as a *Non-Split* CU. Otherwise, if there exist large differences between the *variances* of the four sub-blocks or the global *variance* of the 32x32 CU is high, the classifier interprets the sub-CUs as having different textures and complexities and therefore suggests the CU partitioning in order to be encoded efficiently.

The *decision tree* for the *Node32* was created using the WEKA *Machine Learning* tool, and it is shown in Figure 3.12. Based on the *variance* of the 32x32 CUs and the *variance* of the *variances* of their four 16x16 sub-CUs, the *decision tree* classifies each of them in the *Split* or *Non-Split* class. *Node32* is defined with 2 inner nodes and one condition for each rule with a specific threshold $\mathrm{Th_i}$ ($\forall\, i = 4,5.$) which determines the binary decision within the inner nodes. The d*ecision tree* methodology is described below:

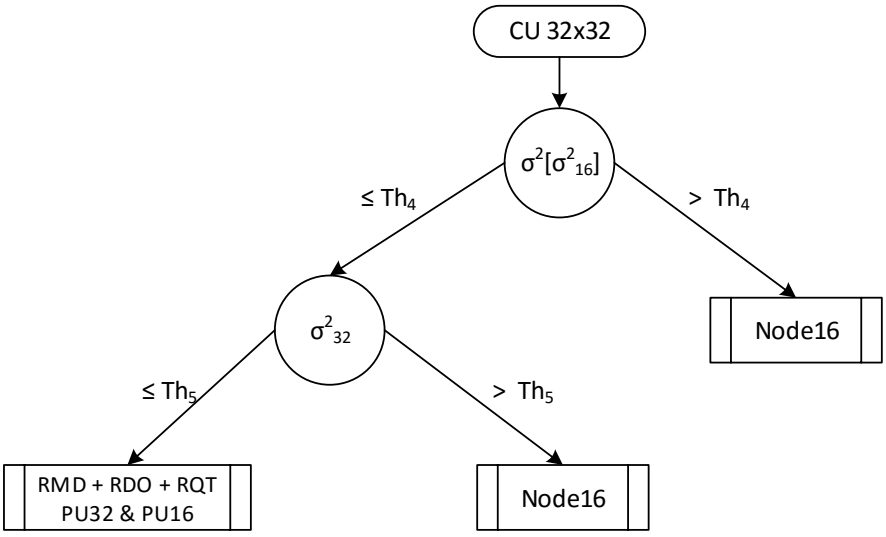


Figure 3.7. *Decision tree* for *Node32*

- *Inner Node 1*. The first inner node receives the $CU_{1,k}$ from the split 64x64 CUs of *Node64*, and computes the *variance* of the four 16x16 sub-CUs $\{\sigma^2(CU_{2,4k+i})| \forall\, i = 0,..,3\}$ following Equation (3.6). Thereafter, using those *variances* $\sigma^2[\sigma^2_{16}]$ is computed using the Equation (3.8), and the results are compared to Th_4. If the $\sigma^2[\sigma^2_{16}] > Th_4$, the $CU_{1,k}$ is classified as *Split* and it is partitioned into four 16x16 Sub-CUs which are sent to *Node16* to be further processed. Otherwise, the $CU_{1,k}$ for which the *variance* of the four sub-CUs is quite similar (low *variance* of the *variances*) is sent to the next inner node for additional evaluation.

- *Inner Node 2.* The last inner node just computes the *variance* of the $CU_{1,k}$ (σ^2_{32},) following the Equation (3.7). Its value is compared to Th_5, and the $CU_{1,k}$ is classified as *Split* if the $\sigma^2_{32} > Th_5$, meaning that it has a high texture complexity in overall, and it will be more efficiently encoded if it is divided into smaller sub-CUs. Otherwise, *means* CU attributes are all of them under the respective thresholds (Th_4, Th_5) and therefore the $CU_{1,k}$ can be likely considered as a high homogeneity region of the image. If that condition is met, the CU is conducted to the default HM *intra-prediction* stage for an exhaustive evaluation composed by the RMD, RDO and RQT stages. In this case the *intra-prediction* is evaluated but just for the 32x32 and 16x16 PU sizes, avoiding the evaluation of the other 64x64 PU size, and the 256 4x4 PU sizes. The CUs which remain at this node and therefore do not go through *Node16*, are considered as CU with medium complexity and they are more abundant in the natural images, obtaining a global high computational complexity reduction compared to the *Node64* implementation.

As was explained for the previous node, in order to achieve the highest classification accuracy, the thresholds obtained with WEKA for each inner node were different for each QP. The Th_i thresholds used for *Node32* are summarized in Table 3.10, which shows the same trend, their values increasing with increasing QP, favouring the classification of the largest block size for high QP values.

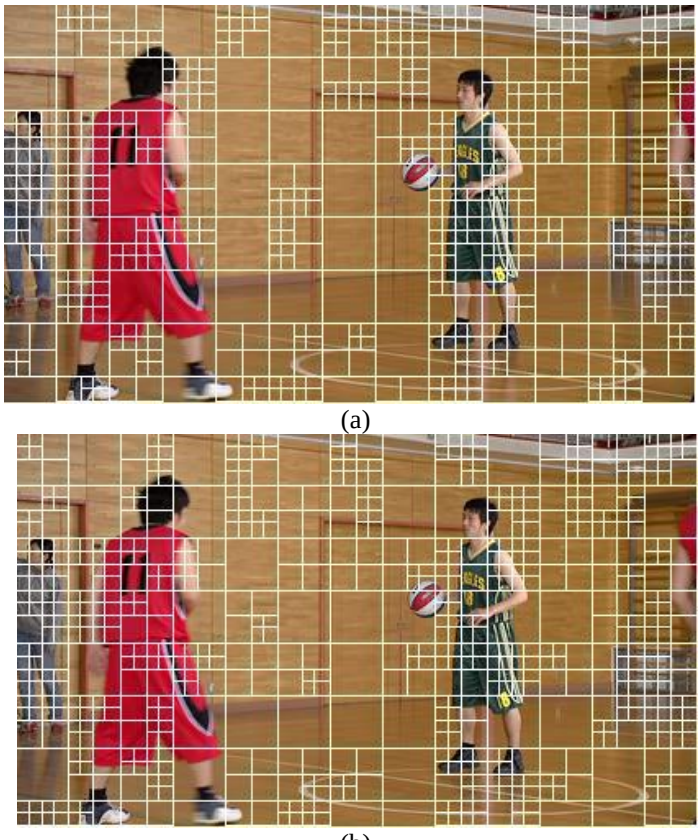
Table 3.10. Thresholds value for the *decision tree* of *Node32*

QP	Th_4	Th_5
22	768	26
27	1643	30
32	1767	37
37	7427	62

In order to analyse the partitioning performance of the fast *intra-prediction* proposal, the optimal CTU partitioning map was obtained from the HM reference software by using the

exhaustive evaluation, and it was also obtained using the *decision tree* of *Node64* and *Node32* jointly. Both results are depicted in Figure 3.13 and Figure 3.14 for the first frame of the *BasketballPass* sequence, using the lowest and highest training QPs (QP22 and QP37). As can be observed, for the QP22 encoding, the smallest CU sizes are adapted to the textured areas in both simulations, and just some of the biggest CU sizes of 32x32 are incorrectly split into smaller sub-CUs. It can be seen that for QP37 the HM reference software partitioning is mainly in larger CU sizes, and just a few CUTs are split into smaller sizes conforming to the content details.

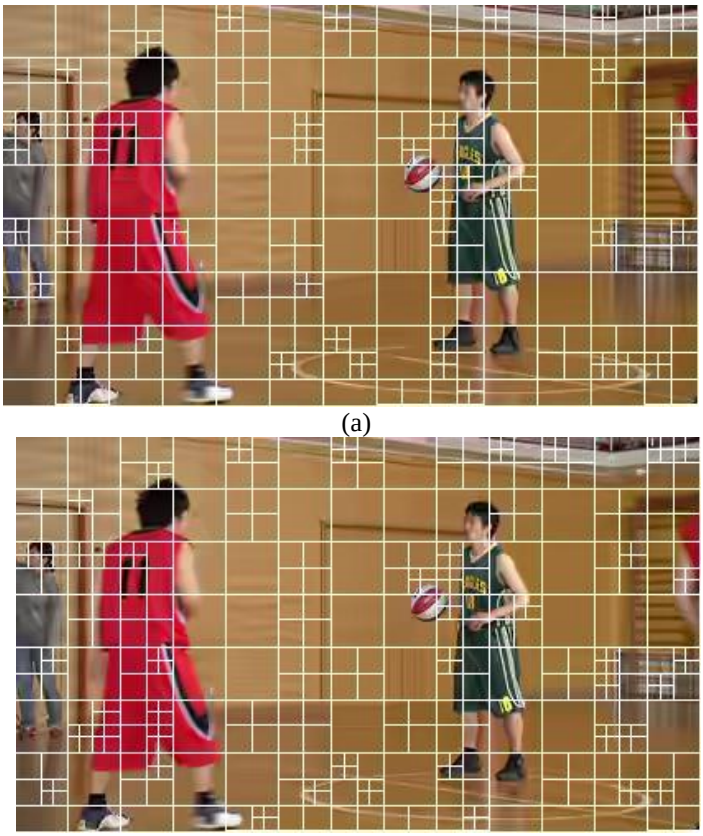
It is worth pointing out that the classification accuracy of the *Node64+Node32* approach for the highest QP (QP37) is higher than that obtained for QP22, matching mostly with the CTU partitioning carried out by the HM reference software. This happens for the largest CUs but also for the smallest one. That proves the accuracy of the CCI obtained in the training stage for *Node32,* but also the correct threshold selection that correctly covers a wide range of QPs. To summarize, the *decision tree* for *Node32* was trained using the WEKA *Machine Learning* tool and two attributes, σ_{32}^2, and $\sigma^2[\sigma_{16}^2]$, from the 32x32 CUs split from *Node64* ($CU_{1,k} \mid \forall\, k = 0,..,3$) and their respective four 16x16 sub-CUs ($CU_{2,4k+i} \mid \forall\, i = 0,..,3$). The data set was obtained from the first frame of the eight training sequences, which involves a data set size of 16,924 instances.



(a)

(b)

Figure 3.8. Performance partitioning of first frame of sequence BasketballPass for QP22. (a) HM. (b) *Node64+ Node32* proposal



(a) (b)

Figure 3.9. Performance partitioning of first frame of sequence BasketballPass for QP37. (a) HM. (b) *Node64+Node32* proposal

The imbalanced data set for QP22 and QP27 was undersampled using the *Random* method, and a new balanced class distribution was used for the d*ecision tree* training. Based on that data set, the WEKA tool was used to create four decision trees, one for each training QP, with classification accuracy by over 83% of CCI. The four trees were unified in a final *decision tree* with two inner nodes, a rule for each inner node, and a threshold for each rule and each QP. The visual results shown the partitioning performance of the proposed classifier for *Node64+Node32*, fitting with the results obtained from the optimal partitioning using the HM reference software for *intra-prediction*, in terms of CTU size classification, especially for the highest QPs.

3.3.5.3 Decision Tree Specification for Node16

This section discusses the low complexity CTU partitioning decision proposal [Rui15b] for the third decision node, *Node16*. This node processes the split 16x16 CUs from *Node32*, and the decision to *Split* or *Non-Split* the CUs is taken, forwarding the CUs classified as *Split* to the HM *intra-prediction* stage for an exhaustive evaluation of the 8x8 and 4x4 PU sizes, and otherwise sending the *Non-Split* to the HM *intra-prediction* stage but for the evaluation of the 16x16 and 8x8 sizes.

As was described for *Node32*, some of the 16x16 CUs reaching this node may have also been wrongly classified from the previous nodes (*Node64 and Node32*). This node is responsible for classifying the CUs which represent the most complex areas of the image with high detailed textures and strong edges, requiring more bits for encoding. Therefore, a wrong partitioning decision at this node will have a more severe impact in terms of quality degradation and bit rate increase than the previous nodes. At this node, the CUs correctly classified as *Non-Split* will avoid sixteen 4x4 PU size evaluations, and the CUs correctly classified as *Split* will just avoid one 16x16 evaluation. Consequently, the highest computational reduction will be obtained for the least complex CUs which do not require a 4x4 partitioning.

The following sections describe the data set size, its class distribution and the set of attributes selected in the training process for *Node16*, as well as the *decision tree* specification in term of inner nodes and rules.

3.3.5.3.1 Classifier Training

The 4,231 CTUs which comprise the data set, belonging to the 8 test sequences, are divided into 16x16 CUs for the *Node16* training, and therefore the total training data set size for this node is 67,696 instances, as is shown in Table 3.11. The attributes of each 16x16 CU are an instance for the classifier, which finds the correlation between the CU's features and its class.

Table 3.11. Data set size for *Node16*

Training Sequences class	Number of Sequences	Horizontal resolution	Vertical resolution	Data set size for *Node16*
Class A	2	2560	1600	32000
Class B	4	1920	1080	30720
Class F (BasketballDrillText)	1	832	480	1456
Class F (SliceEditing)	1	1280	720	3520
TOTAL				**67696**

The data set for the training of this node also suffers from a highly imbalanced class distribution, as can be observed in Table 3.12. For QP22 a 60% vs. 40% class distribution is obtained, but the *Non-Split* class distribution increases quickly with rising QP, reaching a 86.5% vs. 13.5% distribution for the highest QP (QP37). This trend is reported in [Rui14b], and it proves the attributes obtained from the input image are not sufficient to achieve a high accuracy partitioning classifier, because for the same CU the partitioning decision is strongly conditioned by the QP parameter.

Like the first node (*Node64*), the random subsampling filter was applied to the data set instances of the four training QPs, achieving a data set with a balanced class.

Table 3.12. Class distribution of training instances for the *Node16* decision tree

	Node16			
Class	QP22	QP27	QP32	QP37
Split	62.22%	30.40%	22.35%	13.51%
Non-Split	37.78%	69.60%	77.65%	86.49%

3.3.5.3.2 Decision Tree Specification

The training results obtained after the 10-fold cross-validation step for the Decision Trees of *Node16* are shown in Table 3.13, which includes the CCI, the trained tree sizes and the number of leaves of the decision trees. As can be noted, the CCIs are in the range of 72% to 79%, which can be considered as 20% lower than the accuracy obtained for *Node64*. As for the tree topology for the different QPs, it also presents a high dispersion with tree sizes in the range of 21 to 34, and the number of leaves is in the range of 7 to 17. Following the same methodology that was used for the previous nodes, the four decision trees were unified in just one topology, and the threshold used in each inner node for each QP is kept according to the value obtained in the training.

Table 3.13. *Decision tree* accuracy and topology for the *Node16*

	Node16			
	QP22	QP27	QP32	QP37
CCI	72.94%	74.72%	79.94%	77.37%
Tree size	34	33	21	33
# Leaves	7	17	11	17

The attributes evaluation carried out in the training stage selected as the most relevant attributes the *variance* of the 16x16 CU_s, denoted as σ_{16}^2, and the *variance* of the *variances* in the four 8x8 Sub-CUs, denoted as $\sigma^2[\sigma_8^2]$ and computed as the $\sigma^2\left[\sigma^2\left(\mathrm{CU}_{3,4k+i}\right) \mid \forall\, i = 0,..,3\right]$. The classifier determines that a CU has to be classified as *Non-Split* if the $\sigma^2[\sigma_8^2]$ is very low, regardless of the value of σ_{16}^2. Otherwise, the *Split* or *Non-Split* decision is strongly conditioned by both attributes $\sigma^2[\sigma_8^2]$ and σ_{16}^2, as described below.

The *decision tree* for *Node16* was created using the WEKA *Machine Learning* tool, and is shown in Figure 3.15. Based on the *variance* of the 16x16 CUs and the *variance* of the *variances* of their four 8x8 sub-CUs, the *decision tree* classifies each of them in the *Split* or

Non-Split class. The Split CUs are taken to the default HM *intra-prediction* stage for their evaluation for the 16x16 size and the next depth level (8x8). Otherwise, the *Split* class is selected, the CU is split and the sub-CUs are also sent to the HM *intra-prediction* stage for their evaluation for the 8x8 and 4x4 PU sizes. *Node16* is defined with 3 inner nodes and one condition for each rule with a specific threshold Th_i ($\forall i = 6,7,8.$) which determines the binary decision within the inner nodes. The *decision tree* methodology is described below:

- *Inner Node 1*. The first inner node receives the 16x16 CUs ($CU_{2,k}$) from the *Node32*, and computes the individual *variances* of the four 8x8 sub-CUs $\{\sigma^2(CU_{3,4k+i}) | \forall\, i = 0,..,3\}$ following Equation (3.5). Thereafter, using those *variances* the $\sigma^2[\sigma_8^2]$ is computed using the Equation (3.8), and the results are compared to Th_6. If the $\sigma^2[\sigma_8^2] > Th_6$ the $CU_{2,k}$ is sent to the inner Node 2 for further evaluation. Otherwise, the *variance* of the four sub-CUs is quite similar, the $CU_{2,k}$ is classified as *Non-Split* and it is sent to the default HM *intra-prediction* stage for an exhaustive evaluation for the 16x16 and 8x8 PU sizes.

- *Inner Node 2*. This node receives the $CU_{2,k}$ that are not classified as *Non-Split*, and their *variance* (σ_{16}^2,) is computed following the Equation (3.7). Its value is compared to Th_7, and each $CU_{2,k}$ is classified as *Split* if the $\sigma_{16}^2 > Th_7$, meaning that it has a high texture complexity overall, and it will be more efficiently encoded if it is divided into smaller sub-CUs.

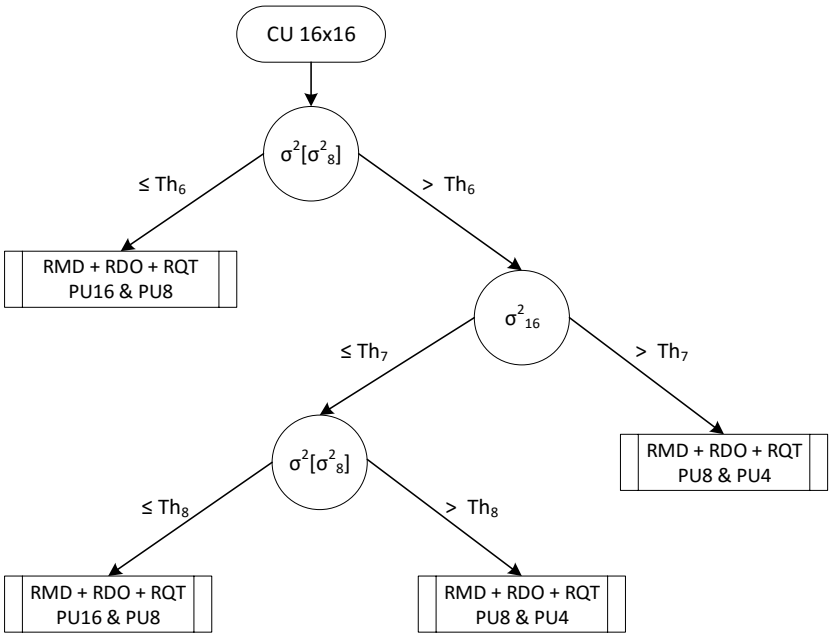


Figure 3.10. *Decision tree* for *Node16*

Because no more nodes are defined, the $CU_{2,k}$ is sent to the default HM *intra-prediction* stage for an exhaustive evaluation for the 8x8 and 4x4 PU sizes, avoiding the 16x16 evaluation. Otherwise, the CU is driven to the last inner node for additional evaluation.

– *Inner Node 3.* The last inner node analyses $\sigma^2[\sigma_8^2]$ again, but now its value is compared to a new threshold Th_8, and the $CU_{2,k}$ is classified as *Split* if $\sigma^2[\sigma_8^2] > Th_8$, meaning that although the σ_{16}^2 has a low value and theoretically the overall texture is soft, but the dispersion in the *variance* values of the sub-CUs determinates the best efficiency encoding performance can be achieved by dividing the CU into smaller sub-CUs. In this case, the $CU_{2,k}$ is sent to the default HM *intra-prediction* stage for an exhaustive evaluation for the 8x8 and 4x4 PU sizes. Otherwise, the CU is classified as *Non-Split*, considered as being of medium homogeneity, and the CU is sent to the default HM *intra-prediction* stage for an exhaustive evaluation just for the 16x16 and 8x8 PU sizes, avoiding the evaluation of the 16 4x4 PUs.

The Th_i thresholds obtained with WEKA for each inner node and QP were used for *Node16, and they* are included in Table 3.14, which shows the same trend, their values increasing with increasing QP, favouring the classification of the largest block size for high QP values.

Table 3.14. Thresholds values for the *decision tree* of *Node16*

QP	Th_6	Th_7	Th_8
22	1255	831	8301
27	3333	2001	50000
32	10699	2321	108930
37	28326	3163	5671926

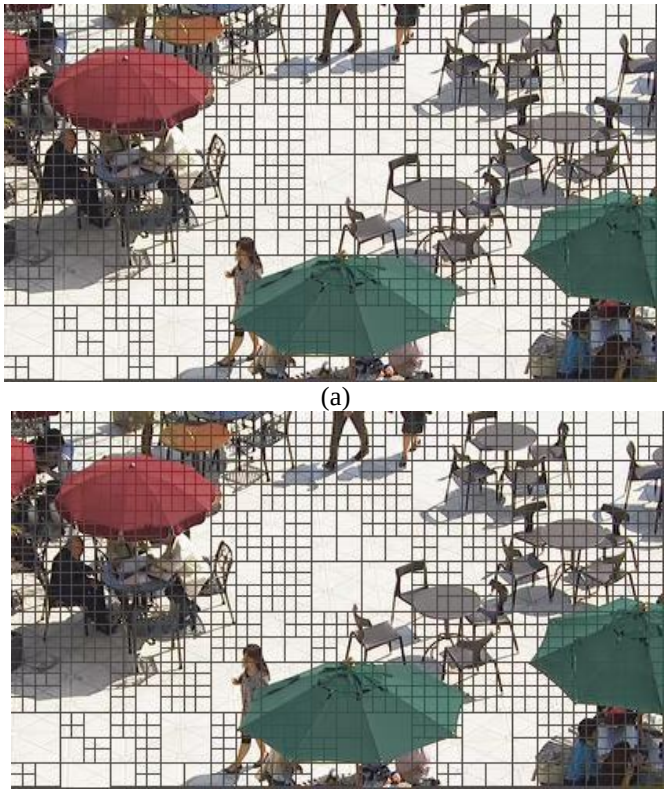
In order to analyse the partitioning performance of the overall fast *intra-prediction* proposal with three nodes, the optimal CTU partitioning map was obtained from the HM reference software by using the exhaustive evaluation of the RMD, RDO and RQT, and it was also obtained using the d*ecision trees* of *Node64*, *Node32* and *Node16* jointly.

Both results are depicted in Figure 3.16 to Figure 3.19 for the first frame of the *BQSquare* sequence, using the four training QPs (QP22, QP27, QP32 and QP37). As can be observed, for QP22 both partitioning maps match with high precision for small CU sizes and also for the biggest one. For QP27 several differences appear in the centre of the image, with small partitions being selected for the HM reference software and large CUs for the proposal, because those CUs are quite homogeneous.

The results for QP32 and QP37 show a high accuracy partitioning selection between the HM and the overall proposal implementation, and just small drifts can be found in the decision of the smaller CUs in the top and bottom areas of the image.

The reported example shows that the overall implementation of the *decision trees* in *Node64*, *Node32*, and *Node16*, achieves a high precision matching with the optimal decision from the HM reference software. That classification accuracy is uniform for the whole range of QPs, proving correct thresholds selection is obtained for WEKA in the training stage.

To summarize, the *decision tree* for *Node16* was trained using the WEKA *ML* tool and two attributes, σ_{16}^2, and $\sigma^2[\sigma_8^2]$, from the 16x16 CUs split from *Node32* ($CU_{2,k}$ $|\forall\, k = 0,..,3$) and their respective four 8x8 sub-CUs ($CU_{3,4k+i}$ $|\forall\, i = 0,..,3$). The data set was obtained from the first frame of the eight training sequences, which involves a data set size of 16,924 instances. The imbalanced data sets were undersampled using the *Random* method, and the new balanced class distribution was used for the *decision tree* training. Based on that data set, the WEKA tool was used to create four *decision trees*, one for each training QP, with classification accuracy by over 72% of CCI.

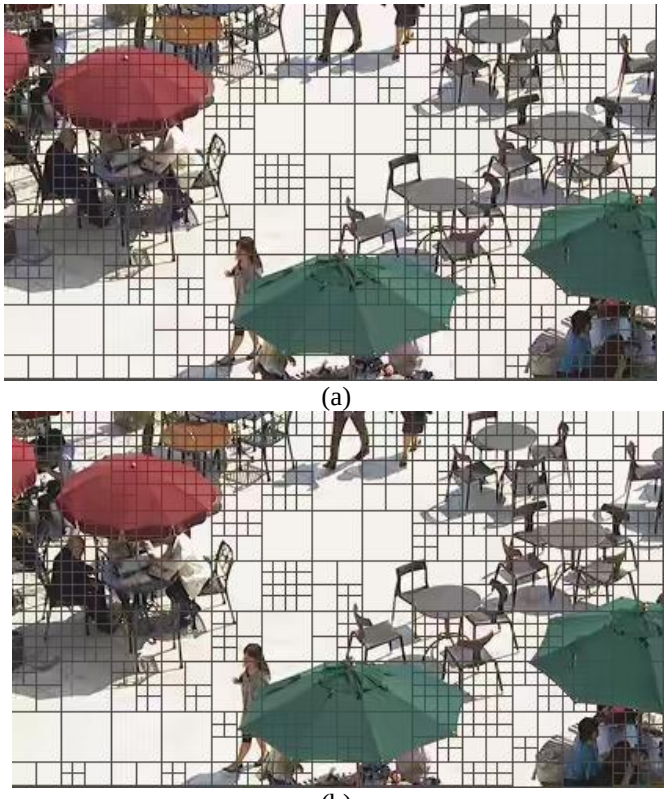


(a) (b)

Figure 3.11. Performance partitioning of first frame of sequence BQSquare for QP22. (a) HM. (b) *Node64+Node32+Node16* proposal

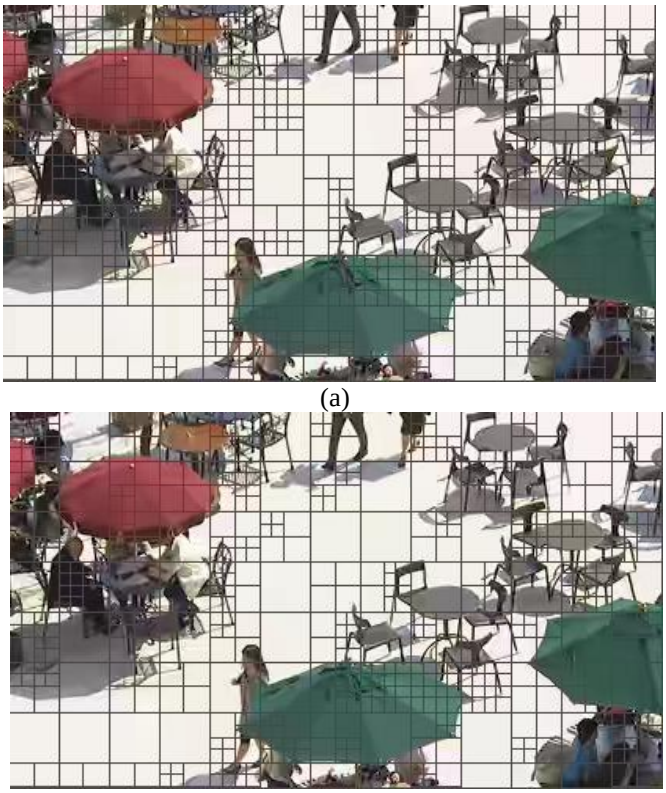
(a) (b)

Figure 3.12. Performance partitioning of first frame of sequence *BQSquare for QP27*. (a) HM. (b) *Node64+Node32+Node16* proposal



(a) (b)

Figure 3.13. Performance partitioning of first frame of sequence BQSquare for QP32. (a) HM. (b) *Node64+Node32+Node16* proposal



(a)

(b)

Figure 3.14. Performance partitioning of first frame of sequence BQSquare for QP37. (a) HM. (b) *Node64+Node32+Node16* proposal

The four trees were unified in a final *decision tree* with three inner nodes, a rule for each inner node, and a threshold for each rule and each QP. The visual results show the partitioning performance of the proposed classifier for *Node64+Node32+Node16* jointly, fitting with the results obtained from the optimal partitioning using the HM reference software for intra-prediction, in terms of CTU size classification.

3.3.6 Classifier Implementation in HM Reference Software

Once the three decision trees were trained, the model was implemented in the HM reference software, replacing the full size searching scheme used by the *intra-prediction* through the RMD and RDO algorithms. Undoubtedly, the key factor to achieving a fast partitioning algorithm is an efficient classifier, but also an efficient implementation of the CU features computation. . The most noteworthy actions are described below:

1. The *decision tree* rules are converted to "*if-else*" statements, allowing a fast CU partitioning classification that avoids the evaluation of the RMD and RDO for the full range of CU sizes. The next figure shows the pseudocode of the *decision tree* implementation for *Node64* and QP32, where we can see the three rules implemented and their respective thresholds: Th_1, Th_2 and Th_3.

```
VarMean32 = Var[Mean32(0,0),Mean32(0,1),Mean32(1,0),Mean32(1,0)]
if (VarMean32 > TH1)
        return Node32;
else
        VarVar32 = Var[Var32(0,0),Var32(0,1),Var32(1,0),Var32(1,0)]
        if (VarVar32 > TH2)
                return Node32;
        else
                Var64 = Var[CU32(0,0),CU32(0,1),CU32(1,0),CU32(1,0)]
                if (Var64 > TH3)
                        return Node32;
                else
                        return RMD_RDO_RQT_64_32;
```

Figure 3.15. Example of *decision tree* implementation of *Node64* for QP32.

2. The attributes are computed exclusively when they are required for the inner node of the decision tree, instead of being computed all together at the beginning of the classifier stage.

3. Attribute computations are reused from *Node2N* to *NodeN*, if they were computed previously, thereby avoiding further computational costs. For instance, the *variance* of a 32x32 CU at *Node32* can be reused to compute the attributes at *Node64*, and also the *variances* of the 16×16 CUs are reused to compute the attributes of *Node32*.

4. A polynomial fitting is computed to find the thresholds $Th_i(QP)$ for the full range of QPs (52 values), using the "*polyfit*" function of MATLAB. That function finds the coefficients of a polynomial $P(x)$ of degree "*n*" which fits the data $P(x_i)$ by minimizing the sum of the deviations of the data from the model (*least-squares fit*). Table 3.15 depicts the polynomials obtained for the calculation of the thresholds of the look-up table.

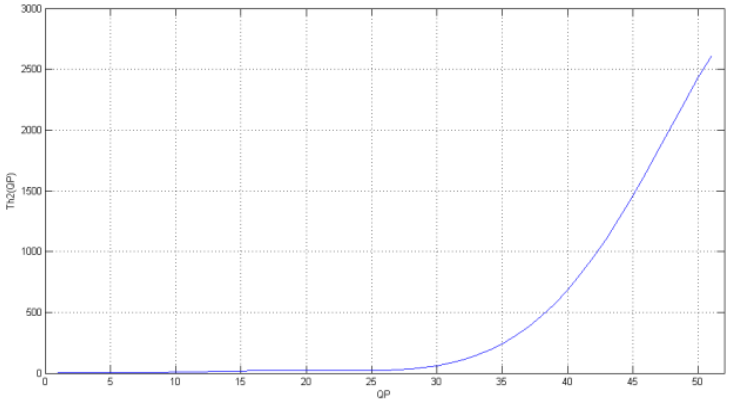
5. The thresholds are stored in a look-up table of 52 positions for fast access, avoiding threshold computation on the fly.

The polynomial functions for the Th_2 and Th_8 thresholds are given in Figure 3.21, showing similar characterization with a flat slope for lower QPs and an exponential growth starting in the range of QP15 to QP40.

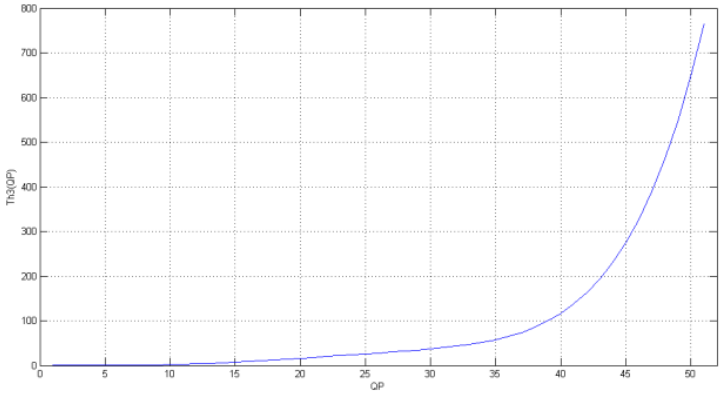
Table 3.15. Polynomials for the thresholds Th_2 to Th_8

Threshold	Polynomial
Th_2	$-5.23\ 10^{-08}\ QP^7 + 5.93\ 10^{-06}\ QP^6 - 0.0002\ QP^5 + 0.0022\ QP^4$
Th_3	$5.79\ 10^{-10}\ QP^7 + 2.22\ 10^{-07}\ QP^6 - 0.0000177\ QP^5 + 0.00036\ QP^4$
Th_4	$0.0002\ QP^6 - 0.0184\ QP^5 + 0.5126\ QP^4 - 4.5984\ QP^3$
Th_5	$-4.36\ 10^{-08}\ QP^6 + 1.39\ 10^{-05}\ QP^5 - 0.0008\ QP^4 + 0.0135\ QP^3$
Th_6	$0.0006\ QP^5 + 0.0273\ QP^4 - 2.0704\ QP^3 + 28.2882\ QP^2$
Th_7	$0.0001 QP^6 - 0.009\ 4\ QP^5 + 0.2797\ QP^4 - 2.6215\ QP^3$
Th_8	$0.1\ QP^6 - 11.8\ QP^5 + 315.1\ QP^4 - 2777.2\ QP^3$

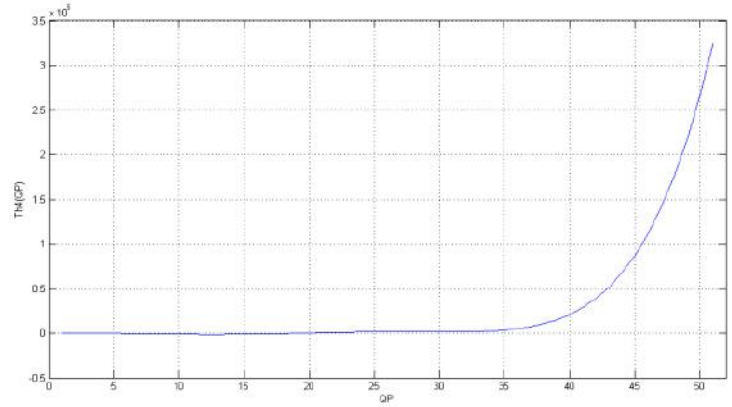
The overall algorithm implementation is depicted in Figure 3.22, and the following generic process is applied. Each $CU_{d,k}$ is evaluated at *Node2N* ($\forall\ N = 32, 12, 8$) and if the *Non-Split* decision is taken (left branch), the original *intra-prediction* process implemented in the HM is carried out for all the prediction modes, but exclusively for the *d* and *d+1* depths, skipping the evaluation of the smaller CU sizes. Otherwise, the splitting CU decision is taken (right branch), the $CU_{d,k}$ is split into four sub-CUs, $CU_{d+1,4k+i}$, and each of them goes to the lower *NodeN* to be further processed by the next *decision tree* by applying the same method.



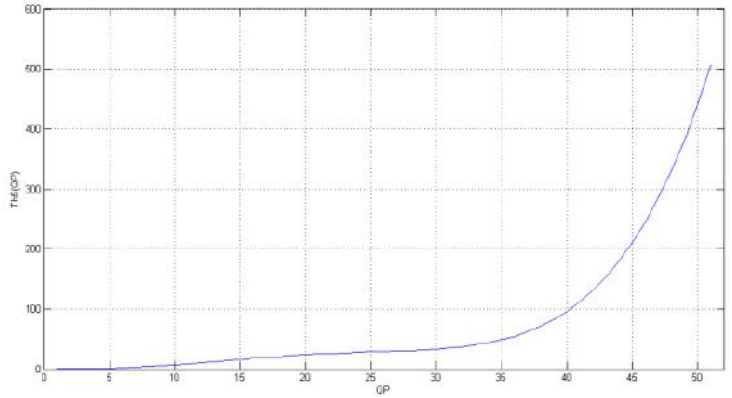
(a)



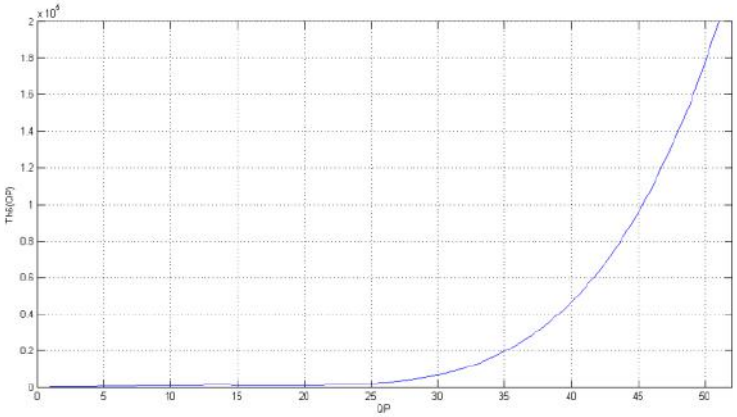
(b)



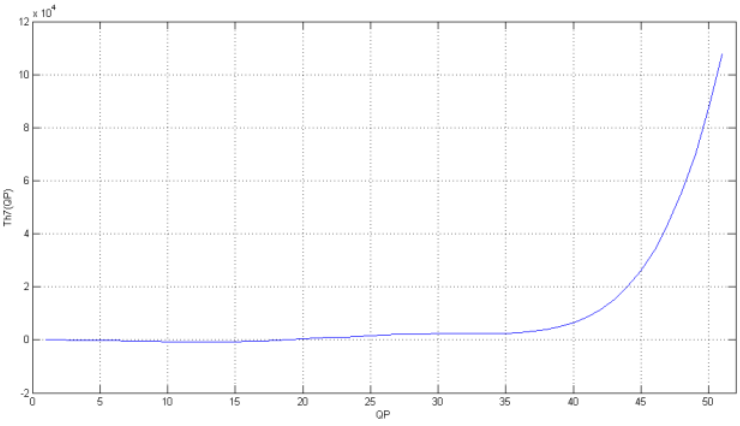
(c)



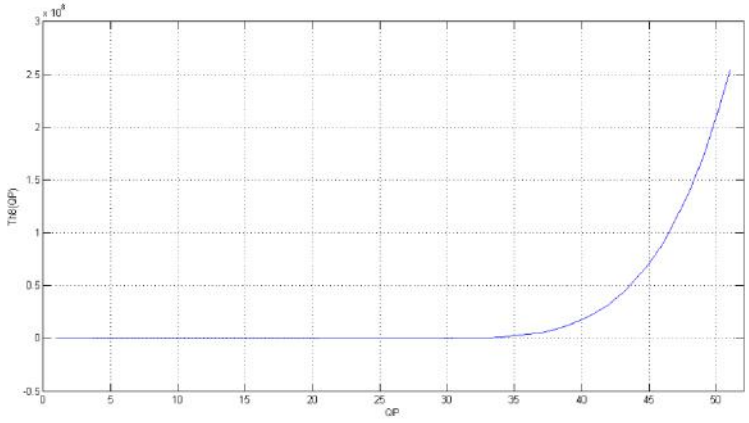
(d)



(e)



(f)



(g)

Figure 3.16. Polynomial function for thresholds Th_2 (a) to Th_8 (g).

It should be noted that in the left branches of each node, a double CU sizes evaluation, 2N×2N and N×N, is carried out with the aim of avoiding the CU misclassification in cases where the CU attributes do not permit an accurate characterization.

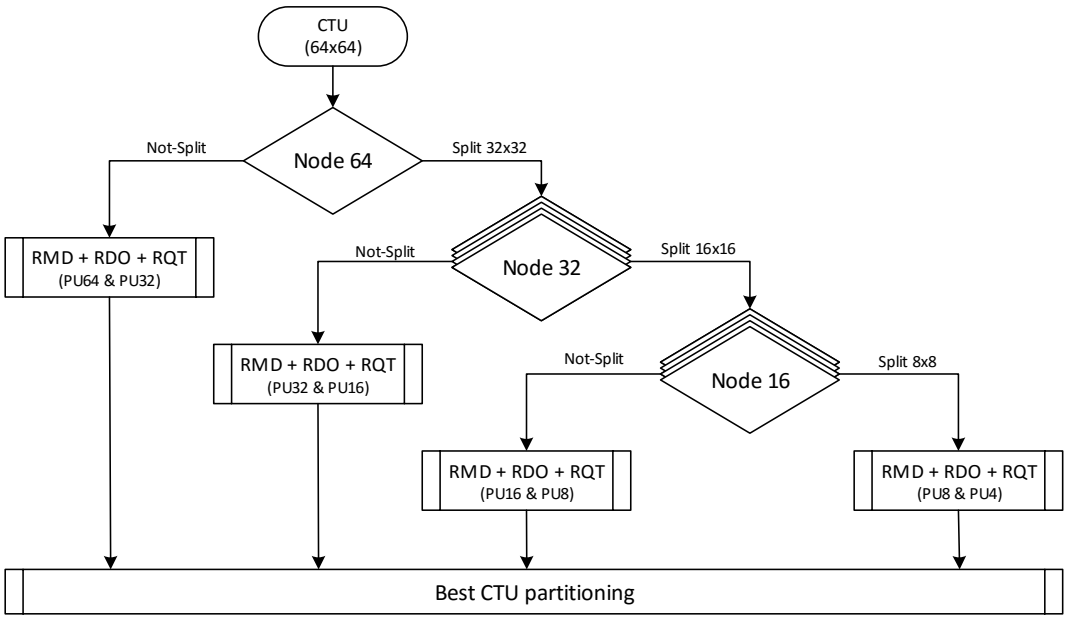


Figure 3.17. Fast-partitioning algorithm architecture using proposed decision trees

3.4 PERFORMANCE EVALUATION

In order to evaluate the CU partitioning performance of the proposed algorithm, the *Node64*, *Node32* and *Node16* decision tree nodes were implemented in the latest version of the HEVC reference software, HM 16.6 [JCT-VC Reference Software]. The non-modified HM 16.6

was used as anchor by using the same sequences and encoding parameters. The simulations were run over on PC with an Intel Core i7-4770 3.40 GHz processor, 16GB of RAM memory and a 64-bit Operative System.

3.4.1 Encoding Parameters

The experiments were conducted under the "*Common Test Conditions and Software Reference Configurations*" (CTC) recommended by the JCT-VC [JCTVC-L1100] for the "*All-Intra*" mode configuration, and the Main profile (AI-Main). That recommendation specifies the use of four QPs (QP22, QP27, QP32 and QP37) and a set of 22 test sequences classified in five classes, A to F, which cover a wide range of resolutions from the largest with *Ultra High Definition* (Ultra-HD) resolution, to the smaller with 416x240 pixels, and frame rates from 60 fps to 24 fps. All the sequences use 4:2:0 chroma subsampling and a bit-depth of 8 bits.

Table 3.16 summarizes the test sequence features in terms of resolution, frame rate and number of frames, and Figure 3.23 depicts the first frames of the 12 test sequences which were not used in the training stage.

Table 3.16. Test sequences parameters defined in the JCT-VC *Common Test Conditions*

Class	Sequence	Number Frames	Frame Rate	Horizontal Resolution	Vertical Resolution
Class A	Traffic	150	30	2560	1600
	PeopleOnStreet	150	30		
Class B	BasketballDrive	500	50	1920	1080
	BQTerrace	600	60		
	Cactus	500	50		
	Kimono	240	24		
	ParkScene	240	24		
Class C	BasketballDrill	500	50	832	480
	BQMall	600	60		
	PartyScene	500	50		
	RaceHorses	300	30		
Class D	BasketballPass	500	50	416	240
	BQSquare	600	60		
	BlowingBubbles	500	50		
	RaceHorses	300	30		
Class E	FourPeople	600	60	1280	720
	Johnny	600	60		
	KristenAndSara	600	60		
Class F	BasketballDrillText	500	50	832	480
	ChinaSpeed	500	30	1024	768
	SlideEditing	300	30	1280	720
	SlideShow	500	20	1280	720

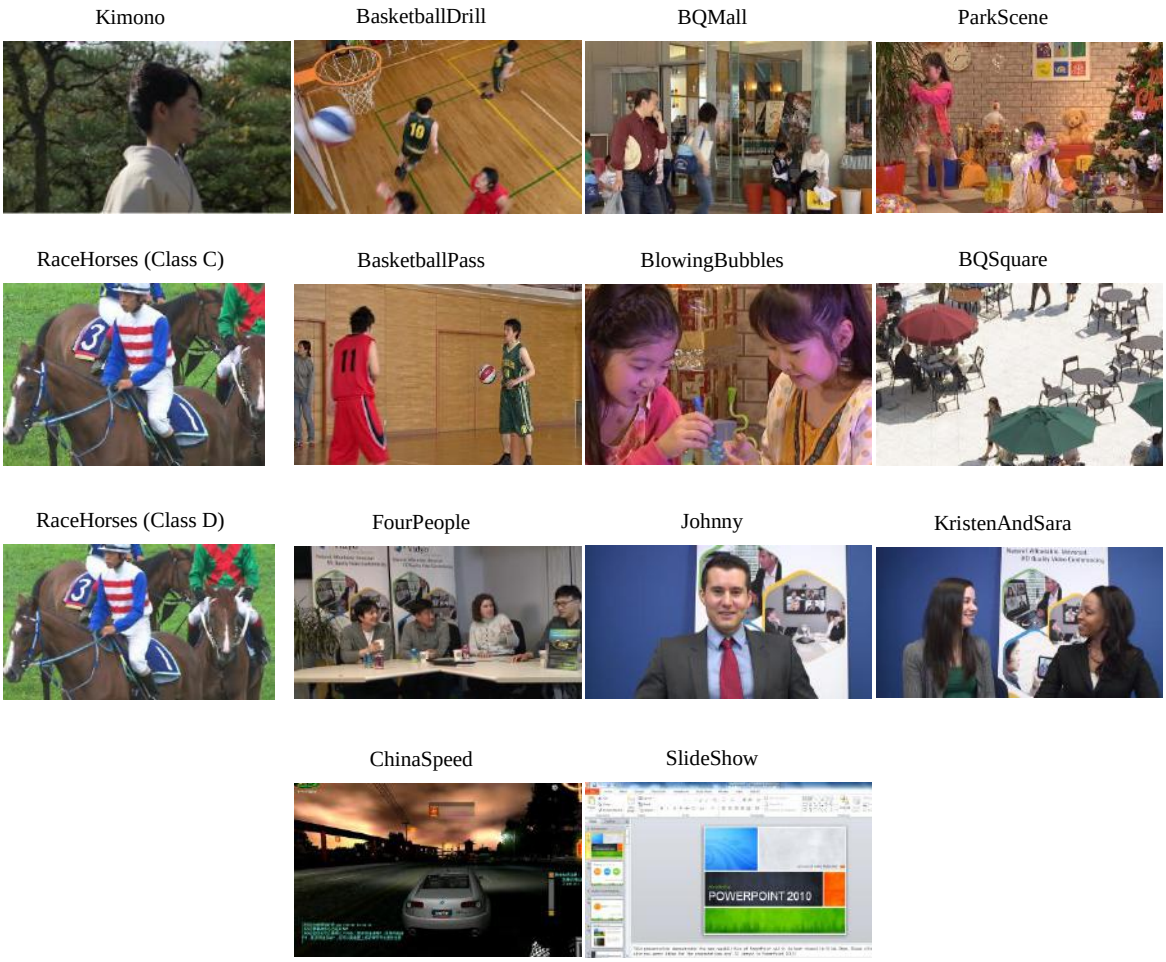


Figure 3.18. First frame of the non-training test sequences

3.4.2 Metrics

The algorithm's performance was evaluated in terms of Computational Complexity Reduction (CCR) and *Rate-Distortion* (RD) performance, and both of them were compared to the HM results. For the CCR measure, the *Time Saving* metric (*T.Saving*) was computed following the Equation (3.10):

$$T.Saving(\%) = \frac{\text{Enc. Time (HM16.6)} - \text{Enc. Time(Prop)}}{\text{Enc. Time (HM16.6)}} \cdot 100 \quad (3.10)$$

Regarding the RD performance, the average *Peak Signal to Noise Ratio* (PSNR) metric was calculated for each luma (*Y_PSNR*) and chroma components (U_*PSNR, V_PSNR*) and for the full number of frames of each sequence. In order to obtain a global quality measure, the average PSNR of the three components, denoted as *YUV_PSNR*, was also computed.

As mentioned above, the chroma subsampling format of the test sequences was 4:2:0, so the well-known *YUV_PSNR* according to Equation (3.11) was used, which applies weight pondering to the PSNR obtained for each component. Those weights are recommended by the JCT-VC in [JCTVC H0012] and they are considered a fair representation of the visual quality, which is more sensitive to the luminance stimulus than the chrominance.

$$YUV\_PSNR(dB) = \frac{6 \cdot Y\_PSNR + U\_PSNR + V\_PSNR}{8} \qquad (3.11)$$

The *YUV_PSNRs* for the four QPs were used for the computation of the RD performance by using the *Bjøntegaard Delta rate metric* (*BD-rate*) defined by ITU [VCEG-M33] and recommended in the CTC. The *BD-rate* provides the average difference between the RD curves measured as a percentage of bit rate that is necessary to increase or decrease to achieve the same PSNR quality in both curves. In our simulation, a positive *BD-rate means* the encoded bit rate using the fast partitioning algorithm is higher than the bit rate obtained with the HM, thus that is denoted as the proposal penalty in terms of bit rate.

3.4.3 Simulation Results

In order to obtain the *decision trees'* performance in terms of CCR and average rate penalty in an independent and concise way, two sets of simulations were performed, first for the training set, and also for the non-training set (testing sequences). The first was carried out using the eight test sequences selected for classifier training, while the second simulation used the remaining 14 test sequences from the JCT-VC data set. For both simulations, the results were reported by applying the algorithm in a scalable way, starting with *Node64*, afterward the *Node64* and *Node32* combined, denoted as *Node64+Node32*, and finally the implementation of three nodes, denoted as *Node64+Node32+Node16*.

The results in terms of the quality difference between the HEVC reference software and the proposed algorithm are shown by using the Rate-Distortion curves. Finally, and with the aim of validating the proposed threshold computation method using the polynomial fitting, the results for the non-training QPs are reported.

3.4.3.1 Simulation Results for Training Sequences

Table 3.17 shows the experimental results of the proposed algorithm using the training sequences compared with the HM 16.6 reference software. It can be observed that for the *Node64* implementation, all the sequences obtain a negligible 0.05% penalty in terms of *BD-*

rate, with a moderate average *Time Saving* of around 13%. The results for the implementation of the combined *Node64+Node32* decision trees report much better *Time Savings* of over 30% compared with the reference model, and a bit rate increase lower than 1% in terms of *BD-rate*, which is a very good balance between speed-up and rate penalty. Finally, the overall algorithm (*Node64+Node32+Node16*) shows a huge computational complexity reduction of over 50%, while increasing the bit rate penalty by around 2.1%.

It should be noted that for the overall implementation, the best speed-up performance of nearly 60% is obtained by the *BasketballDrive* sequence, and the sequence *PeopleOnStreet* obtains the best *BD-rate* with a negligible penalty of 1%. The worst performance is obtained by the *SlideEditing* sequence, with a lower computational burden reduction and the highest rate increase. However, it should be noted that the *SlideEditing* sequence is a synthetic sequence captured from the PC screen, which comprises complex and non-natural patterns, which has proved to be difficult to predict by the model.

Table 3.17. Performance comparison for the proposed algorithm with HM16.6, for the training sequences

Classification	Sequence	Frames	Node 64		Node 64 + Node 32		Node 64 + Node 32 + Node 16	
			T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)
Class A	Traffic	150	11.60	0.0	29.39	0.9	56.61	1.6
	PeopleOnStreet	150	12.19	0.0	28.03	0.6	50.09	1.1
Class B	BasketballDrive	500	14.36	0.2	36.87	2.4	59.19	3.1
	BQTerrace	600	14.15	0.1	30.96	0.8	52.57	1.3
	Cactus	500	13.18	0.1	29.77	1.0	56.95	2.1
	ParkScene	240	14.51	0.0	30.74	0.9	58.75	1.6
Class F	BasketballDrillText	500	11.80	0.0	28.17	0.3	51.97	2.2
	SlideEditing	300	11.97	0.0	27.33	0.4	39.24	4.3
TOTAL			**12.97**	**0.05**	**30.16**	**0.91**	**53.17**	**2.16**

3.4.3.2 Simulation Results for Non-Training Sequences

The results for the 14 non-training sequences data set are depicted in Table 3.18. These testing sequences cover a wide resolution and complexity spectrum, including natural and non-natural images, such as the *ChinaSpeed* and *SlidShow* computer screen sequences.

The results for *Node64* are slightly better, 0.5%, than those obtained previously for the training sequences in terms of speed increases. However, it is relevant to highlight that two non-training sequences, "*KristenAndSara*" and "*SlideShow*", achieve 26.8% and 35% *Time Savings*, respectively, double the performance of any of the training sequences. This finding proves the accurate classifier performance of sequences not used for the training decision tree.

Regarding the *Node64+Node32* implementation, the performance is also very similar to that obtained previously for the training sequences, with an average complexity reduction of 30% and a rate penalty of around 1.1%. The class D sequences have a lower complexity reduction of around 20%, as well as a lower bit rate penalty of 0.3%. One sequence, "*SlideShow*", surpasses by over 20% the performance of the best training sequence, "*BasketballDrive*", in terms of speed increase, with half the bit rate penalty.

Finally, the overall implementation (*Node64+Node32+Node16*) achieves practically the same 52% complexity reduction as that obtained by using the training sequences, with a *BD-rate* slightly higher by 0.5% compared to the training sequences. The best complexity reduction performance was achieved for class B sequences with a 63% complexity reduction, and the worst complexity reduction was obtained for the *BQSquare* sequence, but it achieved the best rate performance with a negligible 1.1% of *BD-rate*.

Table 3.18. Performance comparison for the proposed algorithm with HM16.6, for the non-training sequences

Classification	Sequence	Frames	Node 64		Node 64 + Node 32		Node 64 + Node 32 + Node 16	
			T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)
Class B	Kimono	240	15.43	0.1	33.90	5.1	62.05	5.3
Class C	BasketballDrill	500	15.4	0.0	27.63	0.5	53.50	2.3
	BQMall	600	10.27	0.0	26.88	0.8	50.76	2.5
	PartyScene	500	10.89	0.0	22.08	0.1	45.48	2.1
	RaceHorses	300	10.18	0.0	26.41	0.7	53.14	1.7
Class D	BasketballPass	500	10.25	0.0	23.54	0.6	45.66	2.1
	BQSquare	600	5.98	0.0	21.11	0.3	35.61	1.1
	BlowingBubbles	500	8.82	0.0	18.58	0.0	40.55	1.7
	RaceHorses	300	6.80	0.0	19.75	0.4	43.20	1.8
Class E	FourPeople	600	8.13	0.0	33.88	0.8	56.64	2.0
	Johnny	600	12.48	0.3	45.54	3.4	63.00	4.4
	KristenAndSara	600	26.81	0.3	44.39	1.7	61.19	2.6
Class F	ChinaSpeed	500	13.63	0.0	33.81	0.5	53.54	4.5
	SlideShow	500	35.28	0.2	55.30	1.2	63.51	3.3
Class B			15.43	0.10	33.90	5.10	62.05	5.30
Class C			10.40	0.00	25.75	0.53	50.72	2.15
Class D			7.43	0.00	20.74	0.33	41.25	1.68
Class E			18.22	0.20	41.27	1.97	60.28	3.00
Class F			24.46	0.10	44.56	0.85	58.53	3.90
TOTAL			**13.59**	**0.06**	**30.91**	**1.15**	**51.99**	**2.67**

It should be noted that only eight frames were used for the training of the decision tree, which is 0.081% of the total frames that comprise 22 simulated sequences (9,780 frames). The global experimental results confirm the proposed algorithm can reduce the computational complexity of HEVC intra-picture prediction by over 52% with a slight bit rate increase, favoring real-time software and hardware implementation.

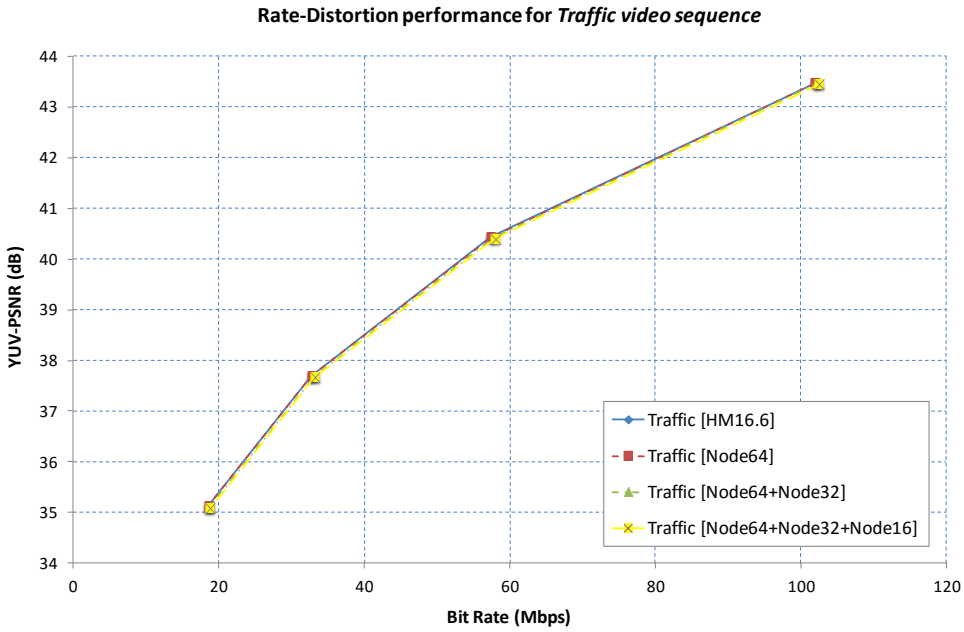
3.4.3.3 Rate-Distortion Performance

With the aim of showing the quality performance of the proposed algorithm, the Rate-Distortion results are shown in Fig 24 to 29, individualized for each sequence class and depicting the best sequence performance and the worst of each of them. These figures depict the YUV_PSNR for the four QPs of the non-modified HM implementation, and also of the three proposed fast tree decision implementations N*ode64*, *Node64+Node32*, and *Node64+Node32+Node16*.

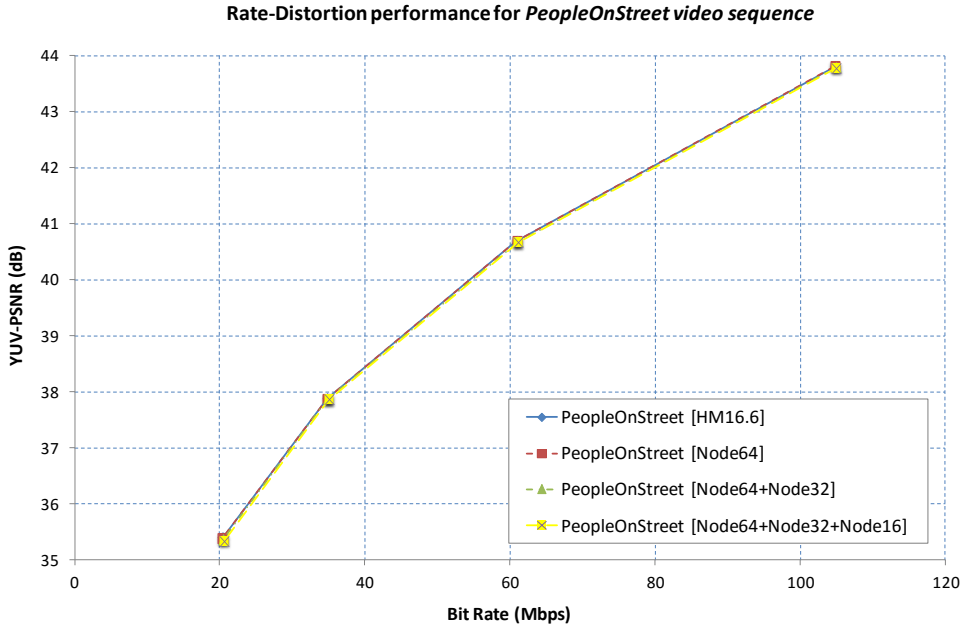
For class A sequences, Figure 3.24(a), it can be observed that the three node implementations achieve practically the same RD results as the HM implementation, with the four curves overlapping even for the worst sequence, *PeopleOnStreet* in Figure 3.24(b). In Figure 3.25(b), the worst performance sequence for class B, *Kimono*, obtains slightly lower quality for the two lower QPs, and the same quality for the other two QPs, but in Figure 3.25(a) the proposed algorithm's performance for the *BQTerrace* sequence matches the HM results.

The results for class C sequences, Figure 3.26, also show a very similar performance, and only the worst sequence, *BQMall*, obtains small differences for the overall implementation, namely *Node64+Node32+Node16*. Similar conclusions can be drawn from the class D and E sequences. The largest performance differences are shown for the class F sequences. In this case, the best performance sequence, *BasketballDrillText*, reports a perceptibly worse performance compared to the HM implementation for the overall implementation.

As a final conclusion, no differences with the original HM implementation can be found for *Node64* and *Node64+Node32* in terms of RD performance, for the whole set of sequences. However, the RD performance results for *Node64+Node32+Node16* are slightly lower than the anchor, except for the class A sequences, whose performance is practically the same as that of HM.

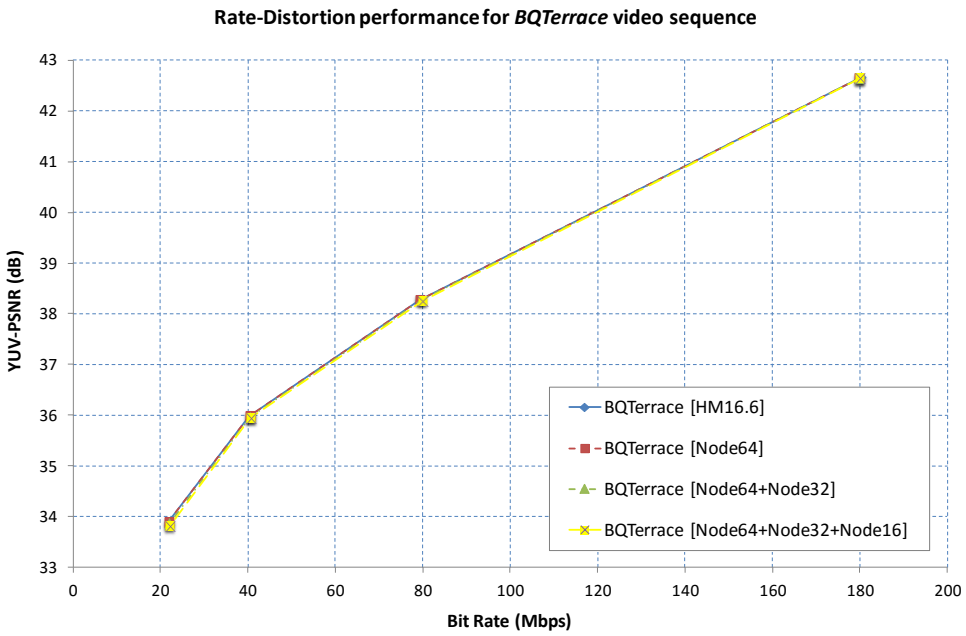


(a)

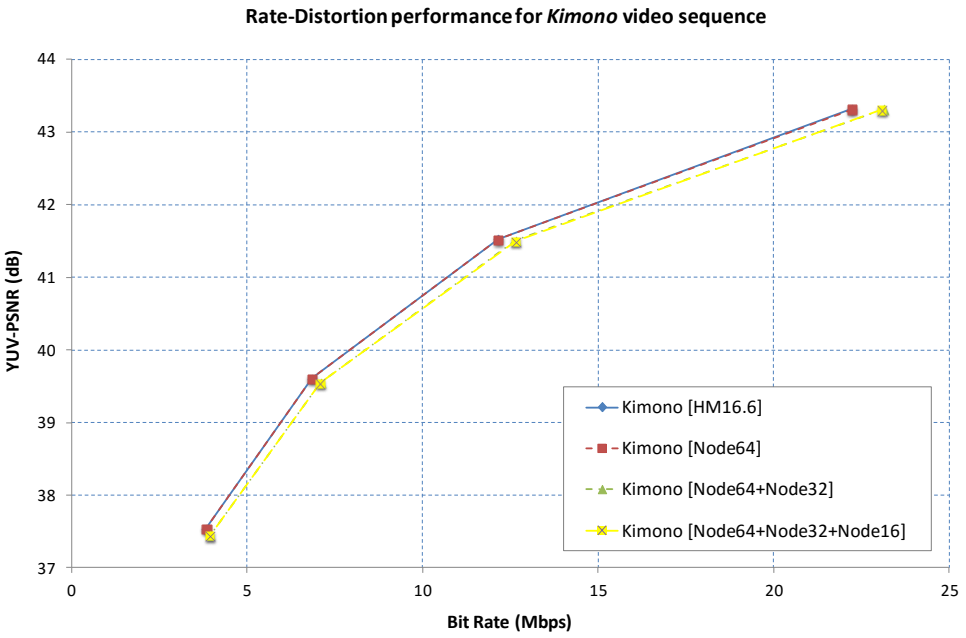


(b)

Figure 3.19. Rate-Distortion performance of Class A. (a) Best performance sequence. (b) Worst performance sequence

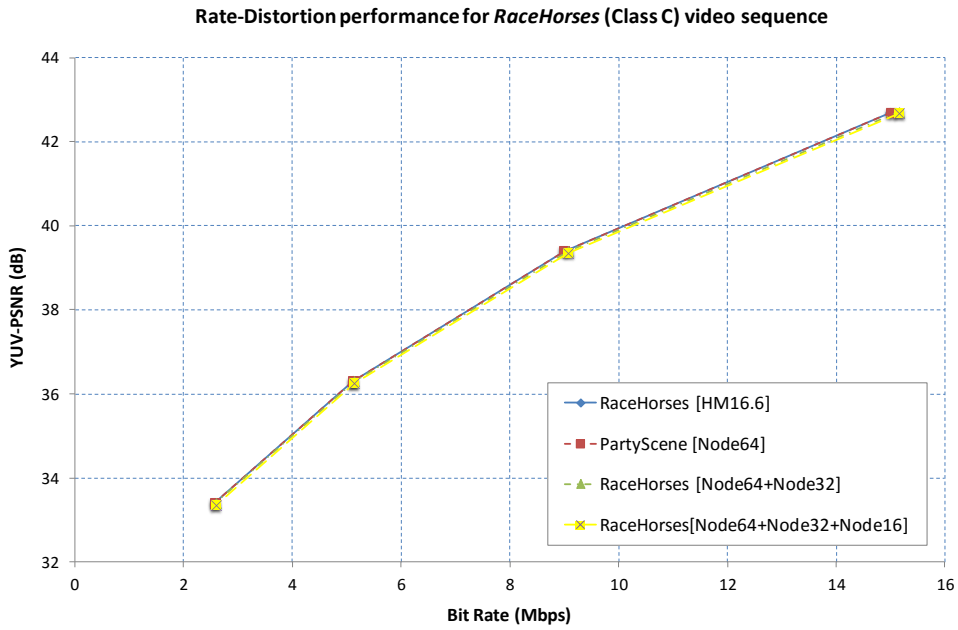


(a)

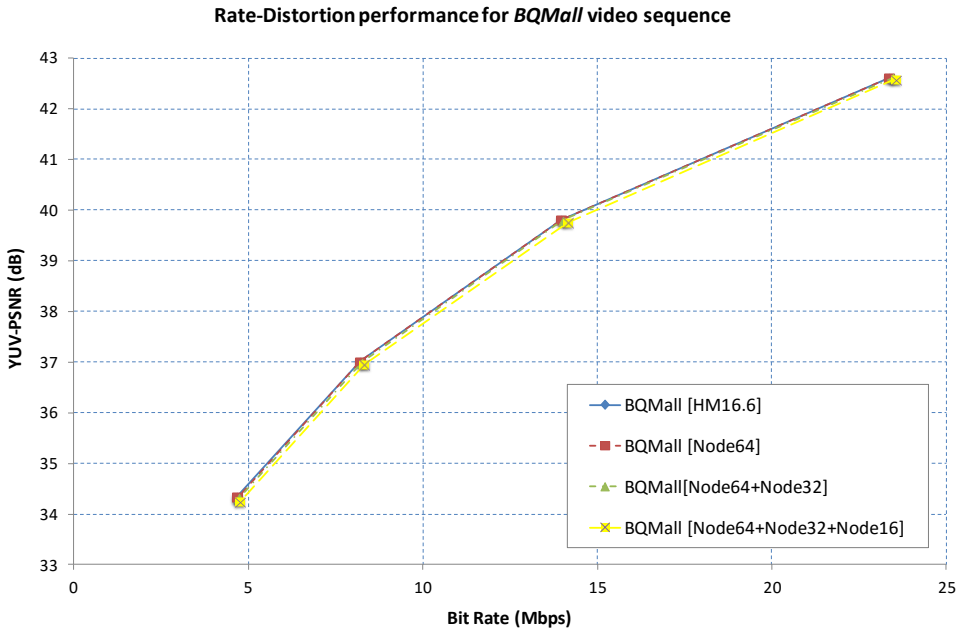


(b)

Figure 3.20. Rate-Distortion performance of Class B. (a) Best performance sequence. (b) Worst performance sequence

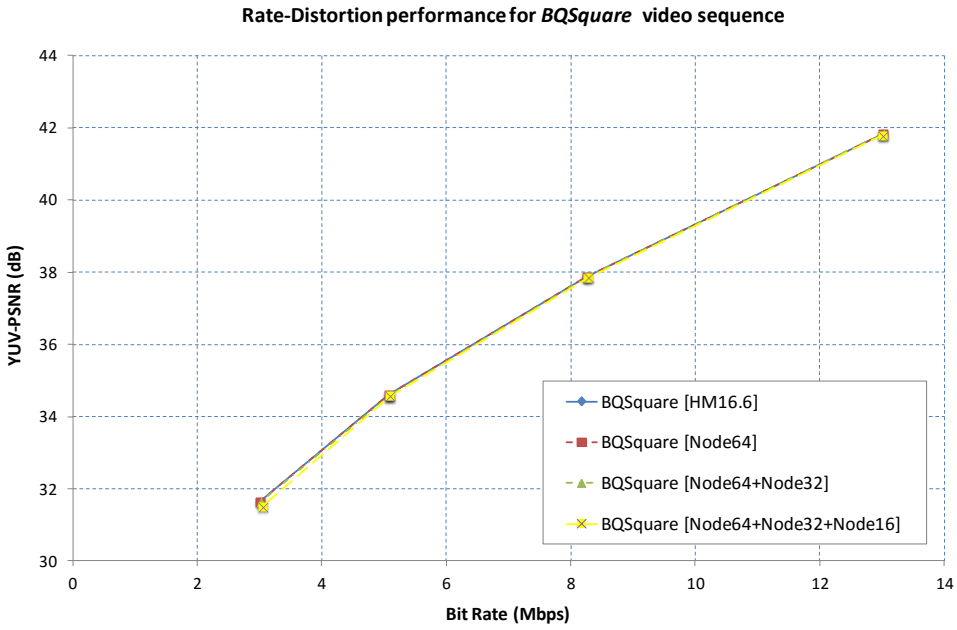


(a)

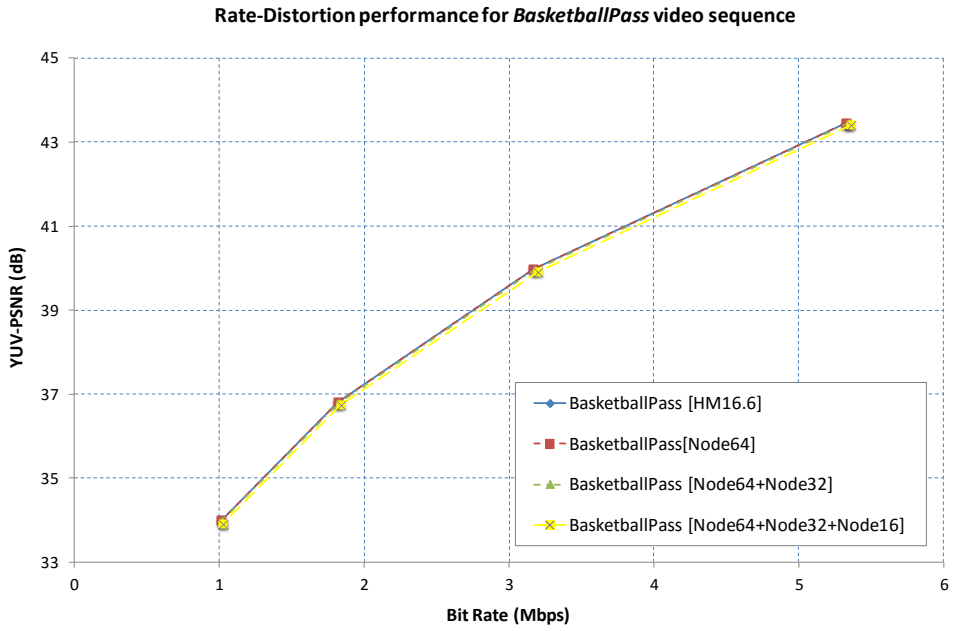


(b)

Figure 3.21. Rate-Distortion performance of Class C. (a) Best performance sequence. (b) Worst performance sequence

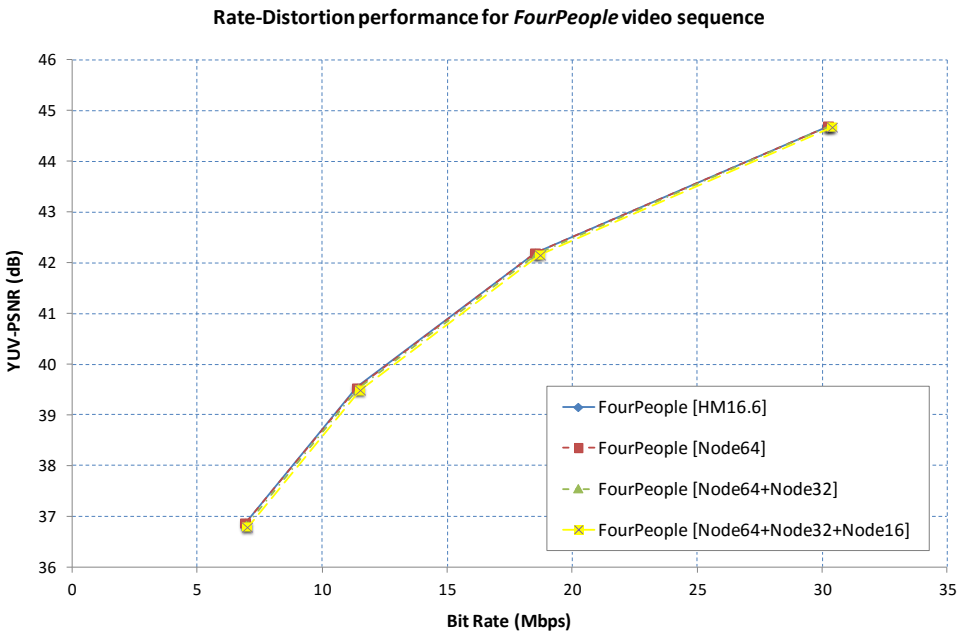


(a)

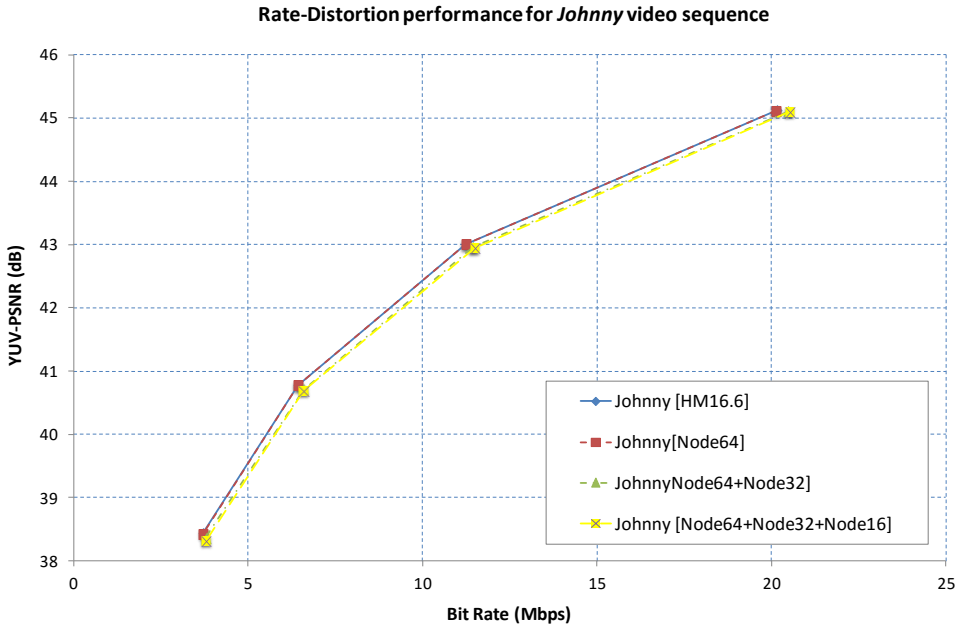


(b)

Figure 3.22. Rate-Distortion performance of Class D. (a) Best performance sequence. (b) Worst performance sequence

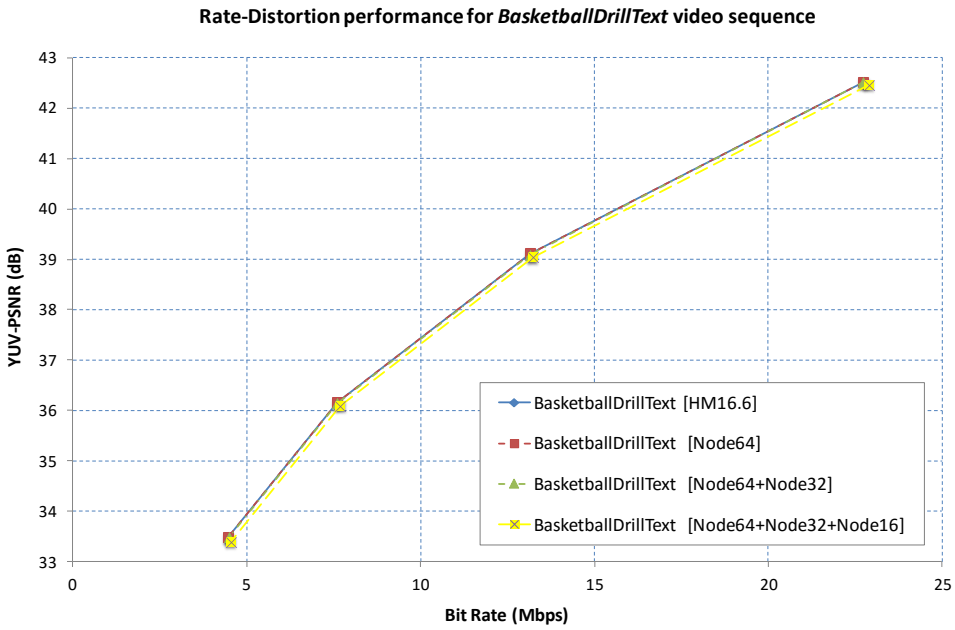


(a)

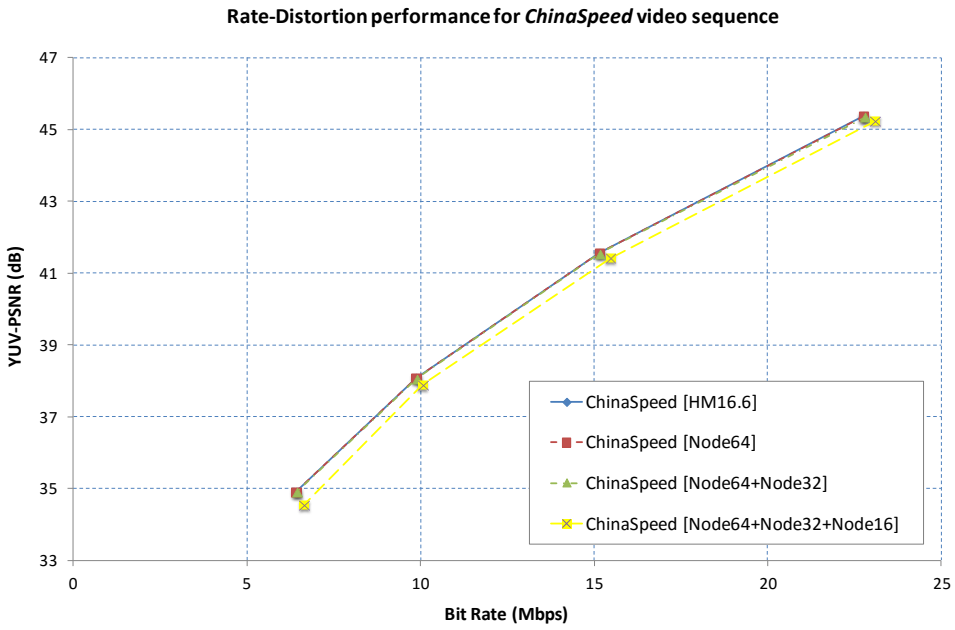


(b)

Figure 3.23. Rate-Distortion performance of Class E. (a) Best performance sequence. (b) Worst performance sequence



(a)



(b)

Figure 3.24. Rate-Distortion performance of Class F. (a) Best performance sequence. (b) Worst performance sequence

3.4.3.4 Simulation Results for Non-Trained QPs

As mentioned above in Section 3.3.6, a polynomial fitting was used to generate the thresholds $Th_i(QP)$, based on the "*polyfit*" function of MATLAB. Figure 3.30 shows an example source code implemented for the computation of the Th_2 for the full range of QPs (52 values). The four values of the training QPs (22, 27, 32, 37) are used as x_i points of the polynomial $P(x)$, and the four thresholds obtained in the *decision tree* stage (20, 28, 110, 380) are used as the $P(x_i)$ values. In this example, the *polyfit* function returns the coefficients in the approximating polynomial of degree 7, which are used by the *polyval* function to obtain the thresholds for the full range of QPs. For the lowest x_i values, due to fitting algorithm errors, negative thresholds could be obtained; therefore, a clipping to zero is applied for those values.

In order to evaluate the model's performance for a set of different QPs to those used for the training stage, additional simulations were conducted using two new sets of QPs: a *Low Tier* (QP15, QP20, QP25, and QP30) and a *High Tier* (QP25, QP30, QP35, and QP40). The experiment was run for the overall implementation, namely *Node64+Node32+Node16*, using the same CTCs but with the new QP sets.

```
% Thresholds interpolation for Th2 Node64x64
TrainedQPs =[22 27  32  37];
TrainedThs =[20 28 110 380];
QPrange=(1:52);
Coeff=polyfit(TrainedQPs,TrainedThs,7);
ListThresholds = round(polyval(Coeff,QPrange));
for i=1:52
    if ListThresholds(i)<0
        ListThresholds(i)=0;
    end
end
plot(ListThresholds(1,1:52));
```

Figure 3.250. Example of the computation of Th_2 for the full range of QPs, using the *polyfit* function.

Table 3.19 shows the results for both sets of QPs, showing similar average *Time Savings* with a slight CCR reduction for the *Low Tier* (47.34%), but a slight CCR improvement for the *High Tier* (56.56%). That variation is caused by CU size depending on the QP value, favouring the selection of the largest CU for high QPs, and therefore increasing the *Time Saving* for the *High Tier*, but reducing the *Time Savings* for the *Low Tier* because fewer CUs with large sizes (64x64 and 32x32) appear for lower QPs.

Table 3.19. Performance comparison for the proposed algorithm with HM16.6, for the non-trained QPs

Classification	Sequence	Frames	Low Tier QPs		High Tiers QPs	
			T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)
Class A (2560x1600)	Traffic	150	50.24	1.0	60.63	1.9
	PeopleOnStreet	150	42.85	0.8	55.49	1.5
Class B (1920x1080)	BasketballDrive	500	54.59	1.3	61.74	4.0
	BQTerrace	600	46.58	0.6	56.54	2.2
	Cactus	500	51.94	1.1	59.89	2.9
	Kimono	240	58.72	4.0	63.42	5.9
	ParkScene	240	52.61	1.2	60.75	2.0
Class C (832x480)	BasketballDrill	500	47.25	2.0	57.23	3.1
	BQMall	600	42.50	1.5	53.84	3.1
	PartyScene	500	37.04	0.9	51.19	3.0
	RaceHorses	300	48.59	1.3	56.91	2.2
Class D (416x240)	BasketballPass	500	40.11	1.4	49.24	2.7
	BQSquare	600	31.05	0.6	39.89	2.0
	BlowingBubbles	500	32.47	0.9	45.25	2.2
	RaceHorses	300	38.12	1.2	46.91	2.3
Class E (128x720)	FourPeople	600	49.65	1.3	60.73	2.6
	Johnny	600	59.01	2.9	65.49	5.0
	KristenAndSara	600	57.17	1.6	63.78	3.5
Class A			46.67	0.9	58.13	1.7
Class B			53.05	1.6	60.53	3.4
Class C			44.02	1.4	54.86	2.9
Class D			35.55	1.0	45.43	2.3
Class E			55.45	1.9	63.39	3.7
TOTAL			**47.34%**	**1.4%**	**56.56%**	**2.9%**

Both sets obtained a similar rate penalty to that obtained for the trained QPs, it being slightly lower (1.4%) for the *Low Tier* and slightly higher (2.9%) for the *High Tier*, confirming the good accuracy of the polynomial model for the computation of the thresholds of the decision trees.

3.5 PERFORMANCE COMPARISON TO DIFFERENT FAST CTU PARTITIONING DECISION ALGORITHMS IN HEVC

In Section 3.2.1, several fast *intra-prediction* algorithms were described that are based on the reduction of the number of CU sizes to be evaluated by the RMD, RDO and RQT stages. Those proposals achieved a *Time Saving* in the range of 20% to 65%, with different rate penalties, from a moderate 0.5% bit rate increasing to a significant 3.5% penalty in terms of *BD-rate*.

In order to compare the performance of those algorithms with the proposed classifier approach for the first two nodes, denoted as *Proposed Node(64+32)* and the overall implementation, denoted as *Proposed Node(64+32+16*), Table 3.20 summarizes the performance in terms of *Time Saving* and the *BD-rate*. The results are arranged in ascending order with respect to the *Time Saving* parameter. As can be observed, the *Proposed Node(64+32)* implementation with a 31% speed-up outperforms the first five proposals

[Cen15][Hua13][She13][Tia12][Kim13] in terms of *Time Saving*, obtaining a lower penalty for three of them [Cen15][She13][Kim13]. The Khan *et al.* [Kha13] and Sun *et al.* [Sun12] proposals achieve a better complexity reduction than *Proposed Node(64+32*), with 42% and 50%, respectively, but they are below the full *Proposed Node(64+32+16*) implementation, which achieves nearly 52%. Only the Cho *et al.* proposal [Cho13] with a 65% *Time Saving* surpasses the overall implementation, but it obtains a substantial rate penalty of 3.5%, nearly 1% more than the *Proposed Node(64+32+16*).

Table 3.20. Performance comparison between proposed algorithm and the Related Works

	Cen [Cen15]	Huang [Hua13]	Shen [She13]	Tian [Tia12]	Kim [Kim13]
HM Version	HM10.0	HM8.2	HM5.2	HM4.0	HM2.1
Time Saving	16%	20%	21%	29%	30.5%
BD-Rate	2.8%	0.5%	1.7%	0.5%	1.2%
	Proposed *Node(64+32)*	Khan [Kha13]	Sun [Sun12]	**Proposed *Node(64+32+16)***	Cho [Cho13]
HM Version	HM16.6	HM7.2	HM3.0	HM16.6	HM6.0
Time Saving	30.9%	42%	50%	51.9%	63.5%
BD-Rate	1.1%	1.2%	2.3%	2.6%	3.5%

A fair comparison between the proposed algorithm and the related works is difficult because those other works did not always use the full set of test sequences from the JCT-VC, and some of them do not use the CTCs recommended by the JCT-VC. Moreover, as can be observed in Table 3.20, most of them were implemented using the first HM versions, leading to significant differences in computational time and performance. The Sun *et al.* algorithm [Sun12] can be considered the best performing algorithm in the balance of *Time Saving*, 50%, and bit rate penalty, 2.3%. However, those results are obtained with a reduced number of JCT-VC test sequences. In order to make a fair performance comparison to the best performing algorithm using the same set of test sequences, Table 3.21 reports the results claimed by Sun's proposal and the proposed *Node(64+32)* and the overall *Node(64+32+16)* implementations.

The Sun *et al.* proposal outperforms the *Node(64+32)* implementation in terms of encoder time reduction, with 49.4% instead of 26%, but the bit rate penalty also increases by over 1.5%. On the contrary, the *Node(64+32+16*) implementation yields a lower bit rate of 1.85% instead of the 2% reported by Sun's proposal, and it also achieves a better computational complexity reduction with 50.92%, vs. 49.4%, improving the speed-up by over 1.5%. Consequently, the reported results prove that the proposed algorithm for the overall implementation achieves a high *Time Saving* for *intra-prediction* coding of over 50% compared to the HEVC reference model, with a slight bit rate penalty in terms of *BD-rate*.

Table 3.21. Performance comparison between the Sun et al. algorithm [Sun12] and the proposed algorithm

Classification	Sequence	Frames	**Proposed** ***Node (64+32)***		Sun [Sun12]		**Proposed** ***Node (64+32+16)***	
			T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)	T. Saving (%)	BD-Rate (%)
Class A	Traffic	150	29.39	0.9	49	2.41	56.61	1.6
	PeopleOnStreet	150	28.03	0.6	49	2.27	50.09	1.1
Class B	Cactus	500	29.77	1.0	51	3.04	56.95	2.1
	ParkScene	240	30.74	0.9	50	3.24	58.75	1.6
Class C	BasketballDrill	500	27.63	0.5	50	2.29	53.50	2.3
	BQMall	600	26.88	0.8	51	1.19	50.76	2.5
	RaceHorses	300	26.41	0.7	50	1.35	53.14	1.7
Class D	BasketballPass	500	23.54	0.6	50	2.25	45.66	2.1
	BlowingBubbles	500	18.58	0	47	1.02	40.55	1.7
	RaceHorses	300	19.75	0.4	47	1.05	43.20	1.8
Class A			28.71	0.75	49.00	2.34	53.35	1.35
Class B			30.26	0.95	50.50	3.14	57.85	1.85
Class C			26.97	0.67	50.33	1.61	52.47	2.17
Class D			20.62	0.33	48.00	1.44	43.14	1.87
TOTAL			**26.07**	**0.64**	**49.40**	**2.01**	**50.92**	**1.85**

CHAPTER 4

MODE DECISION BASED ON TEXTURE ORIENTATION DETECTION

In this chapter, a novel texture orientation detection algorithm is proposed for the fast *intra-prediction* mode decision in HEVC, which computes the Mean Directional Variance (MDV) along a set of *co-lines* with rational slopes. Based on the observation that the optimal *intra-prediction* mode is strongly correlated with the edge and texture orientation of the coding unit, a fast mode decision can be achieved by detecting the dominant gradient. The key point of the proposed algorithm is based on the hypothesis that pixel correlation is maximum in the texture orientation, and consequently the variance computed in that direction will obtain a low value compared to the variance computed in other directions. Another noteworthy feature of this proposal is the use of a set of rational slopes, which are exclusively defined in an integer position of the discrete lattice Λ, thus no pixel interpolation is required. According to the dominant gradient detected, a reduced set of candidate modes is selected to be further checked by the RDO stage. This approach avoids the exhaustive evaluation of the full number of directional modes carried out by the RMD in the HEVC reference model, significatively reducing the computational complexity. Simulation results show that the

proposed approach reduces the complexity of the *intra-prediction* mode decision by over 30%, with a slight bit rate increase of around 1%, compared to the HEVC reference model. In order to reduce the bit rate penalty of the MDV proposal, an enhancement to the MDV metric is also presented in this thesis, which introduces the concept of *Sliding Window*, denoted as MDV-SW. In this new approach, the directional variance computation for each NxN PU is expanded to the $(N+1)x(N+1)$ window, thus the neighbouring pixels used as reference samples for the construction of the predictor are also included in the calculation of the directional variance. This approach significatively improves the algorithm's performance for high QPs, because the distortion over the reference pixels is considered in the MDV computation, allowing a more accurate gradient detection. It is worth pointing out that this novel approach is not used in any other proposal in the literature for fast *intra-prediction* coding. Simulation results of MDV-SW report similar complexity reduction of 29.7%; however, the rate penalty is now reduced by more than half, reaching 0.4% in terms of *BD-rate*, which can be considered negligible. Finally, a performance comparison to state-of-the-art proposals shows that the proposed algorithm outperforms the best proposal in the balance of complexity reduction and bit rate penalty.

4.1 OBSERVATIONS AND MOTIVATION

As mentioned in Chapter 3, the latest two video coding standards, namely H.264/AVC [ISO-14496-10] and High Efficiency Video Coding (HEVC) [ISO-23008-2], have introduced directional predictive coding in the *intra-prediction* stage, which exploits the homogeneity of the pixels in the dominant texture and edge orientation of the coding units. The high efficiency of the *intra-prediction* stage in HEVC is mainly due to the high density of directional predictors, 33 modes compared to the four or nine modes available in H.264.

Optimal coding performance is obtained in HEVC by carrying out an exhaustive evaluation of all possible prediction modes, that is 33 directional modes and the two non-directional modes, for the full range of available PU sizes that a 64x64 CTU can split, namely 341. As was described in Chapter 3, the intra-prediction stage requires the computation of a set of sequential algorithms to obtain the final residual block. Those processes include the interpolation of the pixels used as reference samples from the neighboring PUs, sample pre-filtering, the construction of the predictor block, the computation of the residual block subtracting the predictor from the original block, and finally the post-filtering of the residual samples, depending on the PU size and the mode.
Consequently, the computation of the full *intra-prediction* process for the full number of available modes involves a huge computational complexity, which becomes a challenge in the real time implementation of the *intra-prediction* coding.

Although there exists a strong dependency between the optimal decision of the three key parameters of *intra-prediction*, the prediction mode, the PU size and the TU sizes, the HM reference software [JCT-VC Reference Software] has addressed this problem by dividing the optimal selection into three independent steps, namely RMD, RDO and RQT. A complexity analysis of the processes involved in the HEVC *intra-prediction* was presented in Section 3.1, showing that the RMD and RDO steps still take up over 80% of the total computing time of the *intra-prediction*. A solution replacing the RDO stage, which carried out the evaluation of all PU sizes, was proposed in Chapter 3 through a fast CTU partitioning decision algorithm. That approach proved that PU size selection using ML techniques can achieve a complexity reduction of over 50%, in a trade-off with a moderate efficiency penalty of less than 2%.

In this chapter, a fast *intra-prediction* mode decision algorithm is proposed which can improve upon the performance of the RMD scheme implemented in the HM reference software. The RMD algorithm can be considered as a fast mode algorithm, because while it evaluates the 35 prediction modes, it does not compute the full *intra-prediction* process described above, instead carrying out a rough cost estimation of each mode by using the Hadamard transform. Although that RMD stage speeds up the mode decision, the approach proposed in this chapter will show that a 30% complexity reduction in encoding time can be obtained, with a negligible rate increase of 0.4%.

Many proposals have been presented in the literature for the fast optimal mode decision in HEVC [Pan05][Jia12][Sil12][Yan12][Che13][Yao14][Zha14]. Most of them are based on pixel gradient detection in the spatial domain by computing the gradient of the image using the Sobel filter [Gup13] or other similar filters. Then, those algorithms estimate the global block orientation, selecting the angle with the highest magnitude from the gradient histogram, and the HEVC mode nearest to the detected gradient angle is selected as candidate. This technique has been proved to be robust when high-energy edges are present in the image, but natural images often have wide areas with weak edges, or even no edges, so this approach can be inefficient for the intra-prediction mode decision.

This fact has motivated the algorithm presented in this chapter, which proposes a novel metric based on the directional variance computation along the digital *co-lines* with rational slopes, and named the Mean Directional Variance metric [Rui15a][Rui15c]. The proposed approach is based on the hypothesis that the optimal mode for intra-prediction has a strong correlation with the image texture orientation, which is characterized by a low variance in the texture direction. Because most of the angular modes defined in HEVC require pixel interpolation due to the slope orientation, the MDV metric proposes the use of different slopes with orientations that can be computed in integer pixel location within the PU, avoiding the pixel interpolation process, which involves a high computational complexity.

The MDV is a modified version of the metric proposed by Jayachandra and Makur in [Jay10], which calculates the cumulative variance along digital lines. The directional variance metric has proved to be an accurate technique to detect texture orientation with low complexity, achieving a high level of accuracy in the framework of image processing using directional wavelets and directionlets.

Unlike [Jay10], this proposal estimates the dominant image orientation by computing the MDV along the digital *co-lines* using rational slopes defined in a discrete lattice [Con98]. In addition, the MDV metric can be implemented in just one pass in a scalable manner for different block sizes, as is required by most image processing and video coding applications.

In [Rui15d] is proved that there exists a strong dependence between the optimal mode selected by the RDO and the distortion applied, set by the QP parameter. Based on that observation, an enhancement to the MDV algorithm is proposed in this thesis, which uses a sliding window for the directional variance computation including the reference pixels used for the construction of the intra predictors, and denoted as MDV-SW. This approach is the key point presented in this proposal, because it allows a high level of performance for the full range of the QPs, reducing by half the rate penalty obtained by MDV. The use of the reference pixel for texture detection is a novel approach that is not used in any other proposal in the literature.

The experimental results show that the MDV-SW approach slightly improves upon the encoding performance of the RMD algorithm in terms of bit rate, reducing the computational complexity by around 30%, with a quality degradation of 0.05dB in terms of PSNR. Regarding the bit rate penalty, this is reduced to a negligible 0.4% in terms of *BD-rate*, which is a good trade-off between computational complexity and Rate-Distortion performance.

4.2 RELATED WORKS FOR FAST INTRA-PREDICTION MODE DECISION IN HEVC

In this section[1], the most relevant fast mode selection proposals for *intra-prediction* coding in HEVC are presented. Although the fast partitioning algorithms can reduce *intra-prediction* complexity by over 50%, most of them use an exhaustive search of the optimal mode by applying the RMD scheme implemented in the HM reference software. Due to the high density of angular prediction directions in HEVC, the scientific community has mainly addressed this challenge with approaches based on gradient detection algorithms.

Gradient detection plays an important role in a number of disciplines, such as computer vision, image processing, pattern recognition, and image classification, and special interest has arisen for image and video coding. Natural and synthetic images usually have a mixture of smooth regions, textured regions with different patterns, and edges of different strength levels. The texture and edge directionality is a local feature or attribute of the images, which has been frequently exploited in the field of image processing for image segmentation, object identification or labelling, and for diagnosis in many modalities of medical images.

Edge and texture directionality detection is commonly performed in the spatial domain by using first-order and second-order derivatives. Sobel, Prewitt, Roberts and Canny [Cas96] are some of the well-known first derivative methods used for gradient detection, while the Laplacian operator is the preferred method for the second-order derivative class [Cas96]. Those algorithms are commonly computed in two orthogonal directions by the image convolution with a local 3x3 mask, achieving a high level of performance when strong edges are present. However, there are areas in natural images that often lack sharp edges, or the edges are softened or noisy, causing poor performance of those algorithms. A description of edge detection problems and a proof review of the linear and non-linear Gaussian-based edge detection methods are presented in [Mit02].

Many proposals have also been presented by the research community for texture gradient detection in the frecuency domain, which allows an edge characterization in different scales and orientations. The most popular approaches are the separable 2D Discrete Wavelets Transform (DWT) [Mal89], Gabor filter [Gab46], and more complex algorithms using anisotropic directional basis functions such as *curvelets* [Can99] and *contourlets* [Do05]. Adaptive versions of those algorithms, which carry out a directional adaptive decomposition, have also been proposed, such as *wedgelets* [Lok10] and *directionlets* [Bai11].

[1] Matrix and vectors are denoted by boldfaced letters.

Recently, many proposals [Pan05] [Jia12] [Sil12] [Yan12] [Che13] [Yao14] [Zha14] have been presented for the fast *intra-prediction* mode decision in HEVC, using some of the above-mentioned approaches. Most of them are based on pixel gradient detection in the spatial domain computing the first-order derivative in two orthogonal directions, which is applied to each pixel of the image. The gradient $\vec{\boldsymbol{G}}$ of an image or a pixel array, denoted as *f*, can be calculated by Equation (4.1), where G_x and G_y represent the horizontal and vertical gradient respectively.

$$\vec{\boldsymbol{G}} = \nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial \mathrm{x}} \\ \frac{\partial \mathrm{f}}{\partial \mathrm{y}} \end{bmatrix} \quad (4.1)$$

The most simple method for computing G_x and G_y is the pixel difference along the rows and columns of the image, but this approach shows poor sensitivity for the edges with slope orientation with directions other than the horizontal and vertical. Several two-dimensional operators have been proposed to compute the orthogonal gradients, the most popular being the 3x3 operators in the spatial domain, such as Sobel [Gup13], Prewitt [Pre70] and Roberts [Rob65], which are shown in Figure 4.1. The direction of the gradient, denoted as α, is then given by the hyper-function depicted in Equation (4.2), and because the gradient is perpendicular to the edge direction, the pixel edge orientation is obtained by $\alpha \pm 90º$.

	G_x	G_y
Sobel	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Figure 4.1. Sobel, Prewitt and Roberts 3x3 impulse response array for 3x3 operators.

$$\alpha = arctan\left(\frac{G_y}{G_x}\right) \quad (4.2)$$

The authors in [Pan05] [Jia12] [Che13] use this approach by computing the Sobel pixel gradient, followed by a rough estimation of the gradient amplitude, denoted as GradAmp($\vec{G}$) and given in Equation (4.3), which is less accurate than the computation of the square root of the sum of squares of G_x and G_y, but it requires a smaller computational cost. In order to obtain the dominant gradient of an image, the Histogram of Oriented Gradients (HOG) is

calculated by adding the gradient amplitude of each orientation detected, and the direction with the highest gradient amplitude is selected as the dominant orientation.

$$\text{GradAmp}(\vec{\boldsymbol{G}}) = |G_x| + |G_y| \quad (4.3)$$

The Four Edge Orientation Detection (FEOD) algorithm is proposed in [Sil12], which divides an image or block of pixels B into four sub-blocks B0, B1, B2 and B3 with half resolution, and the mean of each sub-block, denoted as $\mu_{B0,}$ μ_{B1}, μ_{B2} and μ_{B3} ,is computed. Afterwards, the edge energy of four directions (0, $\pi/4$, $\pi/2$, $3\pi/4$), and also the absence of borders, is calculated according to Equations (4.4)-(4.8), selecting as dominant edge orientation the direction with the highest value, as is depicted in Equation (4.9).

$$\text{Edge}(0) = |\mu_{B0} + \mu_{B1} - \mu_{B2} - \mu_{B3}| \quad (4.4)$$

$$\text{Edge}(\pi/4) = |\sqrt{2}\mu_{B0} - \sqrt{2}\mu_{B3}| \quad (4.5)$$

$$\text{Edge}(\pi/2) = |\mu_{B0} - \mu_{B1} + \mu_{B2} - \mu_{B3}| \quad (4.6)$$

$$\text{Edge}(3\pi/4) = |\mu_{B0} - \mu_{B1} + \mu_{B2} - \mu_{B3}| \quad (4.7)$$

$$\text{Non\_Edge} = |2\mu_{B0} - 2\mu_{B1} - 2\mu_{B2} + 2\mu_{B3}| \quad (4.8)$$

$$\text{Dominat\_Edge} = max\{\text{Edge}(0), \text{Edge}(\pi/4), \text{Edge}(\pi/2), \text{Edge}(3\pi/4), \text{Non}_{\text{Edge}}\} \quad (4.9)$$

A similar approach, named Dominant Edge Assents (DEA), is proposed by *Yao et al.* in [Yao14], which computes the same four gradient directions (0, $\pi/4$, $\pi/2$, $3\pi/4$), but using a different algorithm. A new sub-block B4 is defined in the centre of the image, as it is depicted in Figure 4.2, and its mean μ_{B4} is also used together with the $\mu_{B0,}$ μ_{B1}, μ_{B2} and μ_{B3} means to compute the gradient directions following the Equations (4.10)-(4.13). Lastly, the dominant gradient orientation is given by selecting the minimum DEA direction value, as is shown in Equation (4.14).

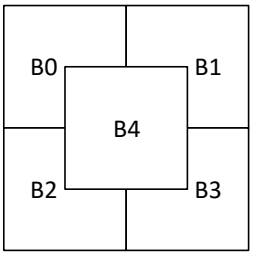


Figure 4.2. Sub-blocks definition in the DEA algorithm

$$\mathrm{DEA}(0) = |\mu_{B1} - \mu_{B0}| + |\mu_{B3} - \mu_{B2}| \quad (4.10)$$

$$\mathrm{DEA}(\pi/4) = |\mu_{B2} - \mu_{B4}| + |\mu_{B4} - \mu_{B1}| \quad (4.11)$$

$$\mathrm{DEA}(\pi/2) = |\mu_{B2} - \mu_{B0}| + |\mu_{B3} - \mu_{B1}| \quad (4.12)$$

$$\mathrm{DEA}(3\pi/4) = |\mu_{B3} - \mu_{B4}| + |\mu_{B4} - \mu_{B0}| \quad (4.13)$$

$$\text{Dominat\_Edge} = min\{\mathrm{DEA}(0), \mathrm{DEA}(\pi/4), \mathrm{DEA}(\pi/2), \ \mathrm{DEA}(3\pi/4)\} \quad (4.14)$$

In [Yan12], a more accurate gradient detector is proposed by computing the Directional Sum of Absolute Differences (DSAD) in 33 angular directions. The orientation having the lowest DSAD is selected as the dominant edge direction, but most of the 33 orientations do not have an integer pixel position, so a 2-taps interpolation method is applied in those instances, which increases the complexity. Equations (4.15)-(4.16) show a DSAD computation example for two directions, $(\pi/4\,, 13\pi/32)$, p(i,j) being the pixels of a 4x4 pixel array.

$$\begin{aligned}\mathrm{DSAD}(\pi/4) = \frac{1}{9}[&|\mathrm{p}(0,1) - \mathrm{p}(1,0)| + |\mathrm{p}(0,2) - \mathrm{p}(1,1)| + |\mathrm{p}(1,1) - \mathrm{p}(2,0)| \\ &+ |\mathrm{p}(0,3) - \mathrm{p}(1,2)| + |\mathrm{p}(1,2) - \mathrm{p}(2,1)| + |\mathrm{p}(2,1) - \mathrm{p}(3,0)| \\ &+ |\mathrm{p}(1,3) - \mathrm{p}(2,2)| + |\mathrm{p}(2,2) - \mathrm{p}(3,1)| + |\mathrm{p}(2,3) - \mathrm{p}(3,2)|] \quad (4.15)\end{aligned}$$

$$\begin{aligned}&\mathrm{DSAD}(13\pi/32) \\ &= \frac{1}{9}\begin{bmatrix} |\mathrm{p}(0,1) - 13/32\,\mathrm{p}(1,0) - (1 - 13/32)\mathrm{p}(0,0)| + \\ |\mathrm{p}(0,2) - 13/32\,\mathrm{p}(1,1) - (1 - 13/32)\mathrm{p}(0,1)| + \cdots \\ \ldots\ldots\ldots\ldots + |\mathrm{p}(2,3) - 13/32\,\mathrm{p}(3,2) - (1 - 13/32)\mathrm{p}(2,2)| \end{bmatrix} \quad (4.16)\end{aligned}$$

There are also other proposals which use approaches that are completely different to the ones mentioned above, for instance, the one proposed by *Zhao et al.* in [Zha14], which uses the spatial distribution of the best intra-prediction modes in the picture to model a 2-order Markov random field.

As will be described below, the algorithm presented in this chapter can be included in this group of non-conventional approaches for intra-prediction, because it does not use any spatial filter or frequency analysis to detect the texture or edge orientation of the image.

4.3 FAST INTRA-PREDICTION MODE DECISION APPROACH

In this section, a novel intra-prediction mode decision algorithm based on texture orientation detection is presented, and which is used as part of a low complexity *intra-prediction* algorithm in HEVC.

The approach of this proposal is based on texture orientation detection by computing the directional variance along certain spatial directions with rational slopes defined in an integer lattice Λ. The orientation with the lowest average directional variance is chosen as the dominant local texture orientation, selecting a reduced set of candidate modes to be evaluated by the RDO stage. The proposed algorithm [Rui15a][Rui15c] makes it possible to replace the exhaustive evaluation of the 35 *intra-prediction* modes carried out for the RMD stage, and also to reduce the number of candidate modes to be further evaluated by the RDO to three for any PU size. The complexity reduction in both stages, that is RMD and RDO, achieves a considerable speedup for the overall HEVC *intra-prediction* coding, with a negligible bit rate increase. Finally, the novel concept of *Sliding Window* is introduced in the proposed algorithm, achieving a notable penalty reduction in terms of bit rate for the same complexity reduction. This enhanced approach makes it possible to outperform other state-of-the-art solutions in the field of fast intra-prediction mode decision algorithms.

4.3.1 Orientation of the Angular Intra-Prediction Modes in HEVC

As was described in Chapter 2, the new *intra-prediction* scheme defined in HEVC [Min08] uses a similar scheme to the one used in H.264/AVC, but increases the number of directional prediction modes from 9 to 35. That huge density of angular orientations in HEVC exploits the spatial correlation between PU pixels and the pixels from the top-row and left-column of the neighbouring PUs much better, allowing an efficient coding of a high number of texture orientations.

In HEVC, the 33 angular *intra-prediction* modes are defined covering an arc of π radians and with 1/32 of fractional precision between two integer pixel positions of reference pixels, $P_{ref}(i,j)$, which are used for the construction of the prediction samples. Figure 4.3, depicts the angular modes for a top-left pixel of a PU, denoted as $P(0,0)$, which uses as reference samples five neighbouring pixels, $P_{ref}(-1,1)$ and $P_{ref}(-1,0)$ from the left neighbouring

PU, $P_{ref}(-1,-1)$ from the top-left neighbouring PU, and $P_{ref}(0,-1)$ and $P_{ref}(1-10)$ from the top neighbouring PU. As can be observed, just five modes are using as prediction sample an integer pixel; the other 28 modes need to compute an interpolation between the two nearest pixels.

Consequently, the angular prediction modes can be classified into two categories: the first one is composed of the five modes whose orientations match the integer position of the reference pixels (solid pixels in Figure 4.3), named *Integer Position Modes* (IPM). They are the horizontal H_{10}, vertical V_{26}, and the three diagonal modes: H_2, V_{18} and V_{34}. The second category includes the rest of the modes, whose orientation point is between two reference samples (dashed pixels), and therefore the predictor is computed by the interpolation of the two nearest P_{ref} samples. The second category is named *Fractional Position Modes* (FPM). The DC and Planar modes are not angular modes, so they are not included in any of these categories.

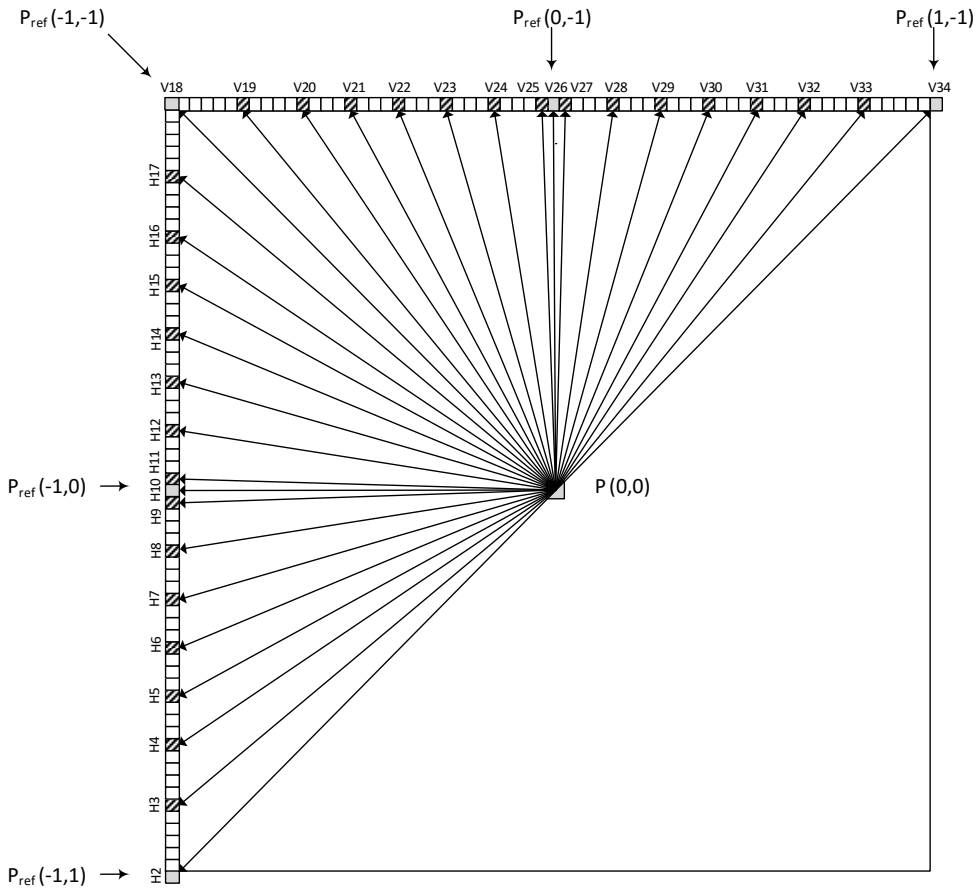


Figure 4.3. Orientation of the angular intra-prediction modes in HEVC.

Table 4.1 summarizes the details of the 33 angular modes, denoted as M_i, r_i being the angular mode orientation, which is defined as a rational slope $r = r_y/r_x$, and Θ_i is the orientation angle, such that $\Theta_i = \arctan(r_i)$. As can be noted, the IPMs have an integer value of slope r_i, meanwhile the FPMs do not.

Lines with rational slopes are commonly used in digital image processing, as they favour their definition in the discrete space $\mathbb{Z}^2$, denoted as integer lattice $\Lambda \in \mathbb{Z}^2$. Given a continuous line with rational slope $r = r_y/r_x$, it can be represented by Equation (4.17),

$$y = r \cdot x + d, \ \forall r \in Q \qquad (4.17)$$

Table 4.1. Orientations and slopes of angular modes in HEVC

Mode M_i	Angle Θ_i	Slope r_i	Mode M_i	Angle Θ_i	Slope r_i
H_2	5π/4	-1	V_{18}	3π/4	1
H_3	39π/32	-13/16	V_{19}	23π/32	16/13
H_4	38π/32	-21/32	V_{20}	22π/32	32/21
H_5	37π/32	-17/32	V_{21}	21π/32	32/17
H_6	36π/32	-13/32	V_{22}	20π/32	32/13
H_7	35π/32	-9/32	V_{23}	19π/32	32/9
H_8	34π/32	-5/32	V_{24}	18π/32	32/5
H_9	33π/32	-1/16	V_{25}	17π/32	16/1
H_{10}	π	0	V_{26}	π/2	∞
H_{11}	31π/32	1/16	V_{27}	15π/32	-16/1
H_{12}	30π/32	5/32	V_{28}	14π/32	-32/5
H_{13}	29π/32	9/32	V_{29}	13π/32	-32/9
H_{14}	28π/32	13/32	V_{30}	12π/32	-32/13
H_{15}	27π/32	17/32	V_{31}	11π/32	-32/17
H_{16}	26π/32	21/32	V_{32}	10π/32	-32/21
H_{17}	25π/32	13/16	V_{33}	9π/32	-16/13
			V_{34}	π/4	-1

Only if '*x*' takes integer values that are multiples of r_x, that is $x = k \cdot r_x, \forall k \in \mathbb{Z}$, '*y*' takes an integer position in Λ. Accordingly, the distance between two points with integer positions belonging to a line with rational slope $r_i$is determined by the r_x, r_y parameters. As can be observed in Table 4.1, the FPM modes, H_3, V_{19}, H_9, V_{25}, H_{11}, V_{27}, H_{17} and V_{33}, have rational slopes of *±m/16 and ±16/m,* $(\forall m = 1,13)$. That means they are defined with at least two points in integer position for PU sizes larger than 16x16, so for 32x32 and 64x64 PUs, but not for 16x16, 8x8 and 4x4 PUs. The other 20 FPM modes have a rational slope of *±m/32 and ±32/m,* $\forall m = 5,9,13,17,21$, thus their lines are defined with two points in integer position for PU sizes larger than 32x32, so that means only for the 64x64 PU size.

This is the reason why in this proposal the orientations with the slopes defined by the FPM modes in HEVC are not used. Instead, the proposed texture orientation detection algorithm only uses the IPM slopes of HEVC and a new set of eight rational slopes, but with smaller r_x, r_y parameters to make sure the lines are fully defined in integer position, even for the smaller PUs of 4x4.

4.3.2 Sub-sampling of Integer Lattice

The directional variance metric described in [Jay10] uses the traditional definition of *digital line* for the discrete space $\mathbb{Z}^2$, such that $L(r, n)$ is a *digital line* with rational slope $\forall\, r \in Q$ and $\forall\, n \in \mathbb{Z}$, meaning that every pixel with position $(x, y) \in \mathbb{Z}^2$ is associated exclusively to one *digital line*, defined as:

$$y = \lceil r \cdot x \rceil + n, \quad \forall\, \mathrm{x, n} \in \mathbb{Z}, \forall\, |\mathrm{r}| \leq 1, \text{ or} \qquad (4.18)$$

$$x = \lceil \mathrm{y}/r \rceil + n, \quad \forall\, \mathrm{y, n} \in \mathbb{Z}, \forall\, |\mathrm{r}| > 1 \qquad (4.19)$$

Therefore, for each rational slope r a set of integers, denoted as n, can be found which define the *digital lines* $L(r, n)$ and that completely partitions the 2D discrete space $\mathbb{Z}^2$. *Digital lines* have been widely used in different fields, including image processing, image coding and artificial vision among others. However, they do not provide enough accuracy for gradient detection in images or pixel blocks with low resolution, such as the PUs defined in HEVC which small size is 4x4 pixels.

Figure 4.4(a) depicts an example of a set of *digital lines L(*$1/3$ *, n).* As can be observed, most of the pixels belonging to a *digital line*, represent a horizontal orientation ($r = 0$) instead of the defined slope $r = 1/3$, and consequently that expression does not represent the line orientation with high accuracy. For this reason, the concept of down-sampling lattice in two-dimensional space described in [Lei08] has been used, together with the *co-lines* definition used in lattice theory [Vel06].

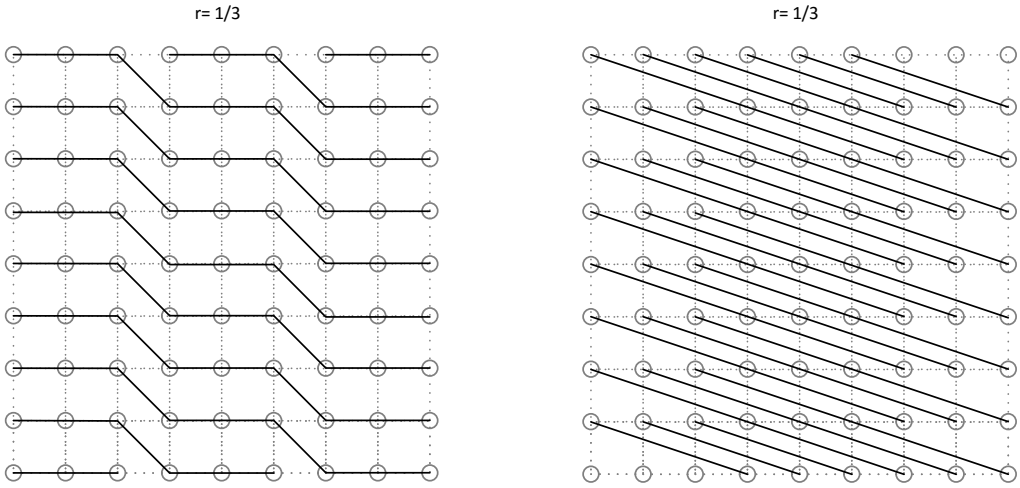


Figure 4.3. (a) Digital line L(r=1⁄3, n), ∀ n ∈ Ƶ. (b) *Co-lines* ⟦CL⟧ _Sk (1⁄3,n) , ∀ n ∈ Ƶ

According to [Lei08], in a 2D system an integer lattice Λ can be obtained by down-sampling in two directions $\boldsymbol{d_1} = [d_{x1}, d_{y1}]^{\mathrm{T}}$ and $\boldsymbol{d_2} = [d_{x2}, d_{y2}]^{\mathrm{T}}$ of the cubic integer lattice $\mathbb{Z}^2$. Hence, the sublattice $\Lambda \subset \mathbb{Z}^2$ can be formally represented by a non-singular 2x2 integer matrix, denoted as *sub-sampling matrix* or *generator matrix*, $\boldsymbol{M}_{\Lambda}$, such that $d_{x1}, d_{y1}, d_{x2}, d_{y2} \in \mathbb{Z}$, that is:

$$\boldsymbol{M}_{\Lambda} = [d_1\ d_2] = \begin{bmatrix} d_{x1} & d_{x2} \\ d_{y1} & d_{y2} \end{bmatrix} \qquad (4.20)$$

The use of the rational slopes as sub-sampling directions in $\boldsymbol{M}_{\Lambda}$, in their vector form $\boldsymbol{r}_1$=$[r_{x1}, r_{y1}]^{\mathrm{T}}$ and $\boldsymbol{r}_2$=$[r_{x2}, r_{y2}]^{\mathrm{T}}$, allows to achieve a sub-lattice Λ, which describes the points of the line with slopes r_1 *and* r_2, and thus it facilitates the variance computation along of those orientations. Lattice theory [Con98] states that given a *generator matrix* $\boldsymbol{M}_{\Lambda}$, the lattice $\mathbb{Z}^2$ is partitioned into $|det(\boldsymbol{M}_{\Lambda})|$ number of cosets of the lattice Λ, which are shifted versions of the sub-lattice Λ. Each coset can be obtained by a shifting vector $\boldsymbol{s}_k = [s_{kx}, s_{ky}]^T \ \forall \in\ k = 0,1,..det(\boldsymbol{M}_{\Lambda}) - 1$.

In [Vel06] the concept of *co-line*, denoted as $CL_{Sk}(r, n)$, is introduced, and it is defined as the intersection of a kth coset in lattice Λ and a *digital line* $L(r, n)$. Therefore, the pixels (x, y) belonging to the *co-lines* with slope r_1 and r_2 can be obtained by the lineal combination of both vectors $\boldsymbol{r}_1, \boldsymbol{r}_2$, $\forall\ \mathrm{c}_1, \mathrm{c}_2 \in \mathbb{Z}$, such that,

$$\begin{bmatrix} x \\ y \end{bmatrix} = c_1 \begin{bmatrix} r_{x1} \\ r_{y1} \end{bmatrix} + c_2 \begin{bmatrix} r_{x2} \\ r_{y2} \end{bmatrix} + \begin{bmatrix} s_{kx} \\ s_{ky} \end{bmatrix} \qquad (4.21)$$

The *co-lines* approach has been proved as an efficient tool in image processing and image coding when it is applied in non-Cartesian orientations, such as *directionlets* [Bai11] and directional wavelet transforms [Bam92]. It should be noted that the distance between two consecutive pixels belonging to the same *co-line* is always the same, and its position is free of any type of interpolation process. Figure 4.4(b) shows an example of the *co-lines* $CL_{sk}(1/3\,, n)$, equivalent to the *digital lines* depicted in Figure 4.4(a). As can be noted, the *co-lines* represent the geometrical orientation with greater accuracy than the traditional *digital lines*.

Figure 4.5 shows two cosets examples. In Figure 4.5(a) the cosets are obtained from the subsampling matrix with directional vectors $\boldsymbol{r}_1 = [1,-1]^{\mathrm{T}}$ and $\boldsymbol{r}_2 = [1,1]^{\mathrm{T}}$. Given that $|det(\mathbf{M}_{\boldsymbol{\Lambda}})| = 2$ there are two cosets, the first, represented by grey points, corresponds to the shifting vector $\boldsymbol{s}_1 = [0,0]$; and the second coset with white points corresponds to $\boldsymbol{s}_2 = [0,1]$. Figure 4.5(b) depicts similar example for slopes defined by directional vectors $\boldsymbol{r}_1 = [2,-1]^{\mathrm{T}}$ and $\boldsymbol{r}_2 = [1,2]^{\mathrm{T}}$, but given that $|det(\mathbf{M}_{\boldsymbol{\Lambda}})| = 5$. In this case exist five cosets determined by the shifting vectors $\boldsymbol{s}_0 = [0,0]$, $\boldsymbol{s}_1 = [-1,0]$, $\boldsymbol{s}_2 = [0,-1]$, $\boldsymbol{s}_3 = [0,1]$, $\boldsymbol{s}_4 = [1,0]$, and these are represented by the grey, doted, white, back, and lined points, respectively.

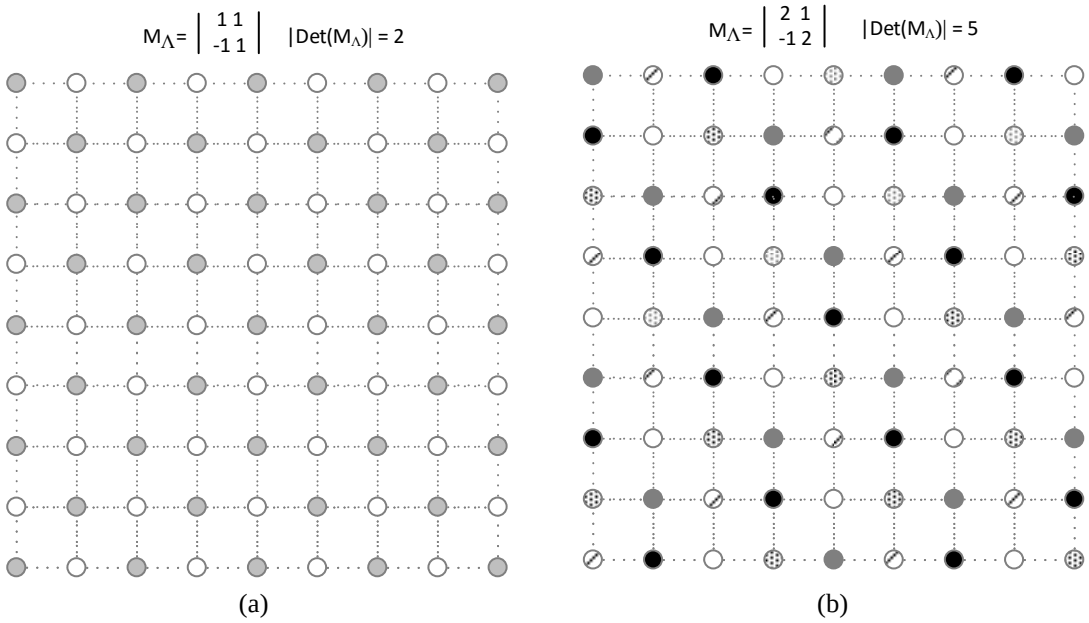


Figure 4.4. (a) Two cosets generated from direction vectors $\boldsymbol{r_1} = [\mathbf{1},-\mathbf{1}]^{\mathbf{T}}$ and $\boldsymbol{r_2} = [\mathbf{1},\mathbf{1}]^{\mathbf{T}}$ of the sub-sampling matrix $\mathbf{M}_{\boldsymbol{\Lambda}}$. (b) Five cosets generated from direction vectors $\boldsymbol{r_1} = [\mathbf{2},-\mathbf{1}]^{\mathbf{T}}$ and $\boldsymbol{r_2} = [\mathbf{1},\mathbf{2}]^{\mathbf{T}}$ of the sub-sampling matrix $\mathbf{M}_{\boldsymbol{\Lambda}}$.

4.3.3 Selection of the Co-lines Orientation Selection

The proposed algorithm [Rui15a][Rui15c] computes the variance along the pixels of the *co-lines* with a set of rational slopes r_i, which can be obtained using the *generator matrix* $\boldsymbol{M_\Lambda}$ described above, where any couple of rational slopes r_i, r_j can be used as vector directions $\boldsymbol{d_1}$ and $\boldsymbol{d_2}$ in $\boldsymbol{M_\Lambda}$. However, each set of *co-lines* with a specific slope r_i can be calculated independently from the second direction in $\boldsymbol{M_\Lambda}$, which simplifies the computing process. Consequently, from Equation (4.21) the *co-lines* definition with a similar expression from the traditional definition of the *digital line* can be obtained, as is shown in Equation (4.18) and (4.19). They are defined in Equation (4.22) and (4.23), where $\boldsymbol{r}_1$ is the rational slope of the *co-lines*, $\forall\, c_2 \in \mathbb{Z},\ k = 0,..det(\mathbf{M_\Lambda}) - 1$.

$$y = r_1\, x + n \ \ \forall\, |r_1| > 1,\ n = c_2\left(r_{y2} - r_1 r_{x2}\right) - r_1 s_{kx} + s_{ky} \ \ or \qquad (4.22)$$

$$x = y/r_1 + n \ \ \forall\, |r_1| \le 1,\ n = c_2\left(r_{x2} - r_{y2}/r_1\right) - s_{ky}/r_1 + s_{kx} \qquad (4.23)$$

From Equation (4.22) and (4.23), fixing c_2, for each coset defined for the vector $\boldsymbol{s}_k$ the expression of the pixels (x, y) belonging to *co-lines* with slope r_1 can be achieved. In order to estimate the dominant texture orientation in each PU, twelve rational slopes r_i, $(\forall\, i = 0,..,11)$ have been selected. Four of them with the same slope as the IPM modes, that is horizontal, vertical and two diagonal orientation (the diagonal H_2 and V_{34} are considered the same in terms of slope), and the other eight slopes are defined with slopes close to some of the FPMs modes, but with rational slopes of $\pm 1/2$, $\pm 1/4$, ± 2, and ± 4. Table 4.2 collects the set of six *generator matrices* M_{Λ_i}, the vector directions used for the integer lattice sub-sampling, and the respective cosets defined by $det(\mathbf{M_{\Lambda_i}})$.

Table 4.2. Six generator matrices defining the twelve rational slopes and the respective cosets

M_{Λ_i}	Vector directions	Generator matrix	Cosets
M_{Λ_0}	r_3, r_9	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1
M_{Λ_1}	r_0, r_6	$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$	2
M_{Λ_2}	r_1, r_7	$\begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix}$	5
M_{Λ_3}	r_5, r_{11}	$\begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix}$	5
M_{Λ_4}	r_2, r_8	$\begin{bmatrix} 4 & 1 \\ -1 & 4 \end{bmatrix}$	17
M_{Λ_5}	r_4, r_{10}	$\begin{bmatrix} 4 & 1 \\ 1 & -4 \end{bmatrix}$	17

The chosen orientations have a maximum slope factor of 4 instead of 32, as the HEVC modes have. Consequently, there are always two points that determine a *co-line* with rational slope r_i even for the smaller PU size of 4x4. Figure 4.6 shows the set of twelve rational slopes that have been chosen for the analysis of the dominant gradient in each PU, defined as co-lines in the sub-sampled integer lattice Λ.

With the aim of comparing the orientations of the selected slopes with the HEVC modes directionality, Figure 4.7 depicts both, the 33 angular *intra-prediction* modes defined in HEVC (grey lines) and the twelve proposed *co-lines* with rational slope (blue dashed lines) which have been chosen for the analysis of the dominant gradient. As can be observed, the *co-lines* r_0, r_3, r_6, and r_9 overlap the angular modes H_2, V_{10}, V_{18}, V_{26} and V_{34}, thus such modes can be estimated with high accuracy from respective *co-lines*.

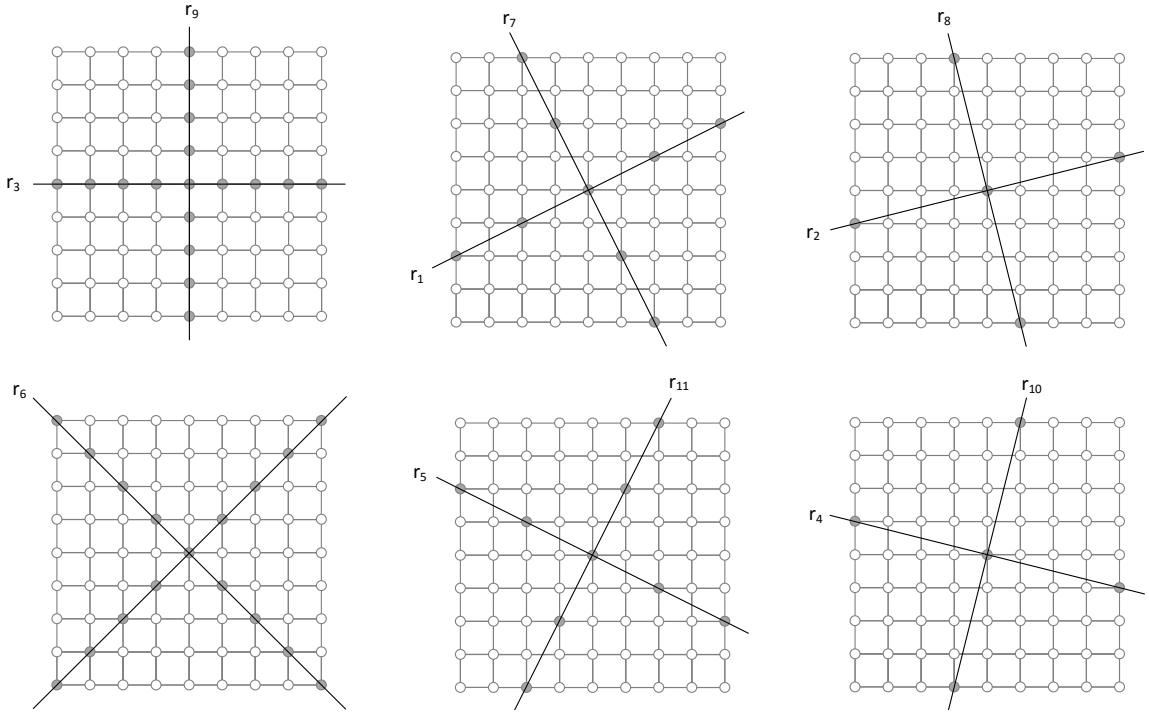


Figure 4.5. *Co-lines* $\boldsymbol{r_0}$ to $\boldsymbol{r_{11}}$ selected to compute of directional variance

The *co-lines* with slopes r_1, r_5, r_7 and r_{11}, are mostly overlapping at the angular modes H_5, V_{15}, V_{21}, and V_{31}, so these *co-lines* can also be considered as a good estimation of the co-located modes. Finally, the four *co-lines* with slopes r_2, r_4, r_8 and r_{10}, are located near the middle of two modes H_7 and H_8 for the slope r_2, H_{12} and H_{13} for the slope r_4, V_{23} and V_{24} for the slope r_8, and V_{28} and V_{29} for the slope r_{10}. Although these last four slopes do not match with only one specific angular mode, they are useful for highlighting that the dominant gradient is in-between two angular modes of HEVC.

With the aim of considering the remaining angular modes that are not covered by each of the twelve proposed slopes r_i, twelve classes denoted as C_i have been defined. Each class selects a set of candidate modes M_i on the left and right side of the respective rational slopes r_i. Table 4.3 shows the features of the proposed *co-lines* r_i, including the rational slope, the orientation angle Θ_i, class C_i, and the candidate modes M_i for the different PU sizes. It should be noted that the *co-line* r_1 is the only one slope that selects four angular modes in its class. This is due to modes H_2 and H_{34} of HEVC being the same in terms of orientation, so the two horizontal H_2, H_3, and the two vertical V_{33}, V_{34} modes, are selected as candidates.

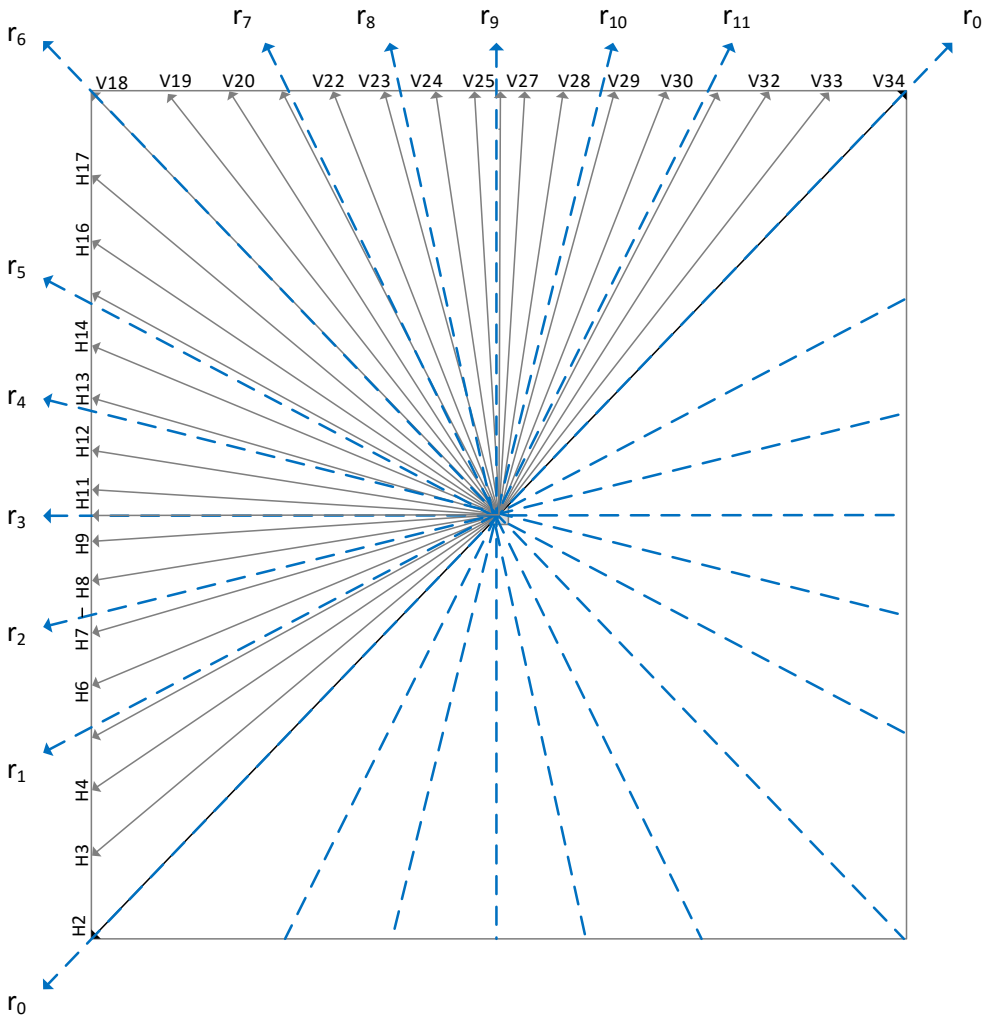


Figure 4.6. Set of angular intra predictions in HEVC (solid lines), and the *co-lines* defined with rational slopes $\boldsymbol{r_0}$ to $\boldsymbol{r_{11}}$ (dashed blue lines) for the dominant gradient analysis.

Based on empirical simulations, one candidate mode on the left and right side of each class has been added for the smaller PU sizes of 8x8 and 4x4. This is motivated by the fact that the *co-lines* are defined by a few pixels in small size blocks, and the accuracy of the

orientation detection is also lower. In addition, the two non-angular modes, DC and Planar are included as candidates for all the classes, because they match quite well with weak strength edge images, but the DC mode is also a good predictor for the extremely high complexity PUs, which have strong edges in several orientations simultaneously.

Table 4.3. Slope, angle and candidate modes assigned to defined *co-lines*

Co-line	Θ_i	Slope (r_y/r_x)	PU Class (C_i)	Candidates modes M_i for PU_{64},PU_{32}, PU_{16}	Additional Candidates M_i for PU_8, PU_4
r_0	5π/4	-1/1	I	H_2, H_3, V_{33}, V_{34}	H_4,V_{32}
r_1	8π/7	-1/2	II	H_4, H_5, H_6	H_3,H_7
r_2	27π/25	-1/4	III	H_7, H_8	H_6,H_9
r_3	π	0	IV	H_9, H_{10}, H_{11}	H_8, H_{12}
r_4	23π/25	1/4	V	H_{12}, H_{13}	H_{11}, H_{14}
r_5	6π/7	1/2	VI	H_{14}, H_{15}, H_{16}	H_{13}.H_{17}
r_6	3π/4	1/1	VII	H_{17}, V_{18}, V_{19}	H_{16}, V_{20}
r_7	9π/14	2/1	VIII	V_{20}, V_{21}, V_{22}	V_{19},V_{23}
r_8	29π/50	4/1	IX	V_{23}, V_{24}	V_{22}, V_{25}
r_9	π/2	∞	X	V_{25}, V_{26}, V_{27}	V_{24}, V_{28}
r_{10}	21π/50	-4/1	XI	V_{28}, V_{29}	V_{27}, V_{30}
r_{11}	5π/14	-2/1	XII	V_{30}, V_{31}, V_{32}	V_{29}, V_{33}

4.3.4 Computation of Mean Directional Variance along Co-lines

Cumulative variance along *digital lines* is proposed in [Jay10], which is proven as an efficient metric for texture orientation detection in images with high resolution. However, due to the constraints imposed by factors such as the PU sizes, the high density of the *intra-prediction* modes in HEVC, and the need to reduce the encoder's computational burden, the following modifications and novel features are proposed:

1. The variance is computed along the *co-lines* according to Equation (4.22) and Equation (4.23), instead of using the *digital lines* expression. As mentioned above, the digital lines do not describe texture orientation with enough fidelity in small image blocks. The location of the pixels belonging to each *co-line* is calculated off-line, and their (x, y) coordinates are stored in a Look-Up-Table for fast computation.

2. In order to reduce the computational complexity of the variance, an approximation of the variance, denoted *textbook one-pass* algorithm, proposed by *Chan et al*. in [Cha79] is used. This approach does not require passing through the data twice, once to calculate the mean and again to compute the sum of squares of deviations from

the mean. Equation (4.24) shows the directional variance expression using the *textbook one-pass* approach,$p_j(r_i, n)$ being the pixels belonging to $CL(r_i, n)$, and N the number of pixels of the nth *co-line* with slope r_i:

$$\sigma^2[\mathrm{CL}(r_i, n)] = \frac{1}{N}\left[\sum_{\mathrm{j}=0}^{\mathrm{N}} \mathrm{p_j}^2(r_i, n) - \frac{1}{\mathrm{N}}\left(\sum_{\mathrm{j}=0}^{\mathrm{N}} \mathrm{p_j}(r_i, n)\right)^2\right] \qquad (4.24)$$

3. The directional variance is also computed in a scalable manner for all PU sizes, using the concept of *co-segment* belonging to a *co-line*. A *co-line* $CL(r_i, n)$ can be divided into K *co-segments* $CS_m(r_i, n)$, such that,

$$\mathrm{CL}(r_i, n) = \bigcup_{\mathrm{m}=1}^{\mathrm{K}} \mathrm{CS_m}(r_i, n) \quad (4.25)$$

The concept of *co-segment* allows the understanding of the scalability feature of the proposed mean directional variance algorithm. Since the directional variance computation along a $CL(r_i, n)$ for the largest 64x64 PU requires going through the pixels belonging to smaller PUs, the variance of the *co-segments* $CS_m(r_i, n)$belonging to smaller PUs are computed in the same pass. This approach makes it possible to calculate the directional variance of the 64x64 PU, but at the same time, the directional variance of the remaining 340 PUs is obtained, saving a huge computation time.

Figure 4.8 exemplifies a *co-line* for the slope r_5 for one 16x16 PU and its respective *co-segments*, which cross the PU but also pass through the three 8x8 PUs (lined, dotted and trellis blocks) and four 4x4 PUs. As can be observed, there are four *co-segments*, $CS_1(r_5, n)$ to $CS_4(r_5, n)$, made up of two pixels each (gray shading pixels), which allow the computation of the variance for the four respective 4x4 PUs. The variance for the *co-line* of the top-left 8x8 PU can be computed from the union of $CS_1(r_5, n)$ and $CS_2(r_5, n)$, while the variance for the top-right and bottom-right 8x8 PUs matches with the variance of the *co-segments* $CS_3(r_5, n)$ and $CS_4(r_5, n)$, respectively. Following the same logic, the variance for the 16x16 PU is obtained from $m = 14 CS_m(r_5, n)$.

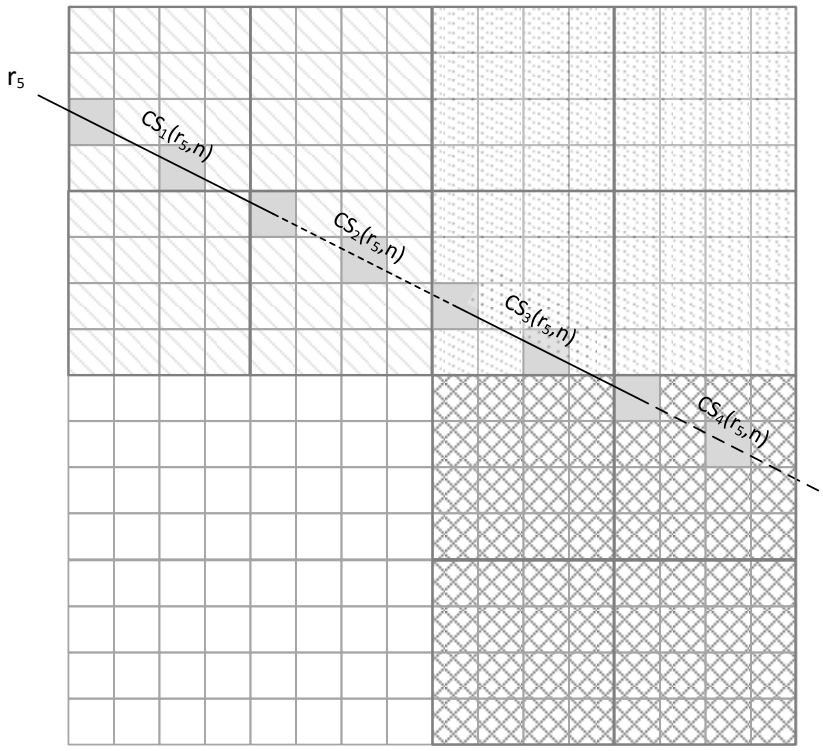


Figure 4.7. Example of four *co-segments* with slope $\boldsymbol{r_5}$comprising a *co-line*, in a 16x16 PU.

4. Finally, instead of calculating the cumulative variance of the *digital lines* as is proposed in [Jay10], the MDV is computed as the average of the individual variances obtained along the *L co-lines* with the same r_i orientation, as is shown in Equation (4.26).

$$MDV(r_i) = \frac{1}{L}\sum_{\mathrm{n}=1}^{L} \sigma^2[\mathrm{CL}(r_i, n)] \quad (4.26)$$

For small block sizes where few lines are defined for a specific r_i orientation, the cumulative variance is not able to obtain the dominant gradient with accuracy. Based on the simulations carried out, it has been found that the proposed MDV metric in this thesis describes the directional homogeneity better than the cumulative variance, especially for small-sized blocks.

It should be noted that the MDV can be applied to any image size, and any rational slope. For each combination of block size and slope r_i, a different number of *co-lines* (*L*) and different length of the *co-line* (*N*) will be obtained. As an example, Table 4.4 shows the *L* and *N* values for an 8x8 PU, where it can be observed that the longest lines are the horizontal, vertical and diagonal orientations (r_0, r_3, r_6 and r_9)) with a length of 8. The slopes with more lines are the slopes with the highest rational factors (r_2, r_4, r_8 and r_{10}), but they are also the shortest ones, with a length of two pixels.

Table 4.4. Example of number and lengths of *co-lines* for an 8x8 PU.

Co-line	Number of co-lines (L)	Lengths of the co-lines (N)
r_0	13	2,3,4,5,6,7,8,7,6,5,4,3,2
r_1	18	2,2,3,3,4,4,4,4,4,4,4,4,4,3,3,2,2
r_2	28	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
r_3	8	8,8,8,8,8,8,8,8
r_4	28	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
r_5	18	2,2,3,3,4,4,4,4,4,4,4,4,4,3,3,2,2
r_6	13	2,3,4,5,6,7,8,7,6,5,4,3,2
r_7	18	2,2,3,3,4,4,4,4,4,4,4,4,4,3,3,2,2
r_8	28	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
r_9	8	8,8,8,8,8,8,8,8
r_{10}	28	2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2
r_{11}	18	2,2,3,3,4,4,4,4,4,4,4,4,4,3,3,2,2

4.3.5 Computation of Mean Directional Variance Using Sliding Window

As has been described in Section 4.3.4, the MDV is computed for each PU using the pixels of *co-segments* $CS_m(r_i, n)$ belonging to that PU in a scalable manner. However, the angular modes in HEVC *intra-prediction* use as reference sample the neighboring pixels of *decoded* PUs, denoted as $PU_d(x, y)$, which have been previously distorted by the quantification factor QP. The analysis described in [Rui14b] proves the strong dependence of the optimal mode selected by the RDO on the QP parameter, especially for high QP values, which cause a strong smoothing of the reference pixels, thus the correlation between the reference samples and the PU's pixels (which are not yet distorted) can be modified. That misalignment is the reason that, for the same original PU, the optimal mode changes depending on the QP used.

Consequently, an enhancement to the MDV algorithm is proposed in this section. The main idea of this approach is to expand the window of the MDV computation for an NxN PU to a window of $(N+1)x(N+1)$ pixels, overlapping the left column and top row of the window with the left and top decoded pixels of the neighbouring PUs. Figure 4.9(a) depicts an example of the computation of the MDV for a 4x4 PU(i,j) along the slope r_6, where the reference samples belonging to decoded PUs (PU_d) are lined. As can be observed, five *co-segments* with lengths of 2, 3 and 4 pixels are defined for such slope. Figure 4.9(b) presents the new MDV approach, where the new $(N+1)x(N+1)$ window allows the MDV computation using the reference pixels from the neighbouring PUs, and two new *co-segments* are now available.

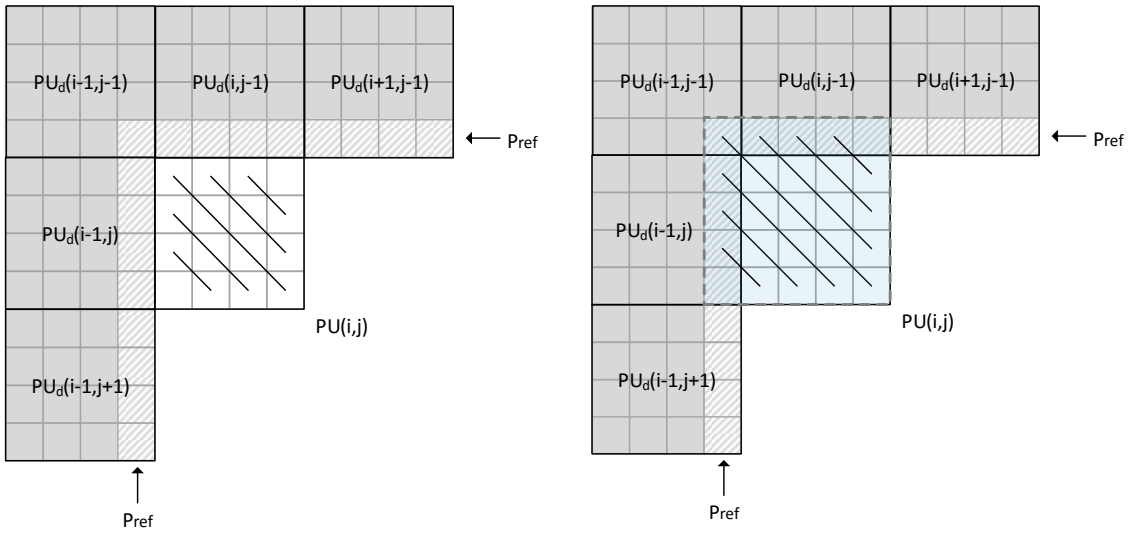


Figure 4.8. (a) Example of MDV computation of a 4x4 PU. (b) Example of MDV-SW computation over an expanded window for a 4x4 PUs.

Accordingly, the MDV is computed in five overlapping window sizes of 65x65, 33x33, 17x17, 9x9 and 5x5 pixels, and thus this novel approach is named *Mean Directional Variance with Sliding Window*. Figure 4.10 illustrates an example of the MDV-SW computation of four consecutive 4x4 PUs $\{PU(i+k, j+l) \mid \forall\, k,l = 0,1\}$, showing that the sliding windows are overlapping each other, in order to capture the neighboring reference samples.

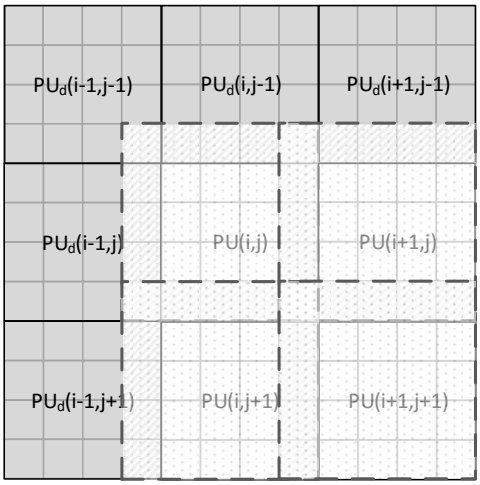


Figure 4.9. Example of MDV-SW using overlapped windows for the evaluation of 4x4 PUs

The MDV-SW implementation needs a slight computation increase compared with the MDV due to the extended window size and therefore the border pixels are being computed twice for the adjacent PUs. However, the texture orientation accuracy will be significantly better,

because the pixels used as references by the angular intra-*prediction* process are now considered in the MDV computation.

4.4 PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed fast *intra-prediction* mode decision algorithm, it was implemented in the HEVC reference software HM 16.6 [JCT-VC Reference Software]. The non-modified HM 16.6 was used as anchor using the same test sequences and encoding parameters. The simulations were run on the same architecture that was used for the evaluation of the fast CTU partitioning decision algorithm proposed in this thesis, and described in Section 3.4.

4.4.1 Encoding Parameters

The experiments were conducted on the same set of 22 test sequences and using the same test conditions described in Section 3.4.1 for the evaluation of the fast coding tree block partitioning decision algorithm proposed in this thesis. Both the test sequences and test conditions are recommended by the JCT-VC [JCTVC-L1100]. Table 3.16 of Section 3.4.1 gives the test sequence features in terms of resolution, frame rate and number of frames, and Figure 3.23 of that section depicts the first frame of each test sequence used in the simulations.

4.4.2 Metrics

The algorithm's performance was evaluated using the same two metrics described in Section 3.4.2 for the evaluation of the fast coding tree block partitioning decision algorithm. These are the computational complexity reduction in terms of *Time Saving*, and rate-distortion in terms of *BD-rate*. In the performance comparison process between different algorithms, it is quite common to find that one algorithm can obtain better a *Time Saving* but a worse *BD-rate* than the other proposal, or vice versa, thus a neutral comparison is sometimes difficult to carry out.

With the aim of establishing a fair criterion for the proposal ranking, a new metric was proposed by *Correa et al.* in [Cor15], which computes the ratio between the rate increase and the complexity reduction (multiplied by 100), denoted as $\Delta Rate/\Delta Time$ and expressed in Equation (4.27). This metric is a good indicator of the overall algorithm performance, reporting the trade-off between compression and implementation efficiency. Consequently,

a lower ΔRate/ΔTime ratio points to a better performance, since it is interpreted as a good balance between a low rate penalty and a high complexity reduction.

$$\Delta\text{Rate}/\Delta\text{Time} = \frac{\text{BD}-\text{rate}}{\text{Time}-\text{Saving}} \cdot 100\% \quad (4.27)$$

The above metrics are proved as good measures of the overall average rate penalty and complexity reduction for a set of QPs used in the simulations, but they do not indicate the performance for individual QPs. Thus, with the aim of analysing the algorithm performance dependence on the QP parameter, two additional metrics have been included:

- The quality difference between the proposed algorithm and the HEVC reference software (HM16.6) for a specific QP, in terms of PSNR, denoted as ΔPSNR and expressed in Equation (4.28).

$$\Delta\text{PSNR (dB)} = PSNR_{prop} - PSNR_{HM16.6} \quad (4.28)$$

- The bit rate difference between the proposed algorithm and the HEVC reference software (HM16.6) for a specific QP, as a percentage of the bit rate used for the HEVC reference software, denoted as $\Delta Bitrate$ and expressed in Equation (4.29).

$$\Delta\text{Bitrate (\%)} = \frac{(Bitrate_{prop} - Bitrate_{HM16.6})}{Bitrate_{HM16.6}} \cdot 100\% \quad (4.29)$$

In order to show the performance of proposed MDV metric graphically, a polar representation of the normalized MDV has also been used. For this purpose, the MDV is computed for each of the twelve rational slopes r_i, and later the maximum variance is obtained by calculating $\max\{MDV(PU, r_j) \mid \forall\, j = 0,..,11\}$; and finally each MDV is normalized by this factor. This metric is denoted as $\overline{MDV(PU, r_i)}$, PU being the block size analyzed and r_i the rational slope, that is,

$$\overline{MDV(PU, r_i)} = \frac{MDV(PU, \text{r}_\text{i})}{\max\{MDV(PU, r_\text{j}) \mid \forall\, \text{j}=0,..,11\}} \quad (4.30)$$

Given that the r_i orientations just covering an arc of π radians, from $\pi/4$ to $5\pi/4$, the polar representation of the $\overline{\text{MDV}(\text{PU}, \text{r}_\text{i})}$ is also plotted using the twelve symmetric values of the normalized variance, covering the 2π radians. It allows the understanding of the MDV in terms of directionality.

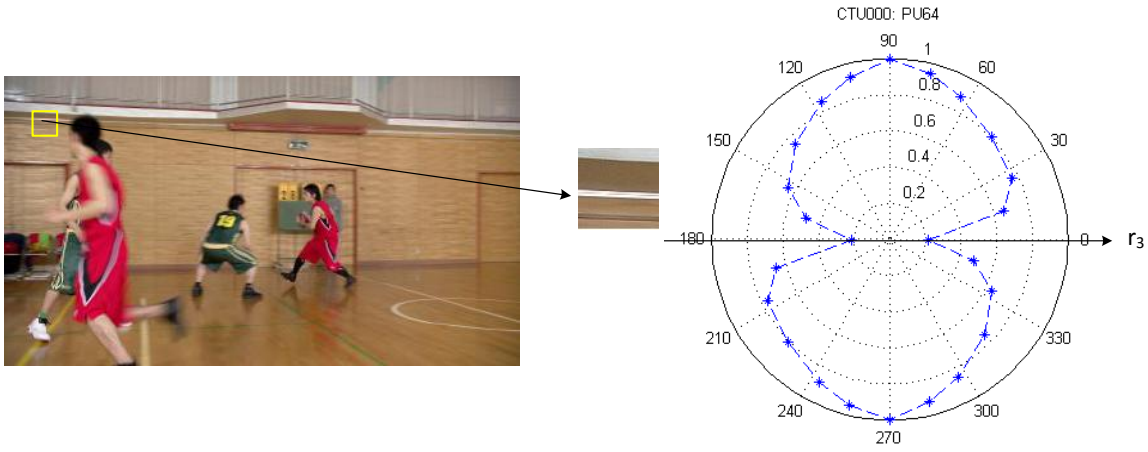
4.4.3 Simulation Results

In this section, simulation results for three different test is shown: the results for a set of natural and synthetic images in term of polar diagram using the $\overline{\mathrm{MDV}(\mathrm{PU}, \mathrm{r}_1)}$ metric, and the results for the MDV and MDV-SW algorithms using the JCT-VC test sequences in terms of *BD-rate*, TS, ΔPSNR and ΔBitrate metrics.

4.4.3.1 Simulation Results for Natural and Synthetics Images

In this section, the performance of the proposed MDV metric is shown in a perceptual way, using the polar representation of the normalized MDV, as was described in Section 4.4.2. Three simulations are reported using some natural and synthetic test images, which present different texture features, which allow the understanding of the MDV metric in terms of directionality.

The first simulation is carried out using the *Basketball* sequence belonging to the JCT-VC test sequences [JCTVC-L1100], and its first frame is shown in Figure 4.11(a). As can be observed, the selected PU has strong edges in the top, middle and bottom part of the image, with an orientation that slightly slop over the horizontal. Figure 4.11(b) depicts the polar diagram showing a minimum proof in the orientation of the r_3 slope, which represents the horizontal directionality. The horizontal texture orientation is clearly detected, obtaining for the horizontal direction a directional variance value of 80% lower than the maximum variance value, corresponding to a $\pi/2$ orientation.



(a) (b)

Figure 4.10. (a) First frame of *BasketballDrive* sequence and the selected 64x64 PU. (b) Polar diagram of $\overline{\boldsymbol{MDV}(\boldsymbol{PU}, \boldsymbol{r_i})}$ for selected PU.

Regarding the second test, a synthetic image representing the "*unit hyperbola*" function has been evaluated. That function is defined as $x^2 - y^2 = 1$, and its conjugate function as $x^2 - y^2 = -1$, both functions being rotated $\pi/4$ in relation to the Cartesian axes. The 64x64 PU is depicted in Figure 4.12(a), and its 3D representation is shown in Figure 4.12(b). As can be noted, the image has two dominant gradients which are strongly fixed in the $\pi/4$ and $3\pi/4$ directions, orthogonal to one another.

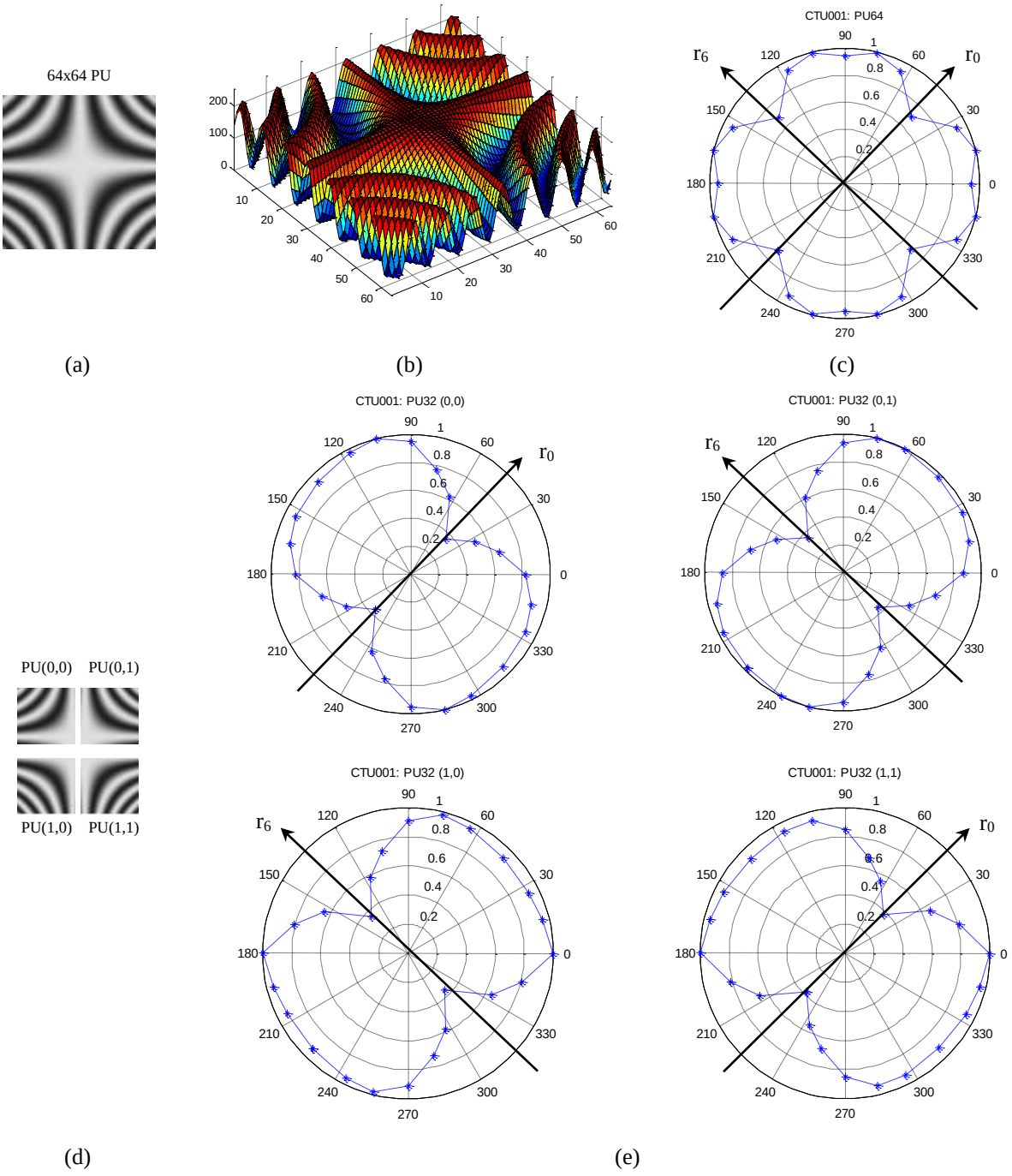
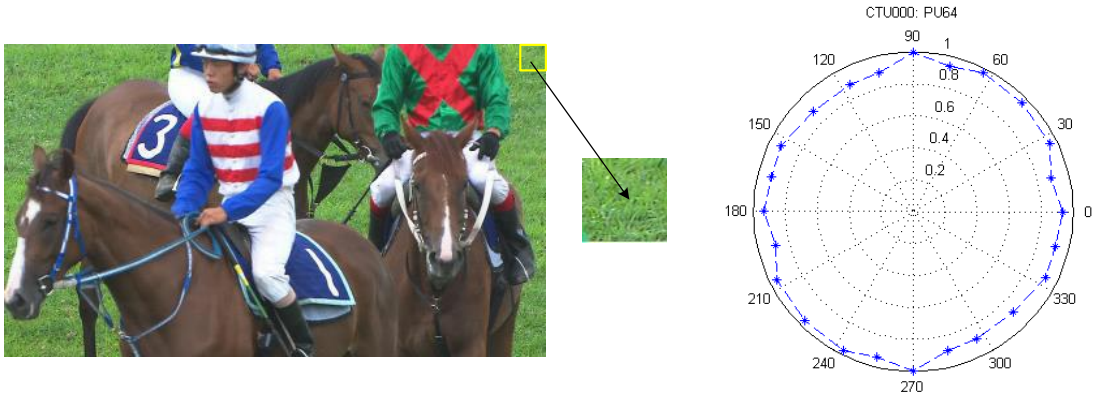


Figure 4.11. (a) 64x64 PU representing the unit hyperbola and its conjugate function, rotated $\boldsymbol{\pi/4}$. (b) 3D representation of same PU. (c) Polar diagram of the $\overline{\boldsymbol{MDV}(\boldsymbol{PU}, \boldsymbol{r_\iota})}$ of 64x64 PU. (d) Four 32x32 sub-PUs of previous 64x64 PU. (e) Polar diagrams of the $\overline{\boldsymbol{MDV}(\boldsymbol{PU}, \boldsymbol{r_\iota})}$ of four 32x32 sub-PUs.

The polar diagram of $\overline{\mathrm{MDV}(\mathrm{PU},\mathrm{r}_1)}$ is plotted in Figure 4.12(c), showing two minimums in the r_0 and r_6 directions, which matches with the $\pi/4$ and $3\pi/4$ orientations, respectively. This proves that our metric is able to detect the dominant texture orientation properly, even when the image has more than one dominant direction. However, in terms of gradient magnitude, those minimums are only 30% lower than the variance of the other orientations, confirming that the image has no clearly dominant gradient.

Figure 4.12(d) illustrates the four 32x32 sub-PUs obtained by the partitioning of the previous 64x64 PU, and their respective polar diagrams for each $\overline{\mathrm{MDV}(\mathrm{PU},\mathrm{r}_1)}$ are shown in Figure 4.12(e). Now, the polar diagram of each 32x32 sub-PU clearly shows only one minimum in the proper direction, and the magnitudes of those minimums are 80% lower than the maximum normalized MDV, confirming the sub-blocks have a strong dominant pattern.

Finally, a 64x64 PU from the *Racehorses* sequence belonging to the JCT-VC test sequences [JCTVC-L1100] has been selected, which is shown in Figure 4.13(a). This image corresponds to a portion of grass with complex texture and where no dominant edges can be clearly perceived. Consequently, the polar diagram depicted in Fig 13(b) does not show any minimum that point to the texture orientation, existing a difference between the maximum and minimum directional variance values of less than 10%. These results prove that the MDV metric can also detect blocks with no dominant orientation with high accuracy.



(a) (b)

Figure 4.12. (a) First frame of *Racehorses* sequence and the selected 64x64 PU. (b) Polar diagram of $\overline{\boldsymbol{MDV}(\boldsymbol{PU},\boldsymbol{r_i})}$ for selected 64x64 PU.

4.4.3.2 Simulation Results for the Mean Directional Variance Proposal

Table 4.5 shows the experimental results of the proposed algorithm using the test sequences compared to the HM 16.6 reference software. As can be observed, the MDV proposal achieves a *Time Saving* of over 30% compared with the reference model, and an average bit rate increase lower than 1% in terms of *BD-rate*, which is a very good balance between speed-up and rate penalty.

It should be noted that complexity reduction is quite similar for all the test sequences, in the range of 28% to 30%, which is a good indicator of the algorithm's efficiency for any type of resolutions and image complexities. Otherwise, in terms of bit rate penalty, there are two sequences of class B, *Kimono* and *ParkScene*, which have a negligible rate increase of 0.2%, proving the high texture estimation accuracy of the MDV algorithm. However, it can be observed that the class D sequences obtain the worst performance, with a 1.3% and 1.4% *BD-rate*, showing the MDV is not as efficient with low resolution images.

Table 4.5. Performance comparison for the proposed MDV algorithm to the HM16.6 reference software.

Classification	Sequence	Frames	T. Saving (%)	BD-Rate (%)
Class A (2560x1600)	Traffic	150	30.87	0.5
	PeopleOnStreet	150	30.34	0.9
Class B (1920x1080)	BasketballDrive	500	31.34	0.6
	BQTerrace	600	30.47	0.6
	Cactus	500	30.33	0.8
	Kimono	240	31.69	0.2
	ParkScene	240	30.98	0.2
Class C (832x480)	BasketballDrill	500	28.71	1.5
	BQMall	600	29.69	1.0
	PartyScene	500	29.16	1.0
	RaceHorses	300	30.34	0.9
Class D (416x240)	BasketballPass	500	30.39	1.3
	BQSquare	600	28.27	1.4
	BlowingBubbles	500	27.96	1.3
	RaceHorses	300	29.04	1.4
Class E (128x720)	FourPeople	600	30.53	0.8
	Johnny	600	30.90	0.8
	KristenAndSara	600	30.60	0.9
Class A			30.61	0.7
Class B			30.96	0.5
Class C			29.48	1.1
Class D			28.92	1.3
Class E			30.68	0.8
AVERAGE			**30.10**	**0.9**

With the aim of analysing the performance of the MDV algorithm with different QPs, Table 4.6 shows the simulation results individualized for each QP, and for one sequence of each class that has been found representative of the general performance behaviour. The $\Delta Bitrate$ and $\Delta PSNR$ metrics described in Section 4.4.2 are now used for the performance characterization, which shows the quality and bit rate penalty of the proposal compared with the HEVC reference model implementation.

As can be noted, the quality degradation is uniform for all QPs, and practically negligible for most of the sequences, being lower than 0.06 dB. However, the $\Delta Bitrate$ behaviour shows a strong dependence on the QP value. It is worth noting that for the smaller QP, QP22, unlike the state-of-the-art proposals, the MDV algorithm does have a bit rate penalty, but it achieves a slight bit rate reduction in the range from 0.11% to 0.62%. Thus, the MDV approach for the lowest QPs improves slightly upon the HM performance in terms of rate.

Nevertheless, it should be noted that a rate penalty is obtained for QP27 and higher QPs, increasing the bit rate by over 1% for some sequences. As was described in Section 4.3.4, that phenomenon is due to the *intra-prediction* using as reference samples, P_{ref}, the decoded pixels from the top and left PUs, and those pixels are distorted by the QP parameter. The quantification process in the HEVC coding can be considered as a low-pass filtering process, which modifies the original correlation between the pixels of the block and their neighbouring pixels.

Table 4.6. Comparison of the MDV algorithm's performance with the HM16.6 reference software

Classification	Sequence	QP	ΔBitrate (%)	ΔPSNR (dB)	T. Saving (%)
Class A (2560x1600)	PeopleOnStreet	22	-0.24	-0.04	29.82
		27	0.22	-0.03	30.15
		32	0.43	-0.03	30.36
		37	0.57	-0.03	31.04
Class B (1920x1080)	BasketballDrive	22	-0.62	-0.03	29.95
		27	0.03	-0.01	31.20
		32	0.22	-0.01	32.08
		37	0.28	-0.02	32.11
Class C (832x480)	BasketballDrill	22	-0.11	-0.04	28.60
		27	0.59	-0.04	28.34
		32	1.00	-0.04	28.68
		37	1.06	-0.04	29.22
Class D (416x240)	RaceHorses	22	-0.11	-0.06	27.62
		27	0.64	-0.05	28.25
		32	0.68	-0.04	27.81
		37	1.03	-0.04	32.37
Class E (128x720)	Johnny	22	-0.11	-0.02	30.62
		27	0.44	-0.01	30.87
		32	0.54	-0.02	31.04
		37	0.64	-0.02	31.05

None of the fast *intra-prediction* proposals described in Section 4.2 take into account the effect of the quantification on the reference samples, and the gradient estimation is only and exclusively carried out with the pixels belonging to the PU that is being evaluated. The next section reports the simulation results for the enhanced MDV approach proposed in this thesis, namely MDV-SW, proving that the use of the sliding window can significatively improve MDV's performance.

4.4.3.3 Simulation Results for the Mean Directional Variance with Sliding the Window Proposal

Table 4.7 shows the simulation results for the novel MDV-SW algorithm using the JCT-VC test sequences compared with the HM 16.6 reference software, using the same CTCs referred to above. As can be observed, the average *Time Saving* has been reduced slightly by 0.4% compared with the MDV proposal, achieving a 29.7% speed-up, which is still a notable complexity reduction. As was described in Section 4.3.5, such a *Time Saving* decrease is due to the algorithm computing the directional variance over an expanded window of $(N+1)x(N+1)$ pixels, so a slight complexity increase is required compared with the original MDV algorithm.

Regarding the bit rate penalty, it should be noted the average *BD-rate* is now 0.4%, which means a reduction of 0.5% compared with the previous MDV algorithm. The results prove that when using the reference samples in directional variance computation, the MDV-SW achieves a considerably better performance for the full range of QPs, reducing the penalty by over half for most of the test sequences, such as *BasketBallDrill*, *RaceHorses* (class C), *BasketBallPass* and *RaceHorses* (class D).

Table 4.7. Performance comparison for the proposed MDV-SW algorithm to the HM16.6 reference software

Classification	Sequence	Frames	T. Saving (%)	BD-Rate (%)
Class A (2560x1600)	Traffic	150	30.43	0.3
	PeopleOnStreet	150	29.84	0.4
Class B (1920x1080)	BasketballDrive	500	31.14	0.2
	BQTerrace	600	30.30	0.3
	Cactus	500	30.12	0.4
	Kimono	240	31.16	0.1
	ParkScene	240	30.53	0.1
Class C (832x480)	BasketballDrill	500	28.41	0.6
	BQMall	600	29.21	0.5
	PartyScene	500	28.64	0.7
	RaceHorses	300	29.65	0.3
Class D (416x240)	BasketballPass	500	29.11	0.6
	BQSquare	600	28.27	1.0
	BlowingBubbles	500	27.96	0.8
	RaceHorses	300	28.10	0.6
Class E (128x720)	FourPeople	600	30.33	0.3
	Johnny	600	30.85	0.4
	KristenAndSara	600	30.31	0.4
Class A			30.14	0.3
Class B			30.65	0.2
Class C			28.98	0.6
Class D			28.36	0.7
Class E			30.50	0.4
AVERAGE			**29.70**	**0.4**

Again, in order to compare the performance of the MDV-SW algorithm with different QPs, Table 4.8 shows the simulation results individualized for each QP, for the same test sequences reported above in Table 4.6, and the same $\Delta Bitrate$ and $\Delta PSNR$ metrics. As can

be seen, the quality degradation in terms of $\Delta PSNR$ is slightly reduced for all QPs compared with the MDV algorithm, showing a better performance for the new MDV approach.

Concerning the rate penalty, the behaviour for the lower QP, QP22, is the same as the one previously obtained with the MDV approach, confirming this proposal outperforms the HM implementation, with a slight rate reduction. Nevertheless, it should be highlighted that now the bit rate reduction is also obtained for higher QPs in most of the sequences, proving the sliding window approach can achieve good texture orientation detection when high distortion is applied. Only the "*BasketballDrill*" and "*Johnny*" sequences report a slight rate increase of nearly 0.2% for the highest QP of 32, which can be considered negligible.

Table 4.8. Comparison of the MDV-SW algorithm performance to HM16.6 reference software

Classification	Sequence	QP	ΔBitrate (%)	ΔPSNR (dB)	T. Saving (%)
Class A (2560x1600)	PeopleOnStreet	22	-0.24	-0.04	28.97
		27	-0.20	-0.03	29.90
		32	-0.15	-0.03	30.05
		37	-0.10	-0.02	30.42
Class B (1920x1080)	BasketballDrive	22	-0.62	-0.03	29.81
		27	-0.28	-0.01	30.88
		32	-0.14	-0.01	31.88
		37	-0.08	-0.01	31.95
Class C (832x480)	BasketballDrill	22	-0.11	-0.04	28.01
		27	-0.08	-0.03	28.06
		32	0.02	-0.03	28.51
		37	0.18	-0.02	29.03
Class D (416x240)	RaceHorses	22	-0.11	-0.06	27.62
		27	-0.15	-0.05	28.25
		32	-0.24	-0.04	27.81
		37	-0.08	-0.03	28.74
Class E (128x720)	Johnny	22	-0.11	-0.02	30.33
		27	0.00	-0.01	30.80
		32	0.05	-0.01	31.04
		37	0.20	-0.01	31.21

To summarize, the sliding window strategy introduced in the MDV algorithm makes it possible to improve the algorithm's performance in terms of quality degradation and bit rate penalty, with a slight *Time Saving* reduction of 0.4%. The MDV-SW approach proposed in this thesis achieves a notable complexity reduction of nearly 30%, and a negligible *BD-rate* of 0.4%.

4.4.3.4 Rate-Distortion Performance

With the aim of showing the quality performance of the proposed algorithm, the Rate-Distortion results are shown in Fig 14 to 17, individualized for each sequence class and depicting the best sequence performance and the worst one for each. These figures show the

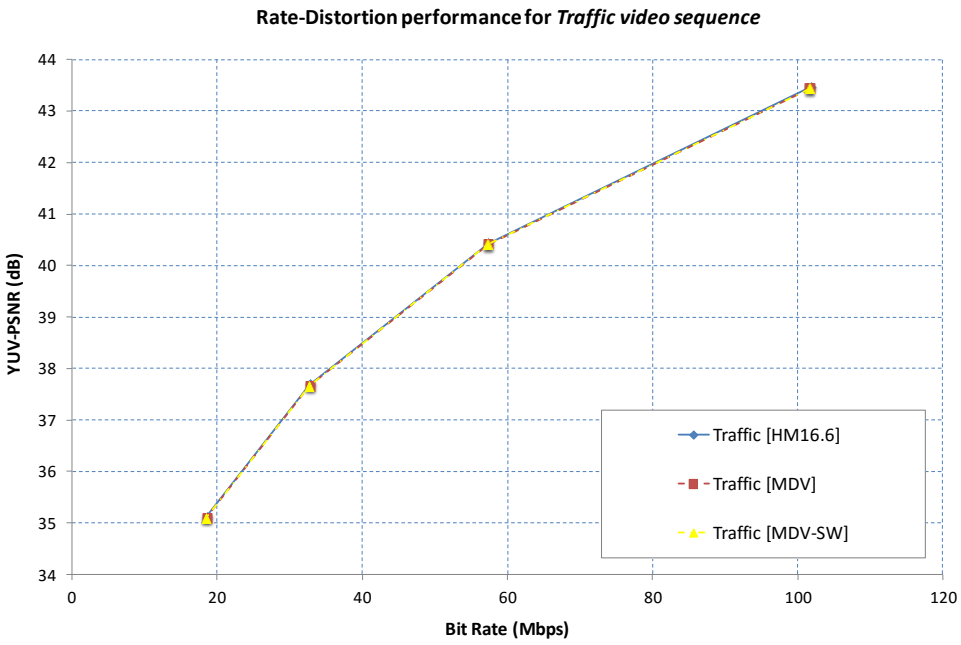
YUV_PSNR for the four QPs of the non-modified HM implementation, and of the two proposed fast mode decision algorithms, MDV and MDV-SW.

Figure 4.14(a) shows that for the best performance sequence of class A, *Traffic*, both algorithms achieve practically the same RD results as the HM reference software, with the three curves overlapped. Just a slight difference can be observed in Figure 4.14(b), for the worst sequence, *PeopleOnStreet.* However, for class B, the best and worst performance sequences, *Kimono* and *Cactus*, respectively, obtain the same RD results as the HM implementation for the whole range of QPs, as can be observed in Figure 4.15.

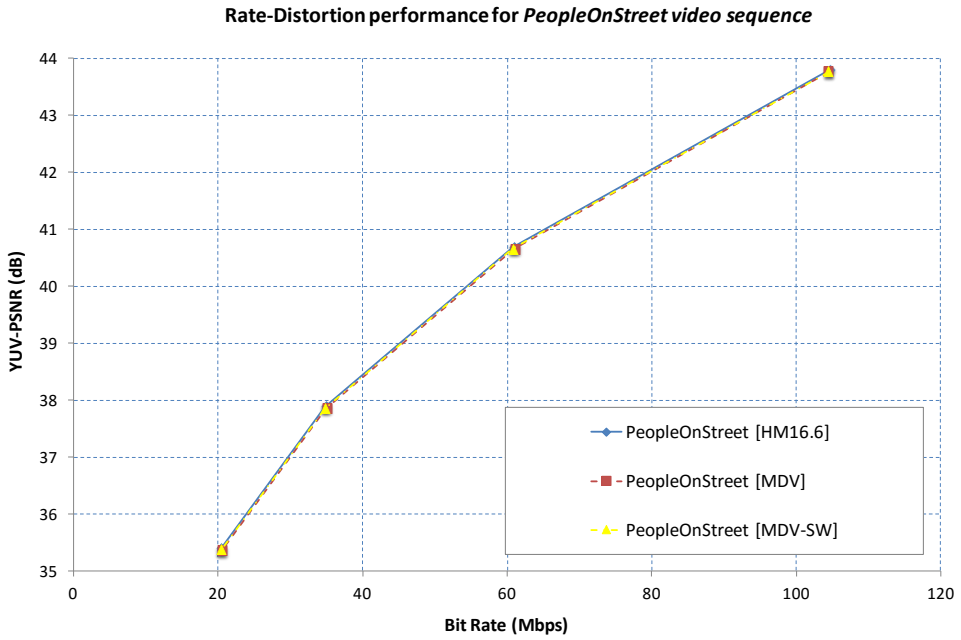
Class C and D show similar results in Figures 4.16 and 4.17, with a slightly lower performance for the MDV algorithm in the best performance sequences, *Racehorses* and *BasketballPass*, and for the worst performance sequences, *PartyScene* and *BQSquare*, a slightly lower RD performance is reported for both algorithms, MDV and MDV-SW.

Finally, the results for class F sequences in Figure 4.18 are quite similar to those obtained for class B, where the best and worst performance sequences, *FourPeople* and *KristenAndSara,* respectively, achieve practically the same RD results as the HM reference software.

As a final conclusion, no differences with the non-modified HM can be found for the MDV-SW algorithm in terms of RD performance for the whole set of sequences. However, the RD results for the MDV proposal reports a slightly lower performance than the anchor, for some of the worst sequences.

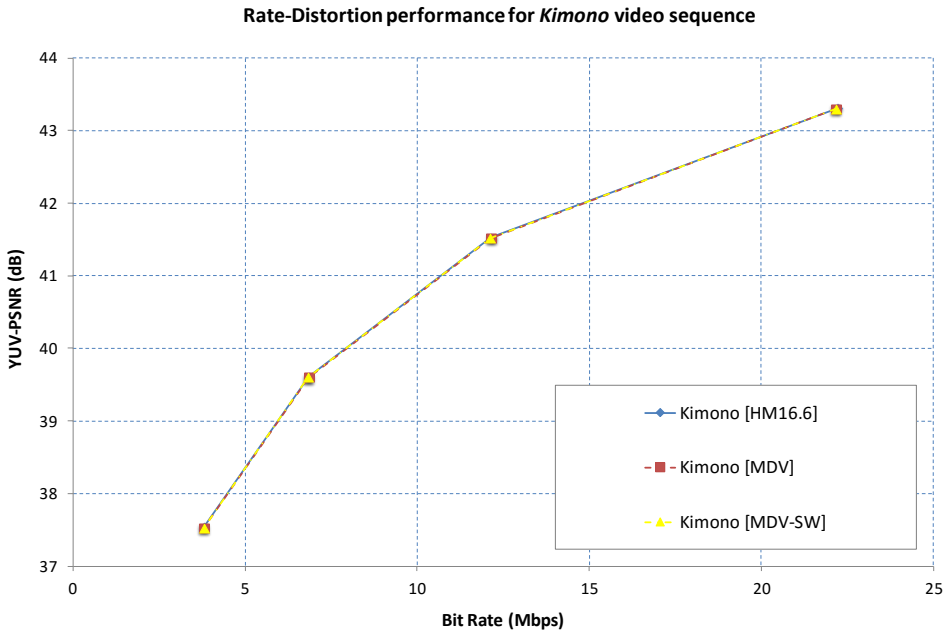


(a)

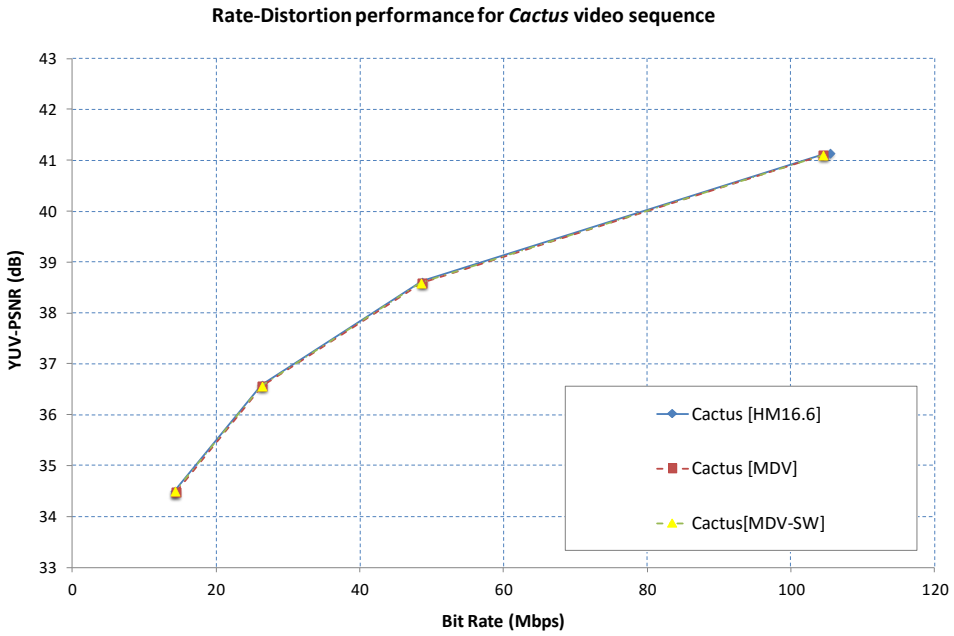


(b)

Figure 4.13. Rate-Distortion performance of Class A. (a) Best performance sequence. (b) Worst performance sequence

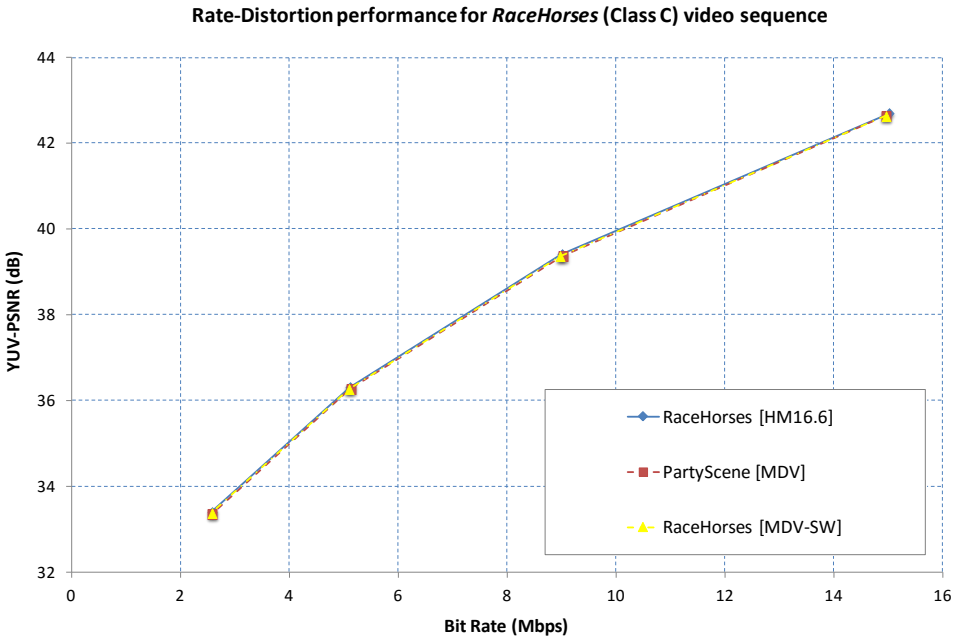


(a)

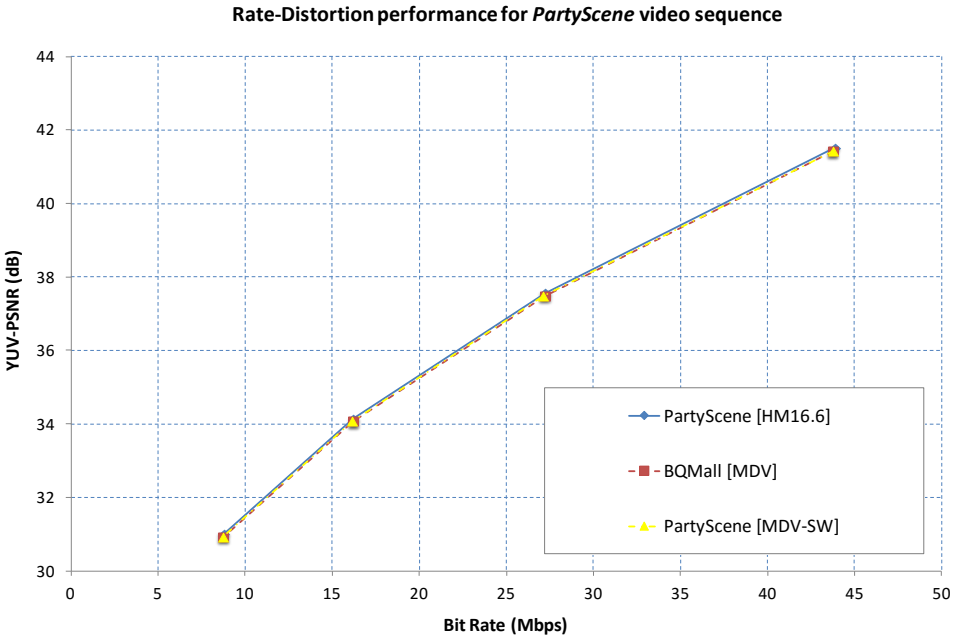


(b)

Figure 4.14. Rate-Distortion performance of Class B. (a) Best performance sequence. (b) Worst performance sequence

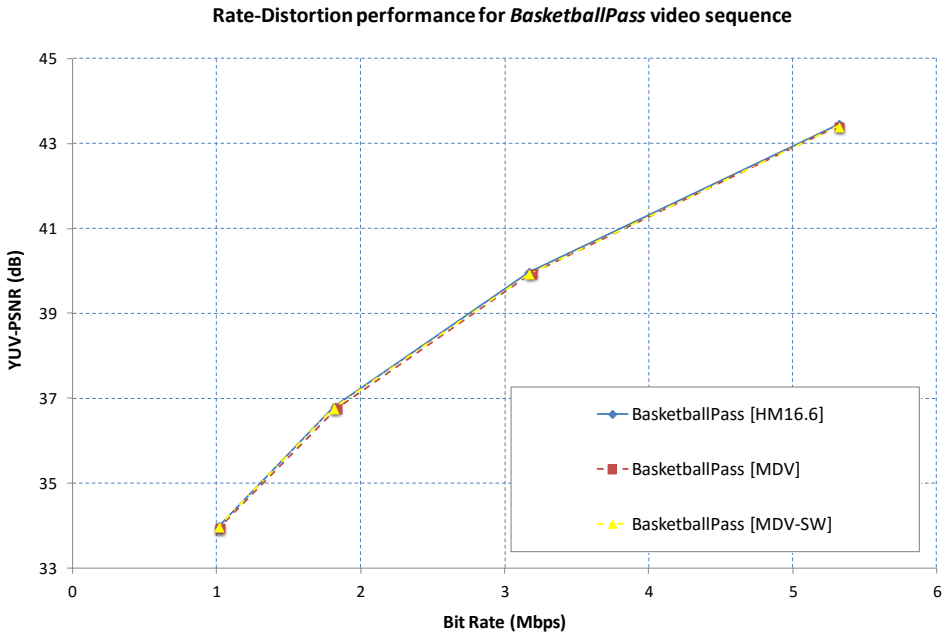


(a)

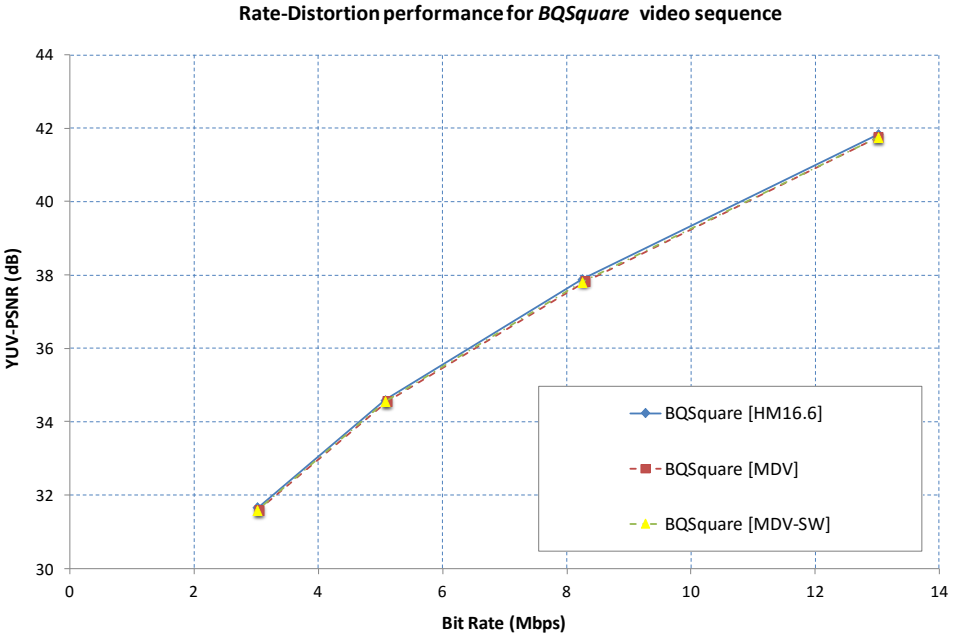


(b)

Figure 4.15. Rate-Distortion performance of Class C. (a) Best performance sequence. (b) Worst performance sequence

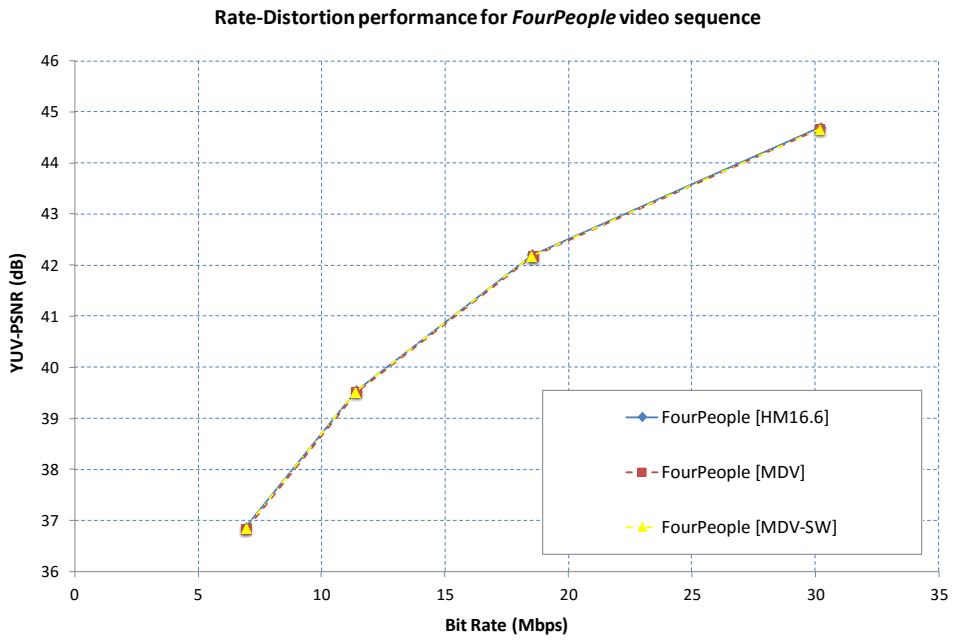


(a)



(b)

Figure 4.16. Rate-Distortion performance of Class D. (a) Best performance sequence. (b) Worst performance sequence



(a)

(b)

Figure 4.17. Rate-Distortion performance of Class E. (a) Best performance sequence. (b) Worst performance sequence

4.5 PERFORMANCE COMPARISON WITH DIFFERENT FAST INTRA-PREDICTION MODE DECISION ALGORITHMS IN HEVC

In Section 4.2, several fast *intra-prediction* mode decision proposals based on the reduction of the number of angular modes to be evaluated by the RDO stage were described. Those proposals achieved an average *Time Saving* in the range of 20% to 35%, with different rate penalties, from a moderate 0.5% bit rate increase to nearly a 2% penalty in terms of *BD-rate*.

In this section, a performance comparison between those proposals and the MDV and MDV-SW approaches proposed in this thesis is carried out. From the state-of-the-art, the five algorithms with the best performance have been selected, and their simulation results have been reported using the same JCT-VC test sequences, CTCs and performance metrics. These are the following:

- Two approaches using the Sobel operator, *Jiang et al.* [Jia12] and *Chen et al.* [Che13], denoted in this section as Sobel-1 and Sobel-2 respectively
- The FEOD algorithm proposed by da *Silva et al.* in [Sil12].
- The DEA approach proposed by *Yao et al.* in [Yao14].
- The DSAD algorithm proposed by *Yan et al.* in [Yan12].

The proposals' performances are evaluated using four of the metrics described in Section 4.4.2, and these are the quality degradation (ΔPSNR), bit rate penalty (*BD-rate*), computational complexity reduction (*Time Saving*), and the global penalty-complexity performance (ΔRate/ΔTime). Table 4.9 shows the algorithms' performances, including the MDV and MDV-SW proposals, ordered in terms of ΔRate/ΔTime, from worst performance [Yan12] to best performance (MDV-SW). As can be observed, some proposals achieve a high speed-up but also a high rate penalty, such as the DEA [Yao14] algorithm with a 36% time reduction and a *BD-rate* of 1.86%. On the other hand, other algorithms achieve a high rate performance but a low complexity reduction, such as Sobel-2 [Che13].

Table 4.9. Performance comparison between the proposed algorithm and the Related Works

QP	ΔPSNR (dB)	BD-rate (%)	Time Saving (%)	ΔRate/ΔTime
DSAD [Yan12]	NA	1.30	23.5	**5.5**
DEA [Yao14]	-0.007	1.86	36.2	**5.1**
FEOD [Sil12]	-0.023	0.90	20.0	**4.5**
Sobel-1 [Jia12]	-0.040	0.74	20.0	**3.5**
Proposed MDV	-0.040	0.90	30.1	**1.9**
Sobel-2 [Che13]	-0.038	0.53	28.4	**1.8**
Proposed MDV-SW	-0.030	0.40	29.7	**1.3**

In general, all the proposals have low quality degradation with PSNR reduction under 0.05dB, compared with the HEVC reference software. In terms of rate penalty, the DSAD and DEA algorithms obtain the worst results with a bit rate increase of over 1%, which can be considered as significant for some applications. The FEOD algorithm improves upon the DSAD and DEA proposals in terms of rate penalty with 0.9%, but achieves a lower speed-up, just a 20%. The three algorithms, DSAD, DEA and FEOD, obtain a moderate global performance in terms of ΔRate/ΔTime, ranging from 5.5 to 4.5.

The Sobel approaches, namely [Jia12] and [Che13], show a good trade-off between rate penalty (0.74% and 0.53%, respectively) and complexity reduction with a moderate 20% and 28%, improving upon the performance of the proposed MDV algorithm in terms of *BD-rate*, 0.9%. However, the MDV approach achieves a higher complexity reduction of 30.1%, and it exceeds the Sobel-1 algorithm in terms of ΔRate/ΔTime, with a ratio of 1.9 vs. 3.5, and it is just slightly below the Sobel-2 proposal, 1.9 compared to 1.8.

As can be observed, the MDV-SW algorithm outperforms all approaches in terms of rate penalty, reporting just a 0.4% *BD-rate*, while achieving one of the highest complexity reductions, nearly 30%. The MDV-SW approach also achieves the best ranking in terms of global performance with a ΔRate/ΔTime ratio of 1.3, followed by the Sobel-2 [Che13] algorithm with a ΔRate/ΔTime ratio of 1.8.

Consequently, the comparison results confirm that the MDV-SW algorithm proposed in this thesis outperforms the state-of-the art proposals, achieving a noteworthy complexity reduction with a negligible rate penalty, compared with the HEVC reference software.

CHAPTER 5

FAST INTRA-PREDICTION CODING ALGORITHM IN HEVC

Based on all the algorithms and techniques that have been proposed in this thesis, an overall fast *intra-prediction* coding algorithm for HEVC is proposed in this chapter. This approach combines the fast partitioning decision algorithm, based on the three node decision trees, which are trained using *Machine Learning* techniques, and a fast mode decision, based on a novel texture orientation detection algorithm, which computes the *Mean Directional Variance* along a set of *co-lines* with rational slopes using a sliding window over the PU. Both algorithms apply a similar approach, exploiting the strong correlation between several image features and the optimal CTU partitioning and prediction mode. The key point of the combined approach is that both algorithms compute the image features with low complexity, and the partition decision and the mode decision can also be taken with low complexity, using decision trees (if-else statements) and by selecting the minimum directional variance between a reduced set of directions. This approach can be implemented using any combination of nodes, obtaining a wide range of speed-ups, from 44% to 67%, and light bit rate penalties from 1.1% to 4.6%.

5.1 FAST INTRA-PREDICTION MODE DECISION APPROACH

In this section, the fast CTU partitioning decision approach proposed in Chapter 3 [Rui14][Rui15b], and the fast mode decision approach proposed in Chapter 4 [Rui15a] [Rui15c], are combined in an unified architecture forming a novel fast HEVC *intra-prediction* coding algorithm, denoted as *Fast Partitioning and Mode Decision* (FPMD). The optimal CTU partitioning decision is a tree of PUs, and the prediction mode decision for each PU of the tree is the key to achieve a high efficiency intra coding. The FPFM algorithm presented in this thesis achieves a considerable complexity reduction of over 67%, at the expense of a slight penalty in terms of rate increase, due to the sub-optimal partitioning and the sub-optimal mode decision given by the FPMD.

The architecture of the FPMD is depicted in Figure 5.1, which shows the new functionalities introduced by FPMD (shaded in grey). The FPDM workflow works at the CTU level, and as was described in Chapter 3, by evaluating the CTU attributes, which are carried out by the *decision trees* in the CTU classifier stage, the different PU sizes are selected. With the aim of organizing the set of PUs that split the CTU, a *Partition Map* is arranged by depth levels ($\forall\ d = 0,..,4$). For each d level, a list with the PUs $\left(PU_{d,k}\right)$ belonging to that depth level is recorded in a list. This *Partition Map* is the first key point of the FPMD algorithm because it avoids the exhaustive evaluation of the full range of PU sizes.

Intra-prediction is carried out by the evaluation of all PUs included in the *Partition Map* list, which are processed in depth level order (*Top-Bottom).* In order to reduce a misclassification error for every $PU_{d,k}$, the four sub-PUs ($PU_{d+1,4k+i}\ \forall\ i = 0,..3$) are also evaluated by the RMD, RDO and RQT stages. Consequently, five evaluations are always performed for each element of the *Partition Map*), following the fast partitioning algorithm described in Section 3.3.6.

Then, the MDV-SW is computed for each PU and respective sub-PUs, and a class C_i is assigned to each of them according with the minimum MDV. Each C_i includes a set of 3 or 4 angular candidate modes in addition to the two non-angular modes, DC and Planar, as was described in Section 4.3.3. Those modes are arranged in a *Mode List*, and they will be the only modes that will be evaluated by the RMD, instead of the 35 modes checked by the original RMD stage of the HM reference software, as was described in Section 2.4.2. This reduction in the number of evaluation modes is responsible for the time saving of the MDV-SW algorithm. As a result of the RMD process, a set of three candidate modes is selected to be checked by the RDO stage, and the mode with the lowest cost is selected to be further evaluated by the RQT, which selects the optimal TU size.

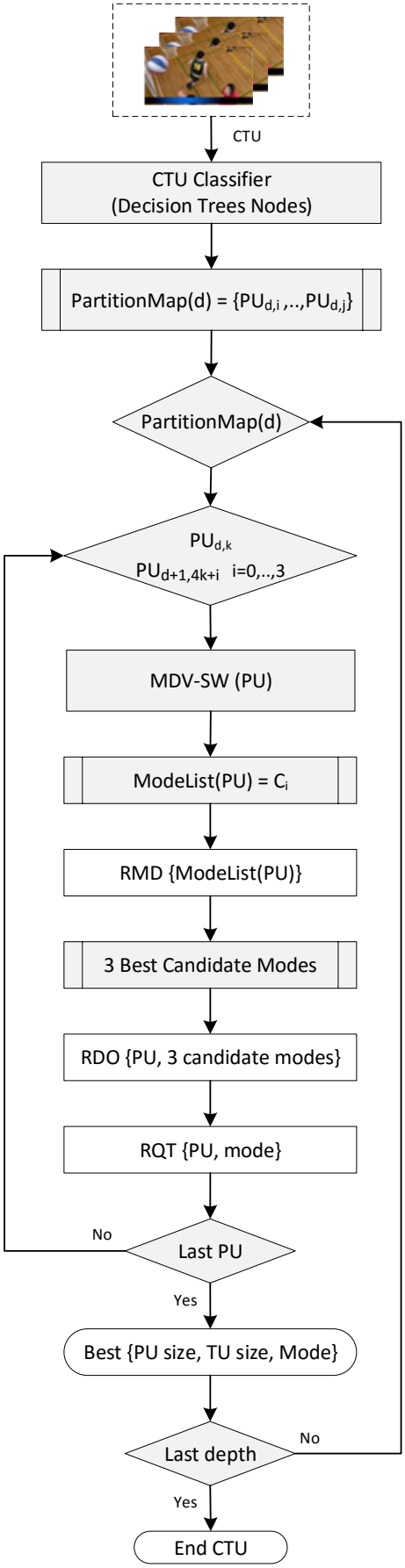


Figure 5.1. FPMD algorithm architecture

Finally, by comparing the cost of the $PU_{d,k}$ with the sum of the costs of the four sub-PUs, using the optimal modes and TU sizes selected by the RDO and RQT stages, the best option between $PU_{d,k}$ (*Non-Split*) and the four $PU_{d+1,4k+i}$ ($\forall\, i = 0,..3$) is selected.

5.2 PERFORMANCE EVALUATION

With the aim of evaluating the proposed algorithms, the fast CTU partitioning decision and the fast mode decision for *Intra-prediction* in HEVC were implemented in the HEVC HM 16.6 reference software [JCT-VC Reference Software]. The non-modified HM 16.6 reference software was used as anchor by using the same test sequences and encoding parameters. The simulations were run on the same architecture as was previously used for the individual evaluation of both algorithms proposed in this thesis, as was described in Section 3.4 and Section 4.4.

The simulations were independently run for the three node configurations of the fast CTU partitioning proposal, *Node64*, *Node64+Node32* and *Node64+Node32+Node16*, and the fast mode decision based on the MDV-SW proposal was always activated for all of them. Consequently, three different sets of results were obtained, denoted as *N64+MDV-SW*, *N64+N32+MDV-SW* and *N64+N32+N16+MDV-SW*.

5.2.1 Encoding Parameters

The experiments were conducted on the same set of 22 test sequences, and under the same test conditions as those described in Sections 3.4.1 and 4.4.1. Both the test sequences and test conditions are recommended by the JCT-VC [JCTVC-L1100]. Table 3.16 in Section 3.4.1 shows the test sequence features in terms of resolution, frame rate and number of frames, and Figure 3.23 in Section 3.4.1 depicts the first frame of each test sequence used in the simulations.

The use of the same set of encoding parameters allows a fair comparison between the results of the combined proposals and the individual results of each proposal. Therefore, for each individual node a thorough analysis of the combined fast partitioning and mode decision proposal performance can be obtained.

5.2.2 Metrics

The FPMD algorithm's performance was evaluated using the three main metrics used previously in the simulation of the individual fast partitioning decision and fast mode decision approaches, as were described in Section 3.4.2 and Section 4.4.2. These metrics are the computational complexity reduction, in terms of *Time Saving*, Rate-Distortion in terms of *BD-rate*, and the ratio between both metrics, denoted as the $\Delta Rate/\Delta Time$ ratio.

In order to obtain the increase in the *BD-rate* and the increase in the *Time Saving* introduced by the fast mode decision algorithm when it is combined in the architecture of each node, the ΔBD_rate and $\Delta T.Saving$ metrics are also used, following Equations (5.1) and (5.2), and where $N = 64, 64+32, 64+32+16$:

$$\Delta\text{BD\_rate} = BD\_rate_{Node_N} - BD\_rate_{Node_N+MDV-SW} \quad (5.1)$$

$$\Delta\text{T.Saving} = T.Saving_{Node_N} - T.Saving_{Node_N+MDV-SW} \quad (5.2)$$

5.2.3 Simulation Results

In this section, san analysis of the simulation results for the fast *intra-prediction* architecture, which combines both proposals, is shown. Table 5.1 reports the simulation results individualized for each decision trees in terms of *BD-rate* and *Time Saving*.

As can be noted, the average speed-up is now improved from the initial range, from 12% to 52%, obtained for the fast partitioning decision algorithm without the MDV-SW approach, (Tables 3.17 and Table 3.18 in Chapter 3), to the new range of 41% to 67% (Table 5.1). This enhancement is at the expense of a bit rate increase, where the *BD-rate* is practically doubled compared with the fast partitioning approach due to the error introduced by the fast decision mode. An initial conclusion drawn from this observation is that the penalty of the fast mode decision is not additive to the penalty due to the fast partitioning decision; instead, the error attributed to the wrong mode decision is amplified when the wrong PU size classification decision is given.

The first node, *N64+MDV-SW*, achieves an average *Time Saving* of around 40%, which is quite similar for the classes A, B, C and D, and only class E reaches a speed-up of over 45%. In terms of rate penalty, the results are quite uniform, around 1%, except for the *BQSquare* sequence, which obtains a *BD-rate* of over 1.6%. Regarding the second node, *N64+N32+MDV-SW*, the *Time Saving* increases by 15% with respect to the *N64+MDV-SW* implementation, achieving a notable average complexity reduction of 55.7%.

The best *Time Saving* performance is obtained by the *Johnny* sequence, of over 64%, however its *BD-rate* penalty is the highest of the whole set of test sequence, 4.8%, which is over the double of the average *BD-rate, 2.3%*.

Table 5.1. Performance results of combined fast intra-prediction proposals compared to HM16.6 reference software

Classification	Sequence	Frames	N64+MDV-SW		N64+N32+MDV-SW		N64+N32+N16+MDV-SW	
			T. Saving (%)	BD-rate (%)	T. Saving (%)	BD-rate (%)	T. Saving (%)	BD-rate (%)
Class A (2560x1600)	Traffic	150	41.09	0.9	55.98	2.0	69.55	2.0
	PeopleOnStreet	150	41.09	0.9	55.98	1.6	66.88	3.0
Class B (1920x1080)	BasketballDrive	500	42.56	0.9	59.31	3.2	70.55	8.0
	BQTerrace	600	42.86	0.7	56.70	1.4	67.41	6.5
	Cactus	500	42.00	1.0	55.67	2.1	69.66	3.2
	Kimono	240	44.08	1.1	57.92	6.7	72.23	7.6
	ParkScene	240	43.20	0.8	56.82	1.8	70.64	2.0
Class C (832x480)	BasketballDrill	500	39.41	1.0	54.09	1.5	67.11	2.5
	BQMall	600	40.46	1.2	54.65	2.1	66.29	4.7
	PartyScene	500	40.43	1.3	52.47	1.4	63.68	3.7
	RaceHorses	300	39.60	0.7	53.74	1.5	67.15	5.0
Class D (416x240)	BasketballPass	500	37.97	1.2	50.61	1.9	62.35	3.7
	BQSquare	600	39.01	1.6	50.30	1.8	57.62	5.5
	BlowingBubbles	500	38.36	1.3	48.65	1.4	60.02	3.4
	RaceHorses	300	39.20	1.1	49.34	1.5	61.69	3.5
Class E (128x720)	FourPeople	600	42.14	1.1	58.34	1.9	69.74	4.3
	Johnny	600	50.23	1.3	64.72	4.8	72.96	8.8
	KristenAndSara	600	43.73	1.4	63.63	2.8	71.74	4.6
Class A			41.78	1.0	55.98	1.8	68.24	2.5
Class B			42.94	0.9	57.30	3.0	70.14	5.5
Class C			39.97	1.1	53.75	1.6	66.09	4.0
Class D			38.64	1.3	49.74	1.7	60.46	4.0
Class E			45.48	1.3	62.33	3.2	71.51	5.9
AVERAGE			**41.67**	**1.1**	**55.71**	**2.3**	**67.34**	**4.6**

Finally, the results for the overall node implementation including the MDV-SW approach, *N64+N32+N16+MDV-SW*, show a considerable complexity reduction of 67%. Again the *Johnny* sequence obtains the best performance of over 73%. In terms of rate penalty, the class A test sequences obtain the best performance with a 2.5% *BD-rate*, which is nearly half the average rate increase of 4.6%. Three sequences, *BasketballDrive*, *Kimono* and *Johnny*, obtain a *BD-rate* of near 8%, doubling the average rate increase, which indicates that they are more sensitive to the mode decision errors when a wrong partitioning decision is given.

As was mentioned above, one of the most interesting conclusions that can be drawn from the experiments with the combined architectures concerns the impact of the fast mode decision on the performance of the fast partitioning decision for each decision tree node. Consequently, Table 5.2 summarizes the results in terms of the Δ*BD-ra*te and Δ*Time-Saving* for the three nodes.

In terms of complexity reduction, the MDV-SW algorithm obviously provides the highest reduction for the first node, *N64*, of 30%, which is practically the speed-up achieved when MDV-SW is applied alone. However, for the full node implementation, *N64+N32+N16*, the speed-up of the MDV-SW is just around 15%, because the fast mode partitioning has already reduced the complexity by 50%, so the 30% speed-up due to the fast mode decision just affects the remaining 50% of the computational burden. Therefore, the benefits of the fast mode decision are masked when high complexity reduction is achieved by the fast partitioning decision.

Table 5.2. Performance differences between the non-combined proposals and the combined proposals for each node

Classification	Sequence	N64 vs. N64+MDV-SW		N64+N32 vs. 64+N32+MDV-SW		N64+N32+N16 vs. N64+N32+N16+MDV-SW	
		Δ*T.Saving* (%)	Δ*BDrate* (%)	Δ*T.Saving* (%)	Δ*BDrate* (%)	Δ*T.Saving* (%)	Δ*BDrate* (%)
Class A (2560x1600)	Traffic	29.49	0.90	26.59	1.02	12.94	0.40
	PeopleOnStreet	30.27	0.99	27.95	1.04	16.79	1.88
Class B (1920x1080)	BasketballDrive	28.20	0.72	22.44	0.79	11.35	4.93
	BQTerrace	28.71	0.65	25.75	0.68	14.84	5.22
	Cactus	28.82	0.91	25.89	1.03	12.71	1.07
	Kimono	28.65	0.96	24.02	1.52	10.18	2.31
	ParkScene	28.68	0.78	26.08	0.91	11.89	0.37
Class C (832x480)	BasketballDrill	29.14	1.04	26.46	1.06	13.62	0.15
	BQMall	29.57	1.19	27.77	1.27	15.52	2.26
	PartyScene	30.24	1.33	30.39	1.34	18.19	1.61
	RaceHorses	29.35	0.73	27.34	0.78	14.01	3.31
Class D (416x240)	BasketballPass	31.98	1.19	27.08	1.24	16.69	1.59
	BQSquare	30.19	1.56	29.21	1.57	22.01	4.42
	BlowingBubbles	31.55	1.32	30.07	1.35	19.46	1.67
	RaceHorses	31.07	1.0	29.59	1.11	18.49	1.67
Class E (128x720)	FourPeople	29.66	1.08	24.45	1.13	13.10	2.36
	Johnny	23.42	1.03	19.18	1.39	9.96	4.39
	KristenAndSara	28.36	1.12	19.24	1.15	10.54	1.94
Class A		29.88	0.95	27.26	1.03	14.78	1.14
Class B		28.61	0.81	24.80	0.99	12.12	2.78
Class C		29.58	1.07	27.96	1.11	15.26	1.83
Class D		31.20	1.28	28.97	1.32	19.09	2.34
Class E		27.02	1.08	20.83	1.23	11.14	2.90
AVERAGE		**29.25**	**1.03**	**25.89**	**1.13**	**14.26**	**2.31**

The behaviour of the rate penalty is quite different from the *Time Saving*. Unexpectedly, for the first two nodes, *N64+MDV-SW* and *N64+N32+MDV-SW*, the rate increase due to the fast mode decision is practically the same, 1%. For both nodes, the *BD-rate* obtained for the MDV-SW alone, 0.4%, is practically doubled when it is computed jointly with the fast partitioning mode. This is despite the fact that the nodes' performance is remarkably different in *Time Saving* and rate penalty terms.

Nevertheless, in the overall implementation, *N64+N32+N16+MDV-SW*, the *BD-rate* increase due to MDV-SW is multiplied by 6 compared with MDV-SW alone, an increase of

2.3% with respect to the rate penalty of the same node without the MDV-SW approach. It should be noted that for the *BQTerrace* sequence the rate penalty increases by over 5%, but, on the other hand, the impact on the rate penalty for the *BasketballDrill* sequence is just 0.15%. This demonstrates that rate penalty sensitivity due to a wrong mode decision has a strong dependency on the image content, especially for the overall nodes implementation.

Table 5.3 shows the performance results for the non-combined proposals, *MDV-SW*, *N64*, *N64+N32* and *N64+N32+N16*, as described in Chapters 3 and 4, and also for the respective combined proposals, *N64+MDV-SW*, *N64+N32+MDV-SW* and *N64+N32+N16+MDV-SW*. All of them are given in terms of *Time Saving*, *BD-rate* and $Rate/\Delta Time$ ratio.

The approaches are sorted by $\Delta Rate/\Delta Time$ ratio, from the lowest, indicating the best performance, to the highest, indicating the worst balance in terms of rate penalty and complexity reduction. As can be observed, the best performance is for the *N64* alone implementation with a 0.8 ratio; however, the low speed-up of this approach, 12%, would not make it be worth using in most scenarios. The *MDV-SW* approach is the next approach in the ranking, with a ratio of 1.34, thanks to the negligible *BD-rate* penalty of 0.4%.

Table 5.3. Performance results of non-combined and combined fast partitioning decision and fast mode decision proposals, compared to HM16.6 reference software

Sequence	T. Saving (%)	BD-Rate (%)	ΔRate/ΔTime
N64	12.42	0.1	0.80
MDV-SW	29.70	0.4	1.34
N64+ MDV-SW	41.67	1.1	2.63
N64+N32	29.82	1.2	4.02
N64+N32+ MDV-SW	55.71	2.3	4.12
N64+N32+N16	53.08	2.2	4.14
N64+N32+N16+ MDV-SW	67.34	4.6	6.83

The next two combinations, *N64+MDV-SW* and *N64+N32*, report a similar rate increase of 1%. Nevertheless, the $\Delta Rate/\Delta Time$ ratio shows that *N64+MDV-SW* obtains a better performance than the *N64+N32* implementation, with a 2.63 ratio vs. 4.02. That is a remarkable result because it shows that the first node implementation including the MDV-SW algorithm outperforms the two node partitioning decision without the fast mode partitioning.

The next two architectures in the ranking, *N64+N32+MDV-SW* and *N64+N32+N16*, obtain practically the same results for all metrics, with a $\Delta Rate/\Delta Time$ ratio of 4.12 and 4.14 respectively. The conclusion is the same as above, that is, the first two node implementation including the fast mode decision obtains the same performance as the whole three node architecture. Finally, the worst ratio is obtained by the full node and MDV-SW combined

implementation with a 6.8 ratio, which achieves the highest complexity reduction, but also the highest rate penalty. This architecture can be used in scenarios where low complexity is required, and a moderate rate penalty is accepted.

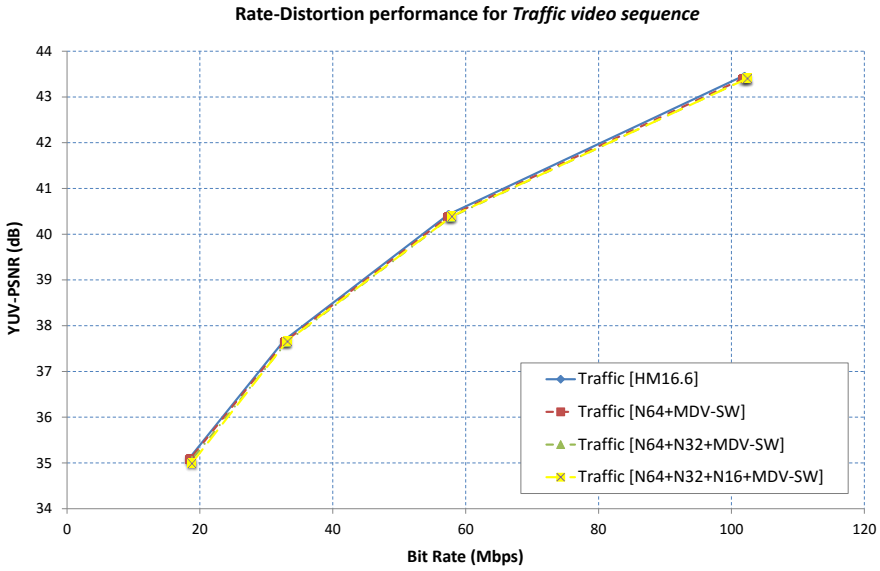
To summarize, the combination of the fast mode decision algorithm and the fast partitioning decision algorithm, using the hierarchical node implementation with different depth levels, proves to be a flexible and efficient architecture for *intra-prediction* coding in HEVC. This approach makes it possible to choose the optimal implementation, based on the design requirements, such as computational complexity and rate penalty, compared with the HEVC reference software.

5.2.4 Rate-Distortion Performance

In this section, the quality performance of the combined architecture of the proposed FPMD is shown, in Figures 5.2 to 5.6, in terms of Rate-Distortion, individualized for each sequence class. In order to show representative simulation results, the best sequence performance and the worst sequence performance are depicted for every sequence class. The figures show the YUV_PSNR for the four QPs of the non-modified HM reference software, denoted as HM16.6, and also for the three node implementations combined with the MDV-SW algorithm, denoted as *N64+MDV-SW*, *N64+N32+MDV-SW*, and *N64+N32+N16+MDV-SW*.

For class A, the *Traffic* sequence obtains the best performance, overlapping the HM16.6 reference software curve for the three node implementations and for all QPs, as is shown in Figure 5.2(a). The *PeopleOnStreet* sequence also obtains a good performance for high bit rates, but the performance for the overall node implementation, *N64+N32+N16+MDV-SW*, is slightly below the HM16.6 reference software performance for low bit rates. This phenomenon indicates that the partitioning and mode estimations for the FPMD approach are not as efficient when the image is highly distorted, high QPs. This is shown in Figure 5.2(b).

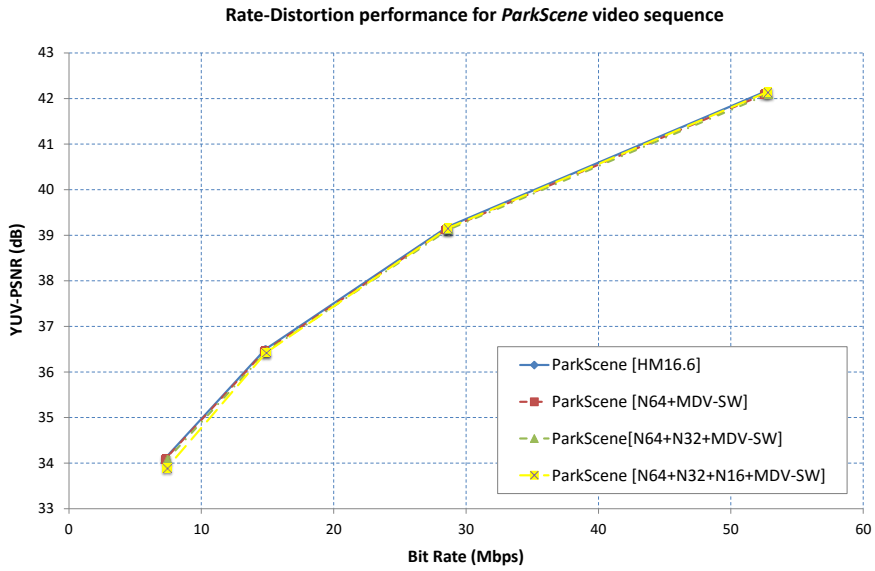
Figure 5.3(a) depicts the results of the *ParkScene* sequence, which achieve the best performance from class B sequences, showing a high quality performance for all QPs and node implementations, quite similar to the *Traffic* sequence of class A. However, the *Kimono* sequence achieves the worst performance of the class B sequences, showing a significant performance reduction for the *N64+N32+MDV-SW and N64+N32+N16+MDV-SW* architectures, which remains constant for all QPs, as shown in Figure 5.3(b).



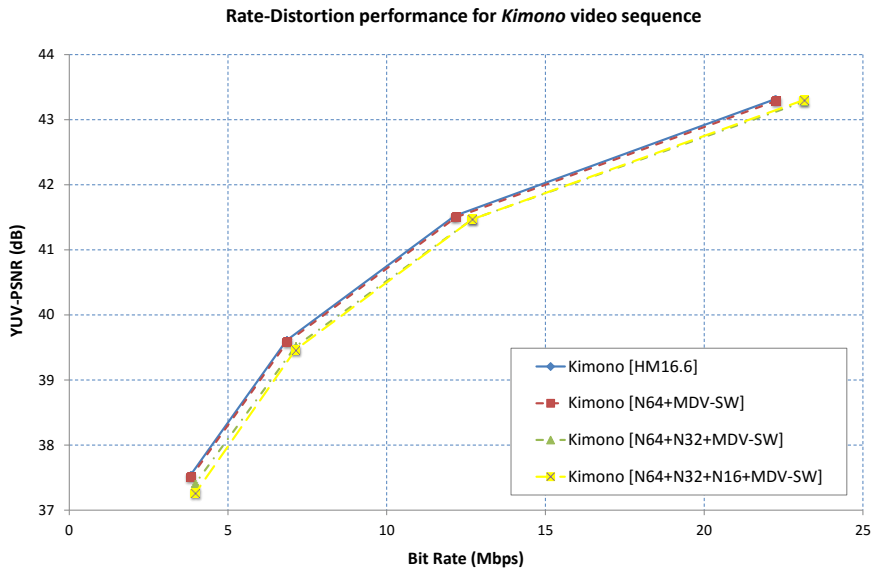
(a)

(b)

Figure 5.2. Rate-Distortion performance of Class A. (a) Best performance sequence. (b) Worst performance sequence

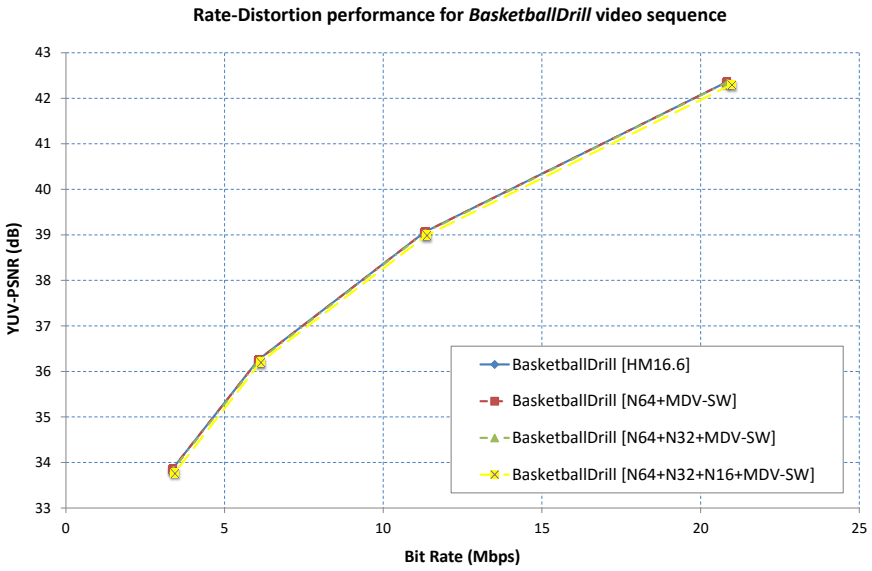


(a)

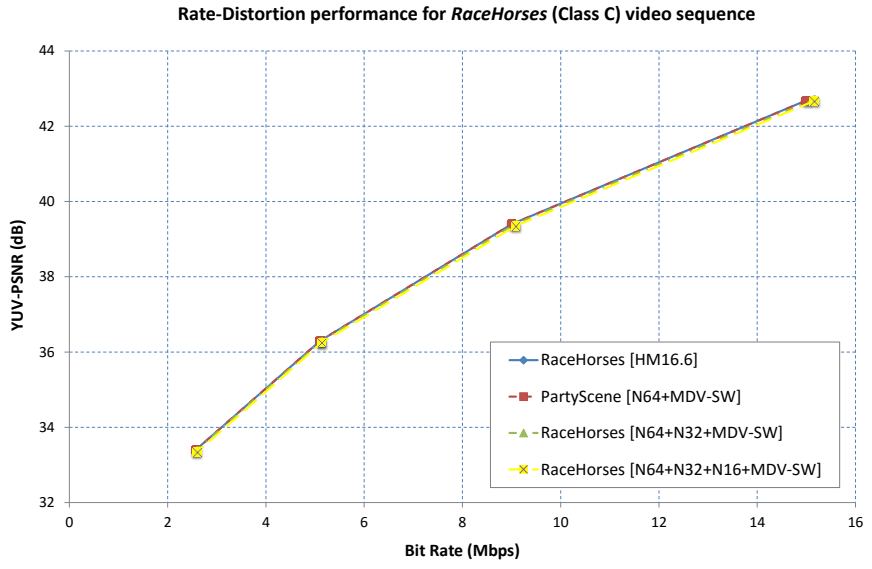


(b)

Figure 5.3. Rate-Distortion performance of Class B. (a) Best performance sequence. (b) Worst performance sequence

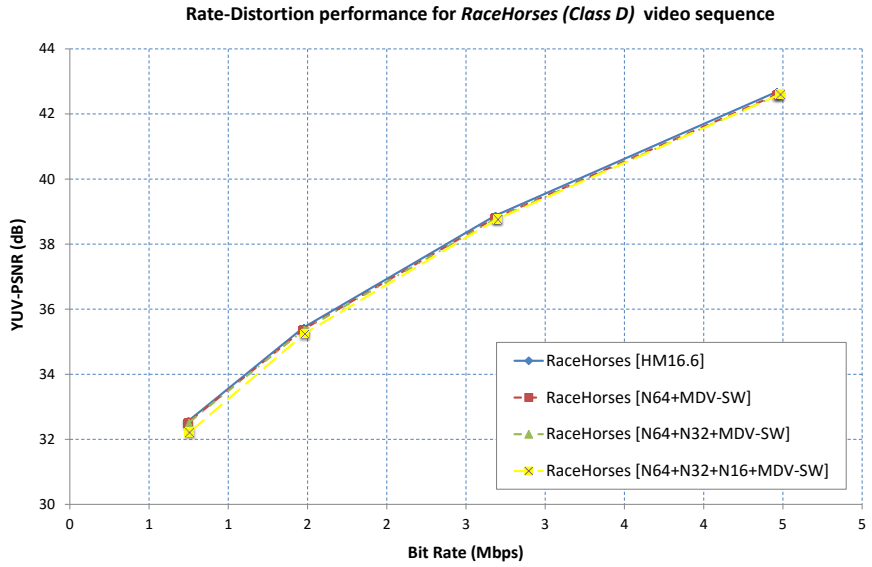


(a)

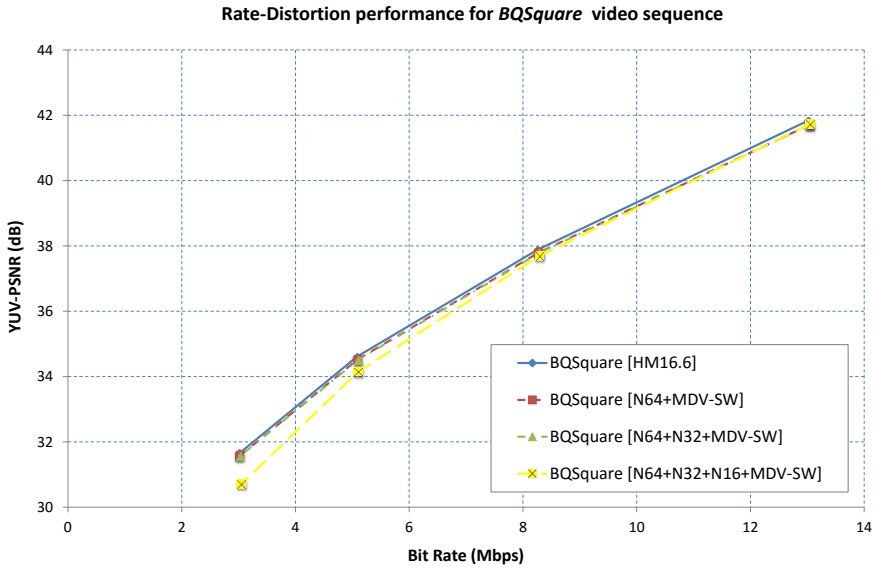


(b)

Figure 5.4. Rate-Distortion performance of Class C. (a) Best performance sequence. (b) Worst performance sequence

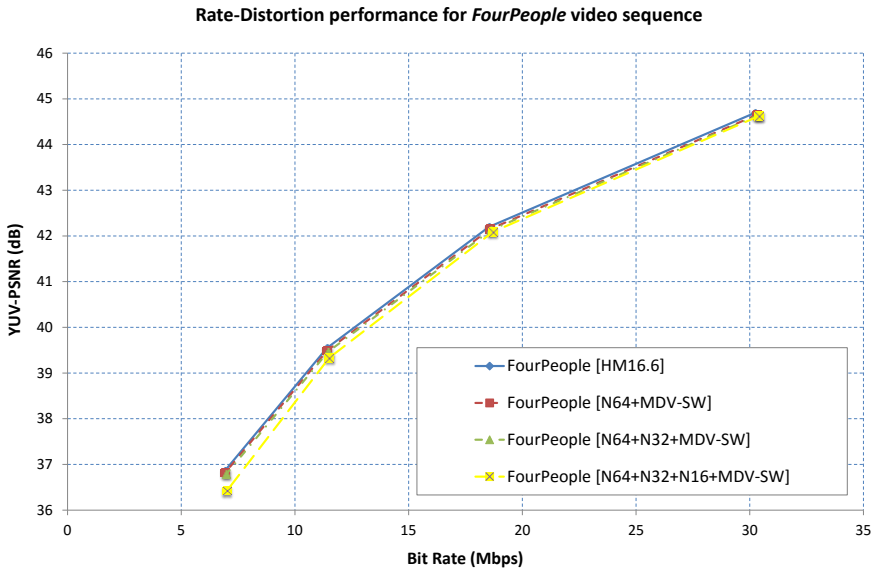


(a)

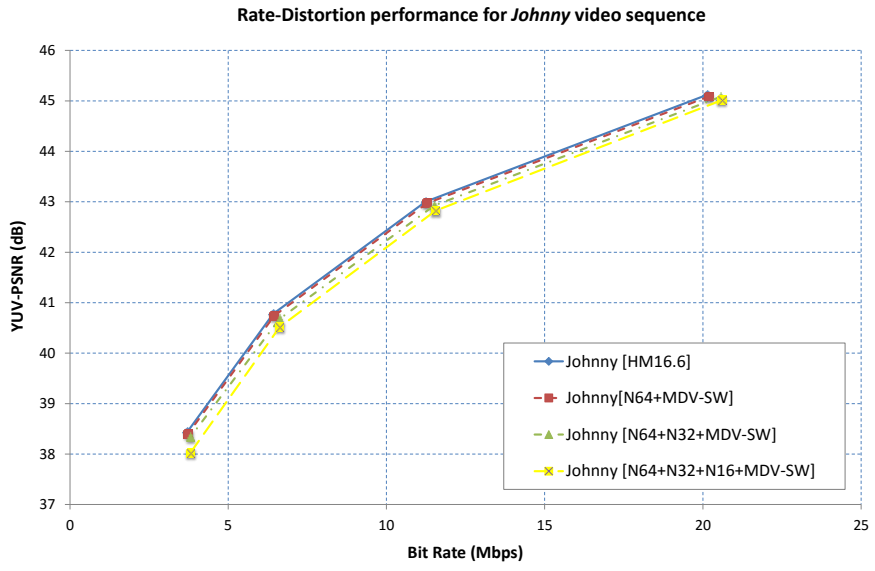


(b)

Figure 5.5. Rate-Distortion performance of Class D. (a) Best performance sequence. (b) Worst performance sequence



(a)



(b)

Figure 5.6. Rate-Distortion performance of Class E. (a) Best performance sequence. (b) Worst performance sequence

The *BasketballDrill* and *RaceHorses* sequences are the Class C sequences with the best and worst performance respectively. Both sequences obtain practically the same RD results as the HM reference software, overlapping the curves for the full range of bit rates and for the three node architectures, as is shown in Figure 5.4(a) and 5.4(b).

The class D sequences depicted in Figure 5.5 show the same pattern as that observed for class A. Their performance is extremely good for the *N64+MDV-SW and N64+N32+MDV-SW* architectures, but coding efficiency for node *N64+N32+N16+MDV-SW* gets progressively lower with the QP increasing (lower bit rates).

Lastly, the results for the best performance sequence of class E, namely *FourPeople,* is depicted in Figure 5.6(a), which shows the same pattern that the *BQSquare* sequence, a fairly good performance except for the higher QPs of node *N64+N32+N16+MDV-SW*. The *Johnny* sequence has the worst performance in class F, reporting a perceptible quality reduction for *N64+N32+N16+MDV-SW and N64+N32+N16+MDV-SW* compared with the HM implementation.

As a final conclusion, the combined approaches proposed in this thesis, for the *N64+MDV-SW* and *N64+N32+MDV-SW* nodes, achieve an extremely high performance for most sequences in terms of Rate-Distortion, equivalent to the HM16.6 reference software. On the contrary, the overall combined architecture, *N64+N32+N16+MDV-SW*, slightly reduces the PSNR for high QPs. Some sequences, such as *Kimono* and *Johnny*, show noticeable performance differences for the *N64+N32+MDV-SW* and *N64+N32+N16+MDV-SW* nodes, with the HM16.6 implementation for the full range of QPs.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter concludes the thesis. Firstly, the main conclusions which can be drawn from this thesis are presented. Then, based on these conclusions, new research lines are proposed which could be developed as future work. Finally, the list of the publications that have been derived from this thesis is given.

6.1 CONCLUSIONS

The major conclusions obtained in this thesis are summarized in the following lines:

1. *Regarding HEVC intra-prediction coding*. Nowadays, multimedia services are widely delivered through wireless and broadcast networks to a broad range of devices with very diverse computational capabilities. With the aim of achieving high compression efficiency, these services commonly use a long GOP coding structure, which applies an *inter-picture* prediction scheme, but *intra-prediction* coding is also used at least in one picture of each GOP. *Intra-prediction* is currently an active research topic due to the practical application of the intra-coding scheme in several fields, such as: professional media production, where high quality and random access to the individual pictures are required. The *intra-prediction* coding scheme is also used for the very low delay communications, in which intra-coding appears to be the only available solution; and for delivery through networks with transmission error, typically packet loss, because the non-temporal dependency between pictures avoids the temporal error propagation.

   Nowadays, the high efficiency of HEVC and its performance supremacy over previous video coding standards, such as H.264/AVC, is unquestionable. However, the extremely high computational burden demanded by the HEVC algorithms makes HEVC coding a challenge for developers and for the research community. In the last few years, wireless and portable devices have experienced an incredible evolution of their multimedia capabilities, increasing the quality with a very high spatial. The evolution of wireless networks has followed a similar trend, especially the cellular networks supporting 3G, 4G and the forthcoming 5G technologies, which allow huge bandwidths of tens, and soon hundreds of Mbps, which were unthinkable a few years ago. However, video communication using these new multimedia functionalities imposes a huge computational burden that becomes a severe problem for battery-powered portable devices.

   The novel approaches presented in this thesis prove that the non-conventional techniques used in the video coding field, such as *Machine Learning* and image processing tools, make it possible to achieve a considerable complexity reduction when they are efficiently applied, which can be regarded as a solution for the above mentioned scenarios under computational or consumption constraints.

The results reported in Chapter 5 show that the combination of both approaches obtain a wide range of speed-ups, from 30% to nearly 70%, in a trade-off with slight performance losses in the range of 0.4% to 4%, when it is compared to the HM reference software. This noteworthy complexity reduction for HEVC *intra-prediction* coding will encourage the quick introduction and development of the new generation of video coding standards in a new generation of multimedia devices such as smartphones, tablets and smart TVs.

2. *Regarding a fast partitioning decision for HEVC intra-prediction coding*

There is no doubt that the optimal CTU partitioning decision is the most computationally intensive process in *intra-prediction*, due to the high number of available PU sizes evaluated by the RDO stage. The proposal presented in this thesis addresses this problem by introducing a fast PU classifier which has been designed using Machine Learning models.

ML techniques are commonly used to solve big data problems where a big number, which can be tens or sometimes hundreds, of attributes are known. These techniques make it possible to carry out a high accuracy data classification or infer a behavior with high reliability. However, the application of ML techniques to the *intra-prediction* scenario, where initial features of the CTU are unavailable or cannot be gathered from other stages of the encoder, constitutes one of the most challenging aspects addressed in this thesis due to the critical balance that has to be reached between the number of attributes to be computed, and the cost of computing these attributes.

The approach presented in this thesis is based on decision trees whose rules were obtained by training a model using ML techniques supported by the WEKA tool. It is worth highlighting that the size of the training data set and the type of data set constituted the key elements in achieving a high accuracy classifier. Regarding the selection of the data set, this was assisted by the SI and TI metrics, which have proven to be very effective criteria for image complexity characterization. While attending to these parameters, the data type was selected by covering the full range of possible CU patterns, from very low textured images to high complexity images. As for the size of the data set, just 8 frames were used for the training stage, which is 0.081% of the total frames (9,780 frames belonging to the 22 test sequences) used for the validation test. The results obtained confirm that a high accuracy classifier can be achieved by training decision trees with a small data set, as long as a high quality data type selection is performed.

Regarding the attributes used by the classifier, it has been found that first order statistics such as variance and the mean of PUs are not sufficient to classify a PU as *Split* or *Not-Split* with high accuracy. However, by introducing as an attribute the variance of these statistics, the mean and variance, but computed over the four sub-PUs, the classifier accuracy is significantly increased. In addition, these statistics can be computed with a very low computational burden, thus the balance between the number and effectiveness of attributes and their cost is very profitable.

Results for the training and non-training sequences have demonstrated that a significant speed-up of over 52%, can be achieved for the CTU partitioning decision in *intra-prediction* coding, with only a slight bit rate increase of less than 2%, favoring real-time software and hardware implementation. Finally, a comparative study with the most prominent fast CTU partitioning decision algorithms proposed in the literature has been conducted. The results confirm that the approach presented in this thesis achieves better performance for the full node implementation, with a higher complexity reduction and a lower bit rate penalty.

3. *Regarding fast mode decision for HEVC intra-prediction coding*

In terms of computational burden, the optimal mode decision constitutes the second critical parameter that has to be selected for *intra-prediction*. In this thesis, the correlation between the optimal mode decision and the edge and texture orientation is exploited. Based on that observation, a novel texture orientation detection approach has been proposed which computes the MDV along a set of co-lines with rational slopes.

The first accomplishment reached in this proposal is the definition of the reduced number of orientations, just 12 directions, which can estimate a set of modes close to the optimal mode in-between the 33 angular modes of HEVC. A noteworthy feature of this approach is that these orientations with rational slopes are exclusively defined in integer positions of the discrete lattice Λ, thus no pixel interpolation is required, significatively reducing the computational complexity.

The second key point of this proposal has been the demonstration that the pixel correlation is maximal in the dominant texture orientation, and consequently the variance computed in that direction obtains the lowest value compared to the variance computed in other directions. This observation constitutes the basis of the proposed MDV algorithm, which presents several advantages such as the low

complexity computation of the variance, and a scalability implementation in co-segments, allowing the computation of the all PU sizes in just one pass.

The next achievement has been the improvement of the MDV algorithm by introducing the concept of *Sliding Window*. In this new approach, the directional variance computation for a NxN PU is expanded to a $(N+1)x(N+1)$ window, thus the neighbouring pixels used as reference samples for the construction of the predictor are also included in the calculation of the directional variance. MDV-SW makes it possible to reduce the bit rate penalty of the MDV proposal by half.

The results show that the proposed MDV-SW achieves a significant time saving of 30% with a negligible impact on the coding performance of just 0.4%, in terms of *BD-rate*. Finally, a comparison with the 5 most noteworthy fast mode decision algorithms proposed in the literature has been presented. The results show that the MDV-SW proposal obtains the best performance with the lowest $\Delta Rate/\Delta Time$ ratio.

4. *Regarding the combination of fast partitioning and mode decision approaches for HEVC intra-prediction coding*

Finally, both proposals presented in this thesis, the fast partitioning and fast mode decision, are combined, creating an overall fast *intra-prediction* algorithm, called FPMD, which achieves a very high overall performance. The key point of the combined approach is that both decisions are taken by computing low complexity algorithms, first using decision trees (*if-else* statements) for the partitioning decision, and then the optimal orientation is estimated by computing the MDV-SW in 12 directions.

The combined approach is a flexible solution because it can be implemented using any combination of nodes for the fast partitioning decision, including the fast mode decision, and thus better performance is achieved compared with the standalone partitioning approach. The simulation results for FPMD show a wide range of speed-ups, from 44% for the *N64+MDV-SW* to 67% for the overall implementation, namely *N64+N32+N16+MDV-SW*, at the expense of slight penalties in terms of *BD-rate*, from 1.1% to 4.6%, respectively.

6.2 FUTURE WORK

During the development of this thesis, several novel algorithms have been proposed for speeding up the *intra-prediction* coding in HEVC. Although high valued results have been reached in terms of complexity reduction and efficiency performance, there exist several issues that are still open to further research. The algorithms presented in this thesis can be considered as a starting point for the development of new approaches in *intra-prediction* coding. These should consider new factors that have not been addressed in this work, such as the use of perceptual models for the selection of the optimal partitioning and angular modes, or the adaptation of the algorithm to future hardware architectures.

As a final thought, a series of new research lines are proposed for future work:

- ***Use of directional information as attributes for the CTU partitioning decision***. As was depicted in Figure 1 of Chapter 5, the partitioning decision and the mode decision are carried out in an independent and sequential manner. However, it is unquestionable that there exists a strong correlation between texture directionality and the optimal partitioning size. A highly textured block, or a block presenting more than one dominant gradient, is a good candidate block to be split. On the other hand, a block with a strong gradient will achieve a high efficiency if it is not split. Consequently, as future research we propose the use of texture directionality as an attribute for the decision trees. Features such as the number of dominant gradients, the gradient energy, the number of co-lines with similar gradient or the correlation between the dominant gradient of the PU and the respective four sub-PUs, could be considered for the accuracy improvement of the partitioning decision algorithms proposed in this thesis.

- ***Detection of optimal non-angular mode.*** The statistical analysis presented in [Rui15d] reported that more than 50% of the intra modes are selected as non-angular modes. However, the DC and Planar modes are always included as candidates for the RDM evaluation in addition to 3 or 4 angular modes. An ML technique could be applied again to design a directionality classifier, using attributes derived from the MDV algorithm. The decision trees could determine if a PU has only to consider the non-directional modes or the directional modes as candidates for a further evaluation, making it possible to speed up the mode decision in the *intra-prediction* coding.

- ***Adaptive selection of the number of candidate modes***. Both the HM reference software as well as the proposed MDV-SW algorithm use a fixed number of candidates to be evaluated for the RD. However, sometimes the cost function of the RMD reports big differences between the candidate modes, which indicates that the

probability that a candidate mode with a high cost will be chosen as optimal mode by the RDO is very low. Otherwise, if exist more than 3 candidates with a very similar cost, a better match can be obtained dynamically increasing the number of candidate modes that are selected. Consequently, an adaptive selection of the number of candidates to be evaluated by the RDO could reduce the complexity in some cases and improve the mode hit in other cases.

- ***Training of combined partitioning and mode decision***. The results obtained for the combined approach show that there exists a mutual inference between the optimal partitioning decision and the mode decision. Therefore, an optimized architecture could be achieved by iterating both running processes, including the MDW-SW algorithm in the decision tree training stage using ML, and including the fast partitioning decision approach in the MDV-SW optimization stage. This new approach will make it possible to achieve optimized decision rules and optimized tree topologies in the partitioning decision stage, but it will also enable the improvement of the orientation decision in terms of the class definition, and the number of candidate modes selected.

- ***Use of perceptual models for the cost function of RMD and RDO***. The selection of the optimal CTU partitioning and prediction mode decision is formulated as an *optimization* problem for the distortion, subject to bit rate constraints. RMD and RDO address this optimization by using a similar so-called *Lagrangian Rate-Distortion* cost function, as was described in Section 2.4.1. As distortion parameter is commonly used objective metrics such as SSE, SAD or MAD, which provide a good estimation in terms of the differences between the original picture and the decoded picture. However, from the perceptual point of view, the partition and mode selection obtained by using that cost functions might not be optimal. Figures 3.13 and 3.14 in Chapter 3 show that sometimes the CTU partitioning is not totally coherent with the object shape perceived in the scene. It is well known that the human visual perception system has different sensitivity to different spatial orientations, and consequently the distortion could be pondered depending on the orientation of angular intra mode selected. Based on these arguments, a new cost function for the RMD and RDO could be formulated, with the aim of achieving a better perceptual coding performance.

- ***Algorithm optimization for parallel architectures***. As mentioned above, the proposed algorithm's workflow is implemented in a sequential order, which is not suitable for current and future multicore and heterogeneous processor architectures, but it is the architecture used by the HM reference model. A new architecture model could be studied for the algorithms proposed in this thesis, which exploits

parallelization at different granularity levels. For instance, the attributes for the decision trees and the MDV-SW computation could be simultaneously obtained; even the global mean and global variance could share most of the computation process with the directional variance, applying a correct arrangement. The variance computation along the co-lines could also be executed in a parallel way. A new parallel architecture that takes advantage of these approaches could improve upon the complexity reduction reported in this thesis.

- ***Algorithms extrapolation for inter-prediction coding***. Considering the fact that CTU partitioning is also a critical process in *inter-prediction*, the *Machine Learning* methodology proposed for the intra coding could be applied to solve the partitioning decision in the *inter-prediction* coding. Regarding the directional orientation information derived from the MDV-SW, it could be used for speeding up the *motion-estimation* stage, by searching the PUs with similar texture orientation patterns, making it possible to carry out an initial selection of the PU candidates for the *motion-estimation* search.

- ***Algorithm validation for the Range Extensions profiles of HEVC.*** In July 2014, the ISO/MPEG and ITU-T committees released the second version of the HEVC standard, which introduces a new set of profiles which include support for higher pixel-depths of 12-bits and 16-bits, and 4:2:2 and 4:4:4 *chroma* sub-subsampling. These profiles are especially useful for high quality edition and post-production tasks, where *intra-prediction* is the preferred coding scheme. For this reason, the adaptation and evaluation of the algorithms proposed in this thesis to this new set of extended profiles could be of great interest for this field of application.

6.3 PUBLICATIONS

The publications already derived from this thesis are as follows:

6.3.1 International Journals

- *Damián Ruiz, Gerardo Fernández-Escribano, Velibor Adzic, Hari Kalva, José Luis Martínez, Pedro Cuenca, "Fast CU partitioning algorithm for HEVC intra coding using Data Mining." Accepted to be published in the* ***Multimedia Tools and Application*** *Journal during 2016. ISSN: 1573-7721, ONLINE (doi: 10.1007/s11042-015-3014-6.), November 2016. Impact: 1,346. Ranking: 32/102 (2nd quarter) in Computer Science, Theory and Methods category.*

In this paper, a low complexity partitioning decision algorithm for *intra-prediction* is developed. The algorithm is based on decision trees with three hierarchical nodes covering the full range of CU sizes. The decision trees are based on a binary classifier whose rules are obtained by an off-line training stage based on Data Mining models. The low complexity content texture properties of CUs are used as attributes for the classifiers, such as the variance and the mean of the CU and four sub-CUs. The algorithm's performance has been individually evaluated for each node, reporting a considerable encoding time saving for the overall architecture implementation. These results have been described in Section 3.4.3.1 and 3.4.3.2.

– *Damián Ruiz, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, "Fast intra mode decision algorithm based on texture orientation detection in HEVC." Accepted to be published in the* ***Journal of Signal Processing: Image Communication****. In process of major revisions. Impact factor: 1.462 (JRC 2014). Ranking: 106/249 (2nd quarter) in Engineering, Electrical & Electronic category.*

  In this paper, an optimization to the MDV algorithm is introduced by using a *Sliding Window* approach (MDV-SW). The work shows how by including the neighbouring pixels used as reference samples for the construction of the intra predictors in the MDV computation, the bit rate penalty can be significantly reduced. The MDV-SW algorithm is described in Section 4.3.5, and the results are presented in Section 4.4.3.3.

6.3.2 Submitted Journals

– *Damián Ruiz, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, "A novel texture orientation detection algorithm using Mean Directional Variance along co-lines." Submitted to the* ***Journal of Computational and Applied Mathematics****. Impact factor: 1.266 (JRC 2014). Ranking: 59/257 (1st quarter) in Mathematics, Applied category.*

  This work describes a new texture orientation algorithm based on the *Mean Directional Variance* along co-lines of the image. The novelty of this approach is the use of integer lattice sub-sample theory for the selection of a reduced set of co-lines with rational slopes, which do not require pixel interpolation. This approach proves that the orientation with minimum variance has a strong correlation with the texture orientation. This algorithm is described in Section 4.3.2. The simulation results for natural and synthetic images show a high accuracy gradient detection, even when

more than one gradient is present in the image. The results are described in Section 4.4.3.1.

- *Damián Ruiz, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, "Fast intra-prediction coding in HEVC using Machine Learning." Will be submitted to the* ***Journal of IEEE Transactions on Circuits and Systems for Video Technology****. Impact factor: 2.615 (JRC 2014). Ranking: 30/249 (1st quarter) in Engineering, Electrical & Electronic category.*

  This work describes the new *intra-prediction* architecture that combines both approaches, the fast partitioning decision using ML techniques and the fast mode decision using the MDV-SW algorithm. This approach proves that a high complexity reduction of nearly 70% can be achieved by introducing an efficient algorithm for the CTU partitioning decision and the prediction mode decision. This algorithm is described in Section 5.1. The simulation results for the full set of JCT-VC test sequences are reported and compared with the *state-of-the-art* proposals. The results have been described in Section 5.2.3.

6.3.3 International Conferences

- *Ruiz, D.; Adzic, V.; Fernandez-Escribano, G.; Kalva, H.; Martinez, J.L.; Cuenca, P., "Fast partitioning algorithm for HEVC Intra frame coding using machine learning," in Image Processing (****ICIP****), 2014 IEEE International Conference on , vol., no., pp.4112-4116, 27-30 Oct. 2014. (doi: 10.1109/ICIP.2014.7025835).*

  In this paper, a novel fast classifier of CU sizes is presented for HEVC *intra-prediction coding* based on Machine Learning techniques. This approach proposes two nodes architecture for the 64 and 32 CU sizes partitioning decision, which use binary decision trees trained by ML models. The performance of this approach shows that a significant complexity reduction of the CU partitioning decision can be achieved with a negligible performance reduction, compared to the HM reference software. The results have been described in Sections 3.4.3.1 and 3.4.3.2.

- *Damian Ruiz Coll, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, "Dominant gradient detection using Mean Directional Variance for intra-picture prediction in HEVC." International Conference Computational and Mathematical Methods in Science and Engineering (****CMMSE****), Rota, Cadiz, Spain, July 6-10, 2015.*

This paper describes the use of the MDV algorithm for the speeding up of the mode decision in HEVC *intra-prediction* coding. This work describes how the MDV algorithm can be efficiently applied to the small coding units in HEVC. The matching between the rational slopes orientations used by the MDV algorithm and the HEVC angular intra prediction directions is described in Section 4.3.3. This approach makes it possible to reduce the number of intra modes to be evaluated by the RMD from 35 to 5 or 6. The performance results are given in Section 4.4.3.2, showing that a significant complexity reduction can be achieved at the expense of a slight bit rate increase.

– *Damian Ruiz, Maria Pantoja, Gerardo Fernádez-Escribano, "CTB Directional Gradient Detection Using 2D-DWT for Intra-Frame Prediction in HEVC."* ***GPU Technology Conference****, San Jose, California, USA, March 17-20, 2015.*

  In this paper, a fast mode decision for HEVC *intra-prediction* is presented based on the observation that the optimal mode is strongly correlated with the dominant gradient orientation of the CU. This approach proposes gradient detection by computing the 2D Directional Wavelet Transform (DWT) along the same rational slopes proposed in the MDV algorithm, described in Section 4.3.3. The simulation results show that the parallel implementation of 2D-DWT on GPU architecture makes it possible to achieve a high speed-up of the *intra-prediction* coding compared to the HM reference software.

6.3.4 National Conferences

– *Damian Ruiz Coll, Velibor Adzic, Gerardo Fernández-Escribano, Hari Kalva, "Análisis estadístico del particionado de bloques en la predicción Intra-Frame de HEVC."* ***XXV Jornadas de paralelismo****, 17-19 Septiembre 2014, Valladolid, Spain.*

  In this paper, an exhaustive statistical analysis of the CTU partitioning in HEVC *intra-prediction* is presented. The study covers the following four essentials: the computational cost required for the RDO stage analysed by PU size, the statistical distribution of each PU size, the correlation between the optimal PU size and the QP parameter, and the PU size dependence on content features. The conclusions drawn from this work constitute the essential basis for the fast partitioning algorithm presented in Chapter 3, including the top-bottom scheme and the threshold adaptation to the QP parameter.

- *Damian Ruiz Coll, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, "Análisis estadístico de la eficiencia de predicción direccional intra-cuadro en HEVC."* ***XXVI Jornadas de paralelismo****, Cordoba, Spain 23-25 Septiembre, 2015.*

  This paper presents an in-depth statistical analysis of the prediction modes in HEVC *intra-prediction* coding. In order to know the performance of the RMD stage implemented in the HM reference software, this work covers the following main areas: the statistical distribution models of the angular modes, the hit rate of the RMD compared with an exhaustive evaluation of the 35 modes by RDO, and the RMD error rate in terms of angular mode distance. Some of the most relevant conclusions derived from this study are the high probability of the non-angular modes, the low probability of the FPM, and the strong dependence of the optimal mode on the QP parameter. These conclusions have comprised the mainstay for the development of the fast mode decision algorithm presented in Chapter 4.

6.3.5 Other International Conferences

- *Garcia, R.; Ruiz-Coll, D.; Kalva, H.; Fernandez-Escribano, G., "HEVC decision optimization for low bandwidth in video conferencing applications in mobile environments," in Multimedia and Expo Workshops (****ICMEW****), 2013 IEEE International Conference on, vol., no., pp.1-6, 15-19 July 2013. (doi: 10.1109/ICMEW.2013.6618270).*

  In this paper, an exhaustive complexity and performance evaluation of HEVC is carried out for the full range of CU, PU and TU sizes. The results report that different levels of complexity reduction can be obtained by limiting the search for the optimal partitioning at the three unit levels, at the expense of quality losses. By using *Machine Learning* techniques, this work proposes a predictive model for complexity reduction and quality degradation which is achieved depending on a set of parameters such as the sequence resolution, the target bit rate, and the limits to the CU, PU and TU sizes.

- *Saurin Parikhi, Damian Ruiz, Hari Kalva, Gerardo Fernandez, "Content Dependent Intra Mode Selection for Medical Image Compression Using HEVC." 2016 IEEE International Conference on Consumer Electronics (****ICCE****). Las Vegas, Nevada, USA, 9-12 January, 2016.*

This paper exploits the structure and texture detail in several modalities of medical images, to reduce the number of CU sizes and directional modes to be evaluated by the RMD and RDO in HEVC *intra-prediction* coding. This approach uses that correlation to infer a Limited Coding Tree Depth (LCTD) and a set of Angular Directional Prediction (ADP) modes. The simulation results show that the combination of LCTD with an optimal selection of the ADP makes it possible to halve the encoding time and also the file size of the medical images.

REFERENCES

[Bai11] Jing Bai; Huaji Zhou, "Edge detection approach based on directionlet transform," in Multimedia Technology (ICMT), 2011 International Conference on , vol., no., pp.3512-3515, 26-28 July 2011.

[Bam92] Bamberger, R.H.; Smith, M.J.T., "A filter bank for the directional decomposition of images: theory and design," in Signal Processing, IEEE Transactions on , vol.40, no.4, pp.882-893, Apr 1992.

[Bha97] Vasudev Bhaskaran and Konstantinos Konstantinides. 1997. Image and Video Compression Standards: Algorithms and Architectures (2nd ed.). Kluwer Academic Publishers, Norwell, MA, USA.

[Bos11] F. Bossen, On Software Complexity, HM 6.0 Reference Software, JCTVC-G757, Geneva, Switzerland, Nov. 2011.

[Boy15] Boyce, J.M.; Ye, Y.; Chen, J.; Ramasubramonian, A.K., "Overview of SHVC: Scalable Extensions of the High Efficiency Video Coding (HEVC) Standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.PP, no.99, pp.1-1, 2015.

[Bri06] Britanak, V., et al.: Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations. Academic Press, New York (2006).

[Can99] E. J. Cand‘es and D. L. Donoho, “Curvelets - a surprisingly effective nonadaptive representation for objects with edges,” in Curve and Surface Fitting, A. Cohen, C. Rabut, and L. L. Schumaker, Eds. Saint-Malo:Vanderbilt University Press, 1999.

[Car10] Carrillo, P.; Tao Pin; Kalva, H., "Low complexity H.264 video encoder design using machine learning techniques," Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on , vol., no., pp.461,462, 9-13 Jan. 2010.

[Cas96] Castleman K. R., 1996. Digital Image Processing. Englewood Cliffs, NJ: Prentice-Hall.

[Cen15] Y.-F. Cen et al., A fast CU depth decision mechanism for HEVC, Inf Process. Lett. (2015).

[Cha79] Tony F. Chan, Gene H. Golub, and Randall J. LeVequ. “Updating Formulae and a Pairwise Algorithm for Computing Sample Variances”. Technical Report. Stanford University, Stanford, CA, USA, 1979.

[Che13] Gaoxing Chen; Zhenyu Pei; Lei Sun; Zhenyu Liu; Ikenaga, T., "Fast intra prediction for HEVC based on pixel gradient statistics and mode refinement," Signal and Information Processing (ChinaSIP), IEEE China Summit & International Conference on, pp. 514-517, 6-10 Jul. 2013.

[Che14] Keji Chen; Yizhou Duan; Jun Sun; Zongming Guo, "Towards efficient wavefront parallel encoding of HEVC: Parallelism analysis and improvement," in Multimedia Signal Processing (MMSP), 2014 IEEE 16th International Workshop on , vol., no., pp.1-6, 22-24 Sept. 2014.

[Chi12] Chih-Ming Fu; Alshina, E.; Alshin, A.; Yu-Wen Huang; Ching-Yeh Chen; Chia-Yang Tsai; Chih-Wei Hsu; Shaw-Min Lei; Jeong-Hoon Park; Woo-Jin Han, "Sample Adaptive Offset in the HEVC Standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1755-1764, Dec. 2012.

[Cho13] Seunghyun Cho; Munchurl Kim, "Fast CU Splitting and Pruning for Suboptimal CU Partitioning in HEVC Intra Coding," Circuits and Systems for Video Technology, IEEE Transactions on , vol.23, no.9, pp.1555,1564, Sept. 2013.

[Con98] J.H. Conway and N.J.A. Sloane. Sphere packing, lattices and groups. Springer-Verlag, 1998.

[Cor15] Correa, G.; Assuncao, P.A.; Volcan Agostini, L.; da Silva Cruz, L.A., "Fast HEVC Encoding Decisions Using Data Mining," Circuits and Systems for Video Technology, IEEE Transactions on , vol.25, no.4, pp.660,673, April 2015.

[Do05] M. N. Do and M. Vetterli, “The contourlet transform: An efficient directional multiresolution image representation,” IEEE Trans. Image Process. vol. 14, no. 12, pp. 2091-2106, Dec. 2005.

[Duf09] F. Dufaux, G. J. Sullivan, and T. Ebrahimi, “The JPEG XR image coding standard [Standards in a Nutshell]” IEEE Signal Processing Magazine, Vol. 26, Nov. 2009.

[Efr83] Efron, B., & Gong, G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. American Statistician, 37, 36–48.

[Fay96] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R, Editors (1996), Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA

[Fer08a] Gerardo Fernández-Escribano, Rashid Jillani, Christopher Holder, Hari Kalva, Jose Luis Martinez Martinez, and Pedro Cuenca, “Video encoding and transcoding using machine learning,” Multimedia Data Mining: held in conjunction with the ACM SIGKDD 2008 (MDM '08), 9th International Workshop on, pp. 53-62, Las Vegas, Nevada, 24-27 Aug. 2008.

[Fer08b] Gerardo Fernández-Escribano, Hari Kalva, Pedro Cuenca, Luis Orozco-Barbosa, Antonio Garrido, “A Fast MB Mode Decision Algorithm for MPEG-2 to H.264 P-Frame Transcoding,” Circuits and Systems for Video Technology, IEEE Trans on, vol. 18, no. 2, pp. 172–185, Feb. 2008.

[Fer10] Fernandez-Escribano, G.; Kalva, H.; Martinez, J.L.; Cuenca, P.; Orozco-Barbosa, L.; Garrido, A., "An MPEG-2 to H.264 Video Transcoder in the Baseline Profile," Circuits and Systems for Video Technology, IEEE Transactions on , vol.20, no.5, pp.763,768, May 2010.

[Fly15] Flynn, D.; Marpe, D.; Naccari, M.; Nguyen, T.; Rosewarne, C.; Sharman, K.; Sole, J.; Xu, J., "Overview of the Range Extensions for the HEVC Standard: Tools, Profiles and Performance," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.PP, no.99, pp.1-1, 2015.

[Gab46] D. Gabor, “Theory of Communication,” J. Institute of Electrical Engineers, 93, 1946, 429–457.

[Gup13] S. Gupta and S. G. Mazumdar, “Sobel Edge Detection Algorithm,” vol. 2, no. 2, pp. 1578–1583, 2013.

[Han98] Hand, D. J. (1998), “Data Mining: Statistics and More?”, The American Statistician, May (52:2), 112-118.

[Han10] Han, W.-J.; Min, J.; Kim, I.-K.; Alshina, E.; Alshin, A.; Lee, T.; Chen, J.; Seregin, V.; Lee, S.; Hong, Y. M.; Cheon, M.-S.; Shlyakhov, N.; McCann, K.; Davies, T.; Park, J.-H.,"Improved Video Compression Efficiency Through Flexible Unit Representation and Corresponding Extension of Coding Tools," IEEE Transactions on, vol.20, no.12, pp.1709-1720, Dec. 2010.

[Han12] Philippe Hanhart ; Martin Rerabek ; Francesca De Simone ; Touradj Ebrahimi; Subjective quality evaluation of the upcoming HEVC video compression standard . Proc. SPIE 8499, Applications of Digital Image Processing XXXV, 84990V (October 15, 2012).

[Hon13] Honghai Yu; Winkler, S., "Image complexity and spatial information," Quality of Multimedia Experience (QoMEX), 2013 Fifth International Workshop on , vol., no., pp.12,17, 3-5 July 2013.

[Hua13] Han Huang; Yao Zhao; Chunyu Lin; Huihui Bai, "Fast bottom-up pruning for HEVC intraframe coding," Visual Communications and Image Processing (VCIP), 2013 , vol., no., pp.1,5, 17-20 Nov. 2013.

[Hul07] J.V. Hulse, T.M. Khoshgoftaar, and A. Napolitano, “Experimental Perspectives on Learning from Imbalanced Data,” Proc. 24th Int’l Conf. Machine Learning, pp. 935-942, 2007.

[ISO-11172] ISO/IEC 11172-2. – Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1988.

[ISO-13818-2] ISO/IEC 13818-2. – Information technology — Generic coding of moving pictures and associated audio information — Part 2: Video, 1996.

[ISO-14496-2] ISO/IEC 14496-2. – Information technology — Coding of audio-visual objects — Part 2: Visual, 1999.

[ISO-14496-10] ISO/IEC 14496-10. – Information technology — Coding of audio-visual objects, 2003.

[ISO-23008-2] ISO/IEC DIS 23008-2. Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding, 2013.

[ITU-T H.261] ITU-T Recommendation H.261, Video codec for audiovisual services at px64 kbits/s, 1988.

[ITU-T H.262] ITU-T Recommendation H.262, Information technology - Generic coding of moving pictures and associated audio information: Video, 1996.

[ITU-T H.263] ITU-T Recommendation H.263, Video coding for low bit rate communication, Version 2, 1998.

[ITU-T H.264] ITU-T Recommendation H.264, Advanced video coding for generic audiovisual services, 2003.

[ITU-T H.265] ITU-T Recommendation H.265, High efficiency video coding, 2013.

[ITU-T T. T.81] ITU-T Rec. T.81, "Information technologies – Digital compression and coding of continuous-tone still images – requirements and guidelines", 1992.

[ITU-T T.800] ITU-T Rec. T.800, "JPEG2000 Image Coding System: Core Coding System" (JPEG2000 Part 1), 2000.

[ITU-T T.832] ITU-T Rec. T.832, "Information technology – JPEG XR image coding system – Image coding specification", 2012.

[ITU-T P.910] ITU-T Recommendation P.910, "Subjective Video Quality Assessment Methods for Multimedia Applications," International Telecommunication Union, Geneva (1999).

[Jap00] N. Japkowicz, editor, Proceedings of the AAAI'2000. Workshopon Learning from Imbalanced Data Sets,. AAAI Tech Report WS-00-05.

[Jay10] Jayachandra, D.; Makur, A., "Directional Variance: A measure to find the directionality in a given image segment," Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on , vol., no., pp.1551,1554, May 30 2010-June 2 2010.

[JCTVC- D122] C.-M. Fu, C.-Y. Chen, Y.-W. Huang, and S. Lei, "CE8 Subset 3: Picture Quadtree Adaptive Offset", document JCTVC-D122, Jan. 2011.

[JCTVC-H0012] Gary Sullivan, Koohyar Minoo, "Objective quality metric and alternative methods for measuring coding efficiency", document JCTVC-H0012, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), 8th Meeting: San Jose, CA, USA, 1 – 10 February 2012.

[JCTVC-K1002] Il-Koo Kim, Ken McCann, Kazuo Sugimoto, Benjamin Bross, Woo-Jin Han, "HM9: High Efficiency Video Coding (HEVC) Test Model 9 Encoder Description", document JCTVC-K1002, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), 11th Meeting: Shanghai, CN, 10–19 October 2012.

[JCTVC-L1100] F. Bossen, "Common Test Conditions and Software Reference Configurations," document JCTVC-L1100, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), 12th Meeting: Geneve, CH 14 – 23 Jan. 2013.

[JCT-VC Reference Software] Joint Collaborative Team on Video Coding Reference Software, version HM 16.6 [Online]. Available: https://hevc.hhi.fraunhofer.de/

[Jia12] Wei Jiang; Ma, Hanjie; Yaowu Chen, "Gradient based fast mode decision algorithm for intra prediction in HEVC," Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on, pp.1836-1840, 21-23 Apr. 2012.

[Jil09] Jillani, R.; Kalva, H., "Low complexity intra MB encoding in AVC/H.264," Consumer Electronics, IEEE Transactions on , vol.55, no.1, pp.277,285, February 2009.

[Kan11] S. Kanumuri, T. K. Tan, and F. Bossen, Enhancements to Intra Coding, JCTVC-D235, Daegu, Korea, Jan. 2011.

[Kha13] Khan, Muhammad Usman Karim; Shafique, Muhammad; Henkel, Jorg, "An adaptive complexity reduction scheme with fast prediction unit decision for HEVC intra encoding," Image Processing (ICIP), 2013 20th IEEE International Conference on , vol., no., pp.1578,1582, 15-18 Sept. 2013.

[Kim13] Younhee Kim, DongSan Jun, Soon-heung Jung, Jin Soo Choi, and Jinwoong Kim, "A Fast Intra-Prediction Method in HEVC Using Rate-Distortion Estimation Based on Hadamard Transform," ETRI Journal, vol. 35, no. 2, pp. 270-280, Apr. 2013.

[Kor10] J. Korhonen and J. You, “Improving objective video quality assessment with content analysis,” in Proc. VPQM, Scottsdale, AZ, USA, Jan. 2010.

[Lai12] Lainema, J.; Bossen, F.; Woo-Jin Han; Junghye Min; Ugur, K., "Intra Coding of the HEVC Standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1792-1801, Dec. 2012

[Lei08] Zhang Lei; Makur, A., "Enumeration of Downsampling Lattices in Two-Dimensional Multirate Systems," Signal Processing, IEEE Transactions on , vol.56, no.1, pp.414,418, Jan. 2008.

[Li11] B. Li, G. J. Sullivan, and J. Xu, “Comparison of Compression Performance of HEVC Working Draft 4 with AVC High Profile,” Doc. JCTVC-G399, Nov. 2011

[Lia11] Liang Zhao; Li Zhang; Ma, Siwei; Debin Zhao, "Fast mode decision algorithm for intra prediction in HEVC," Visual Communications and Image Processing (VCIP), 2011 IEEE , vol., no., pp.1,4, 6-9 Nov. 2011.

[Loh11] Loh, W.-Y. (2011), Classification and regression trees. WIREs Data Mining Knowl Discov, 1: 14–23.

[Lok10] Kar Seng Loke, "Wedgelets-based automatic object contour detection," in Natural Computation (ICNC), 2010 Sixth International Conference on , vol.7, no., pp.3664-3668, 10-12 Aug. 2010.

[Lon13] Long-Sheng Chen; Jui-Yu Lin, "A study on review manipulation classification using decision tree," Service Systems and Service Management (ICSSSM), 2013 10th International Conference on, vol., no., pp.680,685, 17-19 July 2013.

[Mal89] S. G. Mallat, “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation,” IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-11, 7, July 1989, 674–693.

[Mar03] Marpe, D.; Schwarz, H.; Wiegand, T., "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.13, no.7, pp.620-636, July 2003.

[Mar04] J. M. Martínez, “MPEG-7 Overview (version 10),” MPEG Document, ISO/IEC JTC1/SC29/WG11 N6828, Palma de Mallorca, October 2004.

[Mar05] Marpe, D.; Wiegand, T.; Gordon, S., "H.264/MPEG4-AVC fidelity range extensions: tools, profiles, performance, and application areas," in Image Processing, 2005. ICIP 2005. IEEE International Conference on , vol.1, no., pp.I-593-6, 11-14 Sept. 2005.

[Mar08] Martinez, J.L.; Fernandez-Escribano, G.; Kalva, H.; Weerakkody, W. A R J; Fernando, W. A C; Garrido, A., "feedback free DVC architecture using machine learning," Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on, vol., no., pp.1140,1143, 12-15 Oct. 2008.

[Mar10] Marpe, D.; Schwarz, H.; Bosse, S.; Bross, B.; Helle, P.; Hinz, T.; Kirchhoffer, H.; Lakshman, H.; Tung Nguyen; Oudin, S.; Siekmann, M.; Sühring, K.; Winken, M.; Wiegand, T., "Video Compression Using Nested Quadtree Structures, Leaf Merging, and Improved Techniques for Motion Representation and Entropy Coding," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.20, no.12, pp.1676-1687, Dec. 2010.

[Neḿ10] Neḿić, O.; Rimac-Drlje, S.; Vranjes, M., "Multiview Video Coding extension of the H.264/AVC standard," in ELMAR, 2010 PROCEEDINGS , vol., no., pp.73-76, 15-17 Sept. 2010.

[Ngu12] Tung Nguyen; Marpe, D., "Performance analysis of HEVC-based intra coding for still image compression," Picture Coding Symposium (PCS), 2012 , vol., no., pp.233,236, 7-9 May 2012.

[Min08] J. Min, S. Lee, I. Kim, W.-J. Han, J. Lainema, and K. Ugur, “Unification of the Directional Intra Prediction Methods in TMuC,” JCTVC-B100, Geneva, Switzerland, Jul. 2010.

[Min10] J. Min, S. Lee, I. Kim, W.-J. Han, J. Lainema, and K. Ugur, “Unification of the Directional Intra Prediction Methods in TMuC,” JCTVC-B100, Geneva, Switzerland, Jul. 2010.

[Min11]A. Minezawa, K. Sugimoto, and S. Sekiguchi, An Improved Intra Vertical and Horizontal Prediction, JCTVC-F172, Torino, Italy, Jul. 2011.

[Mis13] Misra, K.; Segall, A.; Horowitz, M.; Shilin Xu; Fuldseth, A.; Minhua Zhou, "An Overview of Tiles in HEVC," in Selected Topics in Signal Processing, IEEE Journal of , vol.7, no.6, pp.969-977, Dec. 2013.

[Mit02] Mitra Basu, 2002. Gaussian-Based Edge-Detection Methods—A Survey. IEEE Transactions on Systems, Man, and Cybernetic, 252-260.

[Nor12] Norkin, A.; Bjontegaard, G.; Fuldseth, A.; Narroschke, M.; Ikeda, M.; Andersson, K.; Minhua Zhou; Van der Auwera, G., "HEVC Deblocking Filter," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1746-1754, Dec. 2012.

[Ohm12] Ohm, J.; Sullivan, G.J.; Schwarz, H.; Thiow Keng Tan; Wiegand, T., "Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC)," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1669-1684, Dec. 2012.

[Ohm13] Ohm, J.; Sullivan, G.J., "High efficiency video coding: the next frontier in video compression [Standards in a Nutshell]," in Signal Processing Magazine, IEEE, vol.30, no.1, pp.152-158, Jan. 2013.

[Ort99] Ortega A, Ramchandran K (1999) Rate-distortion methods for image and video compression: an overview. IEEE Signal Process J 23–50.

[Pan05] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu, “Fast mode decision algorithm for intra prediction in H.264/AVC video coding,” IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 7, pp. 813–822, Jul. 2005.

[Pia10] Y. Piao, J. H. Min, and J. Chen, Encoder Improvement of Unified Intra Prediction, JCTVC-C207, JCT-VC of ISO/IEC and ITU-T, Guangzhou, Chnia, Oct. 2010.

[Pin04] M H Pinson and S Wolf, “A new standardized method for objectively measuring video quality,” IEEE Trans. Broadcasting, vol. 50, no. 3, pp. 312–322, 2004.

[Pra01] William K. Pratt. 2001. *Digital Image Processing: PIKS Inside* (3rd ed.). John Wiley & Sons, Inc., New York, NY, USA.

[Pre70] J. M. S. Prewitt, “Object Enhancement and Extraction,” in Picture Processing and Psychopictorics, B. S. Lipkin and A. Rosenfeld, Eds., Academic Press, New York. 1970.

[Qui93] J.R. Quinlan. “C4.5: Programs for Machine Learning”. Morgan Kaufmann, 1993.

[Rao74] Ahmed, N.; Natarajan, T.; Rao, K.R., "Discrete Cosine Transform," in Computers, IEEE Transactions on , vol.C-23, no.1, pp.90-93, Jan. 1974.

[Rao76] Rao, K.R.; Ahmed, N., "Orthogonal transforms for digital signal processing," in Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '76. , vol.1, no., pp.136-140, Apr 1976.

[Rob65] L. G. Roberts, “Machine Perception of Three-Dimensional Solids,” in Optical and Electro-Optical Information Processing, J. T. Tippett et al., Eds., MIT Press, Cambridge, MA, 1965, 159–197.

[Rui14a] Ruiz, D.; Adzic, V.; Fernandez-Escribano, G.; Kalva, H.; Martinez, J.L.; Cuenca, P., "Fast partitioning algorithm for HEVC Intra frame coding using machine learning," in Image Processing (ICIP), 2014 IEEE International Conference on , vol., no., pp.4112-4116, 27-30 Oct. 2014.

[Rui14b] Damián Ruiz, Velibor Adzic, Hari Kalva, Gerardo Fernández-Escribano, “Análisis estadístico del particionado de bloques en la predicción Intra-frame de HEVC.” Jornadas de Paralelismo (JP2014), Valladolid, Spain, Sep. 2014.

[Rui15a] Ruiz-Coll1, D.; Fernandez-Escribano, G.; Martinez, J.L.; Cuenca, P., "Dominant gradient detection using Mean Directional Variance for intra-picture prediction in HEVC,” 15th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2015), Rota, Spain, Volume: 1233-1244, Ju. 2015.

[Rui15b] Damián Ruiz, Gerardo Fernández-Escribano, Velibor Adzic, Hari Kalva, José Luis Martínez, Pedro Cuenca, “Fast CU partitioning algorithm for HEVC intra coding using Data Mining.” Accepted to be published in the Multimedia Tools and Application Journal in 2016.

[Rui15c] Damián Ruiz, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, “Fast intra mode decision algorithm based on texture orientation detection in HEVC.” Submitted to the Journal of Signal Processing: Image Communication. In process of major revisions.

[Rui15d] Damián Ruiz, Gerardo Fernández-Escribano, José Luis Martínez, Pedro Cuenca, “Análisis estadístico de la eficiencia de la predicción direccional intra-cuadro en HEVC.” Jornadas de Paralelismo (JP2015), Cordoba, Spain, Sep. 2015.

[Sax11] A. Saxena and F. C. Fernandes, CE7: Mode-Dependent DCT/DST Without 4*4 Full Matrix Multiplication for Intra Prediction, JCTVCE125, Geneva, Switzerland, Mar. 2011.

[Sch07] Schwarz, H.; Marpe, D.; Wiegand, T., "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.17, no.9, pp.1103-1120, Sept. 2007.

[She13] Liquan Shen; Zhaoyang Zhang; Ping An, "Fast CU size decision and mode decision algorithm for HEVC intra coding," Consumer Electronics, IEEE Transactions on, vol.59, no.1, pp.207-213, Feb. 2013.

[Sil12] da Silva, T.L.; Agostini, L.V.; da Silva Cruz, L.A., "Fast HEVC intra prediction mode decision based on EDGE direction information," Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, vol., no., pp.1214,1218, 27-31 Aug. 2012.

[Sol12] Sole, J.; Joshi, R.; Nguyen Nguyen; Tianying Ji; Karczewicz, M.; Clare, G.; Henry, F.; Duenas, A., "Transform Coefficient Coding in HEVC," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1765-1777, Dec. 2012.

[Sto01] Stolberg, H.-J.; Berekovic, M.; Pirsch, P.; Runge, H., "Implementing the MPEG-4 advanced simple profile for streaming video applications," in Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on , vol., no., pp.230-233, 22-25 Aug. 2001.

[Sul99] Sullivan GJ, Wiegand T (1999) Rate-distortion optimization for video compression. IEEE Signal Process J 74 -90.

[Sul12] Sullivan, G.J.; Ohm, J.; Woo-Jin Han; Wiegand, T., "Overview of the High Efficiency Video Coding (HEVC) Standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1649-1668, Dec. 2012.

[Sun12] Heming Sun; Dajiang Zhou; Goto, S., "A Low-Complexity HEVC Intra Prediction Algorithm Based on Level and Mode Filtering," Multimedia and Expo (ICME), 2012 IEEE International Conference on , vol., no., pp.1085,1090, 9-13 July 2012

[Sze12] Sze, V.; Budagavi, M., "High Throughput CABAC Entropy Coding in HEVC," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.22, no.12, pp.1778-1791, Dec. 2012.

[Sze13] Sze, V.; Budagavi, M., "A comparison of CABAC throughput for HEVC/H.265 VS. AVC/H.264," Signal Processing Systems (SiPS), 2013 IEEE Workshop on , vol., no., pp.165,170, 16-18 Oct. 2013.

[Tia12] Guifen Tian; Goto, S., "Content adaptive prediction unit size decision algorithm for HEVC intra coding," Picture Coding Symposium (PCS), 2012, vol., no., pp.405,408, 7-9 May 2012.

[Tou05] Tourapis, A.M.; Feng Wu; Shipeng Li, "Direct mode coding for bipredictive slices in the H.264 standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.15, no.1, pp.119-126, Jan. 2005.

[Ugu10] K. Ugur, K. R. Andersson, and A. Fuldseth, Video Coding Technology Proposal by Tandberg, Nokia, and Ericsson, JCTVC-A119, Dresden, Germany, Apr. 2010.

[VCEG-AM91]VCEG and MPEG. Joint call for proposals on video compression technology. Doc. VCEG-AM91. ITU-T SG16/Q6 VCEG, Kyoto, JP (2010).

[VCEG-M33] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves", ITU-T SG16 Q.6 Document, VCEG-M33, Austin, US, Apr. 2001.

[Vel06] Velisavljevic, V.; Beferull-Lozano, B.; Vetterli, M.; Dragotti, P.L., "Directionlets: anisotropic multidirectional representation with separable filtering," Image Processing, IEEE Transactions on , vol.15, no.7, pp.1916,1933, July 2006.

[Was10] Wasikowski, M.; Xue-wen Chen, "Combating the Small Sample Class Imbalance Problem Using Feature Selection," Knowledge and Data Engineering, IEEE Transactions on , vol.22, no.10, pp.1388,1400, Oct. 2010.

[Wie03] Wiegand, T.; Sullivan, G.J.; Bjontegaard, G.; Luthra, A., "Overview of the H.264/AVC video coding standard," in Circuits and Systems for Video Technology, IEEE Transactions on , vol.13, no.7, pp.560-576, July 2003.

[Wie03] M. Wien, “Variable block-size transform for H.264/AVC,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 604–619, Jul. 2003.

[Wit05] Ian H. Witten and Eibe Frank. “Data Mining: Practical Machine Learning Tools and Techniques”. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[Yan12] Shunqing Yan; Liang Hong; Weifeng He; Qin Wang, "Group-Based Fast Mode Decision Algorithm for Intra Prediction in HEVC," Signal Image Technology and Internet Based Systems (SITIS), 2012 Eighth International Conference on , vol., no., pp.225,229, 25-29 Nov. 2012.

[Yao14] Yingbiao Yao;, Xiaojuan Li,Yu Lu, "Fast intra mode decision algorithm for HEVC based on dominant edge assent distribution," Multimedia Tools and Applications journal, Springer US, vol., pp. 1,19, 2014.

[Zha14] Liang Zhao, Xiaopeng Fan, Siwei Ma, Debin Zhao, “Fast intra-encoding algorithm for High Efficiency Video Coding,” Signal Processing: Image Communication, Volume 29, Issue 9, October 2014, Pages 935-944.